```
; ****************************************************************************
;
; UNIX.ASM (RETRO UNIX 8086 Kernel - Only for 1.44 MB floppy disks)
; ----------------------------------------------------------------------------
; U8.ASM (include u8.asm) //// UNIX v1 -> u8.s

; RETRO UNIX 8086 (Retro Unix == Turkish Rational Unix)
; Operating System Project (v0.1) by ERDOGAN TAN (Beginning: 11/07/2012)
; 1.44 MB Floppy Disk
; (13/03/2013)
;
; [ Last Modification: 18/01/2014 ] ;;; completed ;;;
;
; Derivation from UNIX Operating System (v1.0 for PDP-11)
; (Original) Source Code by Ken Thompson (1971-1972)
; <Bell Laboratories (17/3/1972)>
; <Preliminary Release of UNIX Implementation Document>
;
; ****************************************************************************

; 18/01/2014
; 03/08/2013 dskwr
; 31/07/2013
; 29/07/2013
; 26/07/2013 bread, bwrite (bug) note
; 23/07/2013 poke
; 20/07/2013 poke, bufaloc, bread, bwrite, dskrd, dskwr, wslot
; 17/07/2013 poke
; 09/07/2013 bufaloc, poke
; 26/04/2013 device number modifications (cdev/0/1 -> 0/rdev, 1/mdev -> drv)
; 18/04/2013
; 24/03/2013 poke
; 15/03/2013 poke, diskio (runix)
; 14/03/2013
; 13/03/2013

;; I/O Buffer ((8+512 bytes in original Unix v1))
;;           ((4+512 bytes in Retro UNIX 8086 v1))
;;
;; I/O Queue Entry (of original UNIX operating system v1)
;; Word 1, Byte 0 = device id
;; Word 1, Byte 1 = (bits 8 to 15)
;;          bit  9 = write bit
;;          bit 10 = read bit
;;          bit 12 = waiting to write bit
;;          bit 13 = waiting to read bit
;;          bit 15 = inhibit bit
;; Word 2 = physical block number (In fact, it is LBA for Retro UNIX 8086 v1)
;;
;; Original UNIX v1 ->
;;            Word 3 = number of words in buffer (=256)
;; Original UNIX v1 ->
;;            Word 4 = bus address (addr of first word of data buffer)
;;
;; Retro UNIX 8086 v1 -> Buffer Header (I/O Queue Entry) size is 4 bytes !
;;
;; Device IDs (of Retro Unix 8086 v1)
;;          0 = fd0
;;          1 = fd1
;;          2 = hd0
;;          3 = hd1
;;          4 = hd2
;;          5 = hd3

rfd:    ; 26/04/2013
        ; 13/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
        ;sub    ax, 3 ; zero based device number (Floppy disk)
        mov     cx, 2880 ; size of floppy disks (1.44 MB)
        call    bread ; **** returns to routine that called readi ('jmp ret')
wfd:     ; 26/04/2013
        ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
        ;sub    ax, 3 ; zero based device number (Hard disk)
        mov     cx, 2880 ; size of floppy disks (1.44 MB)
        call    bwrite ; **** returns to routine that called writei ('jmp ret')
rhd:    ; 26/04/2013
        ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
        ;sub    ax, 3 ; zero based device number (Hard disk)
        mov     cx, 0FFFFh ; size of fixed disks (32 MB, first 65535 sectors)
        call    bread ; **** returns to routine that called readi ('jmp ret')
```

```
whd:
        ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
        ;sub    ax, 3 ; zero based device number (Hard disk)
        mov     cx, 0FFFFh ; size of fixed disks (32 MB, first 65535 sectors)
        call    bwrite ; **** returns to routine that called writei ('jmp ret')

bread:
        ; 29/07/2013
        ; 20/07/2013
        ; 26/04/2013 Retro Unix 8086 v1 feature (device number) modifications
        ; 14/03/2013
        ; 13/03/2013 Retro UNIX 8086 v1 modification on original unix code
        ;; / read a block from a block structured device
        ;
        ; INPUTS ->
        ;    [u.fopf] points to the block number
        ;    CX = maximum block number allowed on device
        ;        ; that was an arg to bread, in original Unix v1, but
        ;        ; CX register is used instead of arg in Retro Unix 8086 v1
        ;    [u.count] number of bytes to read in
        ; OUTPUTS ->
        ;    [u.base] starting address of data block or blocks in user area
        ;    [u.fopf] points to next consecutive block to be read
        ;
        ; ((Modified registers: DX, CX, BX, SI, DI, BP))
        ;
        ; NOTE: Original UNIX v1 has/had a defect/bug here, even if read
        ;        byte count is less than 512, block number in *u.fofp (u.off)
        ;       is increased by 1. For example: If user/program request
        ;        to read 16 bytes in current block, 'sys read' increaces
        ;       the next block number just as 512 byte reading is done.
