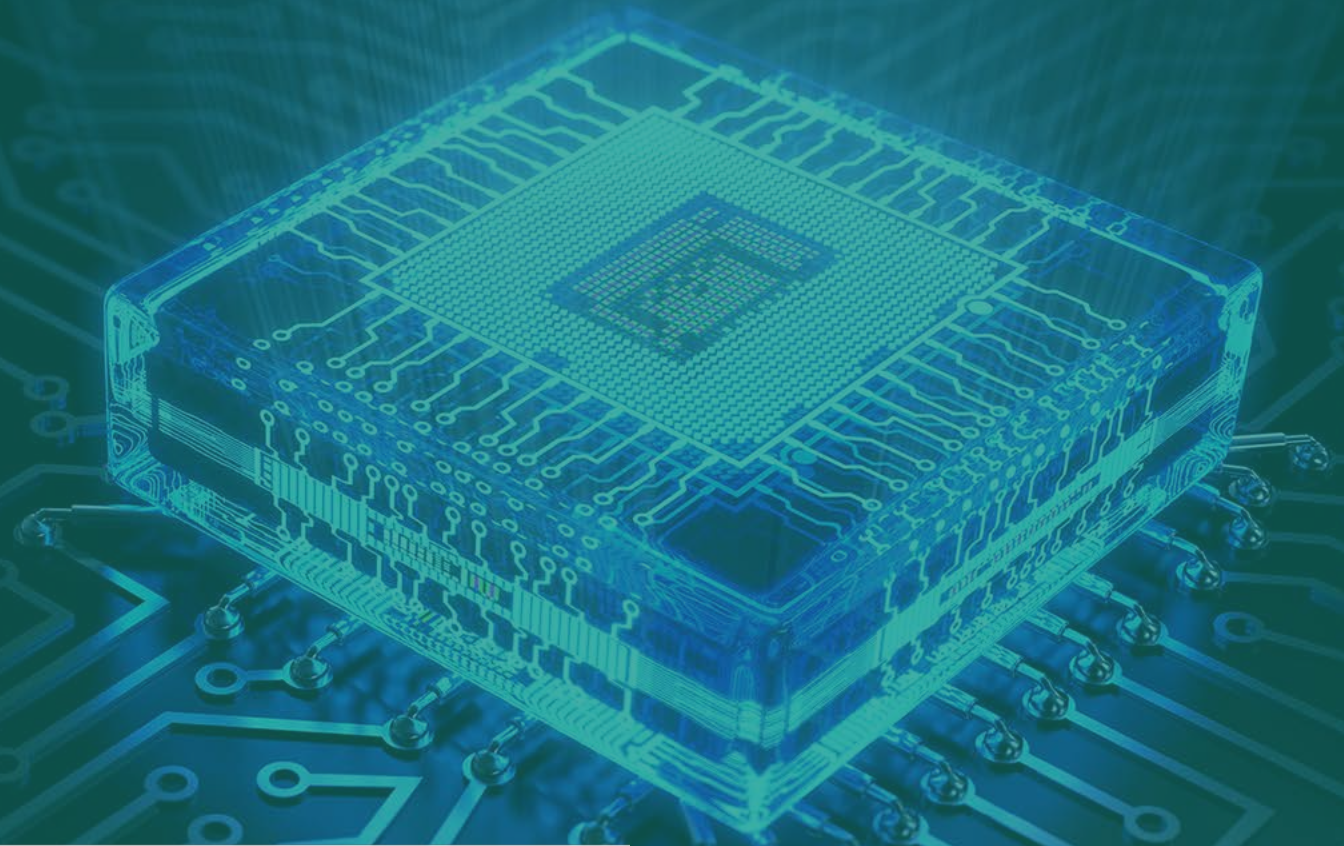
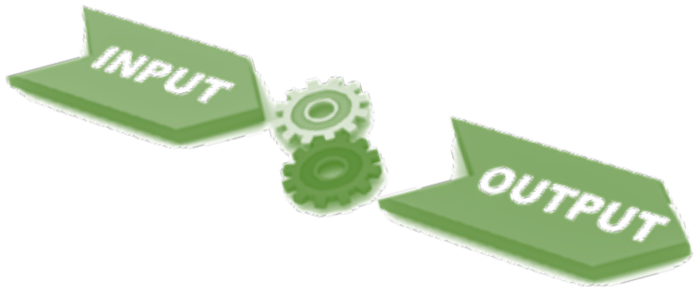
華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST



计算机组成原理

九、输入输出原理



本章主要内容

- 9.1 输入输出设备与特性
- 9.2 I/O接口
- 9.3 数据传送控制方式
- 9.4 程序控制方式
- 9.5 程序中断方式
- 9.6 DMA方式
- 9.7 通道方式
- 9.8 常见I/O设备

9.1 输入输出设备与特性

- 输入输出设备是计算机与人或者机器系统进行数据交互的装置，用于实现计算机内部二进制信息与外部不同形式信息的转换，简称外部设备或外设
  - 输入设备：负责将数据、文字、图像、声音、电信号等转换成计算机可以识别的二进制信息，如键盘、鼠标、扫描仪、摄像头等；
  - 输出设备：负责将计算机处理结果转换成数字、文字、图形、图像、声音或电信号，如显示器、打印机等；
  - 输入输出设备：既能输入也能输出，如磁盘、网卡等。
- 输入输出设备特性
  - 异步性、实时性、独立性

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I/O系统
  - 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

- 主要功能
  - 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

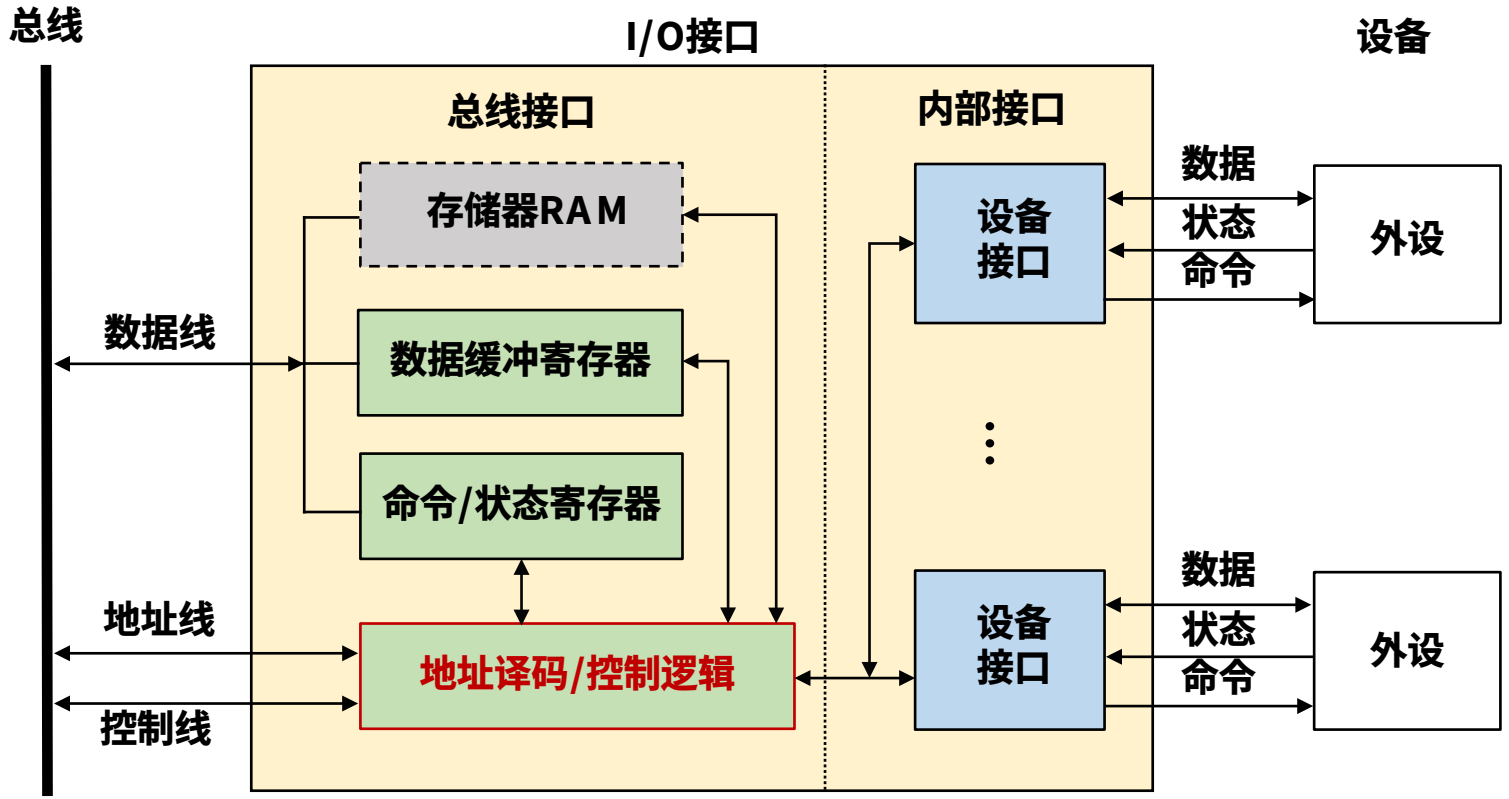
本章主要内容

- 9.1 输入输出设备与特性
- 9.2 I/O接口
- 9.3 数据传送控制方式
- 9.4 程序控制方式
- 9.5 程序中断方式
- 9.6 DMA方式
- 9.7 通道方式
- 9.8 常见I/O设备

I/O接口定义与功能

- I/O接口：连接总线与I/O设备的物理和逻辑界面
  - 既包括物理连接电路，也包括软件逻辑接口
  - 所有设备均通过 I/O接口（总线接口）与总线相连
  - CPU使用设备地址经总线与I/O接口通信访问I/O设备
  - 标准接口有利于提升I/O系统的独立性，降低连接复杂度
- I/O接口功能
  - 设备寻址、数据交互、设备控制
  - 状态检测、数据缓冲、格式转换

