

How It Works: DOS Floppy Disk Boot Sector

[\[Home\]](#) [\[Q&A\]](#) [\[History\]](#) [\[Help\]](#) [\[SATA\]](#) [\[SCSI/1394\]](#) [\[HowItWorks\]](#) [\[Search\]](#)

[\[Hale's Test Software\]](#) [\[ATACT\]](#) [\[ATADEMO\]](#) [\[ATAMDT\]](#) [\[ATADRVR\]](#) [\[ADVDRVR\]](#)

[\[HIW Topics\]](#) [\[CHSxlat\]](#) [\[DOSboot\]](#) [\[FAQ\]](#) [\[FNF\]](#) [\[MBR\]](#) [\[OS2boot\]](#) [\[PartTable\]](#)

Disassembly of a DOS Floppy Boot Sector

Note: I will leave it to someone else to provide you with a disassembly of an OS/2 HPFS boot sector, or a Linux boot sector, or a WinNT boot sector, etc.

This article is a disassembly of a floppy disk boot sector for a DOS floppy. The boot sector of a floppy disk is located at cylinder 0, head 0, sector 1. This sector is created by a floppy disk formatting program, such as the DOS FORMAT program. The boot sector of a FAT hard disk partition has a similar layout and function. Basically a bootable FAT hard disk partition looks like a big floppy during the early stages of the system's boot processing.

At the completion of your system's Power On Self Test (POST), INT 19 is called. Usually INT 19 tries to read a boot sector from the first floppy drive. If a boot sector is found on the floppy disk, the that boot sector is read into memory at location 0000:7C00 and INT 19 jumps to memory location 0000:7C00. However, if no boot sector is found on the first floppy drive, INT 19 tries to read the MBR from the first hard drive. If an MBR is found it is read into memory at location 0000:7c00 and INT 19 jumps to memory location 0000:7c00. The small program in the MBR will attempt to locate an active (bootable) partition in its partition table. If such a partition is found, the boot sector of that partition is read into memory at location 0000:7C00 and the MBR program jumps to memory location 0000:7C00. Each operating system has its own boot sector format. The small program in the boot sector must locate the first part of the operating system's kernel loader program (or perhaps the kernel itself or perhaps a "boot manager program") and read that into memory.

INT 19 is also called when the CTRL-ALT-DEL keys are used. On most systems, CTRL-ALT-DEL causes an short version of the POST to be executed before INT 19 is called.

- [Where stuff is](#)
- [Summary of what this thing does](#)

- [Entire sector in hex and ASCII](#)
- [The BPB and other data areas](#)
- [Disassembly of the boot sector](#)

Where stuff is

- The BIOS Parameter Block (BPB) starts at offset 0.
- The boot sector program starts at offset 3e.
- The messages issued by this program start at offset 19e.
- The DOS hidden file names start at offset 1e6.
- The boot sector signature is at offset 1fe.

Summary of what this thing does

1. Copy Diskette Parameter Table which is pointed to by INT 1E.
2. Alter the copy of the Diskette Parameter Table.
3. Alter INT 1E to point to altered Diskette Parameter Table.
4. Do INT 13 AH=00, disk reset call.
5. Compute sector address of root directory.
6. Read first sector of root directory into 0000:0500.
7. Confirm that first two directory entries are for IO.SYS and MSDOS.SYS.
8. Read first 3 sectors of IO.SYS into 0000:0700 (or 0070:0000).
9. Leave some information in the registers and jump to IO.SYS at 0070:0000.

Note: This program uses the CHS based INT 13H AH=02 to read the FAT root directory and to read the IO.SYS file. If the drive is >528MB, this CHS must be a translated CHS (or L-CHS, see my BIOS TYPES document). Except for internal computations no addresses in LBA form are used, another reason why LBA does not solve the >528MB problem.

