

第二十讲

上一讲简要回顾

❖ 辅助存储器

- 磁表面存储器的读写原理和编码方式（RZ/NRZ/PM/FM）
- 磁盘的结构、原理、数据格式（柱面、磁头、扇区）
- 磁盘的指标（道密度/位密度、容量、访问时间、数据传输率）
- 磁盘的类型 和 磁盘阵列（RAID 0, 1, 2, 3, 4, 5）
- 光盘的原理、特点、数据格式、类型

❖ 虚拟存储器

- 虚拟存储器技术：把主存当做辅助存储器的“高速缓存”
- 页式虚拟存储器：虚存空间/主存空间、虚页/实页、虚地址/实地址
- 页表：
 - 记录虚页与实页的映射关系，实现虚实地址的转换
 - OS为每道程序建立一个页表，页表通常存放在内存中
 - 当前运行程序的页表首地址 被装入到页表寄存器中
 - 页表用虚页号作为索引，页表项包括虚页对应的实页号和有效位(装入位)

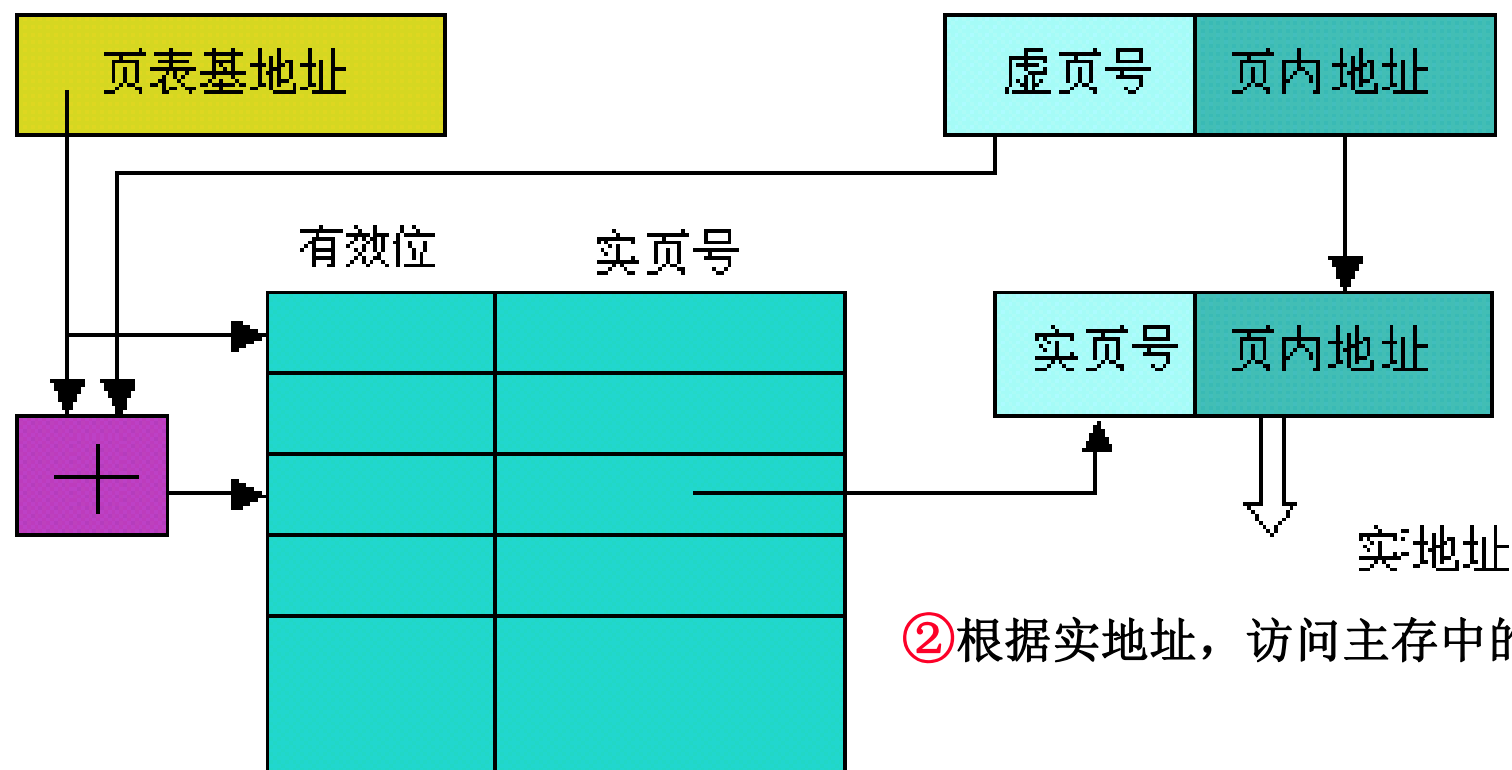
上一讲简要回顾

❖ 虚实地址的转换 —— 访存次数问题

页表寄存器

① 根据虚地址，访问主存中的页表

虚地址



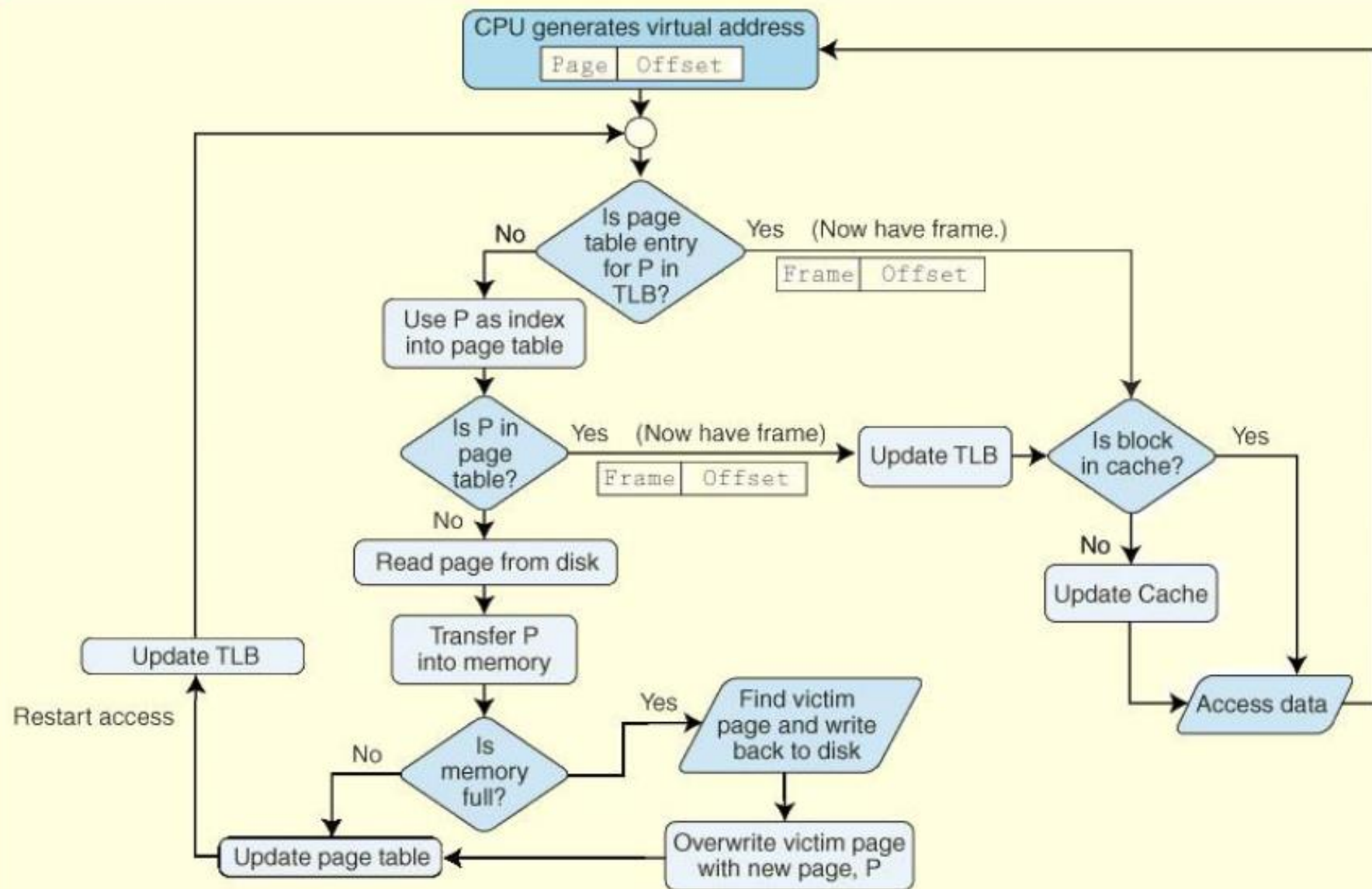
上一讲简要回顾

- ❖ 快表TLB (Translation Lookaside Buffer, 转换后备缓冲器)
 - 问题：每次虚拟存储器的访问带来两次存储器访问，一次访问页表，一次访问所需的数据（或指令），简单的虚拟存储器速度太慢
 - 解决办法：使用Cache存储部分活跃的页表项，称为TLB（快表），它包含了最近使用的那些页表项
 - TLB内容：虚页号（标记）、对应实页号（数据）、有效位、修改位
 - TLB一般采用全相联模式

有效位	修改位	标记 (tag)	数据
		虚页号	实页号
		虚页号	实页号
		虚页号	实页号
		虚页号	实页号

快表 (TLB)

上一讲简要回顾



5.2 页式虚拟存储器

假定页式虚拟存储系统按字节编址，逻辑地址36位，页大小16KB，物理地址32位，页表中包括有效位和修改位各1位、使用位和存期方式位各2位，且所有虚拟页都在使用中。请问：

- (1) 每个进程的页表大小为多少？
- (2) 如果所使用的快表（TLB）总表项数为256项，且采用2路组相联Cache实现，则快表大小至少为多少？

❖ 解答（1）

页面大小： $16\text{KB}=2^{14}$ ，页内偏移14位

虚地址36位：虚页号= $36-14=22$ 位

实地址32为：实页号= $32-14=18$ 位

每个进程最多可有： 2^{22} 个虚页

每个页表项： $1+1+2+2+18=24$ 位

每个页表所占空间： $2^{22} \times 24=12\text{MB}$

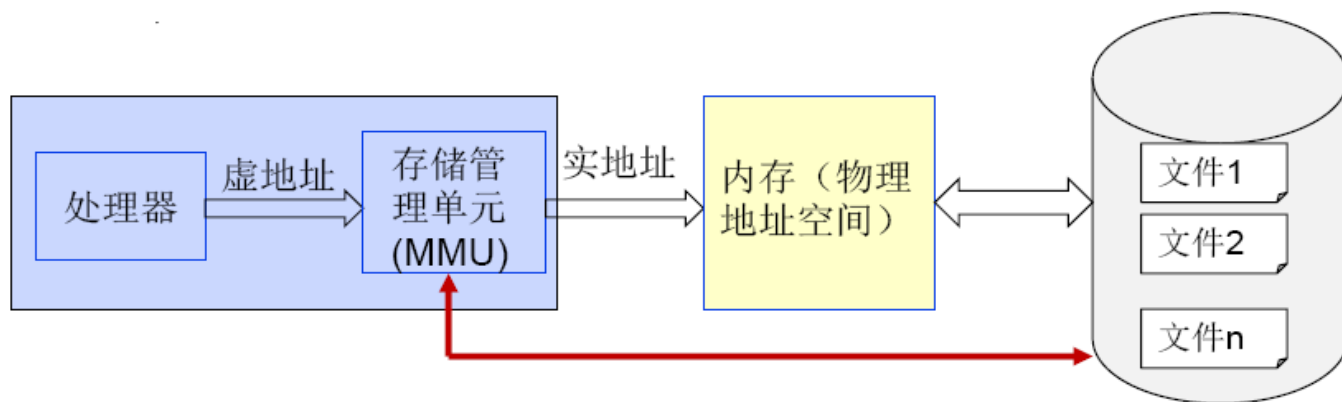
(2)

- TLB：256个表项，2路组相联，所以共有128组
- 22位虚页号：7位组地址，15位Tag
- TLB每个表项： $15+24=39$ 位
- TLB容量： $39 \times 256=9984$ 位
=1248字节

虚拟存储系统小结

❖ 虚拟存储系统的特征

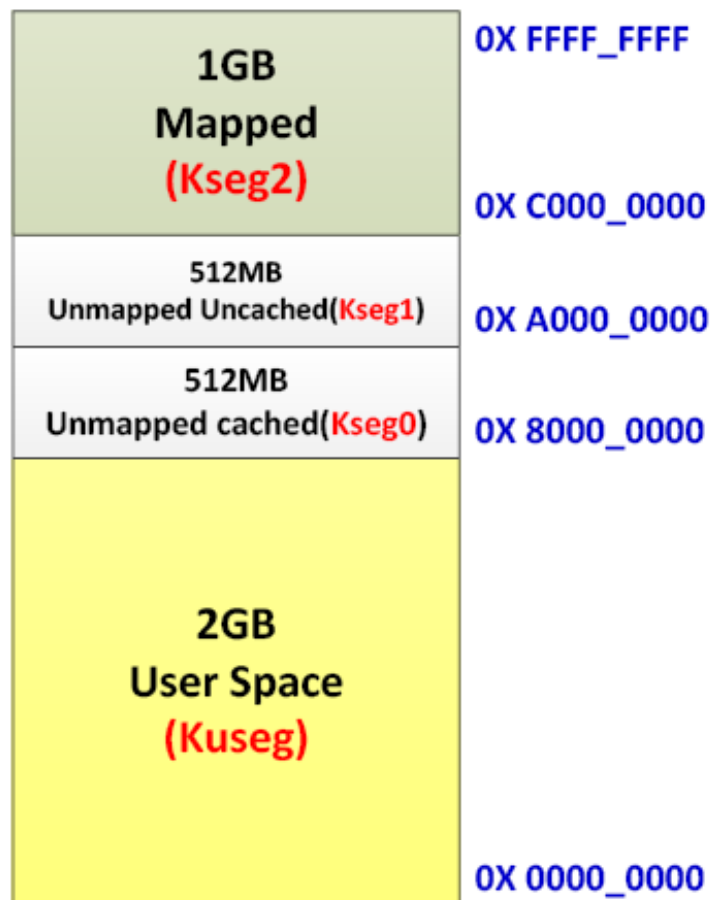
- 程序员在比实际主存空间大得多的逻辑地址空间中编写程序
- 程序执行时，把当前需要的程序段和相应的数据块调入主存，其他暂不用的部分存放在磁盘上
- 指令执行时，通过硬件（MMU）将逻辑地址（也称虚拟地址或虚地址）转化为物理地址（也称主存地址或实地址）
- 在发生程序或数据访问失效时，由操作系统进行主存和磁盘之间的信息交换。



