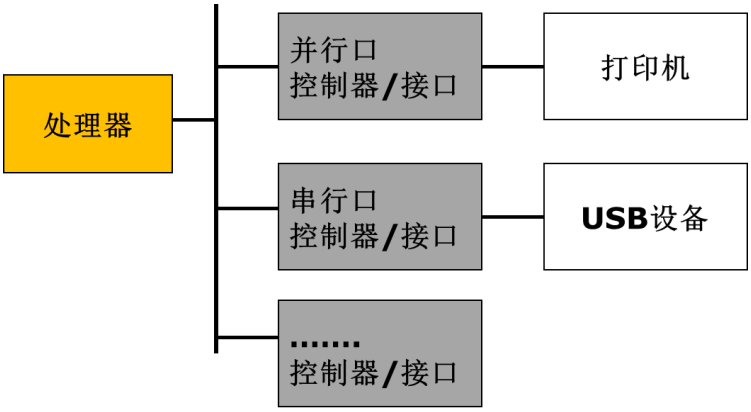
第 8 章　输入输出和中断

8.1　输入输出的基本概念

8.1.1　I/O 基本概念

外设连接



输入/输出

站在处理器或主机立场上而言的输入/输出

处理器访问（存取）接口上的特定的一组寄存器

I/O 端口地址

为了存取接口上的寄存器，系统给这些寄存器分配专门的存取地址，这样的地址被称为 I/O 端口地址。

两类端口地址

I/O 端口地址和存储单元地址统一编址

I/O 端口地址和存储单元地址各自独立编址

IA-32 系列处理器支持独立的 I/O 端口地址空间

地址空间达 64K

实际的系统中，只用很小的一部分

直接显示输出

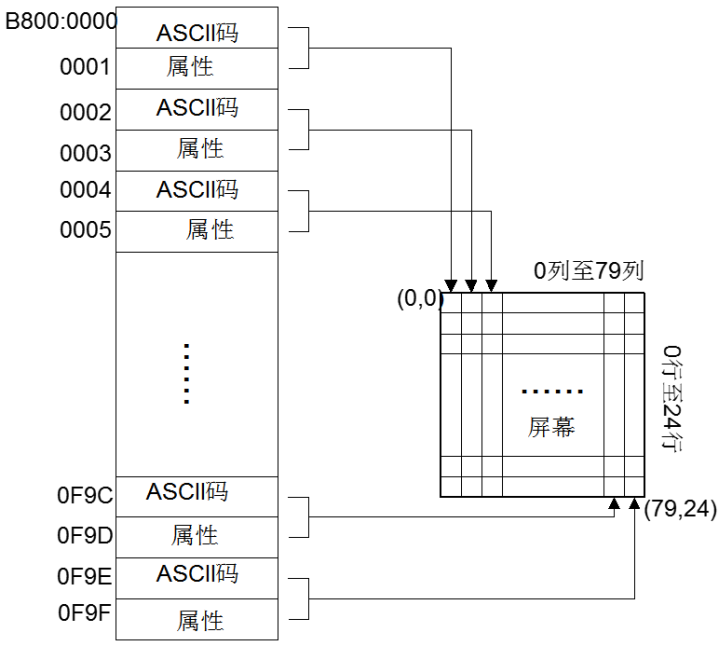
显示缓冲区

显示缓冲区指存储被显示信息的内存区域。

典型的显示缓冲区地址是 B800:0000H 开始的内存区域。

一种典型的文本显示方式，分辨率为 80X25，每屏 25 行，每行 80 列。

文本 80X25 模式下显缓与位置对应关系



在屏幕左上角(0，0)显示字母 A　直接写屏

```
MOV     AX, B800H
MOV     DS, AX
MOV     BX, 0
MOV     AL, 'A'
MOV     AH, 07H
MOV     [BX], AX
```

取得屏幕右下角(79,24)所显示字符的代码及属性　直接读屏

```
MOV     AX, B800H
```

```
MOV     DS, AX
MOV     BX, (80*24+79)*2
MOV     AX, [BX]
```

演示程序 dp81

采用直接写屏方式，在屏幕指定位置显示字符串“Hello,world”

主引导记录（MBR）形式的源程序

设置显示开始位置和获取被显示字符串地址信息

循环填写显示缓冲区（显示）

不足 510 字节

MBR 标记

源程序

```
ROW         EQU     5                       ;开始行号
COLUMN      EQU     8                       ;开始列号
COLOR       EQU     0x47                    ;显示字符属性（红底白字）
    ;
    section     text
    bits     16
BEGIN:
    MOV     AX, 0B800H                  ;显示缓冲区的段值
    MOV     ES, AX                      ;ES=显示缓冲区段
    MOV     DI, (ROW*80+COLUMN)*2 ;DI=开始显示位置
    MOV     AX, CS
    MOV     DS, AX                      ;DS=CS
    MOV     SI, hello                   ;指向字符串首（代码段的相对地址）
ADD     SI, 7C00H                   ;指向字符串首（内存中的固定地址）
    MOV     AH, COLOR                   ;AH=显示属性
NEXT:
    MOV     AL, [SI]                    ;取一个字符
    INC     SI
    ;
    OR      AL, AL                      ;判断结束标记
    JZ      OVER                        ;是，跳转结束
    ;
    MOV     [ES:DI], AX                 ;显示（填到显示缓冲区）
    ADD     DI, 2
    JMP     NEXT                        ;继续
OVER:
    JMP     OVER                        ;进入无限循环
hello     db      "Hello,world",0
          ;
          times     510 - ($ - $$) db     0     ;填充 0，直到 510 字节
          db      55h, 0aah                       ;最后 2 字节，共计 512 字节
```

8.1.2 I/O 指令

I/O 指令

专门用于存取独立编址端口的指令

归入数据传送指令组

输入指令

输入指令的一般格式：IN　　累加器，端口地址

输入指令从某个指定端口，读取 8 位、16 位或 32 位，传送至累加器 AL、AX 或 EAX（分别对应 8 位、16 位或 32 位）。

端口地址可采用直接方式表示，也可采用间接方式表示。

当采用直接方式表示端口地址时，端口地址仅为 8 位，即 0 至 255；

当采用间接方式表示端口地址时，端口地址存放在 DX 寄存器中，端口地址可为 16 位。

```
IN    AL，21H          端口 20H 的值送到 AL

IN    AX，20H          端口 20H 的值送到 AL  端口 21H 的值送到 AH

MOV   DX, 2FCH
IN    AL，DX
IN    AX，DX                端口 2FCH 的值送到 AL  端口 2FDH 的值送到 AH
IN    EAX，DX
```

