Computer Organization and Design

Ch 8: Storage, Networks, and Other Peripherals

外存、网络与其他外设

第一讲 I/O设备、磁盘存储器和网络

第二讲 I/O总线、I/O控制器和I/O接口

第三讲 I/O设备和主机之间的传输方式

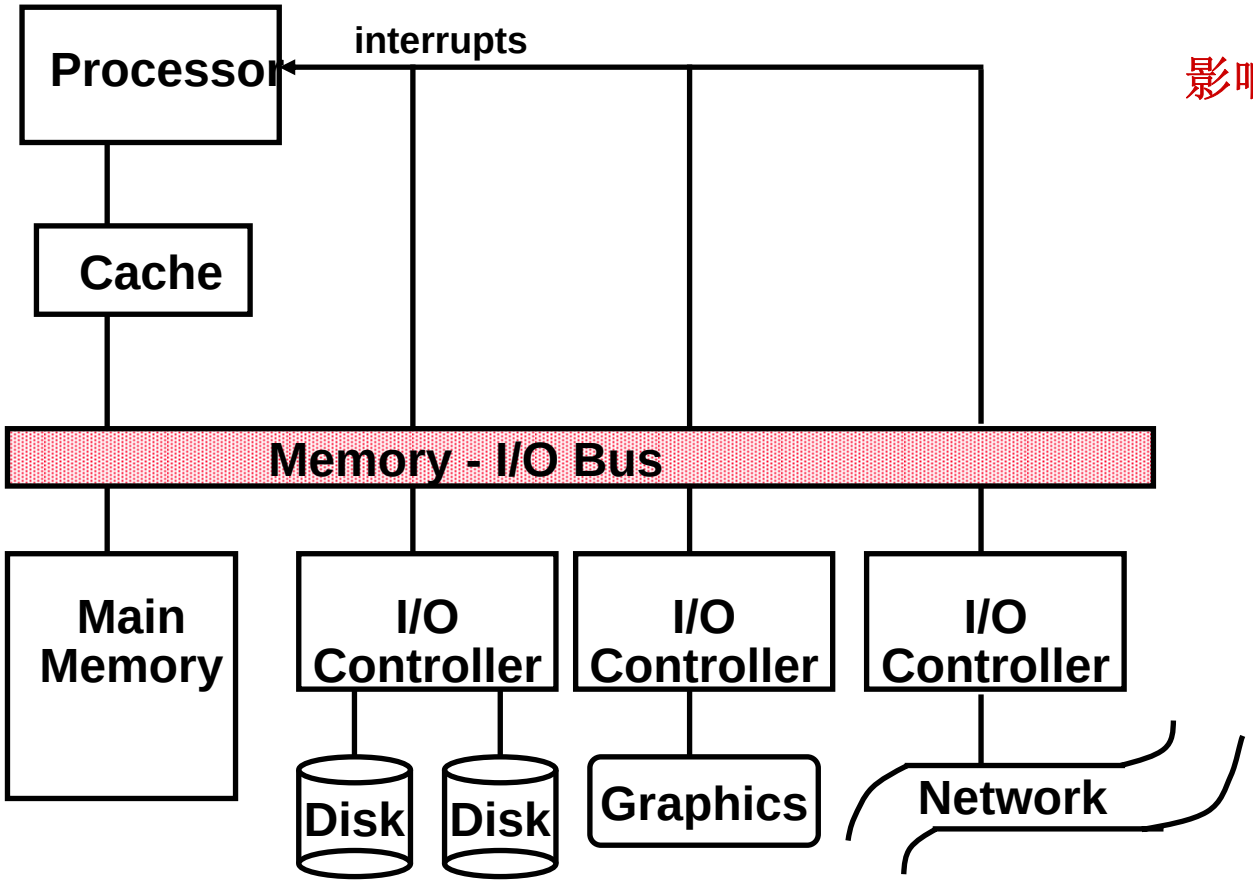
第一讲 I/O设备、磁盘存储器和网络

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
  - 常用的I/O设备
- 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)
- 网络的分类和连网设备

I/O System 设计时考虑的问题

- Performance and cost (性能和成本)
- Expandability (可扩充性和多样性)
- Resilience in the face of failure (故障恢复能力-可靠性)



PC机的I/O设备的典型连接方式

影响 I/O 性能的因素很多：

The CPU

The memory system:

Caches / Main Memory

interconnection (buses)

The I/O controller

The I/O device

I/O软件的速度

I/O设备的使用效率

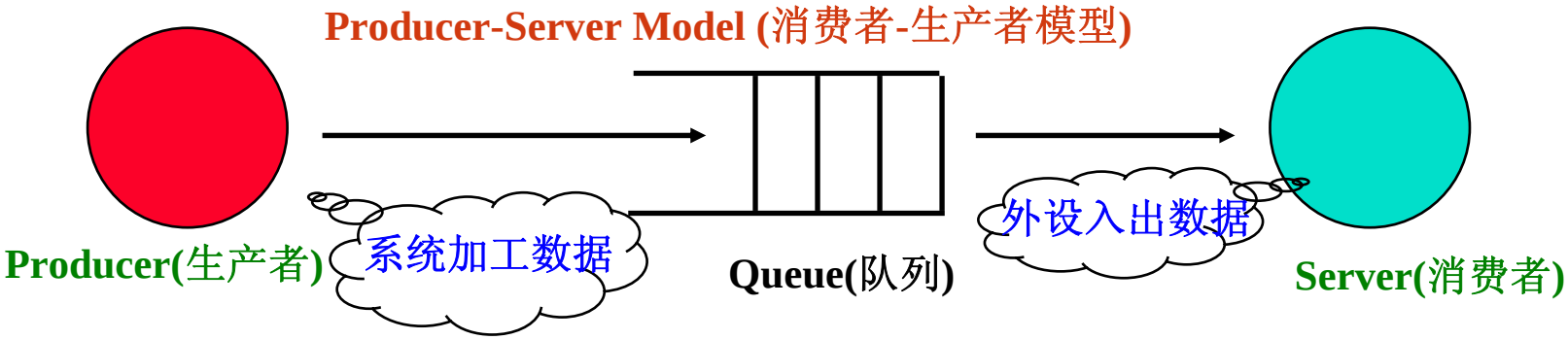
I/O System的性能

- 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？

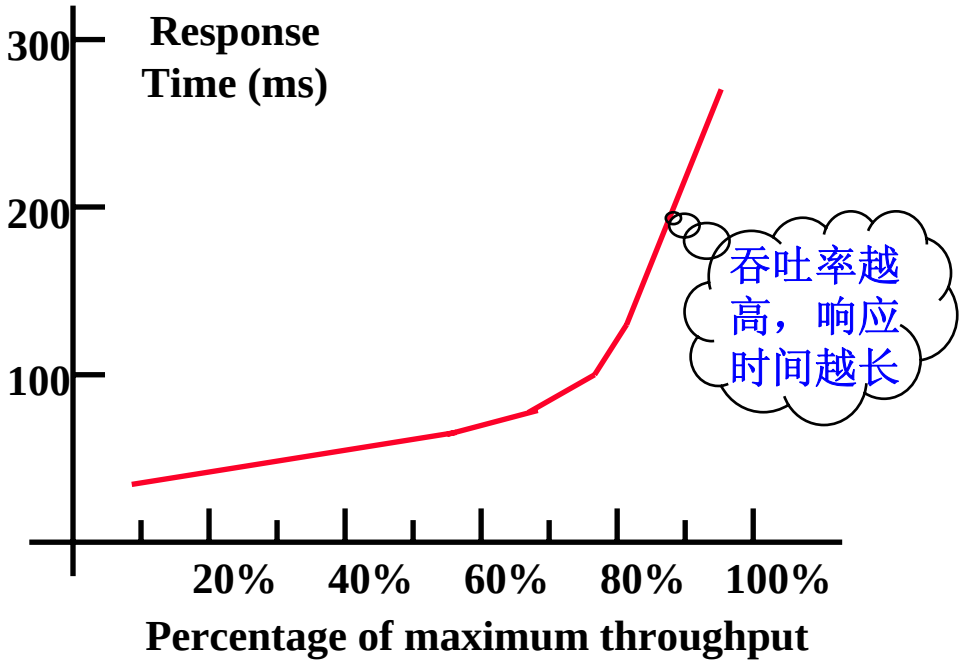
      (服务器更关注)
  - **Response time: Latency (** 响应时间，即：等待延迟**)**：
    - 在多长时间内完成请求的任务？

      (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

Throughput和Response time的关系



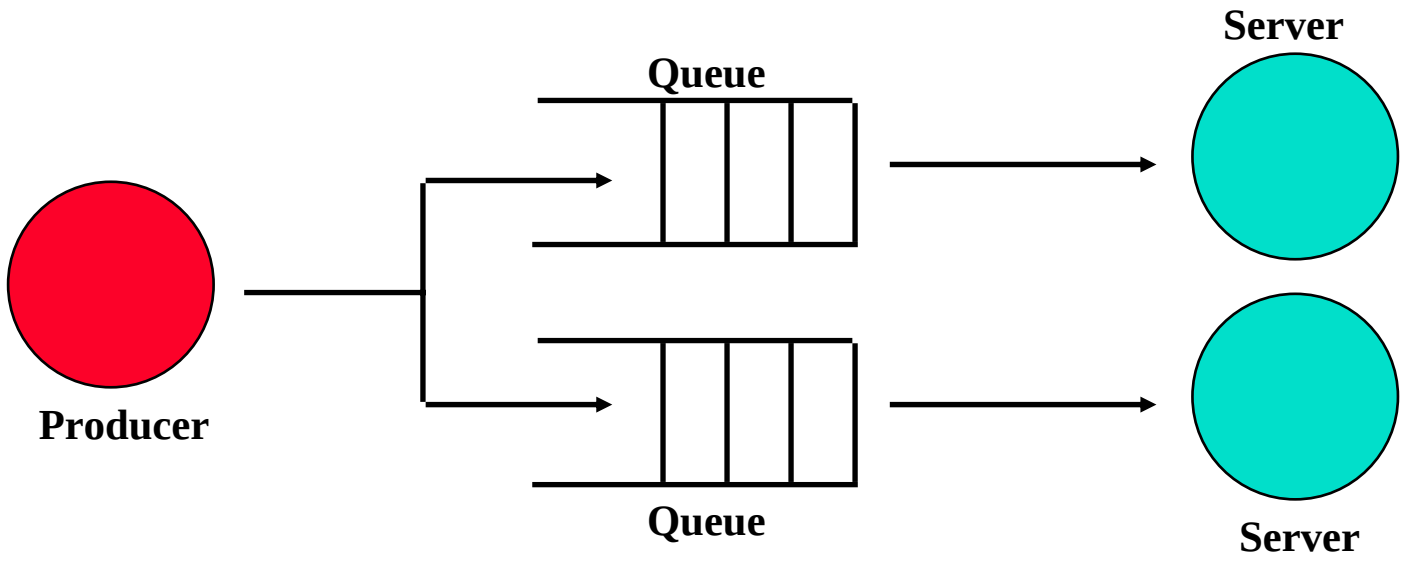
- **Throughput (吞吐率):**
  - 消费者在单位时间内完成的任务数
  - 如何最大程度地提高吞吐率？
    - 消费者一直不能空闲
    - 队列应该永远不空
- **Response time (响应时间):**
  - 从任务入队列开始, 到消费者完成
  - 如何降低响应时间？
    - 队列应该空
    - 消费者需空闲



吞吐率和响应时间是一对矛盾！

如何在两者之间进行很好权衡？

增加吞吐率的措施



- 增加吞吐率的措施：
  - 使用更多的硬件：（就像：四车道改成八车道）
  - 成批进行：（就像：汽车改成火车（车厢多））
- 响应时间的缩短比较困难，最终受光速的限制！
  - 只有让车跑得更快！

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

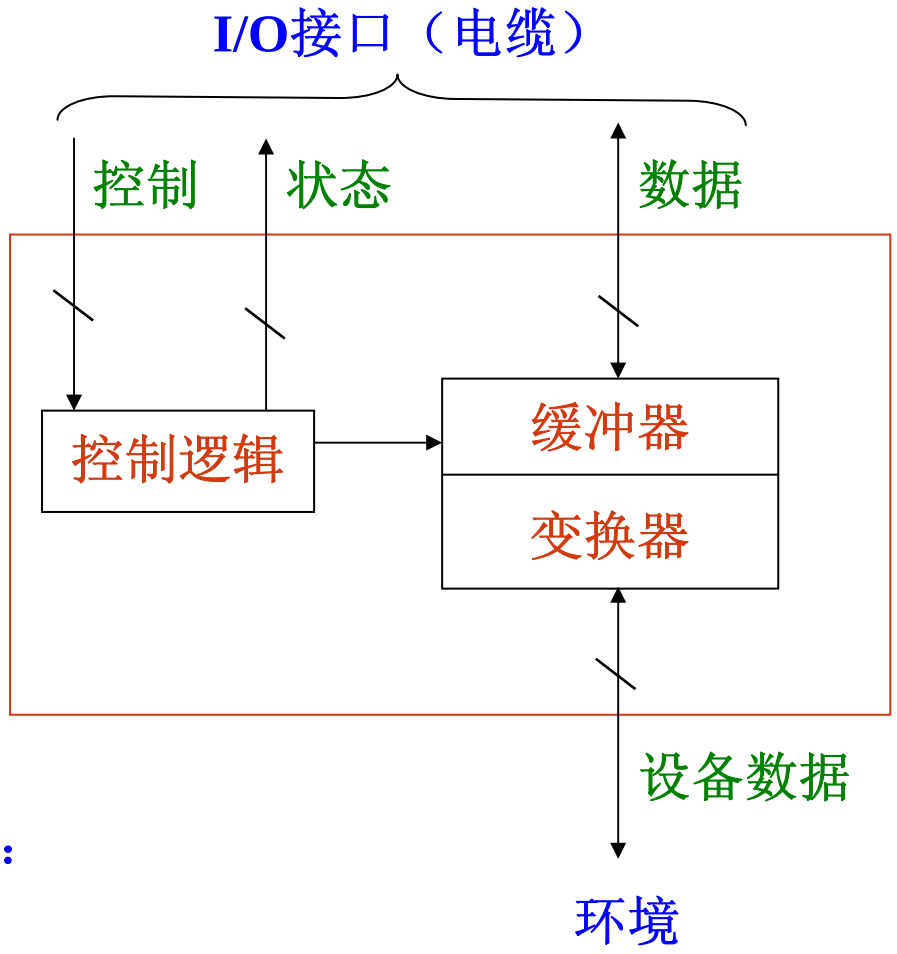
- 从交互方式上来分，外设分为：
  - 人-机交互设备
    - 输入/输出的信息是人可读的
    - 如：键盘、鼠标、扫描仪、打印机、显示器等
  - 机器可读设备
    - 输入/输出的信息是机器可读的，人无法读取
    - 如：网络、Modem、D/A、A/D、磁盘、声音输入设备等
- 从功能行为来分，外设分为：
  - 输入/输出设备
    - 用于信息的输入/输出
    - 输入设备：键盘、鼠标、扫描仪等
    - 输出设备：打印机、显示器等
  - 外部存储设备
    - 用于信息的存储（其输入/出的信息是机器可读的）
    - 如：磁盘、磁带、光盘等

