第7章 I/O程序设计

7.1 I/O接口

7.2 I/O操作

7.3 文件I/O

- ✓输入/输出（I/O）是CPU对外设进行控制和数据交换的过程。
- ✓微型计算机由主机和外设两部分组成，主机包括微处理器（CPU）、存储器和I/O接口。
- ✓外部设备又被称为I/O设备，简称外设。I/O设备与I/O接口相连，是微机的重要组成部分。与存储器相比，外设有其自身的特点。
- ✓存储器功能比较单一，体现在CPU对指定地址的存储单元进行读写操作上，而外设的功能多种多样，每一种外设的控制方式和数据交换都各不相同。
- ✓在操作系统中，需要为不同的外部设备配置不同的设备驱动程序。

7.1 I/O接口

- ✓CPU与内存可以直接进行数据交换，但CPU与外设之间实现信息交换，必须通过接口电路中的I/O端口来进行。
- ✓每一个外设都有其特定的I/O端口。CPU通过I/O指令对端口进行读、写操作。

7.1.1 接口、端口、端口地址

- ✓接口电路是主机与外设的通信桥梁，接口电路有时简称为接口。接口是用于实现主机与外设传输信息的通道，传输信息的种类有数据信息、控制信息和状态信息。由于外设的种类繁多、速度相差很大，接口电路中必须设置传输信息所需要的缓冲寄存器。这些缓冲寄存器依据所传输信息的种类被分为数据缓冲寄存器、控制缓冲寄存器和状态缓冲寄存器。
- ✓通常把I/O接口电路中能被CPU直接访问的寄存器取名为端口，端口分为三类：数据端口、控制端口和状态端口。数据端口有两种，即输出数据端口和输入数据端口。数据端口用于实现CPU与外设之间的数据交换，所以有两种方向。控制端口是用于实现CPU对外设的控制，所以它只能是输出。状态端口是用于反馈外设的状态信息给CPU的，所以它只能是输入。

- ✓和存储器一样，每一个端口都有一个地址，叫做端口地址，又称为端口号。端口地址是端口位置的二进制编码，CPU通过这些端口编码向外部设备发送命令、读取状态、发送数据、接收数据等。
- ✓每一个端口的宽度为8位，可容纳一个字节，端口地址是按字节编排的。
- ✓CPU访问I/O端口有三种方法：

（1）用I/O指令实现信息传输；

（2）通过BIOS中断调用实现信息传输；

（3）通过DOS中断调用实现信息传输。

- ✓BIOS中断调用是更低级的中断调用，它是与硬件相关的，兼容性较差，但速度较快。DOS中断调用是较高级的中断调用，它与硬件无直接关系，兼容性较好，但速度较慢，大多数情况下其内部会再调用BIOS中断调用。

接口、端口、外设在微机中的连接如图7.1所示。

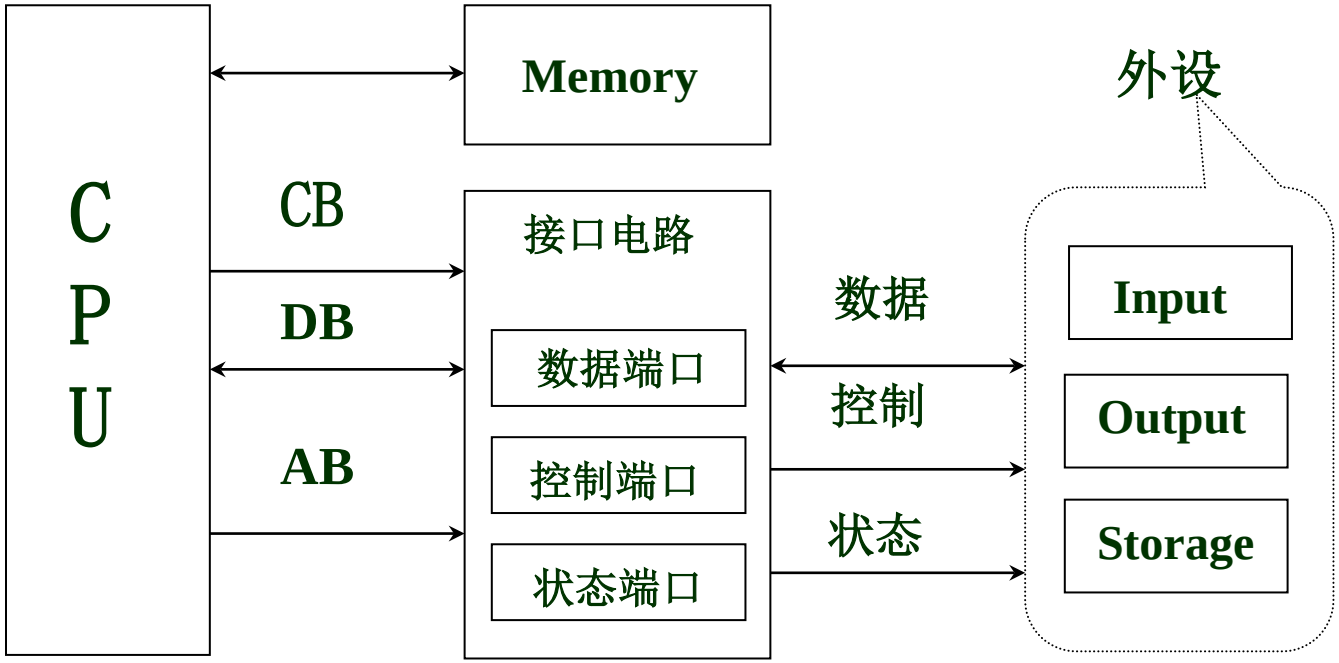

图7.1 主机、外设、接口电路连接示意图

7.1.2 I/O接口的硬件分类

按照电路和设备的复杂程度，I/O接口的硬件主要分成两大类。

1．I/O接口芯片

主板上的I/O接口芯片，大多是可编程的大规模集成电路，能够完成相应的接口操作，如定时器、计数器、中断控制器、DMA控制器、并行接口等。在PC系列微机中，这些接口芯片是由8259，8237A等芯片组成。随着集成电路技术的发展，目前PC机系统主板上的所有接口功能都已集成在一片或几片大规模集成电路芯片中，称为芯片组。

2．I/O接口卡

这些接口控制卡是由若干个集成电路按一定的逻辑结构组装而成的接口部件，它们或直接集成在电脑主板上，或制成插件安装在系统总线扩展槽上。按照所连接的外部设备控制的难易程度，该接口卡的核心部件或为一般的接口芯片或为微处理器。凡安装微处理器的接口卡通常称为智能接口卡，这种卡上具有一片EPROM芯片，芯片内固化了控制程序，如硬盘驱动器的接口控制卡。

