汇编语言程序设计
Assembly Language Programming

主讲：徐娟

计算机与信息学院　计算机系 分布式控制研究所

E-mail: xujuan@hfut.edu.cn,

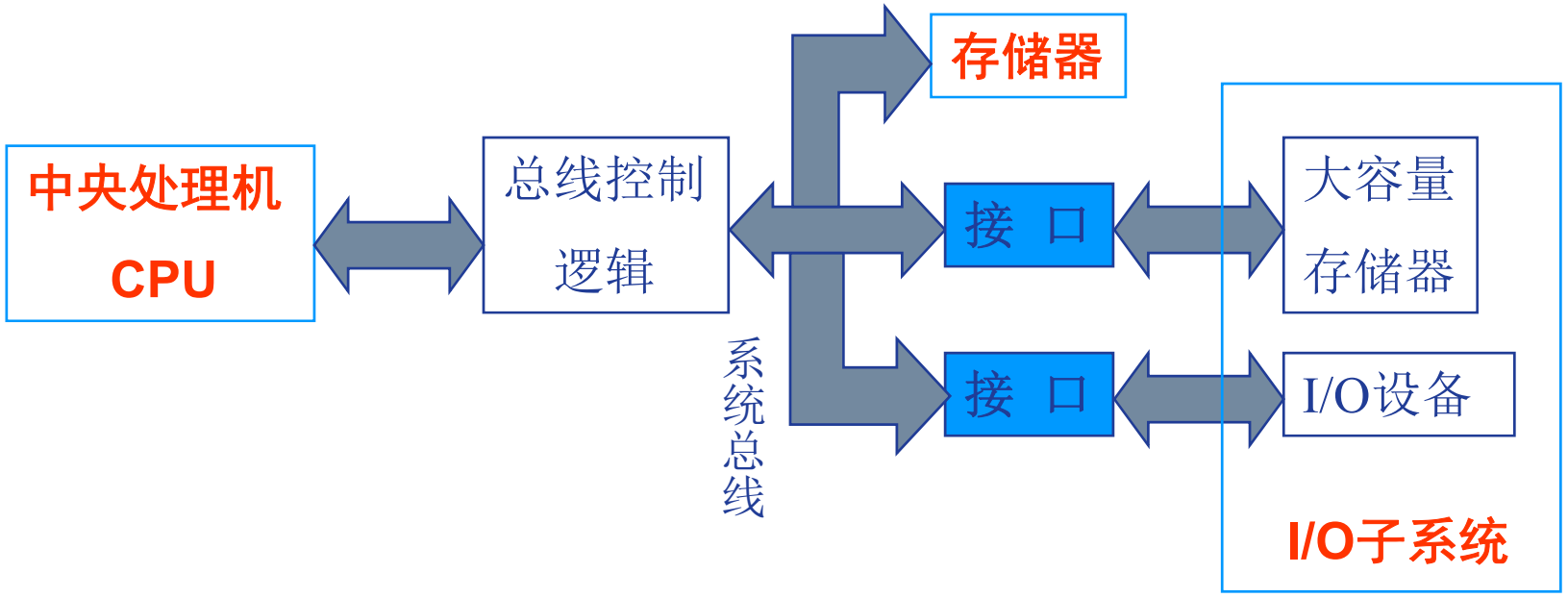
Mobile: 18055100485

第五章
高级汇编语言程序设计

5.3 输入输出程序设计

❖接口

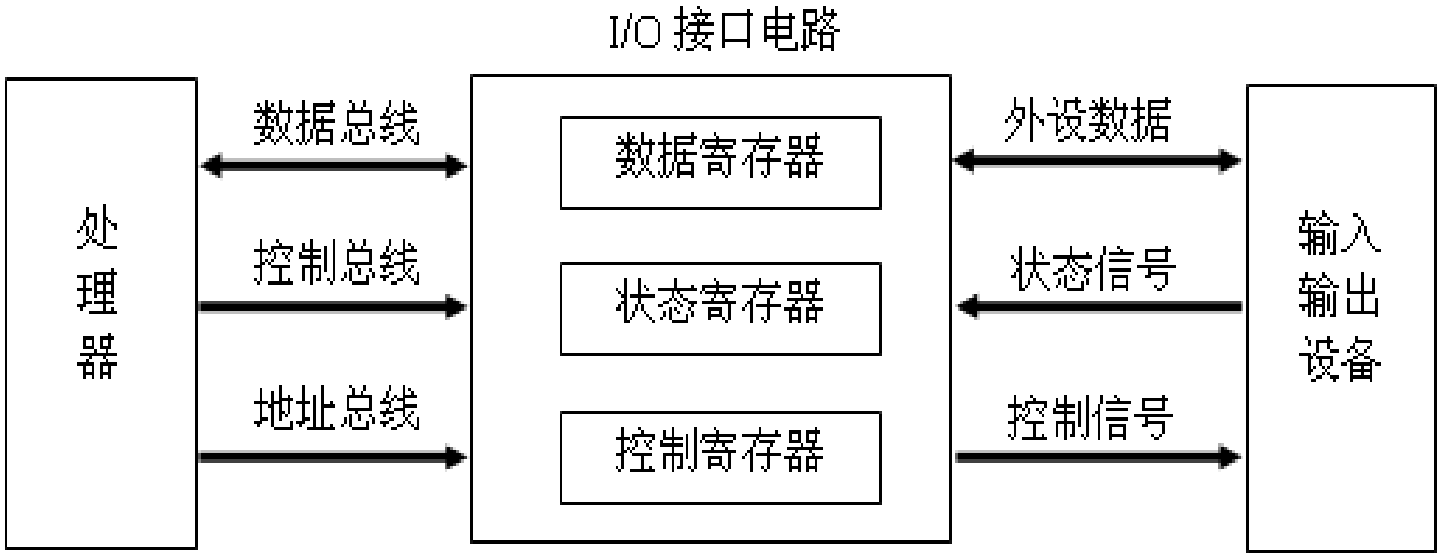
- CPU和存储器、外部设备或者两种外部设备之间，或者两种机器之间通过系统总线进行连接的逻辑部件（或称电路）
- 是CPU与外界进行信息交换的中转站。



接口与端口

❖ I/O接口电路组成

设备状态寄存器、设备控制寄存器、数据寄存器。



- 源程序和原始数据通过接口从输入设备（例如键盘）送入，运算结果通过接口向输出设备（例如CRT显示器、打印机）送出去；
- 控制命令通过接口发出去（例如步进电机）
- 现场信息通过接口取进来（例如温度值、转速值）。

接口与端口

I/O端口

- 接口中的寄存器又叫做I/O端口，每一个端口有一个编号，叫做端口号，又叫端口地址。
- 数据寄存器：数据端口，用于对来自CPU和外设的数据起缓冲作用。
- 状态寄存器：状态端口，用来存放外部设备或者接口部件本身的状态。CPU通过对状态端口的访问和测试，可以知道外部设备或接口本身的当前状态。
- 控制寄存器：控制端口，用来存放CPU发出的控制信息，以控制接口和外部设备的动作。
- CPU与外部设备之间传送信息都是通过数据总线写入端口或从端口中读出的，所以，CPU对外部设备的寻址，实质上是对I/O端口的寻址。

