```
asm d:romload.dzx
DOS/65 ASSEMBLER
VERSION 2.02-S-6
END OF FIRST PASS
SORTING SYMBOL TABLE
 0000            ;DOS/65 LOADER
 0000            ;OSI Version
 0000            ;for c1p and sbii
 0000            ;Version 2.10
 0000            ;released:      4 august 1983
 0000            ;The OSI LOADER and BOOT are unique due to
 0000            ;the total impossibility of putting all BOOT
 0000            ;code into a single DOS/65 sector.  The
 0000            ;approach taken is to have all the code in
 0000            ;LOADER and to have BOOT supply the variable
 0000            ;data such as load address and sectors to
 0000            ;load.  LOADER can be located in ROM
 0000            ;and need not change as MSIZE or PAGES
 0000            ;changes.
 0000            ;definitions
 0000            speed   =       49              49=1MHz
 0000            numtrk  =       40              number of tracks
 0000            sectrs  =       16              sectors per track
 0000            stprte  =       5               step rate in ms
 0000            ;osi addresses and parameters
 0000            scrpgs  =       4               pages to clear in screen
 0000            screen  =       $d385           prompt location
 0000            scrcnt  =       $d800           control port for video
 0000            romkbd  =       $fd00           rom polled keyboard input
 0000            ;pia
 0000            flpsts  =       $c000           status port
 0000            ;bit definitions
 0000            ; 0     drive 0 ready if 0
 0000            ; 1     track 0 if 0
 0000            ; 2
 0000            ; 3
 0000            ; 4
 0000            ; 5     write protect if 0
 0000            ; 6     drive select (0=B or D, 1=A or C)
 0000            ; 7     index if 0
 0000            flpcnt  =       flpsts+2
 0000            ;bit definitions
 0000            ; 0     write enable if 0
 0000            ; 1     erase enable if 0
 0000            ;         enable 200us after write enable
 0000            ;         disable 530us after write disable
 0000            ; 2     step
 0000            ;         0 if in
 0000            ;         1 if out (to track zero)
 0000            ; 3     step on falling edge
 0000            ; 4
 0000            ; 5     side select (0=C or D, 1=A or B)
 0000            ; 6
 0000            ; 7     head load if 0
 0000            ;acia
 0000            flpdta  =       $c011           data port
 0000            ;page zero data initialized by boot
```

```
0000                    *       =       $00
0000                    ldeadr  *=      *+2             load address
0002                    pointr  =       ldeadr          just used for screen clear
0002                    simadr  *=      *+2             sim entry address
0004                    secLde  *=      *+1             sectors to load
0005                    sectrk  *=      *+1             sectors per track
0006                    nxtsec  *=      *+1             next sector to read
0007                    ;page zero data used by LOADER but not
0007                    ;initialized by BOOT
0007                    trkpnt  *=      *+2             track pointer
0009                    rdeadr  *=      *+2             read address
000B                    curtrk  *=      *+1             current track
000C                    ;main program
000C                    ;if in rom change the next line to the appropriate address
000C                    *       =       $e340
E340 A2 FF                      ldx     #$ff            set stack
E342 9A                         txs
E343 D8                         cld                     binary mode
E344 78                         sei                     disable interrupts
E345                    ;clear screen
E345 A2 04              loader  ldx     #scrpgs         pages to clear
E347 A9 D0                      lda     #$d0            starting page
E349 85 01                      sta     pointr+1        and set
E34B A0 00                      ldy     #0              clear lower part
E34D 84 00                      sty     pointr          of pointer
E34F A9 20                      lda     #' '            get space
E351 91 00              clrslp  sta     (pointr),y      put space on screen
E353 C8                         iny                     bump index
E354 D0 FB                      bne     clrslp          loop if more
E356 E6 01                      inc     pointr+1        bump high pointer
E358 CA                         dex                     drop page count
E359 D0 F6                      bne     clrslp          loop if more
E35B                    ;do opening
E35B BD FD E3           opnlpe  lda     opnmsg,x        get char
