



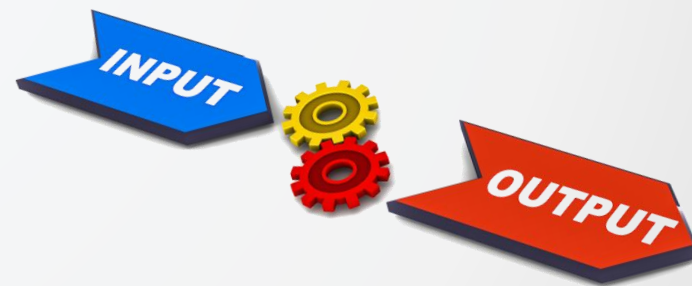
# 计算机组成原理





# 计算机组成原理

## 八、输入输出系统



## || 本章主要内容

- **输入输出系统概述**
- 外围设备定时方式与信息交换方式
- 程序中断方式
- DMA方式
- 通道方式

# 输入/输出系统的组成与功能

- 外部设备、接口部件、总线以及相应的管理软件统称为计算机的输入/输出系统，简称**I/O系统**
  - 完成计算机内部二进制信息与外部多种信息形式间的交流
  - 保证CPU能够正确选择I/O设备并实现对其控制，与数据传输
  - 利用数据缓冲、合适的数据传送方式，实现主机外设间速度匹配
  - I/O硬件
    - ◆ 外设、控制器、I/O接口、总线
  - I/O软件
    - ◆ OS无关库，设备无关库，驱动



# 现代计算机中的I/O接口软件

- 现代计算机中用户必须通过操作系统间接访问设备，屏蔽设备细节，使用更方便

- **与OS无关的I/O库 (用户态)**

如C语言中的标准I/O库stdio.h,

printf、scanf、getchar、putchar、fopen、fseek、fread、fwrite、fclose等

用户程序主要通过调用I/O库访问设备，方便程序在不同OS间移植

- **与设备无关的OS调用库 (内核态)**

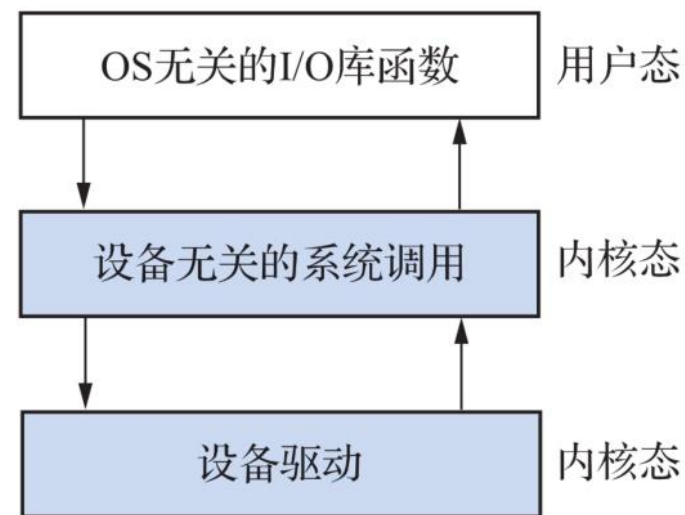
open、read、write、seek、ioctl、close

- **独立的设备驱动程序 (内核态)**

设备驱动程序是与设备相关的I/O软件部分

不同设备对应不同的驱动程序

遵循具体设备的I/O接口约定，包含设备接口细节



## || 输入输出设备的特性

- 异步性。外部设备种类繁多，速度差异大（CPU与设备，设备与设备）
- 实时性。需要及时响应，人机交互，数据传输
- 独立性。设备差异性，控制方式、资源、数据格式、接口类型

# 基本输入输出过程

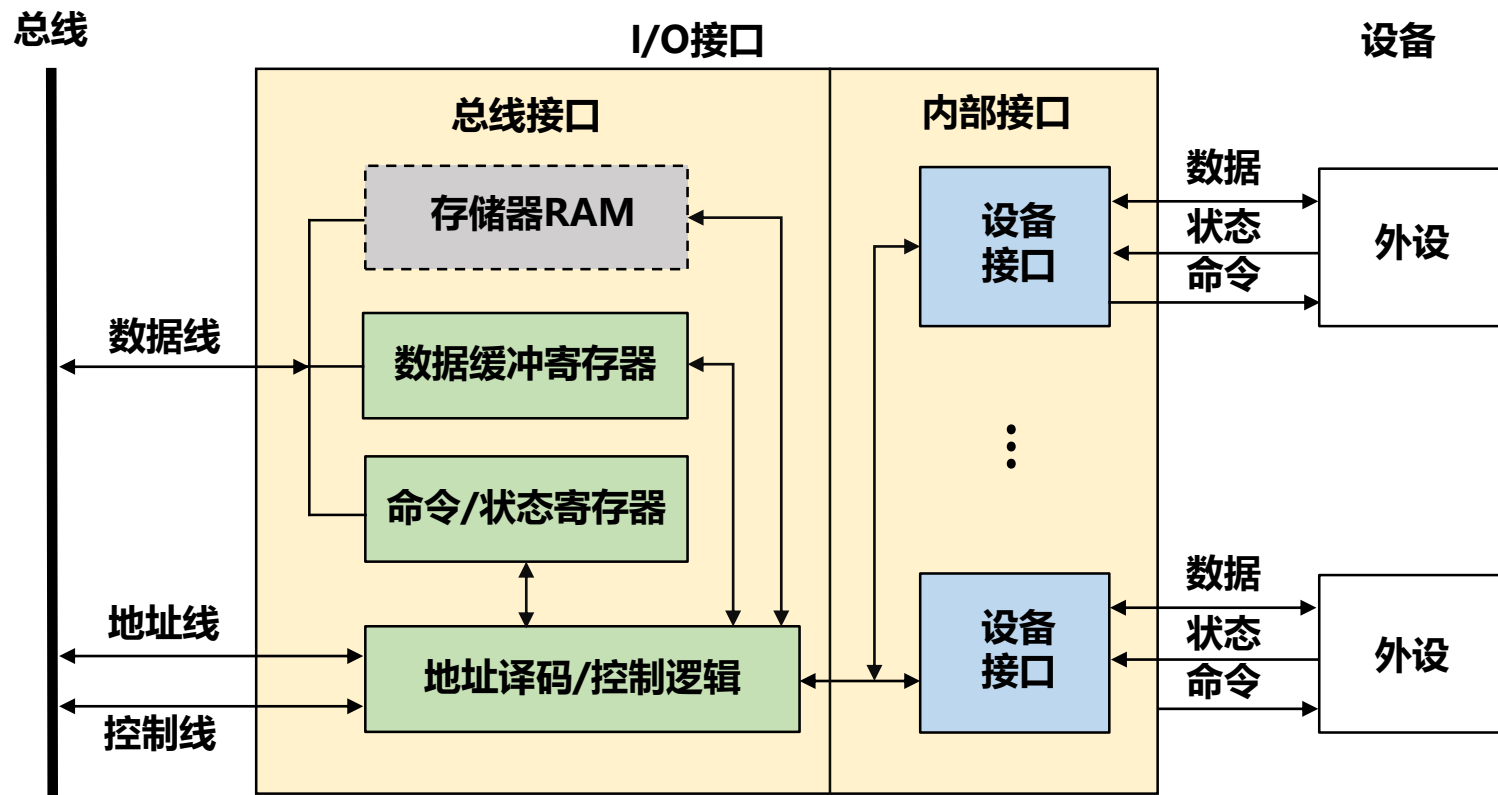
## ■ 输入过程

- CPU将一个地址放在地址总线上，选择设备
- 被选中的输入设备将数据驱动到数据总线
- CPU从数据总线读入数据

## ■ 输出过程

- CPU将一个地址放在地址总线上，选择设备
- CPU把数据驱动到数据总线上
- 被选中的输出设备锁存数据总线上的数据，并取走使用

# I/O接口结构与功能



## I/O接口的功能

- 设备寻址
- 数据交互
- 设备控制
- 状态检测
- 数据缓冲
- 格式转换

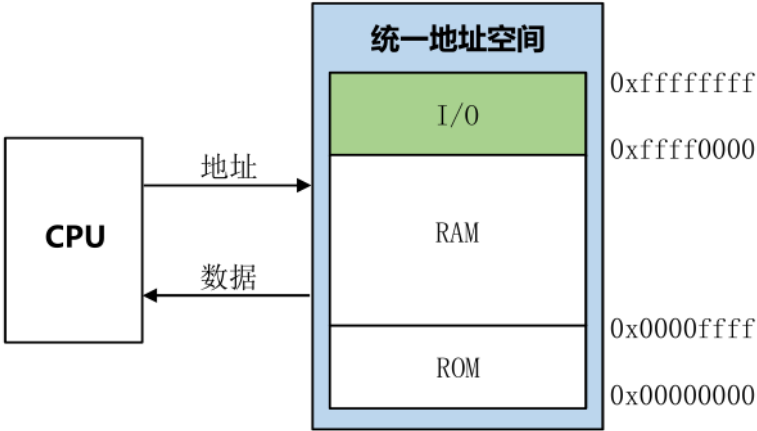
某些IO本身集成了DMA模块，因此地址线和控制线也有可能成为双向的



# I/O接口的编址

## ■ 统一编址

- 内存映射编址 (Memory-mapped)
- 外设地址与内存地址统一编址，同一个地址空间
- 不需要设置专用的I/O指令
- 采用访存指令访问外设，具体访问什么设备取决于地址



## ■ 独立编址 (端口映射)

- 端口映射编址 (Port-mapped)
- I/O地址空间与主存地址空间相互独立
- I/O地址又称为I/O端口
- 不同设备中的不同寄存器和存储器都有唯一的端口地址
- 使用I/O指令访问外设

I/O 地址范围 (十六进制)	IRQ 中断号	设备
0000 ~ 000F	00000004	DMA 控制器
0020 ~ 0021	无	中断控制器
0040 ~ 0043	00000000	计时器
0060 ~ 0064		
0070 ~ 0071		
02F8 ~ 02FF		
F060 ~ F07F		
03D0 ~ 03DF		



## 不同编址方式程序实例

- 统一编址：访存指令访问外设，具体访问什么设备取决于内存地址

<b>lw</b>	\$t0, 0x00000004	# 从 ROM 读取一个字
<b>sw</b>	\$t0, 0x00000004	# 写 ROM (会产生总线错误)
<b>lbu</b>	\$t0, 0x00010001	# 从内存读取一个字节
<b>sb</b>	\$t0, 0xff000002	# 写一个字节到内存
<b>lbu</b>	\$t0, 0xffff0000	# 从 I/O 设备读一个字节
<b>sb</b>	\$t0, 0xffff0004	# 写一个字节到 I/O 设备

- 独立编址：I/O指令访问外设，具体访问什么设备取决于端口地址

<b>OUT</b> DX, AL	I/O 写：将 AL 寄存器中的字节写入 DX 寄存器对应的 I/O 设备
<b>IN</b> AL, DX	I/O 读：从 DX 寄存器对应的 I/O 端口地址读取一个字节
<b>MOV</b> [BX], AL	内存写：将 AL 寄存器中的值送到 BX 寄存器对应的内存

# I/O接口分类

- 并行 – 串行
- 可编程 – 不可编程
- 通用 – 专用
- 同步- 异步
- 访问方式，编址方式
  - 直接传送方式接口、程序控制方式接口、程序中断控制方式接口、DMA接口、通道处理机接口