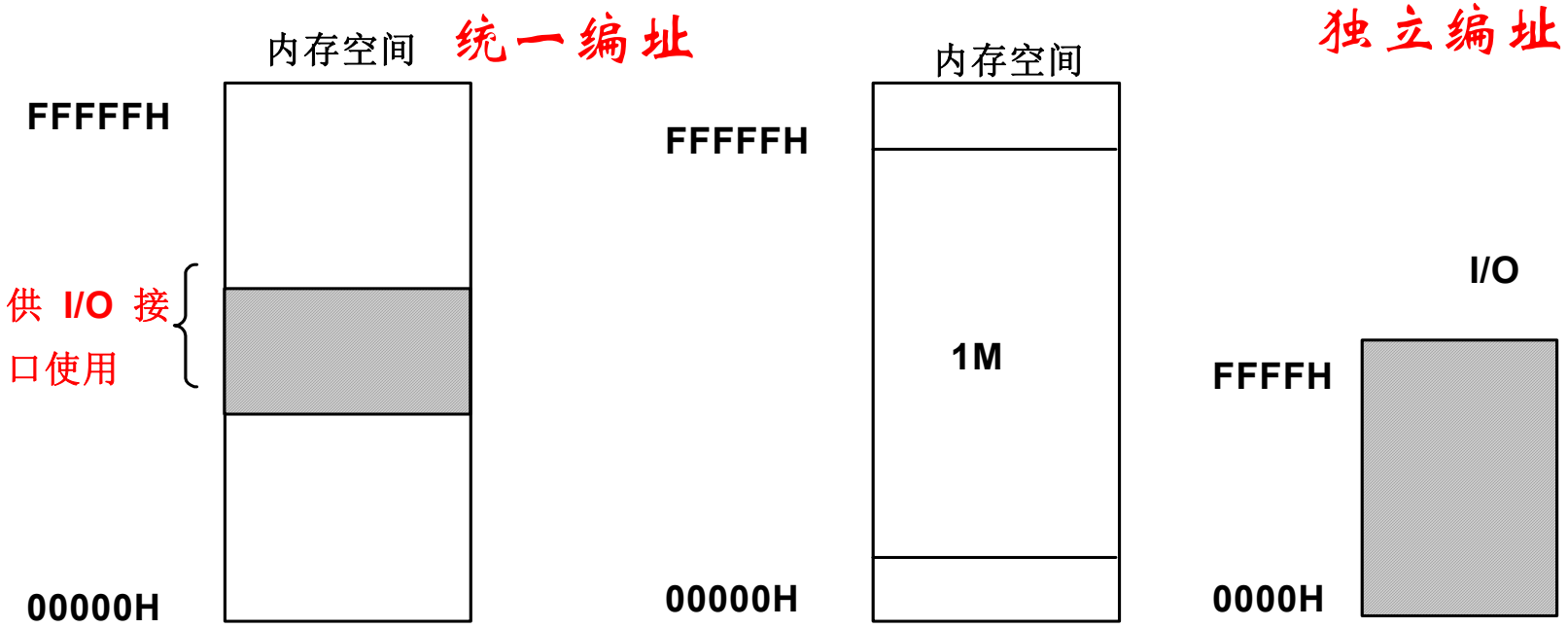
接口与端口

端口

- 8086通过输入输出指令与外设进行数据交换；呈现给程序员的外设是端口（Port）即I/O地址
- 端口分类：数据端口，状态端口，控制端口
- 8086用于寻址外设端口的地址线为16条，端口最多为2^{16}＝65536（64K）个，端口号（地位）为0000H～FFFFH
- 每个端口用于传送一个字节的外设数据，都是8位端口
- 独立编址方式

接口与端口

I/O端口编址方式



（a）存储器映射方式示意图

（b）I/O 映射方式示意图

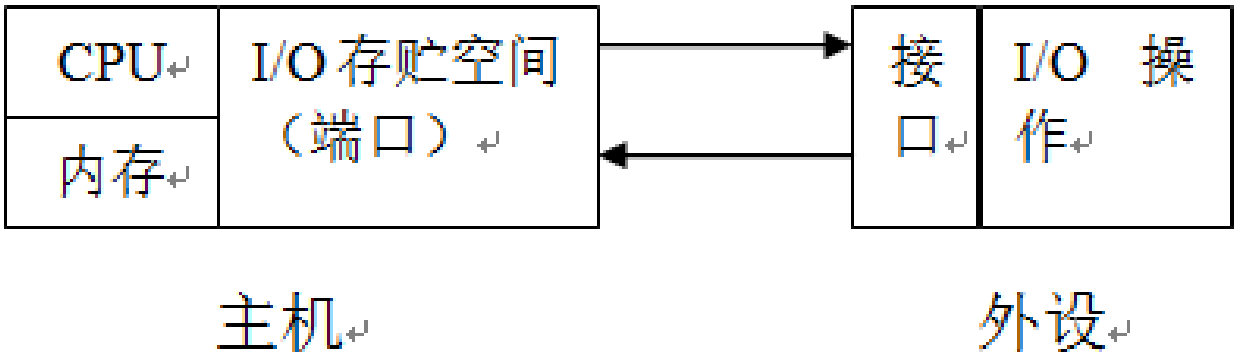
所有对内存操作的指令对I/O端口同样有效，指令丰富，但会损失一部分的内存空间。

对I/O端口有专门的指令。

缺点是对I/O端口操作的指令不及统一编址时丰富（例如，8086/8088中对I/O端口就只有最基本的输入输出指令），但能最大程度地满足存储空间的寻址范围。

接口与端口(小结)

- 计算机的外设都是通过接口连接到系统上，每个接口由一组寄存器组成，寄存器都有一个称为I/O端口的地址编码。
- 每一台外设都通过硬件接口与主机端口相连，并交换信息。
- 8086/8088对端口的操作通过独立编址



输入输出寻址方式

- 8086的端口有64K个，无需分段，设计有两种寻址方式
- 直接寻址：只用于寻址00H～FFH前256个端口，操作数i8表示端口号
- 间接寻址：可用于寻址全部64K个端口，DX寄存器的值就是端口号，端口号≥ 256，端口号 → DX
- 大于FFH的端口只能采用间接寻址方式
- 可对0～65535个端口地址进行访问

输入指令IN

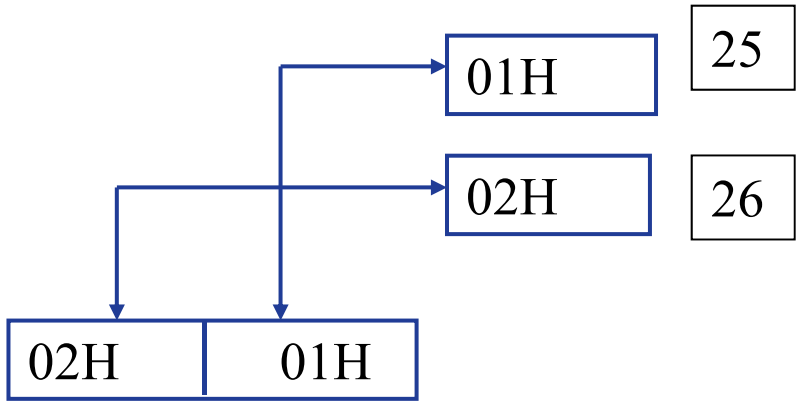
- 格式：IN AL/AX, PORT/DX
- 举例

IN AL,25 ;

AL←25号端口的内容

IN AX,25 ;

AX← 25和26端口的内容



IN AL,DX ;

AL ← (DX)所指端口的内容

IN AX,DX ;

AX ← (DX)、(DX)+1所指端口的内容

输出指令

- 格式：OUT PORT/DX, AL/AX
- 举例：

```
OUT 25,AL ;      (AL) → 25号端口
OUT 25,AX ;      (AX) → 25号端口和26号端口
OUT DX,AL;       AL → (DX)所指端口
OUT DX,AX;       (AX) → (DX)和(DX)+1二个端口
```

输入输出传送方式

- 计算机与外部设备之间就要进行数据交换。但由于外部设备与存储器不同，它们用各自不同的速度在工作，而且它们的工作速度相差很大，有些外部设备的工作速度极高，有些则很低。因此需要用某种方法调整数据传输时的定时，这种方法称为输入/输出控制。

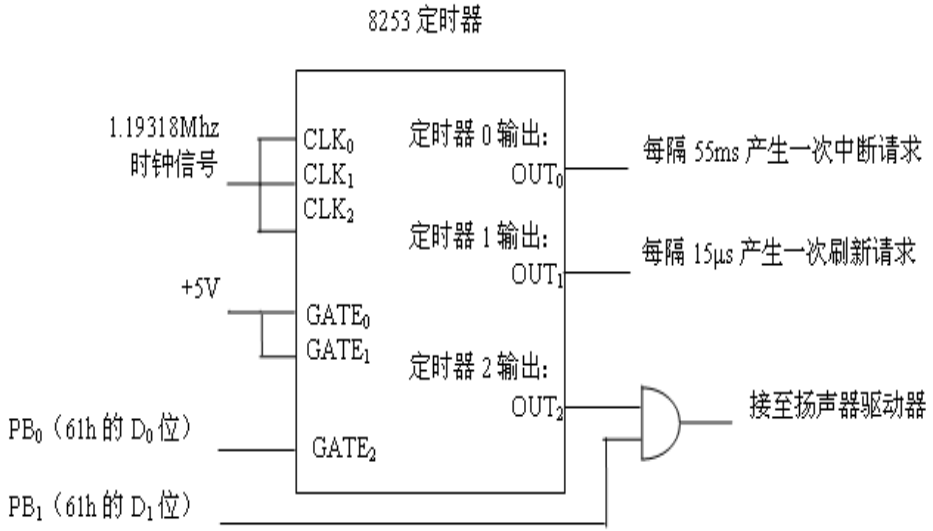
- **程序传送方式**
  - 程序直接控制（无条件传送）
  - 程序查询（有条件传送）
- **中断传送方式**
- DMA传送方式
- I/O处理机方式

