硬盘

第一节 概要

硬盘（Hard Disk Drive，简称HDD）是电脑上使用坚硬的旋转盘片为基础的不挥发性存储设备，它在平整的磁性表面存储和检索数字数据，信息通过距离磁性表面很近的磁头，由电磁流来改变极性的方式被电磁流写到磁盘上，信息可以通过相反的方式读取，例如读取头经过纪录数据的上方时磁场导致线圈中电气信号的改变。硬盘的读写是采用随机存取的方式，因此可以以任意顺序读取硬盘中的数据。

硬盘是由IBM在1956年9月13日开始使用的，即IBM 305 RAMAC（Random Access Method of Accounting and Control）计算机系统的IBM 350磁盘单元，是现代硬盘的最早雏形。在1960年代初成为通用式电脑中主要的辅助存放设备，随着技术的进步，硬盘也成为服务器及个人电脑的主要组件。

早期的硬盘存储介质是可替换的，不过今日典型的硬盘采用的是固定的存储介质，盘片与磁头被封装在机身里（除了一个有过滤的气孔，用来平衡工作时产生的热量导致的气压差）。

第二节 物理结构

硬盘的物理结构一般由磁头与盘片、电动机、主控芯片与排线等部件组成。就硬盘而言，一般是由一至数片坚硬材料制成的并涂以磁性介质的盘片压制而成。每个盘有两面，每面各有一个磁头，都可记录信息。硬盘的盘片上分布着很多小磁体，利用磁体磁化时形成的S极或N极表示0或1。当主电动机带动盘片旋转时，副电动机带动一组（磁头）到相对应的盘片上并确定读取正面还是反面的盘面，磁头悬浮在盘面上画出一个与盘片同心的圆形轨道（磁轨或称柱面），这时由磁头的磁感线圈感应盘面上的磁性与使用硬盘厂商指定的读取时间或数据间隔定位扇区，从而得到该扇区的数据内容。

虽然读写磁头不是贴着盘片运行的，而是和盘面有着一定的距离，但还是会发生磁头划碰的。磁头划碰是一种硬盘故障，在硬盘读写头和旋转的硬盘片接触时发生，在磁盘表面的介质产生永久不可恢复的损害。

磁头通常包裹在盘片表面的很薄的空气层中（1990年代中期采用薄液体层）。盘片的最上层是聚四氟乙烯类似的物质，作为润滑剂。下面是一层溅射碳。这两层保护磁层（数据存储区），防止读写头的意外接触。

磁盘读写头使用薄膜技术，材料足够坚硬，较难通过保护层划伤。磁头划碰比较可能是由于外力通过读写头，对盘片产生足够的压力，导致磁性存储层划伤。其他的污垢或碎屑，过度冲击或振动，意外掉落，能使读写头对盘片造成冲击，在这个过程中通常损坏读写头。

第三节 接口

显卡，网卡，声卡等各种卡，却没有硬盘卡。其实硬盘和硬盘I/O接口(硬盘控制器)整合在一起了，也就是以前所说的IDE接口。这还得从ATA说起。

ATA，Advanced Technology Attachment，高级技术配置（简称“ATA”）与由Integrated Drive Electronics，集成驱动电子设备（简称IDE）技术实现的磁盘驱动器关系最密切。IDE是一种计算机系统接口，主要用于硬盘和CD-ROM，本意为“把控制器与盘体集成在一起的硬盘”。

一个IDE线上可以挂2块硬盘, 一个是主盘(Master)、一个是从盘(Slave)。一个主板支持4个IDE(PATA)硬盘, 所以提供了2个IDE插槽- IDE0和IDE1, 不过按照标准: 插槽叫做通道。

IDE0 -> Primary通道 -> Master

IDE1 -> Secondary通道 -> Slave

SATA因为兼容, 所以也是这一套规范。

SATA，串行ATA（Serial ATA: Serial Advanced Technology Attachment）是串列SCSI（SAS: Serial Attached SCSI）的孪生兄弟。两者的排线兼容，SATA硬盘可接上SAS接口。它是一种电脑总线，主要功能是用作主板和大量存储设备（如硬盘及光盘驱动器）之间的数据传输之用。SATA主要用于个人电脑。

