

Ch 9: Input/Output System

输入/输出系统

第一讲 磁盘存储器和I/O接口

第二讲 I/O传输方式

第一讲 磁盘存储器和I/O接口

主 要 内 容

- I/O系统概述
 - I/O系统的性能
 - I/O系统的功能
 - OS在I/O系统中的角色
- I/O设备概述
 - I/O设备的通用模型
- 磁盘存储器
 - 回顾：磁盘存储器的读写原理
 - 回顾：磁盘存储器的性能指标
 - 冗余磁盘阵列 (RAID)
- I/O接口的分类
- I/O控制器的结构
- I/O控制器的职能
- I/O端口的概念
- I/O设备的寻址
- I/O设备、I/O总线、I/O接口、I/O控制器的连接

I/O System的性能

- 两个常用的性能指标:

- **Throughput: I/O bandwidth** (吞吐率, 即: **I/O带宽**):

- 单位时间内从系统输入/输出多少数据?
 - 单位时间内实现了多少次输入/输出操作?

(服务器更关注)

- **Response time: Latency** (响应时间, 即: 等待延迟):

- 在多长时间时间内完成请求的任务?

(台式机和嵌入式更关注)

- 不同的任务对性能的要求不同:

- 要求吞吐量高的场合:

- 如: 多媒体应用 (音/视频的播放要流畅!)

- 要求响应时间短的场合:

- 如: 事务处理系统 (存/取款的速度要快!)

- 要求吞吐率高且响应时间短的场合:

- **ATM**、文件服务器、**Web**服务器等

I/O System的功能

- 输入/出系统的功能:

- 解决各种形式信息的输入和输出

即：用户如何将所需的信息（文字、图表、声音、视频等）通过不同的外设输入到计算机中，以及计算机内部处理的结果信息如何通过相应的外设输出给用户

- 要实现上述功能需解决以下一系列的问题:

- 怎样在**CPU**、主存和外设间建立一个高效信息传输“通路”;
- 怎样将用户的**I/O**请求转换成设备的命令;
- 如何对外设进行编址;
- 怎样使**CPU**方便地寻找到要访问的外设;
- **I/O**硬件和操作系统如何协调完成主机和外设之间的数据传送等等

以上是本章的主要内容

外设发展与分类

◦ 从交互方式上来分，外设分为：

- 人-机交互设备

- 输入/输出的信息是人可读的
- 如：键盘、鼠标、扫描仪、打印机、显示器等

- 机器可读设备

- 输入/输出的信息是机器可读的，人无法读取
- 如：网络、**Modem**、**D/A**、**A/D**、磁盘、声音输入设备等

◦ 从功能行为来分，外设分为：

- 输入/输出设备

- 用于信息的输入/输出
- 输入设备：键盘、鼠标、扫描仪等
- 输出设备：打印机、显示器等

- 外部存储设备

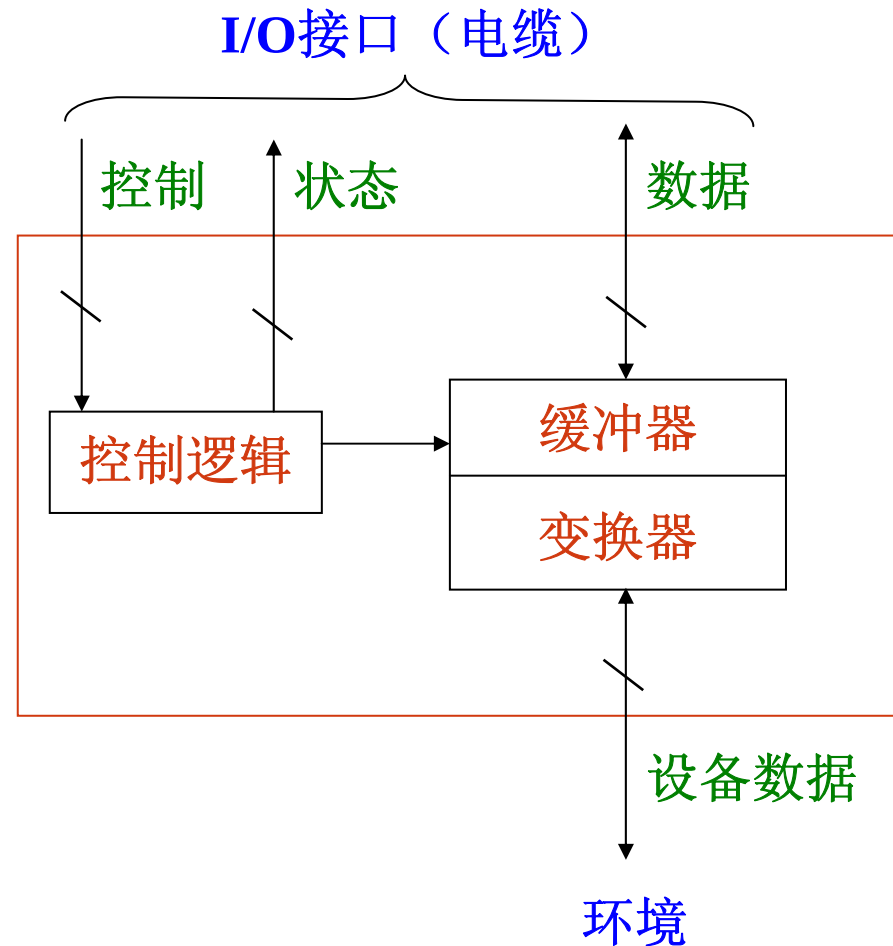
- 用于信息的存储（其输入/出的信息是机器可读的）
- 如：磁盘、磁带、光盘等

常用外部设备

- 输入设备：
 - 键盘、触摸屏
 - 图形输入设备(鼠标、图形板、跟踪球、操纵杆、光笔)
 - 图像输入设备(摄像机、扫描仪、传真机)
 - 条形码阅读机、光学字符识别设备(OCR)
 - 音、视频输入设备
- 输出设备：
 - 显示器(字符、汉字、图形、图像)
 - 打印设备(点阵、激光、喷墨)
 - 绘图仪(平板式、滚筒式)
 - 声音输出设备
- 其它：
 - 终端设备(键盘+显示器)
 - 外存储器(磁盘、磁带、光盘)

外部设备的通用模型

- 通过**电缆**与计算机内部I/O接口进行数据、状态和控制信息的传送
- 控制逻辑**根据控制信息控制设备的操作，并检测设备状态
- 缓冲器**用于保存交换的数据信息
- 变换器**用于在电信号形式（内部数据）和其他形式的设备数据之间进行转换



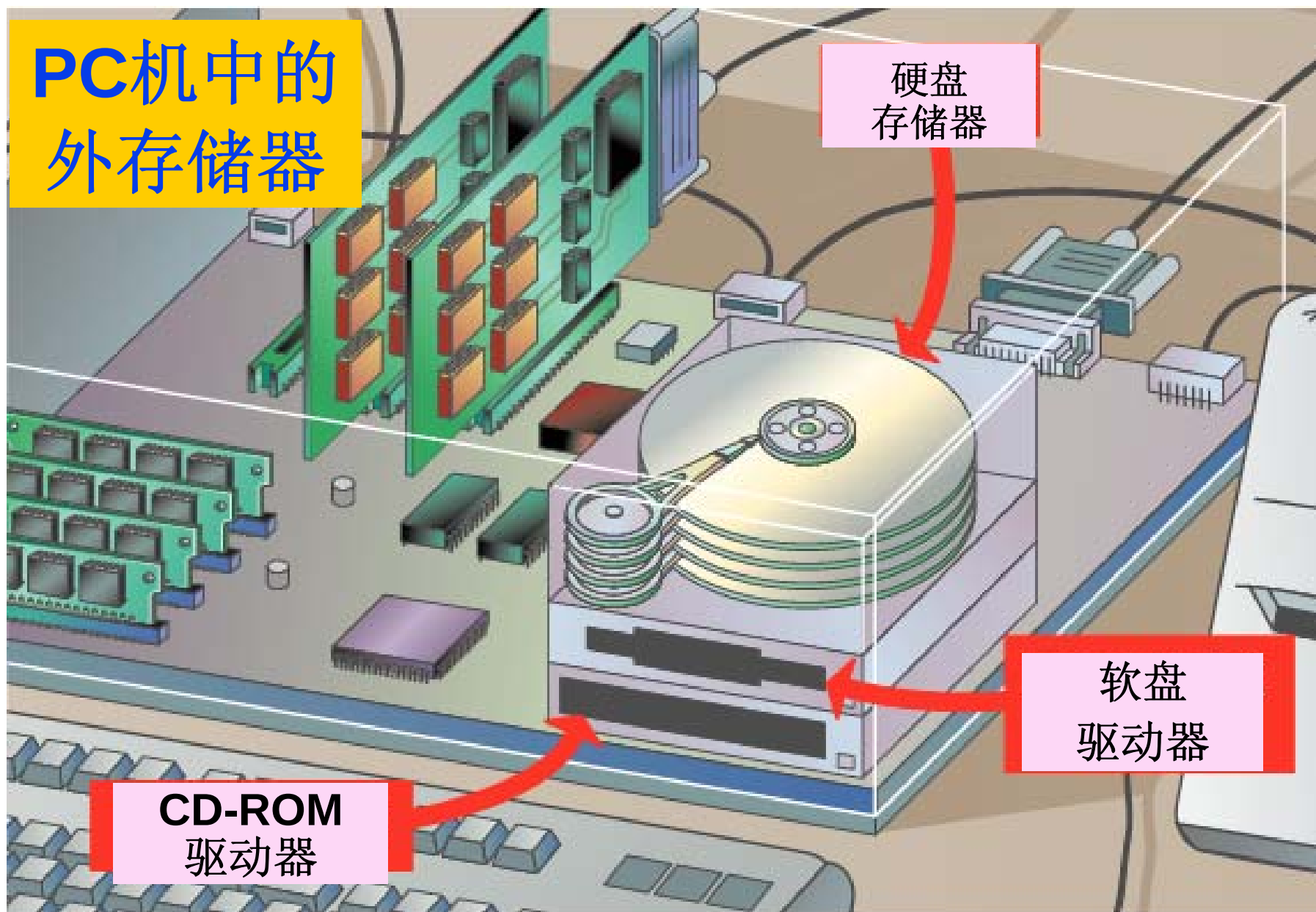
所有设备都可以抽象成这个通用模型！

设备所用的电缆线中有以下三种信号线：

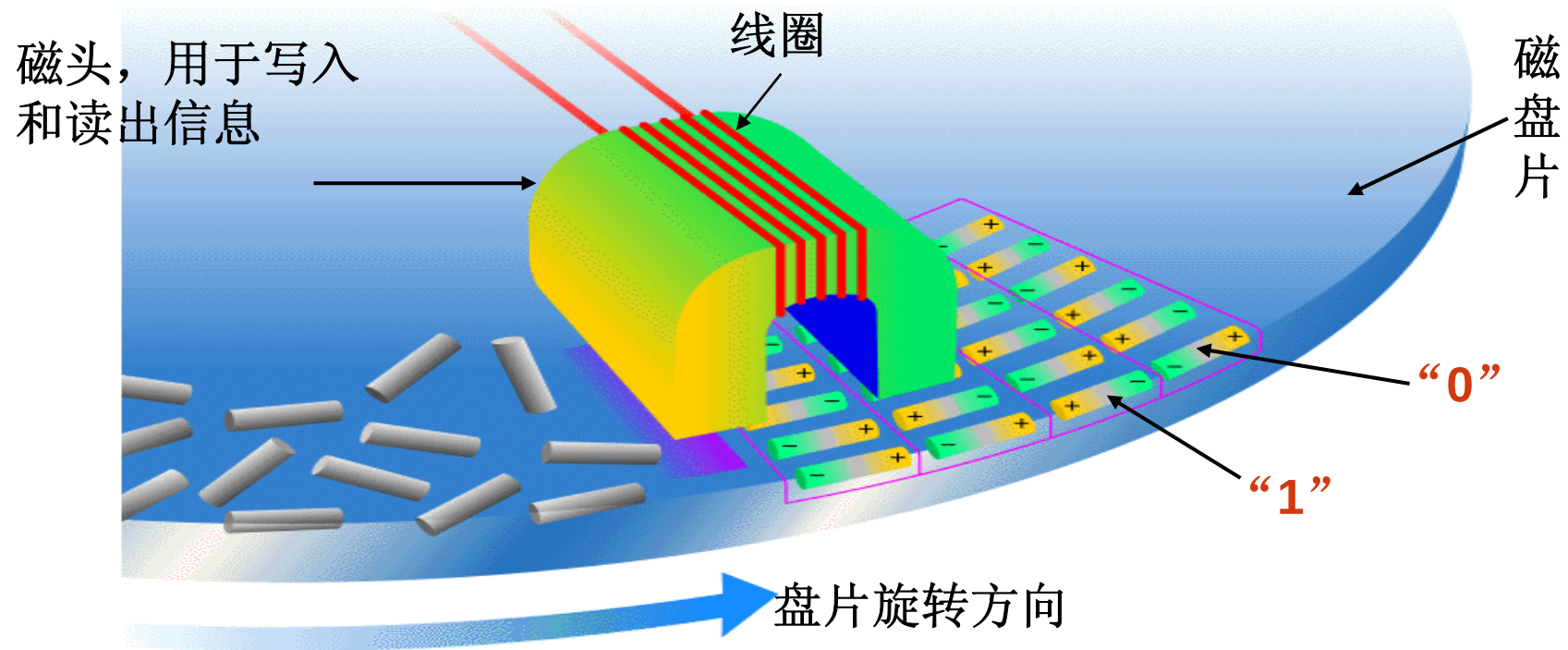
控制信号、状态信号、数据信号

下面以磁盘为例，说明外部设备的工作原理

回顾：PC中的外存储器



回顾：磁盘存储器的信息存储原理



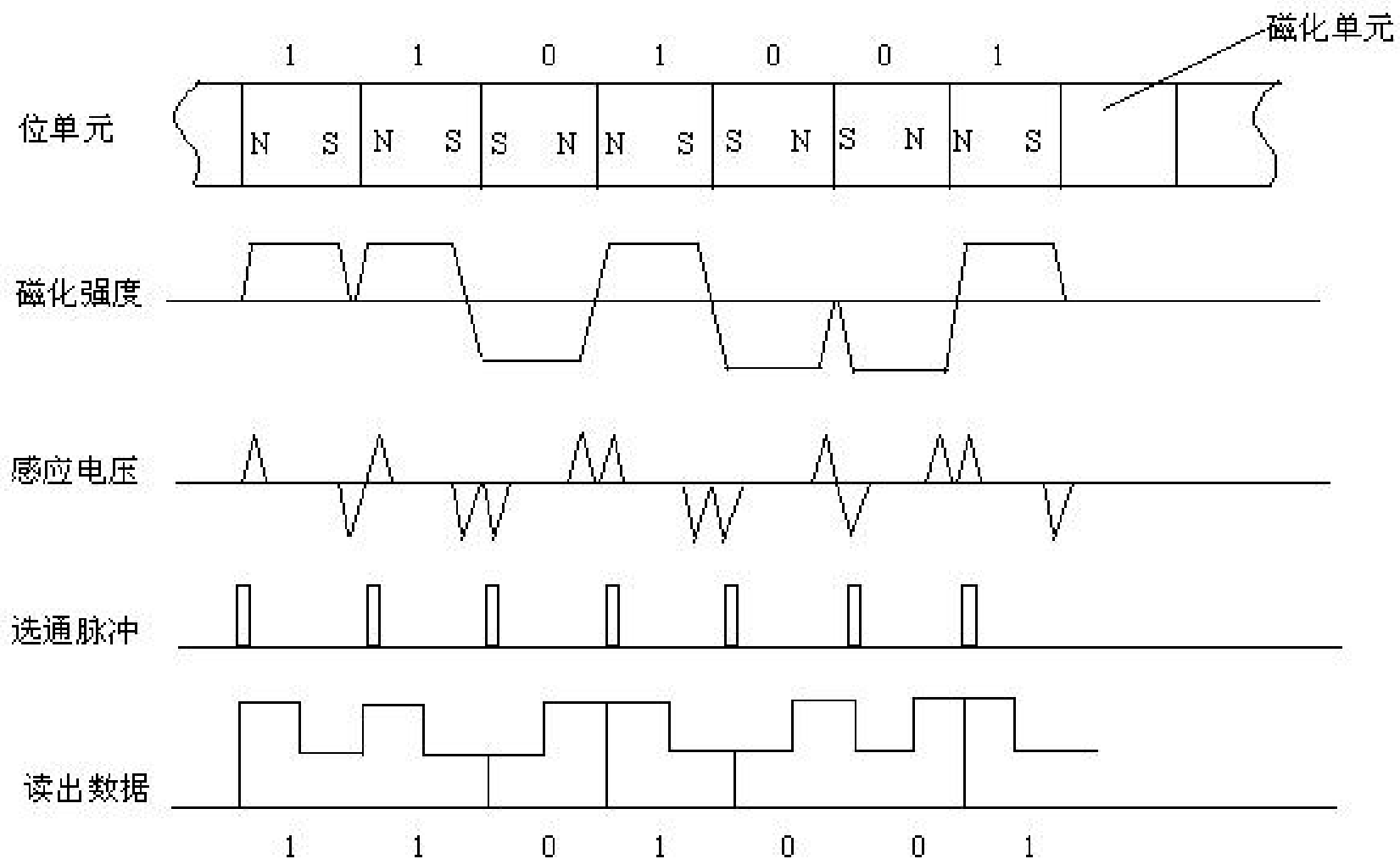
写1：线圈通以正向电流，使呈N-S状态

写0：线圈通以反向电流，使呈S-N状态

} 不同的磁化状态被记录在磁盘表面

读时：磁头固定不动，载体运动。因为载体上小的磁化单元外部的磁力线通过磁头铁芯形成闭合回路，在铁芯线圈两端得到感应电压。根据不同的极性，可确定读出为0或1。

(自学) 磁表面信息读出过程



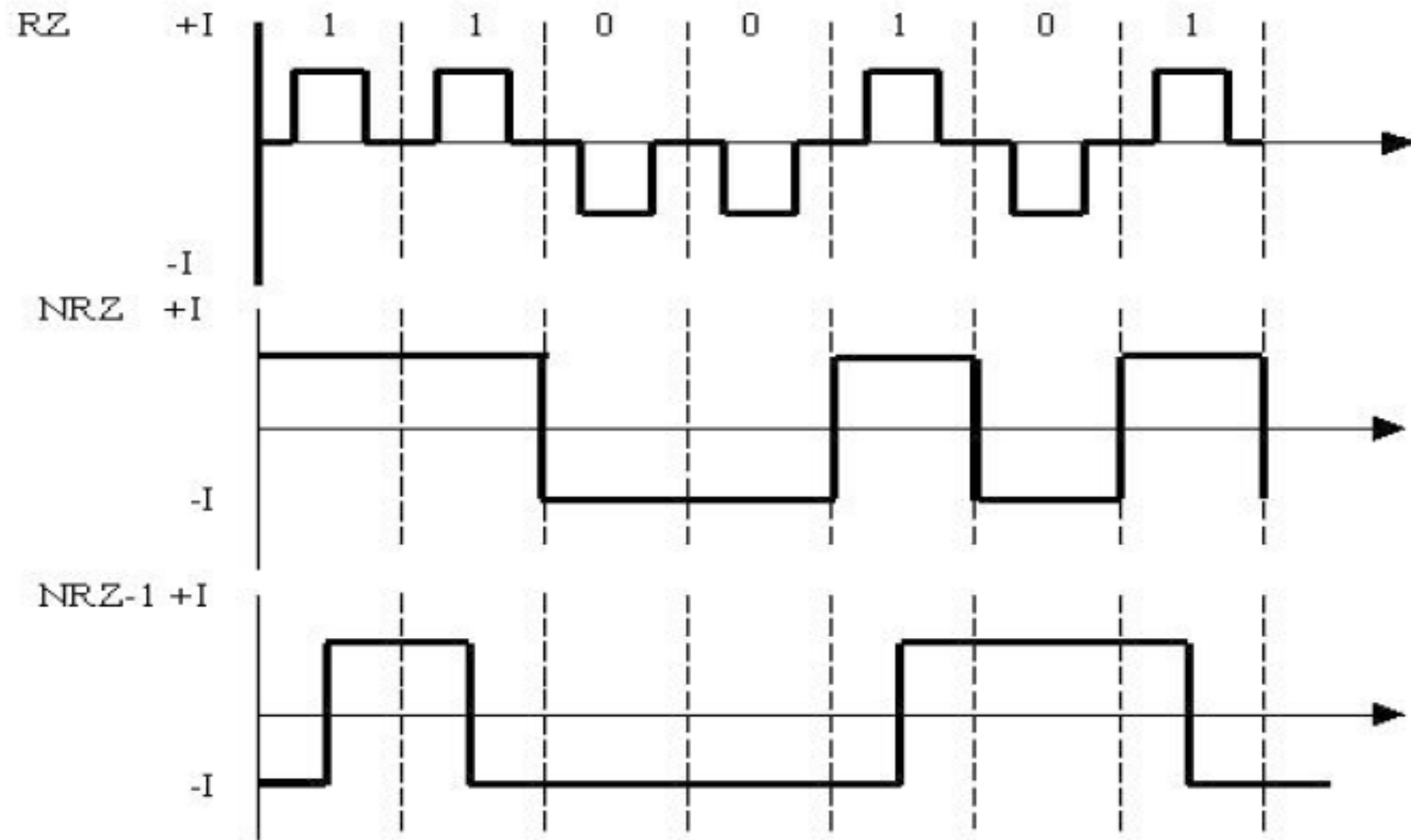
（自学）磁表面记忆原理和记录方式

° 什么叫磁记录方式？有哪几种？

- 对数字信息按一定的规律进行编码，变成相应的写电流序列，通过磁头造成磁层表面上的磁化翻转单元，从而把信息写到磁表面的方式
- 数据记录方式按照写电流波形的极性、频率和相位的不同有：
 - 归零制（RZ）：写1用正脉冲，写0用负脉冲，一位信息写完后，电流总回到零，又叫双向归零制或典型归零制
 - 不归零1制（NRZ-1）：写电流只在写1时改变方向，写0时写电流不变，故又称为“见1就翻”不归零制。各信息位间也无“间隙”，记录密度较高。存1才能读出信号，存0无读出信号，故无自同步能力
 - 调相制（PM）：利用写电流相位的不同实现写1和写0，写0时，先正后负，写1时，先负后正。无论写1还是写0，在一个位信息期间，写电流相位至少有一次改变
 - 调频制（FM）：写0和1时，写电流频率不相同。写1翻转2次（在位单元的前沿和数据位中央各一次），写0翻转1次（在位单元的前沿）
 - 改进调频制（MFM）：逢1在位中央翻转一次；独立一个0不翻转；两个0在位之间翻转一次

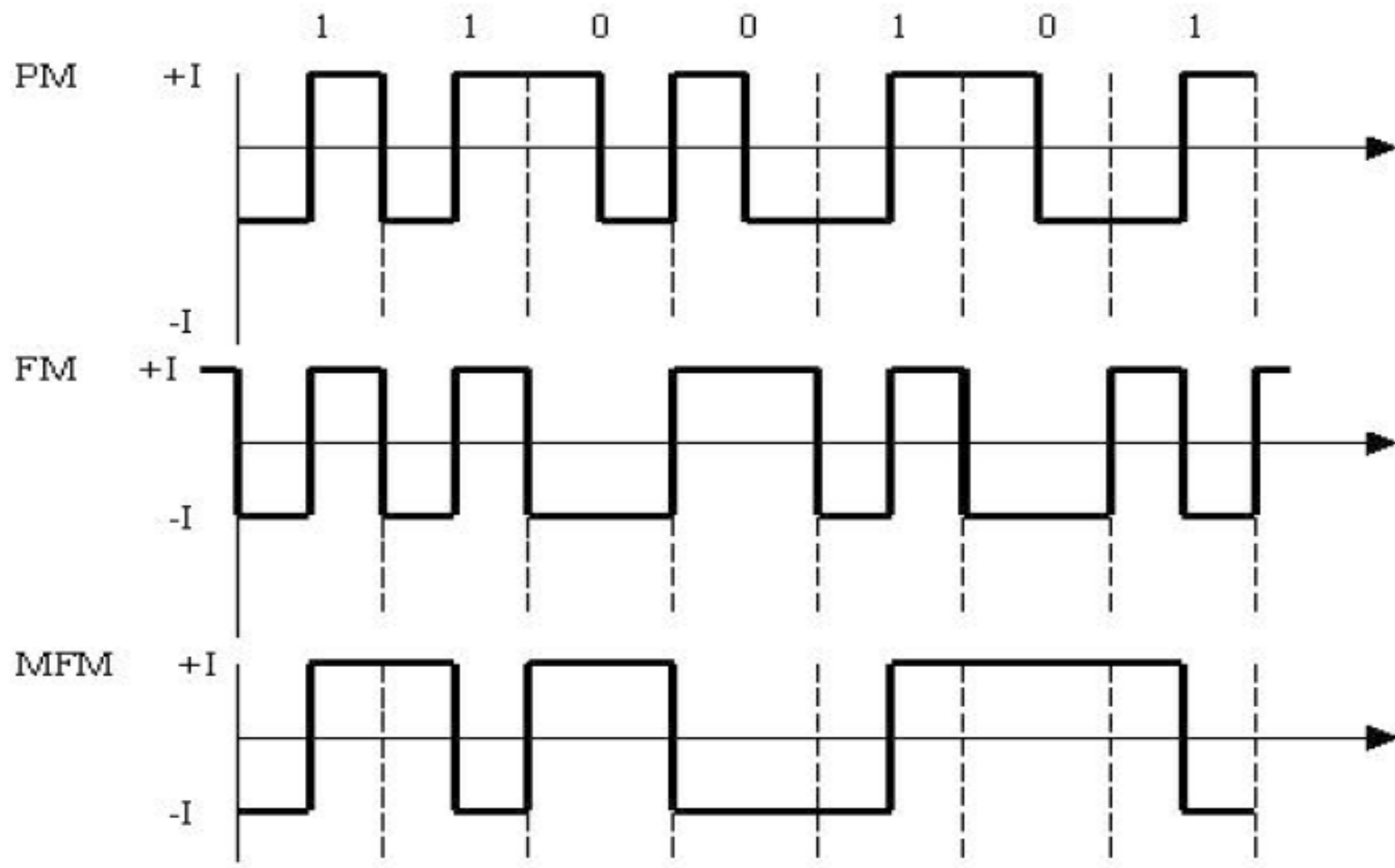
SKIP

(自学) 磁记录方式1



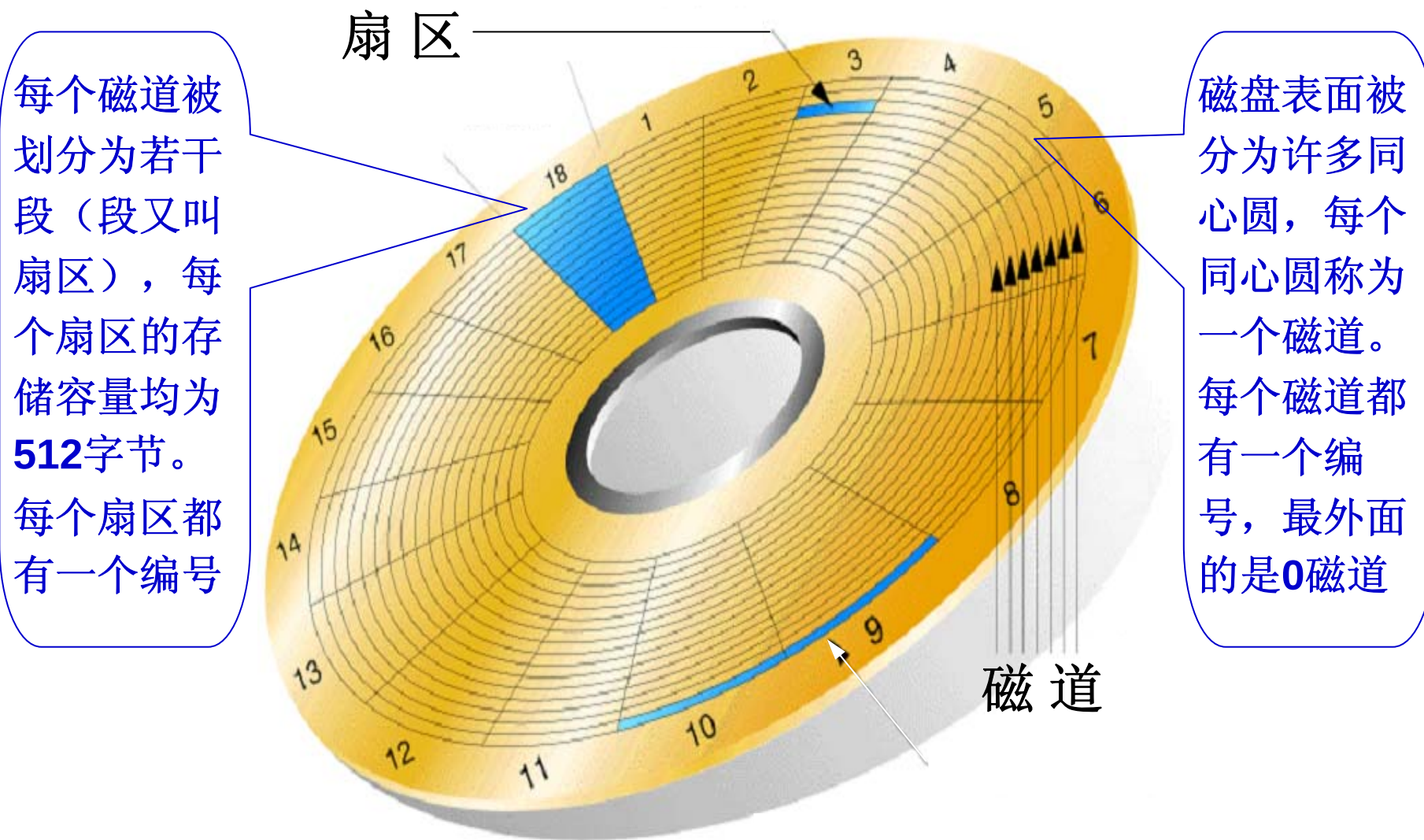
[BACK](#)