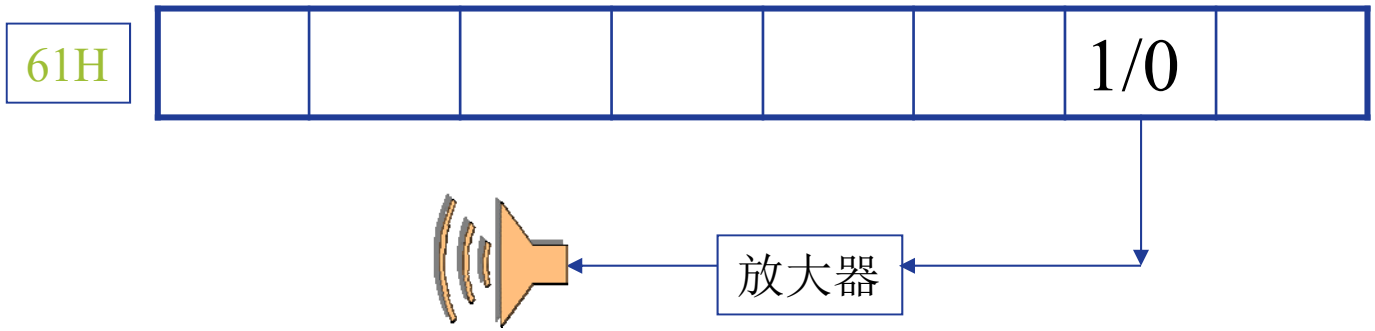
程序直接控制输入输出

程序执行IN或OUT指令实现数据传送；

主要用于外设的定时是固定的或已知的场合

- 61H端口的D_1控制扬声器的开关
- D_0为定时器，置为1，则由8253的2号计数器来驱动；置为0，则手工驱动。
- 11——发声；00-不发声；





程序直接控制输入输出

例5.13：主程序

```
.model tiny                ;形成com格式的程序
.code
.startup
call speaker_on            ;打开扬声器声音
mov ah,1                   ;等待按键
int 21h
call speaker_off           ;关闭扬声器声音
.exit 0
```

- 确认你的PC机主板具有蜂鸣器（不是音箱）
- 进入模拟DOS（COMMAND.COM，不是CMD.EXE）
- 再次执行该程序，注意听声音（较小）

程序直接控制输入输出

例5.13：子程序

```
speaker_on proc          ;扬声器开子程序
        push ax
        in al,61h        ;读取原来控制信息
        or al,03h        ;D1D0=11b
        out 61h,al       ;直接控制发声
        pop ax
        ret
speaker_on endp
```

扬声器关

```
;扬声器关子程序speaker_off
        and al,0fch              ; D1D0=00b
```

查询式I/O方式

查询传送方式也称为条件传送方式。也就是微处理器在执行输入/输出指令读取数据之前，要通过执行程序不断地读取并测试外部设备的状态。适合于低速外设与CPU传送信息。

完成一个数据传送的步骤：

（1）微处理器用输入指令从接口中的状态端口读取状态字；

（2）微处理器测试所读取的状态字的相应状态位是否满足数据传输的条件，如果不满足，则回到第（1）步，继续读状态字；

（3）如果状态位表明外部设备已满足传输数据的条件，则进行传送数据的操作。

查询式I/O方式

查询式I/O方式

其工作原理分析如下：

输入设备：在数据准备好以后，就往接口发一个选通信号STB，该选通信号将准备好的数据锁入锁存器，同时将接口中的D触发器置1，表明锁存器中有数据，它作为状态信息，使接口中三态缓冲器的READY位置1。数据信息和状态信息从数据端口和状态端口经过数据总线送入微处理器。

微处理器从外设输入数据的步骤：

- 先读取状态字并检查状态字的相应位，查明数据是否准备就绪，即数据是否已进入接口的锁存器中；
- 如果准备就绪，则执行输入指令，读取数据，此时将状态位清零，这样便开始下一个数据传输过程。

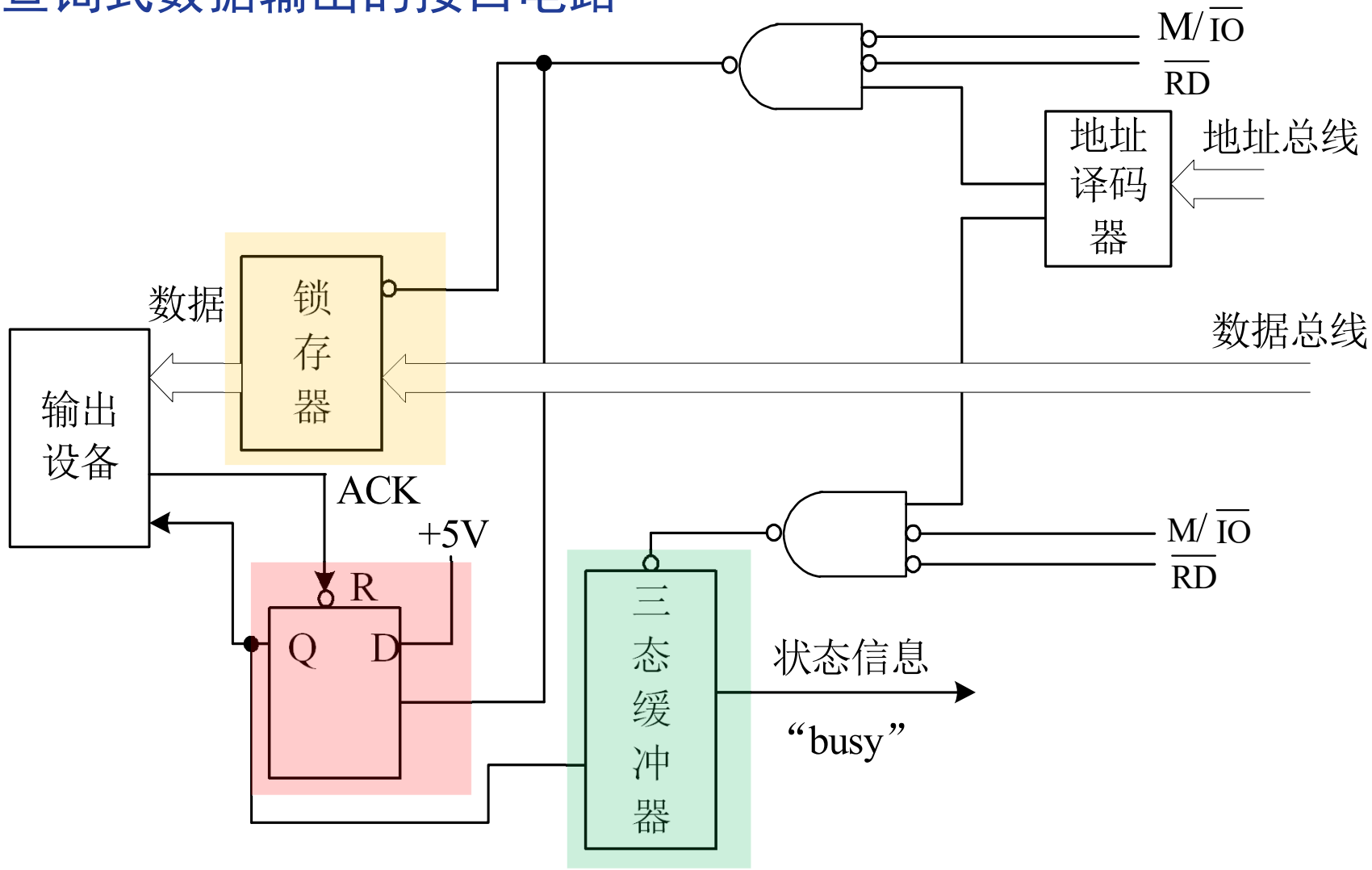
查询式I/O方式

查询输入部分的参考程序如下：

```
POLL：MOV   DX，STATUS-PORT  ；状态端口号送DX
IN    AL，DX                 ；输入状态信息
TEST  AL，80H                ；检查Ready是否为高电平
JE    POLL                   ；如果未准备好，进行循环检测
MOV   DX，DATA-PORT          ；准备就绪，读入数据
IN    AL，DX
```