I/O接口结构

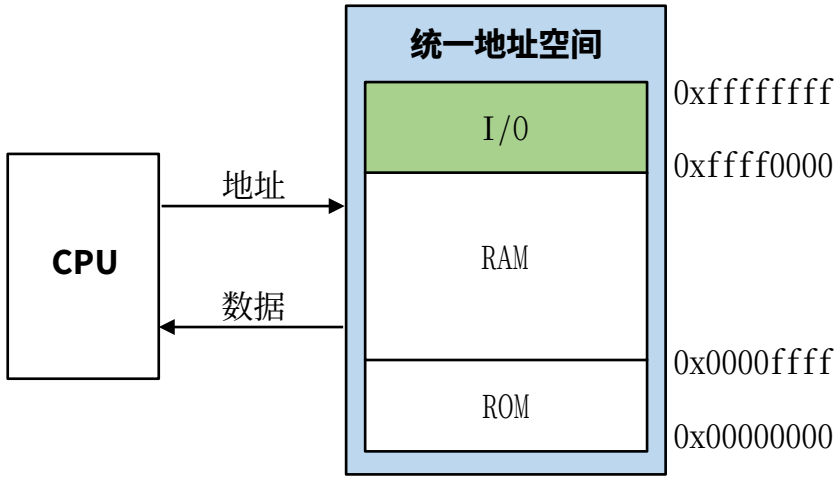


- I/O接口的功能
  - 设备寻址
  - 数据交互
  - 设备控制
  - 状态检测
  - 数据缓冲
  - 格式转换

I/O接口编址

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
- 访存指令访问外设，访问什么设备取决于地址



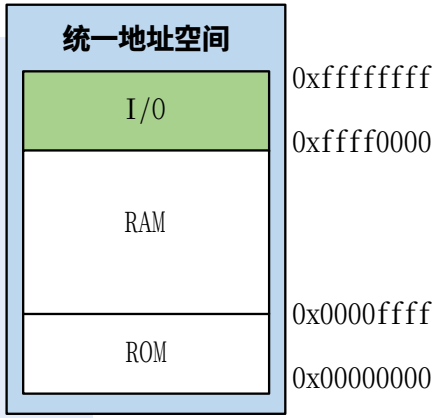
独立编址

- 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

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**

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

```
OUT  DX, AL          I/O 写：将 AL 寄存器中的字节写入 DX 寄存器对应的 I
IN   AL, DX          I/O 读：从 DX 寄存器对应的 I/O 端口地址读取一个字
MOV  [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接口的软件

- 现代计算机中用户须通过OS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

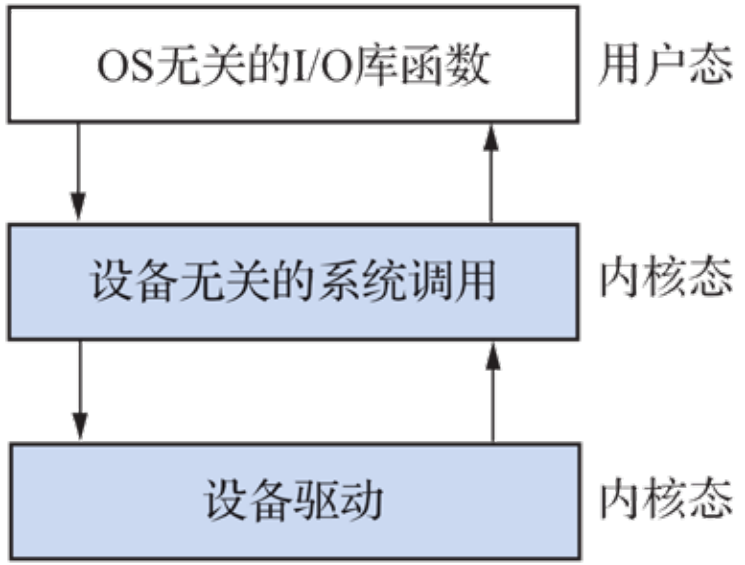
  用户程序主要通过调用I/O库访问设备，方便程序在不同OS间移植
- 与设备无关的OS调用库 （内核态）

  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



I/O接口分类

- 按数据传送方式
  - 并行、串行接口
- 按接口的灵活性
  - 可编程接口、不可编程接口
- 按通用性： **通用、专用接口**
- 按总线传输的通信方式：**同步、异步接口**
- 按访问外设的方式
  - 程序控制方式、程序中断方式、DMA及通道处理机接口

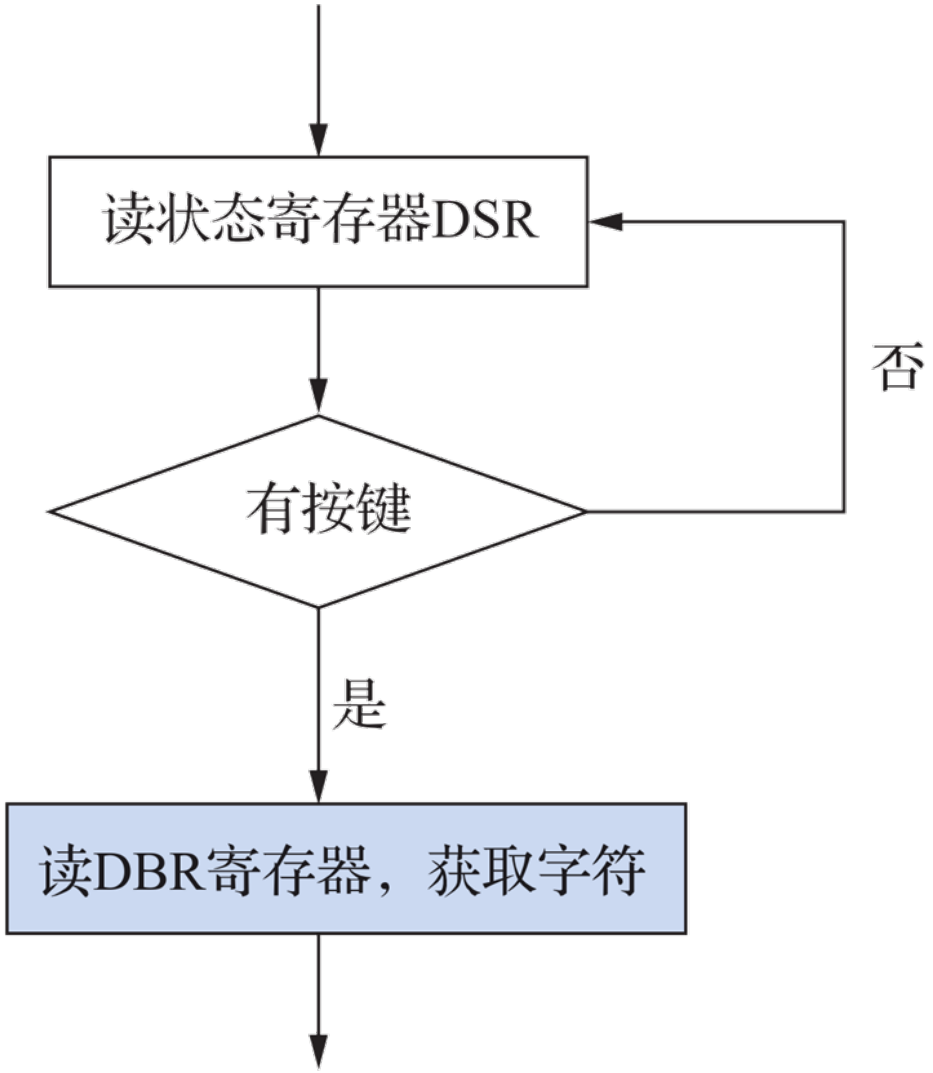
本章主要内容

- 9.1 输入输出设备与特性
- 9.2 I/O接口
- **9.3 数据传送控制方式**
- 9.4 程序控制方式
- 9.5 程序中断方式
- 9.6 DMA方式
- 9.7 通道方式
- 9.8 常见I/O设备

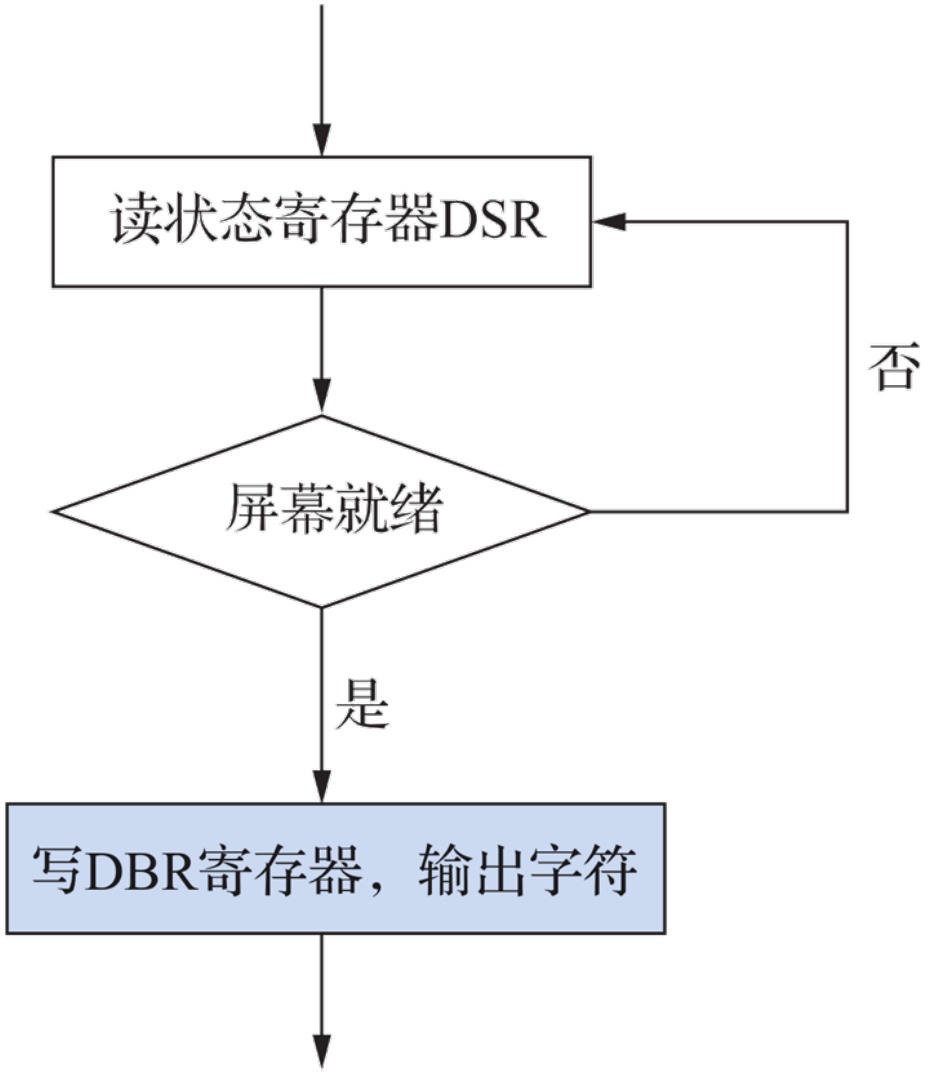
数据传输控制方式

- **程序控制方式**
  - 程序查询方式、直接输入输出方式
- 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

简单设备查询流程