        ;        This wrong is done in 'bread'. So, in Retro UNIX 8086 v1,
        ;        for user (u) structure compatibility (because 16 bit is not
        ;        enough to keep byte position/offset of the disk), this
        ;       defect will not be corrected, user/program must request
        ;       512 byte read per every 'sys read' call to block devices
        ;        for achieving correct result. In future version(s),
        ;       this defect will be corrected by using different
        ;        user (u) structure.  26/07/2013 - Erdogan Tan

                ; jsr r0,tstdeve / error on special file I/O
                               ; / (only works on tape)
                ; mov *u.fofp,r1 / move block number to r1
                ; mov $2.-cold,-(sp) / "2-cold" to stack
; 1:
                ; cmp r1,(r0) / is this block # greater than or equal to
                            ; / maximum block # allowed on device
                ; jnb short @f
                ; bhis  1f / yes, 1f (error)
                ; mov r1,-(sp) / no, put block # on stack
                ; jsr r0,preread / read in the block into an I/O buffer
                ; mov (sp)+,r1 / return block # to r1
                ; inc r1 / bump block # to next consecutive block
                ; dec (sp) / "2-1-cold" on stack
                ; bgt 1b / 2-1-cold = 0?  No, go back and read in next block
;1:
                ; tst (sp)+ / yes, pop stack to clear off cold calculation
        push    cx ; **
        ;26/04/2013
        ;sub    ax, 3 ; 3 to 8 -> 0 to 5
        sub     al, 3
                ; AL = Retro Unix 8086 v1 disk (block device) number
        mov     di, offset brwdev ; block device number for direct I/O
        mov     byte ptr [DI], al
        ;; 20/07/2013
        ;;xor   dx, dx ; 0 is needed for bufaloc_0
        ;
        mov     bx, word ptr [u.fofp]
        mov     ax, word ptr [BX]
                ; mov *u.fofp,r1 / restore r1 to initial value of the
                               ; / block #
        cmp     ax, cx
                ; cmp r1,(r0)+ / block # greater than or equal to maximum
                              ; / block number allowed
        jnb     error        ; 18/04/2013
                ; bhis error10 / yes, error
        inc     word ptr [BX]
                ; inc *u.fofp / no, *u.fofp has next block number
        ; AX = Block number (zero based)
                ;;jsr r0,preread / read in the block whose number is in r1
```

```
preread: ;; call preread
        call    bufaloc_0 ; 26/04/2013
        ;; jc   error
        ; BX = Buffer (Header) Address (r5) (ES=CS=DS, system/kernel segment)
         ; AX = Block/Sector number (r1)
                ; jsr r0,bufaloc / get a free I/O buffer (r1 has block number)
        ; 14/03/2013
         jz     short @f ; Retro UNIX 8086 v1 modification
                ; br 1f / branch if block already in a I/O buffer
        or      word ptr [BX], 400h ; set read bit (10) in I/O Buffer
                ; bis $2000,(r5) / set read bit (bit 100 in I/O buffer)
        call    poke
                ; jsr r0,poke / perform the read
        ;;jc    error ;2 0/07/2013
; 1:
                ; clr *$ps / ps = 0
                ; rts r0
;; return from of preread
@@:
        or      word ptr [BX], 4000h
                ; bis $40000,(r5)
                        ; / set bit 14 of the 1st word of the I/O buffer
@@: ; 1:
        test    word ptr [BX], 2400h
                ; bit $22000,(r5) / are 10th and 13th bits set (read bits)
        jz      short @f
                ; beq 1f / no
                ; cmp cdev,$1 / disk or drum?
                ; ble 2f / yes
                ; tstb uquant / is the time quantum = 0?
                ; bne 2f / no, 2f
                ; mov r5,-(sp) / yes, save r5 (buffer address)
                ; jsr r0,sleep; 31.
                        ; / put process to sleep in channel 31 (tape)
                ; mov (sp)+,r5 / restore r5
                ; br 1b / go back
;@@: ; 2: / drum or disk
         ;; mov     cx, word ptr [s.wait_]+2 ;; 29/07/2013
        call    idle
                ; jsr r0,idle; s.wait+2 / wait
        jmp     short @b
                ; br 1b
@@: ; 1: / 10th and 13th bits not set
        and     word ptr [BX], 0BFFFh ; 1011111111111111b
                ; bic $40000,(r5) / clear bit 14
                ; jsr r0,tstdeve / test device for error (tape)
        ;add    bx, 8
        ; 26/04/2013
        add     bx, 4 ; Retro Unix 8086 v1 modification !
                ; add $8,r5 / r5 points to data in I/O buffer
        ; BX = system (I/O) buffer address
        call    dioreg
                ; jsr r0,dioreg / do bookkeeping on u.count etc.