2000年11月由“Serial ATA Working Group”团体所制定，SATA已经完全替换旧式PATA（Parallel ATA或旧称IDE）接口的旧式硬盘，因采用串行方式传输数据而得名。在数据传输上这一方面，SATA的速度比以往更加快捷，并支持热插拔，使电脑运作时可以插上或拔除硬件。另一方面，SATA总线使用嵌入式时钟频率信号，具备比以往更强的纠错能力，能对传输指令（不仅是数据）进行检查，如果发现错误会自动矫正，提高数据传输的可靠性。不过，SATA和以往最明显的分别，是使用较细的排线，有利机箱内部的空气流通，某程度上增加了整个平台的稳定性。

而在SATA技术日益发展下，不使用ATA的主板已经出现，而且Intel在新型的芯片组中已经不默认支持ATA接口，主机版厂商需要另加芯片去对ATA作出支持（通常是为了兼容旧有硬盘和光盘驱动器）。

2002年以后，由于SATA（Serial ATA）的推出，原有的ATA改名为PATA（并行高技术配置，Parallel ATA）。2013年12月29日，西部数据正式停止PATA硬盘供应，而希捷科技则已停售产多年，这意味着1986年设计的PATA接口在经历27年后正式退出历史舞台。

现时，SATA分别有SATA 1.5Gbit/s、SATA 3Gbit/s和SATA 6Gbit/s三种规格。2013年推出更快速的SATA Express规格。

第四节 柱面-磁头-扇区寻址方法

CHS，是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所谓的三维寻址方法。

扇区，在计算机磁盘存储器中，一个扇区（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。

磁道，磁盘上划分了很多个同心圆，这些同心圆就是磁道。但打开硬盘，用户不能看到这些，实际上磁道是被磁头磁化的同心圆。磁道之间是有间隔的，因为磁化单元太近会产生干扰。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）。

柱面，在有多个盘片构成的磁盘中，由不同盘片的面，但处于同一半径圆的多个磁道组成的一个圆柱面（Cylinder）。

假如一块磁盘只有3个磁盘片，每块盘片的磁道数是相等的.从外圈开始，这些磁道被分成了0磁道、1磁道、2磁道……。具有相同磁道编号的同心圆组成面就称作柱面。为了便于理解，柱面可以看作没有底的铁桶。柱面数就是磁盘上的磁道数。柱面是硬盘分区的最小单位。

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)。在CHS规范中，磁头的最大数就是16，扇区数是63。

1024×16×63×512≒504MB

1024×256×63×512≒7.875GB

IDE硬盘参数的限制

|  |  |  |  |
| --- | --- | --- | --- |
|  | **HDD** | **BIOS** | **较小值** |
| **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。

第五节 逻辑区块寻址方法

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。

CHS与LBA的转换，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

需要注意的是，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。

第六节 硬盘BIOS 中断

BIOS Int 13H 调用是 BIOS 提供的磁盘基本输入输出中断调用，它可以完成磁盘(包括硬盘和软盘)的复位、读写、校验、定位、诊断、格式化等功能。在其规范发展的过程中，分别提供了标准的int 13h 和扩展的int 13h。如前所叙，标准的int 13h 使用CHS 寻址模式，扩展的int 13h 使用LBA 寻址模式。

标准的int 13h

标准的int13h 提供的服务一般从功能0 到功能25。通过中断13h 访问硬盘服务，所有的功能都要求在DL 中用一个号来指定要使用的驱动器。将DL 中的第7 位置高表示正在访问硬盘，否则就是在访问软盘。将DL 设为80h,可访问硬盘驱动器0；同样地，当DL 设为81h 时可访问驱动器1。这里仅仅举个例子，下读取硬盘驱动器参数的API：

//读磁盘驱动器参数

//入口： AH=8；DL=驱动器号(80h+从零开始地硬盘驱动器号)

//返回：CF=0：

AH=0；AL=未定义

CH=最大磁柱号（10 位磁柱号的低8 位）

CL：第6、7 位=磁柱号的第8、9 位；第0~5 位=最大扇区号(1~63)

DH=最大磁头号(0~255)

