

```

1 ; *****
2 ; UNIX386.ASM (RETRO UNIX 386 Kernel) - v0.2.0.16
3 ; -----
4 ; NASM version 2.11 (unix386.s)
5 ;
6 ; RETRO UNIX 386 (Retro Unix == Turkish Rational Unix)
7 ; Operating System Project (v0.2) by ERDOGAN TAN (Beginning: 24/12/2013)
8 ;
9 ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
10 ; (v0.1 - Beginning: 11/07/2012)
11 ;
12 ; [ Last Modification: 09/12/2015 ]
13 ;
14 ; Derived from UNIX Operating System (v1.0 for PDP-11)
15 ; (Original) Source Code by Ken Thompson (1971-1972)
16 ; <Bell Laboratories (17/3/1972)>
17 ; <Preliminary Release of UNIX Implementation Document>
18 ;
19 ; Derived from 'UNIX v7/x86' source code by Robert Nordier (1999)
20 ; UNIX V7/x86 source code: see www.nordier.com/v7x86 for details.
21 ;
22 ; *****
23
24 ; 24/12/2013
25
26 ; Entering protected mode:
27 ; Derived from 'simple_asm.txt' source code file and
28 ; 'The world of Protected mode' tutorial/article by Gregor Brunmar (2003)
29 ; (gregor.brunmar@home.se)
30 ; http://www.osdever.net/tutorials/view/the-world-of-protected-mode
31 ;
32
33 ; "The Real, Protected, Long mode assembly tutorial for PCs"
34 ; by Michael Chourdakis (2009)
35 ; http://www.codeproject.com/Articles/45788/
36 ; http://www.michaelchourdakis.com
37 ;
38
39 ; Global Descriptor Table:
40 ; Derived from 'head.s" source code of Linux v1.0 kernel
41 ; by Linus Torvalds (1991-1992)
42 ;
43
44 KLOAD equ 10000h ; Kernel loading address
45 ; NOTE: Retro UNIX 8086 v1 /boot code loads kernel at 1000h:0000h
46 KCODE equ 08h ; Code segment descriptor (ring 0)
47 KDATA equ 10h ; Data segment descriptor (ring 0)
48 ; 19/03/2015
49 UCODE equ 1Bh ; 18h + 3h (ring 3)
50 UDATA equ 23h ; 20h + 3h (ring 3)
51 ; 24/03/2015
52 TSS equ 28h ; Task state segment descriptor (ring 0)
53 ; 19/03/2015
54 CORE equ 400000h ; Start of USER's virtual/linear address space
55 ; (at the end of the 1st 4MB)
56 ECORE equ 0FFC0000h ; End of USER's virtual address space (4GB - 4MB)
57 ; ULIMIT = (ECORE/4096) - 1 = 0FFBFFh (in GDT)
58
59 ; 27/12/2013
60 KEND equ KLOAD + 65536 ; (28/12/2013) (end of kernel space)
61
62 ; IBM PC/AT BIOS ----- 10/06/85 (postequ.inc)
63 ;----- CMOS TABLE LOCATION ADDRESS'S -----
64 CMOS_SECONDS EQU 00H ; SECONDS (BCD)
65 CMOS_MINUTES EQU 02H ; MINUTES (BCD)
66 CMOS_HOURS EQU 04H ; HOURS (BCD)
67 CMOS_DAY_WEEK EQU 06H ; DAY OF THE WEEK (BCD)
68 CMOS_DAY_MONTH EQU 07H ; DAY OF THE MONTH (BCD)
69 CMOS_MONTH EQU 08H ; MONTH (BCD)
70 CMOS_YEAR EQU 09H ; YEAR (TWO DIGITS) (BCD)
71 CMOS_CENTURY EQU 32H ; DATE CENTURY BYTE (BCD)
72 CMOS_REG_A EQU 0AH ; STATUS REGISTER A
73 CMOS_REG_B EQU 00BH ; STATUS REGISTER B ALARM
74 CMOS_REG_C EQU 00CH ; STATUS REGISTER C FLAGS
75 CMOS_REG_D EQU 0DH ; STATUS REGISTER D BATTERY
76 CMOS_SHUT_DOWN EQU 0FH ; SHUTDOWN STATUS COMMAND BYTE
77 ;-----
78 ; CMOS EQUATES FOR THIS SYSTEM ;
79 ;-----
80 CMOS_PORT EQU 070H ; I/O ADDRESS OF CMOS ADDRESS PORT
81 CMOS_DATA EQU 071H ; I/O ADDRESS OF CMOS DATA PORT
82 NMI EQU 10000000B ; DISABLE NMI INTERRUPTS MASK -
83 ; HIGH BIT OF CMOS LOCATION ADDRESS
84
85 ; Memory Allocation Table Address
86 ; 05/11/2014
87 ; 31/10/2014
88 MEM_ALLOC_TBLequ 100000h ; Memory Allocation Table at the end of
89 ; the 1st 1 MB memory space.
90 ; (This address must be aligned
91 ; on 128 KB boundary, if it will be
92 ; changed later.)

```

```

93                                     ; ((lower 17 bits of 32 bit M.A.T.
94                                     ;   address must be ZERO)).
95                                     ; (((Reason: 32 bit allocation
96                                     ;   instructions, dword steps)))
97                                     ; ((byte >> 12 --> page >> 5)))
98 ;04/11/2014
99 PDE_A_PRESENT equ 1                ; Present flag for PDE
100 PDE_A_WRITE   equ 2                ; Writable (write permission) flag
101 PDE_A_USER    equ 4                ; User (non-system/kernel) page flag
102 ;
103 PTE_A_PRESENT equ 1                ; Present flag for PTE (bit 0)
104 PTE_A_WRITE   equ 2                ; Writable (write permission) flag (bit 1)
105 PTE_A_USER    equ 4                ; User (non-system/kernel) page flag (bit 2)
106 PTE_A_ACCESS  equ 32               ; Accessed flag (bit 5) ; 09/03/2015
107
108 ; 17/02/2015 (unix386.s)
109 ; 10/12/2014 - 30/12/2014 (0B000h -> 9000h) (dsectrm2.s)
110 DPT_SEGM equ 09000h ; FDPT segment (EDD v1.1, EDD v3)
111 ;
112 HD0_DPT equ 0                    ; Disk parameter table address for hd0
113 HD1_DPT equ 32                   ; Disk parameter table address for hd1
114 HD2_DPT equ 64                   ; Disk parameter table address for hd2
115 HD3_DPT equ 96                   ; Disk parameter table address for hd3
116
117
118 ; FDPT (Phoenix, Enhanced Disk Drive Specification v1.1, v3.0)
119 ;   (HDPT: Programmer's Guide to the AMIBIOS, 1993)
120 ;
121 FDPT_CYLS     equ 0 ; 1 word, number of cylinders
122 FDPT_HDS      equ 2 ; 1 byte, number of heads
123 FDPT_TT       equ 3 ; 1 byte, A0h = translated FDPT with logical values
124               ; otherwise it is standard FDPT with physical values
125 FDPT_PCOMP    equ 5 ; 1 word, starting write precompensation cylinder
126               ; (obsolete for IDE/ATA drives)
127 FDPT_CB       equ 8 ; 1 byte, drive control byte
128               ; Bits 7-6 : Enable or disable retries (00h = enable)
129               ; Bit 5      : 1 = Defect map is located at last cyl. + 1
130               ; Bit 4      : Reserved. Always 0
131               ; Bit 3      : Set to 1 if more than 8 heads
132               ; Bit 2-0    : Reserved. Always 0
133 FDPT_LZ       equ 12 ; 1 word, landing zone (obsolete for IDE/ATA drives)
134 FDPT_SPT      equ 14 ; 1 byte, sectors per track
135
136 ; Floppy Drive Parameters Table (Programmer's Guide to the AMIBIOS, 1993)
137 ; (11 bytes long) will be used by diskette handler/bios
138 ; which is derived from IBM PC-AT BIOS (DISKETTE.ASM, 21/04/1986).
139
140 [BITS 16]                ; We need 16-bit instructions for Real mode
141
142 [ORG 0]
143 ; 12/11/2014
144 ; Save boot drive number (that is default root drive)
145 00000000 8816[D46D]     mov [boot_drv], dl ; physical drv number
146
147 ; Determine installed memory
148 ; 31/10/2014
149 ;
150 00000004 B801E8         mov ax, 0E801h ; Get memory size
151 00000007 CD15           int 15h        ; for large configurations
152 00000009 7308           jnc short chk_ms
153 0000000B B488           mov ah, 88h    ; Get extended memory size
154 0000000D CD15           int 15h
155 ;
156 ;mov al, 17h            ; Extended memory (1K blocks) low byte
157 ;out 70h, al ; select CMOS register
158 ;in al, 71h ; read data (1 byte)
159 ;mov cl, al
160 ;mov al, 18h ; Extended memory (1K blocks) high byte
161 ;out 70h, al ; select CMOS register
162 ;in al, 71h ; read data (1 byte)
163 ;mov ch, al
164 ;
165 0000000F 89C1           mov cx, ax
166 00000011 31D2           xor dx, dx
167 chk_ms:
168 00000013 890E[D06D]     mov [mem_1m_1k], cx
169 00000017 8916[D26D]     mov [mem_16m_64k], dx
170 ; 05/11/2014
171 ;and dx, dx
172 ;jz short L2
173 0000001B 81F90004       cmp cx, 1024
174 0000001F 7315           jnb short L0
175 ; insufficient memory_error
176 ; Minimum 2 MB memory is needed...
177 ; 05/11/2014
178 ; (real mode error printing)
179 00000021 FB            sti
180 00000022 BE[496C]       mov si, msg_out_of_memory
181 00000025 BB0700         mov bx, 7
182 00000028 B40E           mov ah, 0Eh    ; write tty
183 oom_1:
184 0000002A AC            lodsb

```

```

185 0000002B 08C0      or     al, al
186 0000002D 7404      jz      short oom_2
187 0000002F CD10      int     10h
188 00000031 EBF7      jmp     short oom_1
189                      oom_2:
190 00000033 F4          hlt
191 00000034 EBF0      jmp     short oom_2
192
193                      L0:
194                      %include 'diskinit.inc' ; 07/03/2015
195 <1> ; Retro UNIX 386 v1 Kernel - DISKINIT.INC
196 <1> ; Last Modification: 10/07/2015
197 <1>
198 <1> ; DISK I/O SYSTEM INITIALIZATION - Erdogan Tan (Retro UNIX 386 v1 project)
199 <1>
200 <1> ; //////////// DISK I/O SYSTEM STRUCTURE INITIALIZATION //////////////////////
201 <1>
202 <1> ; 10/12/2014 - 02/02/2015 - dsectrm2.s
203 <1> ;L0:
204 <1> ; 12/11/2014 (Retro UNIX 386 v1 - beginning)
205 <1> ; Detecting disk drives... (by help of ROM-BIOS)
206 00000036 BA7F00    <1> mov     dx, 7Fh
207 <1> L1:
208 00000039 FEC2    <1> inc     dl
209 0000003B B441    <1> mov     ah, 41h ; Check extensions present
210 <1> ; Phoenix EDD v1.1 - EDD v3
211 0000003D BBAA55    <1> mov     bx, 55AAh
212 00000040 CD13    <1> int     13h
213 00000042 721A    <1> jc      short L2
214 00000044 81FB55AA  <1> cmp     bx, 0AA55h
215 00000048 7514    <1> jne     short L2
216 0000004A FE06[D76D] <1> inc     byte [hdc] ; count of hard disks (EDD present)
217 0000004E 8816[D66D] <1> mov     [last_drv], dl ; last hard disk number
218 00000052 BB[5A6D] <1> mov     bx, hd0_type - 80h
219 00000055 01D3    <1> add     bx, dx
220 00000057 880F    <1> mov     [bx], cl ; Interface support bit map in CX
221 <1> ; Bit 0 - 1, Fixed disk access subset ready
222 <1> ; Bit 1 - 1, Drv locking and ejecting ready
223 <1> ; Bit 2 - 1, Enhanced Disk Drive Support
224 <1> ; (EDD) ready (DPTE ready)
225 <1> ; Bit 3 - 1, 64bit extensions are present
226 <1> ; (EDD-3)
227 <1> ; Bit 4 to 15 - 0, Reserved
228 00000059 80FA83    <1> cmp     dl, 83h ; drive number < 83h
229 0000005C 72DB    <1> jnb     short L1
230 <1> L2:
231 <1> ; 23/11/2014
232 <1> ; 19/11/2014
233 0000005E 30D2    <1> xor     dl, dl ; 0
234 00000060 66BE[D86D0000] <1> mov     esi, fd0_type
235 <1> L3:
236 <1> ; 14/01/2015
237 00000066 8816[D56D] <1> mov     [drv], dl
238 <1> ;
239 0000006A B408    <1> mov     ah, 08h ; Return drive parameters
240 0000006C CD13    <1> int     13h
241 0000006E 7215    <1> jc      short L4
242 <1> ; BL = drive type (for floppy drives)
243 <1> ; DL = number of floppy drives
244 <1> ;
245 <1> ; ES:DI = Address of DPT from BIOS
246 <1> ;
247 00000070 67881E    <1> mov     [esi], bl ; Drive type
248 <1> ; 4 = 1.44 MB, 80 track, 3 1/2"
249 <1> ; 14/01/2015
250 00000073 E8BE02    <1> call    set_disk_parms
251 <1> ; 10/12/2014
252 00000076 6681FE[D86D0000] <1> cmp     esi, fd0_type
253 0000007D 7706    <1> ja      short L4
254 0000007F 6646    <1> inc     esi ; fd1_type
255 00000081 B201    <1> mov     dl, 1
256 00000083 EBE1    <1> jmp     short L3
257 <1> L4:
258 <1> ; Older BIOS (INT 13h, AH = 48h is not available)
259 00000085 B27F    <1> mov     dl, 7Fh
260 <1> ; 24/12/2014 (Temporary)
261 00000087 803E[D76D]00 <1> cmp     byte [hdc], 0 ; EDD present or not ?
262 0000008C 0F879000    <1> ja      L10 ; yes, all fixed disk operations
263 <1> ; will be performed according to
264 <1> ; present EDD specification
265 <1> L6:
266 00000090 FEC2    <1> inc     dl
267 00000092 8816[D56D] <1> mov     [drv], dl
268 00000096 8816[D66D] <1> mov     [last_drv], dl ; 14/01/2015
269 0000009A B408    <1> mov     ah, 08h ; Return drive parameters
270 0000009C CD13    <1> int     13h ; (conventional function)
271 0000009E 0F829901    <1> jc      L13 ; fixed disk drive not ready
272 000000A2 8816[D76D] <1> mov     [hdc], dl ; number of drives
273 <1> ;; 14/01/2013
274 <1> ;;push     cx
275 000000A6 E88B02    <1> call    set_disk_parms
276 <1> ;;pop     cx

```

```

277
278
279 000000A9 8A16[D56D]
280 000000AD BB0401
281 000000B0 80FA80
282 000000B3 7603
283 000000B5 83C314
284
285 000000B8 31C0
286 000000BA 8ED8
287 000000BC 8B37
288 000000BE 8B4702
289 000000C1 8ED8
290 000000C3 3A4C0E
291 000000C6 0F856D01
292 000000CA BF0000
293 000000CD 80FA80
294 000000D0 7603
295 000000D2 BF2000
296
297
298 000000D5 B80090
299 000000D8 8EC0
300
301 000000DA B90800
302 000000DD F3A5
303 000000DF 8CC8
304 000000E1 8ED8
305
306 000000E3 8A0E[D56D]
307 000000E7 88CB
308 000000E9 B8F001
309 000000EC 80E301
310 000000EF 7406
311 000000F1 C0E304
312 000000F4 2D8000
313
314 000000F7 AB
315 000000F8 050602
316 000000FB AB
317 000000FC 88D8
318 000000FE 04A0
319 00000100 AA
320
321 00000101 FE06[D56D]
322 00000105 BB[5A6D]
323 00000108 01CB
324 0000010A 800F80
325 0000010D A0[D76D]
326 00000110 FEC8
327 00000112 0F842501
328 00000116 80FA80
329 00000119 0F8673FF
330 0000011D E91B01
331
332 00000120 FEC2
333
334 00000122 8816[D56D]
335 00000126 B408
336 00000128 CD13
337 0000012A 0F820D01
338
339 0000012E 8A16[D56D]
340 00000132 52
341 00000133 51
342 00000134 E8FD01
343 00000137 59
344 00000138 5A
345
346 00000139 66BE[54810000]
347
348 0000013F 67C7061E00
349 00000144 B448
350 00000146 CD13
351 00000148 0F82EF00
352
353 0000014C 6629DB
354 0000014F 88D3
355 00000151 80EB80
356 00000154 6681C3[DA6D0000]
357 0000015B 678A03
358 0000015E 0C80
359 00000160 678803
360 00000163 6681EB[D86D0000]
361 0000016A 6681C3[246E0000]
362 00000171 678803
363 00000174 66678B4610
364
365 00000179 6621C0
366 0000017C 7416
367
368

<1> ;
<1> ;;and cl, 3Fh ; sectors per track (bits 0-6)
<1> mov dl, [drv]
<1> mov bx, 65*4 ; hd0 parameters table (INT 41h)
<1> cmp dl, 80h
<1> jna short L7
<1> add bx, 5*4 ; hdl parameters table (INT 46h)
<1> L7:
<1> xor ax, ax
<1> mov ds, ax
<1> mov si, [bx]
<1> mov ax, [bx+2]
<1> mov ds, ax
<1> cmp cl, [si+FDPT_SPT] ; sectors per track
<1> jne L12 ; invalid FDPT
<1> mov di, HD0_DPT
<1> cmp dl, 80h
<1> jna short L8
<1> mov di, HD1_DPT
<1> L8:
<1> ; 30/12/2014
<1> mov ax, DPT_SEGM
<1> mov es, ax
<1> ; 24/12/2014
<1> mov cx, 8
<1> rep movsw ; copy 16 bytes to the kernel's DPT location
<1> mov ax, cs
<1> mov ds, ax
<1> ; 02/02/2015
<1> mov cl, [drv]
<1> mov bl, cl
<1> mov ax, 1F0h
<1> and bl, 1
<1> jz short L9
<1> shl bl, 4
<1> sub ax, 1F0h-170h
<1> L9:
<1> stosw ; I/O PORT Base Address (1F0h, 170h)
<1> add ax, 206h
<1> stosw ; CONTROL PORT Address (3F6h, 376h)
<1> mov al, bl
<1> add al, 0A0h
<1> stosb ; Device/Head Register upper nibble
<1> ;
<1> inc byte [drv]
<1> mov bx, hd0_type - 80h
<1> add bx, cx
<1> or byte [bx], 80h ; present sign (when lower nibble is 0)
<1> mov al, [hdc]
<1> dec al
<1> jz L13
<1> cmp dl, 80h
<1> jna L6
<1> jmp L13
<1> L10:
<1> inc dl
<1> ; 25/12/2014
<1> mov [drv], dl
<1> mov ah, 08h ; Return drive parameters
<1> int 13h ; (conventional function)
<1> jc L13
<1> ; 14/01/2015
<1> mov dl, [drv]
<1> push dx
<1> push cx
<1> call set_disk_parms
<1> pop cx
<1> pop dx
<1> ;
<1> mov esi, _end ; 30 byte temporary buffer address
<1> ; at the '_end' of kernel.
<1> mov word [esi], 30
<1> mov ah, 48h ; Get drive parameters (EDD function)
<1> int 13h
<1> jc L13
<1> ; 14/01/2015
<1> sub ebx, ebx
<1> mov bl, dl
<1> sub bl, 80h
<1> add ebx, hd0_type
<1> mov al, [ebx]
<1> or al, 80h
<1> mov [ebx], al
<1> sub ebx, hd0_type - 2 ; 15/01/2015
<1> add ebx, drv.status
<1> mov [ebx], al
<1> mov eax, [esi+16]
<1> ; 28/02/2015
<1> and eax, eax
<1> jz short L10_A0h
<1> ; 'CHS only' disks on EDD system
<1> ; are reported with ZERO disk size

```

369	0000017E	6681EB[246E0000]	<1>	sub	ebx, drv.status
370	00000185	66C1E302	<1>	shl	ebx, 2
371	00000189	6681C3[086E0000]	<1>	add	ebx, drv.size ; disk size (in sectors)
372	00000190	66678903	<1>	mov	[ebx], eax
373			<1>	L10_A0h:	; Jump here to fix a ZERO (LBA) disk size problem
374			<1>		; for CHS disks (28/02/2015)
375			<1>		; 30/12/2014
376	00000194	BF0000	<1>	mov	di, HD0_DPT
377	00000197	88D0	<1>	mov	al, dl
378	00000199	83E003	<1>	and	ax, 3
379	0000019C	C0E005	<1>	shl	al, 5 ; *32
380	0000019F	01C7	<1>	add	di, ax
381	000001A1	B80090	<1>	mov	ax, DPT_SEGM
382	000001A4	8EC0	<1>	mov	es, ax
383			<1>		
384	000001A6	88E8	<1>	mov	al, ch ; max. cylinder number (bits 0-7)
385	000001A8	88CC	<1>	mov	ah, cl
386	000001AA	C0EC06	<1>	shr	ah, 6 ; max. cylinder number (bits 8-9)
387	000001AD	40	<1>	inc	ax ; logical cylinders (limit 1024)
388	000001AE	AB	<1>	stosw	
389	000001AF	88F0	<1>	mov	al, dh ; max. head number
390	000001B1	FEC0	<1>	inc	al
391	000001B3	AA	<1>	stosb	; logical heads (limits 256)
392	000001B4	B0A0	<1>	mov	al, 0A0h ; Indicates translated table
393	000001B6	AA	<1>	stosb	
394	000001B7	8A440C	<1>	mov	al, [si+12]
395	000001BA	AA	<1>	stosb	; physical sectors per track
396	000001BB	31C0	<1>	xor	ax, ax
397			<1>		;dec ax ; 02/01/2015
398	000001BD	AB	<1>	stosw	; precompensation (obsolete)
399			<1>		;xor al, al ; 02/01/2015
400	000001BE	AA	<1>	stosb	; reserved
401	000001BF	B008	<1>	mov	al, 8 ; drive control byte
402			<1>		; (do not disable retries,
403			<1>		; more than 8 heads)
404	000001C1	AA	<1>	stosb	
405	000001C2	8B4404	<1>	mov	ax, [si+4]
406	000001C5	AB	<1>	stosw	; physical number of cylinders
407			<1>		;push ax ; 02/01/2015
408	000001C6	8A4408	<1>	mov	al, [si+8]
409	000001C9	AA	<1>	stosb	; physical num. of heads (limit 16)
410	000001CA	29C0	<1>	sub	ax, ax
411			<1>		;pop ax ; 02/01/2015
412	000001CC	AB	<1>	stosw	; landing zone (obsolete)
413	000001CD	88C8	<1>	mov	al, cl ; logical sectors per track (limit 63)
414	000001CF	243F	<1>	and	al, 3Fh
415	000001D1	AA	<1>	stosb	
416			<1>		;sub al, al ; checksum
417			<1>		;stosb
418			<1>		
419	000001D2	83C61A	<1>	add	si, 26 ; (BIOS) DPTE address pointer
420	000001D5	AD	<1>	lodsw	
421	000001D6	50	<1>	push	ax ; (BIOS) DPTE offset
422	000001D7	AD	<1>	lodsw	
423	000001D8	50	<1>	push	ax ; (BIOS) DPTE segment
424			<1>		
425			<1>		; checksum calculation
426	000001D9	89FE	<1>	mov	si, di
427	000001DB	06	<1>	push	es
428	000001DC	1F	<1>	pop	ds
429			<1>		;mov cx, 16
430	000001DD	B90F00	<1>	mov	cx, 15
431	000001E0	29CE	<1>	sub	si, cx
432	000001E2	30E4	<1>	xor	ah, ah
433			<1>		;del cl
434			<1>	L11:	
435	000001E4	AC	<1>	lodsb	
436	000001E5	00C4	<1>	add	ah, al
437	000001E7	E2FB	<1>	loop	L11
438			<1>		
439	000001E9	88E0	<1>	mov	al, ah
440	000001EB	F6D8	<1>	neg	al ; -x+x = 0
441	000001ED	AA	<1>	stosb	; put checksum in byte 15 of the tbl
442			<1>		
443	000001EE	1F	<1>	pop	ds ; (BIOS) DPTE segment
444	000001EF	5E	<1>	pop	si ; (BIOS) DPTE offset
445			<1>		
446			<1>		; 23/02/2015
447	000001F0	57	<1>	push	di
448			<1>		; ES:DI points to DPTE (FDPTE) location
449			<1>		;mov cx, 8
450	000001F1	B108	<1>	mov	cl, 8
451	000001F3	F3A5	<1>	rep	movsw
452			<1>		
453			<1>		; 23/02/2015
454			<1>		; (P)ATA drive and LBA validation
455			<1>		; (invalidating SATA drives and setting
456			<1>		; CHS type I/O for old type fixed disks)
457	000001F5	5B	<1>	pop	bx
458	000001F6	8CC8	<1>	mov	ax, cs
459	000001F8	8ED8	<1>	mov	ds, ax
460	000001FA	268B07	<1>	mov	ax, [es:bx]

```

461 000001FD 3DF001      <1>  cmp    ax, 1F0h
462 00000200 7418        <1>  je     short L11a
463 00000202 3D7001      <1>  cmp    ax, 170h
464 00000205 7413        <1>  je     short L11a
465                      <1>  ; invalidation
466                      <1>  ; (because base port address is not 1F0h or 170h)
467 00000207 30FF        <1>  xor     bh, bh
468 00000209 88D3        <1>  mov     bl, dl
469 0000020B 80EB80      <1>  sub     bl, 80h
470 0000020E C687[DA6D]00 <1>  mov     byte [bx+hd0_type], 0 ; not a valid disk drive !
471 00000213 808F[266E]F0 <1>  or      byte [bx+drv.status+2], 0F0h ; (failure sign)
472 00000218 EB14        <1>  jmp     short L11b
473                      <1>  L11a:
474                      <1>  ; LBA validation
475 0000021A 268A4704      <1>  mov     al, [es:bx+4] ; Head register upper nibble
476 0000021E A840        <1>  test    al, 40h ; LBA bit (bit 6)
477 00000220 750C        <1>  jnz     short L11b ; LBA type I/O is OK! (E0h or F0h)
478                      <1>  ; force CHS type I/O for this drive (A0h or B0h)
479 00000222 28FF        <1>  sub     bh, bh
480 00000224 88D3        <1>  mov     bl, dl
481 00000226 80EB80      <1>  sub     bl, 80h ; 26/02/2015
482 00000229 80A7[266E]FE <1>  and     byte [bx+drv.status+2], 0FEh ; clear bit 0
483                      <1>  ; bit 0 = LBA ready bit
484                      <1>  ; 'diskio' procedure will check this bit !
485                      <1>  L11b:
486 0000022E 3A16[D66D]    <1>  cmp     dl, [last_drv] ; 25/12/2014
487 00000232 7307        <1>  jnb     short L13
488 00000234 E9E9FE        <1>  jmp     L10
489                      <1>  L12:
490                      <1>  ; Restore data registers
491 00000237 8CC8        <1>  mov     ax, cs
492 00000239 8ED8        <1>  mov     ds, ax
493                      <1>  L13:
494                      <1>  ; 13/12/2014
495 0000023B 0E          <1>  push    cs
496 0000023C 07          <1>  pop     es
497                      <1>  L14:
498 0000023D B411        <1>  mov     ah, 11h
499 0000023F CD16        <1>  int     16h
500 00000241 7406        <1>  jz      short L15 ; no keys in keyboard buffer
501 00000243 B010        <1>  mov     al, 10h
502 00000245 CD16        <1>  int     16h
503 00000247 EBF4        <1>  jmp     short L14
504                      <1>  L15:
505                      <1>  ; //////////
506                      <1>  ; 24/11/2014
507                      <1>  ; 19/11/2014
508                      <1>  ; 14/11/2014
509                      <1>  ; Temporary code for disk searching code check
510                      <1>  ;
511                      <1>  ; This code will show existing (usable) drives and also
512                      <1>  ; will show EDD interface support status for hard disks
513                      <1>  ; (If status bit 7 is 1, Identify Device info is ready,
514                      <1>  ; no need to get it again in protected mode...)
515                      <1>  ;
516                      <1>  ; 13/11/2014
517 00000249 BB0700      <1>  mov     bx, 7
518 0000024C B40E        <1>  mov     ah, 0Eh
519 0000024E A0[D86D]    <1>  mov     al, [fd0_type]
520 00000251 20C0        <1>  and     al, al
521 00000253 743D        <1>  jz      short L15a
522 00000255 88C2        <1>  mov     dl, al
523 00000257 B046        <1>  mov     al, 'F'
524 00000259 CD10        <1>  int     10h
525 0000025B B044        <1>  mov     al, 'D'
526 0000025D CD10        <1>  int     10h
527 0000025F B030        <1>  mov     al, '0'
528 00000261 CD10        <1>  int     10h
529 00000263 B020        <1>  mov     al, ' '
530 00000265 CD10        <1>  int     10h
531 00000267 E8B300      <1>  call    L15c
532 0000026A B020        <1>  mov     al, ' '
533 0000026C CD10        <1>  int     10h
534                      <1>  ;
535 0000026E A0[D96D]    <1>  mov     al, [fd1_type]
536 00000271 20C0        <1>  and     al, al
537 00000273 741D        <1>  jz      short L15a
538 00000275 88C2        <1>  mov     dl, al
539 00000277 B046        <1>  mov     al, 'F'
540 00000279 CD10        <1>  int     10h
541 0000027B B044        <1>  mov     al, 'D'
542 0000027D CD10        <1>  int     10h
543 0000027F B031        <1>  mov     al, '1'
544 00000281 CD10        <1>  int     10h
545 00000283 B020        <1>  mov     al, ' '
546 00000285 CD10        <1>  int     10h
547 00000287 E89300      <1>  call    L15c
548 0000028A B020        <1>  mov     al, ' '
549 0000028C CD10        <1>  int     10h
550 0000028E B020        <1>  mov     al, ' '
551 00000290 CD10        <1>  int     10h
552                      <1>  L15a:

```

553	00000292	A0[DA6D]	<1>	mov	al, [hd0_type]
554	00000295	20C0	<1>	and	al, al
555	00000297	7479	<1>	jz	short L15b
556	00000299	88C2	<1>	mov	dl, al
557	0000029B	B048	<1>	mov	al, 'H'
558	0000029D	CD10	<1>	int	10h
559	0000029F	B044	<1>	mov	al, 'D'
560	000002A1	CD10	<1>	int	10h
561	000002A3	B030	<1>	mov	al, '0'
562	000002A5	CD10	<1>	int	10h
563	000002A7	B020	<1>	mov	al, ' '
564	000002A9	CD10	<1>	int	10h
565	000002AB	E86F00	<1>	call	L15c
566	000002AE	B020	<1>	mov	al, ' '
567	000002B0	CD10	<1>	int	10h
568			<1>	;	
569	000002B2	A0[DB6D]	<1>	mov	al, [hd1_type]
570	000002B5	20C0	<1>	and	al, al
571	000002B7	7459	<1>	jz	short L15b
572	000002B9	88C2	<1>	mov	dl, al
573	000002BB	B048	<1>	mov	al, 'H'
574	000002BD	CD10	<1>	int	10h
575	000002BF	B044	<1>	mov	al, 'D'
576	000002C1	CD10	<1>	int	10h
577	000002C3	B031	<1>	mov	al, '1'
578	000002C5	CD10	<1>	int	10h
579	000002C7	B020	<1>	mov	al, ' '
580	000002C9	CD10	<1>	int	10h
581	000002CB	E84F00	<1>	call	L15c
582	000002CE	B020	<1>	mov	al, ' '
583	000002D0	CD10	<1>	int	10h
584			<1>	;	
585	000002D2	A0[DC6D]	<1>	mov	al, [hd2_type]
586	000002D5	20C0	<1>	and	al, al
587	000002D7	7439	<1>	jz	short L15b
588	000002D9	88C2	<1>	mov	dl, al
589	000002DB	B048	<1>	mov	al, 'H'
590	000002DD	CD10	<1>	int	10h
591	000002DF	B044	<1>	mov	al, 'D'
592	000002E1	CD10	<1>	int	10h
593	000002E3	B032	<1>	mov	al, '2'
594	000002E5	CD10	<1>	int	10h
595	000002E7	B020	<1>	mov	al, ' '
596	000002E9	CD10	<1>	int	10h
597	000002EB	E82F00	<1>	call	L15c
598	000002EE	B020	<1>	mov	al, ' '
599	000002F0	CD10	<1>	int	10h
600			<1>	;	
601	000002F2	A0[DD6D]	<1>	mov	al, [hd3_type]
602	000002F5	20C0	<1>	and	al, al
603	000002F7	7419	<1>	jz	short L15b
604	000002F9	88C2	<1>	mov	dl, al
605	000002FB	B048	<1>	mov	al, 'H'
606	000002FD	CD10	<1>	int	10h
607	000002FF	B044	<1>	mov	al, 'D'
608	00000301	CD10	<1>	int	10h
609	00000303	B033	<1>	mov	al, '3'
610	00000305	CD10	<1>	int	10h
611	00000307	B020	<1>	mov	al, ' '
612	00000309	CD10	<1>	int	10h
613	0000030B	E80F00	<1>	call	L15c
614	0000030E	B020	<1>	mov	al, ' '
615	00000310	CD10	<1>	int	10h
616			<1>	;	
617			<1>	L15b:	
618	00000312	B00D	<1>	mov	al, 0Dh
619	00000314	CD10	<1>	int	10h
620	00000316	B00A	<1>	mov	al, 0Ah
621	00000318	CD10	<1>	int	10h
622			<1>	;;xor	ah, ah
623			<1>	;;int	16h
624			<1>	;	
625	0000031A	E99A00	<1>	jmp	L16 ; jmp short L16
626			<1>	;	
627			<1>	L15c:	
628	0000031D	88D6	<1>	mov	dh, dl
629	0000031F	C0EE04	<1>	shr	dh, 4
630	00000322	80C630	<1>	add	dh, 30h
631	00000325	80E20F	<1>	and	dl, 15
632	00000328	80C230	<1>	add	dl, 30h
633	0000032B	88F0	<1>	mov	al, dh
634	0000032D	CD10	<1>	int	10h
635	0000032F	88D0	<1>	mov	al, dl
636	00000331	CD10	<1>	int	10h
637	00000333	C3	<1>	retn	
638			<1>	;	
639			<1>	;	end of temporary code for disk searching code check
640			<1>		
641			<1>	;	////////
642			<1>		
643			<1>	set_disk_parms:	
644			<1>	;	10/07/2015

```

645      <1> ; 14/01/2015
646      <1> ;push ebx
647 00000334 6629DB      <1> sub ebx, ebx
648 00000337 8A1E[D56D]  <1> mov bl, [drv]
649 0000033B 80FB80      <1> cmp bl, 80h
650 0000033E 7203      <1> jnb short sdp0
651 00000340 80EB7E      <1> sub bl, 7Eh
652      <1> sdp0:
653 00000343 6681C3[246E0000] <1> add ebx, drv.status
654 0000034A 67C60380      <1> mov byte [ebx], 80h ; 'Present' flag
655      <1> ;
656 0000034E 88E8      <1> mov al, ch ; last cylinder (bits 0-7)
657 00000350 88CC      <1> mov ah, cl ;
658 00000352 C0EC06      <1> shr ah, 6 ; last cylinder (bits 8-9)
659 00000355 6681EB[246E0000] <1> sub ebx, drv.status
660 0000035C D0E3      <1> shl bl, 1
661 0000035E 6681C3[DE6D0000] <1> add ebx, drv.cylinders
662 00000365 40      <1> inc ax ; convert max. cyl number to cyl count
663 00000366 678903      <1> mov [ebx], ax
664 00000369 50      <1> push ax ; ** cylinders
665 0000036A 6681EB[DE6D0000] <1> sub ebx, drv.cylinders
666 00000371 6681C3[EC6D0000] <1> add ebx, drv.heads
667 00000378 30E4      <1> xor ah, ah
668 0000037A 88F0      <1> mov al, dh ; heads
669 0000037C 40      <1> inc ax
670 0000037D 678903      <1> mov [ebx], ax
671 00000380 6681EB[EC6D0000] <1> sub ebx, drv.heads
672 00000387 6681C3[FA6D0000] <1> add ebx, drv.spt
673 0000038E 30ED      <1> xor ch, ch
674 00000390 80E13F      <1> and cl, 3Fh ; sectors (bits 0-6)
675 00000393 67890B      <1> mov [ebx], cx
676 00000396 6681EB[FA6D0000] <1> sub ebx, drv.spt
677 0000039D 66D1E3      <1> shl ebx, 1
678 000003A0 6681C3[086E0000] <1> add ebx, drv.size ; disk size (in sectors)
679      <1> ; LBA size = cylinders * heads * secpertrack
680 000003A7 F7E1      <1> mul cx
681 000003A9 89C2      <1> mov dx, ax ; heads*spt
682 000003AB 58      <1> pop ax ; ** cylinders
683 000003AC 48      <1> dec ax ; 1 cylinder reserved (!?)
684 000003AD F7E2      <1> mul dx ; cylinders * (heads*spt)
685 000003AF 678903      <1> mov [ebx], ax
686 000003B2 67895302      <1> mov [ebx+2], dx
687      <1> ;
688      <1> ;pop ebx
689 000003B6 C3      <1> retn
690      <1>
691      <1> ;align 2
692      <1>
693      <1> ;cylinders : dw 0, 0, 0, 0, 0, 0
694      <1> ;heads : dw 0, 0, 0, 0, 0, 0
695      <1> ;spt : dw 0, 0, 0, 0, 0, 0
696      <1> ;disk_size : dd 0, 0, 0, 0, 0, 0
697      <1>
698      <1> ;last_drv:
699      <1> ;db 0
700      <1> ;drv_status:
701      <1> ;db 0,0,0,0,0,0
702      <1> ;db 0
703      <1>
704      <1>
705      <1> ; End Of DISK I/O SYSTEM STRUCTURE INITIALIZATION /// 06/02/2015
706      <1>
707      <1> L16:
708
709      ; 10/11/2014
710 000003B7 FA      cli ; Disable interrupts (clear interrupt flag)
711      ; Reset Interrupt MASK Registers (Master&Slave)
712      ;mov al, 0FFh ; mask off all interrupts
713      ;out 21h, al ; on master PIC (8259)
714      ;jmp $+2 ; (delay)
715      ;out 0A1h, al ; on slave PIC (8259)
716      ;
717      ; Disable NMI
718 000003B8 B080      mov al, 80h
719 000003BA E670      out 70h, al ; set bit 7 to 1 for disabling NMI
720      ;23/02/2015
721 000003BC 90      nop ;
722      ;in al, 71h ; read in 71h just after writing out to 70h
723      ; for preventing unknown state (!?)
724      ;
725      ; 20/08/2014
726      ; Moving the kernel 64 KB back (to physical address 0)
727      ; DS = CS = 1000h
728      ; 05/11/2014
729 000003BD 31C0      xor ax, ax
730 000003BF 8EC0      mov es, ax ; ES = 0
731      ;
732 000003C1 B90040      mov cx, (KEND - KLOAD)/4
733 000003C4 31F6      xor si, si
734 000003C6 31FF      xor di, di
735 000003C8 F366A5      rep movsd
736      ;

```



```

737 000003CB 06          push es ; 0
738 000003CC 68[D003]    push L17
739 000003CF CB          retf
740                      ;
741                      L17:
742                      ; Turn off the floppy drive motor
743 000003D0 BAF203      mov     dx, 3F2h
744 000003D3 EE          out     dx, al ; 0 ; 31/12/2013
745
746                      ; Enable access to memory above one megabyte
747                      L18:
748 000003D4 E464        in      al, 64h
749 000003D6 A802        test   al, 2
750 000003D8 75FA        jnz     short L18
751 000003DA B0D1        mov     al, 0D1h ; Write output port
752 000003DC E664        out     64h, al
753                      L19:
754 000003DE E464        in      al, 64h
755 000003E0 A802        test   al, 2
756 000003E2 75FA        jnz     short L19
757 000003E4 B0DF        mov     al, 0DFh ; Enable A20 line
758 000003E6 E660        out     60h, al
759                      ;L20:
760                      ;
761                      ; Load global descriptor table register
762
763                      ;mov     ax, cs
764                      ;mov     ds, ax
765
766 000003E8 2E0F0116[8068] lgdt    [cs:gdt]
767
768 000003EE 0F20C0      mov     eax, cr0
769                      ; or eax, 1
770 000003F1 40          inc     ax
771 000003F2 0F22C0      mov     cr0, eax
772
773                      ; Jump to 32 bit code
774
775 000003F5 66          db 66h ; Prefix for 32-bit
776 000003F6 EA          db 0EAh ; Opcode for far jump
777 000003F7 [FD030000] dd StartPM ; Offset to start, 32-bit
778                      ; (1000h:StartPM = StartPM + 10000h)
779 000003FB 0800        dw KCODE ; This is the selector for CODE32_DESCRIPTOR,
780                      ; assuming that StartPM resides in code32
781
782                      [BITS 32]
783
784                      StartPM:
785                      ; Kernel Base Address = 0 ; 30/12/2013
786 000003FD 66B81000    mov ax, KDATA ; Save data segment identifier
787 00000401 8ED8        mov ds, ax ; Move a valid data segment into DS register
788 00000403 8EC0        mov es, ax ; Move data segment into ES register
789 00000405 8EE0        mov fs, ax ; Move data segment into FS register
790 00000407 8EE8        mov gs, ax ; Move data segment into GS register
791 00000409 8ED0        mov ss, ax ; Move data segment into SS register
792 0000040B BC00000900  mov esp, 90000h ; Move the stack pointer to 090000h
793
794                      clear_bss: ; Clear uninitialized data area
795                      ; 11/03/2015
796 00000410 31C0        xor     eax, eax ; 0
797 00000412 B9C5040000  mov     ecx, (bss_end - bss_start)/4
798                      ;shr     ecx, 2 ; bss section is already aligned for double words
799 00000417 BF[406E0000] mov     edi, bss_start
800 0000041C F3AB        rep     stosd
801
802                      memory_init:
803                      ; Initialize memory allocation table and page tables
804                      ; 16/11/2014
805                      ; 15/11/2014
806                      ; 07/11/2014
807                      ; 06/11/2014
808                      ; 05/11/2014
809                      ; 04/11/2014
810                      ; 31/10/2014 (Retro UNIX 386 v1 - Beginning)
811                      ;
812                      ;xor     eax, eax
813                      ;xor     ecx, ecx
814 0000041E B108        mov     cl, 8
815 00000420 BF00001000  mov     edi, MEM_ALLOC_TBL
816 00000425 F3AB        rep     stosd ; clear Memory Allocation Table
817                      ; for the first 1 MB memory
818                      ;
819 00000427 668B0D[D06D0000] mov     cx, [mem_1m_1k] ; Number of contiguous KB between
820                      ; 1 and 16 MB, max. 3C00h = 15 MB.
821 0000042E 66C1E902    shr     cx, 2 ; convert 1 KB count to 4 KB count
822 00000432 890D[B0700000] mov     [free_pages], ecx
823 00000438 668B15[D26D0000] mov     dx, [mem_16m_64k] ; Number of contiguous 64 KB blocks
824                      ; between 16 MB and 4 GB.
825 0000043F 6609D2    or      dx, dx
826 00000442 7413        jz      short mi_0
827                      ;
828 00000444 6689D0    mov     ax, dx

```



```

921                                     ; (1 M.A.T. byte points to
922                                     ;      32768 bytes)
923                                     ; Note: MEM_ALLOC_TBL address
924                                     ; must be aligned on 128 KB
925                                     ; boundary!
926 000004F3 01D7                      add  edi, edx      ; points to M.A.T.'s itself
927                                     ; eax = 0
928 000004F5 290D[B0700000]           sub  [free_pages], ecx ; 07/11/2014
929
930 000004FB 0FB307                   mi_8:
931                                     btr   [edi], eax    ; clear bit 0 to bit x (1 to 31)
932                                     ;dec  bl
933 000004FE FEC9                      dec   cl
934 00000500 7404                      jz    short mi_9
935 00000502 FEC0                      inc   al
936 00000504 EBF5                      jmp   short mi_8
937
938                                     mi_9:
939                                     ;
940                                     ; Reset Kernel's Page Dir. and Page Table bits in M.A.T.
941                                     ;      (allocate pages for system page tables)
942
943                                     ; edx = MEM_ALLOC_TBL
944 00000506 8B0D[AC700000]           mov   ecx, [memory_size] ; memory size in pages (PTEs)
945 0000050C 81C1FF030000           add   ecx, 1023      ; round up (1024 PTEs per table)
946 00000512 C1E90A                 shr   ecx, 10        ; convert memory page count to
947                                     ;      page table count (PDE count)
948
949 00000515 51                      ;
950                                     push  ecx          ; (**) PDE count (<= 1024)
951                                     ;
952 00000516 41                      inc   ecx          ; +1 for kernel page directory
953                                     ;
954 00000517 290D[B0700000]           sub  [free_pages], ecx ; 07/11/2014
955                                     ;
956 0000051D 8B35[A8700000]           mov   esi, [k_page_dir] ; Kernel's Page Directory address
957 00000523 C1EE0C                 shr   esi, 12        ; convert to page number
958
959 00000526 89F0                   mi_10:
960 00000528 89C3                   mov   eax, esi      ; allocation bit offset
961 0000052A C1EB03                   mov   ebx, eax
962 0000052D 80E3FC                   shr   ebx, 3        ; convert to alloc. byte offset
963                                     and   bl, 0FCh       ; clear bit 0 and bit 1
964                                     ;      to align on dword boundary
965 00000530 83E01F                   and   eax, 31       ; set allocation bit position
966                                     ;      (bit 0 to bit 31)
967
968 00000533 01D3                   ;
969 00000535 0FB303                   add   ebx, edx      ; offset in M.A.T. + M.A.T. address
970                                     ;
971 00000538 46                      btr   [ebx], eax    ; reset relevant bit (0 to 31)
972 00000539 E2EB                   ;
973                                     inc   esi          ; next page table
974                                     loop  mi_10         ; allocate next kernel page table
975                                     ;      (ecx = page table count + 1)
976
977 0000053B 59                      ;
978                                     pop   ecx          ; (**) PDE count (= pg. tbl. count)
979                                     ;
980                                     ; Initialize Kernel Page Directory and Kernel Page Tables
981                                     ;
982                                     ; Initialize Kernel's Page Directory
983 0000053C 8B3D[A8700000]           mov   edi, [k_page_dir]
984 00000542 89F8                   mov   eax, edi
985 00000544 0C03                   or    al, PDE_A_PRESENT + PDE_A_WRITE
986                                     ; supervisor + read&write + present
987 00000546 89CA                   mov   edx, ecx      ; (**) PDE count (= pg. tbl. count)
988
989 00000548 0500100000           mi_11:
990 0000054A F3AB                   add   eax, 4096     ; Add page size (PGSZ)
991                                     ; EAX points to next page table
992
993 0000054D AB                      stosd
994 0000054E E2F8                   loop  mi_11
995 00000550 29C0                   sub   eax, eax      ; Empty PDE
996 00000552 66B90004           mov   cx, 1024     ; Entry count (PGSZ/4)
997 00000556 29D1                   sub   ecx, edx
998 00000558 7402                   jz    short mi_12
999 0000055A F3AB                   rep   stosd        ; clear remain (empty) PDEs
1000                                     ;
1001                                     ; Initialization of Kernel's Page Directory is OK, here.
1002
1003                                     mi_12:
1004                                     ; Initialize Kernel's Page Tables
1005                                     ;
1006                                     ; (EDI points to address of page table 0)
1007 0000055C 8B0D[AC700000]           ; eax = 0
1008 00000562 89CA                   mov   ecx, [memory_size] ; memory size in pages
1009 00000564 B003                   mov   edx, ecx      ; (***)
1010                                     mov   al, PTE_A_PRESENT + PTE_A_WRITE
1011                                     ; supervisor + read&write + present
1012
1013                                     mi_13:
1014                                     stosd
1015 00000566 AB                      add   eax, 4096
1016 00000567 0500100000           loop  mi_13
1017 0000056C E2F8                   and   dx, 1023     ; (***)
1018 0000056E 6681E2FF03           jz    short mi_14
1019 00000573 740B                   mov   cx, 1024
1020 00000575 66B90004           sub   cx, dx        ; from dx (<= 1023) to 1024
1021 00000579 6629D1                   xor   eax, eax
1022 0000057C 31C0                   rep   stosd        ; clear remain (empty) PTEs
1023 0000057E F3AB

```

```

1013                                     ; of the last page table
1014 mi_14:
1015     ; Initialization of Kernel's Page Tables is OK, here.
1016     ;
1017 00000580 89F8      mov     eax, edi      ; end of the last page table page
1018                                     ; (beginning of user space pages)
1019 00000582 C1E80F    shr     eax, 15      ; convert to M.A.T. byte offset
1020 00000585 24FC      and     al, 0FCh     ; clear bit 0 and bit 1 for
1021                                     ; aligning on dword boundary
1022
1023 00000587 A3[BC700000] mov     [first_page], eax
1024 0000058C A3[B4700000] mov     [next_page], eax ; The first free page pointer
1025                                     ; for user programs
1026                                     ; (Offset in Mem. Alloc. Tbl.)
1027     ;
1028     ; Linear/FLAT (1 to 1) memory paging for the kernel is OK, here.
1029     ;
1030
1031     ; Enable paging
1032     ;
1033 00000591 A1[A8700000]      mov     eax, [k_page_dir]
1034 00000596 0F22D8      mov     cr3, eax
1035 00000599 0F20C0      mov     eax, cr0
1036 0000059C 0D00000080 or      eax, 80000000h     ; set paging bit (bit 31)
1037 000005A1 0F22C0      mov     cr0, eax
1038                                     ; jmp     KCODE:StartPMP
1039
1040 000005A4 EA          db 0EAh          ; Opcode for far jump
1041 000005A5 [AB050000]      dd StartPMP      ; 32 bit offset
1042 000005A9 0800      dw KCODE          ; kernel code segment descriptor
1043
1044
1045 StartPMP:
1046     ; 06/11//2014
1047     ; Clear video page 0
1048     ;
1049     ; Temporary Code
1050     ;
1051 000005AB B9E8030000      mov     ecx, 80*25/2
1052 000005B0 BF00800B00      mov     edi, 0B8000h
1053 000005B5 31C0      xor     eax, eax     ; black background, black fore color
1054 000005B7 F3AB      rep     stosd
1055
1056     ; 19/08/2014
1057     ; Kernel Base Address = 0
1058     ; It is mapped to (physically) 0 in the page table.
1059     ; So, here is exactly 'StartPMP' address.
1060     ;
1061     ; mov ah, 4Eh      ; Red background, yellow forecolor
1062     ; mov esi, msgPM
1063     ; 14/08/2015 (kernel version message will appear
1064     ; when protected mode and paging is enabled)
1065 000005B9 B40B      mov     ah, 0Bh ; Black background, light cyan forecolor
1066 000005BB BE[616B0000]      mov     esi, msgKVER
1067 000005C0 BF00800B00      mov     edi, 0B8000h ; 27/08/2014
1068     ; 20/08/2014
1069 000005C5 E896010000      call    printk
1070
1071     ; 'UNIX v7/x86' source code by Robert Nordier (1999)
1072     ; // Set IRQ offsets
1073     ;
1074     ; Linux (v0.12) source code by Linus Torvalds (1991)
1075     ;
1076                                     ;; ICW1
1077 000005CA B011      mov     al, 11h          ; Initialization sequence
1078 000005CC E620      out     20h, al          ; 8259A-1
1079     ; jmp     $+2
1080 000005CE E6A0      out     0A0h, al         ; 8259A-2
1081                                     ;; ICW2
1082 000005D0 B020      mov     al, 20h          ; Start of hardware ints (20h)
1083 000005D2 E621      out     21h, al          ; for 8259A-1
1084     ; jmp     $+2
1085 000005D4 B028      mov     al, 28h          ; Start of hardware ints (28h)
1086 000005D6 E6A1      out     0A1h, al         ; for 8259A-2
1087                                     ;
1088                                     ;; ICW3
1089 000005D8 B004      mov     al, 04h          ;
1090 000005DA E621      out     21h, al          ; IRQ2 of 8259A-1 (master)
1091     ; jmp     $+2
1092 000005DC B002      mov     al, 02h          ; is 8259A-2 (slave)
1093 000005DE E6A1      out     0A1h, al         ;
1094                                     ;; ICW4
1095 000005E0 B001      mov     al, 01h          ;
1096 000005E2 E621      out     21h, al          ; 8086 mode, normal EOI
1097     ; jmp     $+2
1098 000005E4 E6A1      out     0A1h, al         ; for both chips.
1099
1100     ; mov al, 0FFh     ; mask off all interrupts for now
1101     ; out 21h, al
1102     ; jmp     $+2
1103     ; out 0A1h, al
1104
1105     ; 02/04/2015

```

```

1105 ; 26/03/2015 System call (INT 30h) modification
1106 ; DPL = 3 (Interrupt service routine can be called from user mode)

1107 ;
1108 ;; Linux (v0.12) source code by Linus Torvalds (1991)
1109 ; setup_idt:
1110 ;
1111 ;;; 16/02/2015
1112 ;;mov dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
1113 ; 21/08/2014 (timer_int)
1114 000005E6 BE[8C680000] mov esi, ilist
1115 000005EB 8D3D[406E0000] lea edi, [idt]
1116 ; 26/03/2015
1117 000005F1 B930000000 mov ecx, 48 ; 48 hardware interrupts (INT 0 to INT 2Fh)
1118 ; 02/04/2015
1119 000005F6 BB00000800 mov ebx, 80000h
1120 rp_sidtl:
1121 000005FB AD lodsd
1122 000005FC 89C2 mov edx, eax
1123 000005FE 66BA008E mov dx, 8E00h
1124 00000602 6689C3 mov bx, ax
1125 00000605 89D8 mov eax, ebx ; /* selector = 0x0008 = cs */
1126 ; /* interrupt gate - dpl=0, present */
1127 00000607 AB stosd ; selector & offset bits 0-15
1128 00000608 89D0 mov eax, edx
1129 0000060A AB stosd ; attributes & offset bits 16-23
1130 0000060B E2EE loop rp_sidtl
1131 0000060D B110 mov cl, 16 ; 16 software interrupts (INT 30h to INT 3Fh)
1132 rp_sidtl2:
1133 0000060F AD lodsd
1134 00000610 21C0 and eax, eax
1135 00000612 7413 jz short rp_sidtl3
1136 00000614 89C2 mov edx, eax
1137 00000616 66BA00EE mov dx, 0EE00h ; P=1b/DPL=11b/01110b
1138 0000061A 6689C3 mov bx, ax
1139 0000061D 89D8 mov eax, ebx ; selector & offset bits 0-15
1140 0000061F AB stosd
1141 00000620 89D0 mov eax, edx
1142 00000622 AB stosd
1143 00000623 E2EA loop rp_sidtl2
1144 00000625 EB16 jmp short sidt_OK
1145 rp_sidtl3:
1146 00000627 B8[8A0A0000] mov eax, ignore_int
1147 0000062C 89C2 mov edx, eax
1148 0000062E 66BA00EE mov dx, 0EE00h ; P=1b/DPL=11b/01110b
1149 00000632 6689C3 mov bx, ax
1150 00000635 89D8 mov eax, ebx ; selector & offset bits 0-15
1151 rp_sidtl4:
1152 00000637 AB stosd
1153 00000638 92 xchg eax, edx
1154 00000639 AB stosd
1155 0000063A 92 xchg edx, eax
1156 0000063B E2FA loop rp_sidtl4
1157 sidt_OK:
1158 0000063D 0F011D[86680000] lidt [idtd]
1159 ;
1160 ; TSS descriptor setup ; 24/03/2015
1161 00000644 B8[40700000] mov eax, task_state_segment
1162 00000649 66A3[7A680000] mov [gdt_tss0], ax
1163 0000064F C1C010 rol eax, 16
1164 00000652 A2[7C680000] mov [gdt_tss1], al
1165 00000657 8825[7F680000] mov [gdt_tss2], ah
1166 0000065D 66C705[A6700000]68- mov word [tss.IOPB], tss_end - task_state_segment
1167 00000665 00
1168 ;
1169 ; IO Map Base address (When this address points
1170 ; to end of the TSS, CPU does not use IO port
1171 ; permission bit map for RING 3 IO permissions,
1172 ; access to any IO ports in ring 3 will be forbidden.)
1173 ;
1174 ;mov [tss.esp0], esp ; TSS offset 4
1175 ;mov word [tss.ss0], KDATA ; TSS offset 8 (SS)
1176 00000666 66B82800 mov ax, TSS ; It is needed when an interrupt
1177 ; occurs (or a system call -software INT- is requested)
1178 ; while cpu running in ring 3 (in user mode).
1179 ; (Kernel stack pointer and segment will be loaded
1180 ; from offset 4 and 8 of the TSS, by the CPU.)
1181 0000066A 0F00D8 ltr ax ; Load task register
1182 ;
1183 esp0_set0:
1184 ; 30/07/2015
1185 0000066D 8B0D[AC700000] mov ecx, [memory_size] ; memory size in pages
1186 00000673 C1E10C shl ecx, 12 ; convert page count to byte count
1187 00000676 81F900004000 cmp ecx, CORE ; beginning of user's memory space (400000h)
1188 ; (kernel mode virtual address)
1189 0000067C 7605 jna short esp0_set1
1190 ;
1191 ; If available memory > CORE (end of the 1st 4 MB)
1192 ; set stack pointer to CORE
1193 ; (Because, PDE 0 is reserved for kernel space in user's page directory)
1194 ; (PDE 0 points to page table of the 1st 4 MB virtual address space)
1195 0000067E B900004000 mov ecx, CORE

```

```

1196
1197 00000683 89CC
1198
1199
1200 00000685 8925[44700000]
1201 0000068B 66C705[48700000]10-
1202 00000693 00
1203
1204
1205
1206
1207
1208
1209 00000694 30C0
1210 00000696 E621
1211 00000698 EB00
1212 0000069A E6A1
1213
1214
1215
1216 0000069C B07F
1217 0000069E E670
1218
1219 000006A0 90
1220 000006A1 E471
1221
1222
1223
1224
1225
1226 000006A3 6631DB
1227 000006A6 66BA0002
1228 000006AA E8920F0000
1229
1230
1231
1232
1233 000006AF E8C1110000
1234
1235
1236
1237 000006B4 FB
1238
1239 000006B5 8B15[DA6D0000]
1240 000006BB 668B1D[D86D0000]
1241
1242 000006C2 6621DB
1243 000006C5 751B
1244
1245 000006C7 09D2
1246 000006C9 7529
1247
1248
1249 000006CB BE[856C0000]
1250
1251 000006D0 AC
1252 000006D1 08C0
1253
1254 000006D3 7426
1255 000006D5 56
1256 000006D6 31DB
1257
1258 000006D8 B407
1259
1260 000006DA E83E0E0000
1261 000006DF 5E
1262 000006E0 EBEE
1263
1264
1265
1266
1267 000006E2 66C705[E1070000]90-
1268 000006EA 90
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278 000006EB E8D6200000
1279
1280 000006F0 09D2
1281 000006F2 7407
1282
1283 000006F4 E813210000
1284 000006F9 72D0
1285
1286 000006FB E849110000
1287

esp0_set1:
mov esp, ecx ; top of kernel stack (**tss.esp0**)
esp0_set_ok:
; 30/07/2015 (**tss.esp0**)
mov [tss.esp0], esp
mov word [tss.ss0], KDATA

; 14/08/2015
; 10/11/2014 (Retro UNIX 386 v1 - Erdogan Tan)
;
; cli ; Disable interrupts (for CPU)
; (CPU will not handle hardware interrupts, except NMI!)
;
xor al, al ; Enable all hardware interrupts!
out 21h, al ; (IBM PC-AT compatibility)
jmp $+2 ; (All conventional PC-AT hardware
out 0A1h, al ; interrupts will be in use.)
; (Even if related hardware component
; does not exist!)

; Enable NMI
mov al, 7Fh ; Clear bit 7 to enable NMI (again)
out 70h, al
; 23/02/2015
nop
in al, 71h ; read in 71h just after writing out to 70h
; for preventing unknown state (!?)

;
; Only a NMI can occur here... (Before a 'STI' instruction)
;
; 02/09/2014
xor bx, bx
mov dx, 0200h ; Row 2, column 0 ; 07/03/2015
call set_cpos
;
; 06/11/2014
; Temporary Code
;
call memory_info
; 14/08/2015
; call getch ; 28/02/2015
drv_init:
sti ; Enable Interrupts
; 06/02/2015
mov edx, [hd0_type] ; hd0, hd1, hd2, hd3
mov bx, [fd0_type] ; fd0, fd1
; 22/02/2015
and bx, bx
jnz short di1
;
or edx, edx
jnz short di2
;
setup_error:
mov esi, setup_error_msg
psem:
lodsb
or al, al
; jz short haltx ; 22/02/2015
jz short di3
push esi
xor ebx, ebx ; 0
; Video page 0 (b1=0)
mov ah, 07h ; Black background,
; light gray forecolor
call write_tty
pop esi
jmp short psem

di1:
; supress 'jmp short T6'
; (activate fdc motor control code)
mov word [T5], 9090h ; nop

;
; mov ax, int_0Eh ; IRQ 6 handler
; mov di, 0Eh*4 ; IRQ 6 vector
; stosw
; mov ax, cs
; stosw
; ; 16/02/2015
; ; mov dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
;
CALL DSKETTE_SETUP ; Initialize Floppy Disks
;
or edx, edx
jz short di3
di2:
call DISK_SETUP ; Initialize Fixed Disks
jc short setup_error
di3:
call setup_rtc_int ; 22/05/2015 (dsctrpm.s)
;

```

```

1288 00000700 E8A2600000      call display_disks ; 07/03/2015  (Temporary)
1289                          ;haltx:
1290                          ; 14/08/2015
1291                          ;call getch ; 22/02/2015
1292 00000705 FB              sti    ; Enable interrupts (for CPU)
1293                          ; 14/08/2015
1294 00000706 B9FFFFFF0F      mov    ecx, 0FFFFFFFh
1295                          md_info_msg_wait:
1296 0000070B 51              push   ecx
1297 0000070C B001            mov     al, 1
1298 0000070E 8A25[D6700000]   mov     ah, [ptty] ; active (current) video page
1299 00000714 E8BF5D0000      call   getc_n
1300 00000719 59              pop     ecx
1301 0000071A 7502            jnz     short md_info_msg_ok
1302 0000071C E2ED            loop   md_info_msg_wait
1303                          md_info_msg_ok:
1304                          ; 30/06/2015
1305 0000071E E800310000      call   sys_init
1306                          ;
1307                          ;jmp  cpu_reset ; 22/02/2015
1308                          hang:
1309                          ; 23/02/2015
1310                          ;sti          ; Enable interrupts
1311 00000723 F4              hlt
1312                          ;
1313                          ;nop
1314                          ;; 03/12/2014
1315                          ;; 28/08/2014
1316                          ;mov  ah, 11h
1317                          ;call getc
1318                          ;jz     _c8
1319                          ;
1320                          ; 23/02/2015
1321                          ; 06/02/2015
1322                          ; 07/09/2014
1323 00000724 31DB            xor     ebx, ebx
1324 00000726 8A1D[D6700000]   mov     bl, [ptty] ; active_page
1325 0000072C 89DE            mov     esi, ebx
1326 0000072E 66D1E6          shl     si, 1
1327 00000731 81C6[D8700000]   add     esi, ttychr
1328 00000737 668B06          mov     ax, [esi]
1329 0000073A 6621C0          and     ax, ax
1330                          ;jz     short _c8
1331 0000073D 74E4            jz      short hang
1332 0000073F 66C7060000          mov     word [esi], 0
1333 00000744 80FB03          cmp     bl, 3 ; Video page 3
1334                          ;jb     short _c8
1335 00000747 72DA            jb      short hang
1336                          ;
1337                          ; 02/09/2014
1338 00000749 B40E            mov     ah, 0Eh ; Yellow character
1339                          ; on black background
1340                          ; 07/09/2014
1341                          nextl:
1342 0000074B 6653            push    bx
1343                          ;
1344                          ;xor    bx, bx ; bl = 0 (video page 0)
1345                          ; bh = 0 (video mode)
1346                          ; Retro UNIX 386 v1 - Video Mode 0
1347                          ; (PC/AT Video Mode 3 - 80x25 Alpha.)
1348 0000074D 6650            push    ax
1349 0000074F E8C90D0000      call   write_tty
1350 00000754 6658            pop     ax
1351 00000756 665B            pop     bx ; 07/09/2014
1352 00000758 3C0D            cmp     al, 0Dh ; carriage return (enter)
1353                          ;jne     short _c8
1354 0000075A 75C7            jne     short hang
1355 0000075C B00A            mov     al, 0Ah ; next line
1356 0000075E EBEB            jmp     short nextl
1357
1358                          ;_c8:
1359                          ; 25/08/2014
1360                          ;cli          ; Disable interrupts
1361                          ;mov  al, [scounter + 1]
1362                          ;and  al, al
1363                          ;jnz  hang
1364                          ;call  rtc_p
1365                          ;jmp   hang
1366
1367
1368                          ; 27/08/2014
1369                          ; 20/08/2014
1370                          printk:
1371                          ;mov     edi, [scr_row]
1372                          pkl:
1373 00000760 AC            lodsb
1374 00000761 08C0            or     al, al
1375 00000763 7404            jz      short pkr
1376 00000765 66AB            stosw
1377 00000767 EBF7            jmp     short pkl
1378                          pkr:
1379 00000769 C3            retn

```

```

1380
1381 ; 25/07/2015
1382 ; 14/05/2015 (multi tasking -time sharing- 'clock', x_timer)
1383 ; 17/02/2015
1384 ; 06/02/2015 (unix386.s)
1385 ; 11/12/2014 - 22/12/2014 (dsectrm2.s)
1386 ;
1387 ; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)
1388 ;
1389 ;-- HARDWARE INT 08 H - ( IRQ LEVEL 0 ) -----
1390 ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF :
1391 ; THE 8254 TIMER. INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR :
1392 ; IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND. :
1393 ; :
1394 ; THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE :
1395 ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY. :
1396 ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40) :
1397 ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE :
1398 ; DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS. :
1399 ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH :
1400 ; INTERRUPT 1CH AT EVERY TIME TICK. THE USER MUST CODE A :
1401 ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE. :
1402 ;-----
1403 ;
1404
1405 timer_int: ; IRQ 0
1406 ;int_08h: ; Timer
1407 ; 14/10/2015
1408 ; Here, we are simulating system call entry (for task switch)
1409 ; (If multitasking is enabled,
1410 ; 'clock' procedure may jump to 'sysrelease')
1411 0000076A 1E push ds
1412 0000076B 06 push es
1413 0000076C 0FA0 push fs
1414 0000076E 0FA8 push gs
1415 00000770 60 pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
1416 00000771 66B91000 mov cx, KDATA
1417 00000775 8ED9 mov ds, cx
1418 00000777 8EC1 mov es, cx
1419 00000779 8EE1 mov fs, cx
1420 0000077B 8EE9 mov gs, cx
1421 ;
1422 0000077D 0F20D9 mov ecx, cr3
1423 00000780 890D[1F080000] mov [cr3reg], ecx ; save current cr3 register value/content
1424 ;
1425 00000786 3B0D[A8700000] cmp ecx, [k_page_dir]
1426 0000078C 741F je short T3
1427 ;
1428 ; timer interrupt has been occurred while OS is in user mode
1429 0000078E A3[80740000] mov [u.r0], eax
1430 00000793 89E1 mov ecx, esp
1431 00000795 83C130 add ecx, ESPACE ; 4 * 12 (stack frame)
1432 00000798 890D[78740000] mov [u.sp], ecx ; kernel stack pointer at the start of interrupt
1433 0000079E 8925[7C740000] mov [u.usp], esp ; kernel stack points to user's registers
1434 ;
1435 000007A4 8B0D[A8700000] mov ecx, [k_page_dir]
1436 000007AA 0F22D9 mov cr3, ecx
1437 T3:
1438 000007AD FB sti ; INTERRUPTS BACK ON
1439 000007AE 66FF05[24710000] INC word [TIMER_LOW] ; INCREMENT TIME
1440 000007B5 7507 JNZ short T4 ; GO TO TEST_DAY
1441 000007B7 66FF05[26710000] INC word [TIMER_HIGH] ; INCREMENT HIGH WORD OF TIME
1442 T4: ; TEST_DAY
1443 000007BE 66833D[26710000]18 CMP word [TIMER_HIGH],018H ; TEST FOR COUNT EQUALING 24 HOURS
1444 000007C6 7519 JNZ short T5 ; GO TO DISKETTE_CTL
1445 000007C8 66813D[24710000]B0- CMP word [TIMER_LOW],0B0H
1446 000007D0 00
1447 000007D1 750E JNZ short T5 ; GO TO DISKETTE_CTL
1448
1449 ;----- TIMER HAS GONE 24 HOURS
1450 ;;SUB AX,AX
1451 ;MOV [TIMER_HIGH],AX
1452 ;MOV [TIMER_LOW],AX
1453 000007D3 29C0 sub eax, eax
1454 000007D5 A3[24710000] mov [TIMER_LH], eax
1455 ;
1456 000007DA C605[28710000]01 MOV byte [TIMER_OFL],1
1457
1458 ;----- TEST FOR DISKETTE TIME OUT
1459
1460 T5:
1461 ; 23/12/2014
1462 000007E1 EB1D jmp short T6 ; will be replaced with nop, nop
1463 ; (9090h) if a floppy disk
1464 ; is detected.
1465 ;mov al,[CS:MOTOR_COUNT]
1466 000007E3 A0[2B710000] mov al, [MOTOR_COUNT]
1467 000007E8 FEC8 dec al
1468 ;mov [CS:MOTOR_COUNT], al ; DECREMENT DISKETTE MOTOR CONTROL
1469 000007EA A2[2B710000] mov [MOTOR_COUNT], al
1470 ;mov [ORG MOTOR_COUNT], al
1471 000007EF 750F JNZ short T6 ; RETURN IF COUNT NOT OUT

```



```

1472 000007F1 B0F0      mov     al,0F0h
1473                    ;AND [CS:MOTOR_STATUS],al      ; TURN OFF MOTOR RUNNING BITS
1474 000007F3 2005[2A710000] and     [MOTOR_STATUS], al
1475                    ;and [ORG_MOTOR_STATUS], al
1476 000007F9 B00C      MOV     AL,0CH      ; bit 3 = enable IRQ & DMA,
1477                    ; bit 2 = enable controller
1478                    ;      1 = normal operation
1479                    ;      0 = reset
1480                    ; bit 0, 1 = drive select
1481                    ; bit 4-7 = motor running bits
1482 000007FB 66BAF203    MOV     DX,03F2H      ; FDC CTL PORT
1483 000007FF EE         OUT     DX,AL      ; TURN OFF THE MOTOR
1484
T6:
1485      ;inc word [CS:wait_count]      ; 22/12/2014 (byte -> word)
1486      ;                                ; TIMER TICK INTERRUPT
1487      ;;inc word [wait_count] ;;27/02/2015
1488      ;INT 1CH      ; TRANSFER CONTROL TO A USER ROUTINE
1489      ;;;cli
1490      ;call      u_timer      ; TRANSFER CONTROL TO A USER ROUTINE
1491 00000800 FF15[1B080000] call     [x_timer] ; 14/05/2015
1492
T7:
1493      ; 14/10/2015
1494 00000806 B020      MOV     AL,EOI      ; GET END OF INTERRUPT MASK
1495 00000808 FA        CLI      ; DISABLE INTERRUPTS TILL STACK CLEARED
1496 00000809 E620      OUT     INTA00,AL      ; END OF INTERRUPT TO 8259 - 1
1497
1498 0000080B A1[1F080000] mov     eax, [cr3reg]      ; previous value/content of cr3 register
1499 00000810 0F22D8    mov     cr3, eax ; restore cr3 register content
1500
1501 00000813 61        popad ; edi, esi, ebp, temp (increment esp by 4), ebx, edx, ecx, eax
1502
1503 00000814 0FA9      pop     gs
1504 00000816 0FA1      pop     fs
1505 00000818 07        pop     es
1506 00000819 1F        pop     ds
1507 0000081A CF        iretd ; return from interrupt
1508
1509
1510      ; //////////////////////////////////
1511
1512      ; 14/05/2015 - Multi tasking 'clock' procedure (sys emt)
1513 x_timer:
1514 0000081B [23080000] dd      u_timer      ; 14/05/2015
1515      ;dd clock
1516
1517      ; 14/10/2015
1518 0000081F 00000000 cr3reg: dd 0
1519
1520      ; 06/02/2015
1521      ; 07/09/2014
1522      ; 21/08/2014
1523 u_timer:
1524 ;timer_int: ; IRQ 0
1525      ; 06/02/2015
1526      ;push eax
1527      ;push edx
1528      ;push ecx
1529      ;push ebx
1530      ;push ds
1531      ;push es
1532      ;mov eax, KDATA
1533      ;mov ds, ax
1534      ;mov es, ax
1535 00000823 FF05[EC700000] inc     dword [tcount]
1536 00000829 BB[D26B0000] mov     ebx, tcountstr + 4
1537 0000082E 66A1[EC700000] mov     ax, [tcount]
1538 00000834 B90A000000 mov     ecx, 10
1539
rp_divtcnt:
1540 00000839 31D2      xor     edx, edx
1541 0000083B F7F1      div     ecx
1542 0000083D 80C230    add     dl, 30h
1543 00000840 8813      mov     [ebx], dl
1544 00000842 6609C0    or      ax, ax
1545 00000845 7403      jz      short print_lzero
1546 00000847 4B        dec     ebx
1547 00000848 EBEF      jmp     short rp_divtcnt
1548
print_lzero:
1549 0000084A 81FB[CE6B0000] cmp     ebx, tcountstr
1550 00000850 7606      jna     short print_tcount
1551 00000852 4B        dec     ebx
1552 00000853 C60330    mov     byte [ebx], 30h
1553 00000856 EBF2      jmp     short print_lzero
1554
print_tcount:
1555 00000858 56        push    esi
1556 00000859 57        push    edi
1557 0000085A BE[AA6B0000] mov     esi, timer_msg ; Timer interrupt message
1558      ; 07/09/2014
1559 0000085F 66BB0100 mov     bx, 1      ; Video page 1
1560
ptmsg:
1561 00000863 AC        lodsb
1562 00000864 08C0      or      al, al
1563 00000866 740F      jz      short ptmsg_ok

```

1564	00000868	56	push esi
1565	00000869	6653	push bx
1566	0000086B	B42F	mov ah, 2Fh ; Green background, white forecolor
1567	0000086D	E8AB0C0000	call write_tty
1568	00000872	665B	pop bx
1569	00000874	5E	pop esi
1570	00000875	EBEC	jmp short ptmsg
1571			; 27/08/2014
1572			;mov edi, 0B8000h + 0A0h ; Row 1
1573			;call printk
1574			;
1575			ptmsg_ok:
1576			; 07/09/2014
1577	00000877	6631D2	xor dx, dx ; column 0, row 0
1578	0000087A	E8C20D0000	call set_cpos ; set cursor position to 0,0
1579			; 23/02/2015
1580			; 25/08/2014
1581			;mov ebx, sccounter ; (seconds counter)
1582			;dec byte [ebx+1] ; (for reading real time clock)
1583			;dec byte [sccounter+1]
1584			; jns short timer_eoi ; 0 -> 0FFh ?
1585			;jns short u_timer_retn
1586			; 26/02/2015
1587			;call rtc_p
1588			;mov ebx, sccounter ; (seconds counter)
1589			;mov byte [ebx+1], 18 ; (18.2 timer ticks per second)
1590			;dec byte [ebx] ; 19+18+18+18+18 (5)
1591			;jnz short timer_eoi ; (109 timer ticks in 5 seconds)
1592			;jnz short u_timer_retn ; 06/02/2015
1593			;mov byte [ebx], 5
1594			;inc byte [ebx+1] ; 19
1595			;timer_eoi:
1596			; mov al, 20h ; END OF INTERRUPT COMMAND TO 8259
1597			; out 20h, al ; 8259 PORT
1598			;
1599			;u_timer_retn: ; 06/02/2015
1600	0000087F	5F	pop edi
1601	00000880	5E	pop esi
1602			;pop es
1603			;pop ds
1604			;pop ebx
1605			;pop ecx
1606			;pop edx
1607			;pop eax
1608			;iret
1609	00000881	C3	retn ; 06/02/2015
1610			
1611			; 28/08/2014
1612			irq0:
1613	00000882	6A00	push dword 0
1614	00000884	EB48	jmp short which_irq
1615			irq1:
1616	00000886	6A01	push dword 1
1617	00000888	EB44	jmp short which_irq
1618			irq2:
1619	0000088A	6A02	push dword 2
1620	0000088C	EB40	jmp short which_irq
1621			irq3:
1622			; 20/11/2015
1623			; 24/10/2015
1624	0000088E	2EFF15[7B3F0000]	call dword [cs:com2_irq3]
1625	00000895	6A03	push dword 3
1626	00000897	EB35	jmp short which_irq
1627			irq4:
1628			; 20/11/2015
1629			; 24/10/2015
1630	00000899	2EFF15[773F0000]	call dword [cs:com1_irq4]
1631	000008A0	6A04	push dword 4
1632	000008A2	EB2A	jmp short which_irq
1633			irq5:
1634	000008A4	6A05	push dword 5
1635	000008A6	EB26	jmp short which_irq
1636			irq6:
1637	000008A8	6A06	push dword 6
1638	000008AA	EB22	jmp short which_irq
1639			irq7:
1640	000008AC	6A07	push dword 7
1641	000008AE	EB1E	jmp short which_irq
1642			irq8:
1643	000008B0	6A08	push dword 8
1644	000008B2	EB1A	jmp short which_irq
1645			irq9:
1646	000008B4	6A09	push dword 9
1647	000008B6	EB16	jmp short which_irq
1648			irq10:
1649	000008B8	6A0A	push dword 10
1650	000008BA	EB12	jmp short which_irq
1651			irq11:
1652	000008BC	6A0B	push dword 11
1653	000008BE	EB0E	jmp short which_irq
1654			irq12:
1655	000008C0	6A0C	push dword 12

1656	000008C2	EB0A	jmp	short	which_irq
1657			irq13:		
1658	000008C4	6A0D		push	dword 13
1659	000008C6	EB06	jmp	short	which_irq
1660			irq14:		
1661	000008C8	6A0E		push	dword 14
1662	000008CA	EB02	jmp	short	which_irq
1663			irq15:		
1664	000008CC	6A0F		push	dword 15
1665			;jmp	short	which_irq
1666					
1667					; 19/10/2015
1668					; 29/08/2014
1669					; 21/08/2014
1670			which_irq:		
1671	000008CE	870424	xchg	eax,	[esp] ; 28/08/2014
1672	000008D1	53	push	ebx	
1673	000008D2	56	push	esi	
1674	000008D3	57	push	edi	
1675	000008D4	1E	push	ds	
1676	000008D5	06	push	es	
1677					
1678	000008D6	88C3	mov	bl,	al
1679					
1680	000008D8	B810000000	mov	eax,	KDATA
1681	000008DD	8ED8	mov	ds,	ax
1682	000008DF	8EC0	mov	es,	ax
1683					; 19/10/2015
1684	000008E1	FC	cld		
1685					; 27/08/2014
1686	000008E2	8105[596B0000]A000-	add	dword	[scr_row], 0A0h
1687	000008EA	0000			
1688					
1689	000008EC	B417	mov	ah,	17h ; blue (1) background,
1690					; light gray (7) forecolor
1691	000008EE	8B3D[596B0000]	mov	edi,	[scr_row]
1692	000008F4	B049	mov	al,	'I'
1693	000008F6	66AB	stosw		
1694	000008F8	B052	mov	al,	'R'
1695	000008FA	66AB	stosw		
1696	000008FC	B051	mov	al,	'Q'
1697	000008FE	66AB	stosw		
1698	00000900	B020	mov	al,	' '
1699	00000902	66AB	stosw		
1700	00000904	88D8	mov	al,	bl
1701	00000906	3C0A	cmp	al,	10
1702	00000908	7208	jb	short	iix
1703	0000090A	B031	mov	al,	'1'
1704	0000090C	66AB	stosw		
1705	0000090E	88D8	mov	al,	bl
1706	00000910	2C0A	sub	al,	10
1707			iix:		
1708	00000912	0430	add	al,	'0'
1709	00000914	66AB	stosw		
1710	00000916	B020	mov	al,	' '
1711	00000918	66AB	stosw		
1712	0000091A	B021	mov	al,	'!'
1713	0000091C	66AB	stosw		
1714	0000091E	B020	mov	al,	' '
1715	00000920	66AB	stosw		
1716					; 23/02/2015
1717	00000922	80FB07	cmp	bl,	7 ; check for IRQ 8 to IRQ 15
1718	00000925	0F868D010000	jna	iiret	
1719	0000092B	B020	mov	al,	20h ; END OF INTERRUPT COMMAND TO
1720	0000092D	E6A0	out	0A0h,	al ; the 2nd 8259
1721	0000092F	E984010000	jmp	iiret	
1722					
1723					; 22/08/2014
1724			;mov	al,	20h ; END OF INTERRUPT COMMAND TO 8259
1725			;out	20h,	al ; 8259 PORT
1726					
1727			;pop	es	
1728			;pop	ds	
1729			;pop	edi	
1730			;pop	esi	
1731			;pop	ebx	
1732			;pop	eax	
1733			;iret		
1734					
1735					; 02/04/2015
1736					; 25/08/2014
1737			exc0:		
1738	00000934	6A00		push	dword 0
1739	00000936	E990000000	jmp	cpu_except	
1740			exc1:		
1741	0000093B	6A01		push	dword 1
1742	0000093D	E989000000	jmp	cpu_except	
1743			exc2:		
1744	00000942	6A02		push	dword 2
1745	00000944	E982000000	jmp	cpu_except	
1746			exc3:		
1747	00000949	6A03		push	dword 3

1748	0000094B	EB7E			jmp	cpu_except
1749			exc4:		push dword 4	
1750	0000094D	6A04			jmp	cpu_except
1751	0000094F	EB7A				
1752			exc5:		push dword 5	
1753	00000951	6A05			jmp	cpu_except
1754	00000953	EB76				
1755			exc6:		push dword 6	
1756	00000955	6A06			jmp	cpu_except
1757	00000957	EB72				
1758			exc7:		push dword 7	
1759	00000959	6A07			jmp	cpu_except
1760	0000095B	EB6E				
1761			exc8:		; [esp] = Error code	
1762					push dword 8	
1763	0000095D	6A08			jmp	cpu_except_en
1764	0000095F	EB5C				
1765			exc9:		push dword 9	
1766	00000961	6A09			jmp	cpu_except
1767	00000963	EB66				
1768			exc10:		; [esp] = Error code	
1769					push dword 10	
1770	00000965	6A0A			jmp	cpu_except_en
1771	00000967	EB54				
1772			exc11:		; [esp] = Error code	
1773					push dword 11	
1774	00000969	6A0B			jmp	cpu_except_en
1775	0000096B	EB50				
1776			exc12:		; [esp] = Error code	
1777					push dword 12	
1778	0000096D	6A0C			jmp	cpu_except_en
1779	0000096F	EB4C				
1780			exc13:		; [esp] = Error code	
1781					push dword 13	
1782	00000971	6A0D			jmp	cpu_except_en
1783	00000973	EB48				
1784			exc14:		; [esp] = Error code	
1785					push dword 14	
1786	00000975	6A0E			jmp	short cpu_except_en
1787	00000977	EB44				
1788			exc15:		push dword 15	
1789	00000979	6A0F			jmp	cpu_except
1790	0000097B	EB4E				
1791			exc16:		push dword 16	
1792	0000097D	6A10			jmp	cpu_except
1793	0000097F	EB4A				
1794			exc17:		; [esp] = Error code	
1795					push dword 17	
1796	00000981	6A11			jmp	short cpu_except_en
1797	00000983	EB38				
1798			exc18:		push dword 18	
1799	00000985	6A12			jmp	short cpu_except
1800	00000987	EB42				
1801			exc19:		push dword 19	
1802	00000989	6A13			jmp	short cpu_except
1803	0000098B	EB3E				
1804			exc20:		push dword 20	
1805	0000098D	6A14			jmp	short cpu_except
1806	0000098F	EB3A				
1807			exc21:		push dword 21	
1808	00000991	6A15			jmp	short cpu_except
1809	00000993	EB36				
1810			exc22:		push dword 22	
1811	00000995	6A16			jmp	short cpu_except
1812	00000997	EB32				
1813			exc23:		push dword 23	
1814	00000999	6A17			jmp	short cpu_except
1815	0000099B	EB2E				
1816			exc24:		push dword 24	
1817	0000099D	6A18			jmp	short cpu_except
1818	0000099F	EB2A				
1819			exc25:		push dword 25	
1820	000009A1	6A19			jmp	short cpu_except
1821	000009A3	EB26				
1822			exc26:		push dword 26	
1823	000009A5	6A1A			jmp	short cpu_except
1824	000009A7	EB22				
1825			exc27:		push dword 27	
1826	000009A9	6A1B			jmp	short cpu_except
1827	000009AB	EB1E				
1828			exc28:		push dword 28	
1829	000009AD	6A1C			jmp	short cpu_except
1830	000009AF	EB1A				
1831			exc29:		push dword 29	
1832	000009B1	6A1D			jmp	short cpu_except
1833	000009B3	EB16				
1834			exc30:		push dword 30	
1835	000009B5	6A1E			jmp	short cpu_except_en
1836	000009B7	EB04				
1837			exc31:		push dword 31	
1838	000009B9	6A1F			jmp	short cpu_except
1839	000009BB	EB0E				

```

1840
1841 ; 19/10/2015
1842 ; 19/09/2015
1843 ; 01/09/2015
1844 ; 28/08/2015
1845 ; 28/08/2014
1846 cpu_except_en:
1847 000009BD 87442404 xchg eax, [esp+4] ; Error code
1848 000009C1 36A3[48810000] mov [ss:error_code], eax
1849 000009C7 58 pop eax ; Exception number
1850 000009C8 870424 xchg eax, [esp]
1851 ; eax = eax before exception
1852 ; [esp] -> exception number
1853 ; [esp+4] -> EIP to return
1854 ; 19/10/2015
1855 ; 19/09/2015
1856 ; 01/09/2015
1857 ; 28/08/2015
1858 ; 29/08/2014
1859 ; 28/08/2014
1860 ; 25/08/2014
1861 ; 21/08/2014
1862 cpu_except: ; CPU Exceptions
1863 000009CB FC cld
1864 000009CC 870424 xchg eax, [esp]
1865 ; eax = Exception number
1866 ; [esp] = eax (before exception)
1867 000009CF 53 push ebx
1868 000009D0 56 push esi
1869 000009D1 57 push edi
1870 000009D2 1E push ds
1871 000009D3 06 push es
1872 ; 28/08/2015
1873 000009D4 66BB1000 mov bx, KDATA
1874 000009D8 8EDB mov ds, bx
1875 000009DA 8EC3 mov es, bx
1876 000009DC 0F20DB mov ebx, cr3
1877 000009DF 53 push ebx ; (*) page directory
1878 ; 19/10/2015
1879 000009E0 FC cld
1880 ; 25/03/2015
1881 000009E1 8B1D[A8700000] mov ebx, [k_page_dir]
1882 000009E7 0F22DB mov cr3, ebx
1883 ; 28/08/2015
1884 000009EA 83F80E cmp eax, 0Eh ; 14, PAGE FAULT
1885 000009ED 7512 jne short cpu_except_nfp
1886 000009EF E815290000 call page_fault_handler
1887 000009F4 21C0 and eax, eax
1888 000009F6 0F84B8000000 jz iiretp ; 01/09/2015
1889 000009FC B80E000000 mov eax, 0Eh ; 14
1890 cpu_except_nfp:
1891 ; 02/04/2015
1892 00000A01 BB[23070000] mov ebx, hang
1893 00000A06 875C241C xchg ebx, [esp+28]
1894 ; EIP (points to instruction which faults)
1895 ; New EIP (hang)
1896 00000A0A 891D[4C810000] mov [FaultOffset], ebx
1897 00000A10 C744242008000000 mov dword [esp+32], KCODE ; kernel's code segment
1898 00000A18 814C242400020000 or dword [esp+36], 200h ; enable interrupts (set IF)
1899 ;
1900 00000A20 88C4 mov ah, al
1901 00000A22 240F and al, 0Fh
1902 00000A24 3C09 cmp al, 9
1903 00000A26 7602 jna short hlok
1904 00000A28 0407 add al, 'A'-' ':'
1905 hlok:
1906 00000A2A D0EC shr ah, 1
1907 00000A2C D0EC shr ah, 1
1908 00000A2E D0EC shr ah, 1
1909 00000A30 D0EC shr ah, 1
1910 00000A32 80FC09 cmp ah, 9
1911 00000A35 7603 jna short h2ok
1912 00000A37 80C407 add ah, 'A'-' ':'
1913 h2ok:
1914 00000A3A 86E0 xchg ah, al
1915 00000A3C 66053030 add ax, '00'
1916 00000A40 66A3[E66B0000] mov [excnstr], ax
1917 ;
1918 ; 29/08/2014
1919 00000A46 A1[4C810000] mov eax, [FaultOffset]
1920 00000A4B 51 push ecx
1921 00000A4C 52 push edx
1922 00000A4D 89E3 mov ebx, esp
1923 ; 28/08/2015
1924 00000A4F B910000000 mov ecx, 16 ; divisor value to convert binary number
1925 ; to hexadecimal string
1926 ;mov ecx, 10 ; divisor to convert
1927 ; binary number to decimal string
1928 b2d1:
1929 00000A54 31D2 xor edx, edx
1930 00000A56 F7F1 div ecx
1931 00000A58 6652 push dx

```

1932	00000A5A	39C8	cmp	eax, ecx
1933	00000A5C	73F6	jnb	short b2d1
1934	00000A5E	BF[F16B0000]	mov	edi, EIPstr ; EIP value
1935				; points to instruction which faults
1936				; 28/08/2015
1937	00000A63	89C2	mov	edx, eax
1938			b2d2:	
1939				;add al, '0'
1940	00000A65	8A82[31190000]	mov	al, [edx+hexchrs]
1941	00000A6B	AA	stosb	; write hexadecimal digit to its place
1942	00000A6C	39E3	cmp	ebx, esp
1943	00000A6E	7606	jna	short b2d3
1944	00000A70	6658	pop	ax
1945	00000A72	88C2	mov	dl, al
1946	00000A74	EBEF	jmp	short b2d2
1947			b2d3:	
1948	00000A76	B068	mov	al, 'h' ; 28/08/2015
1949	00000A78	AA	stosb	
1950	00000A79	B020	mov	al, 20h ; space
1951	00000A7B	AA	stosb	
1952	00000A7C	30C0	xor	al, al ; to do it an ASCIIZ string
1953	00000A7E	AA	stosb	
1954				
1955	00000A7F	5A	pop	edx
1956	00000A80	59	pop	ecx
1957				
1958	00000A81	B44F	mov	ah, 4Fh ; red (4) background,
1959				; white (F) forecolor
1960	00000A83	BE[D66B0000]	mov	esi, exc_msg ; message offset
1961				
1962	00000A88	EB11	jmp	short piemsg
1963				
1964				;add dword [scr_row], 0A0h
1965				;mov edi, [scr_row]
1966				
1967				;call printk
1968				
1969				;mov al, 20h ; END OF INTERRUPT COMMAND TO 8259
1970				;out 20h, al ; 8259 PORT
1971				
1972				;pop es
1973				;pop ds
1974				;pop edi
1975				;pop esi
1976				;pop eax
1977				;iret
1978				
1979				; 28/08/2015
1980				; 23/02/2015
1981				; 20/08/2014
1982			ignore_int:	
1983	00000A8A	50	push	eax
1984	00000A8B	53	push	ebx ; 23/02/2015
1985	00000A8C	56	push	esi
1986	00000A8D	57	push	edi
1987	00000A8E	1E	push	ds
1988	00000A8F	06	push	es
1989				; 28/08/2015
1990	00000A90	0F20D8	mov	eax, cr3
1991	00000A93	50	push	eax ; (*) page directory
1992				
1993	00000A94	B467	mov	ah, 67h ; brown (6) background,
1994				; light gray (7) forecolor
1995	00000A96	BE[946B0000]	mov	esi, int_msg ; message offset
1996			piemsg:	
1997				; 27/08/2014
1998	00000A9B	8105[596B0000]A000-	add	dword [scr_row], 0A0h
1999	00000AA3	0000		
2000	00000AA5	8B3D[596B0000]	mov	edi, [scr_row]
2001				
2002	00000AAB	E8B0FCFFFF	call	printk
2003				
2004				; 23/02/2015
2005	00000AB0	B020	mov	al, 20h ; END OF INTERRUPT COMMAND TO
2006	00000AB2	E6A0	out	0A0h, al ; the 2nd 8259
2007			iiiretp:	; 01/09/2015
2008				; 28/08/2015
2009	00000AB4	58	pop	eax ; (*) page directory
2010	00000AB5	0F22D8	mov	cr3, eax
2011				
2012			iiiret:	
2013				; 22/08/2014
2014	00000AB8	B020	mov	al, 20h ; END OF INTERRUPT COMMAND TO 8259
2015	00000ABA	E620	out	20h, al ; 8259 PORT
2016				
2017	00000ABC	07	pop	es
2018	00000ABD	1F	pop	ds
2019	00000ABE	5F	pop	edi
2020	00000ABF	5E	pop	esi
2021	00000AC0	5B	pop	ebx ; 29/08/2014
2022	00000AC1	58	pop	eax
2023	00000AC2	CF	iretd	

```

2024
2025 ; 26/02/2015
2026 ; 07/09/2014
2027 ; 25/08/2014
2028 rtc_int: ; Real Time Clock Interrupt (IRQ 8)
2029 ; 22/08/2014
2030 00000AC3 50 push eax
2031 00000AC4 53 push ebx ; 29/08/2014
2032 00000AC5 56 push esi
2033 00000AC6 57 push edi
2034 00000AC7 1E push ds
2035 00000AC8 06 push es
2036 ;
2037 00000AC9 B810000000 mov eax, KDATA
2038 00000ACE 8ED8 mov ds, ax
2039 00000AD0 8EC0 mov es, ax
2040 ;
2041 ; 25/08/2014
2042 00000AD2 E884000000 call rtc_p
2043 ;
2044 ; 22/02/2015 - dsectpm.s
2045 ; [ source: http://wiki.osdev.org/RTC ]
2046 ; read status register C to complete procedure
2047 ; (it is needed to get a next IRQ 8)
2048 00000AD7 B00C mov al, 0Ch ;
2049 00000AD9 E670 out 70h, al ; select register C
2050 00000ADB 90 nop
2051 00000ADC E471 in al, 71h ; just throw away contents
2052 ; 22/02/2015
2053 00000ADE B020 MOV AL,EOI ; END OF INTERRUPT
2054 00000AE0 E6A0 OUT INTB00,AL ; FOR CONTROLLER #2
2055 ;
2056 00000AE2 EBD4 jmp short iiret
2057
2058 ; 22/08/2014
2059 ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)
2060 ; (INT 1Ah)
2061 ;; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)
2062 time_of_day:
2063 00000AE4 E866010000 call UPD_IPR ; WAIT TILL UPDATE NOT IN PROGRESS
2064 00000AE9 726F jc short rtc_retn
2065 00000AEB B000 mov al, CMOS_SECONDS
2066 00000AED E845010000 call CMOS_READ
2067 00000AF2 A2[1C710000] mov [time_seconds], al
2068 00000AF7 B002 mov al, CMOS_MINUTES
2069 00000AF9 E839010000 call CMOS_READ
2070 00000AFE A2[1D710000] mov [time_minutes], al
2071 00000B03 B004 mov al, CMOS_HOURS
2072 00000B05 E82D010000 call CMOS_READ
2073 00000B0A A2[1E710000] mov [time_hours], al
2074 00000B0F B006 mov al, CMOS_DAY_WEEK
2075 00000B11 E821010000 call CMOS_READ
2076 00000B16 A2[1F710000] mov [date_wday], al
2077 00000B1B B007 mov al, CMOS_DAY_MONTH
2078 00000B1D E815010000 call CMOS_READ
2079 00000B22 A2[20710000] mov [date_day], al
2080 00000B27 B008 mov al, CMOS_MONTH
2081 00000B29 E809010000 call CMOS_READ
2082 00000B2E A2[21710000] mov [date_month], al
2083 00000B33 B009 mov al, CMOS_YEAR
2084 00000B35 E8FD000000 call CMOS_READ
2085 00000B3A A2[22710000] mov [date_year], al
2086 00000B3F B032 mov al, CMOS_CENTURY
2087 00000B41 E8F1000000 call CMOS_READ
2088 00000B46 A2[23710000] mov [date_century], al
2089 ;
2090 00000B4B B000 mov al, CMOS_SECONDS
2091 00000B4D E8E5000000 call CMOS_READ
2092 00000B52 3A05[1C710000] cmp al, [time_seconds]
2093 00000B58 758A jne short time_of_day
2094
2095 rtc_retn:
2096 00000B5A C3 retn
2097
2098 rtc_p:
2099 ; 07/09/2014
2100 ; 29/08/2014
2101 ; 27/08/2014
2102 ; 25/08/2014
2103 ; Print Real Time Clock content
2104 ;
2105 ;
2106 00000B5B E884FFFFFF call time_of_day
2107 00000B60 72F8 jc short rtc_retn
2108 ;
2109 00000B62 3A05[486C0000] cmp al, [ptime_seconds]
2110 00000B68 74F0 je short rtc_retn ; 29/08/2014
2111 ;
2112 00000B6A A2[486C0000] mov [ptime_seconds], al
2113 ;
2114 00000B6F A0[23710000] mov al, [date_century]
2115 00000B74 E8F1000000 call bcd_to_ascii

```

```

2116 00000B79 66A3[156C0000]      mov    [datestr+6], ax
2117 00000B7F A0[22710000]      mov    al, [date_year]
2118 00000B84 E8E1000000      call   bcd_to_ascii
2119 00000B89 66A3[176C0000]      mov    [datestr+8], ax
2120 00000B8F A0[21710000]      mov    al, [date_month]
2121 00000B94 E8D1000000      call   bcd_to_ascii
2122 00000B99 66A3[126C0000]      mov    [datestr+3], ax
2123 00000B9F A0[20710000]      mov    al, [date_day]
2124 00000BA4 E8C1000000      call   bcd_to_ascii
2125 00000BA9 66A3[0F6C0000]      mov    [datestr], ax
2126                                ;
2127 00000BAF 0FB61D[1F710000]      movzx  ebx, byte [date_wday]
2128 00000BB6 C0E302          shl    bl, 2
2129 00000BB9 81C3[286C0000]      add    ebx, daytmp
2130 00000BBF 8B03          mov    eax, [ebx]
2131 00000BC1 A3[1A6C0000]      mov    [daystr], eax
2132                                ;
2133 00000BC6 A0[1E710000]      mov    al, [time_hours]
2134 00000BCB E89A000000      call   bcd_to_ascii
2135 00000BD0 66A3[1E6C0000]      mov    [timestr], ax
2136 00000BD6 A0[1D710000]      mov    al, [time_minutes]
2137 00000BDB E88A000000      call   bcd_to_ascii
2138 00000BE0 66A3[216C0000]      mov    [timestr+3], ax
2139 00000BE6 A0[1C710000]      mov    al, [time_seconds]
2140 00000BEB E87A000000      call   bcd_to_ascii
2141 00000BF0 66A3[246C0000]      mov    [timestr+6], ax
2142                                ;
2143 00000BF6 BE[FD6B0000]      mov    esi, rtc_msg ; message offset
2144                                ; 23/02/2015
2145 00000BFB 52          push  edx
2146 00000BFC 51          push  ecx
2147                                ; 07/09/2014
2148 00000BFD 66BB0200      mov    bx, 2          ; Video page 2
2149 prtmsg:
2150 00000C01 AC          lodsb
2151 00000C02 08C0          or     al, al
2152 00000C04 740F          jz     short prtmsg_ok
2153 00000C06 56          push  esi
2154 00000C07 6653          push  bx
2155 00000C09 B43F          mov    ah, 3Fh      ; cyan (6) background,
2156                                ; white (F) forecolor
2157 00000C0B E80D090000      call   write_tty
2158 00000C10 665B          pop    bx
2159 00000C12 5E          pop    esi
2160 00000C13 EBEC          jmp     short prtmsg
2161                                ;
2162                                ;mov edi, 0B8000h+0A0h+0A0h ; Row 2
2163                                ;call printk
2164 prtmsg_ok:
2165                                ; 07/09/2014
2166 00000C15 6631D2      xor    dx, dx          ; column 0, row 0
2167 00000C18 E8240A0000      call   set_cpos      ; set cursor position to 0,0
2168                                ; 23/02/2015
2169 00000C1D 59          pop    ecx
2170 00000C1E 5A          pop    edx
2171 00000C1F C3          retn
2172
2173 ; Default IRQ 7 handler against spurious IRQs (from master PIC)
2174 ; 25/02/2015 (source: http://wiki.osdev.org/8259\_PIC)
2175 default_irq7:
2176 00000C20 6650          push  ax
2177 00000C22 B00B          mov    al, 0Bh      ; In-Service register
2178 00000C24 E620          out    20h, al
2179 00000C26 EB00          jmp     short $+2
2180 00000C28 EB00          jmp     short $+2
2181 00000C2A E420          in     al, 20h
2182 00000C2C 2480          and    al, 80h ; bit 7 (is it real IRQ 7 or fake?)
2183 00000C2E 7404          jz     short irq7_iret ; Fake (spurious) IRQ, do not send EOI
2184 00000C30 B020          mov    al, 20h ; EOI
2185 00000C32 E620          out    20h, al
2186 irq7_iret:
2187 00000C34 6658          pop    ax
2188 00000C36 CF          iretd
2189
2190                                ; 22/08/2014
2191                                ; IBM PC/AT BIOS source code ----- 10/06/85 (test4.asm)
2192 CMOS_READ:
2193 00000C37 9C          pushf      ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
2194 00000C38 D0C0          rol    al, 1 ; MOVE NMI BIT TO LOW POSITION
2195 00000C3A F9          stc          ; FORCE NMI BIT ON IN CARRY FLAG
2196 00000C3B D0D8          rcr    al, 1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
2197 00000C3D FA          cli          ; DISABLE INTERRUPTS
2198 00000C3E E670          out    CMOS_PORT, al ; ADDRESS LOCATION AND DISABLE NMI
2199 00000C40 90          nop          ; I/O DELAY
2200 00000C41 E471          in     al, CMOS_DATA ; READ THE REQUESTED CMOS LOCATION
2201 00000C43 6650          push  ax      ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
2202                                ; 15/03/2015 ; IBM PC/XT Model 286 BIOS source code
2203                                ; ----- 10/06/85 (test4.asm)
2204 00000C45 B01E          mov    al, CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
2205 ;mov al, CMOS_REG_D*2 ; GET ADDRESS OF DEFAULT LOCATION
2206 00000C47 D0D8          rcr    al, 1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
2207 00000C49 E670          out    CMOS_PORT, al ; SET DEFAULT TO READ ONLY REGISTER

```



```

2208 00000C4B 6658      pop  ax      ; RESTORE (AH) AND (AL), CMOS BYTE
2209 00000C4D 9D        popf
2210 00000C4E C3        retn          ; RETURN WITH FLAGS RESTORED
2211
2212                    ; 22/08/2014
2213                    ; IBM PC/AT BIOS source code ----- 10/06/85 (bios2.asm)
2214 UPD_IPR:              ; WAIT TILL UPDATE NOT IN PROGRESS
2215 00000C4F 51          push  ecx
2216 00000C50 B9FFFF0000 mov  ecx, 65535      ; SET TIMEOUT LOOP COUNT (= 800)
2217                    ; mov cx, 800
2218 UPD_10:
2219 00000C55 B00A        mov  al, CMOS_REG_A      ; ADDRESS STATUS REGISTER A
2220 00000C57 FA          cli          ; NO TIMER INTERRUPTS DURING UPDATES
2221 00000C58 E8DAFFFFFF call  CMOS_READ      ; READ UPDATE IN PROCESS FLAG
2222 00000C5D A880        test  al, 80h          ; IF UIP BIT IS ON ( CANNOT READ TIME )
2223 00000C5F 7406        jz    short UPD_90      ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
2224 00000C61 FB          sti          ; ALLOW INTERRUPTS WHILE WAITING
2225 00000C62 E2F1        loop  UPD_10          ; LOOP TILL READY OR TIMEOUT
2226 00000C64 31C0        xor    eax, eax      ; CLEAR RESULTS IF ERROR
2227                    ; xor ax, ax
2228 00000C66 F9          stc          ; SET CARRY FOR ERROR
2229 UPD_90:
2230 00000C67 59          pop    ecx      ; RESTORE CALLERS REGISTER
2231 00000C68 FA          cli          ; INTERRUPTS OFF DURING SET
2232 00000C69 C3        retn          ; RETURN WITH CY FLAG SET
2233
2234 bcd_to_ascii:
2235                    ; 25/08/2014
2236                    ; INPUT ->
2237                    ;     al = Packed BCD number
2238                    ; OUTPUT ->
2239                    ;     ax = ASCII word/number
2240                    ;
2241                    ; Erdogan Tan - 1998 (proc_hex) - TRDOS.ASM (2004-2011)
2242                    ;
2243 00000C6A D410        db  0D4h,10h          ; Undocumented inst. AAM
2244                    ; AH = AL / 10h
2245                    ; AL = AL MOD 10h
2246 00000C6C 660D3030    or  ax,'00'          ; Make it ASCII based
2247
2248 00000C70 86E0        xchg  ah, al
2249
2250 00000C72 C3        retn
2251
2252
2253                    %include 'keyboard.inc' ; 07/03/2015
2254 <1> ; Retro UNIX 386 v1 Kernel - KEYBOARD.INC
2255 <1> ; Last Modification: 17/10/2015
2256 <1> ;           (Keyboard Data is in 'KYBDATA.INC')
2257 <1> ;
2258 <1> ; //////////// KEYBOARD FUNCTIONS (PROCEDURES) ////////////
2259 <1> ;
2260 <1> ; 30/06/2015
2261 <1> ; 11/03/2015
2262 <1> ; 28/02/2015
2263 <1> ; 25/02/2015
2264 <1> ; 20/02/2015
2265 <1> ; 18/02/2015
2266 <1> ; 03/12/2014
2267 <1> ; 07/09/2014
2268 <1> ; KEYBOARD INTERRUPT HANDLER
2269 <1> ; (kb_int - Retro UNIX 8086 v1 - U0.ASM, 30/06/2014)
2270 <1> ;
2271 <1> ;getch:
2272 <1> ; ; 18/02/2015
2273 <1> ; ; This routine will be replaced with Retro UNIX 386
2274 <1> ; ; version of Retro UNIX 8086 getch (tty input)
2275 <1> ; ; routine, later... (multi tasking ability)
2276 <1> ; ; 28/02/2015
2277 <1> ; sti      ; enable interrupts
2278 <1> ; ;
2279 <1> ; ;push esi
2280 <1> ; ;push ebx
2281 <1> ; ;xor  ebx, ebx
2282 <1> ; ;mov  bl, [ptty] ; active_page
2283 <1> ; ;mov  esi, ebx
2284 <1> ; ;shl  si, 1
2285 <1> ; ;add  esi, ttychr
2286 <1> ;getch_1:
2287 <1> ; ;mov  ax, [esi]
2288 <1> ; mov  ax, [ttychr] ; video page 0 (tty0)
2289 <1> ; and  ax, ax
2290 <1> ; jz    short getch_2
2291 <1> ; mov  word [ttychr], 0
2292 <1> ; ;mov  word [esi], 0
2293 <1> ; ;pop  ebx
2294 <1> ; ;pop  esi
2295 <1> ; retn
2296 <1> ;getch_2:
2297 <1> ; hlt      ; not proper for multi tasking!
2298 <1> ;           ; (temporary halt for now)
2299 <1> ;           ; 'sleep' on tty

```

2300	<1> ;	; will (must) be located here
2301	<1> ;nop	
2302	<1> ;jmp	short getch_1
2303	<1>	
2304	<1>	keyb_int:
2305	<1>	; 30/06/2015
2306	<1>	; 25/02/2015
2307	<1>	; 20/02/2015
2308	<1>	; 03/12/2014 (getc_int - INT 16h modifications)
2309	<1>	; 07/09/2014 - Retro UNIX 386 v1
2310	<1>	; 30/06/2014
2311	<1>	; 10/05/2013
2312	<1>	; Retro Unix 8086 v1 feature only!
2313	<1>	; 03/03/2014
2314	<1>	
2315 00000C73 1E	<1>	push ds
2316 00000C74 53	<1>	push ebx
2317 00000C75 50	<1>	push eax
2318	<1>	;
2319 00000C76 66B81000	<1>	mov ax, KDATA
2320 00000C7A 8ED8	<1>	mov ds, ax
2321	<1>	;
2322 00000C7C 9C	<1>	pushfd
2323 00000C7D 0E	<1>	push cs
2324 00000C7E E823020000	<1>	call kb_int ; int_09h
2325	<1>	;
2326 00000C83 B411	<1>	mov ah, 11h ; 03/12/2014
2327	<1>	;call getc
2328 00000C85 E856000000	<1>	call int_16h ; 30/06/2015
2329 00000C8A 7450	<1>	jz short keyb_int4
2330	<1>	;
2331 00000C8C B410	<1>	mov ah, 10h ; 03/12/2014
2332	<1>	;call getc
2333 00000C8E E84D000000	<1>	call int_16h ; 30/06/2015
2334	<1>	;
2335	<1>	; 20/02/2015
2336 00000C93 0FB61D[D6700000]	<1>	movzx ebx, byte [ptty] ; active_page
2337	<1>	;
2338 00000C9A 20C0	<1>	and al, al
2339 00000C9C 751E	<1>	jnz short keyb_int1
2340	<1>	;
2341 00000C9E 80FC68	<1>	cmp ah, 68h ; ALT + F1 key
2342 00000CA1 7219	<1>	jb short keyb_int1
2343 00000CA3 80FC6F	<1>	cmp ah, 6Fh ; ALT + F8 key
2344 00000CA6 7714	<1>	ja short keyb_int1
2345	<1>	;
2346 00000CA8 88D8	<1>	mov al, bl
2347 00000CAA 0468	<1>	add al, 68h
2348 00000CAC 38E0	<1>	cmp al, ah
2349 00000CAE 7409	<1>	je short keyb_int0
2350 00000CB0 88E0	<1>	mov al, ah
2351 00000CB2 2C68	<1>	sub al, 68h
2352 00000CB4 E8510B0000	<1>	call tty_sw
2353	<1>	;movzx ebx, [ptty] ; active_page
2354	<1>	keyb_int0: ; 30/06/2015
2355 00000CB9 6631C0	<1>	xor ax, ax
2356	<1>	keyb_int1:
2357 00000CBC D0E3	<1>	shl bl, 1
2358 00000CBE 81C3[D8700000]	<1>	add ebx, ttychr
2359	<1>	;
2360 00000CC4 6609C0	<1>	or ax, ax
2361 00000CC7 7406	<1>	jz short keyb_int2
2362	<1>	;
2363 00000CC9 66833B00	<1>	cmp word [ebx], 0
2364 00000CCD 7703	<1>	ja short keyb_int3
2365	<1>	keyb_int2:
2366 00000CCF 668903	<1>	mov [ebx], ax ; Save ascii code
2367	<1>	; and scan code of the character
2368	<1>	; for current tty (or last tty
2369	<1>	; just before tty switch).
2370	<1>	keyb_int3:
2371 00000CD2 A0[D6700000]	<1>	mov al, [ptty]
2372 00000CD7 E811480000	<1>	call wakeup
2373	<1>	;
2374	<1>	keyb_int4:
2375 00000CDC 58	<1>	pop eax
2376 00000CDD 5B	<1>	pop ebx
2377 00000CDE 1F	<1>	pop ds
2378 00000CDF CF	<1>	iret
2379	<1>	
2380	<1>	; 18/02/2015
2381	<1>	; REMINDER: Only 'keyb_int' (IRQ 9) must call getc.
2382	<1>	; 'keyb_int' always handles 'getc' at 1st and puts the
2383	<1>	; scancode and ascii code of the character
2384	<1>	; in the tty input (ttychr) buffer.
2385	<1>	; Test procedures must call 'getch' for tty input
2386	<1>	; otherwise, 'getc' will not be able to return to the caller
2387	<1>	; due to infinite (key press) waiting loop.
2388	<1>	;
2389	<1>	; 03/12/2014
2390	<1>	; 26/08/2014
2391	<1>	; KEYBOARD I/O

```

2392 <1> ; (INT_16h - Retro UNIX 8086 v1 - U9.ASM, 30/06/2014)
2393 <1>
2394 <1> ;NOTE: 'k0' to 'k7' are name of OPMASK registers.
2395 <1> ; (The reason of using '_k' labels!!!) (27/08/2014)
2396 <1> ;NOTE: 'NOT' keyword is '~' unary operator in NASM.
2397 <1> ; ('NOT LC_HC' --> '~LC_HC') (bit reversing operator)
2398 <1>
2399 <1> int_16h: ; 30/06/2015
2400 <1> ;getc:
2401 00000CE0 9C <1> pushfd ; 28/08/2014
2402 00000CE1 0E <1> push cs
2403 00000CE2 E801000000 <1> call getc_int
2404 00000CE7 C3 <1> retn
2405 <1>
2406 <1> getc_int:
2407 <1> ; 28/02/2015
2408 <1> ; 03/12/2014 (derivation from pc-xt-286 bios source code -1986-,
2409 <1> ; instead of pc-at bios - 1985-)
2410 <1> ; 28/08/2014 (_kld)
2411 <1> ; 30/06/2014
2412 <1> ; 03/03/2014
2413 <1> ; 28/02/2014
2414 <1> ; Derived from "KEYBOARD_IO_1" procedure of IBM "pc-xt-286"
2415 <1> ; rombios source code (21/04/1986)
2416 <1> ; 'keybd.asm', INT 16H, KEYBOARD_IO
2417 <1> ;
2418 <1> ; KYBD --- 03/06/86 KEYBOARD BIOS
2419 <1> ;
2420 <1> ;--- INT 16 H -----
2421 <1> ; KEYBOARD I/O :
2422 <1> ; THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT :
2423 <1> ; INPUT :
2424 <1> ; (AH)= 00H READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD, :
2425 <1> ; RETURN THE RESULT IN (AL), SCAN CODE IN (AH). :
2426 <1> ; THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO THE :
2427 <1> ; STANDARD PC OR PCAT KEYBOARD :
2428 <1> ;-----:
2429 <1> ; (AH)= 01H SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER IS :
2430 <1> ; AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER. :
2431 <1> ; (ZF)= 1 -- NO CODE AVAILABLE :
2432 <1> ; (ZF)= 0 -- CODE IS AVAILABLE (AX)= CHARACTER :
2433 <1> ; IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS :
2434 <1> ; IN (AX), AND THE ENTRY REMAINS IN THE BUFFER. :
2435 <1> ; THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE CODES :
2436 <1> ;-----:

2437 <1> ; (AH)= 02H RETURN THE CURRENT SHIFT STATUS IN AL REGISTER :
2438 <1> ; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE :
2439 <1> ; EQUATES FOR @KB_FLAG :
2440 <1> ;-----:

2441 <1> ; (AH)= 03H SET TYPAMATIC RATE AND DELAY :
2442 <1> ; (AL) = 05H :
2443 <1> ; (BL) = TYPAMATIC RATE (BITS 5 - 7 MUST BE RESET TO 0) :
2444 <1> ; :
2445 <1> ; REGISTER RATE REGISTER RATE :
2446 <1> ; VALUE SELECTED VALUE SELECTED :
2447 <1> ; -----:
2448 <1> ; 00H 30.0 10H 7.5 :
2449 <1> ; 01H 26.7 11H 6.7 :
2450 <1> ; 02H 24.0 12H 6.0 :
2451 <1> ; 03H 21.8 13H 5.5 :
2452 <1> ; 04H 20.0 14H 5.0 :
2453 <1> ; 05H 18.5 15H 4.6 :
2454 <1> ; 06H 17.1 16H 4.3 :
2455 <1> ; 07H 16.0 17H 4.0 :
2456 <1> ; 08H 15.0 18H 3.7 :
2457 <1> ; 09H 13.3 19H 3.3 :
2458 <1> ; 0AH 12.0 1AH 3.0 :
2459 <1> ; 0BH 10.9 1BH 2.7 :
2460 <1> ; ; 0CH 10.0 1CH 2.5 :
2461 <1> ; 0DH 9.2 1DH 2.3 :
2462 <1> ; 0EH 8.6 1EH 2.1 :
2463 <1> ; 0FH 8.0 1FH 2.0 :
2464 <1> ; :
2465 <1> ; (BH) = TYPAMATIC DELAY (BITS 2 - 7 MUST BE RESET TO 0) :
2466 <1> ; :
2467 <1> ; REGISTER DELAY :
2468 <1> ; VALUE VALUE :
2469 <1> ; -----:
2470 <1> ; 00H 250 ms :
2471 <1> ; 01H 500 ms :
2472 <1> ; 02H 750 ms :
2473 <1> ; 03H 1000 ms :
2474 <1> ;-----:
2475 <1> ; (AH)= 05H PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD :
2476 <1> ; BUFFER AS IF STRUCK FROM KEYBOARD :
2477 <1> ; ENTRY: (CL) = ASCII CHARACTER :
2478 <1> ; (CH) = SCAN CODE :
2479 <1> ; EXIT: (AH) = 00H = SUCCESSFUL OPERATION :
2480 <1> ; (AL) = 01H = UNSUCCESSFUL - BUFFER FULL :
2481 <1> ; FLAGS: CARRY IF ERROR :

```

```

2482      <1> ;-----:
2483      <1> ;      (AH)= 10H  EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD,      :
2484      <1> ;      OTHERWISE SAME AS FUNCTION AH=0      :
2485      <1> ;-----:
2486      <1> ;      (AH)= 11H  EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,      :
2487      <1> ;      OTHERWISE SAME AS FUNCTION AH=1      :
2488      <1> ;-----:

2489      <1> ;      (AH)= 12H  RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER      :
2490      <1> ;      AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT      :
2491      <1> ;      CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3      :
2492      <1> ; OUTPUT      :
2493      <1> ;      AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED      :
2494      <1> ;      ALL REGISTERS RETAINED      :
2495      <1> ;-----:
2496      <1>
2497      00000CE8 FB      <1> sti      ; INTERRUPTS BACK ON
2498      00000CE9 1E      <1> push ds      ; SAVE CURRENT DS
2499      00000CEA 53      <1> push ebx      ; SAVE BX TEMPORARILY
2500      <1> ;push ecx      ; SAVE CX TEMPORARILY
2501      00000CEB 66BB1000      <1> mov      bx, KDATA
2502      00000CEF 8EDB      <1> mov      ds, bx      ; PUT SEGMENT VALUE OF DATA AREA INTO DS
2503      00000CF1 08E4      <1> or      ah, ah      ; CHECK FOR (AH)= 00H
2504      00000CF3 7439      <1> jz      short _K1      ; ASCII_READ
2505      00000CF5 FECC      <1> dec      ah      ; CHECK FOR (AH)= 01H
2506      00000CF7 7452      <1> jz      short _K2      ; ASCII_STATUS
2507      00000CF9 FECC      <1> dec      ah      ; CHECK FOR (AH)= 02H
2508      00000CFB 0F8485000000      <1> jz      _K3      ; SHIFT STATUS
2509      00000D01 FECC      <1> dec      ah      ; CHECK FOR (AH)= 03H
2510      00000D03 0F8484000000      <1> jz      _K300      ; SET TYPAMATIC RATE/DELAY
2511      00000D09 80EC02      <1> sub      ah, 2      ; CHECK FOR (AH)= 05H
2512      00000D0C 0F84A1000000      <1> jz      _K500      ; KEYBOARD WRITE
2513      <1> _KIO1:
2514      00000D12 80EC0B      <1> sub      ah, 11      ; AH = 10H
2515      00000D15 740B      <1> jz      short _K1E      ; EXTENDED ASCII READ
2516      00000D17 FECC      <1> dec      ah      ; CHECK FOR (AH)= 11H
2517      00000D19 7421      <1> jz      short _K2E      ; EXTENDED_ASCII_STATUS
2518      00000D1B FECC      <1> dec      ah      ; CHECK FOR (AH)= 12H
2519      00000D1D 7449      <1> jz      short _K3E      ; EXTENDED_SHIFT_STATUS
2520      <1> _KIO_EXIT:
2521      <1> ;pop ecx      ; RECOVER REGISTER
2522      00000D1F 5B      <1> pop      ebx      ; RECOVER REGISTER
2523      00000D20 1F      <1> pop      ds      ; RECOVER SEGMENT
2524      00000D21 CF      <1> iretd      ; INVALID COMMAND, EXIT
2525      <1>
2526      <1> ;----- ASCII CHARACTER
2527      <1> _K1E:
2528      00000D22 E8B9000000      <1> call     _K1S      ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
2529      00000D27 E82E010000      <1> call     _KIO_E_XLAT      ; ROUTINE TO XLATE FOR EXTENDED CALLS
2530      00000D2C EBF1      <1> jmp      short _KIO_EXIT      ; GIVE IT TO THE CALLER
2531      <1> _K1:
2532      00000D2E E8AD000000      <1> call     _K1S      ; GET A CHARACTER FROM THE BUFFER
2533      00000D33 E82D010000      <1> call     _KIO_S_XLAT      ; ROUTINE TO XLATE FOR STANDARD CALLS
2534      00000D38 72F4      <1> jc      short _K1      ; CARRY SET MEANS TROW CODE AWAY
2535      <1> _K1A:
2536      00000D3A EBE3      <1> jmp      short _KIO_EXIT      ; RETURN TO CALLER
2537      <1>
2538      <1> ;----- ASCII STATUS
2539      <1> _K2E:
2540      00000D3C E8EA000000      <1> call     _K2S      ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
2541      00000D41 7420      <1> jz      short _K2B      ; RETURN IF BUFFER EMPTY
2542      00000D43 9C      <1> pushf      ; SAVE ZF FROM TEST
2543      00000D44 E811010000      <1> call     _KIO_E_XLAT      ; ROUTINE TO XLATE FOR EXTENDED CALLS
2544      00000D49 EB17      <1> jmp      short _K2A      ; GIVE IT TO THE CALLER
2545      <1> _K2:
2546      00000D4B E8DB000000      <1> call     _K2S      ; TEST FOR CHARACTER IN BUFFER
2547      00000D50 7411      <1> jz      short _K2B      ; RETURN IF BUFFER EMPTY
2548      00000D52 9C      <1> pushf      ; SAVE ZF FROM TEST
2549      00000D53 E80D010000      <1> call     _KIO_S_XLAT      ; ROUTINE TO XLATE FOR STANDARD CALLS
2550      00000D58 7308      <1> jnc     short _K2A      ; CARRY CLEAR MEANS PASS VALID CODE
2551      00000D5A 9D      <1> popf      ; INVALID CODE FOR THIS TYPE OF CALL
2552      00000D5B E880000000      <1> call     _K1S      ; THROW THE CHARACTER AWAY
2553      00000D60 EBE9      <1> jmp      short _K2      ; GO LOOK FOR NEXT CHAR, IF ANY
2554      <1> _K2A:
2555      00000D62 9D      <1> popf      ; RESTORE ZF FROM TEST
2556      <1> _K2B:
2557      <1> ;pop ecx      ; RECOVER REGISTER
2558      00000D63 5B      <1> pop      ebx      ; RECOVER REGISTER
2559      00000D64 1F      <1> pop      ds      ; RECOVER SEGMENT
2560      00000D65 CA0400      <1> retf      4      ; THROW AWAY (e)FLAGS
2561      <1>
2562      <1> ;----- SHIFT STATUS
2563      <1> _K3E:      ; GET THE EXTENDED SHIFT STATUS FLAGS
2564      00000D68 8A25[946A0000]      <1> mov      ah, [KB_FLAG_1]      ; GET SYSTEM SHIFT KEY STATUS
2565      00000D6E 80E404      <1> and      ah, SYS_SHIFT      ; MASK ALL BUT SYS KEY BIT
2566      <1> ;mov cl, 5      ; SHIFT THEW SYSTEMKEY BIT OVER TO
2567      <1> ;shl ah, cl      ; BIT 7 POSITION
2568      00000D71 C0E405      <1> shl      ah, 5
2569      00000D74 A0[946A0000]      <1> mov      al, [KB_FLAG_1]      ; GET SYSTEM SHIFT STATES BACK
2570      00000D79 2473      <1> and      al, 01110011b      ; ELIMINATE SYS SHIFT, HOLD_STATE AND INS_SHIFT
2571      00000D7B 08C4      <1> or      ah, al      ; MERGE REMAINING BITS INTO AH

```

2572	00000D7D	A0[966A0000]	<1>	mov	al, [KB_FLAG_3]	; GET RIGHT CTL AND ALT
2573	00000D82	240C	<1>	and	al, 00001100b	; ELIMINATE LC_E0 AND LC_E1
2574	00000D84	08C4	<1>	or	ah, al	; OR THE SHIFT FLAGS TOGETHER
2575			<1>	_K3:		
2576	00000D86	A0[936A0000]	<1>	mov	al, [KB_FLAG]	; GET THE SHIFT STATUS FLAGS
2577	00000D8B	EB92	<1>	jmp	short _KIO_EXIT	; RETURN TO CALLER
2578			<1>			
2579			<1>	;-----	SET TYPAMATIC RATE AND DELAY	
2580			<1>	_K300:		
2581	00000D8D	3C05	<1>	cmp	al, 5	; CORRECT FUNCTION CALL?
2582	00000D8F	758E	<1>	jne	short _KIO_EXIT	; NO, RETURN
2583	00000D91	F6C3E0	<1>	test	bl, 0E0h	; TEST FOR OUT-OF-RANGE RATE
2584	00000D94	7589	<1>	jnz	short _KIO_EXIT	; RETURN IF SO
2585	00000D96	F6C7FC	<1>	test	BH, 0FCh	; TEST FOR OUT-OF-RANGE DELAY
2586	00000D99	7584	<1>	jnz	short _KIO_EXIT	; RETURN IF SO
2587	00000D9B	B0F3	<1>	mov	al, KB_TYP_A_RD	; COMMAND FOR TYPAMATIC RATE/DELAY
2588	00000D9D	E8B6060000	<1>	call	SND_DATA	; SEND TO KEYBOARD
2589			<1>	;mov	cx, 5	; SHIFT COUNT
2590			<1>	;shl	bh, cl	; SHIFT DELAY OVER
2591	00000DA2	C0E705	<1>	shl	bh, 5	
2592	00000DA5	88D8	<1>	mov	al, bl	; PUT IN RATE
2593	00000DA7	08F8	<1>	or	al, bh	; AND DELAY
2594	00000DA9	E8AA060000	<1>	call	SND_DATA	; SEND TO KEYBOARD
2595	00000DAE	E96CFFFFFF	<1>	jmp	_KIO_EXIT	; RETURN TO CALLER
2596			<1>			
2597			<1>	;-----	WRITE TO KEYBOARD BUFFER	
2598			<1>	_K500:		
2599	00000DB3	56	<1>	push	esi	; SAVE SI (esi)
2600	00000DB4	FA	<1>	cli		
2601	00000DB5	8B1D[A46A0000]	<1>	mov	ebx, [BUFFER_TAIL]	; GET THE 'IN TO' POINTER TO THE BUFFER
2602	00000DBB	89DE	<1>	mov	esi, ebx	; SAVE A COPY IN CASE BUFFER NOT FULL
2603	00000DBD	E8D3000000	<1>	call	_K4	; BUMP THE POINTER TO SEE IF BUFFER IS FULL
2604	00000DC2	3B1D[A06A0000]	<1>	cmp	ebx, [BUFFER_HEAD]	; WILL THE BUFFER OVERRUN IF WE STORE THIS?
2605	00000DC8	740D	<1>	je	short _K502	; YES - INFORM CALLER OF ERROR
2606	00000DCA	66890E	<1>	mov	[esi], cx	; NO - PUT ASCII/SCAN CODE INTO BUFFER
2607	00000DCD	891D[A46A0000]	<1>	mov	[BUFFER_TAIL], ebx	; ADJUST 'IN TO' POINTER TO REFLECT CHANGE
2608	00000DD3	28C0	<1>	sub	al, al	; TELL CALLER THAT OPERATION WAS SUCCESSFUL
2609	00000DD5	EB02	<1>	jmp	short _K504	; SUB INSTRUCTION ALSO RESETS CARRY FLAG
2610			<1>	_K502:		
2611	00000DD7	B001	<1>	mov	al, 01h	; BUFFER FULL INDICATION
2612			<1>	_K504:		
2613	00000DD9	FB	<1>	sti		
2614	00000DDA	5E	<1>	pop	esi	; RECOVER SI (esi)
2615	00000ddb	E93FFFFFFF	<1>	jmp	_KIO_EXIT	; RETURN TO CALLER WITH STATUS IN AL
2616			<1>			
2617			<1>	;-----	READ THE KEY TO FIGURE OUT WHAT TO DO -----	
2618			<1>	_K1S:		
2619	00000DE0	FA	<1>	cli	; 03/12/2014	
2620	00000DE1	8B1D[A06A0000]	<1>	mov	ebx, [BUFFER_HEAD]	; GET POINTER TO HEAD OF BUFFER
2621	00000DE7	3B1D[A46A0000]	<1>	cmp	ebx, [BUFFER_TAIL]	; TEST END OF BUFFER
2622			<1>	;jne	short _K1U	; IF ANYTHING IN BUFFER SKIP INTERRUPT
2623	00000DED	750F	<1>	jne	short _K1x ; 03/12/2014	
2624			<1>			
2625			<1>		; 03/12/2014	
2626			<1>		; 28/08/2014	
2627			<1>		; PERFORM OTHER FUNCTION ?? here !	
2628			<1>	;; MOV	AX, 9002h	; MOVE IN WAIT CODE & TYPE
2629			<1>	;; INT	15H	; PERFORM OTHER FUNCTION
2630			<1>	_K1T:		; ASCII READ
2631	00000DEF	FB	<1>	sti		; INTERRUPTS BACK ON DURING LOOP
2632	00000DF0	90	<1>	nop		; ALLOW AN INTERRUPT TO OCCUR
2633			<1>	_K1U:		
2634	00000DF1	FA	<1>	cli		; INTERRUPTS BACK OFF
2635	00000DF2	8B1D[A06A0000]	<1>	mov	ebx, [BUFFER_HEAD]	; GET POINTER TO HEAD OF BUFFER
2636	00000DF8	3B1D[A46A0000]	<1>	cmp	ebx, [BUFFER_TAIL]	; TEST END OF BUFFER
2637			<1>	_K1x:		
2638	00000DFE	53	<1>	push	ebx	; SAVE ADDRESS
2639	00000DFF	9C	<1>	pushf		; SAVE FLAGS
2640	00000E00	E80B070000	<1>	call	MAKE_LED	; GO GET MODE INDICATOR DATA BYTE
2641	00000E05	8A1D[956A0000]	<1>	mov	bl, [KB_FLAG_2]	; GET PREVIOUS BITS
2642	00000E0B	30C3	<1>	xor	bl, al	; SEE IF ANY DIFFERENT
2643	00000E0D	80E307	<1>	and	bl, 07h	; KB_LEDS ; ISOLATE INDICATOR BITS
2644	00000E10	7406	<1>	jz	short _K1V	; IF NO CHANGE BYPASS UPDATE
2645	00000E12	E8A5060000	<1>	call	SND_LED1	
2646	00000E17	FA	<1>	cli		; DISABLE INTERRUPTS
2647			<1>	_K1V:		
2648	00000E18	9D	<1>	popf		; RESTORE FLAGS
2649	00000E19	5B	<1>	pop	ebx	; RESTORE ADDRESS
2650	00000E1A	74D3	<1>	je	short _K1T	; LOOP UNTIL SOMETHING IN BUFFER
2651			<1>			
2652	00000E1C	668B03	<1>	mov	ax, [ebx]	; GET SCAN CODE AND ASCII CODE
2653	00000E1F	E871000000	<1>	call	_K4	; MOVE POINTER TO NEXT POSITION
2654	00000E24	891D[A06A0000]	<1>	mov	[BUFFER_HEAD], ebx	; STORE VALUE IN VARIABLE
2655	00000E2A	C3	<1>	ret		; RETURN
2656			<1>			
2657			<1>	;-----	READ THE KEY TO SEE IF ONE IS PRESENT -----	
2658			<1>	_K2S:		
2659	00000E2B	FA	<1>	cli		; INTERRUPTS OFF
2660	00000E2C	8B1D[A06A0000]	<1>	mov	ebx, [BUFFER_HEAD]	; GET HEAD POINTER
2661	00000E32	3B1D[A46A0000]	<1>	cmp	ebx, [BUFFER_TAIL]	; IF EQUAL (Z=1) THEN NOTHING THERE
2662	00000E38	668B03	<1>	mov	ax, [ebx]	
2663	00000E3B	9C	<1>	pushf		; SAVE FLAGS

```

2664 00000E3C 6650      <1>  push  ax                ; SAVE CODE
2665 00000E3E E8CD060000 <1>  call  MAKE_LED           ; GO GET MODE INDICATOR DATA BYTE
2666 00000E43 8A1D[956A0000] <1>  mov   bl, [KB_FLAG_2]    ; GET PREVIOUS BITS
2667 00000E49 30C3      <1>  xor    bl, al             ; SEE IF ANY DIFFERENT
2668 00000E4B 80E307     <1>  and    bl, 07h ; KB_LEDS ; ISOLATE INDICATOR BITS
2669 00000E4E 7405      <1>  jz     short _K2T       ; IF NO CHANGE BYPASS UPDATE
2670 00000E50 E850060000 <1>  call  SND_LED           ; GO TURN ON MODE INDICATORS
2671                                <1>  _K2T:
2672 00000E55 6658      <1>  pop    ax                ; RESTORE CODE
2673 00000E57 9D         <1>  popf   ; RESTORE FLAGS
2674 00000E58 FB         <1>  sti     ; INTERRUPTS BACK ON
2675 00000E59 C3         <1>  retn   ; RETURN
2676                                <1>
2677                                <1> ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----
2678                                <1> _KIO_E_XLAT:
2679 00000E5A 3CF0      <1>  cmp    al, 0F0h        ; IS IT ONE OF THE FILL-INS?
2680 00000E5C 7506      <1>  jne    short _KIO_E_RET ; NO, PASS IT ON
2681 00000E5E 08E4      <1>          or    ah, ah        ; AH = 0 IS SPECIAL CASE
2682 00000E60 7402      <1>          jz     short _KIO_E_RET ; PASS THIS ON UNCHANGED
2683 00000E62 30C0      <1>  xor    al, al            ; OTHERWISE SET AL = 0
2684                                <1> _KIO_E_RET:
2685 00000E64 C3         <1>  retn   ; GO BACK
2686                                <1>
2687                                <1> ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----
2688                                <1> _KIO_S_XLAT:
2689 00000E65 80FCE0     <1>  cmp    ah, 0E0h        ; IS IT KEYPAD ENTER OR / ?
2690 00000E68 750F      <1>  jne    short _KIO_S2      ; NO, CONTINUE
2691 00000E6A 3C0D      <1>  cmp    al, 0Dh          ; KEYPAD ENTER CODE?
2692 00000E6C 7408      <1>          je     short _KIO_S1      ; YES, MESSAGE A BIT
2693 00000E6E 3C0A      <1>  cmp    al, 0Ah          ; CTRL KEYPAD ENTER CODE?
2694 00000E70 7404      <1>          je     short _KIO_S1      ; YES, MESSAGE THE SAME
2695 00000E72 B435      <1>  mov    ah, 35h          ; NO, MUST BE KEYPAD /
2696                                <1> _kio_ret: ; 03/12/2014
2697 00000E74 F8         <1>  cld
2698 00000E75 C3         <1>  retn
2699                                <1> ;jmp short _KIO_USE ; GIVE TO CALLER
2700                                <1> _KIO_S1:
2701 00000E76 B41C      <1>  mov    ah, 1Ch          ; CONVERT TO COMPATIBLE OUTPUT
2702                                <1> ;jmp short _KIO_USE ; GIVE TO CALLER
2703 00000E78 C3         <1>  retn
2704                                <1> _KIO_S2:
2705 00000E79 80FC84     <1>  cmp    ah, 84h          ; IS IT ONE OF EXTENDED ONES?
2706 00000E7C 7715      <1>  ja     short _KIO_DIS      ; YES, THROW AWAY AND GET ANOTHER CHAR
2707 00000E7E 3CF0      <1>  cmp    al, 0F0h        ; IS IT ONE OF THE FILL-INS?
2708 00000E80 7506      <1>  jne    short _KIO_S3      ; NO, TRY LAST TEST
2709 00000E82 08E4      <1>  or     ah, ah            ; AH = 0 IS SPECIAL CASE
2710 00000E84 740C      <1>          jz     short _KIO_USE      ; PASS THIS ON UNCHANGED
2711 00000E86 EB0B      <1>  jmp    short _KIO_DIS      ; THROW AWAY THE REST
2712                                <1> _KIO_S3:
2713 00000E88 3CE0      <1>  cmp    al, 0E0h        ; IS IT AN EXTENSION OF A PREVIOUS ONE?
2714                                <1> ;jne short _KIO_USE ; NO, MUST BE A STANDARD CODE
2715 00000E8A 75E8      <1>  jne    short _kio_ret
2716 00000E8C 08E4      <1>  or     ah, ah            ; AH = 0 IS SPECIAL CASE
2717 00000E8E 7402      <1>          jz     short _KIO_USE      ; JUMP IF AH = 0
2718 00000E90 30C0      <1>  xor    al, al            ; CONVERT TO COMPATIBLE OUTPUT
2719                                <1> ;jmp short _KIO_USE ; PASS IT ON TO CALLER
2720                                <1> _KIO_USE:
2721                                <1> ;cld ; CLEAR CARRY TO INDICATE GOOD CODE
2722 00000E92 C3         <1>  retn   ; RETURN
2723                                <1> _KIO_DIS:
2724 00000E93 F9         <1>  stc     ; SET CARRY TO INDICATE DISCARD CODE
2725 00000E94 C3         <1>  retn   ; RETURN
2726                                <1>
2727                                <1> ;----- INCREMENT BUFFER POINTER ROUTINE -----
2728                                <1> _K4:
2729 00000E95 43         <1>  inc     ebx
2730 00000E96 43         <1>  inc     ebx ; MOVE TO NEXT WORD IN LIST
2731 00000E97 3B1D[9C6A0000] <1>  cmp     ebx, [BUFFER_END] ; AT END OF BUFFER?
2732                                <1> ;jne short _K5 ; NO, CONTINUE
2733 00000E9D 7206      <1>  jb     short _K5
2734 00000E9F 8B1D[986A0000] <1>  mov     ebx, [BUFFER_START] ; YES, RESET TO BUFFER BEGINNING
2735                                <1> _K5:
2736 00000EA5 C3         <1>  retn
2737                                <1>
2738                                <1> ; 20/02/2015
2739                                <1> ; 05/12/2014
2740                                <1> ; 26/08/2014
2741                                <1> ; KEYBOARD (HARDWARE) INTERRUPT - IRQ LEVEL 1
2742                                <1> ; (INT_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)
2743                                <1> ;
2744                                <1> ; Derived from "KB_INT_1" procedure of IBM "pc-at"
2745                                <1> ; rombios source code (06/10/1985)
2746                                <1> ; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)
2747                                <1>
2748                                <1> ;----- 8042 COMMANDS -----
2749                                <1> ENA_KBD equ 0AEh ; ENABLE KEYBOARD COMMAND
2750                                <1> DIS_KBD equ 0ADh ; DISABLE KEYBOARD COMMAND
2751                                <1> SHUT_CMD equ 0FEh ; CAUSE A SHUTDOWN COMMAND
2752                                <1> ;----- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----
2753                                <1> STATUS_PORT equ 064h ; 8042 STATUS PORT
2754                                <1> INPT_BUF_FULLequ 00000010b ; 1 = +INPUT BUFFER FULL
2755                                <1> PORT_A equ 060h ; 8042 KEYBOARD SCAN CODE/CONTROL PORT

```

```

2756 <1> ;----- 8042 KEYBOARD RESPONSE -----
2757 <1> KB_ACK equ 0FAh ; ACKNOWLEDGE PROM TRANSMISSION
2758 <1> KB_RESEND equ 0FEh ; RESEND REQUEST
2759 <1> KB_OVER_RUN equ 0FFh ; OVER RUN SCAN CODE
2760 <1> ;----- KEYBOARD/LED COMMANDS -----
2761 <1> KB_ENABLE equ 0F4h ; KEYBOARD ENABLE
2762 <1> LED_CMD equ 0EDh ; LED WRITE COMMAND
2763 <1> KB_TYPA_RD equ 0F3h ; TYPAMATIC RATE/DELAY COMMAND
2764 <1> ;----- KEYBOARD SCAN CODES -----
2765 <1> NUM_KEY equ 69 ; SCAN CODE FOR NUMBER LOCK KEY
2766 <1> SCROLL_KEY equ 70 ; SCAN CODE FOR SCROLL LOCK KEY
2767 <1> ALT_KEY equ 56 ; SCAN CODE FOR ALTERNATE SHIFT KEY
2768 <1> CTL_KEY equ 29 ; SCAN CODE FOR CONTROL KEY
2769 <1> CAPS_KEY equ 58 ; SCAN CODE FOR SHIFT LOCK KEY
2770 <1> DEL_KEY equ 83 ; SCAN CODE FOR DELETE KEY
2771 <1> INS_KEY equ 82 ; SCAN CODE FOR INSERT KEY
2772 <1> LEFT_KEY equ 42 ; SCAN CODE FOR LEFT SHIFT
2773 <1> RIGHT_KEY equ 54 ; SCAN CODE FOR RIGHT SHIFT
2774 <1> SYS_KEY equ 84 ; SCAN CODE FOR SYSTEM KEY
2775 <1> ;----- ENHANCED KEYBOARD SCAN CODES -----
2776 <1> ID_1 equ 0ABh ; 1ST ID CHARACTER FOR KBX
2777 <1> ID_2 equ 041h ; 2ND ID CHARACTER FOR KBX
2778 <1> ID_2A equ 054h ; ALTERNATE 2ND ID CHARACTER FOR KBX
2779 <1> F11_M equ 87 ; F11 KEY MAKE
2780 <1> F12_M equ 88 ; F12 KEY MAKE
2781 <1> MC_E0 equ 224 ; GENERAL MARKER CODE
2782 <1> MC_E1 equ 225 ; PAUSE KEY MARKER CODE
2783 <1> ;----- FLAG EQUATES WITHIN @KB_FLAG-----
2784 <1> RIGHT_SHIFT equ 00000001b ; RIGHT SHIFT KEY DEPRESSED
2785 <1> LEFT_SHIFT equ 00000010b ; LEFT SHIFT KEY DEPRESSED
2786 <1> CTL_SHIFT equ 00000100b ; CONTROL SHIFT KEY DEPRESSED
2787 <1> ALT_SHIFT equ 00001000b ; ALTERNATE SHIFT KEY DEPRESSED
2788 <1> SCROLL_STATE equ 00010000b ; SCROLL LOCK STATE IS ACTIVE
2789 <1> NUM_STATE equ 00100000b ; NUM LOCK STATE IS ACTIVE
2790 <1> CAPS_STATE equ 01000000b ; CAPS LOCK STATE IS ACTIVE
2791 <1> INS_STATE equ 10000000b ; INSERT STATE IS ACTIVE
2792 <1> ;----- FLAG EQUATES WITHIN @KB_FLAG_1 -----
---
2793 <1> L_CTL_SHIFT equ 00000001b ; LEFT CTL KEY DOWN
2794 <1> L_ALT_SHIFT equ 00000010b ; LEFT ALT KEY DOWN
2795 <1> SYS_SHIFT equ 00000100b ; SYSTEM KEY DEPRESSED AND HELD
2796 <1> HOLD_STATE equ 00001000b ; SUSPEND KEY HAS BEEN TOGGLED
2797 <1> SCROLL_SHIFT equ 00010000b ; SCROLL LOCK KEY IS DEPRESSED
2798 <1> NUM_SHIFT equ 00100000b ; NUM LOCK KEY IS DEPRESSED
2799 <1> CAPS_SHIFT equ 01000000b ; CAPS LOCK KEY IS DEPRESSED
2800 <1> INS_SHIFT equ 10000000b ; INSERT KEY IS DEPRESSED
2801 <1> ;----- FLAGS EQUATES WITHIN @KB_FLAG_2 -----
2802 <1> KB_LEDS equ 00000111b ; KEYBOARD LED STATE BITS
2803 <1> ; equ 00000001b ; SCROLL LOCK INDICATOR
2804 <1> ; equ 00000010b ; NUM LOCK INDICATOR
2805 <1> ; equ 00000100b ; CAPS LOCK INDICATOR
2806 <1> ; equ 00001000b ; RESERVED (MUST BE ZERO)
2807 <1> KB_FA equ 00010000b ; ACKNOWLEDGMENT RECEIVED
2808 <1> KB_FE equ 00100000b ; RESEND RECEIVED FLAG
2809 <1> KB_PR_LED equ 01000000b ; MODE INDICATOR UPDATE
2810 <1> KB_ERR equ 10000000b ; KEYBOARD TRANSMIT ERROR FLAG
2811 <1> ;----- FLAGS EQUATES WITHIN @KB_FLAG_3 -----
2812 <1> LC_E1 equ 00000001b ; LAST CODE WAS THE E1 HIDDEN CODE
2813 <1> LC_E0 equ 00000010b ; LAST CODE WAS THE E0 HIDDEN CODE
2814 <1> R_CTL_SHIFT equ 00000100b ; RIGHT CTL KEY DOWN
2815 <1> R_ALT_SHIFT equ 00001000b ; RIGHT ALT KEY DOWN
2816 <1> GRAPH_ON equ 00001000b ; ALT GRAPHICS KEY DOWN (WT ONLY)
2817 <1> KBX equ 00010000b ; ENHANCED KEYBOARD INSTALLED
2818 <1> SET_NUM_LK equ 00100000b ; FORCE NUM LOCK IF READ ID AND KBX
2819 <1> LC_AB equ 01000000b ; LAST CHARACTER WAS FIRST ID CHARACTER
2820 <1> RD_ID equ 10000000b ; DOING A READ ID (MUST BE BIT0)
2821 <1> ;
2822 <1> ;----- INTERRUPT EQUATES -----
2823 <1> EOI equ 020h ; END OF INTERRUPT COMMAND TO 8259
2824 <1> INTA00 equ 020h ; 8259 PORT
2825 <1>
2826 <1>
2827 <1> kb_int:
2828 <1>
2829 <1> ; 17/10/2015 ('ctrlbrk')
2830 <1> ; 05/12/2014
2831 <1> ; 04/12/2014 (derivation from pc-xt-286 bios source code -1986-,
2832 <1> ; instead of pc-at bios - 1985-)
2833 <1> ; 26/08/2014
2834 <1> ;
2835 <1> ; 03/06/86 KEYBOARD BIOS
2836 <1> ;
2837 <1> ;--- HARDWARE INT 09H -- (IRQ LEVEL 1) -----
2838 <1> ;
2839 <1> ; KEYBOARD INTERRUPT ROUTINE
2840 <1> ;
2841 <1> ;-----
2842 <1>
2843 <1> KB_INT_1:
2844 00000EA6 FB <1> sti ; ENABLE INTERRUPTS
2845 <1> ;push ebp
2846 00000EA7 50 <1> push eax

```

```

2847 00000EA8 53      <1>  push  ebx
2848 00000EA9 51      <1>  push  ecx
2849 00000EAA 52      <1>  push  edx
2850 00000EAB 56      <1>  push  esi
2851 00000EAC 57      <1>  push  edi
2852 00000EAD 1E      <1>  push  ds
2853 00000EAE 06      <1>  push  es
2854 00000EAF FC      <1>  cld                      ; FORWARD DIRECTION
2855 00000EB0 66B81000 <1>  mov   ax, KDATA
2856 00000EB4 8ED8     <1>  mov   ds, ax
2857 00000EB6 8EC0     <1>  mov   es, ax
2858                      <1>  ;
2859                      <1>  ;----- WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
2860 00000EB8 B0AD     <1>  mov   al, DIS_KBD          ; DISABLE THE KEYBOARD COMMAND
2861 00000EBA E885050000 <1>  call  SHIP_IT             ; EXECUTE DISABLE
2862 00000EBF FA       <1>  cli                      ; DISABLE INTERRUPTS
2863 00000EC0 B900000100 <1>  mov   ecx, 10000h         ; SET MAXIMUM TIMEOUT
2864                      <1>  KB_INT_01:
2865 00000EC5 E464     <1>  in    al, STATUS_PORT      ; READ ADAPTER STATUS
2866 00000EC7 A802     <1>  test  al, INPT_BUF_FULL ; CHECK INPUT BUFFER FULL STATUS BIT
2867 00000EC9 E0FA     <1>  loopnz KB_INT_01         ; WAIT FOR COMMAND TO BE ACCEPTED
2868                      <1>  ;
2869                      <1>  ;----- READ CHARACTER FROM KEYBOARD INTERFACE
2870 00000ECB E460     <1>  in    al, PORT_A          ; READ IN THE CHARACTER
2871                      <1>  ;
2872                      <1>  ;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INT LEVEL 9H)
2873                      <1>  ;MOV  AH, 04FH          ; SYSTEM INTERCEPT - KEY CODE FUNCTION
2874                      <1>  ;STC                      ; SET CY=1 (IN CASE OF IRET)
2875                      <1>  ;INT  15H          ; CASSETTE CALL (AL)=KEY SCAN CODE
2876                      <1>  ;                      ; RETURNS CY=1 FOR INVALID FUNCTION
2877                      <1>  ;JC   KB_INT_02      ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
2878                      <1>  ;JMP  K26          ; EXIT IF SYSTEM HANDLES SCAN CODE
2879                      <1>  ;                      ; EX|T HANDLES HARDWARE EOI AND ENABLE
2880                      <1>  ;
2881                      <1>  ;----- CHECK FOR A RESEND COMMAND TO KEYBOARD
2882                      <1>  KB_INT_02:          ; (AL)= SCAN CODE
2883 00000ECD FB       <1>  sti                      ; ENABLE INTERRUPTS AGAIN
2884 00000ECE 3CFE     <1>  cmp    al, KB_RESEND      ; IS THE INPUT A RESEND
2885 00000ED0 7411     <1>  je      short KB_INT_4      ; GO IF RESEND
2886                      <1>  ;
2887                      <1>  ;----- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
2888 00000ED2 3CFA     <1>  cmp    al, KB_ACK          ; IS THE INPUT AN ACKNOWLEDGE
2889 00000ED4 751A     <1>  jne     short KB_INT_2      ; GO IF NOT
2890                      <1>  ;
2891                      <1>  ;----- A COMMAND TO THE KEYBOARD WAS ISSUED
2892 00000ED6 FA       <1>  cli                      ; DISABLE INTERRUPTS
2893 00000ED7 800D[956A0000]10 <1>  or     byte [KB_FLAG_2], KB_FA ; INDICATE ACK RECEIVED
2894 00000EDE E97A020000 <1>  jmp     K26                ; RETURN IF NOT (ACK RETURNED FOR DATA)
2895                      <1>  ;
2896                      <1>  ;----- RESEND THE LAST BYTE
2897                      <1>  KB_INT_4:
2898 00000EE3 FA       <1>  cli                      ; DISABLE INTERRUPTS
2899 00000EE4 800D[956A0000]20 <1>  or     byte [KB_FLAG_2], KB_FE ; INDICATE RESEND RECEIVED
2900 00000EEB E96D020000 <1>  jmp     K26                ; RETURN IF NOT ACK RETURNED FOR DATA)
2901                      <1>  ;
2902                      <1>  ;----- UPDATE MODE INDICATORS IF CHANGE IN STATE
2903                      <1>  KB_INT_2:
2904 00000EF0 6650     <1>  push  ax                ; SAVE DATA IN
2905 00000EF2 E819060000 <1>  call  MAKE_LED           ; GO GET MODE INDICATOR DATA BYTE
2906 00000EF7 8A1D[956A0000] <1>  mov   bl, [KB_FLAG_2]    ; GET PREVIOUS BITS
2907 00000EFD 30C3     <1>  xor    bl, al            ; SEE IF ANY DIFFERENT
2908 00000EFF 80E307    <1>  and    bl, KB_LEDS       ; ISOLATE INDICATOR BITS
2909 00000F02 7405     <1>  jz     short UP0        ; IF NO CHANGE BYPASS UPDATE
2910 00000F04 E89C050000 <1>  call  SND_LED           ; GO TURN ON MODE INDICATORS
2911                      <1>  UP0:
2912 00000F09 6658     <1>  pop   ax                ; RESTORE DATA IN
2913                      <1>  ;-----
2914                      <1>  ; START OF KEY PROCESSING
2915                      <1>  ;-----
2916 00000F0B 88C4     <1>  mov   ah, al            ; SAVE SCAN CODE IN AH ALSO
2917                      <1>  ;
2918                      <1>  ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
2919 00000F0D 3CFF     <1>  cmp    al, KB_OVER_RUN    ; IS THIS AN OVERRUN CHAR
2920 00000F0F 0F841B050000 <1>  je     K62                ; BUFFER_FULL_BEEP
2921                      <1>  ;
2922                      <1>  K16:
2923 00000F15 8A3D[966A0000] <1>  mov   bh, [KB_FLAG_3]    ; LOAD FLAGS FOR TESTING
2924                      <1>  ;
2925                      <1>  ;----- TEST TO SEE IF A READ_ID IS IN PROGRESS
2926 00000F1B F6C7C0    <1>  test  bh, RD_ID+LC_AB    ; ARE WE DOING A READ ID?
2927 00000F1E 7449     <1>  jz     short NOT_ID      ; CONTINUE IF NOT
2928 00000F20 7917     <1>  jns    short TST_ID_2    ; IS THE RD_ID FLAG ON?
2929 00000F22 3CAB     <1>  cmp    al, ID_1          ; IS THIS THE 1ST ID CHARACTER?
2930 00000F24 7507     <1>  jne    short RST_RD_ID   ;
2931 00000F26 800D[966A0000]40 <1>  or     byte [KB_FLAG_3], LC_AB ; INDICATE 1ST ID WAS OK
2932                      <1>  RST_RD_ID:
2933 00000F2D 8025[966A0000]7F <1>  and    byte [KB_FLAG_3], ~RD_ID ; RESET THE READ ID FLAG
2934                      <1>  ;jmp    short ID_EX          ; AND EXIT
2935 00000F34 E924020000 <1>  jmp    K26
2936                      <1>  ;
2937                      <1>  TST_ID_2:
2938 00000F39 8025[966A0000]BF <1>  and    byte [KB_FLAG_3], ~LC_AB ; RESET FLAG

```



```

2939 00000F40 3C54      <1>  cmp    al, ID_2A      ; IS THIS THE 2ND ID CHARACTER?
2940 00000F42 7419      <1>      je     short KX_BIT      ; JUMP IF SO
2941 00000F44 3C41      <1>  cmp    al, ID_2      ; IS THIS THE 2ND ID CHARACTER?
2942                        <1>      ;jne short ID_EX      ; LEAVE IF NOT
2943 00000F46 0F8511020000 <1>  jne     K26
2944                        <1>      ;
2945                        <1>  ;----- A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
2946 00000F4C F6C720      <1>  test   bh, SET_NUM_LK      ; SHOULD WE SET NUM LOCK?
2947 00000F4F 740C      <1>      jz     short KX_BIT      ; EXIT IF NOT
2948 00000F51 800D[936A0000]20 <1>  or     byte [KB_FLAG_3], NUM_STATE ; FORCE NUM LOCK ON
2949 00000F58 E848050000      <1>  call   SND_LED      ; GO SET THE NUM LOCK INDICATOR
2950                        <1>  KX_BIT:
2951 00000F5D 800D[966A0000]10 <1>  or     byte [KB_FLAG_3], KBX ; INDICATE ENHANCED KEYBOARD WAS FOUND
2952 00000F64 E9F4010000      <1>  ID_EX: jmp     K26      ; EXIT
2953                        <1>      ;
2954                        <1>  NOT_ID:
2955 00000F69 3CE0      <1>  cmp    al, MC_E0      ; IS THIS THE GENERAL MARKER CODE?
2956 00000F6B 750C      <1>      jne     short TEST_E1      ;
2957 00000F6D 800D[966A0000]12 <1>  or     byte [KB_FLAG_3], LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
2958                        <1>      ;jmp short EXIT      ; THROW AWAY THIS CODE
2959 00000F74 E9EB010000      <1>  jmp     K26A
2960                        <1>  TEST_E1:
2961 00000F79 3CE1      <1>  cmp    al, MC_E1      ; IS THIS THE PAUSE KEY?
2962 00000F7B 750C      <1>      jne     short NOT_HC      ;
2963 00000F7D 800D[966A0000]11 <1>  or     byte [KB_FLAG_3], LC_E1+KBX ; SET FLAG BIT, SET KBX, AND
2964 00000F84 E9DB010000      <1>  EXIT: jmp     K26A      ; THROW AWAY THIS CODE
2965                        <1>      ;
2966                        <1>  NOT_HC:
2967 00000F89 247F      <1>  and    al, 07Fh      ; TURN OFF THE BREAK BIT
2968 00000F8B F6C702      <1>  test   bh, LC_E0      ; LAST CODE THE E0 MARKER CODE
2969 00000F8E 7414      <1>      jz     short NOT_LC_E0      ; JUMP IF NOT
2970                        <1>      ;
2971 00000F90 BF[7E690000]      <1>  mov     edi, _K6+6      ; IS THIS A SHIFT KEY?
2972 00000F95 AE          <1>  scasb
2973 00000F96 0F84C1010000      <1>      je     K26 ; K16B      ; YES, THROW AWAY & RESET FLAG
2974 00000F9C AE          <1>  scasb
2975 00000F9D 757C      <1>      jne     short K16A      ; NO, CONTINUE KEY PROCESSING
2976                        <1>      ;jmp short K16B      ; YES, THROW AWAY & RESET FLAG
2977 00000F9F E9B9010000      <1>  jmp     K26
2978                        <1>      ;
2979                        <1>  NOT_LC_E0:
2980 00000FA4 F6C701      <1>  test   bh, LC_E1      ; LAST CODE THE E1 MARKER CODE?
2981 00000FA7 7435      <1>      jz     short T_SYS_KEY      ; JUMP IF NOT
2982 00000FA9 B904000000      <1>  mov     ecx, 4      ; LENGHT OF SEARCH
2983 00000FAE BF[7C690000]      <1>  mov     edi, _K6+4      ; IS THIS AN ALT, CTL, OR SHIFT?
2984 00000FB3 F2AE          <1>  repne scasb      ; CHECK IT
2985                        <1>      ;je short EXIT      ; THROW AWAY IF SO
2986 00000FB5 0F84A9010000      <1>  je     K26A
2987                        <1>      ;
2988 00000FBB 3C45      <1>  cmp    al, NUM_KEY      ; IS IT THE PAUSE KEY?
2989                        <1>      ;jne short K16B      ; NO, THROW AWAY & RESET FLAG
2990 00000FBD 0F859A010000      <1>  jne     K26
2991 00000FC3 F6C480      <1>  test   ah, 80h      ; YES, IS IT THE BREAK OF THE KEY?
2992                        <1>      ;jnz short K16B      ; YES, THROW THIS AWAY, TOO
2993 00000FC6 0F8591010000      <1>  jnz     K26
2994                        <1>      ; 20/02/2015
2995 00000FCC F605[946A0000]08 <1>  test   byte [KB_FLAG_1],HOLD_STATE ; NO, ARE WE PAUSED ALREADY?
2996                        <1>      ;jnz short K16B      ; YES, THROW AWAY
2997 00000FD3 0F8584010000      <1>  jnz     K26
2998 00000FD9 E9E1020000      <1>  jmp     K39P      ; NO, THIS IS THE REAL PAUSE STATE
2999                        <1>      ;
3000                        <1>  ;----- TEST FOR SYSTEM KEY
3001                        <1>  T_SYS_KEY:
3002 00000FDE 3C54      <1>  cmp    al, SYS_KEY      ; IS IT THE SYSTEM KEY?
3003 00000FE0 7539      <1>      jnz     short K16A      ; CONTINUE IF NOT
3004                        <1>      ;
3005 00000FE2 F6C480      <1>  test   ah, 80h      ; CHECK IF THIS A BREAK CODE
3006 00000FE5 7524      <1>      jnz     short K16C      ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE
3007                        <1>      ;
3008 00000FE7 F605[946A0000]04 <1>  test   byte [KB_FLAG_1], SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
3009                        <1>      ;jnz short K16B      ; IF YES, DO NOT PROCESS SYSTEM INDICATOR
3010 00000FEE 0F8569010000      <1>  jnz     K26
3011                        <1>      ;
3012 00000FF4 800D[946A0000]04 <1>  or     byte [KB_FLAG_1], SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
3013 00000FFB B020      <1>  mov     al, EOI      ; END OF INTERRUPT COMMAND
3014 00000FFD E620      <1>  out     20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
3015                        <1>      ; INTERRUPT-RETURN-NO-EOI
3016 00000FFF B0AE          <1>  mov     al, ENA_KBD      ; INSURE KEYBOARD IS ENABLED
3017 00001001 E83E040000      <1>  call   SHIP_IT      ; EXECUTE ENABLE
3018                        <1>  ; !!! SYSREQ !!! function/system call (INTERRUPT) must be here !!!
3019                        <1>  ;MOV AL, 8500H      ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
3020                        <1>  ;STI      ; MAKE SURE INTERRUPTS ENABLED
3021                        <1>  ;INT 15H      ; USER INTERRUPT
3022 00001006 E965010000      <1>      jmp     K27A      ; END PROCESSING
3023                        <1>      ;
3024                        <1>  ;K16B: jmp     K26      ; IGNORE SYSTEM KEY
3025                        <1>      ;
3026                        <1>  K16C:
3027 0000100B 8025[946A0000]FB <1>  and    byte [KB_FLAG_1], ~SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
3028 00001012 B020      <1>  mov     al, EOI      ; END OF INTERRUPT COMMAND
3029 00001014 E620      <1>  out     20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
3030                        <1>      ; INTERRUPT-RETURN-NO-EOI

```

```

3031      <1> ;MOV AL, ENA_KBD ; INSURE KEYBOARD IS ENABLED
3032      <1> ;CALL SHIP_IT ; EXECUTE ENABLE
3033      <1> ;
3034      <1> ;MOV AX, 8501H ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
3035      <1> ;STI ; MAKE SURE INTERRUPTS ENABLED
3036      <1> ;INT 15H ; USER INTERRUPT
3037      <1> ;JMP K27A ; INCONRE SYSTEM KEY
3038      <1> ;
3039      <1> jmp K27 ; IGNORE SYSTEM KEY
3040      <1> ;
3041      <1> ;----- TEST FOR SHIFT KEYS
3042      <1> K16A:
3043      <1> mov bl, [KB_FLAG] ; PUT STATE FLAGS IN BL
3044      <1> mov edi, _K6 ; SHIFT KEY TABLE offset
3045      <1> mov ecx, _K6L ; LENGTH
3046      <1> repne scasb ; LOOK THROUGH THE TABLE FOR A MATCH
3047      <1> mov al, ah ; RECOVER SCAN CODE
3048      <1> jne K25 ; IF NO MATCH, THEN SHIFT NOT FOUND
3049      <1> ;
3050      <1> ;----- SHIFT KEY FOUND
3051      <1> K17:
3052      <1> sub edi, _K6+1 ; ADJUST PTR TO SCAN CODE MATCH
3053      <1> mov ah, [edi+_K7] ; GET MASK INTO AH
3054      <1> mov cl, 2 ; SETUP COUNT FOR FLAG SHIFTS
3055      <1> test al, 80h ; TEST FOR BREAK KEY
3056      <1> jnz K23 ; JUMP OF BREAK
3057      <1> ;
3058      <1> ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
3059      <1> K17C:
3060      <1> cmp ah, SCROLL_SHIFT
3061      <1> jae short K18 ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
3062      <1> ;
3063      <1> ;----- PLAIN SHIFT KEY, SET SHIFT ON
3064      <1> or [KB_FLAG], ah ; TURN ON SHIFT BIT
3065      <1> test al, CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
3066      <1> ;jnz short K17D ; YES, MORE FLAGS TO SET
3067      <1> jz K26 ; NO, INTERRUPT RETURN
3068      <1> K17D:
3069      <1> test bh, LC_E0 ; IS THIS ONE OF NEW KEYS?
3070      <1> jz short K17E ; NO, JUMP
3071      <1> or [KB_FLAG_3], ah ; SET BITS FOR RIGHT CTRL, ALT
3072      <1> jmp K26 ; INTERRUPT RETURN
3073      <1> K17E:
3074      <1> shr ah, cl ; MOVE FLAG BITS TWO POSITIONS
3075      <1> or [KB_FLAG_1], ah ; SET BITS FOR LEFT CTRL, ALT
3076      <1> jmp K26
3077      <1> ;
3078      <1> ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
3079      <1> K18:
3080      <1> test bl, CTL_SHIFT ; CHECK CTL SHIFT STATE
3081      <1> ;jz short K18A ; JUMP IF NOT CTL STATE
3082      <1> jnz K25 ; JUMP IF CTL STATE
3083      <1> K18A:
3084      <1> cmp al, INS_KEY ; CHECK FOR INSERT KEY
3085      <1> jne short K22 ; JUMP IF NOT INSERT KEY
3086      <1> test bl, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
3087      <1> ;jz short K18B ; JUMP IF NOT ALTERNATE SHIFT
3088      <1> jnz K25 ; JUMP IF ALTERNATE SHIFT
3089      <1> K18B:
3090      <1> test bh, LC_E0 ;20/02/2015 ; IS THIS NEW INSERT KEY?
3091      <1> jnz short K22 ; YES, THIS ONE'S NEVER A '0'
3092      <1> K19:
3093      <1> test bl, NUM_STATE ; CHECK FOR BASE STATE
3094      <1> jnz short K21 ; JUMP IF NUM LOCK IS ON
3095      <1> test bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
3096      <1> jz short K22 ; JUMP IF BASE STATE
3097      <1> K20:
3098      <1> ; NUMERIC ZERO, NOT INSERT KEY
3099      <1> mov ah, al ; PUT SCAN CODE BACK IN AH
3100      <1> jmp K25 ; NUMERAL '0', STNDRD. PROCESSING
3101      <1> K21:
3102      <1> ; MIGHT BE NUMERIC
3103      <1> test bl, LEFT_SHIFT+RIGHT_SHIFT
3104      <1> jz short K20 ; IS NUMERIC, STD. PROC.
3105      <1> ;
3106      <1> K22:
3107      <1> ; SHIFT TOGGLE KEY HIT; PROCESS IT
3108      <1> test ah, [KB_FLAG_1] ; IS KEY ALREADY DEPRESSED
3109      <1> jnz K26 ; JUMP IF KEY ALREADY DEPRESSED
3110      <1> K22A:
3111      <1> or [KB_FLAG_1], ah ; INDICATE THAT THE KEY IS DEPRESSED
3112      <1> xor [KB_FLAG], ah ; TOGGLE THE SHIFT STATE
3113      <1> ;
3114      <1> ;----- TOGGLE LED IF CAPS, NUM OR SCROLL KEY DEPRESSED
3115      <1> test ah, CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
3116      <1> jz short K22B ; GO IF NOT
3117      <1> ;
3118      <1> push ax ; SAVE SCAN CODE AND SHIFT MASK
3119      <1> call SND_LED ; GO TURN MODE INDICATORS ON
3120      <1> pop ax ; RESTORE SCAN CODE
3121      <1> K22B:
3122      <1> cmp al, INS_KEY ; TEST FOR 1ST MAKE OF INSERT KEY
3123      <1> jne K26 ; JUMP IF NOT INSERT KEY
3124      <1> mov ah, al ; SCAN CODE IN BOTH HALVES OF AX
3125      <1> jmp K28 ; FLAGS UPDATED, PROC. FOR BUFFER

```

```

3123                                     <1> ;
3124                                     <1> ;----- BREAK SHIFT FOUND
3125                                     <1> K23: ; BREAK-SHIFT-FOUND
3126 000010E1 80FC10 <1> cmp ah, SCROLL_SHIFT ; IS THIS A TOGGLE KEY
3127 000010E4 F6D4 <1> not ah ; INVERT MASK
3128 000010E6 7355 <1> jae short K24 ; YES, HANDLE BREAK TOGGLE
3129 000010E8 2025[936A0000] <1> and [KB_FLAG], ah ; TURN OFF SHIFT BIT
3130 000010EE 80FCFB <1> cmp ah, ~CTL_SHIFT ; IS THIS ALT OR CTL?
3131 000010F1 7730 <1> ja short K23D ; NO, ALL DONE
3132 <1> ;
3133 000010F3 F6C702 <1> test bh, LC_E0 ; 2ND ALT OR CTL?
3134 000010F6 7408 <1> jz short K23A ; NO, HANSLE NORMALLY
3135 000010F8 2025[966A0000] <1> and [KB_FLAG_3], ah ; RESET BIT FOR RIGHT ALT OR CTL
3136 000010FE EB08 <1> jmp short K23B ; CONTINUE
3137 <1> K23A:
3138 00001100 D2FC <1> sar ah, cl ; MOVE THE MASK BIT TWO POSITIONS
3139 00001102 2025[946A0000] <1> and [KB_FLAG_1], ah ; RESET BIT FOR LEFT ALT AND CTL
3140 <1> K23B:
3141 00001108 88C4 <1> mov ah, al ; SAVE SCAN CODE
3142 0000110A A0[966A0000] <1> mov al, [KB_FLAG_3] ; GET RIGHT ALT & CTRL FLAGS
3143 0000110F D2E8 <1> shr al, cl ; MOVE TO BITS 1 & 0
3144 00001111 0A05[946A0000] <1> or al, [KB_FLAG_1] ; PUT IN LEFT ALT & CTL FLAGS
3145 00001117 D2E0 <1> shl al, cl ; MOVE BACK TO BITS 3 & 2
3146 00001119 240C <1> and al, ALT_SHIFT+CTL_SHIFT ; FILTER OUT OTHER GARBAGE
3147 0000111B 0805[936A0000] <1> or [KB_FLAG], al ; PUT RESULT IN THE REAL FLAGS
3148 00001121 88E0 <1> mov al, ah
3149 <1> K23D:
3150 00001123 3CB8 <1> cmp al, ALT_KEY+80h ; IS THIS ALTERNATE SHIFT RELEASE
3151 00001125 7536 <1> jne short K26 ; INTERRUPT RETURN
3152 <1> ;
3153 <1> ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
3154 00001127 A0[976A0000] <1> mov al, [ALT_INPUT]
3155 0000112C B400 <1> mov ah, 0 ; SCAN CODE OF 0
3156 0000112E 8825[976A0000] <1> mov [ALT_INPUT], ah ; ZERO OUT THE FIELD
3157 00001134 3C00 <1> cmp al, 0 ; WAS THE INPUT = 0?
3158 00001136 7425 <1> je short K26 ; INTERRUPT_RETURN
3159 00001138 E9D0020000 <1> jmp K61 ; IT WASN'T, SO PUT IN BUFFER
3160 <1> ;
3161 <1> K24: ; BREAK-TOGGLE
3162 0000113D 2025[946A0000] <1> and [KB_FLAG_1], ah ; INDICATE NO LONGER DEPRESSED
3163 00001143 EB18 <1> jmp short K26 ; INTERRUPT_RETURN
3164 <1> ;
3165 <1> ;----- TEST FOR HOLD STATE
3166 <1> ; AL, AH = SCAN CODE
3167 <1> K25: ; NO-SHIFT-FOUND
3168 00001145 3C80 <1> cmp al, 80h ; TEST FOR BREAK KEY
3169 00001147 7314 <1> jae short K26 ; NOTHING FOR BREAK CHARS FROM HERE ON
3170 00001149 F605[946A0000]08 <1> test byte [KB_FLAG_1], HOLD_STATE ; ARE WE IN HOLD STATE
3171 00001150 7428 <1> jz short K28 ; BRANCH AROUND TEST IF NOT
3172 00001152 3C45 <1> cmp al, NUM_KEY
3173 00001154 7407 <1> je short K26 ; CAN'T END HOLD ON NUM_LOCK
3174 00001156 8025[946A0000]F7 <1> and byte [KB_FLAG_1], ~HOLD_STATE ; TURN OFF THE HOLD STATE BIT
3175 <1> ;
3176 <1> K26:
3177 0000115D 8025[966A0000]FC <1> and byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
3178 <1> K26A: ; INTERRUPT-RETURN
3179 00001164 FA <1> cli ; TURN OFF INTERRUPTS
3180 00001165 B020 <1> mov al, EOI ; END OF INTERRUPT COMMAND
3181 00001167 E620 <1> out 20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
3182 <1> K27: ; INTERRUPT-RETURN-NO-EOI
3183 00001169 B0AE <1> mov al, ENA_KBD ; INSURE KEYBOARD IS ENABLED
3184 0000116B E8D4020000 <1> call SHIP_IT ; EXECUTE ENABLE
3185 <1> K27A:
3186 00001170 FA <1> cli ; DISABLE INTERRUPTS
3187 00001171 07 <1> pop es ; RESTORE REGISTERS
3188 00001172 1F <1> pop ds
3189 00001173 5F <1> pop edi
3190 00001174 5E <1> pop esi
3191 00001175 5A <1> pop edx
3192 00001176 59 <1> pop ecx
3193 00001177 5B <1> pop ebx
3194 00001178 58 <1> pop eax
3195 <1> ;pop ebp
3196 00001179 CF <1> iret ; RETURN
3197 <1>
3198 <1> ;----- NOT IN HOLD STATE
3199 <1> K28: ; NO-HOLD-STATE
3200 0000117A 3C58 <1> cmp al, 88 ; TEST FOR OUT-OF-RANGE SCAN CODES
3201 0000117C 77DF <1> ja short K26 ; IGNORE IF OUT-OF-RANGE
3202 <1> ;
3203 0000117E F6C308 <1> test bl, ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
3204 <1> ;jz short K28A ; IF NOT ALTERNATE
3205 00001181 0F84F1000000 <1> jz K38
3206 <1> ;
3207 00001187 F6C710 <1> test bh, KBX ; IS THIS THE ENCHANCED KEYBOARD?
3208 0000118A 740D <1> jz short K29 ; NO, ALT STATE IS REAL
3209 <1> ;28/02/2015
3210 0000118C F605[946A0000]04 <1> test byte [KB_FLAG_1], SYS_SHIFT ; YES, IS SYSREQ KEY DOWN?
3211 <1> ;jz short K29 ; NO, ALT STATE IS REAL
3212 00001193 0F85DF000000 <1> jnz K38 ; YES, THIS IS PHONY ALT STATE
3213 <1> ; ; DUE TO PRESSING SYSREQ
3214 <1> ;K28A: jmp short K38

```

3215		<1>	;
3216		<1>	;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
3217		<1>	K29: ; TEST-RESET
3218	00001199 F6C304	<1>	test bl, CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO?
3219	0000119C 740B	<1>	jz short K31 ; NO_RESET
3220	0000119E 3C53	<1>	cmp al, DEL_KEY ; CTL-ALT STATE, TEST FOR DELETE KEY
3221	000011A0 7507	<1>	jne short K31 ; NO_RESET, IGNORE
3222		<1>	;
3223		<1>	;----- CTL-ALT-DEL HAS BEEN FOUND
3224		<1>	; 26/08/2014
3225		<1>	cpu_reset:
3226		<1>	; IBM PC/AT ROM BIOS source code - 10/06/85 (TEST4.ASM - PROC_SHUTDOWN)
3227		<1>	; Send FEh (system reset command) to the keyboard controller.
3228	000011A2 B0FE	<1>	mov al, SHUT_CMD ; SHUTDOWN COMMAND
3229	000011A4 E664	<1>	out STATUS_PORT, al ; SEND TO KEYBOARD CONTROL PORT
3230		<1>	khere:
3231	000011A6 F4	<1>	hlt ; WAIT FOR 80286 RESET
3232	000011A7 EBFD	<1>	jmp short khere ; INSURE HALT
3233		<1>	
3234		<1>	;
3235		<1>	;----- IN ALTERNATE SHIFT, RESET NOT FOUND
3236		<1>	K31: ; NO-RESET
3237	000011A9 3C39	<1>	cmp al, 57 ; TEST FOR SPACE KEY
3238	000011AB 7507	<1>	jne short K311 ; NOT THERE
3239	000011AD B020	<1>	mov al, ' ' ; SET SPACE CHAR
3240	000011AF E948020000	<1>	jmp K57 ; BUFFER_FILL
3241		<1>	K311:
3242	000011B4 3C0F	<1>	cmp al, 15 ; TEST FOR TAB KEY
3243	000011B6 7509	<1>	jne short K312 ; NOT THERE
3244	000011B8 66B800A5	<1>	mov ax, 0A500h ; SET SPECIAL CODE FOR ALT-TAB
3245	000011BC E93B020000	<1>	jmp K57 ; BUFFER_FILL
3246		<1>	K312:
3247	000011C1 3C4A	<1>	cmp al, 74 ; TEST FOR KEY PAD -
3248	000011C3 0F84A2000000	<1>	je K37B ; GO PROCESS
3249	000011C9 3C4E	<1>	cmp al, 78 ; TEST FOR KEY PAD +
3250	000011CB 0F849A000000	<1>	je K37B ; GO PROCESS
3251		<1>	;
3252		<1>	;----- LOOK FOR KEY PAD ENTRY
3253		<1>	K32: ; ALT-KEY-PAD
3254	000011D1 BF[54690000]	<1>	mov edi, K30 ; ALT-INPUT-TABLE offset
3255	000011D6 B90A000000	<1>	mov ecx, 10 ; LOOK FOR ENTRY USING KEYPAD
3256	000011DB F2AE	<1>	repne scasb ; LOOK FOR MATCH
3257	000011DD 7525	<1>	jne short K33 ; NO_ALT_KEYPAD
3258	000011DF F6C702	<1>	test bh, LC_E0 ; IS THIS ONE OF THE NEW KEYS?
3259	000011E2 0F858A000000	<1>	jnz K37C ; YES, JUMP, NOT NUMPAD KEY
3260	000011E8 81EF[55690000]	<1>	sub edi, K30+1 ; DI NOW HAS ENTRY VALUE
3261	000011EE A0[976A0000]	<1>	mov al, [ALT_INPUT] ; GET THE CURRENT BYTE
3262	000011F3 B40A	<1>	mov ah, 10 ; MULTIPLY BY 10
3263	000011F5 F6E4	<1>	mul ah
3264	000011F7 6601F8	<1>	add ax, di ; ADD IN THE LATEST ENTRY
3265	000011FA A2[976A0000]	<1>	mov [ALT_INPUT], al ; STORE IT AWAY
3266		<1>	;K32A:
3267	000011FF E959FFFFFF	<1>	jmp K26 ; THROW AWAY THAT KEYSTROKE
3268		<1>	;
3269		<1>	;----- LOOK FOR SUPERSHIFT ENTRY
3270		<1>	K33: ; NO-ALT-KEYPAD
3271	00001204 C605[976A0000]00	<1>	mov byte [ALT_INPUT], 0 ; ZERO ANY PREVIOUS ENTRY INTO INPUT
3272	0000120B B91A000000	<1>	mov ecx, 26 ; (DI), (ES) ALREADY POINTING
3273	00001210 F2AE	<1>	repne scasb ; LOOK FOR MATCH IN ALPHABET
3274	00001212 7450	<1>	je short K37A ; MATCH FOUND, GO FILL THE BUFFER
3275		<1>	;
3276		<1>	;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
3277		<1>	K34: ; ALT-TOP-ROW
3278	00001214 3C02	<1>	cmp al, 2 ; KEY WITH '1' ON IT
3279	00001216 7253	<1>	jb short K37B ; MUST BE ESCAPE
3280	00001218 3C0D	<1>	cmp al, 13 ; IS IT IN THE REGION
3281	0000121A 7705	<1>	ja short K35 ; NO, ALT SOMETHING ELSE
3282	0000121C 80C476	<1>	add ah, 118 ; CONVERT PSEUDO SCAN CODE TO RANGE
3283	0000121F EB43	<1>	jmp short K37A ; GO FILL THE BUFFER
3284		<1>	;
3285		<1>	;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
3286		<1>	K35: ; ALT-FUNCTION
3287	00001221 3C57	<1>	cmp al, F11_M ; IS IT F11?
3288	00001223 7209	<1>	jb short K35A ; 20/02/2015 ; NO, BRANCH
3289	00001225 3C58	<1>	cmp al, F12_M ; IS IT F12?
3290	00001227 7705	<1>	ja short K35A ; 20/02/2015 ; NO, BRANCH
3291	00001229 80C434	<1>	add ah, 52 ; CONVERT TO PSEUDO SCAN CODE
3292	0000122C EB36	<1>	jmp short K37A ; GO FILL THE BUFFER
3293		<1>	K35A:
3294	0000122E F6C702	<1>	test bh, LC_E0 ; DO WE HAVE ONE OF THE NEW KEYS?
3295	00001231 7422	<1>	jz short K37 ; NO, JUMP
3296	00001233 3C1C	<1>	cmp al, 28 ; TEST FOR KEYPAD ENTER
3297	00001235 7509	<1>	jne short K35B ; NOT THERE
3298	00001237 66B800A6	<1>	mov ax, 0A600h ; SPECIAL CODE
3299	0000123B E9BC010000	<1>	jmp K57 ; BUFFER FILL
3300		<1>	K35B:
3301	00001240 3C53	<1>	cmp al, 83 ; TEST FOR DELETE KEY
3302	00001242 742E	<1>	je short K37C ; HANDLE WITH OTHER EDIT KEYS
3303	00001244 3C35	<1>	cmp al, 53 ; TEST FOR KEYPAD /
3304		<1>	;jne short K32A ; NOT THERE, NO OTHER E0 SPECIALS
3305	00001246 0F8511FFFFFF	<1>	jne K26
3306	0000124C 66B800A4	<1>	mov ax, 0A400h ; SPECIAL CODE

