

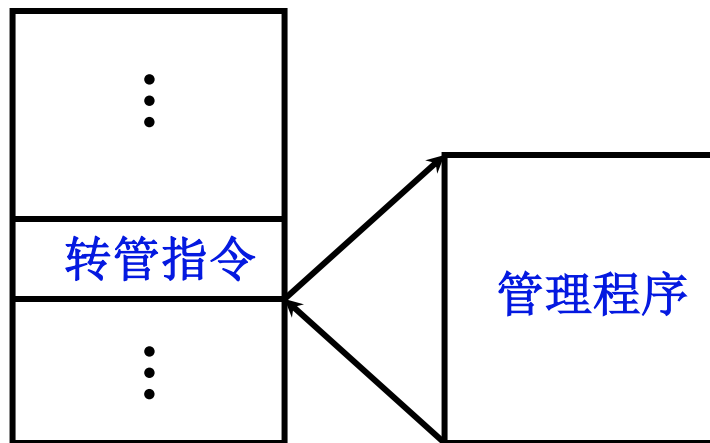
8.4 中断系统

一、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

功能 — 结构

8.4

- (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？ 优先级？ 避免冲突？
- (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

8.4

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

1	2	3	4	5			<i>n</i>
掉电	过热	主存读写校验错	阶上溢	非法除法		键盘输入	打印机输出

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

8.4

(1) 硬件实现（排队器）

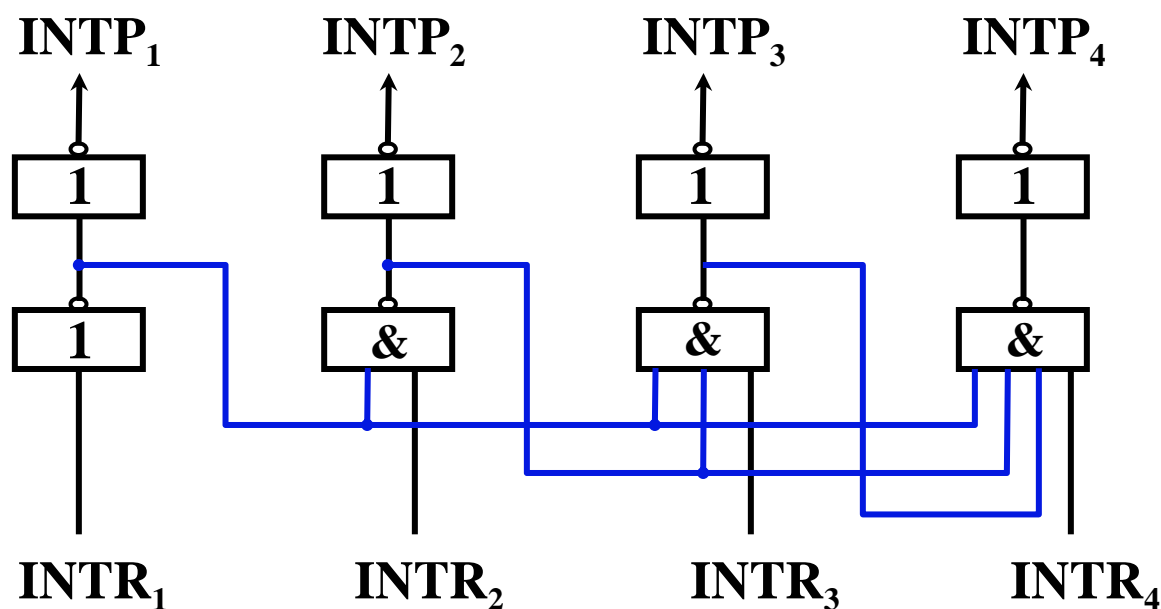
① 分散 在各个中断源的 接口电路中 链式排队器

