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

;

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

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

; U4.ASM (include u4.asm) //// UNIX v1 -> u4.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: 04/07/2014 ] !!! 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>

;

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

; 04/07/2014 (swakeup has been removed)

; 11/06/2014 swakeup

; 02/06/2014 swakeup

; 30/05/2014 isintr

; 20/03/2014 sleep

; 18/03/2014 clock

; 25/02/2014 sleep

; 23/02/2014 wakeup, sleep

; 17/02/2014 wakeup

; 14/02/2014 clock

; 14/02/2014 sleep, wakeup (sigle level runq) ((to prevent s/w locking))

; 05/02/2014 sleep, wakeup (SSLEEP/SRUN, p.waitc)

; 26/01/2014

; 10/12/2013

; 07/12/2013 clock

; 23/10/2013 wakeup, sleep

; 20/10/2013 isintr, clock, wakeup, sleep

; 05/10/2013 clock, wakeup, sleep

; 24/09/2013 sleep, wakeup (consistency check)

; 22/09/2013 sleep, wakeup (completed/modified)

; 20/09/2013 clock, sleep

; NOTE: 'sleep' and 'wakeup' need to be modified according to

; original Unix v1 waiting channel feature.

; Currently 'wakeup' is disabled and 'sleep' is not written

; properly and clock, sleep, wakeup are not similar

; to original unix v1 (musti tasking, time sharing feature).

; 03/09/2013 clock, isintr

; 30/08/2013 clock

; 21/08/2013

; 29/07/2013 sleep

; 09/07/2013 clock (INT 1Ch handler)

; 16/05/2013 'isintr' modifications

; 15/05/2013

; 09/05/2013

; 11/03/2013

;setisp:

;mov r1,-(sp)

;mov r2,-(sp)

;mov r3,-(sp)

;mov clockp,-(sp)

;mov $s.syst+2,clockp

;jmp (r0)

clock: ; / interrupt from 60 cycle clock

; 10/04/2014

; 18/03/2014

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

; 10/12/2013

; 07/12/2013

;; Retro Unix 8086 v1 Modification: INT 1Ch interrupt handler !

;; 30/08/2013

;; 09/07/2013

;mov r0,-(sp) / save r0

;tst \*$lks / restart clock?

;mov $s.time+2,r0 / increment the time of day

;inc (r0)

;bne 1f

;inc -(r0)

;1:

;mov clockp,r0 / increment appropriate time category

;inc (r0)

;bne 1f

;inc -(r0)

;1:

;; 30/08/2013

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; 09/07/2013

; 20/10/2013

push ds

push cs

pop ds

;

;; 10/04/2014

;pushf

;call dword ptr [int1Ch] ; Old INT 1Ch

; ; (Turn off floppy motor)

cmp byte ptr [u.quant], 0

ja short clk\_1

; 03/09/2013

cmp byte ptr [sysflg], 0FFh ; user or system space ?

jne short clk\_2 ; system space (sysflg <> 0FFh)

;; 06/12/2013

cmp byte ptr [u.uno], 1 ; /etc/init ?

; 14/02/2014

jna short clk\_1 ; yes, do not swap out

cmp word ptr [u.intr], 0

; 14/02/2014

jna short clk\_2

clk\_0:

; 30/08/2013

;cli

;;push cs

;;pop ds

; 18/03/2014

inc byte ptr [sysflg] ; Now, we are in system spacee

;

mov word ptr [u.r0], ax

; 07/12/2013

pop ax ; DS (user)

;

mov word ptr [u.usp], sp

;; 07/12/2013

;;mov ax, ss ; mov ax, es

;;mov word ptr [u.segmnt], ax

mov ax, cs

;mov es, ax ; 18/03/2014

mov sp, sstack

mov ss, ax

;

push word ptr [u.usp]

push dx

push cx

push bx

push si

push di

push bp

;

mov word ptr [u.sp\_], sp

;sti

; 07/12/2013

jmp sysrelease ; 'sys release' by clock/timer

clk\_1:

dec byte ptr [u.quant]

clk\_2:

; 20/10/2013

pop ds

iret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;mov $uquant,r0 / decrement user time quantum

;decb (r0)

;bge 1f / if less than 0

;clrb (r0) / make it 0

;1: / decrement time out counts return now if priority was not 0

;cmp 4(sp),$200 / ps greater than or equal to 200

;bge 2f / yes, check time outs

;tstb (r0) / no, user timed out?

