; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; UNIX.ASM (RETRO UNIX 8086 Kernel - Only for 1.44 MB floppy disks)

; ----------------------------------------------------------------------------

; U0.ASM (include u0.asm) //// UNIX v1 -> u0.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

; (11/03/2013)

;

; [ Last Modification: 15/04/2015 ] ;;; 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>

;

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; 23/07/2014, 27/07/2014, 28/07/2014

; 07/07/2014, 08/07/2014, 12/07/2014, 20/07/2014

; 30/06/2014, 03/07/2014, 04/07/2014, 05/07/2014

; 23/06/2014, 25/06/2014, 26/06/2014, 27/06/2014

; 22/05/2014, 26/05/2014, 02/06/2014, 03/06/2014

; 01/05/2014, 05/05/2014, 19/05/2014, 20/05/2014

; 14/04/2014, 25/04/2014, 29/04/2014, 30/04/2014

; 03/03/2014, 04/03/2014, 07/03/2014, 12/03/2014

; 05/02/2014, 14/02/2014, 23/02/2014, 28/02/2014

; 17/01/2014, 18/01/2014, 20/01/2014, 01/02/2014

; 30/10/2013, 04/12/2013, 06/12/2013, 10/12/2013

; 24/09/2013, 29/09/2013, 05/10/2013, 10/10/2013

; 30/08/2013, 03/09/2013, 17/09/2013, 20/09/2013

; 23/07/2013, 29/07/2013, 11/08/2013, 12/08/2013

; 16/07/2013, 17/07/2013, 18/07/2013, 22/07/2013

; 15/07/2013, 20/05/2013, 21/05/2013, 27/05/2013

; 15/05/2013, 17/05/2013, 13/07/2013, 14/07/2013

; 11/03/2013, 11/04/2013, 09/05/2013, 10/05/2013

; 29/04/2014 --> serial port (terminal) login functionality test

; by using fake INT 14h, tty6, tty7

; etc/init has been modified for leaving tty6 and tty7 free

kernel\_init:

; 15/04/2015

; 07/03/2014

; 04/03/2013

; 28/02/2014

; 14/02/2014

; 05/02/2014

; 04/12/2013

; 05/10/2013

; 29/07/2013

; 18/07/2013

; 17/07/2013

; 14/07/2013

; 13/07/2013

; Retro UNIX 8086 v1 feature only !

;

; Retro UNIX 8086 v1

; kernel relies on data from its 'boot' program ...

;

;;mov ax, cs

;mov ds, ax

;mov es, ax

;cli

;mov ss, ax

;mov sp, 32766

;sti

; mov bp, sp

mov byte ptr [unixbootdrive], dl

mov ds, cx ; boot sector segment

; bx = boot sector buffer

mov ax, word ptr [BX]+2 ; 14/07/2013

mov dx, word ptr [BX]+4 ; 14/07/2013

push cs

pop ds

cmp ax, 'UR'

jne kernel\_init\_err ; jne short kernel\_init\_err

cmp dx, 'SF'

jne kernel\_init\_err ; jne short kernel\_init\_err

;

call drv\_init

jc kernel\_init\_err ; jne short kernel\_init\_err

;

; 14/02/2014

; 14/07/2013

mov ax, 41

mov word ptr [rootdir], ax

mov word ptr [u.cdir], ax

mov ax, 1 ; 15/04/2015 (mov al, 1)

mov byte ptr [u.uno], al

mov word ptr [mpid], ax

mov word ptr [p.pid], ax

mov byte ptr [p.stat], al ; SRUN, 05/02/2014

;

mov al, time\_count ; 30/08/2013

;; 29/07/2013

;;mov byte ptr [s.wait\_]+2, al

;;mov byte ptr [s.idlet]+2, al

; 14/02/2014 uquant -> u.quant

mov byte ptr [u.quant], al ; 14/07/2013

; 22/07/2013

mov ax, cs

mov word ptr [u.segmnt], ax ; reset to CS

;

call epoch

mov word ptr [s.time], ax

mov word ptr [s.time]+2, dx

;

call kb\_init

; ES = 0 (30/06/2014)

;

; 28/02/2014 INT 16h handler

mov ax, offset int\_16h

mov di, 22\*4 ; INT 16h vector - offset

stosw

mov ax, cs

stosw

;mov es, ax ; 30/06/2014)

;

;; 10/12/2013

;; INT 1Ch handling disabled here,

;; it will be enabled by 'sys emt'