(自学) 磁记录方式2



[BACK](#)

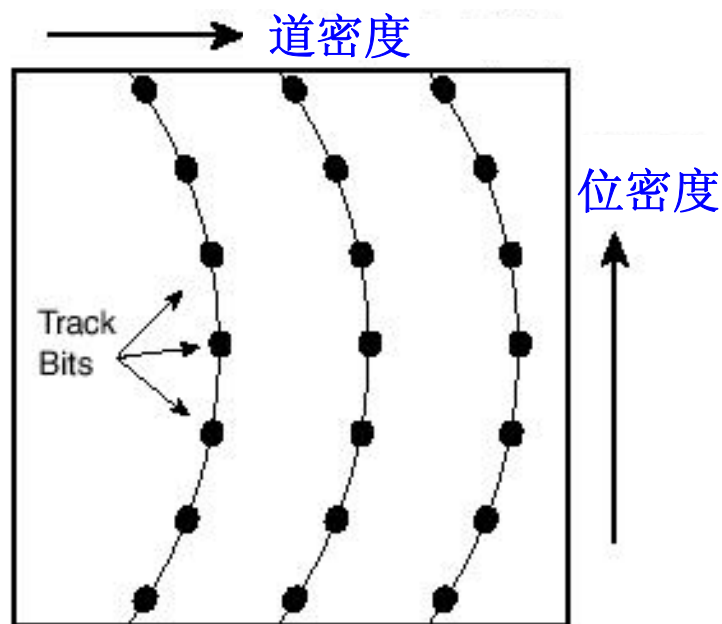
磁盘的磁道和扇区



注：所谓磁盘的格式化操作，就是在盘面上划分磁道和扇区，并在扇区中填写扇区号等信息的过程

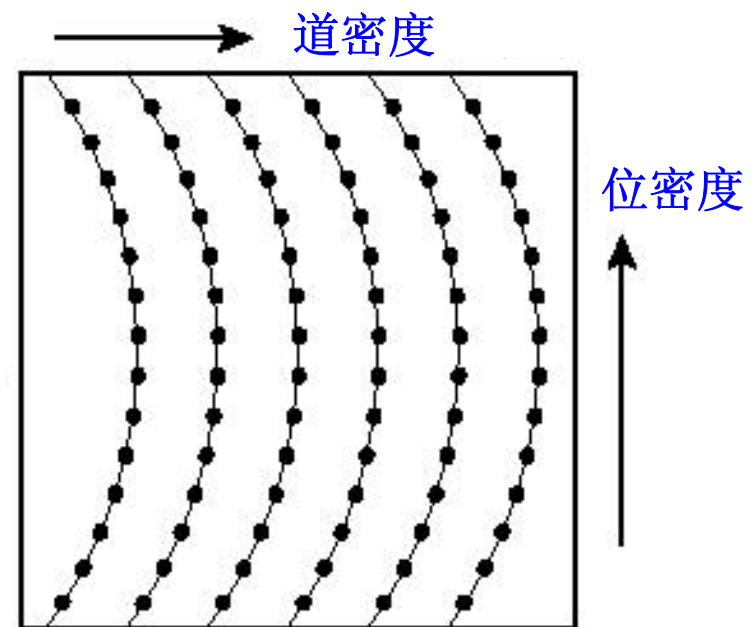
如何增大磁盘片的容量？

- 提高盘片上的信息记录密度！
 - 增加磁道数目——提高磁道密度
 - 增加扇区数目——提高位密度，采用可变扇区数



低密度存储示意图

早期的磁盘所有磁道上的扇区数相同，所以位数相同，内道上的位密度比外道位密度高



高密度存储示意图

现代磁盘磁道上的位密度相同，所以，外道上的扇区数比内道上扇区数多，使整个磁盘的容量提高

磁盘磁道的格式

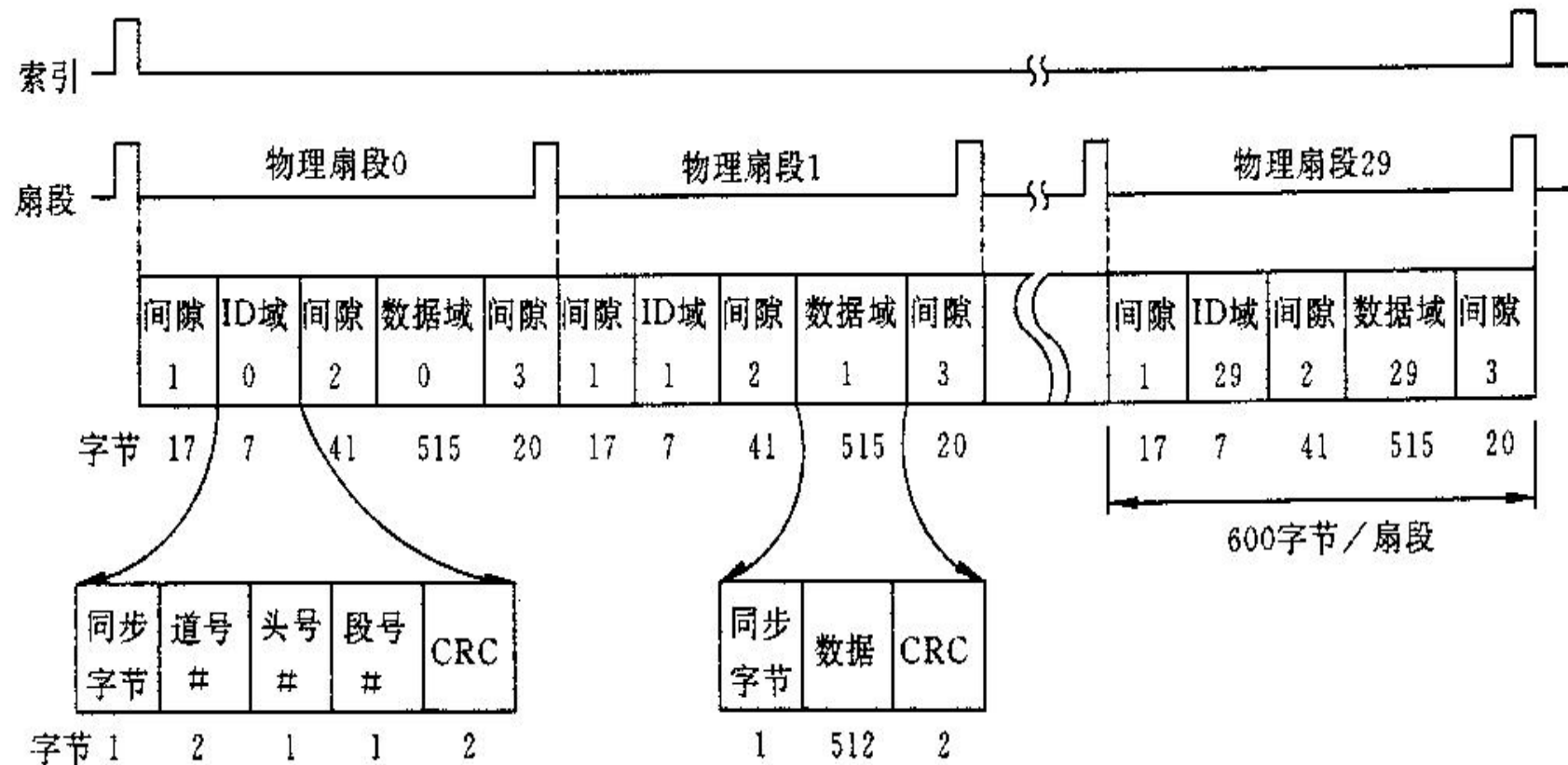
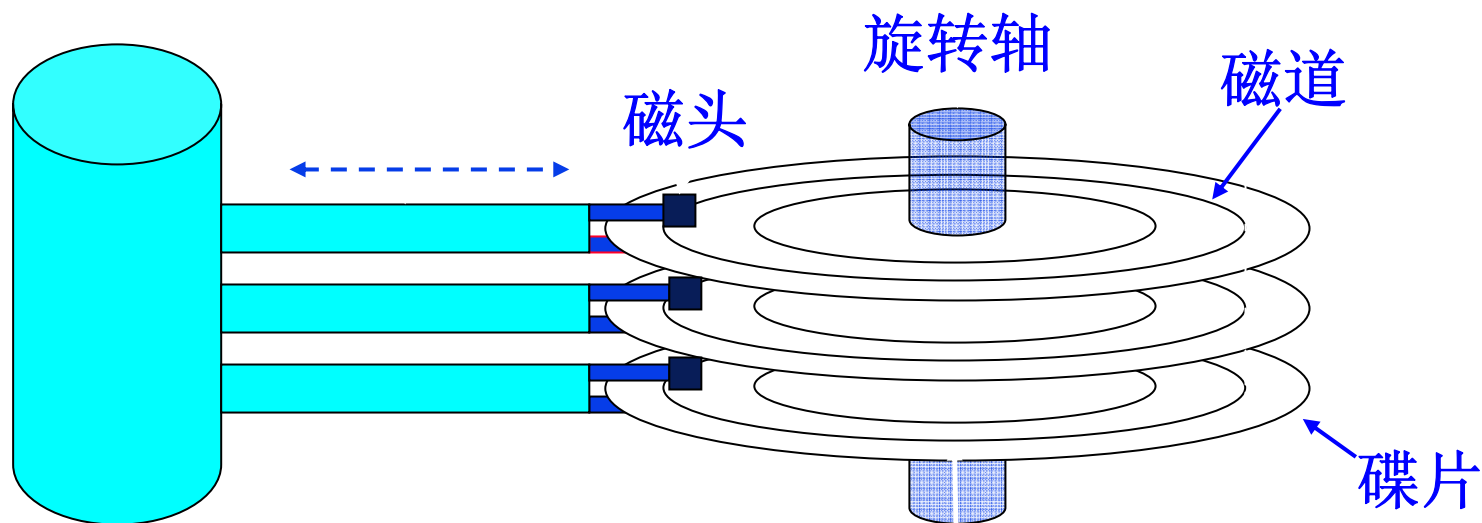


图 5.2 温彻斯特磁盘磁道格式(Seagate ST506)

在此例中，每个磁道包含30个固定长度的扇段，每个扇段有600个字节 (17+7+41+515+20=600)。

平均存取时间



硬盘的操作流程如下：

所有磁头同步寻道（由柱面号控制）→ 选择磁头（由磁头号控制）→
被选中的磁头等待扇区到达磁头下方（由扇区号控制）→ 读写该扇区中的数据

◦ 磁盘上的信息以扇区为单位进行读写，平均存取时间为：

$T = \text{寻道时间} + \text{旋转等待时间} + \text{数据传输时间}$

- 寻道时间——磁头寻找到指定磁道所需时间(大约5ms)
- 旋转等待时间——指定扇区旋转到磁头下方所需要的时间(大约4~6ms)（转速：
4200 / 5400 / 7200 / 10000rpm）
- 数据传输时间——(大约0.01ms / 扇区)

磁盘响应时间计算举例

- 假定每个扇区**512**字节， 磁盘转速为**5400 RPM**， 广告声称寻道时间为**12 ms**， 数据传输率为**4 MB/s**， 磁盘控制器开销为**1 ms**， 不考虑排队时间， 则磁盘响应时间为多少？

$$\begin{aligned}\text{Disk Response Time} &= \text{Seek time} + \text{Rotational Latency} + \text{Transfer time} \\ &\quad + \text{Controller Time} + \text{Queuing Delay} \\ &= 12 \text{ ms} + 0.5 / 5400 \text{ RPM} + 0.5 \text{ KB} / 4 \text{ MB/s} + 1 \text{ ms} + 0 \\ &= 12 \text{ ms} + 0.5 / 90 \text{ RPS} + 0.125 / 1024 \text{ s} + 1 \text{ ms} + 0 \\ &= 12 \text{ ms} + 5.5 \text{ ms} + 0.1 \text{ ms} + 1 \text{ ms} + 0 \text{ ms} \\ &= 18.6 \text{ ms}\end{aligned}$$

如果实际的寻道时间只有1/3的话， 则为**10.6ms**， 这样旋转等待时间就占了近50%！

$$12/3 + 5.5 + 0.1 + 1 = 10.6 \text{ ms}$$

所以， 磁盘转速非常重要！

为什么实际的寻道时间只有1/3？

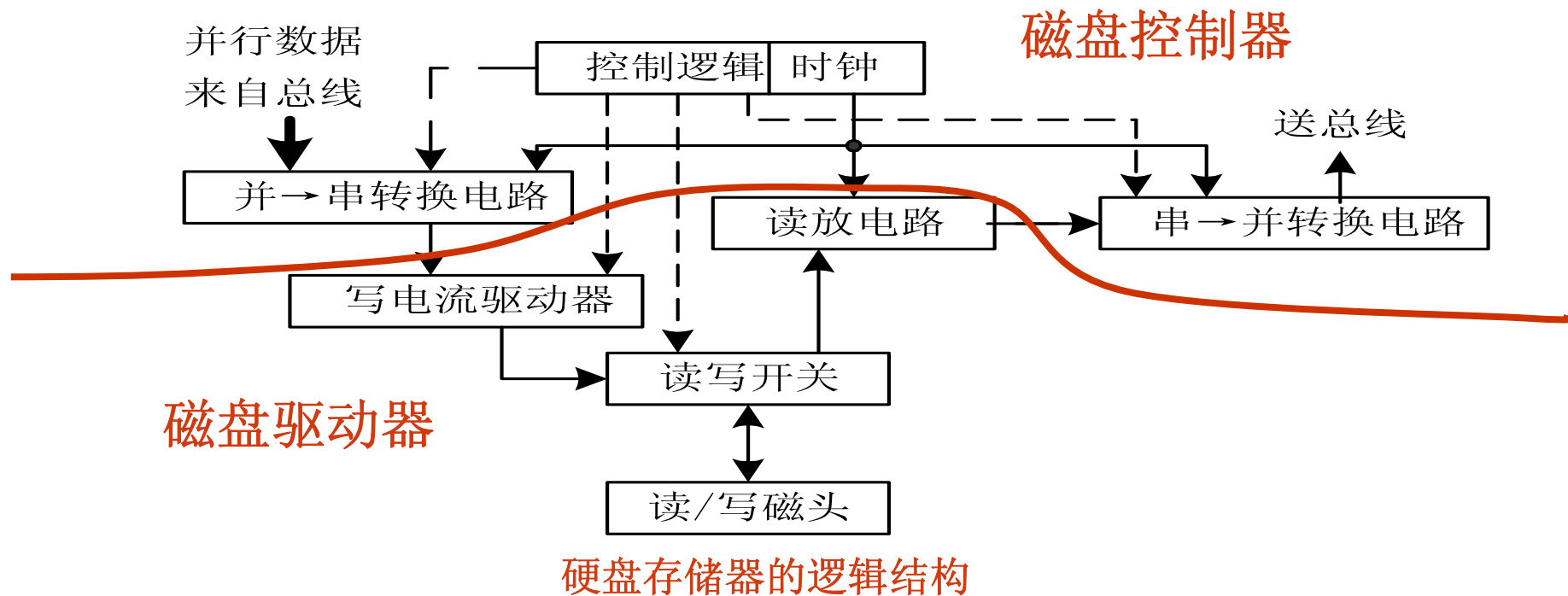
访问局部性使得每次磁盘访问大多在局部磁道！

能否算出每道有多少扇区？ $4\text{MB} \times 60 / 512\text{B} \times 5400 \approx 87$ 个扇区

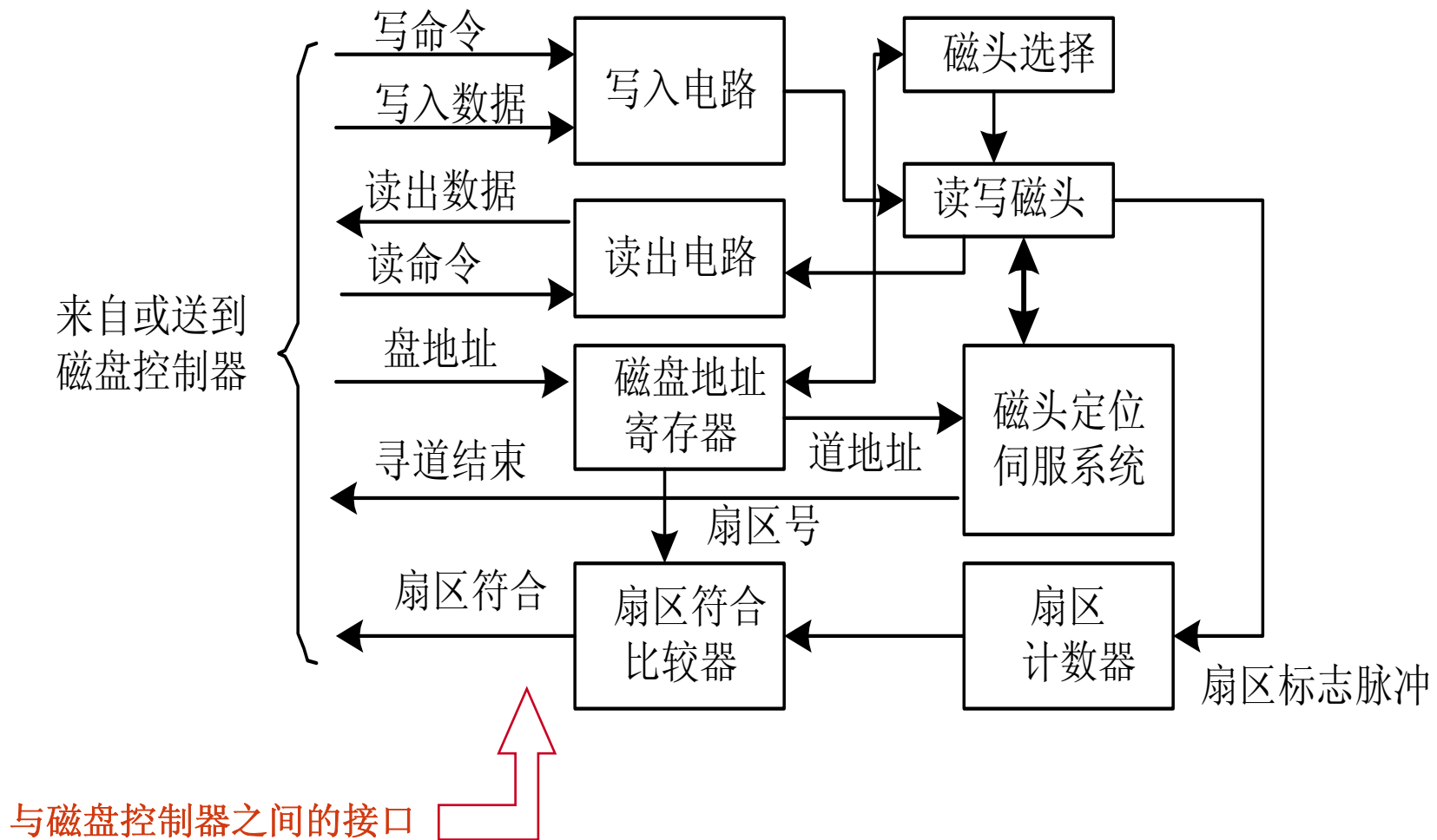
硬盘存储器的组成

° 硬盘存储器的基本组成

- 磁记录介质：用来保存信息
- 磁盘驱动器：包括读写电路、读\写转换开关、读写磁头与磁头定位伺服系统等
- 磁盘控制器：包括控制逻辑、时序电路、“并→串”转换和“串→并”转换电路等。
(用于连接主机与盘驱动器)



硬盘驱动器的逻辑结构

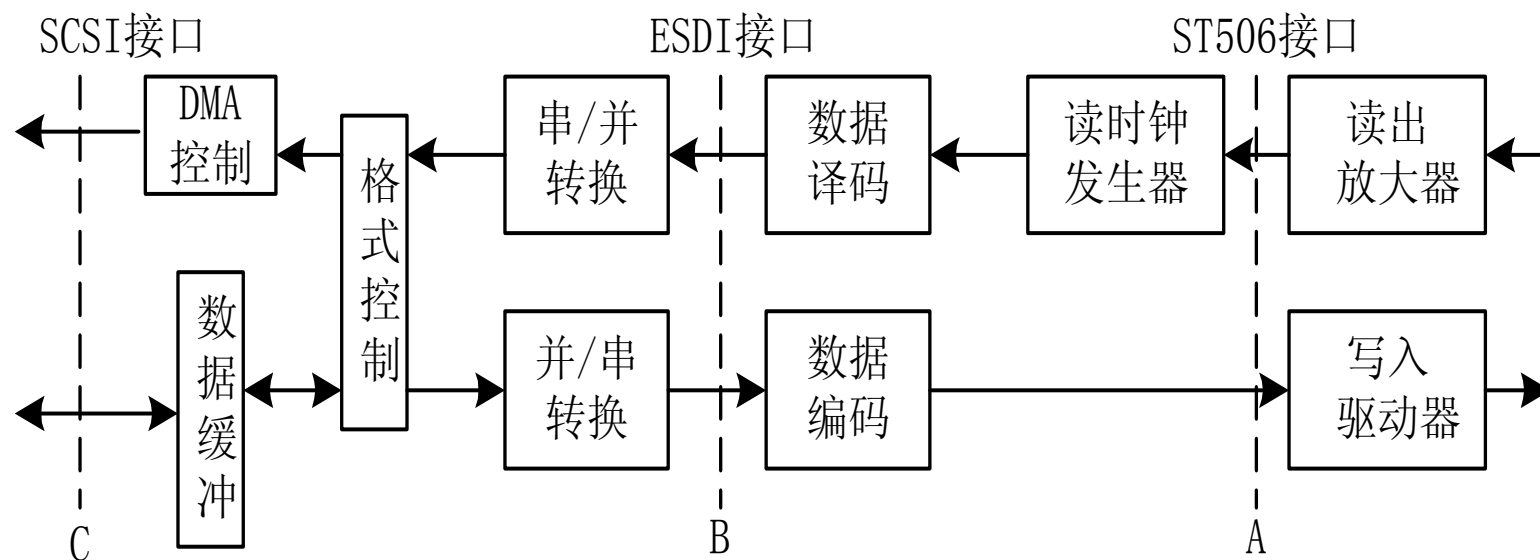


硬盘控制器的逻辑结构

- 磁盘控制器是主机与磁盘驱动器之间的接口
- 磁盘控制器与磁盘驱动器之间并没有明确的界线
(可以在 **A点 / B点 / C点**)

