

第7章 输入/输出系统

7.1 I/O系统性能与CPU性能

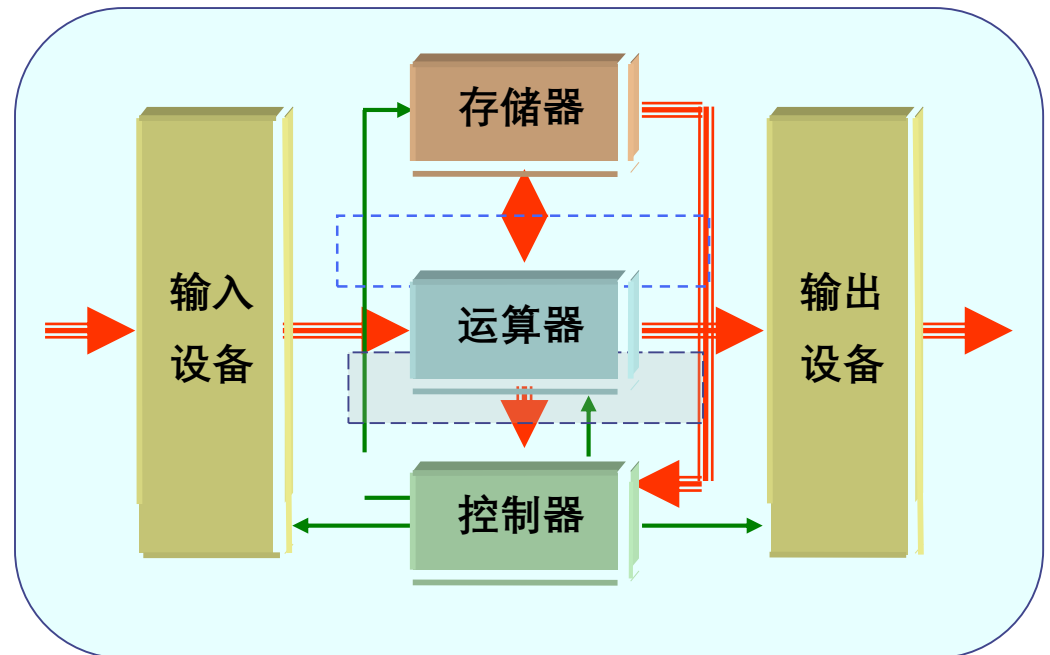
7.2 I/O系统的可靠性、可用性和可信性

7.3 廉价磁盘冗余阵列RAID

7.4 总线

7.5 通道处理机

7.6 I/O与操作系统



7.1 I/O系统性能与CPU性能

1. 输入/输出系统，包括：

- I/O设备
- I/O设备与处理机的连接
- I/O软件

2. I/O系统是计算机系统中的一个重要组成部分

- 完成计算机与外界的信息交换
- 给计算机提供大容量的外部存储器

3. 按照主要完成的工作进行分类：

- 存储I/O系统（本章内容）
- 通信I/O系统

7.1 I/O系统性能与CPU性能

4. 人们对I/O系统的作用和性能没有给予足够的重视

➤ 人们更多地关注：CPU的性能

许多人甚至认为CPU的速度就是计算机的速度。

➤ I/O设备通常被称为外围设备。

(外围的就似乎没那么重要了)

5. 系统的响应时间 (衡量计算机系统的一个更好的指标)

➤ 从用户输入命令开始，到得到结果所花费的时间。

➤ 由两部分构成：

- I/O系统的响应时间
- CPU的处理时间

7.1 I/O系统性能与CPU性能

6. 误区：使用多进程技术可以忽略I/O性能对系统性能的影响

- 多进程技术**只能提高系统吞吐率**，**并不能够减少系统响应时间**
- 进程切换时可能需要**增加I/O操作**
- 可切换的进程数量有限，当I/O处理较慢时，仍然会导致CPU处于空闲状态。

7.1 I/O系统性能与CPU性能

例7.1 假设一台计算机的I/O处理时间占响应时间的10%，当I/O性能保持不变，而对CPU的性能分别提高10倍和100倍时，该计算机系统的总体性能会发生什么样的变化？

解：假设改进前程序的执行时间为1个单位时间。

如果CPU的性能提高10倍，程序的执行时间减少为：

$$(1-10\%)/10+10\%=0.19$$

即整机性能只能提高到原来的约5倍，约50%的CPU性能被浪费在I/O处理上。

如果CPU的性能提高100倍，程序的执行时间减少为：

$$(1-10\%)/100+10\%=0.109$$

整机性能只能提高约10倍，约90%的性能被浪费在没有改进的I/O处理上。

7.1 I/O系统性能与CPU性能

7. 评价I/O系统性能的参数：

- 连接特性：哪些I/O设备可以和计算机系统相连接
- I/O系统的容量：I/O系统可以容纳的I/O设备数
- 响应时间和吞吐率等

8. 另一种衡量I/O系统性能的方法：

- 考虑I/O操作对CPU的打扰情况：

即考查某个进程在执行时，由于其他进程的I/O操作，使得该进程的执行时间增加了多少。

第7章 输入/输出系统

7.1 I/O系统性能与CPU性能

7.2 I/O系统的可靠性、可用性和可信性

7.3 廉价磁盘冗余阵列RAID

7.4 总线

7.5 通道处理机

7.6 I/O与操作系统

7.2 I/O系统的可靠性、可用性和可信性

1. 处理器性能已经很高，人们更加关注系统可靠性
2. 反映外设的性能的参数有：
 - 可靠性 (Reliability)
 - 可用性 (Availability)
 - 可信性 (Dependability)
3. 系统的可靠性：系统从某个初始参考点开始一直连续提供服务的能力。
 - 用平均无故障时间MTTF来衡量 (Mean Time To Failure)
 - MTTF的倒数就是系统的失效率
 - 如果系统中每个模块的生存期服从指数分布，则：
系统整体的失效率是各部件的失效率之和

7.2 I/O系统的可靠性、可用性和可信性

4. **系统的可用性**：系统正常工作的时间在连续两次正常服务间隔时间中所占的比率。

$$\text{可用性} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

平均无故障时间

平均修复时间
Mean Time To Repair

- **MTTF+MTTR：平均失效间隔时间MTBF**
(Mean Time Between Failure)

5. **系统的可信性**：服务的质量。即在多大程度上可以合理地认为服务是可靠的。 (不可以度量)

7.2 I/O系统的可靠性、可用性和可信性

例7.2 假设磁盘子系统的组成部件和它们的MTTF如下：

- (1) 磁盘子系统由10个磁盘构成，每个磁盘的MTTF为1000000小时；
- (2) 1个SCSI接口控制器，其MTTF为500000小时；
- (3) 1个不间断电源，其MTTF为200000小时；
- (4) 1个风扇，其MTTF为200000小时；
- (5) 1根SCSI连线，其MTTF为1000000小时。

假定：每个部件的生存期服从指数分布，同时假定各部件的故障是相互独立的，**求整个系统的MTTF。**

7.2 I/O系统的可靠性、可用性和可信性

解：整个系统的失效率为：

$$\text{系统失效率} = 10 \times \frac{1}{1000000} + \frac{1}{500000} + \frac{1}{200000} + \frac{1}{200000} + \frac{1}{1000000} = \frac{23}{1000000}$$

系统的MTTF为系统失效率的倒数，即：

$$\text{MTTF} = \frac{1000000}{23} = 43500 \text{小时}$$

即将近5年。

7.2 I/O系统的可靠性、可用性和可信性

6. 提高系统组成部件可靠性的方法

➤ 有效构建方法 (valid construction)

- 在构建系统的过程中消除故障隐患，这样建立起来的系统就不会出现故障。

➤ 纠错方法 (error correction)

- 在系统构建中采用容错的方法。这样即使出现故障，也可以通过容错信息保证系统正常工作。

第7章 输入/输出系统

7.1 I/O系统性能与CPU性能

7.2 I/O系统的可靠性、可用性和可信性

7.3 廉价磁盘冗余阵列RAID

7.4 总线

7.5 通道处理机

7.6 I/O与操作系统

7.3 廉价磁盘冗余阵列RAID

1. **磁盘阵列DA** (Disk Array) : 使用多个磁盘 (包括驱动器) 的组合来代替一个大容量的磁盘。
 - 多个磁盘并行工作
 - 以条带为单位把数据均匀地分布到多个磁盘上 (交叉存放)
 - 条带存放可以使多个数据读/写请求并行地被处理, 从而提高总的I/O性能。
2. 这里**并行性**有两方面的含义:
 - 多个独立的请求可以由多个盘来并行地处理 (减少了I/O请求的排队等待时间)
 - 如果一个请求访问多个块, 就可以由多个盘合作来并行处理 (提高了单个请求的数据传输率)

7.3 廉价磁盘冗余阵列RAID

3. **问题：**阵列中磁盘数量的增加会导致磁盘阵列可靠性的下降

- 如果使用了N个磁盘构成磁盘阵列，那么整个阵列的可靠性将降低为单个磁盘的 $1/N$ 。

➤ **解决方法：**在磁盘阵列中设置冗余信息盘。

当单个磁盘失效时，丢失的信息可以通过冗余盘中的信息重新构建。

7.3 廉价磁盘冗余阵列RAID

4. 廉价磁盘冗余阵列RAID

Redundant Arrays of Inexpensive Disks

- 1988年，Patterson教授首先提出
- 磁盘冗余阵列RAID
- 简称磁盘阵列技术DA

5. 大多数磁盘阵列的组成可以用两个特征来区分：

- 冗余数据的计算方法以及在磁盘阵列中的存放方式
- 数据交叉存放的粒度：细粒度，粗粒度

7.3 廉价磁盘冗余阵列RAID

- **细粒度磁盘阵列：**把数据分割成相对较小的单位交叉存放。
 - ✓ **优点：**所有I/O请求都能够获得很高的数据传输率。
 - ✓ **缺点：**在任何时间，都只有一个逻辑上的I/O在处理当中，而且所有的磁盘都会因为为每个请求进行定位而浪费时间。
- **粗粒度磁盘阵列：**把数据以相对较大的单位交叉存放。
 - ✓ 多个较小规模的请求可以同时得到处理。
 - ✓ 对于较大规模的请求又能获得较高的传输率。

7.3 廉价磁盘冗余阵列RAID

6. 在磁盘阵列中设置冗余需要解决以下两个问题：

➤ 如何计算冗余信息？

- 大多都是采用奇偶校验码；
- 也有采用汉明码（Hamming code）或Reed-Solomon码。

➤ 如何把冗余信息分布到磁盘阵列中的各个盘？

有两种方法：

- 把冗余信息集中存放在少数的几个盘中。
- 把冗余信息均匀地存放到所有的盘中。

（能避免出现热点问题）

7. RAID的分级及其特性

7.3 廉价磁盘冗余阵列RAID

RAID级别	可以容忍的故障个数以及当数据盘为8个时，所需要的检测盘的个数	优点	缺点	公司产品
0 非冗余，条带存放	0个故障； 0个检测盘	没有空间开销	没有纠错能力	广泛应用
1 镜像	1个故障； 8个检测盘	不需要计算奇偶校验，数据恢复快，读数据快。而且其小规模写操作比更高级别的RAID快	检测空间开销最大 (即需要的检测盘最多)	EMC, HP (Tandem), IBM
2 存储器式ECC	1个故障； 4个检测盘	不依靠故障盘进行自诊断	检测空间开销的级别是 $\log_2 m$ 级 (m 为数据盘的个数)	没有
3 位交叉奇偶校验	1个故障； 1个检测盘	检测空间开销小 (即需要的检测盘少)，大规模读写操作的带宽高	对小规模、随机的读写操作没有提供特别的支持	外存概念

7. RAID的分级及其特性

RAID级别	可以容忍的故障个数以及当数据盘为8个时，所需要的检测盘的个数	优点	缺点	公司产品
4 块交叉奇偶校验	1个故障； 1个检测盘	检测空间开销小， 小规模读操作 带宽更高	校验盘是小规模 写的瓶颈	网络设备
5 块交叉分布奇偶校验	1个故障； 1个检测盘	检测空间开销小， 小规模读写操作 带宽更高	小规模写操作 需要访问磁盘4次	广泛应用
6 P+Q双奇偶校验	2个故障； 2个检测盘	具有容忍2个 故障的能力	小规模写操作需要 访问磁盘6次，检 测空间开销加倍 (与RAID3、4、5比 较)	网络设备

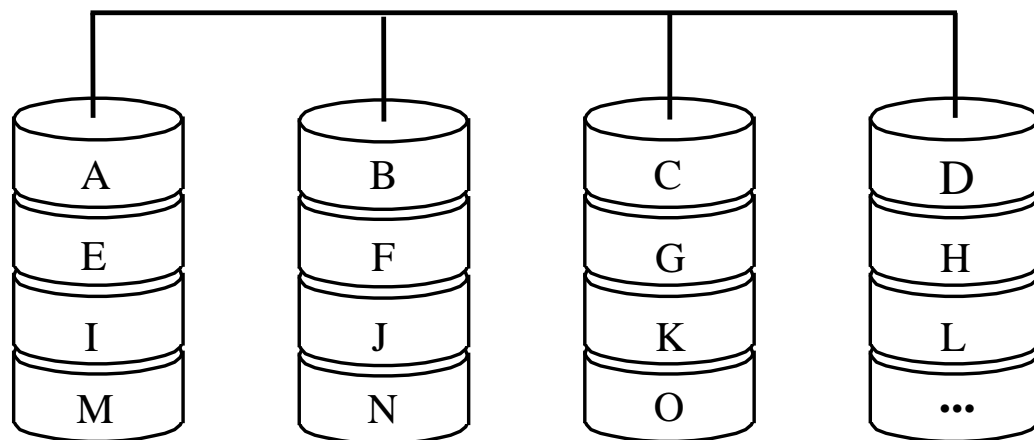
7.3 廉价磁盘冗余阵列RAID

8. 有关RAID的几个问题

- **关键问题：** 如何发现磁盘的故障
 - 磁盘技术提供了故障检测操作的信息
- **设计的另一个问题**
 - 如何减少平均修复时间MTTR
 - 典型的做法：在系统中增加热备份盘
- **热交换技术**
 - 与热备份盘相关的一种技术
 - 允许系统在不关机的情况下更换设备

