



微机原理和接口技术

第十一讲 中断系统2



提 纲

1. 中断系统概述

2. 8051微控制器的中断系统

3. 中断处理过程

4. 中断程序设计

5. IO端口扩展外部中断源

提 纲

2. 8051微控制器的中断系统

51的中断系统

3.中断允许控制寄存器IE（Interrupt Enable）

中断允许控制寄存器 IE的字节地址为A8H，是可位寻址的SFR。IE用于管理各中断源中断的允许与禁止。

	7	6	5	4	3	2	1	0
位 符	EA	—	—	ES	ET1	EX1	ET0	EX0
英 文	Enable			Enable	Enable	Enable	Enable	Enable
注 释	All interrupts			Serial interrupt	Timer1 interrupt	External 1 interrupt	Timer0 interrupt	External 0 interrupt

微控制器复位后，IE中各位均被清0,即禁止所有中断。

51的中断系统

3.中断允许控制寄存器IE (Interrupt Enable)

➤ **EA**: CPU中断允许位。

EA=1, CPU开中断, 结合各中断源的中断允许位, 确定各中断源的允许和禁止。

EA=0, CPU关中断, 禁止响应任何中断请求。

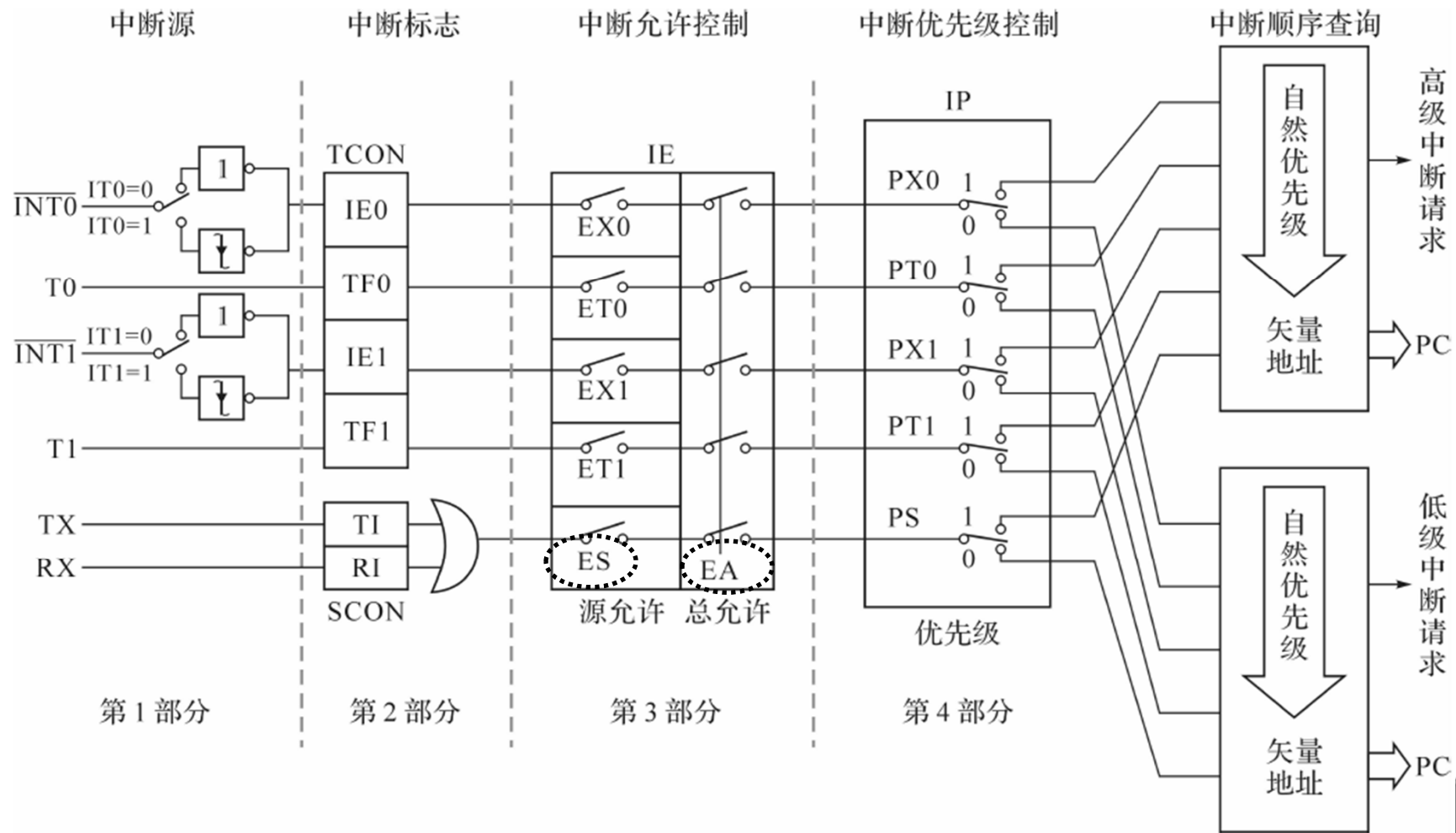
➤ **ES**: 串行口中断允许位。

ES=1, 允许串行口的接收和发送中断; ES=0, 禁止串行口中断。

	7	6	5	4	3	2	1	0
位 符	EA	—	—	ES	ET1	EX1	ET0	EX0
英 文	Enable			Enable	Enable	Enable	Enable	Enable
注 释	All interrupts			Serial interrupt	Timer1 interrupt	External 1 interrupt	Timer0 interrupt	External 0 interrupt

51的中断系统

EA和ES在哪里？



51的中断系统

3.中断允许控制寄存器IE (Interrupt Enable)