;; system call (in 'etc/init')

; INT 1Ch (clock/timer) transfer to unix kernel

;; 30/06/2014

;;xor ax, ax

;;mov es, ax ; 0

;; ES = 0

;mov di, 28\*4 ; INT 1Ch vector - offset

;cli

;mov ax, offset clock

;stosw ; offset

;mov ax, cs

;stosw ; segment

;sti

;

; setting up syscall vector (int 20h)

mov ax, offset sysent

mov di, 32\*4 ; INT 20h for system calls

stosw

mov ax, cs

stosw

;mov es, ax ; 14/04/2014

;

;

;; 13/07/2013

;; Kernel is running message ... (temporary)

;

mov si, offset kernel\_init\_ok\_msg

; 07/03/2014

;call print\_msg

lodsb

mov ah, 0Eh

mov bx, 07h

@@:

int 10h

lodsb

and al, al

jnz short @b

;

; 17/01/2014

; ES = 0

call sp\_init ; serial port interrupts

; 14/04/2014

mov ax, cs

mov es, ax

;

; 05/10/2013 Temporary

xor al, al ; mov al, 0

; mov byte ptr [u.ttyn], 0

call getc

; 16/07/2013

;xor al, al

; 04/12/2013

xor bl, bl ; video page 0

@@: ; clear video pages (reset cursor positions)

call vp\_clr ; 17/07/2013

inc bl

cmp bl, 8

jb short @b

;

; 17/07/2013

;mov al, byte ptr [unixbootdrive]

;cmp al, 80h ; 128 (80h->hd0)

;jna short @f

;sub al, 7Eh ; 126 (2->hd0)

;@@:

;mov byte ptr [rdev], al

;

call bf\_init ; buffer initialization ; 17/07/2013

;; original UNIX v1 (PDP-11) code here:

; / make current program a user

;

; mov $41.,r0 / rootdir set to 41 and never changed

; mov r0,rootdir / rootdir is i-number of root directory

; mov r0,u.cdir / u.cdir is i-number of process current directory

; mov $1,r0

; movb r0,u.uno / set process table index for this process to 1

; mov r0,mpid / initialize mpid to 1

; mov r0,p.pid / p.pid identifies process

; movb r0,p.stat / process status = 1 i.e., active

; / = 0 free

; / = 2 waiting for a child to die

; / = 3 terminated but not yet waited

; / for

; 18/01/2014

;sti

; 24/07/2013

mov bx, offset init\_file

mov cx, offset init\_argp

; (([u.segmnt] = CS))

; BX contains 'etc/init' asciiz file name address

; CX contains address of argument list pointer

;

dec byte ptr [sysflg] ; FFh = ready for system call

; 0 = executing a system call

;mov ax, \_exec

;int 20h

sys \_exec ; execute file

;

jnc short panic

;

mov si, offset etc\_init\_err\_msg

jmp short @f

;; original UNIX v1 (PDP-11) code here:

; 1:

; decb sysflg / normally sysflag=0, indicates executing in system

; sys exec; 2f; 1f / generates trap interrupt; trap vector =

; / sysent; 0

; br panic / execute file/etc/init

; 1:

; 2f;0

; 2:

; </etc/init\0> / UNIX looks for strings term, noted by nul\0

kernel\_init\_err:

;; NOTE: UNix kernel will load boot sector

;;

mov si, offset kernel\_init\_err\_msg

@@:

call print\_msg

jmp short key\_to\_reboot

align 2

init\_argp:

dw offset init\_file, 0

init\_file:

db '/etc/init', 0

panic:

; 07/03/2014

; 05/10/2013 ('call getc' instead of 'int 16h')

; 14/07/2013 (panic\_msg/print\_msg)

; 10/04/2013

;

; Retro Unix 8086 v1 modification on original Unix v1 panic procedure!