参见 第五章

② 集中 在 CPU 内

如果有多个中断请求怎么办？

响应哪一个？

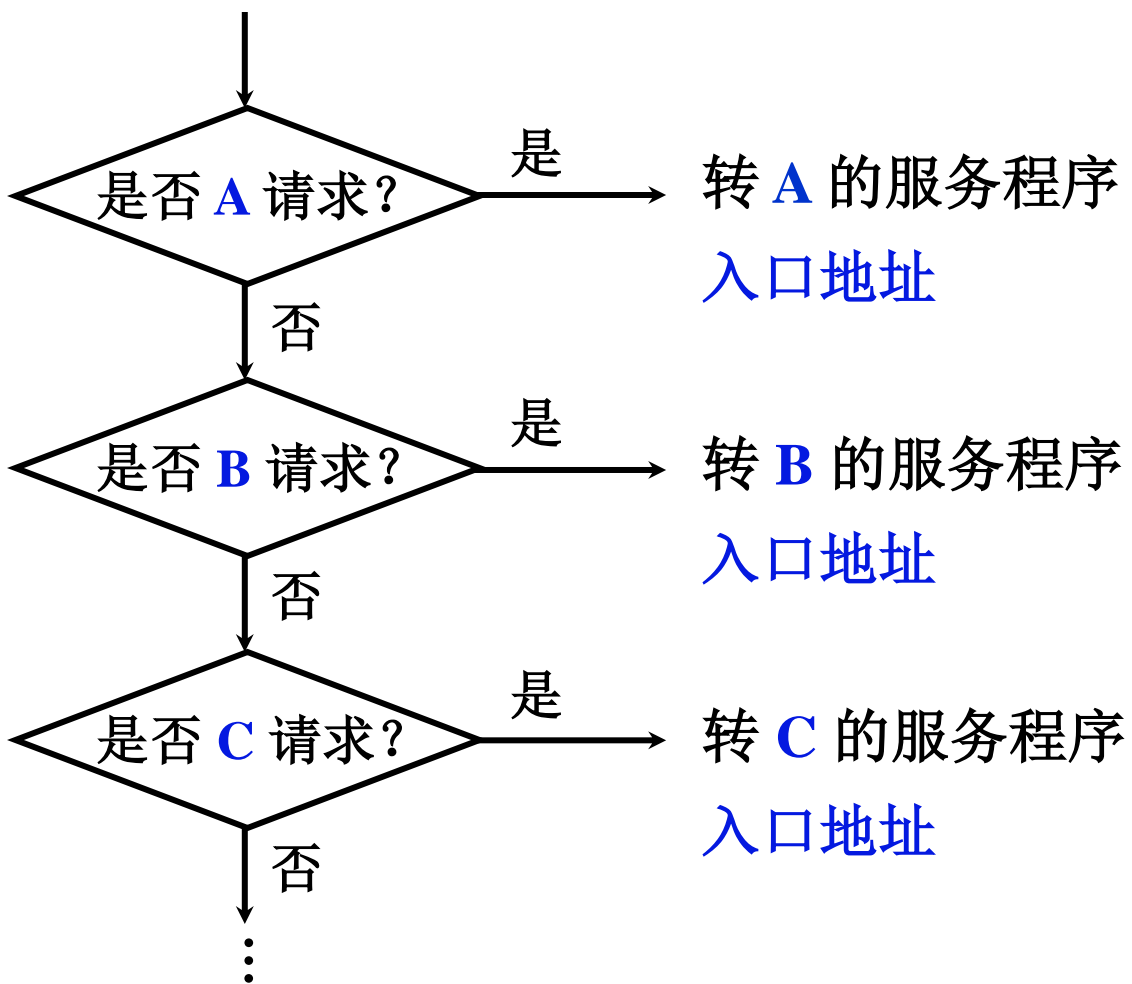


INTR₁、INTR₂、INTR₃、INTR₄ 优先级 按降序 排列

(2) 软件实现（程序查询）

8.4

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 8.4

1. 硬件向量法

如何找到中断服务程序的入口地址？

知道了要响应的中断源，才能确定要执行那个中断服务程序



向量地址 12H、13H、14H

入口地址 200、300、400

2. 软件查询法

8.4

八个中断源 1, 2, ... 8 按 降序 排列

中断识别程序 (入口地址 **M**)

用软件如何实现寻找中断服务程序的入口地址呢？

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 [#] <u>D = 1</u> 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