DL=驱动器数目

CF=1：

如果至少激活了一个驱动器：

AH=7；AL=CX=DX=0

如果没有硬盘存在：

AH=1；AL=BX=CX=DX=ES=0

可以看出标准的int13h 只能简单的获取硬盘的CHS 参数而已，无法满足更多的要求。

扩展的int 13h

设计扩展 Int13H 接口的目的是为了扩展 BIOS 的功能，使其支持多于1024 柱面的硬盘，以及可移动介质的琐定、解锁及弹出等功能。扩展int 13h 的功能号从41h 开始，功能48h 提供的功能是获取扩展驱动器参数。

首先给出扩展驱动器参数的数据结构：

typedef struct DriveParametersPackettag {

unsigned int InfoSize; // 数据包尺寸 (26 字节)

unsigned int Flags; // 信息标志

long int Cylinders; // 磁盘柱面数

long int Heads; // 磁盘磁头数

long int SectorsPerTrack; // 每磁道扇区数

long int Sectors1; // 磁盘总扇区数

long int Sectors2;

unsigned int SectorSize; // 扇区尺寸 (以字节为单位)

unsigned int DPTE\_ofs; // 指向DPTE 的指针

unsigned int DPTE\_seg;

}DriveParametersPacket;

请注意，unsigned int 是16 位长，long int 为32 位长。以上给出的是在使用扩展int 13h

的功能48h 时必须用到的数据结构。API 的说明如下：

//获取硬盘扩展驱动器参数

//入口:

AH = 48h

DL = 驱动器号

DS:DI = 返回数据缓冲区地址(即指向上述的扩展驱动器参数数据结构)

//返回:

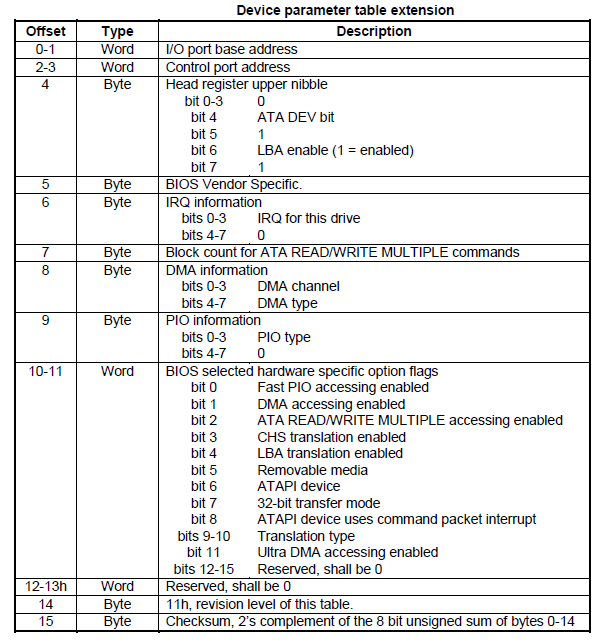
CF = 0, AH = 0 成功 DS:DI 驱动器参数数据包地址

CF = 1, AH = 错误码

注意在扩展驱动器参数中的DPTE 指针，只有在InfoSize 设置为30 或者更大，并且硬盘支

持增强磁盘驱动器（现在一般的硬盘都支持）时，才能够获得，否则此值会设为FFFF:FFFFh

指示无效。以下为DPTE 指向的参数表：



图DPTE(Device parameter table extension)

图中给出的Byte是8位，Word为16位。我们需要注意的是偏移0和2给出的I/O Baseaddress(I/O基址)和Control port address(控制端口地址)。在实际应用中，IDE主硬盘的I/O基址一般是1F0h，控制端口地址为3f6h；从盘为170h和376h。但是SATA硬盘一般不是这么确定的，我们就可以通过DPTE给出的参数表获取当前访问硬盘的I/O基址和控制端口地址。

从扩展int 13h给出的资料可以看出，功能48h提供的硬盘参数还是比较有限，比如硬盘序列号就无法获取。如果需要获取硬盘更详细的信息，就必须使用I/O指令来获取了。

第七节 通过I/O 指令访问硬盘

为便于下面的说明，我们简要介绍硬盘控制器的端口（IO\_BASE\_ADDR表示硬盘控制器的I/O基址）：



表1 硬盘控制器I/O 端口说明

获取硬盘参数的I/O 指令

查阅ATA的标准协议，可知获取硬盘信息的命令是ECh。往命令寄存器发送ECh后，可以通过数据端口寄存器获取相应的驱动器信息。驱动器信息由512个字节给出，其主要内容有（以字节偏移计算）：