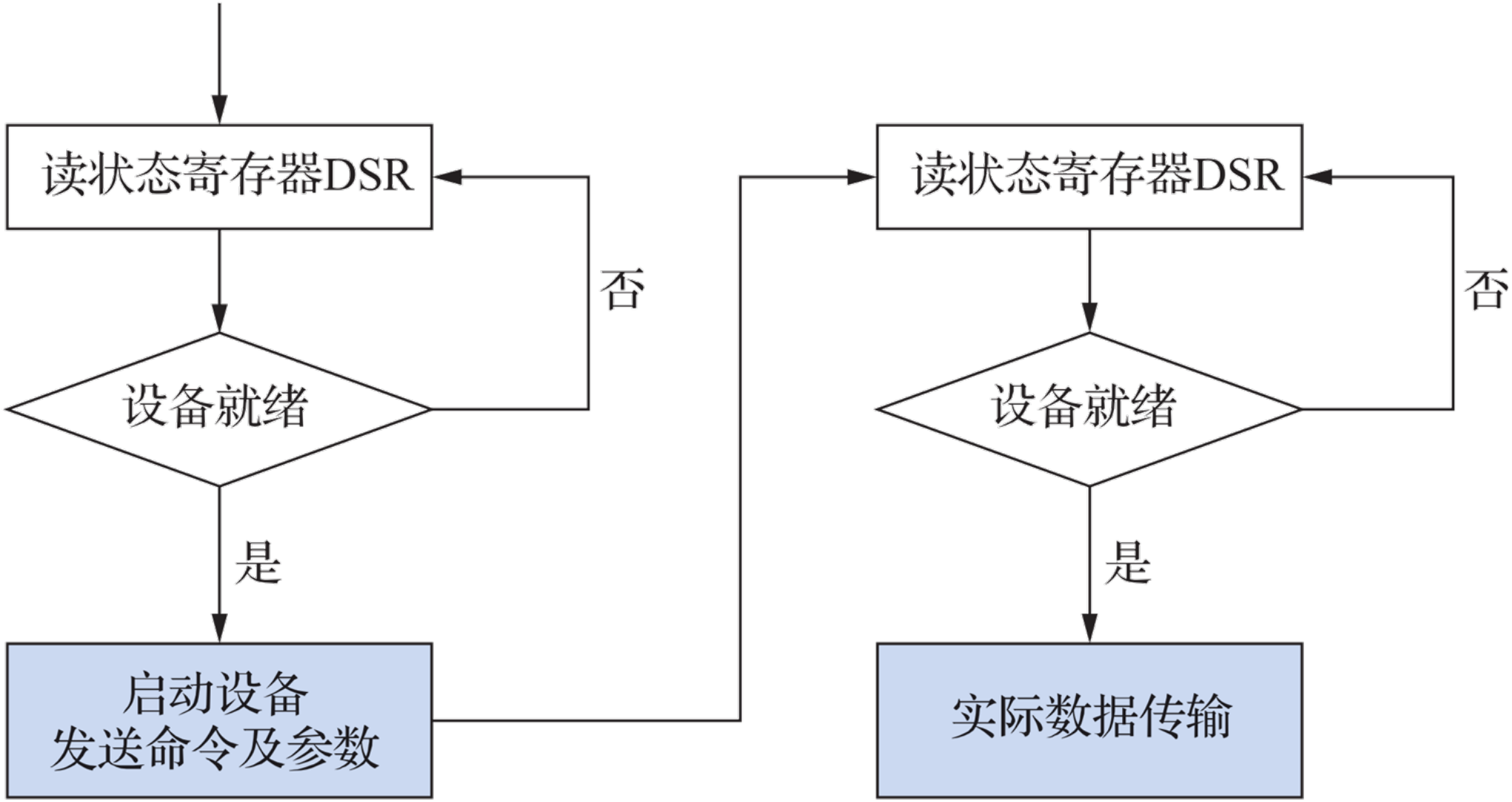


（a）键盘程序查询流程

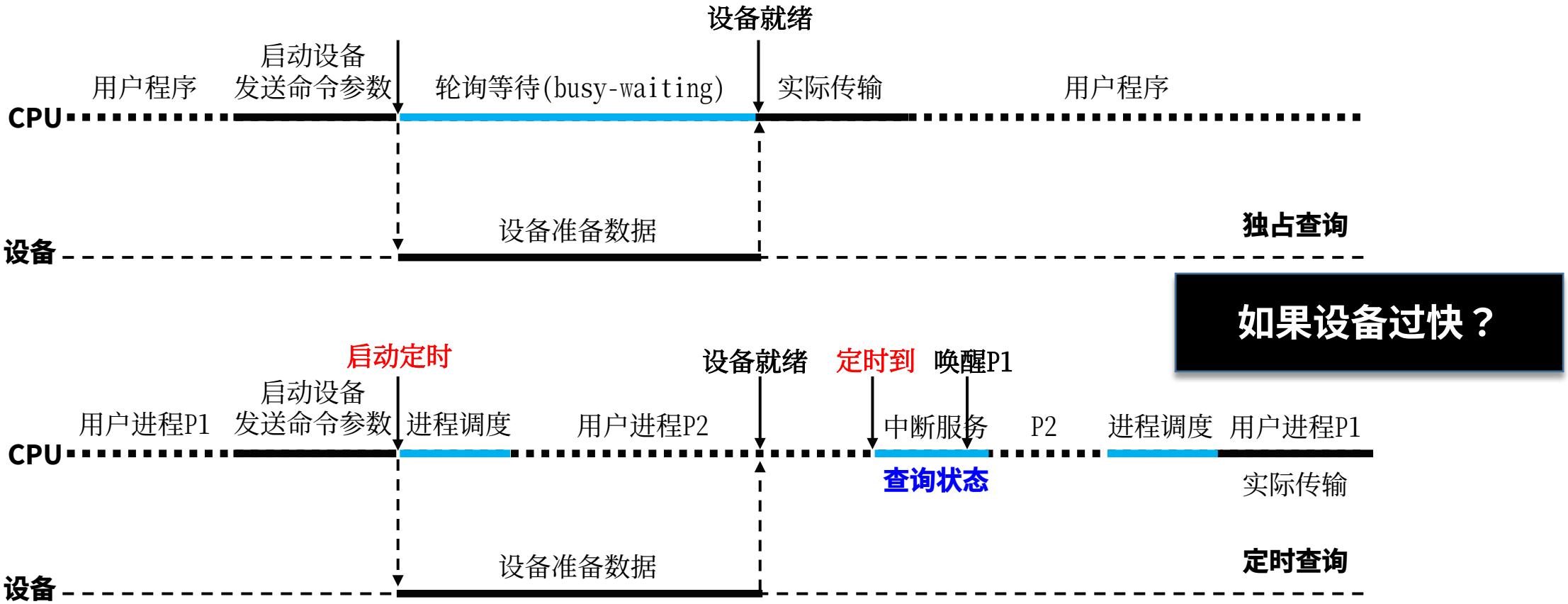


（b）字符终端程序查询流程

复杂设备查询流程



程序查询方式运行轨迹


设备就绪
启动设备
用户程序
发送命令参数
轮询等待(busy-waiting)
实际传输
用户程序
CPU
设备准备数据
设备
独占查询
如果设备过快？
启动定时
设备就绪
定时到
唤醒P1
启动设备
用户进程P1
发送命令参数
进程调度
用户进程P2
中断服务
P2
进程调度
用户进程P1
CPU
查询状态
实际传输
设备准备数据
设备
定时查询

例题

- 例 9.1 假设某程序查询方式的输入输出系统采用定时轮询方式，每次定时中断服务开销需要 400 个时钟周期，CPU 的时钟频率为 200MHz，包括鼠标和硬盘两个外部设备，求两种不同外部设备进行 I/O 操作时的 CPU 时间占用率。

  （1）鼠标以字节为单位进行数据传输，假设每秒必须进行 50 次轮询才能保证不会错过任何鼠标操作，每次轮询成功后的实际数据传输需要 13 个时钟周期。

- CPU 每秒用于鼠标 I/O 操作的时间为 (400T + 13T)×50 =20650T
- 鼠标 I/O 的 CPU 时间占用率 = 20650T/1s ≈ 0.01%
- 由此可见，对鼠标进行定时轮询操作基本不影响 CPU 的性能

例题

- 例 9.1 假设某程序查询方式的输入输出系统采用定时轮询方式，每次定时中断服务开销需要 400 个时钟周期，CPU 的时钟频率为 200MHz，包括鼠标和硬盘两个外部设备，求两种不同外部设备进行 I/O 操作时的 CPU 时间占用率。

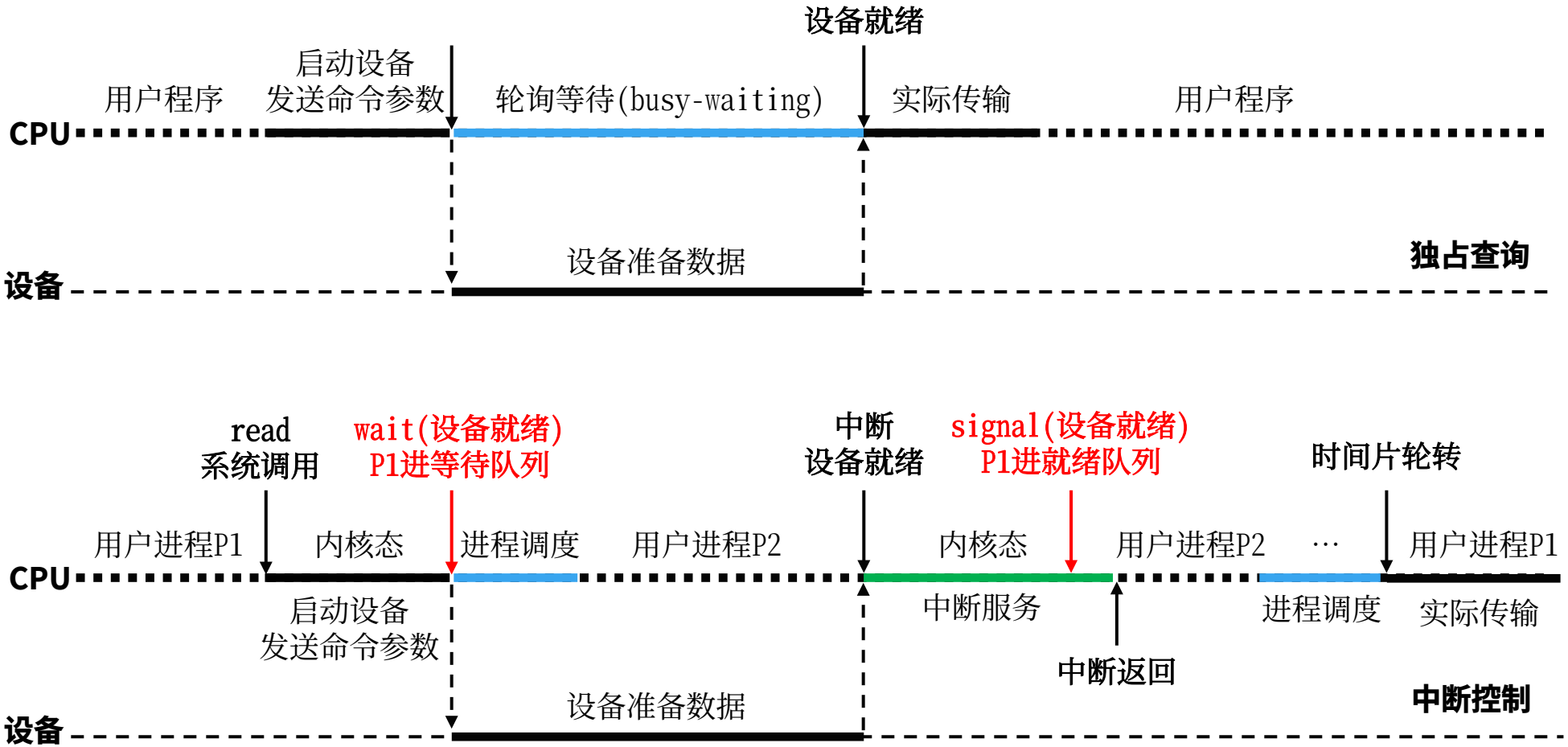
  （2）硬盘以512字节的扇区为单位传输数据，启动阶段发送命令和参数需要90个时钟周期，实际传输阶段需要 1555 个时钟周期，CPU 访问硬盘的速率为 20MB/s。

- 每秒传输的次数为：20MB / 512B = 39062.5 次（数传率、频率均以 10 为基数）
- 理想情况下每次传输只定时轮询一次
- 每个扇区传输开销 = 启动开销90T + 中断开销400T + 数据传输开销 1555T = 2045T
- 总 CPU 占用率 = 2045T×39062.5/1s = 39.94%。

本章主要内容

- 9.1 输入输出设备与特性
- 9.2 I/O接口
- 9.3 数据传送控制方式
- 9.4 程序控制方式
- **9.5 程序中断方式**
- 9.6 DMA方式
- 9.7 通道方式
- 9.8 常见I/O设备

中断控制方式


设备就绪
启动设备
发送命令参数
用户程序
轮询等待(busy-waiting)
实际传输
用户程序
CPU
设备准备数据
设备
独占查询
read
系统调用
