微计算机实验讲义

东南大学信息科学与工程学院

二 O 一五年三月

目　　录

第一部分　　微计算机原理实验

第二部分　　TPC—2003 通用微机接口实验

第三部分　　附录总汇

第一部分　微计算机原理实验

微计算机原理实验内容共安排了 12 个实验，其目的是使读者熟悉 8086 指令系统和掌握汇编语言程序设计的方法，并能利用 Tubo Debugger 调试工具来调试汇编程序；为读者从事计算机研究与应用打下一个坚实的基础。

汇编语言是唯一能够充分利用计算机硬件特性的面向机器的语言，后半部分实验是编制有关控制程序，使用 IBM-PC 机器内部的中断控制器、定时器、并行接口完成相应的工作；藉以熟悉机内各接口芯片的工作和使用方法，了解 Intel 8086CPU 的中断和系统功能调用的使用。

所列出的实验均可用 IBM-PC / XT 以上型号的微机来完成。

实验一　数据传送

一、实验目的

1. 熟悉 8086 指令系统的数据传送指令，进一步掌握传送指令的寻址方式。

2. 利用 Turbo Debugger 调试工具来调试汇编程序。

二、实验任务

1. 通过下述程序段的输入和执行，来熟悉 Turbo Debugger 调试工具的使用，并通过显示屏来观察程序的执行情况。

程序段

```
MOV BL，08H
MOV CL，BL
MOV AX，03FFH
MOV BX，AX
MOV DS：[0020]，BX
```

步骤；

(1)TD↙(划线部分是实际输入的内容，↙代表回车键，下同)，此命令是调用调试程序 Turbo Debugger，显示屏出观 CPU 窗口(操作步聚的详细说明可参考附录一)。

(2)输入程序段(利用 ↓、↑ 光标键移动光标，从光标所指的行开始输入)

例：MOV BL，08H↙
　　MOV CL，BL↙
　　⋮

(3)执行程序段

①单步跟踪：把光标移到程序段开始处，按 ALT + F10 键，CPU 窗口显示代码区局部菜单，利用 ↓、↑ 光标键将光标移到 **NEW CS：IP** 处，按 Enter 键，这时 **IP** 指针(程序位置计数器)就移到当前光标所指示的地址处，然后每按一下 F7 键，就执行一条指令，如果跟踪执行程序时，错过了认为可能有错误的地方，并且想返回到该处时，可以用 ALT + F4 键将 IP 指针返回到某一指定点。

②设置断点:光标移到某一条指令行上，按 F2 键来设置或清除断点．运行程序段时，按 F4 键，CPU 从 **IP** 指针开始执行到断点位置停止。

(4)检查各寄存器和存贮单元的内容

寄存器区显示在 CPU 窗口的中部，若检查存贮单元的内容，按 ALT + V 键，弹出 View 的下拉菜单，选 **DUMP** 项，按 Enter 键，出现 DUMP 窗口，再按 ALT + F10 ，弹出局部菜单，选 GOTO 项后，输入存储单元的地址 DS：0020 √，在 DUMP 窗口显示该单元内容，按 ALT + F3 键关闭当前窗口。

2. 用 PUSH 指令将一组数据压入堆栈区，通过三种不同的出栈方式出栈，看出栈后数据的变化情况，并把结果填入表 1-1-1 中。

程序段

```
MOV  AX，0102H
MOV  BX，0304H
MOV  CX，0506H
MOV  DX，0708H
PUSH  AX
PUSH  BX
PUSH  CX
PUSH  DX
```

第一种出栈方式：

```
POP   DX
POP   CX
POP   BX
POP   AX
```

第二种出栈方式：

```
POP   AX
POP   BX
POP   CX
POP   DX
```

第三种出栈方式：

```
POP   CX
POP   DX
POP   AX
POP   BX
```

表 1-1-1

第一种	第二种	第三种
(AX)=	(AX)=	(AX)=
(BX)=	(BX)=	(BX)=
(CX)=	(CX)=	(CX)=
(DX)=	(DX)=	(DX)=

3.指出下列指令的错误，并加以改正

(1)MOV [BX]，[SI]

(2)MOV AH，BX

(3)MOV AX，[SI][DI]

(4)MOV BYTE PTR[BX]，2000H

(5)MOV CS，AX

(6)MOV DS，2000H

4．设各寄存器及存储单元的内容如下：

(DS)=1000H，(BX)=0010H，(SI)=0001H，(10010)=12H，(10011)=34H

(10012)=56H，(10013)=78H，(10120)=ABH，(10121)=CDH，(10122)=EFH

说明下列各条指令执行完后 AX 寄存器中的内容，并上机验证

(1)MOV　AX，1200H

(2)MOV　AX，BX

(3)MOV　AX，[0120]

(4)MOV　AX，[BX]

(5)MOV　AX，0110[BX]

(6)MOV　AX，[BX][SI]

(7)MOV　AX，0110[BX][SI]

5.将 DS：1000H 存储单元的内容送到 DS：2020H 单元中存放，试分别用 8086 的直接寻址、寄存器间接寻址、变址寻址传送指令编写程序段，并上机运行检查结果。

1000：0010	FF
1000：0011	EE
	⋮
2000：0020	DD
2000：0021	CC

图 1-1

6．设 AX 寄存器中的内容为 1111H，BX 寄存器中的内容为 2222H，DS:0010H 单元中的内容为 3333H。将 AX 和 BX 寄存器中内容进行交换，然后再将 BX 寄存器中的内容和 DS：0010H 单元中的内容进行交换。试编写程序段，并上机运行检查结果。

7.设(DS)=1000H，(ES)=2000H，有关存储器的内容如图 1-1-1 所示，将 DS 段的内容传送到 AX 寄存器；ES 段的内容传送到 BX 寄存器，编写程序段。

三、实验设备

IBM-PC / XT 微机　　一台

四、实验预习要求

1．复习 8086 指令系统的中传送类指令，了解传送指令的寻址方式。

2．预习附录一上机操作中第 8 节的内容．调试程序 Turbo Debugger 及其使用。

3．按照题意要求在实验前编写好程序。

五、实验报告要求

1．写明本次实验内容和实验步骤。

2．整理出运行正确的各题源程序段和结果。

3．小结 Turbo Debugger 调试工具的便用方法。

实验二　基本算术和逻辑运算

一、实验目的

1. 熟悉算术和逻辑运算指令的功能。

2. 进一步了解标志寄存器各标志位的意义和指令执行对它的影响。

二、实验任务

1.采用单步执行方式执行下列各程序段，检查各标志位的情况。

程序段 1：

```
MOV  AX，1010H
MOV  SI，2000H
ADD  AL，30H
ADD  AX，SI
MOV  BX，03FFH
ADD  AX，BX
MOV [0020]，1000H
ADD [0020],AX
```

程序段 2：

```
MOV  AX，0A0A0H
ADO  AX，0FFFFH
MOV  CX，0FF00H
ADD  AX，CX
SUB  AX，AX
INC  AX
OR   CX，00FFH
AND  CX，0F0FH
MOV  [0010]，CX
```

程序段 3：

```
MOV  BL，25H
MOV  [0010]，04H
MOV  AL，[0010]
MUL  BL
```

程序段 4：

```
MOV  BL，04H
MOV  WORD  PTR[0010]，0080H
```

```
MOV  AX，[0010]
DIV   BL
```

程序段 5：

```
MOV  AX，00
DEC  AX
ADC  AX，3FFFH
ADD  AX，AX
NOT  AX
SUB  AX，3
OR   AX，0FBFDH
AND  AX，0AFCFH
SHL  AX，1
RCL  AX，1
```

步骤：

(1)进入 Turbo Debugger，在 CPU 窗口下输入程序段。

(2)将 IP 指针指向程序段开始处。

(3)按下 F7 键(单步)运行程序。

(4)分析各条指令执行后的结果与各标志位在指令执行后对它的影响。

2．将寄存器 BX 作地址指针，自 BX 所指的内存单元(0010H)开始连续存放着三个无符号数(10H、04H、30H)。试编写程序分别求它们的和与积，并将结果存放在这三个数之后的单元中。

3．写出完成下述功能的程序段

(1)传送 15H 到 AL 寄存器。

(2)将 AL 的内容乘以 2。

(3)传送 15H 到 BL 寄存器。

(4)AL 的内容乘以 BL 的内容。

最后结果(AX)=?

4．写出完成下述功能的程序段

(1)从地址 DS：0000H 单元中，传送一个数据 58H 到 AL 寄存器。

(2)把 AL 寄存器的内容右移两位。

(3)再把 AL 寄存器的内容与字节单元 DS：0001H 中的数据 12H 相乘。

(4)将乘积存入字单元 DS：0002H 中。

5.假设下面的程序段用来清除数据段中相应字存储单元的内容(即零送到这些存储单元中去)，其偏移地址从 0010H 到 0020H

(1)将第 4 条比较指令语句填写完整(划线处).

```
        MOV  SI，0010H
NEXT：  MOV  WORD  PTR[SI],00
```

```
ADD   SI，2
CMP   SI，_____
JNE   NEXT
```

(2)假设要清除偏移地址从0020H到0010H字存储单元中的内容(即由高地址到低地址清零)，试编写程序段。

三、实验设备

IBM-TCC / XT 微机　　一台

四、实验预习要求

1.复习 8086 指令系统的算术和逻辑运算指令。

2.按照题目要求在实验前编写好程序。

五、实验报告要求

1. 整理出运行正确的各题源程序段和结果。

2. 简要说明 ADD 指令和 AND 指令对标志位的影响。

实验三　简单编程练习

一、实验目的

1.利用已掌握的宏汇编语言，进行简单的程序设计练习。

2．学习和掌握建立与运行汇编语言源程序各个步骤的命令。

3．熟悉汇编程序的调试过程。

二、实验任务

1．在一个数据块中找出最大数。

假设有数据 22、46、32，72、84、16，且为不带符号的正整数，数据块的长度存放在 CX 寄存器中，找出的最大数存放在以 MAXN 为符号的单元中。

2．求无符号字节数据之和，和数为 8 位二进制数。

假设有数据 38 、55、26、12、23，数据块的长度存放在 CX 寄存器中，和数存放在以 SUM 为符号的单元中。

3．求无符号字节数据之和，和数为 16 位二进制数。

假设有数据 58、25，45，73、64，43，数据块的长度存放在 CX 寄存器中，和数存放在以 SUM 为符号的字单元中。

4.求两个十进制数相乘的积(56093 X 5 = ?)，被乘数和乘数均以非组合 BCD 码表示，并存放在内存中，乘积存放在以 SUM 为符号的单元中。

5.试分别用数据传送指令和串传送指令编写程序，将以 STR1 为首地址的字节存储单元中的数据 30H、31H，32H、33H，34H、35H、36H、37H、38H，39H、40H、41H，42H，43H，44H、45H，传送到以 STR2 为首地址的字节存储单元中。

三、实验原理

为完成以上的实验任务，现举例说明编程的步骤。

程序实例

在一个有正、负数的数据块中，找出负数的个数，假设有数据-19、28，37、-46，55、61、-74，数据块的长度存放在 CX 寄存器中，负数的个数存放在以 SUM 为符号的单元中。

步骤：

(1)根据题目要求，画出程序流程图如图 1-3-1，并编写程序。

(2)编辑源文件(编辑过程参见附录中的建立与修改源文件)，设此时源文件已建立在磁盘上，文件名：FNUM. ASM

```
DATA    SEGMENT
NUM     DB -19，28，27，-46，55，61，-74
SUM     DB ?
DATA    ENDS
MAIN    SEGMENT
        ASSUME  CS：MAIN，DS：DATA
```

```
START：MOV  AX，DATA
       MOV  DS，AX
       MOV  CX，7
       MOV  AL，00
       LEA   SI，NUM
AGAIN：MOV  BL，[SI]
       CMP   BL,00
       JGE   NEXT
       INC  AL
NEXT：  INC  SI
       LOOP   AGAIN
       MOV   SUM，AL
MAIN    ENDS
       END   START
```

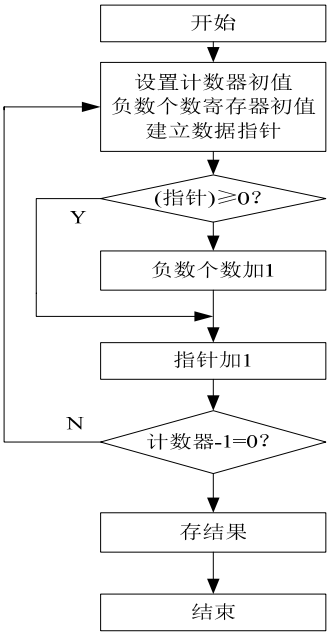


图1-3-1 程序流程图

(3)汇编源文件

C〉MASM FNUM. ASM；↙

如汇编结果正确则做第 4 步，否则月

(4)C>LINK FNUM；↙

(5)C>TD FNUM↙

进入 Turbo Debugger 调试程序，使用单步执行键 F7 执行程序，并检查 SUM 为符号的单元[DS：0007]=03H

四、实验设备

IBM-PC / XT 微机　　一台

五，实验预习要求

1．仔细阅读实验指导书。

2．按照题目要求画出流程图，并在实验前编写好源程序。

六、实验报告要求

1．整理出经过运行而且证明是正确的源程序，并加上注释。

2. 整理出正确的运行结果。

3. 比较数据传送指令和串传送指令两种传送方式各有何特点。

4. 回答思考题。

七、思考题

1．在例题中，“CMP BL，00”指令有何作用？

2．“CMP BL，00”指令是否可以用其它指令代替？

实验四　代码转换

一、实验目的

1.掌握 ASCII 码转换为 BCD 码和 BCD 码转换成二进制数的方法。

2．进一步掌握建立与运行汇编语言源程序的操作方法。

二、实验任务

1.将十进制数字(0～9)的 ASCII 码 30H～39H 转换为组合的 BCD 码；其在内存中的存放形式为 10H、32H、54H、76H、98H。

2．将用 BCD 码表示的数 8765H 转换成用二进制表示的数据。

三、实验原理

1.通常在微计算机中，从键盘上输入的十进制数的每一位数码(即 0-9)都是以它的 ACII 码表示的，要向 CRT 输出的十进制数的每一位也是用 ASCII 码表示，而计算机中的十进制数，是以 BCD 码形式存放，或是将其转换成二进制数存放．因此，在进行十进制数运算之前，首先要将 ASCII 码转换为 BCD 码，这时只要将一个字节，ASCIl 码的高四位清零即可，如果要以组合的 BCD 码形式表示，即一个字节存放两个数字的 BCD 码，则要将两个字节 BCD 码中高位字节数字左移四位，再与低位字节组合在一起就可以了。ASCII 码转换为 BCD 码的参考流程图如图 l-4-l 所示。

2．在进行较复杂的十进制运算时，就要将 BCD 码转换为二进制数。若将两个存储单元存放的 4 个数字的 BCD 码转换成二进制数，常用的一种方法是

((千位数 X10+百位数)X10+十位数)X10+个位数

就可以得到转换结果。

BCD 码转换为二进制数的参考流程图如图 1-4-2 所示。

四、实验设备

IBM-PC/XT 微机　　一台

五、实验预习要求

1. 仔细阅读实验指导书。

2．按题目要求，参考流程图，在实验前编写好源程序。

六、实验报告要求

1. 整理经过运行而且证明是正确的源程序，并加上注释。

2．整理正确的运行结果。

3. 回答思考题

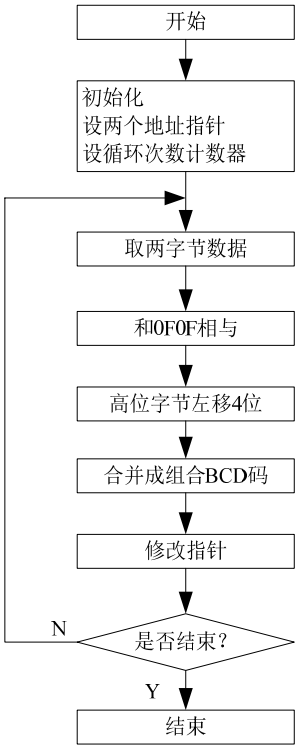


图 1-4-1　ASCII 码转换为 BCD 码的程序流程图

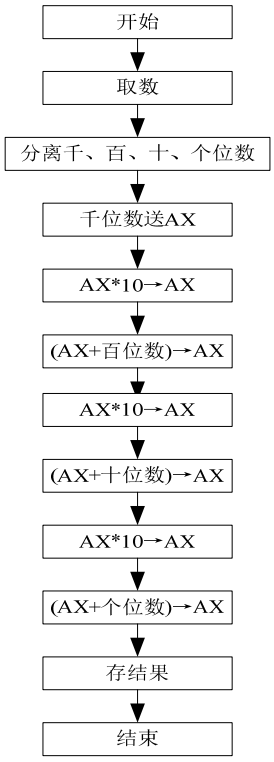


图 1-4-2　BCD 转换为二进制数的程序流程图

七、思考题

1．十进制数字在计算机中常用 BCD 码表示，试问如何将其转换成 ASCII 码?

2．若十进制数用组合的 BCD 码表示，又如何将其转换成相应的 ASCII 码?

实验五　字符匹配

一、实验目的

1. 掌握串操作指令的使用方法。

2. 学习如何使用系统功能调用从键盘输入字符串，并在显示器上显示。

二、实验任务

用串操作指令设计程序，由键盘输入字符(≤50 个)，并存放于数据区中，现要求在存储区中寻找空格字符(20H)，同时进行计数，返回 DOS 前，将计数结果显示在屏幕上，否则显示“没有空格字符”提示。

三、实验原理

1. 在系统功能调用 INT 21H 调用号 01H 的功能是由键盘输入字符。当执行 01H 时，计算机等待键盘输入，直到按下一个键后，将这个字符送到 AL 寄存器，并在屏幕上显示出来。

调用号 02H 的功能是将 DL 寄存器中的字符显示在屏幕。

调用号 09H 的功能是将一缓冲区中的字符串显示在当前的光标处，使用时，要求用 DS 和 DX 指出要显示的字符串的首地址。另要注意用“$”作为字符串的结束符。

调用号 0AH 的功能是将键盘输入的字符串送到一缓冲区，使用时，在入口参数中，要求 DS 和 DX 寄存器给出输入缓冲区的首地址，并且在缓冲区第一个字节中预置缓冲区长度。功能调用被执行后，缓冲区第二个字节中为实际输入字符信息的长度，从第三个字节开始才是输入的字符串，最后以回车(0DH)作结束，回车符也被送入缓冲区，但不计入输入的字符个数之中。

2. 举例说调用号的用法

利用系统功能调用 INT 21H 中调用号 02H 来显示提示符“:’，调用号 01H 由键盘输入字符。按下回车键，用调用号 09H 来输出的字符串。

程序如下：

```
DATA  SEGMENT
      NUM  20 DUP(?)
DATA  ENDS
CODE  SEGMENT
      ASSUME  CS：CODE，DS:DATA
STA：  MOV  AX，DATA
       MOV  DS，AX
       MOV  DL，″：″
       MOV  AH,02H
       INT   21H
       LEA   BX，NUM
       MOV  DL，0DH
```

```
        MOV    [BX], DL
        INC    BX
        MOV    DL, 0AH
        MOV    [BX1, DL
        INC    BX
LLL:    MOV    AH, 01H
        INT    21H
        MOV     [BX], AL
        INC     BX
        CMP     AL, 0DH
        JNZ     LLL
        MOV     DL, "$"
        MOV     [BX-1], "D"
        LEA     DX, NUM
        MOV     AH, 09H
        INT     21H
        MOV     AH, 4CH
        INT     21H
CODE    ENDS
ENDS    STA
```

3．完成字节匹配程序的参考流程图，如图 1-5-1 所示。

四、实验设备

IBM-PC / XT 微机　　一台

五、实验预习要求

1．仔细阅读实验指导书。

2．按题目要求，参考流程图，在实验前按图 1-5-1 程序流程图编写好源程序。

六、实验报告要求

1．整理经过运行而且证明是正确的源程序，并加上注释。

2．记录程序执行后的结果。

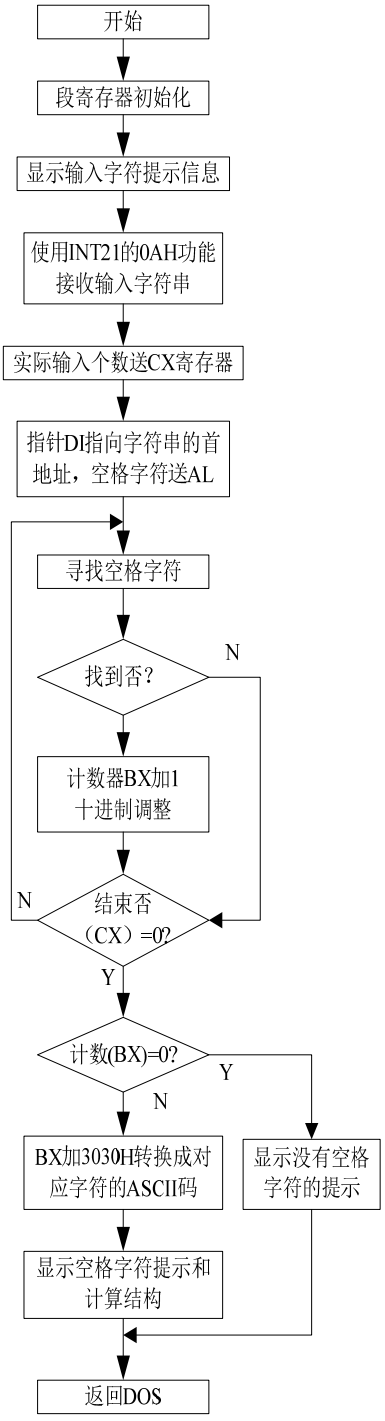


图 1-5-1 程序流程图

实验六　排　序

一、实验目的

1. 掌握用汇编语言编写排序程序的思路和方法。

2. 进一步巩固系统功能调用的使用知识。

二、实验任务

由键盘输入字符(≤50 个)，并存放于数据区中，将这些字符按照 ASCII 码大小，从小到大进行排序，排序后，仍放在该数据区中，并将这些字符在屏幕上显示出来。

三、实验原理

排序方法:采用先取数据区中第 N 个数 ε_n 与第 N-1 个数 ε_{N-1} 相比较，若 $\varepsilon_N \geqslant \varepsilon_{N-1}$ 则不交换位置，反之则交换；然后取 ε_{N-1} 与 ε_{N-2} 相比较，按同样原则决定是否要交换，这样一直比下去，最后将 ε_2 与 ε_1 相比较，也按同样原则决定是否要交换。则当第一次循环比较结束时，数据区中的最小值就被升到了顶部。但数组还未按大小顺序排列好，就要进行第二次循环比较，这样数据区中第二个最小值就升到了顶部的相应位置，再进行第三次循环比较.... 经过数次以后，数据区中的数就可以按大小顺序排列了。

另外，为了在程序中除去不必要的循环，就要设置一个标志，每次循环开始时，置此标志为零；若在一次循环比较中，没有发生数据交换，则此标志仍为零；若发生过交换，则此标志置为-1，然后在下一次循环开始前，检查此标志，若不为零，表示数组尚未排列有序，继续进行比较排序；若为零，则表示数组已按大小排列有序，就停止循环。

排序的参考流程图，如图 1—6-1 所示。

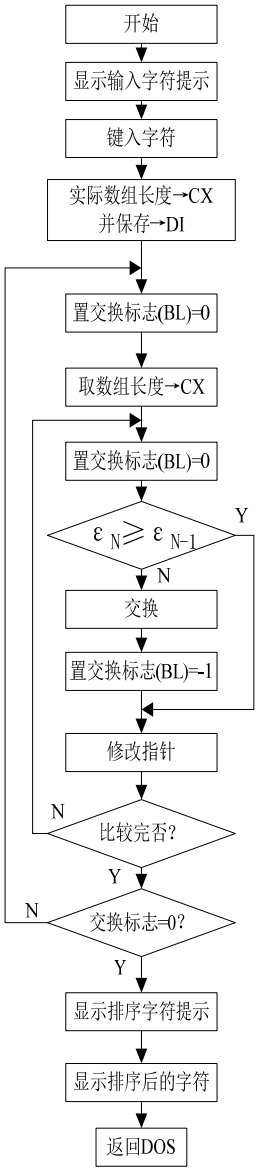


图 1-6-1 程序流程图

四、实验设备

IBM-PC / XT 微机　　一台

五、实验预习要求

1. 仔细阅读实验指导书。

2. 按题目要求，参考流程图，在实验前编写好源程序。

六、实验报告要求

1. 整理经过运行而且证明是正确的源程序,并加上注释。
2. 记录程序执行后的结果。

实验七　设置光标位置

一、实验目的

1.了解 IBM-PC 基本输入输出系统 ROM-BIOS 以及屏幕显示软中断(INT 1OH)的各种功能。

2．掌握使用 INT 1OH 的 02H 和 0AH 功能调用设置光标位置以及在当前光标位置写字符的方法。

二、实验任务

1．设置光标起始位置为第 11 行第 36 列，并从当前光标位置开始连续显示 8 个梅花，其 ASCII 码为 05H(可查附录三中 IBM-PC ASCII 码字符)。

2.在最左边梅花的下方显示光标，并且每隔 1 秒左右光标向右移动一次，直至移到最右边的梅花下方为止。即光标移动 8 次后,返回 DOS 系统。

其屏幕显示情况如图 1-7-1 所示。

图 1-7-1

三、实验原理

IBM-PC 基本输入输出系统 ROM—BIOS 的屏幕显示软中断 INT 1OH 功能很强，它不仅能设置不同的显示方式，在屏幕的任意位置上显示字符或图形点，还能控制屏幕滚动、换页、设光标位置等。

在汇编源程序中可以用"INT 1OH" 调用显示器驱动程序,能方便地控制屏幕显示。通常 BIOS 子程序不保护寄存器 AX 的内容。如果程序中使用了 AX，则一定要在调用 BIOS 子程序前、后、保护、恢复 AX 的内容。

本次实验中所调用屏幕显示软中断(TNT 1OH)的 0 号、01 号、02 号、0AH 号和 0FH 号功能说明如下：

1. 功能号 0　设置显示方式；

入口参数：(AH)=0, (AL)=方式号(0—7)。

（AL）=0　　40X25 黑白文本方式

（AL）=1　　40X25 彩色文本方式

（AL）=2　　80X25 黑白文本方式

（AL）=3　　80X25 彩色图形文本方式

（AL）=4　　320X200 黑白图形方式

（AL）=5　　320X200 彩色图形方式

（AL）=6　　640X200 黑白图形方式

（AL）=7　　80X25 黑白文本方式(单色显示板)

出口参数：无

例如：

```
MOV    AH，0      ；请求设置显示方式
MOV    AL，2      ；80X25 黑白文本方式 2→AL
INT     10H       ；调用 BIOS 的 10H 号中断处理程序
```

2．功能号 01　设置光标大小

入口参数：(AH)=1，(CH)=光标顶值，(CL)=光标底值；

出口参数：无

光标符号类似于下横线，它不是 ASCII 字符。根据需要可在规定的范围内调整光标的大小。

例如：

```
MOV  AH，01        ；请求设置光标大小
MOV  CX，0B0CH   ；光标顶值 0BH→CH，底值 0CH→CL
INT  10H
```

上述三条指令执行后，光标闪亮，类似于下横线。CX 取不同的值 ，则光标大小随着做相应的变化。CH 取值范围为 1—11,CL 取值范围为 1--12。

3.功能号 02　设置光标位置

入口参数：(AH)=2，(BH)=页号，(DH)=行号，(DL)=列号。

出口参数：无，

当显示器的行于列被确定为 mXn 之后，屏幕变成了一个坐标网。根据需要将光标设置在屏幕上可寻址任何位置。

例如：

```
MOV  AH，02      ；请求设置光标位置
MOV  BH，0       ；页号为 0
MOV  DX，050AH   ；行号为 5；列号为 10
INT  10H
```

上述四条指令执行后，光标设置在第 5 行第 10 列位置

4.功能号 0AH　在当前光标位置写字符

例如：

```
MOV    AH，0AH    ；请求写字符
MOV    AL，2AH    ；"*"的 ASCII 码→AL
MOV    BH，0      ；页号 0→BH
MOV    CX，4      ；重复次数 4→CX
INT    10H
```

上述五条指令执行后，在当前光标位置开始连续显示出 5 个“*”。

5．功能号 0FH　读当前显示状态

入口参数：(AH)=0FH

出口参数：(AL)=当前显示方式，(BH)=当前页号，(AH)=屏幕上字符列数。
根据图 1-7-2 程序流程图,编写源程序。

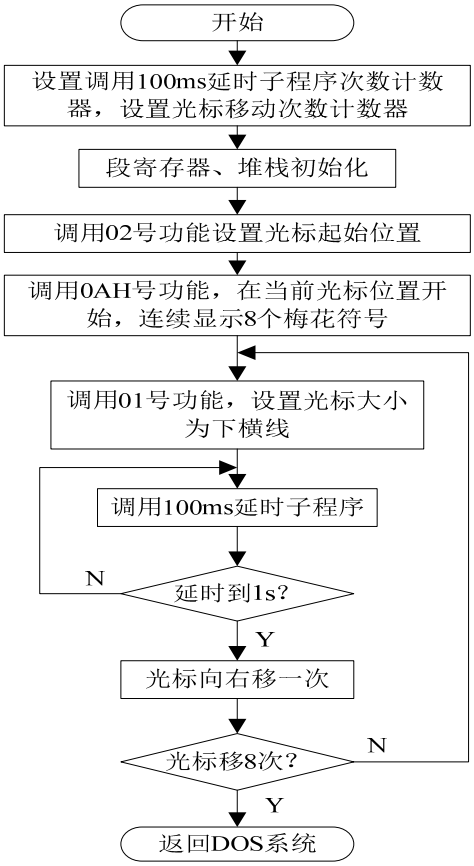


图1-12　程序程程图

给出 100ms 延时子程序如下：(仅供参考)

```
DELAY   PROC   NEAR
        PUSH   CX
        PUSH   DX
        MOV    DX，50
DL500：MOV    CX，2801
DL10MS：LOOP   DL10MS
        DEC    DX
        JNZ    DL500
        POP    DX
```

```
        POP    CX
        RET
DELAY   ENDP
```

四、实验设备

IBX-PC / XT 微机　　一台

五、实验预习要求

1. 正确理解本次实验的目的、内容和原理。

2. 根据流程图在实验前编写好源程序。

六、实验报告要求

1. 整理出经过运行而且证明是正确的源程序，并加上注释。

2，写出实验结果。

3. 分析在实验中所出现的问题。

4. 回答思考题。

七、思考题

1. 试问在当前光标位置写字符时，是否要改变光标位置?