MIPS的存储空间管理

❖ MIPS CPU运行模式：用户态和核心态

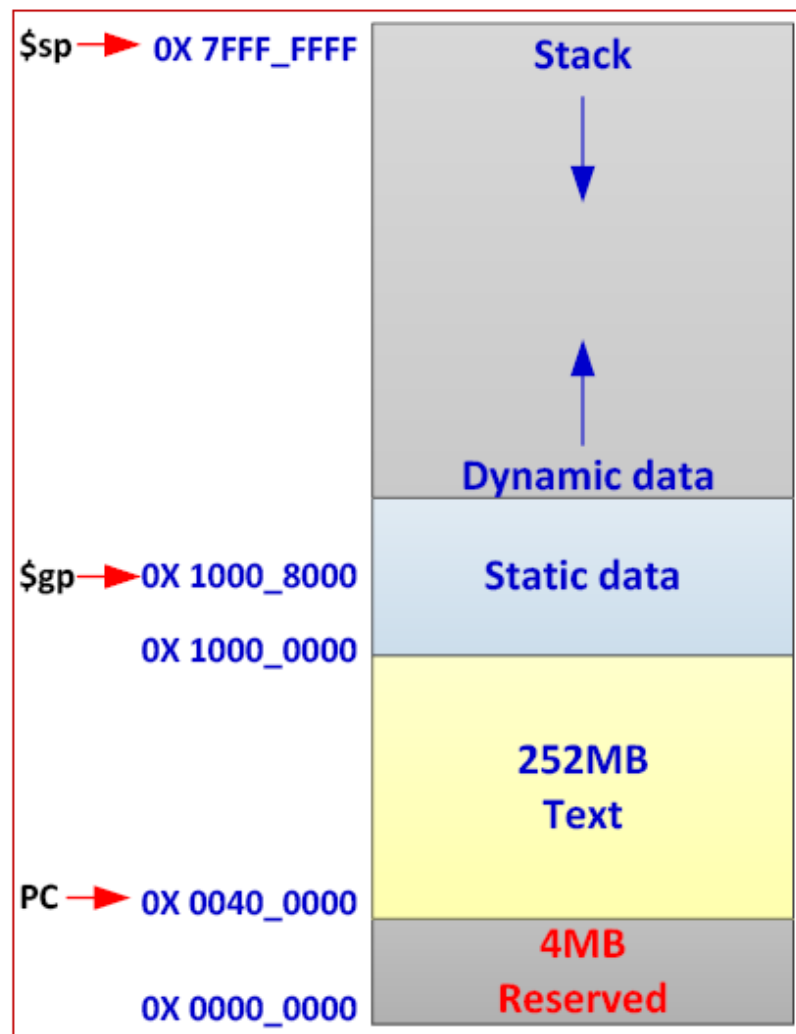
- **Kuseg** : 00000000 - 7FFF FFFF (2G), **用户模式下可用**, 即MIPS规范约定用户空间为2G在带有MMU的机器里, 这些地址都将由MMU转换。
- **KSeg0** : 80000000 - 9FFFFFFF (512M), 通过将最高位清零映射到物理地址低端512M(00000000 - 1FFF FFFF)空间。这种映射简单, 无需MMU转换 (Unmapped)。这段地址的存取都会通过高速缓存(cached), 对于有MMU 的系统, 操作系统内核会存放在该区域。
- **KSeg1** : A0000000 - BFFFFFFF (512M), 将最高3位清零映射到物理地址低端512M(00000000 - 1FFFFFFF)空间, 无需MMU转换 (Unmapped), kseg1不使用缓存 (Uncached)。kseg1是系统重启时能正常工作的内存映射地址空间, 重新启动时的入口向量是BFC00000, 这个向量对应的物理地址是1FC00000。因此你可以使用这段地址空间来访问你的初始化程序的ROM。
- **KSeg2** : C0000000 - FFFFFFFF (1G), 只能在**核心态下使用**, 并且要经过MMU转换, 一般情况下你不需要使用这段地址空间。



MIPS的存储空间管理

❖ MIPS按如下约定为程序分配空间

- 堆栈在高地址区，从高到低增长。
- 过程调用时，生成当前“栈帧”，返回后退回当前栈帧
- 程序的动态数据（如：C中的malloc申请区域、链表）在堆(heap)中从低向高进行存放和释放（free时）栈区位于堆栈高端，堆区位于堆栈低端，静态数据区上方。
- 全局指针\$gp固定设为0x10008000，其16位偏移量的访问范围为0x10000000 到0x1000FFFF
- 静态数据区从固定的0x10000000处开始存放
- 程序代码从固定的0x00400000处开始存放，故PC的初始值为0x00400000。



MIPS每个进程的虚拟（逻辑）地址空间

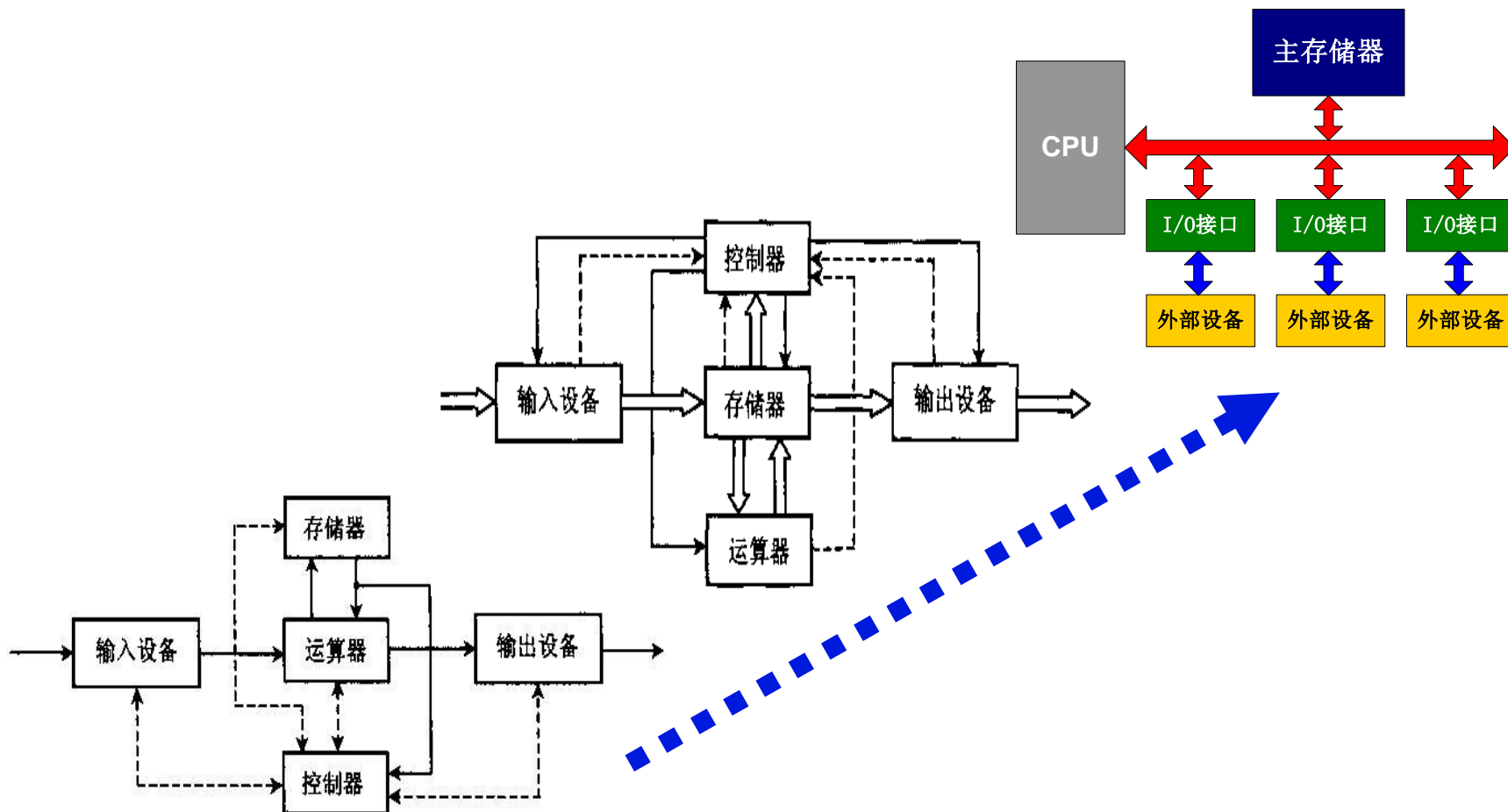
第九讲 总线与I/O

- 一、总线
- 二、I/O接口
- 三、程序查询I/O方式
- 四、中断与中断I/O方式
- 五、DMA I/O方式
- 六、I/O通道

1.1 总线的一般概念

❖ 总线的由来

计算机部件的互连方式：分散连接 → 总线连接



1.1 总线的一般概念

❖ 总线：一组公共的信号通道

- 连接多个部件的信息传输线，各部件共享的传输介质
- 通过总线，计算机在各部件之间实现地址、数据、控制信息的交换
- 在某一时刻，只允许有一个部件向总线发送信息，而多个部件可同时从总线上接收相同的信息
- 总线实际上是由许多传输线或通路组成，每条线可传输一位二进制码，一串二进制代码可以在一段时间内逐一传输。若干条传输线可同时传输若干位二进制代码

1.1 总线的一般概念

❖ 总线的分类

- **片内总线：** **CPU**内部的总线。是**CPU**内部各寄存器之间、寄存器与**ALU**之间传递信息的公共通道
- **系统总线：** **CPU**、主存、**I/O**部件（**I/O**接口）之间传递信息的公共通道。一般分为数据总线、地址总线和控制总线三部分
 - ✓ **数据总线：** 传输数据
 - ✓ **地址总线：** 传输存储器地址和**I/O**地址
 - ✓ **控制总线：**
 - 数据传输控制信号：存储器读写控制信号、**I/O**读写控制信号，应答信号等
 - 总线请求和响应信号：总线请求与仲裁信号，中断请求与响应信号等
 - 其它控制信号：时钟、复位、电源线等
- **通信总线：** 用于计算机系统间或计算机系统与其它系统间的通信

1.1 总线的一般概念

❖ 总线特性

- 机械特性：机械连接方式。如几何尺寸、引脚数量、插头标准。
- 电气特性：信号传输方向、有效电平、电平逻辑等。
- 功能特性：信号功能定义。
- 时间特性：信号之间的时序关系。

❖ 性能指标

- 总线宽度：指数据总线的位数（根数），如32位，64位。
- 标准传输率：每秒传输的最大字节量(B/s)，与总线宽度和频率相关
- 同步/异步方式：总线上的数据与时钟同步工作的总线为同步总线，与时钟异步的总线为异步总线。
- 信号线数：所有信号线的总数。
- 总线控制方式：指总线上各部件使用总线的仲裁方式。
- 总线复用：地址总线与数据总线是否复用（时分多路复用）。

例题

❖ 设一个32位微处理器配有16位的外部数据总线，时钟频率为50MHz，若最短的总线传输周期为4个时钟周期，问处理器的最大数据传输率是多少？若想提高一倍数据传输率，可采用什么措施？