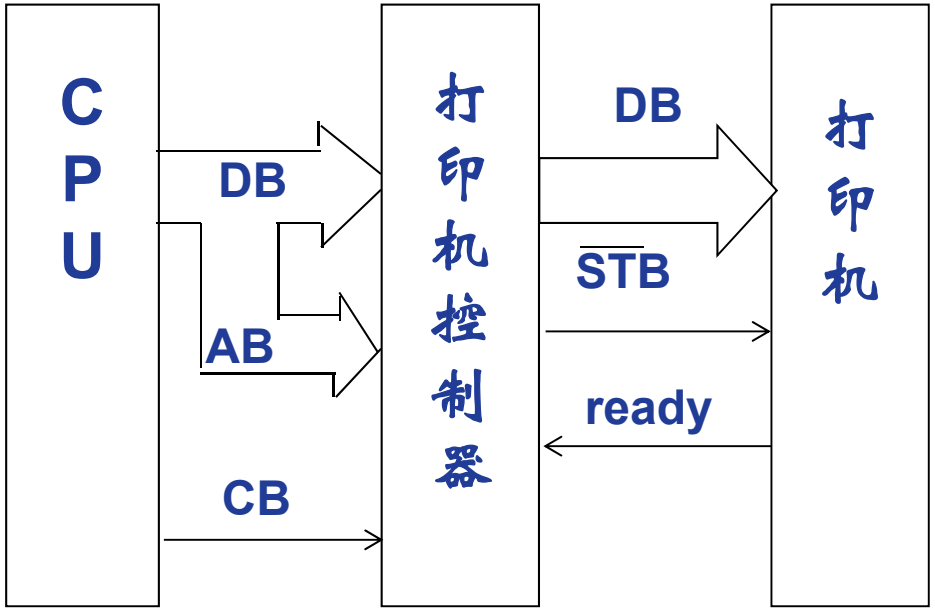
查询式I/O方式

查询式数据输出的接口电路

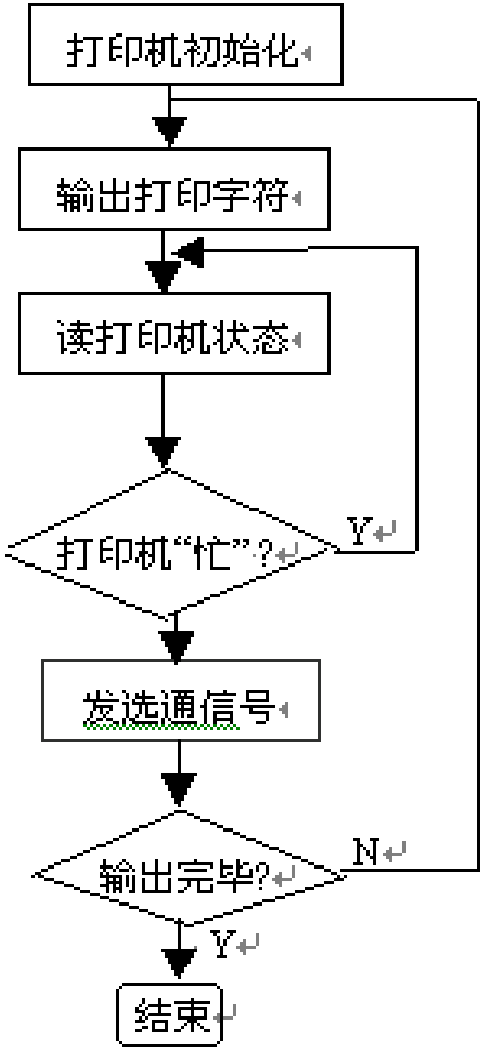


查询式I/O方式

【例】向打印机输出字符。



打印机连接示意图



中断（Interrupt）传送方式

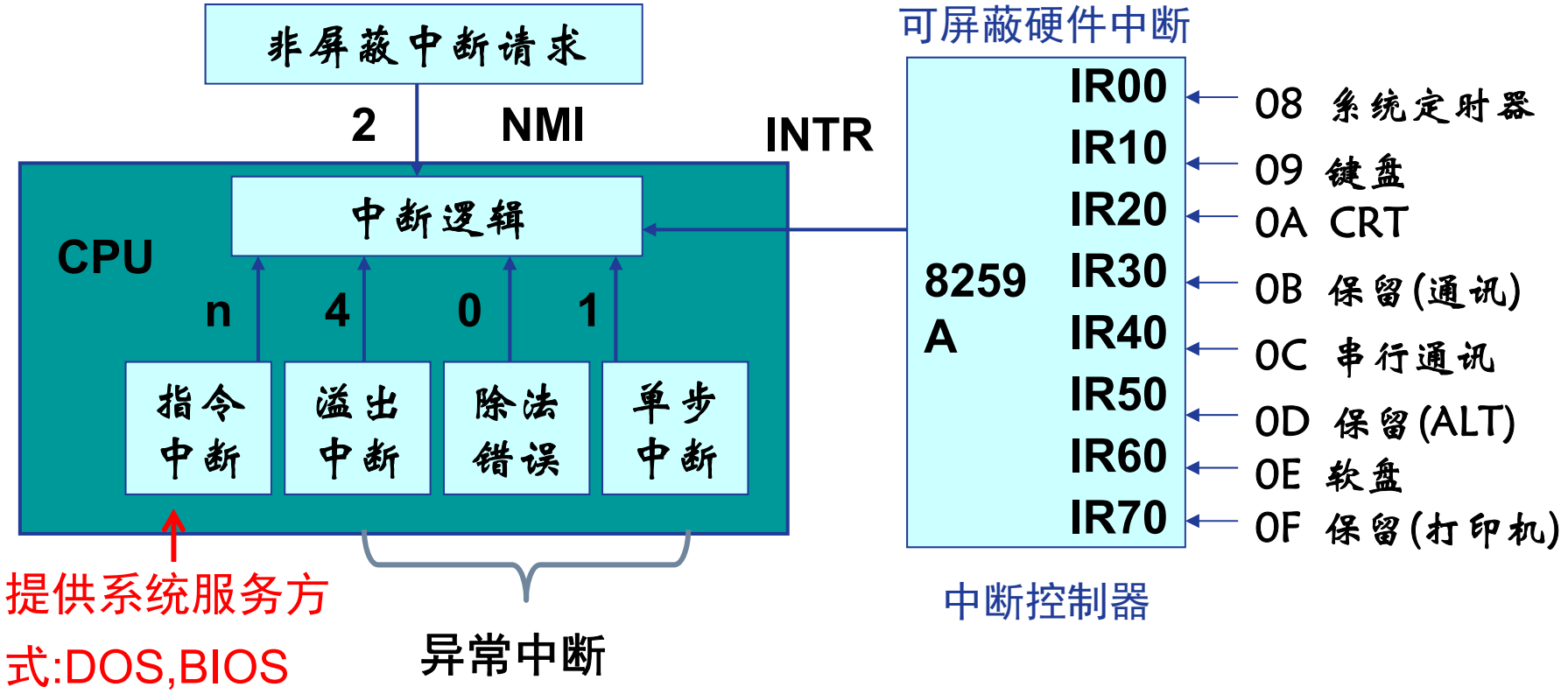
- 查询方式的缺点：CPU反复等待状态位，浪费大量CPU资源。为了提高CPU的利用效率和进行实时数据处理，CPU常采用中断方式与外设进行数据交换。
- 需要交换数据的外设，采用中断请求向处理器提出要求
- CPU在满足一定的条件下，暂停执行当前正在执行的主程序，转入执行相应能够进行输入/输出操作的子程序，待输入/输出操作执行完毕之后CPU即返回继续执行原来被中断的主程序。
- 中断的优点：分时操作；　实现实时处理；　故障处理

