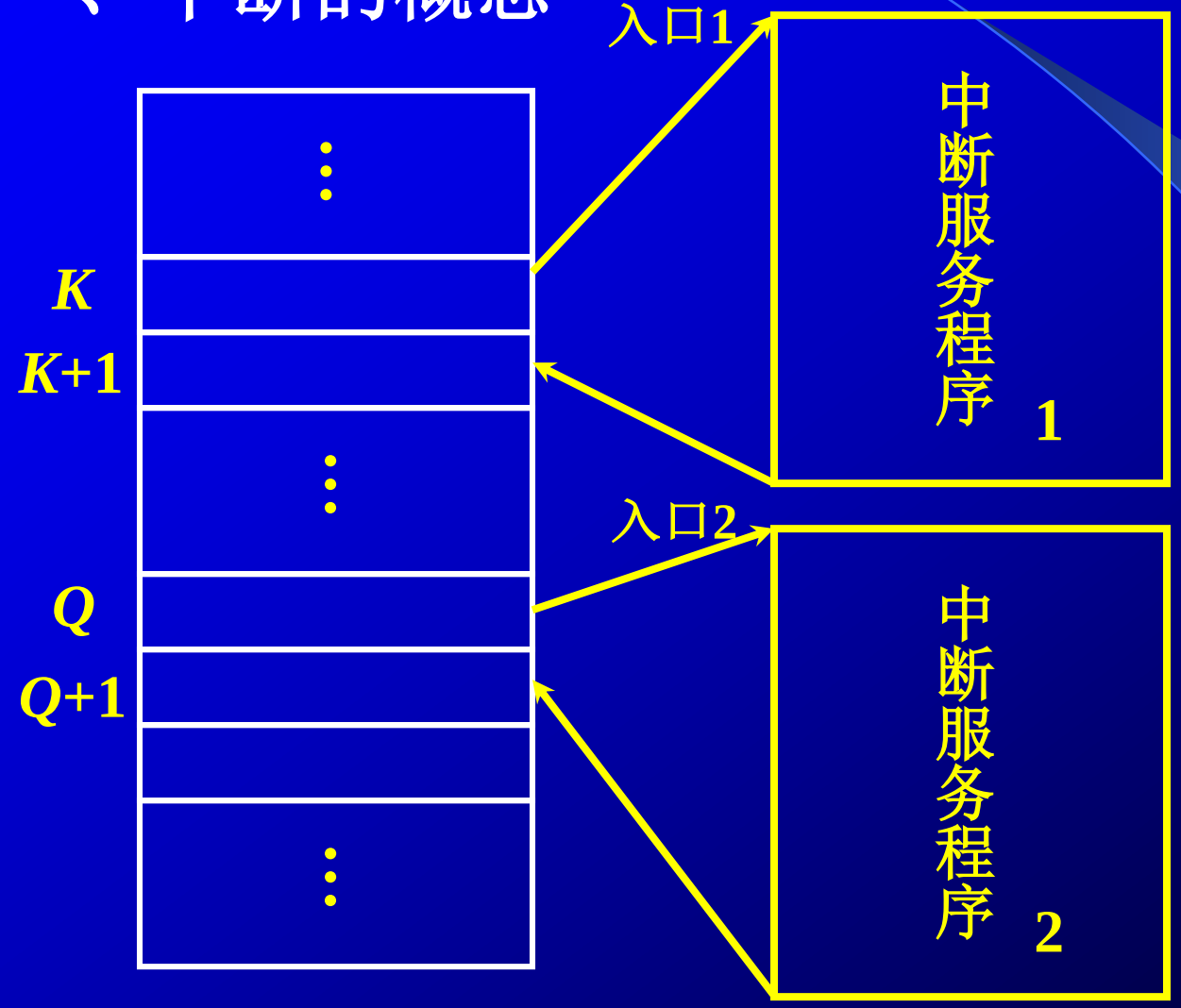
第11章　中断

11.1 中断系统

11.2 I/O中断

11.1 中断系统

一、中断的概念



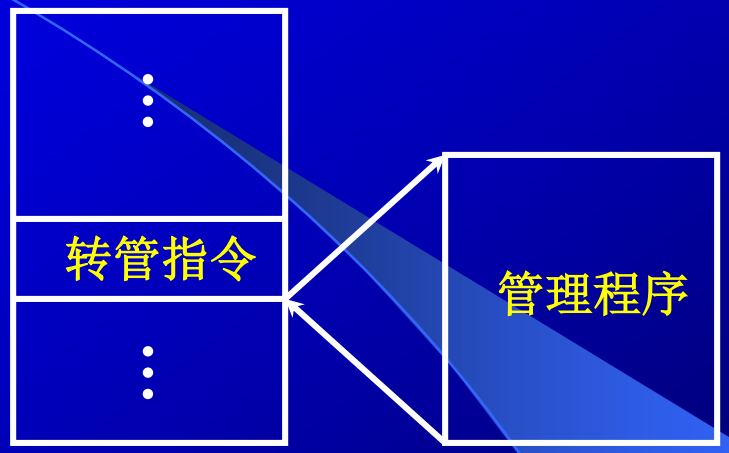
异常情况或特殊请求时，停止现行程序的运行，转向对这些异常情况或特殊请求的处理，处理结束后再返回到现行程序的间断处