7.3 廉价磁盘冗余阵列RAID 0

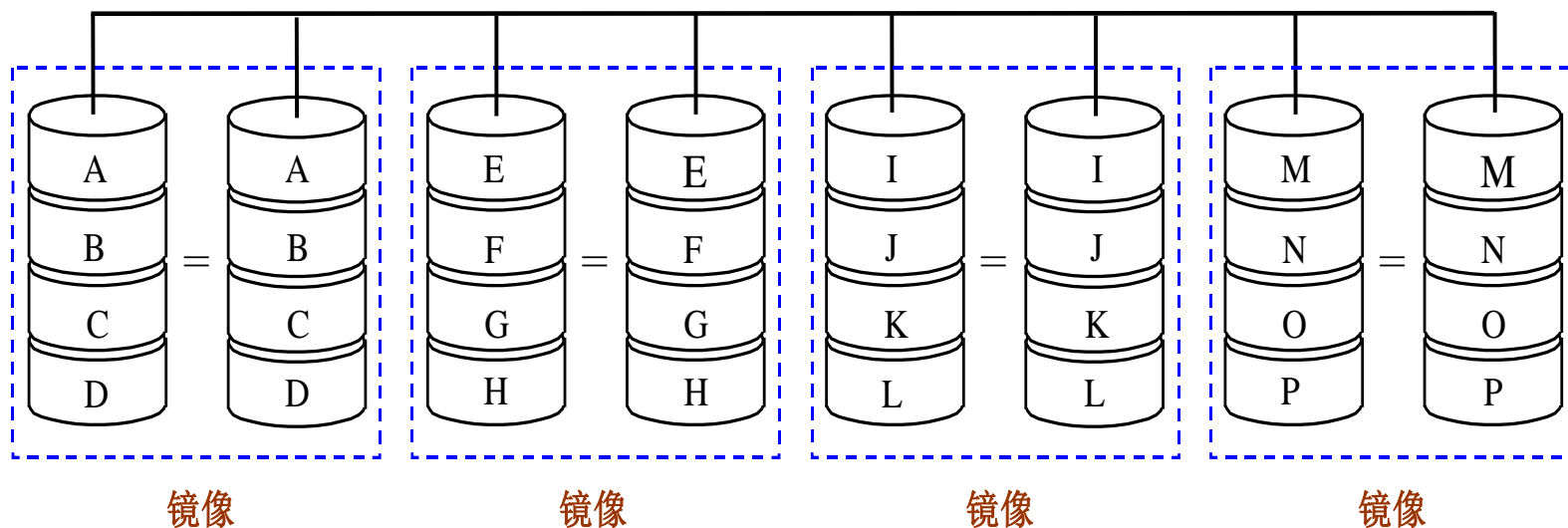
- **非冗余阵列**：无冗余信息。
- 严格地说，它不属于RAID系列。
- 把数据切分成条带，以条带为单位交叉地分布存放到多个磁盘中。



7.3 廉价磁盘冗余阵列RAID 1

- 亦称**镜像盘**，使用**双备份磁盘**

➤ 每当把数据写入磁盘时，将该数据也写入其镜像盘，形成信息的两个副本。



7.3 廉价磁盘冗余阵列RAID 1

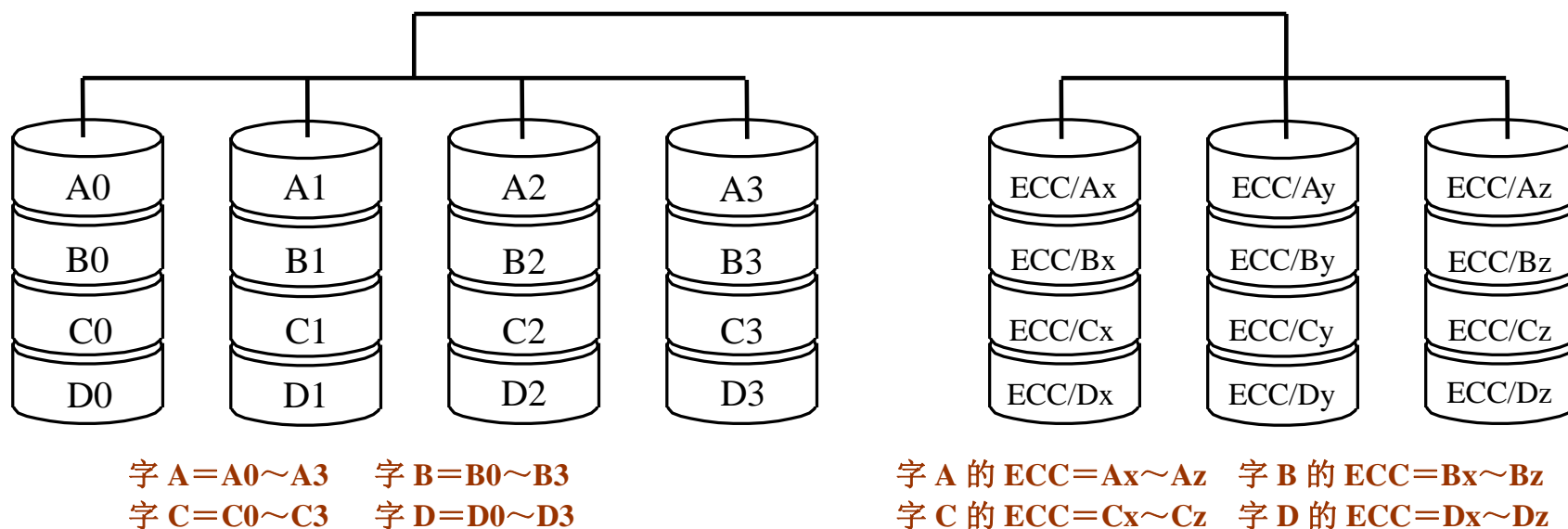
- RAID1的特点

- 能实现快速的读取操作。
- 写性能由写性能最差的磁盘决定。相对以后各级RAID来说，RAID1的写速度较快。
- 可靠性很高，数据的恢复很简单。
- 最昂贵的解决方法，物理磁盘空间是逻辑磁盘空间的两倍。

7.3 廉价磁盘冗余阵列RAID 2

- 存储器式的磁盘阵列（按汉明纠错码的思路构建）

- 含4个数据盘的RAID2



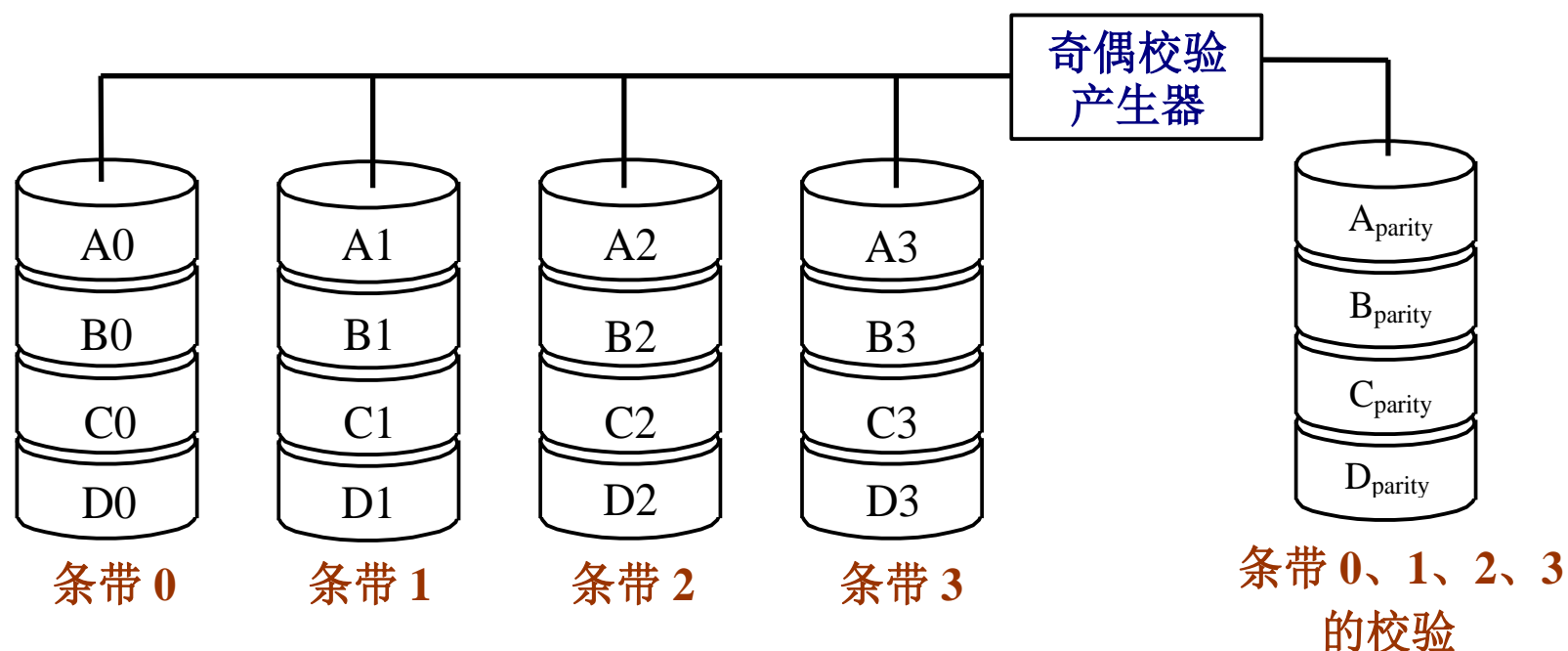
7.3 廉价磁盘冗余阵列RAID 2

● RAID2的特点

- 每个数据盘存放所有数据字的一位（位交叉存放）
- 各个数据盘上的相应位计算汉明校验码，编码位被存放在多个校验（ECC）磁盘的对应位上。
- 冗余盘是用来存放汉明码的，其个数为 $\log_2 m$ 级。
m：数据盘的个数（也就是数据字的位数）
- 并未被广泛应用，目前还没有商业化产品。

7.3 廉价磁盘冗余阵列RAID 3

- 位交叉奇偶校验盘阵列



7.3 廉价磁盘冗余阵列RAID 3

● RAID3的特点

➤ 采用奇偶校验

➤ 写数据时

- 为每行数据形成奇偶校验位并写入校验盘

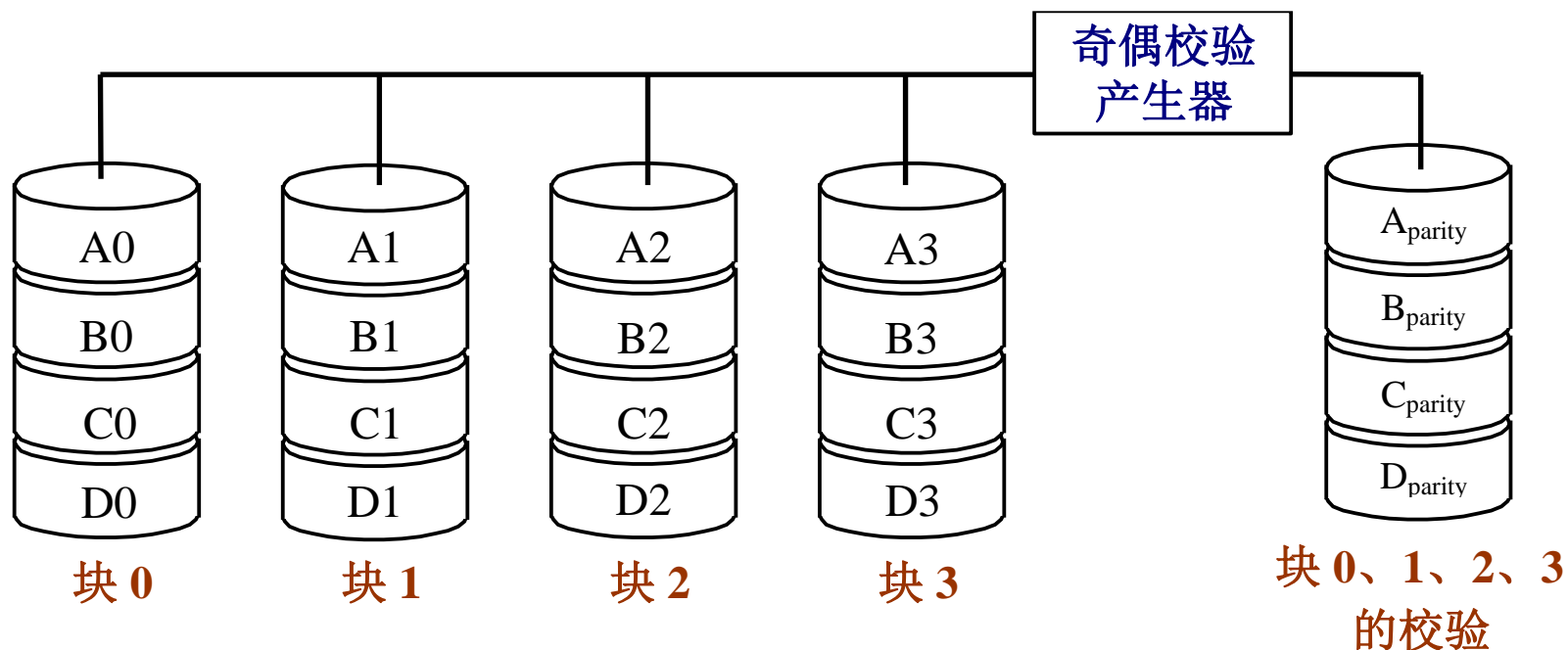
➤ 读出数据时

- 如果控制器发现某个磁盘出故障，就可以根据故障盘以外的所有其他盘中的正确信息恢复故障盘中的数据。（通过异或运算实现）
- 细粒度的磁盘阵列，即采用的条带宽度较小。
 - 可以是1个字节或1位
- 只需要一个校验盘，校验空间开销比较小。