;bne 1f / no

;cmpb sysflg,$-1 / yes, are we outside the system?

;bne 1f / no, 1f

;mov (sp)+,r0 / yes, put users r0 in r0

;sys 0 / sysrele

;rti

;2: / priority is high so just decrement time out counts

;mov $toutt,r0 / r0 points to beginning of time out table

;2:

;tstb (r0) / is the time out?

;beq 3f / yes, 3f (get next entry)

;decb (r0) / no, decrement the time

;bne 3f / isit zero now?

;incb (r0) / yes, increment the time

;3:

;inc r0 / next entry

;cmp r0,$touts / end of toutt table?

;blo 2b / no, check this entry

;mov (sp)+,r0 / yes, restore r0

;rti / return from interrupt

;1: / decrement time out counts; if 0 call subroutine

;mov (sp)+,r0 / restore r0

;mov $240,\*$ps / set processor priority to 5

;jsr r0,setisp / save registers

;mov $touts-toutt-1,r0 / set up r0 as index to decrement thru

; / the table

;1:

;tstb toutt(r0) / is the time out for this entry

;beq 2f / yes

;decb toutt(r0) / no, decrement the time

;bne 2f / is the time 0, now

;asl r0 / yes, 2 x r0 to get word index for tout entry

;jsr r0,\*touts(r0) / go to appropriate routine specified in this

;asr r0 / touts entry; set r0 back to toutt index

;2:

;dec r0 / set up r0 for next entry

;bge 1b / finished? , no, go back

;br retisp / yes, restore registers and do a rti

;retisp:

;mov (sp)+,clockp / pop values before interrupt off the stack

;mov (sp)+,r3

;mov (sp)+,r2

;mov (sp)+,r1

;mov (sp)+,r0

;rti / return from interrupt

@@: ; 22/09/2013

retn

wakeup: ; / wakeup processes waiting for an event

; / by linking them to the queue

;

; 02/06/2014

; 23/02/2014

; 17/02/2014

; 14/02/2014 single level runq (BX input is not needed)

; 05/02/2014 SSLEEP/SRUN, p.waitc

; 23/10/2013 (consistency check is OK)

; 20/10/2013

; 10/10/2013

; 05/10/2013

; 24/09/2013 (consistency check is OK)

; 22/09/2013

; 18/08/2013 -> tty lock and console tty setting (p.ttyc)

; 15/05/2013

; Retro UNIX 8086 v1 modification !

; (Process/task switching routine by using

; Retro UNIX 8086 v1 keyboard interrupt output.))

;

; In original UNIX v1, 'wakeup' is called to wake the process

; sleeping in the specified wait channel by creating a link

; to it from the last user process on the run queue.

; If there is no process to wake up, nothing happens.

;

; In Retro UNIX 8086 v1, Int 09h keyboard interrupt will set

; 'switching' status of the current process (owns current tty)

; (via alt + function keys) to a process which has highest

; priority (on run queue) on the requested tty (0 to 7, except

; 8 and 9 which are tty identifiers of COM1, COM2 serial ports)

; as it's console tty. (NOTE: 'p.ttyc' is used to set console

; tty for tty switching by keyboard.)