二、概述

1. 引起中断的各种因素

(1) 人为设置的中断

如　转管指令



(2) 程序性事故　溢出、操作码不能识别、除法非法

(3) 硬件故障

(4) I/O 设备

(5) 外部事件　　用 键盘中断 现行程序

2. 中断系统需解决的问题

(1) 各中断源 如何 向 CPU 提出请求？

(2) 各中断源 同时 提出 请求 怎么办？

(3) CPU 什么 条件、什么 时间、以什么 方式 响应中断？

(4) 如何 保护现场？

(5) 如何 寻找入口地址？

(6) 如何 恢复现场，如何 返回？

(7) 处理中断的过程中又 出现新的中断 怎么办？

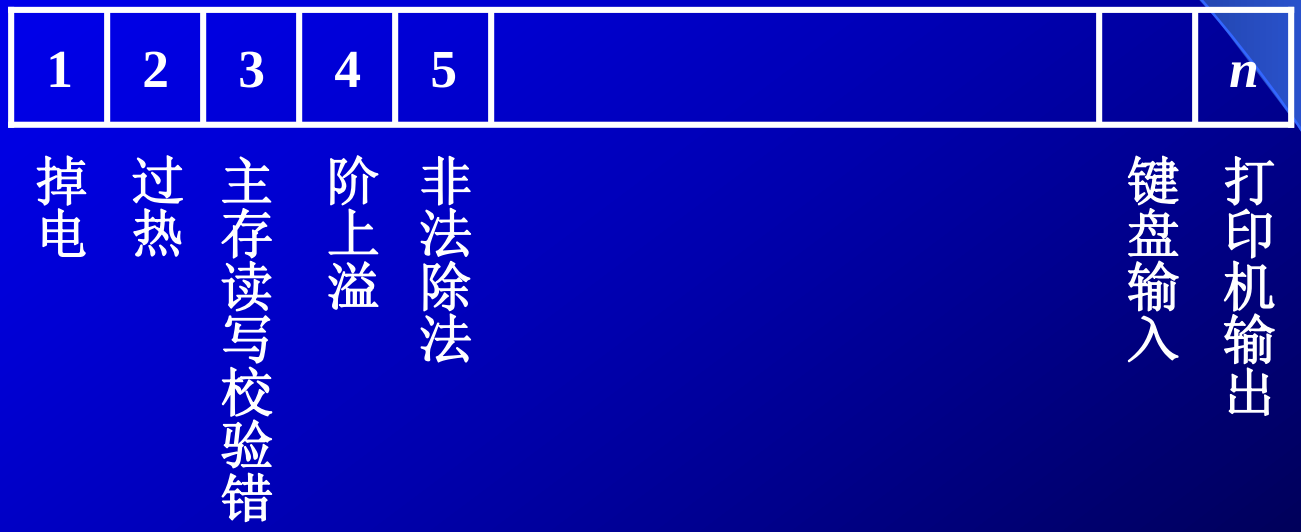
硬件 + 软件

二、中断请求标记和中断判优逻辑 11.1

1. 中断请求标记 INTR

一个请求源 一个 INTR 中断请求标记触发器

多个INTR 组成 中断请求标记寄存器



INTR 分散 在各个中断源的 接口电路中

INTR 集中 在 CPU 的中断系统 内

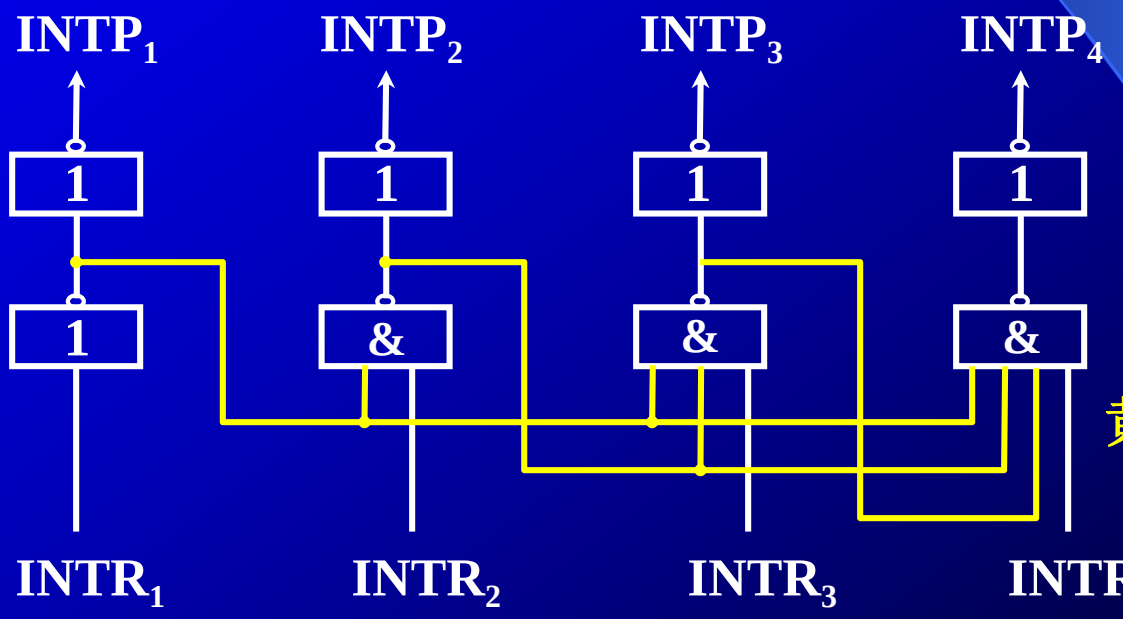
2. 中断判优逻辑（中断源优先级）

(1) 硬件实现（排队器）

① 分散 在各个中断源的 接口电路中　链式排队器

参见 11.2 I/O中断

② 集中 在 CPU 内

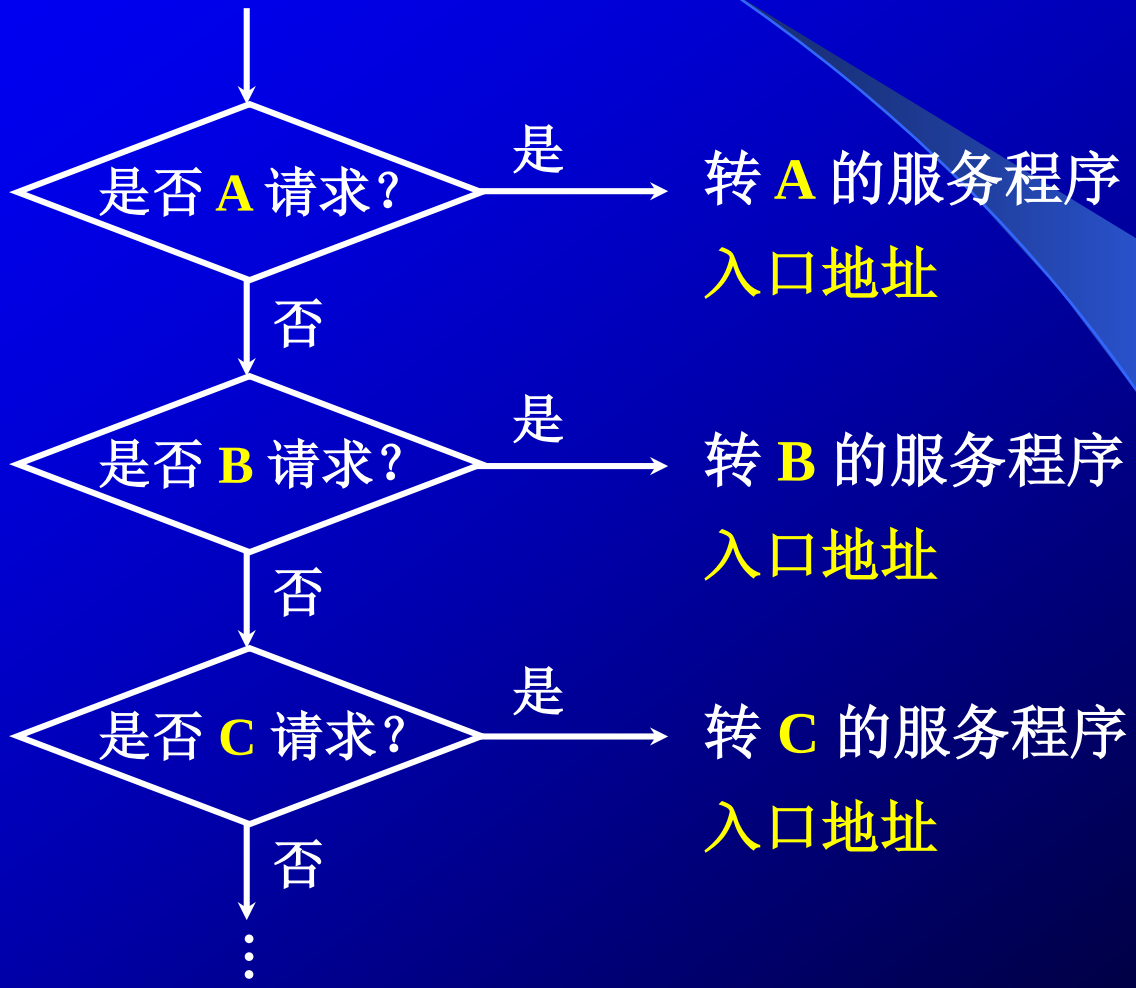


黄线“封住”低优先级中断源

$INTR_1$、$INTR_2$、$INTR_3$、$INTR_4$ 优先级 按 降序 排列

(2) 软件实现（程序查询）

A、B、C 优先级按 降序 排列



三、中断服务程序入口地址的寻找

1. 硬件向量法（硬件→向量地址→入口地址）

向量地址　12H、13H、14H

入口地址　200、 300、 400

2. 软件查询法

八个中断源 1，2，…　8 按 降序 排列

中断识别程序（入口地址 M）

地 址	指 令	说 明
M	SKP　DZ $1^{\#}$	$1^{\#}$　D = 0 跳 (D为完成触发器)
	JMP　$1^{\#}$ SR	$1^{\#}$　D = 1 转$1^{\#}$ 服务程序
	SKP　DZ $2^{\#}$	$2^{\#}$　D = 0 跳
	JMP　$2^{\#}$ SR	$2^{\#}$　D = 1 转$2^{\#}$ 服务程序
	⋮	
	SKP　DZ $8^{\#}$	$8^{\#}$　D = 0 跳
	JMP　$8^{\#}$ SR	$8^{\#}$　D = 1 转$8^{\#}$ 服务程序

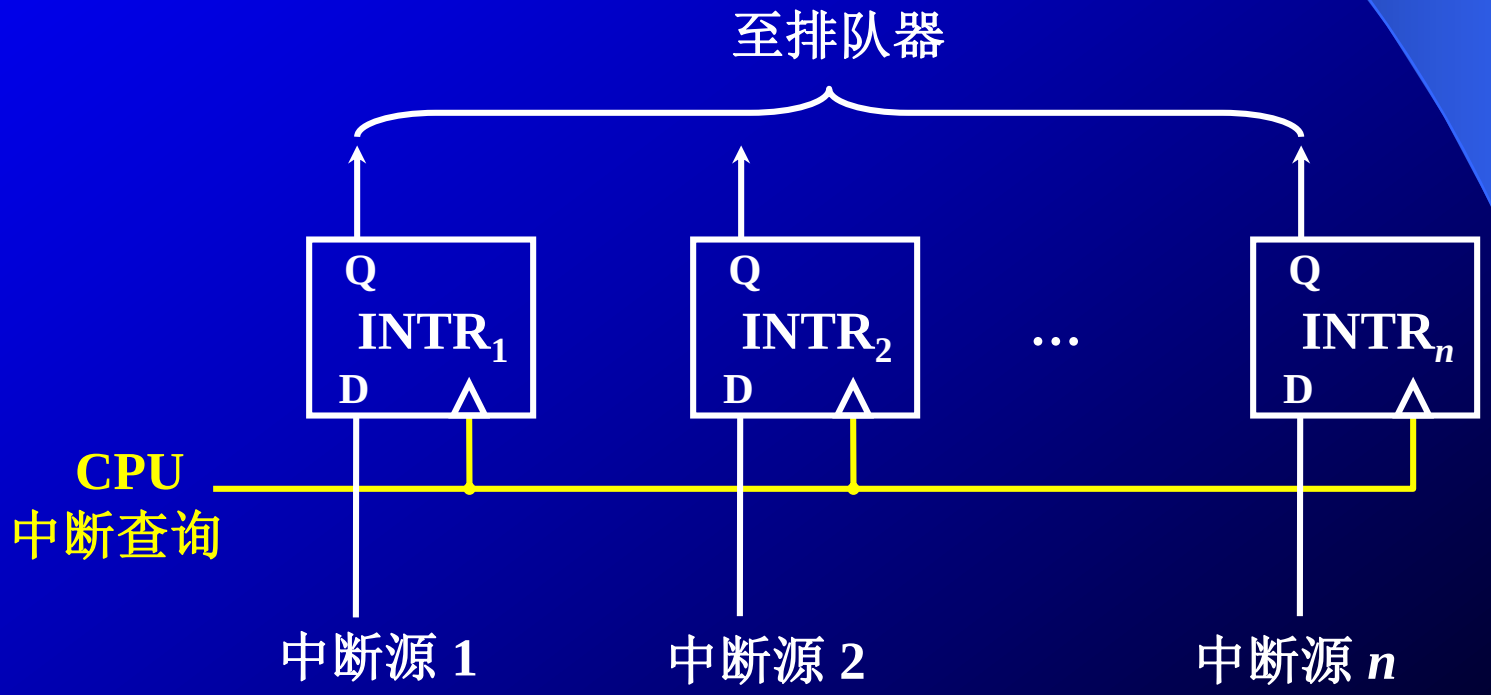
四、中断响应

1. 响应中断的 条件

允许中断触发器　EINT = 1 (开中断置1，关中断置0)

