

```
. *****
;
; ls386.s (ls1.s) - Retro Unix 386 v1.1 - /bin/ls - list file or directory
; -----
; RETRO UNIX 386 (Retro Unix == Turkish Rational Unix)
; Operating System Project (v0.2) by ERDOGAN TAN (Beginning: 24/12/2013)
;
; [ Last Modification: 04/12/2015 ]
;
; Derived from 'ls0.asm' source code file of 'Retro UNIX 8086 v1'
; operating system project, /bin/ls source code by Erdogan Tan
; (19/11/2013-29/11/2013)
;
; Derived from 'ls.s' file of original UNIX operating system
; (v1.0 for PDP-11)
; *****
; ls1.s (04/12/2015), Retro UNIX 386 v1.1 (14 byte file names)
; ls0.s (07/10/2015), Retro UNIX 386 v1 (8 byte file names)
; ls386.s (23/09/2015, Retro UNIX 386 v1, NASM 2.11, 32 bit version)
; LS0.ASM, 19/11/2013 - 29/11/2013 (Retro UNIX 8086 v1, MASM 6.11)

; Assembler: NASM 2.11

; 06/10/2015
; 05/10/2015
; 23/09/2015

; UNIX v1 system calls
_rele    equ 0
_exit    equ 1
_fork    equ 2
_read    equ 3
_write   equ 4
_open    equ 5
_close   equ 6
_wait    equ 7
_creat   equ 8
_link    equ 9
_unlink  equ 10
_exec    equ 11
_chdir   equ 12
_time    equ 13
_mkdir   equ 14
_chmod   equ 15
_chown   equ 16
_break   equ 17
_stat    equ 18
_seek    equ 19
_tell    equ 20
_mount   equ 21
_umount  equ 22
_setuid  equ 23
_getuid  equ 24
_stime   equ 25
_quit    equ 26
_intr    equ 27
_fstat   equ 28
_emt     equ 29
_mdate   equ 30
_stty    equ 31
_gtty    equ 32
_ilgins  equ 33
_sleep   equ 34 ; Retro UNIX 8086 v1 feature only !
_msg     equ 35 ; Retro UNIX 386 v1 feature only !
_geterr  equ 36 ; Retro UNIX 386 v1 feature only !
```

```
%macro sys 1-4
; 03/09/2015
; 13/04/2015
; Retro UNIX 386 v1 system call.
%if %0 >= 2
    mov ebx, %2
    %if %0 >= 3
        mov ecx, %3
        %if %0 = 4
            mov edx, %4
        %endif
    %endif
    mov eax, %1
    int 30h
%endmacro

; Retro UNIX 386 v1 system call format:
; sys syscall (eax) <arg1 (ebx)>, <arg2 (ecx)>, <arg3 (edx)>
```

[BITS 32] ; We need 32-bit instructions for protected mode

[ORG 0]

START_CODE: ; / ls -- list file or directory

```

; .globl flush
; .globl fopen
; .globl getw
; .globl getc
; .globl putc
; .globl ctime
; .globl end

; mov    eax, _end + 512
; and    al, 0FEh
; cmp    eax, esp
; jna    short S1
; mov    esp, eax
; S1:
; mov    ebx, eax

; 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).

; clear bss area
; (It is not necessary to clear bss area
; because Retro UNIX 386 v1 kernel will clear
; this bss area during memory page allocation.)
; mov    ecx, _end
; mov    edi, bss_start
; sub    ecx, edi
; shr    cx, 2
; sub    eax, eax
; rep    stosd
```

```

; sys break
; clears memory from 'bss' to 'bss._end+512'
mov     ebx, _end + 512
sys     _break
        ; sys break; end+512.
sys     _write, 1, nl, 2
;
mov     [obuf], bx ; 1
        ; mov $1,obuf
mov     esi, esp
        ; mov sp,r5

lodsd
dec     eax
mov     [count], ax
        ; mov (r5)+,count
        ; tst (r5)+
        ; dec count
mov     [ocount], ax
        ; mov count,ocount
        ; bgt loop
        ; mov $dotp,r5
jna     short B0
;and    eax, eax
;jnz    short S2
;jz     short B0
;mov    esi, dotp
;jmp    short sloop
;S2:
lodsd
sloop:  ;loop:
lodsd
mov     ebx, eax
        ;mov (r5)+,r4
cmp     byte [ebx], '-'
        ; cmpb (r4)+,$'-
jne     short A1
        ; bne 1f
inc     ebx
dec     word [ocount]
        ; dec ocount
A3: ;3:
mov     al, [ebx]
        ; movb (r4)+,r0
inc     ebx
or      al, al
jz      short eloop
        ; beq eloop
cmp     al, 'l'
        ; cmp r0,$'l'
jne     short S3
        ; bne 4f
inc     word [longf]
        ; inc longf
jmp     short A3
        ; br 3b
S3: ;4:
cmp     al, 't'
        ; cmpb r0,$'t'
jne     short S4
        ; bne 4f
mov     dword [sortoff], 14
        ; mov $14.,sortoff
jmp     short A3
        ; br 3b

