; UNIXCOPY.ASM (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

; Bootable Unix (RUFS) File System - DOS & UNIX FS file export/import Utility

; (08/12/2012)

;

; [ Last Modification: 14/07/2015 ]

;

; 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>

;

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

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

; DTA (PSP+80h= Offset 128)

DTA\_Attrib equ 149 ; PDP+21 ;05/01/2013

DTA\_Time equ 150 ; PSP+22

DTA\_Date equ 152 ; PSP 24

DTA\_FileSize equ 154 ; PSP + 26

DTA\_FileName equ 158 ; PSP + 30

;err\_INVALIDDATA equ 100h

;err\_NOFREEBLOCK equ 200h

i\_flags equ 001Eh

.8086

UNIXCOPY SEGMENT PUBLIC 'CODE'

assume cs:UNIXCOPY,ds:UNIXCOPY,es:UNIXCOPY,ss:UNIXCOPY

org 100h

START\_CODE:

proc\_start proc near

; 08/12/2012 (UNIXCOPY)

;

; 30/11/2012 (UNIXBOOT)

;

mov bx, SizeOfFile+100

add bx, 15

shr bx, 1

shr bx, 1

shr bx, 1

shr bx, 1

mov ah, 4Ah ; modify memory allocation

;push cs

;pop es

int 21h

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

; see if drive specified

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

mov si, offset 80h ; PSP command tail

lodsb

mov cl, al

or cl, cl

jnz short loc\_get\_args

dec cl

jmp loc\_unix\_welcome

loc\_get\_args:

lodsb

cmp al, ' '

jne short loc\_check\_fd\_name

;dec cl

;jz short loc\_unix\_welcome

jmp short loc\_get\_args

loc\_check\_fd\_name:

; 07/07/2015

mov di, offset img\_file\_name

cmp al, "f"

jne short loc\_chk\_fname1

stosb

lodsb

cmp al, "d"

jne short loc\_chk\_fname1

stosb

lodsb

cmp al, '0'

jb short loc\_chk\_fname2

cmp al, '1'

ja short loc\_chk\_fname2

stosb

mov dl, al

lodsb

cmp al, 0Dh

ja short loc\_chk\_fname2

mov byte ptr [UNIX\_FD\_Number], dl

sub dl, '0'

mov byte ptr [PhysicalDriveNumber], dl

jmp load\_boot\_sector

loc\_check\_file\_name:

; 07/07/2015

lodsb

loc\_chk\_fname1:

cmp al, 0Dh

jna short loc\_chk\_fname\_ok

loc\_chk\_fname2:

stosb

cmp di, offset img\_file\_name + 12

jb short loc\_check\_file\_name

je short loc\_chk\_fname\_ok

loc\_inv\_fname:

mov si, offset msg\_inv\_file\_name

jmp @f

loc\_chk\_fname\_ok:

sub al, al

stosb

loc\_cap\_file\_name:

; file name capitalization

mov si, offset img\_file\_name

mov di, si

mov bx, si

loc\_cap\_file\_name0:

lodsb

cmp al, 'a'

jnb short loc\_cap\_file\_name2

and al, al

jz short loc\_cap\_file\_name3

cmp al, '.'

jne short loc\_cap\_file\_name1

mov bx, di ; dot position

loc\_cap\_file\_name1:

stosb

jmp short loc\_cap\_file\_name0

loc\_cap\_file\_name2:

cmp al, 'z'

ja short loc\_cap\_file\_name1

and al, 0DFh ; NOT 32

stosb

jmp short loc\_cap\_file\_name0

loc\_cap\_file\_name3:

mov [di], al

dec di

cmp bx, di

jnb short loc\_inv\_fname

sub di, bx

sub bx, offset img\_file\_name

cmp di, 3

jna short loc\_cap\_file\_name4

and bx, bx

jnz short loc\_inv\_fname

jmp short loc\_find\_image\_file

loc\_cap\_file\_name4:

cmp bx, 8

ja short loc\_inv\_fname

loc\_find\_image\_file:

; 07/07/2015

mov dx, offset img\_file\_name

mov cx, 3Fh ; File Attributes

mov ah, 4Eh ; MS Dos Function = Find First File

int 21h

jnc short loc\_chk\_image\_file\_features

cmp ah,03h ; dos error number > 3

ja loc\_error

mov si, offset msg\_file\_not\_found

jmp @f

loc\_chk\_image\_file\_features:

mov si, DTA\_Attrib

mov al, byte ptr [SI]

and al, 1Fh ; directory, volume label, system, hidden, read only

jnz loc\_error

mov si, DTA\_FileSize

lodsw

cmp word ptr [SI], 16h

jne short loc\_inv\_image\_file

cmp ax, 8000h ;1.44 MB floppy disk image (168000h bytes)

je short loc\_open\_image\_file

loc\_inv\_image\_file:

mov si, offset msg\_inv\_image\_file

jmp short @f

loc\_open\_image\_file:

mov al, 2 ; open for reading and writing

;mov dx, offset img\_file\_name

mov ah, 3Dh ; open file

int 21h

jc short loc\_error

mov word ptr [img\_file\_handle], ax

mov byte ptr [PhysicalDriveNumber], 90h ; image file sign

;

mov bx, ax

mov cx, 1024 ; read 1024 bytes (2 sectors)

mov dx, offset BSBuffer ; bootsector (& super block) buffer

mov ah, 3Fh ; read file

int 21h

jc short loc\_error

cmp ax, 1024

jne short loc\_error

mov bx, dx ; offset BSBuffer

jmp short load\_fd\_img\_boot\_sect\_ok

load\_boot\_sector:

; input -> dl = drive number

xor ah,ah

int 13h

jc short loc\_drv\_read\_error

load\_boot\_sector\_ok:

mov bx, offset BSBuffer

mov ax,0202h ; Read boot sector & super block

mov cx,1

xor dh,dh

int 13h

jc short loc\_drv\_read\_error

load\_fd\_img\_boot\_sect\_ok:

cmp word ptr [BX]+510, 0AA55h

jne short loc\_not\_fd\_rufs

cmp word ptr [BX]+bsFSystemID, 'UR'

jne short loc\_not\_fd\_rufs

cmp word ptr [BX]+bsFSystemID+2, 'SF'

je short loc\_check\_fd\_sign

loc\_not\_fd\_rufs:

mov si, offset msg\_Not\_Unix\_FS

jmp short @f

loc\_check\_fd\_sign:

cmp word ptr [BX]+bsFDSign, 'df'

jne short loc\_not\_fd\_rufs

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

; Write message

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

loc\_unix\_welcome:

pushf

mov si, offset UNIX\_Welcome

call UNIX\_PRINTMSG

popf

je short loc\_call\_unix\_prompt

mov si, offset usage

@@:

call UNIX\_PRINTMSG

loc\_close\_file\_then\_terminate:

; 07/07/2015

mov bx, [img\_file\_handle]

and bx, bx

jz short terminate

close\_img\_file:

mov ah, 3Eh ; close (floppy disk image) file

int 21h

terminate:

int 20h

loc\_drv\_read\_error:

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

jmp short @b

loc\_error:

mov si, offset error\_msg

jmp short @b

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

; call command interpreter

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

loc\_call\_unix\_prompt:

call unix\_prompt

; 07/07/2015

jmp short loc\_close\_file\_then\_terminate

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

; 07/07/2015

; 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\_0:

; 07/07/2015

cmp byte ptr [PhysicalDriveNumber], 90h

jb short unix\_prompt\_1

mov si, offset img\_file\_name

call unix\_printmsg

mov si, offset unix\_img\_cdir

jmp short unix\_prompt\_15

unix\_prompt\_1:

mov si, offset unix\_cdrv

call unix\_printmsg

unix\_prompt\_2:

mov si, offset unix\_cdir

unix\_prompt\_15:

call unix\_printmsg

unix\_prompt\_3:

mov al, byte ptr [unix\_prompt\_char]

;mov ah,0Eh

;mov bx,07h

int 10h

unix\_prompt\_4:

mov ah,03h

;mov bx,07h

int 10h

mov byte ptr [CursorColumn],dl

unix\_prompt\_5:

mov si, offset CommandBuffer

call proc\_rw\_char

;mov byte ptr [CommandBuffer]+75,0

;mov si, offset CommandBuffer

mov di, si

xor bx, bx

xor cx, cx

unix\_prompt\_6:

mov al, byte ptr [SI][BX]

inc bl

cmp al, 20h

ja short unix\_prompt\_8

jb short unix\_prompt\_13

cmp bl, 74 ; 75 ?

jb short unix\_prompt\_6

jmp short unix\_prompt\_13

unix\_prompt\_7:

mov al, byte ptr [SI][BX]

inc bl

cmp al,20h

jna short unix\_prompt\_9

unix\_prompt\_8:

stosb

inc cl

cmp bl, 74 ; 75 ?

jb short unix\_prompt\_7

;jmp short unix\_prompt\_12

unix\_prompt\_9:

xor al, al ; 0

unix\_prompt\_10:

mov byte ptr [DI], al

inc di

cmp bl, 74 ; 75 ?

jnb short unix\_prompt\_12

mov al, byte ptr [SI][BX]

inc bl

cmp al, 20h

jnb short unix\_prompt\_10

unix\_prompt\_11:

mov byte ptr [DI], 0

unix\_prompt\_12:

call command\_interpreter

cmp byte ptr [program\_exit], 1

jnb short unix\_prompt\_14

mov cx, 74 ; 75 ?

mov di, offset CommandBuffer

xor al,al

rep stosb

unix\_prompt\_13:

mov bx,07h

mov al,0Dh

mov ah,0Eh

int 10h

mov al,0Ah

int 10h

jmp unix\_prompt\_0 ; loop

unix\_prompt\_14:

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

; 01/03/2013

; 25/02/2013

; 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 ; 25/02/2013

cl4:

cmp cl, 4

ja cl5

jb cl3

; EXIT

loc\_cmd\_exit:

cmp ax, 'xe'

jne short loc\_cmd\_show

lodsw

cmp ax, 'ti'

jne short @f

lodsb

or al, al

jnz short @f

mov byte ptr [program\_exit], 1

@@:

retn

; SHOW

loc\_cmd\_show:

