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

;

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

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

;

; 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 ]

;

; 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>

;

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

; 28/08/2014, 01/09/2014

; 20/07/2014, 21/07/2014, 23/07/2014, 24/07/2014, 27/07/2014, 28/07/2014

; 05/07/2014, 07/07/2014, 08/07/2014, 09/07/2014, 12/07/2014, 18/07/2014

; 26/06/2014, 27/06/2014, 30/06/2014, 01/07/2014, 03/07/2014, 04/07/2014

; 31/05/2014, 02/06/2014, 03/06/2014, 11/06/2014, 23/06/2014, 25/06/2014

; 05/05/2014, 19/05/2014, 20/05/2014, 22/05/2014, 26/05/2014, 30/05/2014

; 17/04/2014, 22/04/2014, 25/04/2014, 29/04/2014, 30/04/2014, 01/05/2014

; 24/03/2014, 04/04/2014, 10/04/2014, 11/04/2014, 14/04/2014, 15/04/2014

; 04/03/2014, 07/03/2014, 08/03/2014, 12/03/2014, 18/03/2014, 20/03/2014

; 14/02/2014, 17/02/2014, 23/02/2014, 25/02/2014, 28/02/2014, 03/03/2014

; 18/01/2014, 20/01/2014, 21/01/2014, 26/01/2014, 01/02/2014, 05/02/2014

; 10/01/2014, 12/01/2014, 13/01/2014, 14/01/2014, 16/01/2014, 17/01/2014

; 03/12/2013, 04/12/2013, 06/12/2013, 07/12/2013, 10/12/2013, 12/12/2013

; 24/10/2013, 30/10/2013, 04/11/2013, 18/11/2013, 19/11/2013, 30/11/2013

; 22/09/2013, 24/09/2013, 05/10/2013, 10/10/2013, 20/10/2013, 23/10/2013

; 30/08/2013, 26/08/2013, 03/09/2013, 13/09/2013, 17/09/2013, 20/09/2013

; 18/08/2013, 16/08/2013, 14/08/2013, 13/08/2013, 12/08/2013, 11/08/2013

; 09/08/2013, 08/08/2013, 05/08/2013, 03/08/2013, 02/08/2013, 01/08/2013

; 31/07/2013 user/u structure (u.rw and u.namei\_r has been removed)

; 30/07/2013, 29/07/2013

; 28/07/2013 u.rw, u.namei\_r, u.ttyn, u.errn

; 26/07/2013, 25/07/2013, 24/07/2013, 17/07/2013, 16/07/2013, 14/07/2013

; 13/07/2013 kernel initialization additions & modifications

; 09/07/2013

; 20/06/2013 set date & time (for 'sysstime' system call)

; 04/06/2013 ecore (sysexec)

; 03/06/2013 p\_time (systime, sysmdate)

; 26/05/2013

; 24/05/2013 (end of core)

; 21/05/2013 com\_stat: owner and status of COM/serial port (1&2)

; 10/05/2013 tty modifications (keyboard functions)

; 26/04/2013 device numbers, structure modifications

; 11/03/2013

nproc equ 16 ; number of processes

nfiles equ 50

ntty equ 8 ; 8+1 -> 8 (10/05/2013)

nbuf equ 6

csgmnt equ 2000h ; 26/05/2013 (segment of process 1)

core equ 0 ; 19/04/2013

ecore equ 32768 - 64 ; 04/06/2013 (24/05/2013)

; (if total size of argument list and arguments is 128 bytes)

; maximum executable file size = 32768-(64+40+128-6) = 32530 bytes

; maximum stack size = 40 bytes (+6 bytes for 'IRET' at 32570)

; initial value of user's stack pointer = 32768-64-128-2 = 32574

; (sp=32768-args\_space-2 at the beginning of execution)

; argument list offset = 32768-64-128 = 32576 (if it is 128 bytes)

; 'u' structure offset (for the '/core' dump file) = 32704

; '/core' dump file size = 32768 bytes

; 08/03/2014

sdsegmnt equ 6C0h ; 256\*16 bytes (swap data segment size for 16 processes)

; 19/04/2013 Retro UNIX 8086 v1 feaure only !

