南京農業大學
NANJING AGRICULTURAL UNIVERSITY

计算机组成与系统结构

主讲人：陆明洲

人工智能学院计算机系

南京農業大學
NANJING AGRICULTURAL UNIVERSITY

08

CHAPTER

系统互联及输入输出组织

目录

CONTENT

外设与CPU和主存的互联

- 总线概述
- 基于总线的互连结构
- I/O接口
  - 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

总线的基本概念

- 将多个部件连接到一组公共信息传输线上，这种方式称为总线连接
  - 优点：灵活、成本低
  - 分类：
    - 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
      - 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。

总线的基本概念

■总线裁决

早期：总线多是共享传输，需确定哪个设备使用总线。

现在：总线多是点对点传输，无需裁决。

■总线定时

定义总线事务中的每一步何时开始、何时结束。

Synchronous (同步)：用时钟信号来确定每个步骤

Asynchronous(异步)：用握手信号来定时，前一个信号结束就是下一个信号的开始

半同步：结合使用时钟信号和握手信号来定时

- 并行/串行传输

  并行传输：一个方向同时传输多位数据信号，故位与位需同步，慢！

  串行传输：一个方向只传输一位数据信号，无需在位之间同步，快！

现在总线设计的趋势是：点对点、异步、串行

总线性能指标

- 总线宽度
  - 总线中数据线的条数，决定了每次能同时传输的信息位数。
- 总线工作频率
  - 早期的总线通常一个时钟周期传送一次数据，此时，工作频率等于总线时钟频率；现在有些总线一个时钟周期可以传送2次或4次数据，因此，工作频率是时钟频率的2倍或4倍。
- 总线带宽
  - 总线的最大数据传输率
  - 对于同步总线，总线带宽计算公式： $B=W\times F/N$
    - W-总线宽度；F-总线时钟频率；N-完成一次数据传送所用时钟周期数。
    - F/N实际上就是总线工作频率
- 总线传送方式
  - 非突发传送：每个总线事务都传送地址，一个地址对应一次数据传送。
  - 突发传送：即为成块数据传送。突发传送总线事务中，先传送一个地址，后传送多次数据，后续数据的地址默认为前面地址自动增量。

总线带宽计算

例 某同步总线在一个时钟周期内传送一个4字节数据，总线时钟频率为33MHz，求总线带宽是多少？如果总线宽度改为64位，一个时钟周期能传送2次数据，总线时钟频率为66MHz，则总线带宽为多少？提高了多少倍?

解:

B1=4B*33Mhz/1=132MB/s

B2=8B*2*66MHz/1=1056MB/s

总线带宽提高了7倍。

I/O总线

■I/O总线用于为系统中的各种I/O设备提供输入输出通道

■I/O总线在物理上可以是主板上的I/O扩展槽，如：

第一代：ISA/EISA总线、VESA总线，早被淘汰

第二代：PCI、AGP、PCI-X，被逐渐淘汰

第三代：PCI-Express（串行总线，主流总线）

■PCI-Express总线

两个PCI-Express设备之间以一个链路（link）相连，每个链路包含多条通路（lane），可以是1，2，4，8，16或32条；

PCI-Express×n表示一个具有n条通路的PCI-Express链路，每条通路可同时发送和接受，每个数据字节被转换为10位信息被传输；

PCI-Express1.0下，每条通路的发送和接受速率都是2.5Gb/s，故PCI-Express×n的带宽为：

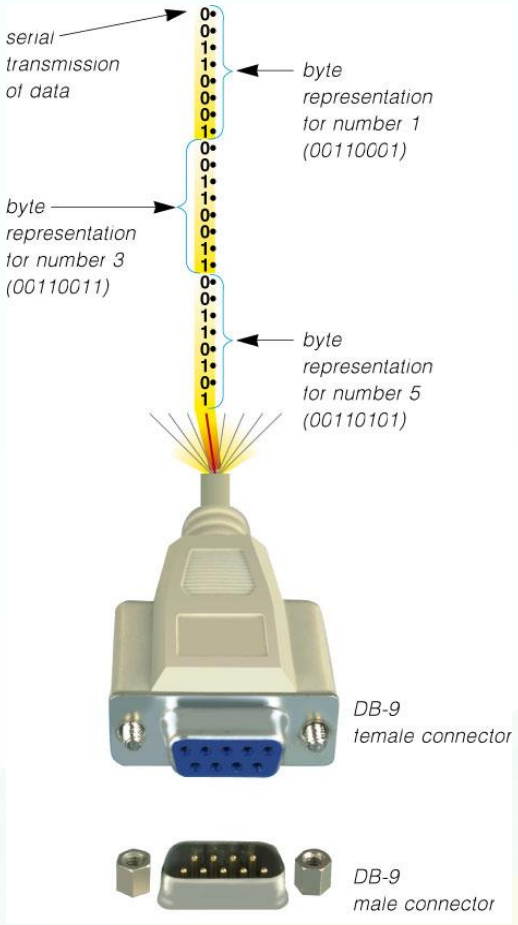
2.5Gb/s×2×n/10=0.5GB/s×n。

I/O接口

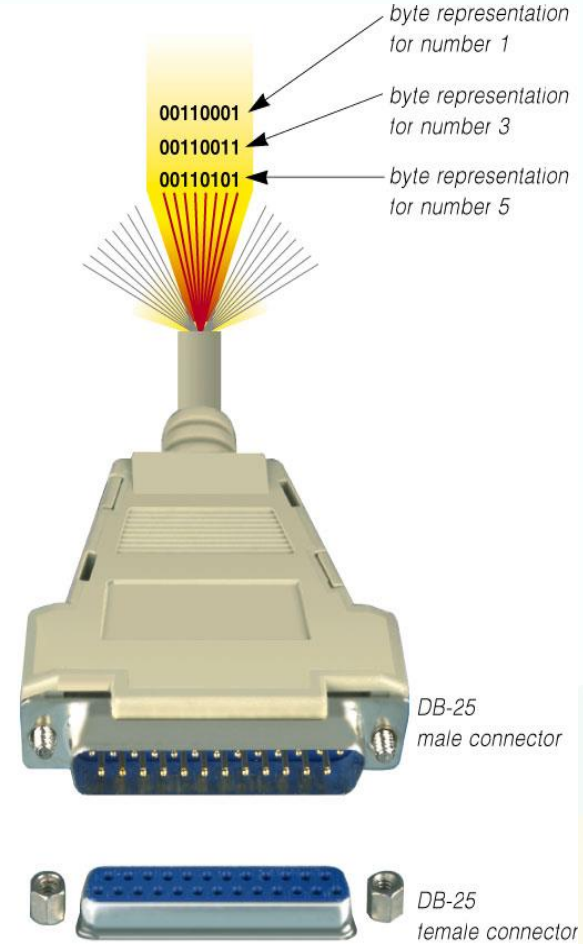
- I/O接口：I/O设备控制器及其插座（如网卡、显卡、键盘适配器、磁盘控制器）

  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



串行口



并行口

誠樸篤仁

I/O接口的职能

- **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**

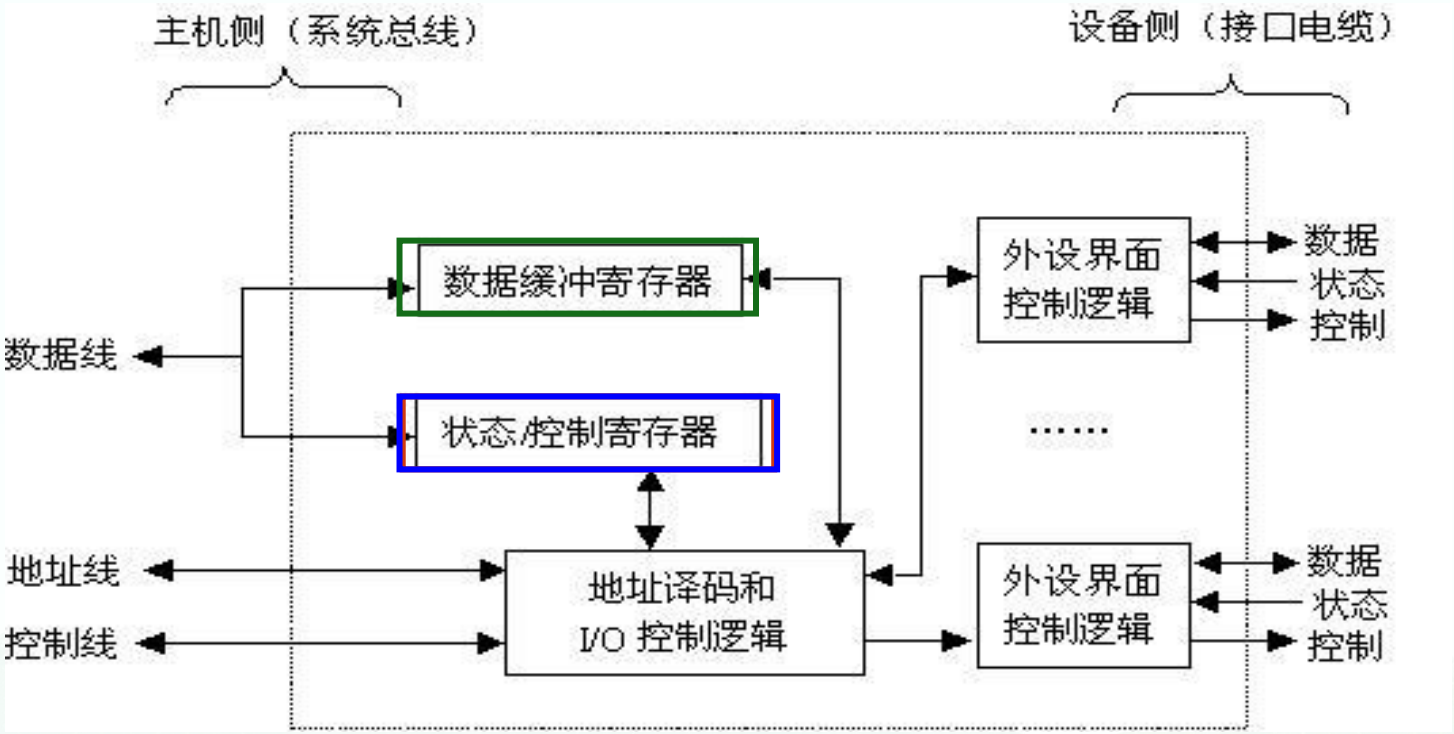
  提供数据格式转换部件使通过外部接口得到的数据转换为内部接口需要的格式，或在相反的方向进行数据格式转换。

- **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接口的通用结构

- 不同的I/O接口在复杂性和控制外设的数量上相差很大



对数据缓冲寄存器、控制/状态寄存器的访问操作是通过相应的指令完成的，通常把这类指令称为**I/O指令**

通过发送命令字到I/O控制寄存器来向设备发送命令，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

将I/O接口中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

I/O端口的寻址方式

- 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

（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。

例如，RISC机器、Motorola公司的处理器等采用该方案

（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

例如，Intel公司的处理器就是独立编址方式

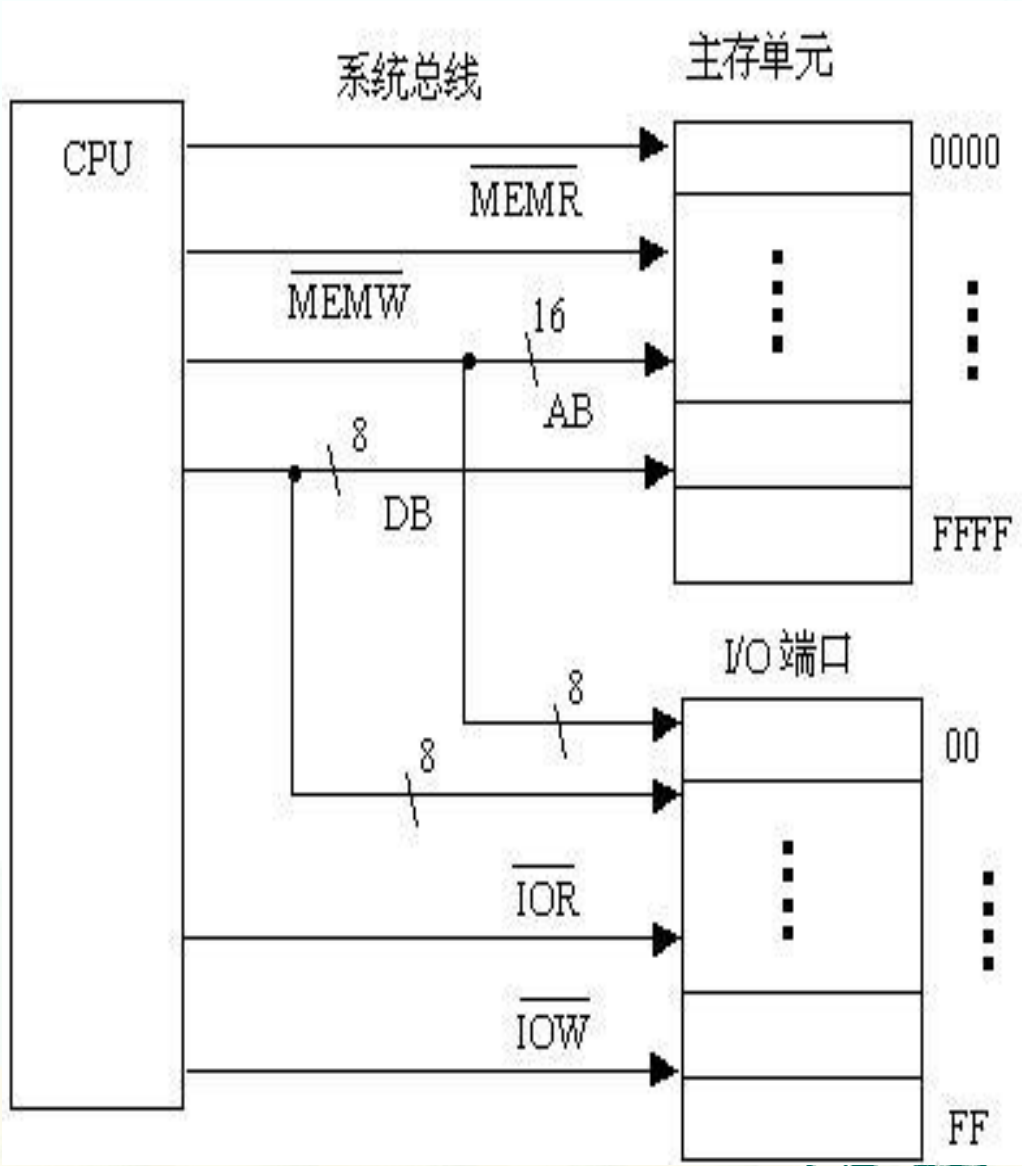
独立编址方式

- 指令系统需要有专门的I/O指令来访问I/O端口，在I/O指令的地址码部分给出I/O端口号
- IA-32提供了4条专门的IO指令：in、ins、out、outs