```

```

S4: ;4:
    cmp    al, 'a'
           ; cmpb r0,$'a'
    jne    short S5
           ; bne 4f
    inc    word [allflg]
           ; inc allflg
    jmp    short A3

S5: ;4:
    cmp    al, 's'
           ; cmpb r0,$'s'
    jne    short S6
           ; bne 4f
    inc    byte [longf+1]
           ; incb longf+1
    jmp    short A3
           ; br 3b

S6: ;4:
    cmp    al, 'd'
           ; cmpb r0,$'d'
    jne    short A3
           ; bne 3b
    inc    word [dirflg]
           ; inc dirflg
    jmp    short A3
           ; br 3b

A1: ;1:
    ;dec    ebx
           ; dec r4
    call   do
           ; jsr pc,do

eloop:
    dec    word [count]
           ; dec count
    jg     short sloop
           ; bgt loop
    mov    eax, [dnp]
    and    eax, eax
           ; tst dnp
    jnz    short S7
           ; bne 1f

B0:
    mov    esi, dotp
           ; mov $dotp,r5
    jmp    short sloop
           ; br loop

S7: ;1:
    mov    esi, obuf
    call   flush
           ; jsr r5,flush; obuf
    sys    _exit
           ; sys exit

```

```

; 04/12/2015 (14 byte file names)
;; 26 bytes listing (source) data
;; structure:
;; offset
;; 0-13 : file name
;; 14-15 : flags
;; 16-17 : nlinks, uid
;; 18-19 : size
;; 20-21-22-23 : mtime
;; 24-25 : inode number

do:
    push    esi ; r5
    sub     eax, eax
    mov     [tblocks], eax ; 0
           ; clr tblocks
    mov     ebp, _end
           ; mov $end,r1
    mov     edi, filnam
           ; mov $filnam,r3
    mov     [dnp], ebx
           ; mov r4,dnp
    mov     esi, ebx ; r4
    mov     [isadir], ax ; 0
           ; clr isadir
    cmp     [dirflg], ax ; 0
           ; tst dirflg
    ja      nondir
           ; bne nondir
    ;mov     ebx, [dnp]
    mov     ecx, statb
    sys     _stat
           ; sys stat; dnp: 0; statb
    jnc     short B1
           ; bec 1f
    ; EBX = file name address
    mov     esi, S8

do_err:
    call    questf
    pop     esi
    retn

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

S8:
    db      ' nonexistent', 0Dh, 0Ah, 0

B1: ;1:
    ;test    word [statb+2], 4000h
    test    byte [statb+3], 40h
           ; bit $40000,statb+2 /test directory
    jz      nondir
           ; beq nondir
    inc     word [isadir]
           ; inc isadir
    ;mov     eax, ebx
           ; mov r4,r0
    push    edi
    mov     edi, dbuf
    call    fopen
           ; jsr r5,fopen; dbuf
    pop     edi
    jnc     short B2
           ; bcc 1f
    ; EBX = file name address
    mov     esi, S9