7.1.3 I/O端口的地址分配

一个接口中可能有多个端口，一个计算机系统内又有很多个接口，所以通过端口传输信息首先需要区分端口，区分端口可通过端口地址不同来区分。既然I/O接口分成I/O接口芯片和I/O接口控制卡两大类，那么系统对I/O空间的布局也分成两部分，总共1024个端口。前256个端口（000－0FFH）专供I/O接口芯片使用，后756个端口（100－3FFH）专为I/O接口卡使用。

对于一个具体的计算机而言，I/O端口地址实际上只用了其中很小的一部分，因为一个系统中只有十几个外部设备和大容量存储设备与主机相连。对不同型号的计算机及其接口，I/O端口的编号有时也不完全相同。

表 7.1 I/O端口地址分配表

I/O地址	功能	I/O地址	功能
00-0F	DMA控制器	2F8-2FE	2号串行口（ＣＯＭ２）
２０－３Ｆ	可编程中断控制器８２５９Ａ	３２０－３２４	硬盘适配器
４０－５Ｆ	可编程中断计时器	３６６－３６Ｆ	ＰＣ网络
６０－６３	８２５５Ａ　ＰＰＩ	３７２－３７７	软盘适配器
７０－７１	ＣＭＯＳ　ＲＡＭ	３７８－３７Ａ	２号并行口（ＬＰＴ１打印机）
８１－８Ｆ	ＤＭＡ页表地址寄存器	３８０－３８Ｆ	ＳＤＬＣ及ＢＢＣ通信
９３－９Ｆ	ＤＭＡ控制器	３９０－３９３	Ｃｌｕｓｔｅｒ适配器
Ａ０－Ａ１	可编程中断控制器	３Ａ０－３ＡＦ	ＢＢＣ通信
Ｃ０－ＣＥ	ＤＭＡ通道、内存/传输地址寄存器	3B0-3BF	MDA视频寄存器
F0-FF	协处理器	3BC-3BE	1号并行口
170-1F7	硬盘控制器	3C0-3CF	EGA/VGA视频寄存器
200-20F	游戏控制端口	3D0-3D7	CGA视频寄存器
278-27A	3号并行口（LPT2打印机）	3F0-3F7	软盘控制寄存器
2E0-2E3	EGA/VGA使用	3F8-3FE	1号串行口（COM）

7.1.4 I/O端口的寻址方式

一个I/O接口电路总要包括若干个端口，除常见的数据端口、命令端口和状态端口外，还包括方式控制、操作结果和地址索引等用作特殊用途的端口。端口可被处理器如同存储单元一样访问，因此每个端口就存在着寻址方式问题。

在现代微机系统中，对I/O接口的端口编址有两种方式：统一编址和独立编址。

1．统一编址

统一编址又称为“存储器映像编址”，是把每一个端口视为一个存储单元，并赋予相应的存储器地址。微处理器访问该端口，如同访问存储器一样，只是单元地址值不同而已。所有访问内存的指令同样适合于访问I/O端口。

优点：无需专门的I/O指令，简化了指令系统，可通过功能较强的访问内存指令直接对I/O数据进行算术或逻辑运算。

缺点：I/O数据传输时间延长。

2. 独立编址

独立编址又称“覆盖编址方式”，是把所有I/O端口看做一个独立于存储器空间的I/O空间。在这个空间内，每个端口都被分配1个地址与之对应。微处理器对I/O端口和存储单元的不同寻址是通过不同的读写控制信号来实现的。由于系统需要的I/O端口寄存器一般比存储器单元要少得多，一般只设置256～1024个端口，因此选择I/O端口只需用8～10根地址线即可。例如：Intel公司生产的微处理器，像8086/80X86等系列，

显然，要访问独立于存储空间的端口，必须用专门的I/O指令。为加快I/O数据的传输速度，设计的这种I/O指令均为单字节或多字节（指令直接带端口地址）。通常这种I/O指令有2种，即输入指令IN、输出指令OUT及其相关的指令组。不过这种I/O指令仅做数据传输而无算术或逻辑运算功能。

3．独立编址优缺点：

优点：I/O端口地址不占用存储器地址空间。I/O端口地址译码器较简单，寻址速度较快。使用专用I/O指令和真正的存储器访问指令有明显区别，可使程序编制得更清晰，便于理解和检查。

缺点：专用I/O指令类型少，程序设计灵活性差。使用I/O指令只能在累加器（AL、AX、EAX）和I/O端口间交换信息，处理能力不如存储器映像方式强。同时要求处理器能提供存储器读/写、I/O端口读/写两组控制信号，增加了控制逻辑的复杂性。

7.2 I/O操作

在统一编址方式中，CPU不需要使用专门的I/O指令来访问I/O端口，可以将I/O端口看作是内存单元。在独立编址方式中，CPU必须使用专门的I/O指令来访问端口。对一个地址而言，它要么是内存单元，要么是I/O端口。内存操作指令中使用的地址是内存地址，而I/O指令使用的地址是端口地址。

7.2.1 I/O指令

✓I/O指令共有四条，所有I/O端口与CPU之间的通信都是由这四条指令完成。

1．IN指令

格式：IN {AL|AX|EAX}，{端口地址|DX}

功能：将I/O端口中的信息读入到累加器中。

✓说明：累加器为AL时，从指定端口中读出1个字节送给AL；累加器为AX时，从指定端口中读出1个字节送给AL，从下一端口地址中读出另1个字节送给AH；累加器为EAX时，连续读出从指定地址开始的4个端口中的内容送给EAX。

✓ 当端口号小于256时，在IN指令中可以直接给出端口号。当端口号大于等于256时，必须先把端口号传送到DX寄存器中，然后执行IN指令。

例7.1：IN指令使用举例

⑴从端口地址为61H的端口上读取一个字节：

```
IN  AL，61H
```

⑵从端口地址为379H的端口上读取一个字节：

```
MOV  DX，  379H
IN  AL，DX
```

⑶读取60H，61H两个端口：

```
IN  AX，60H
```

⑷从端口地址为60H、61H、62H、63H的端口上读取一个双字：

```
IN  EAX，60H
```

⑸从端口地址为0E000H，0E001H的端口上读取一个字：

```
MOV  DX，0E000H
IN  AX，DX
```

2．OUT指令

格式：OUT{端口地址|DX}，{AL|AX|EAX}

功能：将累加器中的信息输出到I/O端口。

✓说明：累加器为AL时，AL中的内容输出到指定端口中；累加器为AX时，AL中的内容输出到指定端口中，AH中的内容输出到下一端口中；累加器为EAX时，EAX中的四字节内容依次输出到指定地址开始的4个端口中。