7.3 廉价磁盘冗余阵列RAID 4

- 块交叉奇偶校验磁盘阵列
- 采用比较大的条带，以块为单位进行交叉存放和计算奇偶校验。

➤ **实现目标：**能同时处理多个小规模访问请求



7.3 廉价磁盘冗余阵列RAID 4

- RAID4读写特点

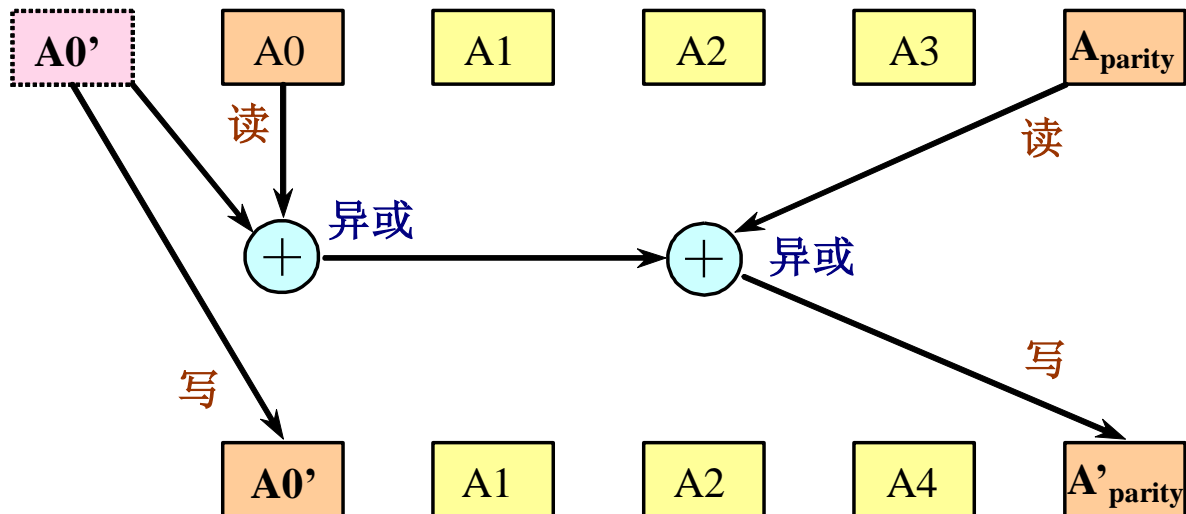
- 读取操作

- 每次只需访问数据所在的磁盘。
 - 仅在该磁盘出现故障时，才会去读校验盘，并进行数据的重建。

- 写入操作

- 假定: 有4个数据盘和一个冗余盘。
 - 写数据需要2次磁盘读和2次磁盘写操作。

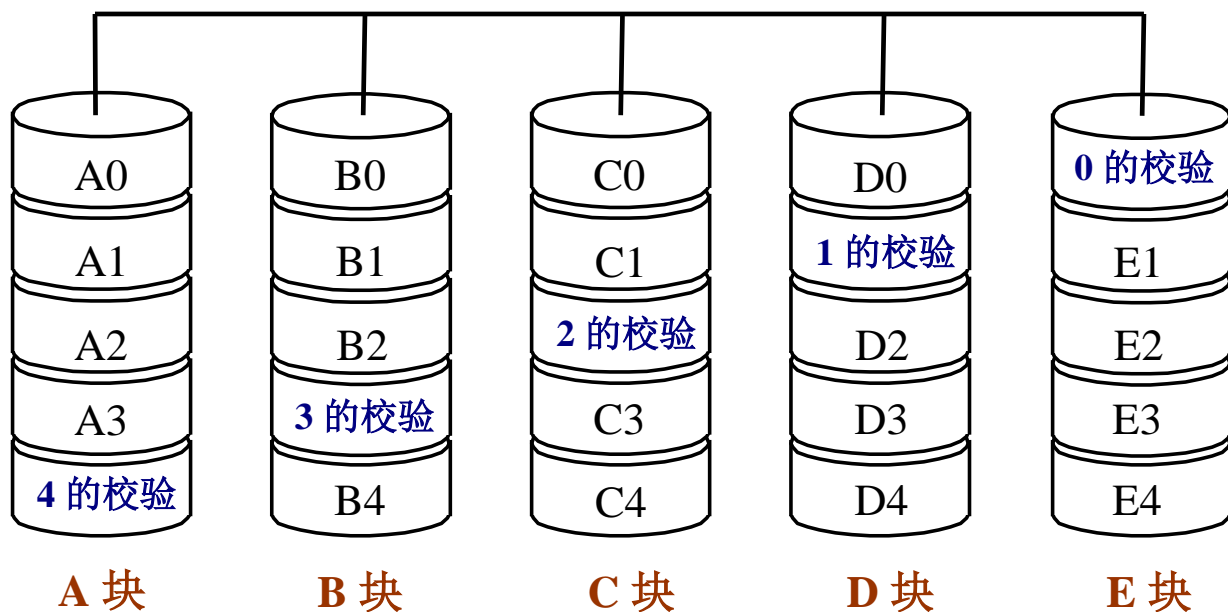
7.3 廉价磁盘冗余阵列RAID 4



- RAID4能有效地处理小规模访问，快速处理大规模访问，校验空间开销比较小。

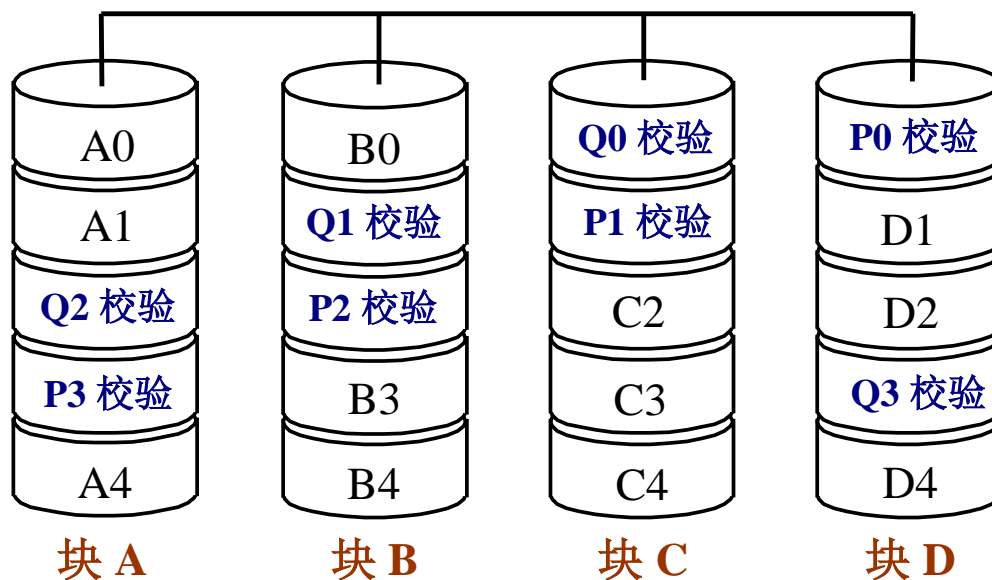
7.3 廉价磁盘冗余阵列RAID 5

- 块交叉分布奇偶校验磁盘阵列
- 数据以块交叉的方式存于各盘，无专用冗余盘，奇偶校验信息均匀分布在所有磁盘上。



7.3 廉价磁盘冗余阵列RAID 6

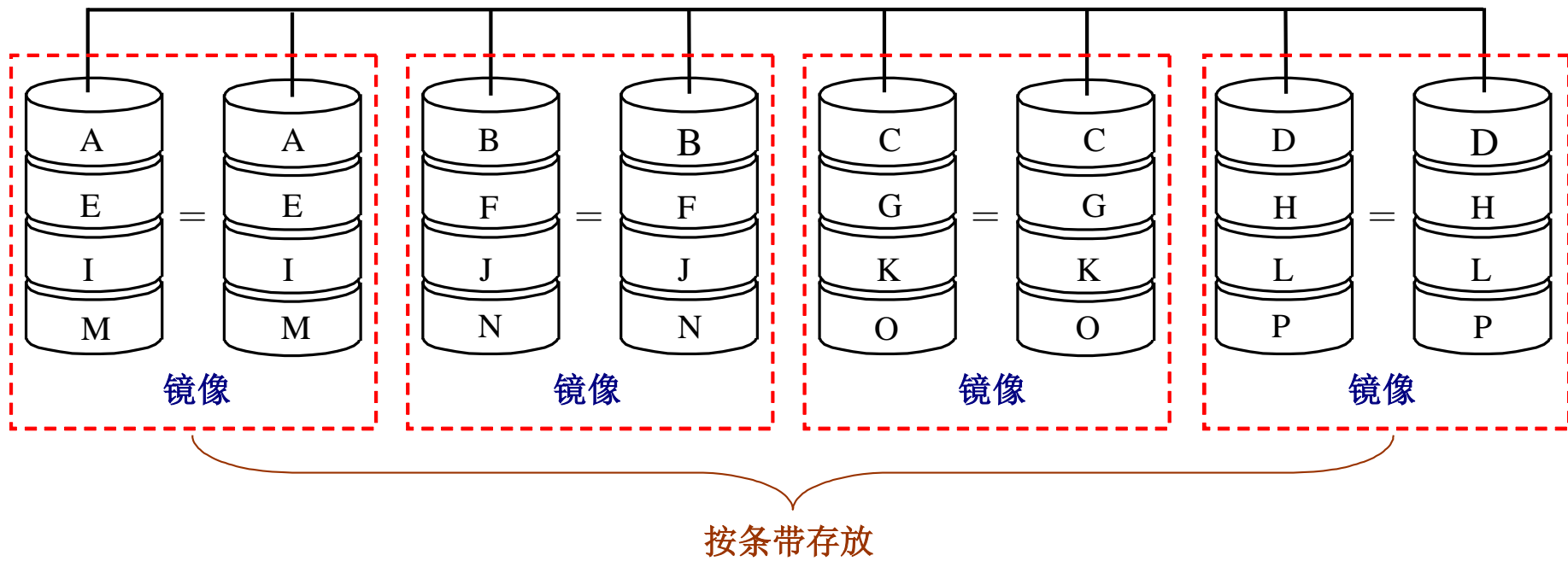
- P + Q双校验磁盘阵列
- 特点：校验空间开销是RAID5的两倍
容忍两个磁盘出错