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



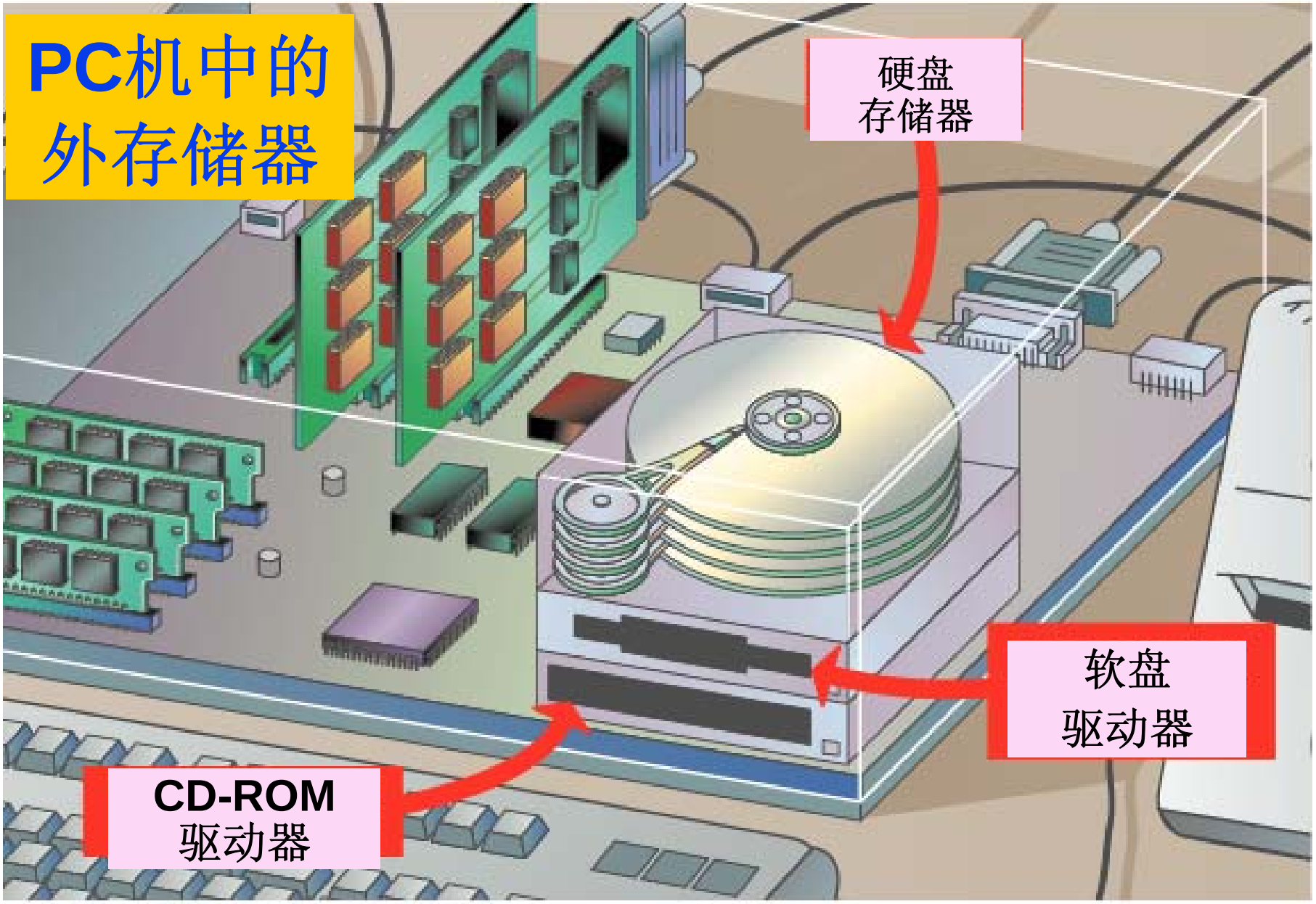
所有设备都可以抽象成这个通用模型！

设备所用的电缆线中有以下三种信号线：

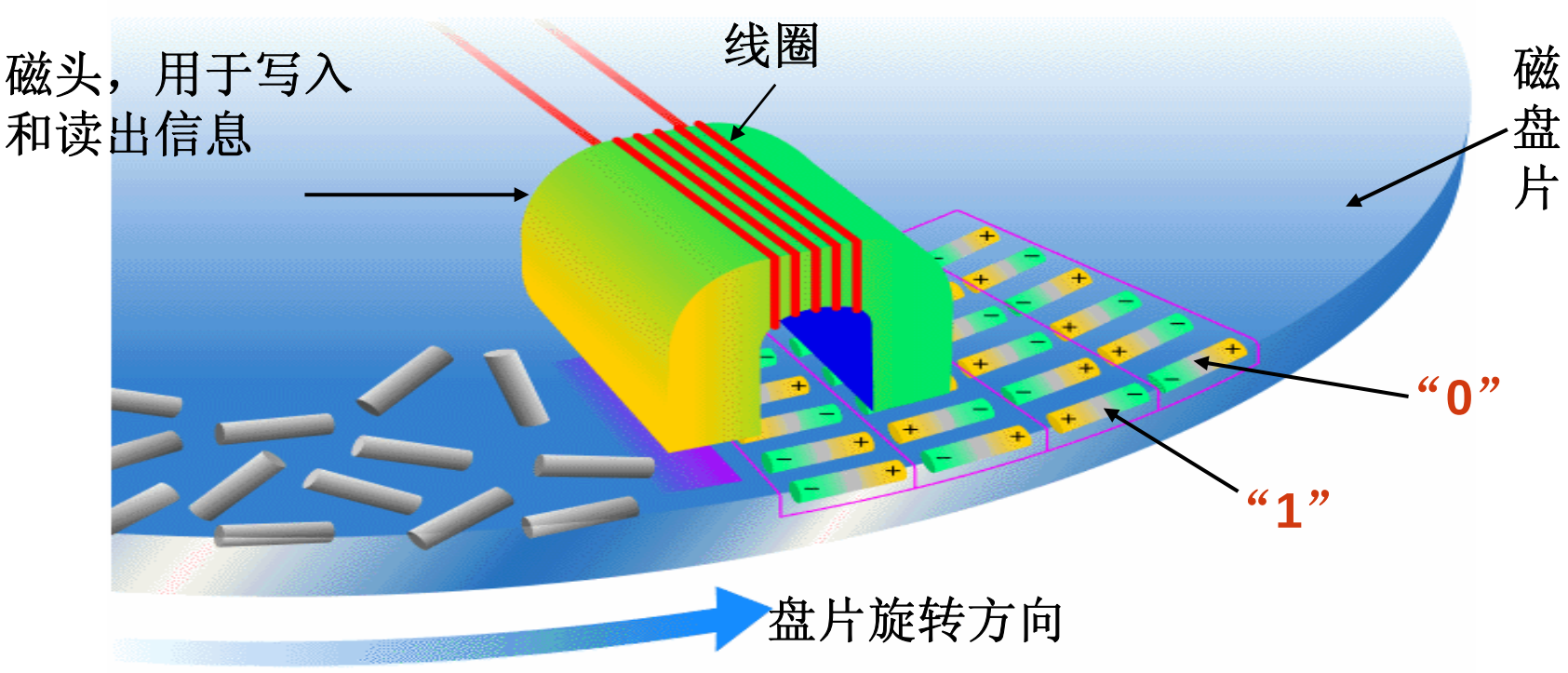
控制信号、状态信号、数据信号

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



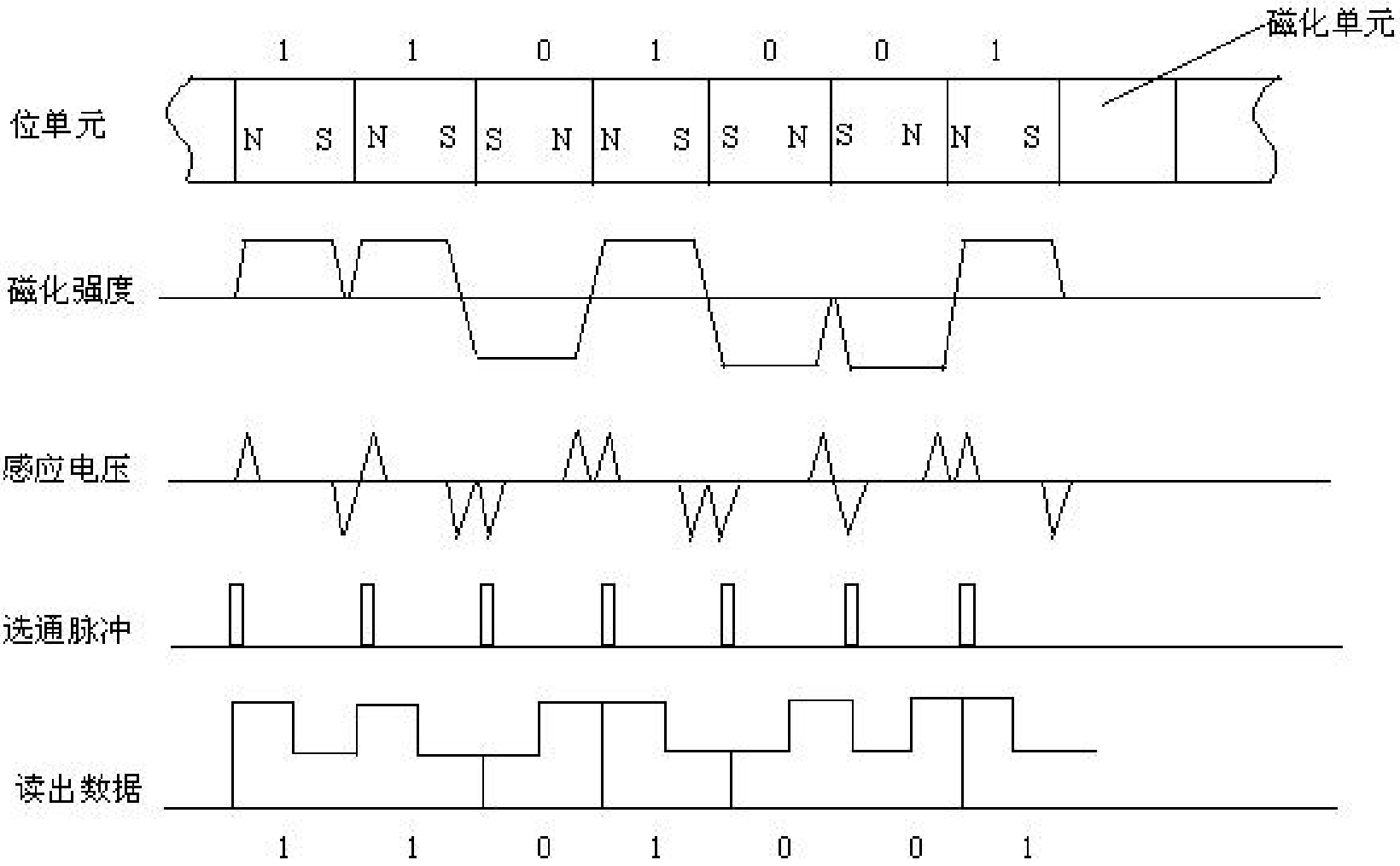
写1：线圈通以正向电流，使呈N-S状态

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

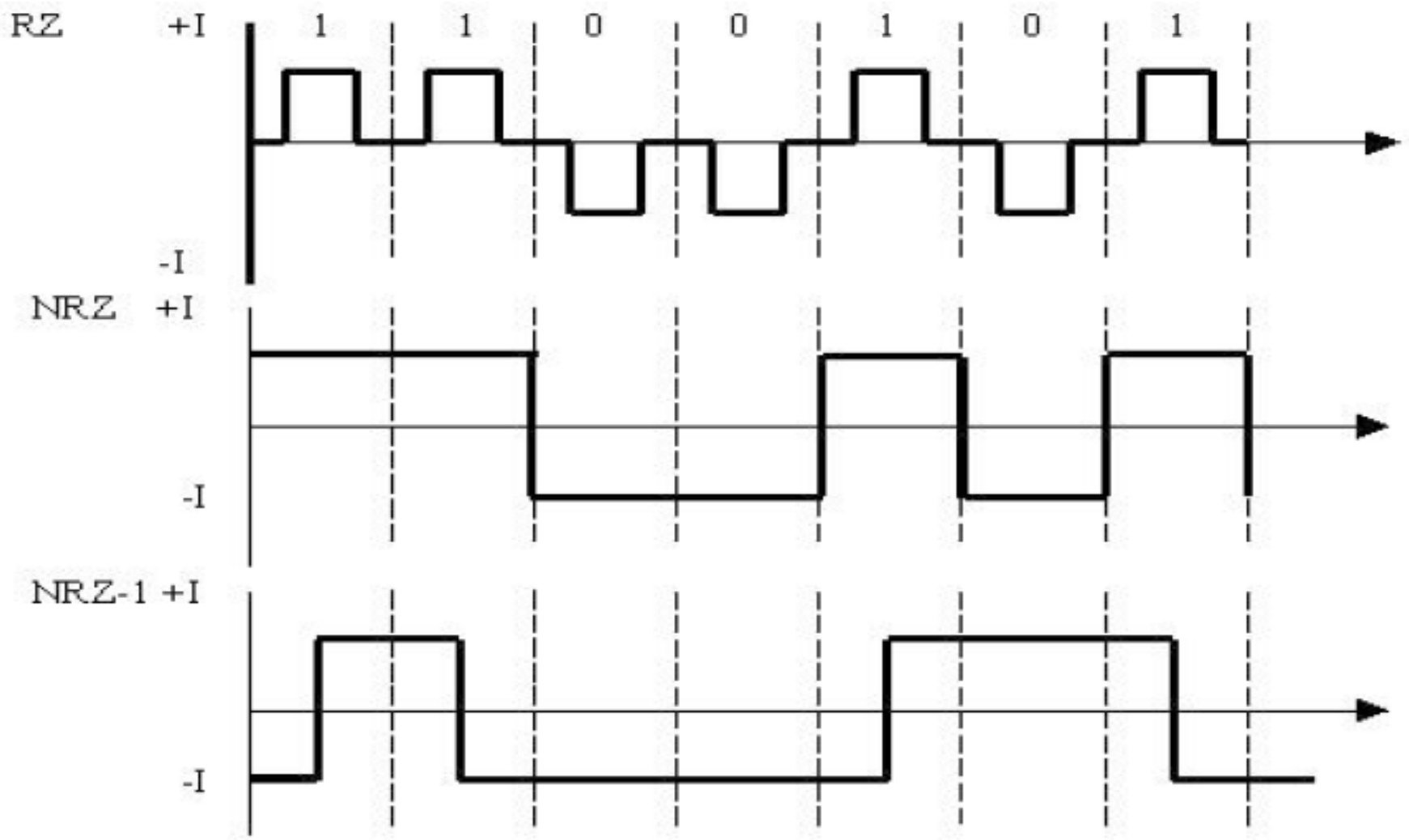


（自学）磁表面记忆原理和记录方式

- 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

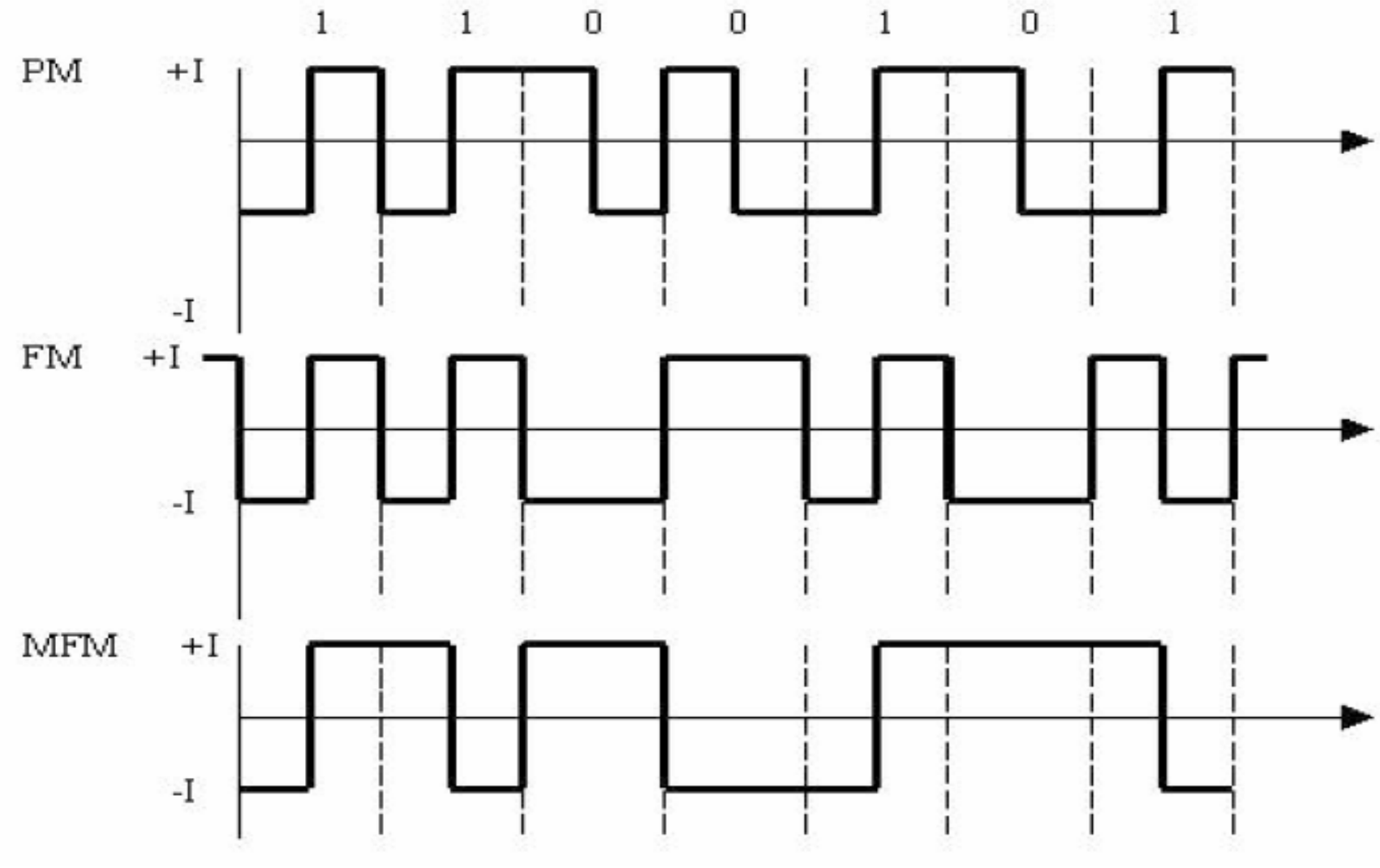
SKIP

（自学）磁记录方式1



BACK

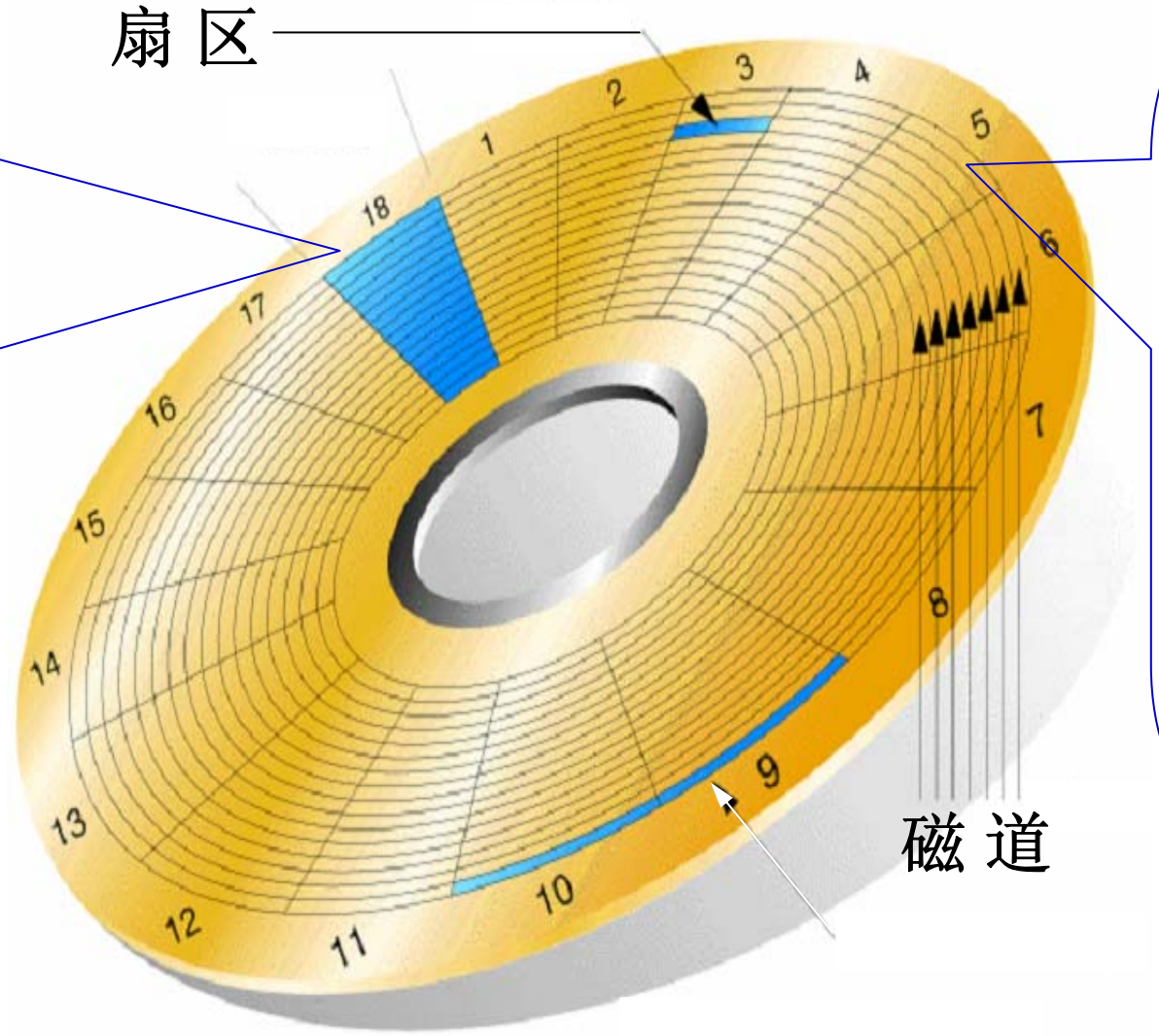
（自学）磁记录方式2



BACK

回顾：磁盘的磁道和扇区

每个磁道被划分为若干段（段又叫扇区），每个扇区的存储容量均为**512**字节。每个扇区都有一个编号

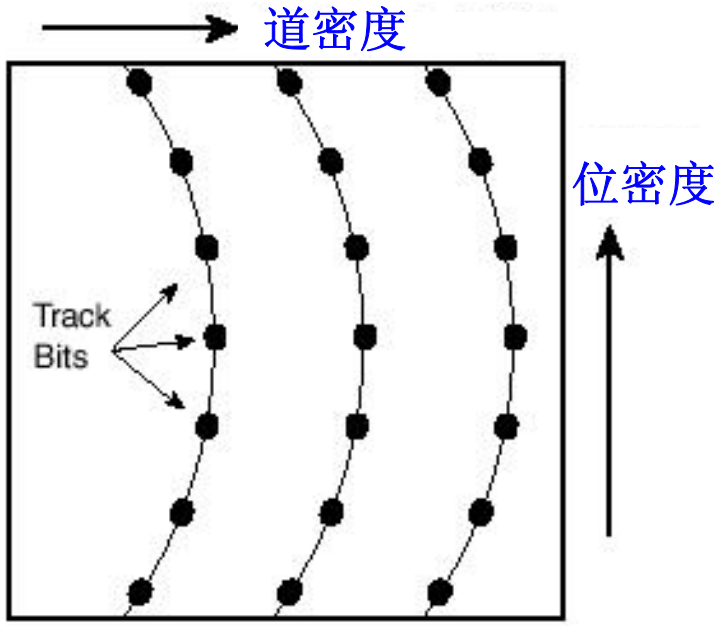


磁盘表面被分为许多同心圆，每个同心圆称为一个磁道。每个磁道都有一个编号，最外面的是**0**磁道

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

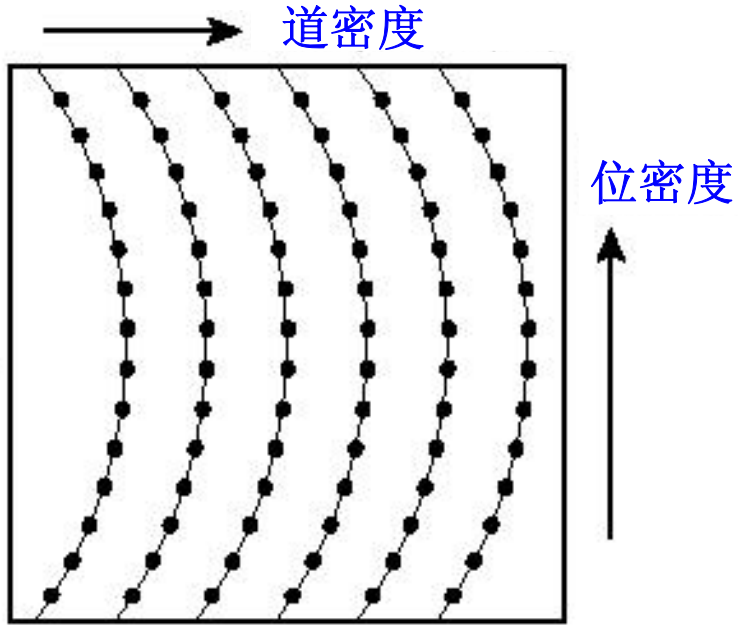
回顾：如何增大磁盘片的容量？

- 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

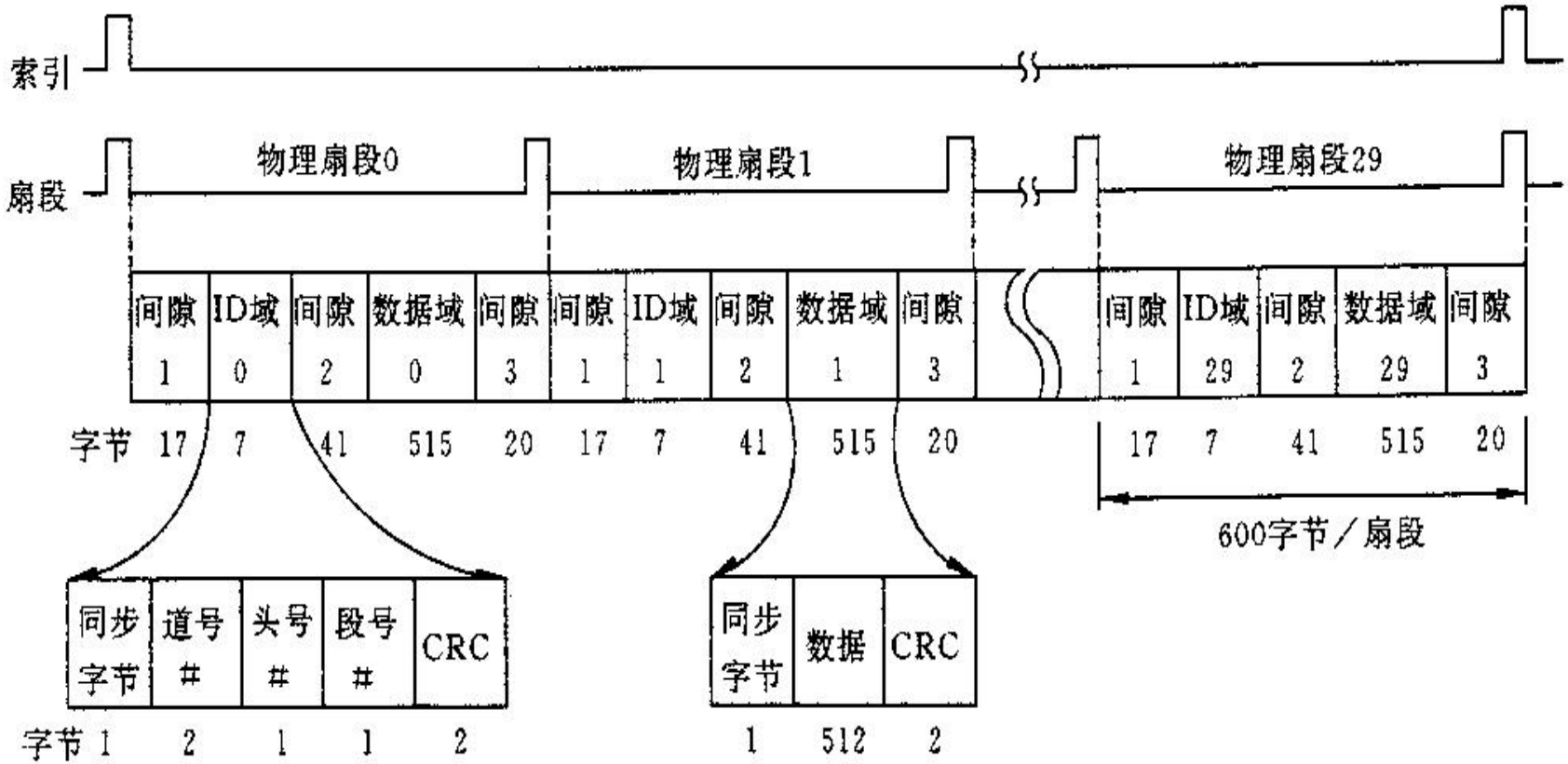


图 5.2 温彻斯特磁盘磁道格式(Seagate ST506)

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)。

回顾：3.5英寸软盘片存储容量的计算

- 共 2个记录面（双面记录）
- 每个记录面 80个磁道 (编号:0-79)
- 每个磁道划分为 18个扇区 (编号:1-18)
- 每个扇区的存储容量为 512字节
- 所以，每张盘片的总存储容量为

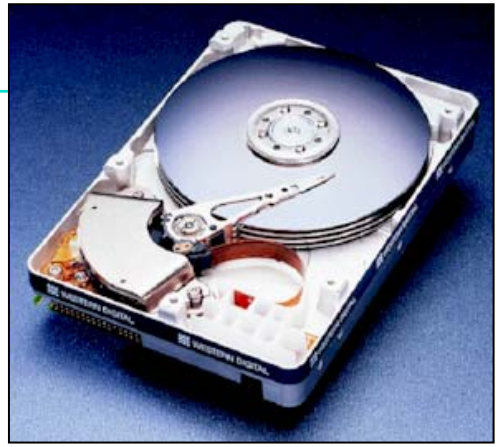
$$存储容量= 2 \times 80 \times 18 \times 512 = 1.44MB$$

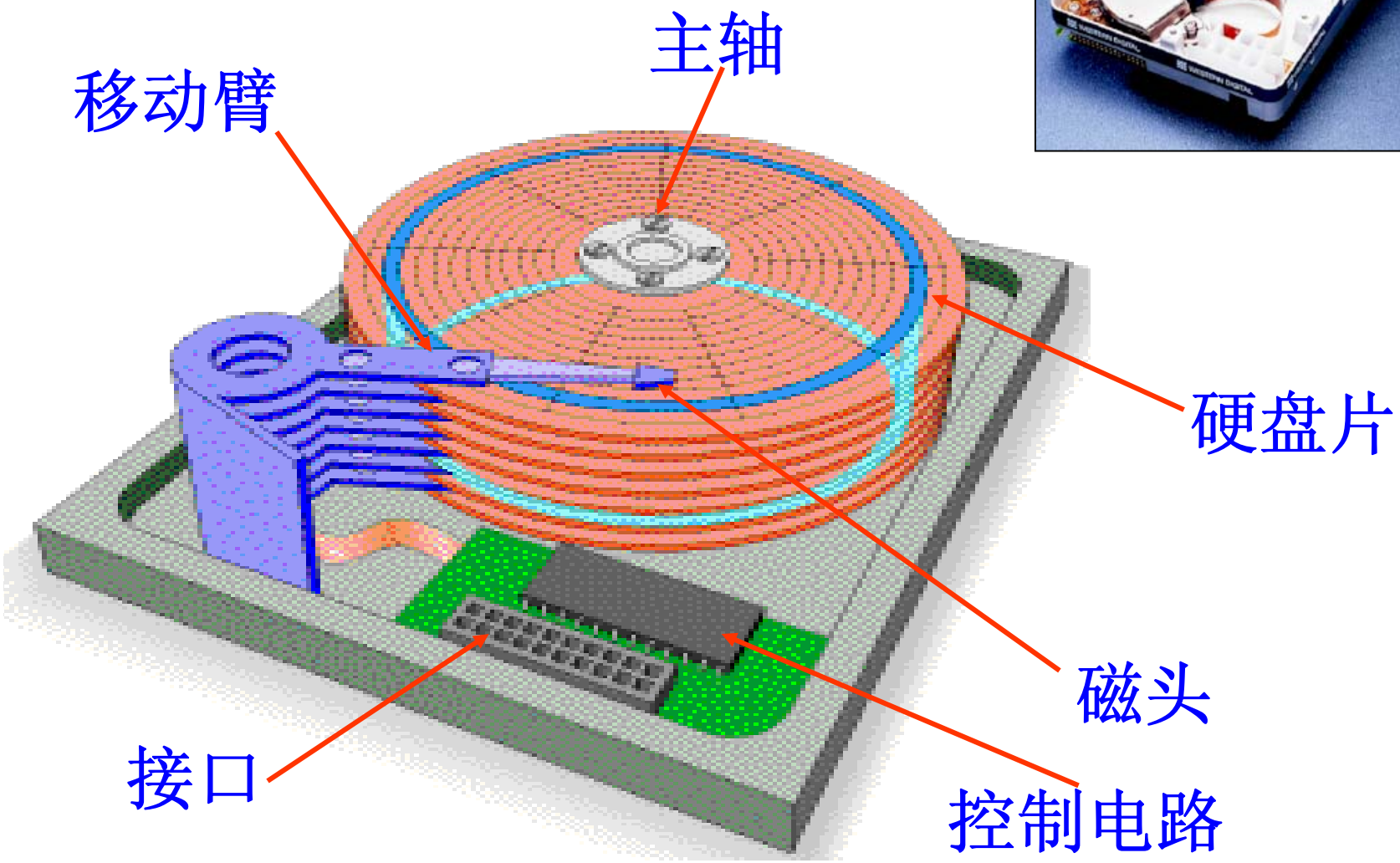
缺点：

- 盘片容量太小
- 单位存储容量的成本：软盘片 > 优盘 ！

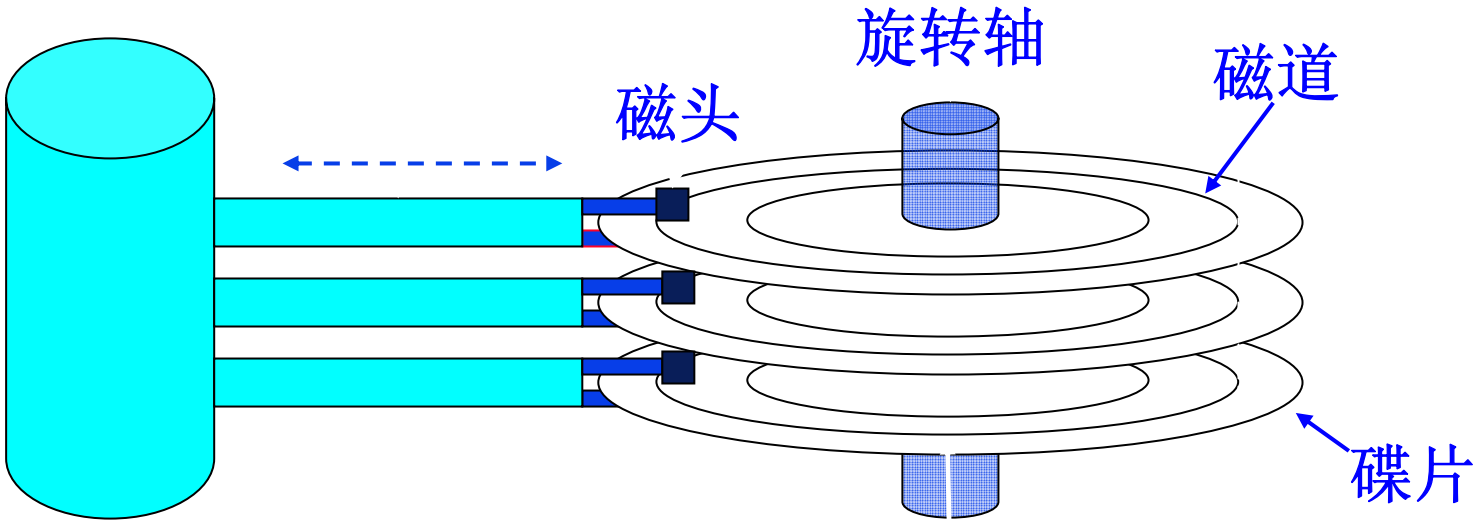
优盘：512MB～1GB
（100元）

回顾：硬盘存储器的结构





回顾：信息的平均存取时间



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）→

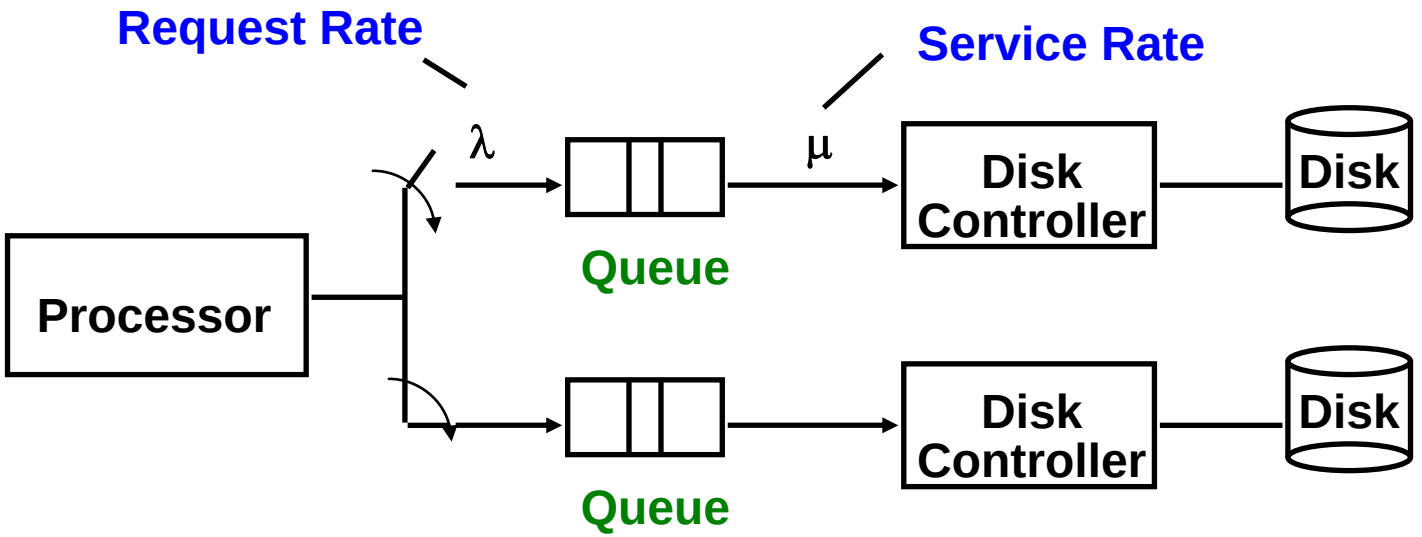
被选中的磁头等待扇区到达磁头下方（由扇区号控制）→ 读写该扇区中的数据

◦ 磁盘上的信息以扇区为单位进行读写，平均存取时间为：

T = 寻道时间 + 旋转等待时间 + 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)

磁盘的响应时间



- 磁盘响应读写请求的过程

响应时间(Disk Response Time)

= 排队Queueing Delay+控制器Controller Time + 存取时间 Access

存取时间 Access

= 寻道时间 Seek time+旋转等待时间 Rotational Latency+传输Transfer time

磁盘响应时间计算举例

- **512 byte sector, rotate at 5400 RPM, advertised seeks is 12 ms, transfer rate is 4 MB/sec, controller overhead is 1 ms, queue idle so no service time**

Disk Response Time= Seek time + Rotational Latency + Transfer time + Controller Time + Queuing Delay

= 12 ms + 0.5 / 5400 RPM + 0.5 KB / 4 MB/s + 1 ms + 0

= 12 ms + 0.5 / 90 RPS + 0.125 / 1024 s + 1 ms + 0

= 12 ms + 5.5 ms + 0.1 ms + 1 ms + 0 ms

= 18.6 ms

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=10.6ms　　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读磁盘在局部磁道！

回顾：硬盘存储器的性能指标

（1）容量：以GB为单位，目前硬盘单碟容量约为40～100GB

（2）平均存取时间：在几ms～几十ms之间，由硬盘的旋转速度、磁头寻道时间和数据传输速率所决定

（3）缓存容量：原则上越大越好，通常为2MB～8MB，该缓存的传输速度一般很高，达每秒几百MB

（4）数据传输速率

- 内部传输速率指硬盘在盘片上读写数据的速度，是扇区大小、旋转速度、记录密度的函数。转速越高，内部传输速率越快
- 外部传输速率指主机从（向）硬盘缓存读出（写入）数据的速度，与采用的接口类型有关，比内部传输速率高得多

（5）与主机的接口：

- 文件服务器使用：SCSI接口
- 前些年使用：并行ATA（IDE）接口
- 当前流行：　串行ATA（SATA）接口

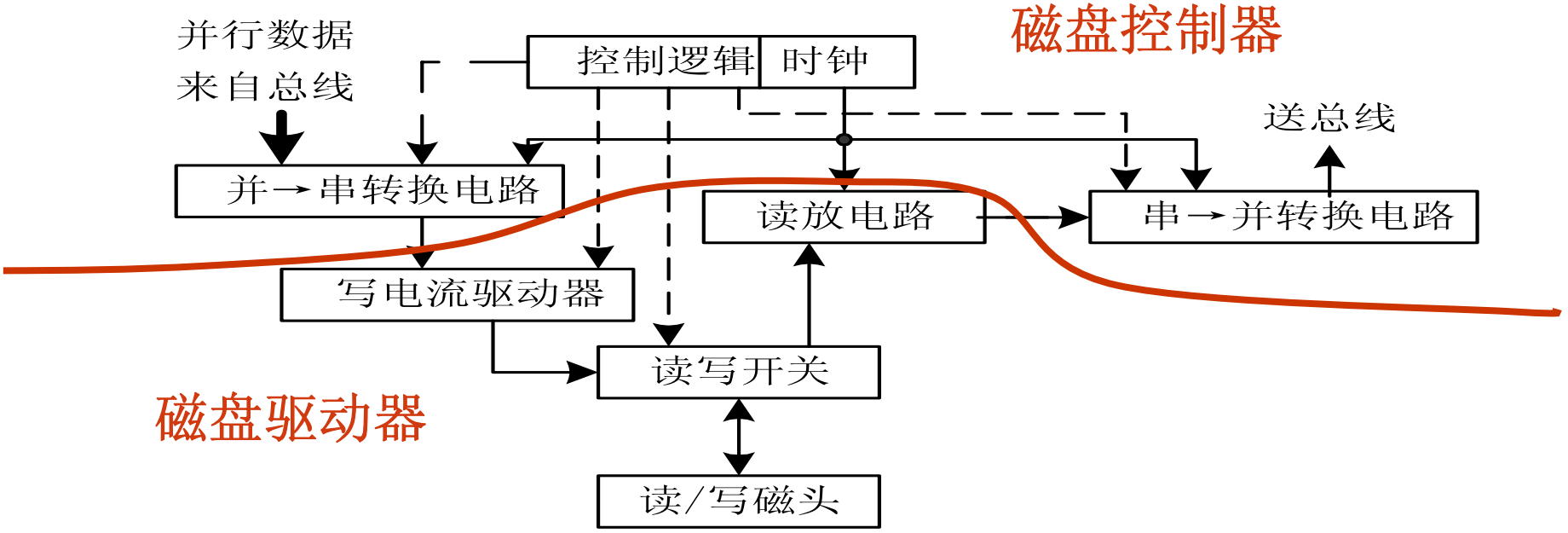
（6）可靠性和可用性

故障间平均时间MTBF = 平均故障时间MTTF + 平均修理时间MTTR

可靠性用MTTF度量，可用性=MTTF / MTBF

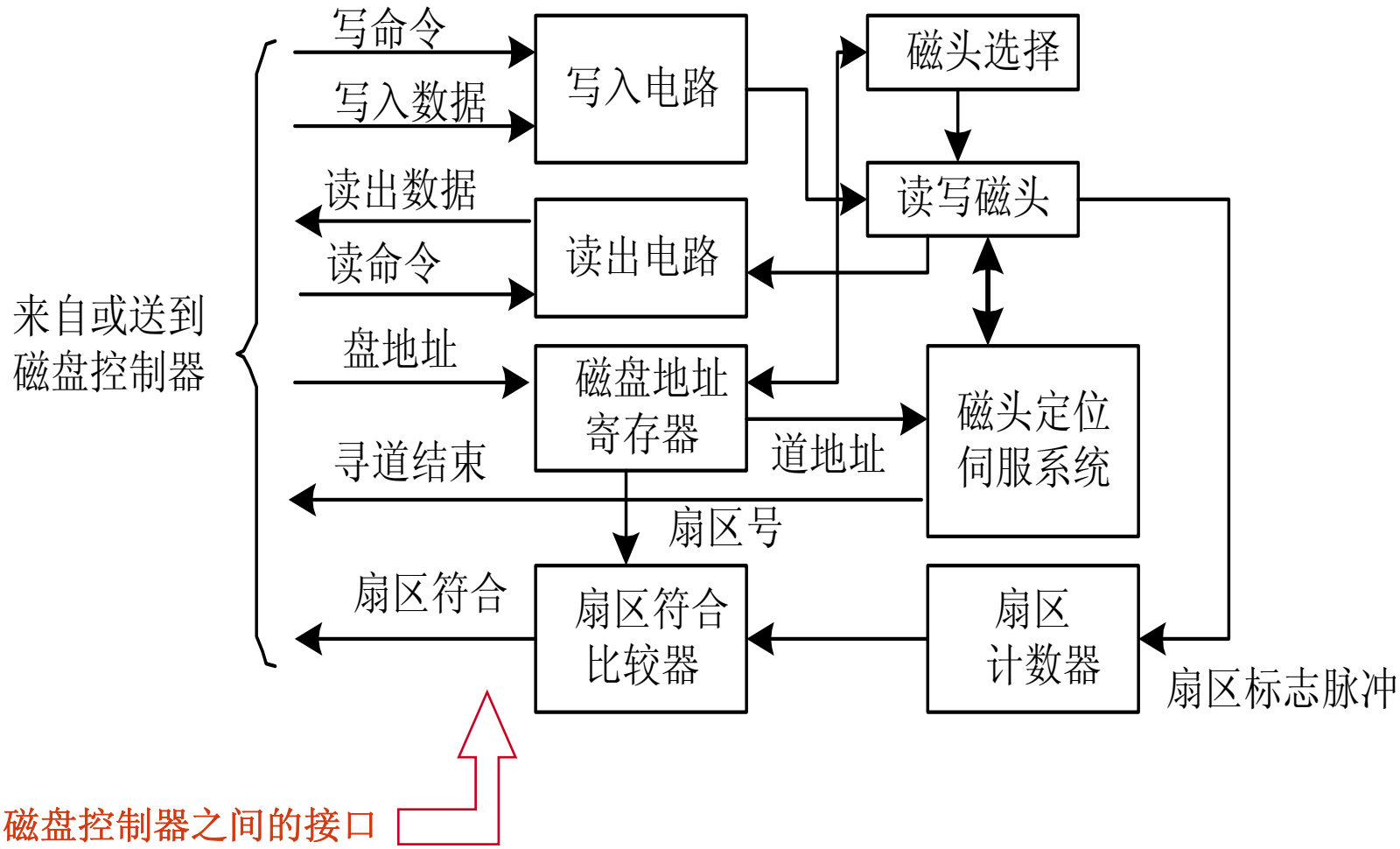
硬盘存储器的组成

- 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）



硬盘存储器的逻辑结构

硬盘驱动器的逻辑结构


写命令
写入数据
写入电路
磁头选择
读写磁头
读出数据
读命令
读出电路
来自或送到
磁盘控制器
盘地址
磁盘地址
寄存器
磁头定位
伺服系统
寻道结束
道地址
扇区号
扇区符合
扇区符合
比较器
扇区
计数器
扇区标志脉冲
与磁盘控制器之间的接口

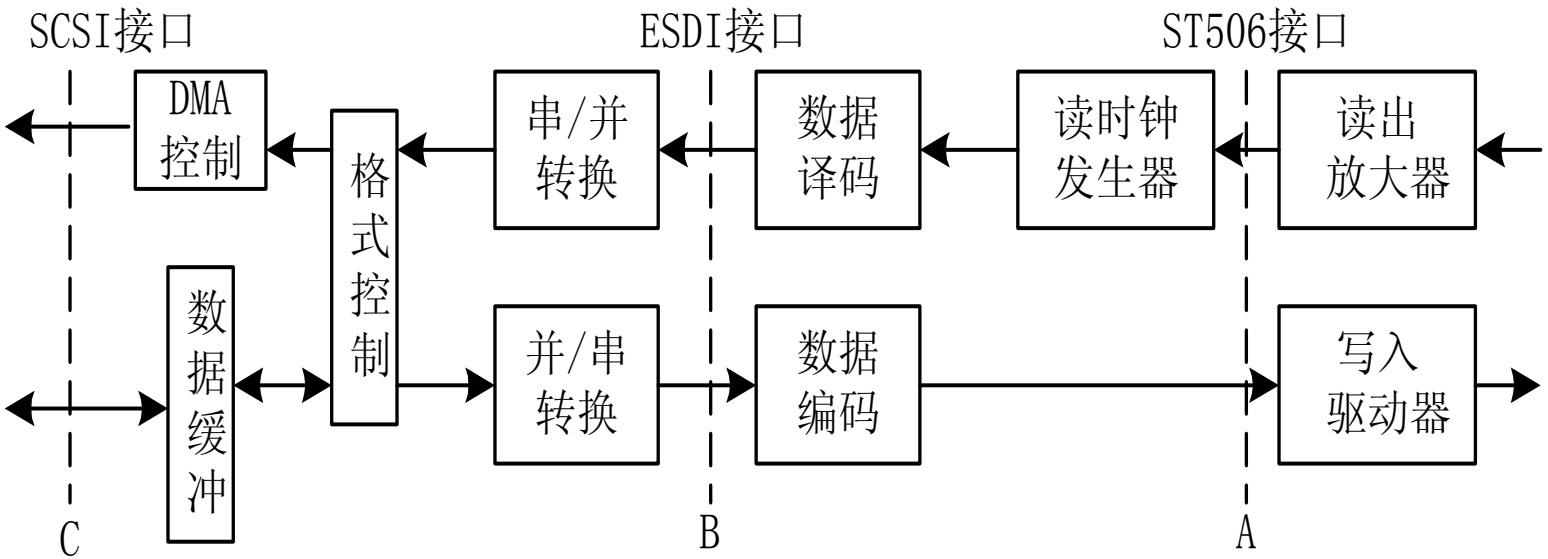
硬盘控制器的逻辑结构

- 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

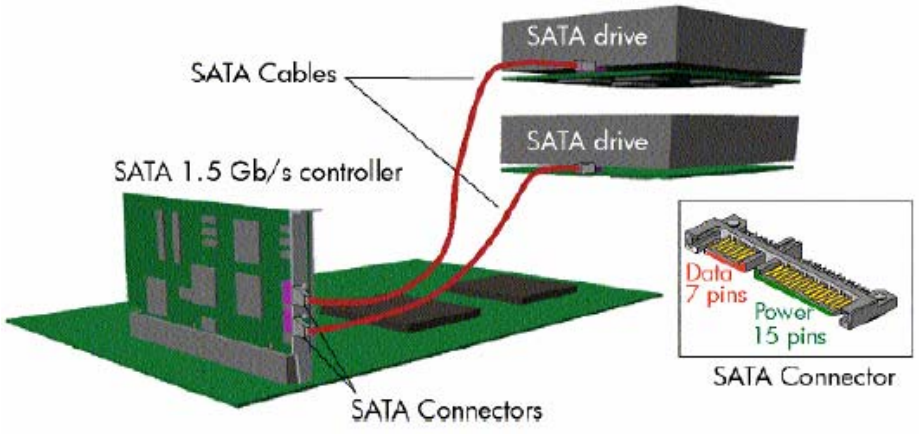
附：关于硬盘的SATA接口(1)

- PC前几年大多使用IDE(ATA)接口
  - Ultra ATA100或Ultra ATA133接口，传输速率最高分别为100MB/s和133MB/s
- 近两年开始大量采用SATA接口(150～300MB/s)
  - 串行传输方式，工作频率高(1.5GHz-3GHz)

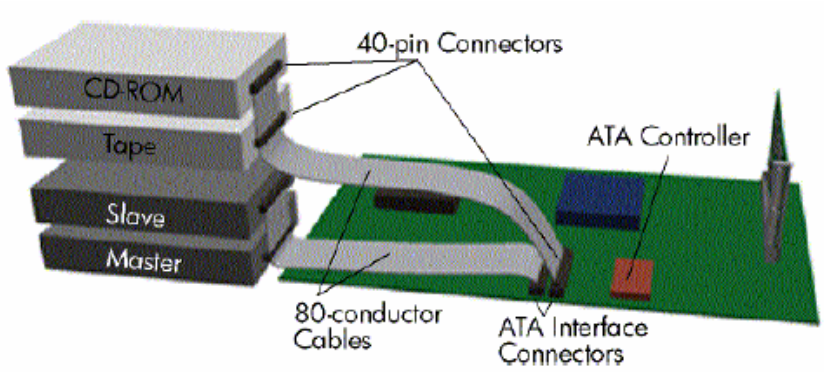
    (串行方式无需考虑位的同步，所以频率可以非常高！)
  - 低电平差分信号，内嵌时钟信号
  - 传输线长度增加，插头插座体积缩小

	Parallel ATA	SATA 1.5 Gb/s
Bandwidth	133 MB/s	150 MB/s
Volts	5V	250 mV
Number of pins	40	7
Cable length	18 in. (45.7 cm)	39 in. (1 m)

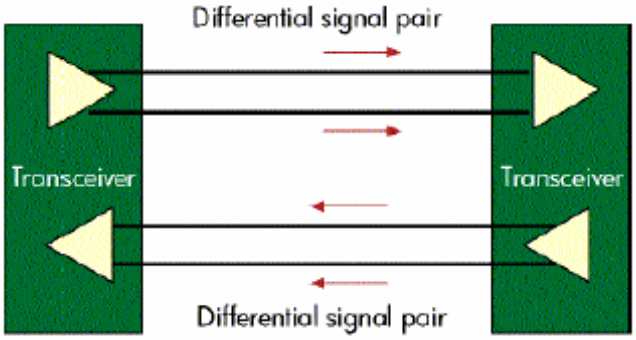
附：关于硬盘的SATA接口(2)



