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

; LS.ASM (Retro Unix 8086 v1 - /bin/ls - list file or directory)

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

; RETRO UNIX 8086 (Retro Unix == Turkish Rational Unix)

; Operating System Project (v0.1) by ERDOGAN TAN (Beginning: 11/07/2012)

; Retro UNIX 8086 v1 - /bin/ls file

;

; [ Last Modification: 07/10/2015 ]

;

; Derivation from UNIX Operating System (v1.0 for PDP-11)

; (Original) Source Code by Ken Thompson (Bell Laboratories, 1971-1972)

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

; Derived from 'ls.s' file of original UNIX v1

;

; LS1.ASM, 07/10/2015

; LS0.ASM, 19/11/2013 - 29/11/2013

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

.8086

; UNIX v1 system calls

\_rele equ 0

\_exit equ 1

\_fork equ 2

\_read equ 3

\_write equ 4

\_open equ 5

\_close equ 6

\_wait equ 7

\_creat equ 8

\_link equ 9

\_unlink equ 10

\_exec equ 11

\_chdir equ 12

\_time equ 13

\_mkdir equ 14

\_chmod equ 15

\_chown equ 16

\_break equ 17

\_stat equ 18

\_seek equ 19

\_tell equ 20

\_mount equ 21

\_umount equ 22

\_setuid equ 23

\_getuid equ 24

\_stime equ 25

\_quit equ 26

\_intr equ 27

\_fstat equ 28

\_emt equ 29

\_mdate equ 30

\_stty equ 31

\_gtty equ 32

\_ilgins equ 33

sys macro syscallnumber, arg1, arg2, arg3

; Retro UNIX 8086 v1 system call.

ifnb <arg1>

mov bx, arg1

endif

ifnb <arg2>

mov cx, arg2

endif

ifnb <arg3>

mov dx, arg3

endif

mov ax, syscallnumber

int 20h

endm

; Retro UNIX 8086 v1 system call format:

; sys systemcall (ax) <arg1 (bx)>, <arg2 (cx)>, <arg3 (dx)>

UNIX SEGMENT PUBLIC 'CODE'

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

START\_CODE:

; / ls -- list file or directory

;.globl flush

;.globl fopen

;.globl getw

;.globl getc

;.globl putc

;.globl ctime

;.globl end

;mov ax, offset \_end + 512

;and al, 0FEh

;cmp ax, sp

;jna short @f

;mov sp, ax

;@@:

;mov bx, ax

; Retro UNIX 8086 v1 modification:

; 'sys break' is not needed to extend

; current user core memory

; (because of 8086 segmentation and 32 kB

; memory allocation);

; but, it is needed to clear/reset

; user core memory beyond of (after) previous

; 'u.break' which depends on executable

; file size; because 'bss'

; data is not in current executable file

; ('BSS' is an external data structure after

; the last byte of the executable file).

;

; sys break

; clears memory from 'bss' to 'bss.\_end+512'

mov bx, offset \_end + 512

sys \_break

; sys break; end+512.

;

sys \_write, 1, nl, 2

;

mov word ptr [obuf], bx ; 1

; mov $1,obuf

mov si, sp

; mov sp,r5

lodsw

dec ax

mov word ptr [count], ax

; mov (r5)+,count

; tst (r5)+

; dec count

mov word ptr [ocount], ax

; mov count,ocount

; bgt loop

; mov $dotp,r5

jna short B0

;and ax, ax

;jnz short @f

;jz short B0

;mov si, offset dotp

;jmp short @loop

;@@:

lodsw

@loop: ;loop:

lodsw

mov bx, ax

;mov (r5)+,r4

cmp byte ptr [BX], '-'

; cmpb (r4)+,$'-

jne short A1

; bne 1f

inc bx

dec word ptr [ocount]

; dec ocount

A3: ;3:

mov al, byte ptr [BX]

; movb (r4)+,r0

inc bx

or al, al

jz short eloop

; beq eloop

cmp al, 'l'

; cmp r0,$'l

jne short @f

; bne 4f

inc word ptr [longf]

; inc longf

jmp short A3

; br 3b

@@: ;4:

cmp al, 't'

; cmpb r0,$'t

jne short @f

; bne 4f

mov word ptr [sortoff], 14

; mov $14.,sortoff

jmp short A3

; br 3b

@@: ;4:

cmp al, 'a'

; cmpb r0,$'a

jne short @f

; bne 4f

inc word ptr [allflg]

; inc allflg

jmp short A3

; br 3b

@@: ;4:

cmp al, 's'

; cmpb r0,$'s

jne short @f

; bne 4f

inc byte ptr [longf]+1

; incb longf+1

jmp short A3

; br 3b

@@: ;4:

cmp al, 'd'