3307	00001250	E9A7010000	<1>	jmp	K57	; BUFFER_FILL
3308			<1>	K37:		
3309	00001255	3C3B	<1>	cmp	al, 59	; TEST FOR FUNCTION KEYS (F1)
3310	00001257	7212	<1>	jb	short K37B	; NO FN, HANDLE W/OTHER EXTENDED
3311	00001259	3C44	<1>	cmp	al, 68	; IN KEYPAD REGION?
3312			<1>	;ja	short K32A	; IF SO, IGNORE
3313	0000125B	0F87FCFEFFFF	<1>	ja	K26	
3314	00001261	80C42D	<1>	add	ah, 45	; CONVERT TO PSEUDO SCAN CODE
3315			<1>	K37A:		
3316	00001264	B000	<1>	mov	al, 0	; ASCII CODE OF ZERO
3317	00001266	E991010000	<1>	jmp	K57	; PUT IT IN THE BUFFER
3318			<1>	K37B:		
3319	0000126B	B0F0	<1>	mov	al, 0F0h	; USE SPECIAL ASCII CODE
3320	0000126D	E98A010000	<1>	jmp	K57	; PUT IT IN THE BUFFER
3321			<1>	K37C:		
3322	00001272	0450	<1>	add	al, 80	; CONVERT SCAN CODE (EDIT KEYS)
3323	00001274	88C4	<1>	mov	ah, al	; (SCAN CODE NOT IN AH FOR INSERT)
3324	00001276	EBEC	<1>	jmp	short K37A	; PUT IT IN THE BUFFER
3325			<1>	;		
3326			<1>	;-----	NOT IN ALTERNATE SHIFT	
3327			<1>	K38:		; NOT-ALT-SHIFT
3328			<1>			; BL STILL HAS SHIFT FLAGS
3329	00001278	F6C304	<1>	test	bl, CTL_SHIFT	; ARE WE IN CONTROL SHIFT?
3330			<1>	;jnz	short K38A	; YES, START PROCESSING
3331	0000127B	0F84B0000000	<1>	jz	K44	; NOT-CTL-SHIFT
3332			<1>	;		
3333			<1>	;-----	CONTROL SHIFT, TEST SPECIAL CHARACTERS	
3334			<1>	;-----	TEST FOR BREAK	
3335			<1>	K38A:		
3336	00001281	3C46	<1>	cmp	al, SCROLL_KEY	; TEST FOR BREAK
3337	00001283	7531	<1>	jne	short K39	; JUMP, NO-BREAK
3338	00001285	F6C710	<1>	test	bh, KBX	; IS THIS THE ENHANCED KEYBOARD?
3339	00001288	7405	<1>	jz	short K38B	; NO, BREAK IS VALID
3340	0000128A	F6C702	<1>	test	bh, LC_E0	; YES, WAS LAST CODE AN E0?
3341	0000128D	7427	<1>	jz	short K39	; NO-BREAK, TEST FOR PAUSE
3342			<1>	K38B:		
3343	0000128F	8B1D[A06A0000]	<1>	mov	ebx, [BUFFER_HEAD]	; RESET BUFFER TO EMPTY
3344	00001295	891D[A46A0000]	<1>	mov	[BUFFER_TAIL], ebx	
3345	0000129B	C605[926A0000]80	<1>	mov	byte [BIOS_BREAK], 80h	; TURN ON BIOS_BREAK BIT
3346			<1>	;		
3347			<1>	;-----	ENABLE KEYBOARD	
3348	000012A2	B0AE	<1>	mov	al, ENA_KBD	; ENABLE KEYBOARD
3349	000012A4	E89B010000	<1>	call	SHIP_IT	; EXECUTE ENABLE
3350			<1>	;		
3351			<1>	;	CTRL+BREAK code here !!!	
3352			<1>	;	INT 1BH	; BREAK INTERRUPT VECTOR
3353			<1>	;	17/10/2015	
3354	000012A9	E89E260000	<1>	call	ctrlbrk	; control+break subroutine
3355			<1>	;		
3356	000012AE	6629C0	<1>	sub	ax, ax	; PUT OUT DUMMY CHARACTER
3357	000012B1	E946010000	<1>	jmp	K57	; BUFFER_FILL
3358			<1>	;		
3359			<1>	;-----	TEST FOR PAUSE	
3360			<1>	K39:		; NO_BREAK
3361	000012B6	F6C710	<1>	test	bh, KBX	; IS THIS THE ENHANCED KEYBOARD?
3362	000012B9	7537	<1>	jnz	short K41	; YES, THEN THIS CAN'T BE PAUSE
3363	000012BB	3C45	<1>	cmp	al, NUM_KEY	; LOOK FOR PAUSE KEY
3364	000012BD	7533	<1>	jne	short K41	; NO-PAUSE
3365			<1>	K39P:		
3366	000012BF	800D[946A0000]08	<1>	or	byte [KB_FLAG_1], HOLD_STATE	; TURN ON THE HOLD FLAG
3367			<1>	;		
3368			<1>	;-----	ENABLE KEYBOARD	
3369	000012C6	B0AE	<1>	mov	al, ENA_KBD	; ENABLE KEYBOARD
3370	000012C8	E877010000	<1>	call	SHIP_IT	; EXECUTE ENABLE
3371			<1>	K39A:		
3372	000012CD	B020	<1>	mov	al, EOI	; END OF INTERRUPT TO CONTROL PORT
3373	000012CF	E620	<1>	out	20h, al ;out INTA00, al	; ALLOW FURTHER KEYSTROKE INTERRUPTS
3374			<1>	;		
3375			<1>	;-----	DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON	
3376	000012D1	803D[906A0000]07	<1>	cmp	byte [CRT_MODE], 7	; IS THIS BLACK AND WHITE CARD
3377	000012D8	740A	<1>	je	short K40	; YES, NOTHING TO DO
3378	000012DA	66BAD803	<1>	mov	dx, 03D8h	; PORT FOR COLOR CARD
3379	000012DE	A0[916A0000]	<1>	mov	al, [CRT_MODE_SET]	; GET THE VALUE OF THE CURRENT MODE
3380	000012E3	EE	<1>	out	dx, al	; SET THE CRT MODE, SO THAT CRT IS ON
3381			<1>	;		
3382			<1>	K40:		; PAUSE-LOOP
3383	000012E4	F605[946A0000]08	<1>	test	byte [KB_FLAG_1], HOLD_STATE	; CHECK HOLD STATE FLAG
3384	000012EB	75F7	<1>	jnz	short K40	; LOOP UNTIL FLAG TURNED OFF
3385			<1>	;		
3386	000012ED	E977FEFFFF	<1>	jmp	K27	; INTERRUPT_RETURN_NO_EOI
3387			<1>	;		
3388			<1>	;-----	TEST SPECIAL CASE KEY 55	
3389			<1>	K41:		; NO-PAUSE
3390	000012F2	3C37	<1>	cmp	al, 55	; TEST FOR */PRTSC KEY
3391	000012F4	7513	<1>	jne	short K42	; NOT-KEY-55
3392	000012F6	F6C710	<1>	test	bh, KBX	; IS THIS THE ENHANCED KEYBOARD?
3393	000012F9	7405	<1>	jz	short K41A	; NO, CTL-PRTSC IS VALID
3394	000012FB	F6C702	<1>	test	bh, LC_E0	; YES, WAS LAST CODE AN E0?
3395	000012FE	7421	<1>	jz	short K42B	; NO, TRANSLATE TO A FUNCTION
3396			<1>	K41A:		
3397	00001300	66B80072	<1>	mov	ax, 114*256	; START/STOP PRINTING SWITCH
3398	00001304	E9F3000000	<1>	jmp	K57	; BUFFER_FILL

```

3399
3400
3401
3402 00001309 3C0F
3403 0000130B 7414
3404 0000130D 3C35
3405 0000130F 750E
3406 00001311 F6C702
3407 00001314 7409
3408 00001316 66B80095
3409 0000131A E9DD000000
3410
3411
3412 0000131F 3C3B
3413
3414
3415
3416
3417 00001321 BB[88690000]
3418 00001326 0F82AE000000
3419 0000132C E9B9000000
3420
3421
3422
3423 00001331 3C37
3424 00001333 7528
3425 00001335 F6C710
3426 00001338 7407
3427 0000133A F6C702
3428 0000133D 7507
3429 0000133F EB41
3430
3431 00001341 F6C303
3432 00001344 743C
3433
3434
3435
3436 00001346 B0AE
3437 00001348 E8F7000000
3438 0000134D B020
3439 0000134F E620
3440
3441
3442
3443
3444 00001351 8025[966A0000]FC
3445 00001358 E90CFEFFFF
3446
3447
3448
3449 0000135D 3C3A
3450 0000135F 7734
3451 00001361 3C35
3452 00001363 7505
3453 00001365 F6C702
3454 00001368 7518
3455
3456 0000136A B91A000000
3457 0000136F BF[5E690000]
3458 00001374 F2AE
3459
3460 00001376 7505
3461
3462 00001378 F6C340
3463 0000137B 750C
3464
3465 0000137D F6C303
3466 00001380 750C
3467
3468
3469 00001382 BB[E0690000]
3470 00001387 EB51
3471
3472 00001389 F6C303
3473 0000138C 75F4
3474
3475 0000138E BB[386A0000]
3476 00001393 EB45
3477
3478
3479
3480 00001395 3C44
3481
3482
3483 00001397 7635
3484
3485
3486
3487 00001399 3C53
3488 0000139B 772D
3489
3490

```

```

<1> ;
<1> ;----- SET UP TO TRANSLATE CONTROL SHIFT
<1> K42: ; NOT-KEY-55
<1> cmp al, 15 ; IS IT THE TAB KEY?
<1> je short K42B ; YES, XLATE TO FUNCTION CODE
<1> cmp al, 53 ; IS IT THE / KEY?
<1> jne short K42A ; NO, NO MORE SPECIAL CASES
<1> test bh, LC_E0 ; YES, IS IT FROM THE KEY PAD?
<1> jz short K42A ; NO, JUST TRANSLATE
<1> mov ax, 9500h ; YES, SPECIAL CODE FOR THIS ONE
<1> jmp K57 ; BUFFER FILL
<1> K42A:
<1> ;mov ebx, _K8 ; SET UP TO TRANSLATE CTL
<1> cmp al, 59 ; IS IT IN CHARACTER TABLE?
<1> ;jb short K45F ; YES, GO TRANSLATE CHAR
<1> ;jb K56 ; 20/02/2015
<1> ;jmp K64 ; 20/02/2015
<1> K42B:
<1> mov ebx, _K8 ; SET UP TO TRANSLATE CTL
<1> jb K56 ; 20/02/2015
<1> jmp K64
<1> ;
<1> ;----- NOT IN CONTROL SHIFT
<1> K44: ; NOT-CTL-SHIFT
<1> cmp al, 55 ; PRINT SCREEN KEY?
<1> jne short K45 ; NOT PRINT SCREEN
<1> test bh, KBX ; IS THIS ENHANCED KEYBOARD?
<1> jz short K44A ; NO, TEST FOR SHIFT STATE
<1> test bh, LC_E0 ; YES, LAST CODE A MARKER?
<1> jnz short K44B ; YES, IS PRINT SCREEN
<1> jmp short K45C ; NO, TRANSLATE TO '*' CHARACTER
<1> K44A:
<1> test bl, LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
<1> jz short K45C ; NO, TRANSLATE TO '*' CHARACTER
<1> ;
<1> ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
<1> K44B:
<1> mov al, ENA_KBD ; INSURE KEYBOARD IS ENABLED
<1> call SHIP_IT ; EXECUTE ENABLE
<1> mov al, EOI ; END OF CURRENT INTERRUPT
<1> out 20h, al ;out INTA00, al ; SO FURTHER THINGS CAN HAPPEN
<1> ; Print Screen !!! ; ISSUE PRINT SCREEN INTERRUPT (INT 05h)
<1> ;PUSH BP ; SAVE POINTER
<1> ;INT 5H ; ISSUE PRINT SCREEN INTERRUPT
<1> ;POP BP ; RESTORE POINTER
<1> and byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; ZERO OUT THESE FLAGS
<1> jmp K27 ; GO BACK WITHOUT EOI OCCURRING
<1> ;
<1> ;----- HANDLE IN-CORE KEYS
<1> K45: ; NOT-PRINT-SCREEN
<1> cmp al, 58 ; TEST FOR IN-CORE AREA
<1> ja short K46 ; JUMP IF NOT
<1> cmp al, 53 ; IS THIS THE '/' KEY?
<1> jne short K45A ; NO, JUMP
<1> test bh, LC_E0 ; WAS THE LAST CODE THE MARKER?
<1> jnz short K45C ; YES, TRANSLATE TO CHARACTER
<1> K45A:
<1> mov ecx, 26 ; LENGHT OF SEARCH
<1> mov edi, K30+10 ; POINT TO TABLE OF A-Z CHARS
<1> repne scasb ; IS THIS A LETTER KEY?
<1> ; 20/02/2015
<1> jne short K45B ; NO, SYMBOL KEY
<1> ;
<1> test bl, CAPS_STATE ; ARE WE IN CAPS_LOCK?
<1> jnz short K45D ; TEST FOR SURE
<1> K45B:
<1> test bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
<1> jnz short K45E ; YES, UPPERCASE
<1> ; NO, LOWERCASE
<1> K45C:
<1> mov ebx, K10 ; TRANSLATE TO LOWERCASE LETTERS
<1> jmp short K56
<1> K45D: ; ALMOST-CAPS-STATE
<1> test bl, LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
<1> jnz short K45C ; SHIFTED TEMP OUT OF CAPS STATE
<1> K45E:
<1> mov ebx, K11 ; TRANSLATE TO UPPER CASE LETTERS
<1> K45F: jmp short K56
<1> ;
<1> ;----- TEST FOR KEYS F1 - F10
<1> K46: ; NOT IN-CORE AREA
<1> cmp al, 68 ; TEST FOR F1 - F10
<1> ;ja short K47 ; JUMP IF NOT
<1> ;jmp short K53 ; YES, GO DO FN KEY PROCESS
<1> jna short K53
<1> ;
<1> ;----- HANDLE THE NUMERIC PAD KEYS
<1> K47: ; NOT F1 - F10
<1> cmp al, 83 ; TEST NUMPAD KEYS
<1> ja short K52 ; JUMP IF NOT
<1> ;
<1> ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION

```

```

3491
3492 0000139D 3C4A
3493 0000139F 74ED
3494 000013A1 3C4E
3495 000013A3 74E9
3496 000013A5 F6C702
3497 000013A8 750A
3498
3499 000013AA F6C320
3500 000013AD 7514
3501 000013AF F6C303
3502
3503 000013B2 75DA
3504
3505
3506
3507 000013B4 3C4C
3508 000013B6 7504
3509 000013B8 B0F0
3510 000013BA EB40
3511
3512 000013BC BB[E0690000]
3513 000013C1 EB27
3514
3515
3516
3517 000013C3 F6C303
3518 000013C6 75EC
3519 000013C8 EBC4
3520
3521
3522
3523 000013CA 3C56
3524
3525
3526 000013CC 74AF
3527
3528
3529
3530 000013CE F6C303
3531 000013D1 74E1
3532
3533 000013D3 BB[386A0000]
3534 000013D8 EB10
3535
3536
3537
3538 000013DA FEC8
3539 000013DC D7
3540 000013DD F605[966A0000]02
3541 000013E4 7416
3542 000013E6 B4E0
3543 000013E8 EB12
3544
3545
3546
3547 000013EA FEC8
3548 000013EC D7
3549 000013ED 88C4
3550 000013EF B000
3551 000013F1 F605[966A0000]02
3552 000013F8 7402
3553 000013FA B0E0
3554
3555
3556
3557 000013FC 3CFF
3558
3559 000013FE 0F8459FDFFFF
3560 00001404 80FCFF
3561
3562 00001407 0F8450FDFFFF
3563
3564
3565
3566 0000140D 8B1D[A46A0000]
3567 00001413 89DE
3568 00001415 E87BFAFFFF
3569 0000141A 3B1D[A06A0000]
3570 00001420 740E
3571 00001422 668906
3572 00001425 891D[A46A0000]
3573 0000142B E92DFDFFFF
3574
3575
3576
3577
3578
3579
3580
3581
3582

<1> K48:
<1> cmp al, 74 ; SPECIAL CASE FOR MINUS
<1> je short K45E ; GO TRANSLATE
<1> cmp al, 78 ; SPECIAL CASE FOR PLUS
<1> je short K45E ; GO TRANSLATE
<1> test bh, LC_E0 ; IS THIS ONE OF THE NEW KEYS?
<1> jnz short K49 ; YES, TRANSLATE TO BASE STATE
<1> ;
<1> test bl, NUM_STATE ; ARE WE IN NUM LOCK
<1> jnz short K50 ; TEST FOR SURE
<1> test bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
<1> ;jnz short K51 ; IF SHIFTED, REALLY NUM STATE
<1> jnz short K45E
<1> ;
<1> ;----- BASE CASE FOR KEYPAD
<1> K49:
<1> cmp al, 76 ; SPECIAL CASE FOR BASE STATE 5
<1> jne short K49A ; CONTINUE IF NOT KEYPAD 5
<1> mov al, 0F0h ; SPECIAL ASCII CODE
<1> jmp short K57 ; BUFFER FILL
<1> K49A:
<1> mov ebx, K10 ; BASE CASE TABLE
<1> jmp short K64 ; CONVERT TO PSEUDO SCAN
<1> ;
<1> ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
<1> K50: ; ALMOST-NUM-STATE
<1> test bl, LEFT_SHIFT+RIGHT_SHIFT
<1> jnz short K49 ; SHIFTED TEMP OUT OF NUM STATE
<1> K51: jmp short K45E ; REALLY NUM STATE
<1> ;
<1> ;----- TEST FOR THE NEW KEYS ON WT KEYBOARDS
<1> K52: ; NOT A NUMPAD KEY
<1> cmp al, 86 ; IS IT THE NEW WT KEY?
<1> ;jne short K53 ; JUMP IF NOT
<1> ;jmp short K45B ; HANDLE WITH REST OF LETTER KEYS
<1> je short K45B
<1> ;
<1> ;----- MUST BE F11 OR F12
<1> K53: ; F1 - F10 COME HERE, TOO
<1> test bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST SHIFT STATE
<1> jz short K49 ; JUMP, LOWER CASE PSEUDO SC'S
<1> ; 20/02/2015
<1> mov ebx, K11 ; UPPER CASE PSEUDO SCAN CODES
<1> jmp short K64 ; TRANSLATE SCAN
<1> ;
<1> ;----- TRANSLATE THE CHARACTER
<1> K56: ; TRANSLATE-CHAR
<1> dec al ; CONVERT ORIGIN
<1> xlat ; CONVERT THE SCAN CODE TO ASCII
<1> test byte [KB_FLAG_3], LC_E0 ; IS THIS A NEW KEY?
<1> jz short K57 ; NO, GO FILL BUFFER
<1> mov ah, MC_E0 ; YES, PUT SPECIAL MARKER IN AH
<1> jmp short K57 ; PUT IT INTO THE BUFFER
<1> ;
<1> ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
<1> K64: ; TRANSLATE-SCAN-ORGD
<1> dec al ; CONVERT ORIGIN
<1> xlat ; CTL TABLE SCAN
<1> mov ah, al ; PUT VALUE INTO AH
<1> mov al, 0 ; ZERO ASCII CODE
<1> test byte [KB_FLAG_3], LC_E0 ; IS THIS A NEW KEY?
<1> jz short K57 ; NO, GO FILL BUFFER
<1> mov al, MC_E0 ; YES, PUT SPECIAL MARKER IN AL
<1> ;
<1> ;----- PUT CHARACTER INTO BUFFER
<1> K57: ; BUFFER_FILL
<1> cmp al, -1 ; IS THIS AN IGNORE CHAR
<1> ;je short K59 ; YES, DO NOTHING WITH IT
<1> je K26 ; YES, DO NOTHING WITH IT
<1> cmp ah, -1 ; LOOK FOR -1 PSEUDO SCAN
<1> ;jne short K61 ; NEAR_INTERRUPT_RETURN
<1> je K26 ; INTERRUPT_RETURN
<1> ;K59: ; NEAR_INTERRUPT_RETURN
<1> ; jmp K26 ; INTERRUPT_RETURN
<1> K61: ; NOT-CAPS-STATE
<1> mov ebx, [BUFFER_TAIL] ; GET THE END POINTER TO THE BUFFER
<1> mov esi, ebx ; SAVE THE VALUE
<1> call _K4 ; ADVANCE THE TAIL
<1> cmp ebx, [BUFFER_HEAD] ; HAS THE BUFFER WRAPPED AROUND
<1> je short K62 ; BUFFER_FULL_BEEP
<1> mov [esi], ax ; STORE THE VALUE
<1> mov [BUFFER_TAIL], ebx ; MOVE THE POINTER UP
<1> jmp K26
<1> ;cli ; TURN OFF INTERRUPTS
<1> ;mov al, EOI ; END OF INTERRUPT COMMAND
<1> ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
<1> ;MOV AL, ENA_KBD ; INSURE KEYBOARD IS ENABLED
<1> ;CALL SHIP_IT ; EXECUTE ENABLE
<1> ;MOV AX, 9102H ; MOVE IN POST CODE & TYPE
<1> ;INT 15H ; PERFORM OTHER FUNCTION
<1> ;and byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
<1> ;JMP K27A ; INTERRUPT_RETURN

```

```

3583      <1> ;;jmp    K27
3584      <1> ;
3585      <1> ;-----    BUFFER IS FULL SOUND THE BEEPER
3586      <1> K62:
3587 00001430 B020      <1> mov    al, EOI                ; ENABLE INTERRUPT CONTROLLER CHIP
3588 00001432 E620      <1> out    INTA00, al
3589 00001434 66B9A602   <1> mov    cx, 678                ; DIVISOR FOR 1760 HZ
3590 00001438 B304      <1> mov    bl, 4                ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
3591 0000143A E8A1010000 <1> call   beep                ; GO TO COMMON BEEP HANDLER
3592 0000143F E925FDFFFF <1> jmp     K27                ; EXIT
3593      <1>
3594      <1> SHIP_IT:
3595      <1> ;-----
--
3596      <1> ; SHIP_IT
3597      <1> ;     THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
3598      <1> ;     TO THE KEYBOARD CONTROLLER.
3599      <1> ;-----
--
3600      <1> ;
3601 00001444 6650      <1> push   ax                ; SAVE DATA TO SEND
3602      <1>
3603      <1> ;-----    WAIT FOR COMMAND TO ACCEPTED
3604 00001446 FA        <1> cli                ; DISABLE INTERRUPTS TILL DATA SENT
3605      <1> ; xor    ecx, ecx                ; CLEAR TIMEOUT COUNTER
3606 00001447 B900000100 <1> mov    ecx, 10000h
3607      <1> S10:
3608 0000144C E464      <1> in     al, STATUS_PORT      ; READ KEYBOARD CONTROLLER STATUS
3609 0000144E A802      <1> test   al, INPT_BUF_FULL ; CHECK FOR ITS INPUT BUFFER BUSY
3610 00001450 E0FA      <1> loopnz  S10                ; WAIT FOR COMMAND TO BE ACCEPTED
3611      <1>
3612 00001452 6658      <1> pop    ax                ; GET DATA TO SEND
3613 00001454 E664      <1> out    STATUS_PORT, al      ; SEND TO KEYBOARD CONTROLLER
3614 00001456 FB        <1> sti                ; ENABLE INTERRUPTS AGAIN
3615 00001457 C3        <1> retn                ; RETURN TO CALLER
3616      <1>
3617      <1> SND_DATA:
3618      <1> ;-----
--
3619      <1> ; SND_DATA
3620      <1> ;     THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
3621      <1> ;     TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO
3622      <1> ;     HANDLES ANY RETRIES IF REQUIRED
3623      <1> ;-----
--
3624      <1> ;
3625 00001458 6650      <1> push   ax                ; SAVE REGISTERS
3626 0000145A 6653      <1> push   bx
3627 0000145C 51        <1> push   ecx
3628 0000145D 88C7      <1> mov    bh, al                ; SAVE TRANSMITTED BYTE FOR RETRIES
3629 0000145F B303      <1> mov    bl, 3                ; LOAD RETRY COUNT
3630      <1> SD0:
3631 00001461 FA        <1> cli                ; DISABLE INTERRUPTS
3632 00001462 8025[956A0000]CF <1> and    byte [KB_FLAG_2], ~(KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
3633      <1> ;
3634      <1> ;-----    WAIT FOR COMMAND TO BE ACCEPTED
3635 00001469 B900000100 <1> mov    ecx, 10000h          ; MAXIMUM WAIT COUNT
3636      <1> SD5:
3637 0000146E E464      <1> in     al, STATUS_PORT      ; READ KEYBOARD PROCESSOR STATUS PORT
3638 00001470 A802      <1> test   al, INPT_BUF_FULL ; CHECK FOR ANY PENDING COMMAND
3639 00001472 E0FA      <1> loopnz  SD5                ; WAIT FOR COMMAND TO BE ACCEPTED
3640      <1> ;
3641 00001474 88F8      <1> mov    al, bh                ; REESTABLISH BYTE TO TRANSMIT
3642 00001476 E660      <1> out    PORT_A, al          ; SEND BYTE
3643 00001478 FB        <1> sti                ; ENABLE INTERRUPTS
3644      <1> ;mov    cx, 01A00h          ; LOAD COUNT FOR 10 ms+
3645 00001479 B9FFFF0000 <1> mov    ecx, 0FFFFh
3646      <1> SD1:
3647 0000147E F605[956A0000]30 <1> test   byte [KB_FLAG_2], KB_FE+KB_FA ; SEE IF EITHER BIT SET
3648 00001485 750F      <1> jnz     short SD3          ; IF SET, SOMETHING RECEIVED GO PROCESS
3649 00001487 E2F5      <1> loop    SD1                ; OTHERWISE WAIT
3650      <1> SD2:
3651 00001489 FECB      <1> dec    bl                ; DECREMENT RETRY COUNT
3652 0000148B 75D4      <1> jnz     short SD0          ; RETRY TRANSMISSION
3653 0000148D 800D[956A0000]80 <1> or     byte [KB_FLAG_2], KB_ERR ; TURN ON TRANSMIT ERROR FLAG
3654 00001494 EB09      <1> jmp     short SD4          ; RETRIES EXHAUSTED FORGET TRANSMISSION
3655      <1> SD3:
3656 00001496 F605[956A0000]10 <1> test   byte [KB_FLAG_2], KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
3657 0000149D 74EA      <1> jz      short SD2          ; IF NOT, GO RESEND
3658      <1> SD4:
3659 0000149F 59        <1> pop    ecx                ; RESTORE REGISTERS
3660 000014A0 665B      <1> pop    bx
3661 000014A2 6658      <1> pop    ax
3662 000014A4 C3        <1> retn                ; RETURN, GOOD TRANSMISSION
3663      <1>
3664      <1> SND_LED:
3665      <1> ;-----
--
3666      <1> ; SND_LED
3667      <1> ;     THIS ROUTINES TURNS ON THE MODE INDICATORS.
3668      <1> ;

```



```

3669      <1> ;-----
--
3670      <1> ;
3671 000014A5 FA      <1> cli ; TURN OFF INTERRUPTS
3672 000014A6 F605[956A0000]40 <1> test byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
3673 000014AD 755F      <1> jnz short SL1 ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
3674      <1> ;
3675 000014AF 800D[956A0000]40 <1> or byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
3676 000014B6 B020      <1> mov al, EOI ; END OF INTERRUPT COMMAND
3677 000014B8 E620      <1> out 20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
3678 000014BA EB11      <1> jmp short SL0 ; GO SEND MODE INDICATOR COMMAND
3679      <1> SND_LED1:
3680 000014BC FA      <1> cli ; TURN OFF INTERRUPTS
3681 000014BD F605[956A0000]40 <1> test byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
3682 000014C4 7548      <1> jnz short SL1 ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
3683      <1> ;
3684 000014C6 800D[956A0000]40 <1> or byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
3685      <1> SL0:
3686 000014CD B0ED      <1> mov al, LED_CMD ; LED CMD BYTE
3687 000014CF E884FFFFFF      <1> call SND_DATA ; SEND DATA TO KEYBOARD
3688 000014D4 FA      <1> cli
3689 000014D5 E836000000      <1> call MAKE_LED ; GO FORM INDICATOR DATA BYTE
3690 000014DA 8025[956A0000]F8 <1> and byte [KB_FLAG_2], 0F8h ; ~KB_LEDS ; CLEAR MODE INDICATOR BITS
3691 000014E1 0805[956A0000]      <1> or [KB_FLAG_2], al ; SAVE PRESENT INDICATORS FOR NEXT TIME
3692 000014E7 F605[956A0000]80 <1> test byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
3693 000014EE 750F      <1> jnz short SL2 ; IF SO, BYPASS SECOND BYTE TRANSMISSION
3694      <1> ;
3695 000014F0 E863FFFFFF      <1> call SND_DATA ; SEND DATA TO KEYBOARD
3696 000014F5 FA      <1> cli ; TURN OFF INTERRUPTS
3697 000014F6 F605[956A0000]80 <1> test byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
3698 000014FD 7408      <1> jz short SL3 ; IF NOT, DON'T SEND AN ENABLE COMMAND
3699      <1> SL2:
3700 000014FF B0F4      <1> mov al, KB_ENABLE ; GET KEYBOARD CSA ENABLE COMMAND
3701 00001501 E852FFFFFF      <1> call SND_DATA ; SEND DATA TO KEYBOARD
3702 00001506 FA      <1> cli ; TURN OFF INTERRUPTS
3703      <1> SL3:
3704 00001507 8025[956A0000]3F <1> and byte [KB_FLAG_2], ~(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
3705      <1> SL1: ; UPDATE AND TRANSMIT ERROR FLAG
3706 0000150E FB      <1> sti ; ENABLE INTERRUPTS
3707 0000150F C3      <1> retn ; RETURN TO CALLER
3708      <1>
3709      <1> MAKE_LED:
3710      <1> ;-----
-
3711      <1> ; MAKE_LED
3712      <1> ; THIS ROUTINES FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF
3713      <1> ; THE MODE INDICATORS.
3714      <1> ;-----
-
3715      <1> ;
3716      <1> ;push cx ; SAVE CX
3717 00001510 A0[936A0000] <1> mov al, [KB_FLAG] ; GET CAPS & NUM LOCK INDICATORS
3718 00001515 2470      <1> and al, CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
3719      <1> ;mov cl, 4 ; SHIFT COUNT
3720      <1> ;rol al, cl ; SHIFT BITS OVER TO TURN ON INDICATORS
3721 00001517 C0C004      <1> rol al, 4 ; 20/02/2015
3722 0000151A 2407      <1> and al, 07h ; MAKE SURE ONLY MODE BITS ON
3723      <1> ;pop cx
3724 0000151C C3      <1> retn ; RETURN TO CALLER
3725      <1>
3726      <1> ; % include 'kybdata.inc' ; KEYBOARD DATA ; 11/03/2015
3727      <1>
3728      <1>
3729      <1> ; /// End Of KEYBOARD FUNCTIONS ///
3730
3731      %include 'video.inc' ; 07/03/2015
3732      <1> ; Retro UNIX 386 v1 Kernel - VIDEO.INC
3733      <1> ; Last Modification: 13/08/2015
3734      <1> ; (Video Data is in 'VIDATA.INC')
3735      <1> ;
3736      <1> ; ////////// VIDEO (CGA) FUNCTIONS //////////
3737      <1>
3738      <1> ; 30/06/2015
3739      <1> ; 27/06/2015
3740      <1> ; 11/03/2015
3741      <1> ; 02/09/2014
3742      <1> ; 30/08/2014
3743      <1> ; VIDEO FUNCTIONS
3744      <1> ; (write_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)
3745      <1>
3746      <1> write_tty:
3747      <1> ; 13/08/2015
3748      <1> ; 02/09/2014
3749      <1> ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
3750      <1> ; 01/02/2014 (Retro UNIX 8086 v1 - last update)
3751      <1> ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)
3752      <1> ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
3753      <1> ;
3754      <1> ; INPUT -> AH = Color (Forecolor, Backcolor)
3755      <1> ; AL = Character to be written
3756      <1> ; EBX = Video Page (0 to 7)
3757      <1> ; (BH = 0 --> Video Mode 3)

```

```

3758 <1>
3759 <1> RVRT equ 00001000b ; VIDEO VERTICAL RETRACE BIT
3760 <1> RHRZ equ 00000001b ; VIDEO HORIZONTAL RETRACE BIT
3761 <1>
3762 <1> ; Derived from "WRITE_TTY" procedure of IBM "pc-at" rombios source code
3763 <1> ; (06/10/1985), 'video.asm', INT 10H, VIDEO_IO
3764 <1> ;
3765 <1> ; 06/10/85 VIDEO DISPLAY BIOS
3766 <1> ;
3767 <1> ;--- WRITE_TTY -----
3768 <1> ;
3769 <1> ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE :
3770 <1> ; VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT :
3771 <1> ; CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION. :
3772 <1> ; IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN :
3773 <1> ; IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW :
3774 <1> ; ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW, :
3775 <1> ; FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE. :
3776 <1> ; WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE :
3777 <1> ; NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS :
3778 <1> ; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE, :
3779 <1> ; THE 0 COLOR IS USED. :
3780 <1> ; ENTRY -- :
3781 <1> ; (AH) = CURRENT CRT MODE :
3782 <1> ; (AL) = CHARACTER TO BE WRITTEN :
3783 <1> ; NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE :
3784 <1> ; HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS :
3785 <1> ; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE :
3786 <1> ; EXIT -- :
3787 <1> ; ALL REGISTERS SAVED :
3788 <1> ;-----
3789 <1>
3790 0000151D FA <1> cli
3791 <1> ;
3792 <1> ; READ CURSOR (04/12/2013)
3793 <1> ; Retro UNIX 386 v1 Modifications: 30/08/2014
3794 0000151E 08FF <1> or bh, bh
3795 00001520 0F85AB000000 <1> jnz beeper
3796 <1> ; 01/09/2014
3797 00001526 803D[906A0000]03 <1> cmp byte [CRT_MODE], 3
3798 0000152D 7405 <1> je short m3
3799 <1> ;
3800 0000152F E889020000 <1> call set_mode
3801 <1> m3:
3802 00001534 89DE <1> mov esi, ebx ; 13/08/2015 (0 to 7)
3803 00001536 66D1E6 <1> shl si, 1
3804 00001539 81C6[C6700000] <1> add esi, cursor_posn
3805 0000153F 668B16 <1> mov dx, [esi]
3806 <1> ;
3807 <1> ; dx now has the current cursor position
3808 <1> ;
3809 00001542 3C0D <1> cmp al, 0Dh ; is it carriage return or control character
3810 00001544 764D <1> jbe short u8
3811 <1> ;
3812 <1> ; write the char to the screen
3813 <1> u0:
3814 <1> ; ah = attribute/color
3815 <1> ; al = character
3816 <1> ; bh = video page number (0 to 7)
3817 <1> ; bh = 0
3818 <1> ;
3819 00001546 E841020000 <1> call write_c_current
3820 <1> ;
3821 <1> ; position the cursor for next char
3822 0000154B FEC2 <1> inc dl ; next column
3823 <1> ;cmp dl, [CRT_COLS]
3824 0000154D 80FA50 <1> cmp dl, 80 ; test for column overflow
3825 00001550 0F85EB000000 <1> jne set_cpos
3826 00001556 B200 <1> mov dl, 0 ; column = 0
3827 <1> u10: ; (line feed found)
3828 00001558 80FE18 <1> cmp dh, 25-1 ; check for last row
3829 0000155B 722F <1> jb short u6
3830 <1> ;
3831 <1> ; scroll required
3832 <1> u1:
3833 <1> ; SET CURSOR POSITION (04/12/2013)
3834 0000155D E8DF000000 <1> call set_cpos
3835 <1> ;
3836 <1> ; determine value to fill with during scroll
3837 <1> u2:
3838 <1> ; READ_AC_CURRENT :
3839 <1> ; THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER
3840 <1> ; AT THE CURRENT CURSOR POSITION
3841 <1> ;
3842 <1> ; INPUT
3843 <1> ; (AH) = CURRENT CRT MODE
3844 <1> ; (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
3845 <1> ; (DS) = DATA SEGMENT
3846 <1> ; (ES) = REGEN SEGMENT
3847 <1> ; OUTPUT
3848 <1> ; (AL) = CHARACTER READ
3849 <1> ; (AH) = ATTRIBUTE READ

```

3850		<1>	;
3851		<1>	; mov ah, [CRT_MODE] ; move current mode into ah
3852		<1>	;
3853		<1>	; bl = video page number
3854		<1>	;
3855	00001562 E837010000	<1>	call find_position ; get regen location and port address
3856		<1>	; dx = status port
3857		<1>	; esi = cursor location/address
3858		<1>	p11:
3859	00001567 FB	<1>	sti ; enable interrupts
3860	00001568 90	<1>	nop ; allow for small interrupts window
3861	00001569 FA	<1>	cli ; blocks interrupts for single loop
3862	0000156A EC	<1>	in al, dx ; get status from adapter
3863	0000156B A801	<1>	test al, RHRZ ; is horizontal retrace low
3864	0000156D 75F8	<1>	jnz short p11 ; wait until it is
3865		<1>	p12: ; now wait for either retrace high
3866	0000156F EC	<1>	in al, dx ; get status
3867	00001570 A809	<1>	test al, RVRT+RHRZ ; is horizontal or vertical retrace high
3868	00001572 74FB	<1>	jz short p12 ; wait until either is active
3869		<1>	p13:
3870	00001574 81C600800B00	<1>	add esi, 0B8000h ; 30/08/2014 (Retro UNIX 386 v1)
3871	0000157A 668B06	<1>	mov ax, [esi] ; get the character and attribute
3872		<1>	;
3873		<1>	; al = character, ah = attribute
3874		<1>	;
3875	0000157D FB	<1>	sti
3876		<1>	; bl = video page number
3877		<1>	u3:
3878		<1>	;;mov ax, 0601h ; scroll one line
3879		<1>	;;sub cx, cx ; upper left corner
3880		<1>	;;mov dh, 25-1 ; lower right row
3881		<1>	;;;mov dl, [CRT_COLS]
3882		<1>	;mov dl, 80 ; lower right column
3883		<1>	;;dec dl
3884		<1>	;;mov dl, 79
3885		<1>	
3886		<1>	;;call scroll_up ; 04/12/2013
3887		<1>	;;; 11/03/2015
3888		<1>	; 02/09/2014
3889		<1>	;;;mov cx, [crt_ulc] ; Upper left corner (0000h)
3890		<1>	;;;mov dx, [crt_lrc] ; Lower right corner (184Fh)
3891		<1>	; 11/03/2015
3892	0000157E 6629C9	<1>	sub cx, cx
3893	00001581 66BA4F18	<1>	mov dx, 184Fh ; dl= 79 (column), dh = 24 (row)
3894		<1>	;
3895	00001585 B001	<1>	mov al, 1 ; scroll 1 line up
3896		<1>	; ah = attribute
3897	00001587 E93E010000	<1>	jmp scroll_up
3898		<1>	;u4:
3899		<1>	;;int 10h ; video-call return
3900		<1>	; scroll up the screen
3901		<1>	; tty return
3902		<1>	;u5:
3903		<1>	;retn ; return to the caller
3904		<1>	
3905		<1>	u6: ; set-cursor-inc
3906	0000158C FEC6	<1>	inc dh ; next row
3907		<1>	; set cursor
3908		<1>	;u7:
3909		<1>	;;mov ah, 02h
3910		<1>	;;jmp short u4 ; establish the new cursor
3911		<1>	;call set_cpos
3912		<1>	;jmp short u5
3913	0000158E E9AE000000	<1>	jmp set_cpos
3914		<1>	
3915		<1>	; check for control characters
3916		<1>	u8:
3917	00001593 7438	<1>	je short u9
3918	00001595 3C0A	<1>	cmp al, 0Ah ; is it a line feed (0Ah)
3919	00001597 74BF	<1>	je short u10
3920	00001599 3C07	<1>	cmp al, 07h ; is it a bell
3921	0000159B 7434	<1>	je short u11
3922	0000159D 3C08	<1>	cmp al, 08h ; is it a backspace
3923		<1>	;jne short u0
3924	0000159F 7424	<1>	je short bs ; 12/12/2013
3925		<1>	; 12/12/2013 (tab stop)
3926	000015A1 3C09	<1>	cmp al, 09h ; is it a tab stop
3927	000015A3 75A1	<1>	jne short u0
3928	000015A5 88D0	<1>	mov al, dl
3929	000015A7 6698	<1>	cbw
3930	000015A9 B108	<1>	mov cl, 8
3931	000015AB F6F1	<1>	div cl
3932	000015AD 28E1	<1>	sub cl, ah
3933		<1>	ts:
3934		<1>	; 02/09/2014
3935		<1>	; 01/09/2014
3936	000015AF B020	<1>	mov al, 20h
3937		<1>	tsloop:
3938	000015B1 6651	<1>	push cx
3939	000015B3 6650	<1>	push ax
3940	000015B5 30FF	<1>	xor bh, bh
3941		<1>	;mov bl, [active_page]

3942	000015B7	E878FFFFFF	<1>	call	m3	
3943	000015BC	6658	<1>	pop	ax	; ah = attribute/color
3944	000015BE	6659	<1>	pop	cx	
3945	000015C0	FEC9	<1>	dec	cl	
3946	000015C2	75ED	<1>	jnz	short	tsloop
3947	000015C4	C3	<1>	retn		
3948			<1>	bs:		
3949			<1>		; back space found	
3950			<1>			
3951	000015C5	08D2	<1>	or	dl, dl	; is it already at start of line
3952			<1>	;je	short u7	; set_cursor
3953	000015C7	7478	<1>	jz	short	set_cpos
3954	000015C9	664A	<1>	dec	dx	; no -- just move it back
3955			<1>	;jmp	short u7	
3956	000015CB	EB74	<1>	jmp	short	set_cpos
3957			<1>			
3958			<1>		; carriage return found	
3959			<1>	u9:		
3960	000015CD	B200	<1>	mov	dl, 0	; move to first column
3961			<1>	;jmp	short u7	
3962	000015CF	EB70	<1>	jmp	short	set_cpos
3963			<1>			
3964			<1>		; line feed found	
3965			<1>	u10:		
3966			<1>	;cmp	dh, 25-1	; bottom of screen
3967			<1>	;jne	short u6	; no, just set the cursor
3968			<1>	; jmp	u1	; yes, scroll the screen
3969			<1>			
3970			<1>	beeper:		
3971			<1>		; 30/08/2014 (Retro UNIX 386 v1)	
3972			<1>		; 18/01/2014	
3973			<1>		; 03/12/2013	
3974			<1>		; bell found	
3975			<1>	u11:		
3976	000015D1	FB	<1>	sti		
3977	000015D2	3A1D[D6700000]	<1>	cmp	bl, [active_page]	
3978	000015D8	7551	<1>	jne	short u12	; Do not sound the beep
3979			<1>		; if it is not written on the active page	
3980	000015DA	66B93305	<1>	mov	cx, 1331	; divisor for 896 hz tone
3981	000015DE	B31F	<1>	mov	bl, 31	; set count for 31/64 second for beep
3982			<1>	;call	beep	; sound the pod bell
3983			<1>	;jmp	short u5	; tty_return
3984			<1>	;retn		
3985			<1>			
3986			<1>	TIMER	equ 040h	; 8254 TIMER - BASE ADDRESS
3987			<1>	PORT_B	equ 061h	; PORT B READ/WRITE DIAGNOSTIC REGISTER
3988			<1>	GATE2	equ 00000001b	; TIMER 2 INPUT CATE CLOCK BIT
3989			<1>	SPK2	equ 00000010b	; SPEAKER OUTPUT DATA ENABLE BIT
3990			<1>			
3991			<1>	beep:		
3992			<1>		; 07/02/2015	
3993			<1>		; 30/08/2014 (Retro UNIX 386 v1)	
3994			<1>		; 18/01/2014	
3995			<1>		; 03/12/2013	
3996			<1>			
3997			<1>		; TEST4.ASM - 06/10/85 POST AND BIOS UTILITY ROUTINES	
3998			<1>			
3999			<1>		; ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE	
4000			<1>			
4001			<1>		; ENTRY:	
4002			<1>		; (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )	
4003			<1>		; (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)	
4004			<1>		; EXIT:	
4005			<1>		; (AX), (BL), (CX) MODIFIED.	
4006			<1>			
4007	000015E0	9C	<1>	pushf	; 18/01/2014	; save interrupt status
4008	000015E1	FA	<1>	cli		; block interrupts during update
4009	000015E2	B0B6	<1>	mov	al, 10110110b	; select timer 2, lsb, msb binary
4010	000015E4	E643	<1>	out	TIMER+3, al	; write timer mode register
4011	000015E6	EB00	<1>	jmp	\$+2	; I/O delay
4012	000015E8	88C8	<1>	mov	al, cl	; divisor for hz (low)
4013	000015EA	E642	<1>	out	TIMER+2, AL	; write timer 2 count - lsb
4014	000015EC	EB00	<1>	jmp	\$+2	; I/O delay
4015	000015EE	88E8	<1>	mov	al, ch	; divisor for hz (high)
4016	000015F0	E642	<1>	out	TIMER+2, al	; write timer 2 count - msb
4017	000015F2	E461	<1>	in	al, PORT_B	; get current setting of port
4018	000015F4	88C4	<1>	mov	ah, al	; save that setting
4019	000015F6	0C03	<1>	or	al, GATE2+SPK2	; gate timer 2 and turn speaker on
4020	000015F8	E661	<1>	out	PORT_B, al	; and restore interrupt status
4021			<1>	;popf	; 18/01/2014	
4022	000015FA	FB	<1>	sti		
4023			<1>	g7:		; 1/64 second per count (bl)
4024	000015FB	B90B040000	<1>	mov	ecx, 1035	; delay count for 1/64 of a second
4025	00001600	E827000000	<1>	call	waitf	; go to beep delay 1/64 count
4026	00001605	FECB	<1>	dec	bl	; (bl) length count expired?
4027	00001607	75F2	<1>	jnz	short g7	; no - continue beeping speaker
4028			<1>			
4029			<1>	;pushf		; save interrupt status
4030	00001609	FA	<1>	cli	; 18/01/2014	; block interrupts during update
4031	0000160A	E461	<1>	in	al, PORT_B	; get current port value
4032			<1>	;or	al, not (GATE2+SPK2)	; isolate current speaker bits in case
4033	0000160C	0CFC	<1>	or	al, ~(GATE2+SPK2)	

```

4034 0000160E 20C4      <1>      and ah, al          ; someone turned them off during beep
4035 00001610 88E0      <1>      mov al, ah          ; recover value of port
4036                      <1>      ;or      al, not (GATE2+SPK2) ; force speaker data off
4037 00001612 0CFC      <1>      or al, ~(GATE2+SPK2) ; isolate current speaker bits in case
4038 00001614 E661      <1>      out PORT_B, al ; and stop speaker timer
4039                      <1>      ;popf          ; restore interrupt flag state
4040 00001616 FB         <1>      sti
4041 00001617 B90B040000 <1>      mov ecx, 1035      ; force 1/64 second delay (short)
4042 0000161C E80B000000 <1>      call waitf        ; minimum delay between all beeps
4043                      <1>      ;pushf          ; save interrupt status
4044 00001621 FA         <1>      cli          ; block interrupts during update
4045 00001622 E461      <1>      in al, PORT_B ; get current port value in case
4046 00001624 2403      <1>      and al, GATE2+SPK2 ; someone turned them on
4047 00001626 08E0      <1>      or al, ah          ; recover value of port_b
4048 00001628 E661      <1>      out PORT_B, al ; restore speaker status
4049 0000162A 9D         <1>      popf          ; restore interrupt flag state
4050                      <1>      ul2:
4051 0000162B C3         <1>      retn
4052                      <1>
4053                      <1>      REFRESH_BIT equ 00010000b ; REFRESH TEST BIT
4054                      <1>
4055                      <1>      WAITF:
4056                      <1>      waitf:
4057                      <1>      ; 30/08/2014 (Retro UNIX 386 v1)
4058                      <1>      ; 03/12/2013
4059                      <1>      ;
4060                      <1>      ;push ax          ; save work register (ah)
4061                      <1>      ;waitf1:
4062                      <1>      ; use timer 1 output bits
4063                      <1>      ;in al, PORT_B ; read current counter output status
4064                      <1>      ;and al, REFRESH_BIT ; mask for refresh determine bit
4065                      <1>      ;cmp al, ah          ; did it just change
4066                      <1>      ;je short waitf1 ; wait for a change in output line
4067                      <1>      ;;
4068                      <1>      ;mov ah, al          ; save new lflag state
4069                      <1>      ;loop waitf1 ; decrement half cycles till count end
4070                      <1>      ;;
4071                      <1>      ;pop ax          ; restore (ah)
4072                      <1>      ;retn          ; return (cx)=0
4073                      <1>
4074                      <1>      ; 06/02/2015 (unix386.s <-- dsectrm2.s)
4075                      <1>      ; 17/12/2014 (dsectrm2.s)
4076                      <1>      ; WAITF
4077                      <1>      ; /// IBM PC-XT Model 286 System BIOS Source Code - Test 4 - 06/10/85 ///
4078                      <1>      ;
4079                      <1>      ;---WAITF-----
4080                      <1>      ; FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)
4081                      <1>      ; ENTRY:
4082                      <1>      ; (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
4083                      <1>      ; MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE
4084                      <1>      ; EXIT:
4085                      <1>      ; AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)
4086                      <1>      ; (CX) = 0
4087                      <1>      ;-----
4088                      <1>
4089                      <1>      ; Refresh period: 30 micro seconds (15-80 us)
4090                      <1>      ; (16/12/2014 - AWARDBIOS 1999 - ATORGS.ASM, WAIT_REFRESH)
4091                      <1>
4092                      <1>      ;WAITF: ; DELAY FOR (CX)*15.085737 US
4093 0000162C 6650      <1>      PUSH AX ; SAVE WORK REGISTER (AH)
4094                      <1>      ; 16/12/2014
4095                      <1>      ;shr cx, 1 ; convert to count of 30 micro seconds
4096 0000162E D1E9      <1>      shr ecx, 1 ; 21/02/2015
4097                      <1>      ;17/12/2014
4098                      <1>      ;WAITF1:
4099                      <1>      ;IN AL, PORT_B ;061h ; READ CURRENT COUNTER OUTPUT STATUS
4100                      <1>      ;AND AL, REFRESH_BIT ;00010000b ; MASK FOR REFRESH DETERMINE BIT
4101                      <1>      ;CMP AL, AH ; DID IT JUST CHANGE
4102                      <1>      ;JE short WAITF1 ; WAIT FOR A CHANGE IN OUTPUT LINE
4103                      <1>      ;MOV AH, AL ; SAVE NEW FLAG STATE
4104                      <1>      ;LOOP WAITF1 ; DECREMENT HALF CYCLES TILL COUNT END
4105                      <1>      ;
4106                      <1>      ; 17/12/2014
4107                      <1>      ;
4108                      <1>      ; Modification from 'WAIT_REFRESH' procedure of AWARD BIOS - 1999
4109                      <1>      ;
4110                      <1>      ;WAIT_REFRESH: Uses port 61, bit 4 to have CPU speed independent waiting.
4111                      <1>      ; INPUT: CX = number of refresh periods to wait
4112                      <1>      ; (refresh periods = 1 per 30 microseconds on most machines)
4113                      <1>      WR_STATE_0:
4114 00001630 E461      <1>      IN AL,PORT_B ; IN AL,SYS1
4115 00001632 A810      <1>      TEST AL,010H
4116 00001634 74FA      <1>      JZ SHORT WR_STATE_0
4117                      <1>      WR_STATE_1:
4118 00001636 E461      <1>      IN AL,PORT_B ; IN AL,SYS1
4119 00001638 A810      <1>      TEST AL,010H
4120 0000163A 75FA      <1>      JNZ SHORT WR_STATE_1
4121 0000163C E2F2      <1>      LOOP WR_STATE_0
4122                      <1>      ;
4123 0000163E 6658      <1>      POP AX ; RESTORE (AH)
4124 00001640 C3         <1>      RETN ; (CX) = 0
4125                      <1>

```

```

4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145 00001641 0FB6C3
4146 00001644 D0E0
4147 00001646 BE[C6700000]
4148 0000164B 01C6
4149 0000164D 668916
4150 00001650 381D[D6700000]
4151 00001656 7532
4152
4153
4154
4155
4156
4157
4158 00001658 E832000000
4159 0000165D 668B0D[C4700000]
4160 00001664 6601C1
4161
4162 00001667 66D1E9
4163 0000166A B40E
4164
4165
4166
4167
4168
4169
4170 0000166C FA
4171
4172 0000166D 66BAD403
4173 00001671 88E0
4174 00001673 EE
4175 00001674 6642
4176 00001676 EB00
4177 00001678 88E8
4178 0000167A EE
4179 0000167B 664A
4180 0000167D 88E0
4181 0000167F FEC0
4182 00001681 EE
4183 00001682 6642
4184 00001684 EB00
4185 00001686 88C8
4186 00001688 EE
4187 00001689 FB
4188
4189 0000168A C3
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212 0000168B B40A
4213
4214
4215
4216 0000168D EBDD
4217

<1> set_cpos:
<1> ; 27/06/2015
<1> ; 01/09/2014
<1> ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
<1> ;
<1> ; 12/12/2013 (Retro UNIX 8086 v1 - last update)
<1> ; 04/12/2013 (Retro UNIX 8086 v1 - beginning)
<1> ;
<1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
<1> ;
<1> ; SET_CPOS
<1> ; THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
<1> ; NEW X-Y VALUES PASSED
<1> ; INPUT
<1> ; DX - ROW,COLUMN OF NEW CURSOR
<1> ; BH - DISPLAY PAGE OF CURSOR
<1> ; OUTPUT
<1> ; CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
<1> ;
<1> movzx eax, bl ; BL = video page number ; 27/06/2015 (movzx)
<1> shl al, 1 ; word offset
<1> mov esi, cursor_posn
<1> add esi, eax
<1> mov [esi], dx ; save the pointer
<1> cmp [active_page], bl
<1> jne short m17
<1> ;call m18 ; CURSOR SET
<1> ;m17: ; SET_CPOS_RETURN
<1> ; 01/09/2014
<1> ;retn
<1> ; DX = row/column
<1> m18:
<1> call position ; determine location in regen buffer
<1> mov cx, [CRT_START]
<1> add cx, ax ; add char position in regen buffer
<1> ; to the start address (offset) for this page
<1> shr cx, 1 ; divide by 2 for charonly count
<1> mov ah, 14 ; register number for cursor
<1> ;call m16 ; output value to the 6845
<1> ;retn
<1>
<1> ;----- THIS ROUTINE OUTPUTS THE CX REGISTER
<1> ; TO THE 6845 REGISTERS NAMED IN (AH)
<1> m16:
<1> cli
<1> ;mov dx, [addr_6845] ; address register
<1> mov dx, 03D4h ; I/O address of color card
<1> mov al, ah ; get value
<1> out dx, al ; register set
<1> inc dx ; data register
<1> jmp $+2 ; i/o delay
<1> mov al, ch ; data
<1> out dx, al
<1> dec dx
<1> mov al, ah
<1> inc al ; point to other data register
<1> out dx, al ; set for second register
<1> inc dx
<1> jmp $+2 ; i/o delay
<1> mov al, cl ; second data value
<1> out dx, al
<1> sti
<1> m17:
<1> retn
<1>
<1>
<1> set_ctype:
<1> ; 02/09/2014 (Retro UNIX 386 v1)
<1> ;
<1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
<1> ;
<1> ; CH) = BITS 4-0 = START LINE FOR CURSOR
<1> ; ** HARDWARE WILL ALWAYS CAUSE BLINK
<1> ; ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
<1> ; OR NO CURSOR AT ALL
<1> ; (CL) = BITS 4-0 = END LINE FOR CURSOR
<1> ;
<1> ;-----
<1> ; SET_CTYPE
<1> ; THIS ROUTINE SETS THE CURSOR VALUE
<1> ; INPUT
<1> ; (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
<1> ; OUTPUT
<1> ; NONE
<1> ;-----
<1>
<1> mov ah, 10 ; 6845 register for cursor set
<1> ;mov [CURSOR_MODE], cx ; save in data area
<1> ;call m16 ; output cx register
<1> ;retn
<1> jmp m16
<1>

```

```

4218 <1>
4219 <1> position:
4220 <1> ; 27/06/2015
4221 <1> ; 02/09/2014
4222 <1> ; 30/08/2014 (Retro UNIX 386 v1)
4223 <1> ; 04/12/2013 (Retro UNIX 8086 v1)
4224 <1> ;
4225 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
4226 <1> ;
4227 <1> ; POSITION
4228 <1> ; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
4229 <1> ; OF A CHARACTER IN THE ALPHA MODE
4230 <1> ; INPUT
4231 <1> ; AX = ROW, COLUMN POSITION
4232 <1> ; OUTPUT
4233 <1> ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
4234 <1>
4235 <1> ; DX = ROW, COLUMN POSITION
4236 <1> ;movzx eax, byte [CRT_COLS] ; 27/06/2015
4237 0000168F 31C0 <1> xor eax, eax ; 02/09/2014
4238 00001691 B050 <1> mov al, 80 ; determine bytes to row
4239 00001693 F6E6 <1> mul dh ; row value
4240 00001695 30F6 <1> xor dh, dh ; 0
4241 00001697 6601D0 <1> add ax, dx ; add column value to the result
4242 0000169A 66D1E0 <1> shl ax, 1 ; * 2 for attribute bytes
4243 <1> ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
4244 0000169D C3 <1> retn
4245 <1>
4246 <1> find_position:
4247 <1> ; 27/06/2015
4248 <1> ; 07/09/2014
4249 <1> ; 02/09/2014
4250 <1> ; 30/08/2014 (Retro UNIX 386 v1)
4251 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
4252 0000169E 0FB6CB <1> movzx ecx, bl ; video page number ; 27/06/2015 (movzx)
4253 000016A1 89CE <1> mov esi, ecx
4254 000016A3 66D1E6 <1> shl si, 1
4255 000016A6 668B96[C6700000] <1> mov dx, [esi + cursor_posn]
4256 000016AD 740A <1> jz short p21
4257 000016AF 6631F6 <1> xor si, si
4258 <1> p20:
4259 <1> ;add si, [CRT_LEN]
4260 000016B2 6681C6A00F <1> add si, 80*25*2 ; add length of buffer for one page
4261 000016B7 E2F9 <1> loop p20
4262 <1> p21:
4263 000016B9 6621D2 <1> and dx, dx
4264 000016BC 7407 <1> jz short p22
4265 000016BE E8CCFFFFFF <1> call position ; determine location in regen in page
4266 000016C3 01C6 <1> add esi, eax ; add location to start of regen page
4267 <1> p22:
4268 <1> ;mov dx, [addr_6845] ; get base address of active display
4269 <1> ;mov dx, 03D4h ; I/O address of color card
4270 <1> ;add dx, 6 ; point at status port
4271 000016C5 66BADA03 <1> mov dx, 03DAh ; status port
4272 <1> ; cx = 0
4273 000016C9 C3 <1> retn
4274 <1>
4275 <1> scroll_up:
4276 <1> ; 07/09/2014
4277 <1> ; 02/09/2014
4278 <1> ; 01/09/2014 (Retro UNIX 386 v1 - beginning)
4279 <1> ; 04/04/2014
4280 <1> ; 04/12/2013
4281 <1> ;
4282 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
4283 <1> ;
4284 <1> ; SCROLL UP
4285 <1> ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
4286 <1> ; ON THE SCREEN
4287 <1> ; INPUT
4288 <1> ; (AH) = CURRENT CRT MODE
4289 <1> ; (AL) = NUMBER OF ROWS TO SCROLL
4290 <1> ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER
4291 <1> ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
4292 <1> ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
4293 <1> ; (DS) = DATA SEGMENT
4294 <1> ; (ES) = REGEN BUFFER SEGMENT
4295 <1> ; OUTPUT
4296 <1> ; NONE -- THE REGEN BUFFER IS MODIFIED
4297 <1> ;
4298 <1> ; bh = 0 (02/09/2014)
4299 <1> ;
4300 <1> ; ((ah = 3))
4301 <1> ; cl = left upper column
4302 <1> ; ch = left upper row
4303 <1> ; dl = right lower column
4304 <1> ; dh = right lower row
4305 <1> ;
4306 <1> ; al = line count
4307 <1> ; ah = attribute to be used on blanked line
4308 <1> ; bl = video page number (0 to 7)
4309 <1> ;

```

4310		<1>
4311		<1> ; Test Line Count
4312 000016CA 08C0		<1> or al, al
4313 000016CC 740C		<1> jz short al_set
4314 000016CE 88F7		<1> mov bh, dh ; subtract lower row from upper row
4315 000016D0 28EF		<1> sub bh, ch
4316 000016D2 FEC7		<1> inc bh ; adjust difference by 1
4317 000016D4 38C7		<1> cmp bh, al ; line count = amount of rows in window?
4318 000016D6 7502		<1> jne short al_set ; if not the we're all set
4319 000016D8 30C0		<1> xor al, al ; otherwise set al to zero
4320		<1> al_set:
4321 000016DA 30FF		<1> xor bh, bh ; 0
4322 000016DC 6650		<1> push ax
4323		<1> ;mov esi, [crt_base]
4324 000016DE BE00800B00		<1> mov esi, 0B8000h
4325 000016E3 3A1D[D6700000]		<1> cmp bl, [active_page]
4326 000016E9 750B		<1> jne short n0
4327		<1> ;
4328 000016EB 66A1[C4700000]		<1> mov ax, [CRT_START]
4329 000016F1 6601C6		<1> add si, ax
4330 000016F4 EB0F		<1> jmp short n1
4331		<1> n0:
4332 000016F6 20DB		<1> and bl, bl
4333 000016F8 740B		<1> jz short n1
4334 000016FA 88D8		<1> mov al, bl
4335		<1> n0x:
4336		<1> ;add si, [CRT_LEN]
4337		<1> ;add esi, 80*25*2
4338 000016FC 6681C6A00F		<1> add si, 80*25*2
4339 00001701 FEC8		<1> dec al
4340 00001703 75F7		<1> jnz short n0x
4341		<1> n1:
4342		<1> ;Scroll position
4343 00001705 6652		<1> push dx
4344 00001707 6689CA		<1> mov dx, cx ; now, upper left position in DX
4345 0000170A E880FFFFFF		<1> call position
4346 0000170F 01C6		<1> add esi, eax
4347 00001711 89F7		<1> mov edi, esi
4348 00001713 665A		<1> pop dx ; lower right position in DX
4349 00001715 6629CA		<1> sub dx, cx
4350 00001718 FEC6		<1> inc dh ; dh = #rows
4351 0000171A FEC2		<1> inc dl ; dl = #cols in block
4352 0000171C 6658		<1> pop ax ; al = line count, ah = attribute
4353 0000171E 31C9		<1> xor ecx, ecx
4354 00001720 6689C1		<1> mov cx, ax
4355		<1> ;mov ah, [CRT_COLS]
4356 00001723 B450		<1> mov ah, 80
4357 00001725 F6E4		<1> mul ah ; determine offset to from address
4358 00001727 6601C0		<1> add ax, ax ; *2 for attribute byte
4359		<1> ;
4360 0000172A 6650		<1> push ax ; offset
4361 0000172C 6652		<1> push dx
4362		<1> ;
4363		<1> ; 04/04/2014
4364 0000172E 66BADA03		<1> mov dx, 3DAh ; guaranteed to be color card here
4365		<1> n8: ; wait_display_enable
4366 00001732 EC		<1> in al, dx ; get port
4367 00001733 A808		<1> test al, RVRT ; wait for vertical retrace
4368 00001735 74FB		<1> jz short n8 ; wait_display_enable
4369 00001737 B025		<1> mov al, 25h
4370 00001739 B2D8		<1> mov dl, 0D8h ; address control port
4371 0000173B EE		<1> out dx, al ; turn off video during vertical retrace
4372 0000173C 665A		<1> pop dx ; #rows, #cols
4373 0000173E 6658		<1> pop ax ; offset
4374 00001740 6691		<1> xchg ax, cx ;
4375		<1> ; ecx = offset, al = line count, ah = attribute
4376		<1> ;n9:
4377 00001742 08C0		<1> or al, al
4378 00001744 7420		<1> jz short n3
4379 00001746 01CE		<1> add esi, ecx ; from address for scroll
4380 00001748 88F7		<1> mov bh, dh ; #rows in block
4381 0000174A 28C7		<1> sub bh, al ; #rows to be moved
4382		<1> n2:
4383		<1> ; Move rows
4384 0000174C 88D1		<1> mov cl, dl ; get # of cols to move
4385 0000174E 56		<1> push esi
4386 0000174F 57		<1> push edi ; save start address
4387		<1> n10:
4388 00001750 66A5		<1> movsw ; move that line on screen
4389 00001752 FEC9		<1> dec cl
4390 00001754 75FA		<1> jnz short n10
4391 00001756 5F		<1> pop edi
4392 00001757 5E		<1> pop esi ; recover addresses
4393		<1> ;mov cl, [CRT_COLS]
4394		<1> ;add cl, cl
4395		<1> ;mov ecx, 80*2
4396 00001758 66B9A000		<1> mov cx, 80*2
4397 0000175C 01CE		<1> add esi, ecx ; next line
4398 0000175E 01CF		<1> add edi, ecx
4399 00001760 FECF		<1> dec bh ; count of lines to move
4400 00001762 75E8		<1> jnz short n2 ; row loop
4401		<1> ; bh = 0



4402	00001764	88C6	<1>	mov	dh, al	; #rows
4403			<1>	n3:		
4404			<1>		; attribute in ah	
4405	00001766	B020	<1>	mov	al, ' '	; fill with blanks
4406			<1>		; Clear rows	
4407			<1>		; dh = #rows	
4408	00001768	88D1	<1>	mov	cl, dl	; get # of cols to clear
4409	0000176A	57	<1>	push	edi	; save address
4410			<1>	n11:		
4411	0000176B	66AB	<1>	stosw		; store fill character
4412	0000176D	FEC9	<1>	dec	cl	
4413	0000176F	75FA	<1>	jnz	short n11	
4414	00001771	5F	<1>	pop	edi	; recover address
4415			<1>	;mov	cl, [CRT_COLS]	
4416			<1>	;add	cl, cl	
4417			<1>	;mov	ecx, 80*2	
4418	00001772	B1A0	<1>	mov	cl, 80*2	
4419	00001774	01CE	<1>	add	esi, ecx	; next line
4420	00001776	01CF	<1>	add	edi, ecx	
4421	00001778	FECE	<1>	dec	dh	
4422	0000177A	75EA	<1>	jnz	short n3	
4423			<1>			
4424	0000177C	3A1D[D6700000]	<1>	cmp	bl, [active_page]	
4425	00001782	7507	<1>	jne	short n6	
4426			<1>	;mov	al, [CRT_MODE_SET]	; get the value of mode set
4427	00001784	B029	<1>	mov	al, 29h	; (ORGS.ASM), M7 mode set table value for mode 3
4428	00001786	66BAD803	<1>	mov	dx, 03D8h	; always set color card port
4429	0000178A	EE	<1>	out	dx, al	
4430			<1>	n6:		
4431	0000178B	C3	<1>	retn		
4432			<1>			
4433			<1>			
4434			<1>	write_c_current:		
4435			<1>		; 30/08/2014 (Retro UNIX 386 v1)	
4436			<1>		; 18/01/2014	
4437			<1>		; 04/12/2013	
4438			<1>			
4439			<1>		; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS	
4440			<1>			
4441			<1>		; WRITE_C_CURRENT	
4442			<1>		; THIS ROUTINE WRITES THE CHARACTER AT	
4443			<1>		; THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED	
4444			<1>		; INPUT	
4445			<1>		; (AH) = CURRENT CRT MODE	
4446			<1>		; (BH) = DISPLAY PAGE	
4447			<1>		; (CX) = COUNT OF CHARACTERS TO WRITE	
4448			<1>		; (AL) = CHAR TO WRITE	
4449			<1>		; (DS) = DATA SEGMENT	
4450			<1>		; (ES) = REGEN SEGMENT	
4451			<1>		; OUTPUT	
4452			<1>		; DISPLAY REGEN BUFFER UPDATED	
4453			<1>			
4454	0000178C	FA	<1>	cli		
4455			<1>		; bl = video page	
4456			<1>		; al = character	
4457			<1>		; ah = color/attribute	
4458	0000178D	6652	<1>	push	dx	
4459	0000178F	6650	<1>	push	ax	; save character & attribute/color
4460	00001791	E808FFFFFF	<1>	call	find_position	; get regen location and port address
4461			<1>		; esi = regen location	
4462			<1>		; dx = status port	
4463			<1>			
4464			<1>		; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE	
4465			<1>			
4466			<1>	p41:		; wait for horizontal retrace is low or vertical
4467	00001796	FB	<1>	sti		; enable interrupts first
4468	00001797	3A1D[D6700000]	<1>	cmp	bl, [active_page]	
4469	0000179D	7510	<1>	jne	short p44	
4470	0000179F	FA	<1>	cli		; block interrupts for single loop
4471	000017A0	EC	<1>	in	al, dx	; get status from the adapter
4472	000017A1	A808	<1>	test	al, RVRT	; check for vertical retrace first
4473	000017A3	7509	<1>	jnz	short p43	; Do fast write now if vertical retrace
4474	000017A5	A801	<1>	test	al, RHRZ	; is horizontal retrace low
4475	000017A7	75ED	<1>	jnz	short p41	; wait until it is
4476			<1>	p42:		; wait for either retrace high
4477	000017A9	EC	<1>	in	al, dx	; get status again
4478	000017AA	A809	<1>	test	al, RVRT+RHRZ	; is horizontal or vertical retrace high
4479	000017AC	74FB	<1>	jz	short p42	; wait until either retrace active
4480			<1>	p43:		
4481	000017AE	FB	<1>	sti		
4482			<1>	p44:		
4483	000017AF	6658	<1>	pop	ax	; restore the character (al) & attribute (ah)
4484	000017B1	81C600800B00	<1>	add	esi, 0B8000h	; 30/08/2014 (crt_base)
4485			<1>			; Retro UNIX 386 v1 feature only!
4486	000017B7	668906	<1>	mov	[esi], ax	
4487	000017BA	665A	<1>	pop	dx	
4488	000017BC	C3	<1>	retn		
4489			<1>			
4490			<1>	set_mode:		
4491			<1>		; 02/09/2014 (Retro UNIX 386 v1)	
4492			<1>			
4493			<1>		; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS	

```

4494 <1>
4495 <1> ;-----
4496 <1> ; SET MODE :
4497 <1> ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO :
4498 <1> ; THE SELECTED MODE, THE SCREEN IS BLANKED. :
4499 <1> ; INPUT :
4500 <1> ; (AL) - MODE SELECTED (RANGE 0-7) :
4501 <1> ; OUTPUT :
4502 <1> ; NONE :
4503 <1> ;-----
4504 <1>
4505 000017BD 53 <1> push ebx
4506 000017BE 52 <1> push edx
4507 000017BF 50 <1> push eax
4508 <1>
4509 <1> ;mov dx, 03D4h ; address or color card
4510 000017C0 B003 <1> mov al, 3
4511 <1> ;M8:
4512 000017C2 A2[906A0000] <1> mov [CRT_MODE], al ; save mode in global variable
4513 000017C7 B029 <1> mov al, 29h
4514 <1> ;mov [CRT_MODE_SET], al ; save the mode set value
4515 000017C9 2437 <1> and al, 037h ; video off, save high resolution bit
4516 <1> ;push dx ; save port value
4517 <1> ;add dx, 4 ; point to control register
4518 000017CB 66BAD803 <1> mov dx, 3D8h
4519 000017CF EE <1> out dx, al ; reset video to off to suppress rolling
4520 <1> ;pop dx
4521 <1> ;M9:
4522 000017D0 30E4 <1> xor ah, ah
4523 000017D2 BB[C86A0000] <1> mov ebx, video_params ; initialization table
4524 <1> ;mov ax, [ebx+10] ; get the cursor mode from the table
4525 <1> ;xchg ah, al
4526 <1> ;mov [CURSOR_MODE], ax ; save cursor mode
4527 000017D7 30E4 <1> xor ah, ah ; ah is register number during loop
4528 <1>
4529 <1> ;----- LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
4530 <1>
4531 <1> M10: ; initialization loop
4532 000017D9 88E0 <1> mov al, ah ; get 6845 register number
4533 000017DB EE <1> out dx, al
4534 000017DC 6642 <1> inc dx ; point to data port
4535 000017DE FEC4 <1> inc ah ; next register value
4536 000017E0 8A03 <1> mov al, [ebx] ; get table value
4537 000017E2 EE <1> out dx, al ; out to chip
4538 000017E3 43 <1> inc ebx ; next in table
4539 000017E4 664A <1> dec dx ; back to pointer register
4540 000017E6 E2F1 <1> loop M10 ; do the whole table
4541 <1>
4542 <1> ;----- FILL REGEN AREA WITH BLANK
4543 <1> ;xor ax, ax
4544 <1> ;mov [CRT_START], ax ; start address saved in global
4545 <1> ;mov [ACTIVE_PAGE], al ; 0 ; (re)set page value
4546 <1> ;mov ecx, 8192 ; number of words in color card
4547 <1> ; black background, light gray character color, space character
4548 <1> ;mov ax, 0720h ; fill char for alpha - attribute
4549 <1> ;M13: ; clear buffer
4550 <1> ;add edi, 0B8000h ; [crt_base]
4551 <1> ;rep stosw ; FILL THE REGEN BUFFER WITH BLANKS
4552 <1>
4553 <1> ;----- ENABLE VIDEO AND CORRECT PORT SETTING
4554 <1> ;mov dx, 3D4h ; mov dx, word [ADDR_6845]
4555 <1> ; ; prepare to output to video enable port
4556 <1> ;add dx, 4 ; point to the mode control gerister
4557 000017E8 66BAD803 <1> mov dx, 3D8h
4558 <1> ;mov al, [CRT_MODE_SET] ; get the mode set value
4559 000017EC B029 <1> mov al, 29h
4560 000017EE EE <1> out dx, al ; set video enable port
4561 <1>
4562 <1> ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
4563 <1> ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
4564 <1> ;
4565 <1> ;mov byte [CRT_COLS], 80h ; initialize number of columns count
4566 <1> ;
4567 <1> ;----- SET CURSOR POSITIONS
4568 000017EF 57 <1> push edi
4569 <1> ;mov word [CRT_LEN], 80*25*2
4570 000017F0 51 <1> push ecx
4571 000017F1 BF[C6700000] <1> mov edi, cursor_posn
4572 000017F6 B904000000 <1> mov ecx, 4 ; clear all cursor positions (16 bytes)
4573 000017FB 31C0 <1> xor eax, eax
4574 000017FD F3AB <1> rep stosd ; fill with zeroes
4575 000017FF 59 <1> pop ecx
4576 00001800 5F <1> pop edi
4577 <1>
4578 <1> ;----- SET UP OVERSCAN REGISTER
4579 00001801 6642 <1> inc dx ; set overscan port to a default
4580 00001803 B030 <1> mov al, 30h ; 30H valuye for all modes except 640X200 bw
4581 <1> ;M14:
4582 00001805 EE <1> out dx, al ; output the correct value to 3D9 port
4583 <1> ;mov [CRT_PALETTE], al ; save the value for future use
4584 <1>
4585 <1> ;----- NORMAL RETURN FROM ALL VIDEO RETURNS

```

```

4586                                     <1> ;
4587 00001806 58                       <1> pop  eax
4588 00001807 5A                       <1> pop  edx
4589 00001808 5B                       <1> pop  ebx
4590 00001809 C3                       <1>  retn
4591                                     <1>
4592                                     <1> tty_sw:
4593                                     <1> ; 30/06/2015
4594                                     <1> ; 27/06/2015
4595                                     <1> ; 07/09/2014
4596                                     <1> ; 02/09/2014 (Retro UNIX 386 v1 - beginning)
4597                                     <1> ;
4598                                     <1> ; (Modified registers : EAX)
4599                                     <1> ;
4600                                     <1> ;mov      byte [u.quant], 0 ; 04/03/2014
4601                                     <1> ;
4602                                     <1> ;act_disp_page:
4603                                     <1> ; 30/06/2015
4604                                     <1> ; 04/03/2014 (act_disp_page --> tty_sw)
4605                                     <1> ; 10/12/2013
4606                                     <1> ; 04/12/2013
4607                                     <1> ;
4608                                     <1> ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
4609                                     <1> ;
4610                                     <1> ; ACT_DISP_PAGE
4611                                     <1> ; THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
4612                                     <1> ; THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
4613                                     <1> ; INPUT
4614                                     <1> ; AL HAS THE NEW ACTIVE DISPLAY PAGE
4615                                     <1> ; OUTPUT
4616                                     <1> ; THE 6845 IS RESET TO DISPLAY THAT PAGE
4617                                     <1>
4618                                     <1> ;cli
4619                                     <1>
4620 0000180A 53                       <1> push  ebx
4621 0000180B 6651                     <1> push  cx
4622 0000180D 6652                     <1> push  dx
4623                                     <1> ;
4624 0000180F A2[D6700000]             <1> mov   [active_page], al ; save active page value ; [ptty]
4625                                     <1> ;mov  cx, [CRT_LEN] ; get saved length of regen buffer
4626 00001814 66B9A00F                 <1> mov   cx, 25*80*2
4627                                     <1> ; 27/06/2015
4628 00001818 0FB6D8                   <1> movzx ebx, al
4629                                     <1> ;
4630 0000181B 6698                       <1> cbw   ; 07/09/2014 (ah=0)
4631 0000181D 66F7E1                   <1> mul   cx ; display page times regen length
4632                                     <1> ; 10/12/2013
4633 00001820 66A3[C4700000]           <1> mov   [CRT_START], ax ; save start address for later
4634 00001826 6689C1                   <1> mov   cx, ax ; start address to cx
4635                                     <1> ;sar  cx, 1
4636 00001829 66D1E9                   <1> shr   cx, 1 ; divide by 2 for 6845 handling
4637 0000182C B40C                     <1> mov   ah, 12 ; 6845 register for start address
4638 0000182E E839FEFFFF               <1> call  m16
4639                                     <1> ;sal  bx, 1
4640                                     <1> ; 01/09/2014
4641 00001833 D0E3                       <1> shl   bl, 1 ; *2 for word offset
4642 00001835 81C3[C6700000]           <1> add   ebx, cursor_posn
4643 0000183B 668B13                   <1> mov   dx, [ebx] ; get cursor for this page
4644 0000183E E815FEFFFF               <1> call  m18
4645                                     <1> ;
4646 00001843 665A                       <1> pop   dx
4647 00001845 6659                       <1> pop   cx
4648 00001847 5B                       <1> pop   ebx
4649                                     <1> ;
4650                                     <1> ;sti
4651                                     <1> ;
4652 00001848 C3                       <1>  retn
4653                                     <1>
4654                                     <1> ; % include 'vidata.inc' ; VIDEO DATA ; 11/03/2015
4655                                     <1>
4656                                     <1>
4657                                     <1> ; /// End Of VIDEO FUNCTIONS ///
4658
4659                                     setup_rtc_int:
4660                                     ; source: http://wiki.osdev.org/RTC
4661 00001849 FA                         cli ; disable interrupts
4662                                     ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b or 6)
4663                                     ; in order to change this ...
4664                                     ; frequency = 32768 >> (rate-1) --> 32768 >> 5 = 1024
4665                                     ; (rate must be above 2 and not over 15)
4666                                     ; new rate = 15 --> 32768 >> (15-1) = 2 Hz
4667 0000184A B08A                       mov   al, 8Ah
4668 0000184C E670                       out   70h, al ; set index to register A, disable NMI
4669 0000184E 90                         nop
4670 0000184F E471                       in    al, 71h ; get initial value of register A
4671 00001851 88C4                       mov   ah, al
4672 00001853 80E4F0                     and   ah, 0F0h
4673 00001856 B08A                       mov   al, 8Ah
4674 00001858 E670                       out   70h, al ; reset index to register A
4675 0000185A 88E0                       mov   al, ah
4676 0000185C 0C0F                       or    al, 0Fh ; new rate (0Fh -> 15)
4677 0000185E E671                       out   71h, al ; write only our rate to A. Note, rate is the bottom 4 bits.

```

```

4678                                     ; enable RTC interrupt
4679 00001860 B08B                     mov  al, 8Bh ;
4680 00001862 E670                     out  70h, al ; select register B and disable NMI
4681 00001864 90                       nop
4682 00001865 E471                     in   al, 71h ; read the current value of register B
4683 00001867 88C4                     mov  ah, al ;
4684 00001869 B08B                     mov  al, 8Bh ;
4685 0000186B E670                     out  70h, al ; set the index again (a read will reset the index to register B)

4686 0000186D 88E0                     mov  al, ah ;
4687 0000186F 0C40                     or   al, 40h ;
4688 00001871 E671                     out  71h, al ; write the previous value ORed with 0x40. This turns on bit 6 of
register B
4689 00001873 FB                       sti
4690 00001874 C3                       retn
4691
4692                                     ; Write memory information
4693                                     ; Temporary Code
4694                                     ; 06/11/2014
4695                                     ; 14/08/2015
4696 memory_info:
4697 00001875 A1[AC700000]              mov  eax, [memory_size] ; in pages
4698 0000187A 50                         push eax
4699 0000187B C1E00C                     shl  eax, 12 ; in bytes
4700 0000187E BB0A000000                mov  ebx, 10
4701 00001883 89D9                       mov  ecx, ebx ; 10
4702 00001885 BE[AD6C0000]              mov  esi, mem_total_b_str
4703 0000188A E8B2000000                call bintdstr
4704 0000188F 58                         pop  eax
4705 00001890 B107                       mov  cl, 7
4706 00001892 BE[D16C0000]              mov  esi, mem_total_p_str
4707 00001897 E8A5000000                call bintdstr
4708                                     ; 14/08/2015
4709 0000189C E8BD000000                call calc_free_mem
4710                                     ; edx = calculated free pages
4711                                     ; ecx = 0
4712 000018A1 A1[B0700000]              mov  eax, [free_pages]
4713 000018A6 39D0                       cmp  eax, edx ; calculated free mem value
4714                                     ; and initial free mem value are same or not?
4715 000018A8 751D                       jne  short pmim ; print mem info with '?' if not
4716 000018AA 52                         push edx ; free memory in pages
4717                                     ;mov  eax, edx
4718 000018AB C1E00C                     shl  eax, 12 ; convert page count
4719                                     ; to byte count
4720 000018AE B10A                       mov  cl, 10
4721 000018B0 BE[F16C0000]              mov  esi, free_mem_b_str
4722 000018B5 E887000000                call bintdstr
4723 000018BA 58                         pop  eax
4724 000018BB B107                       mov  cl, 7
4725 000018BD BE[156D0000]              mov  esi, free_mem_p_str
4726 000018C2 E87A000000                call bintdstr
4727 pmim:
4728 000018C7 BE[9B6C0000]              mov  esi, msg_memory_info
4729 pmim_nb:
4730 000018CC AC                         lodsb
4731 000018CD 08C0                       or   al, al
4732 000018CF 740D                       jz   short pmim_ok
4733 000018D1 56                         push esi
4734 000018D2 31DB                       xor  ebx, ebx ; 0
4735                                     ; Video page 0 (bl=0)
4736 000018D4 B407                       mov  ah, 07h ; Black background,
4737                                     ; light gray forecolor
4738 000018D6 E842FCFFFF                call write_tty
4739 000018DB 5E                         pop  esi
4740 000018DC EBEE                       jmp  short pmim_nb
4741 pmim_ok:
4742 000018DE C3                         retn
4743
4744                                     ; Convert binary number to hexadecimal string
4745                                     ; 10/05/2015
4746                                     ; dsectpm.s (28/02/2015)
4747                                     ; Retro UNIX 386 v1 - Kernel v0.2.0.6
4748                                     ; 01/12/2014
4749                                     ; 25/11/2014
4750                                     ;
4751 bytetohex:
4752                                     ; INPUT ->
4753                                     ; AL = byte (binary number)
4754                                     ; OUTPUT ->
4755                                     ; AX = hexadecimal string
4756                                     ;
4757 000018DF 53                         push ebx
4758 000018E0 31DB                       xor  ebx, ebx
4759 000018E2 88C3                       mov  bl, al
4760 000018E4 C0EB04                     shr  bl, 4
4761 000018E7 8A9B[31190000]            mov  bl, [ebx+hexchrs]
4762 000018ED 86D8                       xchg bl, al
4763 000018EF 80E30F                     and  bl, 0Fh
4764 000018F2 8AA3[31190000]            mov  ah, [ebx+hexchrs]
4765 000018F8 5B                         pop  ebx
4766 000018F9 C3                         retn
4767

```

```

4768
4769
4770
4771
4772
4773
4774 000018FA 53
4775 000018FB 31DB
4776 000018FD 86E0
4777 000018FF 6650
4778 00001901 88E3
4779 00001903 C0EB04
4780 00001906 8A83[31190000]
4781 0000190C 88E3
4782 0000190E 80E30F
4783 00001911 8AA3[31190000]
4784 00001917 C1E010
4785 0000191A 6658
4786 0000191C 5B
4787 0000191D EBC0
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803 0000191F 50
4804 00001920 C1E810
4805 00001923 E8D2FFFFFF
4806 00001928 89C2
4807 0000192A 58
4808 0000192B E8CAFFFFFF
4809 00001930 C3
4810
4811
4812
4813
4814 00001931 303132333435363738-
4815 0000193A 39414243444546
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827 00001941 01CE
4828
4829 00001943 4E
4830 00001944 31D2
4831 00001946 F7F3
4832 00001948 80C230
4833 0000194B 8816
4834 0000194D FEC9
4835 0000194F 740C
4836 00001951 09C0
4837 00001953 75EE
4838
4839 00001955 4E
4840 00001956 C60620
4841 00001959 FEC9
4842 0000195B 75F8
4843
4844 0000195D C3
4845
4846
4847
4848
4849
4850
4851
4852 0000195E 31D2
4853
4854 00001960 668B0D[C0700000]
4855 00001967 C1E10A
4856 0000196A BE00001000
4857
4858 0000196F AD
4859 00001970 51

wordtohex:
; INPUT ->
; AX = word (binary number)
; OUTPUT ->
; EAX = hexadecimal string
;
push ebx
xor ebx, ebx
xchg ah, al
push ax
mov bl, ah
shr bl, 4
mov al, [ebx+hexchrs]
mov bl, ah
and bl, 0Fh
mov ah, [ebx+hexchrs]
shl eax, 16
pop ax
pop ebx
jmp short bytetohex
;mov bl, al
;shr bl, 4
;mov bl, [ebx+hexchrs]
;xchg bl, al
;and bl, 0Fh
;mov ah, [ebx+hexchrs]
;pop ebx
;retn

dwordtohex:
; INPUT ->
; EAX = dword (binary number)
; OUTPUT ->
; EDX:EAX = hexadecimal string
;
push eax
shr eax, 16
call wordtohex
mov edx, eax
pop eax
call wordtohex
retn

; 10/05/2015
hex_digits:
hexchrs:
db '0123456789ABCDEF'

; Convert binary number to decimal/numeric string
; 06/11/2014
; Temporary Code
;

bintdstr:
; EAX = binary number
; ESI = decimal/numeric string address
; EBX = divisor (10)
; ECX = string length (<=10)
add esi, ecx
btdstr0:
dec esi
xor edx, edx
div ebx
add dl, 30h
mov [esi], dl
dec cl
jz btdstr2
or eax, eax
jnz short btdstr0
btdstr1:
dec esi
mov byte [esi], 20h ; blank space
dec cl
jnz short btdstr1
btdstr2:
retn

; Calculate free memory pages on M.A.T.
; 06/11/2014
; Temporary Code
;

calc_free_mem:
xor edx, edx
;xor ecx, ecx
mov cx, [mat_size] ; in pages
shl ecx, 10 ; 1024 dwords per page
mov esi, MEM_ALLOC_TBL
cfm0:
lodsd
push ecx

```

```

4860 00001971 B920000000      mov     ecx, 32
4861                          cfm1:
4862 00001976 D1E8              shr     eax, 1
4863 00001978 7301              jnc     short cfm2
4864 0000197A 42                inc     edx
4865                          cfm2:
4866 0000197B E2F9              loop    cfm1
4867 0000197D 59                pop     ecx
4868 0000197E E2EF              loop    cfm0
4869 00001980 C3                retn
4870
4871                          %include 'diskio.inc' ; 07/03/2015
4872 <1> ; Retro UNIX 386 v1 Kernel - DISKIO.INC
4873 <1> ; Last Modification: 22/08/2015
4874 <1> ;      (Initialized Disk Parameters Data is in 'DISKDATA.INC')
4875 <1> ;      (Uninitialized Disk Parameters Data is in 'DISKBSS.INC')
4876 <1>
4877 <1> ; DISK I/O SYSTEM - Erdogan Tan (Retro UNIX 386 v1 project)
4878 <1>
4879 <1> ; ////////// DISK I/O SYSTEM //////////
4880 <1>
4881 <1> ; 06/02/2015
4882 <1> diskette_io:
4883 00001981 9C                <1> pushfd
4884 00001982 0E                <1> push  cs
4885 00001983 E809000000        <1> call  DISKETTE_IO_1
4886 00001988 C3                <1> retn
4887 <1>
4888 <1> ;;;; DISKETTE I/O ;;;; 06/02/2015 ;;;
4889 <1> ;////////////////////////////////////
4890 <1>
4891 <1> ; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)
4892 <1> ; 20/02/2015
4893 <1> ; 06/02/2015 (unix386.s)
4894 <1> ; 16/12/2014 - 02/01/2015 (dsectrm2.s)
4895 <1> ;
4896 <1> ; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)
4897 <1> ;
4898 <1> ; ADISK.EQU
4899 <1>
4900 <1> ;----- Wait control constants
4901 <1>
4902 <1> ;amount of time to wait while RESET is active.
4903 <1>
4904 <1> WAITCPU_RESET_ON    EQU    21          ;Reset on must last at least 14us
4905 <1>                                     ;at 250 KBS xfer rate.
4906 <1>                                     ;see INTEL MCS, 1985, pg. 5-456
4907 <1>
4908 <1> WAITCPU_FOR_STATUS EQU    100         ;allow 30 microseconds for
4909 <1>                                     ;status register to become valid
4910 <1>                                     ;before re-reading.
4911 <1>
4912 <1> ;After sending a byte to NEC, status register may remain
4913 <1> ;incorrectly set for 24 us.
4914 <1>
4915 <1> WAITCPU_RQM_LOW      EQU    24          ;number of loops to check for
4916 <1>                                     ;RQM low.
4917 <1>
4918 <1> ; COMMON.MAC
4919 <1> ;
4920 <1> ;Timing macros
4921 <1> ;
4922 <1>
4923 <1> %macro      SIODELAY 0                ; SHORT IODELAY
4924 <1>          jmp short $+2
4925 <1> %endmacro
4926 <1>
4927 <1> %macro      IODELAY  0                ; NORMAL IODELAY
4928 <1>          jmp short $+2
4929 <1>          jmp short $+2
4930 <1> %endmacro
4931 <1>
4932 <1> %macro      NEWIODELAY 0
4933 <1>          out    0ebh,al
4934 <1> %endmacro
4935 <1>
4936 <1> ; (According to) AWARD BIOS 1999 - ATORGS.ASM (dw -> equ, db -> equ)
4937 <1> ;;; WAIT FOR MEM
4938 <1> ;WAIT_FDU_INT_LO    equ    017798          ; 2.5 secs in 30 micro units.
4939 <1> ;WAIT_FDU_INT_HI    equ    1
4940 <1> ;WAIT_FDU_INT_LH     equ    83334          ; 27/02/2015 (2.5 seconds waiting)
4941 <1> ;;; WAIT FOR PORT
4942 <1> ;WAIT_FDU_SEND_LO   equ    16667          ; .5 secons in 30 us units.
4943 <1> ;WAIT_FDU_SEND_HI   equ    0
4944 <1> ;WAIT_FDU_SEND_LH   equ    16667          ; 27/02/2015
4945 <1> ;Time to wait while waiting for each byte of NEC results = .5
4946 <1> ;seconds. .5 seconds = 500,000 micros. 500,000/30 = 16,667.
4947 <1> ;WAIT_FDU_RESULTS_LO equ    16667          ; .5 seconds in 30 micro units.
4948 <1> ;WAIT_FDU_RESULTS_HI equ    0
4949 <1> ;WAIT_FDU_RESULTS_LH equ    16667          ; 27/02/2015
4950 <1> ;;; WAIT_REFRESH
4951 <1> ;amount of time to wait for head settle, per unit in parameter

```

```

4952 <1> ;table = 1 ms.
4953 <1> WAIT_FDU_HEAD_SETTLE      equ    33          ; 1 ms in 30 micro units.
4954 <1>
4955 <1>
4956 <1> ; ////////////////////////////////// DISKETTE I/O //////////////////////////////////
4957 <1>
4958 <1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - POSTEQU.INC)
4959 <1>
4960 <1> ;-----
4961 <1> ;EQUATES USED BY POST AND BIOS :
4962 <1> ;-----
4963 <1>
4964 <1> ;----- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----
4965 <1> ;PORT_A      EQU    060H          ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
4966 <1> ;PORT_B      EQU    061H          ; PORT B READ/WRITE DIAGNOSTIC REGISTER
4967 <1> ;REFRESH_BIT EQU    00010000B     ; REFRESH TEST BIT
4968 <1>
4969 <1> ;-----
4970 <1> ;CMOS EQUATES FOR THIS SYSTEM :
4971 <1> ;-----
4972 <1> ;CMOS_PORT    EQU    070H          ; I/O ADDRESS OF CMOS ADDRESS PORT
4973 <1> ;CMOS_DATA    EQU    071H          ; I/O ADDRESS OF CMOS DATA PORT
4974 <1> ;NMI          EQU    10000000B     ; DISABLE NMI INTERRUPTS MASK -
4975 <1> ;              ; HIGH BIT OF CMOS LOCATION ADDRESS
4976 <1>
4977 <1> ;----- CMOS TABLE LOCATION ADDRESS'S ## -----
4978 <1> CMOS_DISKETTE EQU    010H          ; DISKETTE DRIVE TYPE BYTE          ;
4979 <1> ;      EQU    011H          ; - RESERVED                      ;C
4980 <1> CMOS_DISK     EQU    012H          ; FIXED DISK TYPE BYTE              ;H
4981 <1> ;      EQU    013H          ; - RESERVED                      ;E
4982 <1> CMOS_EQUIP    EQU    014H          ; EQUIPMENT WORD LOW BYTE           ;C
4983 <1>
4984 <1> ;----- DISKETTE EQUATES -----
4985 <1> INT_FLAG      EQU    10000000B     ; INTERRUPT OCCURRENCE FLAG
4986 <1> DSK_CHG       EQU    10000000B     ; DISKETTE CHANGE FLAG MASK BIT
4987 <1> DETERMINED    EQU    00010000B     ; SET STATE DETERMINED IN STATE BITS
4988 <1> HOME         EQU    00010000B     ; TRACK 0 MASK
4989 <1> SENSE_DRV_ST EQU    00000100B     ; SENSE DRIVE STATUS COMMAND
4990 <1> TRK_SLAP      EQU    030H          ; CRASH STOP (48 TPI DRIVES)
4991 <1> QUIET_SEEK    EQU    00AH          ; SEEK TO TRACK 10
4992 <1> ;MAX_DRV      EQU    2            ; MAX NUMBER OF DRIVES
4993 <1> HD12_SETTLE   EQU    15            ; 1.2 M HEAD SETTLE TIME
4994 <1> HD320_SETTLE EQU    20            ; 320 K HEAD SETTLE TIME
4995 <1> MOTOR_WAIT    EQU    37            ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
4996 <1>
4997 <1> ;----- DISKETTE ERRORS -----
4998 <1> ;TIME_OUT     EQU    080H          ; ATTACHMENT FAILED TO RESPOND
4999 <1> ;BAD_SEEK     EQU    040H          ; SEEK OPERATION FAILED
5000 <1> BAD_NEC       EQU    020H          ; DISKETTE CONTROLLER HAS FAILED
5001 <1> BAD_CRC       EQU    010H          ; BAD CRC ON DISKETTE READ
5002 <1> MED_NOT_FND   EQU    00CH          ; MEDIA TYPE NOT FOUND
5003 <1> DMA_BOUNDARY  EQU    009H          ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
5004 <1> BAD_DMA       EQU    008H          ; DMA OVERRUN ON OPERATION
5005 <1> MEDIA_CHANGE  EQU    006H          ; MEDIA REMOVED ON DUAL ATTACH CARD
5006 <1> RECORD_NOT_FND EQU    004H          ; REQUESTED SECTOR NOT FOUND
5007 <1> WRITE_PROTECT EQU    003H          ; WRITE ATTEMPTED ON WRITE PROTECT DISK
5008 <1> BAD_ADDR_MARK EQU    002H          ; ADDRESS MARK NOT FOUND
5009 <1> BAD_CMD       EQU    001H          ; BAD COMMAND PASSED TO DISKETTE I/O
5010 <1>
5011 <1> ;----- DISK CHANGE LINE EQUATES -----
5012 <1> NOCHGLN      EQU    001H          ; NO DISK CHANGE LINE AVAILABLE
5013 <1> CHGLN        EQU    002H          ; DISK CHANGE LINE AVAILABLE
5014 <1>
5015 <1> ;----- MEDIA/DRIVE STATE INDICATORS -----
5016 <1> TRK_CAPA      EQU    00000001B     ; 80 TRACK CAPABILITY
5017 <1> FMT_CAPA      EQU    00000010B     ; MULTIPLE FORMAT CAPABILITY (1.2M)
5018 <1> DRV_DET       EQU    00000100B     ; DRIVE DETERMINED
5019 <1> MED_DET       EQU    00010000B     ; MEDIA DETERMINED BIT
5020 <1> DBL_STEP      EQU    00100000B     ; DOUBLE STEP BIT
5021 <1> RATE_MSK      EQU    11000000B     ; MASK FOR CLEARING ALL BUT RATE
5022 <1> RATE_500      EQU    00000000B     ; 500 KBS DATA RATE
5023 <1> RATE_300      EQU    01000000B     ; 300 KBS DATA RATE
5024 <1> RATE_250      EQU    10000000B     ; 250 KBS DATA RATE
5025 <1> STRT_MSK      EQU    00001100B     ; OPERATION START RATE MASK
5026 <1> SEND_MSK      EQU    11000000B     ; MASK FOR SEND RATE BITS
5027 <1>
5028 <1> ;----- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -----
5029 <1> M3D3U         EQU    00000000B     ; 360 MEDIA/DRIVE NOT ESTABLISHED
5030 <1> M3D1U         EQU    00000001B     ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
5031 <1> M1D1U         EQU    00000010B     ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
5032 <1> MED_UNK       EQU    00000111B     ; NONE OF THE ABOVE
5033 <1>
5034 <1> ;----- INTERRUPT EQUATES -----
5035 <1> ;EOI          EQU    020H          ; END OF INTERRUPT COMMAND TO 8259
5036 <1> ;INTA00       EQU    020H          ; 8259 PORT
5037 <1> INTA01        EQU    021H          ; 8259 PORT
5038 <1> INTB00        EQU    0A0H          ; 2ND 8259
5039 <1> INTB01        EQU    0A1H          ;
5040 <1>
5041 <1> ;-----
5042 <1> DMA08         EQU    008H          ; DMA STATUS REGISTER PORT ADDRESS
5043 <1> DMA           EQU    000H          ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS

```

```

5044 <1> DMA18 EQU 0D0H ; 2ND DMA STATUS PORT ADDRESS
5045 <1> DMA1 EQU 0C0H ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
5046 <1> ;-----
5047 <1> ;TIMER EQU 040H ; 8254 TIMER - BASE ADDRESS
5048 <1>
5049 <1> ;-----
5050 <1> DMA_PAGE EQU 081H ; START OF DMA PAGE REGISTERS
5051 <1>
5052 <1> ; 06/02/2015 (unix386.s, protected mode modifications)
5053 <1> ; (unix386.s <-- dsectrm2.s)
5054 <1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - DSEG.INC)
5055 <1>
5056 <1> ; 10/12/2014
5057 <1> ;
5058 <1> ;int40h:
5059 <1> ;pushf
5060 <1> ;push cs
5061 <1> ;;cli
5062 <1> ;call DISKETTE_IO_1
5063 <1> ;ret
5064 <1>
5065 <1> ; DSKETTE ----- 04/21/86 DISKETTE BIOS
5066 <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
5067 <1> ;
5068 <1>
5069 <1> ;-- INT13H -----
5070 <1> ; DISKETTE I/O
5071 <1> ; THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 INCH 360 KB,
5072 <1> ; 1.2 MB, 720 KB AND 1.44 MB DISKETTE DRIVES.
5073 <1> ; INPUT
5074 <1> ; (AH) = 00H RESET DISKETTE SYSTEM
5075 <1> ; HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
5076 <1> ; ON ALL DRIVES
5077 <1> ;-----
5078 <1> ; (AH)= 01H READ THE STATUS OF THE SYSTEM INTO (AH)
5079 <1> ; @DISKETTE_STATUS FROM LAST OPERATION IS USED
5080 <1> ;-----
5081 <1> ; REGISTERS FOR READ/WRITE/VERIFY/FORMAT
5082 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
5083 <1> ; (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
5084 <1> ; (CH) - TRACK NUMBER (NOT VALUE CHECKED)
5085 <1> ; MEDIA DRIVE TRACK NUMBER
5086 <1> ; 320/360 320/360 0-39
5087 <1> ; 320/360 1.2M 0-39
5088 <1> ; 1.2M 1.2M 0-79
5089 <1> ; 720K 720K 0-79
5090 <1> ; 1.44M 1.44M 0-79
5091 <1> ; (CL) - SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
5092 <1> ; MEDIA DRIVE SECTOR NUMBER
5093 <1> ; 320/360 320/360 1-8/9
5094 <1> ; 320/360 1.2M 1-8/9
5095 <1> ; 1.2M 1.2M 1-15
5096 <1> ; 720K 720K 1-9
5097 <1> ; 1.44M 1.44M 1-18
5098 <1> ; (AL) NUMBER OF SECTORS (NOT VALUE CHECKED)
5099 <1> ; MEDIA DRIVE MAX NUMBER OF SECTORS
5100 <1> ; 320/360 320/360 8/9
5101 <1> ; 320/360 1.2M 8/9
5102 <1> ; 1.2M 1.2M 15
5103 <1> ; 720K 720K 9
5104 <1> ; 1.44M 1.44M 18
5105 <1> ;
5106 <1> ; (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
5107 <1> ;
5108 <1> ;-----
5109 <1> ; (AH)= 02H READ THE DESIRED SECTORS INTO MEMORY
5110 <1> ;-----
5111 <1> ; (AH)= 03H WRITE THE DESIRED SECTORS FROM MEMORY
5112 <1> ;-----
5113 <1> ; (AH)= 04H VERIFY THE DESIRED SECTORS
5114 <1> ;-----
5115 <1> ; (AH)= 05H FORMAT THE DESIRED TRACK
5116 <1> ; (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
5117 <1> ; FOR THE TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),
5118 <1> ; WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
5119 <1> ; N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024),
5120 <1> ; THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
5121 <1> ; THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
5122 <1> ; READ/WRITE ACCESS.
5123 <1> ; PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
5124 <1> ; ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
5125 <1> ; THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR 'SET MEDIA TYPE'
5126 <1> ; (INT 13H, AH = 18H) MUST BE CALLED TO SET THE DISKETTE TYPE
5127 <1> ; THAT IS TO BE FORMATTED. IF "SET DASD TYPE" OR "SET MEDIA TYPE"
5128 <1> ; IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE
5129 <1> ; MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
5130 <1> ;
5131 <1> ; THESE PARAMETERS OF DISK BASE MUST BE CHANGED IN ORDER TO
5132 <1> ; FORMAT THE FOLLOWING MEDIAS:
5133 <1> ; -----
5134 <1> ; : MEDIA : DRIVE : PARM 1 : PARM 2 :
5135 <1> ; -----

```



```

5136 <1> ; : 320K : 320K/360K/1.2M : 50H : 8 :
5137 <1> ; : 360K : 320K/360K/1.2M : 50H : 9 :
5138 <1> ; : 1.2M : 1.2M : 54H : 15 :
5139 <1> ; : 720K : 720K/1.44M : 50H : 9 :
5140 <1> ; : 1.44M : 1.44M : 6CH : 18 :
5141 <1> ; -----
5142 <1> ; NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
5143 <1> ; - PARM 2 = EOT (LAST SECTOR ON TRACK)
5144 <1> ; - DISK BASE IS POINTED BY DISK POINTER LOCATED
5145 <1> ; AT ABSOLUTE ADDRESS 0:78.
5146 <1> ; - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
5147 <1> ; SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.

5148 <1> ;-----
5149 <1> ; (AH) = 08H READ DRIVE PARAMETERS
5150 <1> ; REGISTERS
5151 <1> ; INPUT
5152 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
5153 <1> ; OUTPUT
5154 <1> ; (ES:DI) POINTS TO DRIVE PARAMETER TABLE
5155 <1> ; (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
5156 <1> ; (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
5157 <1> ; BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
5158 <1> ; (DH) - MAXIMUM HEAD NUMBER
5159 <1> ; (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
5160 <1> ; (BH) - 0
5161 <1> ; (BL) - BITS 7 THRU 4 - 0
5162 <1> ; BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
5163 <1> ; (AX) - 0
5164 <1> ; UNDER THE FOLLOWING CIRCUMSTANCES:
5165 <1> ; (1) THE DRIVE NUMBER IS INVALID,
5166 <1> ; (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
5167 <1> ; (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
5168 <1> ; (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
5169 <1> ; THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
5170 <1> ; IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
5171 <1> ; @DISKETTE_STATUS = 0 AND CY IS RESET.
5172 <1> ;-----
5173 <1> ; (AH)= 15H READ DASD TYPE
5174 <1> ; OUTPUT REGISTERS
5175 <1> ; (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
5176 <1> ; 00 - DRIVE NOT PRESENT
5177 <1> ; 01 - DISKETTE, NO CHANGE LINE AVAILABLE
5178 <1> ; 02 - DISKETTE, CHANGE LINE AVAILABLE
5179 <1> ; 03 - RESERVED (FIXED DISK)
5180 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
5181 <1> ;-----
5182 <1> ; (AH)= 16H DISK CHANGE LINE STATUS
5183 <1> ; OUTPUT REGISTERS
5184 <1> ; (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
5185 <1> ; 06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
5186 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
5187 <1> ;-----
5188 <1> ; (AH)= 17H SET DASD TYPE FOR FORMAT
5189 <1> ; INPUT REGISTERS
5190 <1> ; (AL) - 00 - NOT USED
5191 <1> ; 01 - DISKETTE 320/360K IN 360K DRIVE
5192 <1> ; 02 - DISKETTE 360K IN 1.2M DRIVE
5193 <1> ; 03 - DISKETTE 1.2M IN 1.2M DRIVE
5194 <1> ; 04 - DISKETTE 720K IN 720K DRIVE
5195 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED:
5196 <1> ; (DO NOT USE WHEN DISKETTE ATTACH CARD USED)
5197 <1> ;-----
5198 <1> ; (AH)= 18H SET MEDIA TYPE FOR FORMAT
5199 <1> ; INPUT REGISTERS
5200 <1> ; (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS
5201 <1> ; (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
5202 <1> ; BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
5203 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)
5204 <1> ; OUTPUT REGISTERS:
5205 <1> ; (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
5206 <1> ; UNCHANGED IF (AH) IS NON-ZERO
5207 <1> ; (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
5208 <1> ; - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
5209 <1> ; - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
5210 <1> ; - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
5211 <1> ;-----
5212 <1> ; DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
5213 <1> ; THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
5214 <1> ; ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
5215 <1> ; ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
5216 <1> ; IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
5217 <1> ; CHANGE ERROR CODE
5218 <1> ; IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
5219 <1> ; TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
5220 <1> ; IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
5221 <1> ;
5222 <1> ; DATA VARIABLE -- @DISK_POINTER
5223 <1> ; DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
5224 <1> ;-----
5225 <1> ; OUTPUT FOR ALL FUNCTIONS
5226 <1> ; AH = STATUS OF OPERATION

```

```

5227 <1> ; STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
5228 <1> ; VARIABLE IN THE DATA SEGMENT OF THIS MODULE
5229 <1> ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD)
5230 <1> ; TYPE AH=(15)).
5231 <1> ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
5232 <1> ; FOR READ/WRITE/VERIFY
5233 <1> ; DS,BX,DX,CX PRESERVED
5234 <1> ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
5235 <1> ; ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
5236 <1> ; ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
5237 <1> ; THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
5238 <1> ; PROBLEM IS NOT DUE TO MOTOR START-UP.
5239 <1> ;-----
5240 <1> ;
5241 <1> ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
5242 <1> ;
5243 <1> ; -----
5244 <1> ; | | | | | | | | | |
5245 <1> ; | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
5246 <1> ; | | | | | | | | | |
5247 <1> ; -----
5248 <1> ; | | | | | | | |
5249 <1> ; | | | | | | | |
5250 <1> ; | | | | | | | |
5251 <1> ; | | | | | | | |
5252 <1> ; | | | | | | | |
5253 <1> ; | | | | | | | |
5254 <1> ; | | | | | | | |
5255 <1> ; | | | | | | | |
5256 <1> ; | | | | | | | |
5257 <1> ; | | | | | | | |
5258 <1> ; | | | | | | | |
5259 <1> ; | | | | | | | |
5260 <1> ; | | | | | | | |
5261 <1> ; | | | | | | | |
5262 <1> ; | | | | | | | |
5263 <1> ; | | | | | | | |
5264 <1> ; | | | | | | | |
5265 <1> ; | | | | | | | |
5266 <1> ; | | | | | | | |
5267 <1> ; -----> DATA TRANSFER RATE FOR THIS DRIVE:
5268 <1> ;
5269 <1> ; 00: 500 KBS
5270 <1> ; 01: 300 KBS
5271 <1> ; 10: 250 KBS
5272 <1> ; 11: RESERVED
5273 <1> ;
5274 <1> ;
5275 <1> ;-----
5276 <1> ; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
5277 <1> ;-----
5278 <1> ; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
5279 <1> ;-----
5280 <1> ;
5281 <1> struc MD
5282 00000000 <res 00000001> <1> .SPEC1 resb 1 ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
5283 00000001 <res 00000001> <1> .SPEC2 resb 1 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
5284 00000002 <res 00000001> <1> .OFF_TIM resb 1 ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
5285 00000003 <res 00000001> <1> .BYT_SEC resb 1 ; 512 BYTES/SECTOR
5286 00000004 <res 00000001> <1> .SEC_TRK resb 1 ; EOT (LAST SECTOR ON TRACK)
5287 00000005 <res 00000001> <1> .GAP resb 1 ; GAP LENGTH
5288 00000006 <res 00000001> <1> .DTL resb 1 ; DTL
5289 00000007 <res 00000001> <1> .GAP3 resb 1 ; GAP LENGTH FOR FORMAT
5290 00000008 <res 00000001> <1> .FIL_BYT resb 1 ; FILL BYTE FOR FORMAT
5291 00000009 <res 00000001> <1> .HD_TIM resb 1 ; HEAD SETTLE TIME (MILLISECONDS)
5292 0000000A <res 00000001> <1> .STR_TIM resb 1 ; MOTOR START TIME (1/8 SECONDS)
5293 0000000B <res 00000001> <1> .MAX_TRK resb 1 ; MAX. TRACK NUMBER
5294 0000000C <res 00000001> <1> .RATE resb 1 ; DATA TRANSFER RATE
5295 <1> endstruc
5296 <1>
5297 <1> BIT7OFF EQU 7FH
5298 <1> BIT7ON EQU 80H
5299 <1>
5300 <1> ;;int13h: ; 16/02/2015
5301 <1> ;; 16/02/2015 - 21/02/2015
5302 <1> int40h:
5303 00001989 9C <1> pushfd
5304 0000198A 0E <1> push cs
5305 0000198B E801000000 <1> call DISKETTE_IO_1
5306 00001990 C3 <1> retn
5307 <1>
5308 <1> DISKETTE_IO_1:
5309 <1>
5310 00001991 FB <1> STI ; INTERRUPTS BACK ON
5311 00001992 55 <1> PUSH eBP ; USER REGISTER
5312 00001993 57 <1> PUSH eDI ; USER REGISTER
5313 00001994 52 <1> PUSH eDX ; HEAD #, DRIVE # OR USER REGISTER
5314 00001995 53 <1> PUSH eBX ; BUFFER OFFSET PARAMETER OR REGISTER
5315 00001996 51 <1> PUSH eCX ; TRACK #-SECTOR # OR USER REGISTER
5316 00001997 89E5 <1> MOV eBP,eSP ; BP => PARAMETER LIST DEP. ON AH
5317 <1> ; [BP] = SECTOR #
5318 <1> ; [BP+1] = TRACK #

```

```

5319                                     <1>                                     ; [BP+2] = BUFFER OFFSET
5320                                     <1>                                     ; FOR RETURN OF DRIVE PARAMETERS:
5321                                     <1>                                     ; CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
5322                                     <1>                                     ;          BITS 0-5 MAX SECTORS/TRACK
5323                                     <1>                                     ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
5324                                     <1>                                     ; BL/[BP+2] = BITS 7-4 = 0
5325                                     <1>                                     ;          BITS 3-0 = VALID CMOS TYPE
5326                                     <1>                                     ; BH/[BP+3] = 0
5327                                     <1>                                     ; DL/[BP+4] = # DRIVES INSTALLED
5328                                     <1>                                     ; DH/[BP+5] = MAX HEAD #
5329                                     <1>                                     ; DI/[BP+6] = OFFSET TO DISK BASE
5330 00001999 06                         <1>  push  es ; 06/02/2015
5331 0000199A 1E                         <1>  PUSH   DS                                     ; BUFFER SEGMENT PARM OR USER REGISTER
5332 0000199B 56                         <1>  PUSH   eSI                                    ; USER REGISTERS
5333                                     <1>  ;CALL  DDS                                     ; SEGMENT OF BIOS DATA AREA TO DS
5334                                     <1>  ;mov   cx,  cs
5335                                     <1>  ;mov   ds,  cx
5336 0000199C 66B91000                   <1>  mov    cx,  KDATA
5337 000019A0 8ED9                       <1>          mov    ds,  cx
5338 000019A2 8EC1                       <1>          mov    es,  cx
5339                                     <1>
5340                                     <1>  ;CMP   AH, (FNC_TAE-FNC_TAB)/2 ; CHECK FOR > LARGEST FUNCTION
5341 000019A4 80FC19                     <1>  cmp    ah, (FNC_TAE-FNC_TAB)/4 ; 18/02/2015
5342 000019A7 7202                       <1>  JB     short OK_FUNC ; FUNCTION OK
5343 000019A9 B414                       <1>  MOV    AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION
5344                                     <1> OK_FUNC:
5345 000019AB 80FC01                     <1>  CMP    AH,1 ; RESET OR STATUS ?
5346 000019AE 760C                       <1>  JBE    short OK_DRV ; IF RESET OR STATUS DRIVE ALWAYS OK
5347 000019B0 80FC08                     <1>  CMP    AH,8 ; READ DRIVE PARMS ?
5348 000019B3 7407                       <1>  JZ     short OK_DRV ; IF SO DRIVE CHECKED LATER
5349 000019B5 80FA01                     <1>  CMP    DL,1 ; DRIVES 0 AND 1 OK
5350 000019B8 7602                       <1>  JBE    short OK_DRV ; IF 0 OR 1 THEN JUMP
5351 000019BA B414                       <1>  MOV    AH,14H ; REPLACE WITH KNOWN INVALID FUNCTION
5352                                     <1> OK_DRV:
5353 000019BC 31C9                       <1>  xor     ecx, ecx
5354                                     <1>  ;mov   esi, ecx ; 08/02/2015
5355 000019BE 89CF                       <1>  mov     edi, ecx ; 08/02/2015
5356 000019C0 88E1                       <1>  MOV     CL,AH ; CL = FUNCTION
5357                                     <1>  ;XOR   CH,CH ; CX = FUNCTION
5358                                     <1>  ;SHL   CL, 1 ; FUNCTION TIMES 2
5359 000019C2 C0E102                     <1>  SHL     CL, 2 ; 20/02/2015 ; FUNCTION TIMES 4 (for 32 bit offset)
5360 000019C5 BB[FD190000]               <1>  MOV     eBX,FNC_TAB ; LOAD START OF FUNCTION TABLE
5361 000019CA 01CB                       <1>  ADD     eBX,eCX ; ADD OFFSET INTO TABLE => ROUTINE
5362 000019CC 88F4                       <1>  MOV     AH,DH ; AX = HEAD #,# OF SECTORS OR DASD TYPE
5363 000019CE 30F6                       <1>  XOR     DH,DH ; DX = DRIVE #
5364 000019D0 6689C6                     <1>  MOV     SI,AX ; SI = HEAD #,# OF SECTORS OR DASD TYPE
5365 000019D3 6689D7                     <1>  MOV     DI,DX ; DI = DRIVE #
5366                                     <1>  ;
5367                                     <1>  ; 11/12/2014
5368 000019D6 8815[516B0000]             <1>          mov     [cfd], dl ; current floppy drive (for 'GET_PARM')
5369                                     <1>  ;
5370 000019DC 8A25[2C710000]             <1>  MOV     AH, [DSKETTE_STATUS] ; LOAD STATUS TO AH FOR STATUS FUNCTION
5371 000019E2 C605[2C710000]00           <1>  MOV     byte [DSKETTE_STATUS],0 ; INITIALIZE FOR ALL OTHERS
5372                                     <1>
5373                                     <1> ; THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
5374                                     <1> ; THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
5375                                     <1> ; FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
5376                                     <1> ;
5377                                     <1> ;      DI      : DRIVE #
5378                                     <1> ;      SI-HI   : HEAD #
5379                                     <1> ;      SI-LOW  : # OF SECTORS OR DASD TYPE FOR FORMAT
5380                                     <1> ;      ES      : BUFFER SEGMENT
5381                                     <1> ;      [BP]    : SECTOR #
5382                                     <1> ;      [BP+1]  : TRACK #
5383                                     <1> ;      [BP+2]  : BUFFER OFFSET
5384                                     <1> ;
5385                                     <1> ; ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
5386                                     <1> ; SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
5387                                     <1> ; CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
5388                                     <1> ; SPECIFIC ERROR CODE.
5389                                     <1> ;
5390                                     <1>                                     ; (AH) = @DSKETTE_STATUS
5391 000019E9 FF13                         <1>  CALL    dword [eBX] ; CALL THE REQUESTED FUNCTION
5392 000019EB 5E                         <1>  POP     eSI ; RESTORE ALL REGISTERS
5393 000019EC 1F                         <1>  POP     DS
5394 000019ED 07                         <1>  pop     es ; 06/02/2015
5395 000019EE 59                         <1>  POP     eCX
5396 000019EF 5B                         <1>  POP     eBX
5397 000019F0 5A                         <1>  POP     eDX
5398 000019F1 5F                         <1>  POP     eDI
5399 000019F2 89E5                       <1>  MOV     eBP, eSP
5400 000019F4 50                         <1>  PUSH    eAX
5401 000019F5 9C                         <1>  PUSHF   d
5402 000019F6 58                         <1>  POP     eAX
5403                                     <1>  ;MOV   [BP+6], AX
5404 000019F7 89450C                     <1>  mov     [ebp+12], eax ; 18/02/2015, flags
5405 000019FA 58                         <1>  POP     eAX
5406 000019FB 5D                         <1>  POP     eBP
5407 000019FC CF                         <1>  IRETD
5408                                     <1>
5409                                     <1> ;-----
5410                                     <1> ; DW --> dd (06/02/2015)

```

```

5411 000019FD [611A0000] <1> FNC_TABdd DSK_RESET ; AH = 00H; RESET
5412 00001A01 [DA1A0000] <1> dd DSK_STATUS ; AH = 01H; STATUS
5413 00001A05 [EB1A0000] <1> dd DSK_READ ; AH = 02H; READ
5414 00001A09 [FC1A0000] <1> dd DSK_WRITE ; AH = 03H; WRITE
5415 00001A0D [0D1B0000] <1> dd DSK_VERF ; AH = 04H; VERIFY
5416 00001A11 [1E1B0000] <1> dd DSK_FORMAT ; AH = 05H; FORMAT
5417 00001A15 [A31B0000] <1> dd FNC_ERR ; AH = 06H; INVALID
5418 00001A19 [A31B0000] <1> dd FNC_ERR ; AH = 07H; INVALID
5419 00001A1D [B01B0000] <1> dd DSK_PARMS ; AH = 08H; READ DRIVE PARAMETERS
5420 00001A21 [A31B0000] <1> dd FNC_ERR ; AH = 09H; INVALID
5421 00001A25 [A31B0000] <1> dd FNC_ERR ; AH = 0AH; INVALID
5422 00001A29 [A31B0000] <1> dd FNC_ERR ; AH = 0BH; INVALID
5423 00001A2D [A31B0000] <1> dd FNC_ERR ; AH = 0CH; INVALID
5424 00001A31 [A31B0000] <1> dd FNC_ERR ; AH = 0DH; INVALID
5425 00001A35 [A31B0000] <1> dd FNC_ERR ; AH = 0EH; INVALID
5426 00001A39 [A31B0000] <1> dd FNC_ERR ; AH = 0FH; INVALID
5427 00001A3D [A31B0000] <1> dd FNC_ERR ; AH = 10H; INVALID
5428 00001A41 [A31B0000] <1> dd FNC_ERR ; AH = 11H; INVALID
5429 00001A45 [A31B0000] <1> dd FNC_ERR ; AH = 12H; INVALID
5430 00001A49 [A31B0000] <1> dd FNC_ERR ; AH = 13H; INVALID
5431 00001A4D [A31B0000] <1> dd FNC_ERR ; AH = 14H; INVALID
5432 00001A51 [711C0000] <1> dd DSK_TYPE ; AH = 15H; READ DASD TYPE
5433 00001A55 [9C1C0000] <1> dd DSK_CHANGE ; AH = 16H; CHANGE STATUS
5434 00001A59 [D61C0000] <1> dd FORMAT_SET ; AH = 17H; SET DASD TYPE
5435 00001A5D [591D0000] <1> dd SET_MEDIA ; AH = 18H; SET MEDIA TYPE
5436 <1> FNC_TAE EQU $ ; END
5437 <1>
5438 <1> ;-----
5439 <1> ; DISK_RESET (AH = 00H)
5440 <1> ; RESET THE DISKETTE SYSTEM.
5441 <1> ;
5442 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5443 <1> ;-----
5444 <1> DSK_RESET:
5445 00001A61 66BAF203 <1> MOV DX,03F2H ; ADAPTER CONTROL PORT
5446 00001A65 FA <1> CLI ; NO INTERRUPTS
5447 00001A66 A0[2A710000] <1> MOV AL,[MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
5448 00001A6B 243F <1> AND AL,00111111B ; KEEP SELECTED AND MOTOR ON BITS
5449 00001A6D C0C004 <1> ROL AL,4 ; MOTOR VALUE TO HIGH NIBBLE
5450 <1> ; DRIVE SELECT TO LOW NIBBLE
5451 00001A70 0C08 <1> OR AL,00001000B ; TURN ON INTERRUPT ENABLE
5452 00001A72 EE <1> OUT DX,AL ; RESET THE ADAPTER
5453 00001A73 C605[29710000]00 <1> MOV byte [SEEK_STATUS],0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES
5454 <1> ;JMP $+2 ; WAIT FOR I/O
5455 <1> ;JMP $+2 ; WAIT FOR I/O (TO INSURE MINIMUM
5456 <1> ; PULSE WIDTH)
5457 <1> ; 19/12/2014
5458 <1> NEWIODELAY
5459 00001A7A E6EB <2> out 0ebh,al
5460 <1>
5461 <1> ; 17/12/2014
5462 <1> ; AWARD BIOS 1999 - RESETDRIVES (ADISK.ASM)
5463 00001A7C B915000000 <1> mov ecx, WAITCPU_RESET_ON ; cx = 21 -- Min. 14 micro seconds !?
5464 <1> wdw1:
5465 <1> NEWIODELAY ; 27/02/2015
5466 00001A81 E6EB <2> out 0ebh,al
5467 00001A83 E2FC <1> loop wdw1
5468 <1> ;
5469 00001A85 0C04 <1> OR AL,00000100B ; TURN OFF RESET BIT
5470 00001A87 EE <1> OUT DX,AL ; RESET THE ADAPTER
5471 <1> ; 16/12/2014
5472 <1> IODELAY
5473 00001A88 EB00 <2> jmp short $+2
5474 00001A8A EB00 <2> jmp short $+2
5475 <1> ;
5476 <1> ;STI ; ENABLE THE INTERRUPTS
5477 00001A8C E8240C0000 <1> CALL WAIT_INT ; WAIT FOR THE INTERRUPT
5478 00001A91 723E <1> JC short DR_ERR ; IF ERROR, RETURN IT
5479 00001A93 66B9C000 <1> MOV CX,11000000B ; CL = EXPECTED @NEC_STATUS
5480 <1> NXT_DRV:
5481 00001A97 6651 <1> PUSH CX ; SAVE FOR CALL
5482 00001A99 B8[CF1A0000] <1> MOV eAX, DR_POP_ERR ; LOAD NEC_OUTPUT ERROR ADDRESS
5483 00001A9E 50 <1> PUSH eAX ; "
5484 00001A9F B408 <1> MOV AH,08H ; SENSE INTERRUPT STATUS COMMAND
5485 00001AA1 E8020B0000 <1> CALL NEC_OUTPUT
5486 00001AA6 58 <1> POP eAX ; THROW AWAY ERROR RETURN
5487 00001AA7 E8390C0000 <1> CALL RESULTS ; READ IN THE RESULTS
5488 00001AAC 6659 <1> POP CX ; RESTORE AFTER CALL
5489 00001AAE 7221 <1> JC short DR_ERR ; ERROR RETURN
5490 00001AB0 3A0D[2D710000] <1> CMP CL, [NEC_STATUS] ; TEST FOR DRIVE READY TRANSITION
5491 00001AB6 7519 <1> JNZ short DR_ERR ; EVERYTHING OK
5492 00001AB8 FEC1 <1> INC CL ; NEXT EXPECTED @NEC_STATUS
5493 00001ABA 80F9C3 <1> CMP CL,11000011B ; ALL POSSIBLE DRIVES CLEARED
5494 00001ABD 76D8 <1> JBE short NXT_DRV ; FALL THRU IF 11000100B OR >
5495 <1> ;
5496 00001ABF E852030000 <1> CALL SEND_SPEC ; SEND SPECIFY COMMAND TO NEC
5497 <1> RESBAC:
5498 00001AC4 E805090000 <1> CALL SETUP_END ; VARIOUS CLEANUPS
5499 00001AC9 6689F3 <1> MOV BX,SI ; GET SAVED AL TO BL
5500 00001ACC 88D8 <1> MOV AL,BL ; PUT BACK FOR RETURN
5501 00001ACE C3 <1> RETn
5502 <1> DR_POP_ERR:

```

5503	00001ACF	6659	<1>	POP	CX		; CLEAR STACK
5504			<1>	DR_ERR:			
5505	00001AD1	800D[2C710000]20	<1>	OR	byte [DSKETTE_STATUS],BAD_NEC		; SET ERROR CODE
5506	00001AD8	E8EA	<1>	JMP	SHORT RESBAC		; RETURN FROM RESET
5507			<1>				
5508			<1>				-----
5509			<1>				; DISK_STATUS (AH = 01H)
5510			<1>				; DISKETTE STATUS.
5511			<1>				;
5512			<1>				; ON ENTRY: AH : STATUS OF PREVIOUS OPERATION
5513			<1>				;
5514			<1>				; ON EXIT: AH, @DSKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
5515			<1>				-----
5516			<1>	DSK_STATUS:			
5517	00001ADA	8825[2C710000]	<1>	MOV	[DSKETTE_STATUS],AH		; PUT BACK FOR SETUP END
5518	00001AE0	E8E9080000	<1>	CALL	SETUP_END		; VARIOUS CLEANUPS
5519	00001AE5	6689F3	<1>	MOV	BX,SI		; GET SAVED AL TO BL
5520	00001AE8	88D8	<1>	MOV	AL,BL		; PUT BACK FOR RETURN
5521	00001AEA	C3	<1>	RETn			
5522			<1>				
5523			<1>				-----
5524			<1>				; DISK_READ (AH = 02H)
5525			<1>				; DISKETTE READ.
5526			<1>				;
5527			<1>				; ON ENTRY: DI : DRIVE #
5528			<1>				; SI-HI : HEAD #
5529			<1>				; SI-LOW : # OF SECTORS
5530			<1>				; ES : BUFFER SEGMENT
5531			<1>				; [BP] : SECTOR #
5532			<1>				; [BP+1] : TRACK #
5533			<1>				; [BP+2] : BUFFER OFFSET
5534			<1>				;
5535			<1>				; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5536			<1>				-----
5537			<1>				
5538			<1>				; 06/02/2015, ES:BX -> EBX (unix386.s)
5539			<1>				
5540			<1>	DSK_READ:			
5541	00001AEB	8025[2A710000]7F	<1>	AND	byte [MOTOR_STATUS],01111111B		; INDICATE A READ OPERATION
5542	00001AF2	66B846E6	<1>	MOV	AX,0E646H		; AX = NEC COMMAND, DMA COMMAND
5543	00001AF6	E825040000	<1>	CALL	RD_WR_VF		; COMMON READ/WRITE/VERIFY
5544	00001AFB	C3	<1>	RETn			
5545			<1>				
5546			<1>				-----
5547			<1>				; DISK_WRITE (AH = 03H)
5548			<1>				; DISKETTE WRITE.
5549			<1>				;
5550			<1>				; ON ENTRY: DI : DRIVE #
5551			<1>				; SI-HI : HEAD #
5552			<1>				; SI-LOW : # OF SECTORS
5553			<1>				; ES : BUFFER SEGMENT
5554			<1>				; [BP] : SECTOR #
5555			<1>				; [BP+1] : TRACK #
5556			<1>				; [BP+2] : BUFFER OFFSET
5557			<1>				;
5558			<1>				; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5559			<1>				-----
5560			<1>				
5561			<1>				; 06/02/2015, ES:BX -> EBX (unix386.s)
5562			<1>				
5563			<1>	DSK_WRITE:			
5564	00001AFC	66B84AC5	<1>	MOV	AX,0C54AH		; AX = NEC COMMAND, DMA COMMAND
5565	00001B00	800D[2A710000]80	<1>		OR	byte [MOTOR_STATUS],10000000B	; INDICATE WRITE OPERATION
5566	00001B07	E814040000	<1>	CALL	RD_WR_VF		; COMMON READ/WRITE/VERIFY
5567	00001B0C	C3	<1>	RETn			
5568			<1>				
5569			<1>				-----
5570			<1>				; DISK_VERIFY (AH = 04H)
5571			<1>				; DISKETTE VERIFY.
5572			<1>				;
5573			<1>				; ON ENTRY: DI : DRIVE #
5574			<1>				; SI-HI : HEAD #
5575			<1>				; SI-LOW : # OF SECTORS
5576			<1>				; ES : BUFFER SEGMENT
5577			<1>				; [BP] : SECTOR #
5578			<1>				; [BP+1] : TRACK #
5579			<1>				; [BP+2] : BUFFER OFFSET
5580			<1>				;
5581			<1>				; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5582			<1>				-----
5583			<1>	DSK_VERIFY:			
5584	00001B0D	8025[2A710000]7F	<1>	AND	byte [MOTOR_STATUS],01111111B		; INDICATE A READ OPERATION
5585	00001B14	66B842E6	<1>	MOV	AX,0E642H		; AX = NEC COMMAND, DMA COMMAND
5586	00001B18	E803040000	<1>	CALL	RD_WR_VF		; COMMON READ/WRITE/VERIFY
5587	00001B1D	C3	<1>	RETn			
5588			<1>				
5589			<1>				-----
5590			<1>				; DISK_FORMAT (AH = 05H)
5591			<1>				; DISKETTE FORMAT.
5592			<1>				;
5593			<1>				; ON ENTRY: DI : DRIVE #
5594			<1>				; SI-HI : HEAD #

```

5595 <1> ; SI-LOW : # OF SECTORS
5596 <1> ; ES : BUFFER SEGMENT
5597 <1> ; [BP] : SECTOR #
5598 <1> ; [BP+1] : TRACK #
5599 <1> ; [BP+2] : BUFFER OFFSET
5600 <1> ; @DISK_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE
5601 <1> ;
5602 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5603 <1> ;-----
5604 <1> DSK_FORMAT:
5605 00001B1E E83C030000 <1> CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
5606 00001B23 E838050000 <1> CALL FMT_INIT ; ESTABLISH STATE IF UNESTABLISHED
5607 00001B28 800D[2A710000]80 <1> OR byte [MOTOR_STATUS], 10000000B ; INDICATE WRITE OPERATION
5608 00001B2F E880050000 <1> CALL MED_CHANGE ; CHECK MEDIA CHANGE AND RESET IF SO
5609 00001B34 725D <1> JC short FM_DON ; MEDIA CHANGED, SKIP
5610 00001B36 E8DB020000 <1> CALL SEND_SPEC ; SEND SPECIFY COMMAND TO NEC
5611 00001B3B E8E6050000 <1> CALL CHK_LASRATE ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
5612 00001B40 7405 <1> JZ short FM_WR ; YES, SKIP SPECIFY COMMAND
5613 00001B42 E8BD050000 <1> CALL SEND_RATE ; SEND DATA RATE TO CONTROLLER
5614 <1> FM_WR:
5615 00001B47 E872060000 <1> CALL FMTDMA_SET ; SET UP THE DMA FOR FORMAT
5616 00001B4C 7245 <1> JC short FM_DON ; RETURN WITH ERROR
5617 00001B4E B44D <1> MOV AH,04DH ; ESTABLISH THE FORMAT COMMAND
5618 00001B50 E8CF060000 <1> CALL NEC_INIT ; INITIALIZE THE NEC
5619 00001B55 723C <1> JC short FM_DON ; ERROR - EXIT
5620 00001B57 B8[931B0000] <1> MOV eAX, FM_DON ; LOAD ERROR ADDRESS
5621 00001B5C 50 <1> PUSH eAX ; PUSH NEC_OUT ERROR RETURN
5622 00001B5D B203 <1> MOV DL,3 ; BYTES/SECTOR VALUE TO NEC
5623 00001B5F E83E090000 <1> CALL GET_PARM
5624 00001B64 E83F0A0000 <1> CALL NEC_OUTPUT
5625 00001B69 B204 <1> MOV DL,4 ; SECTORS/TRACK VALUE TO NEC
5626 00001B6B E832090000 <1> CALL GET_PARM
5627 00001B70 E8330A0000 <1> CALL NEC_OUTPUT
5628 00001B75 B207 <1> MOV DL,7 ; GAP LENGTH VALUE TO NEC
5629 00001B77 E826090000 <1> CALL GET_PARM
5630 00001B7C E8270A0000 <1> CALL NEC_OUTPUT
5631 00001B81 B208 <1> MOV DL,8 ; FILLER BYTE TO NEC
5632 00001B83 E81A090000 <1> CALL GET_PARM
5633 00001B88 E81B0A0000 <1> CALL NEC_OUTPUT
5634 00001B8D 58 <1> POP eAX ; THROW AWAY ERROR
5635 00001B8E E80F070000 <1> CALL NEC_TERM ; TERMINATE, RECEIVE STATUS, ETC,
5636 <1> FM_DON:
5637 00001B93 E8F8020000 <1> CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
5638 00001B98 E831080000 <1> CALL SETUP_END ; VARIOUS CLEANUPS
5639 00001B9D 6689F3 <1> MOV BX,SI ; GET SAVED AL TO BL
5640 00001BA0 88D8 <1> MOV AL,BL ; PUT BACK FOR RETURN
5641 00001BA2 C3 <1> RETn
5642 <1>
5643 <1> ;-----
5644 <1> ; FNC_ERR
5645 <1> ; INVALID FUNCTION REQUESTED OR INVALID DRIVE:
5646 <1> ; SET BAD COMMAND IN STATUS.
5647 <1> ;
5648 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
5649 <1> ;-----
5650 <1> FNC_ERR: ; INVALID FUNCTION REQUEST
5651 00001BA3 6689F0 <1> MOV AX,SI ; RESTORE AL
5652 00001BA6 B401 <1> MOV AH,BAD_CMD ; SET BAD COMMAND ERROR
5653 00001BA8 8825[2C710000] <1> MOV [DSKETTE_STATUS],AH ; STORE IN DATA AREA
5654 00001BAE F9 <1> STC ; SET CARRY INDICATING ERROR
5655 00001BAF C3 <1> RETn
5656 <1>
5657 <1> ;-----
5658 <1> ; DISK_PARMS (AH = 08H)
5659 <1> ; READ DRIVE PARAMETERS.
5660 <1> ;
5661 <1> ; ON ENTRY: DI : DRIVE #
5662 <1> ;
5663 <1> ; ON EXIT: CL/[BP] = BITS 7 & 6 HI 2 BITS OF MAX CYLINDER
5664 <1> ; BITS 0-5 MAX SECTORS/TRACK
5665 <1> ; CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
5666 <1> ; BL/[BP+2] = BITS 7-4 = 0
5667 <1> ; BITS 3-0 = VALID CMOS DRIVE TYPE
5668 <1> ; BH/[BP+3] = 0
5669 <1> ; DL/[BP+4] = # DRIVES INSTALLED (VALUE CHECKED)
5670 <1> ; DH/[BP+5] = MAX HEAD #
5671 <1> ; DI/[BP+6] = OFFSET TO DISK_BASE
5672 <1> ; ES = SEGMENT OF DISK_BASE
5673 <1> ; AX = 0
5674 <1> ;
5675 <1> ; NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
5676 <1> ; THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
5677 <1> ; INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
5678 <1> ; CALLER.
5679 <1> ;-----
5680 <1> DSK_PARMS:
5681 00001BB0 E8AA020000 <1> CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH,
5682 <1> ; MOV WORD [BP+2],0 ; DRIVE TYPE = 0
5683 00001BB5 29D2 <1> sub edx, edx ; 20/02/2015
5684 00001BB7 895504 <1> mov [ebp+4], edx ; 20/02/2015
5685 <1> ; MOV AX, [EQUIP_FLAG] ; LOAD EQUIPMENT FLAG FOR # DISKETTES
5686 <1> ; AND AL,11000001B ; KEEP DISKETTE DRIVE BITS

```

```

5687      <1>      ; MOV      DL,2                ; DISKETTE DRIVES = 2
5688      <1>      ; CMP      AL,01000001B        ; 2 DRIVES INSTALLED ?
5689      <1>      ; JZ       short STO_DL          ; IF YES JUMP
5690      <1>      ; DEC      DL                ; DISKETTE DRIVES = 1
5691      <1>      ; CMP      AL,00000001B        ; 1 DRIVE INSTALLED ?
5692      <1>      ; JNZ      short NON_DRV         ; IF NO JUMP
5693      <1>      ;sub     edx, edx
5694 00001BBA 66A1[D86D0000] <1> mov     ax, [fd0_type]
5695 00001BC0 6621C0 <1> and     ax, ax
5696 00001BC3 7474 <1> jz      short NON_DRV
5697 00001BC5 FEC2 <1> inc     dl
5698 00001BC7 20E4 <1> and     ah, ah
5699 00001BC9 7402 <1> jz      short STO_DL
5700 00001BCB FEC2 <1> inc     dl
5701 <1> STO_DL:
5702 <1> ;MOV [BP+4],DL ; STORE NUMBER OF DRIVES
5703 00001BCD 895508 <1> mov     [ebp+8], edx ; 20/02/2015
5704 00001BD0 6683FF01 <1> CMP     DI,1 ; CHECK FOR VALID DRIVE
5705 00001BD4 7766 <1> JA      short NON_DRV1 ; DRIVE INVALID
5706 <1> ;MOV BYTE [BP+5],1 ; MAXIMUM HEAD NUMBER = 1
5707 00001BD6 C6450901 <1> mov     byte [ebp+9], 1 ; 20/02/2015
5708 00001BDA E8BA080000 <1> CALL    CMOS_TYPE ; RETURN DRIVE TYPE IN AL
5709 <1> ;;20/02/2015
5710 <1> ;;JC short CHK_EST ; IF CMOS BAD CHECKSUM ESTABLISHED
5711 <1> ;;OR AL,AL ; TEST FOR NO DRIVE TYPE
5712 00001BDF 7412 <1> JZ      short CHK_EST ; JUMP IF SO
5713 00001BE1 E805020000 <1> CALL    DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5714 00001BE6 720B <1> JC      short CHK_EST ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
5715 <1> ;MOV [BP+2],AL ; STORE VALID CMOS DRIVE TYPE
5716 00001BE8 884504 <1> mov     [ebp+4], al ; 06/02/2015
5717 00001BEB 8A4B04 <1> MOV     CL, [ebx+MD.SEC_TRK] ; GET SECTOR/TRACK
5718 00001BEE 8A6B0B <1> MOV     CH, [ebx+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
5719 00001BF1 EB36 <1> JMP     SHORT STO_CX ; CMOS GOOD, USE CMOS
5720 <1> CHK_EST:
5721 00001BF3 8AA7[39710000] <1> MOV     AH, [DSK_STATE+eDI] ; LOAD STATE FOR THIS DRIVE
5722 00001BF9 F6C410 <1> TEST    AH,MED_DET ; CHECK FOR ESTABLISHED STATE
5723 00001BFC 743E <1> JZ      short NON_DRV1 ; CMOS BAD/INVALID OR UNESTABLISHED
5724 <1> USE_EST:
5725 00001BFE 80E4C0 <1> AND     AH,RATE_MSK ; ISOLATE STATE
5726 00001C01 80FC80 <1> CMP     AH,RATE_250 ; RATE 250 ?
5727 00001C04 7557 <1> JNE     short USE_EST2 ; NO, GO CHECK OTHER RATE
5728 <1>
5729 <1> ;----- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
5730 <1>
5731 00001C06 B001 <1> MOV     AL,01 ; DRIVE TYPE 1 (360KB)
5732 00001C08 E8DE010000 <1> CALL    DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5733 00001C0D 8A4B04 <1> MOV     CL, [ebx+MD.SEC_TRK] ; GET SECTOR/TRACK
5734 00001C10 8A6B0B <1> MOV     CH, [ebx+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
5735 00001C13 F687[39710000]01 <1> TEST    byte [DSK_STATE+eDI],TRK_CAPA ; 80 TRACK ?
5736 00001C1A 740D <1> JZ      short STO_CX ; MUST BE 360KB DRIVE
5737 <1>
5738 <1> ;----- IT IS 1.44 MB DRIVE
5739 <1>
5740 <1> PARM144:
5741 00001C1C B004 <1> MOV     AL,04 ; DRIVE TYPE 4 (1.44MB)
5742 00001C1E E8C8010000 <1> CALL    DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5743 00001C23 8A4B04 <1> MOV     CL, [ebx+MD.SEC_TRK] ; GET SECTOR/TRACK
5744 00001C26 8A6B0B <1> MOV     CH, [ebx+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
5745 <1> STO_CX:
5746 00001C29 894D00 <1> MOV     [ebp],ecx ; SAVE POINTER IN STACK FOR RETURN
5747 <1> ES_DI:
5748 <1> ;MOV [BP+6],BX ; ADDRESS OF MEDIA/DRIVE PARM TABLE
5749 00001C2C 895D0C <1> mov     [ebp+12], ebx ; 06/02/2015
5750 <1> ;MOV AX,CS ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
5751 <1> ;MOV ES,AX ; ES IS SEGMENT OF TABLE
5752 <1> DP_OUT:
5753 00001C2F E85C020000 <1> CALL    XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
5754 00001C34 6631C0 <1> XOR     AX,AX ; CLEAR
5755 00001C37 F8 <1> CLC
5756 00001C38 C3 <1> RETn
5757 <1>
5758 <1> ;----- NO DRIYE PRESENT HANDLER
5759 <1>
5760 <1> NON_DRV:
5761 <1> ;MOV BYTE [BP+4],0 ; CLEAR NUMBER OF DRIVES
5762 00001C39 895508 <1> mov     [ebp+8], edx ; 0 ; 20/02/2015
5763 <1> NON_DRV1:
5764 00001C3C 6681FF8000 <1> CMP     DI,80H ; CHECK FOR FIXED MEDIA TYPE REQUEST
5765 00001C41 720C <1> JB      short NON_DRV2 ; CONTINUE IF NOT REQUEST FALL THROUGH
5766 <1>
5767 <1> ;----- FIXED DISK REQUEST FALL THROUGH ERROR
5768 <1>
5769 00001C43 E848020000 <1> CALL    XLAT_OLD ; ELSE TRANSLATE TO COMPATIBLE MODE
5770 00001C48 6689F0 <1> MOV     AX,SI ; RESTORE AL
5771 00001C4B B401 <1> MOV     AH,BAD_CMD ; SET BAD COMMAND ERROR
5772 00001C4D F9 <1> STC
5773 00001C4E C3 <1> RETn
5774 <1>
5775 <1> NON_DRV2:
5776 <1> ;XOR AX,AX ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
5777 00001C4F 31C0 <1> xor     eax, eax
5778 00001C51 66894500 <1> MOV     [ebp],AX ; TRACKS, SECTORS/TRACK = 0

```

5779		<1>	;MOV [BP+5],AH ; HEAD = 0
5780	00001C55 886509	<1>	mov [ebp+9], ah ; 06/02/2015
5781		<1>	;MOV [BP+6],AX ; OFFSET TO DISK_BASE = 0
5782	00001C58 89450C	<1>	mov [ebp+12], eax
5783		<1>	;MOV ES,AX ; ES IS SEGMENT OF TABLE
5784	00001C5B EBD2	<1>	JMP SHORT DP_OUT
5785		<1>	
5786		<1>	;----- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
5787		<1>	
5788		<1>	USE_EST2:
5789	00001C5D B002	<1>	MOV AL,02 ; DRIVE TYPE 2 (1.2MB)
5790	00001C5F E887010000	<1>	CALL DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5791	00001C64 8A4B04	<1>	MOV CL, [eBX+MD.SEC_TRK] ; GET SECTOR/TRACK
5792	00001C67 8A6B0B	<1>	MOV CH, [eBX+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
5793	00001C6A 80FC40	<1>	CMP AH,RATE_300 ; RATE 300 ?
5794	00001C6D 74BA	<1>	JZ short STO_CX ; MUST BE 1.2MB DRIVE
5795	00001C6F EBAB	<1>	JMP SHORT PARM144 ; ELSE, IT IS 1.44MB DRIVE
5796		<1>	
5797		<1>	;-----
5798		<1>	; DISK_TYPE (AH = 15H)
5799		<1>	; THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
5800		<1>	
5801		<1>	; ON ENTRY: DI = DRIVE #
5802		<1>	
5803		<1>	; ON EXIT: AH = DRIVE TYPE, CY=0
5804		<1>	;-----
5805		<1>	DSK_TYPE:
5806	00001C71 E8E9010000	<1>	CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
5807	00001C76 8A87[39710000]	<1>	MOV AL, [DSK_STATE+eDI] ; GET PRESENT STATE INFORMATION
5808	00001C7C 08C0	<1>	OR AL,AL ; CHECK FOR NO DRIVE
5809	00001C7E 7418	<1>	JZ short NO_DRV
5810	00001C80 B401	<1>	MOV AH,NOCHGLN ; NO CHANGE LINE FOR 40 TRACK DRIVE
5811	00001C82 A801	<1>	TEST AL,TRK_CAPA ; IS THIS DRIVE AN 80 TRACK DRIVE?
5812	00001C84 7402	<1>	JZ short DT_BACK ; IF NO JUMP
5813	00001C86 B402	<1>	MOV AH,CHGLN ; CHANGE LINE FOR 80 TRACK DRIVE
5814		<1>	DT_BACK:
5815	00001C88 6650	<1>	PUSH AX ; SAVE RETURN VALUE
5816	00001C8A E801020000	<1>	CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
5817	00001C8F 6658	<1>	POP AX ; RESTORE RETURN VALUE
5818	00001C91 F8	<1>	CLC ; NO ERROR
5819	00001C92 6689F3	<1>	MOV BX,SI ; GET SAVED AL TO BL
5820	00001C95 88D8	<1>	MOV AL,BL ; PUT BACK FOR RETURN
5821	00001C97 C3	<1>	RETn
5822		<1>	NO_DRV:
5823	00001C98 30E4	<1>	XOR AH,AH ; NO DRIVE PRESENT OR UNKNOWN
5824	00001C9A EBEC	<1>	JMP SHORT DT_BACK
5825		<1>	
5826		<1>	;-----
5827		<1>	; DISK_CHANGE (AH = 16H)
5828		<1>	; THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.
5829		<1>	
5830		<1>	; ON ENTRY: DI = DRIVE #
5831		<1>	
5832		<1>	; ON EXIT: AH = @DSKETTE_STATUS
5833		<1>	; 00 - DISK CHANGE LINE INACTIVE, CY = 0
5834		<1>	; 06 - DISK CHANGE LINE ACTIVE, CY = 1
5835		<1>	;-----
5836		<1>	DSK_CHANGE:
5837	00001C9C E8BE010000	<1>	CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
5838	00001CA1 8A87[39710000]	<1>	MOV AL, [DSK_STATE+eDI] ; GET MEDIA STATE INFORMATION
5839	00001CA7 08C0	<1>	OR AL,AL ; DRIVE PRESENT ?
5840	00001CA9 7422	<1>	JZ short DC_NON ; JUMP IF NO DRIVE
5841	00001CAB A801	<1>	TEST AL,TRK_CAPA ; 80 TRACK DRIVE ?
5842	00001CAD 7407	<1>	JZ short SETIT ; IF SO , CHECK CHANGE LINE
5843		<1>	DC0:
5844	00001CAF E88C0A0000	<1>	CALL READ_DSKCHNG ; GO CHECK STATE OF DISK CHANGE LINE
5845	00001CB4 7407	<1>	JZ short FINIS ; CHANGE LINE NOT ACTIVE
5846		<1>	
5847	00001CB6 C605[2C710000]06	<1>	SETIT: MOV byte [DSKETTE_STATUS], MEDIA_CHANGE ; INDICATE MEDIA REMOVED
5848		<1>	
5849	00001CBD E8CE010000	<1>	FINIS: CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
5850	00001CC2 E807070000	<1>	CALL SETUP_END ; VARIOUS CLEANUPS
5851	00001CC7 6689F3	<1>	MOV BX,SI ; GET SAVED AL TO BL
5852	00001CCA 88D8	<1>	MOV AL,BL ; PUT BACK FOR RETURN
5853	00001CCC C3	<1>	RETn
5854		<1>	DC_NON:
5855	00001CCD 800D[2C710000]80	<1>	OR byte [DSKETTE_STATUS], TIME_OUT ; SET TIMEOUT, NO DRIVE
5856	00001CD4 EBE7	<1>	JMP SHORT FINIS
5857		<1>	
5858		<1>	;-----
5859		<1>	; FORMAT_SET (AH = 17H)
5860		<1>	; THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED
5861		<1>	; FOR THE FOLLOWING FORMAT OPERATION.
5862		<1>	
5863		<1>	; ON ENTRY: SI LOW = DASD TYPE FOR FORMAT
5864		<1>	; DI = DRIVE #
5865		<1>	
5866		<1>	; ON EXIT: @DSKETTE_STATUS REFLECTS STATUS
5867		<1>	; AH = @DSKETTE_STATUS
5868		<1>	; CY = 1 IF ERROR
5869		<1>	;-----
5870		<1>	FORMAT_SET:



5871	00001CD6	E884010000	<1>	CALL	XLAT_NEW	; TRANSLATE STATE TO PRESENT ARCH.
5872	00001CDB	6656	<1>	PUSH	SI	; SAVE DASD TYPE
5873	00001CDD	6689F0	<1>	MOV	AX,SI	; AH = ? , AL , DASD TYPE
5874	00001CE0	30E4	<1>	XOR	AH,AH	; AH , 0 , AL , DASD TYPE
5875	00001CE2	6689C6	<1>	MOV	SI,AX	; SI = DASD TYPE
5876	00001CE5	80A7[39710000]0F	<1>	AND	byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK)	; CLEAR STATE
5877	00001CEC	664E	<1>	DEC	SI	; CHECK FOR 320/360K MEDIA & DRIVE
5878	00001CEE	7509	<1>	JNZ	short NOT_320	; BYPASS IF NOT
5879	00001CF0	808F[39710000]90	<1>	OR	byte [DSK_STATE+eDI], MED_DET+RATE_250	; SET TO 320/360
5880	00001CF7	EB48	<1>	JMP	SHORT S0	
5881			<1>			
5882			<1>	NOT_320:		
5883	00001CF9	E8B6030000	<1>	CALL	MED_CHANGE	; CHECK FOR TIME_OUT
5884	00001CFE	803D[2C710000]80	<1>	CMP	byte [DSKETTE_STATUS], TIME_OUT	
5885	00001D05	743A	<1>	JZ	short S0	; IF TIME OUT TELL CALLER
5886			<1>	S3:		
5887	00001D07	664E	<1>	DEC	SI	; CHECK FOR 320/360K IN 1.2M DRIVE
5888	00001D09	7509	<1>	JNZ	short NOT_320_12	; BYPASS IF NOT
5889	00001D0B	808F[39710000]70	<1>	OR	byte [DSK_STATE+eDI], MED_DET+DBL_STEP+RATE_300	; SET STATE
5890	00001D12	EB2D	<1>	JMP	SHORT S0	
5891			<1>			
5892			<1>	NOT_320_12:		
5893	00001D14	664E	<1>	DEC	SI	; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
5894	00001D16	7509	<1>	JNZ	short NOT_12	; BYPASS IF NOT
5895	00001D18	808F[39710000]10	<1>	OR	byte [DSK_STATE+eDI], MED_DET+RATE_500	; SET STATE VARIABLE
5896	00001D1F	EB20	<1>	JMP	SHORT S0	; RETURN TO CALLER
5897			<1>			
5898			<1>	NOT_12:		
5899	00001D21	664E	<1>	DEC	SI	; CHECK FOR SET DASD TYPE 04
5900	00001D23	752B	<1>	JNZ	short FS_ERR	; BAD COMMAND EXIT IF NOT VALID TYPE
5901			<1>			
5902	00001D25	F687[39710000]04	<1>	TEST	byte [DSK_STATE+eDI], DRV_DET	; DRIVE DETERMINED ?
5903	00001D2C	740B	<1>	JZ	short ASSUME	; IF STILL NOT DETERMINED ASSUME
5904	00001D2E	B050	<1>	MOV	AL,MED_DET+RATE_300	
5905	00001D30	F687[39710000]02	<1>	TEST	byte [DSK_STATE+eDI], FMT_CAPA	; MULTIPLE FORMAT CAPABILITY ?
5906	00001D37	7502	<1>	JNZ	short OR_IT_IN	; IF 1.2 M THEN DATA RATE 300
5907			<1>			
5908			<1>	ASSUME:		
5909	00001D39	B090	<1>	MOV	AL,MED_DET+RATE_250	; SET UP
5910			<1>			
5911			<1>	OR_IT_IN:		
5912	00001D3B	0887[39710000]	<1>	OR	[DSK_STATE+eDI], AL	; OR IN THE CORRECT STATE
5913			<1>	S0:		
5914	00001D41	E84A010000	<1>	CALL	XLAT_OLD	; TRANSLATE STATE TO COMPATIBLE MODE
5915	00001D46	E883060000	<1>	CALL	SETUP_END	; VARIOUS CLEANUPS
5916	00001D4B	665B	<1>	POP	BX	; GET SAVED AL TO BL
5917	00001D4D	88D8	<1>	MOV	AL,BL	; PUT BACK FOR RETURN
5918	00001D4F	C3	<1>	RETn		
5919			<1>			
5920			<1>	FS_ERR:		
5921	00001D50	C605[2C710000]01	<1>	MOV	byte [DSKETTE_STATUS], BAD_CMD	; UNKNOWN STATE,BAD COMMAND
5922	00001D57	EBE8	<1>	JMP	SHORT S0	
5923			<1>			
5924			<1>			
5925			<1>			
5926			<1>			
5927			<1>			
5928			<1>			
5929			<1>			
5930			<1>			
5931			<1>			
5932			<1>			
5933			<1>			
5934			<1>			
5935			<1>			
5936			<1>			
5937			<1>			
5938			<1>			
5939			<1>			
5940			<1>			
5941			<1>			
5942			<1>			
5943			<1>			
5944			<1>			
5945			<1>			
5946	00001D59	E801010000	<1>	CALL	XLAT_NEW	; TRANSLATE STATE TO PRESENT ARCH.
5947	00001D5E	F687[39710000]01	<1>	TEST	byte [DSK_STATE+eDI], TRK_CAPA	; CHECK FOR CHANGE LINE AVAILABLE
5948	00001D65	7415	<1>	JZ	short SM_CMOS	; JUMP IF 40 TRACK DRIVE
5949	00001D67	E848030000	<1>	CALL	MED_CHANGE	; RESET CHANGE LINE
5950	00001D6C	803D[2C710000]80	<1>	CMP	byte [DSKETTE_STATUS], TIME_OUT	; IF TIME OUT TELL CALLER
5951	00001D73	746B	<1>	JE	short SM_RTN	
5952	00001D75	C605[2C710000]00	<1>	MOV	byte [DSKETTE_STATUS], 0	; CLEAR STATUS
5953			<1>	SM_CMOS:		
5954	00001D7C	E818070000	<1>	CALL	CMOS_TYPE	; RETURN DRIVE TYPE IN (AL)
5955			<1>			
5956			<1>			
5957			<1>			
5958	00001D81	745D	<1>	JZ	short SM_RTN	; RETURN IF SO
5959	00001D83	E863000000	<1>	CALL	DR_TYPE_CHECK	; RTN CS:BX = MEDIA/DRIVE PARAM TBL
5960	00001D88	7231	<1>	JC	short MD_NOT_FND	; TYPE NOT IN TABLE (BAD CMOS)
5961	00001D8A	57	<1>	PUSH	eDI	; SAVE REG.
5962	00001D8B	31DB	<1>	XOR	eBX,eBX	; BX = INDEX TO DR. TYPE TABLE

```

5963 00001D8D B906000000 <1> MOV eCX,DR_CNT ; CX = LOOP COUNT
5964 DR_SEARCH: <1> DR_SEARCH:
5965 00001D92 8AA3[DC6A0000] <1> MOV AH, [DR_TYPE+eBX] ; GET DRIVE TYPE
5966 00001D98 80E47F <1> AND AH,BIT7OFF ; MASK OUT MSB
5967 00001D9B 38E0 <1> CMP AL,AH ; DRIVE TYPE MATCH ?
5968 00001D9D 7516 <1> JNE short NXT_MD ; NO, CHECK NEXT DRIVE TYPE
5969 <1> DR_FND:
5970 00001D9F 8BBB[DD6A0000] <1> MOV eDI, [DR_TYPE+eBX+1] ; DI = MEDIA/DRIVE PARAM TABLE
5971 <1> MD_SEARCH:
5972 00001DA5 8A6704 <1> MOV AH, [eDI+MD.SEC_TRK] ; GET SECTOR/TRACK
5973 00001DA8 386500 <1> CMP [eBP],AH ; MATCH?
5974 00001DAB 7508 <1> JNE short NXT_MD ; NO, CHECK NEXT MEDIA
5975 00001DAD 8A670B <1> MOV AH, [eDI+MD.MAX_TRK] ; GET MAX. TRACK #
5976 00001DB0 386501 <1> CMP [eBP+1],AH ; MATCH?
5977 00001DB3 740F <1> JE short MD_FND ; YES, GO GET RATE
5978 <1> NXT_MD:
5979 <1> ;ADD BX,3 ; CHECK NEXT DRIVE TYPE
5980 00001DB5 83C305 <1> add ebx, 5 ; 18/02/2015
5981 00001DB8 E2D8 <1> LOOP DR_SEARCH
5982 00001DBA 5F <1> POP eDI ; RESTORE REG.
5983 <1> MD_NOT_FND:
5984 00001DBB C605[2C710000]0C <1> MOV byte [DSKETTE_STATUS], MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
5985 00001DC2 EB1C <1> JMP SHORT SM_RTN ; RETURN
5986 <1> MD_FND:
5987 00001DC4 8A470C <1> MOV AL, [eDI+MD.RATE] ; GET RATE
5988 00001DC7 3C40 <1> CMP AL,RATE_300 ; DOUBLE STEP REQUIRED FOR RATE 300
5989 00001DC9 7502 <1> JNE short MD_SET
5990 00001DCB 0C20 <1> OR AL,DBL_STEP
5991 <1> MD_SET:
5992 <1> ;MOV [BP+6],DI ; SAVE TABLE POINTER IN STACK
5993 00001DCD 897D0C <1> mov [ebp+12], edi ; 18/02/2015
5994 00001DD0 0C10 <1> OR AL,MED_DET ; SET MEDIA ESTABLISHED
5995 00001DD2 5F <1> POP eDI
5996 00001DD3 80A7[39710000]0F <1> AND byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
5997 00001DDA 0887[39710000] <1> OR [DSK_STATE+eDI], AL
5998 <1> ;MOV AX, CS ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
5999 <1> ;MOV ES, AX ; ES IS SEGMENT OF TABLE
6000 <1> SM_RTN:
6001 00001DE0 E8AB000000 <1> CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
6002 00001DE5 E8E4050000 <1> CALL SETUP_END ; VARIOUS CLEANUPS
6003 00001DEA C3 <1> RETn
6004 <1>
6005 <1> ;-----
6006 <1> ; DR_TYPE_CHECK :
6007 <1> ; CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL) :
6008 <1> ; IS SUPPORTED IN BIOS DRIVE TYPE TABLE :
6009 <1> ; ON ENTRY: :
6010 <1> ; AL = DRIVE TYPE :
6011 <1> ; ON EXIT: :
6012 <1> ; CS = SEGMENT MEDIA/DRIVE PARAMETER TABLE (CODE) :
6013 <1> ; CY = 0 DRIVE TYPE SUPPORTED :
6014 <1> ; BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE :
6015 <1> ; CY = 1 DRIVE TYPE NOT SUPPORTED :
6016 <1> ; REGISTERS ALTERED: eBX :
6017 <1> ;-----
6018 <1> DR_TYPE_CHECK:
6019 00001DEB 6650 <1> PUSH AX
6020 00001DED 51 <1> PUSH eCX
6021 00001DEE 31DB <1> XOR eBX,eBX ; BX = INDEX TO DR_TYPE TABLE
6022 00001DF0 B906000000 <1> MOV eCX,DR_CNT ; CX = LOOP COUNT
6023 <1> TYPE_CHK:
6024 00001DF5 8AA3[DC6A0000] <1> MOV AH,[DR_TYPE+eBX] ; GET DRIVE TYPE
6025 00001DFB 38E0 <1> CMP AL,AH ; DRIVE TYPE MATCH?
6026 00001DFD 740D <1> JE short DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
6027 <1> ;ADD BX,3 ; CHECK NEXT DRIVE TYPE
6028 00001DFF 83C305 <1> add ebx, 5 ; 16/02/2015 (32 bit address modification)
6029 00001E02 E2F1 <1> LOOP TYPE_CHK
6030 <1> ;
6031 00001E04 BB[3B6B0000] <1> mov ebx, MD_TBL6 ; 1.44MB fd parameter table
6032 <1> ; Default for GET_PARM (11/12/2014)
6033 <1> ;
6034 00001E09 F9 <1> STC ; DRIVE TYPE NOT FOUND IN TABLE
6035 00001E0A EB06 <1> JMP SHORT TYPE_RTN
6036 <1> DR_TYPE_VALID:
6037 00001E0C 8B9B[DD6A0000] <1> MOV eBX,[DR_TYPE+eBX+1] ; BX = MEDIA TABLE
6038 <1> TYPE_RTN:
6039 00001E12 59 <1> POP eCX
6040 00001E13 6658 <1> POP AX
6041 00001E15 C3 <1> RETn
6042 <1>
6043 <1> ;-----
6044 <1> ; SEND_SPEC :
6045 <1> ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM :
6046 <1> ; THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER :
6047 <1> ; ON ENTRY: @DISK_POINTER = DRIVE PARAMETER TABLE :
6048 <1> ; ON EXIT: NONE :
6049 <1> ; REGISTERS ALTERED: CX, DX :
6050 <1> ;-----
6051 <1> SEND_SPEC:
6052 00001E16 50 <1> PUSH eAX ; SAVE AX
6053 00001E17 B8[3D1E0000] <1> MOV eAX, SPECBAC ; LOAD ERROR ADDRESS
6054 00001E1C 50 <1> PUSH eAX ; PUSH NEC_OUT ERROR RETURN

```

```

6055 00001E1D B403          <1>  MOV    AH,03H                ; SPECIFY COMMAND
6056 00001E1F E884070000    <1>  CALL   NEC_OUTPUT            ; OUTPUT THE COMMAND
6057 00001E24 28D2          <1>  SUB     DL,DL                ; FIRST SPECIFY BYTE
6058 00001E26 E877060000    <1>  CALL   GET_PARM              ; GET PARAMETER TO AH
6059 00001E2B E878070000    <1>  CALL   NEC_OUTPUT            ; OUTPUT THE COMMAND
6060 00001E30 B201          <1>  MOV     DL,1                ; SECOND SPECIFY BYTE
6061 00001E32 E86B060000    <1>  CALL   GET_PARM              ; GET PARAMETER TO AH
6062 00001E37 E86C070000    <1>  CALL   NEC_OUTPUT            ; OUTPUT THE COMMAND
6063 00001E3C 58            <1>  POP     eAX                  ; POP ERROR RETURN
6064                                <1>  SPECBAC:
6065 00001E3D 58            <1>  POP     eAX                  ; RESTORE ORIGINAL AX VALUE
6066 00001E3E C3            <1>  RETn
6067                                <1>
6068                                <1> ;-----
6069                                <1> ; SEND_SPEC_MD                :
6070                                <1> ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM      :
6071                                <1> ; THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)        :
6072                                <1> ; ON ENTRY:  CS:BX = MEDIA/DRIVE PARAMETER TABLE            :
6073                                <1> ; ON EXIT:   NONE                                           :
6074                                <1> ; REGISTERS ALTERED: AX                                     :
6075                                <1> ;-----
6076                                <1> SEND_SPEC_MD:
6077 00001E3F 50            <1>  PUSH    eAX                  ; SAVE RATE DATA
6078 00001E40 B8[5D1E0000]  <1>  MOV     eAX, SPEC_ESBAC      ; LOAD ERROR ADDRESS
6079 00001E45 50            <1>  PUSH    eAX                  ; PUSH NEC_OUT ERROR RETURN
6080 00001E46 B403          <1>  MOV     AH,03H                ; SPECIFY COMMAND
6081 00001E48 E85B070000    <1>  CALL   NEC_OUTPUT            ; OUTPUT THE COMMAND
6082 00001E4D 8A23          <1>  MOV     AH, [eBX+MD.SPEC1]    ; GET 1ST SPECIFY BYTE
6083 00001E4F E854070000    <1>  CALL   NEC_OUTPUT            ; OUTPUT THE COMMAND
6084 00001E54 8A6301        <1>  MOV     AH, [eBX+MD.SPEC2]    ; GET SECOND SPECIFY BYTE
6085 00001E57 E84C070000    <1>  CALL   NEC_OUTPUT            ; OUTPUT THE COMMAND
6086 00001E5C 58            <1>  POP     eAX                  ; POP ERROR RETURN
6087                                <1> SPEC_ESBAC:
6088 00001E5D 58            <1>  POP     eAX                  ; RESTORE ORIGINAL AX VALUE
6089 00001E5E C3            <1>  RETn
6090                                <1>
6091                                <1> ;-----
6092                                <1> ; XLAT_NEW
6093                                <1> ; TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
6094                                <1> ; MODE TO NEW ARCHITECTURE.
6095                                <1> ;
6096                                <1> ; ON ENTRY:  DI = DRIVE #
6097                                <1> ;-----
6098                                <1> XLAT_NEW:
6099 00001E5F 83FF01        <1>  CMP     eDI,1                ; VALID DRIVE
6100 00001E62 7725          <1>  JA      short XN_OUT          ; IF INVALID BACK
6101 00001E64 80BF[39710000]00 <1>  CMP     byte [DSK_STATE+eDI], 0 ; NO DRIVE ?
6102 00001E6B 741D          <1>  JZ      short DO_DET          ; IF NO DRIVE ATTEMPT DETERMINE
6103 00001E6D 6689F9        <1>  MOV     CX,DI                ; CX = DRIVE NUMBER
6104 00001E70 C0E102        <1>  SHL     CL,2                ; CL = SHIFT COUNT, A=0, B=4
6105 00001E73 A0[38710000]  <1>  MOV     AL, [HF_CNTRL]        ; DRIVE INFORMATION
6106 00001E78 D2C8          <1>  ROR     AL,CL                ; TO LOW NIBBLE
6107 00001E7A 2407          <1>  AND     AL,DRV_DET+_FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
6108 00001E7C 80A7[39710000]F8 <1>  AND     byte [DSK_STATE+eDI], ~(DRV_DET+_FMT_CAPA+TRK_CAPA)
6109 00001E83 0887[39710000] <1>  OR      [DSK_STATE+eDI], AL    ; UPDATE DRIVE STATE
6110                                <1> XN_OUT:
6111 00001E89 C3            <1>  RETn
6112                                <1> DO_DET:
6113 00001E8A E8BE080000    <1>  CALL   DRIVE_DET            ; TRY TO DETERMINE
6114 00001E8F C3            <1>  RETn
6115                                <1>
6116                                <1> ;-----
6117                                <1> ; XLAT_OLD
6118                                <1> ; TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
6119                                <1> ; ARCHITECTURE TO COMPATIBLE MODE.
6120                                <1> ;
6121                                <1> ; ON ENTRY:  DI = DRIVE
6122                                <1> ;-----
6123                                <1> XLAT_OLD:
6124 00001E90 83FF01        <1>  CMP     eDI,1                ; VALID DRIVE ?
6125                                <1>  ;JA      short XO_OUT          ; IF INVALID BACK
6126 00001E93 0F8786000000    <1>  ja      XO_OUT
6127 00001E99 80BF[39710000]00 <1>  CMP     byte [DSK_STATE+eDI],0 ; NO DRIVE ?
6128 00001EA0 747D          <1>  JZ      short XO_OUT          ; IF NO DRIVE TRANSLATE DONE
6129                                <1>
6130                                <1> ;----- TEST FOR SAVED DRIVE INFORMATION ALREADY SET
6131                                <1>
6132 00001EA2 6689F9        <1>  MOV     CX,DI                ; CX = DRIVE NUMBER
6133 00001EA5 C0E102        <1>  SHL     CL,2                ; CL = SHIFT COUNT, A=0, B=4
6134 00001EA8 B402          <1>  MOV     AH,FMT_CAPA          ; LOAD MULTIPLE DATA RATE BIT MASK
6135 00001EAA D2CC          <1>  ROR     AH,CL                ; ROTATE BY MASK
6136 00001EAC 8425[38710000] <1>  TEST    [HF_CNTRL], AH        ; MULTIPLE-DATA RATE DETERMINED ?
6137 00001EB2 751C          <1>  JNZ     short SAVE_SET        ; IF SO, NO NEED TO RE-SAVE
6138                                <1>
6139                                <1> ;----- ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE
6140                                <1>
6141 00001EB4 B407          <1>  MOV     AH,DRV_DET+_FMT_CAPA+TRK_CAPA ; MASK TO KEEP
6142 00001EB6 D2CC          <1>  ROR     AH,CL                ; FIX MASK TO KEEP
6143 00001EB8 F6D4          <1>  NOT     AH                    ; TRANSLATE MASK
6144 00001EBA 2025[38710000] <1>  AND     [HF_CNTRL], AH        ; KEEP BITS FROM OTHER DRIVE INTACT
6145                                <1>
6146                                <1> ;----- ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL

```

```

6147
6148 00001EC0 8A87[39710000]
6149 00001EC6 2407
6150 00001EC8 D2C8
6151 00001ECA 0805[38710000]
6152
6153
6154
6155
6156 00001ED0 8AA7[39710000]
6157 00001ED6 88E7
6158 00001ED8 80E4C0
6159 00001EDB 80FC00
6160 00001EDE 7410
6161 00001EE0 B001
6162 00001EE2 80FC40
6163 00001EE5 7518
6164 00001EE7 F6C720
6165 00001EEA 751F
6166
6167 00001EEC B007
6168 00001EEE EB22
6169
6170 00001EF0 E8A4050000
6171
6172
6173 00001EF5 74F5
6174 00001EF7 3C02
6175 00001EF9 75F1
6176 00001EFB B002
6177 00001EFD EB0C
6178
6179 00001EFF B000
6180 00001F01 80FC80
6181 00001F04 75E6
6182 00001F06 F6C701
6183 00001F09 75E1
6184
6185 00001F0B F6C710
6186 00001F0E 7402
6187 00001F10 0403
6188
6189 00001F12 80A7[39710000]F8
6190 00001F19 0887[39710000]
6191
6192 00001F1F C3
6193
6194
6195
6196
6197
6198
6199
6200
6201
6202
6203
6204
6205 00001F20 6650
6206 00001F22 E838FFFFFF
6207 00001F27 E8F3000000
6208 00001F2C 6658
6209
6210 00001F2E 6650
6211 00001F30 E87F010000
6212 00001F35 6658
6213 00001F37 0F82C9000000
6214
6215 00001F3D 6650
6216 00001F3F 8AB7[39710000]
6217 00001F45 80E6C0
6218 00001F48 E84C050000
6219
6220
6221 00001F4D 7451
6222 00001F4F 3C01
6223 00001F51 750D
6224 00001F53 F687[39710000]01
6225 00001F5A 7413
6226 00001F5C B002
6227 00001F5E EB0F
6228
6229 00001F60 720D
6230 00001F62 F687[39710000]01
6231 00001F69 7504
6232 00001F6B B001
6233 00001F6D EB04
6234
6235 00001F6F 08C0
6236 00001F71 742D
6237
6238 00001F73 E873FEFFFF

```

```

<1>
<1> MOV AL, [DSK_STATE+eDI] ; ACCESS STATE
<1> AND AL,DRV_DET+_FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
<1> ROR AL,CL ; FIX FOR THIS DRIVE
<1> OR [HF_CNTRL], AL ; UPDATE SAVED DRIVE STATE
<1>
<1> ;----- TRANSLATE TO COMPATIBILITY MODE
<1>
<1> SAVE_SET:
<1> MOV AH, [DSK_STATE+eDI] ; ACCESS STATE
<1> MOV BH,AH ; TO BH FOR LATER
<1> AND AH,RATE_MSK ; KEEP ONLY RATE
<1> CMP AH,RATE_500 ; RATE 500 ?
<1> JZ short CHK_144 ; YES 1.2/1.2 OR 1.44/1.44
<1> MOV AL,M3D1U ; AL = 360 IN 1.2 UNESTABLISHED
<1> CMP AH,RATE_300 ; RATE 300 ?
<1> JNZ short CHK_250 ; NO, 360/360, 720/720 OR 720/1.44
<1> TEST BH,DBL_STEP ; CHECK FOR DOUBLE STEP
<1> JNZ short TST_DET ; MUST BE 360 IN 1.2
<1> UNKNO:
<1> MOV AL,MED_UNK ; NONE OF THE ABOVE
<1> JMP SHORT AL_SET ; PROCESS COMPLETE
<1> CHK_144:
<1> CALL CMOS_TYPE ; RETURN DRIVE TYPE IN (AL)
<1> ;;20/02/2015
<1> ;;JC short UNKNO ; ERROR, SET 'NONE OF ABOVE'
<1> jz short UNKNO ;; 20/02/2015
<1> CMP AL,2 ; 1.2MB DRIVE ?
<1> JNE short UNKNO ; NO, GO SET 'NONE OF ABOVE'
<1> MOV AL,M1D1U ; AL = 1.2 IN 1.2 UNESTABLISHED
<1> JMP SHORT TST_DET
<1> CHK_250:
<1> MOV AL,M3D3U ; AL = 360 IN 360 UNESTABLISHED
<1> CMP AH,RATE_250 ; RATE 250 ?
<1> JNZ short UNKNO ; IF SO FALL IHRU
<1> TEST BH,TRK_CAPA ; 80 TRACK CAPABILITY ?
<1> JNZ short UNKNO ; IF SO JUMP, FALL THRU TEST DET
<1> TST_DET:
<1> TEST BH,MED_DET ; DETERMINED ?
<1> JZ short AL_SET ; IF NOT THEN SET
<1> ADD AL,3 ; MAKE DETERMINED/ESTABLISHED
<1> AL_SET:
<1> AND byte [DSK_STATE+eDI], ~(DRV_DET+_FMT_CAPA+TRK_CAPA) ; CLEAR DRIVE
<1> OR [DSK_STATE+eDI], AL ; REPLACE WITH COMPATIBLE MODE
<1> XO_OUT:
<1> RETn
<1>
<1> ;-----
<1> ; RD_WR_VF
<1> ; COMMON READ, WRITE AND VERIFY:
<1> ; MAIN LOOP FOR STATE RETRIES.
<1> ;
<1> ; ON ENTRY: AH = READ/WRITE/VERIFY NEC PARAMETER
<1> ; AL = READ/WRITE/VERIFY DMA PARAMETER
<1> ;
<1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
<1> ;-----
<1> RD_WR_VF:
<1> PUSH AX ; SAVE DMA, NEC PARAMETERS
<1> CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH.
<1> CALL SETUP_STATE ; INITIALIZE START AND END RATE
<1> POP AX ; RESTORE READ/WRITE/VERIFY
<1> DO_AGAIN:
<1> PUSH AX ; SAVE READ/WRITE/VERIFY PARAMETER
<1> CALL MED_CHANGE ; MEDIA CHANGE AND RESET IF CHANGED
<1> POP AX ; RESTORE READ/WRITE/VERIFY
<1> JC RWV_END ; MEDIA CHANGE ERROR OR TIME-OUT
<1> RWV:
<1> PUSH AX ; SAVE READ/WRITE/VERIFY PARAMETER
<1> MOV DH, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
<1> AND DH,RATE_MSK ; KEEP ONLY RATE
<1> CALL CMOS_TYPE ; RETURN DRIVE TYPE IN AL (AL)
<1> ;;20/02/2015
<1> ;;JC short RWV_ASSUME ; ERROR IN CMOS
<1> jz short RWV_ASSUME ; 20/02/2015
<1> CMP AL,1 ; 40 TRACK DRIVE?
<1> JNE short RWV_1 ; NO, BYPASS CMOS VALIDITY CHECK
<1> TEST byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
<1> JZ short RWV_2 ; YES, CMOS IS CORRECT
<1> MOV AL,2 ; CHANGE TO 1.2M
<1> JMP SHORT RWV_2
<1> RWV_1:
<1> JB short RWV_2 ; NO DRIVE SPECIFIED, CONTINUE
<1> TEST byte [DSK_STATE+eDI], TRK_CAPA ; IS IT REALLY 40 TRACK?
<1> JNZ short RWV_2 ; NO, 80 TRACK
<1> MOV AL,1 ; IT IS 40 TRACK, FIX CMOS VALUE
<1> jmp short rwv_3
<1> RWV_2:
<1> OR AL,AL ; TEST FOR NO DRIVE
<1> JZ short RWV_ASSUME ; ASSUME TYPE, USE MAX TRACK
<1> rwv_3:
<1> CALL DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL.