        ; AX =  [u.base] value before it gets updated
        ; CX =  Byte count to transfer
        ; BX is not changed in dioreg
;1: / r5 points to beginning of data in I/O buffer, r2 points to beginning
;   / of users data
        mov     si, bx
        mov     di, ax
        mov      ax, word ptr [u.segmnt]
                ; Retro Unix 8086 v1 feature only
        mov     es, ax
        rep     movsb
        mov     ax, ds
        mov     es, ax
                ; movb (r5)+,(r2)+ / move data from the I/O buffer
                ; dec r3 / to the user's area in core starting at u.base
                ; bne 1b
        pop     cx ; **
        cmp     word ptr [u.count], 0
                ; tst u.count / done
        jna     short @f
                ; beq 1f / yes, return
                ; tst -(r0) / no, point r0 to the argument again
        jmp     short bread
                ; br bread / read some more
@@: ; 1:
        pop     ax ; ****
```

```
                ; mov (sp)+,r0
         jmp      ret_
                ;jmp ret  / jump to routine that called readi

bwrite: ; 20/07/2013
        ; 26/04/2013 Retro Unix 8086 v1 feature (device number) modifications
        ; 14/03/2013
        ;; / write on block structured device
        ; INPUTS ->
        ;    [u.fopf] points to the block number
        ;    CX = maximum block number allowed on device
        ;       ; that was an arg to bwrite, in original Unix v1, but
        ;       ; CX register is used instead of arg in Retro Unix 8086 v1
        ;    [u.count] number of bytes to user desires to write
        ; OUTPUTS ->
        ;    [u.fopf] points to next consecutive block to be written into
        ;
        ; ((Modified registers: DX, CX, BX, SI, DI, BP))
        ;
        ; NOTE: Original UNIX v1 has/had a defect/bug here, even if write
        ;       byte count is less than 512, block number in *u.fofp (u.off)
        ;       is increased by 1. For example: If user/program request
        ;       to write 16 bytes in current block, 'sys write' increaces
        ;       the next block number just as 512 byte writing is done.
        ;       This wrong is done in 'bwrite'. So, in Retro UNIX 8086 v1,
        ;       for user (u) structure compatibility (because 16 bit is not
        ;       enough to keep byte position/offset of the disk), this
        ;       defect will not be corrected, user/program must request
        ;       512 byte write per every 'sys write' call to block devices
        ;       for achieving correct result. In future version(s),
        ;       this defect will be corrected by using different
        ;       user (u) structure.  26/07/2013 - Erdogan Tan

                ; jsr r0,tstdeve / test the device for an error
        push    cx ; **
        ;26/04/2013
        ;sub    ax, 3 ; 3 to 8 -> 0 to 5
        sub     al, 3
                ; AL = Retro Unix 8086 v1 disk (block device) number
        mov     di, offset brwdev ; block device number for direct I/O
        mov     byte ptr [DI], al
        ;; 20/07/2013
        ;;xor   dx, dx ; 0 is needed for bufaloc_0
        ;
        mov     bx, word ptr [u.fofp]
        mov     ax, word ptr [BX]
                ; mov *u.fofp,r1 / put the block number in r1
        cmp     ax, cx
                ; cmp r1,(r0)+ / does block number exceed maximum allowable #
                              ; / block number allowed
        jnb     error        ; 18/04/2013
                ; bhis error10 / yes, error
        inc     word ptr [BX]
                ; inc *u.fofp / no, increment block number
        call    bwslot ; 26/04/2013 (wslot -> bwslot)
                ; jsr r0,wslot / get an I/O buffer to write into
         call   dioreg
                ; jsr r0,dioreg / do the necessary bookkeeping
        ; AX = [u.base] before it gets updated
        ; CX = byte count
        ; BX is not changed
; 1: / r2 points to the users data; r5 points to the I/O buffers data area
        mov     di, bx ; system (I/O) buffer (data) address
        mov     si, ax ; beginning of user data
         mov     ax, word ptr [u.segmnt]
                ; Retro Unix 8086 v1 feature only
        mov     ds, ax
        rep     movsb
        mov     ax, cs
        mov     ds, ax
                ; movb (r2)+,(r5)+ / ; r3, has the byte count
                ; dec r3 / area to the I/O buffer
                ; bne 1b
        call    dskwr
                ; jsr r0,dskwr / write it out on the device
        pop     cx ; **
         cmp     word ptr [u.count], 0
                ; tst u.count / done
        jna     short @f
```

```
		; beq 1f / yes, 1f
		; tst -(r0) / no, point r0 to the argument of the call
	jmp	short bwrite
		; br bwrite / go back and write next block
@@: ; 1:
	pop	ax ; ****
		; mov (sp)+,r0
	 jmp	  ret_
		; jmp ret / return to routine that called writei
;error10:
;	 jmp	  error	 ; / see 'error' routine

dioreg:
	; 14/03/2013
	; bookkeeping on block transfers of data
	;
	; returns value of u.base before it gets updated, in AX (r2)
	; returns byte count (to transfer) in CX (<=512)

	mov	cx, word ptr [u.count]
		; mov u.count,r3 / move char count to r3
	cmp	cx, 512
		; cmp r3,$512. / more than 512. char?
	jna	short @f
		; blos 1f / no, branch
	mov	cx, 512
		; mov $512.,r3 / yes, just take 512.