输出指令

输出指令的一般格式：OUT　　端口地址，累加器

输出指令把累加器 AL、AX 或 EAX（分别对应 8 位、16 位或 32 位）输出到某个指定端口。

端口地址可采用直接方式表示，也可采用间接方式表示。

当采用直接方式表示端口地址时，端口地址仅为 8 位，即 0 至 255；

当采用间接方式表示端口地址时，端口地址存放在 DX 寄存器中，端口地址可为 16 位。

```
    MOV    AL, 13H        向端口 20H，输出值 13H
    OUT    20H, AL

    MOV    AX,1234H
    MOV    DX, 2FCH
    OUT    DX, AL
OUT    DX, AX
```

CPU 与外设之间交换的信息

CPU 与外设之间交换的信息包括三类：

数据

控制信息

状态信息

这三类信息具有不同性质，但它们都通过 IN 和 OUT 指令在数据总线上进行传送，通常采用分配不同端口的方法将它们加以区别。

数据是 CPU 和外设真正要交换的信息。数据可以 8 位、16 位或 32 位，可分为各种不同类型。不同外设要传送的数据类型也是不同的。

控制信息输出到 I/O 接口，告诉接口和设备要做什么工作。

从接口输入的状态信息表示 I/O 设备当前的状态。在输入数据前，通常要先取得表示设备是否已准备好的状态信息；在输出数据前，往往要先取得表示设备是否忙的状态信息。

8.1.3 数据传送方式

数据传送方式

CPU 与外设之间交换的信息包括三类：

无条件传送方式

查询方式

中断方式

直接存储器传送(DMA)方式

无条件传送方式

在不需要查询外设的状态，即已知外设已准备好或不忙时，可以直接使用 IN 或 OUT 指令实现数据传送。

这种方式软件实现简单，只要在指令中指明端口地址，就可选通指定外设进行输入输出。但要求外设工作速度能与 CPU 同步，否则就可能出错。

8.1.4 存取 RT/CMOS RAM

关于 RT/CMOS RAM

PC 机上安装有一个 RT/COMS RAM 芯片，它是互补金属氧化物半导体随机存取存储器，不仅可长期保存系统配置状况，而且记录包括世纪、年、月、日和时分秒在内的实时钟(Real_Time Clock)信息。

RT/CMOS RAM 作为一个 I/O 接口芯片，系统分配的 I/O 端口地址区为 70H 至 7FH，通过 IN 和 OUT 指令可对其进行存取。

RT/CMOS RAM 提供 64 个字节 RAM 单元，分配使用情况如表所示。前 14 个字节用于实时钟，剩下的 50 个字节用于系统配置。

位移	用　　途	位移	用　　途
0	秒	10	软盘驱动器类型
1	报警秒	11	保留
2	分	12	硬盘驱动器类型
3	报警分	13	保留
4	时	14	设备标志
5	报警时	15	常规 RAM 容量低字节
6	星期	16	常规 RAM 容量高字节
7	日	17	扩展 RAM 容量低字节
8	月	18	扩展 RAM 容量高字节
9	年	19-1A	硬驱类型扩展字节
A-D	状态寄存器 A 至 D	1B-2D	保留
E	诊断状态	2E-2F	配置信息字节累加和
F	停止状态	30-3F	其它(含世纪信息)

存取 RT/CMOS RAM

存取方法

分两步存取 RT/CMOS RAM 芯片内部的 64 个字节内容：

把要存取单元的地址送端口 70H

存取端口 71H

注意事项：14 个记录实时钟信息的单元(位移 0 置 0DH)的地址就是表中位移，其他单元的地址是表所示位移

上加 80H。

读操作代码片段

```
MOV    AL, n          ;n 是要访问单元地址   第一步:确定要存取单元
OUT    70H, AL        ;把要访问单元的地址送地址端口
JMP    short $+2      ;延时
IN     AL, 71H        ;从数据端口取访问单元的内容   第二步:存取指定的单元
```

写操作代码片段

```
MOV    AL, n          ;n 是要访问单元地址      第一步:确定要存取单元
OUT    70H, AL        ;把要访问单元的地址送地址端口
JMP    short  $+2     ;延时
MOV    AL, m          ;m 是要输出数据
OUT    71, AL         ;把数据从数据端口输出   第二步:存取指定的单元
```

8.2 查询方式传送数据

8.2.1 查询传送方式

查询传送方式

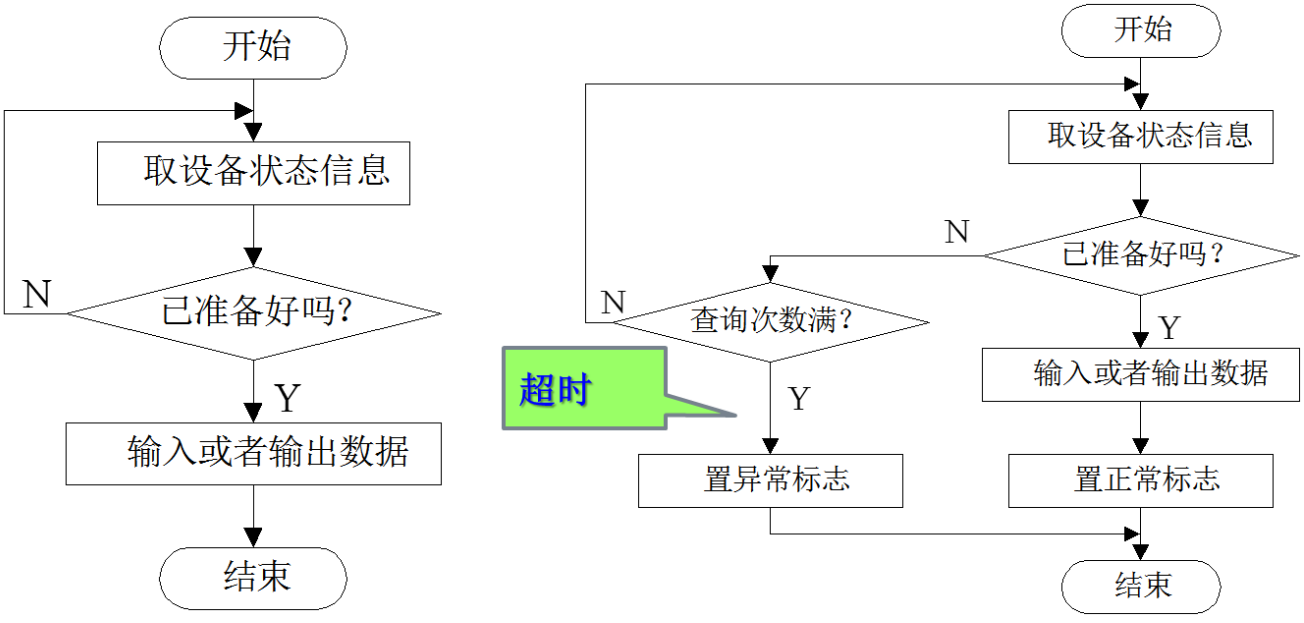
查询方式的基本思想：由 CPU 主动地通过输入输出指令查询指定的外部设备的当前状态，若设备就绪，则立即与设备进行数据交换，否则循环查询。