wait(设备就绪)
P1进等待队列
中断
设备就绪
signal(设备就绪)
P1进就绪队列
时间片轮转
用户进程P1
内核态
进程调度
用户进程P2
内核态
用户进程P2
…
用户进程P1
CPU
启动设备
发送命令参数
中断服务
进程调度
实际传输
中断返回
设备准备数据
中断控制
设备

中断优势

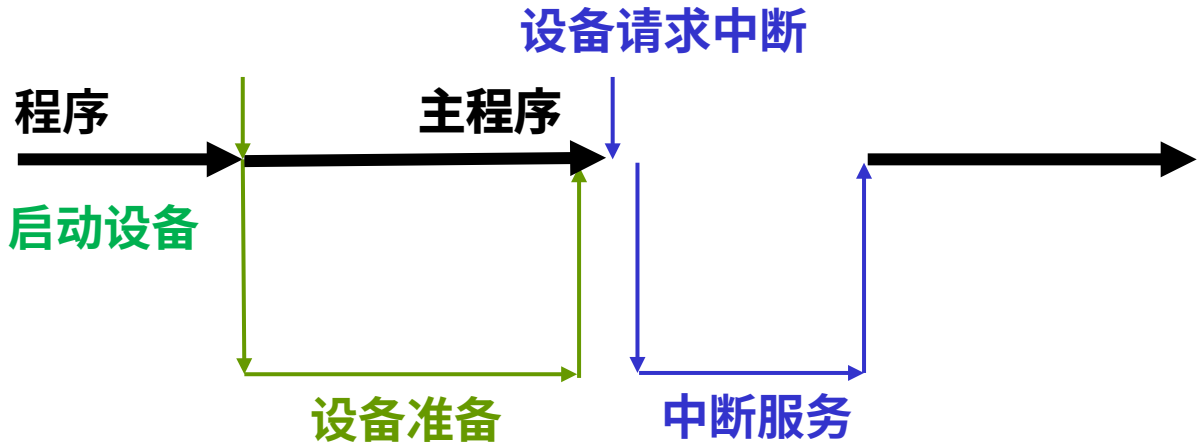
- 提高了CPU的使用效率
  - 主动告知机制避免了反复查询设备状态
  - 仍需CPU占用（中断服务子程序运行时间+中断开销）
- 适合随机出现的服务
- 需要专门的硬件

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

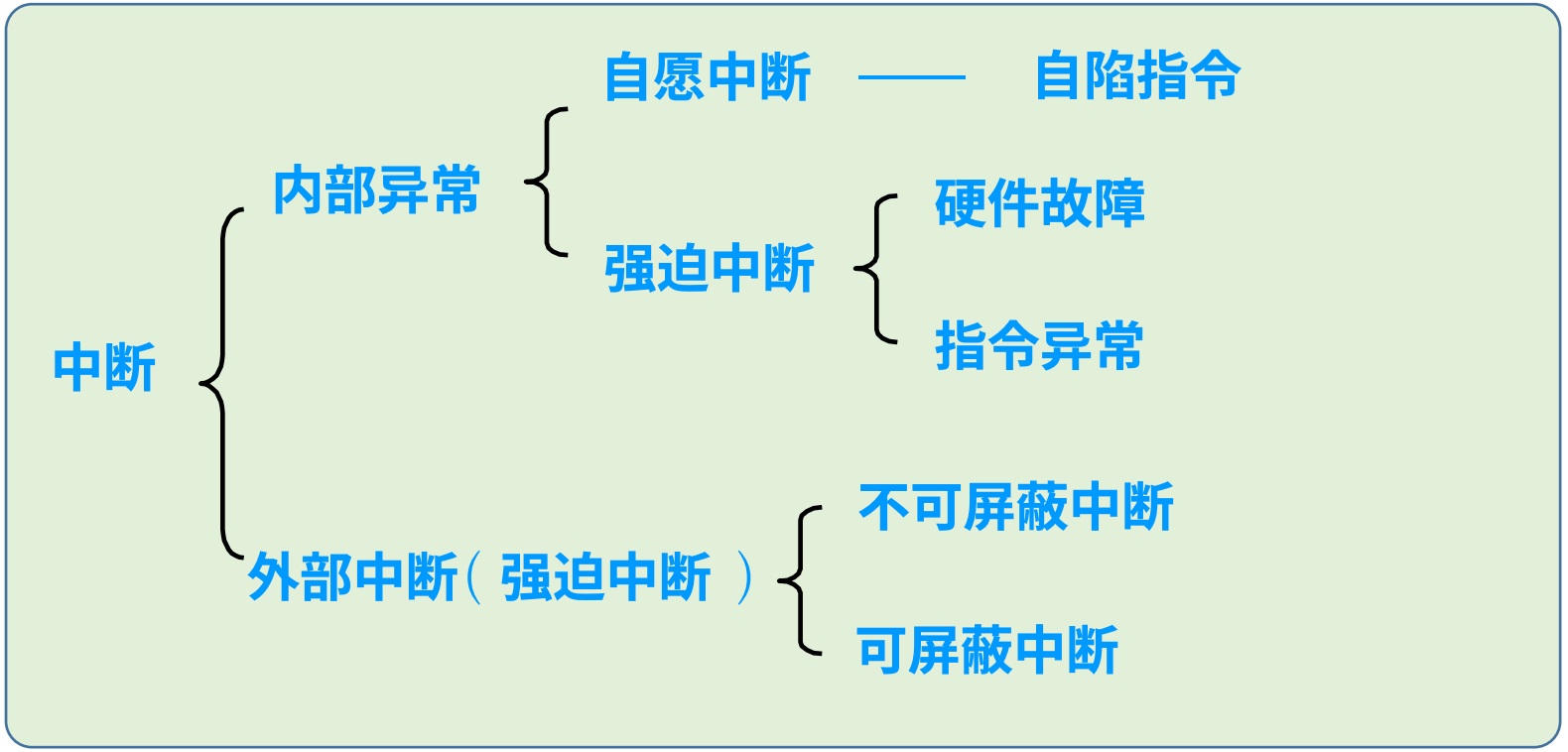
中断基本概念

- CPU暂时中止现行程序的执行，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
  - 原因：不处理会体验不好，会丢失数据，甚至造成灾难
- 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

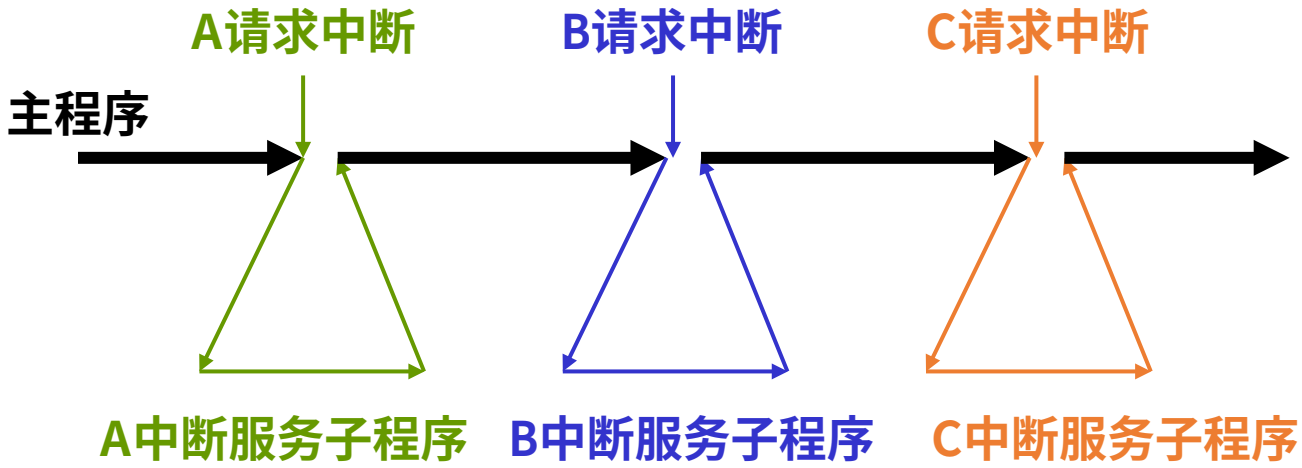


中断的分类与作用

- 中断技术赋于计算机应变能力，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

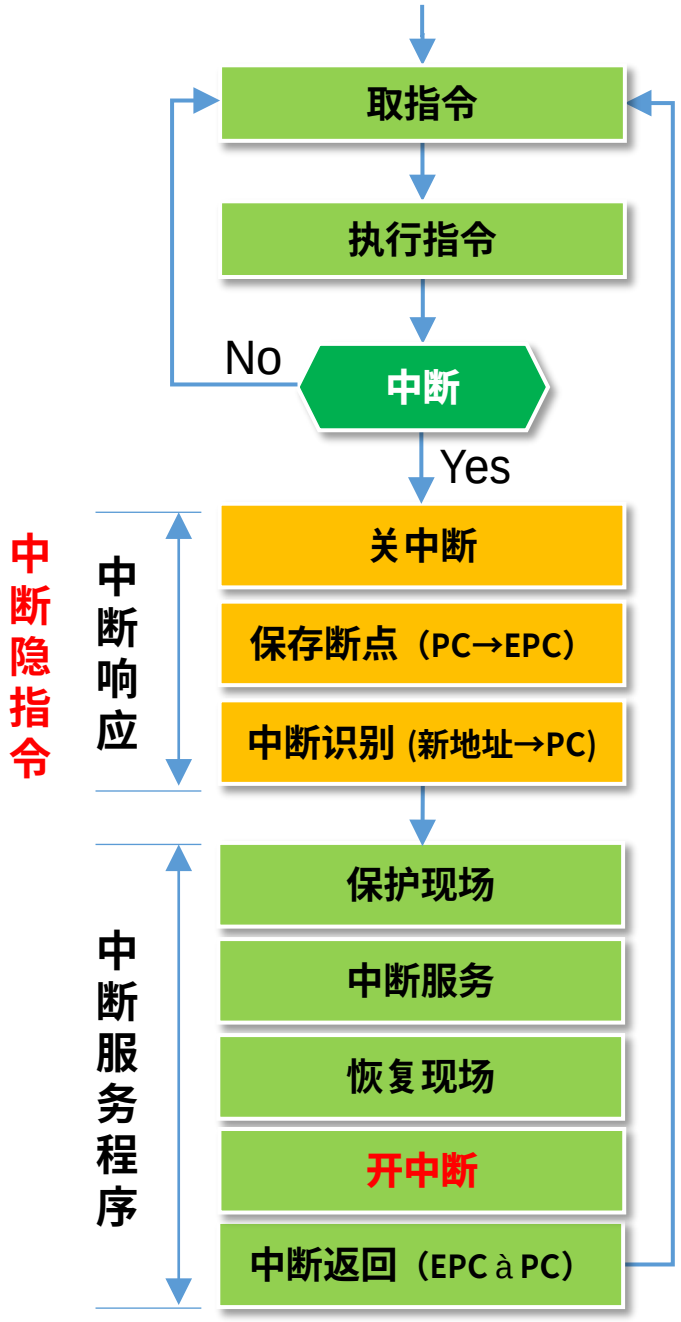


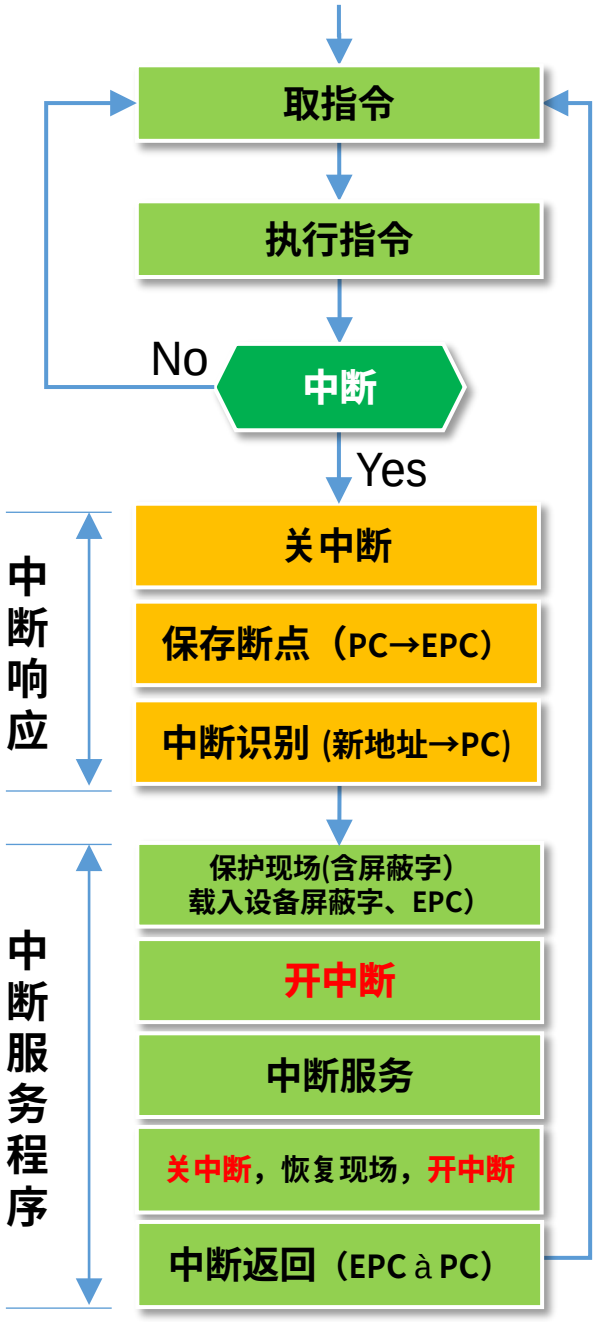
程序中断处理示意图



- 子程序与中断服务子程序的异同？
  - 共同点：调用后要返回
  - 差异：调用方式（显式/随机），保存寄存器（中断更多），返回位置与方式（调用位置/中断位置）
- 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?


取指令
执行指令
No
中断
Yes
中断隐指令
中断响应
关中断
保存断点（PC→EPC）
中断识别 (新地址→PC)
中断服务程序
保护现场
中断服务
恢复现场
开中断
中断返回（EPC à PC）

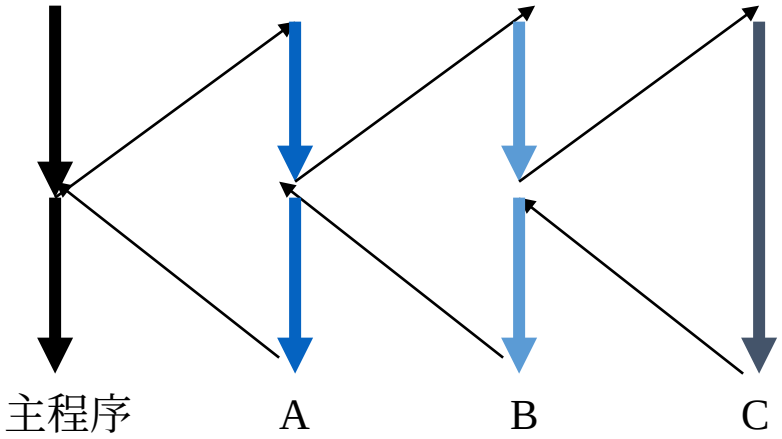

取指令
执行指令
No
中断
Yes
中断响应
关中断
保存断点（PC→EPC）
中断识别 (新地址→PC)
中断服务程序
保护现场(含屏蔽字)
载入设备屏蔽字、EPC）
开中断
中断服务
关中断，恢复现场，开中断
中断返回（EPC à PC）

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？

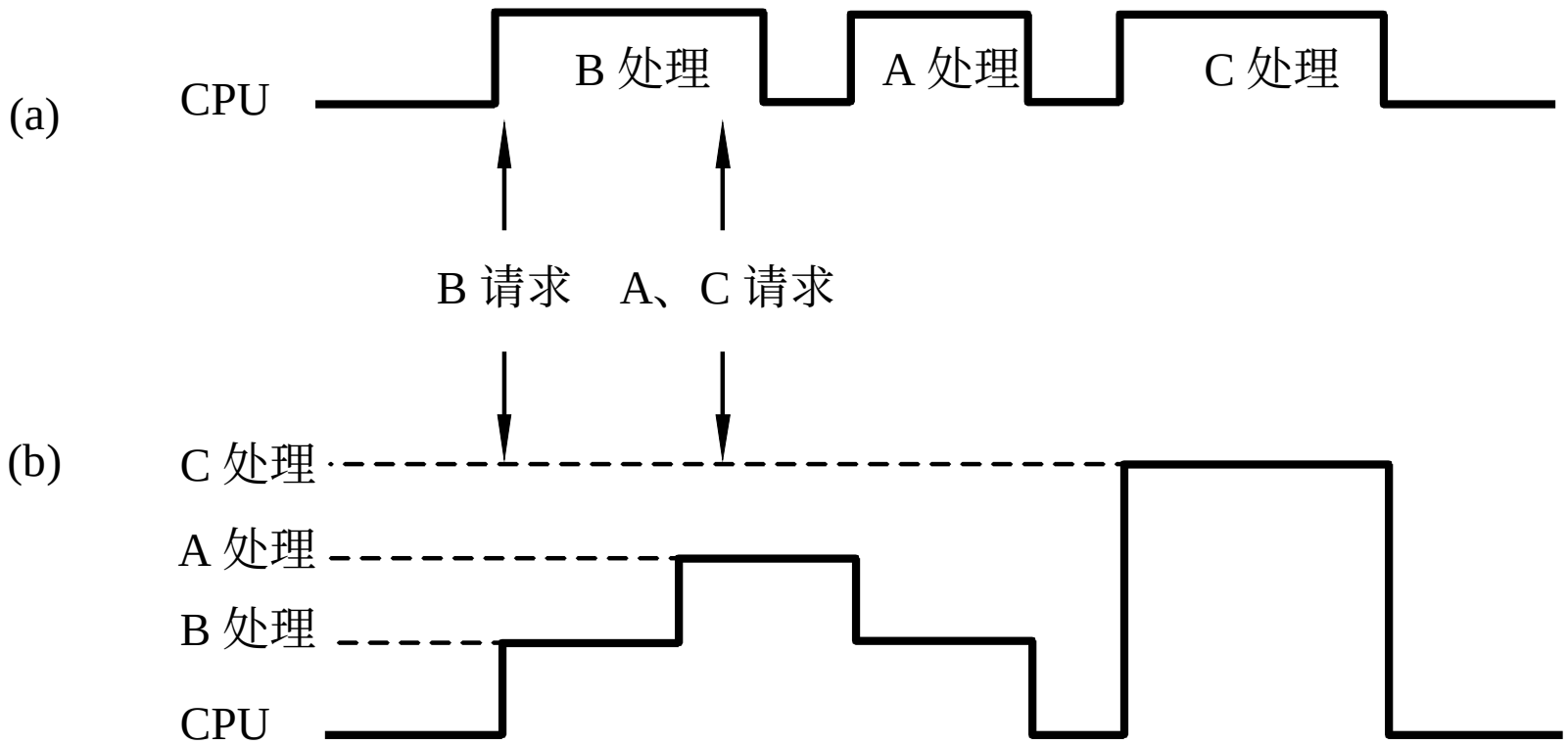