❖ 该总线的最短传输周期为：

$$T = 4/50\text{MHz} = 80 \times 10^{-9} \text{ s}$$

❖ 对于外部总线为16位的处理器，最大数据传输率为：

$$2\text{B}/T = 25 \times 10^6 \text{ Bps} = 25\text{MBps}$$

❖ 若想提高一倍数据传输率，可采用以下两种措施：

(1) 外部数据总线宽度改为32bits，则最大数据传输率为：

$$4\text{B}/T = 50 \times 10^6 \text{ Bps} = 50 \text{ MBps}$$

(2) 时钟频率加倍，则总线传输周期为： $T_1 = 4/100\text{MHz} = 40 \times 10^{-9} \text{ s}$,

最大数据传输率： $2\text{B}/T_1 = 50 \times 10^6 \text{ Bps} = 50 \text{ MBps}$

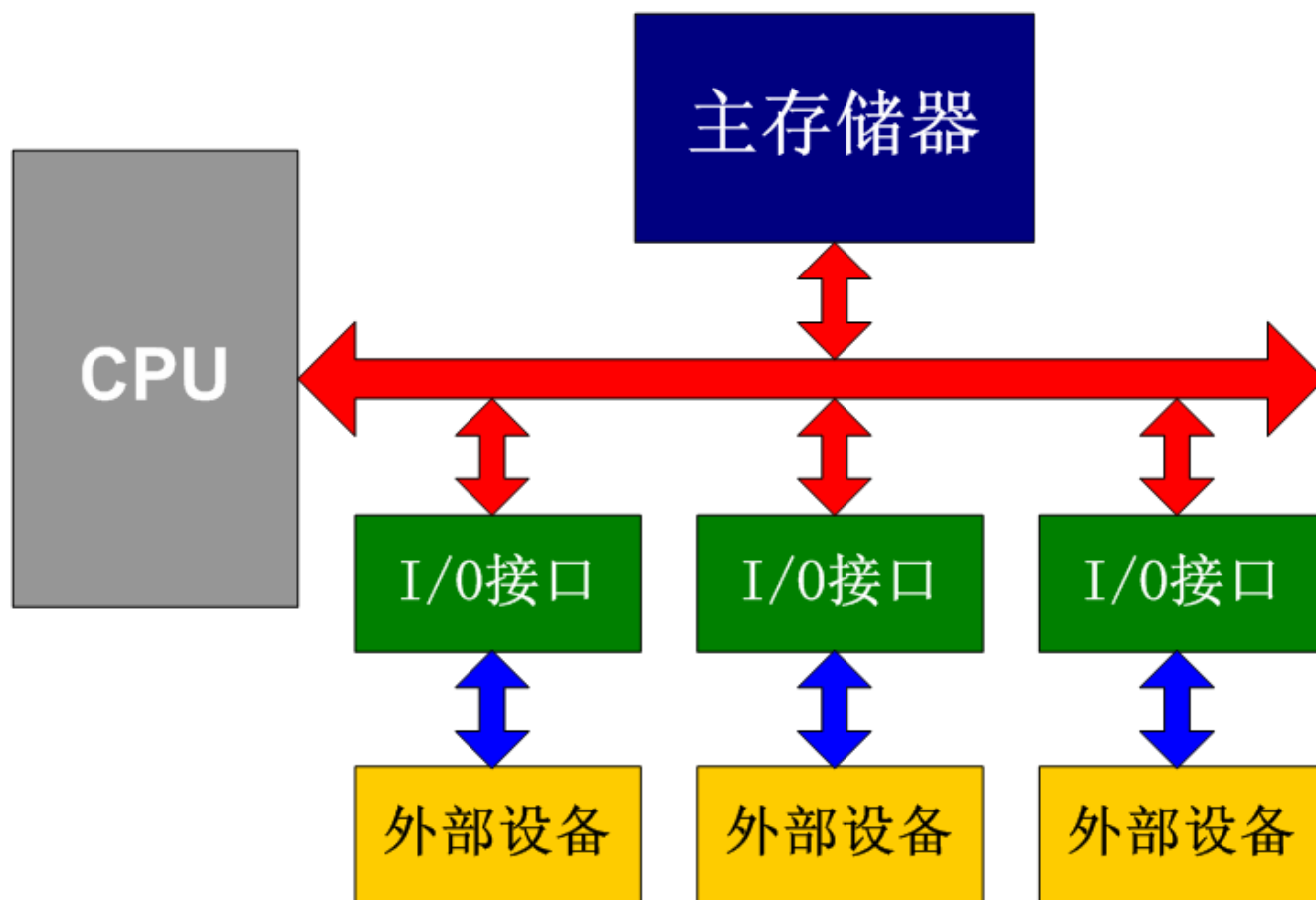
1.1 总线的一般概念

❖ 总线标准

- 可视为系统与各部件、部件与部件间的一个互连标准界面。该界面对它两端的部件都是透明的，即界面的任一方只需根据总线标准的要求，完成自身一面接口的功能，而无需了解对方接口与总线的连接要求
- **ISA (Industrial Standard Architecture)**：16位数据总线，24位地址总线，总线时钟频率8MHz，最大数据传输率16MB/s。
- **EISA (Extended Industrial Standard Architecture)**：32位数据总线，32位地址总线，总线时钟频率8MHz，最大数据传输率33MB/s。
- **PCI (Peripheral Component Interconnect)**：32位数据总线，可扩展到64位，32位地址(或64位地址)总线，总线时钟频率33MHz ~ 66MHz，相应的数据传输率132MB/s~528MB/s。
- **PCI-X**：64位数据线，总线时钟频率66~1066MHz，数据传输率428MB/s~8.6GB/s，兼容PCI
- **PCI-Express**：串行差分传输，点对点通信机制。有x1、x2、x4、x8、x16和 x32多种线宽，频率2.5GHz，x32最大数据传输率可达10GB/s，软件兼容PCI和PCI-X
- **USB (Universal Serial Bus)** 总线：通用串行总线。
 - **USB 1.0**：数据传输率1.5Mbps ~12Mbps
 - **USB 2.0**：数据传输率最高可达480Mbps

1.2 总线结构

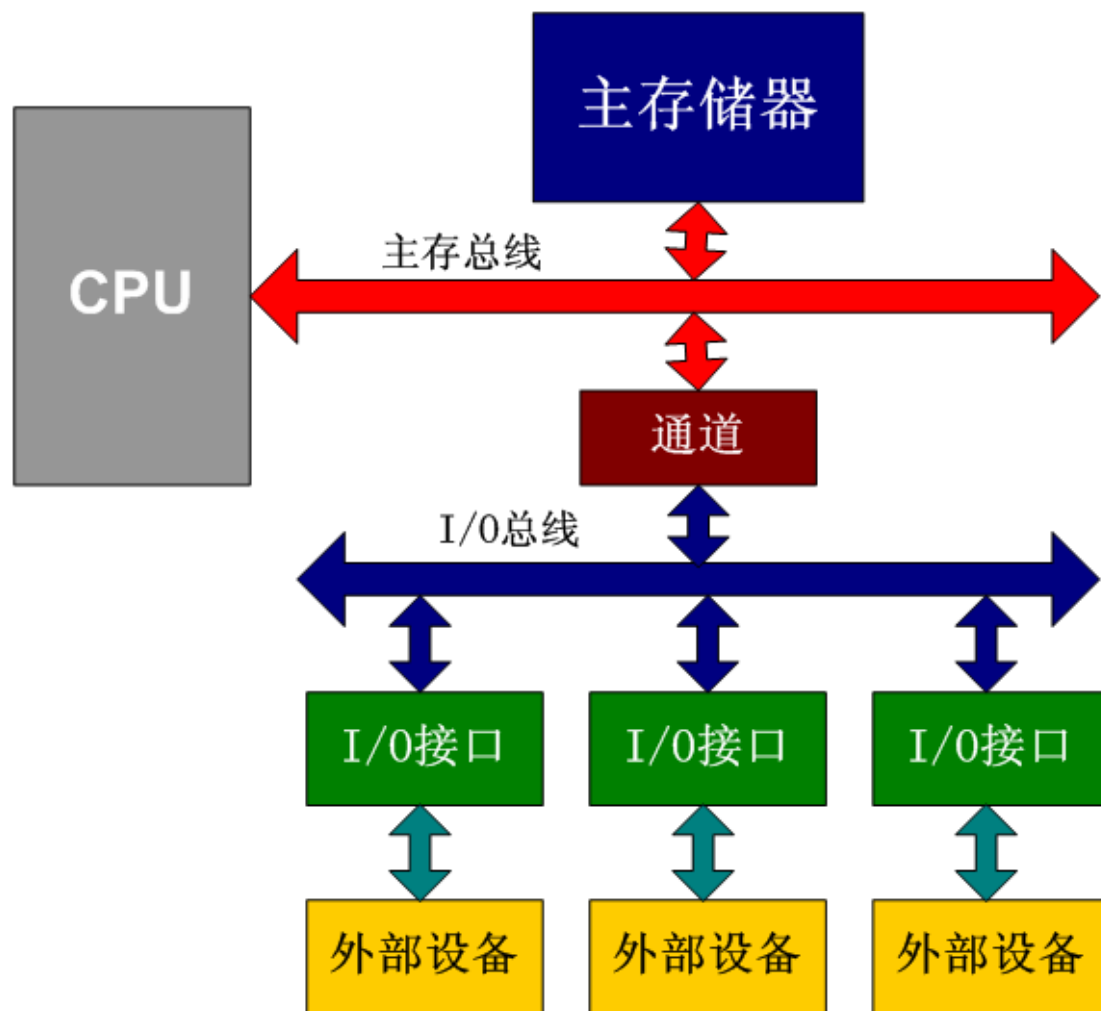
❖ 单总线结构



1.2 总线结构

❖ 多总线结构

双总线结构



1.3 总线的通信过程

❖ 总线的一次信息传送过程，大致可分五个阶段：

- 请求总线：由需要使用总线的部件或设备，提出总线使用申请
- 总线仲裁：仲裁器决定下一传输周期的总线使用权是否授予该部件或设备
- 寻址：获得总线使用权的部件或设备，发出地址和有关命令
- 信息传送：进行数据传输
- 状态返回：该部件或设备有关信息从总线上撤除，让出总线使用权

❖ 整个过程涉及两个方面的控制：

- 总线仲裁
- 通信控制

1.4 总线的仲裁（控制）方式

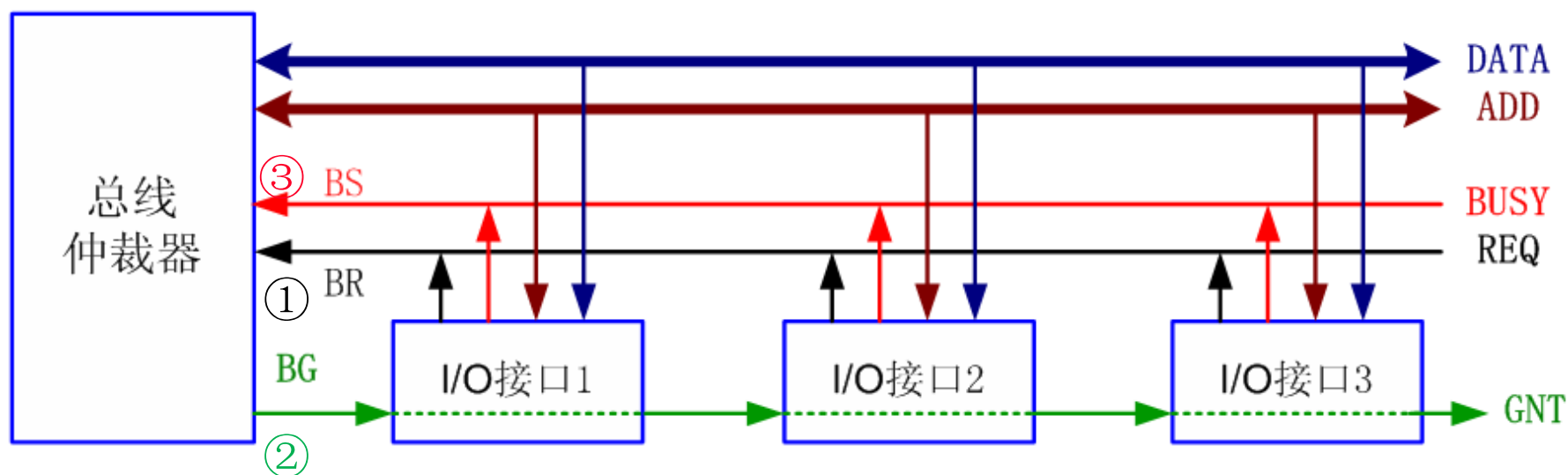
❖ 总线仲裁（控制）方式

- 总线仲裁的策略：优先级或公平
- 分布式的控制方式：总线仲裁逻辑分散在与总线连接的各部件上
- 集中式的控制方式：总线仲裁逻辑集中在一处（如在CPU中）
 - 链式查询控制方式
 - 计数器定时查询方式
 - 独立请求方式

1.4 总线的仲裁（控制）方式

❖ 链式查询方式

- 接口向总线控制器（仲裁器）发出总线申请**BR**（总线申请信号）；
- 总线仲裁器收到总线申请**BR**，将**BG**（总线同意信号）逐个往下传；
- 遇到某接口有总线申请，**BG**停止往下传；该接口获得总线使用权，并建立总线忙信号**BS**（总线忙信号）。



