磁盘的 LBA 寻址转换为 CHS 寻址

CHS：

CHS，是英语：Cylinder-head-sector的缩写，翻译过来是：柱面-磁头-扇区。这是早期对磁盘驱动器的每一个物理数据块进行编址的一种方法。

扇区：在计算机磁盘存储器中，一个扇区（英语：sector）是磁盘或光盘上一个磁道的分区。每个扇区存储固定数量用户可访问的数据，传统上，对于硬盘驱动器（HDD）来说是512字节，对于CD-ROM和DVD-ROM来说是2048字节。较新的硬盘驱动器使用4096字节（4 KiB）扇区，这被称为Advanced Format（AF）。可以这样来理解扇区，磁盘上的每个磁道被等分为若干个弧段，这些弧段便是硬盘的扇区（Sector）。硬盘的第一个扇区，叫做引导扇区。

磁头

磁盘由许多盘片组成，每个盘有两面，每面各有一个磁头，都可记录信息。

柱面

当磁盘旋转时，磁头若保持在一个位置上，则每个磁头都会在磁盘表面划出一个圆形轨迹，这些圆形轨迹就叫做磁道（Track）。在磁盘中，处于同一半径圆的磁道组成的一个圆柱面（Cylinder）。

在CHS规范中，磁头的最大数就是16，扇区数是63。

一个硬盘的容量=柱面数（或磁道数）×磁头数×扇区数×每个扇区的大小（通常是512字节)。

1024×16×63×512≒504MB

1024×256×63×512≒7.875GB

IDE硬盘参数的限制

|  |  |  |  |
| --- | --- | --- | --- |
|  | **HDD** | **BIOS** | **小** |
| **柱面(C)** | 0 - 65535 | 0 - **1023** | 0 - 1023 |
| **磁头(H)** | 0 - **15** | 0 - 254 | 0 - 15 |
| **扇区(S)** | 1 - 255 | 1 - **63** | 1 - 63 |
| **最大容量** | 128GB | 7.8GB | **504MB** |

EIDE（Enhanced IDE：增强性IDE）是Pentium以上主板必备的标准接口。有标准标准模式、大模式、逻辑块地址式（LBA）3种工作方式。

增强型（有时也做扩展型）IDE接口是一个在你的计算机和它的大容量存储设备之间的标准电子接口。EIDE对集成电路设备的增强使得对大于528MB(实际大小504MB)的硬盘驱动器可寻址成为可能。EIDE还提供对硬盘驱动器的快速访问，支持内存直接访问(DMA)，并且通过AT附件包接口支持附加磁盘，包括CD-ROM和磁带设备。访问大于528MB(504MB)的驱动器时，EIDE （或与它一起提供的基本输入/输出系统）使用28位的逻辑字组地址(LBA) 来标明磁盘上的实际柱面，头以及数据的扇区。LBA 的28位地址提供了足够的信息来标明高达8.4 GB 容量的磁盘驱动器的每个扇区。 EIDE 在1994年被ANSI 做为一种标准采用。ANSI 把它叫做高级技术附加装置-2（它也叫做“高速硬盘接口”）。

为了兼容CHS寻址，能识别的最大容量为7.875GB，相当于LBA24bit。

LBA

逻辑区块寻址(Logical Block Address, LBA)是非常单纯的一种寻址模式﹔从0开始编号来定位区块，第一区块LBA=0，第二区块LBA=1，依此类推。这种定址模式替换了原先操作系统必须面对存储设备硬件构造的方式。最具代表性的首推CHS(cylinders-heads-sectors，磁柱-磁头-扇区)定址模式，区块必须以硬盘上某个磁柱、磁头、扇区的硬件位置所合成的地址来指定。CHS模式对硬盘以外的设备来说没什么作用(例如磁带或是网络存储设备)，所以通常也不会用在这些地方。过去MFM(Modified Frequency Modulation, 改良调频式)和RLL(Run Length Limited)存储设备都曾使用CHS模式，ATA-1设备更将延伸CHS(Extended Cylinders-Heads-Sectors, ECHS)也派上了用场。