;

mov si, offset panic\_msg

call print\_msg

key\_to\_reboot:

;hlt

; 05/10/2013

xor al, al

call getc

;

mov al, 0Ah

mov ah, byte ptr [ptty] ; [active\_page]

call write\_tty

;

; 15/07/2013

;mov ah, 0Eh

;;mov bx, 07h

;;mov al, 0Dh

;;int 10h

;mov al, 0Ah

;int 10h

cpu\_reset:

; 07/03/2014

; CPU reset (power on) address

db 0EAh ; far jump (jmp 0FFFFh:0000h)

dw 0

dw 0FFFFh ; F000:0FFF0h

;khere: hlt

; jmp short khere

;@@:

; 24/09/2013

; Reset INT 09h vector for next start-up

;xor di, di

;mov es, di

;mov di, 4\*9

;mov si, offset int09h

;movsw

;movsw

;

;int 19h

; hlt

; jmp short @b

; clr ps

;1:

; dec $0

; bne 1b

; dec $5

; bne 1b

; jmp \*$173700 / rom loader address

print\_msg:

; 07/03/2014

; (Modified registers: AX, BX, CX, DX, SI, DI)

;

lodsb

@@:

push si

mov ah, byte ptr [ptty]

call write\_tty

pop si

lodsb

and al, al

jnz short @b

retn

; 14/07/2013

; 13/07/2013

;lodsb

;mov bx, 07h

;mov ah, 0Eh

;@@:

;int 10h

;lodsb

;and al, al

;jnz short @b

;retn

kb\_init:

; 30/06/2014

; 03/03/2014

; 11/08/2013

; 16/07/2013

; 15/07/2013

; 13/07/2013

; 21/05/2013

; 17/05/2013

; 10/05/2013

;

; Initialization of keyboard handlers

;

; Retro Unix 8086 v1 feature only!

;

; ((Modified registers: AX, CX, SI, DI, ES))

;

xor ax, ax ; 11/08/2013

mov di, offset int09h

mov ds, ax ; 0

mov ax, 9\*4 ; INT 09h vector - offset

mov si, ax

movsw ; offset

movsw ; segment

mov di, ax

mov ax, ds

mov es, ax

mov ax, cs

mov ds, ax

cli

mov ax, offset kb\_int

stosw

mov ax, cs

stosw

mov ax, offset ctrlbrk

mov di, 27\*4 ; INT 1Bh vector - offset

stosw ; offset

mov ax, cs

stosw ; segment

sti

;mov es, ax ; 30/06/2014 (ES = 0)

;

; 03/03/2014

; SETUP KEYBOARD PARAMETERS

;mov si, offset KB\_BUFFER

;mov word ptr [BUFFER\_HEAD], si

;mov word ptr [BUFFER\_TAIL], si

;mov word ptr [BUFFER\_START], si

;add si, 32 ; DEFAULT BUFFER OF 32 BYTES

;mov word ptr [BUFFER\_END], si

;

retn

ctrlbrk:

; 06/12/2013

; 20/09/2013

; 03/09/2013

; 09/05/2013

;

; INT 1Bh (control+break) handler

;

; Retro Unix 8086 v1 feature only!

;

cmp word ptr CS:[u.intr], 0

ja short cbrk1

iret

cbrk1:

; 20/09/2013

push ax

mov al, byte ptr CS:[ptty]

inc al

; 06/12/2013

cmp al, byte ptr CS:[u.ttyp]

je short cbrk2

cmp al, byte ptr CS:[u.ttyp]+1

jne short cbrk3

cbrk2:

; 06/12/2013

mov ax, word ptr CS:[u.quit]

and ax, ax

jz short cbrk3

xor ax, ax ; 0

dec ax

; 0FFFFh = 'ctrl+brk' keystroke

mov word ptr CS:[u.quit], ax

cbrk3:

pop ax

iret

;tty\_sw: ; < tty switch >

; 23/02/2014

; 04/12/2013 'act\_disp\_page' (U9.ASM)

; 29/09/2013 (simplified)

; 29/09/2013 u1.asm -> u0.asm

; 22/09/2013

; 17/09/2013

; 03/09/2013

; 21/08/2013

; 18/08/2013

; 16/07/2013

; 15/07/2013

; 20/05/2013

;

; Retro UNIX 8086 v1 feature only !

;

; INPUTS:

; AL = tty number to be switched on

; OUTPUTS:

; Keyboard buffer will be reset and

; active video page will be changed

; according to the requested tty number.

;

; ((Modified registers: AX))

;

; 29/09/2013

; 03/09/2013

;

;mov al, byte ptr [nxtty] ; tty number

; ; video page

;;;

; 04/12/2013

;;mov ah, 5 ; Set video page

;;int 10h

;;mov byte ptr [ptty], al ; byte ptr [active\_page], al

;call act\_disp\_page

; 23/02/2014

;mov byte ptr [u.quant], 0

;retn

kb\_int:

; INT 09h Keyboard Handler

;

; 30/06/2014

; 12/03/2014

; 07/03/2014

; 04/03/2014

; 03/03/2014 major modification

; 25/02/2013 ;;

; 23/02/2014

; 14/02/2014

; 01/02/2014

; 20/01/2014

; 18/01/2014

; 17/01/2014

; 10/10/2013

; 05/10/2013

; 29/09/2013

; 24/09/2013

; 03/09/2013

; 12/08/2013

; 11/08/2013

; 20/05/2013

; 15/05/2013

; 10/05/2013

;

; Retro Unix 8086 v1 feature only!