; cmpb r0,$'d

jne short A3

; bne 3b

inc word ptr [dirflg]

; inc dirflg

jmp short A3

; br 3b

A1: ;1:

;dec bx

; dec r4

call do

; jsr pc,do

eloop:

dec word ptr [count]

; dec count

jg short @loop

; bgt loop

mov ax, word ptr [dnp]

and ax, ax

;tst dnp

jnz short @f

; bne 1f

B0:

mov si, offset dotp

; mov $dotp,r5

jmp short @loop

; br loop

@@: ;1:

mov si, offset obuf

call flush

; jsr r5,flush; obuf

sys \_exit

; sys exit

;; 20 bytes listing (source) data

;; structure:

;; offset

;; 0-7 : file name

;; 8-9 : flags

;; 10-11 : nlinks, uid

;; 12-13 : size

;; 14-15-16-17 : mtime

;; 18-19 : inode number

do:

push si ; r5

sub ax, ax

mov word ptr [tblocks], ax ; 0

; clr tblocks

mov bp, offset \_end

; mov $end,r1

mov di, offset filnam

; mov $filnam,r3

mov word ptr [dnp], bx

; mov r4,dnp

mov si, bx ; r4

mov word ptr [isadir], ax ; 0

; clr isadir

cmp word ptr [dirflg], ax ; 0

; tst dirflg

ja nondir

; bne nondir

;mov bx, word ptr [dnp]

mov cx, offset statb

sys \_stat

; sys stat; dnp: 0; statb

jnc short B1

; bec 1f

; BX = file name

mov si, offset @f

do\_err:

call questf

pop si

retn

;jsr r5,questf; < nonexistent\n\0>; .even

; rts pc

@@:

db ' nonexistent', 0Dh, 0Ah, 0

B1: ;1:

;test word ptr [statb]+2, 4000h

test byte ptr [statb]+3, 40h

; bit $40000,statb+2 /test directory

jz short nondir

; beq nondir

inc word ptr [isadir]

; inc isadir

;mov ax, bx

; mov r4,r0

push di

mov di, offset dbuf

call fopen

; jsr r5,fopen; dbuf

pop di

jnc short B2

; bcc 1f

; BX = file name

mov si, offset @f

jmp short do\_err

;call questf

;pop si

;retn

; jsr r5,questf; < unreadable\n\0>; .even

; rts pc

@@:

db ' unreadable', 0Dh, 0Ah, 0

B2:

; mov si, bx ; r4

@@: ;1:

lodsb

stosb

;movb (r4)+,(r3)+

or al, al

jnz short @b

; bne 1b

dec di

; dec r3

;

cmp byte ptr [DI]-1,'/'

; cmpb -1(r3),$'/

je short B3

; beq 1f

mov al, '/'

stosb

; movb $'/,(r3)+

B3: ;1:

;mov bx, offset dbuf

mov si, offset dbuf

@@:

call getw

; jsr r5,getw; dbuf

jc short pass2

; bcs pass2

mov cx, 4

; mov $4,-(sp)

and ax, ax

; tst r0

jnz short B5

; bne 2f

B4: ;3:

push cx

; mov si, offset dbuf

call getw

; jsr r5,getw; dbuf

pop cx

loop B4

; dec (sp)

; bne 3b

; tst (sp)+

jmp short @b

;jmp short B3

; br 1b

B5: ;2:

; DI == r2

; mov r3,r2

push di ; r3 (filnam +'/'+1)

B6: ;2:

;; copying file name

;; to listing (source) data address (BP)

;; (offset 0-7)

;; and filnam (DI)

;

push cx

; mov si, offset dbuf

call getw

; jsr r5,getw; dbuf

mov word ptr [BP], ax

inc bp

inc bp

; mov r0,(r1)+

stosw

;stosb

; movb r0,(r2)+

;xchg al, ah

; swab r0

;stosb

; movb r0,(r2)+

pop cx

loop B6

; dec (sp)

; bne 2b

; tst (sp)+

xor ax, ax ; 0

stosb

; clrb (r2)+

pop di ; r3

cmp word ptr [allflg], ax ; 0

; tst allflg

ja short B7

; bne 2f

cmp byte ptr [DI], '.'