@@: ; 1:
	mov	ax, word ptr [u.base]
		 ; mov u.base,r2 / put users base in r2
	add	word ptr [u.nread], cx
		; add r3,u.nread / add the number to be read to u.nread
	sub	word ptr [u.count], cx
		; sub r3,u.count / update count
	add	word ptr [u.base], cx
		; add r3,u.base / update base
	 retn
		; rts r0 / return
dskrd:
	; 29/07/2013
	; 20/07/2013, 26/04/2013, 14/03/2013
	;
	; 'dskrd' acquires an I/O buffer, puts in the proper
	; I/O queue entries (via bufaloc) then reads a block
	; (number specified in r1) in the acquired buffer.)
	; If the device is busy at the time dskrd is called,
	; dskrd calls idle.
	;
	; INPUTS ->
	;    r1 - block number
	;    cdev - current device number
	; OUTPUTS ->
	;    r5 - points to first data word in I/O buffer
	;
	; ((AX = R1)) input/output
	; ((BX = R5)) output
	;
	; ((Modified registers: DX, CX, BX, SI, DI, BP))
	;
	call	bufaloc
		; jsr r0,bufaloc / shuffle off to bufaloc;
		               ; / get a free I/O buffer
	;;jc	error ; 20/07/2013
	jz	short @f ; Retro UNIX 8086 v1 modification
		; br 1f / branch if block already in a I/O buffer
	or	word ptr [BX], 400h ; set read bit (10) in I/O Buffer
		; bis $2000,(r5) / set bit 10 of word 1 of
		                ; / I/O queue entry for buffer
	call	poke
		; jsr r0,poke / just assigned in bufaloc,
		            ; /bit 10=1 says read
	;;jc	error ; 20/07/2013
@@: ; 1:
		;clr *$ps
	test	word ptr [BX], 2400h
		; bit $22000,(r5) / if either bits 10, or 13 are 1;
		; jump to idle
	jz	short @f
		; beq 1f
```

```
        ;; mov     cx, word ptr [s.wait_]+2 ;; 29/07/2013
        call    idle
                ; jsr r0,idle; s.wait+2
        jmp short @b
                ; br 1b
@@: ; 1:
        ;add   bx, 8
        ; 26/04/2013
        add     bx, 4 ; Retro Unix 8086 v1 modification !
                ; add $8,r5 / r5 points to first word of data in block
                          ; / just read in
        retn
                ; rts r0

bwslot:
        ; 26/04/2013
        ; Retro UNIX 8086 v1 modification !
        ; ('bwslot' will be called from 'bwrite' only!)
        ; INPUT -> DI - points to device id (in bwdev)
        ;       -> AX = block number
        ;
        call    bufaloc_0
        ;; jc   error
        jmp     short @f

wslot:
        ; 29/07/2013
        ; 20/07/2013
        ; 26/04/2013
        ; 14/03/2013
        ;
        ; 'wslot' calls 'bufaloc' and obtains as a result, a pointer
        ; to the I/O queue of an I/O buffer for a block structured
        ; device. It then checks the first word of I/O queue entry.
        ; If bits 10 and/or 13 (read bit, waiting to read bit) are set,
        ; wslot calls 'idle'. When 'idle' returns, or if bits 10
        ; and/or 13 are not set, 'wslot' sets bits 9 and 15 of the first
        ; word of the I/O queue entry (write bit, inhibit bit).
        ;
        ; INPUTS ->
        ;    r1 - block number
        ;    cdev - current (block/disk) device number
        ;
        ; OUTPUTS ->
        ;    bufp - bits 9 and 15 are set,
        ;           the remainder of the word left unchanged
        ;    r5 - points to first data word in I/O buffer
        ;
        ; ((AX = R1)) input/output
        ; ((BX = R5)) output
        ;
        ; ((Modified registers: DX, CX, BX, SI, DI, BP))

        call    bufaloc
                ; jsr r0,bufaloc / get a free I/O buffer; pointer to first
        ;;jc    error ; 20/07/2013
        ; BX = Buffer (Header) Address (r5) (ES=CS=DS, system/kernel segment)
        ; AX = Block/Sector number (r1)
        ; jz short @f
                ; br 1f / word in buffer in r5
@@: ;1:
        test    word ptr [BX], 2400h
                ; bit $22000,(r5) / check bits 10, 13 (read, waiting to read)
                               ; / of I/O queue entry
        jz      short @f
                 ; beq 1f   / branch if 10, 13 zero (i.e., not reading,
                     ; / or not waiting to read)

        ;; mov     cx, word ptr [s.wait_]+2 ; 29/07/2013
        call    idle
                ; jsr r0,idle; / if buffer is reading or writing to read,
                              ; / idle
        jmp     short @b
                ; br 1b / till finished
@@: ;1:
        or       word ptr [BX], 8200h
                ; bis $101000,(r5) / set bits 9, 15 in 1st word of I/O queue
                                ; / (write, inhibit bits)
                ; clr     *$ps / clear processor status
        ;add   bx, 8
        ; 26/04/2013
        add     bx, 4 ; Retro Unix 8086 v1 modification !
