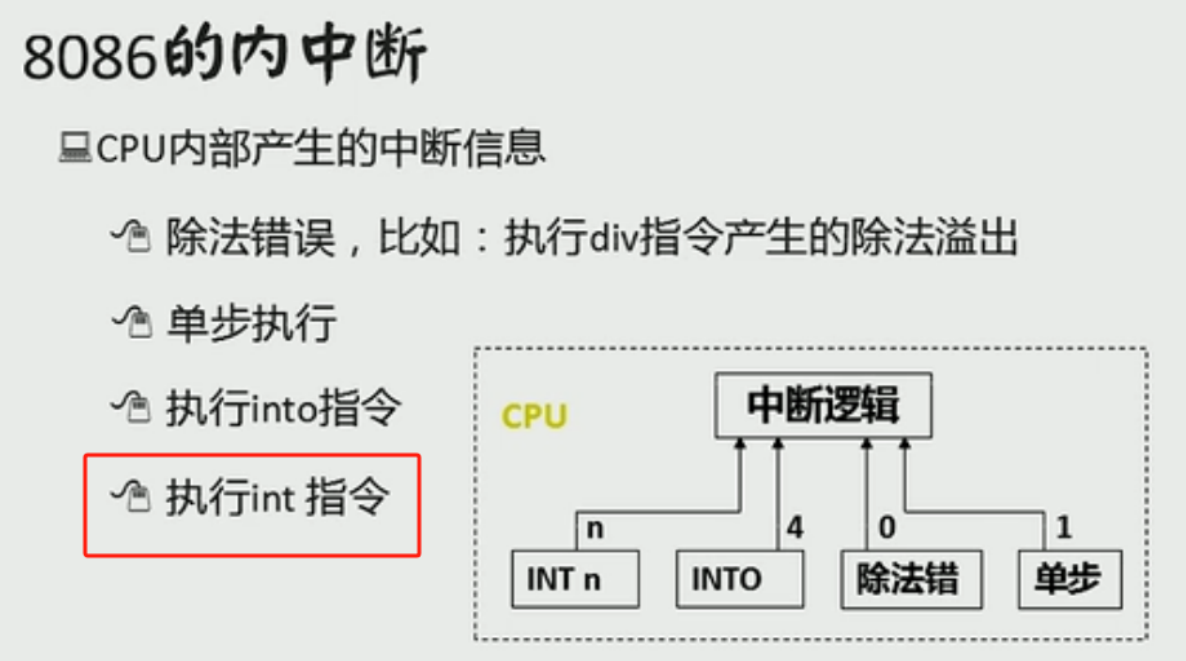
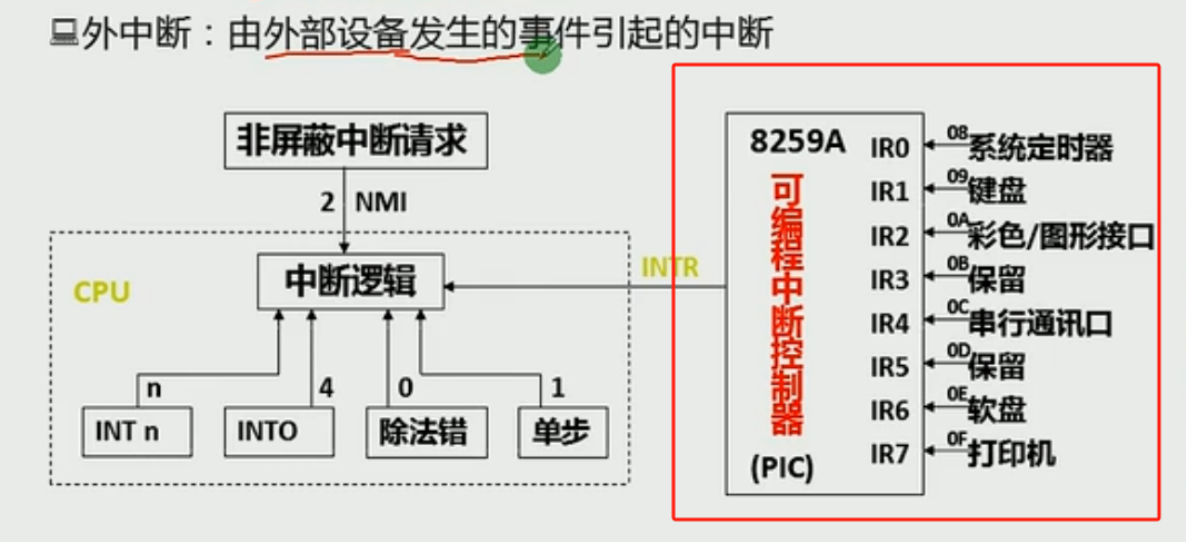
第6节 中断程序的编写与安装