在输入之前，要查询外设的数据是否已准备好，直到外设把数据准备好后才输入；在输出之前，要查询外设是否“忙”，直到外设不“忙”后才输出。

查询传送方式适用于 CPU 与外设不同步的情况。

通常外设速度远远慢于 CPU 速度，于是查询过程就将花费大量的时间。

查询方式输入输出的示意流程



查询传送方式的特点

为了采用查询方式输入或输出，相应的外设(或接口)不仅要有数据寄存器，而且还要有状态寄存器，有些外设还需要控制寄存器。

数据寄存器用来存放要传送的数据，状态寄存器用来存放表示设备所处状态的信息。通常，在状态寄存器中有一个“就绪(Ready)”位或一个“忙(Busy)”位来反映外设是否已准备好。

查询方式的优点是：软硬件实现比较简单。缺点是浪费了 CPU 原本可执行大量指令的时间。

8.2.2 读实时钟

读实时钟

计时更新标志：RT/CMOS RAM 的状态寄存器 A 的位 7 是计时更新标志位。为 1 表示实时钟正在计时；为 0 表示实时钟信息可用于读出。在读实时钟前，要判别该标志位是否为 0。

把更新标志位理解为状态寄存器中的“就绪”位，采用查询方式检测是否就绪。

演示程序 dp82.asm

写一个程序，显示当前时间。用户按键后，更新时间，直到用户按回车键。

主要步骤：

[1]查询是否可读实时时钟

[2]读实时时钟（时、分、秒）

[3]显示时间值

[4]等待用户按键

[5]如果非回车键，跳转到[1]

源程序

```
CMOS_PORT    EQU    70H          ;CMOS 端口地址
CMOS_REGA    EQU    0AH          ;状态寄存器 A 地址
UPDATE_F     EQU    80H          ;更新标志位
CMOS_SEC     EQU    00H          ;秒单元地址
CMOS_MIN     EQU    02H          ;分单元地址
CMOS_HOUR    EQU    04H          ;时单元地
    ;
    section    text
    bits    16          16 位段模式
    org     100H        COM 类型可执行程序从 100H 开始
    ;
    MOV    AX, CS    数据段同代码段
    MOV    DS, AX
NEXT:
UIP:
    MOV    AL, CMOS_REGA       ;判是否可读实时钟
    OUT    CMOS_PORT, AL       ;准备读状态寄存器 A
    JMP    SHORT $+2
    IN     AL, CMOS_PORT+1     ;读状态寄存器 A              查询方式判断可否读取时间
    TEST   AL, UPDATE_F        ;测更新标志
    JNZ    UIP                 ;如不可读则继续测试
    ;
    MOV    AL, CMOS_SEC
    OUT    CMOS_PORT, AL
    JMP    SHORT $+2
    IN     AL, CMOS_PORT+1     ;读秒值          直接获取时间的秒值
    MOV    [second], AL        ;保存之
    MOV    AL, CMOS_MIN
    OUT    CMOS_PORT, AL
    JMP    SHORT $+2
    IN     AL, CMOS_PORT+1     ;读分值
    MOV    [minute], AL        ;保存之
```

```
    ;
    MOV    AL, CMOS_HOUR
    OUT    CMOS_PORT, AL
    JMP    SHORT $+2
    IN     AL, CMOS_PORT+1    ;读时值
    MOV    [hour], AL         ;保存之
    MOV    AL, [hour]
    CALL   EchoBCD            ;显示时值
    MOV    AL, ':'
    CALL   PutChar            ;显示间隔符      显示时间（时:分:秒）
    MOV    AL, [minute]
    CALL   EchoBCD            ;显示分值
    MOV    AL, ':'
    CALL   PutChar            ;显示间隔符
    MOV    AL, [second]
    CALL   EchoBCD            ;显示秒值
    ;
    MOV    AL, 0DH            ;形成回车换行
    CALL   PutChar
    MOV    AL, 0AH        形成回车换行
CALL   PutChar
    MOV    AH, 0              ;等待并接受用户按键
    INT    16H
    CMP    AL, 0DH            ;如果按回车键，结束
    JNZ    NEXT
    ;
    MOV    AH, 4CH            ;结束程序
    INT    21H                ;假设 DOS 环境下运行
    ;-----------------
PutChar:                      ;TTY 方式显示一个字符
    MOV    BH, 0
    MOV    AH, 14         调用 BIOS：TTY 方式显示一个字符
    INT    10H
    RET
EchoBCD:
    PUSH   AX
    SHR    AL, 4              ;把高位 BCD 码转成 ASCII 码  显示两位 BCD 码的值 AL=BCD 码
    ADD    AL, '0'
    CALL   PutChar            ;显示之
    POP    AX
    ;
    AND    AL, 0FH            ;把低位 BCD 码转成 ASCII 码
    ADD    AL, '0'
    CALL   PutChar            ;显示之
    RET
second   DB    0              ;秒数保存单元
minute   DB    0              ;分数保存单元
hour     DB    0              ;时数保存单元
```

8.2.3 查询方式打印输出

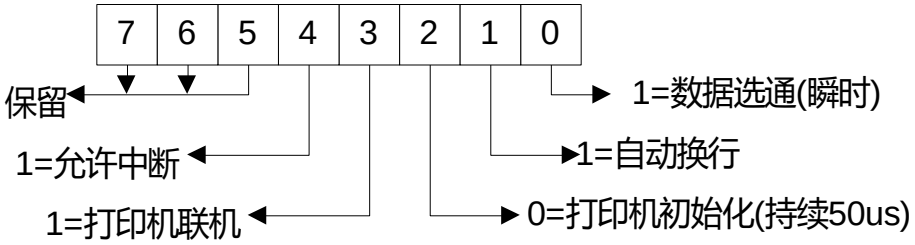
打印输出的并行接口

假设打印机通过打印接口(并行口)连入系统。

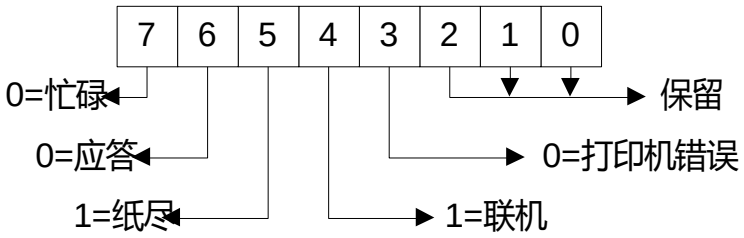
打印接口的功能是传递打印命令和数据到打印机并返回打印机状态。打印接口包含数据寄存器、状态寄存器和控制寄存器。

