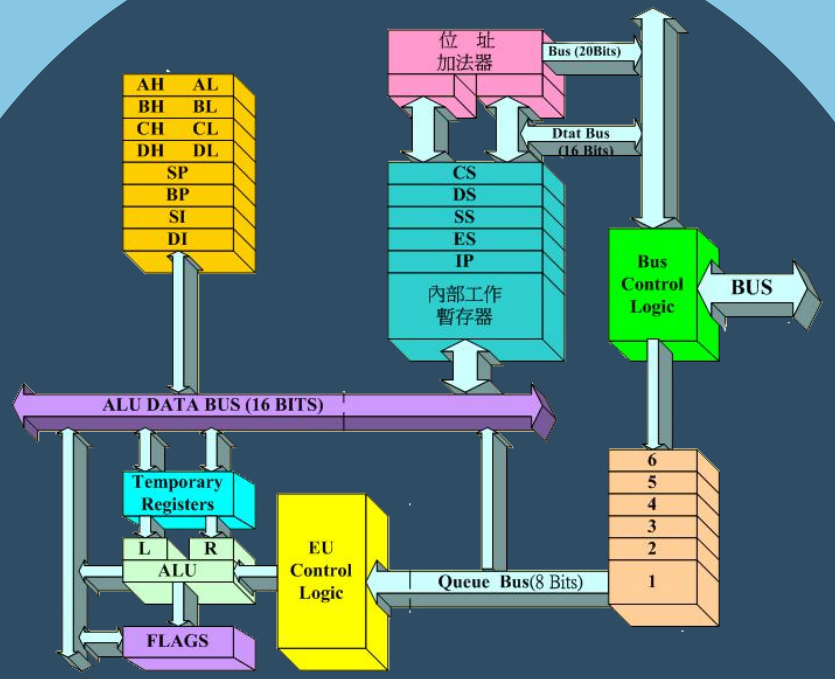
中断及其处理

贺利坚　主讲

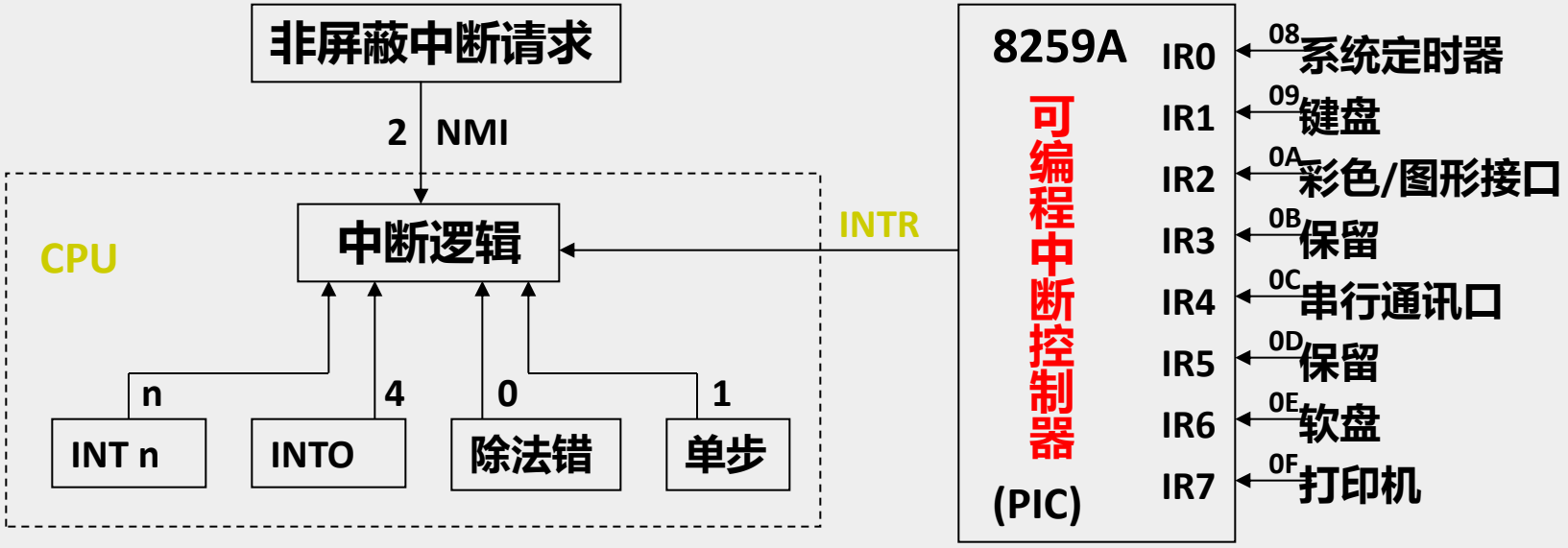


汇编语言程序设计

Assembly Language

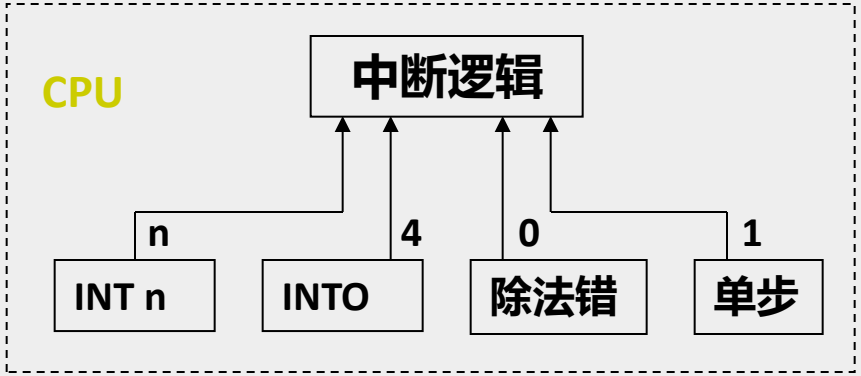
中断的概念

- 中断：CPU不再接着（刚执行完的指令）向下执行，而是转去处理中断信息。
- 内中断：由CPU内部发生的事件而引起的中断
- 外中断：由外部设备发生的事件引起的中断



8086的内中断

- CPU内部产生的中断信息
  - 除法错误，比如：执行div指令产生的除法溢出
  - 单步执行
  - 执行into指令
  - 执行int 指令



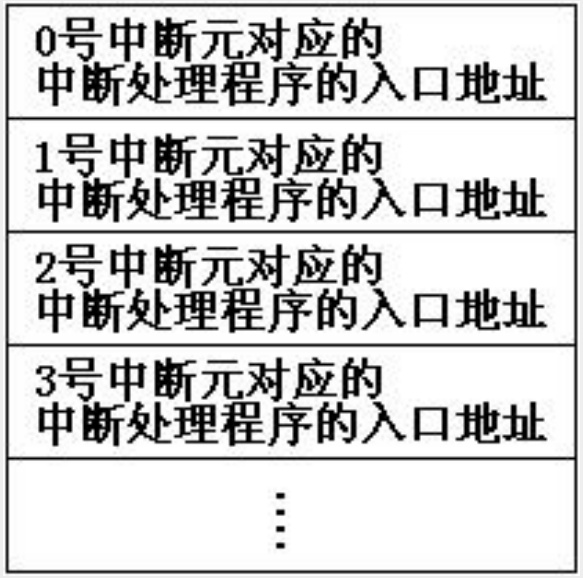
- 8086的中断类型码
  - （1）除法错误：0
  - （2）单步执行：1
  - （3）执行 into 指令：4
  - （4）执行 int n指令 ，立即数 n 为中断类型码。

```
assume cs:codesg, ss:stacksg, ds:datasg
stacksg segment
   db  200h dup (0)
stacksg ends
datasg segment
    szmsg db 13,10,'hello world!',13,10,'$'
datasg ends
codesg segment
start:
     mov ax,datasg
     mov ds,ax
     lea dx, szmsg
     mov ah,9
     int 21h

     mov ax,4c00h
     int 21h
codesg ends
end start
```