✓ 当端口号大于等于256时，必须先把端口号放到DX寄存器中。当端口号小于256时，在OUT指令中可以直接给出端口号。

例7.2：OUT指令使用举例

(1)将AL中的内容输出到61H端口：

```
OUT  61H，AL
```

(2)将AL中的内容输出到37AH端口：

```
MOV  DX，  37AH
OUT  DX，  AL
```

(3)将AL和AH中的内容输出到3CEH，3CFH端口：

```
MOV  DX，  3CEH
OUT  DX，  AX
```

3．INS指令

格式：INSB/INSW/INSD

功能；将I/O端口中的信息读入到ES：[EDI]指向的内存单元中；

（1）使用INSB时，从端口号为DX的端口中读出1个字节送给ES：[EDI]单元；

（2）使用INSW时，从端口号为DX和DX+1的端口中分别读出2个字节送给ES：[EDI]，ES：[EDI+1]单元；

（3）使用INSD时，从端口号为DX，DX+1，DX+2，DX+3的端口中分别读出4个字节送给ES：[EDI]，ES：[EDI+1]，ES：[EDI+2]，ES：[EDI+3]单元；

INS指令在读出端口中的信息并保存到内存单元后，自动调整EDI。DF=0时，EDI分别增加1，2，4；DF=1时，EDI分别减去1，2，4；

结合REP前缀，INS指令可以对I/O端口连续读取ECX次。

例7.3：从170H端口连续读取512次，将读取到的信息存放在Buffer数组中：

```
Buffer  DB  512  DUP(?)
MOV  DX, 170H
MOV  ECX, 512
LEA  EDI, Buffer
CLD
REP  INSB
```

4．OUTS指令

格式：OUTSB/OUTSW/OUTSD

功能；将DS：[ESI]指向的内存单元中的信息输出到I/O端口中；

（1）使用OUTSB时，DS：[ESI]内存单元中的信息输出到端口号为DX的端口中；

（2）使用OUTSW时，DS：[ESI]，DS：[ESI+1]内存单元中的信息分别输出到端口号为DX和DX+1的端口中；

（3）使用OUTSD时，DS：[ESI]，DS：[ESI+1]，DS：[ESI+2]，DS：[ESI+3]内存单元中的信息分别输出到端口号为DX，DX+1，DX+2，DX+3的端口中；

OUTS指令在内存单元中的信息输出到I/O端口后，自动调整ESI。DF=0时，ESI分别增加1，2，4；DF=1时，ESI分别减去1，2，4；

结合REP前缀，OUTS指令可以对I/O端口连续输出ECX次。

例7.4：连续输出512次，将Buffer中的信息不断输出到170H端口。

```
Buffer  DB  512  DUP(?)
MOV  DX, 170H
MOV  ECX, 512
LEA  ESI, Buffer
CLD
REP  OUTSB
```

7.2.2 I/O控制方式

由于各种外设的工作速度不同，即使是高速的I/O外设，其速度与CPU相比也相差很大，因此，它们之间的数据传输必须通过接口中的数据缓冲器来进行。

CPU对I/O设备的控制有三种方式，即：程序控制方式、中断方式、DMA方式。

程序控制方式：指CPU与外设在程序控制下进行数据传输，主要使用I/O指令进行端口编程来实现。该方式又可以分为无条件传送方式和查询方式。

中断传送方式：由外部设备产生中断信号，中断信号传递给CPU后，由CPU执行中断处理程序，最后由中断处理程序来完成数据的传递，实现的方法是调用系统中断服务。

DMA传送方式：即直接存储器存取方式，它可以满足高速I/O设备与RAM进行批量数据传输的需要。特点是通过一个专用的DMA控制器直接控制I/O设备与RAM之间的数据传输，而无需CPU介入。由此可见，DMA控制方式与前两种方式的本质区别是用硬件代替软件实现数据传输。

DMA控制器（Intel 8237A）一般包括四个寄存器：控制寄存器、状态寄存器、地址寄存器和字节计数器

系统完成DMA传送的步骤如下：

（1）DMA控制器向CPU发出HOLD信号，请求使用总线。

（2）CPU发出响应信号HOLD给DMA控制器，并将总线让出，这时CPU放弃了对总线的控制，而DMA控制器获得了总线控制权。

（3）传输数据的存储器地址（地址寄存器内容）通过地址总线发出。

（4）传输数据的字节数通过数据总线进行传送。

（5）地址寄存器增1，以指向下一个要传送的字节。

（6）字节计数器减1。

（7）如字节计数器非0，转向第3步。

（8）否则，DMA控制器撤销总线请求信号HOLD，传送结束

7.2.3 I/O端口编程

1．CMOS数据的读取

现代微机中都包含一个时钟电路，定时更新时间与日期。时钟电路自带电池，因此无论微机是否开机，时钟都会不断更新。因此，这个硬件时钟也叫实时钟（简称RTC）。实时钟信息（年/月/日/时/分/秒）保存在CMOS RAM中，在系统关机后，时钟电路自带的后备电池继续给RAM供电。由于CMOS芯片的功耗很低，后备电池能用几年甚至更长。

CMOS RAM容量至少为64字节，一般为256字节。实时钟信息使用了最前面的14个字节，其它字节用于保存微机的配置信息，例如系统内存容量、软盘、硬盘类型等。开机时进入BIOS设置后，对系统进行的一些设置就保存在CMOS RAM中，实时钟信息的格式如表7－2所示。

表7－2 RTC信息格式简表

偏移量	内容	取值范围	格式
0	秒	00H～59H	BCD码，10H＝10秒，59H＝59秒
2	分	00H～59H	BCD码，05H＝06分，59H＝59分
4	小时	00H～23H	BCD码，12H＝10点，20H＝20点
6	星期	1～7	1＝星期日，2＝星期一，7＝星期日
7	日	01H～31H	BCD码，30H＝30日
8	月	01H～12H	BCD码，12H＝12月
9	年	00H～99H	BCD码，05H＝2005年

CMOS RAM是一种存储器，但对它的访问必须通过I/O接口电路用I/O指令来完成。接口电路设置了两个寄存器，其I/O地址分别是70H和71H。70H是RTC的地址寄存器，71H是RTC的数据寄存器。CPU要访问某个CMOS RAM单元时，先用OUT指令将该单元的地址（00H～7FH）写到70H端口中，再使用IN指令读入该CMOS RAM单元的内容，具体操作过程如图7.2所示，具体代码编写如例程7.5。