通过总线与主机连接

通过接口电缆与磁盘驱动器连接



服务器大多使用**SCSI**接口

PC机前几年大多使用**IDE(ATA)**接口

近两年**PC**机开始大量使用**SATA**接口

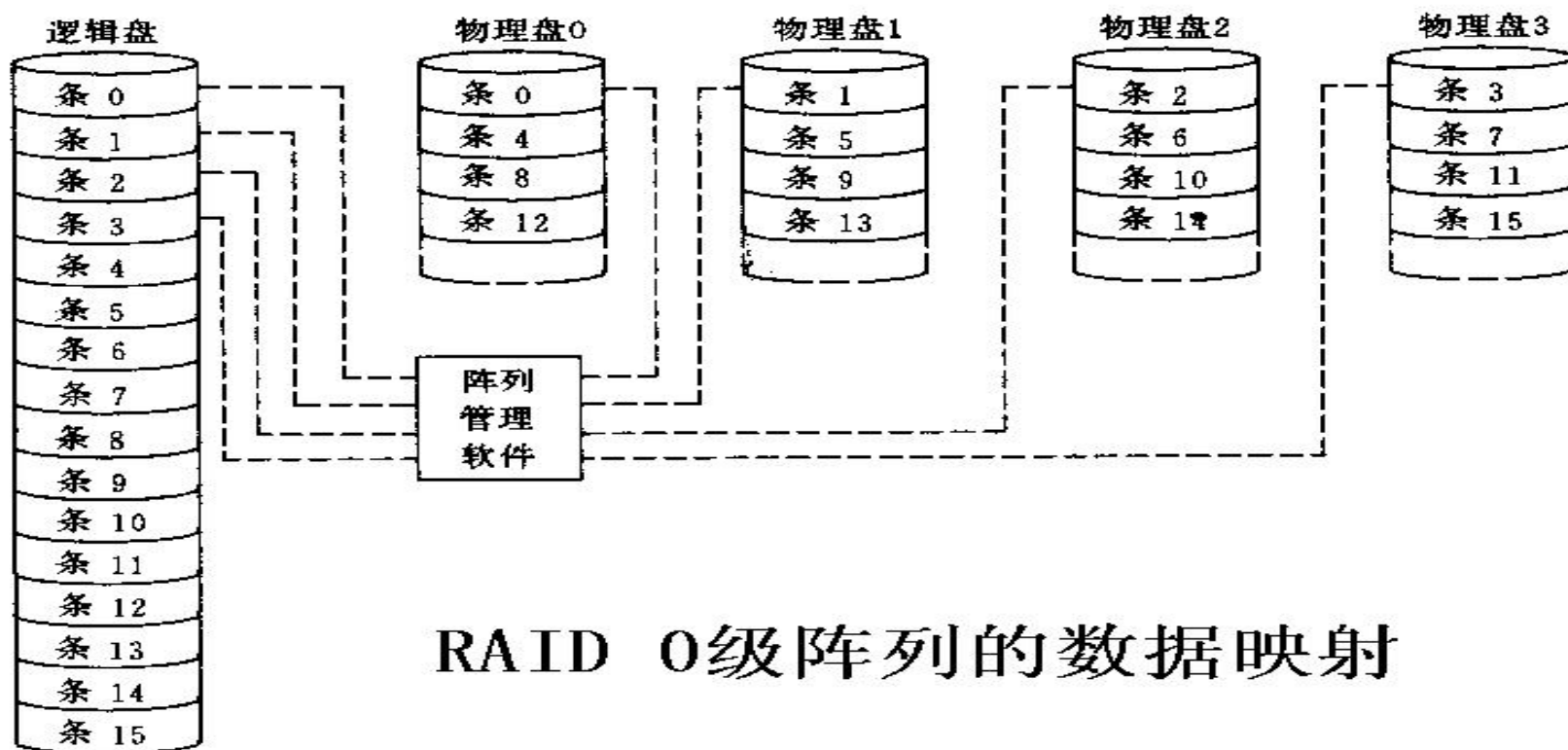
冗余磁盘阵列(RAID)

- 系统总体性能的提高不匹配
 - 处理器和主存性能改进快
 - 辅存性能性能改进慢
- 所用措施: **RAID-Redundant Arrays of Inexpensive Disk** (冗余磁盘阵列)
- **RAID**的基本思想:

将多个独立操作的磁盘按某种方式组织成磁盘阵列(**Disk Array**), 以增加容量, 利用类似于主存中的多体交叉技术, 将数据存储在多个盘体上, 通过使这些盘并行工作来提高数据传输速度, 并用冗余(**redundancy**)磁盘技术来进行错误恢复(**error correction**)以提高系统可靠性。
- **RAID**特性:
 - (1) **RAID**是一组物理磁盘驱动器, 在操作系统下被视为一个单个逻辑驱动器。
 - (2) 数据分布在一组物理磁盘上。
 - (3) 冗余磁盘容量用于存储奇偶校验信息, 保证磁盘万一损坏时能恢复数据。
- **RAID**级别
 - 目前已知的**RAID**方案分为8级(0-7级), 以及**RAID10**(结合0和1级)和**RAID30**(结合0和3级)和 **RAID50**(结合0和5级)。但这些级别不是简单地表示层次关系, 而是表示具有上述3个共同特性的不同设计结构。

冗余磁盘阵列(RAID0)

- 不遵循特性(3)，所以无冗余。适用于容量和速度要求高的非关键数据存储的场合
 - 与单个大容量磁盘相比有两个优点：
 - (1) 连续分布或大条区交叉分布时，如果两个I/O请求访问不同盘上的数据，则可并行发送。减少了I/O排队时间。具有较快的I/O响应能力。
 - (2) 小条区交叉分布时，同一个I/O请求有可能并行传送其不同的数据块(条区)，因而可达较高的数据传输率。例如，可以用在视频编辑和播放系统中，以快速传输视频流



RAID 0级阵列的数据映射

冗余磁盘阵列(RAID1)

- 镜像盘实现1对1冗余(100% redundancy)

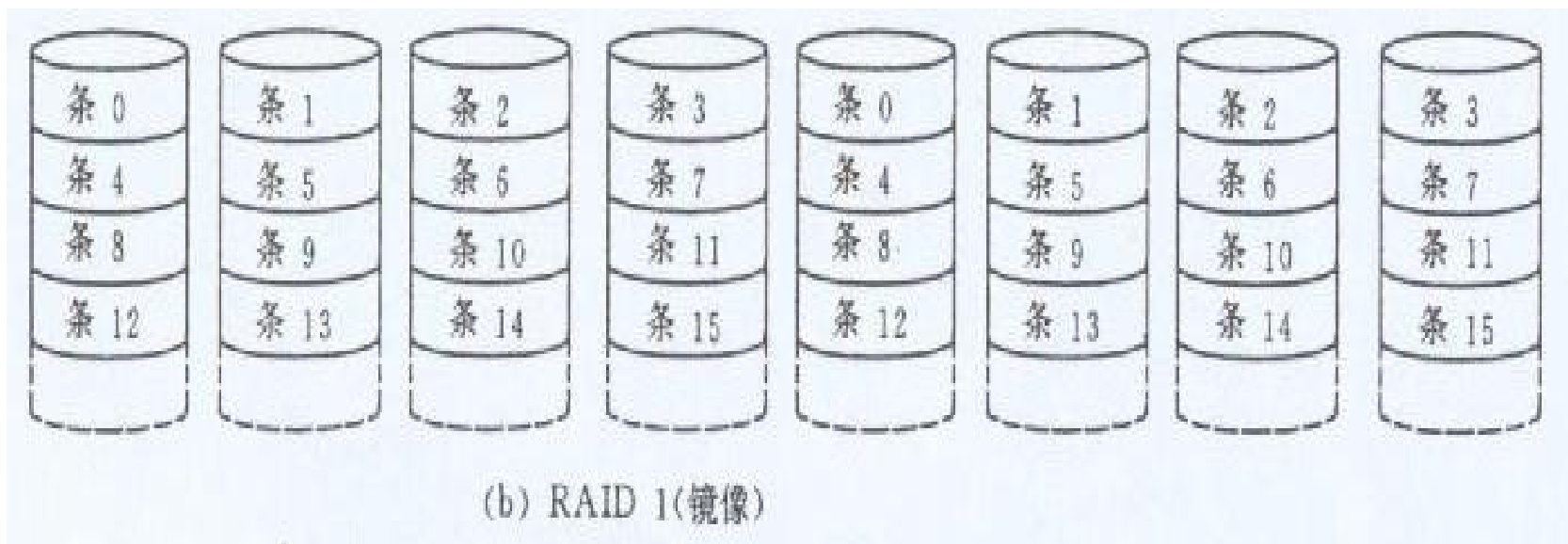
- (1) 一个读请求可由其中一个定位时间更少的磁盘提供数据。

- (2) 一个写请求对对应的两个磁盘并行更新。故写性能由两次中较慢的一次写来决定，即定位时间更长的那一次。

- (3) 数据恢复很简单。当一个磁盘损坏时，数据仍能从另一个磁盘中读取。

- 特点：可靠性高，但价格昂贵。

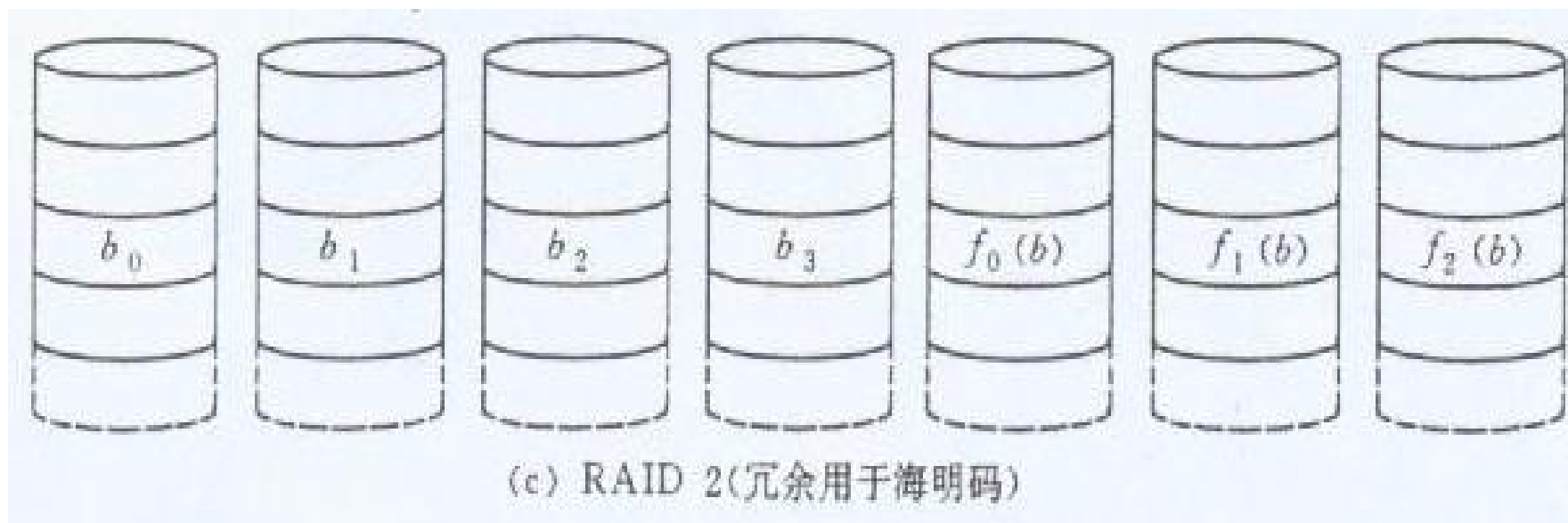
常用于可靠性要求很高的场合，如系统软件的存储，金融、证券等系统。



冗余磁盘阵列(RAID2)

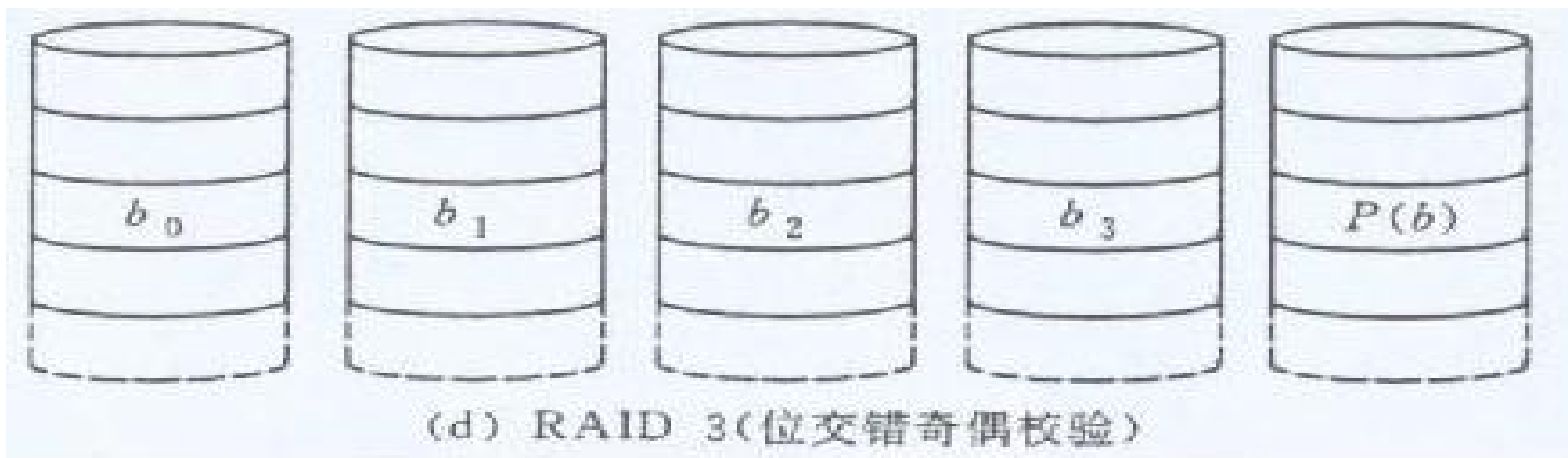
- 用海明校验法生成多个冗余校验盘，实现纠正一位错误、检测两位错误的功能。
- 采用条区交叉分布方式，且条区非常小（有时为一个字或一个字节）。这样，可获得较高的数据传输率，但I/O响应时间差。
- 采用海明码，虽然冗余盘的个数比RAID1少，但校验盘与数据盘成正比。所以冗余信息开销还是太大，价格也较贵
- 读操作性能高（多盘并行）。
- 写操作时要同时写数据盘和校验盘。

RAID2已不再使用！



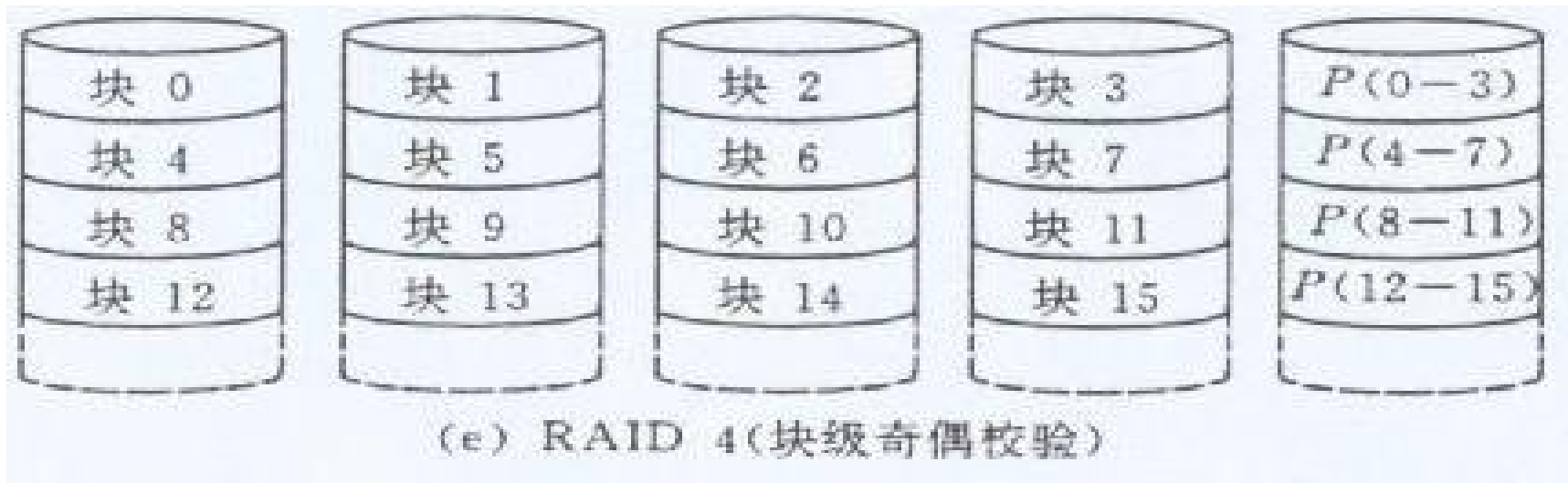
冗余磁盘阵列(RAID3)

- 采用奇偶校验法生成单个冗余盘。
- 与**RAID 2**相同，也采用条区交叉分布方式，并使用小条区。这样，可获得较高的数据传输率，但**I/O**响应时间差。
- 用于大容量的 **I/O**请求的场合，如：图像处理、**CAD** 系统中。
- 某个磁盘损坏但数据仍有效的情况，称为简化模式。此时损坏的磁盘数据可以通过其它磁盘重新生成。数据重新生成非常简单，这种数据恢复方式同时适用于**RAID 3、4、5**级。



冗余磁盘阵列(RAID4)

- 用一个冗余盘存放相应块（较大的数据条区）的奇偶校验位。
- 采用独立存取技术，每个磁盘的操作独立进行，所以可同时响应多个I/O请求。因而它适合于要求I/O响应速度快的场合。
- 对于写操作，校验盘成为I/O瓶颈，因为每次写都要对校验盘进行。
 - 少量写（只涉及个别磁盘）时，有“写损失”，因为一次写操作包含两次读和两次写
 - 大量写（涉及所有磁盘的数据条区）时，则只需直接写入奇偶校验盘和数据盘。因为奇偶校验位可全部用新数据计算得到。而无须读原数据



RAID 3/4/5的数据恢复与RAID4的少量写（自学）

RAID3/4/5的数据恢复操作

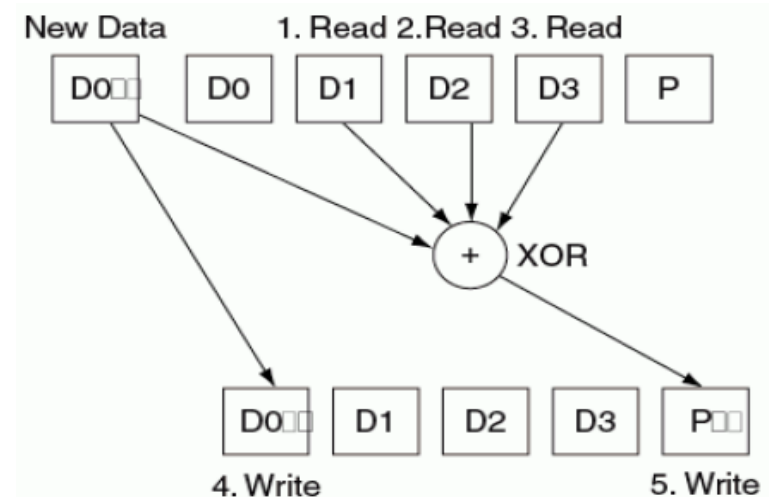
假定考虑一个有5个磁盘的阵列， X_0 到 X_3 保存数据， p 是奇偶校验盘，奇偶校验的第 i 位计算公式如下：

$$p(i) = X_3(i) \oplus X_2(i) \oplus X_1(i) \oplus X_0(i)$$

若磁盘 X_0 损坏，上述等式两边同时异或 $p(i) \oplus X_0(i)$ ，则得到以下等式：

$$X_0(i) = p(i) \oplus X_3(i) \oplus X_2(i) \oplus X_1(i)$$

因此，在阵列中，某一数据磁盘中的任何一个数据条区的内容都能从剩余磁盘的相应条区中重新生成



RAID4的少量写操作

假定考虑一个有5个磁盘的阵列， X_0 到 X_3 保存数据， P 是奇偶校验盘，初始时对每位 i 有下列关系式：

$$p(i) = X_3(i) \oplus X_2(i) \oplus X_1(i) \oplus X_0(i)$$

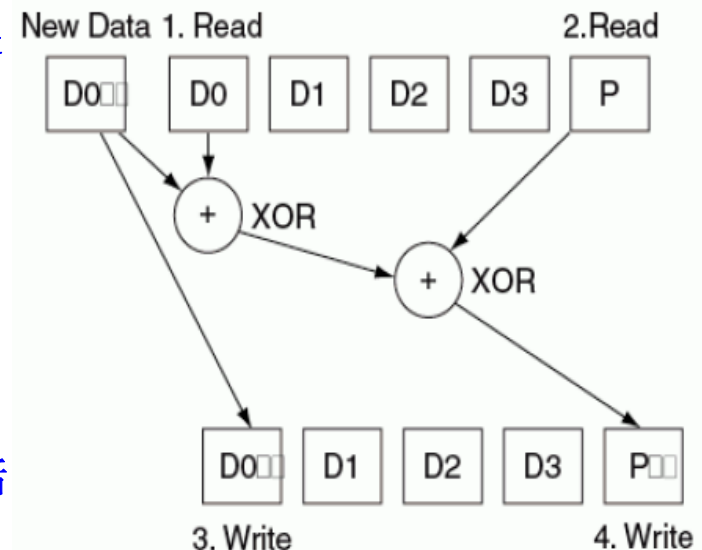
写操作后，可能改变的位数表示如下：

$$P'(i) = X_3(i) \oplus X_2(i) \oplus X_1(i) \oplus X'_0(i)$$

$$= X_3(i) \oplus X_2(i) \oplus X_1(i) \oplus X'_0(i) \oplus X_0(i) \oplus X_0(i)$$

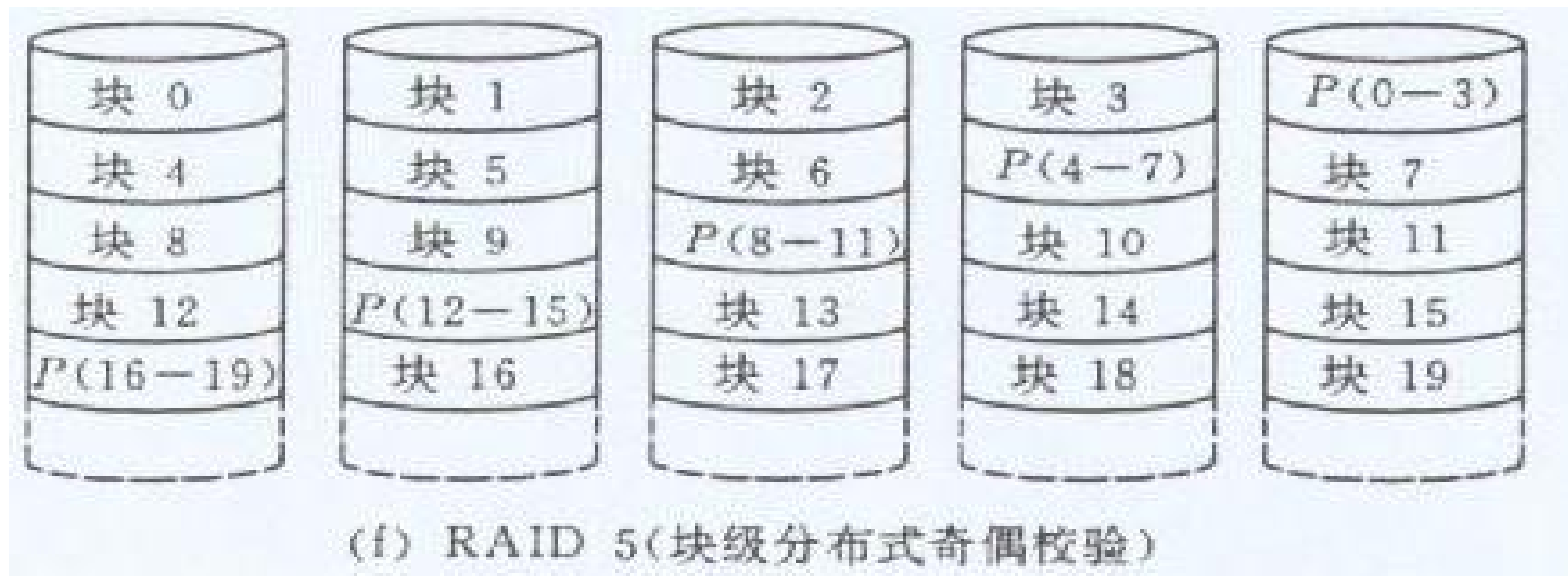
化简后得到： $p'(i) = p(i) \oplus X_0(i) \oplus X'_0(i)$

由此可见，要更新一个 $X_0(i)$ ，必须先读 $p(i)$ 和 $X_0(i)$ ，然后写 $X'_0(i)$ 和 $p'(i)$



冗余磁盘阵列(RAID5)

- 与RAID 4组织方式类似，只是奇偶校验块分布在各个磁盘中，所以所有磁盘地位等价，这样可提高容错性，并且避免了使用专门校验盘时潜在的I/O瓶颈。
- 与RAID 4一样，采用独立的存取技术，因而有较高的I/O响应速度。
- 小数据量的操作可以多个磁盘并行操作
- 成本不高但效率高，所以被广泛使用



P块为校验块，分布在不同的磁盘中

冗余磁盘阵列(RAID6)

- 冗余信息均匀分布在所有磁盘上，而数据仍以块交叉方式存放
- 双维块交叉奇偶校验独立存取盘阵列，容许双盘出错
- 它是对**RAID 5**的扩展，主要是用于要求数据绝对不能出错的场合
- 由于引入了第二种奇偶校验值，对控制器的设计变得十分复杂，写入速度也比较慢，用于计算奇偶校验值和验证数据正确性所花费的时间比较多
- **RAID 6**级以增大开销的代价保证了高度可靠性



RAID 6 级双维块交叉奇偶校验独立存取盘阵列示意图

P0代表第**0**条区的奇偶校验值，而**PA**代表数据块**A**的奇偶校验值

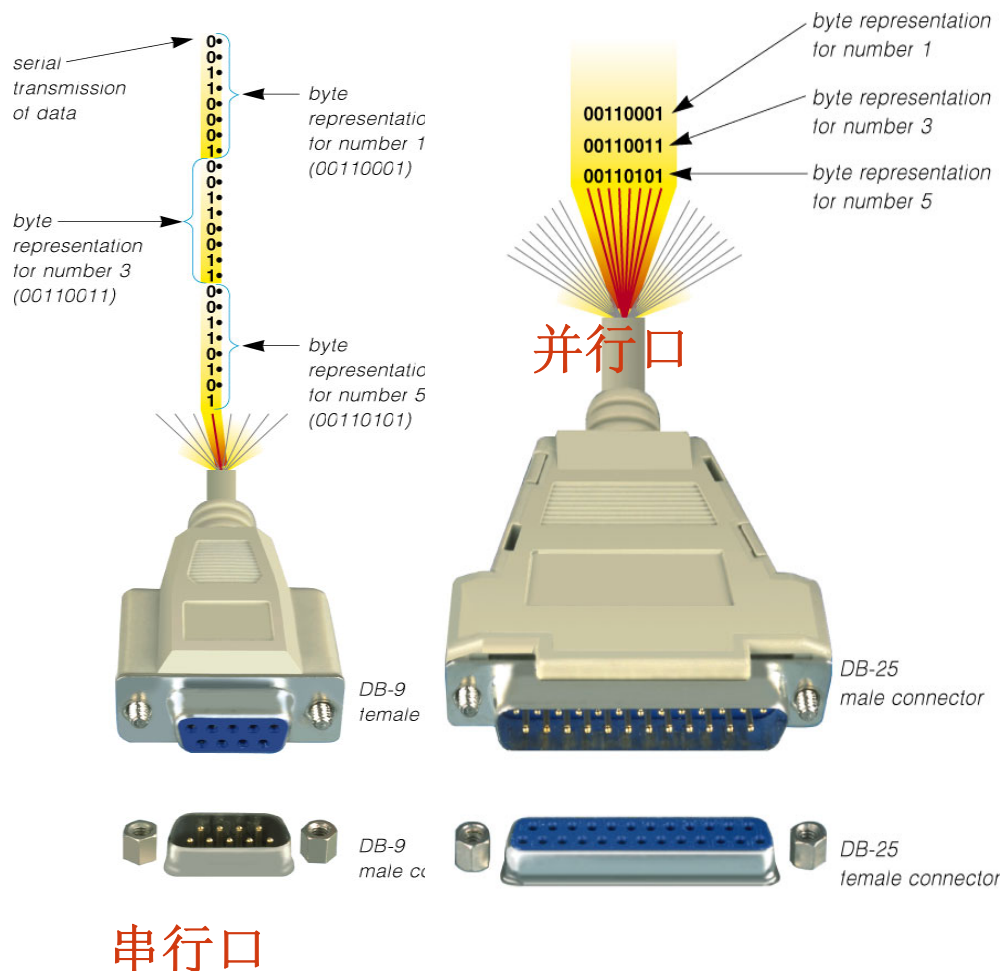
冗余磁盘阵列(RAID7)

