INT 13H

From Wikipedia, the free encyclopedia

INT 13h is shorthand for BIOS interrupt call 13_{hex}, the 20th interrupt vector in an x86-based computer system. The BIOS typically sets up a real mode interrupt handler at this vector that provides sector-based hard disk and floppy disk read and write services using cylinder-head-sector (CHS) addressing.

INT is an x86 instruction that triggers a software interrupt, and 13_{hex} is the interrupt number (as a hexadecimal value) being called.

Contents

Overview

Under real mode operating systems, such as MS-DOS, calling INT 13h would jump into the computer's ROM-BIOS code for low-level disk services, which would carry out physical sector-based disk read or write operations for the program. In MS-DOS, it serves as the low-level interface for the built-in block device drivers for hard disks and floppy disks. This allows INT 25h and INT 26h to provide absolute disk read/write functions for logical sectors to the FAT file system driver in the DOS kernel, which handles file-related requests through MS-DOS API (INT 21h) functions.

Under protected mode operating systems, such as Microsoft Windows NT derivatives (e.g. NT4, 2000, XP, and Server 2003) and Linux with dosemu, the OS intercepts the call and passes it to the operating system's native disk I/O mechanism. Windows 9x and Windows for Workgroups 3.11 also bypass BIOS routines when using 32-bit File Access.

The original BIOS real-mode INT 13h interface supports drives of sizes up to about 504 MB using what is commonly referred to as physical CHS addressing. This limit originates from the hardware interface of the IBM PC/XT disk hardware. The BIOS used the cylinder-head-sector (CHS) address given in the INT 13h call, and transferred it directly to the hardware interface.

This interface was later extended to support addressing of up to exactly 8064 MB using what is commonly referred to as logical CHS addressing. This limit originates from a combination of the register value based calling convention used in the INT 13h interface, and the goal of maintaining backward compatibility. There were originally a number of BIOSes that offered incompatible versions of this interface, but eventually the computer industry standardized on the interface developed in the Microid Research ("MR BIOS") in 1989. This limit uses 1024 cylinders, 256 heads, 63 sectors, and 512 byte blocks, allowing exactly 7.875 GiB of addressing (1024 * 256 * 63 * 512 bytes).

To support even larger addressing modes, an interface known as INT 13h Extensions was introduced by Western Digital and Phoenix Technologies as part of BIOS Enhanced Disk Drive Services (EDD).[1][2] It uses 64-bit logical block addressing (LBA) which allows addressing up to 8 ZiB (the drive can also support 28-bit or 48-bit LBA which allows up to 128 GiB or 128 PiB respectively, assuming a 512-byte sector/block size). This is a packet interface, because it uses a pointer to a packet of information rather than the register based calling convention of the original INT 13h interface. This packet is a data structure that contains an interface version, data size, and LBAs.

"[Most] versions of MS-DOS, (including MS-DOS 7, and Windows 95) have a bug which prevents booting hard disks with 256 heads (register value 0xFF), so many modern BIOSes provide mappings with at most 255 (FEh) heads",[3][4] thus reducing the total addressable space to exactly 8032.5 MiB (approx 7.844 GiB).[5] Some cache drivers flush their buffers when detecting that MS-DOS is bypassed by directly issuing INT 13h from applications. A dummy read can be used as one of several methods to force cache flushing for unknown caches (e.g. before rebooting).[3][4] AMI BIOSes from around 1990-1991 trash word unaligned buffers. Some MS-DOS and TSR programs clobber interrupt enabling and registers so PC DOS and MS-DOS install their own filters to prevent this.[6]

AWARD AT BIOS and AMI 386sx BIOS have been extended to handle more than 1024 cylinders by placing bits 10 and 11 of the cylinder number into bits 6 and 7 of DH

List of INT 13h services

Drive Table

DL = 00h	1st floppy disk ("drive A:")
DL = 01h	2nd floppy disk ("drive B:")
DL = 80h	1st hard disk
DL = 81h	2nd hard disk