```

```

        jmp     short do_err
        ;call   questf
        ;pop    esi
        ;retn

        ; jsr r5,questf; < unreadable\n\0>; .even
        ; rts pc
S9:
        db      ' unreadable', 0Dh, 0Ah, 0
B2:
        ; mov   esi, ebx ; r4
S10: ;1:
        lodsb
        stosb
        ; movb (r4)+,(r3)+
        or      al, al
        jnz     short S10
        ; bne 1b
        dec     edi
        ; dec r3
        ;
        cmp     byte [edi-1], '/'
        ; cmpb -1(r3), $'/
        je      short B3
        ; beq 1f
        mov     al, '/'
        stosb
        ; movb $'/',(r3)+
B3: ;1:
        ; mov   ebx, dbuf
        mov     esi, dbuf
S11:
        call    getw
        ; jsr r5,getw; dbuf
        jc      short pass2
        ; bcs pass2
        mov     ecx, 7 ; 04/12/2015 (14 byte file names)
        ; mov $4,-(sp)
        and     ax, ax
        ; tst r0
        jnz     short B5
        ; bne 2f
B4: ;3:
        push    ecx
        ; mov   esi, dbuf
        call    getw
        ; jsr r5,getw; dbuf
        pop     ecx
        loop    B4
        ; dec (sp)
        ; bne 3b
        ; tst (sp)+
        jmp     short S11
        ; jmp   short B3
        ; br 1b
B5: ;2:
        ; EDI == r2
        ; mov r3,r2
        push    edi ; r3 (filnam +'/' +1)

```

B6: ;2:

```

;; copying file name
;; to listing (source) data address (ebp)
;; (offset 0-13) ; 04/12/2015 (14 byte file names)
;; and filnam (edi)
;
push    ecx
; mov   esi, dbuf
call    getw
        ; jsr r5,getw; dbuf
mov     [ebp], ax
inc     ebp
inc     ebp
        ; mov r0,(r1)+

stosw
;stosb

        ; movb r0,(r2)+
;xchg   al, ah
        ; swab r0
;stosb

        ; movb r0,(r2)+
pop     ecx
loop    B6
        ; dec (sp)
        ; bne 2b
        ; tst (sp)+
xor     eax, eax ; 0
stosb

        ; clrb (r2)+
pop     edi ; r3
cmp     [allflg], ax ; 0
        ; tst allflg
ja      short B7
        ; bne 2f
cmp     byte [edi], '.'
        ; cmpb (r3),$.
jne     short B7
        ; bne 2f
sub     ebp, 14 ; 04/12/2015 (14 byte file names)
        ; sub $8.,r1
jmp     short S11
;jmp    short B3
        ; br 1b

```

B7: ;2:

```

;; copying 12 bytes inode data to
;; listing (source) data from offset
;; 14 to offset 25 (of 26 data bytes)
;
call    gstat
        ; jsr r5,gstat
jmp     short B3
        ; br 1b

```

nondir:

```

; mov   esi, ebx ; r4
mov     ebx, edi ; offset filnam
;mov    r3,r2

```

S12: ;1:

```

; ESI points to file name (input)
lodsrb
stosrb

        ; movb (r4)+,(r2)+
and     al, al
jnz     short S12
        ; bne 1b

```

```

S13: ;1:
    cmp     edi, ebx ; offset filnam
           ; cmp r2,r3
    jna     short S14
           ; blos 1f
    dec     edi
    cmp     byte [edi], '/'
           ; cmpb -(r2),$/
    jne     short S13
           ; bne 1b
    inc     edi
           ; inc r2
           ;; EDI points to last name
           ;; of the path (after "/" )

S14: ;1:
    mov     ecx, 14 ; 04/12/2015 (14 byte file names)
           ; mov $8,-(sp)

ndloop: ;1:
    mov     al, [edi]
    mov     [ebp], al
    inc     ebp
           ; movb (r2)+,(r1)+
           ; bne 2f
           ; dec r2
    or      al, al
    jz      short S15
    inc     edi

S15:
    loop    ndloop
    call    gstat ; fill/get 12 bytes listing data
    jmp     short pass2

S16: ;2:
           ; dec (sp)
           ; bne 1b
           ; jsr r5,gstat
           ; tst (sp)+

pass2:
    movzx   ebx, word [dbuf]
           ; mov dbuf,r0
    sys     _close
           ; sys close
    mov     ecx, ebx ; file descriptor
    mov     ebx, _end
           ; mov $end,r2
    cmp     ebp, ebx ; ebp >= _end (= last word + 4)
           ; cmp r1,r2
    jne     short C1
           ; bne 1f
    pop     esi ; r5
    retn
           ; rts pc

C1: ;1:
    ; sorting begins here
           ; mov r5,-(sp)
    mov     edi, ebp ; current end of listing words (+4)
    push    ebp ; r1
           ; mov r1,-(sp)
           ; EBX will point to mtime or file name (+14 or 0)
           ; offset of 26 bytes listing (source) data
    add     ebx, [sortoff]
           ; add sortoff,r2