                ; add $8,r5 / r5 points to first word in data area
                          ; / for this block
        retn
                ; rts r0
dskwr:
        ; 03/08/2013
        ; 31/07/2013
        ; 20/07/2013
        ; 26/04/2013
        ; 14/03/2013
        ;
        ; 'dskwr' writes a block out on disk, via ppoke. The only
        ; thing dskwr does is clear bit 15 in the first word of I/O queue
        ; entry pointed by 'bufp'. 'wslot' which must have been called
        ; previously has supplied all the information required in the
        ; I/O queue entry.
        ;
        ; (Modified registers: CX, DX, BX, SI, DI)
        ;
        ;
        ; 03/08/2013 (si -> bx)
        mov     bx, word ptr [bufp]
        and     word ptr [bx], 7FFFh ; 0111111111111111b
                ; bic $100000,*bufp / clear bit 15 of I/O queue entry at
                                   ; / bottom of queue
ppoke:
                ; mov $340,*$ps
                ; jsr r0,poke
                ; clr *$ps
                ; rts r0
poke:
        ; 18/01/2014
        ; 31/07/2013
        ; 23/07/2013
        ; 20/07/2013
        ; 17/07/2013
        ; 09/07/2013
        ; 26/04/2013
        ; 24/03/2013 AX (r1) -> push/pop (to save physical block number)
        ; 15/03/2013
        ; (NOTE: There are some disk I/O code modifications & extensions
        ; & exclusions on original 'poke' & other device I/O procedures of
        ; UNIX v1 OS for performing disk I/O functions by using IBM PC
        ; compatible rombios calls in Retro UNIX 8086 v1 kernel.)
        ;
        ; Basic I/O functions for all block structured devices
         ; (Modified registers: CX, DX, SI, DI)

        ; 20/07/2013 modifications
        ;            (Retro UNIX 8086 v1 features only !)
        ; INPUTS ->
        ;        (BX = buffer header address)
        ; OUTPUTS ->
        ;        cf=0 -> successed r/w (at least, for the caller's buffer)
        ;        cf=1 -> error, word ptr [BX] = 0FFFFh
        ;              (drive not readi or r/w error!)
        ;        (word ptr [BX]+2 <> 0FFFFh indicates r/w success)
        ;        (word ptr [BX]+2 = FFFFh mean RW/IO error)
        ;         (also it indicates invalid buffer data)

        ; 17/07/2013
        push    bx
        ; 24/03/2013
                ; mov r1,-(sp)
                ; mov r2,-(sp)
                ; mov r3,-(sp)
        push    ax ; Physical Block Number (r1) (mget)
        ;mov    si, offset bufp + nbuf + nbuf + 6
                ; mov $bufp+nbuf+nbuf+6,r2 / r2 points to highest priority
                                       ; / I/O queue pointer
         mov     si, offset bufp + (2*nbuf) + (2*2)  ; 09/07/2013
poke_1: ; 1:
         dec    si
        dec     si
        mov     bx, word ptr [SI]
                ; mov -(r2),r1 / r1 points to an I/O queue entry
        mov     ax, word ptr [BX] ; 17/07/2013
        test    ah, 06h
        ;test   word ptr [BX], 600h ; 0000011000000000b
                ; bit $3000,(r1) / test bits 9 and 10 of word 1 of I/O
                               ; / queue entry
         jz      short poke_2
                ; beq 2f / branch to 2f if both are clear
        ; 31/07/2013
        ;test   ah, 0B0h ; (*)
        ;;test  word ptr [BX], 0B000h ; 1011000000000000b
                ; bit $130000,(r1) / test bits 12, 13, and 15
         ;jnz     short poke_2 ; 31/07/2013 (*)
                ; bne 2f / branch if any are set
        mov     cl, byte ptr [BX] ; 26/04/2013 ; Device Id
                ; movb (r1),r3 / get device id
        xor     ch, ch ; mov ch, 0 ; 26/04/2013
        ;mov    di, cx ; 26/04/2013
        xor     ax, ax ; 0
        ;cmp    byte ptr [DI]+drv.err, al ; 0 ; 26/04/2013
                ; tstb deverr(r3) / test for errors on this device
        ;jna    short poke_3
                ; beq 3f / branch if no errors
        ; 20/07/2013
        ;dec    ax
        ;mov    word ptr [BX]+2, ax ; FFFFh ; -1
                ; mov $-1,2(r1) / destroy associativity
        ;inc    ah ; 0
        ;mov    word ptr [BX], ax ; 00FFh, reset
                ; clrb 1(r1) / do not do I/O
        ;jmp     short poke_2
         ;        ; br 2f
                 ; rts r0
poke_3: ; 3:
        ; 26/04/2013 Modification
        inc     al ; mov ax, 1
        or      cl, cl ; Retro UNIX 8086 v1 device id.