8.4

1. 响应中断的条件

允许中断触发器 $EINT = 1$

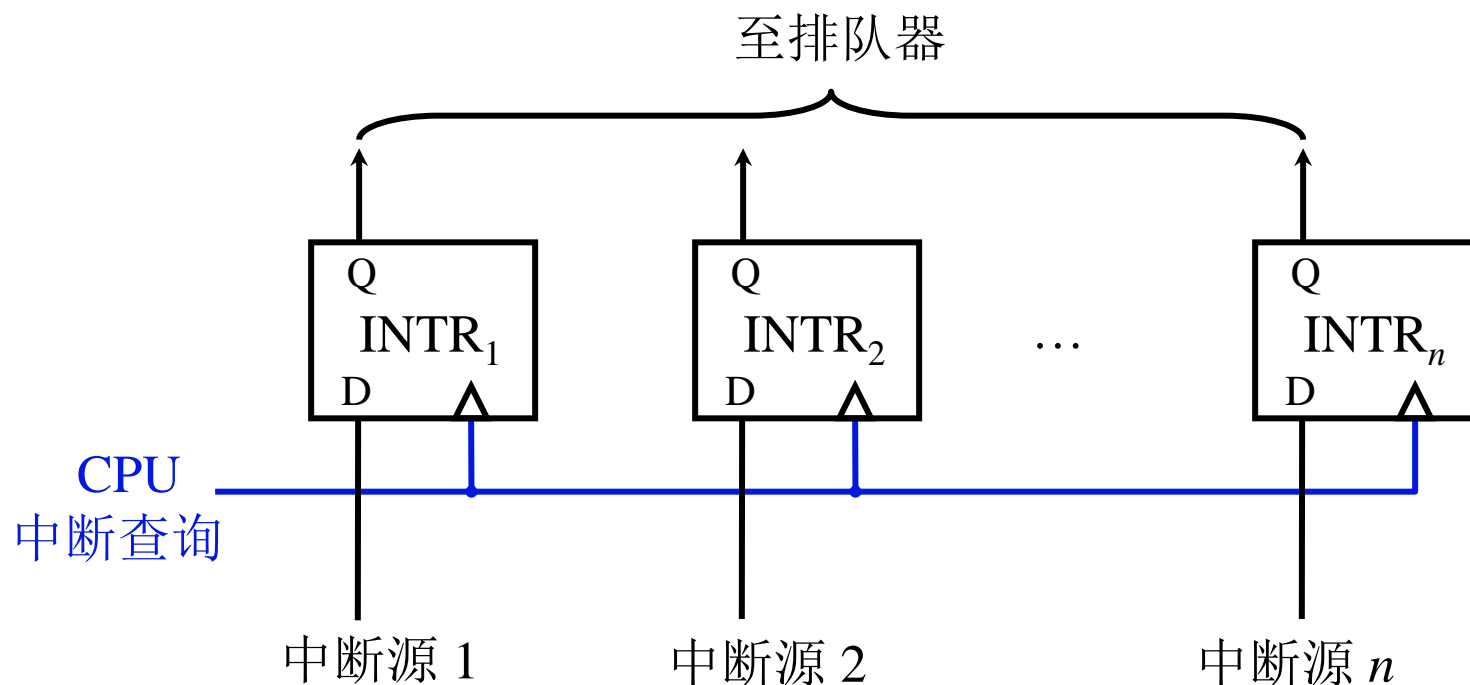
2. 响应中断的时间

指令执行周期结束时刻由CPU 发查询信号

CPU在什么时间、什么条件下响应中断？

CPU在任何条件下都要立即响应中断吗？

CPU在任何时间都能响应中断吗？



8.4

响应中断，要去执行 中断服务程序

为将来的中断返回做准备

- (1) 保护程序断点
- (2) 保护程序运行的软硬件状态

单重中断：执行中断服务程序时不允许再发生中断

多重中断：保护程序
软硬件状态的过程中，
不允许发生中断

(1) 保护程序断点

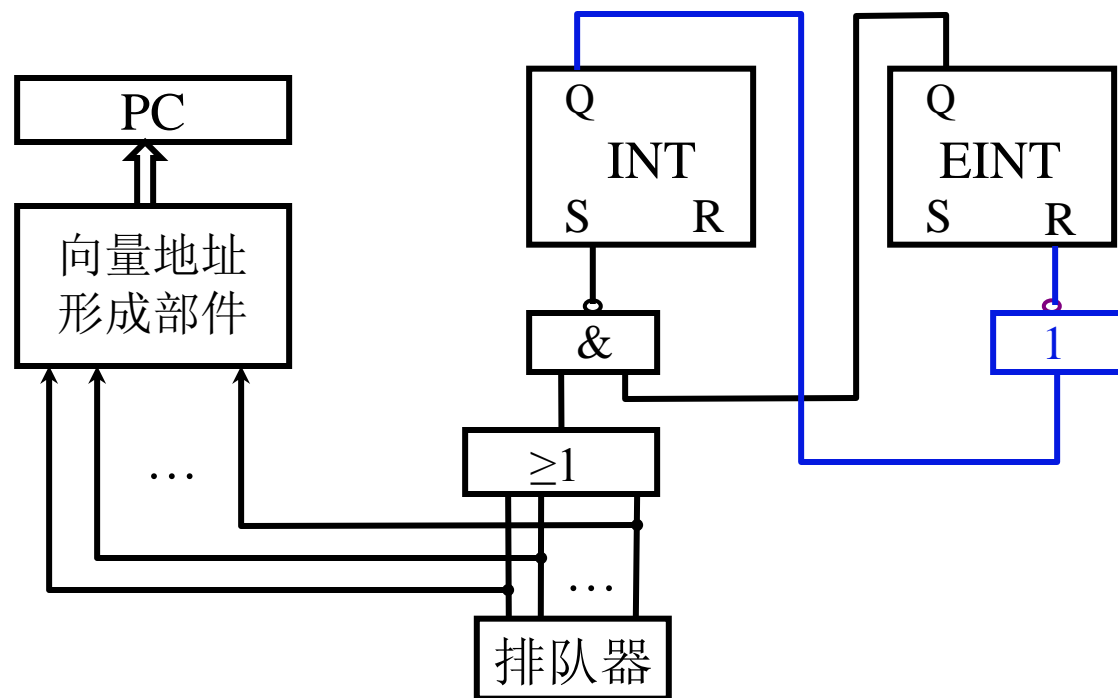
断点存于 特定地址（0 号地址） 内 断点 进栈

(2) 寻找服务程序入口地址

向量地址 \rightarrow PC (硬件向量法)