```

```

6239 00001F78 7226      <1> JC      short RWV_ASSUME ; TYPE NOT IN TABLE (BAD CMOS)
6240                      <1>
6241                      <1> ;----- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
6242                      <1>
6243 00001F7A 57          <1> PUSH  eDI                ; SAVE DRIVE #
6244 00001F7B 31DB        <1> XOR    eBX,eBX                ; BX = INDEX TO DR_TYPE TABLE
6245 00001F7D B906000000  <1> MOV    eCX,DR_CNT            ; CX = LOOP COUNT
6246                      <1> RWV_DR_SEARCH:
6247 00001F82 8AA3[DC6A0000] <1> MOV    AH, [DR_TYPE+eBX] ; GET DRIVE TYPE
6248 00001F88 80E47F      <1> AND    AH,BIT7OFF          ; MASK OUT MSB
6249 00001F8B 38E0        <1> CMP    AL,AH                ; DRIVE TYPE MATCH?
6250 00001F8D 750B        <1> JNE     short RWV_NXT_MD    ; NO, CHECK NEXT DRIVE TYPE
6251                      <1> RWV_DR_FND:
6252 00001F8F 8BBB[DD6A0000] <1> MOV    eDI, [DR_TYPE+eBX+1] ; DI = MEDIA/DRIVE PARAMETER TABLE
6253                      <1> RWV_MD_SEARH:
6254 00001F95 3A770C      <1> CMP     DH, [eDI+MD.RATE]    ; MATCH?
6255 00001F98 741B        <1> JE      short RWV_MD_FND ; YES, GO GET 1ST SPECIFY BYTE
6256                      <1> RWV_NXT_MD:
6257                      <1> ;ADD  BX,3                ; CHECK NEXT DRIVE TYPE
6258 00001F9A 83C305      <1> add     eBX, 5
6259 00001F9D E2E3        <1> LOOP   RWV_DR_SEARCH
6260 00001F9F 5F          <1> POP     eDI                ; RESTORE DRIVE #
6261                      <1>
6262                      <1> ;----- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
6263                      <1>
6264                      <1> RWV_ASSUME:
6265 00001FA0 BB[FA6A0000]  <1> MOV     eBX, MD_TBL1          ; POINT TO 40 TRACK 250 KBS
6266 00001FA5 F687[39710000]01 <1> TEST   byte [DSK_STATE+eDI], TRK_CAPA ; TEST FOR 80 TRACK
6267 00001FAC 740A        <1> JZ      short RWV_MD_FND1 ; MUST BE 40 TRACK
6268 00001FAE BB[146B0000] <1> MOV     eBX, MD_TBL3          ; POINT TO 80 TRACK 500 KBS
6269 00001FB3 EB03        <1> JMP     short RWV_MD_FND1 ; GO SPECIFY PARAMETERS
6270                      <1>
6271                      <1> ;----- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
6272                      <1>
6273                      <1> RWV_MD_FND:
6274 00001FB5 89FB        <1> MOV     eBX,eDI                ; BX = MEDIA/DRIVE PARAMETER TABLE
6275 00001FB7 5F          <1> POP     eDI                ; RESTORE DRIVE #
6276                      <1>
6277                      <1> ;----- SEND THE SPECIFY COMMAND TO THE CONTROLLER
6278                      <1>
6279                      <1> RWV_MD_FND1:
6280 00001FB8 E882FEFFFF  <1> CALL    SEND_SPEC_MD
6281 00001FBD E864010000  <1> CALL    CHK_LASRATE          ; ZF=1 ATTEMP RATE IS SAME AS LAST RATE
6282 00001FC2 7405        <1> JZ      short RWV_DBL        ; YES,SKIP SEND RATE COMMAND
6283 00001FC4 E83B010000  <1> CALL    SEND_RATE           ; SEND DATA RATE TO NEC
6284                      <1> RWV_DBL:
6285 00001FC9 53          <1> PUSH    eBX                ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
6286 00001FCA E821040000  <1> CALL    SETUP_DBL           ; CHECK FOR DOUBLE STEP
6287 00001FCF 5B          <1> POP     eBX                ; RESTORE ADDRESS
6288 00001FD0 7226        <1> JC      short CHK_RET        ; ERROR FROM READ ID, POSSIBLE RETRY
6289 00001FD2 6658        <1> POP     AX                 ; RESTORE NEC, DMA COMMAND
6290 00001FD4 6650        <1> PUSH    AX                 ; SAVE NEC COMMAND
6291 00001FD6 53          <1> PUSH    eBX                ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
6292 00001FD7 E861010000  <1> CALL    DMA_SETUP           ; SET UP THE DMA
6293 00001FDC 5B          <1> POP     eBX
6294 00001FDD 6658        <1> POP     AX                 ; RESTORE NEC COMMAND
6295 00001FDF 722F        <1> JC      short RWV_BAC        ; CHECK FOR DMA BOUNDARY ERROR
6296 00001FE1 6650        <1> PUSH    AX                 ; SAVE NEC COMMAND
6297 00001FE3 53          <1> PUSH    eBX                ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
6298 00001FE4 E83B020000  <1> CALL    NEC_INIT           ; INITIALIZE NEC
6299 00001FE9 5B          <1> POP     eBX                ; RESTORE ADDRESS
6300 00001FEA 720C        <1> JC      short CHK_RET        ; ERROR - EXIT
6301 00001FEC E865020000  <1> CALL    RWV_COM            ; OP CODE COMMON TO READ/WRITE/VERIFY
6302 00001FF1 7205        <1> JC      short CHK_RET        ; ERROR - EXIT
6303 00001FF3 E8AA020000  <1> CALL    NEC_TERM           ; TERMINATE, GET STATUS, ETC.
6304                      <1> CHK_RET:
6305 00001FF8 E849030000  <1> CALL    RETRY              ; CHECK FOR, SETUP RETRY
6306 00001FFD 6658        <1> POP     AX                 ; RESTORE READ/WRITE/VERIFY PARAMETER
6307 00001FFF 7305        <1> JNC     short RWV_END        ; CY = 0 NO RETRY
6308 00002001 E928FFFFFF  <1> JMP     DO_AGAIN            ; CY = 1 MEANS RETRY
6309                      <1> RWV_END:
6310 00002006 E8F3020000  <1> CALL    DSTATE             ; ESTABLISH STATE IF SUCCESSFUL
6311 0000200B E886030000  <1> CALL    NUM_TRANS          ; AL = NUMBER TRANSFERRED
6312                      <1> RWV_BAC:
6313 00002010 6650        <1> PUSH    AX                 ; SAVE NUMBER TRANSFERRED
6314 00002012 E879FEFFFF  <1> CALL    XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
6315 00002017 6658        <1> POP     AX                 ; RESTORE NUMBER TRANSFERRED
6316 00002019 E8B0030000  <1> CALL    SETUP_END          ; VARIOUS CLEANUPS
6317 0000201E C3          <1> RETN
6318                      <1>
6319                      <1> ;-----
6320                      <1> ; SETUP_STATE:      INITIALIZES START AND END RATES.
6321                      <1> ;-----
6322                      <1> SETUP_STATE:
6323 0000201F F687[39710000]10 <1> TEST   byte [DSK_STATE+eDI], MED_DET ; MEDIA DETERMINED ?
6324 00002026 7537        <1> JNZ     short J1C            ; NO STATES IF DETERMINED
6325 00002028 66B84000      <1> MOV     AX, (RATE_500*256)+RATE_300 ; AH = START RATE, AL = END RATE
6326 0000202C F687[39710000]04 <1> TEST   byte [DSK_STATE+eDI], DRV_DET ; DRIVE ?
6327 00002033 740D        <1> JZ      short AX_SET         ; DO NOT KNOW DRIVE
6328 00002035 F687[39710000]02 <1> TEST   byte [DSK_STATE+eDI], FMT_CAPA ; MULTI-RATE?
6329 0000203C 7504        <1> JNZ     short AX_SET         ; JUMP IF YES
6330 0000203E 66B88080      <1> MOV     AX,RATE_250*257      ; START A END RATE 250 FOR 360 DRIVE

```

6331		<1> AX_SET:
6332	00002042 80A7[39710000]1F	<1> AND byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP) ; TURN OFF THE RATE
6333	00002049 08A7[39710000]	<1> OR [DSK_STATE+eDI], AH ; RATE FIRST TO TRY
6334	0000204F 8025[34710000]F3	<1> AND byte [LASTRATE], ~STRT_MSK ; ERASE LAST TO TRY RATE BITS
6335	00002056 C0C804	<1> ROR AL,4 ; TO OPERATION LAST RATE LOCATION
6336	00002059 0805[34710000]	<1> OR [LASTRATE], AL ; LAST RATE
6337		<1> J1C:
6338	0000205F C3	<1> RETn
6339		<1>
6340		<1> ;-----
6341		<1> ; FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
6342		<1> ;-----
6343		<1> FMT_INIT:
6344	00002060 F687[39710000]10	<1> TEST byte [DSK_STATE+eDI], MED_DET ; IS MEDIA ESTABLISHED
6345	00002067 7546	<1> JNZ short F1_OUT ; IF SO RETURN
6346	00002069 E82B040000	<1> CALL CMOS_TYPE ; RETURN DRIVE TYPE IN AL
6347		<1> ;; 20/02/2015
6348		<1> ;;JC short CL_DRV ; ERROR IN CMOS ASSUME NO DRIVE
6349	0000206E 7440	<1> jz short CL_DRV ;; 20/02/2015
6350	00002070 FEC8	<1> DEC AL ; MAKE ZERO ORIGIN
6351		<1> ;;JS short CL_DRV ; NO DRIVE IF AL 0
6352	00002072 8AA7[39710000]	<1> MOV AH, [DSK_STATE+eDI] ; AH = CURRENT STATE
6353	00002078 80E40F	<1> AND AH, ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR
6354	0000207B 08C0	<1> OR AL,AL ; CHECK FOR 360
6355	0000207D 7505	<1> JNZ short N_360 ; IF 360 WILL BE 0
6356	0000207F 80CC90	<1> OR AH,MED_DET+RATE_250 ; ESTABLISH MEDIA
6357	00002082 EB25	<1> JMP SHORT SKP_STATE ; SKIP OTHER STATE PROCESSING
6358		<1> N_360:
6359	00002084 FEC8	<1> DEC AL ; 1.2 M DRIVE
6360	00002086 7505	<1> JNZ short N_12 ; JUMP IF NOT
6361		<1> F1_RATE:
6362	00002088 80CC10	<1> OR AH,MED_DET+RATE_500 ; SET FORMAT RATE
6363	0000208B EB1C	<1> JMP SHORT SKP_STATE ; SKIP OTHER STATE PROCESSING
6364		<1> N_12:
6365	0000208D FEC8	<1> DEC AL ; CHECK FOR TYPE 3
6366	0000208F 750F	<1> JNZ short N_720 ; JUMP IF NOT
6367	00002091 F6C404	<1> TEST AH,DRV_DET ; IS DRIVE DETERMINED
6368	00002094 7410	<1> JZ short ISNT_12 ; TREAT AS NON 1.2 DRIVE
6369	00002096 F6C402	<1> TEST AH,FMT_CAPA ; IS 1.2M
6370	00002099 740B	<1> JZ short ISNT_12 ; JUMP IF NOT
6371	0000209B 80CC50	<1> OR AH,MED_DET+RATE_300 ; RATE 300
6372	0000209E EB09	<1> JMP SHORT SKP_STATE ; CONTINUE
6373		<1> N_720:
6374	000020A0 FEC8	<1> DEC AL ; CHECK FOR TYPE 4
6375	000020A2 750C	<1> JNZ short CL_DRV ; NO DRIVE, CMOS BAD
6376	000020A4 EBE2	<1> JMP SHORT F1_RATE
6377		<1> ISNT_12:
6378	000020A6 80CC90	<1> OR AH,MED_DET+RATE_250 ; MUST BE RATE 250
6379		<1>
6380		<1> SKP_STATE:
6381	000020A9 88A7[39710000]	<1> MOV [DSK_STATE+eDI], AH ; STORE AWAY
6382		<1> F1_OUT:
6383	000020AF C3	<1> RETn
6384		<1> CL_DRV:
6385	000020B0 30E4	<1> XOR AH,AH ; CLEAR STATE
6386	000020B2 EBF5	<1> JMP SHORT SKP_STATE ; SAVE IT
6387		<1>
6388		<1> ;-----
6389		<1> ; MED_CHANGE
6390		<1> ; CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
6391		<1> ; CHECKS MEDIA CHANGE AGAIN.
6392		<1> ;
6393		<1> ; ON EXIT: CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
6394		<1> ; @DSKETTE_STATUS = ERROR CODE
6395		<1> ;-----
6396		<1> MED_CHANGE:
6397	000020B4 E887060000	<1> CALL READ_DSKCHNG ; READ DISK CHANCE LINE STATE
6398	000020B9 7447	<1> JZ short MC_OUT ; BYPASS HANDLING DISK CHANGE LINE
6399	000020BB 80A7[39710000]EF	<1> AND byte [DSK_STATE+eDI], ~MED_DET ; CLEAR STATE FOR THIS DRIVE
6400		<1>
6401		<1> ; THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
6402		<1> ; ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
6403		<1> ; BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
6404		<1>
6405	000020C2 6689F9	<1> MOV CX,DI ; CL = DRIVE 0
6406	000020C5 B001	<1> MOV AL,1 ; MOTOR ON BIT MASK
6407	000020C7 D2E0	<1> SHL AL,CL ; TO APPROPRIATE POSITION
6408	000020C9 F6D0	<1> NOT AL ; KEEP ALL BUT MOTOR ON
6409	000020CB FA	<1> CLI ; NO INTERRUPTS
6410	000020CC 2005[2A710000]	<1> AND [MOTOR_STATUS], AL ; TURN MOTOR OFF INDICATOR
6411	000020D2 FB	<1> STI ; INTERRUPTS ENABLED
6412	000020D3 E80F040000	<1> CALL MOTOR_ON ; TURN MOTOR ON
6413		<1>
6414		<1> ;----- THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
6415		<1>
6416	000020D8 E884F9FFFF	<1> CALL DSK_RESET ; RESET NEC
6417	000020DD B501	<1> MOV CH,01H ; MOVE TO CYLINDER 1
6418	000020DF E8FE040000	<1> CALL SEEK ; ISSUE SEEK
6419	000020E4 30ED	<1> XOR CH,CH ; MOVE TO CYLINDER 0
6420	000020E6 E8F7040000	<1> CALL SEEK ; ISSUE SEEK
6421	000020EB C605[2C710000]06	<1> MOV byte [DSKETTE_STATUS], MEDIA_CHANGE ; STORE IN STATUS
6422		<1> OK1:

6423	000020F2	E849060000	<1>	CALL	READ_DSKCHNG		; CHECK MEDIA CHANGED AGAIN
6424	000020F7	7407	<1>	JZ	short OK2		; IF ACTIVE, NO DISKETTE, TIMEOUT
6425			<1>	OK4:			
6426	000020F9	C605[2C710000]80	<1>	MOV	byte [DSKETTE_STATUS],	TIME_OUT	; TIMEOUT IF DRIVE EMPTY
6427			<1>	OK2:			
6428	00002100	F9	<1>	STC			; MEDIA CHANGED, SET CY
6429	00002101	C3	<1>	RETn			
6430			<1>	MC_OUT:			
6431	00002102	F8	<1>	CLC			; NO MEDIA CHANGED, CLEAR CY
6432	00002103	C3	<1>	RETn			
6433			<1>				
6434			<1>				
6435			<1>				
6436			<1>				
6437			<1>				
6438			<1>				
6439			<1>				
6440			<1>				
6441			<1>				
6442	00002104	6650	<1>	PUSH	AX		; SAVE REG.
6443	00002106	8025[34710000]3F	<1>	AND	byte [LASTRATE],	~SEND_MSK	; ELSE CLEAR LAST RATE ATTEMPTED
6444	0000210D	8A87[39710000]	<1>	MOV	AL,	[DSK_STATE+eDI]	; GET RATE STATE OF THIS DRIVE
6445	00002113	24C0	<1>	AND	AL,	SEND_MSK	; KEEP ONLY RATE BITS
6446	00002115	0805[34710000]	<1>	OR	[LASTRATE],	AL	; SAVE NEW RATE FOR NEXT CHECK
6447	0000211B	C0C002	<1>	ROL	AL,	2	; MOVE TO BIT OUTPUT POSITIONS
6448	0000211E	66BAF703	<1>	MOV	DX,	03F7H	; OUTPUT NEW DATA RATE
6449	00002122	EE	<1>	OUT	DX,	AL	
6450	00002123	6658	<1>	POP	AX		; RESTORE REG.
6451	00002125	C3	<1>	RETn			
6452			<1>				
6453			<1>				
6454			<1>				
6455			<1>				
6456			<1>				
6457			<1>				
6458			<1>				
6459			<1>				
6460			<1>				
6461			<1>				
6462			<1>				
6463			<1>				
6464	00002126	6650	<1>	PUSH	AX		; SAVE REG
6465	00002128	2225[34710000]	<1>	AND	AH,	[LASTRATE]	; GET LAST DATA RATE SELECTED
6466	0000212E	8A87[39710000]	<1>	MOV	AL,	[DSK_STATE+eDI]	; GET RATE STATE OF THIS DRIVE
6467	00002134	6625C0C0	<1>				
6468	00002138	38E0	<1>				
6469			<1>				
6470	0000213A	6658	<1>				
6471	0000213C	C3	<1>				
6472			<1>				
6473			<1>				
6474			<1>				
6475			<1>				
6476			<1>				
6477			<1>				
6478			<1>				
6479			<1>				
6480			<1>				
6481			<1>				
6482			<1>				
6483			<1>				
6484			<1>				
6485			<1>				
6486			<1>				
6487			<1>				
6488			<1>				
6489			<1>				
6490			<1>				
6491			<1>				
6492			<1>				
6493			<1>				
6494			<1>				
6495			<1>				
6496			<1>				
6497	0000213D	8B5504	<1>	mov	edx,	[ebp+4]	; Buffer address
6498	00002140	F7C2000000FF	<1>	test	edx,	0FF000000h	; 16 MB limit (22/08/2015, bugfix)
6499	00002146	756D	<1>	jnz	short	dma_bnd_err_stc	
6500			<1>				
6501	00002148	6650	<1>	push	ax		; DMA command
6502	0000214A	52	<1>	push	edx		; *
6503	0000214B	B203	<1>	mov	dl,	3	; GET BYTES/SECTOR PARAMETER
6504	0000214D	E850030000	<1>	call	GET_PARM		
6505	00002152	88E1	<1>	mov	cl,	ah	; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
6506	00002154	6689F0	<1>	mov	ax,	si	; Sector count
6507	00002157	88C4	<1>	mov	ah,	al	; AH = # OF SECTORS
6508	00002159	28C0	<1>	sub	al,	al	; AL = 0, AX = # SECTORS * 256
6509	0000215B	66D1E8	<1>	shr	ax,	1	; AX = # SECTORS * 128
6510	0000215E	66D3E0	<1>	shl	ax,	cl	; SHIFT BY PARAMETER VALUE
6511	00002161	6648	<1>	dec	ax		; -1 FOR DMA VALUE
6512	00002163	6689C1	<1>	mov	cx,	ax	
6513	00002166	5A	<1>	pop	edx		; *
6514	00002167	6658	<1>	pop	ax		

```

6515 00002169 3C42      <1>  cmp    al, 42h
6516 0000216B 7507      <1>      jne     short NOT_VERF
6517 0000216D BA0000FF00 <1>  mov    edx, 0FF0000h
6518 00002172 EB08      <1>      jmp     short J33
6519                      <1> NOT_VERF:
6520 00002174 6601CA      <1>      add    dx, cx          ; check for overflow
6521 00002177 723D      <1>      jc     short dma_bnd_err
6522                      <1>      ;
6523 00002179 6629CA      <1>      sub    dx, cx          ; Restore start address
6524                      <1> J33:
6525 0000217C FA          <1>      CLI             ; DISABLE INTERRUPTS DURING DMA SET-UP
6526 0000217D E60C      <1>      OUT     DMA+12,AL      ; SET THE FIRST/LA5T F/F
6527                      <1>      IODELAY          ; WAIT FOR I/O
6528 0000217F EB00      <2>      jmp     short $+2
6529 00002181 EB00      <2>      jmp     short $+2
6530 00002183 E60B      <1>      OUT     DMA+11,AL      ; OUTPUT THE MODE BYTE
6531 00002185 89D0      <1>      mov    eax, edx        ; Buffer address
6532 00002187 E604      <1>      OUT     DMA+4,AL      ; OUTPUT LOW ADDRESS
6533                      <1>      IODELAY          ; WAIT FOR I/O
6534 00002189 EB00      <2>      jmp     short $+2
6535 0000218B EB00      <2>      jmp     short $+2
6536 0000218D 88E0      <1>      MOV     AL,AH
6537 0000218F E604      <1>      OUT     DMA+4,AL      ; OUTPUT HIGH ADDRESS
6538 00002191 C1E810    <1>      shr    eax, 16
6539                      <1>      IODELAY          ; I/O WAIT STATE
6540 00002194 EB00      <2>      jmp     short $+2
6541 00002196 EB00      <2>      jmp     short $+2
6542 00002198 E681      <1>      OUT     081H,AL      ; OUTPUT highest BITS TO PAGE REGISTER
6543                      <1>      IODELAY
6544 0000219A EB00      <2>      jmp     short $+2
6545 0000219C EB00      <2>      jmp     short $+2
6546 0000219E 6689C8    <1>      mov    ax, cx          ; Byte count - 1
6547 000021A1 E605      <1>      OUT     DMA+5,AL      ; LOW BYTE OF COUNT
6548                      <1>      IODELAY          ; WAIT FOR I/O
6549 000021A3 EB00      <2>      jmp     short $+2
6550 000021A5 EB00      <2>      jmp     short $+2
6551 000021A7 88E0      <1>      MOV     AL, AH
6552 000021A9 E605      <1>      OUT     DMA+5,AL      ; HIGH BYTE OF COUNT
6553                      <1>      IODELAY
6554 000021AB EB00      <2>      jmp     short $+2
6555 000021AD EB00      <2>      jmp     short $+2
6556 000021AF FB          <1>      STI             ; RE-ENABLE INTERRUPTS
6557 000021B0 B002      <1>      MOV     AL, 2          ; MODE FOR 8237
6558 000021B2 E60A      <1>      OUT     DMA+10, AL      ; INITIALIZE THE DISKETTE CHANNEL
6559 000021B4 C3          <1>      retn
6560                      <1> dma_bnd_err_stc:
6561 000021B5 F9          <1>      stc
6562                      <1> dma_bnd_err:
6563 000021B6 C605[2C710000]09 <1>  MOV     byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
6564 000021BD C3          <1>      RETn             ; CY SET BY ABOVE IF ERROR
6565                      <1>
6566                      <1> ;; 16/12/2014
6567                      <1> ;;      CLI             ; DISABLE INTERRUPTS DURING DMA SET-UP
6568                      <1> ;;      OUT     DMA+12,AL      ; SET THE FIRST/LA5T F/F
6569                      <1> ;;      ;JMP $+2          ; WAIT FOR I/O
6570                      <1> ;;      IODELAY
6571                      <1> ;;      OUT     DMA+11,AL      ; OUTPUT THE MODE BYTE
6572                      <1> ;;      ;SIODELAY
6573                      <1> ;;      ;CMP AL, 42H          ; DMA VERIFY COMMAND
6574                      <1> ;;      ;JNE short NOT_VERF      ; NO
6575                      <1> ;;      ;XOR AX, AX          ; START ADDRESS
6576                      <1> ;;      ;JMP SHORT J33
6577                      <1> ;;;NOT_VERF:
6578                      <1> ;;      ;MOV AX,ES          ; GET THE ES VALUE
6579                      <1> ;;      ;ROL AX,4          ; ROTATE LEFT
6580                      <1> ;;      ;MOV CH,AL          ; GET HIGHEST NIBBLE OF ES TO CH
6581                      <1> ;;      ;AND AL,11110000B      ; ZERO THE LOW NIBBLE FROM SEGMENT
6582                      <1> ;;      ;ADD AX,[BP+2]          ; TEST FOR CARRY FROM ADDITION
6583                      <1> ;;      mov    eax, [ebp+4] ; 06/02/2015
6584                      <1> ;;      ;JNC short J33
6585                      <1> ;;      ;INC CH          ; CARRY MEANS HIGH 4 BITS MUST BE INC
6586                      <1> ;;;J33:
6587                      <1> ;;      PUSH    eAX          ; SAVE START ADDRESS
6588                      <1> ;;      OUT     DMA+4,AL      ; OUTPUT LOW ADDRESS
6589                      <1> ;;      ;JMP $+2          ; WAIT FOR I/O
6590                      <1> ;;      IODELAY
6591                      <1> ;;      MOV     AL,AH
6592                      <1> ;;      OUT     DMA+4,AL      ; OUTPUT HIGH ADDRESS
6593                      <1> ;;      shr    eax, 16          ; 07/02/2015
6594                      <1> ;;      ;MOV AL,CH          ; GET HIGH 4 BITS
6595                      <1> ;;      ;JMP $+2          ; I/O WAIT STATE
6596                      <1> ;;      IODELAY
6597                      <1> ;;      ;AND AL,00001111B
6598                      <1> ;;      OUT     081H,AL      ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
6599                      <1> ;;      ;SIODELAY
6600                      <1> ;;
6601                      <1> ;;;----- DETERMINE COUNT
6602                      <1> ;;      sub    eax, eax ; 08/02/2015
6603                      <1> ;;      MOV     AX, SI          ; AL = # OF SECTORS
6604                      <1> ;;      XCHG    AL, AH          ; AH = # OF SECTORS
6605                      <1> ;;      SUB     AL, AL          ; AL = 0, AX = # SECTORS * 256
6606                      <1> ;;      SHR     AX, 1          ; AX = # SECTORS * 128

```



```

6607      <1> ;;      PUSH  AX                      ; SAVE # OF SECTORS * 128
6608      <1> ;;      MOV   DL, 3                  ; GET BYTES/SECTOR PARAMETER
6609      <1> ;;      CALL  GET_PARM                ; "
6610      <1> ;;      MOV   CL,AH                    ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
6611      <1> ;;      POP   AX                      ; AX = # SECTORS * 128
6612      <1> ;;      SHL   AX,CL                    ; SHIFT BY PARAMETER VALUE
6613      <1> ;;      DEC   AX                      ; -1 FOR DMA VALUE
6614      <1> ;;      PUSH  eAX ; 08/02/2015        ; SAVE COUNT VALUE
6615      <1> ;;      OUT   DMA+5,AL                  ; LOW BYTE OF COUNT
6616      <1> ;;      ;JMP  $+2                      ; WAIT FOR I/O
6617      <1> ;;      IODELAY
6618      <1> ;;      MOV   AL, AH
6619      <1> ;;      OUT   DMA+5,AL                  ; HIGH BYTE OF COUNT
6620      <1> ;;      ;IODELAY
6621      <1> ;;      STI                               ; RE-ENABLE INTERRUPTS
6622      <1> ;;      POP   eCX ; 08/02/2015        ; RECOVER COUNT VALUE
6623      <1> ;;      POP   eAX ; 08/02/2015        ; RECOVER ADDRESS VALUE
6624      <1> ;;      ;ADD  AX, CX                    ; ADD, TEST FOR 64K OVERFLOW
6625      <1> ;;      add   ecx, eax ; 08/02/2015
6626      <1> ;;      MOV   AL, 2                    ; MODE FOR 8237
6627      <1> ;;      ;JMP  $+2                      ; WAIT FOR I/O
6628      <1> ;;      SIODELAY
6629      <1> ;;      OUT   DMA+10, AL                ; INITIALIZE THE DISKETTE CHANNEL
6630      <1> ;;      ;JNC  short NO_BAD              ; CHECK FOR ERROR
6631      <1> ;;      jc    short dma_bnd_err ; 08/02/2015
6632      <1> ;;      and   ecx, 0FFF0000h ; 16 MB limit
6633      <1> ;;      jz    short NO_BAD
6634      <1> ;;dma_bnd_err:
6635      <1> ;;      MOV   byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
6636      <1> ;;NO_BAD:
6637      <1> ;;      RETn                          ; CY SET BY ABOVE IF ERROR
6638      <1>
6639      <1> ;-----
6640      <1> ; FMTDMA_SET
6641      <1> ; THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT OPERATION.
6642      <1> ;
6643      <1> ; ON ENTRY:  NOTHING REQUIRED
6644      <1> ;
6645      <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
6646      <1> ;-----
6647      <1>
6648      <1> FMTDMA_SET:
6649      <1> ;; 20/02/2015 modification
6650      <1> mov   edx, [ebp+4]                      ; Buffer address
6651      <1> test  edx, 0FFF0000h                    ; 16 MB limit
6652      <1> jnz   short dma_bnd_err_stc
6653      <1> ;
6654      <1> push  dx                                ; *
6655      <1> mov   DL, 4                              ; SECTORS/TRACK VALUE IN PARM TABLE
6656      <1> call  GET_PARM                          ; "
6657      <1> mov   al, ah                              ; AL = SECTORS/TRACK VALUE
6658      <1> sub   ah, ah                              ; AX = SECTORS/TRACK VALUE
6659      <1> shl   ax, 2                              ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
6660      <1> dec   ax                                ; -1 FOR DMA VALUE
6661      <1> mov   cx, ax
6662      <1> pop   dx                                ; *
6663      <1> add   dx, cx                              ; check for overflow
6664      <1> jc    short dma_bnd_err
6665      <1> ;
6666      <1> sub   dx, cx                              ; Restore start address
6667      <1> ;
6668      <1> MOV   AL, 04AH                          ; WILL WRITE TO THE DISKETTE
6669      <1> CLI                               ; DISABLE INTERRUPTS DURING DMA SET-UP
6670      <1> OUT   DMA+12,AL                        ; SET THE FIRST/LAST F/F
6671      <1> IODELAY                                ; WAIT FOR I/O
6672      <2> jmp short $+2
6673      <2> jmp short $+2
6674      <1> OUT   DMA+11,AL                        ; OUTPUT THE MODE BYTE
6675      <1> mov   eax, edx                          ; Buffer address
6676      <1> OUT   DMA+4,AL                        ; OUTPUT LOW ADDRESS
6677      <1> IODELAY                                ; WAIT FOR I/O
6678      <2> jmp short $+2
6679      <2> jmp short $+2
6680      <1> MOV   AL,AH
6681      <1> OUT   DMA+4,AL                        ; OUTPUT HIGH ADDRESS
6682      <1> shr   eax, 16
6683      <1> IODELAY                                ; I/O WAIT STATE
6684      <2> jmp short $+2
6685      <2> jmp short $+2
6686      <1> OUT   081H,AL                          ; OUTPUT highest BITS TO PAGE REGISTER
6687      <1> IODELAY
6688      <2> jmp short $+2
6689      <2> jmp short $+2
6690      <1> mov   ax, cx                              ; Byte count - 1
6691      <1> OUT   DMA+5,AL                        ; LOW BYTE OF COUNT
6692      <1> IODELAY                                ; WAIT FOR I/O
6693      <2> jmp short $+2
6694      <2> jmp short $+2
6695      <1> MOV   AL, AH
6696      <1> OUT   DMA+5,AL                        ; HIGH BYTE OF COUNT
6697      <1> IODELAY
6698      <2> jmp short $+2

```

```

6699 0000221C EB00      <2>  jmp short $+2
6700 0000221E FB        <1>  STI                ; RE-ENABLE INTERRUPTS
6701 0000221F B002      <1>  MOV     AL, 2          ; MODE FOR 8237
6702 00002221 E60A      <1>  OUT     DMA+10, AL       ; INITIALIZE THE DISKETTE CHANNEL
6703 00002223 C3        <1>  retn
6704                    <1>
6705                    <1> ;; 08/02/2015 - Protected Mode Modification
6706                    <1> ;;      MOV     AL, 04AH          ; WILL WRITE TO THE DISKETTE
6707                    <1> ;;      CLI                ; DISABLE INTERRUPTS DURING DMA SET-UP
6708                    <1> ;;      OUT     DMA+12,AL          ; SET THE FIRST/LA5T F/F
6709                    <1> ;;      ;JMP $+2                ; WAIT FOR I/O
6710                    <1> ;;      IODELAY
6711                    <1> ;;      OUT     DMA+11,AL          ; OUTPUT THE MODE BYTE
6712                    <1> ;;      ;MOV     AX,ES            ; GET THE ES VALUE
6713                    <1> ;;      ;ROL     AX,4              ; ROTATE LEFT
6714                    <1> ;;      ;MOV     CH,AL            ; GET HIGHEST NIBBLE OF ES TO CH
6715                    <1> ;;      ;AND     AL,11110000B        ; ZERO THE LOW NIBBLE FROM SEGMENT
6716                    <1> ;;      ;ADD     AX,[BP+2]          ; TEST FOR CARRY FROM ADDITION
6717                    <1> ;;      ;JNC     short J33A
6718                    <1> ;;      ;INC     CH                ; CARRY MEANS HIGH 4 BITS MUST BE INC
6719                    <1> ;;      mov     eax, [ebp+4] ; 08/02/2015
6720                    <1> ;;J33A:
6721                    <1> ;;      PUSH    eAX ; 08/02/2015    ; SAVE START ADDRESS
6722                    <1> ;;      OUT     DMA+4,AL           ; OUTPUT LOW ADDRESS
6723                    <1> ;;      ;JMP $+2                ; WAIT FOR I/O
6724                    <1> ;;      IODELAY
6725                    <1> ;;      MOV     AL,AH
6726                    <1> ;;      OUT     DMA+4,AL           ; OUTPUT HIGH ADDRESS
6727                    <1> ;;      shr     eax, 16 ; 08/02/2015
6728                    <1> ;;      ;MOV     AL,CH            ; GET HIGH 4 BITS
6729                    <1> ;;      ;JMP $+2                ; I/O WAIT STATE
6730                    <1> ;;      IODELAY
6731                    <1> ;;      ;AND     AL,00001111B
6732                    <1> ;;      OUT     081H,AL            ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
6733                    <1> ;;
6734                    <1> ;;;----- DETERMINE COUNT
6735                    <1> ;;      sub     eax, eax ; 08/02/2015
6736                    <1> ;;      MOV     DL, 4              ; SECTORS/TRACK VALUE IN PARM TABLE
6737                    <1> ;;      CALL    GET_PARM            ; "
6738                    <1> ;;      XCHG    AL, AH              ; AL = SECTORS/TRACK VALUE
6739                    <1> ;;      SUB     AH, AH              ; AX = SECTORS/TRACK VALUE
6740                    <1> ;;      SHL     AX, 2              ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
6741                    <1> ;;      DEC     AX                ; -1 FOR DMA VALUE
6742                    <1> ;;      PUSH    eAX ; 08/02/2015    ; SAVE # OF BYTES TO BE TRANSFERED
6743                    <1> ;;      OUT     DMA+5,AL           ; LOW BYTE OF COUNT
6744                    <1> ;;      ;JMP $+2                ; WAIT FOR I/O
6745                    <1> ;;      IODELAY
6746                    <1> ;;      MOV     AL, AH
6747                    <1> ;;      OUT     DMA+5,AL           ; HIGH BYTE OF COUNT
6748                    <1> ;;      STI                ; RE-ENABLE INTERRUPTS
6749                    <1> ;;      POP     eCX ; 08/02/2015    ; RECOVER COUNT VALUE
6750                    <1> ;;      POP     eAX ; 08/02/2015    ; RECOVER ADDRESS VALUE
6751                    <1> ;;      ;ADD     AX, CX            ; ADD, TEST FOR 64K OVERFLOW
6752                    <1> ;;      add     ecx, eax ; 08/02/2015
6753                    <1> ;;      MOV     AL, 2              ; MODE FOR 8237
6754                    <1> ;;      ;JMP $+2                ; WAIT FOR I/O
6755                    <1> ;;      SIODELAY
6756                    <1> ;;      OUT     DMA+10, AL         ; INITIALIZE THE DISKETTE CHANNEL
6757                    <1> ;;      ;JNC     short FMTDMA_OK      ; CHECK FOR ERROR
6758                    <1> ;;      jc      short fmtdma_bnd_err ; 08/02/2015
6759                    <1> ;;      and     ecx, 0FFF0000h ; 16 MB limit
6760                    <1> ;;      jz      short FMTDMA_OK
6761                    <1> ;;      stc ; 20/02/2015
6762                    <1> ;;fmtdma_bnd_err:
6763                    <1> ;;      MOV     byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
6764                    <1> ;;FMTDMA_OK:
6765                    <1> ;;      RETn                ; CY SET BY ABOVE IF ERROR
6766                    <1>
6767                    <1> ;-----
6768                    <1> ; NEC_INIT
6769                    <1> ; THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND INITIALIZES
6770                    <1> ; THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
6771                    <1> ;
6772                    <1> ; ON ENTRY:  AH = NEC COMMAND TO BE PERFORMED
6773                    <1> ;
6774                    <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
6775                    <1> ;-----
6776                    <1> NEC_INIT:
6777 00002224 6650        <1>  PUSH    AX                ; SAVE NEC COMMAND
6778 00002226 E8BC020000 <1>  CALL    MOTOR_ON          ; TURN MOTOR ON FOR SPECIFIC DRIVE
6779                    <1>
6780                    <1> ;----- DO THE SEEK OPERATION
6781                    <1>
6782 0000222B 8A6D01      <1>  MOV     CH,[eBP+1]          ; CH = TRACK #
6783 0000222E E8AF030000 <1>  CALL    SEEK                ; MOVE TO CORRECT TRACK
6784 00002233 6658        <1>  POP     AX                ; RECOVER COMMAND
6785 00002235 721E        <1>  JC      short ER_1          ; ERROR ON SEEK
6786 00002237 BB[55220000] <1>  MOV     eBX, ER_1          ; LOAD ERROR ADDRESS
6787 0000223C 53          <1>  PUSH    eBX                ; PUSH NEC_OUT ERROR RETURN
6788                    <1>
6789                    <1> ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
6790                    <1>

```

6791	0000223D	E866030000	<1>	CALL	NEC_OUTPUT	; OUTPUT THE OPERATION COMMAND
6792	00002242	6689F0	<1>	MOV	AX,SI	; AH = HEAD #
6793	00002245	89FB	<1>	MOV	EBX,eDI	; BL = DRIVE #
6794	00002247	C0E402	<1>	SAL	AH,2	; MOVE IT TO BIT 2
6795	0000224A	80E404	<1>	AND	AH,00000100B	; ISOLATE THAT BIT
6796	0000224D	08DC	<1>	OR	AH,BL	; OR IN THE DRIVE NUMBER
6797	0000224F	E854030000	<1>	CALL	NEC_OUTPUT	; FALL THRU CY SET IF ERROR
6798	00002254	5B	<1>	POP	EBX	; THROW AWAY ERROR RETURN
6799			<1>	ER_1:		
6800	00002255	C3	<1>	RETn		
6801			<1>			
6802			<1>	;-----		
6803			<1>	; RWV_COM		
6804			<1>	; THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE		
6805			<1>	; READ/WRITE/VERIFY OPERATIONS.		
6806			<1>	;		
6807			<1>	; ON ENTRY: CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE		
6808			<1>	; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION		
6809			<1>	;-----		
6810			<1>	RWV_COM:		
6811	00002256	B8[A1220000]	<1>	MOV	EAX, ER_2	; LOAD ERROR ADDRESS
6812	0000225B	50	<1>	PUSH	EAX	; PUSH NEC_OUT ERROR RETURN
6813	0000225C	8A6501	<1>	MOV	AH,[EBP+1]	; OUTPUT TRACK #
6814	0000225F	E844030000	<1>	CALL	NEC_OUTPUT	
6815	00002264	6689F0	<1>	MOV	AX,SI	; OUTPUT HEAD #
6816	00002267	E83C030000	<1>	CALL	NEC_OUTPUT	
6817	0000226C	8A6500	<1>	MOV	AH,[EBP]	; OUTPUT SECTOR #
6818	0000226F	E834030000	<1>	CALL	NEC_OUTPUT	
6819	00002274	B203	<1>	MOV	DL,3	; BYTES/SECTOR PARAMETER FROM BLOCK
6820	00002276	E827020000	<1>	CALL	GET_PARM	; ... TO THE NEC
6821	0000227B	E828030000	<1>	CALL	NEC_OUTPUT	; OUTPUT TO CONTROLLER
6822	00002280	B204	<1>	MOV	DL,4	; EOT PARAMETER FROM BLOCK
6823	00002282	E81B020000	<1>	CALL	GET_PARM	; ... TO THE NEC
6824	00002287	E81C030000	<1>	CALL	NEC_OUTPUT	; OUTPUT TO CONTROLLER
6825	0000228C	8A6305	<1>	MOV	AH,[EBX+MD.GAP]	; GET GAP LENGTH
6826			<1>	R15:		
6827	0000228F	E814030000	<1>	CALL	NEC_OUTPUT	
6828	00002294	B206	<1>	MOV	DL,6	; DTL PARAMETER FROM BLOCK
6829	00002296	E807020000	<1>	CALL	GET_PARM	; TO THE NEC
6830	0000229B	E808030000	<1>	CALL	NEC_OUTPUT	; OUTPUT TO CONTROLLER
6831	000022A0	58	<1>	POP	EAX	; THROW AWAY ERROR EXIT
6832			<1>	ER_2:		
6833	000022A1	C3	<1>	RETn		
6834			<1>			
6835			<1>	;-----		
6836			<1>	; NEC_TERM		
6837			<1>	; THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS		
6838			<1>	; FROM THE NEC FOR THE READ/WRITE/VERIFY/FORWAT OPERATION.		
6839			<1>	;		
6840			<1>	; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION		
6841			<1>	;-----		
6842			<1>	NEC_TERM:		
6843			<1>			
6844			<1>	;----- LET THE OPERATION HAPPEN		
6845			<1>			
6846	000022A2	56	<1>	PUSH	EAX	; SAVE HEAD #, # OF SECTORS
6847	000022A3	E80D040000	<1>	CALL	WAIT_INT	; WAIT FOR THE INTERRUPT
6848	000022A8	9C	<1>	PUSHF		
6849	000022A9	E837040000	<1>	CALL	RESULTS	; GET THE NEC STATUS
6850	000022AE	724B	<1>	JC	short SET_END_POP	
6851	000022B0	9D	<1>	POPF		
6852	000022B1	723E	<1>	JC	short SET_END	; LOOK FOR ERROR
6853			<1>			
6854			<1>	;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER		
6855			<1>			
6856	000022B3	FC	<1>	CLD		; SET THE CORRECT DIRECTION
6857	000022B4	BE[2D710000]	<1>	MOV	EAX, NEC_STATUS	; POINT TO STATUS FIELD
6858	000022B9	AC	<1>	lodsb		; GET ST0
6859	000022BA	24C0	<1>	AND	AL,11000000B	; TEST FOR NORMAL TERMINATION
6860	000022BC	7433	<1>	JZ	short SET_END	
6861	000022BE	3C40	<1>	CMP	AL,01000000B	; TEST FOR ABNORMAL TERMINATION
6862	000022C0	7527	<1>	JNZ	short J18	; NOT ABNORMAL, BAD NEC
6863			<1>			
6864			<1>	;----- ABNORMAL TERMINATION, FIND OUT WHY		
6865			<1>			
6866	000022C2	AC	<1>	lodsb		; GET ST1
6867	000022C3	D0E0	<1>	SAL	AL,1	; TEST FOR EDT FOUND
6868	000022C5	B404	<1>	MOV	AH,RECORD_NOT_FND	
6869	000022C7	7222	<1>	JC	short J19	
6870	000022C9	C0E002	<1>	SAL	AL,2	
6871	000022CC	B410	<1>	MOV	AH,BAD_CRC	
6872	000022CE	721B	<1>	JC	short J19	
6873	000022D0	D0E0	<1>	SAL	AL,1	; TEST FOR DMA OVERRUN
6874	000022D2	B408	<1>	MOV	AH,BAD_DMA	
6875	000022D4	7215	<1>	JC	short J19	
6876	000022D6	C0E002	<1>	SAL	AL,2	; TEST FOR RECORD NOT FOUND
6877	000022D9	B404	<1>	MOV	AH,RECORD_NOT_FND	
6878	000022DB	720E	<1>	JC	short J19	
6879	000022DD	D0E0	<1>	SAL	AL,1	
6880	000022DF	B403	<1>	MOV	AH,WRITE_PROTECT	; TEST FOR WRITE_PROTECT
6881	000022E1	7208	<1>	JC	short J19	
6882	000022E3	D0E0	<1>	SAL	AL,1	; TEST MISSING ADDRESS MARK

6883	000022E5	B402	<1>	MOV	AH,BAD_ADDR_MARK
6884	000022E7	7202	<1>	JC	short J19
6885			<1>		
6886			<1>	;-----	NEC MUST HAVE FAILED
6887			<1>	J18:	
6888	000022E9	B420	<1>	MOV	AH,BAD_NEC
6889			<1>	J19:	
6890	000022EB	0825[2C710000]	<1>	OR	[DSKETTE_STATUS], AH
6891			<1>	SET_END:	
6892	000022F1	803D[2C710000]01	<1>	CMP	byte [DSKETTE_STATUS], 1 ; SET ERROR CONDITION
6893	000022F8	F5	<1>	CMC	
6894	000022F9	5E	<1>	POP	eSI
6895	000022FA	C3	<1>	RETn	; RESTORE HEAD #, # OF SECTORS
6896			<1>		
6897			<1>	SET_END_POP:	
6898	000022FB	9D	<1>	POPF	
6899	000022FC	EBF3	<1>	JMP	SHORT SET_END
6900			<1>		
6901			<1>		
6902			<1>	; DSTATE:	ESTABLISH STATE UPON SUCCESSFUL OPERATION.
6903			<1>		
6904			<1>	DSTATE:	
6905	000022FE	803D[2C710000]00	<1>	CMP	byte [DSKETTE_STATUS],0 ; CHECK FOR ERROR
6906	00002305	753E	<1>	JNZ	short SETBAC ; IF ERROR JUMP
6907	00002307	808F[39710000]10	<1>	OR	byte [DSK_STATE+eDI],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
6908	0000230E	F687[39710000]04	<1>	TEST	byte [DSK_STATE+eDI],DRV_DET ; DRIVE DETERMINED ?
6909	00002315	752E	<1>	JNZ	short SETBAC ; IF DETERMINED NO TRY TO DETERMINE
6910	00002317	8A87[39710000]	<1>	MOV	AL,[DSK_STATE+eDI] ; LOAD STATE
6911	0000231D	24C0	<1>	AND	AL,RATE_MSK ; KEEP ONLY RATE
6912	0000231F	3C80	<1>	CMP	AL,RATE_250 ; RATE 250 ?
6913	00002321	751B	<1>	JNE	short M_12 ; NO, MUST BE 1.2M OR 1.44M DRIVE
6914			<1>		
6915			<1>	;-----	CHECK IF IT IS 1.44M
6916			<1>		
6917	00002323	E871010000	<1>	CALL	CMOS_TYPE ; RETURN DRIVE TYPE IN (AL)
6918			<1>		;20/02/2015
6919			<1>	;JC	short M_12 ; CMOS BAD
6920	00002328	7414	<1>	jz	short M_12 ; 20/02/2015
6921	0000232A	3C04	<1>	CMP	AL, 4 ; 1.44MB DRIVE ?
6922	0000232C	7410	<1>	JE	short M_12 ; YES
6923			<1>	M_720:	
6924	0000232E	80A7[39710000]FD	<1>	AND	byte [DSK_STATE+eDI], ~FMT_CAPA ; TURN OFF FORMAT CAPABILITY
6925	00002335	808F[39710000]04	<1>	OR	byte [DSK_STATE+eDI],DRV_DET ; MARK DRIVE DETERMINED
6926	0000233C	EB07	<1>	JMP	SHORT SETBAC ; BACK
6927			<1>	M_12:	
6928	0000233E	808F[39710000]06	<1>	OR	byte [DSK_STATE+eDI],DRV_DET+FMT_CAPA
6929			<1>		; TURN ON DETERMINED & FMT CAPA
6930			<1>	SETBAC:	
6931	00002345	C3	<1>	RETn	
6932			<1>		
6933			<1>		
6934			<1>	; RETRY	
6935			<1>	; DETERMINES WHETHER A RETRY IS NECESSARY.	
6936			<1>	; IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.	
6937			<1>		
6938			<1>	; ON EXIT: CY = 1 FOR RETRY, CY = 0 FOR NO RETRY	
6939			<1>		
6940			<1>	RETRY:	
6941	00002346	803D[2C710000]00	<1>	CMP	byte [DSKETTE_STATUS],0 ; GET STATUS OF OPERATION
6942	0000234D	7445	<1>	JZ	short NO_RETRY ; SUCCESSFUL OPERATION
6943	0000234F	803D[2C710000]80	<1>	CMP	byte [DSKETTE_STATUS],TIME_OUT ; IF TIME OUT NO RETRY
6944	00002356	743C	<1>	JZ	short NO_RETRY
6945	00002358	8AA7[39710000]	<1>	MOV	AH,[DSK_STATE+eDI] ; GET MEDIA STATE OF DRIVE
6946	0000235E	F6C410	<1>	TEST	AH,MED_DET ; ESTABLISHED/DETERMINED ?
6947	00002361	7531	<1>	JNZ	short NO_RETRY ; IF ESTABLISHED STATE THEN TRUE ERROR
6948	00002363	80E4C0	<1>	AND	AH,RATE_MSK ; ISOLATE RATE
6949	00002366	8A2D[34710000]	<1>	MOV	CH,[LASTRATE] ; GET START OPERATION STATE
6950	0000236C	C0C504	<1>	ROL	CH,4 ; TO CORRESPONDING BITS
6951	0000236F	80E5C0	<1>	AND	CH,RATE_MSK ; ISOLATE RATE BITS
6952	00002372	38E5	<1>	CMP	CH,AH ; ALL RATES TRIED
6953	00002374	741E	<1>	JE	short NO_RETRY ; IF YES, THEN TRUE ERROR
6954			<1>		
6955			<1>	; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE	
6956			<1>		; 00000000B (500) -> 10000000B (250)
6957			<1>		; 10000000B (250) -> 01000000B (300)
6958			<1>		; 01000000B (300) -> 00000000B (500)
6959			<1>		
6960	00002376	80FC01	<1>	CMP	AH,RATE_500+1 ; SET CY FOR RATE 500
6961	00002379	D0DC	<1>	RCR	AH,1 ; TO NEXT STATE
6962	0000237B	80E4C0	<1>	AND	AH,RATE_MSK ; KEEP ONLY RATE BITS
6963	0000237E	80A7[39710000]1F	<1>	AND	byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP)
6964			<1>		; RATE, DBL STEP OFF
6965	00002385	08A7[39710000]	<1>	OR	[DSK_STATE+eDI],AH ; TURN ON NEW RATE
6966	0000238B	C605[2C710000]00	<1>	MOV	byte [DSKETTE_STATUS],0 ; RESET STATUS FOR RETRY
6967	00002392	F9	<1>	STC	; SET CARRY FOR RETRY
6968	00002393	C3	<1>	RETn	; RETRY RETURN
6969			<1>		
6970			<1>	NO_RETRY:	
6971	00002394	F8	<1>	CLC	; CLEAR CARRY NO RETRY
6972	00002395	C3	<1>	RETn	; NO RETRY RETURN
6973			<1>		
6974			<1>		

```

6975 <1> ; NUM_TRANS
6976 <1> ; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
6977 <1> ; ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
6978 <1> ;
6979 <1> ; ON ENTRY: [BP+1] = TRACK
6980 <1> ; SI-HI = HEAD
6981 <1> ; [BP] = START SECTOR
6982 <1> ;
6983 <1> ; ON EXIT: AL = NUMBER ACTUALLY TRANSFERRED
6984 <1> ;-----
6985 <1> NUM_TRANS:
6986 00002396 30C0 <1> XOR AL,AL ; CLEAR FOR ERROR
6987 00002398 803D[2C710000]00 <1> CMP byte [DSKETTE_STATUS],0 ; CHECK FOR ERROR
6988 0000239F 752C <1> JNZ NT_OUT ; IF ERROR 0 TRANSFERRED
6989 000023A1 B204 <1> MOV DL,4 ; SECTORS/TRACK OFFSET TO DL
6990 000023A3 E8FA000000 <1> CALL GET_PARM ; AH = SECTORS/TRACK
6991 000023A8 8A1D[32710000] <1> MOV BL,[NEC_STATUS+5] ; GET ENDING SECTOR
6992 000023AE 6689F1 <1> MOV CX,SI ; CH = HEAD # STARTED
6993 000023B1 3A2D[31710000] <1> CMP CH,[NEC_STATUS+4] ; GET HEAD ENDED UP ON
6994 000023B7 750D <1> JNZ DIF_HD ; IF ON SAME HEAD, THEN NO ADJUST
6995 000023B9 8A2D[30710000] <1> MOV CH,[NEC_STATUS+3] ; GET TRACK ENDED UP ON
6996 000023BF 3A6D01 <1> CMP CH,[eBP+1] ; IS IT ASKED FOR TRACK
6997 000023C2 7404 <1> JZ short SAME_TRK ; IF SAME TRACK NO INCREASE
6998 000023C4 00E3 <1> ADD BL,AH ; ADD SECTORS/TRACK
6999 <1> DIF_HD:
7000 000023C6 00E3 <1> ADD BL,AH ; ADD SECTORS/TRACK
7001 <1> SAME_TRK:
7002 000023C8 2A5D00 <1> SUB BL,[eBP] ; SUBTRACT START FROM END
7003 000023CB 88D8 <1> MOV AL,BL ; TO AL
7004 <1> NT_OUT:
7005 000023CD C3 <1> RETn
7006 <1>
7007 <1> ;-----
7008 <1> ; SETUP_END
7009 <1> ; RESTORES @MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE
7010 <1> ; AND LOADS @DSKETTE_STATUS TO AH, AND SETS CY.
7011 <1> ;
7012 <1> ; ON EXIT:
7013 <1> ; AH, @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
7014 <1> ;-----
7015 <1> SETUP_END:
7016 000023CE B202 <1> MOV DL,2 ; GET THE MOTOR WAIT PARAMETER
7017 000023D0 6650 <1> PUSH AX ; SAVE NUMBER TRANSFERRED
7018 000023D2 E8CB000000 <1> CALL GET_PARM
7019 000023D7 8825[2B710000] <1> MOV [MOTOR_COUNT],AH ; STORE UPON RETURN
7020 000023DD 6658 <1> POP AX ; RESTORE NUMBER TRANSFERRED
7021 000023DF 8A25[2C710000] <1> MOV AH,[DSKETTE_STATUS] ; GET STATUS OF OPERATION
7022 000023E5 08E4 <1> OR AH,AH ; CHECK FOR ERROR
7023 000023E7 7402 <1> JZ short NUN_ERR ; NO ERROR
7024 000023E9 30C0 <1> XOR AL,AL ; CLEAR NUMBER RETURNED
7025 <1> NUN_ERR:
7026 000023EB 80FC01 <1> CMP AH,1 ; SET THE CARRY FLAG TO INDICATE
7027 000023EE F5 <1> CMC ; SUCCESS OR FAILURE
7028 000023EF C3 <1> RETn
7029 <1>
7030 <1> ;-----
7031 <1> ; SETUP_DBL
7032 <1> ; CHECK DOUBLE STEP.
7033 <1> ;
7034 <1> ; ON ENTRY : DI = DRIVE
7035 <1> ;
7036 <1> ; ON EXIT : CY = 1 MEANS ERROR
7037 <1> ;-----
7038 <1> SETUP_DBL:
7039 000023F0 8AA7[39710000] <1> MOV AH,[DSK_STATE+eDI] ; ACCESS STATE
7040 000023F6 F6C410 <1> TEST AH,MED_DET ; ESTABLISHED STATE ?
7041 000023F9 757E <1> JNZ short NO_DBL ; IF ESTABLISHED THEN DOUBLE DONE
7042 <1>
7043 <1> ;----- CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
7044 <1>
7045 000023FB C605[29710000]00 <1> MOV byte [SEEK_STATUS],0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES
7046 00002402 E8E0000000 <1> CALL MOTOR_ON ; ENSURE MOTOR STAY ON
7047 00002407 B500 <1> MOV CH,0 ; LOAD TRACK 0
7048 00002409 E8D4010000 <1> CALL SEEK ; SEEK TO TRACK 0
7049 0000240E E868000000 <1> CALL READ_ID ; READ ID FUNCTION
7050 00002413 7249 <1> JC short SD_ERR ; IF ERROR NO TRACK 0
7051 <1>
7052 <1> ;----- INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
7053 <1>
7054 00002415 66B95004 <1> MOV CX,0450H ; START, MAX TRACKS
7055 00002419 F687[39710000]01 <1> TEST byte [DSK_STATE+eDI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
7056 00002420 7402 <1> JZ short CNT_OK ; IF NOT COUNT IS SETUP
7057 00002422 B1A0 <1> MOV CL,0A0H ; MAXIMUM TRACK 1.2 MB
7058 <1>
7059 <1> ; ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
7060 <1> ; MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
7061 <1> ; THEN SET DOUBLE STEP ON.
7062 <1>
7063 <1> CNT_OK:
7064 00002424 C605[2B710000]FF <1> MOV byte [MOTOR_COUNT],0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
7065 0000242B 6651 <1> PUSH CX ; SAVE TRACK, COUNT
7066 0000242D C605[2C710000]00 <1> MOV byte [DSKETTE_STATUS],0 ; CLEAR STATUS, EXPECT ERRORS

```

```

7067 00002434 6631C0      <1> XOR    AX,AX                ; CLEAR AX
7068 00002437 D0ED        <1> SHR     CH,1                ; HALVE TRACK, CY = HEAD
7069 00002439 C0D003      <1> RCL     AL,3                ; AX = HEAD IN CORRECT BIT
7070 0000243C 6650        <1> PUSH    AX                ; SAVE HEAD
7071 0000243E E89F010000   <1> CALL    SEEK                ; SEEK TO TRACK
7072 00002443 6658        <1> POP     AX                ; RESTORE HEAD
7073 00002445 6609C7      <1> OR      DI,AX                ; DI = HEAD OR'ED DRIVE
7074 00002448 E82E000000   <1> CALL    READ_ID              ; READ ID HEAD 0
7075 0000244D 9C          <1> PUSHF                    ; SAVE RETURN FROM READ_ID
7076 0000244E 6681E7FB00   <1> AND     DI,11111011B        ; TURN OFF HEAD 1 BIT
7077 00002453 9D          <1> POPF                     ; RESTORE ERROR RETURN
7078 00002454 6659        <1> POP     CX                ; RESTORE COUNT
7079 00002456 7308        <1> JNC     short DO_CHK         ; IF OK, ASKED = RETURNED TRACK ?
7080 00002458 FEC5        <1> INC     CH                ; INC FOR NEXT TRACK
7081 0000245A 38CD        <1> CMP     CH,CL              ; REACHED MAXIMUM YET
7082 0000245C 75C6        <1> JNZ     short CNT_OK         ; CONTINUE TILL ALL TRIED
7083                                     <1>
7084                                     <1> ;----- FALL THRU, READ ID FAILED FOR ALL TRACKS
7085                                     <1>
7086                                     <1> SD_ERR:
7087 0000245E F9          <1> STC                      ; SET CARRY FOR ERROR
7088 0000245F C3          <1> RETn                     ; SETUP_DBL ERROR EXIT
7089                                     <1>
7090                                     <1> DO_CHK:
7091 00002460 8A0D[30710000] <1> MOV     CL, [NEC_STATUS+3]    ; LOAD RETURNED TRACK
7092 00002466 888F[3D710000] <1> MOV     [DSK_TRK+eDI], CL    ; STORE TRACK NUMBER
7093 0000246C D0ED        <1> SHR     CH,1                ; HALVE TRACK
7094 0000246E 38CD        <1> CMP     CH,CL              ; IS IT THE SAME AS ASKED FOR TRACK
7095 00002470 7407        <1> JZ      short NO_DBL        ; IF SAME THEN NO DOUBLE STEP
7096 00002472 808F[39710000]20 <1> OR     byte [DSK_STATE+eDI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
7097                                     <1> NO_DBL:
7098 00002479 F8          <1> CLC                      ; CLEAR ERROR FLAG
7099 0000247A C3          <1> RETn
7100                                     <1>
7101                                     <1> ;-----
7102                                     <1> ; READ_ID
7103                                     <1> ; READ ID FUNCTION.
7104                                     <1> ;
7105                                     <1> ; ON ENTRY:  DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
7106                                     <1> ;
7107                                     <1> ; ON EXIT:   DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
7108                                     <1> ; @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
7109                                     <1> ;-----
7110                                     <1> READ_ID:
7111 0000247B B8[98240000] <1> MOV     eAX, ER_3            ; MOVE NEC OUTPUT ERROR ADDRESS
7112 00002480 50          <1> PUSH    eAX
7113 00002481 B44A        <1> MOV     AH,4AH              ; READ ID COMMAND
7114 00002483 E820010000   <1> CALL    NEC_OUTPUT          ; TO CONTROLLER
7115 00002488 6689F8      <1> MOV     AX,DI              ; DRIVE # TO AH, HEAD 0
7116 0000248B 88C4        <1> MOV     AH,AL
7117 0000248D E816010000   <1> CALL    NEC_OUTPUT          ; TO CONTROLLER
7118 00002492 E80BF0FFFF   <1> CALL    NEC_TERM            ; WAIT FOR OPERATION, GET STATUS
7119 00002497 58          <1> POP     eAX                ; THROW AWAY ERROR ADDRESS
7120                                     <1> ER_3:
7121 00002498 C3          <1> RETn
7122                                     <1>
7123                                     <1> ;-----
7124                                     <1> ; CMOS_TYPE
7125                                     <1> ; RETURNS DISKETTE TYPE FROM CMOS
7126                                     <1> ;
7127                                     <1> ; ON ENTRY:  DI = DRIVE #
7128                                     <1> ;
7129                                     <1> ; ON EXIT:   AL = TYPE; CY REFLECTS STATUS
7130                                     <1> ;-----
7131                                     <1>
7132                                     <1> CMOS_TYPE: ; 11/12/2014
7133 00002499 8A87[D86D0000] <1> mov     al, [eDI+fd0_type]
7134 0000249F 20C0        <1> and     al, al ; 18/12/2014
7135 000024A1 C3          <1> retn
7136                                     <1>
7137                                     <1> ;CMOS_TYPE:
7138 <1> ;MOV     AL, CMOS_DIAG        ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
7139 <1> ;CALL    CMOS_READ            ; GET CMOS STATUS
7140 <1> ;TEST    AL,BAD_BAT+BAD_CKSUM  ; BATTERY GOOD AND CHECKSUM VALID
7141 <1> ;STC                      ; SET CY = 1 INDICATING ERROR FOR RETURN
7142 <1> ;JNZ     short BAD_CM         ; ERROR IF EITHER BIT ON
7143 <1> ;MOV     AL,CMOS_DISKETTE     ; ADDRESS OF DISKETTE BYTE IN CMOS
7144 <1> ;CALL    CMOS_READ            ; GET DISKETTE BYTE
7145 <1> ;OR      DI,DI                ; SEE WHICH DRIVE IN QUESTION
7146 <1> ;JNZ     short TB             ; IF DRIVE 1, DATA IN LOW NIBBLE
7147 <1> ;ROR     AL,4                ; EXCHANGE NIBBLES IF SECOND DRIVE
7148 <1> ;TB:
7149 <1> ;AND     AL,0FH                ; KEEP ONLY DRIVE DATA, RESET CY, 0
7150 <1> ;BAD_CM:
7151 <1> ;RETn                     ; CY, STATUS OF READ
7152 <1>
7153 <1> ;-----
7154 <1> ; GET_PARM
7155 <1> ; THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK_BASE
7156 <1> ; BLOCK POINTED TO BY THE DATA VARIABLE @DISK_POINTER. A BYTE FROM
7157 <1> ; THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
7158 <1> ; THE PARAMETER IN DI.

```

```

7159 <1> ;
7160 <1> ; ON ENTRY: DL = INDEX OF BYTE TO BE FETCHED
7161 <1> ;
7162 <1> ; ON EXIT: AH = THAT BYTE FROM BLOCK
7163 <1> ; AL,DH DESTROYED
7164 <1> ;-----
7165 <1> GET_PARM:
7166 <1> ;PUSH DS
7167 000024A2 56 <1> PUSH esi
7168 <1> ;SUB AX,AX ; DS = 0, BIOS DATA AREA
7169 <1> ;MOV DS,AX
7170 <1> ;;mov ax, cs
7171 <1> ;;mov ds, ax
7172 <1> ; 08/02/2015 (protected mode modifications, bx -> ebx)
7173 000024A3 87D3 <1> XCHG eDX,eBX ; BL = INDEX
7174 <1> ;SUB BH,BH ; BX = INDEX
7175 000024A5 81E3FF000000 <1> and ebx, 0FFh
7176 <1> ;LDS SI, [DISK_POINTER] ; POINT TO BLOCK
7177 <1> ;
7178 <1> ; 17/12/2014
7179 000024AB 66A1[516B0000] <1> mov ax, [cfd] ; current (AL) and previous fd (AH)
7180 000024B1 38E0 <1> cmp al, ah
7181 000024B3 7425 <1> je short gpndc
7182 000024B5 A2[526B0000] <1> mov [pfd], al ; current drive -> previous drive
7183 000024BA 53 <1> push ebx ; 08/02/2015
7184 000024BB 88C3 <1> mov bl, al
7185 <1> ; 11/12/2014
7186 000024BD 8A83[D86D0000] <1> mov al, [eBX+fd0_type] ; Drive type (0,1,2,3,4)
7187 <1> ; 18/12/2014
7188 000024C3 20C0 <1> and al, al
7189 000024C5 7507 <1> jnz short gpdtc
7190 000024C7 BB[3B6B0000] <1> mov ebx, MD_TBL6 ; 1.44 MB param. tbl. (default)
7191 000024CC EB05 <1> jmp short gpdpu
7192 <1> gpdtc:
7193 000024CE E818F9FFFF <1> call DR_TYPE_CHECK
7194 <1> ; cf = 1 -> eBX points to 1.44MB fd parameter table (default)
7195 <1> gpdpu:
7196 000024D3 891D[D86A0000] <1> mov [DISK_POINTER], ebx
7197 000024D9 5B <1> pop ebx
7198 <1> gpndc:
7199 000024DA 8B35[D86A0000] <1> mov esi, [DISK_POINTER] ; 08/02/2015, si -> esi
7200 000024E0 8A241E <1> MOV AH, [eSI+eBX] ; GET THE WORD
7201 000024E3 87D3 <1> XCHG eDX,eBX ; RESTORE BX
7202 000024E5 5E <1> POP esi
7203 <1> ;POP DS
7204 000024E6 C3 <1> RETn
7205 <1>
7206 <1> ;-----
7207 <1> ; MOTOR_ON
7208 <1> ; TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT
7209 <1> ; IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE
7210 <1> ; THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
7211 <1> ; MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
7212 <1> ; (AX=90FDH, INT 15) IS CALLED TELLING THE OPERATING SYSTEM
7213 <1> ; THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
7214 <1> ; FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
7215 <1> ; HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
7216 <1> ; THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
7217 <1> ; NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
7218 <1> ; PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
7219 <1> ; IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
7220 <1> ; WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.
7221 <1> ;
7222 <1> ; ON ENTRY: DI = DRIVE #
7223 <1> ; ON EXIT: AX,CX,DX DESTROYED
7224 <1> ;-----
7225 <1> MOTOR_ON:
7226 000024E7 53 <1> PUSH eBX ; SAVE REG.
7227 000024E8 E82A000000 <1> CALL TURN_ON ; TURN ON MOTOR
7228 000024ED 7226 <1> JC short MOT_IS_ON ; IF CY=1 NO WAIT
7229 000024EF E89CF9FFFF <1> CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
7230 000024F4 E866F9FFFF <1> CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH,
7231 <1> ;CALL TURN_ON ; CHECK AGAIN IF MOTOR ON
7232 <1> ;JC MOT_IS_ON ; IF NO WAIT MEANS IT IS ON
7233 <1> M_WAIT:
7234 000024F9 B20A <1> MOV DL,10 ; GET THE MOTOR WAIT PARAMETER
7235 000024FB E8A2FFFFFF <1> CALL GET_PARM
7236 <1> ;MOV AL,AH ; AL = MOTOR WAIT PARAMETER
7237 <1> ;XOR AH,AH ; AX = MOTOR WAIT PARAMETER
7238 <1> ;CMP AL,8 ; SEE IF AT LEAST A SECOND IS SPECIFIED
7239 00002500 80FC08 <1> cmp ah, 8
7240 <1> ;JAE short GP2 ; IF YES, CONTINUE
7241 00002503 7702 <1> ja short J13
7242 <1> ;MOV AL,8 ; ONE SECOND WAIT FOR MOTOR START UP
7243 00002505 B408 <1> mov ah, 8
7244 <1>
7245 <1> ;----- AS CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
7246 <1> GP2:
7247 <1> ;----- FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
7248 <1> J13: ; WAIT FOR 1/8 SECOND PER (AL)
7249 00002507 B95E200000 <1> MOV eCX,8286 ; COUNT FOR 1/8 SECOND AT 15.085737 US
7250 0000250C E81BF1FFFF <1> CALL WAITF ; GO TO FIXED WAIT ROUTINE

```

7251		<1>	;DEC AL	; DECREMENT TIME VALUE
7252	00002511 FECC	<1>	dec ah	
7253	00002513 75F2	<1>	JNZ short J13	; ARE WE DONE YET
7254		<1>	MOT_IS_ON:	
7255	00002515 5B	<1>	POP eBX	; RESTORE REG.
7256	00002516 C3	<1>	RETn	
7257		<1>		
7258		<1>	;-----	
7259		<1>	; TURN_ON	
7260		<1>	; TURN MOTOR ON AND RETURN WAIT STATE.	
7261		<1>		
7262		<1>	; ON ENTRY: DI = DRIVE #	
7263		<1>		
7264		<1>	; ON EXIT: CY = 0 MEANS WAIT REQUIRED	
7265		<1>	; CY = 1 MEANS NO WAIT REQUIRED	
7266		<1>	; AX,BX,CX,DX DESTROYED	
7267		<1>	;-----	
7268		<1>	TURN_ON:	
7269	00002517 89FB	<1>	MOV eBX,eDI	; BX = DRIVE #
7270	00002519 88D9	<1>	MOV CL,BL	; CL = DRIVE #
7271	0000251B C0C304	<1>	ROL BL,4	; BL = DRIVE SELECT
7272	0000251E FA	<1>	CLI	; NO INTERRUPTS WHILE DETERMINING STATUS
7273	0000251F C605[2B710000]FF	<1>	MOV byte [MOTOR_COUNT],0FFh	; ENSURE MOTOR STAYS ON FOR OPERATION
7274	00002526 A0[2A710000]	<1>	MOV AL, [MOTOR_STATUS]	; GET DIGITAL OUTPUT REGISTER REFLECTION
7275	0000252B 2430	<1>	AND AL,00110000B	; KEEP ONLY DRIVE SELECT BITS
7276	0000252D B401	<1>	MOV AH,1	; MASK FOR DETERMINING MOTOR BIT
7277	0000252F D2E4	<1>	SHL AH,CL	; AH = MOTOR ON, A=00000001, B=00000010
7278		<1>		
7279		<1>	; AL = DRIVE SELECT FROM @MOTOR_STATUS	
7280		<1>	; BL = DRIVE SELECT DESIRED	
7281		<1>	; AH = MOTOR ON MASK DESIRED	
7282		<1>		
7283	00002531 38D8	<1>	CMP AL,BL	; REQUESTED DRIVE ALREADY SELECTED ?
7284	00002533 7508	<1>	JNZ short TURN_IT_ON	; IF NOT SELECTED JUMP
7285	00002535 8425[2A710000]	<1>	TEST AH, [MOTOR_STATUS]	; TEST MOTOR ON BIT
7286	0000253B 7535	<1>	JNZ short NO_MOT_WAIT	; JUMP IF MOTOR ON AND SELECTED
7287		<1>		
7288		<1>	TURN_IT_ON:	
7289	0000253D 08DC	<1>	OR AH,BL	; AH = DRIVE SELECT AND MOTOR ON
7290	0000253F 8A3D[2A710000]	<1>	MOV BH,[MOTOR_STATUS]	; SAVE COPY OF @MOTOR_STATUS BEFORE
7291	00002545 80E70F	<1>	AND BH,00001111B	; KEEP ONLY MOTOR BITS
7292	00002548 8025[2A710000]CF	<1>	AND byte [MOTOR_STATUS],11001111B	; CLEAR OUT DRIVE SELECT
7293	0000254F 0825[2A710000]	<1>	OR [MOTOR_STATUS],AH	; OR IN DRIVE SELECTED AND MOTOR ON
7294	00002555 A0[2A710000]	<1>	MOV AL,[MOTOR_STATUS]	; GET DIGITAL OUTPUT REGISTER REFLECTION
7295	0000255A 88C3	<1>	MOV BL,AL	; BL=@MOTOR_STATUS AFTER, BH=BEFORE
7296	0000255C 80E30F	<1>	AND BL,00001111B	; KEEP ONLY MOTOR BITS
7297	0000255F FB	<1>	STI	; ENABLE INTERRUPTS AGAIN
7298	00002560 243F	<1>	AND AL,00111111B	; STRIP AWAY UNWANTED BITS
7299	00002562 C0C004	<1>	ROL AL,4	; PUT BITS IN DESIRED POSITIONS
7300	00002565 0C0C	<1>	OR AL,00001100B	; NO RESET, ENABLE DMA/INTERRUPT
7301	00002567 66BAF203	<1>	MOV DX,03F2H	; SELECT DRIVE AND TURN ON MOTOR
7302	0000256B EE	<1>	OUT DX,AL	
7303	0000256C 38FB	<1>	CMP BL,BH	; NEW MOTOR TURNED ON ?
7304		<1>	;JZ short NO_MOT_WAIT	; NO WAIT REQUIRED IF JUST SELECT
7305	0000256E 7403	<1>	je short no_mot_w1	; 27/02/2015
7306	00002570 F8	<1>	CLC	; (re)SET CARRY MEANING WAIT
7307	00002571 C3	<1>	RETn	
7308		<1>		
7309		<1>	NO_MOT_WAIT:	
7310	00002572 FB	<1>	sti	
7311		<1>	no_mot_w1: ; 27/02/2015	
7312	00002573 F9	<1>	STC	; SET NO WAIT REQUIRED
7313		<1>	;STI	; INTERRUPTS BACK ON
7314	00002574 C3	<1>	RETn	
7315		<1>		
7316		<1>	;-----	
7317		<1>	; HD_WAIT	
7318		<1>	; WAIT FOR HEAD SETTLE TIME.	
7319		<1>		
7320		<1>	; ON ENTRY: DI = DRIVE #	
7321		<1>		
7322		<1>	; ON EXIT: AX,BX,CX,DX DESTROYED	
7323		<1>	;-----	
7324		<1>	HD_WAIT:	
7325	00002575 B209	<1>	MOV DL,9	; GET HEAD SETTLE PARAMETER
7326	00002577 E826FFFFFF	<1>	CALL GET_PARM	
7327	0000257C 08E4	<1>	or ah, ah	; 17/12/2014
7328	0000257E 7519	<1>	jnz short DO_WAT	
7329	00002580 F605[2A710000]80	<1>	TEST byte [MOTOR_STATUS],10000000B	; SEE IF A WRITE OPERATION
7330		<1>	;JZ short ISNT_WRITE	; IF NOT, DO NOT ENFORCE ANY VALUES
7331		<1>	;OR AH,AH	; CHECK FOR ANY WAIT?
7332		<1>	;JNZ short DO_WAT	; IF THERE DO NOT ENFORCE
7333	00002587 741E	<1>	jz short HW_DONE	
7334	00002589 B40F	<1>	MOV AH,HD12_SETTLE	; LOAD 1.2M HEAD SETTLE MINIMUM
7335	0000258B 8A87[39710000]	<1>	MOV AL,[DSK_STATE+eDI]	; LOAD STATE
7336	00002591 24C0	<1>	AND AL,RATE_MSK	; KEEP ONLY RATE
7337	00002593 3C80	<1>	CMP AL,RATE_250	; 1.2 M DRIVE ?
7338	00002595 7502	<1>	JNZ short DO_WAT	; DEFAULT HEAD SETTLE LOADED
7339		<1>	;GP3:	
7340	00002597 B414	<1>	MOV AH,HD320_SETTLE	; USE 320/360 HEAD SETTLE
7341		<1>	; JMP SHORT DO_WAT	
7342		<1>		



```

7343 <1> ;ISNT_WRITE:
7344 <1> ;OR AH,AH ; CHECK FOR NO WAIT
7345 <1> ;JZ short HW_DONE ; IF NOT WRITE AND 0 ITS OK
7346 <1>
7347 <1> ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
7348 <1> DO_WAT:
7349 <1> ;MOV AL,AH ; AL = # MILLISECONDS
7350 <1> ;;XOR AH,AH ; AX = # MILLISECONDS
7351 <1> J29: ; 1 MILLISECOND LOOP
7352 <1> ;mov cx, WAIT_FDU_HEAD_SETTLE ; 33 ; 1 ms in 30 micro units.
7353 00002599 B942000000 <1> MOV eCX,66 ; COUNT AT 15.085737 US PER COUNT
7354 0000259E E889F0FFFF <1> CALL WAITF ; DELAY FOR 1 MILLISECOND
7355 <1> ;DEC AL ; DECREMENT THE COUNT
7356 000025A3 FECC <1> dec ah
7357 000025A5 75F2 <1> JNZ short J29 ; DO AL MILLISECOND # OF TIMES
7358 <1> HW_DONE:
7359 000025A7 C3 <1> RETn
7360 <1>
7361 <1> ;-----
7362 <1> ; NEC_OUTPUT
7363 <1> ; THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
7364 <1> ; FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
7365 <1> ; TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
7366 <1> ; OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
7367 <1> ;
7368 <1> ; ON ENTRY: AH = BYTE TO BE OUTPUT
7369 <1> ;
7370 <1> ; ON EXIT: CY = 0 SUCCESS
7371 <1> ; CY = 1 FAILURE -- DISKETTE STATUS UPDATED
7372 <1> ; IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
7373 <1> ; HIGHER THAN THE CALLER OF NEC_OUTPUT. THIS REMOVES THE
7374 <1> ; REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
7375 <1> ; AX,CX,DX DESTROYED
7376 <1> ;-----
7377 <1>
7378 <1> ; 09/12/2014 [Erdogan Tan]
7379 <1> ; (from 'PS2 Hardware Interface Tech. Ref. May 88', Page 09-05.)
7380 <1> ; Diskette Drive Controller Status Register (3F4h)
7381 <1> ; This read only register facilitates the transfer of data between
7382 <1> ; the system microprocessor and the controller.
7383 <1> ; Bit 7 - When set to 1, the Data register is ready to transfer data
7384 <1> ; with the system micrprocessor.
7385 <1> ; Bit 6 - The direction of data transfer. If this bit is set to 0,
7386 <1> ; the transfer is to the controller.
7387 <1> ; Bit 5 - When this bit is set to 1, the controller is in the non-DMA mode.
7388 <1> ; Bit 4 - When this bit is set to 1, a Read or Write command is being executed.
7389 <1> ; Bit 3 - Reserved.
7390 <1> ; Bit 2 - Reserved.
7391 <1> ; Bit 1 - When this bit is set to 1, diskette drive 1 is in the seek mode.
7392 <1> ; Bit 0 - When this bit is set to 1, diskette drive 1 is in the seek mode.
7393 <1>
7394 <1> ; Data Register (3F5h)
7395 <1> ; This read/write register passes data, commands and parameters, and provides
7396 <1> ; diskette status information.
7397 <1>
7398 <1> NEC_OUTPUT:
7399 <1> ;PUSH BX ; SAVE REG.
7400 000025A8 66BAF403 <1> MOV DX,03F4H ; STATUS PORT
7401 <1> ;MOV BL,2 ; HIGH ORDER COUNTER
7402 <1> ;XOR CX,CX ; COUNT FOR TIME OUT
7403 <1> ; 16/12/2014
7404 <1> ; waiting for (max.) 0.5 seconds
7405 <1> ;;mov byte [wait_count], 0 ;; 27/02/2015
7406 <1> ;
7407 <1> ; 17/12/2014
7408 <1> ; Modified from AWARD BIOS 1999 - ADISK.ASM - SEND_COMMAND
7409 <1> ;
7410 <1> ;WAIT_FOR_PORT: Waits for a bit at a port pointed to by DX to
7411 <1> ; go on.
7412 <1> ;INPUT:
7413 <1> ; AH=Mask for isolation bits.
7414 <1> ; AL=pattern to look for.
7415 <1> ; DX=Port to test for
7416 <1> ; BH:CX=Number of memory refresh periods to delay.
7417 <1> ; (normally 30 microseconds per period.)
7418 <1> ;
7419 <1> ;WFP_SHORT:
7420 <1> ; Wait for port if refresh cycle is short (15-80 Us range).
7421 <1> ;
7422 <1>
7423 <1> ;mov bl, WAIT_FDU_SEND_HI+1 ; 0+1
7424 <1> ;mov cx, WAIT_FDU_SEND_LO ; 16667
7425 000025AC B91B410000 <1> mov ecx, WAIT_FDU_SEND_LH ; 16667 (27/02/2015)
7426 <1> ;
7427 <1> ;WFPS_OUTER_LP:
7428 <1> ;;
7429 <1> ;WFPS_CHECK_PORT:
7430 <1> J23:
7431 000025B1 EC <1> IN AL,DX ; GET STATUS
7432 000025B2 24C0 <1> AND AL,11000000B ; KEEP STATUS AND DIRECTION
7433 000025B4 3C80 <1> CMP AL,10000000B ; STATUS 1 AND DIRECTION 0 ?
7434 000025B6 7418 <1> JZ short J27 ; STATUS AND DIRECTION OK

```

7435		<1> WFPS_HI:
7436	000025B8 E461	<1> IN AL, PORT_B ;061h ; SYS1 ; wait for hi to lo
7437	000025BA A810	<1> TEST AL,010H ; transition on memory
7438	000025BC 75FA	<1> JNZ SHORT WFPS_HI ; refresh.
7439		<1> WFPS_LO:
7440	000025BE E461	<1> IN AL, PORT_B ; SYS1
7441	000025C0 A810	<1> TEST AL,010H
7442	000025C2 74FA	<1> JZ SHORT WFPS_LO
7443		<1> ;LOOP SHORT WFPS_CHECK_PORT
7444	000025C4 E2EB	<1> loop J23 ; 27/02/2015
7445		<1> ; ;
7446		<1> ;dec bl
7447		<1> ;jnz short WFPS_OUTER_LP
7448		<1> ;jmp short WFPS_TIMEOUT ; fail
7449		<1> ;J23:
7450		<1> ; IN AL,DX ; GET STATUS
7451		<1> ;AND AL,11000000B ; KEEP STATUS AND DIRECTION
7452		<1> ;CMP AL,10000000B ; STATUS 1 AND DIRECTION 0 ?
7453		<1> ;JZ short J27 ; STATUS AND DIRECTION OK
7454		<1> ;LOOP J23 ; CONTINUE TILL CX EXHAUSTED
7455		<1> ;DEC BL ; DECREMENT COUNTER
7456		<1> ;JNZ short J23 ; REPEAT TILL DELAY FINISHED, CX = 0
7457		<1>
7458		<1> ;;27/02/2015
7459		<1> ;16/12/2014
7460		<1> ;;cmp byte [wait_count], 10 ; (10/18.2 seconds)
7461		<1> ;;jb short J23
7462		<1>
7463		<1> ;WFPS_TIMEOUT:
7464		<1>
7465		<1> ;----- FALL THRU TO ERROR RETURN
7466		<1>
7467	000025C6 800D[2C710000]80	<1> OR byte [DSKETTE_STATUS],TIME_OUT
7468		<1> ;POP BX ; RESTORE REG.
7469	000025CD 58	<1> POP eAX ; 08/02/2015 ; DISCARD THE RETURN ADDRESS
7470	000025CE F9	<1> STC ; INDICATE ERROR TO CALLER
7471	000025CF C3	<1> RETn
7472		<1>
7473		<1> ;----- DIRECTION AND STATUS OK; OUTPUT BYTE
7474		<1>
7475		<1> J27:
7476	000025D0 88E0	<1> MOV AL,AH ; GET BYTE TO OUTPUT
7477	000025D2 6642	<1> INC DX ; DATA PORT = STATUS PORT + 1
7478	000025D4 EE	<1> OUT DX,AL ; OUTPUT THE BYTE
7479		<1> ;;NEWIODELAY ;; 27/02/2015
7480		<1> ; 27/02/2015
7481	000025D5 9C	<1> PUSHF ; SAVE FLAGS
7482	000025D6 B903000000	<1> MOV eCX, 3 ; 30 TO 45 MICROSECONDS WAIT FOR
7483	000025DB E84CF0FFFF	<1> CALL WAITF ; NEC FLAGS UPDATE CYCLE
7484	000025E0 9D	<1> POPF ; RESTORE FLAGS FOR EXIT
7485		<1> ;POP BX ; RESTORE REG
7486	000025E1 C3	<1> RETn ; CY = 0 FROM TEST INSTRUCTION
7487		<1>
7488		<1> ;-----
7489		<1> ; SEEK
7490		<1> ; THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE NAMED
7491		<1> ; TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE DRIVE
7492		<1> ; RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
7493		<1> ;
7494		<1> ; ON ENTRY: DI = DRIVE #
7495		<1> ; CH = TRACK #
7496		<1> ;
7497		<1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
7498		<1> ; AX,BX,CX DX DESTROYED
7499		<1> ;-----
7500		<1> SEEK:
7501	000025E2 89FB	<1> MOV eBX,eDI ; BX = DRIVE #
7502	000025E4 B001	<1> MOV AL,1 ; ESTABLISH MASK FOR RECALIBRATE TEST
7503	000025E6 86CB	<1> XCHG CL,BL ; SET DRIVE VALULE INTO CL
7504	000025E8 D2C0	<1> ROL AL,CL ; SHIFT MASK BY THE DRIVE VALUE
7505	000025EA 86CB	<1> XCHG CL,BL ; RECOVER TRACK VALUE
7506	000025EC 8405[29710000]	<1> TEST AL,[SEEK_STATUS] ; TEST FOR RECALIBRATE REQUIRED
7507	000025F2 7526	<1> JNZ short J28A ; JUMP IF RECALIBRATE NOT REQUIRED
7508		<1>
7509	000025F4 0805[29710000]	<1> OR [SEEK_STATUS],AL ; TURN ON THE NO RECALIBRATE BIT IN FLAG
7510	000025FA E862000000	<1> CALL RECAL ; RECALIBRATE DRIVE
7511	000025FF 730E	<1> JNC short AFT_RECAL ; RECALIBRATE DONE
7512		<1>
7513		<1> ;----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
7514		<1>
7515	00002601 C605[2C710000]00	<1> MOV byte [DSKETTE_STATUS],0 ; CLEAR OUT INVALID STATUS
7516	00002608 E854000000	<1> CALL RECAL ; RECALIBRATE DRIVE
7517	0000260D 7251	<1> JC short RB ; IF RECALIBRATE FAILS TWICE THEN ERROR
7518		<1>
7519		<1> AFT_RECAL:
7520	0000260F C687[3D710000]00	<1> MOV byte [DSK_TRK+eDI],0 ; SAVE NEW CYLINDER AS PRESENT POSITION
7521	00002616 08ED	<1> OR CH,CH ; CHECK FOR SEEK TO TRACK 0
7522	00002618 743F	<1> JZ short DO_WAIT ; HEAD SETTLE, CY = 0 IF JUMP
7523		<1>
7524		<1> ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
7525		<1>
7526	0000261A F687[39710000]20	<1> J28A: TEST byte [DSK_STATE+eDI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED

7527	00002621	7402	<1>	JZ	short _R7	; SINGLE STEP REQUIRED BYPASS DOUBLE
7528	00002623	D0E5	<1>	SHL	CH,1	; DOUBLE NUMBER OF STEP TO TAKE
7529			<1>			
7530	00002625	3AAF[3D710000]	<1>	_R7:	CMP CH, [DSK_TRK+eDI]	; SEE IF ALREADY AT THE DESIRED TRACK
7531	0000262B	7433	<1>	JE	short RB	; IF YES, DO NOT NEED TO SEEK
7532			<1>			
7533	0000262D	BA[60260000]	<1>	MOV	eDX, NEC_ERR	; LOAD RETURN ADDRESS
7534	00002632	52	<1>	PUSH	eDX ; (*)	; ON STACK FOR NEC OUTPUT ERROR
7535	00002633	88AF[3D710000]	<1>	MOV	[DSK_TRK+eDI],CH	; SAVE NEW CYLINDER AS PRESENT POSITION
7536	00002639	B40F	<1>	MOV	AH,0FH	; SEEK COMMAND TO NEC
7537	0000263B	E868FFFFFF	<1>	CALL	NEC_OUTPUT	
7538	00002640	89FB	<1>	MOV	eBX,eDI	; BX = DRIVE #
7539	00002642	88DC	<1>	MOV	AH,BL	; OUTPUT DRIVE NUMBER
7540	00002644	E85FFFFFFF	<1>	CALL	NEC_OUTPUT	
7541	00002649	8AA7[3D710000]	<1>	MOV	AH, [DSK_TRK+eDI]	; GET CYLINDER NUMBER
7542	0000264F	E854FFFFFF	<1>	CALL	NEC_OUTPUT	
7543	00002654	E829000000	<1>	CALL	CHK_STAT_2	; ENDING INTERRUPT AND SENSE STATUS
7544			<1>			
7545			<1>	;-----	WAIT FOR HEAD SETTLE	
7546			<1>			
7547			<1>	DO_WAIT:		
7548	00002659	9C	<1>	PUSHF		; SAVE STATUS
7549	0000265A	E816FFFFFF	<1>	CALL	HD_WAIT	; WAIT FOR HEAD SETTLE TIME
7550	0000265F	9D	<1>	POPF		; RESTORE STATUS
7551			<1>	RB:		
7552			<1>	NEC_ERR:		
7553			<1>		; 08/02/2015 (code trick here from original IBM PC/AT DISKETTE.ASM)	
7554			<1>		; (*) nec_err -> retn (push edx -> pop edx) -> nec_err -> retn	
7555	00002660	C3	<1>	RETN		; RETURN TO CALLER
7556			<1>			
7557			<1>		;-----	
7558			<1>		; RECAL	
7559			<1>		; RECALIBRATE DRIVE	
7560			<1>			
7561			<1>		; ON ENTRY: DI = DRIVE #	
7562			<1>			
7563			<1>		; ON EXIT: CY REFLECTS STATUS OF OPERATION.	
7564			<1>		;-----	
7565			<1>	RECAL:		
7566	00002661	6651	<1>	PUSH	CX	
7567	00002663	B8[7F260000]	<1>	MOV	eAX, RC_BACK	; LOAD NEC_OUTPUT ERROR
7568	00002668	50	<1>	PUSH	eAX	
7569	00002669	B407	<1>	MOV	AH,07H	; RECALIBRATE COMMAND
7570	0000266B	E838FFFFFF	<1>	CALL	NEC_OUTPUT	
7571	00002670	89FB	<1>	MOV	eBX,eDI	; BX = DRIVE #
7572	00002672	88DC	<1>	MOV	AH,BL	
7573	00002674	E82FFFFFFF	<1>	CALL	NEC_OUTPUT	; OUTPUT THE DRIVE NUMBER
7574	00002679	E804000000	<1>	CALL	CHK_STAT_2	; GET THE INTERRUPT AND SENSE INT STATUS
7575	0000267E	58	<1>	POP	eAX	; THROW AWAY ERROR
7576			<1>	RC_BACK:		
7577	0000267F	6659	<1>	POP	CX	
7578	00002681	C3	<1>	RETN		
7579			<1>			
7580			<1>		;-----	
7581			<1>		; CHK_STAT_2	
7582			<1>		; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,	
7583			<1>		; OR SEEK TO THE ADAPTER. THE INTERRUPT IS WAITED FOR, THE	
7584			<1>		; INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.	
7585			<1>			
7586			<1>		; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.	
7587			<1>		;-----	
7588			<1>	CHK_STAT_2:		
7589	00002682	B8[AA260000]	<1>	MOV	eAX, CS_BACK	; LOAD NEC_OUTPUT ERROR ADDRESS
7590	00002687	50	<1>	PUSH	eAX	
7591	00002688	E828000000	<1>	CALL	WAIT_INT	; WAIT FOR THE INTERRUPT
7592	0000268D	721A	<1>	JC	short J34	; IF ERROR, RETURN IT
7593	0000268F	B408	<1>	MOV	AH,08H	; SENSE INTERRUPT STATUS COMMAND
7594	00002691	E812FFFFFF	<1>	CALL	NEC_OUTPUT	
7595	00002696	E84A000000	<1>	CALL	RESULTS	; READ IN THE RESULTS
7596	0000269B	720C	<1>	JC	short J34	
7597	0000269D	A0[2D710000]	<1>	MOV	AL,[NEC_STATUS]	; GET THE FIRST STATUS BYTE
7598	000026A2	2460	<1>	AND	AL,01100000B	; ISOLATE THE BITS
7599	000026A4	3C60	<1>	CMP	AL,01100000B	; TEST FOR CORRECT VALUE
7600	000026A6	7403	<1>	JZ	short J35	; IF ERROR, GO MARK IT
7601	000026A8	F8	<1>	CLC		; GOOD RETURN
7602			<1>	J34:		
7603	000026A9	58	<1>	POP	eAX	; THROW AWAY ERROR RETURN
7604			<1>	CS_BACK:		
7605	000026AA	C3	<1>	RETN		
7606			<1>	J35:		
7607	000026AB	800D[2C710000]40	<1>	OR	byte [DSKETTE_STATUS], BAD_SEEK	
7608	000026B2	F9	<1>	STC		; ERROR RETURN CODE
7609	000026B3	EBF4	<1>	JMP	SHORT J34	
7610			<1>			
7611			<1>		;-----	
7612			<1>		; WAIT_INT	
7613			<1>		; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE	
7614			<1>		; TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED	
7615			<1>		; IF THE DRIVE IS NOT READY.	
7616			<1>			
7617			<1>		; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.	
7618			<1>		;-----	

```

7619 <1>
7620 <1> ; 17/12/2014
7621 <1> ; 2.5 seconds waiting !
7622 <1> ; (AWARD BIOS - 1999, WAIT_FDU_INT_LO, WAIT_FDU_INT_HI)
7623 <1> ; amount of time to wait for completion interrupt from NEC.
7624 <1>
7625 <1>
7626 <1> WAIT_INT:
7627 000026B5 FB <1> STI ; TURN ON INTERRUPTS, JUST IN CASE
7628 000026B6 F8 <1> CLC ; CLEAR TIMEOUT INDICATOR
7629 <1> ;MOV BL,10 ; CLEAR THE COUNTERS
7630 <1> ;XOR CX,CX ; FOR 2 SECOND WAIT
7631 <1>
7632 <1> ; Modification from AWARD BIOS - 1999 (ATORGS.ASM, WAIT
7633 <1> ;
7634 <1> ;WAIT_FOR_MEM:
7635 <1> ; Waits for a bit at a specified memory location pointed
7636 <1> ; to by ES:[DI] to become set.
7637 <1> ;INPUT:
7638 <1> ; AH=Mask to test with.
7639 <1> ; ES:[DI] = memory location to watch.
7640 <1> ; BH:CX=Number of memory refresh periods to delay.
7641 <1> ; (normally 30 microseconds per period.)
7642 <1>
7643 <1> ; waiting for (max.) 2.5 secs in 30 micro units.
7644 <1> ;mov cx, WAIT_FDU_INT_LO ; 017798
7645 <1> ;; mov bl, WAIT_FDU_INT_HI
7646 <1> ;mov bl, WAIT_FDU_INT_HI + 1
7647 <1> ; 27/02/2015
7648 000026B7 B986450100 <1> mov ecx, WAIT_FDU_INT_LH ; 83334 (2.5 seconds)
7649 <1> WFMS_CHECK_MEM:
7650 000026BC F605[29710000]80 <1> test byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
7651 000026C3 7516 <1> jnz short J37
7652 <1> WFMS_HI:
7653 000026C5 E461 <1> IN AL,PORT_B ; 061h ; SYS1, wait for lo to hi
7654 000026C7 A810 <1> TEST AL,010H ; transition on memory
7655 000026C9 75FA <1> JNZ SHORT WFMS_HI ; refresh.
7656 <1> WFMS_LO:
7657 000026CB E461 <1> IN AL,PORT_B ;SYS1
7658 000026CD A810 <1> TEST AL,010H
7659 000026CF 74FA <1> JZ SHORT WFMS_LO
7660 000026D1 E2E9 <1> LOOP WFMS_CHECK_MEM
7661 <1> ;WFMS_OUTER_LP:
7662 <1> ;; or bl, bl ; check outer counter
7663 <1> ;; jz short J36A ; WFMS_TIMEOUT
7664 <1> ;dec bl
7665 <1> ;jz short J36A
7666 <1> ;jmp short WFMS_CHECK_MEM
7667 <1>
7668 <1> ;17/12/2014
7669 <1> ;16/12/2014
7670 <1> ; mov byte [wait_count], 0 ; Reset (INT 08H) counter
7671 <1> ;J36:
7672 <1> ;TEST byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
7673 <1> ;JNZ short J37
7674 <1> ;16/12/2014
7675 <1> ;LOOP J36 ; COUNT DOWN WHILE WAITING
7676 <1> ;DEC BL ; SECOND LEVEL COUNTER
7677 <1> ;JNZ short J36
7678 <1> ; cmp byte [wait_count], 46 ; (46/18.2 seconds)
7679 <1> ;jb short J36
7680 <1>
7681 <1> ;WFMS_TIMEOUT:
7682 <1> ;J36A:
7683 000026D3 800D[2C710000]80 <1> OR byte [DSKETTE_STATUS], TIME_OUT ; NOTHING HAPPENED
7684 000026DA F9 <1> STC ; ERROR RETURN
7685 <1> J37:
7686 000026DB 9C <1> PUSHF ; SAVE CURRENT CARRY
7687 000026DC 8025[29710000]7F <1> AND byte [SEEK_STATUS], ~INT_FLAG ; TURN OFF INTERRUPT FLAG
7688 000026E3 9D <1> POPF ; RECOVER CARRY
7689 000026E4 C3 <1> RETn ; GOOD RETURN CODE
7690 <1>
7691 <1> ;-----
7692 <1> ; RESULTS
7693 <1> ; THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER RETURNS
7694 <1> ; FOLLOWING AN INTERRUPT.
7695 <1> ;
7696 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
7697 <1> ; AX,BX,CX,DX DESTROYED
7698 <1> ;-----
7699 <1> RESULTS:
7700 000026E5 57 <1> PUSH eDI
7701 000026E6 BF[2D710000] <1> MOV eDI, NEC_STATUS ; POINTER TO DATA AREA
7702 000026EB B307 <1> MOV BL,7 ; MAX STATUS BYTES
7703 000026ED 66BAF403 <1> MOV DX,03F4H ; STATUS PORT
7704 <1>
7705 <1> ;----- WAIT FOR REQUEST FOR MASTER
7706 <1>
7707 <1> _R10:
7708 <1> ; 16/12/2014
7709 <1> ; wait for (max) 0.5 seconds
7710 <1> ;MOV BH,2 ; HIGH ORDER COUNTER

```

```

7711      <1>      ;XOR  CX,CX                      ; COUNTER
7712      <1>
7713      <1>      ;Time to wait while waiting for each byte of NEC results = .5
7714      <1>      ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
7715      <1>      ; 27/02/2015
7716 000026F1 B91B410000      <1>      mov  ecx, WAIT_FDU_RESULTS_LH ; 16667
7717      <1>      ;mov  cx, WAIT_FDU_RESULTS_LO  ; 16667
7718      <1>      ;mov  bh, WAIT_FDU_RESULTS_HI+1 ; 0+1
7719      <1>
7720      <1>      WFPSR_OUTER_LP:
7721      <1>      ;
7722      <1>      WFPSR_CHECK_PORT:
7723      <1>      J39:                      ; WAIT FOR MASTER
7724 000026F6 EC      <1>      IN    AL,DX                      ; GET STATUS
7725 000026F7 24C0      <1>      AND    AL,11000000B          ; KEEP ONLY STATUS AND DIRECTION
7726 000026F9 3CC0      <1>      CMP    AL,11000000B          ; STATUS 1 AND DIRECTION 1 ?
7727 000026FB 7418      <1>      JZ     short J42              ; STATUS AND DIRECTION OK
7728      <1>      WFPSR_HI:
7729 000026FD E461      <1>      IN     AL, PORT_B ;061h ; SYS1      ; wait for hi to lo
7730 000026FF A810      <1>      TEST   AL,010H              ; transition on memory
7731 00002701 75FA      <1>      JNZ    SHORT WFPSR_HI          ; refresh.
7732      <1>      WFPSR_LO:
7733 00002703 E461      <1>      IN     AL, PORT_B          ; SYS1
7734 00002705 A810      <1>      TEST   AL,010H
7735 00002707 74FA      <1>      JZ     SHORT WFPSR_LO
7736 00002709 E2EB      <1>      LOOP    WFPSR_CHECK_PORT
7737      <1>      ;; 27/02/2015
7738      <1>      ;;dec bh
7739      <1>      ;;jnz short WFPSR_OUTER_LP
7740      <1>      ;jmp  short WFPSR_TIMEOUT      ; fail
7741      <1>
7742      <1>      ;;mov byte [wait_count], 0
7743      <1>      ;J39:                      ; WAIT FOR MASTER
7744      <1>      ; IN    AL,DX                      ; GET STATUS
7745      <1>      ;AND    AL,11000000B          ; KEEP ONLY STATUS AND DIRECTION
7746      <1>      ;CMP    AL,11000000B          ; STATUS 1 AND DIRECTION 1 ?
7747      <1>      ;JZ     short J42              ; STATUS AND DIRECTION OK
7748      <1>      ;LOOP J39                      ; LOOP TILL TIMEOUT
7749      <1>      ;DEC  BH                      ; DECREMENT HIGH ORDER COUNTER
7750      <1>      ;JNZ  short J39              ; REPEAT TILL DELAY DONE
7751      <1>      ;
7752      <1>      ;;cmp byte [wait_count], 10 ; (10/18.2 seconds)
7753      <1>      ;;jb  short J39
7754      <1>
7755      <1>      ;WFPSR_TIMEOUT:
7756 0000270B 800D[2C710000]80      <1>      OR     byte [DSKETTE_STATUS],TIME_OUT
7757 00002712 F9      <1>      STC                      ; SET ERROR RETURN
7758 00002713 EB29      <1>      JMP     SHORT POPRES          ; POP REGISTERS AND RETURN
7759      <1>
7760      <1>      ;----- READ IN THE STATUS
7761      <1>
7762      <1>      J42:
7763 00002715 EB00      <1>      JMP     $+2                      ; I/O DELAY
7764 00002717 6642      <1>      INC     DX                      ; POINT AT DATA PORT
7765 00002719 EC      <1>      IN     AL,DX                      ; GET THE DATA
7766      <1>      ; 16/12/2014
7767      <1>      NEWIODELAY
7768 0000271A E6EB      <2>      out 0ebh,al
7769 0000271C 8807      <1>      MOV     [eDI],AL          ; STORE THE BYTE
7770 0000271E 47      <1>      INC     eDI          ; INCREMENT THE POINTER
7771      <1>      ; 16/12/2014
7772      <1>      ;push  cx
7773      <1>      ;mov  cx, 30
7774      <1>      ;wdw2:
7775      <1>      ;NEWIODELAY
7776      <1>      ;loop  wdw2
7777      <1>      ;pop  cx
7778      <1>
7779 0000271F B903000000      <1>      MOV     eCX,3          ; MINIMUM 24 MICROSECONDS FOR NEC
7780 00002724 E803FFFFFF      <1>      CALL    WAITF          ; WAIT 30 TO 45 MICROSECONDS
7781 00002729 664A      <1>      DEC     DX                      ; POINT AT STATUS PORT
7782 0000272B EC      <1>      IN     AL,DX                      ; GET STATUS
7783      <1>      ; 16/12/2014
7784      <1>      NEWIODELAY
7785 0000272C E6EB      <2>      out 0ebh,al
7786      <1>      ;
7787 0000272E A810      <1>      TEST   AL,00010000B          ; TEST FOR NEC STILL BUSY
7788 00002730 740C      <1>      JZ     short POPRES          ; RESULTS DONE ?
7789      <1>
7790 00002732 FECB      <1>      DEC     BL          ; DECREMENT THE STATUS COUNTER
7791 00002734 75BB      <1>      JNZ     short _R10          ; GO BACK FOR MORE
7792 00002736 800D[2C710000]20      <1>      OR     byte [DSKETTE_STATUS],BAD_NEC ; TOO MANY STATUS BYTES
7793 0000273D F9      <1>      STC                      ; SET ERROR FLAG
7794      <1>
7795      <1>      ;----- RESULT OPERATION IS DONE
7796      <1>      POPRES:
7797 0000273E 5F      <1>      POP     eDI
7798 0000273F C3      <1>      RETn          ; RETURN WITH CARRY SET
7799      <1>
7800      <1>      ;-----
7801      <1>      ; READ_DSKCHNG
7802      <1>      ; READS THE STATE OF THE DISK CHANGE LINE.

```

```

7803 <1> ;
7804 <1> ; ON ENTRY: DI = DRIVE #
7805 <1> ;
7806 <1> ; ON EXIT: DI = DRIVE #
7807 <1> ; ZF = 0 : DISK CHANGE LINE INACTIVE
7808 <1> ; ZF = 1 : DISK CHANGE LINE ACTIVE
7809 <1> ; AX,CX,DX DESTROYED
7810 <1> ;-----
7811 <1> READ_DSKCHNG:
7812 00002740 E8A2FDEFFF <1> CALL MOTOR_ON ; TURN ON THE MOTOR IF OFF
7813 00002745 66BAF703 <1> MOV DX,03F7H ; ADDRESS DIGITAL INPUT REGISTER
7814 00002749 EC <1> IN AL,DX ; INPUT DIGITAL INPUT REGISTER
7815 0000274A A880 <1> TEST AL,DSK_CHG ; CHECK FOR DISK CHANGE LINE ACTIVE
7816 0000274C C3 <1> RETn ; RETURN TO CALLER WITH ZERO FLAG SET
7817 <1>
7818 <1> ;-----
7819 <1> ; DRIVE_DET
7820 <1> ; DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
7821 <1> ; UPDATES STATE INFORMATION ACCORDINGLY.
7822 <1> ; ON ENTRY: DI = DRIVE #
7823 <1> ;-----
7824 <1> DRIVE_DET:
7825 0000274D E895FDEFFF <1> CALL MOTOR_ON ; TURN ON MOTOR IF NOT ALREADY ON
7826 00002752 E80AFFFEFF <1> CALL RECAL ; RECALIBRATE DRIVE
7827 00002757 7251 <1> JC short DD_BAC ; ASSUME NO DRIVE PRESENT
7828 00002759 B530 <1> MOV CH,TRK_SLAP ; SEEK TO TRACK 48
7829 0000275B E882FEFFFF <1> CALL SEEK
7830 00002760 7248 <1> JC short DD_BAC ; ERROR NO DRIVE
7831 00002762 B50B <1> MOV CH,QUIET_SEEK+1 ; SEEK TO TRACK 10
7832 <1> SK_GIN:
7833 00002764 FECD <1> DEC CH ; DECREMENT TO NEXT TRACK
7834 00002766 6651 <1> PUSH CX ; SAVE TRACK
7835 00002768 E875FEFFFF <1> CALL SEEK
7836 0000276D 723C <1> JC short POP_BAC ; POP AND RETURN
7837 0000276F B8[AB270000] <1> MOV eAX, POP_BAC ; LOAD NEC OUTPUT ERROR ADDRESS
7838 00002774 50 <1> PUSH eAX
7839 00002775 B404 <1> MOV AH,SENSE_DRV_ST ; SENSE DRIVE STATUS COMMAND BYTE
7840 00002777 E82CFEFFFF <1> CALL NEC_OUTPUT ; OUTPUT TO NEC
7841 0000277C 6689F8 <1> MOV AX,DI ; AL = DRIVE
7842 0000277F 88C4 <1> MOV AH,AL ; AH = DRIVE
7843 00002781 E822FEFFFF <1> CALL NEC_OUTPUT ; OUTPUT TO NEC
7844 00002786 E85AFFFEFF <1> CALL RESULTS ; GO GET STATUS
7845 0000278B 58 <1> POP eAX ; THROW AWAY ERROR ADDRESS
7846 0000278C 6659 <1> POP CX ; RESTORE TRACK
7847 0000278E F605[2D710000]10 <1> TEST byte [NEC_STATUS], HOME ; TRACK 0 ?
7848 00002795 74CD <1> JZ short SK_GIN ; GO TILL TRACK 0
7849 00002797 08ED <1> OR CH,CH ; IS HOME AT TRACK 0
7850 00002799 7408 <1> JZ short IS_80 ; MUST BE 80 TRACK DRIVE
7851 <1>
7852 <1> ; DRIVE IS A 360; SET DRIVE TO DETERMINED;
7853 <1> ; SET MEDIA TO DETERMINED AT RATE 250.
7854 <1>
7855 0000279B 808F[39710000]94 <1> OR byte [DSK_STATE+eDI], DRV_DET+MED_DET+RATE_250
7856 000027A2 C3 <1> RETn ; ALL INFORMATION SET
7857 <1> IS_80:
7858 000027A3 808F[39710000]01 <1> OR byte [DSK_STATE+eDI], TRK_CAPA ; SETUP 80 TRACK CAPABILITY
7859 <1> DD_BAC:
7860 000027AA C3 <1> RETn
7861 <1> POP_BAC:
7862 000027AB 6659 <1> POP CX ; THROW AWAY
7863 000027AD C3 <1> RETn
7864 <1>
7865 <1> fdc_int:
7866 <1> ; 30/07/2015
7867 <1> ; 16/02/2015
7868 <1> ;int_0Eh: ; 11/12/2014
7869 <1>
7870 <1> ;--- HARDWARE INT 0EH -- ( IRQ LEVEL 6 ) -----
7871 <1> ; DISK_INT
7872 <1> ; THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
7873 <1> ;
7874 <1> ; ON EXIT: THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.
7875 <1> ;-----
7876 <1> DISK_INT_1:
7877 <1>
7878 000027AE 6650 <1> PUSH AX ; SAVE WORK REGISTER
7879 000027B0 1E <1> push ds
7880 000027B1 66B81000 <1> mov ax, KDATA
7881 000027B5 8ED8 <1> mov ds, ax
7882 000027B7 800D[29710000]80 <1> OR byte [SEEK_STATUS], INT_FLAG ; TURN ON INTERRUPT OCCURRED
7883 000027BE B020 <1> MOV AL,EOI ; END OF INTERRUPT MARKER
7884 000027C0 E620 <1> OUT INTA00,AL ; INTERRUPT CONTROL PORT
7885 000027C2 1F <1> pop ds
7886 000027C3 6658 <1> POP AX ; RECOVER REGISTER
7887 000027C5 CF <1> IRET ; RETURN FROM INTERRUPT
7888 <1>
7889 <1> ;-----
7890 <1> ; DSKETTE_SETUP
7891 <1> ; THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
7892 <1> ; DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
7893 <1> ;-----
7894 <1> DSKETTE_SETUP:

```

```

7895          <1> ;PUSH AX                      ; SAVE REGISTERS
7896          <1> ;PUSH BX
7897          <1> ;PUSH CX
7898 000027C6 52          <1> PUSH     eDX
7899          <1> ;PUSH DI
7900          <1> ;;PUSH      DS
7901          <1> ; 14/12/2014
7902          <1> ;mov  word [DISK_POINTER], MD_TBL6
7903          <1> ;mov  [DISK_POINTER+2], cs
7904          <1> ;
7905          <1> ;OR   byte [RTC_WAIT_FLAG], 1 ; NO RTC WAIT, FORCE USE OF LOOP
7906 000027C7 31FF          <1> XOR     eDI,eDI      ; INITIALIZE DRIVE POINTER
7907 000027C9 66C705[39710000]00- <1> MOV     WORD [DSK_STATE],0      ; INITIALIZE STATES
7908 000027D1 00          <1>
7909 000027D2 8025[34710000]33 <1> AND     byte [LASTRATE], ~(STRT_MSK+SEND_MSK) ; CLEAR START & SEND
7910 000027D9 800D[34710000]C0 <1> OR      byte [LASTRATE], SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
7911 000027E0 C605[29710000]00 <1> MOV     byte [SEEK_STATUS],0    ; INDICATE RECALIBRATE NEEDED
7912 000027E7 C605[2B710000]00 <1> MOV     byte [MOTOR_COUNT],0    ; INITIALIZE MOTOR COUNT
7913 000027EE C605[2A710000]00 <1> MOV     byte [MOTOR_STATUS],0   ; INITIALIZE DRIVES TO OFF STATE
7914 000027F5 C605[2C710000]00 <1> MOV     byte [DSKETTE_STATUS],0 ; NO ERRORS
7915          <1> ;
7916          <1> ; 28/02/2015
7917          <1> ;mov  word [cfd], 100h
7918 000027FC E860F2FFFF          <1> call    DSK_RESET
7919 00002801 5A          <1> pop     edx
7920 00002802 C3          <1> retn
7921          <1>
7922          <1> ;SUP0:
7923          <1> ;CALL  DRIVE_DET          ; DETERMINE DRIVE
7924          <1> ;CALL  XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
7925          <1> ;; 02/01/2015
7926          <1> ;;INC  DI              ; POINT TO NEXT DRIVE
7927          <1> ;;CMP  DI,MAX_DRV        ; SEE IF DONE
7928          <1> ;;JNZ  short SUP0        ; REPEAT FOR EACH ORIVE
7929          <1> ;      cmp     byte [fdl_type], 0
7930          <1> ;jna   short sup1
7931          <1> ;or    di, di
7932          <1> ;jnz  short sup1
7933          <1> ;inc   di
7934          <1> ;      jmp     short SUP0
7935          <1> ;sup1:
7936          <1> ;MOV   byte [SEEK_STATUS],0    ; FORCE RECALIBRATE
7937          <1> ;;AND  byte [RTC_WAIT_FLAG],0FEH ; ALLOW FOR RTC WAIT
7938          <1> ;CALL  SETUP_END          ; VARIOUS CLEANUPS
7939          <1> ;;;POP  DS              ; RESTORE CALLERS REGISTERS
7940          <1> ;;;POP  DI
7941          <1> ;POP   eDX
7942          <1> ;;;POP  CX
7943          <1> ;;;POP  BX
7944          <1> ;;;POP  AX
7945          <1> ;RETN
7946          <1>
7947          <1> ;////////////////////////////////////
7948          <1> ;; END OF DISKETTE I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
7949          <1> ;
7950          <1>
7951          <1> int13h: ; 21/02/2015
7952 00002803 9C          <1> pushfd
7953 00002804 0E          <1> push  cs
7954 00002805 E859000000 <1> call  DISK_IO
7955 0000280A C3          <1> retn
7956          <1>
7957          <1> ;;;; DISK I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; 21/02/2015 ;;;
7958          <1> ;////////////////////////////////////
7959          <1>
7960          <1> ; DISK I/O - Erdogan Tan (Retro UNIX 386 v1 project)
7961          <1> ; 23/02/2015
7962          <1> ; 21/02/2015 (unix386.s)
7963          <1> ; 22/12/2014 - 14/02/2015 (dsectrm2.s)
7964          <1> ;
7965          <1> ; Original Source Code:
7966          <1> ; DISK ----- 09/25/85 FIXED DISK BIOS
7967          <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
7968          <1> ;
7969          <1> ; Modifications: by reference of AWARD BIOS 1999 (D1A0622)
7970          <1> ;      Source Code - ATORGS.ASM, AHDSK.ASM
7971          <1> ;
7972          <1>
7973          <1>
7974          <1> ;The wait for controller to be not busy is 10 seconds.
7975          <1> ;10,000,000 / 30 = 333,333. 333,333 decimal = 051615h
7976          <1> ;;WAIT_HDU_CTLR_BUSY_LO equ 1615h
7977          <1> ;;WAIT_HDU_CTLR_BUSY_HI equ 05h
7978          <1> WAIT_HDU_CTRL_BUSY_LH equ 51615h ;21/02/2015
7979          <1>
7980          <1> ;The wait for controller to issue completion interrupt is 10 seconds.
7981          <1> ;10,000,000 / 30 = 333,333. 333,333 decimal = 051615h
7982          <1> ;;WAIT_HDU_INT_LO equ 1615h
7983          <1> ;;WAIT_HDU_INT_HI equ 05h
7984          <1> WAIT_HDU_INT_LH equ 51615h ; 21/02/2015
7985          <1>
7986          <1> ;The wait for Data request on read and write longs is

```

```

7987 <1> ;2000 us. (?)
7988 <1> ;;WAIT_HDU_DRQ_LO equ 1000 ; 03E8h
7989 <1> ;;WAIT_HDU_DRQ_HI equ 0
7990 <1> WAIT_HDU_DRQ_LH equ 1000 ; 21/02/2015
7991 <1>
7992 <1> ; Port 61h (PORT_B)
7993 <1> SYS1 equ 61h ; PORT_B (diskette.inc)
7994 <1>
7995 <1> ; 23/12/2014
7996 <1> %define CMD_BLOCK eBP-8 ; 21/02/2015
7997 <1>
7998 <1>
7999 <1> ;--- INT 13H -----
8000 <1> ; :
8001 <1> ; FIXED DISK I/O INTERFACE :
8002 <1> ; :
8003 <1> ; THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH :
8004 <1> ; THE IBM FIXED DISK CONTROLLER. :
8005 <1> ; :
8006 <1> ; THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH :
8007 <1> ; SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN :
8008 <1> ; THESE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS, :
8009 <1> ; NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE ANY :
8010 <1> ; ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENTS OF BIOS :
8011 <1> ; VIOLATE THE STRUCTURE AND DESIGN OF BIOS. :
8012 <1> ; :
8013 <1> ; -----:
8014 <1> ; :
8015 <1> ; INPUT (AH)= HEX COMMAND VALUE :
8016 <1> ; :
8017 <1> ; (AH)= 00H RESET DISK (DL = 80H,81H) / DISKETTE :
8018 <1> ; (AH)= 01H READ THE STATUS OF THE LAST DISK OPERATION INTO (AL) :
8019 <1> ; NOTE: DL < 80H - DISKETTE :
8020 <1> ; DL > 80H - DISK :
8021 <1> ; (AH)= 02H READ THE DESIRED SECTORS INTO MEMORY :
8022 <1> ; (AH)= 03H WRITE THE DESIRED SECTORS FROM MEMORY :
8023 <1> ; (AH)= 04H VERIFY THE DESIRED SECTORS :
8024 <1> ; (AH)= 05H FORMAT THE DESIRED TRACK :
8025 <1> ; (AH)= 06H UNUSED :
8026 <1> ; (AH)= 07H UNUSED :
8027 <1> ; (AH)= 08H RETURN THE CURRENT DRIVE PARAMETERS :
8028 <1> ; (AH)= 09H INITIALIZE DRIVE PAIR CHARACTERISTICS :
8029 <1> ; INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0 :
8030 <1> ; INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1 :
8031 <1> ; (AH)= 0AH READ LONG :
8032 <1> ; (AH)= 0BH WRITE LONG (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC) :
8033 <1> ; (AH)= 0CH SEEK :
8034 <1> ; (AH)= 0DH ALTERNATE DISK RESET (SEE DL) :
8035 <1> ; (AH)= 0EH UNUSED :
8036 <1> ; (AH)= 0FH UNUSED :
8037 <1> ; (AH)= 10H TEST DRIVE READY :
8038 <1> ; (AH)= 11H RECALIBRATE :
8039 <1> ; (AH)= 12H UNUSED :
8040 <1> ; (AH)= 13H UNUSED :
8041 <1> ; (AH)= 14H CONTROLLER INTERNAL DIAGNOSTIC :
8042 <1> ; (AH)= 15H READ DASD TYPE :
8043 <1> ; :
8044 <1> ; -----
8045 <1> ; :
8046 <1> ; REGISTERS USED FOR FIXED DISK OPERATIONS :
8047 <1> ; :
8048 <1> ; (DL) - DRIVE NUMBER (80H-81H FOR DISK. VALUE CHECKED) :
8049 <1> ; (DH) - HEAD NUMBER (0-15 ALLOWED, NOT VALUE CHECKED) :
8050 <1> ; (CH) - CYLINDER NUMBER (0-1023, NOT VALUE CHECKED) (SEE CL):
8051 <1> ; (CL) - SECTOR NUMBER (1-17, NOT VALUE CHECKED) :
8052 <1> ; :
8053 <1> ; NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED :
8054 <1> ; IN THE HIGH 2 BITS OF THE CL REGISTER :
8055 <1> ; (10 BITS TOTAL) :
8056 <1> ; :
8057 <1> ; (AL) - NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H, :
8058 <1> ; FOR READ/WRITE LONG 1-79H) :
8059 <1> ; :
8060 <1> ; (ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES, :
8061 <1> ; (NOT REQUIRED FOR VERIFY) :
8062 <1> ; :
8063 <1> ; FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST :
8064 <1> ; 2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:
8065 <1> ; F = 00H FOR A GOOD SECTOR :
8066 <1> ; 80H FOR A BAD SECTOR :
8067 <1> ; N = SECTOR NUMBER :
8068 <1> ; FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK :
8069 <1> ; THE TABLE SHOULD BE: :
8070 <1> ; :
8071 <1> ; DB 00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH :
8072 <1> ; DB 00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH :
8073 <1> ; DB 00H,07H,00H,10H,00H,08H,00H,11H,00H,09H :
8074 <1> ; :
8075 <1> ; -----
8076 <1>
8077 <1> ; -----
8078 <1> ; OUTPUT :

```



```

8079 <1> ; AH = STATUS OF CURRENT OPERATION :
8080 <1> ; STATUS BITS ARE DEFINED IN THE EQUATES BELOW :
8081 <1> ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN) :
8082 <1> ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON) :
8083 <1> ; :
8084 <1> ; NOTE: ERROR 11H INDICATES THAT THE DATA READ HAD A RECOVERABLE :
8085 <1> ; ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM. THE DATA :
8086 <1> ; IS PROBABLY GOOD, HOWEVER THE BIOS ROUTINE INDICATES AN :
8087 <1> ; ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE :
8088 <1> ; FOR ITSELF. THE ERROR MAY NOT RECUR IF THE DATA IS :
8089 <1> ; REWRITTEN. :
8090 <1> ; :
8091 <1> ; IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H), :
8092 <1> ; INPUT: :
8093 <1> ; (DL) = DRIVE NUMBER :
8094 <1> ; OUTPUT: :
8095 <1> ; (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2) :
8096 <1> ; (CONTROLLER CARD ZERO TALLY ONLY) :
8097 <1> ; (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER :
8098 <1> ; (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER :
8099 <1> ; (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER :
8100 <1> ; AND CYLINDER NUMBER HIGH BITS :
8101 <1> ; :
8102 <1> ; IF READ DASD TYPE WAS REQUESTED, :
8103 <1> ; :
8104 <1> ; AH = 0 - NOT PRESENT :
8105 <1> ; 1 - DISKETTE - NO CHANGE LINE AVAILABLE :
8106 <1> ; 2 - DISKETTE - CHANGE LINE AVAILABLE :
8107 <1> ; 3 - FIXED DISK :
8108 <1> ; :
8109 <1> ; CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3 :
8110 <1> ; :
8111 <1> ; REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN :
8112 <1> ; INFORMATION. :
8113 <1> ; :
8114 <1> ; NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE :
8115 <1> ; ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION. :
8116 <1> ; :
8117 <1> ; -----
8118 <1> ;
8119 <1> SENSE_FAIL EQU 0FFH ; NOT IMPLEMENTED
8120 <1> NO_ERR EQU 0E0H ; STATUS ERROR/ERROR REGISTER=0
8121 <1> WRITE_FAULT EQU 0CCH ; WRITE FAULT ON SELECTED DRIVE
8122 <1> UNDEF_ERR EQU 0BBH ; UNDEFINED ERROR OCCURRED
8123 <1> NOT_RDY EQU 0AAH ; DRIVE NOT READY
8124 <1> TIME_OUT EQU 80H ; ATTACHMENT FAILED TO RESPOND
8125 <1> BAD_SEEK EQU 40H ; SEEK OPERATION FAILED
8126 <1> BAD_CNTLRLR EQU 20H ; CONTROLLER HAS FAILED
8127 <1> DATA_CORRECTED EQU 11H ; ECC CORRECTED DATA ERROR
8128 <1> BAD_ECC EQU 10H ; BAD ECC ON DISK READ
8129 <1> BAD_TRACK EQU 0BH ; NOT IMPLEMENTED
8130 <1> BAD_SECTOR EQU 0AH ; BAD SECTOR FLAG DETECTED
8131 <1> ; DMA_BOUNDARY EQU 09H ; DATA EXTENDS TOO FAR
8132 <1> INIT_FAIL EQU 07H ; DRIVE PARAMETER ACTIVITY FAILED
8133 <1> BAD_RESET EQU 05H ; RESET FAILED
8134 <1> ; RECORD_NOT_FND EQU 04H ; REQUESTED SECTOR NOT FOUND
8135 <1> ; BAD_ADDR_MARK EQU 02H ; ADDRESS MARK NOT FOUND
8136 <1> ; BAD_CMD EQU 01H ; BAD COMMAND PASSED TO DISK I/O
8137 <1> ;
8138 <1> ; -----
8139 <1> ; :
8140 <1> ; FIXED DISK PARAMETER TABLE :
8141 <1> ; - THE TABLE IS COMPOSED OF A BLOCK DEFINED AS: :
8142 <1> ; :
8143 <1> ; +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS :
8144 <1> ; +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS :
8145 <1> ; +3 (1 WORD) - NOT USED/SEE PC-XT :
8146 <1> ; +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL :
8147 <1> ; +7 (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH :
8148 <1> ; +8 (1 BYTE) - CONTROL BYTE :
8149 <1> ; BIT 7 DISABLE RETRIES -OR- :
8150 <1> ; BIT 6 DISABLE RETRIES :
8151 <1> ; BIT 3 MORE THAN 8 HEADS :
8152 <1> ; +9 (3 BYTES) - NOT USED/SEE PC-XT :
8153 <1> ; +12 (1 WORD) - LANDING ZONE :
8154 <1> ; +14 (1 BYTE) - NUMBER OF SECTORS/TRACK :
8155 <1> ; +15 (1 BYTE) - RESERVED FOR FUTURE USE :
8156 <1> ; :
8157 <1> ; - TO DYNAMICALLY DEFINE A SET OF PARAMETERS :
8158 <1> ; BUILD A TABLE FOR UP TO 15 TYPES AND PLACE :
8159 <1> ; THE CORRESPONDING VECTOR INTO INTERRUPT 41 :
8160 <1> ; FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1. :
8161 <1> ; :
8162 <1> ; -----
8163 <1> ;
8164 <1> ; -----
8165 <1> ; :
8166 <1> ; HARDWARE SPECIFIC VALUES :
8167 <1> ; :
8168 <1> ; - CONTROLLER I/O PORT :
8169 <1> ; :
8170 <1> ; > WHEN READ FROM: :

```

```

8171 <1> ; HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU) :
8172 <1> ; HF_PORT+1 - GET ERROR REGISTER :
8173 <1> ; HF_PORT+2 - GET SECTOR COUNT :
8174 <1> ; HF_PORT+3 - GET SECTOR NUMBER :
8175 <1> ; HF_PORT+4 - GET CYLINDER LOW :
8176 <1> ; HF_PORT+5 - GET CYLINDER HIGH (2 BITS) :
8177 <1> ; HF_PORT+6 - GET SIZE/DRIVE/HEAD :
8178 <1> ; HF_PORT+7 - GET STATUS REGISTER :
8179 <1> ; :
8180 <1> ; > WHEN WRITTEN TO: :
8181 <1> ; HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :
8182 <1> ; HF_PORT+1 - SET PRECOMPENSATION CYLINDER :
8183 <1> ; HF_PORT+2 - SET SECTOR COUNT :
8184 <1> ; HF_PORT+3 - SET SECTOR NUMBER :
8185 <1> ; HF_PORT+4 - SET CYLINDER LOW :
8186 <1> ; HF_PORT+5 - SET CYLINDER HIGH (2 BITS) :
8187 <1> ; HF_PORT+6 - SET SIZE/DRIVE/HEAD :
8188 <1> ; HF_PORT+7 - SET COMMAND REGISTER :
8189 <1> ; :
8190 <1> ; -----
8191 <1>
8192 <1> ; HF_PORT EQU 01F0H ; DISK PORT
8193 <1> ; HF1_PORT equ 0170h
8194 <1> ; HF_REG_PORT EQU 03F6H
8195 <1> ; HF1_REG_PORT equ 0376h
8196 <1>
8197 <1> HDC1_BASEPORTEqu 1F0h
8198 <1> HDC2_BASEPORTEqu 170h
8199 <1>
8200 0000280B 90 <1> align 2
8201 <1>
8202 <1> ;----- STATUS REGISTER
8203 <1>
8204 <1> ST_ERROR EQU 00000001B ;
8205 <1> ST_INDEX EQU 00000010B ;
8206 <1> ST_CORRCTD EQU 00000100B ; ECC CORRECTION SUCCESSFUL
8207 <1> ST_DRQ EQU 00001000B ;
8208 <1> ST_SEEK_COMPLEQU 00010000B ; SEEK COMPLETE
8209 <1> ST_WRT_FLT EQU 00100000B ; WRITE FAULT
8210 <1> ST_READY EQU 01000000B ;
8211 <1> ST_BUSY EQU 10000000B ;
8212 <1>
8213 <1> ;----- ERROR REGISTER
8214 <1>
8215 <1> ERR_DAM EQU 00000001B ; DATA ADDRESS MARK NOT FOUND
8216 <1> ERR_TRK_0 EQU 00000010B ; TRACK 0 NOT FOUND ON RECAL
8217 <1> ERR_ABORT EQU 00000100B ; ABORTED COMMAND
8218 <1> ; EQU 00001000B ; NOT USED
8219 <1> ERR_ID EQU 00010000B ; ID NOT FOUND
8220 <1> ; EQU 00100000B ; NOT USED
8221 <1> ERR_DATA_ECC EQU 01000000B
8222 <1> ERR_BAD_BLOCKEQU 10000000B
8223 <1>
8224 <1>
8225 <1> RECAL_CMD EQU 00010000B ; DRIVE RECAL (10H)
8226 <1> READ_CMD EQU 00100000B ; READ (20H)
8227 <1> WRITE_CMD EQU 00110000B ; WRITE (30H)
8228 <1> VERIFY_CMD EQU 01000000B ; VERIFY (40H)
8229 <1> FMTTRK_CMD EQU 01010000B ; FORMAT TRACK (50H)
8230 <1> INIT_CMD EQU 01100000B ; INITIALIZE (60H)
8231 <1> SEEK_CMD EQU 01110000B ; SEEK (70H)
8232 <1> DIAG_CMD EQU 10010000B ; DIAGNOSTIC (90H)
8233 <1> SET_PARM_CMD EQU 10010001B ; DRIVE PARMS (91H)
8234 <1> NO_RETRIES EQU 00000001B ; CHD MODIFIER (01H)
8235 <1> ECC_MODE EQU 00000010B ; CMD MODIFIER (02H)
8236 <1> BUFFER_MODE EQU 00001000B ; CMD MODIFIER (08H)
8237 <1>
8238 <1> ; MAX_FILE EQU 2
8239 <1> ; S_MAX_FILE EQU 2
8240 <1> MAX_FILE equ 4 ; 22/12/2014
8241 <1> S_MAX_FILE equ 4 ; 22/12/2014
8242 <1>
8243 <1> DELAY_1 EQU 25H ; DELAY FOR OPERATION COMPLETE
8244 <1> DELAY_2 EQU 0600H ; DELAY FOR READY
8245 <1> DELAY_3 EQU 0100H ; DELAY FOR DATA REQUEST
8246 <1>
8247 <1> HF_FAIL EQU 08H ; CMOS FLAG IN BYTE 0EH
8248 <1>
8249 <1> ;----- COMMAND BLOCK REFERENCE
8250 <1>
8251 <1> ; CMD_BLOCK EQU BP-8 ; @CMD_BLOCK REFERENCES BLOCK HEAD IN SS
8252 <1> ; (BP) POINTS TO COMMAND BLOCK TAIL
8253 <1> ; AS DEFINED BY THE "ENTER" PARMS
8254 <1> ; 19/12/2014
8255 <1> ORG_VECTOR equ 4*13h ; INT 13h vector
8256 <1> DISK_VECTOR equ 4*40h ; INT 40h vector (for floppy disks)
8257 <1> ; HDISK_INT equ 4*76h ; Primary HDC - Hardware interrupt (IRQ14)
8258 <1> ; HDISK_INT1 equ 4*76h ; Primary HDC - Hardware interrupt (IRQ14)
8259 <1> ; HDISK_INT2 equ 4*77h ; Secondary HDC - Hardware interrupt (IRQ15)
8260 <1> ; HF_TBL_VEC equ 4*41h ; Pointer to 1st fixed disk parameter table
8261 <1> ; HF1_TBL_VEC equ 4*46h ; Pointer to 2nd fixed disk parameter table
8262 <1>

```

```

8263 <1> align 2
8264 <1>
8265 <1> ;-----
8266 <1> ; FIXED DISK I/O SETUP :
8267 <1> ; :
8268 <1> ; - ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK :
8269 <1> ; - PERFORM POWER ON DIAGNOSTICS :
8270 <1> ; SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED :
8271 <1> ; :
8272 <1> ;-----
8273 <1>
8274 <1> DISK_SETUP:
8275 <1> ;CLI
8276 <1> ;;MOV AX,ABS0 ; GET ABSOLUTE SEGMENT
8277 <1> ;xor ax,ax
8278 <1> ;MOV DS,AX ; SET SEGMENT REGISTER
8279 <1> ;MOV AX, [ORG_VECTOR] ; GET DISKETTE VECTOR
8280 <1> ;MOV [DISK_VECTOR],AX ; INTO INT 40H
8281 <1> ;MOV AX, [ORG_VECTOR+2]
8282 <1> ;MOV [DISK_VECTOR+2],AX
8283 <1> ;MOV word [ORG_VECTOR],DISK_IO ; FIXED DISK HANDLER
8284 <1> ;MOV [ORG_VECTOR+2],CS
8285 <1> ; 1st controller (primary master, slave) - IRQ 14
8286 <1> ;;MOV word [HDISK_INT],HD_INT ; FIXED DISK INTERRUPT
8287 <1> ;mov word [HDISK_INT1],HD_INT ;
8288 <1> ;;MOV [HDISK_INT+2],CS
8289 <1> ;mov [HDISK_INT1+2],CS
8290 <1> ; 2nd controller (secondary master, slave) - IRQ 15
8291 <1> ;mov word [HDISK_INT2],HD1_INT ;
8292 <1> ;mov [HDISK_INT2+2],CS
8293 <1> ;
8294 <1> ;;MOV word [HF_TBL_VEC],HD0_DPT ; PARM TABLE DRIVE 80
8295 <1> ;;MOV word [HF_TBL_VEC+2],DPT_SEGM
8296 <1> ;;MOV word [HF1_TBL_VEC],HD1_DPT ; PARM TABLE DRIVE 81
8297 <1> ;;MOV word [HF1_TBL_VEC+2],DPT_SEGM
8298 <1> ;push cs
8299 <1> ;pop ds
8300 <1> ;mov word [HDFM_TBL_VEC],HD0_DPT ; PARM TABLE DRIVE 80h
8301 <1> ;mov word [HDFM_TBL_VEC+2],DPT_SEGM
8302 0000280C C705[44710000]0000-<1> mov dword [HDFM_TBL_VEC], (DPT_SEGM*16)+HD0_DPT
8303 00002814 0900 <1>
8304 <1> ;mov word [HDPS_TBL_VEC],HD1_DPT ; PARM TABLE DRIVE 81h
8305 <1> ;mov word [HDPS_TBL_VEC+2],DPT_SEGM
8306 00002816 C705[48710000]2000-<1> mov dword [HDPS_TBL_VEC], (DPT_SEGM*16)+HD1_DPT
8307 0000281E 0900 <1>
8308 <1> ;mov word [HDSM_TBL_VEC],HD2_DPT ; PARM TABLE DRIVE 82h
8309 <1> ;mov word [HDSM_TBL_VEC+2],DPT_SEGM
8310 00002820 C705[4C710000]4000-<1> mov dword [HDSM_TBL_VEC], (DPT_SEGM*16)+HD2_DPT
8311 00002828 0900 <1>
8312 <1> ;mov word [HDSS_TBL_VEC],HD3_DPT ; PARM TABLE DRIVE 83h
8313 <1> ;mov word [HDSS_TBL_VEC+2],DPT_SEGM
8314 0000282A C705[50710000]6000-<1> mov dword [HDSS_TBL_VEC], (DPT_SEGM*16)+HD3_DPT
8315 00002832 0900 <1>
8316 <1> ;
8317 <1> ;;IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
8318 <1> ;;AND AL,0BFH
8319 <1> ;;and al, 3Fh ; enable IRQ 14 and IRQ 15
8320 <1> ;;JMP $+2
8321 <1> ;;IODELAY
8322 <1> ;;OUT INTB01,AL
8323 <1> ;;IODELAY
8324 <1> ;;IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
8325 <1> ;;AND AL,0FBH ; SECOND CHIP
8326 <1> ;;JMP $+2
8327 <1> ;;IODELAY
8328 <1> ;;OUT INTA01,AL
8329 <1> ;
8330 <1> ;STI
8331 <1> ;;PUSH DS ; MOVE ABS0 POINTER TO
8332 <1> ;;POP ES ; EXTRA SEGMENT POINTER
8333 <1> ;;CALL DDS ; ESTABLISH DATA SEGMENT
8334 <1> ;;MOV byte [DISK_STATUS1],0 ; RESET THE STATUS INDICATOR
8335 <1> ;;MOV byte [HF_NUM],0 ; ZERO NUMBER OF FIXED DISKS
8336 <1> ;;MOV byte [CONTROL_BYTE],0
8337 <1> ;;MOV byte [PORT_OFF],0 ; ZERO CARD OFFSET
8338 <1> ; 20/12/2014 - private code by Erdogan Tan
8339 <1> ; (out of original PC-AT, PC-XT BIOS code)
8340 <1> ;mov si, hd0_type
8341 00002834 BE[DA6D0000] <1> mov esi, hd0_type
8342 <1> ;mov cx, 4
8343 00002839 B904000000 <1> mov ecx, 4
8344 <1> hde_1:
8345 0000283E AC <1> lodsb
8346 0000283F 3C80 <1> cmp al, 80h ; 8?h = existing
8347 00002841 7206 <1> jnb short _L4
8348 00002843 FE05[40710000] <1> inc byte [HF_NUM] ; + 1 hard (fixed) disk drives
8349 <1> _L4: ; 26/02/2015
8350 00002849 E2F3 <1> loop hde_1
8351 <1> ;_L4: ; 0 <= [HF_NUM] =< 4
8352 <1> ;L4:
8353 <1> ;
8354 <1> ;; 31/12/2014 - cancel controller diagnostics here

```

```

8355 <1> ;;;mov cx, 3 ; 26/12/2014 (Award BIOS 1999)
8356 <1> ;;mov cl, 3
8357 <1> ;;
8358 <1> ;;MOV DL,80H ; CHECK THE CONTROLLER
8359 <1> ;;hdc_dl:
8360 <1> ;;MOV AH,14H ; USE CONTROLLER DIAGNOSTIC COMMAND
8361 <1> ;;INT 13H ; CALL BIOS WITH DIAGNOSTIC COMMAND
8362 <1> ;;;JC short CTL_ERRX ; DISPLAY ERROR MESSAGE IF BAD RETURN
8363 <1> ;;;jc short POD_DONE ;22/12/2014
8364 <1> ;;jnc short hdc_reset0
8365 <1> ;;loop hdc_dl
8366 <1> ;;; 27/12/2014
8367 <1> ;;stc
8368 <1> ;;retn
8369 <1> ;
8370 <1> ;;hdc_reset0:
8371 <1> ; 18/01/2015
8372 0000284B 8A0D[40710000] <1> mov cl, [HF_NUM]
8373 00002851 20C9 <1> and cl, cl
8374 00002853 740D <1> jz short POD_DONE
8375 <1> ;
8376 00002855 B27F <1> mov dl, 7Fh
8377 <1> hdc_reset1:
8378 00002857 FEC2 <1> inc dl
8379 <1> ;; 31/12/2015
8380 <1> ;;push dx
8381 <1> ;;push cx
8382 <1> ;;push ds
8383 <1> ;;sub ax, ax
8384 <1> ;;mov ds, ax
8385 <1> ;;MOV AX, [TIMER_LOW] ; GET START TIMER COUNTS
8386 <1> ;;pop ds
8387 <1> ;;MOV BX,AX
8388 <1> ;;ADD AX,6*182 ; 60 SECONDS* 18.2
8389 <1> ;;MOV CX,AX
8390 <1> ;;mov word [wait_count], 0 ; 22/12/2014 (reset wait counter)
8391 <1> ;;
8392 <1> ;; 31/12/2014 - cancel HD_RESET_1
8393 <1> ;;CALL HD_RESET_1 ; SET UP DRIVE 0, (1,2,3)
8394 <1> ;;pop cx
8395 <1> ;;pop dx
8396 <1> ;;
8397 <1> ; 18/01/2015
8398 00002859 B40D <1> mov ah, 0Dh ; ALTERNATE RESET
8399 <1> ;int 13h
8400 0000285B E8A3FFFFFF <1> call int13h
8401 00002860 E2F5 <1> loop hdc_reset1
8402 <1> POD_DONE:
8403 00002862 C3 <1> RETn
8404 <1>
8405 <1> ;;-----POD_ERROR
8406 <1>
8407 <1> ;;CTL_ERRX:
8408 <1> ;;MOV SI,OFFSET F1782 ; CONTROLLER ERROR
8409 <1> ;;CALL SET_FAIL ; DO NOT IPL FROM DISK
8410 <1> ;;CALL E_MSG ; DISPLAY ERROR AND SET (BP) ERROR FLAG
8411 <1> ;;JMP short POD_DONE
8412 <1>
8413 <1> ;;HD_RESET_1:
8414 <1> ;; ;PUSH BX ; SAVE TIMER LIMITS
8415 <1> ;; ;PUSH CX
8416 <1> ;;RES_1: MOV AH,09H ; SET DRIVE PARAMETERS
8417 <1> ;; INT 13H
8418 <1> ;; JC short RES_2
8419 <1> ;; MOV AH,11H ; RECALIBRATE DRIVE
8420 <1> ;; INT 13H
8421 <1> ;; JNC short RES_CHK ; DRIVE OK
8422 <1> ;;RES_2: ;CALL POD_TCHK ; CHECK TIME OUT
8423 <1> ;; cmp word [wait_count], 6*182 ; waiting time (in timer ticks)
8424 <1> ;; ; (30 seconds)
8425 <1> ;; ;cmc
8426 <1> ;; ;JNC short RES_1
8427 <1> ;; ;jb short RES_1
8428 <1> ;;;RES_FL: ;MOV SI,OFFSET F1781 ; INDICATE DISK 1 FAILURE;
8429 <1> ;; ;TEST DL,1
8430 <1> ;; ;JNZ RES_E1
8431 <1> ;; ;MOV SI,OFFSET F1780 ; INDICATE DISK 0 FAILURE
8432 <1> ;; ;CALL SET_FAIL ; DO NOT TRY TO IPL DISK 0
8433 <1> ;; ;JMP SHORT RES_E1
8434 <1> ;;RES_ER: ; 22/12/2014
8435 <1> ;;RES_OK:
8436 <1> ;; ;POP CX ; RESTORE TIMER LIMITS
8437 <1> ;; ;POP BX
8438 <1> ;; ;RETN
8439 <1> ;;
8440 <1> ;;RES_RS: MOV AH,00H ; RESET THE DRIVE
8441 <1> ;; INT 13H
8442 <1> ;;RES_CHK: MOV AH,08H ; GET MAX CYLINDER,HEAD,SECTOR
8443 <1> ;; MOV BL,DL ; SAVE DRIVE CODE
8444 <1> ;; INT 13H
8445 <1> ;; JC short RES_ER
8446 <1> ;; MOV [NEC_STATUS],CX ; SAVE MAX CYLINDER, SECTOR

```

```

8447 <1> ;; MOV DL,BL ; RESTORE DRIVE CODE
8448 <1> ;;RES_3: MOV AX,0401H ; VERIFY THE LAST SECTOR
8449 <1> ;; INT 13H
8450 <1> ;; JNC short RES_OK ; VERIFY OK
8451 <1> ;; CMP AH,BAD_SECTOR ; OK ALSO IF JUST ID READ
8452 <1> ;; JE short RES_OK
8453 <1> ;; CMP AH,DATA_CORRECTED
8454 <1> ;; JE short RES_OK
8455 <1> ;; CMP AH,BAD_ECC
8456 <1> ;; JE short RES_OK
8457 <1> ;; ;CALL POD_TCHK ; CHECK FOR TIME OUT
8458 <1> ;; cmp word [wait_count], 6*182 ; waiting time (in timer ticks)
8459 <1> ;; ; (60 seconds)
8460 <1> ;; cmc
8461 <1> ;; JC short RES_ER ; FAILED
8462 <1> ;; MOV CX,[NEC_STATUS] ; GET SECTOR ADDRESS, AND CYLINDER
8463 <1> ;; MOV AL,CL ; SEPARATE OUT SECTOR NUMBER
8464 <1> ;; AND AL,3FH
8465 <1> ;; DEC AL ; TRY PREVIOUS ONE
8466 <1> ;; JZ short RES_RS ; WE'VE TRIED ALL SECTORS ON TRACK
8467 <1> ;; AND CL,0C0H ; KEEP CYLINDER BITS
8468 <1> ;; OR CL,AL ; MERGE SECTOR WITH CYLINDER BITS
8469 <1> ;; MOV [NEC_STATUS],CX ; SAVE CYLINDER, NEW SECTOR NUMBER
8470 <1> ;; JMP short RES_3 ; TRY AGAIN
8471 <1> ;;RES_ER: MOV SI,OFFSET F1791 ; INDICATE DISK 1 ERROR
8472 <1> ;; ;TEST DL,1
8473 <1> ;; ;JNZ short RES_E1
8474 <1> ;; ;MOV SI,OFFSET F1790 ; INDICATE DISK 0 ERROR
8475 <1> ;;RES_E1:
8476 <1> ;; ;CALL E_MSG ; DISPLAY ERROR AND SET (BP) ERROR FLAG
8477 <1> ;;RES_OK:
8478 <1> ;; ;POP CX ; RESTORE TIMER LIMITS
8479 <1> ;; ;POP BX
8480 <1> ;; ;RETN
8481 <1> ;
8482 <1> ;;SET_FAIL:
8483 <1> ;;MOV AX,X*(CMOS_DIAG+NMI) ; GET CMOS ERROR BYTE
8484 <1> ;;CALL CMOS_READ
8485 <1> ;;OR AL,HF_FAIL ; SET DO NOT IPL FROM DISK FLAG
8486 <1> ;;XCHG AH,AL ; SAVE IT
8487 <1> ;;CALL CMOS_WRITE ; PUT IT OUT
8488 <1> ;;RETN
8489 <1> ;
8490 <1> ;;POD_TCHK: ; CHECK FOR 30 SECOND TIME OUT
8491 <1> ;;POP AX ; SAVE RETURN
8492 <1> ;;POP CX ; GET TIME OUT LIMITS
8493 <1> ;;POP BX
8494 <1> ;;PUSH BX ; AND SAVE THEM AGAIN
8495 <1> ;;PUSH CX
8496 <1> ;;PUSH AX
8497 <1> ;;push ds
8498 <1> ;;xor ax, ax
8499 <1> ;;mov ds, ax ; RESTORE RETURN
8500 <1> ;;MOV AX, [TIMER_LOW] ; AX = CURRENT TIME
8501 <1> ;; ; BX = START TIME
8502 <1> ;; ; CX = END TIME
8503 <1> ;;pop ds
8504 <1> ;;CMP BX,CX
8505 <1> ;;JB short TCHK1 ; START < END
8506 <1> ;;CMP BX,AX
8507 <1> ;;JB short TCHKG ; END < START < CURRENT
8508 <1> ;;JMP SHORT TCHK2 ; END, CURRENT < START
8509 <1> ;;TCHK1: CMP AX,BX
8510 <1> ;; JB short TCHKNG ; CURRENT < START < END
8511 <1> ;;TCHK2: CMP AX,CX
8512 <1> ;; JB short TCHKG ; START < CURRENT < END
8513 <1> ;; ; OR CURRENT < END < START
8514 <1> ;;TCHKNG: STC ; CARRY SET INDICATES TIME OUT
8515 <1> ;; RETN
8516 <1> ;;TCHKG: CLC ; INDICATE STILL TIME
8517 <1> ;; RETN
8518 <1> ;;
8519 <1> ;;int_13h:
8520 <1>
8521 <1> ;-----
8522 <1> ; FIXED DISK BIOS ENTRY POINT :
8523 <1> ;-----
8524 <1>
8525 <1> DISK_IO:
8526 00002863 80FA80 <1> CMP DL,80H ; TEST FOR FIXED DISK DRIVE
8527 <1> ;JAE short A1 ; YES, HANDLE HERE
8528 <1> ;;;INT 40H ; DISKETTE HANDLER
8529 <1> ;;call int40h
8530 00002866 0F8225F1FFFF <1> jnb DISKETTE_IO_1
8531 <1> ;RET_2:
8532 <1> ;RETF 2 ; BACK TO CALLER
8533 <1> ;retf 4
8534 <1> A1:
8535 0000286C FB <1> STI ; ENABLE INTERRUPTS
8536 <1> ;; 04/01/2015
8537 <1> ;;OR AH,AH
8538 <1> ;;JNZ short A2

```

8539		<1>	;;INT 40H	; RESET NEC WHEN AH=0
8540		<1>	;;SUB AH,AH	
8541	0000286D 80FA83	<1>	CMP DL,(80H + S_MAX_FILE - 1)	
8542	00002870 772C	<1>	JA short RET_2	
8543		<1>	; 18/01/2015	
8544	00002872 08E4	<1>	or ah,ah	
8545	00002874 742B	<1>	jz short A4	
8546	00002876 80FC0D	<1>	cmp ah, 0Dh	; Alternate reset
8547	00002879 7504	<1>	jne short A2	
8548	0000287B 28E4	<1>	sub ah,ah ; Reset	
8549	0000287D EB22	<1>	jmp short A4	
8550		<1>	A2:	
8551	0000287F 80FC08	<1>	CMP AH,08H	; GET PARAMETERS IS A SPECIAL CASE
8552		<1>	;JNZ short A3	
8553		<1>	;JMP GET_PARM_N	
8554	00002882 0F841C030000	<1>	je GET_PARM_N	
8555	00002888 80FC15	<1>	A3: CMP AH,15H	; READ DASD TYPE IS ALSO
8556		<1>	;JNZ short A4	
8557		<1>	;JMP READ_DASD_TYPE	
8558	0000288B 0F84C7020000	<1>	je READ_DASD_TYPE	
8559		<1>	; 02/02/2015	
8560	00002891 80FC1D	<1>	cmp ah, 1Dh	; (Temporary for Retro UNIX 386 v1)
8561		<1>	; 12/01/2015	
8562	00002894 F5	<1>	cmc	
8563	00002895 730A	<1>	jnc short A4	
8564		<1>	; 30/01/2015	
8565		<1>	;mov byte [CS:DISK_STATUS1],BAD_CMD ; COMMAND ERROR	
8566	00002897 C605[3F710000]01	<1>	mov byte [DISK_STATUS1], BAD_CMD	
8567		<1>	;jmp short RET_2	
8568		<1>	RET_2:	
8569	0000289E CA0400	<1>	retf 4	
8570		<1>	A4:	; SAVE REGISTERS DURING OPERATION
8571	000028A1 C8080000	<1>	ENTER 8,0	; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
8572	000028A5 53	<1>	PUSH eBX	; IN THE STACK, THE COMMAND BLOCK IS:
8573	000028A6 51	<1>	PUSH eCX	; @CMD_BLOCK == BYTE PTR [BP]-8
8574	000028A7 52	<1>	PUSH eDX	
8575	000028A8 1E	<1>	PUSH DS	
8576	000028A9 06	<1>	PUSH ES	
8577	000028AA 56	<1>	PUSH eSI	
8578	000028AB 57	<1>	PUSH eDI	
8579		<1>	;;04/01/2015	
8580		<1>	;;OR AH,AH	; CHECK FOR RESET
8581		<1>	;;JNZ short A5	
8582		<1>	;;MOV DL,80H	; FORCE DRIVE 80 FOR RESET
8583		<1>	;;A5:	
8584		<1>	;push cs	
8585		<1>	;pop ds	
8586		<1>	; 21/02/2015	
8587	000028AC 6650	<1>	push ax	
8588	000028AE 66B81000	<1>	mov ax, KDATA	
8589	000028B2 8ED8	<1>	mov ds, ax	
8590	000028B4 8EC0	<1>	mov es, ax	
8591	000028B6 6658	<1>	pop ax	
8592	000028B8 E889000000	<1>	CALL DISK_IO_CONT	; PERFORM THE OPERATION
8593		<1>	;;CALL DDS	; ESTABLISH SEGMENT
8594	000028BD 8A25[3F710000]	<1>	MOV AH,[DISK_STATUS1]	; GET STATUS FROM OPERATION
8595	000028C3 80FC01	<1>	CMP AH,1	; SET THE CARRY FLAG TO INDICATE
8596	000028C6 F5	<1>	CMC	; SUCCESS OR FAILURE
8597	000028C7 5F	<1>	POP eDI	; RESTORE REGISTERS
8598	000028C8 5E	<1>	POP eSI	
8599	000028C9 07	<1>	POP ES	
8600	000028CA 1F	<1>	POP DS	
8601	000028CB 5A	<1>	POP eDX	
8602	000028CC 59	<1>	POP eCX	
8603	000028CD 5B	<1>	POP eBX	
8604	000028CE C9	<1>	LEAVE	; ADJUST (SP) AND RESTORE (BP)
8605		<1>	;RETf 2	; THROW AWAY SAVED FLAGS
8606	000028CF CA0400	<1>	retf 4	
8607		<1>	; 21/02/2015	
8608		<1>	; dw --> dd	
8609		<1>	M1:	; FUNCTION TRANSFER TABLE
8610	000028D2 [942A0000]	<1>	dd DISK_RESET	; 000H
8611	000028D6 [0B2B0000]	<1>	dd RETURN_STATUS	; 001H
8612	000028DA [182B0000]	<1>	dd DISK_READ	; 002H
8613	000028DE [212B0000]	<1>	dd DISK_WRITE	; 003H
8614	000028E2 [2A2B0000]	<1>	dd DISK_VERF	; 004H
8615	000028E6 [422B0000]	<1>	dd FMT_TRK	; 005H
8616	000028EA [8A2A0000]	<1>	dd BAD_COMMAND	; 006H FORMAT BAD SECTORS
8617	000028EE [8A2A0000]	<1>	dd BAD_COMMAND	; 007H FORMAT DRIVE
8618	000028F2 [8A2A0000]	<1>	dd BAD_COMMAND	; 008H RETURN PARAMETERS
8619	000028F6 [092C0000]	<1>	dd INIT_DRV	; 009H
8620	000028FA [682C0000]	<1>	dd RD_LONG	; 00AH
8621	000028FE [712C0000]	<1>	dd WR_LONG	; 00BH
8622	00002902 [7A2C0000]	<1>	dd DISK_SEEK	; 00CH
8623	00002906 [942A0000]	<1>	dd DISK_RESET	; 00DH
8624	0000290A [8A2A0000]	<1>	dd BAD_COMMAND	; 00EH READ BUFFER
8625	0000290E [8A2A0000]	<1>	dd BAD_COMMAND	; 00FH WRITE BUFFER
8626	00002912 [A22C0000]	<1>	dd TST_RDY	; 010H
8627	00002916 [C62C0000]	<1>	dd HDISK_RECAL	; 011H
8628	0000291A [8A2A0000]	<1>	dd BAD_COMMAND	; 012H MEMORY DIAGNOSTIC
8629	0000291E [8A2A0000]	<1>	dd BAD_COMMAND	; 013H DRIVE DIAGNOSTIC
8630	00002922 [FC2C0000]	<1>	dd CTLR_DIAGNOSTIC	; 014H CONTROLLER DIAGNOSTIC

8631		<1>	; 02/02/2015 (Temporary - Retro UNIX 386 v1 - DISK I/O test)
8632	00002926 [8A2A0000]	<1>	dd BAD_COMMAND ; 015h
8633	0000292A [8A2A0000]	<1>	dd BAD_COMMAND ; 016h
8634	0000292E [8A2A0000]	<1>	dd BAD_COMMAND ; 017h
8635	00002932 [8A2A0000]	<1>	dd BAD_COMMAND ; 018h
8636	00002936 [8A2A0000]	<1>	dd BAD_COMMAND ; 019h
8637	0000293A [8A2A0000]	<1>	dd BAD_COMMAND ; 01Ah
8638	0000293E [182B0000]	<1>	dd DISK_READ ; 01Bh ; LBA read
8639	00002942 [212B0000]	<1>	dd DISK_WRITE ; 01Ch ; LBA write
8640		<1>	M1L EQU \$-M1
8641		<1>	
8642		<1>	DISK_IO_CONT:
8643		<1>	;;CALL DDS ; ESTABLISH SEGMENT
8644	00002946 80FC01	<1>	CMP AH,01H ; RETURN STATUS
8645		<1>	;;JNZ short SU0
8646		<1>	;;JMP RETURN_STATUS
8647	00002949 0F84BC010000	<1>	je RETURN_STATUS
8648		<1>	SU0:
8649	0000294F C605[3F710000]00	<1>	MOV byte [DISK_STATUS1],0 ; RESET THE STATUS INDICATOR
8650		<1>	;;PUSH BX ; SAVE DATA ADDRESS
8651		<1>	;mov si, bx ;; 14/02/2015
8652	00002956 89DE	<1>	mov esi, ebx ; 21/02/2015
8653	00002958 8A1D[40710000]	<1>	MOV BL,[HF_NUM] ; GET NUMBER OF DRIVES
8654		<1>	;; 04/01/2015
8655		<1>	;;PUSH AX
8656	0000295E 80E27F	<1>	AND DL,7FH ; GET DRIVE AS 0 OR 1
8657		<1>	; (get drive number as 0 to 3)
8658	00002961 38D3	<1>	CMP BL,DL
8659		<1>	;;JBE BAD_COMMAND_POP ; INVALID DRIVE
8660	00002963 0F8621010000	<1>	jbe BAD_COMMAND ;; 14/02/2015
8661		<1>	;
8662		<1>	;;03/01/2015
8663	00002969 29DB	<1>	sub ebx, ebx
8664	0000296B 88D3	<1>	mov bl, dl
8665		<1>	;sub bh, bh
8666	0000296D 883D[54710000]	<1>	mov [LBAMode], bh ; 0
8667		<1>	;;test byte [bx+hd0_type], 1 ; LBA ready ?
8668		<1>	;test byte [ebx+hd0_type], 1
8669		<1>	;jz short sul ; no
8670		<1>	;inc byte [LBAMode]
8671		<1>	;sul:
8672		<1>	; 21/02/2015 (32 bit modification)
8673		<1>	;04/01/2015
8674	00002973 6650	<1>	push ax ; ***
8675		<1>	;PUSH ES ; **
8676	00002975 6652	<1>	PUSH DX ; *
8677	00002977 6650	<1>	push ax
8678	00002979 E85C060000	<1>	CALL GET_VEC ; GET DISK PARAMETERS
8679		<1>	; 02/02/2015
8680		<1>	;mov ax, [ES:BX+16] ; I/O port base address (1F0h, 170h)
8681	0000297E 668B4310	<1>	mov ax, [ebx+16]
8682	00002982 66A3[546B0000]	<1>	mov [HF_PORT], ax
8683		<1>	;mov dx, [ES:BX+18] ; control port address (3F6h, 376h)
8684	00002988 668B5312	<1>	mov dx, [ebx+18]
8685	0000298C 668915[566B0000]	<1>	mov [HF_REG_PORT], dx
8686		<1>	;mov al, [ES:BX+20] ; head register upper nibble (A0h,B0h,E0h,F0h)
8687	00002993 8A4314	<1>	mov al, [ebx+20]
8688		<1>	; 23/02/2015
8689	00002996 A840	<1>	test al, 40h ; LBA bit (bit 6)
8690	00002998 7406	<1>	jz short sul
8691	0000299A FE05[54710000]	<1>	inc byte [LBAMode] ; 1
8692		<1>	sul:
8693	000029A0 C0E804	<1>	shr al, 4
8694	000029A3 2401	<1>	and al, 1
8695	000029A5 A2[586B0000]	<1>	mov [hf_m_s], al
8696		<1>	;
8697		<1>	; 03/01/2015
8698		<1>	;MOV AL,byte [ES:BX+8] ; GET CONTROL BYTE MODIFIER
8699	000029AA 8A4308	<1>	mov al, [ebx+8]
8700		<1>	;MOV DX,[HF_REG_PORT] ; Device Control register
8701	000029AD EE	<1>	OUT DX,AL ; SET EXTRA HEAD OPTION
8702		<1>	; Control Byte: (= 08h, here)
8703		<1>	; bit 0 - 0
8704		<1>	; bit 1 - nIEN (1 = disable irq)
8705		<1>	; bit 2 - SRST (software RESET)
8706		<1>	; bit 3 - use extra heads (8 to 15)
8707		<1>	; -always set to 1-
8708		<1>	; (bits 3 to 7 are reserved
8709		<1>	; for ATA devices)
8710	000029AE 8A25[41710000]	<1>	MOV AH,[CONTROL_BYTE] ; SET EXTRA HEAD OPTION IN
8711	000029B4 80E4C0	<1>	AND AH,0C0H ; CONTROL BYTE
8712	000029B7 08C4	<1>	OR AH,AL
8713	000029B9 8825[41710000]	<1>	MOV [CONTROL_BYTE],AH
8714		<1>	; 04/01/2015
8715	000029BF 6658	<1>	pop ax
8716	000029C1 665A	<1>	pop dx ; * ;; 14/02/2015
8717	000029C3 20E4	<1>	and ah, ah ; Reset function ?
8718	000029C5 7507	<1>	jnz short su2
8719		<1>	;;pop dx ; * ;; 14/02/2015
8720		<1>	;pop es ; **
8721	000029C7 6658	<1>	pop ax ; ***
8722		<1>	;;pop bx

8723	000029C9	E9C6000000	<1>	jmp	DISK_RESET
8724			<1>	su2:	
8725	000029CE	803D[54710000]00	<1>	cmp	byte [LBAMode], 0
8726	000029D5	7661	<1>	jna	short su3
8727			<1>	;	
8728			<1>	;	02/02/2015 (LBA read/write function calls)
8729	000029D7	80FC1B	<1>	cmp	ah, 1Bh
8730	000029DA	720B	<1>	jb	short lbarw1
8731	000029DC	80FC1C	<1>	cmp	ah, 1Ch
8732	000029DF	775C	<1>	ja	short invldfnc
8733			<1>	;;pop	dx ; * ; 14/02/2015
8734			<1>	;mov	ax, cx ; Lower word of LBA address (bits 0-15)
8735	000029E1	89C8	<1>	mov	eax, ecx ; LBA address (21/02/2015)
8736			<1>	;;	14/02/2015
8737	000029E3	88D1	<1>	mov	cl, dl ; 14/02/2015
8738			<1>	;;mov	dx, bx
8739			<1>	;mov	dx, si ; higher word of LBA address (bits 16-23)
8740			<1>	;;mov	bx, di
8741			<1>	;mov	si, di ; Buffer offset
8742	000029E5	EB31	<1>	jmp	short lbarw2
8743			<1>	lbarw1:	
8744			<1>	;	convert CHS to LBA
8745			<1>	;	
8746			<1>	;	LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM
8747			<1>	;	LBA = "# of Heads" * Sectors/Track * Cylinder + Head * Sectors/Track
8748			<1>	;	+ Sector - 1
8749	000029E7	6652	<1>	push	dx ; * ;; 14/02/2015
8750			<1>	;xor	dh, dh
8751	000029E9	31D2	<1>	xor	edx, edx
8752			<1>	;mov	dl, [ES:BX+14] ; sectors per track (logical)
8753	000029EB	8A530E	<1>	mov	dl, [ebx+14]
8754			<1>	;xor	ah, ah
8755	000029EE	31C0	<1>	xor	eax, eax
8756			<1>	;mov	al, [ES:BX+2] ; heads (logical)
8757	000029F0	8A4302	<1>	mov	al, [ebx+2]
8758	000029F3	FEC8	<1>	dec	al
8759	000029F5	6640	<1>	inc	ax ; 0 = 256
8760	000029F7	66F7E2	<1>	mul	dx
8761			<1>	;	AX = # of Heads" * Sectors/Track
8762	000029FA	6689CA	<1>	mov	dx, cx
8763			<1>	;and	cx, 3Fh ; sector (1 to 63)
8764	000029FD	83E13F	<1>	and	ecx, 3fh
8765	00002A00	86D6	<1>	xchg	dl, dh
8766	00002A02	C0EE06	<1>	shr	dh, 6
8767			<1>	;	DX = cylinder (0 to 1023)
8768			<1>	;mul	dx
8769			<1>	;	DX:AX = # of Heads" * Sectors/Track * Cylinder
8770	00002A05	F7E2	<1>	mul	edx
8771	00002A07	FEC9	<1>	dec	cl ; sector - 1
8772			<1>	;add	ax, cx
8773			<1>	;adc	dx, 0
8774			<1>	;	DX:AX = # of Heads" * Sectors/Track * Cylinder + Sector -1
8775	00002A09	01C8	<1>	add	eax, ecx
8776	00002A0B	6659	<1>	pop	cx ; * ; ch = head, cl = drive number (zero based)
8777			<1>	;push	dx
8778			<1>	;push	ax
8779	00002A0D	50	<1>	push	eax
8780			<1>	;mov	al, [ES:BX+14] ; sectors per track (logical)
8781	00002A0E	8A430E	<1>	mov	al, [ebx+14]
8782	00002A11	F6E5	<1>	mul	ch
8783			<1>	;	AX = Head * Sectors/Track
8784	00002A13	6699	<1>	cwd	
8785			<1>	;pop	dx
8786	00002A15	5A	<1>	pop	edx
8787			<1>	;add	ax, dx
8788			<1>	;pop	dx
8789			<1>	;adc	dx, 0 ; add carry bit
8790	00002A16	01D0	<1>	add	eax, edx
8791			<1>	lbarw2:	
8792	00002A18	29D2	<1>	sub	edx, edx ; 21/02/2015
8793	00002A1A	88CA	<1>	mov	dl, cl ; 21/02/2015
8794	00002A1C	C645F800	<1>	mov	byte [CMD_BLOCK], 0 ; Features Register
8795			<1>	;	NOTE: Features register (1F1h, 171h)
8796			<1>	;	is not used for ATA device R/W functions.
8797			<1>	;	It is old/obsolete 'write precompensation'
8798			<1>	;	register and error register
8799			<1>	;	for old ATA/IDE devices.
8800			<1>	;	18/01/2014
8801			<1>	;mov	ch, [hf_m_s] ; Drive 0 (master) or 1 (slave)
8802	00002A20	8A0D[586B0000]	<1>	mov	cl, [hf_m_s]
8803			<1>	;shl	ch, 4 ; bit 4 (drive bit)
8804			<1>	;or	ch, 0E0h ; bit 5 = 1
8805			<1>	;	bit 6 = 1 = LBA mode
8806			<1>	;	bit 7 = 1
8807	00002A26	80C90E	<1>	or	cl, 0Eh ; 1110b
8808			<1>	;and	dh, 0Fh ; LBA byte 4 (bits 24 to 27)
8809	00002A29	25FFFFFFF0F	<1>	and	eax, 0FFFFFFFh
8810	00002A2E	C1E11C	<1>	shl	ecx, 28 ; 21/02/2015
8811			<1>	;or	dh, ch
8812	00002A31	09C8	<1>	or	eax, ecx
8813			<1>	;;mov	[CMD_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)
8814			<1>	;	(Sector Number Register)



8815		<1>	;;mov [CMD_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)
8816		<1>	; (Cylinder Low Register)
8817		<1>	;mov [CMD_BLOCK+2], ax ; LBA byte 1, 2
8818		<1>	;mov [CMD_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)
8819		<1>	; (Cylinder High Register)
8820		<1>	;;mov [CMD_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)
8821		<1>	; (Drive/Head Register)
8822		<1>	
8823		<1>	;mov [CMD_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits
8824	00002A33 8945FA	<1>	mov [CMD_BLOCK+2], eax ; 21/02/2015
8825		<1>	;14/02/2015
8826		<1>	;mov dl, cl ; Drive number (INIT_DRV)
8827	00002A36 EB38	<1>	jmp short su4
8828		<1>	su3:
8829		<1>	; 02/02/2015
8830		<1>	; (Temporary functions 1Bh & 1Ch are not valid for CHS mode)
8831	00002A38 80FC14	<1>	cmp ah, 14h
8832	00002A3B 7604	<1>	jna short chsfnc
8833		<1>	invldfnc:
8834		<1>	; 14/02/2015
8835		<1>	;pop es ; **
8836	00002A3D 6658	<1>	pop ax ; ***
8837		<1>	;jmp short BAD_COMMAND_POP
8838	00002A3F EB49	<1>	jmp short BAD_COMMAND
8839		<1>	chsfnc:
8840		<1>	;MOV AX,[ES:BX+5] ; GET WRITE PRE-COMPENSATION CYLINDER
8841	00002A41 668B4305	<1>	mov ax, [ebx+5]
8842	00002A45 66C1E802	<1>	SHR AX,2
8843	00002A49 8845F8	<1>	MOV [CMD_BLOCK],AL
8844		<1>	;;MOV AL,[ES:BX+8] ; GET CONTROL BYTE MODIFIER
8845		<1>	;;PUSH DX
8846		<1>	;;MOV DX,[HF_REG_PORT]
8847		<1>	;;OUT DX,AL ; SET EXTRA HEAD OPTION
8848		<1>	;;POP DX ; *
8849		<1>	;;POP ES ; **
8850		<1>	;;MOV AH,[CONTROL_BYTE] ; SET EXTRA HEAD OPTION IN
8851		<1>	;;AND AH,0C0H ; CONTROL BYTE
8852		<1>	;;OR AH,AL
8853		<1>	;;MOV [CONTROL_BYTE],AH
8854		<1>	;
8855	00002A4C 88C8	<1>	MOV AL,CL ; GET SECTOR NUMBER
8856	00002A4E 243F	<1>	AND AL,3FH
8857	00002A50 8845FA	<1>	MOV [CMD_BLOCK+2],AL
8858	00002A53 886DFB	<1>	MOV [CMD_BLOCK+3],CH ; GET CYLINDER NUMBER
8859	00002A56 88C8	<1>	MOV AL,CL
8860	00002A58 C0E806	<1>	SHR AL,6
8861	00002A5B 8845FC	<1>	MOV [CMD_BLOCK+4],AL ; CYLINDER HIGH ORDER 2 BITS
8862		<1>	;;05/01/2015
8863		<1>	;;MOV AL,DL ; DRIVE NUMBER
8864	00002A5E A0[586B0000]	<1>	mov al, [hf_m_s]
8865	00002A63 C0E004	<1>	SHL AL,4
8866	00002A66 80E60F	<1>	AND DH,0FH ; HEAD NUMBER
8867	00002A69 08F0	<1>	OR AL,DH
8868		<1>	;OR AL,80H or 20H
8869	00002A6B 0CA0	<1>	OR AL,80h+20h ; ECC AND 512 BYTE SECTORS
8870	00002A6D 8845FD	<1>	MOV [CMD_BLOCK+5],AL ; ECC/SIZE/DRIVE/HEAD
8871		<1>	su4:
8872		<1>	;POP ES ; **
8873		<1>	;; 14/02/2015
8874		<1>	;;POP AX
8875		<1>	;;MOV [CMD_BLOCK+1],AL ; SECTOR COUNT
8876		<1>	;;PUSH AX
8877		<1>	;;MOV AL,AH ; GET INTO LOW BYTE
8878		<1>	;;XOR AH,AH ; ZERO HIGH BYTE
8879		<1>	;;SAL AX,1 ; *2 FOR TABLE LOOKUP
8880	00002A70 6658	<1>	pop ax ; ***
8881	00002A72 8845F9	<1>	mov [CMD_BLOCK+1], al
8882	00002A75 29DB	<1>	sub ebx, ebx
8883	00002A77 88E3	<1>	mov bl, ah
8884		<1>	;xor bh, bh
8885		<1>	;sal bx, 1
8886	00002A79 66C1E302	<1>	sal bx, 2 ; 32 bit offset (21/02/2015)
8887		<1>	;;MOV SI,AX ; PUT INTO SI FOR BRANCH
8888		<1>	;;CMP AX,M1L ; TEST WITHIN RANGE
8889		<1>	;;JNB short BAD_COMMAND_POP
8890		<1>	;cmp bx, M1L
8891	00002A7D 83FB74	<1>	cmp ebx, M1L
8892	00002A80 7308	<1>	jnb short BAD_COMMAND
8893		<1>	;xchg bx, si
8894	00002A82 87DE	<1>	xchg ebx, esi
8895		<1>	;;;POP AX ; RESTORE AX
8896		<1>	;;;POP BX ; AND DATA ADDRESS
8897		<1>	
8898		<1>	;;PUSH CX
8899		<1>	;;PUSH AX ; ADJUST ES:BX
8900		<1>	;MOV CX,BX ; GET 3 HIGH ORDER NIBBLES OF BX
8901		<1>	;SHR CX,4
8902		<1>	;MOV AX,ES
8903		<1>	;ADD AX,CX
8904		<1>	;MOV ES,AX
8905		<1>	;AND BX,000FH ; ES:BX CHANGED TO ES:000X
8906		<1>	;;POP AX

```

8907 <1> ;;POP CX
8908 <1> ;;JMP word [CS:SI+M1]
8909 <1> ;jmp word [SI+M1]
8910 00002A84 FFA6[D2280000] <1> jmp dword [esi+M1]
8911 <1> ;;BAD_COMMAND_POP:
8912 <1> ;; POP AX
8913 <1> ;; POP BX
8914 <1> BAD_COMMAND:
8915 00002A8A C605[3F710000]01 <1> MOV byte [DISK_STATUS1],BAD_CMD ; COMMAND ERROR
8916 00002A91 B000 <1> MOV AL,0
8917 00002A93 C3 <1> RETn
8918 <1>
8919 <1> ;-----
8920 <1> ; RESET THE DISK SYSTEM (AH=00H) :
8921 <1> ;-----
8922 <1>
8923 <1> ; 18-1-2015 : one controller reset (not other one)
8924 <1>
8925 <1> DISK_RESET:
8926 00002A94 FA <1> CLI
8927 00002A95 E4A1 <1> IN AL,INTB01 ; GET THE MASK REGISTER
8928 <1> ;JMP $+2
8929 <1> IODELAY
8930 00002A97 EB00 <2> jmp short $+2
8931 00002A99 EB00 <2> jmp short $+2
8932 <1> ;AND AL,0BFH ; ENABLE FIXED DISK INTERRUPT
8933 00002A9B 243F <1> and al,3Fh ; 22/12/2014 (IRQ 14 & IRQ 15)
8934 00002A9D E6A1 <1> OUT INTB01,AL
8935 00002A9F FB <1> STI ; START INTERRUPTS
8936 <1> ; 14/02/2015
8937 00002AA0 6689D7 <1> mov di, dx
8938 <1> ; 04/01/2015
8939 <1> ;xor di,di
8940 <1> drst0:
8941 00002AA3 B004 <1> MOV AL,04H ; bit 2 - SRST
8942 <1> ;MOV DX,HF_REG_PORT
8943 00002AA5 668B15[566B0000] <1> MOV DX,[HF_REG_PORT]
8944 00002AAC EE <1> OUT DX,AL ; RESET
8945 <1> ;MOV CX,10 ; DELAY COUNT
8946 <1> ;DRD: DEC CX
8947 <1> ;JNZ short DRD ; WAIT 4.8 MICRO-SEC
8948 <1> ;mov cx,2 ; wait for 30 micro seconds
8949 00002AAD B902000000 <1> mov ecx, 2 ; 21/02/2015
8950 00002AB2 E875EBFFFF <1> call WAITF ; (Award Bios 1999 - WAIT_REFRESH,
8951 <1> ; 40 micro seconds)
8952 00002AB7 A0[41710000] <1> mov al,[CONTROL_BYTE]
8953 00002ABC 240F <1> AND AL,0FH ; SET HEAD OPTION
8954 00002ABE EE <1> OUT DX,AL ; TURN RESET OFF
8955 00002ABF E80E040000 <1> CALL NOT_BUSY
8956 00002AC4 7515 <1> JNZ short DRERR ; TIME OUT ON RESET
8957 00002AC6 668B15[546B0000] <1> MOV DX,[HF_PORT]
8958 00002ACD FEC2 <1> inc dl ; HF_PORT+1
8959 <1> ; 02/01/2015 - Award BIOS 1999 - AHDSK.ASM
8960 <1> ;mov cl, 10
8961 00002ACF B90A000000 <1> mov ecx, 10 ; 21/02/2015
8962 <1> drst1:
8963 00002AD4 EC <1> IN AL,DX ; GET RESET STATUS
8964 00002AD5 3C01 <1> CMP AL,1
8965 <1> ; 04/01/2015
8966 00002AD7 740A <1> jz short drst2
8967 <1> ;JNZ short DRERR ; BAD RESET STATUS
8968 <1> ; Drive/Head Register - bit 4
8969 00002AD9 E2F9 <1> loop drst1
8970 <1> DRERR:
8971 00002ADB C605[3F710000]05 <1> MOV byte [DISK_STATUS1],BAD_RESET ; CARD FAILED
8972 00002AE2 C3 <1> RETn
8973 <1> drst2:
8974 <1> ; 14/02/2015
8975 00002AE3 6689FA <1> mov dx,di
8976 <1> ;drst3:
8977 <1> ;; 05/01/2015
8978 <1> ; shl di,1
8979 <1> ;; 04/01/2015
8980 <1> ;mov ax,[di+hd_cports]
8981 <1> ;cmp ax,[HF_REG_PORT]
8982 <1> ;je short drst4
8983 <1> ;mov [HF_REG_PORT], ax
8984 <1> ;; 03/01/2015
8985 <1> ;mov ax,[di+hd_ports]
8986 <1> ; mov [HF_PORT], ax
8987 <1> ;; 05/01/2014
8988 <1> ; shr di,1
8989 <1> ;; 04/01/2015
8990 <1> ; jmp short drst0 ; reset other controller
8991 <1> ;drst4:
8992 <1> ;; 05/01/2015
8993 <1> ; shr di,1
8994 <1> ;mov al,[di+hd_dregs]
8995 <1> ;and al,10h ; bit 4 only
8996 <1> ; shr al,4 ; bit 4 -> bit 0
8997 <1> ;mov [hf_m_s], al ; (0 = master, 1 = slave)
8998 <1> ;

```

8999	00002AE6	A0[586B0000]	<1>	mov	al, [hf_m_s] ; 18/01/2015
9000	00002AEB	A801	<1>	test	al,1
9001			<1>	;jnz	short drst6
9002	00002AED	7516	<1>	jnz	short drst4
9003	00002AEF	8065FDEF	<1>	AND	byte [CMD_BLOCK+5],0EFH ; SET TO DRIVE 0
9004			<1>	;drst5:	
9005			<1>	drst3:	
9006	00002AF3	E811010000	<1>	CALL	INIT_DRV ; SET MAX HEADS
9007			<1>	;mov	dx,di
9008	00002AF8	E8C9010000	<1>	CALL	HDISK_RECAL ; RECAL TO RESET SEEK SPEED
9009			<1>		; 04/01/2014
9010			<1>	;inc	di
9011			<1>	;mov	dx,di
9012			<1>	;cmp	dl,[HF_NUM]
9013			<1>	;jb	short drst3
9014			<1>	;DRE:	
9015	00002AFD	C605[3F710000]00	<1>	MOV	byte [DISK_STATUS1],0 ; IGNORE ANY SET UP ERRORS
9016	00002B04	C3	<1>	RETN	
9017			<1>	;drst6:	
9018			<1>	drst4:	; Drive/Head Register - bit 4
9019	00002B05	804DFD10	<1>	OR	byte [CMD_BLOCK+5],010H ; SET TO DRIVE 1
9020			<1>	;jmp	short drst5
9021	00002B09	EBE8	<1>	jmp	short drst3
9022			<1>		
9023			<1>	;-----	
9024			<1>	; DISK STATUS ROUTINE (AH = 01H) :	
9025			<1>	;-----	
9026			<1>		
9027			<1>	RETURN_STATUS:	
9028	00002B0B	A0[3F710000]	<1>	MOV	AL,[DISK_STATUS1] ; OBTAIN PREVIOUS STATUS
9029	00002B10	C605[3F710000]00	<1>	MOV	byte [DISK_STATUS1],0 ; RESET STATUS
9030	00002B17	C3	<1>	RETN	
9031			<1>		
9032			<1>	;-----	
9033			<1>	; DISK READ ROUTINE (AH = 02H) :	
9034			<1>	;-----	
9035			<1>		
9036			<1>	DISK_READ:	
9037	00002B18	C645FE20	<1>	MOV	byte [CMD_BLOCK+6],READ_CMD
9038	00002B1C	E930020000	<1>	JMP	COMMANDI
9039			<1>		
9040			<1>	;-----	
9041			<1>	; DISK WRITE ROUTINE (AH = 03H) :	
9042			<1>	;-----	
9043			<1>		
9044			<1>	DISK_WRITE:	
9045	00002B21	C645FE30	<1>	MOV	byte [CMD_BLOCK+6],WRITE_CMD
9046	00002B25	E97C020000	<1>	JMP	COMMANDO
9047			<1>		
9048			<1>	;-----	
9049			<1>	; DISK VERIFY (AH = 04H) :	
9050			<1>	;-----	
9051			<1>		
9052			<1>	DISK_VERF:	
9053	00002B2A	C645FE40	<1>	MOV	byte [CMD_BLOCK+6],VERIFY_CMD
9054	00002B2E	E8EA020000	<1>	CALL	COMMAND
9055	00002B33	750C	<1>	JNZ	short VERF_EXIT ; CONTROLLER STILL BUSY
9056	00002B35	E85C030000	<1>	CALL	_WAIT ; (Original: CALL WAIT)
9057	00002B3A	7505	<1>	JNZ	short VERF_EXIT ; TIME OUT
9058	00002B3C	E8E9030000	<1>	CALL	CHECK_STATUS
9059			<1>	VERF_EXIT:	
9060	00002B41	C3	<1>	RETN	
9061			<1>		
9062			<1>	;-----	
9063			<1>	; FORMATTING (AH = 05H) :	
9064			<1>	;-----	
9065			<1>		
9066			<1>	FMT_TRK:	; FORMAT TRACK (AH = 005H)
9067	00002B42	C645FE50	<1>	MOV	byte [CMD_BLOCK+6],FMTTRK_CMD
9068			<1>	;PUSH	ES
9069			<1>	;PUSH	BX
9070	00002B46	53	<1>	push	ebx
9071	00002B47	E88E040000	<1>	CALL	GET_VEC ; GET DISK PARAMETERS ADDRESS
9072			<1>	;MOV	AL,[ES:BX+14] ; GET SECTORS/TRACK
9073	00002B4C	8A430E	<1>	mov	al, [ebx+14]
9074	00002B4F	8845F9	<1>	MOV	[CMD_BLOCK+1],AL ; SET SECTOR COUNT IN COMMAND
9075	00002B52	5B	<1>	pop	ebx
9076			<1>	;POP	BX
9077			<1>	;POP	ES
9078	00002B53	E955020000	<1>	JMP	CMD_OF ; GO EXECUTE THE COMMAND
9079			<1>		
9080			<1>	;-----	
9081			<1>	; READ DASD TYPE (AH = 15H) :	
9082			<1>	;-----	
9083			<1>		
9084			<1>	READ_DASD_TYPE:	
9085			<1>	READ_D_T:	; GET DRIVE PARAMETERS
9086	00002B58	1E	<1>	PUSH	DS ; SAVE REGISTERS
9087			<1>	;PUSH	ES
9088	00002B59	53	<1>	PUSH	ebx
9089			<1>	;CALL	DDS ; ESTABLISH ADDRESSING
9090			<1>	;push	cs

```

9091                                     <1> ;pop ds
9092 00002B5A 66BB1000                 <1> mov bx, KDATA
9093 00002B5E 8EDB                     <1> mov ds, bx
9094                                     <1> ;mov es, bx
9095 00002B60 C605[3F710000]00         <1> MOV byte [DISK_STATUS1],0
9096 00002B67 8A1D[40710000]           <1> MOV BL,[HF_NUM] ; GET NUMBER OF DRIVES
9097 00002B6D 80E27F                   <1> AND DL,7FH ; GET DRIVE NUMBER
9098 00002B70 38D3                     <1> CMP BL,DL
9099 00002B72 7625                     <1> JBE short RDT_NOT_PRESENT ; RETURN DRIVE NOT PRESENT
9100 00002B74 E861040000               <1> CALL GET_VEC ; GET DISK PARAMETER ADDRESS
9101                                     <1> ;MOV AL,[ES:BX+2] ; HEADS
9102 00002B79 8A4302                   <1> mov al, [ebx+2]
9103                                     <1> ;MOV CL,[ES:BX+14]
9104 00002B7C 8A4B0E                   <1> mov cl, [ebx+14]
9105 00002B7F F6E9                     <1> IMUL CL ; * NUMBER OF SECTORS
9106                                     <1> ;MOV CX,[ES:BX] ; MAX NUMBER OF CYLINDERS
9107 00002B81 668B0B                   <1> mov cx, [ebx]
9108                                     <1> ;
9109                                     <1> ; 02/01/2015
9110                                     <1> ; ** leave the last cylinder as reserved for diagnostics **
9111                                     <1> ; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)
9112 00002B84 6649                     <1> DEC CX ; LEAVE ONE FOR DIAGNOSTICS
9113                                     <1> ;
9114 00002B86 66F7E9                   <1> IMUL CX ; NUMBER OF SECTORS
9115 00002B89 6689D1                   <1> MOV CX,DX ; HIGH ORDER HALF
9116 00002B8C 6689C2                   <1> MOV DX,AX ; LOW ORDER HALF
9117                                     <1> ;SUB AX,AX
9118 00002B8F 28C0                     <1> sub al, al
9119 00002B91 B403                     <1> MOV AH,03H ; INDICATE FIXED DISK
9120 00002B93 5B                       <1> RDT2: POP eBX ; RESTORE REGISTERS
9121                                     <1> ;POP ES
9122 00002B94 1F                       <1> POP DS
9123 00002B95 F8                       <1> CLC ; CLEAR CARRY
9124                                     <1> ;RETf 2
9125 00002B96 CA0400                   <1> retf 4
9126                                     <1> RDT_NOT_PRESENT:
9127 00002B99 6629C0                   <1> SUB AX,AX ; DRIVE NOT PRESENT RETURN
9128 00002B9C 6689C1                   <1> MOV CX,AX ; ZERO BLOCK COUNT
9129 00002B9F 6689C2                   <1> MOV DX,AX
9130 00002BA2 EBEF                     <1> JMP short RDT2
9131                                     <1>
9132                                     <1> ;-----
9133                                     <1> ; GET PARAMETERS (AH = 08H) :
9134                                     <1> ;-----
9135                                     <1>
9136                                     <1> GET_PARM_N:
9137                                     <1> ;GET_PARM: ; GET DRIVE PARAMETERS
9138 00002BA4 1E                       <1> PUSH DS ; SAVE REGISTERS
9139                                     <1> ;PUSH ES
9140 00002BA5 53                       <1> PUSH eBX
9141                                     <1> ;MOV AX,ABS0 ; ESTABLISH ADDRESSING
9142                                     <1> ;MOV DS,AX
9143                                     <1> ;TEST DL,1 ; CHECK FOR DRIVE 1
9144                                     <1> ;JZ short G0
9145                                     <1> ;LES BX,@HF1_TBL_VEC
9146                                     <1> ;JMP SHORT G1
9147                                     <1> ;G0: LES BX,@HF_TBL_VEC
9148                                     <1> ;G1:
9149                                     <1> ;CALL DDS ; ESTABLISH SEGMENT
9150                                     <1> ; 22/12/2014
9151                                     <1> ;push cs
9152                                     <1> ;pop ds
9153 00002BA6 66BB1000                 <1> mov bx, KDATA
9154 00002BAA 8EDB                     <1> mov ds, bx
9155                                     <1> ;mov es, bx
9156                                     <1> ;
9157 00002BAC 80EA80                   <1> SUB DL,80H
9158 00002BAF 80FA04                   <1> CMP DL,MAX_FILE ; TEST WITHIN RANGE
9159 00002BB2 7341                     <1> JAE short G4
9160                                     <1> ;
9161 00002BB4 31DB                     <1> xor ebx, ebx ; 21/02/2015
9162                                     <1> ; 22/12/2014
9163 00002BB6 88D3                     <1> mov bl, dl
9164                                     <1> ;xor bh, bh
9165 00002BB8 C0E302                   <1> shl bl, 2 ; convert index to offset
9166                                     <1> ;add bx, HF_TBL_VEC
9167 00002BBB 81C3[44710000]           <1> add ebx, HF_TBL_VEC
9168                                     <1> ;mov ax, [bx+2]
9169                                     <1> ;mov es, ax ; dpt segment
9170                                     <1> ;mov bx, [bx] ; dpt offset
9171 00002BC1 8B1B                     <1> mov ebx, [ebx] ; 32 bit offset
9172                                     <1>
9173 00002BC3 C605[3F710000]00         <1> MOV byte [DISK_STATUS1],0
9174                                     <1> ;MOV AX,[ES:BX] ; MAX NUMBER OF CYLINDERS
9175 00002BCA 668B03                   <1> mov ax, [ebx]
9176                                     <1> ;;SUB AX,2 ; ADJUST FOR 0-N
9177 00002BCD 6648                     <1> dec ax ; max. cylinder number
9178 00002BCF 88C5                     <1> MOV CH,AL
9179 00002BD1 66250003               <1> AND AX,6300H ; HIGH TWO BITS OF CYLINDER
9180 00002BD5 66D1E8                   <1> SHR AX,1
9181 00002BD8 66D1E8                   <1> SHR AX,1
9182                                     <1> ;OR AL,[ES:BX+14] ; SECTORS

```

9183	00002BDB	0A430E	<1>	or	al, [ebx+14]	
9184	00002BDE	88C1	<1>	MOV	CL,AL	
9185			<1>	;MOV	DH,[ES:BX+2]	; HEADS
9186	00002BE0	8A7302	<1>	mov	dh, [ebx+2]	
9187	00002BE3	FECE	<1>	DEC	DH	; 0-N RANGE
9188	00002BE5	8A15[40710000]	<1>	MOV	DL,[HF_NUM]	; DRIVE COUNT
9189	00002BEB	6629C0	<1>	SUB	AX,AX	
9190			<1>		;27/12/2014	
9191			<1>		; ES:DI = Address of disk parameter table from BIOS	
9192			<1>		; (Programmer's Guide to the AMIBIOS - 1993)	
9193			<1>	;mov	di, bx	; HDPT offset
9194	00002BEE	89DF	<1>	mov	edi, ebx	
9195			<1>	G5:		
9196	00002BF0	5B	<1>	POP	eBX	; RESTORE REGISTERS
9197			<1>	;POP	ES	
9198	00002BF1	1F	<1>	POP	DS	
9199			<1>	;RETf	2	
9200	00002BF2	CA0400	<1>	retf	4	
9201			<1>	G4:		
9202	00002BF5	C605[3F710000]07	<1>	MOV	byte [DISK_STATUS1],INIT_FAIL	; OPERATION FAILED
9203	00002BFC	B407	<1>	MOV	AH,INIT_FAIL	
9204	00002BFE	28C0	<1>	SUB	AL,AL	
9205	00002C00	6629D2	<1>	SUB	DX,DX	
9206	00002C03	6629C9	<1>	SUB	CX,CX	
9207	00002C06	F9	<1>	STC		; SET ERROR FLAG
9208	00002C07	EBE7	<1>	JMP	short G5	
9209			<1>			
9210			<1>		;-----	
9211			<1>	; INITIALIZE DRIVE	(AH = 09H) :	
9212			<1>		;-----	
9213			<1>		; 03/01/2015	
9214			<1>		; According to ATA-ATAPI specification v2.0 to v5.0	
9215			<1>		; logical sector per logical track	
9216			<1>		; and logical heads - 1 would be set but	
9217			<1>		; it is seen as it will be good	
9218			<1>		; if physical parameters will be set here	
9219			<1>		; because, number of heads <= 16.	
9220			<1>		; (logical heads usually more than 16)	
9221			<1>		; NOTE: ATA logical parameters (software C, H, S)	
9222			<1>		; == INT 13h physical parameters	
9223			<1>			
9224			<1>	;INIT_DRV:		
9225			<1>	;MOV	byte [CMD_BLOCK+6],SET_PARM_CMD	
9226			<1>	;CALL	GET_VEC	; ES:BX -> PARAMETER BLOCK
9227			<1>	;MOV	AL,[ES:BX+2]	; GET NUMBER OF HEADS
9228			<1>	;DEC	AL	; CONVERT TO 0-INDEX
9229			<1>	;MOV	AH,[CMD_BLOCK+5]	; GET SDH REGISTER
9230			<1>	;AND	AH,0F0H	; CHANGE HEAD NUMBER
9231			<1>	;OR	AH,AL	; TO MAX HEAD
9232			<1>	;MOV	[CMD_BLOCK+5],AH	
9233			<1>	;MOV	AL,[ES:BX+14]	; MAX SECTOR NUMBER
9234			<1>	;MOV	[CMD_BLOCK+1],AL	
9235			<1>	;SUB	AX,AX	
9236			<1>	;MOV	[CMD_BLOCK+3],AL	; ZERO FLAGS
9237			<1>	;CALL	COMMAND	; TELL CONTROLLER
9238			<1>	;JNZ	short INIT_EXIT	; CONTROLLER BUSY ERROR
9239			<1>	;CALL	NOT_BUSY	; WAIT FOR IT TO BE DONE
9240			<1>	;JNZ	short INIT_EXIT	; TIME OUT
9241			<1>	;CALL	CHECK_STATUS	
9242			<1>	;INIT_EXIT:		
9243			<1>	;RETn		
9244			<1>			
9245			<1>		; 04/01/2015	
9246			<1>		; 02/01/2015 - Derived from from AWARD BIOS 1999	
9247			<1>		; AHDSK.ASM - INIT_DRIVE	
9248			<1>	INIT_DRV:		
9249			<1>	;xor	ah,ah	
9250	00002C09	31C0	<1>	xor	eax, eax	; 21/02/2015
9251	00002C0B	B00B	<1>	mov	al,11	; Physical heads from translated HDPT
9252	00002C0D	3825[54710000]	<1>	cmp	[LBAMode], ah	; 0
9253	00002C13	7702	<1>	ja	short idrv0	
9254	00002C15	B002	<1>	mov	al,2	; Physical heads from standard HDPT
9255			<1>	idrv0:		
9256			<1>		; DL = drive number (0 based)	
9257	00002C17	E8BE030000	<1>	call	GET_VEC	
9258			<1>	;push	bx	
9259	00002C1C	53	<1>	push	ebx	; 21/02/2015
9260			<1>	;add	bx,ax	
9261	00002C1D	01C3	<1>	add	ebx, eax	
9262			<1>		; 05/01/2015	
9263	00002C1F	8A25[586B0000]	<1>	mov	ah, [hf_m_s]	; drive number (0= master, 1= slave)
9264			<1>	;and	ah,1	
9265	00002C25	C0E404	<1>	shl	ah,4	
9266	00002C28	80CCA0	<1>	or	ah,0A0h	; Drive/Head register - 10100000b (A0h)
9267			<1>	;mov	al,[es:bx]	
9268	00002C2B	8A03	<1>	mov	al, [ebx]	; 21/02/2015
9269	00002C2D	FEC8	<1>	dec	al	; last head number
9270			<1>	;and	al,0Fh	
9271	00002C2F	08E0	<1>	or	al,ah	; lower 4 bits for head number
9272			<1>			
9273	00002C31	C645FE91	<1>	mov	byte [CMD_BLOCK+6],SET_PARM_CMD	
9274	00002C35	8845FD	<1>	mov	[CMD_BLOCK+5],al	

