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

;

; BOOT1.ASM (Only for bootable UNIX v1 file system on 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

; Bootable Unix (RUFS) File System - UNIX Kernel Loader (Boot) File

; 07/03/2013

; Derived from UNIXCOPY.ASM (25/02/2013)

;

; [ Last Modification: 14/07/2013 ]

;

; Retro Unix is a 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>

;

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

bsFSystemID equ 2 ; 'RUFS'

bsVolumeSerial equ 6 ; (4 bytes)

bsFDSign equ 10 ; 'fd'

bsDriveNumber equ 12 ; fd0 or fd1 (0 or 1)

bsReserved equ 13 ; 0 (512 bytes per sector)

bsSecPerTrack equ 14 ; 18 (9 or 15)

bsHeads equ 15 ; 2

bsTracks equ 16 ; 80

bs\_bf\_inode\_number equ 18 ; 0 or Boot/Startup File I-Number

bsInfoEndsign equ 20 ; '@'

ROOT\_DIR\_INODE\_NUMBER equ 41

kernel\_loading\_segment equ 1000h ; 09/07/2013

;; boot file space (segment 7E0h) = 33280 bytes,

;; kernel space (segment 1000h) = 65536 bytes

;;kernel\_loading\_segment equ 1800h ; 05/03/2013

.8086

BOOT1 SEGMENT PUBLIC 'CODE'

assume cs:BOOT1,ds:BOOT1,es:BOOT1,ss:BOOT1

START\_CODE:

proc\_start proc near

; 07/03/2013 (timer)

; 06/03/2013

; 05/03/2013

; 01/03/2013

; 25/02/2013

; 24/02/2013 (BOOT1.ASM)

; 08/12/2012 (UNIXCOPY)

;

; 30/11/2012 (UNIXBOOT)

;

mov ax, offset EndOfFile

mov word ptr [BSBUFFER], ax

add ax, 512

mov word ptr [SUPERBLOCK], ax

add ax, 512

mov word ptr [DISKBUFFER], ax

add ax, 512

mov word ptr [FILEBUFFER], ax

loc\_copy\_bootsector:

; cli

;mov ax, cs

;mov sp, sizeoffile + 1000h

;mov ss, ax

;mov es, ax

;mov ds, ax

;sti

; cld

;xor cx, cx

;mov ds, cx

mov ax, ds

mov cx, 7C0h

mov ds, cx

xor si, si

mov di, offset EndOfFile ; word ptr [BSBUFFER]

mov cx, 256

rep movsw

mov ds, ax

mov word ptr [EXTRA\_SEGMENT], ax ; RESET ; 06/03/2013

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

; Read Superblock

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

; 25/02/2013

loc\_read\_superblock:

; DL = Drive number

mov byte ptr [PhysicalDriveNumber], dl

mov bx, word ptr [SUPERBLOCK]

mov ax,0201h ; Read 1 sector

;mov cx,2 ; Read superblock

mov cl, 2 ; 07/03/2013 (ch=0)

xor dh,dh

int 13h

jnc short loc\_unix\_welcome

loc\_drv\_read\_error:

mov si, offset msg\_unix\_drv\_read\_error

call UNIX\_PRINTMSG

xor ah, ah

int 16h

int 19h

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

; Write message

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

loc\_unix\_welcome:

mov si, offset UNIX\_Welcome

call UNIX\_PRINTMSG

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

; Timer

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

; 07/03/2013

; Automatic (default) kernel loading with timer tick count

mov ax, ds ; cs

mov cx, 40h

mov es, cx

mov bx, 6Ch

mov cx, word ptr ES:[BX]

add cx, 182\*6 ; 60 seconds ?

mov word ptr [waiting\_count], cx

mov es, ax

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

; call command interpreter

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

loc\_call\_unix\_prompt:

call unix\_prompt

int 19h

proc\_start endp

UNIX\_PRINTMSG proc near

; 20/01/2013 'call unix\_printchr'

UNIX\_PRINTMSG\_LOOP:

lodsb ; Load byte at DS:SI to AL

and AL,AL

jz short UNIX\_PRINTMSG\_OK

mov AH,0Eh

mov BX,07h

int 10h ; BIOS Service func ( ah ) = 0Eh

; Write char as TTY

;AL-char BH-page BL-colo

;call unix\_printchr ; 20/01/2013

jmp short UNIX\_PRINTMSG\_LOOP

UNIX\_PRINTMSG\_OK:

retn

UNIX\_PRINTMSG endp

;unix\_printchr proc near

; ; 20/01/2013

; mov AH,0Eh

; mov BX,07h

; int 10h ; BIOS Service func ( ah ) = 0Eh

; ; Write char as TTY

; ;AL-char BH-page BL-color

; retn

;unix\_printchr endp

unix\_prompt proc near

; 06/03/2013

; 05/03/2013 (default kernel name: unix)

; 25/02/2013 BOOT1 version

; 8/12/2012

; Derived from

; proc\_dos\_prompt procedure of TRDOS,

; MAINPROG.ASM (1/1/2012).

;

; proc\_dos\_prompt (15/09/2011)

;

;push ds

;pop es

unix\_prompt\_1:

mov si, offset Boot\_Msg

call unix\_printmsg

unix\_prompt\_2:

mov ah, 03h

;mov bx, 07h

int 10h

mov byte ptr [CursorColumn],dl

;

unix\_prompt\_14: ; 07/03/2013

; automatic kernel loading timer

;

cmp byte ptr [def\_kernel], bh ; 0

ja short unix\_prompt\_3

unix\_prompt\_15:

hlt ; halt cpu until external interrupt

mov ah, 1h ; Get keystroke status

int 16h

; ZF = 0 if key pressed

jnz short unix\_prompt\_16

mov cx, 40h

mov ax, ds

mov bx, 6Ch ; mov si, 6Ch

mov es, cx

mov cx, word ptr ES:[BX] ; ES:[SI]

mov es, ax

cmp cx, word ptr [waiting\_count]

jb short unix\_prompt\_15

dec byte ptr [def\_kernel] ; FFh

mov si, offset UNIX\_CRLF

call unix\_printmsg

mov si, offset CommandBuffer

mov di, si

jmp short unix\_prompt\_17

unix\_prompt\_16:

inc byte ptr [def\_kernel]

unix\_prompt\_3:

mov si, offset CommandBuffer

call proc\_rw\_char

mov di, si

xor bx, bx

xor cx, cx

unix\_prompt\_4:

mov al, byte ptr [SI][BX]

inc bl

cmp al, 20h

ja short unix\_prompt\_6

jb short unix\_prompt\_10

cmp bl, 74 ; 75 ?

jb short unix\_prompt\_4

unix\_prompt\_11:

mov bx,07h

mov al,0Dh

mov ah,0Eh

int 10h

mov al,0Ah

int 10h

jmp unix\_prompt\_1 ; loop

unix\_prompt\_5:

mov al, byte ptr [SI][BX]

inc bl

cmp al,20h

jna short unix\_prompt\_7

unix\_prompt\_6:

stosb

inc cl

cmp bl, 74 ; 75 ?

jb short unix\_prompt\_5

unix\_prompt\_7:

xor al, al ; 0

unix\_prompt\_8:

mov byte ptr [DI], al

inc di

cmp bl, 74 ; 75 ?

jnb short unix\_prompt\_9

mov al, byte ptr [SI][BX]

inc bl

cmp al, 20h

jnb short unix\_prompt\_8

mov byte ptr [DI], 0

unix\_prompt\_9:

call command\_interpreter

xor al, al ; 07/03/2013

cmp byte ptr [unix\_reboot], al ; 0

ja short unix\_prompt\_13 ; 06/03/2013

unix\_prompt\_12:

mov cx, 74 ; 75 ?

mov di, offset CommandBuffer

xor al,al

rep stosb

jmp short unix\_prompt\_11 ; 06/03/2013

unix\_prompt\_10:

; 05/03/2013

xor al, al

cmp byte ptr [def\_kernel], al ; 0

ja short unix\_prompt\_11 ; 06/03/2013

;mov di, offset CommandBuffer

unix\_prompt\_17: ; 07/03/2013 (timer code jumps here)

mov ax, 'nu'

stosw

mov ax, 'xi'

stosw

xor al, al

stosb

;mov cl, 4

;jmp short unix\_prompt\_9

call loc\_load\_kernel ; jump/go to kernel

jmp short unix\_prompt\_12 ; error return only

unix\_prompt\_13: ; 06/03/2013

retn

unix\_prompt endp

proc\_rw\_char proc near

; 8/12/2012 (modification for UNIXCOPY.ASM)

; OUTPUT -> DS:SI = Entered String (ASCIIZ)