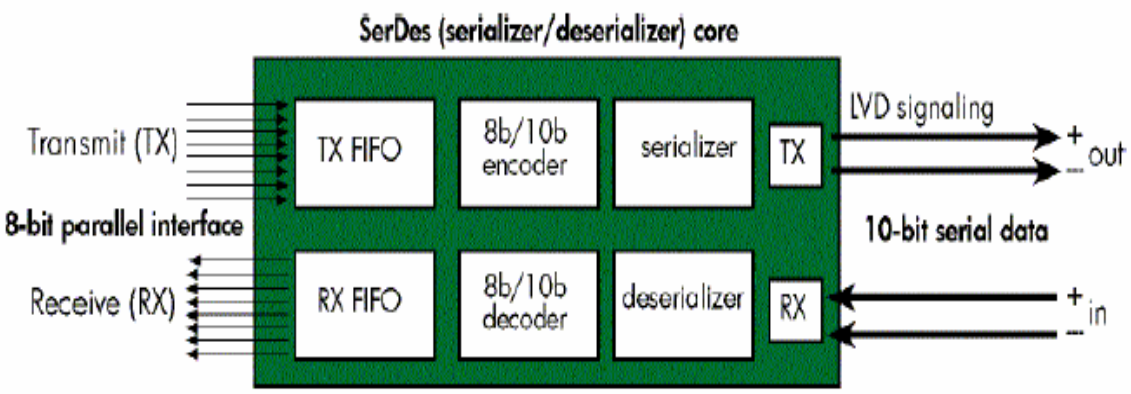
SATA接口的连接电缆



并行ATA接口的连接电缆



SATA接口采用差分
信号串行传输



SATA接口电路功能

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。

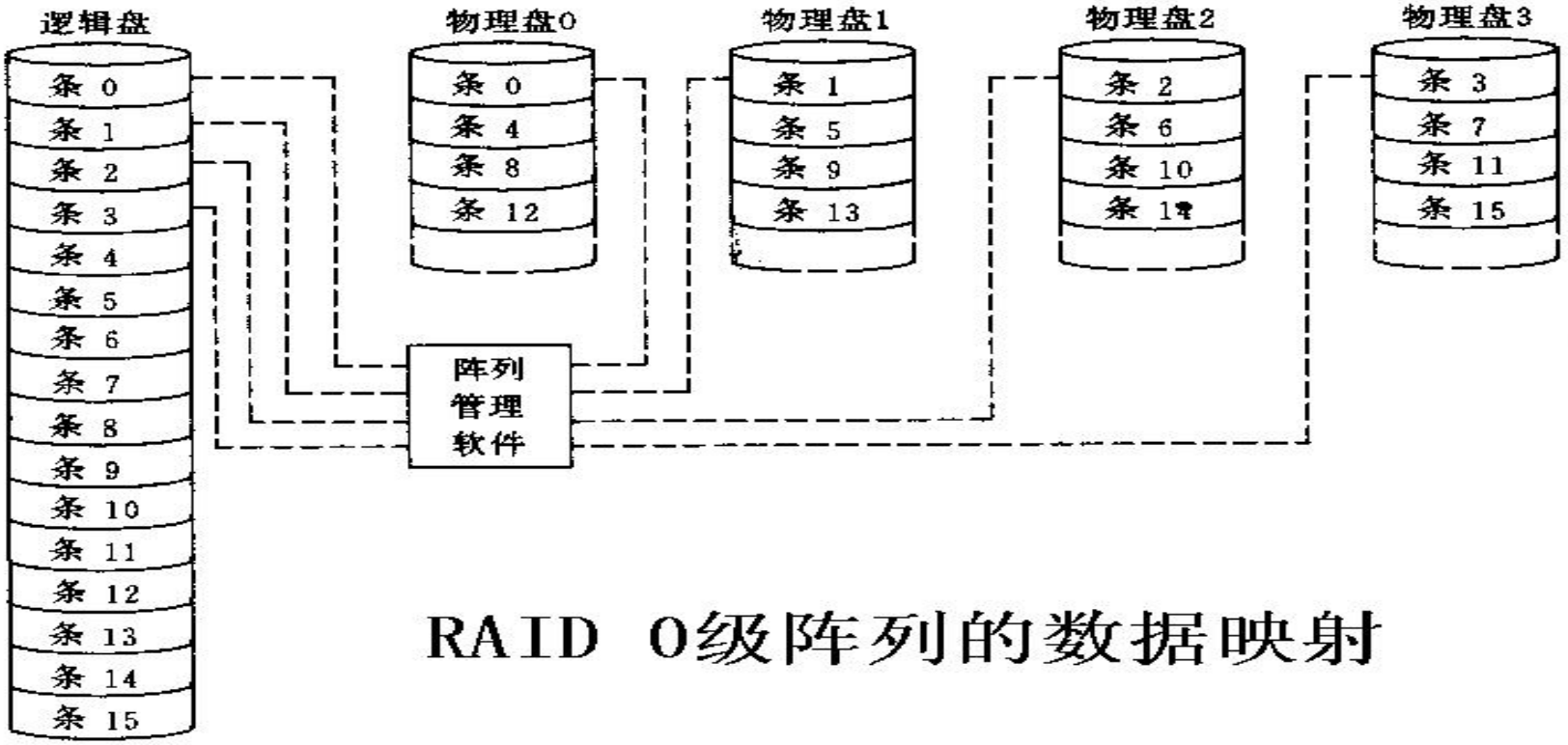
  （**3**）冗余磁盘容量用于存储奇偶校验信息，保证磁盘万一损坏时能恢复数据。
- **RAID**级别
  - 目前已知的**RAID**方案分为**8**级（**0-7**级），以及**RAID10**（结合**0**和**1**级）和**RAID30** （结合**0**和**3**级）和 **RAID50** （结合**0**和**5**级） 。但这些级别不是简单地表示层次关系，而是表示具有上述**3**个共同特性的不同设计结构。

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。能达到RAID0的两倍。

  （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。

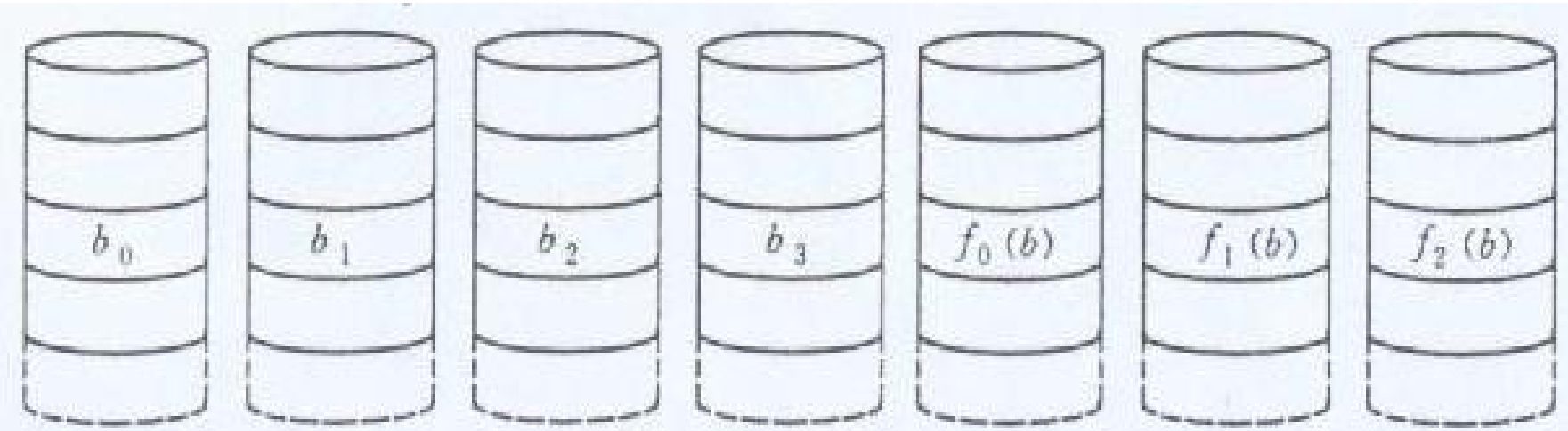
常用于可靠性要求很高的场合，如系统软件的存储，金融、证券等系统。

(b) RAID 1(镜像)

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。

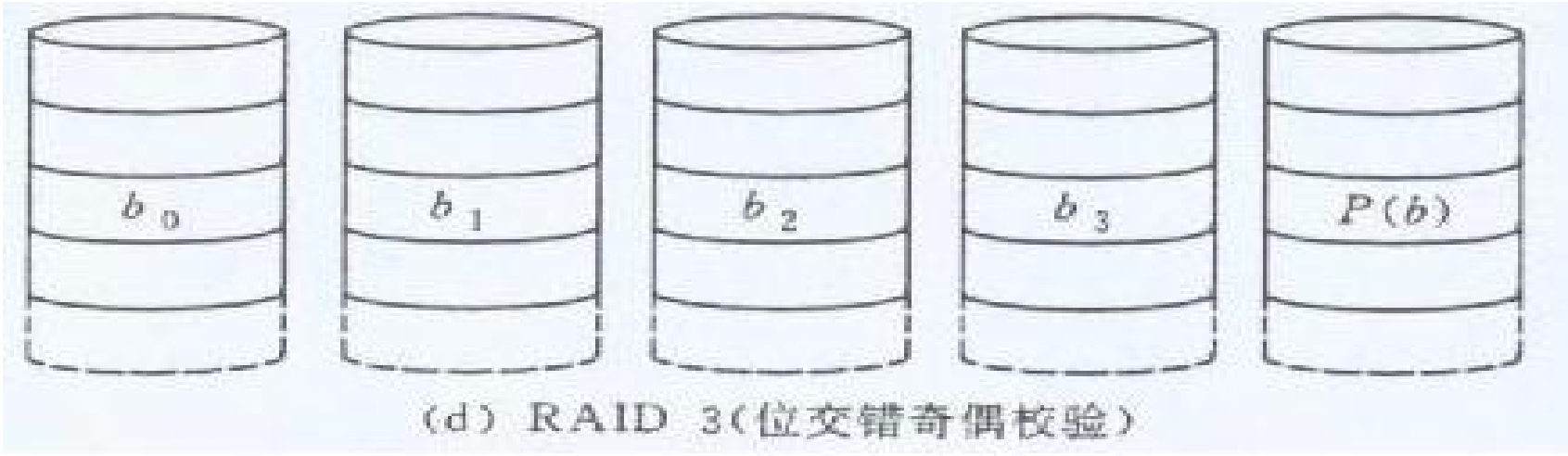
RAID2已不再使用！



(c) RAID 2(冗余用于海明码)

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**RAID3**、**4**、**5**级。



(d) RAID 3(位交错奇偶校验)

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响应速度块的场合。
- 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

(e) RAID 4(块级奇偶校验)

RAID 3/4/5级的数据恢复与RAID4级的少量写

RAID3/4/5的数据恢复操作

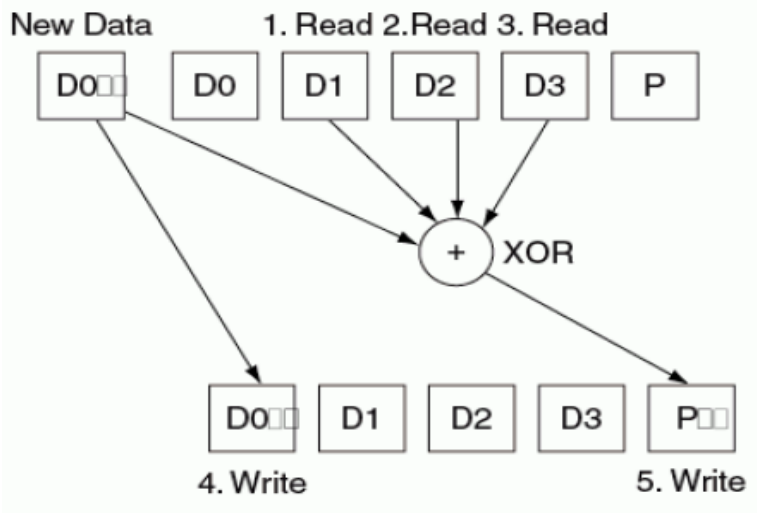
假定考虑一个有5个磁盘的阵列，X_0到X_3保存数据，p是奇偶校验盘，奇偶校验的第i位计算公式如下：

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

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：

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

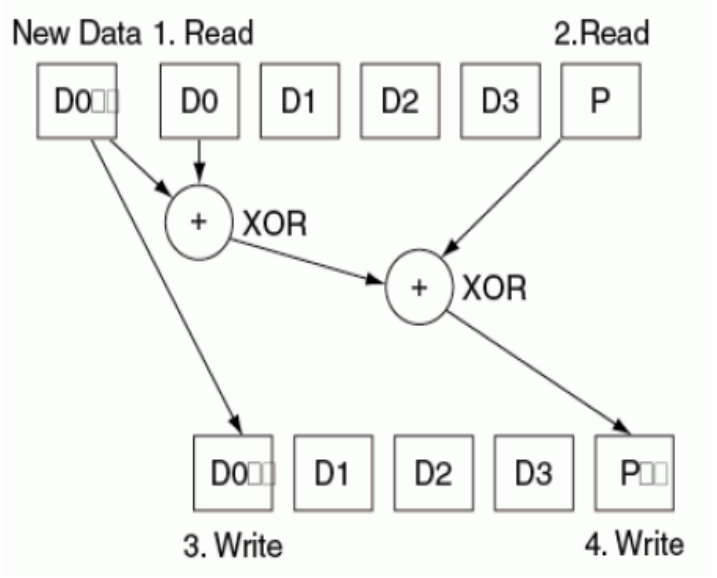
写操作后，可能改变的位数表示如下：

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

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

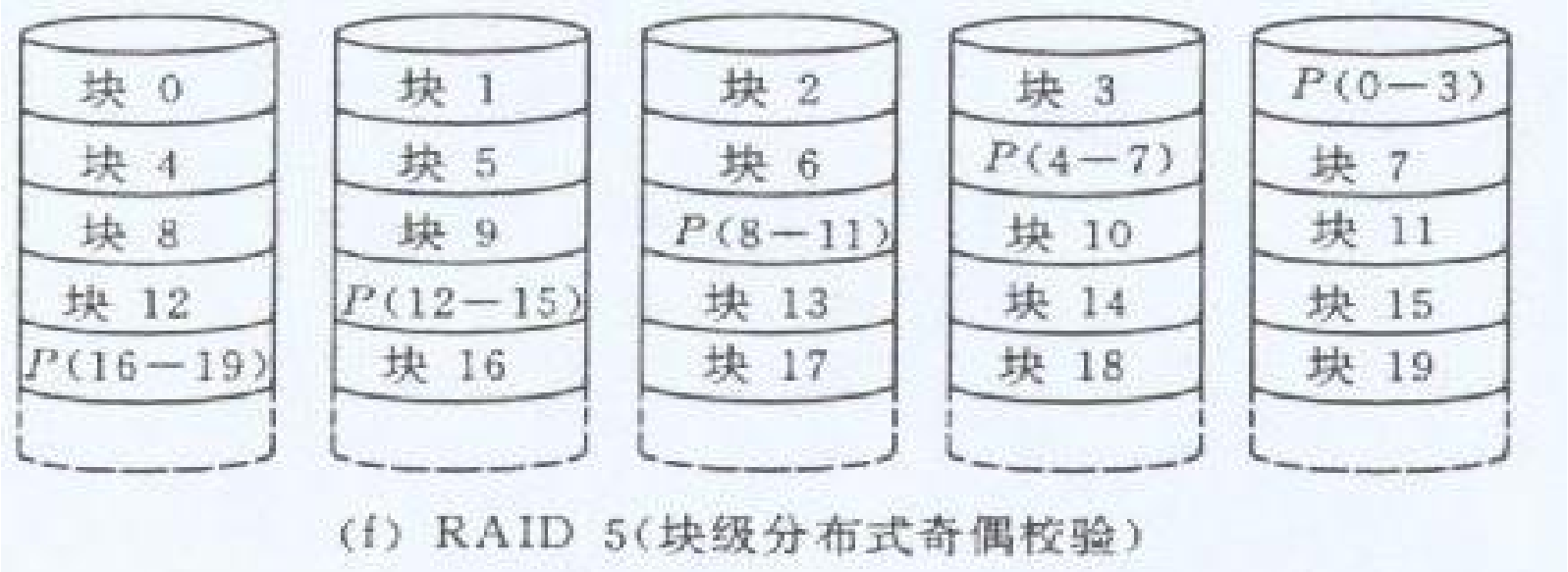
化简后得到：$p'(i)= p(i) \oplus X_0(i) \oplus X'_0(i)$

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

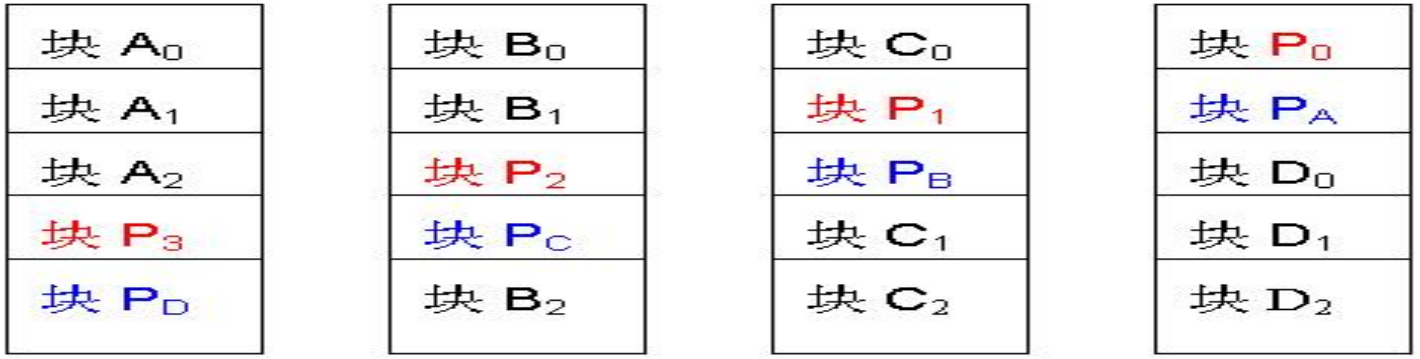


(f) RAID 5(块级分布式奇偶校验)

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：计算机网络的组成

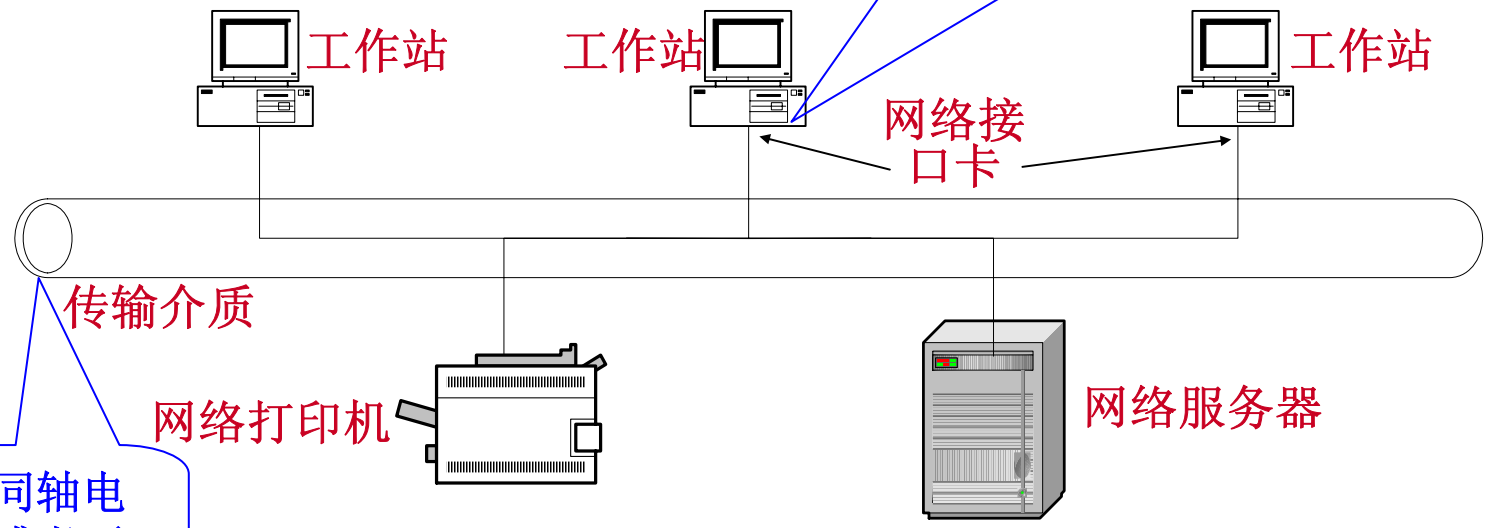
- 数据传输介质：用于传输数据，如双绞线、光缆、无线电波，··
- 通信控制设备：确保通信正确、可靠、有效进行的控制设备。如：网卡、集线器、交换器、调制解调器、路由器，…
- 通信协议——必须共同遵循的一组的规则和约定。
  - 通信如何开始、如何结束？数据如何表示？命令如何表示？通信对象如何区分？其身份如何鉴别？发生错误如何处理？等等。例如： **TCP/IP, NetBEUI, IPX/SPX**
- 网络操作系统：实现通信协议、管理网络资源、运行网络应用程序等
- 计算机等“智能”**(smart)**设备

回顾：计算机网络的分类

- 按使用的传输介质可分为：
  - 有线网 ■ 无线网
- 按网络的使用性质可分为：
  - 公用网 ■ 专用网 ■ 虚拟专网(VPN)
- 按网络的使用对象可以分为：
  - 企业网 ■ 政府网 ■ 金融网 ■ 校园网 …
- 按网络所覆盖的地域范围分为：
  - 局域网(LAN)：使用专用通信线路把地域范围较小（一幢楼房、一个楼群、一个单位或一个小区）的计算机连接而成的网络
  - 广域网(WAN)：把相距遥远的许多局域网和计算机用户互相连接在一起的网络。广域网有时也称为远程网
  - 城域网或市域网(MAN)：作用范围在广域网和局域网之间，其作用距离约为5km～50 km，例如一个城市的范围

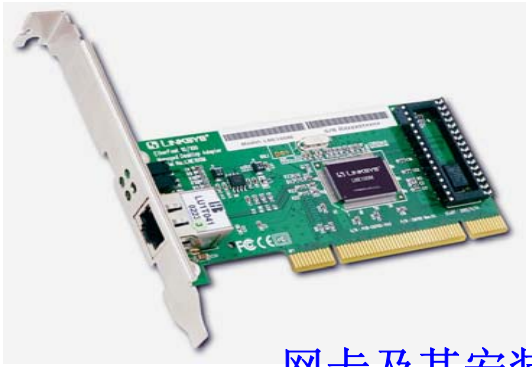
回顾：局域网的组成

网络上的每个设备都装有网络接口卡(NIC,简称网卡)，网卡通过传输介质把网络设备相互连接起来

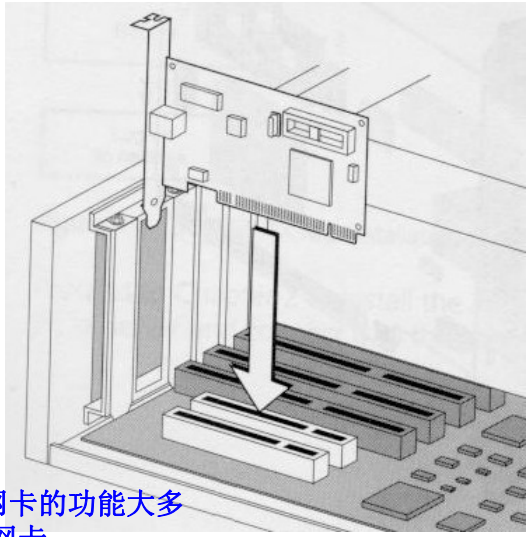


双绞线、同轴电缆、光纤或者无线电波

双绞线和RJ45插头(水晶头)



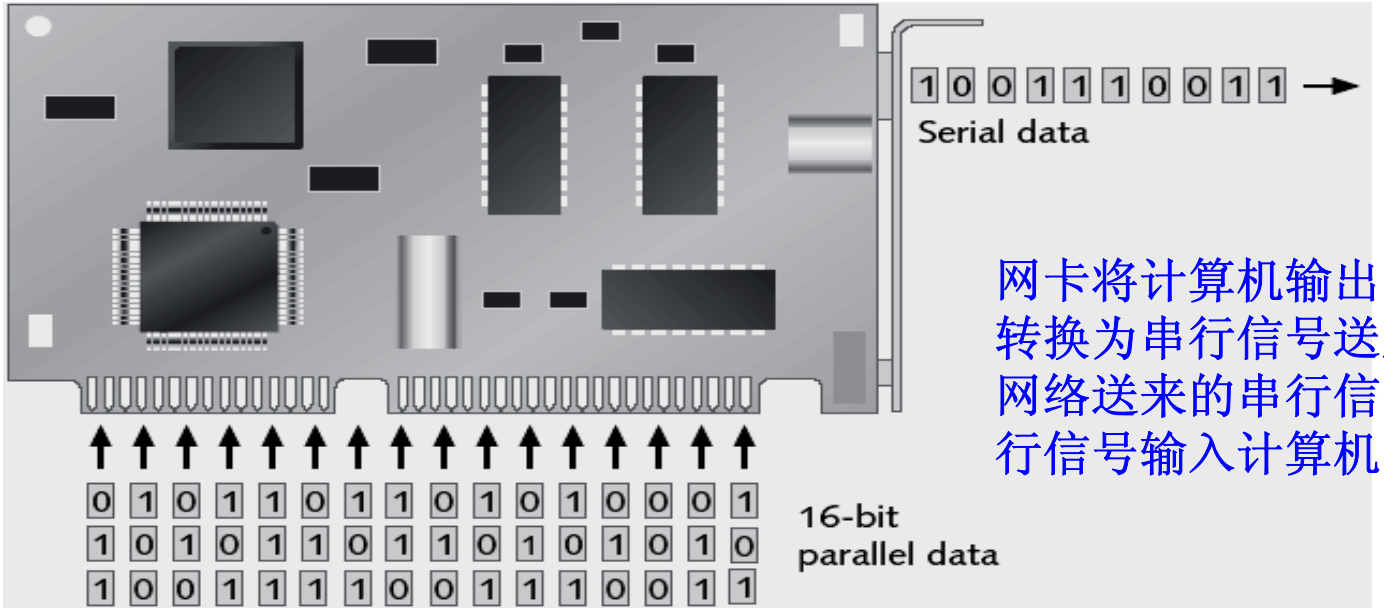
网卡及其安装



- 由于芯片组集成度的提高，现在网卡的功能大多已集成在芯片组中，即所谓的集成网卡

回顾：网卡的功能

- 在计算机与网络之间建立一个链路(link)，通过传输介质发送信息和接收信息（CPU将网卡视同为I/O设备）
- 将数据分成若干帧(frame)、以帧为单位发送和接收信息
- 将计算机的输出转换为适合网络传输的信号，将网络上送来的信号转换为计算机的输入(并－串转换、电平转换…)



网卡将计算机输出的并行信号转换为串行信号送入网络；将网络送来的串行信号转换为并行信号输入计算机

- 通过网卡的MAC地址(全球惟一)标识每个网络节点

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系统中的职责是：
    - 对共享设备及进行管理、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
  - 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
- 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

第一次作业

- 8.1
- 8.8
- 8.9
- 8.12
- 8.14
- 8.15
- 8.16

第二讲 I/O总线、I/O控制器、I/O接口

主 要 内 容

- 总线的基本概念
- 总线的位置和类型
- 总线的定时
  - 同步总线
  - 异步总线
- 总线的裁决
  - 集中裁决方式
  - 分布裁决方式
- 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

复习：一个典型程序的转换处理过程

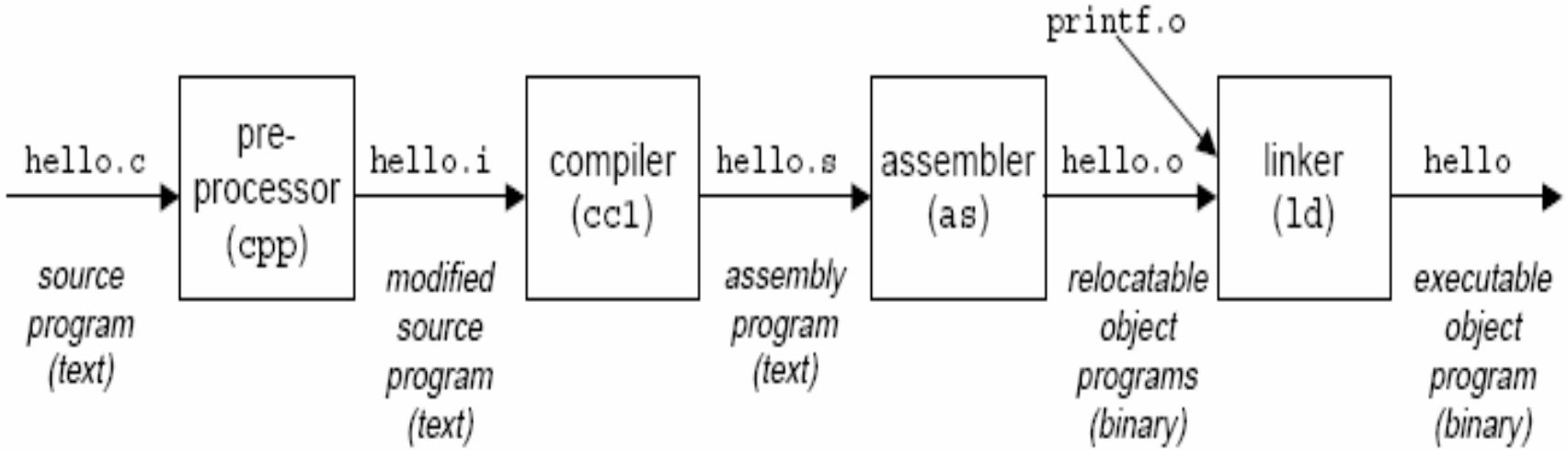
经典的“ hello.c ”C-源程序

```
1 #include <stdio.h>
2
3 int main()
4 {
5 printf("hello, world\n");
6 }
```

程序的功能是：
输出“hello,world”

hello.c的ASCII文本表示

```
# i n c l u d e <sp> < s t d i o .
35 105 110 99 108 117 100 101 32 60 115 116 100 105 111 46
h > \n \n i n t <sp> m a i n ( ) \n {
104 62 10 10 105 110 116 32 109 97 105 110 40 41 10 123
\n <sp> <sp> <sp> <sp> p r i n t f ( " h e l
10 32 32 32 32 112 114 105 110 116 102 40 34 104 101 108
l o , <sp> w o r l d \ n " ) ; \n }
108 111 44 32 119 111 114 108 100 92 110 34 41 59 10 125
```



复习：Hello程序的数据流动过程

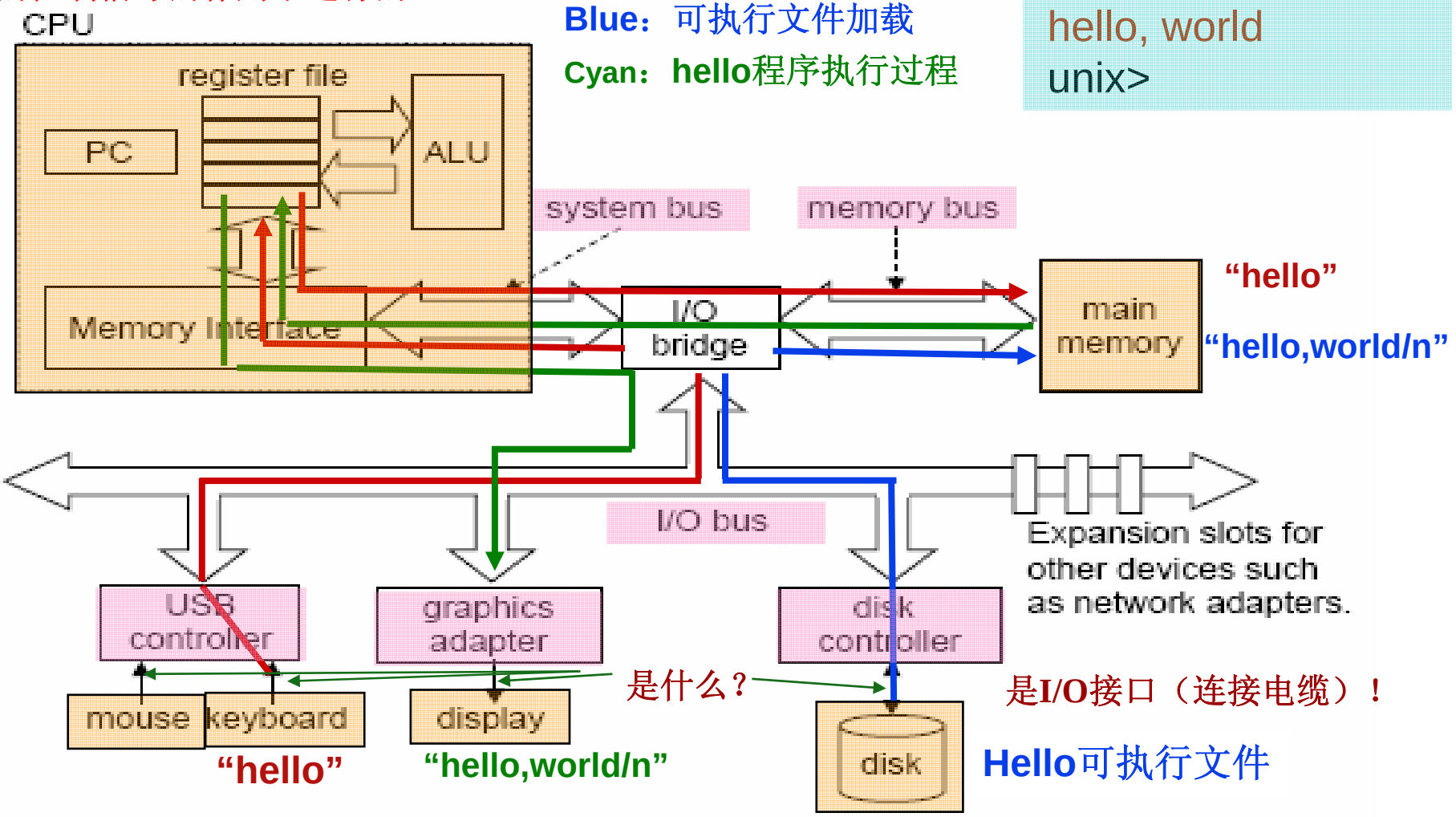
所有过程都是在CPU执行指令所产生的控制信号的作用下进行的。

Red： shell命令行处理

Blue： 可执行文件加载

Cyan： hello程序执行过程

```
unix> ./hello [Enter]
hello, world
unix>
```

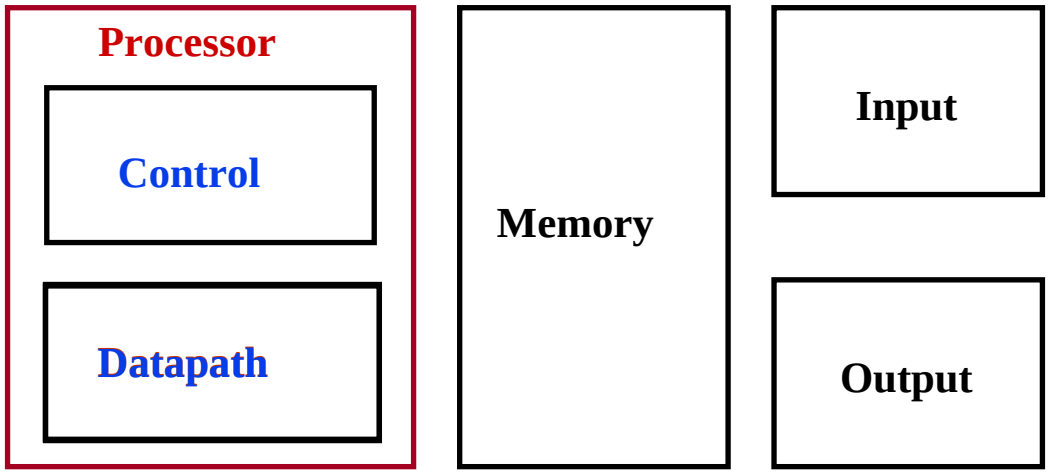


目前为止，已经学过：CPU、（Cache）、MM、I/O Device(disk/mouse/……)