SCSI采用LBA抽象定址。实际上硬件控制器还是以CHS来定址区块，但无论驱动程序还是任何以低级访问磁盘的应用程序(例如数据库软件)通常都不再需要这个参数。各种要求区块低级访问的系统调用把定义好的LBA传给驱动程序﹔最直接的情况下(逻辑器件与实体设备单一对应)驱动程序只是将LBA再传给硬件控制器。

LBA对应与逻辑器件虚拟化

当逻辑器件是经由虚拟化或是集合所构成的，像是RAID(磁盘阵列)和SANs(Storage Area Network)这种复杂的情况，就得把应用程序根据其观点中的磁盘来指定的LBA转换成每个实体存储设备上的LBA。在复杂的分布式存储结构下，从发出要求的应用程序到实体甚至是远程设备之间，会有太多这样的LBA转换。

在容量<7.875GB的时候，CHS与LBA可以互相转换。

CHS地址可用以下公式转成LBA,

#lba=(#c\*H+#h)\*S+#s-1

其中,

#c、#h、#s分别是磁柱、磁头、扇区的编号

#lba是逻辑区块编号

H=heads per cylinder，每个磁柱的磁头数

S=sectors per track，每磁道的扇区数

LBA可用以下公式对应到CHS:

#c=#lba/(S\*H)

#h=(#lba/S)%H

#s=(#lba%S)+1

其中,

/ 是整数除法

% 是取整数除法中的余数

请注意，当今的磁盘使用ZBR(Zone Bit Recording, 等密度记录)方式，实际的每轨扇区数得根据它是哪一轨。不过磁盘还是会提供这个参数来匹配公式，内部再自动调整。

其它公式:

#lba/S=q 余 r

#s=1+r

q/H=#c 余 #h

例如:

CHS总数=[600, 10, 84]，求#lba=1234所对应的CHS编号:

1234/84=14 余 58

#s=1+58=59

14/10=1 余 4

#c=1

#h=4

#chs=(1, 4, 59)

验算: (1\*10+4)\*84+59-1=14\*84+58=1234

LBA、ATA设备以及Enhanced BIOS

ATA-1规范中定义了28位定址模式，当成LBA或是CHS都可以。如果用CHS这28位拆成: 磁柱16位、磁头4位、扇区8位。注意CHS模式扇区是从1开始算，所以在这个规范中扇区数最多只有255个，最大扇区编号为254(0xFE)。

规范采用当时，CHS的BIOS规范只有24位: 磁柱10比特、磁头8位、扇区6比特，定义在BIOS的INT 13H软件中断里，而且已经用在DOS的MBR(Master Boot Record，主要开机记录)。这造成了BIOS CHS跟ATA CHS之间必须经过转换，否则各参数只能用到两者的最大公因数即CHS比特数={10, 4, 6}，也就是1024×16×63个扇区，以每扇区512位组计算得504 MB。转换方式其一是Large模式或称Enhanced BIOS模式(又名Bit Shift Translation, 位移转换)，此方式会重新对应侦测到的磁柱和磁头数而扇区数不变﹔方式其二是将头一种CHS对应到LBA之后再换算成另一种CHS机制，称为LBA-assist。

即使利用这些转址方式，BIOS定给MS-DOS逻辑扇区(以及Windows NT 4.0硬盘分区)的CHS比特数={10, 8, 6}机制顶多也只能达到7.84 GiB。

以每扇区512位组来计算，ATA-1所定义的28位LBA上限达到128 GiB。

2002年ATA-6规范采用48位LBA，同样以每扇区512位组计算容量上限可达128 Petabytes。

磁盘的LBA读取

INT 13h AH=41h: Check Extensions Present

参数：

寄存器：

AH 41h = function number for extensions check

DL drive index (e.g. 1st HDD = 80h)

BX 55AAh

返回值：

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

参数：

寄存器：

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)DAP大小：16字节。

01h 1 byte unused, should be zero，未使用，固定值0.