中断识别程序 入口地址 $M \rightarrow PC$ (软件查询法)

(3) 硬件 关中断



INT 中断标记

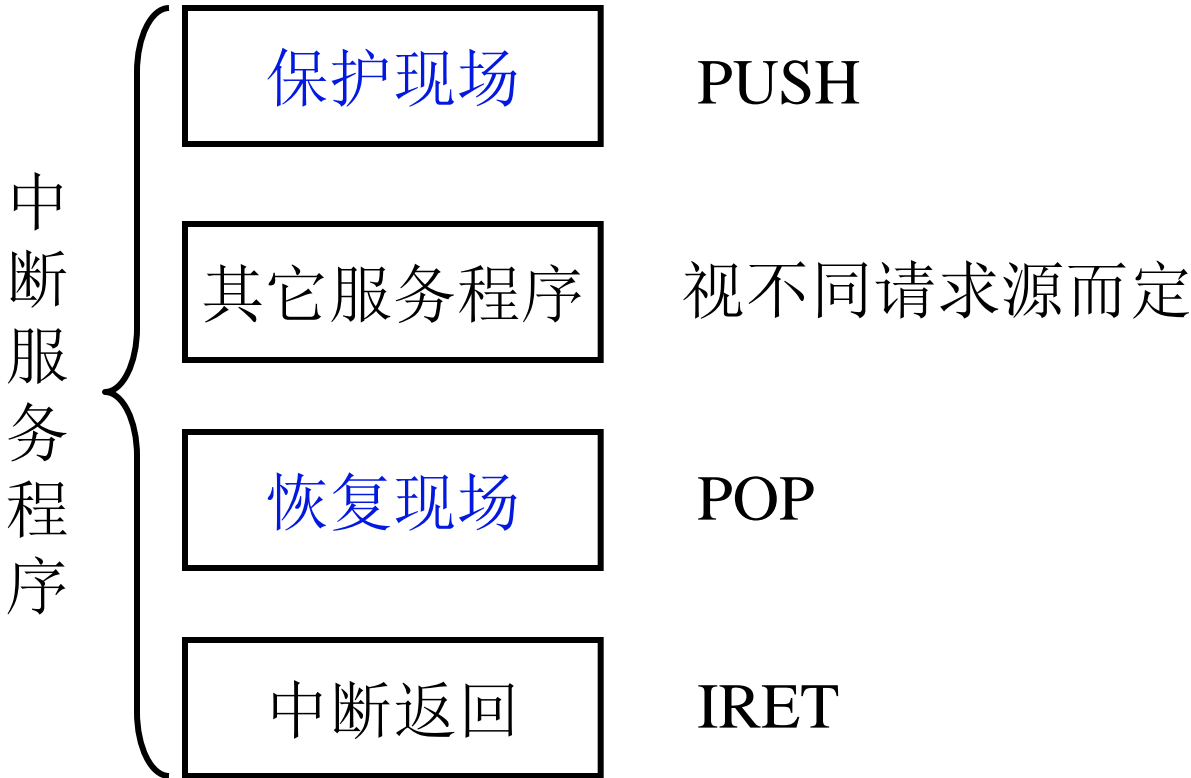
EINT 允许中断

R-S 触发器

五、保护现场和恢复现场

8.4

1. 保护现场 {
- 断点 硬件完成 中断隐指令 完成 保存断点、形成中断服务程序的入口地址、硬件关中断
- 寄存器 内容 中断服务程序 完成 软件完成
2. 恢复现场 中断服务程序 完成