举例：以下两条指令可将AL寄存器中的字符数据送到打印机数据缓冲寄存器（端 口号为378H）

MOV DX，378H

OUT DX，AL

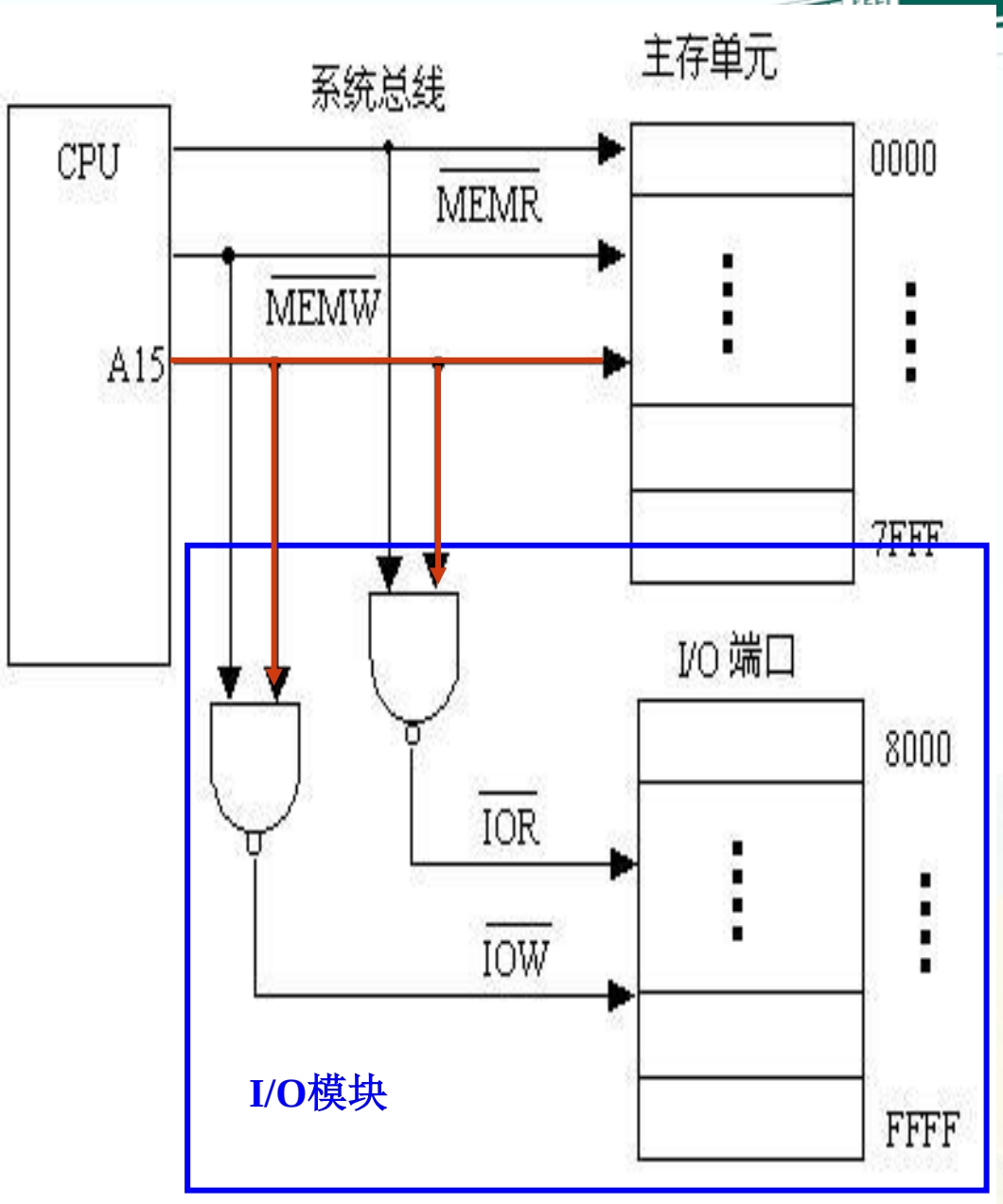


统一编址方式

- I/O端口和主存单元在同一个地址空间的不同分段中，根据地址范围就可以区分访问的是I/O端口还是主存单元
- 无需专门的IO指令，只要用一般的访存指令就可以存取I/O端口

举例：对某统一编址的I/O系统，以下两条指令可将IO端口（端口地址为B0C0 0010）中的信息（数据或状态）取到通用寄存器$t8中

```
lui $t9，0xb0c0
lw $t8，0x10($t9)
```



目录

CONTENT

I/O设备与主机进行数据交换的三种基本方式

- 程序直接控制方式（最简单的I/O方式）
  - 无条件传送：对简单外设定时（同步）进行数据传送
  - 条件传送：Polling (轮询，查询): 主动查询，也称为程序查询方式
    - I/O设备（包括I/O接口）将自己的状态放到一个状态寄存器中
    - 阶段性地查询状态寄存器中的特定状态，以决定下一步动作
- I/O Interrupt (中断I/O方式): 几乎所有系统都支持的中断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，调用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
- 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

程序直接控制方式-无条件传送方式

- 也称为同步传送方式，用于对一些简单、**慢速**外设（如开关、继电器、发光二极管、7段数码管等）在规定的时间内用相应的I/O指令对**接口中的寄存器**进行信息输入或输出
- 处理器对外设I/O接口进行周期性的定时访问，直接对I/O端口进行数据存取

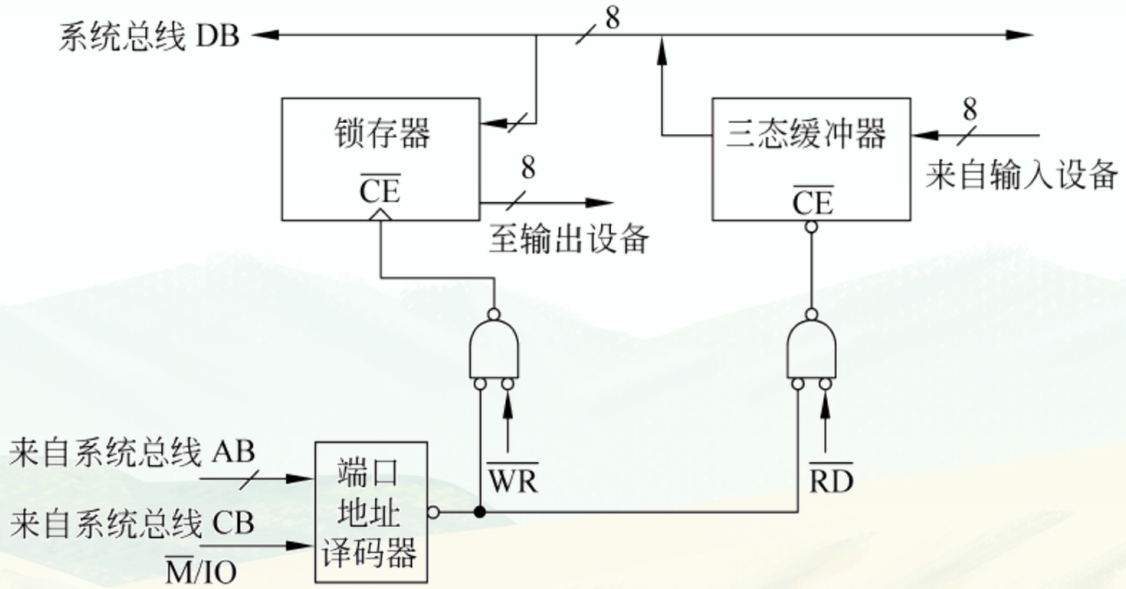


图 8.16　无条件传送接口

程序直接控制方式-条件传送方式

- 也称为异步传送方式，对设备的控制必须在一定的状态条件下才能进行
- 通过查询接口中的状态来控制数据传送的方式，也称为程序查询方式

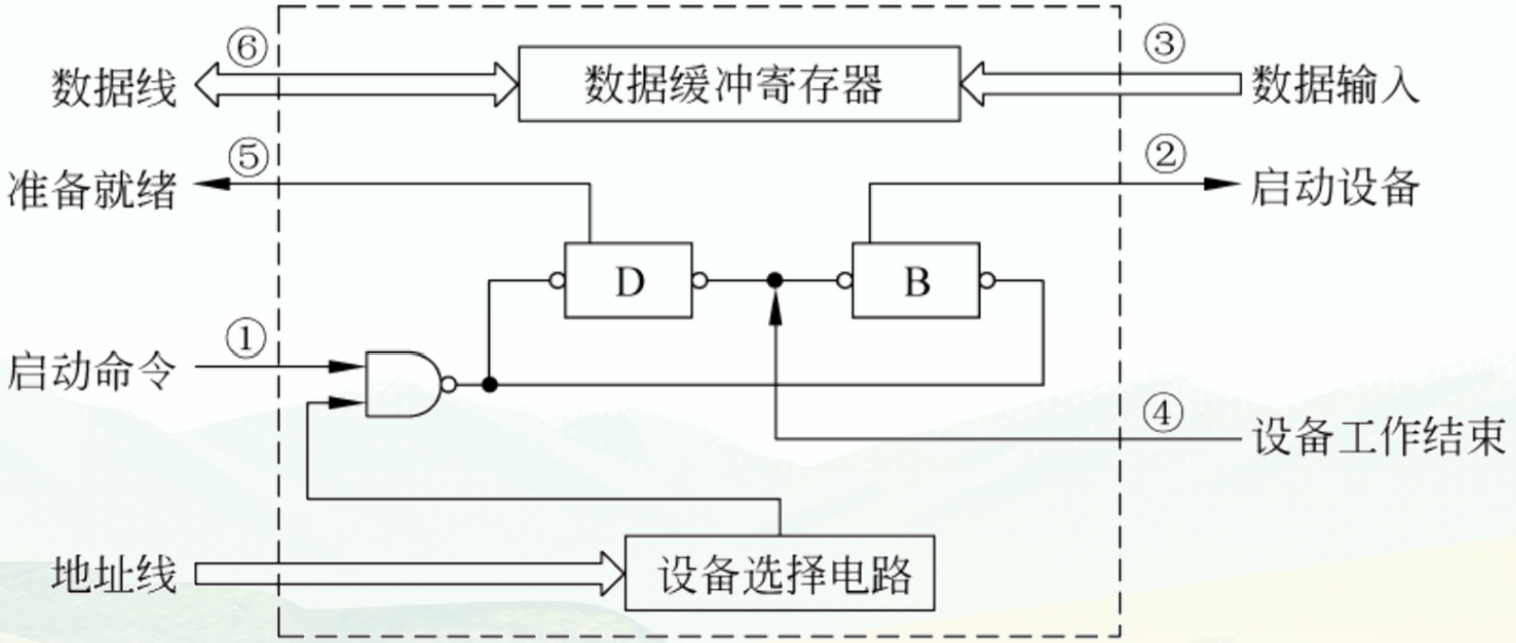
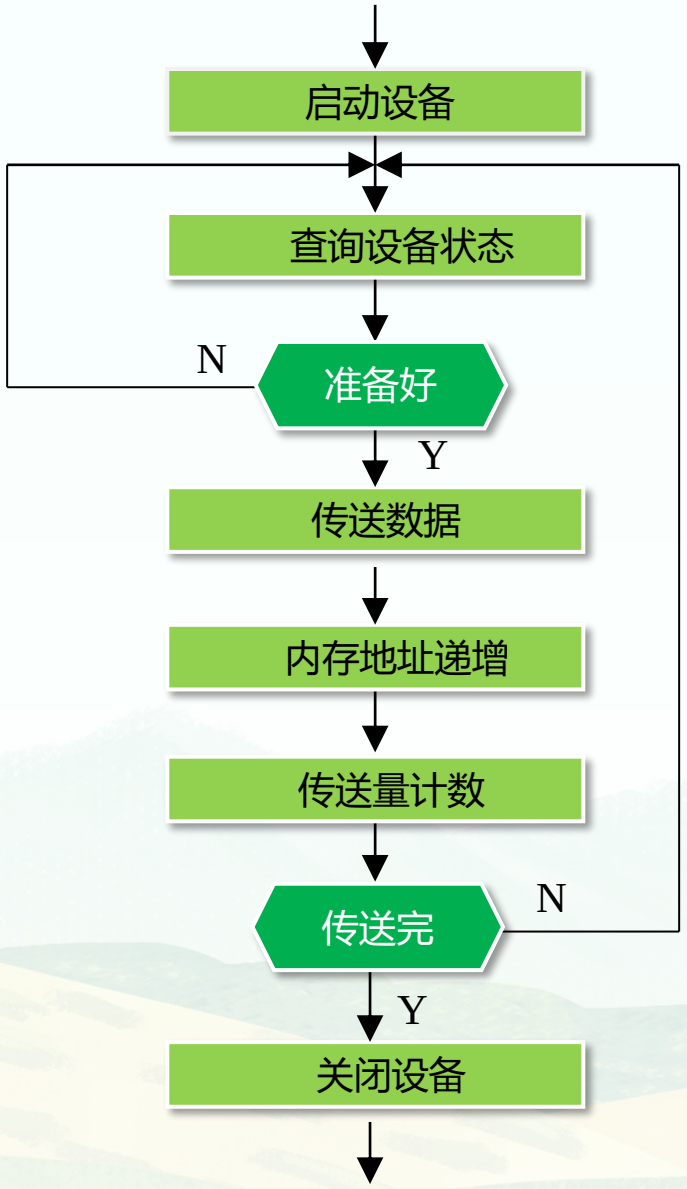