单级中断与多级中断

- 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**



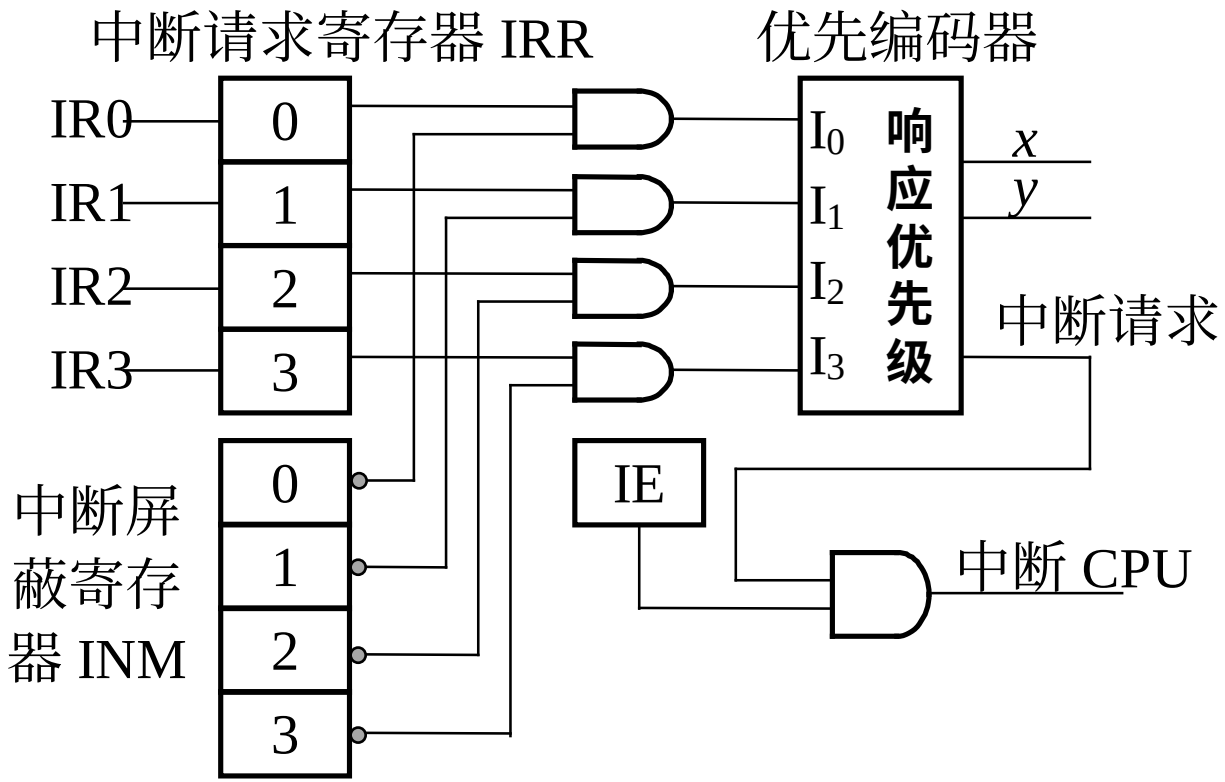
划分优先级的一般规律

- 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

中断屏蔽

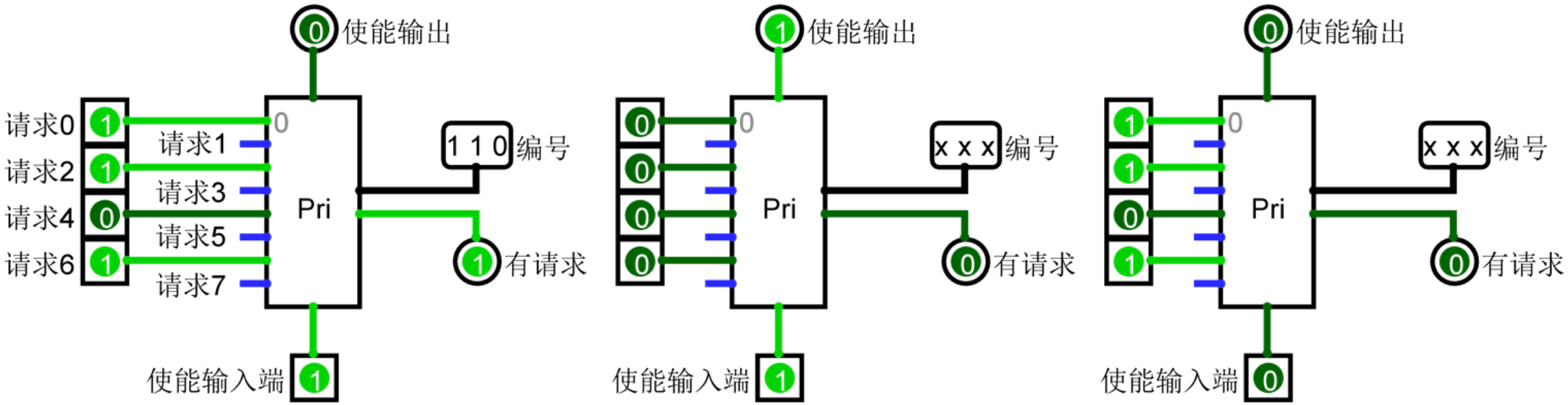
- 响应优先级
  - CPU对各设备中断请求进行响应，并准备好处理的先后次序
  - 这种次序往往在硬件线路上已固定，不便于变动
- 处理优先级
  - CPU实际对各中断请求处理的先后次序
  - 如果不使用屏蔽技术，响应的优先级就是处理优先级
- 中断屏蔽技术可动态改变各设备的处理优先级

中断屏蔽方式



- 优先编码器决定响应优先级，中断屏蔽改变处理优先级
- 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？

优先编码器

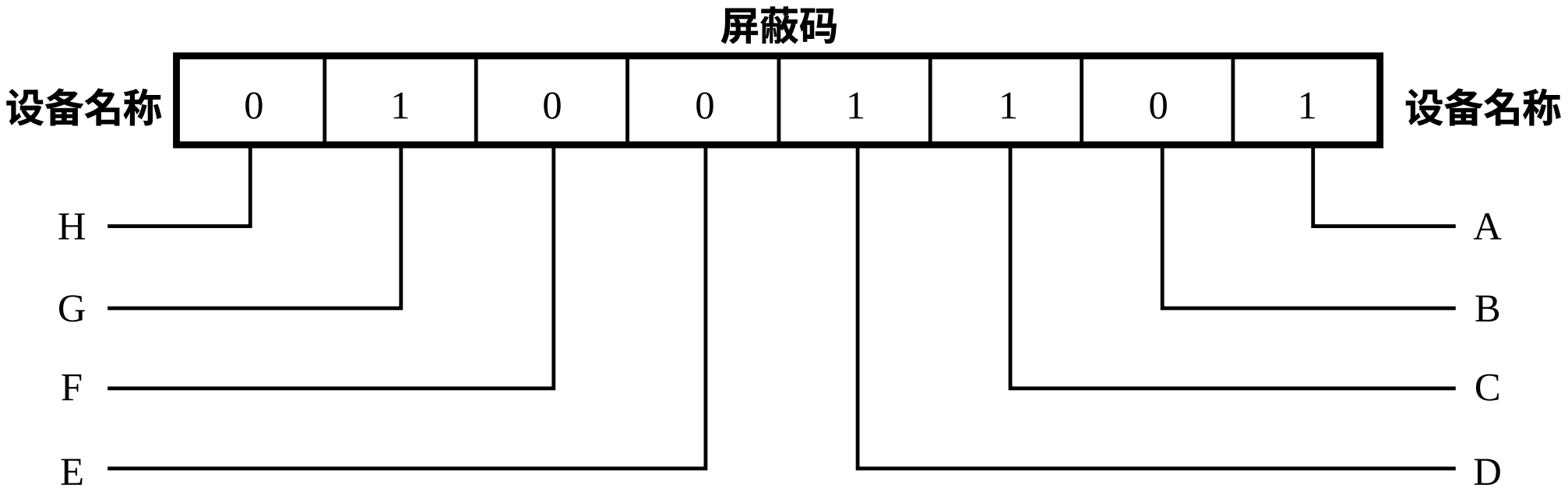

0
使能输出
请求0
1
请求1
请求2
1
请求3
请求4
0
请求5
请求6
1
请求7
0
Pri
1 1 0
编号
1
有请求
使能输入端
1
1
使能输出
0
0
0
0
0
Pri
x x x
编号
0
有请求
使能输入端
1
0
使能输出
1
0
0
Pri
x x x
编号
0
有请求
使能输入端
0

中断屏蔽位

- 中断请求寄存器IRR
  - 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（中断使能寄存器）

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

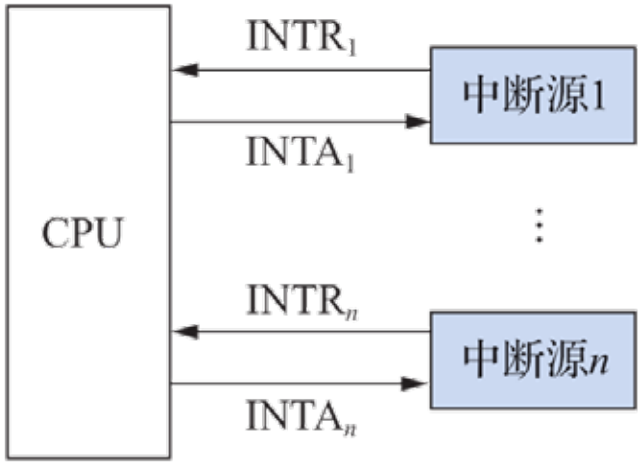
- 假定硬件原来的响应顺序为0>1>2，试设置各设备中断屏蔽字，将中断优先级改为1>2>0。

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

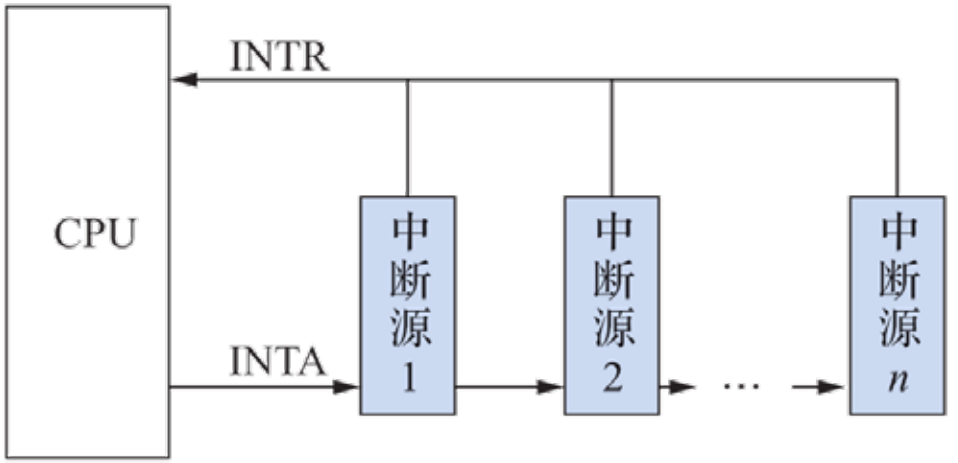
优先级实现---中断仲裁

- 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

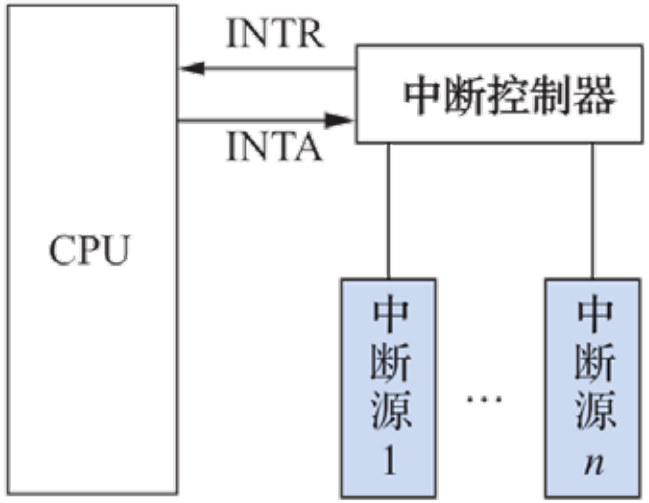
中断请求信号的传输方式



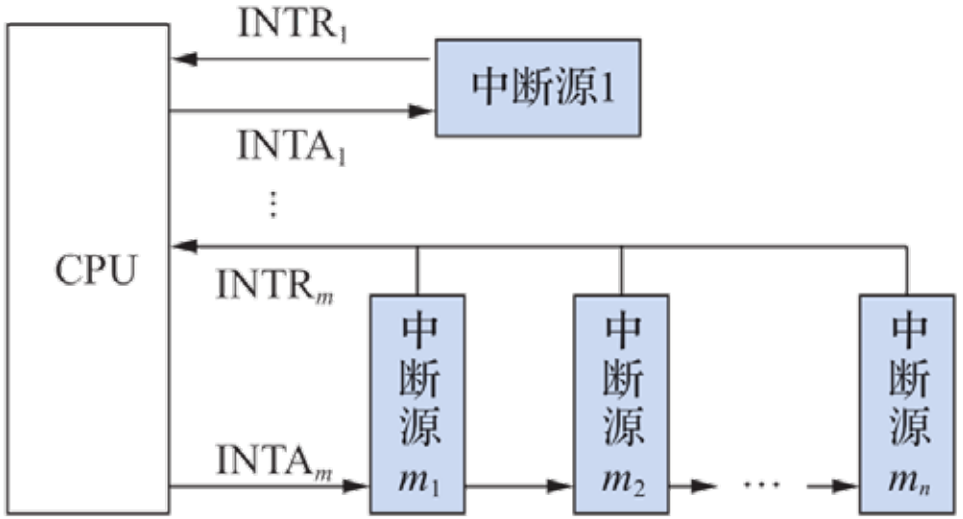
（a）独立请求方式



（b）链式请求方式



（c）中断控制器方式

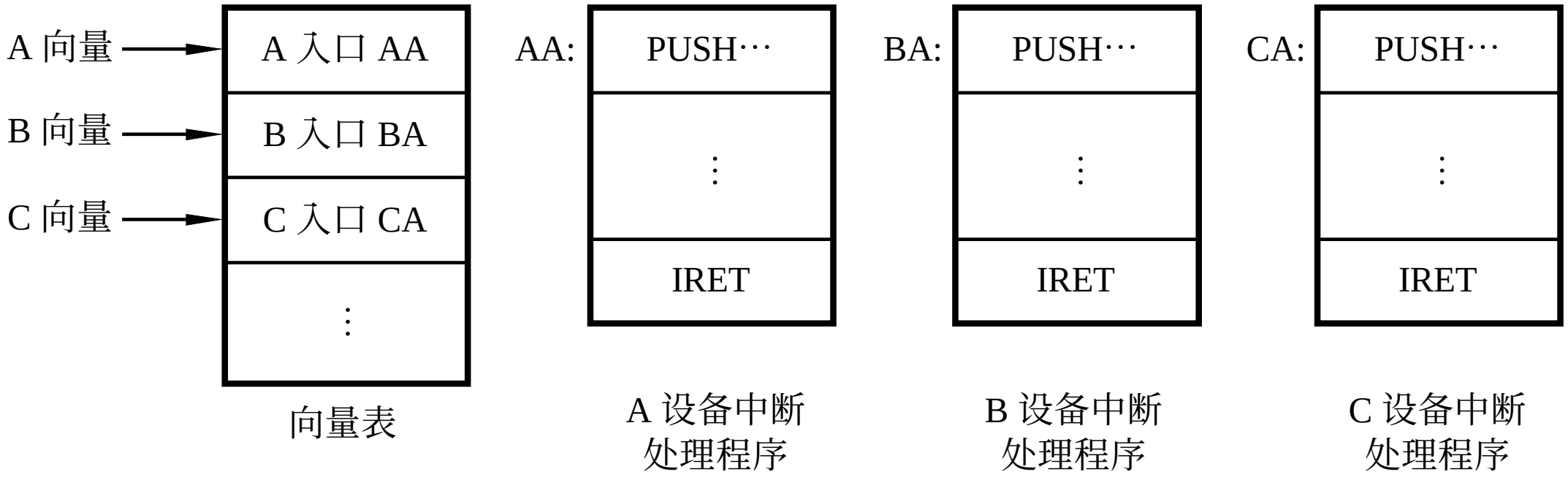


（d）分组链式