;sdsegmnt equ 740h ; swap data segment (for user structures and registers)

; 30/08/2013

time\_count equ 4 ; 10 --> 4 01/02/2014

; 05/02/2014

; process status

;SFREE equ 0

;SRUN equ 1

;SWAIT equ 2

;SZOMB equ 3

;SSLEEP equ 4 ; Retro UNIX 8086 V1 extension (for sleep and wakeup)

user struc

; 10/10/2013

; 11/03/2013.

;Derived from UNIX v1 source code 'user' structure (ux).

;u.

sp\_ dw ? ; sp

usp dw ?

r0 dw ?

cdir dw ?

fp db 10 dup(?)

fofp dw ?

dirp dw ?

namep dw ?

off dw ?

base dw ?

count dw ?

nread dw ?

break\_ dw ? ; break

ttyp dw ?

dirbuf db 10 dup(?)

;pri dw ? ; 14/02/2014

quant db ? ; Retro UNIX 8086 v1 Feature only ! (uquant)

pri db ? ;

intr dw ?

quit dw ?

; emt dw ? ; 10/10/2013

ilgins dw ?

cdrv dw ? ; cdev

uid\_ db ? ; uid

ruid db ?

bsys db ?

uno db ?

; user/program segment (12/03/2013)

segmnt dw ? ; 12/03/2013 - Retro Unix 8086 v1 feature only !

; tty number (rtty, rcvt, wtty)

ttyn db ? ; 28/07/2013 - Retro Unix 8086 v1 feature only !

; last error number (reserved)

errn db ? ; 28/07/2013 - Retro Unix 8086 v1 feature only !

user ends

process struc

; 05/02/2014 ttys -> waitc (waiting channel, tty number)

; 17/09/2013 ttys (10 byte structure)

; 03/09/2013 ttyc (word -> byte) [ 10 bytes -> 9 bytes ]

; 14/08/2013 dska -> ttyc

; 11/03/2013.

;Derived from UNIX v1 source code 'proc' structure (ux).

;p.

pid dw nproc dup(?)

ppid dw nproc dup(?)

break dw nproc dup(?)

ttyc db nproc dup(?) ; console tty in Retro UNIX 8086 v1.

waitc db nproc dup(?) ; waiting channel in Retro UNIX 8086 v1.

link db nproc dup(?)

stat db nproc dup(?)

process ends

inode struc ; 11/03/2013.

;Derived from UNIX v1 source code 'inode' structure (ux).

;i.

flgs dw ?

nlks db ?

uid db ?

size\_ dw ? ; size

dskp dw 8 dup(?) ; 16 bytes

ctim dd ?

mtim dd ?

rsvd dw ? ; Reserved (ZERO/Undefined word for UNIX v1.)

inode ends

systm struc ; 11/03/2013.

;Derived from UNIX v1 source code 'systm' structure (ux).

;s.

dw ?

db 128 dup(?)

dw ?

db 64 dup (?)

time dd ?

syst dd ?

wait\_ dd ? ; wait

idlet dd ?

chrgt dd ?

drerr dw ?

systm ends

; fsp table entry (8 bytes) ;; 19/04/2013

; inum dw 0 ; inode number

; devnum dw 0 ; device number

; ofsp dw 0 ; offset pointer

; oc db 0 ; open count

; df db 0 ; deleted flag

;

phydrv struc ; 26/04/2013 (09/07/2013)

; Physical drv parameters of Retro UNIX 8086 v1 devices

; Retro UNIX 8086 v1 feature only !

err db 6 dup(?) ; error status (>0 means error)

pdn db 6 dup(?) ; physical drive number

spt dw 6 dup(?) ; sectors per track

hds dw 6 dup(?) ; heads

phydrv ends

; 14/07/2013

; UNIX v1 system calls

\_rele equ 0

\_exit equ 1

\_fork equ 2

\_read equ 3

\_write equ 4

\_open equ 5

\_close equ 6

\_wait equ 7

\_creat equ 8

\_link equ 9

\_unlink equ 10

\_exec equ 11

\_chdir equ 12

\_time equ 13

\_mkdir equ 14

\_chmod equ 15

\_chown equ 16

\_break equ 17

\_stat equ 18

\_seek equ 19

\_tell equ 20

\_mount equ 21

\_umount equ 22

\_setuid equ 23

\_getuid equ 24

\_stime equ 25

\_quit equ 26

\_intr equ 27

\_fstat equ 28

\_emt equ 29

\_mdate equ 30

\_stty equ 31

\_gtty equ 32

\_ilgins equ 33

\_sleep equ 34 ; Retro UNIX 8086 v1 feature only !

