**11.1 CPU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

(2) 程序性事故: 溢出、操作码不能识别、除法非法、越界、越权、缺页

(3) 硬件故障：电源掉电、硬件线路故障等

(4) I/O 设备：发出中断请求

(5) 外部事件: 用键盘中断现行程序

**二、中断请求标记和中断判优逻辑**

1. 中断请求标记

中断请求标记触发器 INTR：标记一个中断请求源的请求信号

中断请求标记寄存器：多个INTR组成

**中断请求触发器越多，说明计算机处理中断的能力越强。**

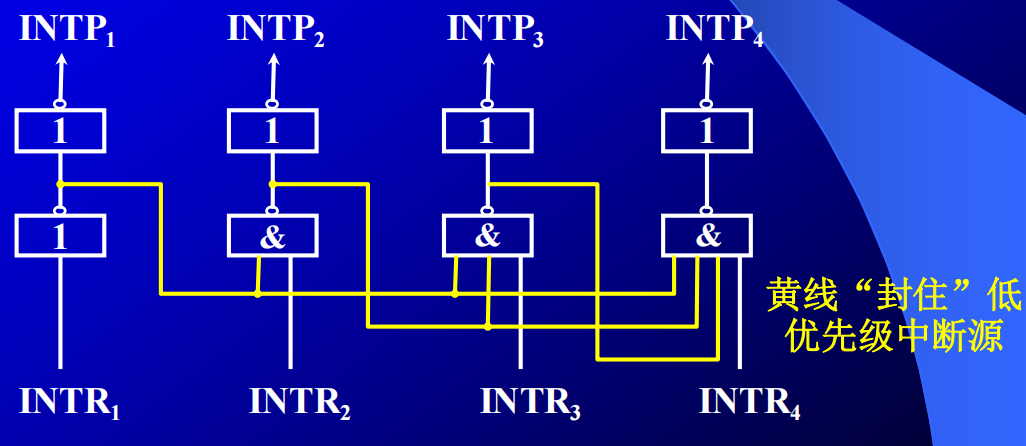
这些触发器既可以集中在CPU的中断系统内，也可以分散在各个中断源的接口电路中

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链式排队器

②集中在CPU内的排队器：



(2) 软件实现（程序查询）

**三、中断服务程序入口地址的寻找**

**1. 硬件向量法**

两种方法



优势：寻找入口地址速度快

**2. 软件查询法**

各中断源对应的入口地址，则**由程序员（或系统）事先确定**。

这种方法不涉及硬件设备，更方便、灵活，但查询时间较长。