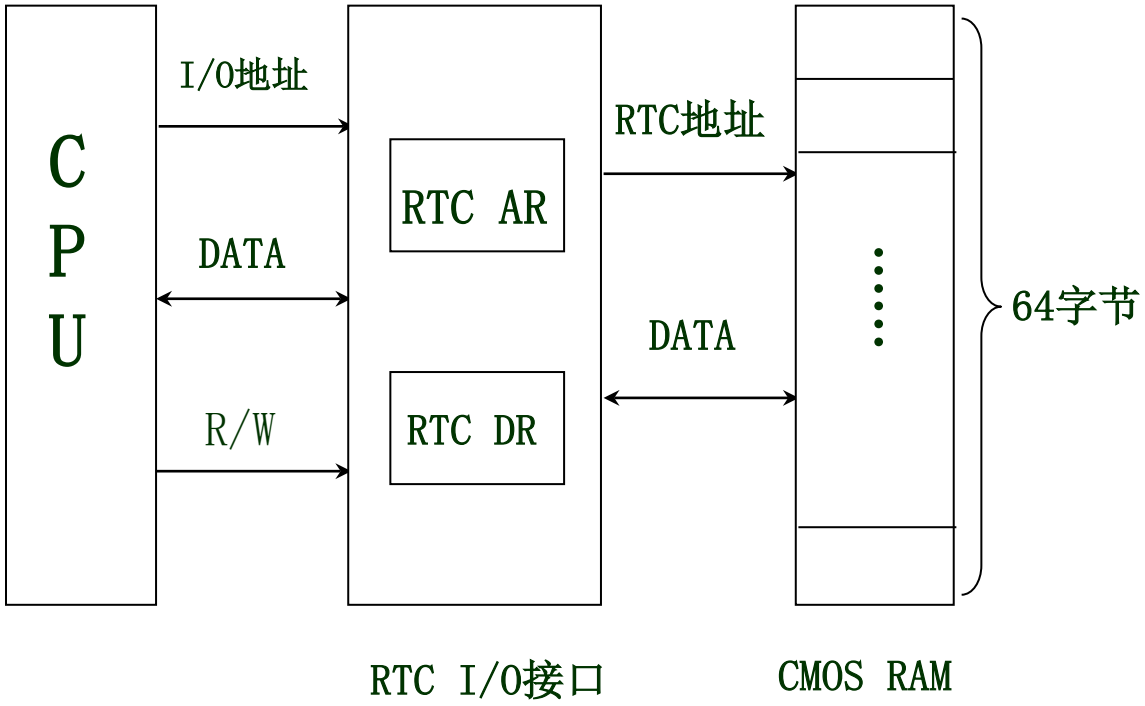

图7.2　RTC输入/输出接口图

例7.5 将CMOS数据参数读入到缓冲区BUF中。

```
DATA SEGMENT
 BUF DB 64 DUP(0)
DATA ENDS
CODE SEGMENT
ASSUME
   CS:CODE,DS:DATA
START:MOV AX,DATA
      MOV DS,AX
      MOV BX,OFFSET
    BUF
      MOV DX,0
      MOV CX,64
NEXT: MOV AL,DL
      OUT 70H,AL
      IN  AL,71H
      MOV [BX],AL
       INC BX
       INC DX
       LOOP NEXT
       MOV AH,4CH
       INT 21H
CODE   ENDS
       END  START
```

2．CMOS口令的破解

CMOS RAM中2EH～2FH字单元中，存放CMOS中的10H～2DH之间的各个字节相加后得到的和。2EH单元是和的高字节，2FH单元是和的低字节。若修改了CMOS中的10H～2DH之间的某一存储单元的内容，而DS：2EH～2FH中的校验和未作改变，则在系统重新启动微机时，系统将会出现类似“CMOS校验和有错”的提示信息，并重新配置系统。

利用这一特性，可去除CMOS SETUP的口令。即在启动微机系统后，向CMOS的某些单元写入与原来不同的配置数据来修改配置，但未修改CMOS的校验和，如可向CMOS RAM中的偏移地址为10H处，写入00，即执行如下步骤：

C:\>DEBUG

－O 70 10

－O 70 00

－O

再次重新启动微机系统，因CMOS中某一数值（10H处）已经改变，此时系统提示“CMOS校验和失败，按<DEL>运行SETUP程序，按<F1>继续”。按<DEL>进入CMOS参数设置后存盘，就可以消除CMOS SETUP程序的密码且可重新设置新密码。

3．扬声器发声程序

为了具有音响输出能力，PC系统板上装有一个微型扬声器以及控制电路和驱动电路。控制电路能以位触发和定时器控制两种方式驱动扬声器发声。位触发方式是通过输出信号直接控制I/O端口号61H的第1位，使该位按所需频率进行0和1交换，从而使扬声器发出声音。定时器控制是通过可编程定时/计数器8254来实现。

8254的端口地址为40H、41H、42H、43H，共有3个独立的计数器，编号为0、1、2。计数器2的端口地址为42H，CPU通过对计数器2进行编程，使计数器2 的输出OUT2与端口61H的第1位（D1）相“与”后输出到扬声器发出声音。扬声器控制电路如图7.3所示