sys macro syscallnumber

; 14/07/2013

; Retro UNIX 8086 v1 system call.

mov ax, syscallnumber

int 20h

endm

.8086

UNIX SEGMENT PUBLIC PARA 'CODE'

assume cs:UNIX,ds:UNIX,es:UNIX,ss:UNIX

START:

; 11/03/2013

; include files according to original UNIX v1 (except ux.s)

; (u0.s, u1.s, u2.s, u3.s, u34.s, u5.s, u6.s, u7.s, u8.s, u9.s)

;

include u0.asm ; u0.s (with major modifications for 8086 PC)

include u1.asm ; u1.s

include u2.asm ; u2.s

include u3.asm ; u3.s

include u4.asm ; u4.s

include u5.asm ; u5.s

include u6.asm ; u6.s

include u7.asm ; u7.s

include u8.asm ; u8.s

include u9.asm ; u9.s

; RETRO UNIX 8086 v1 special/private procedures

;

;

epoch:

; 09/04/2013

; Retro UNIX 8086 v1 feature/procedure only!

; 'epoch' procedure prototype:

; UNIXCOPY.ASM, 10/03/2013

; 14/11/2012

; unixboot.asm (boot file configuration)

; version of "epoch" procedure in "unixproc.asm"

; 21/7/2012

; 15/7/2012

; 14/7/2012

; Erdogan Tan - RETRO UNIX v0.1

; compute current date and time as UNIX Epoch/Time

; UNIX Epoch: seconds since 1/1/1970 00:00:00

;

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

;

; 21/7/2012

;push bx

;push cx

mov ah, 02h ; Return Current Time

int 1Ah

xchg ch,cl

mov word ptr [hour], cx

xchg dh,dl

mov word ptr [second], dx

mov ah, 04h ; Return Current Date

int 1Ah

xchg ch,cl

mov word ptr [year], cx

xchg dh,dl

mov word ptr [month], dx

mov cx, 3030h

mov al, byte ptr [hour] ; Hour

; AL <= BCD number)

db 0D4h,10h ; Undocumented inst. AAM

; AH = AL / 10h

; AL = AL MOD 10h

aad ; AX= AH\*10+AL

mov byte ptr [hour], al

mov al, byte ptr [hour]+1 ; Minute

; AL <= BCD number)

db 0D4h,10h ; Undocumented inst. AAM

; AH = AL / 10h

; AL = AL MOD 10h

aad ; AX= AH\*10+AL

mov byte ptr [minute], al

mov al, byte ptr [second] ; Second

; AL <= BCD number)

db 0D4h,10h ; Undocumented inst. AAM

; AH = AL / 10h

; AL = AL MOD 10h

aad ; AX= AH\*10+AL

mov byte ptr [second], al

mov ax, word ptr [year] ; Year (century)

push ax

; AL <= BCD number)

db 0D4h,10h ; Undocumented inst. AAM

; AH = AL / 10h

; AL = AL MOD 10h

aad ; AX= AH\*10+AL

mov ah, 100

mul ah

mov word ptr [year], ax

pop ax

mov al, ah

; AL <= BCD number)

db 0D4h,10h ; Undocumented inst. AAM

; AH = AL / 10h

; AL = AL MOD 10h

aad ; AX= AH\*10+AL

add word ptr [year], ax

mov al, byte ptr [month] ; Month

; AL <= BCD number)

db 0D4h,10h ; Undocumented inst. AAM

; AH = AL / 10h

; AL = AL MOD 10h

aad ; AX= AH\*10+AL

mov byte ptr [month], al

mov al, byte ptr [month]+1 ; Day

; AL <= BCD number)

db 0D4h,10h ; Undocumented inst. AAM

; AH = AL / 10h

; AL = AL MOD 10h

aad ; AX= AH\*10+AL

mov byte ptr [Day], al

convert\_to\_epoch:

; Derived from DALLAS Semiconductor

; Application Note 31 (DS1602/DS1603)