# 数据传输控制方式

- 直接输入输出方式---适用于简单IO

  - 例如读取开关状态、读取当前温度值、控制加热、制冷、LED灯亮灭，等

- 程序查询方式

- 程序中断方式

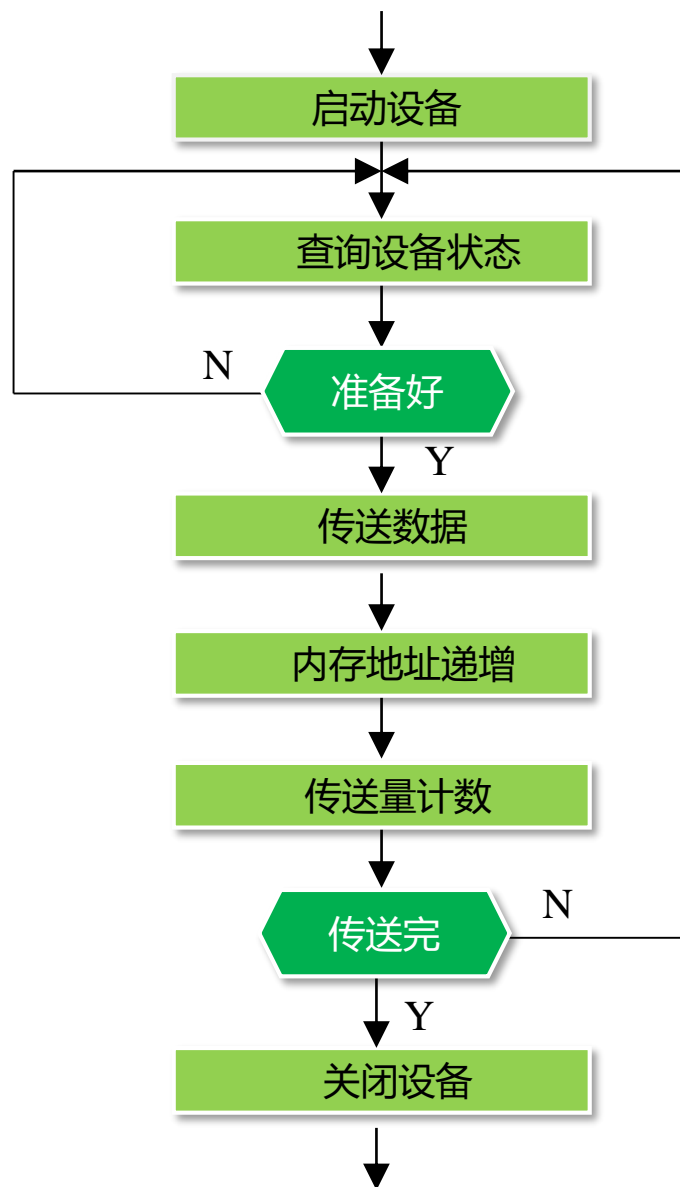
- 直接内存访问方式

- 通道方式

- 外围处理机方式

} 主要由硬件实现

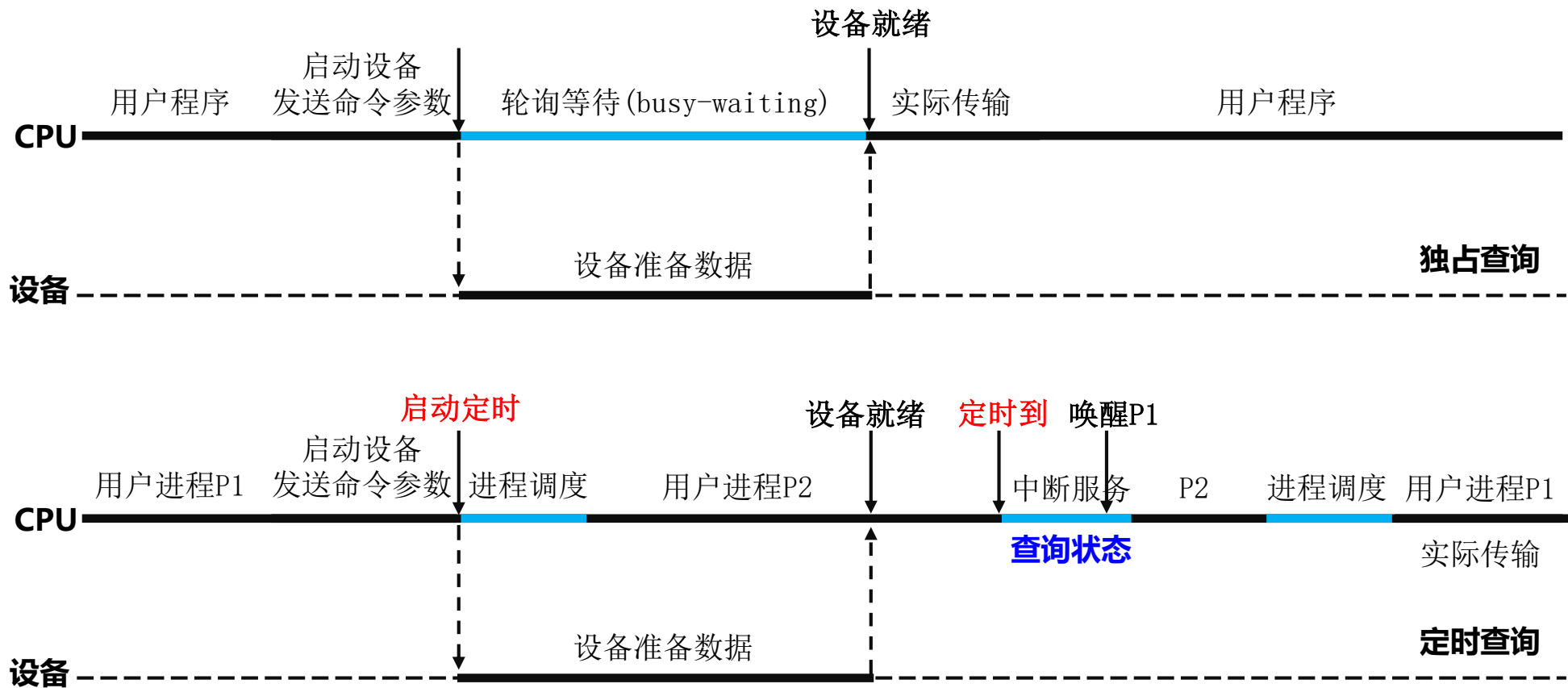
## 程序查询方式



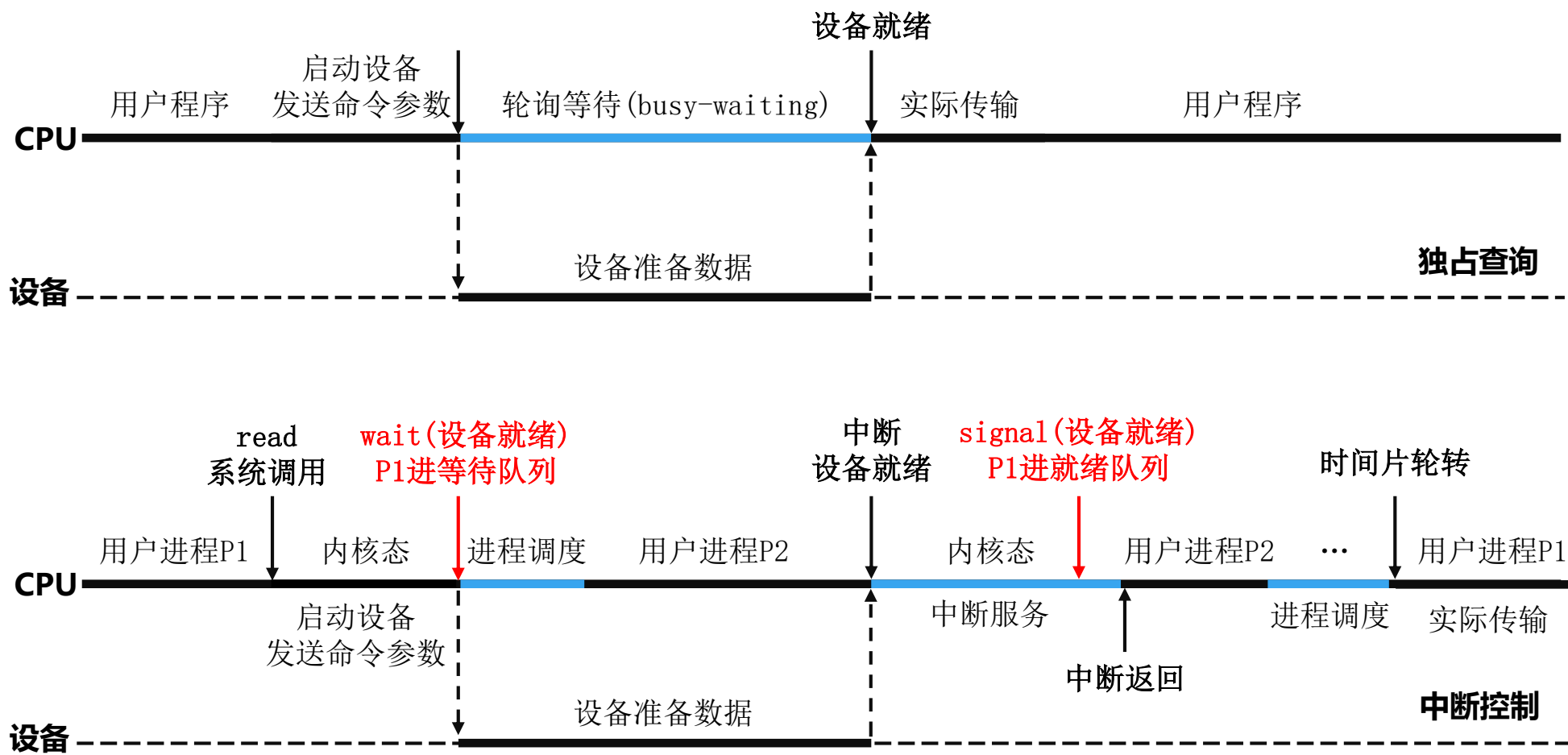
- 信息交换完全由CPU执行程序实现。
  1. 启动设备;
  2. 反复查询设备直至设备准备好;
  3. 传输单个数据
  4. 重复2-3步直至数据传输完毕
- CPU外设串行工作，反复查询设备状态占用较多CPU时间，系统效率低
  - CPU占用率取决于查询频率
- 用于早期的计算机



# 程序查询方式运行轨迹



# 中断控制方式



## 中断优势

- 提高了CPU的使用效率
  - 主动告知机制避免了反复查询设备状态
  - 仍需CPU占用（中断服务子程序运行时间+中断开销）
- 适合随机出现的服务
- 需要专门的硬件

# 直接内存访问DMA方式

- 中断方式用中断服务子程序完成数据交换
  - 效率较低
    - ◆ 一次中断仅传输少量数据，CPU开销大
  - 不适合于成组数据交换
- DMA用于成组交换数据的场合
- 硬件执行I/O交换
  - 准备阶段和结束阶段需要占用CPU
  - 传输阶段DMAC从CPU接管总线，直接在内存及外设之间进行，节约了中断开销
- 需要更多硬件

## 通道方式

- DMA方式的进一步发展，数据的传送方向、内存起始地址及传送的数据块长度等都由独立于CPU的通道来进行控制，可进一步减少CPU的干预。
  - 通道是一个具有特殊功能的处理器IOP
  - 分担CPU的I/O 处理的功能
  - 可实现外设的统一管理和DMA操作
  - 大大提高CPU效率，更多的硬件
- 通道执行通道程序来完成CPU指定的I/O任务，通道程序是由一系列通道指令组成的。
- 当通道执行完通道程序后，就发出中断请求表示I/O结束，CPU响应中断请求，执行相应的中断处理程序实现与通道之间的数据传输。



# 数据传输控制方式

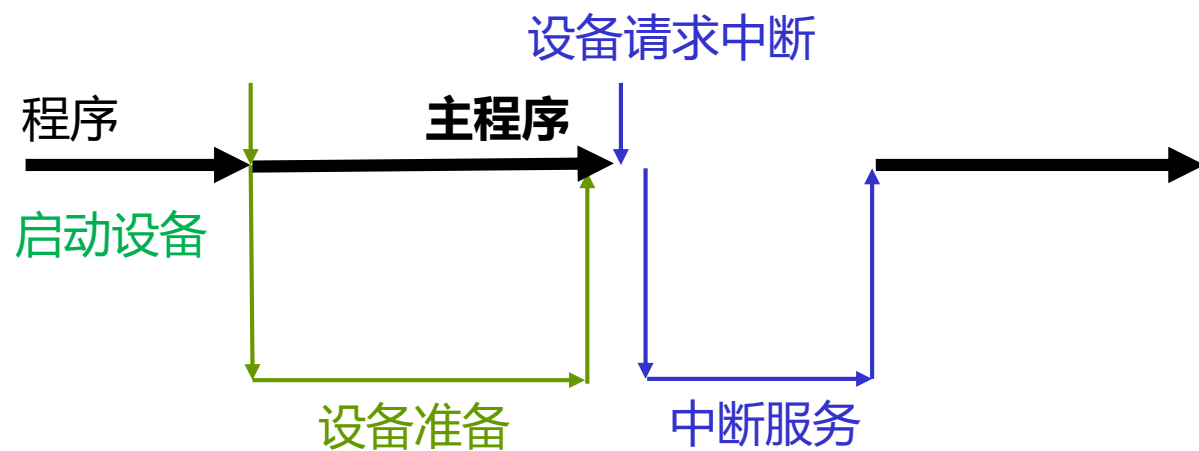
- 程序查询方式
- **程序中断方式**
- 直接内存访问方式
- 通道方式
- 外围处理机方式