7.3 廉价磁盘冗余阵列RAID 10 与 01

- RAID10又称为RAID1+0

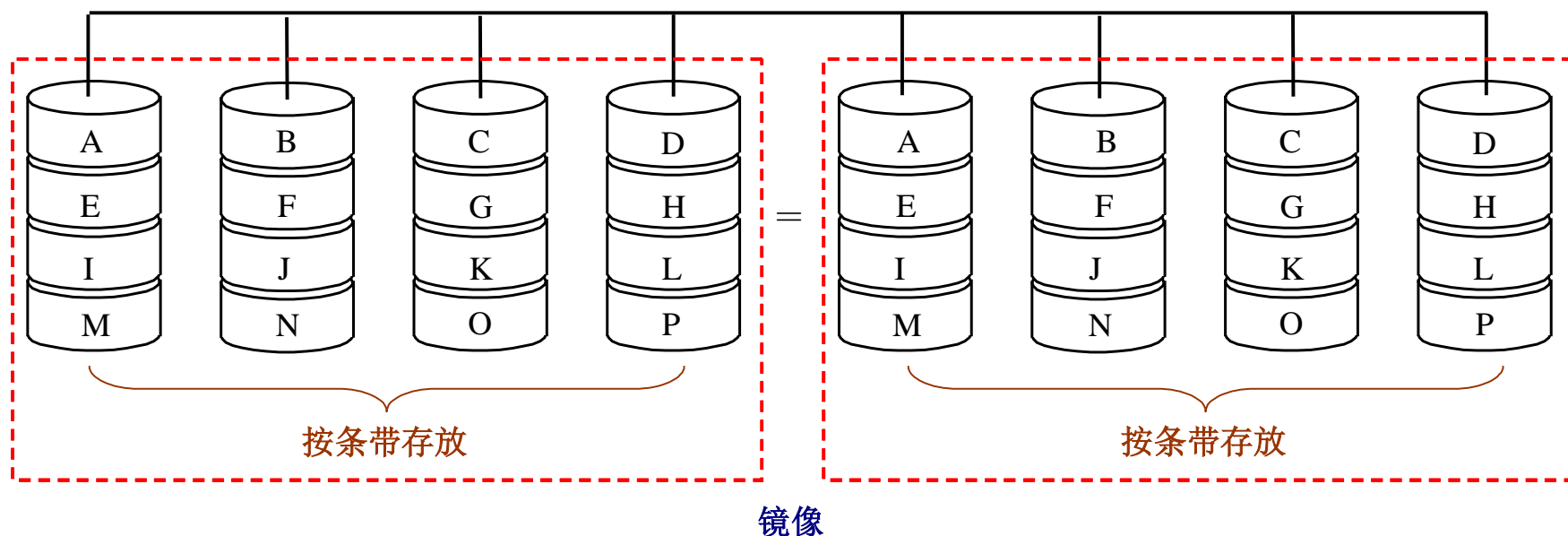
先进行镜像（RAID1），再进行条带存放（RAID0）



7.3 廉价磁盘冗余阵列RAID 10 与 01

- RAID01又称为RAID0+1

先进行条带存放（RAID0），再进行镜像（RAID1）



7.3 廉价磁盘冗余阵列RAID 实现与发展

1. 实现盘阵列的方式主要有三种：

➤ **软件方式：**阵列管理软件由主机来实现

- 优点：成本低
- 缺点：过多地占用主机时间，且带宽指标上不去。

➤ **阵列卡方式：**把RAID管理软件固化在I/O控制卡上，从而可不占用主机时间，一般用于工作站和PC机。

➤ **子系统方式：**一种基于通用接口总线的开放式平台，可用于各种主机平台和网络系统。

7.3 廉价磁盘冗余阵列RAID 实现与发展

2. 磁盘阵列技术研究的主要热点问题

- 新型阵列体系结构
- RAID结构与其所记录文件特性的关系
- 在RAID冗余设计中，综合平衡性能、可靠性和开销的问题
- 超大型磁盘阵列在物理上如何构造和连接

第7章 输入/输出系统

7.1 I/O系统性能与CPU性能

7.2 I/O系统的可靠性、可用性和可信性

7.3 廉价磁盘冗余阵列RAID

7.4 总线

7.5 通道处理机

7.6 I/O与操作系统

7.4 总线

- 计算机系统中，各子系统之间通过**总线**互相连接
 - **优点：**成本低、多样性
 - **主要缺点：**它是由不同的外设分时共享的，形成了信息交换的瓶颈，从而限制了系统中总的I/O吞吐量

7.4 总线

7.4.1 总线的设计

1. 总线设计存在很多技术难点

- **一个重要原因：**总线上信息传送的速度极大地受限于各种物理因素：
 - 如总线的长度、设备的数目、信号的强度等，这些物理因素限制了总线性能的提高。
- **设计需求上的冲突：**一方面要求I/O操作响应快，另一方面又要求高吞吐量

7.4 总线 (1) 总线的设计

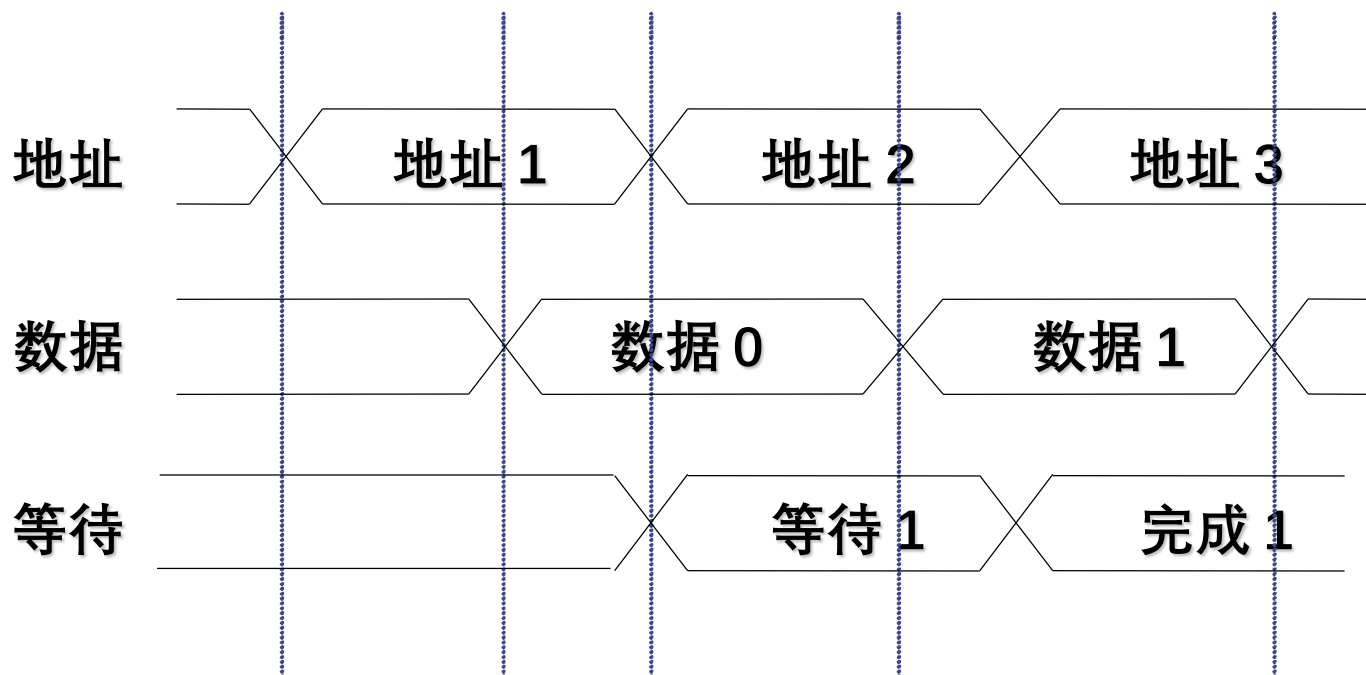
➤ 设计总线时需要考虑的一些问题

特性	高性能	低价格
总线宽度	独立的地址和数据总线	数据和地址分时 共用同一套总线
数据总线宽度	越宽越快 (例如: 64位)	越窄越便宜 (例如: 8位)
传输块大小	块越大总线开销越小	单字传送更简单
总线主设备	多个 (需要仲裁)	单个 (无需仲裁)
分离事务	采用——分离的请求包和 应答包能提高总线带宽	不采用——持续连接成本 更低, 而且延迟更小
定时方式	同步	异步

2. 分离事务总线（又称：流水总线、悬挂总线、包交换总线）

- 在有多个主设备时，可以通过打包技术来提高总线带宽
- 基本思想
 - 将总线事务分成请求和应答两部分。
 - 在请求和应答之间的空闲时间内，总线可以供其他的I/O使用，这样就不必在整个I/O过程中都独占总线。
- 工作过程的示意图

7.4 总线 (1) 总线的设计



- **分离事务总线**有较高的带宽，但是它的数据传送延迟通常比独占总线方法大。

3. 同步总线

- 包含一个供总线上所有设备使用的时钟，并且这些设备是基于该时钟按照一个固定的协议来发送地址和数据的。
- 优点：速度快、成本低
- 缺点：总线操作都必须以同样的时钟频率进行

4. 异步总线

- 没有统一的参考时钟，每个设备都有各自的定时方法
- 采用握手协议

7.4 总线

7.4.2 总线标准和实例

1. **I/O总线标准：**定义如何将设备与计算机进行连接的文档
2. **常见I/O总线的一些典型特征**

几种常用**并行I/O总线**

	IDE / Ultra ATA	SCSI	PCI	PCI-X
数据宽度 (b)	16	8/16	32/64	32/64
时钟频率 (MHz)	100	10 (Fast) 20 (Ultra) 40 (Ultra2) 80 (Ultra3) 160 (Ultra4)	33/66	66/100/133
总线主设备数量	1个	多个	多个	多个
峰值带宽 (MBps)	200	320	533	1066
同步方式	异步	异步	同步	同步
标准	无	ANSI X3.131	无	无

7.4 总线 (2) 总线标准和实例

在嵌入式系统中使用较多的4种**串行I/O总线**的一些典型特征

	I ² C	1-wire	RS-232	SPI
数据宽度 (b)	1	1	2	1
信号线数量	2	1	9/25	3
时钟频率 (MHz)	0.4 ~ 10	异步	0.04或异步	异步
总线主设备数量	多个	多个	多个	多个
峰值带宽 (Mbps)	0.4 ~ 3.4	0.014	0.192	1
同步方式	异步	异步	异步	异步
标准	无	无	EIA, ITU-T V.21	无

7.4 总线 (2) 总线标准和实例

在服务器系统中使用的CPU-存储器互连系统

	HP HyperPlane Crossbar	IBM SP	SUN Gigaplane- XB
数据宽度 (b)	64	128	128
时钟频率 (MHz)	120	111	83.3
总线的主设备数	多个	多个	多个
每端口峰值带宽 (MBps)	960	1700	1300
总峰值带宽 (MBps)	7680	14200	10667
同步方式	同步	同步	同步
标准	无	无	无

7.4 总线

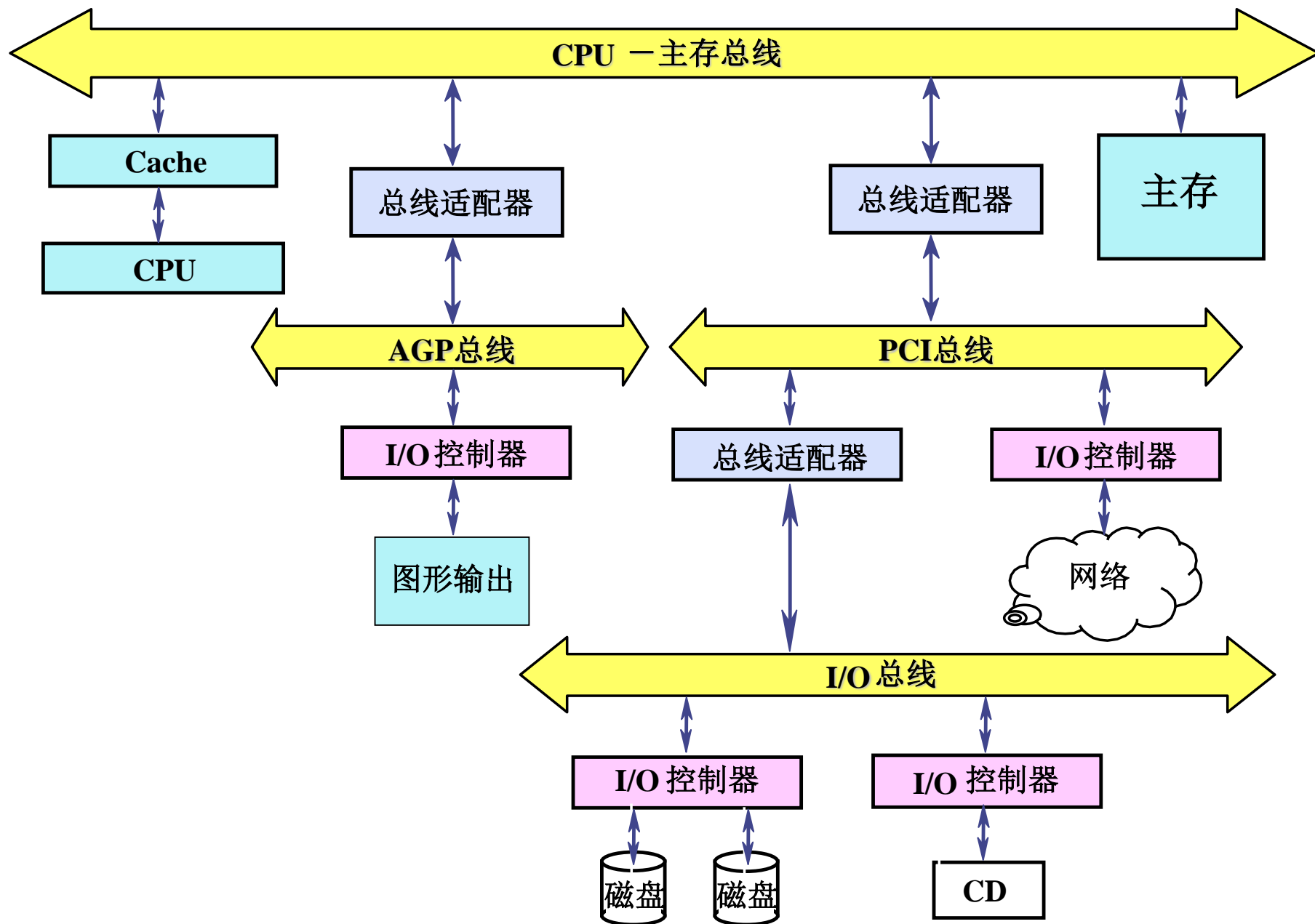
7.4.3 与CPU的连接

1. I/O总线的物理连接方式有两种选择

- 连接到存储器上（更常见）
- 连接到Cache上

2. I/O总线连接到存储器总线上的方式

- 一种典型的组织结构



3. CPU对I/O设备的编址有两种方式

➤ 存储器映射I/O（也称为统一编址）

- 将一部分存储器地址空间分配给I/O设备，用load指令和store指令对这些地址进行读写将引起I/O设备的数据传输。
- 将一部分存储空间留出用于设备控制，对这一部分地址空间进行读写就是向设备发出控制命令。