- 带**Cache**的盘阵列
- 在**RAID6**的基础上，采用**Cache**技术使传输率和响应速度都有较大提高
- **Cache**分块大小和磁盘阵列中数据分块大小相同，一一对应
- 有两个独立的**Cache**，双工运行。在写入时将数据同时分别写入两个独立的**Cache**，这样即使其中有一个**Cache**出故障，数据也不会丢失
- 写入磁盘阵列以前，先写入**Cache**中。同一磁道的信息在一次操作中完成
- 读出时，先从**Cache**中读出，**Cache**中没有要读的信息时，才从**RAID**中读

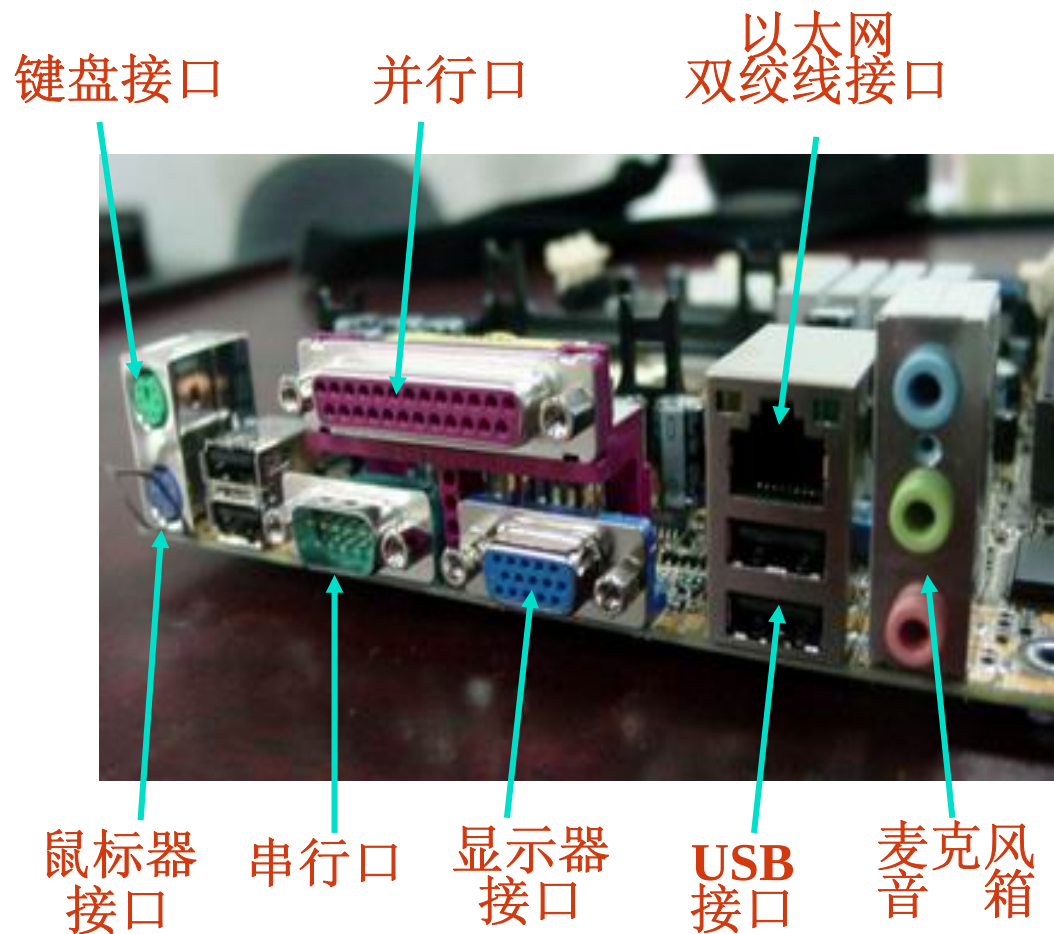
Cache和**RAID**技术结合，弥补了**RAID**的不足（如：分块写请求响应性能差等），从而以高效、快速、大容量、高可靠性，以及灵活方便的存储系统提供给用户

回顾：关于I/O接口

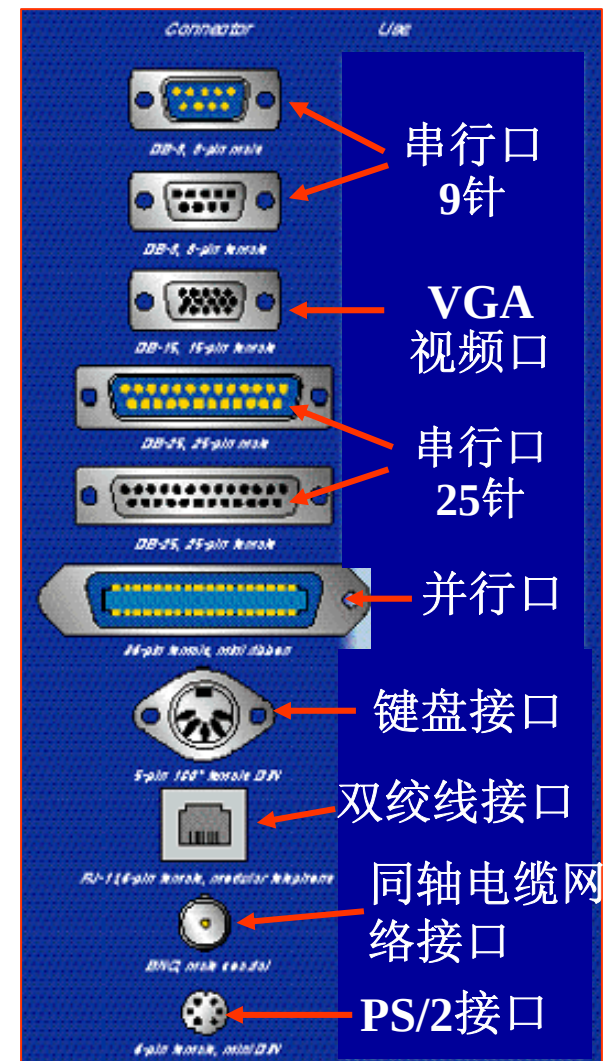
- ° I/O接口：I/O设备与I/O控制器之间的连接器
 - 包括：插头 / 插座的形式、通讯规程和电器特性等
- ° 分类：
 - 从数据传输方式来分：
 - 串行（一次只传输1位）
 - 并行（多位一起进行传输）
 - 从是否能连接多个设备来分：
 - 总线式（可连接多个设备）
 - 独占式（只能连接1个设备）
 - 从是否符合标准来分：
 - 标准接口（通用接口）
 - 专用接口（专用接口）
 - 按功能选择的灵活性来分：
 - 可编程接口
 - 不可编程接口



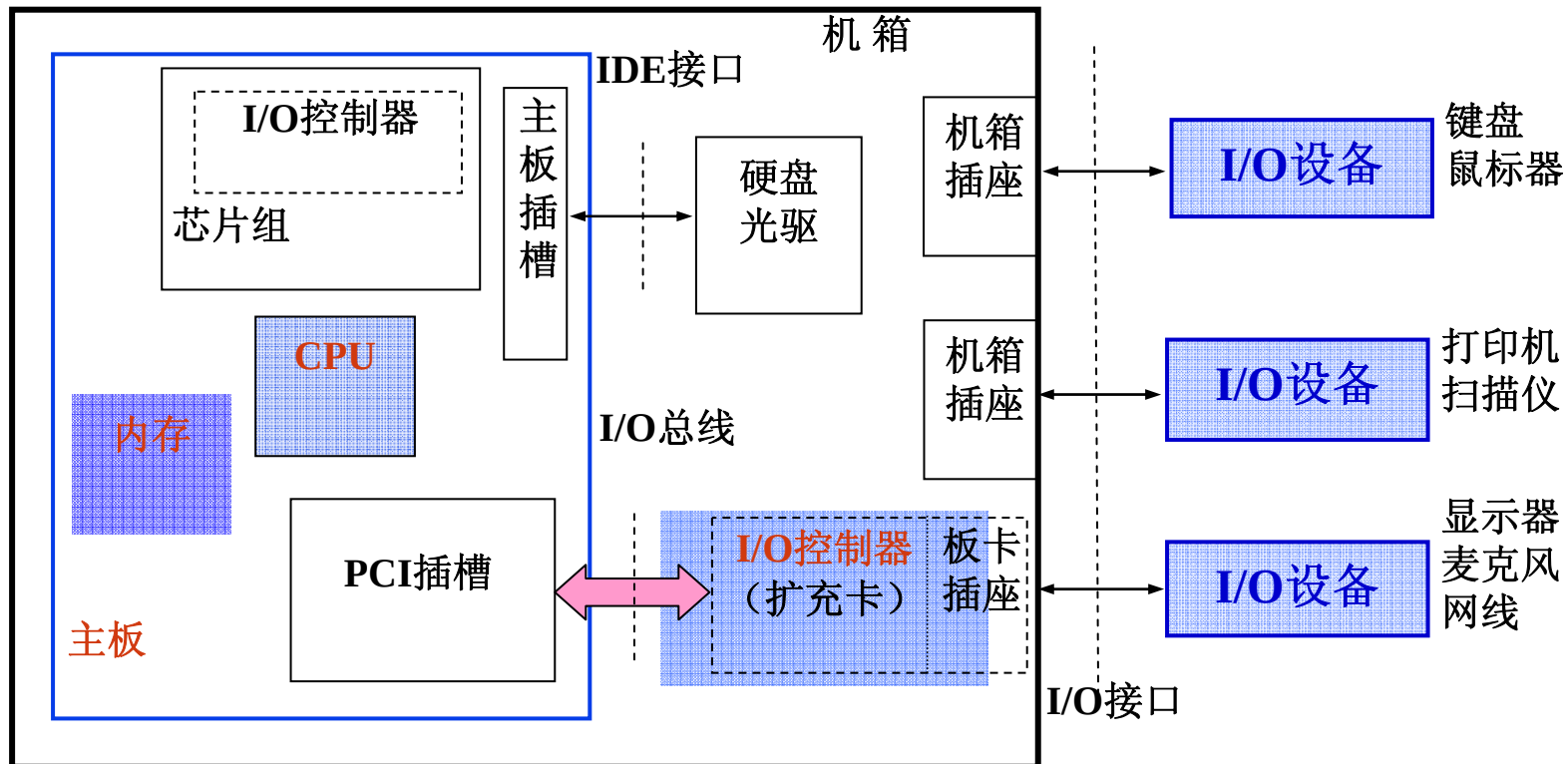
回顾： I/O设备接口



(安装在主板上的I/O设备接口)

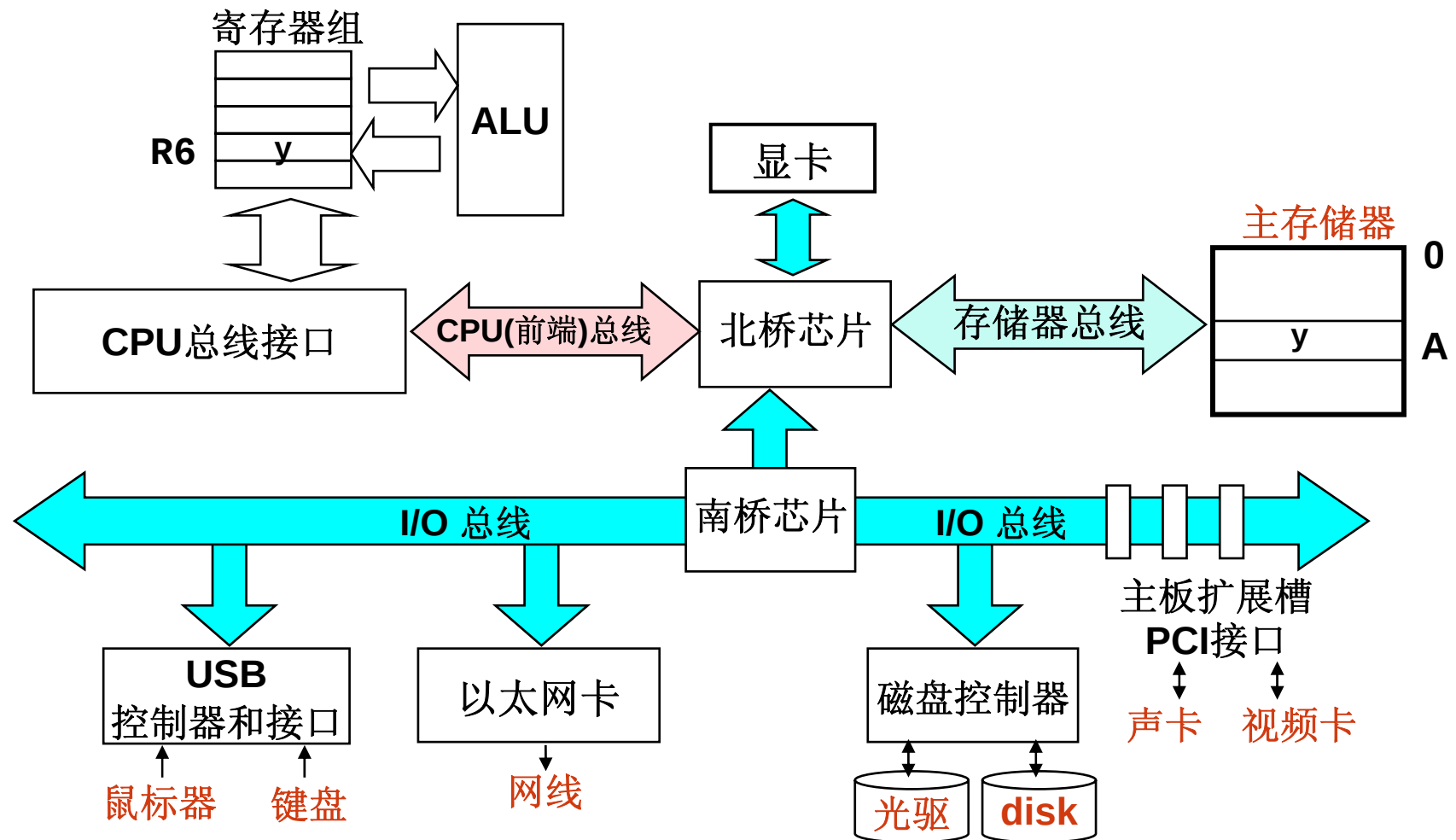


回顾： I/O总线,I/O控制器,I/O接口与I/O设备的关系



- I/O设备通常都是物理上相互独立的设备，它们一般通过I/O接口与I/O控制器连接
- I/O控制器通过扩展卡或者南桥芯片与I/O总线连接
- I/O总线经过北桥芯片与内存、CPU连接

回顾： I/O总线,I/O控制器,I/O接口与I/O设备的关系



本教材把I/O控制器和插座合起来称为I/O接口。

I/O接口（I/O控制器）的职能

- 数据缓冲

提供数据缓冲寄存器，以达到主机和外设工作速度的匹配。

- 错误或状态检测

提供状态寄存器，以保存各种错误或状态信息供**CPU**查用。

- 控制和定时

提供控制和定时逻辑，以接受从系统总线来的控制定时信号。

- 数据格式转换

提供数据格式转换部件使通过外部接口得到的数据转换为内部接口需要的格式，或在相反的方向进行数据格式转换。

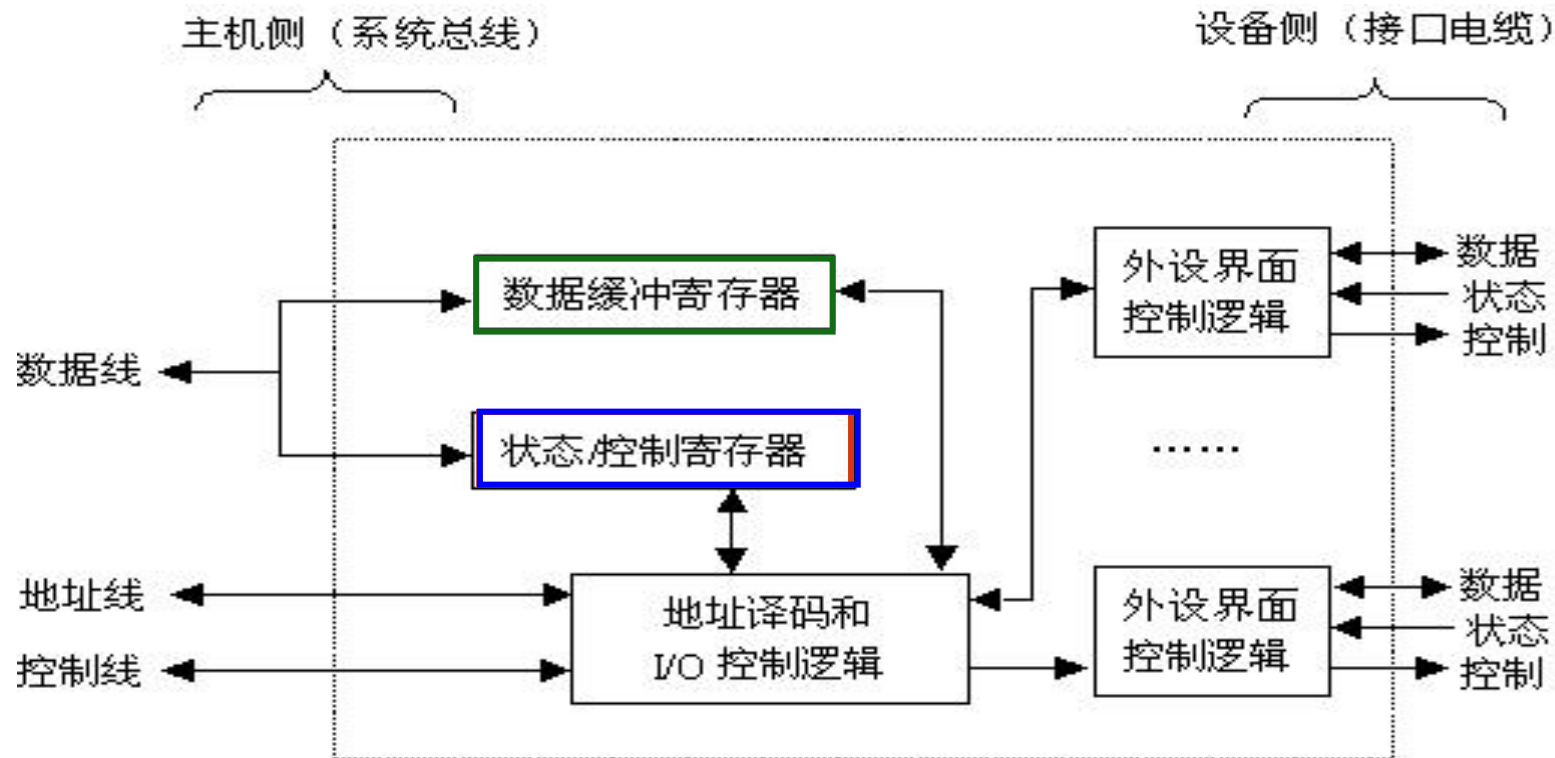
- 与主机和设备通信

上述功能通过**I/O**接口与主机之间、**I/O**接口与设备之间的通信来完成。

I/O接口（I/O控制器）的结构

- I/O控制器的一般结构

（各种不同的I/O模块在复杂性和控制外设的数量上相差很大。）



通过I/O控制寄存器发送命令字来向设备发送命令

通过从状态寄存器读取状态字来获取外设或I/O控制器的状态信息

通过向I/O控制器发送或读取数据来和外设进行数据交换

将I/O控制器中CPU能够访问的各类寄存器称为I/O端口

对外设的访问通过向I/O端口发命令、读状态、读/写数据来进行

I/O设备的寻址方式

- 对I/O端口读写，就是向I/O设备送出命令或从设备取得状态或读/写设备数据
- 一个I/O控制器可能会占有多个端口地址
- I/O端口必须编号后，CPU才能访问
- I/O设备的寻址方式就是I/O端口的编号方式

(1) 统一编址方式（内存映射方式）

与主存空间统一编址，将主存空间分出一部分地址给I/O端口进行编号。

（因该方法是将I/O端口映射到主存空间的某区，故也被称为“存储器映射方式”）

例如，**Motorola**公司生产的处理器就采用该方案

(2) 独立编址方式（特殊I/O指令方式）

不和主存单元一起编号，而是单独编号，使成为一个独立的I/O地址空间

（因需专门I/O指令，故也称为“特殊I/O指令方式”）

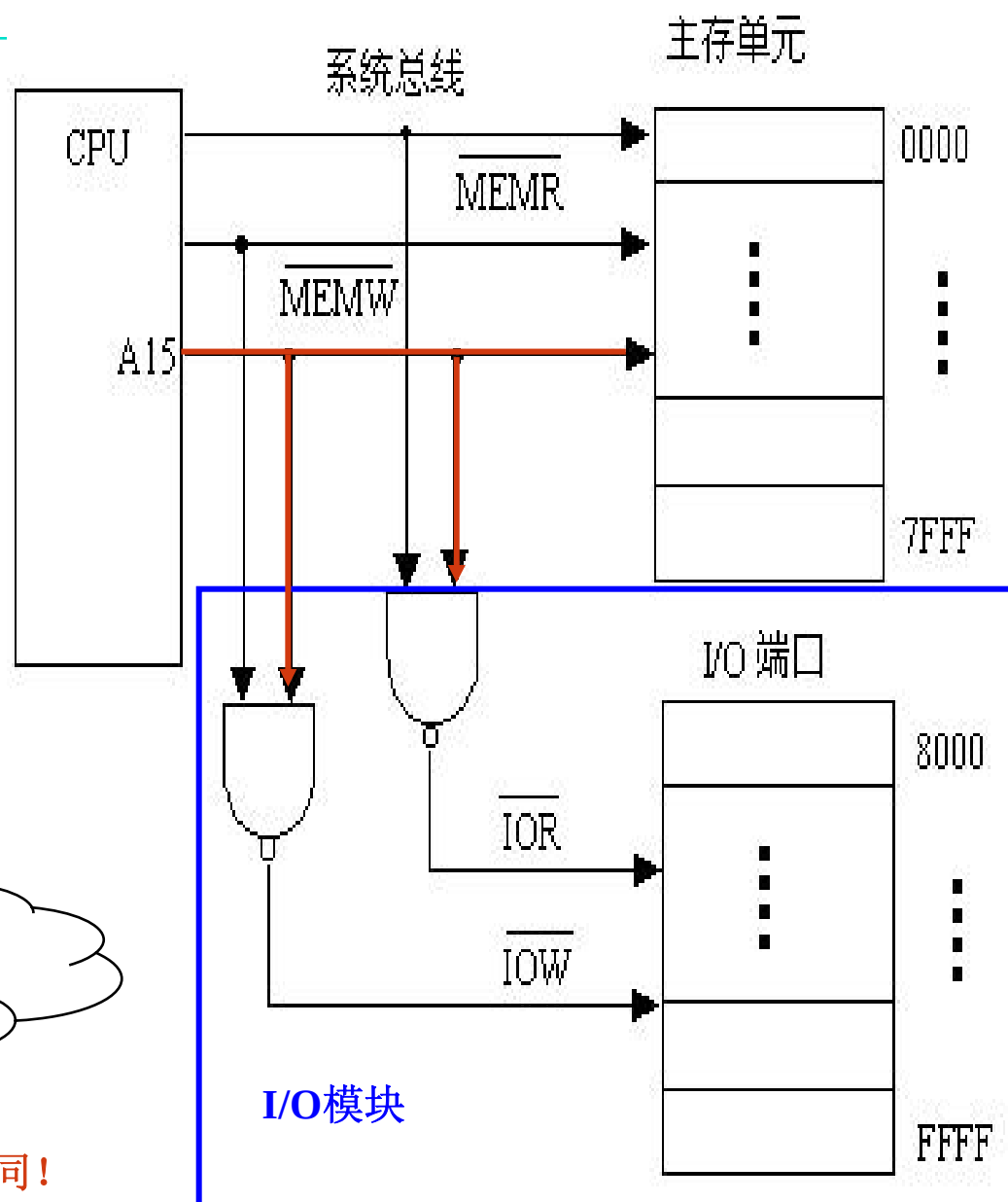
例如，**Intel**公司和**Zilog**公司的处理器就是独立编址方式

统一编址方式

- ° **CPU**不直接通过读写控制信号**IOR#**、**IOW#**对**I/O**端口读写，而是根据**I/O**端口在地址空间的位置，通过地址译码来实现。
- ° 地址线的高位参与片选控制逻辑。
- ° 无需设置专门**I/O**指令，只要用一般访存指令就可存取**I/O**端口。

MemR#或MemW#命令由访存指令发出，**IOR#**和**IOW#**命令怎样呢？

也是访存指令，只是访问的地址范围不同！



统一编址方式的优缺点

- 主要优点:

- 与访存指令一致的存/取指令
 - 减少对专门I/O指令的设计
 - 利用丰富的访存指令进行I/O操作（不仅可对端口进行数据传送，而且还可直接对端口进行移位、测试等）。
- 便于扩大系统吞吐率
 - 外设或I/O寄存器数目几乎不受限制，而只受总存储容量的限制。这在大型控制或数据通信系统等特殊场合很有用。
- 读写控制逻辑简单

- 主要缺点:

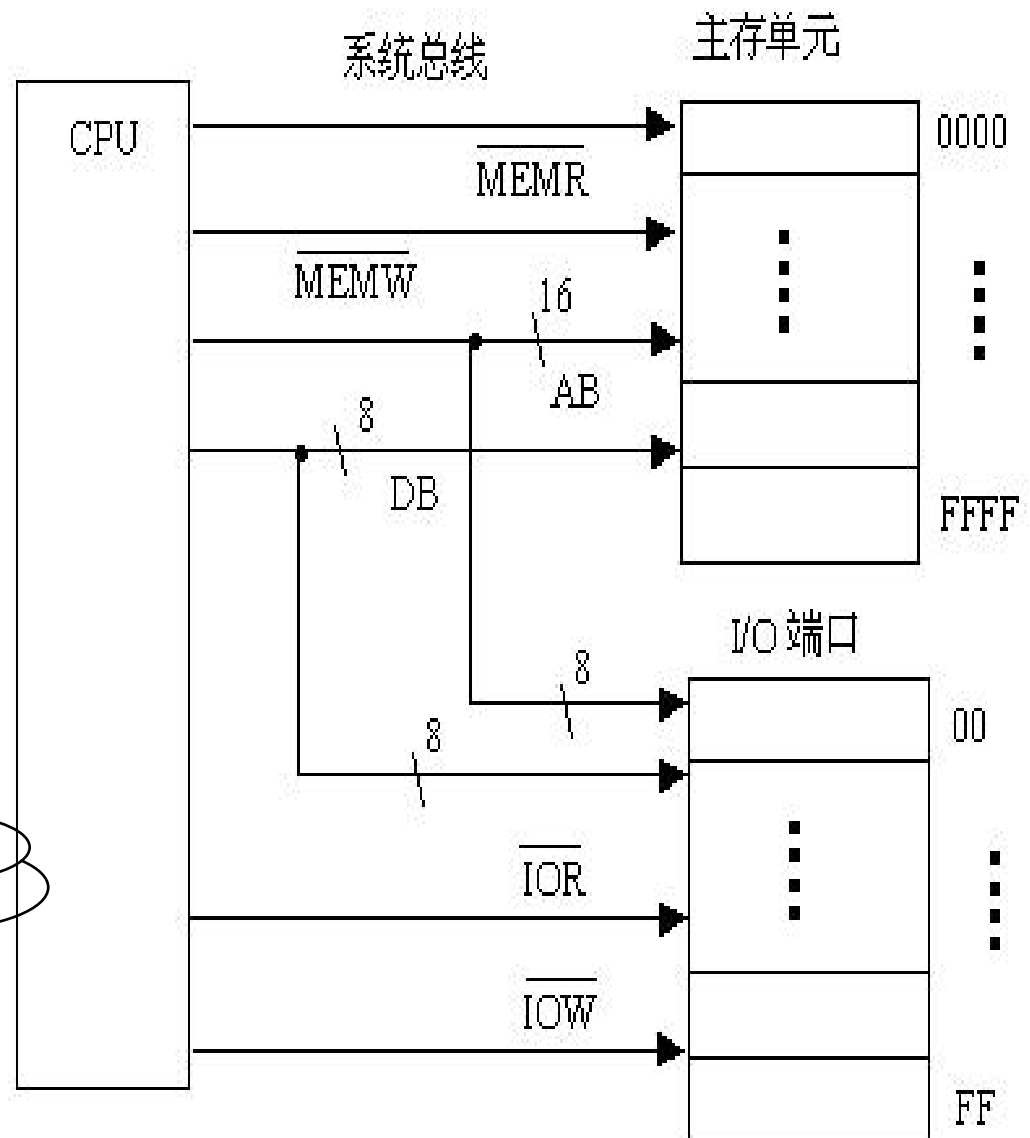
- 主存空间减少。因为被I/O占用了存储空间。
- 外设寻址时间长。为了识别I/O端口,全部地址线都需参与地址译码,使译码电路复杂并需花很长时间。

独立编址方式

- 通过不同的读写控制信号 **IOR#**、**IOW#**和 **MEMR#**、**MEMW#**来实现对**I/O** 端口和存储器的读写。
- 一般**I/O**端口比存储器单元少，所以选择**I/O**端口时，只需少量地址线。
- 指令系统必须设计专门的**I/O**指令。

MemR#或MemW#命令由访存指令发出，**IOR#**和**IOW#**命令怎么办呢？

是专门的**I/O**指令，指令中给的地址可能相同，但操作命令不同！



独立编址方式的优缺点

- 主要优点:

- **I/O**端口地址不占用存储器地址空间，故主存空间不受**I/O**地址的影响。
- **I/O**地址线较少，所以**I/O**端口译码简单，寻址速度快。
- 使用专用**I/O**指令，程序清晰，便于理解和检查。

- 主要缺点:

- 专用 **I/O**指令类型少，只提供简单的传输操作，故程序设计灵活性差。
- 要求处理器提供两组读写命令（**MEMR / MEMW**、**IOR / IOW**），增加了控制逻辑的复杂性和处理器引脚数。

（自学）奔腾机的I/O端口编址方式

- 采用独立编址方式，I/O地址空间由 2^{16} (64K)个8位端口组成
- 虽然具有64K字节的寻址空间，但一般只使用其中1K字节的I/O空间，故只用低10位地址线寻址
- 两个连续的8位端口可作为一个16位端口；四个连续的8位端口可作为一个32位端口。所以一次可传送32位、16位或8位数据
- 采用专门的I/O指令：IN和OUT（处理器执行到这些指令时产生相应的I/O读写命令信号）
- 部分外设的I/O地址分配表

（自学）奔腾机I/O端口地址分配表

部分外设的 I/O 地址分配表

输入/出设备	I/O 地址	占用地址数
DMA 控制器 1	000-01FH	32
中断控制器 1	020-03FH	32
定时器/计数器	040-05FH	32
键盘控制器	060-06FH	32
实时时钟, NMI屏蔽寄存器	070-07FH	16
DMA 页面寄存器	080-09FH	32
中断控制器 2	0A0-0BFH	32
DMA 控制器 2	0C0-0DFH	32
硬盘控制器 2	170-177H	8
硬盘控制器 1	1F0-1F8H	9
游戏 I/O 口	200-207H	8
并行打印机口 2	278-27FH	8
串行口 4	2E8-2EFH	8
串行口 2	2F8-2FFH	8
软盘控制器 2	370-377H	8
并行打印机口 1	378-37FH	8
单色显示器/打印适配器	3B0-3BFH	16
彩色/图形监视器适配器	3D0-3DFH	16
串行口 3	3E8-3EFH	8
软盘控制器 1	3F0-3F7H	8
串行口 1	3F8-3FFH	8

并行传输和串行传输（数据通信课学过？）

并行传输方式

- 多位数据在多条数据线上并行传输
- 最大传输率为：时钟频率 $\times$ 数据位数

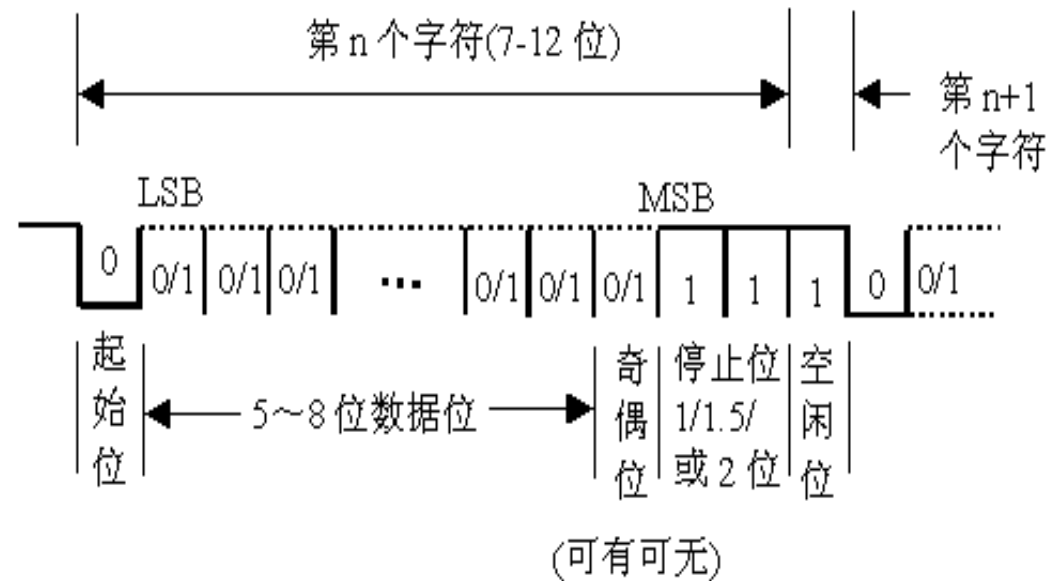
串行传输方式

- 波特率：每秒钟通过信道传输的符号数
- 比特率：每秒钟传输的位数
- 两相调制时，波特率=比特率
- 异步串行

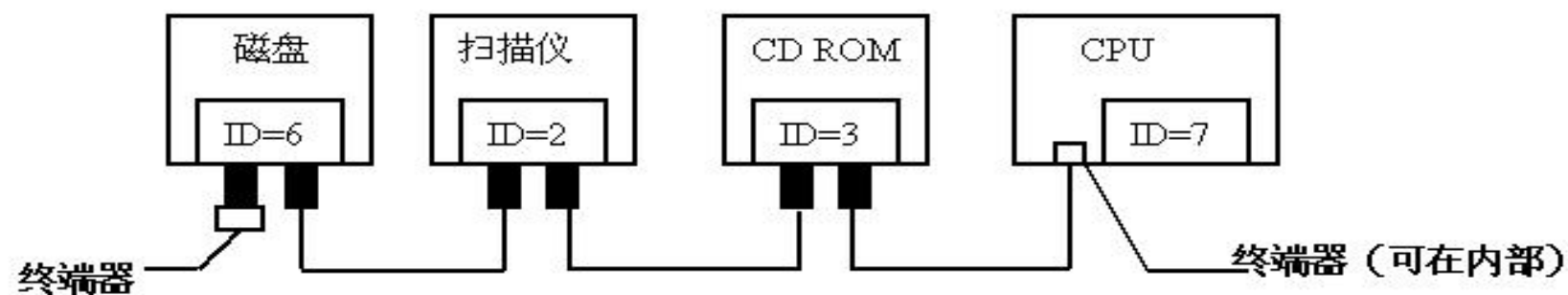
- 每个字符的开始是随机的，字符内的位之间同步
- 有效数据位为5位时，停止位取1位或1.5位，其他情况取1位或2位
- 一个字符可能由7~12位信息组成，称其为一个数据帧
- 缺点：每个字符都有额外信息，实际数据传输率低

同步串行

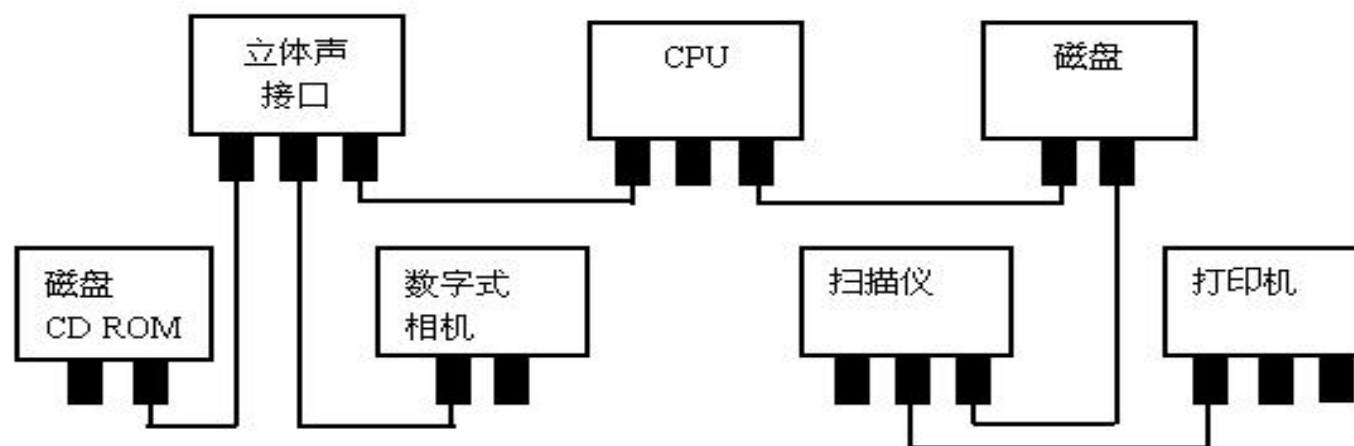
- 字符之间、字符内的位之间都同步



总线式I/O接口



(a) SCSI 配置举例



(b) P1394 配置举例

第一讲小结

- I/O系统概述
 - I/O系统的性能主要有吞吐率和响应时间，两者是对立统一的关系
 - I/O系统的具体任务是：构建传输通路、对设备寻址、向设备发命令、取状态，并提供相应的传输机制来读/写设备数据等（后面一讲的内容）
- I/O设备概述
 - I/O设备通过I/O接口和主机相连
 - 外设分类
 - I/O设备和存储设备
 - 机读设备和人读设备
- 磁盘存储器
 - 读写原理：两种不同的磁化状态
 - 性能指标：寻道时间、旋转等待时间、传输时间
 - 冗余磁盘阵列：多个物理盘组成一个逻辑盘，以提高磁盘存取速度、容量和可靠性
- I/O接口（I/O控制器）中一般有数据缓冲器、状态/控制寄存器、串-并转换、设备控制逻辑、地址译码逻辑等。用于在主机和设备之间进行命令、数据、状态信息的传递和转换。
- I/O端口是指I/O控制器中CPU可访问的寄存器，对I/O设备的寻址就是对I/O端口的访问
- I/O端口的编址方式有两种：内存映射方式（统一编址）和特殊I/O指令方式（独立编址）

本教材把I/O控制器和I/O接口综合称为I/O接口，不严格区分控制逻辑部分（I/O控制器）和插座（I/O接口）部分

第二讲 I/O传输方式

主 要 内 容

- OS在I/O系统中的职责
- I/O传输方式
 - 轮询方式（程序直接控制 / 程序查询方式）
 - 程序中断方式（中断驱动方式）
 - 中断响应的条件和中断响应过程
 - 中断处理过程
 - 中断控制器
 - 多重中断和中断屏蔽
 - 直接存储器访问方式（DMA方式）
 - DMA方式的要点
 - DMA控制器的结构
 - DMA的三种控制方式
 - DMA传输过程
 - 通道方式和I/O处理器方式简介

操作系统在I/O中扮演的角色

- OS的职责由I/O系统的三个特性决定：
 - I/O系统被处理器执行的多个程序共享，由OS统一调度管理
 - I/O系统通常使用外部中断请求来要求处理器执行专门的输入/出程序。中断导致向内核态转移，故必须由OS来处理
 - I/O设备控制细节复杂，不能由上层用户程序来实现，需OS提供专门驱动程序
- OS在I/O中的职能
 - 保证用户程序只能访问I/O设备自己有权访问的那部分。例如，用户程序不能读写没有授权的文件。
 - 为用户程序提供设备的驱动程序以屏蔽设备控制的细节
 - 处理外部I/O中断，提供中断服务程序
 - 对共享的I/O资源提供合理的调度管理，使系统的吞吐率达到最佳
- 主机必须和I/O设备进行以下三类通信
 - OS必须能给I/O设备提供命令，如：磁盘寻道
 - 需要知道何时I/O设备完成操作？何时遇到什么异常问题？
 - 数据必须能在主机（主存或CPU）和I/O设备之间进行传输

这些问题就是后面这部分要讲的内容

I/O设备与主机进行数据交换的三种基本方式

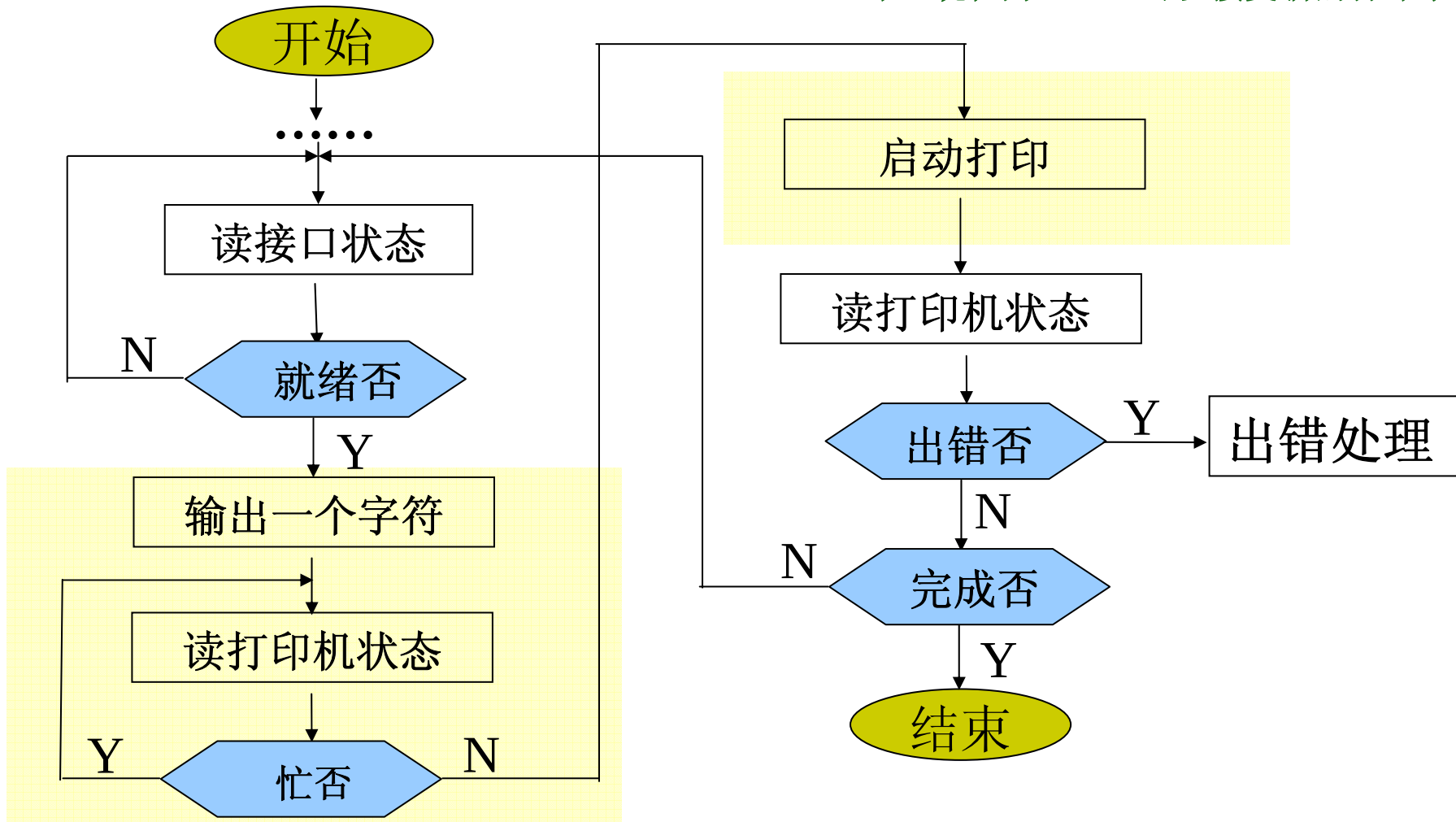
- **Polling (轮询, 查询):** 最简单的I/O方式
 - I/O设备（包括I/O接口）将自己的状态放到一个状态寄存器中
 - OS阶段性地查询状态寄存器中的特定状态，来决定下一步的动作
 - OS主动查询，也称为程序查询方式或程序直接控制方式
- **I/O Interrupt (中断I/O):** 几乎所有系统都支持的中断I/O方式
 - 若一个I/O设备需要CPU干预，它就通过中断请求通知CPU
 - CPU中止当前程序的执行，调出OS（中断处理程序）来执行
 - 处理结束后，再返回到被中止的程序继续执行
 - OS被动调出，也称为中断驱动I/O方式
- **Direct Memory Access (DMA方式):** 磁盘等高速外设特有的I/O方式
 - 磁盘等高速外设成批地直接和主存进行数据交换
 - 需要专门的DMA控制器控制总线，完成数据传送
 - 当外设准备好数据后，向DMA控制器发DMA请求信号，DMA控制器再向CPU发总线请求，CPU让出总线后，由DMA控制器控制总线进行传输，无需CPU干涉

程序直接控制（程序查询）方式

- 举例：用程序直接控制方式控制打印输出

问题：这里“就绪”的含义是什么？

打印控制器的数据缓冲中内容已被取走打印，现在为“空”，可以接受新的打印字符

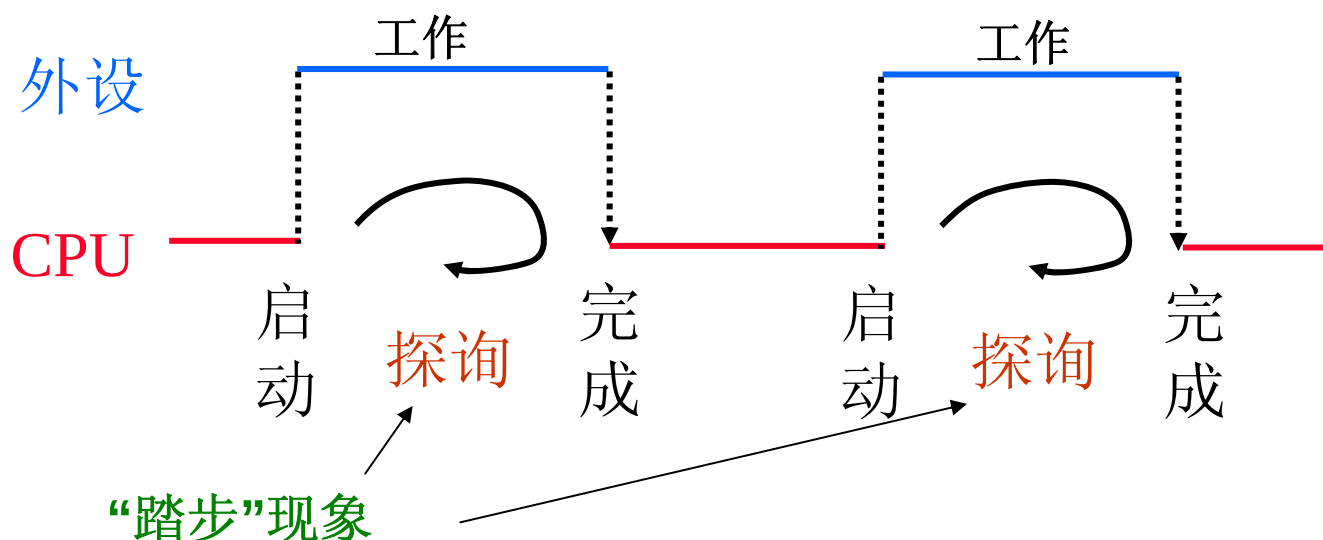


(自学) 打印输出标准子程序

功能：打印AL寄存器中的字符。

```
PRINT      PROC  NEAR
            PUSH  AX          ; 保留用到的寄存器
            PUSH  DX          ; 保留用到的寄存器
            MOV   DX, 378H     ; 输入数据锁存器口地址
            OUT   DX, AL       ; 输出要打印的字符到数据锁存器
            MOV   DX, 379H     ; 输入状态寄存器口地址
WAIT:       IN    AL, DX       ; 读打印机状态位
            TEST  AL, 80H      ; 检查忙碌位
            JE    WAIT        ; 等待直到打印机不忙
            MOV   DX, 37AH     ; 输入命令寄存器口地址
            MOV   AL, 0DH      ; 置选通位=1
            OUT   DX, AL       ; 使控制卡的命令锁存器中选通位置1
            MOV   AL, 0CH      ; 置选通位=0
            OUT   DX, AL       ; 使控制卡的命令锁存器中选通位置0
            POP   DX
            POP   AX          ; 恢复寄存器
            RET
PRINT      ENDP
```

程序控制I/O（程序查询I/O）



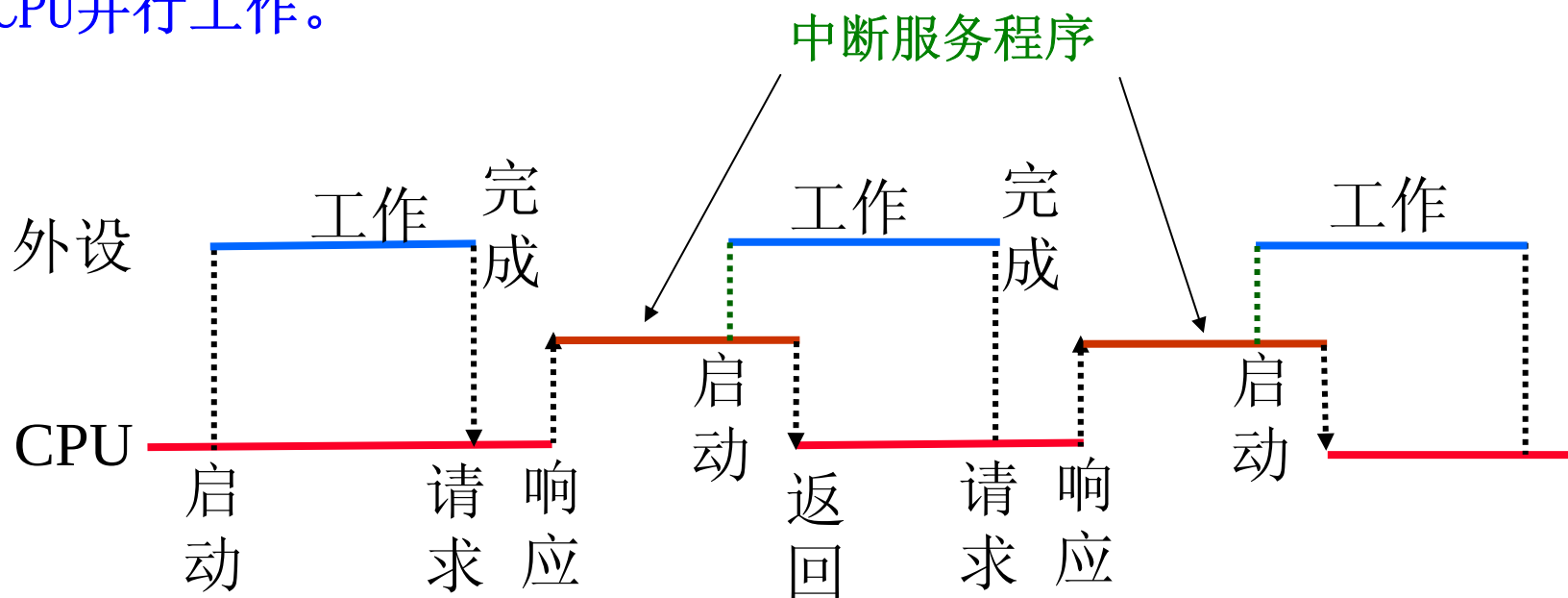
“探测”期间，可以一直不断查询（原地“踏步”），也可以定时查询（但要保证数据不丢失！）。

- 特点：
 - 简单、易控制、外围接口控制逻辑少；
 - CPU**与外设串行工作，效率低、速度慢，适合于慢速设备
 - 查询开销极大 (**CPU**完全在等待“外设完成”)
- 工作方式：完全串行工作方式或部分串行

中断I/O方式

- 基本思想：

当外设准备好时，便向**CPU**发中断请求，**CPU**响应后，中止现行程序的执行，转入一个“中断服务程序”进行输入/出操作，实现主机和外设接口之间的数据传送，并启动外设工作。“中断服务程序”执行完后，返回原被中止的程序断点处继续执行。此时，外设和CPU并行工作。



