```
; ****************************************************************************
;
; SH.ASM  (Retro Unix 8086 v1 Shell - /bin/sh)
; ----------------------------------------------------------------------------
;
; 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/sh file
;
; [ Last Modification: 08/04/2014 ]
;
; Derivation from UNIX Operating System (v1.0 for PDP-11)
; (Original) Source Code by Ken Thompson (Bell Laboratories, 1971-1972)
;
; ****************************************************************************

; <Preliminary Release of UNIX Implementation Document>
; <Isuse: D, Date: 17/3/1972, ID: IMO.1-1, Section: E.11>
; <sh - command interpreter>
;
; SHELL02.ASM, 13/11/2013
;
; ****************************************************************************

.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:
;/ sh -- command interpreter
        ;mov    byte ptr [_echo], 1 ; 06/12/2013
        mov     bp, sp
                ; mov sp,r5
        mov     word ptr [shellarg], bp
                ; mov r5,shellarg / save orig sp in shellarg
        mov     bx, word ptr [BP]+2
        cmp     byte ptr [BX], '-'
                 ; cmpb *2(r5),$'- / was this sh called by init or loginx~
        jne     short @f
                ; bne 2f / no
        sys     _intr, 0
                ; sys intr; 0 / yes, turn off interrupts
        sys     _quit, 0
                ; sys quit; 0
        sys     _write, 1, msg_unix_sh, msgsh_size
@@: ;2:
        sys     _getuid
                ; sys  getuid / who is user
        ;and    ax, ax
        and     al, al
                ; tst r0 / is it superuser
        jnz     short @f
                ; bne 2f / no
        mov     byte ptr [at], '#'
                ; movb $'#,at / yes, set new prompt symbol
@@: ;2:
        cmp     word ptr [BP], 1
                ; cmp (r5),$1 / tty input?
        jna     short newline
                ; ble newline / yes, call with '-(or with no command
                      ; / file name)
        xor     bx, bx
                ; clr r0 / no, set tty
        sys     _close
                ; sys close / close it
        mov     bx, word ptr [BP]+4 ; arg 1
                ; mov 4(r5),0f / get new file name
        xor     cx, cx ; arg 2
        sys     _open
                ; sys open; 0:..; 0 / open it
        jnc     short @f
                ; bec 1f / branch if no error
        mov     si, offset msgNotFound
        call    error
                ; jsr r5,error / error in file name
                ; <Input not found\n\0>; .even
        sys     _exit
                ; sys exit
@@: ;1:
        mov     byte ptr [at], 0
                ; clr at / clear prompt character, if reading non-tty
                   ; / input file
        jmp     short newcom
newline:
        cmp     byte ptr [at], 0
                ; tst at / is there a prompt symbol
        jna     short newcom
                ; beq newcom / no
                ; mov $1,r0 / yes
nl:
        sys     _write, 1, prompt, p_size
        ;sys    _write, 1, at, 2
                ; sys write; at; 2. / print prompt
```

```
newcom:
       mov    sp, word ptr [shellarg]
              ; mov shellarg,sp /
       mov    si, offset parbuf
              ; mov $parbuf,r3 / initialize command list area
       mov    di, offset parp
              ; mov $parp,r4 / initialize command list pointers
       xor    ax, ax
       mov    word ptr [infile], ax ; 0
              ; clr infile / initialize alternate input
       mov    word ptr [outfile], ax ; 0
              ; clr outfile / initialize alternate output
       mov    byte ptr [glflag], al ; 0
       ;mov   word ptr [glflag], ax ; 0
              ; clr glflag / initialize global flag
newarg:
       call   blank
              ; jsr pc,blank / squeeze out leading blanks
       call   delim
       je     short nch4 ; '\n', ';', '&'
              ; jsr r5,delim / is new character a ; \n or &
              ;     br 2f / yes
       push   si
              ; mov r3,-(sp) / no, push arg pointer onto stack
       mov    bp, sp
       cmp    al, '<'
              ; cmp r0,$'< / new input file?
       jne    short na1
              ; bne 1f / no
       mov    word ptr [infile], si
              ; mov (sp),infile / yes, save arg pointer
       jmp    short na2
       ;mov   word ptr [BP], 0
              ; clr (sp) / clear pointer
       ;jmp   short nch1
              ; br 3f
na1: ;1:
       cmp    al, '>'
              ; cmp r0,$'> / new output file?