```



```

C2: ;1:
    mov     eax, ebx
    stosd
        ; mov r2,(r1)+
    ; 04/12/2015 (20 -> 26)
    add     ebx, 26 ; EBX now points to next 26 bytes
        ; add $20.,r2
    cmp     ebx, ebp ; is EBX passed the data limit ?
        ; cmp r2,(sp)
    jb      short C2
        ; blo 1b

S17:
    mov     ebx, ebp
        ; mov (sp),r2
    sub     edi, 4
        ; tst -(r1)

C3: ;1:
    mov     edx, edi ; r1

S18:
    ;mov     ebp, ebx
    ;mov     r2,r3

C4: ;2:
    add     ebp, 4
        ; tst (r3)+
    cmp     ebp, edx
        ; cmp r3,r1
    ja      short C7
        ; bhi 2f
    mov     esi, [ebx] ; file name 1 or time 1
        ; mov (r2),r4
    mov     edi, [ebp] ; file name 2 or time 2
        ; mov (r3),r5
    cmp     dword [sortoff], 0
        ; tst sortoff
    jna      short C5
        ; beq 4f

; sorting by modification time
    cmpsd
    jb      short C6
        ; cmp (r4)+,(r5)+
        ; blo 3f
        ; bhi 2b
        ; cmp (r4)+,(r5)+
        ; blo 3f
    jmp     short C4
        ; br 2b

; sorting by file name
C5: ;4:
    ; ?
    ;; mov     ecx, 14 ; 04/12/2015 (14 byte file names)
;C5x: ;4:
    cmpsb
        ; cmpb (r4)+,(r5)+
    ja      short C6
        ; bhi 3f
    jb      short C4
        ; blo 2b
    ;dec     ecx ; ?
        ; dec r0
    ;jnz short C5x ; ?
    ;jmp     short C5x
    jmp     short C5
        ; br 4b

```

```

C6: ;3:
    ; 05/10/2015
    mov     eax, [ebp]
    xchg    eax, [ebx]
    mov     [ebp], eax
    ; mov (r2), -(sp)
    ; mov (r3), (r2)
    ; mov (sp)+, (r3)
    jmp     short C4
    ; br 2b

C7: ;2:
    add     ebx, 4
    ; tst (r2)+
    cmp     ebx, edx
    ; cmp r2, r1
    ;jb     short S18
    ;jb     short C3
    ;blo 1b
    ;
    jnb     short C8
    mov     ebp, ebx
    jmp     short S18

C8: ;1:
; end of sorting
    pop     ebp ; r1 -> r2
    ; mov (sp)+, r2
    ; mov (sp)+, r5
    ; BP = R2

pass3:
    ; DX = R1 -> 'eol:' points to end of the list
    mov     [eol], edx ; save dx/r1
    ;
    cmp     word [ocount], 1
    ; cmp ocount, $1
    jng     short E1
    ; ble 1f
    cmp     word [isadir], 0
    ; tst isadir
    jna     short E2
    ; beq 2f
    mov     esi, [dnp]
    ; mov dnp, 0f
    call    pstring
    ; jsr r5, pstring; 0:...
    mov     esi, colon
    ; jsr r5, pstring; colon
    call    pstring

E1: ;1:
    cmp     word [longf], 0
    ; tst longf
    jna     E10
    ; beq 1f
    mov     esi, totmes
    call    pstring
    ; jsr r5, pstring; totmes
    mov     eax, [tblocks]
    ; mov tblocks, r0
    mov     ebx, 4
    call    decimal
    ; jsr r5, decimal; 4
    mov     esi, nl
    call    pstring
    ; jsr r5, pstring; nl
    jmp     short S19