2. 响应中断的 时间

指令执行周期结束时刻由CPU 发查询信号



3. 中断隐指令

(1) 保护程序断点

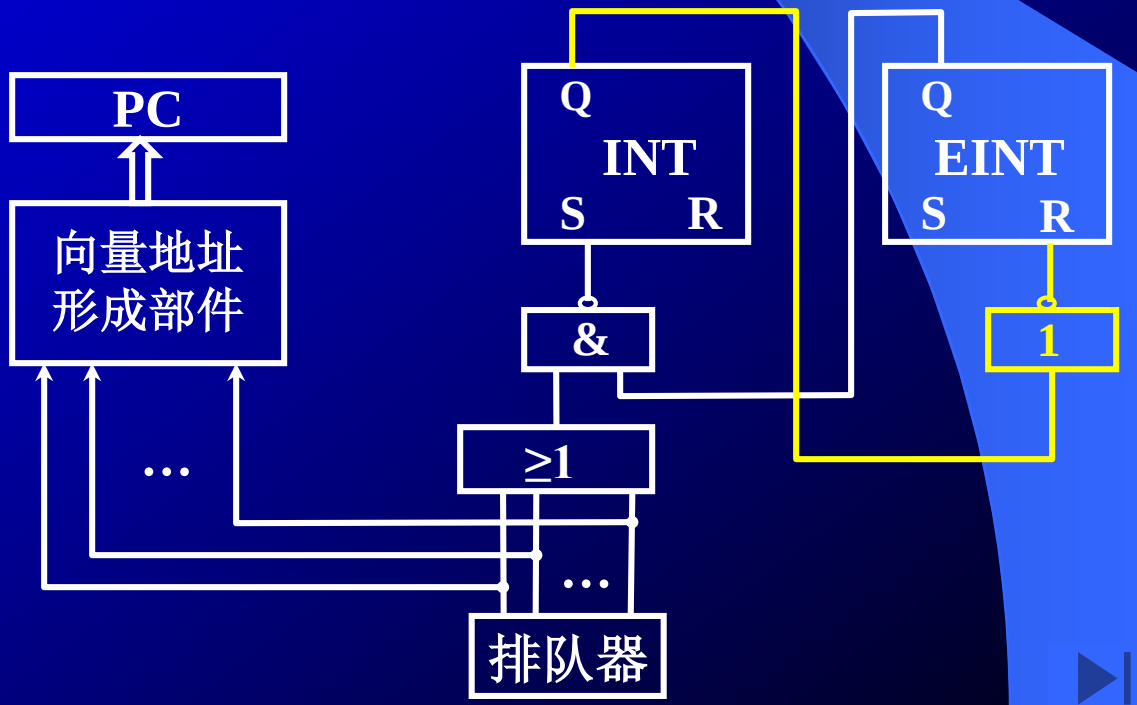
断点(当前PC)存于 特定地址（ 0 号地址） 内　　断点 进栈

(2) 寻找服务程序入口地址

向量地址 ⟶ PC　（硬件向量法）

中断识别程序 入口地址 M ⟶ PC　（软件查询法）

(3) 硬件 关中断



INT 中断标记

EINT 允许中断

R – S 触发器

五、保护现场和恢复现场

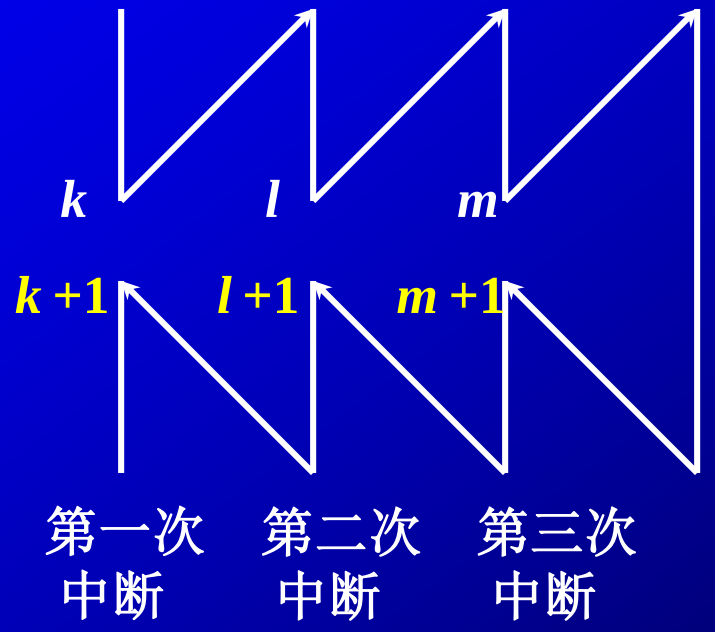
1. 保护现场
   - 断点 —— 中断隐指令 完成
   - 寄存器 内容 —— 中断服务程序 完成
2. 恢复现场 —— 中断服务程序 完成

中断服务程序

保护现场	PUSH
其它服务程序	视不同请求源而定
恢复现场	POP
中断返回	IRET

六、中断屏蔽技术

1. 多重中断的概念

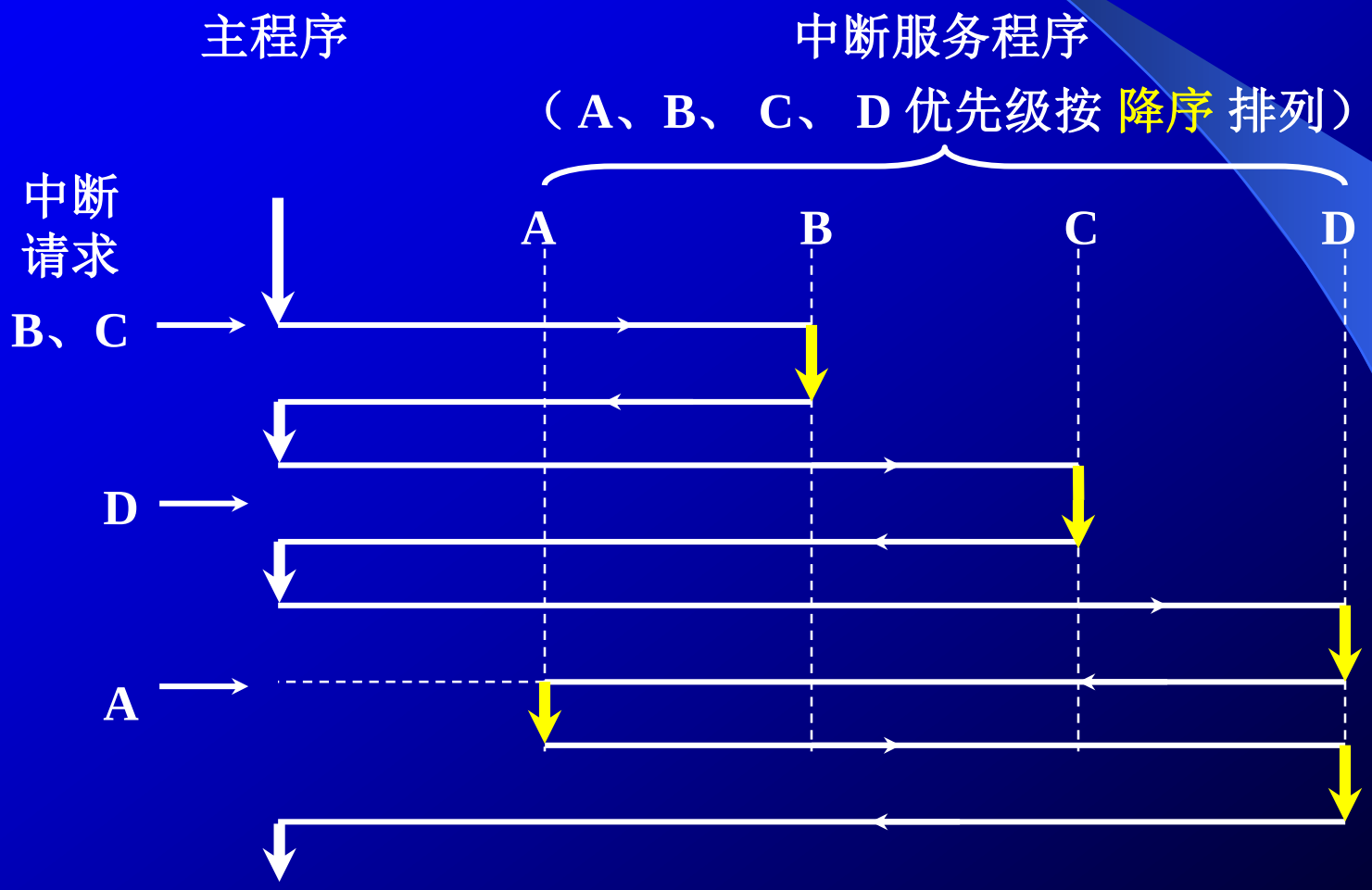


程序断点 $k+1$，$l+1$，$m+1$

2. 实现多重中断的条件

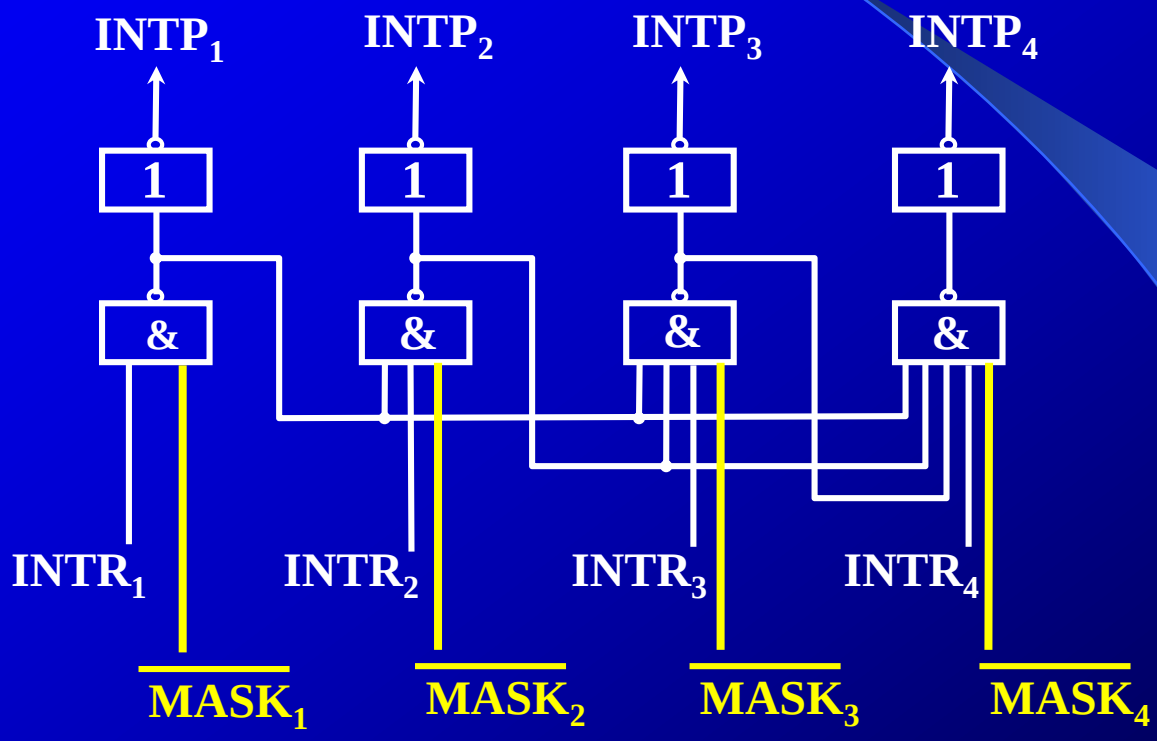
(1) 提前 设置 开中断 指令

(2) 优先级别高 的中断源 有权中断优先级别低 的中断源



3. 屏蔽技术

(1) 屏蔽触发器的作用



MASK = 0（未屏蔽）

$MASK_i$ = 1 （屏蔽）

$INTP_i$ = 0 （不能被排队选中）

(2) 屏蔽字

16个中断源 1，2，3，… 16 按 降序 排列

优先级	屏　　蔽　　字
1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
5	0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
6	0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
⋮	⋮
15	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
16	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