2. 每次写字符之前,是否都要重新设置光标位置?为什么?

实验八　显示两个十进制数之和的实验

一、实验目的

1．学习数据传送和算术运算指令的用法。

2．了解和掌握由键盘接收数据，并显示其运算结果的方法。

3．掌握 INT 21H 的 09H 和 0AH 的功能。

4. 掌握串操作指令的使用方法。

二、实验任务

1．由键盘输入两位十进制数，进行相加，将其和显示或打印出来。

2．要求当程序执行时显示：

INPUT DATA：XX；XX↙(由键盘输入两个任意的两位十进制数)

程序执行后显示：

OUTPUT VALUE：XX＋XX=XXXX(显示结果为四位十进制数)

三、实验原理

键盘输入的数据，是以 ASCII 码的形式存放在由程序所定义的当前数据段中的。将它们转换成两个两位 BCD 码，进行相加，其和也应是组合的 BCD 码，然后再转换为 ASCII 码显示出来。

本次实验中将使用 DOS 系统的功能调用 09H、0AH，其功能说明如下：

1. 功能号 09H　显示字符串

格式：

```
LEA   DX,字符串首偏移地址
MOV   AH,9
INT   21H
```

以上三条指令，使当前数据区中 DS:DX 所指向的以“$”结尾的字符串送显示器显示。

2．功能号 0AH　键盘输入字符串

格式:

```
LEA    DX,缓冲区首偏移地址
MOV    AH,10
INT    21H
```

以上三条指令，由键盘往 DS:DX 所指的输入缓冲区输入字符串并送显示器显示。

```
BUF   DB 11                    ；该缓冲区大小为 11 个字符
      DB ?                     ；存放实际输入字符的个数
      DB 10 DUP(0)             ；存放 10 个字节的字符串
```

缓冲区中第一个字节规定了缓冲区的大小，键盘输入的字符串从第三个字节存放，最后以回车(0DH)作结束。第二个字节存放实际输入的字符个数。

根据程序流程图 1—8—1 编制程序。

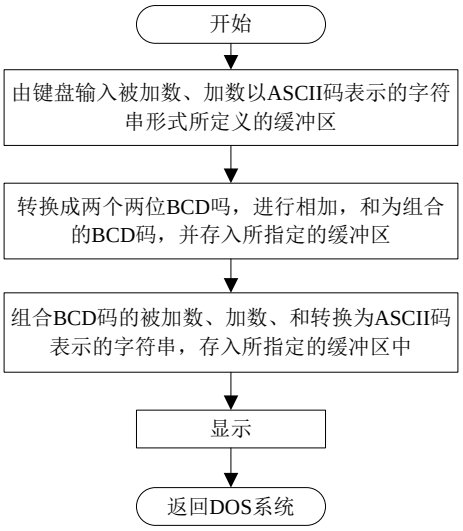


图1-8—1　程序流程图

四、实验设备

IBM-PC / XT 微机　　一台

五、实验预习要求

1．正确理解本次实验的目的、内容和原理。

2．在上机前根据流程图编出源程序。

六、实验报告要求

1．整理出经过运行而且证明是正确的源程序，并加上注释。

2．打印运算的实际结果，其形式为：

OUTPUT VALUE：XX+XX=XXXX

3．分析在实验中所遇到的问题。

4．回答思考题。

七、思考题

1. 使用 INT 21H 的 0AH 功能时，要求用哪几个特定的寄存器?它们起什么作用? 0AH 的功能是如何存放输入字符串?

2. 调用 INT 21H 的 09 功能时，特定的寄存器是哪几个?起何作用?用什么字符作为显示字符串的结束符?

实验九　时钟实验

一、实验目的

1．熟悉系统功能调用 INT 21H 的有关功能。

2．编写时钟程序。

二、实验任务

1．执行时钟程序时，屏幕上显示提示符“：” ，由键盘输入当前时、分和秒值，即 XX：XX：XX√，随即显示时间并不停地计时。

2. 当有键按下时，立即停止计时，返回 DOS。

三、实验原理

首先利用系统调用 INT 21H 中 02H 功能，在 CRT 上显示一个提示符“：”，要求用户从键盘输入时钟初值(即当前时间)，其输入格式为 XX(时)：XX(分)：XX(秒)√。然后利用 0AH 功能调用接收从键盘输入的字符串，并将接收的字符串存入到缓冲区。

在利用 0AH 功能调用前要设置一个缓冲区，在调用时，用 DX 作为输入缓冲区的指针，由键盘输入的字符存入该缓冲区，直至遇到回车键为止。

程序中把输入的‘时’、‘分’、‘秒’初值分别从输入缓冲区中取出，各自放在一个寄存器中，然后调用一个延时 1 秒钟的子程序，每过 1 秒使秒值增 1，然后检查是否已为 60 秒，若不是则转显示；若是，则使秒值为 0，分值增 1，检查是否已为 60 分，若不是则转显示，若是，则使分值为 0，时值增 1，接着检查时值是否为 24 小时，若不是则转显示，若是，则使时值为 0，接着也是转显示。

若使程序运行停止，只要有键按下，即可返回 DOS。下面列出两种判别是否有键按下的方法(仅供参考)。

一种方法是读键扫描码，指令如下：

```
        IN      AL，60H    ；读键扫描码
        TEST    AL, 80H
        JZ      AAA        ；有键按下，就转 AAA
                :
                :
AAA：   MOV     AH，4CH
        INT     21H
```

另一种方法是调用 INT 21H 中 06 功能，来判别是否有键按下，具体指令如下：

```
        MOV     AH，06
        MOV     DL，0FFH        ；判断是否有键按下，有键按下则转 AAA
        INT     21H
        JNZ     AAA
         :
         :
```

```
AAA:  MOV    AH, 4CH
      INT    21H
```

根据图 1—9—1 程序流程图，编写时钟源程序。

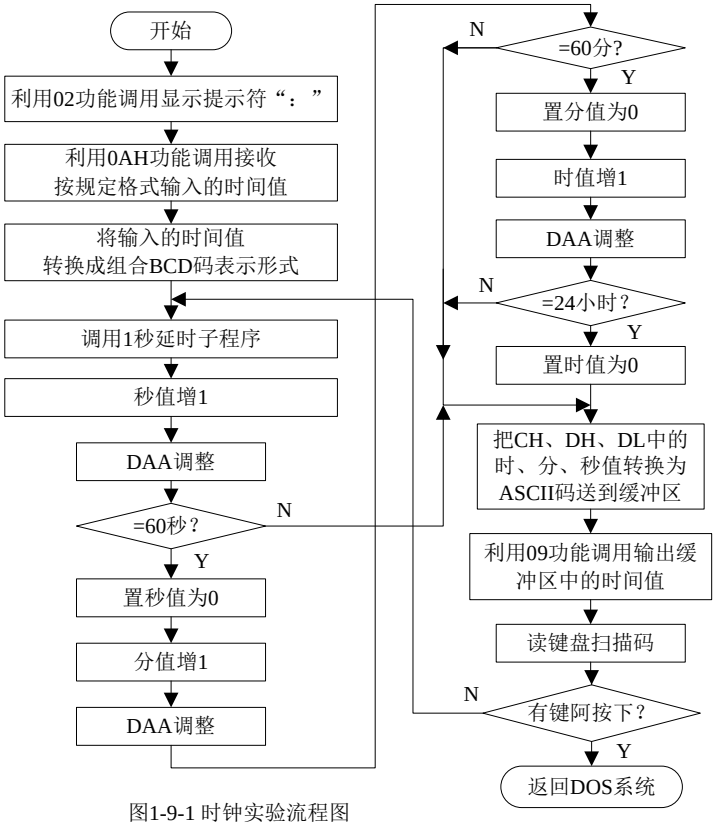


图1-9-1 时钟实验流程图

延时 1 秒子程序 DELAY

```
DELAY  PROC
       PUSH   CX
       PUSH   AX
       MOV    CX, 0FFFFH
GOON:  DEC    CX
       JNE    GOON
       POP    AX
       POP    CX
       RET
DELAY  ENDP
```

四、实验设备

IBM-PC / XT 微机　　一台

五、实验预习要求

认真预习本次实验，根据流程图在实验前编写好源程序。

六、实验报告要求

1．整理出运行正确的源程序，并加上注释。

2．写出实验结果。

3．分析实验中出现的问题并说明原因。

4．回答思考题。

七、思考题

时钟程序中存在时间误差吗?若有误差，其来源在何处?如何进行误差校正?

实验十　键盘中断

一、实验目的

1．了解 Intel 8086CPU 的中断处理功能以及 IBM-PC 的中断结构。

2．了解 8259 中断控制器的使用。

3．掌握键盘中断的编程，观察中断的执行情况。

二、实验任务

要求每按下任意一个键就向 CPU 发出中断请求信号，该信号由 8259 的 IRQ1 引入，中断类型号为 09，CPU 响应中断后转入执行 KEYINTS 中断服务程序，并在 CRT 上显示某字或某个图形(自定)，按下 10 次键后返回 DOS。

三、实验原理

键盘与主机是通过 5 芯螺旋形的电缆相连的，其中包括数据线、时钟线、复位线、+5v 电源线和地线。(电缆插入系统板后部的插座)

每当有键按下或释放时，键盘以串行方式向系统板的键盘接口电路传送数据，即扫码。一个扫描码移位传送完，键盘接口电路便向主机发出中断请求信号 IRQ1，此信号送到 8259A 产生中断请求。

CPU 响应中断请求时，由中断向量表中以 09X4 开始的连续四个单元中提供中断服程序 KEYINTS 的入口地址使之执行中断服务程序 KEYINTS。通过 8255A 的 PA 口(口地址为 60H)读取键盘扫描码后，输出 PB7(口地址为 61H)为高电平，并通过反相器变成低电平控制键盘状态触发器的清零端，使 IRQ1 清零，撤消中断请求信号。PB7 恢复低电平，允许位寄存器输出数据，这样就为传递下一个键盘扫描码作好了准备。

主程序和键盘中断服务程序的流程图如图 1-10-1 和图 1-10-2 所示。根据流程图编写主程序和键盘中断服务程序。

在主程序中应先保存原中断类型号 09H 中的内容，然后把中断服务程序入口地址的内偏移量和段地址存入以 09X4 为起始地址的四个单元内。

在键盘中断程序中保护现场，开中断后，要有键盘状态复位命令，具体指令如下；

```
IN      AL，61H     ；输入 PB 口的当前值
OR      AL，80H     ；PB7 置 1
OUT     61，AL
AND     AL，7FH     ；PB7 清零
OUT     61H，AL
```

因为端口地址为 61H 的 I／O 接口 D7 位(8255A 的 PB7)通过反相器接键盘状态触发器清零端，上述程序段先使 D7 位上产生一个正脉冲，从而使状态触发器复位，这就使键盘的中断请求信号得以清除。

在键盘中断程序结束前，应发出“中断结束”(EOI)命令(控制字为 20H)，给 8259A 中断控制寄存器(端口地址为 20H)。具体指令如下；

```
MOV     AL，20H
```

OUT　20H，AL

然后,实现中断返回。

另外，当键按下时，送向主机的扫描码是键编号，而键释放时，扫描码为键编号加80H(即第 7 位置 1)。例如按下和释放“A”键将向主机发送两个扫描码：1EH 和 9EH。

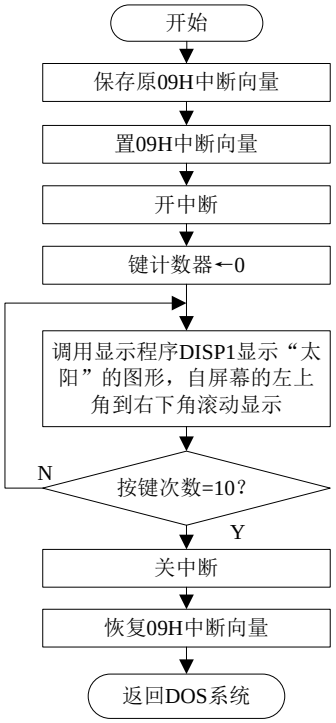


图 1-10-1

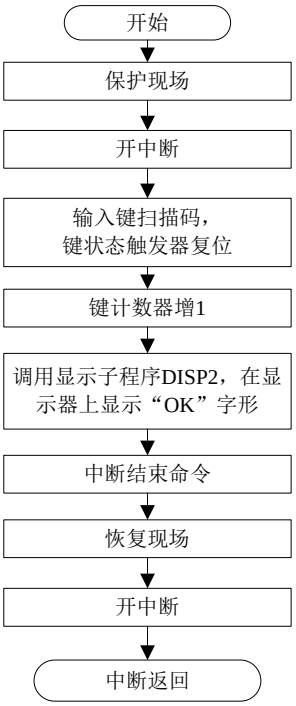


图 1-10-2

以下是主程序中显示“太阳”图形的子程序 DISP1，仅供参考。

```
DISP1  PROC  FAR
       PUSH  AX
       PUSH  BX
       PUSH  CX
       PUSH  DX
       MOV   AH，15      ；读当前显示状态
       INT   10H
       MOV   AH，0       ；设置显示方式
       INT   10H
       MOV   CX，1       ；要显示字符个数
       MOV   DX，0       ；行号为 0，列号为 0
```

```
REPT:   MOV  AH, 2          ; 设置光标位
        INT  10H
        MOV  AL, 0FH        ; 读出太阳图形
        MOV  AH, 10         ; 写字符
        INT   10H
        CALL  DELAY
        SUB   AL, AL
        MOV  AH, 10           ; 清除原图形
        INT   10H
        ADD  DL, 2
        CMP  DH, 25
        JNE   REPT
        POP   DX
        POP   CX
        POP   BX
        POP   AX
        RET
DISP1  ENDP
```

以下是中断服务程序中显示"OK"字符子程序 DISP2，仅供参考。

```
DISP2  PROC  FAR
       PUSH   CX
       PUSH   BX
       PUSH   AX
       MOV    CX,3
NEXTC: LODSB                    ;  AL←[SI]
       MOV   AH, 0EH            ; 写字符，并移动光
       MOV   BX, 01
       INT    10H
       CALL   DELAY
       LOOP   NEXTC
       POP    AX
       POP    BX
       POP    CX
       RET
DISP2  ENDP
```

以下是延时 1 秒子程序

```
DELAY   PROC
        PUSH   CX
```

```
        PUSH    DX
        MOV     DX，20
DL500：MOV     CX，2801
DL10ms：LOOP   DL10ms
        DEC     DX
        JNZ     DL500
        POP     DX
        POP     CX
        RET
DELAY   ENDP
```

四、实验设备

IBM-PC／XT 微机　　一台

五、实验预习要求

1. 正确理解实验目的、内容和原理。
2. 根据流程图在实验前编写好主程序和键盘中断服务程序。

六、实验报告要求

1. 整理出正确的主程序和中断服务程序清单，并加以注释。
2. 写出实验结果。
3. 分析在实验中所出现的问题。
4. 回答思考题。

七、思考题

键盘上某个键按下和释放时都会向 8259 发出中断请求，要求只在键按下时显示‘OK!’，键释放时不显示，则中断服务程序 KEYINTS 应如何修改?

实验十一　定时中断

一 、实验目的

1．熟悉 Intel 8086CPU 的中断处理功能以及 IBM-PC 的中断结构。

2．了解 8253 定时器的使用。

3．掌握定时中断的编程，观察中断响应的情况。

二、实验任务

利用定时器 8253，其中断类型号为 1CH，每隔 55ms 发一次中断请求信号，CPU 响应中断后转入执行 TIMERINTS 中断服务程序，并在显示器上显示某字符或某个图形(自定)。

三、实验原理

在主程序中应先保存原中断类型号 1CH 的中断服务程序入口地址，然后把自行编写的 TIMERINTS 定时中断服务程序入口地址的段内偏移地址和段地址存入以 1CH*4 为起始地址的四个连续单元内。

主程序中安排开中断指令，CPU 响应中断后自动转入 TIMERINTS 定时中断服务程序去执行。

主程序和定时中断服务程序流程图如图 1-11-1 和图 1-11-2 所示。

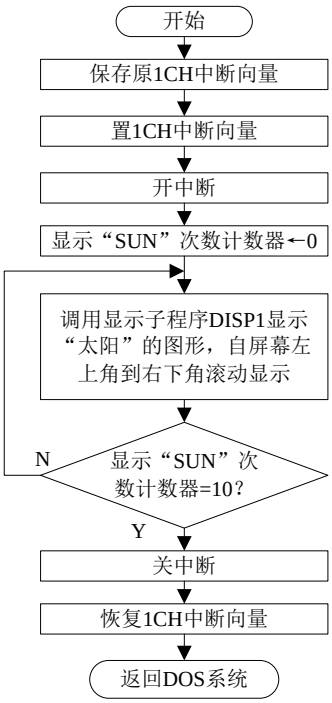


图 1-11-1

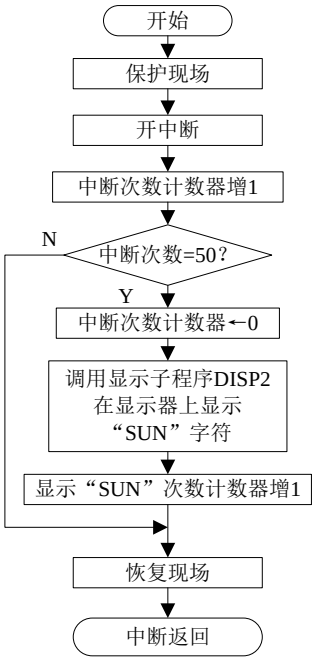


图 1-11-2

自屏幕左上角向右下角移动并显示“太阳”图形的显示子程序如

```
DISP1   PROC   FAR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     AH，15     ；读当前显示状态，
        INT     10H
        MOV     AH，0      ；设置显示方式
        INT     10H
        MOV     CX，1      ；显示的字符个数为 1
        MOV     DX，0      ；行号为 0，列号为 0
REPT：  MOV     AH，2       ；设置光标位置
        INT     10H
        MOV     AL，0FH     ；读出太阳图形
        MOV     AH，10      ；写字符
        INT     10H
        CALL    DELAY
        SUB     AL，AL
        MOV     AH，10       ；清除原图形
        INT     10H
        INC     DH
        ADD     DL，2
        CMP     DH，25
        JNE     REPT
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
DISP1    ENDP
```

在定时中断服务程序中显示字符的子程序 DISP2 程序如下：

```
DISP2   PROC    FAR
        PUSH    CX
        PUSH    BX
        PUSH    AX
        MOV     CX，10
NEXTC：  LODSB
```

```
        MOV    AH，0EH
        MOV    BX，01
        INT     10H
        CALL   DELAY
        LOOP   NEXTC
        POP    AX
        POP    BX
        POP    CX
        RET
DISP2   ENDP
```

延时子程序(延时大约 1 秒左右)如下：

```
DELAY  PROC  FAR
       PUSH   CX
       PUSH   DX
       MOV    DX，20
DL500：MOV    CX，2801
DL10ms：LOOP  DL10ms
        DEC    DX
        JNZ    DL500
        POP    DX
        POP    CX
        RET
DELAY   ENDP
```

四、实验设备

IBM-PC / XT 微机　　一台

五、实验预习要求

1．正确理解实验目的、内容和原理。

2，根据流程图在实验前编写好定时中断实验的主程序和中断服务程序。

六、实验报告要求

1．整理出运行正确的主程序和中断服务程序清单，并加上注释。

2．写出程序运行后的结果。

3．分析在实验中所出现的问题。

4．回答思考题。

七、思考题。

试将实验十键盘中断和本次实验组合形成两级中断。按键数次后（次数自定）禁止中断，返回 DOS 系统。程序如何编制?(上机通程序进行验证)

实验十二　演奏乐曲

一、实验目的

利用 IBM-PC 机上的发音装置产生声响与音调，编制音乐演奏程序。

二、实验任务

图 1-12-1 给出了乐曲“四季歌”的简谱，其中有四分音符、二分音符、和八分音符，现定全音符分配 4 秒时间，那么可以得到此乐曲对应的节拍时间数据。

根据图 1-12-2 程序流程图以及利用 SING 子程序（其作用是从数据段中取出每个音符的频率数据和节拍时间数据，然后调用 SOUND 子程序发出声音），编制演奏“SING OF FOUR SEASON”乐曲程序。

在程序的数据段中要定义频率数据（FREQ）和节拍时间数据（TIME），并以 0000H 作为频率数据结束标志。在代码段中，要求把频率数据首地址送 SI，节拍时间数据的首地址送 BP。

四季歌

1=C 4/4

. .. .. .　　. .. ..
3 32 12 17|6 6 6 -|4 43 21 24|3 - - -|
. .. .　. . .. .
4 43 2 24|3 31 6 1|7 3 21 71|6 - - -|

图 1-12-1

三、实验原理

IBM-PC 主机内装有一只喇叭，由定时器 8253(端口地址为 40H～43H)和并行接口芯片 8255(端口地址为 60H～63H)控制其发音。在 ROM BIOS 基本输入输出系统中有一个 BEEP 程序，这程序根据 BL 中给出的时间计数值控制 8253 定时器产生一个或几个 500ms 左右、频率为 896Hz 的脉冲信号，此信号经过 8255 芯片 B 口控制喇叭的接通与断开，使其发出长或短的声音。

因为 BEEP 程序只能产生 896Hz 的声音，而且声音的持续时间只能是 500ms 的倍数。然而演奏乐曲，则要求能产生任一音频的声音，并且持续时间要容易调整。(例如可以是 10ms 的倍数)因此要改造 BEEP 程序。

图 1-12-2

BEEP 程序是将计数值 533H 送给定时器 8253 的通道 2 而产生 896Hz 的声音，那么产生其它频率声音的计数值可用比例方法计算：

533HX896÷(给定频率)＝1234DCH÷(给定频率)

也可以直接用定时器时钟 1193180Hz 计算计数值：

1193180÷(给定频率)＝1234DCH÷(给定频率)

若给定频率在 DI 中，则可用下述程序段产生对应常数：

```
MOV    DX，12H
MOV    AX，34DCH
DIV     DI
```

为了不使除法产生溢出．限制 DI 中频率不小于 19Hz。

10ms 延时可调用延时子程序 DL10ms 实现。

因此要编写一段 SOUND 子程序，让它产生任何音频、持续时间是 10ms 倍数的声音。其频率范围是 19—65535Hz(由 DI 决定)，这个频率范围足够了，因为人耳的最高辨听频率是 20000Hz。这个子程序利用 BX 内容控制声音的持续时间，BX 从 1 变到 65535，对应时间是 0.01～655．35s，BX＝0 时对应 655．35s。

利用 SOUND 子程序就可以编写演奏乐曲程序。

在演奏乐曲程序中，需要定义两组数据，二组是频率数据，另一组是节拍时间数据。频率可由表 1-12-1 查得。节拍时间取决于速度和每个音符持续的节拍,例如在 4／4 中，每小节包括 4 拍，全音符持续为 4 拍，二分音符持续为 2 拍，8 分音符持续半拍等。

表 1-12-1

音名	c d e f g a b	c^1 d^1 e^1 f^1 g^1 a^1 b^1	c^2 d^2 e^2 f^2 g^1 a^2 b^2
唱名	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7
频率	131 147 165 175 196 220 247	262 294 330 349 392 440 494	524 588 660 698 784 880 988

演奏过程 SING 子程序如下

```
SING    PROC   NEAR
        PUSH   DI
        PUSH   SI
        PUSH   BP
        PUSH   BX
RETP:   MOV    DI，[S1]
        CMP    DI，0
        JE     END-SING
        MOV    BX，DS:[BP]
        CALL   SOUND          ；参数为 DI 和 BX
        ADD    SI，2
        ADD    BP，2
        JMP    REPT
```

```
END-SING：POP  BX
          POP     BP
          POP     SI
          POP     DI
          RET
SING      ENDP
```