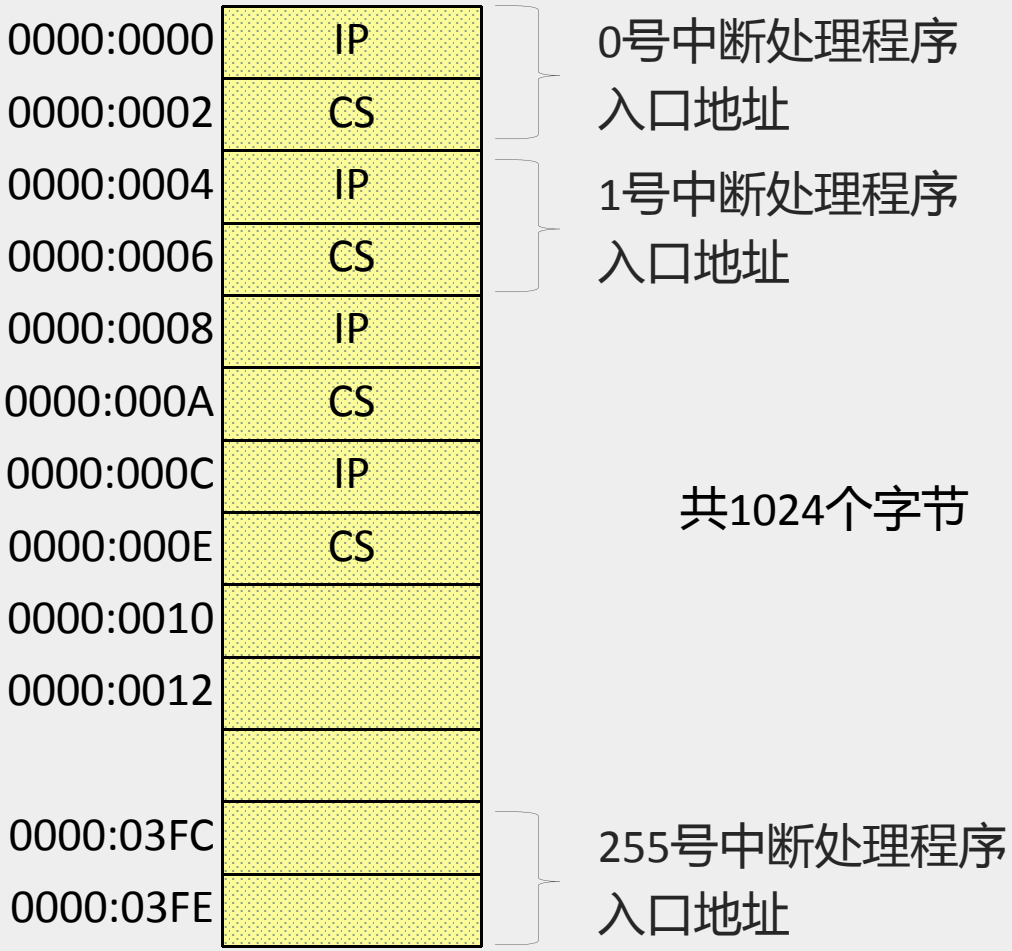
```
C:\>ptest

hello world!

C:\>_
```

中断处理程序

- CPU接到中断信息怎么办？
  - 执行中断处理程序
- 中断处理程序在哪里？
  - 中断信息和其处理程序的入口地址之间有某种联系，CPU根据中断信息可以找到要执行的处理程序。
- 中断向量表
  - 由中断类型码，查表得到中断处理程序的入口地址，从而定位中断处理程序。



8086CPU的中断向量表：



共1024个字节

(IP) = (N*4)，(CS) = (N*4+2)，N为中断类型码

案例：系统中的0号中断

```
-a
073F:0100 mov ax, 8
073F:0103 mov bh,0
073F:0105 div bh
073F:0107
-t

AX=0008  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 B700          MOV     BH,00
-t

AX=0008  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0105   NV UP EI PL NZ NA PO NC
073F:0105 F6F7          DIV     BH
-t

AX=0008  BX=0000  CX=0000  DX=0000  SP=00F7  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=F000  IP=1060   NV UP DI PL NZ NA PO NC
F000:1060 FE38          ???     [BX+SI]                       DS:0000=CD
-d 0:0 f
0000:0000  60 10 00 F0 BB 13 A3 01-08 00 70 00 B1 13 A3 01   `.........p.....
-
```

```
mov ax, 8
mov bh, 0
div bh
```