(3) 屏蔽技术可改变处理优先等级

响应优先级　　不可改变

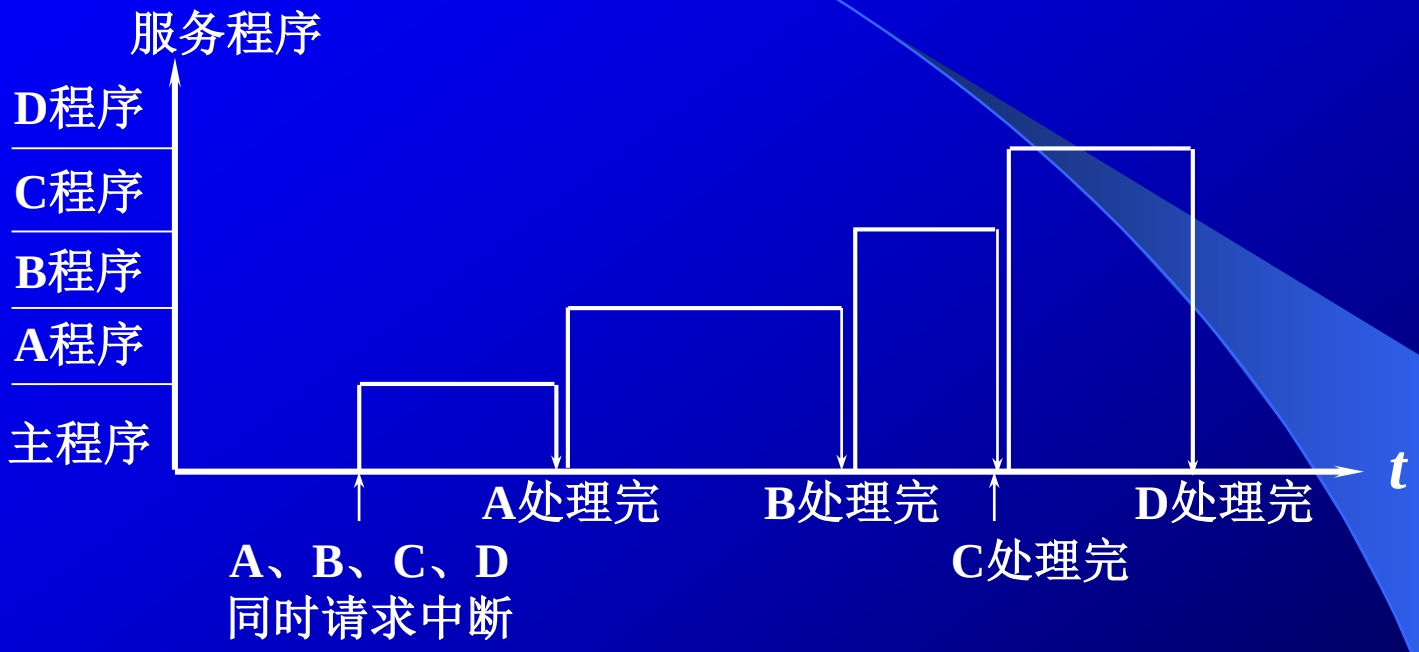
处理优先级　　可改变（通过重新设置屏蔽字）

中断源	原屏蔽字	新屏蔽字
A	1 1 1 1	1 1 1 1
B	0 1 1 1	0 1 0 0
C	0 0 1 1	0 1 1 0
D	0 0 0 1	0 1 1 1