```

```

E2: ;2:
    cmp     byte [longf], 0
           ; tstb longf
    jna     short E10
           ; beq 1f

S19:
    mov     ebx, passwd
           ; mov $passwd,r0
    mov     edi, iobuf
    call    fopen
           ; jsr r5,fopen; iobuf
    jc      short E10
           ; bes 1f
    mov     edi, uidbuf
           ; mov $uidbuf,r3

E3: ;3:
    ; ?

E4: ;2:
    mov     esi, iobuf

S20:
    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     esi, iobuf
    call    getc
           ; jsr r5,getc; iobuf
    cmp     al, ':'
           ; cmpb r0,$':
    jne     short E5
           ; bne 2b

E6: ;2:
    ;mov     esi, 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 $"n,(r3)+
    cmp     edi, euidbuf
           ; cmp r3,$euidbuf
    jnb     short E9
           ; bhis 3f

```

```

E8: ;2:
    ;mov     esi, 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 S20

E9: ;3:
    mov      [euids], edi
           ; mov r3,euids
           ; 07/10/2015 (32 bit modification)
           ; Retro UNIX 8086 v1 modification !!!
    movzx    ebx, word [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      ebp, [eol]
           ; cmp r2,r1
    ja       short E14
           ; bhi 1f
    mov      esi, [ebp]
    add      ebp, 4
           ; mov (r2)+,r3
    sub      esi, [sortoff]
           ; sub sortoff,r3

    ;;
    ;; ESI points to filename offset (0)
    ;; of the listing (source) data (26 bytes)
    ;
    call     pentry
           ; jsr r5,pentry
    ;
    mov      ecx, 14 ; 04/12/2015 (14 byte file names)
           ; 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     ecx
    ;mov     ebx, obuf
    call     putc
           ; jsr r5,putc; obuf
    pop      ecx
    loop     E11
           ; dec (sp)
           ; bne 2b

```

```

E13: ;2:
        ; tst (sp)+
        mov     esi, nl ; new line
        call    pstring
        ; jsr r5,pstring; nl
        jmp     short E10
        ; br 1b

E14: ;1:
        cmp     word [ocount], 1
        ; cmp ocount,$1
        jng     short E15
        ; ble 1f
        cmp     word [isadir], 0
        ; tst isadir
        je      short E15
        ; beq 1f
        mov     esi, nl
        call    pstring
        ; jsr r5,pstring; nl

E15: ;1:
        pop     esi      ; r5
        retn

        ; rts pc

pentry:
        ;mov r2,-(sp)
        cmp     byte [longf], 0
        ; tstb longf
        ja      short listl
        ; bne listl
        cmp     byte [longf+1], 0
        ; tstb longf+1
        ja      short S21
        ; bne 2f
        ; mov (sp)+,r2

        retn

        ; rts r5

S21: ;2:
        movzx   eax, word [esi+18] ; 04/12/2015 (+12 -> +18)
        ; mov 12.(r3),r0
        call    calcb
        ; jsr r5,calcb
        push    esi
        mov     ebx, 3
        call    decimal
        ; jsr r5,decimal; 3
        call    _pstring
        ; jsr r5,pstring; space
        ; mov (sp)+,r2
        pop     esi
        retn

        ; rts r5

```

```

_pstring:
    mov     esi, space

pstring:
    ; mov r5,-(sp)
    ; mov (r5),r5

S22: ;1:
    lodsb
    ; movb (r5)+,r0
    and     al, al
    jz      short S23
    ; beq 1f
    ;mov     ebx, obuf
    call    putc
    ; jsr r5,putc; obuf
    jmp     short S22
    ; br 1b

S23: ;1:
    retn
    ; mov (sp)+,r5
    ; tst (r5)+
    ; rts r5

questf:
    push    esi
    mov     esi, ebx
    ; mov r4,0f
    call    pstring
    ; jsr r5,pstring; 0...
    pop     esi
    ; 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:
    ; 06/10/2015 (32 bit modifications)
    movzx   eax, word [esi+24] ; 04/12/2015 (+18 -> +24)
    ; mov 18.(r3),r0 / inode
    push    esi ; r3
    mov     ebx, 4
    call    decimal
    ; jsr r5,decimal; 4
    call    _pstring
    ; jsr r5,pstring; space

    pop     esi ; r3
    mov     edi, esi
    ; mov r3,r4
    add     edi, 14 ; 04/12/2015 (8 -> 14)
    ; add $8.,r4 / get to flags
    test    byte [edi+1], 10h
    ;test    word [edi], 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 [edi+1], 40h
        ;test    word [edi], 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 [edi], 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 [edi], 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 [edi], 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 [edi], 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 [edi], 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 [edi], 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    esi ; r3
    call    _pstring
           ; jsr r5,pstring; space
    ; inc   edi, 4 ;; (r4)+
    ; inc   edi ;;
    mov     esi, edi
    lodsw   ; (r4)+
    lodsb   ;; nlinks
    movzx   eax, al
           ; movb (r4)+,r0
    mov     ebx, 2
    call    _decimal
           ; jsr r5,decimal; 2
    lodsb   ;; uid
           ; movb (r4)+,r2
    call    puid
           ; jsr pc,puid
    lodsw   ;; size
           ; mov (r4)+,r0
    ;movzx  eax, ax
    mov     ebx, 5
    call    _decimal
           ; jsr r5,decimal; 5
    push    esi
    call    _pstring
           ; jsr r5,pstring; space
    pop     esi
           ; mov r1,-(sp)
    mov     ebx, [eol] ;r1
    lodsd   ; mtime
           ; mov (r4)+,r0
           ; mov (r4)+,r1
           ; sub $16.,sp
           ; mov sp,r2

