

OS2 Boot Sector

by Hale Landis (landis@sugs.tware.com)

<http://www.GDR.w.pl/>

OS2 BOOT SECTOR

Note: I'll 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 or hard disk boot sector for OS/2. Apparently OS/2 uses the same boot sector for both environments. Basically a bootable FAT hard disk partition looks like a big floppy during the early stages of the system's boot processing. This sector is at cylinder 0, head 0, sector 1 of a floppy or it is the first sector within a FAT hard disk partition. OS/2 floppy disk and hard disk boot sectors are created by the OS/2 FORMAT program.

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.

OFFSET	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	*0123456789ABCDEF*
000000	eb449049				424d2032				302e3000				02100100				*.D.IBM 20.0.....*
000010	02000200				00f8d800				3e000e00				3e000000				*.....>...>...*
000020	06780d00				80002900				1c0c234e				4f204e41				*.x.....)....#NO NA*
000030	4d452020				20204641				54202020				20200000				*ME FAT ..*
000040	00100000				0000fa33				db8ed3bc				ff7bfbbba				*.....3.....{...*
000050	c0078eda				803e2400				00753d1e				b840008e				*.....>\$..u=..@...*
000060	c026ff0e				1300cd12				c1e0068e				c033ff33				*.&.....3.3*
000070	c08ed8c5				367800fc				b90b00f3				a41fa118				*....6x.....*
000080	0026a204				001e33c0				8ed8a378				008c067a				*.&....3....x...z*
000090	001f8a16				2400cd13				a0100098				f7261600				*....\$.&...*
0000a0	03060e00				5091b820				00f72611				008b1e0b				*....P...&...*

```

0000b0 0003c348 f7f35003 c1a33e00 b800108e *...H..P...>.....*
0000c0 c033ff59 890e4400 58a34200 33d2e873 *.3.Y..D.X.B.3..s*
0000d0 0033db8b 0e11008b fb51b90b 00bed501 *.3.....Q.....*
0000e0 f3a65974 0583c320 e2ede335 268b471c *..Yt... ..5&.G.*
0000f0 268b571e f7360b00 fec08ac8 268b571a *&.W..6.....&.W.*
000100 4a4aa00d 0032e4f7 e203063e 0083d200 *JJ...2.....>....*
000110 bb00088e c333ff06 57e82800 8d360b00 *.....3..W.(..6..*
000120 cbbe9801 eb03bead 01e80900 bec201e8 *.....*
000130 0300fbef feac0ac0 7409b40e bb0700cd *.....t.....*
000140 10ebf2c3 50525103 061c0013 161e00f7 *....PRQ.....*
000150 361800fe c28ada33 d2f7361a 008afa8b *6.....3..6.....*
000160 d0a11800 2ac34050 b402b106 d2e60af3 *.....*..@P.....*
000170 8bca86e9 8a162400 8af78bdf cd1372a6 *.....$......r.*
000180 5b598bc3 f7260b00 03f85a58 03c383d2 *[Y...&....ZX....*
000190 002acb7f afc31200 4f532f32 20212120 *. *.....OS/2 !! *
0001a0 53595330 31343735 0d0a0012 004f532f *SYS01475.....OS/*
0001b0 32202121 20535953 30323032 350d0a00 *2 !! SYS02025...*
0001c0 12004f53 2f322021 21205359 53303230 *..OS/2 !! SYS020*
0001d0 32370d0a 004f5332 424f4f54 20202020 *27...OS2BOOT *
0001e0 00000000 00000000 00000000 00000000 *.....*
0001f0 00000000 00000000 00000000 000055aa *.....U.*