;

read\_next\_char:

xor ah,ah

int 16h

and al,al

jz short loc\_arrow

cmp al,0E0h

je short loc\_arrow

cmp al,08h

jne short char\_return

loc\_back:

mov bl,7

mov ah,3

int 10h

cmp dl,byte ptr [CursorColumn]

ja short prev\_column

loc\_beep:

mov ah, 0Eh

mov al, 7

int 10h

jmp short read\_next\_char

prev\_column:

dec dl

set\_cursor\_pos:

mov ah,02h

int 10h

mov bl, dl

sub bl,byte ptr [CursorColumn]

mov cx,1

mov ah,09h

mov al,20h

mov byte ptr [SI][BX],al

loc\_write\_it:

mov bl,7

int 10h

mov dx,word ptr [CursorColumn]

jmp short read\_next\_char

loc\_arrow:

cmp AH,4Bh

je short loc\_back

cmp AH,53h

je short loc\_back

jmp short read\_next\_char

char\_return:

mov bl,7

mov ah,3

int 10h

mov ah, dl

sub ah,byte ptr [CursorColumn]

cmp al,20h

jb short loc\_escape

cmp ah, 72 ; limit

ja short loc\_beep

mov bl, ah

xor ah, ah

mov word ptr [SI][BX],ax

mov ah, 0Eh

mov bl, 7

int 10h

jmp short read\_next\_char

pass\_escape:

cmp al,0Dh

jne short read\_next\_char

mov ah, 0Eh

mov bl,7

int 10h

mov al,0Ah

int 10h

retn

loc\_escape:

cmp al,1Bh

jne short pass\_escape

stc

retn

proc\_rw\_char endp

command\_interpreter proc near

; 06/03/2013 (loc\_load\_kernel)

; 25/02/2013 BOOT1 version

; 23/02/2013 ?/help

; 17/02/2013 namei, inode, iget

; 16/02/2013 fs, volume

; 21/01/2013 'ls -l'

; 20/01/2013 ls (dir modifications)

; 13/01/2013 chmod, chown, link

; 07/01/2013 show tabspace (div) modif.

; 06/01/2013 show

; 06/01/2013 rm, mkdir, rmdir modifications

; 05/01/2013 check file attributes

; 30/12/2012

; 24/12/2012 todos

; 16/12/2012

; 08/12/2012

;

lodsw

cl3:

cmp cl, 3

jb short cl2

ja cl5

; DIR

loc\_cmd\_dir: ; 05/01/2013 @b->@f, dir\_print modifications

cmp ax, 'id'

jne loc\_load\_kernel ; @f

lodsb

cmp al, 'r'

jne loc\_load\_kernel ; @f

;lodsb

;or al, al

;jnz loc\_load\_kernel ; @f

mov byte ptr [ls\_option], 0

inc si

dir\_getarg: ; 30/12/2012

lodsb

cmp al, 20h

je short dir\_getarg

jnb short dir\_namei

ls\_getarg3:

xor ax, ax

jmp short dir\_print

dir\_namei: ; 30/12/2012

dec si

mov word ptr [u\_namep], si

call name\_i

jc short ci\_error

; ax = i-number

dir\_print:

call print\_directory\_list

jnc short @f

ci\_error:

mov si, offset error\_msg

call unix\_printmsg

@@:

retn

; 23/02/2013

cl1:

cmp al, '?'

jne loc\_load\_kernel ; @b

;cmp ah, 0

;jne loc\_load\_kernel ; @f

mov si, offset Boot\_Commands

call UNIX\_PRINTMSG

@@:

retn

; 16/12/2012

cl2:

cmp cl, 2

jb short cl1 ; 23/02/2013

; jb @b

; LS (DIR)

loc\_cmd\_ls: ; 20/01/2013

cmp ax, 'sl'

jne short loc\_cmd\_cd ; 25/02/2013

;lodsb

;or al, al

;jnz short loc\_load\_kernel ; @b

mov byte ptr [ls\_option], 1

inc si

ls\_getarg1: ; 21/01/2013

lodsb

cmp al, 20h

je short ls\_getarg1

jb short ls\_getarg3

ls\_getarg2:

cmp al,'-'

jne short dir\_namei

lodsb

cmp al, 'l'

jne short ls\_getarg3

ls\_getarg4:

lodsb

inc byte ptr [ls\_option]

cmp al, 20h

je short dir\_getarg

jb short ls\_getarg3

dec byte ptr [ls\_option]

jmp short ls\_getarg3

; CD (CHDIR)

loc\_cmd\_cd:

cmp ax, 'dc'

jne short loc\_cmd\_fs ; 25/02/2023

;lodsb

;or al, al

;jnz short loc\_load\_kernel ; @f

inc si

ci\_cd\_getarg:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short ci\_cd\_getarg

jb short @f

; dec si

mov ax, word ptr [u\_namep]

mov word ptr [arg], ax

call sys\_chdir

jc ci\_error

mov si, word ptr [arg]

call update\_cdir\_string

@@:

retn

; FS (Volume) ; 16/02/2013 (File System / Volume Info)

loc\_cmd\_fs: cmp ax, 'sf'

jne loc\_load\_kernel ; @b

;lodsb

;or al, al

;jnz short loc\_load\_kernel ; @b

fs\_info\_print:

call print\_volume\_info

@@:

retn

cl5:

cmp cl, 5

ja cl6

jb cl4

; NAMEI ; 17/02/2013, print i-number of file/directory

loc\_cmd\_namei:

cmp ax, 'an'

jne short loc\_cmd\_inode

lodsw

cmp ax, 'em'

jne loc\_load\_kernel ; @f

lodsb

cmp al, 'i'

jne loc\_load\_kernel ; @f

;lodsb

;or al, al

;jnz short loc\_load\_kernel ; @f

inc si

namei\_sf1:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short namei\_sf1

jb short @f

namei\_sf2:

lodsb

cmp al, 20h

ja short namei\_sf2

dec si

xor al, al

mov byte ptr [SI], al

namei\_fsf:

call name\_i

jnc short namei\_iget

namei\_unix\_stc:

cmp ah, 0FFh

jb ci\_error

mov si, offset NotFound\_msg

call UNIX\_PRINTMSG

@@:

retn

namei\_iget:

call i\_get

namei\_print\_inum:

jc ci\_error

mov cx, ax

mov si, offset msgINumber

call UNIX\_PRINTMSG

mov ax, cx

mov cx, 3

call print\_decimal\_number

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

retn

; INODE ; 17/02/2013, print inode structure/details

loc\_cmd\_inode:

cmp ax, 'ni'

jne short loc\_load\_kernel ; @b

lodsw

cmp ax, 'do'

jne short loc\_load\_kernel ; @b

lodsb

cmp al, 'e'

jne short loc\_load\_kernel ; @b

;lodsb

;or al, al

;jnz short loc\_load\_kernel ; @b

inc si

inode\_getarg1:

mov bx, si

lodsb

cmp al, 20h

je short inode\_getarg1

ja short inode\_getarg2

mov ax, word ptr [ii]

jmp short @f

inode\_getarg2:

lodsb

cmp al, 20h

ja short inode\_getarg2

dec si

xor ax, ax

mov byte ptr [SI], al

mov si, bx

@@:

call show\_inode

jc ci\_error

@@:

retn

cl4:

;cmp cl, 4

;jb cl3

; SHOW

loc\_cmd\_show:

; 06/01/2013

cmp ax, 'hs'

jne short loc\_load\_kernel ; loc\_cmd\_unix ; 05/03/2013

lodsw

cmp ax, 'wo'

jne short loc\_load\_kernel ; @b

;lodsb

;or al, al

;jnz short loc\_load\_kernel ; @b

inc si

show\_uf1:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short show\_uf1

jb short @f

show\_uf2:

lodsb

cmp al, 20h

ja short show\_uf2

xor al, al

mov byte ptr [SI]-1, al

show\_uf3:

call show\_file

jc ci\_error

@@:

retn

; UNIX (default kernel name) ; 06/03/2013

;loc\_cmd\_unix:

loc\_load\_kernel: ; 07/03/2013

; 06/03/2013

mov word ptr [u\_namep], offset CommandBuffer

call load\_kernel

jnc short @f

cmp byte ptr [def\_kernel], 0FFh ; auto loading

jb namei\_unix\_stc

; no error msg when it was auto kernel loading

retn

@@:

mov si, offset UNIX\_CRLF

call unix\_printmsg

; 14/07/2013

mov dl, byte ptr [PhysicalDriveNumber]

xor dh, dh

mov ax, kernel\_loading\_segment ; 1000h

mov ds, ax

mov es, ax

cli

mov ss, ax

mov sp, 32766 ; 09/07/2013

; FFFEh

sti

mov bx, offset EndOfFile ; Relocated BS buffer address

mov bp, sp

mov cx, cs ; 07/03/2013 (CX = Buffer segment)