➤ 给I/O设备独立编址

- 需要在CPU中设置专用的I/O指令来访问I/O设备。
- CPU需要发出一个标志信号来表示所访问的地址是I/O设备的地址。

4. CPU与外部设备进行输入/输出的方式可分为4种

- 程序查询
- 中断
- DMA
- 通道

第7章 输入/输出系统

7.1 I/O系统性能与CPU性能

7.2 I/O系统的可靠性、可用性和可信性

7.3 廉价磁盘冗余阵列RAID

7.4 总线

7.5 通道处理机

7.6 I/O与操作系统

7.5 通道处理机

- 通道处理机能够负担外围设备的大部分I/O工作。
 - **通道处理机**（简称通道）：专门负责整个计算机系统的输入/输出工作。通道处理机只能执行有限的一组输入/输出指令。
 - (1) 通道的作用和功能
 - (2) 通道的工作过程
 - (3) 通道的种类
 - (4) 通道中的数据传送过程与流量分析

7.5 通道处理机

7.5.1 通道的作用和功能

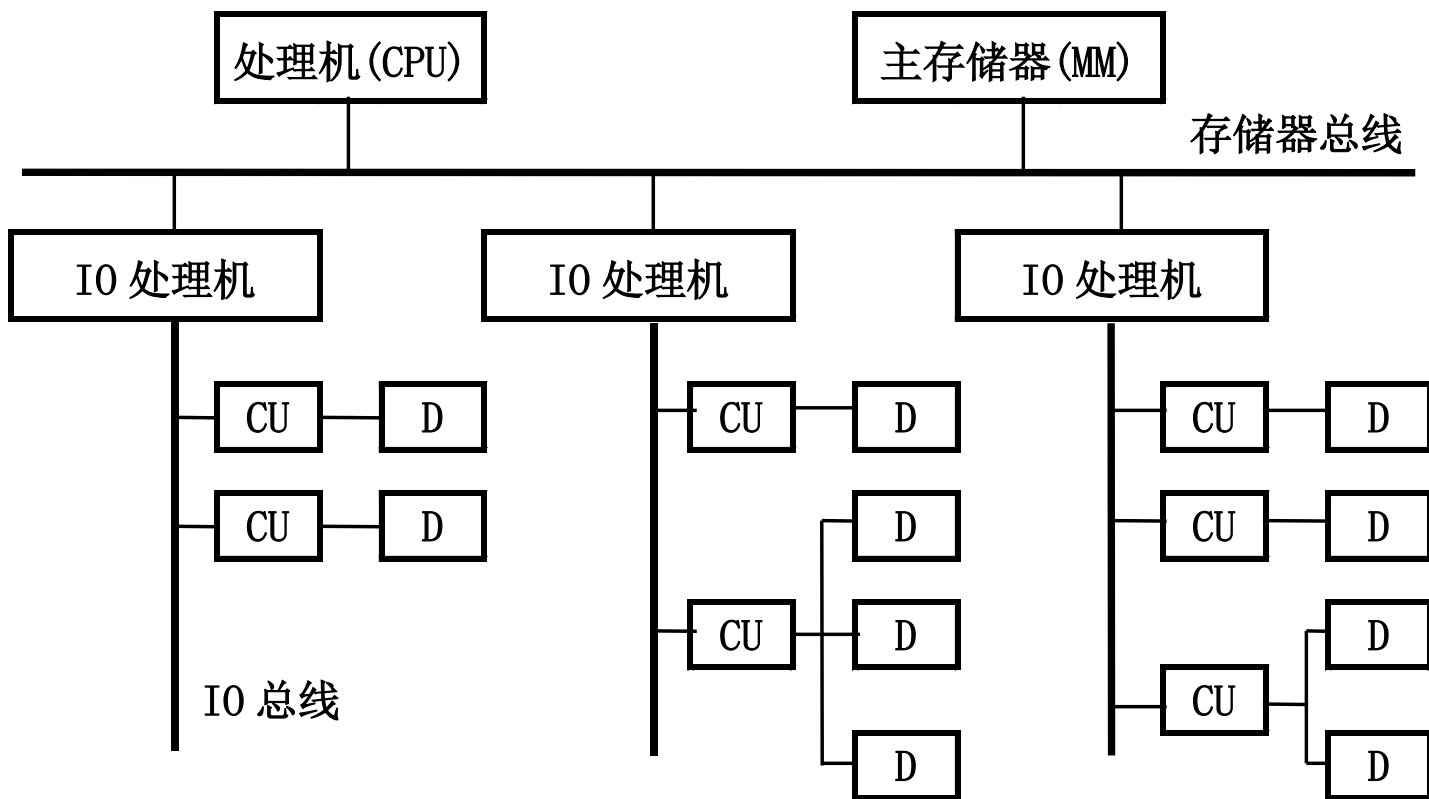
1. 程序控制、中断和DMA方式管理外围设备会引起两个问题：

- 所有外设的输入/输出工作均由CPU承担，CPU的计算工作经常被打断而去处理输入/输出的事务，不能充分发挥CPU的计算能力。
- 大型计算机系统的外设虽然很多，但同时工作的机会不是很多。

解决上述问题的方法：采用通道处理机

7.5 通道处理机 (1) 通道的作用和功能

2. 一个典型的由主机、通道、设备控制器、外设构成的
4级层次结构的输入/输出系统。



CU 是设备控制器，D 是外围设备
输入输出系统的层次结构

3. 通道的功能

- 接收CPU发来的I/O指令，并根据指令要求选择指定的外设与通道相连接。
 - 执行通道程序：
 - 从主存中逐条取出通道指令，对通道指令进行译码，并根据需要向被选中的设备控制器发出各种操作命令
 - 给出外设中要进行读/写操作的数据所在的地址
 - 如磁盘存储器的柱面号、磁头号、扇区号等
 - 给出主存缓冲区的首地址
- 该缓冲区存放从外设输入的数据或者将要输出到外设中去的数据。

7.5 通道处理机 (1) 通道的作用和功能

➤ 控制外设与主存缓冲区之间的数据传送的长度

对传送的数据个数进行计数，并判断数据传送是否结束。

➤ 指定传送工作结束时要进行的操作

例如：将外设的中断请求及通道的中断请求送往CPU等。

➤ 检查外设的工作状态是否正常，并将该状态信息送往主存指定单元保存。

➤ 在数据传输过程中完成必要的格式变换

例如：把字拆分为字节，或者把字节装配成字等。

4. 通道的主要硬件

➤ 寄存器

- 数据缓冲寄存器
- 主存地址计数器
- 传输字节数计数器
- 通道命令字寄存器
- 通道状态字寄存器

➤ 控制逻辑

- 分时控制
- 地址分配
- 数据传送、装配和拆分等

5. 通道对外设的控制通过输入/输出接口和设备控制器进行

- 通道与设备控制器之间一般采用标准的输入/输出接口来连接。
- 通道通过标准接口把**操作命令**送到**设备控制器**，设备控制器解释并执行这些通道命令，完成命令指定的操作。
- 设备控制器能够记录外设的状态，并把状态信息送往**通道和CPU**。

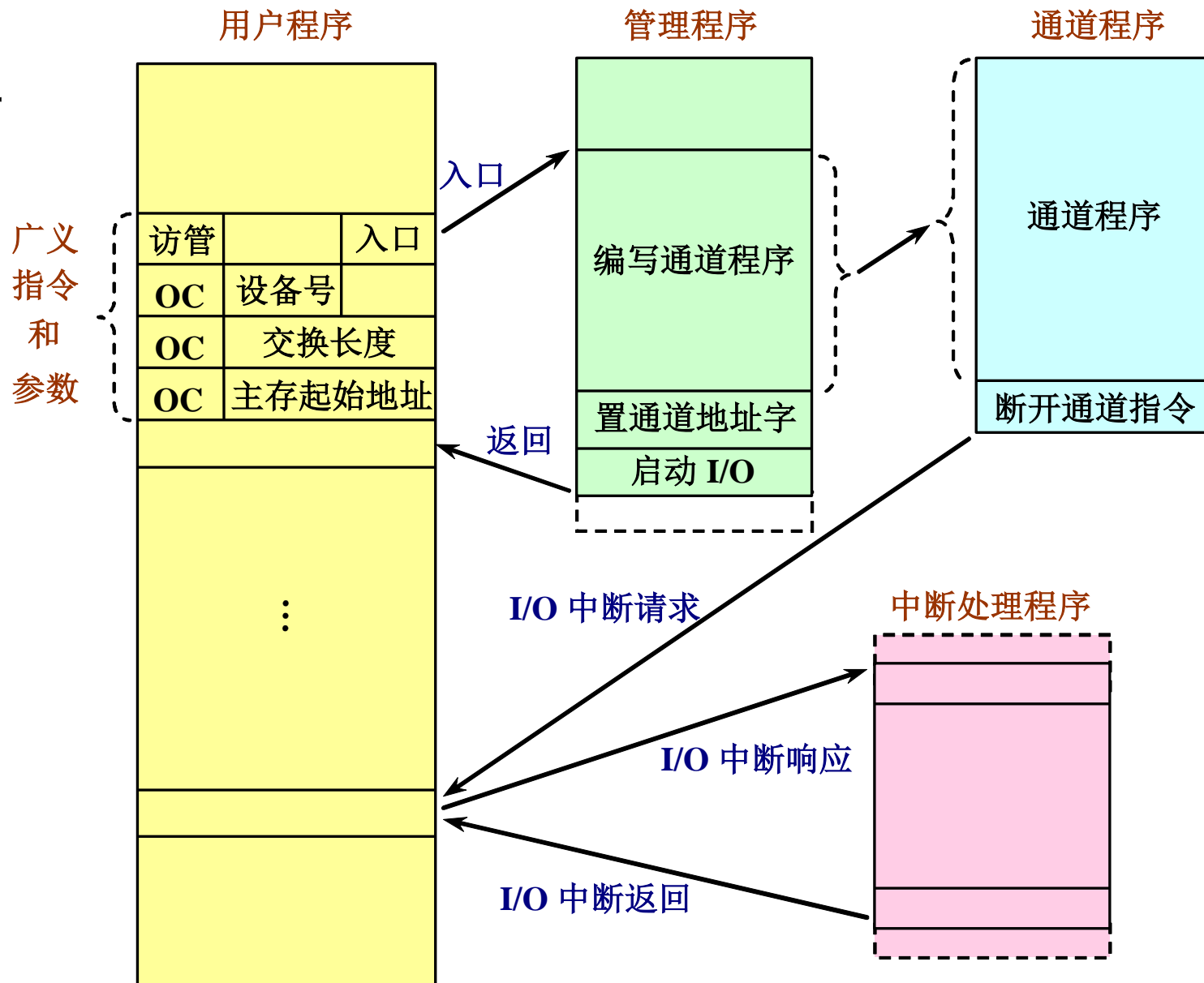
7.5.2 通道的工作过程

1. 通道完成一次数据输入/输出的工作过程

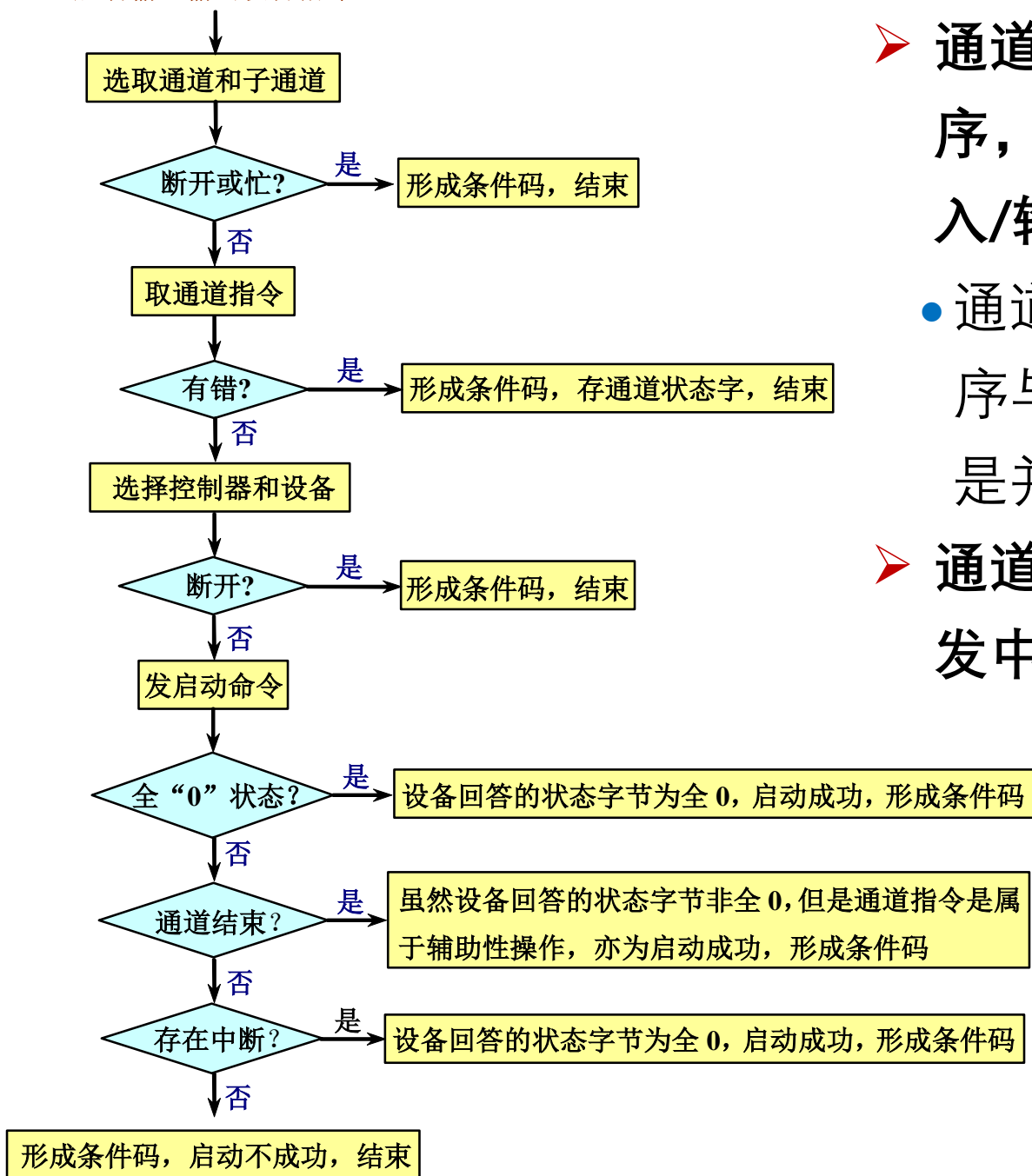
- 在用户程序中使用**访管指令**进入管理程序，由管理程序生成一个**通道程序**，并启动通道。
 - 用户在目标程序中设置一条**广义指令**，通过调用操作系统的**管理程序**来实现。
 - **管理程序**根据**广义指令**提供的参数来编制**通道程序**
 - **启动输入/输出设备指令**是一条主要的输入/输出指令，属于**特权指令**。