; 06/01/2013

cmp ax, 'hs'

jne short loc\_cmd\_link

lodsw

cmp ax, 'wo'

jne short @b

lodsb

or al, al

jnz short @b

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

; LINK

loc\_cmd\_link:

cmp ax, 'il'

jne loc\_cmd\_iget ; 17/02/2013

lodsw

cmp ax, 'kn'

jne short @b

lodsb

or al, al

jnz short @b

link\_sf1:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short link\_sf1

jb short @b

link\_sf2:

lodsb

cmp al, 20h

ja short link\_sf2

xor al, al

mov byte ptr [SI]-1, al

link\_df1:

mov word ptr [arg], si

lodsb

cmp al, 20h

je short link\_df1

jb short @b

link\_df2:

lodsb

cmp al, 20h

ja short link\_df2

dec si

xor al, al

mov byte ptr [SI], al

link\_fsf:

call name\_i

jc ci\_error

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

link\_fdf:

mov si, word ptr [arg]

mov word ptr [u\_namep], si

call name\_i

jnc ci\_error

cmp ah, 0FFh

jne ci\_error

; [u\_dirp] = empty directory entry slot

mov ax, word ptr [ii]

mov word ptr [pdir], ax

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

call i\_get

jc ci\_error

call set\_imod ; jsr r0,setimod / set modified flag

inc byte ptr [inode\_nlks] ; link count

mov ax, word ptr [pdir]

call i\_get

jc ci\_error

; name\_i -> u\_namep points filename

; after the last '/' of the path

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

mov word ptr [u\_dirbuf], ax

call mk\_dir ; make directory entry

jc ci\_error

jmp ci\_sync\_exit

; IGET

loc\_cmd\_iget: ; 17/02/2013, inode/iget

cmp ax, 'gi'

jne short loc\_cmd\_help ; 23/02/2013

lodsw

cmp ax, 'te'

jne short @f

lodsb

or al, al

jnz short @f

ci\_iget\_getarg:

mov bx, si

lodsb

cmp al, 20h

ja inode\_getarg2

je short ci\_iget\_getarg

; HELP

loc\_cmd\_help: ; 23/02/2013

cmp ax, 'eh'

jne short @f

lodsw

cmp ax, 'pl'

jne short @f

lodsb

and al, al

jnz short @f

ci\_?:

mov si, offset UNIXCOPY\_Commands

call UNIX\_PRINTMSG

@@:

retn

cl5:

cmp cl, 5

ja cl7

;jb short @f

; CHDIR

loc\_cmd\_chdir:

cmp ax,'hc'

jne loc\_cmd\_todos

lodsw

cmp ax, 'id'

jne short loc\_cmd\_chmod

lodsb

cmp al, 'r'

jne short @f

lodsb

or al, al

jnz short @f

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

; CHMOD

loc\_cmd\_chmod: ; 13/01/2013

;cmp ax, 'hc'

;jne short loc\_cmd\_todos

;lodsw

cmp ax, 'om'

jne short loc\_cmd\_chown

lodsb

cmp al, 'd'

jne short @b

lodsb

or al, al

jnz short @b

ci\_chmod\_getarg:

lodsb

cmp al, 20h

je short ci\_chmod\_getarg

jb short @b

dec si

call chmode

jc ci\_error

mov ax, word ptr [arg]

or ax, ax

jz short @b

xor al, al

mov byte ptr [arg]+2, al

ci\_chown\_print:

mov si, offset msg\_arg

call UNIX\_PRINTMSG

jmp ci\_sync\_exit

; CHOWN

loc\_cmd\_chown: ; 13/01/2013

;cmp ax, 'hc'

;jne short loc\_cmd\_todos

;lodsw

cmp ax, 'wo'

jne short @b

lodsb

cmp al, 'n'

jne short @b

lodsb

or al, al

jnz short @b

ci\_chown\_getarg:

lodsb

cmp al, 20h

je short ci\_chown\_getarg

jb short @b

dec si

call chowner

jc ci\_error

and bx, bx

jnz short ci\_chown\_print

@@:

retn

; TODOS

loc\_cmd\_todos:

; 24/12/2012

cmp ax, 'ot'

jne loc\_cmd\_mkdir ; 30/12/2012

lodsw

cmp ax, 'od'

jne short @b

lodsb

cmp al, 's'

jne short @b

lodsb

or al, al

jnz short @b

todos\_uf1:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short todos\_uf1

jb short @b

todos\_uf2:

lodsb

cmp al, 20h

ja short todos\_uf2

xor al, al

mov byte ptr [SI]-1, al

todos\_df1:

mov word ptr [arg], si

lodsb

cmp al, 20h

je short todos\_df1

jb short @b

todos\_df2:

lodsb

cmp al, 20h

ja short todos\_df2

dec si

xor al, al

mov byte ptr [SI], al

todos\_fuf:

call name\_i

;jnc short @f

jc ci\_error

;cmp ah, 0FFh

;jne ci\_error

; jmp ci\_error ; 'file not found' error

@@:

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

todos\_fdf:

mov dx, word ptr [arg]

mov cx, 3Fh ; File Attributes ; 05/01/2013 (3Fh)

mov ah, 4Eh ; MS Dos Function = Find First File

int 21h

;jnc short todos\_afow

jnc short @f ; 05/01/2013

todos\_chk\_err:

cmp ah,03h ; dos error number > 3

ja ci\_error

jmp short todos\_crdf

@@: ; 05/01/2013

mov si, DTA\_Attrib

mov al, byte ptr [SI]

and al, 1Fh ; directory, volume label, system, hidden, read only

jnz ci\_error

todos\_afow: ; overwrite question

mov si, offset msg\_overwrite\_question1

call UNIX\_PRINTMSG

mov si, DTA\_FileName

call UNIX\_PRINTMSG

mov si, offset msg\_overwrite\_question2

call UNIX\_PRINTMSG

mov si, offset msg\_yes\_no

call UNIX\_PRINTMSG

todos\_afow\_input: ; ask for overwrite

xor ax, ax

int 16h ; wait for keyboard command

cmp al, 'C'-40h

je short @f

cmp al, 27

je short @f

and al, 0DFh

cmp al, 'Y' ; Yes?

je short todos\_afow\_yes ; overwrite

cmp al, 'N' ; No?

jne short todos\_afow\_input

todos\_afow\_no:

mov si, offset msg\_No

call UNIX\_PRINTMSG

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

@@:

retn

todos\_afow\_yes:

mov si, offset msg\_YES

call UNIX\_PRINTMSG

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

todos\_crdf:

;mov dx, word ptr [arg]

xor cx, cx ; File Attributes = 0

mov ah, 3Ch ; MS Dos Function = Create File

int 21h

jc ci\_error

mov word ptr [FileHandle], ax

todos\_odf:

mov dx, word ptr [arg]

mov ah, 3Dh ; MS Dos Function = Open File

xor al, al

int 21h

jc ci\_error

todos\_ruf\_wdf:

xor ax, ax

mov word ptr [u\_off], ax

todos\_wf\_msg:

mov si, offset Msg\_writing\_file

call UNIX\_PRINTMSG

todos\_iget:

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

call i\_get

jc short todos\_cdf

mov ax, word ptr [inode\_size]

mov cx, 512

cmp ax, cx

jna short loc\_read\_unix\_sf

mov ax, cx

@@:

loc\_read\_unix\_sf:

mov word ptr [u\_count], ax

mov word ptr [u\_base], offset ReadBuffer

mov ax, word ptr [uf\_i\_number] ; word ptr [u\_dirbuf]

call read\_i

jc short todos\_cdf

@@:

;loc\_write\_dos\_df:

mov ah, 40h ; Write File

mov cx, word ptr [u\_nread] ; 0 -> eof

mov dx, offset ReadBUFFER

mov bx, word ptr [FileHandle]

int 21h

jc short todos\_cdf

cmp ax, cx ; write count = read count ?

jne short todos\_cdf ; jb short todos\_cdf

or ax, ax ; or cx, cx

jnz short loc\_read\_unix\_sf

todos\_cdf:

pushf

jc short @f

todos\_set\_dfdt:

mov ax, word ptr [inode\_ctim] ; fromdos command ->

mov dx, word ptr [inode\_ctim]+2 ; dos lmdt -> unix ctim

call convert\_from\_epoch

mov dx, word ptr [hour]

mov cl, 11

shl dx, cl

mov ax, word ptr [minute]

mov cl, 5

shl ax, cl

or dx, ax

mov ax, word ptr [second]

shr ax, 1

or ax, dx

push ax ; time

mov dx, word ptr [year]

sub dx, 1980

mov cl, 9

shl dx, cl

mov ax, word ptr [month]

mov cl, 5

shl ax, cl

or dx, ax

mov ax, word ptr [day]

or dx, ax

pop cx ; time

mov ax, 5701h ; set lm date&time

mov bx, word ptr [FileHandle]

int 21h

@@:

mov ah, 3Eh ; Close File

mov bx, word ptr [FileHandle]

int 21h

popf

jc ci\_error

todos\_retn:

mov si, offset Msg\_OK

call UNIX\_PRINTMSG

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

@@:

retn

; MKDIR

loc\_cmd\_mkdir:

; 30/12/2012

cmp ax, 'km'

jne short loc\_cmd\_rmdir

lodsw

cmp ax, 'id'

jne short @b

lodsb

cmp al, 'r'

jne short @b

lodsb

or al, al

jnz short @b

ci\_mkdir\_getarg1:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short ci\_mkdir\_getarg1

jb short @b

ci\_mkdir\_getarg2: ; 06/01/2013

lodsb

cmp al, 20h

ja short ci\_mkdir\_getarg2

dec si

xor al, al

mov byte ptr [SI], al

mov si, offset Msg\_Making\_Directory

call UNIX\_PRINTMSG

call make\_directory

jc ci\_error

jmp ci\_sync\_exit

; RMDIR

loc\_cmd\_rmdir:

; 05/01/2013

cmp ax, 'mr'

jne short loc\_cmd\_namei ; 17/02/2013

lodsw

cmp ax, 'id'

jne short @b

lodsb

cmp al, 'r'

jne short @b

lodsb

or al, al

jnz short @b

ci\_rmdir\_getarg1:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short ci\_rmdir\_getarg1

jb short @b

; 06/01/2013

mov ah, al

ci\_rmdir\_getarg2:

lodsb

cmp al, 20h

ja short ci\_rmdir\_getarg2

dec si

xor al, al

mov byte ptr [SI], al

mov al, '.'

cmp ah, al ; dot

jne short @f

mov ah, byte ptr [SI]

cmp ah, 21h

jb ci\_error

cmp ah, al ;'.' ; dotdot (parent dir)

jne short @f

inc si

cmp byte ptr [SI], 21h

jb ci\_error

@@:

; u\_namep = pointer to directory path name

call name\_i

jc ci\_error

cmp ax, ROOT\_DIR\_INODE\_NUMBER

je ci\_error

cmp ax, word ptr [u\_cdir]

je ci\_error

push ax

mov si, offset Msg\_Removing\_Directory

call UNIX\_PRINTMSG

pop ax

call remove\_directory

jc ci\_error

jmp ci\_sync\_exit

; 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 short @f

lodsb

cmp al, 'i'

jne short @f

lodsb

or al, al

jnz short @f

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

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 @b

lodsw

cmp ax, 'do'

jne short @b

lodsb

cmp al, 'e'

jne short @b

lodsb

or al, al

jnz short @b

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

cl3:

cmp cl, 3

jb short cl2

; DIR

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

cmp ax, 'id'

jne short @f

lodsb

cmp al, 'r'

jne short @f

lodsb

or al, al

jnz short @f

mov byte ptr [ls\_option], al ; 20/01/2013

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 @b

cmp ah, 0

je ci\_?