```

=====

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	= eb44
db NOP instruction	7c02	1	90
db OEMname	7c03	8	'IBM 20.0'
dw bytesPerSector	7c0b	2	0200
db sectPerCluster	7c0d	1	01
dw reservedSectors	7c0e	2	0001
db numFAT	7c10	1	02
dw numRootDirEntries	7c11	2	0200
dw numSectors	7c13	2	0000 (use
numSectorsHuge)			
db mediaType	7c15	1	f8
dw numFATsectors	7c16	2	00d8
dw sectorsPerTrack	7c18	2	003e
dw numHeads	7c1a	2	000e
dd numHiddenSectors	7c1c	4	00000000
dd numSectorsHuge	7c20	4	000d7806
db driveNum	7c24	1	80
db reserved	7c25	1	00
db signature	7c26	1	29
dd volumeID	7c27	4	001c0c23
db volumeLabel	7c2b	11	'NO NAME'
db fileSysType	7c36	8	'FAT'

=====

Here is the boot sector...

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

```
0000:7C00 EB44          JMP      START
0000:7C02 90             NOP
```

Here is the rest of the BPB.

```
0000:7C00 eb449049 424d2032 302e3000 02100100 *.D.IBM 20.0.....*
0000:7C10 02000200 00f8d800 3e000e00 3e000000 *.....>...>...*
0000:7C20 06780d00 80002900 1c0c234e 4f204e41 *.x....)....#NO NA*
0000:7C30 4d452020 20204641 54202020 20200000 *ME    FAT    ..*
```

Additional data areas.

```
0000:7C30 ..... 0000 *          ..*
0000:7C40 00100000 0000.... *          *
```

Note:

0000:7c3e (DS:003e) = number of sectors in the FATs and root dir.

0000:7c42 (DS:0042) = number of sectors in the FAT.

0000:7c44 (DS:0044) = number of sectors in the root dir.

START:

START OF BOOT SECTOR

PROGRAM

```
0000:7C46 FA          CLI             interrupts off
0000:7C47 33DB        XOR      BX,BX      zero BX
0000:7C49 8ED3        MOV      SS,BX      SS now zero
0000:7C4B BCFF7B      MOV      SP,7BFF    SP now 7bff
0000:7C4E FB          STI             interrupts on
0000:7C4F BAC007      MOV      DX,07C0    set DX to
0000:7C52 8EDA        MOV      DS,DX      07c0
```

Are we booting from a floppy or a hard disk partition?

```
0000:7C54 803E240000 CMP      BYTE PTR [0024],00    is driveNum in
BPB 00?
0000:7C59 753D        JNZ      NOT_FLOPPY      jmp if not zero
```

We are booting from a floppy. The Diskette Parameter Table must be copied and altered...

Diskette Parameter Table is pointed to by INT 1E. This program moves this table to high memory, alters the table, and changes INT 1E to point to the altered table.

This table contains the following data:

???? : 0000 = Step rate and head unload time.
 ???? : 0001 = Head load time and DMA mode flag.
 ???? : 0002 = Delay for motor turn off.
 ???? : 0003 = Bytes per sector.
 ???? : 0004 = Sectors per track.
 ???? : 0005 = Intersector gap length.
 ???? : 0006 = Data length.
 ???? : 0007 = Intersector gap length during format.
 ???? : 0008 = Format byte value.
 ???? : 0009 = Head settling time.
 ???? : 000a = Delay until motor at normal speed.

Compute a valid high memory address.

0000:7C5B 1E	PUSH	DS	save DS
0000:7C5C B84000	MOV	AX,0040	set ES
0000:7C5F 8EC0	MOV	ES,AX	to 0040
(BIOS data area)			
0000:7C61 26	ES:		reduce system
memory			
0000:7C62 FF0E1300	DEC	WORD PTR [0013]	size by 1024
0000:7C66 CD12	INT	12	get system
memory size			
0000:7C68 C1E06	SHL	AX,06	shift AX (mult
by 64)			
0000:7C6B 8EC0	MOV	ES,AX	move to ES
0000:7C6D 33FF	XOR	DI,DI	zero DI

Move the diskette param table to high memory.