9275		<1>	;pop bx
9276	00002C38 5B	<1>	pop ebx
9277	00002C39 29C0	<1>	sub eax, eax ; 21/02/2015
9278	00002C3B B004	<1>	mov al,4 ; Physical sec per track from translated HDPT
9279	00002C3D 803D[54710000]00	<1>	cmp byte [LBAMode], 0
9280	00002C44 7702	<1>	ja short idrv1
9281	00002C46 B00E	<1>	mov al,14 ; Physical sec per track from standard HDPT
9282		<1>	idrv1:
9283		<1>	;xor ah,ah
9284		<1>	;add bx,ax
9285	00002C48 01C3	<1>	add ebx, eax ; 21/02/2015
9286		<1>	;mov al,[es:bx]
9287		<1>	; sector number
9288	00002C4A 8A03	<1>	mov al, [ebx]
9289	00002C4C 8845F9	<1>	mov [CMD_BLOCK+1],al
9290	00002C4F 28C0	<1>	sub al,al
9291	00002C51 8845FB	<1>	mov [CMD_BLOCK+3],al ; ZERO FLAGS
9292	00002C54 E8C4010000	<1>	call COMMAND ; TELL CONTROLLER
9293	00002C59 750C	<1>	jnz short INIT_EXIT ; CONTROLLER BUSY ERROR
9294	00002C5B E872020000	<1>	call NOT_BUSY ; WAIT FOR IT TO BE DONE
9295	00002C60 7505	<1>	jnz short INIT_EXIT ; TIME OUT
9296	00002C62 E8C3020000	<1>	call CHECK_STATUS
9297		<1>	INIT_EXIT:
9298	00002C67 C3	<1>	RETn
9299		<1>	
9300		<1>	;-----
9301		<1>	; READ LONG (AH = 0AH) :
9302		<1>	;-----
9303		<1>	
9304		<1>	RD_LONG:
9305		<1>	;MOV @CMD_BLOCK+6,READ_CMD OR ECC_MODE
9306	00002C68 C645FE22	<1>	mov byte [CMD_BLOCK+6],READ_CMD + ECC_MODE
9307	00002C6C E9E0000000	<1>	JMP COMMANDI
9308		<1>	
9309		<1>	;-----
9310		<1>	; WRITE LONG (AH = 0BH) :
9311		<1>	;-----
9312		<1>	
9313		<1>	WR_LONG:
9314		<1>	;MOV @CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
9315	00002C71 C645FE32	<1>	MOV byte [CMD_BLOCK+6],WRITE_CMD + ECC_MODE
9316	00002C75 E92C010000	<1>	JMP COMMANDO
9317		<1>	
9318		<1>	;-----
9319		<1>	; SEEK (AH = 0CH) :
9320		<1>	;-----
9321		<1>	
9322		<1>	DISK_SEEK:
9323	00002C7A C645FE70	<1>	MOV byte [CMD_BLOCK+6],SEEK_CMD
9324	00002C7E E89A010000	<1>	CALL COMMAND
9325	00002C83 751C	<1>	JNZ short DS_EXIT ; CONTROLLER BUSY ERROR
9326	00002C85 E80C020000	<1>	CALL _WAIT
9327	00002C8A 7515	<1>	JNZ DS_EXIT ; TIME OUT ON SEEK
9328	00002C8C E899020000	<1>	CALL CHECK_STATUS
9329	00002C91 803D[3F710000]40	<1>	cmp byte [DISK_STATUS1],BAD_SEEK
9330	00002C98 7507	<1>	JNE short DS_EXIT
9331	00002C9A C605[3F710000]00	<1>	MOV byte [DISK_STATUS1],0
9332		<1>	DS_EXIT:
9333	00002CA1 C3	<1>	RETn
9334		<1>	
9335		<1>	;-----
9336		<1>	; TEST DISK READY (AH = 10H) :
9337		<1>	;-----
9338		<1>	
9339		<1>	TST_RDY: ; WAIT FOR CONTROLLER
9340	00002CA2 E82B020000	<1>	CALL NOT_BUSY
9341	00002CA7 751C	<1>	JNZ short TR_EX
9342	00002CA9 8A45FD	<1>	MOV AL,[CMD_BLOCK+5] ; SELECT DRIVE
9343	00002CAC 668B15[546B0000]	<1>	MOV DX,[HF_PORT]
9344	00002CB3 80C206	<1>	add dl,6
9345	00002CB6 EE	<1>	OUT DX,AL
9346	00002CB7 E886020000	<1>	CALL CHECK_ST ; CHECK STATUS ONLY
9347	00002CBC 7507	<1>	JNZ short TR_EX
9348	00002CBE C605[3F710000]00	<1>	MOV byte [DISK_STATUS1],0 ; WIPE OUT DATA CORRECTED ERROR
9349		<1>	TR_EX:
9350	00002CC5 C3	<1>	RETn
9351		<1>	
9352		<1>	;-----
9353		<1>	; RECALIBRATE (AH = 11H) :
9354		<1>	;-----
9355		<1>	
9356		<1>	HDISK_RECAL:
9357	00002CC6 C645FE10	<1>	MOV byte [CMD_BLOCK+6],RECAL_CMD ; 10h, 16
9358	00002CCA E84E010000	<1>	CALL COMMAND ; START THE OPERATION
9359	00002CCF 7523	<1>	JNZ short RECAL_EXIT ; ERROR
9360	00002CD1 E8C0010000	<1>	CALL _WAIT ; WAIT FOR COMPLETION
9361	00002CD6 7407	<1>	JZ short RECAL_X ; TIME OUT ONE OK ?
9362	00002CD8 E8B9010000	<1>	CALL _WAIT ; WAIT FOR COMPLETION LONGER
9363	00002CDD 7515	<1>	JNZ short RECAL_EXIT ; TIME OUT TWO TIMES IS ERROR
9364		<1>	RECAL_X:
9365	00002CDF E846020000	<1>	CALL CHECK_STATUS
9366	00002CE4 803D[3F710000]40	<1>	cmp byte [DISK_STATUS1],BAD_SEEK ; SEEK NOT COMPLETE

9367	00002CEB	7507	<1>	JNE	short	RECAL_EXIT	; IS OK
9368	00002CED	C605[3F710000]00	<1>	MOV	byte	[DISK_STATUS1],0	
9369			<1>	RECAL_EXIT:			
9370	00002CF4	803D[3F710000]00	<1>	CMP	byte	[DISK_STATUS1],0	
9371	00002CFB	C3	<1>	RETN			
9372			<1>				
9373			<1>	;-----			
9374			<1>	; CONTROLLER DIAGNOSTIC (AH = 14H) :			
9375			<1>	;-----			
9376			<1>				
9377			<1>	CTLR_DIAGNOSTIC:			
9378	00002CFC	FA	<1>	CLI			; DISABLE INTERRUPTS WHILE CHANGING MASK
9379	00002CFD	E4A1	<1>	IN	AL,INTB01		; TURN ON SECOND INTERRUPT CHIP
9380			<1>	;AND	AL,0BFH		
9381	00002CFF	243F	<1>	and	al, 3Fh		; enable IRQ 14 & IRQ 15
9382			<1>	;JMP	\$+2		
9383			<1>	IODELAY			
9384	00002D01	EB00	<2>	jmp	short \$+2		
9385	00002D03	EB00	<2>	jmp	short \$+2		
9386	00002D05	E6A1	<1>	OUT	INTB01,AL		
9387			<1>	IODELAY			
9388	00002D07	EB00	<2>	jmp	short \$+2		
9389	00002D09	EB00	<2>	jmp	short \$+2		
9390	00002D0B	E421	<1>	IN	AL,INTA01		; LET INTERRUPTS PASS THRU TO
9391	00002D0D	24FB	<1>	AND	AL,0FBH		; SECOND CHIP
9392			<1>	;JMP	\$+2		
9393			<1>	IODELAY			
9394	00002D0F	EB00	<2>	jmp	short \$+2		
9395	00002D11	EB00	<2>	jmp	short \$+2		
9396	00002D13	E621	<1>	OUT	INTA01,AL		
9397	00002D15	FB	<1>	STI			
9398	00002D16	E8B7010000	<1>	CALL	NOT_BUSY		; WAIT FOR CARD
9399	00002D1B	752B	<1>	JNZ	short CD_ERR		; BAD CARD
9400			<1>	;MOV	DX, HF_PORT+7		
9401	00002D1D	668B15[546B0000]	<1>	mov	dx, [HF_PORT]		
9402	00002D24	80C207	<1>	add	dl, 7		
9403	00002D27	B090	<1>	MOV	AL,DIAG_CMD		; START DIAGNOSE
9404	00002D29	EE	<1>	OUT	DX,AL		
9405	00002D2A	E8A3010000	<1>	CALL	NOT_BUSY		; WAIT FOR IT TO COMPLETE
9406	00002D2F	B480	<1>	MOV	AH,TIME_OUT		
9407	00002D31	7517	<1>	JNZ	short CD_EXIT		; TIME OUT ON DIAGNOSTIC
9408			<1>	;MOV	DX,HF_PORT+1		; GET ERROR REGISTER
9409	00002D33	668B15[546B0000]	<1>	mov	dx, [HF_PORT]		
9410	00002D3A	FEC2	<1>	inc	dl		
9411	00002D3C	EC	<1>	IN	AL,DX		
9412	00002D3D	A2[36710000]	<1>	MOV	[HF_ERROR],AL		; SAVE IT
9413	00002D42	B400	<1>	MOV	AH,0		
9414	00002D44	3C01	<1>	CMP	AL,1		; CHECK FOR ALL OK
9415	00002D46	7402	<1>	JE	SHORT CD_EXIT		
9416	00002D48	B420	<1>	CD_ERR: MOV	AH,BAD_CNTL		
9417			<1>	CD_EXIT:			
9418	00002D4A	8825[3F710000]	<1>	MOV	[DISK_STATUS1],AH		
9419	00002D50	C3	<1>	RETN			
9420			<1>				
9421			<1>	;-----			
9422			<1>	; COMMANDI :			
9423			<1>	; REPEATEDLY INPUTS DATA TILL :			
9424			<1>	; NSECTOR RETURNS ZERO :			
9425			<1>	;-----			
9426			<1>	COMMANDI:			
9427	00002D51	E85A020000	<1>	CALL	CHECK_DMA		; CHECK 64K BOUNDARY ERROR
9428	00002D56	724D	<1>	JC	short CMD_ABORT		
9429			<1>	;MOV	DI,BX		
9430	00002D58	89DF	<1>	mov	edi, ebx ; 21/02/2015		
9431	00002D5A	E8BE000000	<1>	CALL	COMMAND		; OUTPUT COMMAND
9432	00002D5F	7544	<1>	JNZ	short CMD_ABORT		
9433			<1>	CMD_I1:			
9434	00002D61	E830010000	<1>	CALL	_WAIT		; WAIT FOR DATA REQUEST INTERRUPT
9435	00002D66	753D	<1>	JNZ	short TM_OUT		; TIME OUT
9436			<1>	;MOV	CX,256		; SECTOR SIZE IN WORDS
9437	00002D68	B900010000	<1>	mov	ecx, 256 ; 21/02/2015		
9438			<1>	;MOV	DX,HF_PORT		
9439	00002D6D	668B15[546B0000]	<1>	mov	dx, [HF_PORT]		
9440	00002D74	FA	<1>	CLI			
9441	00002D75	FC	<1>	CLD			
9442	00002D76	F3666D	<1>	REP	INSW		; GET THE SECTOR
9443	00002D79	FB	<1>	STI			
9444	00002D7A	F645FE02	<1>	TEST	byte [CMD_BLOCK+6],ECC_MODE		; CHECK FOR NORMAL INPUT
9445	00002D7E	7419	<1>	JZ	CMD_I3		
9446	00002D80	E87A010000	<1>	CALL	WAIT_DRQ		; WAIT FOR DATA REQUEST
9447	00002D85	721E	<1>	JC	short TM_OUT		
9448			<1>	;MOV	DX,HF_PORT		
9449	00002D87	668B15[546B0000]	<1>	mov	dx, [HF_PORT]		
9450			<1>	;MOV	CX,4		; GET ECC BYTES
9451	00002D8E	B904000000	<1>	mov	ecx, 4 ; mov cx, 4		
9452	00002D93	EC	<1>	CMD_I2: IN	AL,DX		
9453			<1>	;MOV	[ES:DI],AL		; GO SLOW FOR BOARD
9454	00002D94	8807	<1>	mov	[edi], al ; 21/02/2015		
9455	00002D96	47	<1>	INC	eDI		
9456	00002D97	E2FA	<1>	LOOP	CMD_I2		
9457	00002D99	E88C010000	<1>	CMD_I3: CALL	CHECK_STATUS		
9458	00002D9E	7505	<1>	JNZ	short CMD_ABORT		; ERROR RETURNED

```

9459 00002DA0 FE4DF9      <1> DEC    byte [CMD_BLOCK+1]      ; CHECK FOR MORE
9460 00002DA3 75BC        <1> JNZ     SHORT CMD_I1
9461                      <1> CMD_ABORT:
9462 00002DA5 C3          <1> TM_OUT: RETn
9463                      <1>
9464                      <1> ;-----
9465                      <1> ; COMMANDO :
9466                      <1> ; REPEATEDLY OUTPUTS DATA TILL :
9467                      <1> ; NSECTOR RETURNS ZERO :
9468                      <1> ;-----
9469                      <1> COMMANDO:
9470 00002DA6 E805020000   <1> CALL    CHECK_DMA      ; CHECK 64K BOUNDARY ERROR
9471 00002DAB 72F8        <1> JC      short CMD_ABORT
9472 00002DAD 89DE        <1> CMD_OF: MOV    eSI,eBX ; 21/02/2015
9473 00002DAF E869000000   <1> CALL    COMMAND      ; OUTPUT COMMAND
9474 00002DB4 75EF        <1> JNZ     short CMD_ABORT
9475 00002DB6 E844010000   <1> CALL    WAIT_DRQ      ; WAIT FOR DATA REQUEST
9476 00002DBB 72E8        <1> JC      short TM_OUT   ; TOO LONG
9477                      <1> CMD_O1: ;PUSHDS
9478                      <1> ;PUSH ES      ; MOVE ES TO DS
9479                      <1> ;POP DS
9480                      <1> ;MOV CX,256      ; PUT THE DATA OUT TO THE CARD
9481                      <1> ;MOV DX,HF_PORT
9482                      <1> ; 01/02/2015
9483 00002DBD 668B15[546B0000] <1> mov     dx, [HF_PORT]
9484                      <1> ;push es
9485                      <1> ;pop ds
9486                      <1> ;mov cx, 256
9487 00002DC4 B900010000   <1> mov     ecx, 256 ; 21/02/2015
9488 00002DC9 FA          <1> CLI
9489 00002DCA FC          <1> CLD
9490 00002DCB F3666F      <1> REP     OUTSW
9491 00002DCE FB          <1> STI
9492                      <1> ;POP DS      ; RESTORE DS
9493 00002DCF F645FE02     <1> TEST    byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL OUTPUT
9494 00002DD3 7419        <1> JZ      short CMD_O3
9495 00002DD5 E825010000   <1> CALL    WAIT_DRQ      ; WAIT FOR DATA REQUEST
9496 00002DDA 72C9        <1> JC      short TM_OUT
9497                      <1> ;MOV DX,HF_PORT
9498 00002DDC 668B15[546B0000] <1> mov     dx, [HF_PORT]
9499                      <1> ;MOV CX,4      ; OUTPUT THE ECC BYTES
9500 00002DE3 B904000000   <1> mov     ecx, 4 ; mov cx, 4
9501                      <1> CMD_O2: ;MOV AL,[ES:SI]
9502 00002DE8 8A06        <1> mov     al, [esi]
9503 00002DEA EE          <1> OUT     DX,AL
9504 00002DEB 46          <1> INC     eSI
9505 00002DEC E2FA        <1> LOOP    CMD_O2
9506                      <1> CMD_O3:
9507 00002DEE E8A3000000   <1> CALL    _WAIT          ; WAIT FOR SECTOR COMPLETE INTERRUPT
9508 00002DF3 75B0        <1> JNZ     short TM_OUT   ; ERROR RETURNED
9509 00002DF5 E830010000   <1> CALL    CHECK_STATUS
9510 00002DFA 75A9        <1> JNZ     short CMD_ABORT
9511 00002DFC F605[35710000]08 <1> TEST    byte [HF_STATUS],ST_DRQ ; CHECK FOR MORE
9512 00002E03 75B8        <1> JNZ     SHORT CMD_O1
9513                      <1> ;MOV DX,HF_PORT+2      ; CHECK RESIDUAL SECTOR COUNT
9514 00002E05 668B15[546B0000] <1> mov     dx, [HF_PORT]
9515                      <1> ;add dl, 2
9516 00002E0C FEC2        <1> inc     dl
9517 00002E0E FEC2        <1> inc     dl
9518 00002E10 EC          <1> IN      AL,DX ;
9519 00002E11 A8FF        <1> TEST    AL,0FFH ;
9520 00002E13 7407        <1> JZ      short CMD_O4      ; COUNT = 0 OK
9521 00002E15 C605[3F710000]BB <1> MOV     byte [DISK_STATUS1],UNDEF_ERR
9522                      <1> ; OPERATION ABORTED - PARTIAL TRANSFER
9523                      <1> CMD_O4:
9524 00002E1C C3          <1> RETn
9525                      <1>
9526                      <1> ;-----
9527                      <1> ; COMMAND :
9528                      <1> ; THIS ROUTINE OUTPUTS THE COMMAND BLOCK :
9529                      <1> ; OUTPUT :
9530                      <1> ; BL = STATUS :
9531                      <1> ; BH = ERROR REGISTER :
9532                      <1> ;-----
9533                      <1>
9534                      <1> COMMAND:
9535 00002E1D 53          <1> PUSH    eBX      ; WAIT FOR SEEK COMPLETE AND READY
9536                      <1> ;;MOV CX,DELAY_2      ; SET INITIAL DELAY BEFORE TEST
9537                      <1> COMMAND1:
9538                      <1> ;;PUSH CX      ; SAVE LOOP COUNT
9539 00002E1E E87FFEFFFF     <1> CALL    TST_RDY      ; CHECK DRIVE READY
9540                      <1> ;;POP CX
9541 00002E23 7419        <1> JZ      short COMMAND2      ; DRIVE IS READY
9542 00002E25 803D[3F710000]80 <1> CMP     byte [DISK_STATUS1],TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
9543                      <1> ;JZ short CMD_TIMEOUT
9544                      <1> ;;LOOP COMMAND1      ; KEEP TRYING FOR A WHILE
9545                      <1> ;JMP SHORT COMMAND4      ; ITS NOT GOING TO GET READY
9546 00002E2C 7507        <1> jne     short COMMAND4
9547                      <1> CMD_TIMEOUT:
9548 00002E2E C605[3F710000]20 <1> MOV     byte [DISK_STATUS1],BAD_CNTL
9549                      <1> COMMAND4:
9550 00002E35 5B          <1> POP     eBX

```



```

9551 00002E36 803D[3F710000]00 <1> CMP byte [DISK_STATUS1],0 ; SET CONDITION CODE FOR CALLER
9552 00002E3D C3 <1> RETn
9553 <1> COMMAND2:
9554 00002E3E 5B <1> POP eBX
9555 00002E3F 57 <1> PUSH eDI
9556 00002E40 C605[37710000]00 <1> MOV byte [HF_INT_FLAG],0 ; RESET INTERRUPT FLAG
9557 00002E47 FA <1> CLI ; INHIBIT INTERRUPTS WHILE CHANGING MASK
9558 00002E48 E4A1 <1> IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
9559 <1> ;AND AL,0BFH
9560 00002E4A 243F <1> and al, 3Fh ; Enable IRQ 14 & 15
9561 <1> ;JMP $+2
9562 <1> IODELAY
9563 00002E4C EB00 <2> jmp short $+2
9564 00002E4E EB00 <2> jmp short $+2
9565 00002E50 E6A1 <1> OUT INTB01,AL
9566 00002E52 E421 <1> IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
9567 00002E54 24FB <1> AND AL,0FBH ; SECOND CHIP
9568 <1> ;JMP $+2
9569 <1> IODELAY
9570 00002E56 EB00 <2> jmp short $+2
9571 00002E58 EB00 <2> jmp short $+2
9572 00002E5A E621 <1> OUT INTA01,AL
9573 00002E5C FB <1> STI
9574 00002E5D 31FF <1> XOR eDI,eDI ; INDEX THE COMMAND TABLE
9575 <1> ;MOV DX,HF_PORT+1 ; DISK ADDRESS
9576 00002E5F 668B15[546B0000] <1> mov dx, [HF_PORT]
9577 00002E66 FEC2 <1> inc dl
9578 00002E68 F605[41710000]C0 <1> TEST byte [CONTROL_BYTE],0C0H ; CHECK FOR RETRY SUPPRESSION
9579 00002E6F 7411 <1> JZ short COMMAND3
9580 00002E71 8A45FE <1> MOV AL, [CMD_BLOCK+6] ; YES-GET OPERATION CODE
9581 00002E74 24F0 <1> AND AL,0F0H ; GET RID OF MODIFIERS
9582 00002E76 3C20 <1> CMP AL,20H ; 20H-40H IS READ, WRITE, VERIFY
9583 00002E78 7208 <1> JB short COMMAND3
9584 00002E7A 3C40 <1> CMP AL,40H
9585 00002E7C 7704 <1> JA short COMMAND3
9586 00002E7E 804DFE01 <1> OR byte [CMD_BLOCK+6],NO_RETRIES
9587 <1> ; VALID OPERATION FOR RETRY SUPPRESS
9588 <1> COMMAND3:
9589 00002E82 8A443DF8 <1> MOV AL,[CMD_BLOCK+eDI] ; GET THE COMMAND STRING BYTE
9590 00002E86 EE <1> OUT DX,AL ; GIVE IT TO CONTROLLER
9591 <1> IODELAY
9592 00002E87 EB00 <2> jmp short $+2
9593 00002E89 EB00 <2> jmp short $+2
9594 00002E8B 47 <1> INC eDI ; NEXT BYTE IN COMMAND BLOCK
9595 00002E8C 6642 <1> INC DX ; NEXT DISK ADAPTER REGISTER
9596 00002E8E 6683FF07 <1> cmp di, 7 ; 1/1/2015 ; ALL DONE?
9597 00002E92 75EE <1> JNZ short COMMAND3 ; NO--GO DO NEXT ONE
9598 00002E94 5F <1> POP eDI
9599 00002E95 C3 <1> RETn ; ZERO FLAG IS SET
9600 <1>
9601 <1> ;CMD_TIMEOUT:
9602 <1> ;MOV byte [DISK_STATUS1],BAD_CNTLRL
9603 <1> ;COMMAND4:
9604 <1> ;POP BX
9605 <1> ;CMP [DISK_STATUS1],0 ; SET CONDITION CODE FOR CALLER
9606 <1> ;RETn
9607 <1>
9608 <1> ;-----
9609 <1> ;WAIT FOR INTERRUPT :
9610 <1> ;-----
9611 <1> ;WAIT:
9612 <1> _WAIT:
9613 00002E96 FB <1> STI ; MAKE SURE INTERRUPTS ARE ON
9614 <1> ;SUB CX,CX ; SET INITIAL DELAY BEFORE TEST
9615 <1> ;CLC
9616 <1> ;MOV AX,9000H ; DEVICE WAIT INTERRUPT
9617 <1> ;INT 15H
9618 <1> ;JC WT2 ; DEVICE TIMED OUT
9619 <1> ;MOV BL,DELAY_1 ; SET DELAY COUNT
9620 <1>
9621 <1> ;mov bl, WAIT_HDU_INT_HI
9622 <1> ;; 21/02/2015
9623 <1> ;;mov bl, WAIT_HDU_INT_HI + 1
9624 <1> ;;mov cx, WAIT_HDU_INT_LO
9625 00002E97 B915160500 <1> mov ecx, WAIT_HDU_INT_LH
9626 <1> ; (AWARD BIOS -> WAIT_FOR_MEM)
9627 <1> ;----- WAIT LOOP
9628 <1>
9629 <1> WT1:
9630 <1> ;TEST byte [HF_INT_FLAG],80H ; TEST FOR INTERRUPT
9631 00002E9C F605[37710000]C0 <1> test byte [HF_INT_FLAG],0C0h
9632 <1> ;LOOPZ WT1
9633 00002EA3 7517 <1> JNZ short WT3 ; INTERRUPT--LETS GO
9634 <1> ;DEC BL
9635 <1> ;JNZ short WT1 ; KEEP TRYING FOR A WHILE
9636 <1>
9637 <1> WT1_hi:
9638 00002EA5 E461 <1> in al, SYS1 ; 61h (PORT_B) ; wait for lo to hi
9639 00002EA7 A810 <1> test al, 10h ; transition on memory
9640 00002EA9 75FA <1> jnz short WT1_hi ; refresh.
9641 <1> WT1_lo:
9642 00002EAB E461 <1> in al, SYS1 ; 061h (PORT_B)

```

9643	00002EAD	A810	<1>	test	al, 10h
9644	00002EAF	74FA	<1>	jz	short WT1_lo
9645	00002EB1	E2E9	<1>	loop	WT1
9646			<1>	;;or	bl, bl
9647			<1>	;;jz	short WT2
9648			<1>	;;dec	bl
9649			<1>	;;jmp	short WT1
9650			<1>	;dec	bl
9651			<1>	;jnz	short WT1
9652			<1>		
9653	00002EB3	C605[3F710000]80	<1>	WT2:	MOV byte [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
9654	00002EBA	EB0E	<1>	JMP	SHORT WT4
9655	00002EBC	C605[3F710000]00	<1>	WT3:	MOV byte [DISK_STATUS1],0
9656	00002EC3	C605[37710000]00	<1>	MOV	byte [HF_INT_FLAG],0
9657	00002ECA	803D[3F710000]00	<1>	WT4:	CMP byte [DISK_STATUS1],0 ; SET CONDITION CODE FOR CALLER
9658	00002ED1	C3	<1>	RETn	
9659			<1>		
9660			<1>	;	-----
9661			<1>	;	WAIT FOR CONTROLLER NOT BUSY :
9662			<1>	;	-----
9663			<1>	NOT_BUSY:	
9664	00002ED2	FB	<1>	STI	; MAKE SURE INTERRUPTS ARE ON
9665			<1>	;PUSH	eBX
9666			<1>	;SUB	CX,CX ; SET INITIAL DELAY BEFORE TEST
9667	00002ED3	668B15[546B0000]	<1>	mov	DX, [HF_PORT]
9668	00002EDA	80C207	<1>	add	dl, 7 ; Status port (HF_PORT+7)
9669			<1>	;MOV	BL,DELAY_1
9670			<1>		
9671			<1>	;mov	cx, WAIT_HDU_INT_LO ; 1615h
9672			<1>	;;mov	bl, WAIT_HDU_INT_HI ; 05h
9673			<1>	;mov	bl, WAIT_HDU_INT_HI + 1
9674	00002EDD	B915160500	<1>	mov	ecx, WAIT_HDU_INT_LH ; 21/02/2015
9675			<1>	;	
9676			<1>	;;	mov byte [wait_count], 0 ; Reset wait counter
9677			<1>	NB1:	
9678	00002EE2	EC	<1>	IN	AL,DX ; CHECK STATUS
9679			<1>	;TEST	AL,ST_BUSY
9680	00002EE3	2480	<1>	and	al, ST_BUSY
9681			<1>	;LOOPNZ	NB1
9682	00002EE5	7410	<1>	JZ	short NB2 ; NOT BUSY--LETS GO
9683			<1>	;DEC	BL
9684			<1>	;JNZ	short NB1 ; KEEP TRYING FOR A WHILE
9685			<1>		
9686	00002EE7	E461	<1>	NB1_hi:	IN AL,SYS1 ; wait for hi to lo
9687	00002EE9	A810	<1>	TEST	AL,010H ; transition on memory
9688	00002EEB	75FA	<1>	JNZ	SHORT NB1_hi ; refresh.
9689	00002EED	E461	<1>	NB1_lo:	IN AL,SYS1
9690	00002EEF	A810	<1>	TEST	AL,010H
9691	00002EF1	74FA	<1>	JZ	short NB1_lo
9692	00002EF3	E2ED	<1>	LOOP	NB1
9693			<1>	;dec	bl
9694			<1>	;jnz	short NB1
9695			<1>	;	
9696			<1>	;;	cmp byte [wait_count], 182 ; 10 seconds (182 timer ticks)
9697			<1>	;;	jb short NB1
9698			<1>	;	
9699			<1>	;MOV	[DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
9700			<1>	;JMP	SHORT NB3
9701	00002EF5	B080	<1>	mov	al, TIME_OUT
9702			<1>	NB2:	
9703			<1>	;MOV	byte [DISK_STATUS1],0
9704			<1>	;NB3:	
9705			<1>	;POP	eBX
9706	00002EF7	A2[3F710000]	<1>	mov	[DISK_STATUS1], al ;;; will be set after return
9707			<1>	;CMP	byte [DISK_STATUS1],0 ; SET CONDITION CODE FOR CALLER
9708	00002EFC	08C0	<1>	or	al, al ; (zf = 0 --> timeout)
9709	00002EFE	C3	<1>	RETn	
9710			<1>		
9711			<1>	;	-----
9712			<1>	;	WAIT FOR DATA REQUEST :
9713			<1>	;	-----
9714			<1>	WAIT_DRQ:	
9715			<1>	;MOV	CX,DELAY_3
9716			<1>	;MOV	DX,HF_PORT+7
9717	00002EFF	668B15[546B0000]	<1>	mov	dx, [HF_PORT]
9718	00002F06	80C207	<1>	add	dl, 7
9719			<1>	;;MOV	bl, WAIT_HDU_DRQ_HI ; 0
9720			<1>	;MOV	cx, WAIT_HDU_DRQ_LO ; 1000 (30 milli seconds)
9721			<1>		; (but it is written as 2000
9722			<1>		; micro seconds in ATORGS.ASM file
9723			<1>		; of Award Bios - 1999, D1A0622)
9724	00002F09	B9E8030000	<1>	mov	ecx, WAIT_HDU_DRQ_LH ; 21/02/2015
9725	00002F0E	EC	<1>	WQ_1:	IN AL,DX ; GET STATUS
9726	00002F0F	A808	<1>	TEST	AL,ST_DRQ ; WAIT FOR DRQ
9727	00002F11	7516	<1>	JNZ	short WQ_OK
9728			<1>	;LOOP	WQ_1 ; KEEP TRYING FOR A SHORT WHILE
9729			<1>	WQ_hi:	
9730	00002F13	E461	<1>	IN	AL,SYS1 ; wait for hi to lo
9731	00002F15	A810	<1>	TEST	AL,010H ; transition on memory
9732	00002F17	75FA	<1>	JNZ	SHORT WQ_hi ; refresh.
9733	00002F19	E461	<1>	WQ_lo:	IN AL,SYS1
9734	00002F1B	A810	<1>	TEST	AL,010H

9735	00002F1D	74FA	<1>	JZ	SHORT WQ_lo	
9736	00002F1F	E2ED	<1>	LOOP	WQ_1	
9737			<1>			
9738	00002F21	C605[3F710000]80	<1>	MOV	byte [DISK_STATUS1],TIME_OUT ;	ERROR
9739	00002F28	F9	<1>	STC		
9740			<1>	WQ_OK:		
9741	00002F29	C3	<1>	RETn		
9742			<1>	;WQ_OK:;CLC		
9743			<1>	;RETn		
9744			<1>			
9745			<1>	;-----		
9746			<1>	;CHECK FIXED DISK STATUS	:	
9747			<1>	;-----		
9748			<1>	CHECK_STATUS:		
9749	00002F2A	E813000000	<1>	CALL	CHECK_ST	; CHECK THE STATUS BYTE
9750	00002F2F	7509	<1>	JNZ	short CHECK_S1	; AN ERROR WAS FOUND
9751	00002F31	A801	<1>	TEST	AL,ST_ERROR	; WERE THERE ANY OTHER ERRORS
9752	00002F33	7405	<1>	JZ	short CHECK_S1	; NO ERROR REPORTED
9753	00002F35	E847000000	<1>	CALL	CHECK_ER	; ERROR REPORTED
9754			<1>	CHECK_S1:		
9755	00002F3A	803D[3F710000]00	<1>	CMP	byte [DISK_STATUS1],0	; SET STATUS FOR CALLER
9756	00002F41	C3	<1>	RETn		
9757			<1>			
9758			<1>	;-----		
9759			<1>	;CHECK FIXED DISK STATUS BYTE	:	
9760			<1>	;-----		
9761			<1>	CHECK_ST:		
9762			<1>	;MOV	DX,HF_PORT+7	; GET THE STATUS
9763	00002F42	668B15[546B0000]	<1>	mov	dx, [HF_PORT]	
9764	00002F49	80C207	<1>	add	dl, 7	
9765	00002F4C	EC	<1>	IN	AL,DX	
9766	00002F4D	A2[35710000]	<1>	MOV	[HF_STATUS],AL	
9767	00002F52	B400	<1>	MOV	AH,0	
9768	00002F54	A880	<1>	TEST	AL,ST_BUSY	; IF STILL BUSY
9769	00002F56	751A	<1>	JNZ	short CKST_EXIT	; REPORT OK
9770	00002F58	B4CC	<1>	MOV	AH,WRITE_FAULT	
9771	00002F5A	A820	<1>	TEST	AL,ST_WRT_FLT	; CHECK FOR WRITE FAULT
9772	00002F5C	7514	<1>	JNZ	short CKST_EXIT	
9773	00002F5E	B4AA	<1>	MOV	AH,NOT_RDY	
9774	00002F60	A840	<1>	TEST	AL,ST_READY	; CHECK FOR NOT READY
9775	00002F62	740E	<1>	JZ	short CKST_EXIT	
9776	00002F64	B440	<1>	MOV	AH,BAD_SEEK	
9777	00002F66	A810	<1>	TEST	AL,ST_SEEK_COMPL	; CHECK FOR SEEK NOT COMPLETE
9778	00002F68	7408	<1>	JZ	short CKST_EXIT	
9779	00002F6A	B411	<1>	MOV	AH,DATA_CORRECTED	
9780	00002F6C	A804	<1>	TEST	AL,ST_CORRCTD	; CHECK FOR CORRECTED ECC
9781	00002F6E	7502	<1>	JNZ	short CKST_EXIT	
9782	00002F70	B400	<1>	MOV	AH,0	
9783			<1>	CKST_EXIT:		
9784	00002F72	8825[3F710000]	<1>	MOV	[DISK_STATUS1],AH	; SET ERROR FLAG
9785	00002F78	80FC11	<1>	CMP	AH,DATA_CORRECTED	; KEEP GOING WITH DATA CORRECTED
9786	00002F7B	7403	<1>	JZ	short CKST_EX1	
9787	00002F7D	80FC00	<1>	CMP	AH,0	
9788			<1>	CKST_EX1:		
9789	00002F80	C3	<1>	RETn		
9790			<1>			
9791			<1>	;-----		
9792			<1>	;CHECK FIXED DISK ERROR REGISTER	:	
9793			<1>	;-----		
9794			<1>	CHECK_ER:		
9795			<1>	;MOV	DX, HF_PORT+1	; GET THE ERROR REGISTER
9796	00002F81	668B15[546B0000]	<1>	mov	dx, [HF_PORT]	
9797	00002F88	FEC2	<1>	inc	dl	
9798	00002F8A	EC	<1>	IN	AL,DX	
9799	00002F8B	A2[36710000]	<1>	MOV	[HF_ERROR],AL	
9800	00002F90	53	<1>	PUSH	ebx ; 21/02/2015	
9801	00002F91	B908000000	<1>	MOV	ecx,8	; TEST ALL 8 BITS
9802	00002F96	D0E0	<1>	CK1:	SHL AL,1	; MOVE NEXT ERROR BIT TO CARRY
9803	00002F98	7202	<1>	JC	short CK2	; FOUND THE ERROR
9804	00002F9A	E2FA	<1>	LOOP	CK1	; KEEP TRYING
9805	00002F9C	BB[486B0000]	<1>	CK2:	MOV ebx, ERR_TBL	; COMPUTE ADDRESS OF
9806	00002FA1	01CB	<1>	ADD	ebx,ecx	; ERROR CODE
9807			<1>	;MOV	AH,BYTE [CS:BX]	; GET ERROR CODE
9808			<1>	;mov	ah, [bx]	
9809	00002FA3	8A23	<1>	mov	ah, [ebx] ; 21/02/2015	
9810	00002FA5	8825[3F710000]	<1>	CKEX:	MOV [DISK_STATUS1],AH	; SAVE ERROR CODE
9811	00002FAB	5B	<1>	POP	ebx	
9812	00002FAC	80FC00	<1>	CMP	AH,0	
9813	00002FAF	C3	<1>	RETn		
9814			<1>			
9815			<1>	;-----		
9816			<1>	;CHECK DMA	:	
9817			<1>	; -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :		
9818			<1>	; FIT WITHOUT SEGMENT OVERFLOW.	:	
9819			<1>	; -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X :		
9820			<1>	; -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE) :		
9821			<1>	; -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) :		
9822			<1>	; -ERROR OTHERWISE :		
9823			<1>	;-----		
9824			<1>	CHECK_DMA:		
9825	00002FB0	6650	<1>	PUSH	AX	; SAVE REGISTERS
9826	00002FB2	66B80080	<1>	MOV	AX,8000H	; AH = MAX # SECTORS AL = MAX OFFSET

9827	00002FB6	F645FE02	<1>	TEST	byte [CMD_BLOCK+6],ECC_MODE
9828	00002FBA	7404	<1>	JZ	short CKD1
9829	00002FBC	66B8047F	<1>	MOV	AX,7F04H ; ECC IS 4 MORE BYTES
9830	00002FC0	3A65F9	<1>	CKD1:	CMP AH, [CMD_BLOCK+1] ; NUMBER OF SECTORS
9831	00002FC3	7706	<1>	JA	short CKDOK ; IT WILL FIT
9832	00002FC5	7208	<1>	JB	short CKDERR ; TOO MANY
9833	00002FC7	38D8	<1>	CMP	AL,BL ; CHECK OFFSET ON MAX SECTORS
9834	00002FC9	7204	<1>	JB	short CKDERR ; ERROR
9835	00002FCB	F8	<1>	CKDOK:	CLC ; CLEAR CARRY
9836	00002FCC	6658	<1>	POP	AX
9837	00002FCE	C3	<1>	RETn	; NORMAL RETURN
9838	00002FCF	F9	<1>	CKDERR:	STC ; INDICATE ERROR
9839	00002FD0	C605[3F710000]09	<1>	MOV	byte [DISK_STATUS1],DMA_BOUNDARY
9840	00002FD7	6658	<1>	POP	AX
9841	00002FD9	C3	<1>	RETn	
9842			<1>		
9843			<1>		;-----
9844			<1>		; SET UP ES:BX-> DISK PARMS :
9845			<1>		;-----
9846			<1>		
9847			<1>		; INPUT -> DL = 0 based drive number
9848			<1>		; OUTPUT -> ES:BX = disk parameter table address
9849			<1>		
9850			<1>	GET_VEC:	
9851			<1>	;SUB	AX,AX ; GET DISK PARAMETER ADDRESS
9852			<1>	;MOV	ES,AX
9853			<1>	;TEST	DL,1
9854			<1>	;JZ	short GV_0
9855			<1>	;LES	BX,[HF1_TBL_VEC] ; ES:BX -> DRIVE PARAMETERS
9856			<1>	;JMP	SHORT GV_EXIT
9857			<1>	;GV_0:	
9858			<1>	;LES	BX,[HF_TBL_VEC] ; ES:BX -> DRIVE PARAMETERS
9859			<1>		
9860			<1>	;xor	bh, bh
9861	00002FDA	31DB	<1>	xor	ebx, ebx
9862	00002FDC	88D3	<1>	mov	bl, dl
9863			<1>		; ;02/01/2015
9864			<1>	;shl	bl, 1 ; port address offset
9865			<1>	;mov	ax, [bx+hd_ports] ; Base port address (1F0h, 170h)
9866			<1>	;shl	bl, 1 ; dpt pointer offset
9867	00002FDE	C0E302	<1>	shl	bl, 2 ; ;
9868			<1>	;add	bx, HF_TBL_VEC ; Disk parameter table pointer
9869	00002FE1	81C3[44710000]	<1>	add	ebx, HF_TBL_VEC ; 21/02/2015
9870			<1>	;push	word [bx+2] ; dpt segment
9871			<1>	;pop	es
9872			<1>	;mov	bx, [bx] ; dpt offset
9873	00002FE7	8B1B	<1>	mov	ebx, [ebx]
9874			<1>	;GV_EXIT:	
9875	00002FE9	C3	<1>	RETn	
9876			<1>		
9877			<1>	hdcl_int:	; 21/02/2015
9878			<1>	;---	HARDWARE INT 76H -- ( IRQ LEVEL 14 ) -----
9879			<1>		:
9880			<1>		:
9881			<1>		:
9882			<1>		;-----
9883			<1>		
9884			<1>		; 22/12/2014
9885			<1>		; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD_INT)
9886			<1>		; '11/15/85'
9887			<1>		; AWARD BIOS 1999 (D1A0622)
9888			<1>		; Source Code - ATORGS.ASM (INT_HDISK, INT_HDISK1)
9889			<1>		
9890			<1>	;int_76h:	
9891			<1>	HD_INT:	
9892	00002FEA	6650	<1>	PUSH	AX
9893	00002FEC	1E	<1>	PUSH	DS
9894			<1>	;CALL	DDS
9895			<1>		; 21/02/2015 (32 bit, 386 pm modification)
9896	00002FED	66B81000	<1>	mov	ax, KDATA
9897	00002FF1	8ED8	<1>	mov	ds, ax
9898			<1>		
9899			<1>	;MOV	@HF_INT_FLAG,0FFh ; ALL DONE
9900			<1>	;mov	byte [CS:HF_INT_FLAG], 0FFh
9901	00002FF3	C605[37710000]FF	<1>	mov	byte [HF_INT_FLAG], 0FFh
9902			<1>		
9903	00002FFA	6652	<1>	push	dx
9904	00002FFC	66BAF701	<1>	mov	dx, HDC1_BASEPORT+7 ; Status Register (1F7h)
9905			<1>		; Clear Controller
9906			<1>	Clear_IRQ1415:	; (Award BIOS - 1999)
9907	00003000	EC	<1>	in	al, dx ;
9908	00003001	665A	<1>	pop	dx
9909			<1>	NEWIODELAY	
9910	00003003	E6EB	<2>	out	0ebh,al
9911			<1>		
9912	00003005	B020	<1>	MOV	AL,E0I ; NON-SPECIFIC END OF INTERRUPT
9913	00003007	E6A0	<1>	OUT	INTB00,AL ; FOR CONTROLLER #2
9914			<1>	;JMP	\$\$+2 ; WAIT
9915			<1>	NEWIODELAY	
9916	00003009	E6EB	<2>	out	0ebh,al
9917	0000300B	E620	<1>	OUT	INTA00,AL ; FOR CONTROLLER #1
9918	0000300D	1F	<1>	POP	DS

```

9919      <1> ;STI                      ; RE-ENABLE INTERRUPTS
9920      <1> ;MOV  AX,9100H          ; DEVICE POST
9921      <1> ;INT  15H              ; INTERRUPT
9922      <1> irq15_iret: ; 25/02/2015
9923 0000300E 6658      <1> POP  AX
9924 00003010 CF        <1> IRETD          ; RETURN FROM INTERRUPT
9925      <1>
9926      <1> hdc2_int: ; 21/02/2015
9927      <1> ;++++ HARDWARE INT 77H ++ ( IRQ LEVEL  15 ) ++++++
9928      <1> ;                      :
9929      <1> ;FIXED DISK INTERRUPT ROUTINE          :
9930      <1> ;                      :
9931      <1> ;+++++
9932      <1>
9933      <1> ;int_77h:
9934      <1> HD1_INT:
9935 00003011 6650      <1> PUSH  AX
9936      <1> ; Check if that is a spurious IRQ (from slave PIC)
9937      <1> ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
9938 00003013 B00B      <1> mov   al, 0Bh ; In-Service Register
9939 00003015 E6A0      <1> out   0A0h, al
9940 00003017 EB00      <1> jmp  short $+2
9941 00003019 EB00      <1> jmp  short $+2
9942 0000301B E4A0      <1> in    al, 0A0h
9943 0000301D 2480      <1> and   al, 80h ; bit 7 (is it real IRQ 15 or fake?)
9944 0000301F 74ED      <1> jz    short irq15_iret ; Fake (spurious)IRQ, do not send EOI)
9945      <1> ;
9946 00003021 1E        <1> PUSH  DS
9947      <1> ;CALL DDS
9948      <1> ; 21/02/2015 (32 bit, 386 pm modification)
9949 00003022 66B81000  <1> mov   ax, KDATA
9950 00003026 8ED8      <1> mov   ds, ax
9951      <1> ;
9952      <1> ;;MOV @HF_INT_FLAG,0FFH ; ALL DONE
9953      <1> ;or      byte [CS:HF_INT_FLAG],0C0h
9954 00003028 800D[37710000]C0 <1> or    byte [HF_INT_FLAG], 0C0h
9955      <1> ;
9956 0000302F 6652      <1> push  dx
9957 00003031 66BA7701  <1> mov   dx, HDC2_BASEPORT+7 ; Status Register (177h)
9958      <1> ; Clear Controller (Award BIOS 1999)
9959 00003035 EBC9      <1> jmp   short Clear_IRQ1415
9960      <1>
9961      <1>
9962      <1> ;%include 'diskdata.inc' ; 11/03/2015
9963      <1> ;%include 'diskbss.inc' ; 11/03/2015
9964      <1>
9965      <1>
9966      <1> ;////////////////////
9967      <1> ;; END OF DISK I/O SYTEM ///
9968      <1> %include 'memory.inc' ; 09/03/2015
9969      <1> ; MEMORY.ASM - Retro UNIX 386 v1 MEMORY MANAGEMENT FUNCTIONS (PROCEDURES)
9970      <1> ; Retro UNIX 386 v1 Kernel (unix386.s, v0.2.0.14) - MEMORY.INC
9971      <1> ; Last Modification: 18/10/2015 (!not completed!)
9972      <1> ;
9973      <1> ; Source code for NASM - Netwide Assembler (2.11)
9974      <1>
9975      <1> ; ////////// MEMORY MANAGEMENT FUNCTIONS (PROCEDURES) //////////
9976      <1>
9977      <1> ;;04/11/2014 (unix386.s)
9978      <1> ;PDE_A_PRESENT equ 1 ; Present flag for PDE
9979      <1> ;PDE_A_WRITE equ 2 ; Writable (write permission) flag
9980      <1> ;PDE_A_USER equ 4 ; User (non-system/kernel) page flag
9981      <1> ;;
9982      <1> ;PTE_A_PRESENT equ 1 ; Present flag for PTE (bit 0)
9983      <1> ;PTE_A_WRITE equ 2 ; Writable (write permission) flag (bit 1)
9984      <1> ;PTE_A_USER equ 4 ; User (non-system/kernel) page flag (bit 2)
9985      <1> ;PTE_A_ACCESS equ 32 ; Accessed flag (bit 5) ; 09/03/2015
9986      <1>
9987      <1> ; 27/04/2015
9988      <1> ; 09/03/2015
9989      <1> PAGE_SIZE equ 4096 ; page size in bytes
9990      <1> PAGE_SHIFT equ 12 ; page table shift count
9991      <1> PAGE_D_SHIFT equ 22 ; 12 + 10 ; page directory shift count
9992      <1> PAGE_OFF equ 0FFFh ; 12 bit byte offset in page frame
9993      <1> PTE_MASK equ 03FFh ; page table entry mask
9994      <1> PTE_DUPLICATED equ 200h ; duplicated page sign (AVL bit 0)
9995      <1> PDE_A_CLEAR equ 0F000h ; to clear PDE attribute bits
9996      <1> PTE_A_CLEAR equ 0F000h ; to clear PTE attribute bits
9997      <1> LOGIC_SECT_SIZE equ 512 ; logical sector size
9998      <1> ERR_MAJOR_PF equ 0E0h ; major error: page fault
9999      <1> ERR_MINOR_IM equ 1 ; insufficient (out of) memory
10000      <1> ERR_MINOR_DS equ 2 ; disk read/write error
10001      <1> ERR_MINOR_PV equ 3 ; protection violation
10002      <1> SWP_DISK_READ_ERR equ 4
10003      <1> SWP_DISK_NOT_PRESENT_ERR equ 5
10004      <1> SWP_SECTOR_NOT_PRESENT_ERR equ 6
10005      <1> SWP_NO_FREE_SPACE_ERR equ 7
10006      <1> SWP_DISK_WRITE_ERR equ 8
10007      <1> SWP_NO_PAGE_TO_SWAP_ERR equ 9
10008      <1> PTE_A_ACCESS_BIT equ 5 ; Bit 5 (accessed flag)
10009      <1> SECTOR_SHIFT equ 3 ; sector shift (to convert page block number)
10010      <1>

```

```

10011 <1> ;
10012 <1> ;; Retro Unix 386 v1 - paging method/principles
10013 <1> ;;
10014 <1> ;; 10/10/2014
10015 <1> ;; RETRO UNIX 386 v1 - PAGING METHOD/PRINCIPLES
10016 <1> ;;
10017 <1> ;; KERNEL PAGE MAP: 1 to 1 physical memory page map
10018 <1> ;; (virtual address = physical address)
10019 <1> ;; KERNEL PAGE TABLES:
10020 <1> ;; Kernel page directory and all page tables are
10021 <1> ;; on memory as initialized, as equal to physical memory
10022 <1> ;; layout. Kernel pages can/must not be swapped out/in.
10023 <1> ;;
10024 <1> ;; what for: User pages may be swapped out, when accessing
10025 <1> ;; a page in kernel/system mode, if it would be swapped out,
10026 <1> ;; kernel would have to swap it in! But it is also may be
10027 <1> ;; in use by a user process. (In system/kernel mode
10028 <1> ;; kernel can access all memory pages even if they are
10029 <1> ;; reserved/allocated for user processes. Swap out/in would
10030 <1> ;; cause conflicts.)
10031 <1> ;;
10032 <1> ;; As result of these conditions,
10033 <1> ;; all kernel pages must be initialized as equal to
10034 <1> ;; physical layout for preventing page faults.
10035 <1> ;; Also, calling "allocate page" procedure after
10036 <1> ;; a page fault can cause another page fault (double fault)
10037 <1> ;; if all kernel page tables would not be initialized.
10038 <1> ;;
10039 <1> ;; [first_page] = Beginning of users space, as offset to
10040 <1> ;; memory allocation table. (double word aligned)
10041 <1> ;;
10042 <1> ;; [next_page] = first/next free space to be searched
10043 <1> ;; as offset to memory allocation table. (dw aligned)
10044 <1> ;;
10045 <1> ;; [last_page] = End of memory (users space), as offset
10046 <1> ;; to memory allocation table. (double word aligned)
10047 <1> ;;
10048 <1> ;; USER PAGE TABLES:
10049 <1> ;; Demand paging (& 'copy on write' allocation method) ...
10050 <1> ;; 'ready only' marked copies of the
10051 <1> ;; parent process's page table entries (for
10052 <1> ;; same physical memory).
10053 <1> ;; (A page will be copied to a new page after
10054 <1> ;; if it causes R/W page fault.)
10055 <1> ;;
10056 <1> ;; Every user process has own (different)
10057 <1> ;; page directory and page tables.
10058 <1> ;;
10059 <1> ;; Code starts at virtual address 0, always.
10060 <1> ;; (Initial value of EIP is 0 in user mode.)
10061 <1> ;; (Programs can be written/developed as simple
10062 <1> ;; flat memory programs.)
10063 <1> ;;
10064 <1> ;; MEMORY ALLOCATION STRATEGY:
10065 <1> ;; Memory page will be allocated by kernel only
10066 <1> ;; (in kernel/system mode only).
10067 <1> ;; * After a
10068 <1> ;; - 'not present' page fault
10069 <1> ;; - 'writing attempt on read only page' page fault
10070 <1> ;; * For loading (opening, reading) a file or disk/drive
10071 <1> ;; * As response to 'allocate additional memory blocks'
10072 <1> ;; request by running process.
10073 <1> ;; * While creating a process, allocating a new buffer,
10074 <1> ;; new page tables etc.
10075 <1> ;;
10076 <1> ;; At first,
10077 <1> ;; - 'allocate page' procedure will be called;
10078 <1> ;; if it will return with a valid (>0) physical address
10079 <1> ;; (that means the relevant M.A.T. bit has been RESET)
10080 <1> ;; relevant memory page/block will be cleared (zeroed).
10081 <1> ;; - 'allocate page' will be called for allocating page
10082 <1> ;; directory, page table and running space (data/code).
10083 <1> ;; - every successful 'allocate page' call will decrease
10084 <1> ;; 'free_pages' count (pointer).
10085 <1> ;; - 'out of (insufficient) memory error' will be returned
10086 <1> ;; if 'free_pages' points to a ZERO.
10087 <1> ;; - swapping out and swapping in (if it is not a new page)
10088 <1> ;; procedures will be called as response to 'out of memory'
10089 <1> ;; error except errors caused by attribute conflicts.
10090 <1> ;; (swapper functions)
10091 <1> ;;
10092 <1> ;; At second,
10093 <1> ;; - page directory entry will be updated then page table
10094 <1> ;; entry will be updated.
10095 <1> ;;
10096 <1> ;; MEMORY ALLOCATION TABLE FORMAT:
10097 <1> ;; - M.A.T. has a size according to available memory as
10098 <1> ;; follows:
10099 <1> ;; - 1 (allocation) bit per 1 page (4096 bytes)
10100 <1> ;; - a bit with value of 0 means allocated page
10101 <1> ;; - a bit with value of 1 means a free page
10102 <1> ;; - 'free_pages' pointer holds count of free pages

```

```

10103 <1> ;; depending on M.A.T.
10104 <1> ;; (NOTE: Free page count will not be checked
10105 <1> ;; again -on M.A.T.- after initialization.
10106 <1> ;; Kernel will trust on initial count.)
10107 <1> ;; - 'free_pages' count will be decreased by allocation
10108 <1> ;; and it will be increased by deallocation procedures.
10109 <1> ;;
10110 <1> ;; - Available memory will be calculated during
10111 <1> ;; the kernel's initialization stage (in real mode).
10112 <1> ;; Memory allocation table and kernel page tables
10113 <1> ;; will be formatted/sized as result of available
10114 <1> ;; memory calculation before paging is enabled.
10115 <1> ;;
10116 <1> ;; For 4GB Available/Present Memory: (max. possible memory size)
10117 <1> ;; - Memory Allocation Table size will be 128 KB.
10118 <1> ;; - Memory allocation for kernel page directory size
10119 <1> ;; is always 4 KB. (in addition to total allocation size
10120 <1> ;; for page tables)
10121 <1> ;; - Memory allocation for kernel page tables (1024 tables)
10122 <1> ;; is 4 MB (1024*4*1024 bytes).
10123 <1> ;; - User (available) space will be started
10124 <1> ;; at 6th MB of the memory (after 1MB+4MB).
10125 <1> ;; - The first 640 KB is for kernel's itself plus
10126 <1> ;; memory allocation table and kernel's page directory
10127 <1> ;; (D0000h-EFFFFh may be used as kernel space...)
10128 <1> ;; - B0000h to B7FFFh address space (32 KB) will be used
10129 <1> ;; for buffers.
10130 <1> ;; - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
10131 <1> ;; (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
10132 <1> ;; - Kernel page tables start at 100000h (2nd MB)
10133 <1> ;;
10134 <1> ;; For 1GB Available Memory:
10135 <1> ;; - Memory Allocation Table size will be 32 KB.
10136 <1> ;; - Memory allocation for kernel page directory size
10137 <1> ;; is always 4 KB. (in addition to total allocation size
10138 <1> ;; for page tables)
10139 <1> ;; - Memory allocation for kernel page tables (256 tables)
10140 <1> ;; is 1 MB (256*4*1024 bytes).
10141 <1> ;; - User (available) space will be started
10142 <1> ;; at 3th MB of the memory (after 1MB+1MB).
10143 <1> ;; - The first 640 KB is for kernel's itself plus
10144 <1> ;; memory allocation table and kernel's page directory
10145 <1> ;; (D0000h-EFFFFh may be used as kernel space...)
10146 <1> ;; - B0000h to B7FFFh address space (32 KB) will be used
10147 <1> ;; for buffers.
10148 <1> ;; - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
10149 <1> ;; (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
10150 <1> ;; - Kernel page tables start at 100000h (2nd MB).
10151 <1> ;;
10152 <1> ;;
10153 <1>
10154 <1>
10155 <1>
; *****
10156 <1> ;;
10157 <1> ;; RETRO UNIX 386 v1 - Paging (Method for Copy On Write paging principle)
10158 <1> ;; DEMAND PAGING - PARENT&CHILD PAGE TABLE DUPLICATION PRINCIPLES (23/04/2015)
10159 <1>
10160 <1> ;; Main factor: "sys fork" system call
10161 <1> ;;
10162 <1> ;; FORK
10163 <1> ;; |----> parent - duplicated PTEs, read only pages
10164 <1> ;; |writable pages ---->|
10165 <1> ;; |----> child - duplicated PTEs, read only pages
10166 <1> ;;
10167 <1> ;; AVL bit (0) of Page Table Entry is used as duplication sign
10168 <1> ;;
10169 <1> ;; AVL Bit 0 [PTE Bit 9] = 'Duplicated PTE belongs to child' sign/flag (if it is
set)
10170 <1> ;; Note: Dirty bit (PTE bit 6) may be used instead of AVL bit 0 (PTE bit 9)
10171 <1> ;; -while R/W bit is 0-.
10172 <1> ;;
10173 <1> ;; Duplicate page tables with writable pages (the 1st sys fork in the process):
10174 <1> ;; # Parent's Page Table Entries are updated to point same pages as read only,
10175 <1> ;; as duplicated PTE bit -AVL bit 0, PTE bit 9- are reset/clear.
10176 <1> ;; # Then Parent's Page Table is copied to Child's Page Table.
10177 <1> ;; # Child's Page Table Entries are updated as duplicated child bit
10178 <1> ;; -AVL bit 0, PTE bit 9- is set.
10179 <1> ;;
10180 <1> ;; Duplicate page tables with read only pages (several sys fork system calls):
10181 <1> ;; # Parent's read only pages are copied to new child pages.
10182 <1> ;; Parent's PTE attributes are not changed.
10183 <1> ;; (Because, there is another parent-child fork before this fork! We must not
10184 <1> ;; destroy/mix previous fork result).
10185 <1> ;; # Child's Page Table Entries (which are corresponding to Parent's
10186 <1> ;; read only pages) are set as writable (while duplicated PTE bit is clear).
10187 <1> ;; # Parent's PTEs with writable page attribute are updated to point same pages
10188 <1> ;; as read only, (while) duplicated PTE bit is reset (clear).
10189 <1> ;; # Parent's Page Table Entries (with writable page attribute) are duplicated
10190 <1> ;; as Child's Page Table Entries without copying actual page.
10191 <1> ;; # Child 's Page Table Entries (which are corresponding to Parent's writable
10192 <1> ;; pages) are updated as duplicated PTE bit (AVL bit 0, PTE bit 9- is set.

```

```

10193 <1> ;;
10194 <1> ;; !? WHAT FOR (duplication after duplication):
10195 <1> ;; In UNIX method for sys fork (a typical 'fork' application in /etc/init)
10196 <1> ;; program/executable code continues from specified location as child process,
10197 <1> ;; returns back previous code location as parent process, every child after
10198 <1> ;; every sys fork uses last image of code and data just prior the fork.
10199 <1> ;; Even if the parent code changes data, the child will not see the changed data
10200 <1> ;; after the fork. In Retro UNIX 8086 v1, parent's process segment (32KB)
10201 <1> ;; was copied to child's process segment (all of code and data) according to
10202 <1> ;; original UNIX v1 which copies all of parent process code and data -core-
10203 <1> ;; to child space -core- but swaps that core image -of child- on to disk.
10204 <1> ;; If I (Erdogan Tan) would use a method of to copy parent's core
10205 <1> ;; (complete running image of parent process) to the child process;
10206 <1> ;; for big sizes, i would force Retro UNIX 386 v1 to spend many memory pages
10207 <1> ;; and times only for a sys fork. (It would excessive reservation for sys fork,
10208 <1> ;; because sys fork usually is prior to sys exec; sys exec always establishes
10209 <1> ;; a new/fresh core -running space-, by clearing all code/data content).
10210 <1> ;; 'Read Only' page flag ensures page fault handler is needed only for a few write
10211 <1> ;; attempts between sys fork and sys exec, not more... (I say so by thinking
10212 <1> ;; of "/etc/init" content, specially.) sys exec will clear page tables and
10213 <1> ;; new/fresh pages will be used to load and run new executable/program.
10214 <1> ;; That is what for i have preferred "copy on write", "duplication" method
10215 <1> ;; for sharing same read only pages between parent and child processes.
10216 <1> ;; That is a pity i have to use new private flag (AVL bit 0, "duplicated PTE
10217 <1> ;; belongs to child" sign) for cooperation on duplicated pages between a parent
10218 <1> ;; and it's child processes; otherwise parent process would destroy data belongs
10219 <1> ;; to its child or vice versa; or some pages would remain unclaimed
10220 <1> ;; -deallocation problem-.
10221 <1> ;; Note: to prevent conflicts, read only pages must not be swapped out...
10222 <1> ;;
10223 <1> ;; WHEN PARENT TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
10224 <1> ;; # Page fault handler will do those:
10225 <1> ;; - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
10226 <1> ;; - If it is reset/clear, there is a child uses same page.
10227 <1> ;; - Parent's read only page -previous page- is copied to a new writable page.
10228 <1> ;; - Parent's PTE is updated as writable page, as unique page (AVL=0)
10229 <1> ;; - (Page fault handler will check this PTE later, if child process causes to
10230 <1> ;; page fault due to write attempt on read only page. Of course, the previous
10231 <1> ;; read only page will be converted to writable and unique page which belongs
10232 <1> ;; to child process.)
10233 <1> ;; WHEN CHILD TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
10234 <1> ;; # Page fault handler will do those:
10235 <1> ;; - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
10236 <1> ;; - If it is set, there is a parent uses -or was using- same page.
10237 <1> ;; - Same PTE address within parent's page table is checked if it has same page
10238 <1> ;; address or not.
10239 <1> ;; - If parent's PTE has same address, child will continue with a new writable
page.
10240 <1> ;; Parent's PTE will point to same (previous) page as writable, unique
(AVL=0) .
10241 <1> ;; - If parent's PTE has different address, child will continue with it's
10242 <1> ;; own/same page but read only flag (0) will be changed to writable flag (1)
and
10243 <1> ;; 'duplicated PTE (belongs to child)' flag/sign will be cleared/reset.

10244 <1> ;;
10245 <1> ;; NOTE: When a child process is terminated, read only flags of parent's page
tables
10246 <1> ;; will be set as writable (and unique) in case of child process was using
10247 <1> ;; same pages with duplicated child PTE sign... Depending on sys fork and
10248 <1> ;; duplication method details, it is not possible multiple child processes
10249 <1> ;; were using same page with duplicated PTEs.
10250 <1> ;;
10251 <1>
; *****
10252 <1>
10253 <1> ;; 08/10/2014
10254 <1> ;; 11/09/2014 - Retro UNIX 386 v1 PAGING (further) draft
10255 <1> ;; by Erdogan Tan (Based on KolibriOS 'memory.inc')
10256 <1>
10257 <1> ;; 'allocate_page' code is derived and modified from KolibriOS
10258 <1> ;; 'alloc_page' procedure in 'memory.inc'
10259 <1> ;; (25/08/2014, Revision: 5057) file
10260 <1> ;; by KolibriOS Team (2004-2012)
10261 <1>
10262 <1> allocate_page:
10263 <1> ; 01/07/2015
10264 <1> ; 05/05/2015
10265 <1> ; 30/04/2015
10266 <1> ; 16/10/2014
10267 <1> ; 08/10/2014
10268 <1> ; 09/09/2014 (Retro UNIX 386 v1 - beginning)
10269 <1> ;
10270 <1> ; INPUT -> none
10271 <1> ;
10272 <1> ; OUTPUT ->
10273 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
10274 <1> ; (corresponding MEMORY ALLOCATION TABLE bit is RESET)
10275 <1> ;
10276 <1> ; CF = 1 and EAX = 0
10277 <1> ; if there is not a free page to be allocated
10278 <1> ;

```



10279		<1>	; Modified Registers -> none (except EAX)
10280		<1>	;
10281	00003037 A1[B0700000]	<1>	mov eax, [free_pages]
10282	0000303C 21C0	<1>	and eax, eax
10283	0000303E 7438	<1>	jz short out_of_memory
10284		<1>	;
10285	00003040 53	<1>	push ebx
10286	00003041 51	<1>	push ecx
10287		<1>	;
10288	00003042 BB00001000	<1>	mov ebx, MEM_ALLOC_TBL ; Memory Allocation Table offset
10289	00003047 89D9	<1>	mov ecx, ebx
10290		<1>	;
10291		<1>	; NOTE: 32 (first_page) is initial
10292		<1>	; value of [next_page].
10293		<1>	; It points to the first available
10294		<1>	; page block for users (ring 3) ...
10295		<1>	; (MAT offset 32 = 1024/32)
10296		<1>	; (at the of the first 4 MB)
10296	00003049 031D[B4700000]	<1>	add ebx, [next_page] ; Free page searching starts from here
10297		<1>	; next_free_page >> 5
10298	0000304F 030D[B8700000]	<1>	add ecx, [last_page] ; Free page searching ends here
10299		<1>	; (total_pages - 1) >> 5
10300		<1>	al_p_scan:
10301	00003055 39CB	<1>	cmp ebx, ecx
10302	00003057 770A	<1>	ja short al_p_notfound
10303		<1>	;
10304		<1>	; 01/07/2015
10305		<1>	; AMD64 Architecture Programmer's Manual
10306		<1>	; Volume 3:
10307		<1>	; General-Purpose and System Instructions
10308		<1>	;
10309		<1>	; BSF - Bit Scan Forward
10310		<1>	;
10311		<1>	; Searches the value in a register or a memory location
10312		<1>	; (second operand) for the least-significant set bit.
10313		<1>	; If a set bit is found, the instruction clears the zero flag (ZF)
10314		<1>	; and stores the index of the least-significant set bit in a destination
10315		<1>	; register (first operand). If the second operand contains 0,
10316		<1>	; the instruction sets ZF to 1 and does not change the contents of the
10317		<1>	; destination register. The bit index is an unsigned offset from bit 0
10318		<1>	; of the searched value
10319		<1>	;
10320	00003059 0FBC03	<1>	bsf eax, [ebx] ; Scans source operand for first bit set (1).
10321		<1>	; Clear ZF if a bit is found set (1) and
10322		<1>	; loads the destination with an index to
10323		<1>	; first set bit. (0 -> 31)
10324		<1>	; Sets ZF to 1 if no bits are found set.
10325	0000305C 7525	<1>	jnz short al_p_found ; ZF = 0 -> a free page has been found
10326		<1>	;
10327		<1>	; NOTE: a Memory Allocation Table bit
10328		<1>	; with value of 1 means
10329		<1>	; the corresponding page is free
10330		<1>	; (Retro UNIX 386 v1 feaure only!)
10331	0000305E 83C304	<1>	add ebx, 4
10332		<1>	; We return back for searching next page block
10333		<1>	; NOTE: [free_pages] is not ZERO; so,
10334		<1>	; we always will find at least 1 free page here.
10335	00003061 EBF2	<1>	jmp short al_p_scan
10336		<1>	;
10337		<1>	al_p_notfound:
10338	00003063 81E900001000	<1>	sub ecx, MEM_ALLOC_TBL
10339	00003069 890D[B4700000]	<1>	mov [next_page], ecx ; next/first free page = last page
10340		<1>	; (deallocate_page procedure will change it)
10341	0000306F 31C0	<1>	xor eax, eax
10342	00003071 A3[B0700000]	<1>	mov [free_pages], eax ; 0
10343	00003076 59	<1>	pop ecx
10344	00003077 5B	<1>	pop ebx
10345		<1>	;
10346		<1>	out_of_memory:
10347	00003078 E857040000	<1>	call swap_out
10348	0000307D 7325	<1>	jnc short al_p_ok ; [free_pages] = 0, re-allocation by swap_out
10349		<1>	;
10350	0000307F 29C0	<1>	sub eax, eax ; 0
10351	00003081 F9	<1>	stc
10352	00003082 C3	<1>	ret
10353		<1>	;
10354		<1>	al_p_found:
10355	00003083 89D9	<1>	mov ecx, ebx
10356	00003085 81E900001000	<1>	sub ecx, MEM_ALLOC_TBL
10357	0000308B 890D[B4700000]	<1>	mov [next_page], ecx ; Set first free page searching start
10358		<1>	; address/offset (to the next)
10359	00003091 FF0D[B0700000]	<1>	dec dword [free_pages] ; 1 page has been allocated (X = X-1)
10360		<1>	;
10361	00003097 0FB303	<1>	btr [ebx], eax ; The destination bit indexed by the source value
10362		<1>	; is copied into the Carry Flag and then cleared
10363		<1>	; in the destination.
10364		<1>	;
10365		<1>	; Reset the bit which is corresponding to the
10366		<1>	; (just) allocated page.
10367		<1>	; 01/07/2015 (4*8 = 32, 1 allocation byte = 8 pages)
10368	0000309A C1E103	<1>	shl ecx, 3 ; (page block offset * 32) + page index
10369	0000309D 01C8	<1>	add eax, ecx ; = page number
10370	0000309F C1E00C	<1>	shl eax, 12 ; physical address of the page (flat/real value)

10371		<1>	; EAX = physical address of memory page
10372		<1>	;
10373		<1>	; NOTE: The relevant page directory and page table entry will be updated
10374		<1>	; according to this EAX value...
10375	000030A2 59	<1>	pop ecx
10376	000030A3 5B	<1>	pop ebx
10377		<1>	al_p_ok:
10378	000030A4 C3	<1>	retn
10379		<1>	
10380		<1>	
10381		<1>	make_page_dir:
10382		<1>	; 18/04/2015
10383		<1>	; 12/04/2015
10384		<1>	; 23/10/2014
10385		<1>	; 16/10/2014
10386		<1>	; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
10387		<1>	;
10388		<1>	; INPUT ->
10389		<1>	; none
10390		<1>	; OUTPUT ->
10391		<1>	; (EAX = 0)
10392		<1>	; cf = 1 -> insufficient (out of) memory error
10393		<1>	; cf = 0 ->
10394		<1>	; u.pgdir = page directory (physical) address of the current
10395		<1>	; process/user.
10396		<1>	;
10397		<1>	; Modified Registers -> EAX
10398		<1>	;
10399	000030A5 E88DFFFFFF	<1>	call allocate_page
10400	000030AA 7216	<1>	jc short mkpd_error
10401		<1>	;
10402	000030AC A3[D3740000]	<1>	mov [u.pgdir], eax ; Page dir address for current user/process
10403		<1>	; (Physical address)
10404		<1>	clear_page:
10405		<1>	; 18/04/2015
10406		<1>	; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
10407		<1>	;
10408		<1>	; INPUT ->
10409		<1>	; EAX = physical address of the page
10410		<1>	; OUTPUT ->
10411		<1>	; all bytes of the page will be cleared
10412		<1>	;
10413		<1>	; Modified Registers -> none
10414		<1>	;
10415	000030B1 57	<1>	push edi
10416	000030B2 51	<1>	push ecx
10417	000030B3 50	<1>	push eax
10418	000030B4 B900040000	<1>	mov ecx, PAGE_SIZE / 4
10419	000030B9 89C7	<1>	mov edi, eax
10420	000030BB 31C0	<1>	xor eax, eax
10421	000030BD F3AB	<1>	rep stosd
10422	000030BF 58	<1>	pop eax
10423	000030C0 59	<1>	pop ecx
10424	000030C1 5F	<1>	pop edi
10425		<1>	mkpd_error:
10426		<1>	mkpt_error:
10427	000030C2 C3	<1>	retn
10428		<1>	
10429		<1>	make_page_table:
10430		<1>	; 23/06/2015
10431		<1>	; 18/04/2015
10432		<1>	; 12/04/2015
10433		<1>	; 16/10/2014
10434		<1>	; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
10435		<1>	;
10436		<1>	; INPUT ->
10437		<1>	; EBX = virtual (linear) address
10438		<1>	; ECX = page table attributes (lower 12 bits)
10439		<1>	; (higher 20 bits must be ZERO)
10440		<1>	; (bit 0 must be 1)
10441		<1>	; u.pgdir = page directory (physical) address
10442		<1>	; OUTPUT ->
10443		<1>	; EDX = Page directory entry address
10444		<1>	; EAX = Page table address
10445		<1>	; cf = 1 -> insufficient (out of) memory error
10446		<1>	; cf = 0 -> page table address in the PDE (EDX)
10447		<1>	;
10448		<1>	; Modified Registers -> EAX, EDX
10449		<1>	;
10450	000030C3 E86FFFFFFF	<1>	call allocate_page
10451	000030C8 72F8	<1>	jc short mkpt_error
10452	000030CA E811000000	<1>	call set_pde
10453	000030CF EBE0	<1>	jmp short clear_page
10454		<1>	
10455		<1>	make_page:
10456		<1>	; 24/07/2015
10457		<1>	; 23/06/2015 ; (Retro UNIX 386 v1 - beginning)
10458		<1>	;
10459		<1>	; INPUT ->
10460		<1>	; EBX = virtual (linear) address
10461		<1>	; ECX = page attributes (lower 12 bits)
10462		<1>	; (higher 20 bits must be ZERO)

10463		<1>	;	(bit 0 must be 1)
10464		<1>	;	u.pgdir = page directory (physical) address
10465		<1>	;	OUTPUT ->
10466		<1>	;	EBX = Virtual address
10467		<1>	;	(EDX = PTE value)
10468		<1>	;	EAX = Physical address
10469		<1>	;	cf = 1 -> insufficient (out of) memory error
10470		<1>	;	
10471		<1>	;	Modified Registers -> EAX, EDX
10472		<1>	;	
10473	000030D1 E861FFFFFF	<1>	call	allocate_page
10474	000030D6 7207	<1>	jc	short mkp_err
10475	000030D8 E821000000	<1>	call	set_pte
10476	000030DD 73D2	<1>	jnc	short clear_page ; 18/04/2015
10477		<1>	mkp_err:	
10478	000030DF C3	<1>	retn	
10479		<1>		
10480		<1>		
10481		<1>	set_pde:	; Set page directory entry (PDE)
10482		<1>	;	20/07/2015
10483		<1>	;	18/04/2015
10484		<1>	;	12/04/2015
10485		<1>	;	23/10/2014
10486		<1>	;	10/10/2014 ; (Retro UNIX 386 v1 - beginning)
10487		<1>	;	
10488		<1>	;	INPUT ->
10489		<1>	;	EAX = physical address
10490		<1>	;	(use present value if EAX = 0)
10491		<1>	;	EBX = virtual (linear) address
10492		<1>	;	ECX = page table attributes (lower 12 bits)
10493		<1>	;	(higher 20 bits must be ZERO)
10494		<1>	;	(bit 0 must be 1)
10495		<1>	;	u.pgdir = page directory (physical) address
10496		<1>	;	OUTPUT ->
10497		<1>	;	EDX = PDE address
10498		<1>	;	EAX = page table address (physical)
10499		<1>	;	;(CF=1 -> Invalid page address)
10500		<1>	;	
10501		<1>	;	Modified Registers -> EDX
10502		<1>	;	
10503	000030E0 89DA	<1>	mov	edx, ebx
10504	000030E2 C1EA16	<1>	shr	edx, PAGE_D_SHIFT ; 22
10505	000030E5 C1E202	<1>	shl	edx, 2 ; offset to page directory (1024*4)
10506	000030E8 0315[D3740000]	<1>	add	edx, [u.pgdir]
10507		<1>	;	
10508	000030EE 21C0	<1>	and	eax, eax
10509	000030F0 7506	<1>	jnz	short spde_1
10510		<1>	;	
10511	000030F2 8B02	<1>	mov	eax, [edx] ; old PDE value
10512		<1>	;test	al, 1
10513		<1>	;jz	short spde_2
10514	000030F4 662500F0	<1>	and	ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
10515		<1>	spde_1:	
10516		<1>	;and	cx, 0FFFh
10517	000030F8 8902	<1>	mov	[edx], eax
10518	000030FA 66090A	<1>	or	[edx], cx
10519	000030FD C3	<1>	retn	
10520		<1>	;	spde_2: ; error
10521		<1>	;	stc
10522		<1>	;	retn
10523		<1>		
10524		<1>	set_pte:	; Set page table entry (PTE)
10525		<1>	;	24/07/2015
10526		<1>	;	20/07/2015
10527		<1>	;	23/06/2015
10528		<1>	;	18/04/2015
10529		<1>	;	12/04/2015
10530		<1>	;	10/10/2014 ; (Retro UNIX 386 v1 - beginning)
10531		<1>	;	
10532		<1>	;	INPUT ->
10533		<1>	;	EAX = physical page address
10534		<1>	;	(use present value if EAX = 0)
10535		<1>	;	EBX = virtual (linear) address
10536		<1>	;	ECX = page attributes (lower 12 bits)
10537		<1>	;	(higher 20 bits must be ZERO)
10538		<1>	;	(bit 0 must be 1)
10539		<1>	;	u.pgdir = page directory (physical) address
10540		<1>	;	OUTPUT ->
10541		<1>	;	EAX = physical page address
10542		<1>	;	(EDX = PTE value)
10543		<1>	;	EBX = virtual address
10544		<1>	;	
10545		<1>	;	CF = 1 -> error
10546		<1>	;	
10547		<1>	;	Modified Registers -> EAX, EDX
10548		<1>	;	
10549	000030FE 50	<1>	push	eax
10550	000030FF A1[D3740000]	<1>	mov	eax, [u.pgdir] ; 20/07/2015
10551	00003104 E837000000	<1>	call	get_pde
10552		<1>	;	EDX = PDE address
10553		<1>	;	EAX = PDE value
10554	00003109 5A	<1>	pop	edx ; physical page address

10555	0000310A	722A	<1>	jc	short	spte_err	; PDE not present
10556			<1>				
10557	0000310C	53	<1>	push	ebx		; 24/07/2015
10558	0000310D	662500F0	<1>	and	ax	PDE_A_CLEAR	; 0F000h ; clear lower 12 bits
10559			<1>				; EDX = PT address (physical)
10560	00003111	C1EB0C	<1>	shr	ebx	PAGE_SHIFT	; 12
10561	00003114	81E3FF030000	<1>	and	ebx	PTE_MASK	; 03FFh
10562			<1>				; clear higher 10 bits (PD bits)
10563	0000311A	C1E302	<1>	shl	ebx	2	; offset to page table (1024*4)
10564	0000311D	01C3	<1>	add	ebx	eax	
10565			<1>				
10566	0000311F	8B03	<1>	mov	eax	[ebx]	; Old PTE value
10567	00003121	A801	<1>	test	al	1	
10568	00003123	740C	<1>	jz	short	spte_0	
10569	00003125	09D2	<1>	or	edx	edx	
10570	00003127	750F	<1>	jnz	short	spte_1	
10571	00003129	662500F0	<1>	and	ax	PTE_A_CLEAR	; 0F000h ; clear lower 12 bits
10572	0000312D	89C2	<1>	mov	edx	eax	
10573	0000312F	EB09	<1>	jmp	short	spte_2	
10574			<1>		spte_0:		
10575			<1>				; If this PTE contains a swap (disk) address,
10576			<1>				; it can be updated by using 'swap_in' procedure
10577			<1>				; only!
10578	00003131	21C0	<1>	and	eax	eax	
10579	00003133	7403	<1>	jz	short	spte_1	
10580			<1>				; 24/07/2015
10581			<1>				; swapped page ! (on disk)
10582	00003135	5B	<1>	pop	ebx		
10583			<1>		spte_err:		
10584	00003136	F9	<1>	stc			
10585	00003137	C3	<1>	retn			
10586			<1>		spte_1:		
10587	00003138	89D0	<1>	mov	eax	edx	
10588			<1>		spte_2:		
10589	0000313A	09CA	<1>	or	edx	ecx	
10590			<1>				; 23/06/2015
10591	0000313C	8913	<1>	mov	[ebx]	edx	; PTE value in EDX
10592			<1>				; 24/07/2015
10593	0000313E	5B	<1>	pop	ebx		
10594	0000313F	C3	<1>	retn			
10595			<1>				
10596			<1>		get_pde:		; Get present value of the relevant PDE
10597			<1>				; 20/07/2015
10598			<1>				; 18/04/2015
10599			<1>				; 12/04/2015
10600			<1>				; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
10601			<1>				
10602			<1>				; INPUT ->
10603			<1>				; EBX = virtual (linear) address
10604			<1>				; EAX = page directory (physical) address
10605			<1>				; OUTPUT ->
10606			<1>				; EDX = Page directory entry address
10607			<1>				; EAX = Page directory entry value
10608			<1>				; CF = 1 -> PDE not present or invalid ?
10609			<1>				; Modified Registers -> EDX, EAX
10610			<1>				
10611	00003140	89DA	<1>	mov	edx	ebx	
10612	00003142	C1EA16	<1>	shr	edx	PAGE_D_SHIFT	; 22 (12+10)
10613	00003145	C1E202	<1>	shl	edx	2	; offset to page directory (1024*4)
10614	00003148	01C2	<1>	add	edx	eax	; page directory address (physical)
10615	0000314A	8B02	<1>	mov	eax	[edx]	
10616	0000314C	A801	<1>	test	al	PDE_A_PRESENT	; page table is present or not !
10617	0000314E	751F	<1>	jnz	short	gpde_retn	
10618	00003150	F9	<1>	stc			
10619			<1>		gpde_retn:		
10620	00003151	C3	<1>	retn			
10621			<1>				
10622			<1>		get_pte:		
10623			<1>				; Get present value of the relevant PTE
10624			<1>				; 29/07/2015
10625			<1>				; 20/07/2015
10626			<1>				; 18/04/2015
10627			<1>				; 12/04/2015
10628			<1>				; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
10629			<1>				
10630			<1>				; INPUT ->
10631			<1>				; EBX = virtual (linear) address
10632			<1>				; EAX = page directory (physical) address
10633			<1>				; OUTPUT ->
10634			<1>				; EDX = Page table entry address (if CF=0)
10635			<1>				; Page directory entry address (if CF=1)
10636			<1>				; (Bit 0 value is 0 if PT is not present)
10637			<1>				; EAX = Page table entry value (page address)
10638			<1>				; CF = 1 -> PDE not present or invalid ?
10639			<1>				; Modified Registers -> EAX, EDX
10640			<1>				
10641	00003152	E8E9FFFFFF	<1>	call	get_pde		
10642	00003157	72F8	<1>	jc	short	gpde_retn	; page table is not present
10643			<1>		jnc	short	gpde_1
10644			<1>				;retn
10645			<1>			gpde_1:	
10646	00003159	662500F0	<1>	and	ax	PDE_A_CLEAR	; 0F000h ; clear lower 12 bits

10647	0000315D	89DA	<1>	mov	edx, ebx
10648	0000315F	C1EA0C	<1>	shr	edx, PAGE_SHIFT ; 12
10649	00003162	81E2FF030000	<1>	and	edx, PTE_MASK ; 03FFh
10650			<1>		; clear higher 10 bits (PD bits)
10651	00003168	C1E202	<1>	shl	edx, 2 ; offset from start of page table (1024*4)
10652	0000316B	01C2	<1>	add	edx, eax
10653	0000316D	8B02	<1>	mov	eax, [edx]
10654			<1>	gpte_retn:	
10655	0000316F	C3	<1>	retn	
10656			<1>		
10657			<1>	deallocate_page_dir:	
10658			<1>		; 15/09/2015
10659			<1>		; 05/08/2015
10660			<1>		; 30/04/2015
10661			<1>		; 28/04/2015
10662			<1>		; 17/10/2014
10663			<1>		; 12/10/2014 (Retro UNIX 386 v1 - beginning)
10664			<1>		;
10665			<1>		; INPUT ->
10666			<1>		; EAX = PHYSICAL ADDRESS OF THE PAGE DIRECTORY (CHILD)
10667			<1>		; EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
10668			<1>		; OUTPUT ->
10669			<1>		; All of page tables in the page directory
10670			<1>		; and page dir's itself will be deallocated
10671			<1>		; except 'read only' duplicated pages (will be converted
10672			<1>		; to writable pages).
10673			<1>		;
10674			<1>		; Modified Registers -> EAX
10675			<1>		;
10676			<1>		;
10677	00003170	56	<1>	push	esi
10678	00003171	51	<1>	push	ecx
10679	00003172	50	<1>	push	eax
10680	00003173	89C6	<1>	mov	esi, eax
10681	00003175	31C9	<1>	xor	ecx, ecx
10682			<1>		; The 1st PDE points to Kernel Page Table 0 (the 1st 4MB),
10683			<1>		; it must not be deallocated
10684	00003177	890E	<1>	mov	[esi], ecx ; 0 ; clear PDE 0
10685			<1>	dapd_0:	
10686	00003179	AD	<1>	lodsd	
10687	0000317A	A801	<1>	test	al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
10688	0000317C	7409	<1>	jz	short dapd_1
10689	0000317E	662500F0	<1>	and	ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
10690	00003182	E812000000	<1>	call	deallocate_page_table
10691			<1>	dapd_1:	
10692	00003187	41	<1>	inc	ecx ; page directory entry index
10693	00003188	81F900040000	<1>	cmp	ecx, PAGE_SIZE / 4 ; 1024
10694	0000318E	72E9	<1>	jb	short dapd_0
10695			<1>	dapd_2:	
10696	00003190	58	<1>	pop	eax
10697	00003191	E879000000	<1>	call	deallocate_page ; deallocate the page dir's itself
10698	00003196	59	<1>	pop	ecx
10699	00003197	5E	<1>	pop	esi
10700	00003198	C3	<1>	retn	
10701			<1>		
10702			<1>	deallocate_page_table:	
10703			<1>		; 19/09/2015
10704			<1>		; 15/09/2015
10705			<1>		; 05/08/2015
10706			<1>		; 30/04/2015
10707			<1>		; 28/04/2015
10708			<1>		; 24/10/2014
10709			<1>		; 23/10/2014
10710			<1>		; 12/10/2014 (Retro UNIX 386 v1 - beginning)
10711			<1>		;
10712			<1>		; INPUT ->
10713			<1>		; EAX = PHYSICAL (real/flat) ADDRESS OF THE PAGE TABLE
10714			<1>		; EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
10715			<1>		; (ECX = page directory entry index)
10716			<1>		; OUTPUT ->
10717			<1>		; All of pages in the page table and page table's itself
10718			<1>		; will be deallocated except 'read only' duplicated pages
10719			<1>		; (will be converted to writable pages).
10720			<1>		;
10721			<1>		; Modified Registers -> EAX
10722			<1>		;
10723	00003199	56	<1>	push	esi
10724	0000319A	57	<1>	push	edi
10725	0000319B	52	<1>	push	edx
10726	0000319C	50	<1>	push	eax ; *
10727	0000319D	89C6	<1>	mov	esi, eax
10728	0000319F	31FF	<1>	xor	edi, edi ; 0
10729			<1>	dapt_0:	
10730	000031A1	AD	<1>	lodsd	
10731	000031A2	A801	<1>	test	al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
10732	000031A4	7441	<1>	jz	short dapt_1
10733			<1>		;
10734	000031A6	A802	<1>	test	al, PTE_A_WRITE ; bit 1, writable (r/w) flag
10735			<1>		; (must be 1)
10736	000031A8	754C	<1>	jnz	short dapt_3
10737			<1>		; Read only -duplicated- page (belongs to a parent or a child)
10738	000031AA	66A90002	<1>	test	ax, PTE_DUPLICATED ; Was this page duplicated

10739		<1>		; as child's page ?
10740	000031AE 744B	<1>	jz	short dapt_4 ; Clear PTE but don't deallocate the page!
10741		<1>		; check the parent's PTE value is read only & same page or not..
10742		<1>		; ECX = page directory entry index (0-1023)
10743	000031B0 53	<1>	push	ebx
10744	000031B1 51	<1>	push	ecx
10745	000031B2 66C1E102	<1>	shl	cx, 2 ; *4
10746	000031B6 01CB	<1>	add	ebx, ecx ; PDE offset (for the parent)
10747	000031B8 8B0B	<1>	mov	ecx, [ebx]
10748	000031BA F6C101	<1>	test	cl, PDE_A_PRESENT ; present (valid) or not ?
10749	000031BD 7435	<1>	jz	short dapt_2 ; parent process does not use this page
10750	000031BF 6681E100F0	<1>	and	cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
10751		<1>		; EDI = page table entry index (0-1023)
10752	000031C4 89FA	<1>	mov	edx, edi
10753	000031C6 66C1E202	<1>	shl	dx, 2 ; *4
10754	000031CA 01CA	<1>	add	edx, ecx ; PTE offset (for the parent)
10755	000031CC 8B1A	<1>	mov	ebx, [edx]
10756	000031CE F6C301	<1>	test	bl, PTE_A_PRESENT ; present or not ?
10757	000031D1 7421	<1>	jz	short dapt_2 ; parent process does not use this page
10758	000031D3 662500F0	<1>	and	ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
10759	000031D7 6681E300F0	<1>	and	bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
10760	000031DC 39D8	<1>	cmp	eax, ebx ; parent's and child's pages are same ?
10761	000031DE 7514	<1>	jne	short dapt_2 ; not same page
10762		<1>		; deallocate the child's page
10763	000031E0 800A02	<1>		or byte [edx], PTE_A_WRITE ; convert to writable page (parent)
10764	000031E3 59	<1>	pop	ecx
10765	000031E4 5B	<1>	pop	ebx
10766	000031E5 EB14	<1>	jmp	short dapt_4
10767		<1>	dapt_1:	
10768	000031E7 09C0	<1>	or	eax, eax ; swapped page ?
10769	000031E9 7417	<1>	jz	short dapt_5 ; no
10770		<1>		; yes
10771	000031EB D1E8	<1>	shr	eax, 1
10772	000031ED E848040000	<1>	call	unlink_swap_block ; Deallocate swapped page block
10773		<1>		; on the swap disk (or in file)
10774	000031F2 EB0E	<1>	jmp	short dapt_5
10775		<1>	dapt_2:	
10776	000031F4 59	<1>	pop	ecx
10777	000031F5 5B	<1>	pop	ebx
10778		<1>	dapt_3:	
10779		<1>		;and ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
10780	000031F6 E814000000	<1>	call	deallocate_page
10781		<1>	dapt_4:	
10782	000031FB C746FC00000000	<1>	mov	dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
10783		<1>	dapt_5:	
10784	00003202 47	<1>	inc	edi ; page table entry index
10785	00003203 81FF00040000	<1>	cmp	edi, PAGE_SIZE / 4 ; 1024
10786	00003209 7296	<1>	jb	short dapt_0
10787		<1>		;
10788	0000320B 58	<1>	pop	eax ; *
10789	0000320C 5A	<1>	pop	edx
10790	0000320D 5F	<1>	pop	edi
10791	0000320E 5E	<1>	pop	esi
10792		<1>		;
10793		<1>		;call deallocate_page ; deallocate the page table's itself
10794		<1>		;retn
10795		<1>		
10796		<1>	deallocate_page:	
10797		<1>		; 15/09/2015
10798		<1>		; 28/04/2015
10799		<1>		; 10/03/2015
10800		<1>		; 17/10/2014
10801		<1>		; 12/10/2014 (Retro UNIX 386 v1 - beginning)
10802		<1>		;
10803		<1>		; INPUT ->
10804		<1>		; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
10805		<1>		; OUTPUT ->
10806		<1>		; [free_pages] is increased
10807		<1>		; (corresponding MEMORY ALLOCATION TABLE bit is SET)
10808		<1>		; CF = 1 if the page is already deallocated
10809		<1>		; (or not allocated) before.
10810		<1>		;
10811		<1>		; Modified Registers -> EAX
10812		<1>		;
10813	0000320F 53	<1>	push	ebx
10814	00003210 52	<1>	push	edx
10815		<1>		;
10816	00003211 C1E80C	<1>	shr	eax, PAGE_SHIFT ; shift physical address to
10817		<1>		; 12 bits right
10818		<1>		; to get page number
10819	00003214 89C2	<1>	mov	edx, eax
10820		<1>		; 15/09/2015
10821	00003216 C1EA03	<1>	shr	edx, 3 ; to get offset to M.A.T.
10822		<1>		; (1 allocation bit = 1 page)
10823		<1>		; (1 allocation bytes = 8 pages)
10824	00003219 80E2FC	<1>	and	dl, 0FCh ; clear lower 2 bits
10825		<1>		; (to get 32 bit position)
10826		<1>		;
10827	0000321C BB00001000	<1>	mov	ebx, MEM_ALLOC_TBL ; Memory Allocation Table address
10828	00003221 01D3	<1>	add	ebx, edx
10829	00003223 83E01F	<1>	and	eax, 1Fh ; lower 5 bits only
10830		<1>		; (allocation bit position)

```

10831 00003226 3B15[B4700000] <1> cmp     edx, [next_page]      ; is the new free page address lower
10832                                <1>                                ; than the address in 'next_page' ?
10833                                <1>                                ; (next/first free page value)
10834 0000322C 7306 <1> jnb     short dap_1      ; no
10835 0000322E 8915[B4700000] <1> mov     [next_page], edx      ; yes
10836 <1> dap_1:
10837 00003234 0FAB03 <1> bts     [ebx], eax          ; unlink/release/deallocate page
10838                                <1>                                ; set relevant bit to 1.
10839                                <1>                                ; set CF to the previous bit value
10840                                <1> ;cmc                                ; complement carry flag
10841                                <1> ;jc     short dap_2      ; do not increase free_pages count
10842                                <1>                                ; if the page is already deallocated
10843                                <1>                                ; before.
10844 00003237 FF05[B0700000] <1> inc     dword [free_pages]
10845 <1> dap_2:
10846 0000323D 5A <1> pop     edx
10847 0000323E 5B <1> pop     ebx
10848 0000323F C3 <1> retn
10849 <1>
10850 <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10851 <1> ;;
10852 <1> ;; Copyright (C) KolibriOS team 2004-2012. All rights reserved. ;;
10853 <1> ;; Distributed under terms of the GNU General Public License ;;
10854 <1> ;;
10855 <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10856 <1>
10857 <1> ;;$Revision: 5057 $
10858 <1>
10859 <1>
10860 <1> ;;align 4
10861 <1> ;;proc alloc_page
10862 <1>
10863 <1> ;;      pushfd
10864 <1> ;;      cli
10865 <1> ;;      push     ebx
10866 <1> ;;;//-
10867 <1> ;;      cmp     [pg_data.pages_free], 1
10868 <1> ;;      jle     .out_of_memory
10869 <1> ;;;//-
10870 <1> ;;
10871 <1> ;;      mov     ebx, [page_start]
10872 <1> ;;      mov     ecx, [page_end]
10873 <1> ;;;.l1:
10874 <1> ;;      bsf     eax, [ebx];
10875 <1> ;;      jnz     .found
10876 <1> ;;      add     ebx, 4
10877 <1> ;;      cmp     ebx, ecx
10878 <1> ;;      jb     .l1
10879 <1> ;;      pop     ebx
10880 <1> ;;      popfd
10881 <1> ;;      xor     eax, eax
10882 <1> ;;      ret
10883 <1> ;;;.found:
10884 <1> ;;;//-
10885 <1> ;;      dec     [pg_data.pages_free]
10886 <1> ;;      jz     .out_of_memory
10887 <1> ;;;//-
10888 <1> ;;      btr     [ebx], eax
10889 <1> ;;      mov     [page_start], ebx
10890 <1> ;;      sub     ebx, sys_pgmap
10891 <1> ;;      lea     eax, [eax+ebx*8]
10892 <1> ;;      shl     eax, 12
10893 <1> ;;;//-      dec [pg_data.pages_free]
10894 <1> ;;      pop     ebx
10895 <1> ;;      popfd
10896 <1> ;;      ret
10897 <1> ;;;//-
10898 <1> ;;;.out_of_memory:
10899 <1> ;;      mov     [pg_data.pages_free], 1
10900 <1> ;;      xor     eax, eax
10901 <1> ;;      pop     ebx
10902 <1> ;;      popfd
10903 <1> ;;      ret
10904 <1> ;;;//-
10905 <1> ;;endp
10906 <1>
10907 <1> duplicate_page_dir:
10908 <1> ; 21/09/2015
10909 <1> ; 31/08/2015
10910 <1> ; 20/07/2015
10911 <1> ; 28/04/2015
10912 <1> ; 27/04/2015
10913 <1> ; 18/04/2015
10914 <1> ; 12/04/2015
10915 <1> ; 18/10/2014
10916 <1> ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
10917 <1> ;
10918 <1> ; INPUT ->
10919 <1> ; [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
10920 <1> ; page directory.
10921 <1> ; OUTPUT ->
10922 <1> ; EAX = PHYSICAL (real/flat) ADDRESS of the child's

```

10923		<1>	;	page directory.
10924		<1>	;	(New page directory with new page table entries.)
10925		<1>	;	(New page tables with read only copies of the parent's
10926		<1>	;	pages.)
10927		<1>	;	EAX = 0 -> Error (CF = 1)
10928		<1>	;	
10929		<1>	;	Modified Registers -> none (except EAX)
10930		<1>	;	
10931	00003240 E8F2FDFFFF	<1>	call	allocate_page
10932	00003245 723E	<1>	jc	short dpd_err
10933		<1>	;	
10934	00003247 55	<1>	push	ebp ; 20/07/2015
10935	00003248 56	<1>	push	esi
10936	00003249 57	<1>	push	edi
10937	0000324A 53	<1>	push	ebx
10938	0000324B 51	<1>	push	ecx
10939	0000324C 8B35[D3740000]	<1>	mov	esi, [u.pgdir]
10940	00003252 89C7	<1>	mov	edi, eax
10941	00003254 50	<1>	push	eax ; save child's page directory address
10942		<1>	;	31/08/2015
10943		<1>	;	copy PDE 0 from the parent's page dir to the child's page dir
10944		<1>	;	(use same system space for all user page tables)
10945	00003255 A5	<1>	movsd	
10946	00003256 BD00004000	<1>	mov	ebp, 1024*4096 ; pass the 1st 4MB (system space)
10947	0000325B B9FF030000	<1>	mov	ecx, (PAGE_SIZE / 4) - 1 ; 1023
10948		<1>	dpd_0:	
10949	00003260 AD	<1>	lodsd	
10950		<1>	;	or eax, eax
10951		<1>	;	jnz short dpd_1
10952	00003261 A801	<1>	test	al, PDE_A_PRESENT ; bit 0 = 1
10953	00003263 7508	<1>	jnz	short dpd_1
10954		<1>	;	20/07/2015 (virtual address at the end of the page table)
10955	00003265 81C500004000	<1>	add	ebp, 1024*4096 ; page size * PTE count
10956	0000326B EB0F	<1>	jmp	short dpd_2
10957		<1>	dpd_1:	
10958	0000326D 662500F0	<1>	and	ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
10959	00003271 89C3	<1>	mov	ebx, eax
10960		<1>	;	EBX = Parent's page table address
10961	00003273 E81F000000	<1>	call	duplicate_page_table
10962	00003278 720C	<1>	jc	short dpd_p_err
10963		<1>	;	EAX = Child's page table address
10964	0000327A 0C07	<1>	or	al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
10965		<1>	;	set bit 0, bit 1 and bit 2 to 1
10966		<1>	;	(present, writable, user)
10967		<1>	dpd_2:	
10968	0000327C AB	<1>	stosd	
10969	0000327D E2E1	<1>	loop	dpd_0
10970		<1>	;	
10971	0000327F 58	<1>	pop	eax ; restore child's page directory address
10972		<1>	dpd_3:	
10973	00003280 59	<1>	pop	ecx
10974	00003281 5B	<1>	pop	ebx
10975	00003282 5F	<1>	pop	edi
10976	00003283 5E	<1>	pop	esi
10977	00003284 5D	<1>	pop	ebp ; 20/07/2015
10978		<1>	dpd_err:	
10979	00003285 C3	<1>	retn	
10980		<1>	dpd_p_err:	
10981		<1>	;	release the allocated pages missing (recover free space)
10982	00003286 58	<1>	pop	eax ; the new page directory address (physical)
10983	00003287 8B1D[D3740000]	<1>	mov	ebx, [u.pgdir] ; parent's page directory address
10984	0000328D E8DEFEFFFF	<1>	call	deallocate_page_dir
10985	00003292 29C0	<1>	sub	eax, eax ; 0
10986	00003294 F9	<1>	stc	
10987	00003295 EBE9	<1>	jmp	short dpd_3
10988		<1>		
10989		<1>	duplicate_page_table:	
10990		<1>	;	21/09/2015
10991		<1>	;	20/07/2015
10992		<1>	;	05/05/2015
10993		<1>	;	28/04/2015
10994		<1>	;	27/04/2015
10995		<1>	;	18/04/2015
10996		<1>	;	18/10/2014
10997		<1>	;	16/10/2014 (Retro UNIX 386 v1 - beginning)
10998		<1>	;	
10999		<1>	;	INPUT ->
11000		<1>	;	EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
11001		<1>	;	EBP = page table entry index (from 'duplicate_page_dir')
11002		<1>	;	OUTPUT ->
11003		<1>	;	EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
11004		<1>	;	(with 'read only' attribute of page table entries)
11005		<1>	;	EBP = (recent) page table index (for 'add_to_swap_queue')
11006		<1>	;	CF = 1 -> error
11007		<1>	;	
11008		<1>	;	Modified Registers -> EBP (except EAX)
11009		<1>	;	
11010	00003297 E89BFDEFFF	<1>	call	allocate_page
11011	0000329C 726A	<1>	jc	short dpt_err
11012		<1>	;	
11013	0000329E 50	<1>	push	eax ; *
11014	0000329F 56	<1>	push	esi



11015	000032A0	57	<1>	push	edi
11016	000032A1	52	<1>	push	edx
11017	000032A2	51	<1>	push	ecx
11018			<1>	;	
11019	000032A3	89DE	<1>	mov	esi, ebx
11020	000032A5	89C7	<1>	mov	edi, eax
11021	000032A7	89C2	<1>	mov	edx, eax
11022	000032A9	81C200100000	<1>	add	edx, PAGE_SIZE
11023			<1>	dpt_0:	
11024	000032AF	AD	<1>	lodsd	
11025	000032B0	21C0	<1>	and	eax, eax
11026	000032B2	7444	<1>	jz	short dpt_3
11027	000032B4	A801	<1>	test	al, PTE_A_PRESENT ; bit 0 = 1
11028	000032B6	7507	<1>	jnz	short dpt_1
11029			<1>	;	20/07/2015
11030			<1>	;	ebp = virtual (linear) address of the memory page
11031	000032B8	E887040000	<1>	call	reload_page ; 28/04/2015
11032	000032BD	7244	<1>	jc	short dpt_p_err
11033			<1>	dpt_1:	
11034			<1>	;	21/09/2015
11035	000032BF	89C1	<1>	mov	ecx, eax
11036	000032C1	662500F0	<1>	and	ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
11037	000032C5	F6C102	<1>	test	cl, PTE_A_WRITE ; writable page ?
11038	000032C8	7525	<1>	jnz	short dpt_2
11039			<1>	;	Read only (parent) page
11040			<1>	;	- there is a third process which uses this page -
11041			<1>	;	Allocate a new page for the child process
11042	000032CA	E868FDFFFF	<1>	call	allocate_page
11043	000032CF	7232	<1>	jc	short dpt_p_err
11044	000032D1	57	<1>	push	edi
11045	000032D2	56	<1>	push	esi
11046	000032D3	89CE	<1>	mov	esi, ecx
11047	000032D5	89C7	<1>	mov	edi, eax
11048	000032D7	B900040000	<1>	mov	ecx, PAGE_SIZE/4
11049	000032DC	F3A5	<1>	rep	movsd ; copy page (4096 bytes)
11050	000032DE	5E	<1>	pop	esi
11051	000032DF	5F	<1>	pop	edi
11052			<1>	;	
11053	000032E0	53	<1>	push	ebx
11054	000032E1	50	<1>	push	eax
11055			<1>	;	20/07/2015
11056	000032E2	89EB	<1>	mov	ebx, ebp
11057			<1>	;	ebx = virtual address of the memory page
11058	000032E4	E80B030000	<1>	call	add_to_swap_queue
11059	000032E9	58	<1>	pop	eax
11060	000032EA	5B	<1>	pop	ebx
11061			<1>	;	21/09/2015
11062	000032EB	0C07	<1>	or	al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
11063			<1>	;	user + writable + present page
11064	000032ED	EB09	<1>	jmp	short dpt_3
11065			<1>	dpt_2:	
11066			<1>	;	or ax, PTE_A_USER+PTE_A_PRESENT
11067	000032EF	0C05	<1>	or	al, PTE_A_USER+PTE_A_PRESENT
11068			<1>	;	(read only page!)
11069	000032F1	8946FC	<1>	mov	[esi-4], eax ; update parent's PTE
11070	000032F4	660D0002	<1>	or	ax, PTE_DUPLICATED ; (read only page & duplicated PTE!)
11071			<1>	dpt_3:	
11072	000032F8	AB	<1>	stosd	; EDI points to child's PTE
11073			<1>	;	
11074	000032F9	81C500100000	<1>	add	ebp, 4096 ; 20/07/2015 (next page)
11075			<1>	;	
11076	000032FF	39D7	<1>	cmp	edi, edx
11077	00003301	72AC	<1>	jb	short dpt_0
11078			<1>	dpt_p_err:	
11079	00003303	59	<1>	pop	ecx
11080	00003304	5A	<1>	pop	edx
11081	00003305	5F	<1>	pop	edi
11082	00003306	5E	<1>	pop	esi
11083	00003307	58	<1>	pop	eax ; *
11084			<1>	dpt_err:	
11085	00003308	C3	<1>	retn	
11086			<1>		
11087			<1>	page_fault_handler:	; CPU EXCEPTION 0Eh (14) : Page Fault !
11088			<1>	;	21/09/2015
11089			<1>	;	19/09/2015
11090			<1>	;	17/09/2015
11091			<1>	;	28/08/2015
11092			<1>	;	20/07/2015
11093			<1>	;	28/06/2015
11094			<1>	;	03/05/2015
11095			<1>	;	30/04/2015
11096			<1>	;	18/04/2015
11097			<1>	;	12/04/2015
11098			<1>	;	30/10/2014
11099			<1>	;	11/09/2014
11100			<1>	;	10/09/2014 (Retro UNIX 386 v1 - beginning)
11101			<1>	;	
11102			<1>	;	Note: This is not an interrupt/exception handler.
11103			<1>	;	This is a 'page fault remedy' subroutine
11104			<1>	;	which will be called by standard/uniform
11105			<1>	;	exception handler.
11106			<1>	;	

```

11107 <1> ; INPUT ->
11108 <1> ; [error_code] = 32 bit ERROR CODE (lower 5 bits are valid)
11109 <1> ;
11110 <1> ; cr2 = the virtual (linear) address
11111 <1> ; which has caused to page fault (19/09/2015)
11112 <1> ;
11113 <1> ; OUTPUT ->
11114 <1> ; (corresponding PAGE TABLE ENTRY is mapped/set)
11115 <1> ; EAX = 0 -> no error
11116 <1> ; EAX > 0 -> error code in EAX (also CF = 1)
11117 <1> ;
11118 <1> ; Modified Registers -> none (except EAX)
11119 <1> ;
11120 <1> ;
11121 <1> ; ERROR CODE:
11122 <1> ; 31 ..... 4 3 2 1 0
11123 <1> ; +---+-- --+---+---+---+---+---+
11124 <1> ; | Reserved | I | R | U | W | P |
11125 <1> ; +---+-- --+---+---+---+---+---+
11126 <1> ;
11127 <1> ; P : PRESENT - When set, the page fault was caused by
11128 <1> ; a page-protection violation. When not set,
11129 <1> ; it was caused by a non-present page.
11130 <1> ; W : WRITE - When set, the page fault was caused by
11131 <1> ; a page write. When not set, it was caused
11132 <1> ; by a page read.
11133 <1> ; U : USER - When set, the page fault was caused
11134 <1> ; while CPL = 3.
11135 <1> ; This does not necessarily mean that
11136 <1> ; the page fault was a privilege violation.
11137 <1> ; R : RESERVD - When set, the page fault was caused by
11138 <1> ; WRITE reading a 1 in a reserved field.
11139 <1> ; I : INSTRUC - When set, the page fault was caused by
11140 <1> ; FETCH an instruction fetch
11141 <1> ;
11142 <1> ;; x86 (32 bit) VIRTUAL ADDRESS TRANSLATION
11143 <1> ; 31 ..... 22 ..... 12 11 ..... 0
11144 <1> ; +-----+-----+-----+-----+-----+
11145 <1> ; | PAGE DIR. ENTRY # | PAGE TAB. ENTRY # | OFFSET |
11146 <1> ; +-----+-----+-----+-----+-----+
11147 <1> ;
11148 <1> ;
11149 <1> ;; CR3 REGISTER (Control Register 3)
11150 <1> ; 31 ..... 12 ..... 5 4 3 2 0
11151 <1> ; +-----+-----+-----+-----+-----+
11152 <1> ; | | | | | | | | | |
11153 <1> ; | PAGE DIRECTORY TABLE BASE ADDRESS | reserved|C|W|rsrvrd|
11154 <1> ; | | | | | | | | | |
11155 <1> ; +-----+-----+-----+-----+-----+
11156 <1> ;
11157 <1> ; PWT - WRITE THROUGH
11158 <1> ; PCD - CACHE DISABLE
11159 <1> ;
11160 <1> ;
11161 <1> ;; x86 PAGE DIRECTORY ENTRY (4 KByte Page)
11162 <1> ; 31 ..... 12 11 9 8 7 6 5 4 3 2 1 0
11163 <1> ; +-----+-----+-----+-----+-----+
11164 <1> ; | | | | | | | | | |
11165 <1> ; | PAGE TABLE BASE ADDRESS 31..12 | AVL |G|O|D|A|C|W|/|/|P|
11166 <1> ; | | | | | | | | | |
11167 <1> ; +-----+-----+-----+-----+-----+
11168 <1> ;
11169 <1> ; P - PRESENT
11170 <1> ; R/W - READ/WRITE
11171 <1> ; U/S - USER/SUPERVISOR
11172 <1> ; PWT - WRITE THROUGH
11173 <1> ; PCD - CACHE DISABLE
11174 <1> ; A - ACCESSED
11175 <1> ; D - DIRTY (IGNORED)
11176 <1> ; PAT - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
11177 <1> ; G - GLOBAL (IGNORED)
11178 <1> ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE
11179 <1> ;
11180 <1> ;
11181 <1> ;; x86 PAGE TABLE ENTRY (4 KByte Page)
11182 <1> ; 31 ..... 12 11 9 8 7 6 5 4 3 2 1 0
11183 <1> ; +-----+-----+-----+-----+-----+
11184 <1> ; | | | | | | | | | |
11185 <1> ; | PAGE FRAME BASE ADDRESS 31..12 | AVL |G|A|D|A|C|W|/|/|P|
11186 <1> ; | | | | | | | | | |
11187 <1> ; +-----+-----+-----+-----+-----+
11188 <1> ;
11189 <1> ; P - PRESENT
11190 <1> ; R/W - READ/WRITE
11191 <1> ; U/S - USER/SUPERVISOR
11192 <1> ; PWT - WRITE THROUGH
11193 <1> ; PCD - CACHE DISABLE
11194 <1> ; A - ACCESSED
11195 <1> ; D - DIRTY
11196 <1> ; PAT - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
11197 <1> ; G - GLOBAL
11198 <1> ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE

```

```

11199          <1> ;
11200          <1> ;
11201          <1> ;; 80386 PAGE TABLE ENTRY (4 KByte Page)
11202          <1> ; 31 12 11 9 8 7 6 5 4 3 2 1 0
11203          <1> ; +-----+-----+-----+-----+-----+-----+
11204          <1> ; | | | | | | | | | | | | | | | | | | | | | | | | | | | |
11205          <1> ; | PAGE FRAME BASE ADDRESS 31..12 | AVL | 0 | 0 | D | A | 0 | 0 | / | / | P |
11206          <1> ; | | | | | | | | | | | | | | | | | | | | | | | | | | | |
11207          <1> ; +-----+-----+-----+-----+-----+-----+
11208          <1> ;
11209          <1> ; P - PRESENT
11210          <1> ; R/W - READ/WRITE
11211          <1> ; U/S - USER/SUPERVISOR
11212          <1> ; D - DIRTY
11213          <1> ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE
11214          <1> ;
11215          <1> ; NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.
11216          <1> ;
11217          <1> ;
11218          <1> ;; Invalid Page Table Entry
11219          <1> ; 31 1 0
11220          <1> ; +-----+-----+-----+-----+-----+-----+
11221          <1> ; | | | | | | | | | | | | | | | | | | | | | | | | | | | |
11222          <1> ; | AVAILABLE | |
11223          <1> ; | | | | | | | | | | | | | | | | | | | | | | | | | | | |
11224          <1> ; +-----+-----+-----+-----+-----+-----+
11225          <1> ;
11226          <1> ;
11227 00003309 53 <1> push ebx
11228 0000330A 52 <1> push edx
11229 0000330B 51 <1> push ecx
11230          <1> ;
11231          <1> ; 21/09/2015 (debugging)
11232 0000330C FF05[E3740000] <1> inc dword [u.pfcount] ; page fault count for running process
11233 00003312 FF05[50810000] <1> inc dword [PF_Count] ; total page fault count
11234          <1> ; 28/06/2015
11235          <1> ;mov edx, [error_code] ; Lower 5 bits are valid
11236 00003318 8A15[48810000] <1> mov dl, [error_code]
11237          <1> ;
11238 0000331E F6C201 <1> test dl, 1 ; page fault was caused by a non-present page
11239          <1> ; sign
11240 00003321 7422 <1> jz short pfh_alloc_np
11241          <1> ;
11242          <1> ; If it is not a 'write on read only page' type page fault
11243          <1> ; major page fault error with minor reason must be returned without
11244          <1> ; fixing the problem. 'sys_exit with error' will be needed
11245          <1> ; after return here!
11246          <1> ; Page fault will be remedied, by copying page contents
11247          <1> ; to newly allocated page with write permission;
11248          <1> ; sys_fork -> sys_exec -> copy on write, demand paging method is
11249          <1> ; used for working with minimum possible memory usage.
11250          <1> ; sys_fork will duplicate page directory and tables of parent
11251          <1> ; process with 'read only' flag. If the child process attempts to
11252          <1> ; write on these read only pages, page fault will be directed here
11253          <1> ; for allocating a new page with same data/content.
11254          <1> ;
11255          <1> ; IMPORTANT : Retro UNIX 386 v1 (and SINGLIX and TR-DOS)
11256          <1> ; will not force to separate CODE and DATA space
11257          <1> ; in a process/program...
11258          <1> ; CODE segment/section may contain DATA!
11259          <1> ; It is flat, smoth and simplest programming method already as in
11260          <1> ; Retro UNIX 8086 v1 and MS-DOS programs.
11261          <1> ;
11262 00003323 F6C202 <1> test dl, 2 ; page fault was caused by a page write
11263          <1> ; sign
11264 00003326 0F84AB000000 <1> jz pfh_p_err
11265          <1> ; 31/08/2015
11266 0000332C F6C204 <1> test dl, 4 ; page fault was caused while CPL = 3 (user mode)
11267          <1> ; sign. (U+W+P = 4+2+1 = 7)
11268 0000332F 0F84A2000000 <1> jz pfh_pv_err
11269          <1> ;
11270          <1> ; make a new page and copy the parent's page content
11271          <1> ; as the child's new page content
11272          <1> ;
11273 00003335 0F20D3 <1> mov ebx, cr2 ; CR2 contains the linear address
11274          <1> ; which has caused to page fault
11275 00003338 E8A2000000 <1> call copy_page
11276 0000333D 0F828D000000 <1> jc pfh_im_err ; insufficient memory
11277          <1> ;
11278 00003343 EB7D <1> jmp pfh_cpp_ok
11279          <1> ;
11280          <1> pfh_alloc_np:
11281 00003345 E8EDFCFFFF <1> call allocate_page ; (allocate a new page)
11282 0000334A 0F8280000000 <1> jc pfh_im_err ; 'insufficient memory' error
11283          <1> pfh_chk_cpl:
11284          <1> ; EAX = Physical (base) address of the allocated (new) page
11285          <1> ; (Lower 12 bits are ZERO, because
11286          <1> ; the address is on a page boundary)
11287 00003350 80E204 <1> and dl, 4 ; CPL = 3 ?
11288 00003353 7505 <1> jnz short pfh_um
11289          <1> ; Page fault handler for kernel/system mode (CPL=0)
11290 00003355 0F20DB <1> mov ebx, cr3 ; CR3 (Control Register 3) contains physical address

```

```

11291                                     <1>             ; of the current/active page directory
11292                                     <1>             ; (Always kernel/system mode page directory, here!)
11293                                     <1>             ; Note: Lower 12 bits are 0. (page boundary)
11294 00003358 EB06                       <1>  jmp    short pfh_get_pde
11295                                     <1>             ;
11296                                     <1> pfh_um:           ; Page fault handler for user/appl. mode (CPL=3)
11297 0000335A 8B1D[D3740000]             <1>  mov    ebx, [u.pgdir] ; Page directory of current/active process
11298                                     <1>             ; Physical address of the USER's page directory
11299                                     <1>             ; Note: Lower 12 bits are 0. (page boundary)
11300                                     <1> pfh_get_pde:
11301 00003360 80CA03                       <1>  or     dl, 3 ; USER + WRITE + PRESENT or SYSTEM + WRITE + PRESENT
11302 00003363 0F20D1                     <1>  mov    ecx, cr2 ; CR2 contains the virtual address
11303                                     <1>             ; which has been caused to page fault
11304                                     <1>             ;
11305 00003366 C1E914                       <1>  shr    ecx, 20 ; shift 20 bits right
11306 00003369 80E1FC                       <1>  and    cl, 0FCh ; mask lower 2 bits to get PDE offset
11307                                     <1>             ;
11308 0000336C 01CB                         <1>  add    ebx, ecx ; now, EBX points to the relevant page dir entry
11309 0000336E 8B0B                         <1>  mov    ecx, [ebx] ; physical (base) address of the page table
11310 00003370 F6C101                     <1>  test   cl, 1 ; check bit 0 is set (1) or not (0).
11311 00003373 740B                         <1>  jz     short pfh_set_pde ; Page directory entry is not valid,
11312                                     <1>             ; set/validate page directory entry
11313 00003375 6681E100F0                   <1>  and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
11314 0000337A 89CB                         <1>  mov    ebx, ecx ; Physical address of the page table
11315 0000337C 89C1                         <1>  mov    ecx, eax ; new page address (physical)
11316 0000337E EB16                         <1>  jmp    short pfh_get_pte
11317                                     <1> pfh_set_pde:
11318                                     <1>  ;; NOTE: Page directories and page tables never be swapped out!
11319                                     <1>  ;;      (So, we know this PDE is empty or invalid)
11320                                     <1>             ;
11321 00003380 08D0                         <1>  or     al, dl ; lower 3 bits are used as U/S, R/W, P flags
11322 00003382 8903                         <1>  mov    [ebx], eax ; Let's put the new page directory entry here !
11323 00003384 30C0                         <1>  xor    al, al ; clear lower (3..8) bits
11324 00003386 89C3                         <1>  mov    ebx, eax
11325 00003388 E8AAFCEFFF                   <1>  call   allocate_page ; (allocate a new page)
11326 0000338D 7241                         <1>  jc     short pfh_im_err ; 'insufficient memory' error
11327                                     <1> pfh_spde_1:
11328                                     <1>  ; EAX = Physical (base) address of the allocated (new) page
11329 0000338F 89C1                         <1>  mov    ecx, eax
11330 00003391 E81BFDFFFF                   <1>  call   clear_page ; Clear page content
11331                                     <1> pfh_get_pte:
11332 00003396 0F20D0                       <1>  mov    eax, cr2 ; virtual address
11333                                     <1>             ; which has been caused to page fault
11334 00003399 89C7                         <1>  mov    edi, eax ; 20/07/2015
11335 0000339B C1E80C                       <1>  shr    eax, 12 ; shift 12 bit right to get
11336                                     <1>             ; higher 20 bits of the page fault address
11337 0000339E 25FF030000                   <1>  and    eax, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
11338 000033A3 C1E002                       <1>  shl    eax, 2 ; shift 2 bits left to get PTE offset
11339 000033A6 01C3                         <1>  add    ebx, eax ; now, EBX points to the relevant page table entry
11340 000033A8 8B03                         <1>  mov    eax, [ebx] ; get previous value of pte
11341                                     <1>             ; bit 0 of EAX is always 0 (otherwise we would not be here)
11342 000033AA 21C0                         <1>  and    eax, eax
11343 000033AC 7410                         <1>  jz     short pfh_gpte_1
11344                                     <1>  ; 20/07/2015
11345 000033AE 87D9                         <1>  xchg   ebx, ecx ; new page address (physical)
11346 000033B0 55                         <1>  push   ebp ; 20/07/2015
11347 000033B1 0F20D5                       <1>  mov    ebp, cr2
11348                                     <1>             ; ECX = physical address of the page table entry
11349                                     <1>             ; EBX = Memory page address (physical!)
11350                                     <1>             ; EAX = Swap disk (offset) address
11351                                     <1>             ; EBP = virtual address (page fault address)
11352 000033B4 E8B7000000                   <1>  call   swap_in
11353 000033B9 5D                         <1>  pop    ebp
11354 000033BA 7210                         <1>  jc     short pfh_err_retn
11355 000033BC 87CB                         <1>  xchg   ecx, ebx
11356                                     <1>             ; EBX = physical address of the page table entry
11357                                     <1>             ; ECX = new page
11358                                     <1> pfh_gpte_1:
11359 000033BE 08D1                         <1>  or     cl, dl ; lower 3 bits are used as U/S, R/W, P flags
11360 000033C0 890B                         <1>  mov    [ebx], ecx ; Let's put the new page table entry here !
11361                                     <1> pfh_cpp_ok:
11362                                     <1>  ; 20/07/2015
11363 000033C2 0F20D3                       <1>  mov    ebx, cr2
11364 000033C5 E82A020000                   <1>  call   add_to_swap_queue
11365                                     <1>             ;
11366                                     <1>  ; The new PTE (which contains the new page) will be added to
11367                                     <1>  ; the swap queue, here.
11368                                     <1>  ; (Later, if memory will become insufficient,
11369                                     <1>  ; one page will be swapped out which is at the head of
11370                                     <1>  ; the swap queue by using FIFO and access check methods.)
11371                                     <1>             ;
11372 000033CA 31C0                         <1>  xor    eax, eax ; 0
11373                                     <1>             ;
11374                                     <1> pfh_err_retn:
11375 000033CC 59                         <1>  pop    ecx
11376 000033CD 5A                         <1>  pop    edx
11377 000033CE 5B                         <1>  pop    ebx
11378 000033CF C3                         <1>  retn
11379                                     <1>
11380                                     <1> pfh_im_err:
11381 000033D0 B8E1000000                   <1>  mov    eax, ERR_MAJOR_PF + ERR_MINOR_IM ; Error code in AX
11382                                     <1>             ; Major (Primary) Error: Page Fault

```

```

11383                                     <1>                                     ; Minor (Secondary) Error: Insufficient Memory !
11384 000033D5 EBF5                     <1> jmp short pfh_err_retn
11385                                     <1>
11386                                     <1>
11387                                     <1> pfh_p_err: ; 09/03/2015
11388                                     <1> pfh_pv_err:
11389                                     <1> ; Page fault was caused by a protection-violation
11390 000033D7 B8E3000000                <1> mov eax, ERR_MAJOR_PF + ERR_MINOR_PV ; Error code in AX
11391                                     <1> ; Major (Primary) Error: Page Fault
11392                                     <1> ; Minor (Secondary) Error: Protection violation !
11393 000033DC F9                         <1> stc
11394 000033DD EBED                     <1> jmp short pfh_err_retn
11395                                     <1>
11396                                     <1> copy_page:
11397                                     <1> ; 22/09/2015
11398                                     <1> ; 21/09/2015
11399                                     <1> ; 19/09/2015
11400                                     <1> ; 07/09/2015
11401                                     <1> ; 31/08/2015
11402                                     <1> ; 20/07/2015
11403                                     <1> ; 05/05/2015
11404                                     <1> ; 03/05/2015
11405                                     <1> ; 18/04/2015
11406                                     <1> ; 12/04/2015
11407                                     <1> ; 30/10/2014
11408                                     <1> ; 18/10/2014 (Retro UNIX 386 v1 - beginning)
11409                                     <1> ;
11410                                     <1> ; INPUT ->
11411                                     <1> ; EBX = Virtual (linear) address of source page
11412                                     <1> ; (Page fault address)
11413                                     <1> ; OUTPUT ->
11414                                     <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
11415                                     <1> ; (corresponding PAGE TABLE ENTRY is mapped/set)
11416                                     <1> ; EAX = 0 (CF = 1)
11417                                     <1> ; if there is not a free page to be allocated
11418                                     <1> ; (page content of the source page will be copied
11419                                     <1> ; onto the target/new page)
11420                                     <1> ;
11421                                     <1> ; Modified Registers -> ecx, ebx (except EAX)
11422                                     <1> ;
11423 000033DF 56                         <1> push esi
11424 000033E0 57                         <1> push edi
11425                                     <1> ;push ebx
11426                                     <1> ;push ecx
11427 000033E1 31F6                     <1> xor esi, esi
11428 000033E3 C1EB0C                   <1> shr ebx, 12 ; shift 12 bits right to get PDE & PTE numbers
11429 000033E6 89D9                     <1> mov ecx, ebx ; save page fault address (as 12 bit shifted)
11430 000033E8 C1EB08                   <1> shr ebx, 8 ; shift 8 bits right and then
11431 000033EB 80E3FC                   <1> and bl, 0FCh ; mask lower 2 bits to get PDE offset
11432 000033EE 89DF                     <1> mov edi, ebx ; save it for the parent of current process
11433 000033F0 031D[D3740000]           <1> add ebx, [u.pgdir] ; EBX points to the relevant page dir entry
11434 000033F6 8B03                     <1> mov eax, [ebx] ; physical (base) address of the page table
11435 000033F8 662500F0                 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
11436 000033FC 89CB                     <1> mov ebx, ecx ; (restore higher 20 bits of page fault address)
11437 000033FE 81E3FF030000            <1> and ebx, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
11438 00003404 66C1E302                 <1> shl bx, 2 ; shift 2 bits left to get PTE offset
11439 00003408 01C3                     <1> add ebx, eax ; EBX points to the relevant page table entry
11440                                     <1> ; 07/09/2015
11441 0000340A 66F7030002                <1> test word [ebx], PTE_DUPLICATED ; (Does current process share this
11442                                     <1> ; read only page as a child process?)
11443 0000340F 7509                     <1> jnz short cpp_0 ; yes
11444 00003411 8B0B                     <1> mov ecx, [ebx] ; PTE value
11445 00003413 6681E100F0                <1> and cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
11446 00003418 EB32                     <1> jmp short cpp_1
11447                                     <1> cpp_0:
11448 0000341A 89FE                     <1> mov esi, edi
11449 0000341C 0335[D7740000]           <1> add esi, [u.ppgdir] ; the parent's page directory entry
11450 00003422 8B06                     <1> mov eax, [esi] ; physical (base) address of the page table
11451 00003424 662500F0                 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
11452 00003428 89CE                     <1> mov esi, ecx ; (restore higher 20 bits of page fault address)
11453 0000342A 81E6FF030000            <1> and esi, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
11454 00003430 66C1E602                 <1> shl si, 2 ; shift 2 bits left to get PTE offset
11455 00003434 01C6                     <1> add esi, eax ; EDX points to the relevant page table entry
11456 00003436 8B0E                     <1> mov ecx, [esi] ; PTE value of the parent process
11457                                     <1> ; 21/09/2015
11458 00003438 8B03                     <1> mov eax, [ebx] ; PTE value of the child process
11459 0000343A 662500F0                 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear page attributes
11460                                     <1> ;
11461 0000343E F6C101                     <1> test cl, PTE_A_PRESENT ; is it a present/valid page ?
11462 00003441 7424                     <1> jz short cpp_3 ; the parent's page is not same page
11463                                     <1> ;
11464 00003443 6681E100F0                <1> and cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
11465 00003448 39C8                     <1> cmp eax, ecx ; Same page?
11466 0000344A 751B                     <1> jne short cpp_3 ; Parent page and child page are not same
11467                                     <1> ; Convert child's page to writable page
11468                                     <1> cpp_1:
11469 0000344C E8E6FBFFFF                 <1> call allocate_page
11470 00003451 721A                     <1> jc short cpp_4 ; 'insufficient memory' error
11471 00003453 21F6                     <1> and esi, esi ; check ESI is valid or not
11472 00003455 7405                     <1> jz short cpp_2
11473                                     <1> ; Convert read only page to writable page
11474                                     <1> ; (for the parent of the current process)

```

```

11475 <1> ;and word [esi], PTE_A_CLEAR ; 0F000h
11476 <1> ; 22/09/2015
11477 00003457 890E <1> mov [esi], ecx
11478 00003459 800E07 <1> or byte [esi], PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER
11479 <1> ; 1+2+4 = 7
11480 <1> cpp_2:
11481 0000345C 89C7 <1> mov edi, eax ; new page address of the child process
11482 <1> ; 07/09/2015
11483 0000345E 89CE <1> mov esi, ecx ; the page address of the parent process
11484 00003460 B900040000 <1> mov ecx, PAGE_SIZE / 4
11485 00003465 F3A5 <1> rep movsd ; 31/08/2015
11486 <1> cpp_3:
11487 00003467 0C07 <1> or al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 1+2+4 = 7
11488 00003469 8903 <1> mov [ebx], eax ; Update PTE
11489 0000346B 28C0 <1> sub al, al ; clear attributes
11490 <1> cpp_4:
11491 <1> ;pop ecx
11492 <1> ;pop ebx
11493 0000346D 5F <1> pop edi
11494 0000346E 5E <1> pop esi
11495 0000346F C3 <1> retn
11496 <1>
11497 <1> ;; 28/04/2015
11498 <1> ;; 24/10/2014
11499 <1> ;; 21/10/2014 (Retro UNIX 386 v1 - beginning)
11500 <1> ;; SWAP_PAGE_QUEUE (4096 bytes)
11501 <1> ;;
11502 <1> ;; 0000 0001 0002 0003 .... 1020 1021 1022 1023
11503 <1> ;; +-----+-----+-----+-----+-----+-----+-----+-----+
11504 <1> ;; | pg1 | pg2 | pg3 | pg4 | .... |pg1021|pg1022|pg1023|pg1024|
11505 <1> ;; +-----+-----+-----+-----+-----+-----+-----+-----+
11506 <1> ;;
11507 <1> ;; [swpq_last] = 0 to 4096 (step 4) -> the last position on the queue
11508 <1> ;;
11509 <1> ;; Method:
11510 <1> ;; Swap page queue is a list of allocated pages with physical
11511 <1> ;; addresses (system mode virtual addresses = physical addresses).
11512 <1> ;; It is used for 'swap_in' and 'swap_out' procedures.
11513 <1> ;; When a new page is being allocated, swap queue is updated
11514 <1> ;; by 'swap_queue_shift' procedure, header of the queue (offset 0)
11515 <1> ;; is checked for 'accessed' flag. If the 1st page on the queue
11516 <1> ;; is 'accessed' or 'read only', it is dropped from the list;
11517 <1> ;; other pages from the 2nd to the last (in [swpq_last]) shifted
11518 <1> ;; to head then the 2nd page becomes the 1st and '[swpq_last]'
11519 <1> ;; offset value becomes it's previous offset value - 4.
11520 <1> ;; If the 1st page of the swap page queue is not 'accessed'
11521 <1> ;; the queue/list is not shifted.
11522 <1> ;; After the queue/list shift, newly allocated page is added
11523 <1> ;; to the tail of the queue at the [swpq_count*4] position.
11524 <1> ;; But, if [swpq_count] > 1023, the newly allocated page
11525 <1> ;; will not be added to the tail of swap page queue.
11526 <1> ;;
11527 <1> ;; During 'swap_out' procedure, swap page queue is checked for
11528 <1> ;; the first non-accessed, writable page in the list,
11529 <1> ;; from the head to the tail. The list is shifted to left
11530 <1> ;; (to the head) till a non-accessed page will be found in the list.
11531 <1> ;; Then, this page is swapped out (to disk) and then it is dropped
11532 <1> ;; from the list by a final swap queue shift. [swpq_count] value
11533 <1> ;; is changed. If all pages on the queue are 'accessed',
11534 <1> ;; 'insufficient memory' error will be returned ('swap_out'
11535 <1> ;; procedure will be failed)...
11536 <1> ;;
11537 <1> ;; Note: If the 1st page of the queue is an 'accessed' page,
11538 <1> ;; 'accessed' flag of the page will be reset (0) and that page
11539 <1> ;; (PTE) will be added to the tail of the queue after
11540 <1> ;; the check, if [swpq_count] < 1023. If [swpq_count] = 1024
11541 <1> ;; the queue will be rotated and the PTE in the head will be
11542 <1> ;; added to the tail after resetting 'accessed' bit.
11543 <1> ;;
11544 <1> ;;
11545 <1> ;;
11546 <1> ;; SWAP DISK/FILE (with 4096 bytes swapped page blocks)
11547 <1> ;;
11548 <1> ;; 00000000 00000004 00000008 0000000C ... size-8 size-4
11549 <1> ;; +-----+-----+-----+-----+-----+-----+-----+-----+
11550 <1> ;; |descriptr| page(1) | page(2) | page(3) | ... |page(n-1)| page(n) |
11551 <1> ;; +-----+-----+-----+-----+-----+-----+-----+-----+
11552 <1> ;;
11553 <1> ;; [swpd_next] = the first free block address in swapped page records
11554 <1> ;; for next free block search by 'swap_out' procedure.
11555 <1> ;; [swpd_size] = swap disk/file size in sectors (512 bytes)
11556 <1> ;; NOTE: max. possible swap disk size is 1024 GB
11557 <1> ;; (entire swap space must be accessed by using
11558 <1> ;; 31 bit offset address)
11559 <1> ;; [swpd_free] = free block (4096 bytes) count in swap disk/file space
11560 <1> ;; [swpd_start] = absolute/start address of the swap disk/file
11561 <1> ;; 0 for file, or beginning sector of the swap partition
11562 <1> ;; [swp_drv] = logical drive description table addr. of swap disk/file
11563 <1> ;;
11564 <1> ;;
11565 <1> ;; Method:
11566 <1> ;; When the memory (ram) becomes insufficient, page allocation

```

```

11567 <1> ;; procedure swaps out a page from memory to the swap disk
11568 <1> ;; (partition) or swap file to get a new free page at the memory.
11569 <1> ;; Swapping out is performed by using swap page queue.
11570 <1> ;;
11571 <1> ;; Allocation block size of swap disk/file is equal to page size
11572 <1> ;; (4096 bytes). Swapping address (in sectors) is recorded
11573 <1> ;; into relevant page file entry as 31 bit physical (logical)
11574 <1> ;; offset address as 1 bit shifted to left for present flag (0).
11575 <1> ;; Swapped page address is between 1 and swap disk/file size - 4.
11576 <1> ;; Absolute physical (logical) address of the swapped page is
11577 <1> ;; calculated by adding offset value to the swap partition's
11578 <1> ;; start address. If the swap device (disk) is a virtual disk
11579 <1> ;; or it is a file, start address of the swap disk/volume is 0,
11580 <1> ;; and offset value is equal to absolute (physical or logical)
11581 <1> ;; address/position. (It has not to be ZERO if the swap partition
11582 <1> ;; is in a partitioned virtual hard disk.)
11583 <1> ;;
11584 <1> ;; Note: Swap addresses are always specified/declared in sectors,
11585 <1> ;; not in bytes or in blocks/zones/clusters (4096 bytes) as unit.
11586 <1> ;;
11587 <1> ;; Swap disk/file allocation is mapped via 'Swap Allocation Table'
11588 <1> ;; at memory as similar to 'Memory Allocation Table'.
11589 <1> ;;
11590 <1> ;; Every bit of Swap Allocation Table represents one swap block
11591 <1> ;; (equal to page size) respectively. Bit 0 of the S.A.T. byte 0
11592 <1> ;; is reserved for swap disk/file block 0 as descriptor block
11593 <1> ;; (also for compatibility with PTE). If bit value is ZERO,
11594 <1> ;; it means relevant (respective) block is in use, and,
11595 <1> ;; of course, if bit value is 1, it means relevant (respective)
11596 <1> ;; swap disk/file block is free.
11597 <1> ;; For example: bit 1 of the byte 128 represents block 1025
11598 <1> ;; (128*8+1) or sector (offset) 8200 on the swap disk or
11599 <1> ;; byte (offset/position) 4198400 in the swap file.
11600 <1> ;; 4GB swap space is represented via 128KB Swap Allocation Table.
11601 <1> ;; Initial layout of Swap Allocation Table is as follows:
11602 <1> ;; -----
11603 <1> ;; 01111111111111111111111111111111 .... 1111111111111111111111111111
11604 <1> ;; -----
11605 <1> ;; (0 is reserved block, 1s represent free blocks respectively.)
11606 <1> ;; (Note: Allocation cell/unit of the table is bit, not byte)
11607 <1> ;;
11608 <1> ;; .....
11609 <1> ;;
11610 <1> ;; 'swap_out' procedure checks 'free_swap_blocks' count at first,
11611 <1> ;; then it searches Swap Allocation Table if free count is not
11612 <1> ;; zero. From beginning the [swpd_next] dword value, the first bit
11613 <1> ;; position with value of 1 on the table is converted to swap
11614 <1> ;; disk/file offset address, in sectors (not 4096 bytes block).
11615 <1> ;; 'ldrv_write' procedure is called with ldrv (logical drive
11616 <1> ;; number of physical swap disk or virtual swap disk)
11617 <1> ;; number, sector offset (not absolute sector -LBA- number),
11618 <1> ;; and sector count (8, 512*8 = 4096) and buffer address
11619 <1> ;; (memory page). That will be a direct disk write procedure.
11620 <1> ;; (for preventing late memory allocation, significant waiting).
11621 <1> ;; If disk write procedure returns with error or free count of
11622 <1> ;; swap blocks is ZERO, 'swap_out' procedure will return with
11623 <1> ;; 'insufficient memory error' (cf=1).
11624 <1> ;;
11625 <1> ;; (Note: Even if free swap disk/file blocks was not zero,
11626 <1> ;; any disk write error will not be fixed by 'swap_out' procedure,
11627 <1> ;; in other words, 'swap_out' will not check the table for other
11628 <1> ;; free blocks after a disk write error. It will return to
11629 <1> ;; the caller with error (CF=1) which means swapping is failed.
11630 <1> ;;
11631 <1> ;; After writing the page on to swap disk/file address/sector,
11632 <1> ;; 'swap_out' procedure returns with that swap (offset) sector
11633 <1> ;; address (cf=0).
11634 <1> ;;
11635 <1> ;; .....
11636 <1> ;;
11637 <1> ;; 'swap_in' procedure loads addressed (relevant) swap disk or
11638 <1> ;; file sectors at specified memory page. Then page allocation
11639 <1> ;; procedure updates relevant page table entry with 'present'
11640 <1> ;; attribute. If swap disk or file reading fails there is nothing
11641 <1> ;; to do, except to terminate the process which is the owner of
11642 <1> ;; the swapped page.
11643 <1> ;;
11644 <1> ;; 'swap_in' procedure sets the relevant/respective bit value
11645 <1> ;; in the Swap Allocation Table (as free block). 'swap_in' also
11646 <1> ;; updates [swpd_first] pointer if it is required.
11647 <1> ;;
11648 <1> ;; .....
11649 <1> ;;
11650 <1> ;; Note: If [swap_enabled] value is ZERO, that means there is not
11651 <1> ;; a swap disk or swap file in use... 'swap_in' and 'swap_out'
11652 <1> ;; procedures and 'swap page que' procedures will not be active...
11653 <1> ;; 'Insufficient memory' error will be returned by 'swap_out'
11654 <1> ;; and 'general protection fault' will be returned by 'swap_in'
11655 <1> ;; procedure, if it is called mistakenly (a wrong value in a PTE).
11656 <1> ;;
11657 <1>
11658 <1> swap_in:

```

```

11659 <1> ; 31/08/2015
11660 <1> ; 20/07/2015
11661 <1> ; 28/04/2015
11662 <1> ; 18/04/2015
11663 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
11664 <1> ;
11665 <1> ; INPUT ->
11666 <1> ; EBX = PHYSICAL (real/flat) ADDRESS OF THE MEMORY PAGE
11667 <1> ; EBP = VIRTUAL (LINEAR) ADDRESS (page fault address)
11668 <1> ; EAX = Offset Address for the swapped page on the
11669 <1> ; swap disk or in the swap file.
11670 <1> ;
11671 <1> ; OUTPUT ->
11672 <1> ; EAX = 0 if loading at memory has been successful
11673 <1> ;
11674 <1> ; CF = 1 -> swap disk reading error (disk/file not present
11675 <1> ; or sector not present or drive not ready
11676 <1> ; EAX = Error code
11677 <1> ; [u.error] = EAX
11678 <1> ; = The last error code for the process
11679 <1> ; (will be reset after returning to user)
11680 <1> ;
11681 <1> ; Modified Registers -> EAX
11682 <1> ;
11683 <1>
11684 00003470 833D[32810000]00 <1> cmp dword [swp_drv], 0
11685 00003477 7648 <1> jna short swpin_dnp_err
11686 <1>
11687 00003479 3B05[36810000] <1> cmp eax, [swpd_size]
11688 0000347F 734C <1> jnb short swpin_snp_err
11689 <1>
11690 00003481 56 <1> push esi
11691 00003482 53 <1> push ebx
11692 00003483 51 <1> push ecx
11693 00003484 8B35[32810000] <1> mov esi, [swp_drv]
11694 0000348A B908000000 <1> mov ecx, PAGE_SIZE / LOGIC_SECT_SIZE ; 8 !
11695 <1> ; Note: Even if corresponding physical disk's sector
11696 <1> ; size different than 512 bytes, logical disk sector
11697 <1> ; size is 512 bytes and disk reading procedure
11698 <1> ; will be performed for reading 4096 bytes
11699 <1> ; (2*2048, 8*512).
11700 <1> ; ESI = Logical disk description table address
11701 <1> ; EBX = Memory page (buffer) address (physical!)
11702 <1> ; EAX = Sector address (offset address, logical sector number)
11703 <1> ; ECX = Sector count ; 8 sectors
11704 0000348F 50 <1> push eax
11705 00003490 E833020000 <1> call logical_disk_read
11706 00003495 58 <1> pop eax
11707 00003496 730C <1> jnc short swpin_read_ok
11708 <1> ;
11709 00003498 B804000000 <1> mov eax, SWP_DISK_READ_ERR ; drive not ready or read error
11710 0000349D A3[CF740000] <1> mov [u.error], eax
11711 000034A2 EB19 <1> jmp short swpin_retn
11712 <1> ;
11713 <1> swpin_read_ok:
11714 <1> ; EAX = Offset address (logical sector number)
11715 000034A4 E891010000 <1> call unlink_swap_block ; Deallocate swap block
11716 <1> ;
11717 <1> ; EBX = Memory page (buffer) address (physical!)
11718 <1> ; 20/07/2015
11719 000034A9 89EB <1> mov ebx, ebp ; virtual address (page fault address)
11720 000034AB 6681E300F0 <1> and bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
11721 000034B0 8A1D[C9740000] <1> mov bl, [u.uno] ; current process number
11722 <1> ; EBX = Virtual address & process number combination
11723 000034B6 E89E000000 <1> call swap_queue_shift
11724 000034BB 29C0 <1> sub eax, eax ; 0 ; Error Code = 0 (no error)
11725 <1> ;
11726 <1> swpin_retn:
11727 000034BD 59 <1> pop ecx
11728 000034BE 5B <1> pop ebx
11729 000034BF 5E <1> pop esi
11730 000034C0 C3 <1> retn
11731 <1>
11732 <1> swpin_dnp_err:
11733 000034C1 B805000000 <1> mov eax, SWP_DISK_NOT_PRESENT_ERR
11734 <1> swpin_err_retn:
11735 000034C6 A3[CF740000] <1> mov [u.error], eax
11736 000034CB F9 <1> stc
11737 000034CC C3 <1> retn
11738 <1>
11739 <1> swpin_snp_err:
11740 000034CD B806000000 <1> mov eax, SWP_SECTOR_NOT_PRESENT_ERR
11741 000034D2 EBF2 <1> jmp short swpin_err_retn
11742 <1>
11743 <1> swap_out:
11744 <1> ; 31/08/2015
11745 <1> ; 05/05/2015
11746 <1> ; 30/04/2015
11747 <1> ; 28/04/2015
11748 <1> ; 18/04/2015
11749 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
11750 <1> ;

```



```

11751 <1> ; INPUT ->
11752 <1> ; none
11753 <1> ;
11754 <1> ; OUTPUT ->
11755 <1> ; EAX = Physical page address (which is swapped out
11756 <1> ; for allocating a new page)
11757 <1> ; CF = 1 -> swap disk writing error (disk/file not present
11758 <1> ; or sector not present or drive not ready
11759 <1> ; EAX = Error code
11760 <1> ; [u.error] = EAX
11761 <1> ; = The last error code for the process
11762 <1> ; (will be reset after returning to user)
11763 <1> ;
11764 <1> ; Modified Registers -> non (except EAX)
11765 <1> ;
11766 000034D4 66833D[30810000]01 <1> cmp word [swpq_count], 1
11767 000034DC 7274 <1> jc short swpout_im_err ; 'insufficient memory'
11768 <1>
11769 <1> ;cmp dword [swp_drv], 1
11770 <1> ;jc short swpout_dnp_err ; 'swap disk/file not present'
11771 <1>
11772 000034DE 833D[3A810000]01 <1> cmp dword [swpd_free], 1
11773 000034E5 7258 <1> jc short swpout_nfspc_err ; 'no free space on swap disk'
11774 <1>
11775 000034E7 53 <1> push ebx
11776 <1> swpout_1:
11777 000034E8 31DB <1> xor ebx, ebx
11778 000034EA E86A000000 <1> call swap_queue_shift
11779 000034EF 21C0 <1> and eax, eax ; entry count (before shifting)
11780 000034F1 7457 <1> jz short swpout_npts_err ; There is no any PTE in
11781 <1> ; the swap queue
11782 000034F3 BB00E00800 <1> mov ebx, swap_queue ; Addres of the head of
11783 <1> ; the swap queue
11784 000034F8 8B03 <1> mov eax, [ebx] ; The PTE in the queue head
11785 <1>
11786 <1> ;test al, PTE_A_PRESENT ; bit 0 = 1
11787 <1> ;jz short swpout_1 ; non-present page already
11788 <1> ; must not be in the queue
11789 <1>
11790 <1> ;test al, PTE_A_WRITE ; bit 1 = 0
11791 <1> ;jz short swpout_1 ; read only page (must not be
11792 <1> ; swapped out)
11793 <1>
11794 000034FA A820 <1> test al, PTE_A_ACCESS ; bit 5 = 1 (Accessed)
11795 000034FC 75EA <1> jnz short swpout_1 ; accessed page (must not be
11796 <1> ; swapped out, at this stage)
11797 <1> ;
11798 000034FE 662500F0 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
11799 <1> ;
11800 00003502 52 <1> push edx
11801 00003503 89DA <1> mov edx, ebx ; Page table entry address
11802 00003505 89C3 <1> mov ebx, eax ; Buffer (Page) Address
11803 <1> ;
11804 00003507 E861010000 <1> call link_swap_block
11805 0000350C 7304 <1> jnc short swpout_2 ; It may not be needed here
11806 0000350E 5A <1> pop edx ; because [swpd_free] value
11807 0000350F 5B <1> pop ebx
11808 00003510 EB2D <1> jmp short swpout_nfspc_err ; was checked at the beginging.
11809 <1> swpout_2:
11810 00003512 56 <1> push esi
11811 00003513 51 <1> push ecx
11812 00003514 50 <1> push eax ; sector address
11813 00003515 8B35[32810000] <1> mov esi, [swp_drv]
11814 0000351B B908000000 <1> mov ecx, PAGE_SIZE / LOGIC_SECT_SIZE ; 8 !
11815 <1> ; Note: Even if corresponding physical disk's sector
11816 <1> ; size different than 512 bytes, logical disk sector
11817 <1> ; size is 512 bytes and disk writing procedure
11818 <1> ; will be performed for writing 4096 bytes
11819 <1> ; (2*2048, 8*512).
11820 <1> ; ESI = Logical disk description table address
11821 <1> ; EBX = Buffer address
11822 <1> ; EAX = Sector adress (offset address, logical sector number)
11823 <1> ; ECX = Sector count ; 8 sectors
11824 00003520 E8A4010000 <1> call logical_disk_write
11825 00003525 59 <1> pop ecx ; sector address
11826 00003526 730C <1> jnc short swpout_write_ok
11827 <1> ;
11828 <1> ;; call unlink_swap_block ; this block must be left as 'in use'
11829 <1> swpout_dw_err:
11830 00003528 B808000000 <1> mov eax, SWP_DISK_WRITE_ERR ; drive not ready or write error
11831 0000352D A3[CF740000] <1> mov [u.error], eax
11832 00003532 EB06 <1> jmp short swpout_retn
11833 <1> ;
11834 <1> swpout_write_ok:
11835 <1> ; EBX = Buffer (page) address
11836 <1> ; EDX = Page Table entry address
11837 <1> ; ECX = Swap disk sector (file block) address (31 bit)
11838 00003534 D1E1 <1> shl ecx, 1 ; 31 bit sector address from bit 1 to bit 31
11839 00003536 890A <1> mov [edx], ecx
11840 <1> ; bit 0 = 0 (swapped page)
11841 00003538 89D8 <1> mov eax, ebx
11842 <1> swpout_retn:

```

11843	0000353A	59	<1>	pop	ecx
11844	0000353B	5E	<1>	pop	esi
11845	0000353C	5A	<1>	pop	edx
11846	0000353D	5B	<1>	pop	ebx
11847	0000353E	C3	<1>	retn	
11848			<1>		
11849			<1>	; Note: Swap_queue will not be updated in 'swap_out' procedure	
11850			<1>	; after the page is swapped out. (the PTE at the queue head	
11851			<1>	; -with 'non-present' attribute- will be dropped from the	
11852			<1>	; the queue in next 'swap_out' or in next 'swap_queue_shift'.	
11853			<1>		
11854			<1>	; swpout_dnp_err:	
11855			<1>	; mov eax, SWP_DISK_NOT_PRESENT_ERR ; disk not present	
11856			<1>	; jmp short swpout_err_retn	
11857			<1>	; swpout_nfspc_err:	
11858	0000353F	B807000000	<1>	mov eax, SWP_NO_FREE_SPACE_ERR ; no free space	
11859			<1>	; swpout_err_retn:	
11860	00003544	A3[CF740000]	<1>	mov [u.error], eax	
11861			<1>	; stc	
11862	00003549	C3	<1>	retn	
11863			<1>	; swpout_npts_err:	
11864	0000354A	B809000000	<1>	mov eax, SWP_NO_PAGE_TO_SWAP_ERR	
11865	0000354F	5B	<1>	pop ebx	
11866	00003550	EBF2	<1>	jmp short swpout_err_retn	
11867			<1>	; swpout_im_err:	
11868	00003552	B801000000	<1>	mov eax, ERR_MINOR_IM ; insufficient (out of) memory	
11869	00003557	EBEB	<1>	jmp short swpout_err_retn	
11870			<1>		
11871			<1>	; swap_queue_shift:	
11872			<1>	; 20/07/2015	
11873			<1>	; 28/04/2015	
11874			<1>	; 18/04/2015	
11875			<1>	; 23/10/2014 (Retro UNIX 386 v1 - beginning)	
11876			<1>	;	
11877			<1>	; INPUT ->	
11878			<1>	; EBX = Virtual (linear) address (bit 12 to 31)	
11879			<1>	; and process number combination (bit 0 to 11)	
11880			<1>	; EBX = 0 -> shift/drop from the head (offset 0)	
11881			<1>	; OUTPUT ->	
11882			<1>	; If EBX input > 0	
11883			<1>	; the queue will be shifted 4 bytes (dword),	
11884			<1>	; from the tail to the head, up to entry offset	
11885			<1>	; which points to EBX input value or nothing	
11886			<1>	; to do if EBX value is not found in the queue.	
11887			<1>	; (The entry -with EBX value- will be removed	
11888			<1>	; from the queue if it is found.)	
11889			<1>	; If EBX input = 0	
11890			<1>	; the queue will be shifted 4 bytes (dword),	
11891			<1>	; from the tail to the head, if the PTE address	
11892			<1>	; in head of the queue is marked as "accessed"	
11893			<1>	; or it is marked as "non present".	
11894			<1>	; (If "accessed" flag of the PTE -in the head-	
11895			<1>	; is set -to 1-, it will be reset -to 0- and then,	
11896			<1>	; the queue will be rotated -without dropping	
11897			<1>	; the PTE from the queue-, for 4 bytes on head	
11898			<1>	; to tail direction. The PTE in the head will be	
11899			<1>	; moved in the tail, other PTEs will be shifted on	
11900			<1>	; head direction.)	
11901			<1>	;	
11902			<1>	; EAX = [swpq_count] (before the shifting)	
11903			<1>	; (EAX = 0 -> next 'swap_out' stage	
11904			<1>	; is not applicable)	
11905			<1>	;	
11906			<1>	; Modified Registers -> EAX	
11907			<1>	;	
11908	00003559	0FB705[30810000]	<1>	movzx eax, word [swpq_count] ; Max. 1024	
11909	00003560	6621C0	<1>	and ax, ax	
11910	00003563	7433	<1>	jz short swpqs_retn	
11911	00003565	57	<1>	push edi	
11912	00003566	56	<1>	push esi	
11913	00003567	53	<1>	push ebx	
11914	00003568	51	<1>	push ecx	
11915	00003569	50	<1>	push eax	
11916	0000356A	BE00E00800	<1>	mov esi, swap_queue	
11917	0000356F	89C1	<1>	mov ecx, eax	
11918	00003571	09DB	<1>	or ebx, ebx	
11919	00003573	7424	<1>	jz short swpqs_7	
11920			<1>	; swpqs_1:	
11921	00003575	AD	<1>	lodsd	
11922	00003576	39D8	<1>	cmp eax, ebx	
11923	00003578	7404	<1>	je short swpqs_2	
11924	0000357A	E2F9	<1>	loop swpqs_1	
11925	0000357C	EB15	<1>	jmp short swpqs_6	
11926			<1>	; swpqs_2:	
11927	0000357E	89F7	<1>	mov edi, esi	
11928	00003580	83EF04	<1>	sub edi, 4	
11929			<1>	; swpqs_3:	
11930	00003583	66FF0D[30810000]	<1>	dec word [swpq_count]	
11931	0000358A	7403	<1>	jz short swpqs_5	
11932			<1>	; swpqs_4:	
11933	0000358C	49	<1>	dec ecx	
11934	0000358D	F3A5	<1>	rep movsd ; shift up (to the head)	

11935		<1> swpqs_5:
11936	0000358F 31C0	<1> xor eax, eax
11937	00003591 8907	<1> mov [edi], eax
11938		<1> swpqs_6:
11939	00003593 58	<1> pop eax
11940	00003594 59	<1> pop ecx
11941	00003595 5B	<1> pop ebx
11942	00003596 5E	<1> pop esi
11943	00003597 5F	<1> pop edi
11944		<1> swpqs_retn:
11945	00003598 C3	<1> retn
11946		<1> swpqs_7:
11947	00003599 89F7	<1> mov edi, esi ; head
11948	0000359B AD	<1> lodsd
11949		<1> ; 20/07/2015
11950	0000359C 89C3	<1> mov ebx, eax
11951	0000359E 81E300F0FFFF	<1> and ebx, ~PAGE_OFF ; ~0FFFh
11952		<1> ; ebx = virtual address (at page boundary)
11953	000035A4 25FF0F0000	<1> and eax, PAGE_OFF ; 0FFFh
11954		<1> ; ax = process number (1 to 4095)
11955	000035A9 3A05[C9740000]	<1> cmp al, [u.uno]
11956		<1> ; Max. 16 (nproc) processes for Retro UNIX 386 v1
11957	000035AF 7507	<1> jne short swpqs_8
11958	000035B1 A1[D3740000]	<1> mov eax, [u.pgdir]
11959	000035B6 EB16	<1> jmp short swpqs_9
11960		<1> swpqs_8:
11961		<1> ;shl ax, 2
11962	000035B8 C0E002	<1> shl al, 2
11963	000035BB 8B80[12720000]	<1> mov eax, [eax+p.upage-4]
11964	000035C1 09C0	<1> or eax, eax
11965	000035C3 74BE	<1> jz short swpqs_3 ; invalid upage
11966	000035C5 83C05B	<1> add eax, u.pgdir - user
11967		<1> ; u.pgdir value for the process
11968		<1> ; is in [eax]
11969	000035C8 8B00	<1> mov eax, [eax]
11970	000035CA 21C0	<1> and eax, eax
11971	000035CC 74B5	<1> jz short swpqs_3 ; invalid page directory
11972		<1> swpqs_9:
11973	000035CE 52	<1> push edx
11974		<1> ; eax = page directory
11975		<1> ; ebx = virtual address
11976	000035CF E87EFBFFFF	<1> call get_pte
11977	000035D4 89D3	<1> mov ebx, edx ; PTE address
11978	000035D6 5A	<1> pop edx
11979	000035D7 72AA	<1> jc short swpqs_3 ; empty PDE
11980		<1> ; EAX = PTE value
11981	000035D9 A801	<1> test al, PTE_A_PRESENT ; bit 0 = 1
11982	000035DB 74A6	<1> jz short swpqs_3 ; Drop non-present page
11983		<1> ; from the queue (head)
11984	000035DD A802	<1> test al, PTE_A_WRITE ; bit 1 = 0
11985	000035DF 74A2	<1> jz short swpqs_3 ; Drop read only page
11986		<1> ; from the queue (head)
11987		<1> ;test al, PTE_A_ACCESS ; bit 5 = 1 (Accessed)
11988		<1> ;jz short swpqs_6 ; present
11989		<1> ; non-accessed page
11990	000035E1 0FBAF005	<1> btr eax, PTE_A_ACCESS_BIT ; reset 'accessed' bit
11991	000035E5 73AC	<1> jnc short swpqs_6 ; non-accessed page
11992	000035E7 8903	<1> mov [ebx], eax ; save changed attribute
11993		<1> ;
11994		<1> ; Rotation (head -> tail)
11995	000035E9 49	<1> dec ecx ; entry count -> last entry number
11996	000035EA 74A7	<1> jz short swpqs_6
11997		<1> ; esi = head + 4
11998		<1> ; edi = head
11999	000035EC 8B07	<1> mov eax, [edi] ; 20/07/2015
12000	000035EE F3A5	<1> rep movsd ; n = 1 to k-1, [n - 1] = [n]
12001	000035F0 8907	<1> mov [edi], eax ; head -> tail ; [k] = [1]
12002	000035F2 EB9F	<1> jmp short swpqs_6
12003		<1>
12004		<1> add_to_swap_queue:
12005		<1> ; temporary - 16/09/2015
12006	000035F4 C3	<1> retn
12007		<1> ; 20/07/2015
12008		<1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
12009		<1> ;
12010		<1> ; Adds new page to swap queue
12011		<1> ; (page directories and page tables must not be added
12012		<1> ; to swap queue)
12013		<1> ;
12014		<1> ; INPUT ->
12015		<1> ; EBX = Virtual address (for current process, [u.uno])
12016		<1> ;
12017		<1> ; OUTPUT ->
12018		<1> ; EAX = [swpq_count]
12019		<1> ; (after the PTE has been added)
12020		<1> ; EAX = 0 -> Swap queue is full, (1024 entries)
12021		<1> ; the pte could not be added.
12022		<1> ;
12023		<1> ; Modified Registers -> EAX
12024		<1> ;
12025	000035F5 53	<1> push ebx
12026	000035F6 6681E300F0	<1> and bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11

```

12027 000035FB 8A1D[C9740000] <1> mov bl, [u.uno] ; current process number
12028 00003601 E853FFFFFF <1> call swap_queue_shift ; drop from the queue if
12029 <1> ; it is already in the queue
12030 <1> ; Then add it to the tail of the queue
12031 00003606 0FB705[30810000] <1> movzx eax, word [swpq_count]
12032 0000360D 663D0004 <1> cmp ax, 1024
12033 00003611 7205 <1> jb short atsq_1
12034 00003613 6629C0 <1> sub ax, ax
12035 00003616 5B <1> pop ebx
12036 00003617 C3 <1> retn
12037 <1> atsq_1:
12038 00003618 56 <1> push esi
12039 00003619 BE00E00800 <1> mov esi, swap_queue
12040 0000361E 6621C0 <1> and ax, ax
12041 00003621 740A <1> jz short atsq_2
12042 00003623 66C1E002 <1> shl ax, 2 ; convert to offset
12043 00003627 01C6 <1> add esi, eax
12044 00003629 66C1E802 <1> shr ax, 2
12045 <1> atsq_2:
12046 0000362D 6640 <1> inc ax
12047 0000362F 891E <1> mov [esi], ebx ; Virtual address + [u.uno] combination
12048 00003631 66A3[30810000] <1> mov [swpq_count], ax
12049 00003637 5E <1> pop esi
12050 00003638 5B <1> pop ebx
12051 00003639 C3 <1> retn
12052 <1>
12053 <1> unlink_swap_block:
12054 <1> ; 15/09/2015
12055 <1> ; 30/04/2015
12056 <1> ; 18/04/2015
12057 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
12058 <1> ;
12059 <1> ; INPUT ->
12060 <1> ; EAX = swap disk/file offset address
12061 <1> ; (bit 1 to bit 31)
12062 <1> ; OUTPUT ->
12063 <1> ; [swpd_free] is increased
12064 <1> ; (corresponding SWAP DISK ALLOC. TABLE bit is SET)
12065 <1> ;
12066 <1> ; Modified Registers -> EAX
12067 <1> ;
12068 0000363A 53 <1> push ebx
12069 0000363B 52 <1> push edx
12070 <1> ;
12071 0000363C C1E804 <1> shr eax, SECTOR_SHIFT+1 ;3+1 ; shift sector address to
12072 <1> ; 3 bits right
12073 <1> ; to get swap block/page number
12074 0000363F 89C2 <1> mov edx, eax
12075 <1> ; 15/09/2015
12076 00003641 C1EA03 <1> shr edx, 3 ; to get offset to S.A.T.
12077 <1> ; (1 allocation bit = 1 page)
12078 <1> ; (1 allocation bytes = 8 pages)
12079 00003644 80E2FC <1> and dl, 0FCh ; clear lower 2 bits
12080 <1> ; (to get 32 bit position)
12081 <1> ;
12082 00003647 BB00000D00 <1> mov ebx, swap_alloc_table ; Swap Allocation Table address
12083 0000364C 01D3 <1> add ebx, edx
12084 0000364E 83E01F <1> and eax, 1Fh ; lower 5 bits only
12085 <1> ; (allocation bit position)
12086 00003651 3B05[3E810000] <1> cmp eax, [swpd_next] ; is the new free block addr. lower
12087 <1> ; than the address in 'swpd_next' ?
12088 <1> ; (next/first free block value)
12089 00003657 7305 <1> jnb short uswpbl_1 ; no
12090 00003659 A3[3E810000] <1> mov [swpd_next], eax ; yes
12091 <1> uswpbl_1:
12092 0000365E 0FAB03 <1> bts [ebx], eax ; unlink/release/deallocate block
12093 <1> ; set relevant bit to 1.
12094 <1> ; set CF to the previous bit value
12095 00003661 F5 <1> cmc ; complement carry flag
12096 00003662 7206 <1> jc short uswpbl_2 ; do not increase swfd_free count
12097 <1> ; if the block is already deallocated
12098 <1> ; before.
12099 00003664 FF05[3A810000] <1> inc dword [swpd_free]
12100 <1> uswpbl_2:
12101 0000366A 5A <1> pop edx
12102 0000366B 5B <1> pop ebx
12103 0000366C C3 <1> retn
12104 <1>
12105 <1> link_swap_block:
12106 <1> ; 01/07/2015
12107 <1> ; 18/04/2015
12108 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
12109 <1> ;
12110 <1> ; INPUT -> none
12111 <1> ;
12112 <1> ; OUTPUT ->
12113 <1> ; EAX = OFFSET ADDRESS OF THE ALLOCATED BLOCK (4096 bytes)
12114 <1> ; in sectors (corresponding
12115 <1> ; SWAP DISK ALLOCATION TABLE bit is RESET)
12116 <1> ;
12117 <1> ; CF = 1 and EAX = 0
12118 <1> ; if there is not a free block to be allocated

```

12119		<1>	;
12120		<1>	; Modified Registers -> none (except EAX)
12121		<1>	;
12122		<1>	;
12123		<1>	;mov eax, [swpd_free]
12124		<1>	;and eax, eax
12125		<1>	;jz short out_of_swpspc
12126		<1>	;
12127	0000366D 53	<1>	push ebx
12128	0000366E 51	<1>	push ecx
12129		<1>	;
12130	0000366F BB00000D00	<1>	mov ebx, swap_alloc_table ; Swap Allocation Table offset
12131	00003674 89D9	<1>	mov ecx, ebx
12132	00003676 031D[3E810000]	<1>	add ebx, [swpd_next] ; Free block searching starts from here
12133		<1>	; next_free_swap_block >> 5
12134	0000367C 030D[42810000]	<1>	add ecx, [swpd_last] ; Free block searching ends here
12135		<1>	; (total_swap_blocks - 1) >> 5
12136		<1>	lswbl_scan:
12137	00003682 39CB	<1>	cmp ebx, ecx
12138	00003684 770A	<1>	ja short lswbl_notfound
12139		<1>	;
12140	00003686 0FBC03	<1>	bsf eax, [ebx] ; Scans source operand for first bit set (1).
12141		<1>	; Clears ZF if a bit is found set (1) and
12142		<1>	; loads the destination with an index to
12143		<1>	; first set bit. (0 -> 31)
12144		<1>	; Sets ZF to 1 if no bits are found set.
12145		<1>	; 01/07/2015
12146	00003689 751C	<1>	jnz short lswbl_found ; ZF = 0 -> a free block has been found
12147		<1>	;
12148		<1>	; NOTE: a Swap Disk Allocation Table bit
12149		<1>	; with value of 1 means
12150		<1>	; the corresponding page is free
12151		<1>	; (Retro UNIX 386 v1 feaure only!)
12152	0000368B 83C304	<1>	add ebx, 4
12153		<1>	; We return back for searching next page block
12154		<1>	; NOTE: [swpd_free] is not ZERO; so,
12155		<1>	; we always will find at least 1 free block here.
12156	0000368E EBF2	<1>	jmp short lswbl_scan
12157		<1>	;
12158		<1>	lswbl_notfound:
12159	00003690 81E900000D00	<1>	sub ecx, swap_alloc_table
12160	00003696 890D[3E810000]	<1>	mov [swpd_next], ecx ; next/first free page = last page
12161		<1>	; (unlink_swap_block procedure will change it)
12162	0000369C 31C0	<1>	xor eax, eax
12163	0000369E A3[3A810000]	<1>	mov [swpd_free], eax
12164	000036A3 F9	<1>	stc
12165		<1>	lswbl_ok:
12166	000036A4 59	<1>	pop ecx
12167	000036A5 5B	<1>	pop ebx
12168	000036A6 C3	<1>	retn
12169		<1>	;
12170		<1>	;out_of_swpspc:
12171		<1>	;stc
12172		<1>	;retn
12173		<1>	;
12174		<1>	lswbl_found:
12175	000036A7 89D9	<1>	mov ecx, ebx
12176	000036A9 81E900000D00	<1>	sub ecx, swap_alloc_table
12177	000036AF 890D[3E810000]	<1>	mov [swpd_next], ecx ; Set first free block searching start
12178		<1>	; address/offset (to the next)
12179	000036B5 FF0D[3A810000]	<1>	dec dword [swpd_free] ; 1 block has been allocated (X = X-1)
12180		<1>	;
12181	000036BB 0FB303	<1>	btr [ebx], eax ; The destination bit indexed by the source value
12182		<1>	; is copied into the Carry Flag and then cleared
12183		<1>	; in the destination.
12184		<1>	;
12185		<1>	; Reset the bit which is corresponding to the
12186		<1>	; (just) allocated block.
12187	000036BE C1E105	<1>	shl ecx, 5 ; (block offset * 32) + block index
12188	000036C1 01C8	<1>	add eax, ecx ; = block number
12189	000036C3 C1E003	<1>	shl eax, SECTOR_SHIFT ; 3, sector (offset) address of the block
12190		<1>	; 1 block = 8 sectors
12191		<1>	;
12192		<1>	; EAX = offset address of swap disk/file sector (beginning of the block)
12193		<1>	;
12194		<1>	; NOTE: The relevant page table entry will be updated
12195		<1>	; according to this EAX value...
12196		<1>	;
12197	000036C6 EBDC	<1>	jmp short lswbl_ok
12198		<1>	;
12199		<1>	logical_disk_read:
12200		<1>	; 20/07/2015
12201		<1>	; 09/03/2015 (temporary code here)
12202		<1>	;
12203		<1>	; INPUT ->
12204		<1>	; ESI = Logical disk description table address
12205		<1>	; EBX = Memory page (buffer) address (physical!)
12206		<1>	; EAX = Sector address (offset address, logical sector number)
12207		<1>	; ECX = Sector count
12208		<1>	;
12209		<1>	;
12210	000036C8 C3	<1>	retn

12211		<1>
12212		<1> logical_disk_write:
12213		<1> ; 20/07/2015
12214		<1> ; 09/03/2015 (temporary code here)
12215		<1> ;
12216		<1> ; INPUT ->
12217		<1> ; ESI = Logical disk description table address
12218		<1> ; EBX = Memory page (buffer) address (physical!)
12219		<1> ; EAX = Sector address (offset address, logical sector number)
12220		<1> ; ECX = Sector count
12221		<1> ;
12222	000036C9 C3	<1> retn
12223		<1>
12224		<1> get_physical_addr:
12225		<1> ; 18/10/2015
12226		<1> ; 29/07/2015
12227		<1> ; 20/07/2015
12228		<1> ; 04/06/2015
12229		<1> ; 20/05/2015
12230		<1> ; 28/04/2015
12231		<1> ; 18/04/2015
12232		<1> ; Get physical address
12233		<1> ; (allocates a new page for user if it is not present)
12234		<1> ;
12235		<1> ; (This subroutine is needed for mapping user's virtual
12236		<1> ; (buffer) address to physical address (of the buffer).)
12237		<1> ; ('sys write', 'sys read' system calls...)
12238		<1> ;
12239		<1> ; INPUT ->
12240		<1> ; EBX = virtual address
12241		<1> ; u.pgdir = page directory (physical) address
12242		<1> ;
12243		<1> ; OUTPUT ->
12244		<1> ; EAX = physical address
12245		<1> ; EBX = linear address
12246		<1> ; EDX = physical address of the page frame
12247		<1> ; (with attribute bits)
12248		<1> ; ECX = byte count within the page frame
12249		<1> ;
12250		<1> ; Modified Registers -> EAX, EBX, ECX, EDX
12251		<1> ;
12252	000036CA 81C300004000	<1> add ebx, CORE ; 18/10/2015
12253		<1> ;
12254	000036D0 A1[D3740000]	<1> mov eax, [u.pgdir]
12255	000036D5 E878FAFFFF	<1> call get_pte
12256		<1> ; EDX = Page table entry address (if CF=0)
12257		<1> ; Page directory entry address (if CF=1)
12258		<1> ; (Bit 0 value is 0 if PT is not present)
12259		<1> ; EAX = Page table entry value (page address)
12260		<1> ; CF = 1 -> PDE not present or invalid ?
12261	000036DA 731C	<1> jnc short gpa_1
12262		<1> ;
12263	000036DC E856F9FFFF	<1> call allocate_page
12264	000036E1 725B	<1> jc short gpa_im_err ; 'insufficient memory' error
12265		<1> gpa_0:
12266	000036E3 E8C9F9FFFF	<1> call clear_page
12267		<1> ; EAX = Physical (base) address of the allocated (new) page
12268	000036E8 0C07	<1> or al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
12269		<1> ; lower 3 bits are used as U/S, R/W, P flags
12270		<1> ; (user, writable, present page)
12271	000036EA 8902	<1> mov [edx], eax ; Let's put the new page directory entry here !
12272	000036EC A1[D3740000]	<1> mov eax, [u.pgdir]
12273	000036F1 E85CFAFFFF	<1> call get_pte
12274	000036F6 7246	<1> jc short gpa_im_err ; 'insufficient memory' error
12275		<1> gpa_1:
12276		<1> ; EAX = PTE value, EDX = PTE address
12277	000036F8 A801	<1> test al, PTE_A_PRESENT
12278	000036FA 751A	<1> jnz short gpa_3
12279	000036FC 09C0	<1> or eax, eax
12280	000036FE 7430	<1> jz short gpa_4 ; Allocate a new page
12281		<1> ; 20/07/2015
12282	00003700 55	<1> push ebp
12283	00003701 89DD	<1> mov ebp, ebx ; virtual (linear) address
12284		<1> ; reload swapped page
12285	00003703 E83C000000	<1> call reload_page ; 28/04/2015
12286	00003708 5D	<1> pop ebp
12287	00003709 7224	<1> jc short gpa_retn
12288		<1> gpa_2:
12289		<1> ; 20/07/2015
12290		<1> ; 20/05/2015
12291		<1> ; add this page to swap queue
12292	0000370B 50	<1> push eax
12293		<1> ; EBX = virtual address
12294	0000370C E8E3FEFFFF	<1> call add_to_swap_queue
12295	00003711 58	<1> pop eax
12296		<1> ; PTE address in EDX
12297		<1> ; virtual address in EBX
12298		<1> ; EAX = memory page address
12299	00003712 0C07	<1> or al, PTE_A_PRESENT + PTE_A_USER + PTE_A_WRITE
12300		<1> ; present flag, bit 0 = 1
12301		<1> ; user flag, bit 2 = 1
12302		<1> ; writable flag, bit 1 = 1

12303	00003714	8902	<1>	mov	[edx], eax	; Update PTE value
12304			<1>	gpa_3:		
12305			<1>	; 18/10/2015		
12306	00003716	89D9	<1>	mov	ecx, ebx	
12307	00003718	81E1FF0F0000	<1>	and	ecx, PAGE_OFF	
12308	0000371E	89C2	<1>	mov	edx, eax	
12309	00003720	662500F0	<1>	and	ax, PTE_A_CLEAR	
12310	00003724	01C8	<1>	add	eax, ecx	
12311	00003726	F7D9	<1>	neg	ecx ; 1 -> -1 (0FFFFFFFh), 4095 (0FFFh) -> -4095	
12312	00003728	81C100100000	<1>	add	ecx, PAGE_SIZE	
12313	0000372E	F8	<1>	clc		
12314			<1>	gpa_retn:		
12315	0000372F	C3	<1>	retn		
12316			<1>	gpa_4:		
12317	00003730	E802F9FFFF	<1>	call	allocate_page	
12318	00003735	7207	<1>	jc	short gpa_im_err ; 'insufficient memory' error	
12319	00003737	E875F9FFFF	<1>	call	clear_page	
12320	0000373C	EBCD	<1>	jmp	short gpa_2	
12321			<1>			
12322			<1>	gpa_im_err:		
12323	0000373E	B801000000	<1>	mov	eax, ERR_MINOR_IM ; Insufficient memory (minor) error!	
12324			<1>		; Major error = 0 (No protection fault)	
12325	00003743	C3	<1>	retn		
12326			<1>			
12327			<1>	reload_page:		
12328			<1>	; 20/07/2015		
12329			<1>	; 28/04/2015 (Retro UNIX 386 v1 - beginning)		
12330			<1>	;		
12331			<1>	; Reload (Restore) swapped page at memory		
12332			<1>	;		
12333			<1>	; INPUT ->		
12334			<1>	; EBP = Virtual (linear) memory address		
12335			<1>	; EAX = PTE value (swap disk sector address)		
12336			<1>	; (Swap disk sector address = bit 1 to bit 31 of EAX)		
12337			<1>	; OUTPUT ->		
12338			<1>	; EAX = PHYSICAL (real/flat) ADDRESS OF RELOADED PAGE		
12339			<1>	;		
12340			<1>	; CF = 1 and EAX = error code		
12341			<1>	;		
12342			<1>	; Modified Registers -> none (except EAX)		
12343			<1>	;		
12344	00003744	D1E8	<1>	shr	eax, 1 ; Convert PTE value to swap disk address	
12345	00003746	53	<1>	push	ebx ;	
12346	00003747	89C3	<1>	mov	ebx, eax ; Swap disk (offset) address	
12347	00003749	E8E9F8FFFF	<1>	call	allocate_page	
12348	0000374E	720C	<1>	jc	short rlp_im_err	
12349	00003750	93	<1>	xchg	eax, ebx	
12350			<1>	; EBX = Physical memory (page) address		
12351			<1>	; EAX = Swap disk (offset) address		
12352			<1>	; EBP = Virtual (linear) memory address		
12353	00003751	E81AFDFFFF	<1>	call	swap_in	
12354	00003756	720B	<1>	jc	short rlp_swp_err ; (swap disk/file read error)	
12355	00003758	89D8	<1>	mov	eax, ebx	
12356			<1>	rlp_retn:		
12357	0000375A	5B	<1>	pop	ebx	
12358	0000375B	C3	<1>	retn		
12359			<1>			
12360			<1>	rlp_im_err:		
12361	0000375C	B801000000	<1>	mov	eax, ERR_MINOR_IM ; Insufficient memory (minor) error!	
12362			<1>		; Major error = 0 (No protection fault)	
12363	00003761	EBF7	<1>	jmp	short rlp_retn	
12364			<1>			
12365			<1>	rlp_swp_err:		
12366	00003763	B804000000	<1>	mov	eax, SWP_DISK_READ_ERR ; Swap disk read error !	
12367	00003768	EBF0	<1>	jmp	short rlp_retn	
12368			<1>			
12369			<1>			
12370			<1>	copy_page_dir:		
12371			<1>	; 19/09/2015		
12372			<1>	; temporary - 07/09/2015		
12373			<1>	; 07/09/2015 (Retro UNIX 386 v1 - beginning)		
12374			<1>	;		
12375			<1>	; INPUT ->		
12376			<1>	; [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's		
12377			<1>	page directory.		
12378			<1>	; OUTPUT ->		
12379			<1>	; EAX = PHYSICAL (real/flat) ADDRESS of the child's		
12380			<1>	page directory.		
12381			<1>	; (New page directory with new page table entries.)		
12382			<1>	; (New page tables with read only copies of the parent's		
12383			<1>	pages.)		
12384			<1>	; EAX = 0 -> Error (CF = 1)		
12385			<1>	;		
12386			<1>	; Modified Registers -> none (except EAX)		
12387			<1>	;		
12388	0000376A	E8C8F8FFFF	<1>	call	allocate_page	
12389	0000376F	723E	<1>	jc	short cpd_err	
12390			<1>	;		
12391	00003771	55	<1>	push	ebp ; 20/07/2015	
12392	00003772	56	<1>	push	esi	
12393	00003773	57	<1>	push	edi	
12394	00003774	53	<1>	push	ebx	

12395	00003775	51	<1>	push ecx
12396	00003776	8B35[D3740000]	<1>	mov esi, [u.pgdir]
12397	0000377C	89C7	<1>	mov edi, eax
12398	0000377E	50	<1>	push eax ; save child's page directory address
12399			<1>	; copy PDE 0 from the parent's page dir to the child's page dir
12400			<1>	; (use same system space for all user page tables)
12401	0000377F	A5	<1>	movsd
12402	00003780	BD00004000	<1>	mov ebp, 1024*4096 ; pass the 1st 4MB (system space)
12403	00003785	B9FF030000	<1>	mov ecx, (PAGE_SIZE / 4) - 1 ; 1023
12404			<1>	cpd_0:
12405	0000378A	AD	<1>	lodsd
12406			<1>	;or eax, eax
12407			<1>	;jnz short cpd_1
12408	0000378B	A801	<1>	test al, PDE_A_PRESENT ; bit 0 = 1
12409	0000378D	7508	<1>	jnz short cpd_1
12410			<1>	; (virtual address at the end of the page table)
12411	0000378F	81C500004000	<1>	add ebp, 1024*4096 ; page size * PTE count
12412	00003795	EB0F	<1>	jmp short cpd_2
12413			<1>	cpd_1:
12414	00003797	662500F0	<1>	and ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
12415	0000379B	89C3	<1>	mov ebx, eax
12416			<1>	; EBX = Parent's page table address
12417	0000379D	E81F000000	<1>	call copy_page_table
12418	000037A2	720C	<1>	jc short cpd_p_err
12419			<1>	; EAX = Child's page table address
12420	000037A4	0C07	<1>	or al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
12421			<1>	; set bit 0, bit 1 and bit 2 to 1
12422			<1>	; (present, writable, user)
12423			<1>	cpd_2:
12424	000037A6	AB	<1>	stosd
12425	000037A7	E2E1	<1>	loop cpd_0
12426			<1>	;
12427	000037A9	58	<1>	pop eax ; restore child's page directory address
12428			<1>	cpd_3:
12429	000037AA	59	<1>	pop ecx
12430	000037AB	5B	<1>	pop ebx
12431	000037AC	5F	<1>	pop edi
12432	000037AD	5E	<1>	pop esi
12433	000037AE	5D	<1>	pop ebp
12434			<1>	cpd_err:
12435	000037AF	C3	<1>	retn
12436			<1>	cpd_p_err:
12437			<1>	; release the allocated pages missing (recover free space)
12438	000037B0	58	<1>	pop eax ; the new page directory address (physical)
12439	000037B1	8B1D[D3740000]	<1>	mov ebx, [u.pgdir] ; parent's page directory address
12440	000037B7	E8B4F9FFFF	<1>	call deallocate_page_dir
12441	000037BC	29C0	<1>	sub eax, eax ; 0
12442	000037BE	F9	<1>	stc
12443	000037BF	EBE9	<1>	jmp short cpd_3
12444			<1>	
12445			<1>	copy_page_table:
12446			<1>	; 19/09/2015
12447			<1>	; temporary - 07/09/2015
12448			<1>	; 07/09/2015 (Retro UNIX 386 v1 - beginning)
12449			<1>	;
12450			<1>	; INPUT ->
12451			<1>	; EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
12452			<1>	; EBP = page table entry index (from 'copy_page_dir')
12453			<1>	; OUTPUT ->
12454			<1>	; EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
12455			<1>	; EBP = (recent) page table index (for 'add_to_swap_queue')
12456			<1>	; CF = 1 -> error
12457			<1>	;
12458			<1>	; Modified Registers -> EBP (except EAX)
12459			<1>	;
12460	000037C1	E871F8FFFF	<1>	call allocate_page
12461	000037C6	725A	<1>	jc short cpt_err
12462			<1>	;
12463	000037C8	50	<1>	push eax ; *
12464			<1>	;push ebx
12465	000037C9	56	<1>	push esi
12466	000037CA	57	<1>	push edi
12467	000037CB	52	<1>	push edx
12468	000037CC	51	<1>	push ecx
12469			<1>	;
12470	000037CD	89DE	<1>	mov esi, ebx
12471	000037CF	89C7	<1>	mov edi, eax
12472	000037D1	89C2	<1>	mov edx, eax
12473	000037D3	81C200100000	<1>	add edx, PAGE_SIZE
12474			<1>	cpt_0:
12475	000037D9	AD	<1>	lodsd
12476	000037DA	A801	<1>	test al, PTE_A_PRESENT ; bit 0 = 1
12477	000037DC	750B	<1>	jnz short cpt_1
12478	000037DE	21C0	<1>	and eax, eax
12479	000037E0	7430	<1>	jz short cpt_2
12480			<1>	; ebp = virtual (linear) address of the memory page
12481	000037E2	E85DFFFFFF	<1>	call reload_page ; 28/04/2015
12482	000037E7	7234	<1>	jc short cpt_p_err
12483			<1>	cpt_1:
12484	000037E9	662500F0	<1>	and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
12485	000037ED	89C1	<1>	mov ecx, eax
12486			<1>	; Allocate a new page for the child process



```

12487 000037EF E843F8FFFF <1> call allocate_page
12488 000037F4 7227 <1> jc short cpt_p_err
12489 000037F6 57 <1> push edi
12490 000037F7 56 <1> push esi
12491 000037F8 89CE <1> mov esi, ecx
12492 000037FA 89C7 <1> mov edi, eax
12493 000037FC B900040000 <1> mov ecx, PAGE_SIZE/4
12494 00003801 F3A5 <1> rep movsd ; copy page (4096 bytes)
12495 00003803 5E <1> pop esi
12496 00003804 5F <1> pop edi
12497 <1> ;
12498 00003805 53 <1> push ebx
12499 00003806 50 <1> push eax
12500 00003807 89EB <1> mov ebx, ebp
12501 <1> ; ebx = virtual address of the memory page
12502 00003809 E8E6FDFFFF <1> call add_to_swap_queue
12503 0000380E 58 <1> pop eax
12504 0000380F 5B <1> pop ebx
12505 <1> ;
12506 <1> ;or ax, PTE_A_USER+PTE_A_PRESENT
12507 00003810 0C07 <1> or al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
12508 <1> cpt_2:
12509 00003812 AB <1> stosd ; EDI points to child's PTE
12510 <1> ;
12511 00003813 81C500100000 <1> add ebp, 4096 ; 20/07/2015 (next page)
12512 <1> ;
12513 00003819 39D7 <1> cmp edi, edx
12514 0000381B 72BC <1> jb short cpt_0
12515 <1> cpt_p_err:
12516 0000381D 59 <1> pop ecx
12517 0000381E 5A <1> pop edx
12518 0000381F 5F <1> pop edi
12519 00003820 5E <1> pop esi
12520 <1> ;pop ebx
12521 00003821 58 <1> pop eax ; *
12522 <1> cpt_err:
12523 00003822 C3 <1> retn
12524 <1>
12525 <1>
12526 <1> ; /// End Of MEMORY MANAGEMENT FUNCTIONS ///
12527 <1>
12528 <1> ;; Data:
12529 <1>
12530 <1> ; 09/03/2015
12531 <1> ;swpq_count: dw 0 ; count of pages on the swap que
12532 <1> ;swp_drv: dd 0 ; logical drive description table address of the swap drive/disk
12533 <1> ;swpd_size: dd 0 ; size of swap drive/disk (volume) in sectors (512 bytes).

12534 <1> ;swpd_free: dd 0 ; free page blocks (4096 bytes) on swap disk/drive (logical)
12535 <1> ;swpd_next: dd 0 ; next free page block
12536 <1> ;swpd_last: dd 0 ; last swap page block
12537 <1> %include 'sysdefs.inc' ; 09/03/2015
12538 <1> ; Retro UNIX 386 v1 Kernel - SYSDEFS.INC
12539 <1> ; Last Modification: 09/12/2015
12540 <1> ;
12541 <1> ; ////////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS //////////
12542 <1> ; (Modified from
12543 <1> ; Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
12544 <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
12545 <1> ; UNIX.ASM (MASM 6.11) --> SYSDEFS.INC (NASM 2.11)
12546 <1> ; -----
12547 <1> ;
12548 <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
12549 <1> ; (Original) Source Code by Ken Thompson (1971-1972)
12550 <1> ; <Bell Laboratories (17/3/1972)>
12551 <1> ; <Preliminary Release of UNIX Implementation Document>
12552 <1> ;
12553 <1> ; *****
12554 <1>
12555 <1> nproc equ 16 ; number of processes
12556 <1> nfiles equ 50
12557 <1> ntty equ 8 ; 8+1 -> 8 (10/05/2013)
12558 <1> nbuf equ 4 ; 6 ; 21/08/2015 - 'namei' buffer problem when nbuf > 4
12559 <1> ; NOTE: If fd0 super block buffer addres is beyond of the 1st
12560 <1> ; 32K, DMA r/w routine or someting else causes a jump to
12561 <1> ; kernel panic routine (in 'alloc' routine, in u5.s)
12562 <1> ; because of invalid buffer content (r/w error).
12563 <1> ; When all buffers are set before the end of the 1st 32k,
12564 <1> ; there is no problem!? (14/11/2015)
12565 <1>
12566 <1> ;csgmntequ 2000h ; 26/05/2013 (segment of process 1)
12567 <1> ;core equ 0 ; 19/04/2013
12568 <1> ;ecore equ 32768 - 64 ; 04/06/2013 (24/05/2013)
12569 <1> ; (if total size of argument list and arguments is 128 bytes)
12570 <1> ; maximum executable file size = 32768-(64+40+128-6) = 32530 bytes
12571 <1> ; maximum stack size = 40 bytes (+6 bytes for 'IRET' at 32570)
12572 <1> ; initial value of user's stack pointer = 32768-64-128-2 = 32574
12573 <1> ; (sp=32768-args_space-2 at the beginning of execution)
12574 <1> ; argument list offset = 32768-64-128 = 32576 (if it is 128 bytes)
12575 <1> ; 'u' structure offset (for the '/core' dump file) = 32704
12576 <1> ; '/core' dump file size = 32768 bytes
12577 <1>