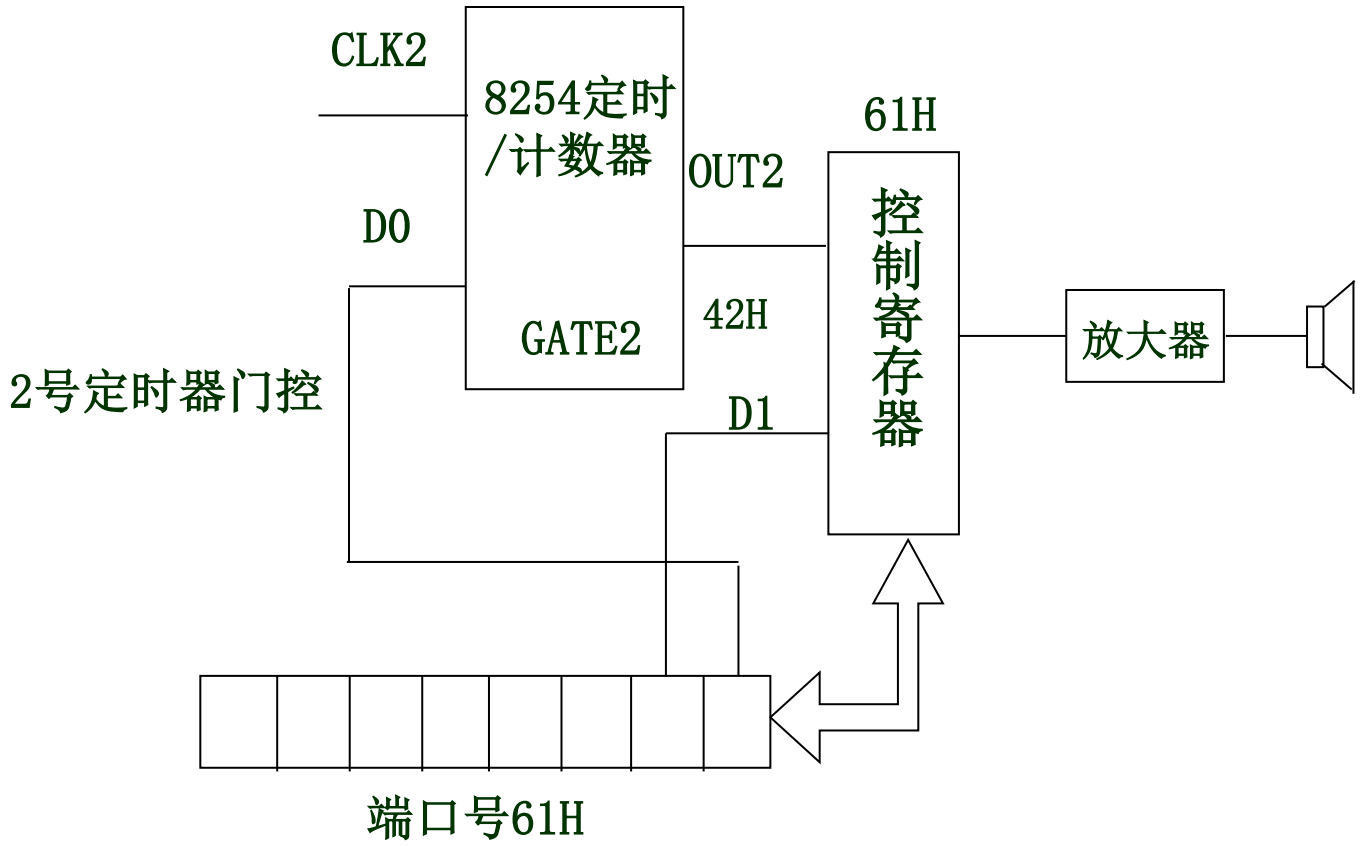

图7.3　8254扬声器控制电路图

CPU通过对定时器的计数器2（端口地址为42H）进行编程，使其I/O寄存器接收一个控制声音频率的16位计数值，端口61H的最低位控制计数器2门控的开关，以产生特殊的音响效果。当定时器接收的计数值为533H时，能产生896Hz的声音，因此产生其它频率（FREQ）的计数值就可由下式计算出来：

533H×896÷FREQ =1234DCH÷FREQ

在送出频率计数值之前，还要给方式寄存器（其端口地址为43H）送一个方式值，也称为幻数，这个幻数决定对哪一个计数器通道编程，采用什么模式，送入通道的计数值是字节还是字，是二进制还是BCD码。具体应用详见例程7.6

例7.6 通过PC机扬声器播放“太湖船”音乐程序。

```
SHOW    MACRO B
        LEA DX,B
        MOV AH,9
        INT 21H
        ENDM
DATA    SEGMENT PARA 'DATA'
INFO2   DB 0DH,0AH,'THIS IS A MUSIC PROGRAM!$'
MUS_F   DW 330,392,330,294,330,392,330,294,330
        DW 330,392,330,294,262,294,330,392,294
        DW 262,262,220,196,196,220,262,294,332,262,0
MUS_T   DW 3 DUP(50),25,25,50,25,25,100
        DW 2 DUP(50,50,25,25),100
        DW 3 DUP(50,25,25),100
DATA    ENDS
CODE    SEGMENT
        ASSUME DS:DATA,CS:CODEMAIN    PROC FAR
        MOV AX,DATA
        MOV DS,AX
        SHOW INFO2
        LEA SI,MUS_F
        LEA BP,MUS_T
```

```
CALL MUSIC2
        MOV AH,4CH
        INT 21H
MAIN    ENDP
MUSIC2  PROC NEAR
      PUSH AX
      PUSH BX
      PUSH CX
      PUSH DX
      PUSH DI
 FREQ: MOV DI,[SI]
      CMP DI,0
      JE END_F
      MOV AX,100
      MOV BX,DS:[BP]
      MUL BX
MOV BX,AX
      MOV AL,0B6H
      OUT 43H,AL
      MOV DX,12H
      MOV AX,533H*896
      DIV DI
      OUT 42H,AL
    MOV AL,AH
      OUT 42H,AL
IN AL,61H
      MOV AH,AL
      OR AL,3
      OUT 61H,AL时间延迟
NEXT:  MOV CX,2801*1000
DELAY: LOOP DELAY
      DEC BX
      JNZ NEXT
      MOV AL,AH
      OUT 61H,AL
      ADD SI,2
      ADD BP,2
      JMP FREQ
END_F: MOV AL,48H
      OUT 61H,AL
      POP DI
      POP DX
      POP CX
      POP BX
      POP AX
      RET
MUSIC2   ENDP
CODE   ENDS
      END MAIN
```

程序中时间延迟代码是通过LOOP循环实现的，这个循环受CPU运行速度的影响，速度快的CPU执行这个循环所花的时间较短。一般情况下，执行2801次LOOP指令约需10MS的时间，因此用10MS的倍数值来控制扬声器开关的时间间隔，就可控制音长。

以上程序可以控制扬声器发出一定频率和音长（持续时间）的声音。利用这个原理，可以编写演奏乐曲的程序。当然和音箱相比，扬声器的音色要逊色的多，所以不能通过这种方式得到和声霸卡驱动的音箱相媲美的音响效果。

7.3 文件I/O

在程序运行时，程序本身和数据都存放在内存中。当程序运行结束后，这些内存被释放，保存在内存中数据随之丢失，不能继续使用。内存是一种“临时性”的存储媒介。

如果需要永久地保存程序运行所需的原始数据，或程序运行产生的结果，就必须以文件的形式存储到外部存储介质上。程序运行时，可以从文件中读取所需的的原始数据，也可以将运行结果保存在文件中。文件一般都保存在磁盘上，例如硬盘、光盘、USB盘等。和内存相比，这些存储介质是“永久性”的，在计算机关机的情况下仍然能够保存其内容。

7.3.1文件

1．文件

文件是指存放在外部存储设备上的数据集合，是一种数据的组织形式。为标识一个文件，每个文件都必须有一个文件名，格式如下：

文件名[.扩展名]

扩展名也叫后缀。一般可根据后缀决定文件的类型，例如：file.doc是一个Word文档文件，file.ppt是一个PowerPoint文件，file.txt是一个文本文件，file.asm是一个汇编源程序文件。

不同操作系统和文件系统约定了该系统下的文件命名规则，例如，FAT文件系统不区分大小写，文件名中不能使用空格等，而NTFS文件系统中就不存在这些限制。

2. 文件分类

为了有效、方便地组织和管理文件，经常从不同的角度对文件进行分类，这里介绍几种常见的分类方法：

(1)根据文件的内容，可分为程序文件和数据文件，程序文件又可分为源文件、目标文件和可执行文件。

(2)根据文件的组织形式，可分为顺序存取文件和随机存取文件。

(3)根据文件的存储格式，可分为ASCII码文件（又称为文本文件）和二进制文件。

3．对文件的基本操作

基本操作包括打开文件、读文件、写文件、关闭文件。

(1)在读写一个文件之前，需要先打开文件。

(2)读文件是将文件中的数据传送到内存的操作。

(3)写文件是从内存向文件中传送数据的操作。

(4)不再需要对文件读写时，关闭文件。

对文件的读或写操作的最小单位是1字节，一次读或写操作可以传送多个字节。

7.3.2文件缓冲系统

文件缓冲系统是指系统自动地在内存中为每个正在使用的文件开辟一个缓冲区。

从内存向文件写入数据时，首先将数据复制到缓冲区中。待缓冲区填满后，再将缓冲区的内容输出到文件中。

例如，程序每次向文件写入2K字节，缓冲区的大小是8K字节。写入4次后，缓冲区被填满（4×2K＝8K），系统将缓冲区的内容一次性输出到文件。前面3次写入操作仅仅将数据写入到缓冲区中，最后一次写入操作在填满缓冲区后，将整个8K字节数据写入到文件中。程序的4次写文件操作被系统合并为一次。具体写入过程如图7.4所示。