0000:7C6F 33C0	XOR	AX,AX	zero AX
0000:7C71 8ED8	MOV	DS,AX	DS now zero
0000:7C73 C5367800	LDS	SI,[0078]	DS:SI = INT 1E
vector			
0000:7C77 FC	CLD		clear direction
0000:7C78 B90B00	MOV	CX,000B	count is 11
0000:7C7B F3	REPZ		copy diskette
param table			
0000:7C7C A4	MOVSB		to top of
memory			

Alter the number of sectors per track in the diskette param table in high memory.

0000:7C7D 1F	POP	DS	restore DS
0000:7C7E A11800	MOV	AX,[0018]	get
sectorsPerTrack from BPB			
0000:7C81 26	ES:		alter sectors
per track			

0000:7C82 A20400	MOV	[0004],AL	in diskette
param table			

diskette

Change INT 1E to point to altered
param table and do a INT 13 disk
reset call.

0000:7C85 1E	PUSH	DS	save DS
0000:7C86 33C0	XOR	AX,AX	AX now zero
0000:7C88 8ED8	MOV	DS,AX	DS no zero
0000:7C8A A37800	MOV	[0078],AX	alter INT 1E
vector			
0000:7C8D 8C067A00	MOV	[007A],ES	to point to
altered			

diskette

param table

0000:7C91 1F	POP	DS	restore DS
0000:7C92 8A162400	MOV	DL,[0024]	driveNum from
BPB			
0000:7C96 CD13	INT	13	diskette reset

NOT_FLOPPY:

Compute the location and the size of
the root directory. Read the entire
root directory into memory.

0000:7C98 A01000	MOV	AL,[0010]	get numFAT
0000:7C9B 98	CBW		make into a
word			
0000:7C9C F7261600	MUL	WORD PTR [0016]	mult by
numFatSectors			
0000:7CA0 03060E00	ADD	AX,[000E]	add
reservedSectors			
0000:7CA4 50	PUSH	AX	save
0000:7CA5 91	XCHG	CX,AX	move to CX
0000:7CA6 B82000	MOV	AX,0020	dir entry size
0000:7CA9 F7261100	MUL	WORD PTR [0011]	mult by
numRootDirEntries			
0000:7CAD 8B1E0B00	MOV	BX,[000B]	get
bytesPerSector			
0000:7CB1 03C3	ADD	AX,BX	add
0000:7CB3 48	DEC	AX	subtract 1
0000:7CB4 F7F3	DIV	BX	div by
bytesPerSector			
0000:7CB6 50	PUSH	AX	save number of
dir sectors			
0000:7CB7 03C1	ADD	AX,CX	add number of
fat sectors			
0000:7CB9 A33E00	MOV	[003E],AX	save
0000:7CBC B80010	MOV	AX,1000	AX is now 1000
0000:7CBF 8EC0	MOV	ES,AX	ES is now 1000

0000:7CC1 33FF	XOR	DI,DI	DI is now zero
0000:7CC3 59	POP	CX	get number dir
sectors			
0000:7CC4 890E4400	MOV	[0044],CX	save
0000:7CC8 58	POP	AX	get number fat
sectors			
0000:7CC9 A34200	MOV	[0042],AX	save
0000:7CCC 33D2	XOR	DX,DX	DX now zero
0000:7CCE E87300	CALL	READ_SECTOR	read 1st sect
of root dir			
0000:7CD1 33DB	XOR	BX,BX	BX is now zero
0000:7CD3 8B0E1100	MOV	CX,[0011]	number of root
dir entries			

DIR_SEARCH:

SEARCH FOR OS2BOOT.

file
Search the root directory for the
name OS2BOOT.