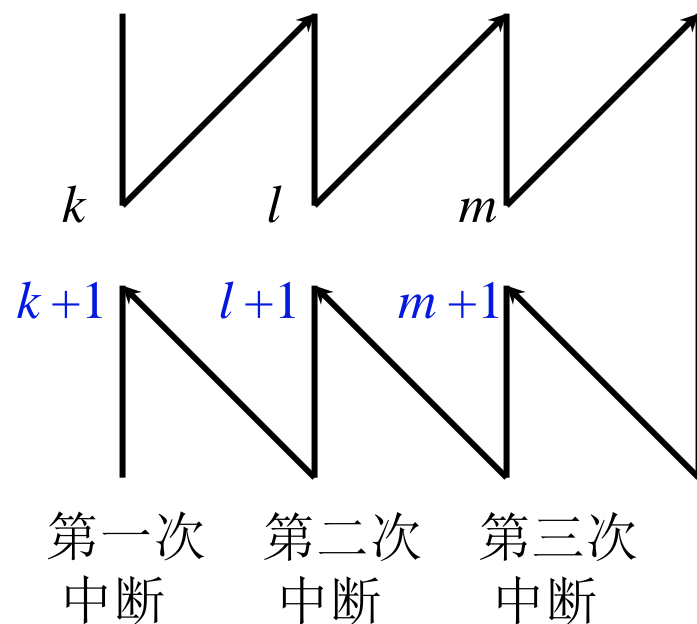


六、多重中断

8.4

1. 多重中断的概念

如果在执行中断服务程序的过程中，出现了更重要的，需要及时处理的新事件，怎么办呢？



程序断点 $k+1$, $l+1$, $m+1$

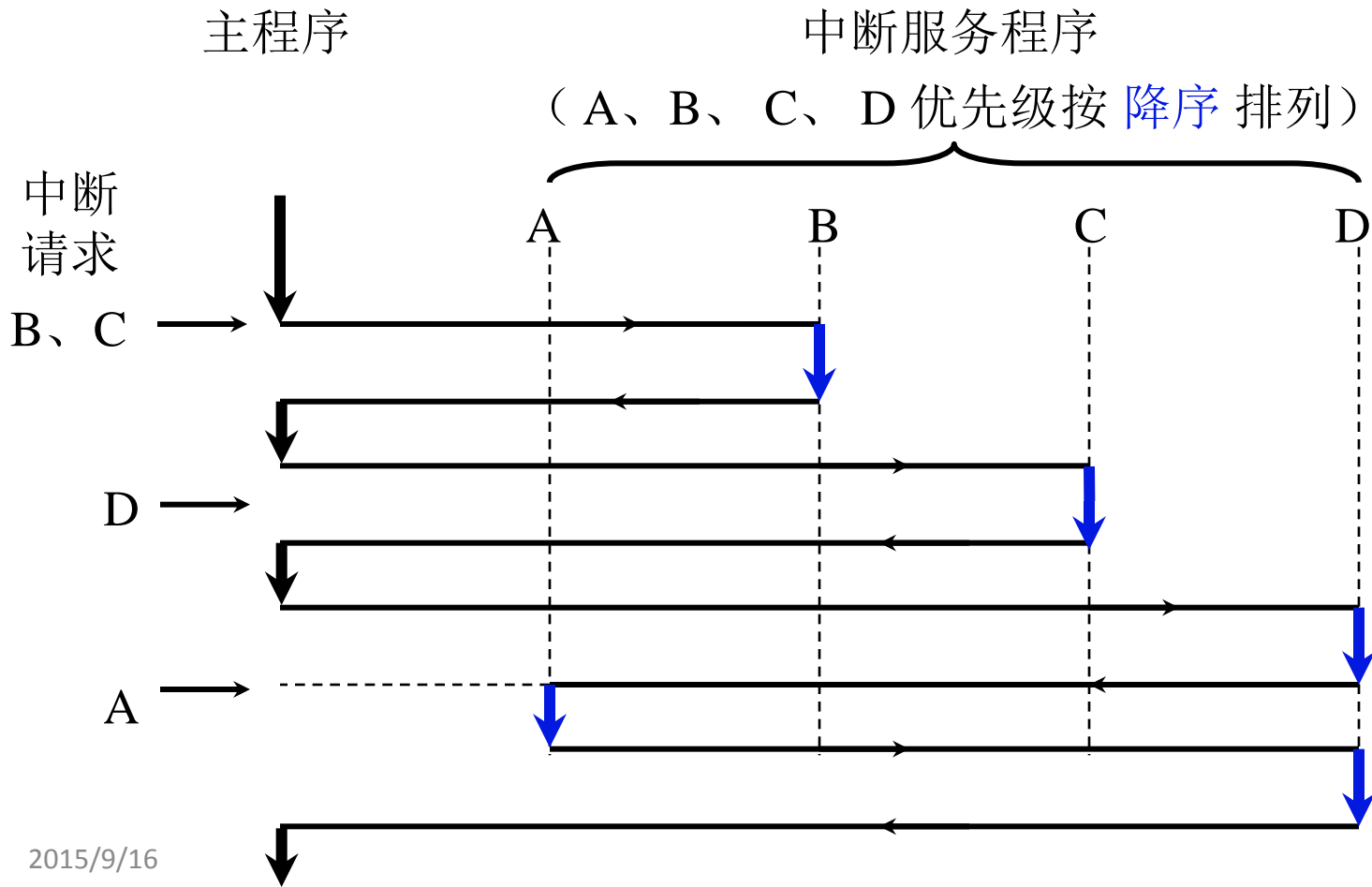
2. 实现多重中断的条件

8.4

- (1) 提前 设置 开中断 指令