本讲主要介绍总线（Bus）、I/O控制器、I/O接口，以及如何用它们来连接主机和外设

复习：计算机主要功能部件

计算机五大组成部分：



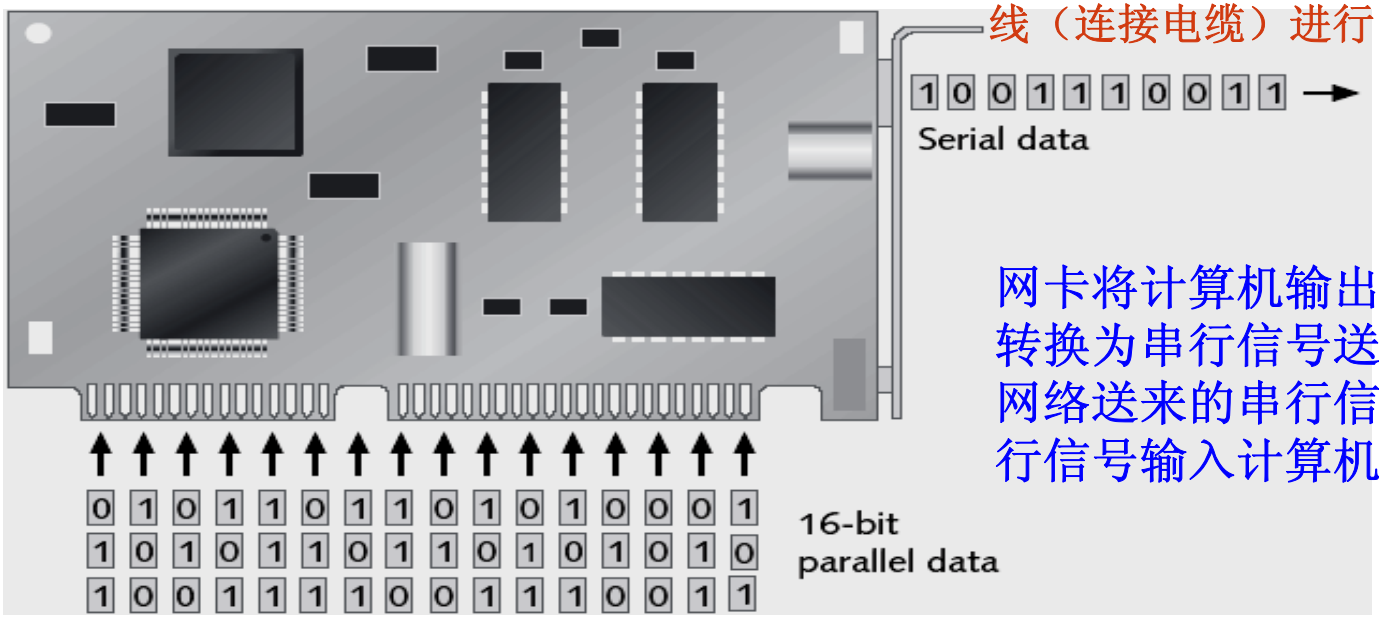
- 各主要部件（都学过）的功能和结构为：
  - **CPU**：执行指令，并处理异常（中断）
    - 和外界交换的信息有：
      - 指令和数据的地址、指令、数据、读/写命令、中断识别号、……
  - 存储器：存储指令和数据
    - 和外界交换的信息有：
      - 指令和数据的地址、指令、数据、读/写命令、……
  - **I/O**（I/O控制器和I/O设备）： 实现信息的输入输出
    - I/O控制器和主机侧交换的信息有：
      - 数据的地址、内（外）部数据、读/写命令、中断识别号……

SKIP

回顾：网卡的功能

- 在计算机与网络之间建立一个链路(link)，通过传输介质发送信息和接收信息（CPU将网卡视同为I/O设备）
- 将数据分成若干帧(frame)、以帧为单位发送和接收信息
- 将计算机的输出转换为适合网络传输的信号，将网络上送来的信号转换为计算机的输入(并－串转换、电平转换…)

网卡与网络之间的信息交换通过网线（连接电缆）进行，串行方式



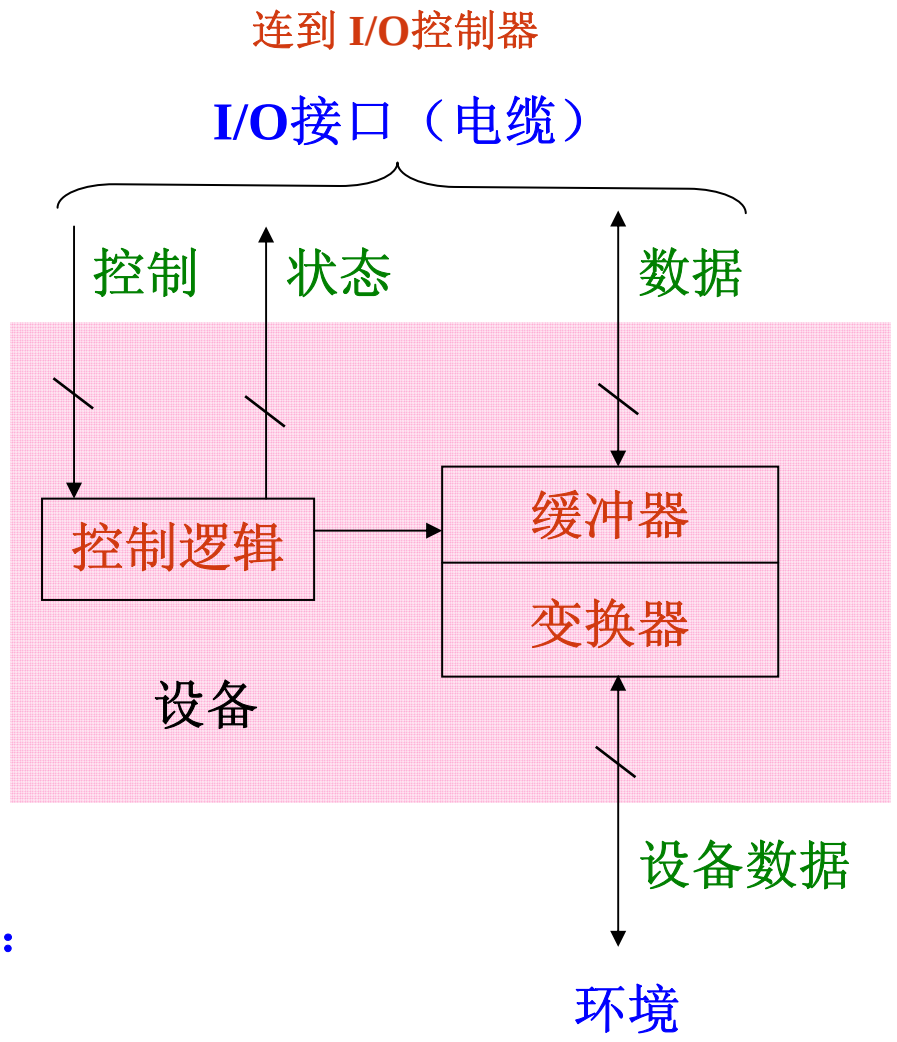
网卡将计算机输出的并行信号转换为串行信号送入网络；将网络送来的串行信号转换为并行信号输入计算机

网卡与主机侧的信息交换通过I/O总线（PCI插槽）进行，并行方式

所有I/O控制器都有主机侧和设备侧的连接

复习：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：

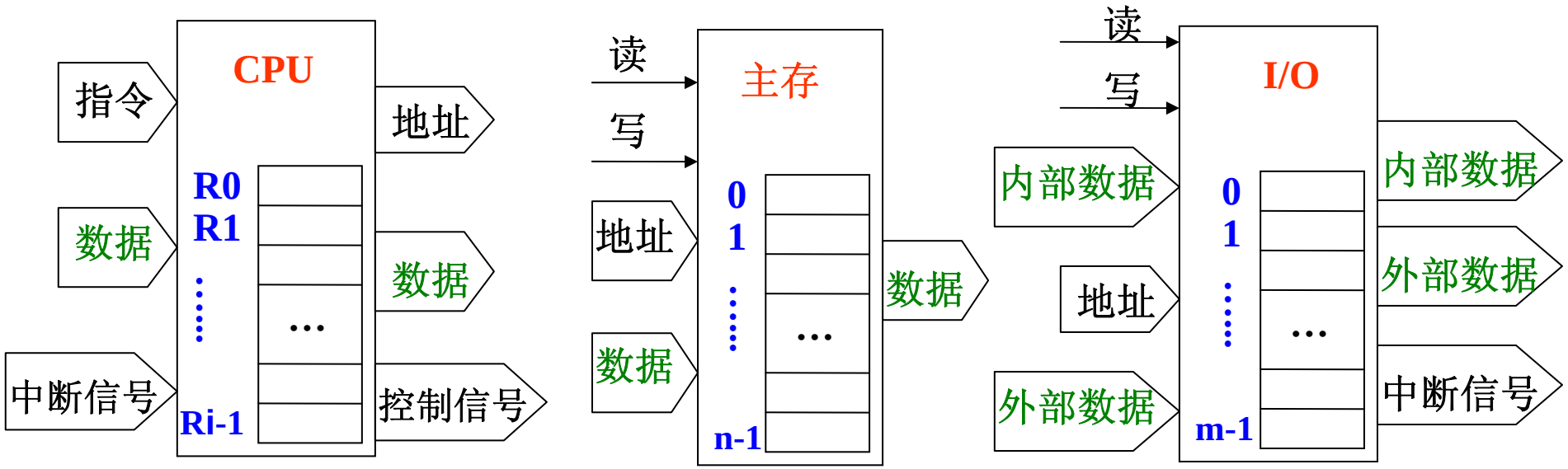
控制信号、状态信号、数据信号

也即: I/O控制器和设备侧交换的信号有控制、状态和数据三种

BACK

复习：计算机各部件的外部特性

- 计算机各主要部件的输入/出信息



•如何进行信息交换

–通过在部件之间架设通信线路

–我们把连接各部件的通路的集合称为互连结构

–互连结构主要是总线结构

总线是共享的传输介质，从来在部件之间传输信息

总线的分类

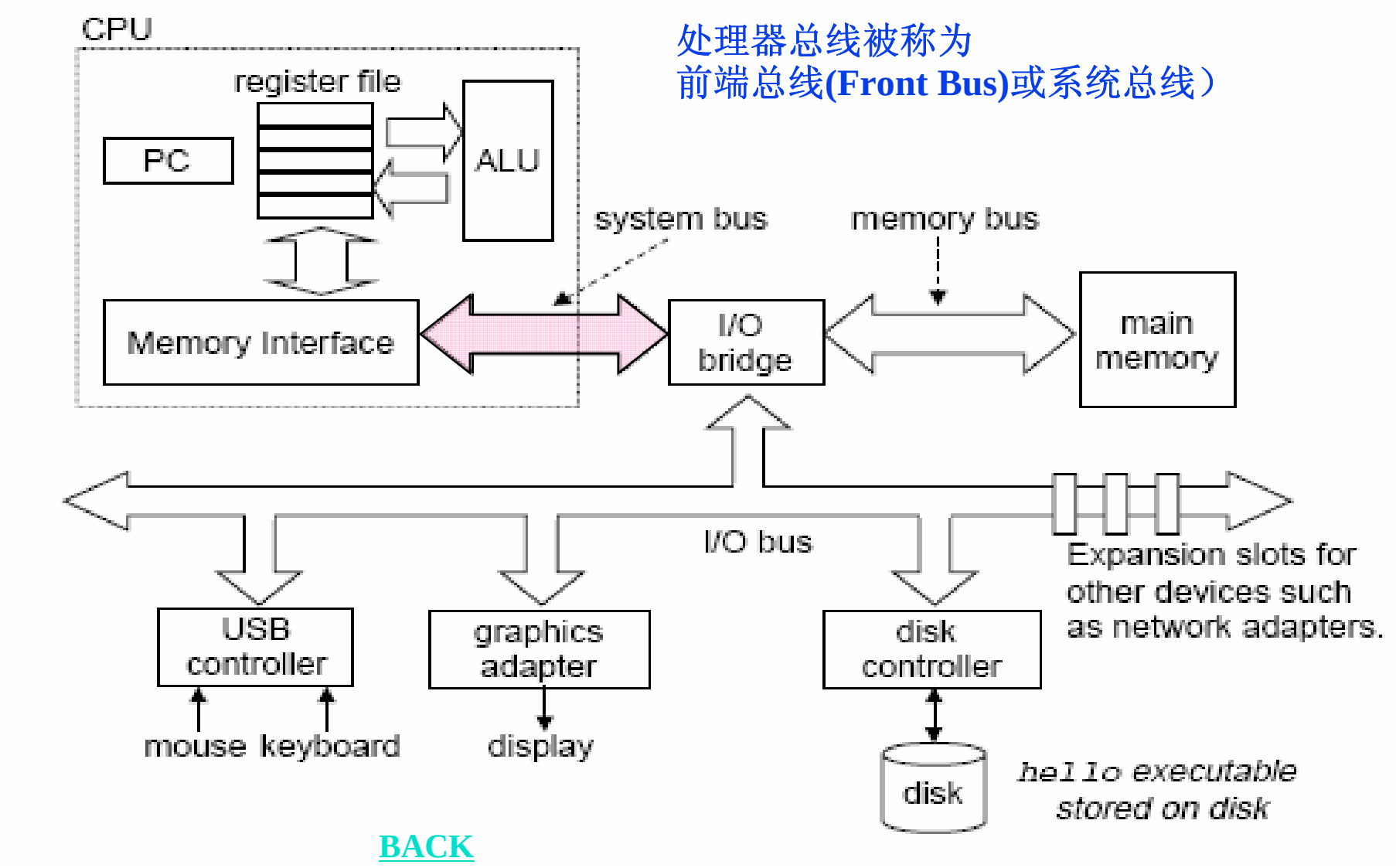
- 总线在各层次上提供部件之间的连接和交换信息通路
- 分为以下几类：
  - 芯片内总线：在芯片内部各元件之间提供连接
    - 例如，**CPU**芯片内部，各寄存器、**ALU**、指令部件等之间有总线相连
  - 系统总线：在系统主要功能部件（**CPU** 、**MM**和各种**I/O**控制器）间提供连接
    - 单总线结构：将**CPU**、**MM**和各种**I/O**适配卡通过底板总线**(Backplane Bus)**互连，底板总线为标准总线**(Industry standard)**
    - 多总线结构：将**CPU**、**Cache**、**MM**和各种**I/O**适配卡用局部总线、处理器-主存总线、高速**I/O**总线、扩充**I/O**总线等互连。主要有两大类：
      - **Processor- Memory Bus (Design specific or proprietary)**
        - » 短而快，仅需与内存匹配，使**CPU-MM**之间达最大带宽
      - **I/O Bus (Industry standard)**
        - » 长而慢，需适应多种设备，一侧连接到**Processor- Memory Bus** 或 **Backplane Bus**，另一侧连到**I/O**控制器

（注：**Intel**公司在推出**845**、**850**等芯片组时，对**“System Bus”**有专门的定义，将处理器总线称为前端总线**(Front Bus)**或系统总线）

SKIP

  - 通信总线：在主机和**I/O**设备之间或计算机系统之间提供连接

Intel 体系结构中特指的“系统总线”



处理器总线被称为
前端总线(Front Bus)或系统总线）

BACK

系统总线的组成

- 系统总线通常由一组控制线、一组数据线和一组地址线构成。也有些总线没有单独的地址线，地址信息通过数据线来传送，这种情况称为数据/地址复用。
  - 数据线（**Data Bus**）：承载在源和目部件之间传输的信息。数据线的宽度反映一次能传送的数据的位数。
  - 地址线（**Address Bus**）：给出源数据或目的数据所在的主存单元或I/O端口的地址。地址线的宽度反映最大的寻址空间。
  - 控制线（**Control Bus**）：控制对数据线和地址线的访问和使用。用来传输定时信号和命令信息。典型的控制信号包括：
    - 时钟（**Clock**）：用于总线同步。
    - 复位（**Reset**）：初始化所有设备。
    - 总线请求（**Bus Request**）：表明发出该请求信号的设备要使用总线。
    - 总线允许（**Bus Grant**）：表明接收到该允许信号的设备可以使用总线。
    - 中断请求（**Interrupt Request**）：表明某个中断正在请求。
    - 中断回答（**Interrupt Acknowledge**）：表明某个中断请求已被接受。
    - 存储器读（**memory read**）：从指定的主存单元中读数据到数据总线上。
    - 存储器写（**memory read**）：将数据总线上的数据写到指定的主存单元中。
    - **I/O**读（**I/O read**）：从指定的I/O端口中读数据到数据总线上。
    - **I/O**写（**I/O Write**）：将数据总线上的数据写到指定的I/O端口中。
    - 传输确认（**transmission Acknowledge**）：表示数据已被接收或已被送到总线

总线设计要素

- 总线设计要考虑的基本要素

尽管有许多不同的总线实现方式，但总线设计的基本要素和考察的性能指标一样

①信号线类型(Signal line type)：

专用(Separate) / 复用(Multiplexed)

②仲裁方法(Arbitrating)：

集中式(Center) / 分布式(distributed)

③定时方式(Timing)：

同步通信 (Synchronous) / 异步通信 (Asynchronous)

④事务类型(Bus Transaction)：

总线所支持的各种数据传输类型和其他总线操作类型，如：

存储器读、存储器写、I/O读、I/O写、读指令、中断响应等

⑤总线带宽(Bus Bandwidth)：

单位时间内在总线上传输的最大数据量（是一种传输能力）

相当于公路的最大载客量。例如，沪宁高速每车道最多每5分钟发一辆车，每辆车最多50人，共有6个车道，则最大流量为多少（?人/小时）？

最大载客量：6道x12车/小时x50人/车= 3600人/小时

信号线类型

总线的信号线类型有：专用、复用

- 专用信号线：
  - 信号线专用来传送某一种信息。

    例如，使用分立的数据线和地址线，使得数据信息专门由数据线传输，地址信息专门由地址线传输。
- 复用信号线：
  - 信号线在不同的时间传输不同的信息。

    例如，许多总线采用数据/地址线分时复用方式，用一组数据线在总线事务的地址阶段传送地址信息，在数据阶段传送数据信息。这样就使得地址和数据通过同一组数据线进行传输。
- 信号分时复用的优缺点：
  - 优：减少总线条数，缩小体积、降低成本。
  - 缺：总线模块的电路变复杂，且不能并行。

总线裁决（总线控制/使用/访问权的获得）

总线被多个设备共享，但每一时刻只能有一对设备使用总线传输信息。

- 什么是总线裁决？

  当多个设备需要使用总线进行通信时，采用某种策略选择一个设备使用总线

- 为什么要进行总线裁决 ？

  总线被连接在其上的所有设备共享，如果没有任何控制，那么当多个设备需要进行通信时，每个设备都试图为各自的传输将信号送到总线上，这样就会产生混乱。所以必须进行总线裁决

- 如何避免上述混乱？

  - 在总线中引入一个或多个总线主控设备，只能主控设备控制总线
    - 主控设备：能发起总线请求并控制总线。（如：处理器）
    - 从设备：只能响应从主控设备发来的总线命令。（如：主存）
  - 利用总线裁决决定哪个总线主控设备将在下次得到总线使用权

总线裁决（总线控制/使用/访问权的获得）

①总线裁决信号： 总线请求线和总线许可线

信号线专用 / 信号线复用

如：数据线和总线请求线复用时，总线裁决和数据传输不能同时进行

②总线裁决有两种方式：集中式和分布式

集中式：将控制逻辑做在一个专门的总线控制器或总线裁决器中，通过将所有的总线请求集中起来利用一个特定的裁决算法进行裁决

菊花链（**Daisy chain**）

集中并行（**Centralized, Parallel**）

分布式：没有专门的总线控制器，其控制逻辑分散在各个部件或设备中

自举式（**Self-selection**）

冲突检测（**Collision detection**）

③裁决方案应在以下两个因素间进行平衡

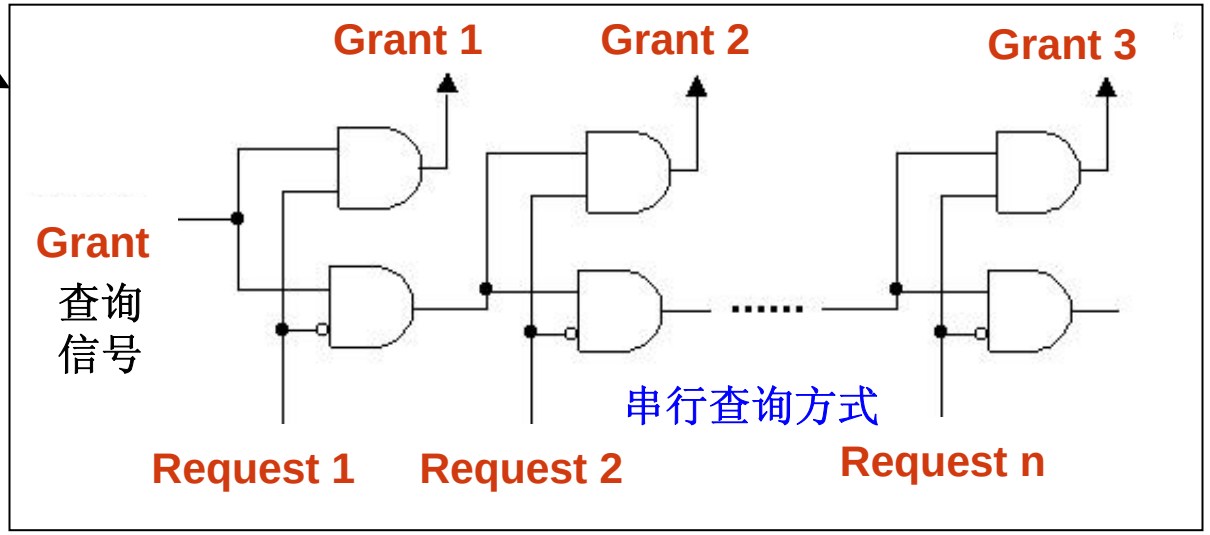
等级性**(Priority)**—具有高优先级的设备应该先被服务

公平性**(Fairness)**—即使具有最低优先权的设备也不能永远得不到总线使用权

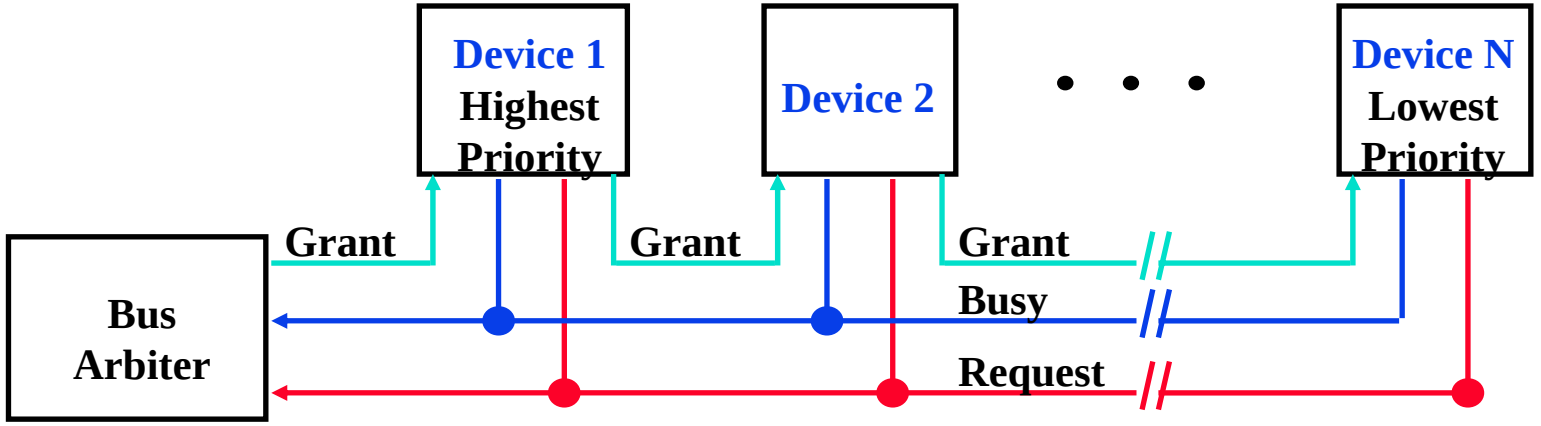
SKIP

（自学）菊花链总线裁决

菊花链查询电路



Grant从最高优先权的设备依次向最低优先权的设备串行相连。如果到达的设备有总线请求，则**Grant**信号就不再往下传，该设备建立总线忙**Busy**信号，表示它已获得了总线使用权。



BACK

Advantage：

① 简单**(simple)**，只需几根线就能按一定优先次序实现总线裁决。 ② 易扩充设备**(flexible)**

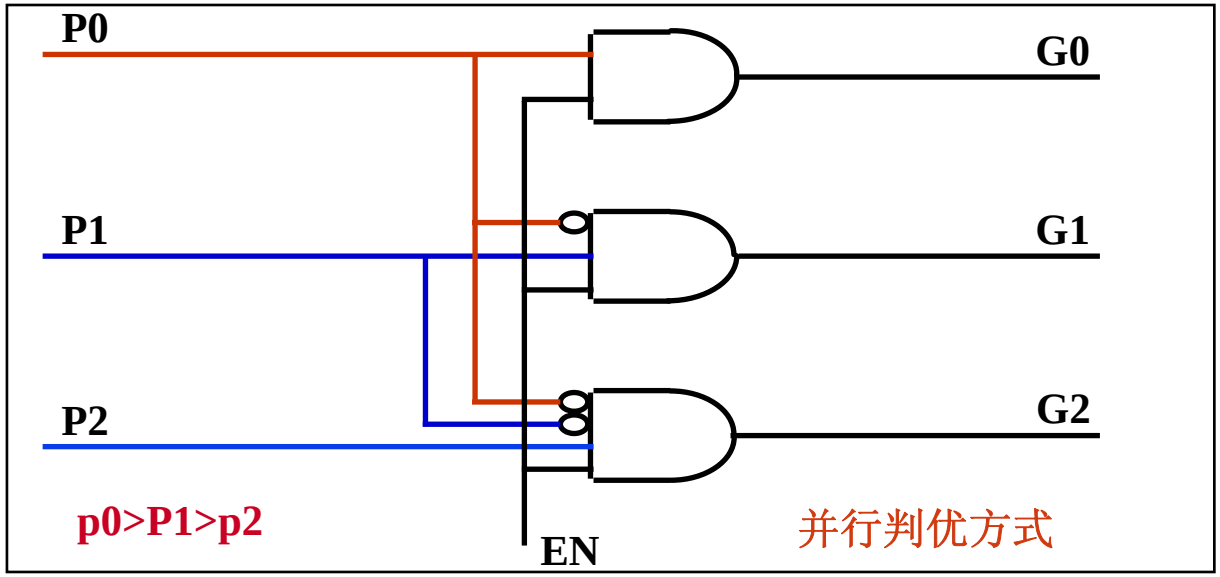
Disadvantages：

① 不能保证公正性 ② 对电路故障敏感
③ 菊花链的使用限制了总线速度

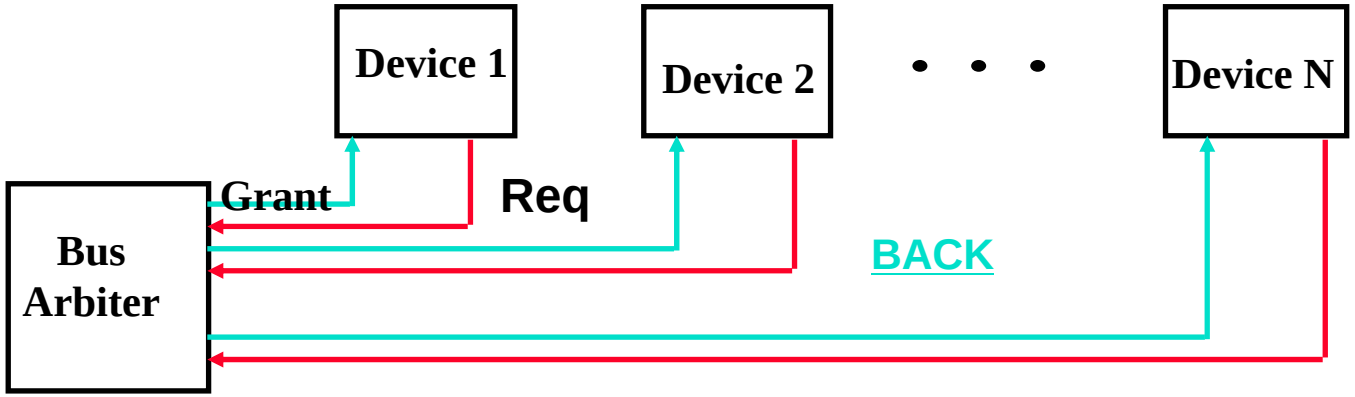
（自学）独立请求方式裁决

并行判优电路

p0、P1、p2优先级怎样？



- 各设备都有一对总线请求线**Req**和总线允许线**Grant**。
- 当某设备要使用总线时，就通过对应的总线请求线将请求信号送到总线控制器。
- 总线控制器中有一个判优电路，可根据各设备的优先级确定选择哪个设备。控制器可给各请求线以固定的优先级，也可编程设置



问题：如果有N个设备，则菊花链和独立请求各需多少裁决线？

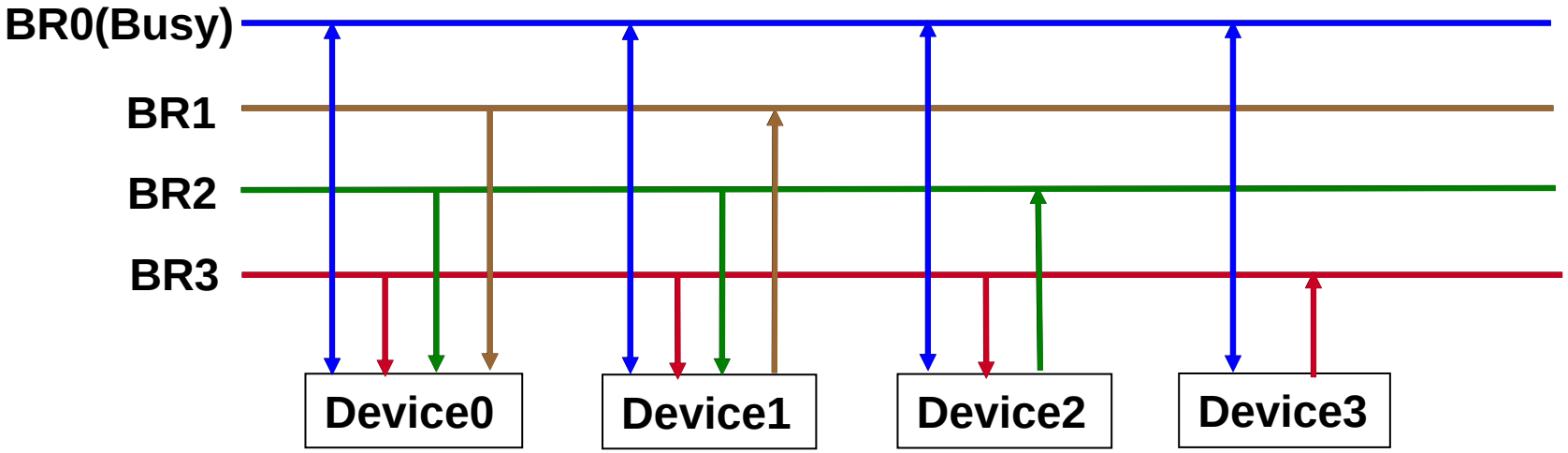
2 ~ 2N

优点：① 响应速度快。 ② 可编程时，则优先级灵活

缺点：① 控制逻辑复杂，控制线数量多。

裁决算法：总线控制器可采用固定的并行判优算法、平等的循环菊花链算法、动态优先级算法（如：最近最少用算法、先来先服务算法）等。

（自学）自举分布式裁决



- 优先级固定，各设备独立决定自己是否是最高优先级请求者
- 需请求总线的设备在各自对应的总线请求线上送出请求信号
- 在总线裁决期间每个设备将比自己优先级高的请求线上的信号取回分析：
  - 若有总线请求信号，则本设备不能立即使用总线
  - 若没有，则可以立即使用总线，并通过总线忙信号阻止其他设备使用总线
    - 最低优先级设备可以不需要总线请求线，为什么？
    - 需要较多连线用于请求信号，所以，许多总线用数据线**DB**作为总线请求线

      N个设备要多少请求信号？　**N**条！
    - **NuBus**（**MacintoshII** 中的底板式总线）、**SCSI**总线等采用该方案

上图中的优先级 **(**优先级**)**是什么？

设备3>设备2>设备1>设备0

BACK

（自学）冲突检测方式裁决

基本思想：

当某个设备要使用总线时，它首先检查一下是否有其他设备正在使用总线

如果没有，那它就置总线忙，然后使用总线；

若两个设备同时检测到总线空闲，则可能会同时使用总线，此时发生冲突；

一个设备在传输过程中，它会帧听总线以检测是否发生了冲突；

当冲突发生时，两个设备都会停止传输，延迟一个随机时间后再重新使用总线

- 该方案一般用在网络通信总线上，如：**Ethernet**总线等。

BACK

总线定时方式

- 什么是总线的定时

  通过总线裁决确定了哪个设备可以使用总线，那么一个取得了总线控制权的设备如何控制总线进行总线操作呢？也即如何来定义总线事务中的每一步何时开始、何时结束呢？这就是总线通信的定时问题。

- 总线通信的定时方式
  - Synchronous (同步)：用时钟来同步定时
  - Asynchronous(异步)：用握手信号定时
  - Semi-Synchronous（半同步）：同步(时钟)和异步(握手信号)结合
  - Split transaction（拆分事务）：在从设备准备数据时，释放总线

CPU-处理器总线都采用同步方式

异步方式只有I/O总线才会使用

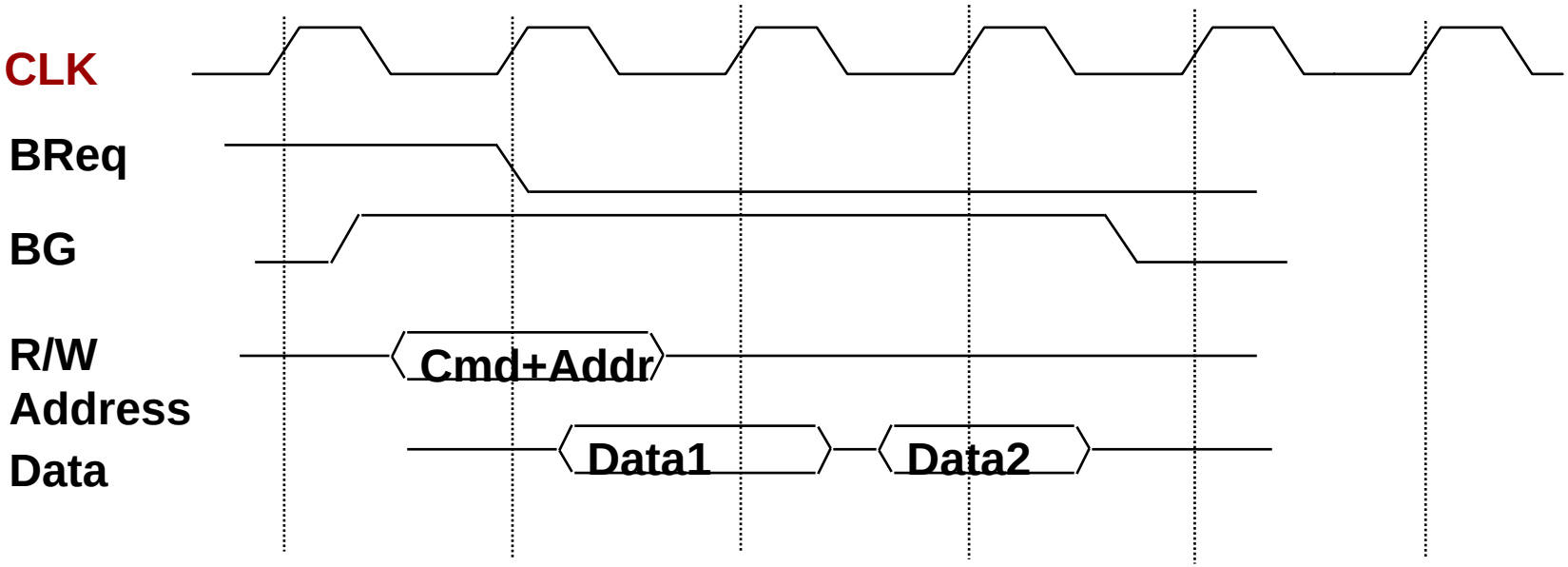
I/O总线大多采用半同步方式

拆分事务方式可以提高总线的有效带宽

SKIP

同步总线 (Synchronous Bus)

简单的同步协议如下图：　一个总线事务：地址阶段 + 数据阶段 + … + 数据阶段



控制线上有一个时钟信号进行定时，有确定的通信协议

BACK

Advantage(优点):　控制逻辑少而速度快

Disadvantages(缺点):