响应优先级 **A→B→C→D** 降序排列

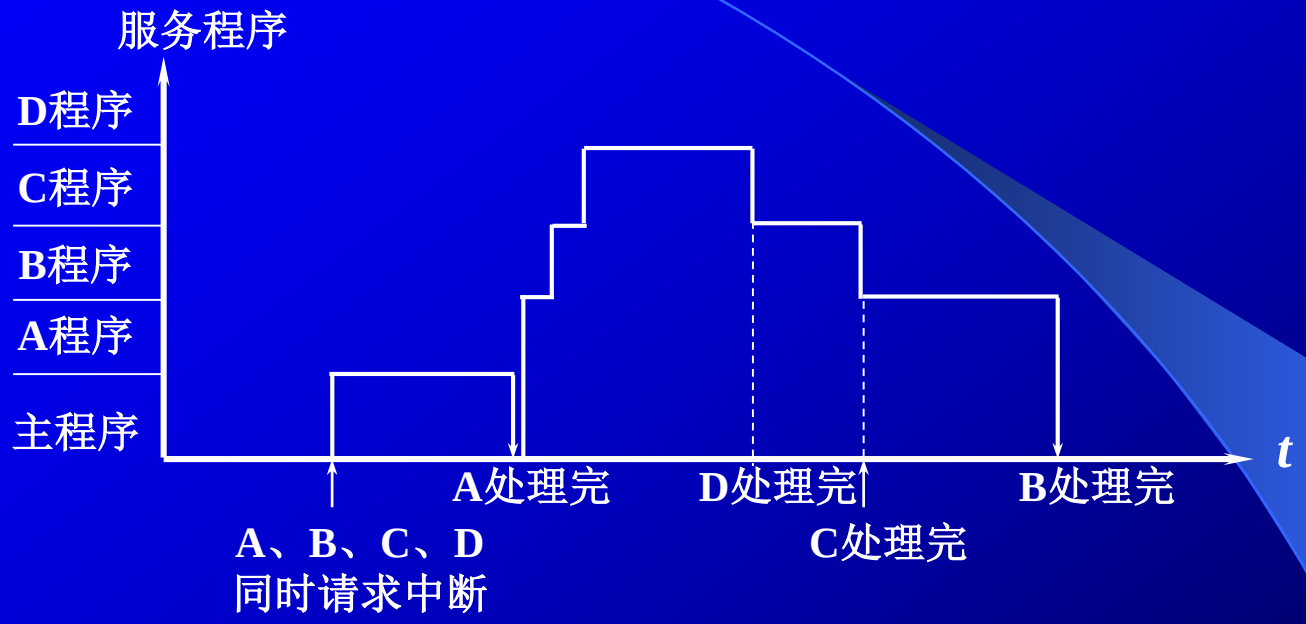
处理优先级 **A→D→C→B** 降序排列

(3) 屏蔽技术可改变处理优先等级



CPU 执行程序轨迹（原屏蔽字）

(3) 屏蔽技术可改变处理优先等级



CPU 执行程序轨迹（新屏蔽字）

(4) 屏蔽技术的其他作用

可以 人为地屏蔽 某个中断源的请求

便于程序控制

4. 多重中断的断点保护

(1) 断点进栈　　中断隐指令 完成

(2) 断点存入“ 0 ” 地址　　中断隐指令 完成

中断周期　　**0 ⟶ MAR**

命令存储器写

PC ⟶ MDR　　断点 ⟶ **MDR**

(MDR) ⟶ 存入存储器

三次中断，三个断点都存入 “ 0 ” 地址

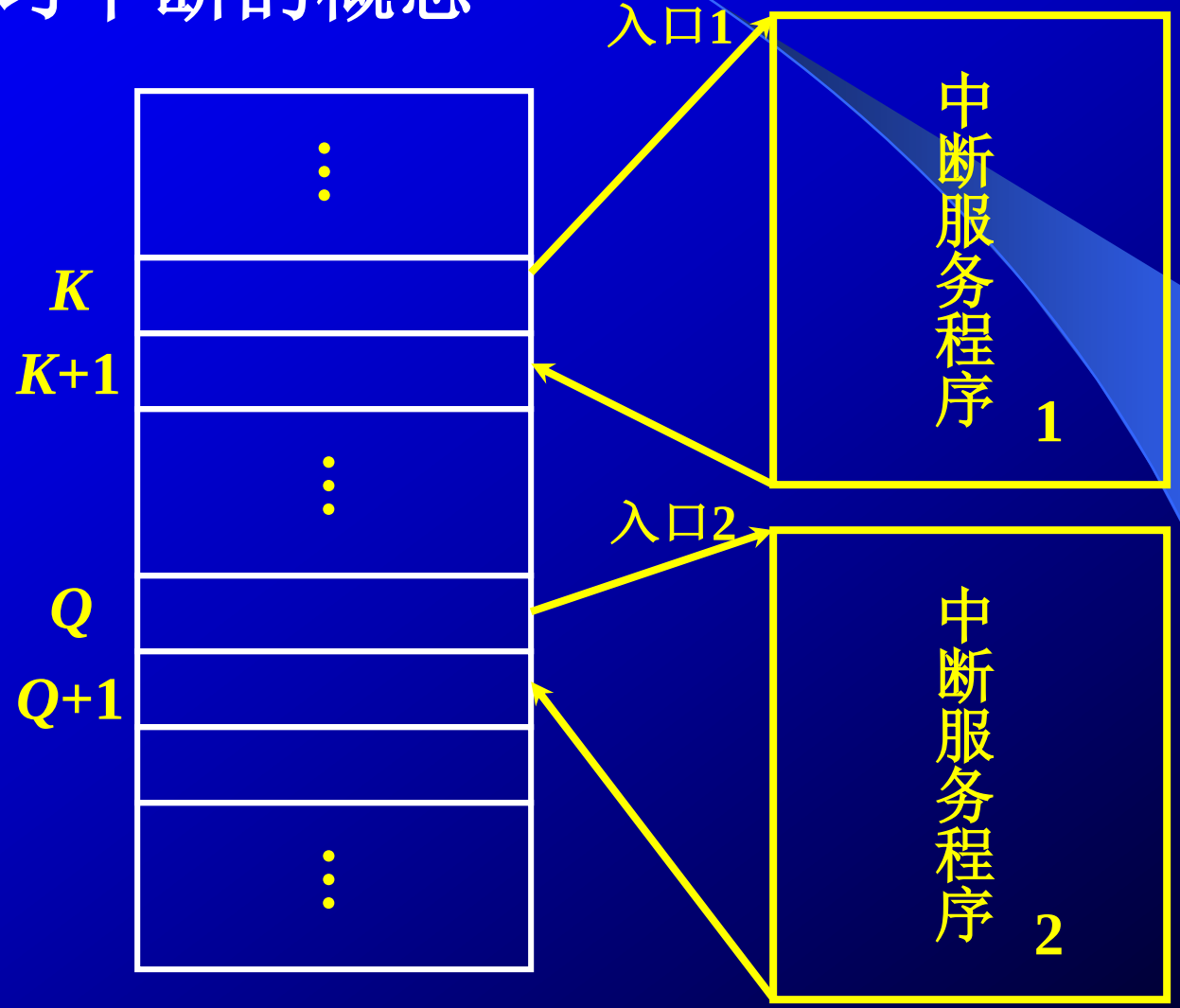
？ 如何保证断点不丢失？

(3) 程序断点存入“0”地址的断点保护11.1

地 址	内 容	说 明
0	××××	存程序断点
5	JMP SERVE	5 为向量地址
SERVE	STA SAVE	保护现场
	⋮	
置屏蔽字 →	LDA 0 STA RETURN	} 0 地址内容转存
	ENI	开中断
	⋮	} 其他服务内容
	LDA SAVE	恢复现场
	JMP @ RETURN	间址返回
SAVE	××××	存放 ACC 内容
RETURN	××××	转存 0 地址内容