```



```

; EAX = unix time (epoch)
call    ctime
; jsr pc,ctime
; mov sp,r2
mov     ecx, 25
;;mov   ecx, 15
; mov $15,-(sp)
mov     esi, cbuf
F15: ;1:
push    ecx
lodsb
; movb (r2)+,r0
;mov    ebx, obuf
call    putc
; jsr r5,putc; obuf
pop     ecx
loop    F15
; dec (sp)
; bne 1b
; add $18.,sp
; mov (sp)+,r1
;call   _pstring
; jsr r5,pstring; space
; mov (sp)+,r2
pop     esi ; r3
retn
; rts r5

puid:
; 06/10/2015 (32 bit modifications)
; print user name
; AL = user id/number
push    esi ; r3
G0:
push    eax ; r2
; mov r1,-(sp)
mov     esi, uidbuf
; mov $uidbuf,r1
G1: ;1:
;cmp    esi, euids
; cmp r1,euids
;jnb    short G8
; bhis 1f
push    esi ; 0:
; mov r1,0f
G2: ;2:
lodsb
and     al, al
; tstb (r1)+
jz      short G3
; beq 3f
cmp     al, ':'
; cmpb -1(r1),$:
jne     short G2
; bne 2b
xor     al, al ; 0
mov     [esi-1], al ;0
; clrb -1(r1)
G3: ;3:
xor     ebx, ebx
; clr -(sp)
;mov    ecx, 10
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
movzx  eax, al
xchg   eax, ebx
; mov 2(sp),r1
mul    ecx
; mpy $10.,r1
add    ebx, eax
; add r0,r1
; mov r1,2(sp)
; mov (sp)+,r1
jmp    short G4
; br 3b
G5: ;3:
pop    esi ; 0:
pop    eax ; r2
; mov (sp)+,r0
cmp    ebx, eax
; cmp r0,r2
;jne   short G1
; bne 1b
je     short S24
cmp    ebx, euid$
jb     short G0
;jb    short G1
;jmp   short G8
G8:
push   eax ; r2/UID
call   _pstring
;jsr r5,pstring; space
pop    eax
; mov r2,r0
mov    ebx, 3
call   decimal
; jsr r5,decimal; 3
mov    esi, space3
call   pstring
; jsr r5,pstring; space3
pop    esi ; r3
; mov (sp)+,r1
retn
; rts pc
S24:
push   esi ; 0:
call   _pstring
; jsr r5,pstring; space
pop    esi ; 0:
push   esi ; 0:
call   pstring
; jsr r5,pstring; 0:...
pop    esi ; 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    esi ; 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    ebx, obuf
    ;    call    putc
        ; jsr r5,putc; obuf
    ;    retn
        ; rts r5

gstat:
    ; 06/10/2015 (32 bit modifications)
    push   ebp
        ; mov r1,-(sp)
    add    ebp, 512
        ; add $512.,r1
    cmp    ebp, [brk]
        ; cmp r1,0f
    jb     short D1
        ; blo 1f
    mov    [brk], ebp
        ; mov r1,0f
    sys    _break, ebp ; sys _break, brk
        ; sys break; 0: end+512.