# || 程序中断方式

- 中断基本概念
- 程序中断基本接口
- 中断仲裁方式
- 中断控制器

## 中断基本概念

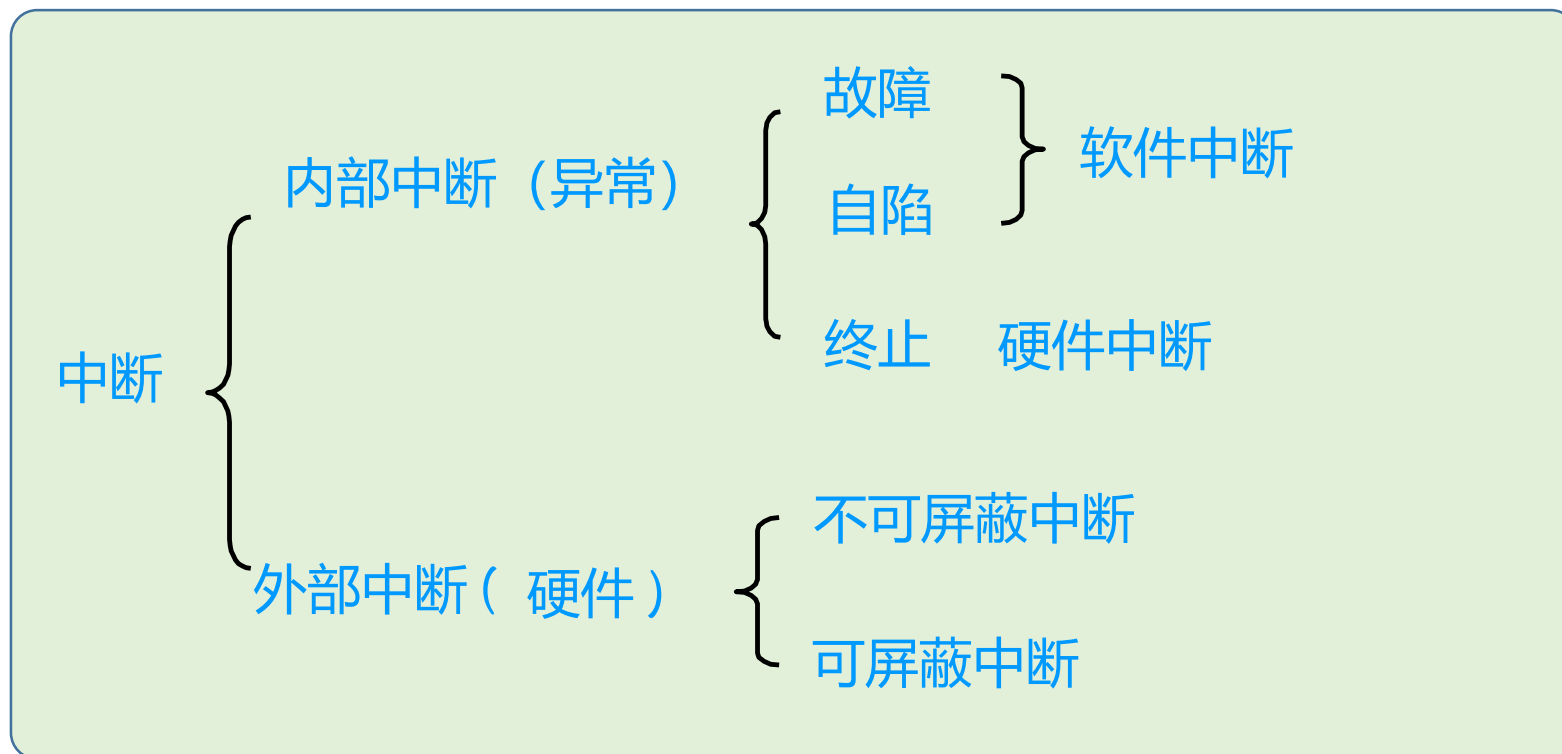
- CPU暂时中止现程序的执行，转去执行为某个**随机事件**服务的中断处理子程序，处理完后自动恢复原程序的执行
- 实现主机和外设准备阶段的并行工作
  - 避免重复查询外设状态、提升工作效率



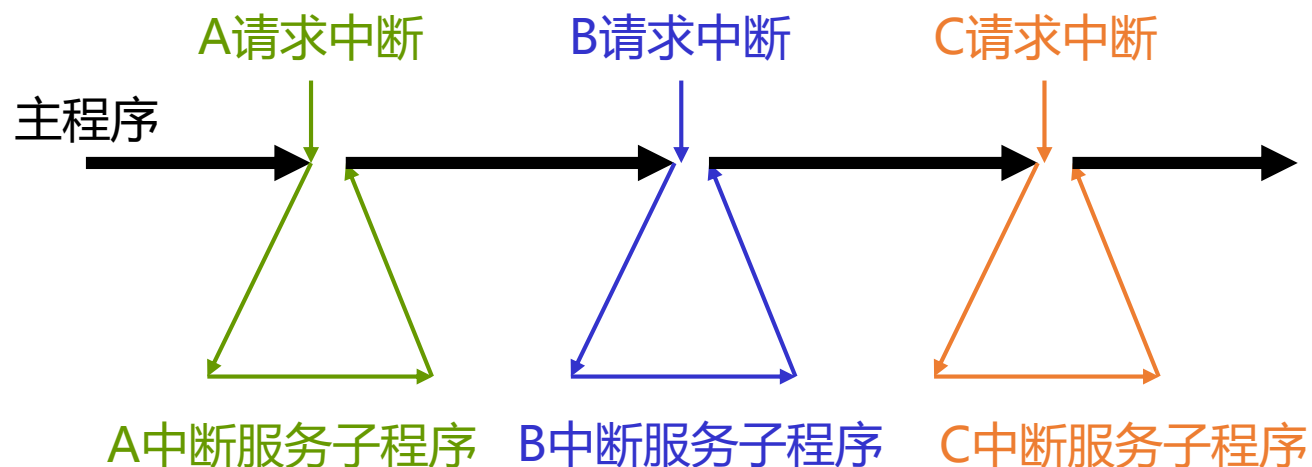
# 中断的分类与作用

- 中断技术赋予计算机应变能力，将有序的运行和无序的事件统一起来，大大增强了系统的处理能力

- 主机外设并行工作
- 程序调试
- 故障处理
- 实时处理
- 人机交互



# 程序中中断处理示意图



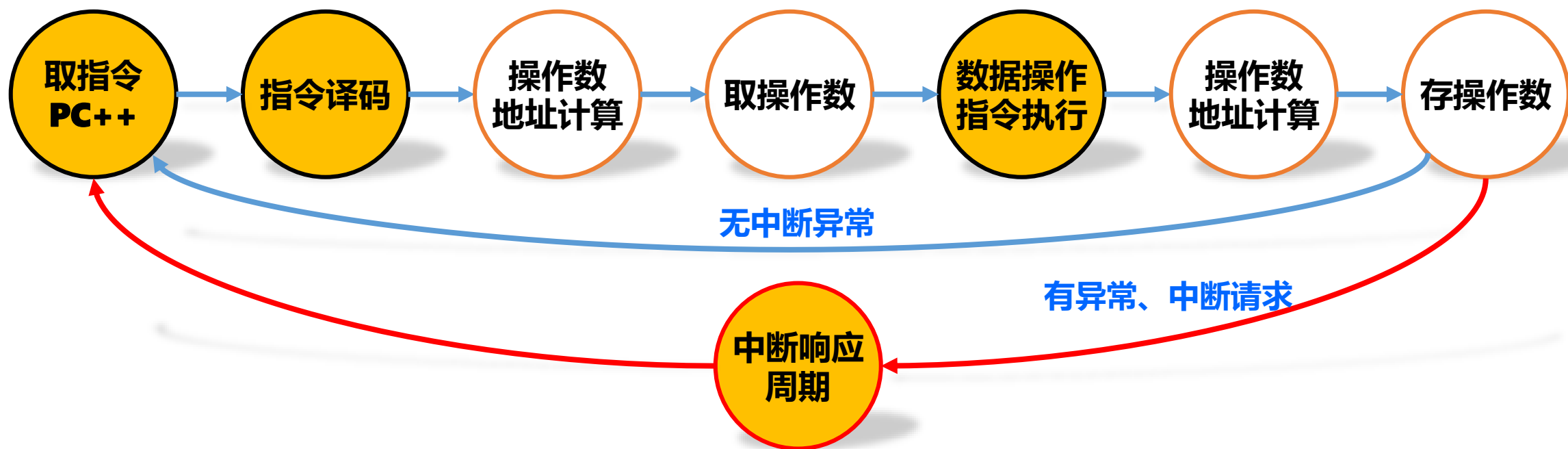
- 子程序与中断服务子程序的区别？
  - 子程序在特定位置显式调用，后者随机调用，现场不同？
- 如果A，B，C同时产生中断？
  - 中断优先级问题，中断仲裁
- 如果正在运行A中断服务子程序，又收到B中断？
  - 中断嵌套

什么是主程序？



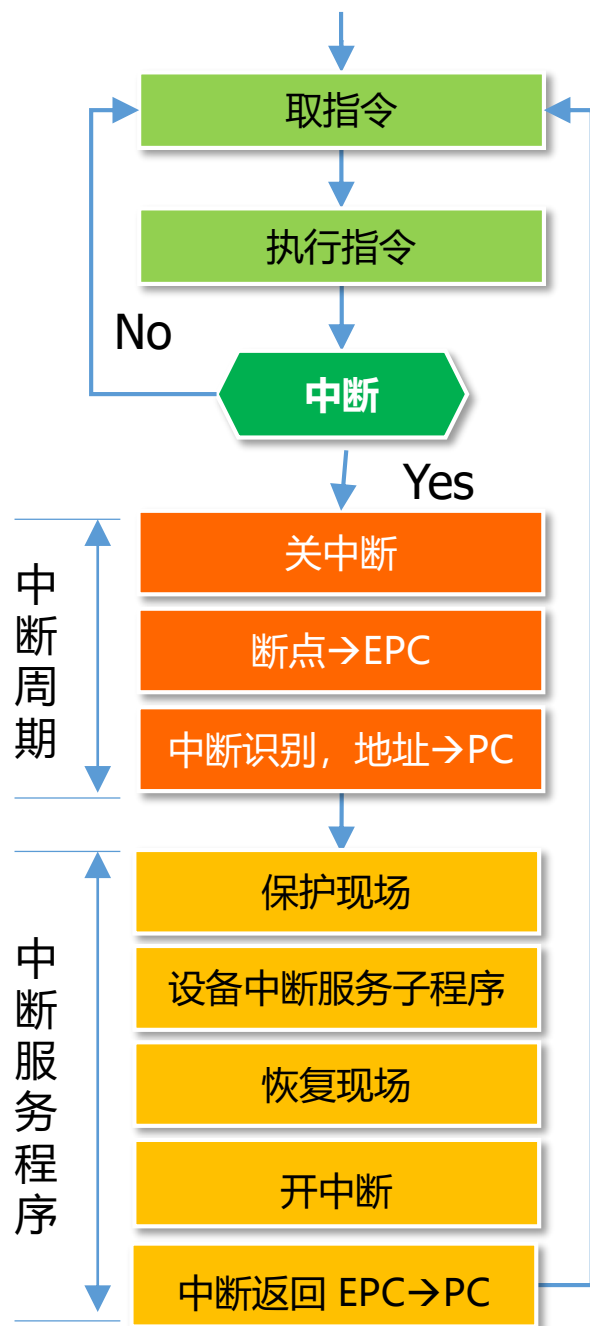
# 指令执行一般流程

- 取指令、执行指令反复循环
- 指令功能、寻址方式不同，数据通路不同，执行时间不同，**如何安排时序？**
  - 访存指令、寄存器运算指令、加法指令、除法指令

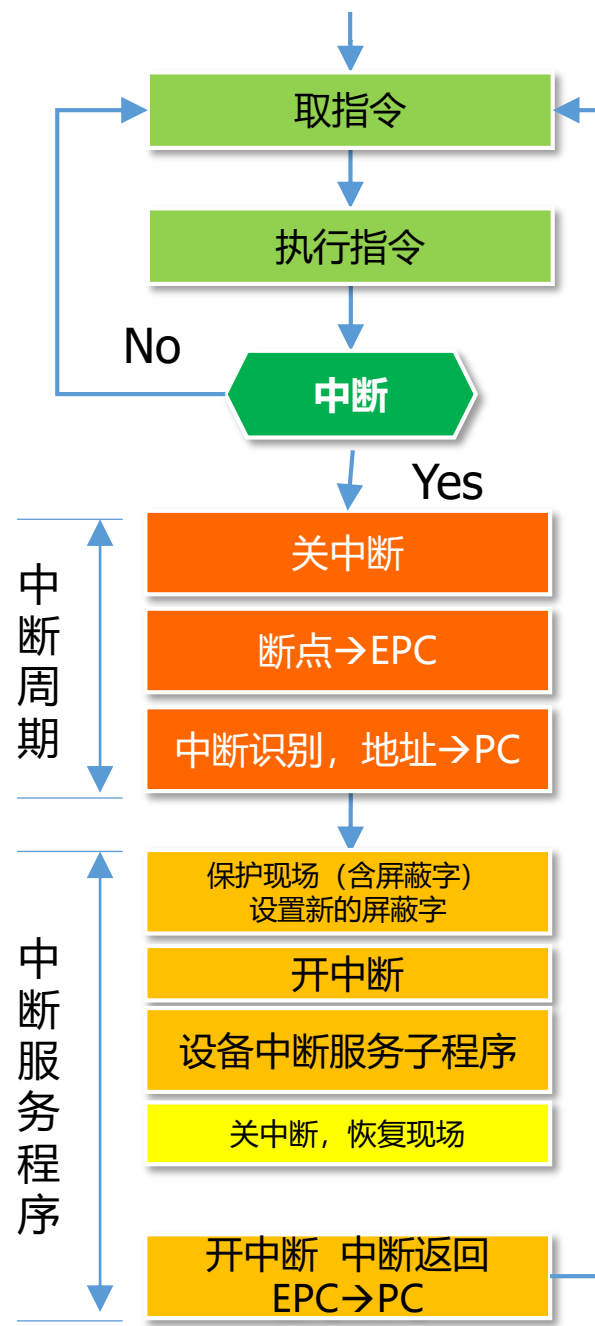


## 单重中断

### 中断隐指令



## 多重中断



# 中断优先级

- 多设备同时产生中断请求时，如何处理？
  - 优先级高的先响应，优先级低的后响应
  - CPU优先级随不同中断服务程序而改变
    - ◆ 执行某设备中断服务子程序
    - ◆ CPU优先级就与该设备的优先级一样
  - 优先级高的中断请求可否中断优先级低的程序？