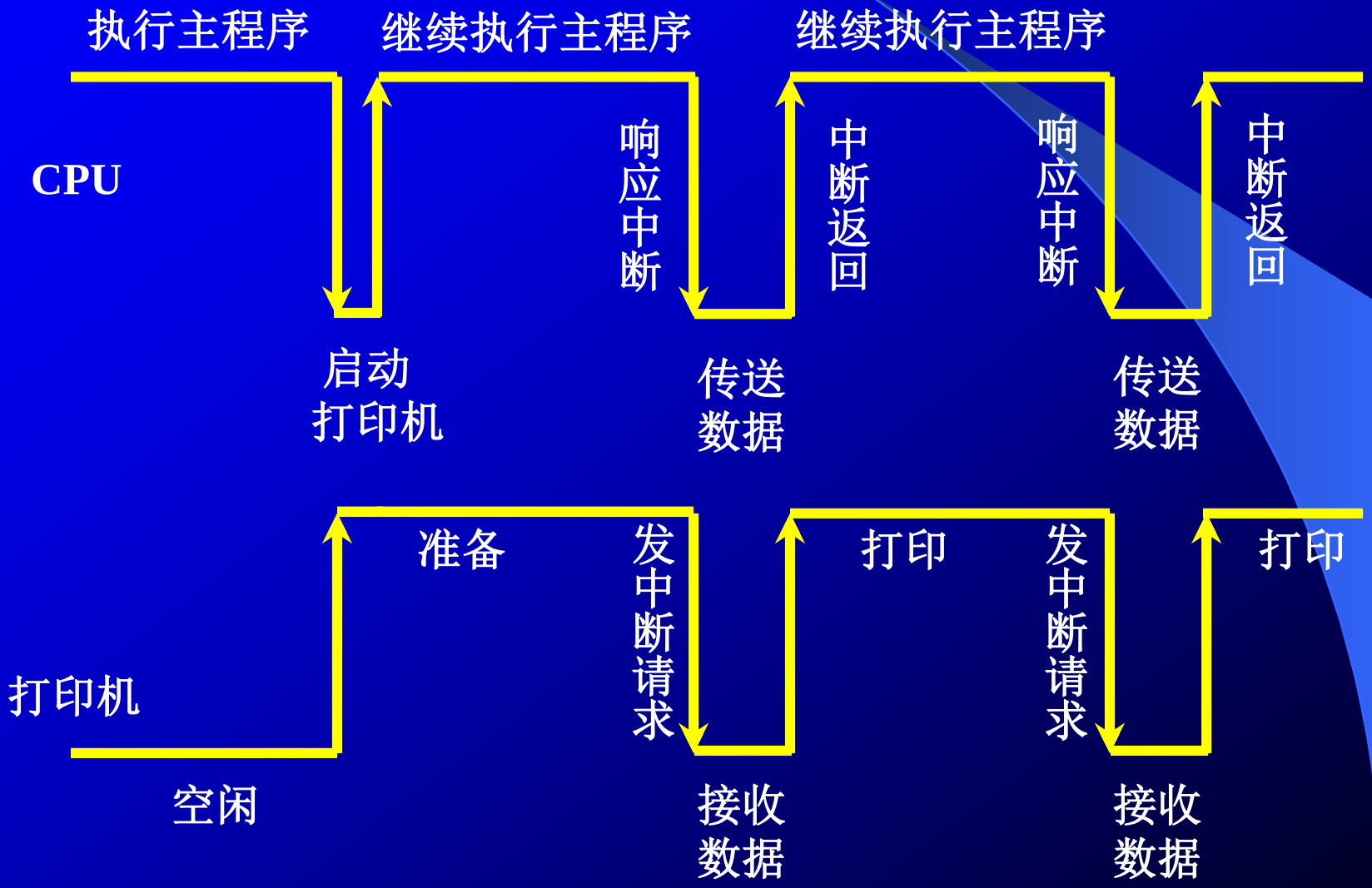
11.2 I/O中断

一、复习中断的概念



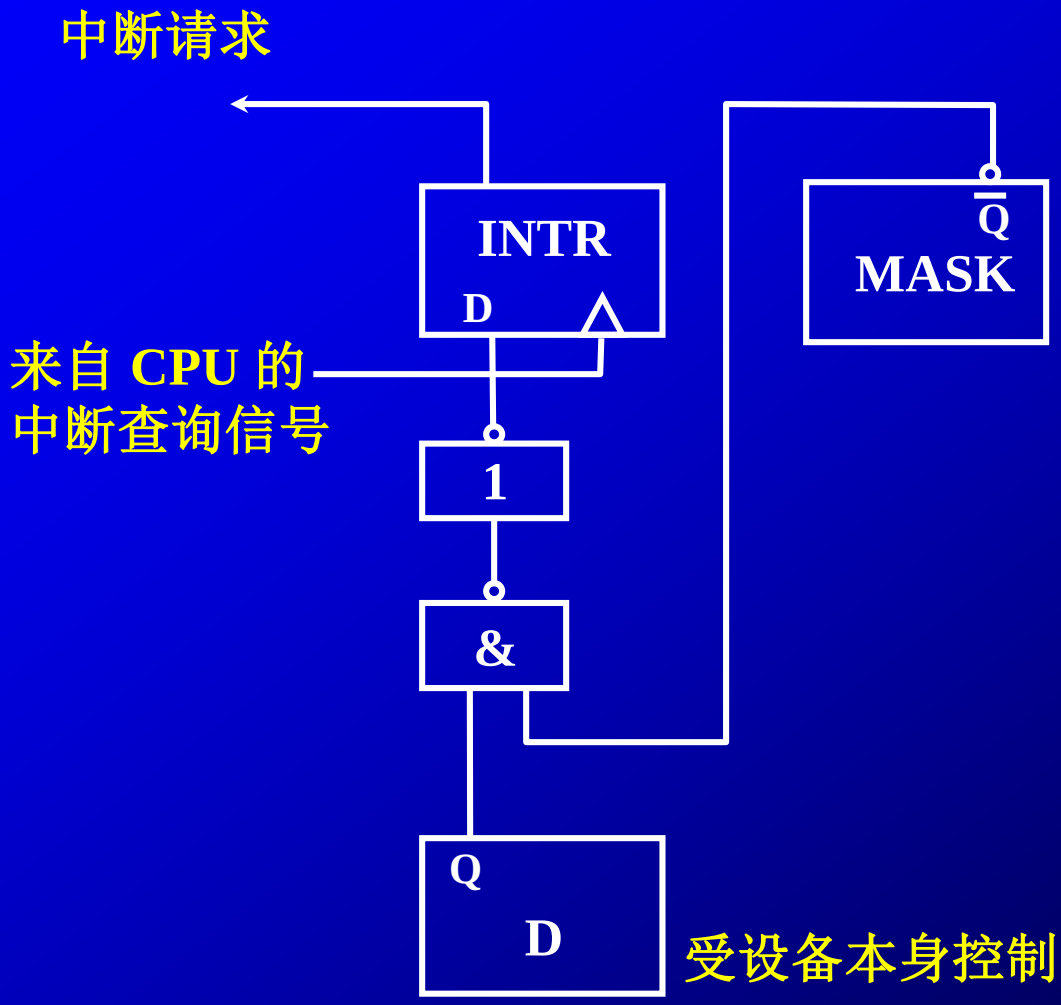
二、I/O 中断的产生

以打印机为例　CPU 与打印机并行工作



三、程序中断方式的接口电路

1. 配置中断请求触发器和中断屏蔽触发器



INTR
中断请求触发器

INTR = 1 有请求

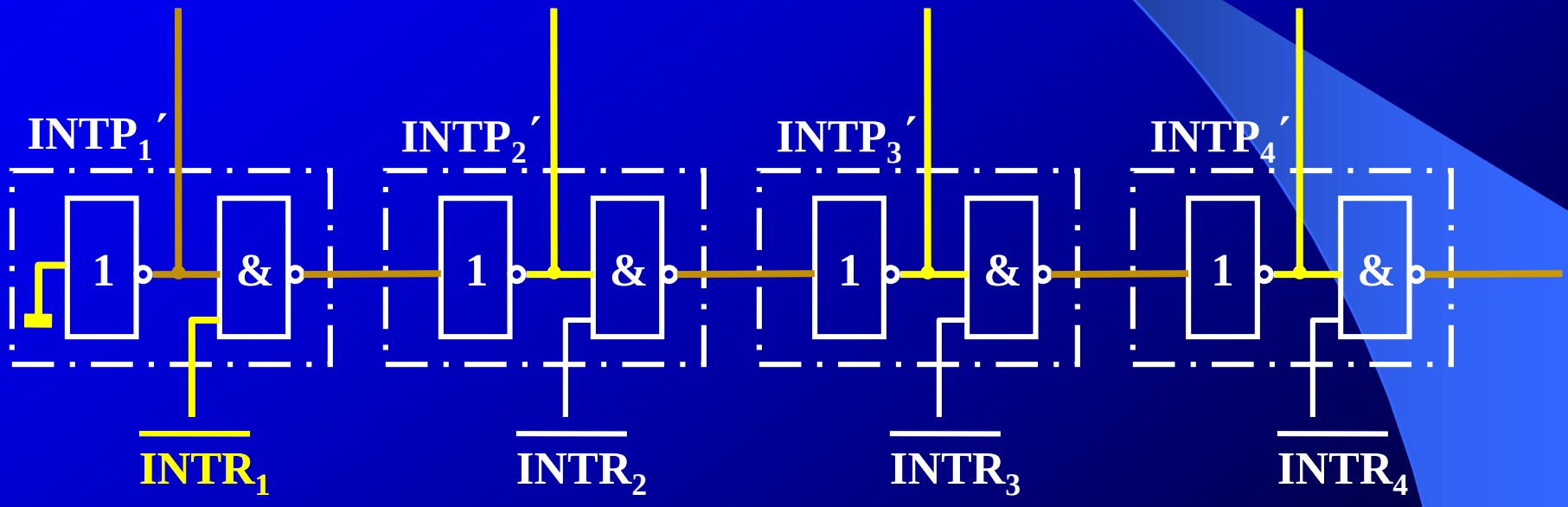
MASK
中断屏蔽触发器

MASK = 1 被屏蔽

D 完成触发器

2. 排队器

排队 { 硬件 在 CPU 内或在接口电路中（链式排队器）
软件

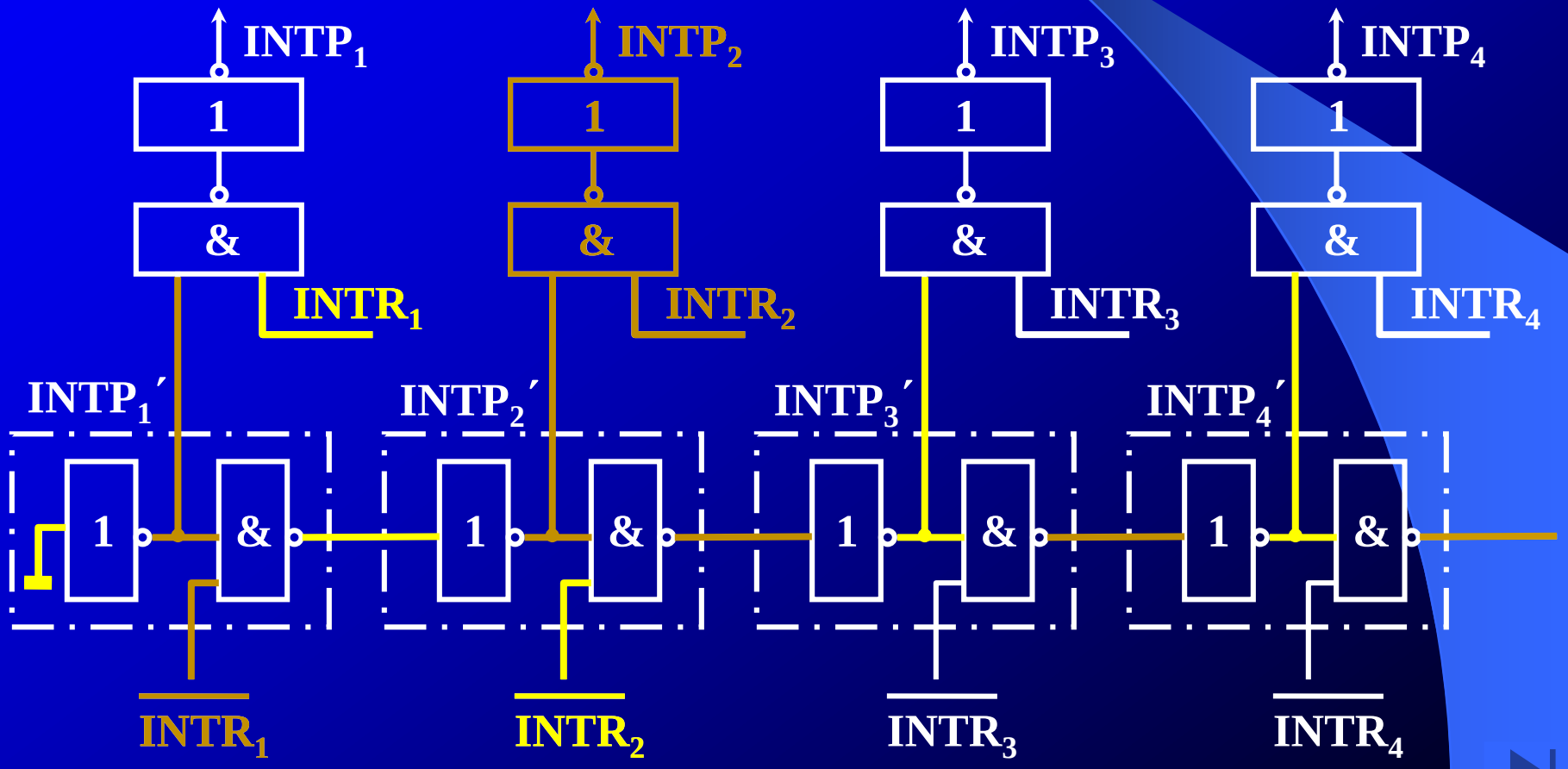