        jz      short @f ; cl = 0
        shl     al, cl ; shl ax, cl
@@::
        ;test   word ptr [active], ax
        test    byte ptr [active], al
                ; bit $2,active / test disk busy bit
         jnz     short poke_2
                ; bne 2f / branch if bit is set
        ;or     word ptr [active], ax
        or      byte ptr [active], al
                ; bis $2,active / set disk busy bit
        push    ax ; 17/07/2013
        call    diskio ; Retro UNIX 8086 v1 Only !
        mov      byte ptr [DI]+drv.err, ah
        pop     ax
        jnc     short @f ; 20/07/2013
                ; tstb deverr(r3) / test for errors on this device
                ; beq 3f / branch if no errors
        ; 20/07/2013
        mov     word ptr [BX]+2, 0FFFFh ; -1
                ; mov $-1,2(r1) / destroy associativity
        mov     byte ptr [BX]+1, 0
                ; clrb 1(r1) / do not do I/O
        jmp      short poke_2
@@:     ; 20/07/2013
        ; 17/07/2013
        not     al
        and     byte ptr [active], al ; reset, not busy
        ; BX = system I/O buffer header (queue entry) address
seta: ; / I/O queue bookkeeping; set read/write waiting bits.
        mov     ax, word ptr [BX]
                ; mov (r1),r3 / move word 1 of I/O queue entry into r3
         and    ax, 600h
                ; bic $!3000,r3 / clear all bits except 9 and 10
        and     word ptr [BX], 0F9FFh
                ; bic $3000,(r1) / clear only bits 9 and 10
        ;shl    ax, 1
        ;shl    ax, 1
        ;shl    ax, 1
                ; rol r3
                  ; rol r3
                  ; rol r3
        ; 23/07/2013
        shl     ah, 1
        shl     ah, 1
        shl     ah, 1
        or      word ptr [BX], ax
                ; bis r3,(r1) / or old value of bits 9 and 10 with
                          ; bits 12 and 13
        call    idle ; 18/01/2014
        ;; sti
        ;hlt    ; wait for a hardware interrupt
        ;; cli
        ; NOTE: In fact, disk controller's 'disk I/O completed'
         ; interrupt would be used to reset busy bits, but INT 13h
        ; returns when disk I/O is completed. So, here, as temporary
        ; method, this procedure will wait for a time according to
        ; multi tasking and time sharing concept.
        not     ax
        and     word ptr [BX], ax ; clear bits 12 and 13
poke_2: ;2:
         cmp     si, offset bufp
                ; cmp r2,$bufp / test to see if entire I/O queue
                             ; / has been scanned
         ja      short poke_1
                ; bhi 1b
        ; 24/03/2013
                ; mov (sp)+,r3
                ; mov (sp)+,r2
                ; mov (sp)+,r1
         pop    ax  ; Physical Block Number (r1) (mget)
        ; 17/07/2013
        pop     bx
        ; 20/07/2013
        cmp     word ptr [BX]+2, 0FFFFh
        je      error
        ; 'poke' returns with cf=0 if the requested buffer is read
        ; or written succesfully; even if an error occurs while
        ; reading to or writing from other buffers. 20/07/2013
        ;
        ;cmc
        retn
                 ; rts r0

bufaloc:
        ; 29/07/2013
        ; 20/07/2013
        ; 09/07/2013
        ; 26/04/2013 (device number/id modifications)
        ; 13/03/2013
        ; bufaloc - Block device I/O buffer allocation
        ;
        ; INPUTS ->
        ;    r1 - block number
        ;    cdev - current (block/disk) device number
        ;    bufp+(2*n)-2 --- n = 1 ... nbuff
        ; OUTPUTS ->
        ;    r5 - pointer to buffer allocated
        ;    bufp ... bufp+12 --- (bufp), (bufp)+2
        ;
        ; ((AX = R1)) input/output
        ; ((BX = R5)) output
         ;    ((Modified registers: DX, CX, BX, SI, DI, BP))
        ;    zf=1 -> block already in a I/O buffer
        ;    zf=0 -> a new I/O buffer has been allocated
        ;    ((DL = Device ID))
        ;    (((DH = 0 or 1)))
        ;    (((CX = previous value of word ptr [bufp])))
        ;    ((CX and DH will not be used after return)))

        ;;push  si ; ***
                ; mov r2,-(sp) / save r2 on stack
                ; mov $340,*$ps / set processor priority to 7
        ; 20/07/2013
        ; 26/04/2013
        xor     bh, bh
        mov     bl, byte ptr [cdev] ; 0 or 1
        mov     di, offset rdev  ; offset mdev = offset rdev + 1
        add     di, bx
```

```
bufaloc_0: ; 26/04/2013 !! here is called from bread or bwrite !!