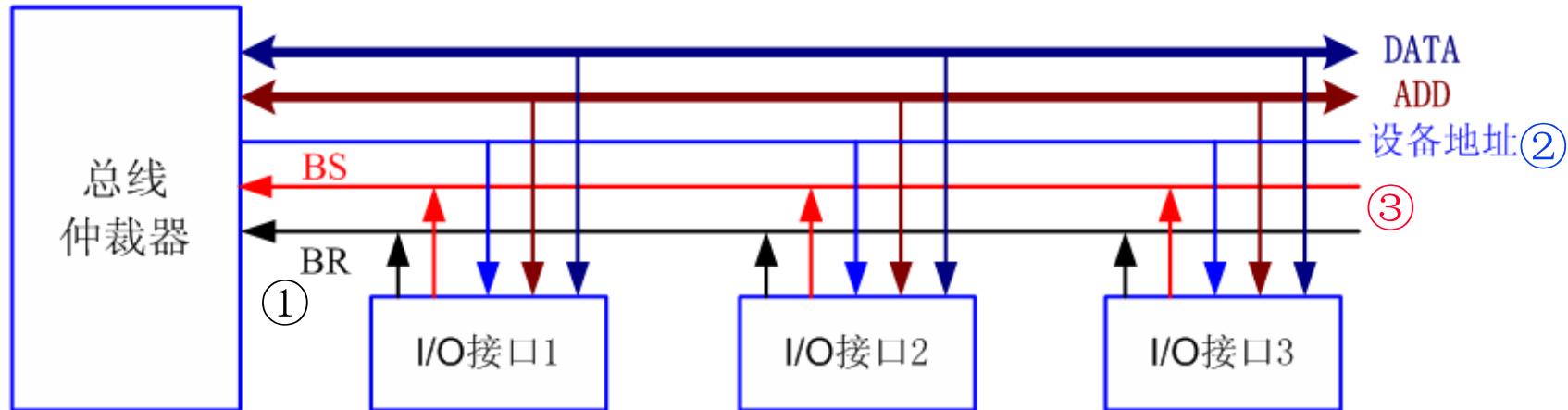
优点：线少、易扩充。

缺点：故障敏感、优先级固定。

1.4 总线的仲裁（控制）方式

❖ 计数器定时查询方式

- 总线仲裁器收到总线申请**BR**，计数器开始计数；
- 当某个有总线申请的设备地址与计数器一致，便获得总线使用权，并建立总线忙信号**BS**。



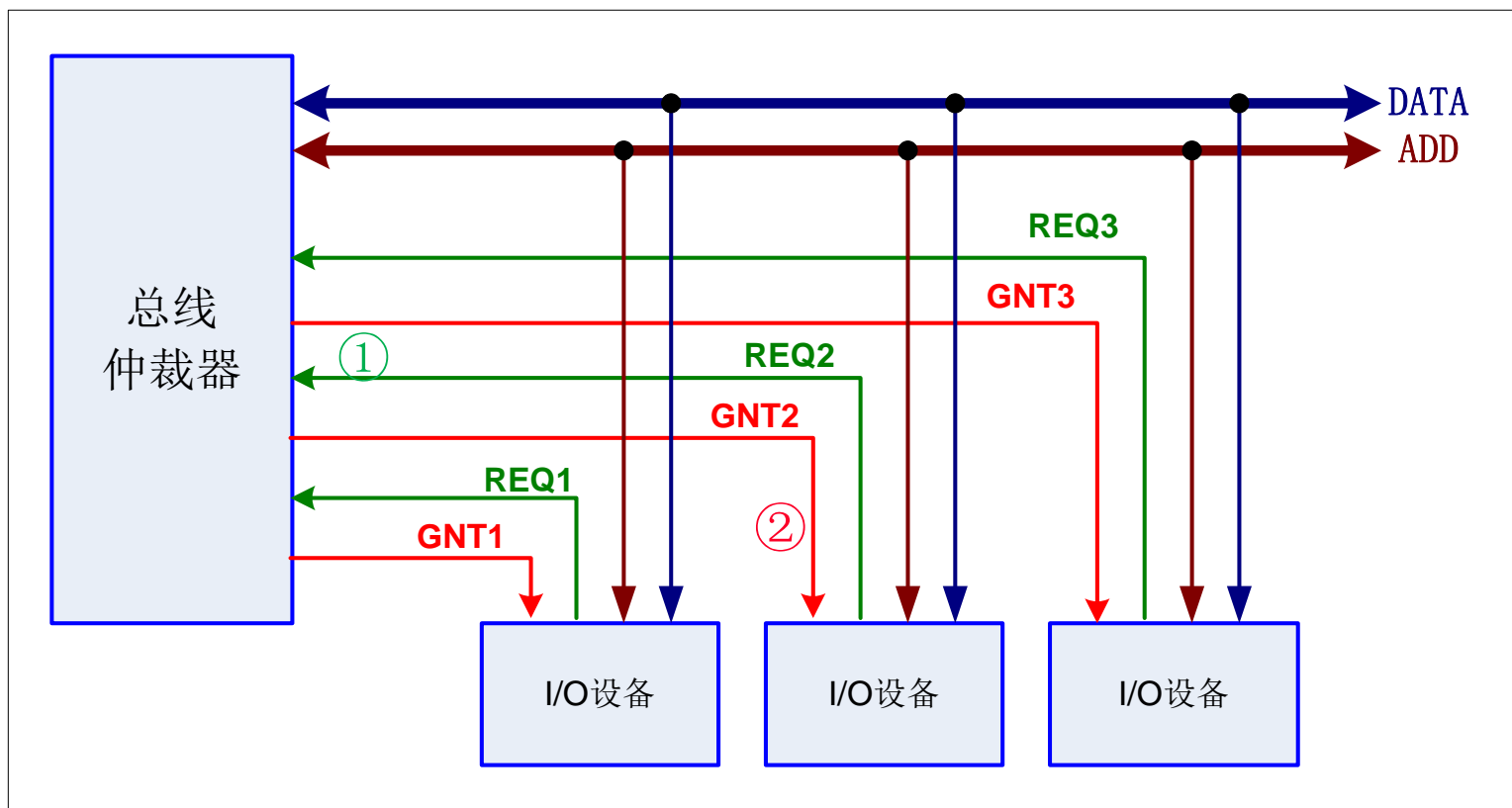
优点：优先级灵活。

缺点：线多。

1.4 总线的仲裁（控制）方式

❖ 独立请求方式

- 每个设备有独立的请求信号和总线同意信号；
- 总线控制器根据设备的优先级决定将总线的使用权交给哪个设备。



优点：响应快，优先级灵活，请求可屏蔽。

缺点：线多。

1.5 总线的通信控制方式

❖ 总线的一次信息传送过程，大致可分五个阶段：

- 请求总线：由需要使用总线的部件或设备提出总线使用申请
- 总线仲裁：仲裁器决定下一传输周期的总线使用权是否授予该部件或设备
- 寻址：获得总线使用权的部件或设备，发出地址和有关命令
- 信息传送：进行数据传输
- 状态返回：该部件或设备有关信息从总线上撤除，让出总线使用权