Entire sector in hex and ASCII

```

OFFSET 0 1 2 3 4 5 6 7 8 9 A B C D E F *0123456789ABCDEF*
000000 eb3c904d 53444f53 352e3000 02010100 *...MSDOS5.0.....*
000010 02e00040 0bf00900 12000200 00000000 *...@.....*
000020 00000000 0000295a 5418264e 4f204e41 *.....)ZT..NO NA*
000030 4d452020 20204641 54313220 2020fa33 *ME FAT12 .3*
000040 c08ed0bc 007c1607 bb780036 c5371e56 *.....|...x.6.7.V*
000050 1653bf3e 7cb90b00 fcf3a406 1fc645fe *.S..|.....E.*
000060 0f8b0e18 7c884df9 894702c7 073e7cfb *...|.M..G....|. *
000070 cd137279 33c03906 137c7408 8b0e137c *..ry3.9..|t....| *
000080 890e207c a0107cf7 26167c03 061c7c13 *.. |..|...|...|. *

```

```

000090 161e7c03 060e7c83 d200a350 7c891652 *..|...|....P|..R*
0000a0 7ca3497c 89164b7c b82000f7 26117c8b *|.I|.K|. ....|. *
0000b0 1e0b7c03 c348f7f3 0106497c 83164b7c *..|..H....I|..K|*
0000c0 00bb0005 8b16527c a1507ce8 9200721d *.....R|.P|...r.*
0000d0 b001e8ac 0072168b fbb90b00 bee67df3 *.....r.....}. *
0000e0 a6750a8d 7f20b90b 00f3a674 18be9e7d *.u... ..t...}*
0000f0 e85f0033 c0cd165e 1f8f048f 4402cd19 *_.3...^....D...*
000100 585858eb e88b471a 48488a1e 0d7c32ff *XXX...G.HH...|2.*
000110 f7e30306 497c1316 4b7cbb00 07b90300 *....I|.K|.....*
000120 505251e8 3a0072d8 b001e854 00595a58 *PRQ.:.r....T.YZX*
000130 72bb0501 0083d200 031e0b7c e2e28a2e *r.....|....*
000140 157c8a16 247c8b1e 497ca14b 7cea0000 *.|..$|.I|.K|...*
000150 7000ac0a c07429b4 0ebb0700 cd10ebf2 *p....t).....*
000160 3b16187c 7319f736 187cfec2 88164f7c *;..|s..6.|....O|*
000170 33d2f736 1a7c8816 257ca34d 7cf8c3f9 *3..6.|..%|.M|...*
000180 c3b4028b 164d7cb1 06d2e60a 364f7c8b *.....M|.....6O|. *
000190 ca86e98a 16247c8a 36257ccd 13c30d0a *.....$|.6%|.....*
0001a0 4e6f6e2d 53797374 656d2064 69736b20 *Non-System disk *
0001b0 6f722064 69736b20 6572726f 720d0a52 *or disk error..R*
0001c0 65706c61 63652061 6e642070 72657373 *eplace and press*
0001d0 20616e79 206b6579 20776865 6e207265 * any key when re*
0001e0 6164790d 0a00494f 20202020 20205359 *ady...IO          SY*
0001f0 534d5344 4f532020 20535953 000055aa *SMSDOS          SYS..U.*

```

The BPB and other data areas

The first 62 bytes of a boot sector are known as the BIOS Parameter Block (BPB). Here is the layout of the BPB fields and the values they are assigned in this boot sector:

```

db JMP instruction          at 7c00 size 2 = eb3c
db NOP instruction         7c02      1  90
db OEMname                 7c03      8  'MSDOS5.0'
dw bytesPerSector         7c0b      2  0200
db sectPerCluster         7c0d      1  01
dw reservedSectors       7c0e      2  0001
db numFAT                 7c10      1  02
dw numRootDirEntries     7c11      2  00e0
dw numSectors            7c13      2  0b40 (ignore
numSectorsHuge)
db mediaType              7c15      1  f0
dw numFATsectors         7c16      2  0009