图 8.17　条件传送方式接口

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

例 假定查询程序中所有操作（包括读取并分析状态、传送数据以及重启用户程序等步骤）所用的时钟周期数至少是400个。处理器的主频为500MHz，即处理器每秒钟产生500×10^6个时钟周期。假定设备一直持续工作，采用定时查询方式，则下列3种情况下，CPU用于I/O的时间占整个CPU时间的百分比各是多少？

(1) 鼠标必须每秒钟至少被查询30次，才能保证不错过用户的任何一次移动。

(2) 软盘以16位为单位进行数据传送，数据传输率为50kBps，要求没有任何数据传送被错过。

(3) 硬盘以16字节为单位进行数据传送，数据传输率为4MBps，要求没有任何数据传送被错过。

解：

(1) CPU每秒对鼠标进行30次查询，所需要的时钟周期为：400 * 30 = 12000。

所以CPU此时应该要花费的时间比率为：12000 / (500 * 10^6) = 0.002%，即鼠标查询基本不影响CPU的性能。

程序查询方式

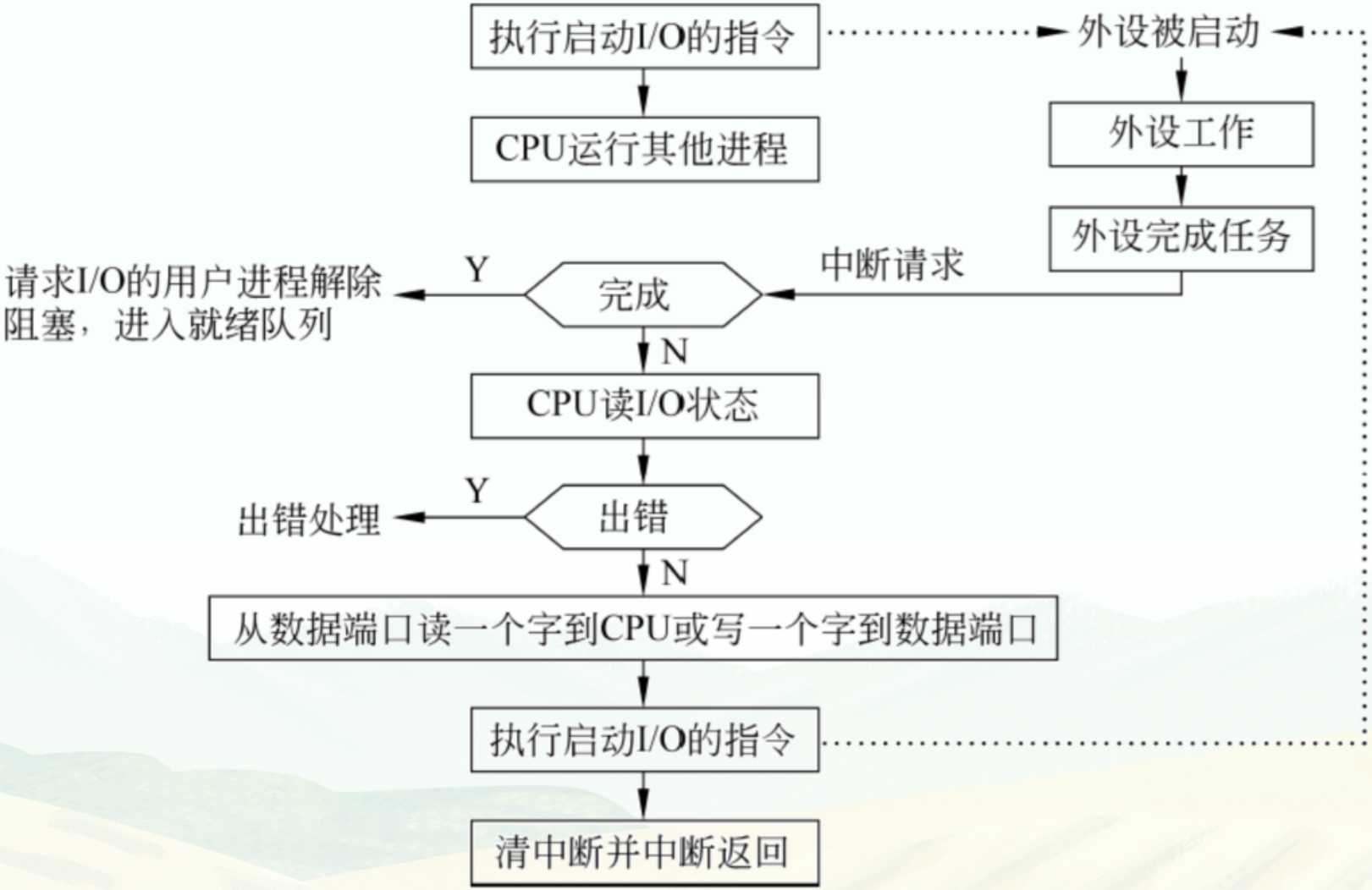
(2)对于软盘来说，每次数据传输单位为16位，只有当查询速率达到每秒50KB/ 2B = 25K次才能保证没有任何数据丢失。

所以每秒种内用于查询的时间周期为：25k * 400，在整个CPU时间中所占百分比为25k * 400/(500 * 10^6) = 2%。

(3)对于硬盘来说，每次数据传输单位为16B，只有当查询速率达到每秒4MB/ 16B = 250K次才能保证没有任何数据丢失。

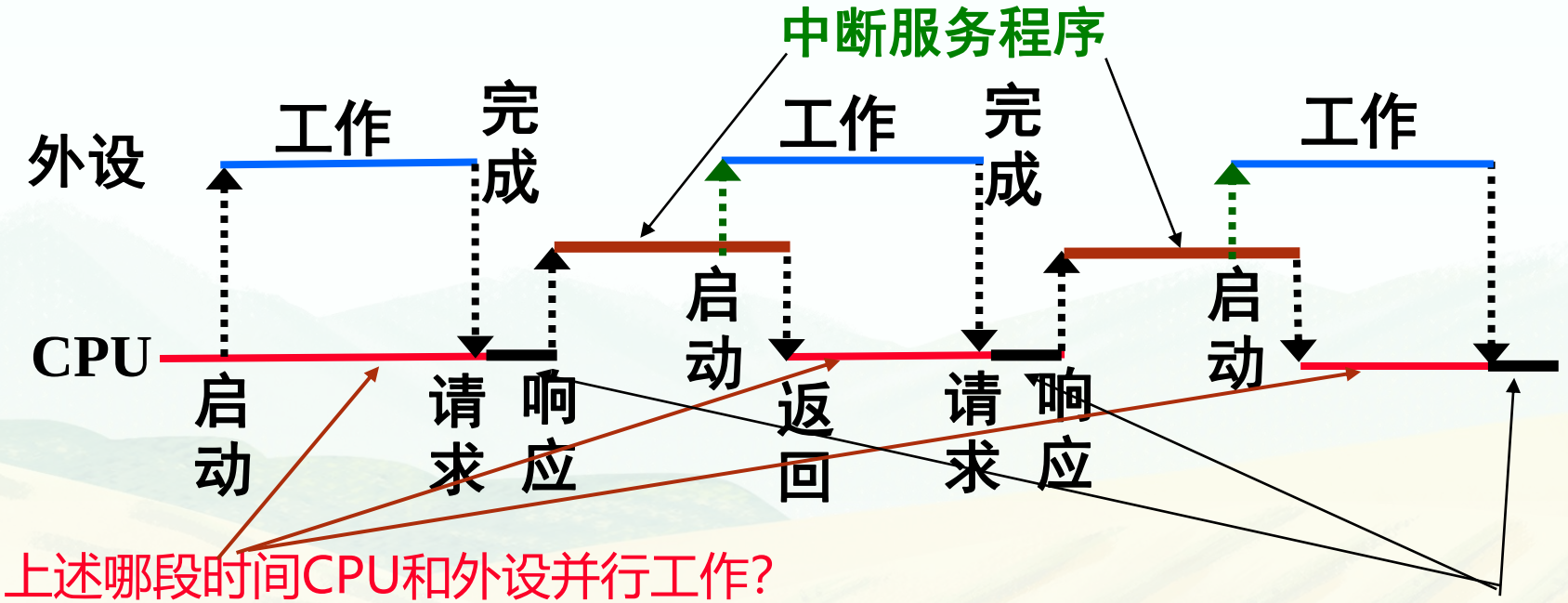
所以每秒种内用于查询的时间周期为：250k * 400，在整个CPU时间中所占百分比为250k * 400/(500 * 10^6) = 20%，即CPU的五分之一的时间用于查询硬盘，对硬盘用查询方式是不可取的。

中断控制I/O的过程


执行启动I/O的指令
CPU运行其他进程
外设被启动
外设工作
外设完成任务
中断请求
请求I/O的用户进程解除
阻塞，进入就绪队列
Y
完成
N
CPU读I/O状态
出错处理
Y
出错
N
从数据端口读一个字到CPU或写一个字到数据端口
执行启动I/O的指令
清中断并中断返回

中断基本概念

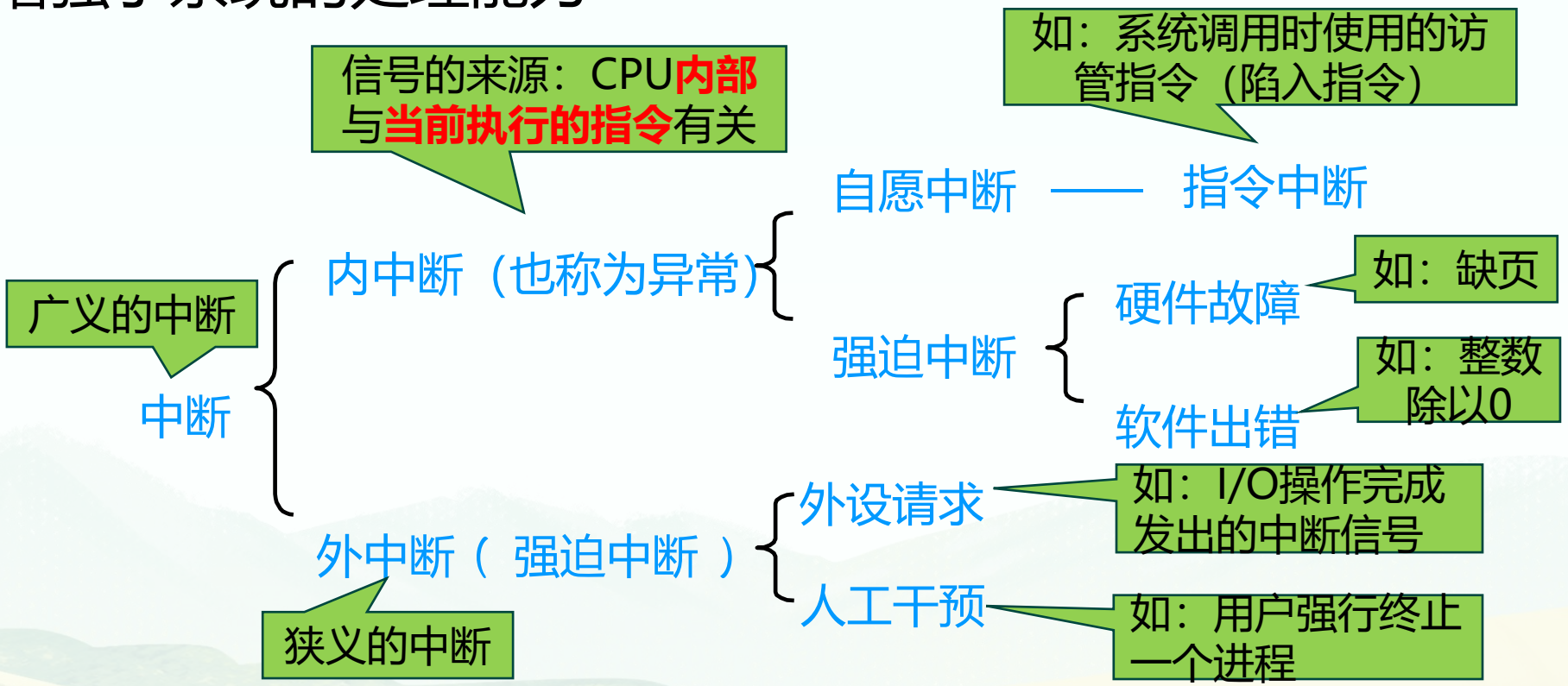
- 当外设准备好时，便向CPU发中断请求，CPU响应后，中止现行程序的执行，转入一个“中断服务程序”进行输入/出操作，实现主机和外设接口之间的数据传送，并启动外设工作。“中断服务程序”执行完后，返回原被中止的程序断点处继续执行。此时，外设和CPU并行工作。实现主机和外设准备阶段的并行工作