三个寄存器有各自的端口地址，并且三个端口地址是连续的。设数据寄存器端口地址是 378H，那么，状态寄存器端口地址是 379H，控制寄存器端口地址是 37AH。

打印接口的状态寄存器和控制寄存器各位的定义



(a)控制寄存器各位定义



(b)状态寄存器各位定义

三个寄存器有各自的端口地址，并且三个端口地址是连续的。设数据寄存器端口地址是 378H，那么，状态寄存器端口地址是 379H，控制寄存器端口地址是 37AH

查询方式打印一个字符的流程

查询方式打印一个字符的子程序

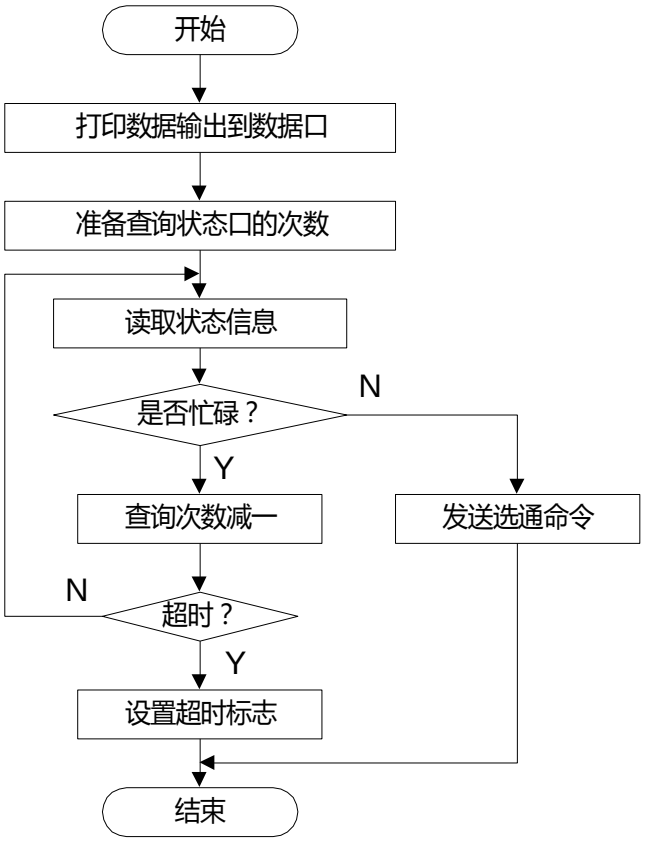
可以假设：

数据寄存器端口 378H

状态寄存器端口 379H

控制寄存器端口 37AH

```
;子程序名：PRINT
;功      能：打印一个字符
;入口参数：DX=数据寄存器端口地址
;          BL=超时参数
;          AL=打印字符的代码
;出口参数：AH=打印机状态，各位意义如下：
;          位 0：1 表示超时，即超过规定的查询次数
;          位 1 和位 2：不用
;          位 3：1 表示出错
;          位 4：1 表示联机
;          位 5：1 表示无纸
```

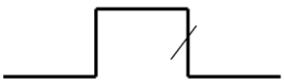


```
;               位 6：1 表示应答
;               位 7：0 表示忙碌
```

演示程序 dp81

源程序

```
PRINT:
    PUSH   DX
    PUSH   AX
    OUT        DX, AL      ;输出打印数据
    INC    DX              ;使 DX 含状态寄存器端口地址
WAIT:
    XOR    CX, CX      ;1 个超时参数单位表示查询 65536 次
WAIT1:
    IN     AL, DX      ;读取状态信息
    MOV    AH, AL      ;保存到 AH
    TEST   AL, 80H     ;测是否忙碌
    JNZ    NEXT        ;不忙碌，则转
    LOOP   WAIT1       ;继续查询   查询等待：二重循环，每个超时参数单位查询 65536 次。
    DEC        BL          ;超时参数减 1
    JNZ    WAIT        ;未超时，继续查询
    ;
    AND    AH, 0F8H   ;已超时，去掉状态信息中的无用位
    OR     AH, 1       ;置超时标志
JMP        EXIT        ;转结束   超时处理：准备参数，并返回
NEXT:
    INC    DX              ;不忙碌，使 DX 含控制寄存器端口地址
    MOV        AL, 0DH         ;准备选通命令
    OUT        DX, AL          ;选通
    MOV        AL, 0CH         ;准备复位选通命令
    JMP        short $+2
    OUT        DX, AL          ;复位选通位
    AND        AH, 0F8H        ;去掉状态信息中的不用位
EXIT:
    XOR        AH, 48H         ;使返回的状态消息中有关位符合要求
    POP        DX
    MOV        AL, DL          ;恢复 AL 寄存器值
    POP        DX
    RET
```



00001101

00001100

选通操作：高电平到低电平有效

8.3 中断及其处理

8.3.1 中断和中断传送方式

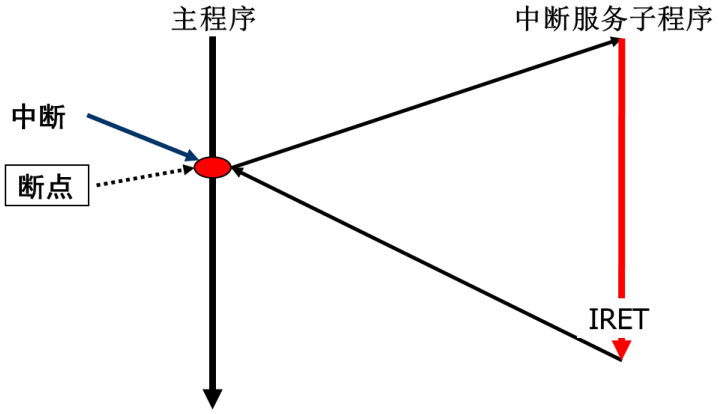
中断

中断是一种使 CPU 挂起正在执行的程序而转去处理特殊事件的操作。

这些引起中断的事件称为中断源

来自 CPU 外部。例如：外设的输入输出请求（如，由按键引起的键盘中断；又如由串行口接收到信息引起的串行口中断等）

来自 CPU 内部的一些异常事件。例如：除数为 0，等。



中断传送方式

中断传送方式的具体过程是：当 CPU 需要输入或输出数据时，先作一些必要的准备工作(有时包括启动外部设备)，然后继续执行程序；当外设完成一个数据的输入或输出后，则向 CPU 发出中断请求，CPU 就挂起正在执行的程序，转去执行输入或输出操作，在完成输入或输出操作后，返回原程序继续执行。