声音过程 SOUND 子程序如下：

```
SOUND  PROC  NEAR
        PUSH   AX
        PUSH   BX          ；BX 节拍时间数据
        PUSH   CX
        PUSH   DX
        PUSH   DI           ；入口参数 DI 给定频率数据
        MOV    AL，0B6H     ；8253 初始化(选通道 2，产生方波信号)
        OUT    43H，AL      ；43H 端口是 8253 的命令寄存器
        MOV    DX，12H      ；计算时间常数
        MOV    AX，34DCH
        DIV    DI
        OUT    42H，AL      ；设置时间常数
        MOV    AL，AH
        OUT    42H，AL
        IN     AL，61H
        MOV    AH，AL
        OR     AL，3         ；开喇叭，8255I／O 端口 61H 的低两位置
        OUT    61H，AL
DELAY:  MOV    CX，2801H     ；延时
DL10ms:  LOOP  DL10ms
        DEC    BX
        JNZ    DEIAY
        MOV    AL，AH
        OUT    61H，AL       ；关喇叭
        POP    DI
        POP    DX
        POP    CX
        POP    BX
        POP    AX
        RET
SUND   ENDP
```

四、实验设备

IBM—PC / XT 微机　　一台

五、实验预习要求

1. 正确理解实验的目的、内容和原理。
2. 根据流程图在实验前编写好源程序。

六、实验报告要求

1. 整理出运行正确的源程序清单，并加上注释。
2. 写出实验结果。
3. 分析实验中所遇到的问题，并说明是如何解决的。
4，回答思考题。

七、思考题

1. 若要求乐曲中有休止符，程序应作何改进?
2，试编程序调用 SING 和 SOUND 过程，演奏一曲你所喜欢的乐曲。

第二部分　TPC-2003通用微机接口实验

一、概述

八十年代以来，国内大中专院校很多专业都相继开设了“微机原理及应用”方面的课程，讲授内容主要是8位机（Z80），实验设备多采用TP801单板机。随着计算机技术的发展，讲授内容开始逐步转向16位或32位的PC系列微机，实验设备亦需更新，“TPC-2005通用32位微机接口(PCI)实验系统”是我公司继“TPC-2003通用32位微机接口实验系统”的基础上，综合了各学校讲课及实验老师的意见之后推出的微机硬件实验教学设备的新产品。该仪器增加了实验系统的开放能力和灵活性。它不仅使一些典型的微机接口实验方便，而且对一些计算机硬件要求较高的专业提供了锻炼学生动手能力，发挥创造才能的平台。该系统主要有以下特点：

★ 实验电路连接采用了国家专利、获奖产品“自锁紧”插座及导线，消除了连线接触不良的现象。

★ 电路设计中增加了多项保护措施,可有效的避免由于学生实验时错接、错编程损坏主机或接口集成电路的现象。

★ 接口实验增加了实用性、趣味性的项目，使用C语言进行实验的程序。

★ 实验台上增加了逻辑笔、通用IC插座等电路。可作为数字电路实验仪器使用,也可以用于学生毕业设计、实验数据的采集及科研开发。仪器硬件包括接口卡、实验台（箱）两部分组成，两者之间通过50线扁平电缆相连。接口卡可以插入PC系列微机中任意一个PCI扩展插槽，它的主要功能是将与实验有关的总线信号加以驱动后引到实验台上，同时引出信号还有与“中断”和“DMA”实验操作有关的信号及+5V、+12V、-12V电源。实验台上设有I/O地址译码电路、总线插孔、接口实验常用集成电路、外围电路及通用IC插座等部分组成。外围电路包括逻辑电平开关电路、LED显示电路、时钟电路、单脉冲电路、逻辑笔、复位电路、七段数码管显示电路、基本门电路、继电器及步进电机、小直流电机的驱动电路。

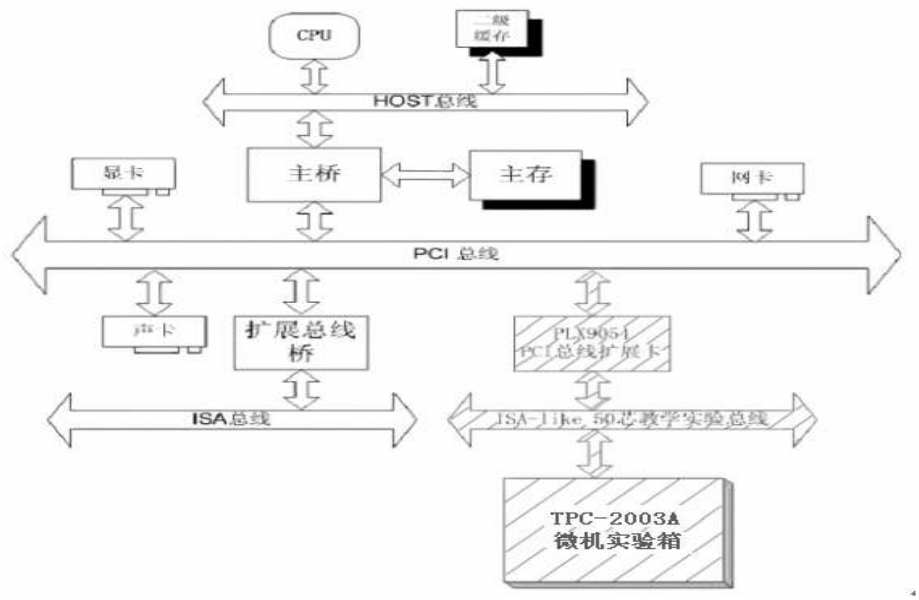
使用说明书中介绍了四十多个微机接口实验。覆盖了大中专院校微机接口实验教学大纲中的内容。教师可以根据课时计划安排选作,也可以在此基础上重新设计新的实验项目。

二、安装

1、实验装置基本组成：

（1）、硬件：PCI接口卡一块；实验台（箱）一个；50线扁平电缆一根；自锁紧导线50根；集成电路芯片（8251、74LS273、74LS244、6116）共4片。

该实验装置在PC系统中的位置如图所示（斜线标出）：



2、安装步骤：

（1）关上PC机电源，打开微机主机箱。

（2）将PC总线接口卡插在任意扩展槽中。接口卡的结构如图1。

（3）用50芯线扁平电缆线连接接口卡和实验台。

（4）接上PC机电源，启动微机，Windows提示发现新硬件，请安装光盘“Disk驱动”目录里的安装配置文件（TPC.inf）

（5）重新启动计算机后，运行光盘里的Setup程序安装运行支持库文件。

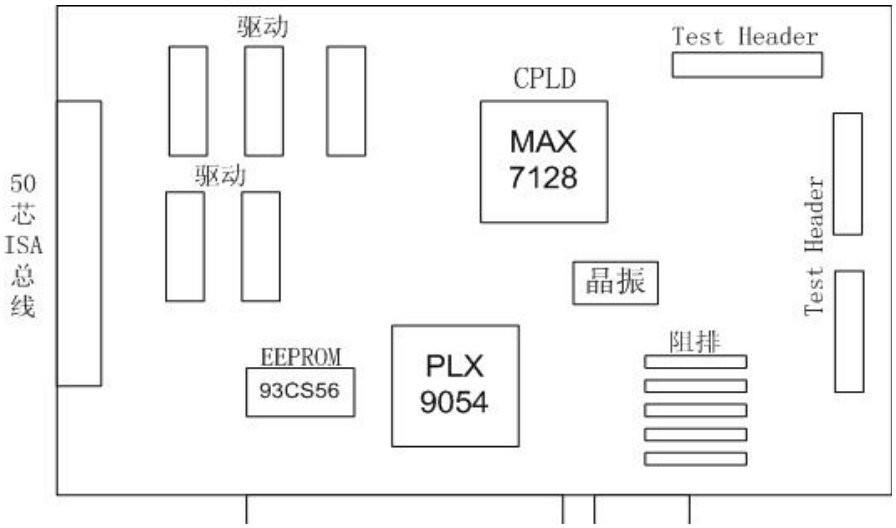
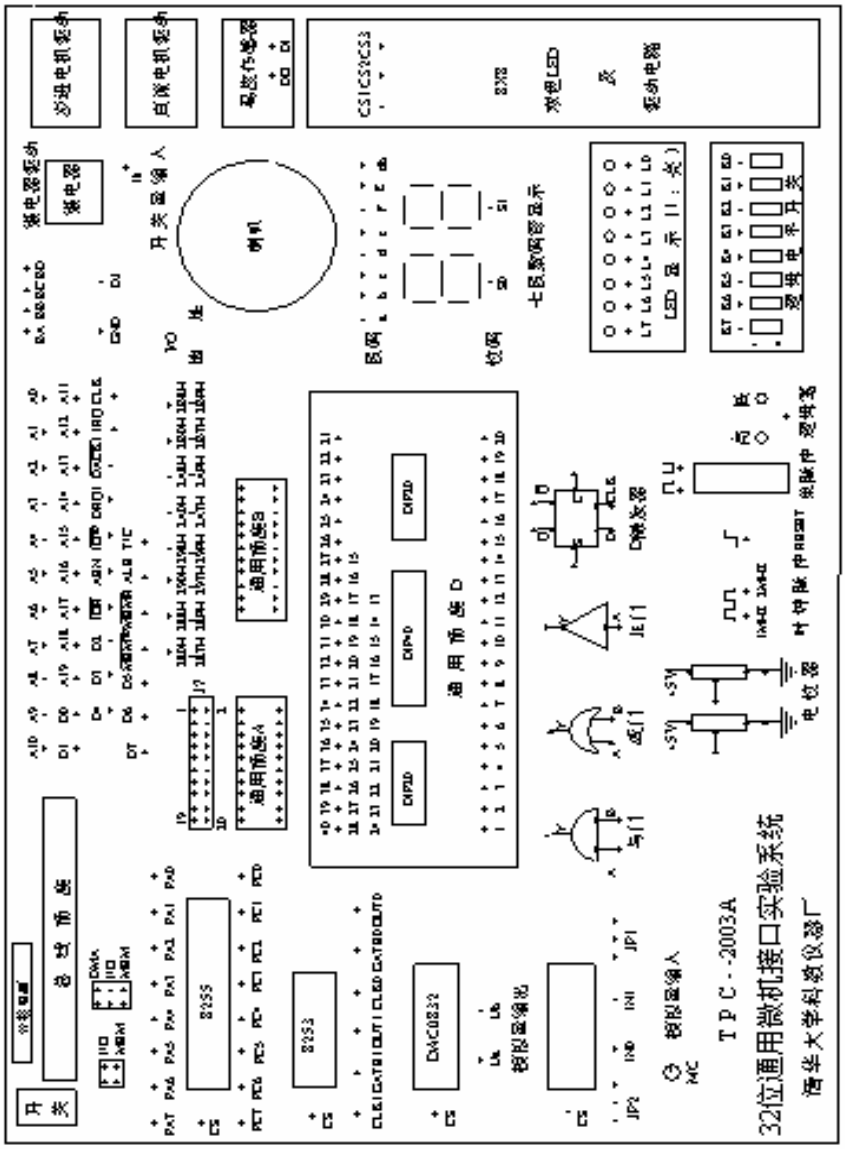


图 1

PLX9054PCI总线扩展卡结构，该卡使用PLX9054和CPLD把PCI总线时序转换成50芯ISA总线时序，提供给微机实验台使用。

三、实验台结构：



实验台上有微机原理硬件实验最常用接口电路芯片，包括：可编程定时器/计数器（ 8253 ） 、可编程并行接口（ 8255 ） 、数/ 模转换器（ DAC0832 ） 、模/ 数转换器（ADC0809），这里芯片与CPU相连的引线除片选信号CS外都已连好，与外界连接的关键引脚在芯片周围用“自锁紧”插座引出，供学生实验时使用。其中数/模转换器附有双极性输出，模/数转换器附有双极性输入插座。具体电路可见下面实验说明。

实验一　I/O地址译码

一、实验目的

掌握I/O地址译码电路的工作原理。

二、实验原理和内容

实验电路如图2-1所示，其中74LS74为D触发器，可直接使用实验台上数字电路实验区的D触发器,74LS138为地址译码器。译码输出端Y0～Y7在实验台上“I/O地址“输出端引出，每个输出端包含8个地址，Y0：280H～287H，Y1：288H～28FH，…… 当CPU执行I/ O指令且地址在280H～2BFH范围内，译码器选中，必有一根译码线输出负脉冲。

例如：执行下面两条指令

MOV　DX，2A0H

OUT　DX，AL（或IN AL，DX）

Y4输出一个负脉冲，执行下面两条指令

MOV　DX，2A8H

OUT　DX，AL（或IN AL，DX）

Y5 输出一个负脉冲。

注意：命令中的端口地址D820、D82A 是根据PCI卡的基址再加上偏移量计算出来的，不同的微机器PCI卡的基址可能不同，需要事先查找出来，查找方法见本书末尾文章中的介绍。计算公式如下：

计算出的地址= 查找出的PCI卡的基址+ 偏移量；

（其中：偏移量= 2A0H － 280H 或 2A8H －A80H）

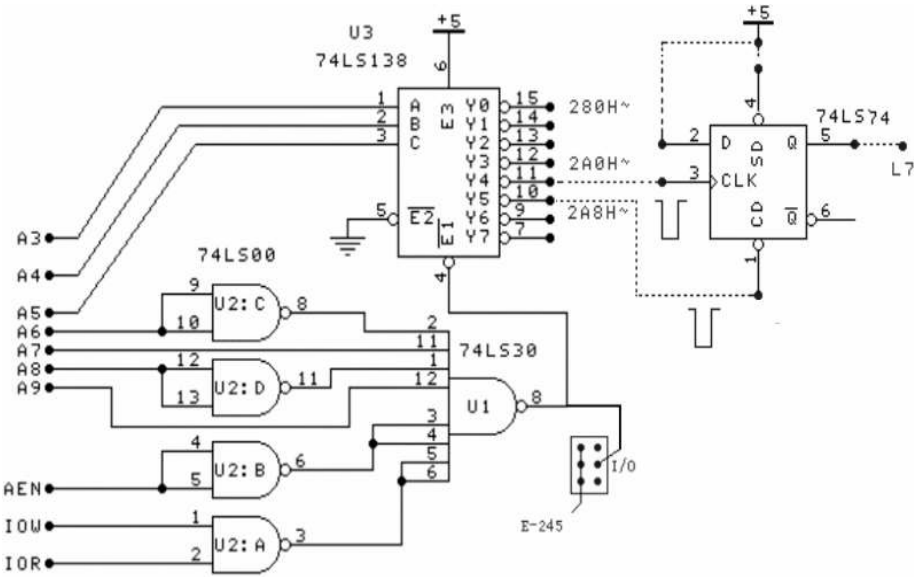


图 2-1

利用这个负脉冲控制L7闪烁发光（亮、灭、亮、灭、……），时间间隔通过软件延时实现。

三、编程提示

1、实验电路中D触发器CLK端输入脉冲时，上升沿使Q端输出高电平L7发光，CD端加低电平L7灭。

2、由于TPC卡使用PCI总线，所以分配的IO地址每台微机可能都不同，编程时需要了解当前的微机使用那段IO地址并进行设置，获取方法请参看汇编程序使用方法的介绍。（也可使用自动获取资源分配的程序取得中断号）

3、参考程序：程序名:YMQ.ASM

```
ioport   equ 0d400h-0280h
outport1 equ ioport+2a0h
outport2 equ ioport+2a8h
code segment
         assume cs:code
start: mov dx,outport1
         out dx,al
         call delay                    ; 调延时子程序
         mov  dx,outport2
         out  dx,al
         call delay                     ; 调延时子程序
         mov  ah,1
         int  16h
         je   start
         mov  ah,4ch
         int  21h
delay proc near                        ; 延时子程序
      mov  bx,2000
lll:  mov  cx,0
ll:   loop ll
      dec  bx
      jne  lll
      ret
delay endp
code  ends
end   start
```

实验二　简单并行接口

一、实验目的

掌握简单并行接口的工作原理及使用方法。

二、实验内容

1、按下面图2-2简单并行输出接口电路图连接线路（74LS273插通用插座，74LS32用实验台上的“或门”）。74LS273为八D触发器，8个D输入端分别接数据总线D0～D7，8个Q输出端接LED显示电路L0～L7。

2、编程从键盘输入一个字符或数字，将其ASCⅡ码通过这个输出接口输出，根据8个发光二极管发光情况验证正确性。

3、按下面图2-3简单并行输入接口电路图连接电路（74LS244插通用插座，74LS32用实验台上的“或门”）。74LS244为八缓冲器，8个数据输入端分别接逻辑电平开关输出K0～K7，8个数据输出端分别接数据总线D0～D7。

4、用逻辑电平开关预置某个字母的ASCⅡ码，编程输入这个ASCⅡ码，并将其对应字母在屏幕上显示出来。

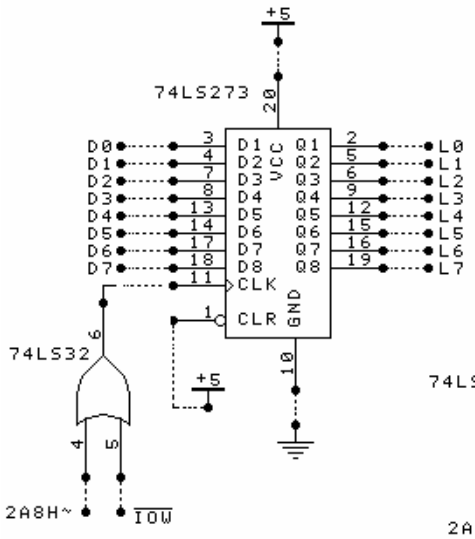


图 2-2

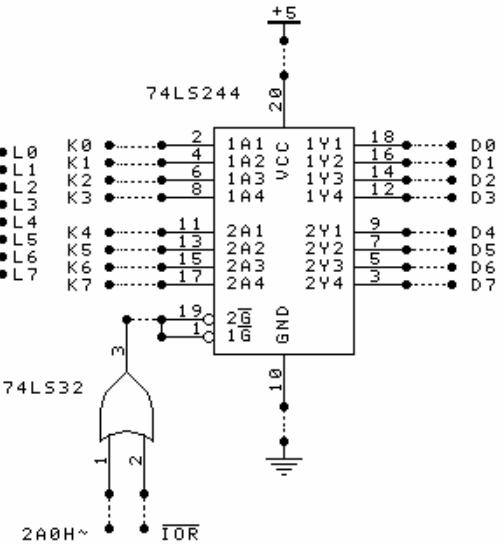


图 2-3

三、编程提示

1、上述并行输出接口的地址为2A8H，并行输入接口的地址为2A0H，通过上述并行接口电路输出数据需要3条指令：

```
MOV   AL,数据
MOV   DX,2A8H
OUT   DX,AL
```

通过上述并行接口输入数据需要2条指令：

```
MOV   DX,2ADH
IN    AL,DX
```

2、参考流程图

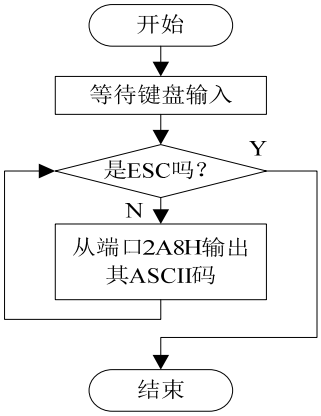


图 2-4 参考程序1

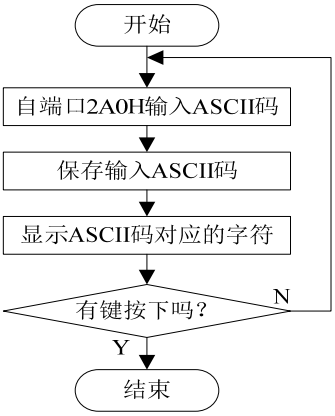


图 2-5 参考程序2

3、参考程序1：程序名:E273.ASM

```
ioport equ 0d400h-0280h
ls273 equ ioport+2a8h
code segment
         assume cs:code
start:   mov ah,2            ; 回车符
         mov dl,0dh
         int 21h
         mov ah,1            ; 等待键盘输入
         int 21h
         cmp al,27           ; 判断是否为ESC键
         je exit             ; 若是则退出
         mov dx,ls273        ; 若不是,从2A8H输出其ASCII码
         out dx,al
         jmp start           ; 转start
exit:    mov ah,4ch          ; 返回DOS
         int 21h
code ends
end start
```

4、参考程序2：程序名:E244.ASM

```
ioport equ 0d400h-0280h
ls244 equ ioport+2a0h
code segment
         assume cs:code
start:   mov dx,ls244      ; 从2A0输入一数据
         in al,dx
         mov dl,al         ; 将所读数据保存在DL中
         mov ah,02
         int 21h
         mov dl,0dh        ; 显示回车符
         int 21h
         mov dl,0ah        ; 显示换行符
         int 21h
         mov ah,06         ; 是否有键按下
         mov dl,0ffh
         int 21h
         jnz exit
         je start          ; 若无,则转start
exit:     mov ah,4ch       ; 返回DOS
         int 21h
code ends
end start
```

实验三　可编程定时器／计数器（8253）

一、实验目的

掌握8253的基本工作原理和编程方法。

二、实验内容

1. 按图2-6虚线连接电路，将计数器0设置为方式2，计数器初值为N（N≤0FH），用手动逐个输入单脉冲，编程使计数值在屏幕上显示，并同时用逻辑笔观察OUT0电平变化（当输入N+1个脉冲后OUT0变高电平）。

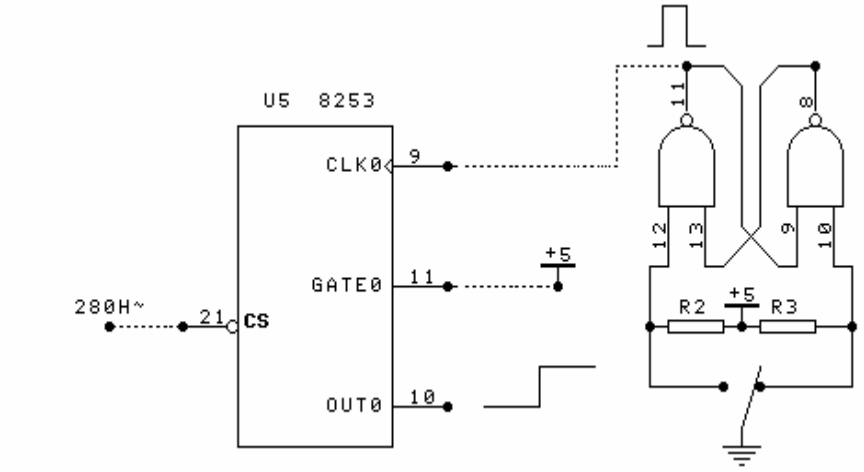


图 2-6

2. 按图2-7连接电路，将计数器0、计数器1分别设置为方式3，计数初值设为1000，用逻辑笔观察OUT1输出电平的变化（频率1HZ）。

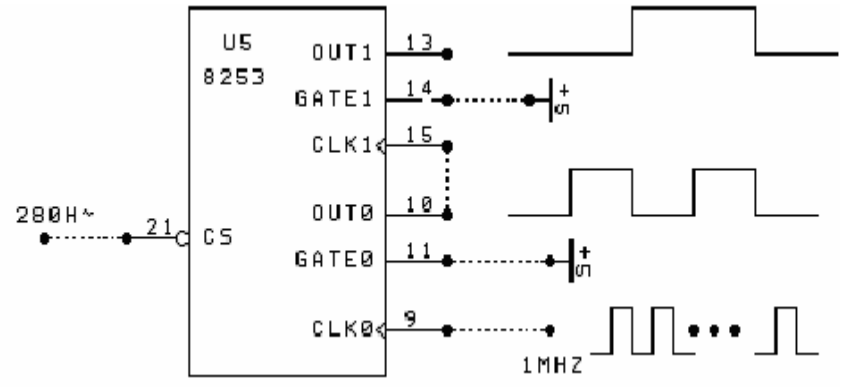


图 2-7

三、编程提示

1、 8253控制寄存器地址　　283H

计数器0地址　　280H

计数器1地址　　281H

CLK0 连接时钟　　1MHZ

2、参考流程图（见图 2-8、图 2-9）

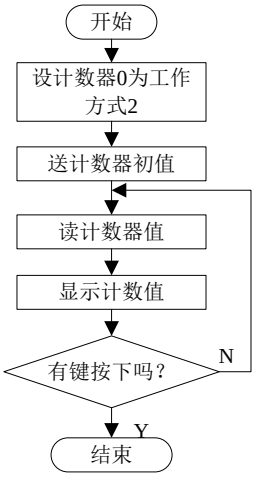


图 2-8

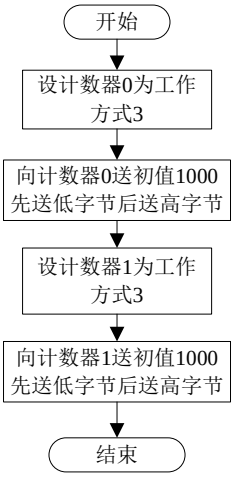


图 2-9

3、参考程序1：　（程序名：E8253_1.ASM）

```
ioport equ 0d400h-0280h
io8253a equ ioport+283h
io8253b equ ioport+280h
code segment
      assume cs:code
start: mov al,14h              ；设置8253通道0为工作方式2,二进制计数
       mov dx,io8253a
       out dx,al
       mov dx,io8253b          ；送计数初值为0FH
       mov al,0fh
       out dx,al
lll:   in al,dx                ；读计数初值
       call disp               ；调显示子程序
       push dx
       mov ah,06h
       mov dl,0ffh
```

```
     int 21h
     pop dx
     jz lll
     mov ah,4ch              ; 退出
     int 21h
disp proc near               ; 显示子程序
     push dx
     and al,0fh              ; 首先取低四位
     mov dl,al
     cmp dl,9                ; 判断是否<=9
     jle num                 ; 若是则为'0'-'9',ASCII码加30H
     add dl,7                ; 否则为'A'-'F',ASCII码加37H
num: add dl,30h
     mov ah,02h              ; 显示
     int 21h
     mov dl,0dh              ; 加回车符
     int 21h
     mov dl,0ah              ; 加换行符
     int 21h
     pop dx
     ret                     ; 子程序返回
disp endp
code ends
      end start
```

4、参考程序2：（程序名：E8253_2.ASM）

```
ioport equ 0d400h-0280h
io8253a equ ioport+280h
io8253b equ ioport+281h
io8253c equ ioport+283h
code segment
      assume cs:code
start:mov dx,io8253c          ; 向8253写控制字
      mov al,36h              ; 使0通道为工作方式3
      out dx,al
      mov ax,1000             ; 写入循环计数初值1000
      mov dx,io8253a
      out dx,al               ; 先写入低字节
```

```
    mov al,ah
    out dx,al                ; 后写入高字节
    mov dx,io8253c
    mov al,76h               ; 设8253通道1工作方式3
    out dx,al
    mov ax,1000              ; 写入循环计数初值1000
    mov dx,io8253b
    out dx,al                ; 先写低字节
    mov al,ah
    out dx,al                ; 后写高字节
    mov ah,4ch               ; 程序退出
    int 21h
code ends
    end start
```