中断相关概念

- 8088/8086中断系统：256种中断，分为内部、外部两种类型

中断源：引起中断的事件；

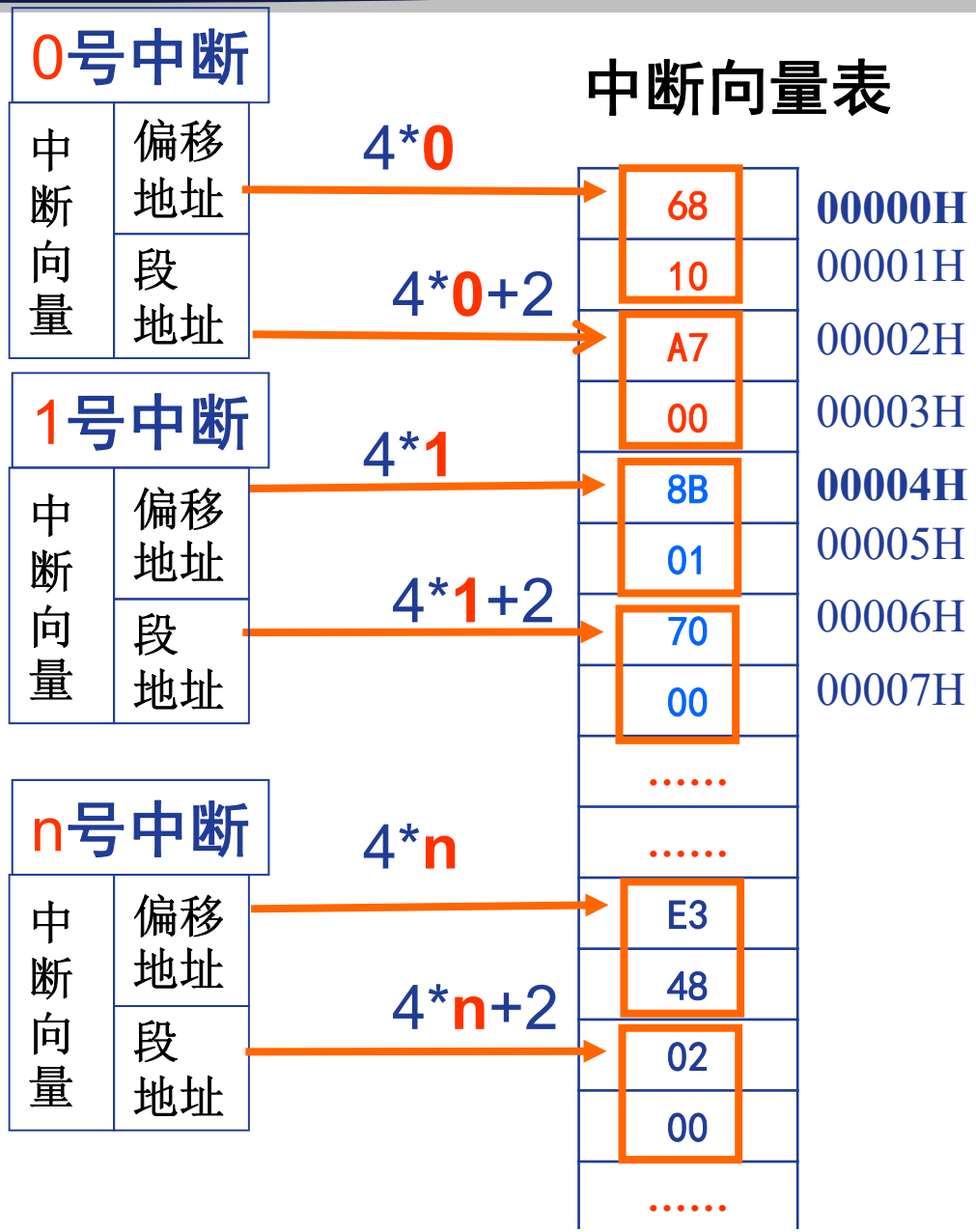
输入/输出设备；实时时钟；故障源；为调试程序而设置的中断源



中断相关概念

- 中断服务子程序
  - 每种中断都有与之对应的处理程序
- 中断向量
  - 中断服务子程序的入口地址(16位偏移地址，16位段地址)
- 中断向量表
  - 存放中断向量的表格。
  - 256个，00000H-003FFH，1KB
- 中断类型码
  - 表格的编号 n

中断相关概念


0号中断
中断向量
偏移地址
段地址
1号中断
中断向量
偏移地址
段地址
n号中断
中断向量
偏移地址
段地址
中断向量表
4*0
4*0+2
4*1
4*1+2
4*n
4*n+2
68
10
A7
00
8B
01
70
00
.....
.....
E3
48
02
00
.....
00000H
00001H
00002H
00003H
00004H
00005H
00006H
00007H

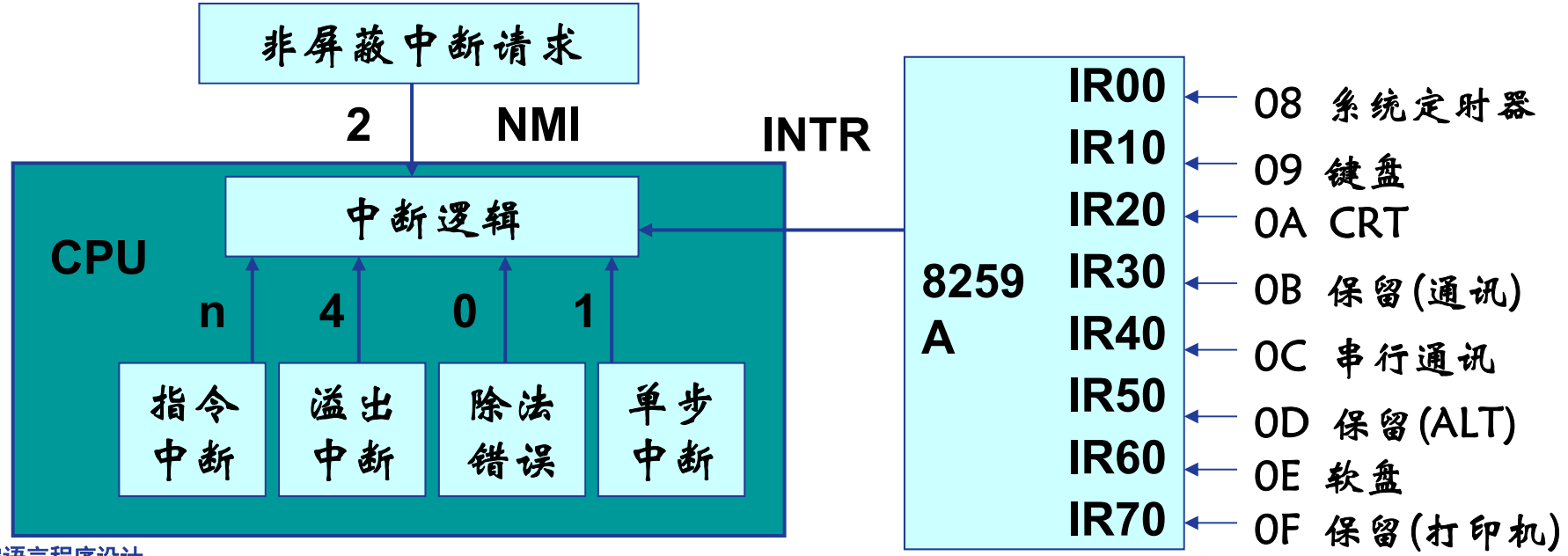
中断相关概念

```
C:\>debug
-d 0000:0000
0000:0000  68 10 A7 00 8B 01 70 00-16 00 A9 03 8B 01 70 00
0000:0010  8B 01 70 00 B9 06 12 02-40 07 12 02 FF 03 12 02
0000:0020  46 07 12 02 0A 04 12 02-3A 00 A9 03 54 00 A9 03
0000:0030  6E 00 A9 03 88 00 A9 03-A2 00 A9 03 FF 03 12 02
0000:0040  A9 08 12 02 A4 09 12 02-AA 09 12 02 5D 04 12 02
0000:0050  B0 09 12 02 0D 02 E1 02-C4 09 12 02 8B 05 12 02
0000:0060  0E 0C 12 02 14 0C 12 02-1F 0C 12 02 AD 06 12 02
0000:0070  AD 06 12 02 A4 F0 00 F0-37 05 12 02 18 3B 00 C0
0000:0080  72 10 A7 00 7C 10 A7 00-4F 03 E2 05 8A 03 E2 05
0000:0090  17 03 E2 05 86 10 A7 00-90 10 A7 00 9A 10 A7 00
0000:00A0  B8 10 A7 00 54 02 70 00-F2 04 47 D7 B8 10 A7 00
0000:00B0  B8 10 A7 00 B8 10 A7 00-40 01 27 04 50 09 7E DF
0000:00C0  EA AE 10 A7 00 E8 00 F0-B8 10 A7 00 A6 24 02 D4
0000:00D0  B8 10 A7 00 B8 10 A7 00-B8 10 A7 00 B8 10 A7 00
.........
```

中断相关概念

中断类型号的获取

- 0～5号中断请求：一旦被响应，系统自动提供中断类型号，并自动地转到中断处理程序中去。
- 可屏蔽的外部中断INTR：经过中断控制器8259，在CPU中断响应的第二个周期，通过中断响应信号，将对应的中断类型号送至数据总线。
- 内部中断：通过INT n指令将中断号直接发送给CPU。