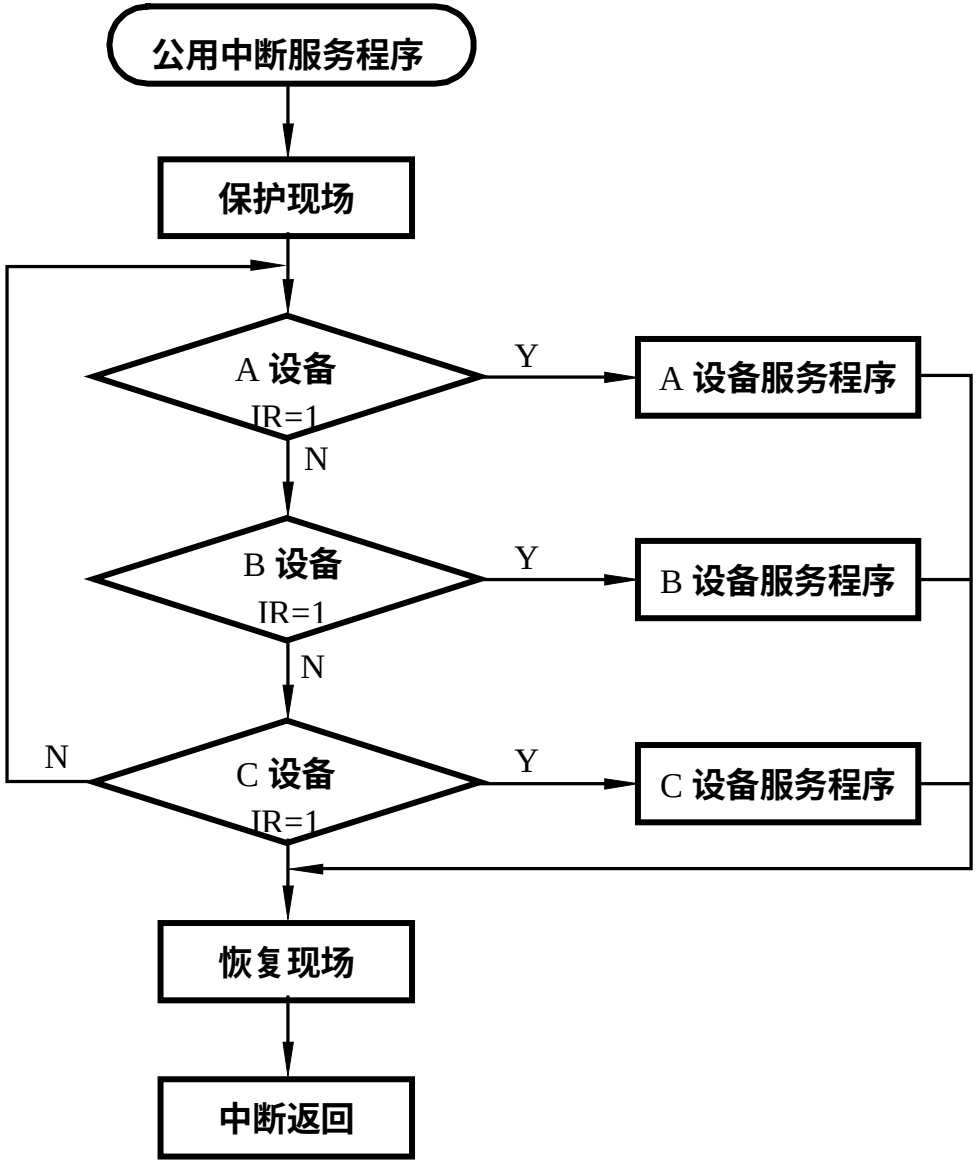
中断识别（寻找入口地址）

- 独立请求：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中断号（向量地址），查中断向量表取得服务程序入口，转入相应服务程序。
    - 硬件查询法
- 中断共享：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，查询特定端口(GPIO)识别中断源
    - 程序识别　　（软件方法）

中断向量法

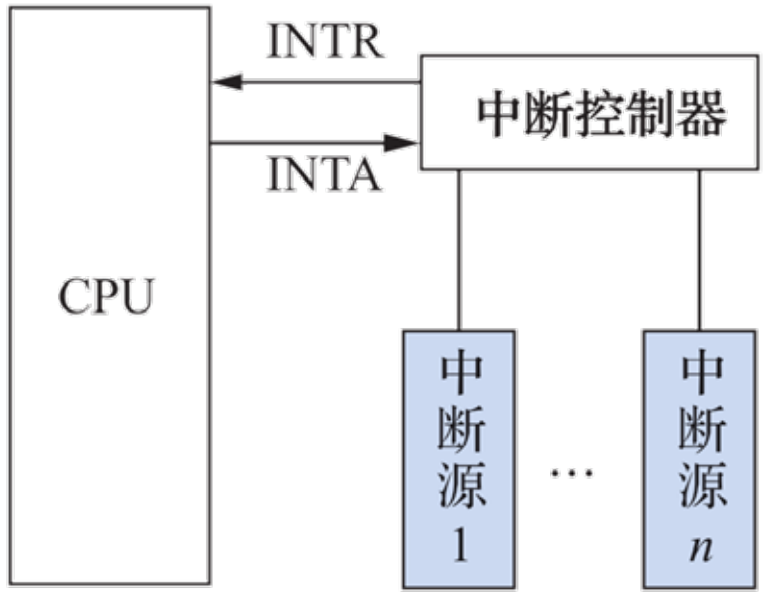

A 向量
B 向量
C 向量
A 入口 AA
B 入口 BA
C 入口 CA
向量表
AA:
PUSH…
IRET
A 设备中断
处理程序
BA:
PUSH…
IRET
B 设备中断
处理程序
CA:
PUSH…
IRET
C 设备中断
处理程序

程序识别


公用中断服务程序
保护现场
A 设备
IR=1
Y
A 设备服务程序
N
B 设备
IR=1
Y
B 设备服务程序
N
C 设备
IR=1
N
Y
C 设备服务程序
恢复现场
中断返回

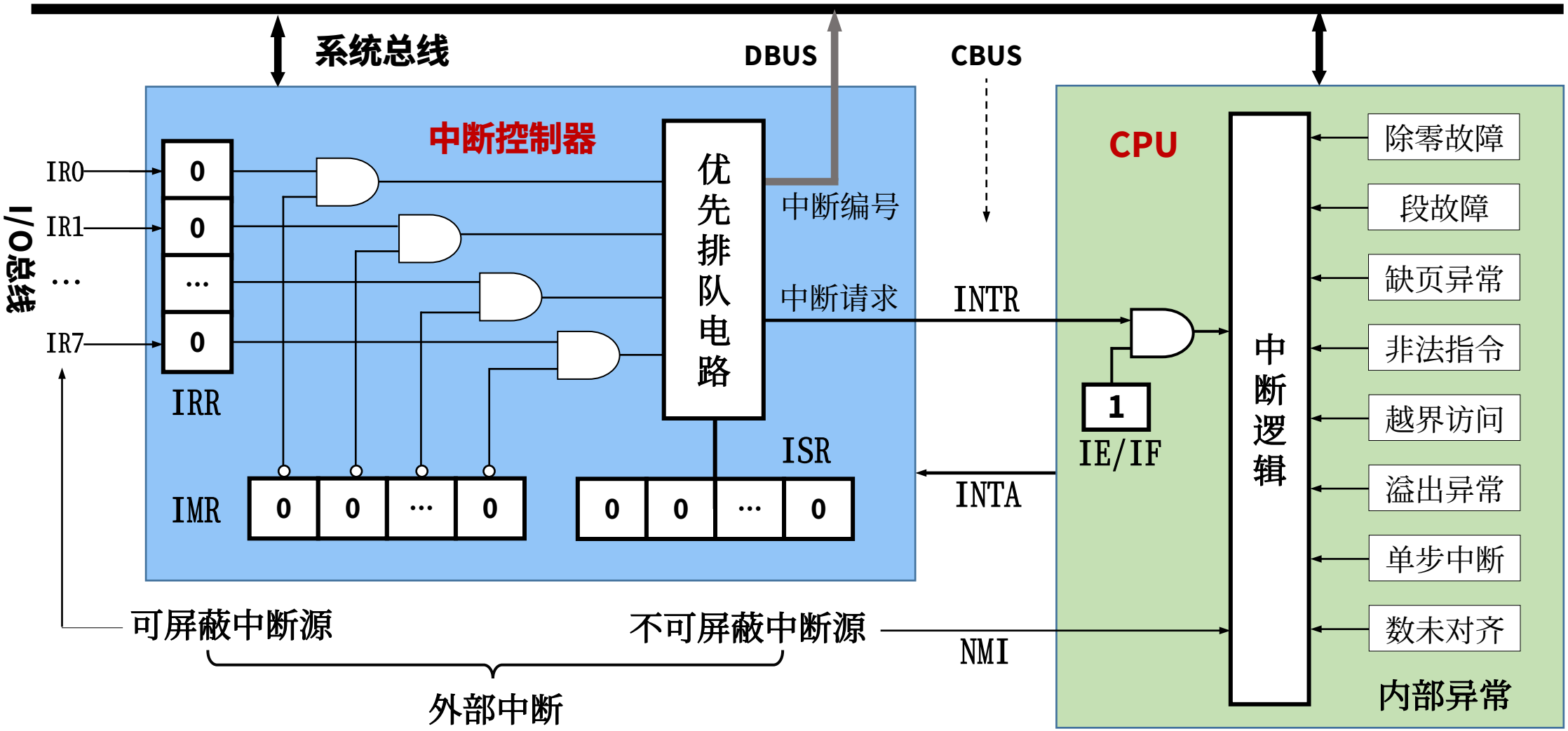
中断处理中的问题

- 中断响应条件
  - 中断允许触发器IE=1
  - 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
- 保存现场，恢复现场
  - 中断程序用到的通用寄存器，EPC，屏蔽字
  - 缺页异常的断点和外部中断断点不一致
- 中断过程由软硬件结合完成

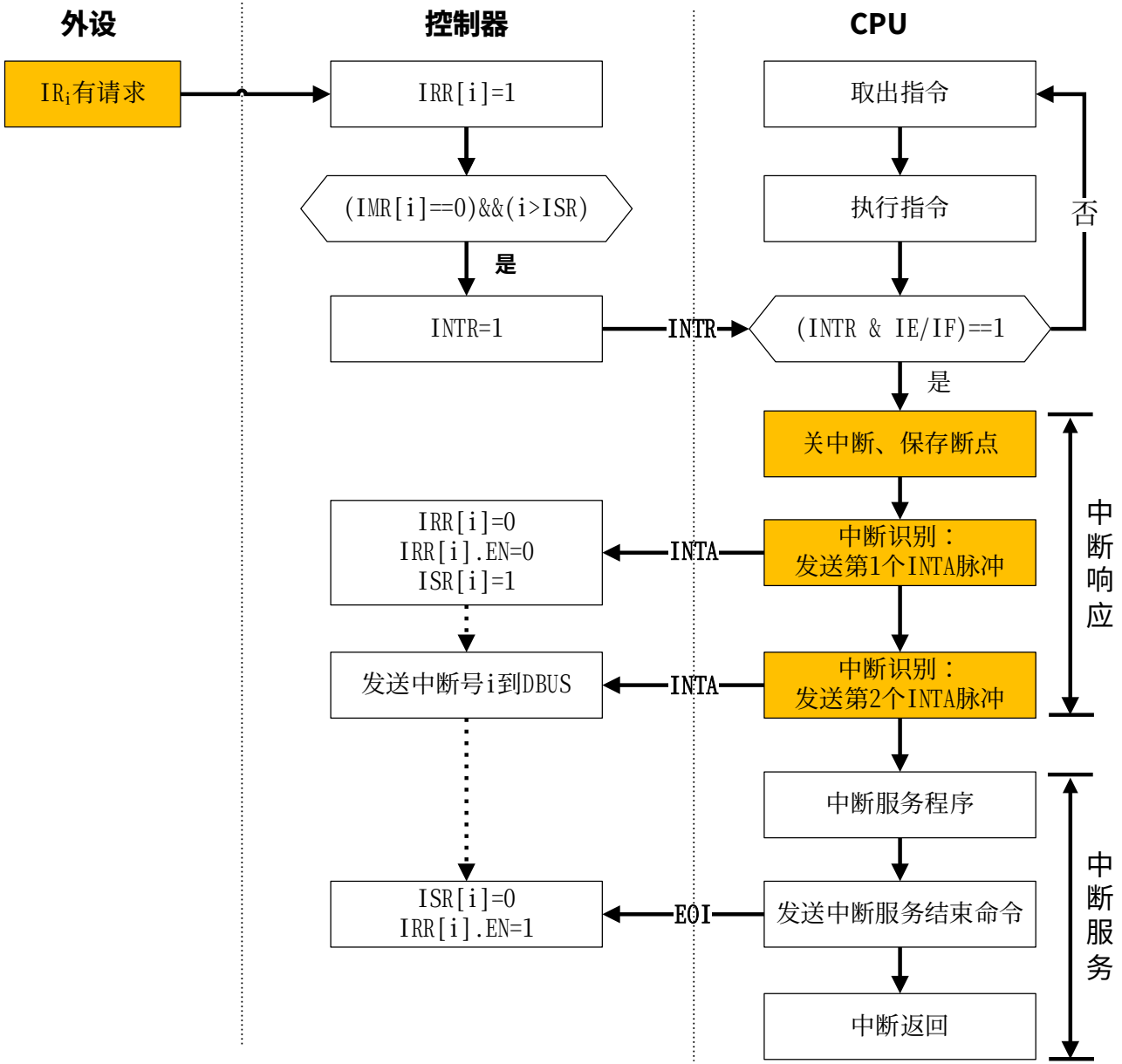


（c）中断控制器方式

中断硬件接口


系统总线
DBUS
CBUS
中断控制器
CPU
I/O总线
IR0
IR1
…
IR7
0
0
…
0
IRR
IMR
0
0
…
0
优先排队电路
ISR
0
0
…
0
中断编号
中断请求
INTR
INTA
1
IE/IF
中断逻辑
除零故障
段故障
缺页异常
非法指令
越界访问
溢出异常
单步中断
数未对齐
内部异常
可屏蔽中断源
不可屏蔽中断源
NMI
外部中断

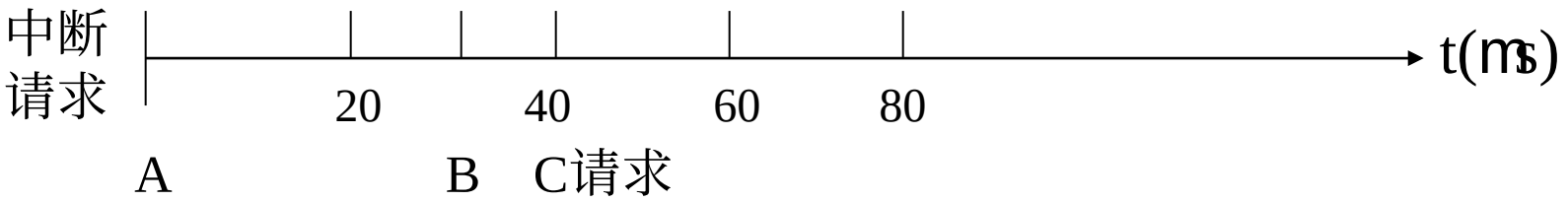
中断控制器工作流程


外设
控制器
CPU
IRi有请求
IRR[i]=1
取出指令
(IMR[i]==0)&&(i>ISR)
执行指令
否
是
INTR=1
INTR
(INTR & IE/IF)==1
是
关中断、保存断点
IRR[i]=0
IRR[i].EN=0
ISR[i]=1
INTA
中断识别：
发送第1个INTA脉冲
中断响应
发送中断号i到DBUS
INTA
中断识别：
发送第2个INTA脉冲
中断服务程序
ISR[i]=0
IRR[i].EN=1
EOI
发送中断服务结束命令