                   ;; DI points to device id.
       ; 20/07/2013
       mov     bl, byte ptr [DI] ; DI -> rdev/mdev or brwdev
       xor     bh, bh
       cmp     byte ptr [BX]+drv.pdn, 0FFh ; Drive not ready !
       je      error ; 20/07/2013
@@:
       mov     dx, bx  ; dh = 0, dl = device number (0 to 5)
       xor     bp, bp ; 0
       push    bp ; 0
        mov     bp, sp
       ;
bufaloc_1: ;1:
               ; clr -(sp) / vacant buffer
        mov    si, offset bufp
               ; mov $bufp,r2 / bufp contains pointers to I/O queue
                           ; / entrys in buffer area
bufaloc_2: ;2:
       mov     bx, word ptr [SI]
       inc     si
       inc     si
               ; mov (r2)+,r5 / move pointer to word 1 of an I/O
                          ; queue entry into r5
       test    word ptr [BX], 0F600h
               ; bit $173000,(r5) / lock+keep+active+outstanding
        jnz    short bufaloc_3
               ; bne 3f / branch when
                      ; / any of bits 9,10,12,13,14,15 are set
                       ; / (i.e., buffer busy)
        mov     word ptr [BP], si ; pointer to word 2 of I/0 queue
                               ; entry
                ; mov  r2,(sp) ;/ save pointer to last non-busy buffer
                      ; / found points to word 2 of I/O queue entry)
bufaloc_3: ;3:
       ;mov    dl, byte ptr [DI] ; 26/04/2013
       ;
       cmp     byte ptr [BX], dl
               ; cmpb (r5),cdev / is device in I/O queue entry same
                             ; / as current device
       jne     short bufaloc_4
               ; bne 3f
       cmp     word ptr [BX]+2, ax
               ; cmp 2(r5),r1 / is block number in I/O queue entry,
                           ; / same as current block number
       jne     short bufaloc_4
               ; bne 3f
       ;add    sp, 2
       pop     cx
               ; tst (sp)+ / bump stack pointer
       dec     si ; 09/07/2013
       dec     si ; 09/07/2013
       jmp     short bufaloc_7 ; Retro Unix 8086 v1 modification
                              ; jump to bufaloc_6 in original Unix v1
               ; br 1f / use this buffer
bufaloc_4: ;3:
       cmp     si, offset bufp + nbuf + nbuf
               ; cmp r2,$bufp+nbuf+nbuf
       jb      short bufaloc_2
               ; blo 2b / go to 2b if r2 less than bufp+nbuf+nbuf (all
                       ; / buffers not checked)
        pop    si
               ; mov (sp)+,r2 / once all bufs are examined move pointer
                           ; / to last free block
       or      si, si
       jnz     short bufaloc_5
               ; bne 2f / if (sp) is non zero, i.e.,
                ; / if a free buffer is found branch to 2f
        ;; mov  cx, word ptr [s.wait_]+2 ;; 29/07/2013
       call    idle
               ; jsr r0,idle; s.wait+2 / idle if no free buffers
       ; 26/04/2013
       ;xor    dx, dx
       xor     dl, dl
       push    dx ; 0
       ;
       jmp     short bufaloc_1
               ; br 1b
```

```
bufaloc_5: ;2:
                ; tst (r0)+ / skip if warmed over buffer
        inc     dh ; Retro UNIX 8086 v1 modification
bufaloc_6: ;1:
        dec     si
        dec     si
         mov            bx, word ptr [SI]
                ; mov -(r2),r5 / put pointer to word 1 of I/O queue
                             ; / entry in r5
        ;; 26/04/2013
         ;mov   dl, byte ptr [DI] ; byte ptr [rdev] or byte ptr [mdev]
        mov     byte ptr [BX], dl
                ; movb cdev,(r5) / put current device number
                                ; / in I/O queue entry
        mov     word ptr [BX]+2, ax
                ; mov r1,2(r5) / move block number into word 2
                             ; / of I/O queue entry
bufaloc_7: ;1:
         cmp    si, offset bufp
                ; cmp r2,$bufp / bump all entrys in bufp
                             ; / and put latest assigned
        jna     short bufaloc_8
                ; blos 1f / buffer on the top
                        ; / (this makes if the lowest priority)
        dec     si
        dec     si
        mov     cx, word ptr [SI]
        mov     word ptr [SI]+2, cx
                ; mov -(r2),2(r2) / job for a particular device
        jmp     short bufaloc_7
                ; br 1b
bufaloc_8: ;1:
         mov    word ptr [SI], bx
                ; mov r5,(r2)
        ;;pop   si ; ***
                ; mov (sp)+,r2 / restore r2
        or      dh, dh ; 0 or 1 ?