# 单级中断与多级中断

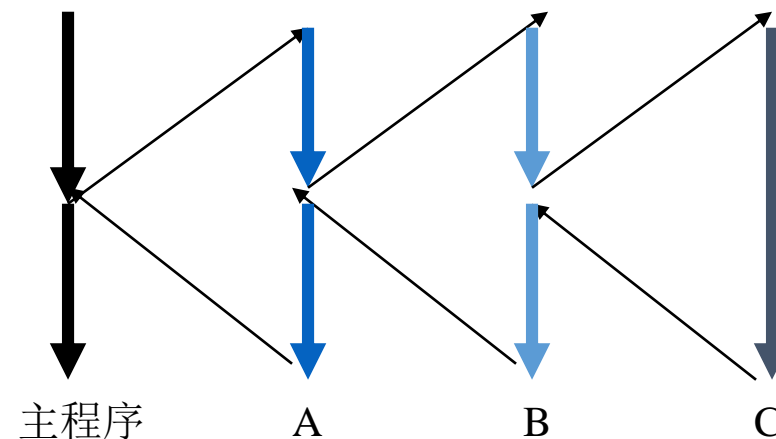
- 高优先级中断请求能否中断运行中的程序呢？
- 系统硬件、软件开销的权衡

## □ 单级中断

- ◆ 所有中断源均属同一级，离CPU近的优先级高
- ◆ CPU处理某个中断时，不响应其他中断

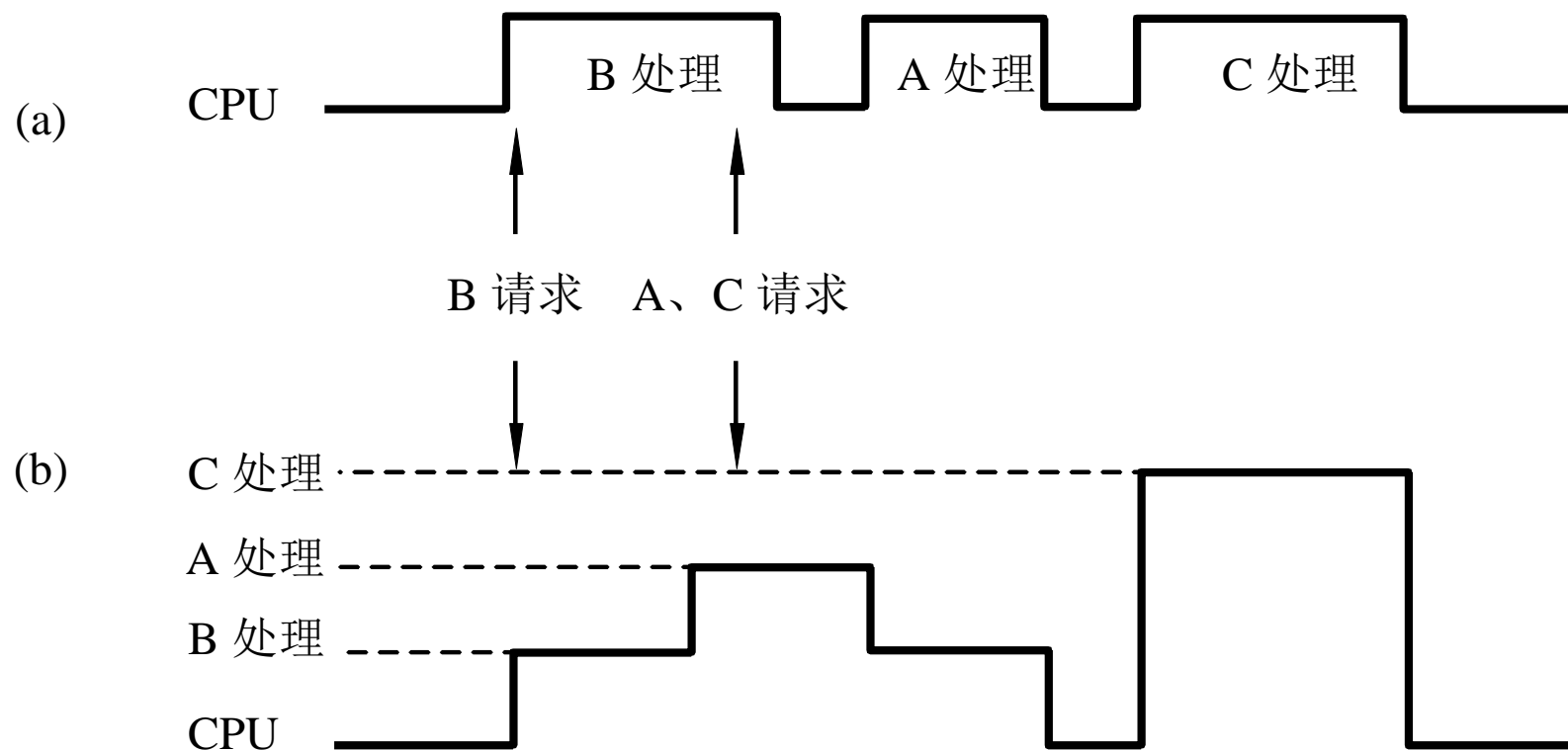
## □ 多重中断

- ◆ 优先级高的中断可以打断优先级低的中断服务程序
- ◆ 中断嵌套



# 同时中断请求的处理方法

■  $A > B > C$





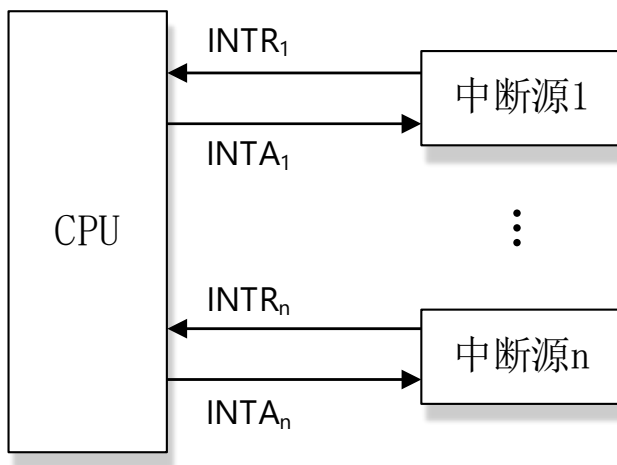
## 划分优先级的一般规律

- 硬件故障中断属于最高级，其次是程序错误中断
- 非屏蔽中断优于可屏蔽中断
- DMA请求优先于I/O设备传送的中断请求
- 高速设备优于低速设备
- 输入设备的中断优于输出设备
- 实时设备优先于普通设备

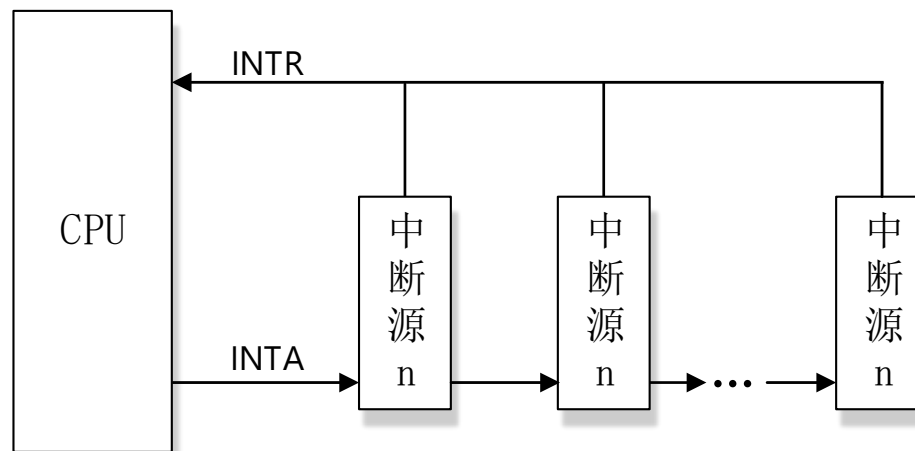
## || 优先级实现---中断仲裁

- 同一时刻可能有多个设备同时发出中断请求，响应谁？
  - 独立请求
  - 链式查询
  - 中断控制器方式
  - 分组链式结构

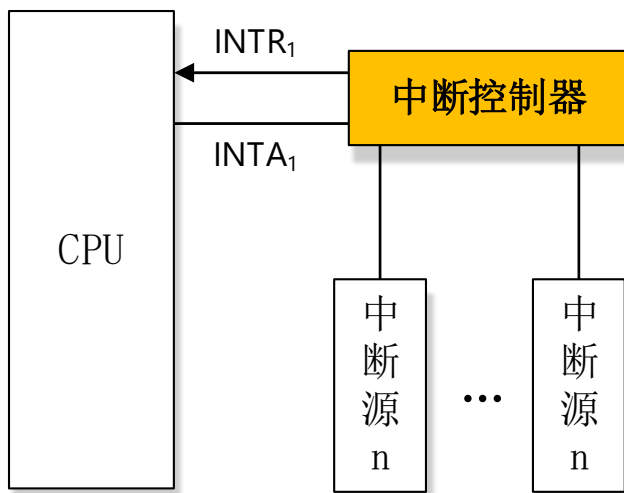
# 中断请求信号的传输方式



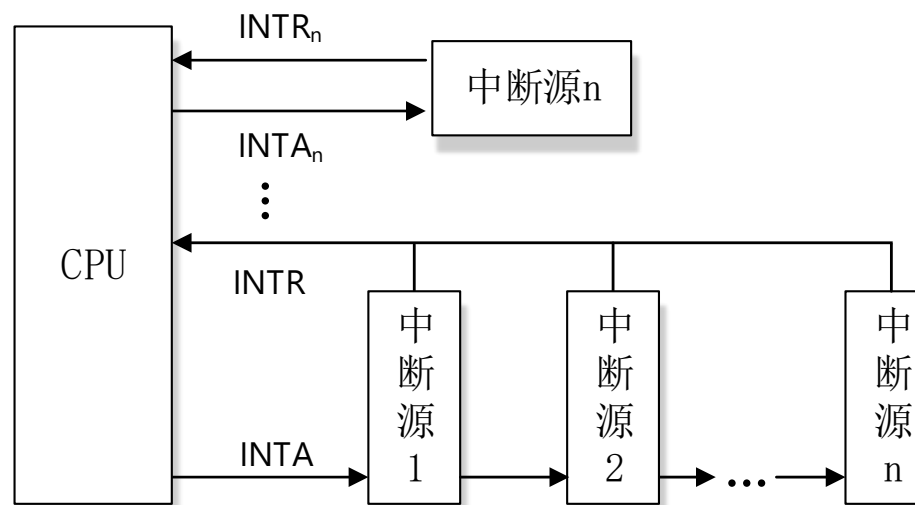
(a) 独立请求方式



(b) 链式请求方式



(c) 中断控制器方式



(d) 分组链式请求

# 中断屏蔽

## ■ 响应优先级

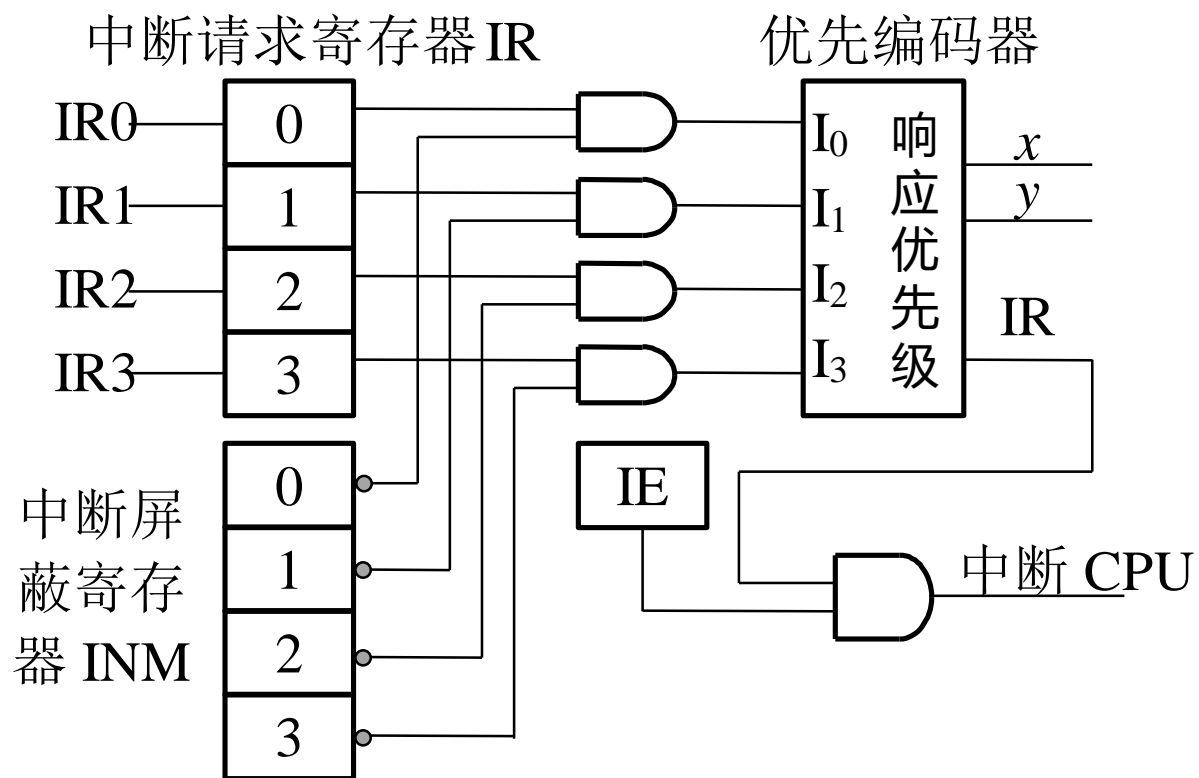
- CPU对各设备中断请求进行响应，并准备好处理的先后次序，这种次序往往在硬件线路上已固定，不便于变动。