1. 中断类型





内中断，就是执行一些指令时发生的中断，也叫软中断；

外中断，就是外部有个8259A控制器，接收外部设备中断信号发给cpu，也叫硬中断。

2. int n 中断

常见的中断

```
（1）01h：键盘输入并回显
    mov ah,01h;输入字符ASCII存到AL中
    int 21h
（2）02h：屏幕显示输出
    mov dl,al;入口参数送AL中
    mov ah,02h
    int 21h
```

```
（3）09h：屏幕输出字符串
    mov dx,data;字符串首地址送dx
    mov ah,09h
    int 21h
（4）10(0Ah):屏幕输入字符串
    mov dx,he;字符串存放地址附加段里要事先开辟
    mov ah,10
    int 21h
（5）4ch：返回DOS系统
    mov ah,4ch
    int 21h
```

8086cpu的dos系统中断向量表的首地址就是0，然后10h,13h,16h就是用于找到中断向量的偏移，这个偏移的地方存着每个中断处理函数的入口地址，中断向量表以及中断函数是属于操作系统的一部分，用于服务应用程序

```
中断服务程序入口地址 = 中断号 × 4
```

一般来说，一个中断向量对应的中断处理程序功能比较单一，但是也有功能很复杂的，比如21h的中断号，

这个中断定位到的中断程序的参数是ah，然后它会根据ah的值决定进行什么样的操作。但注意，21h处的中断处理函数，只有一个，只不过这一个函数用分支结构实现了很多功能，通过设置ah寄存器决定使用是什么功能；
比如ah=9，此时这个功能号的函数表示将ds:dx处的字符显示到屏幕上，所以说你还要多设置ds和dx两个寄存器参数的值。

再次强调一点，int n，n表示中断向量号，表示n号中断，乘以4就是其中断向量函数的指针（中断向量表起始地址为0），每个中断函数指针占用4个字节。

example06a.asm

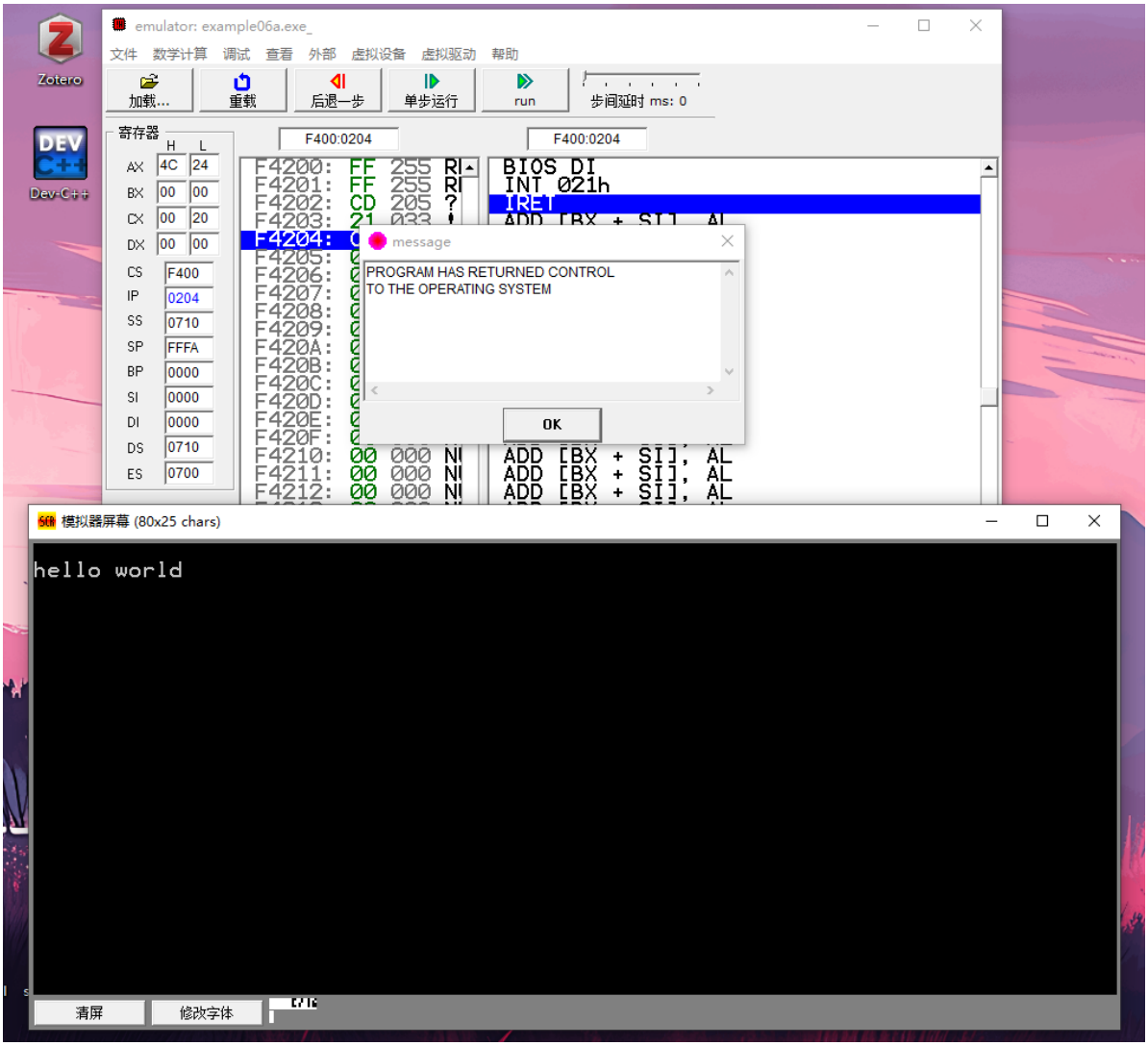
```
assume cs:code,ds:data
data segment
    a db 13,10,'hello world',13,10,'$'
data ends
code segment
main:
    ;设定21h号中断，ah=9时的参数
    mov ax,data
    mov ds,ax
    lea dx,a
    ;设定21h的功能号参数，由于在ah里面，所以前面的参数设定不能
    mov ah,9
    ;调用中断函数
    int 21h

    ;调用程序结束的中断函数
    mov ah,4ch
    int 21h
end main
code ends
```