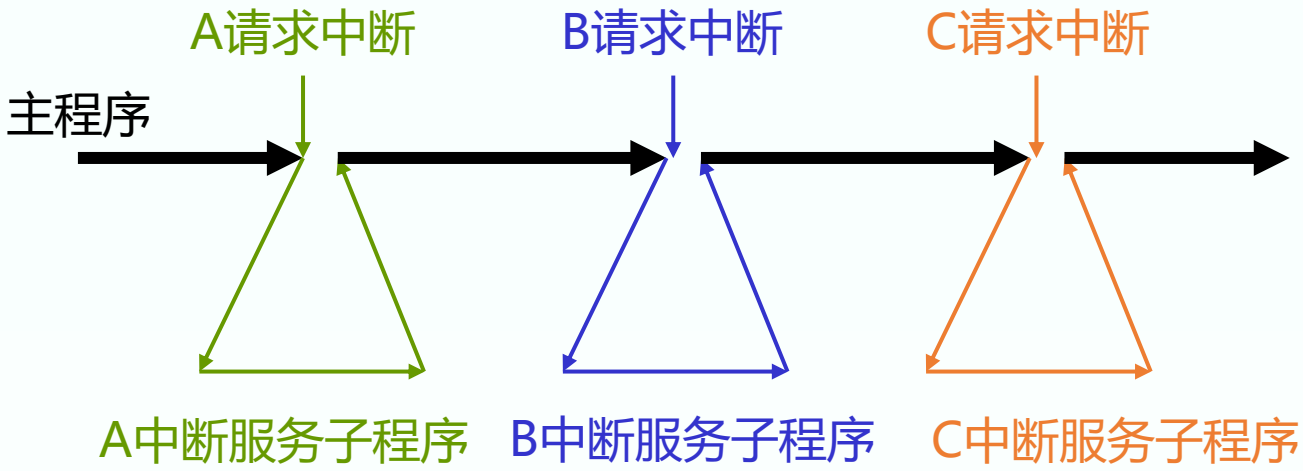
上述哪段时间CPU和外设并行工作?
程序切换（响应中断过程）由硬件完成，即执行“中断隐指令”，时间为

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



程序中断处理示意图



- 子程序与中断服务子程序与的区别?
  - 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

中断系统的基本职能与结构

① 及时记录各种中断请求信号，通常用一个中断请求寄存器来保持

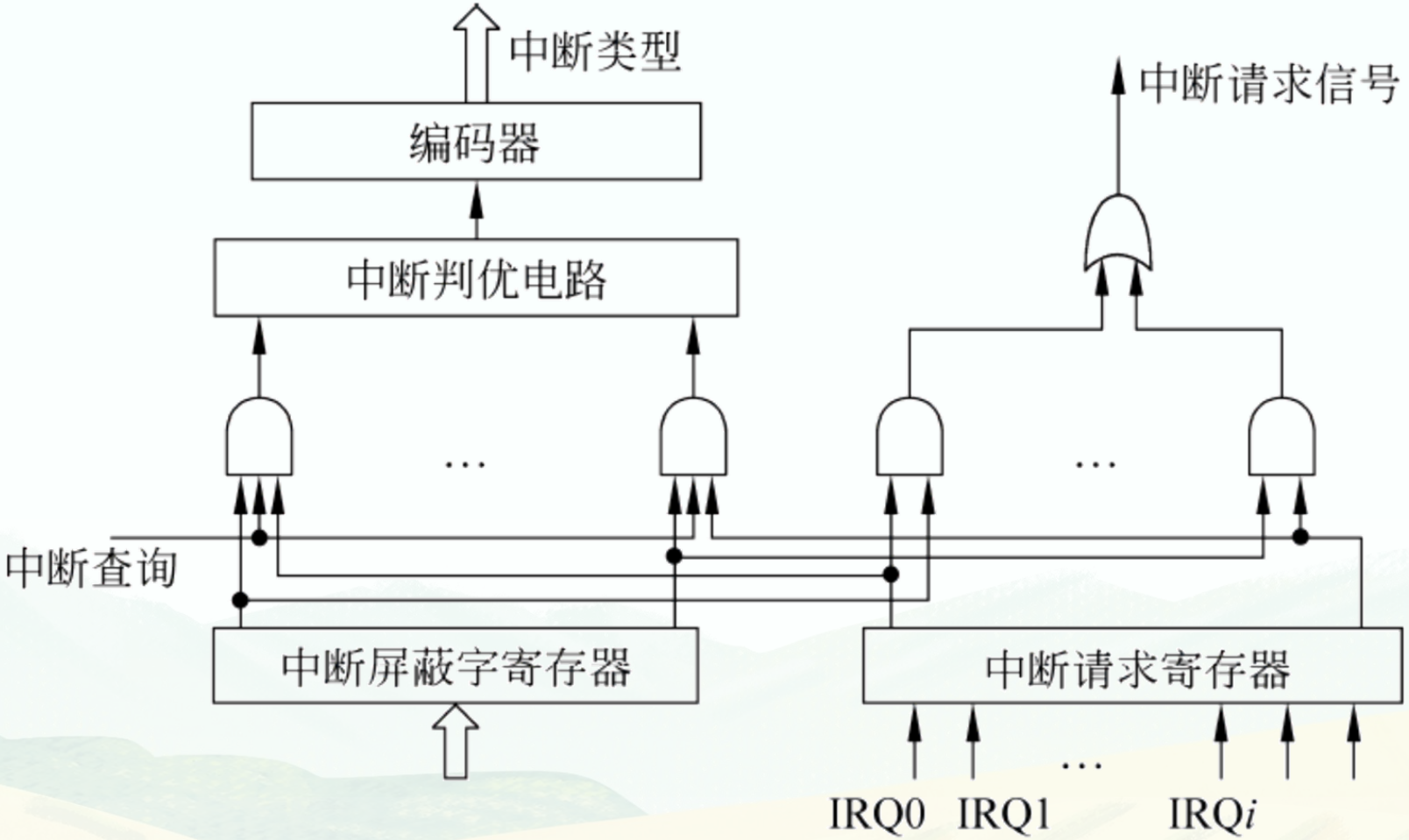
② 自动响应中断请求。CPU在每条指令执行完、下条指令取出前，会自动检测中断请求引脚，发现中断请求时，根据情况决定是否响应和响应那个中断请求

③ 自动判优。在有多个中断请求同时产生时，能够判断出哪个中断的优先级高，优先级搞的中断先被响应

④ 保护被中断程序的断点和现场

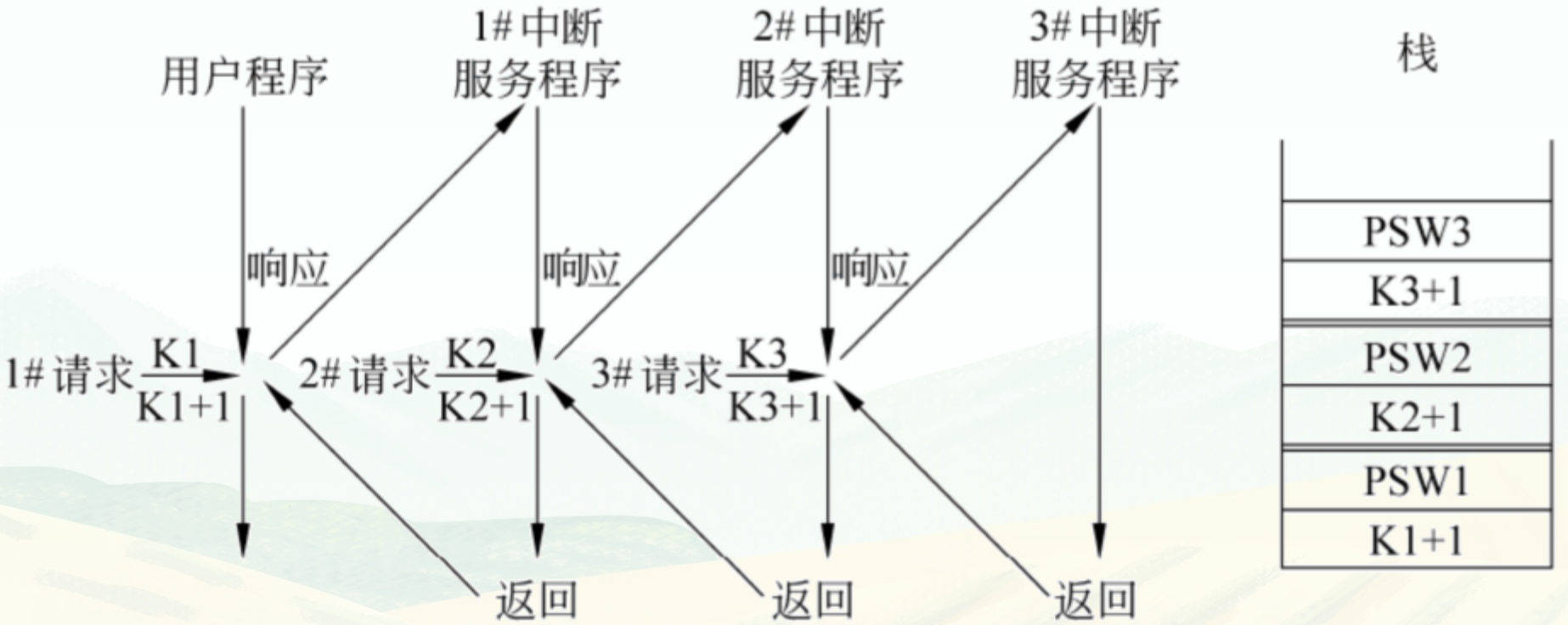
⑤ 中断屏蔽。通过中断屏蔽实现多重中断的嵌套执行，中断屏蔽功能通过一个中断屏蔽字寄存器来实现

中断系统中的中断控制器



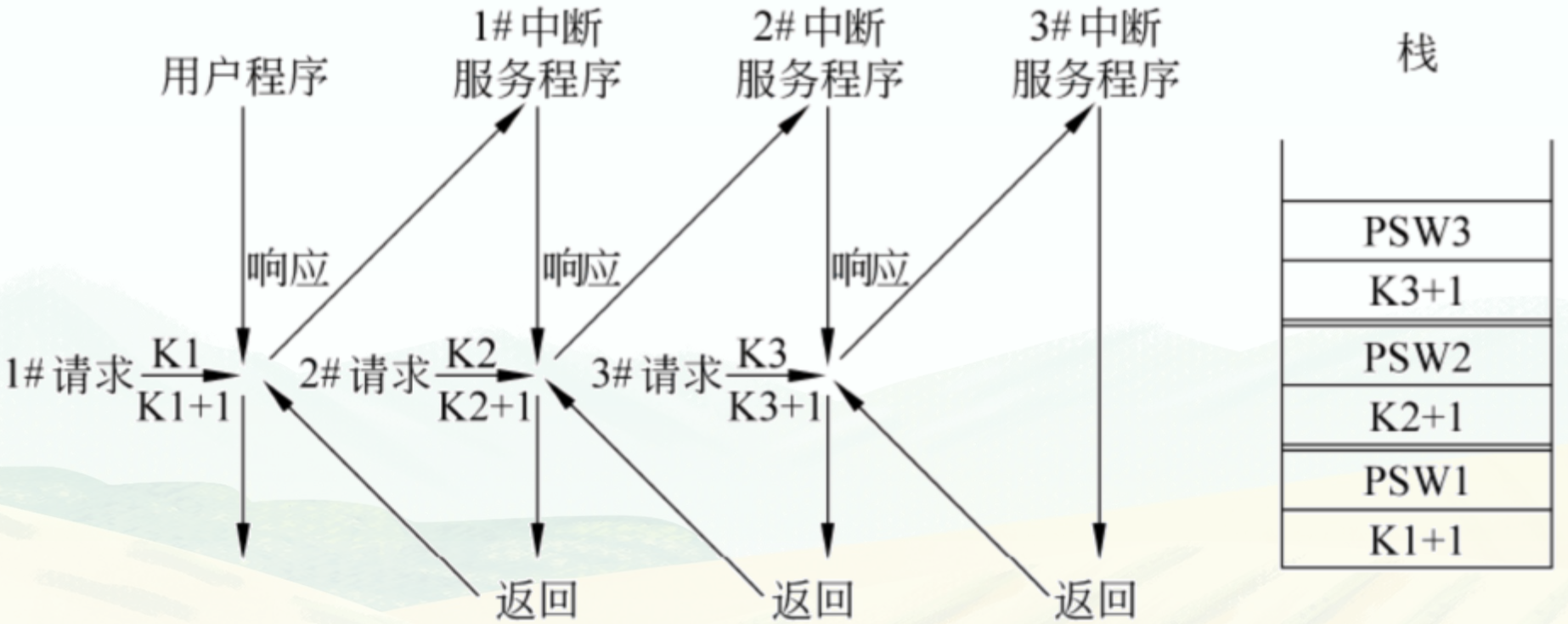
中断的嵌套

- 在中断处理（执行中断服务程序）过程中，若又有新的优先级更高的中断请求发生，那么CPU应立即体质正在执行的中断服务子程序，转去处理新的中断，这种情况被称为多重中断或中断嵌套。