; MASM.EXE don't accept

; jmp 1000h:0000h

; for OP Code: EA00000010

db 0EAh

dw 0

dw kernel\_loading\_segment ; 09/07/2013

cl6: ; 16/02/2013

cmp cl, 6

ja short cl8

; REBOOT ; 25/02/201

loc\_cmd\_reboot:

cmp ax, 'er'

jne short loc\_load\_kernel ; @b

lodsw

cmp ax, 'ob'

jne short loc\_load\_kernel ; @b

lodsw

cmp ax, 'to'

jne short loc\_load\_kernel ; @b

;lodsb

;or al, al

;jnz short loc\_load\_kernel ; @f

mov byte ptr [unix\_reboot], 1

@@:

retn

cl8:

cmp cl, 8

ja short @b ; bad command or file name

jb short loc\_load\_kernel ; @b

; BOOTFILE

loc\_cmd\_bootfile:

cmp ax, 'ob'

jne short loc\_load\_kernel ; @b

lodsw

cmp ax, 'to'

jne short loc\_load\_kernel ; @b

lodsw

cmp ax, 'if'

jne short loc\_load\_kernel ; @b

lodsw

cmp ax, 'el'

jne short loc\_load\_kernel ; @b

;lodsb

;or al, al

;jnz short loc\_load\_kernel ; @b

mov si, word ptr [BSBuffer] ; 06/03/2013

add si, bs\_BF\_inode\_Number

mov ax, word ptr [SI]

; and ax, ax

; jnz short @f

;ci\_no\_bootfile:

; mov si, offset msg\_Startup\_File\_Not\_Exists

; call UNIX\_PRINTMSG

; retn

@@:

call find\_bfn

jc ci\_error

ci\_move\_bfn\_1:

mov si, offset u\_dirbuf + 2

mov di, offset Boot\_File\_Name

mov cx, 8

ci\_move\_bfn\_2:

lodsb

and al, al

jnz short @f

mov byte ptr [DI], al ; 0

@@:

stosb

loop ci\_move\_bfn\_2

call proc\_display\_startupfile\_info

retn

command\_interpreter endp

update\_cdir\_string proc near

; 13/01/2013 bugfix

; 10/12/2012

; 09/12/2012

; input -> SI= chdir argument

ucds\_0:

mov bx, offset unix\_cdir

inc bx ; 13/01/2013

mov di, bx

lodsb

cmp al, '/'

jne short @f

xor dx, dx

mov word ptr [CDirOffset], dx

jmp short ucds\_6

@@:

mov dx, word ptr [CDirOffset]

; 13/01/2013

or dx, dx

jz short @f

add di, dx

mov byte ptr [DI], '/'

inc di

;

jmp short @f

ucds\_8:

inc di

ucds\_6:

lodsb

cmp al, '/'

je short ucds\_6

@@:

or al, al

jz short ucds\_5

cmp al, '.'

jne short ucds\_3

lodsb

cmp al, '.'

je short ucds\_2 ; dotdot

ucds\_1: ;dot

cmp al, '/'

je short ucds\_6

or al, al

jz short ucds\_5

mov ah, '.'

xchg ah, al

stosw

jmp short ucds\_6

ucds\_2: ; dotdot

cmp di, bx

ja short @f

xor dx, dx

mov byte ptr [DI], dl ; 0

jmp short ucds\_7

@@: ; 13/01/2013

dec di

@@: ; move back

dec di ; 13/01/2013

mov al, byte ptr [DI]

cmp al, '/'

jne short @b ; 13/01/2013

jmp short ucds\_8

ucds\_4:

stosb

jmp short ucds\_6

ucds\_3:

stosb

lodsb

cmp al, '/'

je short ucds\_4

and al, al

jnz short ucds\_3

ucds\_5: ; 13/01/2013

cmp di, bx

jna short ucds\_9

dec di

cmp byte ptr [DI], '/'

je short ucds\_9

inc di

ucds\_9:

; 13/01/2013

mov byte ptr [DI], al ; 0

mov dx, di

sub dx, bx

ucds\_7:

mov word ptr [CDirOffset], dx

retn

update\_cdir\_string endp

print\_directory\_list proc near

; 23/02/2013 long list printing (list\_count)

; 03/02/2013

; 22/01/2013 ls -l command feature

; 21/01/2013 dir/ls options

; 20/01/2013 directory sign ("/")

; 30/12/2012

or ax, ax ; i-number of directory

jnz short @f

; 09/12/2012

pdl\_0:

mov ax, word ptr [u\_cdir]

@@:

call i\_get

jc short @f ; 20/01/2013 ; jc short pdl\_9

test word ptr [inode\_flgs], 4000h ; directory i-node ?

jnz short pdl\_2

pdl\_1:

mov ah, 0FFh ; error number

stc

@@: ; 20/01/2013

;jmp short pdl\_9

retn

pdl\_2:

; 25/02/2013

mov si, offset unix\_cdrv ; print current directory

call unix\_printmsg

;

;mov ax, word ptr [inode\_size]

;mov word ptr [u\_dirp], ax ; put size of directory in u.dirp

;xor ax, ax

xor ah, ah

mov word ptr [u\_off], ax ; u.off is file offset used by user

;mov word ptr [u\_fofp], offset u.off

; u.fofp is a pointer to the offset portion

; of fsp entry

mov byte ptr [list\_count], al ; 0 ; 23/02/2013

pdl\_3:

mov word ptr [u\_base], offset u\_dirbuf

; u.dirbuf holds a file name copied from

; a directory

mov word ptr [u\_count], 10

; u.dirbuff holds a file name copied from

; a directory

mov ax, word ptr [ii]

call read\_i ; read 10 bytes of file with i-number (R1)

; i.e. read a directory entry

jc short @b ; jc short pdl\_9

mov cx, word ptr [u\_nread]

or cx, cx

jna short pdl\_1 ; gives error return

mov bx, word ptr [u\_dirbuf]

and bx, bx

jz pdl\_8

pdl\_4:

mov si, offset u\_dirbuf + 2 ; r3, points to file name of directory entry

mov cx, 8 ; max. file name length

mov di, offset DirFileName + 1 ; boot\_File\_Name

pdl\_5:

lodsb ; mov al, byte ptr [SI], inc si

or al, al

jz short pdl\_6 ; 3f. If char is nul, then the last char in string has

; been compared

stosb ; mov byte ptr [DI], al, inc di

loop pdl\_5

pdl\_6:

; 21/01/2013

mov si, offset UNIX\_CRLF

call unix\_printmsg

cmp byte ptr [ls\_option], 1

je short pdl\_7

;mov al, 0

mov byte ptr [DI], al

jb short pdl\_13

pdl\_7:

; 20/01/2013

push di

mov ax, word ptr [ii]

mov word ptr [pdir], ax

mov ax, word ptr [u\_dirbuf]

call i\_get

pop di

jc pdl\_9

; 22/01/2012

cmp byte ptr [ls\_option], 1

jna short @f

pdl\_11: ; 21/01/2013 ; Inode number

mov ax, word ptr [u\_dirbuf]

mov cx, 3 ; 03/02/2013

call print\_decimal\_number

jmp short pdl\_10

@@:

mov ax, word ptr [inode\_flgs]

test ah, 40h ; 'directory' flag

jz short pdl\_10

mov si, offset u\_dirbuf + 2

lodsb

@@:

cmp al, '.' ; '.'

jne short @f

lodsb

or al, al

jz short pdl\_10

jmp short @b

@@:

mov al, '/'

mov byte ptr [DI], al

inc di

pdl\_10:

; 21/03/2013

xor al, al

mov byte ptr [DI], al

pdl\_13: ; File/Directory name

inc byte ptr [list\_count] ; 23/02/2013

mov si, offset DirFileName

call unix\_printmsg

; 22/01/2013

cmp byte ptr [ls\_option], 1

je pdl\_12 ; 03/02/2013 short -> near

jb pdl\_8 ; 23/02/2013

; 03/02/2013

@@: ; Owner (uid)

;xor bh, bh ; mov bh, 0

mov ah, 03h ; get cursor position and size.

int 10h

cmp dl, 13

jnb short @f

mov al, 20h

call putc

jmp short @b

@@:

xor ah, ah

mov al, byte ptr [inode\_uid]

mov cx, 3

call print\_decimal\_number

@@:

mov al, 20h

call putc

mov al, 20h

call putc

@@: ; Flags/Attributes

mov dx, word ptr [inode\_flgs]

mov cl, '-'

shl dh, 1

shl dh, 1

jnc short @f

add al, 'd'-'-'

@@:

add al, cl

call putc

shl dl, 1

shl dl, 1

shl dl, 1

shl dl, 1

jnc short @f

add al, 'x'-'-'

@@:

add al, cl

call putc

shl dl, 1

jnc short @f

add al, 'r'-'-'

@@:

add al, cl

call putc

shl dl, 1

jnc short @f

add al, 'w'-'-'

@@:

add al, cl

call putc

shl dl, 1

jnc short @f

add al, 'r'-'-'

@@:

add al, '-'

call putc

shl dl, 1

jnc short @f

add al, 'w'-'-'

@@:

add al, cl

call putc

mov al, 20h

call putc

@@: ; File Size ; 03/02/2013

mov ax, word ptr [inode\_size]

;mov cx, 5

mov cl, 5

call print\_decimal\_number

@@:

mov al, 20h

call putc

mov al, 20h

call putc

@@: ; 03/02/2013 ; File creation date & time

;mov ax, word ptr [inode\_ctim]