; cmpb (r3),$'.

jne short B7

; bne 2f

sub bp, 8

; sub $8.,r1

jmp short @b

;jmp short B3

; br 1b

B7: ;2:

;; copying 12 bytes inode data to

;; listing (source) data from offset

;; 8 to offset 19 (of 20 data bytes)

;

call gstat

; jsr r5,gstat

jmp short B3

; br 1b

nondir:

; mov si, bx ; r4

mov bx, di ; offset filnam

;mov r3,r2

@@: ;1:

; SI points to file name (input)

lodsb

stosb

; movb (r4)+,(r2)+

and al, al

jnz @b

; bne 1b

@@: ;1:

cmp di, bx ; offset filnam

; cmp r2,r3

jna short @f

; blos 1f

dec di

cmp byte ptr [DI], '/'

; cmpb -(r2),$'/

jne short @b

; bne 1b

inc di

; inc r2

;; DI points to last name

;; of the path (after "/")

@@: ;1:

mov cx, 8

; mov $8.,-(sp)

ndloop: ;1:

mov al, byte ptr [DI]

mov byte ptr [BP], al

inc bp

; movb (r2)+,(r1)+

; bne 2f

; dec r2

or al, al

jz short @f

inc di

@@:

loop ndloop

call gstat ; fill/get 12 bytes listing data

;jmp short pass2

@@: ;2:

; dec (sp)

; bne 1b

; jsr r5,gstat

; tst (sp)+

pass2:

mov bx, word ptr [dbuf]

; mov dbuf,r0

sys \_close

; sys close

mov cx, bx ; file descriptor

mov bx, offset \_end

; mov $end,r2

cmp bp, bx ; bp >= \_end (= last word + 2)

; cmp r1,r2

jne short C1

; bne 1f

pop si ; r5

retn

; rts pc

C1: ;1:

; sorting begins here

; mov r5,-(sp)

mov di, bp ; current end of listing words (+2)

push bp ; r1

; mov r1,-(sp)

; BX will point to mtime or file name (+14 or 0)

; offset of 20 bytes listing (source) data

add bx, word ptr [sortoff]

; add sortoff,r2

C2: ;1:

mov ax, bx

stosw

; mov r2,(r1)+

add bx, 20 ; bx now points to next 20 bytes

; add $20.,r2

cmp bx, bp ; is BX passed the data limit ?

; cmp r2,(sp)

jb short C2

; blo 1b

@@:

mov bx, bp

; mov (sp),r2

dec di

dec di

; tst -(r1)

C3: ;1:

mov dx, di ; r1

@@:

;mov bp, bx

; mov r2,r3

C4: ;2:

inc bp

inc bp

; tst (r3)+

cmp bp, dx

; cmp r3,r1

ja short C7

; bhi 2f

mov si, word ptr [BX] ; file name 1 or time 1

; mov (r2),r4

mov di, word ptr [BP] ; file name 2 or time 2

; mov (r3),r5

cmp word ptr [sortoff], 0

; tst sortoff

jna short C5

; beq 4f

; sorting by modification time

cmpsw

lahf

; cmp (r4)+,(r5)+

;jb short C6

; blo 3f

;ja short C4

; bhi 2b

cmpsw

; cmp (r4)+,(r5)+

jb short C6

; blo 3f

ja short C4

shr ah, 1

jc short C6

jmp short C4

; br 2b

; sorting by file name

C5: ;4:

; ?

;; mov cx, 8

C5x: ;4:

cmpsb

; cmpb (r4)+,(r5)+

ja short C6

; bhi 3f

jb short C4

; blo 2b

;dec cx ; ?

; dec r0

;jnz short C5x ?

jmp short C5x

;jmp short C5

; br 4b

C6: ;3:

push word ptr [BX]

mov ax, word ptr [BP]

mov word ptr [BX], ax

pop word ptr [BP]

; mov (r2),-(sp)

; mov (r3),(r2)

; mov (sp)+,(r3)

jmp short C4

; br 2b

C7: ;2:

inc bx

inc bx

; tst (r2)+

cmp bx, dx

; cmp r2,r1

;jb short @b

;jb short C3

; blo 1b

;

jnb short C8

mov bp, bx

jmp short @b

C8: ;1:

; end of sorting

pop bp ; r1 -> r2

; mov (sp)+,r2

; mov (sp)+,r5

; BP = R2

pass3:

; DX = R1 -> 'eol:' points to end of the list

mov word ptr [eol], dx ; save dx/r1

;

cmp word ptr [ocount], 1

; cmp ocount,$1

jng short E1

; ble 1f

cmp word ptr [isadir], 0

; tst isadir

jna short E2

; beq 2f

mov si, word ptr [dnp]

; mov dnp,0f

call pstring

; jsr r5,pstring; 0:..

mov si, offset colon

; jsr r5,pstring; colon

call pstring

E1: ;1:

cmp word ptr [longf], 0

; tst longf

jna short E10

; beq 1f

mov si, offset totmes

call pstring

; jsr r5,pstring; totmes

mov ax, word ptr [tblocks]

; mov tblocks,r0

mov bx, 4

call decimal

; jsr r5,decimal; 4

mov si, offset nl

call pstring

; jsr r5,pstring; nl

jmp short @f

E2: ;2:

cmp byte ptr [longf], 0

; tstb longf

jna short E10

; beq 1f

@@:

mov bx, offset passwd

; mov $passwd,r0

mov di, offset iobuf

call fopen

; jsr r5,fopen; iobuf

jc short E10

; bes 1f

mov di, offset uidbuf

; mov $uidbuf,r3

E3: ;3:

; ?