```

```

12578 <1> ; 08/03/2014
12579 <1> ;sdsegmnt equ6C0h ; 256*16 bytes (swap data segment size for 16 processes)

12580 <1> ; 19/04/2013 Retro UNIX 8086 v1 feaure only !
12581 <1> ;;sdsegmnt equ 740h ; swap data segment (for user structures and registers)
12582 <1>
12583 <1> ; 30/08/2013
12584 <1> time_count equ 4 ; 10 --> 4 01/02/2014
12585 <1>
12586 <1> ; 05/02/2014
12587 <1> ; process status
12588 <1> ;SFREE equ 0
12589 <1> ;SRUN equ 1
12590 <1> ;SWAIT equ 2
12591 <1> ;SZOMB equ 3
12592 <1> ;SSLEEPequ 4 ; Retro UNIX 8086 V1 extension (for sleep and wakeup)
12593 <1>
12594 <1> ; 09/03/2015
12595 <1> userdata equ 80000h ; user structure data address for current user ; temporary
12596 <1> swap_queue equ 90000h - 2000h ; swap queue address ; temporary
12597 <1> swap_alloc_table equ 0D0000h ; swap allocation table address ; temporary
12598 <1>
12599 <1> ; 17/09/2015
12600 <1> ESPACE equ 48 ; [u.usp] (at 'sysent') - [u.sp] value for error return
12601 <1>
12602 <1> ; 21/09/2015 (36)
12603 <1> ; 01/07/2015 (35)
12604 <1> ; 14/07/2013 (0-34)
12605 <1> ; UNIX v1 system calls
12606 <1> _rele equ 0
12607 <1> _exit equ 1
12608 <1> _fork equ 2
12609 <1> _read equ 3
12610 <1> _write equ 4
12611 <1> _open equ 5
12612 <1> _close equ 6
12613 <1> _wait equ 7
12614 <1> _creat equ 8
12615 <1> _link equ 9
12616 <1> _unlinkequ 10
12617 <1> _exec equ 11
12618 <1> _chdir equ 12
12619 <1> _time equ 13
12620 <1> _mkdir equ 14
12621 <1> _chmod equ 15
12622 <1> _chown equ 16
12623 <1> _break equ 17
12624 <1> _stat equ 18
12625 <1> _seek equ 19
12626 <1> _tell equ 20
12627 <1> _mount equ 21
12628 <1> _umountequ 22
12629 <1> _setuidequ 23
12630 <1> _getuidequ 24
12631 <1> _stime equ 25
12632 <1> _quit equ 26
12633 <1> _intr equ 27
12634 <1> _fstat equ 28
12635 <1> _emt equ 29
12636 <1> _mdate equ 30
12637 <1> _stty equ 31
12638 <1> _gtty equ 32
12639 <1> _ilginsequ 33
12640 <1> _sleep equ 34 ; Retro UNIX 8086 v1 feature only !
12641 <1> _msg equ 35 ; Retro UNIX 386 v1 feature only !
12642 <1> _geterrequ 36 ; Retro UNIX 386 v1 feature only !
12643 <1>
12644 <1> %macro sys 1-4
12645 <1> ; 03/09/2015
12646 <1> ; 13/04/2015
12647 <1> ; Retro UNIX 386 v1 system call.
12648 <1> %if %0 >= 2
12649 <1> mov ebx, %2
12650 <1> %if %0 >= 3
12651 <1> mov ecx, %3
12652 <1> %if %0 = 4
12653 <1> mov edx, %4
12654 <1> %endif
12655 <1> %endif
12656 <1> %endif
12657 <1> mov eax, %1
12658 <1> int 30h
12659 <1> %endmacro
12660 <1>
12661 <1> ; 13/05/2015 - ERROR CODES
12662 <1> ERR_FILE_NOT_OPEN equ 10 ; 'file not open !' error
12663 <1> ERR_FILE_ACCESS equ 11 ; 'permission denied !' error
12664 <1> ; 14/05/2015
12665 <1> ERR_DIR_ACCESS equ 11 ; 'permission denied !' error
12666 <1> ERR_FILE_NOT_FOUND equ 12 ; 'file not found !' error
12667 <1> ERR_TOO_MANY_FILES equ 13 ; 'too many open files !' error
12668 <1> ERR_DIR_EXISTS equ 14 ; 'directory already exists !' error

```

```

12669 <1> ; 16/05/2015
12670 <1> ERR_DRV_NOT_RDY equ 15 ; 'drive not ready !' error
12671 <1> ; 18/05/2015
12672 <1> ERR_DEV_NOT_RDY equ 15 ; 'device not ready !' error
12673 <1> ERR_DEV_ACCESS equ 11 ; 'permission denied !' error
12674 <1> ERR_DEV_NOT_OPEN equ 10 ; 'device not open !' error
12675 <1> ; 07/06/2015
12676 <1> ERR_FILE_EOF equ 16 ; 'end of file !' error
12677 <1> ERR_DEV_VOL_SIZE equ 16 ; 'out of volume' error
12678 <1> ; 09/06/2015
12679 <1> ERR_DRV_READ equ 17 ; 'disk read error !'
12680 <1> ERR_DRV_WRITE equ 18 ; 'disk write error !'
12681 <1> ; 16/06/2015
12682 <1> ERR_NOT_DIR equ 19 ; 'not a (valid) directory !' error
12683 <1> ERR_FILE_SIZE equ 20 ; 'file size error !'
12684 <1> ; 22/06/2015
12685 <1> ERR_NOT_SUPERUSER equ 11 ; 'permission denied !' error
12686 <1> ERR_NOT_OWNER equ 11 ; 'permission denied !' error
12687 <1> ERR_NOT_FILE equ 11 ; 'permission denied !' error
12688 <1> ; 23/06/2015
12689 <1> ERR_FILE_EXISTS equ 14 ; 'file already exists !' error
12690 <1> ERR_DRV_NOT_SAME equ 21 ; 'not same drive !' error
12691 <1> ERR_DIR_NOT_FOUND equ 12 ; 'directory not found !' error
12692 <1> ERR_NOT_EXECUTABLE equ 22 ; 'not executable file !' error
12693 <1> ; 27/06/2015
12694 <1> ERR_INV_PARAMETER equ 23 ; 'invalid parameter !' error
12695 <1> ERR_INV_DEV_NAME equ 24 ; 'invalid device name !' error
12696 <1> ; 29/06/2015
12697 <1> ERR_TIME_OUT equ 25 ; 'time out !' error
12698 <1> ERR_DEV_NOT_RESP equ 25 ; 'device not responding !' error
12699 <1>
12700 <1> ; 26/08/2015
12701 <1> ; 24/07/2015
12702 <1> ; 24/06/2015
12703 <1> MAX_ARG_LEN equ 256 ; max. length of sys exec arguments
12704 <1> ; 01/07/2015
12705 <1> MAX_MSG_LEN equ 255 ; max. msg length for 'sysmsg'
12706 <1> ;
12707 %include 'u0.s' ; 15/03/2015
12708 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS0.INC
12709 <1> ; Last Modification: 21/11/2015
12710 <1> ; -----
12711 <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
12712 <1> ; (v0.1 - Beginning: 11/07/2012)
12713 <1> ;
12714 <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
12715 <1> ; (Original) Source Code by Ken Thompson (1971-1972)
12716 <1> ; <Bell Laboratories (17/3/1972)>
12717 <1> ; <Preliminary Release of UNIX Implementation Document>
12718 <1> ;
12719 <1> ; Retro UNIX 8086 v1 - U0.ASM (28/07/2014) //// UNIX v1 -> u0.s
12720 <1> ;
12721 <1> ; *****
12722 <1>
12723 <1> sys_init:
12724 <1> ; 18/10/2015
12725 <1> ; 28/08/2015
12726 <1> ; 24/08/2015
12727 <1> ; 14/08/2015
12728 <1> ; 24/07/2015
12729 <1> ; 02/07/2015
12730 <1> ; 01/07/2015
12731 <1> ; 23/06/2015
12732 <1> ; 15/04/2015
12733 <1> ; 13/04/2015
12734 <1> ; 11/03/2015 (Retro UNIX 386 v1 - Beginning)
12735 <1> ; 28/07/2014 (Retro UNIX 8086 v1)
12736 <1> ;
12737 <1> ;call ldrv_init ; Logical drive description tables initialization
12738 <1> ;
12739 <1> ; 14/02/2014
12740 <1> ; 14/07/2013
12741 00003823 66B82900 <1> mov ax, 41
12742 00003827 66A3[70740000] <1> mov [rootdir], ax
12743 0000382D 66A3[84740000] <1> mov [u.cdir], ax
12744 00003833 2401 <1> and al, 1 ; 15/04/2015
12745 00003835 A2[C9740000] <1> mov [u.uno], al
12746 0000383A 66A3[6E740000] <1> mov [mpid], ax
12747 00003840 66A3[76710000] <1> mov [p.pid], ax
12748 00003846 A2[06720000] <1> mov [p.stat], al ; SRUN, 05/02/2014
12749 <1> ;
12750 0000384B B004 <1> mov al, time_count ; 30/08/2013
12751 0000384D A2[BC740000] <1> mov [u.quant], al ; 14/07/2013
12752 <1> ; 02/07/2015
12753 00003852 A1[A8700000] <1> mov eax, [k_page_dir]
12754 <1> ;sub eax, eax
12755 00003857 A3[D3740000] <1> mov [u.pgdir], eax ; reset
12756 <1> ; 18/10/2015
12757 <1> ;mov [u.ppgdir], eax ; 0
12758 <1> ;
12759 0000385C E856030000 <1> call epoch
12760 00003861 A3[B47E0000] <1> mov [s.time], eax ; 13/03/2015

```

12761		<1>	; 17/07/2013
12762	00003866 E8B0060000	<1>	call bf_init ; buffer initialization
12763		<1>	; 23/06/2015
12764	0000386B E8C7F7FFFF	<1>	call allocate_page
12765		<1>	;;jc error
12766	00003870 0F829C000000	<1>	jc panic ; jc short panic (01/07/2015)
12767	00003876 A3[CA740000]	<1>	mov [u.upage], eax ; user structure page
12768	0000387B A3[16720000]	<1>	mov [p.upage], eax
12769		<1>	;
12770	00003880 E82CF8FFFF	<1>	call clear_page
12771		<1>	;
12772		<1>	; 14/08/2015
12773	00003885 FA	<1>	cli
12774		<1>	; 14/03/2015
12775		<1>	; 17/01/2014
12776	00003886 E8DE010000	<1>	call sp_init ; serial port initialization
12777		<1>	; 14/08/2015
12778	0000388B FB	<1>	sti
12779		<1>	;
12780		<1>	; 30/06/2015
12781		<1>	;mov esi, kernel_init_ok_msg
12782		<1>	;call print_msg
12783		<1>	;
12784	0000388C 30DB	<1>	xor bl, bl ; video page 0
12785		<1>	vp_clr_nxt: ; clear video pages (reset cursor positions)
12786	0000388E E8FE2D0000	<1>	call vp_clr ; 17/07/2013
12787	00003893 FEC3	<1>	inc bl
12788	00003895 80FB08	<1>	cmp bl, 8
12789	00003898 72F4	<1>	jnb short vp_clr_nxt
12790		<1>	;
12791		<1>	; 24/07/2015
12792		<1>	;push KDATA
12793		<1>	;push esp
12794		<1>	;mov [tss.esp0], esp
12795		<1>	;mov word [tss.ss0], KDATA
12796		<1>	;
12797		<1>	; 24/08/2015
12798		<1>	;; temporary (01/07/2015)
12799	0000389A C605[BC740000]04	<1>	mov byte [u.quant], time_count ; 4
12800		<1>	; it is not needed here !
12801		<1>	;;inc byte [u.kcall] ; 'the caller is kernel' sign
12802	000038A1 FE0D[77740000]	<1>	dec byte [sysflg] ; FFh = ready for system call
12803		<1>	; 0 = executing a system call
12804		<1>	;;sys _msg, kernel_init_ok_msg, 255, 0
12805		<1>	;
12806		<1>	;;; 06/08/2015
12807		<1>	;;;call getch ; wait for a key stroke
12808		<1>	;;mov ecx, 0FFFFFFFh
12809		<1>	;;sys_init_msg_wait:
12810		<1>	;; push ecx
12811		<1>	;; mov al, 1
12812		<1>	;; mov ah, [ptty] ; active (current) video page
12813		<1>	;; call getc_n
12814		<1>	;; pop ecx
12815		<1>	;; jnz short sys_init_msg_ok
12816		<1>	;; loop sys_init_msg_wait
12817		<1>	;
12818		<1>	;;sys_init_msg_ok:
12819		<1>	; 28/08/2015 (initial settings for the 1st 'rswap')
12820	000038A7 6A10	<1>	push KDATA ; ss
12821	000038A9 54	<1>	push esp
12822	000038AA 9C	<1>	pushfd
12823	000038AB 6A08	<1>	push KCODE ; cs
12824	000038AD 68[E1380000]	<1>	push init_exec ; eip
12825	000038B2 8925[78740000]	<1>	mov [u.sp], esp
12826	000038B8 1E	<1>	push ds
12827	000038B9 06	<1>	push es
12828	000038BA 0FA0	<1>	push fs
12829	000038BC 0FA8	<1>	push gs
12830	000038BE 60	<1>	pushad
12831	000038BF 8925[7C740000]	<1>	mov [u.usp], esp
12832	000038C5 E8481B0000	<1>	call wswap ; save current user (u) structure, user registers
12833		<1>	; and interrupt return components (for IRET)
12834	000038CA 61	<1>	popad
12835	000038CB 6658	<1>	pop ax ; gs
12836	000038CD 6658	<1>	pop ax ; fs
12837	000038CF 6658	<1>	pop ax ; es
12838	000038D1 6658	<1>	pop ax ; ds
12839	000038D3 58	<1>	pop eax ; eip (init_exec)
12840	000038D4 6658	<1>	pop ax ; cs (KCODE)
12841	000038D6 58	<1>	pop eax ; E-FLAGS
12842	000038D7 58	<1>	pop eax ; esp
12843	000038D8 6658	<1>	pop ax ; ss (KDATA)
12844		<1>	;
12845	000038DA 31C0	<1>	xor eax, eax ; 0
12846	000038DC A3[D7740000]	<1>	mov [u.ppgdir], eax ; reset (to zero) for '/etc/init'
12847		<1>	;
12848		<1>	; 02/07/2015
12849		<1>	; [u.pgdir] = [k_page_dir]
12850		<1>	; [u.ppgdir] = 0 (page dir of the parent process)
12851		<1>	; (The caller is os kernel sign for 'sysexec')
12852		<1>	init_exec:

12853		<1>	; 13/03/2013
12854		<1>	; 24/07/2013
12855	000038E1 BB[08390000]	<1>	mov ebx, init_file
12856	000038E6 B9[00390000]	<1>	mov ecx, init_argp
12857		<1>	; EBX contains 'etc/init' asciiz file name address
12858		<1>	; ECX contains address of argument list pointer
12859		<1>	;
12860		<1>	;dec byte [sysflg] ; FFh = ready for system call
12861		<1>	; 0 = executing a system call
12862		<1>	sys _exec ; execute file
12863		<2>	
12864		<2>	
12865		<2>	
12866		<2>	%if %0 >= 2
12867		<2>	mov ebx, %2
12868		<2>	%if %0 >= 3
12869		<2>	mov ecx, %3
12870		<2>	%if %0 = 4
12871		<2>	mov edx, %4
12872		<2>	%endif
12873		<2>	%endif
12874		<2>	%endif
12875	000038EB B80B000000	<2>	mov eax, %1
12876	000038F0 CD30	<2>	int 30h
12877	000038F2 731E	<1>	jnc short panic
12878		<1>	;
12879	000038F4 BE[586D0000]	<1>	mov esi, etc_init_err_msg
12880	000038F9 E837000000	<1>	call print_msg
12881	000038FE EB1C	<1>	jmp short key_to_reboot
12882		<1>	
12883		<1>	;align 4
12884		<1>	init_argp:
12885	00003900 [08390000]00000000	<1>	dd init_file, 0 ; 23/06/2015 (dw -> dd)
12886		<1>	init_file:
12887		<1>	; 24/08/2015
12888	00003908 2F6574632F696E6974-	<1>	db '/etc/init', 0
12889	00003911 00	<1>	
12890		<1>	panic:
12891		<1>	; 13/03/2015 (Retro UNIX 386 v1)
12892		<1>	; 07/03/2014 (Retro UNIX 8086 v1)
12893	00003912 BE[3D6D0000]	<1>	mov esi, panic_msg
12894	00003917 E819000000	<1>	call print_msg
12895		<1>	key_to_reboot:
12896		<1>	; 15/11/2015
12897	0000391C E8A72B0000	<1>	call getch
12898		<1>	; wait for a character from the current tty
12899		<1>	;
12900	00003921 B00A	<1>	mov al, 0Ah
12901	00003923 8A1D[D6700000]	<1>	mov bl, [ptty] ; [active_page]
12902	00003929 B407	<1>	mov ah, 07h ; Black background,
12903		<1>	; light gray forecolor
12904	0000392B E8EDDBFFFF	<1>	call write_tty
12905	00003930 E96DD8FFFF	<1>	jmp cpu_reset
12906		<1>	
12907		<1>	print_msg:
12908		<1>	; 01/07/2015
12909		<1>	; 13/03/2015 (Retro UNIX 386 v1)
12910		<1>	; 07/03/2014 (Retro UNIX 8086 v1)
12911		<1>	; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
12912		<1>	;
12913		<1>	;
12914	00003935 AC	<1>	lodsb
12915		<1>	pmsg1:
12916	00003936 56	<1>	push esi
12917	00003937 0FB61D[D6700000]	<1>	movzx ebx, byte [ptty]
12918	0000393E B407	<1>	mov ah, 07h ; Black background, light gray forecolor
12919	00003940 E8D8DBFFFF	<1>	call write_tty
12920	00003945 5E	<1>	pop esi
12921	00003946 AC	<1>	lodsb
12922	00003947 20C0	<1>	and al, al
12923	00003949 75EB	<1>	jnz short pmsg1
12924	0000394B C3	<1>	retn
12925		<1>	
12926		<1>	ctrlbrk:
12927		<1>	; 12/11/2015
12928		<1>	; 13/03/2015 (Retro UNIX 386 v1)
12929		<1>	; 06/12/2013 (Retro UNIX 8086 v1)
12930		<1>	;
12931		<1>	; INT 1Bh (control+break) handler
12932		<1>	;
12933		<1>	; Retro Unix 8086 v1 feature only!
12934		<1>	;
12935	0000394C 66833D[BE740000]00	<1>	cmp word [u.intr], 0
12936	00003954 7645	<1>	jna short cbrk4
12937		<1>	cbrk0:
12938		<1>	; 12/11/2015
12939		<1>	; 06/12/2013
12940	00003956 66833D[C0740000]00	<1>	cmp word [u.quit], 0
12941	0000395E 743B	<1>	jz short cbrk4
12942		<1>	;
12943		<1>	; 20/09/2013
12944	00003960 6650	<1>	push ax

12945	00003962	A0[D6700000]	<1>	mov	al, [ptty]
12946			<1>	;	
12947			<1>	;	12/11/2015
12948			<1>	;	
12949			<1>	;	ctrl+break (EOT, CTRL+D) from serial port
12950			<1>	;	or ctrl+break from console (pseudo) tty
12951			<1>	;	(!redirection!)
12952			<1>	;	
12953	00003967	3C08	<1>	cmp	al, 8 ; serial port tty nums > 7
12954	00003969	7211	<1>	jb	short cbrk1 ; console (pseudo) tty
12955			<1>	;	
12956			<1>	;	Serial port interrupt handler sets [ptty]
12957			<1>	;	to the port's tty number (as temporary).
12958			<1>	;	
12959			<1>	;	If active process is using a stdin or
12960			<1>	;	stdout redirection (by the shell),
12961			<1>	;	console tty keyboard must be available
12962			<1>	;	to terminate running process,
12963			<1>	;	in order to prevent a deadlock.
12964			<1>	;	
12965	0000396B	52	<1>	push	edx
12966	0000396C	0FB615[C9740000]	<1>	movzx	edx, byte [u.uno]
12967	00003973	3A82[D5710000]	<1>	cmp	al, [edx+p.ttyc-1] ; console tty (rw)
12968	00003979	5A	<1>	pop	edx
12969	0000397A	7412	<1>	je	short cbrk2
12970			<1>	cbrk1:	
12971	0000397C	FEC0	<1>	inc	al ; [u.ttyp] : 1 based tty number
12972			<1>	;	06/12/2013
12973	0000397E	3A05[B0740000]	<1>	cmp	al, [u.ttyp] ; recent open tty (r)
12974	00003984	7408	<1>	je	short cbrk2
12975	00003986	3A05[B1740000]	<1>	cmp	al, [u.ttyp+1] ; recent open tty (w)
12976	0000398C	750B	<1>	jne	short cbrk3
12977			<1>	cbrk2:	
12978			<1>	;;	06/12/2013
12979			<1>	;mov	ax, [u.quit]
12980			<1>	;and	ax, ax
12981			<1>	;jz	short cbrk3
12982			<1>	;	
12983	0000398E	6631C0	<1>	xor	ax, ax ; 0
12984	00003991	6648	<1>	dec	ax
12985			<1>	;	0FFFFh = 'ctrl+brk' keystroke
12986	00003993	66A3[C0740000]	<1>	mov	[u.quit], ax
12987			<1>	cbrk3:	
12988	00003999	6658	<1>	pop	ax
12989			<1>	cbrk4:	
12990	0000399B	C3	<1>	retn	
12991			<1>		
12992			<1>	com2_int:	
12993			<1>	;	07/11/2015
12994			<1>	;	24/10/2015
12995			<1>	;	23/10/2015
12996			<1>	;	14/03/2015 (Retro UNIX 386 v1 - Beginning)
12997			<1>	;	28/07/2014 (Retro UNIX 8086 v1)
12998			<1>	;	< serial port 2 interrupt handler >
12999			<1>	;	
13000	0000399C	890424	<1>	mov	[esp], eax ; overwrite call return address
13001			<1>	;	push eax
13002	0000399F	66B80900	<1>	mov	ax, 9
13003	000039A3	EB07	<1>	jmp	short comm_int
13004			<1>	com1_int:	
13005			<1>	;	07/11/2015
13006			<1>	;	24/10/2015
13007	000039A5	890424	<1>	mov	[esp], eax ; overwrite call return address
13008			<1>	;	23/10/2015
13009			<1>	;	push eax
13010	000039A8	66B80800	<1>	mov	ax, 8
13011			<1>	comm_int:	
13012			<1>	;	20/11/2015
13013			<1>	;	18/11/2015
13014			<1>	;	17/11/2015
13015			<1>	;	16/11/2015
13016			<1>	;	09/11/2015
13017			<1>	;	08/11/2015
13018			<1>	;	07/11/2015
13019			<1>	;	06/11/2015 (serial4.asm, 'serial')
13020			<1>	;	01/11/2015
13021			<1>	;	26/10/2015
13022			<1>	;	23/10/2015
13023	000039AC	53	<1>	push	ebx
13024	000039AD	56	<1>	push	esi
13025	000039AE	57	<1>	push	edi
13026	000039AF	1E	<1>	push	ds
13027	000039B0	06	<1>	push	es
13028			<1>	;	18/11/2015
13029	000039B1	0F20DB	<1>	mov	ebx, cr3
13030	000039B4	53	<1>	push	ebx ; ****
13031			<1>	;	
13032	000039B5	51	<1>	push	ecx ; ***
13033	000039B6	52	<1>	push	edx ; **
13034			<1>	;	
13035	000039B7	BB10000000	<1>	mov	ebx, KDATA
13036	000039BC	8EDB	<1>	mov	ds, bx

13037	000039BE	8EC3	<1>	mov	es, bx
13038			<1>	;	
13039	000039C0	8B0D[A8700000]	<1>	mov	ecx, [k_page_dir]
13040	000039C6	0F22D9	<1>	mov	cr3, ecx
13041			<1>	;	20/11/2015
13042			<1>	;	Interrupt identification register
13043	000039C9	66BAFA02	<1>	mov	dx, 2FAh ; COM2
13044			<1>	;	
13045	000039CD	3C08	<1>	cmp	al, 8
13046	000039CF	7702	<1>	ja	short com_i0
13047			<1>	;	
13048			<1>	;	20/11/2015
13049			<1>	;	17/11/2015
13050			<1>	;	16/11/2015
13051			<1>	;	15/11/2015
13052			<1>	;	24/10/2015
13053			<1>	;	14/03/2015 (Retro UNIX 386 v1 - Beginning)
13054			<1>	;	28/07/2014 (Retro UNIX 8086 v1)
13055			<1>	;	< serial port 1 interrupt handler >
13056			<1>	;	
13057	000039D1	FEC6	<1>	inc	dh ; 3FAh ; COM1 Interrupt id. register
13058			<1>	com_i0:	
13059			<1>	;	push eax ; *
13060			<1>	;	07/11/2015
13061	000039D3	A2[16710000]	<1>	mov	byte [ccomport], al
13062			<1>	;	09/11/2015
13063	000039D8	0FB7D8	<1>	movzx	ebx, ax ; 8 or 9
13064			<1>	;	17/11/2015
13065			<1>	;	reset request for response status
13066	000039DB	88A3[0C710000]	<1>	mov	[ebx+req_resp-8], ah ; 0
13067			<1>	;	
13068			<1>	;	20/11/2015
13069	000039E1	EC	<1>	in	al, dx ; read interrupt id. register
13070	000039E2	EB00	<1>	JMP	\$+2 ; I/O DELAY
13071	000039E4	2404	<1>	and	al, 4 ; received data available?
13072	000039E6	7470	<1>	jz	short com_eoi ; (transmit. holding reg. empty)
13073			<1>	;	
13074			<1>	;	20/11/2015
13075	000039E8	80EA02	<1>	sub	dl, 3FAh-3F8h ; data register (3F8h, 2F8h)
13076	000039EB	EC	<1>	in	al, dx ; read character
13077			<1>	;	JMP \$+2 ; I/O DELAY
13078			<1>	;	08/11/2015
13079			<1>	;	07/11/2015
13080	000039EC	89DE	<1>	mov	esi, ebx
13081	000039EE	89DF	<1>	mov	edi, ebx
13082	000039F0	81C6[10710000]	<1>	add	esi, rchar - 8 ; points to last received char
13083	000039F6	81C7[12710000]	<1>	add	edi, schar - 8 ; points to last sent char
13084	000039FC	8806	<1>	mov	[esi], al ; received char (current char)
13085			<1>	;	query
13086	000039FE	20C0	<1>	and	al, al
13087	00003A00	7527	<1>	jnz	short com_i2
13088			<1>	;	response
13089			<1>	;	17/11/2015
13090			<1>	;	set request for response status
13091	00003A02	FE83[0C710000]	<1>	inc	byte [ebx+req_resp-8] ; 1
13092			<1>	;	
13093	00003A08	6683C205	<1>	add	dx, 3FDh-3F8h ; (3FDh, 2FDh)
13094	00003A0C	EC	<1>	in	al, dx ; read line status register
13095	00003A0D	EB00	<1>	JMP	\$+2 ; I/O DELAY
13096	00003A0F	2420	<1>	and	al, 20h ; transmitter holding reg. empty?
13097	00003A11	7445	<1>	jz	short com_eoi ; no
13098	00003A13	B0FF	<1>	mov	al, 0FFh ; response
13099	00003A15	6683EA05	<1>	sub	dx, 3FDh-3F8h ; data port (3F8h, 2F8h)
13100	00003A19	EE	<1>	out	dx, al ; send on serial port
13101			<1>	;	17/11/2015
13102	00003A1A	803F00	<1>	cmp	byte [edi], 0 ; query ? (schar)
13103	00003A1D	7502	<1>	jne	short com_i1 ; no
13104	00003A1F	8807	<1>	mov	[edi], al ; 0FFh (responded)
13105			<1>	com_i1:	
13106			<1>	;	17/11/2015
13107			<1>	;	reset request for response status (again)
13108	00003A21	FE8B[0C710000]	<1>	dec	byte [ebx+req_resp-8] ; 0
13109	00003A27	EB2F	<1>	jmp	short com_eoi
13110			<1>	com_i2:	
13111			<1>	;	08/11/2015
13112	00003A29	3CFF	<1>	cmp	al, 0FFh ; (response ?)
13113	00003A2B	7417	<1>	je	short com_i3 ; (check for response signal)
13114			<1>	;	07/11/2015
13115	00003A2D	3C04	<1>	cmp	al, 04h ; EOT
13116	00003A2F	751C	<1>	jne	short com_i4
13117			<1>	;	EOT = 04h (End of Transmit) - 'CTRL + D'
13118			<1>	;	(an EOT char is supposed as a ctrl+brk from the terminal)
13119			<1>	;	08/11/2015
13120			<1>	;	pty -> tty 0 to 7 (pseudo screens)
13121	00003A31	861D[D6700000]	<1>	xchg	bl, [pty] ; tty number (8 or 9)
13122	00003A37	E810FFFFFF	<1>	call	ctrlbrk
13123	00003A3C	861D[D6700000]	<1>	xchg	[pty], bl ; (restore pty value and BL value)
13124			<1>	;	mov al, 04h ; EOT
13125			<1>	;	08/11/2015
13126	00003A42	EB09	<1>	jmp	short com_i4
13127			<1>	com_i3:	
13128			<1>	;	08/11/2015

13129		<1>	; If 0FFh has been received just after a query
13130		<1>	; (schar, ZERO), it is a response signal.
13131		<1>	; 17/11/2015
13132	00003A44 803F00	<1>	cmp byte [edi], 0 ; query ? (schar)
13133	00003A47 7704	<1>	ja short com_i4 ; no
13134		<1>	; reset query status (schar)
13135	00003A49 8807	<1>	mov [edi], al ; 0FFh
13136	00003A4B FEC0	<1>	inc al ; 0
13137		<1>	com_i4:
13138		<1>	; 27/07/2014
13139		<1>	; 09/07/2014
13140	00003A4D D0E3	<1>	shl bl, 1
13141	00003A4F 81C3[D8700000]	<1>	add ebx, ttychr
13142		<1>	; 23/07/2014 (always overwrite)
13143		<1>	;;cmp word [ebx], 0
13144		<1>	;;ja short com_eoi
13145		<1>	;
13146	00003A55 668903	<1>	mov [ebx], ax ; Save ascii code
13147		<1>	; scan code = 0
13148		<1>	com_eoi:
13149		<1>	;mov al, 20h
13150		<1>	;out 20h, al ; end of interrupt
13151		<1>	;
13152		<1>	; 07/11/2015
13153		<1>	;pop eax ; *
13154	00003A58 A0[16710000]	<1>	mov al, byte [ccomport] ; current COM port
13155		<1>	; al = tty number (8 or 9)
13156	00003A5D E88B1A0000	<1>	call wakeup
13157		<1>	com_iret:
13158		<1>	; 23/10/2015
13159	00003A62 5A	<1>	pop edx ; **
13160	00003A63 59	<1>	pop ecx ; ***
13161		<1>	; 18/11/2015
13162		<1>	;pop eax ; ****
13163		<1>	;mov cr3, eax
13164		<1>	;jmp iiret
13165	00003A64 E94BD0FFFF	<1>	jmp iiretp
13166		<1>	
13167		<1>	;iiretp: ; 01/09/2015
13168		<1>	;; 28/08/2015
13169		<1>	;pop eax ; (*) page directory
13170		<1>	;mov cr3, eax
13171		<1>	;iiret:
13172		<1>	;; 22/08/2014
13173		<1>	;mov al, 20h ; END OF INTERRUPT COMMAND TO 8259
13174		<1>	;out 20h, al ; 8259 PORT
13175		<1>	;;
13176		<1>	;pop es
13177		<1>	;pop ds
13178		<1>	;pop edi
13179		<1>	;pop esi
13180		<1>	;pop ebx ; 29/08/2014
13181		<1>	;pop eax
13182		<1>	;iretd
13183		<1>	
13184		<1>	sp_init:
13185		<1>	; 07/11/2015
13186		<1>	; 29/10/2015
13187		<1>	; 26/10/2015
13188		<1>	; 23/10/2015
13189		<1>	; 29/06/2015
13190		<1>	; 14/03/2015 (Retro UNIX 386 v1 - 115200 baud)
13191		<1>	; 28/07/2014 (Retro UNIX 8086 v1 - 9600 baud)
13192		<1>	; Initialization of Serial Port Communication Parameters
13193		<1>	; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
13194		<1>	; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
13195		<1>	;
13196		<1>	; ((Modified registers: EAX, ECX, EDX, EBX))
13197		<1>	;
13198		<1>	; INPUT: (29/06/2015)
13199		<1>	; AL = 0 for COM1
13200		<1>	; 1 for COM2
13201		<1>	; AH = Communication parameters
13202		<1>	;
13203		<1>	; (*) Communication parameters (except BAUD RATE):
13204		<1>	; Bit 4 3 2 1 0
13205		<1>	; -PARITY-- STOP BIT -WORD LENGTH-
13206		<1>	; this one --> 00 = none 0 = 1 bit 11 = 8 bits
13207		<1>	; 01 = odd 1 = 2 bits 10 = 7 bits
13208		<1>	; 11 = even
13209		<1>	; Baud rate setting bits: (29/06/2015)
13210		<1>	; Retro UNIX 386 v1 feature only !
13211		<1>	; Bit 7 6 5   Baud rate
13212		<1>	;
13213		<1>	; value 0 0 0   Default (Divisor = 1)
13214		<1>	; 0 0 1   9600 (12)
13215		<1>	; 0 1 0   19200 (6)
13216		<1>	; 0 1 1   38400 (3)
13217		<1>	; 1 0 0   14400 (8)
13218		<1>	; 1 0 1   28800 (4)
13219		<1>	; 1 1 0   57600 (2)
13220		<1>	; 1 1 1   115200 (1)



```

13221
13222
13223
13224
13225
13226
13227
13228
13229
13230 00003A69 BB[12710000]
13231 00003A6E 66BAF803
13232
13233 00003A72 66B90103
13234 00003A76 E86F000000
13235 00003A7B A880
13236 00003A7D 7410
13237
13238
13239 00003A7F 80EA05
13240 00003A82 66B90E03
13241 00003A86 E85F000000
13242 00003A8B A880
13243 00003A8D 7508
13244
13245
13246
13247
13248 00003A8F C603E3
13249 00003A92 E8DC000000
13250
13251 00003A97 43
13252 00003A98 66BAF802
13253
13254 00003A9C 66B90103
13255 00003AA0 E845000000
13256 00003AA5 A880
13257 00003AA7 7410
13258
13259
13260 00003AA9 80EA05
13261 00003AAC 66B90E03
13262 00003AB0 E835000000
13263 00003AB5 A880
13264 00003AB7 7530
13265
13266 00003AB9 C603E3
13267
13268
13269
13270
13271 00003ABC 9C
13272 00003ABD FA
13273
13274 00003ABE 66BAFC02
13275 00003AC2 EC
13276 00003AC3 EB00
13277 00003AC5 0C08
13278 00003AC7 EE
13279 00003AC8 EB00
13280 00003ACA 66BAF902
13281 00003ACE EC
13282 00003ACF EB00
13283
13284 00003AD1 0C03
13285 00003AD3 EE
13286 00003AD4 EB00
13287 00003AD6 E421
13288 00003AD8 EB00
13289 00003ADA 24F7
13290 00003ADC E621
13291
13292
13293 00003ADE B8[9C390000]
13294 00003AE3 A3[7B3F0000]
13295
13296 00003AE8 9D
13297
13298 00003AE9 C3
13299
13300
13301
13302
13303 00003AEA FEC2
13304 00003AEC B000
13305 00003AEE EE
13306 00003AEF EB00
13307 00003AF1 80C202
13308 00003AF4 B080
13309 00003AF6 EE
13310
13311
13312 00003AF7 80EA03

<1>
<1> ; References:
<1> ; (1) IBM PC-XT Model 286 BIOS Source Code
<1> ; RS232.ASM --- 10/06/1985 COMMUNICATIONS BIOS (RS232)
<1> ; (2) Award BIOS 1999 - ATORGS.ASM
<1> ; (3) http://wiki.osdev.org/Serial_Ports
<1> ;
<1> ; Set communication parameters for COM1 (= 03h)
<1> ;
<1> mov ebx, comlp ; COM1 parameters
<1> mov dx, 3F8h ; COM1
<1> ; 29/10/2015
<1> mov cx, 301h ; divisor = 1 (115200 baud)
<1> call sp_i3 ; call A4
<1> test al, 80h
<1> jz short sp_i0 ; OK..
<1> ; Error !
<1> ;mov dx, 3F8h
<1> sub dl, 5 ; 3FDh -> 3F8h
<1> mov cx, 30Eh ; divisor = 12 (9600 baud)
<1> call sp_i3 ; call A4
<1> test al, 80h
<1> jnz short sp_i1
<1> sp_i0:
<1> ; (Note: Serial port interrupts will be disabled here...)
<1> ; (INT 14h initialization code disables interrupts.)
<1> ;
<1> mov byte [ebx], 0E3h ; 11100011b
<1> call sp_i5 ; 29/06/2015
<1> sp_i1:
<1> inc ebx
<1> mov dx, 2F8h ; COM2
<1> ; 29/10/2015
<1> mov cx, 301h ; divisor = 1 (115200 baud)
<1> call sp_i3 ; call A4
<1> test al, 80h
<1> jz short sp_i2 ; OK..
<1> ; Error !
<1> ;mov dx, 2F8h
<1> sub dl, 5 ; 2FDh -> 2F8h
<1> mov cx, 30Eh ; divisor = 12 (9600 baud)
<1> call sp_i3 ; call A4
<1> test al, 80h
<1> jnz short sp_i7
<1> sp_i2:
<1> mov byte [ebx], 0E3h ; 11100011b
<1> sp_i6:
<1> ;; COM2 - enabling IRQ 3
<1> ; 07/11/2015
<1> ; 26/10/2015
<1> pushf
<1> cli
<1> ;
<1> mov dx, 2FCh ; modem control register
<1> in al, dx ; read register
<1> JMP $+2 ; I/O DELAY
<1> or al, 8 ; enable bit 3 (OUT2)
<1> out dx, al ; write back to register
<1> JMP $+2 ; I/O DELAY
<1> mov dx, 2F9h ; interrupt enable register
<1> in al, dx ; read register
<1> JMP $+2 ; I/O DELAY
<1> ;or al, 1 ; receiver data interrupt enable and
<1> or al, 3 ; transmitter empty interrupt enable
<1> out dx, al ; write back to register
<1> JMP $+2 ; I/O DELAY
<1> in al, 21h ; read interrupt mask register
<1> JMP $+2 ; I/O DELAY
<1> and al, 0F7h ; enable IRQ 3 (COM2)
<1> out 21h, al ; write back to register
<1> ;
<1> ; 23/10/2015
<1> mov eax, com2_int
<1> mov [com2_irq3], eax
<1> ; 26/10/2015
<1> popf
<1> sp_i7:
<1> retn
<1>
<1> sp_i3:
<1> ;A4: ;----- INITIALIZE THE COMMUNICATIONS PORT
<1> ; 28/10/2015
<1> inc dl ; 3F9h (2F9h) ; 3F9h, COM1 Interrupt enable register
<1> mov al, 0
<1> out dx, al ; disable serial port interrupt
<1> JMP $+2 ; I/O DELAY
<1> add dl, 2 ; 3FBh (2FBh) ; COM1 Line control register (3FBh)
<1> mov al, 80h
<1> out dx, al ; SET DLAB=1 ; divisor latch access bit
<1> ;----- SET BAUD RATE DIVISOR
<1> ; 26/10/2015
<1> sub dl, 3 ; 3F8h (2F8h) ; register for least significant byte

```

```

13313                                     <1>                                     ; of the divisor value
13314 00003AFA 88C8                     <1>  mov  al, cl          ; 1
13315 00003AFC EE                       <1>  out  dx, al          ; 1 = 115200 baud (Retro UNIX 386 v1)
13316                                     <1>                                     ; 2 = 57600 baud
13317                                     <1>                                     ; 3 = 38400 baud
13318                                     <1>                                     ; 6 = 19200 baud
13319                                     <1>                                     ; 12 = 9600 baud (Retro UNIX 8086 v1)
13320 00003AFD EB00                     <1>  JMP   $+2          ; I/O DELAY
13321 00003AFF 28C0                     <1>  sub  al, al
13322 00003B01 FEC2                     <1>  inc  dl          ; 3F9h (2F9h) ; register for most significant byte
13323                                     <1>                                     ; of the divisor value
13324 00003B03 EE                       <1>  out  dx, al ; 0
13325 00003B04 EB00                     <1>  JMP   $+2          ; I/O DELAY
13326                                     <1>                                     ;
13327 00003B06 88E8                     <1>  mov  al, ch ; 3          ; 8 data bits, 1 stop bit, no parity
13328                                     <1>  ;and al, 1Fh ; Bits 0,1,2,3,4
13329 00003B08 80C202                   <1>  add  dl, 2 ; 3FBh (2FBh) ; Line control register
13330 00003B0B EE                       <1>  out  dx, al
13331 00003B0C EB00                     <1>  JMP   $+2          ; I/O DELAY
13332                                     <1>  ; 29/10/2015
13333 00003B0E FECA                     <1>  dec  dl          ; 3FAh (2FAh) ; FIFO Control register (16550/16750)
13334 00003B10 30C0                     <1>  xor  al, al          ; 0
13335 00003B12 EE                       <1>  out  dx, al          ; Disable FIFOs (reset to 8250 mode)
13336 00003B13 EB00                     <1>  JMP   $+2
13337                                     <1>  sp_i4:
13338                                     <1>  ;A18: ;----- COMM PORT STATUS ROUTINE
13339                                     <1>  ; 29/06/2015 (line status after modem status)
13340 00003B15 80C204                   <1>  add  dl, 4 ; 3FEh (2FEh) ; Modem status register
13341                                     <1>  sp_i4s:
13342 00003B18 EC                       <1>  in   al, dx          ; GET MODEM CONTROL STATUS
13343 00003B19 EB00                     <1>  JMP   $+2          ; I/O DELAY
13344 00003B1B 88C4                     <1>  mov  ah, al          ; PUT IN (AH) FOR RETURN
13345 00003B1D FECA                     <1>  dec  dl          ; 3FDh (2FDh) ; POINT TO LINE STATUS REGISTER
13346                                     <1>                                     ; dx = 3FDh for COM1, 2FDh for COM2
13347 00003B1F EC                       <1>  in   al, dx          ; GET LINE CONTROL STATUS
13348                                     <1>  ; AL = Line status, AH = Modem status
13349 00003B20 C3                       <1>  retn
13350                                     <1>
13351                                     <1>  sp_status:
13352                                     <1>  ; 29/06/2015
13353                                     <1>  ; 27/06/2015 (Retro UNIX 386 v1)
13354                                     <1>  ; Get serial port status
13355 00003B21 66BAFE03                 <1>  mov  dx, 3FEh          ; Modem status register (COM1)
13356 00003B25 28C6                     <1>  sub  dh, al          ; dh = 2 for COM2 (al = 1)
13357                                     <1>                                     ; dx = 2FEh for COM2
13358 00003B27 EBEF                     <1>  jmp  short sp_i4s
13359                                     <1>
13360                                     <1>  sp_setp: ; Set serial port communication parameters
13361                                     <1>  ; 07/11/2015
13362                                     <1>  ; 29/10/2015
13363                                     <1>  ; 29/06/2015
13364                                     <1>  ; Retro UNIX 386 v1 feature only !
13365                                     <1>  ;
13366                                     <1>  ; INPUT:
13367                                     <1>  ;     AL = 0 for COM1
13368                                     <1>  ;     1 for COM2
13369                                     <1>  ;     AH = Communication parameters (*)
13370                                     <1>  ; OUTPUT:
13371                                     <1>  ;     CL = Line status
13372                                     <1>  ;     CH = Modem status
13373                                     <1>  ; If cf = 1 -> Error code in [u.error]
13374                                     <1>  ;     'invalid parameter !'
13375                                     <1>  ;     or
13376                                     <1>  ;     'device not ready !' error
13377                                     <1>  ;
13378                                     <1>  ; (*) Communication parameters (except BAUD RATE):
13379                                     <1>  ;     Bit   4     3     2     1     0
13380                                     <1>  ;     -PARITY-- STOP BIT -WORD LENGTH-
13381                                     <1>  ; this one -->  00 = none    0 = 1 bit  11 = 8 bits
13382                                     <1>  ;     01 = odd    1 = 2 bits 10 = 7 bits
13383                                     <1>  ;     11 = even
13384                                     <1>  ; Baud rate setting bits: (29/06/2015)
13385                                     <1>  ;     Retro UNIX 386 v1 feature only !
13386                                     <1>  ;     Bit   7     6     5 | Baud rate
13387                                     <1>  ;     -----
13388                                     <1>  ;     value 0     0     0 | Default (Divisor = 1)
13389                                     <1>  ;           0     0     1 | 9600 (12)
13390                                     <1>  ;           0     1     0 | 19200 (6)
13391                                     <1>  ;           0     1     1 | 38400 (3)
13392                                     <1>  ;           1     0     0 | 14400 (8)
13393                                     <1>  ;           1     0     1 | 28800 (4)
13394                                     <1>  ;           1     1     0 | 57600 (2)
13395                                     <1>  ;           1     1     1 | 115200 (1)
13396                                     <1>  ;
13397                                     <1>  ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
13398                                     <1>  ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
13399                                     <1>  ;
13400                                     <1>  ; ((Modified registers: EAX, ECX, EDX, EBX))
13401                                     <1>  ;
13402 00003B29 66BAF803                 <1>  mov  dx, 3F8h
13403 00003B2D BB[12710000]             <1>  mov  ebx, com1p ; COM1 control byte offset
13404 00003B32 3C01                     <1>  cmp  al, 1

```

```

13405 00003B34 776B      <1>  ja    short sp_invp_err
13406 00003B36 7203      <1>  jnb    short sp_setp1 ; COM1 (AL = 0)
13407 00003B38 FECE      <1>  dec    dh ; 2F8h
13408 00003B3A 43        <1>  inc    ebx ; COM2 control byte offset
13409                      <1>  sp_setp1:
13410                      <1>  ; 29/10/2015
13411 00003B3B 8823      <1>  mov     [ebx], ah
13412 00003B3D 0FB6CC     <1>  movzx   ecx, ah
13413 00003B40 C0E905     <1>  shr     cl, 5 ; -> baud rate index
13414 00003B43 80E41F     <1>  and     ah, 1Fh ; communication parameters except baud rate
13415 00003B46 8A81[B03B0000] <1>  mov     al, [ecx+b_div_tbl]
13416 00003B4C 6689C1     <1>  mov     cx, ax
13417 00003B4F E896FFFFFF     <1>  call    sp_i3
13418 00003B54 6689C1     <1>  mov     cx, ax ; CL = Line status, CH = Modem status
13419 00003B57 A880      <1>  test    al, 80h
13420 00003B59 740F      <1>  jz      short sp_setp2
13421 00003B5B C603E3     <1>  mov     byte [ebx], 0E3h ; Reset to initial value (11100011b)
13422                      <1>  stp_dnr_err:
13423 00003B5E C705[CF740000]0F00- <1>  mov     dword [u.error], ERR_DEV_NOT_RDY ; 'device not ready !'
13424 00003B66 0000      <1>
13425                      <1>  ; CL = Line status, CH = Modem status
13426 00003B68 F9        <1>  stc
13427 00003B69 C3        <1>  retn
13428                      <1>  sp_setp2:
13429 00003B6A 80FE02     <1>  cmp     dh, 2 ; COM2 (2F?h)
13430 00003B6D 0F8649FFFFFF     <1>  jna     sp_i6
13431                      <1>  ; COM1 (3F?h)
13432                      <1>  sp_i5:
13433                      <1>  ; 07/11/2015
13434                      <1>  ; 26/10/2015
13435                      <1>  ; 29/06/2015
13436                      <1>  ;
13437                      <1>  ;; COM1 - enabling IRQ 4
13438 00003B73 9C        <1>  pushf
13439 00003B74 FA        <1>  cli
13440 00003B75 66BAFC03    <1>  mov     dx, 3FCh ; modem control register
13441 00003B79 EC        <1>  in      al, dx ; read register
13442 00003B7A EB00      <1>  JMP     $+2 ; I/O DELAY
13443 00003B7C 0C08      <1>  or      al, 8 ; enable bit 3 (OUT2)
13444 00003B7E EE        <1>  out     dx, al ; write back to register
13445 00003B7F EB00      <1>  JMP     $+2 ; I/O DELAY
13446 00003B81 66BAF903    <1>  mov     dx, 3F9h ; interrupt enable register
13447 00003B85 EC        <1>  in      al, dx ; read register
13448 00003B86 EB00      <1>  JMP     $+2 ; I/O DELAY
13449                      <1>  ;or al, 1 ; receiver data interrupt enable and
13450 00003B88 0C03      <1>  or      al, 3 ; transmitter empty interrupt enable
13451 00003B8A EE        <1>  out     dx, al ; write back to register
13452 00003B8B EB00      <1>  JMP     $+2 ; I/O DELAY
13453 00003B8D E421      <1>  in      al, 21h ; read interrupt mask register
13454 00003B8F EB00      <1>  JMP     $+2 ; I/O DELAY
13455 00003B91 24EF      <1>  and     al, 0EFh ; enable IRQ 4 (COM1)
13456 00003B93 E621      <1>  out     21h, al ; write back to register
13457                      <1>  ;
13458                      <1>  ; 23/10/2015
13459 00003B95 B8[A5390000] <1>  mov     eax, com1_int
13460 00003B9A A3[773F0000] <1>  mov     [com1_irq4], eax
13461                      <1>  ; 26/10/2015
13462 00003B9F 9D        <1>  popf
13463 00003BA0 C3        <1>  retn
13464                      <1>
13465                      <1>  sp_invp_err:
13466 00003BA1 C705[CF740000]1700- <1>  mov     dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
13467 00003BA9 0000      <1>
13468 00003BAB 31C9      <1>  xor     ecx, ecx
13469 00003BAD 49        <1>  dec     ecx ; 0FFFFh
13470 00003BAE F9        <1>  stc
13471 00003BAF C3        <1>  retn
13472                      <1>
13473                      <1>  ; 29/10/2015
13474                      <1>  b_div_tbl: ; Baud rate divisor table (115200/divisor)
13475 00003BB0 010C0603080401 <1>  db 1, 12, 6, 3, 8, 4, 1
13476                      <1>
13477                      <1>  ; Retro UNIX 8086 v1 - UNIX.ASM (01/09/2014)
13478                      <1>  epoch:
13479                      <1>  ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
13480                      <1>  ; 09/04/2013 (Retro UNIX 8086 v1 - UNIX.ASM)
13481                      <1>  ; 'epoch' procedure prototype:
13482                      <1>  ; UNIXCOPY.ASM, 10/03/2013
13483                      <1>  ; 14/11/2012
13484                      <1>  ; unixboot.asm (boot file configuration)
13485                      <1>  ; version of "epoch" procedure in "unixproc.asm"
13486                      <1>  ; 21/7/2012
13487                      <1>  ; 15/7/2012
13488                      <1>  ; 14/7/2012
13489                      <1>  ; Erdogan Tan - RETRO UNIX v0.1
13490                      <1>  ; compute current date and time as UNIX Epoch/Time
13491                      <1>  ; UNIX Epoch: seconds since 1/1/1970 00:00:00
13492                      <1>  ;
13493                      <1>  ; ((Modified registers: EAX, EDX, ECX, EBX))
13494                      <1>  ;
13495 00003BB7 E81D010000    <1>  call    get_rtc_time ; Return Current Time
13496 00003BBC 86E9      <1>  xchg    ch, cl

```

```

13497 00003BBE 66890D[B26D0000] <1>      mov  [hour], cx
13498 00003BC5 86F2             <1>      xchg dh,dl
13499 00003BC7 668915[B66D0000] <1>      mov  [second], dx
13500                                <1>      ;
13501 00003BCE E837010000         <1>      call get_rtc_date          ; Return Current Date
13502 00003BD3 86E9             <1>      xchg ch,cl
13503 00003BD5 66890D[AC6D0000] <1>      mov  [year], cx
13504 00003BDC 86F2             <1>      xchg dh,dl
13505 00003BDE 668915[AE6D0000] <1>      mov  [month], dx
13506                                <1>      ;
13507 00003BE5 66B93030         <1>      mov  cx, 3030h
13508                                <1>      ;
13509 00003BE9 A0[B26D0000]     <1>      mov  al, [hour] ; Hour
13510                                <1>      ; AL <= BCD number)
13511 00003BEE D410             <1>      db    0D4h,10h          ; Undocumented inst. AAM
13512                                <1>      ; AH = AL / 10h
13513                                <1>      ; AL = AL MOD 10h
13514 00003BF0 D50A             <1>      aad  ; AX= AH*10+AL
13515 00003BF2 A2[B26D0000]     <1>      mov  [hour], al
13516 00003BF7 A0[B36D0000]     <1>      mov  al, [hour+1] ; Minute
13517                                <1>      ; AL <= BCD number)
13518 00003BFC D410             <1>      db    0D4h,10h          ; Undocumented inst. AAM
13519                                <1>      ; AH = AL / 10h
13520                                <1>      ; AL = AL MOD 10h
13521 00003BFE D50A             <1>      aad  ; AX= AH*10+AL
13522 00003C00 A2[B46D0000]     <1>      mov  [minute], al
13523 00003C05 A0[B66D0000]     <1>      mov  al, [second] ; Second
13524                                <1>      ; AL <= BCD number)
13525 00003C0A D410             <1>      db    0D4h,10h          ; Undocumented inst. AAM
13526                                <1>      ; AH = AL / 10h
13527                                <1>      ; AL = AL MOD 10h
13528 00003C0C D50A             <1>      aad  ; AX= AH*10+AL
13529 00003C0E A2[B66D0000]     <1>      mov  [second], al
13530 00003C13 66A1[AC6D0000]   <1>      mov  ax, [year] ; Year (century)
13531 00003C19 6650             <1>      push ax
13532                                <1>      ; AL <= BCD number)
13533 00003C1B D410             <1>      db    0D4h,10h          ; Undocumented inst. AAM
13534                                <1>      ; AH = AL / 10h
13535                                <1>      ; AL = AL MOD 10h
13536 00003C1D D50A             <1>      aad  ; AX= AH*10+AL
13537 00003C1F B464             <1>      mov  ah, 100
13538 00003C21 F6E4             <1>      mul  ah
13539 00003C23 66A3[AC6D0000]   <1>      mov  [year], ax
13540 00003C29 6658             <1>      pop  ax
13541 00003C2B 88E0             <1>      mov  al, ah
13542                                <1>      ; AL <= BCD number)
13543 00003C2D D410             <1>      db    0D4h,10h          ; Undocumented inst. AAM
13544                                <1>      ; AH = AL / 10h
13545                                <1>      ; AL = AL MOD 10h
13546 00003C2F D50A             <1>      aad  ; AX= AH*10+AL
13547 00003C31 660105[AC6D0000] <1>      add  [year], ax
13548 00003C38 A0[AE6D0000]     <1>      mov  al, [month] ; Month
13549                                <1>      ; AL <= BCD number)
13550 00003C3D D410             <1>      db    0D4h,10h          ; Undocumented inst. AAM
13551                                <1>      ; AH = AL / 10h
13552                                <1>      ; AL = AL MOD 10h
13553 00003C3F D50A             <1>      aad  ; AX= AH*10+AL
13554 00003C41 A2[AE6D0000]     <1>      mov  [month], al
13555 00003C46 A0[AF6D0000]     <1>      mov  al, [month+1] ; Day
13556                                <1>      ; AL <= BCD number)
13557 00003C4B D410             <1>      db    0D4h,10h          ; Undocumented inst. AAM
13558                                <1>      ; AH = AL / 10h
13559                                <1>      ; AL = AL MOD 10h
13560 00003C4D D50A             <1>      aad  ; AX= AH*10+AL
13561 00003C4F A2[B06D0000]     <1>      mov  [day], al
13562                                <1>
13563                                <1>      convert_to_epoch:
13564                                <1>      ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit modification)
13565                                <1>      ; 09/04/2013 (retro UNIX 8086 v1)
13566                                <1>      ;
13567                                <1>      ; ((Modified registers: EAX, EDX, EBX))
13568                                <1>      ;
13569                                <1>      ; Derived from DALLAS Semiconductor
13570                                <1>      ; Application Note 31 (DS1602/DS1603)
13571                                <1>      ; 6 May 1998
13572 00003C54 29C0             <1>      sub  eax, eax
13573 00003C56 66A1[AC6D0000]   <1>      mov  ax, [year]
13574 00003C5C 662DB207         <1>      sub  ax, 1970
13575 00003C60 BA6D010000       <1>      mov  edx, 365
13576 00003C65 F7E2             <1>      mul  edx
13577 00003C67 31DB             <1>      xor  ebx, ebx
13578 00003C69 8A1D[AE6D0000]   <1>      mov  bl, [month]
13579 00003C6F FECB             <1>      dec  bl
13580 00003C71 D0E3             <1>      shl  bl, 1
13581                                <1>      ;sub  edx, edx
13582 00003C73 668B93[B86D0000]   <1>      mov  dx, [EBX+DMonth]
13583 00003C7A 8A1D[B06D0000]   <1>      mov  bl, [day]
13584 00003C80 FECB             <1>      dec  bl
13585 00003C82 01D0             <1>      add  eax, edx
13586 00003C84 01D8             <1>      add  eax, ebx
13587                                <1>      ; EAX = days since 1/1/1970
13588 00003C86 668B15[AC6D0000] <1>      mov  dx, [year]

```

```

13589 00003C8D 6681EAB107    <1>  sub    dx, 1969
13590 00003C92 66D1EA        <1>  shr    dx, 1
13591 00003C95 66D1EA        <1>  shr    dx, 1
13592                                <1>      ; (year-1969)/4
13593 00003C98 01D0          <1>  add    eax, edx
13594                                <1>      ; + leap days since 1/1/1970
13595 00003C9A 803D[AE6D0000]02 <1>  cmp    byte [month], 2    ; if past february
13596 00003CA1 7610          <1>  jna    short ctel
13597 00003CA3 668B15[AC6D0000] <1>  mov    dx, [year]
13598 00003CAA 6683E203      <1>  and    dx, 3 ; year mod 4
13599 00003CAE 7503          <1>  jnz    short ctel
13600                                <1>      ; and if leap year
13601 00003CB0 83C001        <1>  add    eax, 1    ; add this year's leap day (february 29)
13602                                <1>  ctel:      ; compute seconds since 1/1/1970
13603 00003CB3 BA18000000    <1>  mov    edx, 24
13604 00003CB8 F7E2          <1>  mul    edx
13605 00003CBA 8A15[B26D0000] <1>  mov    dl, [hour]
13606 00003CC0 01D0          <1>  add    eax, edx
13607                                <1>      ; EAX = hours since 1/1/1970 00:00:00
13608                                <1>  ;mov    ebx, 60
13609 00003CC2 B33C          <1>  mov    bl, 60
13610 00003CC4 F7E3          <1>  mul    ebx
13611 00003CC6 8A15[B46D0000] <1>  mov    dl, [minute]
13612 00003CCC 01D0          <1>  add    eax, edx
13613                                <1>      ; EAX = minutes since 1/1/1970 00:00:00
13614                                <1>  ;mov    ebx, 60
13615 00003CCE F7E3          <1>  mul    ebx
13616 00003CD0 8A15[B66D0000] <1>  mov    dl, [second]
13617 00003CD6 01D0          <1>  add    eax, edx
13618                                <1>      ; EAX -> seconds since 1/1/1970 00:00:00
13619 00003CD8 C3            <1>  retn
13620                                <1>
13621                                <1> get_rtc_time:
13622                                <1>      ; 15/03/2015
13623                                <1>      ; Derived from IBM PC-XT Model 286 BIOS Source Code
13624                                <1>      ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
13625                                <1>      ; INT 1Ah
13626                                <1>      ; (AH) = 02H  READ THE REAL TIME CLOCK AND RETURN WITH,      :
13627                                <1>      ;      (CH) = HOURS IN BCD (00-23)                :
13628                                <1>      ;      (CL) = MINUTES IN BCD (00-59)            :
13629                                <1>      ;      (DH) = SECONDS IN BCD (00-59)            :
13630                                <1>      ;      (DL) = DAYLIGHT SAVINGS ENABLE (00-01).    :
13631                                <1>      ;
13632                                <1>  RTC_20:      ; GET RTC TIME
13633 00003CD9 FA              <1>  cli
13634 00003CDA E870CFFFFFFF    <1>  CALL    UPD_IPR      ; CHECK FOR UPDATE IN PROCESS
13635 00003CDF 7227            <1>  JC      short RTC_29    ; EXIT IF ERROR (CY= 1)
13636                                <1>
13637 00003CE1 B000            <1>  MOV     AL,CMOS_SECONDS ; SET ADDRESS OF SECONDS
13638 00003CE3 E84FCFFFFFFF    <1>  CALL    CMOS_READ    ; GET SECONDS
13639 00003CE8 88C6            <1>  MOV     DH,AL         ; SAVE
13640 00003CEA B00B            <1>  MOV     AL,CMOS_REG_B ; ADDRESS ALARM REGISTER
13641 00003CEC E846CFFFFFFF    <1>  CALL    CMOS_READ    ; READ CURRENT VALUE OF DSE BIT
13642 00003CF1 2401            <1>  AND     AL,00000001B  ; MASK FOR VALID DSE BIT
13643 00003CF3 88C2            <1>  MOV     DL,AL         ; SET [DL] TO ZERO FOR NO DSE BIT
13644 00003CF5 B002            <1>  MOV     AL,CMOS_MINUTES ; SET ADDRESS OF MINUTES
13645 00003CF7 E83BCFFFFFFF    <1>  CALL    CMOS_READ    ; GET MINUTES
13646 00003CFC 88C1            <1>  MOV     CL,AL         ; SAVE
13647 00003CFE B004            <1>  MOV     AL,CMOS_HOURS ; SET ADDRESS OF HOURS
13648 00003D00 E832CFFFFFFF    <1>  CALL    CMOS_READ    ; GET HOURS
13649 00003D05 88C5            <1>  MOV     CH,AL         ; SAVE
13650 00003D07 F8              <1>  CLC              ; SET CY= 0
13651                                <1>  RTC_29:
13652 00003D08 FB              <1>  sti
13653 00003D09 C3              <1>  RETn      ; RETURN WITH RESULT IN CARRY FLAG
13654                                <1>
13655                                <1> get_rtc_date:
13656                                <1>      ; 15/03/2015
13657                                <1>      ; Derived from IBM PC-XT Model 286 BIOS Source Code
13658                                <1>      ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
13659                                <1>      ; INT 1Ah
13660                                <1>      ; (AH) = 04H  READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH,:
13661                                <1>      ;      (CH) = CENTURY IN BCD (19 OR 20)                :
13662                                <1>      ;      (CL) = YEAR IN BCD (00-99)                    :
13663                                <1>      ;      (DH) = MONTH IN BCD (01-12)                    :
13664                                <1>      ;      (DL) = DAY IN BCD (01-31).                      :
13665                                <1>      ;
13666                                <1>  RTC_40:      ; GET RTC DATE
13667 00003D0A FA              <1>  cli
13668 00003D0B E83FCFFFFFFF    <1>  CALL    UPD_IPR      ; CHECK FOR UPDATE IN PROCESS
13669 00003D10 7225            <1>  JC      short RTC_49    ; EXIT IF ERROR (CY= 1)
13670                                <1>
13671 00003D12 B007            <1>  MOV     AL,CMOS_DAY_MONTH ; ADDRESS DAY OF MONTH
13672 00003D14 E81ECFFFFFFF    <1>  CALL    CMOS_READ    ; READ DAY OF MONTH
13673 00003D19 88C2            <1>  MOV     DL,AL         ; SAVE
13674 00003D1B B008            <1>  MOV     AL,CMOS_MONTH ; ADDRESS MONTH
13675 00003D1D E815CFFFFFFF    <1>  CALL    CMOS_READ    ; READ MONTH
13676 00003D22 88C6            <1>  MOV     DH,AL         ; SAVE
13677 00003D24 B009            <1>  MOV     AL,CMOS_YEAR  ; ADDRESS YEAR
13678 00003D26 E80CCFFFFFFF    <1>  CALL    CMOS_READ    ; READ YEAR
13679 00003D2B 88C1            <1>  MOV     CL,AL         ; SAVE
13680 00003D2D B032            <1>  MOV     AL,CMOS_CENTURY ; ADDRESS CENTURY LOCATION

```

13681	00003D2F	E803CFFFFFF	<1>	CALL	CMOS_READ	; GET CENTURY BYTE
13682	00003D34	88C5	<1>	MOV	CH,AL	; SAVE
13683	00003D36	F8	<1>	CLC		; SET CY=0
13684			<1>	RTC_49:		
13685	00003D37	FB	<1>	sti		
13686	00003D38	C3	<1>	RETN		; RETURN WITH RESULTS IN CARRY FLAG
13687			<1>			
13688			<1>	set_date_time:		
13689			<1>	convert_from_epoch:		
13690			<1>		; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)	
13691			<1>		; 20/06/2013 (Retro UNIX 8086 v1)	
13692			<1>		; 'convert_from_epoch' procedure prototype:	
13693			<1>		; UNIXCOPY.ASM, 10/03/2013	
13694			<1>			
13695			<1>		; ((Modified registers: EAX, EDX, ECX, EBX))	
13696			<1>			
13697			<1>		; Derived from DALLAS Semiconductor	
13698			<1>		; Application Note 31 (DS1602/DS1603)	
13699			<1>		; 6 May 1998	
13700			<1>			
13701			<1>		; INPUT:	
13702			<1>		; EAX = Unix (Epoch) Time	
13703			<1>			
13704	00003D39	31D2	<1>	xor	edx, edx	
13705	00003D3B	B93C000000	<1>	mov	ecx, 60	
13706	00003D40	F7F1	<1>	div	ecx	
13707			<1>	;mov	[imin], eax ; whole minutes	
13708			<1>		; since 1/1/1970	
13709	00003D42	668915[B66D0000]	<1>	mov	[second], dx ; leftover seconds	
13710	00003D49	29D2	<1>	sub	edx, edx	
13711	00003D4B	F7F1	<1>	div	ecx	
13712			<1>	;mov	[ihrs], eax ; whole hours	
13713			<1>		; since 1/1/1970	
13714	00003D4D	668915[B46D0000]	<1>	mov	[minute], dx ; leftover minutes	
13715	00003D54	31D2	<1>	xor	edx, edx	
13716			<1>	;mov	cx, 24	
13717	00003D56	B118	<1>	mov	cl, 24	
13718	00003D58	F7F1	<1>	div	ecx	
13719			<1>	;mov	[iday], ax ; whole days	
13720			<1>		; since 1/1/1970	
13721	00003D5A	668915[B26D0000]	<1>	mov	[hour], dx ; leftover hours	
13722	00003D61	05DB020000	<1>	add	eax, 365+366 ; whole day since	
13723			<1>		; 1/1/1968	
13724			<1>	;mov	[iday], ax	
13725	00003D66	50	<1>	push	eax	
13726	00003D67	29D2	<1>	sub	edx, edx	
13727	00003D69	B9B5050000	<1>	mov	ecx, (4*365)+1 ; 4 years = 1461 days	
13728	00003D6E	F7F1	<1>	div	ecx	
13729	00003D70	59	<1>	pop	ecx	
13730			<1>	;mov	[lday], ax ; count of quadyrs (4 years)	
13731	00003D71	6652	<1>	push	dx	
13732			<1>	;mov	[qday], dx ; days since quadyr began	
13733	00003D73	6683FA3C	<1>	cmp	dx, 31 + 29 ; if past feb 29 then	
13734	00003D77	F5	<1>	cmc		; add this quadyr's leap day
13735	00003D78	83D000	<1>	adc	eax, 0 ; to # of qadyrs (leap days)	
13736			<1>	;mov	[lday], ax ; since 1968	
13737			<1>	;mov	cx, [iday]	
13738	00003D7B	91	<1>	xchg	ecx, eax ; ECX = lday, EAX = iday	
13739	00003D7C	29C8	<1>	sub	eax, ecx ; iday - lday	
13740	00003D7E	B96D010000	<1>	mov	ecx, 365	
13741	00003D83	31D2	<1>	xor	edx, edx	
13742			<1>		; EAX = iday-lday, EDX = 0	
13743	00003D85	F7F1	<1>	div	ecx	
13744			<1>	;mov	[iyrs], ax ; whole years since 1968	
13745			<1>	;jday = iday - (iyrs*365) - lday		
13746			<1>	;mov	[jday], dx ; days since 1/1 of current year	
13747			<1>	;add	eax, 1968	
13748	00003D87	6605B007	<1>	add	ax, 1968 ; compute year	
13749	00003D8B	66A3[AC6D0000]	<1>	mov	[year], ax	
13750	00003D91	6689D1	<1>	mov	cx, dx	
13751			<1>	;mov	dx, [qday]	
13752	00003D94	665A	<1>	pop	dx	
13753	00003D96	6681FA6D01	<1>	cmp	dx, 365 ; if qday <= 365 and qday >= 60	
13754	00003D9B	7709	<1>	ja	short cfe1 ; jday = jday +1	
13755	00003D9D	6683FA3C	<1>	cmp	dx, 60 ; if past 2/29 and leap year then	
13756	00003DA1	F5	<1>	cmc		; add a leap day to the # of whole
13757	00003DA2	6683D100	<1>	adc	cx, 0 ; days since 1/1 of current year	
13758			<1>	cfe1:		
13759			<1>	;mov	[jday], cx	
13760	00003DA6	66BB0C00	<1>	mov	bx, 12 ; estimate month	
13761	00003DAA	66BA6E01	<1>	mov	dx, 366 ; mday, max. days since 1/1 is 365	
13762	00003DAE	6683E003	<1>	and	ax, 11b ; year mod 4 (and dx, 3)	
13763			<1>	cfe2:	; Month calculation ; 0 to 11 (11 to 0)	
13764	00003DB2	6639D1	<1>	cmp	cx, dx ; mday = # of days passed from 1/1	
13765	00003DB5	731D	<1>	jnb	short cfe3	
13766	00003DB7	664B	<1>	dec	bx ; month = month - 1	
13767	00003DB9	66D1E3	<1>	shl	bx, 1	
13768	00003DBC	668B93[B86D0000]	<1>	mov	dx, [EBX+DMonth] ; # elapsed days at 1st of month	
13769	00003DC3	66D1EB	<1>	shr	bx, 1 ; bx = month - 1 (0 to 11)	
13770	00003DC6	6683FB01	<1>	cmp	bx, 1 ; if month > 2 and year mod 4 = 0	
13771	00003DCA	76E6	<1>	jna	short cfe2 ; then mday = mday + 1	
13772	00003DCC	08C0	<1>	or	al, al ; if past 2/29 and leap year then	

13773	00003DCE	75E2	<1>	jnz	short cfe2	; add leap day (to mday)
13774	00003DD0	6642	<1>	inc	dx	; mday = mday + 1
13775	00003DD2	EBDE	<1>	jmp	short cfe2	
13776			<1>	cfe3:		
13777	00003DD4	6643	<1>	inc	bx	; -> bx = month, 1 to 12
13778	00003DD6	66891D[AE6D0000]	<1>	mov	[month], bx	
13779	00003DDD	6629D1	<1>	sub	cx, dx	; day = jday - mday + 1
13780	00003DE0	6641	<1>	inc	cx	
13781	00003DE2	66890D[B06D0000]	<1>	mov	[day], cx	
13782			<1>			
13783			<1>		; eax, ebx, ecx, edx is changed at return	
13784			<1>		; output ->	
13785			<1>		; [year], [month], [day], [hour], [minute], [second]	
13786			<1>			
13787			<1>		; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)	
13788			<1>		; 20/06/2013 (Retro UNIX 8086 v1)	
13789			<1>	set_date:		
13790	00003DE9	A0[AD6D0000]	<1>	mov	al, [year+1]	
13791	00003DEE	D40A	<1>	aam	; ah = al / 10, al = al mod 10	
13792	00003DF0	D510	<1>	db	0D5h,10h	; Undocumented inst. AAD
13793			<1>		; AL = AH * 10h + AL	
13794	00003DF2	88C5	<1>	mov	ch, al ; century (BCD)	
13795	00003DF4	A0[AC6D0000]	<1>	mov	al, [year]	
13796	00003DF9	D40A	<1>	aam	; ah = al / 10, al = al mod 10	
13797	00003DFB	D510	<1>	db	0D5h,10h	; Undocumented inst. AAD
13798			<1>		; AL = AH * 10h + AL	
13799	00003DFD	88C1	<1>	mov	cl, al ; year (BCD)	
13800	00003DFF	A0[AE6D0000]	<1>	mov	al, [month]	
13801	00003E04	D40A	<1>	aam	; ah = al / 10, al = al mod 10	
13802	00003E06	D510	<1>	db	0D5h,10h	; Undocumented inst. AAD
13803			<1>		; AL = AH * 10h + AL	
13804	00003E08	88C6	<1>	mov	dh, al ; month (BCD)	
13805	00003E0A	A0[B06D0000]	<1>	mov	al, [day]	
13806	00003E0F	D40A	<1>	aam	; ah = al / 10, al = al mod 10	
13807	00003E11	D510	<1>	db	0D5h,10h	; Undocumented inst. AAD
13808			<1>		; AL = AH * 10h + AL	
13809	00003E13	88C6	<1>	mov	dh, al ; day (BCD)	
13810			<1>		; Set real-time clock date	
13811	00003E15	E879000000	<1>	call	set_rtc_date	
13812			<1>	set_time:		
13813			<1>		; Read real-time clock time	
13814			<1>		; (get day light saving time bit status)	
13815	00003E1A	FA	<1>	cli		
13816	00003E1B	E82FCEFFFF	<1>	CALL	UPD_IPR	; CHECK FOR UPDATE IN PROCESS
13817			<1>		; cf = 1 -> al = 0	
13818	00003E20	7207	<1>	jc	short stime1	
13819	00003E22	B00B	<1>	MOV	AL,CMOS_REG_B	; ADDRESS ALARM REGISTER
13820	00003E24	E80ECEFFFF	<1>	CALL	CMOS_READ	; READ CURRENT VALUE OF DSE BIT
13821			<1>	stime1:		
13822	00003E29	FB	<1>	sti		
13823	00003E2A	2401	<1>	AND	AL,00000001B	; MASK FOR VALID DSE BIT
13824	00003E2C	88C2	<1>	MOV	DL,AL	; SET [DL] TO ZERO FOR NO DSE BIT
13825			<1>		; DL = 1 or 0 (day light saving time)	
13826			<1>			
13827	00003E2E	A0[B26D0000]	<1>	mov	al, [hour]	
13828	00003E33	D40A	<1>	aam	; ah = al / 10, al = al mod 10	
13829	00003E35	D510	<1>	db	0D5h,10h	; Undocumented inst. AAD
13830			<1>		; AL = AH * 10h + AL	
13831	00003E37	88C5	<1>	mov	ch, al ; hour (BCD)	
13832	00003E39	A0[B46D0000]	<1>	mov	al, [minute]	
13833	00003E3E	D40A	<1>	aam	; ah = al / 10, al = al mod 10	
13834	00003E40	D510	<1>	db	0D5h,10h	; Undocumented inst. AAD
13835			<1>		; AL = AH * 10h + AL	
13836	00003E42	88C1	<1>	mov	cl, al ; minute (BCD)	
13837	00003E44	A0[B66D0000]	<1>	mov	al, [second]	
13838	00003E49	D40A	<1>	aam	; ah = al / 10, al = al mod 10	
13839	00003E4B	D510	<1>	db	0D5h,10h	; Undocumented inst. AAD
13840			<1>		; AL = AH * 10h + AL	
13841	00003E4D	88C6	<1>	mov	dh, al ; second (BCD)	
13842			<1>		; Set real-time clock time	
13843			<1>		; call set_rtc_time	
13844			<1>	set_rtc_time:		
13845			<1>		; 15/04/2015 (257, POSTEQU.INC -> H EQU 256, X EQU H+1)	
13846			<1>		; 15/03/2015	
13847			<1>		; Derived from IBM PC-XT Model 286 BIOS Source Code	
13848			<1>		; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES	
13849			<1>		; INT 1Ah	
13850			<1>		; (AH) = 03H SET THE REAL TIME CLOCK USING,	:
13851			<1>		; (CH) = HOURS IN BCD (00-23)	:
13852			<1>		; (CL) = MINUTES IN BCD (00-59)	:
13853			<1>		; (DH) = SECONDS IN BCD (00-59)	:
13854			<1>		; (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00.	:
13855			<1>			:
13856			<1>		; NOTE: (DL)= 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED.	:
13857			<1>		; (DL)= 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN	:
13858			<1>		; APRIL (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN	:
13859			<1>		; OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME.	:
13860			<1>			:
13861			<1>	RTC_30:		; SET RTC TIME
13862	00003E4F	FA	<1>	cli		
13863	00003E50	E8FACDFFFF	<1>	CALL	UPD_IPR	; CHECK FOR UPDATE IN PROCESS
13864	00003E55	7305	<1>	JNC	short RTC_35	; GO AROUND IF CLOCK OPERATING

```

13865 00003E57 E886000000    <1> CALL RTC_STA                ; ELSE TRY INITIALIZING CLOCK
13866                                <1> RTC_35:
13867 00003E5C 88F4          <1> MOV AH,DH                    ; GET TIME BYTE - SECONDS
13868 00003E5E B000          <1> MOV AL,CMOS_SECONDS          ; ADDRESS SECONDS
13869 00003E60 E89E000000    <1> CALL CMOS_WRITE              ; UPDATE SECONDS
13870 00003E65 88CC          <1> MOV AH,CL                    ; GET TIME BYTE - MINUTES
13871 00003E67 B002          <1> MOV AL,CMOS_MINUTES          ; ADDRESS MINUTES
13872 00003E69 E895000000    <1> CALL CMOS_WRITE              ; UPDATE MINUTES
13873 00003E6E 88EC          <1> MOV AH,CH                    ; GET TIME BYTE - HOURS
13874 00003E70 B004          <1> MOV AL,CMOS_HOURS            ; ADDRESS HOURS
13875 00003E72 E88C000000    <1> CALL CMOS_WRITE              ; UPDATE ADDRESS
13876                                <1> ;MOV AX,X*CMOS_REG_B        ; ADDRESS ALARM REGISTER
13877 00003E77 66B80B0B      <1> MOV AX,257*CMOS_REG_B        ;
13878 00003E7B E8B7CDFFFF    <1> CALL CMOS_READ              ; READ CURRENT TIME
13879 00003E80 2462          <1> AND AL,01100010B            ; MASK FOR VALID BIT POSITIONS
13880 00003E82 0C02          <1> OR AL,00000010B             ; TURN ON 24 HOUR MODE
13881 00003E84 80E201        <1> AND DL,00000001B           ; USE ONLY THE DSE BIT
13882 00003E87 08D0          <1> OR AL,DL                    ; GET DAY LIGHT SAVINGS TIME BIT (OSE)
13883 00003E89 86E0          <1> XCHG AH,AL                  ; PLACE IN WORK REGISTER AND GET ADDRESS
13884 00003E8B E873000000    <1> CALL CMOS_WRITE              ; SET NEW ALARM BITS
13885 00003E90 F8            <1> CLC                          ; SET CY= 0
13886 00003E91 FB            <1> sti
13887 00003E92 C3            <1> RETn                          ; RETURN WITH CY= 0
13888                                <1>
13889                                <1> set_rtc_date:
13890                                <1> ; 15/04/2015 (257, POSTEQU.INC -> H EQU 256, X EQU H+1)
13891                                <1> ; 15/03/2015
13892                                <1> ; Derived from IBM PC-XT Model 286 BIOS Source Code
13893                                <1> ; BIOS2.ASM ----- 10/06/1985 BIOS INTERRUPT ROUTINES
13894                                <1> ; INT 1Ah
13895                                <1> ; (AH) = 05H SET THE DATE INTO THE REAL TIME CLOCK USING, :
13896                                <1> ; (CH) = CENTURY IN BCD (19 OR 20) :
13897                                <1> ; (CL) = YEAR IN BCD (00-99) :
13898                                <1> ; (DH) = MONTH IN BCD (01-12) :
13899                                <1> ; (DL) = DAY IN BCD (01-31).
13900                                <1> ;
13901                                <1> RTC_50: ; SET RTC DATE
13902 00003E93 FA            <1> cli
13903 00003E94 E8B6CDFFFF    <1> CALL UPD_IPR                ; CHECK FOR UPDATE IN PROCESS
13904 00003E99 7305          <1> JNC short RTC_55            ; GO AROUND IF NO ERROR
13905 00003E9B E842000000    <1> CALL RTC_STA                ; ELSE INITIALIZE CLOCK
13906                                <1> RTC_55:
13907 00003EA0 66B80600      <1> MOV AX,CMOS_DAY_WEEK        ; ADDRESS OF DAY OF WEEK BYTE
13908 00003EA4 E85A000000    <1> CALL CMOS_WRITE              ; LOAD ZEROS TO DAY OF WEEK
13909 00003EA9 88D4          <1> MOV AH,DL                    ; GET DAY OF MONTH BYTE
13910 00003EAB B007          <1> MOV AL,CMOS_DAY_MONTH        ; ADDRESS DAY OF MONTH BYTE
13911 00003EAD E851000000    <1> CALL CMOS_WRITE              ; WRITE OF DAY OF MONTH REGISTER
13912 00003EB2 88F4          <1> MOV AH,DH                    ; GET MONTH
13913 00003EB4 B008          <1> MOV AL,CMOS_MONTH            ; ADDRESS MONTH BYTE
13914 00003EB6 E848000000    <1> CALL CMOS_WRITE              ; WRITE MONTH REGISTER
13915 00003EBB 88CC          <1> MOV AH,CL                    ; GET YEAR BYTE
13916 00003EBD B009          <1> MOV AL,CMOS_YEAR            ; ADDRESS YEAR REGISTER
13917 00003EBF E83F000000    <1> CALL CMOS_WRITE              ; WRITE YEAR REGISTER
13918 00003EC4 88EC          <1> MOV AH,CH                    ; GET CENTURY BYTE
13919 00003EC6 B032          <1> MOV AL,CMOS_CENTURY          ; ADDRESS CENTURY BYTE
13920 00003EC8 E836000000    <1> CALL CMOS_WRITE              ; WRITE CENTURY LOCATION
13921                                <1> ;MOV AX,X*CMOS_REG_B        ; ADDRESS ALARM REGISTER
13922 00003ECD 66B80B0B      <1> MOV AX,257*CMOS_REG_B        ;
13923 00003ED1 E861CDFFFF    <1> CALL CMOS_READ              ; READ CURRENT SETTINGS
13924 00003ED6 247F          <1> AND AL,07FH                 ; CLEAR 'SET BIT'
13925 00003ED8 86E0          <1> XCHG AH,AL                  ; MOVE TO WORK REGISTER
13926 00003EDA E824000000    <1> CALL CMOS_WRITE              ; AND START CLOCK UPDATING
13927 00003EDF F8            <1> CLC                          ; SET CY= 0
13928 00003EE0 FB            <1> sti
13929 00003EE1 C3            <1> RETn                          ; RETURN CY=0
13930                                <1>
13931                                <1> ; 15/03/2015
13932                                <1> RTC_STA: ; INITIALIZE REAL TIME CLOCK
13933 00003EE2 B426          <1> mov ah, 26h
13934 00003EE4 B00A          <1> mov al, CMOS_REG_A          ; ADDRESS REGISTER A AND LOAD DATA MASK
13935 00003EE6 E818000000    <1> CALL CMOS_WRITE              ; INITIALIZE STATUS REGISTER A
13936 00003EEB B482          <1> mov ah, 82h
13937 00003EED B00B          <1> mov al, CMOS_REG_B          ; SET "SET BIT" FOR CLOCK INITIALIZATION
13938 00003EEF E80F000000    <1> CALL CMOS_WRITE              ; AND 24 HOUR MODE TO REGISTER B
13939 00003EF4 B00C          <1> MOV AL,CMOS_REG_C            ; ADDRESS REGISTER C
13940 00003EF6 E83CCDFFFF    <1> CALL CMOS_READ              ; READ REGISTER C TO INITIALIZE
13941 00003EFB B00D          <1> MOV AL,CMOS_REG_D            ; ADDRESS REGISTER D
13942 00003EFD E835CDFFFF    <1> CALL CMOS_READ              ; READ REGISTER D TO INITIALIZE
13943 00003F02 C3            <1> RETn
13944                                <1>
13945                                <1> ; 15/03/2015
13946                                <1> ; IBM PC/XT Model 286 BIOS source code ----- 10/06/85 (test4.asm)
13947                                <1> CMOS_WRITE: ; WRITE (AH) TO LOCATION (AL)
13948 00003F03 9C            <1> pushf ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
13949                                <1> ;push ax ; SAVE WORK REGISTER VALUES
13950 00003F04 D0C0          <1> rol al, 1 ; MOVE NMI BIT TO LOW POSITION
13951 00003F06 F9            <1> stc ; FORCE NMI BIT ON IN CARRY FLAG
13952 00003F07 D0D8          <1> rcr al, 1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
13953 00003F09 FA            <1> cli ; DISABLE INTERRUPTS
13954 00003F0A E670          <1> out CMOS_PORT, al ; ADDRESS LOCATION AND DISABLE NMI
13955 00003F0C 88E0          <1> mov al, ah ; GET THE DATA BYTE TO WRITE
13956 00003F0E E671          <1> out CMOS_DATA, al ; PLACE IN REQUESTED CMOS LOCATION

```



13957	00003F10	B01E	<1>	mov	al, CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
13958	00003F12	D0D8	<1>	rcr	al, 1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
13959	00003F14	E670	<1>	out	CMOS_PORT, al ; SET DEFAULT TO READ ONLY REGISTER
13960	00003F16	90	<1>	nop	; I/O DELAY
13961	00003F17	E471	<1>	in	al, CMOS_DATA ; OPEN STANDBY LATCH
13962			<1>	;pop	ax ; RESTORE WORK REGISTERS
13963	00003F19	9D	<1>	popf	
13964	00003F1A	C3	<1>	RETN	
13965			<1>		
13966			<1>	bf_init:	
13967			<1>		; 14/08/2015
13968			<1>		; 02/07/2015
13969			<1>		; 01/07/2015
13970			<1>		; 15/04/2015 (Retro UNIX 386 v1 - Beginning)
13971			<1>		; Buffer (pointer) initialization !
13972			<1>		;
13973			<1>		; 17/07/2013 - 24/07/2013
13974			<1>		; Retro UNIX 8086 v1 (U9.ASM)
13975			<1>		; (Retro UNIX 8086 v1 feature only !)
13976			<1>		;
13977	00003F1B	BF[4A740000]	<1>	mov	edi, bufp
13978	00003F20	B8[207D0000]	<1>	mov	eax, buffer + (nbuf*520)
13979	00003F25	29D2	<1>	sub	edx, edx
13980	00003F27	FECA	<1>	dec	dl
13981	00003F29	31C9	<1>	xor	ecx, ecx
13982	00003F2B	49	<1>	dec	ecx
13983			<1>	bi0:	
13984	00003F2C	2D08020000	<1>	sub	eax, 520 ; 8 header + 512 data
13985	00003F31	AB	<1>	stosd	
13986	00003F32	89C6	<1>	mov	esi, eax
13987	00003F34	8916	<1>	mov	[esi], edx ; 000000FFh
13988			<1>		; Not a valid device sign
13989	00003F36	894E04	<1>	mov	[esi+4], ecx ; 0FFFFFFFh
13990			<1>		; Not a valid block number sign
13991	00003F39	3D[00750000]	<1>	cmp	eax, buffer
13992	00003F3E	77EC	<1>	ja	short bi0
13993	00003F40	B8[207D0000]	<1>	mov	eax, sb0
13994	00003F45	AB	<1>	stosd	
13995	00003F46	B8[287F0000]	<1>	mov	eax, sb1
13996	00003F4B	AB	<1>	stosd	
13997	00003F4C	89C6	<1>	mov	esi, eax ; offset sb1
13998	00003F4E	8916	<1>	mov	[esi], edx ; 000000FFh
13999			<1>		; Not a valid device sign
14000	00003F50	894E04	<1>	mov	[esi+4], ecx ; 0FFFFFFFh
14001			<1>		; Not a valid block number sign
14002			<1>		; 14/08/2015
14003			<1>	;call	rdev_init
14004			<1>	;retn	
14005			<1>		
14006			<1>	rdev_init:	; root device, super block buffer initialization
14007			<1>		; 14/08/2015
14008			<1>		; Retro UNIX 386 v1 feature only !
14009			<1>		;
14010			<1>		; NOTE: Disk partitions (file systems), logical
14011			<1>		; drive initialization, partition's start sector etc.
14012			<1>		; will be coded here, later in 'ldrv_init'
14013			<1>		
14014	00003F53	0FB605[D46D0000]	<1>	movzx	eax, byte [boot_drv]
14015			<1>	rdi_0:	
14016	00003F5A	3C80	<1>	cmp	al, 80h
14017	00003F5C	7202	<1>	jb	short rdi_1
14018	00003F5E	2C7E	<1>	sub	al, 7Eh ; 80h = 2 (hd0), 81h = 3 (hd1)
14019			<1>	rdi_1:	
14020	00003F60	A2[68740000]	<1>	mov	[rdev], al
14021	00003F65	BB[207D0000]	<1>	mov	ebx, sb0 ; super block buffer
14022	00003F6A	8903	<1>	mov	[ebx], eax
14023	00003F6C	B001	<1>	mov	al, 1 ; eax = 1
14024	00003F6E	894304	<1>	mov	[ebx+4], eax ; super block address on disk
14025	00003F71	E82A240000	<1>	call	diskio
14026	00003F76	C3	<1>	retn	
14027			<1>		
14028			<1>		; 23/10/2015
14029			<1>	com1_irq4:	
14030	00003F77	[7F3F0000]	<1>	dd	dummy_retn
14031			<1>	com2_irq3:	
14032	00003F7B	[7F3F0000]	<1>	dd	dummy_retn
14033			<1>		
14034			<1>	dummy_retn:	
14035	00003F7F	C3	<1>	retn	
14036				%include	'u1.s' ; 10/05/2015
14037			<1>		; Retro UNIX 386 v1 Kernel (v0.2) - SYS1.INC
14038			<1>		; Last Modification: 23/11/2015
14039			<1>		;
14040			<1>		; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
14041			<1>		; (v0.1 - Beginning: 11/07/2012)
14042			<1>		;
14043			<1>		; Derived from UNIX Operating System (v1.0 for PDP-11)
14044			<1>		; (Original) Source Code by Ken Thompson (1971-1972)
14045			<1>		; <Bell Laboratories (17/3/1972)>
14046			<1>		; <Preliminary Release of UNIX Implementation Document>
14047			<1>		;
14048			<1>		; Retro UNIX 8086 v1 - U1.ASM (12/07/2014) //// UNIX v1 -> u1.s

```

14049 <1> ;
14050 <1> ; *****
14051 <1>
14052 <1> unkni: ; / used for all system calls
14053 <1> sysent: ; < enter to system call >
14054 <1> ;19/10/2015
14055 <1> ; 21/09/2015
14056 <1> ; 01/07/2015
14057 <1> ; 19/05/2015
14058 <1> ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14059 <1> ; 10/04/2013 - 18/01/2014 (Retro UNIX 8086 v1)
14060 <1> ;
14061 <1> ; 'unkni' or 'sysent' is sytem entry from various traps.
14062 <1> ; The trap type is determined and an indirect jump is made to
14063 <1> ; the appropriate system call handler. If there is a trap inside
14064 <1> ; the system a jump to panic is made. All user registers are saved
14065 <1> ; and u.sp points to the end of the users stack. The sys (trap)
14066 <1> ; instructor is decoded to get the the system code part (see
14067 <1> ; trap instruction in the PDP-11 handbook) and from this
14068 <1> ; the indirect jump address is calculated. If a bad system call is
14069 <1> ; made, i.e., the limits of the jump table are exceeded, 'badsys'
14070 <1> ; is called. If the call is legitimate control passes to the
14071 <1> ; appropriate system routine.
14072 <1> ;
14073 <1> ; Calling sequence:
14074 <1> ; Through a trap caused by any sys call outside the system.
14075 <1> ; Arguments:
14076 <1> ; Arguments of particular system call.
14077 <1> ; .....
14078 <1> ;
14079 <1> ; Retro UNIX 8086 v1 modification:
14080 <1> ; System call number is in EAX register.
14081 <1> ;
14082 <1> ; Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
14083 <1> ; registers depending of function details.
14084 <1> ;
14085 <1> ; 16/04/2015
14086 00003F80 368925[78740000] <1> mov [ss:u.sp], esp ; Kernel stack points to return address
14087 <1> ; save user registers
14088 00003F87 1E <1> push ds
14089 00003F88 06 <1> push es
14090 00003F89 0FA0 <1> push fs
14091 00003F8B 0FA8 <1> push gs
14092 00003F8D 60 <1> pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
14093 <1> ;
14094 <1> ; ESPACE = esp - [ss:u.sp] ; 4*12 = 48 ; 17/09/2015
14095 <1> ; (ESPACE is size of space in kernel stack
14096 <1> ; for saving/restoring user registers.)
14097 <1> ;
14098 00003F8E 50 <1> push eax ; 01/07/2015
14099 00003F8F 66B81000 <1> mov ax, KDATA
14100 00003F93 8ED8 <1> mov ds, ax
14101 00003F95 8EC0 <1> mov es, ax
14102 00003F97 8EE0 <1> mov fs, ax
14103 00003F99 8EE8 <1> mov gs, ax
14104 00003F9B A1[A8700000] <1> mov eax, [k_page_dir]
14105 00003FA0 0F22D8 <1> mov cr3, eax
14106 00003FA3 58 <1> pop eax ; 01/07/2015
14107 <1> ; 19/10/2015
14108 00003FA4 FC <1> cld
14109 <1> ;
14110 00003FA5 FE05[77740000] <1> inc byte [sysflg]
14111 <1> ; incb sysflg / indicate a system routine is in progress
14112 00003FAB FB <1> sti ; 18/01/2014
14113 00003FAC 0F8560F9FFFF <1> jnz panic ; 24/05/2013
14114 <1> ; beq lf
14115 <1> ; jmp panic ; / called if trap inside system
14116 <1> ;1:
14117 <1> ; 16/04/2015
14118 00003FB2 A3[80740000] <1> mov [u.r0], eax
14119 00003FB7 8925[7C740000] <1> mov [u.usp], esp ; kernel stack points to user's registers
14120 <1> ;
14121 <1> ; mov $s.syst+2,clockp
14122 <1> ; mov r0,-(sp) / save user registers
14123 <1> ; mov sp,u.r0 / pointer to bottom of users stack
14124 <1> ; / in u.r0
14125 <1> ; mov r1,-(sp)
14126 <1> ; mov r2,-(sp)
14127 <1> ; mov r3,-(sp)
14128 <1> ; mov r4,-(sp)
14129 <1> ; mov r5,-(sp)
14130 <1> ; mov ac,-(sp) / "accumulator" register for extended
14131 <1> ; / arithmetic unit
14132 <1> ; mov mq,-(sp) / "multiplier quotient" register for the
14133 <1> ; / extended arithmetic unit
14134 <1> ; mov sc,-(sp) / "step count" register for the extended
14135 <1> ; / arithmetic unit
14136 <1> ; mov sp,u.sp / u.sp points to top of users stack
14137 <1> ; mov 18.(sp),r0 / store pc in r0
14138 <1> ; mov -(r0),r0 / sys inst in r0 10400xxx
14139 <1> ; sub $sys,r0 / get xxx code
14140 00003FBD C1E002 <1> shl eax, 2

```

14141		<1>	; asl r0 / multiply by 2 to jump indirect in bytes
14142	00003FC0 3D94000000	<1>	cmp eax, end_of_syscalls - syscalls
14143		<1>	; cmp r0,\$2f-1f / limit of table (35) exceeded
14144		<1>	;jnb short badsys
14145		<1>	; bhis badsys / yes, bad system call
14146	00003FC5 F5	<1>	cmc
14147	00003FC6 9C	<1>	pushf
14148	00003FC7 50	<1>	push eax
14149	00003FC8 8B2D[78740000]	<1>	mov ebp, [u.sp] ; Kernel stack at the beginning of sys call
14150	00003FCE B0FE	<1>	mov al, 0FEh ; 11111110b
14151	00003FD0 1400	<1>	adc al, 0 ; al = al + cf
14152	00003FD2 204508	<1>	and [ebp+8], al ; flags (reset carry flag)
14153		<1>	; bic \$341,20.(sp) / set users processor priority to 0
14154		<1>	; / and clear carry bit
14155	00003FD5 5D	<1>	pop ebp ; eax
14156	00003FD6 9D	<1>	popf
14157	00003FD7 0F8248010000	<1>	jc badsys
14158	00003FDD A1[80740000]	<1>	mov eax, [u.r0]
14159		<1>	; system call registers: EAX, EDX, ECX, EBX, ESI, EDI
14160	00003FE2 FFA5[E83F0000]	<1>	jmp dword [ebp+syscalls]
14161		<1>	; jmp *1f(r0) / jump indirect thru table of addresses
14162		<1>	; / to proper system routine.
14163		<1>	syscalls: ; 1:
14164		<1>	; 21/09/2015
14165		<1>	; 01/07/2015
14166		<1>	; 16/04/2015 (32 bit address modification)
14167	00003FE8 [EF400000]	<1>	dd sysrele ; / 0
14168	00003FEC [95410000]	<1>	dd sysexit ; / 1
14169	00003FF0 [BA420000]	<1>	dd sysfork ; / 2
14170	00003FF4 [CD430000]	<1>	dd sysread ; / 3
14171	00003FF8 [E8430000]	<1>	dd syswrite ; / 4
14172	00003FFC [52440000]	<1>	dd sysopen ; / 5
14173	00004000 [82450000]	<1>	dd sysclose ; / 6
14174	00004004 [3C420000]	<1>	dd syswait ; / 7
14175	00004008 [02450000]	<1>	dd syscreat ; / 8
14176	0000400C [A9480000]	<1>	dd syslink ; / 9
14177	00004010 [6B490000]	<1>	dd sysunlink ; / 10
14178	00004014 [3B4A0000]	<1>	dd sysexec ; / 11
14179	00004018 [A2500000]	<1>	dd syschdir ; / 12
14180	0000401C [86510000]	<1>	dd systime ; / 13
14181	00004020 [39450000]	<1>	dd sysmkdir ; / 14
14182	00004024 [F4500000]	<1>	dd syschmod ; / 15
14183	00004028 [56510000]	<1>	dd syschown ; / 16
14184	0000402C [B9510000]	<1>	dd sysbreak ; / 17
14185	00004030 [134E0000]	<1>	dd sysstat ; / 18
14186	00004034 [7E520000]	<1>	dd sysseek ; / 19
14187	00004038 [90520000]	<1>	dd systell ; / 20
14188	0000403C [8C5D0000]	<1>	dd sysmount ; / 21
14189	00004040 [3E5E0000]	<1>	dd sysumount ; / 22
14190	00004044 [0D530000]	<1>	dd syssetuid ; / 23
14191	00004048 [3E530000]	<1>	dd sysgetuid ; / 24
14192	0000404C [95510000]	<1>	dd sysstime ; / 25
14193	00004050 [01530000]	<1>	dd sysquit ; / 26
14194	00004054 [F5520000]	<1>	dd sysintr ; / 27
14195	00004058 [EF4D0000]	<1>	dd sysfstat ; / 28
14196	0000405C [9E450000]	<1>	dd sysemu ; / 29
14197	00004060 [CC450000]	<1>	dd sysmdate ; / 30
14198	00004064 [17460000]	<1>	dd sysstty ; / 31
14199	00004068 [96470000]	<1>	dd sysgtty ; / 32
14200	0000406C [C7450000]	<1>	dd sysilgins ; / 33
14201	00004070 [7A660000]	<1>	dd sysssleep ; 34 ; Retro UNIX 8086 v1 feature only !
14202		<1>	; 11/06/2014
14203	00004074 [A9660000]	<1>	dd sysmsg ; 35 ; Retro UNIX 386 v1 feature only !
14204		<1>	; 01/07/2015
14205	00004078 [80670000]	<1>	dd sysgeterr ; 36 ; Retro UNIX 386 v1 feature only !
14206		<1>	; 21/09/2015 - get last error number
14207		<1>	end_of_syscalls:
14208		<1>	
14209		<1>	error:
14210		<1>	; 17/09/2015
14211		<1>	; 03/09/2015
14212		<1>	; 01/09/2015
14213		<1>	; 09/06/2015
14214		<1>	; 13/05/2015
14215		<1>	; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14216		<1>	; 10/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
14217		<1>	;
14218		<1>	; 'error' merely sets the error bit off the processor status (c-bit)
14219		<1>	; then falls right into the 'sysret', 'sysrele' return sequence.
14220		<1>	;
14221		<1>	; INPUTS -> none
14222		<1>	; OUTPUTS ->
14223		<1>	; processor status - carry (c) bit is set (means error)
14224		<1>	;
14225		<1>	; 26/05/2013 (Stack pointer must be reset here!
14226		<1>	; Because, jumps to error procedure
14227		<1>	; disrupts push-pop nesting balance)
14228		<1>	;
14229	0000407C 8B2D[78740000]	<1>	mov ebp, [u.sp] ; interrupt (system call) return (iretd) address
14230	00004082 804D0801	<1>	or byte [ebp+8], 1 ; set carry bit of flags register
14231		<1>	; (system call will return with cf = 1)
14232		<1>	; bis \$1,20.(r1) / set c bit in processor status word below



14325		<1>	; beq 1f / been modified? no, 1f
14326	000040DB A2[76740000]	<1>	mov [mmod], al ; 0
14327		<1>	; clrb mmod / yes, clear mmod
14328		<1>	;mov ax, [mntd]
14329		<1>	;;mov al, [mdev] ; 26/04/2013
14330	000040E0 BB[287F0000]	<1>	mov ebx, sb1 ;; 07/08//2013
14331		<1>	;;mov[ebx], al
14332		<1>	;mov [sb1], al
14333		<1>	; movb mntd,sb1 / set the I/O queue
14334	000040E5 66810B0002	<1>	or word [ebx], 200h
14335		<1>	;or word [sb1], 200h ; write bit, bit 9
14336		<1>	; bis \$1000,sb1 / set write bit in I/O queue for detached sb
14337	000040EA E896210000	<1>	call poke ; 07/08/2013
14338		<1>	;call ppoke
14339		<1>	; jsr r0,ppoke / write it out to its device
14340		<1>	;xor al, al ; 26/04/2013
14341		<1>	;1:
14342		<1>	; tstb uquant / is the time quantum 0?
14343		<1>	; bne 1f / no, don't swap it out
14344		<1>	
14345		<1>	sysrele: ; < release >
14346		<1>	; 14/10/2015
14347		<1>	; 01/09/2015
14348		<1>	; 24/07/2015
14349		<1>	; 14/05/2015
14350		<1>	; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14351		<1>	; 10/04/2013 - 07/03/2014 (Retro UNIX 8086 v1)
14352		<1>	;
14353		<1>	; 'sysrele' first calls 'tswap' if the time quantum for a user is
14354		<1>	; zero (see 'sysret'). It then restores the user's registers and
14355		<1>	; turns off the system flag. It then checked to see if there is
14356		<1>	; an interrupt from the user by calling 'isintr'. If there is,
14357		<1>	; the output gets flashed (see isintr) and interrupt action is
14358		<1>	; taken by a branch to 'intract'. If there is no interrupt from
14359		<1>	; the user, a rti is made.
14360		<1>	;
14361		<1>	; Calling sequence:
14362		<1>	; Fall through a 'bne' in 'sysret' & ?
14363		<1>	; Arguments:
14364		<1>	;
14365		<1>	; .....
14366		<1>	;
14367		<1>	; 23/02/2014 (swapret)
14368		<1>	; 22/09/2013
14369		<1>	sysrel0: ;1:
14370	000040EF 803D[BC740000]00	<1>	cmp byte [u.quant], 0 ; 16/05/2013
14371		<1>	; tstb uquant / is the time quantum 0?
14372	000040F6 7705	<1>	ja short swapret
14373		<1>	; bne 1f / no, don't swap it out
14374		<1>	sysrelease: ; 07/12/2013 (jump from 'clock')
14375	000040F8 E893120000	<1>	call tswap
14376		<1>	; jsr r0,tswap / yes, swap it out
14377		<1>	;
14378		<1>	; Retro Unix 8086 v1 feature: return from 'swap' to 'swapret' address.
14379		<1>	swapret: ;1:
14380		<1>	; 10/09/2015
14381		<1>	; 01/09/2015
14382		<1>	; 14/05/2015
14383		<1>	; 16/04/2015 (Retro UNIX 386 v1 - 32 bit, pm modifications)
14384		<1>	; 26/05/2013 (Retro UNIX 8086 v1)
14385		<1>	; cli
14386		<1>	; 24/07/2015
14387		<1>	;
14388		<1>	;; 'esp' must be already equal to '[u.usp]' here !
14389		<1>	;; mov esp, [u.usp]
14390		<1>	
14391		<1>	; 22/09/2013
14392	000040FD E878140000	<1>	call isintr
14393		<1>	; 20/10/2013
14394	00004102 7405	<1>	jz short sysrel1
14395	00004104 E875000000	<1>	call intract
14396		<1>	; jsr r0,isintr / is there an interrupt from the user
14397		<1>	; br intract / yes, output gets flushed, take interrupt
14398		<1>	; / action
14399		<1>	sysrel1:
14400	00004109 FA	<1>	cli ; 14/10/2015
14401	0000410A FE0D[77740000]	<1>	dec byte [sysflg]
14402		<1>	; decb sysflg / turn system flag off
14403	00004110 A1[D3740000]	<1>	mov eax, [u.pgdir]
14404	00004115 0F22D8	<1>	mov cr3, eax ; 1st PDE points to Kernel Page Table 0 (1st 4 MB)
14405		<1>	; (others are different than kernel page tables)
14406		<1>	; 10/09/2015
14407	00004118 61	<1>	popad ; edi, esi, ebp, temp (increment esp by 4), ebx, edx, ecx, eax
14408		<1>	; mov (sp)+,sc / restore user registers
14409		<1>	; mov (sp)+,mq
14410		<1>	; mov (sp)+,ac
14411		<1>	; mov (sp)+,r5
14412		<1>	; mov (sp)+,r4
14413		<1>	; mov (sp)+,r3
14414		<1>	; mov (sp)+,r2
14415		<1>	;
14416	00004119 A1[80740000]	<1>	mov eax, [u.r0] ; ((return value in EAX))

```

14417 0000411E 0FA9      <1>  pop    gs
14418 00004120 0FA1      <1>  pop    fs
14419 00004122 07        <1>  pop    es
14420 00004123 1F        <1>  pop    ds
14421 00004124 CF        <1>  iretd
14422                      <1>      ; rti / no, return from interrupt
14423                      <1>
14424                      <1> badsys:
14425                      <1>      ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14426                      <1>      ; (Major Modification: 'core' dumping procedure in
14427                      <1>      ;          original UNIX v1 and Retro UNIX 8086 v1
14428                      <1>      ;          has been changed to print 'Invalid System Call !'
14429                      <1>      ;          message on the user's console tty.)
14430                      <1>      ; (EIP, EAX values will be shown on screen with error message)
14431                      <1>      ; (EIP = Return address just after the system call -INT 30h-)
14432                      <1>      ; (EAX = Function number)
14433                      <1>      ;
14434 00004125 FE05[C8740000] <1>  inc     byte [u.bsys]
14435                      <1>      ;
14436 0000412B 8B1D[78740000] <1>  mov     ebx, [u.sp] ; esp at the beginning of 'sysent'
14437 00004131 8B03      <1>  mov     eax, [ebx] ; EIP (return address, not 'INT 30h' address)
14438 00004133 E8E7D7FFFF      <1>  call    dwordtohex
14439 00004138 8915[A06D0000] <1>  mov     [bsys_msg_eip], edx
14440 0000413E A3[A46D0000] <1>  mov     [bsys_msg_eip+4], eax
14441 00004143 A1[80740000] <1>  mov     eax, [u.r0]
14442 00004148 E8D2D7FFFF      <1>  call    dwordtohex
14443 0000414D 8915[906D0000] <1>  mov     [bsys_msg_eax], edx
14444 00004153 A3[946D0000] <1>  mov     [bsys_msg_eax+4], eax
14445 00004158 31C0      <1>  xor     eax, eax
14446 0000415A C705[A0740000]- <1>      mov     dword [u.base], badsys_msg ; "Invalid System call !"
14447 00004160 [716D0000] <1>
14448 00004164 8B1D[90740000] <1>  mov     ebx, [u.fofp]
14449 0000416A 8903      <1>  mov     [ebx], eax
14450                      <1> ;mov  eax, 1 ; inode number of console tty (for user)
14451 0000416C 40          <1>  inc     eax
14452 0000416D C705[A4740000]3B00- <1>  mov     dword [u.count], BSYS_M_SIZE
14453 00004175 0000      <1>
14454                      <1>      ; writei
14455                      <1>      ; INPUTS ->
14456                      <1>      ;      r1 - inode number
14457                      <1>      ;      u.count - byte count to be written
14458                      <1>      ;      u.base - points to user buffer
14459                      <1>      ;      u.fofp - points to word with current file offset
14460                      <1>      ; OUTPUTS ->
14461                      <1>      ;      u.count - cleared
14462                      <1>      ;      u.nread - accumulates total bytes passed back
14463                      <1>      ;
14464                      <1>      ; ((Modified registers: EDX, EBX, ECX, ESI, EDI, EBP))
14465 00004177 E874190000      <1>  call    writei
14466                      <1> ;mov  eax, 1
14467 0000417C EB17      <1>  jmp     sysexit
14468                      <1>
14469                      <1>      ; incb u.bsys / turn on the user's bad-system flag
14470                      <1>      ; mov $3f,u.namep / point u.namep to "core\0\0"
14471                      <1>      ; jsr r0,namei / get the i-number for the core image file
14472                      <1>      ; br 1f / error
14473                      <1>      ; neg r1 / negate the i-number to open the core image file
14474                      <1>      ;      ; / for writing
14475                      <1>      ; jsr r0,iopen / open the core image file
14476                      <1>      ; jsr r0,itrunc / free all associated blocks
14477                      <1>      ; br 2f
14478                      <1> ;1:
14479                      <1>      ; mov $17,r1 / put i-node mode (17) in r1
14480                      <1>      ; jsr r0,maknod / make an i-node
14481                      <1>      ; mov u.dirbuf,r1 / put i-node number in r1
14482                      <1> ;2:
14483                      <1>      ; mov $core,u.base / move address core to u.base
14484                      <1>      ; mov $ecore-core,u.count / put the byte count in u.count
14485                      <1>      ; mov $u.off,u.fofp / more user offset to u.fofp
14486                      <1>      ; clr u.off / clear user offset
14487                      <1>      ; jsr r0,writei / write out the core image to the user
14488                      <1>      ; mov $user,u.base / pt. u.base to user
14489                      <1>      ; mov $64.,u.count / u.count = 64
14490                      <1>      ; jsr r0,writei / write out all the user parameters
14491                      <1>      ; neg r1 / make i-number positive
14492                      <1>      ; jsr r0,iclose / close the core image file
14493                      <1>      ; br sysexit /
14494                      <1> ;3:
14495                      <1>      ; <core\0\0>
14496                      <1>
14497                      <1> intract: ; / interrupt action
14498                      <1>      ; 14/10/2015
14499                      <1>      ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14500                      <1>      ; 09/05/2013 - 07/12/2013 (Retro UNIX 8086 v1)
14501                      <1>      ;
14502                      <1>      ; Retro UNIX 8086 v1 modification !
14503                      <1>      ; (Process/task switching and quit routine by using
14504                      <1>      ; Retro UNIX 8086 v1 keyboard interrupt output.))
14505                      <1>      ;
14506                      <1>      ; input -> 'u.quit' (also value of 'u.intr' > 0)
14507                      <1>      ; output -> If value of 'u.quit' = FFFFh ('ctrl+brk' sign)
14508                      <1>      ;          'intract' will jump to 'sysexit'.

```

```

14509      <1>      ;          Intract will return to the caller
14510      <1>      ;          if value of 'u.quit' <> FFFFh.
14511      <1>      ; 14/10/2015
14512 0000417E FB      <1>      sti
14513      <1>      ; 07/12/2013
14514 0000417F 66FF05[C0740000] <1>      inc word [u.quit]
14515 00004186 7408      <1>      jz short intrct0 ; FFFFh -> 0
14516 00004188 66FF0D[C0740000] <1>      dec word [u.quit]
14517      <1>      ; 16/04/2015
14518 0000418F C3      <1>      retn
14519      <1> intrct0:
14520 00004190 58      <1>      pop eax ; call intract -> retn
14521      <1>      ;
14522 00004191 31C0      <1>      xor eax, eax
14523 00004193 FEC0      <1>      inc al ; mov ax, 1
14524      <1> ;;;
14525      <1> ; UNIX v1 original 'intract' routine...
14526      <1> ; / interrupt action
14527      <1>      ; cmp *(sp), $rti / are you in a clock interrupt?
14528      <1>      ; bne lf / no, lf
14529      <1>      ; cmp (sp)+, (sp)+ / pop clock pointer
14530      <1> ; 1: / now in user area
14531      <1>      ; mov r1, -(sp) / save r1
14532      <1>      ; mov u.ttyp, r1
14533      <1>      ; / pointer to tty buffer in control-to r1
14534      <1>      ; cmpb 6(r1), $177
14535      <1>      ; / is the interrupt char equal to "del"
14536      <1>      ; beq lf / yes, lf
14537      <1>      ; clrb 6(r1)
14538      <1>      ; / no, clear the byte
14539      <1>      ; / (must be a quit character)
14540      <1>      ; mov (sp)+, r1 / restore r1
14541      <1>      ; clr u.quit / clear quit flag
14542      <1>      ; bis $20, 2(sp)
14543      <1>      ; / set trace for quit (sets t bit of
14544      <1>      ; / ps-trace trap)
14545      <1>      ; rti ; / return from interrupt
14546      <1> ; 1: / interrupt char = del
14547      <1>      ; clrb 6(r1) / clear the interrupt byte
14548      <1>      ; / in the buffer
14549      <1>      ; mov (sp)+, r1 / restore r1
14550      <1>      ; cmp u.intr, $core / should control be
14551      <1>      ; / transferred to loc core?
14552      <1>      ; blo lf
14553      <1>      ; jmp *u.intr / user to do rti yes,
14554      <1>      ; / transfer to loc core
14555      <1> ; 1:
14556      <1>      ; sys 1 / exit
14557      <1>
14558      <1> sysexit: ; <terminate process>
14559      <1> ; 01/09/2015
14560      <1> ; 31/08/2015
14561      <1> ; 14/05/2015
14562      <1> ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14563      <1> ; 19/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
14564      <1> ;
14565      <1> ; 'sysexit' terminates a process. First each file that
14566      <1> ; the process has opened is closed by 'flose'. The process
14567      <1> ; status is then set to unused. The 'p.pid' table is then
14568      <1> ; searched to find children of the dying process. If any of
14569      <1> ; children are zombies (died by not waited for), they are
14570      <1> ; set free. The 'p.pid' table is then searched to find the
14571      <1> ; dying process's parent. When the parent is found, it is
14572      <1> ; checked to see if it is free or it is a zombie. If it is
14573      <1> ; one of these, the dying process just dies. If it is waiting
14574      <1> ; for a child process to die, it notified that it doesn't
14575      <1> ; have to wait anymore by setting it's status from 2 to 1
14576      <1> ; (waiting to active). It is awakened and put on runq by
14577      <1> ; 'putlu'. The dying process enters a zombie state in which
14578      <1> ; it will never be run again but stays around until a 'wait'
14579      <1> ; is completed by it's parent process. If the parent is not
14580      <1> ; found, process just dies. This means 'swap' is called with
14581      <1> ; 'u.uno=0'. What this does is the 'wswap' is not called
14582      <1> ; to write out the process and 'rswap' reads the new process
14583      <1> ; over the one that dies..i.e., the dying process is
14584      <1> ; overwritten and destroyed.
14585      <1> ;
14586      <1> ; Calling sequence:
14587      <1> ; sysexit or conditional branch.
14588      <1> ; Arguments:
14589      <1> ; -
14590      <1> ; .....
14591      <1> ;
14592      <1> ; Retro UNIX 8086 v1 modification:
14593      <1> ; System call number (=1) is in EAX register.
14594      <1> ;
14595      <1> ; Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
14596      <1> ; registers depending of function details.
14597      <1> ;
14598      <1> ; ('swap' procedure is mostly different than original UNIX v1.)
14599      <1> ;
14600      <1> ; / terminate process

```

14601		<1>	; AX = 1
14602	00004195 6648	<1>	dec ax ; 0
14603	00004197 66A3[BE740000]	<1>	mov [u.intr], ax ; 0
14604		<1>	; clr u.intr / clear interrupt control word
14605		<1>	; clr r1 / clear r1
14606		<1>	; AX = 0
14607		<1>	sysexit_1: ; 1:
14608		<1>	; AX = File descriptor
14609		<1>	; / r1 has file descriptor (index to u.fp list)
14610		<1>	; / Search the whole list
14611	0000419D E8070D0000	<1>	call fclose
14612		<1>	; jsr r0,fclose / close all files the process opened
14613		<1>	;; ignore error return
14614		<1>	; br .+2 / ignore error return
14615		<1>	;inc ax
14616	000041A2 FEC0	<1>	inc al
14617		<1>	; inc r1 / increment file descriptor
14618		<1>	;cmp ax, 10
14619	000041A4 3C0A	<1>	cmp al, 10
14620		<1>	; cmp r1,\$10. / end of u.fp list?
14621	000041A6 72F5	<1>	jb short sysexit_1
14622		<1>	; blt 1b / no, go back
14623	000041A8 0FB61D[C9740000]	<1>	movzx ebx, byte [u.uno] ; 01/09/2015
14624		<1>	; movb u.uno,r1 / yes, move dying process's number to r1
14625	000041AF 88A3[05720000]	<1>	mov [ebx+p.stat-1], ah ; 0, SFREE, 05/02/2014
14626		<1>	; clrb p.stat-1(r1) / free the process
14627		<1>	;shl bx, 1
14628	000041B5 D0E3	<1>	shl bl, 1
14629		<1>	; asl r1 / use r1 for index into the below tables
14630	000041B7 668B8B[74710000]	<1>	mov cx, [ebx+p.pid-2]
14631		<1>	; mov p.pid-2(r1),r3 / move dying process's name to r3
14632	000041BE 668B93[94710000]	<1>	mov dx, [ebx+p.ppid-2]
14633		<1>	; mov p.ppid-2(r1),r4 / move its parents name to r4
14634		<1>	; xor bx, bx ; 0
14635	000041C5 30DB	<1>	xor bl, bl ; 0
14636		<1>	; clr r2
14637	000041C7 31F6	<1>	xor esi, esi ; 0
14638		<1>	; clr r5 / initialize reg
14639		<1>	sysexit_2: ; 1:
14640		<1>	; / find children of this dying process,
14641		<1>	; / if they are zombies, free them
14642		<1>	;add bx, 2
14643	000041C9 80C302	<1>	add bl, 2
14644		<1>	; add \$2,r2 / search parent process table
14645		<1>	; / for dying process's name
14646	000041CC 66398B[94710000]	<1>	cmp [ebx+p.ppid-2], cx
14647		<1>	; cmp p.ppid-2(r2),r3 / found it?
14648	000041D3 7513	<1>	jne short sysexit_4
14649		<1>	; bne 3f / no
14650		<1>	;shr bx, 1
14651	000041D5 D0EB	<1>	shr bl, 1
14652		<1>	; asr r2 / yes, it is a parent
14653	000041D7 80BB[05720000]03	<1>	cmp byte [ebx+p.stat-1], 3 ; SZOMB, 05/02/2014
14654		<1>	; cmpb p.stat-1(r2),\$3 / is the child of this
14655		<1>	; / dying process a zombie
14656	000041DE 7506	<1>	jne short sysexit_3
14657		<1>	; bne 2f / no
14658	000041E0 88A3[05720000]	<1>	mov [ebx+p.stat-1], ah ; 0, SFREE, 05/02/2014
14659		<1>	; clrb p.stat-1(r2) / yes, free the child process
14660		<1>	sysexit_3: ; 2:
14661		<1>	;shr bx, 1
14662	000041E6 D0E3	<1>	shl bl, 1
14663		<1>	; asl r2
14664		<1>	sysexit_4: ; 3:
14665		<1>	; / search the process name table
14666		<1>	; / for the dying process's parent
14667	000041E8 663993[74710000]	<1>	cmp [ebx+p.pid-2], dx ; 17/09/2013
14668		<1>	; cmp p.pid-2(r2),r4 / found it?
14669	000041EF 7502	<1>	jne short sysexit_5
14670		<1>	; bne 3f / no
14671	000041F1 89DE	<1>	mov esi, ebx
14672		<1>	; mov r2,r5 / yes, put index to p.pid table (parents
14673		<1>	; / process # x2) in r5
14674		<1>	sysexit_5: ; 3:
14675		<1>	;cmp bx, nproc + nproc
14676	000041F3 80FB20	<1>	cmp bl, nproc + nproc
14677		<1>	; cmp r2,\$nproc+nproc / has whole table been searched?
14678	000041F6 72D1	<1>	jb short sysexit_2
14679		<1>	; blt 1b / no, go back
14680		<1>	; mov r5,r1 / yes, r1 now has parents process # x2
14681	000041F8 21F6	<1>	and esi, esi ; r5=r1
14682	000041FA 7431	<1>	jz short sysexit_6
14683		<1>	; beq 2f / no parent has been found.
14684		<1>	; / The process just dies
14685	000041FC 66D1EE	<1>	shr si, 1
14686		<1>	; asr r1 / set up index to p.stat
14687	000041FF 8A86[05720000]	<1>	mov al, [esi+p.stat-1]
14688		<1>	; movb p.stat-1(r1),r2 / move status of parent to r2
14689	00004205 20C0	<1>	and al, al
14690	00004207 7424	<1>	jz short sysexit_6
14691		<1>	; beq 2f / if its been freed, 2f
14692	00004209 3C03	<1>	cmp al, 3



```

14693      <1>      ; cmp r2,$3 / is parent a zombie?
14694 0000420B 7420      <1>      je      short sysexit_6
14695      <1>      ; beq 2f / yes, 2f
14696      <1>      ; BH = 0
14697 0000420D 8A1D[C9740000]      <1>      mov     bl, [u.uno]
14698      <1>      ; movb u.uno,r3 / move dying process's number to r3
14699 00004213 C683[05720000]03      <1>      mov     byte [ebx+p.stat-1], 3 ; SZOMB, 05/02/2014
14700      <1>      ; movb $3,p.stat-1(r3) / make the process a zombie
14701      <1>      ; 05/02/2014
14702 0000421A 3C01      <1>      cmp     al, 1 ; SRUN
14703 0000421C 740F      <1>      je      short sysexit_6
14704      <1>      ;cmp al, 2
14705      <1>      ; cmp r2,$2 / is the parent waiting for
14706      <1>      ; / this child to die
14707      <1>      ;jne short sysexit_6
14708      <1>      ; bne 2f / yes, notify parent not to wait any more
14709      <1>      ; 05/02/2014
14710      <1>      ; p.stat = 2 --> waiting
14711      <1>      ; p.stat = 4 --> sleeping
14712 0000421E C686[05720000]01      <1>      mov     byte [esi+p.stat-1], 1 ; SRUN ; 05/02/2014
14713      <1>      ;dec byte [esi+p.stat-1]
14714      <1>      ; decb      p.stat-1(r1) / awaken it by putting it (parent)
14715 00004225 6689F0      <1>      mov     ax, si ; r1 (process number in AL)
14716      <1>      ;
14717      <1>      ;mov ebx, runq + 4
14718      <1>      ; mov $runq+4,r2 / on the runq
14719 00004228 E83B120000      <1>      call    putlu
14720      <1>      ; jsr r0, putlu
14721      <1>      sysexit_6: ; 2:
14722      <1>      ; 31/08/2015
14723      <1>      ; / the process dies
14724 0000422D C605[C9740000]00      <1>      mov     byte [u.uno], 0
14725      <1>      ; clrb u.uno / put zero as the process number,
14726      <1>      ; / so "swap" will
14727 00004234 E861110000      <1>      call    swap
14728      <1>      ; jsr r0,swap / overwrite process with another process
14729      <1>      hlt_sys:
14730      <1>      ;sti ; 18/01/2014
14731      <1>      hlts0:
14732 00004239 F4      <1>      hlt
14733 0000423A EBFD      <1>      jmp     short hlts0
14734      <1>      ; 0 / and thereby kill it; halt?
14735      <1>
14736      <1>
14737      <1>      syswait: ; < wait for a processs to die >
14738      <1>      ; 17/09/2015
14739      <1>      ; 02/09/2015
14740      <1>      ; 01/09/2015
14741      <1>      ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
14742      <1>      ; 24/05/2013 - 05/02/2014 (Retro UNIX 8086 v1)
14743      <1>      ;
14744      <1>      ; 'syswait' waits for a process die.
14745      <1>      ; It works in following way:
14746      <1>      ; 1) From the parent process number, the parent's
14747      <1>      ; process name is found. The p.ppid table of parent
14748      <1>      ; names is then searched for this process name.
14749      <1>      ; If a match occurs, r2 contains child's process
14750      <1>      ; number. The child status is checked to see if it is
14751      <1>      ; a zombie, i.e; dead but not waited for (p.stat=3)
14752      <1>      ; If it is, the child process is freed and it's name
14753      <1>      ; is put in (u.r0). A return is then made via 'sysret'.
14754      <1>      ; If the child is not a zombie, nothing happens and
14755      <1>      ; the search goes on through the p.ppid table until
14756      <1>      ; all processes are checked or a zombie is found.
14757      <1>      ; 2) If no zombies are found, a check is made to see if
14758      <1>      ; there are any children at all. If there are none,
14759      <1>      ; an error return is made. If there are, the parent's
14760      <1>      ; status is set to 2 (waiting for child to die),
14761      <1>      ; the parent is swapped out, and a branch to 'syswait'
14762      <1>      ; is made to wait on the next process.
14763      <1>      ;
14764      <1>      ; Calling sequence:
14765      <1>      ; ?
14766      <1>      ; Arguments:
14767      <1>      ; -
14768      <1>      ; Inputs: -
14769      <1>      ; Outputs: if zombie found, it's name put in u.r0.
14770      <1>      ; .....
14771      <1>      ;
14772      <1>
14773      <1>      ; / wait for a process to die
14774      <1>
14775      <1>      syswait_0:
14776 0000423C 0FB61D[C9740000]      <1>      movzx   ebx, byte [u.uno] ; 01/09/2015
14777      <1>      ; movb u.uno,r1 / put parents process number in r1
14778 00004243 D0E3      <1>      shl     bl, 1
14779      <1>      ;shl bx, 1
14780      <1>      ; asl r1 / x2 to get index into p.pid table
14781 00004245 668B83[74710000]      <1>      mov     ax, [ebx+p.pid-2]
14782      <1>      ; mov p.pid-2(r1),r1 / get the name of this process
14783 0000424C 31F6      <1>      xor     esi, esi
14784      <1>      ; clr r2

```

14785	0000424E	31C9	<1>	xor	ecx, ecx ; 30/10/2013
14786			<1>	;xor	cl, cl
14787			<1>		; clr r3 / initialize reg 3
14788			<1>	syswait_1:	; 1:
14789	00004250	6683C602	<1>	add	si, 2
14790			<1>		; add \$2,r2 / use r2 for index into p.ppid table
14791			<1>		; / search table of parent processes
14792			<1>		; / for this process name
14793	00004254	663B86[94710000]	<1>	cmp	ax, [esi+p.ppid-2]
14794			<1>		; cmp p.ppid-2(r2),r1 / r2 will contain the childs
14795			<1>		; / process number
14796	0000425B	7535	<1>	jne	short syswait_3
14797			<1>		;bne 3f / branch if no match of parent process name
14798			<1>	;inc	cx
14799	0000425D	FEC1	<1>	inc	cl
14800			<1>		;inc r3 / yes, a match, r3 indicates number of children
14801	0000425F	66D1EE	<1>	shr	si, 1
14802			<1>		; asr r2 / r2/2 to get index to p.stat table
14803			<1>		; The possible states ('p.stat' values) of a process are:
14804			<1>		0 = free or unused
14805			<1>		1 = active
14806			<1>		2 = waiting for a child process to die
14807			<1>		3 = terminated, but not yet waited for (zombie).
14808	00004262	80BE[05720000]03	<1>	cmp	byte [esi+p.stat-1], 3 ; SZOMB, 05/02/2014
14809			<1>		; cmpb p.stat-1(r2),\$3 / is the child process a zombie?
14810	00004269	7524	<1>	jne	short syswait_2
14811			<1>		; bne 2f / no, skip it
14812	0000426B	88BE[05720000]	<1>	mov	[esi+p.stat-1], bh ; 0
14813			<1>		; clrb p.stat-1(r2) / yes, free it
14814	00004271	66D1E6	<1>	shl	si, 1
14815			<1>		; asl r2 / r2x2 to get index into p.pid table
14816	00004274	0FB786[74710000]	<1>	movzx	eax, word [esi+p.pid-2]
14817	0000427B	A3[80740000]	<1>	mov	[u.r0], eax
14818			<1>		; mov p.pid-2(r2),*u.r0
14819			<1>		; / put childs process name in (u.r0)
14820			<1>		;
14821			<1>		; Retro UNIX 386 v1 modification ! (17/09/2015)
14822			<1>		;
14823			<1>		; Parent process ID -p.ppid- field (of the child process)
14824			<1>		; must be cleared in order to prevent infinitive 'syswait'
14825			<1>		; system call loop from the application/program if it calls
14826			<1>		; 'syswait' again (mistakenly) while there is not a zombie
14827			<1>		; or running child process to wait. ('forktest.s', 17/09/2015)
14828			<1>		;
14829			<1>		; Note: syswait will return with error if there is not a
14830			<1>		zombie or running process to wait.
14831			<1>		;
14832	00004280	6629C0	<1>	sub	ax, ax
14833	00004283	668986[94710000]	<1>	mov	[esi+p.ppid-2], ax ; 0 ; 17/09/2015
14834	0000428A	E910FEFFFF	<1>	jmp	sysret0 ; ax = 0
14835			<1>		;
14836			<1>	;jmp	sysret
14837			<1>		; br sysret1 / return cause child is dead
14838			<1>	syswait_2:	; 2:
14839	0000428F	66D1E6	<1>	shl	si, 1
14840			<1>		; asl r2 / r2x2 to get index into p.ppid table
14841			<1>	syswait_3:	; 3:
14842	00004292	6683FE20	<1>	cmp	si, nproc+nproc
14843			<1>		; cmp r2,\$nproc+nproc / have all processes been checked?
14844	00004296	72B8	<1>	jb	short syswait_1
14845			<1>		; blt 1b / no, continue search
14846			<1>	;and	cx, cx
14847	00004298	20C9	<1>	and	cl, cl
14848			<1>		; tst r3 / one gets here if there are no children
14849			<1>		; / or children that are still active
14850			<1>		; 30/10/2013
14851	0000429A	750B	<1>	jnz	short syswait_4
14852			<1>	;jz	error
14853			<1>		; beq error1 / there are no children, error
14854	0000429C	890D[80740000]	<1>	mov	[u.r0], ecx ; 0
14855	000042A2	E9D5FDFFFF	<1>	jmp	error
14856			<1>	syswait_4:	
14857	000042A7	8A1D[C9740000]	<1>	mov	bl, [u.uno]
14858			<1>		; movb u.uno,r1 / there are children so put
14859			<1>		; / parent process number in r1
14860	000042AD	FE83[05720000]	<1>	inc	byte [ebx+p.stat-1] ; 2, SWAIT, 05/02/2014
14861			<1>		; incb p.stat-1(r1) / it is waiting for
14862			<1>		; / other children to die
14863			<1>		; 04/11/2013
14864	000042B3	E8E2100000	<1>	call	swap
14865			<1>		; jsr r0,swap / swap it out, because it's waiting
14866	000042B8	EB82	<1>	jmp	syswait_0
14867			<1>		; br syswait / wait on next process
14868			<1>		
14869			<1>	sysfork:	; < create a new process >
14870			<1>		; 18/09/2015
14871			<1>		; 04/09/2015
14872			<1>		; 02/09/2015
14873			<1>		; 01/09/2015
14874			<1>		; 28/08/2015
14875			<1>		; 14/05/2015
14876			<1>		; 10/05/2015

```

14877 <1> ; 09/05/2015
14878 <1> ; 06/05/2015 (Retro UNIX 386 v1 - Beginning)
14879 <1> ; 24/05/2013 - 14/02/2014 (Retro UNIX 8086 v1)
14880 <1> ;
14881 <1> ; 'sysfork' creates a new process. This process is referred
14882 <1> ; to as the child process. This new process core image is
14883 <1> ; a copy of that of the caller of 'sysfork'. The only
14884 <1> ; distinction is the return location and the fact that (u.r0)
14885 <1> ; in the old process (parent) contains the process id (p.pid)
14886 <1> ; of the new process (child). This id is used by 'syswait'.
14887 <1> ; 'sysfork' works in the following manner:
14888 <1> ; 1) The process status table (p.stat) is searched to find
14889 <1> ; a process number that is unused. If none are found
14890 <1> ; an error occurs.
14891 <1> ; 2) when one is found, it becomes the child process number
14892 <1> ; and it's status (p.stat) is set to active.
14893 <1> ; 3) If the parent had a control tty, the interrupt
14894 <1> ; character in that tty buffer is cleared.
14895 <1> ; 4) The child process is put on the lowest priority run
14896 <1> ; queue via 'putlu'.
14897 <1> ; 5) A new process name is gotten from 'mpid' (actually
14898 <1> ; it is a unique number) and is put in the child's unique
14899 <1> ; identifier; process id (p.pid).
14900 <1> ; 6) The process name of the parent is then obtained and
14901 <1> ; placed in the unique identifier of the parent process
14902 <1> ; name is then put in 'u.r0'.
14903 <1> ; 7) The child process is then written out on disk by
14904 <1> ; 'wswap', i.e., the parent process is copied onto disk
14905 <1> ; and the child is born. (The child process is written
14906 <1> ; out on disk/drum with 'u.uno' being the child process
14907 <1> ; number.)
14908 <1> ; 8) The parent process number is then restored to 'u.uno'.
14909 <1> ; 9) The child process name is put in 'u.r0'.
14910 <1> ; 10) The pc on the stack sp + 18 is incremented by 2 to
14911 <1> ; create the return address for the parent process.
14912 <1> ; 11) The 'u.fp' list is then searched to see what files
14913 <1> ; the parent has opened. For each file the parent has
14914 <1> ; opened, the corresponding 'fsp' entry must be updated
14915 <1> ; to indicate that the child process also has opened
14916 <1> ; the file. A branch to 'sysret' is then made.

14917 <1> ;
14918 <1> ; Calling sequence:
14919 <1> ; from shell ?
14920 <1> ; Arguments:
14921 <1> ; -
14922 <1> ; Inputs: -
14923 <1> ; Outputs: *u.r0 - child process name
14924 <1> ; .....
14925 <1> ;
14926 <1> ; Retro UNIX 8086 v1 modification:
14927 <1> ; AX = r0 = PID (>0) (at the return of 'sysfork')
14928 <1> ; = process id of child a parent process returns
14929 <1> ; = process id of parent when a child process returns
14930 <1> ;
14931 <1> ; In original UNIX v1, sysfork is called and returns as
14932 <1> ; in following manner: (with an example: c library, fork)
14933 <1> ;
14934 <1> ; 1:
14935 <1> ; sys fork
14936 <1> ; br 1f / child process returns here
14937 <1> ; bes 2f / parent process returns here
14938 <1> ; / pid of new process in r0
14939 <1> ; rts pc
14940 <1> ; 2: / parent process conditionally branches here
14941 <1> ; mov $-1,r0 / pid = -1 means error return
14942 <1> ; rts pc
14943 <1> ;
14944 <1> ; 1: / child process branches here
14945 <1> ; clr r0 / pid = 0 in child process
14946 <1> ; rts pc
14947 <1> ;
14948 <1> ; In UNIX v7x86 (386) by Robert Nordier (1999)
14949 <1> ; // pid = fork();
14950 <1> ; //
14951 <1> ; // pid == 0 in child process;
14952 <1> ; // pid == -1 means error return
14953 <1> ; // in child,
14954 <1> ; // parents id is in par_uid if needed
14955 <1> ;
14956 <1> ; _fork:
14957 <1> ; mov $.fork,eax
14958 <1> ; int $0x30
14959 <1> ; jmp 1f
14960 <1> ; jnc 2f
14961 <1> ; jmp cerror
14962 <1> ; 1:
14963 <1> ; mov eax,_par_uid
14964 <1> ; xor eax,eax
14965 <1> ; 2:
14966 <1> ; ret
14967 <1> ;

```

```

14968 <1> ; In Retro UNIX 8086 v1,
14969 <1> ; 'sysfork' returns in following manner:
14970 <1> ;
14971 <1> ; mov ax, sys_fork
14972 <1> ; mov bx, offset @f ; routine for child
14973 <1> ; int 20h
14974 <1> ; jc error
14975 <1> ;
14976 <1> ; ; Routine for parent process here (just after 'jc')
14977 <1> ; mov word ptr [pid_of_child], ax
14978 <1> ; jmp next_routine_for_parent
14979 <1> ;
14980 <1> ; @@: ; routine for child process here
14981 <1> ; ....
14982 <1> ; NOTE: 'sysfork' returns to specified offset
14983 <1> ; for child process by using BX input.
14984 <1> ; (at first, parent process will return then
14985 <1> ; child process will return -after swapped in-
14986 <1> ; 'syswait' is needed in parent process
14987 <1> ; if return from child process will be waited for.)
14988 <1> ;
14989 <1> ;
14990 <1> ; / create a new process
14991 <1> ; EBX = return address for child process
14992 <1> ; (Retro UNIX 8086 v1 modification !)
14993 000042BA 31F6 <1> xor esi, esi
14994 <1> ; clr r1
14995 <1> sysfork_1: ; 1: / search p.stat table for unused process number
14996 000042BC 46 <1> inc esi
14997 <1> ; inc r1
14998 000042BD 80BE[05720000]00 <1> cmp byte [esi+p.stat-1], 0 ; SFREE, 05/02/2014
14999 <1> ; tstb p.stat-1(r1) / is process active, unused, dead
15000 000042C4 760B <1> jna short sysfork_2
15001 <1> ; beq 1f / it's unused so branch
15002 000042C6 6683FE10 <1> cmp si, nproc
15003 <1> ; cmp r1,$nproc / all processes checked
15004 000042CA 72F0 <1> jnb short sysfork_1
15005 <1> ; blt 1b / no, branch back
15006 <1> ;
15007 <1> ; Retro UNIX 8086 v1. modification:
15008 <1> ; Parent process returns from 'sysfork' to address
15009 <1> ; which is just after 'sysfork' system call in parent
15010 <1> ; process. Child process returns to address which is put
15011 <1> ; in BX register by parent process for 'sysfork'.
15012 <1> ;
15013 <1> ; add $2,18.(sp) / add 2 to pc when trap occurred, points
15014 <1> ; / to old process return
15015 <1> ; br error1 / no room for a new process
15016 000042CC E9ABFDFFFF <1> jmp error
15017 <1> sysfork_2: ; 1:
15018 000042D1 E861EDFFFF <1> call allocate_page
15019 000042D6 0F82A0FDFFFF <1> jc error
15020 000042DC 50 <1> push eax ; UPAGE (user structure page) address
15021 <1> ; Retro UNIX 386 v1 modification!
15022 000042DD E85EEFFFFFFF <1> call duplicate_page_dir
15023 <1> ; EAX = New page directory
15024 000042E2 730B <1> jnc short sysfork_3
15025 000042E4 58 <1> pop eax ; UPAGE (user structure page) address
15026 000042E5 E825EFFFFFFF <1> call deallocate_page
15027 000042EA E98DFDFFFF <1> jmp error
15028 <1> sysfork_3:
15029 <1> ; Retro UNIX 386 v1 modification !
15030 000042EF 56 <1> push esi
15031 000042F0 E81D110000 <1> call wswap ; save current user (u) structure, user registers
15032 <1> ; and interrupt return components (for IRET)
15033 000042F5 8705[D3740000] <1> xchg eax, [u.pgdir] ; page directory of the child process
15034 000042FB A3[D7740000] <1> mov [u.ppgdir], eax ; page directory of the parent process
15035 00004300 5E <1> pop esi
15036 00004301 58 <1> pop eax ; UPAGE (user structure page) address
15037 <1> ; [u.usp] = esp
15038 00004302 89F7 <1> mov edi, esi
15039 00004304 66C1E702 <1> shl di, 2
15040 00004308 8987[12720000] <1> mov [edi+p.upage-4], eax ; memory page for 'user' struct
15041 0000430E A3[CA740000] <1> mov [u.upage], eax ; memory page for 'user' struct (child)
15042 <1> ; 28/08/2015
15043 00004313 0FB605[C9740000] <1> movzx eax, byte [u.uno] ; parent process number
15044 <1> ; movb u.uno,-(sp) / save parent process number
15045 0000431A 89C7 <1> mov edi, eax
15046 0000431C 50 <1> push eax ; **
15047 0000431D 8A87[D5710000] <1> mov al, [edi+p.ttyc-1] ; console tty (parent)
15048 <1> ; 18/09/2015
15049 <1> ; mov [esi+p.ttyc-1], al ; set child's console tty
15050 <1> ; mov [esi+p.waitc-1], ah ; 0 ; reset child's wait channel
15051 00004323 668986[D5710000] <1> mov [esi+p.ttyc-1], ax ; al - set child's console tty
15052 <1> ; ah - reset child's wait channel
15053 0000432A 89F0 <1> mov eax, esi
15054 0000432C A2[C9740000] <1> mov [u.uno], al ; child process number
15055 <1> ; movb r1,u.uno / set child process number to r1
15056 00004331 FE86[05720000] <1> inc byte [esi+p.stat-1] ; 1, SRUN, 05/02/2014
15057 <1> ; incb p.stat-1(r1) / set p.stat entry for child
15058 <1> ; / process to active status
15059 <1> ; mov u.ttyp,r2 / put pointer to parent process'

```

15060		<1>	; / control tty buffer in r2
15061		<1>	; beq 2f / branch, if no such tty assigned
15062		<1>	; clrb 6(r2) / clear interrupt character in tty buffer
15063		<1>	; 2:
15064	00004337 53	<1>	push ebx ; * return address for the child process
15065		<1>	; * Retro UNIX 8086 v1 feature only !
15066		<1>	; (Retro UNIX 8086 v1 modification!)
15067		<1>	; mov \$runq+4,r2
15068	00004338 E82B110000	<1>	call putlu
15069		<1>	; jsr r0,putlu / put child process on lowest priority
15070		<1>	; / run queue
15071	0000433D 66D1E6	<1>	shl si, 1
15072		<1>	; asl r1 / multiply r1 by 2 to get index
15073		<1>	; / into p.pid table
15074	00004340 66FF05[6E740000]	<1>	inc word [mpid]
15075		<1>	; inc mpid / increment m.pid; get a new process name
15076	00004347 66A1[6E740000]	<1>	mov ax, [mpid]
15077	0000434D 668986[74710000]	<1>	mov [esi+p.pid-2], ax
15078		<1>	;mov mpid,p.pid-2(r1) / put new process name
15079		<1>	; / in child process' name slot
15080	00004354 5A	<1>	pop edx ; * return address for the child process
15081		<1>	; * Retro UNIX 8086 v1 feature only !
15082	00004355 5B	<1>	pop ebx ; **
15083		<1>	;mov ebx, [esp] ; ** parent process number
15084		<1>	; movb (sp),r2 / put parent process number in r2
15085	00004356 66D1E3	<1>	shl bx, 1
15086		<1>	;asl r2 / multiply by 2 to get index into below tables
15087		<1>	;movzx eax, word [ebx+p.pid-2]
15088	00004359 668B83[74710000]	<1>	mov ax, [ebx+p.pid-2]
15089		<1>	; mov p.pid-2(r2),r2 / get process name of parent
15090		<1>	; / process
15091	00004360 668986[94710000]	<1>	mov [esi+p.ppid-2], ax
15092		<1>	; mov r2,p.ppid-2(r1) / put parent process name
15093		<1>	; / in parent process slot for child
15094	00004367 A3[80740000]	<1>	mov [u.r0], eax
15095		<1>	; mov r2,*u.r0 / put parent process name on stack
15096		<1>	; / at location where r0 was saved
15097	0000436C 8B2D[78740000]	<1>	mov ebp, [u.sp] ; points to return address (EIP for IRET)
15098	00004372 895500	<1>	mov [ebp], edx ; *, CS:EIP -> EIP
15099		<1>	; * return address for the child process
15100		<1>	; mov \$sysret1,-(sp) /
15101		<1>	; mov sp,u.usp / contents of sp at the time when
15102		<1>	; / user is swapped out
15103		<1>	; mov \$sstack,sp / point sp to swapping stack space
15104		<1>	; 04/09/2015 - 01/09/2015
15105		<1>	; [u.usp] = esp
15106	00004375 68[9C400000]	<1>	push sysret ; ***
15107	0000437A 8925[7C740000]	<1>	mov [u.usp], esp ; points to 'sysret' address (***)
15108		<1>	; (for child process)
15109	00004380 31C0	<1>	xor eax, eax
15110	00004382 66A3[B0740000]	<1>	mov [u.ttyp], ax ; 0
15111		<1>	;
15112	00004388 E885100000	<1>	call wswap ; Retro UNIX 8086 v1 modification !
15113		<1>	;jsr r0,wswap / put child process out on drum
15114		<1>	;jsr r0,unpack / unpack user stack
15115		<1>	;mov u.usp,sp / restore user stack pointer
15116		<1>	; tst (sp)+ / bump stack pointer
15117		<1>	; Retro UNIX 386 v1 modification !
15118	0000438D 58	<1>	pop eax ; ***
15119	0000438E 66D1E3	<1>	shl bx, 1
15120	00004391 8B83[12720000]	<1>	mov eax, [ebx+p.upage-4] ; UPAGE address ; 14/05/2015
15121	00004397 E89F100000	<1>	call rswap ; restore parent process 'u' structure,
15122		<1>	; registers and return address (for IRET)
15123		<1>	;movb (sp)+,u.uno / put parent process number in u.uno
15124	0000439C 0FB705[6E740000]	<1>	movzx eax, word [mpid]
15125	000043A3 A3[80740000]	<1>	mov [u.r0], eax
15126		<1>	; mov mpid,*u.r0 / put child process name on stack
15127		<1>	; / where r0 was saved
15128		<1>	; add \$2,18.(sp) / add 2 to pc on stack; gives parent
15129		<1>	; / process return
15130		<1>	;xor ebx, ebx
15131	000043A8 31F6	<1>	xor esi, esi
15132		<1>	;clr r1
15133		<1>	sysfork_4: ; 1: / search u.fp list to find the files
15134		<1>	; / opened by the parent process
15135		<1>	; 01/09/2015
15136		<1>	;xor bh, bh
15137		<1>	;mov bl, [esi+u.fp]
15138	000043AA 8A86[86740000]	<1>	mov al, [esi+u.fp]
15139		<1>	; movb u.fp(r1),r2 / get an open file for this process
15140		<1>	;or bl, bl
15141	000043B0 08C0	<1>	or al, al
15142	000043B2 740D	<1>	jz short sysfork_5
15143		<1>	; beq 2f / file has not been opened by parent,
15144		<1>	; / so branch
15145	000043B4 B40A	<1>	mov ah, 10 ; Retro UNIX 386 v1 fsp structure size = 10 bytes
15146	000043B6 F6E4	<1>	mul ah
15147		<1>	;movzx ebx, ax
15148	000043B8 6689C3	<1>	mov bx, ax
15149		<1>	;shl bx, 3
15150		<1>	; asl r2 / multiply by 8
15151		<1>	; asl r2 / to get index into fsp table

15152		<1>	inc	byte [ebx+fsp-2]
15153	000043BB FE83[54720000]	<1>	incb	fsp-2(r2) / increment number of processes
15154		<1>		; / using file, because child will now be
15155		<1>		; / using this file
15156		<1>	sysfork_5:	; 2:
15157		<1>	inc	esi
15158	000043C1 46	<1>	inc	r1 / get next open file
15159		<1>	cmp	si, 10
15160	000043C2 6683FE0A	<1>	cmp	r1,\$10. / 10. files is the maximum number which
15161		<1>		; / can be opened
15162		<1>	jb	short sysfork_4
15163	000043C6 72E2	<1>	blt	1b / check next entry
15164		<1>	jmp	sysret
15165	000043C8 E9CFFCFFFF	<1>	br	sysret1
15166		<1>		
15167		<1>	sysread:	; < read from file >
15168		<1>		; 13/05/2015
15169		<1>		; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
15170		<1>		; 23/05/2013 (Retro UNIX 8086 v1)
15171		<1>		
15172		<1>		; 'sysread' is given a buffer to read into and the number of
15173		<1>		characters to be read. If finds the file from the file
15174		<1>		descriptor located in *u.r0 (r0). This file descriptor
15175		<1>		is returned from a successful open call (sysopen).
15176		<1>		The i-number of file is obtained via 'rw1' and the data
15177		<1>		is read into core via 'readi'.
15178		<1>		
15179		<1>		; Calling sequence:
15180		<1>		sysread; buffer; nchars
15181		<1>		Arguments:
15182		<1>		buffer - location of contiguous bytes where
15183		<1>		input will be placed.
15184		<1>		nchars - number of bytes or characters to be read.
15185		<1>		Inputs: *u.r0 - file descriptor (& arguments)
15186		<1>		Outputs: *u.r0 - number of bytes read.
15187		<1>		.....
15188		<1>		
15189		<1>		; Retro UNIX 8086 v1 modification:
15190		<1>		'sysread' system call has three arguments; so,
15191		<1>		* 1st argument, file descriptor is in BX register
15192		<1>		* 2nd argument, buffer address/offset in CX register
15193		<1>		* 3rd argument, number of bytes is in DX register
15194		<1>		
15195		<1>		AX register (will be restored via 'u.r0') will return
15196		<1>		to the user with number of bytes read.
15197		<1>		
15198		<1>	call	rw1
15199	000043CD E83D000000	<1>	jc	error ; 13/05/2015, ax < 1
15200	000043D2 0F82A4FCFFFF	<1>		; jsr r0,rw1 / get i-number of file to be read into r1
15201		<1>	test	ah, 80h
15202	000043D8 F6C480	<1>		; tst r1 / negative i-number?
15203		<1>	jnz	error
15204	000043DB 0F859BFCFFFF	<1>		; ble error1 / yes, error 1 to read
15205		<1>		; / it should be positive
15206		<1>	call	readi
15207	000043E1 E814150000	<1>		; jsr r0,readi / read data into core
15208		<1>	jmp	short rw0
15209	000043E6 EB18	<1>		; br 1f
15210		<1>	syswrite:	; < write to file >
15211		<1>		; 13/05/2015
15212		<1>		; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
15213		<1>		; 23/05/2013 (Retro UNIX 8086 v1)
15214		<1>		
15215		<1>		; 'syswrite' is given a buffer to write onto an output file
15216		<1>		and the number of characters to write. If finds the file
15217		<1>		from the file descriptor located in *u.r0 (r0). This file
15218		<1>		descriptor is returned from a successful open or create call
15219		<1>		(sysopen or syscreat). The i-number of file is obtained via
15220		<1>		'rw1' and buffer is written on the output file via 'write'.
15221		<1>		
15222		<1>		; Calling sequence:
15223		<1>		syswrite; buffer; nchars
15224		<1>		Arguments:
15225		<1>		buffer - location of contiguous bytes to be writtten.
15226		<1>		nchars - number of characters to be written.
15227		<1>		Inputs: *u.r0 - file descriptor (& arguments)
15228		<1>		Outputs: *u.r0 - number of bytes written.
15229		<1>		.....
15230		<1>		
15231		<1>		; Retro UNIX 8086 v1 modification:
15232		<1>		'syswrite' system call has three arguments; so,
15233		<1>		* 1st argument, file descriptor is in BX register
15234		<1>		* 2nd argument, buffer address/offset in CX register
15235		<1>		* 3rd argument, number of bytes is in DX register
15236		<1>		
15237		<1>		AX register (will be restored via 'u.r0') will return
15238		<1>		to the user with number of bytes written.
15239		<1>		
15240		<1>	call	rw1
15241	000043E8 E822000000	<1>	jc	error ; 13/05/2015, ax < 1
15242	000043ED 0F8289FCFFFF	<1>		; jsr r0,rw1 / get i-number in r1 of file to write
15243		<1>		

```

15244 000043F3 F6C480      <1>      test ah, 80h
15245                      <1>      ; tst r1 / positive i-number ?
15246 000043F6 744E        <1>      jz      short rw3 ; 13/05/2015
15247                      <1>      ;jz  error
15248                      <1>      ; bge error1 / yes, error 1
15249                      <1>      ; / negative i-number means write
15250 000043F8 66F7D8      <1>      neg  ax
15251                      <1>      ; neg r1 / make it positive
15252 000043FB E8F0160000   <1>      call writei
15253                      <1>      ; jsr r0,writei / write data
15254                      <1>      rw0: ; 1:
15255 00004400 A1[A8740000] <1>      mov  eax, [u.nread]
15256 00004405 A3[80740000] <1>      mov  [u.r0], eax
15257                      <1>      ; mov u.nread,*u.r0 / put no. of bytes transferred
15258                      <1>      ; / into (u.r0)
15259 0000440A E98DFCFFFF   <1>      jmp  sysret
15260                      <1>      ; br sysret1
15261                      <1>      rw1:
15262                      <1>      ; 14/05/2015
15263                      <1>      ; 13/05/2015
15264                      <1>      ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
15265                      <1>      ; 23/05/2013 - 24/05/2013 (Retro UNIX 8086 v1)
15266                      <1>      ; System call registers: bx, cx, dx (through 'sysenter')
15267                      <1>      ;
15268                      <1>      ;mov  [u.base], ecx      ; buffer address/offset
15269                      <1>      ;(in the user's virtual memory space)
15270                      <1>      ;mov  [u.count], edx
15271                      <1>      ; jsr r0,arg; u.base / get buffer pointer
15272                      <1>      ; jsr r0,arg; u.count / get no. of characters
15273                      <1>      ;;mov  eax, ebx ; file descriptor
15274                      <1>      ; mov *u.r0,r1 / put file descriptor
15275                      <1>      ; / (index to u.fp table) in r1
15276                      <1>      ; 13/05/2015
15277 0000440F C705[80740000]0000- <1>      mov  dword [u.r0], 0 ; r/w transfer count = 0 (reset)
15278 00004417 0000         <1>
15279                      <1>      ;
15280                      <1>      ;; call      getf
15281                      <1>      ; eBX = File descriptor
15282 00004419 E8D60A0000   <1>      call  getf1 ; calling point in 'getf' from 'rw1'
15283                      <1>      ; jsr r0,getf / get i-number of the file in r1
15284                      <1>      ; AX = I-number of the file ; negative i-number means write
15285                      <1>      ; 13/05/2015
15286 0000441E 6683F801     <1>      cmp  ax, 1
15287 00004422 7217         <1>      jnb  short rw2
15288                      <1>      ;
15289 00004424 890D[A0740000] <1>      mov  [u.base], ecx      ; buffer address/offset
15290                      <1>      ;(in the user's virtual memory space)
15291 0000442A 8915[A4740000] <1>      mov  [u.count], edx
15292                      <1>      ; 14/05/2015
15293 00004430 C705[CF740000]0000- <1>      mov  dword [u.error], 0 ; reset the last error code
15294 00004438 0000         <1>
15295 0000443A C3           <1>      retn
15296                      <1>      ; rts r0
15297                      <1>      rw2:
15298                      <1>      ; 13/05/2015
15299 0000443B C705[CF740000]0A00- <1>      mov  dword [u.error], ERR_FILE_NOT_OPEN ; file not open !
15300 00004443 0000         <1>
15301 00004445 C3           <1>      retn
15302                      <1>      rw3:
15303                      <1>      ; 13/05/2015
15304 00004446 C705[CF740000]0B00- <1>      mov  dword [u.error], ERR_FILE_ACCESS ; permission denied !
15305 0000444E 0000         <1>
15306 00004450 F9           <1>      stc
15307 00004451 C3           <1>      retn
15308                      <1>
15309                      <1>      sysopen: ;<open file>
15310                      <1>      ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15311                      <1>      ; 22/05/2013 - 27/05/2013 (Retro UNIX 8086 v1)
15312                      <1>      ;
15313                      <1>      ; 'sysopen' opens a file in following manner:
15314                      <1>      ; 1) The second argument in a sysopen says whether to
15315                      <1>      ; open the file ro read (0) or write (>0).
15316                      <1>      ; 2) I-node of the particular file is obtained via 'namei'.
15317                      <1>      ; 3) The file is opened by 'iopen'.
15318                      <1>      ; 4) Next housekeeping is performed on the fsp table
15319                      <1>      ; and the user's open file list - u.fp.
15320                      <1>      ; a) u.fp and fsp are scanned for the next available slot.
15321                      <1>      ; b) An entry for the file is created in the fsp table.
15322                      <1>      ; c) The number of this entry is put on u.fp list.
15323                      <1>      ; d) The file descriptor index to u.fp list is pointed
15324                      <1>      ; to by u.r0.
15325                      <1>      ;
15326                      <1>      ; Calling sequence:
15327                      <1>      ; sysopen; name; mode
15328                      <1>      ; Arguments:
15329                      <1>      ; name - file name or path name
15330                      <1>      ; mode - 0 to open for reading
15331                      <1>      ; 1 to open for writing
15332                      <1>      ; Inputs: (arguments)
15333                      <1>      ; Outputs: *u.r0 - index to u.fp list (the file descriptor)
15334                      <1>      ; is put into r0's location on the stack.
15335                      <1>      ; .....

```

```

15336 <1> ;
15337 <1> ; Retro UNIX 8086 v1 modification:
15338 <1> ; 'sysopen' system call has two arguments; so,
15339 <1> ; * 1st argument, name is pointed to by BX register
15340 <1> ; * 2nd argument, mode is in CX register
15341 <1> ;
15342 <1> ; AX register (will be restored via 'u.r0') will return
15343 <1> ; to the user with the file descriptor/number
15344 <1> ; (index to u.fp list).
15345 <1> ;
15346 <1> ;call arg2
15347 <1> ; * name - 'u.namep' points to address of file/path name
15348 <1> ; in the user's program segment ('u.segmt')
15349 <1> ; with offset in BX register (as sysopen argument 1).
15350 <1> ; * mode - sysopen argument 2 is in CX register
15351 <1> ; which is on top of stack.
15352 <1> ;
15353 <1> ; jsr r0,arg2 / get sys args into u.namep and on stack
15354 <1> ;
15355 <1> ; system call registers: ebx, ecx (through 'sysenter')
15356 <1>
15357 00004452 891D[98740000] <1> mov [u.namep], ebx
15358 00004458 6651 <1> push cx
15359 0000445A E8CC0A0000 <1> call namei
15360 <1> ; jsr r0,namei / i-number of file in r1
15361 <1> ;and ax, ax
15362 <1> ;jz error ; File not found
15363 0000445F 723B <1> jc short fnotfound ; 14/05/2015
15364 <1> ;jc error ; 27/05/2013
15365 <1> ; br error2 / file not found
15366 00004461 665A <1> pop dx ; mode
15367 00004463 6652 <1> push dx
15368 <1> ;or dx, dx
15369 00004465 08D2 <1> or dl, dl
15370 <1> ; tst (sp) / is mode = 0 (2nd arg of call;
15371 <1> ; / 0 means, open for read)
15372 00004467 7403 <1> jz short sysopen_0
15373 <1> ; beq 1f / yes, leave i-number positive
15374 00004469 66F7D8 <1> neg ax
15375 <1> ; neg r1 / open for writing so make i-number negative
15376 <1> sysopen_0: ;1:
15377 0000446C E8281A0000 <1> call iopen
15378 <1> ;jsr r0,iopen / open file whose i-number is in r1
15379 00004471 665A <1> pop dx
15380 <1> ;and dx, dx
15381 00004473 20D2 <1> and dl, dl
15382 <1> ; tst (sp)+ / pop the stack and test the mode
15383 00004475 7403 <1> jz short sysopen_2
15384 <1> ; beq op1 / is open for read op1
15385 <1> sysopen_1: ;op0:
15386 00004477 66F7D8 <1> neg ax
15387 <1> ; neg r1
15388 <1> ;/ make i-number positive if open for writing [???]
15389 <1> ;; NOTE: iopen always make i-number positive.
15390 <1> ;; Here i-number becomes negative again. [22/05/2013]
15391 <1> sysopen_2: ;op1:
15392 0000447A 31F6 <1> xor esi, esi
15393 <1> ; clr r2 / clear registers
15394 0000447C 31DB <1> xor ebx, ebx
15395 <1> ; clr r3
15396 <1> sysopen_3: ;1: / scan the list of entries in fsp table
15397 0000447E 389E[86740000] <1> cmp [esi+u.fp], bl ; 0
15398 <1> ; tstb u.fp(r2) / test the entry in the u.fp list
15399 00004484 7625 <1> jna short sysopen_4
15400 <1> ; beq 1f / if byte in list is 0 branch
15401 00004486 46 <1> inc esi
15402 <1> ; inc r2 / bump r2 so next byte can be checked
15403 00004487 6683FE0A <1> cmp si, 10
15404 <1> ; cmp r2,$10. / reached end of list?
15405 0000448B 72F1 <1> jnb short sysopen_3
15406 <1> ; blt 1b / no, go back
15407 <1> toomanyf:
15408 <1> ; 14/05/2015
15409 0000448D C705[CF740000]0D00- <1> mov dword [u.error], ERR_TOO_MANY_FILES ; too many open files !
15410 00004495 0000 <1>
15411 00004497 E9E0FBFFFF <1> jmp error
15412 <1> ; br error2 / yes, error (no files open)
15413 <1> fnotfound:
15414 <1> ; 14/05/2015
15415 0000449C C705[CF740000]0C00- <1> mov dword [u.error], ERR_FILE_NOT_FOUND ; file not found !
15416 000044A4 0000 <1>
15417 000044A6 E9D1FBFFFF <1> jmp error
15418 <1>
15419 <1> sysopen_4: ; 1:
15420 000044AB 6683BB[56720000]00 <1> cmp word [ebx+fsp], 0
15421 <1> ; tst fsp(r3) / scan fsp entries
15422 000044B3 7610 <1> jna short sysopen_5
15423 <1> ; beq 1f / if 0 branch
15424 <1> ; 14/05/2015 - Retro UNIX 386 v1 modification !
15425 000044B5 6683C30A <1> add bx, 10 ; fsp structure size = 10 bytes/entry
15426 <1> ; add $8.,r3 / add 8 to r3
15427 <1> ; / to bump it to next entry mfsp table

```



```

15428 000044B9 6681FBF401      <1>      cmp      bx, nfiles*10
15429                          <1>      ; cmp r3,$[nfiles*8.] / done scanning
15430 000044BE 72EB             <1>      jnb      short sysopen_4
15431                          <1>      ; blt 1b / no, back
15432 000044C0 E9B7FBFFFF       <1>      jmp      error
15433                          <1>      ; br error2 / yes, error
15434                          <1>      sysopen_5: ; 1: / r2 has index to u.fp list; r3, has index to fsp table
15435 000044C5 668983[56720000] <1>      mov      [ebx+fsp], ax
15436                          <1>      ; mov r1,fsp(r3) / put i-number of open file
15437                          <1>      ; / into next available entry in fsp table,
15438 000044CC 668B3D[66740000] <1>      mov      di, [cdev] ; word ? byte ?
15439 000044D3 6689BB[58720000] <1>      mov      [ebx+fsp+2], di ; device number
15440                          <1>      ; mov cdev,fsp+2(r3) / put # of device in next word
15441 000044DA 31FF             <1>      xor      edi, edi
15442 000044DC 89BB[5A720000] <1>      mov      [ebx+fsp+4], edi ; offset pointer (0)
15443                          <1>      ; clr fsp+4(r3)
15444 000044E2 6689BB[5E720000] <1>      mov      [ebx+fsp+8], di ; open count (0), deleted flag (0)
15445                          <1>      ; clr fsp+6(r3) / clear the next two words
15446 000044E9 89D8             <1>      mov      eax, ebx
15447 000044EB B30A             <1>      mov      bl, 10
15448 000044ED F6F3             <1>      div      bl
15449                          <1>      ; asr r3
15450                          <1>      ; asr r3 / divide by 8
15451                          <1>      ; asr r3 ; / to get number of the fsp entry-1
15452 000044EF FEC0             <1>      inc      al
15453                          <1>      ; inc r3 / add 1 to get fsp entry number
15454 000044F1 8886[86740000] <1>      mov      [esi+u.fp], al
15455                          <1>      ; movb r3,u.fp(r2) / move entry number into
15456                          <1>      ; / next available slot in u.fp list
15457 000044F7 8935[80740000] <1>      mov      [u.r0], esi
15458                          <1>      ; mov r2,*u.r0 / move index to u.fp list
15459                          <1>      ; / into r0 loc on stack
15460 000044FD E99AFBFFFF       <1>      jmp      sysret
15461                          <1>      ; br sysret2
15462                          <1>
15463                          <1>      ;
15464                          <1>      ; 'fsp' table (10 bytes/entry)
15465                          <1>      ; bit 15 bit 0
15466                          <1>      ; ---|-----
15467                          <1>      ; r/w| i-number of open file
15468                          <1>      ; ---|-----
15469                          <1>      ; device number
15470                          <1>      ; -----
15471                          <1>      ; offset pointer, r/w pointer to file (bit 0-15)
15472                          <1>      ; -----
15473                          <1>      ; offset pointer, r/w pointer to file (bit 16-31)
15474                          <1>      ; -----|-----
15475                          <1>      ; flag that says file | number of processes
15476                          <1>      ; has been deleted | that have file open
15477                          <1>      ; -----|-----
15478                          <1>      ;
15479                          <1>
15480                          <1>      syscreat: ; < create file >
15481                          <1>      ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15482                          <1>      ; 27/05/2013 (Retro UNIX 8086 v1)
15483                          <1>      ;
15484                          <1>      ; 'syscreat' called with two arguments; name and mode.
15485                          <1>      ; u.namep points to name of the file and mode is put
15486                          <1>      ; on the stack. 'namei' is called to get i-number of the file.
15487                          <1>      ; If the file already exists, it's mode and owner remain
15488                          <1>      ; unchanged, but it is truncated to zero length. If the file
15489                          <1>      ; did not exist, an i-node is created with the new mode via
15490                          <1>      ; 'maknod' whether or not the file already existed, it is
15491                          <1>      ; open for writing. The fsp table is then searched for a free
15492                          <1>      ; entry. When a free entry is found, proper data is placed
15493                          <1>      ; in it and the number of this entry is put in the u.fp list.
15494                          <1>      ; The index to the u.fp (also know as the file descriptor)
15495                          <1>      ; is put in the user's r0.
15496                          <1>      ;
15497                          <1>      ; Calling sequence:
15498                          <1>      ; syscreate; name; mode
15499                          <1>      ; Arguments:
15500                          <1>      ; name - name of the file to be created
15501                          <1>      ; mode - mode of the file to be created
15502                          <1>      ; Inputs: (arguments)
15503                          <1>      ; Outputs: *u.r0 - index to u.fp list
15504                          <1>      ; (the file descriptor of new file)
15505                          <1>      ; .....
15506                          <1>      ;
15507                          <1>      ; Retro UNIX 8086 v1 modification:
15508                          <1>      ; 'syscreate' system call has two arguments; so,
15509                          <1>      ; * 1st argument, name is pointed to by BX register
15510                          <1>      ; * 2nd argument, mode is in CX register
15511                          <1>      ;
15512                          <1>      ; AX register (will be restored via 'u.r0') will return
15513                          <1>      ; to the user with the file descriptor/number
15514                          <1>      ; (index to u.fp list).
15515                          <1>      ;
15516                          <1>      ; call arg2
15517                          <1>      ; * name - 'u.namep' points to address of file/path name
15518                          <1>      ; in the user's program segment ('u.segmt')
15519                          <1>      ; with offset in BX register (as sysopen argument 1).