（1）所有设备在同一个时钟速率下运行，故以最慢速设备为准

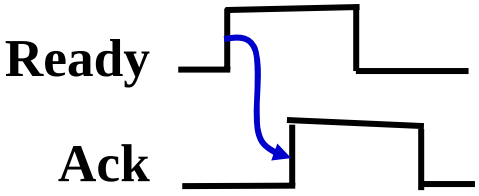
（2）由于时钟偏移问题，同步总线不能很长

实际上，存储器总线比这种协议的总线复杂得多

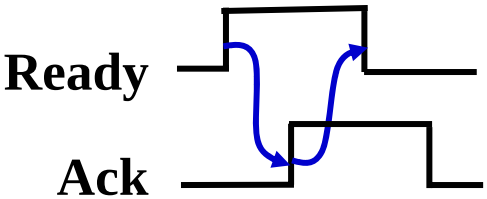
存储器（从设备）响应需要一段时间，并不能在随后的时钟周期就准备好数据

异步总线 (Asynchronous Bus)

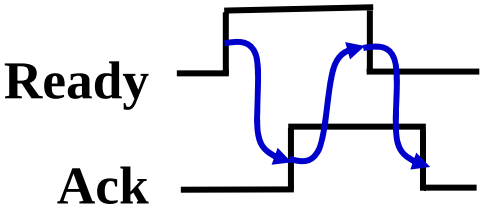
- 非时钟定时，没有一个公共的时钟标准。因此，能够连接带宽范围很大的各种设备。总线能够加长而不用担心时钟偏移（**clock skew**）问题
- 采用握手协议（**handshaking protocol**）即：应答方式。
  - 只有当双方都同意时，发送者或接收者才会进入到下一步，协议通过一对附加的“握手”信号线（**Ready**、**Ack**）来实现
- 异步通信有非互锁、半互锁和全互锁三种方式



非互锁方式



半互锁方式



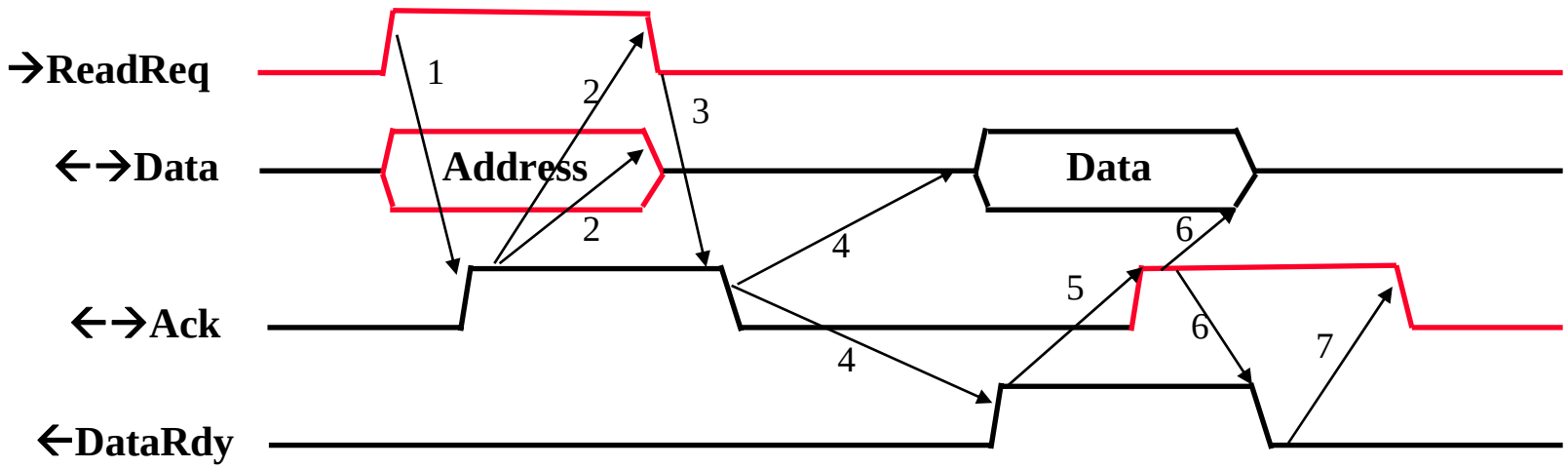
全互锁方式

- 优点：灵活，可挂接各种具有不同工作速度的设备
- 缺点：① 对噪声较敏感（任何时候都可能接收到对方的应答信号）
  ② 接口逻辑较复杂

BACK

Handshaking Protocol(握手协议)

一个总线事务：地址阶段 + 数据阶段 + … + 数据阶段



- Three control lines
  - **ReadReq:** 请求读内存单元
    (地址信息同时送到地址/数据线上)
  - **DataRdy:** 表示已准备好数据
    (数据同时送到地址/数据线上)
  - **Ack: ReadReq or DataRdy**的回答信号
- 上述为**read**过程, 但**write**操作基本类似

ReadReq和**Ack**之间的握手过程
完成地址信息的传输
DataRdy和**Ack**之间的握手过程
完成数据信息的传输
一共有多少次握手？ **7**次
是全互锁方式！

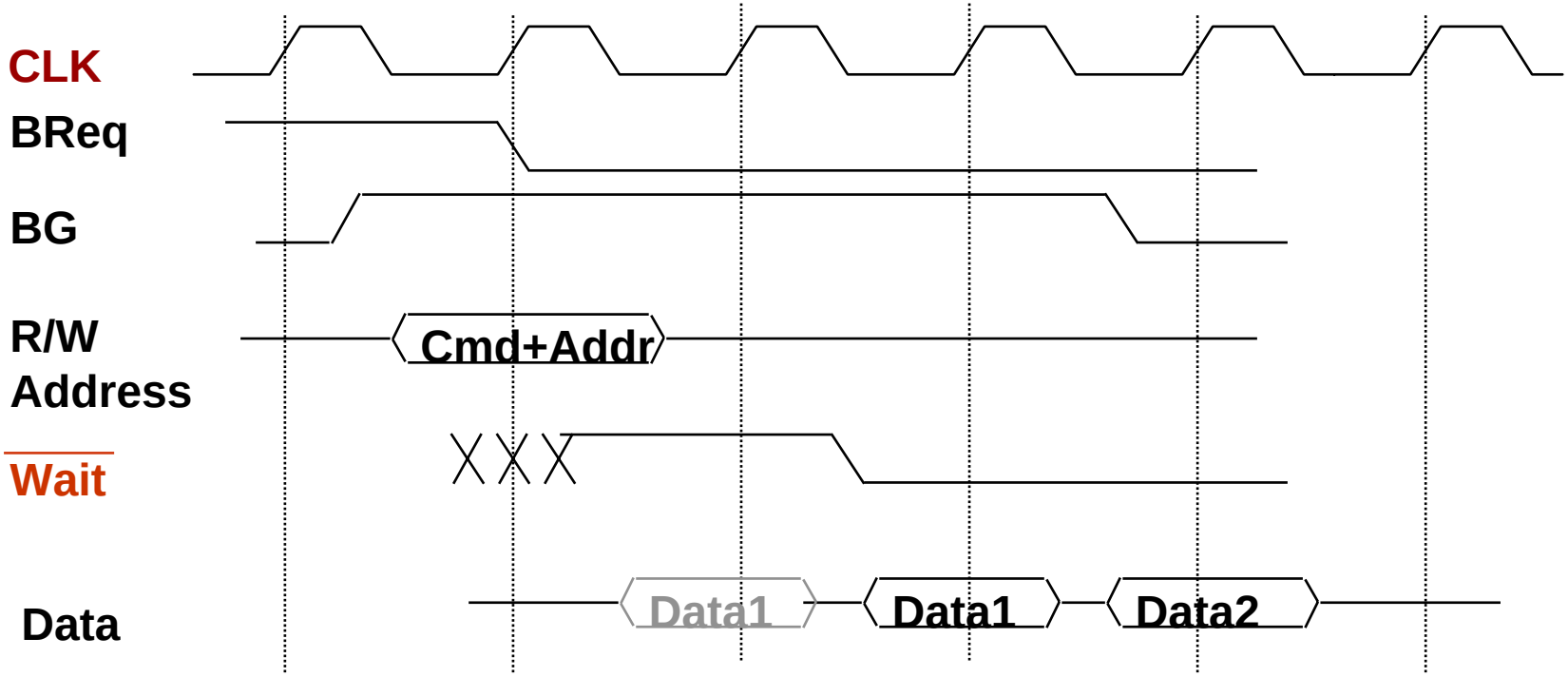
BACK

半同步总线

为解决异步方式对噪声敏感的问题，在异步总线中引入时钟信号

就绪和应答等握手信号 (如：Wait信号、TRDY和IRDY信号等) 都在时钟的上升沿有效

信号的有效时间限制在时钟到达的时刻，而不受其他时间的信号干扰



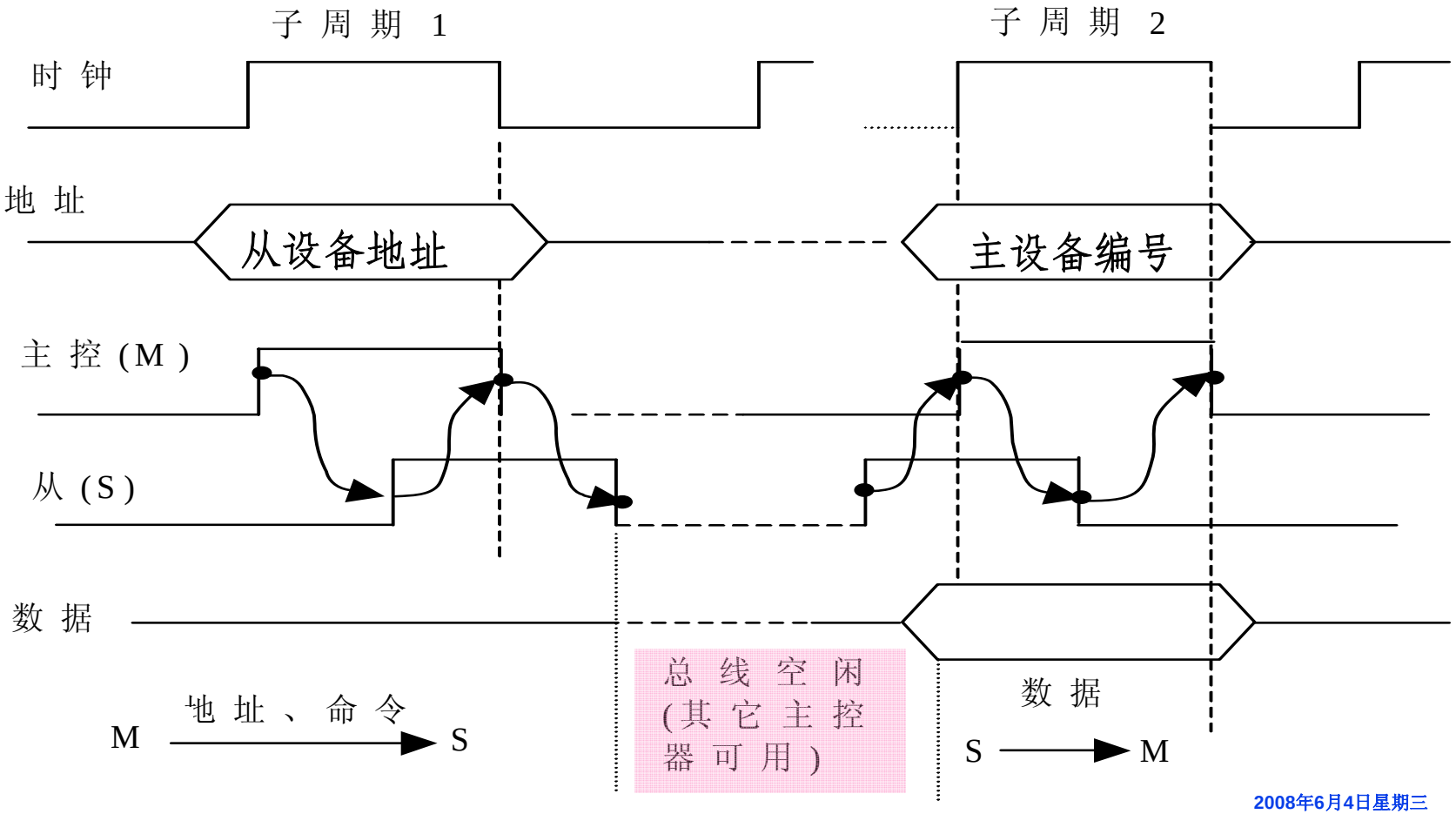
- 通过“Wait”信号从设备告知主设备何时数据有效
- 结合了同步和异步的优点。既保持了“所有信号都由时钟定时”的特点，又允许“不同速度设备共存于总线”

BACK

Split Bus Transaction(拆分总线事务)

将一个事务分成两个子过程：

- 过程1：主控设备A获得总线使用权后，将请求的事务类型、地址及其他信息（如A的标识等）发到总线，从设备B记下这些信息。A发完信息后便立即释放总线，其他设备便可使用总线
- 过程2：B收到A发来的信息后，按照A的要求准备数据，准备好后，B便请求使用总线，获使用权后，B将A的编号及所需数据送到总线，A便可接收



Split Bus Transaction(拆分总线事务)

- 请求- 回答方式（**Request-Reply** ）
  - **CPU**启动一次读或写事务
    - 传送信息：**address, data, and command**
  - 然后等待存储器回答

- 分离总线事务方式（**Split Bus Transaction**）
  - **CPU**启动一次读/写事务后，释放总线
    - 传送信息：**address, data（Write）, and command**
  - 存储器启动一次回答事务，请求使用总线
    - 传送信息： **data (read) or acknowledge (write)**

优点：系统总效率改善

(例如，在存储器存取数据时可以释放总线，以被其他设备使用)

缺点： 单独的事务响应时间变长

增加复杂性

BACK

例1：同步和异步总线的最大带宽比较

举例：假定同步总线的时钟周期为50ns，每次总线传输花1个时钟周期，异步总线每次握手需要40ns，两种总线的数据都是32位宽，存储器的取数时间为200ns。要求求出从该存储器中读出一个字时两种总线的带宽。

分析如下：

同步总线的步骤和时间为：

（1）发送地址和读命令到存储器：50ns

（2）存储器读数据：200ns

（3）传送数据到设备：50ns

如果存储器读为230ns，则结果为多少？

总时间为350ns，4B/350ns=11.4MB/s

所以总时间为300ns，故最大总线带宽为4B/300ns，即：13.3MB/s。

异步总线的步骤和时间为：

第1步为：40ns；

第2、3、4步为：Max(3x40ns，200ns)=200ns；

（第2、3、4步都和存储器访问时间重叠）

第5、6、7步为：3x40ns=120ns。

总时间为360ns，故最大带宽为4B/360ns=11.1MB/s

由此可知：同步总线仅比异步快大约20%。要获得这样的速度，异步总线上的设备和存储器必须足够快，以使每次在40 ns内能完成一个子过程

例2：数据块大小对带宽的影响

假定有一个系统具有下列特性：

(1)系统支持4～16个32位字的块访问。

(2)64位同步总线，时钟频率为200MHz，每个64位数据传输需一个时钟周期，地址发送到存储器需1个时钟周期。

(3)在每次总线操作(事务)间有两个空闲时钟周期。

(4)存储器访问时间对于开始的4个字是200ns，随后每4个字是20ns。

假定读出数据的总线传送和随后4个字的存储器读操作可重叠进行

一个总线事务是由一个地址传送后跟一个数据块传送组成的

请求出分别用4-字块和16-字块方式读取256个字时的持续带宽和等待时间。并且求出两种情况下每秒钟内的有效总线事务数。

举例-数据块大小对带宽的影响

分析 4-字块传送情况：

对于4-字块传送方式，一次总线事务由一个地址传送后跟一个4-字块的数据传送组成。也即每个总线事务传送一个4个字的数据块。

每个数据块所花时间为：

(1) 发送一个地址到主存花1个时钟周期

(2) 从主存读4个字花：200ns/(5ns/Cycle)=40个时钟周期

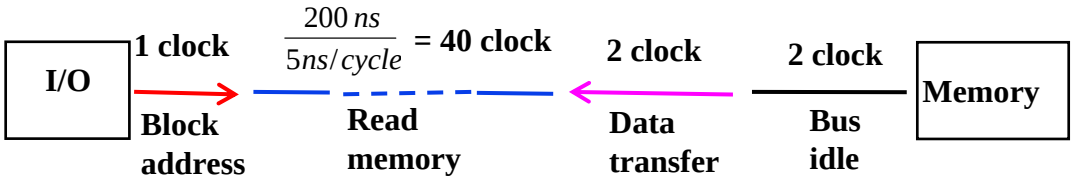
(一个周期是10^9ns/200MHz=1000/200=5ns)

(3) 4个字（128位）的传输需2个时钟周期

(一个64位数据传输需1个时钟周期)

(4) 在这次传送和下次之间有2个空闲时钟周期

所以一次总线事务总共需45个周期，256个字需256/4=64个事务，所以整个传送需45x64=2880个时钟周期，因而总等待时间为：2880周期x 5ns/周期=14400ns。每秒钟的总线事务数为：64 x (1s/14400ns) = 4.44M个。总线带宽为：(256 x 4B)/14400ns =71.11MB/s。



Latency = 2880 clock cycles
Bandwidth = 71.11MB /sec

举例-数据块大小对带宽的影响

分析 16-字块传送情况：

对于16-字块传送，一次总线事务由一个地址传送后跟一个16-字块的数据传送组成。也即每个总线事务传送一个16个字的数据块。

第一个4-字所花时间为：

(1) 发送一个地址到主存花1个时钟周期

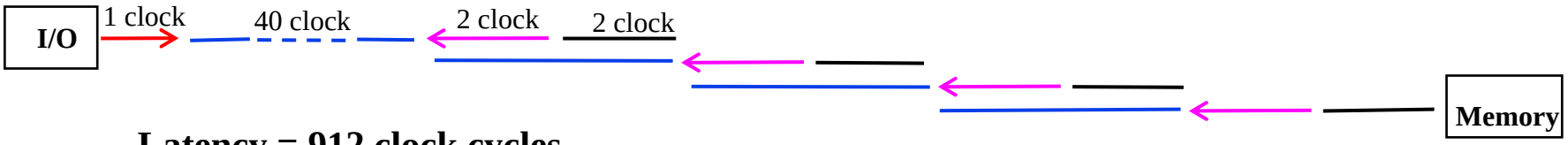
(2) 从主存读开始的4字花：200ns/(5ns/Cycle)=40个时钟周期

(3) 4个字需2个时钟周期，在传输期间存储器开始读取下一个4字

(4) 在本次和下次之间有2个空闲时钟，此期间下一个4字已读完

所以，16字中其余三个4字只要重复上述最后两步。因此对于16-字块传送，一次总线事务共需花费的周期数为：1+40+4 x (2 +2) = 57个周期，256个字需256 / 16=16个事务，因此整个传送需57 x 16 = 912个时钟周期。故总等待时间为：912周期x 5ns / 周期=4560ns。几乎仅是前者的1/3。每秒钟的总线事务个数为：16 x (1s / 4560ns) = 3.51M个。总线带宽为：(256 x 4B)x (1s/4560ns) =224.56MB/s，比前者高3.6倍。

由此可见，大数据块传输的优势非常明显。



Latency = 912 clock cycles
Bandwidth = 224.56MB /sec

Increasing the Bus Bandwidth

- 不采用分时复用方式:
  - 地址和数据可以同时送出
  - **Cost(代价): (a) more bus lines, (b) increased complexity**
- **Data bus width(增加数据线宽度):**
  - 以较少的周期传送多个字
  - **Example: SPARCstation 20's memory bus 有 128 bit**
  - **Cost: more bus lines**
- **Block transfers(允许大数据块传送):**
  - 背对背总线周期，也称为突发**(Burst)**传输方式 **)**
  - 只要开始送一次地址，后面连续送数据
  - **Cost: (a) increased complexity**

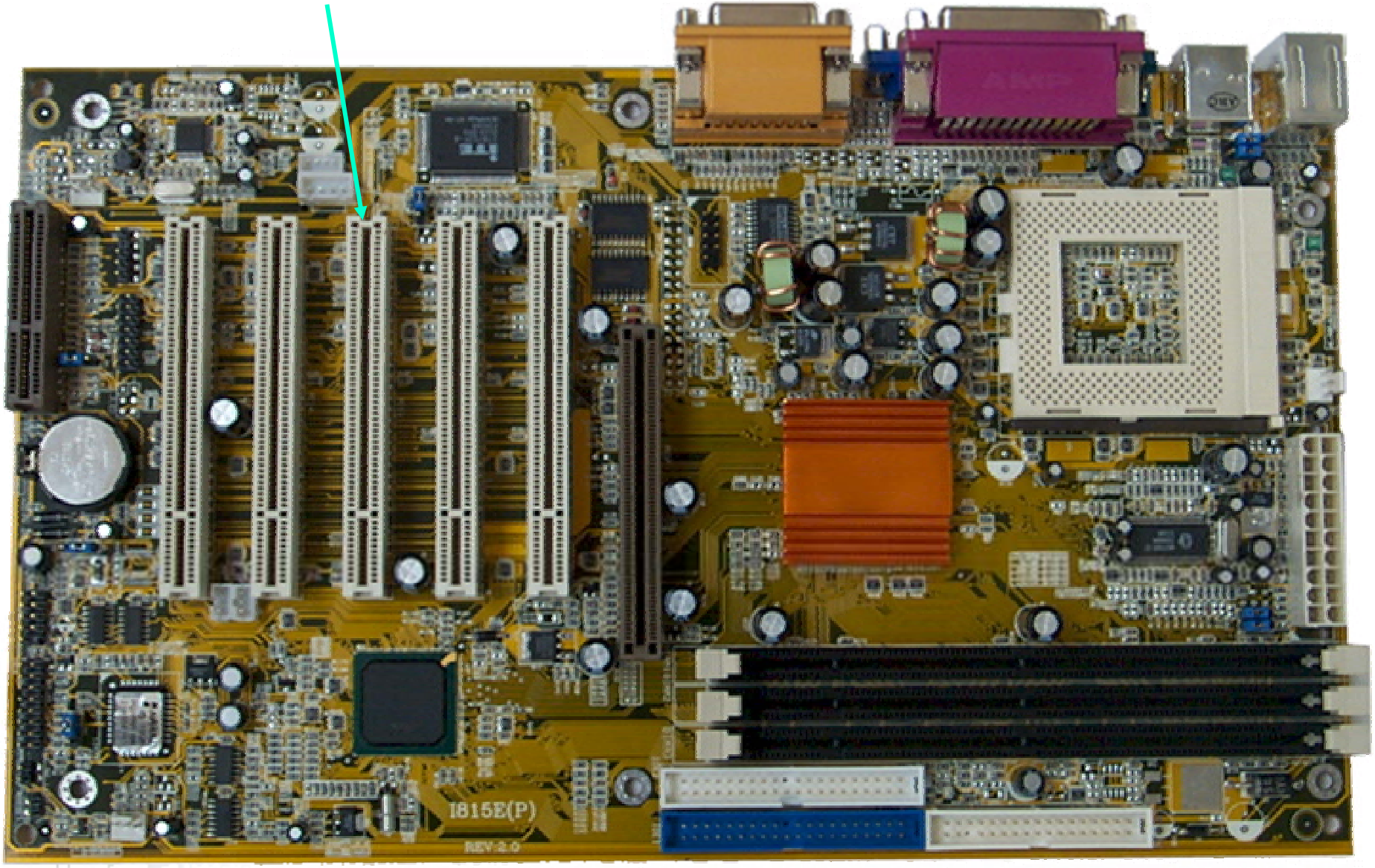
    (b) decreased response time for request

关于I/O总线标准

- I/O总线是各类I/O控制器与CPU、内存之间传输数据的一组公用信号线，这些信号线在物理上与主板扩展槽中插入的扩展卡（I/O控制器）直接连接。
- I/O总线是标准总线，I/O总线标准有：
  - ISA / EISA总线：（已逐步被淘汰）
  - Multibus总线：（已逐步被淘汰）
  - PCI总线：目前PC机所用的主流标准
  - PCI-Express(高速PCI总线)：目前PC机所用的主流标准
- I/O总线的带宽
  - 总线的数据传输速率(MB/s) =

    数据线位数/8×总线工作频率（MHz）×每个总线周期的传输次数

PCI总线扩展槽



（自学） PCI总线标准

(1) 信号线

PCI有50根必须的信号线。按功能可分为以下几组：

- 系统信号：包括时钟和复位线。
- 地址和数据信号：包含32根分时复用的地址/数据线、4根分时复用的总线命令/字节使能线以及对这36根信号线进行奇偶校验的一根校验信号线。
- 接口控制信号：对总线事务进行定时控制，用于在事务的发起者和响应者之间进行协调。
- 裁决信号：它不同于其他信号，不是所有设备共享同一根信号线，而是每个总线主控设备都有一对仲裁线：总线请求和总线允许。PCI采用集中式裁决，所有设备的仲裁线都连接到一个总线裁决器中。
- 错误报告信号：用于报告奇偶校验错以及其他错误。

(2) PCI命令

- 总线活动以发生在总线主控设备和从设备之间的总线事务形式进行。总线主控设备就是事务的发起者，从设备是事务的响应者，即目标。当总线主控设备获得总线使用权后，在事务的地址周期，通过分时复用的总线命令/字节使能信号线C/BE发出总线命令，也即事务类型。

（自学） PCI总线标准

PCI的总线命令（事务类型）有：

- 中断响应：用于对PCI总线上的中断控制器提出的中断请求进行响应。地址线不起作用，在数据周期从中断控制器读取一个中断向量，此时C/BE信号线表示读取的中断向量的长度
- 特殊周期：用于总线主设备向一个或多个目标广播一条消息。
- I/O读和I/O写：I/O读/写命令用于在发起者和一个I/O控制器之间进行数据传送
- 存储器读、存储器行读、存储器多行读：用于总线主控设备从存储器中读取数据。PCI支持突发传送，所以它将占用一个或多个数据周期。这些命令的解释依赖于总线上的存储控制器是否支持PCI的高速缓存协议。如果支持的话，那么，与存储器之间的数据传送以Cache行的方式进行
- 存储器写、存储器写并无效：这两种存储器写命令用于总线主控设备向存储器写数据，它们将占用一个或多个数据周期。其中存储器写并无效命令用于回写Cache行到存储器，所以它必须保证至少有一个Cache行被写回
- 配置读、配置写：用于一个总线主控设备对连接到PCI总线上的设备中的配置参数进行读或更新。每个PCI设备都有一个寄存器组（最多可有256个寄存器），这个寄存器用于系统初始化时对本设备进行配置
- 双地址周期：由一个事务发起者用来表明它将使用64位地址来寻址

（自学） PCI总线标准

PCI 总线上存储器读命令的含义

读命令类型	支持 Cache 的内存	不支持 Cache 的内存
存储器读	突发传送半个或不到一个 Cache 行	突发传送两个数据周期或更少
存储器行读	突发传送半个以上到 3 个 Cache 行	突发传送 3 个到 12 个数据周期
存储器多行读	突发传送 3 个以上 Cache 行	突发传送 12 个以上数据周期

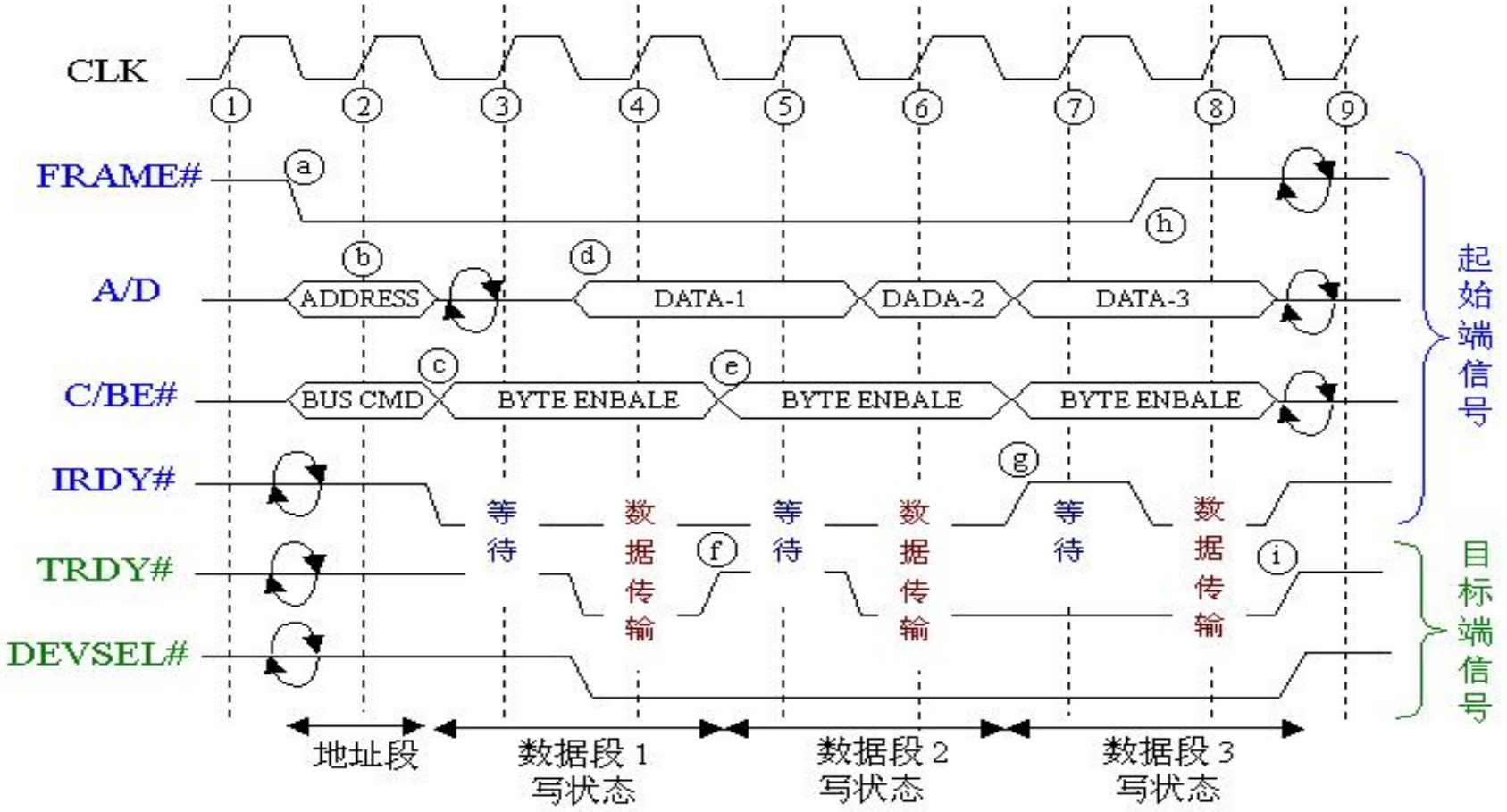
存储器读、存储器行读、存储器多行读：用于总线主控设备从存储器中读取数据。**PCI**支持突发传送，所以它将占用一个或多个数据周期。这些命令的解释依赖于总线上的存储控制器是否支持**PCI**的高速缓存协议。如果支持的话，那么，与存储器之间的数据传送以**Cache**行的方式进行