```

```

D1: ;1:
    pop     ebp
        ; mov (sp)+,r1
    xor     eax, eax
    ; Detailed (Long) listing
    cmp     [longf], ax ;0
        ; tst longf
    ja      short D2
        ; bne 2f
    ; Sorting by modification time
    cmp     [sortoff], eax ;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    ebx, filnam
        ; mov $filnam,r4
    mov     esi, S25
    call    questf
        ; jsr r5,questf; < unstatable\n\0>; .even
        ; mov (sp)+,r4

D4:
    add     ebp, 12
        ; add $12.,r1
    retn
        ; rts r5

S25:
    db      ' unstatable', 0Dh, 0Ah, 0

D3: ;2:
    push    edi
    mov     edi, ebp
    mov     esi, statb + 2
        ; mov $statb+2,r0
    movsd
        ; mov (r0)+,(r1)+ /flags
        ; mov (r0)+,(r1)+ /nlinks, uid
        ; mov r0,-(sp)
    movzx   eax, word [esi]
        ; mov (r0),r0
    call    calcb
        ; jsr r5,calcb
    add     [tblocks], eax
        ; add r0,tblocks
        ; mov (sp)+,r0
    movsw
        ; mov (r0)+,(r1)+ /size
    add     esi, 20
        ; add $20.,r0      /dsk, ctim
    movsd
        ; mov (r0)+,(r1)+ /mtim
        ; mov (r0)+,(r1)+
    mov     ax, [statb]
    stosw
        ; mov statb,(r1)+ /inode
    mov     ebp, edi
    pop     edi
    retn
        ; rts r5

```

```

;D4: ;1:
;      add     ebp, 12
;      ; add $12.,r1
;      retn
;      ; rts      r5

_decimal:
    push     esi
    call     decimal
    pop      esi
    retn

decimal:
; convert number to decimal number chars
; 5/10/2015 (32 bit modifications)
; EAX = number to be converted
; EBX = number of digits (=4)
;      ; mov r1,-(sp)
;      ; mov r2,-(sp)
;      ; mov r3,-(sp)
;push     edi
xor     edx, edx
mov     ecx, 6
;      ; mov $6,r2
mov     edi, numbuf + 6
;      ; mov $numbuf+6,r3
mov     esi, 10

S26: ;1:
;and     eax, eax
;jz      short S27
;      ;mov r0,r1
;xor     edx, edx
;      ; clr r0
;mov     esi, 10
;      ; dvd $10.,r0
div     esi

;S27:
add     dl, '0'
;      ; add $'0',r1
dec     edi
mov     [edi], dl
;      ; movb r1,-(r3)
xor     dl, dl
loop    S26
;      ; sob r2,1b
mov     al, 20h ; space
mov     cl, 5

S28: ;1:
;cmp     edi, numbuf + 5
;      ; cmp r3,$numbuf+5
;je      short S29
;      ; beq 1f
cmp     byte [edi], '0'
;      ; cmpb (r3),$'0'
jne     short S29
;      ; bne 1f
;mov     al, 20h
stosb
;      ; movb $'',(r3)+
;jmp     short S28
;      ; br 1b
loop    S28