复习：处理器中的异常处理机制

异常发生时，处理器必须做以下基本处理：

① 保护断点和程序状态：

将返回原程序执行的断点和程序状态保存到堆栈或特殊寄存器中

PC=>堆栈 或 EPC

PSWR=>堆栈 或 EPSWR

(注： **PSW (Program Status Word)**：程序状态字

PSWR (PSW寄存器)：用于存放程序状态的寄存器。如， **X86的FLAGS**)

② 识别异常事件

有两种不同的方式：软件识别和硬件识别（向量中断方式）

(1) 软件识别 (**MIPS**采用)：设置一个异常状态寄存器 (**MIPS**中为**Cause**寄存器)，用于记录异常原因。操作系统使用一个统一的异常处理程序 (**MIPS**的入口为 **0x8000 0180**)，该程序按优先级顺序查询异常状态寄存器，识别出异常事件。

(2) 硬件识别 (向量中断) (**80x86**采用)：用专门的硬件查询电路按优先级顺序识别异常，得到一个“中断类型号”，根据此号，到中断向量表中读取对应的中断服务程序的入口地址。

③ 切换到具体的异常处理程序执行

问题：还有一个首先要做的基本操作，是什么？

关中断！即：将中断允许标志清0。

复习：8086/8088的中断向量表

中断向量表也称中断入口地址表（或异常表），位于**0000H~03FFH**。共**256**组，每组占四个字节 **CS:IP** 。向量地址=中断类型号**x4**

例1：除法错的中断类型号为**0**，故其向量地址为： **$0 \times 4 = 0$**

例2：**NMI**的中断类型号为**2**，故其向量地址为： **$2 \times 4 = 8$**

CS:IP	除法错	00~03
CS:IP	单步	04~07
CS:IP	NMI	08~0B
⋮	⋮	
CS:IP		
CS:IP		3FC~3FF

中断向量表（异常表）中每一项是对应异常处理程序的入口地址。被称为中断向量(**Interrupt Vector**)

中断向量表的起始地址存放在一个异常表基址寄存器中。

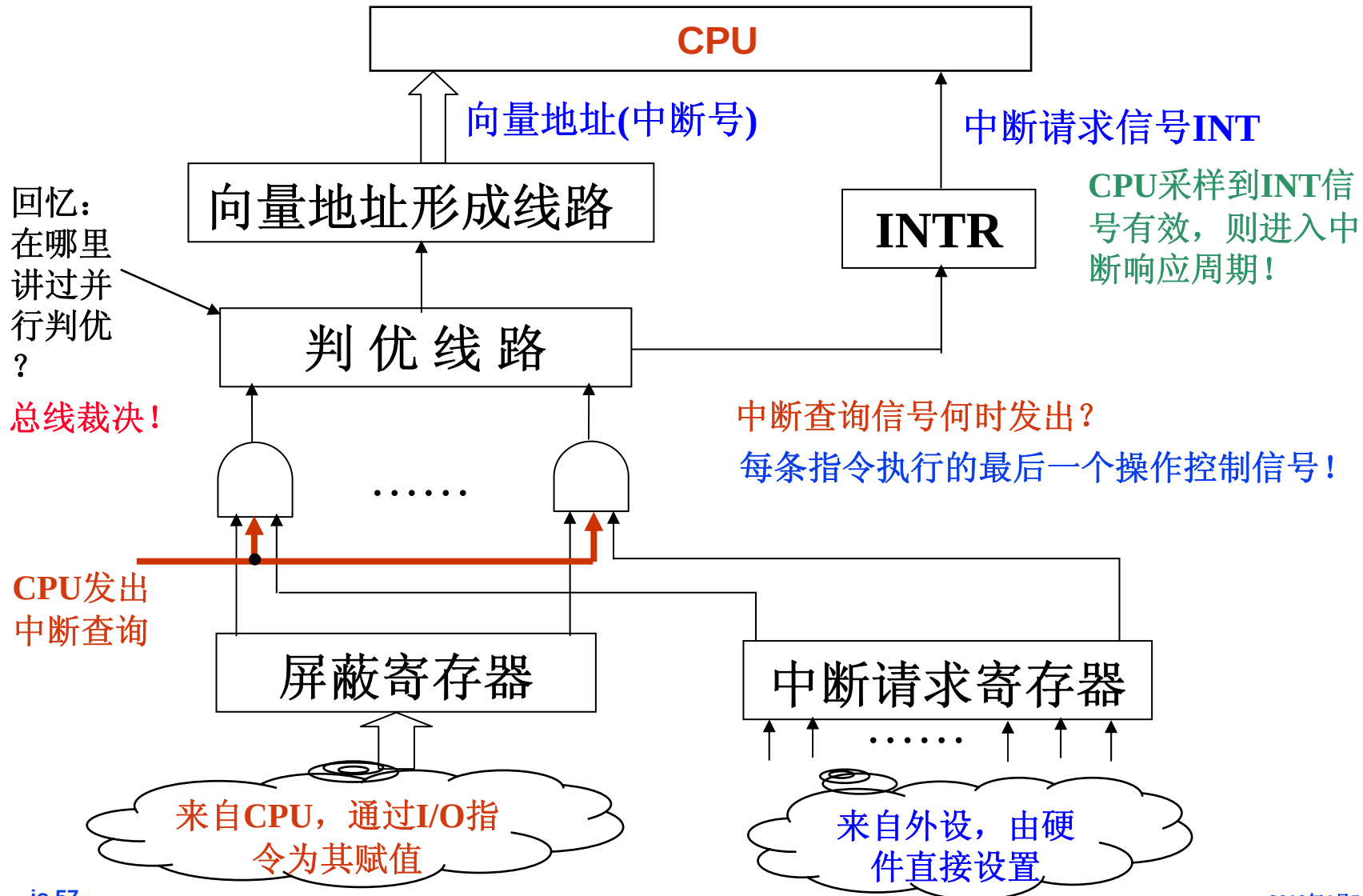
中断控制器的基本结构（如：8259A）

中断号（向量地址）送到什么线上？

数据线 / 地址线？ 数据线上！为什么？

何时采样中断请求信号？

中断查询信号发出后的固定时间内！



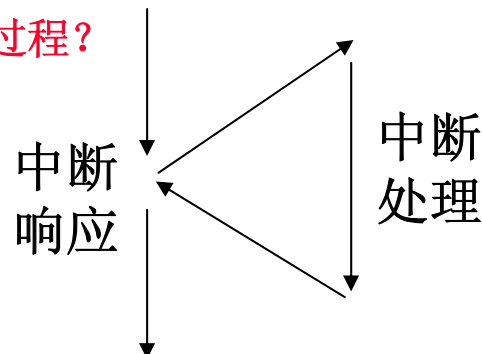
中断驱动I/O方式

- 中断过程 回忆：在哪里讲过具体的“异常”响应过程？

- 中断响应（硬件实现）
- 中断处理（软件实现）

- 中断响应

- 中断响应是指主机发现外部中断请求，中止现行政程序的执行，到调出中断服务程序这一过程。



(1) 中断响应的条件

- ① CPU处于开中断状态
- ② 在一条指令执行完
- ③ 至少要有有一个未被屏蔽的中断请求

问题：中断响应的时点与异常处理的时点是否相同？为什么？

中断一定是在一条指令执行结束后开始查询有无中断请求，有的话立即响应，所以一定是在指令执行完时响应中断，而“异常”发生在指令执行过程中，所以不能等到指令执行完才进行异常处理。

中断I/O方式

(2)中断响应过程

执行一条隐指令，可能需完成一次总线操作，从总线上取中断类型号
具体来说，处理器做三件事：

① 关中断

$0 \Rightarrow$ 中断允许触发器 C_{IEN}

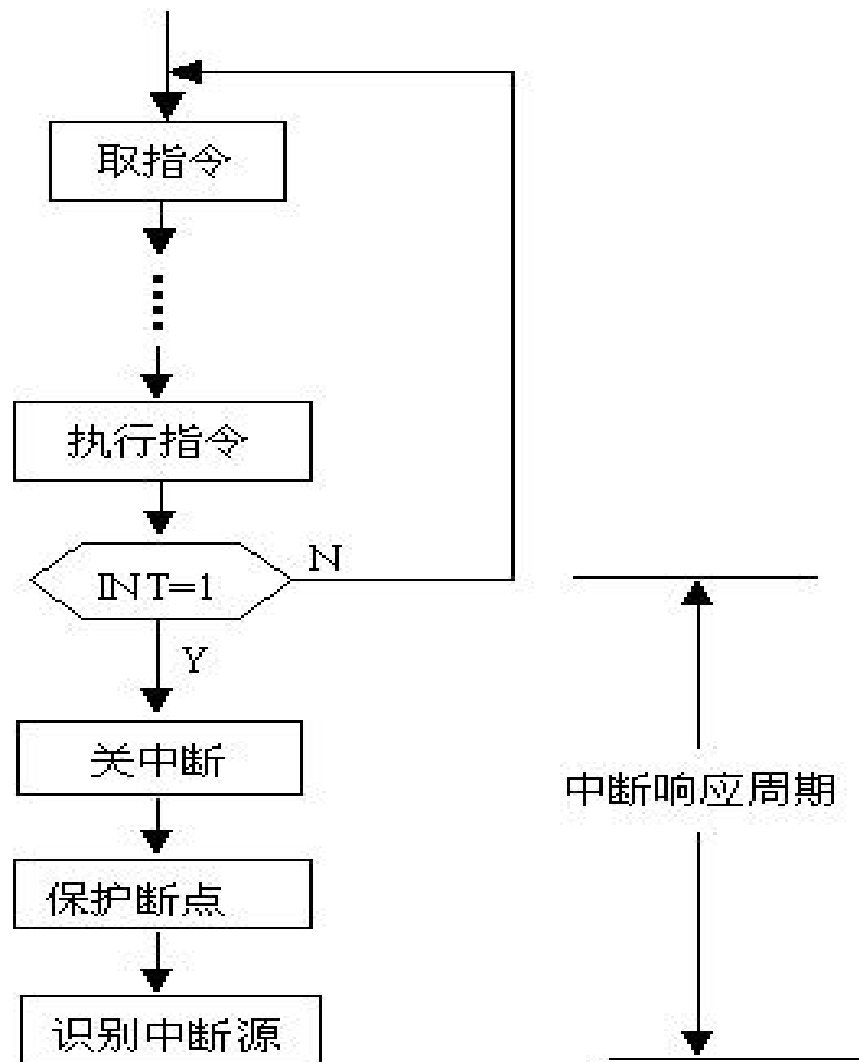
② 保护断点和程序状态

$PC \Rightarrow$ 堆栈（或特殊寄存器 EPC ）

$PSW \Rightarrow$ 堆栈

③ 识别中断源

取得中断服务程序首地址和初始
 PSW 分别送 PC 和 $PSWR$



复习：中断源的识别方法

- 软件方法（轮询）

中断查询程序根据中断请求状态，按优先级顺序来识别

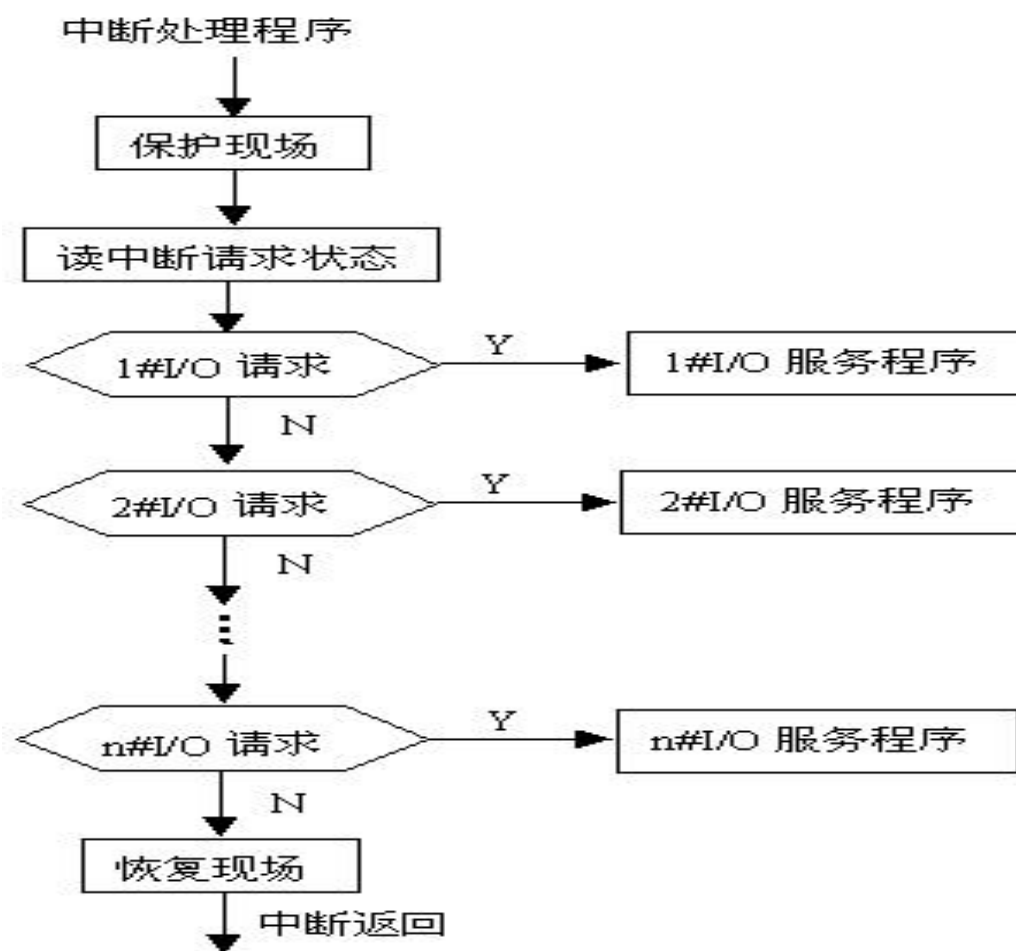
如：**MIPS**的异常/中断查询程序入口为：

0x8000 0180

- 硬件方法（向量中断）

将所有中断请求状态送到一个排队电路中，根据中断优先级识别出最高优先级的中断请求

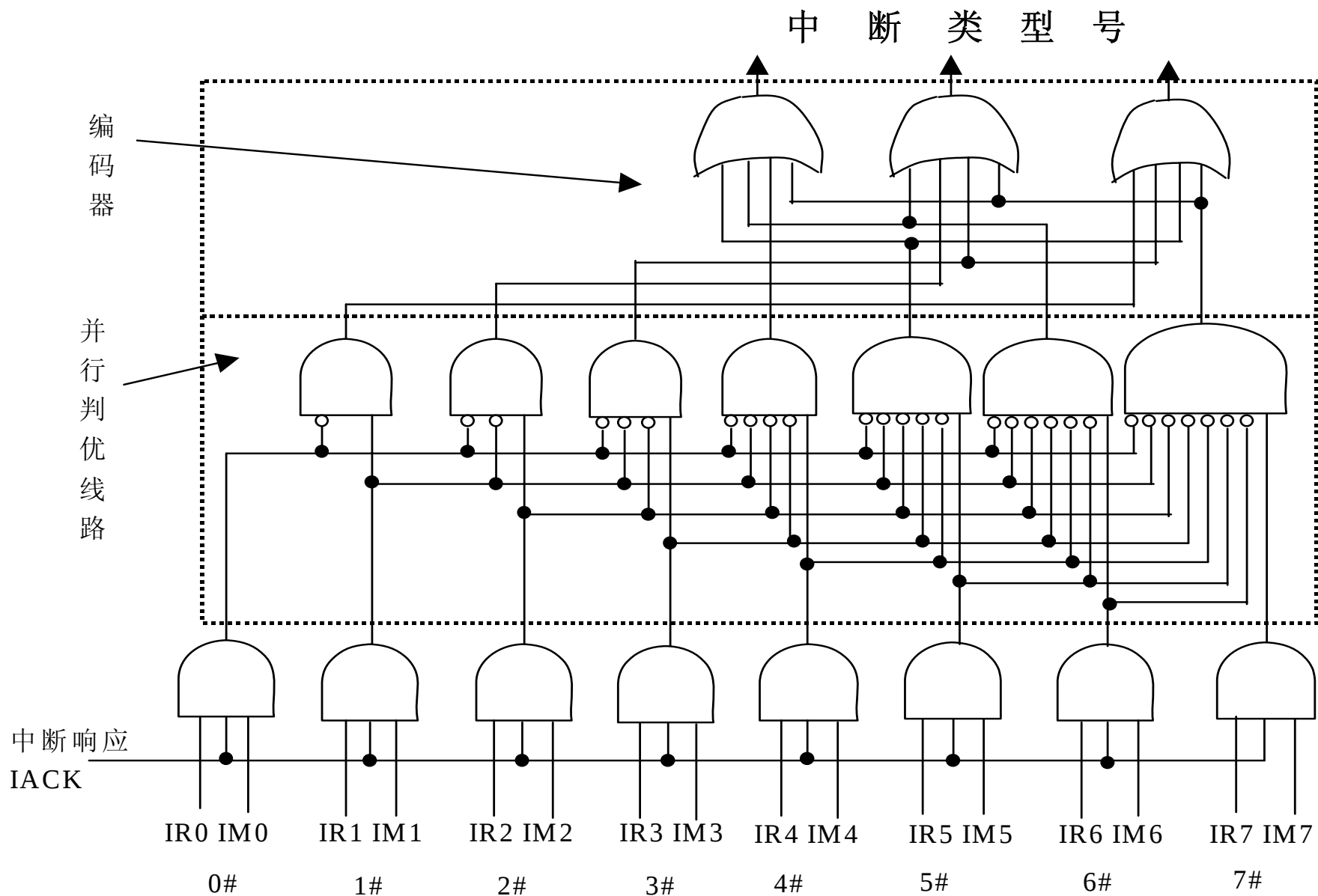
- 链式查询（菊花链）
- 独立请求（并行判优）



中断控制器中的“中断优先权编码器”专门用来进行中断源识别

注：内部异常和外部中断都有优先级，通常所有内部异常的优先级都比外部中断高。为什么？

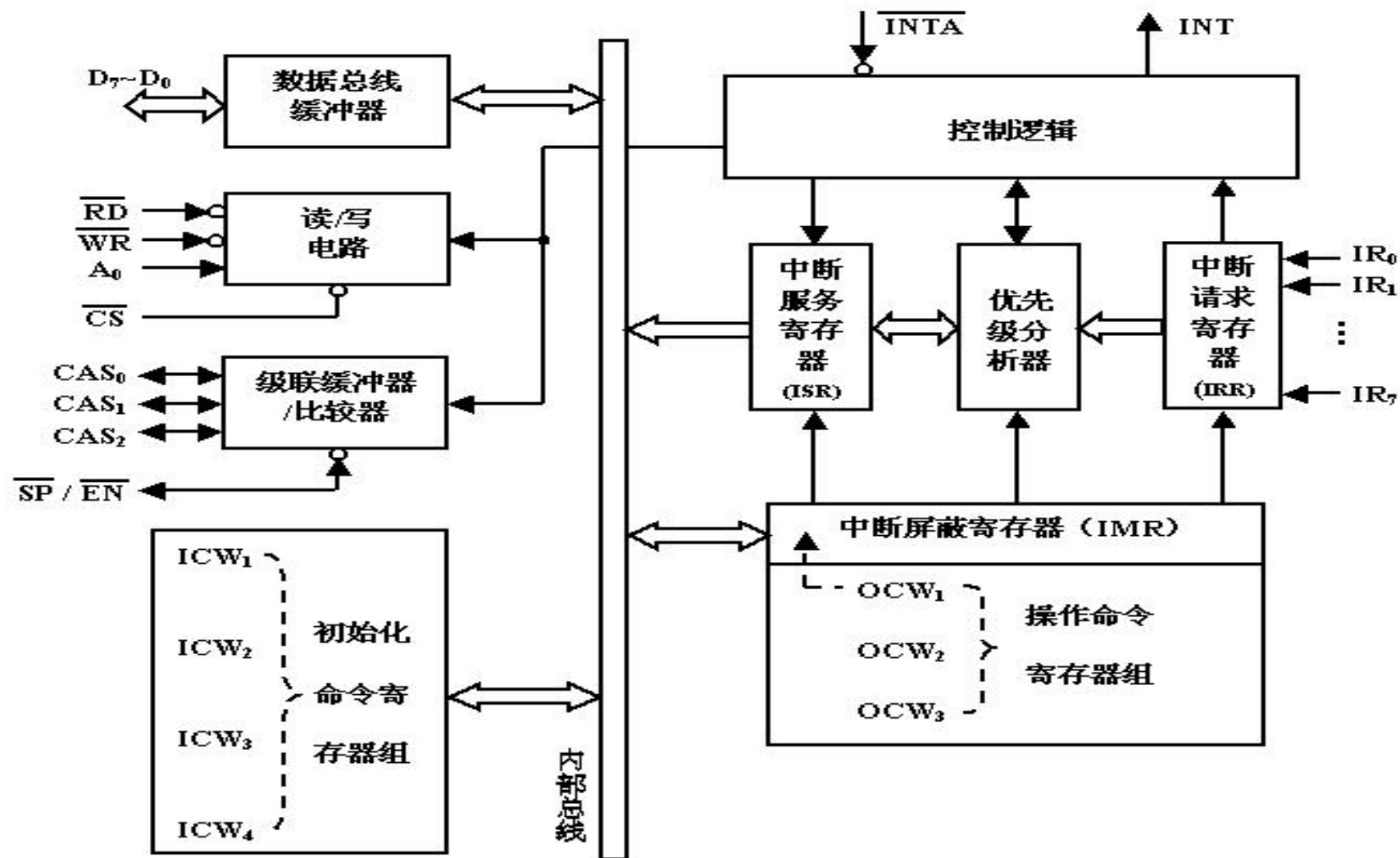
中断优先权编码器



（自学）中断控制器举例-8259A

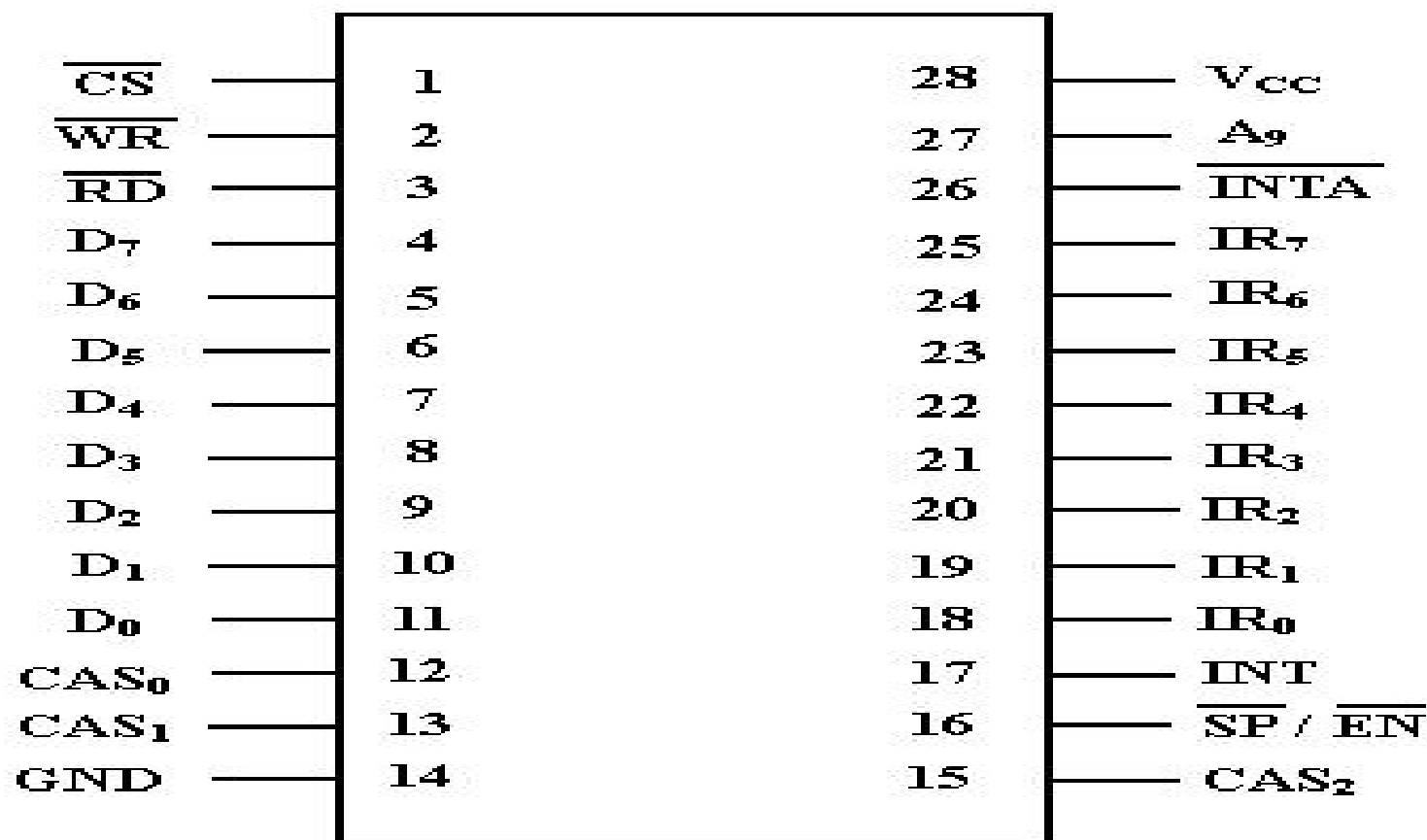
- **8259A**是可编程的中断控制器
- **8259A**的功能
 - 包含中断请求锁存、中断屏蔽、中断优先级排队、中断向量生成（中断优先权编码器）等电路
 - 既可支持程序查询式中断，又可支持向量式中断
 - 支持**8**级优先权，通过多片级联，最多可构成**64**级中断
 - 各种中断功能可通过编程设定或更改