偏移0：介质信息

偏移2：总逻辑磁柱数

偏移20：硬盘序列号码，共20个ASCII字符

偏移46：固件版本

从上面的讨论中我们可以明显看出，通过I/O指令获取的磁盘信息比BIOS中断获取的信息要更为详细。

一般主板有2个IDE通道（是硬盘的I/O控制器），每个通道可以接2个IDE硬盘。第一个IDE通道通过访问I/O地址0x1f0-0x1f7来实现，第二个IDE通道通过访问0x170-0x17f实现。每个通道的主从盘的选择通过第6个I/O偏移地址寄存器来设置。具体参数见下表。

I/O地址 功能

0x1f0 - 数据寄存器。读数据，当0x1f7不为忙状态时，可以读。

1F1 - 错误寄存器，每一位代表一类错误。全零表示操作成功。

0x1f2 - 扇区计数。要读写的扇区数，每次读写前，需要指出要读写几个扇区。

0x1f3 如果是LBA模式，就是LBA参数的0-7位

0x1f4 如果是LBA模式，就是LBA参数的8-15位

0x1f5 如果是LBA模式，就是LBA参数的16-23位

0x1f6 第0~3位：如果是LBA模式就是24-27位 第4位：为0主盘；为1从盘

第6位：为1=LBA模式；0 = CHS模式 第7位和第5位必须为1

0x1f7 状态和命令寄存器。操作时先给命令，再读取内容；如果不是忙状态就从0x1f0端口读数据

1F7 (写) - 命令寄存器

1F7 (读) - 状态寄存器

bit 7 = 1 控制器忙

bit 6 = 1 驱动器就绪

bit 5 = 1 设备错误

bit 4 N/A

bit 3 = 1 扇区缓冲区错误

bit 2 = 1 磁盘已被读校验

bit 1 N/A

bit 0 = 1 上一次命令执行失败

硬盘数据是储存到硬盘扇区中，一个扇区大小为512字节。读一个扇区的流程大致为通过outb指令访问I/O地址:0x1f2~-0x1f7来发出读扇区命令，通过in指令了解硬盘是否空闲且就绪，如果空闲且就绪，则通过inb指令读取硬盘扇区数据都内存中。可进一步参看bootmain.c中的readsect函数实现来了解通过PIO方式访问硬盘扇区的过程。

端口寄存器

|  |  |  |  |
| --- | --- | --- | --- |
| IO端口 |  | 端口用途 |  |
| Primary | Secondary | 读操作 | 写操作 |
| Command Block register: | 向硬盘驱动器中写入命令字或者获取硬盘状态 |  |  |
| 0x1F0 | 0x170 | Data | Data(读取或写入数据) |
| 0x1F1 | 0x171 | Error(读取失败时记录失败信息) | Features(存放命令的额外参数) |
| 0x1F2 | 0x172 | Sector count | Sector count(指定待读取或待写入的扇区数) |
| 0x1F3 | 0x173 | LBA low | LBA low(lba28 0~7位) |
| 0x1F4 | 0x174 | LBA mid | LBA mid(lba28 8~15位) |
| 0x1F5 | 0x175 | LBA high | LBA high(lba28 16~23位) |
| 0x1F6 | 0x176 | device | device(device寄存器是一个杂项:宽度8位) |
| 0x1F7 | 0x177 | status(硬盘状态寄存器) | command(命令写入该寄存器, 硬盘就开始工作了) |
| Control Block register | 控制硬盘工作状态 |  |  |
| 0x3F6 | 0x376 | Alternate status | Device Control |

device寄存器低4位用来存储LBA地址的第24~27位

第4位指定通道上的主盘或从盘

0代表主盘, 1代表从盘

也就是说无论是主板还是从盘,使用的端口都是一样的,只是device寄存器设置的不一样。

第6位用来设置是否启用LBA方式

0代表CHS方式, 1代表LBA方式

第5位和第7位是固定为1, 称为MBS位

(error,feature), (status, command) 这2组都是同一寄存器在不同操作下的多用途

写的时候硬盘控制器会认为是写入命令

读的时候硬盘控制器会认为是想获取硬盘的状态

device和status寄存器的状态

device寄存器 (位) 0 1 2 3 4 5 6 7

(开关) -LBA地址的第23~27位- 主/从盘 1 寻址模式 1

status寄存器 (位) 0 1 2 3 4 5 6 7

(开关) ERR DRQ DRDY BSY

(为1表示有错误

错误信息见error端口)