       jne    short nch0
       ;jne   short newchar
              ; bne newchar / no
       mov    word ptr [outfile], si
              ; mov (sp),outfile / yes, save arg pointer
na2:
       mov    word ptr [BP], 0
              ; clr (sp) / clear pointer
       jmp    short nch1
              ; br 3f
newchar:
       cmp    al, 20h
              ; cmp $' ,r0 / is character a blank
       je     short nch2
              ; beq 1f / branch if it is (blank as arg separator)
        cmp    al, 8Dh ; 128 + 13
       je     short nch2
              ; cmp $'\n+200,r0 / treat \n preceded by \
              ; beq 1f / as blank
nch0:
       call   putc
              ; jsr pc,putc / put this character in parbuf list
nch1: ;3:
       call   getc
              ; jsr pc,getc / get next character
       call   delim
       jne    short newchar
       ;jz    short nch2 ; '\n', ';', '&'
              ; jsr  r5,delim / is char a ; \n or &,
              ;       br 1f / yes
       ;jmp   short newchar
              ; br newchar / no, start new character tests
```

```
nch2: ;1:
       mov    byte ptr [SI], 0
       inc    si
              ; clrb (r3)+ / end name with \0 when read blank,
                        ; or delim
       pop    bx
       mov    word ptr [DI], bx
              ; mov (sp)+,(r4)+ / move arg ptr to parp location
       or     bx, bx
       jz     short nch3
       ;jnz   short nch3
              ; bne 1f / if (sp)=0, in file or out file points
                             ; to arg
       inc    di
       inc    di
              ; tst -(r4) / so ignore dummy (0), in pointer list
nch3: ;1:
       call   delim
       jne    short newarg
       ;jz    short nch4 ; '\n', ';', '&'
              ; jsr r5,delim / is char a ; \n or &.
              ;     br 2f / yes
       ;jmp   short newarg
              ; br newarg / no, start newarg processing
nch4: ;2:
       mov    word ptr [DI], 0
              ; clr (r4) / \n, &, or ; takes to here
                       ; / (end of arg list) after 'delim' call
       push   ax
              ; mov r0,-(sp) / save delimiter in stack
       call   docom
              ; jsr pc,docom / go to exec command in parbuf
       mov    bp, sp
       cmp    byte ptr [BP], '&'
              ; cmpb (sp),$'& / get a new command without wait?
       je     short newcom
              ; beq newcom / yes
       and    dx, dx
              ; tst r1 / was chdir just executed or line ended
                      ; / with ampersand?
       jz     short nch6
              ; beq 2f / yes
nch5: ;1:
       sys    _wait
              ; sys wait / no, wait for new process to terminate
                        ; / command executed)
       jc     short nch6
              ; bcs 2f / no, children not previously waited for
       cmp    ax, dx
              ; cmp r0,r1 / is this my child
       jne    short nch5
              ; bne 1b
nch6: ;2:
       cmp    byte ptr [BP], 0Dh
       ;cmp   byte ptr [BP], 0Ah
              ; cmp (sp),$'\n / was delimiter a new line
        je     newline
              ; beq newline / yes
        jmp    newcom
              ; br newcom / no, pick up next command
```

```
docom:
        sub     di, offset parp
                ; sub $parp,r4 / out arg count in r4
        jne     short dcom1
                ; bne 1f / any arguments?
dcom0:
        sub     dx, dx ; 0
                ; clr r1 / no, line ended with ampersand
        retn
                ; rts pc / return from call
dcom1: ;1:
        mov     bx, di
        ; 06/12/2013
        mov     si, offset qecho
        call    chcom
        jnz     short dcom7
        cmp     bl, 4
        jne     short dcom8
        mov     bx, word ptr [parp+2]
        cmp     byte ptr [bx], 'o'
        jne     short dcom9
        inc     bx
        cmp     byte ptr [BX], 'n'
        jne     short dcom10
        inc     bx
        cmp     byte ptr [BX], 0
        ja      short dcom9
        mov     byte ptr [_echo], 1
        jmp     short dcom0
dcom10: ; 06/12/2013
        cmp     word ptr [BX], 'ff'
        jne     short   dcom9
        inc     bx
        inc     bx
        cmp     byte ptr [BX], 0
        ja      short dcom9
        mov     byte ptr [_echo], 0
        jmp     short dcom0
dcom9: ; 06/12/2013
        mov     si, offset msgNoCmd
        call    error
        jmp     short dcom0
dcom7:
        mov     si, offset qchdir
        call    chcom
        jnz     short dcom4
                ; jsr r5,chcom; qchdir / is command chdir?