0000:7CD7 8BFB	MOV	DI,BX	DI is dir entry
addr			
0000:7CD9 51	PUSH	CX	save CX
0000:7CDA B90B00	MOV	CX,000B	count is 11
0000:7CDD BED501	MOV	SI,01D5	addr of
"OS2BOOT"			
0000:7CE0 F3	REPZ		is 1st dir
entry			
0000:7CE1 A6	CMPSB		for
"OS2BOOT"?			
0000:7CE2 59	POP	CX	restore CX
0000:7CE3 7405	JZ	FOUND_OS2BOOT	jmp if OS2BOOT
0000:7CE5 83C320	ADD	BX,+20	incr to next
dir entry			
0000:7CE8 E2ED	LOOP	DIR_SEARCH	try again

FOUND_OS2BOOT:

FOUND OS2BOOT.

sector
OS2BOOT was found. Get the starting
cluster number and convert to a
address. Read OS2BOOT into memory
and
finally do a far return to enter
the OS2BOOT program.

0000:7CEA E335	JCXZ	FAILED1	JMP if CX zero.
0000:7CEC 26	ES:		get the size of
0000:7CED 8B471C	MOV	AX,[BX+1C]	the OS2BOOT
file			
0000:7CF0 26	ES:		from the
OS2BOOT			
0000:7CF1 8B571E	MOV	DX,[BX+1E]	directory

entry				
0000:7CF4 F7360B00	DIV	WORD PTR [000B]	div by	
bytesPerSect				
0000:7CF8 FEC0	INC	AL	add 1	
0000:7CFA 8AC8	MOV	CL,AL	num sectors	
OS2BOOT				
0000:7CFC 26	ES:		get the	
starting				
0000:7CFD 8B571A	MOV	DX,[BX+1A]	cluster	
number				
0000:7D00 4A	DEC	DX	subtract 1	
0000:7D01 4A	DEC	DX	subtract 1	
0000:7D02 A00D00	MOV	AL,[000D]		
sectorsPerCluster				
0000:7D05 32E4	XOR	AH,AH	mutiply	
0000:7D07 F7E2	MUL	DX	to get LBA	
0000:7D09 03063E00	ADD	AX,[003E]	add number of	
FAT sectors				
0000:7D0D 83D200	ADC	DX,+00	to LBA	
0000:7D10 BB0008	MOV	BX,0800	set ES	
0000:7D13 8EC3	MOV	ES,BX	to 0800	
0000:7D15 33FF	XOR	DI,DI	set ES:DI to	
entry point				
0000:7D17 06	PUSH	ES	address of	
0000:7D18 57	PUSH	DI	OS2BOOT	
0000:7D19 E82800	CALL	READ_SECTOR	read OS2BOOT	
into memory				
0000:7D1C 8D360B00	LEA	SI,[000B]	set DS:SI	
0000:7D20 CB	RETF		"far return" to	
OS2BOOT				

FAILED1:	OS2BOOT WAS NOT
FOUND.	

0000:7D21 BE9801	MOV	SI,0198	"SYS01475"
message			
0000:7D24 EB03	JMP	FAILED3	

FAILED2:	ERROR FROM INT 13.
----------	--------------------

0000:7D26 BEAD01	MOV	SI,01AD	"SYS02025"
message			

FAILED3:	OUTPUT ERROR
MESSAGES.	

0000:7D29 E80900	CALL	MSG_LOOP	display message
0000:7D2C BEC201	MOV	SI,01C2	"SYS02027"
message			
0000:7D2F E80300	CALL	MSG_LOOP	display message
0000:7D32 FB	STI		interrupts on

HANG:	HANG THE SYSTEM!
-------	------------------

0000:7D33	EBFE	JMP	HANG	sit and stay!
MSG_LOOP:		DISPLAY AN ERROR		
MESSAGE.				
Routine to display the message text pointed to by SI.				
0000:7D35	AC	LODSB		get next char of message
0000:7D36	0AC0	OR	AL,AL	end of message?
0000:7D38	7409	JZ	RETURN	jmp if yes
0000:7D3A	B40E	MOV	AH,0E	write 1 char
0000:7D3C	BB0700	MOV	BX,0007	video attributes
0000:7D3F	CD10	INT	10	INT 10 to write 1 char
0000:7D41	EBF2	JMP	MSG_LOOP	do again
RETURN:				
0000:7D43	C3	RET		return
READ_SECTOR:		ROUTINE TO READ		
SECTORS.				
Read sectors into memory. Read multiple sectors but don't read across a track boundary.				
The caller supplies the following:				
DX:AX = sector address to read (as LBA)				
CX = number of sectors to read				
ES:DI = memory address to read into				
0000:7D44	50	PUSH	AX	save lower part of LBA
0000:7D45	52	PUSH	DX	save upper part of LBA
0000:7D46	51	PUSH	CX	save number of sect to read
0000:7D47	03061C00	ADD	AX,[001C]	add numHiddenSectors
0000:7D4B	13161E00	ADC	DX,[001E]	to LBA
0000:7D4F	F7361800	DIV	WORD PTR [0018]	div by sectorsPerTrack
0000:7D53	FEC2	INC	DL	add 1 to sector number