## ■ 处理优先级

- CPU实际对各中断请求处理的先后次序。如果不使用屏蔽技术，响应的优先次序就是处理的优先次序。

## ■ 中断屏蔽技术可**动态改变**各设备的处理优先级

## 中断屏蔽方式



- 当CPU执行某个设备的中断服务程序时，如何设置中断屏蔽字？

# 中断屏蔽位

## ■ 中断请求寄存器IR

- 对应位为1表示相应外设发出了中断请求
- 中断字，中断码

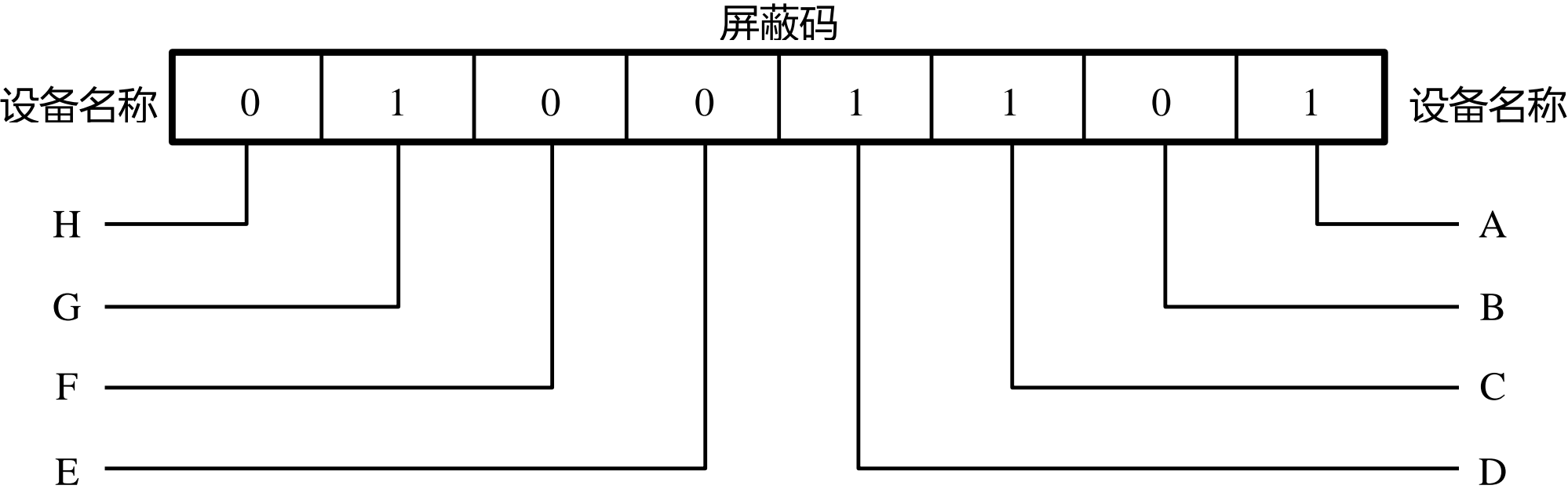
## ■ 中断屏蔽寄存器INM

- 对应位为1设置屏蔽，否则取消屏蔽
- 每个设备都有自己独立的中断屏蔽字
- CPU执行某设备的中断服务子程序时将其中断屏蔽字载入INM
- 不可屏蔽中断不受中断屏蔽寄存器的控制

## ■ 中断允许触发器IE

## 屏蔽码

- 控制各设备接口的屏蔽触发器，可改变处理次序。
- 运行某个设备的中断服务程序时载入对应的屏蔽码



## 例子

- 假定硬件原来的响应顺序为 $0 \rightarrow 1 \rightarrow 2$ ，试设置中断屏蔽字，将中断优先级改为 $1 \rightarrow 2 \rightarrow 0$ 。

设备/屏蔽字	L0	L1	L2
L0	1	0	0
L1	1	1	1
L2	1	0	1



# 中断识别（寻找入口地址）

## ■ 向量中断

□ 将服务程序入口(中断向量)组织在中断向量表中；响应时由硬件直接产生相应向量地址，按地址查表，取得服务程序入口，转入相应服务程序。

◆ 硬件查询法

◆ 独立请求法

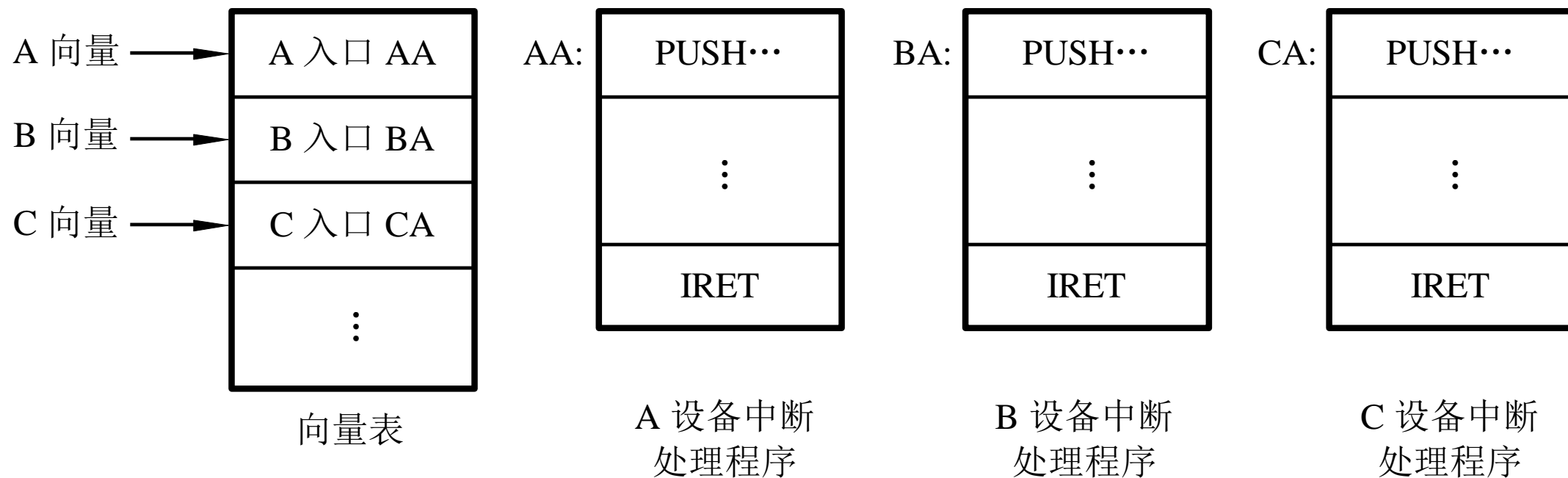
## ■ 非向量中断

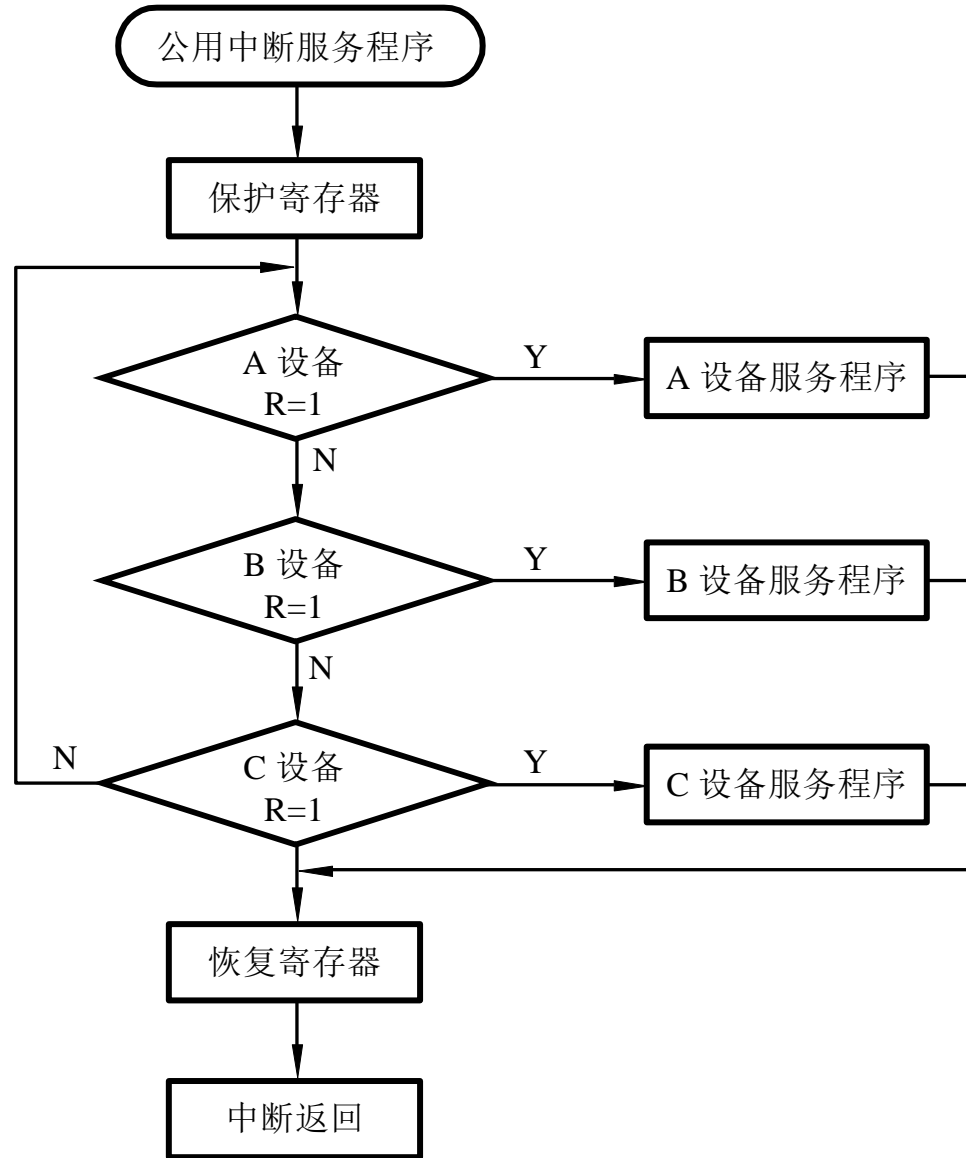
□ 将服务程序入口组织在查询程序中；

□ 响应时执行查询程序查询中断源，转入相应服务程序。

◆ 程序识别      (软件方法)

# 中断向量法





# 中断处理中的问题

## ■ 中断响应条件

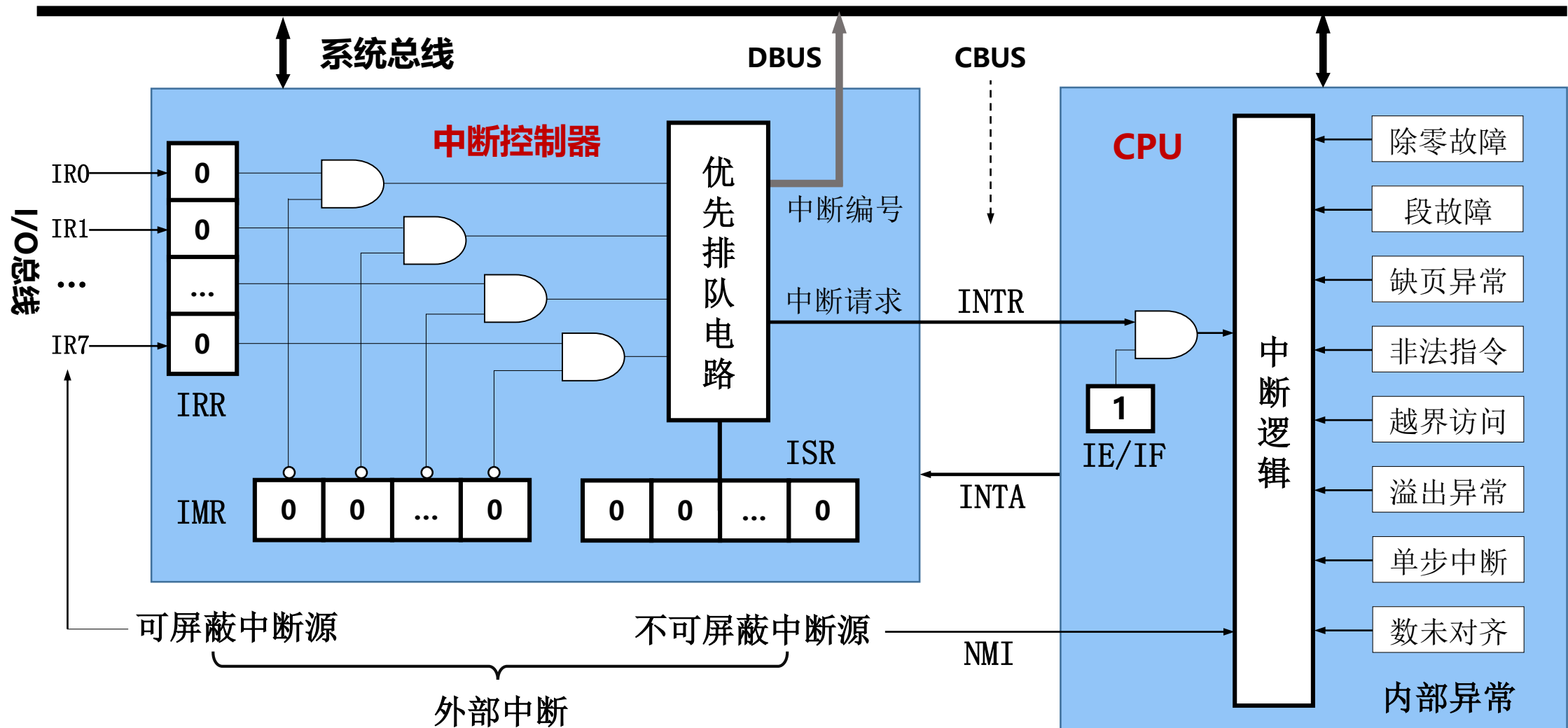
- 中断允许触发器处于允许状态
- 对应的中断未被屏蔽
- 无更高优先级的DMA请求
- 中断嵌套必须优先级更高
- 指令已经执行完最后一个机器周期
  - ◆ 保证指令执行的完整性;
  - ◆ 缺页中断的中断时机?