                ;     br 2f / command not chdir
dcom12:
        cmp     bl, 4
        ;cmp    bx, 4
                ; cmp r4,$4 / prepare to exec chdir,
                          ; 4=arg count x 2
        je      short dcom2
                ; beq 3f
dcom8:
        mov     si, offset msgArgCount
        call    error
                ; jsr r5,error / go to print error
                ;       <Arg count\n\0>; .even
        ;jmp    short dcom3
                ; br 4f
        jmp     short dcom0
dcom2: ;3:
                ;mov parp+2,0f / more directory name to sys call
        mov     bx, word ptr [parp+2]
        sys     _chdir
                ; sys chdir; 0:0 / exec chdir
        jnc     short dcom3
                ; bec 4f / no error exit
        mov     si, msgBadDir
        call    error
                ; jsr r5,error / go to print error
                ;       <Bad directory\n\0>; .even
                               ; / this diagnostic
```

```
dcom3: ;4:
       xor     dx, dx ; 0
               ; clr r1 / set r1 to zero to skip wait
       retn
               ; rts pc / and return
dcom4: ;2:
       ; 06/12/2013
       mov     si, offset qcd
       call    chcom
       jz      short dcom12
dcom11:
       mov     si, offset glogin
       call    chcom
       jnz     short dcom5
               ; jsr r5,chcom; glogin / is command login?
               ;     br 2f / not loqin, go to fork
       sys     _exec, parbuf, parp
               ; sys exec; parbuf; parp / exec login
       sys     _exec, binpb, parp
               ; sys exec; binpb; parp / or /bin/login
dcom5: ;2: / no error return??
       mov     bx, offset newproc
       ; child process will return to 'newproc' address
       sys     _fork
               ; sys fork / generate sh child process
                       ; for command
                ;     br newproc / exec command with
                       ; new process
       ; parent process will return here
       jnc     short dcom6
               ; bec 1f / no error exit, old process
       mov     si, offset msgTryAgain
       call    error
               ; jsr r5,error / go to print error
               ;     <Try again\n\0>; .even / this diagnostic
        jmp     newline
               ; jmp newline / and return for next try
dcom6: ;1:
       mov     dx, ax ; child process ID
               ; mov r0,r1 / save id of child sh
       retn
               ; rts pc / return to "jsr pc, docom" call
                      ; in parent sh
error:
       sys     _write, 1, nextline, 2
@@:
       lodsb
       mov     byte ptr [och], al
                ; movb (r5)+,och / pick up diagnostic character
       and     al, al
       jz      short @f
               ; beq 1f / 0 is end of line
       sys     _write, 1, och, 1
               ; mov $1,r0 / set for tty output
               ; sys write; och; 1 / print it
       jmp     short @b
       ;jmp    short error
               ; br error / continue to get characters
@@: ;1:
       ;inc    si
               ; inc r5 / inc r5 to point to return
       ;and si, 0FFFEh
       ;shr    si, 1
       ;shl    si, 1
               ; bic $1,r5 / make it even
       sys     _seek, 0, 0, 2
               ; clr r0 / set for input
               ; sys seek; 0; 2 / exit from runcom. skip to
                             ; / end of input file
       retn
               ;; ((/ rts r5))
               ;; (not in original unix v1 'sh.s')
```

```
chcom: ; / has no effect if tty input
               ; mov (r5)+,r1 / glogin gchdir r1, bump r5
       mov     di, offset parbuf
               ; mov $parbuf,r2 / command address  r2 'login'
@@: ;1:
       lodsb
               ; movb (r1)+,r0 / is this command 'chdir'
       scasb
                ; cmpb (r2)+,r0 / compare command name byte
                              ; / with 'login' or 'chdir'
       jne     short @f
               ; bne 1f / doesn't compare
       or      al, al
               ; tst r0 / is this
       jnz     @b
               ; bne 1b / end of names
               ; tst (r5)+ / yes, bump r5 again to execute
                       ; / login or chdir
@@: ;1:
       retn
               ; rts r5 / no, return to exec command

putc:
       cmp     al, 27h ; '
               ; cmp r0,$'' / single quote?