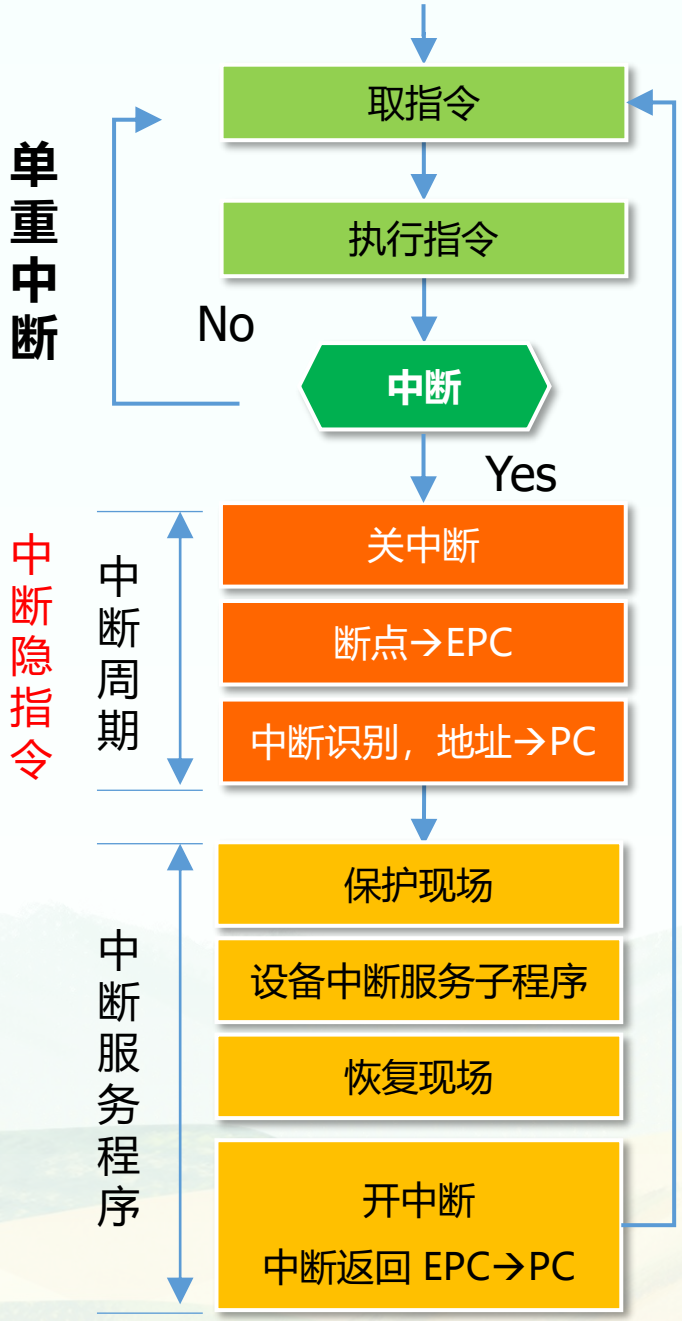


中断过程

- 在中断处理（执行中断服务程序）过程中，若又有新的优先级更高的中断请求发生，那么CPU应立即体质正在执行的中断服务子程序，转去处理新的中断，这种情况被称为多重中断或中断嵌套。



程序中断处理示意图

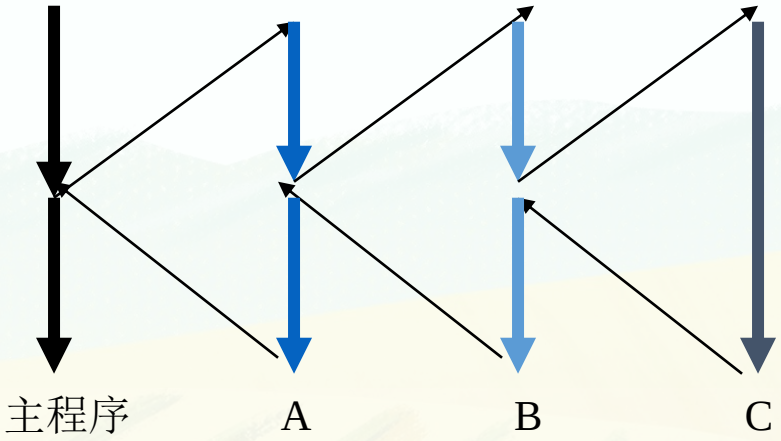

单重中断
取指令
执行指令
No
中断
Yes
中断隐指令
中断周期
关中断
断点→EPC
中断识别，地址→PC
中断服务程序
保护现场
设备中断服务子程序
恢复现场
开中断
中断返回 EPC→PC

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?

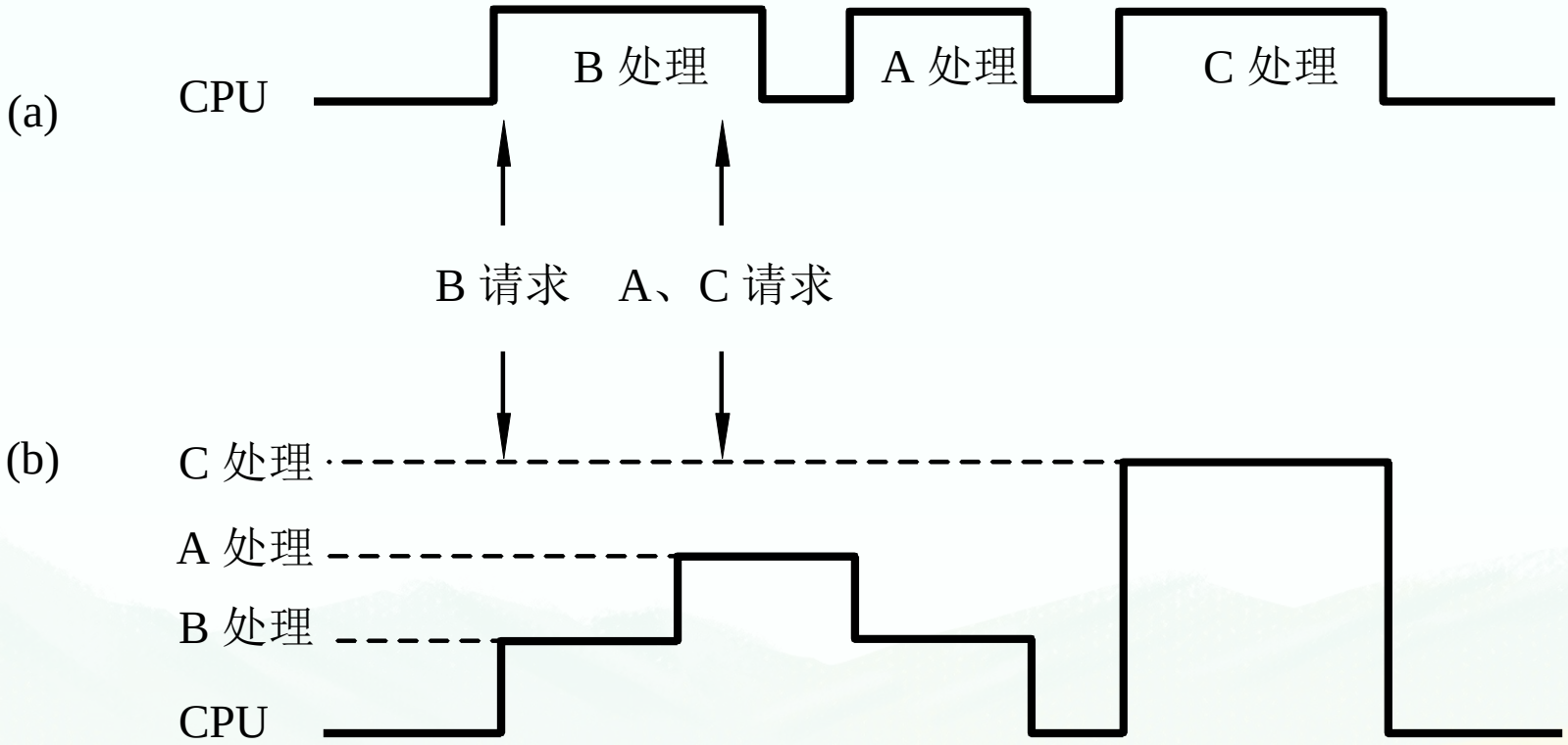
单级中断与多级中断

- 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

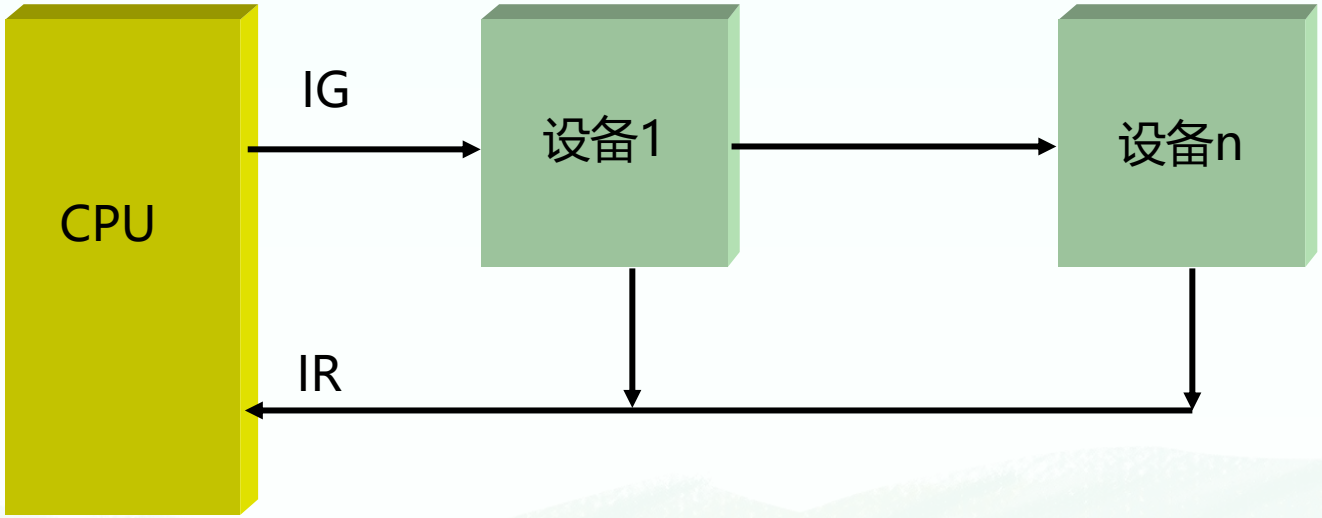


划分优先级的一般原则

- 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?
  - **链式查询**
  - 独立请求
  - 分组链式结构