中断返回
中断服务

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＞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

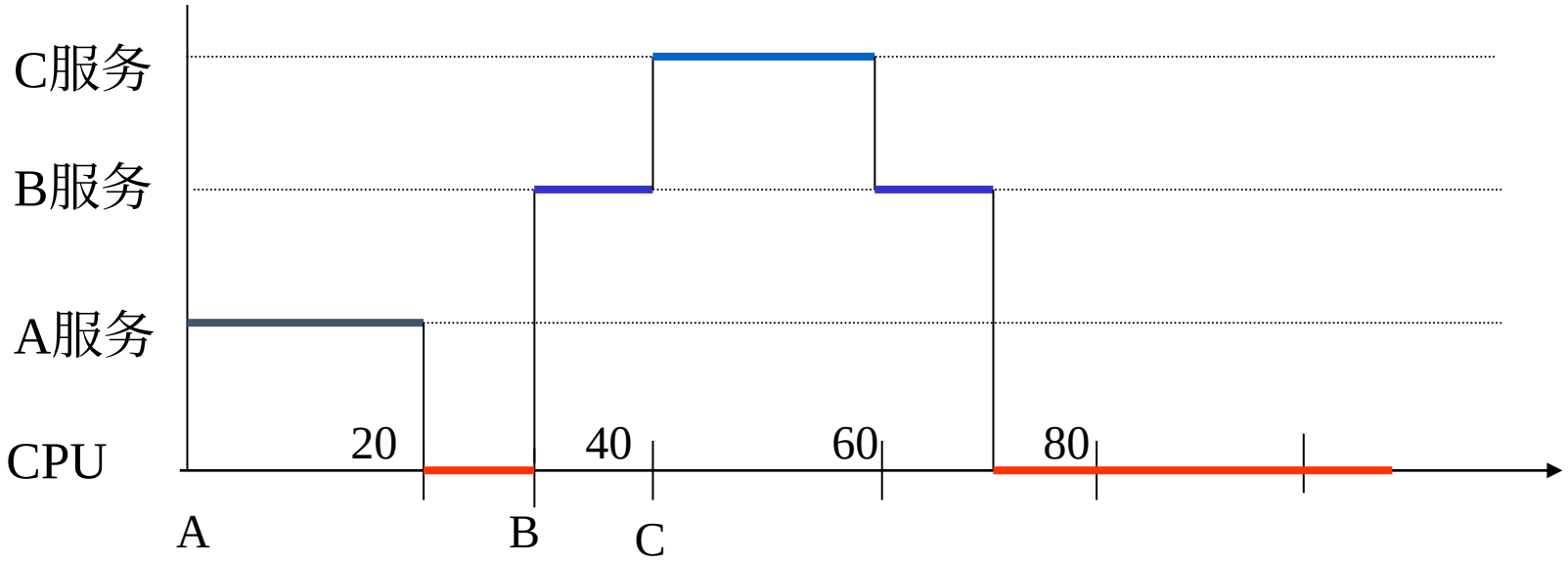
请按下图所示时间轴给出的设备中断请求时刻，画出CPU执行程序的轨迹。A、B、C中断服务程序的时间宽度均为20 ms。



解：从中断屏蔽字看出，其处理优先级为：

A>C>B>CPU

故CPU运行轨迹如下：

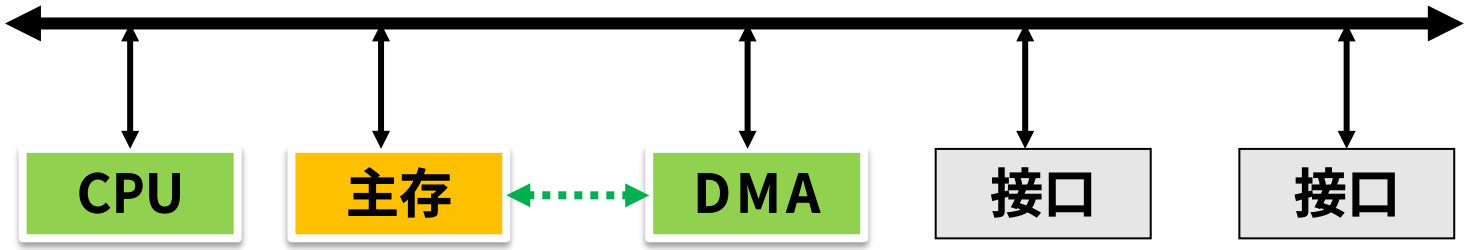


本章主要内容

- 9.1 输入输出设备与特性
- 9.2 I/O接口
- 9.3 数据传送控制方式
- 9.4 程序控制方式
- 9.5 程序中断方式
- **9.6 DMA方式**
- 9.7 通道方式
- 9.8 常见I/O设备

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DMA硬件管理的数据通路（虚拟通路，还是通过系统总线）
  - 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



DMA方式

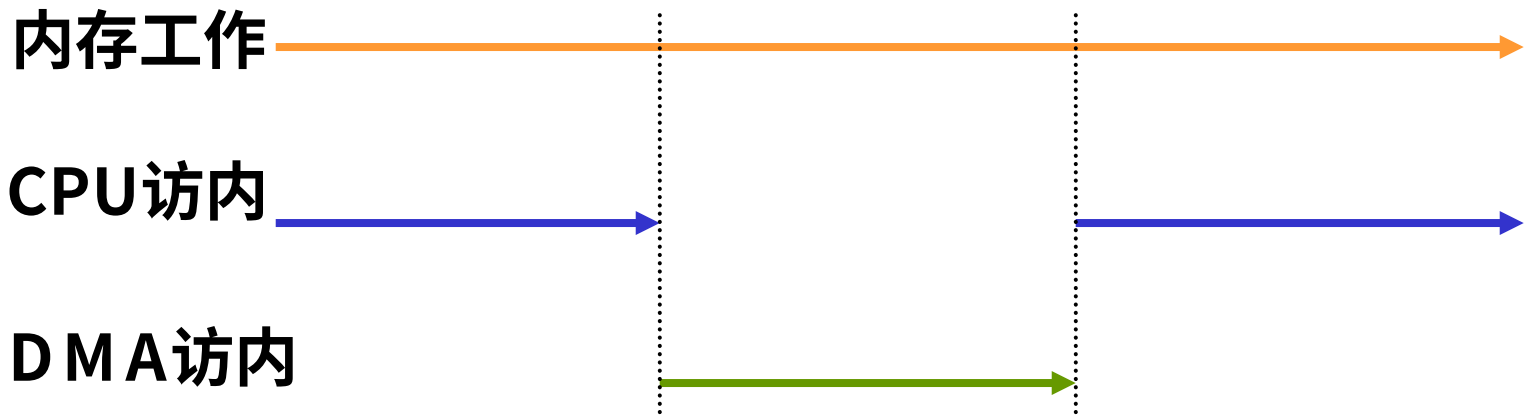
- DMA基本概念
- DMA传输方式
- 基本DMA控制器

内存争用

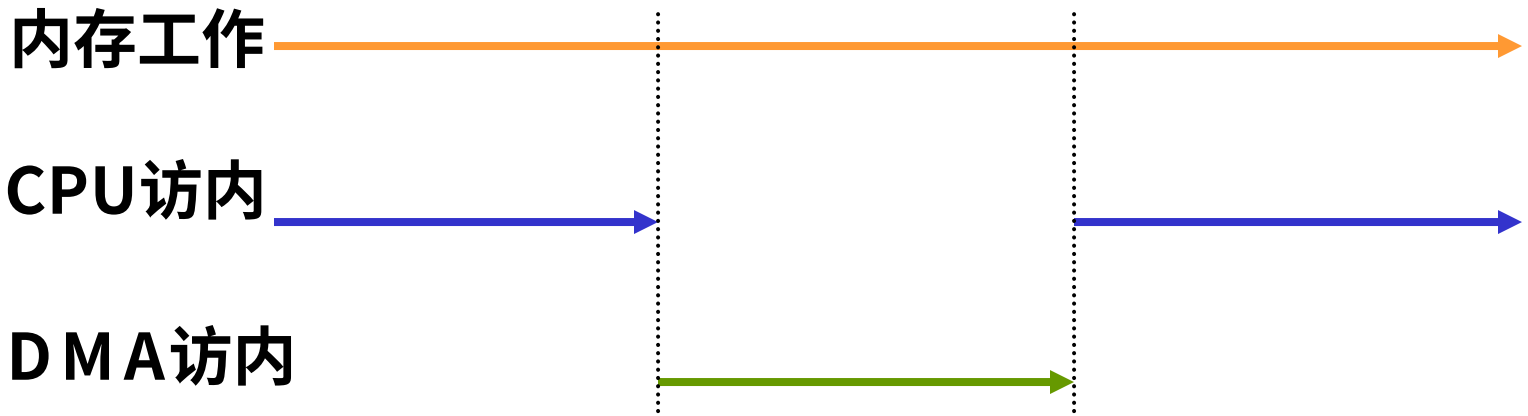
- 访存冲突
  - DMA控制器直接访问内存
  - CPU执行主程序　(需要访内)
- 如何处理这种冲突？
  - 停止CPU使用主存
  - DMA与CPU交替使用主存
  - 周期挪用法

停止CPU使用主存

- DMA传送数据时，CPU停止使用主存
- 一批数据传送结束后，DMA再交还主存使用权
- DMA传送过程中，CPU处于等待状态

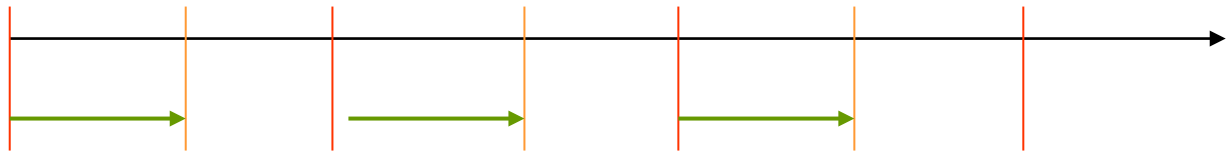


停止CPU访内



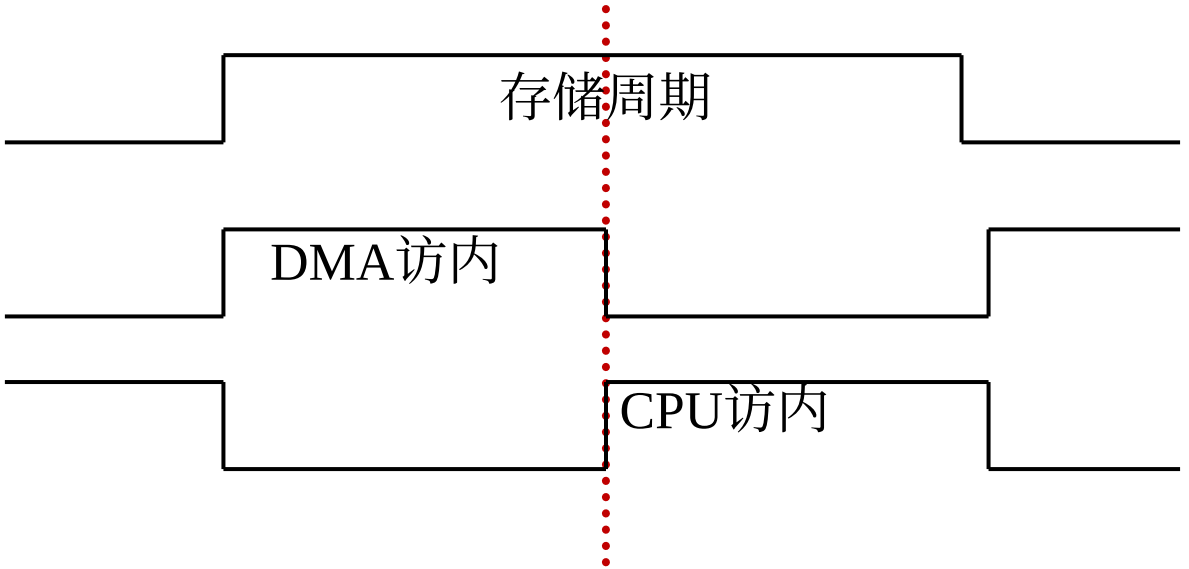
1. DMA批量数据传输周期过长，CPU长期无法访内

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



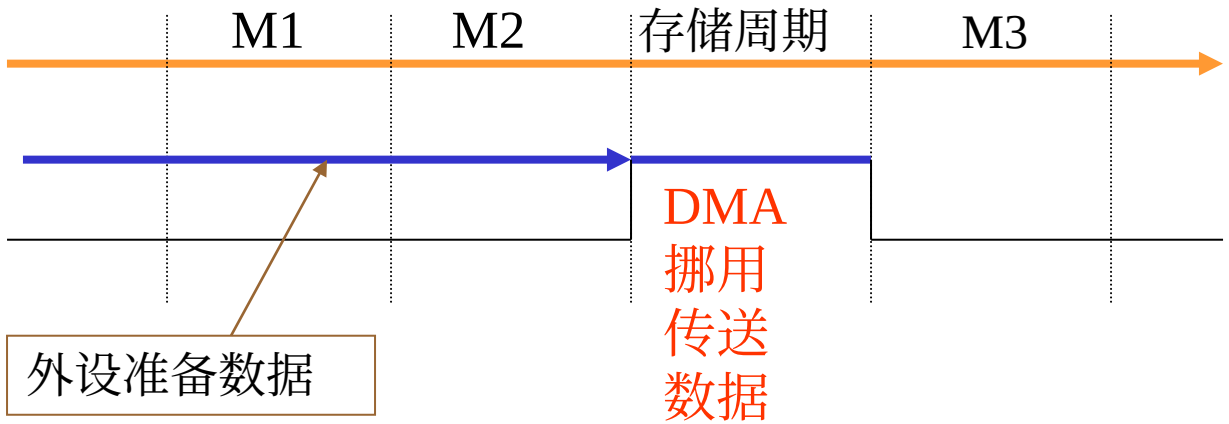
DMA与CPU交替使用主存

- 每个存储周期分成两段
  - 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

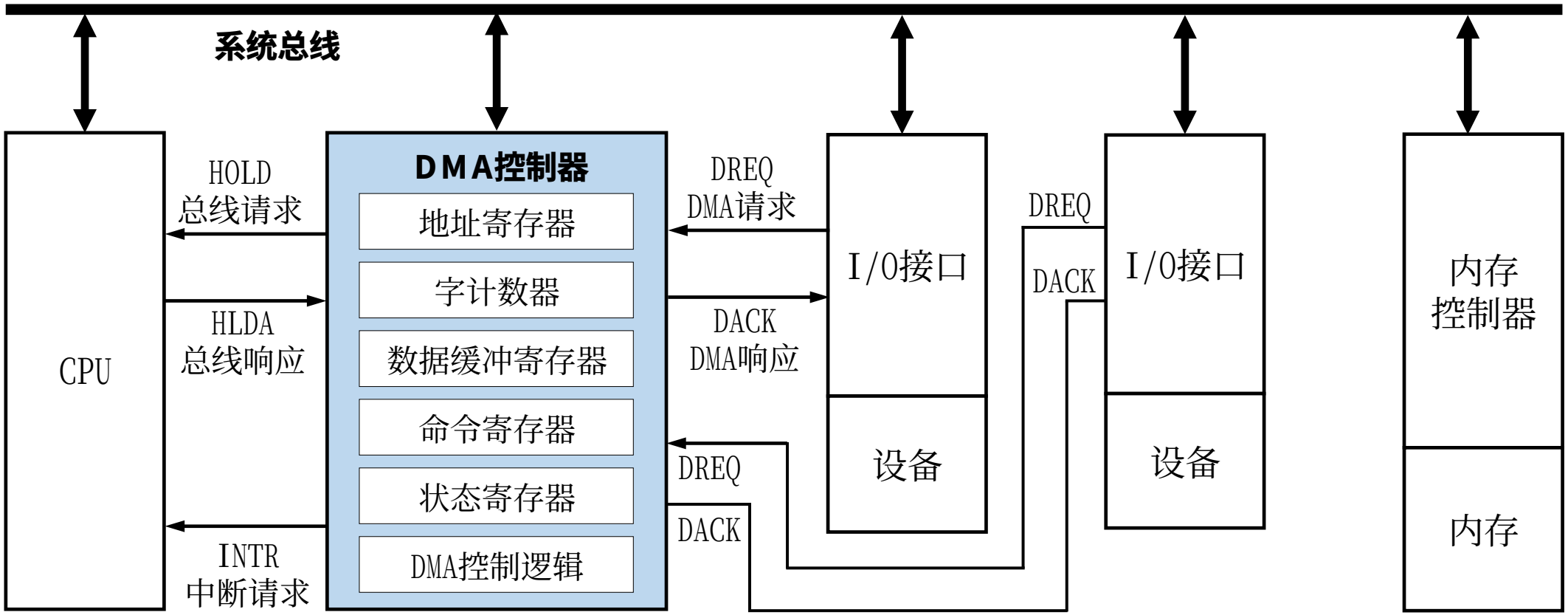


周期挪用法

- 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。