@@:

retn

; 16/12/2012

cl2:

cmp cl, 2

jb short cl1 ; 23/02/2013

; jb @b

; CD (CHDIR)

loc\_cmd\_cd:

cmp ax, 'dc'

jne short loc\_cmd\_ls

lodsb

or al, al

jnz short @b

jmp ci\_cd\_getarg

; LS (DIR)

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

cmp ax, 'sl'

jne short loc\_cmd\_rm

lodsb

or al, al

jnz short @b

mov byte ptr [ls\_option], 1

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

; RM

loc\_cmd\_rm:

cmp ax, 'mr'

jne loc\_cmd\_fs ; 16/02/2013

lodsb

or al, al

jnz short @b

rm\_getarg:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short rm\_getarg

jb short @b

rm\_namei:

call name\_i

jc short ci\_error

;cmp word ptr [ii], 41 ; i-number of the directory

;jne short @f

mov si, offset BSBuffer + bs\_BF\_inode\_Number

cmp ax, word ptr [SI] ; is it i-number of the boot file

;je short ci\_error

jne short @f

cmp word ptr [ii], 41 ; i-number of root directory

je ci\_error

mov word ptr [SI], 0 ; reset wrong boot file configuration

@@: mov word ptr [uf\_i\_number], ax ; word ptr [u\_dirbuf]

; 05/01/2013

mov dx, word ptr [ii]

mov word ptr [pdir], dx

call i\_get

jc ci\_error

mov ax, word ptr [inode\_flgs]

test ah, 40h ; 'directory' flag

jnz ci\_error

test al, 4h

jz ci\_error ; 'write' flag

;

rm\_move\_fn:

mov si, offset u\_dirbuf + 2

mov di, offset Boot\_File\_Name

mov cx, 8

@@:

lodsb

and al, al

jz short @f

stosb

loop @b

xor al, al ; 06/01/2013

@@:

mov byte ptr [DI], al ; 0

mov si, offset msg\_remove\_question1

call UNIX\_PRINTMSG

mov si, Offset Boot\_File\_Name

call UNIX\_PRINTMSG

mov si, offset msg\_remove\_question2

call UNIX\_PRINTMSG

mov si, offset msg\_yes\_no

call UNIX\_PRINTMSG

rm\_yn\_input: ; ask for remove

xor ax, ax

int 16h ; wait for keyboard command

cmp al, 'C'-40h

je short @f

cmp al, 27

je short @f

and al, 0DFh

cmp al, 'Y' ; Yes?

je short rm\_a\_yes ; overwrite

cmp al, 'N' ; No?

jne short rm\_yn\_input

rm\_a\_no:

mov si, offset msg\_No

call UNIX\_PRINTMSG

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

@@:

retn

rm\_a\_yes:

mov si, offset msg\_YES

call UNIX\_PRINTMSG

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

rm\_unlink:

mov si, offset Msg\_removing\_file

call UNIX\_PRINTMSG

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

call unlink

jc ci\_error

jmp ci\_sync\_exit

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

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

jne short @b

lodsb

or al, al

jnz short @b

vol\_infO\_print:

fs\_info\_print:

call print\_volume\_info

@@:

retn

cl6: ; 16/02/2013

cmp cl, 6

jne short @b

; VOLUME (fs) ; 16/02/2013

loc\_cmd\_volume:

cmp ax, 'ov'

jne short @b

lodsw

cmp ax, 'ul'

jne short @b

lodsw

cmp ax, 'em'

jne short @b

lodsb

or al, al

jz short vol\_infO\_print

@@:

retn

; 15/12/2012

cl7:

cmp cl, 7

jb cl6 ;16/02/2013

ja cl8

; FROMDOS

loc\_cmd\_fromdos:

cmp ax, 'rf'

jne short @b

lodsw

cmp ax, 'mo'

jne short @b

lodsw

cmp ax, 'od'

jne short @b

lodsb

cmp al, 's'

jne short @b

lodsb

or al, al

jnz short @b

fromdos\_df1:

mov word ptr [arg], si

lodsb

cmp al, 20h

je short fromdos\_df1

jb short @b

fromdos\_df2:

lodsb

cmp al, 20h

ja short fromdos\_df2

xor al, al

mov byte ptr [SI]-1, al

fromdos\_uf1:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short fromdos\_uf1

jb short @b

fromdos\_uf2:

lodsb

cmp al, 20h

ja short fromdos\_uf2

dec si

xor al, al

mov byte ptr [SI], al

fromdos\_fdf:

mov dx, word ptr [arg]

mov cx, 27h ; File Attributes

mov ah, 4Eh ; MS Dos Function = Find First File

int 21h

jc ci\_error ; file not found

fromdos\_fuf:

call name\_i

jnc short @f

cmp ah, 0FFh

jne ci\_error

xor ax, ax

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

jmp fromdos\_s\_fs\_mdt

@@:

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

; 05/01/2013

mov dx, word ptr [ii]

mov word ptr [pdir], dx

call i\_get

jc ci\_error

mov ax, word ptr [inode\_flgs]

test ah, 40h ; 'directory' flag

jnz ci\_error

test al, 4h ; 'write' flag

jz ci\_error

;

fromdos\_afow:

mov si, offset u\_dirbuf + 2

mov di, offset Boot\_File\_Name

mov cx, 8

@@:

lodsb

and al, al

jz short @f

stosb

loop @b

xor al, al ; 01/03/2013

@@:

mov byte ptr [DI], al ; 0

mov si, offset msg\_overwrite\_question1

call UNIX\_PRINTMSG

mov si, Offset Boot\_File\_Name

call UNIX\_PRINTMSG

mov si, offset msg\_overwrite\_question2

call UNIX\_PRINTMSG

mov si, offset msg\_yes\_no

call UNIX\_PRINTMSG

fromdos\_afow\_input: ; ask for overwrite

xor ax, ax

int 16h ; wait for keyboard command

cmp al, 'C'-40h

je short @f

cmp al, 27

je short @f

and al, 0DFh

cmp al, 'Y' ; Yes?

je short fromdos\_afow\_yes ; overwrite

cmp al, 'N' ; No?

jne short fromdos\_afow\_input

fromdos\_afow\_no:

mov si, offset msg\_No

call UNIX\_PRINTMSG

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

@@:

retn

fromdos\_afow\_yes:

mov si, offset msg\_YES

call UNIX\_PRINTMSG

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

fromdos\_uf\_itrunc:

; 05/01/2013

mov ax, word ptr [pdir]

call i\_get

jc ci\_error

;

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

call itrunc ; truncate file

jc ci\_error

fromdos\_s\_fs\_mdt:

; 15/12/2012

; Derived from UNIXBOOT.ASM (30/11/2012)

;mov si, DTA\_FileSize

mov si, DTA\_FileSize+2

;mov ax, word ptr [SI]

;mov dx, word ptr [SI]+2

;or ax, dx ; 64KB file size limit

mov ax, word ptr [SI]

and ax, ax

jnz ci\_error

;mov word ptr [file\_Size], ax

mov si, DTA\_Date

mov ax, word ptr [SI]

push ax

and ax, 00011111b ; Day Mask

mov word ptr [day], ax

pop ax

mov cl, 5

shr ax, cl ; shift right 5 times

push ax

and ax, 00001111b ; Month Mask

mov word ptr [month], ax

pop ax

mov cl, 4

shr ax, cl

;and ax, 01111111b ; Result = Year - 1980

add ax, 1980

mov word ptr [year], ax

mov si, DTA\_Time

mov ax, word ptr [SI]

push ax

and ax, 0000011111b ; Second Mask

shl al, 1

mov word ptr [second], ax

pop ax

mov cl, 5

shr ax, cl ; shift right 5 times

push ax

and ax, 0000111111b ; Minute Mask

mov word ptr [minute], ax

pop ax

mov cl, 6 ; shift right 6 times

shr ax, cl ; (6+5=11)

mov word ptr [hour], ax ; ax = hours

call convert\_to\_epoch

mov word ptr [uf\_make\_datetime], ax

mov word ptr [uf\_make\_datetime]+2, dx

fromdos\_odf:

mov dx, word ptr [arg]

mov ah, 3Dh ; MS Dos Function = Open File

xor al, al

int 21h

jc ci\_error

mov word ptr [FileHandle], ax

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

; 23/02/2013

and ax, ax

jz short @f

; jnz short fromdos\_wf\_msg ;@f

xor ax, ax

call fromdos\_maknod

jmp short fromdos\_wf\_msg

@@: ; fromdos\_mknod:

mov ax, i\_flags ; 1Eh

call mak\_nod

jc short fromdos\_cf

fromdos\_wf\_msg:

mov si, offset Msg\_writing\_file

call UNIX\_PRINTMSG

@@: ; 16/12/2012

xor ax, ax

mov word ptr [u\_off], ax

@@:

;loc\_read\_dos\_sf:

mov ah, 3Fh ; Read File

mov cx, 512

mov dx, offset ReadBUFFER

mov bx, word ptr [FileHandle]

int 21h

jc short fromdos\_cf

or ax, ax

jz short fromdos\_cf

mov word ptr [u\_count], ax

mov word ptr [u\_base], offset ReadBuffer

mov ax, word ptr [uf\_i\_number] ; word ptr [u\_dirbuf]

call write\_i

jnc short short @b ; loc\_read\_dos\_sf

fromdos\_cf:

pushf

mov ah, 3Eh ; Close File

mov bx, word ptr [FileHandle]

int 21h

popf

jc ci\_error

@@:

; 23/02/2013

xor ax, ax

mov word ptr [inode\_mtim], ax

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

call set\_imod

ci\_sync\_exit:

call sync

jc ci\_error

fromdos\_retn:

mov si, offset Msg\_OK

call UNIX\_PRINTMSG

mov si, offset UNIX\_CRLF

call UNIX\_PRINTMSG

@@:

retn

cl8:

cmp cl, 8

;jb short @b

ja cl10

; BOOTFILE

loc\_cmd\_bootfile:

cmp ax, 'ob'

jne short @b

lodsw

cmp ax, 'to'

jne short @b

lodsw

cmp ax, 'if'

jne short @b

lodsw

cmp ax, 'el'

jne short @b

lodsb

or al, al

jnz short @b

@@:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short @b

ja short ci\_bf\_namei

mov si, offset BSBuffer + 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

ci\_bf\_namei:

call name\_i

jc ci\_error

cmp word ptr [ii], ROOT\_DIR\_INODE\_NUMBER

jne ci\_error

; 05/01/2013

; ax = i-number of (new) boot file

call i\_get

jc ci\_error

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

jnz ci\_error

@@:

mov si, offset BSBuffer + bs\_BF\_inode\_Number

mov word ptr [SI], ax

call sync

jc ci\_error

mov si, offset msg\_sf\_configuration\_set\_ok

call UNIX\_PRINTMSG

@@:

retn

cl10:

cmp cl, 10

jne short @f

; NOBOOTFILE

loc\_cmd\_nobootfile:

cmp ax, 'on'

jne short @f

lodsw

cmp ax, 'ob'

jne short @f

lodsw

cmp ax, 'to'

jne short @f

lodsw

cmp ax, 'if'

jne short @f

lodsw

cmp ax, 'el'

jne short @f

lodsb

or al, al

jnz short @f

mov si, offset BSBuffer + bs\_BF\_inode\_Number

and ax, ax

jz ci\_no\_bootfile

xor ax, ax

mov word ptr [SI], ax

call sync

jc ci\_error

mov si, msg\_sf\_configuration\_reset\_ok

call UNIX\_PRINTMSG

@@:

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:

;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

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:

; 21/02/2013

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

make\_directory proc near

; 30/12/2012

;

; mov word ptr [u.namep], si

call sys\_mkdir

jc short @f

;ax = i-number

;mov ax, word ptr [ii]

;mov word ptr [u\_dirbuf], ax

mov word ptr [u\_namep], offset dot

xor ax, ax

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

call mk\_dir ; make a directory entry

; in current (ii) directory

jc short @f

mov word ptr [u\_dirp], 10

mov ax, word ptr [pdir]

mov word ptr [u\_dirbuf], ax

mov word ptr [u\_namep], offset dotdot

call mk\_dir

@@:

retn

make\_directory endp

sys\_mkdir proc near

; 05/01/2013 (bugfix)

; 30/12/2012 unixcopy.asm

; Retro UNIX v1 FS file import/export version

; of sysmkdir function

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;

; RETRO UNIX v1 FS

; sysmkdir:

; make a directory

;

;

; return => if cf=1 error code in AH

; If cf=0 -> AX = I-Number (also in u.dirbuff)

;jsr r0,arg2 / point u.namep to the file name

;jsr r0,namei / get the i-number

; br .+4 / if file not found branch around error

;br error2 / directory already exists (error)

;tstb u.uid / is user the super user

;bne error2 / no, not allowed

;mov (sp)+,r1 / put the mode in r1

;bic $!317,r1 / all but su and ex

;bis $40000,r1 / directory flag

;jsr r0,maknod / make the i-node for the directory

;br sysret2 /

mov bx, word ptr [u\_namep]

mov si, bx ; 05/01/2013

makdir\_1:

lodsb

or al, al

jz short makdir\_2

cmp al, '/'

jne short makdir\_1

mov bx, si

jmp short makdir\_1

makdir\_2:

cmp bx, word ptr [u\_namep]

je short makdir\_3

dec si

dec si ; 05/01/2013

cmp byte ptr [SI], '/' ; is the last char '/'

jne short makdir\_3

cmp si, word ptr [u\_namep] ; 05/01/2013

je short makdir\_3

stc

@@:

retn

makdir\_3:

mov word ptr [pdir], bx

sysmkdir\_0:

call name\_i

jc short sysmkdir\_1

stc

@@:

retn

sysmkdir\_1:

cmp ah, 0FFh

jne short @b

makdir\_4:

mov ax, word ptr [ii]

mov bx, word ptr [pdir]

mov word ptr [pdir], ax

cmp word ptr [u\_namep], bx

jb short @b ; parent dir of the new sub dir not found

sysmkdir\_flags: ; ax = r1 = mode

mov ax, 0C00Eh ; Flags (1100000000001110b)

sysmkdir\_maknod:

call mak\_nod

; ax = I-Number (also in u.dirbuff)

retn

sys\_mkdir endp

remove\_directory proc near

; 05/01/2013

; mov word ptr [u.namep], si

;call name\_i

;jc @f

;cmp ax, ROOT\_DIR\_INODE\_NUMBER

;je rmdir\_stc\_retn

;cmp ax, word ptr [u\_cdir]

;je rmdir\_stc\_retn

; INPUT ->

; ax = i\_number of directory (to be removed)

; u\_off = directory entry location + 10 (in parent dir)

; [ii] = i\_number of parent directory

cmp ax, word ptr [ii] ; '.' entry

je rmdir\_stc\_retn

mov word ptr [uf\_i\_number], ax ; i\_number of dir or file

mov dx, word ptr [u\_off]

mov word ptr [FileHandle], dx ; directory entry location + 10

mov dx, word ptr [ii] ; i-number of parent directory

mov word ptr [pdir], dx

call i\_get

jc short @f

mov ax, word ptr [inode\_flgs]

test ah, 40h ;'directory' flag

jz short rmdir\_stc\_retn

test al, 4h ; 'write' flag

jz short rmdir\_stc\_retn

xor ax, ax

mov word ptr [u\_off], ax

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

rmdir\_readi\_loop:

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

; i.e. read a directory entry

jc short @f

mov cx, word ptr [u\_nread]

or cx, cx

;jna short rmdir\_stc\_retn

jna short @f

;cmp cx, 10

;jb short @f

mov bx, word ptr [u\_dirbuf]

and bx, bx

jz short rmdir\_readi\_loop

mov ax, word ptr [u\_dirbuf]+2

cmp al, '.'

jne short rmdir\_stc\_retn

and ah, ah

jz short rmdir\_readi\_loop

cmp ah, '.' ; ".."

jne short rmdir\_stc\_retn

mov ah, byte ptr [u\_dirbuf]+4

or ah, ah

jnz short rmdir\_stc\_retn

mov ax, word ptr [u\_off]

cmp ax, 10 ; protection for removing default system directories

jna short rmdir\_stc\_retn ; because, the 1st dir enty of them is ".."

cmp ax, word ptr [inode\_size]

jb short rmdir\_readi\_loop

rmdir\_unlink:

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

mov dx, word ptr [FileHandle]

mov word ptr [u\_off], dx

call unlink

@@:

retn

rmdir\_stc\_retn:

stc

retn

remove\_directory endp

show\_file proc near

; 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 word ptr [u\_base], offset ReadBuffer

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, offset ReadBuffer

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

; 01/03/2013

; 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

jc short read\_inode\_3 ; 01/03/2013

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

jc short read\_inode\_3 ; 01/03/2013

; 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 ; offset WriteBuffer ; 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

rep movsb

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

; 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:

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

; dl = 0 = read

call i\_calc

iget\_3:

mov ax, word ptr [ii]

iget\_4:

retn

i\_get endp

i\_calc proc near

; 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\_4

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, offset WriteBuffer

add si, dx

cmp byte ptr [rw], 0

jna short icalc\_3 ; 0 = read (and copy i-node to memory)

icalc\_2:

xchg si, di

; over write old i-node (in buffer to be written)

rep movsw

mov ax, word ptr [buff\_s] ; 18/11/2012

call dsk\_wr

retn

icalc\_3:

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

rep movsw

icalc\_4:

retn

i\_calc endp

dsk\_rd proc near

; 07/07/2015 (floppy disk image file handling)

; 06/03/2013

; 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, offset WriteBuffer ; 28/11/2012

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

je short dsk\_rd\_3

mov si, ax

cmp byte ptr [PhysicalDriveNumber], 90h ; fd image file sign

jnb short image\_file\_rd

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

image\_file\_rd proc near

; 14/07/2015

; 07/07/2015

; reading a block (sector) from floppy disk image file

; INPUTS:

; ax = si = sector/block number

; bx = offset WriteBuffer = buffer address

; [img\_file\_handle] = file handle

; number of bytes to be written = 512

;

mov dx, 512

mul dx

;push bx

mov cx, dx

mov dx, ax

sub al, al ; specified offset is from the beginning of the file

mov ah, 42h ; seek (move file pointer)

mov bx, word ptr [img\_file\_handle]

int 21h

;pop bx

jc short image\_file\_rd\_err

;mov dx, bx

mov bx, word ptr [img\_file\_handle]

mov cx, 512

mov dx, offset WriteBuffer

mov ah, 3Fh ; read from file

int 21h

jc short image\_file\_rd\_err

mov bx, dx

;xor dx, dx

;cmp ax, cx ; ax = actually written bytes

;jb short image\_file\_rd\_err

mov word ptr [buff\_s], si ; current buffer sector

image\_file\_rd\_err:

retn

image\_file\_rd endp

m\_get proc near

; 05/03/2013

; 03/03/2013

; 01/03/2013

; 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)