- (2) 优先级级别高 的中断源 有权中断优先级级别低 的中断源

要允许CPU在执行某个中断服务程序时，响应新的中断请求

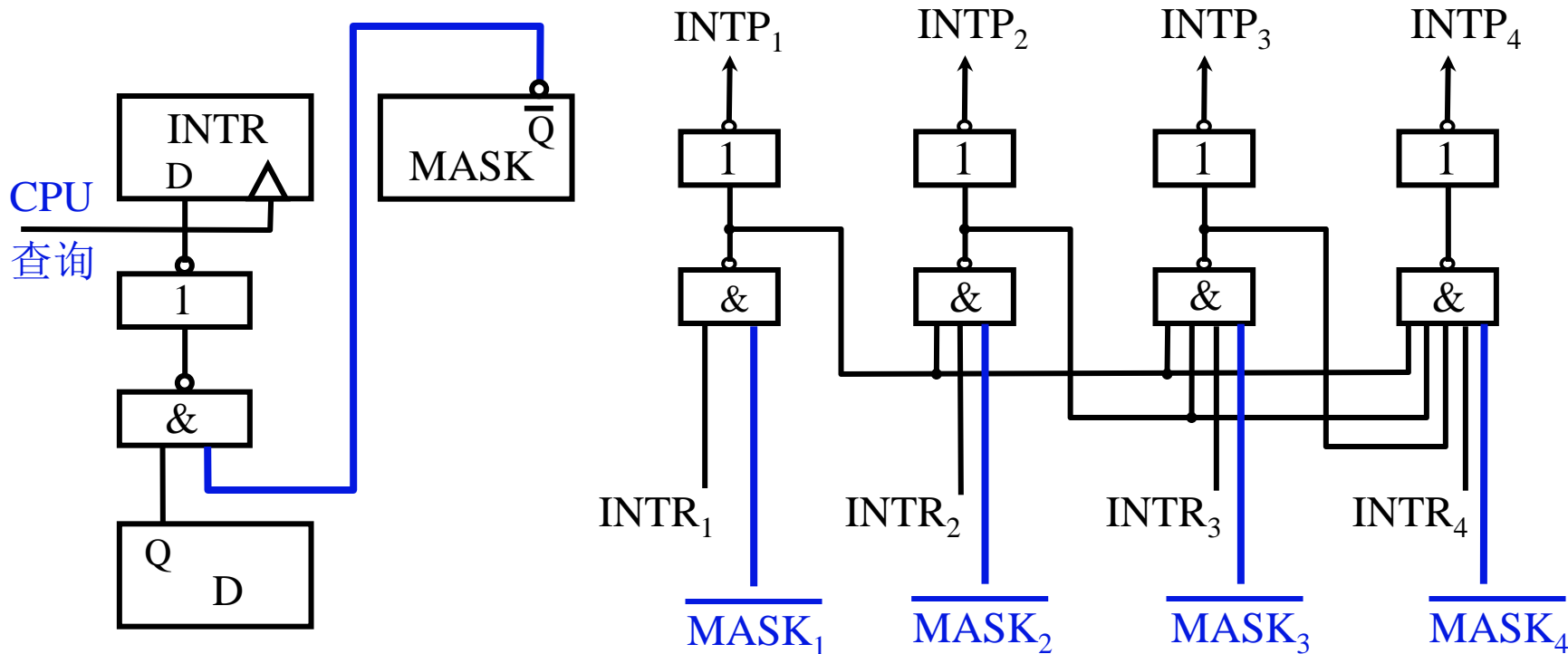
是不是任何一个新的中断请求，均能中断正在进行的 interrupt 服务？



3. 屏蔽技术

8.4

(1) 屏蔽触发器的作用



MASK = 0 (未屏蔽)

$MASK_i = 1$ (屏蔽)

INTR 能被置 “1”

$INTP_i = 0$ (不能被排队选中)