       je      short pch1
               ; beq 1f / yes
       cmp     al, 22h ; "
               ; cmp r0,$'" / double quote
       je      short pch1
               ; beq 1f / yes
       and     al, 7Fh
               ; bic $!177,r0 / no, remove 200, if present
       mov     byte ptr [SI], al
       inc     si
               ; movb r0,(r3)+ / store character in parbuf
       retn
               ; rts pc
pch1: ;1:
       push    ax
               ; mov r0,-(sp) / push quote mark onto stack
pch2: ;1:
       call    getc
               ; jsr pc,getc / get a quoted character
       cmp     al, 0Dh
       ;cmp    al, 0Ah ; \n
               ; cmp r0,$'\n / is it end or line
       jne     short pch3
               ; bne 2f / no
       mov     si, offset msgImbalance
       call    error
               ; jsr r5,error / yes, indicate missing
                             ; quote mark
               ;     <"' imbalance\n\0>; .even
        jmp     newline
               ; jmp newline / ask for new line
pch3: ;2:
       mov     bp, sp
       cmp     byte ptr [BP], al
               ; cmp r0,(sp) / is this closing quote mark
       je      short pch4
               ; beq 1f / yes
       and     al, 7Fh
               ; bic $!177,r0 / no, strip off 200
                             ; if present
       mov     byte ptr [SI], al
       inc     si
               ; movb r0,(r3)+ / store quoted character
                             ; in parbuf
       jmp     short pch2
               ; br 1b / continue
pch4: ;1:
       pop     ax
               ; tst (sp)+ / pop quote mark off stack
       retn
               ; rts pc / return
; / thp`e new process

newproc:
       mov     si, word ptr [infile]
       or      si, si
       jz      short np2
               ; mov infile,0f / move pointer to new file name
               ; beq 1f / branch if no alternate read file given
       cmp     byte  ptr [SI], 0
               ; tstb *0f
       jna     short np1
               ; beq 3f / branch if no file name given
       sys     _close, 0
               ; clr r0 / set tty input file name
               ; sys close / close it
       sys     _open, si, 0
               ; sys open; 0:..; 0 / open new input file
                                 ; for reading
       jnc     short np2
               ; bcc 1f / branch if input file ok
np1: ;3:
       mov     si, offset msgInputFile
       call    error
               ; jsr r5,error / file not ok, print error
               ;      <Input file\n\0>; .even / this diagnostic
       sys     _exit
               ; sys exit / terminate this process
                         ; and make parent sh
np2: ;1:
       mov     si, word ptr [outfile]
               ; mov outfile,r2 / more pointer to new file name
       and     si, si
       jz      short np6
               ; beq 1f / branch if no alternate write file
       cmp     byte ptr [SI], '>'
               ; cmpb (r2),$'> / is > at beqinning of file name?
       jne     short np3
               ; bne 4f / branch if it isn't
       inc     si
               ; inc r2 / yes, increment pointer
       sys     _open, si, 1
               ; mov r2,0f
               ; sys open; 0:..; 1 / open file for writing
       jnc     short np5
               ; bec 3f / if no error
np3: ;4:
       sys     _creat, si, 15 ; Decimal 15 = Octal 17
               ; mov r2,0f
               ; sys creat; 0:..; 17 / create new file
                                   ; with this name
       jnc     short np5
               ; bec 3f / branch if no error
np4: ;2:
       mov     si, offset msgOutputFile
       call    error
               ; jsr r5,error
               ;      <Output file\n\0>; .even
       sys     _exit
               ; sys exit
np5: ;3:
       sys     _close, ax
               ; sys close / close the new write file
               ; mov   r2,0f / move new name to open
       sys     _close, 1
               ; mov $1,r0 / set tty file name
               ; sys close / close it
       sys     _open, si, 1
               ; sys open; 0:..; 1 / open new output file,
                             ; /it now has file descriptor 1
       sys     _seek, ax, 0, 2
               ; sys seek; 0; 2 / set pointer to
                              ; current end of file
```

```
np6: ;1:
       cmp    byte ptr [glflag], 0
              ; tst glflag / was *, ? or [ encountered?