E35E 9D 85 D3                   sta     screen,x        put on screen
E361 E8                         inx                     bump index
E362 E0 0B                      cpx     #11             see if past end
E364 D0 F5                      bne     opnlpe          loop if not
E366                    ;initialize pia
E366 A9 40                      lda     #%01000000
E368 A0 00                      ldy     #0
E36A A2 04                      ldx     #%00000100
E36C 8C 01 C0                   sty     flpsts+1        ddr on a side
E36F 8D 00 C0                   sta     flpsts          all but one are input
E372 8E 01 C0                   stx     flpsts+1        back to data
E375 8D 00 C0                   sta     flpsts          set output to high
E378 8C 03 C0                   sty     flpcnt+1        ddr on b side
E37B 88                         dey                     y to ff
E37C 8C 02 C0                   sty     flpcnt          all are output
E37F 8E 03 C0                   stx     flpcnt+1        back to data
E382 8C 02 C0                   sty     flpcnt          set all high
E385 20 0E E4                   jsr     home            home it
E388 20 7C E4                   jsr     rdytrk          get header
E38B D0 5E                      bne     error           branch if error
E38D 20 B7 E4           getv    jsr     rd1byt          look for rest of header
E390 C9 76                      cmp     #'v'
E392 D0 F9                      bne     getv            loop until v
E394 20 B7 E4                   jsr     rd1byt          now get number
E397 C9 01                      cmp     #1              if not 1
E399 D0 50                      bne     error           is error
E39B 20 B7 E4                   jsr     rd1byt          now get length
E39E C9 08                      cmp     #sectrs/2       see if correct
E3A0 D0 49                      bne     error           error if not
E3A2                    ;got good header
E3A2 A2 00                      ldx     #0              clear index
E3A4 A9 01              btelpe  lda     #%00000001      mask for ready
```

```
E3A6 2C 10 C0  tstflp  bit     #1pdta-1          test acia
E3A9 F0 FB             beq     tstflp            loop if not ready
E3AB AD 11 C0          lda     flpdta            else get byte
E3AE 70 3B             bvs     error             parity error
E3B0 95 00             sta     $00,x             else put in page zero
E3B2 E8                inx                       bump index
E3B3 10 EF             bpl     btelpe            then loop
E3B5           ;got a good boot so all is initialized
E3B5           ;start read again
E3B5 20 DE E4  gettrk  jsr     rdetrk            read entire track
E3B8 B0 31             bcs     error             exit if error
E3BA 20 C1 E4  gottrk  jsr     strkpt            set pointer
E3BD A0 00             ldy     #0                now move a sector
E3BF B1 07     mvesec  lda     (trkpnt),y
E3C1 91 00             sta     (ldeadr),y
E3C3 C8                iny
E3C4 10 F9             bpl     mvesec            of 128 bytes
E3C6 18                clc                       add 128 to pointer
E3C7 98                tya
E3C8 65 00             adc     ldeadr
E3CA 85 00             sta     ldeadr
E3CC 90 02             bcc     *+4
E3CE E6 01             inc     ldeadr+1
E3D0 C6 04             dec     seclde            drop count
E3D2 F0 14             beq     alllde            done if all loaded
E3D4 E6 06             inc     nxtsec            else bump sector
E3D6 A5 06             lda     nxtsec            get it
E3D8 C5 05             cmp     sectrk            compare to max
E3DA F0 DE             beq     gottrk            ok if equal
E3DC 90 DC             bcc     gottrk            or less
E3DE A9 01             lda     #1                else reset
E3E0 85 06             sta     nxtsec            sector
E3E2 20 2D E4          jsr     stepin            step in a track
E3E5 4C B5 E3          jmp     gettrk            and loop to read
E3E8           ;data all read and moved
E3E8 6C 02 00  alllde  jmp     (simadr)          execute
E3EB           ;general error handler
E3EB 20 73 E4  error   jsr     unldhd            unload head
E3EE A2 00             ldx     #0                now send error message
E3F0 BD 08 E4  errlpe  lda     errmsg,x          get char
E3F3 9D 85 D3          sta     screen,x          send to screen
E3F6 E8                inx                       bump index
E3F7 E0 06             cpx     #6                see if too big
E3F9 D0 F5             bne     errlpe            loop if not
E3FB F0 FE     forevr  beq     forevr            else loop forever
E3FD           ;messages
E3FD 44 4F 53  opnmsg  .byt    'DOS/65 V2.1'
E408 45 52 52  errmsg  .byt    'ERROR!'