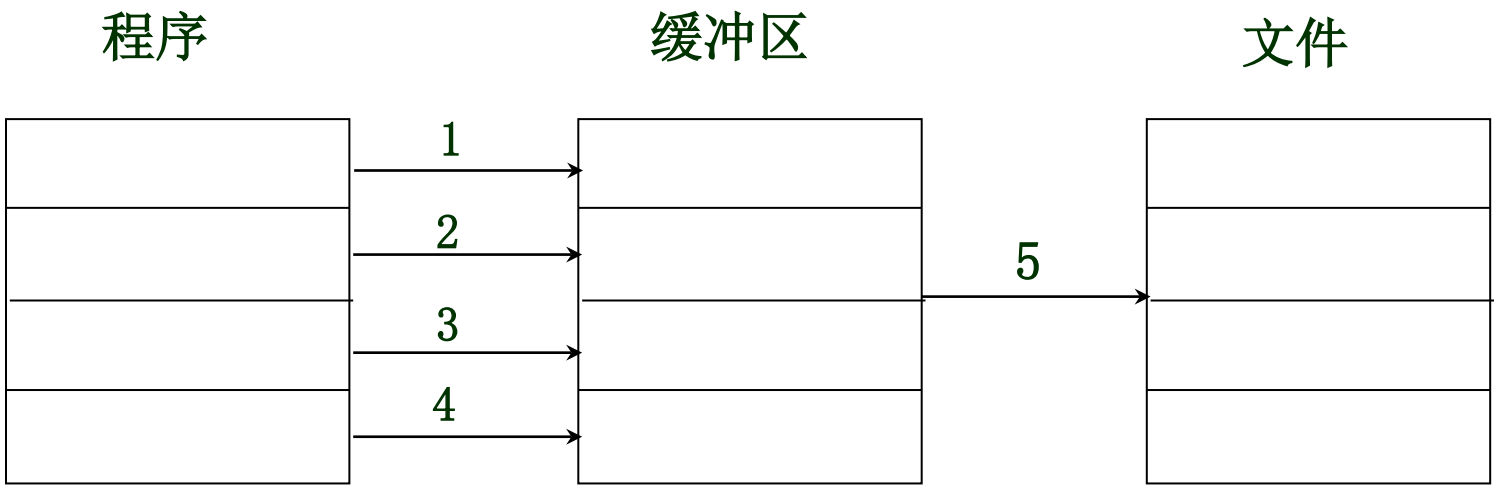

图7.4 写文件操作过程图

从文件向内存读入数据时，系统首先将大量数据读入到缓冲区中，再从缓冲区中将数据送到程序数据区。

例如，程序每次从文件中读取2K字节，缓冲区的大小是8K字节。第1次读取时，系统从文件中读取8K字节，将最前面的2K字节传送给程序。程序接下来的3次读取操作，系统直接将缓冲区中的数据传送给程序，而不必从文件中读取。程序的4次读文件操作被系统合并为一次。具体读取过程如图7.5所示。

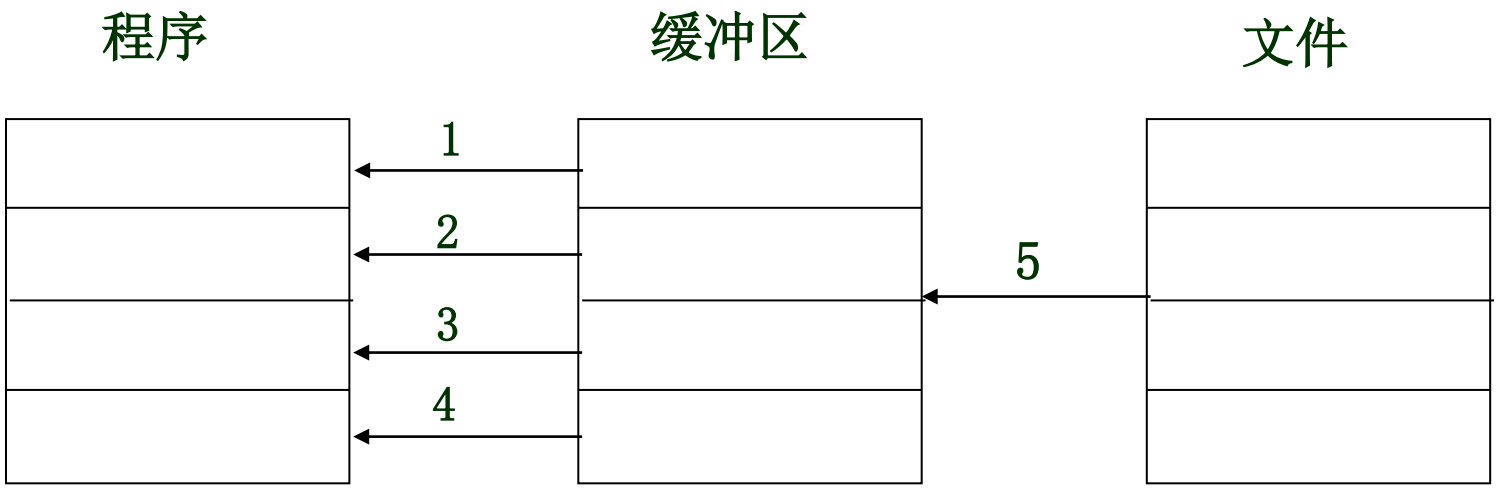

图7.4 读文件操作过程图

在关闭文件时，如果系统缓冲区还有未被写入到文件的数据，则系统将这些数据写入到文件中。系统缓冲区的使用对程序是透明的，程序不需要关心缓冲区的大小和使用状况。设置系统缓冲区主要是为了减少对磁盘的访问次数。如果没有系统缓冲区，每一次读写操作都需要访问磁盘。而磁盘的访问时间很慢，至少要几毫秒，这样，程序的运行速度就会大大地降低。