02h..03h 2 bytes number of sectors to be read, (some Phoenix BIOSes are limited to a maximum of 127 sectors)要读取的扇区数量。

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)存储读取内容的段地址和偏移地址。

08h..0Fh 8 bytes absolute number of the start of the sectors to be read (1st sector of drive has number 0)读取扇区的绝对开始扇区号。

返回值：

CF Set On Error, Clear If No Error

AH Return Code

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

参数：

寄存器：

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

返回值：

CF Set On Error, Clear If No Error

AH Return Code

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

参数：

寄存器：

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)

返回值：

CF Set On Error, Clear If No Error

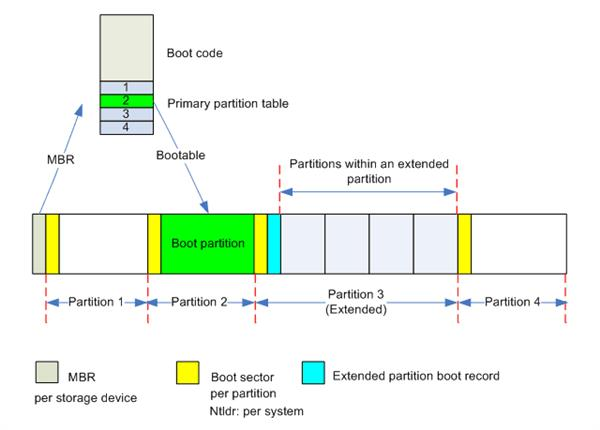
AH Return Code

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

MBR

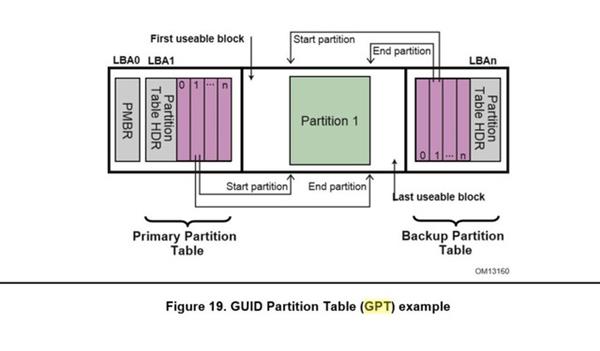
即使使用LBA寻址方式，由于在MBR下LBA最大能使用32位来管理扇区地址，通常1扇区的大小是512字节，所以MBR的最大容量是2的32次方个扇区，也就是2TB。虽然有大扇区（Big Sector 4KB／扇区），那也不过是16TB而已。

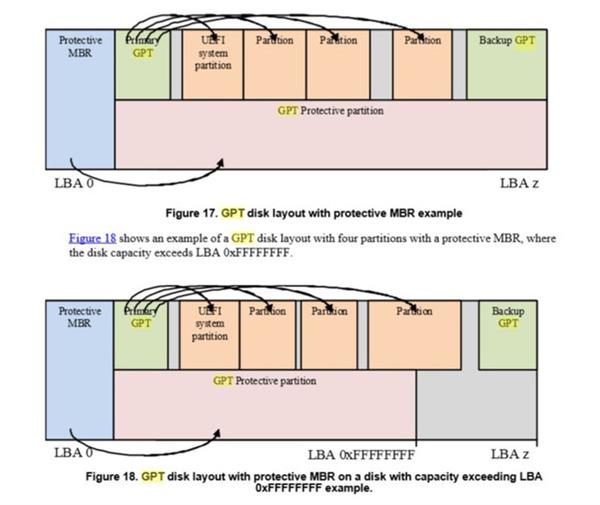
对于容量不超过2TB的磁盘来说，用MBR来管理分区是足够了的。



那么容量超过2TB的怎么办？

目前来看，是因特尔的方案比较流行。GUID Partision Table，Globally Unique Identifier Partition Table，全局唯一标识分区表。还是叫GUID分区表吧，最大容量达到18EB，多大？18艾字节=18\*1024拍字节PB=18\*1024\*1024太字节TB，好吧，目前来看够大。