实验四　可编程并行接口（一）（8255方式0）

一、实验目的

掌握8255方式0的工作原理及使用方法。

二、实验内容

1．实验电路如图2-10，8255C口接逻辑电平开关K0～K7，A口接LED显示电路L0～L7。

2．编程从 8255C 口输入数据，再从 A 口输出。

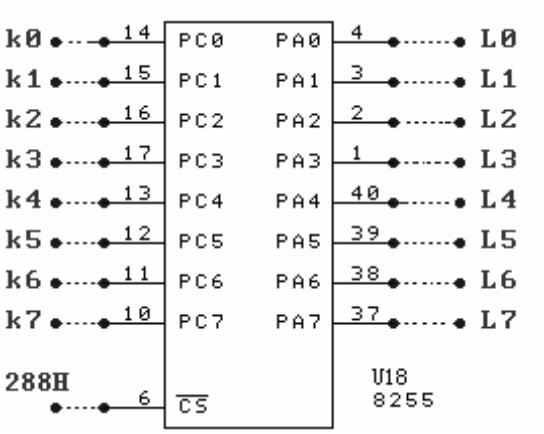


图 2-10

三、编程提示

1、8255控制寄存器端口地址28BH

A口的地址288H

C口的地址28AH

2、参考流程图（见图 21）：

3、参考程序： E8255.ASM

```
ioport equ 0d400h-0280h
io8255a equ ioport+288h
io8255b equ ioport+28bh
io8255c equ ioport+28ah
code segment
     assume cs:code
start:   mov dx,io8255b            ；设8255为C口输入,A口输出
         mov al,8bh
         out dx,al
         inout: mov dx,io8255c     ；从C口输入一数据
```

```
        in al,dx
        mov dx,io8255a          ; 从A口输出刚才自C口
        out dx,al               ; 所输入的数据
        mov dl,0ffh             ; 判断是否有按键
        mov ah,06h
        int 21h
        jz inout                ; 若无,则继续自C口输入,A口输出
        mov ah,4ch              ; 否则返回DOS
        int 21h
code ends
    end start
```

实验五　七段数码管

一、实验目的

掌握数码管显示数字的原理

二、实验内容

1. 静态显示：按图2-11连接好电路，将8255的A口PA0～PA6分别与七段数码管的段码驱动输入端a～ g 相连，位码驱动输入端S1接+5V（选中），S0、dp接地（关闭）。编程从键盘输入一位十进制数字（0～9），在七段数码管上显示出来。

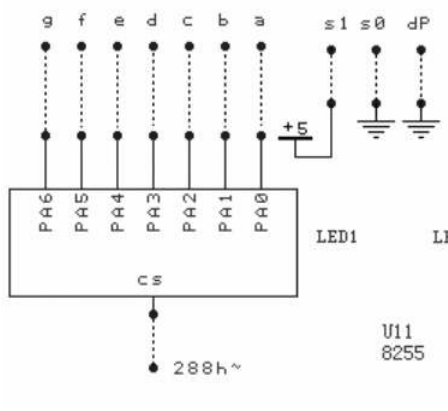


图2-11

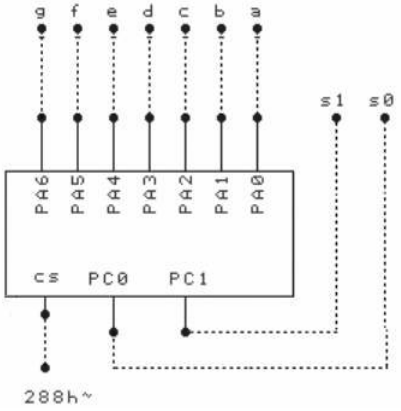


图2-12

2. 动态显示：按图2-12连接好电路，七段数码管段码连接不变，位码驱动输入端S1，S0接8255 C口的PC1，PC0。编程在两个数码管上显示“56”。

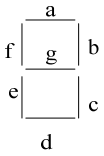
3. 动态显示（选作）：使用图 23 的电路，编程在两个数码管上循环显示“00-99”。

三、编程提示

1、实验台上的七段数码管为共阴型，段码采用同相驱动，输入端加高电平,选中的数码管亮，位码加反相驱动器，位码输入端高电平选中。

2、七段数码管的字型代码表如下表：

显示字形	g	e	f	d	c	b	a	段码
0	0	1	1	1	1	1	1	3fh
1	0	0	0	0	1	1	0	06h
2	1	0	1	1	0	1	1	5bh
3	1	0	0	1	1	1	1	4fh
4	1	1	0	0	1	1	0	66h
5	1	1	0	1	1	0	1	6dh
6	1	1	1	1	1	0	1	7dh
7	0	0	0	0	1	1	1	07h
8	1	1	1	1	1	1	1	7fh
9	1	1	0	1	1	1	1	6fh



3、参考流程图(见图 2-13)

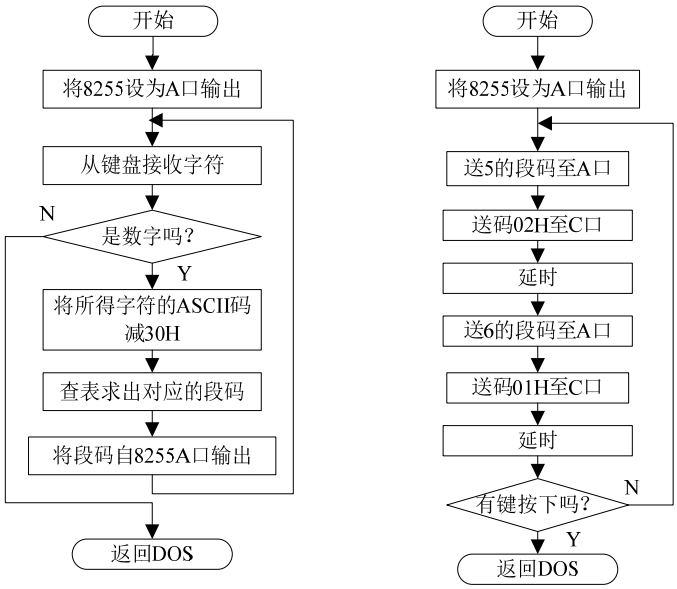


图 2-13

4、参考程序1： （程序名：LED1.ASM）

```
data segment
ioport equ 0d400h-0280h
io8255a equ ioport+288h
io8255b equ ioport+28bh
led db 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh
mesg1 db 0dh,0ah,'Input a num (0--9h):',0dh,0ah,'$'
data ends
     code segment
     assume cs:code,ds:data
start:    mov ax,data
          mov ds,ax
          mov dx,io8255b        ；使8255的A口为输出方式
          mov ax,80h
          out dx,al
sss:      mov dx,offset mesg1   ；显示提示信息
          mov ah,09h
          int 21h
```

```
        mov ah,01               ; 从键盘接收字符
        int 21h
        cmp al,'0'              ; 是否小于0
        jl exit                 ; 若是则退出
        cmp al,'9'              ; 是否大于9
        jg exit                 ; 若是则退出
        sub al,30h              ; 将所得字符的ASCII码减30H
        mov bx,offset led       ; bx为数码表的起始地址
        xlat                    ; 求出相应的段码
        mov dx,io8255a          ; 从8255的A口输出
        out dx,al
        jmp sss                 ; 转SSS
exit:   mov ah,4ch              ; 返回DOS
        int 21h
code ends
       end start
```

5、参考程序2：（程序名：LED2.ASM）

```
data segment
ioport equ 0d400h-0280h
io8255a equ ioport+28ah
io8255b equ ioport+28bh
io8255c equ ioport+288h
led db 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh ;段码
buffer1 db 5,6          ; 存放要显示的个位和十位
bz dw ?                 ; 位码
data ends
code segment
        assume cs:code,ds:data
start:  mov ax,data
        mov ds,ax
        mov dx,io8255b          ; 将8255设为A口输出
        mov al,80h
        out dx,al
        mov di,offset buffer1   ; 设di为显示缓冲区
loop2:  mov bh,02
lll:  mov byte ptr bz,bh
        push di
```

```
        dec di
        add di, bz
        mov bl,[di]             ; bl为要显示的数
        pop di
        mov bh,0
        mov si,offset led       ; 置led数码表偏移地址为SI
        add si,bx               ; 求出对应的led数码
        mov al,byte ptr [si]
        mov dx,io8255c          ; 自8255A的口输出
        out dx,al
        mov al,byte ptr bz      ; 使相应的数码管亮
        mov dx,io8255a
        out dx,al
        mov cx,3000
delay:  loop delay              ; 延时
        mov bh,byte ptr bz
        shr bh,1
        jnz lll
        mov dx,0ffh
        mov ah,06
        int 21h
        je loop2                ; 有键按下则退出
        mov dx,io8255a
        mov al,0                ; 关掉数码管显示
        out dx,al
        mov ah,4ch              ; 返回
        int 21h
code ends
end start
```

6、参考程序3：（程序名：LED3.ASM）

```
data segment
ioport equ 0d400h-0280h
io8255a equ ioport+28ah
io8255b equ ioport+28bh
io8255c equ ioport+288h
```

```
led db 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh ;段码
buffer1 db 0,0              ; 存放要显示的十位和个位
bz dw ?                     ; 位码
data ends
code segment
        assume cs:code,ds:data
start:  mov ax,data
        mov ds,ax
        mov dx,io8255b              ; 将8255设为A口输出
        mov al,80h
        out dx,al
        mov di,offset buffer1       ; 设di为显示缓冲区
loop1:  mov cx,0300h                ; 循环次数
loop2:  mov bh,02
lll:    mov byte ptr bz,bh
        push di
        dec di
        add di, bz
        mov bl,[di]                 ; bl为要显示的数
        pop di
        mov bh,0
        mov si,offset led           ; 置led数码表偏移地址为SI
        add si,bx                   ; 求出对应的led数码
        mov al,byte ptr [si]
        mov dx,io8255c              ; 自8255A的口输出
        out dx,al
        mov al,byte ptr bz          ; 使相应的数码管亮
        mov dx,io8255a
        out dx,al
        push cx
        mov cx,3000                 ; 如果显示过快，可更改cx值为最大0ffffh
delay:  loop delay                  ; 延时
        pop cx
        mov bh,byte ptr bz
        shr bh,1
        jnz lll
        loop loop2                  ; 循环延时
        mov ax,word ptr [di]
```

```
        cmp ah,09
        jnz set
        cmp al,09
        jnz set
        mov ax,0000
        mov [di],al
        mov [di+1],ah
        jmp loop1
set:  mov ah,01
        int 16h
        jne exit                    ; 有键按下则转exit
        mov ax,word ptr [di]
        inc al
        aaa
        mov [di],al                 ; al为十位
        mov [di+1],ah               ; ah中为个位
        jmp loop1
exit:     mov dx,io8255a
        mov al,0                    ; 关掉数码管显示
        out dx,al
        mov ah,4ch                  ; 返回DOS
        int 21h
code ends
        end start
```

实验六　交通灯控制实验

一. 实验目的

通过并行接口8255实现十字路口交通灯的模拟控制,进一步掌握对并行口的使用。

二. 实验内容

如图2-14，L7、L6、L5作为南北路口的交通灯与PC7、PC6、PC5相连，L2、L1、L0作为东西路口的交通灯与PC2、PC1、PC0相连。编程使六个灯按交通灯变化规律亮灭。

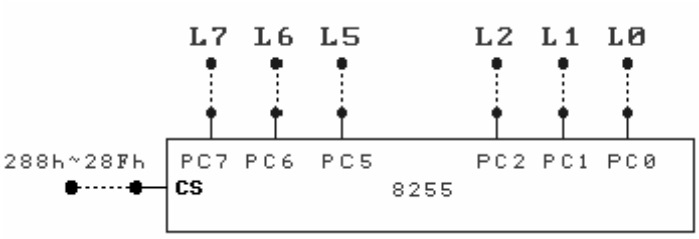


图2-14

三. 编程提示:

十字路口交通灯的变化规律要求：

（1） 南北路口的绿灯、东西路口的红灯同时亮30秒左右。

（2） 南北路口的黄灯闪烁若干次，同时东西路口的红灯继续亮。

（3） 南北路口的红灯、东西路口的绿灯同时亮30秒左右。

（4） 南北路口的红灯继续亮、同时东西路口的黄灯亮闪烁若干次。

（5） 转（1）重复。

四、参考流程图

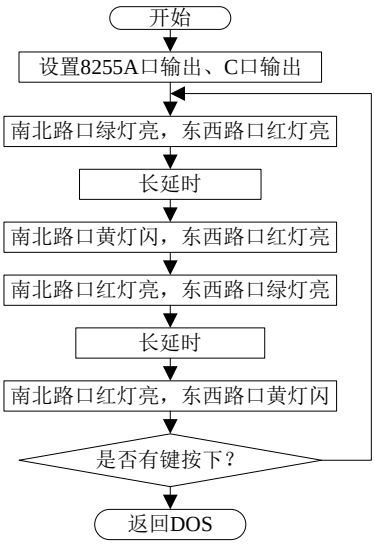


图2-15

五、程序框架：JTD.ASM

```
; ************************************
; * 十字路口红绿灯模拟演示程序       *
; * 端口各灯的设置                   *
; * 1红1黄1绿     2红2黄2绿          *
; ************************************
data segment
ioport equ 0d400h-0280h
io8255a equ ioport+28ah
io8255b equ ioport+28bh
data ends
code segment
         assume cs:code,ds:data
start:   mov ax,data
         mov ds,ax
         .
         .
         .
         .
code ends
end start
```

实验七　中　断

一、实验目的

１、掌握PC机中断处理系统的基本原理。

２、学会编写中断服务程序。

二、实验原理与内容

1、实验原理

PC机用户可使用的硬件中断只有可屏蔽中断，由8259中断控制器管理。中断控制器用于接收外部的中断请求信号，经过优先级判别等处理后向CPU发出可屏蔽中断请求。IBMPC、PC/XT机内有一片8259中断控制器对外可以提供8个中断源：

中断源	中断类型号	中断功能
IRQ0	08H	时钟
IRQ1	09H	键盘
IRQ2	0AH	保留
IRQ3	0BH	串行口2
IRQ4	0CH	串行口1
IRQ5	0DH	硬盘
IRQ6	0EH	软盘
IRQ7	0FH	并行打印机

8个中断源的中断请求信号线IRQ0～IRQ7在主机的62线ISA总线插座中可以引出，系统已设定中断请求信号为“边沿触发”，普通结束方式。对于PC/AT及286以上微机内又扩展了一片8259中断控制，IRQ2用于两片8259之间级连，对外可以提供16个中断源：

中断源	中断类型号	中断功能
IRQ8	070H	实时时钟
IRQ9	071H	用户中断
IRQ10	072H	保留
IRQ11	073H	保留
IRQ12	074H	保留
IRQ13	075H	协处理器
IRQ14	076H	硬盘
IRQ15	077H	保留

PCI总线中的中断线只有四根，INTA#、INTB#、INTC#、INTD#，它们需要通过P&P的设置来和某一根中断相连接才能进行中断申请。

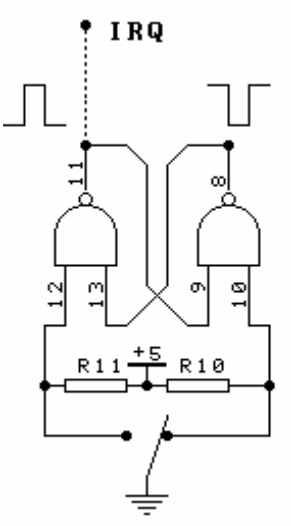


图 2-16 中断电路

2、实验内容

实验电路如图31，直接用手动产单脉冲作为中断请求信号（只需连接一根导线）。要求每按一次开关产生一次中断，在屏幕上显示一次“TPC pci card Interrupt”，中断10次后程序退出。

三、编程提示

1. 由于9054的驱动程序影响直写9054芯片的控制寄存器，中断实验需要在纯DOS的环境中才能正常运行。这里指的纯DOS环境是指微机启动时按F8键进入的DOS环境。WINDOWS重启进入MSDOS方式由于系统资源被重新规划过，所以也不能正常实验。

2. 由于TPC卡使用PCI总线，所以分配的中断号每台微机可能都不同，编程时需要了解当前的微机使用那个中断号并进行设置，获取方法请参看汇编程序使用方法的介绍。（也可使用自动获取资源分配的程序取得中断号）

3. 在纯DOS环境下，有些微机的BIOS设置中有将资源保留给ISA总线使用的选项，致使在纯DOS环境（WINDOWS环境下不会出现此问题）下PCI总线无法获得系统资源，也就无法做实验，这时需要将此选项修改为使用即插即用。

4. 在纯DOS环境下，有些微机的BIOS设置中有使用即插即用操作系统的选项，如果在使用即插即用操作系统状态下，BIOS将不会给TPC卡分配系统资源，致使在纯DOS环境（WINDOWS环境下不会出现此问题）下PCI总线无法获得系统资源，也就无法做实验，这时需要将此选项修改为不使用即插即用操作系统。

5. 由于TPC卡使用9054芯片连接微机，所以在编程使用微机中断前需要使能9054的中断功能，代码如下：

```
mov dx,ioport_cent+68h  ；设置tpc 卡中9054芯片io口,使能中断
in ax,dx
or ax,0900h
```

```
out dx,ax
```

其中IOPORT_CENT是9054芯片寄存器组的I/O起始地址，每台微机可能都不同，编程时需要了解当前的微机使用哪段并进行设置，获取方法请参看本书结尾部分的介绍。（也可使用自动获取资源分配的程序取得），+68H的偏移是关于中断使能的寄存器地址，设置含义如下：

设置INTCSR(68H)寄存器，中断模式设置

BITS	
8	1：能够产生PCI中断 0：禁止产生PCI中断
11	1：能够LOCAL端输入的中断送到PCI端 0：禁止LOCAL端输入的中断送到PCI端
其它位为零即可，更多内容参看9054芯片数据手册	

程序退出前还要关闭9054的中断，代码如下：

```
mov dx,ioport_cent+68h          ；设置tpc 卡中9054芯片io口,关闭中断
in ax,dx
and ax,0f7ffh
out dx,ax
```

6. PC机中断控制器8259 的地址为20H、21H,编程时要根据中断类型号设置中断矢量，8259中断屏蔽寄存器IMR对应位要清零（允许中断），中断服务结束返回前要使用中断结束命令：

```
MOV AL，20H
OUT 20H，AL
```

中断结束返回DOS时应将IMR对应位置1，以关闭中断。

四、参考流程图

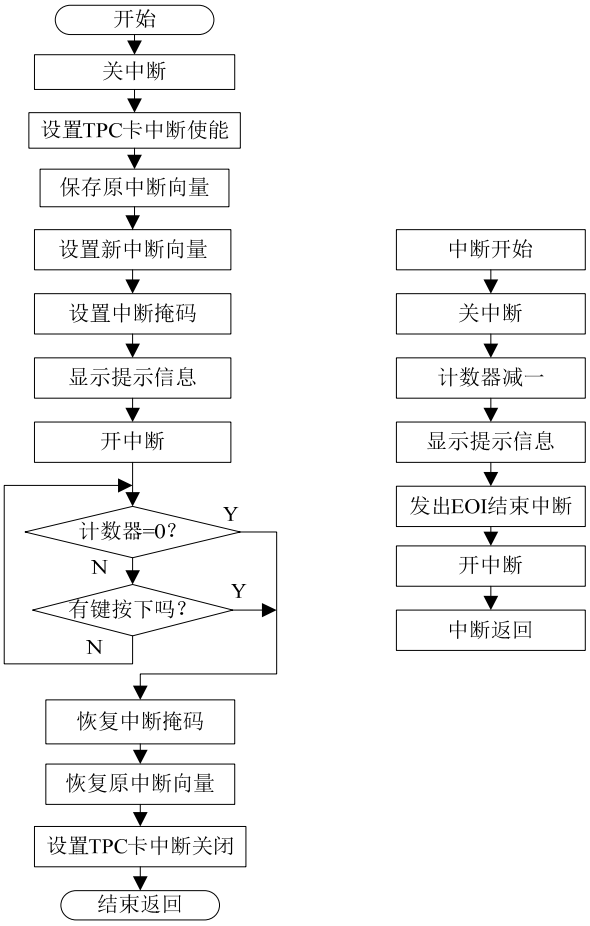


图 2-17 中断

五、参考程序程序名：INT.ASM

```
; 386以上微机适用
; 纯dos下才能使用
; tasm4.1或以上编译
data segment
int_vect EQU 071H          ; 中断0-7的向量为:08h-0fh,中断8-15的向量为:70h-77h
irq_mask_2_7 equ 011111011b ; 中断掩码,中断0-7时从低至高相应位为零,中断8-15
                              时第2位为零
irq_mask_9_15 equ 011111101b ; 中断0-7时全一,中断8-15时从低至高相应位为零
ioport_cent equ 0d800h       ; tpc 卡中9054芯片的io地址
csreg dw ?
ipreg dw ?                   ; 旧中断向量保存空间
irq_times dw 00h             ; 中断计数
```

```
msg1 db 0dh,0ah,'TPC pci card Interrupt',0dh,0ah,'$'
msg2 db 0dh,0ah,'Press any key to exit!',0dh,0ah,'$'
msg3 db 0dh,0ah,'Press DMC to interrupt 10 times and exit!',0dh,0ah,'$'
data ends
stacks segment
db 100 dup (?)
stacks ends
code segment
        assume cs:code,ds:data,ss:stacks,es:data
start:
; Enable Local Interrupt Input

        cli
        mov ax,data
        mov ds,ax
        mov es,ax
        mov ax,stacks
        mov ss,ax

        mov dx,ioport_cent+68h  ; 设置tpc 卡中9054芯片io口,使能中断
        in ax,dx
        or ax,0900h
        out dx,ax

        mov ah,35h
        mov al,int_vect         ; 保存原中断向量
        int 21h
        mov ax,es
        mov csreg,ax
        mov ipreg,bx

        mov ax,cs               ; 设置新中断向量
        mov ds,ax
        mov dx,offset int_proc
        mov al,int_vect
        mov ah,25h
        int 21h
```

```
        in al, 21h              ; 设置中断掩码
        and al, irq_mask_2_7
        out 21h, al
        in al, 0a1h
        and al, irq_mask_9_15
        out 0a1h, al

        mov ax,data
        mov ds,ax
        mov dx,offset msg2
        mov ah,09h
        int 21h
        mov dx,offset msg3
        mov ah,09h
        int 21h
        mov irq_times,0ah
        sti

loop1:  cmp irq_times,0         ; 等待中断并判断中断10次后退出
        jz exit
        mov ah,1
        int 16h
        jnz exit                ; 按任意键退出
        jmp loop1

exit:   cli
        mov bl, irq_mask_2_7    ; 恢复中断掩码
        not bl
        in al, 21h
        or al, bl
        out 21h, al
        mov bl, irq_mask_9_15
        not bl
        in al, 0a1h
        or al, bl
        out 0a1h, al

        mov dx,ipreg            ; 恢复原中断向量
```

```
        mov ax,csreg
        mov ds,ax
        mov ah,25h
        mov al,int_vect
        int 21h
        mov dx,ioport_cent+68h    ; 设置tpc 卡中9054芯片io口,关闭中断
        in ax,dx
        and ax,0f7ffh
        out dx,ax

        mov ax,4c00h
        int 21h

        int_proc proc far         ; 中断处理程序
        cli
        push ax
        push dx
        push ds
        dec irq_times
        mov ax,data               ; Interrupt to do
        mov ds,ax
        mov dx,offset msg1
        mov ah,09h
        int 21h
        mov al,20h                ; Send EOI
        out 0a0h,al
        out 20h,al
        pop ds
        pop dx
        pop ax
        sti
        iret
        int_proc endp
code ends
         end start
```

实验八　可编程并行接口（二）（8255方式１）

一、实验目的

１．掌握8255工作方式１时的使用及编程。

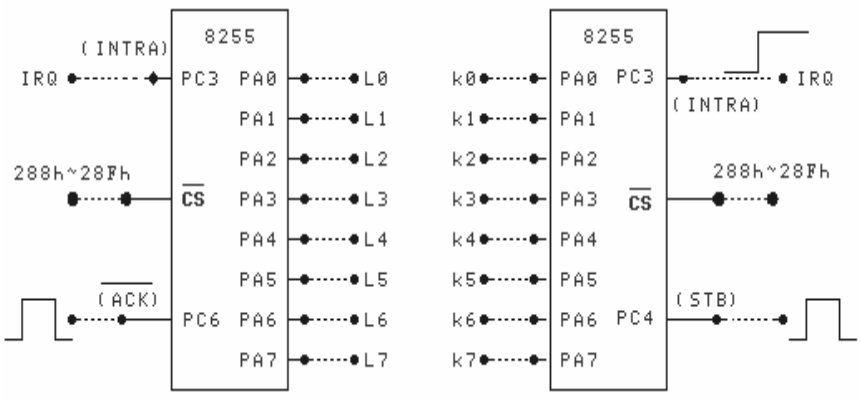
２．进一步掌握中断处理程序的编写。

二、实验内容

１．按图2-18（Ａ）8255方式1的输出电路连好线路。

２．编程：每按一次单脉冲按钮产生一个正脉冲使8255产生一次中断请求，让CPU进行一次中断服务：依次输出01H、02H，04H，08H，10H，20H，40H，80H使L0～L7依次发光，中断８次结束。

３．按图2-18（Ｂ）8255方式1输入电路，连好线路。

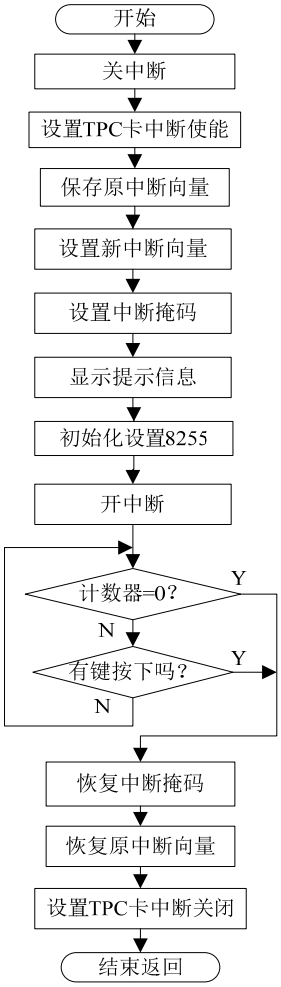


（A） 输出电路　　（B） 输入电路

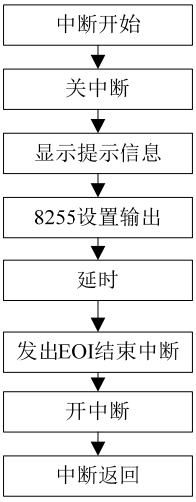
图2-18

４．编程：每按一次单脉冲按钮产生一个正脉冲使8255产生一次中断请求，让CPU进行一次中断服务，读取逻辑电平开关预置的ASCII码，在屏幕上显示其对应的字符，中断８次结束。