//IDE Status Register Bit

#define IDE\_BUSY 0x80 //系统忙标志

#define IDE\_DRDY 0x40 //系统ready标志

#define IDE\_DWF 0x20 //写误差

#define IDE\_DSC 0x10 //完成搜索标志，

#define IDE\_DRQ 0x08 //数据请求

#define IDE\_CORR 0x04 //修正数据

#define IDE\_INDEX 0x02 //

#define IDE\_ERROR 0x01

//IDE Command

#define IDE\_CMD\_CHECKPOWERMODE 0x98 // Check Power Mode 校核电源模式

#define IDE\_CMD\_CHECKPOWERMODE2 0xE5 //Check Power Mode (same as 98h)

#define IDE\_CMD\_DRIVEDIAG 0x90 // Execute Drive Diagnostic 可执行的设备诊断

#define IDE\_CMD\_FORMAT 0x50 // Format Track 格式轨迹

#define IDE\_CMD\_IDENTIFY 0xEC // Identify Drive 鉴别设备

#define IDE\_CMD\_IDLE 0x97 //Idle 空闲

#define IDE\_CMD\_IDLE2 0xE3 //Idle (same as 97h) |

#define IDE\_CMD\_IDLEIMMEDIATE 0x95 //Idle Immediate 立即空闲

#define IDE\_CMD\_IDLEIMMEDIATE2 0xE1 //Idle Immadiate (same as 95h) 立即空闲

#define IDE\_CMD\_INITDRIVERPARAM 0x91 //Initialize Drive Parameters 初始化设备参量

#define IDE\_CMD\_READBUFFER 0xE4 //Read Buffer 读buff|

#define IDE\_CMD\_READDMARETRY 0xC8 //Read DMA With Retry 读DMA | >> Unknown << |

#define IDE\_CMD\_READDMA 0xC9 //Read DMA | >> Unknown << |

#define IDE\_CMD\_READMILTI 0xC4 //Read Multiple 多次读|

#define IDE\_CMD\_READSECTORRETRY 0x20 //Read Sectors With Retry 再次读扇区 |

#define IDE\_CMD\_READSECTOR 0x21 //Read Sectors

#define IDE\_CMD\_READLONGRETRY 0x22 //Read Long With Retry

#define IDE\_CMD\_READLONG 0x23 //Read Long

#define IDE\_CMD\_READVERSECRETRY 0x40 //Read Verify Sectors With Retry 再次验证读扇区

#define IDE\_CMD\_READVERSEC 0x41 //Read Verify Sectors 读验证扇区

#define IDE\_CMD\_SETFEATURES 0xEF //Set Features 设置特色

#define IDE\_CMD\_SETMULTIMODE 0xC6 //Set Multiple Mode 多设置模式

#define IDE\_CMD\_SETSLEEPMODE 0x99 //Set Sleep Mode 设置睡眠模式

#define IDE\_CMD\_SETSLEEPMODE2 0xE6 //Set Sleep Mode (same as 99h)

#define IDE\_CMD\_STANDY 0x96 //Standby 待命

#define IDE\_CMD\_STANDY2 0xE2 //Standby (same as 96h)

#define IDE\_CMD\_STANDYIMMEDIATE 0x94 //Standby Immediate 直接待命

#define IDE\_CMD\_STANDYIMMEDIATE2 0xE0 //Standby Immediate (same as 94h)

对硬盘进行操作的一般顺序

先选择通道(Primary or Secondary),

往该通道的sector count寄存器中写入待操作的扇区数

往该通道上的三个LBA寄存器写入扇区起始地址的低24位

往device寄存器中写入LBA地址的24~27位, 并置第6位为1(LBA模式), 选择操作的硬盘(master or slave)