设备 1#、2#、3#、4# 优先级按 降序排列

$INTR_i = 1$ 有请求　　即 $\overline{INTR_i} = 0$

2. 排队器

排队 { 硬件 在 CPU 内或在接口电路中（链式排队器）
软件

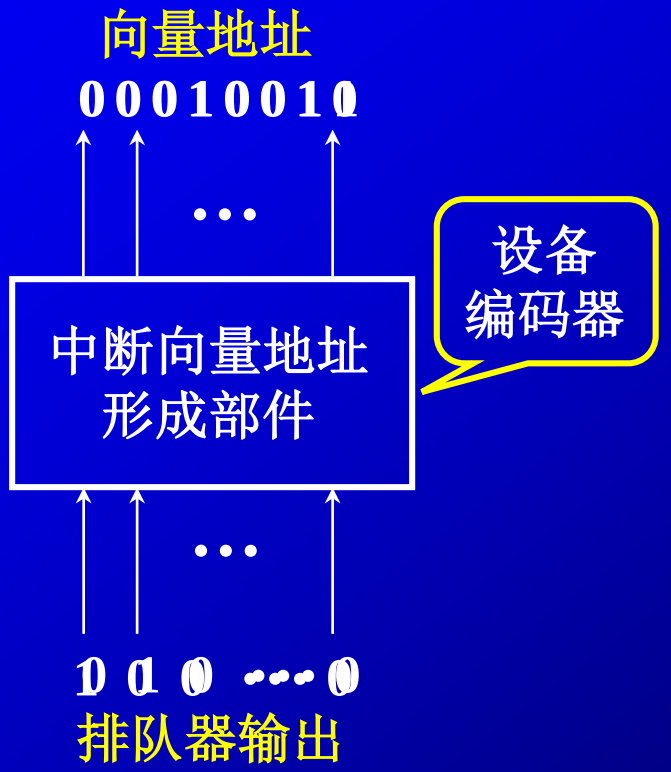


3. 中断向量地址形成部件

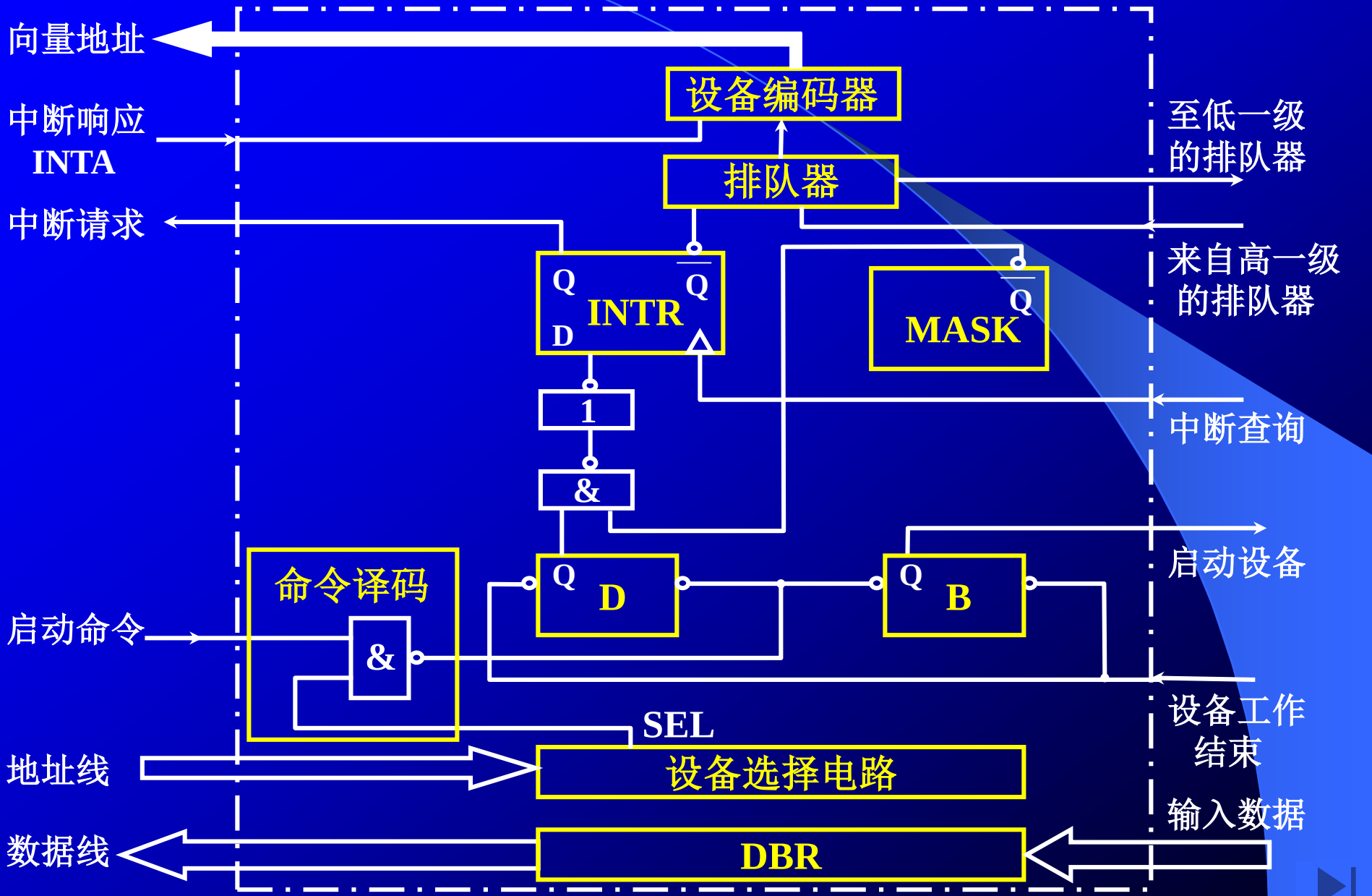
入口地址 { 由软件产生 / 硬件向量法

由 硬件 产生 向量地址

再由 向量地址 找到 入口地址



4. 程序中断方式接口电路的基本组成



四、I/O 中断处理过程

1. CPU 响应中断的条件和时间

(1) 条件

允许中断触发器 **EINT = 1**

用 开中断 指令将 EINT 置 “1”