（自学） PCI总线标准

(3) PCI数据传送过程

PCI总线上的数据传送由一个地址周期和一个或多个数据周期组成。

所有事件在时钟下降沿(即在时钟周期中间)同步。总线设备在时钟上升沿采样总线信号



PCI 读操作过程时序图

写操作时序类似！

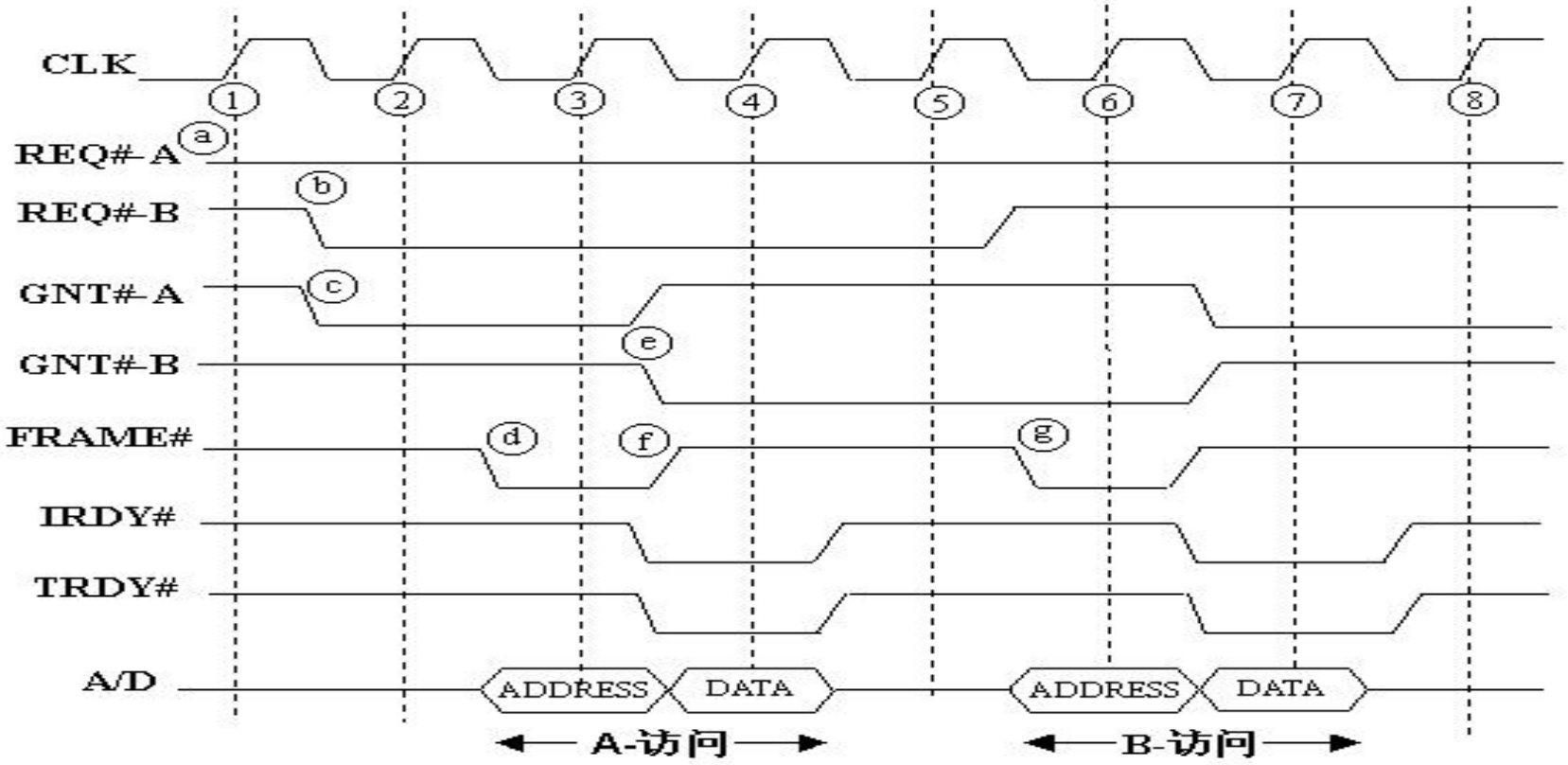
（自学） PCI总线标准

(4) PCI总线裁决

采用独立请求方式，有两个独立的裁决线：请求线**REQ** 和 允许线**GNT**

总线仲裁器可使用静态的固定优先级法、循环优先级法或先来先服务法等仲裁算法

采用隐式仲裁方式，在总线进行数据传送时进行总线仲裁，仲裁不会浪费总线周期



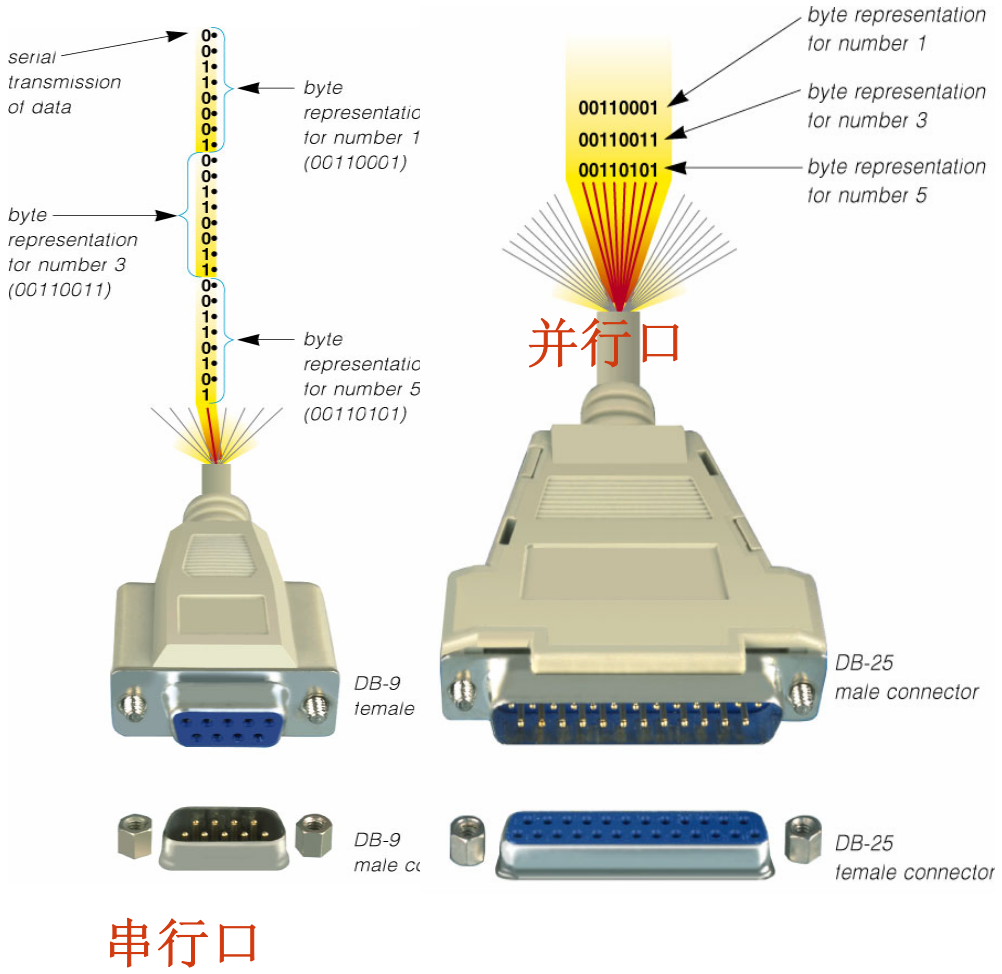
PCI 总线裁决过程示例

回顾：关于I/O接口

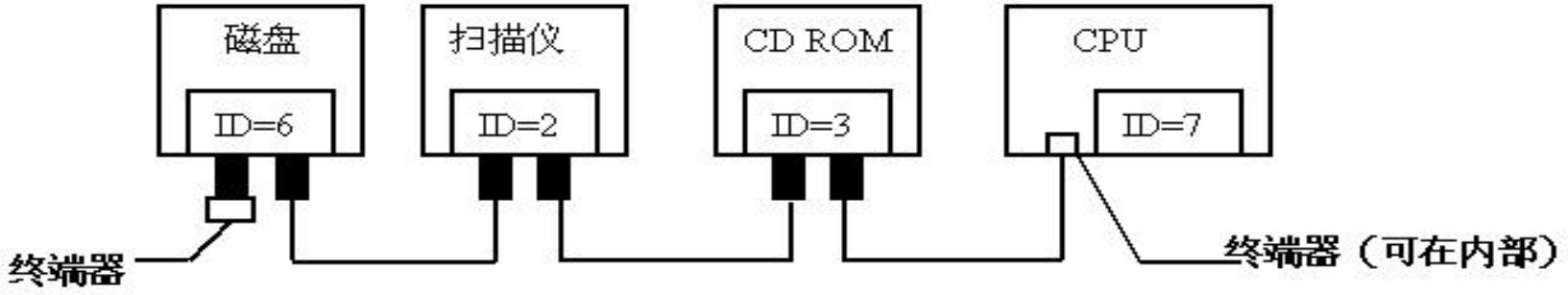
- I/O接口：I/O设备与I/O控制器之间的连接器

  包括：插头 / 插座的形式、通讯规程和电器特性等

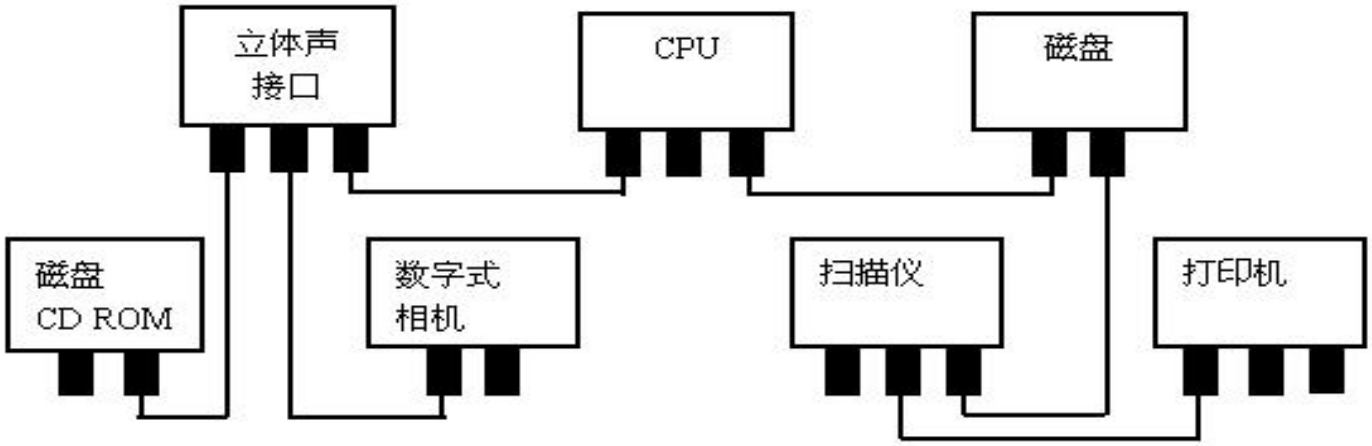
- 分类：
  - 从数据传输方式来分：
    - 串行（一次只传输1位）
    - 并行（多位一起进行传输）
  - 从是否能连接多个设备来分：
    - 总线式（可连接多个设备）
    - 独占式（只能连接1个设备）
  - 从是否符合标准来分：
    - 标准接口 （通用接口）
    - 专用接口 （专用接口）
  - 按功能选择的灵活性来分：
    - 可编程接口
    - 不可编程接口



总线式I/O接口

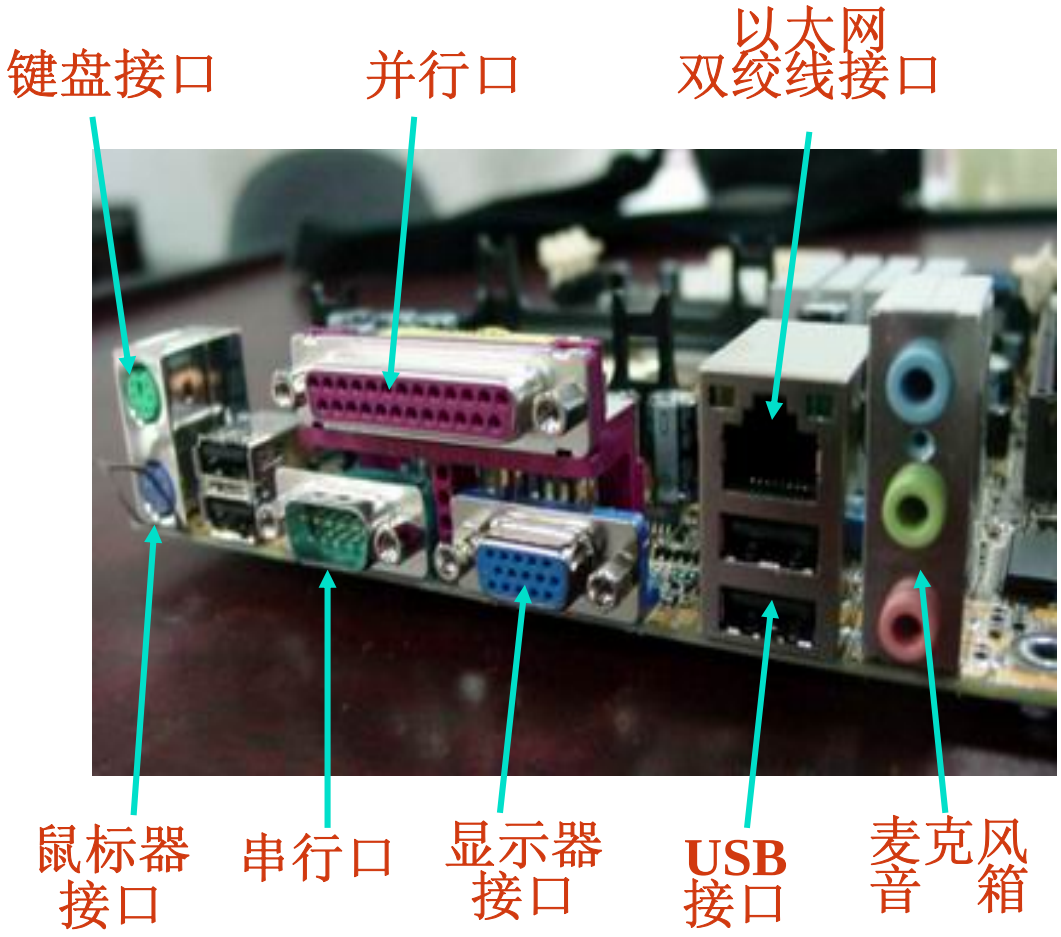


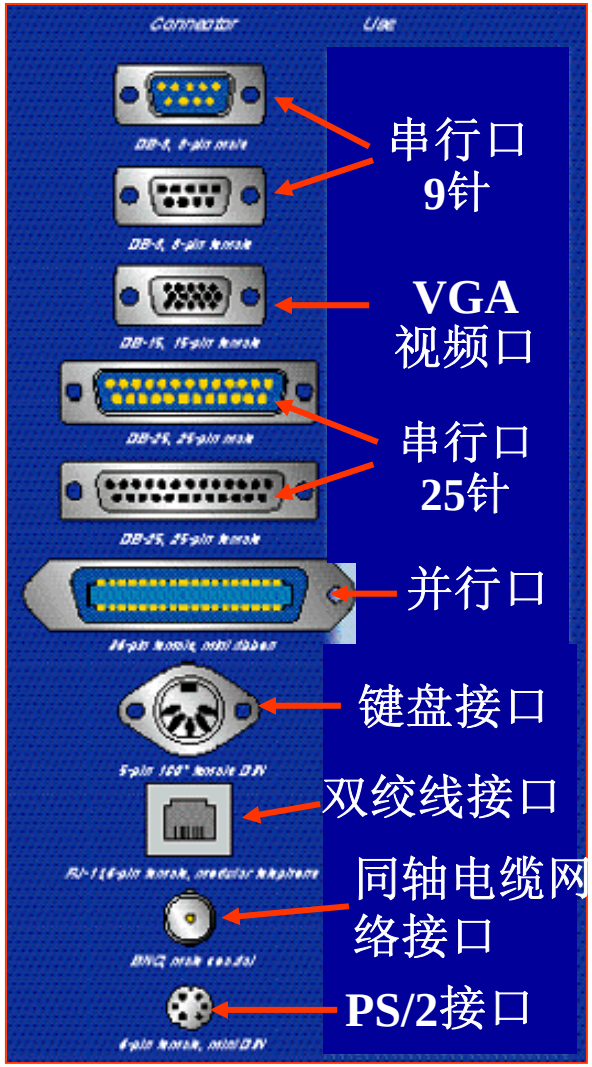
（ a ）SCSI 配置举例



（ b ）P1394 配置举例

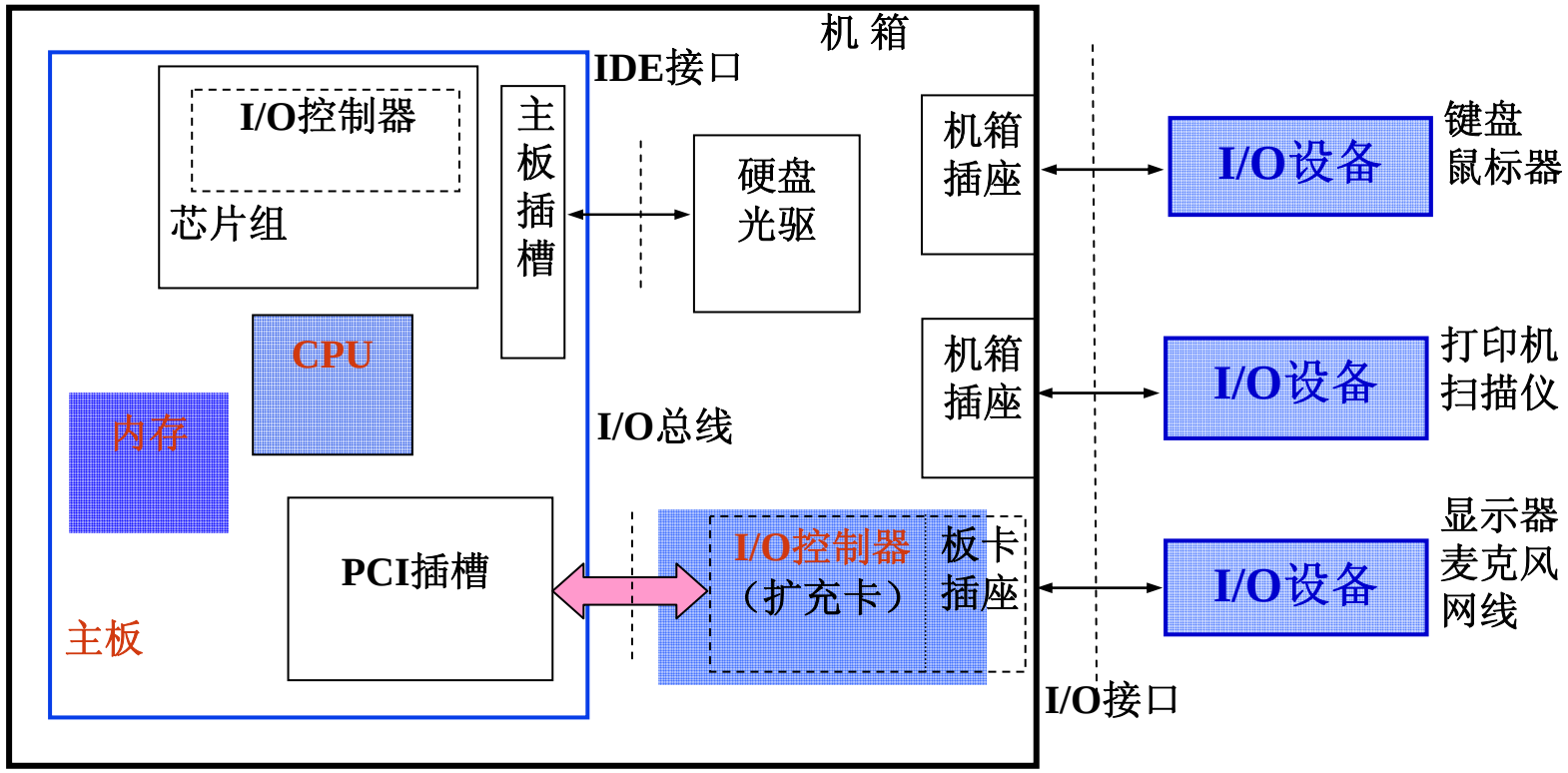
回顾：I/O设备接口





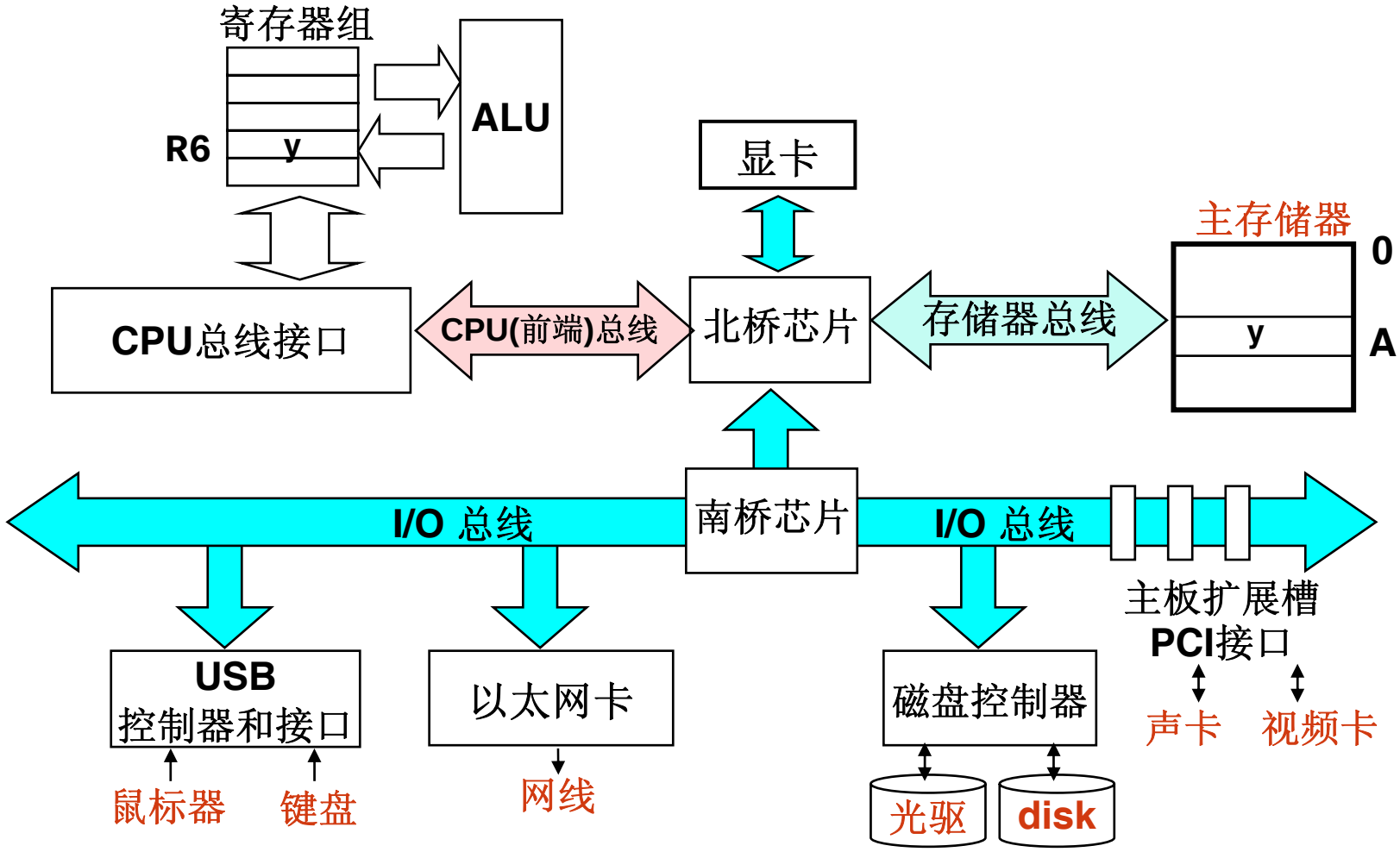
(安装在主板上的I/O设备接口)

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



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。

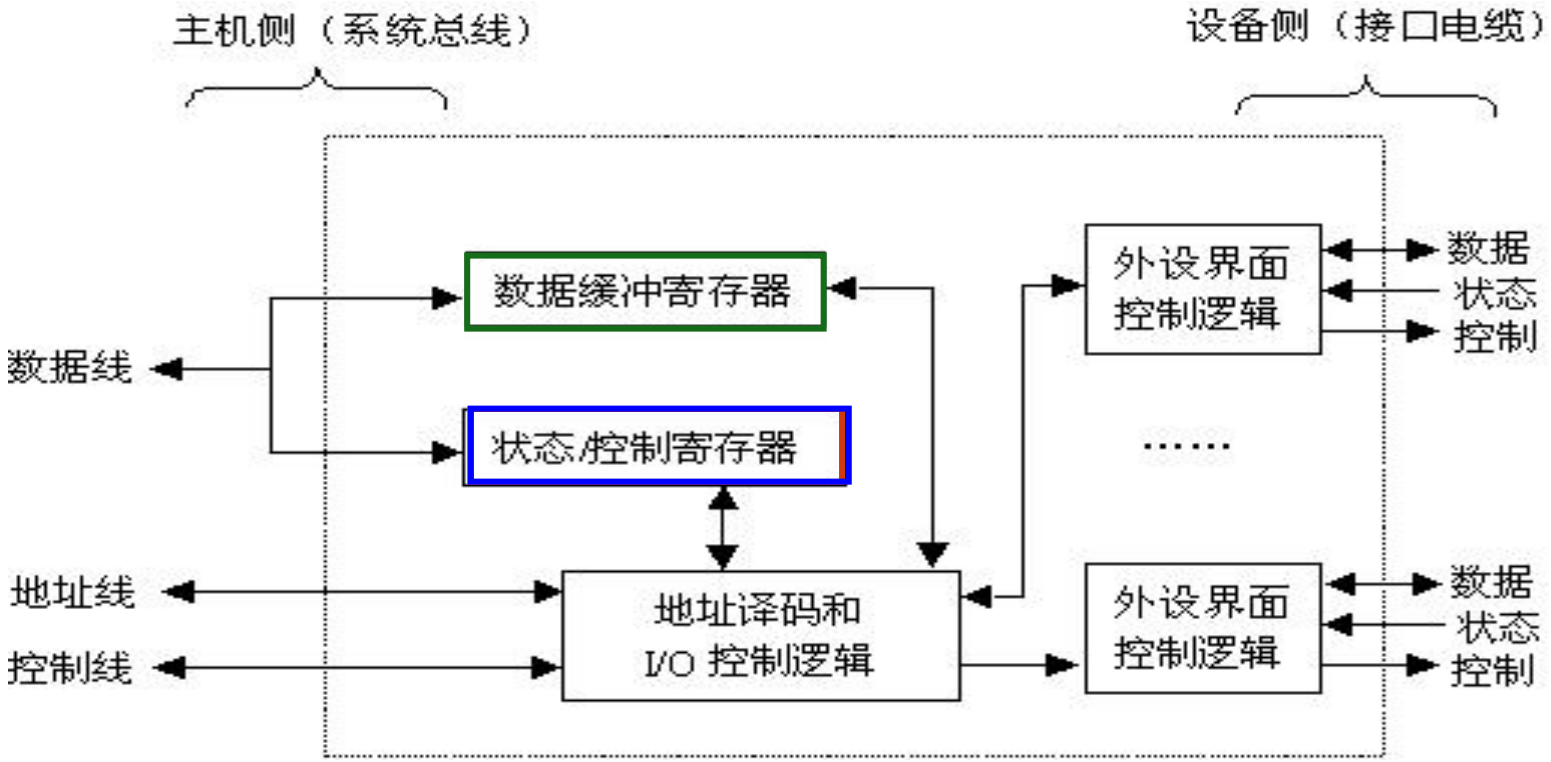
- 与主机和设备通信

  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

操作系统在I/O系统中的角色

- 对于I/O, 操作系统需具有以下三个功能：
  - 使I/O系统被多个运行的程序共享
    - 保证用户程序只能访问I/O设备中用户有权访问的部分
  - 提供对外部中断的中断处理机制
    - 保证与程序异常一致的处理机制
  - 屏蔽I/O设备的底层控制细节，提供标准的I/O设备访问机制给应用程序
    - 提供处理低级设备操作的例行程序，提供对设备的抽象
- 操作系统需提供以下三种与I/O设备的通信方式：
  - 能给I/O设备提供操作命令
    - 如：读/写命令，寻道/旋转/倒带等命令
  - 当I/O设备完成一个操作或遇到一个错误时，能够通知操作系统。
    - 例如：磁盘寻道结束、打印机缺纸等
  - 数据必须能在I/O设备和内存或I/O设备和CPU之间传输
    - 磁盘等成批数据直接和内存交换，I/O端口中的数据和CPU内寄存器交换

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“内存映射方式”）

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）

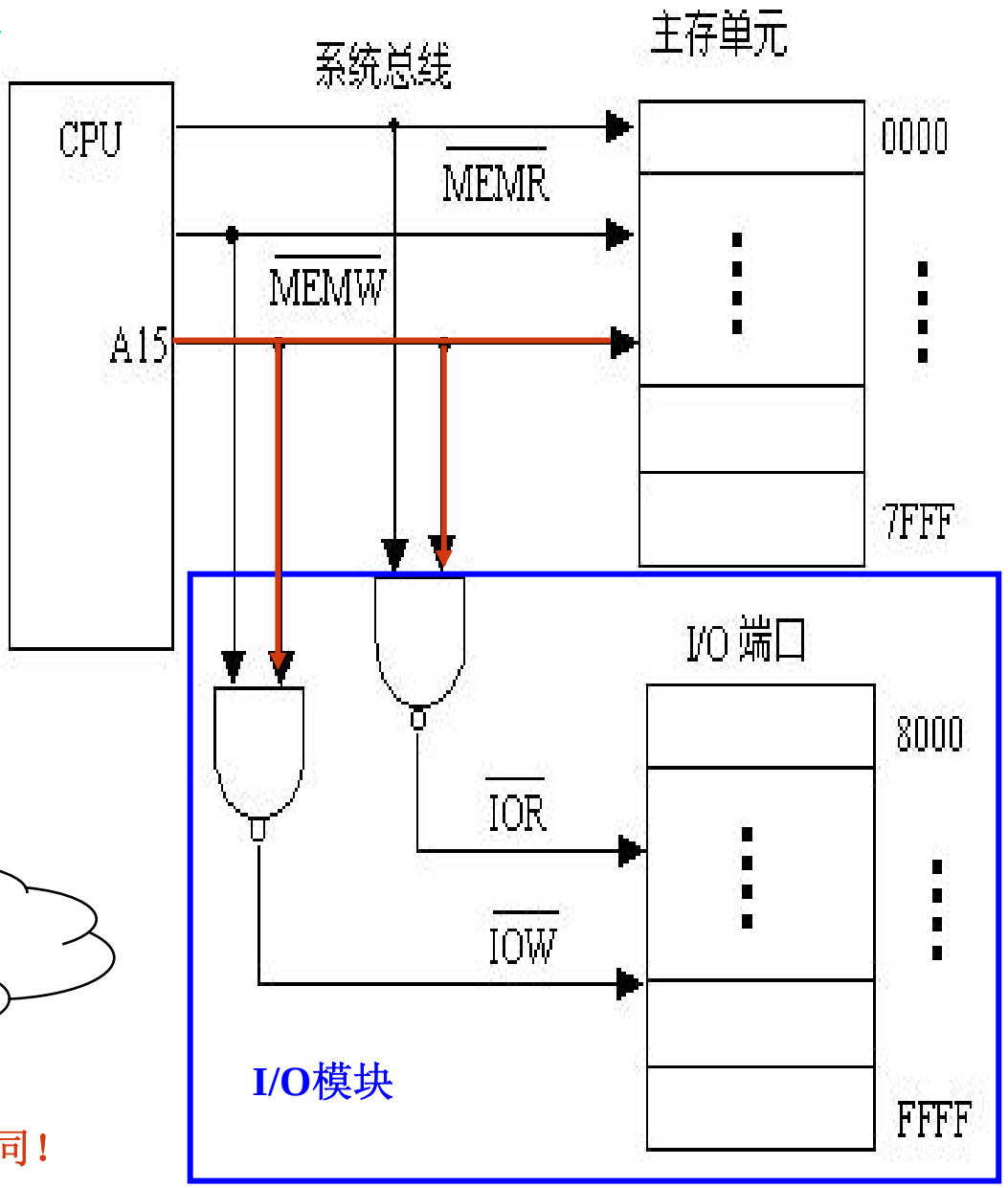
不和主存单元一起编号，而是单独编号，使成为一个独立的I/O地址空间

（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！

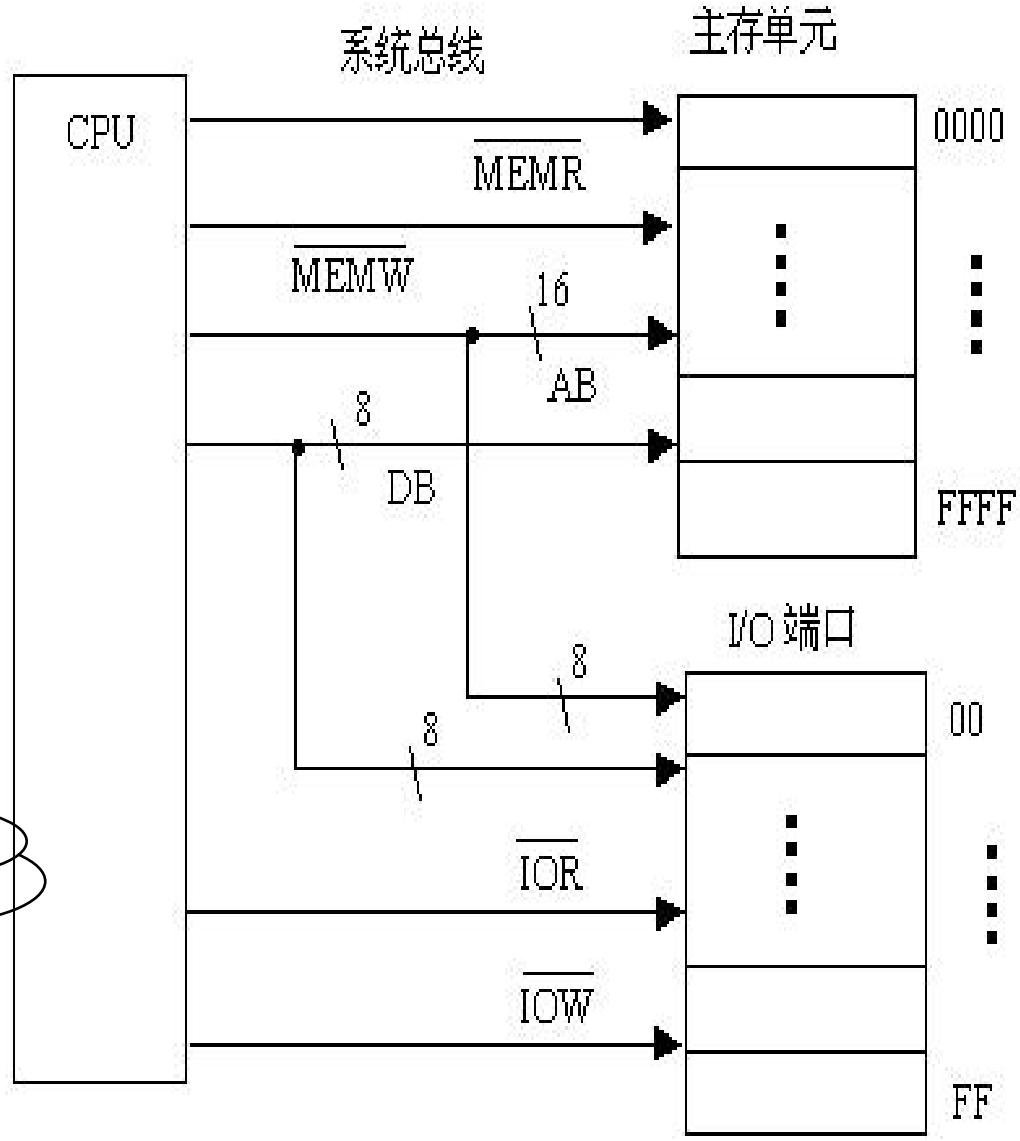
统一编址方式的优缺点

- 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。

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

第二讲小结

- 总线是共享的传输介质和传输控制部件，用于在部件或设备间传输数据
- 总线可能在芯片内、芯片之间、板卡之间和计算机系统之间
- I/O总线是I/O控制器与主机之间传输数据的一组公用信号线，它们在物理上与主板扩展槽中插入的扩展卡（I/O控制器）直接连接。
- 总线可以采用“同步”或“异步”方式进行定时。
  - 同步总线用“时钟”信号定时；异步总线用“握手信号”定时
  - 可以结合同步和异步方式进行半同步定时通信
  - 可以把一个总线事务分离成两个事务，在从设备准备数据时释放总线
- 总线的裁决：有集中和分布两类裁决方式
  - 分布裁决：自举裁决、冲突检测
  - 集中裁决：菊花链、独立请求并行判优
- 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）

第二次作业

- **8.18**
- **8.19**
- **8.20**
- **8.27**

第三讲 I/O设备和主机之间的传输方式

主　要　内　容

- 回顾：OS在I/O系统中的职责
- 最基本的三种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

复习：操作系统在I/O中扮演的角色

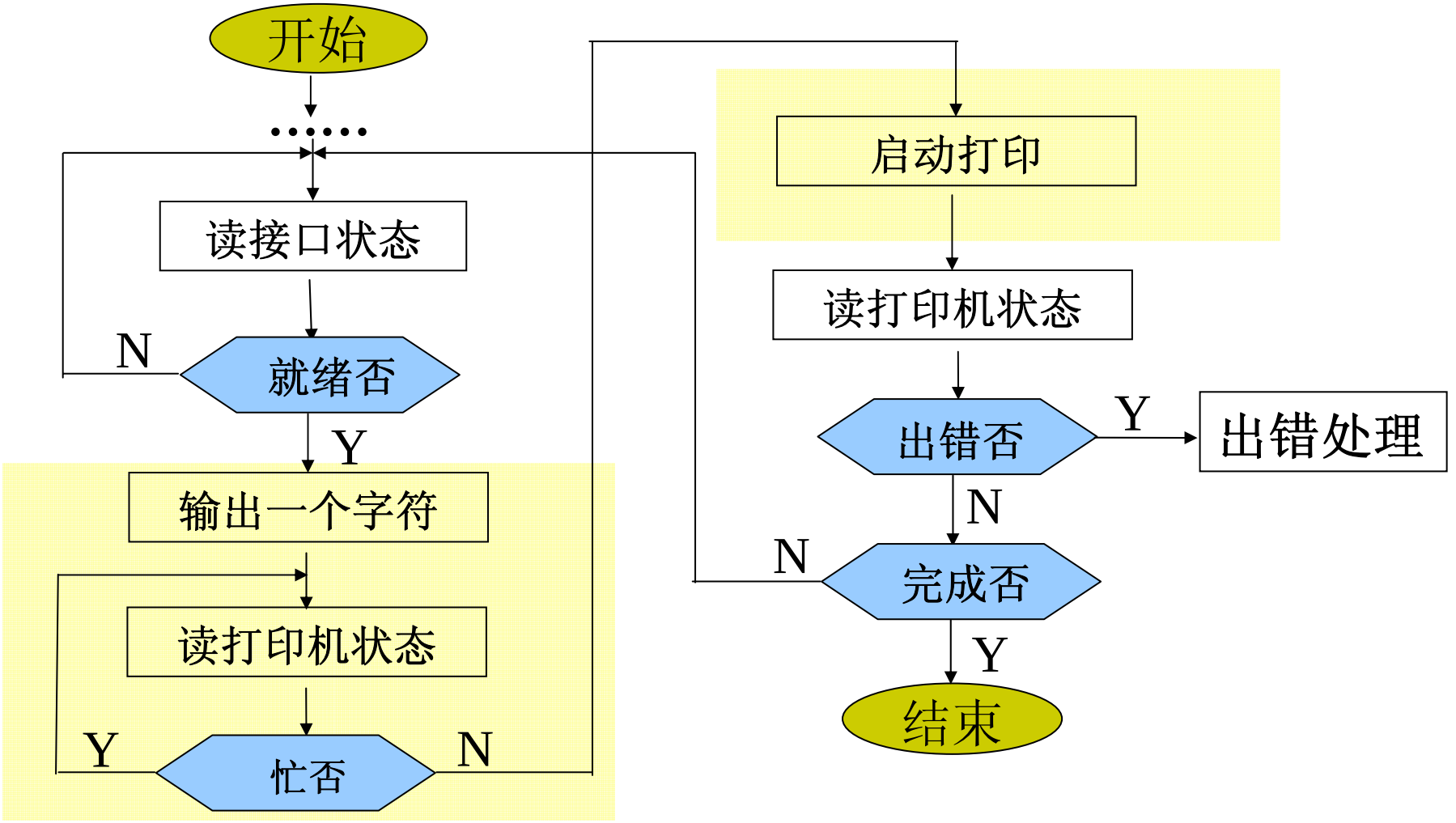
- OS的职责由I/O系统的三个特性决定：
  - I/O系统被处理器执行的多个程序共享，由OS统一调度管理
  - I/O系统通常使用外部中断请求来要求处理器执行专门的输入/出程序。中断导致向内核态转移，故必须由OS来处理
  - I/O设备控制细节复杂，不能由上层的用户程序来实现，需要OS提供专门的驱动程序
- OS在I/O中的职能
  - 保证用户程序只能访问I/O设备自己有权访问的那部分。例如，用户程序不能读/写没有授权的文件。
  - 为用户程序提供设备的驱动程序以屏蔽设备控制的细节
  - 处理外部I/O中断，提供中断服务程序
  - 对共享的I/O资源提供合理的调度管理，使系统的吞吐率达到最佳
- 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