8.3.2 中断向量表

中断向量

IA-32 处理器共能支持 256 种类型的中断，给每一种中断安排一个中断类型号（简称为中断号），中断号依次为 0～0FFH。例如：属于内部中断的除法出错中断号为 0；初始情况下属于外部中断的定时器中断号为 08，键盘中断号为 09。

处理中断（响应中断）的程序被称为中断处理程序，或者中断响应程序，或者中断响应处理程序。每种不同类型的中断，都有对应的中断处理程序。

中断处理程序的开始地址被称为中断向量。地址就是指针，指针的图示是箭头，就是向量（矢量）。

中断向量表

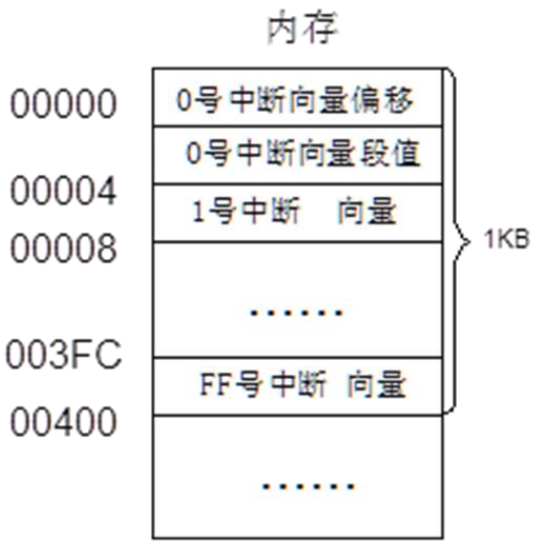
每种类型的中断都由相应的中断处理程序来处理，为了使系统在响应中断后，CPU 能快速地转入对应的中断处理程序，系统用一张表来保存这些中断处理程序的入口地址（中断向量），该表就称为中断向量表。

中断向量表的每一项是一个中断向量，也就是一个中断处理程序的入口地址。

中断向量表中的中断向量也依次编号为 00～FF。n 号中断向量就保存处理中断号为 n 的中断处理程序的入口地址。一般不再区分中断号（中断类型号）和中断向量号。

实方式下的中断向量表位于内存最低端的 1K 字节空间中。

每个中断向量占用四个字节，前(低地址)两字节保存中断处理程序入口地址的偏移，后(高地址)两字节保存中断处理程序入口地址的段值。



含有 256 个中断向量的中断向量表需要占用 1K 字节内存空间。

中断向量号的安排

在系统中，某个中断号分配给哪个中断，即某个中断向量含有哪个中断处理程序的入口地址，存在一些规定和约定。应用程序不能违反规定，不宜不遵守约定。

0	除法出错	8	定时器

1	单步	9	键盘
2	非屏蔽中断	A	保留(从中断控制器)
3	断点	B	串行通信接口 2
4	溢出	C	串行通信接口 1
5	保留（打印屏幕）	D	硬盘(并行口)
6	保留	E	软盘
7	保留	F	打印机

PC 机建立 BIOS 后中断号分配

10	显示 I/O	17	打印 I/O
11	设备设置	18	ROM BASIC
12	存储容量	19	系统自举
13	磁盘 I/O	1A	时钟管理
14	串口 I/O	1B	Ctrl+Break 键处理
15	扩充的 BIOS	1C	定时处理
16	键盘 I/O	1D-1F	参数指针
20-2F	DOS 使用	30-3F	为 DOS 保留

中断向量不一定非要指向中断处理程序，也可作为指向一组数据的指针。

访问中断向量表

把编号为 n 的中断向量，保存到双字单元 vector 中

```
XOR   AX, AX
MOV   ES, AX
MOV   AX, [ES:n*4]
MOV   [vector], AX
MOV   AX, [ES:n*4+2]
MOV   [vector+2], AX
```