E40E           ;home drive to track zero
E40E 20 2D E4  home    jsr     stepin            step head in one
E411 20 5B E4          jsr     dly12m            delay 12ms
E414 A9 02     hlp     lda     #%00000010        mask for track zero
E416 2C 00 C0          bit     flpsts            test it
E419 D0 05             bne     nthome            continue if not there
E41B A9 00             lda     #0                clear current
E41D 85 0B             sta     curtrk            track
E41F 60                rts                       else done
E420 20 26 E4  nthome  jsr     stepot            step out
E423 4C 14 E4          jmp     hlp               and loop
E426           ;step towards track zero
E426 AD 02 C0  stepot  lda     flpcnt            get control
E429 09 04             ora     #%00000100        set direction to out
E42B D0 07             bne     step
E42D           ;step away from track zero
E42D E6 0B     stepin  inc     curtrk            bump track
E42F AD 02 C0          lda     flpcnt            get control
```

```
E432 29 FB          and     #%11111011      set direction to in
E434 8D 02 C0  step sta     flpcnt          set it
E437 20 56 E4       jsr     dly12           wait 12 cycles
E43A 29 F7          and     #%11110111      set step bit
E43C 8D 02 C0       sta     flpcnt          set it
E43F 20 57 E4       jsr     dly24           delay 24 cycles
E442 09 08          ora     #%00001000      clear bit
E444 8D 02 C0       sta     flpcnt          set it
E447 A2 05          ldx     #stprte         get rate in ms
E449 4C 5D E4       jmp     dlyxm           delay the right time
E44C            ;delay=20*y+14 cycles
E44C 20 54 E4  dlyy20 jsr   dly15           delay 15 cycles
E44F 88             dey                     drop count
E450 D0 FA          bne     dlyy20          loop if more
E452 EA             nop                     waste time
E453 60             rts
E454            ;delay=15 cycles (if z=0)
E454 D0 00     dly15  bne   *+2
E456            ;delay=12 cycles
E456 60        dly12  rts
E457            ;delay=24 cycles
E457 20 56 E4  dly24  jsr   dly12           do 12
E45A 60             rts
E45B            ;delay=12ms
E45B A2 0C     dly12m ldx   #12
E45D            ;delay=xms
E45D A0 31     dlyxm  ldy   #speed
E45F 20 4C E4       jsr     dlyy20          do 20 cycles
E462 CA             dex
E463 D0 F8          bne     dlyxm           loop if more
E465 60             rts
E466            ;load head and wait 40 ms
E466 A9 7F     loadhd lda   #%01111111      set load bit
E468 2D 02 C0       and     flpcnt          to active
E46B 8D 02 C0       sta     flpcnt
E46E A2 28          ldx     #40             delay 40 ms
E470 4C 5D E4       jmp     dlyxm
E473            ;unload head
E473 A9 80     unldhd lda   #%10000000      set load bit
E475 0D 02 C0       ora     flpcnt          to inactive
E478 8D 02 C0       sta     flpcnt
E47B 60             rts
E47C            ;set up to read track into buffer
E47C            ;if header ok then z=1 else z=0
E47C 20 66 E4  rdytrk jsr   loadhd          load head with settling delay
E47F 78             sei                     disable interrupts
E480 AD 00 C0  fndind lda   flpsts          read status
E483 30 FB          bmi     fndind          not there yet
E485 AD 00 C0  gotind lda   flpsts          read again
E488 10 FB          bpl     gotind          loop while index
E48A A9 03          lda     #%00000011      master reset
E48C 8D 10 C0       sta     flpdta-1
E48F A9 58          lda     #%01011000      no interrupt,rts* high,8+ep+s,/1
E491 8D 10 C0       sta     flpdta-1
E494 AD 00 C0  trkstr lda   flpsts          get status
E497 10 1B          bpl     inderr          error if index
E499 20 B7 E4       jsr     rd1byt          read a byte
E49C C9 43     tryc   cmp   #'C'            see if start code
E49E D0 F4          bne     trkstr          if not keep looking
E4A0 20 B7 E4       jsr     rd1byt          read next byte
E4A3 C9 57          cmp     #'W'            see if second half
E4A5 D0 F5          bne     tryc            if not try for C
E4A7 20 B7 E4       jsr     rd1byt          get another byte
E4AA            ;the following line works because system tracks
E4AA            ;are low numbers and hex = bcd
E4AA C5 0B          cmp     curtrk          see if correct
```

```
E4AC D0 05          bne     rdyext          error if wrong track
E4AE 20 B7 E4       jsr     rd1byt          get next byte
E4B1 C9 58          cmp     #'X'            see if X
E4B3 60       rdyext rts