7.3.3文件与目录管理

- ✓应用程序对磁盘文件和目录的操作是通过DOS功能调用实现的，调用时必须按各功能的要求设置好入口参数，再按DOS指定的形式进行调用。DOS提供的磁盘管理功能包括三个方面，一是关于文件内部所存放的数据，如创建、读、写等，二是针对文件的外部属性，如查找文件的名称、日期，再就是目录管理的创建、删除功能等。
- ✓DOS在21H号中断服务程序中，不仅能够实现文件操作的创建、打开、读写、关闭等基本功能，为了支持随即文件，还为每个处于打开状态的文件准备了一个双字型的文件指针，中断服务子程序也包括对文件指针的处理。除这些功能外，DOS的21H号中断服务也可实现设置文件属性、文件查找、文件更名，以及目录的创建、删除、修改等功能。

7.3.4文件操作实例

1．建立和打开文件

1）建立文件（3CH）

建立一个文件，使用INT 21H，AH=3CH子功能调用。

使用该子功能调用建立一个文件后，文件便被打开，并返回1个16位的代码。该代码作为文件标识符的句柄，简称文件句柄或文件代号。以后对该文件进行操作，便用该文件的句柄进行，不再使用文件标识符了，这就是文件句柄方式存取的含义。

入口参数：AH=3CH，CX=文件属性；CX=00H 可读写；CX=01H 只读；CX=02H 隐藏；CX=03H 卷标号；CX=04H 系统；DS：DX=文件名字符串首地址。

出口参数：进位标志（CF）=0，成功，此时AX=文件代
（CF）=1，错误，此时AX中为错误代码。

当（CF）=1时，AX的内容不是文件代号，而是错误原因代码；AX＝3，表示路径未找到；AX＝4，表示没有代号可用；AX＝5，表示拒绝存取（属性错或文件代号错）等。

例7.7 建立一个路径名为C:\MYFILE.DAT的文件。

```
MOV   AH, 3CH          ；功能号
XOR   CX, CX           ；定义文件属性CX=0可读写
MOV   DX, SEG  NAME    ；文件名的段地址
MOV   DS, DX
MOV   DX, OFFSET NAME  ；文件名偏移地址
INT   21H              ；建立文件功能调用
JC    ERROR            ；若出错转ERROR
MOV   HANDLE，AX       ；正确则取文件号存入HANDLE
   中
ERROR：
NAME  DB  “C:\MYFILE.DAT”,0；文件名ASCII字符串
HANDLE  DW  ?          ；定义HANDLE为1个字
```

2）打开文件（3DH）

当一个文件已存在时，对文件存取操作前，首先要打开文件。

打开文件使用INT 21H，AH=3DH子功能调用。

入口参数：AH=3DH，AL=存取方式；

AL=00 读文件；

A1=01 写文件；

A1=02 读写文件；

DS：DX=文件名字符串首地址。

出口参数：（CF）=0，成功，此时（AX）=文件句柄；

（CF）=1，错误，此时AX中为错误代码。

例7.8 打开一个文件，路径名为C:\MYFILE.DAT

```
MOV   AH, 3DH
MOV   AL, 2            ；文件为读写属性
MOV   DX, SEG NAME
MOV   DS, DX           ；文件名SEG送DS
MOV   DX, OFFSET NAME  ；文件名EA送DX
INT   21H              ；功能调用
JC    ERROR            ；出错转ERROR
MOV   HANDLE, AX       ；正确，取文件号
ERROR：
NAME： DB “C\MYFILE.DAT”,0  ；文件名字符串
HANDLE DW ？                ；文件句柄
```

当文件建立或打开后，便可进行存取操作了。

2. 文件读写及关闭

1）将数据写入文件（40H）

将数据写入磁盘文件时，数据应首先存到一个数据缓冲区中。

数据写入文件或设备，使用INT 21H, AH=40H子功能调用。

入口参数：AH=40H；

BX=文件句柄；

CX=要写的字节数；

DS：DX=文件名字符串首地址

出口参数：（CF）=0成功，AX中为写入的字节数；

（CF）=1出错，AX中为错误代码。

例7.9 将缓冲区DATABUF中的1024B内容写入文件，文件代号存放在FHANDLE中。

```
MOV   AH，40H          ；功能号
MOV   BX，FHANDLE      ；打开的文件句柄
MOV   CX，1024         ；待写入的字节数
MOV   DX，SEG DATABUF  ；数据缓冲区段地址送DS
MOV   DS，DX
MOV   DX，OFFSET DATABUF ；数据缓冲区偏移地址送DX
INT   21H              ；执行写入功能
JC    ERROR            ；出错，转ERROR
CMP   AX，1024         ；全部字节是否写完？
JNZ   DISKFULL         ；没有，盘满，转
DISKFULL：
┆
ERROR：
┆
DATABUF  DB  1024 DUP （?）；数据缓冲区长度1024B
FHANDLE  DW  0          ；文件句柄变量
```

2）从文件中读数据（3FH）

当一个文件打开后，便可从中读出数据，送到内存缓冲区。

读文件，可使用INT 21H，AH=3FH子功能调用。

入口参数：AH=3FH；

BX=文件句柄；

CX=要读的字节数；

DS：DX=存放文件数据的缓冲区地址。

出口参数：（CF）=0，读成功，AX中为读出的字节数。如AX=0，则表示是文件尾。

（CF）=1，读出错，AX中为错误代码。

例7.10 从文件中读出1024B存放到缓冲区DATABUF中，打开的文件句柄已放入FHANDLE中。

```
MOV   AH，3FH          ；功能号
MOV   BX，FHANDLE      ；打开文件句柄送BX
MOV   CX，1024         ；读入的字节数
MOV   DX，SEG DATABUFF ；读缓冲区的地址
MOV   DS，DX
MOV   DX，OFFSET DATABUFF
INT   21H              ；调用读功能
JC    ERROR            ；出错转ERROR
┆
ERROR：
DATABUFF DB  1024 DUP （？）；缓冲区
FANDLE   DW  0          ；文件句柄
```

3）关闭文件（3EH）

当对建立或打开的文件操作结束后，就要及时关闭文件。关闭文件，可使用INT 21H，AH=3EH子功能调用。

入口参数：AH=3EH；

BX=文件句柄；

出口参数：（CF）=0成功；（CF）=1出错，AX中为出错代码。

例7.11 关闭一个文件，文件句柄在FHANDLE变量中。

```
MOV  AH，3EH        ；功能号
MOV  BX，FHANDLE    ；文件句柄送BX
INT  21H            ；功能调用
JC   ERROR          ；出错转ERROR
┆
ERROR：
FHANDLE  DW  0      ；文件句柄
```

3．移动文件指针

利用文件代号（句柄）存取文件是以字节为存取单位的，一个文件被看作由许多字节组成，每次读写的字节数可任意指定，但一般还是为输入输出缓冲区的大小所限制。所以一个比较大的文件总是要分几次读写，每次读写的字节称为记录。

在读写文件时，每次读写的记录是如何“拼接”起来的呢？原来是操作系统为文件保存了一个称为读写指针（R/ W）的变量，由它指示应从文件的什么地方读出，或应往文件的什么地方写入。