       ja     short np9
              ; bne 1f / yes
       sys    _exec, parbuf, parp
              ; sys exec; parbuf; parp / no, execute
                                    ;  this command
       sys    _exec, binpb, parp
              ; sys exec; binpb; parp / or /bin/this command
np7: ;2:
       sys    _stat, binpb, inbuf
              ; sys stat; binpb; inbuf / if can't execute
                                    ; / does it exist?
       jc     short np8
              ; bes 2f / branch if it doesn't
       mov    si, offset parp-2
       mov    word ptr [SI], offset shell
              ; mov $shell,parp-2 / does exist,
                                ;  not executable
       mov    ax, offset binpb
       mov    word ptr [parp], ax
              ; mov $binpb,parp / so it must be
       sys    _exec, shell, si
              ; sys exec; shell; parp-2 / a command file,
              ; / get it with sh /bin/x (if x name of file)
np8: ;2:
       mov    si, offset msgNoCmd
       call   error
              ; jsr r5,error / a return for exec
                              ; is the diagnostic
              ;     <No command\n\0>; .even
       mov    sp, word ptr [shellarg]
       sys    _exit
              ; sys exit
np9: ;1:
       mov    si, offset parp-2
       mov    word ptr [SI], offset glob
              ; mov $glob,parp-2 / prepare to process *,?
       sys    _exec, glob, si
              ; sys exec; glob; parp-2
                           ; / execute modified command
       jmp    short np8
              ; br 2b

delim:
        cmp     al, 0Dh ; carriage return
       je     short dlim2
       ;cmp   al, 0Ah
              ; cmp r0,$'\n / is character a newline
       ;je    short dlim2
              ; beq 1f
       cmp    al, '&'
              ;cmp r0,$'& / is it &
       je     short dlim2
              ; beq 1f / yes
       cmp    al, ';'
              ; cmp r0,$'; / is it ;
       je     short dlim2
              ; beq 1f / yes
       cmp    al, '?'
              ; cmp r0,$'? / is it ?
       je     short dlim1
              ; beq 3f
       cmp    al, '['
              ; cmp r0,$'[ / is it beginning of character string
                    ; / (for glob)
       jne    short dlim2
              ; bne 2f
dlim1: ;3:
       inc    byte ptr [glflag]
              ; inc glflag / ? or * or [ set flag
;2:
       ;tst   (r5)+ / bump to process all except \n,;,&
dlim2: ;1:
       ; zf = 1 if the char is '\n' or ';' or '&'
       retn
              ; rts r5
```

```
blank:
       call   getc
              ; jsr pc,getc / get next character
       cmp    al, 20h
              ; cmp $' ,r0 / leading blanks
       je     short blank
              ; beq blank / yes, 'squeeze out'
       cmp    al, 8Dh ; 80h + 0Dh
        ;cmp    al, 8Ah ; 80h + 0Ah
       je     short blank
               ; cmp r0,$200+'\n / new-line preceded by \
                              ;  is translated
              ; beq blank / into blank
@@:
       retn
              ; rts pc
getc:
       cmp    word ptr [param], 0
              ; tst param / are we substituting for $n
       ja     short gch3
              ; bne 2f/ yes
gch0:
        mov     bx, word ptr [inbufp]
              ; mov inbufp,r1 / no, move normal input pointer to r1
@@:
       cmp    bx, word ptr [einbuf]
              ; cmp r1,einbuf / end of input line?