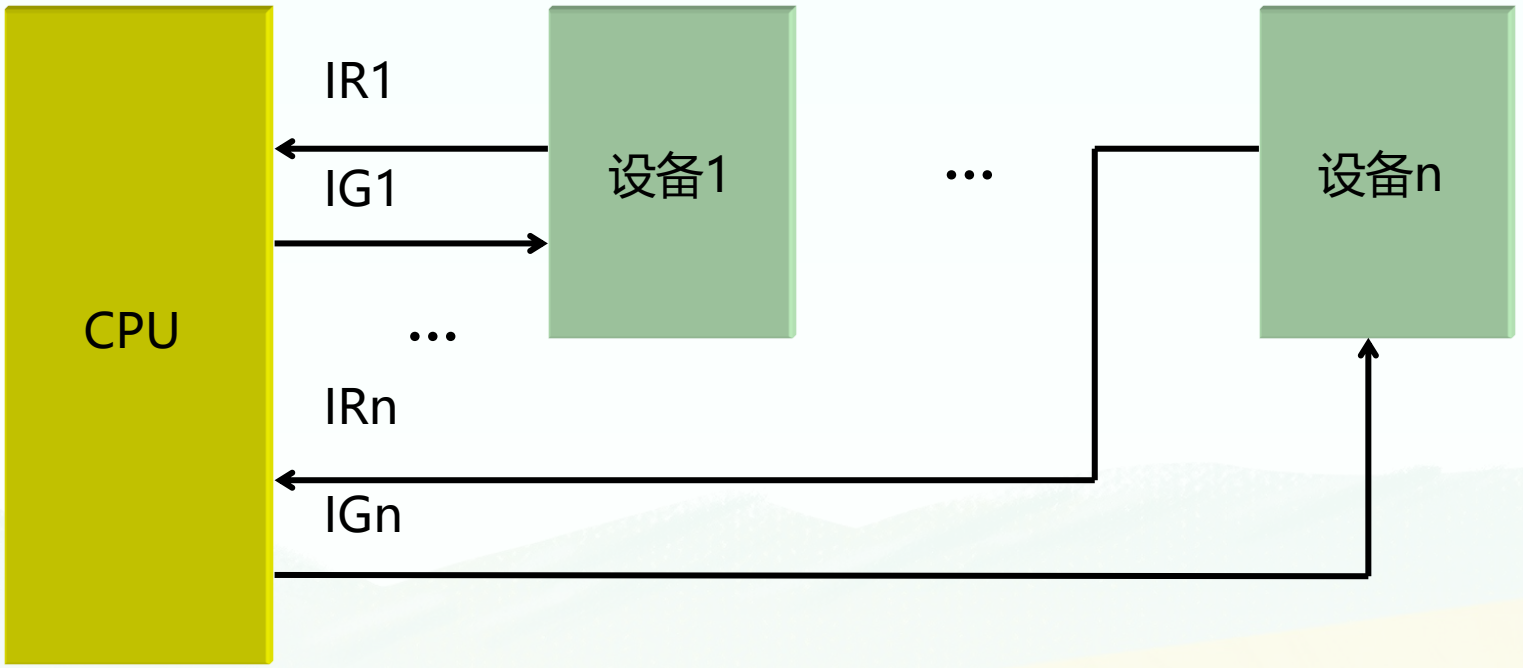


IR: 中断请求

IG: 中断许可

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？
  - 链式查询
  - **独立请求**
  - 分组链式结构

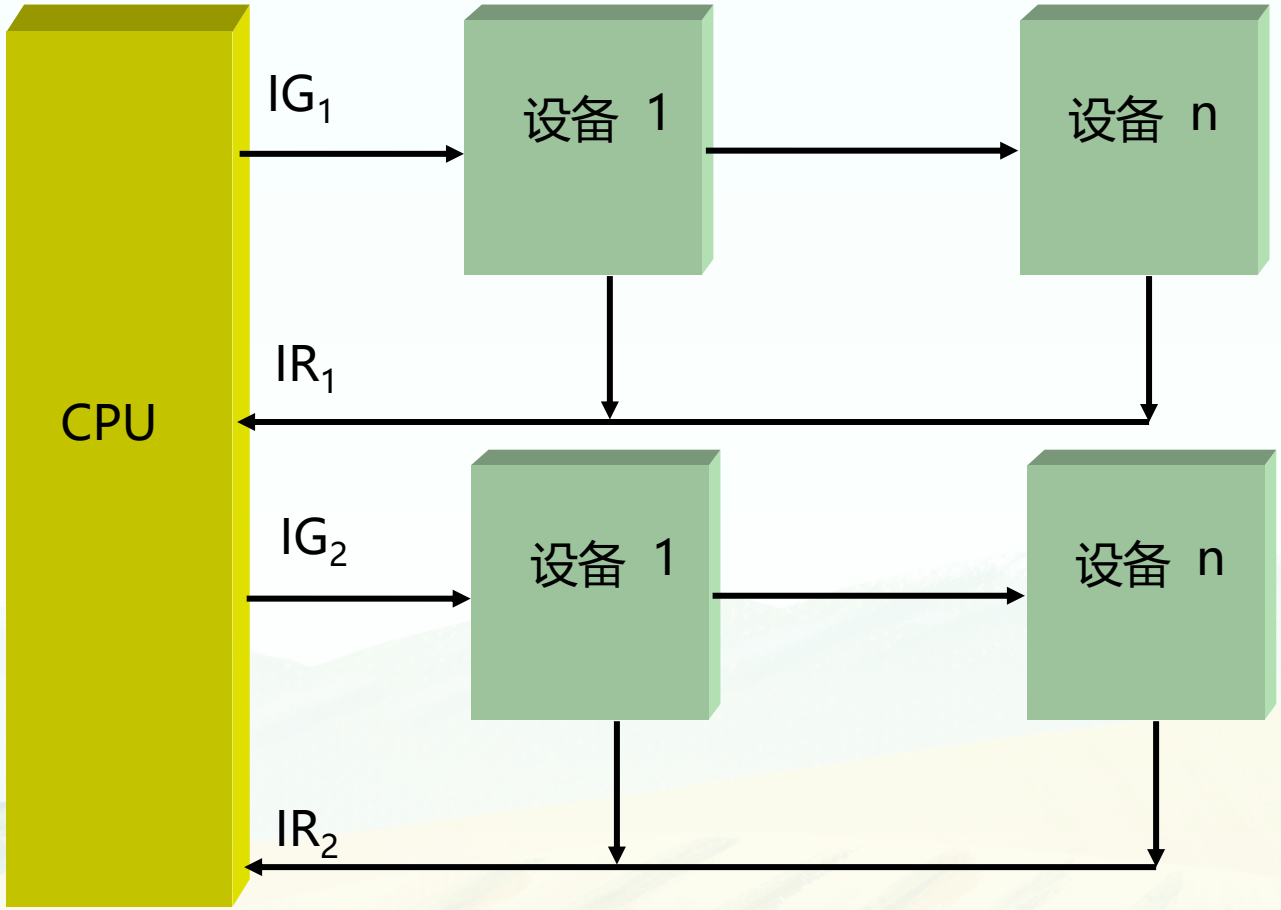


IR*x*：中断请求

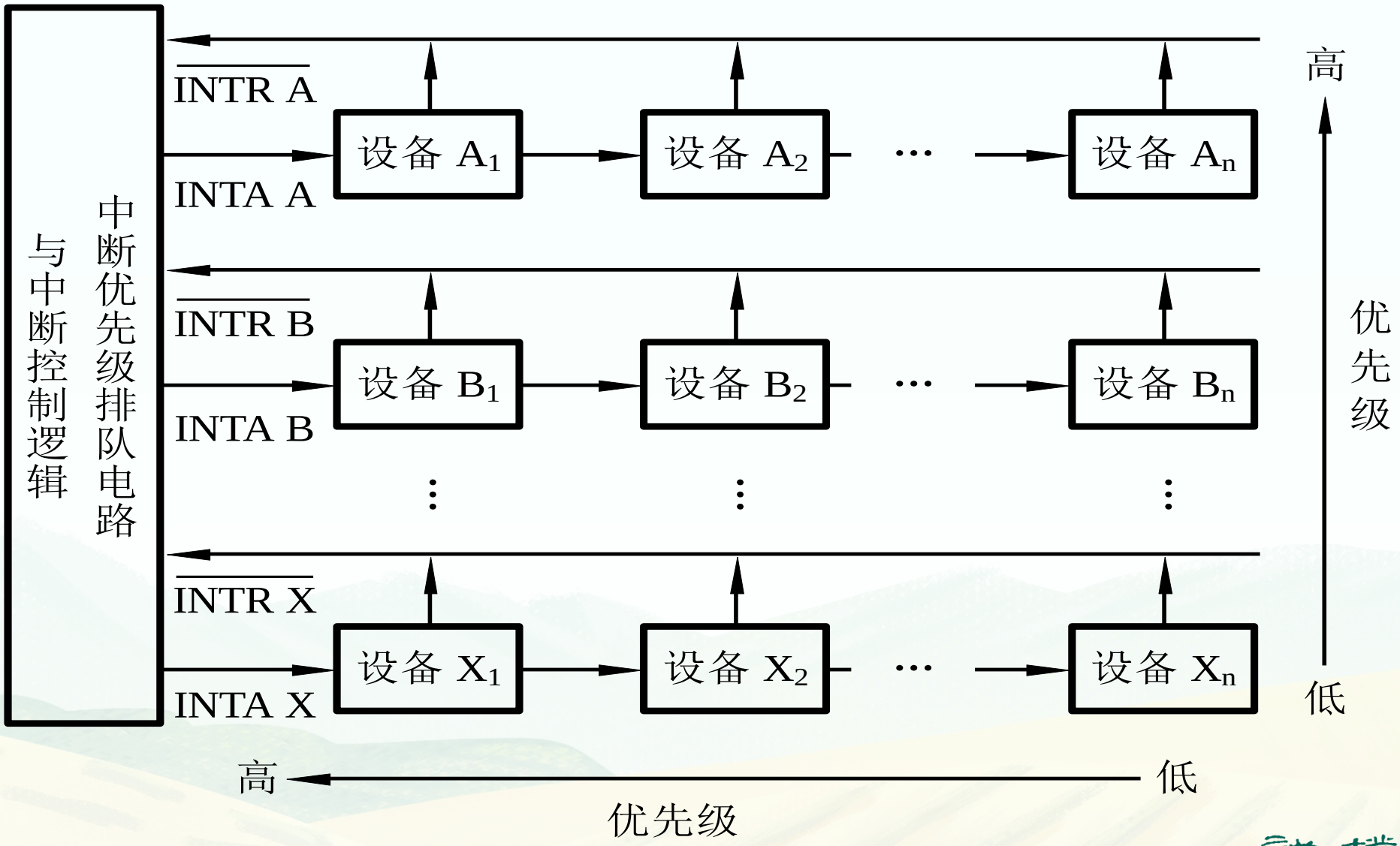
IG*x*：中断许可

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?
  - 链式查询
  - 独立请求
  - **分组链式结构**



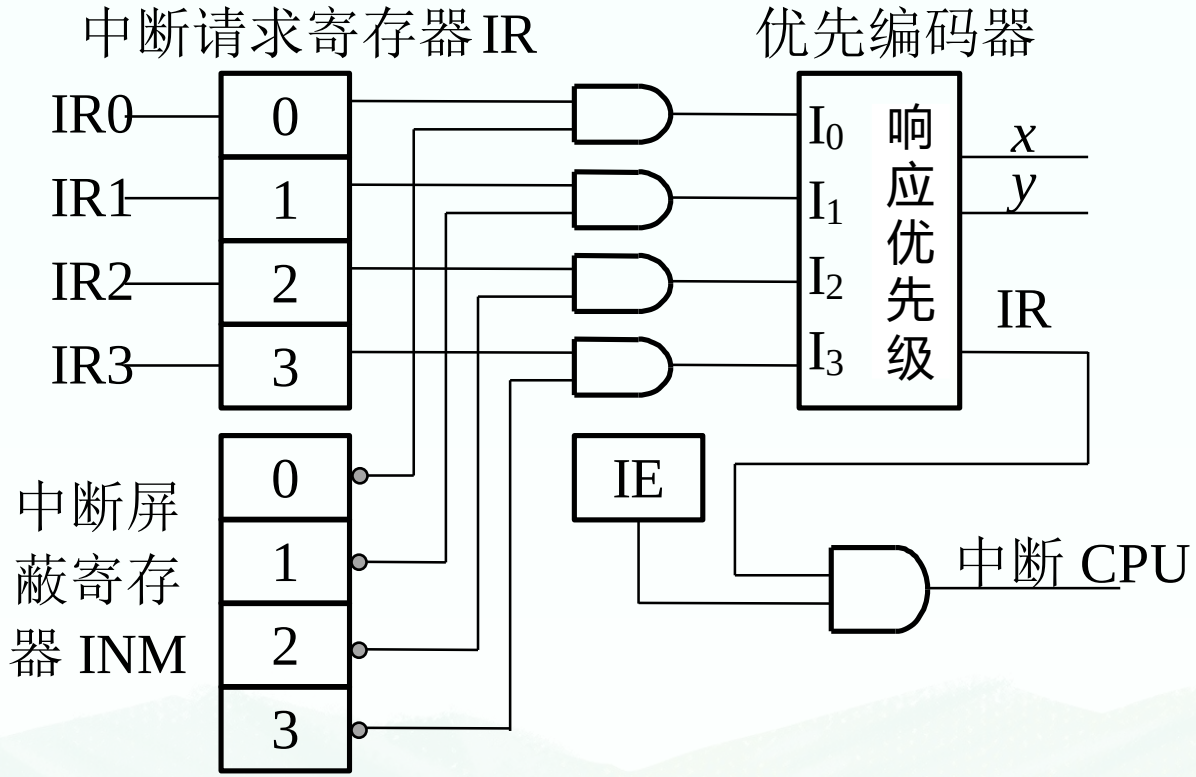
二维优先级示意图 (中断共享)