13,10,'hello world',13,10,'$'

`13,10` 代表ASCII字符集中编码为13和10的字符,分别是回车符(CR)和换行符(LF)。这两个字符通常一起使用来表示一个行末。

`'hello world'` 是一个字符串字面量,表示字符序列"hello world"。

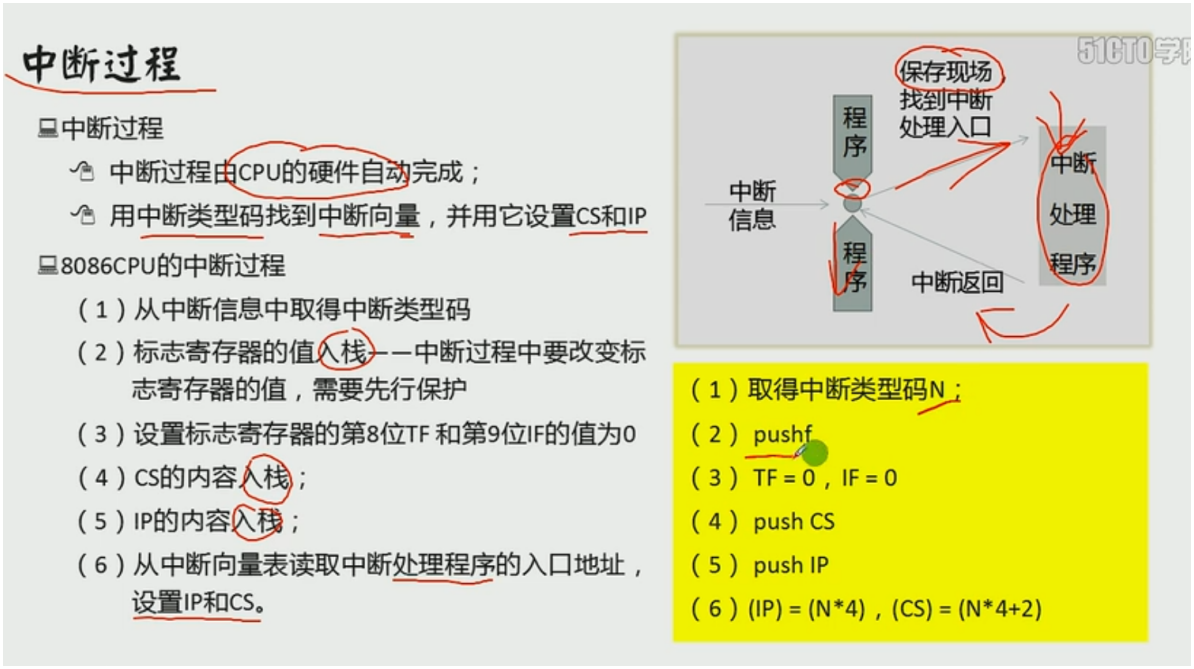
`'$'` 表示输出结束的符号



注意事项

一般还需要设置栈段，因为发生中断的时候会保存现场信息，这些信息会压入栈中，返回的时候还原现场信息

3. 中断发生的过程



注意，上面中断过程的1-6全部由硬件完成，而不是在中断处理函数中处理

并且中断返回的时候还原现场的过程也是完全由硬件完成，这种软硬结合的方式类似保护模式，

执行过程由硬件自动完成，但是执行过程需要用到内存中事先设定好的东西。

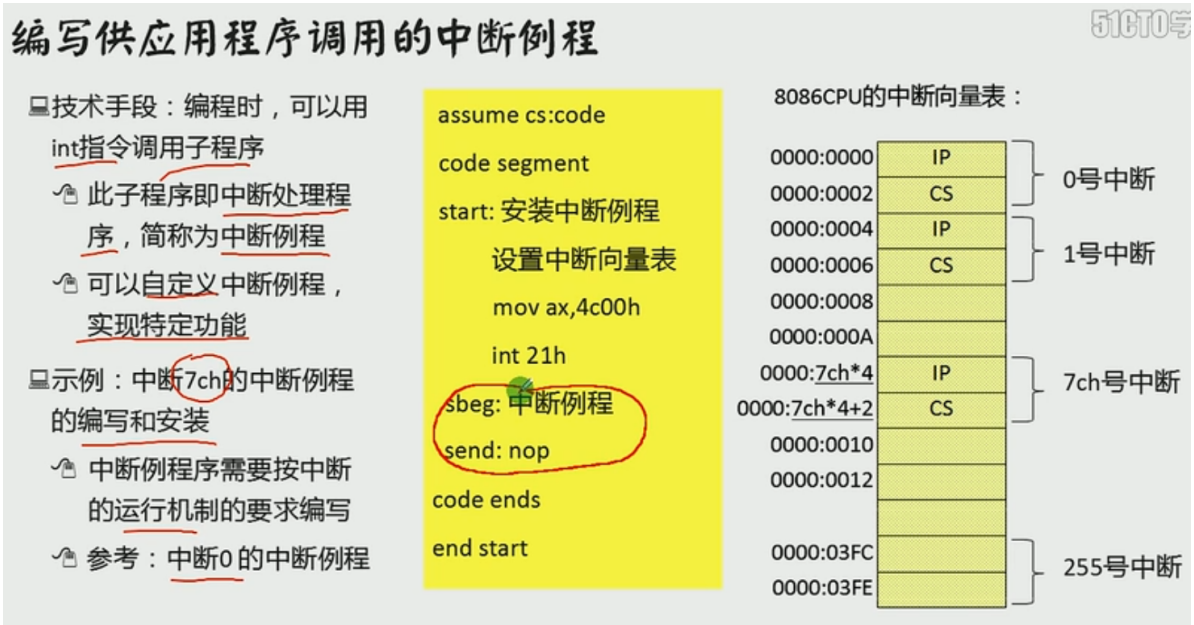
实际上中断程序的设计完全和函数是一回事，函数使用call和ret，每个指令包含了cpu的一系列操作，

而中断程序使用int n和iret，每个指令背后其实也是翻译成一系列指令交给cpu执行

另外一个重要经验：

使用寄存器作为参数的函数或者一些操作，都是硬件实现，都是硬件上的设计。比如寻址操作，以及这里的中断调用

4. 编写并安装中断程序



我们的想法是，将我们程序段编写的中断例程的首先放入到内存中一个比较合适的地址，这个地址保证不破坏系统，使得中断例程常驻内存中，然后将这个地址放入中断向量表7ch的地方。

由于我们对dos系统不是很了解，所以也不清楚哪段内存是有用的，哪段内存是没有的，所以我们的安装程序跳过搬运中断例程指令这一步了，就简单使用代码段中编写好的这个中断例程，把这个地方的地址写入中断向量表中。

中断例程的功能我们就使用example05里面的显示字符，编写并安装号中断例程后，我们尝试调用这个中断，看看能否成功。

example06b.asm

```
assume cs:code, ds:data, ss:stack

data segment
    a db 'hello world!'
data ends

stack segment
    db 100 dup('?')
stack ends

code segment
main:
; 初始化数据段和栈段
mov ax, data
mov ds, ax
mov ax, stack
mov ss, ax

; 设置中断向量表
mov ax, 0
mov es, ax
mov word ptr es:[7Ch*4], offset doStart
mov word ptr es:[7Ch*4+2], cs

; 触发中断
int 7Ch

; 程序结束返回
mov ax, 4C00h
int 21h

; 中断例程
doStart:
    mov ax, 0B800h
    mov es, ax
    mov si, 0
    mov di, 7C6h
    mov cx, 12
copy:
    mov al, a[si]
    mov es:[di], al
    inc si
    add di, 2
    loop copy
```

```
iret
doEnd:nop

code ends
```