       jb     short gch1
              ; bne 1f / no
       call   getbuf
              ; jsr pc,getbuf / yes, put next console line
                             ;  in buffer
       jmp    short gch0
              ; br getc
gch1: ;1:
       mov    al, byte ptr [BX]
       inc    bx
              ; movb (r1)+,r0 / move byte from input buffer to r0
       mov    word ptr [inbufp], bx
              ; mov r1,inbufp / increment routine
       or     al, byte ptr [escap]
       ;or    ax, escap
              ; bis escap,r0 / if last character was \ this adds
                             ; / 200 to current character
       ;mov   byte ptr [escap], 0
       ;mov   word ptr [escap], 0
              ; clr escap / clear, so escap normally zero
       cmp    al, '\'
              ; cmp r0,$'\\ / note that \\ is equal \ in as
       je     short gch2
              ; beq 1f
       mov    byte ptr [escap], 0
       cmp    al, '$'
              ; cmp r0,$'$ / is it $
       je     short gch5
              ; beq 3f / yes
       retn
              ; rts pc / no
gch2: ;1:
        mov     byte ptr [escap], 80h
       ;mov   word ptr [escap], 128
              ; mov $200,escap / mark presence of \ in command line
       jmp    short @b
       ;jmp   short gch0
              ; br getc / get next character
gch3: ;2:
       mov    bx, word ptr [param]
       mov    al, byte ptr [BX]
              ; movb *param,r0 / pick up substitution character
                             ; / put in r0
       or     al, al
       jz     short gch4
              ; beq  1f / if end of substitution arg, branch
       inc    word ptr [param]
              ; inc param / if not end, set for next character
       retn
              ; rts pc / return as though character in ro is normal
                      ; / input
```

```
gch4: ;1:
       mov    word ptr [param], 0
              ; clr param / unset substitution pointer
       jmp    short gch0
              ; br getc / get next char in normal input
gch5: ;3:
       call   gch0
       ;call  getc
              ; jsr pc,getc / get digit after $
       sub    al, '0'
              ; sub $'0,r0 / strip off zone bits
       cmp    al, 9
              ; cmp r0,$9. / compare with digit 9
       jna    short gch6
              ; blos 1f / less than or equal 9
       mov    al, 9
              ; mov $9.,r0 / if larger than 9, force 9
gch6: ;1:
       mov    bx, word ptr [shellarg]
              ; mov shellarg,r1 / get pointer to stack for
                         ; / this call of shell
       cbw    ; al->ax (ah=0)
       inc    al
       ;inc   ax
              ; inc r0 / digit +1
       cmp    ax, word ptr [BX]
              ; cmp r0,(r1) / is it less than # of args
                           ;in this call
       jnb    short gch0
              ; bge getc / no, ignore it. so this $n is not replaced
       shl    ax, 1
              ; asl r0 / yes, multiply by 2 (to skip words)
       add    bx, ax
              ; add r1,r0 / form pointer to arg pointer (-2)
       mov    ax, word ptr [BX]+2
       mov    word ptr [param], ax
              ; mov 2(r0),param / move arg pointer to param
       jmp    short getc
              ; br getc / go to get substitution arg for $n
getbuf:
       mov    cx, offset inbuf
              ; mov $inbuf,r0 / move input buffer address
        mov    word ptr [inbufp], cx
              ; mov r0,inbufp / to input buffer pointer
       mov    word ptr [einbuf], cx
              ; mov r0,einbuf / and initialize pointer to end of
                        ; / character string
       dec    cx
              ; dec r0 / decrement pointer so can utilize normal
                      ; / 100p starting at 1f
              ; mov r0,0f / initialize address for reading 1st char
       mov    dx, 1
gbuf0: ;1:
       inc    cx
              ; inc 0f / this routine filles inbuf with line from
                      ; / console - if there is cnc
       push   cx
       ; dx = 1
       sys    _read, 0, och
       pop    cx
       ;xor   bx, bx ; 0
       ;sys   _read ; sys _read, bx, cx, dx ; bx = 0, dx = 1
              ; sys read; 0:0; 1 / read next char into inbuf
        jc     xit1
              ; bcs xit1 / error exit
       and    ax, ax
              ; tst r0 / a zero input is end of file
        jz     short xit1
              ; beq xit1 / exit
       inc    word ptr [einbuf]  ; 08/04/2014
       mov    al, byte ptr [och]
       cmp    byte ptr [at], 0
       jna    short gbuf1
       cmp    al, 8 ; backspace
       je     short gbuf3
       cmp    al, 127 ; delete
       je     short gbuf6 ; 06/12/2013
```

```
gbuf1:
        ;mov    bx, cx
        ;inc    word ptr [einbuf]
                ; inc einbuf / eventually einbuf points to \n
                       ; / (+1) of this line
        cmp     cx, offset inbuf + 256
                ; cmp 0b,$inbuf+256. / have we exceeded
                                  ; input buffer size
        jnb     short xit1
                ; bhis xit1 / if so, exit assume some sort of binary
        ; 08/04/2014
        cmp     al, 0Dh
        jne     short gbuf8
        mov     bx, word ptr [einbuf]
        dec     bx
        mov     byte ptr [BX], al
        retn
gbuf8:
        mov     bx, cx
        mov     byte ptr [BX], al
        ;cmp    al, 0Ah ; \n
                ; cmpb  *0b,$'\n / end of line?