I/O设备与主机进行数据交换的三种方式

- **Polling (轮询):** 最简单的I/O方式
  - I/O设备（包括I/O模块）将自己的状态放到一个状态寄存器
  - 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 (I/O中断):** 几乎所有系统都支持的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

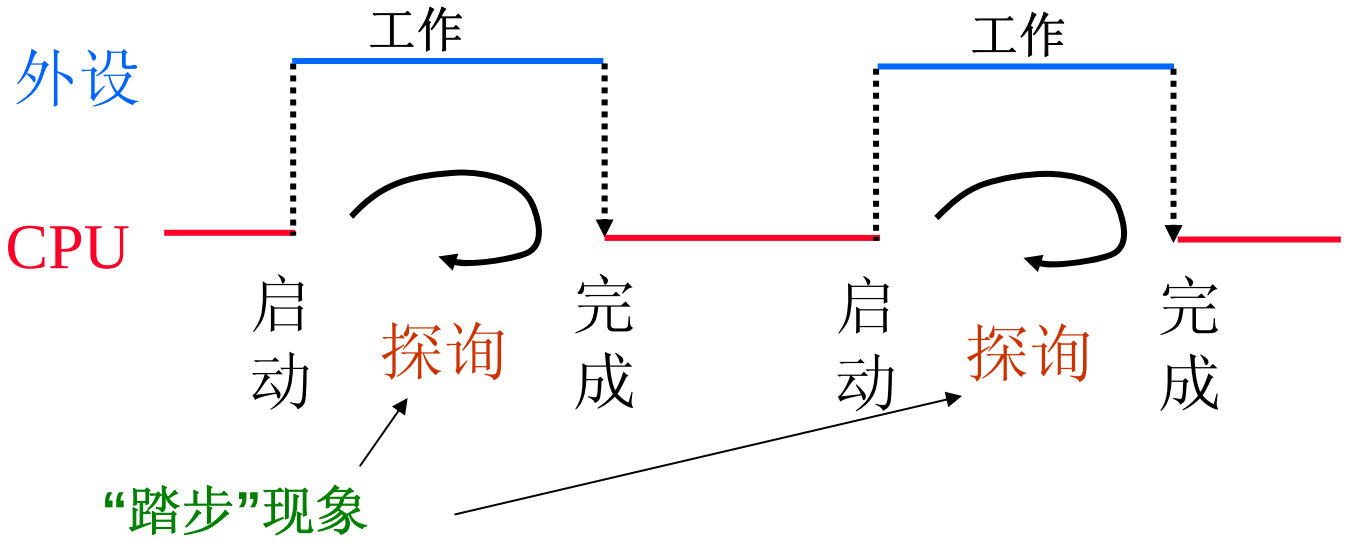


（自学）打印输出标准子程序

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）

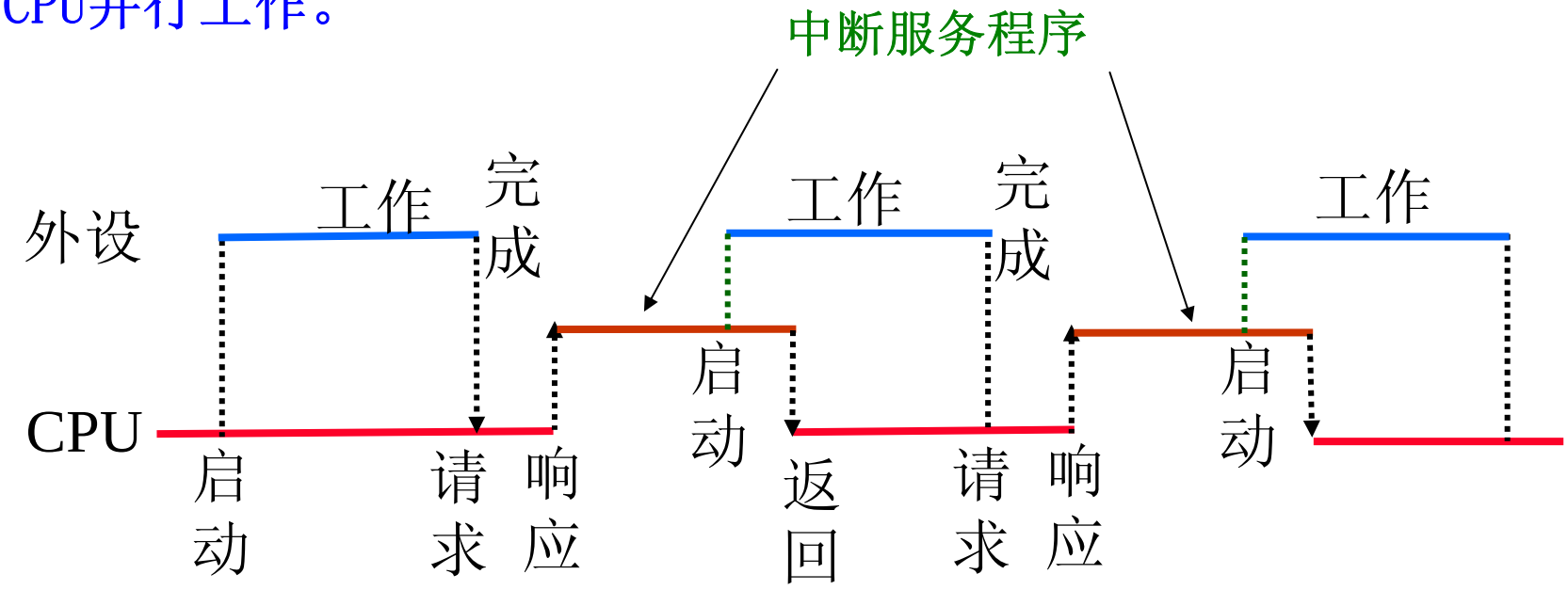


- 特点：
  - 简单、易控制、外围接口控制逻辑少；
  - CPU与外设串行工作，效率低、速度慢，适合于慢速设备
  - 查询开销极大 (CPU完全在等待“外设完成”)
- 工作方式：完全串行工作方式

中断驱动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。

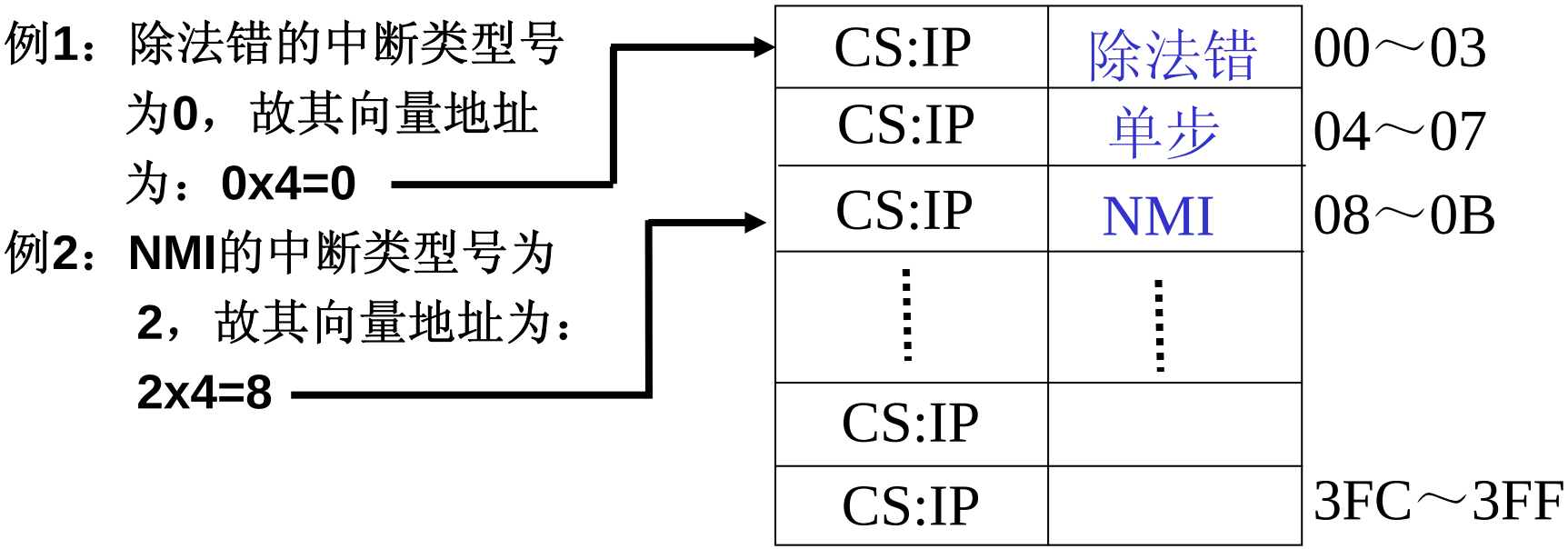
③ 切换到具体的异常处理程序执行

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：**0x4=0**

例**2**：**NMI**的中断类型号为**2**，故其向量地址为：**2x4=8**



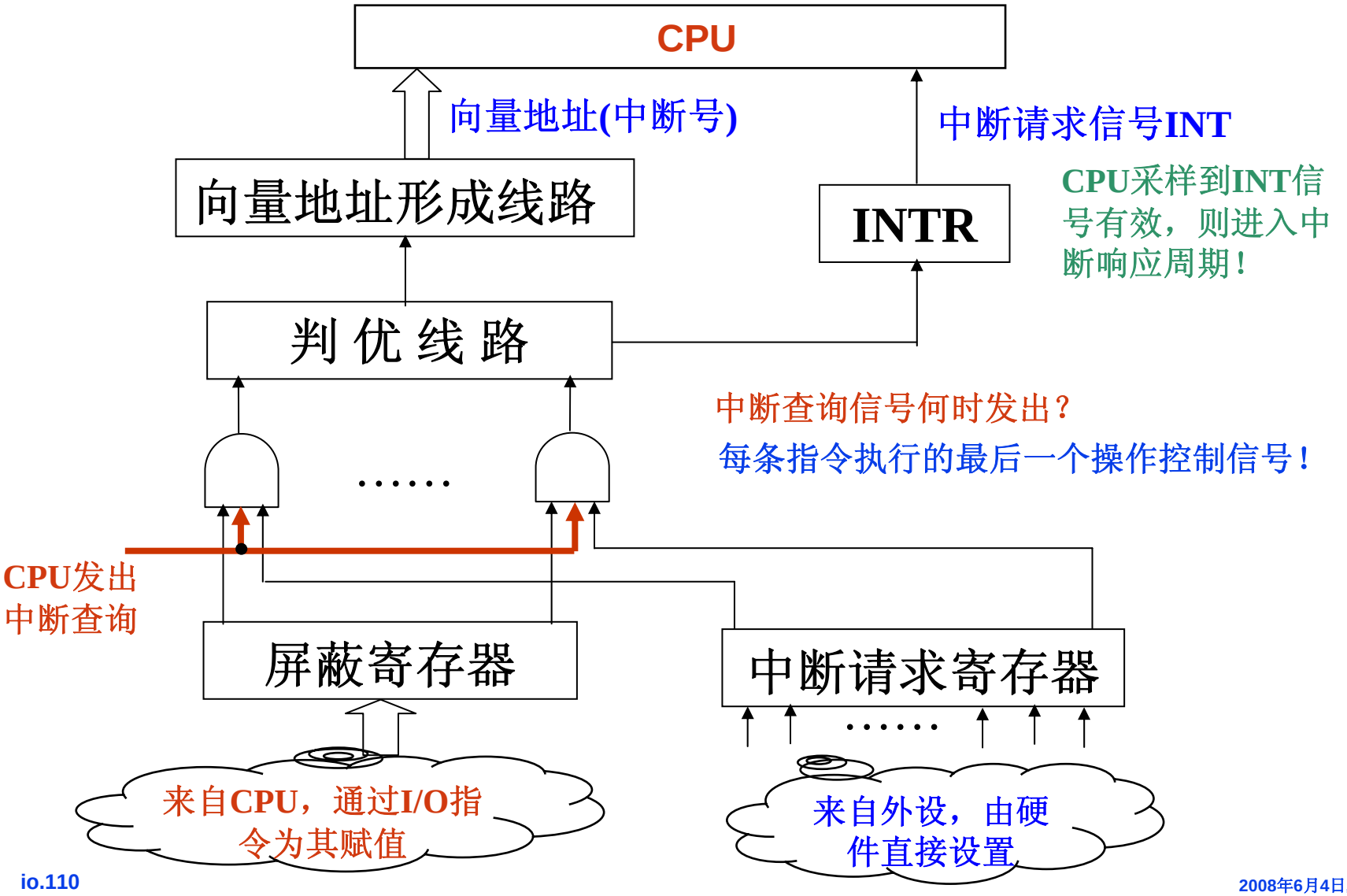
◆ 中断向量表（异常表）中每一项是对应异常处理程序的入口地址。被称为中断向量**(Interrupt Vector)**

◆ 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

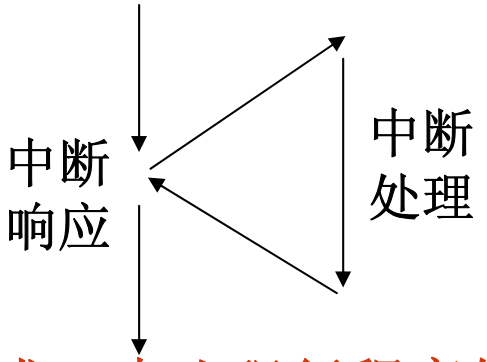
中断号（向量地址）送到什么线上？
数据线 / 地址线？　数据线上！为什么？

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
  - 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。

中断驱动I/O方式

(2)中断响应过程

执行一条隐指令，并完成一次总线操作，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=>中断允许触发器C_{IEN}

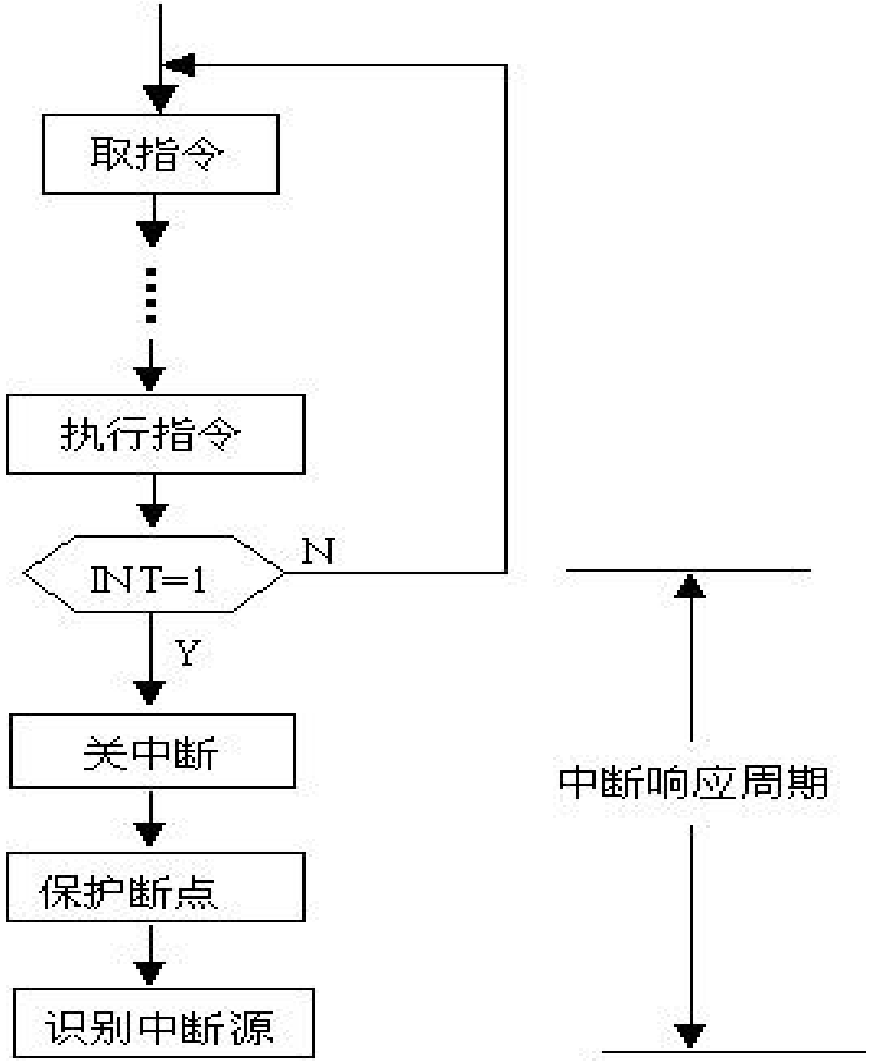
② 保护断点和程序状态

PC=>堆栈（或特殊寄存器EPC）

PSW=>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

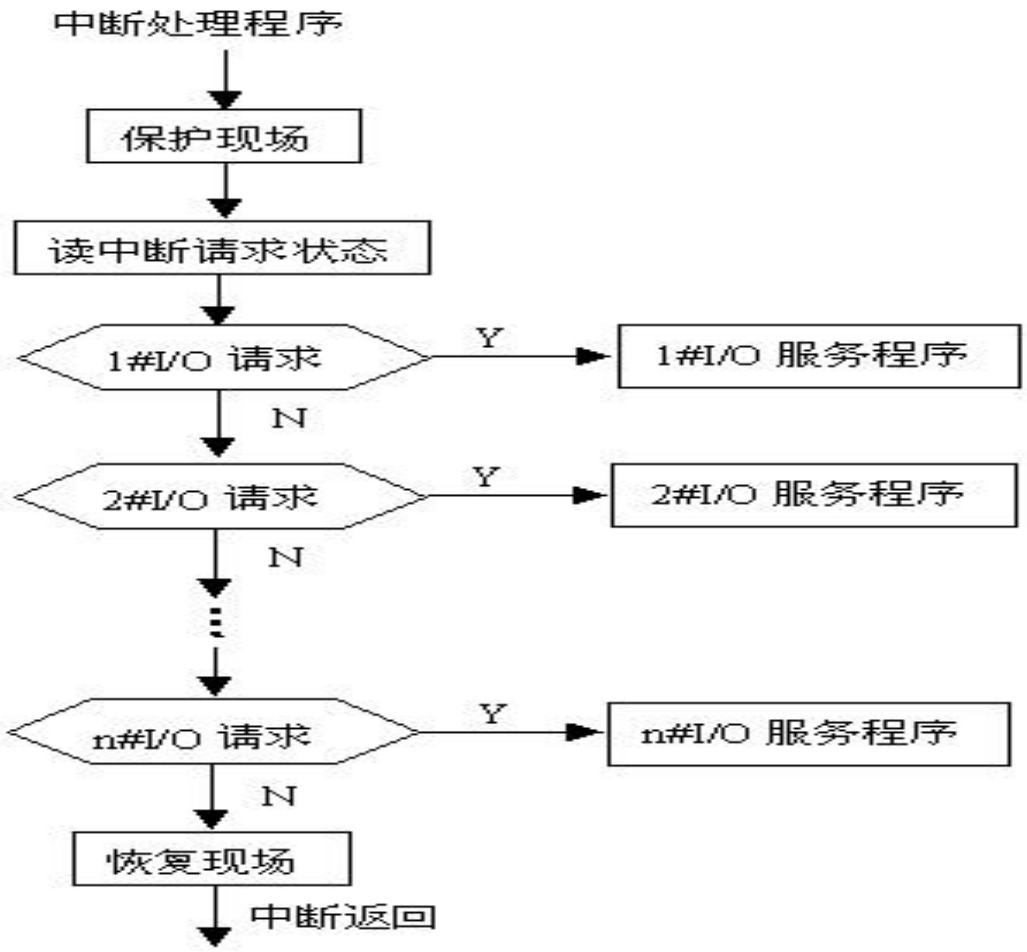
  如：**MIPS的异常/中断查询程序入口为：**

  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

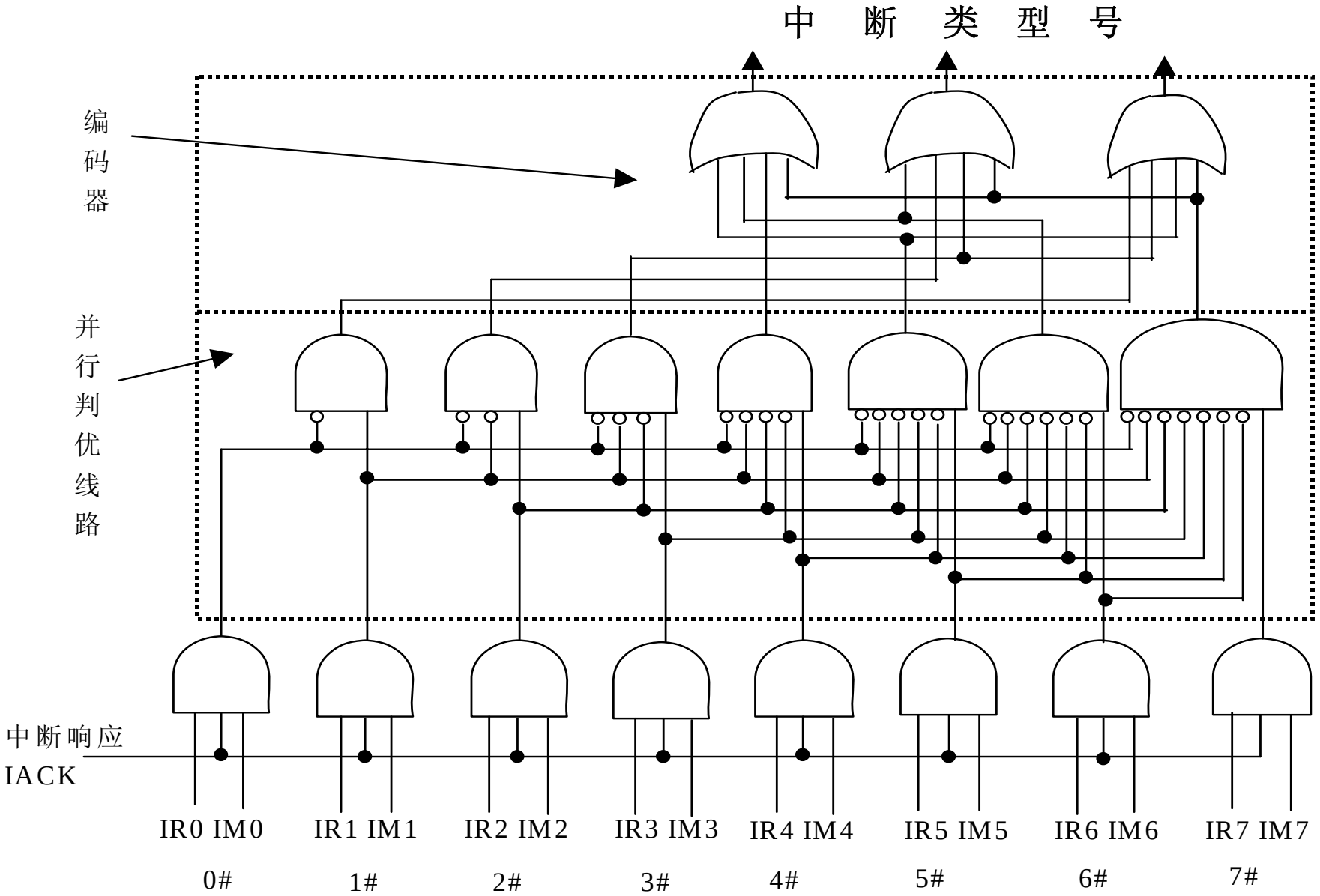
  - 链式查询（菊花链）
  - 独立请求（并行判优）



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修优先级，通常所有内部异常的优先级都比外部中断高。为什么？

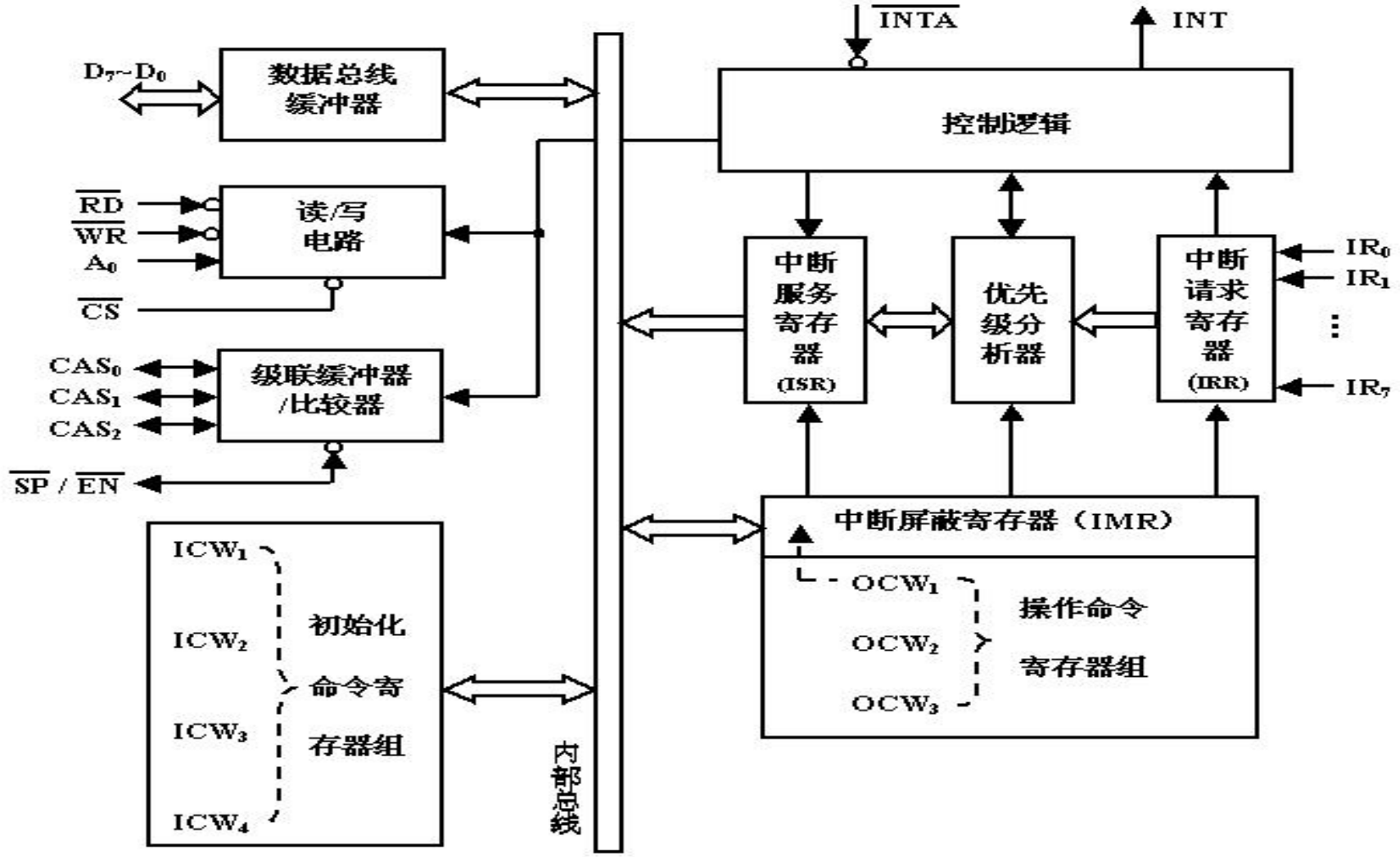
中断优先权编码器


中 断 类 型 号
编码器
并行判优线路
中断响应
IACK
IR0 IM0
IR1 IM1
IR2 IM2
IR3 IM3
IR4 IM4
IR5 IM5
IR6 IM6
IR7 IM7
0#
1#
2#
3#
4#
5#
6#
7#

（自学）中断控制器举例-8259A

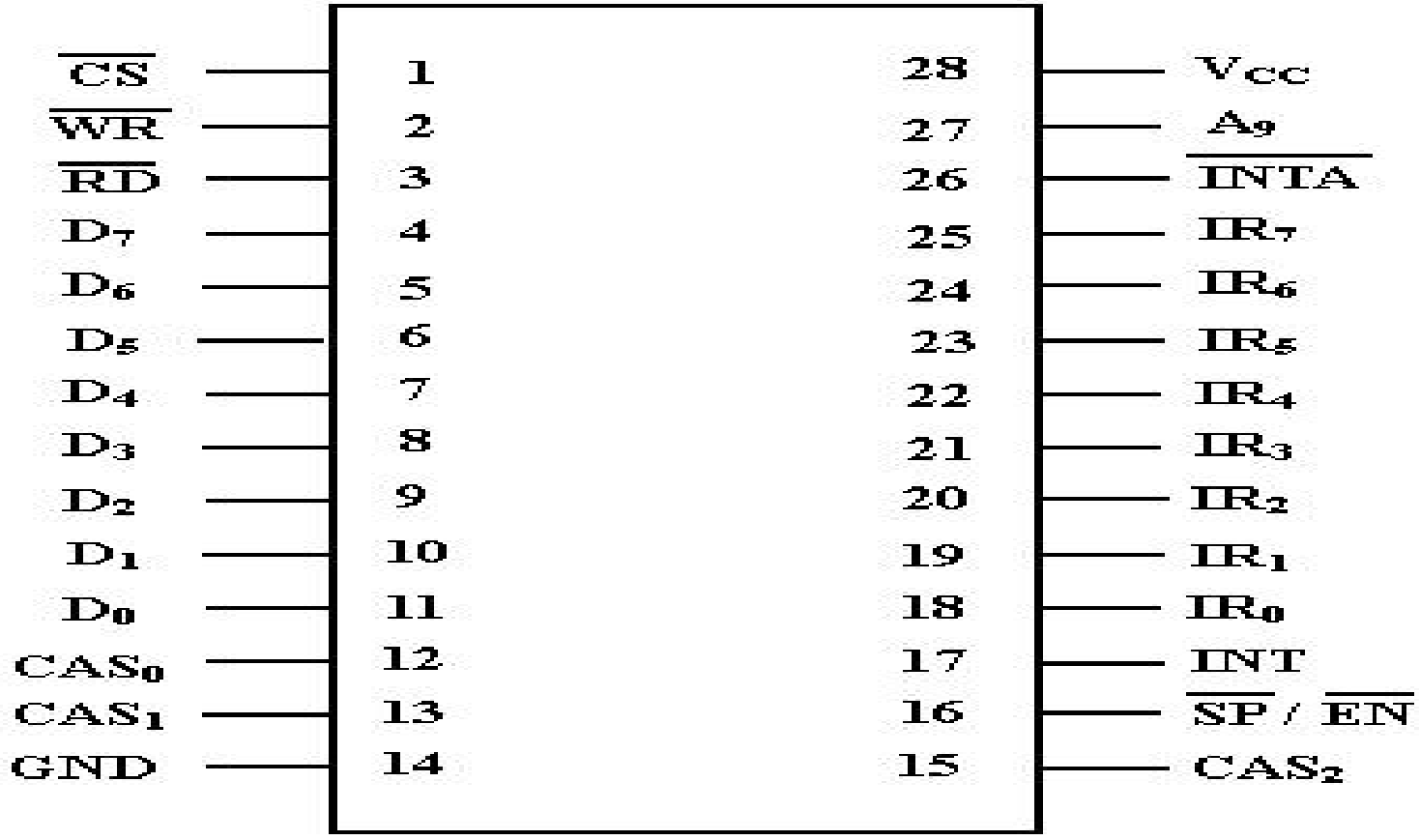
- **8259A**是可编程的中断控制器
- **8259A**的功能
  - 包含中断请求锁存、中断屏蔽、中断优先级排队、中断向量生成（中断优先权编码器）等电路
  - 既可支持程序查询式中断，又可支持向量式中断
  - 支持**8**级优先权，通过多片级联，最多可构成**64**级中断
  - 各种中断功能可通过编程设定或更改