(2) 屏蔽字

8.4

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

包括一号在内，全屏蔽
只有1号开放

(3) 屏蔽技术可改变处理优先等级

8.4

响应优先级 不可改变 硬件电路确定给出，就是上面的中断屏蔽字

处理优先级 可改变（通过重新设置屏蔽字）

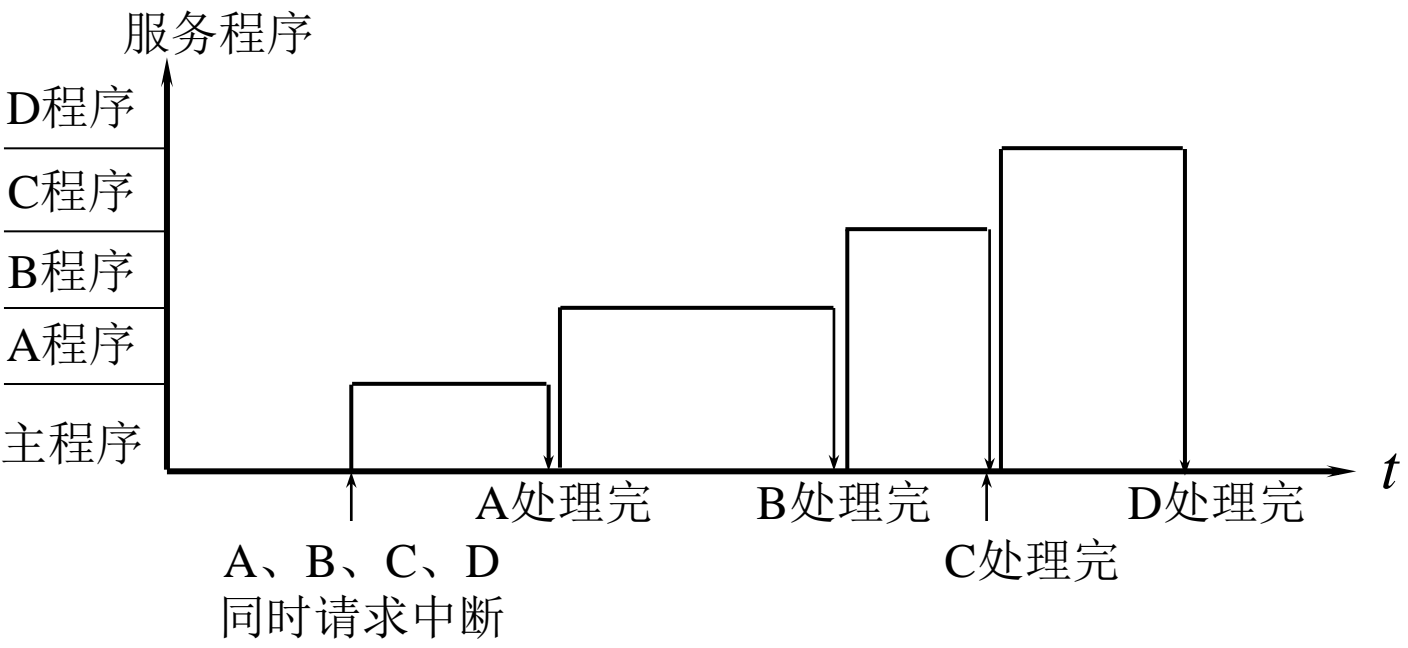
中断源	原屏蔽字	新屏蔽字
A	1 1 1 1	1 1 1 1
B	0 1 1 1	0 1 0 0
C	0 0 1 1	0 1 1 0
D	0 0 0 1	0 1 1 1

