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 (IBM PC-descended) 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. Modern PC BIOSes also include INT 13h extension functions, originated by IBM and Microsoft in 1992, that provide those same disk access services using 64-bit LBA addressing; with minor additions, these were quasi-standardized by Phoenix Technologies and others as the EDD (Enhanced Disk Drive) BIOS extensions.

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

Modern computers come with both BIOS INT 13h and UEFI functionality that provides the same services and more, with the exception of UEFI Class 3 that completely removes CSM thus lacks INT 13h and other interrupts. Typically, UEFI drivers use LBA-addressing instead of CHS-addressing.

Overview

Under real mode operating systems, such as 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 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 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 Disk Access. Besides performing low-level disk access, INT 13h calls and related BIOS data structures also provide information about the types and capacities of disks (or other DASD devices) attached to the system; when a protected-mode OS boots, it may use that information from the BIOS to enumerate disk hardware so that it (the OS) can load and configure appropriate disk I/O drivers.

The original BIOS real-mode INT 13h interface supports drives of sizes up to about 8 GB 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. A lesser limit, about 504 MB, was imposed by the combination of CHS addressing limits used by the BIOS and those used by ATA hard disks, which are dissimilar. When the CHS addressing limits of both the BIOS and ATA are combined (i.e. when they are applied simultaneously), the number of 512-byte sectors that can be addressed represent a total of about 504 MB.

The 504 MB limit was overcome using *CHS translation*, a technique by which the BIOS would simulate a fictitious CHS geometry at the INT 13h interface, while communicating with the ATA drive using its native logical CHS geometry. (By the time the 504 MB barrier was being approached, ATA disks had long before ceased to present their real physical geometry parameters at the external ATA interface.) Translation allows the BIOS, still using CHS addressing, to effectively address ATA disks with sizes up to exactly 8064 MB, the native capacity of the BIOS CHS interface alone. (The ATA interface has a much larger native CHS addressing capacity, so once the "interference" of the CHS limits of BIOS and ATA was resolved by addressing, only the smaller limitation of the BIOS was significant.) *CHS translation* is sometimes referred to as *logical CHS addressing*, but that is actually a misnomer since by the time of this BIOS development, ATA CHS addresses were already logical, not physical. The 8064 MB 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—dictating that the format or size of CHS addresses passed to INT 13h could not be changed to add more bits to one of the fields, e.g. the Cylinder-number field. 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). There were briefly a number of BIOSes that offered incompatible versions of this interface—for example, AWARD AT BIOS and AMI 386sx BIOS have been extended to handle up to 4096 cylinders by placing bits 10 and 11 of the cylinder number into bits 6 and 7 of register DH.

All versions of MS-DOS, (including MS-DOS 7 and Windows 95) have a bug which prevents booting disk drives with 256 heads (register value 0xFF), so many modern BIOSes provide CHS translation mappings with at most 255 (0xFE) heads,[1][2] thus reducing the total addressable space to exactly 8032.5 MiB (approx 7.844 GiB).[3]

To support addressing of even larger disks, an interface known as **INT 13h Extensions** was introduced by IBM and Microsoft, then later re-published and slightly extended by Phoenix Technologies as part of **BIOS Enhanced Disk Drive Services** (EDD).[4][5] It defines new functions within the INT 13h service, all having function numbers greater than 40h, that use 64-bit logical block addressing (LBA), which allows addressing up to 8 ZiB. (An ATA 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 very simple data structure that contains an interface version, data size, and LBAs. For software backward-compatibility, the extended functions are implemented alongside the original CHS functions, and calls to functions from both sets can be intermixed, even for the same drive, with the caveat that the CHS functions cannot reach past the first 8064 MB of the disk.

Some cache drivers flush their buffers when detecting that DOS is bypassed by directly issuing INT 13h from applications. A dummy read via INT 13h can be used as one of several methods to force cache flushing for unknown caches (e.g. before rebooting).[1][2]

AMI BIOSes from around 1990–1991 trash word unaligned buffers. Some DOS and terminate-and-stay-resident programs clobber interrupt enabling and registers so PC DOS and MS-DOS install their own filters to prevent this.[6]

List of `INT 13h` services

Drive Table

`DL = 00h`	1st floppy disk ("drive A:")
`DL = 01h`	2nd floppy disk ("drive B:")
`DL = 02h`	3rd floppy disk ("drive C:")
`...`	
`DL = 7Fh`	128th floppy disk
`DL = 80h`	1st hard disk
`DL = 81h`	2nd hard disk
`DL = 82h`	3rd hard disk
`...`	
`DL = E0h`	CD/DVD, or 97th hard disk
`...`	
`DL = FFh`	128th hard disk

Function Table

AH = 00h		Reset Disk System
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

AH = 4Bh	EXT	Get Drive Emulation Type

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 System

Parameters

AH	00h
DL	Drive (bit 7 set means reset both hard and floppy disks)

Results

CF	Set on error
AH	Return Code

`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

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

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 $FF00) shr 8 ) or ( (CX and $C0) 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.