中断屏蔽

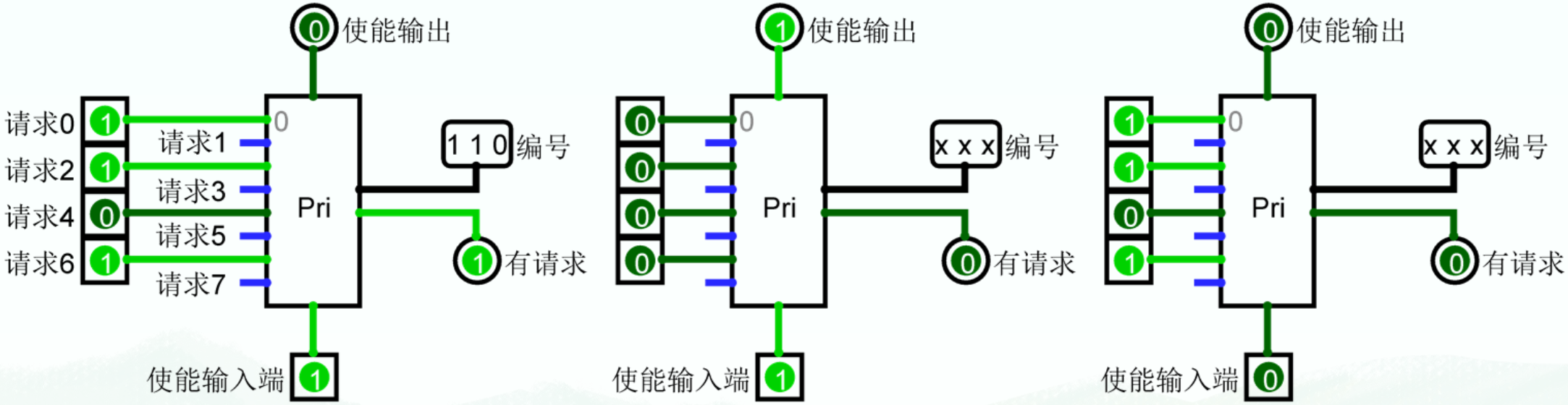
- **响应优先级**
  - CPU对各设备中断请求进行响应，并准备好处理的先后次序，这种次序往往在硬件线路上已固定，不便于变动。
- **处理优先级**
  - CPU实际对各中断请求处理的先后次序。如果不使用屏蔽技术，响应的优先次序就是处理的优先次序。
- **中断屏蔽**技术可**动态改变**各设备的处理优先级

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?

优先编码器

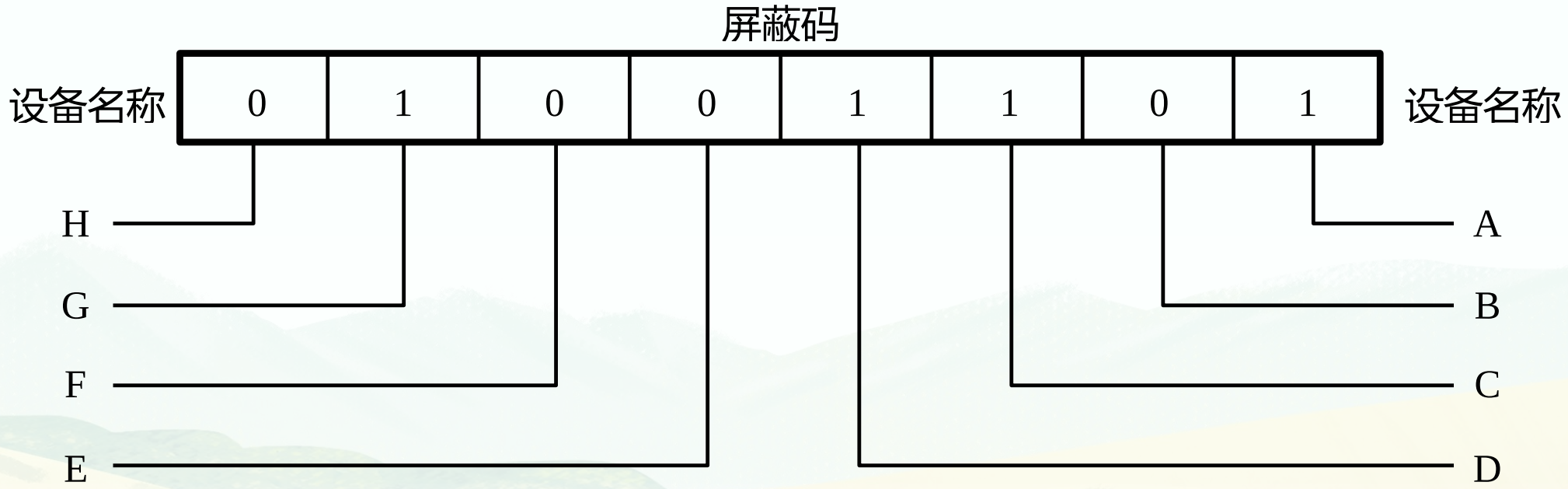

使能输出
0
请求0
请求1
请求2
请求3
请求4
请求5
请求6
请求7
1
1
0
1
1
1
0
Pri
1 1 0
编号
有请求
1
使能输入端
1
使能输出
1
0
0
0
0
0
Pri
x x x
编号
有请求
0
使能输入端
1
使能输出
0
1
1
0
1
0
Pri
x x x
编号
有请求
0
使能输入端
0

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



举例

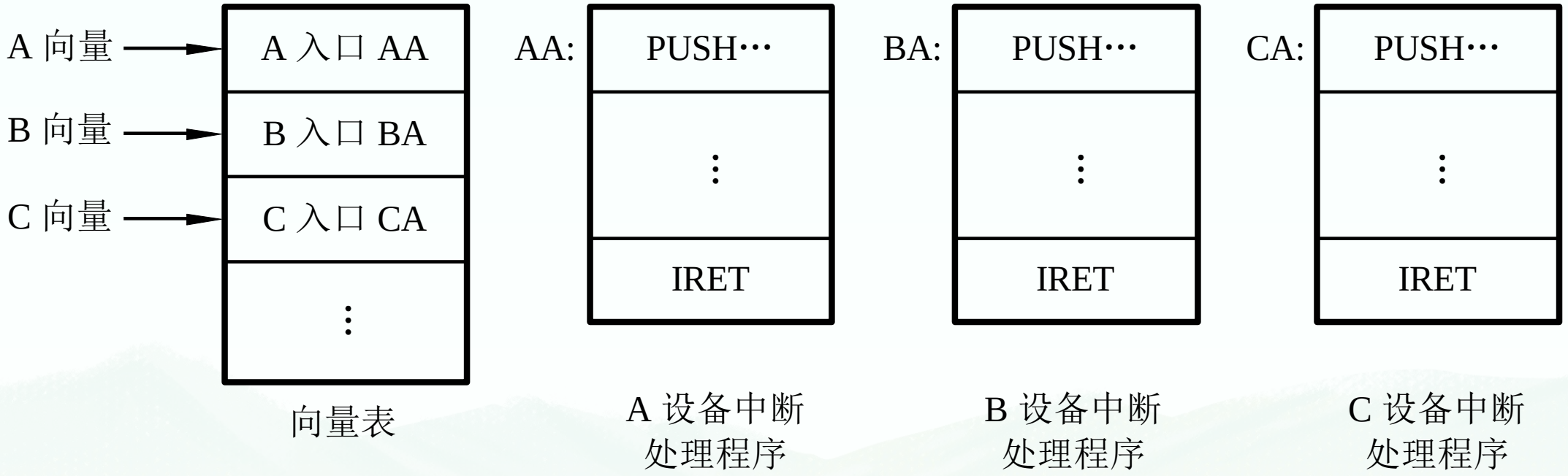
- 假定硬件原来的响应顺序为0→1→2，试设置中断屏蔽字，将中断优先级改为1→2→0。

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

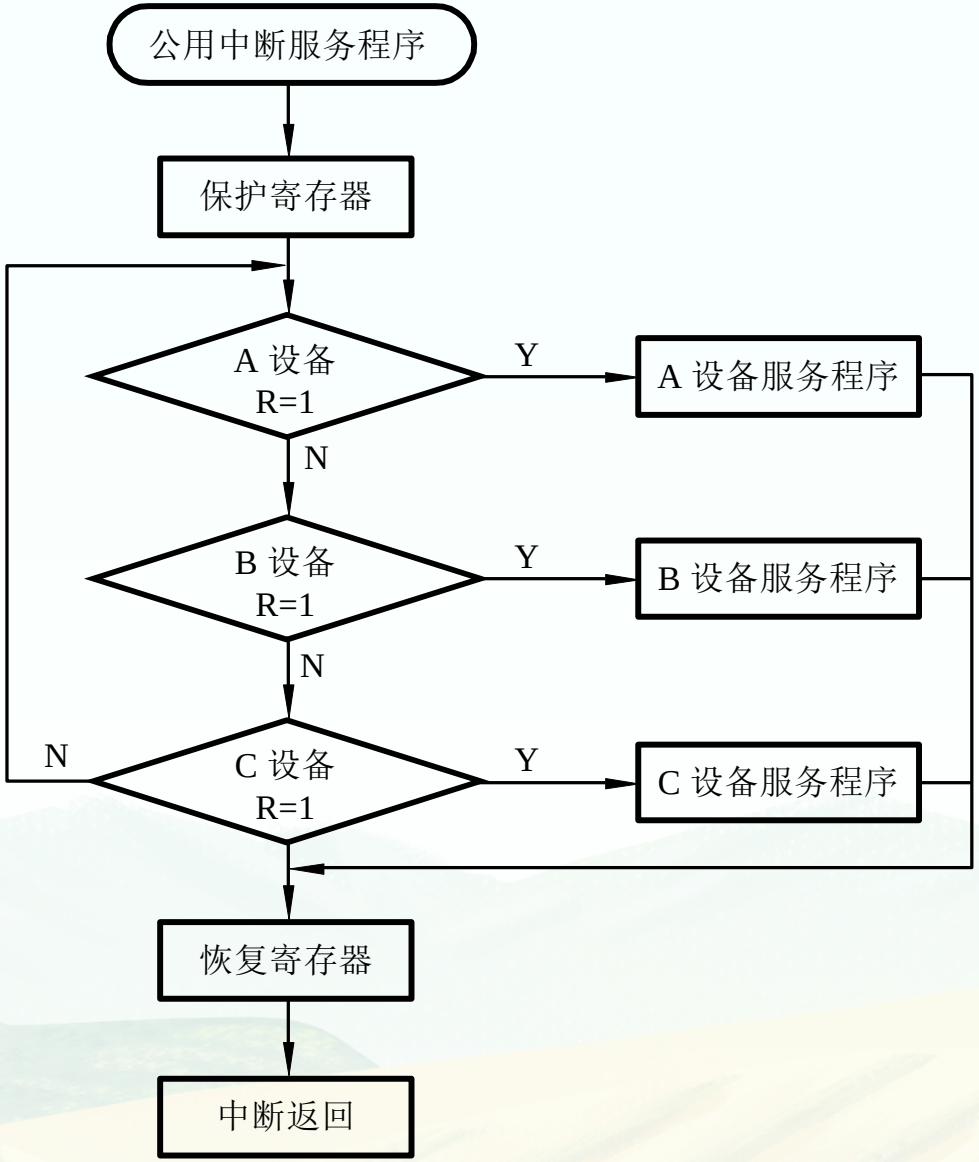
中断识别（寻找入口地址）

- 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)