7.5 通道处理机 (2) 通道的工作过程

✓ 工作流程



启动输入输出设备指令



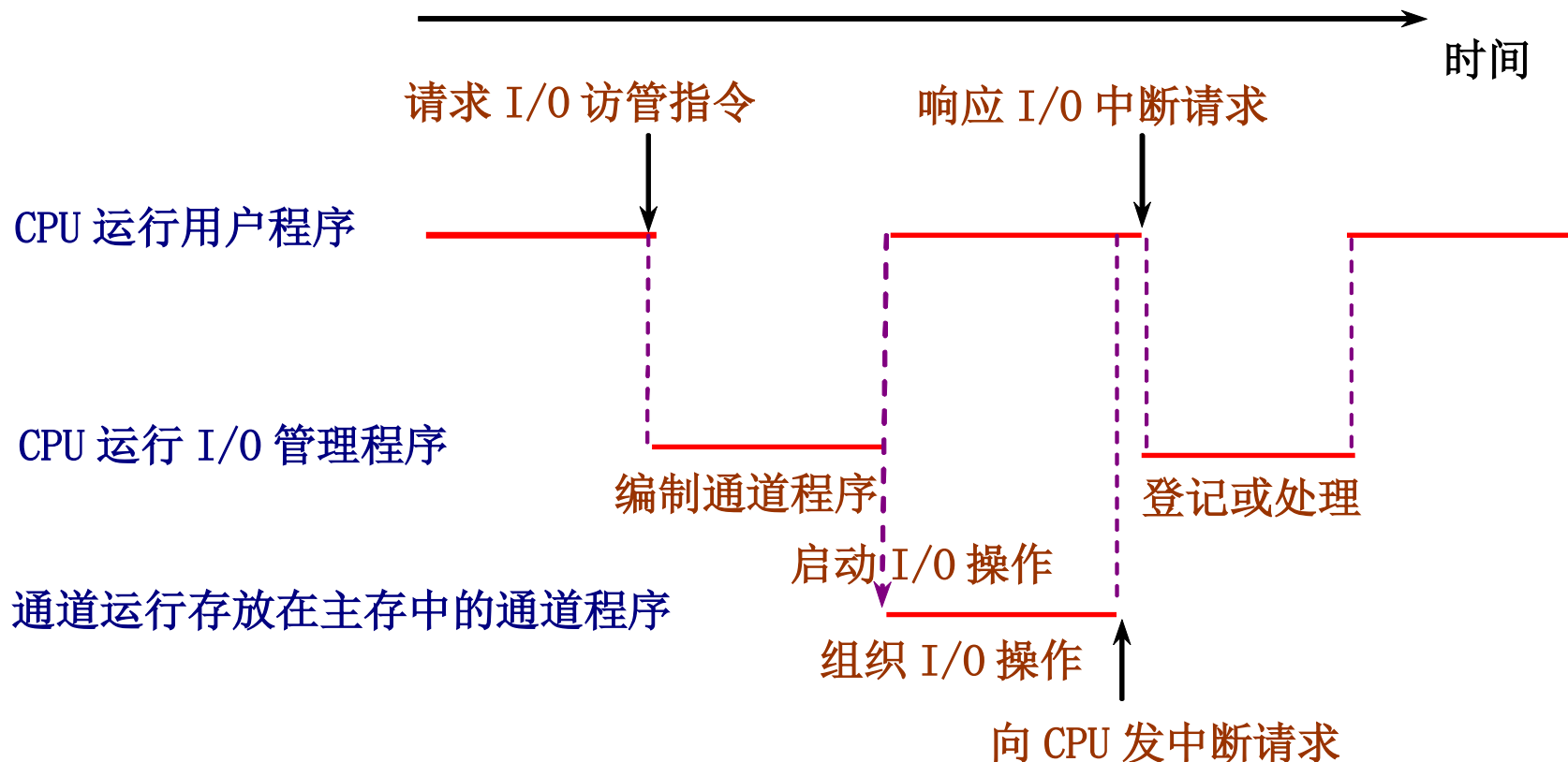
➤ 通道处理机执行通道程序，完成指定的数据输入/输出工作。

● 通道处理机执行通道程序与CPU执行用户程序是并行的。

➤ 通道程序结束后向CPU发中断请求

7.5 通道处理机 (2) 通道的工作过程

2. CPU执行程序 and 通道执行程序的时间关系

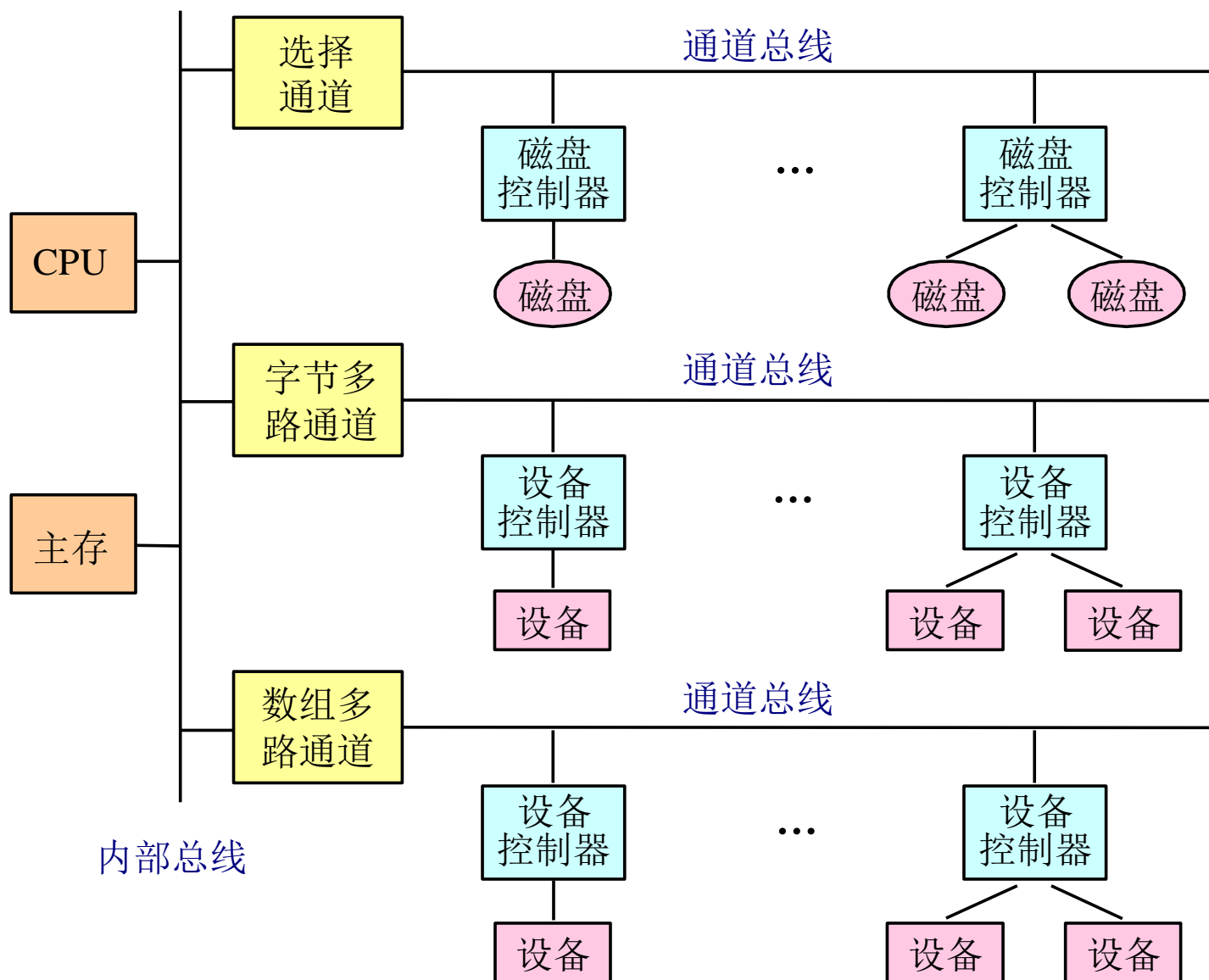


7.5 通道处理机

7.5.3 通道的种类

- 根据信息传送方式的不同，将通道分为**三种类型**：
 - 字节多路通道
 - 选择通道
 - 数组多路通道
- 三种类型的通道与CPU、设备控制器和外设的连接关系：