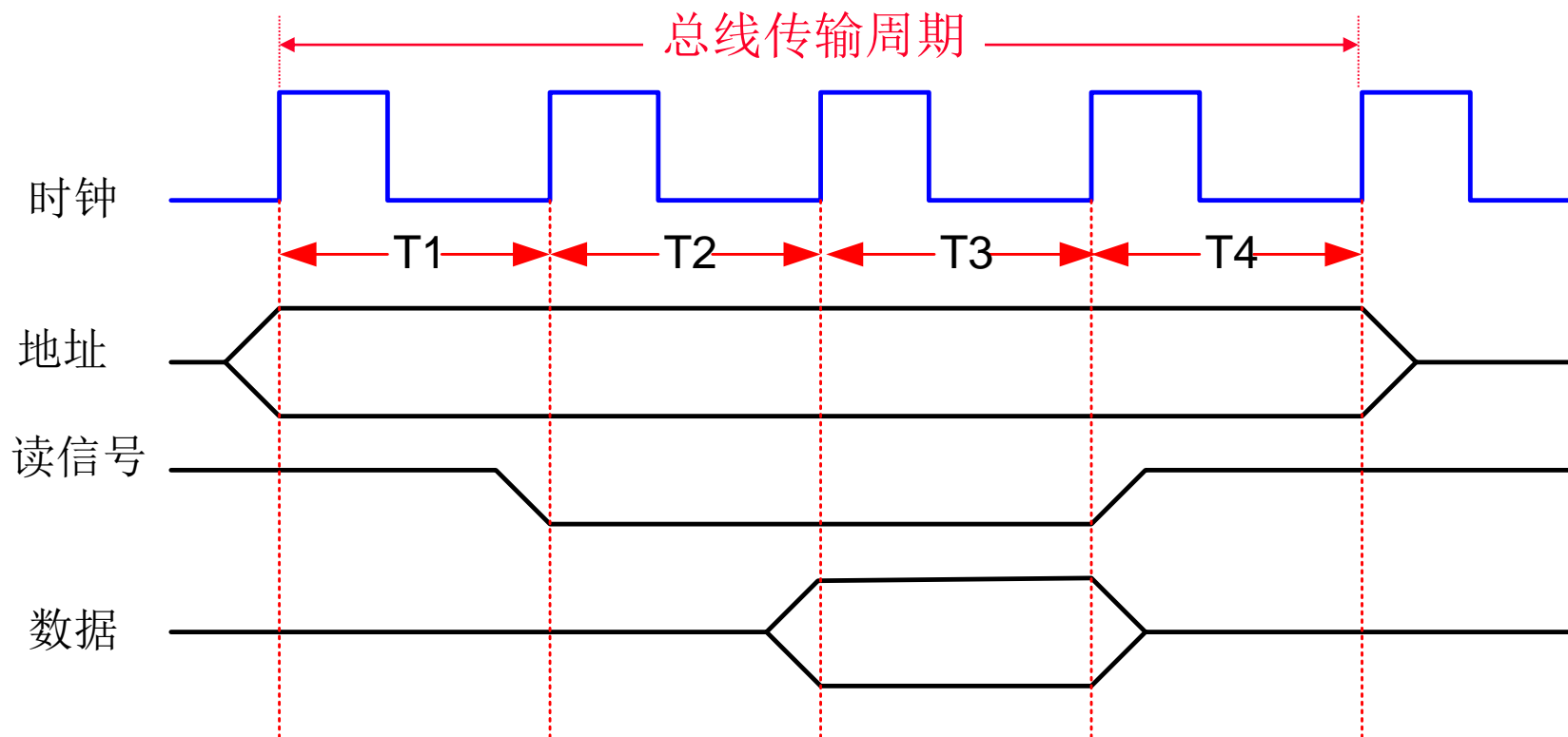
❖ 为协调通信双方，需进行通信控制，常见的方式有：

- 同步通信控制方式
- 异步通信控制方式

1.5 总线的通信控制方式

❖ 同步通信控制方式

➤ 数据传输在一个统一的时钟同步信号的控制下进行

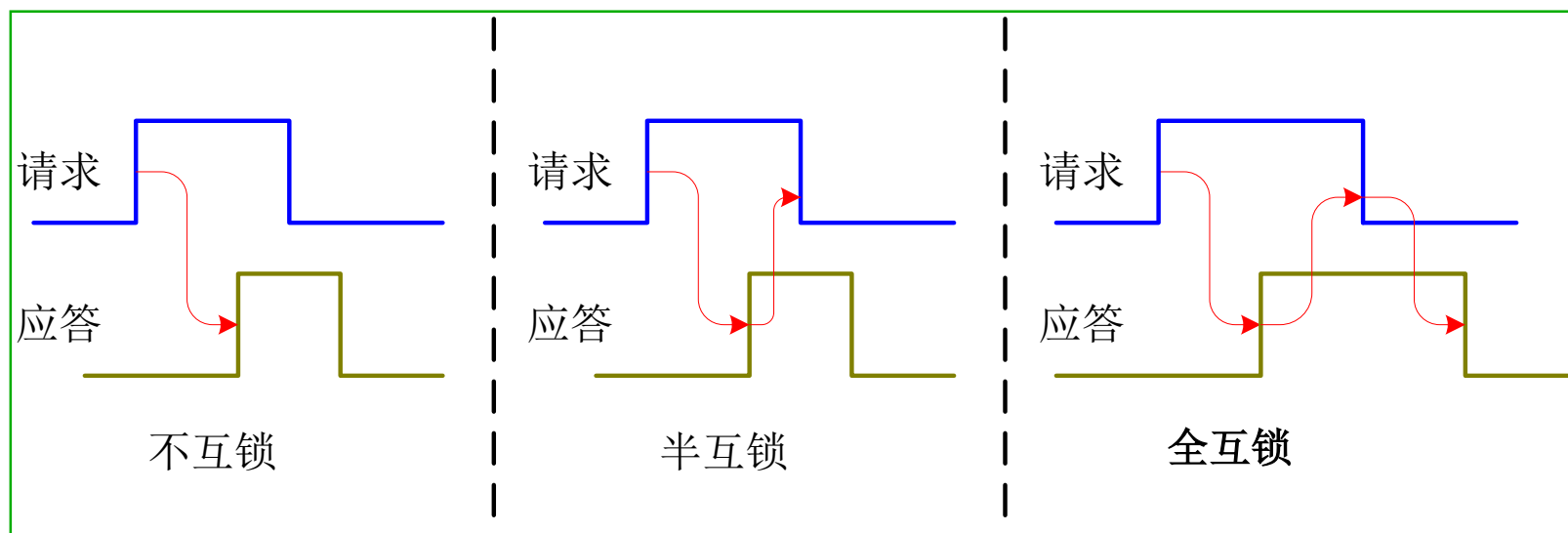


同步通信控制时序

1.5 总线的通信控制方式

❖ 异步通信控制方式

- 不需要统一的公共时钟信号，没有固定的总线周期；
- 采用请求/应答方式完成数据传输；
- 有全互锁、半互锁和不互锁三种时序。



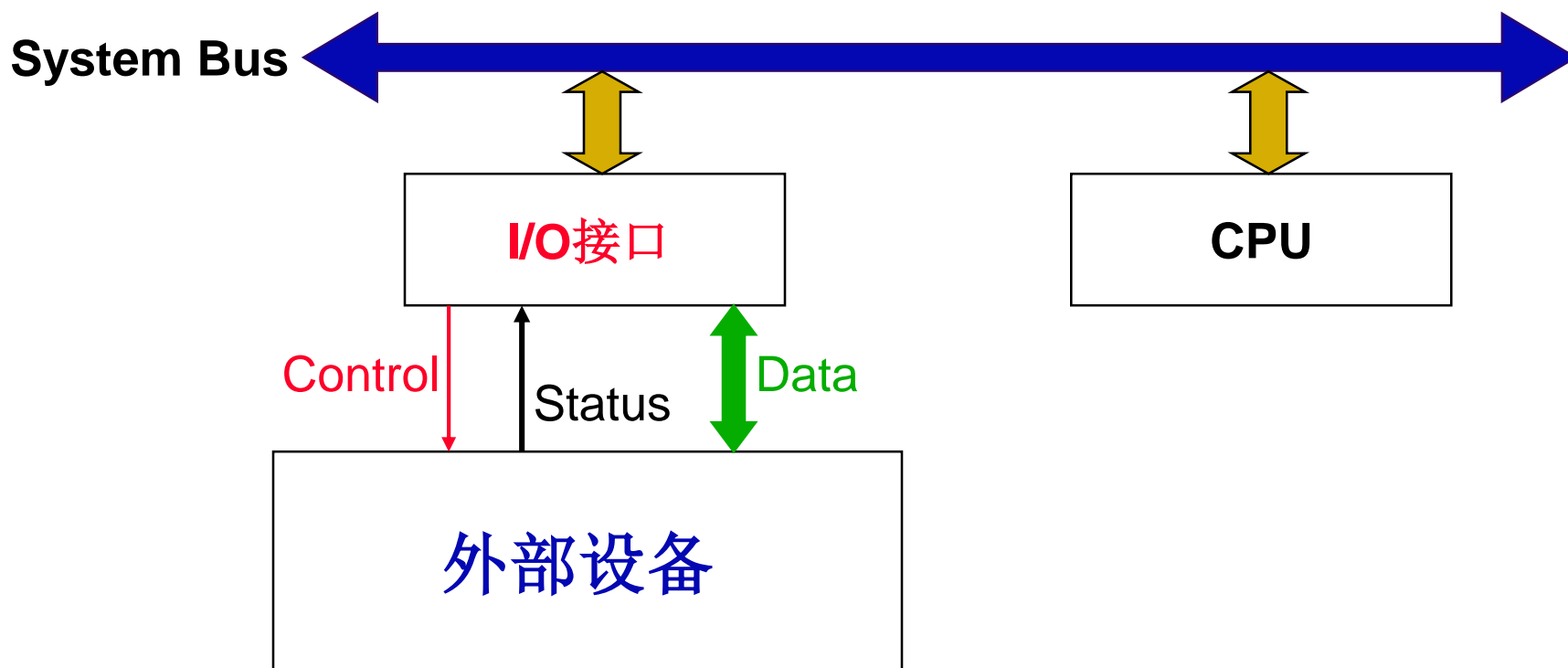
第六部分 总线与I/O

- 一、总线
- 二、I/O接口
- 三、程序查询I/O方式
- 四、中断与中断I/O方式
- 五、DMA I/O方式
- 六、I/O通道

2.1 I/O 接口

❖ 外部设备并不直接挂接在系统总线上, 而是通过I/O接口为桥梁实现与系统总线的连接

- ▶ 各种外设使用不同的操作方法, 由CPU来直接控制不同的外设不切实际。
- ▶ 外设的数据传送速度比存储器和处理器的速度慢得多, 使用高速的系统总线与慢速的外设直接连接, 不切实际。
- ▶ 外设经常使用与处理器不同的数据格式和字长度。



2.1 I/O接口

❖ I/O接口的功能

- 识别I/O地址，即地址译码
- 实现主机与I/O设备的数据交换、控制命令的传递、状态检测与传递
- 支持一定的I/O方式（程序查询、程序中断、DMA等）
- 提供缓冲、暂存、驱动能力
- 进行数据格式、类型方面的转换（串并行转换，电平转换等）
- I/O控制与定时

2.1 I/O接口

❖ I/O 接口的分类

➤ 按传送数据格式：串行接口，并行接口

- 串行接口：适合速度低、传输距离长的环境
- 并行接口：适合速度高、传输距离短的环境

➤ 按I/O方式：

- 程序查询接口、中断接口、**DMA**接口、通道控制接口

➤ 按时序控制方式：同步接口、异步接口

- 同步接口：数据传送由一个统一的时钟信号同步控制
- 异步接口：数据传送采用异步应答方式控制

2.1 I/O接口

❖ I/O 操作的过程

- CPU查询I/O接口状态，以检查其连接设备的状态
- I/O接口回送设备状态给CPU
- 如果设备状态显示设备可用，并准备好，CPU向I/O接口发出命令，请求传送
- I/O接口获得来自外设的数据（字或字节）
- 数据从I/O接口传送至CPU

2.1 I/O接口

❖ I/O设备的编址

➤ I/O接口的编址

➤ I/O地址（I/O接口地址, I/O端口地址）：

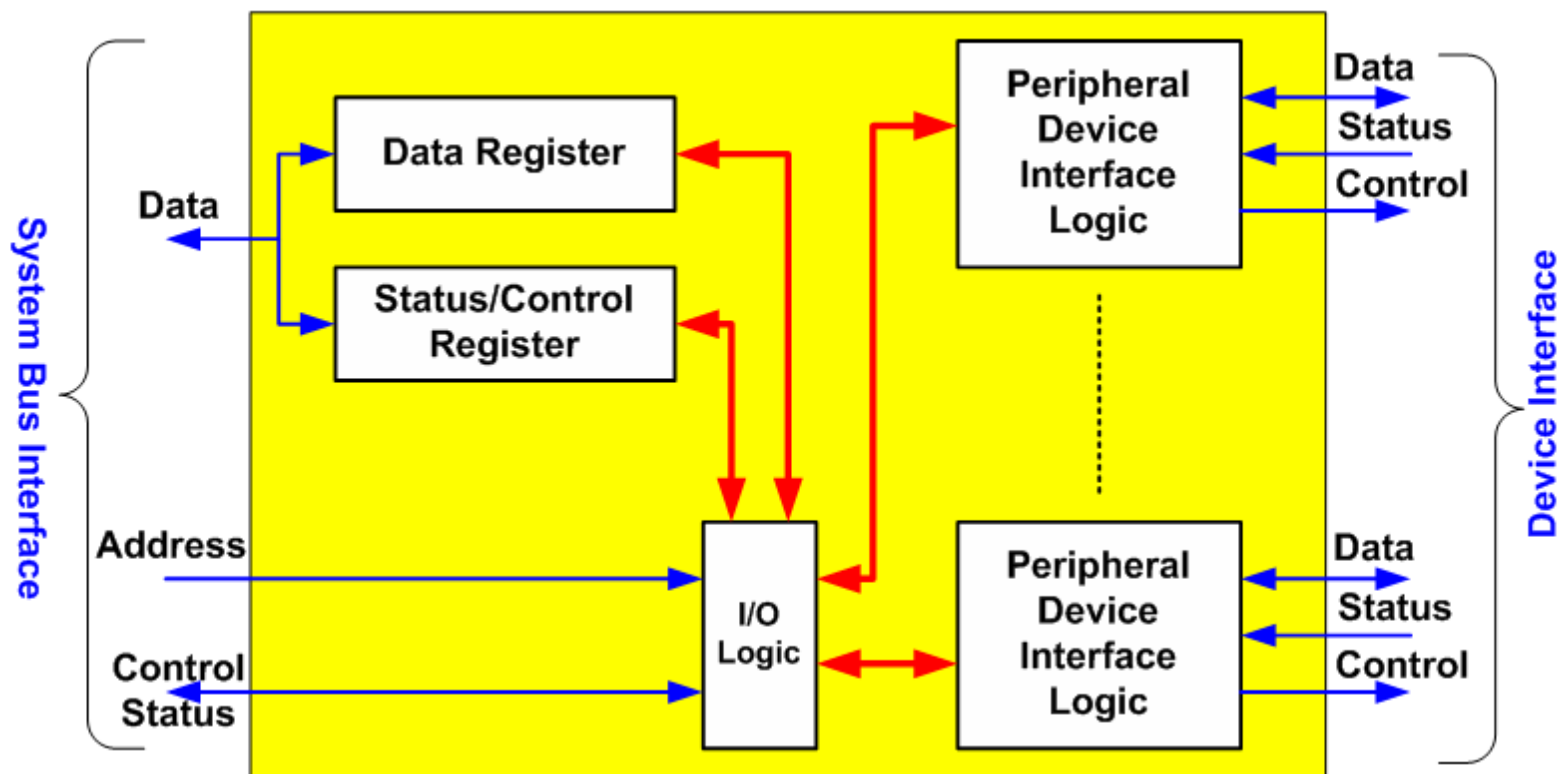
- 实际上是I/O接口电路中寄存器的地址

➤ 编址方式

- **独立编址方式**：存储器地址与I/O地址分开，CPU具有专用的I/O指令，系统总线中具有区别存储器读写和I/O操作的控制信号，并以此区别地址总线上的地址是存储器地址还是I/O地址
- **统一编址方式**：存储器地址与I/O地址统一考虑，地址空间的一部分是存储器，另一部分是I/O，支持存储器操作的指令都可用于I/O操作

2.1 I/O接口

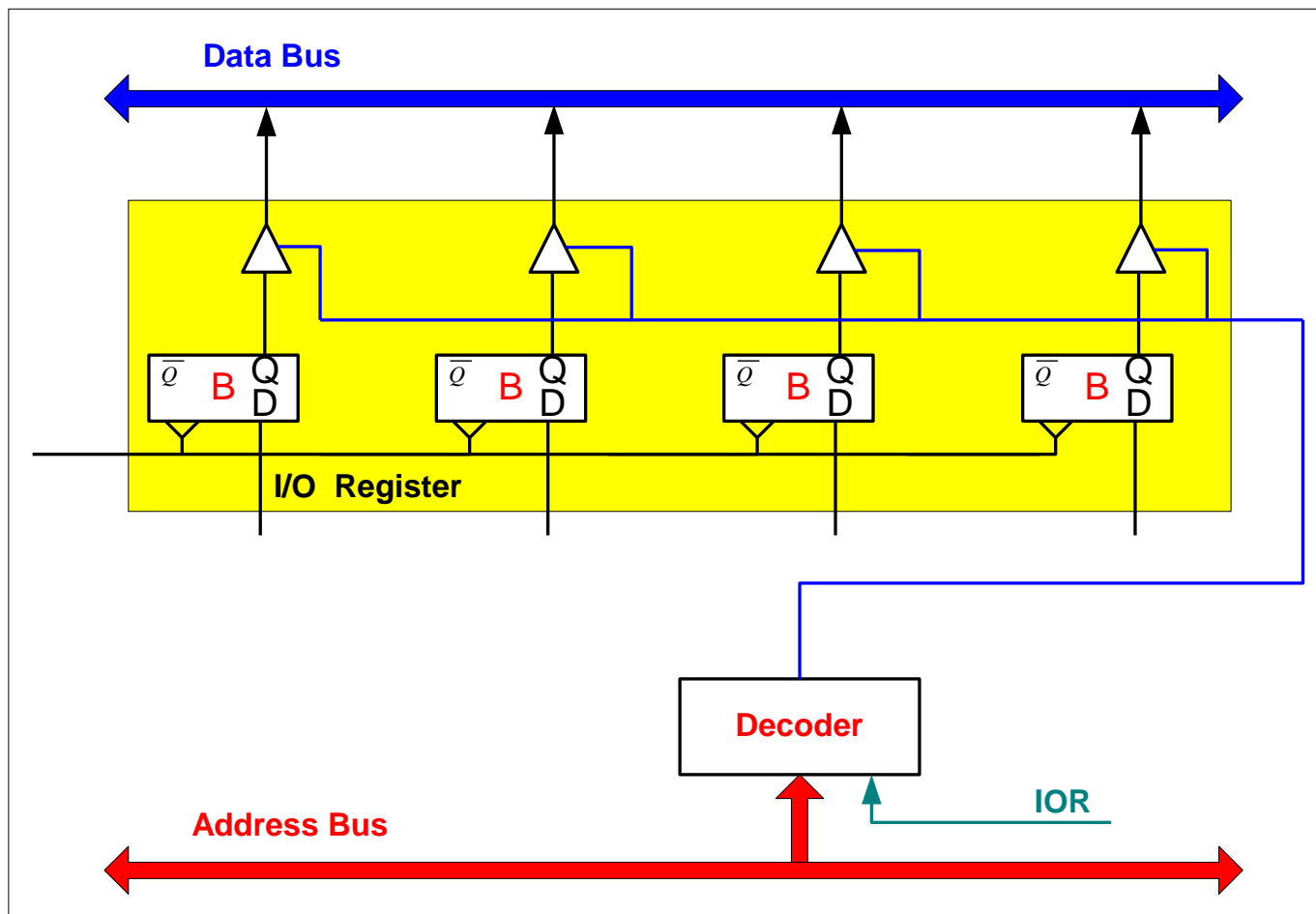
❖ I/O接口的结构



2.1 I/O接口

❖ I/O接口地址选择（译码）

- I/O接口地址是I/O接口电路中寄存器的地址
- 独立编址方式下的I/O地址选择电路



第九讲 总线与I/O

- 一、总线
- 二、I/O接口
- 三、程序查询I/O方式
- 四、中断与中断I/O方式
- 五、DMA I/O方式
- 六、I/O通道

3.0 I/O与主机信息交换的控制方式

- ❖ [例] 假设幼儿园一个老师带 9 个孩子，要给每个孩子分 3 块水果糖，并且要孩子们把 3 块糖都吃完，那么她可以采用什么方法呢？
- ❖ 方法1： 她先给孩子甲一块糖，盯着甲吃完，然后再给第二块，等吃完第二块又给第三块。接着给孩子乙，其过程与孩子甲完全一样。依此类推，直到给第 9 个孩子发完 3 块糖。这种方法效率太低，问题的关键在于，孩子们吃糖时老师一直在守候，没法做其它工作。
- ❖ 方法2： 给每个孩子发一块糖各自去吃，并约定谁吃完后就向老师举手报告，再发第二块，……。看来这种新方法提高了工作效率，而且在未接到孩子们吃完糖的报告时，老师可腾出时间做其它工作。但是这种方法还可以改进。
- ❖ 方法3： 进行批处理：给每个孩子拿 3 块糖各自去吃，吃完 3 块糖后再向老师报告。显然这种方法工作效率大大提高，老师可以腾出更多的时间做其它工作。
- ❖ 方法4： 权力下放：把发糖的事交给另一个人分管，只是必要时老师才过问一下。