三、参考流程图:

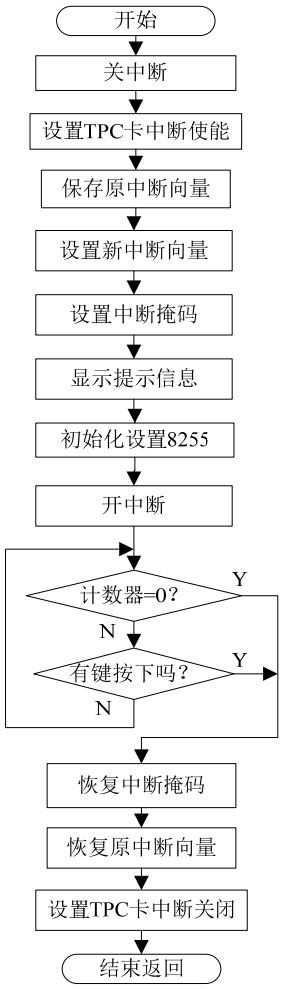


（A）输出主程序

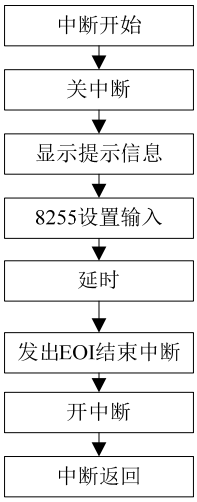


（B）输出中断服务程序

图 2-19



（A）输入主程序



（B）输入中断服务程序

图 2-20

四、参考程序１： E8255-io.ASM

```
; 386以上微机适用
; 纯dos下才能使用
; tasm4.1或以上编译
data segment
  int_vect EQU 071H          ; 中断0-7的向量为:08h-0fh,中断8-15的向量为:70h-77h
  irq_mask_2_7 equ 011111011b  ; 中断掩码,中断0-7时从低至高相应位为零,中断
                               8-15时第2位为零
  irq_mask_9_15 equ 011111101b ; 中断0-7时全一,中断8-15时从低至高相应位为零
  ioport_data equ 0d400h-280h  ; tpc 卡中设备的io地址
  ioport_cent equ 0d800h       ; tpc 卡中9054芯片的io地址
  csreg dw ?
  ipreg dw ?                   ; 旧中断向量保存空间
  portout dw 00h               ; 中断计数
  msg1 db 0dh,0ah,'TPC pci card Interrupt',0dh,0ah,'$'
  msg2 db 0dh,0ah,'Press any key to exit!',0dh,0ah,'$'
  msg3 db 0dh,0ah,'Press DMC to interrupt 8 times and exit!',0dh,0ah,'$'
data ends
stacks segment
db 100 dup (?)
stacks ends
code segment
assume cs:code,ds:data,ss:stacks,es:data
.386
start:
    cli
    mov ax,data
    mov ds,ax
    mov es,ax
    mov ax,stacks
    mov ss,ax
    mov dx,ioport_cent+68h     ; 设置tpc 卡中9054芯片io口,使能中断
    in  ax,dx
    or  ax,0900h
    out dx,ax
    mov al,int_vect            ; 保存原中断向量
    mov ah,35h
    int 21h
```

```
    mov ax,es
    mov csreg,ax
    mov ipreg,bx
    mov ax,cs                    ; 设置新中断向量
    mov ds,ax
    mov dx,offset int_proc
    mov al,int_vect
    mov ah,25h
    int 21h
    in  al, 21h                  ; 设置中断掩码
    and al, irq_mask_2_7
    out 21h, al
    in  al, 0a1h
    and al, irq_mask_9_15
    out 0a1h, al
    mov ax,data
    mov ds,ax
    mov dx,offset msg2
    mov ah,09h
    int 21h
    mov dx,offset msg3
    mov ah,09h
    int 21h
    mov dx,ioport_data+28bh          ; 置8255为A口方式1输出
    mov al,0a0h
    out dx,al
    mov al,0dh                       ; 将PC6置位
    out dx,al
    mov portout,1
    sti                              ; 开中断
loop1:
    cmp portout,100h                 ; 等待中断并判断中断8次后退出
    jz  exit
    mov ah,1
    int 16h
    jnz exit                         ; 按任意键退出
    jmp loop1
exit: cli
```

```
    mov bl, irq_mask_2_7            ; 恢复中断掩码
    not bl
    in  al, 21h
    or  al, bl
    out 21h, al
    mov bl, irq_mask_9_15
    not bl
    in  al, 0a1h
    or  al, bl
    out 0a1h, al
    mov dx,ipreg                    ; 恢复原中断向量
    mov ax,csreg
    mov ds,ax
    mov ah,25h
    mov al,int_vect
    int 21h
    mov dx,ioport_cent+68h          ; 设置tpc 卡中9054芯片io口,关闭中断
    in  ax,dx
    and ax,0f7ffh
    out dx,ax
    mov ax,4c00h
    int 21h
int_proc proc far
    cli
    push ax
    push bx
    push cx
    push dx
    push ds
    mov ax,data
    mov ds,ax
    mov dx,offset msg1
    mov ah,09h
    int 21h
    mov ax,portout
    mov dx,ioport_data+288h          ; 将AL从8255的A口输出
    out dx,al
    shl portout,1
```

```
    mov al,20h ;Send EOI
    out 0a0h,al
    out 20h,al
    mov cx,0ffffh
loop2:
    nop
    loop loop2                          ; 延时
    pop ds
    pop dx
    pop cx
    pop bx
    pop ax
    sti
    iret
    int_proc endp
    code ends
    end start
```

五、参考程序 2： E8255-1i.ASM

```
; 386以上微机适用
; 纯dos下才能使用
; tasm4.1或以上编译
data segment
int_vect EQU 071H          ; 中断0-7的向量为:08h-0fh,中断8-15的向量为:70h-77h
irq_mask_2_7 equ 011111011b ; 中断掩码,中断0-7时从低至高相应位为零,中断8-15
                            时第2位为零
irq_mask_9_15 equ 011111101b  ; 中断0-7时全一,中断8-15时从低至高相应位为零
ioport_data equ 0d400h-280h   ; tpc 卡中设备的io地址
ioport_cent equ 0d800h        ; tpc 卡中9054芯片的io地址
csreg dw ?
ipreg dw ?                    ; 旧中断向量保存空间
irq_times dw 00h              ; 中断计数
msg1 db 0dh,0ah,'TPC pci card Interrupt',0dh,0ah,'$'
msg2 db 0dh,0ah,'Press any key to exit!',0dh,0ah,'$'
msg3 db 0dh,0ah,'Press DMC to interrupt 10 times and exit!',0dh,0ah,'$'
data ends
stacks segment
```

```
db 100 dup (?)
stacks ends
code segment
    assume cs:code,ds:data,ss:stacks,es:data
start: mov ax,cs
.386
        cli
        mov ax,data
        mov ds,ax
        mov es,ax
        mov ax,stacks
        mov ss,ax
        mov dx,ioport_cent+68h     ; 设置tpc 卡中9054芯片io口,使能中断
        in  ax,dx
        or  ax,0900h
        out dx,ax
        mov al,int_vect            ; 保存原中断向量
        mov ah,35h
        int 21h
        mov ax,es
        mov csreg,ax
        mov ipreg,bx
        mov ax,cs                   ; 设置新中断向量
        mov ds,ax
        mov dx,offset int_proc
        mov al,int_vect
        mov ah,25h
        int 21h
        in  al, 21h                 ; 设置中断掩码
        and al, irq_mask_2_7
        out 21h, al
        in  al, 0a1h
        and al, irq_mask_9_15
        out 0a1h, al
        mov ax,data
        mov ds,ax
        mov dx,offset msg2
        mov ah,09h
```

```
        int 21h
        mov dx,offset msg3
        mov ah,09h
        int 21h
        mov dx,ioport_data+28Bh       ; 设8255为A口方式1输入
        mov al,0B8h
        out dx,al
        mov al,09h
        out dx,al
        mov irq_times,8               ; irq_times为中断次数计数器
        sti                           ; 开中断
loop1:  cmp irq_times,0h              ; 等待中断并判断中断8次后退出
        jz  exit
        mov ah,1
        int 16h
        jnz exit                      ; 按任意键退出
        jmp loop1
exit:   cli
        mov bl, irq_mask_2_7          ; 恢复中断掩码
        not bl
        in al, 21h
        or  al, bl
        out 21h, al
        mov bl, irq_mask_9_15
        not bl
        in  al, 0a1h
        or  al, bl
        out 0a1h, al
        mov dx,ipreg                  ; 恢复原中断向量
        mov ax,csreg
        mov ds,ax
        mov ah,25h
        mov al,int_vect
        int 21h
        mov dx,ioport_cent+68h        ; 设置tpc 卡中9054芯片io口,关闭中断
        in  ax,dx
        and ax,0f7ffh
        out dx,ax
```

```
        mov ax,4c00h
        int 21h
        int_proc proc far               ; 中断服务程序
        cli
        push ax
        push bx
        push cx
        push dx
        push ds
        mov ax,data
        mov ds,ax
        mov dx,offset msg1
        mov ah,09h
        int 21h
        mov dx,ioport_data+288h         ; 自8255A口输入一数据
        in al,dx
        mov dl,al                       ; 将所输入的数据保存到DL
        mov ah,02h                      ; 显示ASCII码为DL的字符
        int 21h
        mov dl,0dh                      ; 回车
        int 21h
        mov dl,0ah                      ; 换行
        int 21h
        dec irq_times                   ; 计数器减1
        mov al,20h                      ; Send EOI
        out 0a0h,al
        out 20h,al
        mov cx,0ffffh
loop2:  nop
        loop  loop2                     ; 延时
        pop ds
        pop dx
        pop cx
        pop bx
        pop ax
        sti
        iret
        int_proc endp
```

```
code ends
        end start
```

实验九 ＤＭＡ传送

一、实验目的

1 、掌握PC机工作环境下进行DMA方式数据传送(Block MODE和Demand Mode)(块传送、外部请求传送)方法。

2 、掌握DMA的编程方法。

二、实验内容

1 、用通用插座按图2-21将6116电路连接好。编程将主机内存缓冲区60000H,偏移量为0的一块数据循环写入字符A～Z，用Block MODE DMA方式传送到实验箱上的RAM6116上，再将实验箱上6116的内容用Block MODE DMA方式传送到主机内存缓冲区60000H,偏移量为0400H里。并察看送出与传回的数据是否一致。

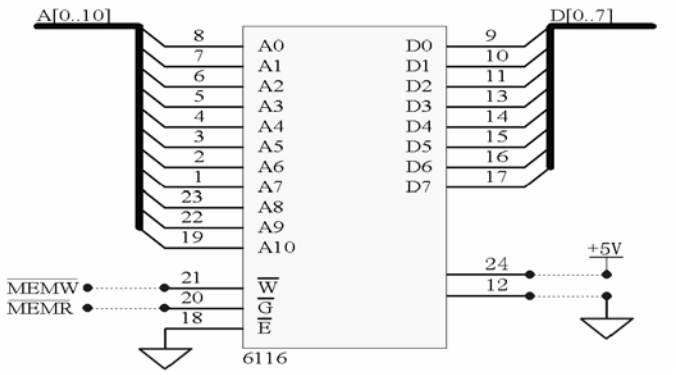


图 2-21

2 、用通用插座按图2-22将6116电路连接好（74LS74利用实验台上的D触发器）。编程将主机内存缓冲区60000H,偏移量为0的一块数据循环写入字符A～Z，用Block MODE DMA方式传送到实验箱上的RAM6116上，再将实验箱上6116的内容用Demand Mode DMA方式传送到主机内存缓冲区60000H,偏移量为0400H里，用户每发出一次DMA请求，PCI板卡上的DMA控制器将6116中的一个双字传送到主机内，程序不断显示主机内容。

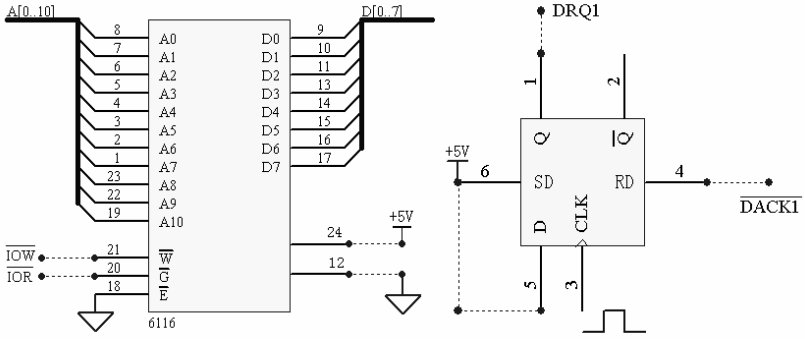


图 2-22

3 、用通用插座按图2-23连接好电路（74LS74利用实验台上的D触发器）。编程将主

机内存缓冲区60000H,偏移量为0的10个数据，使用Demand Mode DMA方式从内存向外设传送。

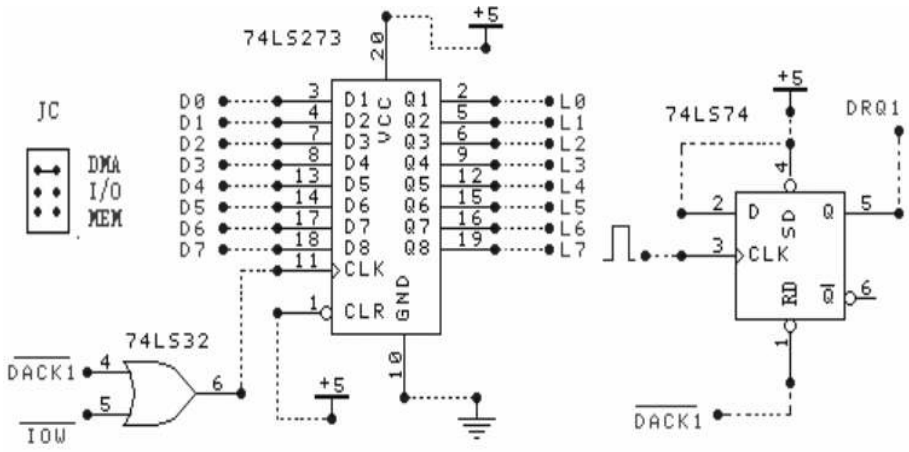


图 2-23

4 、用通用插座按图2-24连接好电路（74LS74利用实验台上的D触发器）。编程在主机内存缓冲区60000H,偏移量为0的位置开辟数据缓冲区，使用Demand Mode DMA方式从外设向内存传送8个数据并存入数据缓冲区，编程不断显示缓冲区的数据。

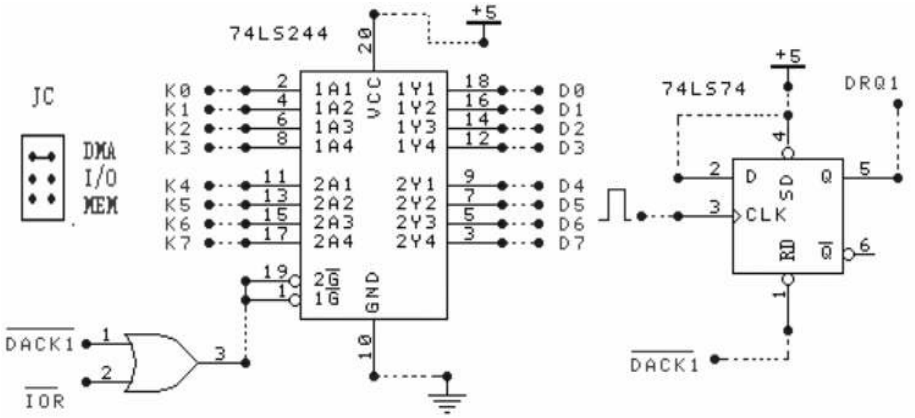


图 2-24

5 、利用上图2-24的电路。编程在主机内存缓冲区60000H,偏移量为0的位置开辟数据缓冲区，使用Demand Mode DMA方式从外设向内存传送8个数据并存入数据缓冲区，编程不断显示缓冲区的数据，并设置DMA传送结束产生硬件中断的方法，使DMA传送结束后显示提示。

三、实验提示

1 、 D M A 请求是由单脉冲输入到 D 触发器，由触发器的 Q 端向DRQ1发出的。CPU响应后发出DACK 1，将触发器Q置成低电平以撤消请求。

2、由于9054的驱动程序影响直写9054芯片的控制寄存器，DMA实验需要在纯DOS的环境中才能正常运行。这里指的纯DOS环境是指微机启动时按F8键进入的DOS环境。WINDOWS重启进入MSDOS方式由于系统资源被重新规划过，所以也不能正常实验。

3、由于TPC卡使用PCI总线，所以分配的中断号每台微机可能都不同，编程时需要了解当前的微机使用那个中断号并进行设置，获取方法请参看汇编程序使用方法的介绍。（也可使用自动获取资源分配的程序取得中断号）

4、在纯DOS环境下，有些微机的BIOS设置中有将资源保留给ISA总线使用的选项，致使在纯DOS环境（WINDOWS环境下不会出现此问题）下PCI总线无法获得系统资源，也就无法做实验，这时需要将此选项修改为使用即插即用。

5、在纯DOS环境下，有些微机的BIOS设置中有使用即插即用操作系统的选项，如果在使用即插即用操作系统状态下，BIOS将不会给TPC卡分配系统资源，致使在纯DOS环境（WINDOWS环境下不会出现此问题）下PCI总线无法获得系统资源，也就无法做实验，这时需要将此选项修改为不使用即插即用操作系统。

6、本实验使用9054接口芯片控制DMA的传送，需要进行初始化的寄存器如下（通道0）：

偏移地址	名称	长度	功　能
68H	INTCSR	32位	中断模式设置
80H	DNANODE0	32位	DMA模式寄存器
84H	DMAPADR0	32位	DMA PCI端起始地址值，微机端
88H	DMALADR0	32位	DMA LOCAL端起始地址值，设备端
8CH	DMASIZ0	32位	传输数据长度，以字节为单位
90H	DMADPR0	32位	DMA 传输控制，方向/中断
A8H	MACSR0	8位	DMA命令/状态、开始/结束/取消/中断/当前传输状态

寄存器设置步骤如下：

i. 如果需要DMA传输结束产生中断，需要设置INTCSR(68H)，中断模式设置，其它中断设置请参看中断程序中的说明

BITS	功　能
8	1：能够产生PCI中断 0：禁止产生PCI中断
11	1：能够LOCAL端输入的中断送到PCI端 0：禁止LOCAL端输入的中断送到PCI端
18	1：DMA通道0中断使能 0：DMA通道0中断关闭
其它位为零即可，更多内容参看9054芯片数据手册	

ii. 设置DMAMODE0(80H)，DMA传输模式

BITS	功　能
1：0	LOCAL端总线宽度： 00：8位 01：16位 10 or 11：32位
6	设为：1
9	设为：0，块模式传输
10	1：DMA传送完成能产生中断 0：DMA传送完成不产生中断
12	1：外部硬件请求模式传输（只进行32位传输） 0：软件启动传输
17	1：DMA传送完成产生中断去PCI端，微机端 0：DMA传送完成产生中断去LOCAL端，设备端
其它位为零即可，更多内容参看9054芯片数据手册	

iii. 设置DMAPADR0(84H)，DMA PCI端起始地址值，微机端，32位物理地址值

iv. 设置DMALADR0(88H)，DMA LOCAL端起始地址值，设备端，32位物理地址值，当设备端为扩展IO范围地址时，最高位（31）应置为“1”，表示为IO范围地址。

v. 设置DMASIZ0(8CH)，待传输数据长度，23位，31：23位未用，可不关心。

vi. 设置DMADPR0(90H)，DMA 传输控制，方向/中断

BITS	功　能
2	1：DMA传送结束产生中断 0：DMA传送结束不产生中断
3	1：LOCAL端向PCI端传送，设备向微机 0：PCI端向LOCAL端传送，微机向设备
其它位为零即可，更多内容参看9054芯片数据手册	

vii. 设置DMACSR0(A8H)，DMA 命令/状态

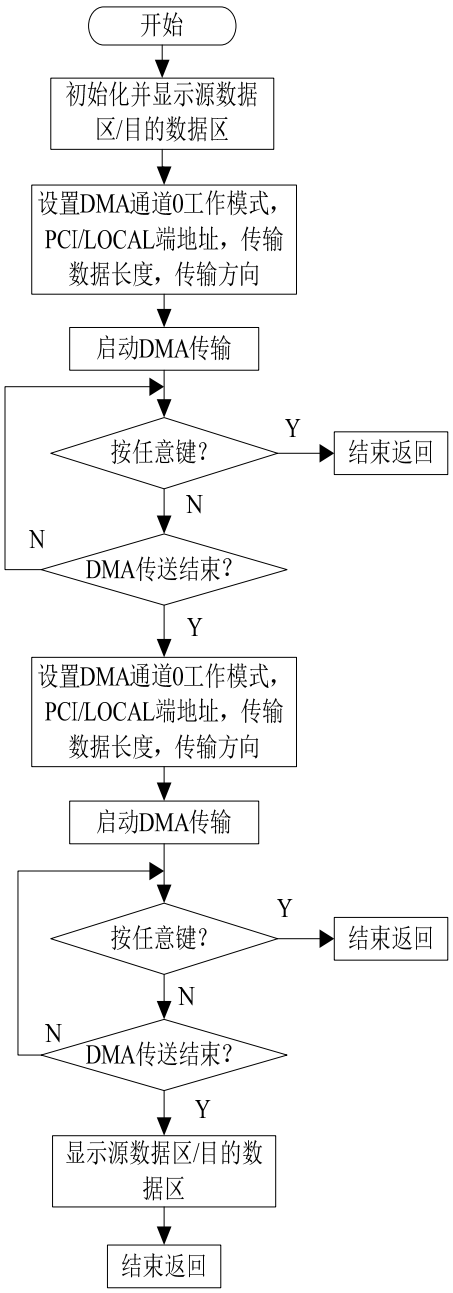
BITS	功　能
0	1：DMA传送使能 0：DMA传送关闭
1	1：启动DMA传输
2	1：中止DMA传输
3	1：清除DMA传送结束中断标志
4	读为1：DMA传送结束 读为0：DMA传送中
7：5	未用
更多内容参看9054芯片数据手册	

7、当使用外部请求传送模式进行DMA传输时，9054只进行32位的传送，传送8、16位数据时需要将数据规整为32位后传输。

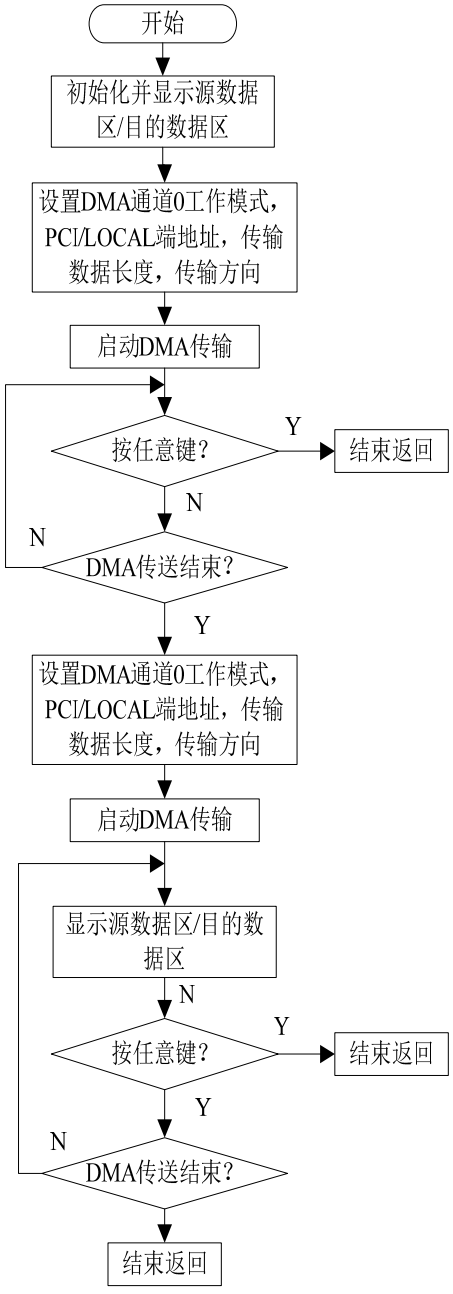
8、因为PCI总线最大宽度为64位，9054芯片为了节省传输资源，当由LOCAL端外部输入数据时，芯片将数据缓存，凑够64位后一次送入PCI端，即微机端。这会出现进行两次外部DMA请求后9054芯片才将64位数据传送到PCI端，即微机端的情况。

9、DMA未执行完撤消流程：1.清除DMA通道使能位(DMACSR0[0]=0) 2.设置通道终止位取消DMA传送(DMACSR0[2]=1) 3.等待DMA传送结束位变高(DMACSR0[4]=1)(或延迟一会)注意：取消后DMA传输仍可能会传送1-2个数据。如果当前没有DMA操作而执行取消命令，会使下一个将要执行的DMA传输被取消。更多内容参看9054芯片数据手册。