往该通道上的command寄存器写入操作命令

读取该通道上的status寄存器, 判断硬盘工作是否完成

如果以上步骤是读硬盘, 进入下一个步骤。否则, 完工

将硬盘数据读出

第八节 实战

硬盘读写的基本单位是扇区。就是说，要读就至少读一个扇区，要写就至少写一个扇区，不可能仅读写一个扇区中的几个字节。这样一来，就使得主机和硬盘之间的数据交换是成块的，所以硬盘是典型的块设备。

从硬盘读写数据，最经典的方式是向硬盘控制器分别发送磁头号、柱面号和扇区号（扇区在某个柱面上的编号），这称为 CHS 模式。这种方法最原始，最自然，也最容易理解。

最早的逻辑扇区编址方法是LBA28，使用 28 个比特来表示逻辑扇区号，从逻辑扇区 0x0000000 到 0xFFFFFFF，共可以表示 2 ^28＝268435456个扇区。每个扇区有512 字节，所以 LBA28 可以管理 128 GB 的硬盘。

但是硬盘技术发展得非常快，最新的硬盘已经达到几百个吉字节的容量，LBA28 已经落后了。在这种情况下，业界又共同推出了 LBA48，采用 48 个比特来表示逻辑扇区号。如此一来，就可以管理131072 TB 的硬盘容量了。在这里我们采用将采用 LBA28 来访问硬盘。

第 1 步：

设置要读取的扇区数量。这个数值要写入 0x1f2 端口。这是个 8 位端口，因此每次只能读写 255 个扇区：

mov dx,0x1f2

mov al,0x01 ;1 个扇区

out dx,al

注意：如果写入的值为 0，则表示要读取 256 个扇区。每读一个扇区，这个数值就减一。因此，如果在读写过程中发生错误，该端口包含着尚未读取的扇区数。

第 2 步：

设置起始 LBA 扇区号。扇区的读写是连续的，因此只需要给出第一个扇区的编号就可以了。28 位的扇区号太长，需要将其分成 4 段，分别写入端口 0x1f3、0x1f4、0x1f5 和 0x1f6 号端口。其中，0x1f3 号端口存放的是 0～7 位；0x1f4 号端口存放的是 8～15 位；0x1f5 号端口存放的是 16～23 位，最后 4 位在 0x1f6 号端口。假定我们要读写的起始逻辑扇区号为 0x02，可编写代码

如下：

mov dx,0x1f3

mov al,0x02

out dx,al ;LBA 地址 7～0

inc dx ;0x1f4

mov al,0x00

out dx,al ;LBA 地址 15～8

inc dx ;0x1f5

out dx,al ;LBA 地址 23～16

inc dx ;0x1f6

mov al,0xe0 ;LBA模式，主硬盘，以及 LBA 地址 27～24

out dx,al

第 3 步:

向端口 0x1f7 写入 0x20，请求硬盘读。这也是一个8 位端口：

mov dx,0x1f7

mov al,0x20 ;读命令

out dx,al

第 4 步:

等待读写操作完成。端口0x1f7 既是命令端口，又是状态端口。在通过这个端口发送读写命令之后，硬盘就忙乎开了。如图 8-12 所示，在它内部操作期间，它将 0x1f7 端口的第 7 位置“1”，表明自己很忙。一旦硬盘系统准备就绪，它再将此位清零，说明自己已经忙完了，同时将第 3位置“1”，意思是准备好了，请求主机发送或者接收数据（图 8-12）。完成这一步的典型代码如下:

mov dx,0x1f7

.waits:

in al,dx

and al,0x88

cmp al,0x08

jnz .waits

第 5 步:

连续取出数据。0x1f0 是硬盘接口的数据端口，而且还是一个 16 位端口。一旦硬盘控制器空闲，且准备就绪，就可以连续从这个端口写入或者读取数据。下面的代码假定是从硬盘读一个扇区（512 字节，或者 256 字节），读取的数据存放到由段寄存器 DS 指定的数据段，偏移地址由寄存器 BX 指定：

mov cx,256 ;总共要读取的字数

mov dx,0x1f0

.readw:

in ax,dx

mov [bx],ax

add bx,2

loop .readw

（完犊子）