➤ **ET1/ET0**: T1/T0中断允许位。

ET1/ET0=1, 允许T1/T0中断; ET1/ET0=0, 禁止T1/T0中断。

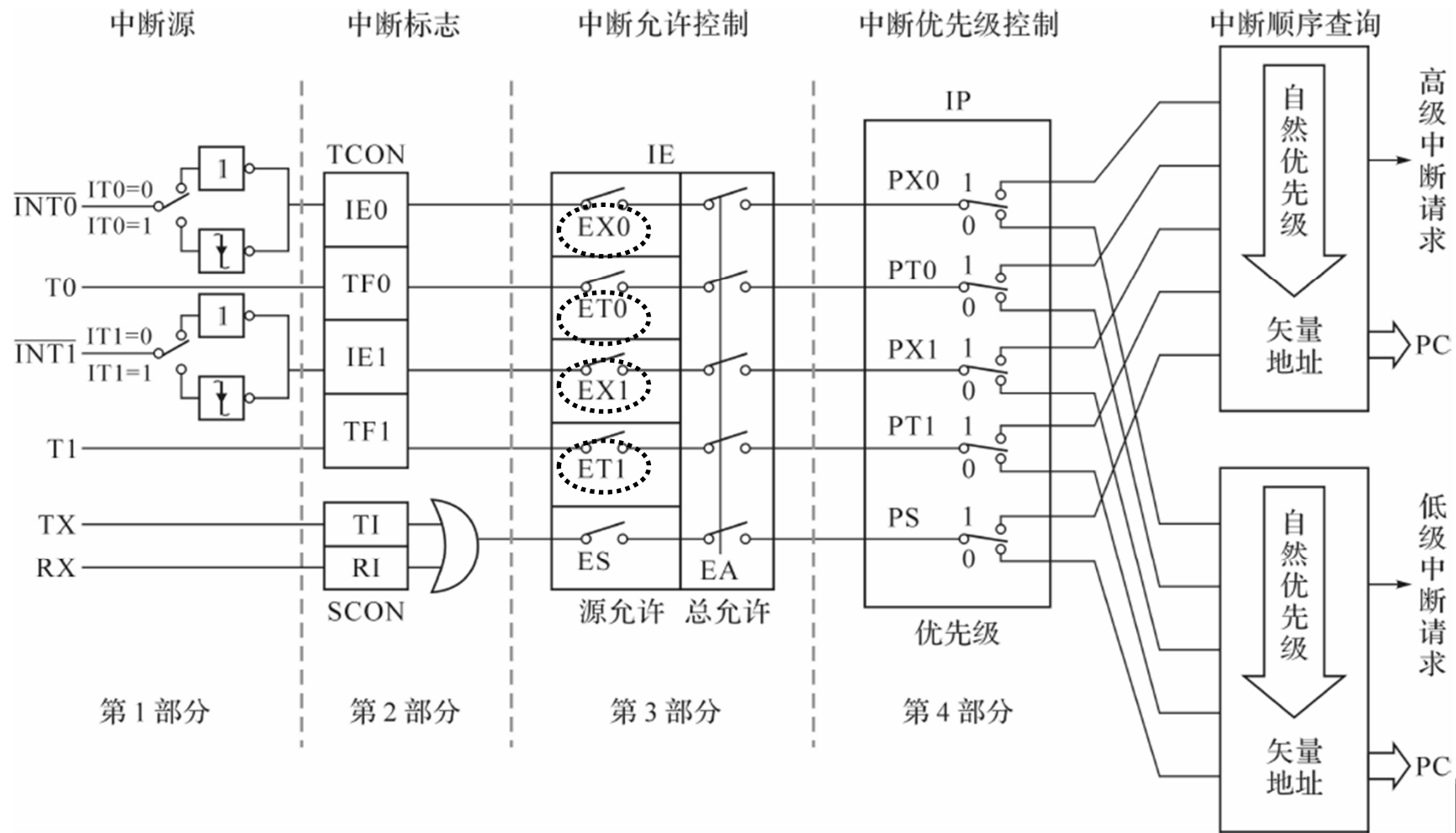
➤ **EX1/EX0**: INT1/INT0中断允许位。

INT1/INT0=1, 允许INT1/INT0中断; INT1/INT0=0, 禁止INT1/INT0中断。

	7	6	5	4	3	2	1	0
位 符	EA	—	—	ES	ET1	EX1	ET0	EX0
英 文	Enable			Enable	Enable	Enable	Enable	Enable
注 释	All interrupts			Serial interrupt	Timer1 interrupt	External 1 interrupt	Timer0 interrupt	External 0 interrupt