; 6 May 1998

mov dx, word ptr [year]

sub dx, 1970

mov ax, 365

mul dx

xor bh, bh

mov bl, byte ptr [month]

dec bl

shl bl, 1

mov cx, word ptr DMonth[BX]

mov bl, byte ptr [Day]

dec bl

add ax, cx

adc dx, 0

add ax, bx

adc dx, 0

; DX:AX = days since 1/1/1970

mov cx, word ptr [year]

sub cx, 1969

shr cx, 1

shr cx, 1

; (year-1969)/4

add ax, cx

adc dx, 0

; + leap days since 1/1/1970

cmp byte ptr [month], 2 ; if past february

jna short @f

mov cx, word ptr [year]

and cx, 3 ; year mod 4

jnz short @f

; and if leap year

add ax, 1 ; add this year's leap day (february 29)

adc dx, 0

@@: ; compute seconds since 1/1/1970

mov bx, 24

call mul32

mov bl, byte ptr [hour]

add ax, bx

adc dx, 0

mov bx, 60

call mul32

mov bl, byte ptr [minute]

add ax, bx

adc dx, 0

mov bx, 60

call mul32

mov bl, byte ptr [second]

add ax, bx

adc dx, 0

; DX:AX -> seconds since 1/1/1970 00:00:00

; 21/7/2012

;pop cx

;pop bx

retn

mul32:

; push cx

mov cx, bx

mov bx, dx

mul cx

xchg ax, bx

push dx

mul cx

pop cx

add ax, cx

adc dx, 0

xchg bx, ax

xchg dx, bx

; pop cx

retn

set\_date\_time: ; 20/06/2013

convert\_from\_epoch:

; 20/06/2013

; Retro UNIX 8086 v1 feature/procedure only!

; 'convert\_from\_epoch' procedure prototype:

; UNIXCOPY.ASM, 10/03/2013

; 30/11/2012

; Derived from DALLAS Semiconductor

; Application Note 31 (DS1602/DS1603)

; 6 May 1998

;

; INPUT:

; DX:AX = Unix (Epoch) Time

;

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

;

mov cx, 60

call div32

;mov word ptr [imin], ax ; whole minutes

;mov word ptr [imin]+2, dx ; since 1/1/1970

mov word ptr [second], bx ; leftover seconds

; mov cx, 60

call div32

;mov word ptr [ihrs], ax ; whole hours

;mov word ptr [ihrs]+2, dx ; since 1/1/1970

mov word ptr [minute], bx ; leftover minutes

; mov cx, 24

mov cl, 24

call div32

;mov word ptr [iday], ax ; whole days

; since 1/1/1970

; mov word ptr [iday]+2, dx ; DX = 0

mov word ptr [hour], bx ; leftover hours

add ax, 365+366 ; whole day since

; 1/1/1968

; adc dx, 0 ; DX = 0

; mov word ptr [iday], ax

push ax

mov cx, (4\*365)+1 ; 4 years = 1461 days

call div32

pop cx

;mov word ptr [lday], ax ; count of quadyrs (4 years)

push bx

;mov word ptr [qday], bx ; days since quadyr began

cmp bx, 31 + 29 ; if past feb 29 then

cmc ; add this quadyr's leap day

adc ax, 0 ; to # of qadyrs (leap days)

;mov word ptr [lday], ax ; since 1968

;mov cx, word ptr [iday]

xchg cx, ax ; CX = lday, AX = iday

sub ax, cx ; iday - lday

mov cx, 365

;xor dx, dx ; DX = 0

; AX = iday-lday, DX = 0

call div32

;mov word ptr [iyrs], ax ; whole years since 1968

; jday = iday - (iyrs\*365) - lday

;mov word ptr [jday], bx ; days since 1/1 of current year

add ax, 1968 ; compute year

mov word ptr [year], ax

mov dx, ax

;mov ax, word ptr [qday]

pop ax

cmp ax, 365 ; if qday <= 365 and qday >= 60

ja short @f ; jday = jday +1

cmp ax, 60 ; if past 2/29 and leap year then

cmc ; add a leap day to the # of whole

adc bx, 0 ; days since 1/1 of current year

@@:

;mov word ptr [jday], bx

mov cx, 12 ; estimate month

xchg cx, bx ; CX = jday, BX = month

mov ax, 366 ; mday, max. days since 1/1 is 365

and dx, 11b ; year mod 4 (and dx, 3)