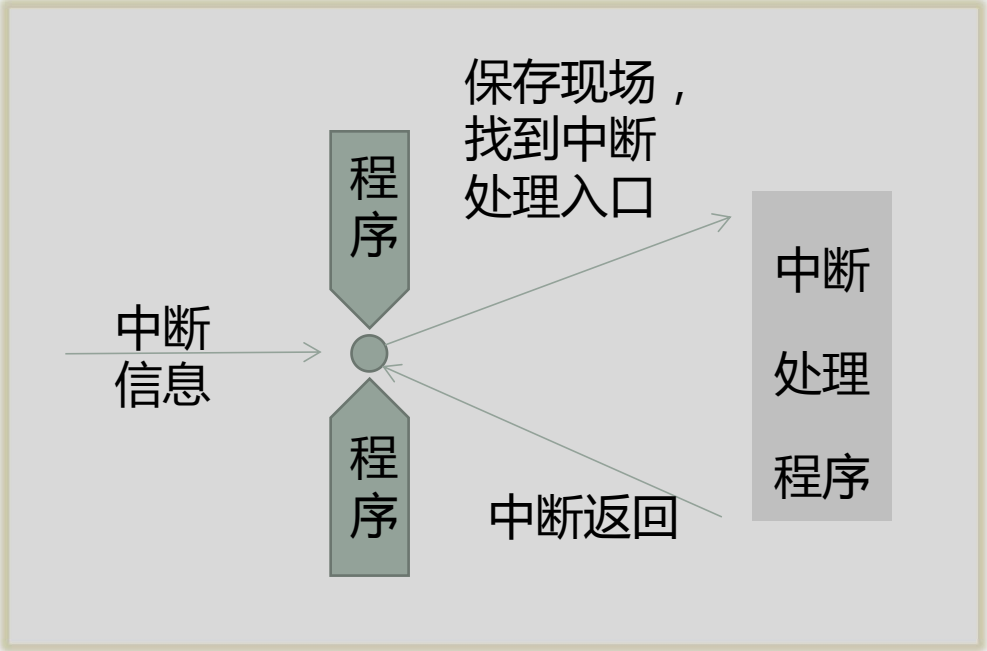
0000:0000	IP
0000:0002	CS
0000:0004	IP
0000:0006	CS
0000:03FC	
0000:03FE	

DOS 中，都是微软编写的代码。
有些代码，是它们内部掌握的，DEBUG 不给你显示。

你想要用 **FE38** 这条指令，只能用机器语言来编程。
因为，微软没有公开对应的汇编语言。

中断过程

- 中断过程
  - 中断过程由CPU的硬件自动完成；
  - 用中断类型码找到中断向量，并用它设置CS和IP
- 8086CPU的中断过程
  - （1）从中断信息中取得中断类型码
  - （2）标志寄存器的值入栈——中断过程中要改变标志寄存器的值，需要先行保护
  - （3）设置标志寄存器的第8位TF 和第9位IF的值为0
  - （4）CS的内容入栈；
  - （5）IP的内容入栈；
  - （6）从中断向量表读取中断处理程序的入口地址，设置IP和CS。