51的中断系统

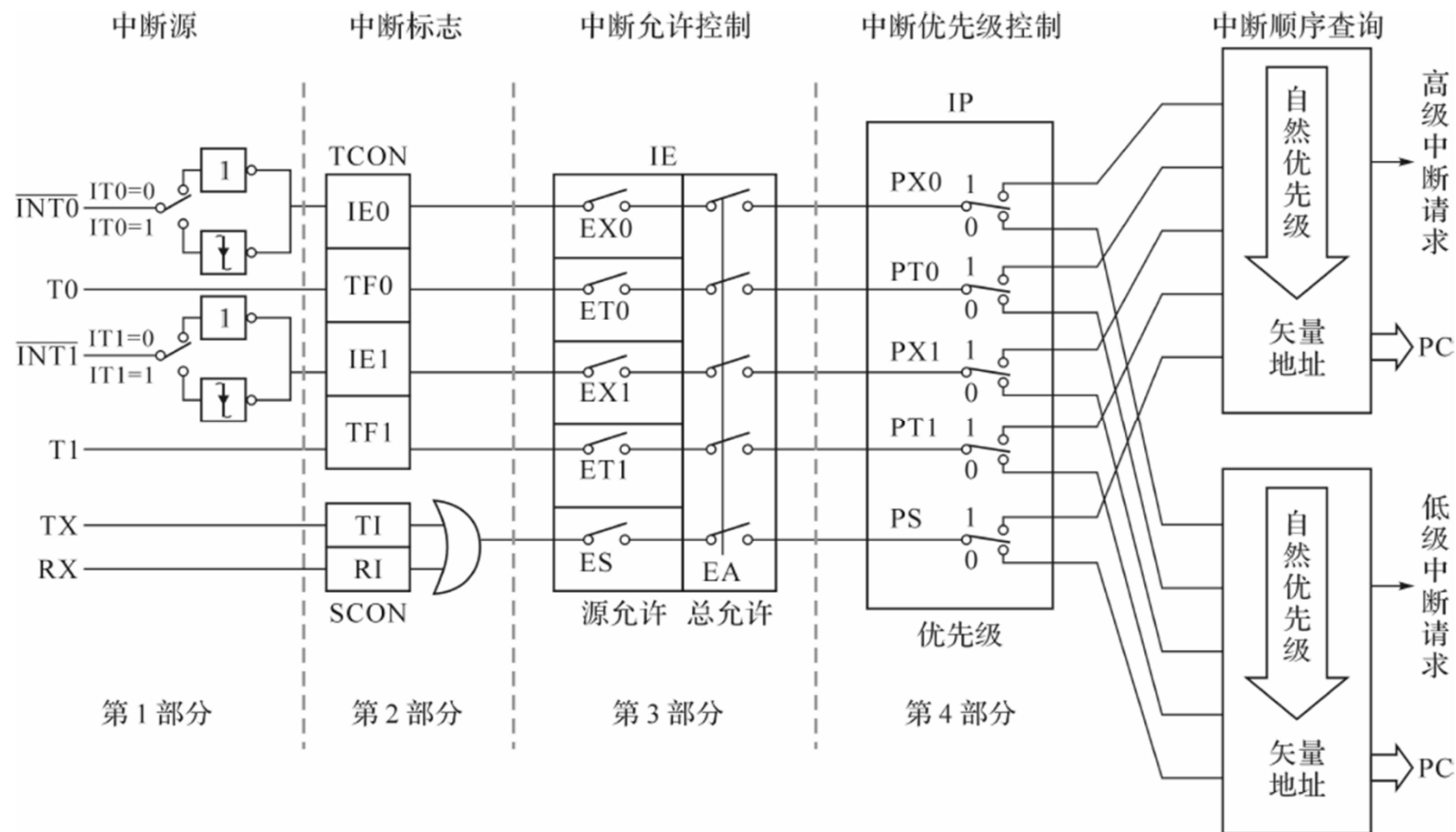
EX0、EX1和ET0、ET1在哪里？



51的中断系统

通过对IE的设置，实现对各中断源中断允许和禁止的控制。

8051微控制器能实行二级控制，EA是总控制位，各中断源有分控制位。只有当总控制位EA=1，即CPU中断开放时，对各分控制位的设置（开放或禁止相应中断源）才有效。



51的中断系统

4.中断优先级寄存器IP (Interrupt Priority)