用 关中断 指令将 EINT 置“ 0” 或硬件 自动复位

(2) 时间

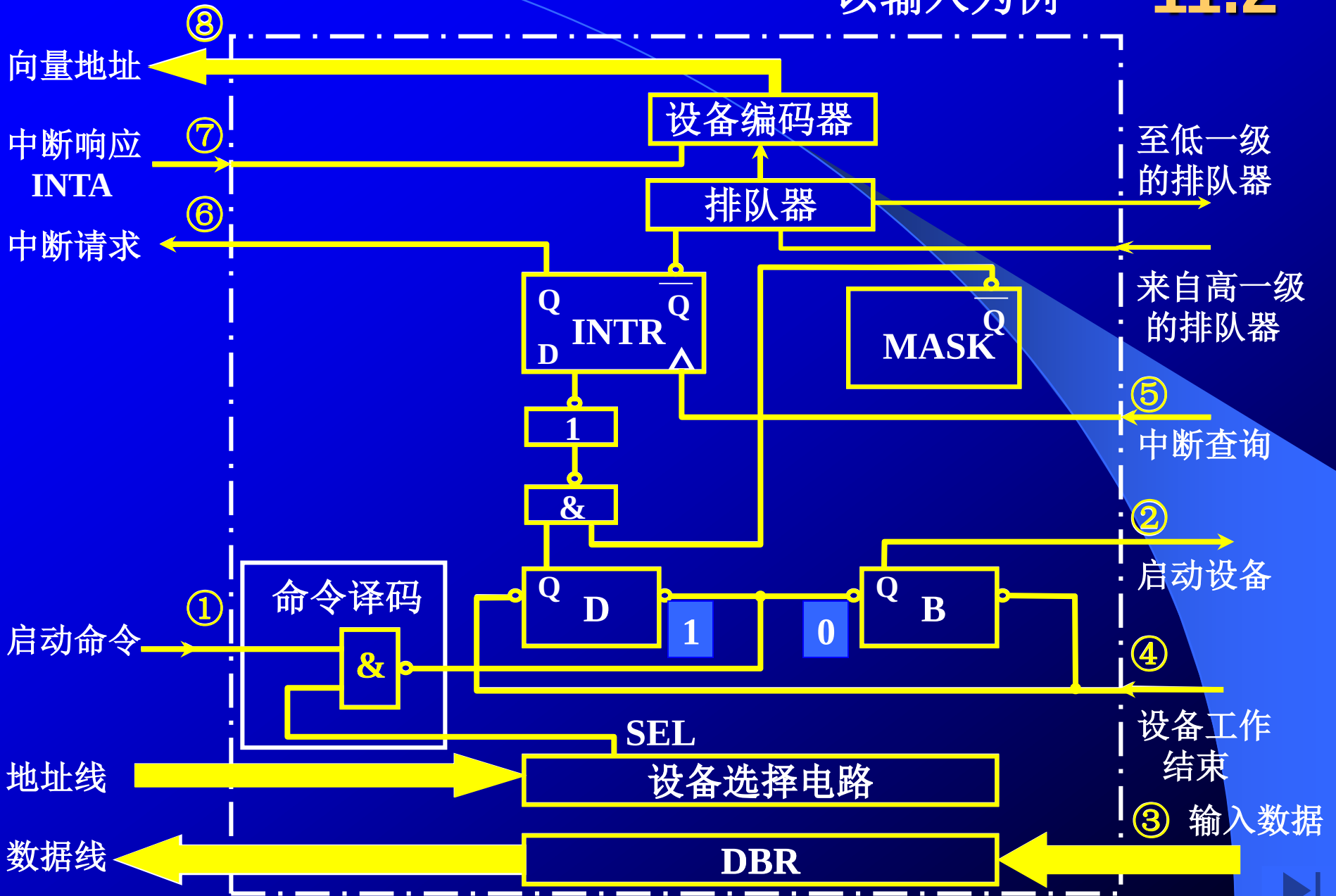
当 **D = 1**（随机）且 **MASK = 0** 时

在每条指令执行阶段的结束前

CPU 发 中断查询信号（将 **INTR** 置“1”）

2. I/O 中断处理过程

以输入为例



五、中断服务程序流程

1. 中断服务程序的流程

(1) 保护现场

- 程序断点的保护　　中断隐指令完成
- 寄存器内容的保护　进栈指令

(2) 中断服务

对不同的 I/O 设备具有不同内容的设备服务

(3) 恢复现场　　出栈指令

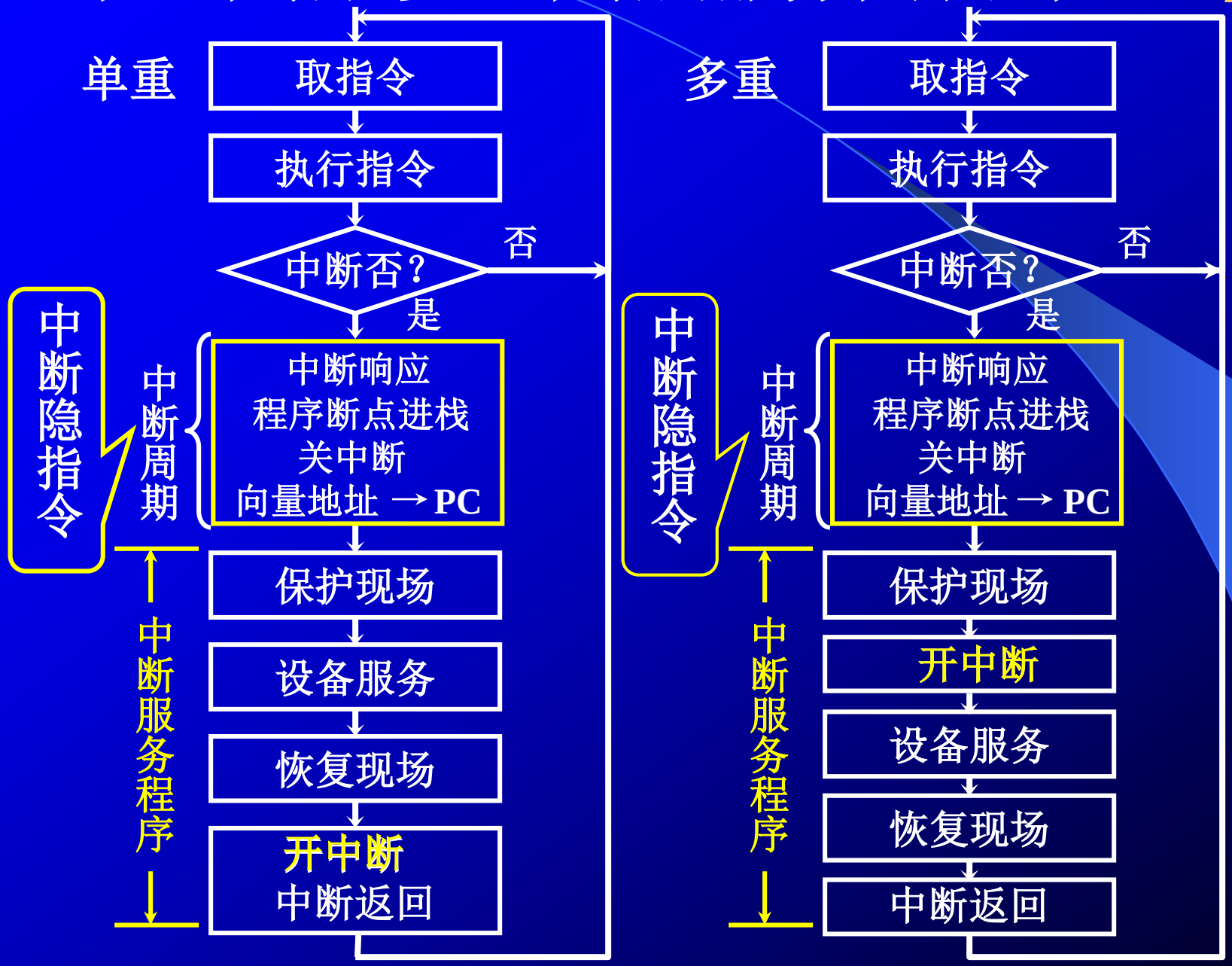
(4) 中断返回　　中断返回指令

2. 单重中断和多重中断

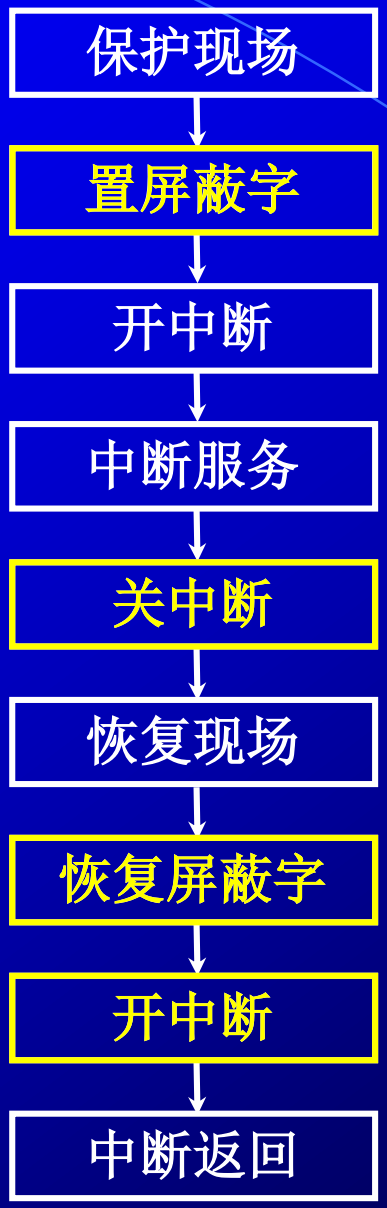
单重 中断　不允许中断 现行的 中断服务程序

多重 中断　允许级别更高 的中断源
中断 现行的 中断服务程序

3. 单重中断和多重中断的服务程序流程

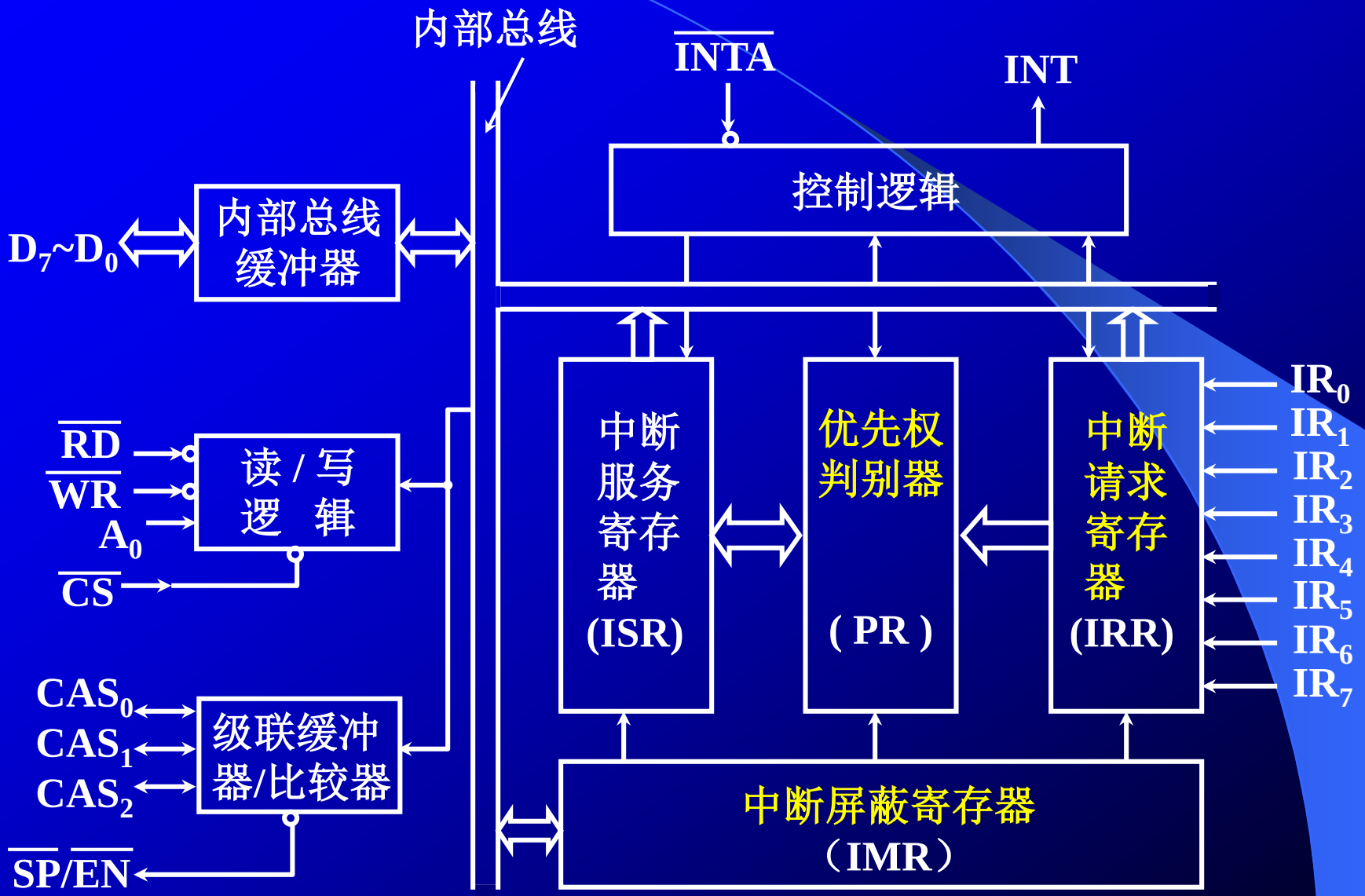


多重中断新屏蔽字的设置

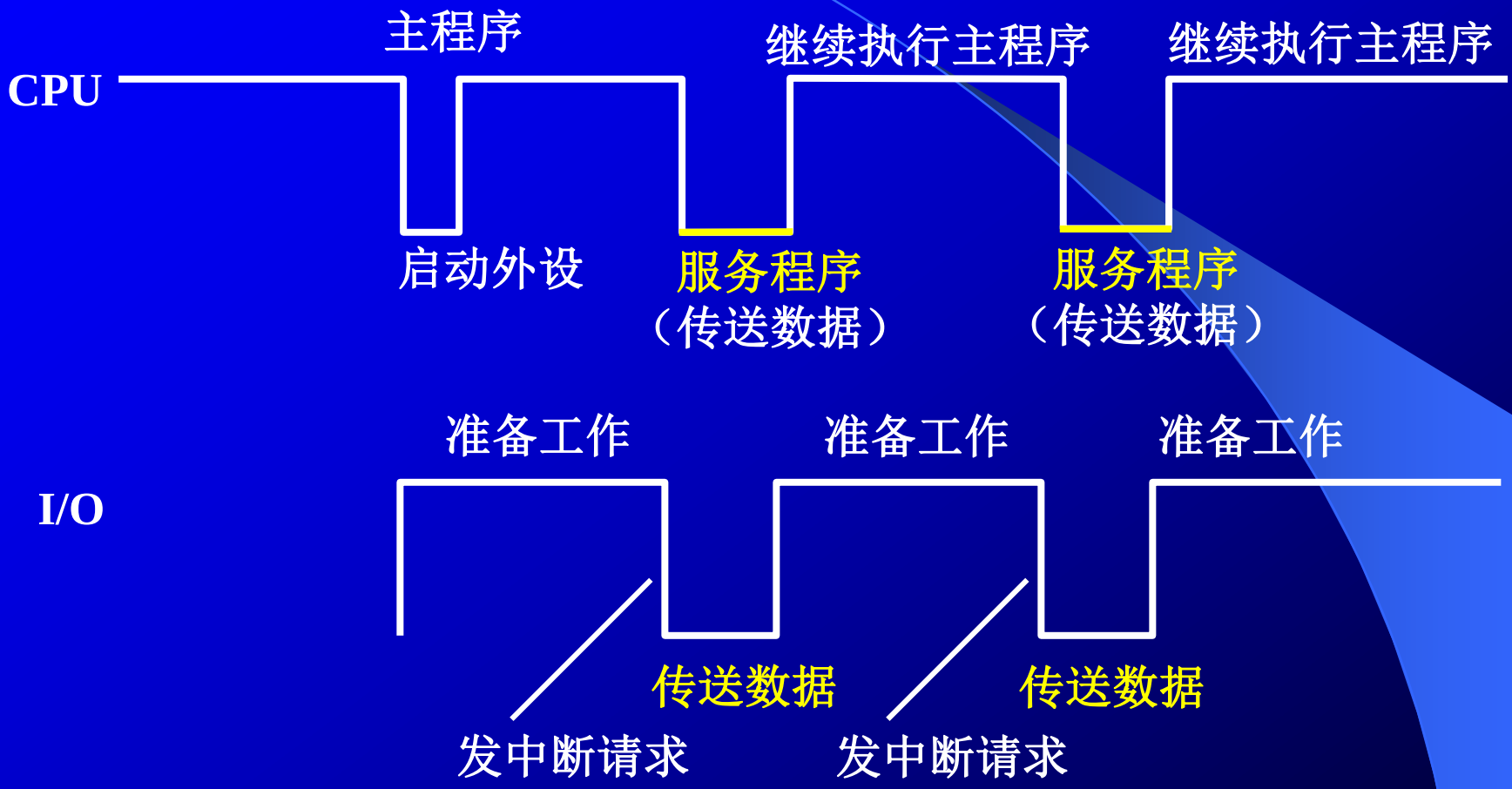


程序中断接口芯片 8259A 的内部结构

11.2



主程序和服务程序抢占 CPU 示意图 11.2



宏观 上 CPU 和 I/O 并行 工作

微观 上 CPU 中断现行程序 为 I/O 服务