Function Table

AH = 00h		Reset Disk Drives
AH = 01h		Get Status of Last Drive Operation
AH = 02h		Read Sectors From Drive
AH = 03h		Write Sectors To Drive
AH = 04h		Verify Sectors
AH = 05h		Format Track
AH = 06h		Format Track Set Bad Sector Flags
AH = 07h		Format Drive starting at Track
AH = 08h		Read Drive Parameters
AH = 09h	HD	Initialize Disk Controller
AH = 0Ah	HD	Read Long Sectors From Drive
AH = 0Bh	HD	Write Long Sectors To Drive
AH = 0Ch	HD	Move Drive Head To Cylinder
AH = 0Dh	HD	Reset Disk Drives
AH = 0Eh	PS/2	Controller Read Test
AH = 0Fh	PS/2	Controller Write Test
AH = 10h	HD	Test Whether Drive Is Ready
AH = 11h	HD	Recalibrate Drive
AH = 12h	PS/2	Controller RAM Test
AH = 13h	PS/2	Drive Test
AH = 14h	HD	Controller Diagnostic
AH = 15h		Read Drive Type
AH = 16h	FD	Detect Media Change
AH = 17h	FD	Set Media Type For Format (used by DOS versions <= 3.1)
AH = 18h	FD	Set Media Type For Format (used by DOS versions >= 3.2)
AH = 19h		Park Heads
AH = 41h	EXT	Test Whether Extensions Are Available
AH = 42h	EXT	Read Sectors From Drive
AH = 43h	EXT	Write Sectors To Drive

AH = 44h	EXT	Verify Sectors
AH = 45h	EXT	Lock/Unlock Drive
AH = 46h	EXT	Eject Drive
AH = 47h	EXT	Move Drive Head To Sector
AH = 48h	EXT	Read Drive Parameters
AH = 49h	EXT	Detect Media Change

If the second column is empty then the function may be used both for floppy and hard disk.

- FD: for floppy disk only.
- HD: for hard disk only.
- PS/2: for hard disk on PS/2 system only.
- EXT: part of the Int 13h Extensions which were written in the 1990s to support hard drives with more than 8 GB.

INT 13h AH=00h: Reset Disk Drive

Parameters:

AH	00h
DL	Drive

Results:

CF	Set on error

INT 13h AH=01h: Get Status of Last Drive Operation

Parameters:

AH	01h
DL	Drive

Bit 7=0 for floppy drive, bit 7=1 for fixed drive

Results:

AH	Return Code
	00h Success
	01h Invalid Command
	02h Cannot Find Address Mark
	03h Attempted Write On Write Protected Disk
	04h Sector Not Found
	05h Reset Failed
	06h Disk change line 'active'
	07h Drive parameter activity failed
	08h DMA overrun
	09h Attempt to DMA over 64kb boundary
	0Ah Bad sector detected
	0Bh Bad cylinder (track) detected
	0Ch Media type not found
	0Dh Invalid number of sectors
	0Eh Control data address mark detected
	0Fh DMA out of range
	10h CRC/ECC data error
	11h ECC corrected data error
	20h Controller failure
	40h Seek failure
	80h Drive timed out, assumed not ready
	AAh Drive not ready
	BBh Undefined error
	CCh Write fault
	E0h Status error
	FFh Sense operation failed
CF	Set On Error, Clear If No Error

INT 13h AH=02h: Read Sectors From Drive

Parameters:

AH	02h
AL	Sectors To Read Count
CH	Cylinder
CL	Sector
DH	Head
DL	Drive
ES:BX	Buffer Address Pointer

Results:

CF	Set On Error, Clear If No Error
AH	Return Code
AL	Actual Sectors Read Count

Remarks:
Register CX contains both the cylinder number (10 bits, possible values are 0 to 1023) and the sector number (6 bits, possible values are 1 to 63). Cylinder and Sector bits are numbered below:

```
CX =          ---CH--- ---CL---
cylinder : 76543210 98
sector   :            543210
```

Examples of translation:

```
CX := ( ( cylinder and 255 ) shl 8 ) or ( ( cylinder and 768 ) shr 2 ) or sector;
cylinder := ( (CX and 0xFF00) shr 8 ) or ( (CX and 0xC0) shl 2)
sector := CX and 63;
```

Addressing of Buffer should guarantee that the complete buffer is inside the given segment, i.e. (BX + size_of_buffer) <= 10000h. Otherwise the interrupt may fail with some BIOS or hardware versions.
Example: Assume you want to read 16 sectors (= 2000h bytes) and your buffer starts at memory address 4FF00h. Utilizing memory segmentation, there are different ways to calculate the register values, e.g.:

```
ES = segment      = 4F00h
```

```
BX = offset         =  0F00h
sum = memory address = 4FF00h
would be a good choice because 0F00h + 2000h = 2F00h <= 10000h
ES = segment        = 4000h
BX = offset         =  FF00h
sum = memory address = 4FF00h
would not be a good choice because FF00h + 2000h = 11F00h > 10000h
```

Function 02h of interrupt 13h may only read sectors of the first 16,450,560 sectors of your hard drive, to read sectors beyond the 8 GB limit you should use function 42h of Int 13h Extensions. Another alternate may be DOS interrupt 25h which reads sectors within a partition.