E4: ;2:

mov si, offset iobuf

@@:

call getc

; jsr r5,getc; iobuf

jc short E9

; bes 3f

stosb

; movb r0,(r3)+

cmp al, ':'

; cmpb r0,$':

jne short E4

; bne 2b

E5: ;2:

;mov si, offset iobuf

call getc

; jsr r5,getc; iobuf

cmp al, ':'

; cmpb r0,$':

jne short E5

; bne 2b

E6: ;2:

;mov si, offset iobuf

call getc

; jsr r5,getc; iobuf

cmp al, ':'

; cmpb r0,$':

je short E7

; bne 2b

stosb

; movb r0,(r3)+

jmp short E6

; br 2b

E7: ;2:

mov al, 0Dh

stosb

; movb $'\n,(r3)+

cmp di, offset euidbuf

; cmp r3,$euidbuf

jnb short E9

; bhis 3f

E8: ;2:

;mov si, offset iobuf

call getc

; jsr r5,getc; iobuf

cmp al, 0Dh ; end of line

; cmpb r0,$'\n

jne short E8

; bne 2b

;jmp short E3

; br 3b

jmp short @b

E9: ;3:

mov word ptr [euids], di

; mov r3,euids

; Retro UNIX 8086 v1 modification !!!

mov bx, word ptr [iobuf]

; ??? (file descriptor ???)

; Original unix v1 'ls.s' has/had source

; code defect here !!!

sys \_close

; sys close

E10: ;1:

; BP = R2

; [eol] = end of the list

; (= r1 in original unix v1 'ls.s')

cmp bp, word ptr [eol]

; cmp r2,r1

ja short E14

; bhi 1f

mov si, word ptr [BP]

inc bp

inc bp

; mov (r2)+,r3

sub si, word ptr [sortoff]

; sub sortoff,r3

;; SI points to filename offset (0)

;; of the listing (source) data (20 bytes)

call pentry

; jsr r5,pentry

mov cx, 8

; mov $8.,-(sp)

;; print/write file name (on the end of

;; the listing row (after time string)

E11: ;2:

lodsb

; movb (r3)+,r0

or al, al

jz short E13

; beq 2f

push cx

;mov bx, offset obuf

call putc

; jsr r5,putc; obuf

pop cx

loop E11

; dec (sp)

; bne 2b

E13: ;2:

; tst (sp)+

mov si, offset nl ; new line

call pstring

; jsr r5,pstring; nl

jmp short E10

; br 1b

E14: ;1:

cmp word ptr [ocount], 1

; cmp ocount,$1

jng short E15

; ble 1f

cmp word ptr [isadir], 0

; tst isadir

je short E15

; beq 1f

mov si, offset nl

call pstring

; jsr r5,pstring; nl

E15: ;1:

pop si ; r5

retn

; rts pc

pentry:

;mov r2,-(sp)

cmp byte ptr [longf], 0

; tstb longf

ja short listl

; bne listl

cmp byte ptr [longf]+1, 0

; tstb longf+1

ja short @f

; bne 2f

; mov (sp)+,r2

retn

; rts r5

@@: ;2:

mov ax, word ptr [SI]+12

; mov 12.(r3),r0

call calcb

; jsr r5,calcb

push si

mov bx, 3

call decimal

; jsr r5,decimal; 3

call \_pstring

; jsr r5,pstring; space

; mov (sp)+,r2

pop si

retn

; rts r5

\_pstring:

mov si, offset space

pstring:

; mov r5,-(sp)

; mov (r5),r5

@@: ;1:

lodsb

; movb (r5)+,r0

and al, al

jz short @f

; beq 1f

;mov bx, offset obuf

call putc

; jsr r5,putc; obuf

jmp short @b

; br 1b

@@: ;1:

retn

; mov (sp)+,r5

; tst (r5)+

; rts r5

questf:

push si

mov si, bx

; mov r4,0f

call pstring

; jsr r5,pstring; 0:..

pop si

; mov r5,0f

;call pstring

; jsr r5,pstring; 0:..