字节地址为B8H，可位寻址。用于管理各中断源的优先级别。

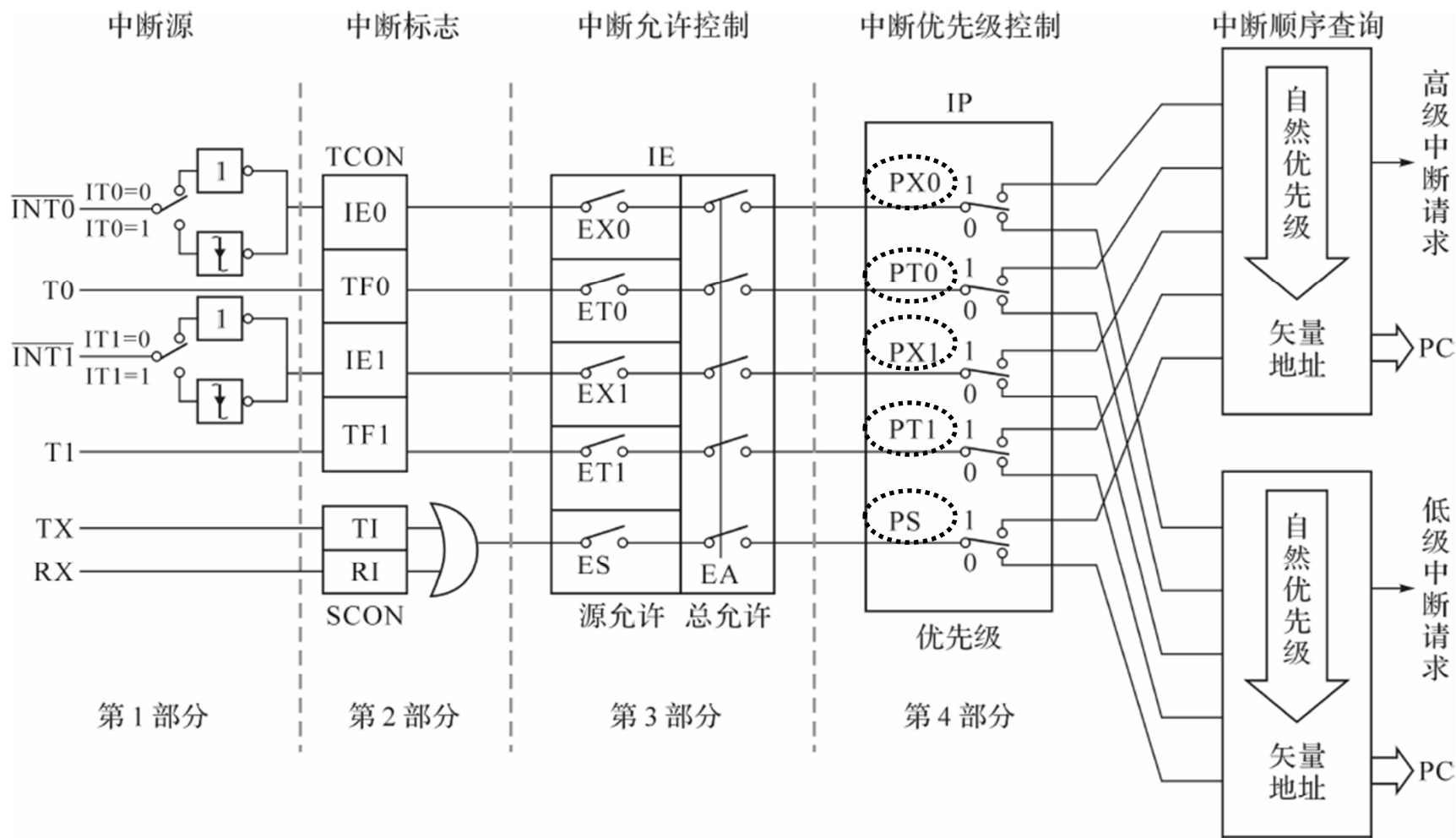
	7	6	5	4	3	2	1	0
位符号	—	—	—	PS	PT1	PX1	PT0	PX0
英文注释				Serial Interrupt Priority	Timer1 Interrupt Priority	External 1 Interrupt Priority	Timer0 Interrupt Priority	External 0 Interrupt Priority