16位处理

```
XOR   AX, AX
MOV   ES, AX
MOV   EAX, [ES:n*4]
MOV   [vector], EAX
```

32位处理

8.3.3 中断响应过程

实方式下中断响应过程

通常，CPU 在执行完每一条指令后均要检测是否有中断请求，在有中断请求且满足一定条件时就响应中断。

相关概念

内部中断

外部中断：
可屏蔽中断
不可屏蔽中断

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

相关概念

中断允许标志 IF（Interrupt Enable Flag）

为 1 时，允许可屏蔽中断

为 0 时，禁止可屏蔽中断

单步标志 TF（Trap Flag）

为 1 时，进入单步

为 0 时，正常

实方式下中断响应过程

中断响应时，硬件还自动完成：

(1)取得中断类型号；

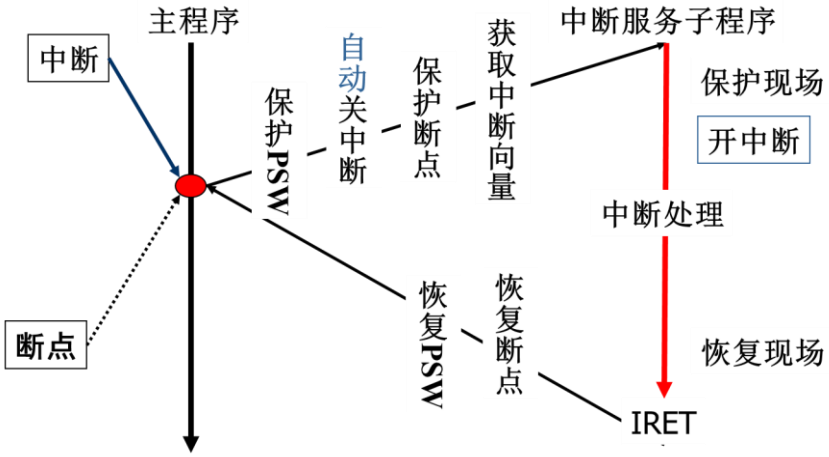
(2)把标志寄存器内容压入堆栈；

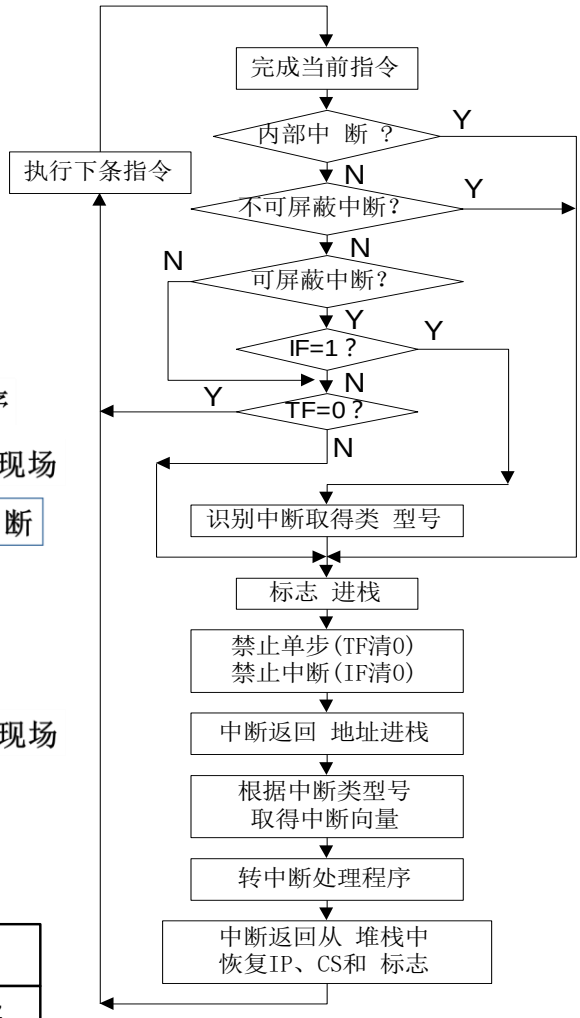
(3)禁止外部中断和单步中断(使 IF 和 TF 标志位为 0)；

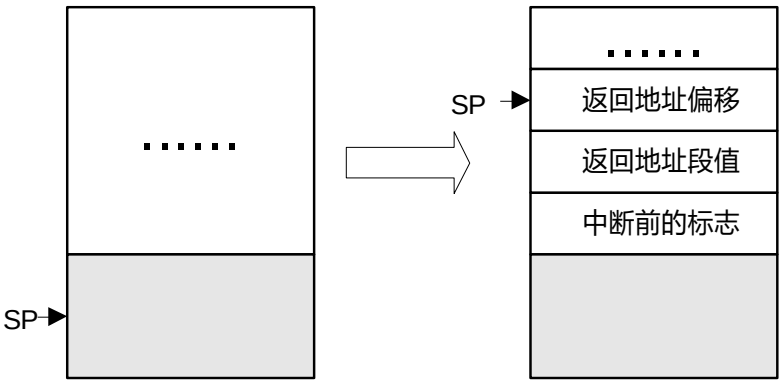
(4)把下一条要执行指令的地址(中断返回地址)压入堆栈(CS 和 IP 内容压入堆栈)；

(5)根据中断号从中断向量表中取中断处理程序入口地址；

(6)转入中断处理程序。







(a)中断前的堆栈

(b)中断后的堆栈

响应中断时，
压入堆栈 3 个字，分别是：
返回地址的偏移
返回地址的段值
标志寄存器值

中断返回指令

中断返回指令格式：IRET

执行指令实现从中断返回。实方式下的具体操作如下：

```
IP <= [SP]
SP <= SP+2
CS <= [SP]
SP <= SP+2
FLAGS <= [SP]
SP <= SP+2
```

弹出返回地址的偏移到 IP
弹出返回地址的段值到 CS
弹出标志值到标志寄存器
中断处理程序通常利用中断返回指令
从堆栈中弹出返回地址和原标志值。
平衡堆栈！

开中断和关中断指令

开中断指令格式：STI

效果：开中断。从而响应（不屏蔽）可屏蔽中断。

关中断指令格式：CLI

效果：关中断。从而不响应（屏蔽）可屏蔽中断。

8.3.4 内部中断

关于内部中断

由发生在 CPU 内部的某个事件引起的中断称为内部中断。由于内部中断是 CPU 在执行某些指令时产生，所以也称为软件中断。

内部中断的特点：不需要 CPU 外部硬件的支持；不受中断允许标志 IF 的控制。

中断指令 INT 引起的中断

中断指令的格式：INT n

其中，n 是一个 0 至 0FFH 的立即数。

CPU 在执行该中断指令后，便产生一个类型号为 n 的中断，从而转入对应的中断处理程序。

```
    MOV   AH, 0
    INT   16H          调用 16H 号中断处理程序（键盘 I/O 程序）

    MOV   AH, 1H
    INT   21H          调用 21H 号中断处理程序（DOS 系统功能）
```

示例

一个显示所按键 ASCII 码的程序

```
    section  text
    bits  16
    org   100h
begin:
```

```
    MOV    AH, 0
    INT    16H
    PUSH   AX
    SHR    AL, 4
    CALL   TOASCII
    MOV    AH, 14
    INT    10H
    POP    AX
    CALL   TOASCII
    MOV    AH, 14
    INT    10H
    MOV    AH,4CH
    INT    21H
    ;
TOASCII:
    AND    AL,0FH
    ......
RET
```

除法错（溢出）中断

在执行除法指令时，如果 CPU 发现除数为 0 或者商超过了规定的范围，那么就产生一个除法错（溢出）中断，中断类型号规定为 0。这是来自于 CPU 内部的中断。

```
MOV    AX, 1234
MOV    CL, 3
DIV    CL
```

单步中断

当标志寄存器中的单步标志 TF 为 1，则在每条指令执行后产生一个单步中断，中断类型号规定为 1。

产生单步中断后，CPU 就执行单步中断处理程序。

由于 CPU 在响应中断时，已把 TF 置为 0，所以，不会以单步方式执行单步中断处理程序。

通常，由调试工具把 TF 置 1，在执行完一条被调试程序的指令后，就转入单步中断处理程序。单步中断处理程序可以报告各寄存器的当前内容，程序员可据此调试程序。

断点中断

处理器提供一条特殊的中断指令“INT 3”。

调试工具可用它替换断点处的代码，当 CPU 执行这条中断指令后，就产生类型号为 3 的中断。这种中断称为断点中断。

通常，断点中断处理程序恢复被替换的代码，并报告各寄存器的当前内容，程序员可据此调试程序。

中断指令“INT 3”特殊是因为它只有一个字节长，其他的中断指令长 2 字节。

8.3.5 外部中断

关于外部中断

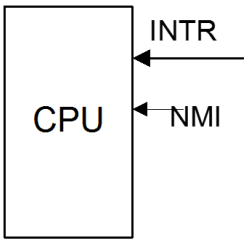
由发生在 CPU 外部的某个事件引起的中断称为外部中断。如，输入输出设备等引起的中断就是外部中断。