@@: ; Month calculation ; 0 to 11 (11 to 0)

cmp cx, ax ; mday = # of days passed from 1/1

jnb short @f

dec bx ; month = month - 1

shl bx, 1

mov ax, word ptr DMonth[BX] ; # elapsed days at 1st of month

shr bx, 1 ; bx = month - 1 (0 to 11)

cmp bx, 1 ; if month > 2 and year mod 4 = 0

jna short @b ; then mday = mday + 1

or dl, dl ; if past 2/29 and leap year then

jnz short @b ; add leap day (to mday)

inc ax ; mday = mday + 1

jmp short @b

@@:

inc bx ; -> bx = month, 1 to 12

mov word ptr [month], bx

sub cx, ax ; day = jday - mday + 1

inc cx

mov word ptr [day], cx

; ax, bx, cx, dx is changed at return

; output ->

; [year], [month], [day], [hour], [minute], [second]

; 20/06/2013

set\_date:

mov al, byte ptr [Year]+1

aam ; ah = al / 10, al = al mod 10

db 0D5h,10h ; Undocumented inst. AAD

; AL = AH \* 10h + AL

mov ch, al ; century (BCD)

mov al, byte ptr [Year]

aam ; ah = al / 10, al = al mod 10

db 0D5h,10h ; Undocumented inst. AAD

; AL = AH \* 10h + AL

mov cl, al ; year (BCD)

mov al, byte ptr [Month]

aam ; ah = al / 10, al = al mod 10

db 0D5h,10h ; Undocumented inst. AAD

; AL = AH \* 10h + AL

mov dh, al ; month (BCD)

mov al, byte ptr [Day]

aam ; ah = al / 10, al = al mod 10

db 0D5h,10h ; Undocumented inst. AAD

; AL = AH \* 10h + AL

mov dh, al ; day (BCD)

; Set real-time clock date

mov ah, 05h

int 1Ah

; retn

set\_time:

; Read real-time clock time

mov ah, 02h

int 1Ah

; DL = 1 or 0 (day light saving time)

mov al, byte ptr [Hour]

aam ; ah = al / 10, al = al mod 10

db 0D5h,10h ; Undocumented inst. AAD

; AL = AH \* 10h + AL

mov ch, al ; hour (BCD)

mov al, byte ptr [Minute]

aam ; ah = al / 10, al = al mod 10

db 0D5h,10h ; Undocumented inst. AAD

; AL = AH \* 10h + AL

mov cl, al ; minute (BCD)

mov al, byte ptr [Second]

aam ; ah = al / 10, al = al mod 10

db 0D5h,10h ; Undocumented inst. AAD

; AL = AH \* 10h + AL

mov dh, al ; second (BCD)

; Set real-time clock time

mov ah, 03h

int 1Ah

retn

div32:

; Input -> DX:AX = 32 bit dividend

; CX = 16 bit divisor

; output -> DX:AX = 32 bit quotient

; BX = 16 bit remainder

mov bx, dx

xchg ax, bx

xor dx, dx

div cx ; at first, divide DX

xchg ax, bx ; remainder is in DX

; now, BX has quotient

; save remainder

div cx ; so, DX\_AX divided and

; AX has quotient

; DX has remainder

xchg dx, bx ; finally, BX has remainder

retn

;; 13/07/2013

unixbootdrive: db 0

;;

; Following (data) section is derived from UNIX v1 'ux.s' file

; 11/03/2013

;

align 2

; 13/07/2013

sb0: db 4 dup(0) ; Retro UNIX 8086 v1 modification !

;systm:

;s: db 218 dup(?)

s: db 512 dup(0) ; Retro UNIX 8086 v1 modification !

;;inode:

;i: db 32 dup(0)

sb1: db 4 dup(0) ; Retro UNIX 8086 v1 modification !

mount: db 512 dup(0) ; Retro UNIX 8086 v1 modification !

;mount: db 1024 dup(0)

;inode:

i: db 32 dup(0)

;

;proc:

;p: db 9\*nproc dup(0) ; 03/09/2013

p: db 10\*nproc dup(0)

;tty: db ntty\*8 dup(0)

fsp: db nfiles\*8 dup(0)

bufp: db ((nbuf\*2)+4) dup(0) ; will be initialized (09/07/2013)

;;bufp: db ((nbuf\*2)+6) dup(0)

;;sb0: db 8 dup(0)

;sb0: db 4 dup(0) ; Retro UNIX 8086 v1 modification !

;;sb1: db 8 dup(0)

;sb1: db 4 dup(0) ; Retro UNIX 8086 v1 modification !