0000:7D55 8ADA	MOV	BL,DL	save sector
number			
0000:7D57 33D2	XOR	DX,DX	zero upper part
of LBA			
0000:7D59 F7361A00	DIV	WORD PTR [001A]	div by numHeads
0000:7D5D 8AFA	MOV	BH,DL	save head
number			
0000:7D5F 8BD0	MOV	DX,AX	save cylinder
number			
0000:7D61 A11800	MOV	AX,[0018]	sectorsPerTrack
0000:7D64 2AC3	SUB	AL,BL	sub sector
number			
0000:7D66 40	INC	AX	add 1
0000:7D67 50	PUSH	AX	save number of
sector to read			
0000:7D68 B402	MOV	AH,02	INT 13 read
sectors			
0000:7D6A B106	MOV	CL,06	shift count
0000:7D6C D2E6	SHL	DH,CL	shift high cyl
left			
0000:7D6E 0AF3	OR	DH,BL	or in sector
number			
0000:7D70 8BCA	MOV	CX,DX	move cyl/sect
to CX			
0000:7D72 86E9	XCHG	CH,CL	swap cyl/sect
0000:7D74 8A162400	MOV	DL,[0024]	driveNum
0000:7D78 8AF7	MOV	DH,BH	head number
0000:7D7A 8BDF	MOV	BX,DI	memory addr to
read into			
0000:7D7C CD13	INT	13	INT 13 read
sectors call			
0000:7D7E 72A6	JB	FAILED2	jmp if any
error			
0000:7D80 5B	POP	BX	get number of
sectors read			
0000:7D81 59	POP	CX	restore CX
0000:7D82 8BC3	MOV	AX,BX	number of
sector to AX			
0000:7D84 F7260B00	MUL	WORD PTR [000B]	multiply by
sector size			
0000:7D88 03F8	ADD	DI,AX	add to memory
address			
0000:7D8A 5A	POP	DX	restore upper
part of LBA			
0000:7D8B 58	POP	AX	restore lower
part of LBA			
0000:7D8C 03C3	ADD	AX,BX	add number of
sector just			
0000:7D8E 83D200	ADC	DX,+00	read to LBA
0000:7D91 2ACB	SUB	CL,BL	decr requested
num of sect			
0000:7D93 7FAF	JG	READ_SECTOR	jmp if not zero

0000:7D95 C3 RET return

Data not used.

0000:7D90 1200 * .. *

Messages here.

0000:7D90 4f532f32 20212120 * OS/2 !! *
0000:7Da0 53595330 31343735 0d0a0012 004f532f *SYS01475.....OS/*
0000:7Db0 32202121 20535953 30323032 350d0a00 *2 !! SYS02025...*
0000:7Dc0 12004f53 2f322021 21205359 53303230 *..OS/2 !! SYS020*
0000:7Dd0 32370d0a 00..... *27.... *

OS/2 loader file name.

0000:7Dd04f5332 424f4f54 20202020 * OS2BOOT *

Data not used.

0000:7De0 00000000 00000000 00000000 00000000 *.....*
0000:7Df0 00000000 00000000 00000000 0000..... *..... *

The last two bytes contain a 55AAH signature.

0000:7Df0 55aa * U.*