Code Example

```
[ORG 7c00h]   ; code starts at 7c00h
xor ax, ax    ; make sure ds is set to 0
mov ds, ax
cld
; start putting in values:
mov ah, 2h    ; int13h function 2
mov al, 63    ; we want to read 63 sectors
mov ch, 0     ; from cylinder number 0
mov cl, 2     ; the sector number 2 - second sector (starts from 1, not 0)
mov dh, 0     ; head number 0
xor bx, bx
mov es, bx    ; es should be 0
mov bx, 7e00h ; 512bytes from origin address 7c00h
int 13h
jmp 7e00h     ; jump to the next sector

; to fill this sector and make it bootable:
times 510-($-$$) db 0
dw 0AA55h
```

After this code section (which the asm file should start with), you may write code and it will be loaded to memory and executed.

Notice how we didn't change dl (the drive). That is because when the computer first loads up, dl is set to the number of the drive that was booted, so assuming we want to read from the drive we booted from, there is no need to change dl.

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

Results

Registers	Description
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

<table>
<tr><th>Registers</th><th>Description</th></tr>
<tr><td>AH</td><td>42h = function number for extended read</td></tr>
<tr><td>DL</td><td>drive index (e.g. 1st HDD = 80h)</td></tr>
<tr><td>DS:SI</td><td>segment:offset pointer to the DAP, see below
<p>DAP : Disk Address Packet</p>
<table>
<tr><th>offset range</th><th>size</th><th>description</th></tr>
<tr><td>00h</td><td>1 byte</td><td>size of DAP (set this to 10h)</td></tr>
<tr><td>01h</td><td>1 byte</td><td>unused, should be zero</td></tr>
<tr><td>02h..03h</td><td>2 bytes</td><td>number of sectors to be read, (some Phoenix BIOSes are limited to a maximum of 127 sectors)</td></tr>
<tr><td>04h..07h</td><td>4 bytes</td><td>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)</td></tr>
<tr><td>08h..0Fh</td><td>8 bytes</td><td>absolute number of the start of the sectors to be read (1st sector of drive has number 0) using logical block addressing (note that the lower half comes before the upper half)[9]</td></tr>
</table>
</td></tr>
</table>

Results

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

As already stated with int 13h AH=02h, care must be taken to ensure that **the complete buffer is inside the given segment**, *i.e. (BX + size_of_buffer) <= 10000h*

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

Parameters

Registers	Description
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

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

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

Parameters

<table>
<tr><th>Registers</th><th>Description</th></tr>
<tr><td>AH</td><td>48h = function number for extended_read_drive_parameters</td></tr>
<tr><td>DL</td><td>drive index (e.g. 1st HDD = 80h)</td></tr>
<tr><td>DS:SI</td><td>segment:offset pointer to Result Buffer, see below
Result Buffer
<table>
<tr><th>offset range</th><th>size</th><th>description</th></tr>
<tr><td>00h..01h</td><td>2 bytes</td><td>size of Result Buffer (set this to 1Eh)</td></tr>
<tr><td>02h..03h</td><td>2 bytes</td><td>information flags</td></tr>
<tr><td>04h..07h</td><td>4 bytes</td><td>physical number of cylinders = last index + 1
(because index starts with 0)</td></tr>
<tr><td>08h..0Bh</td><td>4 bytes</td><td>physical number of heads = last index + 1
(because index starts with 0)</td></tr>
<tr><td>0Ch..0Fh</td><td>4 bytes</td><td>physical number of sectors per track = last index
(because index starts with 1)</td></tr>
<tr><td>10h..17h</td><td>8 bytes</td><td>absolute number of sectors = last index + 1
(because index starts with 0)</td></tr>
<tr><td>18h..19h</td><td>2 bytes</td><td>bytes per sector</td></tr>
<tr><td>1Ah..1Dh</td><td>4 bytes</td><td>optional pointer to Enhanced Disk Drive (EDD) configuration parameters which may be used for subsequent interrupt 13h Extension calls (if supported)</td></tr>
</table></td></tr>
</table>

Results

Registers	Description
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.