中断相关概念

从外设发出的中断请求到cpu相应中断，用两个控制条件起决定作用：

1）外设的中断请求是否被屏蔽，2）cpu是否允许相应中断。

外部设备向cpu发出中断请求，cpu是否响应还与IF有关

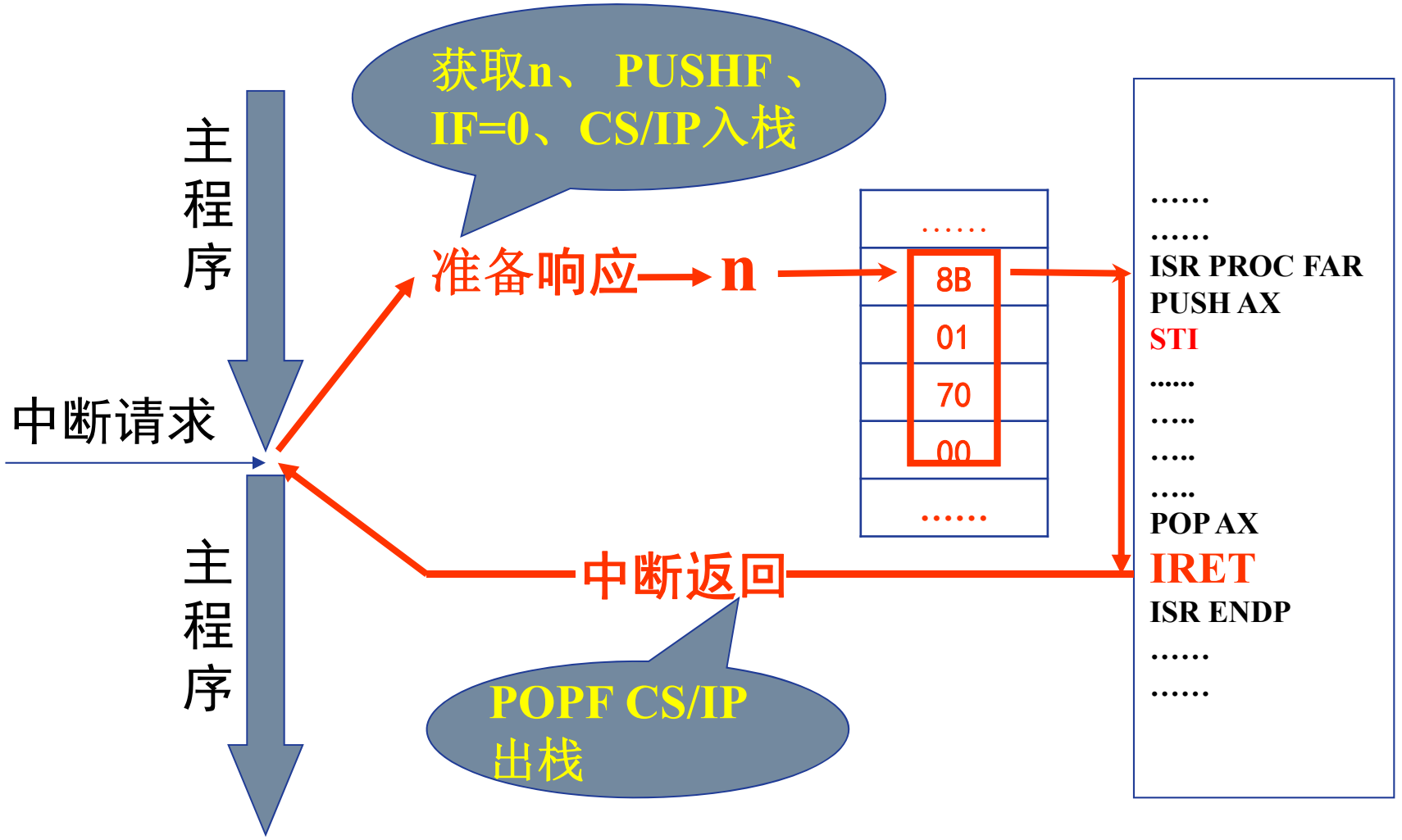
STI——开中断指令

将标志寄存器中的中断标志位IF置1，允许CPU响应来自INTR引脚的中断请求

CLI——关中断指令

将标志寄存器中的中断标志位IF清0，使CPU不响应来自INTR引脚的中断请求

中 断 过 程


主程序
中断请求
主程序
获取n、 PUSHF 、IF=0、CS/IP入栈
准备响应→n
中断返回
POPF CS/IP出栈
.....
8B
01
70
00
.....
.....
ISR PROC FAR
PUSH AX
STI
.....
.....
.....
POP AX
IRET
ISR ENDP
.....
.....

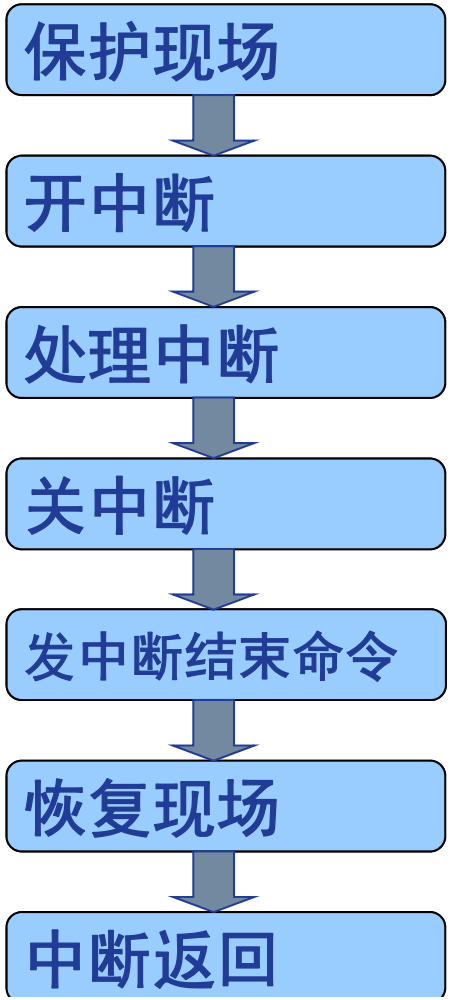
中断服务子程序

- 与一般子程序的差别：
  - 中断服务子程序应为FAR
  - 中断响应时IF=0，子程序里一般应IF←1
  - 硬件中断处理程序，最后发中断结束EOI命令
  - 返回为IRET而非RET
  - 由系统进行调用

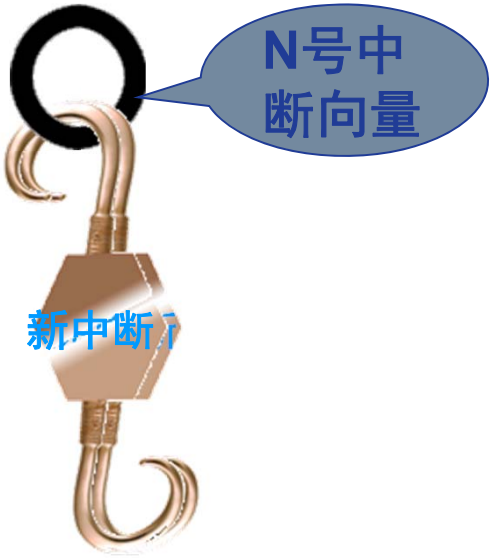
中断服务子程序的编写

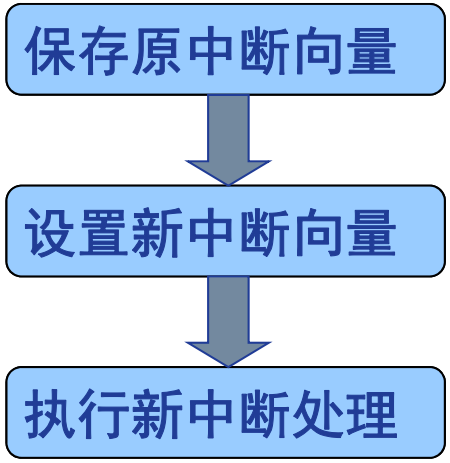
中断处理程序的结构与子程序(即过程)相似，可用定义过程的方式来定义中断处理程序。所有编写过程的一些规定和要求均适用于中断处理程序，包括用伪指令PROC/ENDP定义过程为远类型。

```
ISR PROC FAR
    PUSH AX ……
    STI                    ;便于中断嵌套
    ……
    CLI
    EOI (End Of Interrupt)
    POP AX ……
    IRET                   ;中断返回
ISR ENDP
```