E4B4 A9 01    inderr lda    #1              say error
E4B6 60              rts
E4B7                ;read a byte from disk into a (ignore parity)
E4B7 AD 10 C0 rd1byt lda    flpdta-1        get acia status
E4BA 4A              lsr    a               check for ready
E4BB 90 FA           bcc    rd1byt          loop if not
E4BD AD 11 C0        lda    flpdta          get byte
E4C0 60              rts
E4C1                ;set trkpnt to first byte of desired sector in track buffer
E4C1 A5 06    strkpt lda    nxtsec          get next sector
E4C3 38              sec                    drop by one
E4C4 E9 01           sbc    #1
E4C6 A0 00           ldy    #0              clear high part of pointer
E4C8 84 08           sty    trkpnt+1
E4CA A0 07           ldy    #7              log2 128
E4CC 0A       mul128 asl    a               multiply
E4CD 26 08           rol    trkpnt+1
E4CF 88              dey
E4D0 D0 FA           bne    mul128          loop til done
E4D2 18              clc                    now add buffer start
E4D3 69 00           adc    #<trkbuf
E4D5 85 07           sta    trkpnt
E4D7 A5 08           lda    trkpnt+1
E4D9 69 03           adc    #>trkbuf
E4DB 85 08           sta    trkpnt+1
E4DD 60              rts
E4DE                ;read track into buffer
E4DE                ; if ok then c=0
E4DE                ; if error then c=1
E4DE A9 00    rdetrk lda    #<trkbuf        point to start of buffer
E4E0 A0 03           ldy    #>trkbuf
E4E2 85 09           sta    rdeadr          set pointer
E4E4 84 0A           sty    rdeadr+1
E4E6 20 7C E4        jsr    rdytrk          get ready to read
E4E9 D0 31           bne    rdeerr          bad header error
E4EB 20 B7 E4 tryv   jsr    rd1byt          get next byte
E4EE C9 76           cmp    #'v'            see if v
E4F0 D0 F9           bne    tryv            loop until is
E4F2 20 B7 E4        jsr    rd1byt          and another
E4F5 C9 01           cmp    #1              if not 1
E4F7 D0 23           bne    rdeerr          is error
E4F9 20 B7 E4        jsr    rd1byt          get track length
E4FC C9 08           cmp    #sectrs/2       compare to correct
E4FE D0 1C           bne    rdeerr          error if wrong
E500 AA              tax                    make a counter
E501 A0 00           ldy    #0              clear index
E503 A9 01    rdelpe lda    #%00000001      get mask for ready
E505 2C 10 C0 tstaca bit    flpdta-1        test acia
E508 F0 FB           beq    tstaca          loop if not ready
E50A AD 11 C0        lda    flpdta          get byte
E50D 70 0D           bvs    rdeerr          parity error
E50F 91 09           sta    (rdeadr),y      put in memory
E511 C8              iny                    bump index
E512 D0 EF           bne    rdelpe          loop if more in page
E514 E6 0A           inc    rdeadr+1        bump pointer
E516 CA              dex                    drop page count
E517 D0 EA           bne    rdelpe          loop if more pages
E519 18              clc                    else done and ok
E51A 90 01           bcc    rdeext          then exit
E51C                ;read error
E51C 38       rdeerr sec
E51D                ;common read exit
```

```
E51D                ;unldhd must not alter c bit
E51D 4C 73 E4       rdeext  jmp     unldhd          un load head
E520                ;data area
E520                *       =       $300
0300                trkbuf  *=      sectrs/2*256+*  track buffer
0B00                        .END



ALLLDE  E3E8    BTELPE  E3A4    CLRSLP  E351    CURTRK  000B
DLY12   E456    DLY12M  E45B    DLY15   E454    DLY24   E457
DLYXM   E45D    DLYY20  E44C    ERRLPE  E3F0    ERRMSG  E408
ERROR   E3EB    FLPCNT  C002    FLPDTA  C011    FLPSTS  C000
FNDIND  E480    FOREVR  E3FB    GETTRK  E3B5    GETV    E38D
GOTIND  E485    GOTTRK  E3BA    HLP     E414    HOME    E40E
INDERR  E4B4    LDEADR  0000    LOADER  E345    LOADHD  E466
MUL128  E4CC    MVESEC  E3BF    NTHOME  E420    NUMTRK  0028
NXTSEC  0006    OPNLPE  E35B    OPNMSG  E3FD    POINTR  0000
RD1BYT  E4B7    RDEADR  0009    RDEERR  E51C    RDEEXT  E51D
RDELPE  E503    RDETRK  E4DE    RDYEXT  E4B3    RDYTRK  E47C
ROMKBD  FD00    SCRCNT  D800    SCREEN  D385    SCRPGS  0004
SECLDE  0004    SECTRK  0005    SECTRS  0010    SIMADR  0002
SPEED   0031    STEP    E434    STEPIN  E42D    STEPOT  E426
STPRTE  0005    STRKPT  E4C1    TRKBUF  0300    TRKPNT  0007
TRKSTR  E494    TRYC    E49C    TRYV    E4EB    TSTACA  E505
TSTFLP  E3A6    UNLDHD  E473
END OF ASSEMBLY
NUMBER OF ERRORS = 0000
```