（自学）8259A的内部结构



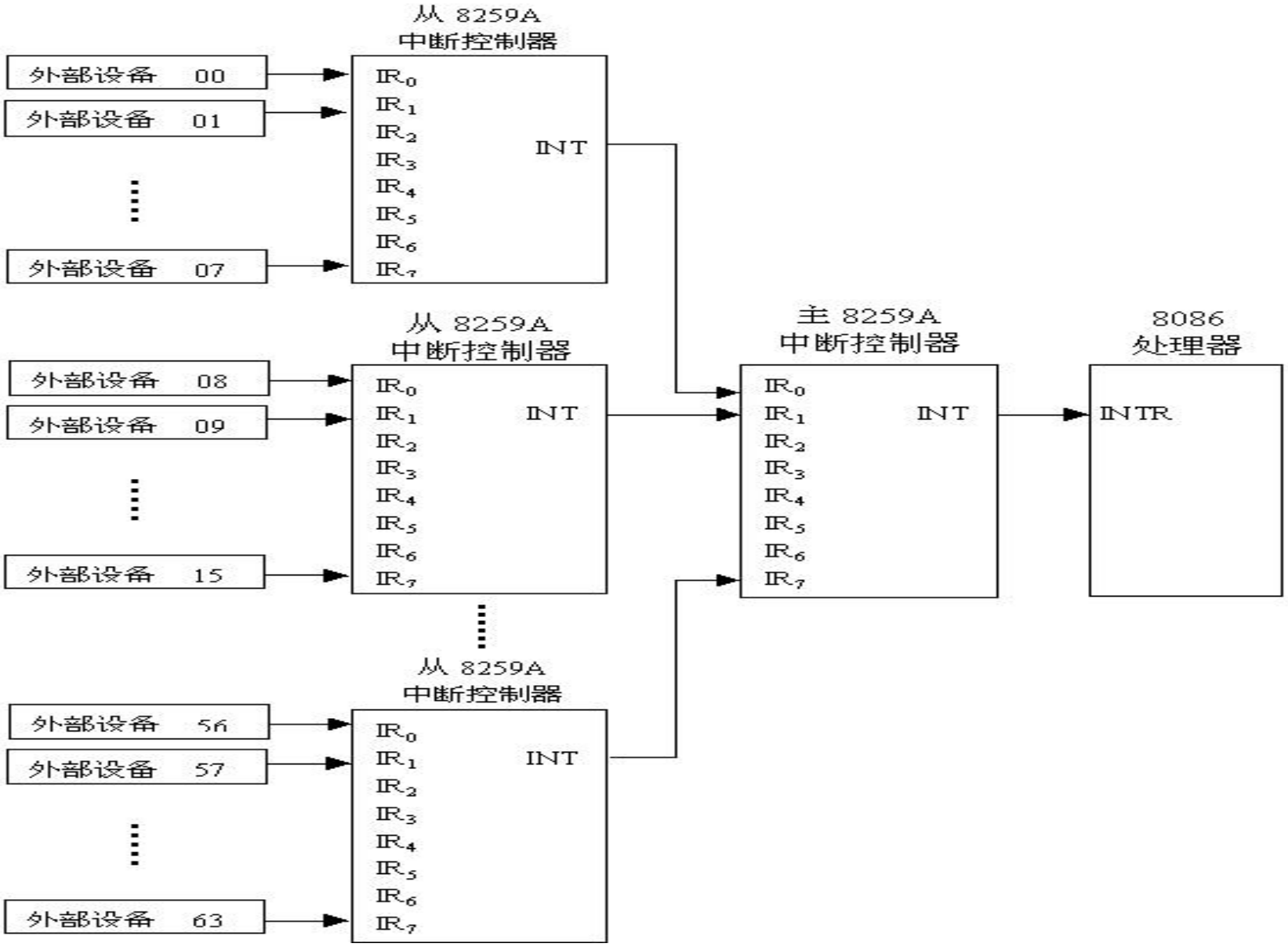
8259A 内部结构框图

（自学）8259A的引脚功能



8259A 引脚图

（自学）8259A芯片的级联



（自学）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：

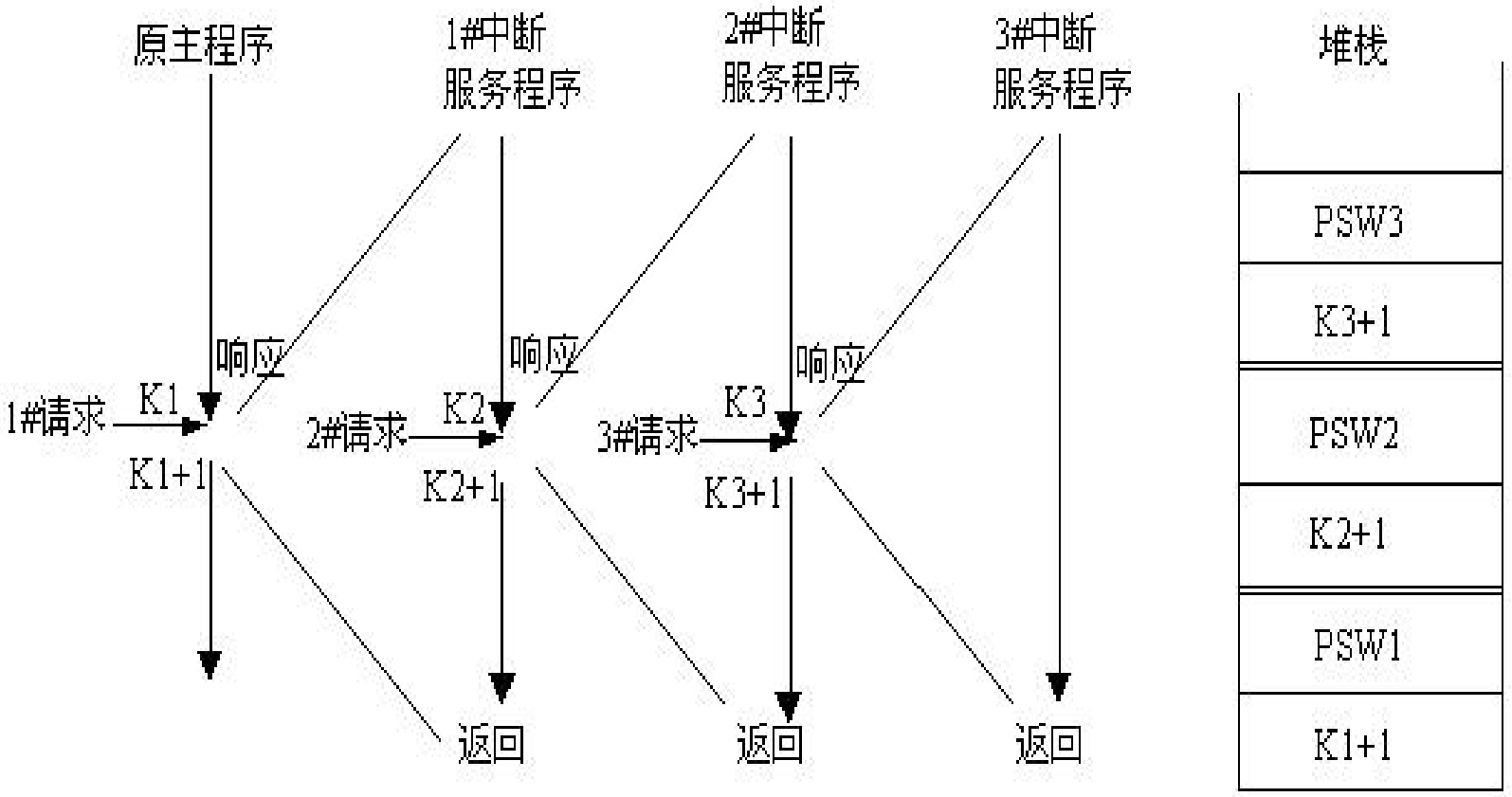
    在一个中断处理（即执行中断服务程序）过程中，若又有新的中断请求发生，而新中断优先级高于正在执行的中断，则应立即中止正在执行的中断服务程序，转取处理新的中断。这种情况为多重中断，也称中断嵌套。

  - 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

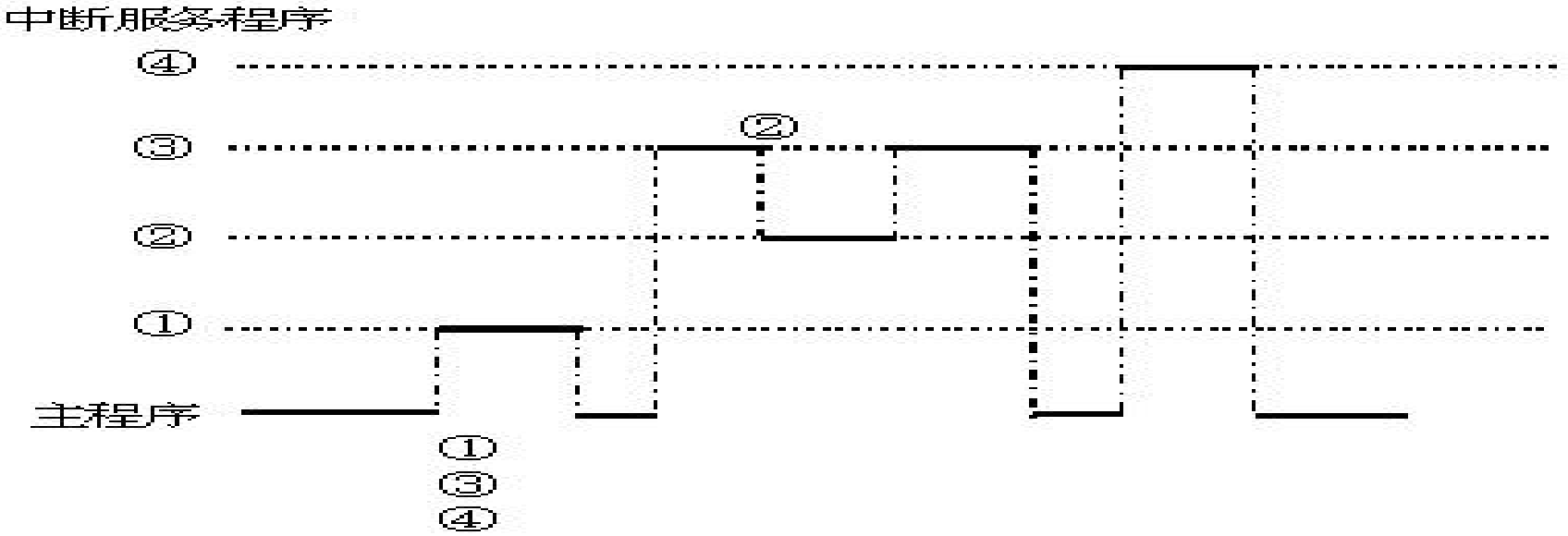
中断优先权的动态分配

- 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）



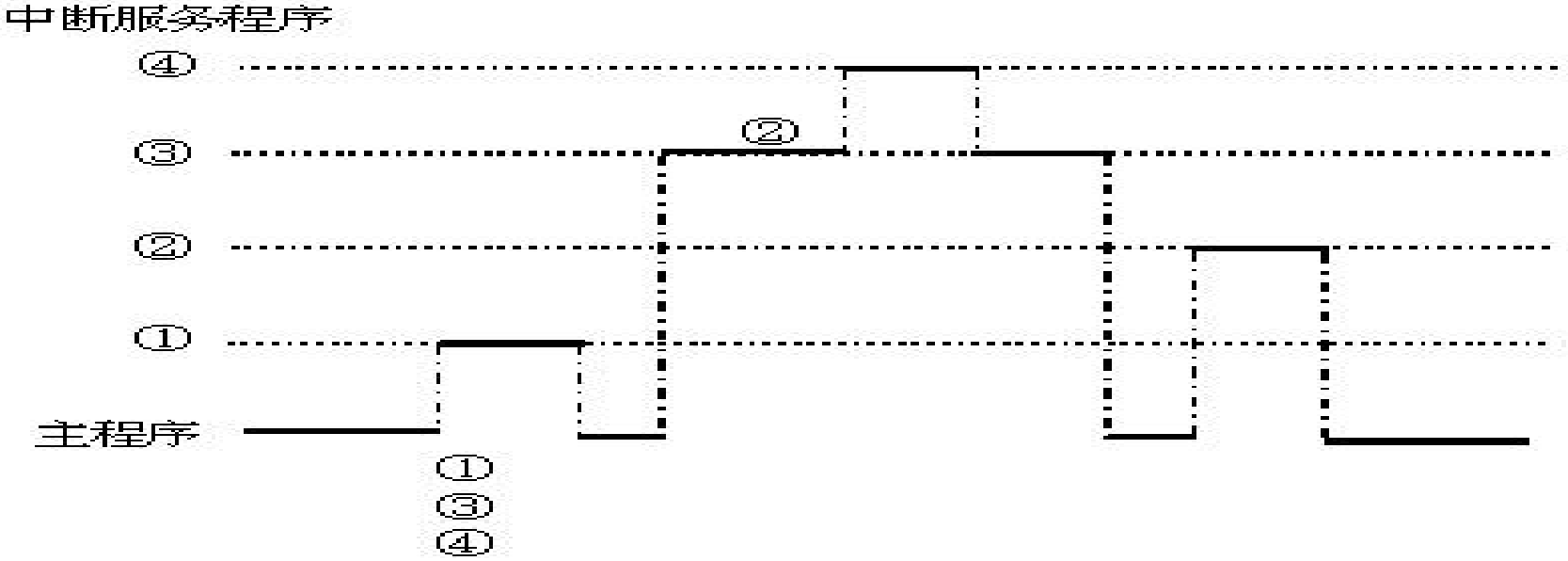
中断优先权的动态分配

- 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：

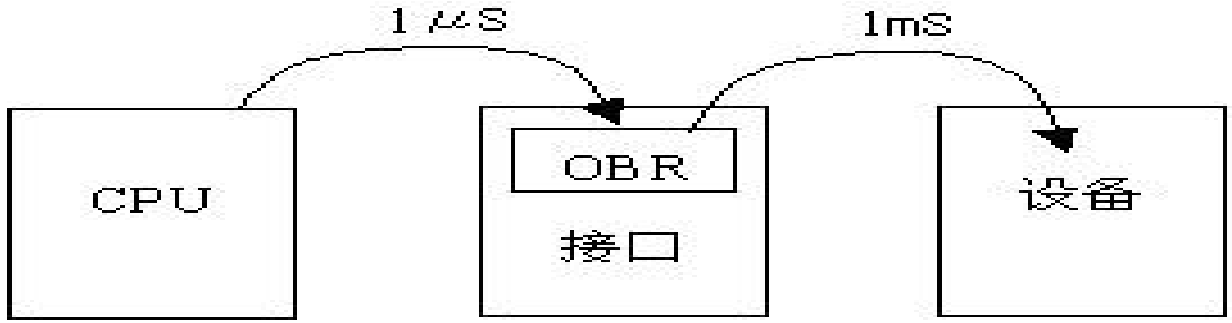
中断程序级别	屏蔽字			
	1 级	2 级	3 级	4 级
第 1 级	1	1	1	1
第 2 级	0	1	0	0
第 3 级	0	1	1	0
第 4 级	0	1	1	1

（假定 1 是屏蔽， 0 是开放）



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1μs。再由OBR输出到设备，需要1ms。设一条指令的执行时间为1μs(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) μs，约为每秒995个数据。

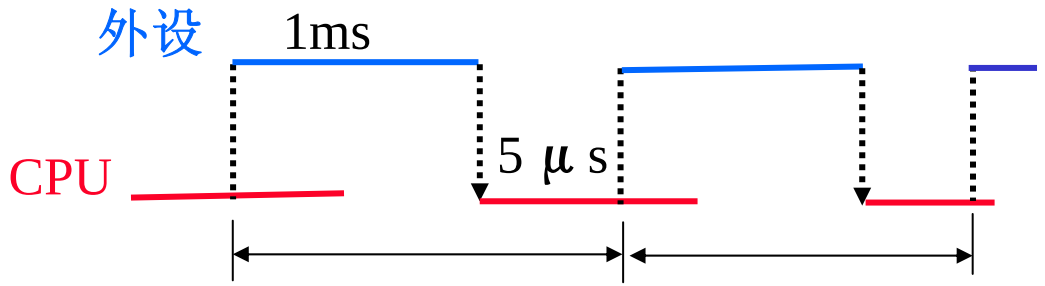
主机占用率=100%

（2）中断传送方式

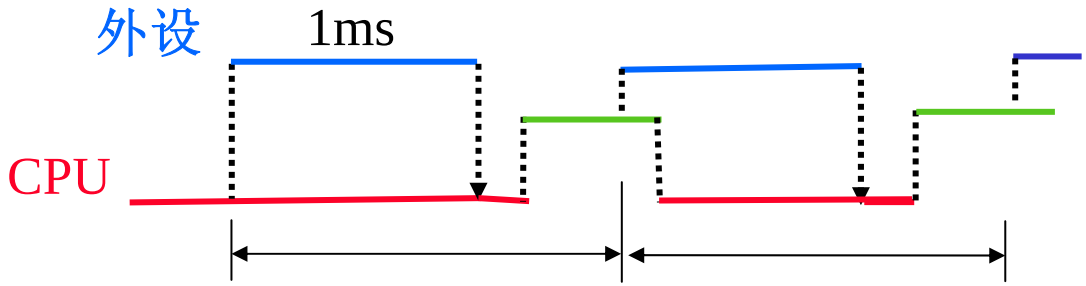
若中断服务程序有30条，在第20条启动设备，则：

数据传输率为：1/(1000+1+20)μs，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

DMA输入/出方式

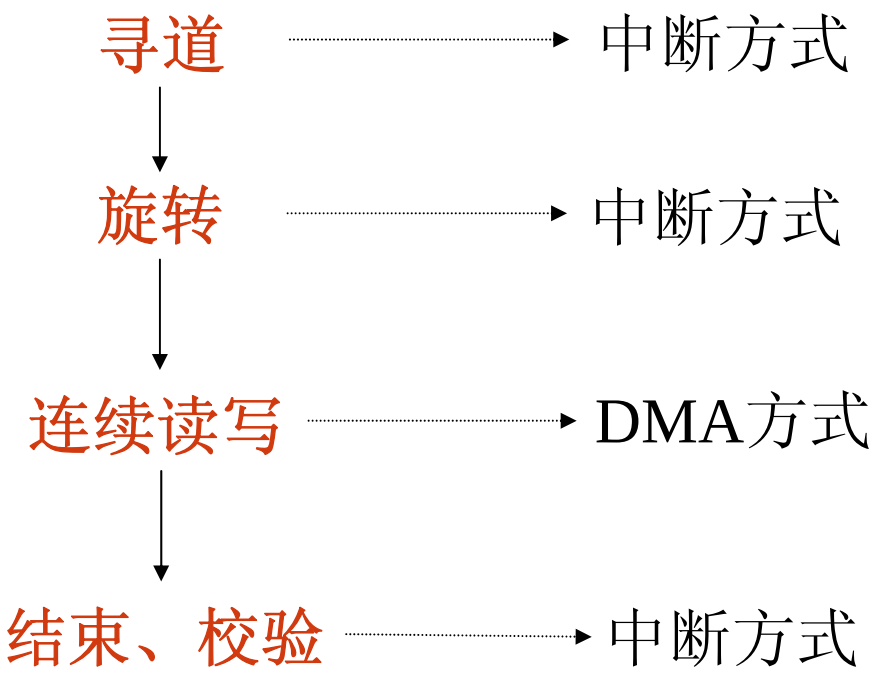
- DMA的全称
  - 直接存储器存取（Direct Memory Access）
- 为什么要引入DMA方式？
  - 程序直接控制方式受“踏步”现象的限制，效率低下，不适合高速设备和主机间的数据传送。
  - 中断控制方式虽比程序直接控制方式有效，CPU和外设有一定的并行度，但由于下列原因也不适合高速设备和主机间的数据传送。
    - 对I/O请求响应慢。每传送一个数据都要等待外设的中断请求，并增加许多中断响应和中断处理前、后的附加开销（保护断点、现场等），不能及时响应I/O请求。
    - 数据传送速度慢。数据传送由软件完成（由CPU执行相应的中断服务程序来完成)，速度慢 。

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。

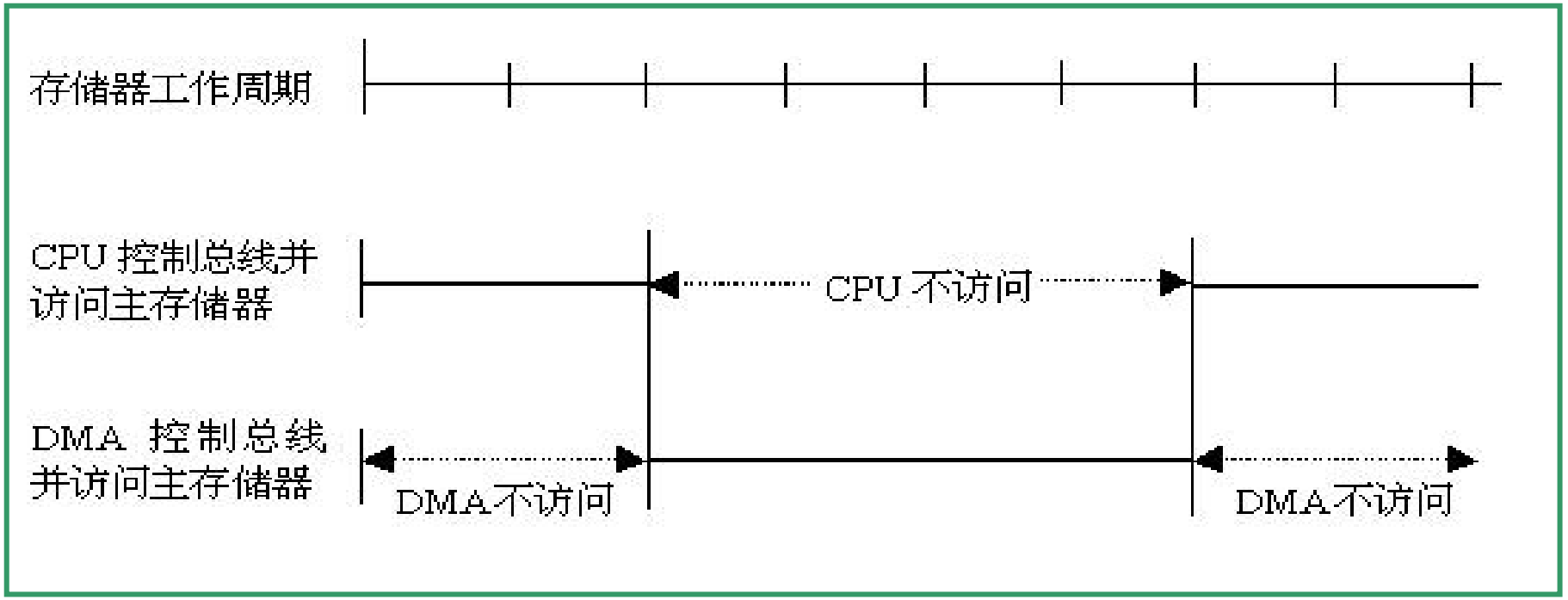
(2) 周期挪用(窃取)法(单字传送)

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。

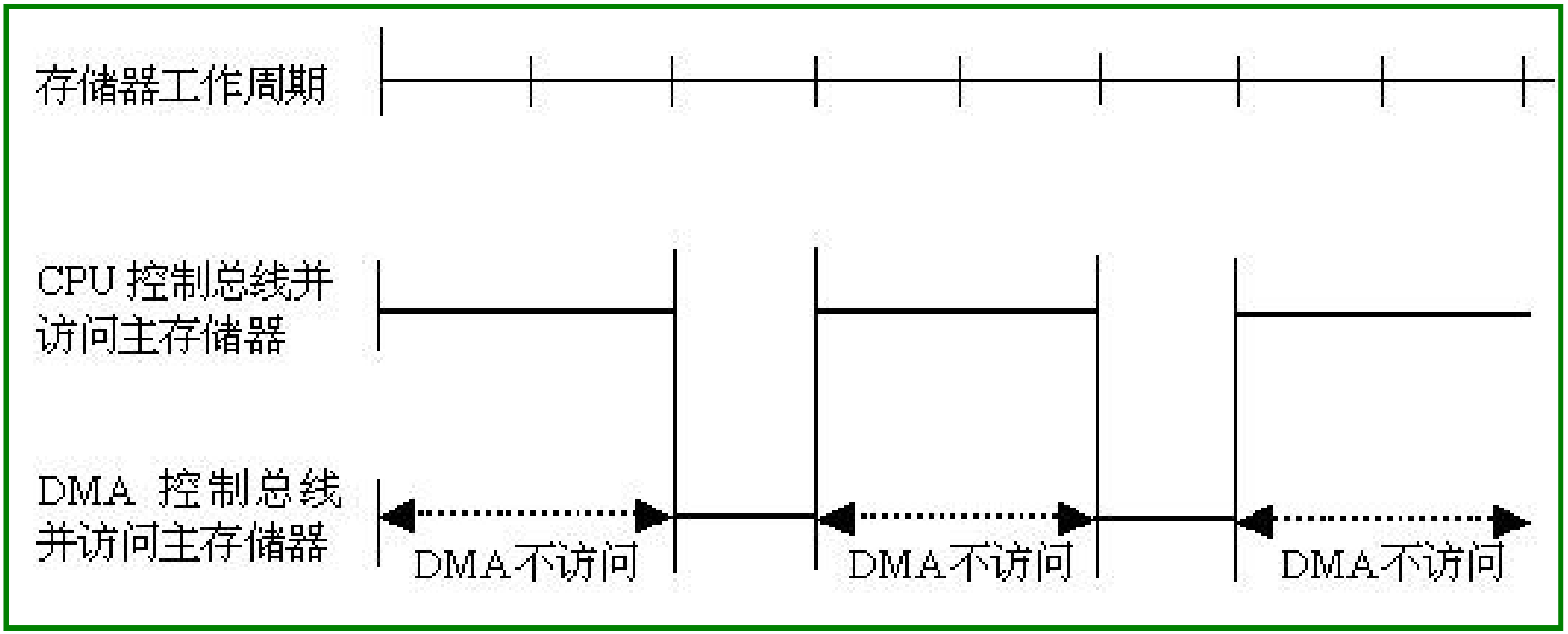
CPU停止法

- 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**

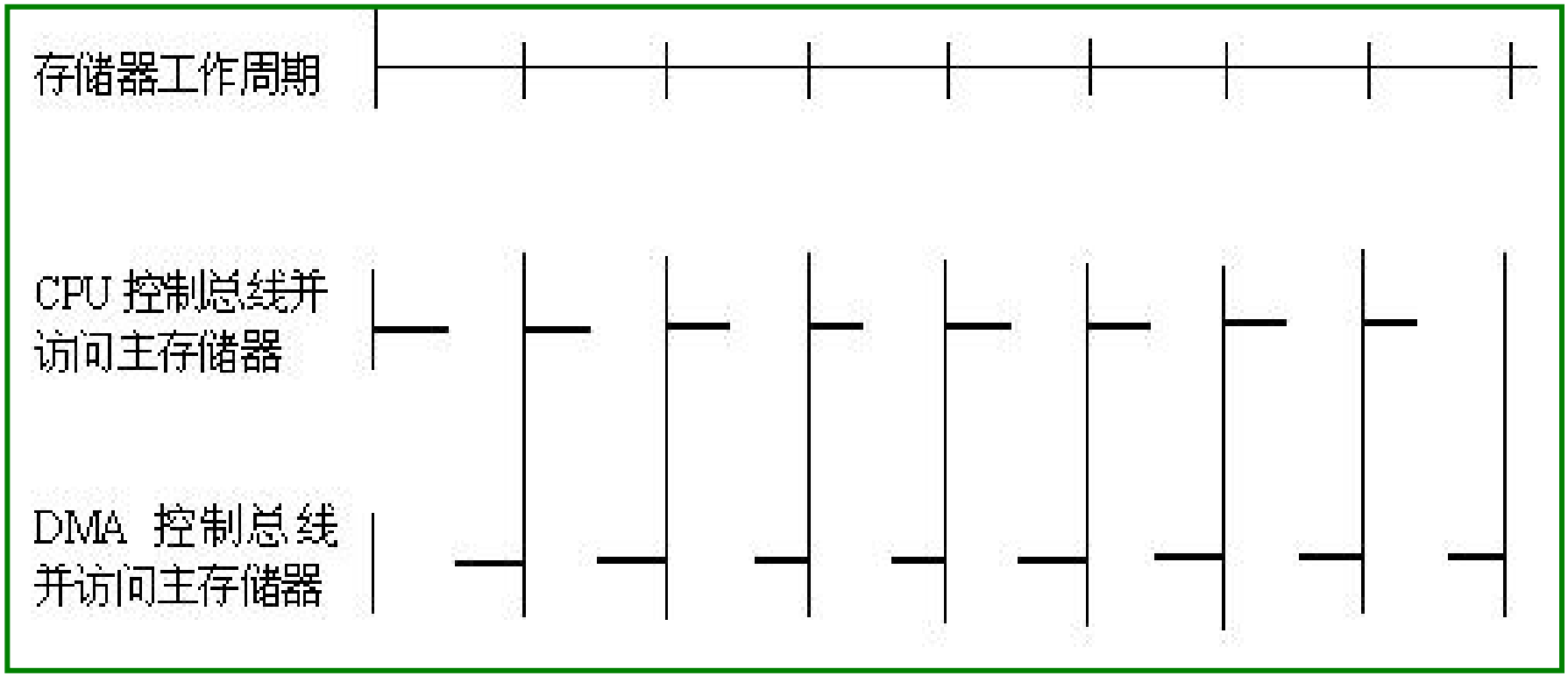
    此时，不会发生冲突，两者并行。

    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



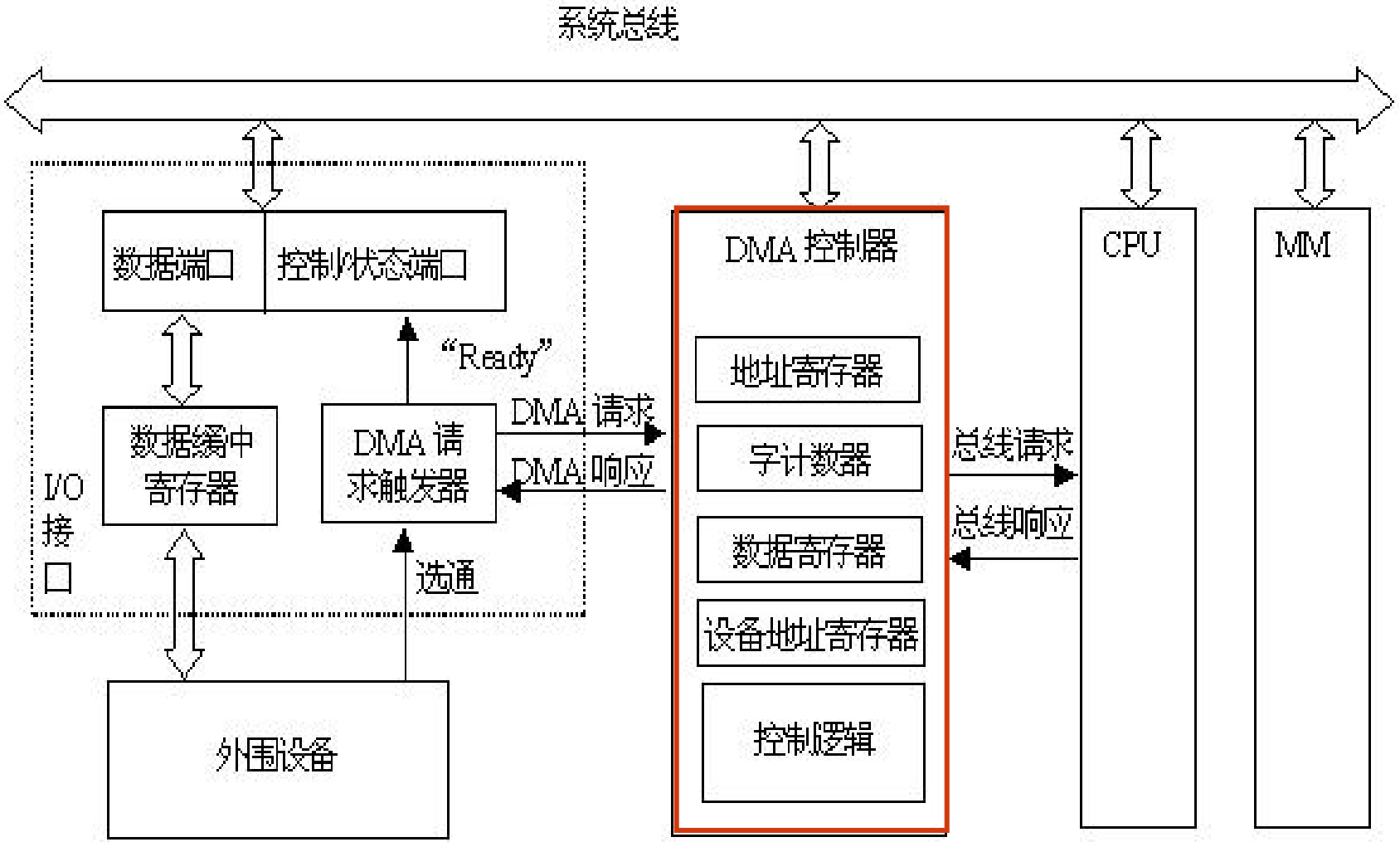
特点：适用于CPU工作周期比主存存取周期更长的情况。

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

  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。

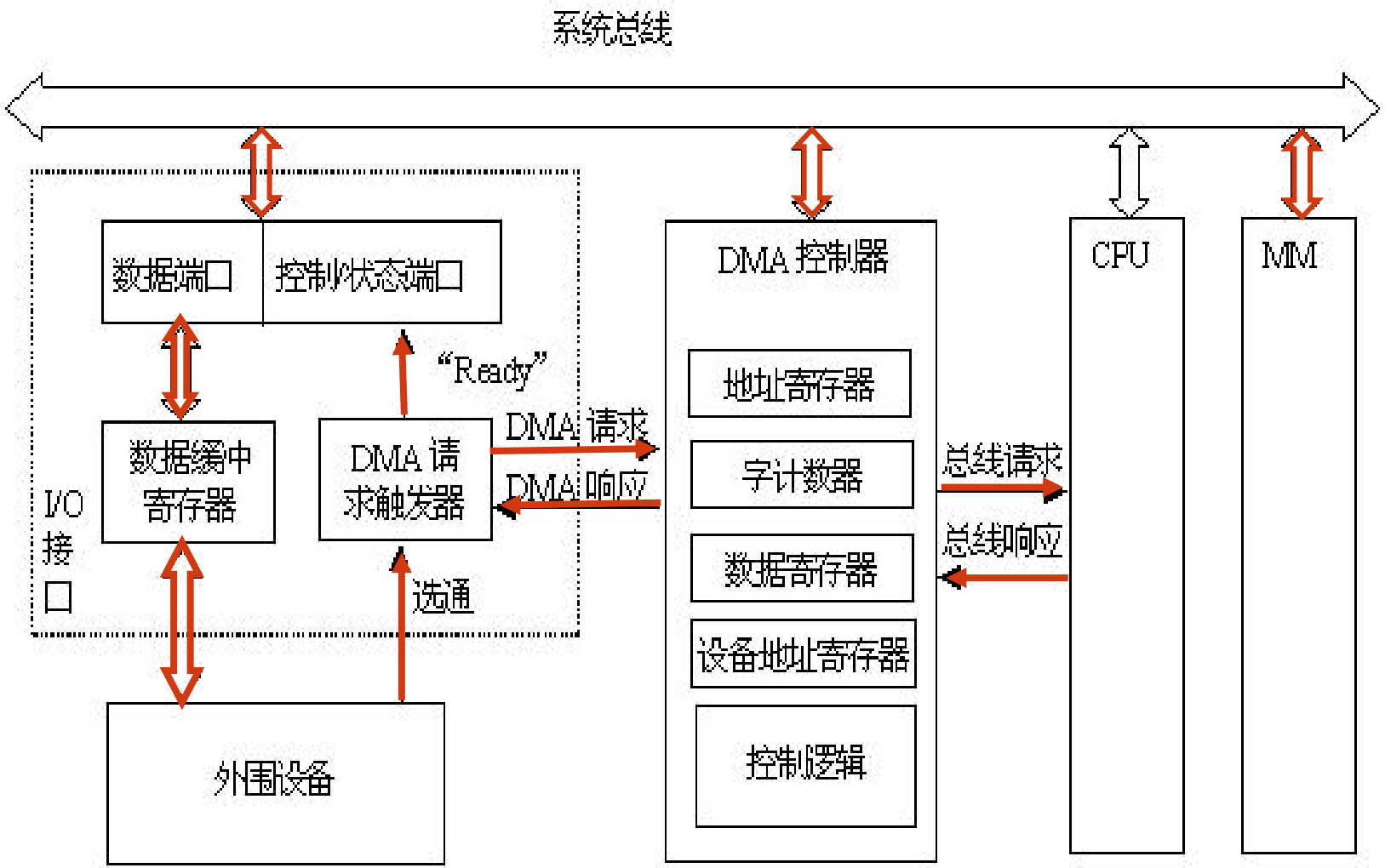
DMA控制器的初始化

- 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4MB/16B=250K次中断的速度；
  - 每秒钟用于中断的周期数为$250Kx500=125x10^6$；
  - 在一次传输中所消耗的处理器时间的百分比为：$125x10^6/(500x10^6)=25\%$；
  - 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 $25\%x5\%=1.25\%$ 。
- DMA传送：
  - 每个DMA传送将花$8KB/(4MB/Sec)=2x10^{-3}$秒；
  - 一秒钟有$1/(2x10^{-3})=500$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 $(1000+500)x500=750x10^3$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：

$$750x10^3/500x10^6=1.5x10^{-3}=0.15\%$$ 。

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

第三次作业

- 8.29
- 8.34
- 8.39（假设条件与8.18相同改为与8.38相同）
- 8.40 （假设条件与8.18相同改为与8.38相同）

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

- 总线是共享的传输介质和传输控制部件，用于在部件或设备间传输数据
- 总线可能在芯片内、芯片之间、板卡之间和计算机系统之间
- I/O总线是I/O控制器与主机之间传输数据的一组公用信号线，它们在物理上与主板扩展槽中插入的扩展卡（I/O控制器）直接连接。
- 总线可以采用“同步”或“异步”方式进行定时。
  - 同步总线用“时钟”信号定时；异步总线用“握手信号”定时
  - 可以结合同步和异步方式进行半同步定时通信
  - 可以把一个总线事务分离成两个事务，在从设备准备数据时释放总线
- 总线的裁决：有集中和分布两类裁决方式
  - 分布裁决：自举裁决、冲突检测
  - 集中裁决：菊花链、独立请求并行判优
- 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