## ■ 保存现场，恢复现场

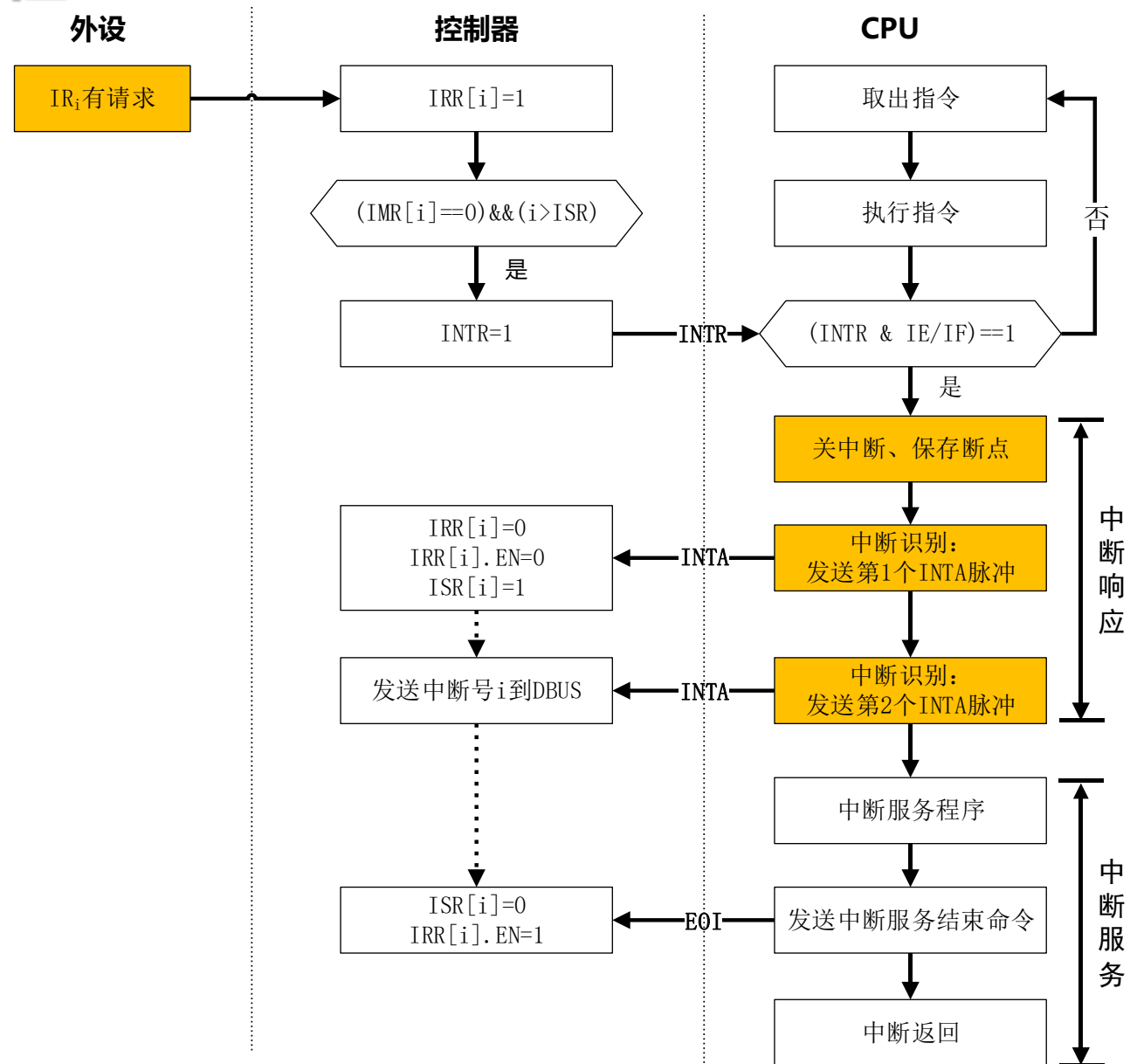
- 中断程序用到的通用寄存器，EPC，屏蔽字
- 缺页中断的断点和普通中断断点不一致

## ■ 中断过程由软硬件结合完成

# 中断硬件接口



# 中断控制器工作流程



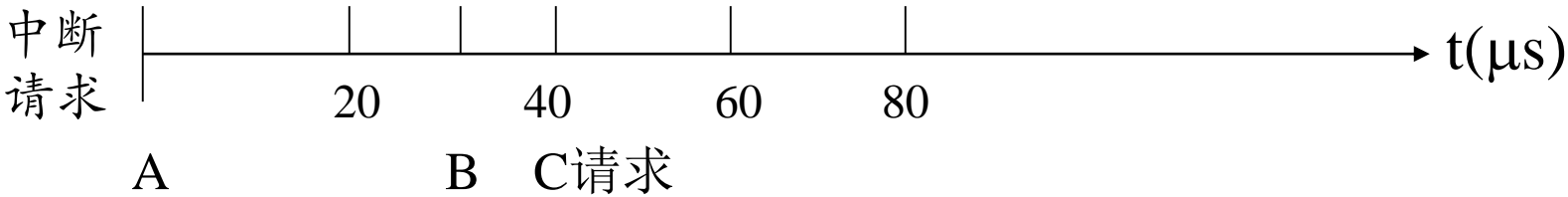
## || 工作过程

- 主机启动设备
- 设备准备传送
- 发中断请求信号
- 主机响应中断
- 数据传送

13.A、B、C是与主机连接的3台设备，在硬件排队线路中，它们的响应优先级是 $A > B > C > \text{CPU}$ ，为改变中断处理的次序，将它们的中断屏蔽字设为：

设备	屏蔽码		
	A	B	C
A	1	1	1
B	0	1	0
C	0	1	1
CPU	0	0	0

请按下图所示时间轴给出的设备中断请求时刻，画出CPU执行程序轨迹。A、B、C中断服务程序的时间宽度均为 $20\mu\text{s}$ 。



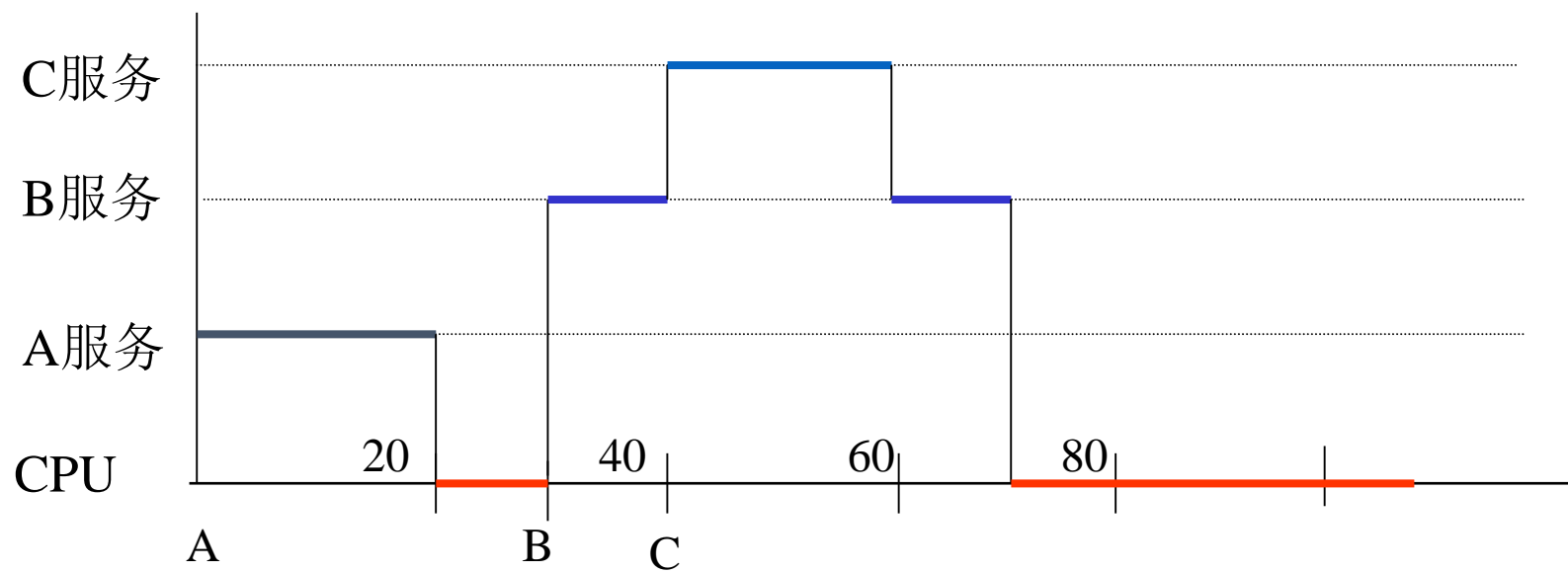




解：从中断屏蔽字看出，其处理优先级为：

$$A > C > B$$

故CPU运行轨迹如下：



# || DMA方式

- DMA基本概念
- DMA传输方式
- 基本DMA控制器

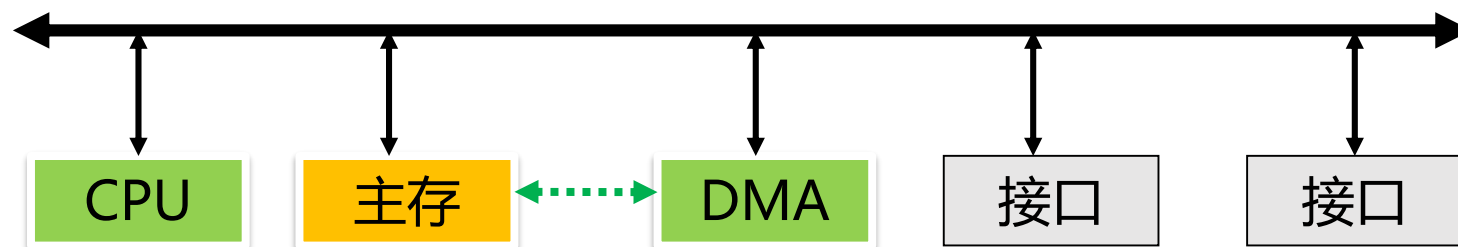
# DMA基本概念

## ■ 中断方式

- 传送一个数据执行一次中断服务子程序（几十条指令）
- 效率低下，不适合于高速传输的系统。

## ■ DMA方式

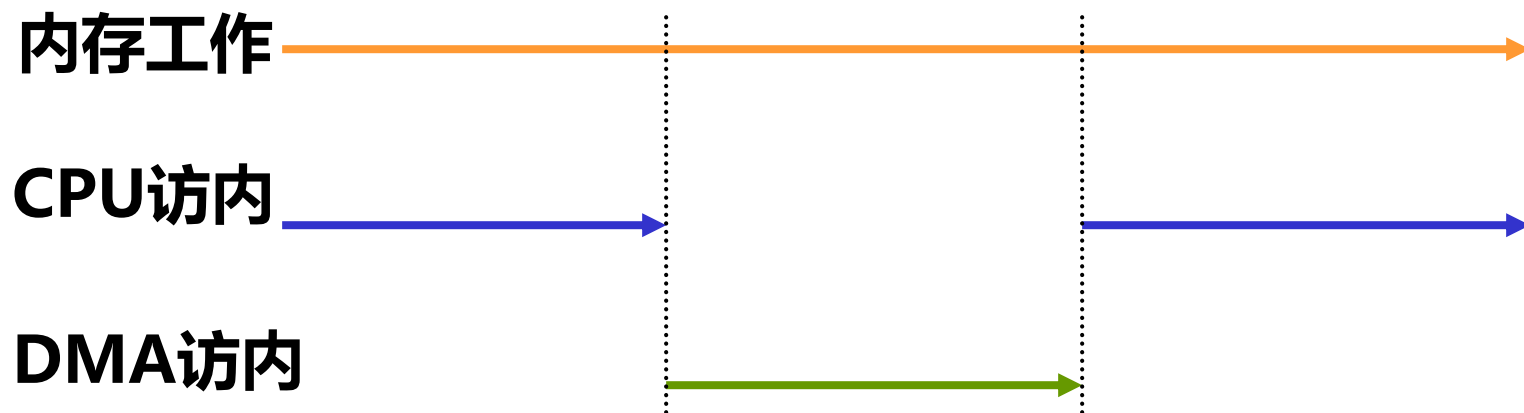
- 外设与主存间建立一个由硬件管理的数据通路（不一定有单独的通路）
- CPU不介入外设与主存的数据传送操作
- 减少CPU开销，提升效率