（1）取得中断类型码N；

（2）pushf

（3）TF = 0，IF = 0

（4）push CS

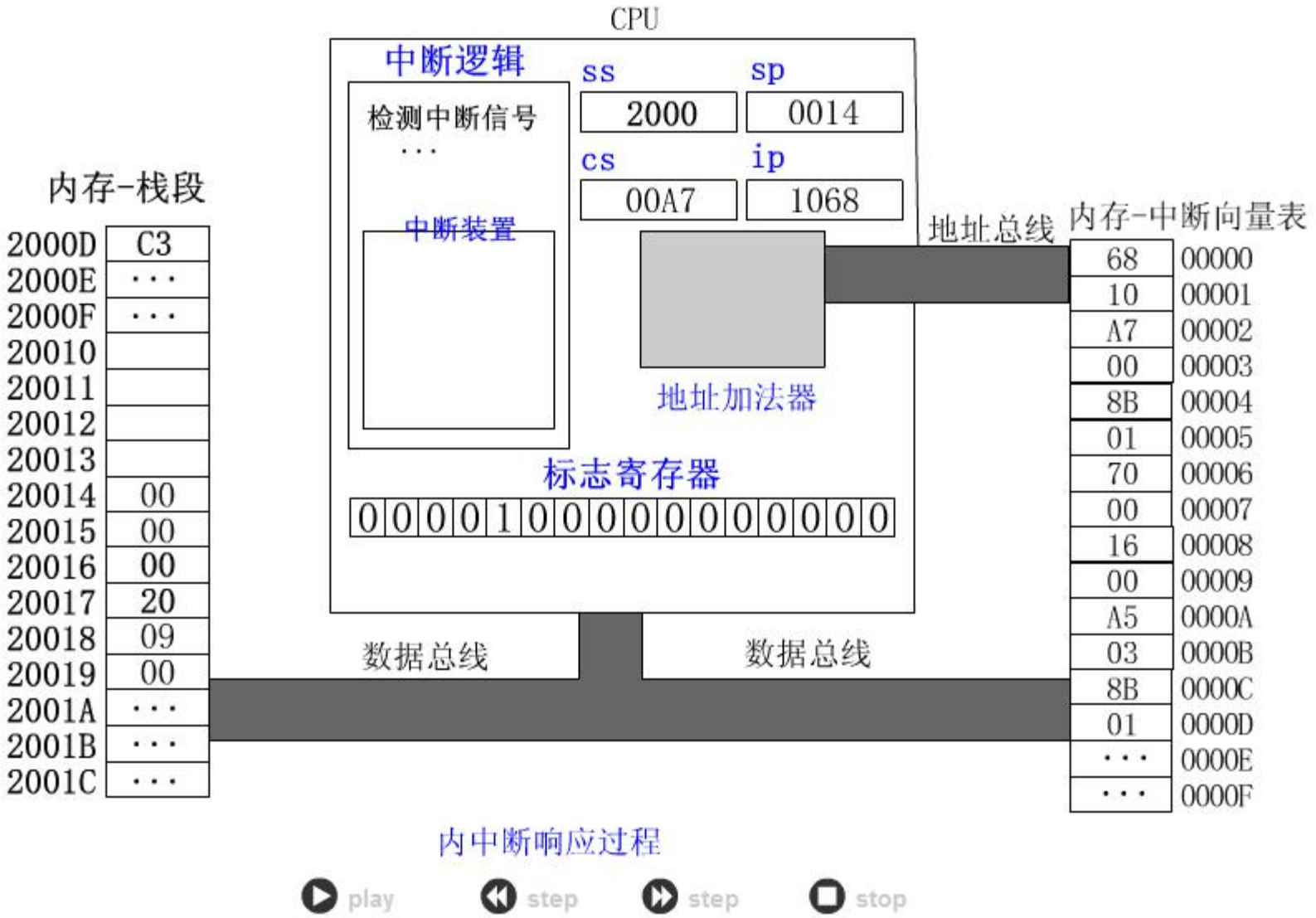
（5）push IP

（6）(IP) = (N*4)，(CS) = (N*4+2)

TF(Trap Flag): 陷阱标志，用于单步调试；

IF(Interrupt Flag)：中断标志；

中断过程演示


CPU
中断逻辑
检测中断信号
...
中断装置
ss
2000
sp
0014
cs
00A7
ip
1068
地址加法器
标志寄存器
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
地址总线
数据总线
数据总线
内存-栈段
2000D C3
2000E ...
2000F ...
20010
20011
20012
20013
20014 00
20015 00
20016 00
20017 20
20018 09
20019 00
2001A ...
2001B ...
2001C ...
内存-中断向量表
68 00000
10 00001
A7 00002
00 00003
8B 00004
01 00005
70 00006
00 00007
16 00008
00 00009
A5 0000A
03 0000B
8B 0000C
01 0000D
... 0000E
... 0000F
内中断响应过程
play
step
step
stop