外部中断以完全随机的方式中断现行程序。

x86 处理器有两条外部中断请求线：

INTR 接受可屏蔽中断请求

NMI 接受非屏蔽中断请求



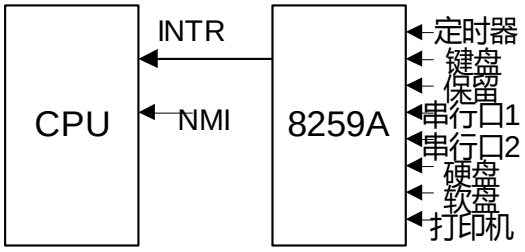
可屏蔽中断

可屏蔽中断指受到中断允许标志 IF 制约的外部中断。

在 PC 系列机中，键盘和硬盘等外设的中断请求都通过中断控制器 8259A 传给可屏蔽中断请求线 INTR。

中断控制器 8259A 共能接收 8 个独立的中断请求信号 IRQ0 至 IRQ7。

系统中，可能有两个中断控制器 8259A。一主一从，从 8259A 连接到主 8259A 的 IRQ2 上，这样系统就可接收 15 个独立的中断请求信号。



中断控制器

中断控制器在控制外设中断方面起着重要的作用。如果接收到一个中断请求信号，并且满足一定的条件，那么它就把中断请求信号传到 CPU 的可屏蔽中断请求线 INTR，使 CPU 感知到有外部中断请求；同时也把相应的中断类型号送给 CPU，使 CPU 在响应中断时可根据中断类型号取得中断向量，转相应的中断处理程序。

中断控制器 8259A 是可编程的。在初始化时规定了在传出中断请求 IRQ0 至 IRQ7 时，送出的对应中断类型号分别是 08H～0FH。例如，设传出中断请求 IRQ1，即传出键盘中断请求，那么送出的中断类型号为 9，所以键盘中断的中断类型号为 9，键盘中断处理程序的入口地址存放在 9 号中断向量中。

中断控制器 8259A 包含两个寄存器：中断屏蔽寄存器和中断命令寄存器，它们决定了传出一个中断请求信号的条件。

中断屏蔽寄存器的 I/O 端口地址是 21H，它的 8 位对应控制 8 个外部设备，通过设置这个寄存器的某位为 0 或为 1 来允许或禁止相应外部设备中断。当第 i 位为 0 时，表示允许传出来自 IRQi 的中断请求信号，当第 i 位为 1 时，表示禁止传出来自 IRQi 的中断请求信号。

```
MOV     AL,11111101B
OUT     21H,AL        使中断控制器 8259A 只传出来自键盘的中断请求信号
```

控制响应外部中断的方式

中断允许标志 IF

可编程中断控制器（8259A）

响应键盘中断的过程

如果用户敲键盘，则 IRQ1 引脚有信号

如果中断控制器允许（没有屏蔽）IRQ1，则 INTR 有信号，同时传出中断类型号 9（键盘中断）

如果开中断（没有屏蔽外部中断），则在执行当前指令后响应，进入键盘中断处理程序

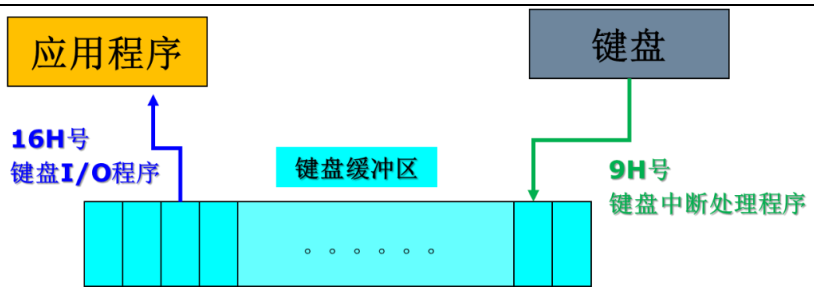
键盘（9H 号）中断处理程序，根据所按键的扫描码进行处理。按普通键的情况下，把字符键的扫描码和对应的 ASCII 码存到键盘缓冲区。

键盘三者（缓冲区、中断处理程序、IO 程序）的关系

仓　库：键盘缓冲区（内存某个区域）

生产者：键盘中断处理程序（9H 号中断处理程序）

消费者：键盘 I/O 程序（16H 号中断处理程序）



前台和后台

```
CONT:
    MOV    AH, 0
    INT    16H        ;主动读取字符
    ;
    MOV    AH, 14
    INT    10H        ;主动显示字符
    ;
    JMP    CONT       ;无限循环
```

8.3.6　中断优先级

中断优先级

系统中有多个中断源，当多个中断源同时向 CPU 请求中断时，CPU 按规定的优先级响应中断请求。

优先级最高　　内部中断(除法错，INT)

　　┃　　非屏蔽中断(NMI)

　　↓　　可屏蔽中断(INTR)

优先级最低　　单步中断

外设的中断请求都通过中断控制器 8259A 传给 CPU 的 INTR 引线。在对 8259A 初始化时规定了 8 个优先级，在正常的优先级方式下，优先级次序如下：

IRQ0，IRQ1，IRQ2，IRQ3，IRQ4，IRQ5，IRQ6，IRQ7

必要的情况下，通过设置中断控制器 8259A 中的中断命令寄存器的有关位可改变上述优先级次序。

中断嵌套

CPU 在执行中断处理程序时，又发生中断，这种情况称为中断嵌套。

外部中断的嵌套比较复杂。

由于 CPU 在响应中断的过程中，已自动关中断，CPU 也就不会再自动响应可屏蔽中断。如果需要在中断处理过程的某些时候响应可屏蔽中断，则可在中断处理程序中安排开中断指令，CPU 在执行开中断指令后，就处于开中断状态，也就可以响应可屏蔽中断了，直到再关中断。如果在中断处理程序中使用了开中断指令，也就可能会发生可屏蔽中断引起的中断嵌套。