**四、中断响应**

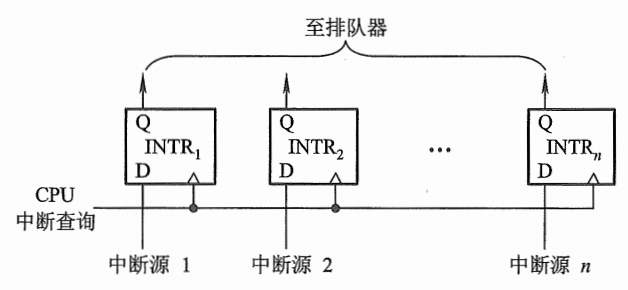
**1. 响应中断的条件**

允许中断触发器 EINT = 1 （EINT可被开中断指令置1，也可被关中断指令置0）

**2. 响应中断的时间**

（规定CPU运行为取指周期+执行周期 (+中断周期)）

CPU 在执行周期的结束时刻统一向所有中断源发中断查询信号（发送至INTR的时钟端），因此CPU**总是在指令执行周期结束后，响应任何中断源的请求**



如果有些指令执行时间很长，也可在**指令执行过程中设置若干个查询断点**， CPU 在每个“查询断点”时刻均发中断查询信号，以便发现有中断请求便可及时响应。

**3. 中断隐指令**

(1) 保护程序断点：将当前程序计数器 PC 的内容存入 存储器的特定单元（如0号地址）内，也可以存入堆栈。

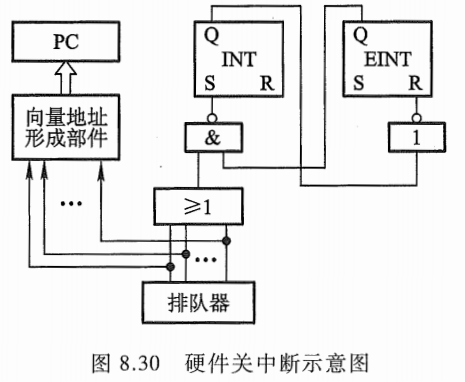
(2) 寻找中断服务程序的入口地址

将向量地址送至PC（硬件向量法） / 将中断识别程序的地址送至PC（软件查询法）

(3) 关中断

中断周期内必须自动关中断，以禁止 CPU 再次响应新的中断请求

硬件关中断：利用中断标记触发器INT关闭中断允许触发器EINT



**五、保护现场和恢复现场**



保存通用寄存器和状态寄存器的内容（到存储器）