## 内存争用

- DMA方式进行数据传送时
  - DMA控制器直接访问内存
  - CPU执行主程序（需要访内）
  - 主存使用权的冲突（资源冲突）
- 如何处理这种冲突？
  - 停止CPU使用主存
  - DMA与CPU交替使用主存
  - 周期挪用法

## || 停止CPU访内

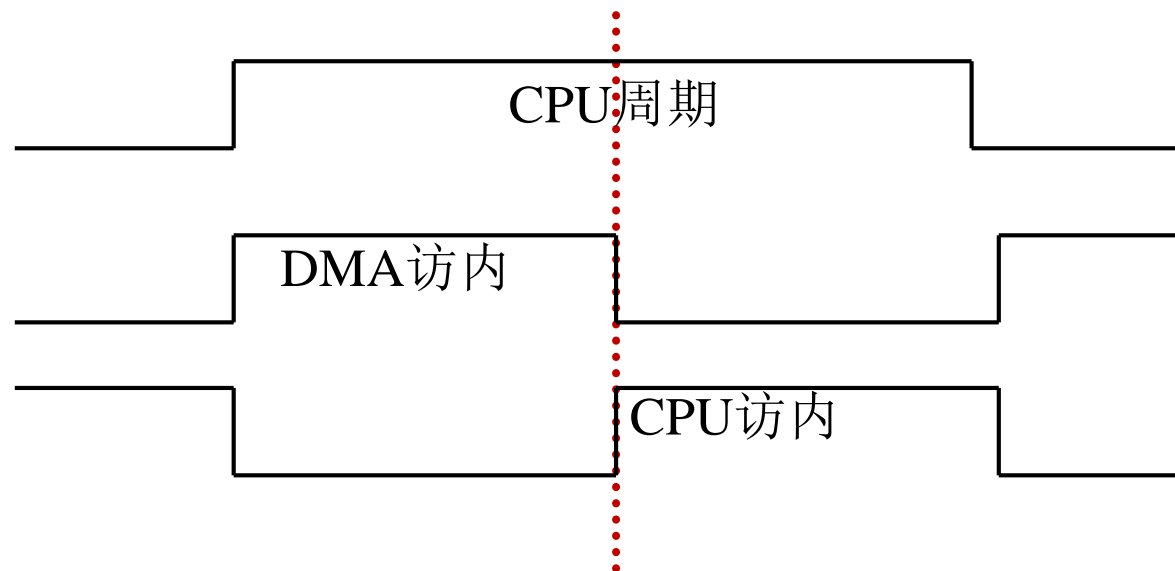


1. DMA批量数据传输周期过长，CPU长期无法访内
2. 外设传送两个数据的时间间隔大于存储周期，内存未充分利用



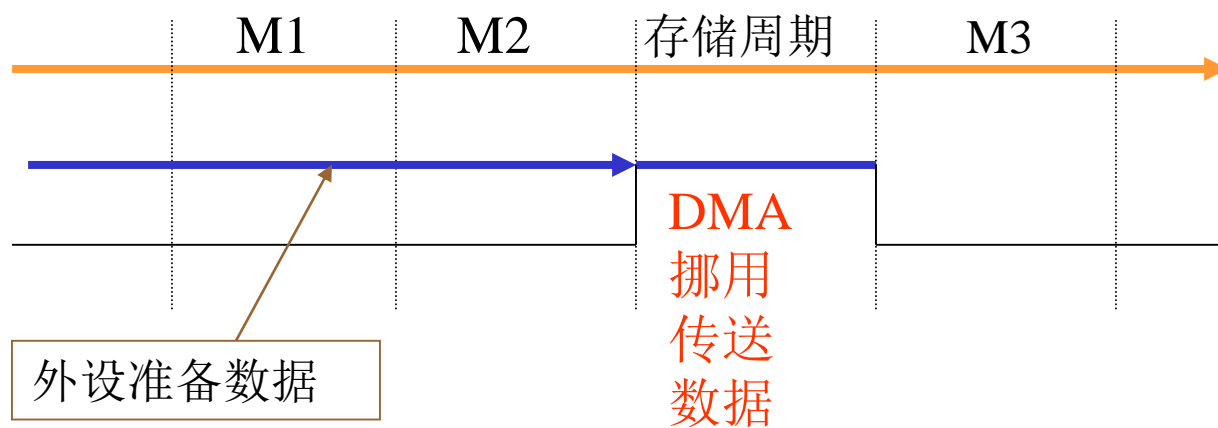
# DMA与CPU交替使用主存

- 每个CPU工作周期分成两段
  - 一段用于 DMA访问主存
  - 一段用于CPU访问主存
- 无主存使用权移交过程

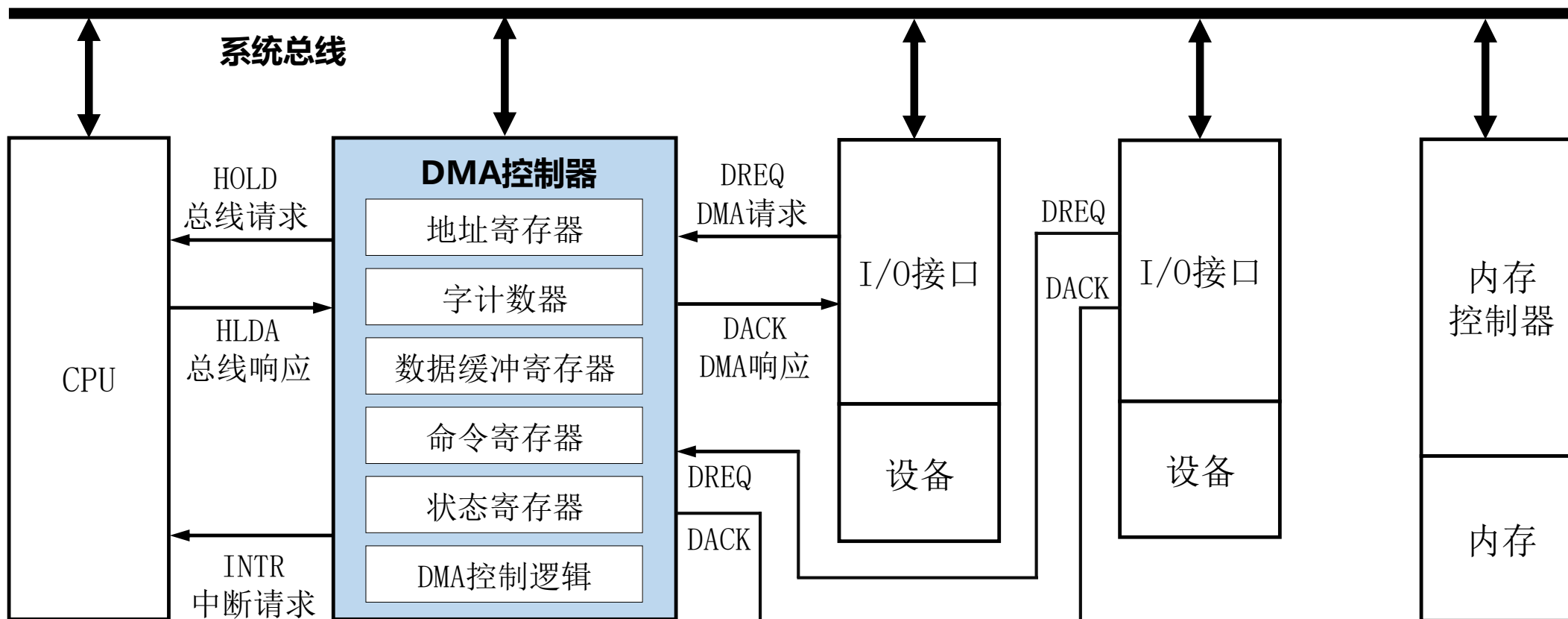


## 周期挪用法

- DMA要求访问主存时，CPU暂停一个或多个存储周期。一个数据传送结束后，CPU继续运行。
- CPU现场没有变动，仅延缓了指令的执行
  - 周期挪用，或称周期窃取。
- 如发生访存冲突，则DMA优先访问。



# DMA控制器





# DMA传输流程

## ■ 准备阶段

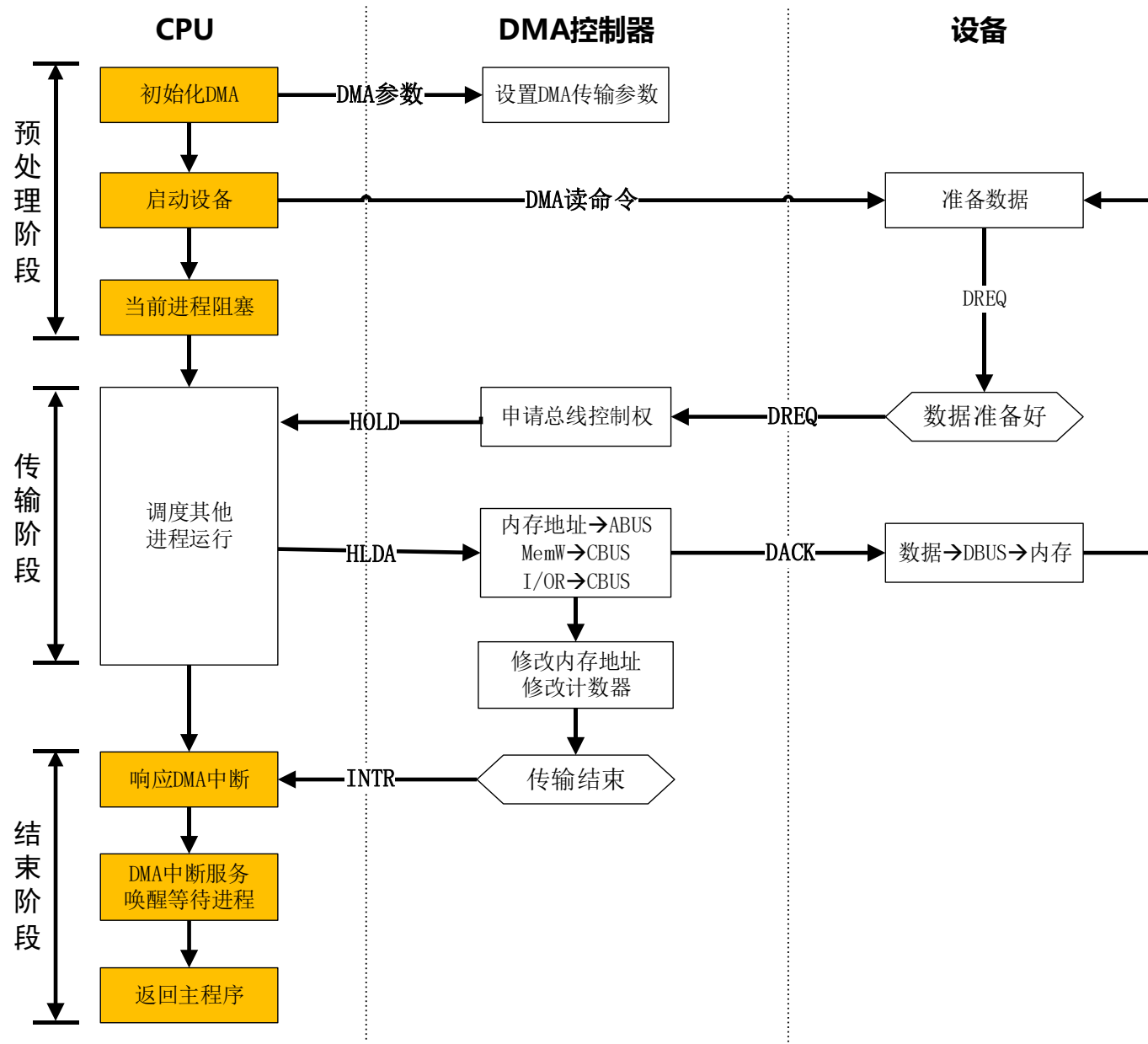
□ CPU干预

## ■ 传送阶段

□ CPU不参与

## ■ 结束阶段

□ CPU干预



## || DMA主要操作过程（准备阶段）

- 主机通过CPU指令向DMA接口发送必要的传送参数，并启动DMA工作。
  1. 数据传送的方向
  2. 数据块在主存的首地址
  3. 数据在外设存储介质上的地址
  4. 数据的传送量

## || DMA主要操作过程（传送阶段）

- **宏观** DMA是连续传送一批数据    **微观** 每传送一个数据，发一次DMA请求
- **传输过程（周期挪用）**
  1. 设备准备数据：当设备接收到 CPU 的 DMA 命令后就可以开始准备数据
  2. 设备发送 DMA 请求：数据准备好后就通过 DREQ 控制线向 DMAC 发出 DMA 请求。
  3. DMAC 申请总线：DMAC 收到 DMA 请求后立即将 HOLD 信号置“1”，向 CPU 申请总线控制权。
  4. 总线授权：CPU 在机器周期结束后响应总线使用申请，让出总线控制权，并发出总线授权信号 HLDA 通知 DMAC。

## || DMA主要操作过程（传送阶段）

5. DMA 数据传输：收到 HLDA 信号将内存地址放置在地址总线上；设置控制总线读写命令控制信号，并向设备DMA 应答信号 DACK。设备收到 DACK 信号后会和内存完成一个机器字的数据交换
6. 传输控制：设备传输完一次数据后会继续重复第 1 步到第 5 步的工作，DMAC 在每次传输时还需要负责维护内存地址和传输计数器，并撤除 HOLD信号释放总线
7. 数据传输结束时，DMAC 会通过 INTR 信号线发送一个 EOP（End OfProcess）的 DMA 中断请求信号，告知 CPU 传输完成

