内中断

## 1.产生

1. 除法溢出
2. 单步执行
3. 执行into指令
4. 执行Int 指令

## 2. 中断处理程序

## 3. 中断向量表

- 中断程序的入口地址
- 存放256个中断程序入口地址
- 存放在 **0000:0000** 到 **0000:03FF**
- 一个表项占两个字，高->段地址**CS** 低->偏移地址**IP**

## 4.中断的过程

1. 取得中断类型码N
2. pushf
3. TF=0,IF=0
4. push CS
5. push IP
6. $(IP)=(N*4),CS=(N*4)+2$
7. 开始执行中断程序

## 5.中断处理程序和iret指令

- 中断程序写法的常规步骤
    - 保存要用的寄存器
    - 处理中断
    - 回复寄存器
    - 用iret指令返回 --> pop ip,pop cs,pop popf(pop psw)

## 6. 除法错误中断的处理

```
1. 出现溢出
```

2. 产生0号中断信息
3. 执行0号中断
4. 返回操作系统

## 7. 编程处理0号中断

> *分析：*
>
> 1. *当发生除法溢出时，产生0号中断信息，从而引发中断过程*
>    a. *取得中断类型码N*
>    b. *pushf*
>    c. *TF=0,IF=0*
>    d. *push CS*
>    e. *push IP*
>    f. $(IP)=(N*4), CS=(N*4)+2$
> 2. *发生0号中断时，Cpu转去执行中断处理程序*
>    a. *相关处理*
>    b. *向显示缓冲区送字符串*
>    c. *返回DOS*
>    d. *do0*
> 3. *do0的程序应该放在那里*
>    a. *放在0号中断的向量表中0000:0200-0000:02FF*
> 4. *中断程序的入口地址放在那里*
>    a. *cs:0000:0002,ip:0000:0000*
>
> *总结*
>
> 1. *编写中断处理程序do0*
> 2. *将do0送入0000:0200*
> 3. *将do0的入口地址送到存储在中断向量表0号表项中*

```
assume cs:code

code segment

start:  do0安装程序
        设置中断向量表
        mov ax,4c00h
        int 21h
do0:    显示字符串“overflow”
        mov ax,4c00h
        int 21h

code ends
end start
```