完整中断程序的编写





完整中断程序的编写

保存原中断向量

设置新中断向量

执行新中断处理

调用原中断处理

恢复原中断处理

完整中断程序的编写

保存原中断向量

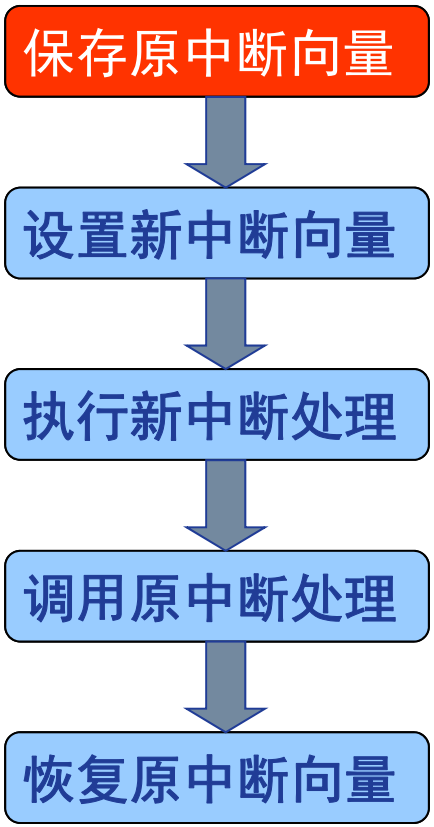
设置新中断向量

执行新中断处理

调用原中断处理

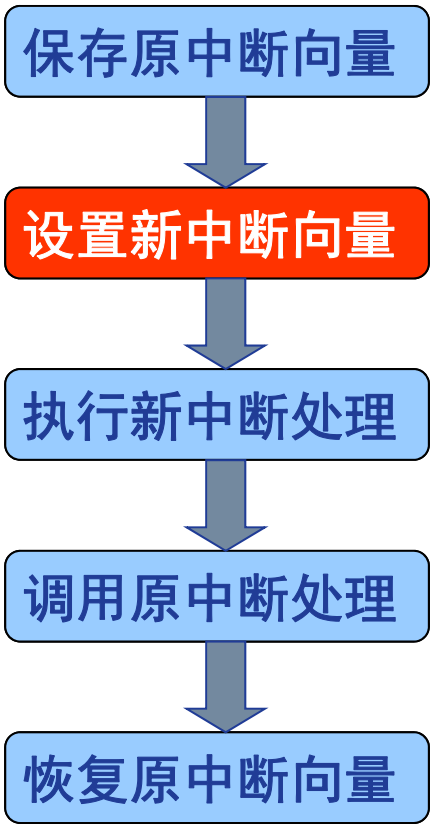
恢复原中断向量

保存原中断向量



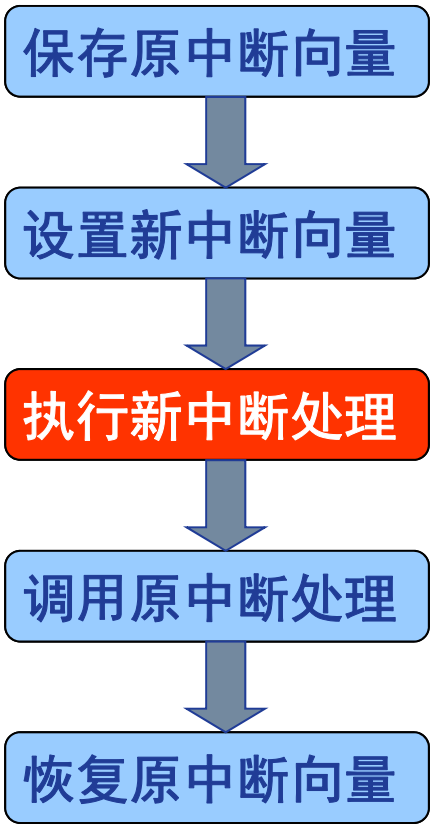

```
OLDISR DW ?,?

; ES = 0
MOV AX, ES:[N*4]
MOV OLDISR[0], AX
MOV AX, ES:[N*4+2]
MOV OLDISR[2], AX
```

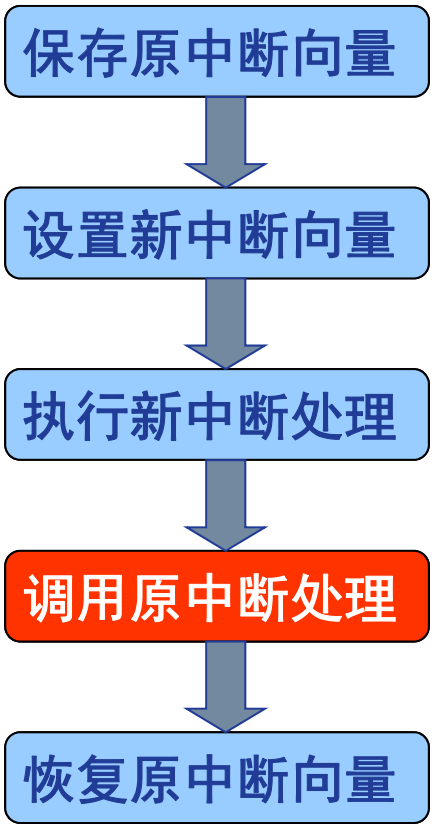
设置新中断向量




```
; ES = 0

MOV ES:[N*4], OFFSET ISR

MOV ES:[N*4+2], SEG ISR


; ISR新中断程序的开始处标号
```

执行新中断处理



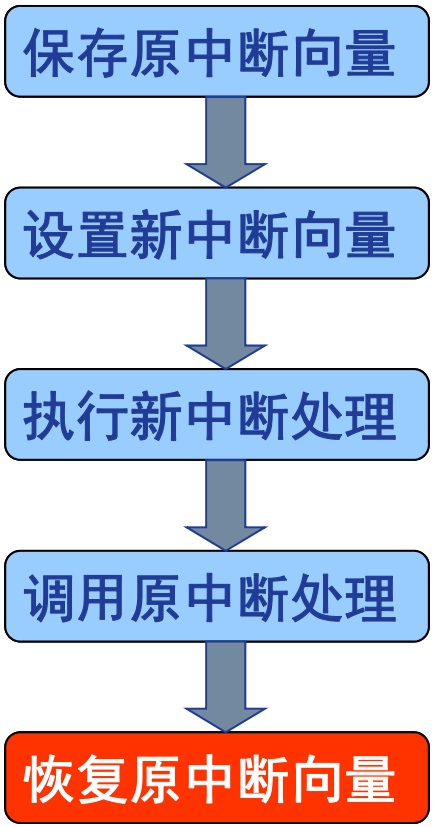


中断过程：

- ……
- PUSHF
- 保护断点：PUSH CS; PUSH IP
- 取中断向量，并执行

```
OLDISR DW ?,?

PUSHF
CALL DWORD PTR OLDISR
```

恢复原中断处理




```
OLDISR DW ?,?

; ES = 0
MOV AX, OLDISR[0]
MOV ES:[N*4], AX
MOV AX, OLDISR[2]
MOV ES:[N*4+2], AX
```

通过DOS中断修改中断

用户在编写自己的中断处理程序代替系统中的某个中断处理功能时，要注意保留原来的中断向量。程序结束时，要恢复原来的中断向量。可以使用dos功能调用来存取中断向量

- AH=25H——把DS:DX指向的中断向量放置到中断向量表中类型号为AL的中断向量处

```
MOV AX, SEG ISR
MOV DS,AX
MOV DX, OFFSET ISR    ；DS:DX=中断程序入口地址
MOV AL, INTNO   ；中断向量号
MOV AH, 25H
INT 21H
```

- AH=35H——把类型号为AL的中断向量取出到ES:BX中

```
MOV AL, INTNO
MOV AH,35H
INT 21H          ;ES:BX= 中断程序入口地址
```

例5.15a：1/5

```
        .data
intoff  dw ?                ;保存原偏移地址
intseg  dw ?                ;保存原段地址
        .code
        mov ax,3580h        ;获取中断80H的中断向量
        int 21h
        mov intoff,bx       ;保存偏移地址
        mov intseg,es       ;保存段基地址
```

获取中断向量：AH＝35H（INT 21H）

入口参数：AL＝中断向量号

出口参数：ES:BX＝中断服务程序的入口地址（段基地址：偏移地址）

```
push ds
mov dx,offset newint80h
mov ax,seg newint80h
mov ds,ax
mov ax,2580h        ;设置中断80H的入口地址
int 21h
pop ds
```

设置中断向量：AH＝25H（INT 21H）

入口参数：AL＝中断向量号

ES:BX＝中断服务程序的段基地址：偏移地址

例5.15a：3/5

```
int 80h
;调用中断80h的服务程序，显示信息
mov dx,intoff      ;恢复中断80H的入口地址
mov ax,intseg
mov ds,ax
mov ax,2580h
int 21h
.exit 0            ;返回DOS
```

设置中断向量：AH＝25H（INT 21H）

入口参数：AL＝中断向量号

ES:BX＝中断服务程序的段基地址：偏移地址

```
newint80h   proc                ;内部中断服务程序
            sti                 ;开中断
            push ax             ;保护现场
            push bx
            push cx
            push si
            mov si,offset intmsg    ;获取显示字符串首地址
            mov cx,sizeof intmsg    ;获取显示字符串个数
```

例5.15a：5/5

```
disp:       mov al,cs:[si]      ;获取显示字符
            mov bx,0            ;显示一个字符
            mov ah,0eh
            int 10h             ;ROM-BIOS功能调用
            inc si
            loop disp
            pop si              ;恢复现场
            pop cx
            pop bx
            pop ax
            iret                ;中断返回
intmsg      db 'I am Great !',0dh,0ah       ;显示信息
newint80h   endp
```