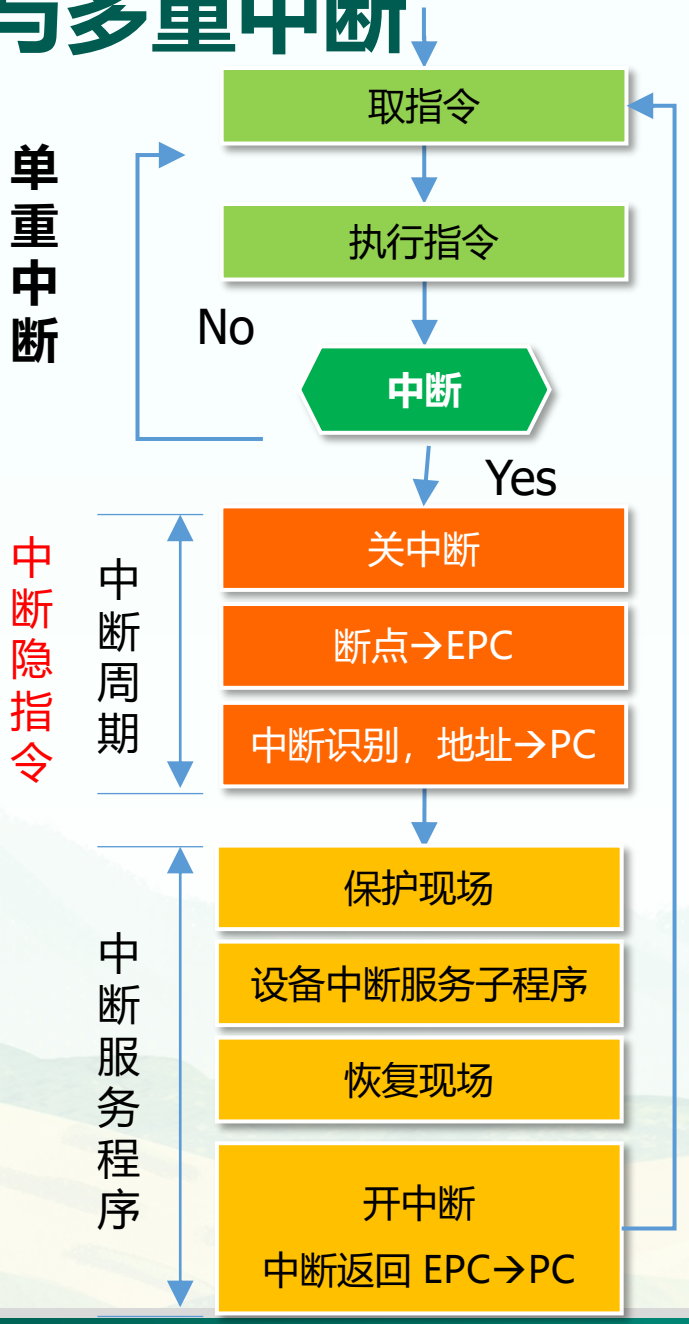
中断向量法

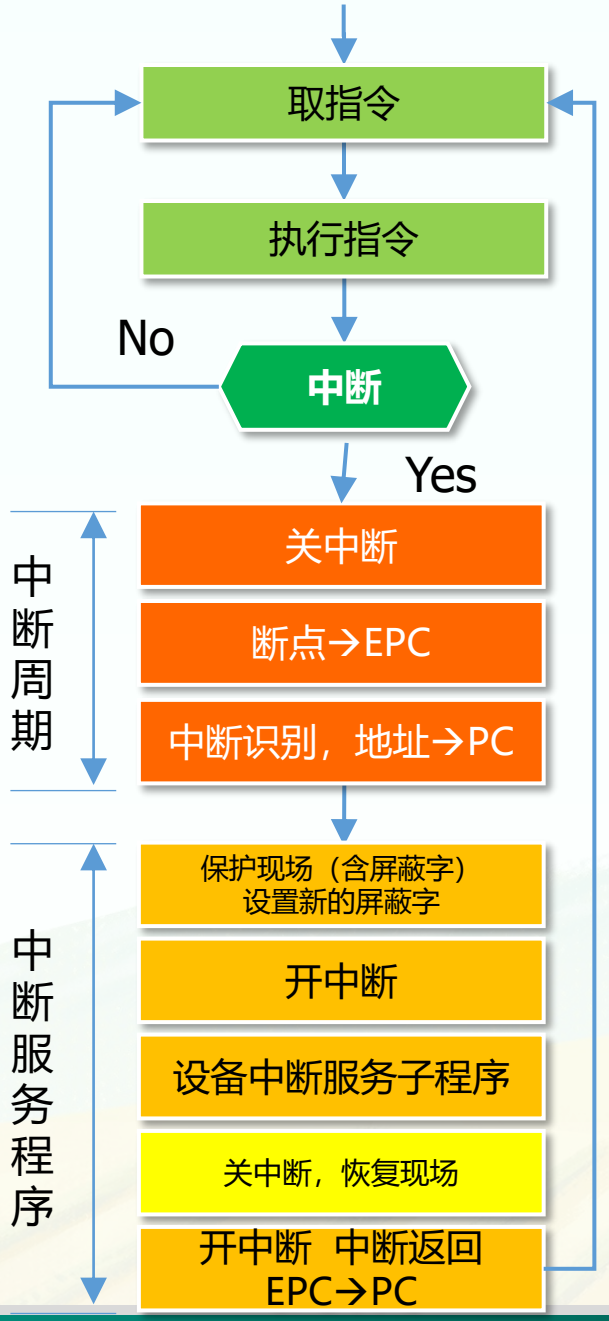

A 向量
B 向量
C 向量
A 入口 AA
B 入口 BA
C 入口 CA
向量表
AA:
PUSH…
IRET
A 设备中断
处理程序
BA:
PUSH…
IRET
B 设备中断
处理程序
CA:
PUSH…
IRET
C 设备中断
处理程序

程序识别


公用中断服务程序
保护寄存器
A 设备
R=1
Y
A 设备服务程序
N
B 设备
R=1
Y
B 设备服务程序
N
C 设备
R=1
N
Y
C 设备服务程序
恢复寄存器
中断返回

单重中断与多重中断

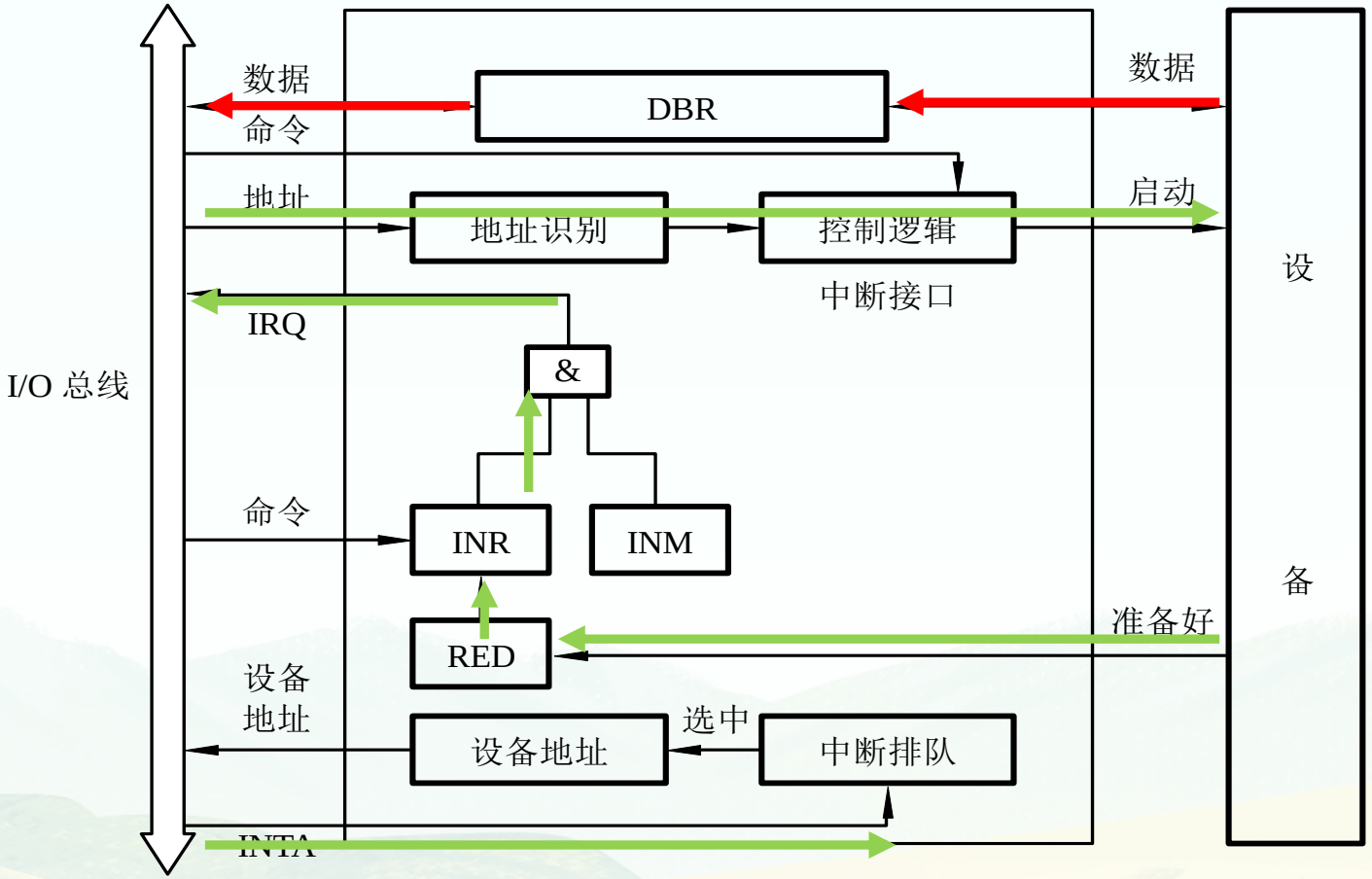

单重中断
取指令
执行指令
No
中断
Yes
中断隐指令
中断周期
关中断
断点→EPC
中断识别，地址→PC
中断服务程序
保护现场
设备中断服务子程序
恢复现场
开中断
中断返回 EPC→PC


多重中断
取指令
执行指令
No
中断
Yes
中断周期
关中断
断点→EPC
中断识别，地址→PC
中断服务程序
保护现场（含屏蔽字）
设置新的屏蔽字
开中断
设备中断服务子程序
关中断，恢复现场
开中断 中断返回
EPC→PC

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

中断方式接口


数据
命令
地址
DBR
地址识别
控制逻辑
中断接口
IRQ
&
I/O 总线
命令
INR
INM
RED
设备
地址
设备地址
选中
中断排队
INTA
数据
启动
设
备
准备好

工作过程

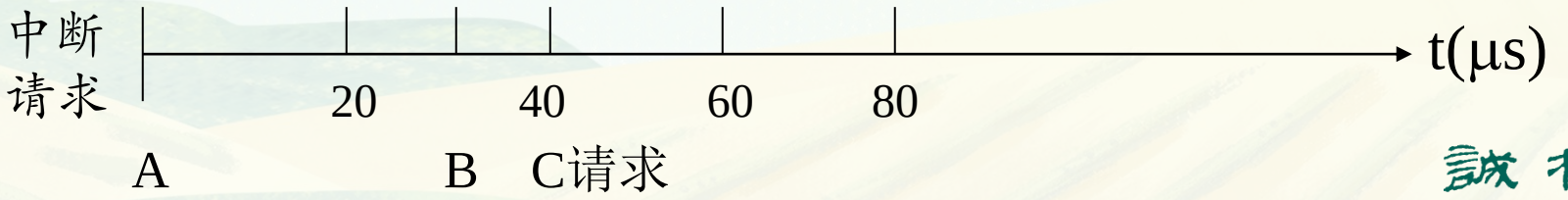
- 主机启动设备
- 设备准备传送
- 发中断请求信号
- 主机响应中断
- 数据传送

举例

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 μ s。

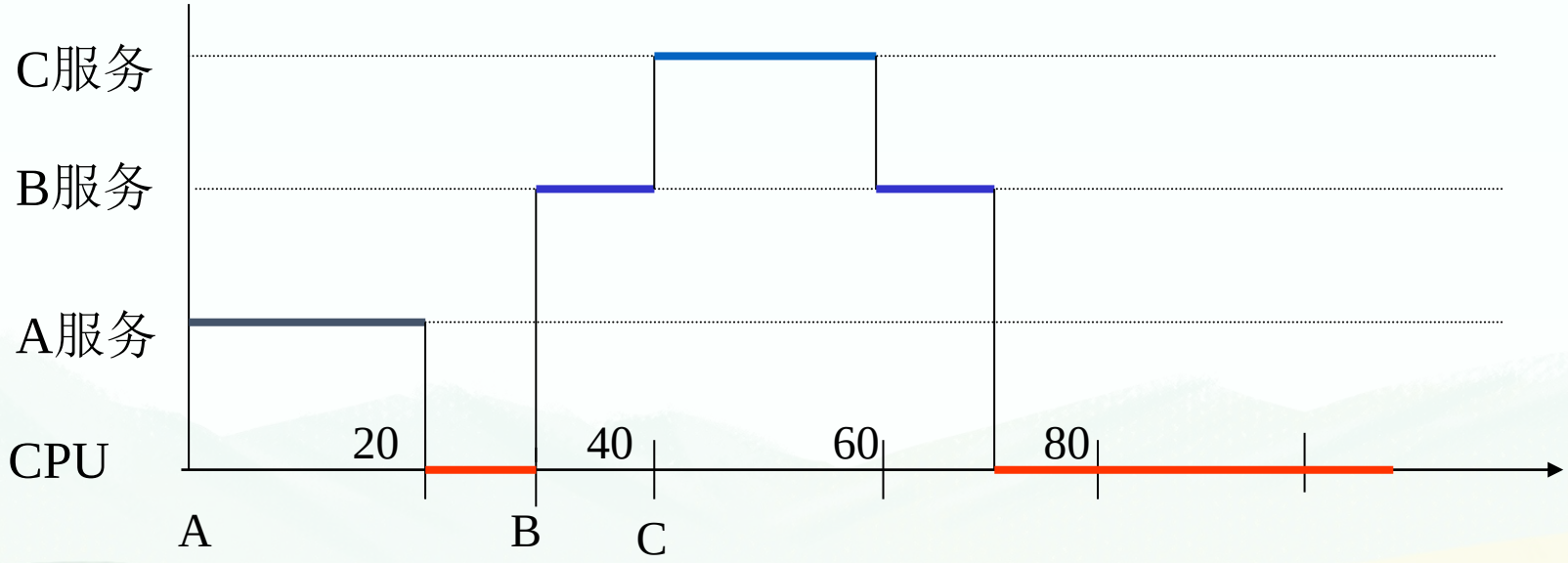


举例

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：



南京農業大學
NANJING AGRICULTURAL UNIVERSITY

谢 谢

计算机组成与系统结构