; 03/03/2014

push ds

push ax

push bx

;

mov ax, cs

mov ds, ax

;

pushf

; 04/03/2014

;call dword ptr [int09h]

; 07/03/2014

push cs

call int\_09h

;

; 24/09/2013

mov ah, 1

int 16h

jz short kb\_int\_4

;

; 04/03/2014

mov bl, byte ptr [ptty]

xor ah, ah

int 16h

;

and al, al

jnz short kb\_int\_1

;

cmp ah, 68h ; ALT + F1 key

jb short kb\_int\_1

cmp ah, 6Fh ; ALT + F8 key

ja short kb\_int\_1

;

mov bh, bl

add bh, 68h

cmp bh, ah

je short kb\_int\_1

mov al, ah

sub al, 68h

;

;mov byte ptr [ptty], al ; [active\_page]

;

call tty\_sw

xor ax, ax ; 0 ; 07/03/2014

; 12/03/2014

mov bl, byte ptr [ptty]

kb\_int\_1:

xor bh, bh

shl bl, 1

add bx, offset ttychr

; 12/03/2014

or ax, ax

jz short kb\_int\_2

; 29/09/2013

cmp word ptr [BX], 0

ja short kb\_int\_3

kb\_int\_2:

;

; 24/09/2013

mov word ptr [BX], ax ; Save ascii code

; and scan code of the character

; for current tty (or last tty

; just before tty switch).

kb\_int\_3:

; 10/10/2013

mov al, byte ptr [ptty]

; 14/02/2014

;mov bx, offset runq

call wakeup

;

kb\_int\_4:

pop bx ; 24/09/2013

pop ax

pop ds

;

iret

vp\_clr:

; Reset/Clear Video Page

;

; 04/12/2013 scroll\_up (U9.ASM)

;

; 30/10/2013

; 17/09/2013

; 17/07/2013

; 21/05/2013

;

; Retro UNIX 8086 v1 feature only !

;

; INPUTS ->

; AL = video page number

;

; OUTPUT ->

; none

; ((Modified registers: AX, BH, CX, DX, SI, DI))

;

; 04/12/2013

sub al, al

; al = 0 (clear video page)

; bl = video page

mov bh, 07h

; bh = 7 (attribute/color)

call scroll\_up

; bh = 7

; bl = video page

xor dx, dx ; 0

;call set\_cpos

;retn

jmp set\_cpos

; 30/10/2013

;push es

;xor ah, ah

;;push ax

;mov di, 0B800h

;mov es, di

;mov cx, 2000

;sub dx, dx ; 30/10/2013

;or al, al

;jz short @f

;; 30/10/2013

;shl al, 1

;; 17/09/2013

;push ax

;mul cx

;pop dx

;@@:

;mov di, ax ; 17/09/2013

;mov ah, 07h ; color

;rep stosw

;;pop ax

;;mov bh, al ; video page

;;mov ah, 2 ; set cursor position

;;xor dx, dx

;;int 10h

;;xor ax, ax

;xor ah, ah

;;pop di ; Video page number

;;shl di, 1

;mov di, dx

;mov es, ax ; 0

;add di, 450h ; 40h:50h or 0h:450h

;; di = cursor position of the video page.

;stosw ; reset cursor position

;pop es

;retn

com2\_int:

; 28/07/2014

; 27/07/2014

; 23/07/2014

; 20/07/2014 (null chr)

; 07/07/2014

; 05/07/2014

; 04/07/2014

; < serial port 2 interrupt handler >

;

; Retro UNIX 8086 v1 feature only !

;

push dx

push ax

mov dx, 2FAh ; interrupt identification register

mov ax, 9 ; tty number of com2

jmp short @f

com1\_int:

; 28/07/2014

; 27/07/2014

; 23/07/2014

; 20/07/2014 (null chr)

; 07/07/2014

; 05/07/2014

; 04/07/2014

; < serial port 1 interrupt handler >

;

; Retro UNIX 8086 v1 feature only !