```

```

15520      <1> ; * mode - sysopen argument 2 is in CX register
15521      <1> ;         which is on top of stack.
15522      <1> ;
15523      <1>         ; jsr r0,arg2 / put file name in u.namep put mode
15524      <1>         ; / on stack
15525 00004502 891D[98740000] <1> mov  [u.namep], ebx ; file name address
15526 00004508 6651          <1> push cx ; mode
15527 0000450A E81C0A0000    <1> call namei
15528      <1>         ; jsr r0,namei / get the i-number
15529      <1>         ; and ax, ax
15530      <1>         ; jz short syscreat_1
15531 0000450F 7214          <1> jc  short syscreat_1
15532      <1>         ; br 2f / if file doesn't exist 2f
15533 00004511 66F7D8        <1> neg  ax
15534      <1>         ; neg r1 / if file already exists make i-number
15535      <1>         ; / negative (open for writing)
15536 00004514 E880190000    <1> call iopen
15537      <1>         ; jsr r0,iopen /
15538 00004519 E831130000    <1> call itrunc
15539      <1>         ; jsr r0,itrunc / truncate to 0 length
15540 0000451E 6659          <1> pop  cx ; pop mode (did not exist in original Unix v1 !?)
15541 00004520 E952FFFFFF    <1> jmp   sysopen_1
15542      <1>         ; br op0
15543      <1> syscreat_1: ; 2: / file doesn't exist
15544 00004525 6658          <1> pop  ax
15545      <1>         ; mov (sp)+,r1 / put the mode in r1
15546 00004527 30E4          <1> xor  ah, ah
15547      <1>         ; bic $!377,r1 / clear upper byte
15548 00004529 E8D00C0000    <1> call maknod
15549      <1>         ; jsr r0,maknod / make an i-node for this file
15550 0000452E 66A1[B2740000] <1> mov  ax, [u.dirbuf]
15551      <1>         ; mov u.dirbuf,r1 / put i-number
15552      <1>         ; / for this new file in r1
15553 00004534 E93EFFFFFF    <1> jmp   sysopen_1
15554      <1>         ; br op0 / open the file
15555      <1>
15556      <1> sysmkdir: ; < make directory >
15557      <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15558      <1> ; 27/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
15559      <1> ;
15560      <1> ; 'sysmkdir' creates an empty directory whose name is
15561      <1> ; pointed to by arg 1. The mode of the directory is arg 2.
15562      <1> ; The special entries '.' and '..' are not present.
15563      <1> ; Errors are indicated if the directory already exists or
15564      <1> ; user is not the super user.
15565      <1> ;
15566      <1> ; Calling sequence:
15567      <1> ;     sysmkdir; name; mode
15568      <1> ; Arguments:
15569      <1> ;     name - points to the name of the directory
15570      <1> ;     mode - mode of the directory
15571      <1> ; Inputs: (arguments)
15572      <1> ; Outputs: -
15573      <1> ;     (sets 'directory' flag to 1;
15574      <1> ;     'set user id on execution' and 'executable' flags to 0)
15575      <1> ; .....
15576      <1> ;
15577      <1> ; Retro UNIX 8086 v1 modification:
15578      <1> ;     'sysmkdir' system call has two arguments; so,
15579      <1> ;     * 1st argument, name is pointed to by BX register
15580      <1> ;     * 2nd argument, mode is in CX register
15581      <1> ;
15582      <1>
15583      <1> ; / make a directory
15584      <1>
15585      <1> ; call arg2
15586      <1> ; * name - 'u.namep' points to address of file/path name
15587      <1> ;         in the user's program segment ('u.segmt')
15588      <1> ;         with offset in BX register (as sysopen argument 1).
15589      <1> ; * mode - sysopen argument 2 is in CX register
15590      <1> ;         which is on top of stack.
15591      <1>
15592      <1>         ; jsr r0,arg2 / put file name in u.namep put mode
15593      <1>         ; / on stack
15594 00004539 891D[98740000] <1> mov  [u.namep], ebx
15595 0000453F 6651          <1> push cx ; mode
15596 00004541 E8E5090000    <1> call namei
15597      <1>         ; jsr r0,namei / get the i-number
15598      <1>         ; br .+4 / if file not found branch around error
15599      <1>         ; xor ax, ax
15600      <1> ; jnz error
15601 00004546 731C          <1> jnc  short dir_exists ; 14/05/2015
15602      <1> ; jnc error
15603      <1>         ; br error2 / directory already exists (error)
15604 00004548 803D[C6740000]00 <1> cmp  byte [u.uid], 0 ; 02/08/2013
15605      <1>         ; tstb u.uid / is user the super user
15606 0000454F 7622          <1> jna  short dir_access_err ; 14/05/2015
15607      <1> ; jna error
15608      <1>         ; bne error2 / no, not allowed
15609 00004551 6658          <1> pop  ax
15610      <1>         ; mov (sp)+,r1 / put the mode in r1
15611 00004553 6683E0CF      <1> and  ax, 0FFCFh ; 111111111001111b

```

```

15612                                <1>          ;bic $!317,r1 / all but su and ex
15613                                <1> ;or   ax , 4000h ; 1011111111111111b
15614 00004557 80CC40                <1> or   ah, 40h ; Set bit 14 to 1
15615                                <1>          ;bis $40000,r1 / directory flag
15616 0000455A E89F0C0000            <1> call  maknod
15617                                <1>          ;jsr r0,maknod / make the i-node for the directory
15618 0000455F E938FBFFFF            <1> jmp   sysret
15619                                <1>          ;br sysret2 /
15620                                <1> dir_exists:
15621                                <1> ; 14/05/2015
15622 00004564 C705[CF740000]0E00-   <1> mov   dword [u.error], ERR_DIR_EXISTS ; dir. already exists !
15623 0000456C 0000                  <1>
15624 0000456E E909FBFFFF            <1> jmp   error
15625                                <1> dir_access_err:
15626                                <1> ; 14/05/2015
15627 00004573 C705[CF740000]0B00-   <1> mov   dword [u.error], ERR_DIR_ACCESS ; permission denied !
15628 0000457B 0000                  <1>
15629 0000457D E9FAFAFFFF            <1> jmp   error
15630                                <1>
15631                                <1> sysclose: ;<close file>
15632                                <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15633                                <1> ; 22/05/2013 - 26/05/2013 (Retro UNIX 8086 v1)
15634                                <1> ;
15635                                <1> ; 'sysclose', given a file descriptor in 'u.r0', closes the
15636                                <1> ; associated file. The file descriptor (index to 'u.fp' list)
15637                                <1> ; is put in r1 and 'fclose' is called.
15638                                <1> ;
15639                                <1> ; Calling sequence:
15640                                <1> ;     sysclose
15641                                <1> ; Arguments:
15642                                <1> ;     -
15643                                <1> ; Inputs: *u.r0 - file descriptor
15644                                <1> ; Outputs: -
15645                                <1> ; .....
15646                                <1> ;
15647                                <1> ; Retro UNIX 8086 v1 modification:
15648                                <1> ;     The user/application program puts file descriptor
15649                                <1> ;     in BX register as 'sysclose' system call argument.
15650                                <1> ;     (argument transfer method 1)
15651                                <1>
15652                                <1> ; / close the file
15653                                <1>
15654 00004582 89D8                    <1> mov   eax, ebx
15655 00004584 E820090000            <1> call  fclose
15656                                <1>          ; mov *u.r0,r1 / move index to u.fp list into r1
15657                                <1>          ; jsr r0,fclose / close the file
15658                                <1>          ; br error2 / unknown file descriptor
15659                                <1>          ; br sysret2
15660                                <1> ; 14/05/2015
15661 00004589 0F830DFBFFFF            <1> jnc   sysret
15662 0000458F C705[CF740000]0A00-   <1> mov   dword [u.error], ERR_FILE_NOT_OPEN ; file not open !
15663 00004597 0000                  <1>
15664 00004599 E9DEFAFFFF            <1> jmp   error
15665                                <1>
15666                                <1> sysemt:
15667                                <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15668                                <1> ; 10/12/2013 - 20/04/2014 (Retro UNIX 8086 v1)
15669                                <1> ;
15670                                <1> ; Retro UNIX 8086 v1 modification:
15671                                <1> ;     'Enable Multi Tasking' system call instead
15672                                <1> ;     of 'Emulator Trap' in original UNIX v1 for PDP-11.
15673                                <1> ;
15674                                <1> ; Retro UNIX 8086 v1 feature only!
15675                                <1> ;     Using purpose: Kernel will start without time-out
15676                                <1> ;     (internal clock/timer) functionality.
15677                                <1> ;     Then etc/init will enable clock/timer for
15678                                <1> ;     multi tasking. (Then it will not be disabled again
15679                                <1> ;     except hardware reset/restart.)
15680                                <1> ;
15681                                <1>
15682 0000459E 803D[C6740000]00        <1> cmp   byte [u.uid], 0 ; root ?
15683                                <1> ;ja   error
15684 000045A5 0F877AFBFFFF            <1> ja    badsys ; 14/05/2015
15685                                <1> emt_0:
15686 000045AB FA                      <1> cli
15687 000045AC 21DB                    <1> and   ebx, ebx
15688 000045AE 7410                    <1> jz    short emt_2
15689                                <1> ; Enable multi tasking -time sharing-
15690 000045B0 B8[B1540000]            <1> mov   eax, clock
15691                                <1> emt_1:
15692 000045B5 A3[1B080000]            <1> mov   [x_timer], eax
15693 000045BA FB                      <1> sti
15694 000045BB E9DCFAFFFF            <1> jmp   sysret
15695                                <1> emt_2:
15696                                <1> ; Disable multi tasking -time sharing-
15697 000045C0 B8[23080000]            <1> mov   eax, u_timer
15698 000045C5 EBEE                    <1> jmp   short emt_1
15699                                <1>
15700                                <1> ; Original UNIX v1 'sysemt' routine
15701                                <1> ;sysemt:
15702                                <1> ;
15703                                <1> ;jsr   r0,arg; 30 / put the argument of the sysemt call

```

```

15704                                <1>                ; / in loc 30
15705                                <1>                ;cmp      30,$score / was the argument a lower address
15706                                <1>                ; / than core
15707                                <1>                ;blo      1f / yes, rtssym
15708                                <1>                ;cmp      30,$ecore / no, was it higher than "core"
15709                                <1>                ; / and less than "ecore"
15710                                <1>                ;blo      2f / yes, sysret2
15711                                <1> ;1:
15712                                <1>                ;mov      $rtssym,30
15713                                <1> ;2:
15714                                <1>                ;br       sysret2
15715                                <1>
15716                                <1> sysilgins:
15717                                <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15718                                <1> ; 03/06/2013
15719                                <1> ; Retro UNIX 8086 v1 modification:
15720                                <1> ;      not a valid system call ! (not in use)
15721                                <1> ;
15722 000045C7 E959FBFFFF             <1> jmp      badsys
15723                                <1> ;jmp      error
15724                                <1> ;;jmp      sysret
15725                                <1>
15726                                <1> ; Original UNIX v1 'sysemt' routine
15727                                <1> ;sysilgins: / calculate proper illegal instruction trap address
15728                                <1>                ;jsr      r0,arg; 10 / take address from sysilgins call
15729                                <1>                ;/ put it in loc 8.,
15730                                <1>                ;cmp      10,$score / making it the illegal instruction
15731                                <1>                ; / trap address
15732                                <1>                ;blo      1f / is the address a user core address?
15733                                <1>                ; / yes, go to 2f
15734                                <1>                ;cmp      10,$ecore
15735                                <1>                ;blo      2f
15736                                <1> ;1:
15737                                <1>                ;mov      $fpsym,10 / no, make 'fpsum' the illegal
15738                                <1>                ; / instruction trap address for the system
15739                                <1> ;2:
15740                                <1>                ;br       sysret2 / return to the caller via 'sysret'
15741                                <1>
15742                                <1> sysmdate: ; < change the modification time of a file >
15743                                <1> ; 16/05/2015 (Retro UNIX 386 v1 - Beginning)
15744                                <1> ; 03/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
15745                                <1> ;
15746                                <1> ; 'sysmdate' is given a file name. It gets inode of this
15747                                <1> ; file into core. The user is checked if he is the owner
15748                                <1> ; or super user. If he is neither an error occurs.
15749                                <1> ; 'setimod' is then called to set the i-node modification
15750                                <1> ; byte and the modification time, but the modification time
15751                                <1> ; is overwritten by whatever get put on the stack during
15752                                <1> ; a 'systime' system call. This calls are restricted to
15753                                <1> ; the super user.
15754                                <1> ;
15755                                <1> ; Calling sequence:
15756                                <1> ;      sysmdate; name
15757                                <1> ; Arguments:
15758                                <1> ;      name - points to the name of file
15759                                <1> ; Inputs: (arguments)
15760                                <1> ; Outputs: -
15761                                <1> ; .....
15762                                <1> ;
15763                                <1> ; Retro UNIX 8086 v1 modification:
15764                                <1> ;      The user/application program puts address
15765                                <1> ;      of the file name in BX register
15766                                <1> ;      as 'sysmdate' system call argument.
15767                                <1> ;
15768                                <1> ; / change the modification time of a file
15769                                <1>                ; jsr r0,arg; u.namep / point u.namep to the file name
15770 000045CC 891D[98740000]         <1>                mov     [u.namep], ebx
15771 000045D2 E854090000             <1> call     namei
15772                                <1>                ; jsr r0,namei / get its i-number
15773 000045D7 0F82BFFFEFFFFF         <1>                jc      fnotfound ; file not found !
15774                                <1>                ;jc      error
15775                                <1>                ; br error2 / no, such file
15776 000045DD E838110000             <1> call     iget
15777                                <1>                ; jsr r0,iget / get i-node into core
15778 000045E2 A0[C6740000]           <1> mov      al, [u.uid]
15779 000045E7 3A05[59710000]         <1> cmp      al, [i.uid]
15780                                <1>                ; cmpb u.uid,i.uid / is user same as owner
15781 000045ED 7413                    <1> je       short mdate_1
15782                                <1>                ; beq 1f / yes
15783 000045EF 20C0                    <1> and      al, al
15784                                <1>                ; tstb u.uid / no, is user the super user
15785                                <1>                ;jnz     error
15786                                <1>                ; bne error2 / no, error
15787 000045F1 740F                    <1> jz       short mdate_1
15788 000045F3 C705[CF740000]0B00-    <1> mov      dword [u.error], ERR_FILE_ACCESS ; permission denied !
15789 000045FB 0000                    <1>
15790 000045FD E97AFAFFFF             <1> jmp      error
15791                                <1> mdate_1: ;1:
15792 00004602 E826120000             <1> call     setimod
15793                                <1>                ; jsr r0,setimod / fill in modification data,
15794                                <1>                ; / time etc.
15795 00004607 BE[F0700000]           <1> mov      esi, p_time

```

```

15796 0000460C BF[70710000]    <1>  mov    edi, i.mtim
15797 00004611 A5              <1>  movsd
15798                          <1>      ; mov 4(sp),i.mtim / move present time to
15799                          <1>      ; mov 2(sp),i.mtim+2 / modification time
15800 00004612 E985FAFFFF      <1>      jmp    sysret
15801                          <1>      ; br sysret2
15802                          <1>
15803                          <1> sysstty: ; < set tty status and mode >
15804                          <1> ; 17/11/2015
15805                          <1> ; 12/11/2015
15806                          <1> ; 29/10/2015
15807                          <1> ; 17/10/2015
15808                          <1> ; 13/10/2015
15809                          <1> ; 29/06/2015
15810                          <1> ; 27/06/2015 (Retro UNIX 386 v1 - Beginning)
15811                          <1> ; 02/06/2013 - 12/07/2014 (Retro UNIX 8086 v1)
15812                          <1> ;
15813                          <1> ; 'sysstty' sets the status and mode of the typewriter
15814                          <1> ; whose file descriptor is in (u.r0).
15815                          <1> ;
15816                          <1> ; Calling sequence:
15817                          <1> ; sysstty; arg
15818                          <1> ; Arguments:
15819                          <1> ; arg - address of 3 consequitive words that contain
15820                          <1> ; the source of status data
15821                          <1> ; Inputs: ((*u.r0 - file descriptor & argument))
15822                          <1> ; Outputs: ((status in address which is pointed to by arg))
15823                          <1> ; .....
15824                          <1> ;
15825                          <1> ; Retro UNIX 8086 v1 modification:
15826                          <1> ; 'sysstty' system call will set the tty
15827                          <1> ; (clear keyboard buffer and set cursor position)
15828                          <1> ; in following manner:
15829                          <1> ; NOTE: All of tty setting functions are here (16/01/2014)
15830                          <1> ;
15831                          <1> ; Inputs:
15832                          <1> ; BX = 0 --> means
15833                          <1> ; If CL = FFh
15834                          <1> ; set cursor position for console tty, only
15835                          <1> ; CH will be ignored (char. will not be written)
15836                          <1> ; If CH = 0 (CL < FFh)
15837                          <1> ; set console tty for (current) process
15838                          <1> ; CL = tty number (0 to 9)
15839                          <1> ; (If CH = 0, character will not be written)
15840                          <1> ; If CH > 0 (CL < FFh)
15841                          <1> ; CL = tty number (0 to 9)
15842                          <1> ; CH = character will be written
15843                          <1> ; at requested cursor position (in DX)
15844                          <1> ; DX = cursor position for tty number 0 to 7.
15845                          <1> ; (only tty number 0 to 7)
15846                          <1> ; DL = communication parameters (for serial ports)
15847                          <1> ; (only for COM1 and COM2 serial ports)
15848                          <1> ; DH < 0FFh -> DL is valid, initialize serial port
15849                          <1> ; or set cursor position
15850                          <1> ; DH = 0FFh -> DL is not valid
15851                          <1> ; do not set serial port parameters
15852                          <1> ; or do not set cursor position
15853                          <1> ;
15854                          <1> ; BX > 0 --> points to name of tty
15855                          <1> ; CH > 0 -->
15856                          <1> ; CH = character will be written in current
15857                          <1> ; cursor position (for tty number from 0 to 7)
15858                          <1> ; or character will be sent to serial port
15859                          <1> ; (for tty number 8 or 9)
15860                          <1> ; CL = color of the character if tty number < 8.
15861                          <1> ; CH = 0 --> Do not write a character,
15862                          <1> ; set mode (tty 8 to 9) or
15863                          <1> ; set current cursor positions (tty 0 to 7) only.
15864                          <1> ; DX = cursor position for tty number 0 to 7.
15865                          <1> ; DH = FFh --> Do not set cursor pos (or comm. params.)
15866                          <1> ; (DL is not valid)
15867                          <1> ; DL = communication parameters
15868                          <1> ; for tty number 8 or 9 (COM1 or COM2).
15869                          <1> ; Outputs:
15870                          <1> ; cf = 0 -> OK
15871                          <1> ; AL = tty number (0 to 9)
15872                          <1> ; AH = line status if tty number is 8 or 9
15873                          <1> ; AH = process number (of the caller)
15874                          <1> ; cf = 1 means error (requested tty is not ready)
15875                          <1> ; AH = FFh if the tty is locked
15876                          <1> ; (owned by another process)
15877                          <1> ; = process number (of the caller)
15878                          <1> ; (if < FFh and tty number < 8)
15879                          <1> ; AL = tty number (0FFh if it does not exist)
15880                          <1> ; AH = line status if tty number is 8 or 9
15881                          <1> ; NOTE: Video page will be cleared if cf = 0.
15882                          <1> ;
15883                          <1> ; 27/06/2015 (32 bit modifications)
15884                          <1> ; 14/01/2014
15885 00004617 31C0            <1>  xor    eax, eax
15886 00004619 6648            <1>  dec    ax ; 17/10/2015
15887 0000461B A3[80740000]    <1>  mov    [u.r0], eax ; 0FFFFh

```

15888	00004620	21DB	<1>	and	ebx, ebx
15889	00004622	0F85CB000000	<1>	jnz	sysstty_6
15890			<1>	; set console tty	
15891			<1>	; 29/10/2015	
15892			<1>	; 17/01/2014	
15893	00004628	80F909	<1>	cmp	cl, 9
15894	0000462B	7613	<1>	jna	short sysstty_0
15895			<1>	; 17/11/2015	
15896	0000462D	80F9FF	<1>	cmp	cl, 0FFh
15897	00004630	7202	<1>	jb	short sysstty_13
15898	00004632	88CD	<1>	mov	ch, cl ; force CH value to FFh
15899			<1>	sysstty_13:	
15900	00004634	8A1D[C9740000]	<1>	mov	bl, [u.uno] ; process number
15901	0000463A	8A8B[D5710000]	<1>	mov	cl, [ebx+p.ttyc-1] ; current/console tty
15902			<1>	sysstty_0:	
15903			<1>	; 29/06/2015	
15904	00004640	6652	<1>	push	dx
15905	00004642	6651	<1>	push	cx
15906	00004644	30D2	<1>	xor	dl, dl ; sysstty call sign
15907	00004646	88C8	<1>	mov	al, cl
15908	00004648	A2[80740000]	<1>	mov	[u.r0], al ; tty number (0 to 9)
15909	0000464D	E8E1180000	<1>	call	ottyp
15910	00004652	6659	<1>	pop	cx
15911	00004654	665A	<1>	pop	dx
15912			<1>	;	
15913	00004656	7257	<1>	jc	short sysstty_pd_err
15914			<1>	;	
15915	00004658	80F908	<1>	cmp	cl, 8
15916	0000465B	7222	<1>	jb	short sysstty_2
15917			<1>	;	
15918	0000465D	80FEFF	<1>	cmp	dh, 0FFh
15919	00004660	741D	<1>	je	short sysstty_2
15920			<1>	; set communication parameters for serial ports	
15921			<1>	; 29/10/2015	
15922	00004662	88D4	<1>	mov	ah, dl ; communication parameters
15923			<1>	; ah = 0E3h = 11100011b = 115200 baud,	
15924			<1>	; THRE int + RDA int	
15925			<1>	; ah = 23h = 00100011b = 9600 baud,	
15926			<1>	; THRE int + RDA int	
15927	00004664	28C0	<1>	sub	al, al ; 0
15928			<1>	; 12/07/2014	
15929	00004666	80F909	<1>	cmp	cl, 9
15930	00004669	7202	<1>	jb	short sysstty_1
15931	0000466B	FEC0	<1>	inc	al
15932			<1>	sysstty_1:	
15933	0000466D	6651	<1>	push	cx
15934			<1>	; 29/06/2015	
15935	0000466F	E8B5F4FFFF	<1>	call	sp_setp ; Set serial port communication parameters
15936	00004674	66890D[81740000]	<1>	mov	[u.r0+1], cx ; Line status (ah)
15937			<1>	; Modem status (EAX bits 16 to 23)	
15938	0000467B	6659	<1>	pop	cx
15939	0000467D	7265	<1>	jc	short sysstty_tmout_err ; 29/10/2015
15940			<1>	sysstty_2:	
15941			<1>	; 17/01/2014	
15942	0000467F	20ED	<1>	and	ch, ch ; set cursor position
15943			<1>	; or comm. parameters ONLY	
15944	00004681	750D	<1>	jnz	short sysstty_3
15945	00004683	0FB61D[C9740000]	<1>	movzx	ebx, byte [u.uno] ; process number
15946	0000468A	888B[D5710000]	<1>	mov	[ebx+p.ttyc-1], cl ; console tty
15947			<1>	sysstty_3:	
15948			<1>	; 16/01/2014	
15949	00004690	88E8	<1>	mov	al, ch ; character ; 0 to FFh
15950			<1>	; 17/11/2015	
15951	00004692	B507	<1>	mov	ch, 7 ; Default color (light gray)
15952	00004694	38E9	<1>	cmp	cl, ch ; 7 (tty number)
15953	00004696	0F86C5000000	<1>	jna	sysstty_9
15954			<1>	sysstty_12:	
15955			<1>	;; BX = 0, CL = 8 or CL = 9	
15956			<1>	; (Set specified serial port as console tty port)	
15957			<1>	; CH = character to be written	
15958			<1>	; 15/04/2014	
15959			<1>	; CH = 0 --> initialization only	
15960			<1>	; AL = character	
15961			<1>	; 26/06/2014	
15962	0000469C	880D[CE740000]	<1>	mov	[u.tty], cl
15963			<1>	; 12/07/2014	
15964	000046A2	88CC	<1>	mov	ah, cl ; tty number (8 or 9)
15965	000046A4	20C0	<1>	and	al, al
15966	000046A6	7416	<1>	jz	short sysstty_4 ; al = ch = 0
15967			<1>	; 04/07/2014	
15968	000046A8	E8941E0000	<1>	call	sndc
15969			<1>	; 12/07/2014	
15970	000046AD	EB1B	<1>	jmp	short sysstty_5
15971			<1>	sysstty_pd_err: ; 29/06/2015	
15972			<1>	; 'permission denied !' error	
15973	000046AF	C705[CF740000]0B00-	<1>	mov	dword [u.error], ERR_NOT_OWNER
15974	000046B7	0000	<1>		
15975	000046B9	E9BEF9FFFF	<1>	jmp	error
15976			<1>	sysstty_4:	
15977			<1>	; 12/07/2014	
15978			<1>	;xchg	ah, al ; al = 0 -> al = ah, ah = 0
15979	000046BE	88E0	<1>	mov	al, ah ; 29/06/2015

15980	000046C0	2C08	<1>	sub	al, 8
15981			<1>		; 27/06/2015
15982	000046C2	E85AF4FFFF	<1>	call	sp_status ; get serial port status
15983			<1>		; AL = Line status, AH = Modem status
15984			<1>		; 12/11/2015
15985	000046C7	3C80	<1>	cmp	al, 80h
15986	000046C9	F5	<1>	cmc	
15987			<1>	sysstty_5:	
15988	000046CA	66A3[81740000]	<1>	mov	[u.r0+1], ax ; ah = line status
15989			<1>		; EAX bits 16-23 = modem status
15990	000046D0	9C	<1>	pushf	
15991	000046D1	30D2	<1>	xor	dl, dl ; sysstty call sign
15992	000046D3	A0[CE740000]	<1>	mov	al, [u.ttyn] ; 26/06/2014
15993	000046D8	E869190000	<1>	call	cttyp
15994	000046DD	9D	<1>	popf	
15995	000046DE	0F83B8F9FFFF	<1>	jnc	sysret ; time out error
15996			<1>		
15997			<1>	sysstty_tmout_err:	
15998	000046E4	C705[CF740000]1900-	<1>	mov	dword [u.error], ERR_TIME_OUT
15999	000046EC	0000	<1>		
16000	000046EE	E989F9FFFF	<1>	jmp	error
16001			<1>	sysstty_6:	
16002	000046F3	6652	<1>	push	dx
16003	000046F5	6651	<1>	push	cx
16004	000046F7	891D[98740000]	<1>	mov	[u.namep], ebx
16005	000046FD	E829080000	<1>	call	namei
16006	00004702	6659	<1>	pop	cx
16007	00004704	665A	<1>	pop	dx
16008	00004706	720E	<1>	jc	short sysstty_inv_dn
16009			<1>		
16010	00004708	6683F813	<1>	cmp	ax, 19 ; inode number of /dev/COM2
16011	0000470C	7708	<1>	ja	short sysstty_inv_dn ; 27/06/2015
16012			<1>		
16013	0000470E	3C0A	<1>	cmp	al, 10 ; /dev/tty0 .. /dev/tty7
16014			<1>		; /dev/COM1, /dev/COM2
16015	00004710	7213	<1>	jb	short sysstty_7
16016	00004712	2C0A	<1>	sub	al, 10
16017	00004714	EB20	<1>	jmp	short sysstty_8
16018			<1>	sysstty_inv_dn:	
16019			<1>		; 27/06/2015
16020			<1>		; Invalid device name (not a tty) ! error
16021			<1>		; (Device is not a tty or device name not found)
16022	00004716	C705[CF740000]1800-	<1>	mov	dword [u.error], ERR_INV_DEV_NAME
16023	0000471E	0000	<1>		
16024	00004720	E957F9FFFF	<1>	jmp	error
16025			<1>	sysstty_7:	
16026	00004725	3C01	<1>	cmp	al, 1 ; /dev/tty
16027	00004727	75ED	<1>	jne	short sysstty_inv_dn ; 27/06/2015
16028	00004729	0FB61D[C9740000]	<1>	movzx	ebx, byte [u.uno] ; process number
16029	00004730	8A83[D5710000]	<1>	mov	al, [ebx+p.ttyc-1] ; console tty
16030			<1>	sysstty_8:	
16031	00004736	A2[80740000]	<1>	mov	[u.r0], al
16032	0000473B	6652	<1>	push	dx
16033	0000473D	6650	<1>	push	ax
16034	0000473F	6651	<1>	push	cx
16035	00004741	E8ED170000	<1>	call	ottyp
16036	00004746	6659	<1>	pop	cx
16037	00004748	6658	<1>	pop	ax
16038	0000474A	665A	<1>	pop	dx
16039	0000474C	0F825DFFFFFF	<1>	jc	sysstty_pd_err ; 'permission denied !'
16040			<1>		; 29/10/2015
16041	00004752	86E9	<1>	xchg	ch, cl
16042			<1>		; cl = character, ch = color code
16043	00004754	86C1	<1>	xchg	al, cl
16044			<1>		; al = character, cl = tty number
16045	00004756	80F907	<1>	cmp	cl, 7
16046	00004759	0F873DFFFFFF	<1>	ja	sysstty_12
16047			<1>		
16048			<1>		; 16/01/2014
16049	0000475F	30FF	<1>	xor	bh, bh
16050			<1>		
16051			<1>	sysstty_9:	; tty 0 to tty 7
16052			<1>		; al = character
16053	00004761	80FEFF	<1>	cmp	dh, 0FFh ; Do not set cursor position
16054	00004764	740F	<1>	je	short sysstty_10
16055	00004766	6651	<1>	push	cx
16056	00004768	6650	<1>	push	ax
16057			<1>		; movzx, ebx, cl
16058	0000476A	88CB	<1>	mov	bl, cl ; (tty number = video page number)
16059	0000476C	E8D0CEFFFF	<1>	call	set_cpos
16060	00004771	6658	<1>	pop	ax
16061	00004773	6659	<1>	pop	cx
16062			<1>	sysstty_10:	
16063			<1>		; 29/10/2015
16064	00004775	08C0	<1>	or	al, al ; character
16065	00004777	740F	<1>	jz	short sysstty_11 ; al = 0
16066			<1>		; 17/11/2015
16067	00004779	3CFF	<1>	cmp	al, 0FFh
16068	0000477B	730B	<1>	jnb	short sysstty_11
16069			<1>		; ch > 0 and ch < FFh
16070			<1>		; write a character at current cursor position
16071	0000477D	88EC	<1>	mov	ah, ch ; color/attribute

```

16072      <1> ; 12/07/2014
16073 0000477F 6651      <1> push cx
16074 00004781 E806D0FFFF <1> call write_c_current
16075 00004786 6659      <1> pop cx
16076      <1> sysstty_11:
16077      <1> ; 14/01/2014
16078 00004788 30D2      <1> xor dl, dl ; sysstty call sign
16079      <1> ; 18/01/2014
16080      <1> ;movzx     eax, cl ; 27/06/2015
16081 0000478A 88C8      <1> mov al, cl
16082 0000478C E8B5180000 <1> call cttyp
16083 00004791 E906F9FFFF <1> jmp sysret
16084      <1>
16085      <1> ; Original UNIX v1 'sysstty' routine:
16086      <1> ; gtty:
16087      <1> ;sysstty: / set mode of typewriter; 3 consecutive word arguments
16088      <1> ;jsr     r0,gtty / r1 will have offset to tty block,
16089      <1> ;      / r2 has source
16090      <1> ;mov     r2,-(sp)
16091      <1> ;mov     r1,-(sp) / put r1 and r2 on the stack
16092      <1> ;1: / flush the clist wait till typewriter is quiescent
16093      <1> ;mov     (sp),r1 / restore r1 to tty block offset
16094      <1> ;movb    tty+3(r1),0f / put cc offset into getc argument
16095      <1> ;mov     $240,$ps / set processor priority to 5
16096      <1> ;jsr     r0,getc; 0:.. / put character from clist in r1
16097      <1> ;      br .+4 / list empty, skip branch
16098      <1> ;br      1b / get another character until list is empty
16099      <1> ;mov     0b,r1 / move cc offset to r1
16100      <1> ;inc     r1 / bump it for output clist
16101      <1> ;tstb    cc(r1) / is it 0
16102      <1> ;beq     1f / yes, no characters to output
16103      <1> ;mov     r1,0f / no, put offset in sleep arg
16104      <1> ;jsr     r0,sleep; 0:.. / put tty output process to sleep
16105      <1> ;br      1b / try to calm it down again
16106      <1> ;1:
16107      <1> ;mov     (sp)+,r1
16108      <1> ;mov     (sp)+,r2 / restore registers
16109      <1> ;mov     (r2)+,r3 / put reader control status in r3
16110      <1> ;beq     1f / if 0, 1f
16111      <1> ;mov     r3,rcsr(r1) / move r.c. status to reader
16112      <1> ;      / control status register
16113      <1> ;1:
16114      <1> ;mov     (r2)+,r3 / move pointer control status to r3
16115      <1> ;beq     1f / if 0 1f
16116      <1> ;mov     r3,tcsr(r1) / move p.c. status to printer
16117      <1> ;      / control status reg
16118      <1> ;1:
16119      <1> ;mov     (r2)+,tty+4(r1) / move to flag byte of tty block
16120      <1> ;jmp     sysret2 / return to user
16121      <1>
16122      <1> sysgtty: ; < get tty status >
16123      <1> ; 23/11/2015
16124      <1> ; 29/10/2015
16125      <1> ; 17/10/2015
16126      <1> ; 28/06/2015 (Retro UNIX 386 v1 - Beginning)
16127      <1> ; 30/05/2013 - 12/07/2014 (Retro UNIX 8086 v1)
16128      <1> ;
16129      <1> ; 'sysgtty' gets the status of tty in question.
16130      <1> ; It stores in the three words addressed by it's argument
16131      <1> ; the status of the typewriter whose file descriptor
16132      <1> ; in (u.r0).
16133      <1> ;
16134      <1> ; Calling sequence:
16135      <1> ; sysgtty; arg
16136      <1> ; Arguments:
16137      <1> ; arg - address of 3 words destination of the status
16138      <1> ; Inputs: ((*u.r0 - file descriptor))
16139      <1> ; Outputs: ((status in address which is pointed to by arg))
16140      <1> ; .....
16141      <1> ;
16142      <1> ; Retro UNIX 8086 v1 modification:
16143      <1> ; 'sysgtty' system call will return status of tty
16144      <1> ; (keyboard, serial port and video page status)
16145      <1> ; in following manner:
16146      <1> ;
16147      <1> ; Inputs:
16148      <1> ; BX = 0 --> means
16149      <1> ; CH = 0 --> 'return status of the console tty'
16150      <1> ; for (current) process
16151      <1> ; CL = 0 --> return keyboard status (tty 0 to 9)
16152      <1> ; CL = 1 --> return video page status (tty 0 to 7)
16153      <1> ; CL = 1 --> return serial port status (tty 8 & 9)
16154      <1> ; CH > 0 --> tty number + 1
16155      <1> ;
16156      <1> ; BX > 0 --> points to name of tty
16157      <1> ; CL = 0 --> return keyboard status
16158      <1> ; CL = 1 --> return video page status
16159      <1> ; CH = undefined
16160      <1> ;
16161      <1> ; Outputs:
16162      <1> ; cf = 0 ->
16163      <1> ;

```



```

16164 <1> ; AL = tty number from 0 to 9
16165 <1> ; (0 to 7 is also the video page of the tty)
16166 <1> ; AH = 0 if the tty is free/unused
16167 <1> ; AH = the process number of the caller
16168 <1> ; AH = FFh if the tty is locked by another process
16169 <1> ;
16170 <1> ; (if calling is for serial port status)
16171 <1> ; BX = serial port status if tty number is 8 or 9
16172 <1> ; (BH = modem status, BL = Line status)
16173 <1> ; CX = 0FFFFh (if data is ready)
16174 <1> ; CX = 0 (if data is not ready or undefined)
16175 <1> ;
16176 <1> ; (if calling is for keyboard status)
16177 <1> ; BX = current character in tty/keyboard buffer
16178 <1> ; (BH = scan code, BL = ascii code)
16179 <1> ; (BX=0 if there is not a waiting character)
16180 <1> ; CX is undefined
16181 <1> ;
16182 <1> ; (if calling is for video page status)
16183 <1> ; BX = cursor position on the video page
16184 <1> ; if tty number < 8
16185 <1> ; (BH = row, BL = column)
16186 <1> ; CX = current character (in cursor position)
16187 <1> ; on the video page of the tty
16188 <1> ; if tty number < 8
16189 <1> ; (CH = color, CL = character)
16190 <1> ;
16191 <1> ; cf = 1 means error (requested tty is not ready)
16192 <1> ;
16193 <1> ; AH = FFh if the caller is not owner of
16194 <1> ; specified tty or console tty
16195 <1> ; AL = tty number (0FFh if it does not exist)
16196 <1> ; BX, CX are undefined if cf = 1
16197 <1> ;
16198 <1> ; (If tty number is 8 or 9)
16199 <1> ; AL = tty number
16200 <1> ; AH = the process number of the caller
16201 <1> ; BX = serial port status
16202 <1> ; (BH = modem status, BL = Line status)
16203 <1> ; CX = 0
16204 <1> ;
16205 <1> ;
16206 <1> gtty: ; get (requested) tty number
16207 <1> ; 17/10/2015
16208 <1> ; 28/06/2015 (Retro UNIX 386 v1 - 32 bit modifications)
16209 <1> ; 30/05/2013 - 12/07/2014
16210 <1> ; Retro UNIX 8086 v1 modification !
16211 <1> ;
16212 <1> ; ((Modified regs: eAX, eBX, eCX, eDX, eSI, eDI, eBP))
16213 <1> ;
16214 <1> ; 28/06/2015 (32 bit modifications)
16215 <1> ; 16/01/2014
16216 00004796 31C0 <1> xor eax, eax
16217 00004798 6648 <1> dec ax ; 17/10/2015
16218 0000479A A3[80740000] <1> mov [u.r0], eax ; 0FFFFh
16219 0000479F 80F901 <1> cmp cl, 1
16220 000047A2 760F <1> jna short sysgtty_0
16221 <1> sysgtty_invp:
16222 <1> ; 28/06/2015
16223 000047A4 C705[CF740000]1700- <1> mov dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
16224 000047AC 0000 <1>
16225 000047AE E9C9F8FFFF <1> jmp error
16226 <1> sysgtty_0:
16227 000047B3 21DB <1> and ebx, ebx
16228 000047B5 7430 <1> jz short sysgtty_1
16229 <1> ;
16230 000047B7 891D[98740000] <1> mov [u.namep], ebx
16231 000047BD 6651 <1> push cx ; 23/11/2015
16232 000047BF E867070000 <1> call namei
16233 000047C4 6659 <1> pop cx ; 23/11/2015
16234 000047C6 7210 <1> jc short sysgtty_inv_dn ; 28/06/2015
16235 <1> ;
16236 000047C8 6683F801 <1> cmp ax, 1
16237 000047CC 7622 <1> jna short sysgtty_2
16238 000047CE 6683E80A <1> sub ax, 10
16239 000047D2 6683F809 <1> cmp ax, 9
16240 <1> ;ja short sysgtty_inv_dn
16241 <1> ;mov ch, al
16242 <1> ;jmp short sysgtty_4
16243 <1> ; 23/11/2015
16244 000047D6 7629 <1> jna short sysgtty_4
16245 <1> sysgtty_inv_dn:
16246 <1> ; 28/06/2015
16247 <1> ; Invalid device name (not a tty) ! error
16248 <1> ; (Device is not a tty or device name not found)
16249 000047D8 C705[CF740000]1800- <1> mov dword [u.error], ERR_INV_DEV_NAME
16250 000047E0 0000 <1>
16251 000047E2 E995F8FFFF <1> jmp error
16252 <1> sysgtty_1:
16253 <1> ; 16/01/2014
16254 000047E7 80FD0A <1> cmp ch, 10
16255 000047EA 77B8 <1> ja short sysgtty_invp ; 28/06/2015

```

16256	000047EC	FECD	<1>	dec	ch ; 0 -> FFh (negative)
16257	000047EE	790F	<1>	jns	short sysgtty_3 ; not negative
16258			<1>	;	
16259			<1>	sysgtty_2:	
16260			<1>	;	get tty number of console tty
16261	000047F0	8A25[C9740000]	<1>	mov	ah, [u.uno]
16262			<1>	;	28/06/2015
16263	000047F6	0FB6DC	<1>	movzx	ebx, ah
16264	000047F9	8AAB[D5710000]	<1>	mov	ch, [ebx+p.ttyc-1]
16265			<1>	sysgtty_3:	
16266	000047FF	88E8	<1>	mov	al, ch
16267			<1>	sysgtty_4:	
16268	00004801	A2[80740000]	<1>	mov	[u.r0], al
16269			<1>	;	28/06/2015
16270			<1>	;	cmp al, 9
16271			<1>	;	ja short sysgtty_invp
16272	00004806	8B2D[7C740000]	<1>	mov	ebp, [u.usp]
16273			<1>	;	23/11/2015
16274	0000480C	20C9	<1>	and	cl, cl
16275	0000480E	7436	<1>	jz	short sysgtty_6 ; keyboard status
16276	00004810	3C08	<1>	cmp	al, 8 ; cmp ch, 8
16277	00004812	7232	<1>	jb	short sysgtty_6 ; video page status
16278			<1>	;	serial port status
16279			<1>	;	12/07/2014
16280			<1>	;	mov dx, 0
16281			<1>	;	je short sysgtty_5
16282			<1>	;	inc dl
16283			<1>	;	sysgtty_5:
16284			<1>	;	28/06/2015
16285	00004814	2C08	<1>	sub	al, 8
16286	00004816	E806F3FFFF	<1>	call	sp_status ; serial (COM) port (line) status
16287			<1>	;	AL = Line status, AH = Modem status
16288	0000481B	66894510	<1>	mov	[ebp+16], ax ; serial port status (in EBX)
16289	0000481F	8A25[C9740000]	<1>	mov	ah, [u.uno]
16290	00004825	8825[81740000]	<1>	mov	[u.r0+1], ah
16291	0000482B	66C745180000	<1>	mov	word [ebp+24], 0 ; data status (0 = not ready)
16292			<1>	;	(in ECX)
16293	00004831	A880	<1>	test	al, 80h
16294	00004833	7565	<1>	jnz	short sysgtty_dnr_err ; 29/06/2015
16295	00004835	A801	<1>	test	al, 1
16296	00004837	0F845FF8FFFF	<1>	jz	sysret
16297	0000483D	66FF4D18	<1>	dec	word [ebp+24] ; data status (FFFFh = ready)
16298	00004841	E956F8FFFF	<1>	jmp	sysret
16299			<1>	sysgtty_6:	
16300	00004846	A2[CE740000]	<1>	mov	[u.tty], al ; tty number
16301			<1>	;	movzx ebx, al
16302	0000484B	88C3	<1>	mov	bl, al ; tty number (0 to 9)
16303	0000484D	D0E3	<1>	shl	bl, 1 ; aligned to word
16304			<1>	;	22/04/2014 - 29/06/2015
16305	0000484F	81C3[F4700000]	<1>	add	ebx, ttyl
16306	00004855	8A23	<1>	mov	ah, [ebx]
16307	00004857	3A25[C9740000]	<1>	cmp	ah, [u.uno]
16308	0000485D	7404	<1>	je	short sysgtty_7
16309	0000485F	20E4	<1>	and	ah, ah
16310			<1>	;	jz short sysgtty_7
16311	00004861	7506	<1>	jnz	short sysgtty_8
16312			<1>	;	mov ah, 0FFh
16313			<1>	sysgtty_7:	
16314	00004863	8825[81740000]	<1>	mov	[u.r0+1], ah
16315			<1>	sysgtty_8:	
16316	00004869	08C9	<1>	or	cl, cl
16317	0000486B	7510	<1>	jnz	short sysgtty_9
16318	0000486D	B001	<1>	mov	al, 1 ; test a key is available
16319	0000486F	E85E1C0000	<1>	call	getc
16320	00004874	66894510	<1>	mov	[ebp+16], ax ; bx, character
16321	00004878	E91FF8FFFF	<1>	jmp	sysret
16322			<1>	sysgtty_9:	
16323	0000487D	8A1D[CE740000]	<1>	mov	bl, [u.tty]
16324			<1>	;	bl = video page number
16325	00004883	E8D21D0000	<1>	call	get_cpos
16326			<1>	;	dx = cursor position
16327	00004888	66895510	<1>	mov	[ebp+16], dx ; bx
16328			<1>	;	mov bl, [u.tty]
16329			<1>	;	bl = video page number
16330	0000488C	E8DA1D0000	<1>	call	read_ac_current
16331			<1>	;	ax = character and attribute/color
16332	00004891	66894518	<1>	mov	[ebp+24], ax ; cx
16333	00004895	E902F8FFFF	<1>	jmp	sysret
16334			<1>	sysgtty_dnr_err:	
16335			<1>	;	'device not responding !' error
16336			<1>	;	mov dword [u.error], ERR_TIME_OUT ; 25
16337	0000489A	C705[CF740000]1900-	<1>	mov	dword [u.error], ERR_DEV_NOT_RESP ; 25
16338	000048A2	0000	<1>		
16339	000048A4	E9D3F7FFFF	<1>	jmp	error
16340			<1>		
16341			<1>	;	Original UNIX v1 'sysgtty' routine:
16342			<1>	;	sysgtty:
16343			<1>	;	jsr r0, tty / r1 will have offset to tty block,
16344			<1>	;	/ r2 has destination
16345			<1>	;	mov rcsr(r1), (r2)+ / put reader control status
16346			<1>	;	/ in 1st word of dest
16347			<1>	;	mov tcsr(r1), (r2)+ / put printer control status

```

16348      <1>      ; / in 2nd word of dest
16349      <1>      ;mov     tty+4(r1),(r2)+ / put mode in 3rd word
16350      <1>      ;jmp     sysret2 / return to user
16351      <1>
16352      <1> ; Original UNIX v1 'gtty' routine:
16353      <1> ; gtty:
16354      <1>      ;jsr     r0,arg; u.off / put first arg in u.off
16355      <1>      ;mov     *u.r0,r1 / put file descriptor in r1
16356      <1>      ;jsr     r0,getf / get the i-number of the file
16357      <1>      ;tst     r1 / is it open for reading
16358      <1>      ;bgt     1f / yes
16359      <1>      ;neg     r1 / no, i-number is negative,
16360      <1>      ; / so make it positive
16361      <1> ;1:
16362      <1>      ;sub     $14.,r1 / get i-number of tty0
16363      <1>      ;cmp     r1,$ntty-1 / is there such a typewriter
16364      <1>      ;bhis    error9 / no, error
16365      <1>      ;asl     r1 / 0%2
16366      <1>      ;asl     r1 / 0%4 / yes
16367      <1>      ;asl     r1 / 0%8 / multiply by 8 so r1 points to
16368      <1>      ; / tty block
16369      <1>      ;mov     u.off,r2 / put argument in r2
16370      <1>      ;rts     r0 / return
16371      %include 'u2.s' ; 11/05/2015
16372      <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS2.INC
16373      <1> ; Last Modification: 23/10/2015
16374      <1> ; -----
16375      <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
16376      <1> ; (v0.1 - Beginning: 11/07/2012)
16377      <1> ;
16378      <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
16379      <1> ; (Original) Source Code by Ken Thompson (1971-1972)
16380      <1> ; <Bell Laboratories (17/3/1972)>
16381      <1> ; <Preliminary Release of UNIX Implementation Document>
16382      <1> ;
16383      <1> ; Retro UNIX 8086 v1 - U2.ASM (24/03/2014) //// UNIX v1 -> u2.s
16384      <1> ;
16385      <1> ; *****
16386      <1>
16387      <1> syslink:
16388      <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
16389      <1> ; 19/06/2013 (Retro UNIX 8086 v1)
16390      <1> ;
16391      <1> ; 'syslink' is given two arguments, name 1 and name 2.
16392      <1> ; name 1 is a file that already exists. name 2 is the name
16393      <1> ; given to the entry that will go in the current directory.
16394      <1> ; name2 will then be a link to the name 1 file. The i-number
16395      <1> ; in the name 2 entry of current directory is the same
16396      <1> ; i-number for the name 1 file.
16397      <1> ;
16398      <1> ; Calling sequence:
16399      <1> ; syslink; name 1; name 2
16400      <1> ; Arguments:
16401      <1> ; name 1 - file name to which link will be created.
16402      <1> ; name 2 - name of entry in current directory that
16403      <1> ; links to name 1.
16404      <1> ; Inputs: -
16405      <1> ; Outputs: -
16406      <1> ; .....
16407      <1> ;
16408      <1> ; Retro UNIX 8086 v1 modification:
16409      <1> ; 'syslink' system call has two arguments; so,
16410      <1> ; * 1st argument, name 1 is pointed to by BX register
16411      <1> ; * 2nd argument, name 2 is pointed to by CX register
16412      <1> ;
16413      <1> ; / name1, name2
16414      <1> ;jsr r0,arg2 / u.namep has 1st arg u.off has 2nd
16415 000048A9 891D[98740000] <1> mov [u.namep], ebx
16416 000048AF 51 <1> push ecx
16417 000048B0 E876060000 <1> call namei
16418 <1> ; jsr r0,namei / find the i-number associated with
16419 <1> ; / the 1st path name
16420 <1> ;;and ax, ax
16421 <1> ;;jz error ; File not found
16422 <1> ;jc error
16423 <1> ; br error9 / cannot be found
16424 000048B5 730F <1> jnc short syslink0
16425 <1> ;pop ecx
16426 <1> ; 'file not found !' error
16427 000048B7 C705[CF740000]0C00- <1> mov dword [u.error], ERR_FILE_NOT_FOUND ; 12
16428 000048BF 0000 <1>
16429 000048C1 E9B6F7FFFF <1> jmp error
16430 <1> syslink0:
16431 000048C6 E84F0E0000 <1> call iget
16432 <1> ; jsr r0,iget / get the i-node into core
16433 000048CB 8F05[98740000] <1> pop dword [u.namep] ; ecx
16434 <1> ; mov (sp)+,u.namep / u.namep points to 2nd name
16435 000048D1 6650 <1> push ax
16436 <1> ; mov r1,-(sp) / put i-number of name1 on the stack
16437 <1> ; / (a link to this file is to be created)
16438 000048D3 66FF35[66740000] <1> push word [cdev]
16439 <1> ; mov cdev,-(sp) / put i-nodes device on the stack

```

```

16440 000048DA E855000000 <1> call isdir
16441 <1> ; jsr r0,isdir / is it a directory
16442 000048DF E847060000 <1> call namei
16443 <1> ; jsr r0,namei / no, get i-number of name2
16444 <1> ;jnc error
16445 <1> ; br .+4 / not found
16446 <1> ; / so r1 = i-number of current directory
16447 <1> ; / ii = i-number of current directory
16448 <1> ; br error9 / file already exists., error
16449 000048E4 720F <1> jc short syslink1
16450 <1> ; pop ax
16451 <1> ; pop ax
16452 <1> ; 'file exists !' error
16453 000048E6 C705[CF740000]0E00- <1> mov dword [u.error], ERR_FILE_EXISTS ; 14
16454 000048EE 0000 <1>
16455 000048F0 E987F7FFFF <1> jmp error
16456 <1> syslink1:
16457 000048F5 6659 <1> pop cx
16458 <1> ;cmp cx, [cdev]
16459 000048F7 3A0D[66740000] <1> cmp cl, [cdev]
16460 <1> ;jne error
16461 <1> ; cmp (sp)+,cdev / u.dirp now points to
16462 <1> ; / end of current directory
16463 <1> ; bne error9
16464 000048FD 740F <1> je short syslink2
16465 <1> ; 'not same drive !' error
16466 000048FF C705[CF740000]1500- <1> mov dword [u.error], ERR_DRV_NOT_SAME ; 21
16467 00004907 0000 <1>
16468 00004909 E96EF7FFFF <1> jmp error
16469 <1> syslink2:
16470 0000490E 6658 <1> pop ax
16471 00004910 6650 <1> push ax
16472 00004912 66A3[B2740000] <1> mov [u.dirbuf], ax
16473 <1> ; mov (sp),u.dirbuf / i-number of name1 into u.dirbuf
16474 00004918 E8A8000000 <1> call mkdir
16475 <1> ; jsr r0,mkdir / make directory entry for name2
16476 <1> ; / in current directory
16477 0000491D 6658 <1> pop ax
16478 <1> ; mov (sp)+,r1 / r1 has i-number of name1
16479 0000491F E8F60D0000 <1> call iget
16480 <1> ; jsr r0,iget / get i-node into core
16481 00004924 FE05[58710000] <1> inc byte [i.nlks]
16482 <1> ; incb i.nlks / add 1 to its number of links
16483 0000492A E8FE0E0000 <1> call setimod
16484 <1> ; jsr r0,setimod / set the i-node modified flag
16485 0000492F E968F7FFFF <1> jmp sysret
16486 <1>
16487 <1> isdir:
16488 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
16489 <1> ; 04/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
16490 <1> ;
16491 <1> ; 'isdir' check to see if the i-node whose i-number is in r1
16492 <1> ; is a directory. If it is, an error occurs, because 'isdir'
16493 <1> ; called by syslink and sysunlink to make sure directories
16494 <1> ; are not linked. If the user is the super user (u.uid=0),
16495 <1> ; 'isdir' does not bother checking. The current i-node
16496 <1> ; is not disturbed.
16497 <1> ;
16498 <1> ; INPUTS ->
16499 <1> ; r1 - contains the i-number whose i-node is being checked.
16500 <1> ; u.uid - user id
16501 <1> ; OUTPUTS ->
16502 <1> ; r1 - contains current i-number upon exit
16503 <1> ; (current i-node back in core)
16504 <1> ;
16505 <1> ; ((AX = R1))
16506 <1> ;
16507 <1> ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
16508 <1> ;
16509 <1>
16510 <1> ; / if the i-node whose i-number is in r1 is a directory
16511 <1> ; / there is an error unless super user made the call
16512 <1>
16513 00004934 803D[C6740000]00 <1> cmp byte [u.uid], 0
16514 <1> ; tstb u.uid / super user
16515 0000493B 762D <1> jna short isdir1
16516 <1> ; beq 1f / yes, don't care
16517 0000493D 66FF35[62740000] <1> push word [ii]
16518 <1> ; mov ii,-(sp) / put current i-number on stack
16519 00004944 E8D10D0000 <1> call iget
16520 <1> ; jsr r0,iget / get i-node into core (i-number in r1)
16521 00004949 66F705[56710000]00- <1> test word [i.flgs], 4000h ; Bit 14 : Directory flag
16522 00004951 40 <1>
16523 <1> ; bit $40000,i.flgs / is it a directory
16524 <1> ;jnz error
16525 <1> ; bne error9 / yes, error
16526 00004952 740F <1> jz short isdir0
16527 00004954 C705[CF740000]0B00- <1> mov dword [u.error], ERR_NOT_FILE ; 11 ; ERR_DIR_ACCESS
16528 0000495C 0000 <1>
16529 <1> ; 'permission denied !' error
16530 <1> ; pop ax
16531 0000495E E919F7FFFF <1> jmp error

```

16532		<1> isdir0:
16533	00004963 6658	<1> pop ax
16534		<1> ; mov (sp)+,r1 / no, put current i-number in r1 (ii)
16535	00004965 E8B00D0000	<1> call iget
16536		<1> ; jsr r0,iget / get it back in
16537		<1> isdir1: ; 1:
16538	0000496A C3	<1> retn
16539		<1> ; rts r0
16540		<1>
16541		<1> sysunlink:
16542		<1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
16543		<1> ; 19/06/2013 (Retro UNIX 8086 v1)
16544		<1> ;
16545		<1> ; 'sysunlink' removes the entry for the file pointed to by
16546		<1> ; name from its directory. If this entry was the last link
16547		<1> ; to the file, the contents of the file are freed and the
16548		<1> ; file is destroyed. If, however, the file was open in any
16549		<1> ; process, the actual destruction is delayed until it is
16550		<1> ; closed, even though the directory entry has disappeared.
16551		<1> ;
16552		<1> ; The error bit (e-bit) is set to indicate that the file
16553		<1> ; does not exist or that its directory can not be written.
16554		<1> ; Write permission is not required on the file itself.
16555		<1> ; It is also illegal to unlink a directory (except for
16556		<1> ; the superuser).
16557		<1> ;
16558		<1> ; Calling sequence:
16559		<1> ; sysunlink; name
16560		<1> ; Arguments:
16561		<1> ; name - name of directory entry to be removed
16562		<1> ; Inputs: -
16563		<1> ; Outputs: -
16564		<1> ; .....
16565		<1> ;
16566		<1> ; Retro UNIX 8086 v1 modification:
16567		<1> ; The user/application program puts address of the name
16568		<1> ; in BX register as 'sysunlink' system call argument.
16569		<1>
16570		<1> ; / name - remove link name
16571	0000496B 891D[98740000]	<1> mov [u.namep], ebx
16572		<1> ; jsr r0,arg; u.namep / u.namep points to name
16573	00004971 E8B5050000	<1> call namei
16574		<1> ; jsr r0,namei / find the i-number associated
16575		<1> ; / with the path name
16576		<1> ;jc error
16577		<1> ; br error9 / not found
16578	00004976 730F	<1> jnc short sysunlink1
16579		<1> ; 'file not found !' error
16580	00004978 C705[CF740000]0C00-	<1> mov dword [u.error], ERR_FILE_NOT_FOUND ; 12
16581	00004980 0000	<1>
16582	00004982 E9F5F6FFFF	<1> jmp error
16583		<1> sysunlink1:
16584	00004987 6650	<1> push ax
16585		<1> ; mov r1,-(sp) / put its i-number on the stack
16586	00004989 E8A6FFFFFF	<1> call isdir
16587		<1> ; jsr r0,isdir / is it a directory
16588	0000498E 6631C0	<1> xor ax, ax
16589	00004991 66A3[B2740000]	<1> mov [u.dirbuf], ax ; 0
16590		<1> ; clr u.dirbuf / no, clear the location that will
16591		<1> ; / get written into the i-number portion
16592		<1> ; / of the entry
16593	00004997 832D[9C740000]0A	<1> sub dword [u.off], 10
16594		<1> ; sub \$10,,u.off / move u.off back 1 directory entry
16595	0000499E E86B000000	<1> call wdir
16596		<1> ; jsr r0,wdir / free the directory entry
16597	000049A3 6658	<1> pop ax
16598		<1> ; mov (sp)+,r1 / get i-number back
16599	000049A5 E8700D0000	<1> call iget
16600		<1> ; jsr r0,iget / get i-node
16601	000049AA E87E0E0000	<1> call setimod
16602		<1> ; jsr r0,setimod / set modified flag
16603	000049AF FE0D[58710000]	<1> dec byte [i.nlks]
16604		<1> ; decb i.nlks / decrement the number of links
16605	000049B5 0F85E1F6FFFF	<1> jnz sysret
16606		<1> ; bgt sysret9 / if this was not the last link
16607		<1> ; / to file return
16608		<1> ; AX = r1 = i-number
16609	000049BB E88F090000	<1> call anyi
16610		<1> ; jsr r0,anyi / if it was, see if anyone has it open.
16611		<1> ; / Then free contents of file and destroy it.
16612	000049C0 E9D7F6FFFF	<1> jmp sysret
16613		<1> ; br sysret9
16614		<1>
16615		<1> mkdir:
16616		<1> ; 12/10/2015
16617		<1> ; 17/06/2015 (Retro UNIX 386 v1 - Beginning)
16618		<1> ; 29/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
16619		<1> ;
16620		<1> ; 'mkdir' makes a directory entry from the name pointed to
16621		<1> ; by u.namep into the current directory.
16622		<1> ;
16623		<1> ; INPUTS ->

```

16624 <1> ; u.namep - points to a file name
16625 <1> ; that is about to be a directory entry.
16626 <1> ; ii - current directory's i-number.
16627 <1> ; OUTPUTS ->
16628 <1> ; u.dirbuf+2 - u.dirbuf+10 - contains file name.
16629 <1> ; u.off - points to entry to be filled
16630 <1> ; in the current directory
16631 <1> ; u.base - points to start of u.dirbuf.
16632 <1> ; r1 - contains i-number of current directory
16633 <1> ;
16634 <1> ; ((AX = R1)) output
16635 <1> ;
16636 <1> ; (Retro UNIX Prototype : 11/11/2012, UNIXCOPY.ASM)
16637 <1> ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
16638 <1> ;
16639 <1>
16640 <1> ; 17/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
16641 000049C5 31C0 <1> xor eax, eax
16642 000049C7 BF[B4740000] <1> mov edi, u.dirbuf+2
16643 000049CC 89FE <1> mov esi, edi
16644 000049CE AB <1> stosd
16645 000049CF AB <1> stosd
16646 <1> ; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this
16647 000049D0 89F7 <1> mov edi, esi ; offset to u.dirbuf
16648 <1> ; 12/10/2015 ([u.namep] -> ebp)
16649 <1> ;mov ebp, [u.namep]
16650 000049D2 E899060000 <1> call trans_addr_nmbp ; convert virtual address to physical
16651 <1> ; esi = physical address (page start + offset)
16652 <1> ; ecx = byte count in the page (1 - 4096)
16653 <1> ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
16654 <1> ; mov u.namep,r2 / r2 points to name of directory entry
16655 <1> ; mov $u.dirbuf+2,r3 / r3 points to u.dirbuf+2
16656 <1> mkdir_1: ; 1:
16657 000049D7 45 <1> inc ebp ; 12/10/2015
16658 <1> ;
16659 <1> ; / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10
16660 <1> ; 01/08/2013
16661 000049D8 AC <1> lodsb
16662 <1> ; movb (r2)+,r1 / move character in name to r1
16663 000049D9 20C0 <1> and al, al
16664 000049DB 7427 <1> jz short mkdir_3
16665 <1> ; beq 1f / if null, done
16666 000049DD 3C2F <1> cmp al, '/'
16667 <1> ; cmp r1,$'/' / is it a "/"?
16668 000049DF 7414 <1> je short mkdir_err
16669 <1> ;je error
16670 <1> ; beq error9 / yes, error
16671 <1> ; 12/10/2015
16672 000049E1 6649 <1> dec cx
16673 000049E3 7505 <1> jnz short mkdir_2
16674 <1> ; 12/10/2015 ([u.namep] -> ebp)
16675 000049E5 E88C060000 <1> call trans_addr_nm ; convert virtual address to physical
16676 <1> ; esi = physical address (page start + offset)
16677 <1> ; ecx = byte count in the page
16678 <1> ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
16679 <1> mkdir_2:
16680 000049EA 81FF[BC740000] <1> cmp edi, u.dirbuf+10
16681 <1> ; cmp r3,$u.dirbuf+10. / have we reached the last slot for
16682 <1> ; / a char?
16683 000049F0 74E5 <1> je short mkdir_1
16684 <1> ; beq 1b / yes, go back
16685 000049F2 AA <1> stosb
16686 <1> ; movb r1,(r3)+ / no, put the char in the u.dirbuf
16687 000049F3 EBE2 <1> jmp short mkdir_1
16688 <1> ; br 1b / get next char
16689 <1> mkdir_err:
16690 <1> ; 17/06/2015
16691 000049F5 C705[CF740000]1300- <1> mov dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
16692 000049FD 0000 <1>
16693 000049FF E978F6FFFF <1> jmp error
16694 <1>
16695 <1> mkdir_3: ; 1:
16696 00004A04 A1[94740000] <1> mov eax, [u.dirp]
16697 00004A09 A3[9C740000] <1> mov [u.off], eax
16698 <1> ; mov u.dirp,u.off / pointer to empty current directory
16699 <1> ; / slot to u.off
16700 <1> wdir: ; 29/04/2013
16701 00004A0E C705[A0740000]- <1> mov dword [u.base], u.dirbuf
16702 00004A14 [B2740000] <1>
16703 <1> ; mov $u.dirbuf,u.base / u.base points to created file name
16704 00004A18 C705[A4740000]0A00- <1> mov dword [u.count], 10
16705 00004A20 0000 <1>
16706 <1> ; mov $10.,u.count / u.count = 10
16707 00004A22 66A1[62740000] <1> mov ax, [ii]
16708 <1> ; mov ii,r1 / r1 has i-number of current directory
16709 00004A28 B201 <1> mov dl, 1 ; owner flag mask ; RETRO UNIX 8086 v1 modification !
16710 00004A2A E8C60D0000 <1> call access
16711 <1> ; jsr r0,access; 1 / get i-node and set its file up
16712 <1> ; / for writing
16713 <1> ; AX = i-number of current directory
16714 <1> ; 01/08/2013
16715 00004A2F FE05[E1740000] <1> inc byte [u.kcall] ; the caller is 'mkdir' sign

```

```

16716 00004A35 E8B6100000 <1> call writei
16717 <1> ; jsr r0,writei / write into directory
16718 00004A3A C3 <1> retn
16719 <1> ; rts r0
16720 <1>
16721 <1> sysexec:
16722 <1> ; 23/10/2015
16723 <1> ; 19/10/2015
16724 <1> ; 18/10/2015
16725 <1> ; 10/10/2015
16726 <1> ; 26/08/2015
16727 <1> ; 05/08/2015
16728 <1> ; 29/07/2015
16729 <1> ; 25/07/2015
16730 <1> ; 24/07/2015
16731 <1> ; 21/07/2015
16732 <1> ; 20/07/2015
16733 <1> ; 02/07/2015
16734 <1> ; 01/07/2015
16735 <1> ; 25/06/2015
16736 <1> ; 24/06/2015
16737 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
16738 <1> ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
16739 <1> ;
16740 <1> ; 'sysexec' initiates execution of a file whose path name if
16741 <1> ; pointed to by 'name' in the sysexec call.
16742 <1> ; 'sysexec' performs the following operations:
16743 <1> ; 1. obtains i-number of file to be executed via 'namei'.
16744 <1> ; 2. obtains i-node of file to be executed via 'iget'.
16745 <1> ; 3. sets trap vectors to system routines.
16746 <1> ; 4. loads arguments to be passed to executing file into
16747 <1> ; highest locations of user's core
16748 <1> ; 5. puts pointers to arguments in locations immediately
16749 <1> ; following arguments.
16750 <1> ; 6. saves number of arguments in next location.
16751 <1> ; 7. initializes user's stack area so that all registers
16752 <1> ; will be zeroed and the PS is cleared and the PC set
16753 <1> ; to core when 'sysret' restores registers
16754 <1> ; and does an rti.
16755 <1> ; 8. initializes u.r0 and u.sp
16756 <1> ; 9. zeros user's core down to u.r0
16757 <1> ; 10. reads executable file from storage device into core
16758 <1> ; starting at location 'core'.
16759 <1> ; 11. sets u.break to point to end of user's code with
16760 <1> ; data area appended.
16761 <1> ; 12. calls 'sysret' which returns control at location
16762 <1> ; 'core' via 'rti' instruction.
16763 <1> ;
16764 <1> ; Calling sequence:
16765 <1> ; sysexec; namep; argp
16766 <1> ; Arguments:
16767 <1> ; namep - points to pathname of file to be executed
16768 <1> ; argp - address of table of argument pointers
16769 <1> ; argp1... argpn - table of argument pointers
16770 <1> ; argp1:<...0> ... argpn:<...0> - argument strings
16771 <1> ; Inputs: (arguments)
16772 <1> ; Outputs: -
16773 <1> ; .....
16774 <1> ;
16775 <1> ; Retro UNIX 386 v1 modification:
16776 <1> ; User application runs in it's own virtual space
16777 <1> ; which is isolated from kernel memory (and other
16778 <1> ; memory pages) via 80386 paging in ring 3
16779 <1> ; privilege mode. Virtual start address is always 0.
16780 <1> ; User's core memory starts at linear address 400000h
16781 <1> ; (the end of the 1st 4MB).
16782 <1> ;
16783 <1> ; Retro UNIX 8086 v1 modification:
16784 <1> ; user/application segment and system/kernel segment
16785 <1> ; are different and sysenter/sysret/sysrele routines
16786 <1> ; are different (user's registers are saved to
16787 <1> ; and then restored from system's stack.)
16788 <1> ;
16789 <1> ; NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
16790 <1> ; arguments which were in these registers;
16791 <1> ; but, it returns by putting the 1st argument
16792 <1> ; in 'u.namep' and the 2nd argument
16793 <1> ; on top of stack. (1st argument is offset of the
16794 <1> ; file/path name in the user's program segment.)
16795 <1>
16796 <1> ;call arg2
16797 <1> ; * name - 'u.namep' points to address of file/path name
16798 <1> ; in the user's program segment ('u.segmt')
16799 <1> ; with offset in BX register (as sysopen argument 1).
16800 <1> ; * argp - sysexec argument 2 is in CX register
16801 <1> ; which is on top of stack.
16802 <1> ;
16803 <1> ; jsr r0,arg2 / arg0 in u.namep,arg1 on top of stack
16804 <1>
16805 <1> ; 23/06/2015 (32 bit modifications)
16806 <1>
16807 00004A3B 891D[98740000] <1> mov [u.namep], ebx ; argument 1

```

```

16808                                <1>         ; 18/10/2015
16809 00004A41 890D[F8740000]      <1> mov     [argv], ecx ; * ; argument 2
16810 00004A47 E8DF040000          <1> call    namei
16811                                <1>         ; jsr r0,namei / namei returns i-number of file
16812                                <1>         ; / named in sysexec call in r1
16813                                <1> ;jc     error
16814                                <1>         ; br error9
16815 00004A4C 731E                  <1> jnc     short sysexec_0
16816                                <1>         ;
16817                                <1>         ; 'file not found !' error
16818 00004A4E C705[CF740000]0C00- <1> mov     dword [u.error], ERR_FILE_NOT_FOUND
16819 00004A56 0000                  <1>
16820 00004A58 E91FF6FFFF          <1> jmp     error
16821                                <1> sysexec_not_exf:
16822                                <1>         ; 'not executable file !' error
16823 00004A5D C705[CF740000]1600- <1> mov     dword [u.error], ERR_NOT_EXECUTABLE
16824 00004A65 0000                  <1>
16825 00004A67 E910F6FFFF          <1> jmp     error
16826                                <1> sysexec_0:
16827 00004A6C E8A90C0000          <1> call    iget
16828                                <1>         ; jsr r0,iget / get i-node for file to be executed
16829 00004A71 66F705[56710000]10- <1> test     word [i.flgs], 10h
16830 00004A79 00                    <1>
16831                                <1>         ; bit $20,i.flgs / is file executable
16832 00004A7A 74E1                  <1> jz      short sysexec_not_exf
16833                                <1> ;jz     error
16834                                <1>         ; beq error9
16835                                <1> ;;
16836 00004A7C E818140000          <1> call    iopen
16837                                <1>         ; jsr r0,iopen / gets i-node for file with i-number
16838                                <1>         ; / given in r1 (opens file)
16839                                <1>         ; AX = i-number of the file
16840 00004A81 66F705[56710000]20- <1> test     word [i.flgs], 20h
16841 00004A89 00                    <1>
16842                                <1>         ; bit $40,i.flgs / test user id on execution bit
16843 00004A8A 7415                  <1> jz      short sysexec_1
16844                                <1>         ; beq 1f
16845 00004A8C 803D[C6740000]00      <1> cmp     byte [u.uid], 0 ; 02/08/2013
16846                                <1>         ; tstb u.uid / test user id
16847 00004A93 760C                  <1> jna     short sysexec_1
16848                                <1>         ; beq 1f / super user
16849 00004A95 8A0D[59710000]        <1> mov     cl, [i.uid]
16850 00004A9B 880D[C6740000]        <1> mov     [u.uid], cl ; 02/08/2013
16851                                <1>         ; movb i.uid,u.uid / put user id of owner of file
16852                                <1>         ; / as process user id
16853                                <1> sysexec_1:
16854                                <1>         ; 18/10/2215
16855                                <1>         ; 10/10/2015
16856                                <1>         ; 24/07/2015
16857                                <1>         ; 21/07/2015
16858                                <1>         ; 25/06/2015
16859                                <1>         ; 24/06/2015
16860                                <1>         ; Moving arguments to the end of [u.upage]
16861                                <1>         ; (by regarding page borders in user's memory space)
16862                                <1>         ;
16863                                <1>         ; 10/10/2015
16864                                <1>         ; 21/07/2015
16865 00004AA1 89E5                  <1> mov     ebp, esp ; (**)
16866                                <1>         ; 18/10/2015
16867 00004AA3 89EF                  <1> mov     edi, ebp
16868 00004AA5 B900010000          <1> mov     ecx, MAX_ARG_LEN ; 256
16869                                <1> ;sub    edi, MAX_ARG_LEN ; 256
16870 00004AAA 29CF                  <1> sub     edi, ecx
16871 00004AAC 89FC                  <1> mov     esp, edi
16872 00004AAE 31C0                  <1> xor     eax, eax
16873 00004AB0 A3[A8740000]          <1> mov     [u.nread], eax ; 0
16874 00004AB5 49                    <1> dec     ecx ; 256 - 1
16875 00004AB6 890D[A4740000]        <1> mov     [u.count], ecx ; MAX_ARG_LEN - 1 ; 255
16876                                <1> ;mov    dword [u.count], MAX_ARG_LEN - 1 ; 255
16877                                <1> sysexec_2:
16878 00004ABC 8B35[F8740000]          <1> mov     esi, [argv] ; 18/10/2015
16879 00004AC2 E873020000          <1> call    get_argp
16880 00004AC7 B904000000          <1> mov     ecx, 4 ; mov ecx, 4
16881                                <1> sysexec_3:
16882 00004ACC 21C0                  <1> and     eax, eax
16883 00004ACE 7456                  <1> jz      short sysexec_6
16884                                <1>         ; 18/10/2015
16885 00004AD0 010D[F8740000]          <1> add     [argv], ecx ; 4
16886 00004AD6 66FF05[F6740000]        <1> inc     word [argc]
16887                                <1>         ;
16888 00004ADD A3[A0740000]          <1> mov     [u.base], eax
16889                                <1>         ; 23/10/2015
16890 00004AE2 66C705[DF740000]00- <1> mov     word [u.pcount], 0
16891 00004AEA 00                    <1>
16892                                <1> sysexec_4:
16893 00004AEB E8BB110000          <1> call    cpass ; get a character from user's core memory
16894 00004AF0 750B                  <1>         jnz      short sysexec_5
16895                                <1>         ; (max. 255 chars + null)
16896                                <1>         ; 18/10/2015
16897 00004AF2 28C0                  <1> sub     al, al
16898 00004AF4 AA                    <1> stosb
16899 00004AF5 FF05[A8740000]        <1> inc     dword [u.nread]

```



16900	00004AFB	EB29	<1>	jmp	short	sysexec_6
16901			<1>	sysexec_5:		
16902	00004AFD	AA	<1>	stosb		
16903	00004AFE	20C0	<1>	and	al, al	
16904	00004B00	75E9	<1>	jnz	short	sysexec_4
16905	00004B02	B904000000	<1>	mov	ecx, 4	
16906	00004B07	390D[F4740000]	<1>	cmp	[ncount], ecx ; 4	
16907	00004B0D	72AD	<1>	jb	short	sysexec_2
16908	00004B0F	8B35[F0740000]	<1>	mov	esi, [nbase]	
16909	00004B15	010D[F0740000]	<1>	add	[nbase], ecx ; 4	
16910	00004B1B	66290D[F4740000]	<1>	sub	[ncount], cx	
16911	00004B22	8B06	<1>	mov	eax, [esi]	
16912	00004B24	EBA6	<1>	jmp	short	sysexec_3
16913			<1>	sysexec_6:		
16914			<1>	; 18/10/2015		
16915			<1>	; argument list transfer from user's core memory to		
16916			<1>	; kernel stack frame is OK here.		
16917			<1>	; [u.nread] = ; argument list length		
16918			<1>	;mov [argv], esp ; start address of argument list		
16919			<1>	;		
16920			<1>	; 18/10/2015		
16921			<1>	; 24/07/2015		
16922			<1>	; 21/07/2015		
16923			<1>	; 02/07/2015		
16924			<1>	; 25/06/2015		
16925			<1>	; 24/06/2015		
16926			<1>	; 23/06/2015		
16927			<1>	;		
16928	00004B26	8B1D[D7740000]	<1>	mov	ebx, [u.ppgdir] ; parent's page directory	
16929	00004B2C	21DB	<1>	and	ebx, ebx ; /etc/init ? (u.ppgdir = 0)	
16930	00004B2E	740A	<1>	jz	short	sysexec_7
16931	00004B30	A1[D3740000]	<1>	mov	eax, [u.ppgdir] ; physical address of page directory	
16932	00004B35	E836E6FFFF	<1>	call	deallocate_page_dir	
16933			<1>	sysexec_7:		
16934	00004B3A	E866E5FFFF	<1>	call	make_page_dir	
16935			<1>	;jc short sysexec_14		
16936	00004B3F	0F82CDEDFFFF	<1>	jc	panic ; allocation error	
16937			<1>	; after a deallocation would be nonsense !?		
16938			<1>	; 24/07/2015		
16939			<1>	; map kernel pages (1st 4MB) to PDE 0		
16940			<1>	; of the user's page directory		
16941			<1>	; (It is needed for interrupts!)		
16942			<1>	; 18/10/2015		
16943	00004B45	8B15[A8700000]	<1>	mov	edx, [k_page_dir] ; Kernel's page directory	
16944	00004B4B	8B02	<1>	mov	eax, [edx] ; physical address of	
16945			<1>	; kernel's first page table (1st 4 MB)		
16946			<1>	; (PDE 0 of kernel's page directory)		
16947	00004B4D	8B15[D3740000]	<1>	mov	edx, [u.pgdir]	
16948	00004B53	8902	<1>	mov	[edx], eax ; PDE 0 (1st 4MB)	
16949			<1>	;		
16950			<1>	; 20/07/2015		
16951	00004B55	BB00004000	<1>	mov	ebx, CORE ; start address = 0 (virtual) + CORE	
16952			<1>	; 18/10/2015		
16953	00004B5A	BE[E8740000]	<1>	mov	esi, pcore ; physical start address	
16954			<1>	sysexec_8:		
16955	00004B5F	B907000000	<1>	mov	ecx, PDE_A_USER + PDE_A_WRITE + PDE_A_PRESENT	
16956	00004B64	E85AE5FFFF	<1>	call	make_page_table	
16957	00004B69	0F82A3EDFFFF	<1>	jc	panic	
16958			<1>	;mov ecx, PTE_A_USER + PTE_A_WRITE + PTE_A_PRESENT		
16959	00004B6F	E85DE5FFFF	<1>	call	make_page ; make new page, clear and set the pte	
16960	00004B74	0F8298EDFFFF	<1>	jc	panic	
16961			<1>	;		
16962	00004B7A	8906	<1>	mov	[esi], eax ; 24/06/2015	
16963			<1>	; ebx = virtual address (24/07/2015)		
16964	00004B7C	E873EAFFFF	<1>	call	add_to_swap_queue	
16965			<1>	; 18/10/2015		
16966	00004B81	81FE[EC740000]	<1>	cmp	esi, ecore ; user's stack (last) page ?	
16967	00004B87	740C	<1>	je	short	sysexec_9 ; yes
16968	00004B89	BE[EC740000]	<1>	mov	esi, ecore ; physical address of the last page	
16969			<1>	; 20/07/2015		
16970	00004B8E	BB00F0FFFF	<1>	mov	ebx, (ECORE - PAGE_SIZE) + CORE	
16971			<1>	; ebx = virtual end address + segment base address - 4K		
16972	00004B93	EBCA	<1>	jmp	short	sysexec_8
16973			<1>			
16974			<1>	sysexec_9:		
16975			<1>	; 18/10/2015		
16976			<1>	; 26/08/2015		
16977			<1>	; 25/06/2015		
16978			<1>	; move arguments from kernel stack to [ecore]		
16979			<1>	; (argument list/line will be copied from kernel stack		
16980			<1>	; frame to the last (stack) page of user's core memory)		
16981			<1>	; 18/10/2015		
16982	00004B95	8B3D[EC740000]	<1>	mov	edi, [ecore]	
16983	00004B9B	81C700100000	<1>	add	edi, PAGE_SIZE	
16984	00004BA1	0FB705[F6740000]	<1>	movzx	eax, word [argc]	
16985	00004BA8	09C0	<1>	or	eax, eax	
16986	00004BAA	7509	<1>	jnz	short	sysexec_10
16987	00004BAC	89FB	<1>	mov	ebx, edi	
16988	00004BAE	83EB04	<1>	sub	ebx, 4	
16989	00004BB1	8903	<1>	mov	[ebx], eax ; 0	
16990	00004BB3	EB40	<1>	jmp	short	sysexec_13
16991			<1>	sysexec_10:		

16992	00004BB5	8B0D[A8740000]	<1>	mov	ecx, [u.nread]
16993			<1>	;mov	esi, [argv]
16994	00004BBB	89E6	<1>	mov	esi, esp ; start address of argument list
16995	00004BBD	29CF	<1>	sub	edi, ecx ; page end address - argument list length
16996	00004BBF	89C2	<1>	mov	edx, eax
16997	00004BC1	FEC2	<1>	inc	dl ; argument count + 1 for argc value
16998	00004BC3	C0E202	<1>	shl	dl, 2 ; 4 * (argument count + 1)
16999	00004BC6	89FB	<1>	mov	ebx, edi
17000	00004BC8	80E3FC	<1>	and	bl, 0FCh ; 32 bit (dword) alignment
17001	00004BCB	29D3	<1>	sub	ebx, edx
17002	00004BCD	89FA	<1>	mov	edx, edi
17003	00004BCF	F3A4	<1>	rep	movsb
17004	00004BD1	89D6	<1>	mov	esi, edx
17005	00004BD3	89DF	<1>	mov	edi, ebx
17006	00004BD5	BA00F0BFFF	<1>	mov	edx, ECORE - PAGE_SIZE ; virtual addr. of the last page
17007	00004BDA	2B15[EC740000]	<1>	sub	edx, [ecore] ; difference (virtual - physical)
17008	00004BE0	AB	<1>	stosd	; eax = argument count
17009			<1>	sysexec_11:	
17010	00004BE1	89F0	<1>	mov	eax, esi
17011	00004BE3	01D0	<1>	add	eax, edx
17012	00004BE5	AB	<1>	stosd	; eax = virtual address
17013	00004BE6	FE0D[F6740000]	<1>	dec	byte [argc]
17014	00004BEC	7407	<1>	jz	short sysexec_13
17015			<1>	sysexec_12:	
17016	00004BEE	AC	<1>	lodsb	
17017	00004BEF	20C0	<1>	and	al, al
17018	00004BF1	75FB	<1>	jnz	short sysexec_12
17019	00004BF3	EBEC	<1>	jmp	short sysexec_11
17020			<1>	;	
17021			<1>	; 1:	
17022			<1>	;	mov (sp)+, r5 / r5 now contains address of list of
17023			<1>	;	pointers to arguments to be passed
17024			<1>	;	mov \$1, u.quit / u.quit determines handling of quits;
17025			<1>	;	u.quit = 1 take quit
17026			<1>	;	mov \$1, u.intr / u.intr determines handling of
17027			<1>	;	interrupts; u.intr = 1 take interrupt
17028			<1>	;	mov \$rtssym, 30 / emt trap vector set to take
17029			<1>	;	system routine
17030			<1>	;	mov \$fpsym, *10 / reserved instruction trap vector
17031			<1>	;	set to take system routine
17032			<1>	;	mov \$sstack, sp / stack space used during swapping
17033			<1>	;	mov r5, -(sp) / save arguments pointer on stack
17034			<1>	;	mov \$ecore, r5 / r5 has end of core
17035			<1>	;	mov \$core, r4 / r4 has start of users core
17036			<1>	;	mov r4, u.base / u.base has start of users core
17037			<1>	;	mov (sp), r2 / move arguments list pointer into r2
17038			<1>	; 1:	
17039			<1>	;	tst (r2)+ / argument char = "nul"
17040			<1>	;	bne 1b
17041			<1>	;	tst -(r2) / decrement r2 by 2; r2 has addr of
17042			<1>	;	end of argument pointer list
17043			<1>	; 1:	
17044			<1>	;	move arguments to bottom of users core
17045			<1>	;	mov -(r2), r3 / (r3) last non zero argument ptr
17046			<1>	;	cmp r2, (sp) / is r2 = beginning of argument
17047			<1>	;	ptr list
17048			<1>	;	blo 1f / branch to 1f when all arguments
17049			<1>	;	are moved
17050			<1>	;	mov -(r2), r3 / (r3) last non zero argument ptr
17051			<1>	; 2:	
17052			<1>	;	tstb (r3)+
17053			<1>	;	bne 2b / scan argument for \0 (nul)
17054			<1>		
17055			<1>	; 2:	
17056			<1>	;	movb -(r3), -(r5) / move argument char
17057			<1>	;	by char starting at "ecore"
17058			<1>	;	cmp r3, (r2) / moved all characters in
17059			<1>	;	this argument
17060			<1>	;	bhi 2b / branch 2b if not
17061			<1>	;	mov r5, (r4)+ / move r5 into top of users core;
17062			<1>	;	r5 has pointer to nth arg
17063			<1>	;	br 1b / string
17064			<1>	; 1:	
17065			<1>	;	clrb -(r5)
17066			<1>	;	bic \$1, r5 / make r5 even, r5 points to
17067			<1>	;	last word of argument strings
17068			<1>	;	mov \$core, r2
17069			<1>		
17070			<1>	; 1: /	move argument pointers into core following
17071			<1>	;	argument strings
17072			<1>	;	cmp r2, r4
17073			<1>	;	bhis 1f / branch to 1f when all pointers
17074			<1>	;	are moved
17075			<1>	;	mov (r2)+, -(r5)
17076			<1>	;	br 1b
17077			<1>	; 1:	
17078			<1>	;	sub \$core, r4 / gives number of arguments *2
17079			<1>	;	asr r4 / divide r4 by 2 to calculate
17080			<1>	;	the number of args stored
17081			<1>	;	mov r4, -(r5) / save number of arguments ahead
17082			<1>	;	of the argument pointers
17083			<1>	sysexec_13:	

```

17084 <1> ; 19/10/2015
17085 <1> ; 18/10/2015
17086 <1> ; 29/07/2015
17087 <1> ; 25/07/2015
17088 <1> ; 24/07/2015
17089 <1> ; 20/07/2015
17090 <1> ; 25/06/2015
17091 <1> ; 24/06/2015
17092 <1> ; 23/06/2015
17093 <1> ;
17094 <1> ; moving arguments to [ecore] is OK here..
17095 <1> ; 18/10/2015
17096 00004BF5 89EC <1> mov esp, ebp ; (**) restore kernel stack pointer
17097 <1> ; ebx = beginning address of argument list pointers
17098 <1> ; in user's stack
17099 <1> ; 19/10/2015
17100 00004BF7 2B1D[EC740000] <1> sub ebx, [ecore]
17101 00004BFD 81C300F0BFFF <1> add ebx, (ECORE - PAGE_SIZE)
17102 <1> ; end of core - 4096 (last page)
17103 <1> ; (virtual address)
17104 00004C03 891D[F8740000] <1> mov [argv], ebx
17105 00004C09 891D[AC740000] <1> mov [u.break], ebx ; available user memory
17106 <1> ;
17107 00004C0F 29C0 <1> sub eax, eax
17108 00004C11 C705[A4740000]2000- <1> mov dword [u.count], 32 ; Executable file header size
17109 00004C19 0000 <1>
17110 <1> ; mov $14,u.count
17111 00004C1B C705[90740000]- <1> mov dword [u.fofp], u.off
17112 00004C21 [9C740000] <1>
17113 <1> ; mov $u.off,u.fofp
17114 00004C25 A3[9C740000] <1> mov [u.off], eax ; 0
17115 <1> ; clr u.off / set offset in file to be read to zero
17116 <1> ; 25/07/2015
17117 00004C2A A3[A0740000] <1> mov [u.base], eax ; 0, start of user's core (virtual)
17118 <1> ; 25/06/2015
17119 00004C2F 66A1[62740000] <1> mov ax, [ii]
17120 <1> ; AX = i-number of the executable file
17121 00004C35 E8C00C0000 <1> call readi
17122 <1> ; jsr r0,readi / read in first six words of
17123 <1> ; / user's file, starting at $core
17124 <1> ; mov sp,r5 / put users stack address in r5
17125 <1> ; sub $core+40.,r5 / subtract $core +40,
17126 <1> ; / from r5 (leaves number of words
17127 <1> ; / less 26 available for
17128 <1> ; / program in user core
17129 <1> ; mov r5,u.count /
17130 <1> ; 25/06/2015
17131 00004C3A 8B0D[AC740000] <1> mov ecx, [u.break] ; top of user's stack (physical addr.)
17132 00004C40 890D[A4740000] <1> mov [u.count], ecx ; save for overrun check
17133 <1> ;
17134 00004C46 8B0D[A8740000] <1> mov ecx, [u.nread]
17135 00004C4C 890D[AC740000] <1> mov [u.break], ecx ; virtual address (offset from start)
17136 00004C52 80F920 <1> cmp cl, 32
17137 00004C55 7540 <1> jne short sysexec_15
17138 <1> ;:
17139 <1> ; 25/06/2015
17140 <1> ; Retro UNIX 386 v1 (32 bit) executable file header format
17141 <1> ; 18/10/2015
17142 00004C57 8B35[E8740000] <1> mov esi, [pcore] ; start address of user's core memory
17143 <1> ; (phys. start addr. of the exec. file)
17144 00004C5D AD <1> lodsd
17145 00004C5E 663DEB1E <1> cmp ax, 1EEBh ; EBH, 1Eh -> jump to +32
17146 00004C62 7533 <1> jne short sysexec_15
17147 <1> ; cmp core,$405 / br .+14 is first instruction
17148 <1> ; / if file is standard a.out format
17149 <1> ; bne 1f / branch, if not standard format
17150 00004C64 AD <1> lodsd
17151 00004C65 89C1 <1> mov ecx, eax ; text (code) section size
17152 00004C67 AD <1> lodsd
17153 00004C68 01C1 <1> add ecx, eax ; + data section size (initialized data)
17154 <1> ; mov core+2,r5 / put 2nd word of users program in r5;
17155 <1> ; / number of bytes in program text
17156 <1> ; sub $14,r5 / subtract 12
17157 00004C6A 89CB <1> mov ebx, ecx
17158 <1> ;
17159 <1> ; 25/06/2015
17160 <1> ; NOTE: These are for next versions of Retro UNIX 386
17161 <1> ; and SINGLIX operating systems (as code template).
17162 <1> ; Current Retro UNIX 386 v1 files can be max. 64KB
17163 <1> ; due to RUFS (floppy disk file system) restriction...
17164 <1> ; Overrun is not possible for current version.
17165 <1> ;
17166 00004C6C AD <1> lodsd
17167 00004C6D 01C3 <1> add ebx, eax ; + bss section size (for overrun checking)
17168 00004C6F 3B1D[A4740000] <1> cmp ebx, [u.count]
17169 00004C75 7711 <1> ja short sysexec_14 ; program overruns stack !
17170 <1> ;
17171 <1> ; 24/07/2015
17172 <1> ; add bss section size to [u.break]
17173 00004C77 0105[AC740000] <1> add [u.break], eax
17174 <1> ;
17175 00004C7D 83E920 <1> sub ecx, 32 ; header size (already loaded)

```

```

17176      <1> ;cmp ecx, [u.count]
17177      <1> ;jnb short sysexec_16
17178      <1> ; cmp r5,u.count /
17179      <1> ; bgt 1f / branch if r5 greater than u.count
17180 00004C80 890D[A4740000] <1> mov [u.count], ecx ; required read count
17181      <1> ; mov r5,u.count
17182      <1> ;
17183 00004C86 EB2A <1> jmp short sysexec_16
17184      <1> ;
17185      <1> sysexec_14:
17186      <1> ; 23/06/2015
17187      <1> ; insufficient (out of) memory
17188 00004C88 C705[CF740000]0100- <1> mov dword [u.error], ERR_MINOR_IM ; 1
17189 00004C90 0000 <1>
17190 00004C92 E9E5F3FFFF <1> jmp error
17191      <1> ;
17192      <1> sysexec_15:
17193      <1> ; 25/06/2015
17194 00004C97 0FB715[5A710000] <1> movzx edx, word [i.size] ; file size
17195 00004C9E 29CA <1> sub edx, ecx ; file size - loaded bytes
17196 00004CA0 7627 <1> jna short sysexec_17 ; no need to next read
17197 00004CA2 01D1 <1> add ecx, edx ; [i.size]
17198 00004CA4 3B0D[A4740000] <1> cmp ecx, [u.count] ; overrun check (!)
17199 00004CAA 77DC <1> ja short sysexec_14
17200 00004CAC 8915[A4740000] <1> mov [u.count], edx
17201      <1> sysexec_16:
17202 00004CB2 66A1[62740000] <1> mov ax, [ii] ; i-number
17203 00004CB8 E83D0C0000 <1> call readi
17204      <1> ; add core+10,u.nread / add size of user data area
17205      <1> ; / to u.nread
17206      <1> ; br 2f
17207      <1> ; 1:
17208      <1> ; jsr r0,readi / read in rest of file
17209      <1> ; 2:
17210 00004CBD 8B0D[A8740000] <1> mov ecx, [u.nread]
17211 00004CC3 010D[AC740000] <1> add [u.break], ecx
17212      <1> ; mov u.nread,u.break / set users program break to end of
17213      <1> ; / user code
17214      <1> ; add $core+14,u.break / plus data area
17215      <1> sysexec_17: ; 20/07/2015
17216      <1> ;mov ax, [ii] ;rgc i-number
17217 00004CC9 E8F8120000 <1> call iclose
17218      <1> ; jsr r0,iclose / does nothing
17219 00004CCE 31C0 <1> xor eax, eax
17220 00004CD0 FEC0 <1> inc al
17221 00004CD2 66A3[BE740000] <1> mov [u.intr], ax ; 1 (interrupt/time-out is enabled)
17222 00004CD8 66A3[C0740000] <1> mov [u.quit], ax ; 1 ('ctrl+brk' signal is enabled)
17223      <1> ; 02/07/2015
17224 00004CDE 833D[D7740000]00 <1> cmp dword [u.ppgdir], 0 ; is the caller sys_init (kernel) ?
17225 00004CE5 770C <1> ja short sysexec_18 ; no, the caller is user process
17226      <1> ; If the caller is kernel (sys_init), 'sysexec' will come here
17227 00004CE7 8B15[A8700000] <1> mov edx, [k_page_dir] ; kernel's page directory
17228 00004CED 8915[D7740000] <1> mov [u.ppgdir], edx ; next time 'sysexec' must not come here
17229      <1> sysexec_18:
17230      <1> ; 18/10/2015
17231      <1> ; 05/08/2015
17232      <1> ; 29/07/2015
17233 00004CF3 8B2D[F8740000] <1> mov ebp, [argv] ; user's stack pointer must point to argument
17234      <1> ; list pointers (argument count)
17235 00004CF9 FA <1> cli
17236 00004CFA 8B25[44700000] <1> mov esp, [tss.esp0] ; ring 0 (kernel) stack pointer
17237      <1> ;mov esp, [u.sp] ; Restore Kernel stack
17238      <1> ; for this process
17239      <1> ;add esp, 20 ; --> EIP, CS, EFLAGS, ESP, SS
17240      <1> ;xor eax, eax ; 0
17241 00004D00 FEC8 <1> dec al ; eax = 0
17242 00004D02 66BA2300 <1> mov dx, UDATA
17243 00004D06 6652 <1> push dx ; user's stack segment
17244 00004D08 55 <1> push ebp ; user's stack pointer
17245      <1> ; (points to number of arguments)
17246 00004D09 FB <1> sti
17247 00004D0A 9C <1> pushfd ; EFLAGS
17248      <1> ; Set IF for enabling interrupts in user mode
17249      <1> ;or dword [esp], 200h
17250      <1> ;
17251      <1> ;mov bx, UCODE
17252      <1> ;push bx ; user's code segment
17253 00004D0B 6A1B <1> push UCODE
17254      <1> ;push 0
17255 00004D0D 50 <1> push eax ; EIP (=0) - start address -
17256      <1> ; clr -(r5) / popped into ps when rti in
17257      <1> ; / sysrele is executed
17258      <1> ; mov $core,-(r5) / popped into pc when rti
17259      <1> ; / in sysrele is executed
17260      <1> ;mov r5,0f / load second copyz argument
17261      <1> ;tst -(r5) / decrement r5
17262 00004D0E 8925[78740000] <1> mov [u.sp], esp ; 29/07/2015
17263      <1> ; 05/08/2015
17264      <1> ; Remedy of a General Protection Fault during 'iretd' is here !
17265      <1> ; ('push dx' would cause to general protection fault,
17266      <1> ; after 'pop ds' etc.)
17267      <1> ;

```

17268		<1>	;; push dx ; ds (UDATA)
17269		<1>	;; push dx ; es (UDATA)
17270		<1>	;; push dx ; fs (UDATA)
17271		<1>	;; push dx ; gs (UDATA)
17272		<1>	;
17273		<1>	; This is a trick to prevent general protection fault
17274		<1>	; during 'iretd' intruction at the end of 'sysrele' (in ul.s):
17275	00004D14 8EC2	<1>	mov es, dx ; UDATA
17276	00004D16 06	<1>	push es ; ds (UDATA)
17277	00004D17 06	<1>	push es ; es (UDATA)
17278	00004D18 06	<1>	push es ; fs (UDATA)
17279	00004D19 06	<1>	push es ; gs (UDATA)
17280	00004D1A 66BA1000	<1>	mov dx, KDATA
17281	00004D1E 8EC2	<1>	mov es, dx
17282		<1>	;
17283		<1>	;; pushad simulation
17284	00004D20 89E5	<1>	mov ebp, esp ; esp before pushad
17285	00004D22 50	<1>	push eax ; eax (0)
17286	00004D23 50	<1>	push eax ; ecx (0)
17287	00004D24 50	<1>	push eax ; edx (0)
17288	00004D25 50	<1>	push eax ; ebx (0)
17289	00004D26 55	<1>	push ebp ; esp before pushad
17290	00004D27 50	<1>	push eax ; ebp (0)
17291	00004D28 50	<1>	push eax ; esi (0)
17292	00004D29 50	<1>	push eax ; edi (0)
17293		<1>	;
17294	00004D2A A3[80740000]	<1>	mov [u.r0], eax ; eax = 0
17295	00004D2F 8925[7C740000]	<1>	mov [u.usp], esp
17296		<1>	; mov r5,u.r0 /
17297		<1>	; sub \$16.,r5 / skip 8 words
17298		<1>	; mov r5,u.sp / assign user stack pointer value,
17299		<1>	; / effectively zeroes all regs
17300		<1>	; / when sysrele is executed
17301		<1>	; jsr r0,copyz; core; 0:0 / zero user's core
17302		<1>	; clr u.break
17303		<1>	; mov r5,sp / point sp to user's stack
17304		<1>	;
17305	00004D35 E965F3FFFF	<1>	jmp sysret0
17306		<1>	;jmp sysret
17307		<1>	; br sysret3 / return to core image at \$core
17308		<1>	
17309		<1>	get_argp:
17310		<1>	; 18/10/2015 (nbase, ncount)
17311		<1>	; 21/07/2015
17312		<1>	; 24/06/2015 (Retro UNIX 386 v1)
17313		<1>	; Get (virtual) address of argument from user's core memory
17314		<1>	;
17315		<1>	; INPUT:
17316		<1>	; esi = virtual address of argument pointer
17317		<1>	; OUTPUT:
17318		<1>	; eax = virtual address of argument
17319		<1>	;
17320		<1>	; Modified registers: EAX, EBX, ECX, EDX, ESI
17321		<1>	;
17322	00004D3A 833D[D7740000]00	<1>	cmp dword [u.ppgdir], 0 ; /etc/init ?
17323		<1>	; (the caller is kernel)
17324	00004D41 7667	<1>	jna short get_argpk
17325		<1>	;
17326	00004D43 89F3	<1>	mov ebx, esi
17327	00004D45 E880E9FFFF	<1>	call get_physical_addr ; get physical address
17328	00004D4A 0F8289000000	<1>	jc get_argp_err
17329	00004D50 A3[F0740000]	<1>	mov [nbase], eax ; physical address
17330	00004D55 66890D[F4740000]	<1>	mov [ncount], cx ; remain byte count in page (1-4096)
17331	00004D5C B804000000	<1>	mov eax, 4 ; 21/07/2015
17332	00004D61 6639C1	<1>	cmp cx, ax ; 4
17333	00004D64 735D	<1>	jnb short get_argp2
17334	00004D66 89F3	<1>	mov ebx, esi
17335	00004D68 01CB	<1>	add ebx, ecx
17336	00004D6A E85BE9FFFF	<1>	call get_physical_addr ; get physical address
17337	00004D6F 7268	<1>	jc short get_argp_err
17338		<1>	;push esi
17339	00004D71 89C6	<1>	mov esi, eax
17340	00004D73 66870D[F4740000]	<1>	xchg cx, [ncount]
17341	00004D7A 8735[F0740000]	<1>	xchg esi, [nbase]
17342	00004D80 B504	<1>	mov ch, 4
17343	00004D82 28CD	<1>	sub ch, cl
17344		<1>	get_argp0:
17345	00004D84 AC	<1>	lodsb
17346	00004D85 6650	<1>	push ax
17347	00004D87 FEC9	<1>	dec cl
17348	00004D89 75F9	<1>	jnz short get_argp0
17349	00004D8B 8B35[F0740000]	<1>	mov esi, [nbase]
17350		<1>	; 21/07/2015
17351	00004D91 0FB6C5	<1>	movzx eax, ch
17352	00004D94 0105[F0740000]	<1>	add [nbase], eax
17353	00004D9A 662905[F4740000]	<1>	sub [ncount], ax
17354		<1>	get_argp1:
17355	00004DA1 AC	<1>	lodsb
17356	00004DA2 FECD	<1>	dec ch
17357	00004DA4 743D	<1>	jz short get_argp3
17358	00004DA6 6650	<1>	push ax
17359	00004DA8 EBF7	<1>	jmp short get_argp1

```

17360                                     <1> get_argpk:
17361                                     <1> ; Argument is in kernel's memory space
17362 00004DAA 66C705[F4740000]00- <1> mov word [ncount], PAGE_SIZE ; 4096
17363 00004DB2 10 <1>
17364 00004DB3 8935[F0740000] <1> mov [nbase], esi
17365 00004DB9 8305[F0740000]04 <1> add dword [nbase], 4
17366 00004DC0 8B06 <1> mov eax, [esi] ; virtual addr. = physcal addr.
17367 00004DC2 C3 <1> retn
17368 <1> get_argp2:
17369 <1> ; 21/07/2015
17370 <1> ;mov eax, 4
17371 00004DC3 8B15[F0740000] <1> mov edx, [nbase] ; 18/10/2015
17372 00004DC9 0105[F0740000] <1> add [nbase], eax
17373 00004DCF 662905[F4740000] <1> sub [ncount], ax
17374 <1> ;
17375 00004DD6 8B02 <1> mov eax, [edx]
17376 00004DD8 C3 <1> retn
17377 <1> get_argp_err:
17378 00004DD9 A3[CF740000] <1> mov [u.error], eax
17379 00004DDE E999F2FFFF <1> jmp error
17380 <1> get_argp3:
17381 00004DE3 B103 <1> mov cl, 3
17382 <1> get_argp4:
17383 00004DE5 C1E008 <1> shl eax, 8
17384 00004DE8 665A <1> pop dx
17385 00004DEA 88D0 <1> mov al, dl
17386 00004DEC E2F7 <1> loop get_argp4
17387 <1> ;pop esi
17388 00004DEE C3 <1> retn
17389 <1>
17390 <1> sysfstat:
17391 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
17392 <1> ; 19/06/2013 (Retro UNIX 8086 v1)
17393 <1> ;
17394 <1> ; 'sysfstat' is identical to 'sysstat' except that it operates
17395 <1> ; on open files instead of files given by name. It puts the
17396 <1> ; buffer address on the stack, gets the i-number and
17397 <1> ; checks to see if the file is open for reading or writing.
17398 <1> ; If the file is open for writing (i-number is negative)
17399 <1> ; the i-number is set positive and a branch into 'sysstat'
17400 <1> ; is made.
17401 <1> ;
17402 <1> ; Calling sequence:
17403 <1> ; sysfstat; buf
17404 <1> ; Arguments:
17405 <1> ; buf - buffer address
17406 <1> ;
17407 <1> ; Inputs: *u.r0 - file descriptor
17408 <1> ; Outputs: buffer is loaded with file information
17409 <1> ; .....
17410 <1> ;
17411 <1> ; Retro UNIX 8086 v1 modification:
17412 <1> ; 'sysfstat' system call has two arguments; so,
17413 <1> ; * 1st argument, file descriptor is in BX register
17414 <1> ; * 2nd argument, buf is pointed to by CX register
17415 <1> ;
17416 <1> ; / set status of open file
17417 <1> ; jsr r0,arg; u.off / put buffer address in u.off
17418 00004DEF 51 <1> push ecx
17419 <1> ; mov u.off, -(sp) / put buffer address on the stack
17420 <1> ; mov *u.r0, r1 / put file descriptor in r1
17421 <1> ; jsr r0, getf / get the files i-number
17422 <1> ; BX = file descriptor (file number)
17423 00004DF0 E8FF000000 <1> call getfl
17424 00004DF5 6621C0 <1> and ax, ax ; i-number of the file
17425 <1> ; tst r1 / is it 0?
17426 <1> ;jz error
17427 <1> ; beq error3 / yes, error
17428 00004DF8 750F <1> jnz short sysfstat1
17429 00004DFA C705[CF740000]0A00- <1> mov dword [u.error], ERR_FILE_NOT_OPEN ; 'file not open !'
17430 00004E02 0000 <1>
17431 00004E04 E973F2FFFF <1> jmp error
17432 <1> sysfstat1:
17433 00004E09 80FC80 <1> cmp ah, 80h
17434 00004E0C 7223 <1> jb short sysstat1
17435 <1> ; bgt 1f / if i-number is negative (open for writing)
17436 00004E0E 66F7D8 <1> neg ax
17437 <1> ; neg r1 / make it positive, then branch
17438 00004E11 EB1E <1> jmp short sysstat1
17439 <1> ; br 1f / to 1f
17440 <1> sysstat:
17441 <1> ; 18/10/2015
17442 <1> ; 07/10/2015
17443 <1> ; 02/09/2015
17444 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
17445 <1> ; 19/06/2013 (Retro UNIX 8086 v1)
17446 <1> ;
17447 <1> ; 'sysstat' gets the status of a file. Its arguments are the
17448 <1> ; name of the file and buffer address. The buffer is 34 bytes
17449 <1> ; long and information about the file placed in it.
17450 <1> ; sysstat calls 'namei' to get the i-number of the file.
17451 <1> ; Then 'iget' is called to get i-node in core. The buffer

```

```

17452      <1> ; is then loaded and the results are given in the UNIX
17453      <1> ; Programmers Manual sysstat (II).
17454      <1> ;
17455      <1> ; Calling sequence:
17456      <1> ;     sysstat; name; buf
17457      <1> ; Arguments:
17458      <1> ;     name - points to the name of the file
17459      <1> ;     buf - address of a 34 bytes buffer
17460      <1> ; Inputs: -
17461      <1> ; Outputs: buffer is loaded with file information
17462      <1> ; .....
17463      <1> ;
17464      <1> ; Retro UNIX 8086 v1 modification:
17465      <1> ;     'sysstat' system call has two arguments; so,
17466      <1> ;     Retro UNIX 8086 v1 argument transfer method 2 is used
17467      <1> ;     to get sysstat system call arguments from the user;
17468      <1> ;     * 1st argument, name is pointed to by BX register
17469      <1> ;     * 2nd argument, buf is pointed to by CX register
17470      <1> ;
17471      <1> ;     NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
17472      <1> ;     arguments which were in these registers;
17473      <1> ;     but, it returns by putting the 1st argument
17474      <1> ;     in 'u.namep' and the 2nd argument
17475      <1> ;     on top of stack. (1st argument is offset of the
17476      <1> ;     file/path name in the user's program segment.)
17477      <1> ;
17478      <1> ; / ; name of file; buffer - get files status
17479      <1> ;     jsr r0,arg2 / get the 2 arguments
17480 00004E13 891D[98740000] <1> mov     [u.namep], ebx
17481 00004E19 51 <1> push    ecx
17482 00004E1A E80C010000 <1> call    namei
17483 <1> ;     jsr r0,namei / get the i-number for the file
17484 <1> ;jc     error
17485 <1> ;     br error3 / no such file, error
17486 00004E1F 7310 <1> jnc     short sysstat1
17487 <1> ; pop     ecx
17488 <1> sysstat_err0:
17489 <1> ; 'file not found !' error
17490 00004E21 C705[CF740000]0C00- <1> mov     dword [u.error], ERR_FILE_NOT_FOUND ; 12
17491 00004E29 0000 <1>
17492 00004E2B E94CF2FFFF <1> jmp     error
17493 <1>
17494 00004E30 00 <1> statx: db 0
17495 <1>
17496 <1> sysstat1: ; 1:
17497 00004E31 E8E4080000 <1> call    iget
17498 <1> ;     jsr r0,iget / get the i-node into core
17499 <1> ; 07/10/2015 (ax = [ii], inode number)
17500 <1> ; 02/09/2015
17501 00004E36 8F05[A0740000] <1> pop     dword [u.base]
17502 <1> ;     mov (sp)+,r3 / move u.off to r3 (points to buffer)
17503 00004E3C E858000000 <1> call    sysstat_gpa ; get physical address
17504 00004E41 730A <1> jnc     short sysstat2
17505 <1> sysstat_err1:
17506 00004E43 A3[CF740000] <1> mov     dword [u.error], eax ; error code
17507 00004E48 E92FF2FFFF <1> jmp     error
17508 <1> sysstat2:
17509 00004E4D A0[62740000] <1> mov     al, [ii] ; 07/10/2015 (result of 'iget' call, above)
17510 00004E52 AA <1> stosb
17511 00004E53 FF05[A0740000] <1> inc     dword [u.base]
17512 00004E59 6649 <1> dec     cx
17513 00004E5B 7505 <1> jnz     short sysstat3
17514 00004E5D E837000000 <1> call    sysstat_gpa
17515 <1> ;jc     short sysstat_err1
17516 <1> sysstat3:
17517 00004E62 A0[63740000] <1> mov     al, [ii+1] ; 07/10/2015 (result of 'iget' call, above)
17518 00004E67 AA <1> stosb
17519 <1> ;     mov r1,(r3)+ / put i-number in 1st word of buffer
17520 00004E68 FF05[A0740000] <1> inc     dword [u.base]
17521 <1> ;dec    word [u.pcount]
17522 00004E6E 6649 <1> dec     cx
17523 00004E70 7505 <1> jnz     short sysstat4
17524 00004E72 E822000000 <1> call    sysstat_gpa
17525 <1> ;jc     short sysstat_err1
17526 <1> sysstat4:
17527 00004E77 BE[56710000] <1> mov     esi, inode
17528 <1> ;     mov $inode,r2 / r2 points to i-node
17529 <1> sysstat5: ; 1:
17530 00004E7C A4 <1> movsb
17531 <1> ;     mov (r2)+,(r3)+ / move rest of i-node to buffer
17532 00004E7D FF05[A0740000] <1> inc     dword [u.base]
17533 <1> ;dec    word [u.pcount]
17534 00004E83 6649 <1> dec     cx
17535 00004E85 7505 <1> jnz     short sysstat6
17536 00004E87 E80D000000 <1> call    sysstat_gpa
17537 <1> ;jc     short sysstat_err1
17538 <1> sysstat6:
17539 00004E8C 81FE[76710000] <1> cmp     esi, inode + 32
17540 <1> ;     cmp r2,$inode+32 / done?
17541 00004E92 75E8 <1> jne     short sysstat5
17542 <1> ;     bne 1b / no, go back
17543 00004E94 E903F2FFFF <1> jmp     sysret

```

17544		<1>	; br sysret3 / return through sysret
17545		<1>	;
17546		<1>	sysstat_gpa: ; get physical address of file status buffer
17547		<1>	; 02/09/2015
17548	00004E99 8B1D[A0740000]	<1>	mov ebx, [u.base]
17549		<1>	; 07/10/2015
17550	00004E9F E826E8FFFF	<1>	call get_physical_addr ; get physical address
17551		<1>	;jc short sysstat_gpal
17552	00004EA4 729D	<1>	jc short sysstat_err1
17553		<1>	; 18/10/2015
17554	00004EA6 89C7	<1>	mov edi, eax ; physical address
17555		<1>	;mov [u.pcount], cx ; remain bytes in page
17556		<1>	;sysstat_gpal:
17557	00004EA8 C3	<1>	retn
17558		<1>	
17559		<1>	fclose:
17560		<1>	; 18/06/2015 (Retro UNIX 386 v1 - Beginning)
17561		<1>	; (32 bit offset pointer modification)
17562		<1>	; 19/04/2013 - 12/01/2014 (Retro UNIX 8086 v1)
17563		<1>	;
17564		<1>	; Given the file descriptor (index to the u.fp list)
17565		<1>	; 'fclose' first gets the i-number of the file via 'getf'.
17566		<1>	; If i-node is active (i-number > 0) the entry in
17567		<1>	; u.fp list is cleared. If all the processes that opened
17568		<1>	; that file close it, then fsp entry is freed and the file
17569		<1>	; is closed. If not a return is taken.
17570		<1>	; If the file has been deleted while open, 'anyi' is called
17571		<1>	; to see anyone else has it open, i.e., see if it is appears
17572		<1>	; in another entry in the fsp table. Upon return from 'anyi'
17573		<1>	; a check is made to see if the file is special.
17574		<1>	;
17575		<1>	; INPUTS ->
17576		<1>	; r1 - contains the file descriptor (value=0,1,2...)
17577		<1>	; u.fp - list of entries in the fsp table
17578		<1>	; fsp - table of entries (4 words/entry) of open files.
17579		<1>	; OUTPUTS ->
17580		<1>	; r1 - contains the same file descriptor
17581		<1>	; r2 - contains i-number
17582		<1>	;
17583		<1>	; ((AX = R1))
17584		<1>	; ((Modified registers: edx, ebx, ecx, esi, edi, ebp))
17585		<1>	;
17586		<1>	; Retro UNIX 8086 v1 modification : CF = 1
17587		<1>	; if i-number of the file is 0. (error)
17588		<1>	;
17589	00004EA9 0FB7D0	<1>	movzx edx, ax ; **
17590	00004EAC 6650	<1>	push ax ; ***
17591		<1>	; mov r1, -(sp) / put r1 on the stack (it contains
17592		<1>	; / the index to u.fp list)
17593	00004EAE E83F000000	<1>	call getf
17594		<1>	; jsr r0, getf / r1 contains i-number,
17595		<1>	; / cdev has device =, u.fofp
17596		<1>	; / points to 3rd word of fsp entry
17597	00004EB3 6683F801	<1>	cmp ax, 1 ; r1
17598		<1>	; tst r1 / is i-number 0?
17599	00004EB7 7236	<1>	jb short fclose_2
17600		<1>	; beq 1f / yes, i-node not active so return
17601		<1>	; tst (r0)+ / no, jump over error return
17602	00004EB9 89D3	<1>	mov ebx, edx ; **
17603	00004EBB 6689C2	<1>	mov dx, ax ; *
17604		<1>	; mov r1, r2 / move i-number to r2 ; *
17605		<1>	; mov (sp), r1 / restore value of r1 from the stack
17606		<1>	; / which is index to u.fp ; **
17607	00004EBE C683[86740000]00	<1>	mov byte [ebx+u.fp], 0
17608		<1>	; clrb u.fp(r1) / clear that entry in the u.fp list
17609	00004EC5 8B1D[90740000]	<1>	mov ebx, [u.fofp]
17610		<1>	; mov u.fofp, r1 / r1 points to 3rd word in fsp entry
17611		<1>	fclose_0:
17612	00004ECB FE4B04	<1>	dec byte [ebx+4] ; 18/06/2015
17613		<1>	; decb 2(r1) / decrement the number of processes
17614		<1>	; / that have opened the file
17615	00004ECE 791F	<1>	jns short fclose_2 ; jump if not negative (jump if bit 7 is 0)
17616		<1>	; bge 1f / if all processes haven't closed the file, return
17617		<1>	;
17618	00004ED0 6652	<1>	push dx ; *
17619		<1>	; mov r2, -(sp) / put r2 on the stack (i-number)
17620	00004ED2 6631C0	<1>	xor ax, ax ; 0
17621	00004ED5 668943FC	<1>	mov [ebx-4], ax ; 0
17622		<1>	; clr -4(r1) / clear 1st word of fsp entry
17623	00004ED9 8A4305	<1>	mov al, [ebx+5] ; 18/06/2015
17624		<1>	; tstb 3(r1) / has this file been deleted
17625	00004EDC 20C0	<1>	and al, al
17626	00004EDE 7408	<1>	jz short fclose_1
17627		<1>	; beq 2f / no, branch
17628	00004EE0 6689D0	<1>	mov ax, dx ; *
17629		<1>	; mov r2, r1 / yes, put i-number back into r1
17630		<1>	; AX = inode number
17631	00004EE3 E867040000	<1>	call anyi
17632		<1>	; jsr r0, anyi / free all blocks related to i-number
17633		<1>	; / check if file appears in fsp again
17634		<1>	fclose_1: ; 2:
17635	00004EE8 6658	<1>	pop ax ; *



17636		<1>	; mov (sp)+,r1 / put i-number back into r1
17637	00004EEA E8D7100000	<1>	call iclose ; close if it is special file
17638		<1>	; jsr r0,iclose / check to see if its a special file
17639		<1>	fclose_2: ; 1:
17640	00004EEF 6658	<1>	pop ax ; ***
17641		<1>	; mov (sp)+,r1 / put index to u.fp back into r1
17642	00004EF1 C3	<1>	retn
17643		<1>	; rts r0
17644		<1>	
17645		<1>	getf: ; / get the device number and the i-number of an open file
17646		<1>	; 13/05/2015
17647		<1>	; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
17648		<1>	; 19/04/2013 - 18/11/2013 (Retro UNIX 8086 v1)
17649		<1>	;
17650	00004EF2 89C3	<1>	mov ebx, eax
17651		<1>	getf1: ;; Calling point from 'rw1' (23/05/2013)
17652	00004EF4 83FB0A	<1>	cmp ebx, 10
17653		<1>	; cmp r1,\$10. / user limited to 10 open files
17654	00004EF7 730A	<1>	jnb short getf2 ; 13/05/2015
17655		<1>	;jnb error
17656		<1>	; bhis error3 / u.fp is table of users open files,
17657		<1>	; / index in fsp table
17658	00004EF9 8A9B[86740000]	<1>	mov bl, [ebx+u.fp]
17659		<1>	; movb u.fp(r1),r1 / r1 contains number of entry
17660		<1>	; / in fsp table
17661	00004EFF 08DB	<1>	or bl, bl
17662	00004F01 7503	<1>	jnz short getf3
17663		<1>	;jz short getf4
17664		<1>	; beq lf / if its zero return
17665		<1>	getf2:
17666		<1>	; 'File not open !' error (ax=0)
17667	00004F03 29C0	<1>	sub eax, eax
17668	00004F05 C3	<1>	retn
17669		<1>	getf3:
17670		<1>	; Retro UNIX 386 v1 modification ! (11/05/2015)
17671		<1>	;
17672		<1>	; 'fsp' table (10 bytes/entry)
17673		<1>	; bit 15 bit 0
17674		<1>	; --- -----
17675		<1>	; r/w  i-number of open file
17676		<1>	; --- -----
17677		<1>	; device number
17678		<1>	; -----
17679		<1>	; offset pointer, r/w pointer to file (bit 0-15)
17680		<1>	; -----
17681		<1>	; offset pointer, r/w pointer to file (bit 16-31)
17682		<1>	; ----- -----
17683		<1>	; flag that says file   number of processes
17684		<1>	; has been deleted   that have file open
17685		<1>	; ----- -----
17686		<1>	;
17687	00004F06 B80A000000	<1>	mov eax, 10
17688	00004F0B F6E3	<1>	mul bl
17689	00004F0D BB[50720000]	<1>	mov ebx, fsp - 6 ; the 3rd word in the fsp entry
17690	00004F12 01C3	<1>	add ebx, eax
17691		<1>	; asl r1
17692		<1>	; asl r1 / multiply by 8 to get index into
17693		<1>	; / fsp table entry
17694		<1>	; asl r1
17695		<1>	; add \$fsp-4,r1 / r1 is pointing at the 3rd word
17696		<1>	; / in the fsp entry
17697	00004F14 891D[90740000]	<1>	mov [u.fofp], ebx
17698		<1>	; mov r1,u.fofp / save address of 3rd word
17699		<1>	; / in fsp entry in u.fofp
17700	00004F1A 4B	<1>	dec ebx
17701	00004F1B 4B	<1>	dec ebx
17702	00004F1C 668B03	<1>	mov ax, [ebx]
17703		<1>	;mov [cdev], al ; ;;Retro UNIX 8086 v1 !
17704	00004F1F 66A3[66740000]	<1>	mov [cdev], ax ; ;;in fact (!)
17705		<1>	;dev number is in 1 byte
17706		<1>	; mov -(r1),cdev / remove the device number cdev
17707	00004F25 4B	<1>	dec ebx
17708	00004F26 4B	<1>	dec ebx
17709	00004F27 668B03	<1>	mov ax, [ebx]
17710		<1>	; mov -(r1),r1 / and the i-number r1
17711		<1>	getf4: ; 1:
17712	00004F2A C3	<1>	retn
17713		<1>	; rts r0
17714		<1>	
17715		<1>	namei:
17716		<1>	; 18/10/2015 (nbase, ncount)
17717		<1>	; 12/10/2015
17718		<1>	; 21/08/2015
17719		<1>	; 18/07/2015
17720		<1>	; 02/07/2015
17721		<1>	; 17/06/2015
17722		<1>	; 16/06/2015 (Retro UNIX 386 v1 - Beginning)
17723		<1>	; 24/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
17724		<1>	;
17725		<1>	; 'namei' takes a file path name and returns i-number of
17726		<1>	; the file in the current directory or the root directory
17727		<1>	; (if the first character of the pathname is '/').

```

17728 <1> ;
17729 <1> ; INPUTS ->
17730 <1> ; u.namep - points to a file path name
17731 <1> ; u.cdir - i-number of users directory
17732 <1> ; u.cdev - device number on which user directory resides
17733 <1> ; OUTPUTS ->
17734 <1> ; r1 - i-number of file
17735 <1> ; cdev
17736 <1> ; u.dirbuf - points to directory entry where a match
17737 <1> ; occurs in the search for file path name.
17738 <1> ; If no match u.dirb points to the end of
17739 <1> ; the directory and r1 = i-number of the current
17740 <1> ; directory.
17741 <1> ; ((AX = R1))
17742 <1> ;
17743 <1> ; (Retro UNIX Prototype : 07/10/2012 - 05/01/2013, UNIXCOPY.ASM)
17744 <1> ; ((Modified registers: EDX, EBX, ECX, ESI, EDI, EBP))
17745 <1> ;
17746 <1> ;
17747 00004F2B 66A1[84740000] <1> mov ax, [u.cdir]
17748 <1> ; mov u.cdir,r1 / put the i-number of current directory
17749 <1> ; / in r1
17750 00004F31 668B15[C4740000] <1> mov dx, [u.cdrv]
17751 00004F38 668915[66740000] <1> mov [cdev], dx ; NOTE: Retro UNIX 8086 v1
17752 <1> ; device/drive number is in 1 byte,
17753 <1> ; not in 1 word!
17754 <1> ; mov u.cdev,cdev / device number for users directory
17755 <1> ; / into cdev
17756 <1> ; 12/10/2015
17757 <1> ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
17758 <1> ; convert virtual (pathname) addr to physical address
17759 00004F3F E82C010000 <1> call trans_addr_nmbp ; 12/10/2015
17760 <1> ; esi = physical address of [u.namep]
17761 <1> ; ecx = byte count in the page
17762 00004F44 803E2F <1> cmp byte [esi], '/'
17763 <1> ; cmpb *u.namep,$'/ / is first char in file name a /
17764 00004F47 751E <1> jne short namei_1
17765 <1> ; bne 1f
17766 00004F49 FF05[98740000] <1> inc dword [u.namep]
17767 <1> ; inc u.namep / go to next char
17768 00004F4F 6649 <1> dec cx ; remain byte count in the page
17769 00004F51 7506 <1> jnz short namei_0
17770 <1> ; 12/10/2015
17771 00004F53 E818010000 <1> call trans_addr_nmbp ; convert virtual address to physical
17772 <1> ; esi = physical address (page start + offset)
17773 <1> ; ecx = byte count in the page
17774 00004F58 4E <1> dec esi
17775 <1> namei_0:
17776 00004F59 46 <1> inc esi ; go to next char
17777 00004F5A 66A1[70740000] <1> mov ax, [rootdir] ; 09/07/2013
17778 <1> ; mov rootdir,r1 / put i-number of rootdirectory in r1
17779 00004F60 C605[66740000]00 <1> mov byte [cdev], 0
17780 <1> ; clr cdev / clear device number
17781 <1> namei_1: ; 1:
17782 00004F67 F606FF <1> test byte [esi], 0FFh
17783 00004F6A 74BE <1> jz short getf4
17784 <1> ;jz nig
17785 <1> ; tstb *u.namep / is the character in file name a nul
17786 <1> ; beq nig / yes, end of file name reached;
17787 <1> ; / branch to "nig"
17788 <1> namei_2: ; 1:
17789 <1> ; 18/10/2015
17790 00004F6C 8935[F0740000] <1> mov [nbase], esi
17791 00004F72 66890D[F4740000] <1> mov [ncount], cx
17792 <1> ;
17793 <1> ;mov dx, 2
17794 00004F79 B202 <1> mov dl, 2 ; user flag (read, non-owner)
17795 00004F7B E875080000 <1> call access
17796 <1> ; jsr r0,access; 2 / get i-node with i-number r1
17797 <1> ; 'access' will not return here if user has not "r" permission !
17798 00004F80 66F705[56710000]00- <1> test word [i.flgs], 4000h
17799 00004F88 40 <1> ;
17800 <1> ; bit $40000,i.flgs / directory i-node?
17801 00004F89 746A <1> jz short namei_err
17802 <1> ; beq error3 / no, got an error
17803 <1> ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
17804 00004F8B 31C0 <1> xor eax, eax
17805 00004F8D A3[9C740000] <1> mov [u.off], eax ; 0
17806 00004F92 66A1[5A710000] <1> mov ax, [i.size]
17807 00004F98 A3[94740000] <1> mov [u.dirp], eax
17808 <1> ; mov i.size,u.dirp / put size of directory in u.dirp
17809 <1> ; clr u.off / u.off is file offset used by user
17810 00004F9D C705[90740000]- <1> mov dword [u.fofp], u.off
17811 00004FA3 [9C740000] <1> ;
17812 <1> ; mov $u.off,u.fofp / u.fofp is a pointer to
17813 <1> ; / the offset portion of fsp entry
17814 <1> namei_3: ; 2:
17815 00004FA7 C705[A0740000]- <1> mov dword [u.base], u.dirbuf
17816 00004FAD [B2740000] <1> ;
17817 <1> ; mov $u.dirbuf,u.base / u.dirbuf holds a file name
17818 <1> ; / copied from a directory
17819 00004FB1 C705[A4740000]0A00- <1> mov dword [u.count], 10

```

```

17820 00004FB9 0000      <1>
17821                      <1>      ; mov $10.,u.count / u.count is byte count
17822                      <1>      ; / for reads and writes
17823 00004FBB 66A1[62740000] <1> mov ax, [ii]
17824                      <1>      ; 31/07/2013 ('namei_r') - 16/06/2015 ('u.kcall')
17825 00004FC1 FE05[E1740000] <1> inc byte [u.kcall] ; the caller is 'namei' sign
17826 00004FC7 E82E090000 <1> call readi
17827                      <1>      ; jsr r0,readi / read 10. bytes of file
17828                      <1>      ; with i-number (r1); i.e. read a directory entry
17829 00004FCC 8B0D[A8740000] <1> mov ecx, [u.nread]
17830 00004FD2 09C9 <1> or ecx, ecx
17831                      <1>      ; tst u.nread
17832 00004FD4 741B <1> jz short nib
17833                      <1>      ; ble nib / gives error return
17834                      <1>      ;
17835 00004FD6 668B1D[B2740000] <1> mov bx, [u.dirbuf]
17836 00004FDD 6621DB <1> and bx, bx
17837                      <1>      ; tst u.dirbuf /
17838 00004FE0 7522 <1> jnz short namei_4
17839                      <1>      ; bne 3f / branch when active directory entry
17840                      <1>      ; / (i-node word in entry non zero)
17841 00004FE2 A1[9C740000] <1> mov eax, [u.off]
17842 00004FE7 83E80A <1> sub eax, 10
17843 00004FEA A3[94740000] <1> mov [u.dirp], eax
17844                      <1>      ; mov u.off,u.dirp
17845                      <1>      ; sub $10.,u.dirp
17846 00004FEF EBB6 <1> jmp short namei_3
17847                      <1>      ; br 2b
17848                      <1>
17849                      <1>      ; 18/07/2013
17850                      <1> nib:
17851 00004FF1 31C0 <1> xor eax, eax ; xor ax, ax ; ax = 0 -> file not found
17852 00004FF3 F9 <1> stc
17853                      <1> nig:
17854 00004FF4 C3 <1> retn
17855                      <1>
17856                      <1> namei_err:
17857                      <1>      ; 16/06/2015
17858 00004FF5 C705[CF740000]1300- <1> mov dword [u.error], ERR_NOT_DIR ; 'not a directory !' error
17859 00004FFD 0000 <1>
17860 00004FFF E978F0FFFF <1> jmp error
17861                      <1>
17862                      <1> namei_4: ; 3:
17863                      <1>      ; 18/10/2015
17864                      <1>      ; 12/10/2015
17865                      <1>      ; 21/08/2015
17866                      <1>      ; 18/07/2015
17867 00005004 8B2D[98740000] <1> mov ebp, [u.namep]
17868                      <1>      ; mov u.namep,r2 / u.namep points into a file name string
17869 0000500A BF[B4740000] <1> mov edi, u.dirbuf + 2
17870                      <1>      ; mov $u.dirbuf+2,r3 / points to file name of directory entry
17871                      <1>      ; 18/10/2015
17872 0000500F 8B35[F0740000] <1> mov esi, [nbase]
17873 00005015 668B0D[F4740000] <1> mov cx, [ncount]
17874                      <1>      ;
17875 0000501C 6621C9 <1> and cx, cx
17876 0000501F 7505 <1> jnz short namei_5
17877                      <1>      ;
17878 00005021 E850000000 <1> call trans_addr_nm ; convert virtual address to physical
17879                      <1>      ; esi = physical address (page start + offset)
17880                      <1>      ; ecx = byte count in the page
17881                      <1> namei_5: ; 3:
17882 00005026 45 <1> inc ebp ; 18/07/2015
17883 00005027 AC <1> lodsb ; mov al, [esi] ; inc esi (al = r4)
17884                      <1>      ; movb (r2)+,r4 / move a character from u.namep string into r4
17885 00005028 08C0 <1> or al, al
17886 0000502A 741D <1> jz short namei_7
17887                      <1>      ; beq 3f / if char is nul, then the last char in string
17888                      <1>      ; / has been moved
17889 0000502C 3C2F <1> cmp al, '/'
17890                      <1>      ; cmp r4,$' / is char a </>
17891 0000502E 7419 <1> je short namei_7
17892                      <1>      ; beq 3f
17893                      <1>      ; 12/10/2015
17894 00005030 6649 <1> dec cx ; remain byte count in the page
17895 00005032 7505 <1> jnz short namei_6
17896 00005034 E83D000000 <1> call trans_addr_nm ; convert virtual address to physical
17897                      <1>      ; esi = physical address (page start + offset)
17898                      <1>      ; ecx = byte count in the page
17899                      <1> namei_6:
17900 00005039 81FF[BC740000] <1> cmp edi, u.dirbuf + 10
17901                      <1>      ; cmp r3,$u.dirbuf+10. / have I checked
17902                      <1>      ; / all 8 bytes of file name
17903 0000503F 74E5 <1> je short namei_5
17904                      <1>      ; beq 3b
17905 00005041 AE <1> scasb
17906                      <1>      ; cmpb (r3)+,r4 / compare char in u.namep string to file name
17907                      <1>      ; / char read from directory
17908 00005042 74E2 <1> je short namei_5
17909                      <1>      ; beq 3b / branch if chars match
17910                      <1>
17911 00005044 E95EFFFFFF <1> jmp namei_3 ; 2b

```

17912		<1>		; br 2b / file names do not match go to next directory entry
17913		<1>	namei_7: ; 3:	
17914	00005049 81FF[BC740000]	<1>	cmp	edi, u.dirbuf + 10
17915		<1>		; cmp r3,\$u.dirbuf+10. / if equal all 8 bytes were matched
17916	0000504F 740A	<1>	je	short namei_8
17917		<1>		; beq 3f
17918	00005051 8A27	<1>	mov	ah, [edi]
17919		<1>	;inc	edi
17920	00005053 20E4	<1>	and	ah, ah
17921		<1>		; tstb (r3)+ /
17922	00005055 0F854CFFFFFF	<1>	jnz	namei_3
17923		<1>		; bne 2b
17924		<1>	namei_8: ; 3	
17925	0000505B 892D[98740000]	<1>	mov	[u.namep], ebp ; 18/07/2015
17926		<1>		; mov r2,u.namep / u.namep points to char
17927		<1>		; / following a / or nul
17928		<1>	;mov	bx, [u.dirbuf]
17929		<1>		; mov u.dirbuf,r1 / move i-node number in directory
17930		<1>		; / entry to r1
17931	00005061 20C0	<1>	and	al, al
17932		<1>		; tst r4 / if r4 = 0 the end of file name reached,
17933		<1>		; / if r4 = </> then go to next directory
17934		<1>		; mov ax, bx
17935	00005063 66A1[B2740000]	<1>	mov	ax, [u.dirbuf] ; 17/06/2015
17936	00005069 0F85FDFEFFFF	<1>	jnz	namei_2
17937		<1>		; bne 1b
17938		<1>		; AX = i-number of the file
17939		<1>	;;nig:	
17940	0000506F C3	<1>	retn	
17941		<1>		; tst (r0)+ / gives non-error return
17942		<1>	;;nib:	
17943		<1>	;;	xor ax, ax ; Retro UNIX 8086 v1 modification !
17944		<1>		; ax = 0 -> file not found
17945		<1>	;;	stc ; 27/05/2013
17946		<1>	;;	retn
17947		<1>		; rts r0
17948		<1>		
17949		<1>	trans_addr_nmbp:	
17950		<1>		; 18/10/2015
17951		<1>		; 12/10/2015
17952	00005070 8B2D[98740000]	<1>	mov	ebp, [u.namep]
17953		<1>	trans_addr_nm:	
17954		<1>		; Convert virtual (pathname) address to physical address
17955		<1>		; (Retro UNIX 386 v1 feature only !)
17956		<1>		; 18/10/2015
17957		<1>		; 12/10/2015 (u.pnbase & u.pncount has been removed from code)
17958		<1>		; 02/07/2015
17959		<1>		; 17/06/2015
17960		<1>		; 16/06/2015
17961		<1>		
17962		<1>	; INPUTS:	
17963		<1>		; ebp = pathname address (virtual) ; [u.namep]
17964		<1>		; [u.pgdir] = user's page directory
17965		<1>	; OUTPUT:	
17966		<1>		; esi = physical address of the pathname
17967		<1>		; ecx = remain byte count in the page
17968		<1>		
17969		<1>		; (Modified registers: EAX, EBX, ECX, EDX, ESI)
17970		<1>		
17971	00005076 833D[D7740000]00	<1>	cmp	dword [u.ppgdir], 0 ; /etc/init ? (sysexec)
17972	0000507D 7618	<1>	jna	short trans_addr_nmk ; the caller is os kernel;
17973		<1>		; it is already physical address
17974	0000507F 50	<1>	push	eax
17975	00005080 89EB	<1>	mov	ebx, ebp ; [u.namep] ; pathname address (virtual)
17976	00005082 E843E6FFFF	<1>	call	get_physical_addr ; get physical address
17977	00005087 7204	<1>	jc	short tr_addr_nm_err
17978		<1>		; 18/10/2015
17979		<1>		; eax = physical address
17980		<1>		; cx = remain byte count in page (1-4096)
17981		<1>		; 12/10/2015 (cx = [u.pncount])
17982	00005089 89C6	<1>	mov	esi, eax ; 12/10/2015 (esi=[u.pnbase])
17983	0000508B 58	<1>	pop	eax
17984	0000508C C3	<1>	retn	
17985		<1>		
17986		<1>	tr_addr_nm_err:	
17987	0000508D A3[CF740000]	<1>	mov	[u.error], eax
17988		<1>	;pop	eax
17989	00005092 E9E5EFFFFF	<1>	jmp	error
17990		<1>		
17991		<1>	trans_addr_nmk:	
17992		<1>		; 12/10/2015
17993		<1>		; 02/07/2015
17994	00005097 8B35[98740000]	<1>	mov	esi, [u.namep] ; [u.pnbase]
17995	0000509D 66B90010	<1>	mov	cx, PAGE_SIZE ; 4096 ; [u.pncount]
17996	000050A1 C3	<1>	retn	
17997		<1>		
17998		<1>	syschdir:	
17999		<1>		; / makes the directory specified in the argument
18000		<1>		; / the current directory
18001		<1>		
18002		<1>		; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18003		<1>		; 19/06/2013 (Retro UNIX 8086 v1)

```

18004      <1> ;
18005      <1> ; 'syschdir' makes the directory specified in its argument
18006      <1> ; the current working directory.
18007      <1> ;
18008      <1> ; Calling sequence:
18009      <1> ;     syschdir; name
18010      <1> ; Arguments:
18011      <1> ;     name - address of the path name of a directory
18012      <1> ;     terminated by nul byte.
18013      <1> ; Inputs: -
18014      <1> ; Outputs: -
18015      <1> ; .....
18016      <1> ;
18017      <1> ; Retro UNIX 8086 v1 modification:
18018      <1> ;     The user/application program puts address of
18019      <1> ;     the path name in BX register as 'syschdir'
18020      <1> ;     system call argument.
18021      <1>
18022 000050A2 891D[98740000] <1> mov    [u.namep], ebx
18023      <1> ;jsr r0,arg; u.namep / u.namep points to path name
18024 000050A8 E87EFFFFFF <1> call   namei
18025      <1> ; jsr r0,namei / find its i-number
18026      <1> ;jc    error
18027      <1> ; br error3
18028 000050AD 730F <1> jnc     short syschdir0
18029      <1> ; 'directory not found !' error
18030 000050AF C705[CF740000]0C00- <1> mov    dword [u.error], ERR_DIR_NOT_FOUND ; 12
18031 000050B7 0000 <1>
18032 000050B9 E9BEEFFFFFFF <1> jmp     error
18033      <1> syschdir0:
18034 000050BE E832070000 <1> call   access
18035      <1> ; jsr r0,access; 2 / get i-node into core
18036 000050C3 66F705[56710000]00- <1> test   word [i.flgs], 4000h
18037 000050CB 40 <1>
18038      <1> ; bit $40000,i.flgs / is it a directory?
18039      <1> ;jz    error
18040      <1> ; beq error3 / no error
18041 000050CC 750F <1> jnz     short syschdir1
18042 000050CE C705[CF740000]1300- <1> mov    dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
18043 000050D6 0000 <1>
18044 000050D8 E99FEFFFFFFF <1> jmp     error
18045      <1> syschdir1:
18046 000050DD 66A3[84740000] <1> mov     [u.cdir], ax
18047      <1> ; mov r1,u.cdir / move i-number to users
18048      <1> ; / current directory
18049 000050E3 66A1[66740000] <1> mov     ax, [cdev]
18050 000050E9 66A3[C4740000] <1> mov     [u.cdrv], ax
18051      <1> ; mov cdev,u.cdev / move its device to users
18052      <1> ; / current device
18053 000050EF E9A8EFFFFFFF <1> jmp     sysret
18054      <1> ; br sysret3
18055      <1>
18056      <1> syschmod: ; < change mode of file >
18057      <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18058      <1> ; 20/06/2013 - 07/07/2013 (Retro UNIX 8086 v1)
18059      <1> ;
18060      <1> ; 'syschmod' changes mode of the file whose name is given as
18061      <1> ; null terminated string pointed to by 'name' has it's mode
18062      <1> ; changed to 'mode'.
18063      <1> ;
18064      <1> ; Calling sequence:
18065      <1> ;     syschmod; name; mode
18066      <1> ; Arguments:
18067      <1> ;     name - address of the file name
18068      <1> ;     terminated by null byte.
18069      <1> ;     mode - (new) mode/flags < attributes >
18070      <1> ;
18071      <1> ; Inputs: -
18072      <1> ; Outputs: -
18073      <1> ; .....
18074      <1> ;
18075      <1> ; Retro UNIX 8086 v1 modification:
18076      <1> ;     'syschmod' system call has two arguments; so,
18077      <1> ;     * 1st argument, name is pointed to by BX register
18078      <1> ;     * 2nd argument, mode is in CX register
18079      <1> ;
18080      <1> ; Mode bits (Flags):
18081      <1> ;     bit 0 - write permission for non-owner (1)
18082      <1> ;     bit 1 - read permission for non-owner (2)
18083      <1> ;     bit 2 - write permission for owner (4)
18084      <1> ;     bit 3 - read permission for owner (8)
18085      <1> ;     bit 4 - executable flag (16)
18086      <1> ;     bit 5 - set user ID on execution flag (32)
18087      <1> ;     bit 6,7,8,9,10,11 are not used (undefined)
18088      <1> ;     bit 12 - large file flag (4096)
18089      <1> ;     bit 13 - file has modified flag (always on) (8192)
18090      <1> ;     bit 14 - directory flag (16384)
18091      <1> ;     bit 15 - 'i-node is allocated' flag (32768)
18092      <1>
18093      <1> ; / name; mode
18094 000050F4 E814000000 <1> call   isown
18095      <1> ;jsr r0,isown / get the i-node and check user status

```

```

18096 000050F9 66F705[56710000]00- <1> test word [i.flgs], 4000h
18097 00005101 40 <1>
18098 <1> ; bit $40000,i.flgs / directory?
18099 00005102 7402 <1> jz short syschmod1
18100 <1> ; beq 2f / no
18101 <1> ; AL = (new) mode
18102 00005104 24CF <1> and al, 0CFh ; 11001111b (clears bit 4 & 5)
18103 <1> ; bic $60,r2 / su & ex / yes, clear set user id and
18104 <1> ; / executable modes
18105 <1> syschmod1: ; 2:
18106 00005106 A2[56710000] <1> mov [i.flgs], al
18107 <1> ; movb r2,i.flgs / move remaining mode to i.flgs
18108 0000510B EB42 <1> jmp short isown1
18109 <1> ; br 1f
18110 <1>
18111 <1> isown:
18112 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18113 <1> ; 04/05/2013 - 07/07/2013 (Retro UNIX 8086 v1)
18114 <1> ;
18115 <1> ; 'isown' is given a file name (the 1st argument).
18116 <1> ; It find the i-number of that file via 'namei'
18117 <1> ; then gets the i-node into core via 'iget'.
18118 <1> ; It then tests to see if the user is super user.
18119 <1> ; If not, it cheks to see if the user is owner of
18120 <1> ; the file. If he is not an error occurs.
18121 <1> ; If user is the owner 'setimod' is called to indicate
18122 <1> ; the inode has been modified and the 2nd argument of
18123 <1> ; the call is put in r2.
18124 <1> ;
18125 <1> ; INPUTS ->
18126 <1> ; arguments of syschmod and syschown calls
18127 <1> ; OUTPUTS ->
18128 <1> ; u.uid - id of user
18129 <1> ; imod - set to a 1
18130 <1> ; r2 - contains second argument of the system call
18131 <1> ;
18132 <1> ; ((AX=R2) output as 2nd argument)
18133 <1> ;
18134 <1> ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
18135 <1> ;
18136 <1> ; jsr r0,arg2 / u.namep points to file name
18137 <1> ;; ! 2nd argument on top of stack !
18138 <1> ;; 22/06/2015 - 32 bit modifications
18139 <1> ;; 07/07/2013
18140 0000510D 891D[98740000] <1> mov [u.namep], ebx ;; 1st argument
18141 00005113 51 <1> push ecx ;; 2nd argument
18142 <1> ;;
18143 00005114 E812FEFFFF <1> call namei
18144 <1> ; jsr r0,namei / get its i-number
18145 <1> ; Retro UNIX 8086 v1 modification !
18146 <1> ; ax = 0 -> file not found
18147 <1> ;and ax, ax
18148 <1> ;jz error
18149 <1> ;jc error ; 27/05/2013
18150 <1> ; br error3
18151 00005119 730F <1> jnc short isown0
18152 <1> ; 'file not found !' error
18153 0000511B C705[CF740000]0C00- <1> mov dword [u.error], ERR_FILE_NOT_FOUND ; 12
18154 00005123 0000 <1>
18155 00005125 E952EFFFFF <1> jmp error
18156 <1> isown0:
18157 0000512A E8EB050000 <1> call iget
18158 <1> ; jsr r0,iget / get i-node into core
18159 0000512F A0[C6740000] <1> mov al, [u.uid] ; 02/08/2013
18160 00005134 08C0 <1> or al, al
18161 <1> ; tstb u.uid / super user?
18162 00005136 7417 <1> jz short isown1
18163 <1> ; beq 1f / yes, branch
18164 00005138 3A05[59710000] <1> cmp al, [i.uid]
18165 <1> ; cmpb i.uid,u.uid / no, is this the owner of
18166 <1> ; / the file
18167 <1> ;jne error
18168 <1> ; beq 1f / yes
18169 <1> ; jmp error3 / no, error
18170 0000513E 740F <1> je short isown1
18171 <1>
18172 00005140 C705[CF740000]0B00- <1> mov dword [u.error], ERR_NOT_OWNER ; 11
18173 00005148 0000 <1>
18174 <1> ; 'permission denied !' error
18175 0000514A E92DEFFFFF <1> jmp error
18176 <1> isown1: ; 1:
18177 0000514F E8D9060000 <1> call setimod
18178 <1> ; jsr r0,setimod / indicates
18179 <1> ; / i-node has been modified
18180 00005154 58 <1> pop eax ; 2nd argument
18181 <1> ; mov (sp)+,r2 / mode is put in r2
18182 <1> ; / (u.off put on stack with 2nd arg)
18183 00005155 C3 <1> retn
18184 <1> ; rts r0
18185 <1>
18186 <1> ;;arg: ; < get system call arguments >
18187 <1> ; 'arg' extracts an argument for a routine whose call is

```

```

18188 <1> ; of form:
18189 <1> ; sys 'routine' ; arg1
18190 <1> ; or
18191 <1> ; sys 'routine' ; arg1 ; arg2
18192 <1> ; or
18193 <1> ; sys 'routine' ; arg1;...;arg10 (sys exec)
18194 <1> ;
18195 <1> ; INPUTS ->
18196 <1> ; u.sp+18 - contains a pointer to one of arg1..argn
18197 <1> ; This pointers's value is actually the value of
18198 <1> ; update pc at the the trap to sysent (unkni) is
18199 <1> ; made to process the sys instruction
18200 <1> ; r0 - contains the return address for the routine
18201 <1> ; that called arg. The data in the word pointer
18202 <1> ; to by the return address is used as address
18203 <1> ; in which the extracted argument is stored
18204 <1> ;
18205 <1> ; OUTPUTS ->
18206 <1> ; 'address' - contains the extracted argument
18207 <1> ; u.sp+18 - is incremented by 2
18208 <1> ; r1 - contains the extracted argument
18209 <1> ; r0 - points to the next instruction to be
18210 <1> ; executed in the calling routine.
18211 <1> ;
18212 <1> ;
18213 <1> ; mov u.sp,r1
18214 <1> ; mov *18.(r1),*(r0)+ / put argument of system call
18215 <1> ; / into argument of arg2
18216 <1> ; add $2,18.(r1) / point pc on stack
18217 <1> ; / to next system argument
18218 <1> ; rts r0
18219 <1> ;
18220 <1> ;;arg2: ; < get system calls arguments - with file name pointer>
18221 <1> ; 'arg2' takes first argument in system call
18222 <1> ; (pointer to name of the file) and puts it in location
18223 <1> ; u.namep; takes second argument and puts it in u.off
18224 <1> ; and on top of the stack
18225 <1> ;
18226 <1> ; INPUTS ->
18227 <1> ; u.sp, r0
18228 <1> ;
18229 <1> ; OUTPUTS ->
18230 <1> ; u.namep
18231 <1> ; u.off
18232 <1> ; u.off pushed on stack
18233 <1> ; r1
18234 <1> ;
18235 <1> ;
18236 <1> ; jsr r0,arg; u.namep / u.namep contains value of
18237 <1> ; / first arg in sys call
18238 <1> ; jsr r0,arg; u.off / u.off contains value of
18239 <1> ; / second arg in sys call
18240 <1> ; mov r0,r1 / r0 points to calling routine
18241 <1> ; mov (sp),r0 / put operation code back in r0
18242 <1> ; mov u.off,(sp) / put pointer to second argument
18243 <1> ; / on stack
18244 <1> ; jmp (r1) / return to calling routine
18245 <1> ;
18246 <1> syschown: ; < change owner of file >
18247 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18248 <1> ; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
18249 <1> ;
18250 <1> ; 'syschown' changes the owner of the file whose name is given
18251 <1> ; as null terminated string pointed to by 'name' has it's owner
18252 <1> ; changed to 'owner'
18253 <1> ;
18254 <1> ; Calling sequence:
18255 <1> ; syschown; name; owner
18256 <1> ; Arguments:
18257 <1> ; name - address of the file name
18258 <1> ; terminated by null byte.
18259 <1> ; owner - (new) owner (number/ID)
18260 <1> ;
18261 <1> ; Inputs: -
18262 <1> ; Outputs: -
18263 <1> ; .....
18264 <1> ;
18265 <1> ; Retro UNIX 8086 v1 modification:
18266 <1> ; 'syschown' system call has two arguments; so,
18267 <1> ; * 1st argument, name is pointed to by BX register
18268 <1> ; * 2nd argument, owner number is in CX register
18269 <1> ;
18270 <1> ; / name; owner
18271 00005156 E8B2FFFFFF <1> call isown
18272 <1> ; jsr r0,isown / get the i-node and check user status
18273 0000515B 803D[C6740000]00 <1> cmp byte [u.uid], 0 ; 02/08/2013
18274 <1> ; tstb u.uid / super user
18275 00005162 7418 <1> jz short syschown1
18276 <1> ; beq 2f / yes, 2f
18277 00005164 F605[56710000]20 <1> test byte [i.flgs], 20h ; 32
18278 <1> ; bit $40,i.flgs / no, set userid on execution?
18279 <1> ;jnz error

```

Address	Hex	Assembly	Comment
18280		<1>	; bne 3f / yes error, could create Trojan Horses
18281	0000516B 740F	<1> jz short syschown1	
18282		<1>	; 'permission denied !'
18283	0000516D C705[CF740000]0B00-	<1> mov dword [u.error], ERR_FILE_ACCESS ; 11	
18284	00005175 0000	<1>	
18285	00005177 E900FFFFFF	<1> jmp error	
18286		<1>	syschown1: ; 2:
18287		<1>	; AL = owner (number/ID)
18288	0000517C A2[59710000]	<1> mov [i.uid], al ; 23/06/2015	
18289		<1>	; movb r2,i.uid / no, put the new owners id
18290		<1>	; / in the i-node
18291	00005181 E916FFFFFF	<1> jmp sysret	
18292		<1>	; 1:
18293		<1>	; jmp sysret4
18294		<1>	; 3:
18295		<1>	; jmp error
18296		<1>	
18297		<1>	sysstime: ; / get time of year
18298		<1>	; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18299		<1>	; 20/06/2013 (Retro UNIX 8086 v1)
18300		<1>	
18301		<1>	; 20/06/2013
18302		<1>	; 'sysstime' gets the time of the year.
18303		<1>	; The present time is put on the stack.
18304		<1>	
18305		<1>	; Calling sequence:
18306		<1>	; sysstime
18307		<1>	; Arguments: -
18308		<1>	
18309		<1>	; Inputs: -
18310		<1>	; Outputs: sp+2, sp+4 - present time
18311		<1>	; .....
18312		<1>	
18313		<1>	; Retro UNIX 8086 v1 modification:
18314		<1>	; 'sysstime' system call will return to the user
18315		<1>	; with unix time (epoch) in DX:AX register pair
18316		<1>	
18317		<1>	; !! Major modification on original Unix v1 'sysstime'
18318		<1>	; system call for PC compatibility !!
18319		<1>	
18320	00005186 E82CEAFF	<1> call epoch	
18321	0000518B A3[80740000]	<1> mov [u.r0], eax	
18322		<1>	; mov s.time,4(sp)
18323		<1>	; mov s.time+2,2(sp) / put the present time
18324		<1>	; / on the stack
18325		<1>	; br sysret4
18326	00005190 E907FFFFFF	<1> jmp sysret	
18327		<1>	
18328		<1>	sysstime: ; / set time
18329		<1>	; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18330		<1>	; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
18331		<1>	
18332		<1>	; 'sysstime' sets the time. Only super user can use this call.
18333		<1>	
18334		<1>	; Calling sequence:
18335		<1>	; sysstime
18336		<1>	; Arguments: -
18337		<1>	
18338		<1>	; Inputs: sp+2, sp+4 - time system is to be set to.
18339		<1>	; Outputs: -
18340		<1>	; .....
18341		<1>	
18342		<1>	; Retro UNIX 8086 v1 modification:
18343		<1>	; the user calls 'sysstime' with unix (epoch) time
18344		<1>	; (to be set) is in CX:BX register pair as two arguments.
18345		<1>	
18346		<1>	; Retro UNIX 8086 v1 argument transfer method 2 is used
18347		<1>	; to get sysstime system call arguments from the user;
18348		<1>	; * 1st argument, lowword of unix time is in BX register
18349		<1>	; * 2nd argument, highword of unix time is in CX register
18350		<1>	
18351		<1>	; !! Major modification on original Unix v1 'sysstime'
18352		<1>	; system call for PC compatibility !!
18353		<1>	
18354	00005195 803D[C6740000]00	<1> cmp byte [u.uid], 0	
18355		<1>	; tstb u.uid / is user the super user
18356		<1>	;ja error
18357		<1>	; bne error4 / no, error
18358	0000519C 760F	<1> jna short systime1	
18359		<1>	; 'permission denied !'
18360	0000519E C705[CF740000]0B00-	<1> mov dword [u.error], ERR_NOT_SUPERUSER ; 11	
18361	000051A6 0000	<1>	
18362	000051A8 E9CFEEEE	<1> jmp error	
18363			



```

18372 <1>
18373 <1> sysbreak:
18374 <1> ; 18/10/2015
18375 <1> ; 07/10/2015
18376 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
18377 <1> ; 20/06/2013 - 24/03/2014 (Retro UNIX 8086 v1)
18378 <1> ;
18379 <1> ; 'sysbreak' sets the programs break points.
18380 <1> ; It checks the current break point (u.break) to see if it is
18381 <1> ; between "core" and the stack (sp). If it is, it is made an
18382 <1> ; even address (if it was odd) and the area between u.break
18383 <1> ; and the stack is cleared. The new breakpoint is then put
18384 <1> ; in u.break and control is passed to 'sysret'.
18385 <1> ;
18386 <1> ; Calling sequence:
18387 <1> ; sysbreak; addr
18388 <1> ; Arguments: -
18389 <1> ;
18390 <1> ; Inputs: u.break - current breakpoint
18391 <1> ; Outputs: u.break - new breakpoint
18392 <1> ; area between old u.break and the stack (sp) is cleared.
18393 <1> ; .....
18394 <1> ;
18395 <1> ; Retro UNIX 8086 v1 modification:
18396 <1> ; The user/application program puts breakpoint address
18397 <1> ; in BX register as 'sysbreak' system call argument.
18398 <1> ; (argument transfer method 1)
18399 <1> ;
18400 <1> ; NOTE: Beginning of core is 0 in Retro UNIX 8086 v1 !
18401 <1> ; (('sysbreak' is not needed in Retro UNIX 8086 v1!))
18402 <1> ; NOTE:
18403 <1> ; 'sysbreak' clears extended part (beyond of previous
18404 <1> ; 'u.break' address) of user's memory for original unix's
18405 <1> ; 'bss' compatibility with Retro UNIX 8086 v1 (19/11/2013)
18406 <1>
18407 <1> ; mov u.break,r1 / move users break point to r1
18408 <1> ; cmp r1,$core / is it the same or lower than core?
18409 <1> ; blos lf / yes, lf
18410 <1> ; 23/06/2015
18411 000051B9 8B2D[AC740000] <1> mov ebp, [u.break] ; virtual address (offset)
18412 <1> ;and ebp, ebp
18413 <1> ;jz short sysbreak_3
18414 <1> ; Retro UNIX 386 v1 NOTE: u.break points to virtual address !!!
18415 <1> ; (Even break point address is not needed for Retro UNIX 386 v1)
18416 000051BF 8B15[78740000] <1> mov edx, [u.sp] ; kernel stack at the beginning of sys call
18417 000051C5 83C20C <1> add edx, 12 ; EIP -4-> CS -4-> EFLAGS -4-> ESP (user)
18418 <1> ; 07/10/2015
18419 000051C8 891D[AC740000] <1> mov [u.break], ebx ; virtual address !!!
18420 <1> ;
18421 000051CE 3B1A <1> cmp ebx, [edx] ; compare new break point with
18422 <1> ; with top of user's stack (virtual!)
18423 000051D0 7327 <1> jnb short sysbreak_3
18424 <1> ; cmp r1,sp / is it the same or higher
18425 <1> ; / than the stack?
18426 <1> ; bhis lf / yes, lf
18427 000051D2 89DE <1> mov esi, ebx
18428 000051D4 29EE <1> sub esi, ebp ; new break point - old break point
18429 000051D6 7621 <1> jna short sysbreak_3
18430 <1> ;push ebx
18431 <1> sysbreak_1:
18432 000051D8 89EB <1> mov ebx, ebp
18433 000051DA E8EBE4FFFF <1> call get_physical_addr ; get physical address
18434 000051DF 0F82A8FEFFFF <1> jc tr_addr_nm_err
18435 <1> ; 18/10/2015
18436 000051E5 89C7 <1> mov edi, eax
18437 000051E7 29C0 <1> sub eax, eax ; 0
18438 <1> ; ECX = remain byte count in page (1-4096)
18439 000051E9 39CE <1> cmp esi, ecx
18440 000051EB 7302 <1> jnb short sysbreak_2
18441 000051ED 89F1 <1> mov ecx, esi
18442 <1> sysbreak_2:
18443 000051EF 29CE <1> sub esi, ecx
18444 000051F1 01CD <1> add ebp, ecx
18445 000051F3 F3AA <1> rep stosb
18446 000051F5 09F6 <1> or esi, esi
18447 000051F7 75DF <1> jnz short sysbreak_1
18448 <1> ;
18449 <1> ; bit $1,r1 / is it an odd address
18450 <1> ; beq 2f / no, its even
18451 <1> ; clrb (r1)+ / yes, make it even
18452 <1> ; 2: / clear area between the break point and the stack
18453 <1> ; cmp r1,sp / is it higher or same than the stack
18454 <1> ; bhis lf / yes, quit
18455 <1> ; clr (r1)+ / clear word
18456 <1> ; br 2b / go back
18457 <1> ;pop ebx
18458 <1> sysbreak_3: ; 1:
18459 <1> ;mov [u.break], ebx ; virtual address !!!
18460 <1> ; jsr r0,arg; u.break / put the "address"
18461 <1> ; / in u.break (set new break point)
18462 <1> ; br sysret4 / br sysret
18463 000051F9 E99EEEEFFF <1> jmp sysret

```

18464	<1>
18465	<1>
18466	<1> maknod:
18467	<1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18468	<1> ; 02/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
18469	<1> ;
18470	<1> ; 'maknod' creates an i-node and makes a directory entry
18471	<1> ; for this i-node in the current directory.
18472	<1> ;
18473	<1> ; INPUTS ->
18474	<1> ; r1 - contains mode
18475	<1> ; ii - current directory's i-number
18476	<1> ;
18477	<1> ; OUTPUTS ->
18478	<1> ; u.dirbuf - contains i-number of free i-node
18479	<1> ; i.flgs - flags in new i-node
18480	<1> ; i.uid - filled with u.uid
18481	<1> ; i.nlks - 1 is put in the number of links
18482	<1> ; i.ctim - creation time
18483	<1> ; i.ctim+2 - modification time
18484	<1> ; imod - set via call to setimod
18485	<1> ;
18486	<1> ; ((AX = R1)) input
18487	<1> ;
18488	<1> ; (Retro UNIX Prototype :
18489	<1> ; 30/10/2012 - 01/03/2013, UNIXCOPY.ASM)
18490	<1> ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
18491	<1>
18492	<1> ; / r1 contains the mode
18493 000051FE 80CC80	<1> or ah, 80h ; 10000000b
18494	<1> ; bis \$100000,r1 / allocate flag set
18495 00005201 6650	<1> push ax
18496	<1> ; mov r1,-(sp) / put mode on stack
18497	<1> ; 31/07/2013
18498 00005203 66A1[62740000]	<1> mov ax, [ii] ; move current i-number to AX/r1
18499	<1> ; mov ii,r1 / move current i-number to r1
18500 00005209 B201	<1> mov dl, 1 ; owner flag mask
18501 0000520B E8E5050000	<1> call access
18502	<1> ; jsr r0,access; 1 / get its i-node into core
18503 00005210 6650	<1> push ax
18504	<1> ; mov r1,-(sp) / put i-number on stack
18505 00005212 66B82800	<1> mov ax, 40
18506	<1> ; mov \$40.,r1 / r1 = 40
18507	<1> maknod1: ; 1: / scan for a free i-node (next 4 instructions)
18508 00005216 6640	<1> inc ax
18509	<1> ; inc r1 / r1 = r1 + 1
18510 00005218 E8A9060000	<1> call imap
18511	<1> ; jsr r0,imap / get byte address and bit position in
18512	<1> ; / inode map in r2 & m
18513	<1> ; DX (MQ) has a 1 in the calculated bit position
18514	<1> ; eBX (R2) has byte address of the byte with allocation bit
18515	<1> ; 22/06/2015 - NOTE for next Retro UNIX version:
18516	<1> ; Inode count must be checked here
18517	<1> ; (Original UNIX v1 did not check inode count here !?)
18518 0000521D 8413	<1> test [ebx], dl
18519	<1> ; bitb mq, (r2) / is the i-node active
18520 0000521F 75F5	<1> jnz short maknod1
18521	<1> ; bne 1b / yes, try the next one
18522 00005221 0813	<1> or [ebx], dl
18523	<1> ; bisb mq, (r2) / no, make it active
18524	<1> ; / (put a 1 in the bit map)
18525 00005223 E8F2040000	<1> call iget
18526	<1> ; jsr r0,iget / get i-node into core
18527 00005228 66F705[56710000]00-	<1> test word [i.flgs], 8000h
18528 00005230 80	<1>
18529	<1> ; tst i.flgs / is i-node already allocated
18530 00005231 75E3	<1> jnz short maknod1
18531	<1> ; blt 1b / yes, look for another one
18532 00005233 66A3[B2740000]	<1> mov [u.dirbuf], ax
18533	<1> ; mov r1,u.dirbuf / no, put i-number in u.dirbuf
18534 00005239 6658	<1> pop ax
18535	<1> ; mov (sp)+,r1 / get current i-number back
18536 0000523B E8DA040000	<1> call iget
18537	<1> ; jsr r0,iget / get i-node in core
18538 00005240 E880F7FFFF	<1> call mkdir
18539	<1> ; jsr r0,mkdir / make a directory entry
18540	<1> ; / in current directory
18541 00005245 66A1[B2740000]	<1> mov ax, [u.dirbuf]
18542	<1> ; mov u.dirbuf,r1 / r1 = new inode number
18543 0000524B E8CA040000	<1> call iget
18544	<1> ; jsr r0,iget / get it into core
18545	<1> ; jsr r0,copyz; inode; inode+32. / 0 it out
18546 00005250 B908000000	<1> mov ecx, 8
18547 00005255 31C0	<1> xor eax, eax ; 0
18548 00005257 BF[56710000]	<1> mov edi, inode
18549 0000525C F3AB	<1> rep stosd
18550	<1> ;
18551 0000525E 668F05[56710000]	<1> pop word [i.flgs]
18552	<1> ; mov (sp)+,i.flgs / fill flags
18553 00005265 8A0D[C6740000]	<1> mov cl, [u.uid] ; 02/08/2013
18554 0000526B 880D[59710000]	<1> mov [i.uid], cl
18555	<1> ; movb u.uid,i.uid / user id

```

18556 00005271 C605[58710000]01 <1> mov byte [i.nlks], 1
18557 <1> ; movb $1,i.nlks / 1 link
18558 <1> ;call epoch ; Retro UNIX 8086 v1 modification !
18559 <1> ;mov eax, [s.time]
18560 <1> ;mov [i.ctim], eax
18561 <1> ; mov s.time,i.ctim / time created
18562 <1> ; mov s.time+2,i.ctim+2 / time modified
18563 <1> ; Retro UNIX 8086 v1 modification !
18564 <1> ; i.ctime=0, i.ctime+2=0 and
18565 <1> ; 'setimod' will set ctime of file via 'epoch'
18566 00005278 E8B0050000 <1> call setimod
18567 <1> ; jsr r0,setimod / set modified flag
18568 0000527D C3 <1> retn
18569 <1> ; rts r0 / return
18570 <1>
18571 <1> sysseek: ; / moves read write pointer in an fsp entry
18572 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18573 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
18574 <1> ;
18575 <1> ; 'sysseek' changes the r/w pointer of (3rd word of in an
18576 <1> ; fsp entry) of an open file whose file descriptor is in u.r0.
18577 <1> ; The file descriptor refers to a file open for reading or
18578 <1> ; writing. The read (or write) pointer is set as follows:
18579 <1> ; * if 'ptrname' is 0, the pointer is set to offset.
18580 <1> ; * if 'ptrname' is 1, the pointer is set to its
18581 <1> ; current location plus offset.
18582 <1> ; * if 'ptrname' is 2, the pointer is set to the
18583 <1> ; size of file plus offset.
18584 <1> ; The error bit (e-bit) is set for an undefined descriptor.
18585 <1> ;
18586 <1> ; Calling sequence:
18587 <1> ; sysseek; offset; ptrname
18588 <1> ; Arguments:
18589 <1> ; offset - number of bytes desired to move
18590 <1> ; the r/w pointer
18591 <1> ; ptrname - a switch indicated above
18592 <1> ;
18593 <1> ; Inputs: r0 - file descriptor
18594 <1> ; Outputs: -
18595 <1> ; .....
18596 <1> ;
18597 <1> ; Retro UNIX 8086 v1 modification:
18598 <1> ; 'sysseek' system call has three arguments; so,
18599 <1> ; * 1st argument, file descriptor is in BX (BL) register
18600 <1> ; * 2nd argument, offset is in CX register
18601 <1> ; * 3rd argument, ptrname/switch is in DX (DL) register
18602 <1> ;
18603 <1>
18604 0000527E E823000000 <1> call seektell
18605 <1> ; AX = u.count
18606 <1> ; BX = *u.fofp
18607 <1> ; jsr r0,seektell / get proper value in u.count
18608 <1> ; add u.base,u.count / add u.base to it
18609 00005283 0305[A0740000] <1> add eax, [u.base] ; add offset (u.base) to base
18610 00005289 8903 <1> mov [ebx], eax
18611 <1> ; mov u.count,*u.fofp / put result into r/w pointer
18612 0000528B E90CEFFFFFF <1> jmp sysret
18613 <1> ; br sysret4
18614 <1>
18615 <1> systell: ; / get the r/w pointer
18616 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18617 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
18618 <1> ;
18619 <1> ; Retro UNIX 8086 v1 modification:
18620 <1> ; ! 'systell' does not work in original UNIX v1,
18621 <1> ; it returns with error !
18622 <1> ; Inputs: r0 - file descriptor
18623 <1> ; Outputs: r0 - file r/w pointer
18624 <1>
18625 <1> ;xor ecx, ecx ; 0
18626 00005290 BA01000000 <1> mov edx, 1 ; 05/08/2013
18627 <1> ;call seektell
18628 00005295 E812000000 <1> call seektell0 ; 05/08/2013
18629 <1> ;mov ebx, [u.fofp]
18630 0000529A 8B03 <1> mov eax, [ebx]
18631 0000529C A3[80740000] <1> mov [u.r0], eax
18632 000052A1 E9F6EDFFFF <1> jmp sysret
18633 <1>
18634 <1> ; Original unix v1 'systell' system call:
18635 <1> ; jsr r0,seektell
18636 <1> ; br error4
18637 <1>
18638 <1> seektell:
18639 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18640 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
18641 <1> ;
18642 <1> ; 'seektell' puts the arguments from sysseek and systell
18643 <1> ; call in u.base and u.count. It then gets the i-number of
18644 <1> ; the file from the file descriptor in u.r0 and by calling
18645 <1> ; getf. The i-node is brought into core and then u.count
18646 <1> ; is checked to see it is a 0, 1, or 2.
18647 <1> ; If it is 0 - u.count stays the same

```

```

18648 <1> ; 1 - u.count = offset (u.fofp)
18649 <1> ; 2 - u.count = i.size (size of file)
18650 <1> ;
18651 <1> ; !! Retro UNIX 8086 v1 modification:
18652 <1> ; Argument 1, file descriptor is in BX;
18653 <1> ; Argument 2, offset is in CX;
18654 <1> ; Argument 3, ptrname/switch is in DX register.
18655 <1> ;
18656 <1> ; mov ax, 3 ; Argument transfer method 3 (three arguments)
18657 <1> ; call arg
18658 <1> ;
18659 <1> ; ((Return -> ax = base for offset (position= base+offset))
18660 <1> ;
18661 000052A6 890D[A0740000] <1> mov [u.base], ecx ; offset
18662 <1> ; jsr r0,arg; u.base / puts offset in u.base
18663 <1> seektell0:
18664 000052AC 8915[A4740000] <1> mov [u.count], edx
18665 <1> ; jsr r0,arg; u.count / put ptr name in u.count
18666 <1> ; mov ax, bx
18667 <1> ; mov *u.r0,r1 / file descriptor in r1
18668 <1> ; / (index in u.fp list)
18669 <1> ; call getf
18670 <1> ; jsr r0,getf / u.fofp points to 3rd word in fsp entry
18671 <1> ; BX = file descriptor (file number)
18672 000052B2 E83DFCFFFF <1> call getf1
18673 000052B7 6609C0 <1> or ax, ax ; i-number of the file
18674 <1> ; mov r1,-(sp) / r1 has i-number of file,
18675 <1> ; / put it on the stack
18676 <1> ;jz error
18677 <1> ; beq error4 / if i-number is 0, not active so error
18678 000052BA 750F <1> jnz short seektell1
18679 000052BC C705[CF740000]0A00- <1> mov dword [u.error], ERR_FILE_NOT_OPEN ; 'file not open !'
18680 000052C4 0000 <1>
18681 000052C6 E9B1EDFFFF <1> jmp error
18682 <1> seektell1:
18683 <1> ;push eax
18684 000052CB 80FC80 <1> cmp ah, 80h
18685 000052CE 7203 <1> jb short seektell2
18686 <1> ; bgt .+4 / if its positive jump
18687 000052D0 66F7D8 <1> neg ax
18688 <1> ; neg r1 / if not make it positive
18689 <1> seektell2:
18690 000052D3 E842040000 <1> call iget
18691 <1> ; jsr r0,iget / get its i-node into core
18692 000052D8 8B1D[90740000] <1> mov ebx, [u.fofp] ; 05/08/2013
18693 000052DE 803D[A4740000]01 <1> cmp byte [u.count], 1
18694 <1> ; cmp u.count,$1 / is ptr name =1
18695 000052E5 7705 <1> ja short seektell3
18696 <1> ; blt 2f / no its zero
18697 000052E7 7409 <1> je short seektell_4
18698 <1> ; beq 1f / yes its 1
18699 000052E9 31C0 <1> xor eax, eax
18700 <1> ;jmp short seektell_5
18701 000052EB C3 <1> retn
18702 <1> seektell3:
18703 000052EC A1[5A710000] <1> mov eax, [i.size]
18704 <1> ; mov i.size,u.count / put number of bytes
18705 <1> ; / in file in u.count
18706 <1> ;jmp short seektell_5
18707 <1> ; br 2f
18708 000052F1 C3 <1> retn
18709 <1> seektell_4: ; 1: / ptrname =1
18710 <1> ;mov ebx, [u.fofp]
18711 000052F2 8B03 <1> mov eax, [ebx]
18712 <1> ; mov *u.fofp,u.count / put offset in u.count
18713 <1> ;seektell_5: ; 2: / ptrname =0
18714 <1> ;mov [u.count], eax
18715 <1> ;pop eax
18716 <1> ; mov (sp)+,r1 / i-number on stack r1
18717 000052F4 C3 <1> retn
18718 <1> ; rts r0
18719 <1>
18720 <1> sysintr: ; / set interrupt handling
18721 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18722 <1> ; 07/07/2013 (Retro UNIX 8086 v1)
18723 <1> ;
18724 <1> ; 'sysintr' sets the interrupt handling value. It puts
18725 <1> ; argument of its call in u.intr then branches into 'sysquit'
18726 <1> ; routine. u.tty is checked if to see if a control tty exists.
18727 <1> ; If one does the interrupt character in the tty buffer is
18728 <1> ; cleared and 'sysret'is called. If one does not exists
18729 <1> ; 'sysret' is just called.
18730 <1> ;
18731 <1> ; Calling sequence:
18732 <1> ; sysintr; arg
18733 <1> ; Argument:
18734 <1> ; arg - if 0, interrupts (ASCII DELETE) are ignored.
18735 <1> ; - if 1, interrupts cause their normal result
18736 <1> ; i.e force an exit.
18737 <1> ; - if arg is a location within the program,
18738 <1> ; control is passed to that location when
18739 <1> ; an interrupt occurs.

```

```

18740 <1> ; Inputs: -
18741 <1> ; Outputs: -
18742 <1> ; .....
18743 <1> ;
18744 <1> ; Retro UNIX 8086 v1 modification:
18745 <1> ; 'sysintr' system call sets u.intr to value of BX
18746 <1> ; then branches into sysquit.
18747 <1> ;
18748 000052F5 66891D[BE740000] <1> mov [u.intr], bx
18749 <1> ; jsr r0,arg; u.intr / put the argument in u.intr
18750 <1> ; br 1f / go into quit routine
18751 000052FC E99BEDFFFF <1> jmp sysret
18752 <1>
18753 <1> sysquit:
18754 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18755 <1> ; 07/07/2013 (Retro UNIX 8086 v1)
18756 <1> ;
18757 <1> ; 'sysquit' turns off the quit signal. it puts the argument of
18758 <1> ; the call in u.quit. u.tty is checked if to see if a control
18759 <1> ; tty exists. If one does the interrupt character in the tty
18760 <1> ; buffer is cleared and 'sysret' is called. If one does not exists
18761 <1> ; 'sysret' is just called.
18762 <1> ;
18763 <1> ; Calling sequence:
18764 <1> ; sysquit; arg
18765 <1> ; Argument:
18766 <1> ; arg - if 0, this call disables quit signals from the
18767 <1> ; typewriter (ASCII FS)
18768 <1> ; - if 1, quits are re-enabled and cause execution to
18769 <1> ; cease and a core image to be produced.
18770 <1> ; i.e force an exit.
18771 <1> ; - if arg is an address in the program,
18772 <1> ; a quit causes control to sent to that
18773 <1> ; location.
18774 <1> ; Inputs: -
18775 <1> ; Outputs: -
18776 <1> ; .....
18777 <1> ;
18778 <1> ; Retro UNIX 8086 v1 modification:
18779 <1> ; 'sysquit' system call sets u.quit to value of BX
18780 <1> ; then branches into 'sysret'.
18781 <1> ;
18782 00005301 66891D[C0740000] <1> mov [u.quit], bx
18783 00005308 E98FEDFFFF <1> jmp sysret
18784 <1> ; jsr r0,arg; u.quit / put argument in u.quit
18785 <1> ;1:
18786 <1> ; mov u.ttyp,r1 / move pointer to control tty buffer
18787 <1> ; / to r1
18788 <1> ; beq sysret4 / return to user
18789 <1> ; clrb 6(r1) / clear the interrupt character
18790 <1> ; / in the tty buffer
18791 <1> ; br sysret4 / return to user
18792 <1>
18793 <1> syssetuid: ; / set process id
18794 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18795 <1> ; 07/07/2013 - 02/08/2013 (Retro UNIX 8086 v1)
18796 <1> ;
18797 <1> ; 'syssetuid' sets the user id (u.uid) of the current process
18798 <1> ; to the process id in (u.r0). Both the effective user and
18799 <1> ; u.uid and the real user u.ruid are set to this.
18800 <1> ; Only the super user can make this call.
18801 <1> ;
18802 <1> ; Calling sequence:
18803 <1> ; syssetuid
18804 <1> ; Arguments: -
18805 <1> ;
18806 <1> ; Inputs: (u.r0) - contains the process id.
18807 <1> ; Outputs: -
18808 <1> ; .....
18809 <1> ;
18810 <1> ; Retro UNIX 8086 v1 modification:
18811 <1> ; BL contains the (new) user ID of the current process
18812 <1>
18813 <1> ; movb *u.r0,r1 / move process id (number) to r1
18814 0000530D 3A1D[C7740000] <1> cmp bl, [u.ruid]
18815 <1> ; cmpb r1,u.ruid / is it equal to the real user
18816 <1> ; / id number
18817 00005313 741E <1> je short setuid1
18818 <1> ; beq 1f / yes
18819 00005315 803D[C6740000]00 <1> cmp byte [u.uid], 0 ; 02/08/2013
18820 <1> ; tstb u.uid / no, is current user the super user?
18821 <1> ;ja error
18822 <1> ; bne error4 / no, error
18823 0000531C 760F <1> jna short setuid0
18824 0000531E C705[CF740000]0B00- <1> mov dword [u.error], ERR_NOT_SUPERUSER ; 11
18825 00005326 0000 <1>
18826 <1> ; 'permission denied !' error
18827 00005328 E94FEDFFFF <1> jmp error
18828 <1> setuid0:
18829 0000532D 881D[C7740000] <1> mov [u.ruid], bl
18830 <1> setuid1: ; 1:
18831 00005333 881D[C6740000] <1> mov [u.uid], bl ; 02/08/2013

```

```

18832      <1>      ; movb r1,u.uid / put process id in u.uid
18833      <1>      ; movb r1,u.ruid / put process id in u.ruid
18834 00005339 E95EEDFFFF      <1>      jmp      sysret
18835      <1>      ; br sysret4 / system return
18836      <1>
18837      <1> sysgetuid: ; < get user id >
18838      <1>      ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18839      <1>      ; 07/07/2013 (Retro UNIX 8086 v1)
18840      <1>      ;
18841      <1>      ; 'sysgetuid' returns the real user ID of the current process.
18842      <1>      ; The real user ID identifies the person who is logged in,
18843      <1>      ; in contradistinction to the effective user ID, which
18844      <1>      ; determines his access permission at each moment. It is thus
18845      <1>      ; useful to programs which operate using the 'set user ID'
18846      <1>      ; mode, to find out who invoked them.
18847      <1>      ;
18848      <1>      ; Calling sequence:
18849      <1>      ;      syssetuid
18850      <1>      ; Arguments: -
18851      <1>      ;
18852      <1>      ; Inputs: -
18853      <1>      ; Outputs: (u.r0) - contains the real user's id.
18854      <1>      ; .....
18855      <1>      ;
18856      <1>      ; Retro UNIX 8086 v1 modification:
18857      <1>      ;      AL contains the real user ID at return.
18858      <1>      ;
18859 0000533E 0FB605[C7740000]      <1>      movzx     eax, byte [u.ruid]
18860 00005345 A3[80740000]      <1>      mov      [u.r0], eax
18861      <1>      ; movb      u.ruid,*u.r0 / move the real user id to (u.r0)
18862 0000534A E94DEDDFFF      <1>      jmp      sysret
18863      <1>      ; br sysret4 / system return, sysret
18864      <1>
18865      <1> anyi:
18866      <1>      ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
18867      <1>      ; 25/04/2013 (Retro UNIX 8086 v1)
18868      <1>      ;
18869      <1>      ; 'anyi' is called if a file deleted while open.
18870      <1>      ; "anyi" checks to see if someone else has opened this file.
18871      <1>      ;
18872      <1>      ; INPUTS ->
18873      <1>      ;      r1 - contains an i-number
18874      <1>      ;      fsp - start of table containing open files
18875      <1>      ;
18876      <1>      ; OUTPUTS ->
18877      <1>      ;      "deleted" flag set in fsp entry of another occurrence of
18878      <1>      ;      this file and r2 points 1st word of this fsp entry.
18879      <1>      ;      if file not found - bit in i-node map is cleared
18880      <1>      ;      (i-node is freed)
18881      <1>      ;      all blocks related to i-node are freed
18882      <1>      ;      all flags in i-node are cleared
18883      <1>      ; ((AX = R1)) input
18884      <1>      ;
18885      <1>      ; (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
18886      <1>      ;      ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
18887      <1>      ;
18888      <1>      ; / r1 contains an i-number
18889 0000534F BB[56720000]      <1>      mov      ebx, fsp
18890      <1>      ; mov $fsp,r2 / move start of fsp table to r2
18891      <1> anyi_1: ; 1:
18892 00005354 663B03      <1>      cmp      ax, [ebx]
18893      <1>      ; cmp r1,(r2) / do i-numbers match?
18894 00005357 7433      <1>      je       short anyi_3
18895      <1>      ; beq 1f / yes, 1f
18896 00005359 66F7D8      <1>      neg      ax
18897      <1>      ; neg r1 / no complement r1
18898 0000535C 663B03      <1>      cmp      ax, [ebx]
18899      <1>      ; cmp r1,(r2) / do they match now?
18900 0000535F 742B      <1>      je       short anyi_3
18901      <1>      ; beq 1f / yes, transfer
18902      <1>      ; / i-numbers do not match
18903 00005361 83C30A      <1>      add      ebx, 10 ; fsp table size is 10 bytes
18904      <1>      ; in Retro UNIX 386 v1 (22/06/2015)
18905      <1>      ; add $8,r2 / no, bump to next entry in fsp table
18906 00005364 81FB[4A740000]      <1>      cmp      ebx, fsp + (nfiles*10) ; 22/06/2015
18907      <1>      ; cmp r2,$fsp+[nfiles*8]
18908      <1>      ; / are we at last entry in the table
18909 0000536A 72E8      <1>      jnb      short anyi_1
18910      <1>      ; blt 1b / no, check next entries i-number
18911      <1>      ;cmp ax, 32768
18912 0000536C 80FC80      <1>      cmp      ah, 80h ; negative number check
18913      <1>      ; tst r1 / yes, no match
18914      <1>      ; bge .+4
18915 0000536F 7203      <1>      jnb      short anyi_2
18916 00005371 66F7D8      <1>      neg      ax
18917      <1>      ; neg r1 / make i-number positive
18918      <1> anyi_2:
18919 00005374 E84D050000      <1>      call     imap
18920      <1>      ; jsr r0,imap / get address of allocation bit
18921      <1>      ; / in the i-map in r2
18922      <1>      ;; DL/DX (MQ) has a 1 in the calculated bit position
18923      <1>      ;; eBX (R2) has address of the byte with allocation bit

```

```

18924      <1> ; not dx
18925 00005379 F6D2      <1> not dl ;; 0 at calculated bit position, other bits are 1
18926      <1> ; and [ebx], dx
18927 0000537B 2013      <1> and [ebx], dl
18928      <1> ; bich mq, (r2) / clear bit for i-node in the imap
18929 0000537D E8CD040000 <1> call itrunc
18930      <1> ; jsr r0,itrunc / free all blocks related to i-node
18931 00005382 66C705[56710000]00- <1> mov word [i.flgs], 0
18932 0000538A 00      <1>
18933      <1> ; clr i.flgs / clear all flags in the i-node
18934 0000538B C3      <1> retn
18935      <1> ; rts r0 / return
18936      <1> anyi_3: ; 1: / i-numbers match
18937 0000538C FE4309      <1> inc byte [ebx+9] ; 22/06/2015
18938      <1> ; incb 7(r2) / increment upper byte of the 4th word
18939      <1> ; / in that fsp entry (deleted flag of fsp entry)
18940 0000538F C3      <1> retn
18941      <1> ; rts r0
18942      %include 'u3.s' ; 10/05/2015
18943      <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS3.INC
18944      <1> ; Last Modification: 15/09/2015
18945      <1> ; -----
18946      <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
18947      <1> ; (v0.1 - Beginning: 11/07/2012)
18948      <1> ;
18949      <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
18950      <1> ; (Original) Source Code by Ken Thompson (1971-1972)
18951      <1> ; <Bell Laboratories (17/3/1972)>
18952      <1> ; <Preliminary Release of UNIX Implementation Document>
18953      <1> ;
18954      <1> ; Retro UNIX 8086 v1 - U3.ASM (08/03/2014) //// UNIX v1 -> u3.s
18955      <1> ;
18956      <1> ; *****
18957      <1>
18958      <1> tswitch: ; Retro UNIX 386 v1
18959      <1> tswap:
18960      <1> ; 01/09/2015
18961      <1> ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
18962      <1> ; 14/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
18963      <1> ; time out swap, called when a user times out.
18964      <1> ; the user is put on the low priority queue.
18965      <1> ; This is done by making a link from the last user
18966      <1> ; on the low priority queue to him via a call to 'putlu'.
18967      <1> ; then he is swapped out.
18968      <1> ;
18969      <1> ; Retro UNIX 386 v1 modification ->
18970      <1> ; swap (software task switch) is performed by changing
18971      <1> ; user's page directory (u.pgdir) instead of segment change
18972      <1> ; as in Retro UNIX 8086 v1.
18973      <1> ;
18974      <1> ; RETRO UNIX 8086 v1 modification ->
18975      <1> ; 'swap to disk' is replaced with 'change running segment'
18976      <1> ; according to 8086 cpu (x86 real mode) architecture.
18977      <1> ; pdp-11 was using 64KB uniform memory while IBM PC
18978      <1> ; compatibles was using 1MB segmented memory
18979      <1> ; in 8086/8088 times.
18980      <1> ;
18981      <1> ; INPUTS ->
18982      <1> ; u.uno - users process number
18983      <1> ; runq+4 - lowest priority queue
18984      <1> ; OUTPUTS ->
18985      <1> ; r0 - users process number
18986      <1> ; r2 - lowest priority queue address
18987      <1> ;
18988      <1> ; ((AX = R0, BX = R2)) output
18989      <1> ; ((Modified registers: EDX, EBX, ECX, ESI, EDI))
18990      <1> ;
18991 00005390 A0[C9740000] <1> mov al, [u.uno]
18992      <1> ; movb u.uno,r1 / move users process number to r1
18993      <1> ; mov $runq+4,r2
18994      <1> ; / move lowest priority queue address to r2
18995 00005395 E8CE000000 <1> call putlu
18996      <1> ; jsr r0,putlu / create link from last user on Q to
18997      <1> ; / u.uno's user
18998      <1>
18999      <1> switch: ; Retro UNIX 386 v1
19000      <1> swap:
19001      <1> ; 02/09/2015
19002      <1> ; 01/09/2015
19003      <1> ; 31/08/2015
19004      <1> ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
19005      <1> ; 14/04/2013 - 08/03/2014 (Retro UNIX 8086 v1)
19006      <1> ; 'swap' is routine that controls the swapping of processes
19007      <1> ; in and out of core.
19008      <1> ;
19009      <1> ; Retro UNIX 386 v1 modification ->
19010      <1> ; swap (software task switch) is performed by changing
19011      <1> ; user's page directory (u.pgdir) instead of segment change
19012      <1> ; as in Retro UNIX 8086 v1.
19013      <1> ;
19014      <1> ; RETRO UNIX 8086 v1 modification ->
19015      <1> ; 'swap to disk' is replaced with 'change running segment'

```

```

19016 <1> ; according to 8086 cpu (x86 real mode) architecture.
19017 <1> ; pdp-11 was using 64KB uniform memory while IBM PC
19018 <1> ; compatibles was using 1MB segmented memory
19019 <1> ; in 8086/8088 times.
19020 <1> ;
19021 <1> ; INPUTS ->
19022 <1> ; runq table - contains processes to run.
19023 <1> ; p.link - contains next process in line to be run.
19024 <1> ; u.uno - process number of process in core
19025 <1> ; s.stack - swap stack used as an internal stack for swapping.
19026 <1> ; OUTPUTS ->
19027 <1> ; (original unix v1 -> present process to its disk block)
19028 <1> ; (original unix v1 -> new process into core ->
19029 <1> ; Retro Unix 8086 v1 -> segment registers changed
19030 <1> ; for new process)
19031 <1> ; u.quant = 3 (Time quantum for a process)
19032 <1> ; ((INT 1Ch count down speed -> 18.2 times per second)
19033 <1> ; RETRO UNIX 8086 v1 will use INT 1Ch (18.2 times per second)
19034 <1> ; for now, it will swap the process if there is not
19035 <1> ; a keyboard event (keystroke) (Int 15h, function 4Fh)
19036 <1> ; or will count down from 3 to 0 even if there is a
19037 <1> ; keyboard event locking due to repetitive key strokes.
19038 <1> ; u.quant will be reset to 3 for RETRO UNIX 8086 v1.
19039 <1> ;
19040 <1> ; u.pri -points to highest priority run Q.
19041 <1> ; r2 - points to the run queue.
19042 <1> ; r1 - contains new process number
19043 <1> ; r0 - points to place in routine or process that called
19044 <1> ; swap all user parameters
19045 <1> ;
19046 <1> ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI))
19047 <1> ;
19048 <1> swap_0:
19049 <1> ;mov $300,$ps / processor priority = 6
19050 0000539A BE[72740000] <1> mov esi, runq
19051 <1> ; mov $runq,r2 / r2 points to runq table
19052 <1> swap_1: ; 1: / search runq table for highest priority process
19053 0000539F 668B06 <1> mov ax, [esi]
19054 000053A2 6621C0 <1> and ax, ax
19055 <1> ; tst (r2)+ / are there any processes to run
19056 <1> ; / in this Q entry
19057 000053A5 7507 <1> jnz short swap_2
19058 <1> ; bne 1f / yes, process 1f
19059 <1> ; cmp r2,$runq+6 / if zero compare address
19060 <1> ; / to end of table
19061 <1> ; bne 1b / if not at end, go back
19062 000053A7 E8E1000000 <1> call idle
19063 <1> ; jsr r0,idle; s.idlet+2 / wait for interrupt;
19064 <1> ; / all queues are empty
19065 000053AC EBF1 <1> jmp short swap_1
19066 <1> ; br swap
19067 <1> swap_2: ; 1:
19068 000053AE 0FB6D8 <1> movzx ebx, al ; 02/09/2015
19069 <1> ; tst -(r2) / restore pointer to right Q entry
19070 <1> ; mov r2,u.pri / set present user to this run queue
19071 <1> ; movb (r2)+,r1 / move 1st process in queue to r1
19072 000053B1 38E0 <1> cmp al, ah
19073 <1> ; cmpb r1,(r2)+ / is there only 1 process
19074 <1> ; / in this Q to be run
19075 000053B3 740A <1> je short swap_3
19076 <1> ; beq 1f / yes
19077 <1> ; tst -(r2) / no, pt r2 back to this Q entry
19078 <1> ;movzx ebx, al
19079 000053B5 8AA3[F5710000] <1> mov ah, [ebx+p.link-1]
19080 000053BB 8826 <1> mov [esi], ah
19081 <1> ; movb p.link-1(r1),(r2) / move next process
19082 <1> ; / in line into run queue
19083 000053BD EB06 <1> jmp short swap_4
19084 <1> ; br 2f
19085 <1> swap_3: ; 1:
19086 000053BF 6631D2 <1> xor dx, dx
19087 000053C2 668916 <1> mov [esi], dx
19088 <1> ; clr -(r2) / zero the entry; no processes on the Q
19089 <1> swap_4: ; / write out core to appropriate disk area and read
19090 <1> ; / in new process if required
19091 <1> ; clr $ps / clear processor status
19092 000053C5 8A25[C9740000] <1> mov ah, [u.uno]
19093 000053CB 38C4 <1> cmp ah, al
19094 <1> ; cmpb r1,u.uno / is this process the same as
19095 <1> ; / the process in core?
19096 000053CD 743B <1> je short swap_8
19097 <1> ; beq 2f / yes, don't have to swap
19098 <1> ; mov r0,-(sp) / no, write out core; save r0
19099 <1> ; / (address in routine that called swap)
19100 <1> ; mov r1,-(sp) / put r1 (new process #) on the stack
19101 <1> ; 01/09/2015
19102 <1> ;mov [u.usp], esp
19103 <1> ; mov sp,u.usp / save stack pointer
19104 <1> ; mov $sstack,sp / move swap stack pointer
19105 <1> ; / to the stack pointer
19106 000053CF 08E4 <1> or ah, ah
19107 <1> ; tstb u.uno / is the process # = 0

```



19108	000053D1	740D	<1>	jz	short swap_6 ; 'sysexit'
19109			<1>		; beq 1f / yes, kill process by overwriting
19110			<1>		; 02/09/2015
19111	000053D3	8925[7C740000]	<1>	mov	[u.usp], esp ; return address for 'syswait' & 'sleep'
19112			<1>		;
19113	000053D9	E834000000	<1>	call	wswap
19114			<1>		;jsr r0,wswap / write out core to disk
19115			<1>		; 31/08/2015
19116			<1>	;movzx	ebx, al ; New (running) process number
19117	000053DE	EB1C	<1>	jmp	short swap_7
19118			<1>	swap_6:	
19119			<1>		; 31/08/2015
19120			<1>		; Deallocate memory pages belong to the process
19121			<1>		; which is being terminated
19122			<1>		; 14/05/2015 ('sysexit')
19123			<1>		; Deallocate memory pages of the process
19124			<1>		; (Retro UNIX 386 v1 modification !)
19125			<1>		;
19126			<1>	; movzx	ebx, al
19127	000053E0	53	<1>	push	ebx
19128	000053E1	A1[D3740000]	<1>	mov	eax, [u.pgdir] ; page directory of the process
19129	000053E6	8B1D[D7740000]	<1>	mov	ebx, [u.ppgdir] ; page directory of the parent process
19130	000053EC	E87FDDFFFF	<1>	call	deallocate_page_dir
19131	000053F1	A1[CA740000]	<1>	mov	eax, [u.upage] ; 'user' structure page of the process
19132	000053F6	E814DEFFFF	<1>	call	deallocate_page
19133	000053FB	5B	<1>	pop	ebx
19134			<1>	swap_7: ;1:	
19135			<1>		; 02/09/2015
19136			<1>		; 31/08/2015
19137			<1>		; 14/05/2015
19138	000053FC	C0E302	<1>	shl	bl, 2 ; * 4
19139	000053FF	8B83[12720000]	<1>	mov	eax, [ebx+p.upage-4] ; the 'u' page of the new process
19140			<1>	;cli	
19141	00005405	E831000000	<1>	call	rswap
19142			<1>		; mov (sp)+,r1 / restore r1 to new process number
19143			<1>		; jsr r0,rswap / read new process into core
19144			<1>		; jsr r0,unpack / unpack the users stack from next
19145			<1>		; / to his program to its normal
19146			<1>		; 01/09/2015
19147			<1>	;mov	esp, [u.usp]
19148			<1>		; mov u.usp,sp / location; restore stack pointer to
19149			<1>		; / new process stack
19150			<1>		; mov (sp)+,r0 / put address of where the process
19151			<1>		; / that just got swapped in, left off.,
19152			<1>		; / i.e., transfer control to new process
19153			<1>	;sti	
19154			<1>	swap_8: ;2:	
19155			<1>		; RETRO UNIX 8086 v1 modification !
19156	0000540A	C605[BC740000]04	<1>	mov	byte [u.quant], time_count
19157			<1>		; movb \$30.,uquant / initialize process time quantum
19158	00005411	C3	<1>	ret	
19159			<1>		; rts r0 / return
19160			<1>		
19161			<1>	wswap: ; < swap out, swap to disk >	
19162			<1>		; 09/05/2015 (Retro UNIX 386 v1 - Beginning)
19163			<1>		; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
19164			<1>		; 'wswap' writes out the process that is in core onto its
19165			<1>		; appropriate disk area.
19166			<1>		;
19167			<1>		; Retro UNIX 386 v1 modification ->
19168			<1>		; User (u) structure content and the user's register content
19169			<1>		; will be copied to the process's/user's UPAGE (a page for
19170			<1>		; saving 'u' structure and user registers for task switching).
19171			<1>		; u.usp - points to kernel stack address which contains
19172			<1>		; user's registers while entering system call.
19173			<1>		; u.sp - points to kernel stack address
19174			<1>		; to return from system call -for IRET-.
19175			<1>		; [u.usp]+32+16 = [u.sp]
19176			<1>		; [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
19177			<1>		; edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
19178			<1>		;
19179			<1>		; Retro UNIX 8086 v1 modification ->
19180			<1>		; 'swap to disk' is replaced with 'change running segment'
19181			<1>		; according to 8086 cpu (x86 real mode) architecture.
19182			<1>		; pdp-11 was using 64KB uniform memory while IBM PC
19183			<1>		; compatibles was using 1MB segmented memory
19184			<1>		; in 8086/8088 times.
19185			<1>		;
19186			<1>		; INPUTS ->
19187			<1>		; u.break - points to end of program
19188			<1>		; u.usp - stack pointer at the moment of swap
19189			<1>		; core - beginning of process program
19190			<1>		; ecore - end of core
19191			<1>		; user - start of user parameter area
19192			<1>		; u.uno - user process number
19193			<1>		; p.dska - holds block number of process
19194			<1>		; OUTPUTS ->
19195			<1>		; swp I/O queue
19196			<1>		; p.break - negative word count of process
19197			<1>		; r1 - process disk address
19198			<1>		; r2 - negative word count
19199			<1>		;

```

19200 <1> ; RETRO UNIX 8086 v1 input/output:
19201 <1> ;
19202 <1> ; INPUTS ->
19203 <1> ; u.uno - process number (to be swapped out)
19204 <1> ; OUTPUTS ->
19205 <1> ; none
19206 <1> ;
19207 <1> ; ((Modified registers: ECX, ESI, EDI))
19208 <1> ;
19209 00005412 8B3D[CA740000] <1> mov edi, [u.upage] ; process's user (u) structure page addr
19210 00005418 B91C000000 <1> mov ecx, (U_SIZE + 3) / 4
19211 0000541D BE[78740000] <1> mov esi, user ; active user (u) structure
19212 00005422 F3A5 <1> rep movsd
19213 <1> ;
19214 00005424 8B35[7C740000] <1> mov esi, [u.usp] ; esp (system stack pointer,
19215 <1> ; points to user registers)
19216 0000542A 8B0D[78740000] <1> mov ecx, [u.sp] ; return address from the system call
19217 <1> ; (for IRET)
19218 <1> ; [u.sp] -> EIP (user)
19219 <1> ; [u.sp+4]-> CS (user)
19220 <1> ; [u.sp+8] -> EFLAGS (user)
19221 <1> ; [u.sp+12] -> ESP (user)
19222 <1> ; [u.sp+16] -> SS (user)
19223 00005430 29F1 <1> sub ecx, esi ; required space for user registers
19224 00005432 83C114 <1> add ecx, 20 ; +5 dwords to return from system call
19225 <1> ; (for IRET)
19226 00005435 C1E902 <1> shr ecx, 2
19227 00005438 F3A5 <1> rep movsd
19228 0000543A C3 <1> retn
19229 <1>
19230 <1> ; Original UNIX v1 'wswap' routine:
19231 <1> ; wswap:
19232 <1> ; mov *$30,u.emt / determines handling of emts
19233 <1> ; mov *$10,u.ilgins / determines handling of
19234 <1> ; / illegal instructions
19235 <1> ; mov u.break,r2 / put process program break address in r2
19236 <1> ; inc r2 / add 1 to it
19237 <1> ; bic $1,r2 / make it even
19238 <1> ; mov r2,u.break / set break to an even location
19239 <1> ; mov u.usp,r3 / put users stack pointer
19240 <1> ; / at moment of swap in r3
19241 <1> ; cmp r2,$core / is u.break less than $core
19242 <1> ; blos 2f / yes
19243 <1> ; cmp r2,r3 / no, is (u.break) greater than stack ptr.
19244 <1> ; bhis 2f / yes
19245 <1> ; 1:
19246 <1> ; mov (r3)+,(r2)+ / no, pack stack next to users program
19247 <1> ; cmp r3,$core / has stack reached end of core
19248 <1> ; bne 1b / no, keep packing
19249 <1> ; br 1f / yes
19250 <1> ; 2:
19251 <1> ; mov $core,r2 / put end of core in r2
19252 <1> ; 1:
19253 <1> ; sub $user,r2 / get number of bytes to write out
19254 <1> ; / (user up to end of stack gets written out)
19255 <1> ; neg r2 / make it negative
19256 <1> ; asr r2 / change bytes to words (divide by 2)
19257 <1> ; mov r2,swp+4 / word count
19258 <1> ; movb u.uno,r1 / move user process number to r1
19259 <1> ; asl r1 / x2 for index
19260 <1> ; mov r2,p.break-2(r1) / put negative of word count
19261 <1> ; / into the p.break table
19262 <1> ; mov p.dska-2(r1),r1 / move disk address of swap area
19263 <1> ; / for process to r1
19264 <1> ; mov r1,swp+2 / put processes dska address in swp+2
19265 <1> ; / (block number)
19266 <1> ; bis $1000,swp / set it up to write (set bit 9)
19267 <1> ; jsr r0,ppoke / write process out on swap area of disk
19268 <1> ; 1:
19269 <1> ; tstb swp+1 / is 1t done writing?
19270 <1> ; bne 1b / no, wait
19271 <1> ; rts r0 / yes, return to swap
19272 <1>
19273 <1> rswap: ; < swap in, swap from disk >
19274 <1> ; 15/09/2015
19275 <1> ; 28/08/2015
19276 <1> ; 14/05/2015
19277 <1> ; 09/05/2015 (Retro UNIX 386 v1 - Beginning)
19278 <1> ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
19279 <1> ; 'rswap' reads a process whose number is in r1,
19280 <1> ; from disk into core.
19281 <1> ;
19282 <1> ; Retro UNIX 386 v1 modification ->
19283 <1> ; User (u) structure content and the user's register content
19284 <1> ; will be restored from process's/user's UPAGE (a page for
19285 <1> ; saving 'u' structure and user registers for task switching).
19286 <1> ; u.usp - points to kernel stack address which contains
19287 <1> ; user's registers while entering system call.
19288 <1> ; u.sp - points to kernel stack address
19289 <1> ; to return from system call -for IRET-.
19290 <1> ; [u.usp]+32+16 = [u.sp]
19291 <1> ; [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,

```

```

19292 <1> ; edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
19293 <1> ;
19294 <1> ; RETRO UNIX 8086 v1 modification ->
19295 <1> ; 'swap to disk' is replaced with 'change running segment'
19296 <1> ; according to 8086 cpu (x86 real mode) architecture.
19297 <1> ; pdp-11 was using 64KB uniform memory while IBM PC
19298 <1> ; compatibles was using 1MB segmented memory
19299 <1> ; in 8086/8088 times.
19300 <1> ;
19301 <1> ; INPUTS ->
19302 <1> ; r1 - process number of process to be read in
19303 <1> ; p.break - negative of word count of process
19304 <1> ; p.dska - disk address of the process
19305 <1> ; u.emt - determines handling of emt's
19306 <1> ; u.ilgins - determines handling of illegal instructions
19307 <1> ; OUTPUTS ->
19308 <1> ; 8 = (u.ilgins)
19309 <1> ; 24 = (u.emt)
19310 <1> ; swp - bit 10 is set to indicate read
19311 <1> ; (bit 15=0 when reading is done)
19312 <1> ; swp+2 - disk block address
19313 <1> ; swp+4 - negative word count
19314 <1> ; ((swp+6 - address of user structure))
19315 <1> ;
19316 <1> ; RETRO UNIX 8086 v1 input/output:
19317 <1> ;
19318 <1> ; INPUTS ->
19319 <1> ; AL - new process number (to be swapped in)
19320 <1> ; OUTPUTS ->
19321 <1> ; none
19322 <1> ;
19323 <1> ; ((Modified registers: EAX, ECX, ESI, EDI, ESP))
19324 <1> ;
19325 <1> ; Retro UNIX 386 v1 - modification ! 14/05/2015
19326 0000543B 89C6 <1> mov esi, eax ; process's user (u) structure page addr
19327 0000543D B91C000000 <1> mov ecx, (U_SIZE + 3) / 4
19328 00005442 BF[78740000] <1> mov edi, user ; active user (u) structure
19329 00005447 F3A5 <1> rep movsd
19330 00005449 58 <1> pop eax ; 15/09/2015, 'rswap' return address
19331 0000544A 8B3D[7C740000] <1> mov edi, [u.usp] ; esp (system stack pointer,
19332 <1> ; points to user registers)
19333 00005450 8B0D[78740000] <1> mov ecx, [u.sp] ; return address from the system call
19334 <1> ; (for IRET)
19335 <1> ; [u.sp] -> EIP (user)
19336 <1> ; [u.sp+4]-> CS (user)
19337 <1> ; [u.sp+8] -> EFLAGS (user)
19338 <1> ; [u.sp+12] -> ESP (user)
19339 <1> ; [u.sp+16] -> SS (user)
19340 <1> ; 28/08/2015
19341 00005456 29F9 <1> sub ecx, edi ; required space for user registers
19342 00005458 83C114 <1> add ecx, 20 ; +5 dwords to return from system call
19343 <1> ; (for IRET)
19344 0000545B C1E902 <1> shr ecx, 2
19345 0000545E F3A5 <1> rep movsd
19346 00005460 8B25[7C740000] <1> mov esp, [u.usp] ; 15/09/2015
19347 00005466 50 <1> push eax ; 15/09/2015 'rswap' return address
19348 00005467 C3 <1> retn
19349 <1>
19350 <1> ; Original UNIX v1 'rswap' and 'unpack' routines:
19351 <1> ;rswap:
19352 <1> ; asl r1 / process number x2 for index
19353 <1> ; mov p.break-2(r1), swp+4 / word count
19354 <1> ; mov p.dska-2(r1),swp+2 / disk address
19355 <1> ; bis $2000,swp / read
19356 <1> ; jsr r0,ppoke / read it in
19357 <1> ; 1:
19358 <1> ; tstb swp+1 / done
19359 <1> ; bne 1b / no, wait for bit 15 to clear (inhibit bit)
19360 <1> ; mov u.emt,$30 / yes move these
19361 <1> ; mov u.ilgins,$10 / back
19362 <1> ; rts r0 / return
19363 <1>
19364 <1> ;unpack: ; / move stack back to its normal place
19365 <1> ; mov u.break,r2 / r2 points to end of user program
19366 <1> ; cmp r2,$core / at beginning of user program yet?
19367 <1> ; blos 2f / yes, return
19368 <1> ; cmp r2,u.usp / is break_above the stack pointer
19369 <1> ; / before swapping
19370 <1> ; bhis 2f / yes, return
19371 <1> ; mov $core,r3 / r3 points to end of core
19372 <1> ; add r3,r2
19373 <1> ; sub u.usp,r2 / end of users stack is in r2
19374 <1> ; 1:
19375 <1> ; mov -(r2),-(r3) / move stack back to its normal place
19376 <1> ; cmp r2,u.break / in core
19377 <1> ; bne 1b
19378 <1> ; 2:
19379 <1> ; rts r0
19380 <1>
19381 <1> putlu:
19382 <1> ; 12/09/2015
19383 <1> ; 02/09/2015

```

```

19384      <1> ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
19385      <1> ; 15/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
19386      <1> ; 'putlu' is called with a process number in r1 and a pointer
19387      <1> ; to lowest priority Q (runq+4) in r2. A link is created from
19388      <1> ; the last process on the queue to process in r1 by putting
19389      <1> ; the process number in r1 into the last process's link.
19390      <1> ;
19391      <1> ; INPUTS ->
19392      <1> ;   r1 - user process number
19393      <1> ;   r2 - points to lowest priority queue
19394      <1> ;   p.dska - disk address of the process
19395      <1> ;   u.emt - determines handling of emt's
19396      <1> ;   u.ilgins - determines handling of illegal instructions
19397      <1> ; OUTPUTS ->
19398      <1> ;   r3 - process number of last process on the queue upon
19399      <1> ;   entering putlu
19400      <1> ;   p.link-1 + r3 - process number in r1
19401      <1> ;   r2 - points to lowest priority queue
19402      <1> ;
19403      <1> ; ((Modified registers: EDX, EBX))
19404      <1> ;
19405      <1> ; / r1 = user process no.; r2 points to lowest priority queue
19406      <1>
19407      <1> ; eBX = r2
19408      <1> ; eAX = r1 (AL=r1b)
19409      <1>
19410      00005468 BB[72740000] <1> mov     ebx, runq
19411      0000546D 0FB613      <1> movzx   edx, byte [ebx]
19412      00005470 43         <1> inc     ebx
19413      00005471 20D2      <1> and     dl, dl
19414      <1> ; tstb (r2)+ / is queue empty?
19415      00005473 740A      <1> jz      short putlu_1
19416      <1> ; beq 1f / yes, branch
19417      00005475 8A13      <1> mov     dl, [ebx] ; 12/09/2015
19418      <1> ; movb (r2),r3 / no, save the "last user" process number
19419      <1> ; / in r3
19420      00005477 8882[F5710000] <1> mov     [edx+p.link-1], al
19421      <1> ; movb r1,p.link-1(r3) / put pointer to user on
19422      <1> ; / "last users" link
19423      0000547D EB03      <1> jmp     short putlu_2
19424      <1> ; br 2f /
19425      <1> putlu_1: ; 1:
19426      0000547F 8843FF      <1> mov     [ebx-1], al
19427      <1> ; movb r1,-1(r2) / user is only user;
19428      <1> ; / put process no. at beginning and at end
19429      <1> putlu_2: ; 2:
19430      00005482 8803      <1> mov     [ebx], al
19431      <1> ; movb r1,(r2) / user process in r1 is now the last entry
19432      <1> ; / on the queue
19433      00005484 88C2      <1> mov     dl, al
19434      00005486 88B2[F5710000] <1> mov     [edx+p.link-1], dh ; 0
19435      <1> ; dec r2 / restore r2
19436      0000548C C3         <1> retn
19437      <1> ; rts r0
19438      <1>
19439      <1> ;copyz:
19440      <1> ;   mov     r1,-(sp) / put r1 on stack
19441      <1> ;   mov     r2,-(sp) / put r2 on stack
19442      <1> ;   mov     (r0)+,r1
19443      <1> ;   mov     (r0)+,r2
19444      <1> ;1:
19445      <1> ;   clr     (r1)+ / clear all locations between r1 and r2
19446      <1> ;   cmp     r1,r2
19447      <1> ;   blo     1b
19448      <1> ;   mov     (sp)+,r2 / restore r2
19449      <1> ;   mov     (sp)+,r1 / restore r1
19450      <1> ;   rts     r0
19451      <1>
19452      <1> idle:
19453      <1> ; 01/09/2015
19454      <1> ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
19455      <1> ; 10/04/2013 - 23/10/2013 (Retro UNIX 8086 v1)
19456      <1> ; (idle & wait loop)
19457      <1> ; Retro Unix 8086 v1 modification on original UNIX v1
19458      <1> ; idle procedure!
19459      <1> ;
19460      <1> ; 01/09/2015
19461      0000548D FB         <1> sti
19462      <1> ; 29/07/2013
19463      0000548E F4         <1> hlt
19464      0000548F 90         <1> nop ; 10/10/2013
19465      00005490 90         <1> nop
19466      00005491 90         <1> nop
19467      <1> ; 23/10/2013
19468      00005492 90         <1> nop
19469      00005493 90         <1> nop
19470      00005494 90         <1> nop
19471      00005495 90         <1> nop
19472      00005496 C3         <1> retn
19473      <1>
19474      <1> ;mov *$ps,-(sp) / save ps on stack
19475      <1> ;clr *$ps / clear ps

```

```

19476 <1> ;mov clockp,-(sp) / save clockp on stack
19477 <1> ;mov (r0)+,clockp / arg to idle in clockp
19478 <1> ;1 / wait for interrupt
19479 <1> ;mov (sp)+,clockp / restore clockp, ps
19480 <1> ;mov (sp)+,*$ps
19481 <1> ;rts r0
19482 <1>
19483 <1> clear:
19484 <1> ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
19485 <1> ; 09/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
19486 <1> ; 'clear' zero's out of a block (whose block number is in r1)
19487 <1> ; on the current device (cdev)
19488 <1> ;
19489 <1> ; INPUTS ->
19490 <1> ; r1 - block number of block to be zeroed
19491 <1> ; cdev - current device number
19492 <1> ; OUTPUTS ->
19493 <1> ; a zeroed I/O buffer onto the current device
19494 <1> ; r1 - points to last entry in the I/O buffer
19495 <1> ;
19496 <1> ; ((AX = R1)) input/output
19497 <1> ; (Retro UNIX Prototype : 18/11/2012 - 14/11/2012, UNIXCOPY.ASM)
19498 <1> ; ((Modified registers: EDX, ECX, EBX, ESI, EDI, EBP))
19499 <1>
19500 00005497 E8AB0D0000 <1> call wslot
19501 <1> ; jsr r0,wslot / get an I/O buffer set bits 9 and 15 in first
19502 <1> ; / word of I/O queue r5 points to first data word in buffer
19503 0000549C 89DF <1> mov edi, ebx ; r5
19504 0000549E 89C2 <1> mov edx, eax
19505 000054A0 B980000000 <1> mov ecx, 128
19506 <1> ; mov $256.,r3
19507 000054A5 31C0 <1> xor eax, eax
19508 000054A7 F3AB <1> rep stosd
19509 000054A9 89D0 <1> mov eax, edx
19510 <1> ; 1:
19511 <1> ; clr (r5)+ / zero data word in buffer
19512 <1> ; dec r3
19513 <1> ; bgt 1b / branch until all data words in buffer are zero
19514 000054AB E8B30D0000 <1> call dskwr
19515 <1> ; jsr r0,dskwr / write zeroed buffer area out onto physical
19516 <1> ; / block specified in r1
19517 <1> ; eAX (r1) = block number
19518 000054B0 C3 <1> retn
19519 <1> ; rts r0
19520 %include 'u4.s' ; 15/04/2015
19521 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS4.INC
19522 <1> ; Last Modification: 14/10/2015
19523 <1> ; -----
19524 <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
19525 <1> ; (v0.1 - Beginning: 11/07/2012)
19526 <1> ;
19527 <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
19528 <1> ; (Original) Source Code by Ken Thompson (1971-1972)
19529 <1> ; <Bell Laboratories (17/3/1972)>
19530 <1> ; <Preliminary Release of UNIX Implementation Document>
19531 <1> ;
19532 <1> ; Retro UNIX 8086 v1 - U4.ASM (04/07/2014) //// UNIX v1 -> u4.s
19533 <1> ;
19534 <1> ; *****
19535 <1>
19536 <1> ;setisp:
19537 <1> ;mov r1,-(sp)
19538 <1> ;mov r2,-(sp)
19539 <1> ;mov r3,-(sp)
19540 <1> ;mov clockp,-(sp)
19541 <1> ;mov $s.syst+2,clockp
19542 <1> ;jmp (r0)
19543 <1>
19544 <1> clock: ; / interrupt from 60 cycle clock
19545 <1>
19546 <1> ; 14/10/2015
19547 <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
19548 <1> ; 07/12/2013 - 10/04/2014 (Retro UNIX 8086 v1)
19549 <1>
19550 <1> ;mov r0,-(sp) / save r0
19551 <1> ;tst *$1ks / restart clock?
19552 <1> ;mov $s.time+2,r0 / increment the time of day
19553 <1> ;inc (r0)
19554 <1> ;bne 1f
19555 <1> ;inc -(r0)
19556 <1> ;1:
19557 <1> ;mov clockp,r0 / increment appropriate time category
19558 <1> ;inc (r0)
19559 <1> ;bne 1f
19560 <1> ;inc -(r0)
19561 <1> ;1:
19562 <1> ;;;;;;;;;;;;;;
19563 <1>
19564 000054B1 803D[BC740000]00 <1> cmp byte [u.quant], 0
19565 000054B8 772C <1> ja short clk_1
19566 <1> ;
19567 000054BA 803D[77740000]FF <1> cmp byte [sysflg], 0FFh ; user or system space ?