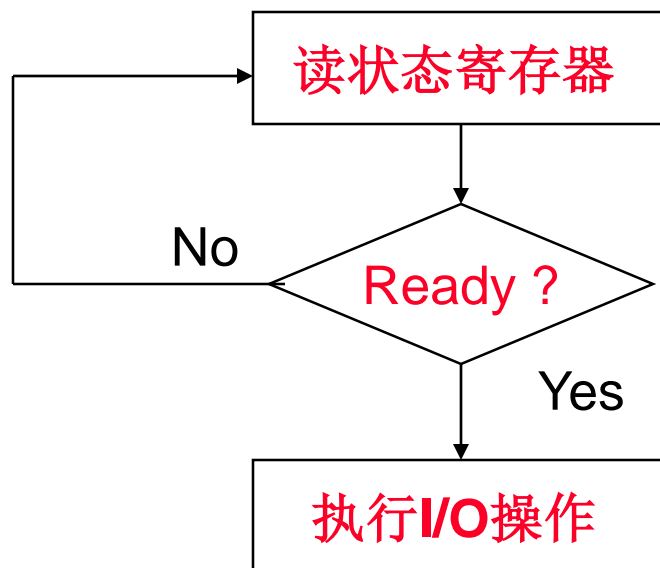
3.0 I/O与主机信息交换的控制方式

- ❖ 程序查询方式
- ❖ 程序中断方式
- ❖ 直接内存访问 (DMA) 方式
- ❖ 通道方式

3.1 程序查询I/O方式

- ❖ I/O接口设置状态寄存器，以表示外部设备的工作状态
- ❖ CPU通过不断读取状态寄存器，以查询外部设备的状态
- ❖ 在外部设备准备就绪的时候，CPU通过I/O接口中的数据寄存器，与外部设备之间完成数据交换

```
RdSta:  MOV DX,3FDH  
        IN  AL,DX   ; 读状态  
        CMP AL,61H  
        JNE RdSta  
        MOV DX,3F8H  
        IN  AX,DX   ; 读数据
```



3.1 程序查询I/O方式

❖ 程序查询I/O接口的基本组成

- Status Register
- Data Register (Input Register, Output Register)
- Address Selected Logic
- Bus Interface Logic

❖ 程序查询I/O方式的特点

- I/O操作全部由**CPU**直接完成（通过执行I/O指令完成）
- 外设速度慢，**CPU**速度快，在外设准备过程中，**CPU**处在不断的查询之中，极为浪费**CPU**的性能
- 外设与**CPU**完全串行工作，**CPU**效率低

例题

- ❖ 在程序查询方式的I/O系统中，假设不考虑处理时间，每一次查询操作需要100个时钟周期，CPU的时钟频率为50Mhz，现有鼠标和硬盘两个设备，而且CPU必须每秒对鼠标进行30次查询；硬盘以32位字长为单位传输数据，即每32位被CPU查询一次，传输率为2MBps。求CPU对这两个设备查询所花费的时间比率，由此可得到什么结论？
- ❖ 每秒内CPU的时钟周期数： $1/(1/50\text{Mhz}) = 50 \times 10^6$ 个
- ❖ 对鼠标查询，每秒所需时钟周期数： 30×100 个
相应的时间比率： $(30 \times 100) / (50 \times 10^6) = 0.006\%$
- ❖ 对硬盘查询，每秒所需时钟周期数： $(2\text{MB}/4\text{B}) \times 100$ 个
相应的时间比率： $((2\text{MB}/4\text{B}) \times 100) / (50 \times 10^6) = 100\%$

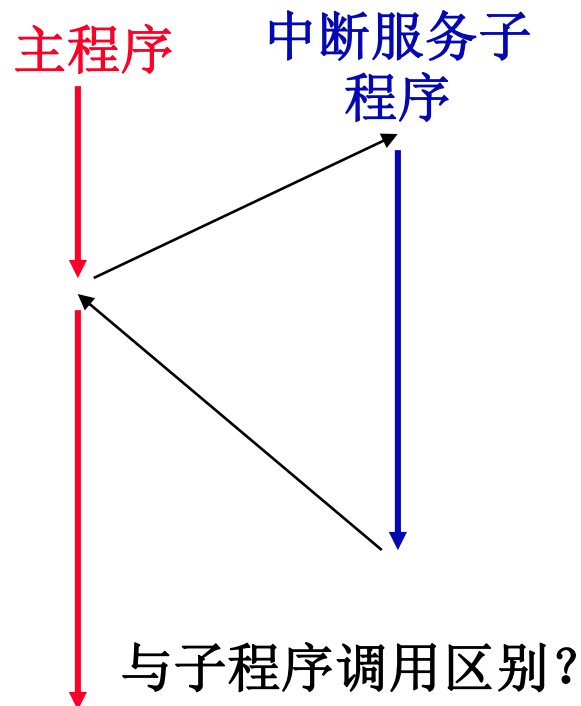
第九讲 总线与I/O

- 一、总线
- 二、I/O接口
- 三、程序查询I/O方式
- 四、中断与中断I/O方式
- 五、DMA I/O方式
- 六、I/O通道

4.1 中断与中断I/O方式

❖ 中断的概念

- 概念：机器出现紧急事务，**CPU**不得不停下当前正在执行的程序，转去处理紧急事务，事务处理完后，再继续执行被中断的程序
- 作用：主机与外设并行、实时处理和过程控制、硬件故障处理、多道程序和分时操作
- 一般情况下，中断是随机的
- 主程序：被中断的程序
- 中断服务子程序：处理中断事务的程序
- 中断向量：中断服务子程序的入口地址
- 中断向量表：保存所有中断向量的内存区域，一般固定。



4.1 中断与中断I/O方式

❖ 引起中断的因素（中断源）

- 人为设置的中断：自愿中断，可重复
- 程序性事故：如溢出、除“零”等
- 硬件故障：如电源掉电、磁盘损坏
- I/O操作：I/O设备准备就绪，请求操作
- 外部事件：如键盘操作

❖ 中断源分类

- 不可屏蔽中断：**CPU**不能不响应；
- 可屏蔽中断：若中断源被屏蔽，**CPU**不响应

❖ 中断的分类

- 非屏蔽中断与可屏蔽中断
- 程序中断与简单中断
- 硬中断与软中断（软中断不是真正的中断）

4.1 中断与中断I/O方式

❖ 中断系统需要解决的主要问题

- 各中断源如何向**CPU**提出中断请求
- 当多个中断源同时提出中断请求时，中断系统如何确定优先响应哪个中断源的请求
- **CPU**在什么条件、什么时候、以什么方式来响应中断
- **CPU**响应中断后如何保护现场
- **CPU**响应中断后，如何停止原程序的执行而转入中断服务程序的入口地址
- 中断处理结束后，**CPU**如何恢复现场，如何返回到原程序的间断处
- 中断处理过程中出现新的中断申请怎么处理

4.1 中断与中断I/O方式

❖ 中断请求

➤ 中断请求触发器（INTR）：

- 每个中断源配置一个中断请求触发器
- 中断源可通过设置中断请求触发器来提出中断申请

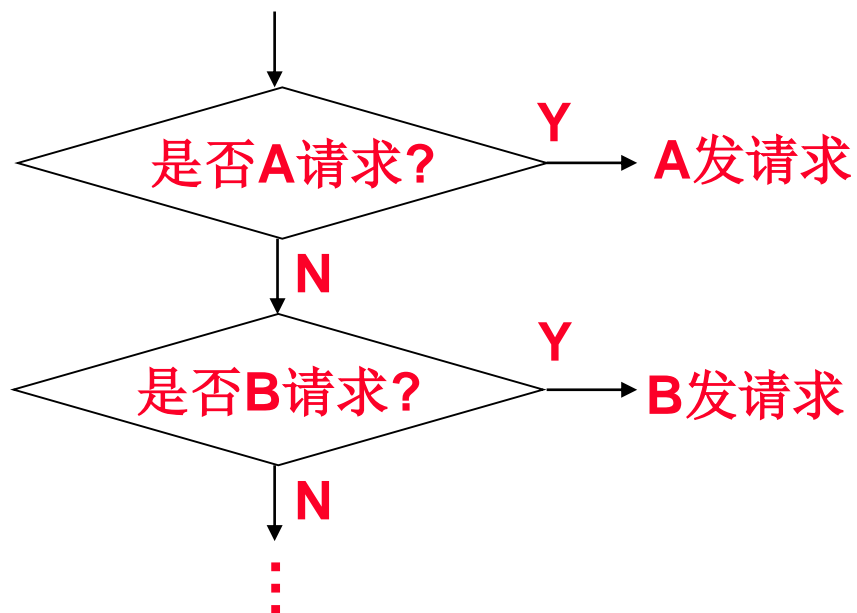
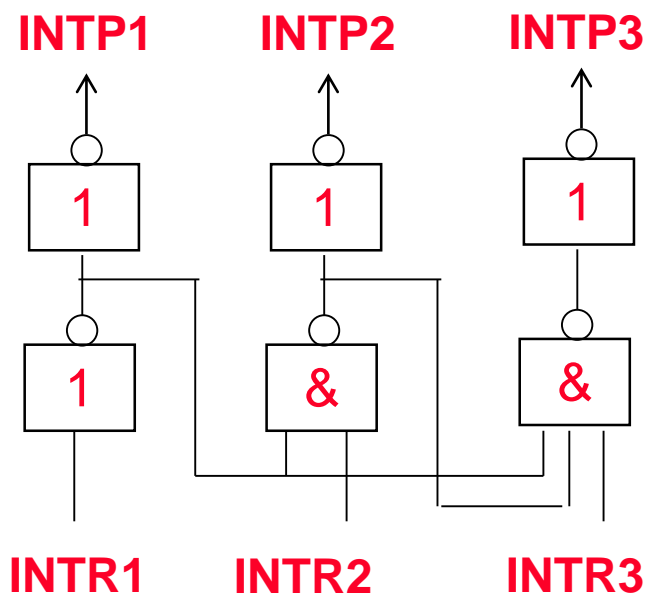
➤ 中断请求标记寄存器：各中断源的请求触发器组成中断请求标记寄存器；

1	2	3	4			n
掉电	过热	主存读写检验错	溢出		键盘输入	打印机输出

4.1 中断与中断I/O方式

❖ 中断判优逻辑

- 中断系统任何时刻最多只能响应一个中断源请求
- 硬件排队判优
- 软件排队判优



4.1 中断与中断I/O方式

❖ 中断响应

- 条件：当前执行的程序允许被中断（即中断允许标志位为允许中断），不可屏蔽中断不受中断允许标志位的限制
- 时机：当前指令执行完后，才能响应中断
- 方式：在允许中断的前提下，每条机器指令的执行周期中实际上包含一个中断周期，在需要时执行中断隐指令

4.1 中断与中断I/O方式

❖ 中断处理

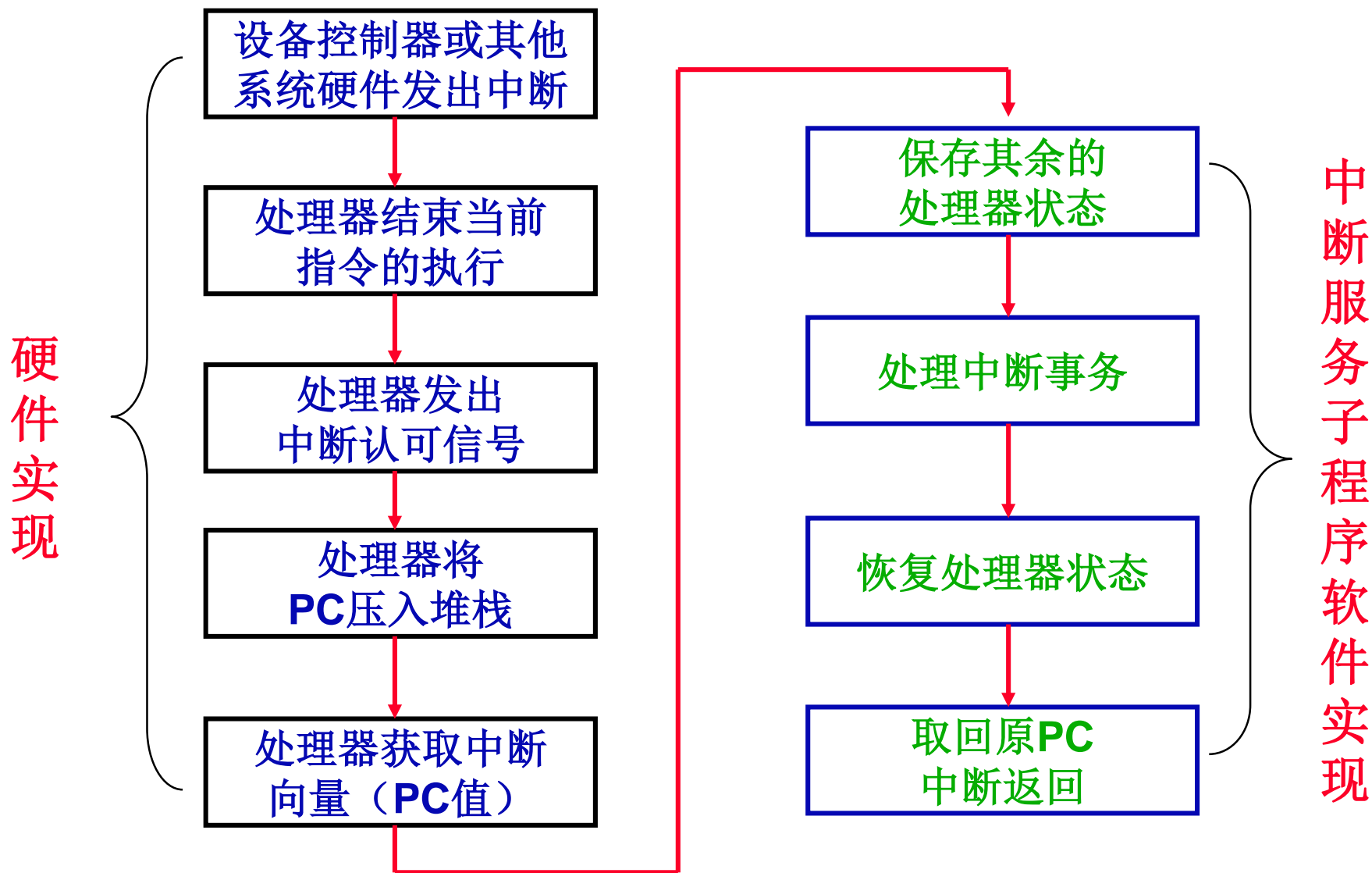
➤ 硬件自动执行中断隐指令

- 保护程序断点：程序计数器**PC**内容入栈；
- 跳转至中断服务子程序：查找中断服务子程序入口地址（中断向量）送**PC**，转向中断服务子程序；
- 关中断。