## || DMA主要操作过程（结束阶段）

- DMA在两种情况下都进入结束阶段。
  - 正常结束，一批数据传送完毕
  - 非正常结束，DMA故障
- 结束阶段DMA向主机发出中断请求
- CPU执行中断服务程序
  - 查询DMA接口状态，根据状态进行不同处理

## || DMA与程序中断的区别

- 中断通过程序传送数据，DMA靠硬件来实现。
- 中断时机为两指令之间，DMA响应时机为两存储周期之间。
- 中断不仅具有数据传送能力，还能处理异常事件。DMA只能进行数据传送。
- DMA仅挪用了一个存储周期，不改变CPU现场。
- DMA请求的优先权比中断请求高。CPU优先响应DMA请求，是为了避免DMA所连接的高速外设丢失数据。
- DMA利用了中断技术

## 例题

- 某计算机CPU主频500MHz，CPI为5。假定某外设的数据传输率为0.5MB/s,采用中断方式与主机进行数据传送，以32位为传输单位，对应的中断服务程序包含18条指令，中断服务的其他开销相当于2条指令的执行时间。

(1) 在中断方式下，CPU用于外设I/O的时间占整个CPU时间的百分比是多少？

传输32bit，需一次中断，

所需CPU开销 $T_{I/O} = (18+2) \times CPI \times T = 20 \times 5 / 500\text{MHz}$

传输32bit，需要的总时间 $T_{\text{total}} = 32 / 8 / 0.5\text{MB/s}$

CPU用于外设I/O时间占整个CPU时间比例=  $T_{I/O} / T_{\text{total}} = 2.5\%$

## 例题

- 某计算机CPU主频500MHz，CPI为5。假定某外设的数据传输率为0.5MB/s,采用中断方式与主机进行数据传送，以32位为传输单位，对应的中断服务程序包含18条指令，中断服务的其他开销相当于2条指令的执行时间。
- 当外设的数传率为5MB/s时，改用DMA方式。假定DMA传送块大小为5000B，且DMA预处理和后处理的总开销为500个时钟周期，则CPU用于该外设I/O的时间占整个CPU时间的百分比是多少（假定DMA与CPU之间没有访存冲突）

DMA传输阶段不需要占用CPU时间。

传输5000B，需一次DMA，所需CPU开销 $T_{IO}=500 \times T=500/500\text{MHz}$

传输5000B 需要的总时间 $T_{total}=5000/5\text{MB/s}$

CPU用于外设I/O时间占整个CPU时间比例=  $T_{IO}/T_{total}=0.1\%$



## || 通道方式

- 通道的功能
- 通道类型
- 通道结构的发展

## || 通道方式

- DMA方式依赖硬件逻辑支持，随着设备数量的增加，DMA控制器增加，成本也相应增加。必须找出一种方法使DMA技术被更多的设备共享。
- DMA接口的起始准备仍需CPU执行一段程序完成。高速设备的信息是成批传送的，一批数据包含了相当多的数据块，每一数据块都要使DMA接口初始化。数据块连续频繁地传送，其占用CPU的时间就不可忽视了。

## || 通道方式

- 设置专用的**输入输出处理机（通道）**，分担输入输出管理的全部或大部分工作。
- 吸取了DMA技术，增加了软件管理，设有专用通道指令
- 层次性的I/O系统
  - 一个主机可以连接多个通道
  - 一个通道可以管理多个设备控制器
  - 一个设备控制器又可以控制多台设备。

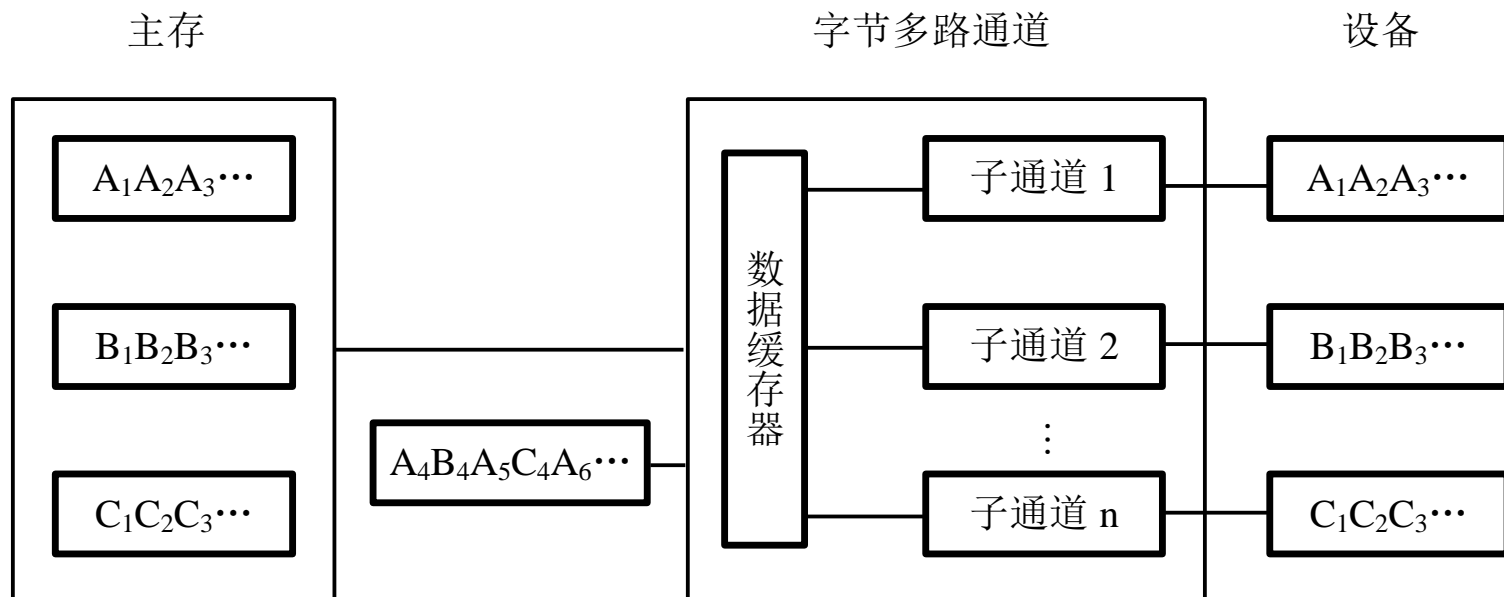
## 通道功能

- 根据CPU要求，组织设备与系统连接和通信；
- 选取通道指令，向设备发出操作命令；
- 指出数据在设备中的位置和主存缓冲区内的位置，组织设备与主存间的数据传输。
- 向CPU反映设备、设备控制器及通道本身的状态信息。
- 将外设和通道本身的中断请求，按次序及时报告CPU。
- 设备控制器介于通道与设备之间，是通道对外部设备实行具体控制的机构。
  - 将通道发送的命令转换为设备能接受的控制信号
  - 向通道反映设备的状态
  - 将设备的各种电平信号转换成通道能识别的标准逻辑信号。

## 通道分类

- 根据设备共享通道的情况及信息传送速度的要求分为3类
  - 字节多路通道
  - 选择通道
  - 数组多路通道

# 字节多路通道



- 包括若干子通道，每个子通道服务于一个慢速设备
- 在一段时间内交替执行多个设备的通道子程序
- 传输单位是字节
- 宏观上这些设备并行工作

## 选择通道

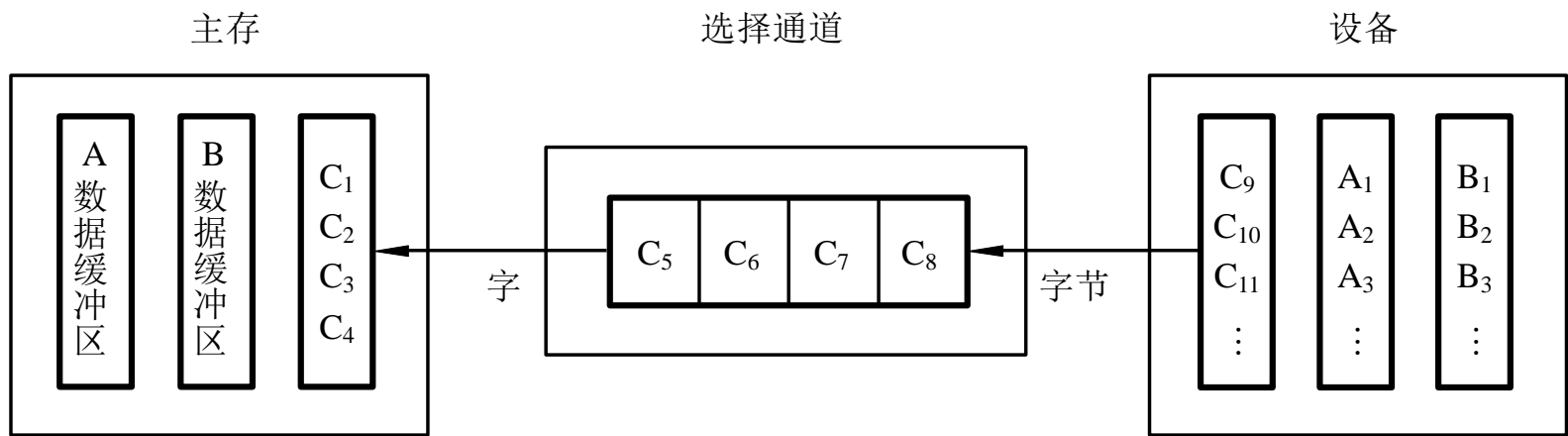
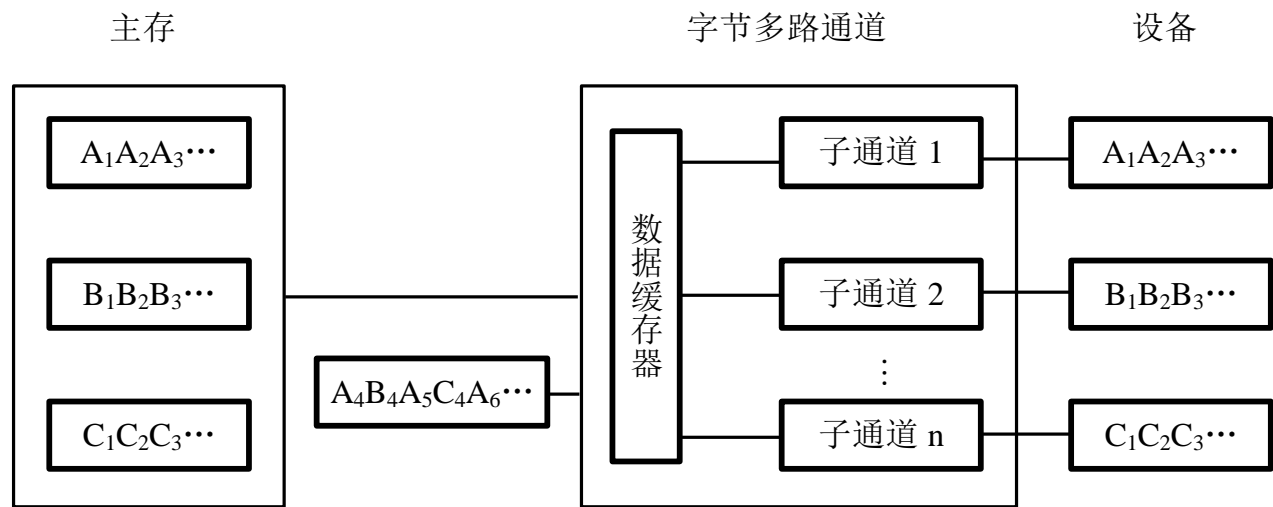
### ■ 字节多路通道

- 适合慢速设备，不适合高速设备
- 高速设备传送两个字节间的空闲很短

### ■ 选择通道

- 设备以成批数据连续传送方式占用通道，直到指定数量的数据全部传送完毕，通道才转为其它设备服务。
- 选择通道在物理上可以连接多个设备，但设备不能同时工作。
- 选择通道只有一个子通道，它适用于大批量数据的高速传送。

# 选择通道





## 数组多路通道

- 通道能高速传送数据，但设备辅助操作时间不能有效利用
  - 如硬盘启动后，平均定位时间较长，磁带机磁头定位时间更长，可达几分钟。导致通道处于等待状态
- 为利用这段时间，将字节多路 and 选择通道折中，称为数组多路通道。
  - 多个设备以数据组（块）为单位交叉使用通道。
  - 设备占用通道时，连续传送一组数据，然后将出让通道使用权
  - 数据组的大小因设备而异，有256B、512B或1KB等。

## 数组多路通道

- 数组多路通道也包含若干个子通道。
  - 数组多路通道适用于中、高速设备，如磁带机、磁盘等。
  - 传送的基本数据单位与字节通道不同。
  - 同一时刻只允许一个设备进行传输型的工作
- 某设备执行辅助操作时
  - 通道暂时断开与该设备的连接，挂起与该设备对应的通道程序
  - 转为其它设备服务，当设备完成了辅助操作，且通道空闲时，通道才重新转为该设备服务。
- 传送效率高，硬件复杂度高

THANKS

计算机组成原理