;push bx

;push cx

;push dx

;; 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\_5 ; 4f ; large file

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

jnz short mget\_2

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

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

or ax, ax

jnz short mget\_1 ; if physical block number is zero

; then need a new block for file

call sb\_alloc ; allocate a new block for this file

; AX (R1) = Block number

jc short mget\_8 ; ; cf -> 1 & ax = 0 -> no free block

mov word ptr inode\_dskp[BX], ax

call set\_imod

call b\_clear

mget\_1: ; 2

; AX (R1) = Physical block number

;pop dx

;pop cx

;pop bx

retn

mget\_2: ; 3

; adding on block which changes small file to large file

call sb\_alloc

; call wslot ; setup I/O buffer for write

; ; R5 points to the first data word in buffer

; push ds

; pop es

mov word ptr [buff\_s], ax ; Block/Sector number

;push si

;push di

push ax

mov cx, 8 ; R3, transfer old physical block pointers

; into new indirect block area for the new

; large file

mov di, offset WriteBuffer ; BX = R5

mov si, offset inode\_dskp

xor ax, ax ; mov ax, 0

mget\_3: ; 1

movsw

mov word ptr [SI]-2, ax

loop mget\_3

mov cl, 256-8 ; clear rest of data buffer

mget\_4: ; 1

rep stosw

pop ax

;pop di

;pop si

;mov byte ptr [buff\_m], 1 ; modified

call dsk\_wr

jc short mget\_1

mov ax, word ptr [buff\_s]

mov word ptr [inode\_dskp], ax

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

call set\_imod

jmp short mget\_0

mget\_9:

pop ax

mget\_8:

;mov ax, err\_NOFREEBLOCK

;pop dx

;pop cx

;pop bx

retn

mget\_5: ; 4 ; large file

; 05/03/2013

;mov ax, bx ; bx <= 255 for this file (UNIX v1, RUFS) system

;mov cx, 256 ; 01/03/2013 no need a division here

;xor dx, dx ; 01/03/2013 no need a division here

;div cx ; 01/03/2013 no need a division here

;and bx, 1FEh ; zero all bit but 1,2,3,4,5,6,7,8

; gives offset in indirect block

;push bx ; R2

;mov bx, ax ; calculate offset in i-node for pointer

; to proper indirect block

;and bx, 0Eh

;mov ax, word ptr inode\_dskp[BX] ; R1

and bl, 0FEh ; ah = 0 ; 01/03/2013

push bx ; i-node pointer offset in indirect block

; 01/03/2013 Max. possible AX (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 ; R1

jnz short mget\_6 ; 2f

call sb\_alloc

jc short mget\_9 ; 01/03/2013

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

mov word ptr [inode\_dskp], ax ; 03/03/2013

call set\_imod

call b\_clear

mget\_6: ;2

; 05/03/2013

; ax = R1, block number

call dsk\_rd ; read indirect block

pop dx ; R2, get offset

jc short mget\_7

; BX = offset WriteBuffer

add bx, dx ; R5, first word of indirect block

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

; in file sought in R1 (AX)

or ax, ax

jnz short mget\_7 ; 2f

call sb\_alloc

jc short mget\_8 ; 01/03/2013

mov word ptr [BX], ax ; R1

push ax

mov ax, word ptr [buff\_s]

;mov byte ptr [buff\_m], 1 ; modified

;call wslot

call dsk\_wr

pop dx ; 18/11/2012

jc short mget\_7

mov ax, dx ; 18/11/2012

; ax = R1, block number of new block

call b\_clear

mget\_7: ; 2

; ax = R1, block number of new block

;pop dx

;pop cx

;pop bx

retn

m\_get endp

sb\_alloc proc near

; 14/11/2012 unix boot file configuration version

; of "alloc" procedure

; 21/8/2012

; 18/8/2012

; 17/8/2012

; 5/8/2012

; 21/7/2012

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;; input -> AX=R1

;; output -> AX=R1

; RETRO UNIX v1 FS

;push cx

push bx ; R2

;push dx ; R3

mov bx, offset systm ; SuperBlock

; start of inode and free storage map for disk

alloc\_1: ; 1

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

; in free storage map

shl ax, 1 ; multiply AX (R1) by 8 gives # of blocks

shl ax, 1

shl ax, 1

mov cx, ax ; R1, bit count of free storage map

xor ax, ax ; 0

alloc\_2: ; 1

inc bx ; 18/8/2012

inc bx ;

mov dx, word ptr [BX] ; mov (R2)+, R3

or dx, dx

jnz short alloc\_3 ; 1f

; branch if any free blocks in this word

add ax, 16

cmp ax, cx

jb short alloc\_2 ; 1b

;jmp short panic ; no free storage

xor ax, ax

stc ; cf=1 --> error: no free block

jmp short alloc\_7

alloc\_3: ; 1

shr dx, 1 ; R3 ; Branch when free block found,

; bit for block k is in byte k/8

; in bit k (mod 8)

jc short alloc\_4 ; 1f

inc ax ; R1 ; increment bit count in bit k (mod 8)

jmp short alloc\_3 ; 1b

alloc\_4:

;call free\_3

sb\_alloc\_free\_3:

mov dx, 1

mov cx, ax

and cx, 0Fh

jz short @f

shl dx, cl

@@:

mov bx, ax

shr bx, 1

shr bx, 1

shr bx, 1

shr bx, 1

free\_4: ; 1

shl bx, 1 ; 21/8/2012

; BX (R2) = k/8

add bx, offset systm+2 ; SuperBlock+2

alloc\_5: ; 1

; 21/8/2012

not dx ; masking bit is '0' and others are '1'

and word ptr [BX], dx ; bic r3, (r2)

; 0 -> allocated retn

alloc\_6:

; inc byte ptr [smod] ; super block modified sign

;mov byte ptr [smod], 1

alloc\_7:

;pop dx ; R3

pop bx ; R2

;pop cx

; AX (R1) = Block number

retn

sb\_alloc endp

set\_imod proc near

; 23/02/2013 (fromdos) file m. date&time modification

; 14/11/2012 unix boot file configuration version

; of "setimod" procedure

; 13/8/2012

; 21/7/2012

; 14/7/2012

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;; AX=R0, BX=R1, CX=R3, DX=R5

; [SP] = Argument 1, 0 = read, 1 = write

; RETRO UNIX v1 FS

; initialization/format version

;

;push dx

push ax

mov byte ptr [imod], 1

; 23/02/2013

mov ax, word ptr [inode\_ctim]

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

and ax, ax

jnz short setimod\_3

and dx, dx

jnz short setimod\_3

setimod\_1:

; Erdogan Tan 14-7-2012

call epoch

mov word ptr [inode\_ctim], ax

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

setimod\_2:

mov word ptr [inode\_mtim], ax

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

setimod\_4:

pop ax

;pop dx

retn

setimod\_3:

; 23/02/2013

xor cx, cx

cmp word ptr [inode\_mtim], cx

jna short setimod\_2

cmp word ptr [inode\_mtim]+2, cx

jna short setimod\_2

call epoch

jmp short setimod\_2

set\_imod endp

b\_clear proc near

; 18/11/2012

; 14/11/2012 unix boot file configuration version

; of "clear" procedure

; 5/8/2012

; 21/7/2012

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;; input -> AX=R1 (block number)

;; output -> AX=R1

; RETRO UNIX v1 FS

; initialization/format version

;call wslot ; setup I/O buffer for write

; ; R5 points to the first data word in buffer

; BX = R5

mov word ptr [buff\_s], ax

;push ds

;pop es

;push di

;push cx

push ax

xor ax, ax

; mov di, bx

mov di, offset WriteBuffer

mov cx, 256

rep stosw

;mov byte ptr [buff\_m], 1 ; modified

mov ax, word ptr [buff\_s] ; 18/11/2012

call dsk\_wr

pop ax

;pop cx

;pop di

retn

b\_clear endp

mak\_nod proc near

; 01/03/2013

; 23/02/2013

; 15/12/2012 UNIXCOPY.ASM version of maknod

; 02/12/2012 (maknod\_imap -> call imap)

; 25/11/2012

; 18/11/2012

; 11/11/2012

; unixboot.asm (boot file configuration)

; version of 'maknod'

;

; 30/10/2012

; AX = R1, mode

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

; RETRO UNIX v1 FS

;

; maknod : create an i-node and make a directory entry

;

; 8086 CPU & IBM PC architecture modifications by Erdogan Tan

;

; return => if cf=1 error code in AH

; If cf=0 -> AX = I-Number (also in u.dirbuff)

or ah, 80h ; 10000000b, allocate flag set

push ax ; put mode on stack

;mov ax, word ptr [ii] ; move current i-number to AX/r1

;push ax

push word ptr [ii]

mov ax, 41 ; r1 = 41

maknod\_1: ; 1 ; scan for a free i-node

inc ax ; r1 = r1 + 1

; 2/12/2012

call imap ; get byte address and bit position in inode map in

; r2 (DX) & mq (BX)

; DX (MQ) has a 1 in the calculated bit position

; BX (R2) has byte address of the byte with allocation bit

test byte ptr [BX], dl ; bitb mq,(r2) / is the i-node active

jnz short maknod\_1 ; bne 1b / yes, try the next one

or byte ptr [BX], dl ; bisb mq,(r2)

; no, make it active (put a 1 in the bit map)

; ax = i-number

call i\_get ; jsr r0,iget / get i-node into core

jc short maknod\_3

test word ptr [inode\_flgs], 8000h ; is i-node already allocated

jnz short maknod\_1 ; 1b / yes, look for another one

mov word ptr [u\_dirbuf], ax ; mov r1, u.dirbuf

; no, put i-number in u.dirbuf

pop ax ; 15/12/2012 ; get currrent i-number back

call i\_get ; jsr r0,iget / get i-node in core

jc short maknod\_2

call mk\_dir ; jsr r0,mkdir

; make a directory entry in current directory

jc short maknod\_2 ; 01/03/2013

mov ax, word ptr [u\_dirbuf] ; mov u.dirbuf,r1

; ax / r1 = new inode number

call i\_get

jc short maknod\_2

; jsr r0,copyz; inode; inode+32. / 0 it out

mov cx, 16

xor ax, ax ; 0

mov di, offset inode

rep stosw

pop word ptr [inode\_flgs] ; mov (sp)+,i.flgs / fill flags

mov cl, byte ptr [u\_uid] ; movb u.uid,i.uid / user id

mov byte ptr [inode\_uid], cl ; 23/02/2013 al -> cl

mov byte ptr [inode\_nlks], 1 ; movb $1,i.nlks / 1 link

;call epoch

;mov word ptr [s\_time], ax

;mov word ptr [s\_time]+2, dx

;mov word ptr [inode\_ctim], ax ; mov s.time,i.ctim / time created

;mov word ptr [inode\_ctim]+2, dx ; mov s.time+2,i.ctim+2

; 25/11/2012

; 23/02/2013

fromdos\_maknod:

; xor ax, ax

xor dx, dx

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

mov word ptr [inode\_mtim]+2, dx ; 0

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

jnz short maknod\_4

mov ax, word ptr [uf\_make\_datetime]

mov dx, word ptr [uf\_make\_datetime]+2

maknod\_4:

mov word ptr [inode\_ctim], ax

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

call set\_imod

mov ax, word ptr [u\_dirbuf]

retn

maknod\_3:

; 15/12/2012

pop ax

maknod\_2:

pop ax

retn

mak\_nod endp

mk\_dir proc near

; 11/11/2012

; unixboot.asm (boot file configuration)

; version of 'mkdir'

;

; 31/10/2012

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

; RETRO UNIX v1 FS

;

; mkdir : make a directory entry

;

; 8086 CPU & IBM PC architecture modifications by Erdogan Tan

;

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

; If cf=0 -> AX = I-Number (also in u.dirbuff)

;

; input:

; u.namep = file name

; ii = current directory's i-number

; u.dirbuf = directory entry (source) location

; output:

; u.dirbuf+2 to u.dirbuf+10 = file name

; u.off = directory entry offset in current directory

; u.base = start of u.dirbuf

; ;;;r1 (AX) = i-number of current directory

mkdir\_0:

; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this

mov cx, 4

xor ax, ax

mov di, offset u\_dirbuf+2

rep stosw

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

; r2 points to name of directory entry

mov di, offset u\_dirbuf+2 ; mov $u.dirbuf+2,r3

; r3 points to u.dirbuf+2

mkdir\_1: ;1 / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10

lodsb ;movb (r2)+,r1 / move character in name to r1

and al, al

jz short mkdir\_2 ; beq 1f / if null, done

cmp al, '/' ; cmp r1,$'/ / is it a "/"?

je short mkdir\_stc ; beq error9 / yes, error

cmp di, offset u\_dirbuf+10 ; cmp r3,$u.dirbuf+10.

; have we reached the last slot for

; a char?

je short mkdir\_1 ; beq 1b / yes, go back

stosb ; movb r1,(r3)+

; no, put the char in the u.dirbuf

jmp short mkdir\_1 ; br 1b / get next char

mkdir\_2: ;1

mov ax, word ptr [u\_dirp] ; mov u.dirp,u.off

mov word ptr [u\_off], ax ; pointer to empty current directory

; slot to u.off

wdir:

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

; mov$u.dirbuf,u.base

; u.base points to created file name

mov word ptr [u\_count], 10 ; mov $10.,u.count

; u.count = 10

mov ax, word ptr [ii] ; mov ii,r1

; r1 has i-number of current directory

call write\_i ; jsr r0,writei / write into directory

@@:

retn ; rts r0

mkdir\_stc:

; invalid file name, al="/", ah=0

mov ah, 1

stc

retn

mk\_dir endp

write\_i proc near

; 18/11/2012

; 11/11/2012

; unixboot.asm (boot file configuration)

; version of 'writei'

;

; 31/10/2012

; 18/08/2012

; 17/07/2012

; BX = R1, i-number

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

;

; RETRO UNIX v1 FS

; initialization/format version

;

; writei: write file

;

; 8086 CPU & IBM PC architecture modifications by Erdogan Tan

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

; input:

; AX = R1 = I-Number

; u.count = byte count

; u.base = user buffer (offset)

; u.off (u.fofp) = (pointer to) current file offset

xor dx, dx ; 0 ; clr u.nread

mov word ptr [u\_nread], dx ; clear the number of bytes transmitted during

; read or write calls

; tst u.count

cmp word ptr [u\_count], dx ; test the byte count specified by the user

;ja short write\_1 ; 1f ; bgt 1f / any bytes to output; yes, branch

;retn ; rts 0 / no, return - no writing to do

jna short write\_inode\_retn

write\_1:

;push ax ; save i-number on stack

call i\_get ; jsr r0,iget

; write i-node out (if modified), read i-node 'r1'

; into i-node area of core

mov dx, word ptr [u\_off]

add dx, word ptr [u\_count]

; add u.count,r2

; no. of bytes to be written + file offset is

; put in r2

cmp dx, word ptr [inode\_size] ; cmp r2,i.size

; is this greater than the present size of

; the file?

jna short dskw\_1 ; blos 1f / no, branch

mov word ptr [inode\_size], dx ; mov r2,i.size

; yes, increase the file size to file offset +

; no. of data bytes

call set\_imod ; jsr r0,setimod

; set imod=1 (i.e., core inode has been

; modified), stuff time of modification into

; core image of i-node

dskw\_1: ; 1

call m\_get ; jsr r0,mget

; get the block no. in which to write the next data

; byte

; AX = R1 = Block Number

mov bx, word ptr [u\_off]

and bx, 1FFh ; bit \*u.fofp,$777

; test the lower 9 bits of the file offset

jnz short dskw\_2 ; bne 2f

; if its non-zero, branch; if zero, file offset = 0,

; 512, 1024,...(i.e., start of new block)

cmp word ptr [u\_count], 512 ; cmp u.count,$512.

; if zero, is there enough data to fill an

; entire block? (i.e., no. of

;jnb short dskw\_3 ; bhis 3f / bytes to be written greater than 512.?

; Yes, branch. / Don't have to read block

; 18/11/2012

jb short dskw\_2

mov word ptr [buff\_s], ax

jmp short short dskw\_3

dskw\_2: ; 2

; in as no past info. is to be saved (the entire block will be

; overwritten).

; AX=R1 (block number)

call dsk\_rd ; jsr r0,dskrd

; no, must retain old info.. Hence, read block 'r1'

; into an I/O buffer

; 11/11/2012

jc short dskw\_5

mov ax,word ptr [buff\_s]

dskw\_3: ; 3

;call wslot

writeinode\_sioreg:

; call sioreg

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

mov cx, di ; cx = R3, di = R2

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

and di, 1FFh ; calculate file offset mod 512

add di, offset WriteBuffer ; di now points to 1st byte in buffer

; where data is to be placed

mov si, 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 writeinode\_sioreg

; SI = user data offset (r1)

; DI = sector (I/O) buffer offset (r2)

; CX = byte count (r3)

dskw\_4: ; 2

rep movsb

; ax = block/sector number

call dsk\_wr ; jsr r0,dskwr / write the block and the i-node

jc short dskw\_5

cmp word ptr [u\_count], 0 ; any more data to write?

ja short dskw\_1 ; 1b ; yes, branch

dskw\_5:

; pop ax ; i-number

write\_inode\_retn:

retn

write\_i endp

dsk\_wr proc near

; 07/07/2015 (floppy disk image file handling)

; 11/11/2012

; unix boot file configuration version of "dskwr" procedure

;

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

; RETRO UNIX v1 FS

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

; ax = sector/block number

cmp byte ptr [PhysicalDriveNumber], 90h ; fd image file sign

jnb short image\_file\_wr ; 07/07/2015

mov bx, offset WriteBuffer

xor ch, ch

mov cl, 4 ; Retry count

dsk\_wr\_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, 3 ; 3=write

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\_wr\_2

loop dsk\_wr\_1

dsk\_wr\_2:

retn

dsk\_wr endp

image\_file\_wr proc near

; 14/07/2015

; 07/07/2015

; writing a block (sector) to floppy disk image file

; INPUTS:

; ax = sector/block number

; offset WriteBuffer = buffer address

; [img\_file\_handle] = file handle

; number of bytes to be written = 512

;

mov dx, 512

mul dx

mov cx, dx

mov dx, ax

sub al, al ; specified offset is from the beginning of the file

mov ah, 42h ; seek (move file pointer)

mov bx, word ptr [img\_file\_handle]

int 21h

;mov bx, word ptr [img\_file\_handle]

mov cx, 512

mov dx, offset WriteBuffer

mov ah, 40h ; write to file

int 21h

jc short image\_file\_wr\_err

mov bx, dx

;xor dx, dx

cmp ax, cx ; ax = actually written bytes

image\_file\_wr\_err:

retn

image\_file\_wr endp

epoch proc near

; 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

; 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 proc\_mul32

mov bl, byte ptr [hour]

add ax, bx

adc dx, 0

mov bx, 60

call proc\_mul32

mov bl, byte ptr [minute]

add ax, bx

adc dx, 0

mov bx, 60

call proc\_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

epoch 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

sync proc near

; 14/07/2015

; 07/07/2015

; 18/11/2012 unix boot file configuration version

; of "sync" procedure of retro unix v1.0 by Erdogan Tan

; 12/8/2012

; updates super block and the last i-node on disk

; if modified

; e.g. smod = 1, imod = 1, buffer\_m = 1

;

; RETRO UNIX v1 FS

xor ax, ax ; mov ax, 0

call i\_get ; (write modified i-node)

jc short sync\_3

sync\_1:

; 14/07/2015

; 07/07/2015

mov dl, byte ptr [PhysicalDriveNumber]

cmp dl, 90h

jb short sync\_2

sub dx, dx ; 0

mov cx, dx ; 0

sub al, al ; specified offset is from the beginning of the file

mov ah, 42h ; seek (move file pointer)

mov bx, word ptr [img\_file\_handle]

int 21h

jc loc\_error

mov bx, word ptr [img\_file\_handle]

mov cx, 1024 ; write 1024 bytes (2 sectors)

mov dx, offset BSBuffer ; bootsector (& super block) buffer

mov ah, 40h ; write file

int 21h

jc loc\_error

cmp ax, 1024

jne loc\_error

;mov bx, dx ; offset BSBuffer

retn

sync\_2:

mov bx, offset BSBuffer

mov ax,0302h ; Write boot sector & super block

mov cx,1

xor dh,dh

mov dl, byte ptr [PhysicalDriveNumber]

int 13h

sync\_3:

retn

sync 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

; 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, offset BSBuffer

mov ax, word ptr [SI]+bs\_bf\_inode\_number

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

unlink proc near

; 05/01/2013 UNIXCOPY.ASM modification (pdir -> i\_get)

; 16/12/2012 UNIXCOPY.ASM version

; 02/12/2012

; unix boot file configuration version

; of "sysunlink" function of retro unix v1.0 by Erdogan Tan

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

; ('sysunlink', unix kernel function)

;

; INPUT -> AX (R1) = inode number

; [u\_off] = Directory Entry Offset + 10

; ;;; [ii] = i-number of current directory

; Return -> CF = 0 -> Successed, CF = 1 -> failed

; (error code in AX)

;jsr r0,arg; u.namep / u.namep points to name

;jsr r0,namei / find the i-number associated

; with the path name

;br error9 / not found

push ax ;mov r1,-(sp) / put its i-number on the stack

;jsr r0,isdir / is it a directory

xor ax, ax

mov word ptr [u\_dirbuf], ax ; clr u.dirbuf / no, clear

;the location that will get written

;/ into the i-number portion of the entry

sub word ptr [u\_off], 10 ; sub $10.,u.off

; / move u.off back 1 directory entry

;mov ax, word ptr [ii]

mov ax, word ptr [pdir] ; 05/01/2013

call i\_get

jnc short @f

pop ax

retn

;

@@:

call wdir ;jsr r0,wdir / free the directory entry

pop ax ;mov (sp)+,r1 / get i-number back

jc short @f

call i\_get ; jsr r0,iget / get i-node

jc short @f

call set\_imod ; jsr r0,setimod / set modified flag

dec byte ptr [inode\_nlks] ; decb i.nlks

; / decrement the number of links

jnz short @f ;bgt sysret9

;/ if this was not the last link to file return

call anyi ;jsr r0,anyi / if it was, see if anyone has it open.