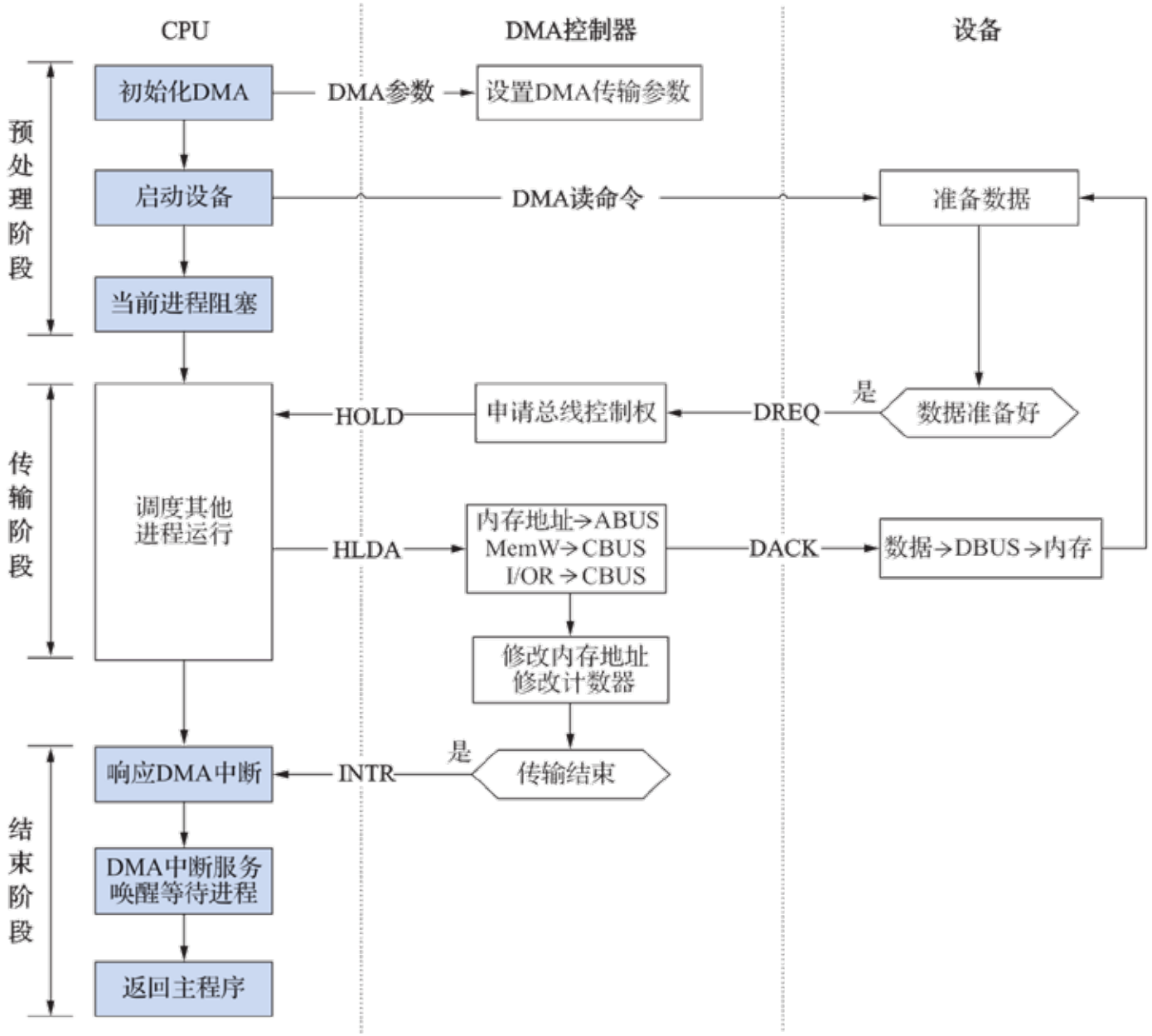


DMA控制器


系统总线
CPU
HOLD
总线请求
HLDA
总线响应
INTR
中断请求
DMA控制器
地址寄存器
字计数器
数据缓冲寄存器
命令寄存器
状态寄存器
DMA控制逻辑
DREQ
DMA请求
DACK
DMA响应
DREQ
DACK
I/O接口
设备
DREQ
DACK
I/O接口
设备
内存
控制器
内存

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 Of Process）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

- 例 9.5　某磁盘采用 DMA 方式与 CPU 交换信息，其传输速率为 20MB/s。若 DMA 的预处理阶段需要 200 个时钟周期，DMA 完成传输后的中断处理阶段需要 400 个时钟周期。如果DMA 平均传输的数据块长度为 512B，问：磁盘工作时，200MHz 的处理器进行 DMA 传输时的CPU 占用率是多少？如果采用 4KB 的传输单位呢？（不考虑 DMAC 与 CPU 争用内存对 CPU 性能的影响

  每秒需要执行的 DMA 次数为 20MB/512B = 39062.5 次。

  每次 DMA 传输所需要的 CPU 时间为 (200+400)T = 600T。

  则 DMA 传输所占用 CPU 时间的比例为 600T×39062.5/1s ≈ 11.72%。

  如果采用 4KB 的大块传输，则传输次数降低为原来的 1/8。

  DMA 操作占用 CPU 时间的比例为 600T×(20MB/4KB)/1s ≈ 1.46%。

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）×CPI×T=20×5/500MHz

传输32bit，需要的总时间T_{total}=32/8/0.5MB/s

CPU用于外设I/O时间占整个CPU时间比例= $T_{I/O}/T_{total}$=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）

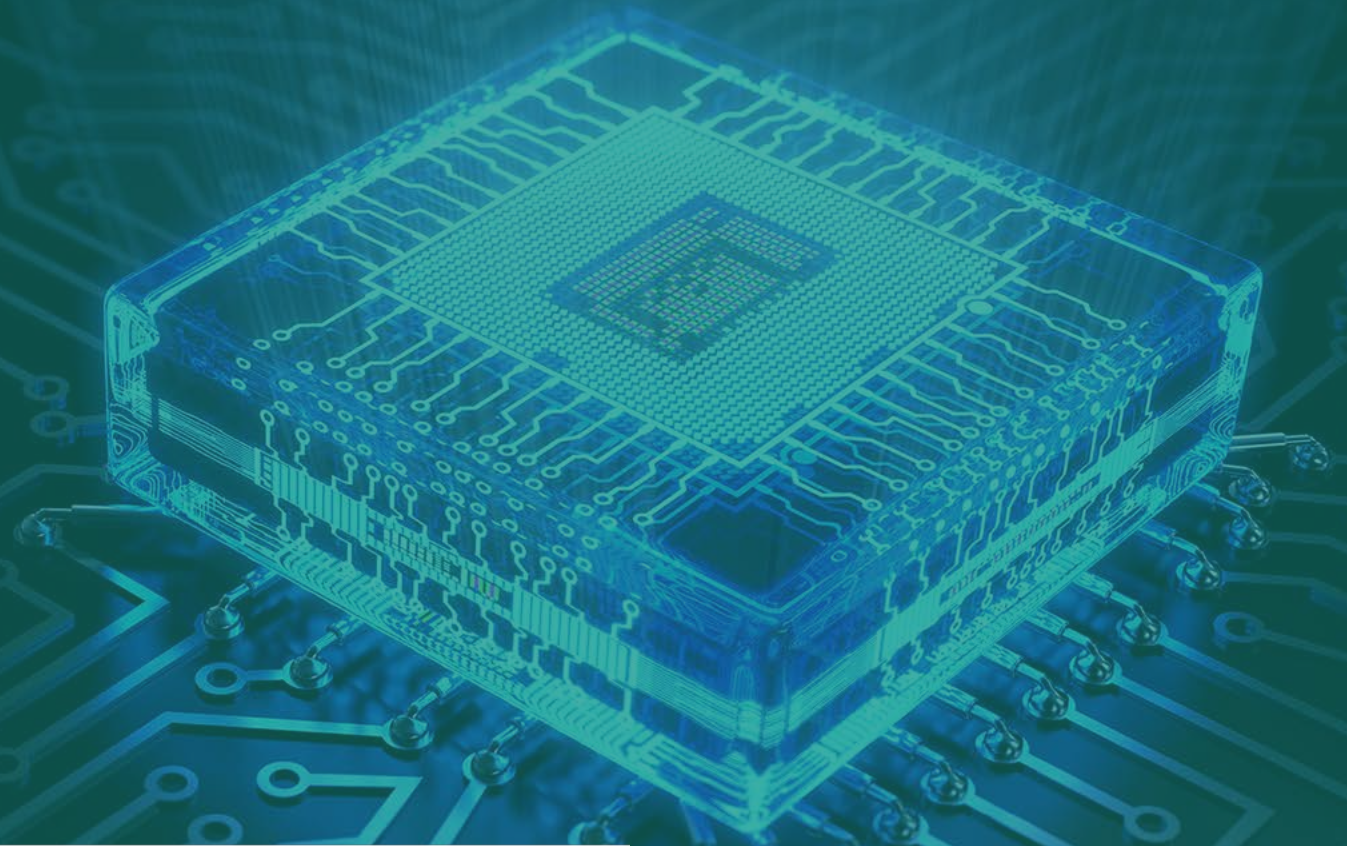
DMA传输阶段不需要占用CPU时间。

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

传输5000B　需要的总时间$T_{total}=5000/5MB/s$

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

華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST



THANKS

计算机组成原理