                ; Retro UNIX 8086 v1 modification
                ; zf=1 --> block already in a I/O buffer
                ; zf=0 --> a new I/O buffer has been allocated
        retn
                ; rts r0

diskio:
        ; 26/04/2013 Device ID modifications
        ; 15/03/2013
        ; Retro UNIX 8086 v1 feature only !
        ;
        ; Derived from proc_chs_read procedure of TRDOS DISKIO.ASM (2011)
        ; 04/07/2009 - 20/07/2011
        ;
        ; NOTE: Reads only 1 block/sector (sector/block size is 512 bytes)
        ;
        ; INPUTS ->
        ;          BX = System I/O Buffer header address
        ; OUTPUTS -> cf=0 --> done
        ;            cf=1 ---> error code in AH
        ;
        ; (Modified registers: CX,DX,AX)

        ;; I/O Queue Entry (of original UNIX operating system v1)
        ;; Word 1, Byte 0 = device id
        ;; Word 1, Byte 1 = (bits 8 to 15)
        ;;          bit  9 = write bit
        ;;          bit 10 = read bit
        ;;          bit 12 = waiting to write bit
        ;;          bit 13 = waiting to read bit
        ;;          bit 15 = inhibit bit
        ;; Word 2 = physical block number (In fact, it is LBA for Retro UNIX 8086 v1)
        ;;
        ;; Original UNIX v1 -> ; 26/04/2013
        ;;            Word 3 = number of words in buffer (=256)
        ;; Original UNIX v1 -> ; 26/04/2013
        ;;            Word 4 = bus address (addr of first word of data buffer)
        ;;
        ;; Retro UNIX 8086 v1 -> Buffer Header (I/O Queue Entry) size is 4 bytes !
        ;;
        ;; Device IDs (of Retro Unix 8086 v1) ; 26/04/2013
        ;;        0 = fd0
        ;;        1 = fd1
        ;;        2 = hd0
        ;;        3 = hd1
        ;;        4 = hd2
        ;;        5 = hd3

        mov     dx, 0201h ; Read 1 sector/block
        mov     ax, word ptr [BX]
        ; 26/04/2013
        push    si ; ****
        mov     cl, al
        xor     ch, ch
        mov     si, cx
        ;
        test    ah, 2
        ;test   ax, 200h ; Bit 9 of word 0 (status word)
                           ; write bit
        jz      short @f
        ;test   ah, 4
        ;;test  ax, 400h ; Bit 10 of word 0 (status word)
        ;                ; read bit
        ;jz     short diskio_ret
        inc     dh ; 03h = write
@@:
        ;mov    cx, 4 ; Retry Count
        mov     cl, 4

        ; push  ds
        ; pop   es
@@:
        push    dx ; ***
        push    bx ; ***
        push    cx ; ***
        push    dx ; ** ; I/O type (Int 13h function, r/w)
        inc     bx ; +1
        inc     bx ; +2
        mov     ax, word ptr [BX] ; Block/Sector number
        xor     dx, dx
        shl     si, 1 ; 2 * device number ; 26/04/2013
        mov     cx, word ptr [SI]+drv.spt
                                ; Sectors per track
        div     cx
        mov     cx, dx ; remainder, sector (zero based)
        inc     cx     ; sector (1 based)
        push    cx ; *
        mov     cx, word ptr [SI]+drv.hds ; Heads
        xor     dx, dx
        ; ax = track number
        div     cx
        mov     dh, dl ; head number (<=255)
        shr     si, 1 ; device number ; 26/04/2013
        mov      dl, byte ptr [SI]+drv.pdn ; 26/04/2013
                        ; Physical device number
        pop     cx ; * ; cx = sector of track (1 to spt)
        inc     bx ; +2
        inc     bx ; +3  ; I/O Buffer (Data)
        mov     ch, al ; low 8 bytes of cylinder number
        ror     ah, 1
        ror     ah, 1
        or      cl, ah
        pop     ax ; ** ; AH=2-read, AH=3-write
        int     13h    ; AL-count CH-track CL-sect
                        ; DH-head DL-drive ES:BX-buffer
                        ; CF-flag AH-stat AL-sec read
        pop     cx ; ***
        pop     bx ; ***
        jnc     short @f
        cmp     cl, 1
        jb      short @f
        xor     ah, ah ; Disk Reset
        int     13h
        dec     cx
        pop     dx ; ***
        jmp     short @b
@@:
        pop     dx ; ***
        pop     si ; ****
        retn
```