```

```

dw sectorsPerTrack      7c18      2    0012
dw numHeads             7c1a      2    0002
dd numHiddenSectors    7c1c      4    00000000
dd numSectorsHuge      7c20      4    00000000
db driveNum             7c24      1    00
db reserved             7c25      1    00
db signature            7c26      1    29
dd volumeID             7c27      4    5a541826
db volumeLabel          7c2b     11    'NO NAME    '
db fileType             7c36      8    'FAT12    '

```

The first 3 bytes of the BPB are JMP and NOP instructions.

```

0000:7C00 EB3C          JMP      START
0000:7C02 90             NOP

```

Rest of the BPB.

```

0000:7C00 .....4d 53444f53 352e3000 02010100 *   MSDOS5.0.....*
0000:7C10 02e00040 0bf00900 12000200 00000000 *...@.....*
0000:7C20 00000000 0000295a 5418264e 4f204e41 *.....)ZT.&NO NA*
0000:7C30 4d452020 20204641 54313220 2020.... *ME      FAT12      *

```

Now pay attention here...

The 11 bytes starting at 0000:7c3e are immediately overlaid by information copied from another part of memory. That information is the Diskette Parameter Table. This data is pointed to by INT 1E. This data is:

- 7c3e = Step rate and head unload time.
- 7c3f = Head load time and DMA mode flag.
- 7c40 = Delay for motor turn off.
- 7c41 = Bytes per sector.
- 7c42 = Sectors per track.
- 7c43 = Intersector gap length.
- 7c44 = Data length.

- 7c45 = Intersector gap length during format.
- 7c46 = Format byte value.
- 7c47 = Head settling time.
- 7c48 = Delay until motor at normal speed.

The 11 bytes starting at 0000:7c49 are also overlaid by the following data:

- 7c49 - 7c4c = diskette sector address (as LBA) of the data area.
- 7c4d - 7c4e = cylinder number to read from.
- 7c4f - 7c4f = sector number to read from.
- 7c50 - 7c53 = diskette sector address (as LBA) of the root directory.

Disassembly of the boot sector

	START:		START OF BOOT SECTOR
PROGRAM			
0000:7C3E FA	CLI		interrupts off
0000:7C3F 33C0	XOR	AX,AX	set AX to zero
0000:7C41 8ED0	MOV	SS,AX	SS is now zero
0000:7C43 BC007C	MOV	SP,7C00	SP is now 7c00
0000:7C46 16	PUSH	SS	also set ES
0000:7C47 07	POP	ES	to zero
			The INT 1E vector is at 0000:0078.
			Get the address that the vector points
to			into the DS:SI registers.
0000:7C48 BB7800	MOV	BX,0078	BX is now 78
0000:7C4B 36	SS:		
0000:7C4C C537	LDS	SI,[BX]	DS:SI is now [0:78]
0000:7C4E 1E	PUSH	DS	save DS:SI --
0000:7C4F 56	PUSH	SI	saves param tbl
addr			
0000:7C50 16	PUSH	SS	save SS:BX --
0000:7C51 53	PUSH	BX	saves INT 1E
address			
			Move the diskette param table to
0000:7c3e.			

```

0000:7C52 BF3E7C      MOV     DI,7C3E      DI is address of
START
0000:7C55 B90B00      MOV     CX,000B     count is 11
0000:7C58 FC          CLD                clear direction
0000:7C59 F3          REPZ              move the diskette
param
0000:7C5A A4          MOVSB              table to
0000:7c3e
0000:7C5B 06          PUSH    ES         also set DS
0000:7C5C 1F          POP     DS         to zero

                                Alter some of the diskette param table
data.

0000:7C5D C645FE0F     MOV     BYTE PTR [DI-02],0F  change head settle
time
                                at 0000:7c47
0000:7C61 8B0E187C     MOV     CX,[7C18]     sectors per track
0000:7C65 884DF9     MOV     [DI-07],CL    save at
0000:7c42

                                Change INT 1E so that it points to the
                                altered Diskette param table at
0000:7c3e.

0000:7C68 894702     MOV     [BX+02],AX    change INT 1E
segment
0000:7C6B C7073E7C     MOV     WORD PTR [BX],7C3E  change INT 1E
offset

                                Call INT 13 with AX=0000, disk reset, so
                                that the new diskette param table is
used.

0000:7C6F FB          STI                interrupts on
0000:7C70 CD13     INT     13         do diskette reset
call
0000:7C72 7279     JB     TALK        jmp if any error

                                Detemine the starting sector address of
                                the root directory as an LBA.

0000:7C74 33C0     XOR     AX,AX      AX is now zero