(自学) 8259A的内部结构



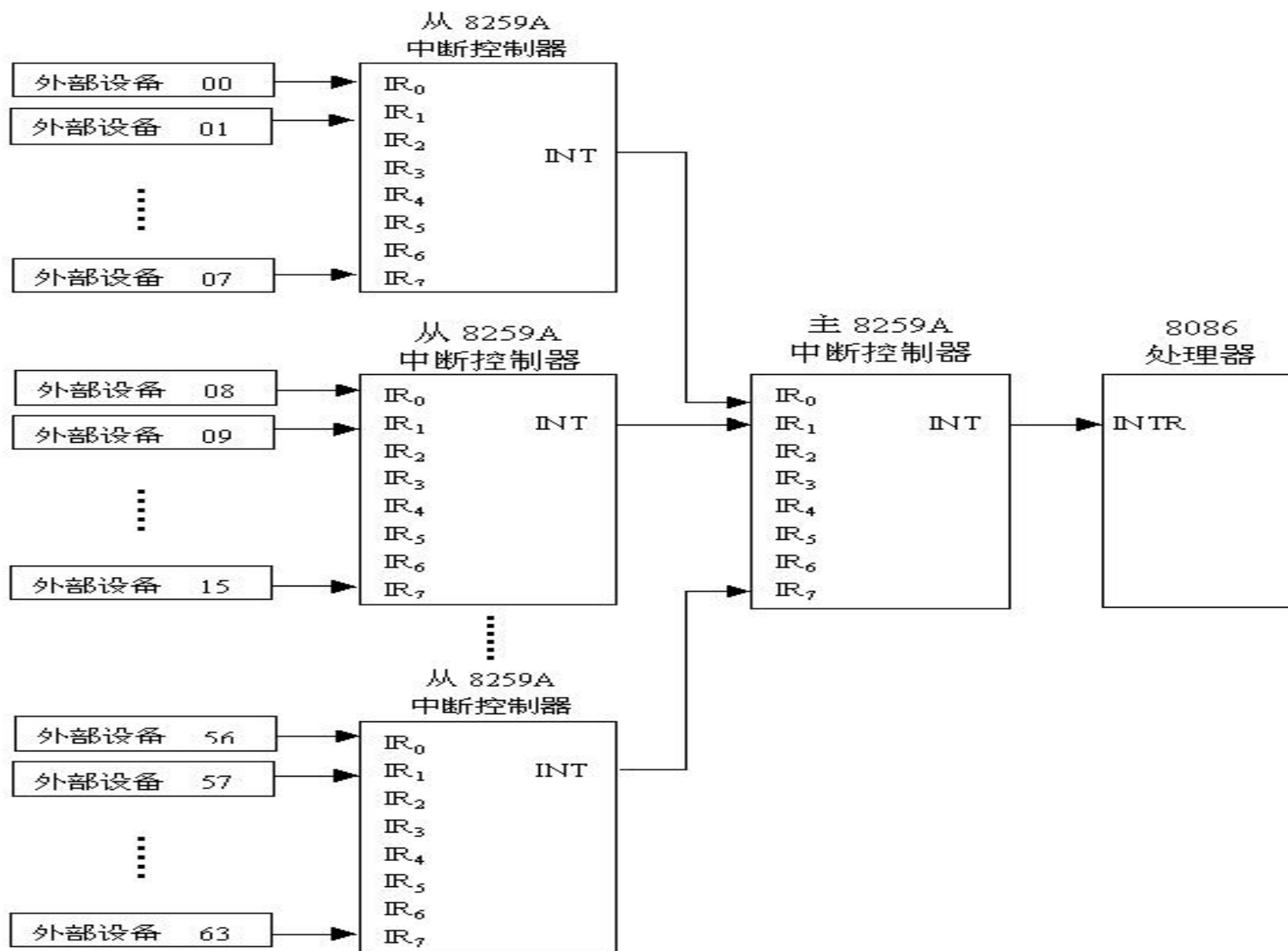
8259A 内部结构框图

(自学) 8259A的引脚功能



8259A 引脚图

(自学) 8259A芯片的级联



中断处理过程

中断过程：中断响应+中断处理

中断响应的结果是调出相应的中断服务程序。

◦ 中断处理：

- 是指执行相应中断服务程序的过程。
- 不同的中断源其对应的中断服务程序不同。
- 典型的中断处理（中断服务程序）分为三个阶段：

- 先行段（准备阶段）

 - 保护现场及旧屏蔽字

 - 查明原因（软件识别中断时）

 - 设置新屏蔽字

 - 开中断

- 本体段（具体的中断处理阶段）

- 结束段（恢复阶段）

 - 关中断

 - 恢复现场及旧屏蔽字

 - 清“中断请求”

 - 开中断

 - 中断返回

多重中断的概念

- 多重中断和中断处理优先权的动态分配

- 多重中断的概念:

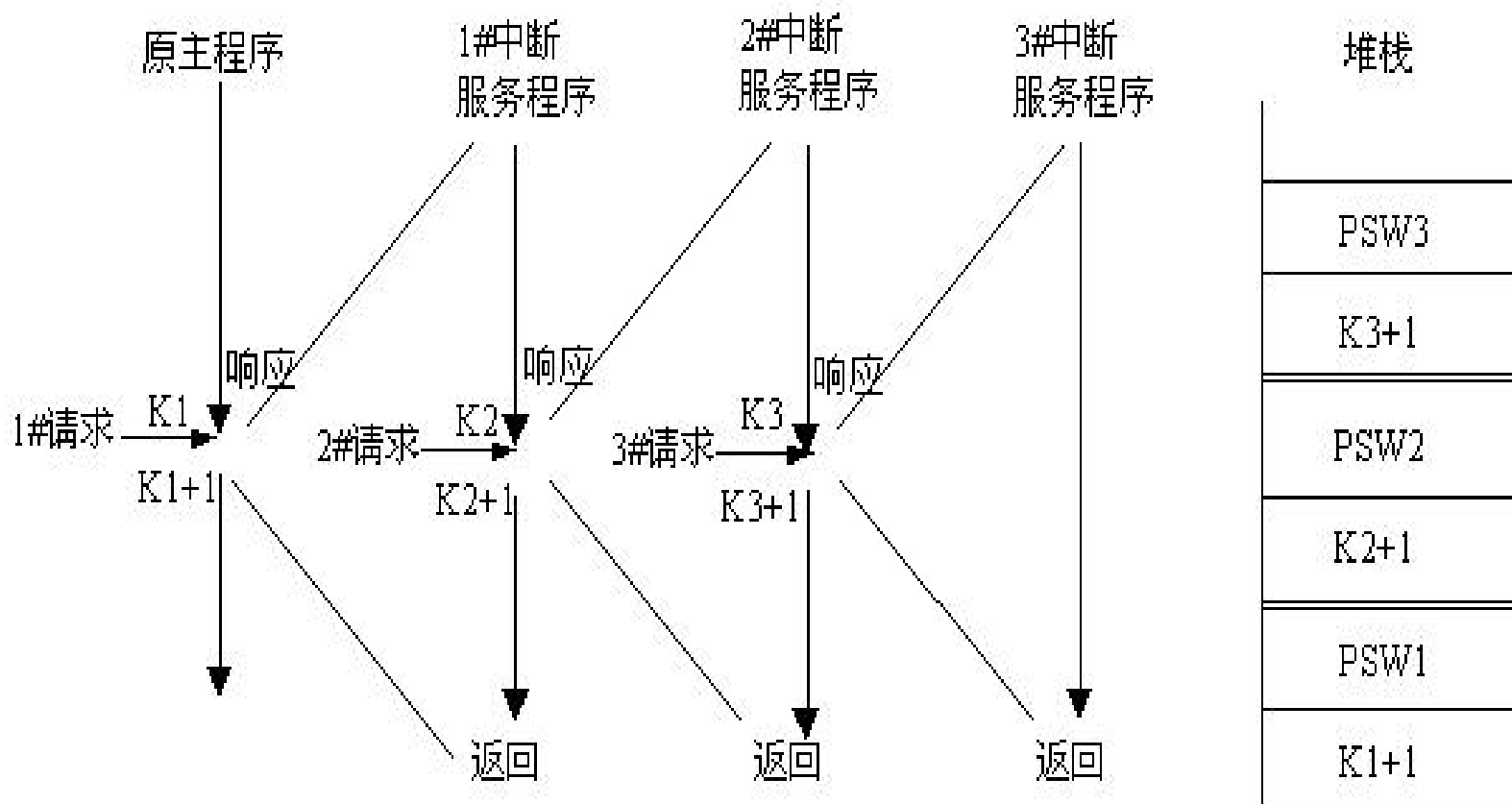
在一个中断处理（即执行中断服务程序）过程中，若又有新的中断请求发生，而新中断优先级高于正在执行的中断，则应立即中止正在执行的中断服务程序，转取处理新的中断。这种情况为多重中断，也称中断嵌套。

- 中断优先级的概念:

中断响应优先级——由查询程序或硬联排队线路决定的优先权，反映多个中断同时请求时选择哪个响应。

中断处理优先级——由各自的中断屏蔽字来动态设定，反映本中断与其它中断间的关系。

多重中断嵌套



中断优先级的顺序是：
 $3\# > 2\# > 1\#$

Stack是内存中采用“FILO”
的一块特殊的存储区

中断优先权的动态分配

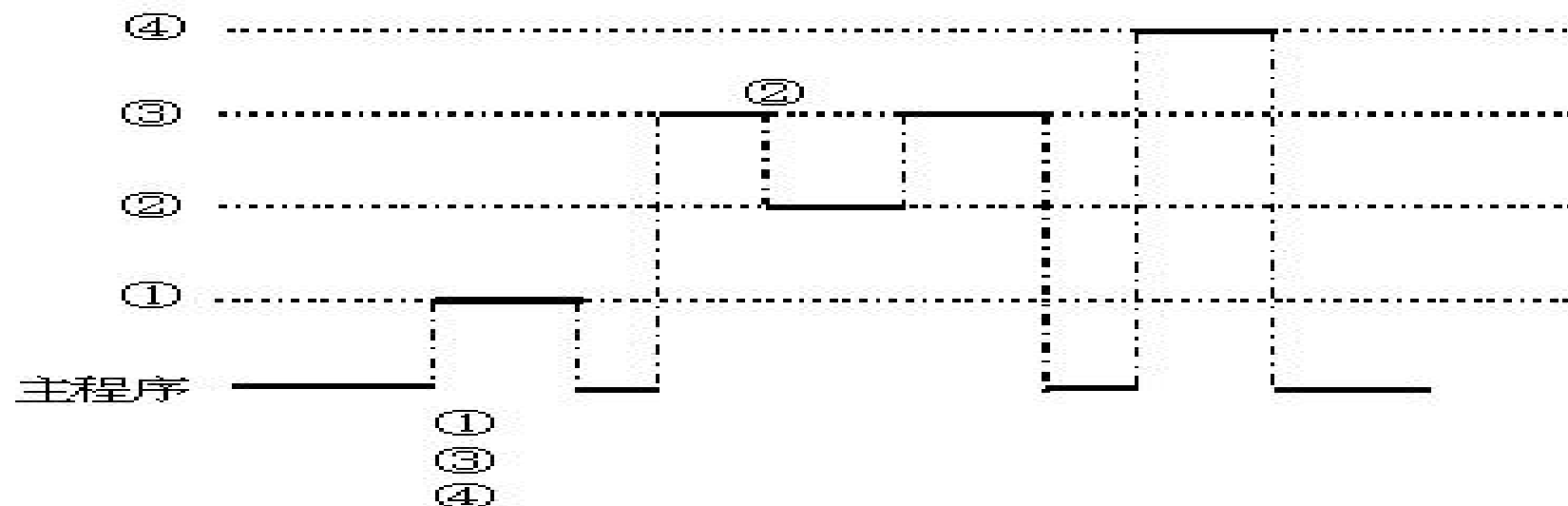
- 举例：假定某中断系统有四个中断源，其响应优先级为 $1>2>3>4$ ，分别写出处理优先级为 $1>2>3>4$ 和 $1>4>3>2$ 时各中断的屏蔽字及CPU完成中断处理的过程。

(1)中断处理优先级为 $1>2>3>4$ 时：

中断程序级别	屏蔽字			
	1 级	2 级	3 级	4 级
第 1 级	1	1	1	1
第 2 级	0	1	1	1
第 3 级	0	0	1	1
第 4 级	0	0	0	1

(假定 1 是屏蔽，0 是开放)

中断服务程序



中断优先权的动态分配

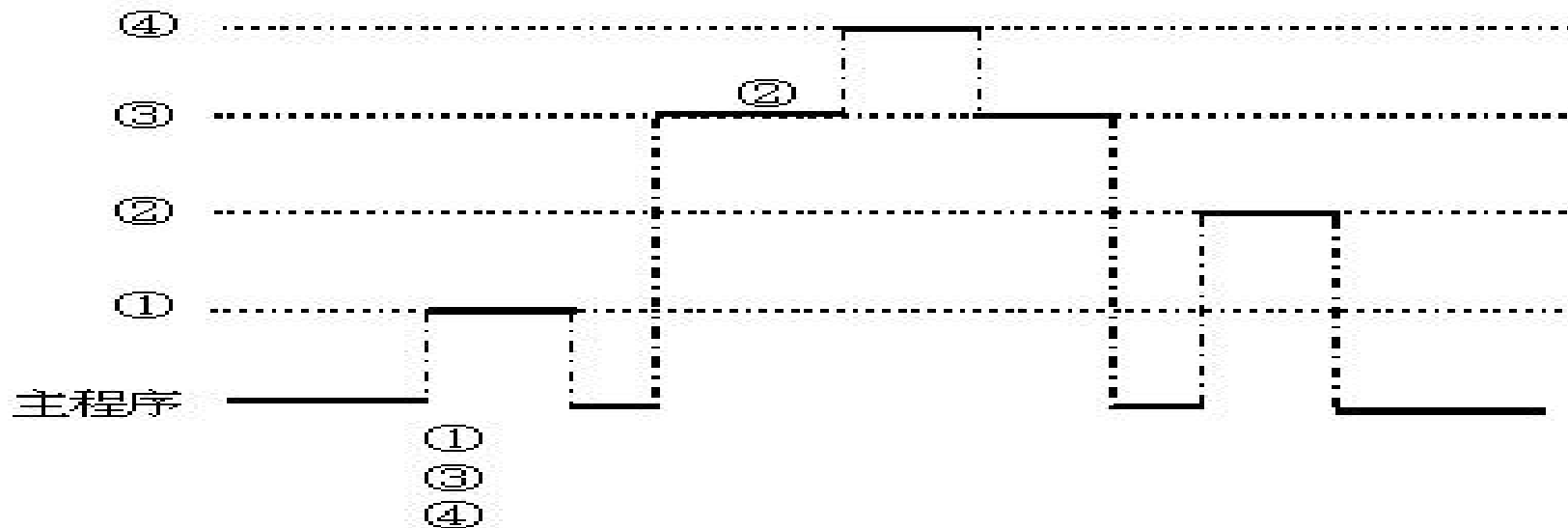
- 举例：假定某中断系统有四个中断源，其响应优先级为 $1>2>3>4$ ，分别写出处理优先级为 $1>2>3>4$ 和 $1>4>3>2$ 时各中断的屏蔽字及CPU完成中断处理的过程。

(1)中断处理优先级为 $1>4>3>2$ 时：

中断程序级别	屏蔽字			
	1 级	2 级	3 级	4 级
第 1 级	1	1	1	1
第 2 级	0	1	0	0
第 3 级	0	1	1	0
第 4 级	0	1	1	1

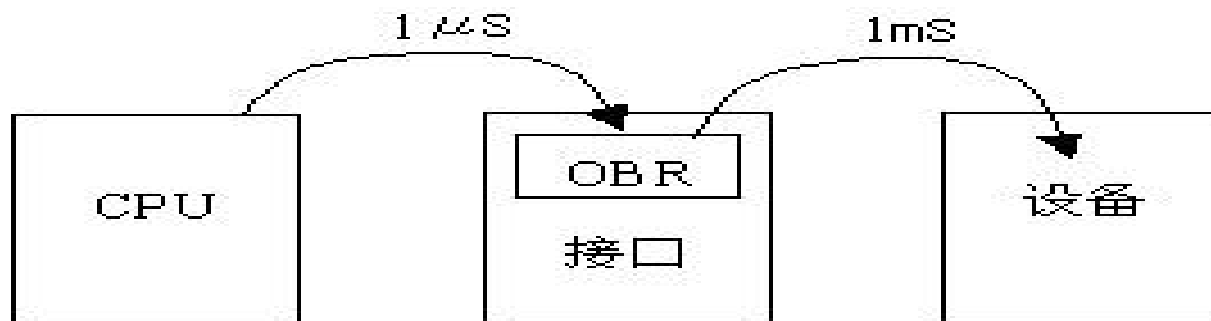
(假定 1 是屏蔽，0 是开放)

中断服务程序



轮询方式和中断方式的比较

- 举例：假定某机控制一台设备输出一批数据。数据由主机输出到接口的数据缓冲器**OBR**，需要 $1\mu\text{s}$ 。再由**OBR**输出到设备，需要 1ms 。设一条指令的执行时间为 $1\mu\text{s}$ (包括隐指令)。试计算采用程序传送方式和中断传送方式的数据传输速度和对主机的占用率。



对主机占用率：

在进行I/O操作过程中，处理器有多少时间花费在输入/出操作上。

数据传送速度（吞吐量、I/O带宽）：

单位时间内传送的数据量。

轮询方式和中断方式的比较

(1) 程序直接控制传送方式

若查询程序有**10**条，第**5**条为启动设备的指令，则：

数据传输率为： **$1/(1000+5) \mu s$** ，约为每秒**995**个数据。

主机占用率=**100%**

(2) 中断传送方式

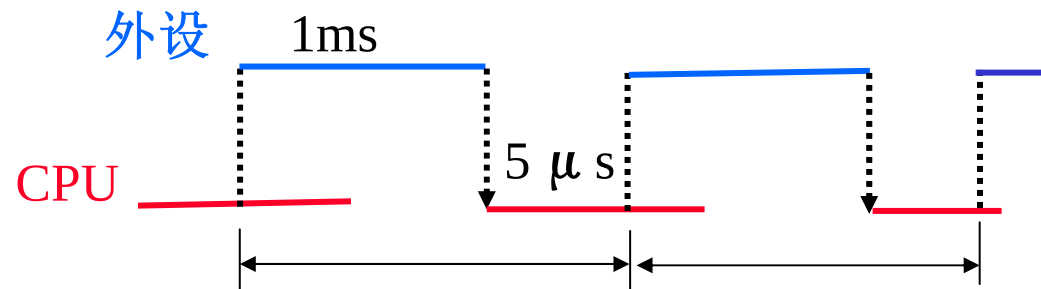
若中断服务程序有**30**条，
在第**20**条启动设备，则：

数据传输率为：

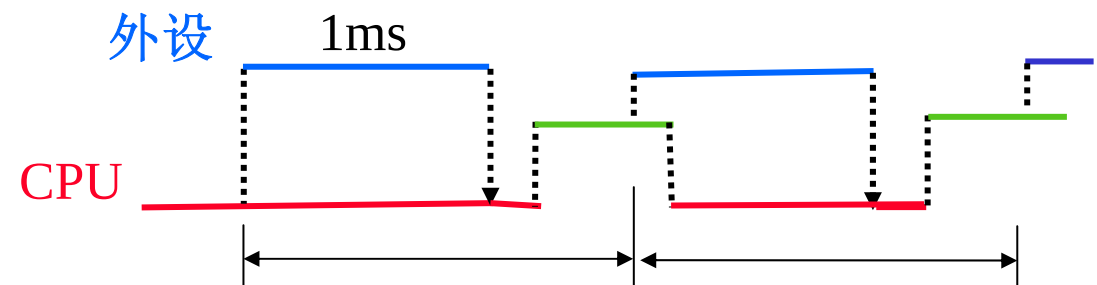
$1/(1000+1+20) \mu s$ ，约为
每秒**979**个数据。

主机占用率为：

$(1+30)/(1000+1+20)=3\%$



程序传送方式



中断传送方式

问题：为什么中断服务程序比查询程序长？

额外开销，如：保存现场、保存旧屏蔽字、开中断、（查询中断源）等

DMA输入/出方式

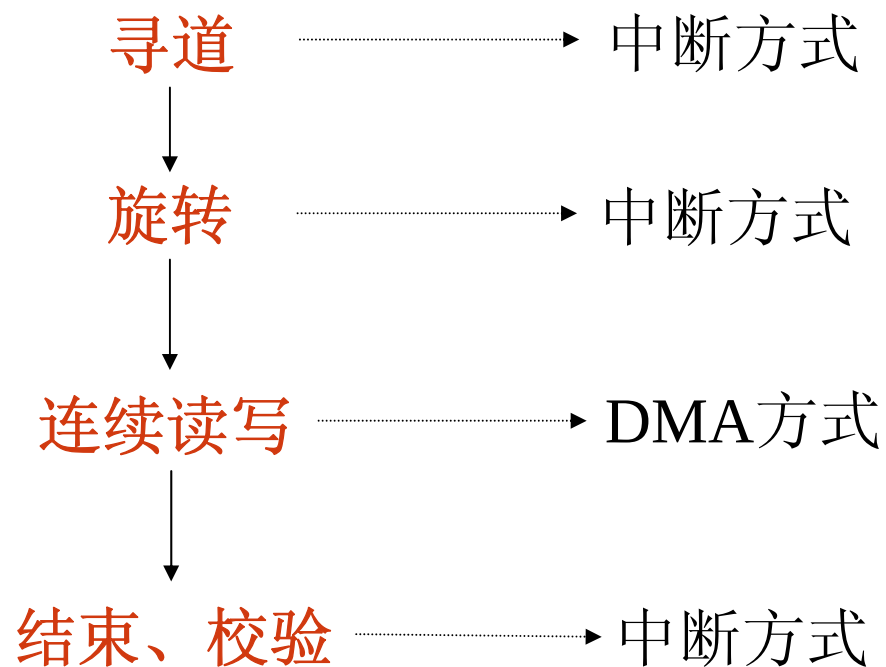
- DMA的全称
 - 直接存储器存取 (**Direct Memory Access**)
- 为什么要引入**DMA**方式?
 - 程序直接控制方式受“踏步”现象的限制，效率低下，不适合高速设备和主机间的数据传送。
 - 中断控制方式虽比程序直接控制方式有效，**CPU**和外设有一定的并行度，但由于下列原因也不适合高速设备和主机间的数据传送。
 - 对**I/O**请求响应慢。每传送一个数据都要等待外设的中断请求，并增加许多中断响应和中断处理前、后的附加开销（保护断点、现场等），不能及时响应**I/O**请求。
 - 数据传送速度慢。数据传送由软件完成（由**CPU**执行相应的中断服务程序来完成），速度慢。

DMA方式的基本要点

- **DMA方式的基本思想**
 - 在高速外设和主存间直接传送数据
 - 由专门硬件（即：**DMA接口**）控制总线进行传输
- **DMA方式适用场合**
 - 高速设备（如：磁盘、光盘等）
 - 成批数据交换，且单位数据间的时间间隔较短
- 采用“请求-响应”方式
 - 每当高速设备准备好数据，就进行一次“**DMA请求**”，**DMA控制器**接受到**DMA请求**后，申请总线使用权
 - **DMA控制器的总线使用优先级比CPU高**，为什么？
- 与中断控制方式结合使用
 - **DMA传送前**，“寻道”“旋转”等操作结束时，通过“中断”告知**CPU**
 - 在**DMA控制器**控制总线进行数据传送时，**CPU**执行其他程序
 - **DMA传送结束时**，要通过“**DMA结束中断**”告知**CPU**

与中断控制方式结合使用

- 举例：用于磁盘和主存间数据交换时



DMA数据传送方式

由于**DMA**接口和**CPU**共享主存，所以可能出现两者争用主存的现象，为使两者协调使用主存，**DMA**通常采用以下三种方式进行数据传送。

(1) CPU停止法(成组传送)

DMA传输时，**CPU**脱离总线，停止访问主存，直到**DMA**传送一块数据结束。

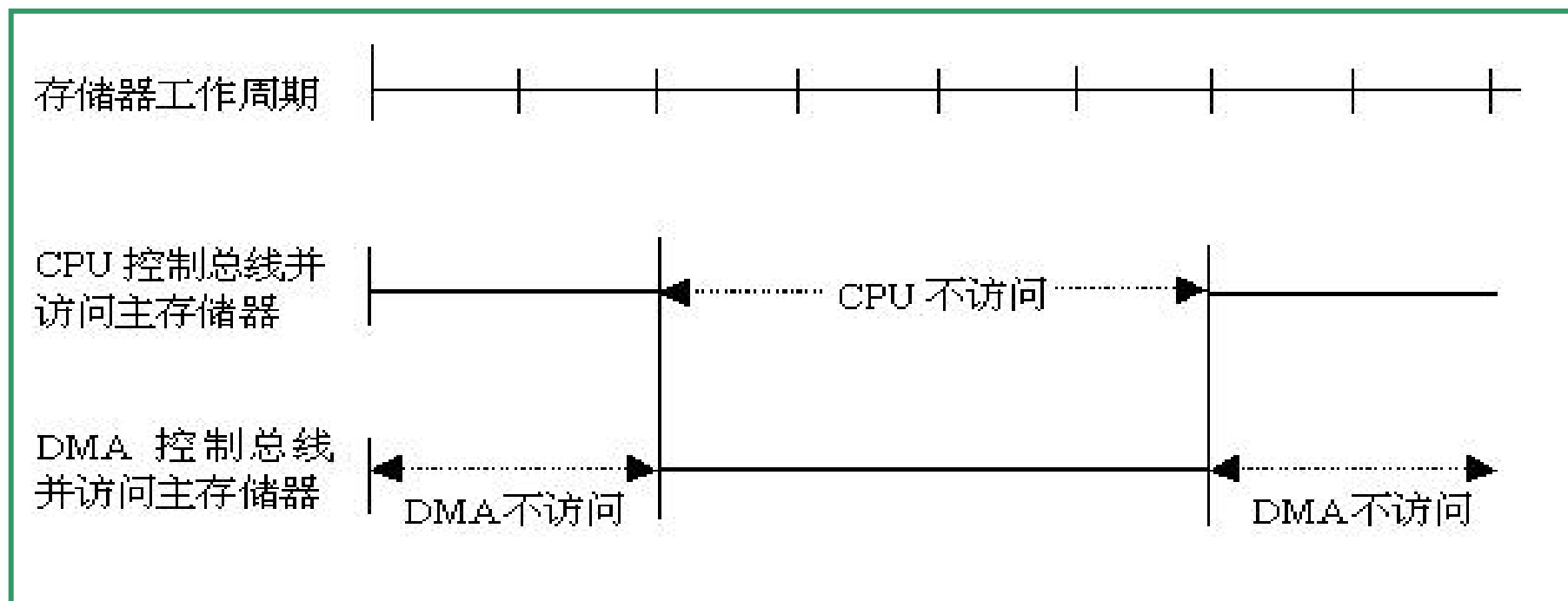
(2) 周期挪用(窃取)法(单字传送)

DMA传输时，**CPU**让出一个总线事务周期，由**DMA**控制总线来访问主存，传送完一个数据后立即释放总线。

(3) 交替分时访问法

每个存储周期分成两个时间片，一个给**CPU**，一个给**DMA**，这样在每个存储周期内，**CPU**和**DMA**都可访问存储器。

CPU停止法



优点：控制简单、适用于传输率很高的外设实现成组数据传送。

缺点：**CPU**工作受影响。**DMA**访存时**CPU**基本上处于停止状态。主存周期没有被充分利用。即使**I/O**设备高速运行，但两个数据之间的准备间隔时间也总大于一个存储周期，所以主存周期没有被充分利用。