8.3.7　中断处理程序设计

外设中断处理程序的设计

在开中断的情况下，外设中断的发生是随机的，在设计外设中断处理程序时必须充分注意到这一点。

外设中断处理程序的主要步骤如下：

(1)必须保护现场。保护的一般方法是把它们压入堆栈。

(2)尽快完成中断处理。外设中断处理必须尽快完成，所以外设中断处理必须追求速度上的高效率。

(3)恢复现场。

(4)通知中断控制器中断已结束。

(5)利用 IRET 指令实现中断返回。

应及时开中断。除非必要，中断处理程序应尽早开中断，以便 CPU 响应具有更高优先级的中断请求。

软件中断不会随机发生。中断指令类似于子程序调用指令，软中断处理程序在很大程度上类似于子程序，但并不等同于子程序。

软中断处理程序的主要步骤如下：

(1)考虑切换堆栈

(2)及时开中断

(3)应该保护现场

(4)完成中断处理

(5)恢复现场

(6)堆栈切换

(7)一般利用 IRET 指令实现中断返回

演示程序 dp85.asm

背景知识

系统定时器被初始化为每隔约 55 毫秒发出一次中断请求。CPU 在响应定时中断请求后转入 8H 号中断处理程序。

BIOS 提供的 8H 号中断处理程序中安排了一条中断指令“INT　1CH”，所以每秒要调用到约 18.2 次 1CH 号中断处理程序。

BIOS 的 1CH 号中断处理程序实际上并没有做任何工作，只有一条中断返回指令。这样安排的目的是为应用程序留下一个软接口，应用程序只要提供新的 1CH 号中断处理程序，就可能实现某些周期性的工作。

背景知识（二）

BIOS 提供的时钟管理程序以 1AH 号中断处理程序的形式存在。

调用 1AH 号中断处理程序的 2 号功能，可以获取当前时间。该功能在 CH、CL 和 DH 寄存器中返回时、分和秒的 BCD 码。

写一个时钟程序，显示当前时间，每秒自动更新。同时，程序接受用户按键，当用户按‘!’时，结束程序运行。假设 DOS 运行环境。

设计思路

前台程序替换定时（1CH 号）中断处理程序，随后，接受用户按键，遇到‘！’时，恢复被替换的 1CH 号中断处理程序，并结束运行。

新的定时中断处理程序，每隔约 1 秒时间，更新屏幕上显示的时间。它调用 1AH 软中断取得时间值。

定时（1CH 号）中断处理程序，相当于后台程序。

源程序

```
ROW             EQU     10              ;开始行号
COLUMN          EQU     18              ;开始列号   显示时间值的屏幕位置
;
    section     text
    bits    16
    org     100H            COM 类型可执行程序从 100H 开始
    ;
Begin:
    MOV     AX, CS              替换定时（1CH 号）中断向量先保存原中断向量
    MOV     DS, AX                ;DS = CS
    MOV     SI, 1CH*4             ;1CH 号中断向量所在地址
    MOV     AX, 0
    MOV     ES, AX                ;ES = 0
    ;                           ;保存 1CH 号中断向量
```

```
    MOV    AX, [ES:SI]
    MOV    [old1ch], AX        ;保存向量之偏移       保存原先的中断向量
    MOV    AX, [ES:SI+2]
    MOV    [old1ch+2], AX      ;保存向量之段值
    ;                          ;设置新的 1CH 号中断向量
    CLI                        ;关中断
    MOV    AX, Entry_1CH
    MOV    [ES:SI], AX         ;设置新向量之偏移
    MOV    AX, CS
    MOV    [ES:SI+2], AX       ;设置新向量之段值    设置新的中断向量
    STI                        ;开中断
Continue:
    MOV    AH, 0                   接受用户按键，并显示之不是‘!’键，则继续
    INT    16H             ;等待并接受用户按键
    ;
    CMP    AL, 20H
    JB     Continue        ;不可显示字符，就不显示
    ;
    MOV    AH, 14
    INT    10H             ;显示所按字符
    ;
    CMP    AL, '!'
    JNZ    Continue        ;只要不是'!'，继续等待并接受按键
    ;
Stop:
    ;恢复原 1C 号中断向量          恢复原先的定时中断向量结束程序运行
    MOV    EAX, [CS:old1ch]  ;获取保存的原 1CH 号中断向量
    MOV    [ES:SI], EAX        ;恢复原 1CH 号中断向量
    ;
    MOV    AH, 4CH             ;结束程序，返回操作系统（DOS）
    INT    21H
    ;
;=====================================
Entry_1CH:              新的定时（1CH 号）中断处理程序
    DEC    BYTE   [CS:count]  ;计数器减 1     计数（约 1 秒刷新一次）刷新时间值
    JZ     ETIME               ;当计数为 0，显示时间
    IRET                       ;否则，中断返回
    ;
ETIME:                         ;显示当前时间
    MOV    BYTE [CS:count], 18    ;重新设置计数初值
    STI                        ;开中断
    CALL   EchoTime            ;显示当前时间
    IRET                       ;中断返回
    ;----------------------
EchoTime:                      ;获得并显示当前时间（时分秒）
    PUSH   DS                  ;保护 DS
    PUSHA                      ;保护通用寄存器
    MOV    AX, CS
```

```
    MOV     DS, AX                  ;DS = CS
    ;
                                        MOV     AH, 2  调用 1AH 号中断处理程序的 2 号功能取得当前时间，
                                    返回时，在 CH、CL 和 DH 寄存器中含有时、分和秒的 BCD 码。
    INT     1AH
    MOV     [hour], CH
    MOV     [minute], CL
    MOV     [second], DH
;设置光标（显示时间的位置）
    MOV     BH, 0
    MOV     AH, 3                   ;取得当前光标位置
    INT     10H
    PUSH    DX                      ;保存当前光标位置
    ;
    MOV     DX, (ROW<<8) + COLUMN
    MOV     AH, 2
    INT     10H                     ;设置显示时间的开始位置
    ;
;显示当前时间（时:分:秒）
        MOV     AL,[hour]
        CALL    EchoBCD
        MOV     AL,':'
        CALL    PutChar
        MOV     AL,[minute]
        CALL    EchoBCD
        MOV     AL,':'
        CALL    PutChar
        MOV     AL,[second]
        CALL    EchoBCD
  ;恢复光标原先位置
        POP     DX                  恢复原光标位置
        MOV     AH, 2
        INT     10H                 ;重新位置位置
        ;
        POPA                        ;恢复通用寄存器
        POP     DS                  ;恢复 DS
        RET
EchoBCD:            子程序显示 2 位 BCD 码值
        PUSH    AX
        SHR     AL, 4
        ADD     AL, '0'
        CALL    PutChar
        POP     AX
        AND     AL, 0FH
        ADD     AL, '0'
        CALL    PutChar
        RET
  PutChar:
```

```
        MOV     BH, 0
        MOV     AH, 14
        INT     10H
        RET
    ;
second   DB     0               ;秒数保存单元
minute   DB   0               ;分数保存单元
hour     DB   0               ;时数保存单元
;
count    DB    1                ;计数器
old1ch   DD    0                ;用于保存原 1CH 号中断向量
;
```