```

```

0000:7C76 3906137C  CMP      [7C13],AX      number sectors
zero?
0000:7C7A 7408          JZ       SMALL_DISK     yes
0000:7C7C 8B0E137C     MOV     CX,[7C13]       number of sectors
0000:7C80 890E207C     MOV     [7C20],CX       save in huge num
sects

```

SMALL_DISK:

```

0000:7C84 A0107C      MOV     AL,[7C10]       number of FAT
tables
0000:7C87 F726167C     MUL     WORD PTR [7C16]  number of fat
sectors
0000:7C8B 03061C7C     ADD     AX,[7C1C]       number of hidden
sectors
0000:7C8F 13161E7C     ADC     DX,[7C1E]       number of hidden
sectors
0000:7C93 03060E7C     ADD     AX,[7C0E]       number of reserved
sectors
0000:7C97 83D200      ADC     DX,+00          number of reserved
sectors
0000:7C9A A3507C      MOV     [7C50],AX       save start addr
0000:7C9D 8916527C     MOV     [7C52],DX       of root dir (as
LBA)
0000:7CA1 A3497C      MOV     [7C49],AX       save start addr
0000:7CA4 89164B7C     MOV     [7C4B],DX       of root dir (as
LBA)

```

Determine sector address of first sector
in the data area as an LBA.

```

0000:7CA8 B82000      MOV     AX,0020         size of a dir
entry (32)
0000:7CAB F726117C     MUL     WORD PTR [7C11]  number of root dir
entries
0000:7CAF 8B1E0B7C     MOV     BX,[7C0B]       bytes per sector
0000:7CB3 03C3        ADD     AX,BX
0000:7CB5 48          DEC     AX
0000:7CB6 F7F3        DIV     BX
0000:7CB8 0106497C     ADD     [7C49],AX       add to start addr
0000:7CBC 83164B7C00  ADC     WORD PTR [7C4B],+00 of root dir (as
LBA)

```

```

                                Read the first root dir sector into
0000:0500.

0000:7CC1 BB0005      MOV      BX,0500      addr to read into
0000:7CC4 8B16527C   MOV      DX,[7C52]   get start of
address
0000:7CC8 A1507C      MOV      AX,[7C50]   of root dir (as
LBA)
0000:7CCB E89200      CALL     CONVERT     call conversion
routine
0000:7CCE 721D      JB       TALK        jmp is any error
0000:7CD0 B001      MOV      AL,01      read 1 sector
0000:7CD2 E8AC00      CALL     READ_SECTORS read 1st root dir
sector
0000:7CD5 7216      JB       TALK        jmp if any error
0000:7CD7 8BFB      MOV      DI,BX      addr of 1st dir
entry
0000:7CD9 B90B00      MOV      CX,000B    count is 11
0000:7CDC BEE67D      MOV      SI,7DE6    addr of file names
0000:7CDF F3          REPZ                     is this "IO.SYS"?
0000:7CE0 A6          CMPSB
0000:7CE1 750A      JNZ      TALK        no
0000:7CE3 8D7F20    LEA      DI,[BX+20]  addr of next dir
entry
0000:7CE6 B90B00      MOV      CX,000B    count is 11
0000:7CE9 F3          REPZ                     is this "MSDOS.
SYS"?
0000:7CEA A6          CMPSB
0000:7CEB 7418      JZ       FOUND_FILES they are equal