;swp: db 8 dup(0)

;;swp: db 4 dup(0) ; Retro UNIX 8086 v1 modification !

ii: dw 0

idev: dw 0 ; device number is 1 byte in Retro UNIX 8086 v1 !

cdev: dw 0 ; device number is 1 byte in Retro UNIX 8086 v1 !

;;deverr: db 12 dup(0)

;

; 26/04/2013 device/drive parameters

; Retro UNIX 8086 v1 feature only!

; there are 8 available Retro UNIX devices

;

; 'UNIX' device numbers (as in 'cdev' and 'u.cdrv')

; 0 -> root device (which has Retro UNIX 8086 v1 file system)

; 1 -> mounted device (which has Retro UNIX 8086 v1 file system)

; 'Retro UNIX 8086 v1' device numbers: (for disk I/O procedures)

; 0 -> fd0 (physical drive, floppy disk 1), physical drive number = 0

; 1 -> fd1 (physical drive, floppy disk 2), physical drive number = 1

; 2 -> hd0 (physical drive, hard disk 1), physical drive number = 80h

; 3 -> hd1 (physical drive, hard disk 2), physical drive number = 81h

; 4 -> hd2 (physical drive, hard disk 3), physical drive number = 82h

; 5 -> hd3 (physical drive, hard disk 4), physical drive number = 83h

rdev: db 0 ; root device number ; Retro UNIX 8086 v1 feature only!

; as above, for physical drives numbers in following table

mdev: db 0 ; mounted device number ; Retro UNIX 8086 v1 feature only!

; as above, for physical drives numbers in following table

; NOTE: the value of 'cdev' and 'u.drv' and 'idev' will be 0 or 1.

; 0 is for rdev, 1 is for mdev

drv: ; Retro UNIX 8086 v1 feature only!

drverr:

db 6 dup(0FFh) ; error status (>0 means error)

drvpdn:

db 6 dup(0FFh) ; physical drive number (FFh = invalid drive)

drvspt:

dw 6 dup(0) ; sectors per track

drvhds:

dw 6 dup(0) ; number of heads

;active: dw 0

active: db 0 ; 15/03/2013

brwdev: db 0 ; 26/04/2013 Retro UNIX 8086 v1 feature only !

;rfap: dw 0

;rkap: dw 0

;tcap: dw 0

;tcstate:dw 0

;tcerrc:dw 0

mnti: dw 0

;mntd: dw 0 ; device number is 1 byte in Retro UNIX 8086 v1 !

mpid: dw 0

;clockp: dw 0

rootdir:dw 0

;toutt: db 16 dup(0)

;touts: db 32 dup(0)

;runq: db 6 dup (0)

; 14/02/2014

; Major Modification: Retro UNIX 8086 v1 feature only!

; Single level run queue

; (in order to solve sleep/wakeup lock)

runq: dw 0

;wlist: db 40 dup(0)

;cc: db 30 dup(0)

;cf: db 31 dup(0)

;cl\_: db 31 dup(0) ; cl

;clist: db 510 dup(0)

imod: db 0

smod: db 0

mmod: db 0

;uquant: db 0 ; 14/02/2014 --> u.quant

sysflg: db 0

;pptiflg:db 0

;ttyoch: db 0

align 2

; Retro Unix 8086 v1 features only !

; 31/07/2013

; 07/04/2013

rw: db 0 ;; Read/Write sign

;; 07/08/2013 (reset in error routine)

;; mov word ptr [namei\_r], 0 -> namei\_r = 0, mkdir\_w = 0

; 26/07/2013

namei\_r: db 0 ; the caller is 'namei' sign for 'dskr' (ES=CS)

; 01/08/2013

mkdir\_w: db 0 ; the caller is 'mkdir' sign for 'dskw' (ES=CS)

;

align 2

; 09/04/2013 epoch variables

; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013

;

year: dw 1970

month: dw 1

day: dw 1

hour: dw 0

minute: dw 0

second: dw 0

DMonth:

dw 0

dw 31

dw 59

dw 90

dw 120

dw 151

dw 181

dw 212

dw 243

dw 273

dw 304

dw 334

; 10/05/2013

; Retro UNIX 8086 v1 feature only !

int09h: ; BIOS INT 09h handler (original)

dw 0 ; offset

dw 0 ; segment

; 03/06/2013

p\_time: dd 0 ; present time (for systime & sysmdate)