;mov dx, word ptr [inode\_ctim]+2

; 23/02/2013 ; File last modification date & time

mov ax, word ptr [inode\_mtim]

mov dx, word ptr [inode\_mtim]+2

call convert\_from\_epoch

; cx = day

mov ax, cx ; word ptr [day]

mov si, offset dec\_num

mov bx, si

add bx, 2

; mov cx, 2

mov cl, 2

call proc\_bin\_to\_decimal

mov byte ptr [BX], '/'

mov si, bx

inc si

mov ax, word ptr [month]

; mov cx, 2

mov cl, 2

call proc\_bin\_to\_decimal

add bx, 3

mov byte ptr [BX], '/'

mov si, bx

inc si

mov ax, word ptr [year]

;mov cx, 4

mov cl, 4

call proc\_bin\_to\_decimal

mov si, offset dec\_num

call unix\_printmsg

mov al, 20h

call putc

mov si, offset dec\_num

mov bx, si

mov ax, word ptr [hour]

; mov cx, 2

mov cl, 2

call proc\_bin\_to\_decimal

add bx, 2

mov byte ptr [BX],':'

mov si, bx

inc si

mov ax, word ptr [minute]

; mov cx, 2

mov cl, 2

call proc\_bin\_to\_decimal

add bx, 3

;mov byte ptr [BX], ':'

;mov si, bx

;inc si

;mov ax, word ptr [second]

;;mov cx, 2

;mov cl, 2

;call proc\_bin\_to\_decimal

;add bx,

xor al, al

mov byte ptr [BX], al

mov si, offset dec\_num

call unix\_printmsg

pdl\_12:

mov ax, word ptr [pdir]

call i\_get

jc pdl\_9

pdl\_8:

; 30/12/2012

mov ax, word ptr [u\_off]

cmp ax, word ptr [inode\_size]

jnb short @f ; 22/02/2013 ; jb pdl\_3

; 23/02/2013

cmp byte ptr [list\_count], 21

jb pdl\_3

xor ah, ah

mov byte ptr [list\_count], ah ; 0

int 16h

cmp al, 1Bh ; ESC key

jne pdl\_3

@@:

mov si, offset UNIX\_CRLF

call unix\_printmsg

pdl\_9:

retn

putc: ; 22/01/2013

mov ah, 0Eh

;mov bx, 07h

int 10h

xor al, al

retn

print\_directory\_list endp

sys\_chdir proc near

; 09/12/2012 unixcopy.asm

; Retro UNIX v1 FS file import/export version

; of syschdir function

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;

; RETRO UNIX v1 FS

; syschdir:

; makes the directory specified in the argument

; the current directory

; mov word ptr [u\_namep], si

syschdir\_0:

call name\_i

jc short syschdir\_5

syschdir\_1:

call i\_get

jc short syschdir\_5

syschdir\_2:

test word ptr [inode\_flgs], 4000h ; directory i-node ?

jnz short syschdir\_4

syschdir\_3:

mov ah, 0FFh

stc

retn

syschdir\_4:

mov word ptr [u\_cdir], ax

; mov dx, word ptr [cdev]

; mov word ptr [u\_cdev], dx

syschdir\_5:

retn

sys\_chdir endp

show\_file proc near

; 05/03/2013

; 07/01/2013

; 06/01/2013

; derived from TRDOS command interpreter file (CMDINTR.ASM)

; 'show' procedure (13/09/2011)

call name\_i

jc short suf\_4

call i\_get

jc short suf\_4

test word ptr [inode\_flgs], 4000h ; Directory

jnz short suf\_4

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

mov ax, word ptr [inode\_size]

mov dx, 512

cmp ax, dx

jna short suf\_1

mov ax, dx

suf\_1:

xor dx, dx

mov word ptr [u\_off], dx

mov cx, 22

suf\_2:

push cx

mov word ptr [u\_count], ax

mov ax, word ptr [FILEBUFFER]

mov word ptr [u\_base], ax

mov ax, word ptr [ii] ; word ptr [u\_dirbuf]

call read\_i

pop cx

jc short suf\_4

mov di, word ptr [u\_nread]

or di, di

jz short suf\_4

mov si, word ptr [FILEBUFFER]

jmp short suf\_6

suf\_3:

and cx, cx

jnz short suf\_6

xor ah, ah

int 16h

cmp al, 1Bh ; ESCAPE Key

jne short suf\_5

suf\_4:

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

retn

suf\_5:

mov cx, 20

suf\_6:

xor bh, bh ; mov bh, 0

mov bl, 7

lodsb

cmp al, 0Dh ; ENTER/RETURN Char

jne short suf\_7

dec cx

jmp short suf\_8

suf\_7:

cmp al, 09h ; TAB Space Char

je short suf\_10

suf\_8:

mov ah, 0Eh

;xor bh, bh ; mov bh, 0

;mov bl, 7

int 10h

suf\_9:

dec di

jnz short suf\_3

mov ax, word ptr [u\_nread]

jmp short suf\_2

suf\_10:

push cx

;xor bh, bh ; mov bh, 0

mov ah, 03h ; get cursor position and size.

int 10h

mov al, dl

mov cx, 8

;suf\_11a:

; cmp al, cl

; jb short suf\_11b

; sub al, cl

; jmp short suf\_11a

;suf\_11b:

; sub cl, al

suf\_11:

; 07/01/2013

xor ah, ah

div cl

sub cl, ah

;

mov al, 20h

mov ah, 0Eh

;mov bl, 7 ; char color attribute

suf\_12:

int 10h

loop suf\_12

pop cx

jmp short suf\_9

show\_file endp

name\_i proc near

; 05/01/2013

; 09/12/2012 unixcopy.asm

; Retro UNIX v1 FS file import/export version

; 31/10/2012

; 14/10/2012

; 07/10/2012

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;

; RETRO UNIX v1 FS

;

; return i-number of file (in AX)

;

; input:

; u\_namep = pointer to file path name

; u\_cdir = i-number of users directory

; ;;u\_cdev = device number

; output:

; cf= 0 -> no error, i-number in AX (R1)

; cf= 1 -> error code in AX

;

mov si, word ptr [u\_namep]

cmp byte ptr [SI], '/' ; is first char in file name a /

jne short @f

mov ax, ROOT\_DIR\_INODE\_NUMBER ; 41

; Put i-number of root directory in R1

; xor dx, dx

inc si ; go to next char

mov word ptr [u\_namep], si

jmp short namei\_0

@@:

;mov dx, word ptr [u\_cdev]

mov ax, word ptr [u\_cdir]

; put i-number of current directory in R1

namei\_0:

;mov word ptr [cdev], dx

; device file for users directory into cdev

; 1

cmp byte ptr [SI], 0 ; is the character in file name a nul

jna short namei\_7 ;nig

namei\_1: ; 1

; get i-node with i-number r1

call i\_get

jc short namei\_7

test word ptr [inode\_flgs], 4000h ; directory i-node ?

;jz short namei\_6 ; got an error

jnz short @f

;nib:

namei\_6:

mov ah, 0FFh ; Error code

stc

;nig:

namei\_7:

retn

@@:

mov ax, word ptr [inode\_size]

mov word ptr [u\_dirp], ax ; put size of directory in u.dirp

xor ax, ax

mov word ptr [u\_off], ax ; u.off is file offset used by user

;mov word ptr [u\_fofp], offset u.off

; u.fofp is a pointer to the offset portion

; of fsp entry

namei\_2: ; 2

mov word ptr [u\_base], offset u\_dirbuf

; u.dirbuf holds a file name copied from

; a directory

mov word ptr [u\_count], 10

mov ax, word ptr [ii]

call read\_i ; read 10 bytes of file with i-number (R1)

; i.e. read a directory entry

jc short namei\_7

mov cx, word ptr [u\_nread]

or cx, cx

jna short namei\_6 ; nib ; gives error return

mov bx, word ptr [u\_dirbuf]

and bx, bx

jnz short namei\_3 ; 3f. branch when active directory entry

; (i-node word in entry non zero)

mov ax, word ptr [u\_off]

sub ax, 10

mov word ptr [u\_dirp], ax

jmp short namei\_2 ; 2b

namei\_3: ; 3

mov si, word ptr [u\_namep] ; r2, u.namep points into a file name string

mov di, offset u\_dirbuf + 2 ; r3, points to file name of directory entry

mov dx, offset u\_dirbuf + 10

@@: ; 3

lodsb ; mov al, byte ptr [SI], inc si (al = r4)

or al, al

jz short namei\_4 ; 3f. If char is nul, then the last char in string has

; been compared

cmp al, "/" ; is char a "/"

je short namei\_4 ; 3f

cmp di,dx ; offset u\_dirbuf + 10 ; r3,

; have i checked all 8 bytes of file name

je short @b ; 3b

scasb ; cmpb (r3)+, r4 (DI=R3, AL=R4)

; compare char in u.namep string to file name char

; read from

je short @b ; directory; brach if chars match

jmp short namei\_2 ; 2b

; File names do not match, go to next directory entry

namei\_4: ; 3

cmp di, dx ; offset u\_dirbuf + 10 ; r3,

; if equal all 8 bytes were matched

je short namei\_5 ; 3f

mov ah, byte ptr [DI]