INT 13h AH=03h: Write Sectors To Drive

Parameters:

AH	03h
AL	Sectors To Write Count
CH	Track
CL	Sector
DH	Head
DL	Drive
ES:BX	Buffer Address Pointer

Results:

CF	Set On Error, Clear If No Error
AH	Return Code
AL	Actual Sectors Written Count

INT 13h AH=04h: Verify Sectors From Drive

Parameters:

AH	04h
AL	Sectors To Verify Count
CH	Track
CL	Sector
DH	Head
DL	Drive
ES:BX	Buffer Address Pointer

Results:

CF	Set On Error, Clear If No Error
AH	Return Code
AL	Actual Sectors Verified Count

INT 13h AH=05h: Format Track

Parameters:

AH	05h
AL	Sectors To Format Count
CH	Track
CL	Sector
DH	Head
DL	Drive
ES:BX	Buffer Address Pointer

4-byte address field (applies to PC/XT 286,AT, PS/1 and PS/2)

Byte	Meaning	Allowable Values
1	Track	
2	Head	
3	Sector	
4	Bytes/Sector	0=128, 1-256, 2-512, 3-1024

Results:

CF	Set On Error, Clear If No Error
AH	Return Code

INT 13h AH=06h: Format Track Set Bad Sector Flags

Parameters:

AH	06h
AL	Interleave
CH	Track
CL	Sector
DH	Head
DL	Drive

Results:

CF	Set On Error, Clear If No Error
AH	Return Code

INT 13h AH=07h: Format Drive Starting at Track

Parameters:

AH	07h
AL	Interleave
CH	Track
CL	Sector
DH	Head
DL	Drive

Results:

CF	Set On Error, Clear If No Error
AH	Return Code

INT 13h AH=08h: Read Drive Parameters

Parameters:

Registers	
AH	08h = function number for read_drive_parameters
DL	drive index (e.g. 1st HDD = 80h)
ES:DI[7]	set to 0000h:0000h to work around some buggy BIOS

Results:

CF	Set On Error, Clear If No Error
AH	Return Code
DL	number of hard disk drives
DH[7]	logical last index of heads = number_of − 1 (because index starts with 0)
CX	[7:6] [15:8][7] logical last index of cylinders = number_of − 1 (because index starts with 0) [5:0][7] logical last index of sectors per track = number_of (because index starts with 1)
BL[7]	drive type (only AT/PS2 floppies)
ES:DI[7]	pointer to drive parameter table (only for floppies)

Remarks:
Logical values of function 08h may/should differ from physical CHS values of function 48h.
Result register CX contains both cylinders and sector/track values, see remark of function 02h.

INT 13h AH=09h: Init Drive Pair Characteristics

Parameters:

AH	09h
DL	Drive

Results:

CF	Set On Error, Clear If No Error
AH	Return Code

INT 13h AH=0Ah: Read Long Sectors From Drive

The only difference between this function and function 02h (see above) is that function 0Ah reads 516 bytes per sector instead of only 512. The last 4 bytes contains the Error Correction Code (ECC), a checksum of sector data.

INT 13h AH=41h: Check Extensions Present

Parameters:

Registers	
AH	41h = function number for extensions check[8]
DL	drive index (e.g. 1st HDD = 80h)
BX	55AAh

Results:

CF	Set On Not Present, Clear If Present
AH	Error Code or Major Version Number
BX	AA55h
CX	Interface support bitmask: 1 – Device Access using the packet structure 2 – Drive Locking and Ejecting 4 – Enhanced Disk Drive Support (EDD)

INT 13h AH=42h: Extended Read Sectors From Drive

Parameters:

Registers	
AH	42h = function number for extended read
DL	drive index (e.g. 1st HDD = 80h)
DS:SI	segment:offset pointer to the DAP, see below