; 04/12/2013 ('putc', 'write\_tty' in U9.ASM)

crt\_start: dw 0 ; starting address in regen buffer

; NOTE: active page only

cursor\_posn: dw 8 dup(0) ; cursor positions for video pages

; 04/12/2013

active\_page: ; = ptty ('putc', 'write\_tty' in U9.ASM)

; 10/05/2013

; Retro UNIX 8086 v1 feature only !

ptty: db 0 ; current tty

;nxtty: db 0 ; next tty (will be switched to)

; 16/07/2013

;getctty: db 0 ; for using in 'getc' routine

; 12/08/2013

;AltKeyDown: db 0 ; INT 09h

align 2

; 03/03/2014

; Derived from IBM "pc-at"

; rombios source code (06/10/1985)

; 'dseg.inc'

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

; SYSTEM DATA AREA ;

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

BIOS\_BREAK db 0 ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED

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

; KEYBOARD DATA AREAS ;

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

KB\_FLAG db 0 ; KEYBOARD SHIFT STATE AND STATUS FLAGS

KB\_FLAG\_1 db 0 ; SECOND BYTE OF KEYBOARD STATUS

KB\_FLAG\_2 db 0 ; KEYBOARD LED FLAGS

KB\_FLAG\_3 db 0 ; KEYBOARD MODE STATE AND TYPE FLAGS

ALT\_INPUT db 0 ; STORAGE FOR ALTERNATE KEY PAD ENTRY

BUFFER\_START dW offset KB\_BUFFER ; OFFSET OF KEYBOARD BUFFER START

BUFFER\_END dW offset KB\_BUFFER + 32 ; OFFSET OF END OF BUFFER

BUFFER\_HEAD dw offset KB\_BUFFER ; POINTER TO HEAD OF KEYBOARD BUFFER

BUFFER\_TAIL dw offset KB\_BUFFER ; POINTER TO TAIL OF KEYBOARD BUFFER

; ------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY

KB\_BUFFER dw 16 DUP (0) ; ROOM FOR 15 SCAN CODE ENTRIES

;

;align 2

; 26/01/2014 'ttyl' lock table instead of 'ttyr' and 'ttyw'

;

; 16/08/2013 'ttypt' owner table -> 'ttyr', 'ttyw' lock table

; byte ptr [BX]+ttyl = owner/lock for read/write

; (process number = locked, 0 = unlocked/free)

; byte ptr [BX]+ttyr+1 = count of open for read&write

; (0 = free, >0 = in use)

;

;; Retro UNIX 8086 v1 feature only!

;;

;; (26/01/2014)

;; (13/01/2014)

;; 06/12/2013

;; <<<Major modification on TTY procedures>>>

;;

; Console TTY for process :

; 'sys fork' system call sets/copies parent process's

; console TTY number as child process's console TTY number.

; It is a zero based number (0 to 9) which is hold in 'p.ttyc'.

; Console TTY setting can be changed by 'sys stty' system call.

; Recent TTY for process:

; Recent TTY number during the last TTY read/write routine

; by process. 'u.ttyp' (word pointer) is used for that purpose.

; TTY num. of the last TTY Read is stored in low byte of 'u.ttyp'.

; TTY num. of the last TTY write is stored in high byte of 'u.ttyp.

;

; TTY 'Open' conditions: (06/12/2013 <--- 16/08/2013)

; 1) A process can open a free/unlocked tty or a tty

; which is locked by it or it's parent process. (13/01/2014)

; (Open count is increased by 1 while a new instance of

; tty is being open.)

; 2) The caller/process locks a tty if it is unlocked/free.

; 3) TTY open procedure sets 'u.ttyp' to related tty number + 1.

; Open for read procedure sets the low byte and open for

; write procedure sets the high byte.

; NOTE: TTY read and write procedures change these recent tty

; (u.ttyp) values. (06/12/2013)

;

; TTY 'close' conditions: (16/08/2013)

; 1) A tty is unlocked if it's open count becomes zero while

; closing it. (26/01/2014)

; (Open count is decreased by 1 when the instance of

; tty is closed.)

; 2) TTY close procedure resets low byte or high byte of

; 'u.ttyp' if it was set to related tty number + 1.