;inc di ; 05/01/2013

and ah, ah ; tstb (r3)+, bne 2b

jnz short namei\_2 ; 2b

namei\_5: ; 3

mov word ptr [u\_namep], si ; r2

; u.namep points to char following a "/" or nul

;mov bx, word ptr [u\_dirbuf] ; r1

and al, al ; r4. If r4=0 the end of file name reached,

; if r4="/" then go to next directory

mov ax, bx

jnz namei\_1 ; 1b

retn

name\_i endp

read\_i proc near

; 06/03/2013 (kernel loading segment)

; 05/03/2013

; 14/10/2012

; Boot sector version of "readi" procedure

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;;AX (R1) = i-number

; RETRO UNIX v1 FS

; Boot sector version

;

; read from an i-node

;

xor dx, dx ; 0

mov word ptr [u\_nread], dx ; accumulated number of bytes transmitted

cmp word ptr [u\_count], dx ; is number of byte to read greater than 0

jna short read\_inode\_retn

read\_inode\_1:

; AX = I-Number

push ax

call i\_get ; get i-node into i-node section of core

mov dx, word ptr [inode\_size] ; file size in bytes in r2 (DX)

sub dx, word ptr [u\_off] ; subtract file offset

jna short read\_inode\_3

cmp dx, word ptr [u\_count]

; are enough bytes left in file to carry out read

jnb short read\_inode\_2

mov word ptr [u\_count], dx

read\_inode\_2:

call m\_get ; returns physical block number of block in file

; where offset points

; AX = Physical block number

call dsk\_rd ; read in block, BX points to 1st word of data in

; buffer

jc short read\_inode\_3

readinode\_sioreg:

mov si, word ptr [u\_off] ; R2

mov cx, si ; cx = R3, si = R2

or cx, 0FE00h ; set bits 9...15 of file offset in R3

and si, 1FFh ; calculate file offset mod 512

add si, bx ; word ptr [DISKBUFFER] ; si now points to 1st byte in buffer

; where data is to be placed

mov di, word ptr [u\_base] ; R1

neg cx ; 512 - file offset(mod512) in R3 (cx)

cmp cx, word ptr [u\_count]

jna short @f ; 2f

mov cx, word ptr [u\_count]

@@:

add word ptr [u\_nread], cx ; r3 + number of bytes

; xmitted during write is put into

; u\_nread

sub word ptr [u\_count], cx

add word ptr [u\_base], cx ; points to 1st of remaining

; data bytes

add word ptr [u\_off], cx ; new file offset = number

; of bytes done + old file offset

; end of readinode\_sioreg

; DI = file (user data) offset

; SI = sector (I/O) buffer offset

; CX = byte count

; 06/03/2013

mov ax, word ptr [EXTRA\_SEGMENT] ; kernel loading segment or CS/DS

mov es, ax

rep movsb

mov ax, ds ; 06/03/2013

mov es, ax

pop ax

cmp word ptr [u\_count], 0

ja short read\_inode\_1

retn

read\_inode\_3:

pop ax ; i-number

read\_inode\_retn:

retn

read\_i endp

i\_get proc near

; 02/03/2013

; 24/02/2013 BOOT1 version

; 18/11/2012 unix boot file configuration version

; of "iget" procedure.

; 16/9/2012

; 14/7/2012

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;; AX=R1

; RETRO UNIX v1 FS

;; return => if cf=1 error number in [Error]

cmp ax, word ptr [ii] ; AX (R1) = i-number of current file

je short iget\_4

iget\_1:

; 24/02/2013

;mov dl, byte ptr [imod]

;and dl, dl ; has i-node of current file been modified ?

;jz short iget\_2

;xor dl, dl ; mov al, 0

;mov byte ptr [imod], dl

;push ax

;mov ax, word ptr [ii]

;inc dl ; mov dl, 1

;; dl = 1 = write

;call i\_calc

;pop dx

;jc short iget\_4

;mov ax, dx

iget\_2:

and ax, ax

jz short iget\_3

mov word ptr [ii], ax

;xor dl, dl ; 02/03/2013

; dl = 0 = read

call i\_calc

iget\_3:

mov ax, word ptr [ii]

iget\_4:

retn

i\_get endp

i\_calc proc near

; 05/03/2013

; 24/02/2013 BOOT1 version

; 18/11/2012 unix boot file configuration version

; of "icalc" procedure.

; 17/8/2012

; 14/7/2012

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;; AX=R1

; 0 = read, 1 = write

; RETRO UNIX v1 FS

;

; i-node is located in block (i+47)/16 and

; begins 32\*(i+47) mod 16 bytes from its start

;; return => if cf=1 error number in [Error]

;;; input -> dl = 0 -> read, 1 = Write

;mov byte ptr [rw], dl

add ax, 47 ; add 47 to inode number

push ax ; R1 -> -(SP)

shr ax, 1 ; divide by 16

shr ax, 1

shr ax, 1

shr ax, 1

; ax contains block number of block in which

; inode exists

call dsk\_rd

pop dx

jc short icalc\_2

icalc\_1:

and dx, 0Fh ; (i+47) mod 16

shl dx, 1

shl dx, 1

shl dx, 1

shl dx, 1

shl dx, 1

; DX = 32 \* ((i+47) mod 16)

; DX (R5) points to first word in i-node i.

mov di, offset inode

; inode is address of first word of current inode

mov cx, 16 ; CX = R3

mov si, word ptr [DISKBUFFER]

add si, dx

; copy new i-node into inode area of (core) memory

rep movsw

icalc\_2:

retn

i\_calc endp

dsk\_rd proc near

; 06/03/2013

; 05/03/2013

; 28/11/2012 BugFix

; 20/10/2012 (buff\_s)

; 14/10/2012

; fd boot sector version of "dskrd" procedure

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

; RETRO UNIX v1 FS

; floppy disk boot sector version

;; return => if cf=1 error number in [Error]

; ax = sector/block number

mov bx, word ptr [DISKBUFFER]

cmp ax, word ptr [buff\_s] ; buffer sector

je short dsk\_rd\_3

mov si, ax

xor ch, ch

mov cl, 4 ; Retry count

dsk\_rd\_1:

push cx

mov dx, 18 ; Sectors per track

div dl

mov cl, ah ; Sector (zero based)

inc cl ; To make it 1 based

shr al, 1 ; Convert Track to Cylinder

adc dh, 0 ; Heads (0 or 1)

mov dl, byte ptr [PhysicalDriveNumber]

mov ch, al

mov ah, 2 ; 2=read

mov al, 01h

int 13h ; BIOS Service func ( ah ) = 2

; Read disk sectors

; BIOS Service func ( ah ) = 3

; Write disk sectors

;AL-sec num CH-cyl CL-sec

; DH-head DL-drive ES:BX-buffer

;CF-flag AH-stat AL-sec read

pop cx

jnc short dsk\_rd\_2

loop dsk\_rd\_1

retn ; 06/03/2013

dsk\_rd\_2:

mov word ptr [buff\_s], si

dsk\_rd\_3:

retn

dsk\_rd endp

m\_get proc near

; 05/03/2013

; 03/03/2013

; 01/03/2013

; 24/02/2013 BOOT1 version

; 18/11/2012

; 14/11/2012 unix boot file configuration version

; of "mget" procedure

; 31/10/2012

; 20/10/2012

; 19/8/2012

; 13/8/2012

; 27/7/2012

; 21/7/2012

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;; return -> AX=R1

; RETRO UNIX v1 FS

; initialization/format version

; cf -> 1 = error (no free block)

;; contents of bx, cx, dx will be destroyed

mget\_0:

mov bl, byte ptr [u\_off]+1

xor bh, bh

; BX = R2

test word ptr [inode\_flgs], 4096 ; 1000h

; is this a large or small file

jnz short mget\_4 ; 4f ; large file

test bl, 0F0h ; !0Fh ; error if BX (R2) >= 16

jnz short mget\_2 ; 28/02/2013

and bl, 0Eh ; clear all bits but bits 1,2,3

mov ax, word ptr inode\_dskp[BX] ; AX = R1, physical block number

mget\_3:

; 24/02/2013

cmp ax, 1

mget\_1: ; 2

retn

mget\_4: ; 4 ; large file

; 05/03/2013

and bl, 0FEh

push bx

; 01/03/2013 Max. possible BX (offset) value is 127 (65535/512)

; for this file system (offset 128 to 255 not in use)

; There is always 1 indirect block for this file system

mov ax, word ptr [inode\_dskp] ; inode\_dskp[0]

or ax, ax

jnz short mget\_5

; 28/02/2013

mget\_2:

stc

mget\_7:

pop bx

retn

mget\_5: ;2

; ax = R1, block number

call dsk\_rd ; read indirect block

jc short mget\_7

mget\_6:

pop ax ; R2, get offset

add bx, ax ; first word of indirect block

mov ax, word ptr [BX] ; put physical block no of block

; in file sought in R1 (AX)

jmp short mget\_3 ; 24/02/2013

m\_get endp

convert\_from\_epoch proc near

; 30/11/2012

; Derived from DALLAS Semiconductor

; Application Note 31 (DS1602/DS1603)

; 6 May 1998

;

; INPUT:

; DX:AX = Unix (Epoch) Time

mov cx, 60

call proc\_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 proc\_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 proc\_div32

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

; 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 proc\_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 proc\_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]

;

retn

convert\_from\_epoch endp

proc\_mul32 proc near

; 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

proc\_mul32 endp

proc\_div32 proc near

; 1999

; (Rx\_Dos\_Div32) 32 bit divide procedure

; by Erdogan Tan

; 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

proc\_div32 endp

find\_bfn proc near

; 26/11/2012

; 25/11/2012

;

; find boot file name by i-number (ax)

;

; cf -> 1 means error, ax = 0 -> not found

mov word ptr [uf\_i\_number], ax

push si

mov ax, ROOT\_DIR\_INODE\_NUMBER ; 41

call i\_get

jc short loc\_find\_bfn\_retn

;test word ptr [inode\_flgs], 4000h ; directory i-node ?