四、参考流程图

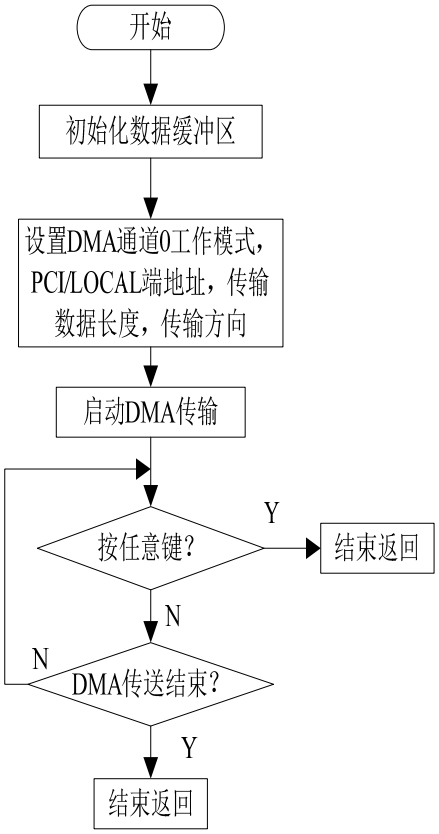


块模式DMA传输

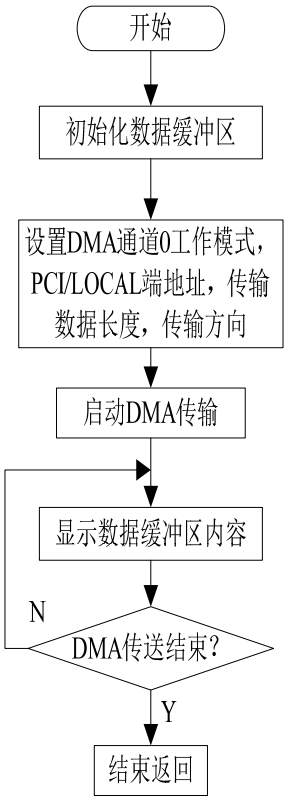


外部请求模式DMA传输

图2-25

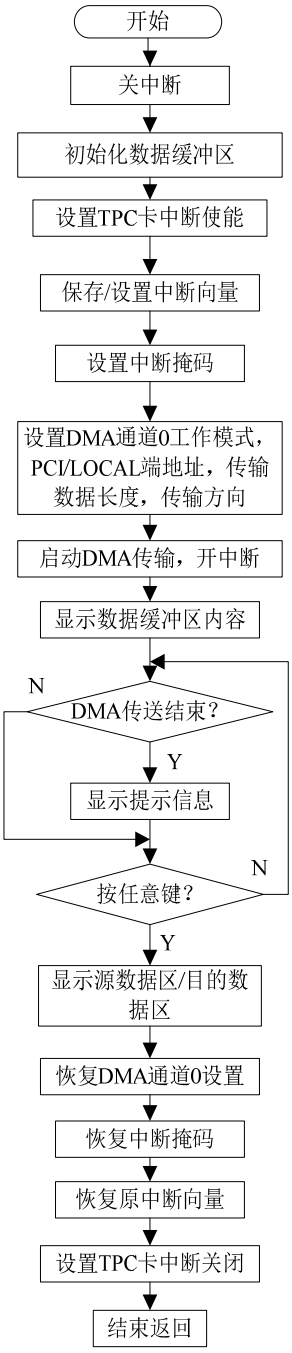


外部请求模式DMA输出

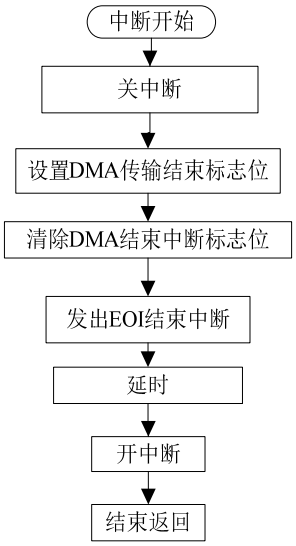


外部请求模式DMA输入

图2-26



DMA传送结束产生中断的外部请求模式DMA输入



DMA传送结束产生中断的中断程序

图2-27

五、参考程序1 DMA.ASM

```
; 386以上微机适用
; 纯dos下可以用
; tasm4.1或以上编译
; **********************
; * DMA实验              *
; * Block MODE          *
; **********************
ioport_cent equ 0d800h          ; TPC卡PCI接口芯片IO寄存器基址
mem_data equ 00h                ; TPC卡逻辑端存储器基址，PCI卡扩展的6116
dmamode equ 001000000b          ; DMA通道模式字
dmacount equ 100h               ; 传输数据长度
dmadpr equ 0000b                ; 数据传输方向，PCI->LOCAL，主机向TPC设备
dmadpr1 equ 1000b               ; 数据传输方向，LOCAL->PCI，TPC设备向主机
dmacsr equ 0011b                ; 启动DMA传输
extra segment at 6000h          ; 主机缓冲区
in_data db 0400h dup (?)        ; 源数据区
in_data1 db 0400h dup (?)       ; 目的数据区
extra ends
stacks segment
db 100h dup(?)
stacks ends
code SEGMENT
start:
.386p
call writemem                   ; 写源数据区，清空目的数据区
call dispmem                    ; 显示源数据区，目的数据区
mov dx,ioport_cent+0a8h
in al,dx
and al,10h                      ; 读取dma控制器当前状态
jnz start_next
mov al,0
out dx,al
mov al,4
out dx,al                       ; dma控制器忙时取消以前dma传输
call delay2
start_next:                     ; dma控制器空闲时
mov dx,ioport_cent+80h
```

```
mov eax,dmamode
out dx,eax                          ; 设置DMA通道0模式
mov dx,ioport_cent+84h
mov eax,060000h
mov ecx,offset in_data
add eax,ecx
out dx,eax                          ; 设置PCI接口端存储器地址，微机内存的地址
mov dx,ioport_cent+88h
mov eax,mem_data
out dx,eax                          ; 设置local存储器地址，PCI卡扩展的6116
mov dx,ioport_cent+8ch
mov eax,dmacount
out dx,eax                          ; 设置传输数据长度
mov dx,ioport_cent+90h
mov eax,dmadpr
out dx,eax                          ; 设置传输方向等
mov dx,ioport_cent+0a8h
mov al,dmacsr
out dx,al                           ; 启动传送
next:
mov ah,1
int 16h
jnz exit                            ; 按任意键退出
in al,dx
and ax,10000b
jz next                             ; 等待DMA传送结束
mov dx,ioport_cent+80h
mov eax,dmamode
out dx,eax                          ; 设置DMA通道0模式
mov dx,ioport_cent+84h
mov eax,060000h
mov ecx,offset in_data1
add eax,ecx
out dx,eax                          ; 设置PCI接口端存储器地址，微机内存的地址
mov dx,ioport_cent+88h
mov eax,mem_data
out dx,eax                          ; 设置local存储器地址，PCI卡扩展的6116
mov dx,ioport_cent+8ch
```

```
mov eax,dmacount
out dx,eax                          ; 设置传输数据长度
mov dx,ioport_cent+90h
mov eax,dmadpr1
out dx,eax                          ; 设置传输方向等
mov dx,ioport_cent+0a8h
mov al,dmacsr
out dx,al                           ; 启动传送
next1:
mov ah,1
int 16h
jnz exit                            ; 按任意键退出
in al,dx
and ax,10000b
jz next1                            ; 等待DMA传送结束
call dispmem                        ; 显示源数据区，目的数据区
exit:
mov dx,ioport_cent+0a8h
in al,dx
and al,10h                          ; 读取dma控制器当前状态
jnz start_exit
mov al,0
out dx,al
mov al,4
out dx,al                           ; dma控制器忙时取消以前dma传输
call delay2
start_exit:                         ; dma控制器空闲时
mov al,0h
out dx,al
mov ax,4c00h
int 21h
writemem proc
mov ax,extra
mov ds,ax
mov si,offset in_data
mov cx,100h
mov dl,40h
loopx1:
```

```
inc dl
mov [si],dl
add si,1
cmp dl,5ah
jnz loopx2
mov dl,40h
loopx2:
loop loopx1                    ; 向源存储空间反复写256个a-z
mov ax,extra
mov ds,ax
mov si,offset in_data1
mov cx,100h
mov dl,30h
loopx4:
mov [si],dl
add si,1
loop loopx4                    ; 向目标存储空间写256个0
ret
writemem endp
dispmem proc
mov ax,extra
mov ds,ax
mov si,offset in_data
mov cx,100h
loopx3:
mov dl,[si]
mov ah,02h
int 21h
add si,1
loop loopx3                    ; 从源存储空间读256个字节内容并显示
mov dl,0dh
int 21h
mov dl,0ah
int 21h
mov ax,extra
mov ds,ax
mov si,offset in_data1
mov cx,100h
```

```
loopx5:
mov dl,[si]
mov ah,02h
int 21h
add si,1
loop loopx5                          ；从目标存储空间读256个字节内容并显示
mov dl,0dh
int 21h
mov dl,0ah
int 21h
ret
dispmem endp
delay2 proc near
pusha ;delay
pushf
mov cx,0100h
delay2_2:
push cx
mov cx,0ffffh
delay2_1:
nop
loop delay2_1
pop cx
nop
loop delay2_2
popf
popa
ret
delay2 endp
code ENDS
END start
```

六、参考程序2 DMA1.ASM

```
；386以上微机适用
；纯dos下可以用
；tasm4.1或以上编译
；**********************
```

```
; * DMA实验              *
; * Demand MODE          *
; ************************
ioport_cent equ 0d800h              ; TPC卡PCI接口芯片IO寄存器基址
io_data equ 080000000h              ;TPC卡逻辑端输入输出基址，PCI卡扩展的6116
dmamode equ 001000000b              ; DMA通道模式字,块模式
dmamode1 equ 01000001000000b        ; DMA通道模式字,Demand模式
dmacount equ 40h                    ; 传输数据长度，40h字节，10h双字
dmadpr equ 0000b                    ; 数据传输方向，PCI->LOCAL，主机向TPC设备
dmadpr1 equ 1000b                   ; 数据传输方向，LOCAL->PCI，TPC设备向主机
dmacsr equ 0011b                    ; 启动DMA传输
extra segment at 6000h
in_data db 0400h dup (?)            ; 源数据区
in_data1 db 0400h dup (?)           ; 目的数据区
extra ends
stacks segment
db 100h dup(?)
stacks ends
code SEGMENT
ASSUME CS:code,DS:stacks,SS:stacks,ES:stacks
start:
.386p
call writemem                       ; 写源数据区，清空目的数据区
in al,dx
and al,10h                          ; 读取dma控制器当前状态
jnz start_next
mov al,0
out dx,al
mov al,4
out dx,al                           ; dma控制器忙时取消以前dma传输
call delay2
start_next:                         ; dma控制器空闲时
mov dx,ioport_cent+80h
mov eax,dmamode
out dx,eax ;设置dma通道0
mov dx,ioport_cent+84h
mov eax,060000h
mov ecx,offset in_data
```

```
add eax,ecx
out dx,eax                          ; 设置PCI接口端存储器地址，微机内存的地址
mov dx,ioport_cent+88h
mov eax,io_data
out dx,eax                          ; 设置local存储器地址，PCI卡扩展的6116
mov dx,ioport_cent+8ch
mov eax,dmacount
out dx,eax                          ; 设置传输数据长度
mov dx,ioport_cent+90h
mov eax,dmadpr
out dx,eax                          ; 设置传输方向等
mov dx,ioport_cent+0a8h
mov al,dmacsr
out dx,al                           ; 启动传送
next3:
mov ah,1
int 16h
jnz exit                            ; 按任意键退出
in al,dx
and ax,10000b
jz next3                            ; 等待DMA传送结束
mov dx,ioport_cent+80h
mov eax,dmamode1
out dx,eax                          ; 设置DMA通道0模式
mov dx,ioport_cent+84h
mov eax,060000h
mov ecx,offset in_data1
add eax,ecx
out dx,eax                          ; 设置PCI接口端存储器地址，微机内存的地址
mov dx,ioport_cent+88h
mov eax,io_data
out dx,eax                          ; 设置local存储器地址，PCI卡扩展的6116
mov dx,ioport_cent+8ch
mov eax,dmacount
out dx,eax                          ; 设置传输数据长度
mov dx,ioport_cent+90h
mov eax,dmadpr1
out dx,eax                          ; 设置传输方向等
```

```
mov dx,ioport_cent+0a8h
mov al,dmacsr
out dx,al                          ; 启动传送
next1:
call dispmem
mov ah,1
int 16h
jnz exit                           ; 按任意键退出
in al,dx
and ax,10000b
jz next1                           ; 等待DMA传送结束
call dispmem
exit:
mov dx,ioport_cent+0a8h
in al,dx
and al,10h                         ; 读取dma控制器当前状态
jnz start_exit
mov al,0
out dx,al
mov al,4
out dx,al                          ; dma控制器忙时取消以前dma传输
call delay2
start_exit:                        ; dma控制器空闲时
mov al,0h
out dx,al
mov ax,4c00h
int 21h
writemem proc
mov ax,extra
mov ds,ax
mov si,offset in_data
mov cx,100h
mov dl,40h
loopx1:
inc dl
mov [si],dl
add si,1
cmp dl,5ah
```

```
jnz loopx2
mov dl,40h
loopx2:
loop loopx1                    ; 向源存储空间反复写256个a-z
mov ax,extra
mov ds,ax
mov si,offset in_data1
mov cx,100h
mov dl,30h
loopx4:
mov [si],dl
add si,1
loop loopx4                    ; 向目标存储空间写256个0
ret
writemem endp
dispmem proc
mov ax,extra
mov ds,ax
mov si,offset in_data1
mov cx,40h
loopx5:
mov dl,[si]
mov ah,02h
int 21h
add si,1
loop loopx5                    ; 从目标存储空间读256个字节内容并显示
mov dl,0dh
int 21h
mov dl,0ah
int 21h
ret
dispmem endp
delay2 proc near
pusha ;delay
pushf
mov cx,0100h
delay2_2:
push cx
```

```
mov cx,0ffffh
delay2_1:
nop
loop delay2_1
pop cx
nop
loop delay2_2
popf
popa
ret
delay2 endp
code ENDS
END start
```

七、参考程序3 DMA_0.ASM

```
; 386以上微机适用
; 纯dos下可以用
; tasm4.1或以上编译
; ***********************
; * DMA实验              *
; * 273 输出电路        *
; ***********************
ioport_cent equ 0d800h              ; TPC卡PCI接口芯片IO寄存器基址
io_data equ 080000000h              ; TPC卡逻辑端输入输出基址，PCI卡扩展
dmamode equ 01000001000000b         ; DMA通道模式字,Demand模式
dmacount equ 40                     ; 传输数据长度，40字节，10双字
dmadpr equ 0000b                    ; 数据传输方向，PCI->LOCAL，主机向TPC设备
dmacsr equ 0011b                    ; 启动DMA传输
data SEGMENT
out_data db 0,0,0,01h,0,0,0,02h,0,0,0,04h,0,0,0,08h,0,0,0,10h
db 0,0,0,20h,0,0,0,40h,0,0,0,80h,0,0,0,0ffh,0,0,0,0h
data ENDS
extra segment at 6000h
ext db dmacount dup(?)              ; 数据缓冲区
extra ends
stacks segment
db 100h dup(?)
stacks ends
code SEGMENT
```

```
ASSUME CS:code,DS:data,SS:stacks,ES:data
start:
.386p
mov ax,data
mov ds,ax
mov ax,extra
mov es,ax
lea si,out_data
lea di,ext
cld
mov cx,dmacount
rep movsb                          ; 将数据写入缓冲区
al,dx
and al,10h                         ; 读取dma控制器当前状态
jnz start_next
mov al,0
out dx,al
mov al,4
out dx,al                          ; dma控制器忙时取消以前dma传输
call delay2
start_next:                        ; dma控制器空闲时
mov dx,ioport_cent+80h
mov eax,dmamode
out dx,eax                         ; 设置dma通道0
mov dx,ioport_cent+84h
mov eax,060000h
mov ecx,offset ext
add eax,ecx
out dx,eax                         ; 设置PCI接口端存储器地址，微机内存的地址
mov dx,ioport_cent+88h
mov eax,io_data
out dx,eax                         ; 设置local存储器地址，PCI卡扩展
mov dx,ioport_cent+8ch
mov eax,dmacount
out dx,eax                         ; 设置传输数据长度
mov dx,ioport_cent+90h
mov eax,dmadpr
out dx,eax                         ; 设置传输方向等
```

```
mov dx,ioport_cent+0a8h
mov al,dmacsr
out dx,al                          ; 启动传送
next3:
mov ah,1
int 16h
jnz exit                           ; 按任意键退出
in al,dx
and ax,10000b
jz next3                           ; 等待DMA传送结束
exit:
mov dx,ioport_cent+0a8h
in al,dx
and al,10h                         ; 读取dma控制器当前状态
jnz start_exit
mov al,0
out dx,al
mov al,4
out dx,al                          ; dma控制器忙时取消以前dma传输
call delay2
start_exit:                        ; dma控制器空闲时
mov al,0h
out dx,al
mov ax,4c00h
int 21h
delay2 proc near
pusha ;delay
pushf
mov cx,0100h
delay2_2:
push cx
mov cx,0ffffh
delay2_1:
nop
loop delay2_1
pop cx
nop
loop delay2_2
```

```
popf
popa
ret
delay2 endp
code ENDS
END start
```

八、参考程序4 DMA_I.ASM

```
; 386以上微机适用
; 纯dos下可以用
; tasm4.1或以上编译
; ***********************
; * DMA实验              *
; * 244 输入电路        *
; ***********************
ioport_cent equ 0d800h              ; TPC卡PCI接口芯片IO寄存器基址
io_data equ 080000000h              ; TPC卡逻辑端输入输出基址，PCI卡扩展
dmamode equ 01000001000000b         ; DMA通道模式字,Demand模式
dmacount equ 32                     ; 传输数据长度，32字节，8双字
dmadpr equ 1000b                    ; 数据传输方向，LOCAL->PCI，TPC设备向主机
dmacsr equ 0011b                    ; 启动DMA传输
data SEGMENT
in_data db dmacount dup(0)
data ENDS
extra segment at 6000h
ext db dmacount dup(?)              ; 数据缓冲区
extra ends
stacks segment
db 100h dup(?)
stacks ends
code SEGMENT
ASSUME CS:code,DS:data,SS:stacks,ES:data
start:
.386p
mov ax,data
mov ds,ax
mov ax,extra
```

```
mov es,ax
lea si,in_data
lea di,ext
cld
mov cx,dmacount
rep movsb                          ; 清空数据区
mov dx,ioport_cent+0a8h
in al,dx
and al,10h                         ; 读取dma控制器当前状态
jnz start_next
mov al,0
out dx,al
mov al,4
out dx,al                          ; dma控制器忙时取消以前dma传输
call delay2
start_next:                        ; dma控制器空闲时
mov dx,ioport_cent+80h
mov eax,dmamode
out dx,eax                         ; 设置dma通道0
mov dx,ioport_cent+84h
mov eax,060000h
mov ecx,offset ext
add eax,ecx
out dx,eax                         ; 设置PCI接口端存储器地址，微机内存的地址
mov dx,ioport_cent+88h
mov eax,io_data
out dx,eax                         ; 设置local存储器地址，PCI卡扩展
mov dx,ioport_cent+8ch
mov eax,dmacount
out dx,eax                         ; 设置传输数据长度
mov dx,ioport_cent+90h
mov eax,dmadpr
out dx,eax                         ; 设置传输方向等
mov dx,ioport_cent+0a8h
mov al,dmacsr
out dx,al                          ; 启动传送
next3:
mov ax,6000h
```

```
mov ds,ax
mov si,offset ext
mov cx,8
loop3:
mov ax,[si+2]
call disp
mov ax,[si]
call disp
add si,4
mov ah,02h
mov dl,20h
int 21h
loop loop3                          ; 从数据区读8个双字内容并显示
mov ah,02h
mov dl,0dh
int 21h
mov dl,0ah
int 21h                             ; 显示回车、换行
mov ah,1
int 16h
jz next3                            ; 按任意键退出
mov dx,ioport_cent+0a8h
in al,dx
and al,10h                          ; 读取dma控制器当前状态
jnz start_exit
mov al,0
out dx,al
mov al,4
out dx,al                           ; dma控制器忙时取消以前dma传输
call delay2
start_exit:                         ; dma控制器空闲时
mov al,0h
out dx,al
mov ax,4c00h
int 21h                             ; 退出
disp proc near                      ; 显示子程序
push dx
push cx
```

```
push bx
mov cx,4
mov bx,16
loop1: push ax
push cx
sub bx,4
mov cx,bx
shr ax,cl
and al,0fh                    ; 首先取低四位
mov dl,al
cmp dl,9                      ; 判断是否<=9
jle num                       ; 若是则为'0'-'9',ASCII码加30H
add dl,7                      ; 否则为'A'-'F',ASCII码加37H
num: add dl,30h
mov ah,02h                    ; 显示
int 21h
pop cx
pop ax
loop loop1
pop bx
pop cx
pop dx
ret                           ; 子程序返回
disp endp
delay2 proc near
pusha ;delay
pushf
mov cx,0100h
delay2_2:
push cx
mov cx,0ffffh
delay2_1:
nop
loop delay2_1
pop cx
nop
loop delay2_2
popf
```

```
popa
ret
delay2 endp
code ENDS
END start
```

九、参考程序5 DMA_I1.ASM

```
; 386以上微机适用
; 纯dos下可以用
; tasm4.1或以上编译
; **********************
; * DMA实验              *
; * 244 输入电路        *
; * 使用DMA结束产生中断 *
; **********************
ioport_cent equ 0d800h          ; TPC卡PCI接口芯片IO寄存器基址
io_data equ 080000000h          ; TPC卡逻辑端输入输出基址，PCI卡扩展
dmamode equ 100001010001000000b ; DMA通道模式字,Demand模式,DMA结束产生中断
dmacount equ 32                 ; 传输数据长度，32字节，8双字
dmadpr equ 1100b                ; 数据传输方向，LOCAL->PCI，TPC设备向主机，
                                  DMA结束产生中断
dmacsr equ 0011b                ; 启动DMA传输
int_vect EQU 071H               ; 新的中断向量,中断0-7的向量为:08h-0fh,
                                  中断8-15的向量为:70h-77h
irq_mask_2_7 equ 011111011b     ; 新的中断掩码,中断0-7时从低至高相应位为零,
                                  中断8-15时第2位为零
irq_mask_9_15 equ 011111101b    ; 新的中断掩码,中断0-7时全一,中断8-15时
                                  从低至高相应位为零
data SEGMENT
csreg dw ?
ipreg dw ?                       ; 旧中断向量保存空间
irq_times db 00h                 ; DMA传输结束标志位,为1时表示DMA传送结束
msg1 db 'DMA Transfare finished! Press any key to exit!',0dh,0ah,'$'
in_data db dmacount dup(0)
data ENDS
extra segment at 6000h
```

```
ext db dmacount dup(?)
extra ends
stacks segment
db 100h dup(?)
stacks ends
code SEGMENT
ASSUME CS:code,DS:data,SS:stacks,ES:data
start:
.386p
cli
mov ax,stacks
mov ss,ax
mov ax,data
mov ds,ax
mov ax,extra
mov es,ax
lea si,in_data
lea di,ext
cld
mov cx,dmacount
rep movsb                          ; 清空数据区
mov irq_times,0h
mov dx,ioport_cent+0a8h
in al,dx
and al,10h                         ; 读取dma控制器当前状态
jnz start_next
mov al,0
out dx,al
mov al,4
out dx,al                          ; dma控制器忙时取消以前dma传输
call delay2
start_next:                        ; dma控制器空闲时
mov dx,ioport_cent+68h             ; 设置TPC卡中9054芯片io口,使能DMA传输结束中
断
in eax,dx
or eax,040100h
out dx,eax
mov al,int_vect                    ; 保存原中断向量
```

```
mov ah,35h
int 21h
mov ax,es
mov csreg,ax
mov ipreg,bx
mov ax,cs                       ; 设置新中断向量
mov ds,ax
mov dx,offset int_proc
mov al,int_vect
mov ah,25h
int 21h
in al, 21h                      ; 设置中断掩码
and al, irq_mask_2_7
out 21h, al
in al, 0a1h
and al, irq_mask_9_15
out 0a1h, al
mov dx,ioport_cent+80h
mov eax,dmamode
out dx,eax                      ; 设置dma通道0
mov dx,ioport_cent+84h
mov eax,060000h
mov ecx,offset ext
add eax,ecx
out dx,eax                      ; 设置PCI接口端存储器地址，微机内存的地址
mov dx,ioport_cent+88h
mov eax,io_data
out dx,eax                      ; 设置local存储器地址，PCI卡扩展
mov dx,ioport_cent+8ch
mov eax,dmacount
out dx,eax                      ; 设置传输数据长度
mov dx,ioport_cent+90h
mov eax,dmadpr
out dx,eax                      ; 设置传输方向等
mov dx,ioport_cent+0a8h
mov al,dmacsr
out dx,al                       ; 启动传送
sti                             ; 开中断
```

```
next3:
mov ax,6000h
mov ds,ax
mov si,offset ext
mov cx,8
cli
loop3:
mov ax,[si+2]
call disp
mov ax,[si]
call disp
add si,4
mov ah,02h
mov dl,20h
int 21h
loop loop3                          ; 从数据区读8个双字内容并显示
mov ah,02h
mov dl,0dh
int 21h
mov dl,0ah
int 21h                             ; 显示回车、换行
mov ax,data
mov ds,ax
cmp irq_times,1h
jnz next2
mov dx,offset msg1
mov ah,09h
mov ah,1
int 16h
jz next3                            ; 按任意键退出
exit:
cli
mov dx,ioport_cent+0a8h
in al,dx
and al,10h                          ; 读取dma控制器当前状态
jnz exit1
mov al,0
out dx,al
```

```
mov al,4
out dx,al                          ; dma控制器忙时取消以前dma传输
call delay2
exit1:                             ; dma控制器空闲时
mov al,0h
out dx,al
mov dx,ioport_cent+80h
mov ebx,dmamode
not ebx
in eax,dx
and eax,ebx
out dx,eax                         ; 恢复dma通道0设置
mov bl, irq_mask_2_7               ; 恢复中断掩码
not bl
in al, 21h
or al, bl
out 21h, al
mov bl, irq_mask_9_15
not bl
in al, 0a1h
or al, bl
out 0a1h, al
mov dx,ipreg                       ; 恢复原中断向量
mov ax,csreg
mov ds,ax
mov ah,25h
mov al,int_vect
int 21h
mov dx,ioport_cent+68h             ; 设置tpc 卡中9054芯片io口,关闭中断
in eax,dx
and eax,0bfeffh
out dx,eax
mov ax,4c00h
int 21h                            ; 退出
disp proc near                     ; 显示子程序
cli
push dx
push cx
```

```
push bx
mov cx,4
mov bx,16
loop1: push ax
push cx
sub bx,4
mov cx,bx
shr ax,cl
and al,0fh                    ; 首先取低四位
mov dl,al
cmp dl,9                      ; 判断是否<=9
jle num                       ; 若是则为'0'-'9',ASCII码加30H
add dl,7                      ; 否则为'A'-'F',ASCII码加37H
num: add dl,30h
mov ah,02h                    ; 显示
int 21h
pop cx
pop ax
loop loop1
pop bx
pop cx
pop dx
sti
ret                           ; 子程序返回
disp endp
int_proc proc far             ; 中断程序
cli
push ax
push cx
push dx
push ds
mov ax,data
mov ds,ax
mov irq_times,1h              ; 设置DMA传输结束标志位
mov dx,ioport_cent+0a8h
in al,dx
mov bl,8h
or al,bl
```

```
out dx,al                       ; 清除DMA结束中断标志位
mov al,20h                      ; Send EOI
out 0a0h,al
out 20h,al
mov cx,0ffffh
loopx:
nop
loop loopx                      ; 延时
pop ds
pop dx
pop cx
pop ax
sti
iret
int_proc endp
delay2 proc near
pusha ;delay
pushf
mov cx,0100h
delay2_2:
push cx
mov cx,0ffffh
delay2_1:
nop
loop delay2_1
pop cx
nop
loop delay2_2
popf
popa
ret
delay2 endp
code ENDS
END start
```

附录一　上机操作

一、在 IBM-PC 机上运行汇编源程序所必须具备的软件

在 IBM-PC 机上运行汇编源程序必须具有以下软件：

- DOS 2．0 以上版本
- 编辑程序：行编辑程序 EDLIN、多功能窗口软件 SIDEKICK
- 宏汇编程序：MASM(1．0 以上版本)
- 连接程序：LINK
- 库管理程序：LIB
- 调试程序；DEBUG、TURBO DEBUGGER