➤ CPU执行中断服务子程序

- 执行中断服务处理功能；
- 从中断服务子程序中返回：恢复程序断点，即把保存在堆栈中的**PC**内容弹出送**PC**，接下来继续执行主程序。

4.1 中断与中断I/O方式



4.1 中断与中断I/O方式

❖ 多重中断（自学）

- 多重中断的概念
- 实现多重中断的条件
- 中断屏蔽触发器
- 中断屏蔽字
- 中断屏蔽字与中断优先级的关系
- 中断处理次序与中断屏蔽字的关系
- 多重中断的断点保护

4.1 中断与中断I/O方式

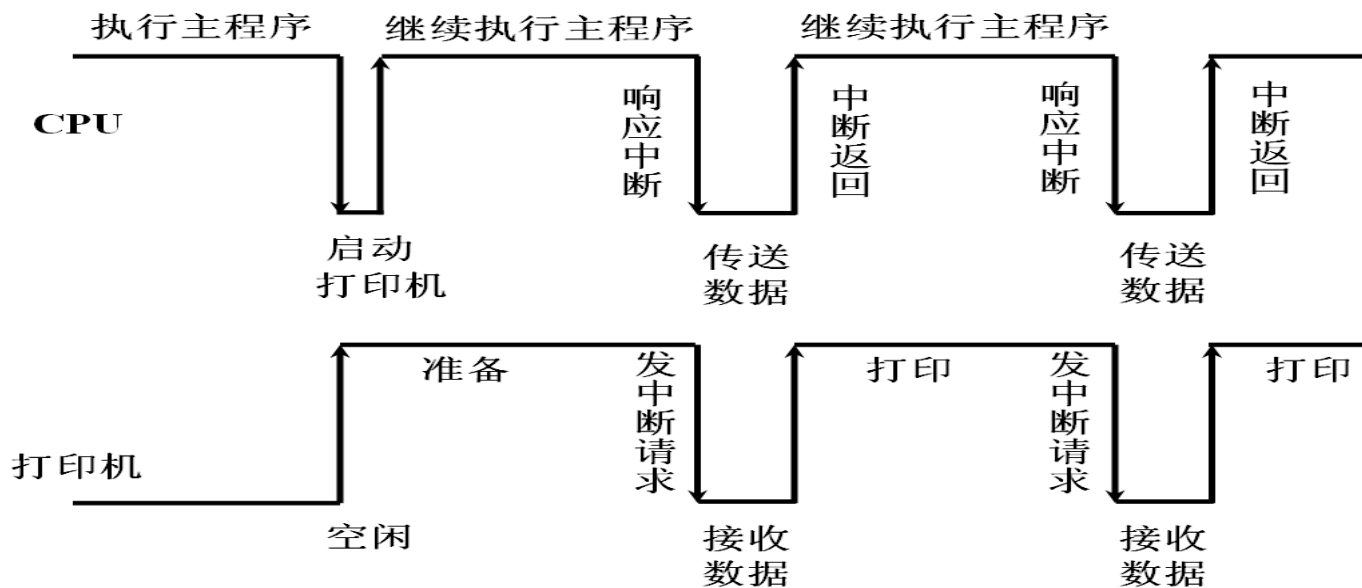
❖ 中断I/O接口的基本组成

- Data Input Register
- Data Output Register
- Status Register
- Control Register
- Address Selected Logic
- Interrupt Control Logic
- Bus Interface Logic

4.1 中断与中断I/O方式

❖ 中断I/O方式的特点

- I/O操作仍然由CPU通过I/O指令完成
- 在外设准备阶段，CPU可以执行其他程序，仅在外设准备就绪后，CPU才中断正在执行的程序，处理I/O事务
- 在外设准备阶段，CPU与外设的工作可以认为是并行的
- 中断I/O方式是目前最主要的I/O方式



思考题

- ❖ 在程序中断方式中，磁盘申请中断的优先级高于打印机。当打印机正在打印时，磁盘申请中断，试问是否要将打印机的打印操作停下来，等磁盘操作结束后，打印机才能继续打印？为什么？
- ❖ 打印操作与磁盘中断的关系
- ❖ 通常，打印机的打印动作只受打印机本身控制，与CPU无直接关系，因此，当打印机正在打印时，即使有优先级更高的磁盘请求中断，打印机也不会停止打印。
- ❖ 但是，如果CPU正在执行打印机的中断服务程序，即打印机可能正在接收数据，此时，若磁盘请求中断，CPU就要中断正在运行的打印机中断服务程序，向打印机的数据传送会受到影响

第九讲 总线与I/O

- 一、总线
- 二、I/O接口
- 三、程序查询I/O方式
- 四、中断与中断I/O方式
- 五、**DMA I/O方式**
- 六、I/O通道

5.1 DMA的一般概念

❖ 程序I/O与中断I/O的不足：CPU仍需参加IO操作

- I/O传送速度受处理器测试和给设备提供服务的速度的限制
- 处理器直接负责管理I/O，对于每一次I/O传送，处理器必须执行一些指令

❖ DMA (Direct Memory Access)

- CPU对总线的控制被临时禁止。DMA控制器接管总线控制权，控制数据直接在存储器与外设之间高速交换
- CPU不再介入具体的I/O操作，由DMA控制器来负责提供存储器地址信号、读写控制信号等。
- CPU与I/O设备在更大的程度上并行工作，效率更高。
- DMA方式适合高速批量的数据传输，如视频显示刷新、磁盘存储系统的读写，存储器到存储器的传输等。

5.2 DMA 过程

❖ CPU的工作：初始化DMA控制器

- 设置数据传送方向：是请求读还是请求写（对存储器而言）
- 设置I/O接口地址：DMA操作所涉及的I/O接口的地址
- 设置存储器起始地址：读或写存储器的起始单元地址
- 设置传送的数据数量：传送数据的字数
- 有关中断方式的设置：DMA结束后通过中断方式请求CPU处理

❖ DMA请求

- 当接口做好数据传输的准备，通过有关逻辑向CPU发出DMA请求信号

❖ DMA响应

- CPU接到DMA请求，在当前总线周期操作结束后，暂停CPU对系统总线的控制和使用，发出DMA响应信号，并交出系统总线的控制权

5.2 DMA 过程

❖ DMA操作

- **DMA**控制器接到**DMA**应答信号后，通过控制逻辑向系统总线发送存储器地址信号、存储器读写控制信号、I/O接口读写控制信号等，完成一次数据传送。这些操作完全由硬件控制，一般仅需要一个总线周期，所以这种方式称为周期窃用（**cycle-stealing**）方式。
- 所有数据传送结束后，通过中断方式告知**CPU**进行善后处理。

❖ **CPU**仅在开始**DMA**操作之前和完成**DMA**操作之后参与I/O处理，在**DMA**过程中，**CPU**可以运行原来的程序

5.2 DMA 过程

❖ DMA 方式

➤ 周期窃取方式（单字传送方式）

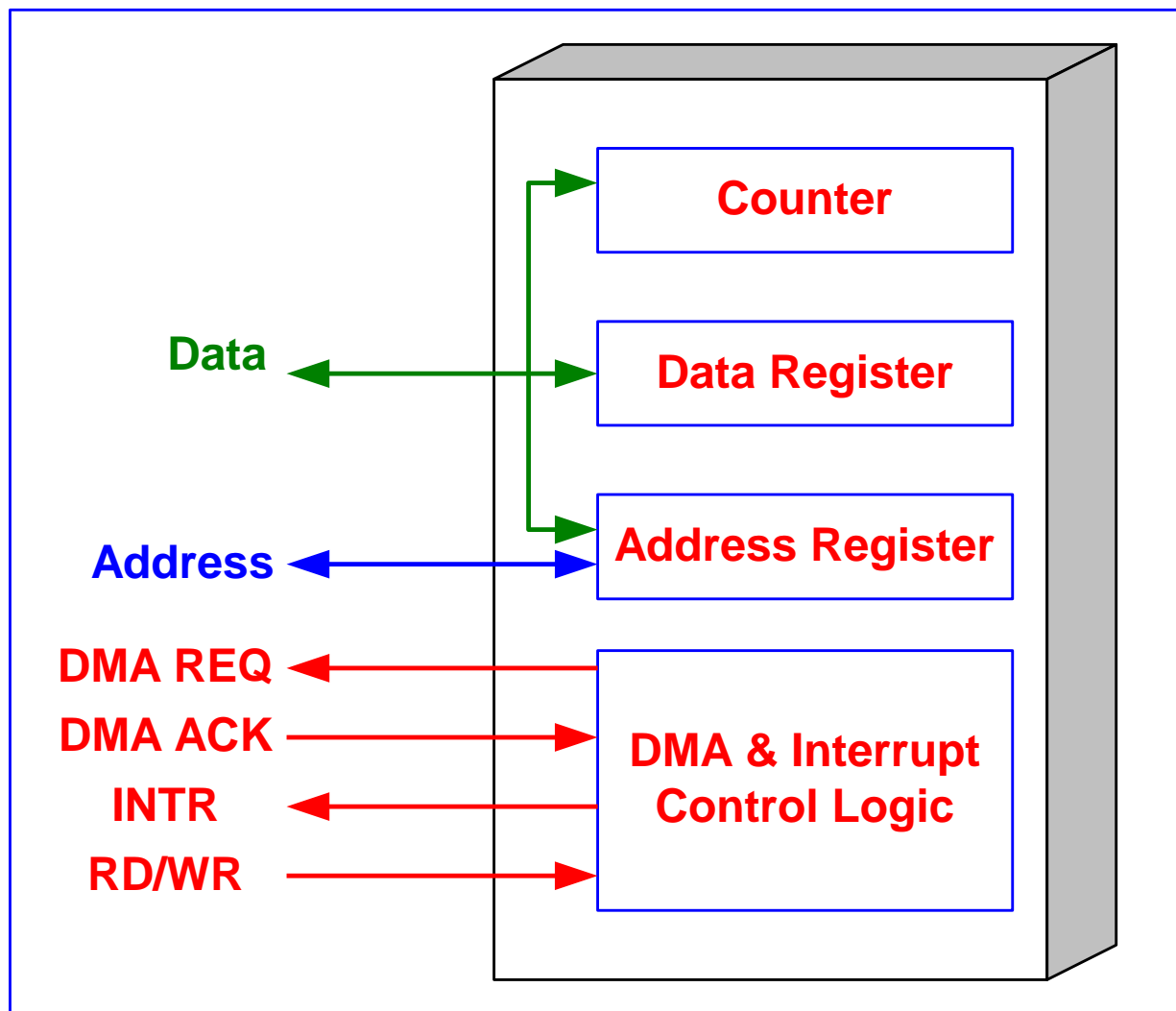
- 每次**DMA**请求得到响应后，**DMA**控制器窃取一个总线周期完成一次数据传送，然后释放总线。
- 一般适应存储器速度远高于**I/O**设备速度的情况。

➤ 停止CPU访问内存（成组传送方式）

- 一次**DMA**请求得到响应后，**DMA**控制器完全占用总线，进行多次**DMA**传送，直到所有数据传送完毕才释放总线，这段时间完全停止**CPU**访问内存。
- 适应高速外设与存储器交换数据的情况。

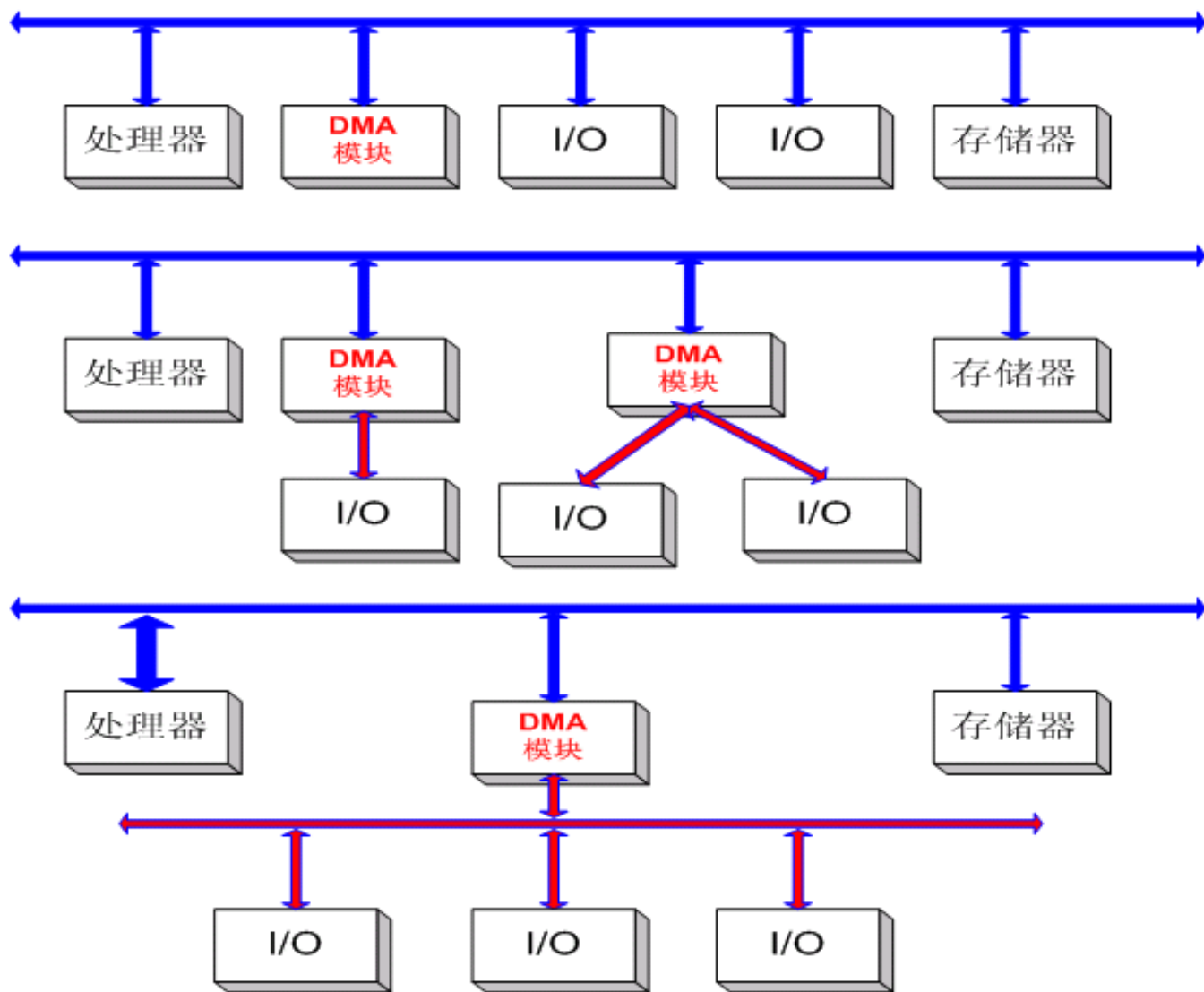
5.3 典型DMA控制器的结构

- **Counter**: 长度计数器, 保存传送数据的字数。
- **Data Reg**: 数据寄存器。
- **Address Reg**: 地址寄存器, 向地址总线提供存储器地址。
- **DMA控制逻辑**
- **DMA状态逻辑**
- **中断控制逻辑**