**六、中断屏蔽技术**

**1. 多重中断的概念**

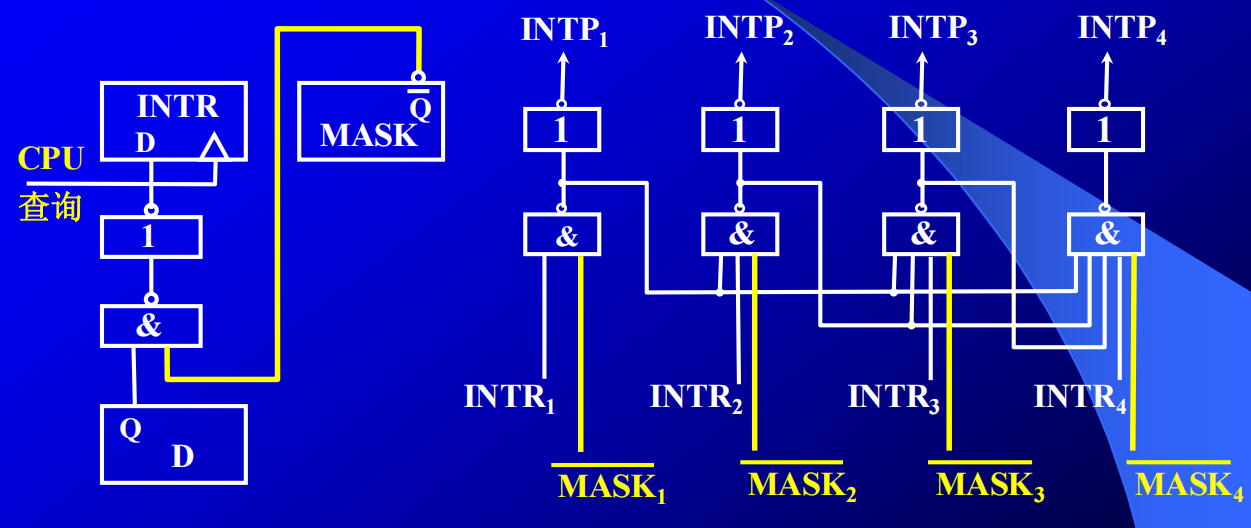
允许级别更高的中断源中断现行的中断服务程序

**2. 实现多重中断的条件**

(1) **提前设置开中断指令**：进入中断周期后，中断隐指令自动关中断，需要中断服务程序将EINT置1，重新开中断。

(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**

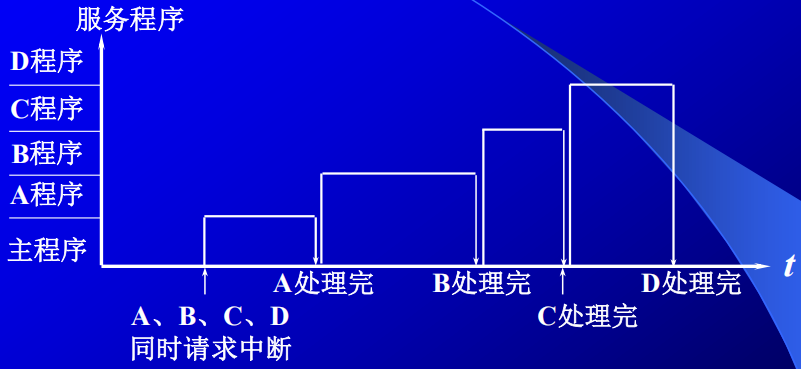


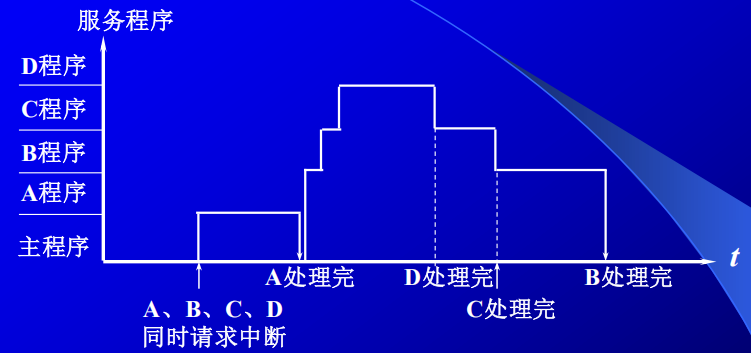
**优先级包括**

响应优先级：硬件设置好，不可改动

处理优先级：通过重新设置屏蔽字来改动（不重设则等于响应优先级）



对于同时发起的请求，会按照响应优先级依次响应。

对于在中断服务A中发起的请求B，（多重中断开启了EINT）如果B处理优先级高则先执行B，在继续执行A；如果B优先级低，则B也不会被忽略，执行完A后再执行

**4.** **多重中断的断点保护**

(1) 断点进栈：中断隐指令完成

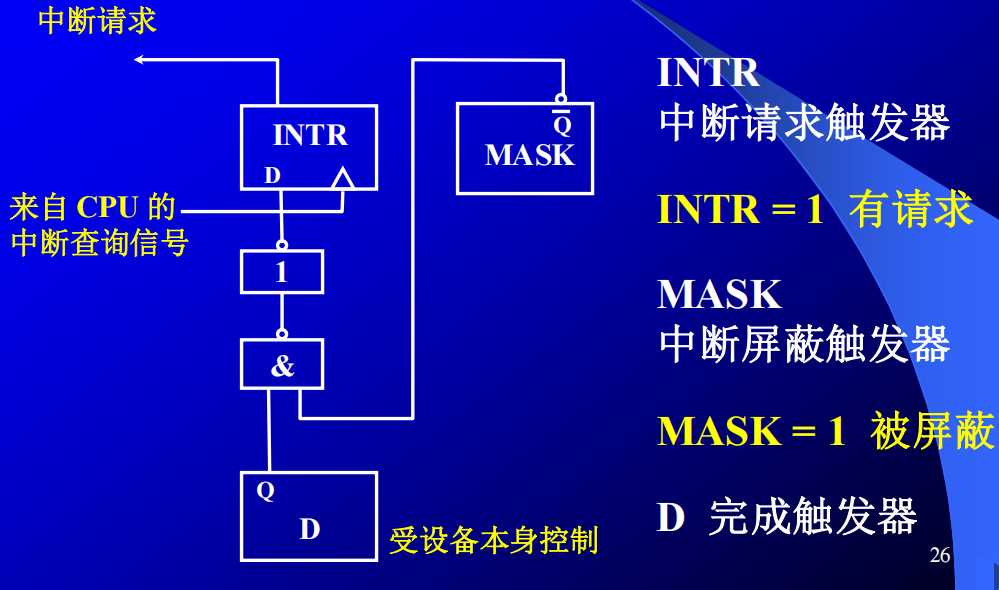
(2) 断点存入主存“ 0 ”地址：中断隐指令完成，但对于多重中断，必须在每次中断服务程序的开中断指令前将0 地址单元的内容转存到其他地址单元中。