;

; INPUT ->

; AL = wait channel (r3) ('tty number' for now)

; ;;BX = Run queue (r2) offset

;

; ((modified registers: AX, BX))

;

; 20/10/2013

; 10/10/2013

;;cmp byte ptr [u.uno], 2

;;jb short wakeup\_4

; 14/02/2014

xor bh, bh

mov bl, al

add bx, offset wlist

; 23/02/2014

mov al, byte ptr [BX] ; waiting list (waiting process number)

and al, al

jz short @f ; nothing to wakeup

;cmp al, 1

;jb short @f ; nothing to wakeup

; 23/02/2014

;

xor ah, ah

mov byte ptr [u.quant], ah ; 0 ; time quantum = 0

mov byte ptr [BX], ah ; 0 ; zero wait channel entry

push di

push dx

call putlu

pop dx

pop di

@@:

retn

;mov r1,-(sp) / put char on stack

;mov (r0)+,r2 / r2 points to a queue

;mov (r0)+,r3 / r3 = wait channel number

;movb wlist(r3),r1 / r1 contains process number

; / in that wait channel that was sleeping

;beq 2f / if 0 return, nothing to wakeup

;cmp r2,u.pri / is runq greater than or equal

; / to users process priority

;bhis 1f / yes, don't set time quantum to zero

;clrb uquant / time quantum = 0

;1:

;clrb wlist(r3) / zero wait channel entry

;jsr r0,putlu / create a link from the last user

; / on the Q to this process number that got woken

;2:

;mov (sp)+,r1 / restore r1

;rts r0

sleep:

; 20/03/2014

; 25/02/2014

; 23/02/2014

; 14/02/2014 single level runq

; 05/02/2014 SSLEEP/SRUN, p.waitc

; 26/01/2014

; 10/12/2013

; 23/10/2013 (consistency check is OK)

; 20/10/2013

; 05/10/2013 (u.uno = 1 --> /etc/init ?) (r1 = ah)

; 24/09/2013 consistency check -> OK

; 22/09/2013

; 20/09/2013

; 29/07/2013 ;;;

; 09/05/2013

; Retro UNIX 8086 v1 modification !

; (Process/task switching and quit routine by using

; Retro UNIX 8086 v1 keyboard interrupt output.))

;

; In original UNIX v1, 'sleep' is called to wait for

; tty and tape output or input becomes available

; and process is put on waiting channel and swapped out,

; then -when the tty or tape is ready to write or read-

; 'wakeup' gets process back to active swapped-in status.)

;

; In Retro UNIX 8086 v1, Int 1Bh ctrl+brk interrupt and

; Int 09h keyboard interrupt will set 'quit' or 'switching'

; status of the current process also INT 1Ch will count down

; 'uquant' value and INT 09h will redirect scancode of keystroke

; to tty buffer of the current process and kernel will get

; user input by using tty buffer of the current process

; (instead of standard INT 16h interrupt).

; TTY output will be redirected to related video page of text mode

; (INT 10h will be called with different video page depending

; on tty assignment of the active process: 0 to 7 for

; pseudo screens.)

;

; In Retro UNIX 8086 v1, 'sleep' will be called to wait for

; a keystroke from keyboard or wait for reading or writing

; characters/data on serial port(s).

;

; Character/Terminal input/output through COM1 and COM2 will be

; performed by related routines in addition to pseudo TTY routines.

;

; R1 = AH = wait channel (0-9 for TTYs) ; 05/10/2013 (22/09/2013)

;

;; 05/10/2013

;10/12/2013

;cmp byte ptr [u.uno], 1

;ja short @f

;retn

; 20/03/2014

;mov bx, word ptr [runq]

;cmp bl, bh

;jne short @f

; 25/02/2014

;cmp word ptr [runq], 0

;ja short @f

;retn

@@:

;

call isintr

jnz sysret

; / wait for event

; jsr r0,isintr / check to see if interrupt

; / or quit from user

; br 2f / something happened

; / yes, his interrupt so return

; / to user

; 20/10/2013

xor bh, bh

mov bl, ah

; 22/09/2013

add bx, offset wlist

; 23/02/2014

mov al, byte ptr [BX]

and al, al

jz short @f

push bx

call putlu

pop bx

@@:

mov al, byte ptr [u.uno]

mov byte ptr [BX], al ; put the process number

; in the wait channel

; mov (r0)+,r1 / put number of wait channel in r1

; movb wlist(r1),-(sp) / put old process number in there,

; / on the stack

; movb u.uno,wlist(r1) / put process number of process

; / to put to sleep in there

push word ptr [cdev]