微控制器复位后，IP内容为0，所有中断源均被设置为低优先级中断。

位符号	功能说明
PS	串行口中断优先级控制位。PS=1,选择高优先级;PS=0,选择低优先级
PT1	T1 中断优先级控制位。PT1=1,选择高优先级;PT1=0,选择低优先级
PX1	$\overline{\text{INT1}}$ 中断优先级控制位。PX1=1,选择高优先级;PX1=0,选择低优先级
PT0	T0 中断优先级控制位。PT0=1,选择高优先级;PT0=0,选择低优先级
PX0	$\overline{\text{INT0}}$ 中断优先级控制位。PX0=1,选择高优先级;PX0=0,选择低优先级

51的中断系统

优先级控制位都在哪里？



51的中断系统

4.中断优先级寄存器IP (Interrupt Priority)

例：若一个微控制器系统有2个中断源，一个是T0的10ms定时中断，一个是外部INT0的按键中断，希望T0中断在任何时刻能及时响应，则可以设置T0中断为高优先级中断，INT0中断为低优先级中断。

即 SETB PT0
 CLR PX0

这样T0中断能打断INT0的中断服务，反之则不能。


对IP编程可把5个中断设置为高低两个优先级，它们遵循：

- 低级中断能被高级中断打断,但不能被同级中断打断;
- 高级中断不能被任何中断打断，一定要返回主程序并再执行一条指令后，才能响应新的中断请求。

51的中断系统

4.中断优先级寄存器IP (Interrupt Priority)

当CPU同时接收到几个同优先级的中断请求时，哪一个先得到响应，取决于CPU中断系统内部的查询顺序。这相当于在每个优先级内，还存在一个辅助优先级结构，其优先顺序如下：

中断源	中断优先级
1. 外部中断0	最高
2. 定时器T0中断	
3. 外部中断1	
4. 定时器T1中断	
5. 串行口中断	
	最低

提 纲

3. 中断处理过程



51中断的处理过程

中断响应的自主操作：在中断检测和中断响应过程中，由中断系统硬件自动完成的操作。

1. 中断请求的自动查询

8051微控制器中，中断系统在每个机器周期的S6状态查询各中断源是否有请求（即相应的中断标志是否为1），并按优先级管理规则处理同时请求的中断源，且在下一个机器周期的S1状态响应最高级中断请求。

有以下情况则除外：

- CPU正在处理相同或更高优先级中断；
- 正在执行多机器周期指令，并还未执行到最后一个机器周期；
- 正在执行RETI 指令或读/写IE、IP的指令，则要延后一条指令予以响应。