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

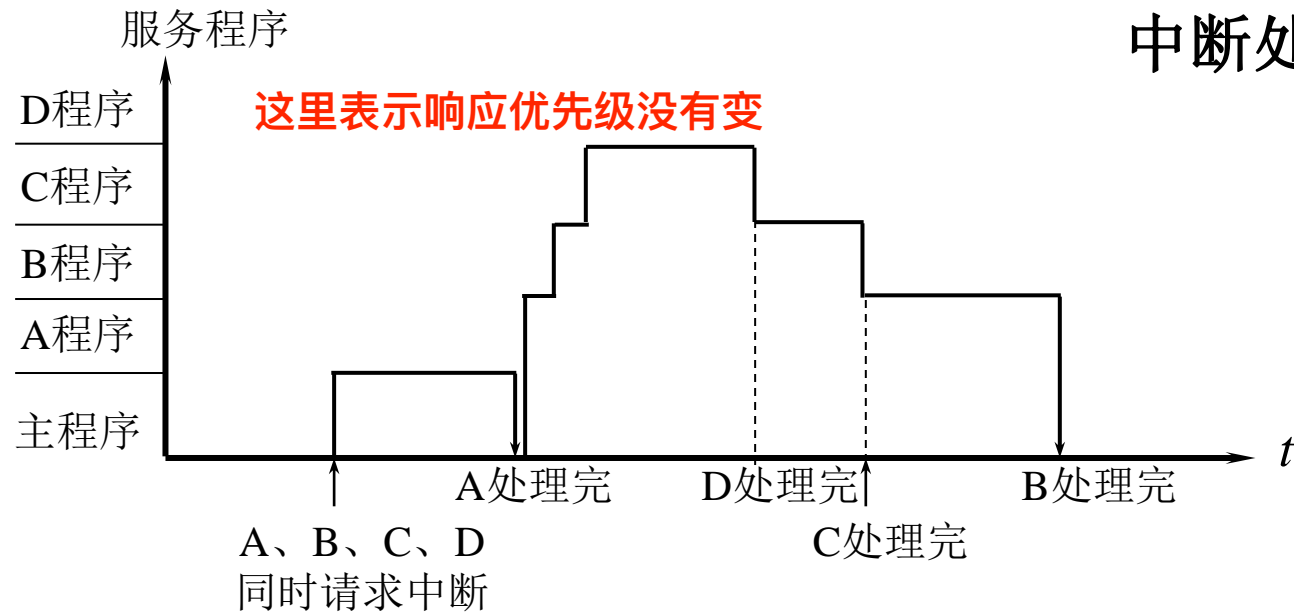
8.4



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

8.4



中断处理的优先级为A、D、C、B

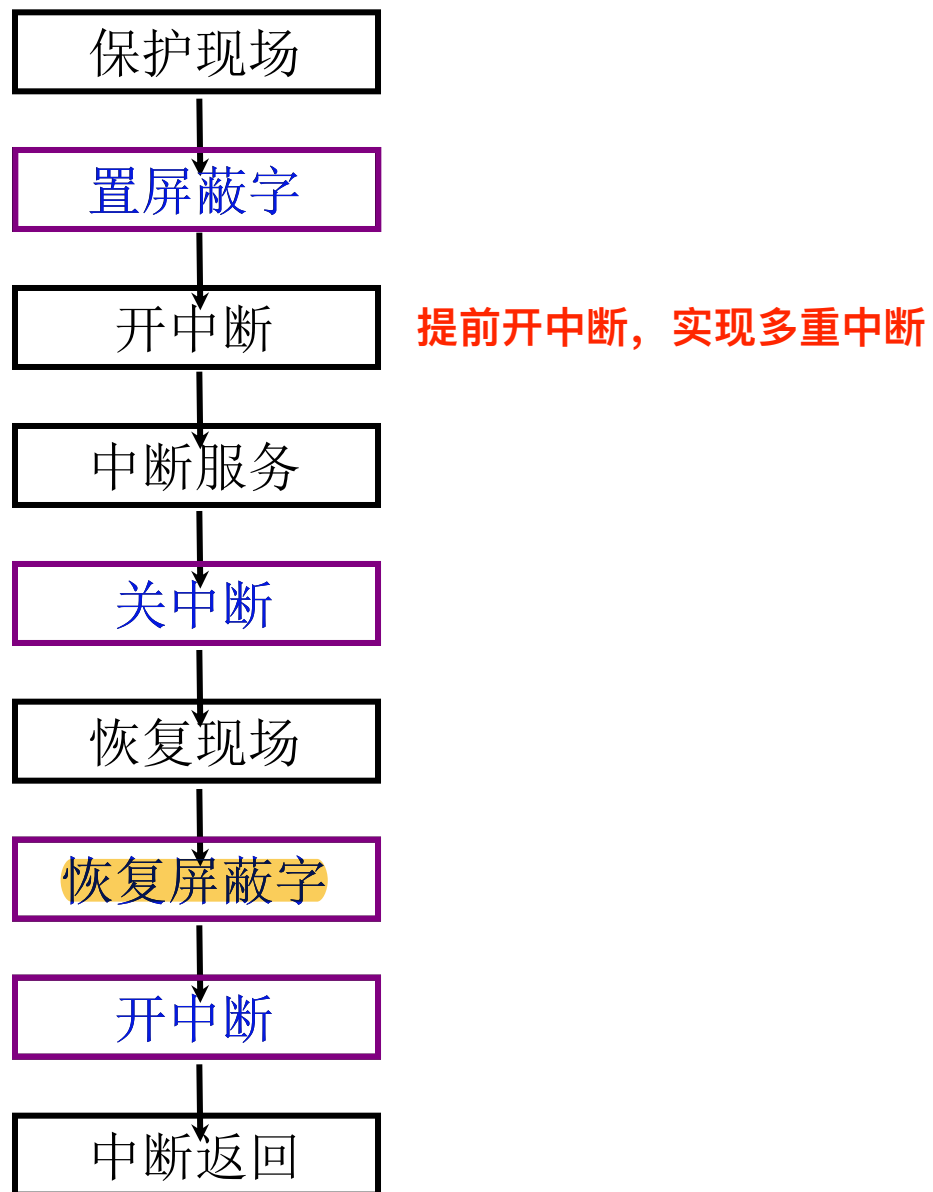
CPU 执行程序轨迹（新屏蔽字）

(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

(5) 新屏蔽字的设置

8.4



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
 不一定是物理地址的0

中断周期 0 \rightarrow MAR

命令存储器写

PC \rightarrow MDR 断点 \rightarrow MDR

(MDR) \rightarrow 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8.4

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
	⋮	
	LDA 0	} 0 地址内容转存
<u>置屏蔽字</u>	STA RETURN	
	ENI	开中断
	⋮	} 其他服务内容
	LDA SAVE	
	JMP @ RETURN	间址返回
SAVE	× × × ×	存放 ACC 内容
RETURN	× × × ×	转存 0 地址内容

该内容就是程序断点