对于编辑程序和调试程序，只要具备其中之一即可。

通常，将以上软件拷贝到磁盘上，当需要运行汇编源程序时，可随时调用以进行相应的处理。

二、运行汇编源程序的过程

运行汇编源程序必须经过以下 4 个步骤。

①建立与修改汇编源程序文件(简称源文件，其扩展名为．ASM)。

②汇编源文件以产生相应的目标文件(扩展名为．OBJ)。

将源文件汇编成目标文件的工作是通过 DOS 调用宏汇编程序 MASM 来实现的。

③连接目标文件以建立可执行文件(扩展名为．EXE)。

连接工作是通过 DOS 调用 LINK 程序来实现的。

④调试、运行可执行文件。

经过①一③步骤之后，可执行文件已存放在磁盘上，此时，在 DOS 提示符下，直接打入文件名(可不要扩展名．EXE)，就可以运行可执行文件。

然而，在设计一个较复杂的汇编语言源程序时，免不了会出现一些错误，很难一次通过，这就需要调用 DOS 支持下的 DEBUG 程序或 TURBO DEBUGGER 程序(调试程序)，调试所生成的可执行文件。在发现程序中的错误之后，要重复编辑过程，修改源程序中的错误，将修改后的源程序再经过汇编、连接、执行这几个步骤，直到满足设计要求为止。

现在用一个具体例子来说明上机过程。

要求在显示器上显示如下信息：

```
IBM MICRO COMPUTER
```

汇编源程序如下：

```
DATA SEGMENT
    BUFDB”IBM MICRO COMPUTER $”
DATA ENDS
    CODE  SEGMENT
    ASSUME CS: CODE，DS：DATA
    START：MOV AX，DATA
           MOV DS，AX
```

```
        LEA DX，BUF
        MOV AH，09H
        INT 21H
        MOV AH，4CH
        INT 21H
CODE    ENDS
        END START
```

按照上述步骤，首先是调用编辑程序以建立源文件并存入磁盘，文件名设为DISCH.ASM，然后对该文件进行汇编、连接、运行，各命令形式如下(假设所有软件都在硬盘上)。

① C>SK↙

此命令是调用编辑程序SIDEKICK，然后将上述源程序通过键盘输入，并可存在磁盘上。(具体用法见三、建立与修改源文件)

② C>MASM DISCH.ASM; ↙

此命令是调用宏汇编程序MASM对源程序文件DISCH. ASM进行汇编，生成目标文件。该命令行后带分号则表示MASM不对用户提问直接生成. OBJ文件。

如汇编没有错误，则做第3步。

③ C>LINK DISCH; ↙

此命令是调用连接程序建立可执行文件。该命令行后带分号则表示LINK不对用户提问直接生成. ExE文件。

如连接成功，则做第4步。

④ C>DISCH↙

运行后，显示结果：

```
IBM MICRO COMPUTER
```

如果未得到预计的结果，可以通过调用TURBO DEBUGGER调试工具来检查，其命令如下：

C>TD DISCH↙

在调试程序TURBO DEBUGGER的管理下，可以单步执行程序，也可以设置断点，并且在屏幕上还显示了CPU内部寄存器和标志位的内容等，为寻找程序中的错误提供了方便。(具体用法见八、调试程序的使用)

注意：在执行TD DISCH命令时，磁盘上必须已建立DISCH.EXE文件。

通过调试工具的检查，发现了错误，必须退出TD调试程序。这时再进行编辑，以修改程序中的错误，然后再进行汇编、连接、运行，直到符合要求为止。

三、建立与修改源文件

在编写好汇编源程序准备运行之前，首先必须在磁盘上建立源文件(源文件的扩展名必须是.ASM)，建立与修改源文件是通过调用编辑程序来完成的。这里简单介绍多功能窗口软　件SIDEKICK(简称SK)的编辑功能和主要命令，供上机时参考使用。

SK是一种比较新颖的多功能窗口软件，它提供了操作方便的全屏幕窗口，可以方便

地
建立、编辑源文件。进入编辑窗口的步骤如下：

首先，在DOS状态下打入SK并按回车，即

C〉SK↙

此时，SK程序便被装入内存，并且长驻内存，此后可随时调用SK，不必重新装入。

其次，按下 ALT +CTRL 键。此时，显示主选择窗口：

Sidekick	Main	Menu
F1	Help	
F2	Notepad	
F3	Calculator	
F4	Calendar	
F5	Dialer	
F6	Ascii-table	
F7	Setup	
F8	exit	

最后，进入SK的编辑窗口。

进入方式有3种：①按 F2 键；②利用光标控制↑、↓将光标移到F2. Notepad ，然后回车，③按下 ALT +N 键。

注；第1步只做一次，一旦SK程序已被调入内存后，每次要进入编辑窗口时，只要操作2、3两个步骤即可。

当进入编辑窗口之后，就可以利用该窗口的操作命令和功能键对源文件进行编辑与修改。

(一)编辑窗口的常用命令及功能键

1. 基本命令和功能键

(1)功能键

① F1 求助

显示关于编辑窗口的详细信息。

② F2 存储

将当前正在编辑的源文件(简称在编文件)按文件打开时的名称存盘，而磁盘上原来的同名文件退为后备文件作为备份(扩展名. BAK)。对于需要存盘的在编文件，在编辑完毕准备退出编辑窗口之前，一定要按 F2 键。

⑧ F3 建立或调入一个新文件。

按 F3 后，显示：

```
NEW note file
```

·输入文件名(包括扩展名)后回车，则将该文件调入窗口，等待编辑。若输入的文

件名在盘上不存在时，SK将提示用户，显示：Newfile，常用这种方法来建立新文件。

• 打*. *或直接回车，则显示该盘的全部目录文件，这时可通过移动光标来选择所需要的文件。一旦选中并按回车键后，该文件被调入窗口，等待修改。

当窗口上方显示文件目录时，只有小键盘上的键有效，所以移动光标只能用小键盘上的光标控制键。若要在前页目录区寻找所需要的文件，必须用 PgUp 键，下页则用 PgDn 键，然后再用光标控制键↑、↓、→、← 来移动光标，选择所需要的文件。

当在窗口上方显示文件目录时，如不需要编辑文件，必须按 Esc 键退出显示目录的窗口。

④ F9 扩展窗口

按一下此键，就可以利用小键盘上的方向键调节窗口，使之扩大。再按此键，窗口尺寸固定，方向键恢复原来的功能。

⑤ F10 收缩窗口(与 F9 的功能相反)

⑥ Esc 退出现行窗口

Esc 键每按一次，只能退出现行窗口，此时，屏幕上显示的是该窗口出现之前的信息，如果屏幕上所显示的窗口信息仍不是所需要的，可再次按 Esc 键，一直退到满意为止。

(2)光标控制键

小键盘上的 4 个键 ：↑、↓、→、←

Home：光标移到行首

END：光标移到行尾

PgUp：光标移到前页

PgDn：光标移到下页

Ctrl＋Home：光标移到页首

Ctrl＋END：光标移到页尾

(3)插入和删除命令

Ins (或 Ctrl＋V):插入状态打开 / 关闭。若屏幕右上角出现“Insert”，则表示可以插入字符，若出现“Overwrite”，则表示不能插入，只能覆盖，此时若要插入，则再按一次 Ins 键，直到出现“Insert”为止，这时可以在光标处插入任意个字符。

Ctrl＋N :行插入

Ctrl＋Y: 行删除

Ctrl＋Q＋Y: 删除从光标处开始，到行尾结束

Del (或 Ctrl＋G): 删除从光标所在的字符

另若在插入状态将光标移到该行的第一个字符上，再按空格键，则可达到将整行字符右移的目的。

2．Notepad 窗口的块操作命令

Ctrl＋K＋B：设定文件块起始标志

Ctrl＋K＋K：设定文件块结束标志

Ctrl＋K＋H：消去文件块标志

Ctrl+K+C：复制文件块
Ctrl+K+V：移动文件块
Ctrl+K+Y：删除文件块，且每次只能删除一个块
Ctrl+K+R：将磁盘上的指定文件读入到窗口光标处
Ctrl+K+W：写文件块到磁盘上的指定文件中
Ctrl+K+P：打印文件块

对于由“Ctrl+K+B”与“Ctrl+K+K”定义的一个文件块，可以用“Ctrl+K+V”命令移动到任意光标位置；可以用“Ctrl+K+C”命令对文件块复制；可以用“Ctrl+K+P”命令将文件块送往打印机输出。如果键入“Ctrl+K+R”，编辑窗口会向你申请一个磁盘文件名，并将指定的磁盘文件读入到编辑窗口光标处。反之，“Ctrl+K+W”执行一个逆操作，将文件块写到指定的磁盘文件中，如果某一个文件中的某一部分不再需要，就可以用“Ctrl+K+B”与“Ctrl+K+K”命令将这部分定义为一个文件块，然后用“Ctrl+K+Y”去删除。

四、源文件的汇编

要运行扩展名为.ASM 的源文件，必须先由宏汇编程序（MASM）把它汇编为目标文件，汇编后将在磁盘上建立三个文件：一个是扩展名为.OBJ 的目标文件，再此文件中，操作码已变成目标代码，但操作数据地址只是一个浮动的相对地址，而不是内存中的实际地址。第二个是扩展名为.LST 的列表文件，它是源程序行、目标代码及其在段内存放的偏移地址的一个对照表。当源程序出现语法错误时，MASM 则在错误行后面给出错误性质提示，该表可打印出来供检查用。第三个是扩展名为.CRF 的符号交叉参考文件，在执行汇编命令时，汇编程序对要不要建立这些文件以及建立时的文件名进行提问。

下面用前面显示“IBM MICRO COMPUTER”的例子来具体说明。

在 DOS 状态下，打入 MASM DISCH.ASM，系统调入宏汇编程序，然后依次提问。

```
C>MASM DISCH.ASM↙
  The Microsoft MACRO assembler, Version 1.27
  (C) Copyright Microsoft Corp 1981. 1984
   Object filename [DISCH .OBJ]: ↙
   Source listing [NUL .LST]: DISCH↙
   Cross reference[NUL .CRF]: DISCH↙
```

其中，第一个提示询问目标文件名，括号内的信息为系统规定的默认文件名，一般直接按回车，表示采用默认文件名，接着提问是否要建立汇编列表文件，如要，打入文件名；否则，按回车键，最后提问是否要建立交叉参考文件，如要，打入文件名，否则按回车。提问后汇编程序开始进行汇编，如发现源程序中有语法错误，则列出有错误的语句和错误代码，并指出错误的类型及错误总数。此时，可参考错误信息表(见附录二)，分析错误原因，修改源程序。

汇编后建立的. LST 文件可打印，如下所示：

```
C>TYPE DISCH .LST↙
 The Microsoft MACRO Assembler, Version 1. 27    Pa9e    1-1
                                                        09-11-93
 1    0000    data segment
 2    0000    49 42 4D 20 4D 49    buf db" IBM MICRO COMPUTER $"
 3            43 52 4F 20 43 4F
 4            4D 50 55 54 45 52
 5            20 24

 6    0014                         data ends
 7    0000                         code segment
 8                                        assume cs:code, ds:data
 9    0000    B8------R            start: mov ax data
 10   0003    8ED8                       mov ds, ax
 11   0005    8D 16 0000 R               lea dx, buf
 12   0009    B4 09                      mov ah, 09h
 13   000B    CD 21                      int 21h
 14   000D    B4 4C                      mov ah, 4ch
 15   000F    CD 21                      int 21h
 16   0011                         code  ends
 17                                      end Start
The Microsoft MACRO Assemble,, Version 1. 27    Page    Symbols
                              -1                 09-11-93
Segments  and groups
    Na m e                      Size   align    combine    Class
  CODE.,.,........              0011    PARA    NONE
  DATA ......,....              0014    PARA     NONE
  Symbols:
      N a m e                     Type    Value    Attr
  BUF...........                  L BYTE   0000    DATA
  STArt.........                  L NEAR   0000    CODE
  WarninSSevere
  Errors Errors
  0    0
```

汇编后建立的.CRF 文件是不能单独使用的，若需要了解源程序中的符号(包括变量)在定义和引用时的情况，还要调用 CREF 文件对. CRF 文件进行处理，生成. REF 文件后，

才能打印(或显示)输出。过程如下：

C>CREF↙

Microsoft Cross Reference

Version 1. 00，Copyright(C)1981，82 by Microsoft lnc，

Creffilename[. CRF)：DISCH：↙

List filentime[DISCH. REF]：↙

在 DOS 状态下，打入 CREF，系统调入 CREF 程序。CREF 程序运行时，首先提问要处理的文件名。打入文件名后，接着又提问. RET 文件名，如果是默认的文件名，就按回车。于是就建立了一个扩展名为.REF 的文件，然后返回 DOS 系统。在 DOS 状态下输入打印命令：

C>TYPE DISCH. REF↙

显示信息

Symbol Cross Reference	(#is definition)	Cref-1
BUF.....	2#	11
CODE	7#	8 16
DATA.....	1#	6 8 9
START.....	9#	17

其中，有#号者是此符号被定义时的语句行号，后面是引用此符号时的语句行号。该文件对于阅读、调试大型多模块程序是很有帮助的，但对简单的程序，则不必建立该文件。所以，如果不需要建立列表文件(. LST)和交叉参考文件(. CRF)，可用如下命令形式：

C>MASM 文件名；↙

例如：

C>MASM DISCH. ASM；↙

此操作表示省略对后续提示的回答，仅取缺省值。

五、目标文件的连接

汇编以后产生的目标文件还必须经过连接，才能生成可执行文件(扩展名. EXE)。连接工作是在 DOS 状态通过调用 LINK 程序来完成的。其过程如下：

C>LINK↙

Microsoft 8086 Object Linker

Version 3. 02(C)Copyright Microsoft Corp 1983，1984，1985

Object Modules[. oBJ]：DISCH↙

Run File[DISCH. EXE]：↙

List FUe[NUL. MAP]：DISCH↙

Libraries[. LIB]：↙

在 DOS 状态，键入 LINK 后，屏幕上将显示提问行。首先询问要连接的目标文件名，

此时键入文件名(此处以 DISCH 为例)作为回答。如果有多个目标文件要连接，应一次键入并且各文件名之间用“+”号隔开。接着提问可执行文件名，一般按回车表示采用括号内规定的默认文件名。然后提问是否要建立地址分配文件，键入文件名再回车表示要建立，否则直接回车。最后提问是否要用到库文件，如果没有，直接回车。如果用户自己建立了库文件，则键入库文件名。(建立库文件的方法后面介绍)提问结束后，开始进行连接，若有错，则显示错误信息。此时还必须再调用编辑程序以修改源程序，然后重新汇编、连接，直至无错。

连接产生的. MAP 文件，可以打印出来，如下所示:

```
C〉TYPE DISCH. MAP↙
Warning:no stack segment
Start    Stop    Length    Name    Class
00000H    00013H  00014H    DATA
00020H    00030H  00011H    CODE
Program entry point at 0002:0000
```

可以看出. MAP 文件提供了地址分配的信息，DATA 段的起始地址是 00000H，结束地址是 00013H，字节数 14H。其它段的情况依此类推。对于简单的程序，可不建立该文件。如果不需要生成. MAP 文件，也不使用库文件，可采用以下命令形式来连接：

```
C>LINK DISCH; ↙
```

这种命令形式同时还免去用户对提问的回答，直接生成可执行文件 DISCH. EXE。

多个文件连接的命令形式：

```
C>LINK MAIN+GETCHR+LISCHR↙
```

六、扩展名. EXE 执行

通过 LINK 连接后生成的可执行文件(扩展名. EXE)，可以在 DOS 状态下，直接键入文件名(不必要扩展名. EXE)就可以执行程序。例如：

```
C>DISCH↙
```

其中 C>是 DOS 提示符，DISCH 是可执行的文件名，↙键入后，DOS 将执行程序。

七、子程序库的建立方法

在处理较大而且很复杂的设计任务时，往往采取模块化结构化的设计方法。对其中公共部分常设计成独立的公用子程序模块，供大家调用。这些公用子程序必须汇编成目标模块，建立在一个“子程序库”(扩展名. LIB)中。

建立子程序库是通过调用库管理程序 LIB 实现的，并且还可修改子程序库。在 DOS 状态下，打入“LIB↙”之后，DOS 将把 LIB 装入内存并向用户提问，用户根据要求键入相应的回答。LIB 提示信息及其回答如表 3-1 所示，回答时可用命令字符，如表 3-2 所示。

表 3-1　LIB 的提示信息及回答

提示信息	提示信息
library name:	欲进行操作的库名(缺省扩展名为. LIB)
operation:	命令字符及模块名或目标文件名
list file:	交叉参考列表文件名(缺省：NVL，无交叉参考列表文件)

表 3-2　库管理命令字符

命令字符	功能
+	把目标代码文件作为最后一个模块加入库中
-	从库中删除一个模块
*	从库中取出模块，写入目标文件中(库中仍保留)
;	剩下的提示取缺省值
&	提示信息在一行内回答不下时，键入此字符将另起一行
CTRL_C	终止库管理操作

下面举例说明。

假设欲从键盘键入一个字符，然后送至显示器显示出来。对于这个问题，可以把键盘接收部分作为一个子程序，文件名为 GETCHR. ASM。显示部分作为一个子程序，文件名为 LISCHR. ASM。主程序用来调用这两个子程序，主程序名为 MAIN. ASM。

其步骤如下：

①建立源程序 GETCHR. ASM、LISCHR. ASM、MAIN. ASM。

```
GETCHR. ASM
PUBLIC GETCHR
CODE    SEGMENT PUBLIC’ CODE’
ASSUME CS：CODE，DS：CODE
GETCHR PROC NEAR
       JMP START
       AHL  DB ?
START: MOV AHL，AH
       MOV AH，01H
       INT 21H
       MOV AH，AHL
       RET
GETCHR  ENDP
CODE    NEDS
        END
```

```
LISCHR. ASM
PUBLIC LISCHR
CODE SEGMENT PUBLIC 'CODE'
     ASSUME CS:CODE, DS: CODE, SS: CODE
LISCHR PROC NEAR
          PUSH AX
          MOV AH, 02H
          INT 21H
          POP AX
          RET
 LISCHR  ENDP
 CODE    ENDS
         END

  MAIN. ASM
  CODE SEGMENT PLIBLIC 'CODE'
       ASSUME CS: CODE, DS: CODE
       EXTRN GETCHR: NEAR, LISCHR:NEAR
  ENTRY:JMP MAIN
  MAIN PROC NEAR
  CONT:  CALl GETCHR
         CMP AL, 0DH
         JNE CONT
          MOV DL, 0AH
          CALL LISCHR
          JMP CONT
   MAIN   ENDP
   CODE   ENDS
          END ENTRY
```

②将子程序源文件分别进行汇编，生成目标文件；GETCHR. OBJ，LISCHR. OBJ。

③调用库管理程序 LIB 建立子程序库，库名为 EXA. LIB，并将两个子程序的目标文件加入库中，既可逐个加入，也可一次加入。如果逐个加入，操作过程如下；

```
C>LIB↙
MiCrosoft Library Manager
Version 3. 00(C)Copyright Microsoft Corp 1983, 1984, 1985
Library name EXA↙
Library does not exist
```

Create ? Y↙

Operations：+GETCHR↙

List file: ↙

C>LIB↙

MiCrosoft Library Manager

Version 3. 00(C)Copyright Microsoft Corp 1983，1984，1985

Library name : EXA↙

Operations：+LISCHR↙

List file：↙

Output library: ↙

如果一次加入，操作过程如下：

C>LIB↙

Microsoft Library Manager

Version 3. 00(C)Copyright MicrosoftCorp 1983，1984，1985

Library name：EXA↙

Library does not exist. Create ? Y↙

Operations: +GETCHR+LISCHR ↙

List file：↙

上述建库的第一种操作中，最后一行提示信息“Output library:”是询问库中内容是否复制到另一个库中，若是，则回答另一个库名，否则，直接按回车。一般直接按回车。

库文件建立后，库中的子程序可以方便地被其它程序调用。此例中由主程序调用，主程序 MAIN. ASM 汇编后生成目标文件 MAIN. OBJ，然后可按如下过程进行连接。

C>LINK↙

Microsoft 8086 Object Linker

Version 3. 20(C)Copyright MicrosoftCorp 1983，1984，1985

Object Modules[. OBJ]：MAIN↙

Run File[MAIN. EXE]：↙

List File[NUL. MAP]：↙

Libraries[. LIB]: EXA↙

连接后得到可执行文件 MAIN. EXE。

八、调试程序 Turbo Debugger 及其使用

1、简　介

Turb oDebugger 是一个先进的源程序级调试器，(Source—Level debugger)。它具有多个互相重叠的窗口、下拉式和弹出式菜单，为用户提供了一个快速的交互式环境。

Turbo Debugger 有一个方便的全局菜单系统，进入 Turbo Debugger 后，就可以使用左右箭头键移动亮度光标来访问屏幕顶部排列的主菜单选择项。每个选择项都可以引

出一个下拉式菜单(pull-down menu)，通过下拉式菜单，用户可以：①执行一个命令；②打开弹出式菜单(pop-up ment)，当选择后面带有一个箭标(>)的条目时，就会出现弹出式菜单；③打开对话框(dialogbox)。当选择后面带有一个对话框图符(……)的条目时，就会出现刘‘话框。

不管在 Turbo Debugger 的什么地方，屏幕底部都有一个相应的参考行，该行提供给用户在当前上下文中一目了然的击键或者菜单命令的有关信息。

在使用 Turbo Debugger 调试源程序前，必须事先把源文件编辑、汇编、连接生成可执行的扩展名为. EXE 的文件，才能使用 Turbo Debugger。因为 Turbo Debugger 没有内在编辑器来建立与修改源程序。同样也不能编译用户的源程序。

2．运行 Turbo Debugger

为了使用 Turbo Debugger 调试程序，在 DOS 提示符下简单地打入 TD 及文件名，然后按 Enter 键，于是 Turbo Debugger 就装载用户程序并显示源代码，这样用户就可以一条语句一条语句地执行程序了。

命令格式:

C>TD 文件名↙

如果用户简单地输入 TD 并按巨回键，TurboDebugger 使用其缺省启动。命令格式：

C>TD↙

一般在熟悉汇编语言指令时，可用这种启动方式。

3．CPU 窗口

当用户输入 TD 并按 Enter 键或使用主菜单选择项的 View / CPU 命令都可以创建 CPU 窗口，如图 3-13 所示。根据当前装载的用户程序，CPU 窗口将显示对应的代码、数据和堆栈位置。把 CPU 的整个状态反映在窗口中，利用该窗口可以：

(1)了解和改变程序代码或数据的位或字节。

(2)在代码区内使用嵌入汇编，对程序进行临时性修改。

(3)存取任何数据结构下的字节内容，并以多种格式显示或改变它。

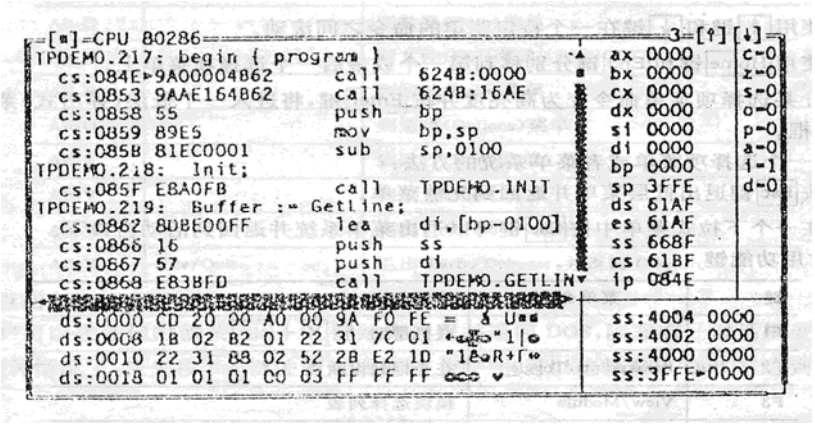


图 3-1　CPU 窗口

CPU 窗口共分 5 个区，左上去（代码区）显示源程序行和相应的反汇编程序代码，中区（寄存器）显示 CPU 寄存器的内容，右上区（标志区）显示 CPU 标志的状态，左下区（数据区）显示用户指定的一块内存数据(十六进制表示)，右下区(堆栈区)显示堆栈内容。代码区中，箭标(>)指示出当前程序位置(CS：IP)，堆栈区中，箭标(>)指示出当前堆栈指针(SS：SP)的位置。在这几个区中，可通过 Tab 或 Shift+Tab 键在区间移动。

4．使用菜单

两种类型菜单的比较。

全局菜单；

(1)可以通过按 F10 键和方向键或菜单名的第一个字母来访问。

(2)总可以从显示在屏幕顶部的选择项上得到。

(3)它们的内容从不改变。

局部菜单；

(1)可以用 Alt+F10 键或， Ctrl+F10 键来访问。

(2)菜单的位置与内容取决于用户所在的窗口或窗口区以及光标的位置。

(3)不同局部菜单之间的内容不同。

有以下几种途径可以打开选择项上的菜单：

(1)按 F10 键，用 → 键或者 ← 键移到想要的选择项菜单下，并按 Enter 键。

(2)按 F10 键，然后再按选择项菜单名的第一个字母(空格，F,V，R，B,D，O，W，H)。

(3)按 Alt 和任何选择项菜单命令的第一个字母(空格、F、V、R、B、D、O、W、H)。

例如，不管在系统的什么地方，按 Alt+F 键将进入 File 菜单。

一旦进入全局菜单系统，就可以用下面的方法在其中移动:

(1)使用 → 键和 ← 键可以从一个下拉式菜单移动到另一个。例如，在 File 菜单下时，按 → 键将进入 View 菜单。

(2)使用 键和 键在一个特定菜单的命令之间滚动。

(3)使用 Home 键和 End 键分别移到第一个或最后一个选择项菜单上。

(4)让某选择项菜单命令变为高亮度并按 Enter 键,将进入一个低层(弹出式)菜单或者一个对话框。

退出一个选择项菜单或者菜单系统的方法；

(1)按 Esc 键退出低层菜单并返回到先前菜单。

(2)在一个下拉式菜单中按 Esc 键可以退出菜单系统并返回到活动窗口下。

5．常用功能键

键	菜单命令	功　能
F1		联机帮助
F2	Breakpoints / Toggle	在光标处设断点
F3	View / Module	模块选择列表

F4	Run / Goto Cursor	运行到光标位置
F5	Window / Zoom	放大 / 缩小当前窗口
F6	Window / Next	Window 到下一个窗口
F7	Run /Trace IntO	执行单行源代码或单条指令
F8	Run / Step Over	执行单行源代码或单条指令，跳过“调用语句”(或指令)
F9	Run / Run	运行程序
F10		激活或退出菜单条
Alt-F1	Help / Previous Topic	显示最直接的帮助屏幕
Alt-F2	Break points / At	在一地址处设置断点
Alt-F3	Window / Close	关闭当前窗口
Alt-F4	Run / Back Trace	返向执行程序
键	菜单命令	功 能
AlL-F5	Window / User SCteen	显示用户程序屏幕
Alt-F6	Window / Undo Close	重新打开刚被关闭的窗口
Alt-F7	Run / Instruction Trace	执行单条指令
Alt-F8	Run / Until Return	执行到从子程序返回
Alt-F9	Run / Execute TO	运行到一特定地址
Alt-F10		激活窗口局部菜单
Alt1～9		到有相应数字标号的窗口
Alt-Space		到系统菜单
Alt-B		到断点(Breakpoints)菜单
Alt-D		到数据(Data)菜单
Alt-F		到文件(File)菜单
Alt-H		到帮助(Help)菜单
Alt-O		到选项(Options)菜单
Alt—R		到运行(Run)菜单
Alt-V		到浏览(View)菜单
Alt-W		到窗口(Window)菜单
Alt-X	File / Ouit	退出 Turbo Debuger，并返回 DOS