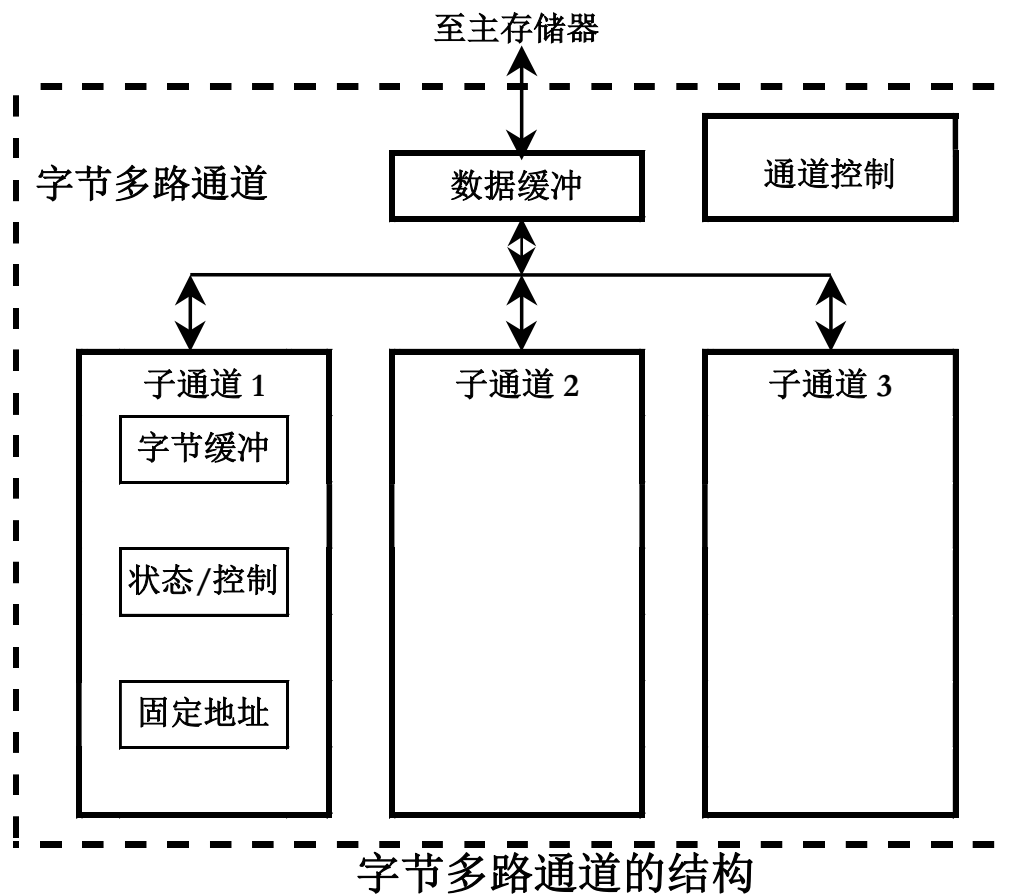
7.5 通道处理机 (3) 通道的种类



7.5 通道处理机 (3) 通道的种类

1. 字节多路通道

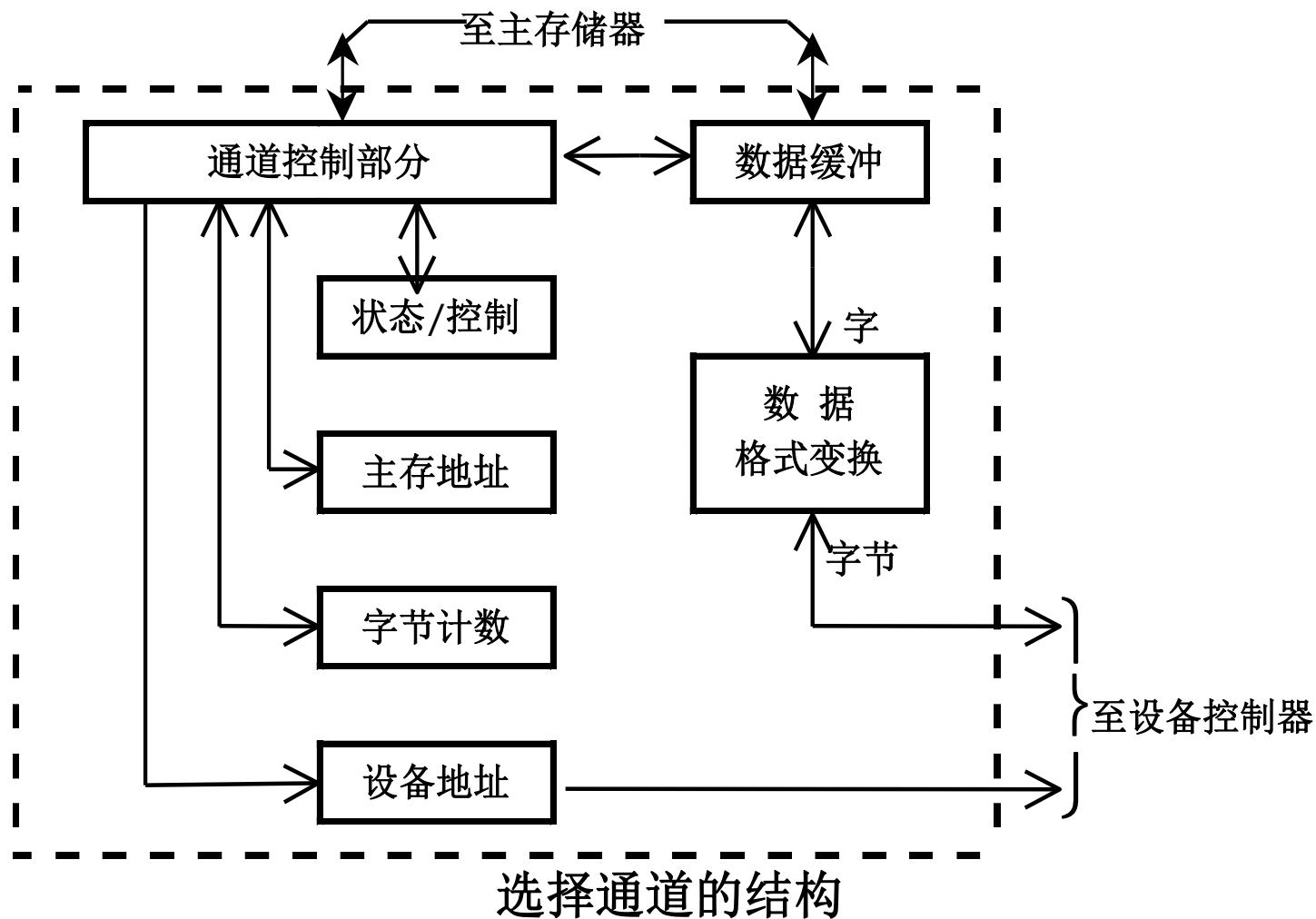
- 为多台低速或中速的外设服务
- 以字节交叉的方式分时轮流地为它们服务
- 字节多路通道可以包含多个子通道，每个子通道连接一台设备控制器。



2. 选择通道

- 为多台高速外围设备服务。
- 在一段时间内只为一台高速外设独占使用。
- 选择通道的硬件
 - 5个寄存器
数据缓冲寄存器、设备地址寄存器、主存地址计数器、交换字节数计数器、设备状态/控制寄存器
 - 格式变换部件
用于在主存和设备之间进行字与字节的拆分和装配
 - 通道控制部件

7.5 通道处理机 (3) 通道的种类



3. 数组多路通道

- 适用于高速设备
 - 每次选择一个高速设备后传送一个数据块，轮流为多台外围设备服务
 - 数组多路通道之所以能够并行地为多台高速设备服务，是因为虽然其所连设备的传输速率很高，但寻址等辅助操作时间很长
- 从磁盘存储器读出文件的过程为：定位、找扇区、读出数据
 - 数组多路通道的实际工作方式是：在为一台高速设备传送数据的同时，有多台高速设备可以在定位或者在找扇区。

与选择通道相比，数组多路通道的数据传输率和通道的硬件利用率都很高，控制硬件的复杂度也高。

7.5 通道处理机

7.5.4 通道中的数据传送过程与流量分析

- 通道流量

- 一个通道在数据传送期间，单位时间内能够传送的数据量。所用单位一般为Bps。
- 又称为**通道吞吐率**、通道数据传输率等。
- **通道最大流量**：通道在满负荷工作状态下的流量

7.5 通道处理机 (4) 通道中的数据传输过程与流量分析

➤ 参数的定义

T_s : 设备选择时间。从通道响应设备发出的数据传输请求开始，到通道实际为这台设备传送数据所需要的时间

T_D : 传送一个字节所用的时间

p : 在一个通道上连接的设备台数，且设备同时都在工作

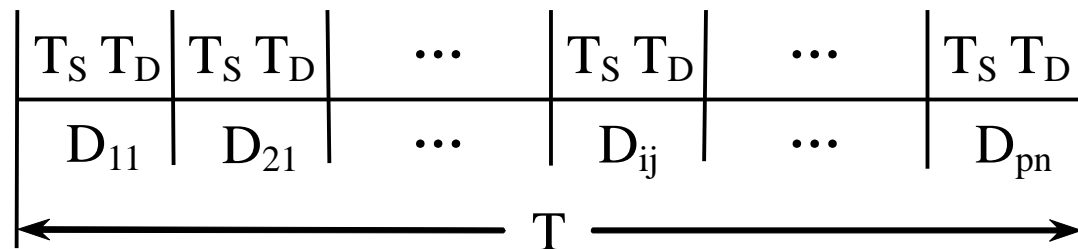
n : 每台设备传送的字节数，这里假设每台设备传送的字节数都相同

k : 数组多路通道传输的一个数据块中包含的字节数。在一般情况下， $k < n$ 。对于磁盘、磁带等磁表面存储器，通常 $k=512$

T : 通道完成全部数据传输工作所需要的时间

1. 字节多路通道

➤ 数据传输过程



- 通道每连接一台外设，只传送一个字节，然后又与另一台设备连接，并传送一个字节。
- p台设备每台传送n个数据总共所需的时间为：

$$T_{\text{BYTE}} = (T_S + T_D) \times p \times n$$

7.5 通道处理机 (4) 通道中的数据传输过程与流量分析

➤ **最大流量**

$$f_{\text{MAX-BYTE}} = \frac{pn}{(T_S + T_D)pn} = \frac{1}{T_S + T_D}$$

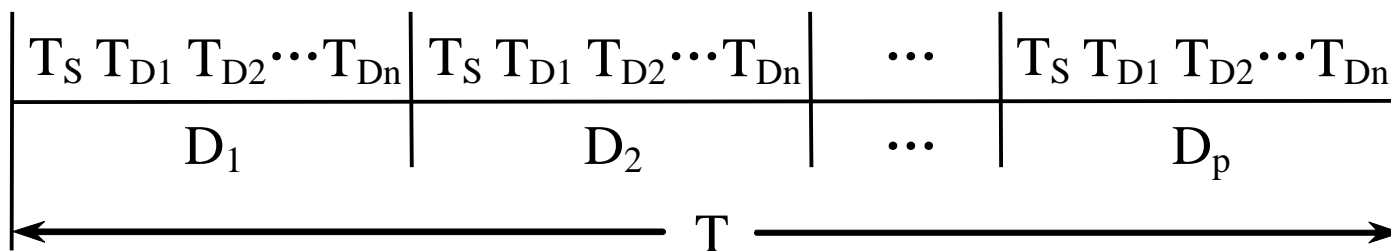
➤ **实际流量**是连接在这个通道上的所有设备的数据传输率之和。

$$f_{\text{BYTE}} = \sum_{i=1}^p f_i$$

- f_i : 第*i*台设备的实际数据传输率

2. 选择通道

- 在一段时间内只能单独为一台高速外设服务，当这台设备的数据传送工作全部完成后，通道才能为另一台设备服务。
- 工作过程



其中： D_i 表示通道正在为第*i*台设备服务

$$T_{D1} = T_{D2} = \cdots = T_{Dn} = T_D$$

➤ 每台传送n个数据，p台设备总共所需的时间

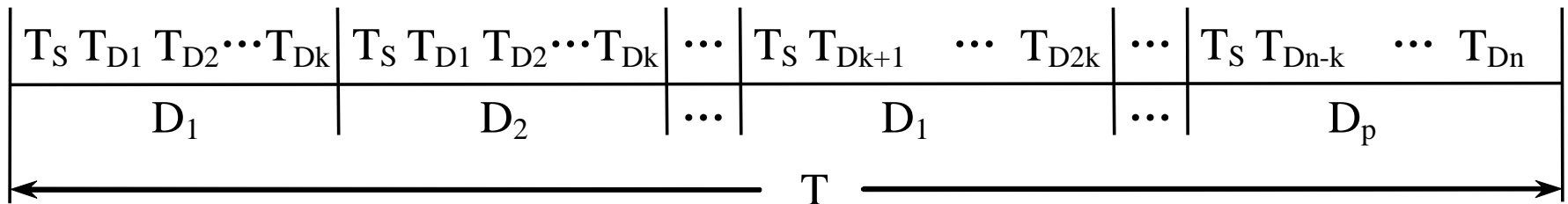
$$T_{\text{SELECT}} = \left(\frac{T_S}{n} + T_D \right) \times p \times n$$

➤ 最大流量

$$f_{\text{MAX-SELECT}} = \frac{pn}{\left(\frac{T_S}{n} + T_D \right) pn} = \frac{1}{\frac{T_S}{n} + T_D}$$

3. 数组多路通道

➤ 工作过程



➤ 每台传送n个数据，p台设备总共所需的时间为：

$$T_{\text{BLOCK}} = \left(\frac{T_S}{k} + T_D \right) \times p \times n$$

➤ 最大流量

$$f_{\text{MAX-BLOCK}} = \frac{pn}{(\frac{T_S}{k} + T_D)pn} = \frac{1}{\frac{T_S}{k} + T_D}$$

- 选择通道和数组多路通道的实际流量就是连接在这个通道上的所有设备中数据流量最大的那一个。

$$f_{\text{BLOCK}} \leq \max_{i=1}^p f_i$$

$$f_{\text{SELECT}} \leq \max_{i=1}^p f_i$$

➤ 各种通道的**实际流量**应该不大于通道的最大流量

$$f_{\text{BYTE}} \leq f_{\text{MAX-BYTE}}$$

$$f_{\text{BLOCK}} \leq f_{\text{MAX-BLOCK}}$$

$$f_{\text{SELECT}} \leq f_{\text{MAX-SELECT}}$$

- 两边的差值越小，通道的利用率就越高
- 当两边相等时，通道处于满负荷工作状态

7.5 通道处理机 (4) 通道中的数据传输过程与流量分析

例7.3: 8台外设的数据传输速率分别如下表所示。现设计一种通道可实现设备选择时间为 $2\mu s$ ，传送一个字节数据为 $2\mu s$ 。

设备号	1	2	3	4	5	6	7	8
数据传输率 (KBps)	500	240	100	75	50	40	14	10

- 1、如果按字节多路通道设计，该通道的最大流量是多少？若希望从8台外设中选择4台外设同时连接到该通道上，而且尽量多连接传输速率高的设备，那么应选择哪些外设连接到该通道上？
- 2、如果按数组多路通道设计，且一次传送定长数据块的大小为512B，该通道的最大流量是多少？从8台外设中可选择哪些外设连接到该通道上？

(自学)

7.5 通道处理机 (4) 通道中的数据传输过程与流量分析

解： (1) 该通道的工作周期为 $4\mu s$ ，则最大流量为：

$$f_{\max-BYTE} = \frac{1}{T_S + T_D} = \frac{1}{(2 + 2) \times 10^{-6}} = 250(KBps)$$

➤ 选择4台外设同时连接到该通道上，而且尽量多连接传输速率高的设备，那么应选择：设备3、4、5、7。

(2) 数组多路通道的最大流量为：

$$f_{\max-BYTE} = \frac{k}{T_S + kT_D} = \frac{512}{(2 + 512 \times 2) \times 10^{-6}} = 0.499 \times 10^6 (Bps) = 499(KBps)$$

➤ 从8台外设中可选择：设备2、3、4、5、6、7、8 这7台外设连接到该通道上。

7.5 通道处理机 (4) 通道中的数据传输过程与流量分析

例7.4: 一个字节多路通道连接有5台设备，它们的数据传输率如下表：

(自学)

设备	A	B	C	D	E
数据传输率 (KB/s)	100	33.3	33.3	20	10

- 1、计算这个字节多路通道的**实际工作流量**；
- 2、为了使通道能正常工作，请设计该通道的**最大流量和工作周期**；
- 3、当这个字节多路通道工作在最大流量时，5台设备都在0时刻同时向通道发出一次传送数据的请求，并在以后的时间里按各自的数据传输率连续工作。**画出通道分时为各台设备服务的时间关系图，并计算这个字节多路通道处理完各台设备的第一次数据服务请求的时刻。**