;Then / free contents of file and destroy it.

;br sysret9

@@:

retn

unlink endp

anyi proc near

; 02/12/2012

; unix boot file configuration version

; of "anyi" procedure of retro unix v1.0 by Erdogan Tan

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

; ('anyi' procedure)

;

; INPUT -> AX (R1) = inode number

; Return -> CF = 0 -> Successed, CF = 1 -> failed

;

; mov $fsp,r2 / move start of fsp table to r2

anyi\_1: ;1

; cmp r1,(r2) / do i-numbers match?

; beq 1f / yes, 1f

; neg r1 / no complement r1

; cmp r1,(r2) / do they match now?

; beq 1f / yes, transfer

; / i-numbers do not match

; add $8,r2 / no, bump to next entry in fsp table

; cmp r2,$fsp+[nfiles\*8] / are we at last entry in the table

; blt 1b / no, check next entries i-number

; tst r1 / yes, no match

; bge .+4

; neg r1 / make i-number positive

call imap ; jsr r0,imap / get address of allocation bit

; in the i-map in r2

; DX (MQ) has a 1 in the calculated bit position

; BX (R2) has byte address of the byte with allocation bit

push bx ; retro unix modification (not as original unix code)

push dx ; retro unix modification (not as original unix code)

; AX = i-number

call itrunc ; jsr r0,itrunc / free all blocks related to i-node

pop dx ; retro unix modification (not as original unix code)

pop bx ; retro unix modification (not as original unix code)

jc short @f

; (AX=0)

; retro unix modification-> 'call itrunc' moved up for

; keeping super block unmodified if itrunc return with an error

not dx

and byte ptr [BX], dl ; bicb mq,(r2)

; / clear bit for i-node in the imap

; xor ax, ax

mov word ptr [inode\_flgs], ax ; 0 ; clr i.flgs

; / clear all flags in the i-node

@@:

retn ; rts r0 / return

;anyi\_2: ;1 / i-numbers match

; incb 7(r2) / increment upper byte of the 4th word

; rts r0 / in that fsp entry (deleted flag of fsp entry)

anyi endp

imap proc near

; 02/12/2012

; unix boot file configuration version

; of "imap" procedure of retro unix v1.0 by Erdogan Tan

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

; ('imap' procedure)

; 11/11/2012 (maknod\_imap location -> imap procedure)

mov bx, ax ; BX = R2, AX = R1 (input, i-number)

sub bx, 41 ; BX has i-41

mov cl, bl ; CX = R3

mov dx, 1 ;

and cl, 7 ; CX has (i-41) mod 8 to get the bit position

jz short @f ; 21/8/2012

shl dx, cl ; DX has 1 in the calculated bit position

@@:

shr bx, 1

shr bx, 1

shr bx, 1 ; BX has (i-41) base 8 of byte number

; from the start of the (inode) map

add bx, word ptr [systm] ; superblock free map size + 4

add bx, offset systm + 4 ; is inode map offset in superblock

; DX (MQ) has a 1 in the calculated bit position

; BX (R2) has byte address of the byte with allocation bit

retn

imap endp

itrunc proc near

; 10/03/2013 BugFix

; 01/12/2012

; unix boot file configuration version

; of "itrunc" procedure of retro unix v1.0 by Erdogan Tan

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

; ('itrunch' procedure)

;

; INPUT -> AX (R1) = inode number

; Return -> CF = 0 -> Successed (AX=0), CF = 1 -> failed

; (error code in AX)

call i\_get ; jsr r0,iget

jc short itrunc\_7 ; 10/03/2013

mov si, offset inode\_dskp ; mov $i.dskp,r2

mov di, si

itrunc\_1:

; 10/03/2013 BugFix ('lodsb' -> 'lodsw')

lodsw ; mov (r2)+,r1 / move physical block number into r1

or ax,ax

jz short itrunc\_5 ; beq 5f

push si ; mov r2,-(sp)

test word ptr [inode\_flgs], 1000h

; bit $10000,i.flgs / test large file bit?

jz short itrunc\_4 ; beq 4f / if clear, branch

push ax ; mov r1,-(sp) / save block number of indirect block

call dsk\_rd ; jsr r0,dskrd / read in block,

; bx = Buffer offset ; 1st data word pointed to by r5

jnc short itrunc\_6 ; 10/03/2013

pop si ; 10/03/2013

pop si ; 10/03/2013

itrunc\_7:

retn ; 10/03/2013

itrunc\_6:

mov cx, 256 ; mov $256.,r3 / move word count into r3

mov si, bx ; 10/03/2013 (SI more proper here than BX)

itrunc\_2:

;mov ax, word ptr [BX] ; mov (r5)+,r1

; / put 1st data word in r1; physical block number

lodsw ; 10/03/2013 ; mov ax, word ptr [SI] ; add si, 2

;inc bx

;inc bx ; 10/03/2013 (BugFix)

and ax, ax

jz itrunc\_3 ; beq 3f / branch if zero

push cx ; mov r3,-(sp) / save r3, r5 on stack

;push bx ; mov r5,-(sp) ; 10/03/2013, push bx is not needed

call free ; jsr r0,free / free block in free storage map

;pop bx ; mov (sp)+,r5 ; 10/03/2013, push bx is not needed

pop cx ; mov (sp)+,r3

itrunc\_3:

loop itrunc\_2 ; dec r3 / decrement word count

; bgt 2b / branch if positive

pop ax ; mov (sp)+,r1

; / put physical block number of indirect block

itrunc\_4:

call free ; jsr r0,free / free indirect block

pop si ; mov (sp)+,r2

itrunc\_5:

cmp si, offset inode\_dskp + 16 ; cmp r2,$i.dskp+16.

jb short itrunc\_1 ; bne 1b

; / branch until all i.dskp entries check

and word ptr [inode\_flgs], 0EFFFh ; 1110111111111111b

; bic $10000,i.flgs / clear large file bit

mov cx, 8

xor ax, ax

mov word ptr [inode\_size], ax

; clr i.size / zero file size

;mov di, offset inode\_dskp

rep stosw ; jsr r0,copyz; i.dskp; i.dskp+16.

; / zero block pointers

call set\_imod ; jsr r0,setimod

; / set i-node modified flag

;mov ax, word ptr [ii] ; mov ii,r1

retn ; rts r0

itrunc endp

free proc near

; 01/12/2012

; unix boot file configuration version

; of "free" procedure of retro unix v1.0 by Erdogan Tan

; Derived from (original) UNIX v1 source code

; PRELIMINARY release of Unix Implementation Document,

; 20/6/1972

; ('free' procedure)

;

; INPUT -> ax (R1) = physical block number

; Return -> CF = 0 -> Successed, CF = 1 -> failed

;

;push bx ; mov r2,-(sp) / save r2, r3

;;push cx ; mov r3,-(sp)

;push dx

;call free\_3 ; jsr r0,3f

; / set up bit mask and word no. in free storage map

; / for block

free\_3: ; 3

mov dx, 1

mov cx, ax ; mov r1,r2 / block number, k, = 1

and cx, 0Fh ; bic $!7,r2 / clear all bits but 0,1,2; r2 = (k) mod (8)

; clr r3

jz short @f

; bisb 2f(r2),r3 / use mask to set bit in r3

; corresponding to / (k) mod 8

shl dx, cl

@@:

mov bx, ax ; mov r1,r2 / divide block number by 16

shr bx, 1 ; asr r2

shr bx, 1 ; asr r2

shr bx, 1 ; asr r2

shr bx, 1 ; asr r2

; bcc 1f / branch if bit 3 in r1 was 0 i.e.,

; bit for block is in / lower half of word

; swab r3 / swap bytes in r3; bit in

; upper half of word in free / storage map

free\_1: ; 1

shl bx, 1 ; asl r2 / multiply block number by 2; r2 = k/8

add bx, offset systm+2 ; add $systm+2,r2

; / address of word of free storage map for drum

; / with block bit in it

; retn ; rts r0 (return from free\_3)

@@:

or word ptr [BX], dx ; bis r3, (r2)

; / set free storage block bit; indicates free block

; 0 -> allocated, 1 -> free

;;inc byte ptr [smod] ; incb smod / set super block modified for drum

;mov byte ptr [smod], 1 ; / set super block modified for drum

;pop dx

;pop cx ; mov (sp)+,r3 / restore r2, r3

;pop bx ; mov (sp)+,r2

; AX (R1) = Block number

retn

free endp

chmode proc near

; 13/01/2013

; 'change mode' procedure

; Format: chmod <octal number> <unix file name>

;

; output -> cf=1 -> error

; -> cf=0 -> word ptr [arg] > 0 -> mode (string, 2 chars)

; word ptr [arg] = 0 -> ignored (none is done)

xor ax, ax

mov word ptr [arg], ax

chmode\_1:

lodsb

cmp al, '0'

jb short chmode\_8

cmp al, '7'

ja short chmode\_stc\_retn ; cmc

chmode\_2:

or ah, ah

jnz short chmode\_3

mov ah, al

jmp short chmode\_1

chmode\_stc\_retn:

cmc

chmode\_retn:

retn

chmode\_8:

or ah, ah

jz short chmode\_stc\_retn

cmp al, 20h

jne short chmode\_stc\_retn

mov al, ah

mov ah, '0'

jmp short chmode\_4

chmode\_3:

cmp byte ptr [SI], 20h

; no error if the 3rd character is a carriage return

jne short chmode\_stc\_retn

inc si

chmode\_4:

mov word ptr [arg], ax

chmode\_5:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short chmode\_5

jb short chmode\_stc\_retn ; no error (carriage return)

chmode\_6:

call name\_i

jc short chmode\_retn

; ax = i-number

call i\_get

jc short chmode\_retn

mov ax, word ptr [arg]

xchg ah, al

mov word ptr [arg], ax

sub ax, '00' ; 3030h

shl al, 1

shl al, 1

shl al, 1

add al, ah

mov bx, offset inode\_flgs

test word ptr [BX], 4000h ; directory ?