;retn

jmp short pstring

;1:

; tstb (r5)+

; bne 1b

; inc r5

; bic $1,r5

; rts r5

listl:

mov ax, word ptr [SI]+18

; mov 18.(r3),r0 / inode

push si ; r3

mov bx, 4

call decimal

; jsr r5,decimal; 4

call \_pstring

; jsr r5,pstring; space

pop si ; r3

mov di, si

; mov r3,r4

add di, 8

; add $8.,r4 / get to flags

test byte ptr [DI]+1, 10h

;test word ptr [DI], 1000h

; bit $10000,(r4) /large

jz short F1

; beq 2f

mov al, 'l'

call mode

; jsr r5,mode; 'l

jmp short F2

; br 3f

F1: ;2:

mov al, 's'

call mode

; jsr r5,mode; 's

F2: ;3:

test byte ptr [DI]+1, 40h

;test word ptr [DI], 4000h

; bit $40000,(r4) /directory

jz short F3

; beq 2f

mov al, 'd'

call mode

; jsr r5,mode; 'd

jmp short F6

; br 3f

F3: ;2:

test byte ptr [DI], 20h

; bit $40,(r4) /set uid

jz short F4

; beq 2f

mov al, 'u'

call mode

; jsr r5,mode; 'u

jmp short F6

; br 3f

F4: ;2:

test byte ptr [DI], 10h

; bit $20,(r4) /executable

jz short F5

; beq 2f

mov al, 'x'

call mode

; jsr r5,mode; 'x

jmp short F6

; br 3f

F5: ;2:

call \_mode

; jsr r5,mode; '-

F6: ;3:

test byte ptr [DI], 8

; bit $10,(r4) /read owner

jz short F7

; beq 2f

mov al, 'r'

call mode

; jsr r5,mode; 'r

jmp short F8

; br 3f

F7: ;2:

call \_mode

; jsr r5, mode; '-

F8: ;3:

test byte ptr [DI], 4

; bit $4,(r4) /write owner

jz short F9

; beq 2f

mov al, 'w'

call mode

; jsr r5,mode; 'w

jmp short F10

; br 3f

F9: ;2:

call \_mode

; jsr r5,mode; '-

F10: ;3:

test byte ptr [DI], 2

; bit $2,(r4) /read non-owner

jz short F11

; beq 2f

mov al, 'r'

call mode

; jsr r5,mode; 'r

jmp short F12

; br 3f

F11: ;2:

call \_mode

; jsr r5,mode; '-

F12: ;3:

test byte ptr [DI], 1 ; (r4)

; bit $1,(r4)+ /write non-owner

jz short F13

; beq 2f

mov al, 'w'

call mode

; jsr r5,mode; 'w

jmp short F14

; br 3f

F13: ;2:

call \_mode

; jsr r5,mode; '-

F14: ;3:

push si ; r3

call \_pstring

; jsr r5,pstring; space

; inc di ;; (r4)+

; inc di ;;

mov si, di

lodsw ; (r4)+

lodsb ;; nlinks

cbw

; movb (r4)+,r0

mov bx, 2

call \_decimal

; jsr r5,decimal; 2

lodsb ;; uid

; movb (r4)+,r2

call puid

; jsr pc,puid

lodsw ;; size

; mov (r4)+,r0

mov bx, 5

call \_decimal

; jsr r5,decimal; 5

push si

call \_pstring

; jsr r5,pstring; space

pop si

; mov r1,-(sp)

mov bx, word ptr [eol] ;r1

lodsw ; mtime, LW

mov dx, ax

; mov (r4)+,r0

lodsw ; mtime, HW

xchg dx, ax ; HW:LW

; mov (r4)+,r1

; sub $16.,sp

; mov sp,r2

; DX:AX = unix time (epoch)

call ctime

; jsr pc,ctime

; mov sp,r2

mov cx, 25

;;mov cx, 15

; mov $15.,-(sp)

mov si, offset cbuf

F15: ;1:

push cx

lodsb

; movb (r2)+,r0

;mov bx, offset obuf

call putc

; jsr r5,putc; obuf

pop cx

loop F15

; dec (sp)

; bne 1b

; add $18.,sp

; mov (sp)+,r1

;call \_pstring

; jsr r5,pstring; space

; mov (sp)+,r2

pop si ; r3

retn

; rts r5

puid:

; print user name

; AL = user id/number

push si ; r3

G0:

push ax ; r2

; mov r1,-(sp)

mov si, offset uidbuf

; mov $uidbuf,r1

G1: ;1:

;cmp si, offset euids

; cmp r1,euids

;jnb short G8

; bhis 1f

push si ; 0:

; mov r1,0f

G2: ;2:

lodsb

and al, al

; tstb (r1)+

jz short G3

; beq 3f

cmp al, ':'

; cmpb -1(r1),$':

jne short G2

; bne 2b

xor al, al ; 0

mov byte ptr [SI]-1, al ;0

; clrb -1(r1)