INT 13h AH=4Bh: Get Drive Emulation Type

Parameters

Regsiters	Description
AH	4Bh = get drive emulation type
AL	01
DL	drive index (e.g. 1st HDD = 80h)
DS:SI	points to an empty structure for result . must be 13h in size

Results

<table>
<tr><th>Registers</th><th>Description</th></tr>
<tr><td>CF</td><td>Set On Error, Clear if No Error</td></tr>
<tr><td>AX</td><td>Return Code</td></tr>
<tr><td>DS:SI</td><td>Points to a specification structure
Specification Structure
<table>
<tr><th>Offset</th><th>Size (byte)</th><th>Description</th></tr>
<tr><td>00h</td><td>1</td><td>Size of packets in byte (13h)</td></tr>
<tr><td>01h</td><td>1</td><td>Boot Media Type :
<table>
<tr><th>Bits</th><th></th></tr>
<tr><td>0 - 3</td><td>0000b: No Emulation
0001b: 1.2M Floppy Disk
0010b: 1.44M Floppy Disk
0011b: 2.88M Floppy Disk
0100b: Hard Disk</td></tr>
<tr><td>4-5</td><td>Reserved</td></tr>
<tr><td>6</td><td>Image Contain ATAPI Driver</td></tr>
<tr><td>7</td><td>Image Contain SCSI Driver</td></tr>
</table>
</td></tr>
<tr><td>02h</td><td>1</td><td>Drive Number (Drive Index)</td></tr>
<tr><td>03h</td><td>1</td><td>CD-ROM Controller Number</td></tr>
<tr><td>04h</td><td>4</td><td>Logical Block Address (LBA) of disk image to emulate</td></tr>
<tr><td>08h</td><td>2</td><td>Device Specification:
bit 0: Drive is slave instead of master
bits 7-0: LUN and PUN</td></tr>
<tr><td>0Ah</td><td>2</td><td>Segment Of 3K Buffer For Caching CD-ROMs Reads</td></tr>
<tr><td>0Ch</td><td>2</td><td>Initial Boot Image Segment Starting From 7c0h Segment</td></tr>
<tr><td>0Eh</td><td>2</td><td>Number Of Sectors (512 bytes long) To Load</td></tr>
<tr><td>10h</td><td>1</td><td>Cylinder Count Low Byte (From int 8h)</td></tr>
<tr><td>11h</td><td>1</td><td>Sector Count (From int 8h)</td></tr>
<tr><td>12h</td><td>1</td><td>Head Count (From int 8h)</td></tr>
</table>
</td></tr>
</table>

See also

- INT 10H
- BIOS interrupt call
- Cylinder-head-sector
- INT (x86 instruction)
- DPMI (DOS Protected Mode Interface)
- Ralf Brown's Interrupt List
- BIOS Enhanced Disk Drive Specification (http://www.o3one.org/hwdocs/bios_doc/bios_specs_edd30.pdf)

References

1. Brown, Ralf D. (2000-07-16). "Ralf Browns Interrupt List (v61 html)" (http://www.delorie.com/djgpp/doc/rbinter/id/13/6.html). Retrieved 2016-11-03.
2. Brown, Ralf D. (2000-07-16). "The x86 Interrupt List (v61 original text) archive: "inter61a.zip", subfile: "INTERRUP.B", heading: "B-1302" (INT13, 02 Read), Notes" (https://www.cs.cmu.edu/~ralf/interrupt-list/inter61b.zip). Retrieved 2016-11-03.
3. Disk size limitations, The 8.4 GB limit (and others) (http://www.tldp.org/HOWTO/Large-Disk-HOWTO-4.html)
4. Stevens, Curtis (1995-01-26). "Enhanced Disk Drive Specification 1.1 -DRAFT- Phoenix Technologies, see 3.0 BIOS Extensions (registration required)" (http://www.t10.org/cgi-bin/ac.pl?t=d&f=95-153r0.pdf) (PDF). Retrieved 2016-11-03.
5. 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)" (http://www.t10.org/cgi-bin/ac.pl?t=d&f=95-156r0.pdf) (PDF). Retrieved 2016-11-03.
6. ctyme.com - Ralf Browns Interrupt List, Indexed html, DISK - READ SECTOR(S) INTO MEMORY (https://web.archive.org/web/20160812112422/http://www.ctyme.com/intr/rb-0607.htm)
7. ctyme.com - DISK - GET DRIVE PARAMETERS (PC,XT286,CONV,PS,ESDI,SCSI) (https://web.archive.org/web/20160619063203/http://www.ctyme.com/intr/rb-0621.htm)
8. ctyme.com - IBM/MS INT 13 Extensions - INSTALLATION CHECK (https://web.archive.org/web/20160619063218/http://www.ctyme.com/intr/rb-0706.htm)
9. - LBA in Extended Mode (https://wiki.osdev.org/ATA_in_x86_RealMode_(BIOS))

External links

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