DAP : Disk Address Packet		
offset range	size	description
00h	1 byte	size of DAP (set this to 10h)
01h	1 byte	unused, should be zero
02h..03h	2 bytes	number of sectors to be read, (some Phoenix BIOSes are limited to a maximum of 127 sectors)
04h..07h	4 bytes	segment:offset pointer to the memory buffer to which sectors will be transferred (note that x86 is little-endian: if declaring the segment and offset separately, the offset must be declared before the segment)
08h..0Fh	8 bytes	absolute number of the start of the sectors to be read (1st sector of drive has number 0)

Results:

CF	Set On Error, Clear If No Error
AH	Return Code

INT 13h AH=43h: Extended Write Sectors to Drive

Parameters:

Registers	
AH	43h = function number for extended write
AL	bit 0 = 0: close write check, bit 0 = 1: open write check, bit 1–7:reserved, set to 0
DL	drive index (e.g. 1st HDD = 80h)
DS:SI	segment:offset pointer to the DAP

Results:

CF	Set On Error, Clear If No Error
AH	Return Code

INT 13h AH=48h: Extended Read Drive Parameters

Parameters:

Registers	
AH	48h = function number for extended_read_drive_parameters
DL	drive index (e.g. 1st HDD = 80h)
DS:SI	segment:offset pointer to Result Buffer, see below

Result Buffer		
offset range	size	description
00h..01h	2 bytes	size of Result Buffer (set this to 1Eh)
02h..03h	2 bytes	information flags
04h..07h	4 bytes	physical number of cylinders = last index + 1 (because index starts with 0)
08h..0Bh	4 bytes	physical number of heads = last index + 1 (because index starts with 0)
0Ch..0Fh	4 bytes	physical number of sectors per track = last index (because index starts with 1)
10h..17h	8 bytes	absolute number of sectors = last index + 1 (because index starts with 0)
18h..19h	2 bytes	bytes per sector
1Ah..1Dh	4 bytes	optional pointer to Enhanced Disk Drive (EDD) configuration parameters which may be used for subsequent interrupt 13h Extension calls (if supported)

Results:

CF	Set On Error, Clear If No Error
AH	Return Code

Remark: Physical CHS values of function 48h may/should differ from logical values of function 08h.

See also

- INT 10H
- BIOS interrupt call
- Cylinder-head-sector
- INT (x86 instruction)
- DPMI (DOS Protected Mode Interface)

References

1. Stevens, Curtis (1995-01-26). "Enhanced Disk Drive Specification 1.1 -DRAFT- Phoenix Technologies, see 3.0 BIOS Extensions (registration required)" (PDF). Retrieved 2016-11-03.
2. Landis, Hale (1995-02-11). "BIOS Types, CHS Translation, LBA and Other Good Stuff, See BIOS Type 6 The Phoenix Enhanced Disk Drive Specification. (registration required)" (PDF). Retrieved 2016-11-03.
3. Brown, Ralf (2000-07-16). "Ralf Browns Interrupt List (v61 html)". Retrieved 2016-11-03.
4. Brown, Ralf (2000-07-16). "The x86 Interrupt List (v61 original text) archive: "inter61a.zip", subfile: "INTERRUP.B", heading: "B-1302" (INT13, 02 Read), Notes". Retrieved 2016-11-03.
5. Disk size limitations, The 8.4 GB limit (and others) (http://www.tldp.org/HOWTO/Large-Disk-HOWTO-4.html)
6. ctyme.com - Ralf Browns Interrupt List, Indexed html, DISK - READ SECTOR(S) INTO MEMORY (http://www.ctyme.com/intr/rb-0607.htm)
7. ctyme.com - DISK - GET DRIVE PARAMETERS (PC,XT286,CONV,PS,ESDI,SCSI) (http://www.ctyme.com/intr/rb-0621.htm)
8. ctyme.com - IBM/MS INT 13 Extensions - INSTALLATION CHECK (http://www.ctyme.com/intr/rb-0706.htm)

External links

- BIOS Interrupt 13h Extensions (http://www.dewassoc.com/support/bios/bios_interrupt_13h_extensions.htm)
- Ralf Brown's comprehensive Interrupt List (http://www.cs.cmu.edu/~ralf/files.html)
- Norton Guide about int 13h, ah = 00h .. 1ah (http://www.ousob.com/ng/bios/ng1706f.php)

Retrieved from "https://en.wikipedia.org/w/index.php?title=INT_13H&oldid=747750642"

Categories: IBM PC compatibles │ BIOS │ Interrupts

- This page was last modified on 4 November 2016, at 04:26.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.