```

```

19568 000054C1 7529      <1>   jne    short clk_2 ; system space (sysflg <> 0FFh)
19569 000054C3 803D[C9740000]01 <1>   cmp     byte [u.uno], 1 ; /etc/init ?
19570 000054CA 761A      <1>   jna     short clk_1 ; yes, do not swap out
19571 000054CC 66833D[BE740000]00 <1>   cmp     word [u.intr], 0
19572 000054D4 7616      <1>   jna     short clk_2
19573                               <1> clk_0:
19574                               <1>   ; 14/10/2015
19575 000054D6 FE05[77740000] <1>   inc     byte [sysflg] ; Now, we are in system space
19576 000054DC 58        <1>   pop     eax ; return address to the timer interrupt
19577                               <1>   ;
19578 000054DD B020      <1>   MOV     AL,EOI ; GET END OF INTERRUPT MASK
19579                               <1>   ;CLI ; DISABLE INTERRUPTS TILL STACK CLEARED
19580 000054DF E620      <1>   OUT     INTA00,AL ; END OF INTERRUPT TO 8259 - 1
19581                               <1>   ;
19582 000054E1 E912ECFFFF <1>   jmp     sysrelease ; 'sys release' by clock/timer
19583                               <1> clk_1:
19584 000054E6 FE0D[BC740000] <1>   dec     byte [u.quant]
19585                               <1> clk_2:
19586 000054EC C3        <1>   retn    ; return to (hardware) timer interrupt routine
19587                               <1>
19588                               <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
19589                               <1>
19590                               <1>   ;mov     $uquant,r0 / decrement user time quantum
19591                               <1>   ;decb     (r0)
19592                               <1>   ;bge     1f / if less than 0
19593                               <1>   ;clrb     (r0) / make it 0
19594                               <1> ;1: / decrement time out counts return now if priority was not 0
19595                               <1>   ;cmp     4(sp),$200 / ps greater than or equal to 200
19596                               <1>   ;bge     2f / yes, check time outs
19597                               <1>   ;tstb     (r0) / no, user timed out?
19598                               <1>   ;bne     1f / no
19599                               <1>   ;cmpb     sysflg,$-1 / yes, are we outside the system?
19600                               <1>   ;bne     1f / no, 1f
19601                               <1>   ;mov     (sp)+,r0 / yes, put users r0 in r0
19602                               <1>   ;sys     0 / sysrele
19603                               <1>   ;rti
19604                               <1> ;2: / priority is high so just decrement time out counts
19605                               <1>   ;mov     $toutt,r0 / r0 points to beginning of time out table
19606                               <1> ;2:
19607                               <1>   ;tstb     (r0) / is the time out?
19608                               <1>   ;beq     3f / yes, 3f (get next entry)
19609                               <1>   ;decb     (r0) / no, decrement the time
19610                               <1>   ;bne     3f / isit zero now?
19611                               <1>   ;incb     (r0) / yes, increment the time
19612                               <1> ;3:
19613                               <1>   ;inc     r0 / next entry
19614                               <1>   ;cmp     r0,$touts / end of toutt table?
19615                               <1>   ;blo     2b / no, check this entry
19616                               <1>   ;mov     (sp)+,r0 / yes, restore r0
19617                               <1>   ;rti / return from interrupt
19618                               <1> ;1: / decrement time out counts; if 0 call subroutine
19619                               <1>   ;mov     (sp)+,r0 / restore r0
19620                               <1>   ;mov     $240,$ps / set processor priority to 5
19621                               <1>   ;jsr     r0,setisp / save registers
19622                               <1>   ;mov     $touts-toutt-1,r0 / set up r0 as index to decrement thru
19623                               <1>   ; / the table
19624                               <1> ;1:
19625                               <1>   ;tstb     toutt(r0) / is the time out for this entry
19626                               <1>   ;beq     2f / yes
19627                               <1>   ;decb     toutt(r0) / no, decrement the time
19628                               <1>   ;bne     2f / is the time 0, now
19629                               <1>   ;asl     r0 / yes, 2 x r0 to get word index for tout entry
19630                               <1>   ;jsr     r0,$touts(r0) / go to appropriate routine specified in this
19631                               <1>   ;asr     r0 / touts entry; set r0 back to toutt index
19632                               <1> ;2:
19633                               <1>   ;dec     r0 / set up r0 for next entry
19634                               <1>   ;bge     1b / finished? , no, go back
19635                               <1>   ;br     retisp / yes, restore registers and do a rti
19636                               <1>
19637                               <1> ;retisp:
19638                               <1>   ;mov     (sp)+,clockp / pop values before interrupt off the stack
19639                               <1>   ;mov     (sp)+,r3
19640                               <1>   ;mov     (sp)+,r2
19641                               <1>   ;mov     (sp)+,r1
19642                               <1>   ;mov     (sp)+,r0
19643                               <1>   ;rti / return from interrupt
19644                               <1>
19645                               <1>
19646                               <1> wakeup: ; / wakeup processes waiting for an event
19647                               <1>   ; / by linking them to the queue
19648                               <1>   ;
19649                               <1>   ; 15/09/2015
19650                               <1>   ; 29/06/2015
19651                               <1>   ; 15/04/2015 (Retro UNIX 386 v1 - Beginning)
19652                               <1>   ;
19653                               <1>   ; 15/05/2013 - 02/06/2014
19654                               <1>   ; Retro UNIX 8086 v1 modification !
19655                               <1>   ; (Process/task switching routine by using
19656                               <1>   ; Retro UNIX 8086 v1 keyboard interrupt output.)
19657                               <1>   ;
19658                               <1>   ; In original UNIX v1, 'wakeup' is called to wake the process
19659                               <1>   ; sleeping in the specified wait channel by creating a link

```

```

19660 <1> ; to it from the last user process on the run queue.
19661 <1> ; If there is no process to wake up, nothing happens.
19662 <1> ;
19663 <1> ; In Retro UNIX 8086 v1, Int 09h keyboard interrupt will set
19664 <1> ; 'switching' status of the current process (owns current tty)
19665 <1> ; (via alt + function keys) to a process which has highest
19666 <1> ; priority (on run queue) on the requested tty (0 to 7, except
19667 <1> ; 8 and 9 which are tty identifiers of COM1, COM2 serial ports)
19668 <1> ; as it's console tty. (NOTE: 'p.ttyc' is used to set console
19669 <1> ; tty for tty switching by keyboard.)
19670 <1> ;
19671 <1> ; INPUT ->
19672 <1> ; AL = wait channel (r3) ('tty number' for now)
19673 <1> ; ;;EBX = Run queue (r2) offset
19674 <1> ;
19675 <1> ; ((modified registers: EAX, EBX))
19676 <1> ;
19677 000054ED 0FB6D8 <1> movzx ebx, al ; 29/06/2015
19678 000054F0 81C3[08710000] <1> add ebx, wlist
19679 000054F6 8A03 <1> mov al, [ebx] ; waiting list (waiting process number)
19680 000054F8 20C0 <1> and al, al
19681 000054FA 7424 <1> jz short wa0 ; nothing to wakeup
19682 <1> ;
19683 000054FC 30E4 <1> xor ah, ah
19684 000054FE 8825[BC740000] <1> mov [u.quant], ah ; 0 ; time quantum = 0
19685 00005504 8823 <1> mov [ebx], ah ; 0 ; zero wait channel entry
19686 <1> ; 15/09/2015
19687 00005506 0FB6D8 <1> movzx ebx, al
19688 00005509 88A3[E5710000] <1> mov [ebx+p.waitc-1], ah ; 0
19689 0000550F FEC4 <1> inc ah
19690 00005511 88A3[05720000] <1> mov byte [ebx+p.stat-1], ah ; 1 ; SRUN
19691 <1> ;
19692 00005517 57 <1> push edi
19693 00005518 52 <1> push edx
19694 00005519 E84AFFFFFF <1> call putlu
19695 0000551E 5A <1> pop edx
19696 0000551F 5F <1> pop edi
19697 <1> wa0:
19698 00005520 C3 <1> retn
19699 <1>
19700 <1> sleep:
19701 <1> ; 15/09/2015
19702 <1> ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
19703 <1> ;
19704 <1> ; 09/05/2013 - 20/03/2014
19705 <1> ;
19706 <1> ; Retro UNIX 8086 v1 modification !
19707 <1> ; (Process/task switching and quit routine by using
19708 <1> ; Retro UNIX 8086 v1 keyboard interrupt output.)
19709 <1> ;
19710 <1> ; In original UNIX v1, 'sleep' is called to wait for
19711 <1> ; tty and tape output or input becomes available
19712 <1> ; and process is put on waiting channel and swapped out,
19713 <1> ; then -when the tty or tape is ready to write or read-
19714 <1> ; 'wakeup' gets process back to active swapped-in status.)
19715 <1> ;
19716 <1> ; In Retro UNIX 8086 v1, Int 1Bh ctrl+brk interrupt and
19717 <1> ; Int 09h keyboard interrupt will set 'quit' or 'switching'
19718 <1> ; status of the current process also INT 1Ch will count down
19719 <1> ; 'uquant' value and INT 09h will redirect scancode of keystroke
19720 <1> ; to tty buffer of the current process and kernel will get
19721 <1> ; user input by using tty buffer of the current process
19722 <1> ; (instead of standard INT 16h interrupt).
19723 <1> ; TTY output will be redirected to related video page of text mode
19724 <1> ; (INT 10h will be called with different video page depending
19725 <1> ; on tty assignment of the active process: 0 to 7 for
19726 <1> ; pseudo screens.)
19727 <1> ;
19728 <1> ; In Retro UNIX 8086 v1, 'sleep' will be called to wait for
19729 <1> ; a keystroke from keyboard or wait for reading or writing
19730 <1> ; characters/data on serial port(s).
19731 <1> ;
19732 <1> ; Character/Terminal input/output through COM1 and COM2 will be
19733 <1> ; performed by related routines in addition to pseudo TTY routines.
19734 <1> ;
19735 <1> ; R1 = AH = wait channel (0-9 for TTYS) ; 05/10/2013 (22/09/2013)
19736 <1> ;
19737 <1> ;; 05/10/2013
19738 <1> ;10/12/2013
19739 <1> ;cmp byte [u.uno], 1
19740 <1> ;ja short sleep0
19741 <1> ;retn
19742 <1>
19743 <1> ; 20/03/2014
19744 <1> ;mov bx, [runq]
19745 <1> ;cmp bl, bh
19746 <1> ;jne short sleep0
19747 <1> ; 25/02/2014
19748 <1> ;cmp word ptr [runq], 0
19749 <1> ;ja short sleep0
19750 <1> ;retn
19751 <1> sleep0:

```

```

19752                                <1> ;
19753 00005521 E854000000          <1> call isintr
19754 00005526 0F8570EBFFFF        <1> jnz sysret
19755                                <1> ; / wait for event
19756                                <1> ; jsr r0,isintr / check to see if interrupt
19757                                <1> ; / or quit from user
19758                                <1> ; br 2f / something happened
19759                                <1> ; / yes, his interrupt so return
19760                                <1> ; / to user
19761                                <1>
19762                                <1> ; 30/06/2015
19763 0000552C 0FB6DC              <1> movzx ebx, ah ; 30/06/2015
19764 0000552F 81C3[08710000]      <1> add ebx, wlist
19765 00005535 8A03                  <1> mov al, [ebx]
19766 00005537 20C0                  <1> and al, al
19767 00005539 7407                  <1> jz short sleep1
19768 0000553B 53                    <1> push ebx
19769 0000553C E827FFFFFF            <1> call putlu
19770 00005541 5B                    <1> pop ebx
19771                                <1> sleep1:
19772 00005542 A0[C9740000]          <1> mov al, [u.uno]
19773 00005547 8803                  <1> mov [ebx], al ; put the process number
19774                                <1> ; in the wait channel
19775                                <1> ; mov (r0)+,r1 / put number of wait channel in r1
19776                                <1> ; movb wlist(r1),-(sp) / put old process number in there,
19777                                <1> ; / on the stack
19778                                <1> ; movb u.uno,wlist(r1) / put process number of process
19779                                <1> ; / to put to sleep in there
19780                                <1> ; 15/09/2015
19781 00005549 0FB6D8              <1> movzx ebx, al
19782 0000554C C683[05720000]04      <1> mov byte [ebx+p.stat-1], 4 ; SSLEEP
19783 00005553 FEC4                  <1> inc ah
19784 00005555 88A3[E5710000]        <1> mov [ebx+p.waitc-1], ah ; wait channel + 1
19785                                <1> ;
19786 0000555B 66FF35[66740000]      <1> push word [cdev]
19787                                <1> ; mov cdev,-(sp) / nothing happened in isintr so
19788 00005562 E833FEFFFFFF          <1> call swap
19789                                <1> ; jsr r0,swap / swap out process that needs to sleep
19790 00005567 668F05[66740000]      <1> pop word [cdev]
19791                                <1> ; mov (sp)+,cdev / restore device
19792 0000556E E807000000            <1> call isintr
19793                                <1> ; 22/09/2013
19794 00005573 0F8523EBFFFF        <1> jnz sysret
19795                                <1> ; jsr r0,isintr / check for interrupt of new process
19796                                <1> ; br 2f / yes, return to new user
19797                                <1> ; movb (sp)+,r1 / no, r1 = old process number that was
19798                                <1> ; / originally on the wait channel
19799                                <1> ; beq 1f / if 0 branch
19800                                <1> ; mov $runq+4,r2 / r2 points to lowest priority queue
19801                                <1> ; mov $300,$ps / processor priority = 6
19802                                <1> ; jsr r0,putlu / create link to old process number
19803                                <1> ; clr *$ps / clear the status; process priority = 0
19804                                <1> ;1:
19805 00005579 C3                    <1> retn
19806                                <1> ; rts r0 / return
19807                                <1> ;2:
19808                                <1> ;;jmpsysret
19809                                <1> ; jmp sysret / return to user
19810                                <1>
19811                                <1> isintr:
19812                                <1> ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
19813                                <1> ;
19814                                <1> ; 09/05/2013 - 30/05/2014
19815                                <1> ;
19816                                <1> ; Retro UNIX 8086 v1 modification !
19817                                <1> ; (Process/task switching and quit routine by using
19818                                <1> ; Retro UNIX 8086 v1 keyboard interrupt output.))
19819                                <1> ;
19820                                <1> ; Retro UNIX 8086 v1 modification:
19821                                <1> ; 'isintr' checks if user interrupt request is enabled
19822                                <1> ; and there is a 'quit' request by user;
19823                                <1> ; otherwise, 'isintr' will return with zf=1 that means
19824                                <1> ; "nothing to do". (20/10/2013)
19825                                <1> ;
19826                                <1> ; 20/10/2013
19827 0000557A 66833D[B0740000]00    <1> cmp word [u.ttyp], 0 ; has process got a tty ?
19828 00005582 7622                  <1> jna short isintr2 ; retn
19829                                <1> ; 03/09/2013
19830                                <1> ; (nothing to do)
19831                                <1> ;retn
19832                                <1> ; 22/09/2013
19833 00005584 66833D[BE740000]00    <1> cmp word [u.intr], 0
19834 0000558C 7618                  <1> jna short isintr2 ; retn
19835                                <1> ; 30/05/2014
19836 0000558E 6650                  <1> push ax
19837 00005590 66A1[C0740000]          <1> mov ax, [u.quit]
19838 00005596 6609C0                  <1> or ax, ax ; 0 ?
19839 00005599 7409                  <1> jz short isintr1 ; zf = 1
19840 0000559B 6683F8FE              <1> cmp ax, 0FFFEh ; 'ctrl + brk' check
19841 0000559F 7703                  <1> ja short isintr1 ; 0FFFFh, zf = 0
19842 000055A1 6631C0                  <1> xor ax, ax ; zf = 1
19843                                <1> isintr1:

```



```

19844 000055A4 6658      <1>  pop  ax
19845                    <1>  isintr2: ; 22/09/2013
19846                    <1>  ; zf=1 -> nothing to do
19847 000055A6 C3        <1>  retn
19848                    <1>
19849                    <1>  ; UNIX v1 original 'isintr' routine...
19850                    <1>      ;mov    r1,-(sp) / put number of wait channel on the stack
19851                    <1>      ;mov    r2,-(sp) / save r2
19852                    <1>      ;mov    u.ttyp,r1 / r1 = pointer to buffer of process control
19853                    <1>      ;      / typewriter
19854                    <1>      ;beq    1f / if 0, do nothing except skip return
19855                    <1>      ;movb   6(r1),r1 / put interrupt char in the tty buffer in r1
19856                    <1>      ;beq    1f / if its 0 do nothing except skip return
19857                    <1>      ;cmp    r1,$177 / is interrupt char = delete?
19858                    <1>      ;bne    3f / no, so it must be a quit (fs)
19859                    <1>      ;tst    u.intr / yes, value of u.intr determines handling
19860                    <1>      ;      / of interrupts
19861                    <1>      ;bne    2f / if not 0, 2f. If zero do nothing.
19862                    <1>      ;1:
19863                    <1>      ;tst    (r0)+ / bump r0 past system return (skip)
19864                    <1>      ;4:
19865                    <1>      ;mov    (sp)+,r2 / restore r1 and r2
19866                    <1>      ;mov    (sp)+,r1
19867                    <1>      ;rts    r0
19868                    <1>      ;3: / interrupt char = quit (fs)
19869                    <1>      ;tst    u.quit / value of u.quit determines handling of quits
19870                    <1>      ;beq    1b / u.quit = 0 means do nothing
19871                    <1>      ;2: / get here because either u.intr <> 0 or u.qult <> 0
19872                    <1>      ;mov    $tty+6,r1 / move pointer to tty block into r1
19873                    <1>      ;1: / find process control tty entry in tty block
19874                    <1>      ;cmp    (r1),u.ttyp / is this the process control tty buffer?
19875                    <1>      ;beq    1f / block found go to 1f
19876                    <1>      ;add    $8,r1 / look at next tty block
19877                    <1>      ;cmp    r1,$tty+[ntty*8]+6 / are we at end of tty blocks
19878                    <1>      ;blo    1b / no
19879                    <1>      ;br     4b / no process control tty found so go to 4b
19880                    <1>      ;1:
19881                    <1>      ;mov    $240,*$ps / set processor priority to 5
19882                    <1>      ;movb   -3(r1),0f / load getc call argument; character llst
19883                    <1>      ;      / identifier
19884                    <1>      ;inc    0f / increment
19885                    <1>      ;1:
19886                    <1>      ;jsr    r0,getc; 0:... / erase output char list for control
19887                    <1>      ;      br 4b / process tty. This prevents a line of stuff
19888                    <1>      ;      / being typed out after you hit the interrupt
19889                    <1>      ;      / key
19890                    <1>      ;br     1b
19891                    %include 'u5.s' ; 03/06/2015
19892                    <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS5.INC
19893                    <1> ; Last Modification: 14/11/2015
19894                    <1> ; -----
19895                    <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
19896                    <1> ; (v0.1 - Beginning: 11/07/2012)
19897                    <1> ;
19898                    <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
19899                    <1> ; (Original) Source Code by Ken Thompson (1971-1972)
19900                    <1> ; <Bell Laboratories (17/3/1972)>
19901                    <1> ; <Preliminary Release of UNIX Implementation Document>
19902                    <1> ;
19903                    <1> ; Retro UNIX 8086 v1 - U5.ASM (07/08/2013) //// UNIX v1 -> u5.s
19904                    <1> ;
19905                    <1> ; *****
19906                    <1>
19907                    <1> mget:
19908                    <1> ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
19909                    <1> ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
19910                    <1> ;
19911                    <1> ; Get existing or (allocate) a new disk block for file
19912                    <1> ;
19913                    <1> ; INPUTS ->
19914                    <1> ;   u.fofp (file offset pointer)
19915                    <1> ;   inode
19916                    <1> ;   u.off (file offset)
19917                    <1> ; OUTPUTS ->
19918                    <1> ;   r1 (physical block number)
19919                    <1> ;   r2, r3, r5 (internal)
19920                    <1> ;
19921                    <1> ; ((AX = R1)) output
19922                    <1> ;   (Retro UNIX Prototype : 05/03/2013 - 14/11/2012, UNIXCOPY.ASM)
19923                    <1> ;   ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
19924                    <1>
19925                    <1>      ; mov *u.fofp,mq / file offset in mq
19926                    <1>      ; clr ac / later to be high sig
19927                    <1>      ; mov $-8,lsh / divide ac/mq by 256.
19928                    <1>      ; mov mq,r2
19929                    <1>      ; bit $10000,i.flgs / lg/sm is this a large or small file
19930                    <1>      ; bne 4f / branch for large file
19931                    <1> mget_0:
19932 000055A7 8B35[90740000] <1>      mov    esi, [u.fofp]
19933 000055AD 0FB65E01        <1>      movzx   ebx, byte [esi+1]
19934                    <1>      ; BX = r2
19935 000055B1 66F705[56710000]00- <1>      test word [i.flgs], 4096 ; 1000h

```

```

19936 000055B9 10 <1>
19937 <1> ; is this a large or small file
19938 000055BA 756F <1> jnz short mget_5 ; 4f ; large file
19939 <1>
19940 000055BC F6C3F0 <1> test bl, 0F0h ; !0Fh
19941 <1> ; bit $!17,r2
19942 000055BF 7526 <1> jnz short mget_2
19943 <1> ; bne 3f / branch if r2 greater than or equal to 16
19944 000055C1 80E30E <1> and bl, 0Eh
19945 <1> ; bic $!16,r2 / clear all bits but bits 1,2,3
19946 000055C4 0FB783[5C710000] <1> movzx eax, word [ebx+i.dskp] ; AX = R1, physical block number
19947 <1> ; mov i.dskp(r2),r1 / r1 has physical block number
19948 000055CB 6609C0 <1> or ax, ax
19949 000055CE 7516 <1> jnz short mget_1
19950 <1> ; bne 2f / if physical block num is zero then need a new block
19951 <1> ; / for file
19952 000055D0 E8AB000000 <1> call alloc
19953 <1> ; jsr r0,alloc / allocate a new block
19954 <1> ; eAX (r1) = Physical block number
19955 000055D5 668983[5C710000] <1> mov [ebx+i.dskp], ax
19956 <1> ; mov r1,i.dskp(r2) / physical block number stored in i-node
19957 000055DC E84C020000 <1> call setimod
19958 <1> ; jsr r0,setimod / set inode modified byte (imod)
19959 000055E1 E8B1FEFFFF <1> call clear
19960 <1> ; jsr r0,clear / zero out disk/drum block just allocated
19961 <1> mget_1: ; 2:
19962 <1> ; eAX (r1) = Physical block number
19963 000055E6 C3 <1> retn
19964 <1> ; rts r0
19965 <1> mget_2: ; 3: / adding on block which changes small file to a large file
19966 000055E7 E894000000 <1> call alloc
19967 <1> ; jsr r0,alloc / allocate a new block for this file;
19968 <1> ; / block number in r1
19969 <1> ; eAX (r1) = Physical block number
19970 000055EC E8560C0000 <1> call wslot
19971 <1> ; jsr r0,wslot / set up I/O buffer for write, r5 points to
19972 <1> ; / first data word in buffer
19973 <1> ; eAX (r1) = Physical block number
19974 000055F1 B908000000 <1> mov ecx, 8 ; R3, transfer old physical block pointers
19975 <1> ; into new indirect block area for the new
19976 <1> ; large file
19977 000055F6 89DF <1> mov edi, ebx ; r5
19978 000055F8 BE[5C710000] <1> mov esi, i.dskp
19979 <1> ; mov $8.,r3 / next 6 instructions transfer old physical
19980 <1> ; / block pointers
19981 <1> ; mov $i.dskp,r2 / into new indirect block for the new
19982 <1> ; / large file
19983 000055FD 6631C0 <1> xor ax, ax ; mov ax, 0
19984 <1> mget_3: ;1:
19985 00005600 66A5 <1> movsw
19986 <1> ; mov (r2),(r5)+
19987 00005602 668946FE <1> mov [esi-2], ax
19988 <1> ; clr (r2)+
19989 00005606 E2F8 <1> loop mget_3 ; 1b
19990 <1> ; dec r3
19991 <1> ; bgt 1b
19992 <1>
19993 00005608 B1F8 <1> mov cl, 256-8
19994 <1> ; mov $256.-8.,r3 / clear rest of data buffer
19995 <1> mget_4: ; 1
19996 0000560A F366AB <1> rep stosw
19997 <1> ; clr (r5)+
19998 <1> ; dec r3
19999 <1> ; bgt 1b
20000 <1> ; 24/03/2013
20001 <1> ; AX (r1) = Physical block number
20002 0000560D E8510C0000 <1> call dskwr
20003 <1> ; jsr r0,dskwr / write new indirect block on disk
20004 <1> ; eAX (r1) = Physical block number
20005 00005612 66A3[5C710000] <1> mov [i.dskp], ax
20006 <1> ; mov r1,i.dskp / put pointer to indirect block in i-node
20007 00005618 66810D[56710000]00- <1> or word [i.flgs], 4096 ; 1000h
20008 00005620 10 <1>
20009 <1> ; bis $10000,i.flgs / set large file bit
20010 <1> ; / in i.flgs word of i-node
20011 00005621 E807020000 <1> call setimod
20012 <1> ; jsr r0,setimod / set i-node modified flag
20013 00005626 E97CFFFFFF <1> jmp mget_0
20014 <1> ; br mget
20015 <1>
20016 <1> mget_5: ; 4 ; large file
20017 <1> ; mov $-8,lsh / divide byte number by 256.
20018 <1> ; bic $!776,r2 / zero all bits but 1,2,3,4,5,6,7,8; gives offset
20019 <1> ; / in indirect block
20020 <1> ; mov r2,-(sp) / save on stack (*)
20021 <1> ; mov mq,r2 / calculate offset in i-node for pointer to proper
20022 <1> ; / indirect block
20023 <1> ; bic $!16,r2
20024 0000562B 80E3FE <1> and bl, 0FEh ; bh = 0
20025 0000562E 53 <1> push ebx ; i-node pointer offset in indirect block (*)
20026 <1> ; 01/03/2013 Max. possible BX (offset) value is 127 (65535/512)
20027 <1> ; for this file system (offset 128 to 255 not in use)

```

```

20028                                     <1> ; There is always 1 indirect block for this file system
20029 0000562F 0FB705[5C710000]         <1> movzx     eax, word [i.dskp] ; i.dskp[0]
20030                                     <1>          ; mov i.dskp(r2),r1
20031 00005636 6609C0                     <1> or      ax, ax ; R1
20032 00005639 7515                       <1> jnz     short mget_6 ; 2f
20033                                     <1>          ; bne 2f / if no indirect block exists
20034 0000563B E840000000                 <1> call    alloc
20035                                     <1>          ; jsr r0,alloc / allocate a new block
20036 00005640 66A3[5C710000]         <1> mov     [i.dskp], ax ; 03/03/2013
20037                                     <1>          ; mov r1,i.dskp(r2) / put block number of new block in i-node
20038 00005646 E8E2010000                 <1> call    setimod
20039                                     <1>          ; jsr r0,setimod / set i-node modified byte
20040                                     <1>          ; eAX = new block number
20041 0000564B E847FEFFFF                 <1> call    clear
20042                                     <1>          ; jsr r0,clear / clear new block
20043                                     <1> mget_6: ;2
20044                                     <1>          ; 05/03/2013
20045                                     <1>          ; eAX = r1, physical block number (of indirect block)
20046 00005650 E8920B0000                 <1> call    dskrd ; read indirect block
20047                                     <1>          ; jsr r0,dskrd / read in indirect block
20048 00005655 5A                         <1> pop     edx ; R2, get offset (*)
20049                                     <1>          ; mov (sp)+,r2 / get offset
20050                                     <1>          ; eAX = r1, physical block number (of indirect block)
20051 00005656 50                         <1> push    eax ; ** ; 24/03/2013
20052                                     <1>          ; mov r1,-(sp) / save block number of indirect block on stack
20053                                     <1>          ; eBX (r5) = pointer to buffer (indirect block)
20054 00005657 01D3                       <1> add     ebx, edx ; / r5 points to first word in indirect block, r2
20055                                     <1>          ; add r5,r2 / r5 points to first word in indirect block, r2
20056                                     <1>          ; / points to location of inter
20057 00005659 0FB703                     <1> movzx    eax, word [ebx] ; put physical block no of block
20058                                     <1>          ; in file sought in R1 (AX)
20059                                     <1>          ; mov (r2),r1 / put physical block no of block in file
20060                                     <1>          ; / sought in r1
20061 0000565C 6609C0                     <1> or      ax, ax
20062 0000565F 751D                       <1>          jnz     short mget_7 ; 2f
20063                                     <1>          ; bne 2f / if no block exists
20064 00005661 E81A000000                 <1> call    alloc
20065                                     <1>          ; jsr r0,alloc / allocate a new block
20066 00005666 668903                     <1> mov     [ebx], ax ; R1
20067                                     <1>          ; mov r1,(r2) / put new block number into proper location in
20068                                     <1>          ; / indirect block
20069 00005669 5A                         <1> pop     edx ; ** ; 24/03/2013
20070                                     <1>          ; mov (sp)+,r1 / get block number of indirect block
20071 0000566A 52                         <1> push    edx ; ** ; 31/07/2013
20072 0000566B 50                         <1> push    eax ; * ; 24/03/2013, 31/07/2013 (new block number)
20073 0000566C 89D0                       <1> mov     eax, edx ; 24/03/2013
20074                                     <1>          ; mov (r2),-(sp) / save block number of new block
20075                                     <1>          ; eAX (r1) = physical block number (of indirect block)
20076 0000566E E8D40B0000                 <1> call    wslot
20077                                     <1>          ; jsr r0,wslot
20078                                     <1>          ; eAX (r1) = physical block number
20079                                     <1>          ; eBX (r5) = pointer to buffer (indirect block)
20080 00005673 E8EB0B0000                 <1> call    dskwr
20081                                     <1>          ; eAX = r1 = physical block number (of indirect block)
20082                                     <1>          ; jsr r0,dskwr / write newly modified indirect block
20083                                     <1>          ; / back out on disk
20084 00005678 58                         <1> pop     eax ; * ; 31/07/2013
20085                                     <1>          ; mov (sp),r1 / restore block number of new block
20086                                     <1>          ; eAX (r1) = physical block number of new block
20087 00005679 E819FEFFFF                 <1> call    clear
20088                                     <1>          ; jsr r0,clear / clear new block
20089                                     <1> mget_7: ; 2
20090 0000567E 5A                         <1> pop     edx ; **
20091                                     <1>          ; tst (sp)+ / bump stack pointer
20092                                     <1>          ; eAX (r1) = Block number of new block
20093 0000567F C3                         <1> retn
20094                                     <1>          ; rts r0
20095                                     <1>
20096                                     <1> alloc:
20097                                     <1>          ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20098                                     <1>          ; 01/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
20099                                     <1>          ;
20100                                     <1>          ; get a free block and
20101                                     <1>          ; set the corresponding bit in the free storage map
20102                                     <1>          ;
20103                                     <1>          ; INPUTS ->
20104                                     <1>          ;   cdev (current device)
20105                                     <1>          ;   r2
20106                                     <1>          ;   r3
20107                                     <1>          ; OUTPUTS ->
20108                                     <1>          ;   r1 (physical block number of block assigned)
20109                                     <1>          ;   smod, mmod, system (super block), mount (mountable super block)
20110                                     <1>          ;
20111                                     <1>          ; ((AX = R1)) output
20112                                     <1>          ;   (Retro UNIX Prototype : 14/11/2012 - 21/07/2012, UNIXCOPY.ASM)
20113                                     <1>          ;   ((Modified registers: DX, CX))
20114                                     <1>
20115                                     <1>          ;mov r2,-(sp) / save r2, r3 on stack
20116                                     <1>          ;mov r3,-(sp)
20117                                     <1>          ;push     ecx
20118 00005680 53                         <1> push    ebx ; R2
20119                                     <1>          ;push     edx ; R3

```

20120	00005681	BB[287D0000]	<1>	mov	ebx, systm ; SuperBlock
20121			<1>		; mov \$systm,r2 / start of inode and free storage map for drum
20122	00005686	803D[66740000]00	<1>	cmp	byte [cdev], 0
20123			<1>		; tst cdev
20124	0000568D	7605	<1>	jna	short alloc_1
20125			<1>		; beq 1f / drum is device
20126	0000568F	BB[307F0000]	<1>	mov	ebx, mount
20127			<1>		; mov \$mount,r2 / disk or tape is device, start of inode and
20128			<1>		; / free storage map
20129			<1>	alloc_1:	; 1
20130	00005694	668B0B	<1>	mov	cx, [ebx]
20131			<1>		; mov (r2)+,r1 / first word contains number of bytes in free
20132			<1>		; / storage map
20133	00005697	66C1E103	<1>	shl	cx, 3
20134			<1>		; asl r1 / multiply r1 by eight gives
20135			<1>		; number of blocks in device
20136			<1>		; asl r1
20137			<1>		; asl r1
20138			<1>		;; push cx ;; 01/08/2013
20139			<1>		; mov r1,-(sp) / save # of blocks in device on stack
20140	0000569B	31C0	<1>	xor	eax, eax ; 0
20141			<1>		; clr r1 / r1 contains bit count of free storage map
20142			<1>	alloc_2:	; 1
20143	0000569D	43	<1>	inc	ebx ; 18/8/2012
20144	0000569E	43	<1>	inc	ebx ;
20145	0000569F	668B13	<1>	mov	dx, [ebx]
20146			<1>		; mov (r2)+,r3 / word of free storage map in r3
20147	000056A2	6609D2	<1>	or	dx, dx
20148	000056A5	750E	<1>	jnz	short alloc_3 ; 1f
20149			<1>		; bne 1f / branch if any free blocks in this word
20150	000056A7	6683C010	<1>	add	ax, 16
20151			<1>		; add \$16.,r1
20152	000056AB	6639C8	<1>	cmp	ax, cx
20153			<1>		; cmp r1 ,(sp) / have we examined all free storage bytes
20154	000056AE	72ED	<1>	jb	short alloc_2
20155			<1>		; blo 1b
20156			<1>		; 14/11/2015
20157			<1>		; Note: If the super block buffer has wrong content (zero bytes)
20158			<1>		; because of a (DMA or another) r/w error,
20159			<1>		; we will be here, at 'jmp panic' code address,
20160			<1>		; even if the (disk) file system space is not full !!!
20161			<1>		; (cx = 0)
20162			<1>		;
20163	000056B0	E95DE2FFFF	<1>	jmp	panic
20164			<1>		; jmp panic / found no free storage
20165			<1>	alloc_3:	; 1
20166	000056B5	66D1EA	<1>	shr	dx, 1
20167			<1>		; asr r3 / find a free block
20168	000056B8	7204	<1>	jc	short alloc_4 ; 1f
20169			<1>		; bcs 1f / branch when free block found; bit for block k
20170			<1>		; / is in byte k/8 / in bit k (mod 8)
20171	000056BA	6640	<1>	inc	ax
20172			<1>		; inc r1 / increment bit count in bit k (mod8)
20173	000056BC	EBF7	<1>	jmp	short alloc_3
20174			<1>		; br 1b
20175			<1>	alloc_4:	; 1:
20176			<1>		;; pop cx ;; 01/08/2013
20177			<1>		; tst (sp)+ / bump sp
20178			<1>		; 02/04/2013
20179	000056BE	E829000000	<1>	call	free3
20180			<1>		; jsr r0,3f / have found a free block
20181			<1>		; 21/8/2012
20182	000056C3	66F7D2	<1>	not	dx ; masking bit is '0' and others are '1'
20183	000056C6	662113	<1>	and	[ebx], dx ;; 0 -> allocated
20184			<1>		; bic r3,(r2) / set bit for this block
20185			<1>		; / i.e. assign block
20186			<1>		; br 2f
20187	000056C9	EB09	<1>	jmp	short alloc_5
20188			<1>		
20189			<1>	free:	
20190			<1>		; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20191			<1>		; 07/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
20192			<1>		;
20193			<1>		; calculates byte address and bit position for given block number
20194			<1>		; then sets the corresponding bit in the free storage map
20195			<1>		;
20196			<1>		; INPUTS ->
20197			<1>		; r1 - block number for a block structured device
20198			<1>		; cdev - current device
20199			<1>		; OUTPUTS ->
20200			<1>		; free storage map is updated
20201			<1>		; smod is incremented if cdev is root device (fixed disk)
20202			<1>		; mmod is incremented if cdev is a removable disk
20203			<1>		;
20204			<1>		; (Retro UNIX Prototype : 01/12/2012, UNIXCOPY.ASM)
20205			<1>		; ((Modified registers: DX, CX))
20206			<1>		
20207			<1>		;mov r2,-(sp) / save r2, r3
20208			<1>		;mov r3,-(sp)
20209			<1>		;push ecx
20210	000056CB	53	<1>	push	ebx ; R2
20211			<1>		;push edx ; R3

```

20212
20213 000056CC E81B000000 <1>
20214 <1> call free3
20215 <1> ; jsr r0,3f / set up bit mask and word no.
20216 <1> ; / in free storage map for block
20217 000056D1 660913 <1> or [ebx], dx
20218 <1> ; bis r3, (r2) / set free storage block bit;
20219 <1> ; / indicates free block
20220 <1> ; 0 -> allocated, 1 -> free
20221 <1>
20222 <1> alloc_5:
20223 <1> ; 07/04/2013
20224 <1> free_1: ; 2:
20225 <1> ; pop edx
20226 000056D4 5B <1> ; mov (sp)+,r3 / restore r2, r3
20227 <1> pop ebx
20228 <1> ; mov (sp)+,r2
20229 000056D5 803D[66740000]00 <1> ; pop ecx
20230 <1> cmp byte [cdev], 0
20231 <1> ; tst cdev / cdev = 0, block structured, drum;
20232 000056DC 7707 <1> ; / cdev = 1, mountable device
20233 <1> ja short alloc_6 ; 1f
20234 <1> ; bne 1f
20235 000056DE FE05[75740000] <1> ;mov byte [smod], 1
20236 <1> inc byte [smod]
20237 <1> ; incb smod / set super block modified for drum
20238 000056E4 C3 <1> ; eAX (r1) = block number
20239 <1> retn
20240 <1> ; rts r0
20241 <1> free_2:
20242 <1> alloc_6: ; 1:
20243 000056E5 FE05[76740000] <1> ;mov byte [mmod], 1
20244 <1> inc byte [mmod]
20245 <1> ; incb mmod
20246 <1> ; / set super block modified for mountable device
20247 000056EB C3 <1> ; eAX (r1) = block number
20248 <1> retn
20249 <1> ; rts r0
20250 <1> free3:
20251 <1> ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20252 <1> ; 02/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
20253 <1> ;
20254 <1> ; free3 is called from 'alloc' and 'free' procedures
20255 <1> ;
20256 000056EC 66BA0100 <1> alloc_free_3: ; 3
20257 000056F0 88C1 <1> mov dx, 1
20258 <1> mov cl, al
20259 000056F2 80E10F <1> ; mov r1,r2 / block number, k, = 1
20260 <1> and cl, 0Fh ; 0Fh <-- (k) mod 16
20261 000056F5 7403 <1> ; bic $!7,r2 / clear all bits but 0,1,2; r2 = (k) mod (8)
20262 <1> jz short free4
20263 <1> ; bisb 2f(r2),r3 / use mask to set bit in r3 corresponding to
20264 000056F7 66D3E2 <1> ; / (k) mod 8
20265 <1> shl dx, cl
20266 000056FA 0FB7D8 <1> free4:
20267 <1> movzx ebx, ax
20268 000056FD 66C1EB04 <1> ; mov r1,r2 / divide block number by 16
20269 <1> shr bx, 4
20270 <1> ; asr r2
20271 <1> ; asr r2
20272 <1> ; asr r2
20273 <1> ; asr r2
20274 <1> ; bcc 1f / branch if bit 3 in r1 was 0 i.e.,
20275 <1> ; / bit for block is in lower half of word
20276 <1> ; swab r3 / swap bytes in r3; bit in upper half of word in free
20277 <1> ; / storage map
20278 00005701 66D1E3 <1> alloc_free_4: ; 1
20279 <1> shl bx, 1
20280 00005704 81C3[2A7D0000] <1> ; asl r2 / multiply block number by 2; r2 = k/8
20281 <1> add ebx, system+2 ; SuperBlock+2
20282 <1> ; add $system+2,r2 / address of word of free storage map for drum
20283 0000570A 803D[66740000]00 <1> ; / with block bit in it
20284 <1> cmp byte [cdev], 0
20285 00005711 7606 <1> ; tst cdev
20286 <1> jna short alloc_free_5
20287 00005713 81C308020000 <1> ; beq 1f / cdev = 0 indicates device is drum
20288 <1> add ebx, mount - system
20289 <1> ; add $mount-system,r2 / address of word of free storage map for
20290 <1> ; / mountable device with bit of block to be
20291 <1> ; / freed
20292 00005719 C3 <1> alloc_free_5: ; 1
20293 <1> retn
20294 <1> ; rts r0 / return to 'free'
20295 <1> ; 2
20296 <1> ; .byte 1,2,4,10,20,40,100,200 / masks for bits 0,...,7
20297 <1>
20298 <1> iget:
20299 <1> ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20300 <1> ; 07/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
20301 <1> ;
20302 <1> ; get a new i-node whose i-number in r1 and whose device is in cdev
20303 <1> ;
20304 <1> ; ('iget' returns current i-number in r1, if input value of r1 is 0)

```

20304		<1>	;
20305		<1>	; INPUTS ->
20306		<1>	; ii - current i-number, rootdir
20307		<1>	; cdev - new i-node device
20308		<1>	; idev - current i-node device
20309		<1>	; imod - current i-node modified flag
20310		<1>	; mnti - cross device file i-number
20311		<1>	; r1 - i-number of new i-node
20312		<1>	; mntd - mountable device number
20313		<1>	;
20314		<1>	; OUTPUTS ->
20315		<1>	; cdev, idev, imod, ii, r1
20316		<1>	;
20317		<1>	; ((AX = R1)) input/output
20318		<1>	;
20319		<1>	; (Retro UNIX Prototype : 14/07/2012 - 18/11/2012, UNIXCOPY.ASM)
20320		<1>	; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
20321		<1>	
20322	0000571A 8A15[66740000]	<1>	mov dl, [cdev] ; 18/07/2013
20323	00005720 8A35[64740000]	<1>	mov dh, [idev] ; 07/08/2013
20324		<1>	;
20325	00005726 663B05[62740000]	<1>	cmp ax, [ii]
20326		<1>	; cmp r1,ii / r1 = i-number of current file
20327	0000572D 7504	<1>	jne short iget_1
20328		<1>	; bne 1f
20329	0000572F 38F2	<1>	cmp dl, dh
20330		<1>	; cmp idev,cdev
20331		<1>	; / is device number of i-node = current device
20332	00005731 7476	<1>	je short iget_5
20333		<1>	; beq 2f
20334		<1>	iget_1: ; 1:
20335	00005733 30DB	<1>	xor bl, bl
20336	00005735 381D[74740000]	<1>	cmp [imod], bl ; 0
20337		<1>	; tstb imod / has i-node of current file
20338		<1>	; / been modified i.e., imod set
20339	0000573B 762D	<1>	jna short iget_2
20340		<1>	; beq 1f
20341	0000573D 881D[74740000]	<1>	mov [imod], bl ; 0
20342		<1>	; clrb imod / if it has,
20343		<1>	; / we must write the new i-node out on disk
20344	00005743 6650	<1>	push ax
20345		<1>	; mov r1,-(sp)
20346		<1>	;mov dl, [cdev]
20347	00005745 6652	<1>	push dx
20348		<1>	; mov cdev,-(sp)
20349	00005747 66A1[62740000]	<1>	mov ax, [ii]
20350		<1>	; mov ii,r1
20351		<1>	;mov dh, [idev]
20352	0000574D 8835[66740000]	<1>	mov [cdev], dh
20353		<1>	; mov idev,cdev
20354	00005753 FEC3	<1>	inc bl ; 1
20355		<1>	; 31/07/2013
20356	00005755 881D[FC740000]	<1>	mov [rw], bl ; 1 == write
20357		<1>	;;28/07/2013 rw -> u.rw
20358		<1>	;;mov [u.rw], bl ; 1 == write
20359	0000575B E84A000000	<1>	call icalc
20360		<1>	; jsr r0,icalc; 1
20361	00005760 665A	<1>	pop dx
20362	00005762 8815[66740000]	<1>	mov [cdev], dl
20363		<1>	; mov (sp)+,cdev
20364	00005768 6658	<1>	pop ax
20365		<1>	; mov (sp)+,r1
20366		<1>	iget_2: ; 1:
20367	0000576A 6621C0	<1>	and ax, ax
20368		<1>	; tst r1 / is new i-number non zero
20369	0000576D 7434	<1>	jz short iget_4 ; 2f
20370		<1>	; beq 2f / branch if r1=0
20371		<1>	
20372		<1>	; mov dl, [cdev]
20373	0000576F 08D2	<1>	or dl, dl
20374		<1>	; tst cdev / is the current device number non zero
20375		<1>	; / (i.e., device != drum)
20376	00005771 7517	<1>	jnz short iget_3 ; 1f
20377		<1>	; bne 1f / branch 1f cdev != 0 ;; (cdev != 0)
20378	00005773 663B05[6C740000]	<1>	cmp ax, [mnti]
20379		<1>	; cmp r1,mnti / mnti is the i-number of the cross device
20380		<1>	; / file (root directory of mounted device)
20381	0000577A 750E	<1>	jne short iget_3 ; 1f
20382		<1>	; bne 1f
20383		<1>	;mov bl, [mntd]
20384	0000577C FEC2	<1>	inc dl ; mov dl, 1 ; 17/07/2013
20385	0000577E 8815[66740000]	<1>	mov [cdev], dl ; 17/07/2013 - 09/07/2013
20386		<1>	; mov mntd,cdev / make mounted device the current device
20387	00005784 66A1[70740000]	<1>	mov ax, [rootdir]
20388		<1>	; mov rootdir,r1
20389		<1>	iget_3: ; 1:
20390	0000578A 66A3[62740000]	<1>	mov [ii], ax
20391		<1>	; mov r1,ii
20392	00005790 8815[64740000]	<1>	mov [idev], dl ; cdev
20393		<1>	; mov cdev,idev
20394	00005796 30DB	<1>	xor bl, bl
20395		<1>	; 31/07/2013

20396	00005798	881D[FC740000]	<1>	mov	[rw], bl ; 0 == read
20397			<1>	;;28/07/2013 rw -> u.rw	
20398			<1>	;;mov [u.rw], bl ; 0 = read	
20399	0000579E	E807000000	<1>	call	icalc
20400			<1>	; jsr r0,icalc; 0 / read in i-node ii	
20401			<1>	iget_4: ; 2:	
20402	000057A3	66A1[62740000]	<1>	mov	ax, [ii]
20403			<1>	; mov ii,r1	
20404			<1>	iget_5:	
20405	000057A9	C3	<1>	retn	
20406			<1>	; rts r0	
20407			<1>		
20408			<1>	icalc:	
20409			<1>	; 02/07/2015	
20410			<1>	; 03/06/2015 (Retro UNIX 386 v1 - Beginning)	
20411			<1>	; 07/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)	
20412			<1>	;	
20413			<1>	; calculate physical block number from i-number then	
20414			<1>	; read or write that block	
20415			<1>	;	
20416			<1>	; 'icalc' is called from 'iget'	
20417			<1>	;	
20418			<1>	; for original unix v1:	
20419			<1>	; / i-node i is located in block (i+31.)/16. and begins 32.*	
20420			<1>	; / (i+31.) mod 16. bytes from its start	
20421			<1>	;	
20422			<1>	; for retro unix 8086 v1:	
20423			<1>	; i-node is located in block (i+47)/16 and	
20424			<1>	; begins 32*(i+47) mod 16 bytes from its start	
20425			<1>	;	
20426			<1>	; INPUTS ->	
20427			<1>	; r1 - i-number of i-node	
20428			<1>	;	
20429			<1>	; OUTPUTS ->	
20430			<1>	; inode r/w	
20431			<1>	;	
20432			<1>	; ((AX = R1)) input	
20433			<1>	;	
20434			<1>	; (Retro UNIX Prototype : 14/07/2012 - 18/11/2012, UNIXCOPY.ASM)	
20435			<1>	; ((Modified registers: eAX, eDX, eCX, eBX, eSI, eDI, eBP))	
20436			<1>	;	
20437	000057AA	0FB7D0	<1>	movzx	edx, ax
20438	000057AD	6683C22F	<1>	add	dx, 47
20439	000057B1	89D0	<1>	mov	eax, edx
20440			<1>	;add ax, 47 ; add 47 to inode number	
20441			<1>	; add \$31.,r1 / add 31. to i-number	
20442	000057B3	50	<1>	push	eax
20443			<1>	; mov r1,-(sp) / save i+31. on stack	
20444	000057B4	66C1E804	<1>	shr	ax, 4
20445			<1>	; asr r1 / divide by 16.	
20446			<1>	; asr r1	
20447			<1>	; asr r1	
20448			<1>	; asr r1 / r1 contains block number of block	
20449			<1>	; / in which i-node exists	
20450	000057B8	E82A0A0000	<1>	call	dskrd
20451			<1>	; jsr r0,dskrd / read in block containing i-node i.	
20452			<1>	; 31/07/2013	
20453	000057BD	803D[FC740000]00	<1>	cmp	byte [rw], 0 ; Retro Unix 8086 v1 feature !
20454			<1>	;; 28/07/2013 rw -> u.rw	
20455			<1>	;;cmp byte [u.rw], 0 ; Retro Unix 8086 v1 feature !	
20456			<1>	; tst (r0)	
20457	000057C4	7605	<1>	jna	short icalc_1
20458			<1>	; beq lf / branch to wslot when argument	
20459			<1>	; / in icalc call = 1	
20460			<1>	; eAX = r1 = block number	
20461	000057C6	E87C0A0000	<1>	call	wslot
20462			<1>	; jsr r0,wslot / set up data buffer for write	
20463			<1>	; / (will be same buffer as dskrd got)	
20464			<1>	; eBX = r5 points to first word in data area for this block	
20465			<1>	icalc_1: ; 1:	
20466	000057CB	5A	<1>	pop	edx
20467	000057CC	83E20F	<1>	and	edx, 0Fh ; (i+47) mod 16
20468			<1>	; bic \$!17,(sp) / zero all but last 4 bits;	
20469			<1>	; / gives (i+31.) mod 16	
20470	000057CF	C1E205	<1>	shl	edx, 5
20471			<1>	; eDX = 32 * ((i+47) mod 16)	
20472	000057D2	89DE	<1>	mov	esi, ebx ; ebx points 1st word of the buffer
20473	000057D4	01D6	<1>	add	esi, edx ; edx is inode offset in the buffer
20474			<1>	; eSI (r5) points to first word in i-node i.	
20475			<1>	; mov (sp)+,mq / calculate offset in data buffer;	
20476			<1>	; / 32.*(i+31.)mod16	
20477			<1>	; mov \$5,lsh / for i-node i.	
20478			<1>	; add mq,r5 / r5 points to first word in i-node i.	
20479	000057D6	BF[56710000]	<1>	mov	edi, inode
20480			<1>	; mov \$inode,r1 / inode is address of first word	
20481			<1>	; / of current i-node	
20482	000057DB	B908000000	<1>	mov	ecx, 8 ; 02/07/2015(32 bit modification)
20483			<1>	; mov \$16.,r3	
20484			<1>	; 31/07/2013	
20485	000057E0	382D[FC740000]	<1>	cmp	[rw], ch ; 0 ; Retro Unix 8086 v1 feature !
20486			<1>	;;28/07/2013 rw -> u.rw	
20487			<1>	;;cmp [u.rw], ch ; 0 ; Retro Unix 8086 v1 feature !	

20488		<1>	; tst (r0)+ / branch to 2f when argument in icalc call = 0
20489	000057E6 760A	<1>	jna short icalc_3
20490		<1>	; beq 2f / r0 now contains proper return address
20491		<1>	; / for rts r0
20492		<1>	icalc_2: ; 1:
20493	000057E8 87F7	<1>	xchg esi, edi
20494		<1>	; overwrite old i-node (in buffer to be written)
20495	000057EA F3A5	<1>	rep movsd
20496		<1>	; mov (r1)+, (r5)+ / over write old i-node
20497		<1>	; dec r3
20498		<1>	; bgt 1b
20499	000057EC E8720A0000	<1>	call dskwr
20500		<1>	; jsr r0, dskwr / write inode out on device
20501	000057F1 C3	<1>	retn
20502		<1>	; rts r0
20503		<1>	icalc_3: ; 2:
20504		<1>	; copy new i-node into inode area of (core) memory
20505	000057F2 F3A5	<1>	rep movsd
20506		<1>	; mov (r5)+, (r1)+ / read new i-node into
20507		<1>	; / "inode" area of core
20508		<1>	; dec r3
20509		<1>	; bgt 2b
20510	000057F4 C3	<1>	retn
20511		<1>	; rts r0
20512		<1>	
20513		<1>	access:
20514		<1>	; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20515		<1>	; 24/04/2013 - 29/04/2013 (Retro UNIX 8086 v1)
20516		<1>	;
20517		<1>	; check whether user is owner of file or user has read or write
20518		<1>	; permission (based on i.flgs).
20519		<1>	;
20520		<1>	; INPUTS ->
20521		<1>	; r1 - i-number of file
20522		<1>	; u.uid
20523		<1>	; arg0 -> (owner flag mask)
20524		<1>	; Retro UNIX 8086 v1 feature -> owner flag mask in DL (DX)
20525		<1>	; OUTPUTS ->
20526		<1>	; inode (or jump to error)
20527		<1>	;
20528		<1>	; ((AX = R1)) input/output
20529		<1>	;
20530		<1>	; ((Modified registers: eCX, eBX, eDX, eSI, eDI, eBP))
20531		<1>	;
20532	000057F5 6652	<1>	push dx ; save flags (DL)
20533	000057F7 E81EFFFFFF	<1>	call iget
20534		<1>	; jsr r0, iget / read in i-node for current directory
20535		<1>	; / (i-number passed in r1)
20536	000057FC 8A0D[56710000]	<1>	mov cl, [i.flgs]
20537		<1>	; mov i.flgs, r2
20538	00005802 665A	<1>	pop dx ; restore flags (DL)
20539	00005804 8A35[C6740000]	<1>	mov dh, [u.uid]
20540	0000580A 3A35[59710000]	<1>	cmp dh, [i.uid]
20541		<1>	; cmpb i.uid, u.uid / is user same as owner of file
20542	00005810 7503	<1>	jne short access_1
20543		<1>	; bne 1f / no, then branch
20544	00005812 C0E902	<1>	shr cl, 2
20545		<1>	; asrb r2 / shift owner read write bits into non owner
20546		<1>	; / read/write bits
20547		<1>	; asrb r2
20548		<1>	access_1: ; 1:
20549	00005815 20D1	<1>	and cl, dl
20550		<1>	; bit r2, (r0)+ / test read-write flags against argument
20551		<1>	; / in access call
20552	00005817 7513	<1>	jnz short access_2
20553		<1>	; bne 1f
20554	00005819 08F6	<1>	or dh, dh ; super user (root) ?
20555		<1>	; tstb u.uid
20556	0000581B 740F	<1>	jz short access_2 ; yes, super user
20557		<1>	; jnz error
20558		<1>	; beq 1f
20559		<1>	; jmp error
20560	0000581D C705[CF740000]0B00-	<1>	mov dword [u.error], ERR_FILE_ACCESS
20561	00005825 0000	<1>	
20562		<1>	; 'permission denied !' error
20563	00005827 E950E8FFFF	<1>	jmp error
20564		<1>	
20565		<1>	access_2: ; 1:
20566		<1>	; DL = flags
20567	0000582C C3	<1>	retn
20568		<1>	; rts r0
20569		<1>	
20570		<1>	setimod:
20571		<1>	; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20572		<1>	; 09/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
20573		<1>	;
20574		<1>	; 'setimod' sets byte at location 'imod' to 1; thus indicating that
20575		<1>	; the inode has been modified. Also puts the time of modification
20576		<1>	; into the inode.
20577		<1>	;
20578		<1>	; (Retro UNIX Prototype : 14/07/2012 - 23/02/2013, UNIXCOPY.ASM)
20579		<1>	; ((Modified registers: eDX, eCX, eBX))



20580		<1>	;
20581		<1>	
20582		<1>	; push edx
20583	0000582D 50	<1>	push eax
20584		<1>	
20585	0000582E C605[74740000]01	<1>	mov byte [imod], 1
20586		<1>	; movb \$1,imod / set current i-node modified bytes
20587		<1>	; Erdogan Tan 14-7-2012
20588	00005835 E87DE3FFFF	<1>	call epoch
20589		<1>	; mov s.time,i.mtim
20590		<1>	; / put present time into file modified time
20591		<1>	; mov s.time+2,i.mtim+2
20592		<1>	
20593	0000583A A3[70710000]	<1>	mov [i.mtim], eax
20594		<1>	
20595		<1>	; Retro UNIX 386 v1 modification ! (cmp)
20596		<1>	; Retro UNIX 8086 v1 modification ! (test)
20597	0000583F 833D[6C710000]00	<1>	cmp dword [i.ctim], 0
20598	00005846 7505	<1>	jnz short setimod_ok
20599		<1>	
20600	00005848 A3[6C710000]	<1>	mov [i.ctim], eax
20601		<1>	
20602		<1>	setimod_ok: ; 31/07/2013
20603	0000584D 58	<1>	pop eax
20604		<1>	;pop edx
20605		<1>	
20606	0000584E C3	<1>	retn
20607		<1>	; rts r0
20608		<1>	
20609		<1>	itrunc:
20610		<1>	; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20611		<1>	; 23/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
20612		<1>	;
20613		<1>	; 'itrunc' truncates a file whose i-number is given in r1
20614		<1>	; to zero length.
20615		<1>	;
20616		<1>	; INPUTS ->
20617		<1>	; r1 - i-number of i-node
20618		<1>	; i.dskp - pointer to contents or indirect block in an i-node
20619		<1>	; i.flgs - large file flag
20620		<1>	; i.size - size of file
20621		<1>	;
20622		<1>	; OUTPUTS ->
20623		<1>	; i.flgs - large file flag is cleared
20624		<1>	; i.size - set to 0
20625		<1>	; i.dskp .. i.dskp+16 - entire list is cleared
20626		<1>	; setimod - set to indicate i-node has been modified
20627		<1>	; r1 - i-number of i-node
20628		<1>	;
20629		<1>	; ((AX = R1)) input/output
20630		<1>	;
20631		<1>	; (Retro UNIX Prototype : 01/12/2012 - 10/03/2013, UNIXCOPY.ASM)
20632		<1>	; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
20633		<1>	
20634	0000584F E8C6FEFFFF	<1>	call iget
20635		<1>	; jsr r0,iget
20636	00005854 BE[5C710000]	<1>	mov esi, i.dskp
20637		<1>	; mov \$i.dskp,r2 / address of block pointers in r2
20638	00005859 31C0	<1>	xor eax, eax
20639		<1>	itrunc_1: ; 1:
20640	0000585B 66AD	<1>	lodsw
20641		<1>	; mov (r2)+,r1 / move physical block number into r1
20642	0000585D 6609C0	<1>	or ax, ax
20643	00005860 743B	<1>	jz short itrunc_5
20644		<1>	; beq 5f
20645	00005862 56	<1>	push esi
20646		<1>	; mov r2,-(sp)
20647	00005863 66F705[56710000]00-	<1>	test word [i.flgs], 1000h
20648	0000586B 10	<1>	
20649		<1>	; bit \$10000,i.flgs / test large file bit?
20650	0000586C 7429	<1>	jz short itrunc_4
20651		<1>	; beq 4f / if clear, branch
20652	0000586E 50	<1>	push eax
20653		<1>	; mov r1,-(sp) / save block number of indirect block
20654	0000586F E873090000	<1>	call dskrd
20655		<1>	; jsr r0,dskrd / read in block, 1st data word
20656		<1>	; / pointed to by r5
20657		<1>	; eBX = r5 = Buffer data address (the 1st word)
20658	00005874 B900010000	<1>	mov ecx, 256
20659		<1>	; mov \$256.,r3 / move word count into r3
20660	00005879 89DE	<1>	mov esi, ebx
20661		<1>	itrunc_2: ; 2:
20662	0000587B 66AD	<1>	lodsw
20663		<1>	; mov (r5)+,r1 / put 1st data word in r1;
20664		<1>	; / physical block number
20665	0000587D 6621C0	<1>	and ax, ax
20666	00005880 7409	<1>	jz short itrunc_3
20667		<1>	; beq 3f / branch if zero
20668		<1>	;push ecx
20669	00005882 6651	<1>	push cx
20670		<1>	; mov r3,-(sp) / save r3, r5 on stack</

```

20672                <1> ; mov r5,-(sp)
20673 00005884 E842FEFFFF <1> call free
20674                <1> ; jsr r0,free / free block in free storage map
20675                <1> ;pop esi
20676                <1> ; mov(sp)+,r5
20677 00005889 6659        <1> pop cx
20678                <1> ;pop ecx
20679                <1> ; mov (sp)+,r3
20680                <1> itrunc_3: ; 3:
20681 0000588B E2EE        <1> loop itrunc_2
20682                <1> ; dec r3 / decrement word count
20683                <1> ; bgt 2b / branch if positive
20684 0000588D 58          <1> pop eax
20685                <1> ; mov (sp)+,r1 / put physical block number of
20686                <1> ; / indirect block
20687                <1> ; 01/08/2013
20688 0000588E 668125[56710000]FF- <1> and word [i.flgs], 0EFFFh ; 111011111111111b
20689 00005896 EF          <1>
20690                <1> itrunc_4: ; 4:
20691 00005897 E82FFFEFFFF <1> call free
20692                <1> ; jsr r0,free / free indirect block
20693 0000589C 5E          <1> pop esi
20694                <1> ; mov (sp)+,r2
20695                <1> itrunc_5: ; 5:
20696 0000589D 81FE[6C710000] <1> cmp esi, i.dskp+16
20697                <1> ; cmp r2,$i.dskp+16.
20698 000058A3 72B6        <1> jb short itrunc_1
20699                <1> ; bne 1b / branch until all i.dskp entries check
20700                <1> ; 01/08/2013
20701                <1> ;and word [i.flgs], 0EFFFh ; 111011111111111b
20702                <1> ; bic $10000,i.flgs / clear large file bit
20703 000058A5 BF[5C710000] <1> mov edi, i.dskp
20704 000058AA 66B90800    <1> mov cx, 8
20705 000058AE 6631C0      <1> xor ax, ax
20706 000058B1 66A3[5A710000] <1> mov [i.size], ax ; 0
20707                <1> ; clr i.size / zero file size
20708 000058B7 F366AB      <1> rep stosw
20709                <1> ; jsr r0,copyz; i.dskp; i.dskp+16.
20710                <1> ; / zero block pointers
20711 000058BA E86EFFFFFFF <1> call setimod
20712                <1> ; jsr r0,setimod / set i-node modified flag
20713 000058BF 66A1[62740000] <1> mov ax, [ii]
20714                <1> ; mov ii,r1
20715 000058C5 C3          <1> retn
20716                <1> ; rts r0
20717                <1>
20718                <1> imap:
20719                <1> ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
20720                <1> ; 26/04/2013 (Retro UNIX 8086 v1)
20721                <1> ;
20722                <1> ; 'imap' finds the byte in core (superblock) containing
20723                <1> ; allocation bit for an i-node whose number in r1.
20724                <1> ;
20725                <1> ; INPUTS ->
20726                <1> ; r1 - contains an i-number
20727                <1> ; fsp - start of table containing open files
20728                <1> ;
20729                <1> ; OUTPUTS ->
20730                <1> ; r2 - byte address of byte with the allocation bit
20731                <1> ; mq - a mask to locate the bit position.
20732                <1> ; (a 1 is in calculated bit posisiton)
20733                <1> ;
20734                <1> ; ((AX = R1)) input/output
20735                <1> ; ((DL/DX = MQ)) output
20736                <1> ; ((BX = R2)) output
20737                <1> ;
20738                <1> ; (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
20739                <1> ; ((Modified registers: eDX, eCX, eBX, eSI))
20740                <1> ;
20741                <1> ; / get the byte that has the allocation bit for
20742                <1> ; / the i-number contained in r1
20743                <1> ;mov dx, 1
20744 000058C6 B201        <1> mov dl, 1
20745                <1> ; mov $1,mq / put 1 in the mq
20746 000058C8 0FB7D8      <1> movzx ebx, ax
20747                <1> ; mov r1,r2 / r2 now has i-number whose byte
20748                <1> ; / in the map we must find
20749 000058CB 6683EB29     <1> sub bx, 41
20750                <1> ; sub $41.,r2 / r2 has i-41
20751 000058CF 88D9        <1> mov cl, bl
20752                <1> ; mov r2,r3 / r3 has i-41
20753 000058D1 80E107       <1> and cl, 7
20754                <1> ; bic $!7,r3 / r3 has (i-41) mod 8 to get
20755                <1> ; / the bit position
20756 000058D4 7402        <1> jz short imap1
20757                <1> ;shl dx, cl
20758 000058D6 D2E2        <1> shl dl, cl
20759                <1> ; mov r3,lsh / move the 1 over (i-41) mod 8 positions
20760                <1> imap1: ; / to the left to mask the correct bit
20761 000058D8 66C1EB03     <1> shr bx, 3
20762                <1> ; asr r2
20763                <1> ; asr r2

```

```

20764      <1>      ; asr r2 / r2 has (i-41) base 8 of the byte number
20765      <1>      ; / from the start of the map
20766      <1>      ; mov r2,-(sp) / put (i-41) base 8 on the stack
20767 000058DC BE[287D0000] <1> mov esi, systm
20768      <1>      ; mov $systm,r2 / r2 points to the in-core image of
20769      <1>      ; / the super block for drum
20770      <1>      ;cmp word [cdev], 0
20771 000058E1 803D[66740000]00 <1> cmp byte [cdev], 0
20772      <1>      ; tst cdev / is the device the disk
20773 000058E8 7606 <1> jna short imap2
20774      <1>      ; beq lf / yes
20775 000058EA 81C608020000 <1> add esi, mount - systm
20776      <1>      ; add $mount-systm,r2 / for mounted device,
20777      <1>      ; / r2 points to 1st word of its super block
20778      <1> imap2: ; 1:
20779 000058F0 66031E <1> add bx, [esi] ;; add free map size to si
20780      <1>      ; add (r2)+,(sp) / get byte address of allocation bit
20781 000058F3 6683C304 <1> add bx, 4
20782 000058F7 01F3 <1> add ebx, esi
20783      <1>      ; add (sp)+,r2 / ?
20784      <1> ;add ebx, 4 ;; inode map offset in superblock
20785      <1>      ;; (2 + free map size + 2)
20786      <1>      ; add $2,r2 / ?
20787      <1> ; DL/DX (MQ) has a 1 in the calculated bit position
20788      <1>      ; BX (R2) has byte address of the byte with allocation bit
20789 000058F9 C3 <1> retn
20790      <1>      ; rts r0
20791      %include 'u6.s' ; 31/05/2015
20792 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS6.INC
20793 <1> ; Last Modification: 18/11/2015
20794 <1> ; -----
20795 <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
20796 <1> ; (v0.1 - Beginning: 11/07/2012)
20797 <1> ;
20798 <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
20799 <1> ; (Original) Source Code by Ken Thompson (1971-1972)
20800 <1> ; <Bell Laboratories (17/3/1972)>
20801 <1> ; <Preliminary Release of UNIX Implementation Document>
20802 <1> ;
20803 <1> ; Retro UNIX 8086 v1 - U6.ASM (23/07/2014) //// UNIX v1 -> u6.s
20804 <1> ;
20805 <1> ; *****
20806 <1>
20807 <1> readi:
20808 <1> ; 20/05/2015
20809 <1> ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
20810 <1> ; 11/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
20811 <1> ;
20812 <1> ; Reads from an inode whose number in R1
20813 <1> ;
20814 <1> ; INPUTS ->
20815 <1> ; r1 - inode number
20816 <1> ; u.count - byte count user desires
20817 <1> ; u.base - points to user buffer
20818 <1> ; u.fofp - points to word with current file offset
20819 <1> ; OUTPUTS ->
20820 <1> ; u.count - cleared
20821 <1> ; u.nread - accumulates total bytes passed back
20822 <1> ;
20823 <1> ; ((AX = R1)) input/output
20824 <1> ; (Retro UNIX Prototype : 01/03/2013 - 14/12/2012, UNIXCOPY.ASM)
20825 <1> ; ((Modified registers: edx, ebx, ecx, esi, esi, ebp))
20826 <1>
20827 000058FA 31D2 <1> xor edx, edx ; 0
20828 000058FC 8915[A8740000] <1> mov [u.nread], edx ; 0
20829 <1> ; clr u.nread / accumulates number of bytes transmitted
20830 00005902 668915[DF740000] <1> mov [u.pcount], dx ; 19/05/2015
20831 00005909 3915[A4740000] <1> cmp [u.count], edx ; 0
20832 <1> ; tst u.count / is number of bytes to be read greater than 0
20833 0000590F 7701 <1> ja short readi_1 ; lf
20834 <1> ; bgt lf / yes, branch
20835 00005911 C3 <1> retn
20836 <1> ; rts r0 / no, nothing to read; return to caller
20837 <1> readi_1: ; 1:
20838 <1> ; mov r1,-(sp) / save i-number on stack
20839 00005912 6683F828 <1> cmp ax, 40
20840 <1> ; cmp r1,$40. / want to read a special file
20841 <1> ; / (i-nodes 1,...,40 are for special files)
20842 00005916 0F87D3000000 <1> ja dskr
20843 <1> ; ble lf / yes, branch
20844 <1> ; jmp dskr / no, jmp to dskr;
20845 <1> ; / read file with i-node number (r1)
20846 <1> ; / starting at byte ((u.fofp)), read in u.count bytes
20847 <1> ; (20/05/2015)
20848 0000591C 50 <1> push eax ; because subroutines will jump to 'ret_'
20849 <1> ; 1:
20850 0000591D 0FB6D8 <1> movzx ebx, al
20851 00005920 66C1E302 <1> shl bx, 2
20852 <1> ; asl r1 / multiply inode number by 2
20853 00005924 81C3[28590000] <1> add ebx, readi_2 - 4
20854 0000592A FF23 <1> jmp dword [ebx]
20855 <1> ; jmp *1f-2(r1)

```

```

20856      <1> readi_2: ; 1:
20857 0000592C [78590000] <1> dd rttty ; tty, AX = 1 (runix)
20858      <1> ;rtty / tty; r1=2
20859      <1> ;rppt / ppt; r1=4
20860 00005930 [CB590000] <1> dd rmem ; mem, AX = 2 (runix)
20861      <1> ;rmem / mem; r1=6
20862      <1> ;rrf0 / rf0
20863      <1> ;rrk0 / rk0
20864      <1> ;rtap / tap0
20865      <1> ;rtap / tap1
20866      <1> ;rtap / tap2
20867      <1> ;rtap / tap3
20868      <1> ;rtap / tap4
20869      <1> ;rtap / tap5
20870      <1> ;rtap / tap6
20871      <1> ;rtap / tap7
20872 00005934 [8D600000] <1> dd rfd ; fd0, AX = 3 (runix only)
20873 00005938 [8D600000] <1> dd rfd ; fd1, AX = 4 (runix only)
20874 0000593C [8D600000] <1> dd rhd ; hd0, AX = 5 (runix only)
20875 00005940 [8D600000] <1> dd rhd ; hd1, AX = 6 (runix only)
20876 00005944 [8D600000] <1> dd rhd ; hd2, AX = 7 (runix only)
20877 00005948 [8D600000] <1> dd rhd ; hd3, AX = 8 (runix only)
20878 0000594C [E0590000] <1> dd rlpr ; lpr, AX = 9 (invalid, write only device !?)
20879 00005950 [C7590000] <1> dd rcvt ; tty0, AX = 10 (runix)
20880      <1> ;rcvt / tty0
20881 00005954 [C7590000] <1> dd rcvt ; tty1, AX = 11 (runix)
20882      <1> ;rcvt / tty1
20883 00005958 [C7590000] <1> dd rcvt ; tty2, AX = 12 (runix)
20884      <1> ;rcvt / tty2
20885 0000595C [C7590000] <1> dd rcvt ; tty3, AX = 13 (runix)
20886      <1> ;rcvt / tty3
20887 00005960 [C7590000] <1> dd rcvt ; tty4, AX = 14 (runix)
20888      <1> ;rcvt / tty4
20889 00005964 [C7590000] <1> dd rcvt ; tty5, AX = 15 (runix)
20890      <1> ;rcvt / tty5
20891 00005968 [C7590000] <1> dd rcvt ; tty6, AX = 16 (runix)
20892      <1> ;rcvt / tty6
20893 0000596C [C7590000] <1> dd rcvt ; tty7, AX = 17 (runix)
20894      <1> ;rcvt / tty7
20895 00005970 [C7590000] <1> dd rcvt ; COM1, AX = 18 (runix only)
20896      <1> ;rcrd / crd
20897 00005974 [C7590000] <1> dd rcvt ; COM2, AX = 19 (runix only)
20898      <1>
20899      <1> rttty: ; / read from console tty
20900      <1> ; 17/10/2015 - 16/07/2015 (Retro UNIX 8086 v1)
20901      <1> ; (Only 1 byte is read, by ignoring byte count!)
20902      <1> ; WHAT FOR: Every character from Keyboard input
20903      <1> ; must be written immediate on video page (screen)
20904      <1> ; when it is required.
20905      <1> ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
20906      <1> ; 11/03/2013 - 19/06/2014 (Retro UNIX 8086 v1)
20907      <1> ;
20908      <1> ; Console tty buffer is PC keyboard buffer
20909      <1> ; and keyboard-keystroke handling is different than original
20910      <1> ; unix (PDP-11) here. TTY/Keyboard procedures here are changed
20911      <1> ; according to IBM PC compatible ROM BIOS keyboard functions.
20912      <1> ;
20913      <1> ; 06/12/2013
20914 00005978 0FB61D[C9740000] <1> movzx ebx, byte [u.uno] ; process number
20915 0000597F 8A83[D5710000] <1> mov al, [ebx+p.ttyc-1] ; current/console tty
20916      <1> rttys:
20917      <1> ; mov tty+[8*ntty]-8+6,r5 / r5 is the address of the 4th word of
20918      <1> ; / of the control and status block
20919      <1> ; tst 2(r5) / for the console tty; this word points to the console
20920      <1> ; / tty buffer
20921      <1> ; 28/07/2013
20922 00005985 A2[CE740000] <1> mov [u.tty], al
20923      <1> ; 13/01/2014
20924 0000598A FEC0 <1> inc al
20925 0000598C A2[B0740000] <1> mov [u.ttyp], al ; tty number + 1
20926      <1> rttty_nc: ; 01/02/2014
20927      <1> ; 29/09/2013
20928 00005991 B90A000000 <1> mov ecx, 10
20929      <1> rttty_1: ; 01/02/2014
20930 00005996 6651 <1> push cx ; 29/09/2013
20931      <1> ; byte [u.tty] = tty number (0 to 9)
20932 00005998 B001 <1> mov al, 1
20933 0000599A E8330B0000 <1> call getc
20934 0000599F 6659 <1> pop cx ; 29/09/2013
20935 000059A1 7516 <1> jnz short rttty_2
20936      <1> ; bne 1f / 2nd word of console tty buffer contains number
20937      <1> ; / of chars. Is this number non-zero?
20938 000059A3 E20D <1> loop rttty_idle ; 01/02/2014
20939      <1> ; 05/10/2013
20940 000059A5 8A25[CE740000] <1> mov ah, [u.tty]
20941      <1> ; 29/09/2013
20942 000059AB E871FBFFFF <1> call sleep
20943      <1> ; jsr r0,canon; ttych / if 0, call 'canon' to get a line
20944      <1> ; / (120 chars.)
20945      <1> ;byte [u.tty] = tty number (0 to 9)
20946 000059B0 EBDF <1> jmp short rttty_nc ; 01/02/2014
20947      <1>

```

20948		<1> rtty_idle:
20949		<1> ; 29/07/2013
20950	000059B2 E8D6FAFFFF	<1> call idle
20951	000059B7 EBDD	<1> jmp short rtty_1 ; 01/02/2014
20952		<1> ;1:
20953		<1> ; tst 2(r5) / is the number of characters zero
20954		<1> ; beq ret1 / yes, return to caller via 'ret1'
20955		<1> ; movb *4(r5),r1 / no, put character in r1
20956		<1> ; inc 4(r5) / 3rd word of console tty buffer points to byte which
20957		<1> ; / contains the next char.
20958		<1> ; dec 2(r5) / decrement the character count
20959		<1> rtty_2:
20960	000059B9 30C0	<1> xor al, al
20961	000059BB E8120B0000	<1> call getc
20962	000059C0 E892000000	<1> call passc
20963		<1> ; jsr r0,passc / move the character to core (user)
20964		<1> ;; 17/10/2015 - 16/07/2015
20965		<1> ; 19/06/2014
20966		<1> ;;jnz short rtty_nc
20967	000059C5 58	<1> pop eax ; (20/05/2015)
20968	000059C6 C3	<1> retn
20969		<1> ;ret1:
20970		<1> ; jmp ret / return to caller via 'ret'
20971		<1>
20972		<1> rcvt: ; < receive/read character from tty >
20973		<1> ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
20974		<1> ; 15/05/2013 - 06/12/2013 (Retro UNIX 8086 v1)
20975		<1> ;
20976		<1> ; Retro UNIX 8086 v1 modification !
20977		<1> ;
20978		<1> ; In original UNIX v1, 'rcvt' routine
20979		<1> ; (exactly different than this one)
20980		<1> ; was in 'u9.s' file.
20981		<1> ;
20982	000059C7 2C0A	<1> sub al, 10
20983		<1> ; AL = tty number (0 to 9), (COM1=8, COM2=9)
20984		<1> ; 16/07/2013
20985		<1> ; 21/05/2013
20986	000059C9 EBBA	<1> jmp short rttys
20987		<1>
20988		<1> ;rppt: / read paper tape
20989		<1> ;jsr r0,pptic / gets next character in clist for ppt input and
20990		<1> ; / places
20991		<1> ; br ret / it in r1; if there is no problem with reader, it
20992		<1> ; / also enables read bit in prs
20993		<1> ;jsr r0,passc / place character in users buffer area
20994		<1> ;br rppt
20995		<1>
20996		<1> rmem: ; / transfer characters from memory to a user area of core
20997		<1> ; 17/10/2015
20998		<1> ; 11/06/2015
20999		<1> ; 24/05/2015
21000		<1> ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21001		<1> ;
21002	000059CB 8B35[90740000]	<1> mov esi, [u.fofp]
21003		<1> rmem_1:
21004	000059D1 8B1E	<1> mov ebx, [esi]
21005		<1> ; mov *u.fofp,r1 / save file offset which points to the char
21006		<1> ; / to be transferred to user
21007	000059D3 FF06	<1> inc dword [esi] ; 17/10/2015
21008		<1> ; inc *u.fofp / increment file offset to point to 'next'
21009		<1> ; / char in memory file
21010	000059D5 8A03	<1> mov al, [ebx]
21011		<1> ; movb (r1),r1 / get character from memory file,
21012		<1> ; / put it in r1
21013	000059D7 E87B000000	<1> call passc ; jsr r0,passc / move this character to
21014		<1> ; / the next byte of the users core area
21015		<1> ; br rmem / continue
21016	000059DC 75F3	<1> jnz short rmem_1
21017		<1> ret_:
21018	000059DE 58	<1> pop eax ; 09/06/2015
21019	000059DF C3	<1> retn
21020		<1>
21021		<1> rlpr:
21022		<1> ;1:
21023		<1> ;rcrd:
21024	000059E0 C705[CF740000]0F00-	<1> mov dword [u.error], ERR_DEV_NOT_RDY ; 19/05/2015
21025	000059E8 0000	<1>
21026	000059EA E98DE6FFFF	<1> jmp error
21027		<1> ;jmp error / see 'error' routine
21028		<1>
21029		<1> dskr:
21030		<1> ; 12/10/2015
21031		<1> ; 21/08/2015
21032		<1> ; 25/07/2015
21033		<1> ; 10/07/2015
21034		<1> ; 16/06/2015
21035		<1> ; 31/05/2015
21036		<1> ; 24/05/2015 (Retro UNIX 386 v1 - Beginning)
21037		<1> ; 26/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
21038		<1> dskr_0:
21039	000059EF 50	<1> push eax

21040		<1>	; mov (sp),r1 / i-number in r1
21041		<1>	; AX = i-number
21042	000059F0 E825FDFFFF	<1>	call iget
21043		<1>	; jsr r0,iget / get i-node (r1) into i-node section of core
21044	000059F5 0FB715[5A710000]	<1>	movzx edx, word [i.size] ; 16/06/2015
21045		<1>	; mov i.size,r2 / file size in bytes in r2
21046	000059FC 8B1D[90740000]	<1>	mov ebx, [u.fofp]
21047	00005A02 2B13	<1>	sub edx, [ebx]
21048		<1>	; sub *u.fofp,r2 / subtract file offset
21049		<1>	; 12/10/2015
21050		<1>	; jna short ret_
21051		<1>	; blos ret
21052	00005A04 7709	<1>	ja short dskr_1
21053		<1>	;
21054		<1>	dskr_retn: ; 12/10/2015
21055	00005A06 58	<1>	pop eax
21056	00005A07 C605[E1740000]00	<1>	mov byte [u.kcall], 0
21057	00005A0E C3	<1>	retn
21058		<1>	dskr_1:
21059	00005A0F 3B15[A4740000]	<1>	cmp edx, [u.count]
21060		<1>	; cmp r2,u.count / are enough bytes left in file
21061		<1>	; / to carry out read
21062	00005A15 7306	<1>	jnb short dskr_2
21063		<1>	; bhis 1f
21064	00005A17 8915[A4740000]	<1>	mov [u.count], edx
21065		<1>	; mov r2,u.count / no, just read to end of file
21066		<1>	dskr_2: ; 1:
21067		<1>	; AX = i-number
21068	00005A1D E885FBFFFF	<1>	call mget
21069		<1>	; jsr r0,mget / returns physical block number of block
21070		<1>	; / in file where offset points
21071		<1>	; eAX = physical block number
21072	00005A22 E8C0070000	<1>	call dskrd
21073		<1>	; jsr r0,dskrd / read in block, r5 points to
21074		<1>	; / 1st word of data in buffer
21075		<1>	; 09/06/2015
21076	00005A27 803D[E1740000]00	<1>	cmp byte [u.kcall], 0 ; the caller is 'namei' sign (=1)
21077	00005A2E 770F	<1>	ja short dskr_4 ; zf=0 -> the caller is 'namei'
21078	00005A30 66833D[DF740000]00	<1>	cmp word [u.pcount], 0
21079	00005A38 7705	<1>	ja short dskr_4
21080		<1>	dskr_3:
21081		<1>	; [u.base] = virtual address to transfer (as destination address)
21082	00005A3A E853000000	<1>	call trans_addr_w ; translate virtual address to physical (w)
21083		<1>	dskr_4:
21084		<1>	; eBX (r5) = system (I/O) buffer address -physical-
21085	00005A3F E8C7020000	<1>	call sioreg
21086		<1>	; jsr r0,sioreg
21087	00005A44 87F7	<1>	xchg esi, edi
21088		<1>	; eDI = file (user data) offset
21089		<1>	; eSI = sector (I/O) buffer offset
21090		<1>	; eCX = byte count
21091	00005A46 F3A4	<1>	rep movsb
21092		<1>	; movb (r2)+,(r1)+ / move data from buffer into working core
21093		<1>	; / starting at u.base
21094		<1>	; dec r3
21095		<1>	; bne 2b / branch until proper number of bytes are transferred
21096		<1>	; 25/07/2015
21097		<1>	; eax = remain bytes in buffer
21098		<1>	; (check if remain bytes in the buffer > [u.pcount])
21099	00005A48 09C0	<1>	or eax, eax
21100	00005A4A 75EE	<1>	jnz short dskr_3 ; (page end before system buffer end!)
21101		<1>	; 03/08/2013
21102		<1>	;pop eax
21103	00005A4C 390D[A4740000]	<1>	cmp [u.count], ecx ; 0
21104		<1>	; tst u.count / all bytes read off disk
21105		<1>	; bne dskr
21106		<1>	; br ret
21107		<1>	;ja short dskr_0
21108		<1>	;mov [u.kcall], cl ; 0 ; 09/06/2015
21109		<1>	;retn
21110		<1>	; 12/10/2015
21111	00005A52 76B2	<1>	jna short dskr_retn
21112	00005A54 58	<1>	pop eax ; (i-node number)
21113	00005A55 EB98	<1>	jmp short dskr_0
21114		<1>	
21115		<1>	passc:
21116		<1>	; 18/10/2015
21117		<1>	; 10/07/2015
21118		<1>	; 01/07/2015
21119		<1>	; 08/06/2015
21120		<1>	; 04/06/2015
21121		<1>	; 20/05/2015
21122		<1>	; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21123		<1>	;
21124		<1>	;(Retro UNIX 386 v1 - translation from user's virtual address
21125		<1>	to physical address
21126	00005A57 66833D[DF740000]00	<1>	cmp word [u.pcount], 0 ; byte count in page = 0 (initial value)
21127		<1>	; 1-4095 --> use previous physical base address
21128		<1>	; in [u.pbase]
21129	00005A5F 7705	<1>	ja short passc_3
21130		<1>	; 08/06/2015 - 10/07/2015
21131	00005A61 E82C000000	<1>	call trans_addr_w

21132		<1> passc_3:
21133		<1> ; 19/05/2015
21134	00005A66 66FF0D[DF740000]	<1> dec word [u.pcount]
21135		<1> ;
21136	00005A6D 8B1D[DB740000]	<1> mov ebx, [u.pbase]
21137	00005A73 8803	<1> mov [ebx], al
21138		<1> ; movb r1,*u.base / move a character to the next byte of the
21139		<1> ; / users buffer
21140	00005A75 FF05[A0740000]	<1> inc dword [u.base]
21141		<1> ; inc u.base / increment the pointer to point to
21142		<1> ; / the next byte in users buffer
21143	00005A7B FF05[DB740000]	<1> inc dword [u.pbase] ; 04/06/2015
21144	00005A81 FF05[A8740000]	<1> inc dword [u.nread]
21145		<1> ; inc u.nread / increment the number of bytes read
21146	00005A87 FF0D[A4740000]	<1> dec dword [u.count]
21147		<1> ; dec u.count / decrement the number of bytes to be read
21148		<1> ; bne lf / any more bytes to read?; yes, branch
21149	00005A8D C3	<1> retn
21150		<1> ; mov (sp)+,r0 / no, do a non-local return to the caller of
21151		<1> ; / 'readi' by:
21152		<1> ;/ (1) pop the return address off the stack into r0
21153		<1> ; mov (sp)+,r1 / (2) pop the i-number off the stack into r1
21154		<1> ;1:
21155		<1> ; clr *\$ps / clear processor status
21156		<1> ; rts r0 / return to address currently on top of stack
21157		<1>
21158		<1> trans_addr_r:
21159		<1> ; Translate virtual address to physical address
21160		<1> ; for reading from user's memory space
21161		<1> ; (Retro UNIX 386 v1 feature only !)
21162		<1> ; 18/10/2015
21163		<1> ; 10/07/2015
21164		<1> ; 09/06/2015
21165		<1> ; 08/06/2015
21166		<1> ; 04/06/2015
21167		<1> ;
21168		<1> ; 18/10/2015
21169	00005A8E 31D2	<1> xor edx, edx ; 0 (read access sign)
21170	00005A90 EB04	<1> jmp short trans_addr_rw
21171		<1>
21172		<1> ;push eax
21173		<1> ;push ebx
21174		<1> ;mov ebx, [u.base]
21175		<1> ;call get_physical_addr ; get physical address
21176		<1> ;;jnc short cpass_0
21177		<1> ;jnc short passc_1
21178		<1> ;mov [u.error], eax
21179		<1> ;;pop ebx
21180		<1> ;;pop eax
21181		<1> ;jmp error
21182		<1> ;cpass_0:
21183		<1> ; 18/10/2015
21184		<1> ; 20/05/2015
21185		<1> ;mov [u.pbase], eax ; physical address
21186		<1> ;mov [u.pcount], cx ; remain byte count in page (1-4096)
21187		<1> ;pop ebx
21188		<1> ;pop eax
21189		<1> ;retn ; 08/06/2015
21190		<1>
21191		<1> trans_addr_w:
21192		<1> ; Translate virtual address to physical address
21193		<1> ; for writing to user's memory space
21194		<1> ; (Retro UNIX 386 v1 feature only !)
21195		<1> ; 18/10/2015
21196		<1> ; 29/07/2015
21197		<1> ; 10/07/2015
21198		<1> ; 09/06/2015
21199		<1> ; 08/06/2015
21200		<1> ; 04/06/2015 (passc)
21201		<1> ;
21202		<1> ; 18/10/2015
21203	00005A92 29D2	<1> sub edx, edx
21204	00005A94 FEC2	<1> inc dl ; 1 (write access sign)
21205		<1> trans_addr_rw:
21206	00005A96 50	<1> push eax
21207	00005A97 53	<1> push ebx
21208		<1> ; 18/10/2015
21209	00005A98 52	<1> push edx ; r/w sign (in DL)
21210		<1> ;
21211	00005A99 8B1D[A0740000]	<1> mov ebx, [u.base]
21212	00005A9F E826DCFFFF	<1> call get_physical_addr ; get physical address
21213	00005AA4 730A	<1> jnc short passc_0
21214	00005AA6 A3[CF740000]	<1> mov [u.error], eax
21215		<1> ;pop edx
21216		<1> ;pop ebx
21217		<1> ;pop eax
21218	00005AAB E9CCE5FFFF	<1> jmp error
21219		<1> passc_0:
21220	00005AB0 F6C202	<1> test dl, PTE_A_WRITE ; writable page ; 18/10/2015
21221	00005AB3 5A	<1> pop edx ; 18/10/2015
21222	00005AB4 751C	<1> jnz short passc_1
21223		<1> ; 18/10/2015

21224	00005AB6	20D2	<1>	and dl, dl
21225	00005AB8	7418	<1>	jz short passc_1
21226			<1>	; 20/05/2015
21227			<1>	; read only (duplicated) page -must be copied to a new page-
21228			<1>	; EBX = linear address
21229	00005ABA	51	<1>	push ecx
21230	00005ABB	E81FD9FFFF	<1>	call copy_page
21231	00005AC0	59	<1>	pop ecx
21232	00005AC1	721E	<1>	jc short passc_2
21233	00005AC3	50	<1>	push eax ; physical address of the new/allocated page
21234	00005AC4	E82BDBFFFF	<1>	call add_to_swap_queue
21235	00005AC9	58	<1>	pop eax
21236			<1>	; 18/10/2015
21237	00005ACA	81E3FF0F0000	<1>	and ebx, PAGE_OFF ; 0FFFh
21238			<1>	;mov ecx, PAGE_SIZE
21239			<1>	;sub ecx, ebx
21240	00005AD0	01D8	<1>	add eax, ebx
21241			<1>	passc_1:
21242			<1>	; 18/10/2015
21243			<1>	; 20/05/2015
21244	00005AD2	A3[DB740000]	<1>	mov [u.pbase], eax ; physical address
21245	00005AD7	66890D[DF740000]	<1>	mov [u.pcount], cx ; remain byte count in page (1-4096)
21246	00005ADE	5B	<1>	pop ebx
21247	00005ADF	58	<1>	pop eax
21248	00005AE0	C3	<1>	retn ; 08/06/2015
21249			<1>	passc_2:
21250	00005AE1	C705[CF740000]0100-	<1>	mov dword [u.error], ERR_MINOR_IM ; "Insufficient memory !" error
21251	00005AE9	0000	<1>	
21252			<1>	;pop ebx
21253			<1>	;pop eax
21254	00005AEB	E98CE5FFFF	<1>	jmp error
21255			<1>	
21256			<1>	writei:
21257			<1>	; 20/05/2015
21258			<1>	; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21259			<1>	; 12/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
21260			<1>	;
21261			<1>	; Write data to file with inode number in R1
21262			<1>	;
21263			<1>	; INPUTS ->
21264			<1>	; r1 - inode number
21265			<1>	; u.count - byte count to be written
21266			<1>	; u.base - points to user buffer
21267			<1>	; u.fofp - points to word with current file offset
21268			<1>	; OUTPUTS ->
21269			<1>	; u.count - cleared
21270			<1>	; u.nread - accumulates total bytes passed back
21271			<1>	; ((AX = R1))
21272			<1>	; (Retro UNIX Prototype : 18/11/2012 - 11/11/2012, UNIXCOPY.ASM)
21273			<1>	; ((Modified registers: DX, BX, CX, SI, DI, BP))
21274			<1>	
21275	00005AF0	31C9	<1>	xor ecx, ecx
21276	00005AF2	890D[A8740000]	<1>	mov [u.nread], ecx ; 0
21277			<1>	; clr u.nread / clear the number of bytes transmitted during
21278			<1>	; / read or write calls
21279	00005AF8	66890D[DF740000]	<1>	mov [u.pcount], cx ; 19/05/2015
21280	00005AFF	390D[A4740000]	<1>	cmp [u.count], ecx
21281			<1>	; ; tst u.count / test the byte count specified by the user
21282	00005B05	7701	<1>	ja short writei_1 ; 1f
21283			<1>	; bgt 1f / any bytes to output; yes, branch
21284	00005B07	C3	<1>	retn
21285			<1>	; ; rts r0 / no, return - no writing to do
21286			<1>	writei_1: ;1:
21287			<1>	; mov r1 ,-(sp) / save the i-node number on the stack
21288	00005B08	6683F828	<1>	cmp ax, 40
21289			<1>	; cmp r1,\$40.
21290			<1>	; / does the i-node number indicate a special file?
21291	00005B0C	0F87F2000000	<1>	ja dskw
21292			<1>	; bgt dskw / no, branch to standard file output
21293			<1>	; (20/05/2015)
21294	00005B12	50	<1>	push eax ; because subroutines will jump to 'ret_'
21295	00005B13	0FB6D8	<1>	movzx ebx, al
21296	00005B16	66C1E302	<1>	shl bx, 2
21297			<1>	; asl r1 / yes, calculate the index into the special file
21298	00005B1A	81C3[1E5B0000]	<1>	add ebx, writei_2 - 4
21299	00005B20	FF23	<1>	jmp dword [ebx]
21300			<1>	; jmp *1f-2(r1)
21301			<1>	; / jump table and jump to the appropriate routine
21302			<1>	writei_2: ;1:
21303	00005B22	[6E5B0000]	<1>	dd tty ; tty, AX = 1 (runix)
21304			<1>	;tty / tty; r1=2
21305			<1>	;wppt / ppt; r1=4
21306	00005B26	[D45B0000]	<1>	dd wmem ; mem, AX = 2 (runix)
21307			<1>	;wmem / mem; r1=6
21308			<1>	;wrf0 / rf0
21309			<1>	;wrk0 / rk0
21310			<1>	;wtap / tap0
21311			<1>	;wtap / tap1
21312			<1>	;wtap / tap2
21313			<1>	;wtap / tap3
21314			<1>	;wtap / tap4
21315			<1>	;wtap / tap5



21316		<1>	;wtap / tap6
21317		<1>	;wtap / tap7
21318	00005B2A [0F610000]	<1>	dd wfd ; fd0, AX = 3 (runix only)
21319	00005B2E [0F610000]	<1>	dd wfd ; fd1, AX = 4 (runix only)
21320	00005B32 [0F610000]	<1>	dd whd ; hd0, AX = 5 (runix only)
21321	00005B36 [0F610000]	<1>	dd whd ; hd1, AX = 6 (runix only)
21322	00005B3A [0F610000]	<1>	dd whd ; hd2, AX = 7 (runix only)
21323	00005B3E [0F610000]	<1>	dd whd ; hd3, AX = 8 (runix only)
21324	00005B42 [C55B0000]	<1>	dd wlpr ; lpr, AX = 9 (runix)
21325	00005B46 [BF5B0000]	<1>	dd xmtt ; tty0, AX = 10 (runix)
21326		<1>	;xmtt / tty0
21327	00005B4A [BF5B0000]	<1>	dd xmtt ; tty1, AX = 11 (runix)
21328		<1>	;xmtt / tty1
21329	00005B4E [BF5B0000]	<1>	dd xmtt ; tty2, AX = 12 (runix)
21330		<1>	;xmtt / tty2
21331	00005B52 [BF5B0000]	<1>	dd xmtt ; tty3, AX = 13 (runix)
21332		<1>	;xmtt / tty3
21333	00005B56 [BF5B0000]	<1>	dd xmtt ; tty4, AX = 14 (runix)
21334		<1>	;xmtt / tty4
21335	00005B5A [BF5B0000]	<1>	dd xmtt ; tty5, AX = 15 (runix)
21336		<1>	;xmtt / tty5
21337	00005B5E [BF5B0000]	<1>	dd xmtt ; tty6, AX = 16 (runix)
21338		<1>	;xmtt / tty6
21339	00005B62 [BF5B0000]	<1>	dd xmtt ; tty7, AX = 17 (runix)
21340		<1>	;xmtt / tty7
21341	00005B66 [BF5B0000]	<1>	dd xmtt ; COM1, AX = 18 (runix only)
21342		<1>	; / wlpr / lpr
21343	00005B6A [BF5B0000]	<1>	dd xmtt ; COM2, AX = 19 (runix only)
21344		<1>	
21345		<1>	wtty: ; write to console tty (write to screen)
21346		<1>	; 18/11/2015
21347		<1>	; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21348		<1>	; 12/03/2013 - 07/07/2014 (Retro UNIX 8086 v1)
21349		<1>	;
21350		<1>	; Console tty output is on current video page
21351		<1>	; Console tty character output procedure is changed here
21352		<1>	; according to IBM PC compatible ROM BIOS video (text mode) functions.
21353		<1>	;
21354	00005B6E 0FB61D[C9740000]	<1>	movzx ebx, byte [u.uno] ; process number
21355	00005B75 8AA3[D5710000]	<1>	mov ah, [ebx+p.ttyc-1] ; current/console tty
21356	00005B7B 88E0	<1>	mov al, ah ; 07/07/2014
21357		<1>	wttys:
21358		<1>	; 10/10/2013
21359	00005B7D 8825[CE740000]	<1>	mov [u.tty], ah
21360		<1>	; 13/01/2014
21361	00005B83 FEC0	<1>	inc al
21362	00005B85 A2[B1740000]	<1>	mov [u.ttyp+1], al ; tty number + 1
21363		<1>	wtty_nc: ; 15/05/2013
21364		<1>	; AH = [u.tty] = tty number ; 28/07/2013
21365	00005B8A E81C010000	<1>	call cpass
21366		<1>	; jsr r0,cpass / get next character from user buffer area; if
21367		<1>	; / none go to return address in syswrite
21368		<1>	; tst r1 / is character = null
21369		<1>	; beq wtty / yes, get next character
21370		<1>	; 10/10/2013
21371	00005B8F 742C	<1>	jz short wret
21372		<1>	;1 :
21373		<1>	;mov \$240,*\$ps / no, set processor priority to five
21374		<1>	;cmpb cc+1,\$20. / is character count for console tty greater
21375		<1>	; / than 20
21376		<1>	;bhis 2f / yes; branch to put process to sleep
21377		<1>	; 27/06/2014
21378		<1>	wtty_1:
21379		<1>	; AH = tty number
21380		<1>	; AL= ASCII code of the character
21381		<1>	; 15/04/2014
21382	00005B91 6650	<1>	push ax
21383	00005B93 E8AF0A0000	<1>	call putc ; 14/05/2013
21384	00005B98 731F	<1>	jnc short wtty_2
21385		<1>	; 18/11/2015
21386	00005B9A E8EEF8FFFF	<1>	call idle
21387	00005B9F 668B0424	<1>	mov ax, [esp]
21388	00005BA3 E89F0A0000	<1>	call putc
21389	00005BA8 730F	<1>	jnc short wtty_2
21390		<1>	; 02/06/2014
21391	00005BAA 8A25[CE740000]	<1>	mov ah, [u.tty]
21392	00005BB0 E86CF9FFFF	<1>	call sleep
21393	00005BB5 6658	<1>	pop ax
21394	00005BB7 EBD8	<1>	jmp short wtty_1
21395		<1>	; jc error ; 15/05/2013 (COM1 or COM2 serial port error)
21396		<1>	; jsr r0,putc; 1 / find place in freelist to assign to
21397		<1>	; / console tty and
21398		<1>	; br 2f / place character in list; if none available
21399		<1>	; / branch to put process to sleep
21400		<1>	; jsr r0,startty / attempt to output character on tty
21401		<1>	wtty_2:
21402		<1>	; 15/04/2014
21403	00005BB9 6658	<1>	pop ax
21404	00005BBB EBCD	<1>	jmp short wtty_nc
21405		<1>	; br wtty
21406		<1>	wret: ; 10/10/2013 (20/05/2015)
21407	00005BBD 58	<1>	pop eax

```

21408 00005BBE C3      <1>  retn
21409                  <1>  ;2:
21410                  <1>      ;mov  r1,-(sp) / place character on stack
21411                  <1>      ;jsr  r0,sleep; 1 / put process to sleep
21412                  <1>      ;mov  (sp)+,r1 / remove character from stack
21413                  <1>      ;br   1b / try again to place character in clist and output
21414                  <1>
21415                  <1> xmtt:  ; < send/write character to tty >
21416                  <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21417                  <1>  ; 15/05/2013 - 06/12/2013 (Retro UNIX 8086 v1)
21418                  <1>  ;
21419                  <1>  ; Retro UNIX 8086 v1 modification !
21420                  <1>  ;
21421                  <1>  ; In original UNIX v1, 'xmtt' routine
21422                  <1>  ;      (exactly different than this one)
21423                  <1>  ;      was in 'u9.s' file.
21424                  <1>  ;
21425 00005BBF 2C0A      <1>  sub   al, 10
21426                  <1>  ; AL = tty number (0 to 9), (COM1=8, COM2=9)
21427                  <1>  ; 10/10/2013
21428 00005BC1 88C4      <1>  mov   ah, al
21429                  <1>  ; 28/07/2013
21430 00005BC3 EBB8      <1>  jmp   short wttys
21431                  <1>
21432                  <1> ;wppt:
21433                  <1> ;jsr   r0,cpass / get next character from user buffer area,
21434                  <1> ;      / if none return to writei's calling routine
21435                  <1> ;jsr   r0,pptoc / output character on ppt
21436                  <1> ;br    wppt
21437                  <1> wlpr:
21438 00005BC5 C705[CF740000]0F00- <1>      mov     dword [u.error], ERR_DEV_NOT_RDY ; 19/05/2015
21439 00005BCD 0000      <1>
21440 00005BCF E9A8E4FFFF      <1>  jmp   error  ; ... Printing procedure will be located here ...
21441                  <1>      ;/   jsr   r0,cpass
21442                  <1>      ;/   cmp   r0,$'a
21443                  <1>      ;/   blo   1f
21444                  <1>      ;/   cmp   r1,$'z
21445                  <1>      ;/   bhi   1f
21446                  <1>      ;/   sub   $40,r1
21447                  <1>      ;/1:
21448                  <1>      ;/   jsr   r0,lptoc
21449                  <1>      ;/   br    wlpr
21450                  <1>      ; br rmem / continue
21451                  <1>
21452                  <1> wmem: ; / transfer characters from a user area of core to memory file
21453                  <1>  ; 17/10/2015
21454                  <1>  ; 11/06/2015
21455                  <1>  ; 24/05/2015
21456                  <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21457                  <1>  ;
21458 00005BD4 813D[1B080000]- <1>  cmp   dword [x_timer], clock ; multi tasking clock/timer
21459 00005BDA [B1540000]      <1>
21460 00005BDE 7415      <1>      je     short wmem_acc_err
21461                  <1>  ;
21462 00005BE0 8B35[90740000]      <1>      mov     esi, [u.fofp]
21463                  <1> wmem_1:
21464 00005BE6 E8C0000000      <1>  call  cpass
21465                  <1>      ; jsr r0,cpass / get next character from users area of
21466                  <1>      ; / core and put it in r1
21467                  <1>      ; mov r1,-(sp) / put character on the stack
21468                  <1>  ; 20/09/2013
21469 00005BEB 74D0      <1>  jz     short wret ; wmem_2
21470 00005BED 8B1E      <1>      mov     ebx, [esi]
21471                  <1>      ; mov *u.fofp,r1 / save file offset in r1
21472 00005BEF FF06      <1>      inc     dword [esi] ; 17/10/2015
21473                  <1>      ; inc *u.fofp / increment file offset to point to next
21474                  <1>      ; / available location in file
21475 00005BF1 8803      <1>  mov     [ebx], al
21476                  <1>      ; movb (sp)+,(r1) / pop char off stack, put in memory loc
21477                  <1>      ; / assigned to it
21478 00005BF3 EBF1      <1>  jmp     short wmem_1
21479                  <1>      ; br wmem / continue
21480                  <1>  ;1:
21481                  <1>  ;jmp error / ?
21482                  <1> ;wmem_2:
21483                  <1> ; ; 20/09/2013
21484                  <1> ;pop  ax
21485                  <1> ;retn
21486                  <1>
21487                  <1> wmem_acc_err:
21488 00005BF5 C705[CF740000]0B00- <1>  mov     dword [u.error], ERR_FILE_ACCESS ; permission denied !
21489 00005BFD 0000      <1>
21490 00005BFF E978E4FFFF      <1>  jmp     error
21491                  <1>
21492                  <1>
21493                  <1> dskw: ; / write routine for non-special files
21494                  <1>  ;
21495                  <1>  ; 25/07/2015
21496                  <1>  ; 16/06/2015
21497                  <1>  ; 09/06/2015
21498                  <1>  ; 31/05/2015 (Retro UNIX 386 v1 - Beginning)
21499                  <1>  ; 26/04/2013 - 20/09/2013 (Retro UNIX 8086 v1)

```

```

21500                                     <1> ;
21501                                     <1> ; 01/08/2013 (mkdir_w check)
21502 00005C04 6650                       <1> push ax ; 26/04/2013
21503                                     <1> ; mov (sp),r1 / get an i-node number from the stack into r1
21504                                     <1> ; AX = inode number
21505 00005C06 E80FFBFFFF                 <1> call iget
21506                                     <1> ; jsr r0,iget / write i-node out (if modified),
21507                                     <1> ; / read i-node 'r1' into i-node area of core
21508 00005C0B 8B1D[90740000]             <1> mov ebx, [u.fofp]
21509 00005C11 8B13                       <1> mov edx, [ebx]
21510                                     <1> ; mov *u.fofp,r2 / put the file offset [(u.off) or the offset
21511                                     <1> ; / in the fsp entry for this file] in r2
21512 00005C13 0315[A4740000]             <1> add edx, [u.count]
21513                                     <1> ; add u.count,r2 / no. of bytes to be written
21514                                     <1> ; / + file offset is put in r2
21515                                     <1> ; 16/06/2015
21516 00005C19 81FAFFFF0000              <1> cmp edx, 65535 ; file size limit (for UNIX v1 file system)
21517 00005C1F 760F                       <1> jna short dskw_0
21518 00005C21 C705[CF740000]1400-        <1> mov dword [u.error], ERR_FILE_SIZE ; 'file size error !'
21519 00005C29 0000                       <1>
21520 00005C2B E94CE4FFFF                 <1> jmp error
21521                                     <1> dskw_0:
21522 00005C30 663B15[5A710000]           <1> cmp dx, [i.size]
21523                                     <1> ; cmp r2,i.size / is this greater than the present size of
21524                                     <1> ; / the file?
21525 00005C37 760C                       <1> jna short dskw_1
21526                                     <1> ; blos 1f / no, branch
21527 00005C39 668915[5A710000]           <1> mov [i.size], dx
21528                                     <1> ; mov r2,i.size / yes, increase the file size to
21529                                     <1> ; / file offset + no. of data bytes
21530 00005C40 E8E8FBFFFF                 <1> call setimod
21531                                     <1> ; jsr r0,setimod / set imod=1 (i.e., core inode has been
21532                                     <1> ; / modified), stuff time of modification into
21533                                     <1> ; / core image of i-node
21534                                     <1> dskw_1: ; 1:
21535 00005C45 E85DF9FFFF                 <1> call mget
21536                                     <1> ; eAX = Block number
21537                                     <1> ; jsr r0,mget / get the block no. in which to write
21538                                     <1> ; / the next data byte
21539                                     <1> ; eax = block number
21540 00005C4A 8B1D[90740000]             <1> mov ebx, [u.fofp]
21541 00005C50 8B13                       <1> mov edx, [ebx]
21542 00005C52 81E2FF010000               <1> and edx, 1FFh
21543                                     <1> ; bit *u.fofp,$777 / test the lower 9 bits of the file offset
21544 00005C58 750C                       <1> jnz short dskw_2
21545                                     <1> ; bne 2f / if its non-zero, branch; if zero, file offset = 0,
21546                                     <1> ; / 512, 1024,...(i.e., start of new block)
21547 00005C5A 813D[A4740000]0002-        <1> cmp dword [u.count], 512
21548 00005C62 0000                       <1>
21549                                     <1> ; cmp u.count,$512. / if zero, is there enough data to fill
21550                                     <1> ; / an entire block? (i.e., no. of
21551 00005C64 7305                       <1> jnb short dskw_3
21552                                     <1> ; bhis 3f / bytes to be written greater than 512.?
21553                                     <1> ; / Yes, branch. Don't have to read block
21554                                     <1> dskw_2: ; 2: / in as no past info. is to be saved (the entire block will be
21555                                     <1> ; / overwritten).
21556 00005C66 E87C050000                 <1> call dskrd
21557                                     <1> ; jsr r0,dskrd / no, must retain old info..
21558                                     <1> ; / Hence, read block 'r1' into an I/O buffer
21559                                     <1> dskw_3: ; 3:
21560                                     <1> ; eAX (r1) = block/sector number
21561 00005C6B E8D7050000                 <1> call wslot
21562                                     <1> ; jsr r0,wslot / set write and inhibit bits in I/O queue,
21563                                     <1> ; / proc. status=0, r5 points to 1st word of data
21564 00005C70 803D[E1740000]00           <1> cmp byte [u.kcall], 0
21565 00005C77 770F                       <1> ja short dskw_5 ; zf=0 -> the caller is 'mkdir'
21566                                     <1> ;
21567 00005C79 66833D[DF740000]00         <1> cmp word [u.pcount], 0
21568 00005C81 7705                       <1> ja short dskw_5
21569                                     <1> dskw_4:
21570                                     <1> ; [u.base] = virtual address to transfer (as source address)
21571 00005C83 E806FEFFFF                 <1> call trans_addr_r ; translate virtual address to physical (r)
21572                                     <1> dskw_5:
21573                                     <1> ; eBX (r5) = system (I/O) buffer address
21574 00005C88 E87E000000                 <1> call sioreg
21575                                     <1> ; jsr r0,sioreg / r3 = no. of bytes of data,
21576                                     <1> ; / r1 = address of data, r2 points to location
21577                                     <1> ; / in buffer in which to start writing data
21578                                     <1> ; eSI = file (user data) offset
21579                                     <1> ; eDI = sector (I/O) buffer offset
21580                                     <1> ; eCX = byte count
21581                                     <1> ;
21582 00005C8D F3A4                       <1> rep movsb
21583                                     <1> ; movb (r1)+,(r2)+
21584                                     <1> ; / transfer a byte of data to the I/O buffer
21585                                     <1> ; dec r3 / decrement no. of bytes to be written
21586                                     <1> ; bne 2b / have all bytes been transferred? No, branch
21587                                     <1> ; 25/07/2015
21588                                     <1> ; eax = remain bytes in buffer
21589                                     <1> ; (check if remain bytes in the buffer > [u.pcount])
21590 00005C8F 09C0                       <1> or eax, eax
21591 00005C91 75F0                       <1> jnz short dskw_4 ; (page end before system buffer end!)

```

```

21592                                <1> dskw_6:
21593 00005C93 E8CB050000          <1>  call  dskwr
21594                                <1>      ; jsr r0,dskwr / yes, write the block and the i-node
21595 00005C98 833D[A4740000]00    <1>      cmp     dword [u.count], 0
21596                                <1>      ; tst u.count / any more data to write?
21597 00005C9F 77A4                <1>      ja     short dskw_1
21598                                <1>      ; bne lb / yes, branch
21599                                <1>      ; 03/08/2013
21600 00005CA1 C605[E1740000]00    <1>      mov     byte [u.kcall], 0
21601                                <1>      ; 20/09/2013 (;;)
21602 00005CA8 6658                <1>      pop     ax
21603 00005CAA C3                  <1>      retn
21604                                <1>      ;;jmp     short dskw_ret
21605                                <1>      ; jmp ret / no, return to the caller via 'ret'
21606                                <1>
21607                                <1> cpass: ; / get next character from user area of core and put it in r1
21608                                <1>      ; 18/10/2015
21609                                <1>      ; 10/10/2015
21610                                <1>      ; 10/07/2015
21611                                <1>      ; 02/07/2015
21612                                <1>      ; 01/07/2015
21613                                <1>      ; 24/06/2015
21614                                <1>      ; 08/06/2015
21615                                <1>      ; 04/06/2015
21616                                <1>      ; 20/05/2015
21617                                <1>      ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21618                                <1>      ;
21619                                <1>      ; INPUTS ->
21620                                <1>      ;      [u.base] = virtual address in user area
21621                                <1>      ;      [u.count] = byte count (max.)
21622                                <1>      ;      [u.pcount] = byte count in page (0 = reset)
21623                                <1>      ; OUTPUTS ->
21624                                <1>      ;      AL = the character which is pointed by [u.base]
21625                                <1>      ;      zf = 1 -> transfer count has been completed
21626                                <1>      ;
21627                                <1>      ; ((Modified registers:  EAX, EDX, ECX))
21628                                <1>      ;
21629                                <1>      ;
21630 00005CAB 833D[A4740000]00    <1>      cmp     dword [u.count], 0 ; 14/08/2013
21631                                <1>      ; tst u.count / have all the characters been transferred
21632                                <1>      ; / (i.e., u.count, # of chars. left
21633 00005CB2 763F                <1>      jna     short cpass_3
21634                                <1>      ; beq lf / to be transferred = 0?) yes, branch
21635 00005CB4 FF0D[A4740000]      <1>      dec     dword [u.count]
21636                                <1>      ; dec u.count / no, decrement u.count
21637                                <1>      ; 19/05/2015
21638                                <1>      ; (Retro UNIX 386 v1 - translation from user's virtual address
21639                                <1>      ;      to physical address
21640 00005CBA 66833D[DF740000]00    <1>      cmp     word [u.pcount], 0 ; byte count in page = 0 (initial value)
21641                                <1>      ; 1-4095 --> use previous physical base address
21642                                <1>      ; in [u.pbase]
21643 00005CC2 770E                <1>      ja     short cpass_1
21644                                <1>      ; 02/07/2015
21645 00005CC4 833D[D7740000]00    <1>      cmp     dword [u.ppgdir], 0 ; is the caller os kernel
21646 00005CCB 7427                <1>      je      short cpass_k ; (sysexec, '/etc/init') ?
21647                                <1>      ; 08/06/2015 - 10/07/2015
21648 00005CCD E8BCFDFFFF          <1>      call  trans_addr_r
21649                                <1> cpass_1:
21650                                <1>      ; 02/07/2015
21651                                <1>      ; 24/06/2015
21652 00005CD2 66FF0D[DF740000]      <1>      dec     word [u.pcount]
21653                                <1> cpass_2:
21654                                <1>      ;10/10/2015
21655                                <1>      ; 02/07/2015
21656 00005CD9 8B15[DB740000]      <1>      mov     edx, [u.pbase]
21657 00005CDF 8A02                <1>      mov     al, [edx] ; 10/10/2015
21658                                <1>      ; movb *u.base,r1 / take the character pointed to
21659                                <1>      ; / by u.base and put it in r1
21660 00005CE1 FF05[A8740000]      <1>      inc     dword [u.nread]
21661                                <1>      ; inc u.nread / increment no. of bytes transferred
21662 00005CE7 FF05[A0740000]      <1>      inc     dword [u.base]
21663                                <1>      ; inc u.base / increment the buffer address to point to the
21664                                <1>      ; / next byte
21665 00005CED FF05[DB740000]      <1>      inc     dword [u.pbase] ; 04/06/2015
21666                                <1> cpass_3:
21667 00005CF3 C3                  <1>      retn
21668                                <1>      ; rts r0 / next byte
21669                                <1>      ; 1:
21670                                <1>      ; mov (sp)+,r0
21671                                <1>      ; / put return address of calling routine into r0
21672                                <1>      ; mov (sp)+,r1 / i-number in r1
21673                                <1>      ; rts r0 / non-local return
21674                                <1> cpass_k:
21675                                <1>      ; 02/07/2015
21676                                <1>      ; The caller is os kernel
21677                                <1>      ; (get sysexec arguments from kernel's memory space)
21678                                <1>      ;
21679 00005CF4 8B1D[A0740000]      <1>      mov     ebx, [u.base]
21680 00005CFA 66C705[DF740000]00- <1>      mov     word [u.pcount], PAGE_SIZE ; 4096
21681 00005D02 10                  <1>
21682 00005D03 891D[DB740000]      <1>      mov     [u.pbase], ebx
21683 00005D09 EBCE                <1>      jmp     short cpass_2

```

21684	<1>
21685	<1> sioreg:
21686	<1> ; 25/07/2015
21687	<1> ; 18/07/2015
21688	<1> ; 02/07/2015
21689	<1> ; 17/06/2015
21690	<1> ; 09/06/2015
21691	<1> ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
21692	<1> ; 12/03/2013 - 22/07/2013 (Retro UNIX 8086 v1)
21693	<1> ;
21694	<1> ; INPUTS ->
21695	<1> ; eBX = system buffer (data) address (r5)
21696	<1> ; [u.fofp] = pointer to file offset pointer
21697	<1> ; [u.base] = virtual address of the user buffer
21698	<1> ; [u.pbase] = physical address of the user buffer
21699	<1> ; [u.count] = byte count
21700	<1> ; [u.pcount] = byte count within page frame
21701	<1> ; OUTPUTS ->
21702	<1> ; eSI = user data offset (r1)
21703	<1> ; eDI = system (I/O) buffer offset (r2)
21704	<1> ; eCX = byte count (r3)
21705	<1> ; EAX = remain bytes after byte count within page frame
21706	<1> ; (If EAX > 0, transfer will continue from the next page)
21707	<1> ;
21708	<1> ; ((Modified registers: EDX))
21709	<1>
21710 00005D0B 8B35[90740000]	<1> mov esi, [u.fofp]
21711 00005D11 8B3E	<1> mov edi, [esi]
21712	<1> ; mov *u.fofp,r2 / file offset (in bytes) is moved to r2
21713 00005D13 89F9	<1> mov ecx, edi
21714	<1> ; mov r2,r3 / and also to r3
21715 00005D15 81C900FEFFFF	<1> or ecx, 0FFFFFFE00h
21716	<1> ; bis \$177000,r3 / set bits 9,...,15 of file offset in r3
21717 00005D1B 81E7FF010000	<1> and edi, 1FFh
21718	<1> ; bic \$!777,r2 / calculate file offset mod 512.
21719 00005D21 01DF	<1> add edi, ebx ; EBX = system buffer (data) address
21720	<1> ; add r5,r2 / r2 now points to 1st byte in system buffer
21721	<1> ; / where data is to be placed
21722	<1> ; mov u.base,r1 / address of data is in r1
21723 00005D23 F7D9	<1> neg ecx
21724	<1> ; neg r3 / 512 - file offset (mod512.) in r3
21725	<1> ; / (i.e., the no. of free bytes in the file block)
21726 00005D25 3B0D[A4740000]	<1> cmp ecx, [u.count]
21727	<1> ; cmp r3,u.count / compare this with the no. of data bytes
21728	<1> ; / to be written to the file
21729 00005D2B 7606	<1> jna short sioreg_0
21730	<1> ; blos 2f / if less than branch. Use the no. of free bytes
21731	<1> ; / in the file block as the number to be written
21732 00005D2D 8B0D[A4740000]	<1> mov ecx, [u.count]
21733	<1> ; mov u.count,r3 / if greater than, use the no. of data
21734	<1> ; / bytes as the number to be written
21735	<1> sioreg_0:
21736	<1> ; 17/06/2015
21737 00005D33 803D[E1740000]00	<1> cmp byte [u.kcall], 0
21738 00005D3A 7613	<1> jna short sioreg_1
21739	<1> ; 25/07/2015
21740	<1> ; the caller is 'mkdir' or 'namei'
21741 00005D3C A1[A0740000]	<1> mov eax, [u.base] ; 25/07/2015
21742 00005D41 A3[DB740000]	<1> mov [u.pbase], eax ; physical address = virtual address
21743 00005D46 66890D[DF740000]	<1> mov word [u.pcount], cx ; remain bytes in buffer (1 sector)
21744 00005D4D EB0B	<1> jmp short sioreg_2
21745	<1> sioreg_1:
21746	<1> ; 25/07/2015
21747	<1> ; 18/07/2015
21748	<1> ; 09/06/2015
21749 00005D4F 0FB715[DF740000]	<1> movzx edx, word [u.pcount]
21750	<1> ; ecx and [u.pcount] are always > 0, here
21751 00005D56 39D1	<1> cmp ecx, edx
21752 00005D58 772A	<1> ja short sioreg_4 ; transfer count > [u.pcount]
21753	<1> sioreg_2: ; 2:
21754 00005D5A 31C0	<1> xor eax, eax ; 25/07/2015
21755	<1> sioreg_3:
21756 00005D5C 010D[A8740000]	<1> add [u.nread], ecx
21757	<1> ; add r3,u.nread / r3 + number of bytes xmitted
21758	<1> ; / during write is put into u.nread
21759 00005D62 290D[A4740000]	<1> sub [u.count], ecx
21760	<1> ; sub r3,u.count / u.count = no. of bytes that still
21761	<1> ; / must be written or read
21762 00005D68 010D[A0740000]	<1> add [u.base], ecx
21763	<1> ; add r3,u.base / u.base points to the 1st of the remaining
21764	<1> ; / data bytes
21765 00005D6E 010E	<1> add [esi], ecx
21766	<1> ; add r3,*u.fofp / new file offset = number of bytes done
21767	<1> ; / + old file offset
21768	<1> ; 25/07/2015
21769 00005D70 8B35[DB740000]	<1> mov esi, [u.pbase]
21770 00005D76 66290D[DF740000]	<1> sub [u.pcount], cx
21771 00005D7D 010D[DB740000]	<1> add [u.pbase], ecx
21772 00005D83 C3	<1> retn
21773	<1> ; rts r0
21774	<1> ; transfer count > [u.pcount]
21775	<1> sioreg_4:

```

21776      <1> ; 25/07/2015
21777      <1> ; transfer count > [u.pcount]
21778      <1> ; (ecx > edx)
21779 00005D84 89C8      <1> mov    eax, ecx
21780 00005D86 29D0      <1> sub    eax, edx ; remain bytes for 1 sector (block) transfer
21781 00005D88 89D1      <1> mov    ecx, edx ; current transfer count = [u.pcount]
21782 00005D8A EBD0      <1> jmp    short sioreg_3
21783      %include 'u7.s' ; 18/04/2015
21784      <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS7.INC
21785      <1> ; Last Modification: 14/11/2015
21786      <1> ; -----
21787      <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
21788      <1> ; (v0.1 - Beginning: 11/07/2012)
21789      <1> ;
21790      <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
21791      <1> ; (Original) Source Code by Ken Thompson (1971-1972)
21792      <1> ; <Bell Laboratories (17/3/1972)>
21793      <1> ; <Preliminary Release of UNIX Implementation Document>
21794      <1> ;
21795      <1> ; Retro UNIX 8086 v1 - U7.ASM (13/07/2014) //// UNIX v1 -> u7.s
21796      <1> ;
21797      <1> ; *****
21798      <1>
21799      <1> sysmount: ; / mount file system; args special; name
21800      <1> ; 14/11/2015
21801      <1> ; 24/10/2015
21802      <1> ; 13/10/2015
21803      <1> ; 10/07/2015
21804      <1> ; 16/05/2015 (Retro UNIX 386 v1 - Beginning)
21805      <1> ; 09/07/2013 - 04/11/2013 (Retro UNIX 8086 v1)
21806      <1> ;
21807      <1> ; 'sysmount' announces to the system that a removable
21808      <1> ; file system has been mounted on a special file.
21809      <1> ; The device number of the special file is obtained via
21810      <1> ; a call to 'getspl'. It is put in the I/O queue entry for
21811      <1> ; dismountable file system (sbl) and the I/O queue entry is
21812      <1> ; set up to read (bit 10 is set). 'ppoke' is then called to
21813      <1> ; to read file system into core, i.e. the first block on the
21814      <1> ; mountable file system is read in. This block is super block
21815      <1> ; for the file system. This call is super user restricted.
21816      <1> ;
21817      <1> ; Calling sequence:
21818      <1> ; sysmount; special; name
21819      <1> ; Arguments:
21820      <1> ; special - pointer to name of special file (device)
21821      <1> ; name - pointer to name of the root directory of the
21822      <1> ; newly mounted file system. 'name' should
21823      <1> ; always be a directory.
21824      <1> ; Inputs: -
21825      <1> ; Outputs: -
21826      <1> ; .....
21827      <1> ;
21828      <1> ; Retro UNIX 8086 v1 modification:
21829      <1> ; 'sysmount' system call has two arguments; so,
21830      <1> ; * 1st argument, special is pointed to by BX register
21831      <1> ; * 2nd argument, name is in CX register
21832      <1> ;
21833      <1> ; NOTE: Device numbers, names and related procedures are
21834      <1> ; already modified for IBM PC compatibility and
21835      <1> ; Retro UNIX 8086 v1 device configuration.
21836      <1>
21837      <1> ;call arg2
21838      <1> ; jsr r0,arg2 / get arguments special and name
21839 00005D8C 891D[98740000]      <1> mov    [u.namep], ebx
21840 00005D92 51      <1> push   ecx ; directory name
21841 00005D93 66833D[6C740000]00      <1> cmp    word [mnti], 0
21842      <1> ; tst mnti / is the i-number of the cross device file
21843      <1> ; / zero?
21844      <1> ;ja error
21845      <1> ; bne errora / no, error
21846 00005D9B 0F87E9000000      <1> ja     sysmnt_err0
21847      <1> ;
21848 00005DA1 E8CC000000      <1> call   getspl
21849      <1> ; jsr r0,getspl / get special files device number in r1
21850      <1> ; 13/10/2015
21851 00005DA6 0FB7D8      <1> movzx  ebx, ax ; ; Retro UNIX 8086 v1 device number (0 to 5)
21852 00005DA9 F683[246E0000]80      <1> test   byte [ebx+drv.status], 80h ; 24/10/2015
21853 00005DB0 750F      <1> jnz    short sysmnt_1
21854      <1> sysmnt_err1:
21855 00005DB2 C705[CF740000]0F00-      <1> mov     dword [u.error], ERR_DRV_NOT_RDY ; drive not ready !
21856 00005DBA 0000      <1>
21857 00005DBC E9BBE2FFFF      <1> jmp     error
21858      <1> sysmnt_1:
21859 00005DC1 8F05[98740000]      <1> pop     dword [u.namep]
21860      <1> ; mov (sp)+,u.namep / put the name of file to be placed
21861      <1> ; / on the device
21862      <1> ; 14/11/2015
21863 00005DC7 53      <1> push    ebx ; 13/10/2015
21864      <1> ; mov r1,-(sp) / save the device number
21865      <1> ;
21866 00005DC8 E85EF1FFFF      <1> call    namei
21867      <1> ;or ax, ax ; Retro UNIX 8086 v1 modification !

```

21868		<1>	; ax = 0 -> file not found
21869		<1>	;jz error
21870		<1>	;jc error
21871		<1>	; jsr r0,namei / get the i-number of the file
21872		<1>	; br errora
21873	00005DCD 730F	<1>	jnc short sysmnt_2
21874		<1>	sysmnt_err2:
21875	00005DCF C705[CF740000]0C00-	<1>	mov dword [u.error], ERR_FILE_NOT_FOUND ; drive not ready !
21876	00005DD7 0000	<1>	
21877	00005DD9 E99EE2FFFF	<1>	jmp error
21878		<1>	sysmnt_2:
21879	00005DDE 66A3[6C740000]	<1>	mov [mnti], ax
21880		<1>	; mov r1,mnti / put it in mnti
21881		<1>	;mov ebx, sb1 ; super block buffer (of mounted disk)
21882		<1>	sysmnt_3: ;1:
21883		<1>	;cmp byte [ebx+1], 0
21884		<1>	; tstb sb1+1 / is 15th bit of I/O queue entry for
21885		<1>	; / dismountable device set?
21886		<1>	;jna short sysmnt_4
21887		<1>	; bne 1b / (inhibit bit) yes, skip writing
21888		<1>	;call idle ; (wait for hardware interrupt)
21889		<1>	;jmp short sysmnt_3
21890		<1>	sysmnt_4:
21891	00005DE4 58	<1>	pop eax ; Retro UNIX 8086 v1 device number/ID (0 to 5)
21892	00005DE5 A2[69740000]	<1>	mov [mdev], al
21893		<1>	; mov (sp),mntd / no, put the device number in mntd
21894	00005DEA 8803	<1>	mov [ebx], al
21895		<1>	; movb (sp),sb1 / put the device number in the lower byte
21896		<1>	; / of the I/O queue entry
21897		<1>	;mov byte [cdev], 1 ; mounted device/drive
21898		<1>	; mov (sp)+,cdev / put device number in cdev
21899	00005DEC 66810B0004	<1>	or word [ebx], 400h ; Bit 10, 'read' flag/bit
21900		<1>	; bis \$2000,sb1 / set the read bit
21901		<1>	; Retro UNIX 386 v1 modification :
21902		<1>	; 32 bit block number at buffer header offset 4
21903	00005DF1 C7430401000000	<1>	mov dword [ebx+4], 1 ; physical block number = 1
21904	00005DF8 E8A3050000	<1>	call diskio
21905	00005DFD 731C	<1>	jnc short sysmnt_5
21906	00005DFF 31C0	<1>	xor eax, eax
21907	00005E01 66A3[6C740000]	<1>	mov [mnti], ax ; 0
21908	00005E07 A2[69740000]	<1>	mov [mdev], al ; 0
21909		<1>	;mov [cdev], al ; 0
21910		<1>	sysmnt_invd:
21911		<1>	; 14/11/2015
21912	00005E0C FEC8	<1>	dec al
21913	00005E0E 8903	<1>	mov [ebx], eax ; 000000FFh
21914	00005E10 FEC0	<1>	inc al
21915	00005E12 48	<1>	dec eax
21916	00005E13 894304	<1>	mov [ebx+4], eax ; 0FFFFFFFh
21917	00005E16 E961E2FFFF	<1>	jmp error
21918		<1>	sysmnt_5:
21919		<1>	; 14/11/2015 (Retro UNIX 386 v1 modification)
21920		<1>	; (Following check is needed to prevent mounting an
21921		<1>	; in valid valid file system (in valid super block).
21922		<1>	;
21923	00005E1B 0FB603	<1>	movzx eax, byte [ebx] ; device number
21924	00005E1E C0E002	<1>	shl al, 2 ; 4*index
21925	00005E21 8B88[086E0000]	<1>	mov ecx, [eax+drv.size] ; volume (fs) size
21926	00005E27 C1E103	<1>	shl ecx, 3
21927	00005E2A 0FB715[2C7F0000]	<1>	movzx edx, word [sb1+4] ; the 1st data word
21928	00005E31 39D1	<1>	cmp ecx, edx ; compare free map bits and volume size
21929		<1>	; (in sectors), if they are not equal
21930		<1>	; the disk to be mounted is an...
21931	00005E33 75D7	<1>	jne short sysmnt_invd ; invalid disk !
21932		<1>	; (which has not got a valid super block)
21933		<1>	;
21934	00005E35 C6430100	<1>	mov byte [ebx+1], 0
21935		<1>	; jsr r0,ppoke / read in entire file system
21936		<1>	;sysmnt_6: ;1:
21937		<1>	;;cmp byte [sb1+1], 0
21938		<1>	; tstb sb1+1 / done reading?
21939		<1>	;;jna sysret
21940		<1>	;;call idle ; (wait for hardware interrupt)
21941		<1>	;;jmp short sysmnt_6
21942		<1>	;bne 1b / no, wait
21943		<1>	;br sysreta / yes
21944	00005E39 E95EE2FFFF	<1>	jmp sysret
21945		<1>	
21946		<1>	sysumount: ; / special dismount file system
21947		<1>	; 16/05/2015 (Retro UNIX 386 v1 - Beginning)
21948		<1>	; 09/07/2013 - 04/11/2013 (Retro UNIX 8086 v1)
21949		<1>	;
21950		<1>	; 04/11/2013
21951		<1>	; 09/07/2013
21952		<1>	; 'sysumount' announces to the system that the special file,
21953		&lt	

21960		<1>	; sysmount; special
21961		<1>	; Arguments:
21962		<1>	; special - special file to dismount (device)
21963		<1>	;
21964		<1>	; Inputs: -
21965		<1>	; Outputs: -
21966		<1>	; .....
21967		<1>	;
21968		<1>	; Retro UNIX 8086 v1 modification:
21969		<1>	; 'sysumount' system call has one argument; so,
21970		<1>	; * Single argument, special is pointed to by BX register
21971		<1>	;
21972		<1>	<1>
21973		<1>	;mov ax, 1 ; one/single argument, put argument in BX
21974		<1>	;call arg
21975		<1>	; jsr r0,arg; u.namep / point u.namep to special
21976	00005E3E 891D[98740000]	<1>	mov [u.namep], ebx
21977	00005E44 E829000000	<1>	call getspl
21978		<1>	; jsr r0,getspl / get the device number in r1
21979	00005E49 3A05[69740000]	<1>	cmp al, [mdev]
21980		<1>	; cmp r1,mntd / is it equal to the last device mounted?
21981	00005E4F 7539	<1>	jne short sysmnt_err0 ; 'permission denied !' error
21982		<1>	;jne error
21983		<1>	; bne errora / no error
21984	00005E51 30C0	<1>	xor al, al ; ah = 0
21985		<1>	sysumnt_0: ;1:
21986	00005E53 3805[297F0000]	<1>	cmp [sbl+1], al ; 0
21987		<1>	; tstb sbl+1 / yes, is the device still doing I/O
21988		<1>	; / (inhibit bit set)?
21989	00005E59 7607	<1>	jna short sysumnt_1
21990		<1>	; bne 1b / yes, wait
21991	00005E5B E82DF6FFFF	<1>	call idle ; (wait for hardware interrupt)
21992	00005E60 EBF1	<1>	jmp short sysumnt_0
21993		<1>	sysumnt_1:
21994	00005E62 A2[69740000]	<1>	mov [mdev], al
21995		<1>	; clr mntd / no, clear these
21996	00005E67 66A3[6C740000]	<1>	mov [mnti], ax
21997		<1>	; clr mnti
21998	00005E6D E92AE2FFFF	<1>	jmp sysret
21999		<1>	; br sysreta / return
22000		<1>	<1>
22001		<1>	getspl: ; / get device number from a special file name
22002	00005E72 E8B4F0FFFF	<1>	call namei
22003		<1>	;or ax, ax ; Retro UNIX 8086 v1 modification !
22004		<1>	; ax = 0 -> file not found
22005	00005E77 0F8252FFFFFF	<1>	jc sysmnt_err2 ; 'file not found !' error
22006		<1>	;jz error
22007		<1>	;jc error
22008		<1>	; jsr r0,namei / get the i-number of the special file
22009		<1>	; br errora / no such file
22010	00005E7D 6683E803	<1>	sub ax, 3 ; Retro UNIX 8086 v1 modification !
22011		<1>	; i-number-3, 0 = fd0, 5 = hd3
22012		<1>	; sub \$4,r1 / i-number-4 rk=1,tap=2+n
22013	00005E81 7207	<1>	jc short sysmnt_err0 ; 'permission denied !' error
22014		<1>	;jc error
22015		<1>	; ble errora / less than 0? yes, error
22016	00005E83 6683F805	<1>	cmp ax, 5 ;
22017		<1>	; cmp r1,\$9. / greater than 9 tap 7
22018	00005E87 7701	<1>	ja short sysmnt_err0 ; 'permission denied !' error
22019		<1>	;ja error
22020		<1>	; bgt errora / yes, error
22021		<1>	; AX = Retro UNIX 8086 v1 Device Number (0 to 5)
22022		<1>	iopen_retn:
22023	00005E89 C3	<1>	retn
22024		<1>	; rts r0 / return with device number in r1
22025		<1>	sysmnt_err0:
22026	00005E8A C705[CF740000]0B00-	<1>	mov dword [u.error], ERR_FILE_ACCESS ; permission denied !
22027	00005E92 0000	<1>	<1>
22028	00005E94 E9E3E1FFFF	<1>	jmp error
22029		<1>	iopen:
22030		<1>	; 19/05/2015
22031		<1>	; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
22032		<1>	; 21/05/2013 - 27/08/2013 (Retro UNIX 8086 v1)
22033		<1>	;
22034		<1>	; open file whose i-number is in r1
22035		<1>	;
22036		<1>	; INPUTS ->
22037		<1>	; r1 - inode number
22038		<1>	; OUTPUTS ->
22039		<1>	; file's inode in core
22040		<1>	; r1 - inode number (positive)
22041		<1>	;
22042		<1>	; ((AX = R1))
22043		<1>	; ((Modified registers: edx, ebx, ecx, esi, edi, ebp))
22044		<1>	;
22045		<1>	; / open file whose i-number is in r1
22046	00005E99 F6C480	<1>	test ah, 80h ; Bit 15 of AX
22047		<1>	;tst r1 / write or read access?
22048	00005E9C 756A	<1>	jnz short iopen_2
22049		<1>	;blt 2f / write, go to 2f
22050	00005E9E B202	<1>	mov dl, 2 ; read access
22051	00005EA0 E850F9FFFF	<1>	call access



22052		<1>	; jsr r0,access; 2
22053		<1>	; / get inode into core with read access
22054		<1>	; DL=2
22055		<1>	iopen_0:
22056	00005EA5 6683F828	<1>	cmp ax, 40
22057		<1>	; cmp r1,\$40. / is it a special file
22058	00005EA9 77DE	<1>	ja short iopen_retn
22059		<1>	;bgt 3f / no. 3f
22060	00005EAB 6650	<1>	push ax
22061		<1>	; mov r1,-(sp) / yes, figure out
22062	00005EAD 0FB6D8	<1>	movzx ebx, al
22063	00005EB0 66C1E302	<1>	shl bx, 2
22064		<1>	; asl r1
22065	00005EB4 81C3[B85E0000]	<1>	add ebx, iopen_1 - 4
22066	00005EBA FF23	<1>	jmp dword [ebx]
22067		<1>	; jmp *1f-2(r1) / which one and transfer to it
22068		<1>	iopen_1: ; 1:
22069	00005EBC [225F0000]	<1>	dd otty ; tty, AX = 1 (runix)
22070		<1>	;otty / tty ; r1=2
22071		<1>	;oppt / ppt ; r1=4
22072	00005EC0 [BF5F0000]	<1>	dd sret ; mem, AX = 2 (runix)
22073		<1>	;sret / mem ; r1=6
22074		<1>	;sret / rf0
22075		<1>	;sret / rk0
22076		<1>	;sret / tap0
22077		<1>	;sret / tap1
22078		<1>	;sret / tap2
22079		<1>	;sret / tap3
22080		<1>	;sret / tap4
22081		<1>	;sret / tap5
22082		<1>	;sret / tap6
22083		<1>	;sret / tap7
22084	00005EC4 [BF5F0000]	<1>	dd sret ; fd0, AX = 3 (runix only)
22085	00005EC8 [BF5F0000]	<1>	dd sret ; fd1, AX = 4 (runix only)
22086	00005ECC [BF5F0000]	<1>	dd sret ; hd0, AX = 5 (runix only)
22087	00005ED0 [BF5F0000]	<1>	dd sret ; hd1, AX = 6 (runix only)
22088	00005ED4 [BF5F0000]	<1>	dd sret ; hd2, AX = 7 (runix only)
22089	00005ED8 [BF5F0000]	<1>	dd sret ; hd3, AX = 8 (runix only)
22090		<1>	;dd error ; lpr, AX = 9 (error !)
22091	00005EDC [BF5F0000]	<1>	dd sret ; lpr, AX = 9 (runix)
22092	00005EE0 [315F0000]	<1>	dd ocvt ; tty0, AX = 10 (runix)
22093		<1>	;ocvt / tty0
22094	00005EE4 [315F0000]	<1>	dd ocvt ; tty1, AX = 11 (runix)
22095		<1>	;ocvt / tty1
22096	00005EE8 [315F0000]	<1>	dd ocvt ; tty2, AX = 12 (runix)
22097		<1>	;ocvt / tty2
22098	00005EEC [315F0000]	<1>	dd ocvt ; tty3, AX = 13 (runix)
22099		<1>	;ocvt / tty3
22100	00005EF0 [315F0000]	<1>	dd ocvt ; tty4, AX = 14 (runix)
22101		<1>	;ocvt / tty4
22102	00005EF4 [315F0000]	<1>	dd ocvt ; tty5, AX = 15 (runix)
22103		<1>	;ocvt / tty5
22104	00005EF8 [315F0000]	<1>	dd ocvt ; tty6, AX = 16 (runix)
22105		<1>	;ocvt / tty6
22106	00005EFC [315F0000]	<1>	dd ocvt ; tty7, AX = 17 (runix)
22107		<1>	;ocvt / tty7
22108	00005F00 [315F0000]	<1>	dd ocvt ; COM1, AX = 18 (runix only)
22109		<1>	;error / crd
22110	00005F04 [315F0000]	<1>	dd ocvt ; COM2, AX = 19 (runix only)
22111		<1>	
22112		<1>	iopen_2: ; 2: / check open write access
22113	00005F08 66F7D8	<1>	neg ax
22114		<1>	;neg r1 / make inode number positive
22115	00005F0B B201	<1>	mov dl, 1 ; write access
22116	00005F0D E8E3F8FFFF	<1>	call access
22117		<1>	;jsr r0,access; 1 / get inode in core
22118		<1>	; DL=1
22119	00005F12 66F705[56710000]00-	<1>	test word [i.flgs], 4000h ; Bit 14 : Directory flag
22120	00005F1A 40	<1>	
22121		<1>	;bit \$40000,i.flgs / is it a directory?
22122	00005F1B 7488	<1>	jz short iopen_0
22123		<1>	;mov [u.error], ERR_DIR_ACCESS
22124		<1>	;jmp error ; permission denied !
22125	00005F1D E968FFFFFF	<1>	jmp sysmnt_err0
22126		<1>	;jnz error
22127		<1>	; bne 2f / yes, transfer (error)
22128		<1>	;jmp short iopen_0
22129		<1>	;cmp ax, 40
22130		<1>	; cmp r1,\$40. / no, is it a special file?
22131		<1>	;ja short iopen_2
22132		<1>	;bgt 3f / no, return
22133		<1>	;push ax
22134		<1>	;mov r1,-(sp) / yes
22135		<1>	;movzx ebx, al
22136		<1>	;shl bx, 1
22137		<1>	; asl r1
22138		<1>	;add ebx, ipen_3 - 2
22139		<1>	;jmp dword [ebx]
22140		<1>	; jmp *1f-2(r1) / figure out
22141		<1>	; / which special file it is and transfer
22142		<1>	;iopen_3: ; 1:
22143		<1>	;dd otty ; tty, AX = 1 (runix)

22144		<1>	;otty / tty ; r1=2
22145		<1>	;leadr / ppt ; r1=4
22146		<1>	;dd sret ; mem, AX = 2 (runix)
22147		<1>	;sret / mem ; r1=6
22148		<1>	;sret / rf0
22149		<1>	;sret / rk0
22150		<1>	;sret / tap0
22151		<1>	;sret / tap1
22152		<1>	;sret / tap2
22153		<1>	;sret / tap3
22154		<1>	;sret / tap4
22155		<1>	;sret / tap5
22156		<1>	;sret / tap6
22157		<1>	;sret / tap7
22158		<1>	;dd sret ; fd0, AX = 3 (runix only)
22159		<1>	;dd sret ; fd1, AX = 4 (runix only)
22160		<1>	;dd sret ; hd0, AX = 5 (runix only)
22161		<1>	;dd sret ; hd1, AX = 6 (runix only)
22162		<1>	;dd sret ; hd2, AX = 7 (runix only)
22163		<1>	;dd sret ; hd3, AX = 8 (runix only)
22164		<1>	;dd sret ; lpr, AX = 9 (runix)
22165		<1>	;dd ejec ; lpr, AX = 9 (runix)
22166		<1>	;dd sret ; tty0, AX = 10 (runix)
22167		<1>	;ocvt / tty0
22168		<1>	;dd sret ; tty1, AX = 11 (runix)
22169		<1>	;ocvt / tty1
22170		<1>	;dd sret ; tty2, AX = 12 (runix)
22171		<1>	;ocvt / tty2
22172		<1>	;dd sret ; tty3, AX = 13 (runix)
22173		<1>	;ocvt / tty3
22174		<1>	;dd sret ; tty4, AX = 14 (runix)
22175		<1>	;ocvt / tty4
22176		<1>	;dd sret ; tty5, AX = 15 (runix)
22177		<1>	;ocvt / tty5
22178		<1>	;dd sret ; tty6, AX = 16 (runix)
22179		<1>	;ocvt / tty6
22180		<1>	;dd sret ; tty7, AX = 17 (runix)
22181		<1>	;ocvt / tty7
22182		<1>	;dd ocvt ; COM1, AX = 18 (runix only)
22183		<1>	; / ejec / lpr
22184		<1>	;dd ocvt ; COM2, AX = 19 (runix only)
22185		<1>	
22186		<1>	
22187		<1>	otty: ; / open console tty for reading or writing
22188		<1>	; 16/11/2015
22189		<1>	; 12/11/2015
22190		<1>	; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
22191		<1>	; 21/05/2013 - 13/07/2014 (Retro UNIX 8086 v1)
22192		<1>	; 16/07/2013
22193		<1>	; Retro UNIX 8086 v1 modification:
22194		<1>	; If a tty is open for read or write by
22195		<1>	; a process (u.uno), only same process can open
22196		<1>	; same tty to write or read (R->R&W or W->W&R).
22197		<1>	
22198		<1>	; (INPUT: DL=2 for Read, DL=1 for Write, DL=0 for sysstty)
22199		<1>	
22200	00005F22 0FB61D[C9740000]	<1>	movzx ebx, byte [u.uno] ; process number
22201	00005F29 8A83[D5710000]	<1>	mov al, [ebx+p.ttyc-1] ; current/console tty
22202		<1>	; 13/01/2014
22203	00005F2F EB02	<1>	jmp short ottyp
22204		<1>	ocvt:
22205	00005F31 2C0A	<1>	sub al, 10
22206		<1>	ottyp:
22207		<1>	; 16/11/2015
22208		<1>	; 12/11/2015
22209		<1>	; 18/05/2015 (32 bit modifications)
22210		<1>	; 06/12/2013 - 13/07/2014
22211	00005F33 88C6	<1>	mov dh, al ; tty number
22212	00005F35 0FB6D8	<1>	movzx ebx, al ; AL = tty number (0 to 9), AH = 0
22213	00005F38 D0E3	<1>	shl bl, 1 ; aligned to word
22214		<1>	;26/01/2014
22215	00005F3A 81C3[F4700000]	<1>	add ebx, ttyl
22216	00005F40 668B0B	<1>	mov cx, [ebx]
22217		<1>	; CL = lock value (0 or process number)
22218		<1>	; CH = open count
22219	00005F43 20C9	<1>	and cl, cl
22220		<1>	; 13/01/2014
22221	00005F45 7439	<1>	jz short otty_ret
22222		<1>	
22223		<1>	; 16/11/2015
22224	00005F47 3A0D[C9740000]	<1>	cmp cl, [u.uno]
22225	00005F4D 745C	<1>	je short ottys_3
22226		<1>	
22227	00005F4F 0FB6D9	<1>	movzx ebx, cl ; the process which has locked the tty
22228	00005F52 D0E3	<1>	shl bl, 1
22229	00005F54 668B83[74710000]	<1>	mov ax, [ebx+p.pid-2]
22230		<1>	;movzx ebx, byte [u.uno]
22231	00005F5B 8A1D[C9740000]	<1>	mov bl, [u.uno]
22232	00005F61 D0E3	<1>	shl bl, 1
22233	00005F63 663B83[94710000]	<1>	cmp ax, [ebx+p.ppid-2]
22234	00005F6A 743F	<1>	je short ottys_3 ; 16/11/2015
22235		<1>	

```

22236 <1> ; the tty is locked by another process
22237 <1> ; except the parent process (p.ppid)
22238 <1> ;
22239 00005F6C C705[CF740000]0B00- <1> mov dword [u.error], ERR_DEV_ACCESS
22240 00005F74 0000 <1>
22241 <1> ; permission denied ! error
22242 <1> otty_err: ; 13/01/2014
22243 00005F76 08D2 <1> or dl, dl ; DL = 0 -> called by sysstty
22244 00005F78 0F85FEE0FFFF <1> jnz error
22245 00005F7E F9 <1> stc
22246 00005F7F C3 <1> retn
22247 <1> otty_ret:
22248 <1> ; 13/01/2014
22249 00005F80 80FE07 <1> cmp dh, 7
22250 00005F83 761C <1> jna short ottys_2
22251 <1> ; 16/11/2015
22252 <1> com_port_check:
22253 00005F85 BE[12710000] <1> mov esi, comlp
22254 00005F8A 80FE08 <1> cmp dh, 8 ; COM1 (tty8) ?
22255 00005F8D 7601 <1> jna short ottys_1 ; yes, it is COM1
22256 00005F8F 46 <1> inc esi ; no, it is COM2 (tty9)
22257 <1> ottys_1:
22258 <1> ; 12/11/2015
22259 00005F90 803E00 <1> cmp byte [esi], 0 ; E3h (or 23h)
22260 00005F93 770C <1> ja short com_port_ready
22261 <1> ;
22262 00005F95 C705[CF740000]0F00- <1> mov dword [u.error], ERR_DEV_NOT_RDY
22263 00005F9D 0000 <1>
22264 <1> ; device not ready ! error
22265 00005F9F EBD5 <1> jmp short otty_err
22266 <1> com_port_ready:
22267 <1> ottys_2:
22268 00005FA1 08C9 <1> or cl, cl ; cl = lock/owner, ch = open count
22269 00005FA3 7506 <1> jnz short ottys_3
22270 00005FA5 8A0D[C9740000] <1> mov cl, [u.uno]
22271 <1> ottys_3:
22272 00005FAB FEC5 <1> inc ch
22273 00005FAD 66890B <1> mov [ebx], cx ; set tty lock again
22274 <1> ; 06/12/2013
22275 00005FB0 FEC6 <1> inc dh ; tty number + 1
22276 00005FB2 BB[B0740000] <1> mov ebx, u.ttyp
22277 <1> ; 13/01/2014
22278 00005FB7 F6C202 <1> test dl, 2 ; open for read sign
22279 00005FBA 7501 <1> jnz short ottys_4
22280 00005FBC 43 <1> inc ebx
22281 <1> ottys_4:
22282 <1> ; Set 'u.ttyp' ('the recent TTY') value
22283 00005FBD 8833 <1> mov [ebx], dh ; tty number + 1
22284 <1> sret:
22285 00005FBF 08D2 <1> or dl, dl ; sysstty system call check (DL=0)
22286 00005FC1 7402 <1> jz short iclose_retn
22287 00005FC3 6658 <1> pop ax
22288 <1> iclose_retn:
22289 00005FC5 C3 <1> retn
22290 <1>
22291 <1> ;
22292 <1> ; Original UNIX v1 'otty' routine:
22293 <1> ;
22294 <1> ;mov $100,$$tk$ / set interrupt enable bit (zero others) in
22295 <1> ; / reader status reg
22296 <1> ;mov $100,$$tps / set interrupt enable bit (zero others) in
22297 <1> ; / punch status reg
22298 <1> ;mov tty+[ntty*8]-8+6,r5 / r5 points to the header of the
22299 <1> ; / console tty buffer
22300 <1> ;incb (r5) / increment the count of processes that opened the
22301 <1> ; / console tty
22302 <1> ;tst u.ttyp / is there a process control tty (i.e., has a tty
22303 <1> ; / buffer header
22304 <1> ;bne sret / address been loaded into u.ttyp yet)? yes, branch
22305 <1> ;mov r5,u.ttyp / no, make the console tty the process control
22306 <1> ; / tty
22307 <1> ;br sret / ?
22308 <1> ;sret:
22309 <1> ;clr $$sps / set processor priority to zero
22310 <1> ;pop ax
22311 <1> ;mov (sp)+,r1 / pop stack to r1
22312 <1> ;3:
22313 <1> ;retn
22314 <1> ;rts r0
22315 <1>
22316 <1> ;ocvt: ; < open tty >
22317 <1> ; 13/01/2014
22318 <1> ; 06/12/2013 (major modification: p.ttyp, u.ttyp)
22319 <1> ; 24/09/2013 consistency check -> ok
22320 <1> ; 16/09/2013
22321 <1> ; 03/09/2013
22322 <1> ; 27/08/2013
22323 <1> ; 16/08/2013
22324 <1> ; 16/07/2013
22325 <1> ; 27/05/2013
22326 <1> ; 21/05/2013
22327 <1> ;

```

```

22328 <1> ; Retro UNIX 8086 v1 modification !
22329 <1> ;
22330 <1> ; In original UNIX v1, 'ocvt' routine
22331 <1> ; (exactly different than this one)
22332 <1> ; was in 'u9.s' file.
22333 <1> ;
22334 <1> ; 16/07/2013
22335 <1> ; Retro UNIX 8086 v1 modification:
22336 <1> ; If a tty is open for read or write by
22337 <1> ; a process (u.uno), only same process can open
22338 <1> ; same tty to write or read (R->R&W or W->W&R).
22339 <1> ;
22340 <1> ; INPUT: DL=2 for Read DL=1 for Write
22341 <1>
22342 <1> ; 16/09/2013
22343 <1> ; sub al, 10
22344 <1>
22345 <1> ; 06/12/2013
22346 <1> ;cmp al, 7
22347 <1> ;jna short ottyp
22348 <1> ; 13/01/2014
22349 <1> ;jmp short ottyp
22350 <1>
22351 <1>
22352 <1> ;oppt: / open paper tape for reading or writing
22353 <1> ; mov $100,$prs / set reader interrupt enable bit
22354 <1> ; tstb pptiflg / is file already open
22355 <1> ; bne 2f / yes, branch
22356 <1> ;1:
22357 <1> ; mov $240,$ps / no, set processor priority to 5
22358 <1> ; jsr r0,getc; 2 / remove all entries in clist
22359 <1> ; br .+4 / for paper tape input and place in free list
22360 <1> ; br 1b
22361 <1> ; movb $2,pptiflg / set pptiflg to indicate file just open
22362 <1> ; movb $10.,toutt+1 / place 10 in paper tape input tout entry
22363 <1> ; br sret
22364 <1> ;2:
22365 <1> ; jmp error / file already open
22366 <1>
22367 <1> iclose:
22368 <1> ; 19/05/2015
22369 <1> ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
22370 <1> ; 21/05/2013 - 13/01/2014 (Retro UNIX 8086 v1)
22371 <1> ;
22372 <1> ; close file whose i-number is in r1
22373 <1> ;
22374 <1> ; INPUTS ->
22375 <1> ; r1 - inode number
22376 <1> ; OUTPUTS ->
22377 <1> ; file's inode in core
22378 <1> ; r1 - inode number (positive)
22379 <1> ;
22380 <1> ; ((AX = R1))
22381 <1> ; ((Modified registers: -ebx-, edx))
22382 <1> ;
22383 <1> ;/ close file whose i-number is in r1
22384 00005FC6 B202 <1> mov dl, 2 ; 12/01/2014
22385 00005FC8 F6C480 <1> test ah, 80h ; Bit 15 of AX
22386 <1> ;tst r1 / test i-number
22387 <1> ;jnz short iclose_2
22388 <1> ;blt 2f / if neg., branch
22389 00005FCB 7405 <1> jz short iclose_0 ; 30/07/2013
22390 <1> ; 16/07/2013
22391 00005FCD 66F7D8 <1> neg ax ; make it positive
22392 <1> ; 12/01/2014
22393 00005FD0 FECA <1> dec dl ; dl = 1 (open for write)
22394 <1> iclose_0:
22395 00005FD2 6683F828 <1> cmp ax, 40
22396 <1> ;cmp r1,$40. / is it a special file
22397 00005FD6 77ED <1> ja short iclose_retn ; 13/01/2014
22398 <1> ;bgt 3b / no, return
22399 <1> ; 12/01/2014
22400 <1> ; DL=2 -> special file was opened for reading
22401 <1> ; DL=1 -> special file was opened for writing
22402 00005FD8 6650 <1> push ax
22403 <1> ;mov r1,-(sp) / yes, save r1 on stack
22404 00005FDA 0FB6D8 <1> movzx ebx, al
22405 00005FDD 66C1E302 <1> shl bx, 2
22406 <1> ; asl r1
22407 00005FE1 81C3[E55F0000] <1> add ebx, iclose_1 - 4
22408 00005FE7 FF23 <1> jmp dword [ebx]
22409 <1> ; jmp *1f-2(r1) / compute jump address and transfer
22410 <1> iclose_1 :
22411 00005FE9 [35600000] <1> dd cttty ; tty, AX = 1 (runix)
22412 00005FED [86600000] <1> dd cret ; mem, AX = 2 (runix)
22413 00005FF1 [86600000] <1> dd cret ; fd0, AX = 3 (runix only)
22414 00005FF5 [86600000] <1> dd cret ; fd1, AX = 4 (runix only)
22415 00005FF9 [86600000] <1> dd cret ; hd0, AX = 5 (runix only)
22416 00005FFD [86600000] <1> dd cret ; hd1, AX = 6 (runix only)
22417 00006001 [86600000] <1> dd cret ; hd2, AX = 7 (runix only)
22418 00006005 [86600000] <1> dd cret ; hd3, AX = 8 (runix only)
22419 00006009 [86600000] <1> dd cret ; lpr, AX = 9 (runix)

```

```

22420      <1> ;dd error; lpr, AX = 9 (error !)
22421      <1> ;;dd offset ejec ;;lpr, AX = 9
22422 0000600D [44600000] <1> dd ccvt ; tty0, AX = 10 (runix)
22423 00006011 [44600000] <1> dd ccvt ; tty1, AX = 11 (runix)
22424 00006015 [44600000] <1> dd ccvt ; tty2, AX = 12 (runix)
22425 00006019 [44600000] <1> dd ccvt ; tty3, AX = 13 (runix)
22426 0000601D [44600000] <1> dd ccvt ; tty4, AX = 14 (runix)
22427 00006021 [44600000] <1> dd ccvt ; tty5, AX = 15 (runix)
22428 00006025 [44600000] <1> dd ccvt ; tty6, AX = 16 (runix)
22429 00006029 [44600000] <1> dd ccvt ; tty7, AX = 17 (runix)
22430 0000602D [44600000] <1> dd ccvt ; COM1, AX = 18 (runix only)
22431 00006031 [44600000] <1> dd ccvt ; COM2, AX = 19 (runix only)
22432      <1>
22433      <1> ; 1:
22434      <1> ; cttty / tty
22435      <1> ; cppt / ppt
22436      <1> ; sret / mem
22437      <1> ; sret / rf0
22438      <1> ; sret / rk0
22439      <1> ; sret / tap0
22440      <1> ; sret / tap1
22441      <1> ; sret / tap2
22442      <1> ; sret / tap3
22443      <1> ; sret / tap4
22444      <1> ; sret / tap5
22445      <1> ; sret / tap6
22446      <1> ; sret / tap7
22447      <1> ; ccvt / tty0
22448      <1> ; ccvt / tty1
22449      <1> ; ccvt / tty2
22450      <1> ; ccvt / tty3
22451      <1> ; ccvt / tty4
22452      <1> ; ccvt / tty5
22453      <1> ; ccvt / tty6
22454      <1> ; ccvt / tty7
22455      <1> ; error / crd
22456      <1>
22457      <1> ;iclose_2: ; 2: / negative i-number
22458      <1> ;neg ax
22459      <1> ;neg r1 / make it positive
22460      <1> ;cmp ax, 40
22461      <1> ;cmp r1,$40. / is it a special file?
22462      <1> ;ja short @b
22463      <1> ;bgt 3b / no. return
22464      <1> ;push ax
22465      <1> ;mov r1,-(sp)
22466      <1> ;movzx ebx, al
22467      <1> ;shl bx, 1
22468      <1> ;asl r1 / yes. compute jump address and transfer
22469      <1> ;add ebx, iclose_3 - 2
22470      <1> ;jmp dword [ebx]
22471      <1> ;jmp *1f-2(r1) / figure out
22472      <1> ;iclose_3:
22473      <1> ;dd cttty ; tty, AX = 1 (runix)
22474      <1> ;dd sret ; mem, AX = 2 (runix)
22475      <1> ;dd sret ; fd0, AX = 3 (runix only)
22476      <1> ;dd sret ; fd1, AX = 4 (runix only)
22477      <1> ;dd sret ; hd0, AX = 5 (runix only)
22478      <1> ;dd sret ; hd1, AX = 6 (runix only)
22479      <1> ;dd sret ; hd2, AX = 7 (runix only)
22480      <1> ;dd sret ; hd3, AX = 8 (runix only)
22481      <1> ;dd sret ; lpr, AX = 9
22482      <1> ;dd ejec ; lpr, AX = 9 (runix)
22483      <1> ;dd ccvt ; tty0, AX = 10 (runix)
22484      <1> ;dd ccvt ; tty1, AX = 11 (runix)
22485      <1> ;dd ccvt ; tty2, AX = 12 (runix)
22486      <1> ;dd ccvt ; tty3, AX = 13 (runix)
22487      <1> ;dd ccvt ; tty4, AX = 14 (runix)
22488      <1> ;dd ccvt ; tty5, AX = 15 (runix)
22489      <1> ;dd ccvt ; tty6, AX = 16 (runix)
22490      <1> ;dd ccvt ; tty7, AX = 17 (runix)
22491      <1> ;dd ccvt ; COM1, AX = 18 (runix only)
22492      <1> ;dd ccvt ; COM2, AX = 19 (runix only)
22493      <1>
22494      <1> ;1:
22495      <1> ; cttty / tty
22496      <1> ; leadr / ppt
22497      <1> ; sret / mem
22498      <1> ; sret / rf0
22499      <1> ; sret / rk0
22500      <1> ; sret / tap0
22501      <1> ; sret / tap1
22502      <1> ; sret / tap2
22503      <1> ; sret / tap3
22504      <1> ; sret / tap4
22505      <1> ; sret / tap5
22506      <1> ; sret / tap6
22507      <1> ; sret / tap7
22508      <1> ; ccvt / tty0
22509      <1> ; ccvt / tty1
22510      <1> ; ccvt / tty2
22511      <1> ; ccvt / tty3

```

22512	<1> ; ccvt / tty4
22513	<1> ; ccvt / tty5
22514	<1> ; ccvt / tty6
22515	<1> ; ccvt / tty7
22516	<1> ;/ ejec / lpr
22517	<1>
22518	<1> cttty: ; / close console tty
22519	<1> ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
22520	<1> ; 21/05/2013 - 26/01/2014 (Retro UNIX 8086 v1)
22521	<1> ;
22522	<1> ; Retro UNIX 8086 v1 modification !
22523	<1> ; (DL = 2 -> it is open for reading)
22524	<1> ; (DL = 1 -> it is open for writing)
22525	<1> ; (DL = 0 -> it is open for sysstty system call)
22526	<1> ;
22527	<1> ; 06/12/2013
22528 00006035 0FB61D[C9740000]	<1> movzx ebx, byte [u.uno] ; process number
22529 0000603C 8A83[D5710000]	<1> mov al, [ebx+p.ttyc-1]
22530	<1> ; 13/01/2014
22531 00006042 EB02	<1> jmp short ctttyp
22532	<1> ccvt:
22533 00006044 2C0A	<1> sub al, 10
22534	<1> ctttyp:
22535	<1> ; 18/05/2015 (32 bit modifications)
22536	<1> ; 16/08/2013 - 26/01/2014
22537 00006046 0FB6D8	<1> movzx ebx, al ; tty number (0 to 9)
22538 00006049 D0E3	<1> shl bl, 1 ; aligned to word
22539	<1> ; 26/01/2014
22540 0000604B 81C3[F4700000]	<1> add ebx, ttyl
22541 00006051 88C6	<1> mov dh, al ; tty number
22542 00006053 668B03	<1> mov ax, [ebx]
22543	<1> ; AL = lock value (0 or process number)
22544	<1> ; AH = open count
22545 00006056 20E4	<1> and ah, ah
22546 00006058 750F	<1> jnz short cttty_ret
22547 0000605A C705[CF740000]0A00-	<1> mov dword [u.error], ERR_DEV_NOT_OPEN
22548 00006062 0000	<1>
22549	<1> ; device not open ! error
22550	<1> ; jmp short cttty_err ; open count = 0, it is not open !
22551 00006064 E913E0FFFF	<1> jmp error
22552	<1> ; 26/01/2014
22553	<1> cttty_ret:
22554 00006069 FECC	<1> dec ah ; decrease open count
22555 0000606B 7502	<1> jnz short cttty_1
22556 0000606D 30C0	<1> xor al, al ; unlock/free tty
22557	<1> cttty_1:
22558 0000606F 668903	<1> mov [ebx], ax ; close tty instance
22559	<1> ;
22560 00006072 BB[B0740000]	<1> mov ebx, u.ttyp
22561 00006077 F6C201	<1> test dl, 1 ; open for write sign
22562 0000607A 7401	<1> jz short cttty_2
22563 0000607C 43	<1> inc ebx
22564	<1> cttty_2:
22565 0000607D FEC6	<1> inc dh ; tty number + 1
22566 0000607F 3A33	<1> cmp dh, [ebx]
22567 00006081 7503	<1> jne short cret
22568	<1> ; Reset/Clear 'u.ttyp' ('the recent TTY') value
22569 00006083 C60300	<1> mov byte [ebx], 0
22570	<1> cret:
22571 00006086 08D2	<1> or dl, dl ; sysstty system call check (DL=0)
22572 00006088 7402	<1> jz short cttty_3
22573 0000608A 6658	<1> pop ax
22574	<1> cttty_3:
22575 0000608C C3	<1> retn
22576	<1>
22577	<1> ; cttty_err: ; 13/01/2014
22578	<1> ; or dl, dl ; DL = 0 -> called by sysstty
22579	<1> ; jnz error
22580	<1> ; stc
22581	<1> ; retn
22582	<1>
22583	<1>
22584	<1> ; Original UNIX v1 'ctty' routine:
22585	<1> ;
22586	<1> ; mov tty+[ntty*8]-8+6,r5
22587	<1> ; ;/ point r5 to the console tty buffer
22588	<1> ; decb (r5) / dec number of processes using console tty
22589	<1> ; br sret / return via sret
22590	<1>
22591	<1> ; ccvt: ; < close tty >
22592	<1> ; 21/05/2013 - 13/01/2014 (Retro UNIX 8086 v1)
22593	<1> ;
22594	<1> ; Retro UNIX 8086 v1 modification !
22595	<1> ;
22596	<1> ; In original UNIX v1, 'ccvt' routine
22597	<1> ; (exactly different than this one)
22598	<1> ; was in 'u9.s' file.
22599	<1> ;
22600	<1> ; DL = 2 -> it is open for reading
22601	<1> ; DL = 1 -> it is open for writing
22602	<1> ;
22603	<1> ; 17/09/2013

```

22604      <1> ;sub  al, 10
22605      <1> ;cmp  al, 7
22606      <1> ;jna  short cttyp
22607      <1> ; 13/01/2014
22608      <1> ;jmp  short cttyp
22609      <1>
22610      <1> ;cppt: / close paper tape
22611      <1> ;      clrb  pptiflg / set pptiflg to indicate file not open
22612      <1> ;1:
22613      <1> ;      mov   $240,*$ps /set process or priority to 5
22614      <1> ;      jsr   r0,getc; 2 / remove all ppt input entries from clist
22615      <1> ;      / and assign to free list
22616      <1> ;      br    sret
22617      <1> ;      br    1b
22618      <1>
22619      <1> ;ejec:
22620      <1> ;jmp  error
22621      <1> ;/ejec:
22622      <1> ;/      mov   $100,*$lps / set line printer interrupt enable bit
22623      <1> ;/      mov   $14,r1 / 'form feed' character in r1 (new page).
22624      <1> ;/      jsr   r0,lptoc / space the printer to a new page
22625      <1> ;/      br    sret / return to caller via 'sret'
22626      <1> ;include 'u8.s' ; 11/06/2015
22627      <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS8.INC
22628      <1> ; Last Modification: 24/10/2015
22629      <1> ; -----
22630      <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
22631      <1> ; (v0.1 - Beginning: 11/07/2012)
22632      <1> ;
22633      <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
22634      <1> ; (Original) Source Code by Ken Thompson (1971-1972)
22635      <1> ; <Bell Laboratories (17/3/1972)>
22636      <1> ; <Preliminary Release of UNIX Implementation Document>
22637      <1> ;
22638      <1> ; Retro UNIX 8086 v1 - U8.ASM (18/01/2014) //// UNIX v1 -> u8.s
22639      <1> ;
22640      <1> ; *****
22641      <1>
22642      <1> ;; I/O Buffer - Retro UNIX 386 v1 modification
22643      <1> ;;      (8+512 bytes, 8 bytes header, 512 bytes data)
22644      <1> ;; Word 1, byte 0 = device id
22645      <1> ;; Word 1, byte 1 = status bits (bits 8 to 15)
22646      <1> ;;      bit 9 = write bit
22647      <1> ;;      bit 10 = read bit
22648      <1> ;;      bit 12 = waiting to write bit
22649      <1> ;;      bit 13 = waiting to read bit
22650      <1> ;;      bit 15 = inhibit bit
22651      <1> ;; Word 2 (byte 2 & byte 3) = reserved (for now - 07/06/2015)
22652      <1> ;; Word 3 + Word 4 (byte 4,5,6,7) = physical block number
22653      <1> ;;      (In fact, it is 32 bit LBA for Retro UNIX 386 v1)
22654      <1> ;;
22655      <1> ;; I/O Buffer ((8+512 bytes in original Unix v1))
22656      <1> ;;      ((4+512 bytes in Retro UNIX 8086 v1))
22657      <1> ;;
22658      <1> ;; I/O Queue Entry (of original UNIX operating system v1)
22659      <1> ;; Word 1, Byte 0 = device id
22660      <1> ;; Word 1, Byte 1 = (bits 8 to 15)
22661      <1> ;;      bit 9 = write bit
22662      <1> ;;      bit 10 = read bit
22663      <1> ;;      bit 12 = waiting to write bit
22664      <1> ;;      bit 13 = waiting to read bit
22665      <1> ;;      bit 15 = inhibit bit
22666      <1> ;; Word 2 = physical block number (In fact, it is LBA for Retro UNIX 8086 v1)
22667      <1> ;;
22668      <1> ;; Original UNIX v1 ->
22669      <1> ;;      Word 3 = number of words in buffer (=256)
22670      <1> ;; Original UNIX v1 ->
22671      <1> ;;      Word 4 = bus address (addr of first word of data buffer)
22672      <1> ;;
22673      <1> ;; Retro UNIX 8086 v1 -> Buffer Header (I/O Queue Entry) size is 4 bytes !
22674      <1> ;;
22675      <1> ;; Device IDs (of Retro Unix 8086 v1)
22676      <1> ;;      0 = fd0
22677      <1> ;;      1 = fd1
22678      <1> ;;      2 = hd0
22679      <1> ;;      3 = hd1
22680      <1> ;;      4 = hd2
22681      <1> ;;      5 = hd3
22682      <1>
22683      <1> ; Retro UNIX 386 v1 - 32 bit modifications (rfd, wfd, rhd, whd) - 09/06/2015
22684      <1>
22685      <1> rfd: ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22686      <1> ; 26/04/2013
22687      <1> ; 13/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
22688
22689      <1> ;sub  ax, 3 ; zero based device number (Floppy disk)
22690
22691      <1> ;jmp  short bread ; **** returns to routine that called readi
22692
22690      <1>
22691      <1> rhd: ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22692      <1> ; 26/04/2013