;jnz short @f

;mov ah, 0FFh ; error number

;stc

;jmp short loc\_find\_bfn\_retn

;;@@:

xor ax, ax

mov word ptr [u\_off], ax ; u\_off is file offset used by user

loc\_find\_bfn\_1:

mov word ptr [u\_base], offset u\_dirbuf

; u.dirbuff holds a file name copied from

; a directory

mov word ptr [u\_count], 10

mov ax, ROOT\_DIR\_INODE\_NUMBER

call read\_i ; read 10 bytes of file with i-number

; i.e. read a directory entry

jc short loc\_find\_bfn\_retn

mov ax, word ptr [u\_nread]

or ax, ax

jz short loc\_find\_bfn\_2 ; gives error return

mov ax, word ptr [u\_dirbuf]

cmp ax, word ptr [uf\_i\_number] ; Check i-number of directory entry

jne short loc\_find\_bfn\_1 ; if same with specified uf\_i\_number

; it is the boot file

loc\_find\_bfn\_3:

call i\_get

loc\_find\_bfn\_retn:

pop si

retn

loc\_find\_bfn\_2:

stc

jmp short loc\_find\_bfn\_retn

find\_bfn endp

proc\_display\_startupfile\_info proc near

; 06/03/2013

; 30/11/2012

; 29/11/2012 ; @@

; 25/11/2012

mov si, offset Msg\_StartupFile\_Name

call UNIX\_PRINTMSG

mov si, offset Boot\_File\_Name

call UNIX\_PRINTMSG

mov si, offset Str\_Inode\_Number

call UNIX\_PRINTMSG

mov si, word ptr [BSBuffer] ; 06/03/2013

add si, bs\_bf\_inode\_number

mov ax, word ptr [SI]

mov si, offset Decimal\_i\_no\_str

mov cx, 5

call proc\_bin\_to\_decimal

mov si, offset Decimal\_i\_no\_str

mov cx, 4

@@:

cmp byte ptr [SI], '0'

ja short @f

inc si

loop @b

@@:

call UNIX\_PRINTMSG

mov si, offset Str\_startup\_file\_size

call UNIX\_PRINTMSG

mov ax, word ptr [Inode\_size]

mov si, offset Decimal\_size\_str

;mov cx, 5

mov cl, 5

call proc\_bin\_to\_decimal

mov si, offset Decimal\_size\_str

mov cl, 4

@@:

cmp byte ptr [SI], '0'

ja short @f

inc si

loop @b

@@:

call UNIX\_PRINTMSG

mov si, offset Str\_Bytes

call UNIX\_PRINTMSG

; 30/11/2012

mov ax, word ptr [Inode\_ctim]

mov dx, word ptr [Inode\_ctim]+2

call convert\_from\_epoch

mov ax, word ptr [year]

mov si, offset str\_cyear

;mov cx, 4

mov cl, 4

call proc\_bin\_to\_decimal

mov ax, word ptr [month]

mov si, offset str\_cmonth

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [day]

mov si, offset str\_cday

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [hour]

mov si, offset str\_chour

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [minute]

mov si, offset str\_cminute

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [second]

mov si, offset str\_csecond

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [Inode\_mtim]

mov dx, word ptr [Inode\_mtim]+2

call convert\_from\_epoch

mov ax, word ptr [year]

mov si, offset str\_myear

;mov cx, 4

mov cl, 4

call proc\_bin\_to\_decimal

mov ax, word ptr [month]

mov si, offset str\_mmonth

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [day]

mov si, offset str\_mday

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [hour]

mov si, offset str\_mhour

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [minute]

mov si, offset str\_mminute

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [second]

mov si, offset str\_msecond

mov cl, 2

call proc\_bin\_to\_decimal

mov si, offset Str\_SF\_date\_Time

call UNIX\_PRINTMSG

retn

proc\_display\_startupfile\_info endp

proc\_bin\_to\_decimal proc near

; 30/11/2012 (CX input)

; 25/11/2012 unixboot.asm version

; 6-5-2009

; Erdogan Tan

; INPUT: DS:SI = Target location

; AX = Binary Number

; CX = Number of digits

; OUTPUT: Decimal chars at DS:SI

; CX, AX, DX will be changed.

;push bp

;push si

loc\_reset\_str\_NumberInput:

mov byte ptr [SI], "0"

inc si

loop loc\_reset\_str\_NumberInput

mov bp, sp

xor dx, dx

mov cx, 10

loc\_rediv\_NumberInput:

div cx

add dl,'0'

push dx

xor dx, dx

dec si

or ax, ax

jnz short loc\_rediv\_NumberInput

loop\_popcx\_NumberInput:

pop dx

mov byte ptr [SI], dl

inc si

cmp bp, sp

jne short loop\_popcx\_NumberInput

;pop si

;pop bp

retn

proc\_bin\_to\_decimal endp

print\_decimal\_number proc near

; 03/02/2013

; 21/01/2013

; print decimal number

;

; INPUT -> AX = Integer

; 32/02/2013 CX = Number of decimal digits

; OUTPUT -> decimal number as string

pdn0:

mov si, offset dec\_num

;

mov bx, si

add si, cx ; 03/02/2013

mov di, si

;mov cx, 10

mov cl, 10

mov dl, '0'

@@:

mov byte ptr [BX], dl

inc bx

loop @b

;

;xor dl, dl

;mov byte ptr [BX], dl

mov bx, 10

xor dx, dx

pdn\_itoa:

div bx

; 03/02/2013

add byte ptr [SI], dl ; 03/02/2013

and dl, dl

jnz short @f

and al, al

jz short pdn\_14

@@:

dec si

xor dl, dl

jmp short pdn\_itoa

pdn\_14:

mov si, offset dec\_num

mov bx, si

@@: ; leading zeros will not be printed

mov al, byte ptr [BX] ; 03/02/2013

cmp al, '0'

ja short @f

cmp bx, di

jnb short @f

mov al, 20h

mov byte ptr [BX], al

inc bx

jmp short @b

@@:

mov ah, 0Eh

mov bx, 07h

@@:

lodsb

pdn\_putc:

int 10h

cmp si, di

jna short @b

;mov al, 20h

;int 10h

retn

print\_decimal\_number endp

print\_volume\_info proc near

; 06/03/2013

; 05/03/2013

; 16/02/2013

mov bx, word ptr [BSBuffer] ; 06/03/2013

add bx, bsVolumeSerial+2

mov cx, 2

mov di, offset msgVolume\_Serial

@@:

mov ax, word ptr [BX]

call proc\_hex\_double

stosw

mov ax, dx

stosw

dec cx

jz short @f

inc di

sub bx, 2

jmp short @b

@@:

mov si, offset msgVolume\_Info

call UNIX\_PRINTMSG

@@:

mov bx, word ptr [SUPERBLOCK] ; SuperBlock

; start of free storage map for disk

@@:

mov ax, word ptr [BX] ; first word contains # of bytes

; in free storage map

shl ax, 1 ; multiply AX by 8 gives # of blocks

shl ax, 1

shl ax, 1

push ax

mov si, offset msgVol\_Size\_Hdr

call UNIX\_PRINTMSG

pop ax

push ax

mov cl, 4 ; mov cx, 4

call print\_decimal\_number

mov si, offset msgVolume\_Size

call UNIX\_PRINTMSG

pop cx ; cx = bit count of free storage map

xor dx, dx ; mov dx, 0

xor bl, bl ; xor bx, bx

mov si, word ptr [SUPERBLOCK]

add si, 2

mov di, 16

pvi\_size\_loop1:

lodsw

or ax, ax

jz short pvi\_size\_loop3

push cx

mov cx, di

pvi\_size\_loop2:

shr ax, 1

jnc short @f

inc bx

@@:

loop pvi\_size\_loop2

pop cx

pvi\_size\_loop3:

add dx, di

cmp dx, cx

jb short pvi\_size\_loop1

push bx

mov si, offset msgVol\_freeblocks\_Hdr

call UNIX\_PRINTMSG

pop ax ; # of free blocks

mov cx, 4

call print\_decimal\_number

mov si, offset msgVolume\_freeblocks

call UNIX\_PRINTMSG

@@:

mov bx, word ptr [SUPERBLOCK]

mov ax, word ptr [BX]

add ax, 2

add bx, ax ; 06/03/2013

; start of inode map for disk

@@:

mov ax, word ptr [BX] ; first word contains # of bytes

; in inode map

shl ax, 1 ; multiply AX by 8 gives # of inodes

shl ax, 1

shl ax, 1

push bx

push ax

mov si, offset msgVol\_icount\_Hdr

call UNIX\_PRINTMSG

pop ax

push ax

mov cl, 4 ; mov cx, 4

call print\_decimal\_number

mov si, offset msgVolume\_icount

call UNIX\_PRINTMSG

pop cx ; cx = bit count of inode map

pop si ; inode map offset

xor dx, dx ; mov dx, 0

xor bl, bl ; xor bx, bx

mov di, 16

pvi\_icount\_loop1:

lodsw

;cmp ax, 0FFFFh

;je short pvi\_icount\_loop3

inc ax

jz short pvi\_icount\_loop3

dec ax

push cx

mov cx, di

pvi\_icount\_loop2:

shr ax, 1

jc short @f

inc bx

@@:

loop pvi\_icount\_loop2

pop cx

pvi\_icount\_loop3:

add dx, di

cmp dx, cx

jb short pvi\_icount\_loop1

push bx

mov si, offset msgVol\_free\_icount\_Hdr

call UNIX\_PRINTMSG

pop ax ; # of free inodes

mov cx, 4

call print\_decimal\_number

mov si, offset msgVolume\_free\_icount

call UNIX\_PRINTMSG

retn

print\_volume\_info endp

proc\_hex\_double proc near

; 16/02/2013 (AX:DX)

; 28/01/2002 (DX:AX)

; From binary (word) to hexadecimal (character) converter

;

; input -> AX = word (binary number) to be converted

; output -> AX = First 2 characters of hexadecimal number

; output -> DX = Last 2 characters of hexadecimal number

push cx

xor dx, dx

mov cx, 10h

div cx ; Q in AX, R in DX (DL)

push dx ; DH= 0, R in DL <- CX= 10h

xor dl, dl

div cx ; DH= 0, R in DL, AX <= FFh

div cl ; AL <= 0Fh

; R in AH, Q in AL

pop cx ; R in CL

mov dh, cl

or dx,'00'

cmp dl,'9'

jna short pass\_cc\_dl

add dl,7

pass\_cc\_dl:

cmp dh,'9'

jna short pass\_cc\_dh

add dh,7

pass\_cc\_dh:

or ax, '00'

cmp al,'9'

jna short pass\_cc\_al

add al,7

pass\_cc\_al:

cmp ah,'9'

jna short pass\_cc\_ah

add ah,7

pass\_cc\_ah:

pop cx

retn

proc\_hex\_double endp

show\_inode proc near

; 05/03/2013

; 17/02/2013

; print inode details

; Format: inode <decimal number>, iget <decimal number>

; INPUT -> AX <> 0 -> Current Inode [ii]

; AX = 0 -> use inode number input

;

and ax, ax

jnz short show\_inode\_7

mov word ptr [arg], ax ; 0

xor dx, dx

show\_inode\_1:

lodsb

cmp al, '0'

jb short show\_inode\_4

cmp al, '9'

ja short show\_inode\_stc\_retn ; cmc

sub al, '0'

show\_inode\_2:

or dx, dx

jnz short show\_inode\_5

show\_inode\_3:

mov dx, ax

jmp short show\_inode\_1

show\_inode\_4:

or dx, dx

jz short show\_inode\_stc\_retn

cmp al, 20h

jna short show\_inode\_6

show\_inode\_stc\_retn:

cmc

show\_inode\_retn:

retn

show\_inode\_5:

cmp dx, 256

jnb short show\_inode\_stc\_retn

mov ah, dl

mov dl, al

mov al, 10

mul ah

add dx, ax

jmp short show\_inode\_1

show\_inode\_6:

mov bx, word ptr [SUPERBLOCK]

mov ax, word ptr [BX]

add ax, 2

mov bx, ax

mov ax, word ptr [BX] ; inode map bytes

shl ax, 1

shl ax, 1

shl ax, 1 ; inode count

add ax, 40 ; + device file inodes

cmp ax, dx

jb short show\_inode\_retn ; not a valid i-number

mov ax, dx

mov word ptr [arg], ax

; ax = i-number

call i\_get

jc short show\_inode\_retn

show\_inode\_7:

;mov ax, word ptr [ii]

call proc\_hex\_double

mov word ptr [txt\_inode\_number], ax

mov word ptr [txt\_inode\_number]+2, dx

mov ax, word ptr [inode\_flgs]

push ax

call proc\_hex\_double

mov word ptr [txt\_inode\_flags\_h], ax

mov word ptr [txt\_inode\_flags\_h]+2, dx

pop dx

mov di, offset txt\_inode\_flags\_b

mov cx, 16

@@:

xor al, al ; 0

shl dx, 1

adc al, '0'

stosb

loop @b

mov ax, word ptr [inode\_nlks] ; & uid

call proc\_hex\_double

mov word ptr [txt\_inode\_nlks], dx

mov word ptr [txt\_inode\_uid], ax

mov ax, word ptr [inode\_size]

call proc\_hex\_double

mov word ptr [txt\_inode\_size], ax

mov word ptr [txt\_inode\_size]+2, dx

mov cl, 8

mov si, offset inode\_dskp

mov di, offset txt\_inode\_dskp

@@:

lodsw

call proc\_hex\_double

stosw

mov ax, dx

stosw

dec cl

jz short @f

inc di

inc di

jmp short @b

@@:

;mov si, offset inode\_ctim

mov ax, word ptr [SI]

mov dx, word ptr [SI]+2

push dx

push ax

push dx

call proc\_hex\_double

mov word ptr [txt\_inode\_ctim\_h]+4, ax

mov word ptr [txt\_inode\_ctim\_h]+6, dx

pop ax

call proc\_hex\_double

mov word ptr [txt\_inode\_ctim\_h], ax

mov word ptr [txt\_inode\_ctim\_h]+2, dx

pop ax

pop dx

call convert\_from\_epoch

mov ax, word ptr [year]

mov si, offset txt\_inode\_cyear

;mov cx, 4

mov cl, 4

call proc\_bin\_to\_decimal

mov ax, word ptr [month]

mov si, offset txt\_inode\_cmonth

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [day]

mov si, offset txt\_inode\_cday

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [hour]

mov si, offset txt\_inode\_chour

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [minute]

mov si, offset txt\_inode\_cminute

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [second]

mov si, offset txt\_inode\_csecond

mov cl, 2

call proc\_bin\_to\_decimal

mov si, offset inode\_mtim

mov ax, word ptr [SI]

mov dx, word ptr [SI]+2

push dx

push ax

push dx

call proc\_hex\_double

mov word ptr [txt\_inode\_mtim\_h]+4, ax

mov word ptr [txt\_inode\_mtim\_h]+6, dx

pop ax

call proc\_hex\_double

mov word ptr [txt\_inode\_mtim\_h], ax

mov word ptr [txt\_inode\_mtim\_h]+2, dx

pop ax

pop dx

call convert\_from\_epoch

mov ax, word ptr [year]

mov si, offset txt\_inode\_myear

;mov cx, 4

mov cl, 4

call proc\_bin\_to\_decimal

mov ax, word ptr [month]

mov si, offset txt\_inode\_mmonth

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [day]

mov si, offset txt\_inode\_mday

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [hour]

mov si, offset txt\_inode\_mhour

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [minute]

mov si, offset txt\_inode\_mminute

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [second]

mov si, offset txt\_inode\_msecond

mov cl, 2

call proc\_bin\_to\_decimal

mov ax, word ptr [inode\_reserved]

call proc\_hex\_double

mov word ptr [txt\_inode\_reserved], ax

mov word ptr [txt\_inode\_reserved]+2, dx

@@:

mov si, offset msg\_inode\_details

call UNIX\_PRINTMSG

retn

show\_inode endp

load\_kernel proc near

; 06/03/2013

;

; loads unix kernel file

;

; INPUT -> u\_namep = unix kernel/file name address

; unix kernel will be loaded at 'kernel\_loading\_segment'

;

load\_k\_1:

call name\_i

jc short @f

load\_k\_2:

call i\_get

jc short @f

mov bx, inode\_flgs

test word ptr [BX], 10h ; executable file attribute bit

jz short load\_k\_stc

mov bx, inode\_size

xor ax, ax

cmp word ptr [BX], ax ; 0

jna short load\_k\_stc

mov word ptr [u\_off], ax ; 0

mov word ptr [u\_base], ax ; 0

;mov bx, inode\_size

mov ax, word ptr [BX]

mov word ptr [u\_count], ax

mov ax, kernel\_loading\_segment

mov word ptr [EXTRA\_SEGMENT], ax

mov ax, word ptr [ii]

call read\_i

jc short load\_k\_retn

mov cx, word ptr [u\_nread]

mov bx, inode\_size

cmp cx, word ptr [BX]

load\_k\_retn:

mov ax, ds

mov word ptr [EXTRA\_SEGMENT], ax

@@:

retn

load\_k\_stc:

stc

retn

load\_kernel endp

align 2 ; 05/03/2013

PhysicalDriveNumber: db 0

db 0

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

; messages

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

UNIX\_Welcome:

db 'Retro UNIX 8086 v1', 0

Boot\_Msg:

db 0Dh, 0Ah

db 'Boot: '

db 0

align 2 ; 05/03/2013

unix\_cdrv:

db 0Dh, 0Ah

UNIX\_FD\_Name:

db 'fd'

UNIX\_FD\_Number:

db '0:'

unix\_cdir: db '/'

db 37 dup(0)

CDirOffset: dw 0

CursorColumn: dw 0

pdir: dw 0

arg: dw 0

msg\_unix\_drv\_read\_error:

db 0Dh, 0Ah

db "Drive not ready or read error!"

db 0Dh, 0Ah, 0

Msg\_StartupFile\_Name:

db 0Dh, 0Ah

db "Startup File Name : ", 0

error\_msg:

db 0Dh, 0Ah

db 'Error !'