为了存取文件中间某一特定的记录，首先要使文件读写指针指向这个记录。

移动文件指针，可使用INT 21H，AH=42H子功能调用。

入口参数：AH=42H，BX=文件句柄；

AL＝方式码，CX:DX双字长的偏移值,可正可负。

出口参数：（CF）=0操作成功；DX：AX新指针位置，（CF）=1出错，AX＝错误代码。

1）AL=00 绝对移动方式

偏移值从文件首开始计算。例如偏移值是50，则读写指针指向文件的第50字节。为了使指针指向文件首，可以在CX，DX，AL中都送入0，那么以后的读写操作就从文件首开始。

例7.12 是从文件首开始，移动指针1024字节：

```
MOV  AH, 42H
MOV  AL, 00
MOV  BX, HANDLE
MOV  CX, 00
MOV  DX, 1024
INT  21H
JC   ERROR
```

2）AL=01 相对移动方式

当前的指针值加上偏移值作为新的指针值，也就是说，偏移值指出了从当前的读写位置起移动的字节数。根据偏移值的正负可正向或反向移动指针。

例7.13 从当前文件指针位置向前或向后移动N个字节。

```
MOV  BX, HANDLE
MOV  CX, 0
MOV  DX, N
CMP  DX, 0
JGE  POINT
NOT  CX
POINT：MOV  AL, 1
MOV  AH, 42H
INT  21H
 JC     ERROR
```

如果希望得到指针的当前值，可以使用方式1和偏移值0。

3）AL=2 绝对倒移方式

新的指针位置通过把偏移值和文件尾的位置相加而确定。如果CX和DX为0，AL为2，则移动后的指针将指向文件尾，实际上这个新指针值就是文件的长度，用这种方式可以检测文件的大小。如果要在一个已存在的文件后面添加记录，则利用方式2在写之前把指针指向文件尾。

```
MOV  AH, 42H
MOV  AL, 2
MOV  BX, HANDLE
MOV  CX, 0
MOV  DX, 0
INT  21H
JC   ERROR
```

移动读写指针功能可出现的错误码是01和06，错误码01说明AL中的方式值是不合法的，错误码06说明BX中的文件代号不合法。

如果指针移动成功，AX和DX将是移动后的指针值，AX中是低位字，DX是高位字（调用之前，DX是偏移值的低位字，CX是偏移值的高位字）。

移动指针的功能使用起来很简单，一般在程序中打开文件之后，使用这个功能的某种方式将读写指针移到文件中需要的位置，以后的读写就从文件的这个地方开始，从而提供了在文件中随机存取的能力。

4．文件操作综合举例，例7.14 文件复制。

```
DATA   SEGMENT
A      DB “FILE1.DAT”, 0  ；源文件名
B      DB “FILE2.DAT”, 0  ；目标文件名
C      DW  0，0，0
D      DB  7D00H DUP （0）  ；缓冲区大小32KB
DATA   ENDS
CODE   SEGMENT
ASSUME CS: CODE, DS: DATA
BBGIN: MOV  AX, DATA
   MOV  DS, AX
   MOV  DX, OFFSET A    ；指向源文件名
  MOV  AX, 3D00H  ；AH=3DH，AL=00H为读
  INT  21H          ；打开被复制文件
  MOV  BX, AX         ；文件句柄送BX
  MOV  AH，3FH        ；功能号
  MOV  CX，7D00H      ；待读长度32KB
  MOV  DX，OFFSET D    ；缓冲区首址
  INT  21H          ；读文件到缓冲区
  MOV  BX，AX         ；读入的字节数
  MOV  AH，3CH        ；功能号
MOV  DX，OFFSET  B   ；指向复制文件名
MOV  CX，0H         ；文件属性，一般
INT  21H          ；建立文件
MOV  CX，BX         ；字节数
MOV  BX，AX         ；文件句柄送BX
MOV  AH，40H        ；功能号
MOV  DX，OFFSET  D   ；缓冲区首址
INT  21H          ；写缓冲区到文件
MOV  AH，3EH
INT  21H          ；关闭BX句柄文件
MOV  AH，4CH
INT  21H
CODE   ENDS
END  BEGIN
```

例7.15 建立C:\MYFIEL.DAT文件，打开后将内存中字符串26个字符“ABCD…Z”写入该文件，然后关闭该文件。重新打开该文件，读出从字符“E”开始的10个字符到内存缓冲区，再建立并打开C:\MYFILE2.DAT文件，然后将缓冲区中的字符写入该文件，最后关闭这两个文件。

```
DATA  SEGMENT
F1    DB  “C:\MYFILE1.DAT”,0
F2    DB  “C:\MYFILE2.DAT’,0
STR   DB
“ABCDEFGHIJKLMNOPQRSTUVW
XYZ”
BUF   DB   20 DUP (0)
DATA  ENDS
CODE  SEGMENT
ASSUME  CS: CODE, DS: DATA
BEGIN: MOV  AX, DATA
    MOV  DS, AX
    MOV  DX, OFFSET F1    ；指向
文件名C:\MYFILE1.DAT
    MOV  CX, 0          ；文件属性，
一般
    MOV  AH, 3CH        ；建立该文
件
   INT  21H
   MOV  SI, AX        ；保存文件句柄
   MOV  DX, OFFSET STR   ；字符
串首址
   MOV  CX, 26        ；字符数26
   MOV  BX, AX    ；文件句柄送BX
   MOV  AH, 40H     ；写字符串到BX
指定的文件
   INT  21H
   MOV  BX, SI     ；重置文件号
   MOV  AH, 3EH      ；关闭文件
   INT  21H       ；BX指定的文件
   MOV  DX, OFFSET F1    ；指向文
件名
   MOV  AL, 0         ；文件属性
   MOV  AH, 3DH ；打开指定的文件
  INT  21H
```

```
  MOV  SI, AX   ；保存文件句柄
  MOV  BX, AX   ；文件句柄送BX
  MOV  AH, 42H
  MOV  AL, 00   ；绝对移动方式
  MOV  CX, 00
  MOV  DX, 5
  INT  21H      ；移动文件指针
  MOV  BX，SI
  MOV  CX,10 ；字符个数10（待读）
  MOV  DX, OFFSET BUF   ；指向缓冲区
  MOV  AH, 3FH   ；读指定文件
  INT  21H
  MOV  DI, AX
  MOV  DX, OFFSET F2
  MOV  CX, 0
  MOV  AH, 3CH        ；建立C:\MYFILE2.DAT文件
  INT  21H
  MOV  DX, OFFSET BUF   ；指向缓冲区
  MOV  CX, DI         ；原读入的字符数变待写字符数
  MOV  BX, AX
  MOV  AH, 40H    ；写数据
  INT  21H
  MOV  AH, 3EH    ；关闭文件MYFILE2.DAT
  MOV  BX，SI
  INT  21H
  MOV  AH，4CH
  INT  21H
CODE  ENDS
END   BEGIN
```