51中断的处理过程

2.中断响应时的自主操作

- 置位相应的“优先级标志”，以标明所响应中断的优先级别；
- 中断源标志清0（TI、RI除外）；
- 中断断点地址装入堆栈保护（不保护寄存器等）；
- 中断入口地址装入PC，以便使程序转到中断入口地址处。

3.中断返回时的自主操作

- “优先级标志”清0；
- 断点地址送入PC，以便使程序返回到断点处。



51中断的处理过程

中断响应：中断源发出中断请求、在满足CPU中断响应条件之后，CPU处理中断请求的过程。

中断响应的基本条件：

- 1) 有中断源发出中断请求；
- 2) CPU中断允许位（总允许）置位，即 $EA=1$ ；
- 3) 申请中断的中断源的中断允许位（源允许）置位，即允许该中断源中断。

CPU在每个机器周期，按优先次序查询各中断标志，找到所有有效的中断请求，并对其优先级排队，如果满足：

- 4) 无同级或高级中断正在服务；
- 5) 现行指令已执行完毕；
- 6) 若执行指令为RETI或是读/写IE或IP指令时，则该指令的下一条指令也执行完毕。

CPU便在紧接着的下一个机器周期响应中断，否则将丢弃中断查询结果。



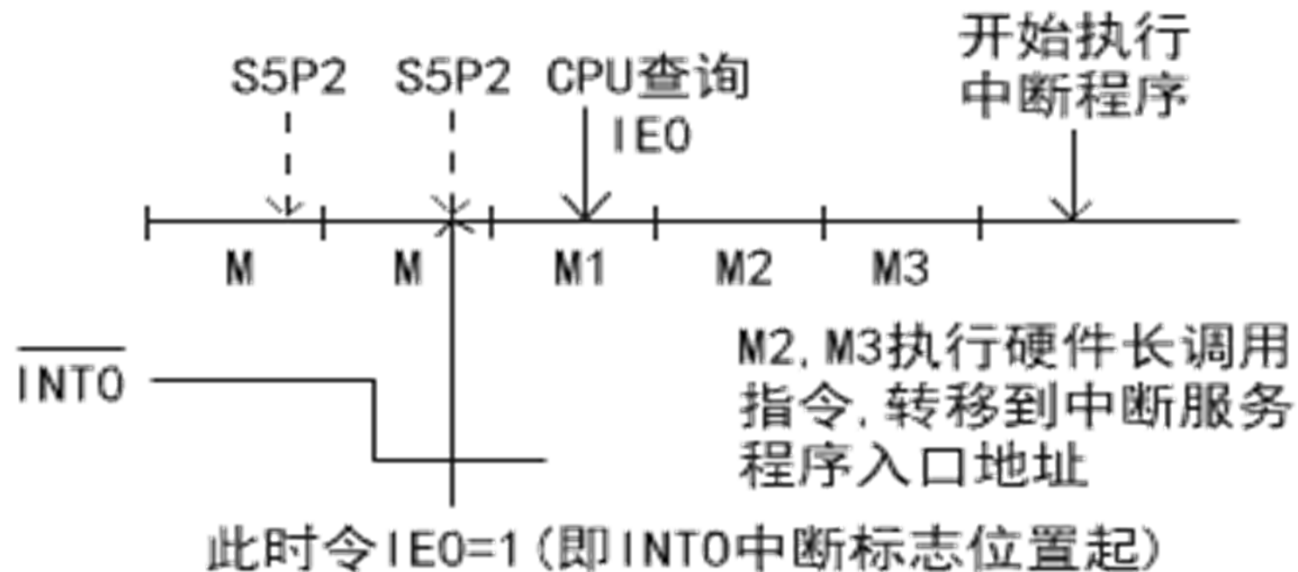
51中断的处理过程

中断响应过程：

- 1) CPU在每个机器周期检测中断源，并按优先级别和自然顺序查询各中断标志。若查询到有效的中断标志（中断标志为1），按优先级别进行处理，即响应中断；
- 2) 自动设置“优先级标志”为1，即指出CPU当前正在处理的中断优先级，以阻断同级或低级中断请求；
- 3) 自动清除中断标志（TI和RI除外）；
- 4) 自动保护断点，即将现行PC值（即断点地址）压入堆栈，并根据中断源把相应的中断程序入口地址装入PC；
- 5) 执行中断服务程序，直至遇到RETI指令为止；
- 6) RETI指令清除“优先级标志”；从堆栈中弹出断点地址给PC，使CPU回到中断处，继续执行主程序。