```

```

S29: ;1:
    mov     esi, numbuf + 6
           ; mov $numbuf+6,r1
    sub     esi, ebx
           ; sub (r5),r1
    ;mov    ecx, ebx
    mov     cl, bl
           ; mov (r5)+,-(sp)

S30: ;1:
    push    ecx
    lodsb
           ; movb (r1)+,r0
    ;mov    ebx, obuf
    call    putc
           ; jsr r5,putc; obuf
    pop     ecx
    loop    S30
           ; dec (sp)
           ; bne 1b
           ; tst (sp)+
           ; mov (sp)+,r3
           ; mov (sp)+,r2
           ; mov (sp)+,r1
    ;pop    edi
    retn
           ; rts r5

calcb:
           ; 06/10/2015 (32 bit modifications)
           ; calculate number of blocks
    add     eax, 511
    shr     eax, 9 ; (eax+511)/512
           ; add $511.,r0
           ; clrb r0
           ; swab r0
           ; asr r0
           ; eax = (size+511)/512
           ; large file ? (>=4096 bytes)
    cmp     eax, 8
           ; cmp r0,$8
    jb      short S31
           ; blo 1f
           ; add indirect block
    inc     eax
           ; inc r0

S31: ;1:
    ;1: ; ?
    retn
           ; rts r5

_mode:
    mov     al, '-'
mode:
           ; AL = mode char
           ;mov    (r5)+,r0
    ;mov    ebx, 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
    ;; EBX = buffer address
    push    esi
            ; mov r1, -(sp)
    mov     esi, obuf
    ; mov
    ; mov esi, ebx
            ; mov (r5)+, r1
S32: ;1:
    dec     word [esi+2]
            ; dec 2(r1)
    jns     short S33
            ; bge 1f
    push    eax
            ; mov r0, -(sp)
    call    fl
            ; jsr pc, fl
    pop     eax
            ; mov (sp)+, r0
    jmp     short S32
            ; br 1b
S33: ;1:
    mov     ebx, [esi+4]
    mov     [ebx], al
            ; movb r0, *4(r1)
    inc     dword [esi+4]
            ; inc 4(r1)
    pop     esi
            ; 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     ecx, esi
            ; mov r1, r0
    add     ecx, 8
            ; add $6, r0
    ; push
    ecx          ; Buffer data address
            ; mov r0, -(sp)
            ; mov r0, 0f
    mov     edx, [esi+4]          ; Buffer offset
            ; mov 4(r1), 0f+2
    or      edx, edx
    jz      short S34
            ; beq 1f

```

```

        sub     edx, ecx ; Byte count
        ; sub (sp),0f+2
        movzx   ebx, word [esi] ; File descriptor (=1)
        ; mov (r1),r0
        sys     _write ; sys _write, ebx, ecx, edx
        ; sys 0; 9f
; .data
; 9:
;           ; sys write; 0:...; ..
; .text
S34: ;1:
        ;pop     ecx
        mov     [esi+4], ecx ; Begin. of buf. data
        ; mov (sp)+,4(r1)
        mov     word [esi+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:
        ; EBX = file name offset
        ; EDI = buffer offset
        ;
        xor     ecx, ecx ; 0 => open for read
        sys     _open ; sys _open, ebx, ecx (0)
        jc      short S35
        stosw   ; file descriptor (in buffer offset 0)
        retn
S35:
        mov     ax, 0FFFFh ; -1
        stosw
        ;stc
        retn

; get words from input file
;
getw:
        ;mov     esi, ebx
        call    getc
        jc      short S36

        push    ax
        call    getc
        pop     dx
        mov     ah, dl
        xchg    ah, al
S36:
        retn

```



```

; get characters from input file
getc:
    ; ESI = buffer address
    mov     ax, [esi+2] ; char count
    and     ax, ax
    jnz     short gch1

gch0:
    mov     ecx, esi
    add     ecx, 8      ; read buff. addr.
    movzx   ebx, word [esi]
    mov     [esi+4], ecx ; char offset
    ;xor     ax, ax
    ;mov     [esi+2], ax ; 0
    mov     edx, 512
    sys     _read ; sys _read, ebx, ecx, edx
    jc      short gch2
    or      eax, eax
    jz      short gch3

gch1:
    dec     ax
    mov     [esi+2], ax
    mov     ebx, [esi+4]
    mov     al, [ebx]
    inc     ebx
    mov     [esi+4], ebx
    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 'ctime386.inc' ; 24/11/2013
dw 417
brk:      dd _end + 512 ; (gstat:)
; directory name pointer
dnp:      dd 0 ; (do:)
dotp:     dd dot
;dotp:    dot

```

```

euids:  dd 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:     dd 0 ; (pass3:)

align 4
bss_start:

ABSOLUTE bss_start
;EVEN
;bss:
    count:    resw 1
    ocount:   resw 1
    longf:     resw 1
    allflg:   resw 1
    dirflg:   resw 1
    isadir:   resw 1
    tblocks:  resd 1
    sortoff:  resd 1
    filnam:   resb 32
    statb:    resb 34
    dbuf:     resb 520
    ; 2 byte file descriptor, 2 byte buffer data size,
    ; 4 byte buffer offset, 512 byte buffer data
    obuf:     resb 520
    numbuf:   resb 6
    uidbuf:   resb 1024
    euidbuf:
    iobuf:    resb 520
alignb 4
    _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.

```