CPU停止法

- 弥补**CPU**停止法缺点的做法：

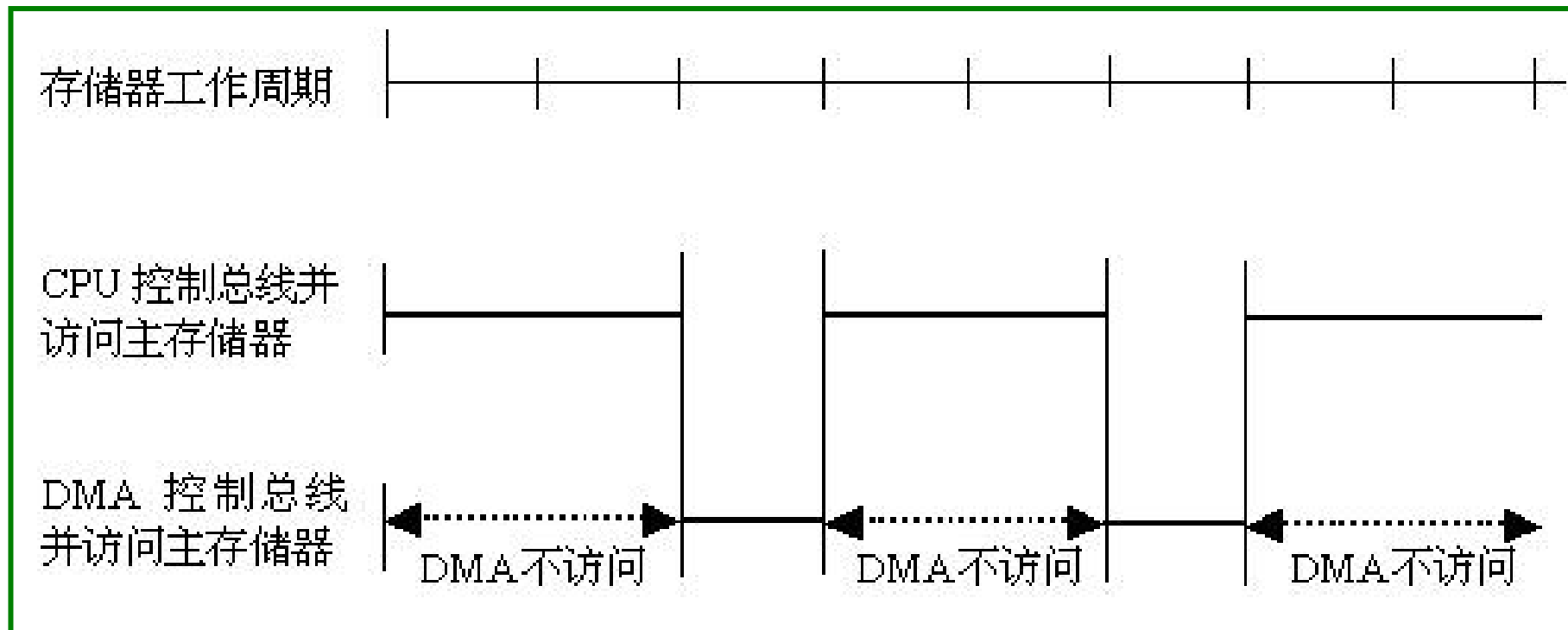
- 在**DMA**接口中引入缓冲器

在**DMA**接口中采用一个小容量的半导体存储器，使**I/O**设备先和这个小容量存储器交换数据，然后再使用总线由小容量存储器与主存进行数据交换。这样可减少**DMA**传送数据时占用总线的时间，也就减少了**CPU**的等待时间。

- 采用周期挪用（窃取）法

每次**DMA**传送完一个数据就释放总线，使在外设准备下一数据时，**CPU**能插空访问主存。

周期挪用法



优点：既能及时响应I/O请求，又能较好地发挥CPU和主存的效率。这种方式下，在下一数据的准备阶段，主存周期被CPU充分利用。因此适合于I/O设备的读写周期大于主存周期的情况。

缺点：每次DMA访存都要申请总线控制权、占用总线进行传送、释放总线，因此，会增加传输开销。

周期挪用法

◦ **I/O**设备要求**DMA**传送时可能会遇到以下三种情况之一：

- **CPU**不需访问主存

此时，不会发生冲突，两者并行。

如: **CPU**正在执行乘法指令，要花很长时间而不需马上访存。

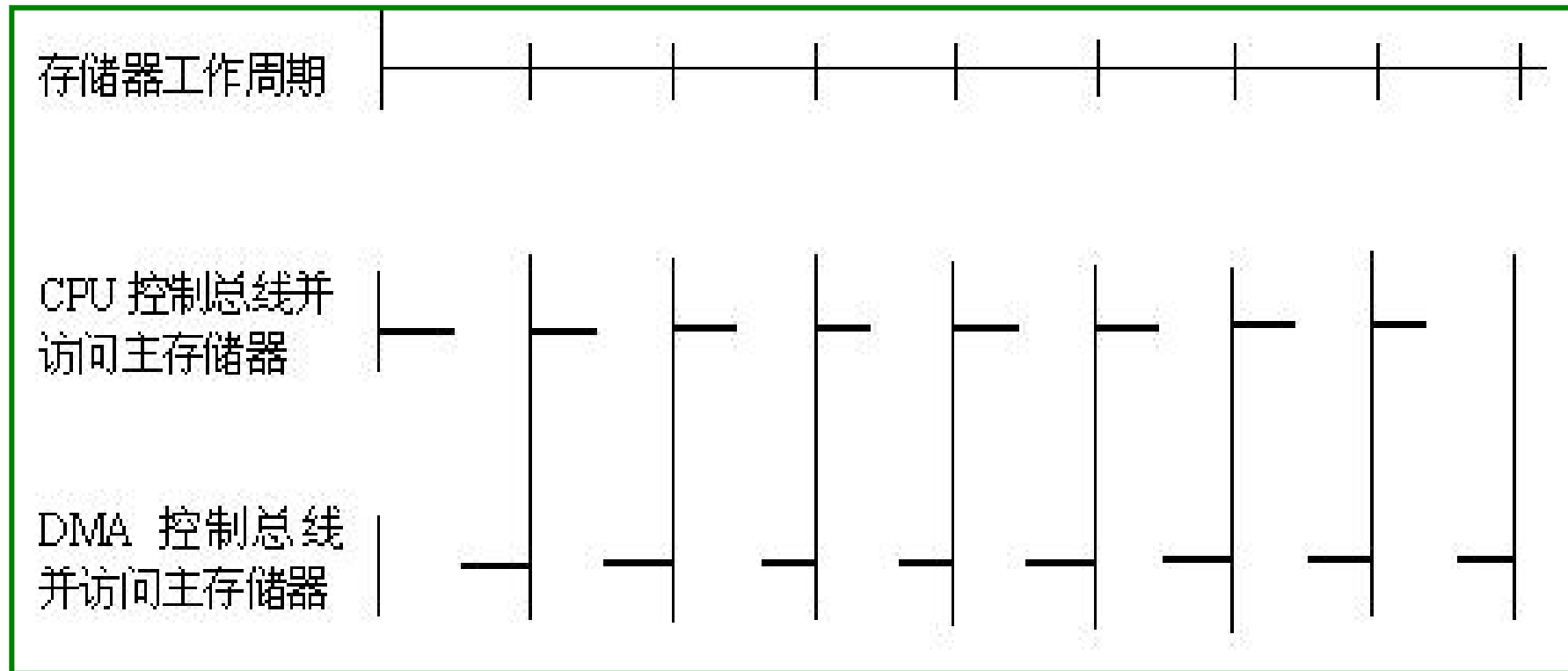
- **CPU**正在访问主存

此时须等到存储周期结束，**CPU**让出总线，**DMA**才能访存。

- **CPU**也同时要访问主存

此时出现访存冲突。因为不马上响应**DMA**请求的话，高速设备可能会发生数据丢失，所以，**DMA**的总线优先权比**CPU**高。这时，先让**DMA**占用总线，窃取一个主存周期，完成一个数据的交换。这样，**CPU**便要延迟一段时间才能访存。

交替分时访问法



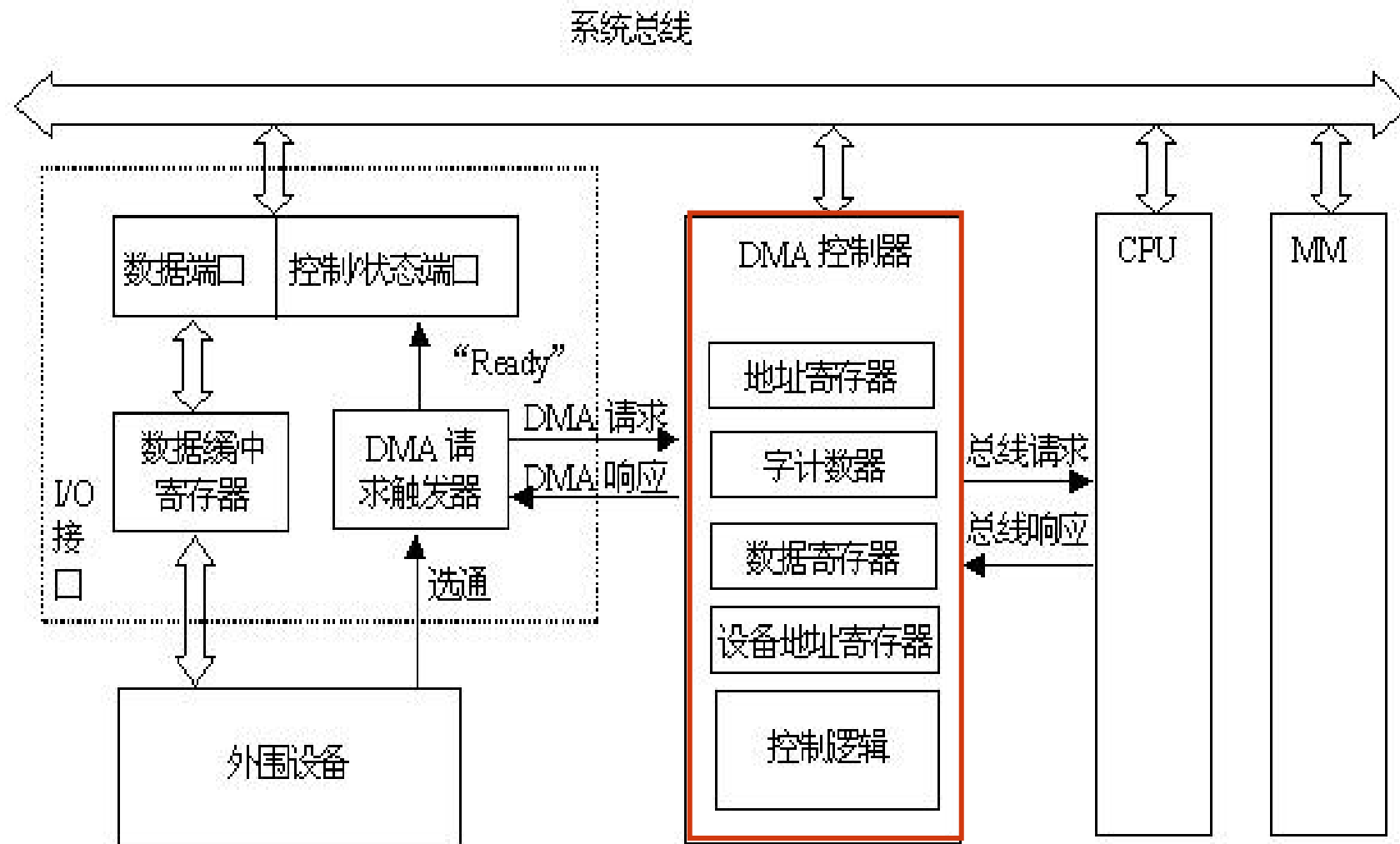
特点：适用于**CPU**工作周期比主存存取周期更长的情况。

不需要总线使用权的申请和释放。

在这种方式下，**CPU**既不停止主程序运行也不进入等待状态，在**CPU**工作过程中，不知不觉完成**DMA**数据传送，故又被称为“透明**DMA**”方式。

DMA方式结构

- DMA方式下的系统逻辑结构



DMA控制器的功能

DMA数据传送过程由**DMA**接口的控制逻辑完成，所以**DMA**接口也称**DMA**控制器。其功能为：

- (1) **请求**。能接收外设发来的“**DMA**请求”信号，并能向**CPU**发“总线请求”信号。
- (2) **响应**。当**CPU**发出“总线响应”信号响应请求后，能接管对总线的控制。
- (3) **修改主存地址**。能在地址线上给出主存地址，并自动修改主存地址。
- (4) **识别传送方向**。能识别传送方向以在控制线上给出正确的读写控制信息。
- (5) **确定传送数据个数**。
- (6) **能发出DMA结束信号**。引起一次**DMA**中断，进行数据校验等一些后处理。

DMA控制器的操作步骤

◦ DMA操作步骤

(1) DMA控制器的预置(初始化)----软件实现

- 准备内存
- 设置参数
- 启动外设

(2) DMA数据传送----硬件实现

- DMA请求：选通-〉DMA请求-〉总线请求
- DMA响应：总线响应(CPU让出总线)-〉DMA响应
- DMA传送：DMA控制总线进行数据传送

(3) DMA结束处理----软件实现

根据计数值为“0”，发出DMA结束信号去接口控制产生DMA中断请求信号，转入中断服务程序，做一些数据校验等后处理工作。

DMA控制器的初始化

◦ DMA控制器的预置(初始化)

- 准备内存区

- * 输入：在内存设置好缓冲区

- * 输出：先在内存准备好数据

- 设置传送参数

- 执行I/O指令，测试外设状态，对DMA控制器设置各种参数：

- * 内存首址=〉地址寄存器

- * 字计数值=〉字计数器

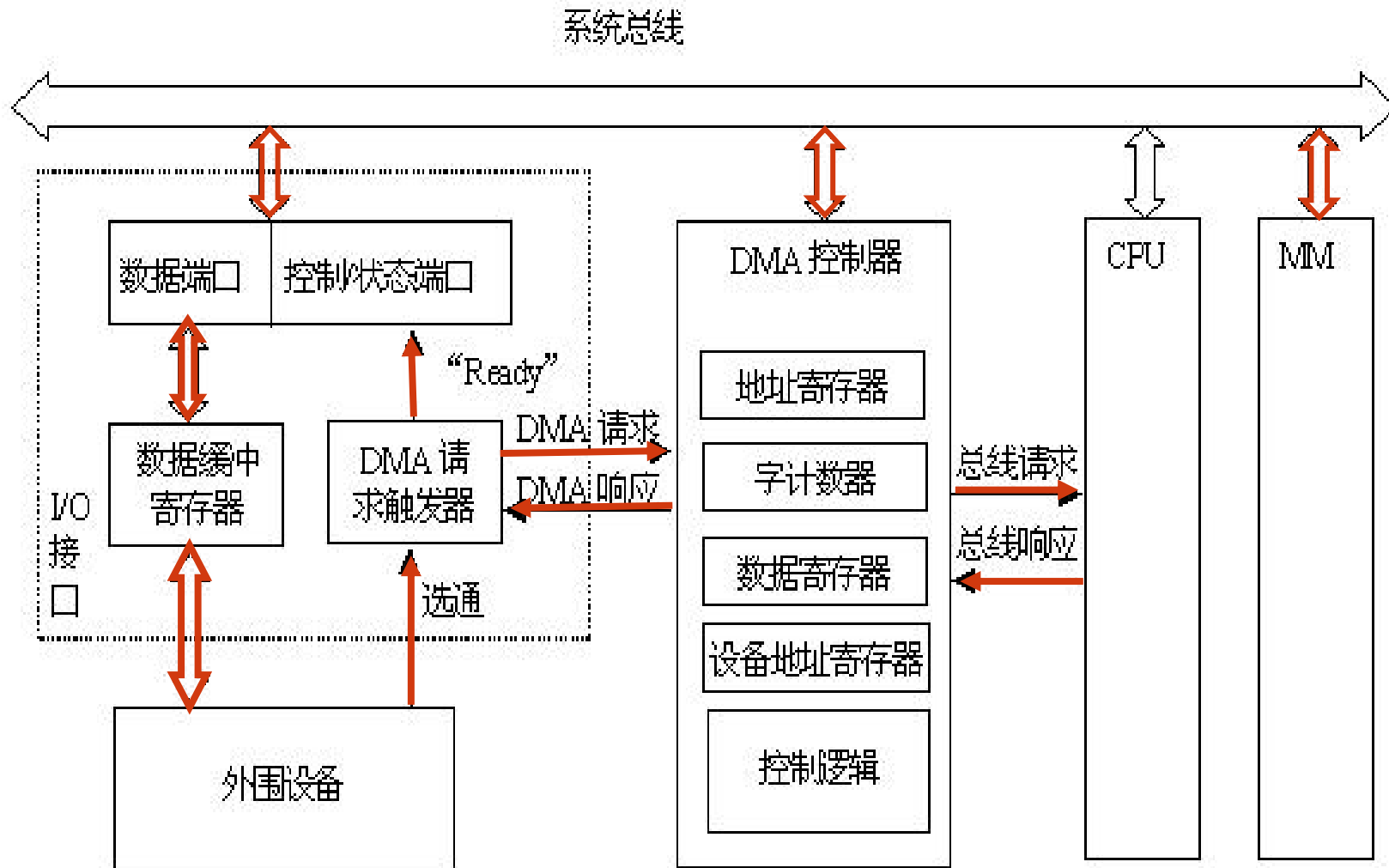
- * 传送方向=〉控制寄存器

- * 设备地址=〉设备地址寄存器

- 启动外设，然后执行其他程序

DMA传输过程

- DMA的结构



DMA传输过程

- (1) 当外设准备好数据，或准备好接收数据时，发“选通”信号，使数据送数据缓冲寄存器，同时**DMA**请求触发器置“1”。
- (2) **DMA**请求触发器向控制/状态端口发“**Ready**”信号，同时向**DMA**控制器发“**DMA**请求”信号。
- (3) **DMA**控制器向**CPU**发“总线请求”信号。
- (4) **CPU**完成现行机器周期后，响应**DMA**请求，发出“总线响应”信号。**DMA**控制器接受到该信号后，发出“**DMA**响应”信号，使**DMA**请求触发器复位。此时，**CPU**浮动它的总线，让出总线控制权，由**DMA**控制器控制总线。
- (5) **DMA**控制器给出内存地址，并在其读/写线上发出“读”或“写”命令，随后在数据总线上给出数据。
- (6) 根据读写命令，将数据总线上的数据写入存储器中，或写入数据端口，并进行主存地址增量，计数值减1，
若采用“**CPU**停止法”，则循环第6步，直到计数值为“0”。
若采用“周期挪用法”，则释放总线（下次数据传送时再按过程(1)到(6)进行）。

DMA方式和中断方式的区别

- (1) **DMA**方式下数据传送由硬件(**DMA**控制器)完成；中断方式下，数据传送由软件（**CPU**执行中断服务程序）完成。
- (2) **DMA**请求对存储器访问的请求，也即对总线控制权的请求，没有中止现行程序的必要；而中断请求要处理器转去执行中断服务程序，因此要中止现行程序，保存断点、现场等。
- (3) 中断除了能完成外设和主机的数据交换，还能处理异常事件；而**DMA**方式下不能处理异常事件。
- (4) 中断响应在一个指令周期结束后；而**DMA**响应是在一个总线周期后。
- (5) **DMA**方式用于高速设备；而中断方式用于低、慢速设备。
- (6) **DMA**方式下，外设与**CPU**并行度高；而中断方式下，外设与**CPU**并行度低。

例：中断、DMA方式下CPU的开销

设处理器按**500MHz**的速度执行，硬盘以4字块(每字**32**位)进行传送，速率为**4MB/Sec**，且没有任何数据传输被错过。

- (1) 若用中断驱动I/O，每次传送的开销（包括用于中断响应和处理的时间）是**500**个时钟周期。如果硬盘仅用**5%**的时间进行传送，那么处理器用在硬盘I/O操作上所花的时间百分比（主机占用率）为多少？
- (2) 若用**DMA**方式，处理器花**1000**个时钟进行**DMA**传送的初始化设置，并且在**DMA**完成后的中断处理需要**500**个时钟。如果从硬盘发出的平均传输量为**8KB**（即每次**DMA**传送**8KB**的数据块），那么当硬盘进行传送的时间占**100%**（即：硬盘一直进行读写，并传输数据）时，处理器用在硬盘I/O操作上的时间百分比（主机占用率）为多少？

例：中断、DMA方式下CPU的开销

◦ 中断传送：

- 硬盘要求每次中断以 4 字块 (=16字节) 进行传送，为了保证没有任何数据传输被错过，传送的速率应达到每秒 $4\text{MB}/16\text{B}=250\text{K}$ 次中断的速度；
- 每秒钟用于中断的周期数为 $250\text{K} \times 500 = 125 \times 10^6$ ；
- 在一次传输中所消耗的处理器时间的百分比为：
 $125 \times 10^6 / (500 \times 10^6) = 25\%$ ；
- 假定硬盘仅用其中 5% 的时间来传送数据，则处理器消耗的百分比为
 $25\% \times 5\% = 1.25\%$ 。

◦ DMA传送：

- 每个DMA传送将花 $8\text{KB}/(4\text{MB}/\text{Sec}) = 2 \times 10^{-3}$ 秒；
- 一秒钟有 $1/(2 \times 10^{-3}) = 500$ 次DMA传送；
- 如果硬盘一直在传送数据的话，处理器必须每秒钟花
 $(1000 + 500) \times 500 = 750 \times 10^3$ 个时钟周期来为硬盘I/O操作服务；
- 在硬盘I/O操作上处理器花费的时间占：

$$750 \times 10^3 / 500 \times 10^6 = 1.5 \times 10^{-3} = 0.15\%。$$

通道和IOP方式

I/O方式的发展过程:

第一阶段: **CPU**直接控制外设。

第二阶段: 增加一个控制器或一个**I/O**模块, **CPU**使用编程**I/O**控制。

第三阶段: 采用中断技术, **CPU**不需要花费时间等待外设执行**I/O**操作, 实现了外设和**CPU**的并行。

第四阶段: **I/O**模块通过**DMA**直接传送一块数据到或从存储器传出, 不需要**CPU**全部参与。

大型计算机系统采用以下**I/O**方式:

(1) **I/O**模块具独立**I/O**处理能力, 具执行专门**I/O**程序的能力。 **CPU**只需在主存中事先组织好**I/O**程序, 发出相应的**I/O**命令即可, 对**I/O**的干预极少。

这种方式为通道方式。

(2) **I/O**模块是一个专门的**I/O**处理器, 拥有自己单独的存储器和指令集, **CPU**参与更少。这种方式为**I/O**处理器方式。

第三讲小结

- 有三种基本的I/O传输方式
 - 轮询方式（程序直接控制 / 程序查询方式）：通过查询程序定时到I/O端口取状态来查询外设和I/O控制器的状况，以控制设备进行相应的动作
 - 程序中断方式（中断驱动方式）：当外设完成任务或发生特殊情况时，由外设主动向CPU提出中断请求，CPU在每条指令结束后查询有无中断请求，有则转内核态，调出操作系统中的中断处理(服务)程序来处理外设请求。在中断处理程序中完成CPU和外设的数据传送
 - 中断响应过程（硬件-处理器）：关中断、保护断点、转中断服务程序
 - 中断服务程序的入口地址可以是一个中断查询程序入口，也可以由中断控制器给出中断类型号，再根据类型号到中断向量表中取入口地址
 - 中断处理过程（软件-OS）：准备阶段、处理阶段、恢复并返回阶段
 - 中断控制器：记录所有中断请求，在中断掩码（屏蔽码）的作用下，对所有未被屏蔽的中断请求进行优先权编码，送出优先级最高的中断类型号给CPU
 - 多重中断：在处理中断时又发生新中断请求的处理机制
 - 直接存储器访问方式（DMA方式）：用于磁盘等高速外设与主存之间直接数据交换
 - DMA控制器的结构：字节计数器、地址寄存器、设备地址寄存器、控制逻辑等
 - 三种控制方式：CPU停止法、周期挪用法、交替分时访问法
 - DMA传输过程：控制器初始化、数据传输、结束处理

本章总结1

- I/O系统概述
 - I/O系统的性能主要有吞吐率和响应时间，两者是对立统一的关系
 - I/O系统的功能是在主机（寄存器和主存）和外设之间传输数据
 - I/O系统的具体任务是：构建传输通路、对设备寻址、向设备发命令、取状态、并提供相应的传输机制来读/写设备数据等。（后面两讲的内容）
 - OS在I/O系统的职责是：
 - 对共享设备进行管理、提供设备驱动程序、处理中断请求
- I/O设备概述
 - I/O设备通过I/O接口和主机相连
 - 外设分类：I/O设备和存储设备、机读设备和人读设备
- 磁盘存储器
 - 磁盘存储器的读写原理：两种不同磁化状态表示“0”和“1”
 - 磁盘存储器的性能指标：寻道时间、旋转等待时间、传输时间
 - 冗余磁盘阵列 (RAID)：多个物理盘组成一个逻辑盘，以提高磁盘存取速度、容量和可靠性
- 网络：作为一种特殊的外部设备，实现计算机系统之间的数据交换

本章总结2

- **I/O接口**是**I/O控制器**和外设之间电缆连接的插座（很多教材把**I/O控制器**和**I/O接口**综合称为**I/O接口**，不严格区分控制逻辑部分（**I/O控制器**）和插座（**I/O接口**）部分）
- **I/O控制器**中一般有数据缓冲器、状态/控制寄存器、串-并转换、设备控制逻辑、地址译码逻辑等。用于在主机和设备之间进行命令、数据、状态信息的传递和转换。
- **I/O端口**是指**I/O控制器**中**CPU**可访问的寄存器，对**I/O设备**的寻址就是对**I/O端口**的访问
- **I/O端口**的编址方式有两种：内存映射方式（统一编址）和特殊**I/O**指令方式（独立编址）

本章总结3

- 有三种基本的I/O传输方式
 - 轮询方式（程序直接控制 / 程序查询方式）：通过查询程序定时到I/O端口取状态来查询外设和I/O控制器的状况，以控制设备进行相应的动作
 - 程序中断方式（中断驱动方式）：当外设完成任务或发生特殊情况时，由外设主动向CPU提出中断请求，CPU在每条指令结束后查询有无中断请求，有则转内核态，调出操作系统中的中断处理(服务)程序来处理外设请求。在中断处理程序中完成CPU和外设的数据传送
 - 中断响应过程（硬件-处理器）：关中断、保护断点、转中断服务程序
 - 中断服务程序的入口地址可以是一个中断查询程序入口，也可以由中断控制器给出中断类型号，再根据类型号到中断向量表中取入口地址
 - 中断处理过程（软件-OS）：准备阶段、处理阶段、恢复并返回阶段
 - 中断控制器：记录所有中断请求，在中断掩码（屏蔽码）的作用下，对所有未被屏蔽的中断请求进行优先权编码，送出优先级最高的中断类型号给CPU
 - 多重中断：在处理中断时又发生新中断请求的处理机制
 - 直接存储器访问方式（DMA方式）：用于磁盘等高速外设与主存之间直接数据交换
 - DMA控制器的结构：字节计数器、地址寄存器、设备地址寄存器、控制逻辑等
 - 三种控制方式：CPU停止法、周期挪用法、交替分时访问法
 - DMA传输过程：控制器初始化、数据传输、结束处理