```

TALK:

Display "Non-System disk..." message,
wait for user to hit a key, restore
the INT 1E vector and then
call INT 19 to start boot processing
all over again.

```

0000:7CED BE9E7D      MOV      SI,7D9E    "Non-System
disk..."
0000:7CF0 E85F00      CALL     MSG_LOOP   display message
0000:7CF3 33C0      XOR      AX,AX      INT 16 function
0000:7CF5 CD16      INT      16        read keyboard

```

```

0000:7CF7 5E          POP     SI          get INT 1E vector's
0000:7CF8 1F          POP     DS          address
0000:7CF9 8F04         POP     [SI]        restore the INT 1E
0000:7CFB 8F4402       POP     [SI+02]     vector's data
0000:7CFE CD19         INT     19         CALL INT 19 to try
again

```

SETUP_TALK:

```

0000:7D00 58          POP     AX          pop junk off stack
0000:7D01 58          POP     AX          pop junk off stack
0000:7D02 58          POP     AX          pop junk off stack
0000:7D03 EBE8        JMP     TALK        now talk to the
user

```

FOUND_FILES:

Compute the sector address of the first sector of IO.SYS.

```

0000:7D05 8B471A       MOV     AX,[BX+1A]  get starting
cluster num
0000:7D08 48          DEC     AX          subtract 1
0000:7D09 48          DEC     AX          subtract 1
0000:7D0A 8A1E0D7C    MOV     BL,[7C0D]  sectors per cluster
0000:7D0E 32FF        XOR     BH,BH
0000:7D10 F7E3        MUL     BX          multiply
0000:7D12 0306497C    ADD     AX,[7C49]  add start addr of
0000:7D16 13164B7C    ADC     DX,[7C4B]  root dir (as
LBA)

```

Read IO.SYS into memory at 0000:0700.

IO.SYS

is 3 sectors long.

```

0000:7D1A BB0007       MOV     BX,0700    address to read
into
0000:7D1D B90300       MOV     CX,0003    read 3 sectors

```

READ_LOOP:

Read the first 3 sectors of IO.SYS (IO.SYS is much longer than 3 sectors).

```

0000:7D20 50          PUSH    AX          save AX
0000:7D21 52          PUSH    DX          save DX
0000:7D22 51          PUSH    CX          save CX
0000:7D23 E83A00        CALL    CONVERT     call conversion
routine
0000:7D26 72D8          JB     SETUP_TALK   jmp if error
0000:7D28 B001          MOV    AL,01        read one sector
0000:7D2A E85400        CALL    READ_SECTORS read one sector
0000:7D2D 59          POP     CX          restore CX
0000:7D2E 5A          POP     DX          restore DX
0000:7D2F 58          POP     AX          restore AX
0000:7D30 72BB          JB     TALK         jmp if any INT 13
error
0000:7D32 050100        ADD    AX,0001      add one to the
sector addr
0000:7D35 83D200        ADC    DX,+00       add one to the
sector addr
0000:7D38 031E0B7C      ADD    BX,[7C0B]    incr mem addr by
sect size
0000:7D3C E2E2          LOOP   READ_LOOP    read next sector

                                Leave information in the AX, BX, CX and
DX

                                registers for IO.SYS to use. Finally,
                                jump to IO.SYS at 0070:0000.

0000:7D3E 8A2E157C      MOV    CH,[7C15]    media type
0000:7D42 8A16247C      MOV    DL,[7C24]    drive number
0000:7D46 8B1E497C      MOV    BX,[7C49]    get start addr of
0000:7D4A A14B7C        MOV    AX,[7C4B]    root dir (as
LBA)
0000:7D4D EA00007000    JMP    0070:0000    JUMP TO 0070:0000

                                MSG_LOOP:

                                This routine displays a message using
                                INT 10 one character at a time.
                                The message address is in DS:SI.

0000:7D52 AC          LODSB             get message
character
0000:7D53 0AC0          OR     AL,AL       end of message?