**11.2** **I/O中断**

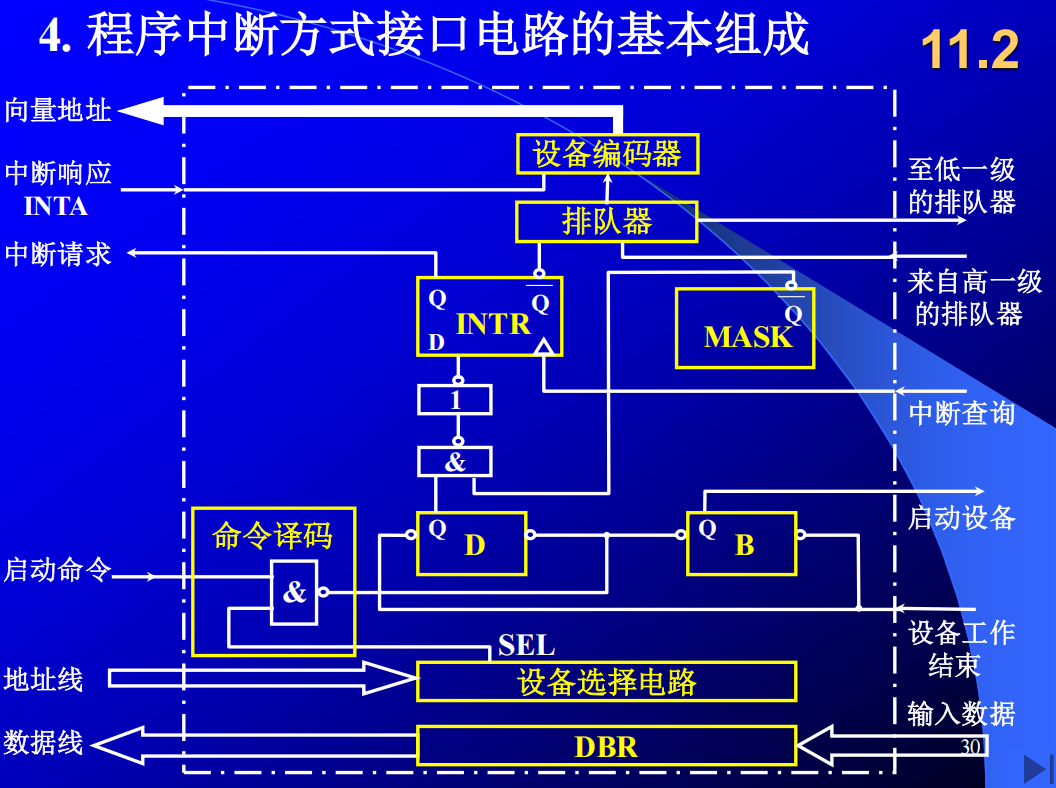
CPU启动I/O设备后继续执行现行程序，只有当I/O设备准备就绪向 CPU 提出请求后，再暂时中断 CPU 现行程序转入I/O服务程序，这便产生了I/O中断

**程序中断方式的接口电路**



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



响应优先级：就I/O中断而言，速度越高的I/O设备，优先级越高，因为若 CPU 不及时响应高速I/O的请求，其信息可能会立即丢失

**中断服务程序流程**

1. 保护现场

程序断点的保护：断隐指令完成

寄存器内容的保护：进栈指令，保存通用寄存器和状态寄存器内容

(2) 中断服务

(3) 恢复现场

出栈指令，恢复各个寄存器的值

(4) 中断返回

返回程序断点