7.5 通道处理机 (4) 通道中的数据传输过程与流量分析

- 解：**
- (1) 该通道的实际流量等于各子通道流量之和：196.6KB/s
 - (2) 取最大流量为200 KB/s，则工作周期为 $5\mu\text{s}$ ；
 - (3) 时间关系图：

各台设备处理完第一次数据服务请求的时刻分别为：
 $5\mu\text{s}$, $10\mu\text{s}$, $20\mu\text{s}$, $30\mu\text{s}$, $90\mu\text{s}$

第7章 输入/输出系统

7.1 I/O系统性能与CPU性能

7.2 I/O系统的可靠性、可用性和可信性

7.3 廉价磁盘冗余阵列RAID

7.4 总线

7.5 通道处理机

7.6 I/O与操作系统

7.6 I/O与操作系统

- 设计I/O系统需要注意**操作系统的因素**
 - 在**用硬件实现的I/O**技术中，由操作系统来决定哪些会实际被采用。
 - I/O操作主要是在**外设和存储器之间进行**，所以操作系统必须保证这些I/O操作的安全性。

7.6.1 DMA和虚拟存储器

- DMA是使用虚拟地址还是物理地址？

1. 使用物理地址进行DMA传输，存在以下两个问题：

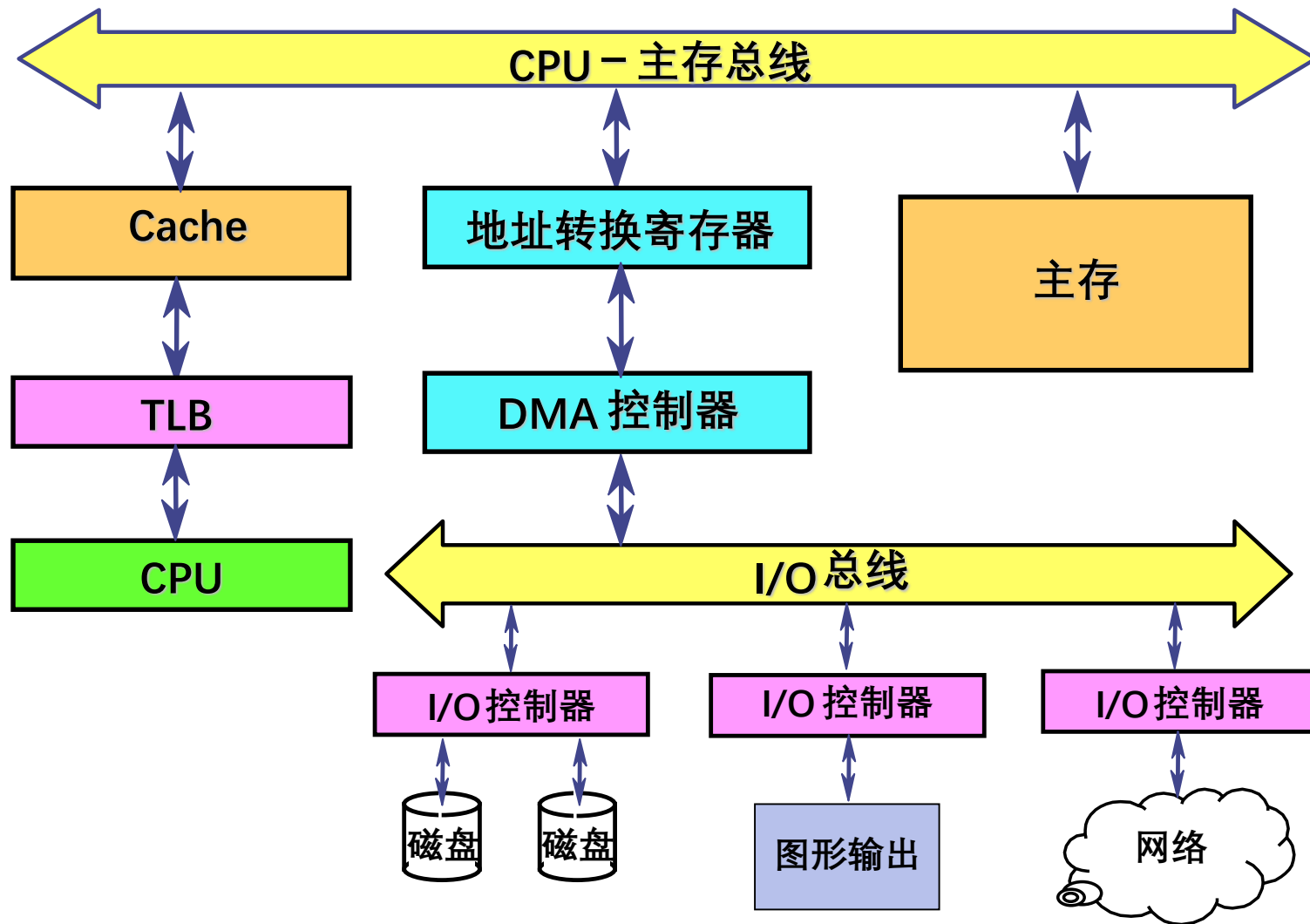
- 对于超过一页的数据缓冲区，由于缓冲区使用的页面在物理存储器中不一定是连续的，所以传输可能会发生问题
- 如果DMA正在存储器和缓冲区之间传输数据时，操作系统从存储器中移出（或重定位）一些页面，那么，DMA将会在存储器中错误的物理页面上进行数据传输

2. 解决这些问题的方法

- 使操作系统在I/O的传输过程中**确保DMA设备所访问的页面都位于物理存储器中**，称这些页面被钉在了主存中。
- **“虚拟DMA”技术**
 - 允许DMA设备直接使用虚拟地址，并在DMA期间由硬件将虚拟地址转换为物理地址。
 - 在采用虚拟DMA的情况下，如果进程在内存中被移动，操作系统应该能够及时地修改相应的DMA地址表。

7.6 I/O与操作系统 (1) DMA和虚拟存储器

虚拟DMA的I/O连接



7.6 I/O与操作系统

7.6.2 I/O和Cache数据的一致性

1. Cache会使一个数据出现两个副本：

一个在Cache中，另一个在主存中。

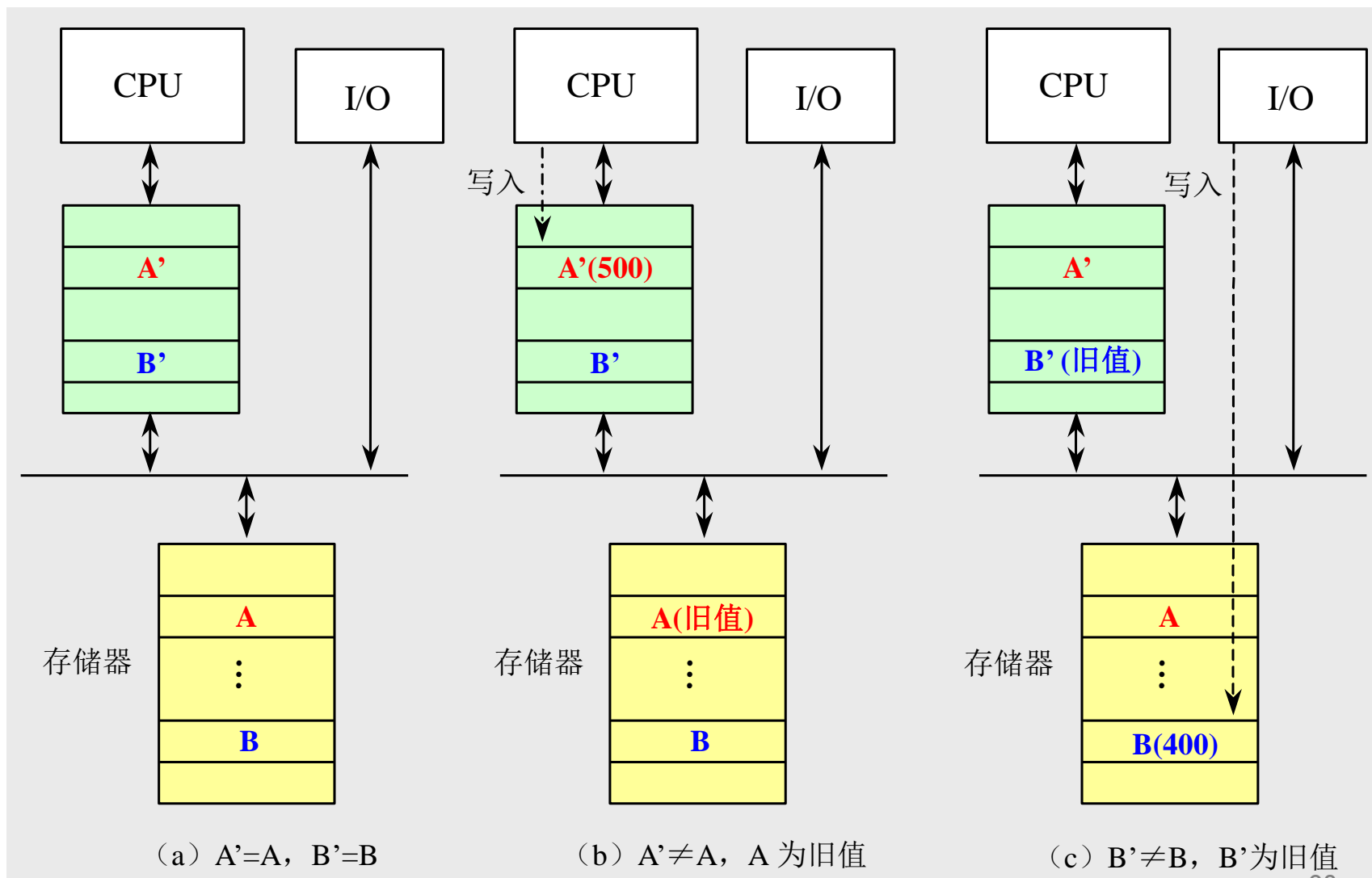
2. I/O设备可以修改存储器中的内容

➤ 把I/O连接到存储器上，会出现以下情况：

- CPU修改了Cache的内容后，由于存储器的内容跟不上Cache内容的变化，I/O系统进行输出操作时所看到的数据是旧值（写直达Cache没有这样的问题）
- I/O系统进行输入操作后，存储器的内容发生了变化，但CPU在Cache中所看到的内容依然是旧值。

7.6 I/O与操作系统 (2) I/O和Cache数据的一致性

举例：假设Cache采用写回法，并且A'是A的副本，B'是B的副本。



➤ 把I/O直接连接到Cache上

- 不会产生由I/O导致的数据不一致的问题

所有I/O设备和CPU都能在Cache中看到最新的数据

- I/O会跟CPU竞争访问Cache，在进行I/O时，会造成CPU的停顿。
- I/O还可能会破坏Cache中CPU访问的内容，因为I/O操作可能导致一些新数据被加入Cache，而这些新数据可能在近期内并不会被CPU访问。

3. 解决内容一致性问题的方法

(不管Cache是采用写直达法还是写回法)

➤ 软件的方法

- 设法保证I/O缓冲器中的所有各块都不在Cache中
- 具体做法有两种：
 - ✓ 把I/O缓冲器的页面设置为不可进入Cache的，在进行输入操作时，操作系统总是把输入的数据放到该页面上。
 - ✓ 在进行输入操作之前，操作系统先把Cache中与I/O缓冲器相关的数据“赶出”Cache，即把相应的数据块设置为“无效”状态。

➤ 硬件的方法

- 在进行输入操作时，**检查相应的I/O地址（I/O缓冲器中的单元）是否在Cache中（即是否有数据副本）**
- 如果发现I/O地址在Cache中有匹配的项，就把相应的Cache块**设置为“无效”**

7.6 I/O与操作系统

7.6.3 异步I/O

1. 提高I/O性能的一种方法：并行

- 在程序中使多个磁盘同时工作，并行地访问数据

2. 同步I/O

- 进程处于等待状态直到所需的数据从磁盘读出

3. 异步I/O

- 允许进程在发出I/O请求后继续执行，直到该进程真正访问这些数据而它们又尚未就绪时，才被挂起

第7章 输入/输出系统

7.1 I/O系统性能与CPU性能

7.2 I/O系统的可靠性、可用性和可信性

7.3 廉价磁盘冗余阵列RAID

7.4 总线

7.5 通道处理机

7.6 I/O与操作系统

● 课本作业： 8.6 8.7 8.10

计算机体系结构教学日历

	内容		作业
第一周	1 基本概念	1 课程体系及要求	1.6 CPU性能CPI
		2 定量分析	1.9 Amdahl 定律
第二周	2 指令集	3 指令格式及CISC	熟悉MIPS指令与程序
		4 RICS及MIPS	
第三周	3 流水线	5 基本概念	3.9 吞吐率和效率
		6 性能指标	3.11 静态流水时空图
第四周		7 相关与冲突	3.12 动态流水时空图
		8 MIPS流水线实现	3.16 向量处理机
第五周	4 指令级并行	9 动态调度	4.2 Tomasulo算法
		10 分支预测	4.4 分支目标缓冲CPI
第六周		11 多指令流出	4.7 指令调度
	5 存储系统	12 Cache层次体系	5.8 Cache命中率
		13 Cache设计	5.10、 5.12 直接相联、组映像，
第七周	6 输入输出	14 I/O与总线	数据传输速率 6.6， 6.8， 6.10
第八周	7 实验课	15 计算机性能评价	计算机上机测试、分析， 提交实验结果和报告
		16 计算机性能评价	