; Open for read procedure resets the low byte and open

; for write procedure resets the high byte. (06/12/2013)

;

; NOTE: 'tty' functionality of 'Retro UNIX 8086 v1' is almost

; different than original UNIX v1 (also v1 to recent

; unix sys v versions). Above logic/methods is/are

; developed by Erdogan Tan, for keeping 'multi screen',

; 'multi tasking' ability of 'Retro UNIX 8086 v1' (tty and

; process switching by 'ALT + Function keys' and

; for ensuring proper/stable process separation between

; pseudo TTYs and serial ports).

;

; 09/07/2014 (tty8, tty9)

; 24/09/2013 (tty0 to tty7)

ttychr: ; (0 to 9)

dw ntty+2 dup(0) ; ascii (lb) & scan code (hb) of keys

; per every pseudo tty (video page)

; 26/01/2014 'ttyl' lock table instead of 'ttyr' and 'ttyw'

; 13/01/2014 (COM1 & COM2 have been added to pseudo TTYs)

; (ntty -> ntty + 2)

; 16/08/2013 (open mode locks for pseudo TTYs)

; [ major tty locks (return error in any conflicts) ]

ttyl: ; Retro UNIX 8086 v1 feature only !

dw ntty+2 dup(0) ; opening locks for TTYs.

; 22/09/2013

wlist: db ntty+2 dup(0) ; wait channel list (0 to 9 for TTYs)

; 27/07/2014

tsleep: dw 0 ; Transmit sleep sign for port processes

; which use serial ports (COM1, COM2) as tty.

;; 16/07/2013

;; tty (keyboard) process/owner table (ttypt)

;ttypt: db ntty\*2 dup(0)

;; 12/07/2014 -> communication status data is not needed here

; <cancel>

; 16/07/2013

; 21/05/2013

;;com\_stat:

; 13/01/2014

;;com1\_stat:

;; db 0 ; COM1 line status

;; db 0 ; COM1 modem status

;;com2\_stat:

;; db 0 ; COM2 line status

;; db 0 ; COM2 modem status

; 16/08/2013

; Communication parameters for serial ports

; Retro UNIX 8086 v1 default:

;; 11100011b ; E3h

;; (111) Baud rate: 9600, (00) parity: none,

;; (0) stop bits: 1, (11) word length: 8 bits

;

; NOTE: Default value (E3h) will be set again

; after an initalization error, even if 'sys stty'

; system call changes the value before

; an initialization error in tty 'open' routine.

; (Serial port initialization is performed

; when a tty 'open' routine runs for

; COM1 or COM2 while the tty is free/closed.)

;; 12/07/2014 -> sp\_init set comm. parameters as 0E3h

;; 0 means serial port is not available

;;comprm: ; 25/06/2014

com1p: db 0 ;;0E3h

com2p: db 0 ;;0E3h

;Buffer:

;db ntty\*140 dup(0)

;db nbuf\*520 dup(0)

align 8

dd 0

Buffer: ; Retro UNIX 8086 v1 modification !

db nbuf\*516 dup(0)

;user:

u: db 64 dup (0) ; (Original Unix v1 'user' structure has 62 bytes)

; 14/07/2013

kernel\_init\_err\_msg:

db 0Dh, 0Ah

db 07h

db 'Kernel initialization ERROR !'

db 0Dh, 0Ah, 0

kernel\_init\_ok\_msg:

db 07h

db 'Welcome to Retro UNIX 8086 v1 Operating System !'

db 0Dh, 0Ah

db 'by Erdogan Tan - 15/04/2015'

db 0Dh, 0Ah, 0

panic\_msg:

db 0Dh, 0Ah, 07h

db 'ERROR: Kernel Panic !'

db 0Dh, 0Ah, 0

etc\_init\_err\_msg:

db 0Dh, 0Ah

db 07h

db 'ERROR: /etc/init !?'

db 0Dh, 0Ah, 0

align 2

; sstack:

; db 256 dup(0)

; 10/12/2013

; 'Enable Multi Tasking' system call (sys emt)

; (time-out enabling/disabling functionality)

; has been added to Retro UNIX 8086 v1 Kernel (in U1.ASM)

SizeOfFile equ $

; 08/03/2014 (system systack size = 256 - 64)

sstack equ SizeOfFile + 256 - 64

;sstack equ SizeOfFile + 256 ; 24/07/2013

UNIX ends

end START