G3: ;3:

xor bx, bx

; clr -(sp)

;mov cx, 10

; ch = 0

mov cl, 10

G4: ;3:

lodsb

; movb (r1)+,r0

sub al, '0'

; sub $'0,r0

cmp al, 9

; cmp r0,$9.

ja short G5

; bhi 3f

; mov r1,-(sp)

; 07/10/2015

cbw

xchg ax, bx

; mov 2(sp),r1

mul cx

; mpy $10.,r1

add bx, ax

; add r0,r1

; mov r1,2(sp)

; mov (sp)+,r1

jmp short G4

; br 3b

G5: ;3:

pop si ; 0:

pop ax ; r2

; mov (sp)+,r0

cmp bx, ax

; cmp r0,r2

;jne short G1

; bne 1b

je short @f

cmp bx, offset euids

jb short G0

;jb short G1

;jmp short G8

G8:

push ax ; r2/UID

call \_pstring

;jsr r5,pstring; space

pop ax

; mov r2,r0

mov bx, 3

call decimal

; jsr r5,decimal; 3

mov si, offset space3

call pstring

; jsr r5,pstring; space3

pop si ; r3

; mov (sp)+,r1

retn

; rts pc

@@:

push si ; 0:

call \_pstring

; jsr r5,pstring; space

pop si ; 0:

push si ; 0:

call pstring

; jsr r5,pstring; 0:..

pop si ; 0:

; mov 0b,r1

mov cx, 6

; mov $6,-(sp)

G6: ;3:

lodsb

; tstb (r1)+

and al, al

jz short G7

; beq 3f

dec cl

; dec (sp)

jmp short G6

; br 3b

G7: ;3:

push cx

call \_pstring

; jsr r5,pstring; space

pop cx

dec cx

; dec (sp)

jg short G7

; bgt 3b

; tst (sp)+

pop si ; r3

; mov (sp)+,r1

retn

; rts pc

;G8: ;1:

;jsr r5,pstring; space

; mov r2,r0

; jsr r5,decimal; 3

; jsr r5,pstring; space3

; mov (sp)+,r1

; rts pc

;\_mode:

; mov al, '-'

;mode:

; AL = mode char

;mov (r5)+,r0

;mov bx, offset obuf

; call putc

; jsr r5,putc; obuf

; retn

; rts r5

gstat:

push bp

; mov r1,-(sp)

add bp, 512

; add $512.,r1

cmp bp, word ptr [brk]

; cmp r1,0f

jb short D1

; blo 1f

mov word ptr [brk], bp

; mov r1,0f

sys \_break, bp ; sys \_break, brk

; sys break; 0: end+512.

D1: ;1:

pop bp

; mov (sp)+,r1

xor ax, ax

; Detailed (Long) listing

cmp word ptr [longf], ax ;0

; tst longf

ja short D2

; bne 2f

; Sorting by modification time

cmp word ptr [sortoff], ax ;0

; tst sortoff

jna short D4

; beq 1f

D2: ;2:

sys \_stat, filnam, statb

; sys stat; filnam; statb

jnc short D3

; bec 2f

; mov r4,-(sp)

;mov bx, offset filnam

; mov $filnam,r4

mov si, offset @f

call questf

; jsr r5,questf; < unstatable\n\0>; .even

; mov (sp)+,r4

D4:

add bp, 12

; add $12.,r1

retn

; rts r5

@@:

db ' unstatable', 0Dh, 0Ah, 0

D3: ;2:

push di

mov di, bp

mov si, offset statb + 2

; mov $statb+2,r0

movsw

; mov (r0)+,(r1)+ /flags

movsw

; mov (r0)+,(r1)+ /nlinks, uid

; mov r0,-(sp)

mov ax, word ptr [SI]

; mov (r0),r0

call calcb

; jsr r5,calcb

add word ptr [tblocks], ax

; add r0,tblocks

; mov (sp)+,r0

movsw

; mov (r0)+,(r1)+ /size

add si, 20

; add $20.,r0 /dska, ctim

movsw

; mov (r0)+,(r1)+ /mtim

movsw

; mov (r0)+,(r1)+

mov ax, word ptr [statb]

stosw

; mov statb,(r1)+ /inode

mov bp, di

pop di

retn

; rts r5

;D4: ;1:

; add bp, 12

; add $12.,r1

; retn

; rts r5

\_decimal:

push si

call decimal

pop si

retn

decimal:

; convert number to decimal number chars

; AX = number to be converted

; BX = number of digits (=4)

; mov r1,-(sp)

; mov r2,-(sp)

; mov r3,-(sp)

;push di

xor dx, dx

mov cx, 6

; mov $6,r2

mov di, offset numbuf + 6

; mov $numbuf+6,r3

mov si, 10

@@: ;1:

;and ax, ax

;jz short @f

;mov r0,r1

;xor dx, dx

; clr r0

;mov si, 10

; dvd $10.,r0

div si

;@@:

add dl, '0'

; add $'0,r1

dec di

mov byte ptr [DI], dl

; movb r1,-(r3)

xor dl, dl

loop @b

; sob r2,1b

mov al, 20h ; space

mov cl, 5

@@: ;1:

;cmp di, offset numbuf + 5

; cmp r3,$numbuf+5

;je short @f

; beq 1f

cmp byte ptr [DI], '0'

; cmpb (r3),$'0

jne short @f

; bne 1f

;mov al, 20h

stosb

; movb $' ,(r3)+

;jmp short @b

; br 1b

loop @b

@@: ;1:

mov si, offset numbuf + 6

; mov $numbuf+6,r1

sub si, bx

; sub (r5),r1

;mov cx, bx

mov cl, bl ; ch = 0, bh = 0

; mov (r5)+,-(sp)

@@: ;1:

push cx

lodsb

; movb (r1)+,r0

;mov bx, offset obuf

call putc

; jsr r5,putc; obuf

pop cx

loop @b

; dec (sp)

; bne 1b

; tst (sp)+

; mov (sp)+,r3

; mov (sp)+,r2

; mov (sp)+,r1

;pop di

retn

; rts r5

calcb:

; calculate number of blocks

add ax, 511

; add $511.,r0

sub al, al

; clrb r0

xchg ah, al

; swab r0

; al= (size+511)/256

shr al, 1 ; ah = 0

; asr r0

; al = (size+511)/512

; large file ? (>=4096 bytes)

cmp al, 8

; cmp r0,$8

jb short @f

; blo 1f

; add indirect block

inc al

; inc r0

@@: ;1:

;1: ; ?

retn

; rts r5

\_mode:

mov al, '-'

mode:

; AL = mode char

;mov (r5)+,r0

;mov bx, offset obuf

; call putc

; jsr r5,putc; obuf

; retn

; rts r5

; 'putc' procedure

; is derived from 'put.s'

; file of original UNIX v5

;

; write characters on output file

putc:

; AL = character to be written

; obuf = output buffer

;; BX = buffer address

push si

;mov r1,-(sp)

mov si, offset obuf

;mov si, bx

;mov (r5)+,r1

@@: ;1:

dec word ptr [SI]+2

; dec 2(r1)

jns short @f

; bge 1f

push ax

; mov r0,-(sp)

call fl

; jsr pc,fl

pop ax

; mov (sp)+,r0

jmp short @b

; br 1b

@@: ;1:

mov bx, word ptr [SI]+4

mov byte ptr [BX], al

; movb r0,\*4(r1)

inc word ptr [SI]+4

; inc 4(r1)

pop si

; mov (sp)+,r1

retn

; rts r5

; 'flush' procedure

; is derived from 'put.s'

; file of original UNIX v5

flush:

; mov r0,-(sp)

; mov r1,-(sp)

; mov (r5)+,r1

; jsr pc,fl

; mov (sp)+,r1

; mov (sp)+,r0

; rts r5

fl:

mov cx, si

; mov r1,r0

add cx, 6

; add $6,r0

;push cx ; Buffer data address

; mov r0,-(sp)

; mov r0,0f

mov dx, word ptr [SI]+4 ; Buffer offset

; mov 4(r1),0f+2

or dx, dx

jz short @f

; beq 1f

sub dx, cx ; Byte count

; sub (sp),0f+2

mov bx, word ptr [SI] ; File descriptor (=1)

; mov (r1),r0

sys \_write ; sys \_write, bx, cx, dx

; sys 0; 9f

;.data

;9:

; ; sys write; 0:..; ..

;.text

@@: ;1:

;pop cx

mov word ptr [SI]+4, cx ; Begin. of buf. data

; mov (sp)+,4(r1)

mov word ptr [SI]+2, 512 ; Buffer data size

; mov $512.,2(r1)

retn

; rts pc

; 'getw', 'getc' and 'fopen' procedures

; are derived from 'get.s'

; file of original UNIX v5

; open a file for use by get(c|w)

fopen:

; bx = file name offset

; di = buffer offset

;

xor cx, cx ; 0 => open for read

sys \_open ; sys \_open, bx, cx (0)

jc short @f

stosw ; file descriptor (in buffer offset 0)

retn

@@:

mov ax, 0FFFFh ; -1

stosw

;stc

retn

; get words from input file

getw:

;mov si, bx

call getc

jc short @f

push ax

call getc

pop dx

mov ah, dl

xchg ah, al

@@:

retn

; get characters from input file

getc:

; SI = buffer address

mov ax, word ptr [SI]+2 ; char count

and ax, ax

jnz short gch1

gch0:

mov cx, si

add cx, 6 ; read buff. addr.

mov bx, word ptr [SI]

mov word ptr [SI]+4, cx ; char offset

;xor ax, ax

;mov word ptr [SI]+2, ax ; 0

mov dx, 512

sys \_read ; sys \_read, bx, cx, dx

jc short gch2

or ax, ax

jz short gch3

gch1:

dec ax

mov word ptr [SI]+2, ax

mov bx, word ptr [SI]+4

mov al, byte ptr [BX]

inc bx

mov word ptr [SI]+4, bx

xor ah, ah

retn

gch2:

xor ax, ax

gch3:

stc

retn

;/ getw/getc -- get words/characters from input file

;/ fopen -- open a file for use by get(c|w)

;/

;/ calling sequences --

;/

;/ mov $filename,r0

;/ jsr r5,fopen; ioptr

;/

;/ on return ioptr buffer is set up or error bit is set if

;/ file could not be opened.

;/

;/ jsr r5,get(c|w)1; ioptr

;/

;/ on return char/word is in r0; error bit is

;/ set on error or end of file.

;/

;/ ioptr is the address of a 518-byte buffer

;/ whose layout is as follows:

;/

;/ ioptr: .=.+2 / file descriptor

; .=.+2 /// buffer size (This is noted by Erdogan Tan; 19/11/2013)

;/ .=.+2 / charact+2 / pointer to next character (reset if no. chars=0)

;/ .=.+512. / the buffer

; .globl getc,getw,fopen

;fopen:

; mov r1,-(sp)

; mov (r5)+,r1

; mov r0,0f

; sys 0; 9f

;.data

;9:

; sys open; 0:..; 0

;.text

; bes 1f

; mov r0,(r1)+

; clr (r1)+

; mov (sp)+,r1

; rts r5

;1:

; mov $-1,(r1)

; mov (sp)+,r1

; sec

; rts r5

;

;.data

;getw:

; mov (r5),9f

; mov (r5)+,8f

; jsr r5,getc; 8:..

; bec 1f

; rts r5

;1:

; mov r0,-(sp)

; jsr r5,getc; 9:..

; swab r0

; bis (sp)+,r0

; rts r5

;.text

;

;getc:

; mov r1,-(sp)

; mov (r5)+,r1

; dec 2(r1)

; bge 1f

; mov r1,r0

; add $6,r0

; mov r0,0f

; mov r0,4(r1)

; mov (r1),r0

; sys 0; 9f

;.data

;9:

; sys read; 0:..; 512.

;.text

; bes 2f

; tst r0

; bne 3f

;2:

; mov (sp)+,r1

; sec

; rts r5

;3:

; dec r0

; mov r0,2(r1)

;1:

; clr r0

; bisb \*4(r1),r0

; inc 4(r1)

; mov (sp)+,r1

; rts r5

include ctime.inc ; 24/11/2013

dw 417

brk: dw offset \_end + 512 ; (gstat:)

dnp: dw 0 ; (do:)

dotp: dw offset dot

;dotp: dot

euids: dw offset uidbuf

; euids: uidbuf

dot: db '.', 0

;dot: <.\0>

nl: db 0Dh, 0Ah, 0

; nl: <\n\0>

totmes: db 'total ', 0

; totmes: <total \0>

space3: db 20h, 20h, 20h

; space3: < >

space: db 20h, 0

; space: < \0>

passwd: db '/etc/passwd', 0

; passwd: </etc/passwd\0>

colon: db ':', 0Dh, 0Ah, 0

; colon: <:\n\0>

eol: dw 0 ; (pass3:)

EVEN

bss:

count: dw 0

ocount: dw 0

longf: dw 0

sortoff: dw 0

allflg: dw 0

dirflg: dw 0

isadir: dw 0

filnam: db 32 dup(0)

statb: db 34 dup(0)

dbuf: db 518 dup(0)

obuf: db 518 dup(0)

numbuf: db 6 dup(0)

tblocks: dw 0

uidbuf: db 1024 dup(0)

euidbuf:

iobuf: db 518 dup(0)

\_end:

; .even

;.bss

;count: .=.+2

;ocount: .=.+2

;longf: .=.+2

;sortoff: .=.+2

;allflg: .=.+2

;dirflg: .=.+2

;isadir: .=.+2

;filnam: .=.+32.

;statb: .=.+34.

;dbuf: .=.+518.

;obuf: .=.+518.

;numbuf: .=.+6

;tblocks: .=.+2

;uidbuf: .=.+1024.

;euidbuf:

;iobuf: .=.+518.

UNIX ends

end START\_CODE