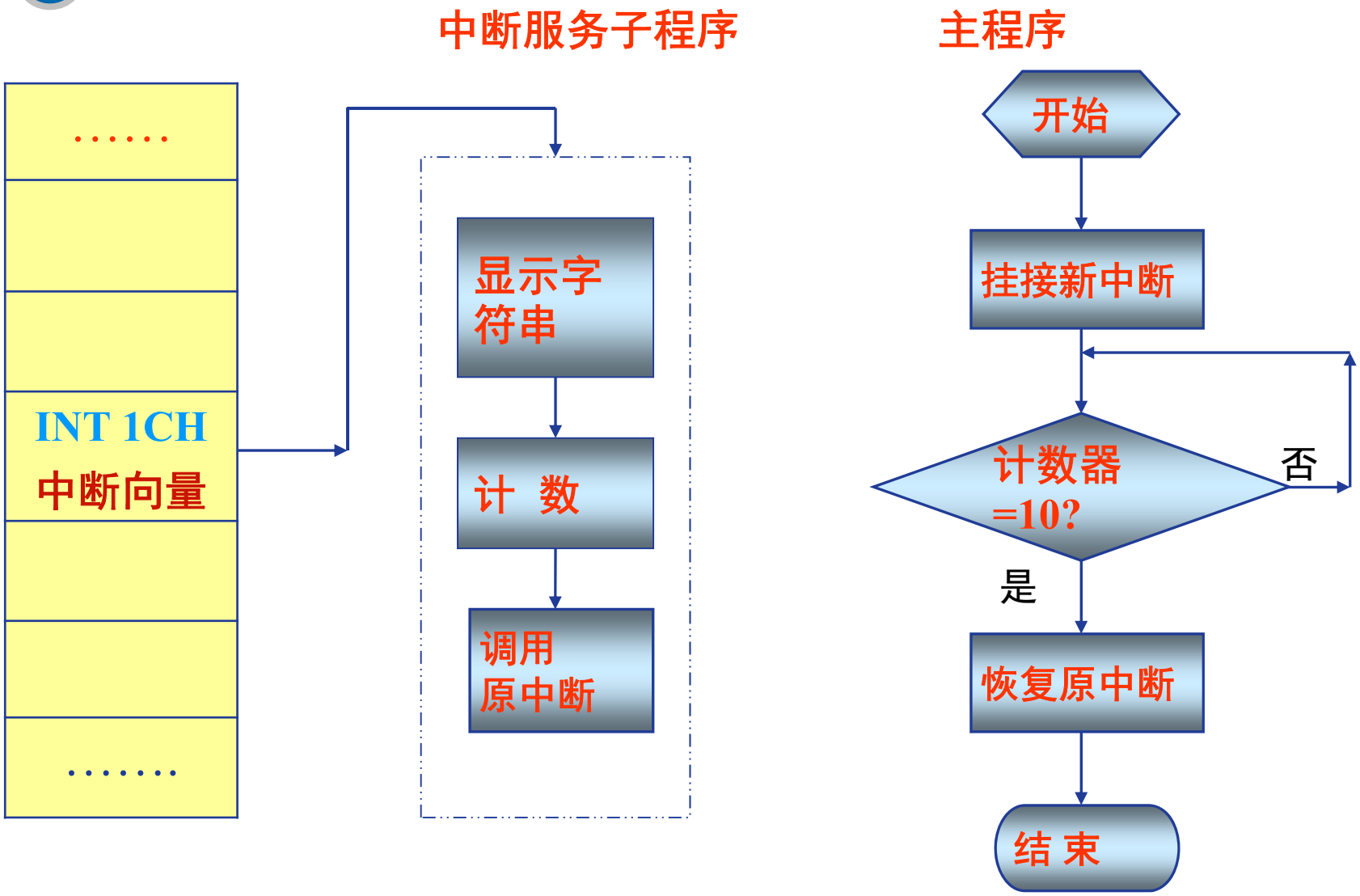
举例：定时器实现

- 定时中断 —BIOS INT 08H

  系统加电初始化后，定时器每隔约55毫秒发出一次中断请求。

- INT 1CH ：BIOS提供的8H号中断处理程序中有一条中断指令INT 1CH,所以每秒要调用到约18.2次1CH号中断处理程序。（1000/55）

- 例子
  - 挂接INT 1CH，显示10次字符串
  - 挂接INT 1CH，从30倒计时到0

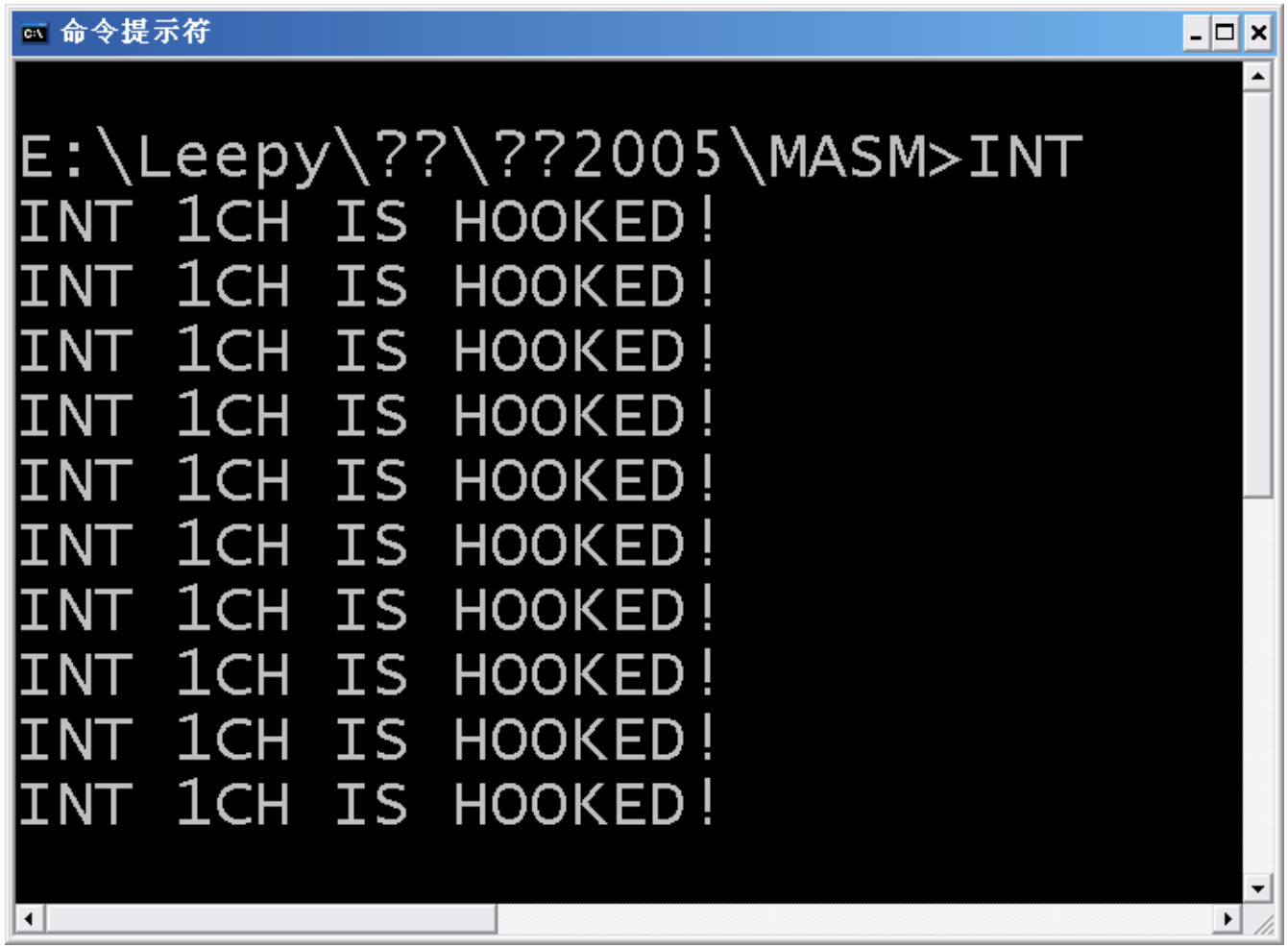

中断服务子程序
主程序
.....
INT 1CH
中断向量
.....
显示字符串
计 数
调用原中断
开始
挂接新中断
计数器=10?
否
是
恢复原中断
结 束

```
DATA SEGMENT
        STRING DB 'INT 1CH IS HOOKED! ',0DH,0AH,'$'
        OLDISR DW ?,?
        TIMER DB 0
DATA ENDS
CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
START:MOV AX,DATA
        MOV DS,AX
        MOV AX,0
        MOV ES,AX

        MOV AX, ES:[1CH*4]
        MOV OLDISR[0], AX
        MOV AX,ES:[1CH*4+2]
        MOV OLDISR[2], AX
```

保存原中断向量

运 行 结 果

- Timer.asm



第5章 教学要求

1. 理解条件控制和循环控制伪指令，熟悉带参数的过程定义和过程声明、过程调用伪指令
2. 了解宏操作符、宏汇编、条件汇编和重复汇编、源程序包含、代码连接和子程序库等程序设计方法
3. 了解程序直接控制、查询和中断的输入输出程序设计
4. 掌握伪指令：PROTO / INVOKE，MACRO/ENDM、LOCAL，INCLUDE / PUBLIC / EXTERN
5. 作业：5.10/5.12/5.15/5.18/5.27