51中断的处理过程

中断的响应时间：最短为3个机器周期，最长为8个机器周期。



3个机器周期情况：查询中断标志位占1个机器周期，硬件自动保护断点地址（相当于自动插入一条长调用LCALL指令）需要2个机器周期。



51中断的处理过程

8个机器周期情况：如果检测到中断标志位时，CPU正在执行RETI指令或访问IE、IP的指令（2个机器周期），则执行该指令后，还必须再执行一条指令才能响应中断。若再执行的一条指令恰好为乘法或除法指令（4个机器周期），再加上自动保护断点地址的2个机器周期，总共需要8个机器周期。

如果存在多个中断源，而且CPU正在处理高级或同级中断，那么中断响应的时间还取决于正在执行的中断服务程序的长短。



51中断的处理过程

1.响应中断与调用子程序的相同点

- 都是中断当前正在执行的程序，转去执行子程序或中断服务程序。
- 都是由硬件自动把断点地址压入堆栈。
- 子程序和中断服务程序的现场保护和恢复，都需要编写程序实现。执行中断程序和子程序的返回指令时，都会自动返回到断点处，继续原程序的执行。
- 都可以实现嵌套。中断程序可以实现2级嵌套，子程序可以更多级的嵌套。

2.响应中断与调用子程序的差别

- 中断请求是随机的，在程序执行的任何时刻都有可能发生；而子程序的调用是由程序设计安排的。
- 响应中断后，转去执行存放在相应中断入口地址处的中断服务程序，而子程序的地址由程序设计时安排的。
- 中断响应是受控的，其响应时间会受一些因素影响；子程序响应时间是固定的。
- 中断服务程序的返回指令是RETI，子程序的返回指令是RET，两者不能互换。



51中断的处理过程

例：假设程序中有2个中断源：INT0和T1，要求INT0为高优先级。

主程序：

```
ORG 0000H
LJMP MAIN ; 跳转到主程序
ORG 0003H ;INT0中断入口地址
LJMP INT0SUB ;跳转到实际INT0中断服务程序存放空间
.....
ORG 001BH ; T1中断入口地址
LJMP T1SUB ;跳转到实际T1中断服务程序存放空间
.....
ORG 0030H ; 实际主程序存放区
MAIN: MOV SP,#5FH ; 设置堆栈区
      :
      SETB IT0 ; 选择INT0为下降沿触发方式
      SETB EA ; CPU开中断
      SETB EX0 ; INT0开中断
      SETB ET1 ; T1开中断
      SETB PX0 ; 设置INT0为高优先级
      .....
      SJMP $ ; 模拟主程序
```



51中断的处理过程

中断程序:

```
ORG 0800H           ; INT0中断服务程序存放区
INT0SUB:  PUSH  ACC
            PUSH  PSW
            .....
            .....

            POP   PSW
            POP   ACC
            RETI      ; 中断返回

T1SUB:    PUSH  ACC           ; 定时器T1中断服务程序
            PUSH  PSW
            .....
            .....
            POP   PSW
            POP   ACC
            RETI      ; 中断返回
```



51中断的处理过程

例：假设程序中有2个中断源：INT0和T1，要求INT0为高优先级。

主程序：

问题：
中断程序执行完后
返回到哪一条程序？

```
ORG 0000H
LJMP MAIN    ; 跳转到主程序
ORG 0003H    ;INT0中断入口地址
LJMP INT0SUB  ;跳转到实际INT0中断服务程序存放空间
.....
ORG 001BH    ; T1中断入口地址
LJMP T1SUB    ;跳转到实际T1中断服务程序存放空间
.....
ORG 0030H    ; 实际主程序存放区
MAIN: MOV SP,#5FH ; 设置堆栈区
      :
      SETB IT0    ; 选择INT0为下降沿触发方式
      SETB EA     ; CPU开中断
      SETB EX0    ; INT0开中断
      SETB ET1    ; T1开中断
      SETB PX0    ; 设置INT0为高优先级
.....
SJMP $        ; 模拟主程序
```


Thank you!