```

```

0000:7D55 7429      JZ      RETURN      jmp if yes
0000:7D57 B40E      MOV     AH,0E       display one
character
0000:7D59 BB0700    MOV     BX,0007     video attrbiutes
0000:7D5C CD10      INT     10          display one
character
0000:7D5E EBF2      JMP     MSG_LOOP     do again

```

CONVERT:

This routine converts a sector address (an LBA) to a CHS address. The LBA is in DX:AX.

```

0000:7D60 3B16187C  CMP     DX,[7C18]   hi part of LBA >
sectPerTrk?
0000:7D64 7319      JNB     SET_CARRY   jmp if yes
0000:7D66 F736187C  DIV     WORD PTR [7C18] div by sectors per
track
0000:7D6A FEC2      INC     DL           add 1 to sector
number
0000:7D6C 88164F7C  MOV     [7C4F],DL   save sector number
0000:7D70 33D2      XOR     DX,DX       zero DX
0000:7D72 F7361A7C  DIV     WORD PTR [7C1A] div number of heads
0000:7D76 8816257C  MOV     [7C25],DL   save head number
0000:7D7A A34D7C    MOV     [7C4D],AX   save cylinder
number
0000:7D7D F8        CLC                    clear carry
0000:7D7E C3        RET                    return

```

SET_CARRY:

```

0000:7D7F F9        STC                    set carry

```

RETURN:

```

0000:7D80 C3        RET                    return

```

READ_SECTORS:

The caller of this routine supplies:
 AL = number of sectors to read
 ES:BX = memory location to read into
 and CHS address to read from in

memory locations 7c25 and 7C4d-7c4f.

```

0000:7D81 B402      MOV     AH,02      INT 13 read sectors
0000:7D83 8B164D7C   MOV     DX,[7C4D]  get cylinder number
0000:7D87 B106      MOV     CL,06     shift count
0000:7D89 D2E6      SHL     DH,CL     shift upper cyl
left 6 bits
0000:7D8B 0A364F7C   OR      DH,[7C4F]  or in sector number
0000:7D8F 8BCA      MOV     CX,DX     move to CX
0000:7D91 86E9      XCHG   CH,CL     CH=cyl lo, CL=cyl
hi + sect
0000:7D93 8A16247C   MOV     DL,[7C24]  drive number
0000:7D97 8A36257C   MOV     DH,[7C25]  head number
0000:7D9B CD13      INT     13        read sectors
0000:7D9D C3        RET              return

```

Data not used.

```
0000:7D90 ca86e98a 16247c8a 36257ccd 13c3.... *.....$|.6%|... *
```

Messages here.

```

0000:7D90 .....0d0a * ..*
0000:7Da0 4e6f6e2d 53797374 656d2064 69736b20 *Non-System disk *
0000:7Db0 6f722064 69736b20 6572726f 720d0a52 *or disk error..R*
0000:7Dc0 65706c61 63652061 6e642070 72657373 *eplace and press*
0000:7Dd0 20616e79 206b6579 20776865 6e207265 * any key when re*
0000:7De0 6164790d 0a00.... *ady... *

```

MS DOS hidden file names (first two root directory entries).

```

0000:7De0 .....494f 20202020 20205359 * IO SY*
0000:7Df0 534d5344 4f532020 20535953 000055aa *SMSDOS SYS..U.*

```

The last two bytes contain a 55AAH signature.

```
0000:7Df0 .....55aa * U.*
```

This page was last updated on 06 May 2002.