UEFI的全称是Unified Extensible Firmware Interface，意即统一可扩展固件接口，它是基于EFI 1.10标准为基础发展而来，值得注意的是在UEFI正式确立之前，Intel就开始积极推进传统BIOS的升级方案，并最终确立了过渡方案EFI标准，直到2007年Intel将EFI标准的改进与完善工作交给Unified EFI Form进行全权负责，EFI标准则正式更名为UEFI。

相比传统BIOS，UEFI最大的几个区别在于：

1、编码99%都是由C语言完成；

2、一改之前的中断、硬件端口操作的方法，而采用了Driver/protocol的新方式；

3、将不支持X86实模式，而直接采用Flat mode（也就是不能用DOS了，现在有些 EFI 或 UEFI 能用是因为做了兼容，但实际上这部分不属于UEFI的定义了）；

4、输出也不再是单纯的二进制code，改为Removable Binary Drivers；

5、OS启动不再是调用Int19，而是直接利用protocol/device Path；

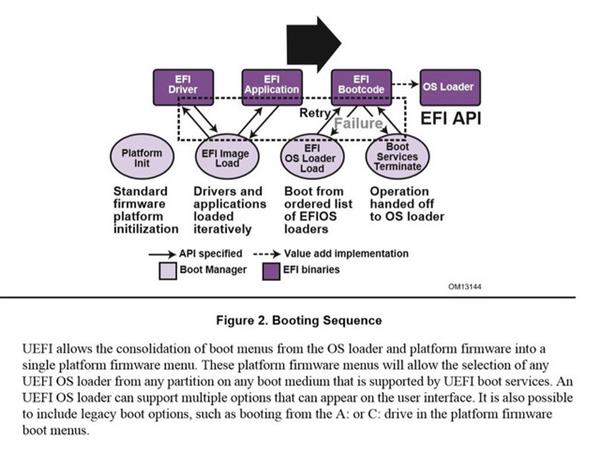
6、对于第三方的开发，前者基本上做不到，除非参与BIOS的设计，但是还要受到ROM的大小限制，而后者就便利多了。

7、弥补BIOS对新硬件的支持不足的问题。

UEFI和GPT是相辅相成的，二者缺一不可，要想使用GPT分区表则必须是UEFI BIOS环境。UEFI于用户而言最典型的特征就是使用了图形化界面，虽然还未达到操作系统界面的图形交互功能，但人性化的界面、鼠标的操作，已经将BIOS变得非常易用，对于不少电脑初级用户来说也可以很好的查看和设置BIOS的相关选项和功能。

除了图形化界面，UEFI相比传统BIOS，还提供了文件系统的支持，它能够直接读取FAT、FAT32分区中的文件，例如华硕、华擎等主板在UEFI BIOS环境下更新BIOS就可以直接读取U盘中的BIOS及其他文件，另外新的UEFI主板基本都提供了截屏功能，这些截屏图片都可以存储在U盘当中。

UEFI还有一个重要特性就是在UEFI下运行应用程序，这类程序文件通常以efi结尾。利用UEFI可以直接识别FAT分区中的文件，又有可直接在其中运行应用程序。我们就可以将Windows安装程序做成efi类型应用程序，然后把它放到任意FAT分区中直接运行即可。



UEFI启动顺序

UEFI的模块化设计，它在逻辑上分为硬件控制与OS（操作系统）软件管理两部分，硬件控制为所有UEFI版本所共有，而OS软件管理其实是一个可编程的开放接口。借助这个接口，主板厂商可以实现各种丰富的功能。比如我们熟悉的各种备份及诊断功能可通过UEFI加以实现，主板或固件厂商可以将它们作为自身产品的一大卖点。UEFI也提供了强大的联网功能，其他用户可以对你的主机进行可靠的远程故障诊断，而这一切并不需要进入操作系统。

当然由于UEFI主要由高级语言编写（C语言），相比于传统BIOS的汇编语言，UEFI BIOS在安全防护性方面相比传统BIOS要弱，比较容易遭到病毒的攻击，安全性有待进一步提升。不过在图形化界面、应用程序扩展面前，UEFI BIOS还是非常成功的。