```

```

22693          <1>          ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)

22694          <1>          ;sub  ax, 3 ; zero based device number (Hard disk)
22695          <1>          ;jmp  short bread ; **** returns to routine that called readi
22696          <1>
22697          <1> bread:
22698          <1>          ; 14/07/2015
22699          <1>          ; 10/07/2015
22700          <1>          ; 09/06/2015
22701          <1>          ; 07/06/2015 (Retro UNIX 386 v1 - Beginning)
22702          <1>          ; 13/03/2013 - 29/07/2013 (Retro UNIX 8086 v1)
22703          <1>          ;
22704          <1>          ; / read a block from a block structured device
22705          <1>          ;
22706          <1>          ; INPUTS ->
22707          <1>          ; [u.fopf] points to the block number
22708          <1>          ; CX = maximum block number allowed on device
22709          <1>          ; ; that was an arg to bread, in original Unix v1, but
22710          <1>          ; ; CX register is used instead of arg in Retro Unix 8086 v1
22711          <1>          ; [u.count] number of bytes to read in
22712          <1>          ; OUTPUTS ->
22713          <1>          ; [u.base] starting address of data block or blocks in user area
22714          <1>          ; [u.fopf] points to next consecutive block to be read
22715          <1>          ;
22716          <1>          ; ((Modified registers: eAX, eDX, eCX, eBX, eSI, eDI, eBP))
22717          <1>          ;
22718          <1>          ; NOTE: Original UNIX v1 has/had a defect/bug here, even if read
22719          <1>          ; byte count is less than 512, block number in *u.fopf (u.off)
22720          <1>          ; is increased by 1. For example: If user/program request
22721          <1>          ; to read 16 bytes in current block, 'sys read' increases
22722          <1>          ; the next block number just as 512 byte reading is done.
22723          <1>          ; This wrong is done in 'bread'. So, in Retro UNIX 8086 v1,
22724          <1>          ; for user (u) structure compatibility (because 16 bit is not
22725          <1>          ; enough to keep byte position/offset of the disk), this
22726          <1>          ; defect will not be corrected, user/program must request
22727          <1>          ; 512 byte read per every 'sys read' call to block devices
22728          <1>          ; for achieving correct result. In future version(s),
22729          <1>          ; this defect will be corrected by using different
22730          <1>          ; user (u) structure. 26/07/2013 - Erdogan Tan
22731          <1>
22732          <1>          ; jsr r0,tstdeve / error on special file I/O
22733          <1>          ; / (only works on tape)
22734          <1>          ; mov *u.fopf,r1 / move block number to r1
22735          <1>          ; mov $2.-cold,-(sp) / "2-cold" to stack
22736          <1> ;1:
22737          <1>          ; cmp r1,(r0) / is this block # greater than or equal to
22738          <1>          ; / maximum block # allowed on device
22739          <1>          ; jnb short @f
22740          <1>          ; bhis 1f / yes, 1f (error)
22741          <1>          ; mov r1,-(sp) / no, put block # on stack
22742          <1>          ; jsr r0,preread / read in the block into an I/O buffer
22743          <1>          ; mov (sp)+,r1 / return block # to r1
22744          <1>          ; inc r1 / bump block # to next consecutive block
22745          <1>          ; dec (sp) / "2-1-cold" on stack
22746          <1>          ; bgt 1b / 2-1-cold = 0? No, go back and read in next block
22747          <1> ;1:
22748          <1>          ; tst (sp)+ / yes, pop stack to clear off cold calculation
22749          <1>          ;push ecx ; **
22750          <1>          ;26/04/2013
22751          <1>          ;sub  ax, 3 ; 3 to 8 -> 0 to 5
22752          <1>          sub  al, 3
22753          <1>          ; AL = Retro Unix 8086 v1 disk (block device) number
22754          <1>          mov  [u.brwdev], al
22755          <1>          ; 09/06/2015
22756          <1>          movzx ebx, al
22757          <1>          mov  ecx, [ebx+drv.size] ; disk size (in sectors)
22758          <1> bread_0:
22759          <1>          push ecx ; ** ; 09/06/2015
22760          <1>          ; 10/07/2015 (Retro UNIX 386 v1 modification!)
22761          <1>          ; [u.fopf] points to byte position in disk, not sector/block !
22762          <1>          mov  ebx, [u.fopf]
22763          <1>          mov  eax, [ebx]
22764          <1>          shr  eax, 9 ; convert byte position to block/sector number
22765          <1>          ; mov *u.fopf,r1 / restore r1 to initial value of the
22766          <1>          ; / block #
22767          <1>          cmp  eax, ecx
22768          <1>          ; cmp r1,(r0)+ / block # greater than or equal to maximum
22769          <1>          ; / block number allowed
22770          <1>          ;jnb error ; 18/04/2013
22771          <1>          ; bhis error10 / yes, error
22772          <1>          jb  short bread_1
22773          <1>          mov  dword [u.error], ERR_DEV_VOL_SIZE ; 'out of volume' error
22774          <1>
22775          <1>          jmp  error
22776          <1> bread_1:
22777          <1>          ; inc dword [ebx] ; 10/07/2015 (Retro UNIX 386 v1 - modification!)
22778          <1>          ; inc *u.fopf / no, *u.fopf has next block number
22779          <1>          ; eAX = Block number (zero based)
22780          <1>          ;;jsr r0,preread / read in the block whose number is in r1
22781          <1> preread: ;; call preread
22782          <1>          mov  edi, u.brwdev ; block device number for direct I/O
22783          <1>          call bufaloc_0 ; 26/04/2013

```



```

22784 <1> ;; jc error
22785 <1> ; eBX = Buffer (Header) Address -Physical-
22786 <1> ; eAX = Block/Sector number (r1)
22787 <1> ; jsr r0,bufaloc / get a free I/O buffer (r1 has block number)
22788 <1> ; 14/03/2013
22789 000060C6 740A <1> jz short bread_2 ; Retro UNIX 8086 v1 modification
22790 <1> ; br 1f / branch if block already in a I/O buffer
22791 000060C8 66810B0004 <1> or word [ebx], 400h ; set read bit (10) in I/O Buffer
22792 <1> ; bis $2000,(r5) / set read bit (bit 10 in I/O buffer)
22793 000060CD E8B3010000 <1> call poke
22794 <1> ; jsr r0,poke / perform the read
22795 <1> ;;jc error ;2 0/07/2013
22796 <1> ; 1:
22797 <1> ; clr *$ps / ps = 0
22798 <1> ; rts r0
22799 <1> ;; return from preread
22800 <1> bread_2:
22801 000060D2 66810B0040 <1> or word [ebx], 4000h
22802 <1> ; bis $40000,(r5)
22803 <1> ; / set bit 14 of the 1st word of the I/O buffer
22804 <1> bread_3: ; 1:
22805 000060D7 66F7030024 <1> test word [ebx], 2400h
22806 <1> ; bit $22000,(r5) / are 10th and 13th bits set (read bits)
22807 000060DC 7407 <1> jz short bread_4
22808 <1> ; beq 1f / no
22809 <1> ; cmp cdev,$1 / disk or drum?
22810 <1> ; ble 2f / yes
22811 <1> ; tstb uquant / is the time quantum = 0?
22812 <1> ; bne 2f / no, 2f
22813 <1> ; mov r5,-(sp) / yes, save r5 (buffer address)
22814 <1> ; jsr r0,sleep; 31.
22815 <1> ; / put process to sleep in channel 31 (tape)
22816 <1> ; mov (sp)+,r5 / restore r5
22817 <1> ; br 1b / go back
22818 <1> ; 2: / drum or disk
22819 <1> ;; mov cx, [s.wait_]+2 ;; 29/07/2013
22820 000060DE E8AAF3FFFF <1> call idle
22821 <1> ; jsr r0,idle; s.wait+2 / wait
22822 000060E3 EBF2 <1> jmp short bread_3
22823 <1> ; br 1b
22824 <1> bread_4: ; 1: / 10th and 13th bits not set
22825 000060E5 668123FFBF <1> and word [ebx], 0BFFFh ; 101111111111111b
22826 <1> ; bic $40000,(r5) / clear bit 14
22827 <1> ; jsr r0,tstdeve / test device for error (tape)
22828 000060EA 83C308 <1> add ebx, 8
22829 <1> ; add $8,r5 / r5 points to data in I/O buffer
22830 <1> ; 09/06/2015
22831 000060ED 66833D[DF740000]00 <1> cmp word [u.pcount], 0
22832 000060F5 7705 <1> ja short bread_5
22833 000060F7 E896F9FFFF <1> call trans_addr_w ; translate virtual address to physical (w)
22834 <1> bread_5:
22835 <1> ; eBX = system (I/O) buffer address
22836 000060FC E870000000 <1> call dioreg
22837 <1> ; jsr r0,dioreg / do bookkeeping on u.count etc.
22838 <1> ; esi = start address of the transfer (in the buffer)
22839 <1> ; edi = [u.pbase], destination address in user's memory space
22840 <1> ; ecx = transfer count (in bytes)
22841 <1> ;
22842 <1> ;1: / r5 points to beginning of data in I/O buffer, r2 points to beginning
22843 <1> ; / of users data
22844 00006101 F3A4 <1> rep movsb
22845 <1> ; movb (r5)+,(r2)+ / move data from the I/O buffer
22846 <1> ; dec r3 / to the user's area in core starting at u.base
22847 <1> ; bne 1b
22848 00006103 59 <1> pop ecx ; **
22849 00006104 833D[A4740000]00 <1> cmp dword [u.count], 0
22850 <1> ; tst u.count / done
22851 0000610B 7790 <1> ja short bread_0 ; 09/06/2015
22852 <1> ; beq 1f / yes, return
22853 <1> ; tst -(r0) / no, point r0 to the argument again
22854 <1> ; br bread / read some more
22855 <1> ; 1:
22856 0000610D 58 <1> pop eax ; ****
22857 <1> ; mov (sp)+,r0
22858 0000610E C3 <1> retn ; 09/06/2015
22859 <1> ;jmp ret_
22860 <1> ;jmp ret / jump to routine that called readi
22861 <1>
22862 <1> wfd: ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22863 <1> ; 26/04/2013
22864 <1> ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)

22865 <1> ;sub ax, 3 ; zero based device number (Hard disk)
22866 <1> ;jmp short bwrite ; **** returns to routine that called writei
22867 <1>
22868 <1> whd: ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22869 <1> ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)

22870 <1> ;sub ax, 3 ; zero based device number (Hard disk)
22871 <1> ;jmp short bwrite ; **** returns to routine that called writei ('jmp ret')
22872 <1>
22873 <1> bwrite:

```

```

22874 <1> ; 14/07/2015
22875 <1> ; 10/07/2015
22876 <1> ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22877 <1> ; 14/03/2013 - 20/07/2013 (Retro UNIX 8086 v1)
22878 <1> ;
22879 <1> ;; / write on block structured device
22880 <1> ;
22881 <1> ; INPUTS ->
22882 <1> ; [u.fopf] points to the block number
22883 <1> ; CX = maximum block number allowed on device
22884 <1> ; ; that was an arg to bwrite, in original Unix v1, but
22885 <1> ; ; CX register is used instead of arg in Retro Unix 8086 v1
22886 <1> ; [u.count] number of bytes to user desires to write
22887 <1> ; OUTPUTS ->
22888 <1> ; [u.fopf] points to next consecutive block to be written into
22889 <1> ;
22890 <1> ; ((Modified registers: edx, ecx, ebx, esi, edi, ebp))
22891 <1> ;
22892 <1> ; NOTE: Original UNIX v1 has/had a defect/bug here, even if write
22893 <1> ; byte count is less than 512, block number in *u.fopf (u.off)
22894 <1> ; is increased by 1. For example: If user/program request
22895 <1> ; to write 16 bytes in current block, 'sys write' increases
22896 <1> ; the next block number just as 512 byte writing is done.
22897 <1> ; This wrong is done in 'bwrite'. So, in Retro UNIX 8086 v1,
22898 <1> ; for user (u) structure compatibility (because 16 bit is not
22899 <1> ; enough to keep byte position/offset of the disk), this
22900 <1> ; defect will not be corrected, user/program must request
22901 <1> ; 512 byte write per every 'sys write' call to block devices
22902 <1> ; for achieving correct result. In future version(s),
22903 <1> ; this defect will be corrected by using different
22904 <1> ; user (u) structure. 26/07/2013 - Erdogan Tan
22905 <1>
22906 <1> ; jsr r0,tstdev / test the device for an error
22907 <1> ;push ecx ; **
22908 <1> ;26/04/2013
22909 <1> ;sub ax, 3 ; 3 to 8 -> 0 to 5
22910 0000610F 2C03 <1> sub al, 3
22911 <1> ; AL = Retro Unix 8086 v1 disk (block device) number
22912 00006111 A2[E2740000] <1> mov [u.brwdev], al
22913 <1> ; 09/06/2015
22914 00006116 0FB6D8 <1> movzx ebx, al
22915 00006119 8B8B[086E0000] <1> mov ecx, [ebx+drv.size] ; disk size (in sectors)
22916 <1> bwrite_0:
22917 0000611F 51 <1> push ecx ; ** ; 09/06/2015
22918 <1> ; 10/07/2015 (Retro UNIX 386 v1 modification!)
22919 <1> ; [u.fopf] points to byte position in disk, not sector/block !
22920 00006120 8B1D[90740000] <1> mov ebx, [u.fopf]
22921 00006126 8B03 <1> mov eax, [ebx]
22922 00006128 C1E809 <1> shr eax, 9 ; convert byte position to block/sector number
22923 <1> ; mov *u.fopf,r1 / put the block number in r1
22924 0000612B 39C8 <1> cmp eax, ecx
22925 <1> ; cmp r1,(r0)+ / does block number exceed maximum allowable #
22926 <1> ; / block number allowed
22927 <1> ;jnb error ; 18/04/2013
22928 <1> ; bhis error10 / yes, error
22929 0000612D 720F <1> jb short bwrite_1
22930 0000612F C705[CF740000]1000- <1> mov dword [u.error], ERR_DEV_VOL_SIZE ; 'out of volume' error
22931 00006137 0000 <1>
22932 00006139 E93EDFFFFF <1> jmp error
22933 <1> bwrite_1:
22934 <1> ; inc dword [ebx] ; 10/07/2015 (Retro UNIX 386 v1 - modification!)
22935 <1> ; inc *u.fopf / no, increment block number
22936 <1> ; 09/06/2015 - 10/07/2015
22937 0000613E 66833D[DF740000]00 <1> cmp word [u.pcount], 0
22938 00006146 7705 <1> ja short bwrite_2
22939 00006148 E841F9FFFF <1> call trans_addr_r ; translate virtual address to physical (r)
22940 <1> bwrite_2:
22941 0000614D BF[E2740000] <1> mov edi, u.brwdev ; block device number for direct I/O
22942 00006152 E8C4000000 <1> call bwslot ; 26/04/2013 (wslot -> bwslot)
22943 <1> ; jsr r0,wslot / get an I/O buffer to write into
22944 <1> ; add $8,r5 / r5 points to data in I/O buffer
22945 00006157 E815000000 <1> call dioreg
22946 <1> ; jsr r0,dioreg / do the necessary bookkeeping
22947 <1> ; esi = destination address (in the buffer)
22948 <1> ; edi = [u.pbase], start address of transfer in user's memory space
22949 <1> ; ecx = transfer count (in bytes)
22950 <1> ; 1: / r2 points to the users data; r5 points to the I/O buffers data area
22951 0000615C 87F7 <1> xchg esi, edi ; 14/07/2015
22952 0000615E F3A4 <1> rep movsb
22953 <1> ; movb (r2)+,(r5)+ / ; r3, has the byte count
22954 <1> ; dec r3 / area to the I/O buffer
22955 <1> ; bne 1b
22956 00006160 E8FE000000 <1> call dskwr
22957 <1> ; jsr r0,dskwr / write it out on the device
22958 00006165 59 <1> pop ecx ; **
22959 00006166 833D[A4740000]00 <1> cmp dword [u.count], 0
22960 <1> ; tst u.count / done
22961 0000616D 77B0 <1> ja short bwrite_0 ; 09/06/2015
22962 <1> ; beq 1f / yes, 1f
22963 <1> ; tst -(r0) / no, point r0 to the argument of the call
22964 <1> ; br bwrite / go back and write next block
22965 <1> ; 1:

```

22966	0000616F	58	<1>	pop	eax ; ****
22967			<1>		; mov (sp)+,r0
22968	00006170	C3	<1>	retn	; 09/06/2015
22969			<1>		; jmp ret_
22970			<1>		; jmp ret / return to routine that called writei
22971			<1>	;error10:	
22972			<1>	; jmp	error ; / see 'error' routine
22973			<1>		
22974			<1>	dioreg:	
22975			<1>		; 14/07/2015
22976			<1>		; 10/07/2015 (UNIX v1 bugfix - [u.fofp]: byte pos., not block)
22977			<1>		; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
22978			<1>		; 14/03/2013 (Retro UNIX 8086 v1)
22979			<1>		
22980			<1>		; bookkeeping on block transfers of data
22981			<1>		
22982			<1>		; * returns value of u.pbase before it gets updated, in EDI
22983			<1>		; * returns byte count (to transfer) in ECX (<=512)
22984			<1>		; 10/07/2015
22985			<1>		; * returns byte offset from beginning of current sector buffer
22986			<1>		; (beginning of data) in ESI
22987			<1>		
22988	00006171	8B0D[A4740000]	<1>	mov	ecx, [u.count]
22989			<1>		; mov u.count,r3 / move char count to r3
22990	00006177	81F900020000	<1>	cmp	ecx, 512
22991			<1>		; cmp r3,\$512. / more than 512. char?
22992	0000617D	7605	<1>	jna	short dioreg_0
22993			<1>		; blos 1f / no, branch
22994	0000617F	B900020000	<1>	mov	ecx, 512
22995			<1>		; mov \$512.,r3 / yes, just take 512.
22996			<1>	dioreg_0:	
22997			<1>		; 09/06/2015
22998	00006184	663B0D[DF740000]	<1>	cmp	cx, [u.pcount]
22999	0000618B	7607	<1>	jna	short dioreg_1
23000	0000618D	668B0D[DF740000]	<1>	mov	cx, [u.pcount]
23001			<1>	dioreg_1:	
23002			<1>		; 1:
23003	00006194	8B15[A0740000]	<1>	mov	edx, [u.base] ; 09/06/2015 (eax -> edx)
23004			<1>		; mov u.base,r2 / put users base in r2
23005	0000619A	010D[A8740000]	<1>	add	[u.nread], ecx
23006			<1>		; add r3,u.nread / add the number to be read to u.nread
23007	000061A0	290D[A4740000]	<1>	sub	[u.count], ecx
23008			<1>		; sub r3,u.count / update count
23009	000061A6	010D[A0740000]	<1>	add	[u.base], ecx
23010			<1>		; add r3,u.base / update base
23011			<1>		; 10/07/2015
23012			<1>		; Retro UNIX 386 v1 - modification !
23013			<1>		; (File pointer points to byte position, not block/sector no.)
23014			<1>		; (It will point to next byte position instead of next block no.)
23015	000061AC	8B35[90740000]	<1>	mov	esi, [u.fofp] ; u.fopf points to byte position pointer
23016	000061B2	8B06	<1>	mov	eax, [esi] ; esi points to current byte pos. on the disk
23017	000061B4	010E	<1>	add	[esi], ecx ; ecx is added to set the next byte position
23018	000061B6	25FF010000	<1>	and	eax, 1FFh ; get offset from beginning of current block
23019	000061BB	89DE	<1>	mov	esi, ebx ; beginning of data in sector/block buffer
23020	000061BD	01C6	<1>	add	esi, eax ; esi contains start address of the transfer
23021			<1>		; 09/06/2015 - 10/07/2015
23022	000061BF	66290D[DF740000]	<1>	sub	[u.pcount], cx
23023	000061C6	81E2FF0F0000	<1>	and	edx, PAGE_OFF ; 0FFFh
23024	000061CC	8B3D[DB740000]	<1>	mov	edi, [u.pbase]
23025	000061D2	81E700F0FFFF	<1>	and	edi, ~PAGE_OFF
23026	000061D8	01D7	<1>	add	edi, edx
23027	000061DA	893D[DB740000]	<1>	mov	[u.pbase], edi
23028	000061E0	010D[DB740000]	<1>	add	[u.pbase], ecx ; 14/07/2015
23029	000061E6	C3	<1>	retn	
23030			<1>		; rts r0 / return
23031			<1>		
23032			<1>	dskrd:	
23033			<1>		; 18/08/2015
23034			<1>		; 02/07/2015
23035			<1>		; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
23036			<1>		; 14/03/2013 - 29/07/2013 (Retro UNIX 8086 v1)
23037			<1>		
23038			<1>		; 'dskrd' acquires an I/O buffer, puts in the proper
23039			<1>		; I/O queue entries (via bufaloc) then reads a block
23040			<1>		; (number specified in r1) in the acquired buffer.)
23041			<1>		; If the device is busy at the time dskrd is called,
23042			<1>		; dskrd calls idle.
23043			<1>		
23044			<1>		; INPUTS ->
23045			<1>		; r1 - block number
23046			<1>		; cdev - current device number
23047			<1>		; OUTPUTS ->
23048			<1>		; r5 - points to first data word in I/O buffer
23049			<1>		
23050			<1>		; ((AX = R1)) input/output
23051			<1>		; ((BX = R5)) output
23052			<1>		
23053			<1>		; ((Modified registers: EDX, ECX, EBX, ESI, EDI, EBP))
23054			<1>		
23055	000061E7	E830010000	<1>	call	bufaloc
23056			<1>		; jsr r0,bufaloc / shuffle off to bufaloc;
23057			<1>		; / get a free I/O buffer

```

23058 <1> ;;jc error ; 20/07/2013
23059 000061EC 741B <1> jz short dskrd_1 ; Retro UNIX 8086 v1 modification
23060 <1> ; br 1f / branch if block already in a I/O buffer
23061 <1> dskrd_0: ; 10/07/2015 (wslot)
23062 000061EE 66810B0004 <1> or word [ebx], 400h ; set read bit (10) in I/O Buffer
23063 <1> ; bis $2000,(r5) / set bit 10 of word 1 of
23064 <1> ; / I/O queue entry for buffer
23065 000061F3 E88D000000 <1> call poke
23066 <1> ; jsr r0,poke / just assigned in bufaloc,
23067 <1> ; / bit 10=1 says read
23068 <1> ; 09/06/2015
23069 000061F8 730F <1> jnc short dskrd_1
23070 <1> ;
23071 000061FA C705[CF740000]1100- <1> mov dword [u.error], ERR_DRV_READ ; disk read error !
23072 00006202 0000 <1>
23073 00006204 E973DEFFFF <1> jmp error
23074 <1> dskrd_1: ; 1:
23075 <1> ;clr *$ps
23076 00006209 66F7030024 <1> test word [ebx], 2400h
23077 <1> ; bit $22000,(r5) / if either bits 10, or 13 are 1;
23078 <1> ; / jump to idle
23079 0000620E 7407 <1> jz short dskrd_2
23080 <1> ; beq 1f
23081 <1> ;;mov ecx, [s.wait_]
23082 00006210 E878F2FFFF <1> call idle
23083 <1> ; jsr r0,idle; s.wait+2
23084 00006215 EBF2 <1> jmp short dskrd_1
23085 <1> ; br 1b
23086 <1> dskrd_2: ; 1:
23087 00006217 83C308 <1> add ebx, 8
23088 <1> ; add $8,r5 / r5 points to first word of data in block
23089 <1> ; / just read in
23090 0000621A C3 <1> retn
23091 <1> ; rts r0
23092 <1>
23093 <1> bwslot:
23094 <1> ; 10/07/2015
23095 <1> ; If the block/sector is not placed in a buffer
23096 <1> ; before 'wslot', it must be read before
23097 <1> ; it is written! (Otherwise transfer counts less
23098 <1> ; than 512 bytes will be able to destroy existing
23099 <1> ; data on disk.)
23100 <1> ;
23101 <1> ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
23102 <1> ; 26/04/2013(Retro UNIX 8086 v1)
23103 <1> ; Retro UNIX 8086 v1 modification !
23104 <1> ; ('bwslot' will be called from 'bwrite' only!)
23105 <1> ; INPUT -> eDI - points to device id (in u.brwdev)
23106 <1> ; -> eAX = block number
23107 <1> ;
23108 0000621B E80A010000 <1> call bufaloc_0
23109 00006220 742A <1> jz short wslot_0 ; block/sector already is in the buffer
23110 <1> bwslot_0:
23111 <1> ; 10/07/2015
23112 00006222 8B35[90740000] <1> mov esi, [u.fofp]
23113 00006228 8B06 <1> mov eax, [esi]
23114 0000622A 25FF010000 <1> and eax, 1FFh ; offset from beginning of the sector/block
23115 0000622F 750C <1> jnz short bwslot_1 ; it is not a full sector write
23116 <1> ; recent disk data must be placed in the buffer
23117 00006231 813D[A4740000]0002- <1> cmp dword [u.count], 512
23118 00006239 0000 <1>
23119 0000623B 730F <1> jnb short wslot_0
23120 <1> bwslot_1:
23121 0000623D E8ACFFFFFF <1> call dskrd_0
23122 00006242 83EB08 <1> sub ebx, 8 ; set ebx to the buffer header address again
23123 00006245 EB05 <1> jmp short wslot_0
23124 <1>
23125 <1> wslot:
23126 <1> ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
23127 <1> ; (32 bit modifications)
23128 <1> ; 14/03/2013 - 29/07/2013(Retro UNIX 8086 v1)
23129 <1> ;
23130 <1> ; 'wslot' calls 'bufaloc' and obtains as a result, a pointer
23131 <1> ; to the I/O queue of an I/O buffer for a block structured
23132 <1> ; device. It then checks the first word of I/O queue entry.
23133 <1> ; If bits 10 and/or 13 (read bit, waiting to read bit) are set,
23134 <1> ; wslot calls 'idle'. When 'idle' returns, or if bits 10
23135 <1> ; and/or 13 are not set, 'wslot' sets bits 9 and 15 of the first
23136 <1> ; word of the I/O queue entry (write bit, inhibit bit).
23137 <1> ;
23138 <1> ; INPUTS ->
23139 <1> ; r1 - block number
23140 <1> ; cdev - current (block/disk) device number
23141 <1> ;
23142 <1> ; OUTPUTS ->
23143 <1> ; bufp - bits 9 and 15 are set,
23144 <1> ; the remainder of the word left unchanged
23145 <1> ; r5 - points to first data word in I/O buffer
23146 <1> ;
23147 <1> ; ((AX = R1)) input/output
23148 <1> ; ((BX = R5)) output
23149 <1> ;

```

```

23150      <1>      ; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
23151      <1>
23152 00006247 E8D0000000      <1>      call  bufaloc
23153      <1>      ; 10/07/2015
23154      <1>      ; jsr r0,bufaloc / get a free I/O buffer; pointer to first
23155      <1>      ; br 1f / word in buffer in r5
23156      <1>      ; eBX = Buffer (Header) Address (r5) (ES=CS=DS, system/kernel segment)
23157      <1>      ; eAX = Block/Sector number (r1)
23158      <1> wslot_0: ;1:
23159 0000624C 66F7030024      <1>      test  word [ebx], 2400h
23160      <1>      ; bit $22000,(r5) / check bits 10, 13 (read, waiting to read)
23161      <1>      ; / of I/O queue entry
23162 00006251 7407      <1>      jz     short wslot_1
23163      <1>      ; beq 1f / branch if 10, 13 zero (i.e., not reading,
23164      <1>      ; / or not waiting to read)
23165      <1>
23166      <1>      ;; mov     ecx, [s.wait_] ; 29/07/2013
23167 00006253 E835F2FFFF      <1>      call  idle
23168      <1>      ; jsr r0,idle; / if buffer is reading or writing to read,
23169      <1>      ; / idle
23170 00006258 EBF2      <1>      jmp     short wslot_0
23171      <1>      ; br 1b / till finished
23172      <1> wslot_1: ;1:
23173 0000625A 66810B0082      <1>      or     word [ebx], 8200h
23174      <1>      ; bis $101000,(r5) / set bits 9, 15 in 1st word of I/O queue
23175      <1>      ; / (write, inhibit bits)
23176      <1>      ; clr     *$ps / clear processor status
23177 0000625F 83C308      <1>      add     ebx, 8 ; 11/06/2015
23178      <1>      ; add $8,r5 / r5 points to first word in data area
23179      <1>      ; / for this block
23180 00006262 C3      <1>      retn
23181      <1>      ; rts r0
23182      <1> dskwr:
23183      <1>      ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
23184      <1>      ; 14/03/2013 - 03/08/2013 (Retro UNIX 8086 v1)
23185      <1>      ;
23186      <1>      ; 'dskwr' writes a block out on disk, via ppoke. The only
23187      <1>      ; thing dskwr does is clear bit 15 in the first word of I/O queue
23188      <1>      ; entry pointed by 'bufp'. 'wslot' which must have been called
23189      <1>      ; previously has supplied all the information required in the
23190      <1>      ; I/O queue entry.
23191      <1>      ;
23192      <1>      ; (Modified registers: eCX, eDX, eBX, eSI, eDI)
23193      <1>      ;
23194      <1>      ;
23195 00006263 8B1D[4A740000]      <1>      mov     ebx, [bufp]
23196 00006269 668123FF7F      <1>      and     word [ebx], 7FFFh ; 011111111111111b
23197      <1>      ; bic $100000,*bufp / clear bit 15 of I/O queue entry at
23198      <1>      ; / bottom of queue
23199 0000626E E812000000      <1>      call  poke
23200      <1>      ; 09/06/2015
23201 00006273 730F      <1>      jnc     short dskwr_1
23202 00006275 C705[CF740000]1200- <1>      mov     dword [u.error], ERR_DRV_WRITE ; disk write error !
23203 0000627D 0000      <1>
23204 0000627F E9F8DDFFFF      <1>      jmp     error
23205      <1> dskwr_1:
23206 00006284 C3      <1>      retn
23207      <1>
23208      <1>
23209      <1> ;ppoke:
23210      <1>      ; mov $340,*$ps
23211      <1>      ; jsr r0,poke
23212      <1>      ; clr *$ps
23213      <1>      ; rts r0
23214      <1> poke:
23215      <1>      ; 24/10/2015
23216      <1>      ; 20/08/2015
23217      <1>      ; 18/08/2015
23218      <1>      ; 02/07/2015
23219      <1>      ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
23220      <1>      ; 15/03/2013 - 18/01/2014 (Retro UNIX 8086 v1)
23221      <1>      ;
23222      <1>      ; (NOTE: There are some disk I/O code modifications & extensions
23223      <1>      ; & exclusions on original 'poke' & other device I/O procedures of
23224      <1>      ; UNIX v1 OS for performing disk I/O functions by using IBM PC
23225      <1>      ; compatible rombios calls in Retro UNIX 8086 v1 kernel.)
23226      <1>      ;
23227      <1>      ; Basic I/O functions for all block structured devices
23228      <1>      ;
23229      <1>      ; (Modified registers: eCX, eDX, eSI, eDI)
23230      <1>      ;
23231      <1>      ; 20/07/2013 modifications
23232      <1>      ; (Retro UNIX 8086 v1 features only !)
23233      <1>      ; INPUTS ->
23234      <1>      ; (EBX = buffer header address)
23235      <1>      ; OUTPUTS ->
23236      <1>      ; cf=0 -> succeeded r/w (at least, for the caller's buffer)
23237      <1>      ; cf=1 -> error, word [eBX] = 0FFFFh
23238      <1>      ; (drive not ready or r/w error!)
23239      <1>      ; (dword [EBX+4] <> 0FFFFFFFFh indicates r/w success)
23240      <1>      ; (dword [EBX+4] = 0FFFFFFFFh means RW/IO error)
23241      <1>      ; (also it indicates invalid buffer data)

```

23242		<1>	;
23243	00006285 53	<1>	push ebx
23244		<1>	; mov r1,-(sp)
23245		<1>	; mov r2,-(sp)
23246		<1>	; mov r3,-(sp)
23247	00006286 50	<1>	push eax ; Physical Block Number (r1) (mget)
23248		<1>	;
23249		<1>	; 09/06/2015
23250		<1>	; (permit read/write after a disk R/W error)
23251	00006287 8A0B	<1>	mov cl, [ebx] ; device id (0 to 5)
23252	00006289 B001	<1>	mov al, 1
23253	0000628B D2E0	<1>	shl al, cl
23254	0000628D 8405[6A740000]	<1>	test al, [active] ; busy ? (error)
23255	00006293 7408	<1>	jz short poke_0
23256	00006295 F6D0	<1>	not al
23257	00006297 2005[6A740000]	<1>	and [active], al ; reset busy bit for this device only
23258		<1>	poke_0:
23259	0000629D BE[62740000]	<1>	mov esi, bufp + (4*(nbuf+2))
23260		<1>	; mov \$bufp+nbuf+nbuf+6,r2 / r2 points to highest priority
23261		<1>	; / I/O queue pointer
23262		<1>	poke_1: ; 1:
23263	000062A2 83EE04	<1>	sub esi, 4
23264	000062A5 8B1E	<1>	mov ebx, [esi]
23265		<1>	; mov -(r2),r1 / r1 points to an I/O queue entry
23266	000062A7 668B03	<1>	mov ax, [ebx] ; 17/07/2013
23267	000062AA F6C406	<1>	test ah, 06h
23268		<1>	;test word [ebx], 600h ; 00000110000000000b
23269		<1>	; bit \$3000,(r1) / test bits 9 and 10 of word 1 of I/O
23270		<1>	; / queue entry
23271	000062AD 745E	<1>	jz short poke_5
23272		<1>	; beq 2f / branch to 2f if both are clear
23273		<1>	; 31/07/2013
23274		<1>	;test ah, 0B0h ; (*)
23275		<1>	;test word [ebx], 0B000h ; 1011000000000000b
23276		<1>	; bit \$130000,(r1) / test bits 12, 13, and 15
23277		<1>	;jnz short poke_5 ; 31/07/2013 (*)
23278		<1>	; bne 2f / branch if any are set
23279		<1>	;movzx ecx, byte [ebx] ; 09/06/2015 ; Device Id
23280		<1>	; movb (r1),r3 / get device id
23281	000062AF 0FB6C8	<1>	movzx ecx, al ; 18/08/2015
23282		<1>	;mov edi, ecx ; 26/04/2013
23283	000062B2 31C0	<1>	xor eax, eax ; 0
23284		<1>	;cmp [edi+drv.error], al ; 0
23285		<1>	; tstb devert(r3) / test for errors on this device
23286		<1>	;jna short poke_2
23287		<1>	; beq 3f / branch if no errors
23288		<1>	; 02/07/2015
23289		<1>	;dec eax
23290		<1>	;mov [ebx+4], ax ; 0FFFFFFFh ; -1
23291		<1>	; mov \$-1,2(r1) / destroy associativity
23292		<1>	;shr eax, 24
23293		<1>	;mov [ebx], eax ; 000000FFh, reset
23294		<1>	; clrb 1(r1) / do not do I/O
23295		<1>	;jmp short poke_5
23296		<1>	; ; br 2f
23297		<1>	; ; rts r0
23298		<1>	poke_2: ; 3:
23299		<1>	; 02/07/2015
23300	000062B4 FEC1	<1>	inc cl ; 0FFh -> 0
23301	000062B6 7455	<1>	jz short poke_5
23302	000062B8 FEC0	<1>	inc al ; mov ax, 1
23303	000062BA FEC9	<1>	dec cl
23304	000062BC 7402	<1>	jz short poke_3
23305		<1>	; 26/04/2013 Modification
23306		<1>	;inc al ; mov ax, 1
23307		<1>	;or cl, cl ; Retro UNIX 8086 v1 device id.
23308		<1>	;jz short poke_3 ; cl = 0
23309	000062BE D2E0	<1>	shl al, cl ; shl ax, cl
23310		<1>	poke_3:
23311		<1>	;test [active], ax
23312	000062C0 8405[6A740000]	<1>	test [active], al
23313		<1>	; bit \$2,active / test disk busy bit
23314	000062C6 7545	<1>	jnz short poke_5
23315		<1>	; bne 2f / branch if bit is set
23316		<1>	;or [active], ax
23317	000062C8 0805[6A740000]	<1>	or [active], al
23318		<1>	; bis \$2,active / set disk busy bit
23319	000062CE 6650	<1>	push ax
23320	000062D0 E8CB000000	<1>	call diskio ; Retro UNIX 8086 v1 Only !
23321		<1>	;mov [edi+drv.error], ah
23322	000062D5 6658	<1>	pop ax
23323	000062D7 730E	<1>	jnc short poke_4 ; 20/07/2013
23324		<1>	;cmp [edi+drv.error], al ; 0
23325		<1>	;jna short poke_4
23326		<1>	; tstb devert(r3) / test for errors on this device
23327		<1>	; beq 3f / branch if no errors
23328		<1>	; 02/07/2015 (32 bit modification)
23329		<1>	; 20/07/2013
23330	000062D9 C74304FFFFFFFF	<1>	mov dword [ebx+4], 0FFFFFFFh ; -1
23331		<1>	; mov \$-1,2(r1) / destroy associativity
23332	000062E0 66C703FF00	<1>	mov word [ebx], 0FFh ; 20/08/2015
23333		<1>	; clrb 1(r1) / do not do I/O

23334	000062E5	EB26	<1>	jmp	short poke_5
23335			<1>	poke_4:;	20/07/2013
23336			<1>		; 17/07/2013
23337	000062E7	F6D0	<1>	not	al
23338	000062E9	2005[6A740000]	<1>	and	[active], al ; reset, not busy
23339			<1>		; eBX = system I/O buffer header (queue entry) address
23340			<1>	seta: ;	/ I/O queue bookkeeping; set read/write waiting bits.
23341	000062EF	668B03	<1>	mov	ax, [ebx]
23342			<1>		; mov (r1),r3 / move word 1 of I/O queue entry into r3
23343	000062F2	66250006	<1>		and ax, 600h
23344			<1>		; bic \$!3000,r3 / clear all bits except 9 and 10
23345	000062F6	668123FFF9	<1>	and	word [ebx], 0F9FFh
23346			<1>		; bic \$3000,(r1) / clear only bits 9 and 10
23347	000062FB	C0E403	<1>	shl	ah, 3
23348			<1>		; rol r3
23349			<1>		; rol r3
23350			<1>		; rol r3
23351	000062FE	660903	<1>	or	[ebx], ax
23352			<1>		; bis r3,(r1) / or old value of bits 9 and 10 with
23353			<1>		; bits 12 and 13
23354	00006301	E887F1FFFF	<1>	call	idle ; 18/01/2014
23355			<1>	;; sti	
23356			<1>	;hlt	; wait for a hardware interrupt
23357			<1>	;; cli	
23358			<1>		; NOTE: In fact, disk controller's 'disk I/O completed'
23359			<1>		; interrupt would be used to reset busy bits, but INT 13h
23360			<1>		; returns when disk I/O is completed. So, here, as temporary
23361			<1>		; method, this procedure will wait for a time according to
23362			<1>		; multi tasking and time sharing concept.
23363			<1>		;
23364			<1>		; 24/10/2015
23365			<1>	;not	ax
23366	00006306	66B8FF00	<1>	mov	ax, 0FFh ; 24/10/2015 (temporary)
23367	0000630A	662103	<1>	and	[ebx], ax ; clear bits 12 and 13
23368			<1>	poke_5: ;2:	
23369	0000630D	81FE[4A740000]	<1>	cmp	esi, bufp
23370			<1>		; cmp r2,\$bufp / test to see if entire I/O queue
23371			<1>		; / has been scanned
23372	00006313	778D	<1>	ja	short poke_1
23373			<1>		; bhi 1b
23374			<1>		; 24/03/2013
23375			<1>		; mov (sp)+,r3
23376			<1>		; mov (sp)+,r2
23377			<1>		; mov (sp)+,r1
23378	00006315	58	<1>	pop	eax ; Physical Block Number (r1) (mget)
23379	00006316	5B	<1>	pop	ebx
23380			<1>		; 02/07/2015 (32 bit modification)
23381			<1>		; 20/07/2013
23382			<1>		;cmp dword [ebx+4], 0FFFFFFFFh
23383	00006317	803BFF	<1>	cmp	byte [ebx], 0FFh ; 20/08/2015
23384			<1>		;
23385			<1>		; 'poke' returns with cf=0 if the requested buffer is read
23386			<1>		; or written succesfully; even if an error occurs while
23387			<1>		; reading to or writing from other buffers. 20/07/2013
23388			<1>		;
23389			<1>		; 09/06/2015
23390	0000631A	F5	<1>	cmc	
23391	0000631B	C3	<1>	retn	
23392			<1>		; rts r0
23393			<1>		
23394			<1>	bufaloc:	
23395			<1>		; 20/08/2015
23396			<1>		; 19/08/2015
23397			<1>		; 02/07/2015
23398			<1>		; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
23399			<1>		; (32 bit modifications)
23400			<1>		; 13/03/2013 - 29/07/2013 (Retro UNIX 8086 v1)
23401			<1>		;
23402			<1>		; bufaloc - Block device I/O buffer allocation
23403			<1>		;
23404			<1>		; INPUTS ->
23405			<1>		; r1 - block number
23406			<1>		; cdev - current (block/disk) device number
23407			<1>		; bufp+(2*n)-2 --- n = 1 ... nbuff
23408			<1>		; OUTPUTS ->
23409			<1>		; r5 - pointer to buffer allocated
23410			<1>		; bufp ... bufp+12 --- (bufp), (bufp)+2
23411			<1>		;
23412			<1>		; ((AX = R1)) input/output
23413			<1>		; ((BX = R5)) output
23414			<1>		; ((Modified registers: DX, CX, BX, SI, DI, BP))
23415			<1>		; zf=1 -> block already in a I/O buffer
23416			<1>		; zf=0 -> a new I/O buffer has been allocated
23417			<1>		; ((DL = Device ID))
23418			<1>		; (((DH = 0 or 1)))
23419			<1>		; (((CX = previous value of word ptr [bufp])))
23420			<1>		; ((CX and DH will not be used after return))
23421			<1>		
23422			<1>	;;push	esi ; ***
23423			<1>		; mov r2,-(sp) / save r2 on stack
23424			<1>		; mov \$340,*\$ps / set processor priority to 7
23425			<1>		; 20/07/2013

23426		<1> ; 26/04/2013
23427	0000631C 0FB61D[66740000]	<1> movzx ebx, byte[cdev] ; 0 or 1
23428	00006323 BF[68740000]	<1> mov edi, rdev ; offset mdev = offset rdev + 1
23429	00006328 01DF	<1> add edi, ebx
23430		<1> bufaloc_0: ; 26/04/2013 !! here is called from bread or bwrite !!
23431		<1> ;;; eDI points to device id.
23432	0000632A 0FB61F	<1> movzx ebx, byte [edi] ; [EDI] -> rdev/mdev or brwdev
23433		<1> ; 11/06/20215
23434	0000632D 80BB[246E0000]F0	<1> cmp byte [ebx+drv.status], 0F0h ; Drive not ready !
23435	00006334 720F	<1> jb short bufaloc_9
23436	00006336 C705[CF740000]0F00-	<1> mov dword [u.error], ERR_DRV_NOT_RDY
23437	0000633E 0000	<1>
23438	00006340 E937DDFFFF	<1> jmp error
23439		<1> bufaloc_9:
23440	00006345 89DA	<1> mov edx, ebx ; dh = 0, dl = device number (0 to 5)
23441		<1> bufaloc_10: ; 02/07/2015
23442	00006347 31ED	<1> xor ebp, ebp ; 0
23443	00006349 55	<1> push ebp ; 0
23444	0000634A 89E5	<1> mov ebp, esp
23445		<1> ;
23446		<1> bufaloc_1: ;1:
23447		<1> ; clr -(sp) / vacant buffer
23448	0000634C BE[4A740000]	<1> mov esi, bufp
23449		<1> ; mov \$bufp,r2 / bufp contains pointers to I/O queue
23450		<1> ; / entrys in buffer area
23451		<1> bufaloc_2: ;2:
23452	00006351 8B1E	<1> mov ebx, [esi]
23453		<1> ; mov (r2)+,r5 / move pointer to word 1 of an I/O
23454		<1> ; queue entry into r5
23455	00006353 66F70300F6	<1> test word [ebx], 0F600h
23456		<1> ; bit \$173000,(r5) / lock+keep+active+outstanding
23457	00006358 7503	<1> jnz short bufaloc_3
23458		<1> ; bne 3f / branch when
23459		<1> ; / any of bits 9,10,12,13,14,15 are set
23460		<1> ; / (i.e., buffer busy)
23461	0000635A 897500	<1> mov [ebp], esi ; pointer to I/O queue entry
23462		<1> ; mov r2,(sp) ;/ save pointer to last non-busy buffer
23463		<1> ; / found points to word 2 of I/O queue entry)
23464		<1> bufaloc_3: ;3:
23465		<1> ;mov dl, [edi] ; 26/04/2013
23466		<1> ;
23467	0000635D 3813	<1> cmp [ebx], dl
23468		<1> ; cmpb (r5),cdev / is device in I/O queue entry same
23469		<1> ; / as current device
23470	0000635F 7508	<1> jne short bufaloc_4
23471		<1> ; bne 3f
23472	00006361 394304	<1> cmp [ebx+4], eax
23473		<1> ; cmp 2(r5),r1 / is block number in I/O queue entry,
23474		<1> ; / same as current block number
23475	00006364 7503	<1> jne short bufaloc_4
23476		<1> ; bne 3f
23477		<1> ;add esp, 4
23478	00006366 59	<1> pop ecx
23479		<1> ; tst (sp)+ / bump stack pointer
23480	00006367 EB20	<1> jmp short bufaloc_7 ; Retro Unix 8086 v1 modification
23481		<1> ; jump to bufaloc_6 in original Unix v1
23482		<1> ; br 1f / use this buffer
23483		<1> bufaloc_4: ;3:
23484	00006369 83C604	<1> add esi, 4 ; 20/08/2015
23485		<1> ;
23486	0000636C 81FE[5A740000]	<1> cmp esi, bufp + (nbuf*4)
23487		<1> ; cmp r2,\$bufp+nbuf+nbuf
23488	00006372 72DD	<1> jb short bufaloc_2
23489		<1> ; blo 2b / go to 2b if r2 less than bufp+nbuf+nbuf (all
23490		<1> ; / buffers not checked)
23491	00006374 5E	<1> pop esi
23492		<1> ; mov (sp)+,r2 / once all bufs are examined move pointer
23493		<1> ; / to last free block
23494	00006375 09F6	<1> or esi, esi
23495	00006377 7507	<1> jnz short bufaloc_5
23496		<1> ; bne 2f / if (sp) is non zero, i.e.,
23497		<1> ; / if a free buffer is found branch to 2f
23498		<1> ;; mov ecx, [s.wait_]
23499	00006379 E80FF1FFFF	<1> call idle
23500		<1> ; jsr r0,idle; s.wait+2 / idle if no free buffers
23501	0000637E EBC7	<1> jmp short bufaloc_10 ; 02/07/2015
23502		<1> ; br 1b
23503		<1> bufaloc_5: ;2:
23504		<1> ; tst (r0)+ / skip if warmed over buffer
23505	00006380 FEC6	<1> inc dh ; Retro UNIX 8086 v1 modification
23506		<1> bufaloc_6: ;1:
23507	00006382 8B1E	<1> mov ebx, [esi]
23508		<1> ; mov -(r2),r5 / put pointer to word 1 of I/O queue
23509		<1> ; / entry in r5
23510		<1> ;; 26/04/2013
23511		<1> ;mov dl, [edi] ; byte [rdev] or byte [mdev]
23512	00006384 8813	<1> mov [ebx], dl
23513		<1> ; movb cdev,(r5) / put current device number
23514		<1> ; / in I/O queue entry
23515	00006386 894304	<1>



23518		<1> bufaloc_7: ;1:
23519	00006389 81FE[4A740000]	<1>     cmp esi, bufp
23520		<1>     ; cmp r2,\$bufp / bump all entrys in bufp
23521		<1>     ; / and put latest assigned
23522	0000638F 760A	<1>     jna    short bufaloc_8
23523		<1>     ; blos lf / buffer on the top
23524		<1>     ; / (this makes if the lowest priority)
23525	00006391 83EE04	<1>     sub    esi, 4
23526	00006394 8B0E	<1>     mov    ecx, [esi]
23527	00006396 894E04	<1>     mov    [esi+4], ecx
23528		<1>     ; mov -(r2),2(r2) / job for a particular device
23529	00006399 EBEE	<1>     jmp    short bufaloc_7
23530		<1>     ; br 1b
23531		<1> bufaloc_8: ;1:
23532	0000639B 891E	<1>     mov    [esi], ebx
23533		<1>     ; mov r5, (r2)
23534		<1>     ;;pop esi ; ***
23535		<1>     ; mov (sp)+,r2 / restore r2
23536	0000639D 08F6	<1>     or     dh, dh ; 0 or 1 ?
23537		<1>     ; Retro UNIX 8086 v1 modification
23538		<1>     ; zf=1 --> block already is in an I/O buffer
23539		<1>     ; zf=0 --> a new I/O buffer has been allocated
23540	0000639F C3	<1>     retn
23541		<1>     ; rts r0
23542		<1>
23543		<1> diskio:
23544		<1>     ; 10/07/2015
23545		<1>     ; 02/07/2015
23546		<1>     ; 16/06/2015
23547		<1>     ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
23548		<1>     ;         (80386 protected mode modifications)
23549		<1>     ; 15/03/2013 - 29/04/2013 (Retro UNIX 8086 v1)
23550		<1>     ;
23551		<1>     ; Retro UNIX 8086 v1 feature only !
23552		<1>     ;
23553		<1>     ; Derived from proc_chs_read procedure of TRDOS DISKIO.ASM (2011)
23554		<1>     ; 04/07/2009 - 20/07/2011
23555		<1>     ;
23556		<1>     ; NOTE: Reads only 1 block/sector (sector/block size is 512 bytes)
23557		<1>     ;
23558		<1>     ;         INPUTS ->
23559		<1>     ;         eBX = System I/O Buffer header address
23560		<1>     ;
23561		<1>     ;         OUTPUTS -> cf=0 --> done
23562		<1>     ;         cf=1 ---> error code in AH
23563		<1>     ;
23564		<1>     ; (Modified registers: eAX, eCX, eDX)
23565		<1>
23566		<1> ;rw_disk_sector:
23567		<1>     ; 10/07/2015
23568		<1>     ; 02/07/2015
23569		<1>     ; 11/06/2015 - Retro UNIX 386 v1 - 'u8.s'
23570		<1>     ; 21/02/2015 ('dsectpm.s', 'read_disk_sector')
23571		<1>     ; 16/02/2015 (Retro UNIX 386 v1 test - 'unix386.s')
23572		<1>     ; 01/12/2014 - 18/01/2015 ('dsectrm2.s')
23573		<1>     ;
23574		<1>     ;mov dx, 0201h ; Read 1 sector/block
23575	000063A0 B602	<1>     mov    dh, 2
23576	000063A2 668B03	<1>     mov    ax, [ebx]
23577		<1>     ;
23578	000063A5 56	<1>     push   esi ; ****
23579	000063A6 53	<1>     push   ebx ; ***
23580		<1>     ;
23581	000063A7 0FB6C8	<1>     movzx  ecx, al
23582	000063AA 89CE	<1>     mov    esi, ecx
23583		<1>     ;
23584	000063AC 38F1	<1>     cmp    cl, dh ; 2
23585	000063AE 7202	<1>     jb     short rwdsk0
23586	000063B0 047E	<1>     add    al, 7Eh ; 80h, 81h, 82h, 83h
23587		<1> rwdsk0:
23588	000063B2 A2[D56D0000]	<1>     mov    [drv], al
23589	000063B7 81C6[246E0000]	<1>     add    esi, drv.status
23590		<1>     ; 11/06/2015
23591	000063BD 803EF0	<1>     cmp    byte [esi], 0F0h
23592	000063C0 720F	<1>     jb     short rwdsk1
23593		<1>     ; 'drive not ready' error
23594	000063C2 C705[CF740000]0F00-	<1>     mov    dword [u.error], ERR_DRV_NOT_RDY
23595	000063CA 0000	<1>
23596	000063CC E9ABDCFFFF	<1>     jmp    error
23597		<1> rwdsk1:
23598	000063D1 F6C402	<1>     test   ah, 2
23599		<1>     ;test ax, 200h ; Bit 9 of word 0 (status word)
23600		<1>     ;         write bit
23601	000063D4 7402	<1>     jz     short rwdsk2
23602		<1>     ;test ah, 4
23603		<1>     ;;test         ax, 400h ; Bit 10 of word 0 (status word)
23604		<1>     ;         read bit
23605		<1>     ;jz     short diskio_ret
23606	000063D6 FEC6	<1>     inc    dh ; 03h = write
23607		<1> rwdsk2:
23608	000063D8 88C2	<1>     mov    dl, al
23609	000063DA 83C304	<1>     add    ebx, 4 ; sector/block address/number pointer

23610	000063DD	8B03	<1>	mov	eax, [ebx] ; sector/block number (LBA)
23611	000063DF	C0E102	<1>	shl	cl, 2
23612	000063E2	81C1[086E0000]	<1>	add	ecx, drv.size ; disk size
23613	000063E8	3B01	<1>	cmp	eax, [ecx] ; Last sector + 1 (number of secs.)
23614	000063EA	720F	<1>	jb	short rwdsk3
23615			<1>		; 'out of volume' error
23616	000063EC	C705[CF740000]1000-	<1>	mov	dword [u.error], ERR_DEV_VOL_SIZE
23617	000063F4	0000	<1>		
23618	000063F6	E981DCFFFF	<1>	jmp	error
23619			<1>		rwdsk3:
23620			<1>		; 11/06/2015
23621	000063FB	83C304	<1>	add	ebx, 4 ; buffer address
23622	000063FE	C605[FE740000]04	<1>	mov	byte [retry_count], 4
23623	00006405	F60601	<1>	test	byte [esi], 1 ; LBA ready ?
23624	00006408	7432	<1>	jz	short rwdsk_chs
23625			<1>		rwdsk_lba:
23626			<1>		; LBA read/write (with private LBA function)
23627			<1>		; ((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
23628	0000640A	83C607	<1>	add	esi, drv.error - drv.status ; 10/07/2015
23629	0000640D	89C1	<1>	mov	ecx, eax ; sector number
23630			<1>		; ebx = buffer (data) address
23631			<1>		; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
23632			<1>		rwdsk_lba_retry:
23633			<1>	mov	dl, [drv]
23634			<1>		; Function 1Bh = LBA read, 1Ch = LBA write
23635	0000640F	B419	<1>	mov	ah, 1Ch - 3h ; LBA write function number - 3
23636	00006411	00F4	<1>	add	ah, dh
23637	00006413	B001	<1>	mov	al, 1
23638			<1>		;int 13h
23639	00006415	E8E9C3FFFF	<1>	call	int13h
23640	0000641A	8826	<1>	mov	[esi], ah ; error code ; 10/07/2015
23641	0000641C	730E	<1>	jnc	short rwdsk_lba_ok
23642	0000641E	80FC80	<1>	cmp	ah, 80h ; time out ?
23643	00006421	7408	<1>	je	short rwdsk_lba_fails
23644	00006423	FE0D[FE740000]	<1>	dec	byte [retry_count]
23645	00006429	7504	<1>	jnz	short rwdsk_lba_reset ; 10/07/2015
23646			<1>		rwdsk_lba_fails:
23647	0000642B	F9	<1>	stc	
23648			<1>		rwdsk_lba_ok:
23649	0000642C	5B	<1>	pop	ebx ; ***
23650	0000642D	5E	<1>	pop	esi ; ****
23651	0000642E	C3	<1>	ret	
23652			<1>		rwdsk_lba_reset:
23653	0000642F	B40D	<1>	mov	ah, 0Dh ; Alternate reset
23654			<1>		;int 13h
23655	00006431	E8CDC3FFFF	<1>	call	int13h
23656	00006436	73D7	<1>	jnc	short rwdsk_lba_retry
23657	00006438	8826	<1>	mov	[esi], ah ; error code ; 10/07/2015
23658	0000643A	EBF0	<1>	jmp	short rwdsk_lba_ok
23659			<1>		;
23660			<1>		; CHS read (convert LBA address to CHS values)
23661			<1>		rwdsk_chs:
23662			<1>		; 10/07/2015
23663	0000643C	81EE[246E0000]	<1>	sub	esi, drv.status
23664	00006442	89F1	<1>	mov	ecx, esi
23665	00006444	81C6[2B6E0000]	<1>	add	esi, drv.error
23666			<1>		; 02/07/2015
23667			<1>		; 16/06/2015
23668			<1>		; 11/06/2015
23669	0000644A	53	<1>	push	ebx ; ** ; buffer
23670	0000644B	D1E1	<1>	shl	ecx, 1
23671	0000644D	51	<1>	push	ecx ; *
23672			<1>		;
23673	0000644E	89CB	<1>	mov	ebx, ecx
23674	00006450	8835[FD740000]	<1>	mov	[rwdsk], dh ; 02/07/2015
23675	00006456	31D2	<1>	xor	edx, edx ; 0
23676	00006458	29C9	<1>	sub	ecx, ecx
23677	0000645A	81C3[FA6D0000]	<1>	add	ebx, drv.spt
23678	00006460	668B0B	<1>	mov	cx, [ebx] ; sector per track
23679			<1>		; EDX:EAX = LBA
23680	00006463	F7F1	<1>	div	ecx
23681	00006465	88D1	<1>	mov	cl, dl ; sector number - 1
23682	00006467	FEC1	<1>	inc	cl ; sector number (1 based)
23683	00006469	5B	<1>	pop	ebx ; * ; 11/06/2015
23684	0000646A	6651	<1>	push	cx
23685	0000646C	81C3[EC6D0000]	<1>	add	ebx, drv.heads
23686	00006472	668B0B	<1>	mov	cx, [ebx] ; heads
23687	00006475	31D2	<1>	xor	edx, edx
23688			<1>		; EAX = cylinders * heads + head
23689	00006477	F7F1	<1>	div	ecx
23690	00006479	6659	<1>	pop	cx ; sector number
23691	0000647B	88D6	<1>	mov	dh, dl ; head number
23692	0000647D	8A15[D56D0000]	<1>	mov	dl, [drv]
23693	00006483	88C5	<1>	mov	ch, al ; cylinder (bits 0-7)
23694	00006485	C0E406	<1>	shl	ah, 6
23695	00006488	08E1	<1>	or	cl, ah ; cylinder (bits 8-9)
23696			<1>		; sector (bits 0-7)
23697	0000648A	5B	<1>	pop	ebx ; ** ; buffer ; 11/06/2015
23698			<1>		; CL = sector (bits 0-5)
23699			<1>		; cylinder (bits 8-9 -> bits 6-7)
23700			<1>		; CH = cylinder (bits 0-7)
23701			<1>		; DH = head

```

23702                ; DL = drive
23703                ;
23704 0000648B C605[FE740000]04    <1>  mov    byte [retry_count], 4
23705                <1>  rwdsk_retry:
23706 00006492 8A25[FD740000]    <1>  mov    ah, [rwdsk] ; 02h = read, 03h = write
23707 00006498 B001              <1>  mov    al, 1 ; sector count
23708                <1>  ;int 13h
23709 0000649A E864C3FFFF          <1>  call   int13h
23710 0000649F 8826              <1>  mov    [esi], ah ; error code ; 10/07/2015
23711 000064A1 730E              <1>  jnc    short rwdsk_ok ; ah = 0
23712 000064A3 80FC80            <1>  cmp    ah, 80h ; time out ?
23713 000064A6 7408              <1>  je     short rwdsk_fails
23714 000064A8 FE0D[FE740000]    <1>  dec    byte [retry_count]
23715 000064AE 7504              <1>  jnz    short rwdsk_reset
23716                <1>  rwdsk_fails:
23717 000064B0 F9                <1>  stc
23718                <1>  rwdsk_ok:
23719 000064B1 5B                <1>  pop    ebx ; ***
23720 000064B2 5E                <1>  pop    esi ; ****
23721 000064B3 C3                <1>  retn
23722                <1>  rwdsk_reset:
23723                <1>  ; 02/02/2015
23724 000064B4 28E4              <1>  sub    ah, ah
23725 000064B6 80FA80            <1>  cmp    dl, 80h
23726 000064B9 7202              <1>  jb     short rwdsk_fd_reset
23727 000064BB B40D              <1>  mov    ah, 0Dh ; Alternate reset
23728                <1>  rwdsk_fd_reset:
23729                <1>  ;int 13h
23730 000064BD E841C3FFFF          <1>  call   int13h
23731 000064C2 73CE              <1>  jnc    short rwdsk_retry
23732 000064C4 8826              <1>  mov    [esi], ah ; error code ; 10/07/2015
23733 000064C6 EBE9              <1>  jmp    short rwdsk_ok
23734                <1>
23735                <1>
23736                <1> ; Original UNIX v1 - drum (& disk) interrupt routine
23737                <1> ; (Equivalent to IRQ 14 & IRQ 15 disk/hardware interrupts)
23738                <1> ;
23739                <1> ; This feature is not used in Retro UNIX 386 (& 8086) for now.
23740                <1> ; Because, current Retro UNIX 386 disk I/O -INT13H- routine is
23741                <1> ; derived from IBM PC AT -infact: XT286- BIOS source code, int 13h
23742                <1> ; that uses hardware -transfer has been completed- interrupt inside it.
23743                <1> ; In a next Retro UNIX 386 version, these interrupts
23744                <1> ; (fdc_int, hdc1_int, hdc2_int) will be handled by a separate routine
23745                <1> ; as in original unix v1.
23746                <1> ; I am not removing IBM BIOS source code derivatives -compatible code-
23747                <1> ; for now, regarding the new/next 32 bit TRDOS project by me
23748                <1> ; (to keep source code files easy adaptable to 32 bit TRDOS.)
23749                <1> ;
23750                <1> ; Erdogan tan (10/07/2015)
23751                <1>
23752                <1> ;drum: / interrupt handler
23753                <1> ;      jsr      r0,setisp / save r1,r2,r3, and clockp on the stack
23754                <1> ;      jsr      r0,trapt; dcs; rfap; 1 / check for stray interrupt or
23755                <1> ;                                     / error
23756                <1> ;
23757                <1> ;      br 3f / no, error
23758                <1> ;      br 2f / error
23759                <1> ;
23760                <1> ;disk:
23761                <1> ;      jsr      r0,setisp / save r1,r2,r3, and clockp on the stack
23762                <1> ;      jmp      *$0f
23763                <1> ;0:
23764                <1> ;      jsr      r0,trapt; rkcs; rkcap; 2
23765                <1> ;      br 3f / no, errors
23766                <1> ;      mov     $115,(r2) / drive reset, errbit was set
23767                <1> ;      mov     $1f,0b-2 / next time jmp *$0f is executed jmp will be
23768                <1> ;                                     / to 1f
23769                <1> ;
23770                <1> ;      br 4f
23771                <1> ;1:
23772                <1> ;      bit     $20000,rkcs
23773                <1> ;      beq     4f / wait for seek complete
23774                <1> ;      mov     $0b,0b-2
23775                <1> ;      mov     rkcap,r1
23776                <1> ;2:
23777                <1> ;      bit     $3000,(r1) / are bits 9 or 10 set in the 1st word of
23778                <1> ;                                     / the disk buffer
23779                <1> ;      bne     3f / no, branch ignore error if outstanding
23780                <1> ;      inc     r1
23781                <1> ;      asr     (r1)
23782                <1> ;      asr     (r1)
23783                <1> ;      asr     (r1) / reissue request
23784                <1> ;      dec     r1
23785                <1> ;3:
23786                <1> ;      bic     $30000,(r1) / clear bits 12 and 13 in 1st word of buffer
23787                <1> ;      mov     ac,-(sp)
23788                <1> ;      mov     mq,-(sp) / put these on the stack
23789                <1> ;      mov     sc,-(sp)
23790                <1> ;      jsr     r0,poke
23791                <1> ;      mov     (sp)+,sc
23792                <1> ;      mov     (sp)+,mq / pop them off stack
23793                <1> ;      mov     (sp)+,ac
23794                <1> ;4:
23795                <1> ;      jmp     retisp / u4-3

```

```

23794 <1> ;
23795 <1> ;trapt: / r2 points to the
23796 <1> ; mov (r0)+,r2 / device control register
23797 <1> ; mov *(r0)+,r1 / transaction pointer points to buffer
23798 <1> ; tst (sp)+
23799 <1> ; tstb (r2) / is ready bit of dcs set?
23800 <1> ; bge 4b / device still active so branch
23801 <1> ; bit (r0),active / was device busy?
23802 <1> ; beq 4b / no, stray interrupt
23803 <1> ; bic (r0)+,active / yes, set active to zero
23804 <1> ; tst (r2) / test the err(bit is) of dcs
23805 <1> ; bge 2f / if no error jump to 2f
23806 <1> ; tst (r0)+ / skip on error
23807 <1> ; 2:
23808 <1> ; jmp (r0)
23809 %include 'u9.s' ; 29/06/2015
23810 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS9.INC
23811 <1> ; Last Modification: 09/12/2015
23812 <1> ; -----
23813 <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
23814 <1> ; (v0.1 - Beginning: 11/07/2012)
23815 <1> ;
23816 <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
23817 <1> ; (Original) Source Code by Ken Thompson (1971-1972)
23818 <1> ; <Bell Laboratories (17/3/1972)>
23819 <1> ; <Preliminary Release of UNIX Implementation Document>
23820 <1> ;
23821 <1> ; Retro UNIX 8086 v1 - U9.ASM (01/09/2014) //// UNIX v1 -> u9.s
23822 <1> ;
23823 <1> ; *****
23824 <1>
23825 <1> getch:
23826 <1> ; 30/06/2015
23827 <1> ; 18/02/2015 - Retro UNIX 386 v1 - feature only!
23828 000064C8 28C0 <1> sub al, al ; 0
23829 <1> getch_q: ; 06/08/2015
23830 000064CA 8A25[D6700000] <1> mov ah, [ptty] ; active (current) video page
23831 000064D0 EB06 <1> jmp short getc_n
23832 <1>
23833 <1> getc:
23834 <1> ; 12/11/2015
23835 <1> ; 15/09/2015
23836 <1> ; 01/07/2015
23837 <1> ; 30/06/2015
23838 <1> ; 18/02/2015 (Retro UNIX 386 v1 - Beginning)
23839 <1> ; 13/05/2013 - 04/07/2014 (Retro UNIX 8086 v1)
23840 <1> ;
23841 <1> ; Retro UNIX 8086 v1 modification !
23842 <1> ;
23843 <1> ; 'getc' gets (next) character
23844 <1> ; from requested TTY (keyboard) buffer
23845 <1> ; INPUTS ->
23846 <1> ; [u.tty] = tty number (0 to 7) (8 is COM1, 9 is COM2)
23847 <1> ; AL=0 -> Get (next) character from requested TTY buffer
23848 <1> ; (Keyboard buffer will point to
23849 <1> ; next character at next call)
23850 <1> ; AL=1 -> Test a key is available in requested TTY buffer
23851 <1> ; (Keyboard buffer will point to
23852 <1> ; current character at next call)
23853 <1> ; OUTPUTS ->
23854 <1> ; (If AL input is 1) ZF=1 -> 'empty buffer' (no chars)
23855 <1> ; ZF=0 -> AX has (current) character
23856 <1> ; AL = ascii code
23857 <1> ; AH = scan code (AH = line status for COM1 or COM2)
23858 <1> ; (cf=1 -> error code/flags in AH)
23859 <1> ; Original UNIX V1 'getc':
23860 <1> ; get a character off character list
23861 <1> ;
23862 <1> ; ((Modified registers: eAX, eBX, eCX, eDX, eSI, eDI))
23863 <1> ;
23864 <1> ; 30/06/20045 (32 bit modifications)
23865 <1> ; 16/07/2013
23866 <1> ; mov [getc_tty], ah
23867 <1> ;
23868 <1>
23869 000064D2 8A25[CE740000] <1> mov ah, [u.tty] ; 28/07/2013
23870 <1> getc_n:
23871 <1> ; 30/06/2015
23872 000064D8 08E4 <1> or ah, ah
23873 000064DA 740D <1> jz short getc0
23874 000064DC D0E4 <1> shl ah, 1
23875 000064DE 0FB6DC <1> movzx ebx, ah
23876 000064E1 81C3[D8700000] <1> add ebx, ttychr
23877 000064E7 EB05 <1> jmp short getc1
23878 <1> getc0:
23879 000064E9 BB[D8700000] <1> mov ebx, ttychr
23880 <1> getc1:
23881 000064EE 668B0B <1> mov cx, [ebx] ; ascii & scan code
23882 <1> ; (by kb_int)
23883 000064F1 6609C9 <1> or cx, cx
23884 000064F4 7508 <1> jnz short getc2
23885 000064F6 20C0 <1> and al, al

```

23886	000064F8	7416	<1>	jz	short	getc_s
23887	000064FA	6631C0	<1>	xor	ax,	ax
23888	000064FD	C3	<1>	retn		
23889			<1>	getc2:		
23890	000064FE	20C0	<1>	and	al,	al
23891	00006500	6689C8	<1>	mov	ax,	cx
23892	00006503	66B90000	<1>	mov	cx,	0
23893	00006507	7506	<1>	jnz	short	getc3
23894			<1>	getc_sn:		
23895	00006509	66890B	<1>	mov	[ebx],	cx ; 0, reset
23896	0000650C	6639C8	<1>	cmp	ax,	cx ; zf = 0
23897			<1>	getc3:		
23898	0000650F	C3	<1>	retn		
23899			<1>	getc_s:		
23900			<1>			; 12/11/2015
23901			<1>			; 15/09/2015
23902			<1>			; 01/07/2015
23903			<1>			; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
23904			<1>			; 16/07/2013 - 14/02/2014 (Retro UNIX 8086 v1)
23905			<1>			;
23906			<1>			; tty of the current process is not
23907			<1>			; current tty (ptty); so, current process only
23908			<1>			; can use keyboard input when its tty becomes
23909			<1>			; current tty (ptty).
23910			<1>			; 'sleep' is for preventing an endless lock
23911			<1>			; during this tty input request.
23912			<1>			; (Because, the user is not looking at the video page
23913			<1>			; of the process to undersand there is a keyboard
23914			<1>			; input request.)
23915			<1>			;
23916			<1>			; ((Modified registers: eAX, eBX, eCX, eDX, eSI, eDI))
23917			<1>			;
23918			<1>			; 05/10/2013
23919			<1>			; ah = byte ptr [u.ttyn] ; (tty number)
23920			<1>			;
23921			<1>			; 10/10/2013
23922			<1>	gcw0:		
23923	00006510	B10A	<1>	mov	cl,	10 ; ch = 0
23924			<1>	gcw1:		
23925			<1>			; 12/11/2015
23926	00006512	E867DCFFFF	<1>	call	intract	; jumps to 'sysexit' if [u.quit] = FFFFh
23927			<1>			; 10/10/2013
23928	00006517	E871EFFFFF	<1>	call	idle	
23929	0000651C	668B03	<1>	mov	ax,	[ebx] ; ascii & scan code
23930			<1>			; (by kb_int)
23931	0000651F	6609C0	<1>	or	ax,	ax
23932			<1>	jnz	short	gcw3
23933	00006522	7519	<1>	jnz	short	gcw2 ; 15/09/2015
23934			<1>			; 30/06/2015
23935	00006524	FEC9	<1>	dec	cl	
23936	00006526	75EA	<1>	jnz	short	gcw1
23937			<1>			;
23938	00006528	8A25[CE740000]	<1>	mov	ah,	[u.ttyn] ; 20/10/2013
23939			<1>			; ; 10/12/2013
23940			<1>	cmp	ah,	[ptty]
23941			<1>	jne	short	gcw2
23942			<1>			; ; 14/02/2014
23943			<1>	cmp	byte	[u.uno], 1
23944			<1>	jna	short	gcw0
23945			<1>	gcw2:		
23946	0000652E	E8EEEEFFFF	<1>	call	sleep	
23947			<1>			;
23948			<1>			; 20/09/2013
23949	00006533	8A25[CE740000]	<1>	mov	ah,	[u.ttyn]
23950	00006539	30C0	<1>	xor	al,	al
23951	0000653B	EB9B	<1>	jmp	short	getc_n
23952			<1>	gcw3:		
23953			<1>	gcw2:		; 15/09/2015
23954			<1>			; 10/10/2013
23955	0000653D	30C9	<1>	xor	cl,	cl
23956	0000653F	EBC8	<1>	jmp	short	getc_sn
23957			<1>			
23958			<1>	sndc:		; <Send character>
23959			<1>			;
23960			<1>			; 21/11/2015
23961			<1>			; 18/11/2015
23962			<1>			; 17/11/2015
23963			<1>			; 16/11/2015
23964			<1>			; 11/11/2015
23965			<1>			; 10/11/2015
23966			<1>			; 09/11/2015
23967			<1>			; 08/11/2015
23968			<1>			; 07/11/2015
23969			<1>			; 06/11/2015 (serial4.asm, 'sendchr')
23970			<1>			; 29/10/2015
23971			<1>			; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
23972			<1>			; 14/05/2013 - 28/07/2014 (Retro UNIX 8086 v1)
23973			<1>			;
23974			<1>			; Retro UNIX 8086 v1 feature only !
23975			<1>			;
23976			<1>			; ah = [u.ttyn]
23977			<1>			;

23978		<1>	; 30/06/2015
23979	00006541 80EC08	<1>	sub ah, 8 ; ; 0 = tty8 or 1 = tty9
23980		<1>	; 07/11/2015
23981	00006544 0FB6DC	<1>	movzx ebx, ah ; serial port index (0 or 1)
23982		<1>	sndc0:
23983		<1>	; 07/11/2015
23984	00006547 E82EF0FFFF	<1>	call isintr ; quit (ctrl+break) check
23985	0000654C 7405	<1>	jz short sndc1
23986	0000654E E82BDCFFFF	<1>	call intract ; quit (ctrl+break) check
23987		<1>	; CPU will jump to 'sysexit' if 'u.quit' = 0FFFFh (yes)
23988		<1>	sndc1:
23989		<1>	; 16/11/2015
23990	00006553 6689C1	<1>	mov cx, ax ; *** al = character (to be sent)
23991		<1>	sndcx:
23992	00006556 8A83[1A710000]	<1>	mov al, [ebx+schar] ; last sent character
23993	0000655C 8AA3[18710000]	<1>	mov ah, [ebx+rchar] ; last received character
23994		<1>	;
23995		<1>	; 17/11/2015
23996		<1>	; check 'request for response' status
23997	00006562 80BB[14710000]00	<1>	cmp byte [ebx+req_resp], 0
23998	00006569 7411	<1>	jz short query
23999		<1>	; 18/11/2015
24000	0000656B C683[14710000]00	<1>	mov byte [ebx+req_resp], 0
24001		<1>	response:
24002	00006572 FE05[17710000]	<1>	inc byte [comqr] ; query or response status
24003	00006578 B0FF	<1>	mov al, 0FFh
24004	0000657A EB14	<1>	jmp short sndc3
24005		<1>	query:
24006	0000657C 08C0	<1>	or al, al ; 0 = query (also end of text)
24007	0000657E 750E	<1>	jnz short sndc2 ; normal character
24008		<1>	;cmp ah, 0FFh ; is it responded by terminal ?
24009		<1>	;je short sndc2 ; yes, already responded
24010		<1>	; 16/11/2015
24011		<1>	; query: request for response (again)
24012	00006580 8883[18710000]	<1>	mov [ebx+rchar], al ; 0 ; reset
24013	00006586 FE05[17710000]	<1>	inc byte [comqr] ; query or response status
24014	0000658C EB02	<1>	jmp short sndc3
24015		<1>	sndc2:
24016	0000658E 88C8	<1>	mov al, cl ; *** character (to be sent)
24017		<1>	sndc3:
24018	00006590 8883[1A710000]	<1>	mov [ebx+schar], al ; current character (to be sent)
24019	00006596 88D8	<1>	mov al, bl ; 0 or 1 (serial port index)
24020		<1>	; 30/06/2015
24021	00006598 E884D5FFFF	<1>	call sp_status ; get serial port status
24022		<1>	; AL = Line status, AH = Modem status
24023		<1>	; 07/11/2015
24024	0000659D A880	<1>	test al, 80h
24025	0000659F 7504	<1>	jnz short sndc4
24026	000065A1 A820	<1>	test al, 20h ; Transmitter holding register empty ?
24027	000065A3 751D	<1>	jnz short sndc5
24028		<1>	sndc4: ; Check line status again
24029		<1>	; 16/11/2015
24030	000065A5 6651	<1>	push cx
24031	000065A7 B906000000	<1>	mov ecx, 6 ; 6*30 micro seconds (~5556 chars/second)
24032	000065AC E87BB0FFFF	<1>	call WAITF
24033	000065B1 6659	<1>	pop cx
24034		<1>	;
24035	000065B3 88D8	<1>	mov al, bl ; 0 or 1 (serial port index)
24036	000065B5 E867D5FFFF	<1>	call sp_status ; get serial port status
24037		<1>	; 16/11/2015
24038		<1>	; 09/11/2015
24039		<1>	; 08/11/2015
24040	000065BA A880	<1>	test al, 80h ; time out error
24041	000065BC 756C	<1>	jnz short sndc7
24042	000065BE A820	<1>	test al, 20h ; Transmitter holding register empty ?
24043	000065C0 7468	<1>	jz short sndc7
24044		<1>	sndc5:
24045	000065C2 8A83[1A710000]	<1>	mov al, [ebx+schar] ; character (to be sent)
24046	000065C8 66BAF803	<1>	mov dx, 3F8h ; data port (COM2)
24047	000065CC 28DE	<1>	sub dh, bl
24048	000065CE EE	<1>	out dx, al ; send on serial port
24049		<1>	; 10/11/2015
24050		<1>	; delay for 3*30 (3*(15..80)) micro seconds
24051		<1>	; (to improve text flow to the terminal)
24052		<1>	; ('diskette.inc': 'WAITF')
24053		<1>	; Uses port 61h, bit 4 to have CPU speed independent waiting.
24054		<1>	; (refresh periods = 1 per 30 microseconds on most machines)
24055	000065CF 6651	<1>	push cx
24056	000065D1 B906000000	<1>	mov ecx, 6 ; 6*30 micro seconds (~5556 chars/second)
24057	000065D6 E851B0FFFF	<1>	call WAITF
24058	000065DB 6659	<1>	pop cx
24059		<1>	;
24060	000065DD E898EFFFFFFF	<1>	call isintr ; quit (ctrl+break) check
24061	000065E2 7405	<1>	jz short sndc6
24062	000065E4 E895DBFFFF	<1>	call intract ; quit (ctrl+break) check
24063		<1>	; CPU will jump to 'sysexit' if 'u.quit' = 0FFFFh (yes)
24064		<1>	;21/11/2015
24065		<1>	sndc6:
24066		<1>	; 18/11/2015
24067		<1>	; 07/11/2015
24068	000065E9 88D8	<1>	mov al, bl ; al = 0 (tty8) or 1 (tty9)
24069	000065EB E831D5FFFF	<1>	call sp_status ; get serial port status

24070		<1>	; AL = Line status, AH = Modem status
24071	000065F0 3C80	<1>	cmp al, 80h
24072	000065F2 7336	<1>	jnb short sndc7
24073		<1>	;
24074	000065F4 803D[17710000]01	<1>	cmp byte [comqr], 1 ; 'query or response' ?
24075	000065FB 7248	<1>	jnb short sndc8 ; no, normal character
24076	000065FD 883D[17710000]	<1>	mov byte [comqr], bh ; 0 ; reset
24077		<1>	; 17/11/2015
24078	00006603 E885EEEEFFFF	<1>	call idle
24079		<1>	;
24080	00006608 38BB[1A710000]	<1>	cmp [ebx+schar], bh ; 0 ; query ?
24081	0000660E 0F877AFFFFFFFF	<1>	ja sndc2 ; response (will be followed by
24082		<1>	; a normal character)
24083		<1>	; Query request must be responded by the terminal
24084		<1>	; before sending a normal character !
24085	00006614 53	<1>	push ebx
24086	00006615 6651	<1>	push cx ; *** cl = character (to be sent)
24087	00006617 8A25[CE740000]	<1>	mov ah, [u.ttyn]
24088	0000661D E8FFEEEEFFFF	<1>	call sleep ; this process will be awakened by
24089		<1>	; received data available interrupt
24090	00006622 6659	<1>	pop cx ; *** cl = character (to be sent)
24091	00006624 5B	<1>	pop ebx
24092	00006625 E92CFFFFFF	<1>	jmp sndcx
24093		<1>	sndc7:
24094		<1>	; 16/11/2015
24095	0000662A 803D[17710000]01	<1>	cmp byte [comqr], 1 ; 'query or response' ?
24096	00006631 7213	<1>	jnb short sndc9 ; no
24097		<1>	;
24098	00006633 88BB[18710000]	<1>	mov [ebx+rchar], bh ; 0 ; reset
24099	00006639 88BB[1A710000]	<1>	mov [ebx+schar], bh ; 0 ; reset
24100		<1>	;
24101	0000663F 883D[17710000]	<1>	mov byte [comqr], bh ; 0 ; reset
24102		<1>	sndc8:
24103	00006645 F5	<1>	cmc ; jnc -> jc, jb -> jnb
24104		<1>	sndc9:
24105		<1>	; AL = Line status, AH = Modem status
24106	00006646 C3	<1>	retn
24107		<1>	
24108		<1>	putc:
24109		<1>	; 13/08/2015
24110		<1>	; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
24111		<1>	; 15/05/2013 - 27/07/2014 (Retro UNIX 8086 v1)
24112		<1>	;
24113		<1>	; Retro UNIX 8086 v1 modification !
24114		<1>	;
24115		<1>	; 'putc' puts a character
24116		<1>	; onto requested (tty) video page or
24117		<1>	; serial port
24118		<1>	; INPUTS ->
24119		<1>	; AL = ascii code of the character
24120		<1>	; AH = video page (tty) number (0 to 7)
24121		<1>	; (8 is COM1, 9 is COM2)
24122		<1>	; OUTPUTS ->
24123		<1>	; (If AL input is 1) ZF=1 -> 'empty buffer' (no chars)
24124		<1>	; ZF=0 -> AX has (current) character
24125		<1>	; cf=0 and AH = 0 -> no error
24126		<1>	; cf=1 and AH > 0 -> error (only for COM1 and COM2)
24127		<1>	;
24128		<1>	; Original UNIX V1 'putc':
24129		<1>	; put a character at the end of character list
24130		<1>	;
24131		<1>	; ((Modified registers: eAX, eBX, eCX, eDX, eSI, eDI))
24132		<1>	;
24133	00006647 80FC07	<1>	cmp ah, 7
24134	0000664A 0F87F1FEFFFF	<1>	ja sndc
24135		<1>	; 30/06/2015
24136	00006650 0FB6DC	<1>	movzx ebx, ah
24137		<1>	; 13/08/2015
24138	00006653 B407	<1>	mov ah, 07h ; black background, light gray character color
24139	00006655 E9C3AEFFFF	<1>	jmp write_tty ; 'video.inc'
24140		<1>	
24141		<1>	get_cpos:
24142		<1>	; 29/06/2015 (Retro UNIX 386 v1)
24143		<1>	; 04/12/2013 (Retro UNIX 8086 v1 - 'sysgtty')
24144		<1>	;
24145		<1>	; INPUT -> bl = video page number
24146		<1>	; RETURN -> dx = cursor position
24147		<1>	
24148	0000665A 53	<1>	push ebx
24149	0000665B 83E30F	<1>	and ebx, 0Fh ; 07h ; tty0 to tty7
24150	0000665E D0E3	<1>	shl bl, 1
24151	00006660 81C3[C6700000]	<1>	add ebx, cursor_posn
24152	00006666 668B13	<1>	mov dx, [ebx]
24153	00006669 5B	<1>	pop ebx
24154	0000666A C3	<1>	retn
24155		<1>	
24156		<1>	read_ac_current:
24157		<1>	; 29/06/2015 (Retro UNIX 386 v1)
24158		<1>	; 04/12/2013 (Retro UNIX 8086 v1 - 'sysgtty')
24159		<1>	;
24160		<1>	; INPUT -> bl = video page number

24161		<1>	; RETURN -> ax = character (al) and attribute (ah)
24162		<1>	
24163	0000666B E82EB0FFFF	<1>	call find_position ; 'video.inc'
24164		<1>	; dx = status port
24165		<1>	; esi = cursor location/address
24166	00006670 81C600800B00	<1>	add esi, 0B8000h ; 30/08/2014 (Retro UNIX 386 v1)
24167	00006676 668B06	<1>	mov ax, [esi] ; get the character and attribute
24168	00006679 C3	<1>	retn
24169		<1>	
24170		<1>	sysssleep:
24171		<1>	; 29/06/2015 - (Retro UNIX 386 v1)
24172		<1>	; 11/06/2014 - (Retro UNIX 8086 v1)
24173		<1>	;
24174		<1>	; Retro UNIX 8086 v1 feature only
24175		<1>	; (INPUT -> none)
24176		<1>	;
24177	0000667A 0FB61D[C9740000]	<1>	movzx ebx, byte [u.uno] ; process number
24178	00006681 8AA3[D5710000]	<1>	mov ah, [ebx+p.ttyc-1] ; current/console tty
24179	00006687 E895EEFFFF	<1>	call sleep
24180	0000668C E90BDAFFFF	<1>	jmp sysret
24181		<1>	
24182		<1>	vp_clr:
24183		<1>	; Reset/Clear Video Page
24184		<1>	;
24185		<1>	; 30/06/2015 - (Retro UNIX 386 v1)
24186		<1>	; 21/05/2013 - 30/10/2013(Retro UNIX 8086 v1) (U0.ASM)
24187		<1>	;
24188		<1>	; Retro UNIX 8086 v1 feature only !
24189		<1>	;
24190		<1>	; INPUTS ->
24191		<1>	; BL = video page number
24192		<1>	;
24193		<1>	; OUTPUT ->
24194		<1>	; none
24195		<1>	; ((Modified registers: eAX, BH, eCX, eDX, eSI, eDI))
24196		<1>	;
24197		<1>	; 04/12/2013
24198	00006691 28C0	<1>	sub al, al
24199		<1>	; al = 0 (clear video page)
24200		<1>	; bl = video page
24201	00006693 B407	<1>	mov ah, 07h
24202		<1>	; ah = 7 (attribute/color)
24203	00006695 6631C9	<1>	xor cx, cx ; 0, left upper column (cl) & row (cl)
24204	00006698 66BA4F18	<1>	mov dx, 184Fh ; right lower column & row (dl=24, dh=79)
24205	0000669C E829B0FFFF	<1>	call scroll_up
24206		<1>	; bl = video page
24207	000066A1 6631D2	<1>	xor dx, dx ; 0 (cursor position)
24208	000066A4 E998AFFFFF	<1>	jmp set_cpos
24209		<1>	
24210		<1>	sysmsg:
24211		<1>	; 11/11/2015
24212		<1>	; 01/07/2015 - (Retro UNIX 386 v1 feature only!)
24213		<1>	; Print user-application message on user's console tty
24214		<1>	;
24215		<1>	; Input -> EBX = Message address
24216		<1>	; ECX = Message length (max. 255)
24217		<1>	; DL = Color (IBM PC Rombios color attributes)
24218		<1>	;
24219	000066A9 81F9FF000000	<1>	cmp ecx, MAX_MSG_LEN ; 255
24220	000066AF 0F87E7D9FFFF	<1>	ja sysret ; nothing to do with big message size
24221	000066B5 08C9	<1>	or cl, cl
24222	000066B7 0F84DFD9FFFF	<1>	jz sysret
24223	000066BD 20D2	<1>	and dl, dl
24224	000066BF 7502	<1>	jnz short sysmsg0
24225	000066C1 B207	<1>	mov dl, 07h ; default color
24226		<1>	; (black background, light gray character)
24227		<1>	sysmsg0:
24228	000066C3 891D[A0740000]	<1>	mov [u.base], ebx
24229	000066C9 8815[D7700000]	<1>	mov [ccolor], dl ; color attributes
24230	000066CF 89E5	<1>	mov ebp, esp
24231	000066D1 31DB	<1>	xor ebx, ebx ; 0
24232	000066D3 891D[A8740000]	<1>	mov [u.nread], ebx ; 0
24233		<1>	;
24234	000066D9 381D[E1740000]	<1>	cmp [u.kcall], bl ; 0
24235	000066DF 7769	<1>	ja short sysmsgk ; Temporary (01/07/2015)
24236		<1>	;
24237	000066E1 890D[A4740000]	<1>	mov [u.count], ecx
24238	000066E7 41	<1>	inc ecx ; + 00h ; ASCII ZZ
24239	000066E8 29CC	<1>	sub esp, ecx
24240	000066EA 89E7	<1>	mov edi, esp
24241	000066EC 89E6	<1>	mov esi, esp
24242	000066EE 66891D[DF740000]	<1>	mov [u.pcount], bx ; reset page (phy. addr.) counter
24243		<1>	; 11/11/2015
24244	000066F5 8A25[B0740000]	<1>	mov ah, [u.ttyp] ; recent open tty
24245		<1>	; 0 = none
24246	000066FB FECC	<1>	dec ah
24247	000066FD 790C	<1>	jns short sysmsg1
24248	000066FF 8A1D[C9740000]	<1>	mov bl, [u.uno] ; process number
24249	00006705 8AA3[D5710000]	<1>	mov ah, [ebx+p.ttyc-1] ; user's (process's) console tty
24250		<1>	sysmsg1:
24251	0000670B 8825[CE740000]	<1>	mov [u.tty], ah
24252		<1>	sysmsg2:



24253	00006711	E895F5FFFF	<1>	call	cpass
24254	00006716	7416	<1>	jz	short sysmsg5
24255	00006718	AA	<1>	stosb	
24256	00006719	20C0	<1>	and	al, al
24257	0000671B	75F4	<1>	jnz	short sysmsg2
24258			<1>	sysmsg3:	
24259	0000671D	80FC07	<1>	cmp	ah, 7 ; tty number
24260	00006720	7711	<1>	ja	short sysmsg6 ; serial port
24261	00006722	E83E000000	<1>	call	print_cmsg
24262			<1>	sysmsg4:	
24263	00006727	89EC	<1>	mov	esp, ebp
24264	00006729	E96ED9FFFF	<1>	jmp	sysret
24265			<1>	sysmsg5:	
24266	0000672E	C60700	<1>	mov	byte [edi], 0
24267	00006731	EBEA	<1>	jmp	short sysmsg3
24268			<1>	sysmsg6:	
24269	00006733	8A06	<1>	mov	al, [esi]
24270	00006735	E807FEFFFF	<1>	call	sndc
24271	0000673A	72EB	<1>	jc	short sysmsg4
24272	0000673C	803E00	<1>	cmp	byte [esi], 0 ; 0 is stop character
24273	0000673F	76E6	<1>	jna	short sysmsg4
24274	00006741	46	<1>	inc	esi
24275	00006742	8A25[CE740000]	<1>	mov	ah, [u.ttyn]
24276	00006748	EBE9	<1>	jmp	short sysmsg6
24277			<1>		
24278			<1>	sysmsgk: ; Temporary (01/07/2015)	
24279			<1>	;	The message has been sent by Kernel (ASCIIZ string)
24280			<1>	;	(ECX -character count- will not be considered)
24281	0000674A	8B35[A0740000]	<1>	mov	esi, [u.base]
24282	00006750	8A25[D6700000]	<1>	mov	ah, [ptty] ; present/current screen (video page)
24283	00006756	8825[CE740000]	<1>	mov	[u.ttyn], ah
24284	0000675C	C605[E1740000]00	<1>	mov	byte [u.kcall], 0
24285	00006763	EBB8	<1>	jmp	short sysmsg3
24286			<1>		
24287			<1>		
24288			<1>	print_cmsg:	
24289			<1>	;	01/07/2015 (retro UNIX 386 v1 feature only !)
24290			<1>	;	
24291			<1>	;	print message (on user's console tty)
24292			<1>	;	with requested color
24293			<1>	;	
24294			<1>	;	INPUTS:
24295			<1>	;	esi = message address
24296			<1>	;	[u.ttyn] = tty number (0 to 7)
24297			<1>	;	[ccolor] = color attributes (IBM PC BIOS colors)
24298			<1>	;	
24299	00006765	AC	<1>	lodsb	
24300			<1>	pcmsg1:	
24301	00006766	56	<1>	push	esi
24302	00006767	0FB61D[CE740000]	<1>	movzx	ebx, byte [u.ttyn]
24303	0000676E	8A25[D7700000]	<1>	mov	ah, [ccolor]
24304	00006774	E8A4ADFFFF	<1>	call	write_tty
24305	00006779	5E	<1>	pop	esi
24306	0000677A	AC	<1>	lodsb	
24307	0000677B	20C0	<1>	and	al, al ; 0
24308	0000677D	75E7	<1>	jnz	short pcmsg1
24309	0000677F	C3	<1>	retn	
24310			<1>		
24311			<1>	sysgeterr:	
24312			<1>	;	09/12/2015
24313			<1>	;	21/09/2015 - (Retro UNIX 386 v1 feature only!)
24314			<1>	;	Get last error number or page fault count
24315			<1>	;	(for debugging)
24316			<1>	;	
24317			<1>	;	Input -> EBX = return type
24318			<1>	;	0 = last error code (which is in 'u.error')
24319			<1>	;	FFFFFFFFh = page fault count for running process
24320			<1>	;	FFFFFFFFh = total page fault count
24321			<1>	;	1 .. FFFFFFFDh = undefined
24322			<1>	;	
24323			<1>	;	Output -> EAX = last error number or page fault count
24324			<1>	;	(depending on EBX input)
24325			<1>	;	
24326	00006780	21DB	<1>	and	ebx, ebx
24327	00006782	750F	<1>	jnz	short glerr_2
24328			<1>	glerr_0:	
24329	00006784	A1[CF740000]	<1>	mov	eax, [u.error]
24330			<1>	glerr_1:	
24331	00006789	A3[80740000]	<1>	mov	[u.r0], eax
24332	0000678E	E909D9FFFF	<1>	jmp	sysret
24333			<1>	glerr_2:	
24334	00006793	43	<1>	inc	ebx ; FFFFFFFFh -> 0, FFFFFFFEh -> FFFFFFFFh
24335	00006794	74FD	<1>	jz	short glerr_2 ; page fault count for process
24336	00006796	43	<1>	inc	ebx ; FFFFFFFFh -> 0
24337	00006797	75EB	<1>	jnz	short glerr_0
24338	00006799	A1[50810000]	<1>	mov	eax, [PF_Count] ; total page fault count
24339	0000679E	EBE9	<1>	jmp	short glerr_1
24340			<1>	glerr_3:	
24341	000067A0	A1[E3740000]	<1>	mov	eax, [u.pfcount]
24342	000067A5	EBE2	<1>	jmp	short glerr_1
24343					
24344					; 07/03/2015

```

24345
24346
24347 000067A7 803D[D86D0000]00
24348 000067AE 7605
24349 000067B0 E87D000000
24350
24351 000067B5 803D[D96D0000]00
24352 000067BC 760C
24353 000067BE C605[2B6D0000]31
24354 000067C5 E868000000
24355
24356 000067CA 803D[DA6D0000]00
24357 000067D1 7654
24358 000067D3 66C705[296D0000]68-
24359 000067DB 64
24360 000067DC C605[2B6D0000]30
24361 000067E3 E84A000000
24362
24363 000067E8 803D[DB6D0000]00
24364 000067EF 7636
24365 000067F1 C605[2B6D0000]31
24366 000067F8 E835000000
24367
24368 000067FD 803D[DC6D0000]00
24369 00006804 7621
24370 00006806 C605[2B6D0000]32
24371 0000680D E820000000
24372
24373 00006812 803D[DD6D0000]00
24374 00006819 760C
24375 0000681B C605[2B6D0000]33
24376 00006822 E80B000000
24377
24378 00006827 BE[3A6D0000]
24379 0000682C E806000000
24380
24381 00006831 C3
24382
24383 00006832 BE[276D0000]
24384
24385 00006837 AC
24386 00006838 08C0
24387 0000683A 74F5
24388 0000683C 56
24389 0000683D 31DB
24390
24391 0000683F B407
24392
24393 00006841 E8D7ACFFFF
24394 00006846 5E
24395 00006847 EBEE
24396
24397 00006849 90<rept>
24398
24399
24400
24401
24402
24403
24404
24405
24406 00006850 0000000000000000
24407
24408
24409 00006858 FFFF0000009ACF00
24410
24411 00006860 FFFF00000092CF00
24412
24413 00006868 FFFB000040FACF00
24414
24415 00006870 FFFB000040F2CF00
24416
24417 00006878 6700
24418
24419
24420 0000687A 0000
24421
24422 0000687C 00
24423
24424 0000687D E9
24425 0000687E 00
24426
24427 0000687F 00
24428
24429
24430
24431
24432
24433
24434
24435
24436

; Temporary Code
display_disks:
    cmp     byte [fd0_type], 0
    jna     short ddsks1
    call    pdskm
ddsks1:
    cmp     byte [fd1_type], 0
    jna     short ddsks2
    mov     byte [dskx], '1'
    call    pdskm
ddsks2:
    cmp     byte [hd0_type], 0
    jna     short ddsks6
    mov     word [dsktype], 'hd'

    mov     byte [dskx], '0'
    call    pdskm
ddsks3:
    cmp     byte [hd1_type], 0
    jna     short ddsks6
    mov     byte [dskx], '1'
    call    pdskm
ddsks4:
    cmp     byte [hd2_type], 0
    jna     short ddsks6
    mov     byte [dskx], '2'
    call    pdskm
ddsks5:
    cmp     byte [hd3_type], 0
    jna     short ddsks6
    mov     byte [dskx], '3'
    call    pdskm
ddsk6:
    mov     esi, nextline
    call    pdskml
pdskm_ok:
    retn
pdskm:
    mov     esi, dsk_ready_msg
pdskml:
    lodsb
    or      al, al
    jz      short pdskm_ok
    push    esi
    xor     ebx, ebx ; 0
                ; Video page 0 (bl=0)
    mov     ah, 07h ; Black background,
                ; light gray forecolor
    call    write_tty
    pop     esi
    jmp     short pdskml

align 16

gdt:      ; Global Descriptor Table
; (30/07/2015, conforming cs)
; (26/03/2015)
; (24/03/2015, tss)
; (19/03/2015)
; (29/12/2013)
;
dw 0, 0, 0, 0 ; NULL descriptor
; 18/08/2014
                ; 8h kernel code segment, base = 00000000h
dw 0FFFFh, 0, 9A00h, 00CFh ; KCODE
                ; 10h kernel data segment, base = 00000000h
dw 0FFFFh, 0, 9200h, 00CFh ; KDATA
                ; 1Bh user code segment, base address = 400000h ; CORE
dw 0FBFFh, 0, 0FA40h, 00CFh ; UCODE
                ; 23h user data segment, base address = 400000h ; CORE
dw 0FBFFh, 0, 0F240h, 00CFh ; UDATA
                ; Task State Segment
dw 0067h ; Limit = 103 ; (104-1, tss size = 104 byte,
                ; no IO permission in ring 3)
gdt_tss0:
    dw 0 ; TSS base address, bits 0-15
gdt_tss1:
    db 0 ; TSS base address, bits 16-23
                ; 49h
    db 11101001b ; E9h => P=1/DPL=11/0/1/0/B/1 --> B = Task is busy (1)
    db 0 ; G/0/0/AVL/LIMIT=0000 ; (Limit bits 16-19 = 0000) (G=0, 1 byte)
gdt_tss2:
    db 0 ; TSS base address, bits 24-31

gdt_end:
;; 9Ah = 1001 1010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
                ;; Type= 1 (code)/C=0/R=1/A=0
                ; P= Present, DPL=0=ring 0, 1= user (0= system)
                ; 1= Code C= non-Conforming, R= Readable, A = Accessed
;; 92h = 1001 0010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
                ;; Type= 0 (data)/E=0/W=1/A=0

```

```

24437 ; P= Present, DPL=0=ring 0, 1= user (0= system)
24438 ; 0= Data E= Expansion direction (1= down, 0= up)
24439 ; W= Writeable, A= Accessed
24440
24441 ;; FAh = 1111 1010b (GDT byte 5) P=1/DPL=11/1/TYPE=1010,
24442 ;;; Type= 1 (code)/C=0/R=1/A=0
24443 ; P= Present, DPL=3=ring 3, 1= user (0= system)
24444 ; 1= Code C= non-Conforming, R= Readable, A = Accessed
24445
24446 ;; F2h = 1111 0010b (GDT byte 5) P=1/DPL=11/1/TYPE=0010,
24447 ;;; Type= 0 (data)/E=0/W=1/A=0
24448 ; P= Present, DPL=3=ring 3, 1= user (0= system)
24449 ; 0= Data E= Expansion direction (1= down, 0= up)
24450
24451 ;; CFh = 1100 1111b (GDT byte 6) G=1/B=1/0/AVL=0, Limit=1111b (3)
24452
24453 ;;; Limit = FFFFFFFh (=> FFFFFFFh+1= 1000000h) // bits 0-15, 48-51 //
24454 ; = 1000000h * 1000h (G=1) = 4GB
24455 ;;; Limit = FFBFFFh (=> FFBFFFh+1= FFC00h) // bits 0-15, 48-51 //
24456 ; = FFC00h * 1000h (G=1) = 4GB - 4MB
24457 ; G= Granularity (1= 4KB), B= Big (32 bit),
24458 ; AVL= Available to programmers
24459
24460 gtdtd:
24461 00006880 2F00 dw gdt_end - gdt - 1 ; Limit (size)
24462 00006882 [50680000] dd gdt ; Address of the GDT
24463
24464 ; 20/08/2014
24465 idtd:
24466 00006886 FF01 dw idt_end - idt - 1 ; Limit (size)
24467 00006888 [406E0000] dd idt ; Address of the IDT
24468
24469 Align 4
24470
24471 ; 21/08/2014
24472 ilist:
24473 ;times 32 dd cpu_except ; INT0 to INT 1Fh
24474 ;
24475 ; Exception list
24476 ; 25/08/2014
24477 0000688C [34090000] dd exc0 ; 0h, Divide-by-zero Error
24478 00006890 [3B090000] dd exc1
24479 00006894 [42090000] dd exc2
24480 00006898 [49090000] dd exc3
24481 0000689C [4D090000] dd exc4
24482 000068A0 [51090000] dd exc5
24483 000068A4 [55090000] dd exc6 ; 06h, Invalid Opcode
24484 000068A8 [59090000] dd exc7
24485 000068AC [5D090000] dd exc8
24486 000068B0 [61090000] dd exc9
24487 000068B4 [65090000] dd exc10
24488 000068B8 [69090000] dd exc11
24489 000068BC [6D090000] dd exc12
24490 000068C0 [71090000] dd exc13 ; 0Dh, General Protection Fault
24491 000068C4 [75090000] dd exc14 ; 0Eh, Page Fault
24492 000068C8 [79090000] dd exc15
24493 000068CC [7D090000] dd exc16
24494 000068D0 [81090000] dd exc17
24495 000068D4 [85090000] dd exc18
24496 000068D8 [89090000] dd exc19
24497 000068DC [8D090000] dd exc20
24498 000068E0 [91090000] dd exc21
24499 000068E4 [95090000] dd exc22
24500 000068E8 [99090000] dd exc23
24501 000068EC [9D090000] dd exc24
24502 000068F0 [A1090000] dd exc25
24503 000068F4 [A5090000] dd exc26
24504 000068F8 [A9090000] dd exc27
24505 000068FC [AD090000] dd exc28
24506 00006900 [B1090000] dd exc29
24507 00006904 [B5090000] dd exc30
24508 00006908 [B9090000] dd exc31
24509 ; Interrupt list
24510 0000690C [6A070000] dd timer_int ; INT 20h
24511 ;dd irq0
24512 00006910 [730C0000] dd keyb_int ; 27/08/2014
24513 ;dd irq1
24514 00006914 [8A080000] dd irq2
24515 ; COM2 int
24516 00006918 [8E080000] dd irq3
24517 ; COM1 int
24518 0000691C [99080000] dd irq4
24519 00006920 [A4080000] dd irq5
24520 ;DISKETTE_INT: ;06/02/2015
24521 00006924 [AE270000] dd fdc_int ; 16/02/2015, IRQ 6 handler
24522 ;dd irq6
24523 ; Default IRQ 7 handler against spurious IRQs (from master PIC)
24524 ; 25/02/2015 (source: http://wiki.osdev.org/8259\_PIC)
24525 00006928 [200C0000] dd default_irq7 ; 25/02/2015
24526 ;dd irq7
24527 ; Real Time Clock Interrupt
24528 0000692C [C30A0000] dd rtc_int ; 23/02/2015, IRQ 8 handler

```

```

24529                                     ;dd   irq8   ; INT 28h
24530 00006930 [B4080000]                dd   irq9
24531 00006934 [B8080000]                dd   irq10
24532 00006938 [BC080000]                dd   irq11
24533 0000693C [C0080000]                dd   irq12
24534 00006940 [C4080000]                dd   irq13
24535 ;HDISK_INT1:   ;06/02/2015
24536 00006944 [EA2F0000]                dd   hdc1_int   ; 21/02/2015, IRQ 14 handler
24537                                     ;dd   irq14
24538 ;HDISK_INT2:   ;06/02/2015
24539 00006948 [11300000]                dd   hdc2_int   ; 21/02/2015, IRQ 15 handler
24540                                     ;dd   irq15 ; INT 2Fh
24541                                     ; 14/08/2015
24542 0000694C [803F0000]                dd   sysent           ; INT 30h (system calls)
24543
24544                                     ;dd   ignore_int
24545 00006950 00000000                dd   0
24546
24547                                     ;;;
24548                                     ;;; 11/03/2015
24549                                     %include 'kybdata.inc'   ; KEYBOARD (BIOS) DATA
24550 <1> ; Retro UNIX 386 v1 Kernel - KYBDATA.INC
24551 <1> ; Last Modification: 11/03/2015
24552 <1> ;      (Data Section for 'KEYBOARD.INC')
24553 <1> ;
24554 <1> ; ////////// KEYBOARD DATA //////////
24555 <1>
24556 <1> ; 05/12/2014
24557 <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
24558 <1> ; 03/06/86   KEYBOARD BIOS
24559 <1>
24560 <1> ;-----
24561 <1> ; KEY IDENTIFICATION SCAN TABLES
24562 <1> ;-----
24563 <1>
24564 <1> ;----- TABLES FOR ALT CASE -----
24565 <1> ;----- ALT-INPUT-TABLE
24566 00006954 524F50514B                <1> K30:   db    82,79,80,81,75
24567 00006959 4C4D474849                <1>   db    76,77,71,72,73           ; 10 NUMBER ON KEYPAD
24568 <1> ;----- SUPER-SHIFT-TABLE
24569 0000695E 101112131415                <1>   db    16,17,18,19,20,21 ; A-Z TYPEWRITER CHARS
24570 00006964 161718191E1F                <1>   db    22,23,24,25,30,31
24571 0000696A 202122232425                <1>   db    32,33,34,35,36,37
24572 00006970 262C2D2E2F30                <1>   db    38,44,45,46,47,48
24573 00006976 3132                      <1>   db    49,50
24574 <1>
24575 <1> ;----- TABLE OF SHIFT KEYS AND MASK VALUES
24576 <1> ;----- KEY_TABLE
24577 00006978 52                      <1> _K6:   db    INS_KEY           ; INSERT KEY
24578 00006979 3A4546381D                <1>   db    CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
24579 0000697E 2A36                      <1>   db    LEFT_KEY,RIGHT_KEY
24580 <1> _K6L   equ    $_K6
24581 <1>
24582 <1> ;----- MASK_TABLE
24583 00006980 80                      <1> _K7:   db    INS_SHIFT           ; INSERT MODE SHIFT
24584 00006981 4020100804                <1>   db    CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
24585 00006986 0201                      <1>   db    LEFT_SHIFT,RIGHT_SHIFT
24586 <1>
24587 <1> ;----- TABLES FOR CTRL CASE ----- ;---- CHARACTERS -----
24588 00006988 1BFF00FFFFFF                <1> _K8:   db    27,-1,0,-1,-1,-1 ; Esc, 1, 2, 3, 4, 5
24589 0000698E 1EFFFFFFF1F                <1>   db    30,-1,-1,-1,-1,31 ; 6, 7, 8, 9, 0, -
24590 00006994 FF7FFF111705                <1>   db    -1,127,-1,17,23,5 ; =, Bksp, Tab, Q, W, E
24591 0000699A 12141915090F                <1>   db    18,20,25,21,9,15 ; R, T, Y, U, I, O
24592 000069A0 101B1D0AFF01                <1>   db    16,27,29,10,-1,1 ; P, [, ], Enter, Ctrl, A
24593 000069A6 13040607080A                <1>   db    19,4,6,7,8,10           ; S, D, F, G, H, J
24594 000069AC 0B0CFFFFFFF                <1>   db    11,12,-1,-1,-1,-1 ; K, L, :, ', ` , LShift
24595 000069B2 1C1A18031602                <1>   db    28,26,24,3,22,2           ; Bkslash, Z, X, C, V, B
24596 000069B8 0E0DFFFFFFF                <1>   db    14,13,-1,-1,-1,-1 ; N, M, ,, ., / , RShift
24597 000069BE 96FF20FF                <1>   db    150,-1,' ', -1           ; *, ALT, Spc, CL
24598 <1> ; ;----- FUNCTIONS -----
24599 000069C2 5E5F60616263                <1>   db    94,95,96,97,98,99 ; F1 - F6
24600 000069C8 64656667FFFF                <1>   db    100,101,102,103,-1,-1 ; F7 - F10, NL, SL
24601 000069CE 778D848E738F                <1>   db    119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
24602 000069D4 749075917692                <1>   db    116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
24603 000069DA 93FFFFFFF898A                <1>   db    147,-1,-1,-1,137,138 ; Del, SysReq, Undef, WT, F11, F12
24604 <1>
24605 <1> ;----- TABLES FOR LOWER CASE -----
24606 000069E0 1B3132333435363738- <1> K10:   db    27,'1234567890-=' ,8,9
24607 000069E9 39302D3D0809                <1>
24608 000069EF 71776572747975696F- <1>   db    'qwertyuiop[]',13,-1,'asdfghjkl;',39
24609 000069F8 705B5D0DFF61736466- <1>
24610 00006A01 67686A6B6C3B27                <1>
24611 00006A08 60FF5C7A786376626E- <1>   db    96,-1,92,'zxcvbnm,./',-1,'*','-1,' ', -1
24612 00006A11 6D2C2E2FFF2AFF20FF                <1>
24613 <1> ;----- LC TABLE SCAN
24614 00006A1A 3B3C3D3E3F                <1>   db    59,60,61,62,63           ; BASE STATE OF F1 - F10
24615 00006A1F 4041424344                <1>   db    64,65,66,67,68
24616 00006A24 FFFF                      <1>   db    -1,-1           ; NL, SL
24617 <1>
24618 <1> ;----- KEYPAD TABLE
24619 00006A26 474849FF4BFF                <1> K15:   db    71,72,73,-1,75,-1 ; BASE STATE OF KEYPAD KEYS
24620 00006A2C 4DFF4F50515253                <1>   db    77,-1,79,80,81,82,83

```

```

24621 00006A33 FFFF5C8586      <1>  db    -1,-1,92,133,134  ; SysRq, Undef, WT, F11, F12
24622                                     <1>
24623                                     <1> ;----- TABLES FOR UPPER CASE -----
24624 00006A38 1B21402324255E262A- <1> K11:   db    27,'!@#$',94,'&*()_+',8,0
24625 00006A41 28295F2B0800      <1>
24626 00006A47 51574552545955494F- <1>  db    'QWERTYUIOP{}',13,-1,'ASDFGHJKL:''
24627 00006A50 507B7D0DFF41534446- <1>
24628 00006A59 47484A4B4C3A22      <1>
24629 00006A60 7EFF7C5A584356424E- <1>  db    126,-1,'|ZXCVCBNM<>?',-1,'*',-1,' ',-1
24630 00006A69 4D3C3E3FFF2AFF20FF <1>
24631                                     <1> ;----- UC TABLE SCAN
24632 00006A72 5455565758      <1> K12:   db    84,85,86,87,88          ; SHIFTED STATE OF F1 - F10
24633 00006A77 595A5B5C5D      <1>  db    89,90,91,92,93
24634 00006A7C FFFF              <1>  db    -1,-1          ; NL, SL
24635                                     <1>
24636                                     <1> ;----- NUM STATE TABLE
24637 00006A7E 3738392D3435362B31- <1> K14:   db    '789-456+1230.'          ; NUMLOCK STATE OF KEYPAD KEYS
24638 00006A87 3233302E      <1>
24639                                     <1> ;
24640 00006A8B FFFF7C8788      <1>  db    -1,-1,124,135,136 ; SysRq, Undef, WT, F11, F12
24641                                     <1>
24642                                     <1> Align 4
24643                                     <1> ;-----
24644                                     <1> ; VIDEO DISPLAY DATA AREA          ;
24645                                     <1> ;-----
24646 00006A90 03      <1> CRT_MODE      db    3          ; CURRENT DISPLAY MODE (TYPE)
24647 00006A91 29      <1> CRT_MODE_SET db    29h          ; CURRENT SETTING OF THE 3X8 REGISTER
24648                                     <1> ; (29h default setting for video mode 3)
24649                                     <1> ; Mode Select register Bits
24650                                     <1> ; BIT 0 - 80x25 (1), 40x25 (0)
24651                                     <1> ; BIT 1 - ALPHA (0), 320x200 GRAPHICS (1)
24652                                     <1> ; BIT 2 - COLOR (0), BW (1)
24653                                     <1> ; BIT 3 - Video Sig. ENABLE (1), DISABLE (0)
24654                                     <1> ; BIT 4 - 640x200 B&W Graphics Mode (1)
24655                                     <1> ; BIT 5 - ALPHA mode BLINKING (1)
24656                                     <1> ; BIT 6, 7 - Not Used
24657                                     <1>
24658                                     <1> ; Mode 0 - 2Ch = 101100b ; 40x25 text, 16 gray colors
24659                                     <1> ; Mode 1 - 28h = 101000b ; 40x25 text, 16 fore colors, 8 back colors
24660                                     <1> ; Mode 2 - 2Dh = 101101b ; 80x25 text, 16 gray colors
24661                                     <1> ; Mode 3 - 29h = 101001b ; 80x25 text, 16 fore color, 8 back color
24662                                     <1> ; Mode 4 - 2Ah = 101010b ; 320x200 graphics, 4 colors
24663                                     <1> ; Mode 5 - 2Eh = 101110b ; 320x200 graphics, 4 gray colors
24664                                     <1> ; Mode 6 - 1Eh = 011110b ; 640x200 graphics, 2 colors
24665                                     <1> ; Mode 7 - 29h = 101001b ; 80x25 text, black & white colors
24666                                     <1> ; Mode & 37h = Video signal OFF
24667                                     <1>
24668                                     <1>
24669                                     <1> ; 26/08/2014
24670                                     <1> ; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)
24671                                     <1> ; Derived from IBM "pc-at"
24672                                     <1> ; rombios source code (06/10/1985)
24673                                     <1> ; 'dseg.inc'
24674                                     <1>
24675                                     <1> ;-----;
24676                                     <1> ; SYSTEM DATA AREA          ;
24677                                     <1> ;-----
24678 00006A92 00      <1> BIOS_BREAK      db    0          ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
24679                                     <1>
24680                                     <1> ;-----
24681                                     <1> ; KEYBOARD DATA AREAS          ;
24682                                     <1> ;-----
24683                                     <1>
24684 00006A93 00      <1> KB_FLAG      db    0          ; KEYBOARD SHIFT STATE AND STATUS FLAGS
24685 00006A94 00      <1> KB_FLAG_1      db    0          ; SECOND BYTE OF KEYBOARD STATUS
24686 00006A95 00      <1> KB_FLAG_2      db    0          ; KEYBOARD LED FLAGS
24687 00006A96 00      <1> KB_FLAG_3      db    0          ; KEYBOARD MODE STATE AND TYPE FLAGS
24688 00006A97 00      <1> ALT_INPUT      db    0          ; STORAGE FOR ALTERNATE KEY PAD ENTRY
24689 00006A98 [A86A0000] <1> BUFFER_START dd    KB_BUFFER ; OFFSET OF KEYBOARD BUFFER START
24690 00006A9C [C86A0000] <1> BUFFER_END      dd    KB_BUFFER + 32 ; OFFSET OF END OF BUFFER
24691 00006AA0 [A86A0000] <1> BUFFER_HEAD      dd    KB_BUFFER ; POINTER TO HEAD OF KEYBOARD BUFFER
24692 00006AA4 [A86A0000] <1> BUFFER_TAIL      dd    KB_BUFFER ; POINTER TO TAIL OF KEYBOARD BUFFER
24693                                     <1> ; ----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
24694 00006AA8 0000<rept> <1> KB_BUFFER      times 16 dw 0          ; ROOM FOR 16 SCAN CODE ENTRIES
24695                                     <1>
24696                                     <1> ; /// End Of KEYBOARD DATA ///
24697                                     %include 'vidata.inc' ; VIDEO (BIOS) DATA
24698                                     <1> ; Retro UNIX 386 v1 Kernel - VIDATA.INC
24699                                     <1> ; Last Modification: 11/03/2015
24700                                     <1> ; (Data section for 'VIDEO.INC')
24701                                     <1> ;
24702                                     <1> ; ////////// VIDEO DATA //////////
24703                                     <1>
24704                                     <1> video_params:
24705                                     <1> ; 02/09/2014 (Retro UNIX 386 v1)
24706                                     <1> ; ORGS.ASM ----- 06/10/85 COMPATIBILITY MODULE
24707                                     <1> ; VIDEO MODE 3
24708 00006AC8 71505A0A1F0619 <1>  db    71h,50h,5Ah,0Ah,1Fh,6,19h          ; SET UP FOR 80X25
24709 00006ACF 1C02070607 <1>  db    1Ch,2,7,6,7 ; cursor start = 6, cursor stop = 7
24710 00006AD4 00000000 <1>  db    0,0,0,0
24711                                     <1>
24712                                     <1> ; /// End Of VIDEO DATA ///

```

```

24713          %include 'diskdata.inc' ; DISK (BIOS) DATA (initialized)
24714 <1> ; Retro UNIX 386 v1 Kernel - DISKDATA.INC
24715 <1> ; Last Modification: 11/03/2015
24716 <1> ; (Initialized Disk Parameters Data section for 'DISKIO.INC')
24717 <1> ;
24718 <1> ; *****
24719 <1>
24720 <1> ;-----
24721 <1> ; 80286 INTERRUPT LOCATIONS      :
24722 <1> ; REFERENCED BY POST & BIOS      :
24723 <1> ;-----
24724 <1>
24725 00006AD8 [3B6B0000] <1> DISK_POINTER:dd MD_TBL6 ; Pointer to Diskette Parameter Table
24726 <1>
24727 <1> ; IBM PC-XT Model 286 source code ORGS.ASM (06/10/85) - 14/12/2014
24728 <1> ;-----
24729 <1> ; DISK_BASE :
24730 <1> ; THIS IS THE SET OF PARAMETERS REQUIRED FOR :
24731 <1> ; DISKETTE OPERATION. THEY ARE POINTED AT BY THE :
24732 <1> ; DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS, :
24733 <1> ; BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT :
24734 <1> ;-----
24735 <1>
24736 <1> ;DISK_BASE:
24737 <1> ; DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
24738 <1> ; DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
24739 <1> ; DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24740 <1> ; DB 2 ; 512 BYTES/SECTOR
24741 <1> ; ;DB 15 ; EOT (LAST SECTOR ON TRACK)
24742 <1> ; db 18 ; (EOT for 1.44MB diskette)
24743 <1> ; DB 01BH ; GAP LENGTH
24744 <1> ; DB 0FFH ; DTL
24745 <1> ; ;DB 054H ; GAP LENGTH FOR FORMAT
24746 <1> ; db 06ch ; (for 1.44MB dsikette)
24747 <1> ; DB 0F6H ; FILL BYTE FOR FORMAT
24748 <1> ; DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
24749 <1> ; DB 8 ; MOTOR START TIME (1/8 SECONDS)
24750 <1>
24751 <1> ;-----
24752 <1> ; ROM BIOS DATA AREAS :
24753 <1> ;-----
24754 <1>
24755 <1> ;DATA SEGMENT AT 40H ; ADDRESS= 0040:0000
24756 <1>
24757 <1> ;@EQUIP_FLAG DW ? ; INSTALLED HARDWARE FLAGS
24758 <1>
24759 <1> ;-----
24760 <1> ; DISKETTE DATA AREAS :
24761 <1> ;-----
24762 <1>
24763 <1> ;@SEEK_STATUSDB ? ; DRIVE RECALIBRATION STATUS
24764 <1> ; ; BIT 3-0 = DRIVE 3-0 RECALIBRATION
24765 <1> ; ; BEFORE NEXT SEEK IF BIT IS = 0
24766 <1> ;@MOTOR_STATUS DB ? ; MOTOR STATUS
24767 <1> ; ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
24768 <1> ; ; BIT 7 = CURRENT OPERATION IS A WRITE
24769 <1> ;@MOTOR_COUNTDB ? ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF
24770 <1> ;@DSKETTE_STATUS DB ? ; RETURN CODE STATUS BYTE
24771 <1> ; ; CMD_BLOCK IN STACK FOR DISK OPERATION
24772 <1> ;@NEC_STATUS DB 7 DUP(?) ; STATUS BYTES FROM DISKETTE OPERATION
24773 <1>
24774 <1> ;-----
24775 <1> ; POST AND BIOS WORK DATA AREA :
24776 <1> ;-----
24777 <1>
24778 <1> ;@INTR_FLAG DB ? ; FLAG INDICATING AN INTERRUPT HAPPENED
24779 <1>
24780 <1> ;-----
24781 <1> ; TIMER DATA AREA :
24782 <1> ;-----
24783 <1>
24784 <1> ; 17/12/2014 (IRQ 0 - INT 08H)
24785 <1> ;TIMER_LOW equ 46Ch ; Timer ticks (counter) @ 40h:006Ch
24786 <1> ;TIMER_HIGH equ 46Eh ; (18.2 timer ticks per second)
24787 <1> ;TIMER_OFL equ 470h ; Timer - 24 hours flag @ 40h:0070h
24788 <1>
24789 <1> ;-----
24790 <1> ; ADDITIONAL MEDIA DATA :
24791 <1> ;-----
24792 <1>
24793 <1> ;@LAstrate DB ? ; LAST DISKETTE DATA RATE SELECTED
24794 <1> ;@DSK_STATE DB ? ; DRIVE 0 MEDIA STATE
24795 <1> ; DB ? ; DRIVE 1 MEDIA STATE
24796 <1> ; DB ? ; DRIVE 0 OPERATION START STATE
24797 <1> ; DB ? ; DRIVE 1 OPERATION START STATE
24798 <1> ;@DSK_TRK DB ? ; DRIVE 0 PRESENT CYLINDER
24799 <1> ; DB ? ; DRIVE 1 PRESENT CYLINDER
24800 <1>
24801 <1> ;DATA ENDS ; END OF BIOS DATA SEGMENT
24802 <1>
24803 <1> ;-----
24804 <1> ; DRIVE TYPE TABLE :

```

```

24805
24806
24807
24808 00006ADC 01
24809
24810 00006ADD [FA6A0000]
24811 00006AE1 82
24812
24813 00006AE2 [076B0000]
24814 00006AE6 02
24815
24816 00006AE7 [146B0000]
24817 00006AEB 03
24818
24819 00006AEC [216B0000]
24820 00006AF0 84
24821
24822 00006AF1 [2E6B0000]
24823 00006AF5 04
24824
24825 00006AF6 [3B6B0000]
24826
24827
24828
24829
24830
24831
24832
24833
24834
24835
24836 00006AFA DF
24837 00006AFB 02
24838 00006AFC 25
24839 00006AFD 02
24840 00006AFE 09
24841 00006AFF 2A
24842 00006B00 FF
24843 00006B01 50
24844 00006B02 F6
24845 00006B03 0F
24846 00006B04 08
24847 00006B05 27
24848 00006B06 80
24849
24850
24851
24852
24853 00006B07 DF
24854 00006B08 02
24855 00006B09 25
24856 00006B0A 02
24857 00006B0B 09
24858 00006B0C 2A
24859 00006B0D FF
24860 00006B0E 50
24861 00006B0F F6
24862 00006B10 0F
24863 00006B11 08
24864 00006B12 27
24865 00006B13 40
24866
24867
24868
24869
24870 00006B14 DF
24871 00006B15 02
24872 00006B16 25
24873 00006B17 02
24874 00006B18 0F
24875 00006B19 1B
24876 00006B1A FF
24877 00006B1B 54
24878 00006B1C F6
24879 00006B1D 0F
24880 00006B1E 08
24881 00006B1F 4F
24882 00006B20 00
24883
24884
24885
24886
24887 00006B21 DF
24888 00006B22 02
24889 00006B23 25
24890 00006B24 02
24891 00006B25 09
24892 00006B26 2A
24893 00006B27 FF
24894 00006B28 50
24895 00006B29 F6
24896 00006B2A 0F

```

```

<1> ;-----
<1> ; 16/02/2015 (unix386.s, 32 bit modifications)
<1> DR_TYPE:
<1> DB 01 ;DRIVE TYPE, MEDIA TABLE
<1> ;DW MD_TBL1
<1> dd MD_TBL1
<1> DB 02+BIT7ON
<1> ;DW MD_TBL2
<1> dd MD_TBL2
<1> DR_DEFAULT: DB 02
<1> ;DW MD_TBL3
<1> dd MD_TBL3
<1> DB 03
<1> ;DW MD_TBL4
<1> dd MD_TBL4
<1> DB 04+BIT7ON
<1> ;DW MD_TBL5
<1> dd MD_TBL5
<1> DB 04
<1> ;DW MD_TBL6
<1> dd MD_TBL6
<1> DR_TYPE_E equ $ ; END OF TABLE
<1> ;DR_CNT EQU (DR_TYPE_E-DR_TYPE)/3
<1> DR_CNT equ (DR_TYPE_E-DR_TYPE)/5
<1> ;-----
<1> ; MEDIA/DRIVE PARAMETER TABLES :
<1> ;-----
<1> ;-----
<1> ; 360 KB MEDIA IN 360 KB DRIVE :
<1> ;-----
<1> MD_TBL1:
<1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
<1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
<1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
<1> DB 2 ; 512 BYTES/SECTOR
<1> DB 09 ; EOT (LAST SECTOR ON TRACK)
<1> DB 02AH ; GAP LENGTH
<1> DB 0FFH ; DTL
<1> DB 050H ; GAP LENGTH FOR FORMAT
<1> DB 0F6H ; FILL BYTE FOR FORMAT
<1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
<1> DB 8 ; MOTOR START TIME (1/8 SECONDS)
<1> DB 39 ; MAX. TRACK NUMBER
<1> DB RATE_250 ; DATA TRANSFER RATE
<1> ;-----
<1> ; 360 KB MEDIA IN 1.2 MB DRIVE :
<1> ;-----
<1> MD_TBL2:
<1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
<1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
<1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
<1> DB 2 ; 512 BYTES/SECTOR
<1> DB 09 ; EOT (LAST SECTOR ON TRACK)
<1> DB 02AH ; GAP LENGTH
<1> DB 0FFH ; DTL
<1> DB 050H ; GAP LENGTH FOR FORMAT
<1> DB 0F6H ; FILL BYTE FOR FORMAT
<1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
<1> DB 8 ; MOTOR START TIME (1/8 SECONDS)
<1> DB 39 ; MAX. TRACK NUMBER
<1> DB RATE_300 ; DATA TRANSFER RATE
<1> ;-----
<1> ; 1.2 MB MEDIA IN 1.2 MB DRIVE :
<1> ;-----
<1> MD_TBL3:
<1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
<1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
<1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
<1> DB 2 ; 512 BYTES/SECTOR
<1> DB 15 ; EOT (LAST SECTOR ON TRACK)
<1> DB 01BH ; GAP LENGTH
<1> DB 0FFH ; DTL
<1> DB 054H ; GAP LENGTH FOR FORMAT
<1> DB 0F6H ; FILL BYTE FOR FORMAT
<1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
<1> DB 8 ; MOTOR START TIME (1/8 SECONDS)
<1> DB 79 ; MAX. TRACK NUMBER
<1> DB RATE_500 ; DATA TRANSFER RATE
<1> ;-----
<1> ; 720 KB MEDIA IN 720 KB DRIVE :
<1> ;-----
<1> MD_TBL4:
<1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
<1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
<1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
<1> DB 2 ; 512 BYTES/SECTOR
<1> DB 09 ; EOT (LAST SECTOR ON TRACK)
<1> DB 02AH ; GAP LENGTH
<1> DB 0FFH ; DTL
<1> DB 050H ; GAP LENGTH FOR FORMAT
<1> DB 0F6H ; FILL BYTE FOR FORMAT
<1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)

```

```

24897 00006B2B 08      <1> DB      8          ; MOTOR START TIME (1/8 SECONDS)
24898 00006B2C 4F      <1> DB      79         ; MAX. TRACK NUMBER
24899 00006B2D 80      <1> DB     RATE_250      ; DATA TRANSFER RATE
24900                    <1> ;-----
24901                    <1> ; 720 KB MEDIA IN 1.44 MB DRIVE          :
24902                    <1> ;-----
24903                    <1> MD_TBL5:
24904 00006B2E DF      <1> DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
24905 00006B2F 02      <1> DB      2          ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
24906 00006B30 25      <1> DB     MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24907 00006B31 02      <1> DB      2          ; 512 BYTES/SECTOR
24908 00006B32 09      <1> DB      09         ; EOT (LAST SECTOR ON TRACK)
24909 00006B33 2A      <1> DB     02AH        ; GAP LENGTH
24910 00006B34 FF      <1> DB     0FFH        ; DTL
24911 00006B35 50      <1> DB     050H        ; GAP LENGTH FOR FORMAT
24912 00006B36 F6      <1> DB     0F6H        ; FILL BYTE FOR FORMAT
24913 00006B37 0F      <1> DB      15         ; HEAD SETTLE TIME (MILLISECONDS)
24914 00006B38 08      <1> DB      8          ; MOTOR START TIME (1/8 SECONDS)
24915 00006B39 4F      <1> DB      79         ; MAX. TRACK NUMBER
24916 00006B3A 80      <1> DB     RATE_250      ; DATA TRANSFER RATE
24917                    <1> ;-----
24918                    <1> ; 1.44 MB MEDIA IN 1.44 MB DRIVE          :
24919                    <1> ;-----
24920                    <1> MD_TBL6:
24921 00006B3B AF      <1> DB     10101111B    ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
24922 00006B3C 02      <1> DB      2          ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
24923 00006B3D 25      <1> DB     MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
24924 00006B3E 02      <1> DB      2          ; 512 BYTES/SECTOR
24925 00006B3F 12      <1> DB     18         ; EOT (LAST SECTOR ON TRACK)
24926 00006B40 1B      <1> DB     01BH        ; GAP LENGTH
24927 00006B41 FF      <1> DB     0FFH        ; DTL
24928 00006B42 6C      <1> DB     06CH        ; GAP LENGTH FOR FORMAT
24929 00006B43 F6      <1> DB     0F6H        ; FILL BYTE FOR FORMAT
24930 00006B44 0F      <1> DB      15         ; HEAD SETTLE TIME (MILLISECONDS)
24931 00006B45 08      <1> DB      8          ; MOTOR START TIME (1/8 SECONDS)
24932 00006B46 4F      <1> DB      79         ; MAX. TRACK NUMBER
24933 00006B47 00      <1> DB     RATE_500      ; DATA TRANSFER RATE
24934                    <1>
24935                    <1>
24936                    <1> ; << diskette.inc >>
24937                    <1> ; ++++++
24938                    <1> ;
24939                    <1> ;-----
24940                    <1> ; ROM BIOS DATA AREAS          :
24941                    <1> ;-----
24942                    <1>
24943                    <1> ; DATA          SEGMENT AT 40H          ; ADDRESS= 0040:0000
24944                    <1>
24945                    <1> ;-----
24946                    <1> ; FIXED DISK DATA AREAS          :
24947                    <1> ;-----
24948                    <1>
24949                    <1> ; DISK_STATUS1:      DB      0          ; FIXED DISK STATUS
24950                    <1> ; HF_NUM:          DB      0          ; COUNT OF FIXED DISK DRIVES
24951                    <1> ; CONTROL_BYTE:      DB      0          ; HEAD CONTROL BYTE
24952                    <1> ; @PORT_OFF      DB      ?          ; RESERVED (PORT OFFSET)
24953                    <1>
24954                    <1> ;-----
24955                    <1> ; ADDITIONAL MEDIA DATA          :
24956                    <1> ;-----
24957                    <1>
24958                    <1> ; @LASTRATE      DB      ?          ; LAST DISKETTE DATA RATE SELECTED
24959                    <1> ; HF_STATUS      DB      0          ; STATUS REGISTER
24960                    <1> ; HF_ERROR       DB      0          ; ERROR REGISTER
24961                    <1> ; HF_INT_FLAG    DB      0          ; FIXED DISK INTERRUPT FLAG
24962                    <1> ; HF_CNTRL       DB      0          ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1
24963                    <1> ; @DSK_STATE    DB      ?          ; DRIVE 0 MEDIA STATE
24964                    <1> ;          DB      ?          ; DRIVE 1 MEDIA STATE
24965                    <1> ;          DB      ?          ; DRIVE 0 OPERATION START STATE
24966                    <1> ;          DB      ?          ; DRIVE 1 OPERATION START STATE
24967                    <1> ; @DSK_TRK      DB      ?          ; DRIVE 0 PRESENT CYLINDER
24968                    <1> ;          DB      ?          ; DRIVE 1 PRESENT CYLINDER
24969                    <1>
24970                    <1> ; DATA          ENDS          ; END OF BIOS DATA SEGMENT
24971                    <1> ;
24972                    <1> ; ++++++
24973                    <1>
24974                    <1> ERR_TBL:
24975 00006B48 E0      <1> db      NO_ERR
24976 00006B49 024001BB <1> db      BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
24977 00006B4D 04BB100A <1> db      RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
24978                    <1>
24979                    <1> ; 17/12/2014 (mov ax, [cfd])
24980                    <1> ; 11/12/2014
24981 00006B51 00      <1> cfd:          db 0          ; current floppy drive (for GET_PARM)
24982                    <1> ; 17/12/2014          ; instead of 'DISK_POINTER'
24983 00006B52 01      <1> pfd:          db 1          ; previous floppy drive (for GET_PARM)
24984                    <1>          ; (initial value of 'pfd'
24985                    <1>          ; must be different then 'cfd' value
24986                    <1>          ; to force updating/initializing
24987                    <1>          ; current drive parameters)
24988 00006B53 90      <1> align 2

```



```

24989 <1>
24990 00006B54 F001 <1> HF_PORT: dw 1F0h ; Default = 1F0h
24991 <1> ; (170h)
24992 00006B56 F603 <1> HF_REG_PORT: dw 3F6h ; HF_PORT + 206h
24993 <1>
24994 <1> ; 05/01/2015
24995 00006B58 00 <1> hf_m_s: db 0 ; (0 = Master, 1 = Slave)
24996 <1>
24997 <1> ; *****
24998 ;;;
24999
25000 ; 27/08/2014
25001 scr_row:
25002 00006B59 E0810B00 dd 0B8000h + 0A0h + 0A0h + 0A0h ; Row 3
25003 scr_col:
25004 00006B5D 00000000 dd 0
25005
25006 ;; 14/08/2015
25007 ;;msgPM:
25008 ;; db "Protected mode and paging are ENABLED ... ", 0
25009 msgKVER:
25010 00006B61 526574726F20554E49- db "Retro UNIX 386 v1 - Kernel v0.2.0.16 [09/12/2015]", 0
25011 00006B6A 582033383620763120-
25012 00006B73 2D204B65726E656C20-
25013 00006B7C 76302E322E302E3136-
25014 00006B85 205B30392F31322F32-
25015 00006B8E 3031355D00
25016
25017 00006B93 90 Align 2
25018
25019 ; 20/08/2014
25020 ; /* This is the default interrupt "handler" :-) */
25021 ; Linux v0.12 (head.s)
25022 int_msg:
25023 00006B94 556E6B6E6F776E2069- db "Unknown interrupt ! ", 0
25024 00006B9D 6E7465727275707420-
25025 00006BA6 212000
25026
25027 00006BA9 90 Align 2
25028
25029 ; 21/08/2014
25030 timer_msg:
25031 00006BAA 49525120302028494E- db "IRQ 0 (INT 20h) ! Timer Interrupt : "
25032 00006BB3 542032306829202120-
25033 00006BBC 54696D657220496E74-
25034 00006BC5 657272757074203A20
25035 tcountstr:
25036 00006BCE 303030303020 db "00000 "
25037 00006BD4 00 db 0
25038
25039 00006BD5 90 Align 2
25040 ; 21/08/2014
25041 exc_msg:
25042 00006BD6 435055206578636570- db "CPU exception ! "
25043 00006BDF 74696F6E202120
25044 excnstr: ; 25/08/2014
25045 00006BE6 3F3F68202045495020- db "??h", " EIP : "
25046 00006BEF 3A20
25047 EIPstr: ; 29/08/2014
25048 00006BF1 00<rept> times 12 db 0
25049 rtc_msg:
25050 00006BFD 5265616C2054696D65- db "Real Time Clock - "
25051 00006C06 20436C6F636B202D20
25052 datestr:
25053 00006C0F 30302F30302F303030- db "00/00/0000"
25054 00006C18 30
25055 00006C19 20 db " "
25056 daystr:
25057 00006C1A 44415920 db "DAY "
25058 timestr:
25059 00006C1E 30303A30303A3030 db "00:00:00"
25060 00006C26 20 db " "
25061 00006C27 00 db 0
25062
25063 daytmp:
25064 ; 28/02/2015
25065 00006C28 3F3F3F2053554E204D- db "??? SUN MON TUE WED THU FRI SAT "
25066 00006C31 4F4E20545545205745-
25067 00006C3A 442054485520465249-
25068 00006C43 2053415420
25069
25070 00006C48 FF ptime_seconds: db 0FFh
25071
25072 ; 23/02/2015
25073 ; 25/08/2014
25074 ;scounter:
25075 ; db 5
25076 ; db 19
25077
25078 ; 05/11/2014
25079 msg_out_of_memory:
25080 00006C49 070D0A db 07h, 0Dh, 0Ah

```

```

25081 00006C4C 496E737566666696369-          db      'Insufficient memory ! (Minimum 2 MB memory is needed.)'
25082 00006C55 656E74206D656D6F72-
25083 00006C5E 79202120284D696E69-
25084 00006C67 6D756D2032204D4220-
25085 00006C70 6D656D6F7279206973-
25086 00006C79 206E65656465642E29
25087 00006C82 0D0A00          db      0Dh, 0Ah, 0
25088                                     ;
25089 setup_error_msg:
25090 00006C85 0D0A          db 0Dh, 0Ah
25091 00006C87 4469736B2053657475-      db 'Disk Setup Error!'
25092 00006C90 70204572726F7221
25093 00006C98 0D0A00          db 0Dh, 0Ah,0
25094
25095          ; 02/09/2014 (Retro UNIX 386 v1)
25096          ; crt_ulc : db 0 ; upper left column (for scroll)
25097          ; db 0 ; upper left row (for scroll)
25098
25099          ; crt_lrc : db 79 ; lower right column (for scroll)
25100          ; db 24 ; lower right row (for scroll)
25101
25102
25103          ; 06/11/2014 (Temporary Data)
25104          ; Memory Information message
25105          ; 14/08/2015
25106 msg_memory_info:
25107 00006C9B 07          db      07h
25108 00006C9C 0D0A          db      0Dh, 0Ah
25109          ;db      "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah
25110 00006C9E 546F74616C206D656D-      db      "Total memory : "
25111 00006CA7 6F7279203A20
25112          mem_total_b_str: ; 10 digits
25113 00006CAD 30303030303030303030-      db      "0000000000 bytes", 0Dh, 0Ah
25114 00006CB6 302062797465730D0A
25115 00006CBF 202020202020202020-      db      "
25116 00006CC8 202020202020202020          ", 20h, 20h, 20h
25117          mem_total_p_str: ; 7 digits
25118 00006CD1 303030303030302070-      db      "00000000 pages", 0Dh, 0Ah
25119 00006CDA 616765730D0A
25120 00006CE0 0D0A          db      0Dh, 0Ah
25121 00006CE2 46726565206D656D6F-      db      "Free memory : "
25122 00006CEB 727920203A20
25123          free_mem_b_str: ; 10 digits
25124 00006CF1 3F3F3F3F3F3F3F3F3F-      db      "????????? bytes", 0Dh, 0Ah
25125 00006CFA 3F2062797465730D0A
25126 00006D03 202020202020202020-      db      "
25127 00006D0C 202020202020202020          ", 20h, 20h, 20h
25128          free_mem_p_str: ; 7 digits
25129 00006D15 3F3F3F3F3F3F3F2070-      db      "??????? pages", 0Dh, 0Ah
25130 00006D1E 616765730D0A
25131 00006D24 0D0A00          db      0Dh, 0Ah, 0
25132
25133          dsk_ready_msg:
25134 00006D27 0D0A          db      0Dh, 0Ah
25135          dsktype:
25136 00006D29 6664          db      'fd'
25137          dskx:
25138 00006D2B 30          db      '0'
25139 00006D2C 20          db      20h
25140 00006D2D 697320524541445920-      db      'is READY ...'
25141 00006D36 2E2E2E
25142 00006D39 00          db      0
25143          nextline:
25144 00006D3A 0D0A00          db      0Dh, 0Ah, 0
25145
25146          ; KERNEL - SYSINIT Messages
25147          ; 24/08/2015
25148          ; 13/04/2015 - (Retro UNIX 386 v1 Beginning)
25149          ; 14/07/2013
25150          ;kernel_init_err_msg:
25151          ;db 0Dh, 0Ah
25152          ;db 07h
25153          ;db 'Kernel initialization ERROR !'
25154          ;db 0Dh, 0Ah, 0
25155          ; 24/08/2015
25156          ;;; (temporary kernel init message has been removed
25157          ;;; from 'sys_init' code)
25158          ;kernel_init_ok_msg:
25159          ;db 0Dh, 0Ah
25160          ;db 07h
25161          ;db 'Welcome to Retro UNIX 386 v1 Operating System !'
25162          ;db 0Dh, 0Ah
25163          ; db 'by Erdogan Tan - 09/12/2015 (v0.2.0.16)'
25164          ;db 0Dh, 0Ah, 0
25165          panic_msg:
25166 00006D3D 0D0A07          db 0Dh, 0Ah, 07h
25167 00006D40 4552524F523A204B65-      db 'ERROR: Kernel Panic !'
25168 00006D49 726E656C2050616E69-
25169 00006D52 632021
25170 00006D55 0D0A00          db 0Dh, 0Ah, 0
25171          etc_init_err_msg:
25172 00006D58 0D0A          db 0Dh, 0Ah

```

```

25173 00006D5A 07 db 07h
25174 00006D5B 4552524F523A202F65- db 'ERROR: /etc/init !?'
25175 00006D64 74632F696E69742021-
25176 00006D6D 3F
25177 00006D6E 0D0A00 db 0Dh, 0Ah, 0
25178
25179 ; 10/05/2015
25180 badsys_msg:
25181 00006D71 0D0A db 0Dh, 0Ah
25182 00006D73 07 db 07h
25183 00006D74 496E76616C69642053- db 'Invalid System Call !'
25184 00006D7D 797374656D2043616C-
25185 00006D86 6C2021
25186 00006D89 0D0A db 0Dh, 0Ah
25187 00006D8B 4541583A20 db 'EAX: '
25188 bsys_msg_eax:
25189 00006D90 30303030303030303068 db '00000000h'
25190 00006D99 0D0A db 0Dh, 0Ah
25191 00006D9B 4549503A20 db 'EIP: '
25192 bsys_msg_eip:
25193 00006DA0 30303030303030303068 db '00000000h'
25194 00006DA9 0D0A00 db 0Dh, 0Ah, 0
25195
25196 BSYS_M_SIZE equ $ - badsys_msg
25197
25198
25199 align 2
25200
25201 ; EPOCH Variables
25202 ; 13/04/2015 - Retro UNIX 386 v1 Beginning
25203 ; 09/04/2013 epoch variables
25204 ; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013
25205 ;
25206 00006DAC B207 year: dw 1970
25207 00006DAE 0100 month: dw 1
25208 00006DB0 0100 day: dw 1
25209 00006DB2 0000 hour: dw 0
25210 00006DB4 0000 minute: dw 0
25211 00006DB6 0000 second: dw 0
25212
25213 DMonth:
25214 00006DB8 0000 dw 0
25215 00006DBA 1F00 dw 31
25216 00006DBC 3B00 dw 59
25217 00006DBE 5A00 dw 90
25218 00006DC0 7800 dw 120
25219 00006DC2 9700 dw 151
25220 00006DC4 B500 dw 181
25221 00006DC6 D400 dw 212
25222 00006DC8 F300 dw 243
25223 00006DCA 1101 dw 273
25224 00006DCC 3001 dw 304
25225 00006DCE 4E01 dw 334
25226
25227 ; 04/11/2014 (Retro UNIX 386 v1)
25228 00006DD0 0000 mem_1m_1k: dw 0 ; Number of contiguous KB between
25229 ; 1 and 16 MB, max. 3C00h = 15 MB.
25230 00006DD2 0000 mem_16m_64k: dw 0 ; Number of contiguous 64 KB blocks
25231 ; between 16 MB and 4 GB.
25232
25233 ; 12/11/2014 (Retro UNIX 386 v1)
25234 00006DD4 00 boot_drv: db 0 ; boot drive number (physical)
25235 ; 24/11/2014
25236 00006DD5 00 drv: db 0
25237 00006DD6 00 last_drv: db 0 ; last hdd
25238 00006DD7 00 hdc: db 0 ; number of hard disk drives
25239 ; (present/detected)
25240 ;
25241 ; 24/11/2014 (Retro UNIX 386 v1)
25242 ; Physical drive type & flags
25243 00006DD8 00 fd0_type: db 0 ; floppy drive type
25244 00006DD9 00 fd1_type: db 0 ; 4 = 1.44 Mb, 80 track, 3.5" (18 spt)
25245 ; 6 = 2.88 Mb, 80 track, 3.5" (36 spt)
25246 ; 3 = 720 Kb, 80 track, 3.5" (9 spt)
25247 ; 2 = 1.2 Mb, 80 track, 5.25" (15 spt)
25248 ; 1 = 360 Kb, 40 track, 5.25" (9 spt)
25249 00006DDA 00 hd0_type: db 0 ; EDD status for hd0 (bit 7 = present flag)
25250 00006ddb 00 hd1_type: db 0 ; EDD status for hd1 (bit 7 = present flag)
25251 00006DDC 00 hd2_type: db 0 ; EDD status for hd2 (bit 7 = present flag)
25252 00006DDD 00 hd3_type: db 0 ; EDD status for hd3 (bit 7 = present flag)
25253 ; bit 0 - Fixed disk access subset supported
25254 ; bit 1 - Drive locking and ejecting
25255 ; bit 2 - Enhanced disk drive support
25256 ; bit 3 = Reserved (64 bit EDD support)
25257 ; (If bit 0 is '1' Retro UNIX 386 v1
25258 ; will interpret it as 'LBA ready'!)
25259
25260 ; 11/03/2015 - 10/07/2015
25261 00006DDE 00000000000000000000- drv.cylinders: dw 0,0,0,0,0,0,0
25262 00006DE7 0000000000
25263 00006DEC 00000000000000000000- drv.heads: dw 0,0,0,0,0,0,0
25264 00006DF5 0000000000

```

```

25265 00006DFA 000000000000000000-   drv.spt:      dw 0,0,0,0,0,0,0
25266 00006E03 000000000000000000-
25267 00006E08 000000000000000000-   drv.size:     dd 0,0,0,0,0,0,0
25268 00006E11 000000000000000000-
25269 00006E1A 000000000000000000-
25270 00006E23 00
25271 00006E24 000000000000000000-   drv.status:   db 0,0,0,0,0,0,0
25272 00006E2B 000000000000000000-   drv.error:    db 0,0,0,0,0,0,0
25273
25274
25275 00006E32 90<rept>               align 16
25276
25277
25278
25279
25280
25281
25282
25283
25284 00006E40 <res 00000200>         idt:
25285                                     resb 64*8 ; INT 0 to INT 3Fh
25286                                     idt_end:
25287                                     ;alignb 4
25288
25289
25290                                     task_state_segment:
25291                                     ; 24/03/2015
25292 00007040 <res 00000002>         tss.link:    resw 1
25293 00007042 <res 00000002>                                     resw 1
25294 00007044 <res 00000004>                                     ; tss offset 4
25295 00007048 <res 00000002>         tss.esp0:    resd 1
25296 0000704A <res 00000002>                                     resw 1
25297 0000704C <res 00000004>         tss.espl:    resd 1
25298 00007050 <res 00000002>                                     resw 1
25299 00007052 <res 00000002>                                     resw 1
25300 00007054 <res 00000004>         tss.esp2:    resd 1
25301 00007058 <res 00000002>         tss.ss2:    resw 1
25302 0000705A <res 00000002>                                     resw 1
25303                                     ; tss offset 28
25304 0000705C <res 00000004>         tss.CR3:    resd 1
25305 00007060 <res 00000004>         tss.eip:    resd 1
25306 00007064 <res 00000004>         tss.eflags: resd 1
25307                                     ; tss offset 40
25308 00007068 <res 00000004>         tss.eax:    resd 1
25309 0000706C <res 00000004>         tss.ecx:    resd 1
25310 00007070 <res 00000004>         tss.edx:    resd 1
25311 00007074 <res 00000004>         tss.ebx:    resd 1
25312 00007078 <res 00000004>         tss.esp:    resd 1
25313 0000707C <res 00000004>         tss.ebp:    resd 1
25314 00007080 <res 00000004>         tss.esi:    resd 1
25315 00007084 <res 00000004>         tss.edi:    resd 1
25316                                     ; tss offset 72
25317 00007088 <res 00000002>         tss.ES:    resw 1
25318 0000708A <res 00000002>                                     resw 1
25319 0000708C <res 00000002>         tss.CS:    resw 1
25320 0000708E <res 00000002>                                     resw 1
25321 00007090 <res 00000002>         tss.SS:    resw 1
25322 00007092 <res 00000002>                                     resw 1
25323 00007094 <res 00000002>         tss.DS:    resw 1
25324 00007096 <res 00000002>                                     resw 1
25325 00007098 <res 00000002>         tss.FS:    resw 1
25326 0000709A <res 00000002>                                     resw 1
25327 0000709C <res 00000002>         tss.GS:    resw 1
25328 0000709E <res 00000002>                                     resw 1
25329 000070A0 <res 00000002>         tss.LDTR:   resw 1
25330 000070A2 <res 00000002>                                     resw 1
25331                                     ; tss offset 100
25332 000070A4 <res 00000002>                                     resw 1
25333 000070A6 <res 00000002>         tss.IOPB:   resw 1
25334                                     ; tss offset 104
25335                                     tss_end:
25336
25337 000070A8 <res 00000004>         k_page_dir:  resd 1 ; Kernel's (System) Page Directory address
25338                                     ; (Physical address = Virtual address)
25339 000070AC <res 00000004>         memory_size: resd 1 ; memory size in pages
25340 000070B0 <res 00000004>         free_pages:  resd 1 ; number of free pages
25341 000070B4 <res 00000004>         next_page:   resd 1 ; offset value in M.A.T. for
25342                                     ; first free page search
25343 000070B8 <res 00000004>         last_page:    resd 1 ; offset value in M.A.T. which
25344                                     ; next free page search will be
25345                                     ; stopped after it. (end of M.A.T.)
25346 000070BC <res 00000004>         first_page:   resd 1 ; offset value in M.A.T. which
25347                                     ; first free page search
25348                                     ; will be started on it. (for user)
25349 000070C0 <res 00000004>         mat_size:    resd 1 ; Memory Allocation Table size in pages
25350
25351                                     ;;;
25352                                     ; 02/09/2014 (Retro UNIX 386 v1)
25353                                     ; 04/12/2013 (Retro UNIX 8086 v1)
25354 000070C4 <res 00000002>         CRT_START:   resw 1 ; starting address in regen buffer
25355                                     ; NOTE: active page only
25356 000070C6 <res 00000010>         cursor_posn: resw 8 ; cursor positions for video pages

```

```

25357 active_page:
25358 000070D6 <res 00000001> ptty: resb 1 ; current tty
25359 ; 01/07/2015
25360 000070D7 <res 00000001> ccolor: resb 1 ; current color attributes ('sysmsg')
25361 ; 26/10/2015
25362 ; 07/09/2014
25363 000070D8 <res 00000014> ttychr: resw ntty+2 ; Character buffer (multiscreen)
25364
25365 ; 21/08/2014
25366 000070EC <res 00000004> tcount: resd 1
25367
25368 ; 18/05/2015 (03/06/2013 - Retro UNIX 8086 v1 feature only!)
25369 000070F0 <res 00000004> p_time: resd 1 ; present time (for systime & sysmdate)
25370
25371 ; 18/05/2015 (16/08/2013 - Retro UNIX 8086 v1 feature only !)
25372 ; (open mode locks for pseudo TTYS)
25373 ; [ major tty locks (return error in any conflicts) ]
25374 000070F4 <res 00000014> ttysl: resw ntty+2 ; opening locks for TTYS.
25375
25376 ; 15/04/2015 (Retro UNIX 386 v1)
25377 ; 22/09/2013 (Retro UNIX 8086 v1)
25378 00007108 <res 0000000A> wlist: resb ntty+2 ; wait channel list (0 to 9 for TTYS)
25379 ; 15/04/2015 (Retro UNIX 386 v1)
25380 ;; 12/07/2014 -> sp_init set comm. parameters as 0E3h
25381 ;; 0 means serial port is not available
25382 ;;comprm: ; 25/06/2014
25383 00007112 <res 00000001> com1p: resb 1 ;;0E3h
25384 00007113 <res 00000001> com2p: resb 1 ;;0E3h
25385
25386 ; 17/11/2015
25387 ; request for response (from the terminal)
25388 00007114 <res 00000002> req_resp: resw 1
25389 ; 07/11/2015
25390 00007116 <res 00000001> ccomport: resb 1 ; current COM (serial) port
25391 ; (0= COM1, 1= COM2)
25392 ; 09/11/2015
25393 00007117 <res 00000001> comqr: resb 1 ; 'query or response' sign (u9.s, 'sndc')
25394 ; 07/11/2015
25395 00007118 <res 00000002> rchar: resw 1 ; last received char for COM 1 and COM 2
25396 0000711A <res 00000002> schar: resw 1 ; last sent char for COM 1 and COM 2
25397
25398 ; 23/10/2015
25399 ; SERIAL PORTS - COMMUNICATION MODES
25400 ; (Retro UNIX 386 v1 feature only!)
25401 ; 0 - command mode (default/initial mode)
25402 ; 1 - terminal mode (Retro UNIX 386 v1 terminal, ascii chars)
25403 ;; communication modes for futre versions:
25404 ; // 2 - keyboard mode (ascii+scancode input)
25405 ; // 3 - mouse mode
25406 ; // 4 - device control (output) mode
25407 ; VALID COMMANDS for current version:
25408 ; 'LOGIN'
25409 ; Login request: db 0FFh, 'LOGIN', 0
25410 ; ("Retro UNIX 386 v1 terminal requests login")
25411 ; Login response: db 0FFh, 'login', 0
25412 ; ("login request accepted, wait for login prompt")
25413 ; When a login requests is received and acknowledged (by
25414 ; serial port interrupt handler (communication procedure),
25415 ; Retro UNIX 386 v1 operating system will start terminal mode
25416 ; (login procedure) by changing comm. mode to 1 (terminal mode)
25417 ; and then running 'etc/getty' for tty8 (COM1) or tty9 (COM2)
25418 ;
25419 ; 'sys connect' system call is used to change communication mode
25420 ; except 'LOGIN' command which is used to start terminal mode
25421 ; by using (COM port) terminal.
25422
25423 ;comlown: resb 1 ; COM1 owner (u.uno)
25424 ;com2own: resb 1 ; COM2 owner (u.uno)
25425 ;comlmode: resb 1 ; communication mode for COM1
25426 ;comlcom: resb 1 ; communication command for COM1
25427 ;com2mode: resb 1 ; communication mode for COM1
25428 ;com2com: resb 1 ; communication command for COM1
25429 ;com1cbufp: resb 8 ; COM1 command buffer char pointer
25430 ;com2cbufp: resb 8 ; COM2 command buffer char pointer
25431 ;com1cbuf: resb 8 ; COM2 command buffer
25432 ;com2cbuf: resb 8 ; COM2 command buffer
25433
25434 ; 22/08/2014 (RTC)
25435 ; (Packed BCD)
25436 0000711C <res 00000001> time_seconds: resb 1
25437 0000711D <res 00000001> time_minutes: resb 1
25438 0000711E <res 00000001> time_hours: resb 1
25439 0000711F <res 00000001> date_wday: resb 1
25440 00007120 <res 00000001> date_day: resb 1
25441 00007121 <res 00000001> date_month: resb 1
25442 00007122 <res 00000001> date_year: resb 1
25443 00007123 <res 00000001> date_century: resb 1
25444
25445 %include 'diskbss.inc' ; UNINITIALIZED DISK (BIOS) DATA
25446 <1> ; Retro UNIX 386 v1 Kernel - DISKBSS.INC
25447 <1> ; Last Modification: 10/07/2015
25448 <1> ; (Unnitialized Disk Parameters Data section for 'DISKIO.INC')

```

```

25449 <1> ;
25450 <1> ; *****
25451 <1>
25452 <1> alignb 2
25453 <1>
25454 <1> ;-----
25455 <1> ; TIMER DATA AREA :
25456 <1> ;-----
25457 <1>
25458 <1> TIMER_LH: ; 16/02/205
25459 00007124 <res 00000002> <1> TIMER_LOW: resw 1 ; LOW WORD OF TIMER COUNT
25460 00007126 <res 00000002> <1> TIMER_HIGH: resw 1 ; HIGH WORD OF TIMER COUNT
25461 00007128 <res 00000001> <1> TIMER_OFL: resb 1 ; TIMER HAS ROLLED OVER SINCE LAST READ
25462 <1>
25463 <1> ;-----
25464 <1> ; DISKETTE DATA AREAS :
25465 <1> ;-----
25466 <1>
25467 00007129 <res 00000001> <1> SEEK_STATUS: resb 1
25468 0000712A <res 00000001> <1> MOTOR_STATUS: resb 1
25469 0000712B <res 00000001> <1> MOTOR_COUNT: resb 1
25470 0000712C <res 00000001> <1> DSKETTE_STATUS: resb 1
25471 0000712D <res 00000007> <1> NEC_STATUS: resb 7
25472 <1>
25473 <1> ;-----
25474 <1> ; ADDITIONAL MEDIA DATA :
25475 <1> ;-----
25476 <1>
25477 00007134 <res 00000001> <1> LASTRATE: resb 1
25478 00007135 <res 00000001> <1> HF_STATUS: resb 1
25479 00007136 <res 00000001> <1> HF_ERROR: resb 1
25480 00007137 <res 00000001> <1> HF_INT_FLAG: resb 1
25481 00007138 <res 00000001> <1> HF_CNTRL: resb 1
25482 00007139 <res 00000004> <1> DSK_STATE: resb 4
25483 0000713D <res 00000002> <1> DSK_TRK: resb 2
25484 <1>
25485 <1> ;-----
25486 <1> ; FIXED DISK DATA AREAS :
25487 <1> ;-----
25488 <1>
25489 0000713F <res 00000001> <1> DISK_STATUS1: resb 1 ; FIXED DISK STATUS
25490 00007140 <res 00000001> <1> HF_NUM: resb 1 ; COUNT OF FIXED DISK DRIVES
25491 00007141 <res 00000001> <1> CONTROL_BYTE: resb 1 ; HEAD CONTROL BYTE
25492 <1> ; @PORT_OFF resb 1 ; RESERVED (PORT OFFSET)
25493 <1> ; port1_off resb 1 ; Hard disk controller 1 - port offset
25494 <1> ; port2_off resb 1 ; Hard idsk controller 2 - port offset
25495 <1>
25496 00007142 <res 00000002> <1> alignb 4
25497 <1>
25498 <1> ; HF_TBL_VEC: resd 1 ; Primary master disk param. tbl. pointer
25499 <1> ; HF1_TBL_VEC: resd 1 ; Primary slave disk param. tbl. pointer
25500 <1> HF_TBL_VEC: ; 22/12/2014
25501 00007144 <res 00000004> <1> HDPM_TBL_VEC: resd 1 ; Primary master disk param. tbl. pointer
25502 00007148 <res 00000004> <1> HDPS_TBL_VEC: resd 1 ; Primary slave disk param. tbl. pointer
25503 0000714C <res 00000004> <1> HDSM_TBL_VEC: resd 1 ; Secondary master disk param. tbl. pointer
25504 00007150 <res 00000004> <1> HDSS_TBL_VEC: resd 1 ; Secondary slave disk param. tbl. pointer
25505 <1>
25506 <1> ; 03/01/2015
25507 00007154 <res 00000001> <1> LBAMode: resb 1
25508 <1>
25509 <1> ; *****
25510
25511 ;;; Real Mode Data (10/07/2015 - BSS)
25512
25513 ; alignb 2
25514
25515 %include 'ux.s' ; 12/04/2015 (unix system/user/process data)
25516 <1> ; Retro UNIX 386 v1 Kernel - ux.s
25517 <1> ; Last Modification: 13/11/2015
25518 <1> ;
25519 <1> ; ////////////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS ///////////////////
25520 <1> ; (Modified from
25521 <1> ; Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
25522 <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
25523 <1> ; -----
25524 <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
25525 <1> ; (Original) Source Code by Ken Thompson (1971-1972)
25526 <1> ; <Bell Laboratories (17/3/1972)>
25527 <1> ; <Preliminary Release of UNIX Implementation Document>
25528 <1> ; (Section E10 (17/3/1972) - ux.s)
25529 <1> ; *****
25530 <1>
25531 00007155 <res 00000001> <1> alignb 2
25532 <1>
25533 <1> inode:
25534 <1> ; 11/03/2013.
25535 <1> ; Derived from UNIX v1 source code 'inode' structure (ux).
25536 <1> ; i.
25537 <1>
25538 00007156 <res 00000002> <1> i.flgs: resw 1
25539 00007158 <res 00000001> <1> i.nlks: resb 1
25540 00007159 <res 00000001> <1> i.uid: resb 1

```

```

25541 0000715A <res 00000002> <1> i.size: resw 1 ; size
25542 0000715C <res 00000010> <1> i.dskp: resw 8 ; 16 bytes
25543 0000716C <res 00000004> <1> i.ctim: resd 1
25544 00007170 <res 00000004> <1> i.mtim: resd 1
25545 00007174 <res 00000002> <1> i.rsvd: resw 1 ; Reserved (ZERO/Undefined word for UNIX v1.)
25546 <1>
25547 <1> I_SIZE equ $ - inode
25548 <1>
25549 <1> process:
25550 <1> ; 06/05/2015
25551 <1> ; 11/03/2013 - 05/02/2014
25552 <1> ;Derived from UNIX v1 source code 'proc' structure (ux).
25553 <1> ;p.
25554 <1>
25555 00007176 <res 00000020> <1> p.pid: resw nproc
25556 00007196 <res 00000020> <1> p.ppid: resw nproc
25557 000071B6 <res 00000020> <1> p.break: resw nproc
25558 000071D6 <res 00000010> <1> p.ttyc: resb nproc ; console tty in Retro UNIX 8086 v1.
25559 000071E6 <res 00000010> <1> p.waitc: resb nproc ; waiting channel in Retro UNIX 8086 v1.
25560 000071F6 <res 00000010> <1> p.link: resb nproc
25561 00007206 <res 00000010> <1> p.stat: resb nproc
25562 <1>
25563 <1> ; 06/05/2015 (Retro UNIX 386 v1 fetaure only !)
25564 00007216 <res 00000040> <1> p.upage: resd nproc ; Physical address of the process's
25565 <1> ; 'user' structure
25566 <1>
25567 <1>
25568 <1> P_SIZE equ $ - process
25569 <1>
25570 <1>
25571 <1> ; fsp table (original UNIX v1)
25572 <1> ;
25573 <1> ;Entry
25574 <1> ; 15 0
25575 <1> ; 1 |---|-----|
25576 <1> ; |r/w| i-number of open file |
25577 <1> ; |---|-----|
25578 <1> ; | device number |
25579 <1> ; |-----|
25580 <1> ; (*) | offset pointer, i.e., r/w pointer to file |
25581 <1> ; |-----|
25582 <1> ; | flag that says | number of processes |
25583 <1> ; | file deleted | that have file open |
25584 <1> ; |-----|
25585 <1> ; 2 |-----|
25586 <1> ; |-----|
25587 <1> ; |-----|
25588 <1> ; |-----|
25589 <1> ; |-----|
25590 <1> ; |-----|
25591 <1> ; |-----|
25592 <1> ; |-----|
25593 <1> ; 3 |-----|
25594 <1> ; |-----|
25595 <1> ;
25596 <1> ; (*) Retro UNIX 386 v1 modification: 32 bit offset pointer
25597 <1>
25598 <1>
25599 <1> ; 15/04/2015
25600 00007256 <res 000001F4> <1> fsp: resb nfiles * 10 ; 11/05/2015 (8 -> 10)
25601 0000744A <res 00000018> <1> bufp: resd (nbuf+2) ; will be initialized
25602 00007462 <res 00000002> <1> ii: resw 1
25603 00007464 <res 00000002> <1> idev: resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
25604 00007466 <res 00000002> <1> cdev: resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
25605 <1> ; 18/05/2015
25606 <1> ; 26/04/2013 device/drive parameters (Retro UNIX 8086 v1 feature only!)
25607 <1> ; 'UNIX' device numbers (as in 'cdev' and 'u.cdrv')
25608 <1> ; 0 -> root device (which has Retro UNIX 8086 v1 file system)
25609 <1> ; 1 -> mounted device (which has Retro UNIX 8086 v1 file system)
25610 <1> ; 'Retro UNIX 8086 v1' device numbers: (for disk I/O procedures)
25611 <1> ; 0 -> fd0 (physical drive, floppy disk 1), physical drive number = 0
25612 <1> ; 1 -> fd1 (physical drive, floppy disk 2), physical drive number = 1
25613 <1> ; 2 -> hd0 (physical drive, hard disk 1), physical drive number = 80h
25614 <1> ; 3 -> hd1 (physical drive, hard disk 2), physical drive number = 81h
25615 <1> ; 4 -> hd2 (physical drive, hard disk 3), physical drive number = 82h
25616 <1> ; 5 -> hd3 (physical drive, hard disk 4), physical drive number = 83h
25617 00007468 <res 00000001> <1> rdev: resb 1 ; root device number ; Retro UNIX 8086 v1 feature only!
25618 <1> ; as above, for physical drives numbers in following table
25619 00007469 <res 00000001> <1> mdev: resb 1 ; mounted device number ; Retro UNIX 8086 v1 feature only!
25620 <1> ; 15/04/2015
25621 0000746A <res 00000001> <1> active: resb 1
25622 0000746B <res 00000001> <1> resb 1 ; 09/06/2015
25623 0000746C <res 00000002> <1> mnti: resw 1
25624 0000746E <res 00000002> <1> mpid: resw 1
25625 00007470 <res 00000002> <1> rootdir: resw 1
25626 <1> ; 14/02/2014
25627 <1> ; Major Modification: Retro UNIX 8086 v1 feature only!
25628 <1> ; Single level run queue
25629 <1> ; (in order to solve sleep/wakeup lock)
25630 00007472 <res 00000002> <1> runq: resw 1
25631 00007474 <res 00000001> <1> imod: resb 1
25632 00007475 <res 00000001> <1> smod: resb 1

```

```

25633 00007476 <res 00000001> <1> mmod: resb 1
25634 00007477 <res 00000001> <1> sysflg: resb 1
25635 <1>
25636 <1> alignb 4
25637 <1>
25638 <1> user:
25639 <1> ; 18/10/2015
25640 <1> ; 12/10/2015
25641 <1> ; 21/09/2015
25642 <1> ; 24/07/2015
25643 <1> ; 16/06/2015
25644 <1> ; 09/06/2015
25645 <1> ; 11/05/2015
25646 <1> ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit modifications)
25647 <1> ; 10/10/2013
25648 <1> ; 11/03/2013.
25649 <1> ;Derived from UNIX v1 source code 'user' structure (ux).
25650 <1> ;u.
25651 <1>
25652 00007478 <res 00000004> <1> u.sp: resd 1 ; esp (kernel stack at the beginning of 'sysent')
25653 0000747C <res 00000004> <1> u.usp: resd 1 ; esp (kernel stack points to user's registers)
25654 00007480 <res 00000004> <1> u.r0: resd 1 ; eax
25655 00007484 <res 00000002> <1> u.cdir: resw 1
25656 00007486 <res 0000000A> <1> u.fp: resb 10
25657 00007490 <res 00000004> <1> u.fofp: resd 1
25658 00007494 <res 00000004> <1> u.dirp: resd 1
25659 00007498 <res 00000004> <1> u.namep: resd 1
25660 0000749C <res 00000004> <1> u.off: resd 1
25661 000074A0 <res 00000004> <1> u.base: resd 1
25662 000074A4 <res 00000004> <1> u.count: resd 1
25663 000074A8 <res 00000004> <1> u.nread: resd 1
25664 000074AC <res 00000004> <1> u.break: resd 1 ; break
25665 000074B0 <res 00000002> <1> u.ttyp: resw 1
25666 000074B2 <res 0000000A> <1> u.dirbuf: resb 10
25667 <1> ;u.pri: resw 1 ; 14/02/2014
25668 000074BC <res 00000001> <1> u.quant: resb 1 ; Retro UNIX 8086 v1 Feature only ! (uquant)
25669 000074BD <res 00000001> <1> u.pri: resb 1 ;
25670 000074BE <res 00000002> <1> u.intr: resw 1
25671 000074C0 <res 00000002> <1> u.quit: resw 1
25672 <1> ;u.emt: resw 1 ; 10/10/2013
25673 000074C2 <res 00000002> <1> u.ilgins: resw 1
25674 000074C4 <res 00000002> <1> u.cdrv: resw 1 ; cdev
25675 000074C6 <res 00000001> <1> u.uid: resb 1 ; uid
25676 000074C7 <res 00000001> <1> u.ruid: resb 1
25677 000074C8 <res 00000001> <1> u.bsyz: resb 1
25678 000074C9 <res 00000001> <1> u.uno: resb 1
25679 000074CA <res 00000004> <1> u.upage: resd 1 ; 16/04/2015 - Retro Unix 386 v1 feature only !
25680 <1> ; tty number (rtty, rcvt, wtty)
25681 000074CE <res 00000001> <1> u.ttyn: resb 1 ; 28/07/2013 - Retro Unix 8086 v1 feature only !
25682 <1> ; last error number
25683 000074CF <res 00000004> <1> u.error: resd 1 ; 28/07/2013 - 09/03/2015
25684 <1> ; Retro UNIX 8086/386 v1 feature only!
25685 000074D3 <res 00000004> <1> u.pgdir: resd 1 ; 09/03/2015 (page dir addr of process)
25686 000074D7 <res 00000004> <1> u.ppgdir: resd 1 ; 06/05/2015 (page dir addr of the parent process)
25687 000074DB <res 00000004> <1> u.pbase: resd 1 ; 20/05/2015 (physical base/transfer address)
25688 000074DF <res 00000002> <1> u.pcount: resw 1 ; 20/05/2015 (byte -transfer- count for page)
25689 <1> ;u.pncount: resw 1
25690 <1> ; 16/06/2015 (byte -transfer- count for page, 'namei', 'mkdir')
25691 <1> ;u.pnbase: resd 1
25692 <1> ; 16/06/2015 (physical base/transfer address, 'namei', 'mkdir')
25693 <1> ; 09/06/2015
25694 000074E1 <res 00000001> <1> u.kcall: resb 1 ; The caller is 'namei' (dskr) or 'mkdir' (dskw) sign
25695 000074E2 <res 00000001> <1> u.brwdev: resb 1 ; Block device number for direct I/O (bread & bwrite)
25696 <1> ; 24/07/2015 - 24/06/2015
25697 <1> ;u.args: resd 1 ; arguments list (line) offset from start of [u.upage]
25698 <1> ; (arg list/line is from offset [u.args] to 4096 in [u.upage])
25699 <1> ; ([u.args] points to argument count -argc- address offset)
25700 <1> ; 24/06/2015
25701 <1> ;u.core: resd 1 ; physical start address of user's memory space (for sys exec)
25702 <1> ;u.ecore: resd 1 ; physical end address of user's memory space (for sys exec)
25703 <1> ; 21/09/2015 (debugging - page fault analyze)
25704 000074E3 <res 00000004> <1> u.pfcount: resd 1 ; page fault count for (this) process (for sys geterr)
25705 <1>
25706 000074E7 <res 00000001> <1> alignb 4
25707 <1>
25708 <1> U_SIZE equ $ - user
25709 <1>
25710 <1> ; 18/10/2015 - Retro UNIX 386 v1 (local variables for 'namei' and 'sysexec')
25711 000074E8 <res 00000004> <1> pcore: resd 1 ; physical start address of user's memory space (for sys exec)
25712 000074EC <res 00000004> <1> ecore: resd 1 ; physical start address of user's memory space (for sys exec)
25713 000074F0 <res 00000004> <1> nbase: resd 1 ; physical base address for 'namei' & 'sysexec'
25714 000074F4 <res 00000002> <1> ncount: resw 1 ; remain byte count in page for 'namei' & 'sysexec'
25715 000074F6 <res 00000002> <1> argc: resw 1 ; argument count for 'sysexec'
25716 000074F8 <res 00000004> <1> argv: resd 1 ; argument list (recent) address for 'sysexec'
25717 <1>
25718 <1> ; 03/06/2015 - Retro UNIX 386 v1 Beginning
25719 <1> ; 07/04/2013 - 31/07/2013 - Retro UNIX 8086 v1
25720 000074FC <res 00000001> <1> rw: resb 1 ;; Read/Write sign (iget)
25721 000074FD <res 00000001> <1> rwdsk: resb 1 ;; Read/Write function number (diskio) - 16/06/2015
25722 000074FE <res 00000001> <1> retry_count: resb 1 ; Disk I/O retry count - 11/06/2015
25723 000074FF <res 00000001> <1> resb 1 ;; Reserved (16/06/2015)
25724 <1>

```



25725		<1> ;alignb 4
25726		<1>
25727		<1> ; 22/08/2015
25728	00007500 <res 00000820>	<1> buffer: resb nbuf * 520
25729		<1>
25730	00007D20 <res 00000008>	<1> sb0: resd 2
25731		<1> ;s:
25732		<1> ; (root disk) super block buffer
25733		<1> systm:
25734		<1> ; 13/11/2015 (Retro UNIX 386 v1)
25735		<1> ; 11/03/2013.
25736		<1> ;Derived from UNIX v1 source code 'systm' structure (ux).
25737		<1> ;s.
25738		<1>
25739	00007D28 <res 00000002>	<1> resw 1
25740	00007D2A <res 00000168>	<1> resb 360 ; 2880 sectors ; original UNIX v1 value: 128
25741	00007E92 <res 00000002>	<1> resw 1
25742	00007E94 <res 00000020>	<1> resb 32 ; 256+40 inodes ; original UNIX v1 value: 64
25743	00007EB4 <res 00000004>	<1> s.time: resd 1
25744	00007EB8 <res 00000004>	<1> s.syst: resd 1
25745	00007EBC <res 00000004>	<1> s.wait_: resd 1 ; wait
25746	00007EC0 <res 00000004>	<1> s.idlet: resd 1
25747	00007EC4 <res 00000004>	<1> s.chrgt: resd 1
25748	00007EC8 <res 00000002>	<1> s.drerr: resw 1
25749		<1>
25750		<1> S_SIZE equ \$ - systm
25751		<1>
25752	00007ECA <res 0000005E>	<1> resb 512-S_SIZE ; 03/06/2015
25753		<1>
25754	00007F28 <res 00000008>	<1> sb1: resd 2
25755		<1> ; (mounted disk) super block buffer
25756		<1> mount:
25757	00007F30 <res 00000200>	<1> resb 512 ; 03/06/2015
25758		<1>
25759		<1> ;/ ux -- unix
25760		<1> ;
25761		<1> ;systm:
25762		<1> ;
25763		<1> ; .+.2
25764		<1> ; .+.128.
25765		<1> ; .+.2
25766		<1> ; .+.64.
25767		<1> ;s.time: .+.4
25768		<1> ;s.syst: .+.4
25769		<1> ;s.wait: .+.4
25770		<1> ;s.idlet: .+.4
25771		<1> ;s.chrgt: .+.4
25772		<1> ;s.drerr: .+.2
25773		<1> ;inode:
25774		<1> ;i.flgs: .+.2
25775		<1> ;i.nlks: .+.1
25776		<1> ;i.uid: .+.1
25777		<1> ;i.size: .+.2
25778		<1> ;i.dskp: .+.16.
25779		<1> ;i.ctim: .+.4
25780		<1> ;i.mtim: .+.4
25781		<1> ;. = inode+32.
25782		<1> ;mount: .+.1024.
25783		<1> ;proc:
25784		<1> ;p.pid: .+.[2*nproc]
25785		<1> ;p.dska: .+.[2*nproc]
25786		<1> ;p.ppid: .+.[2*nproc]
25787		<1> ;p.break: .+.[2*nproc]
25788		<1> ;p.link: .+.nproc
25789		<1> ;p.stat: .+.nproc
25790		<1> ;tty:
25791		<1> ;. = .+[ntty*8.]
25792		<1> ;fsp: .+.[nfiles*8.]
25793		<1> ;bufp: .+.[nbuf*2]+6
25794		<1> ;sb0: .+.8
25795		<1> ;sb1: .+.8
25796		<1> ;swp: .+.8
25797		<1> ;ii: .+.2
25798		<1> ;idev: .+.2
25799		<1> ;cdev: .+.2
25800		<1> ;deverr: .+.12.
25801		<1> ;active: .+.2
25802		<1> ;rfap: .+.2
25803		<1> ;rkap: .+.2
25804		<1> ;tcap: .+.2
25805		<1> ;tcstate: .+.2
25806		<1> ;tcerrc: .+.2
25807		<1> ;mnti: .+.2
25808		<1> ;mntd: .+.2
25809		<1> ;mpid: .+.2
25810		<1> ;clockp: .+.2
25811		<1> ;rootdir: .+.2
25812		<1> ;toutt: .+.16.
25813		<1> ;touts: .+.32.
25814		<1> ;runq: .+.6
25815		<1> ;
25816		<1> ;wlist: .+.40.

```

25817 <1> ;cc:      .+.30.
25818 <1> ;cf:      .+.31.
25819 <1> ;cl:      .+.31.
25820 <1> ;clist:   .+.510.
25821 <1> ;imod:    .+.1
25822 <1> ;smod:    .+.1
25823 <1> ;mmod:    .+.1
25824 <1> ;uquant:  .+.1
25825 <1> ;sysflg:  .+.1
25826 <1> ;pptiflg: .+.1
25827 <1> ;ttyoch:  .+.1
25828 <1> ; .even
25829 <1> ; .+.100.; sstack:
25830 <1> ;buffer:  .+. [ntty*140.]
25831 <1> ; .+. [nbuf*520.]
25832 <1> ;
25833 <1> ; . = core-64.
25834 <1> ;user:
25835 <1> ;u.sp:    .+.2
25836 <1> ;u.usp:   .+.2
25837 <1> ;u.r0:    .+.2
25838 <1> ;u.cdir:  .+.2
25839 <1> ;u.fp:    .+.10.
25840 <1> ;u.fofp: .+.2
25841 <1> ;u.dirp:  .+.2
25842 <1> ;u.namep: .+.2
25843 <1> ;u.off:   .+.2
25844 <1> ;u.base:  .+.2
25845 <1> ;u.count: .+.2
25846 <1> ;u.nread: .+.2
25847 <1> ;u.break: .+.2
25848 <1> ;u.ttyp:  .+.2
25849 <1> ;u.dirbuf: .+.10.
25850 <1> ;u.pri:   .+.2
25851 <1> ;u.intr:  .+.2
25852 <1> ;u.quit:  .+.2
25853 <1> ;u.emt:   .+.2
25854 <1> ;u.ilgins: .+.2
25855 <1> ;u.cdev:  .+.2
25856 <1> ;u.uid:   .+.1
25857 <1> ;u.ruid:  .+.1
25858 <1> ;u.bsys:  .+.1
25859 <1> ;u.uno:   .+.1
25860 <1> ;. = core
25861
25862 ;; Memory (swap) Data (11/03/2015)
25863 ; 09/03/2015
25864 00008130 <res 00000002> swpq_count: resw 1 ; count of pages on the swap que
25865 00008132 <res 00000004> swp_drv:    resd 1 ; logical drive description table address of the swap
drive/disk
25866 00008136 <res 00000004> swpd_size:  resd 1 ; size of swap drive/disk (volume) in sectors (512 bytes).

25867 0000813A <res 00000004> swpd_free:  resd 1 ; free page blocks (4096 bytes) on swap disk/drive (logical)
25868 0000813E <res 00000004> swpd_next:  resd 1 ; next free page block
25869 00008142 <res 00000004> swpd_last:  resd 1 ; last swap page block
25870
25871 00008146 <res 00000002> alignb 4
25872
25873 ; 10/07/2015
25874 ; 28/08/2014
25875 00008148 <res 00000004> error_code: resd 1
25876 ; 29/08/2014
25877 0000814C <res 00000004> FaultOffset: resd 1
25878 ; 21/09/2015
25879 00008150 <res 00000004> PF_Count:   resd 1 ; total page fault count
25880 ; (for debugging - page fault analyze)
25881 ; 'page_fault_handler' (memory.inc)
25882 ; 'sysgeterr' (u9.s)
25883 ;; 21/08/2015
25884 ;;buffer: resb (nbuf*520) ;; sysdefs.inc, ux.s
25885 ;; ((NOTE: nbuf = 6, buffer r/w problem/bug here !? when nbuf > 4))
25886
25887 bss_end:
25888
25889 ; 27/12/2013
25890 _end: ; end of kernel code (and read only data, just before bss)

```