; mov cdev,-(sp) / nothing happened in isintr so

call swap

; jsr r0,swap / swap out process that needs to sleep

pop word ptr [cdev]

; mov (sp)+,cdev / restore device

call isintr

; 22/09/2013

jnz sysret

; jsr r0,isintr / check for interrupt of new process

; br 2f / yes, return to new user

; movb (sp)+,r1 / no, r1 = old process number that was

; / originally on the wait channel

; beq 1f / if 0 branch

; mov $runq+4,r2 / r2 points to lowest priority queue

; mov $300,\*$ps / processor priority = 6

; jsr r0,putlu / create link to old process number

; clr \*$ps / clear the status; process priority = 0

;1:

retn

; rts r0 / return

;2:

;;jmp sysret

; jmp sysret / return to user

isintr:

; 30/05/2014

; 20/10/2013

; 22/09/2013

; 03/09/2013

; 16/05/2013 tty/video\_page switching

; 09/05/2013

; Retro UNIX 8086 v1 modification !

; (Process/task switching and quit routine by using

; Retro UNIX 8086 v1 keyboard interrupt output.))

;

; Retro UNIX 8086 v1 modification:

; 'isintr' checks if user interrupt request is enabled

; and there is a 'quit' request by user;

; otherwise, 'isintr' will return with zf=1 that means

; "nothing to do". (20/10/2013)

;

; 20/10/2013

cmp word ptr [u.ttyp], 0 ; has process got a tty ?

jna short isintr2 ; retn

; 03/09/2013

; (nothing to do)

;retn

; 22/09/2013

cmp word ptr [u.intr], 0

jna short isintr2 ; retn

; 30/05/2014

push ax

mov ax, word ptr [u.quit]

or ax, ax ; 0 ?

jz short isintr1 ; zf = 1

cmp ax, 0FFFEh ; 'ctrl + brk' check

ja short isintr1 ; 0FFFFh, zf = 0

xor ax, ax ; zf = 1

isintr1:

pop ax

isintr2: ; 22/09/2013

; zf=1 -> nothing to do

retn

; UNIX v1 original 'isintr' routine...

;mov r1,-(sp) / put number of wait channel on the stack

;mov r2,-(sp) / save r2

;mov u.ttyp,r1 / r1 = pointer to buffer of process control

; / typewriter

;beq 1f / if 0, do nothing except skip return

;movb 6(r1),r1 / put interrupt char in the tty buffer in r1

;beq 1f / if its 0 do nothing except skip return

;cmp r1,$177 / is interrupt char = delete?

;bne 3f / no, so it must be a quit (fs)

;tst u.intr / yes, value of u.intr determines handling

; / of interrupts

;bne 2f / if not 0, 2f. If zero do nothing.

;1:

;tst (r0)+ / bump r0 past system return (skip)

;4:

;mov (sp)+,r2 / restore r1 and r2

;mov (sp)+,r1

;rts r0

;3: / interrupt char = quit (fs)

;tst u.quit / value of u.quit determines handling of quits

;beq 1b / u.quit = 0 means do nothing

;2: / get here because either u.intr <> 0 or u.qult <> O

;mov $tty+6,r1 / move pointer to tty block into r1

;1: / find process control tty entry in tty block

;cmp (r1),u.ttyp / is this the process control tty buffer?

;beq 1f / block found go to 1f

;add $8,r1 / look at next tty block

;cmp r1,$tty+[ntty\*8]+6 / are we at end of tty blocks

;blo 1b / no

;br 4b / no process control tty found so go to 4b

;1:

;mov $240,\*$ps / set processor priority to 5

;movb -3(r1),0f / load getc call argument; character llst

; / identifier

;inc 0f / increment

;1:

;jsr r0,getc; 0:.. / erase output char list for control

; br 4b / process tty. This prevents a line of stuff

; / being typed out after you hit the interrupt

; / key

;br 1b