;

push dx

push ax

mov dx, 3FAh ; interrupt identification register

mov ax, 8 ; tty number of com1

@@:

push ds

push bx

push cs

pop ds

push ax

;

mov bx, ax

in al, dx ; read register

and al, 0Fh ; leave lowernibble only

; 28/07/2014

cmp al, 2

jne short com\_rdei

;

add bx, offset tsleep - 8

cmp byte ptr [BX], ah ; 0

jna short @f

mov byte ptr [BX], ah ; 0

jmp short com\_eoi

@@:

mov al, 20h

out 20h, al ; end of interrupt

pop ax

jmp short com\_iret

com\_rdei:

cmp al, 4 ; is it receiver data available interrupt?

jne short com\_eoi ; no, leave interrupt handler

;

sub dx, 3FAh-3F8h ; data register (3F8h, 2F8h)

in al, dx ; read character

; 27/07/2014

and al, al

jnz short @f

; null chr (al=0, ah=0)

dec ah ; 0FFh

@@: ; 27/07/2014

; 09/07/2014

shl bl, 1

add bx, offset ttychr

; 23/07/2014 (always overwrite)

;;cmp word ptr [BX], 0

;;ja short com\_eoi

;

mov word ptr [BX], ax ; Save ascii code

; scan code = 0

com\_eoi:

mov al, 20h

out 20h, al ; end of interrupt

;

pop ax ; al = tty number (8 or 9)

call wakeup

com\_iret:

pop bx

pop ds

pop ax

pop dx

iret

sp\_init:

; 28/07/2014

; 27/07/2014

; 12/07/2014

; 08/07/2014

; 05/07/2014

; 03/07/2014

; 17/01/2014

;

; Initialization of serial port interrupt handlers

;

; Retro Unix 8086 v1 feature only!

;

; ((Modified registers: AX, CX, DX, DI))

;

; ES = 0

;

; Set communication parameters for COM1

;

mov cl, 0E3h

xor ah, ah

mov al, cl ; Communication parameters (E3h)

; 9600 baud, parity none, one stop bit

xor dx, dx ; COM1 (DX=0)

int 14h

; 12/07/2014

test ah, 80h

jnz short @f

; (Note: Serial port interrupts will be disabled here...)

; (INT 14h initialization code disables interrupts.)

mov byte ptr [com1p], cl ; 0E3h

;

;; Hook serial port (COM1) interrupt

;

mov di, 12 \* 4 ; 0Ch, COM1 (IRQ 4) interrupt vector

;cli

mov ax, offset com1\_int

stosw

mov ax, cs

stosw

;sti

;

;; COM1 - enabling IRQ 4

mov dx, 3FCh ;modem control register

in al, dx ;read register

or al, 8 ;enable bit 3 (OUT2)

out dx, al ;write back to register

mov dx, 3F9h ;interrupt enable register

in al, dx ;read register

;or al, 1 ;receiver data interrupt enable

; 27/7/2014 ; and

or al, 3 ;Transmitter empty interrupt enable

;

out dx, al ;write back to register

in al, 21h ;read interrupt mask register

and al, 0EFh ;enable IRQ 4 (COM1)

out 21h, al ;write back to register

;

; Set communication parameters for COM2

;

mov dx, 1 ; COM2

sub ah, ah

mov al, cl ; Communication parameters (E3h)

; 9600 baud, parity none, one stop bit

int 14h

; 12/07/2014

test ah, 80h

jnz short @f

; (Note: Serial port interrupts will be disabled here...)

; (INT 14h initialization code disables interrupts.)

mov byte ptr [com2p], cl ; 0E3h

;

;; Hook serial port (COM2) interrupt

;

mov di, 11 \* 4 ; 0Bh, COM2 (IRQ 3) interrupt vector

;cli

mov ax, offset com2\_int

stosw

mov ax, cs

stosw

;sti

;

;; COM2 - enabling IRQ 3

mov dx, 2FCh ;modem control register

in al, dx ;read register

or al, 8 ;enable bit 3 (OUT2)

out dx, al ;write back to register

mov dx, 2F9h ;interrupt enable register

in al, dx ;read register

;or al, 1 ;receiver data interrupt enable

; 27/7/2014 ; and

or al, 3 ;Transmitter empty interrupt enable

;

out dx, al ;write back to register

in al, 21h ;read interrupt mask register

and al, 0F7h ;enable IRQ 3 (COM2)

out 21h, al ;write back to register

@@:

retn