UNIX\_CRLF:

db 0Dh, 0Ah, 0

RetryCount: dw 0

DirFileName: db 20h ; 06/01/2013

BOOT\_FILE\_NAME: db 9 dup(0)

uf\_i\_number: dw 0 ; 25/11/2012

bootfile\_inode:

inode:

inode\_flgs: dw 801Eh ; Flags (1000000000011110b)

inode\_nlks: db 1 ; number of links

inode\_uid: db 0 ; user ID (0 = root)

inode\_size: dw 0 ; file size

inode\_dskp: dw 8 dup (0) ; indirect or contents blocks

inode\_ctim: dd 0 ; creation date & time

inode\_mtim: dd 0 ; modification date & time

inode\_reserved: dw 0 ; unused

align 2 ; 05/03/2013

db 0FFh

U:

u\_uid: db 0

u\_cdir: dw ROOT\_DIR\_INODE\_NUMBER

u\_namep: dw 0

u\_dirp: dw 0

u\_base: dw 0

u\_off: dw 0

u\_count: dw 0

u\_nread: dw 0

u\_dirbuf: db 10 dup(0)

ii: dw 0

buff\_s: dw 0

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

; 25/11/2012

str\_inode\_number:

db 0Dh, 0Ah

db 'Startup File I-Number: ', 0

Decimal\_i\_no\_str:

db 6 dup (0)

Str\_startup\_file\_size:

db 0Dh, 0Ah

db 'Startup File Size : ', 0

Str\_Bytes:

db ' bytes', 0

Decimal\_size\_str: db 6 dup (0)

Str\_sf\_date\_time:

db 0Dh, 0Ah

db 'Creating Date & Time : '

Str\_cday: db '00'

db '/'

Str\_cmonth: db '00'

db '/'

Str\_cyear: db '0000'

db 20h, 20h

Str\_chour: db '00'

db ':'

Str\_cminute: db '00'

db ':'

Str\_csecond: db '00'

db 0Dh, 0Ah

db 'Last Modif. Date & Time : '

Str\_mday: db '00'

db '/'

Str\_mmonth: db '00'

db '/'

Str\_myear: db '0000'

db 20h, 20h

Str\_mhour: db '00'

db ':'

Str\_mminute: db '00'

db ':'

Str\_msecond: db '00'

db 0Dh, 0Ah, 0

;23/02/2013

list\_count: db 0FFh

; 20/01/2013

ls\_option: db 0

; 21/01/2013

dec\_num: db 10 dup(20h) ; 02/03/2012, 3 bytes -> 10 bytes

db 0

;30/12/2012

DotDot:

db '.'

Dot:

db '.'

db 0

;16/02/2013

msgVolume\_Info:

db 0Dh, 0Ah

db "Retro UNIX 8086 v1 (RUFS) File System", 0Dh, 0Ah

db "by Erdogan Tan (2013)"

db 0Dh, 0Ah, 0Dh, 0Ah

db "Volume Serial No: "

msgVolume\_Serial:

db "0000-0000h"

db 0Dh, 0Ah, 0

msgVol\_Size\_Hdr:db "Volume Size : ", 0

msgVolume\_Size: ; db "0000"

db " blocks", 0Dh, 0Ah, 0

msgVol\_freeblocks\_Hdr:db "Free Count : ", 0

msgVolume\_freeblocks : ;db "0000"

db " blocks", 0Dh, 0Ah, 0

msgVol\_icount\_Hdr:

db "# of Inodes : ", 0

msgVolume\_icount: ; db "0000"

db "+40", 0Dh, 0Ah, 0

msgVol\_free\_icount\_Hdr:db 'Free Inodes : ', 0

msgVolume\_free\_icount : ;db "0000"

db 0Dh, 0Ah, 0

NotFound\_msg:

db 0Dh, 0Ah

db "Not found !"

db 0Dh, 0Ah, 0

msgINumber:

db 0Dh, 0Ah

db "Inode Number :", 0

msg\_inode\_details:

db 0Dh, 0Ah

db "UNIX V1 I-NODE STRUCTURE DETAILS OF I-NODE "

txt\_inode\_number:

db "0000h"

db 0Dh, 0Ah, 0Dh, 0Ah

db "Flags : "

txt\_inode\_flags\_h:

db "0000h"

db 20h, 20h

db "["

txt\_inode\_flags\_b:

db "0000000000000000b"

db "]"

db 0Dh, 0Ah

db "# of Links : "

txt\_inode\_nlks:

db "00h"

db 0Dh, 0Ah

db "User ID : "

txt\_inode\_uid:

db "00h"

db 0Dh, 0Ah

db "Size : "

txt\_inode\_size:

db "0000h"

db 0Dh, 0Ah

db "Disk Blocks : "

txt\_inode\_dskp:

db "0000h 0000h 0000h 0000h "

db "0000h 0000h 0000h 0000h"

db 0Dh, 0Ah

db "Creation Time : "

txt\_inode\_ctim\_h:

db "00000000h"

db 20h, 20h

db "["

txt\_inode\_cday:

db "00"

db "/"

txt\_inode\_cmonth:

db "00"

db "/"

txt\_inode\_cyear:

db "0000"

db ","

txt\_inode\_chour:

db "00"

db ":"

txt\_inode\_cminute:

db "00"

db ":"

txt\_inode\_csecond:

db "00"

db "]"

db 0Dh, 0Ah

db "Modification Time : "

txt\_inode\_mtim\_h:

db "00000000h"

db 20h, 20h

db "["

txt\_inode\_mday:

db "00"

db "/"

txt\_inode\_mmonth:

db "00"

db "/"

txt\_inode\_myear:

db "0000"

db ","

txt\_inode\_mhour:

db "00"

db ":"

txt\_inode\_mminute:

db "00"

db ":"

txt\_inode\_msecond:

db "00"

db "]"

db 0Dh, 0Ah

db "Unused : "

txt\_inode\_reserved:

db "0000h"

db 0Dh, 0Ah, 0

Boot\_Commands: ; 25/02/2013

db 0Dh, 0Ah

db "BOOT COMMANDS", 0Dh, 0Ah

db "dir <directory name> : print directory entries without details", 0Dh, 0Ah

db "ls <directory name> : print directory entries, ", 27h, "/", 27h," means entry is directory", 0Dh, 0Ah

db "ls -l <directory name> : print directory entries with details", 0Dh, 0Ah

db "cd <directory name> : change directory", 0Dh, 0Ah

db "show <file name> : show file, print/display file contents", 0Dh, 0Ah

db "inode <inode number> : print inode details for (decimal) inode number", 0Dh, 0Ah

db "namei <file name> : print inode number of file (as decimal)", 0Dh, 0Ah

db "fs : print (current) unix fs (super block) info", 0Dh, 0Ah

db "bootfile : print startup/boot file details", 0Dh, 0Ah

db "reboot : reboot (int 19h)", 0Dh, 0Ah

db "? : print boot commands summary (as above)", 0Dh, 0Ah, 0

align 2 ; 05/03/2013

CommandBuffer: db 74 dup(0)

unix\_reboot: db 0

def\_kernel: db 0

BSBUFFER: dw 0

SUPERBLOCK: dw 0

DISKBUFFER: dw 0

FILEBUFFER: dw 0

EXTRA\_SEGMENT: dw 0 ; 06/03/2013

; 07/03/2013

waiting\_count: dw 182\*3 ; 30 seconds

align 16 ; 05/03/2013

EndOfFile:

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

; buffers

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

;BSBUFFER: db 512 dup(0)

;SUPERBLOCK: db 512 dup(0)

;DISKBUFFER: db 512 dup(0)

;FILEBUFFER: db 512 dup(0)

;;;

;;BootStack:

BOOT1 ends

end START\_CODE