        ;je     short  gbuf5
        ;jne    short gbuf1
                ; bne 1b / no, go to get next char
        ;cmp    al, 0Dh ; ENTER
        ;je     short gbuf5
        cmp     byte ptr [at], 0 ; at > 0 --> tty input
        jna     short gbuf0
        cmp     al, 1Bh ; ESC
        jne     short gbuf2
        mov     ax, offset inbuf
        mov     word ptr [inbufp], ax
        mov     word ptr [einbuf], ax
        jmp     nl  ; cancel current command, new line
gbuf2:
         ; 06/12/2013
        cmp     byte ptr [at], 0
        ja      short gbuf7
        cmp     byte ptr [_echo], 0
        jna     short gbuf0
gbuf7:
        push    cx
        ;mov    byte ptr [och], al
        ; DX = 1
        sys     _write, 1, och
        ;sys    _write, 1, och, 1  ; echo (write char on tty)
        pop     cx
         jmp     short gbuf0
gbuf6: ; DELETE key -> BACKSPACE key
        ; mov   al, 8
        mov     byte ptr [och], 8 ; 06/12/2013
gbuf3:
        ; 08/04/2014
        dec     word ptr [einbuf]
        ; 12/12/2013
        dec     cx
        cmp     cx, offset inbuf
        jb      short gbuf4
        dec     cx
        ; 08/04/2014
        ;jmp    short gbuf2
        jmp     short gbuf7
gbuf4:
        ;mov    al, 7
        mov     byte ptr [och], 07h ; beep
        ; 08/04/2014
        ;jmp    short gbuf2
        jmp     short gbuf7
;gbuf5:
;       retn
                ; rts pc / yes, return

xit1:
        sys     _exit
                ; sys exit
```

```
;quest:
        ;db '?', 0Dh, 0Ah
        ;<?\n>

prompt:
        db 0Dh, 0Ah
at:
        db '@ '
        ;<@ >
p_size  equ $ - offset prompt

; 06/12/2013
_echo:  db 1
qecho:  db 'echo', 0
;
qcd:    db 'cd', 0
;

qchdir:
        db 'chdir', 0
        ;<chdir\0>
glogin:
        db 'login', 0
        ;<login\0>
shell:
        db '/bin/sh', 0
        ;</bin/sh\0>
glob:
        db '/etc/glob', 0
        ;</etc/glob\0>
binpb:
        db '/bin/'
        ;</bin/>
parbuf:
        db 1000 dup(0)
        ; .=.+1000.
EVEN
        ;.even
param:
        dw 0
        ;.=.+2
glflag:
        db 0
        db 0
        ;.=.+2
infile:
        dw 0
        ; .=.+2
outfile:
        dw 0
        ;.=.+2
        dw 0
        ;.=.+2 / room for glob
parp:
        db 200 dup(0)
        ;.=.+200.
inbuf:
        db 256 dup(0)
        ;.=.+256.
;escap:
        ;dw 0
        ;.=.+2
inbufp:
        dw 0
        ;.=.+2
einbuf:
        dw 0
        ;.=.+2
och:
        dw 0
        ;.=.+2
shellarg:
        dw 0
        ;.=.+2
escap:
        db 0
        ;
        db 0
```

```
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;  messages
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

msg_unix_sh:   db 0Dh, 0Ah
               db 'Retro Unix 8086 v1 - shell'
               ;db 0Dh, 0Ah
msgsh_size equ $ - offset msg_unix_sh
               ;db 0
               db '12/12/2013'
nextline:      db 0Dh, 0Ah, 0
;Error messages:
msgNotFound:   db 'Input not found', 0
msgArgCount:   db 'Arg count',  0
msgBadDir:     db 'Bad directory', 0
msgTryAgain:   db 'Try again', 0
msgImbalance:   db 22h, 27h, 20h, 'imbalance',  0
msgInputFile:  db 'Input file', 0
msgOutputFile:        db 'Output file', 0
msgNoCmd:      db 'No command',  0

UNIX           ends

                end     START_CODE
```