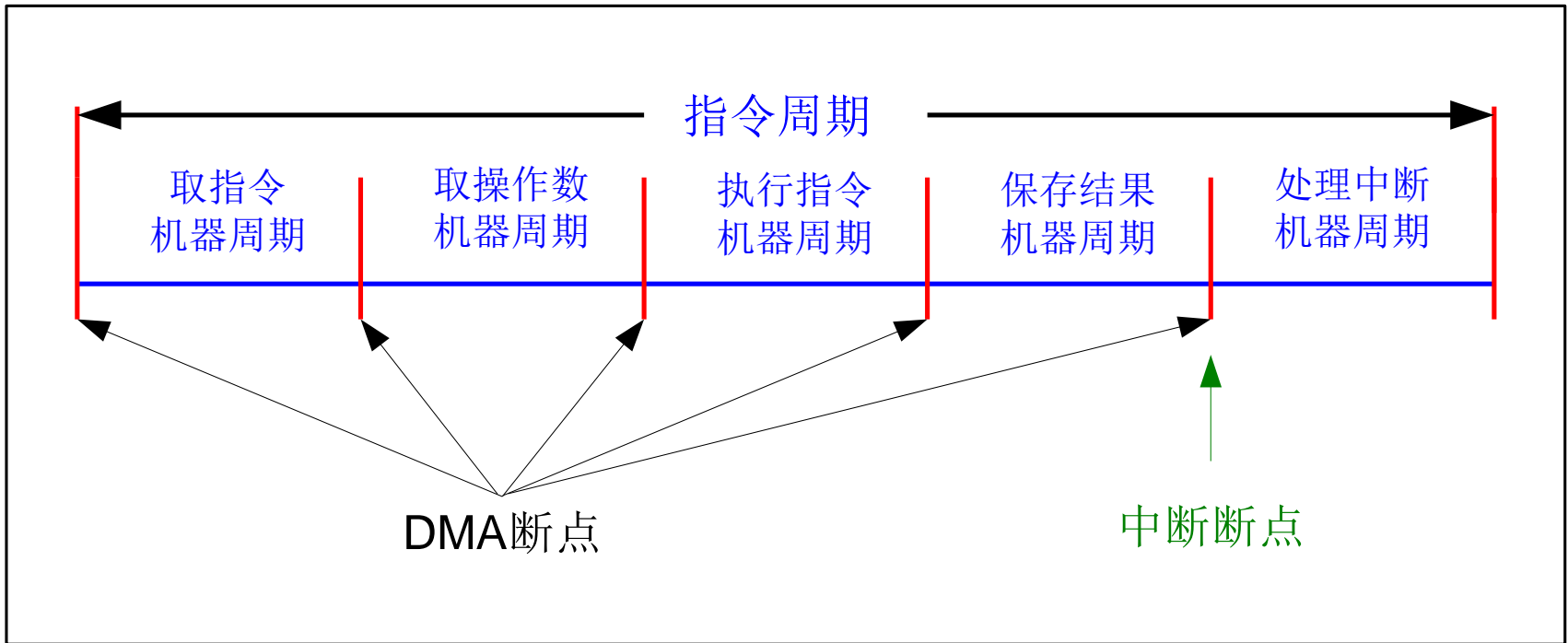


5.3 典型DMA控制器的结构

可能的DMA结构



5.4 DMA vs. 中断



- **响应时机：** 中断是在一条指令结束后响应；而DMA可在指令周期内的任一存取周期结束时响应
- **现场保护：** 中断要中断现行程序，需保护现场；而DMA不中断现行程序，无须保护现场
- **适应场合：** 中断适于处理紧急或异常事件；而DMA适于传送大批数据
- **传送方式：** 中断需要靠程序传送数据；而DMA靠硬件传送

第九讲 总线与I/O

- 一、总线
- 二、I/O接口
- 三、程序查询I/O方式
- 四、中断与中断I/O方式
- 五、DMA I/O方式
- 六、I/O通道

6.1 通道方式及其特点

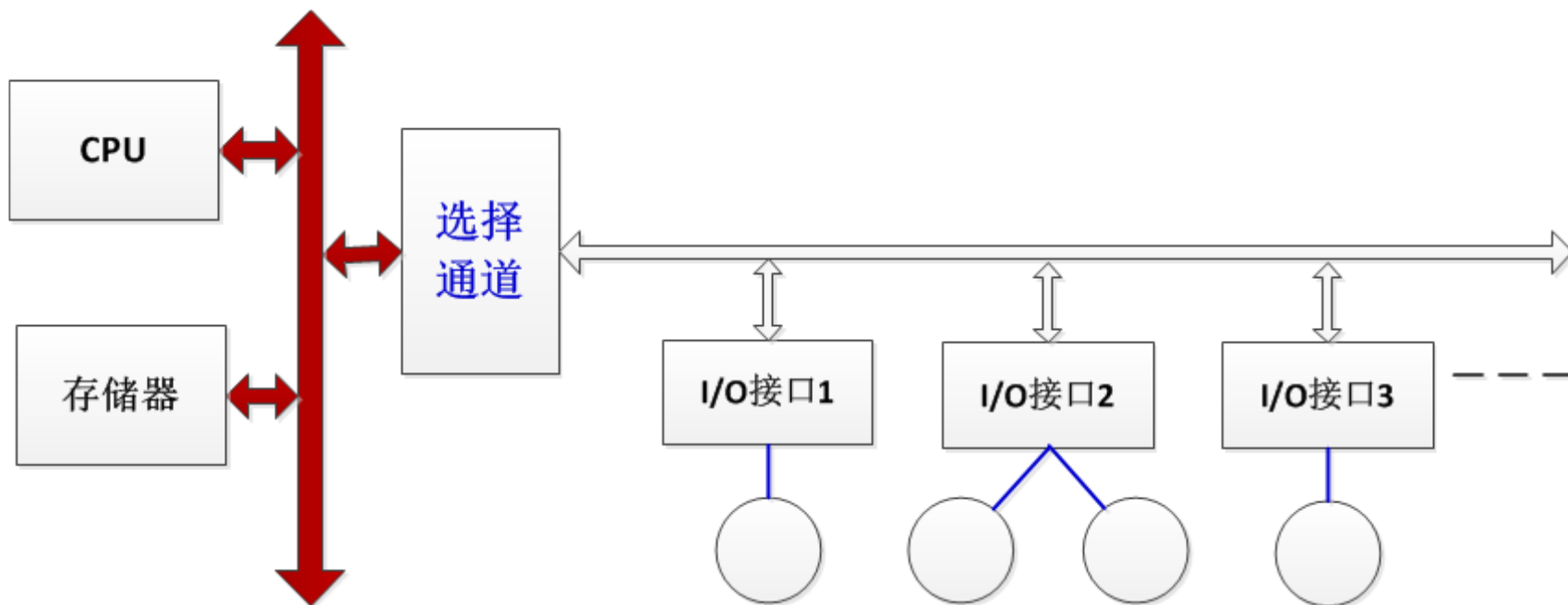
❖ 通道I/O方式的特点

- 通道是一种专业控制器，具有自己的指令系统（基本上都是I/O指令）
- 通道执行通道程序来实现和管理I/O，CPU基本上不需要管理I/O，CPU的效率得到更大的提高
- 通道程序由OS根据I/O任务的需求自动生成，存放在存储器中，通道程序由OS管理，用户程序执行和访问通道程序

6.2 通道分类

❖ 选择通道

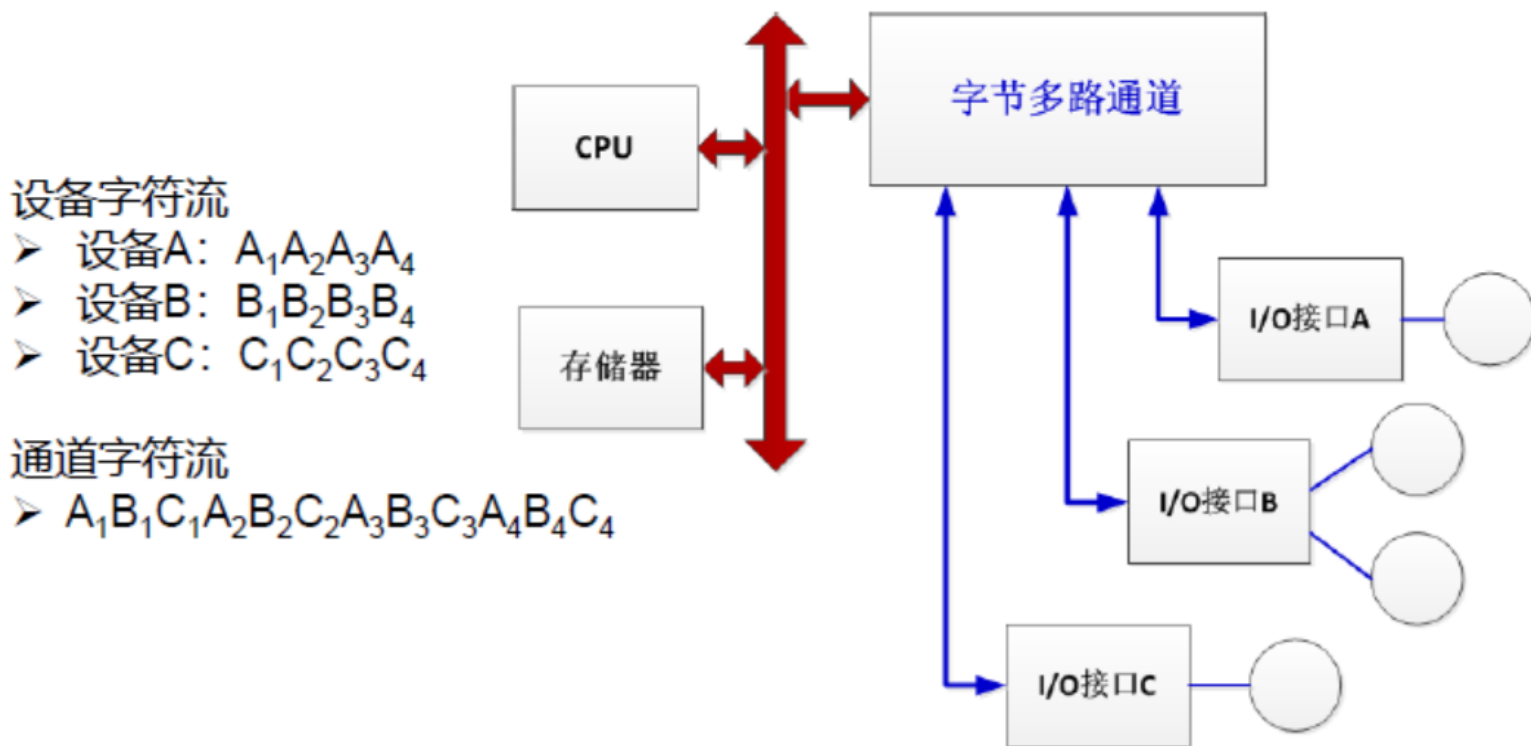
- ▶ 通道可以连接多台高速设备，但一次只能为其中一台设备服务；
- ▶ 与一台设备的成组数据传送结束后，才能选择另一台设备；
- ▶ 通道数据传输率 = 一台设备的数据传输率。
- ▶ 一旦选择了一个外设，即使该外设没有准备好，也只能等待。



6.2 通道分类

❖ 字节多路通道

- 通道连接多台慢速外设，通道可以同时为多台设备服务；
- 以字节为单位交叉传送各外设的数据；
- 通道的数据传输率 = 各外设的数据传输率之和。



6.2 通道分类

❖ 数组多路通道

- 通道可以连接多台高速外设，通道可以同时为多台设备服务；
- 以成组交叉的方式传送数据。
- 通道数据传输率 = 各设备数据传输率之和