jz short chmode\_7

; clear 'set user id' and 'executable' flags

and al , 0Fh ; 1111b

chmode\_7:

mov byte ptr [BX], al

mov byte ptr [imod], 1

;xor ah, ah

retn

chmode endp

chowner proc near

; 13/01/2013

; 'change owner' procedure

; Format: chown <decimal number> <unix file name>

;

; output -> cf=1 -> error

; -> cf=0 ->

; BX > 0 -> offset arg == owner (decimal string, 3 chars)

; BX = 0 -> ignored (none is done)

xor ax, ax

mov di, offset arg

mov [di], ax ; 0

xor bx, bx

; mov cx, 3

mov cx, '90'

; xor dx, dx

xor dl, dl

chowner\_1:

lodsb

cmp al, cl ; '0'

jb short chowner\_5

cmp al, ch ; '9'

ja short chowner\_stc\_retn ; cmc

inc dl

chowner\_2:

or bl, bl

jnz short chowner\_3

cmp al, cl ; '0'

je short chowner\_1

jmp short chowner\_4

chowner\_3:

push ax

mov al, 10

mul bl

mov bx, ax

pop ax

chowner\_4:

sub al, cl ; '0'

add bx, ax

or bh, bh

jz short chowner\_7

xor bx, bx

chowner\_stc\_retn:

cmc

retn

chowner\_5:

and dl, dl

jz short chowner\_retn

cmp al, 20h

je short chowner\_8

chowner\_6:

mov bx, 0

jmp short chowner\_stc\_retn

chowner\_7:

add al, cl ;'0'

stosb

jmp short chowner\_1

;loop chowner\_1

;cmp byte ptr [SI], 20h

;; no error if the 4th character is a carriage return

;jne short chowner\_6

;inc si

chowner\_8:

mov word ptr [u\_namep], si

lodsb

cmp al, 20h

je short chowner\_8

jb short chowner\_6 ; no error (carriage return)

;mov byte ptr [u\_uid], bl

push bx

call name\_i

jc short chowner\_9

; ax = i-number

call i\_get

chowner\_9:

;pushf

;mov bl, byte ptr [u\_uid]

;xor bh, bh

;mov byte ptr [u\_uid], bh ; 0

;popf

pop bx

jc short chowner\_retn

mov byte ptr [inode\_uid], bl

mov byte ptr [imod], 1

or bl, bl

jnz short chowner\_retn

mov bh, '0'

mov byte ptr [arg], bh

chowner\_retn:

retn

chowner 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

; 16/02/2013

mov bx, offset BSBuffer

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, offset systm ; 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, offset systm+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 [systm]

add bx, offset systm + 2

; 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

; 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 [systm]

add bx, offset systm+2

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

PhysicalDriveNumber: db 0

FileHandle: dw 0

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

; messages

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

UNIX\_Welcome:

db 0Dh, 0Ah

db 'Retro UNIX 8086 v1 FS File Import/Export Utility'

db 0Dh, 0Ah

db 'UNIXCOPY by Erdogan TAN 2012 - [14/07/2015]'

db 0Dh, 0Ah

db '(Type ', 27h,'?',27h,' to see valid commands)'

db 0Dh,0Ah

db 0Dh,0Ah, 0

usage:

db 'Usage: unixcopy [Floppy Drive or File Name] '

db 0Dh,0Ah

db 0Dh,0Ah

db "Floppy Drive names:"

db 0Dh,0Ah

db 0Dh,0Ah

db "fd0 (Floppy Disk 1, A:)", 0Dh, 0Ah

db "fd1 (Floppy Disk 2, B:)", 0Dh, 0Ah

db 0Dh,0Ah

db "Floppy Disk Image File name examples:"

db 0Dh,0Ah

db 0Dh,0Ah

db "fd0.img", 0Dh, 0Ah

db "fd1.img", 0Dh, 0Ah

db "runixfd.img", 0Dh, 0Ah

db 0

unix\_cdrv:

UNIX\_FD\_Name:

db 'fd'

UNIX\_FD\_Number:

db '0:', 0

unix\_img\_cdir: db '!' ; 07/07/2015

unix\_cdir: db '/'

db 37 dup(0)

CDirOffset: dw 0

unix\_prompt\_char: db '>'

CursorColumn: dw 0

CommandBuffer: db 74 dup(0)

program\_exit: db 0

pdir: dw 0

msg\_arg: db 0Dh, 0Ah ; 13/01/2012 (chmod)

arg: dw 0

dw 0 ; 13/01/2012

msg\_yes\_no:

db '(Yes/No)? ', 0

msg\_unix\_drv\_read\_error:

db 0Dh, 0Ah

db "Drive not ready or read error!"

db 0Dh, 0Ah, 0

msg\_inv\_file\_name: ; 07/07/2015

db 0dh, 0Ah

db "Invalid file name !", 0Dh, 0Ah

db "(File name must fit for 8.3 DOS format) !"

db 0Dh, 0Ah, 0

msg\_file\_not\_found: ; 07/07/2015

db 0Dh, 0Ah

db "File not found !", 0Dh, 0Ah

db "(File must be in current directory) !"

db 0Dh, 0Ah, 0

msg\_inv\_image\_file: ; 07/07/2015

db 0Dh, 0Ah

db "Invalid floppy disk image file !", 0Dh, 0Ah

db "(File size must be 1474560 bytes) !"

db 0Dh, 0Ah, 0

Msg\_Not\_Unix\_FS:

db 0Dh, 0Ah

db "Drive has not got a Retro UNIX v1 FS !"

db 0Dh, 0Ah, 0

Msg\_writing\_file:

db 0Dh, 0Ah

db "Writing file..."

db 0

Msg\_Removing\_file:

db 0Dh, 0Ah

db "Deleting file..."

db 0

Msg\_DosFile\_Name:

db 0Dh, 0Ah

db "DOS File Name : ", 0

Msg\_StartupFile\_Name:

db 0Dh, 0Ah

db "Startup File Name : ", 0

Msg\_3dot\_OK: db "..."

Msg\_OK:

db ' OK.', 0Dh, 0Ah, 0

msg\_YES: db ' YES'

db 0

msg\_NO: db ' NO'

db 0

error\_msg:

db 0Dh, 0Ah

db 'Error !'

UNIX\_CRLF:

db 0Dh, 0Ah, 0

msg\_making\_directory:

db 0Dh, 0Ah

db "Making directory..."

db 0

msg\_removing\_directory:

db 0Dh, 0Ah

db "Removing directory..."

db 0

msg\_unix\_drv\_write\_error:

db 0Dh, 0Ah

db 'Drive not ready or write error!'

db 0Dh, 0Ah

db 0

msg\_Startup\_File\_Not\_Exists:

db 0Dh, 0Ah

db 'Startup File is not configured yet ! '

db 0Dh, 0Ah, 0

msg\_sf\_configuration\_set\_ok:

db 0Dh, 0Ah

db "Startup file configuration SET is OK."

db 0Dh, 0Ah, 0

msg\_sf\_configuration\_reset\_ok:

db 0Dh, 0Ah

db "Startup file configuration RESET is OK."

db 0Dh, 0Ah, 0

msg\_overwrite\_question1:

; 1/12/2012

db 0Dh, 0Ah

db 'Do you want to overwrite '

db 27h

db 0

msg\_overwrite\_question2:

db 27h

db ' file '

db 0

msg\_remove\_question1:

; 1/12/2012

db 0Dh, 0Ah

db 'Do you want to delete '

db 27h

db 0

msg\_remove\_question2:

db 27h

db ' file '

db 0

align 2

RetryCount: dw 0

; 07/07/2015

img\_file\_name: db 13 dup(0)

db 0

img\_file\_handle: dw 0

;img\_file\_pos: dd 0 ; file (position) pointer

;

DirFileName:

db 20h ; 06/01/2013

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

uf\_make\_datetime: dd 0 ; 25/11/2012

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

rw: db 0

imod: db 0

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

; 30/11/2012

;imin: dd 0

;ihrs: dd 0

;iday: dw 0

;lday: dw 0

;qday: dw 0

;iyrs: dw 0

;jday: dw 0

;mday: dw 0

; 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) ; 03/02/2013, 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

UNIXCOPY\_Commands: ; 23/02/2013

db 0Dh, 0Ah

db "UNIXCOPY COMMANDS [", 27h, "/", 27h, " means alternative, ", 27h, "<...>", 27h, " means command argument]", 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 "chdir/cd <directory name> : change directory", 0Dh, 0Ah

db "fromdos <dos file name> <unix file name> : copy/import dos file to unix fs", 0Dh, 0Ah

db "todos <unix file name> <dos file name> : copy/export unix file to dos fs", 0Dh, 0Ah

db "rm <file name> : remove/delete/unlink file", 0Dh, 0Ah

db "mkdir <directory name> : make new sub directory", 0Dh, 0Ah

db "rmdir <directory name> : remove/delete sub directory", 0Dh, 0Ah

db "link <source file name> <destination file name> : link file to file", 0Dh, 0Ah

db "exit : return to dos", 0Dh, 0Ah

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

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

db "chmod <mode> <file name> : change file mode (according to octal number)", 0Dh, 0Ah

db "chown <owner> <file name> : change file's owner (according to decimal number)", 0Dh, 0Ah

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

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

db "bootfile <file name> : select/configure file as startup/boot file", 0Dh, 0Ah

db "nobootfile : reset/cancel bootfile configuration", 0Dh, 0Ah

db "?/help : print/display UNIXCOPY commands summary (as above)", 0Dh, 0Ah, 0

align 16

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

; buffers

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

BSBUFFER: db 512 dup(0)

; superblock

systm: db 512 dup(0)

ReadBUFFER: db 512 dup(0)

WriteBUFFER: db 512 dup(0)

SizeOfFile equ $-100

UNIXCOPY ends

end START\_CODE