6．返回 DOS

用户可以在任何时候按 Alt + X 键结束过程并返回 DOS，除非有一对话框处于活动状

态(在这种情况下，应先按 Esc 键关闭对话框)。用户也可以选择 File / Quit 来达到同样的目的。

附录二　汇编错误信息

如果在汇编源程序过程中发现错误，宏汇编程序将会显示错误行、错误性质的编号及错

误原因等信息。具体内容如下(其中左边的序号是错误性质的编号)：

0　过程、段、结构、宏、重复等的嵌套出错

1　在语句行中有额外(多余)的字符存在

2　寄存器已定义。当汇编程序有逻辑错误时，才出现这个错误

3　未知的符号类型

4　符号再定义。指在第二遍扫描时连续地定义一个符号

5　符号重复定义

6　相位错误。指一个符号在两遍扫描中的偏移不同，这时，可以在启动汇编时指定 / D 参

数，分别列出两遍扫描的汇编列表文件对照检查

7　试图在已存在的 ELSE 子句中再定义一个 ELSE 子句

8　在 ENDIF 或 ELSE 之前没有条件汇编伪指令

9　使用的符号没有定义

10　语法错误

11　指定的类型在长度上是不能接受的

12　给出的不是程序所要求的组名

13　在第一遍扫描就应知道其值

14　PUBLIC 伪指令中符号类型使用不合法

15　试图定义一个与前面的定义相矛盾的符号

16　使用的符号是保留字

17　向前引用的符号必须是在第一遍扫描中定义过的

18　此处的操作数应是寄存器，但使用的是符号而不是寄存器

19　指定的寄存类型并不是指令或伪指令所要求的

20　必须是段或组，而不是其它的符号

21　试图使用一个具有段属性的符号，但这个符号没有段属性

22　必须是符号的类型

23　试图说明一个符号为外部符号，但它已经被定义为局部符号了

24　段参数被改变

25　SEGMENT 语句中定位 / 组合方式不正确

26　语句中引用了一个多次定义过的符号

27　需要一个操作数

28　需要的是一个操作符

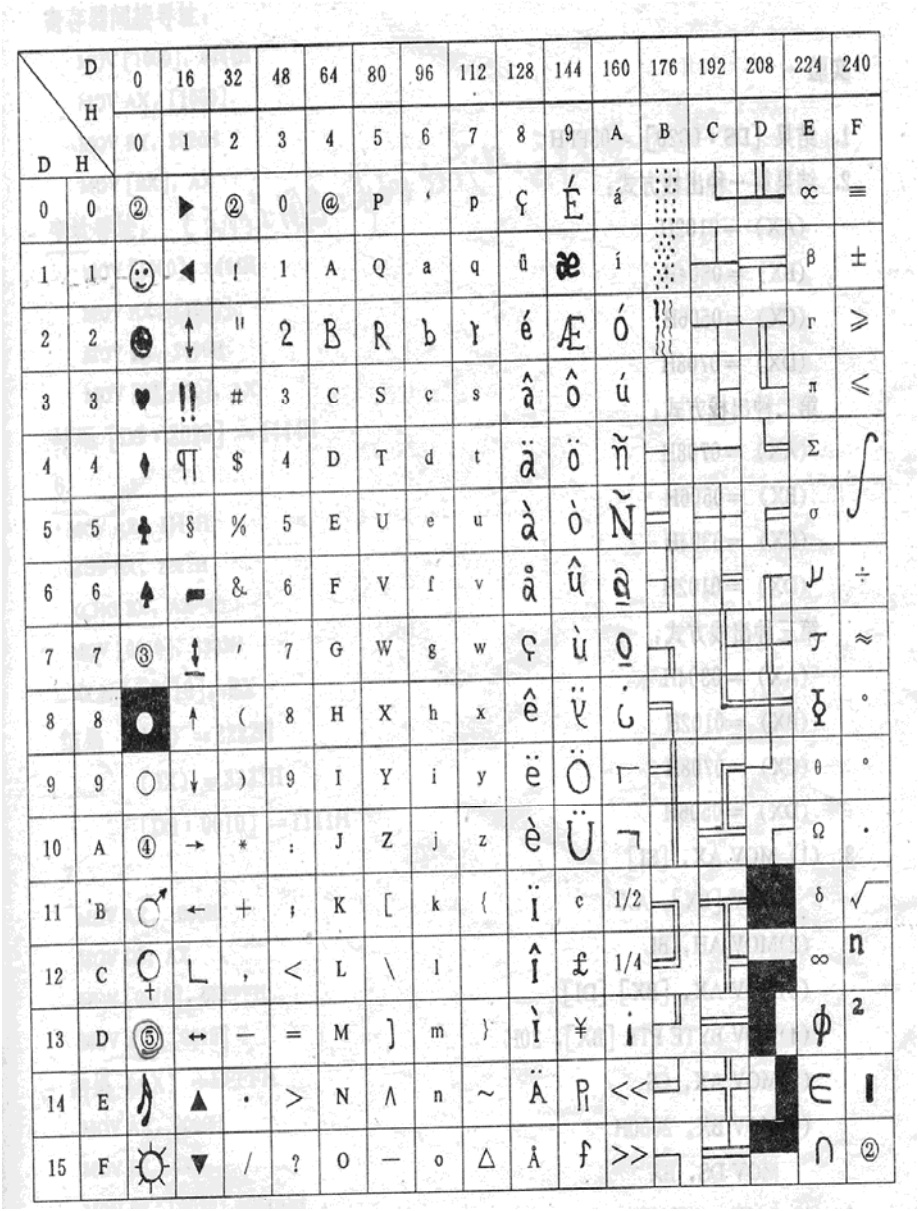
29　给出的是一个以零作除数的表达式或表达式的运算结果溢出

30　移位表达式中的移位次数是负值

31 操作数的类型必须一致
32 非法使用外部符号
33 必须是记录的字段名
34 应该是一个记录或字段名
35 操作数必须有长度
36 必须是变量、标号或常数
37 必须是一个结构字段名
38 左边的操作数必须有段信息
39 加法指令的非法使用
40 减法指令的非法使用
41 需要的是普通类型的操作数，例如变量，标号等
42 需要的是一个常量
43 SEG 算符后面的符号没有段属性
44 必须是与数据段相关的项
45 必须是与代码段相关的项
46 试图重复使用基址寄存器
47 试图重复使用变址寄存器
48 语句中需要的是变址或基址寄存器，但给定的却是别的寄存器
49 在语句中使用了 8088 所没有的寄存器
50 数值超出范围
51 操作数不在当前段中，不能存取
52 错误的操作数类型
53 相对转移超出允许的范围(-128～+127)
54 变址的位移量必须是常量
55 寄存器字段值是非法的
56 这条语句的操作数不能是立即数
57 引用项的长度是非法的
58 不能使用字节寄存器
59 试图非法使用 CS 寄存器
60 寄存器只能是 AX 或 AL
61 段寄存器的使用不合法
62 试图转移到不可能到达的标号
63 在双操作数指令中，两个操作数的组合不正确。
64 试图在一个不同的代码段地址中执行 NEAR 转移或调用。
65 标号不能有跨段前缀算符
66 在指令前缀的后面没有正确的操作码
67 ES 段不能被取代
68 用这个段寄存器不能寻址该变量

69 试图在段外产生代码
70 在起始边界为 BYTE 的段不能用 EVEN 伪操作
71 目前不使用这个信息
72 DUP 前的重复数值是非法的(只能是大于零的整数)
73 符号已被说明是外部符号
74 DUP 的嵌套太长
75 问号的使用不合适。例如?+5
76 超出定义值的个数
77 只有初始值表是合法的
78 STRUC 中的伪指令是非法的
79 用 DUP 取代初值是非法的
80 该字段不能被取代
81 取代初值的数类型不对
82 寄存器不能向前引用
83 EQU 等价名的循环链
84 小汇编(ASM)没有提供的功能
85 没有 END 伪指令

附录三　IBM—PCASCⅡ码符表



D H	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
D H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	②	▶	②	0	@	P	‘	p	ç	É	á	▒	└	┴	∝	≡
1	1	☺	◀	!	1	A	Q	a	q	ü	æ	í	▒	┬	├	β	±
2	2	☻	↕	"	2	B	R	b	r	é	Æ	ó	▓	─	┼	Γ	⩾
3	3	♥	‼	#	3	C	S	c	s	â	ô	ú	│	├	╞	π	⩽
4	4	♦	¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	╟	Σ	∫
5	5	♣	§	%	5	E	U	e	u	à	ò	Ñ	═	┌	╚	σ	⌠
6	6	♠	▬	&	6	F	V	f	v	å	û	ª	╞	┐	╔	μ	÷
7	7	③	↨	'	7	G	W	g	w	ç	ù	º	╟	│	╩	τ	≈
8	8	◘	↑	(	8	H	X	h	x	ê	ÿ	¿	╚	┘	╦	Φ	°
9	9	○	↓	)	9	I	Y	i	y	ë	Ö	⌐	╔	┌	╠	Θ	°
10	A	④	→	*	:	J	Z	j	z	è	Ü	¬	╩	└	═	Ω	·
11	B	♂	←	+	;	K	[	k	{	ï	¢	1/2	╦	┴	╬	δ	√
12	C	♀	∟	,	<	L	\	l	|	î	£	1/4	╠	┬	∞	n
13	D	♪	↔	-	=	M	]	m	}	ì	¥	¡	═	─	╧	φ	2
14	E	♫	▲	.	>	N	^	n	~	Ä	₧	«	╬	┼	╨	∈	■
15	F	☼	▼	/	?	O	_	o	⌂	Å	ƒ	»	╧	┴	╤	∩	②

说明：(1)ASCII 码 128～255 的输入方法为：在按下 ALT 键的同时，再在右端小键盘上输入相应的十进制代码。

(2)表中②代表空格，3 代表响铃，4 代表换行，5 代表回车

(3)表上端横排为高位码，表左端纵排为低位码

附录四 PC—DOS 系统功能调用

功能号	功　能	入　口　参　数	出　口　参　数
00H	程序结束	(AH)=0 (CS)=程序前缀区 (PSP)段地址	无
01H	键盘输入	(AH)=1	(AL)=输入字符
02H	显示字符输出	(AH)=2 (DL)=欲输出字符	无
03H	异步通讯口输入	(AH)=3	(AL)=输入字符
04H	异步通讯口输出	(AH)=4 (DL)= 欲输出字符	无
05H	打印机输出	(AH)=5 (DL)=欲输出字符	无
06H	直接控制台Ⅰ/O (不检查Break键)	(AH)=6 若(DL)= 0FFH 表示输入 若(DL)≠0FFH 表示输出 (DL)为欲输出字符	当(DL)= 0FFH，若有输入，则 (AL)=输入字符，否则(AL) = 0
07H	无回显直接控制台输入(不作字符检查)	(AH)=7	(AL)=输入字符
08H	无回显键盘输入	(AH)=8	(AL)=输入字符
09H	显示字符串	(AH)=9 (DS：DX)=指向字符串首址 (字符串以“$”结尾)	无
0AH	输入字符串	(AH) =0AH (DS：DX)=输入缓冲区的首址。其中第一字节为所能保存的最大字符数；第二字节为接收的字符个数，不含回车符；第三字节存放输入的第一个字符的 ASCII 码	无
0BH	取键盘输入状态	(AH)=0BH	若(AL)=00H 无键入 (AL)=0FFH 有键入
0CH	清键盘缓冲后输入	(AH)=0CH (AL)=功能号(01，06，07，08 或 0AH)	同 01，06，07，08，0AH 功能

功能号	功　能	入　口　参　数	出 口　参　数
0DH	刷新DOS磁盘缓冲区	(AH)=0DH	无
0EH	选择当前盘(缺省)	(AH)=0EH (DL)=盘号	(AL)=系统中盘的数目
0FH	打开文件	(AH)=0FH (DS：DX)＝FCB首址	若(AL)=00成功 (AL)= 0FFH失败
10H	关闭文件	(AH)= 10H (DS：DX)＝FCB首址	同上
11H	查找第一个匹配文件	(AH)= 11H (DS：DX)=FCB首址	若(AL)=00找到 (AL)= 0FFH未找到
12H	查找下一个匹配文件	(AH)=12H (DS:DX)=FCB首址	同上
13H	删除文件	(AH)=13H (DS：DX)= FCB首址	(AH)= 00成功 (AL)=0FFH失败
14H	顺序读一个记录	(AH)= 14H (DS：DX)= FCB首址	若(AL)=00成功 (AL)=01文件结束 (AL)=02缓冲区不够 (AL)=03缓冲区不满
15H	顺序写一个记录	(AH)= 15H (DS：DX)= FCB首址 (DTA缓冲区已设置)	若(AL)= 00成功 (AL)= 0FFH盘满
16H	建立文件(新的或老的)	(AH)＝16H (DS：DX)=FCB首址	若(AL)＝00成功 (AL) =0FFH 目录区满
17H	更改文件名	(AH)= 17H (DS；DX)＝FCB首址 (DS：DX +17)= 新文件名首址	(AL)= 00成功 (AL)= 0FFH失败
18H	DOS保留		
19H	取当前盘号	(AH)= 19H	(AL)＝盘号
1AH	设置磁盘传送缓冲区(DTA)	(AH)= 1AH (DS:DX)= DTA首址	无

功能号	功　能	入　口　参　数	出　口　参　数
1BH	取文件分配表(FAT) 信息(当前盘)	(AH)=1BH	(DS：BX)=盘类型字节地址 (DX)=FAT 表项数 (AL)=每簇扇区数 (CX)=每扇区字节数
1CH	取指定盘的文件分配表(FAT)信息	(AH)=1CH (DL)=盘号	同上
1DH　～ 20H	DOS 保留使用		
21H	随机读一个记录	(AH)=21H (DS：DX)=FCB 首址 (DTA 已设置)	若(AL)=00 成功 (AL)＝01 文件结束 (AL)=02 缓冲不够 (AL)=03 缓冲不满
22H	随机写一十记录	(AH)=22H (DS：DX)=FCB 首址 (DTA 已设置)	(A1)=00 成功 (AL)=0FFH 盘满
23H	取文件长度	(AH)=23H (DS：DX)=FCB 首址	若(AL)＝00 成功 长度在 FCB 中 (AL)=0FFH 失败
24H	置随机记录号	(AH)=24H (DS：DX)=FCB 首址	无
25H	置中断向量	(AH)=25H (DS：DX)=入口地址 (AL)=中断码	无
26H	建立一个程序段	(AH)=26H (DX)=段号	无
27H	随机读若干记录	(AH)=27H (DS：DX)=FCH　首址 (CX)=记录数 (DTA 已设置)	若(AL)=00 成功 (AL)=01 文件结束 (AL)＝02 缓冲不够 (AL)=03 缓冲不满
28H	随机写若干记录	(AH)=28H (DS：DX)=FCH　首址 (CX)=记录数 (DTA 已设置)	若(AL)=00 成功 (AL)=0FFH 盘满

功能号	功　能	入　口　参　数	出　口　参　数
29H	建立 FCB	(AH)=29H (DS：51)=字符串首址(文件名) (ES：D1)=FCB 首址 (AL)=0EH 非法字符检查	(ES:DI)=格式化后的 FCB 首址 (AL)=00 标准文件 (AL)=01 多义文件 (AL)=0FFH 非法盘符
2AH	取日期	(AH)=2AH	(CX：DX)=日期 CX=年(1980—2099) DH=月(1—12) DL=日(1—31) AL=星期几
2BH	置日期	(AH)=2BH (CX：DX)=日期(同 2AH 出口参数)	若(AL)=0 成功 (AL)=0FFH 失败
2CH	取时间	(AH)=2CH	(CX：DX)=时间 CH=小时(0—23) CL=分(0—59) DH=秒(0～59) DL=百分秒(0～99)
2DH	置时间	(AH)=2DH (CX：DX)=时间 (同 2CH 出口参数)	若 AL=00 成功 (AL)=0FFH 失败
2EH	置写校验状态	(AH)=2EH (AL)=状态 (DL)=0	若 AL=00 成功 (AL)=0FFH 失败
2FH	取磁盘传送缓冲区首址(DTA)	(AH)=2FH	(ES：BX)=缓冲区(DTA)首址
30H	读 DOS 版本号	(AH)=30H	(AL)=版本号 (AH)=发行号
31H	程序常驻退出	(AH)=31H (AL)=退出码 (DL)=程序长度	无
32H	DOS 保留		

功能号	功　能	入　口　参　数	出　口　参　数
33H	置/取Break检查状态	(AH)=33H (AL)=0 取状态 (AL)=1 置状态 (DL)=状态 00——表示关 01——表示开	(DL)=状态 (当AL＝0时)
34H	DOS保留		
35H	取中断向量	(AH)=35H (AL)=中断码	(ES：BX)=入口地址

IBM—PCDOS 中断向量一览表

中断码	功 能 说 明	中断码	功 能 说 明
0H	除法错中断	1FH	显示图形字符
1H	单步中断	20H	程序正常结束
2H	不可屏蔽中断	21H	系统功能调用
3H	断点中断	22H	程序结束
4H	溢出中断	23H	Ctrl—C 处理
5H	屏幕拷贝	24H	严重错误处理
6H～7H	未使用	25H	读盘(绝对)
8H	定时中断	26H	写盘(绝对)
9H	键盘中断	27H	程序驻留退出
AH	未使用	28H～2EH	DOS 保留
BH	异步通讯口 2	2FH	假脱机打印
CH	异步通讯口 1	30H～3FH	DOS 保留
DH	硬盘中断	40H	软盘 I/O
EH	软盘中断	41H	硬盘参数
FH	打印机中断	42H～5FH	系统保留
10H	屏幕显示	60H～6FH	用户保留
11H	检测系统配置	70H	IRQ8
12H	检测存储器容量	71H	IRQ9
13H	硬盘 I／O	72H	IRQ10
14H	异步通讯口 I／O	73H	IRQ11
15H	盒带 I／O	74H	IRQ12
16H	键盘 I／O	75H	IRQ13
17H	打印机 I／O	76H	IRQ14
18H	ROM BASIC(热启动)	77H	IRQ15
19H	系统自举(冷启动)	78H～7FH	未使用
1AH	日时钟参数	80H～F0H	BASIC 占用
1BH	键盘 Ctrl 一 Break	F1H～FFH	未使用
1CH	中断		
1DH	间隔时钟		
IEH	显示器初始化参数		
	软盘参数		

注：下面是对 10H～1CH 和 20H 一 27H 部分中断的说明，INT 21H 的系统功能调用在附录 E 中列出。

1．屏幕显示(INT 10H)

功　能	入　口　参　数	出　口　参　数
置显示模式	(AH)=0 (AL)=模式编码 字符模式 0—40 X 25　黑白 1—40 X 25　彩色 2—80 X 25　黑白 3—80 X 25　彩色 图形模式 4—320 X 200　彩色 5—320 X 200　黑白 6—640 X 200　黑白 10—640 X 350　EGA	无
设置光标大小	(AH)=1 $(CH)_{4\sim0}$=光标起始线 $(CL)_{4\sim0}$=光标终止线	无
置光标位置	(AH)=1 (BH)=页号(图形模式为 0) (DH)＝行号 (DL)=列号	无
读当前光标位置	(AH) =3 (BH)=页号(图形模式为 0)	(DH)=行号，(DL)=列号 (CX)=当前光标大小
读光笔位置	(AH)=4	(AH)=0，未按光笔开关 (AH)=1，按光笔开关 且下列寄存器值有效： (DH)=行号，(DL)=列号 图形方式时： (CH)=扫描线号(0～199) (BX)=像素列号(0～319／639)
置当前显示页(字符方式有效)	(AH)=5 (AL)＝页号 模式 0，1=0～7 模式 2，3=0～3	无

功　能	入　口　参　数	出　口　参　数
上滚当前页	(AH)=6 AL)=上滚行数，0 为整个屏幕 (CH,CL)=滚动区域左上角的行、列号 (DH,DL)=滚动区域右下角的行、列号 (BH)=空白行的属性	无
下滚当前页	(AH)=7 (AL)=下滚行数(从窗口顶部算起空白的行数) 0 为整个窗口空白 其他参数同上滚	无
读当前光标位置处的字符及属性	(AH)=8 (BH)=页号(字符方式有效)	(AL)=读出的字符 (AH)=字符的属性
写字符及属性到当前光标位置处	(AH)=9 (A1)=欲写字符 (BL)=字符属性 (CX)=字符计数 (BH)=页号(字符方式有效)	无
写字符到当前光标位置处(属性不变)	(AH)=10 (AL)=欲写字符 (CX)=字符计数 (BH)=页号(字符方式有效)	无
置彩色调色板	(AH)=11 (BH)=调色板色别值 (BL)= 彩色值	无
在屏幕上写点	(AH)=12 (DX)=线号，(CX)＝列号 (AX)=点的颜色	无

功　能	入　口　参　数	出　口　参　数
在屏幕上读点	(AH)=13 (DX)=线号，(CX)=列号	(AL)=点的颜色
写字符到当前光标位置，且光标前进一格	(AH)=14 (AL)=欲写字符 (BL)=字符颜色 (BH)=页号	无
读当前显示状态	(AH)=15	(AH)=屏幕上字符列数 (AL)=当前显示模式 (BH)=当前显示页
保　留	((AH)=16 —18)	
写字符串	(AH)=19 (ES：BP)=指向字符串 (CX)=字符串的长度 (DX)=起始的光标位置 (BH)=页号 (AL)=0，BL=属性 (字符，字符，……)．光标不移动 (AL)=1，BL=属性 (字符，字符，……)，光标移动 (AL)=2,(字符,属性，……)．光标不移动 (AL)=3，同上．光标移动	无

附录五　编程解答

实验一

实验二

实验三

编程练习

1.

```
  DATA  SEGMENT
  NUM DB 22, 46, 32, 72, 84, 16
  MAXN DB ?
  DATA ENDS
  MAIN SEGMENT
        ASSUME CS: MAIN, DS: DATA
  START: MOV AX, DATA
         MOV DS, AX
         LEA BX,NUM
         MOV CX, 05
         MOV AL , [BX]
  AGAIN: INC BX
         CMP AL, [BX]
         JNBE NEXT
         MOV AL. [BX]
  NEXT:  LOOP AGAIN
         MOV MAXN, AL
  MAIN ENDS
       END START
```

结果[DS:0006]= 54H

2,

```
  DATA SEGMENT
    NUM DB 38, 55, 26, 12, 23
    SUM DB ?
  DATA ENDS
  CODE  SEGMENT
    ASSUME CS:CODE, DS:DATA
  START: MOV AX, DATA
         MOV DS, AX
```

```
        MOV CX, 05
        LEA BX, NUM
        SUB AL, AL
NEXT:   ADD AL, [BX]
        INC BX
        LOOP NEXT
        MOV SUM, AL
 CODE ENDS
      END START
```

结果：[DS:0005]= 9AH

3.

```
  DATA SEGMENT
  NUM DB 58, 25, 45, 73, 64, 43
  SUM DW ?
  DATA ENDS
  OODE  SEOMENT
        ASSUME  CS:CODE, DS:DATA
  START: MOV AX, DATA
         MOV DS, AX
         LEA DX,NUM
         MOV CX, 6
         SUB AX, AX
  NEXT:  ADD AL [BX]
         ADC AH, 0
         INC BX
         LOOP NEXT
         MOV SUM, AX
  CODE ENDS
       END START
```

结果：[DS:0006]＝0134H

4.

```
DATA    SEGMENT
    DATA1 DB 03, 09, 00, 06, 05
    DATA2 DB 05
    SUM DB 6 DUP(00)
DATA  ENDS
CODE SEGMENT
     ASSUME CS:CODE, DS:DATA
```

```
START: MOV AX, DATA
       MOV DS, AX
       MOV SI, OFFSET DATA2
       MOV DL, [S1)
       MOV SI, OFFSET DATA1
       MOV DI, OFFSET SUM
       MOV CX, 05
NEXT:  MOV AL, [S1]
       INC SI
       MUL BL
       AAM
       ADD AL, [DI]
       AAA
       MOV [DI], AL
       INC DI
       MOV [DI], AH
       LOOP NEXT
  CODE ENDS
       END START
```

结果[DS:0006—DS:000B]中的内容分别为 05，06，04，00，08，02 即56093X5=280465。

5. 数据传送

```
  DATA SEGMENT
  STR1 DB 30H, 31H, 32H, 33H, 34H
       DB 35H, 36H, 37H, 38H, 39H
       DB 40H, 41H, 42H, 43H, 44H, 45H
  COUNT EQU $-STR1
  STR2 DB COUNT DUP(0)
  DATA ENDS
  CODE SEGMENT
    ASSUME DS: DATA, C5: CODE
    START: MOV AX, DATA
           MOV DS, AX
           LEA SI, STR1
           LEA DI, STR2
           MOV CX, COUNT
    NEXT:  MOV AL, [SI]
           MOV [DI], AL
```

```
        INC SI
        INC DI
        LOOP NEXT
CODE ENDS
     END START
```

串传送

```
DATA SEGMENT
STR1 DB 30H, 31H, 32H, 33H, 34H
     DB 35H, 36H, 37H, 38H, 39H
     D8 40H, 41H, 42H, 43H, 44H, 45H
COUNT EQU $-STR1
STR2 DB COUNT DUP(0)
DATA ENDS
COOE SEGMENT
ASSUME DS:DATA, ES: DATA, CS:CODE
START: MOV AX, DATA
       MOV DS, AX
       MOV ES, AX
       LEA SI, STR1
       LEA DI, STR2
       MOV CX, COUNT
       CLD
       REP MOVSB
CODE ENDS
    END START
```

实验四

1. ASCII 码转换 BCD 码

```
DATA SEGMENT
BUF1 DB 30H, 31H, 32H, 33H, 34H
     DB 35H, 36H, 37H, 38H, 39H
BCDBUF DB 5 DUP(?)
DATA ENDS
COOE SEGMENT
     ASSUME CS:CODE, DS:DATA
BEGIN: MOV AX, DATA
       MOV DS, AX
```

```
        MOV CX, 05
        LEA BX, BUF1
        LEA DI, BCDBUF
NEXT:   MOV AX, [BX]
        AND AX, 0F0FH
        SHL AH, 1
        SHL AH, 1
        SHL AH, 1
        SHL AH, 1
        OR AH, AL
        MOV [D1], AH
        INC BK
        INC BX
        INC DI
        LOOP NEXT
        MOV AH, 4CH
        INT 21H
CODE ENDS
     END BEGIN
```

结果：[DS:000A～DS:000E]中的内容为 10H，32H，54H，76H，98H.

2. BCD 码转换为二进制数

```
DATA SEGMENT
    DNUM DW 8765H
    BNUM DW ?
DATA ENDS
STACK BEGMENT PARA STACK 'STACK'
    DB 100 DUP(?)
STACK ENDS
CODE SEGMENT
     ASSUME CS:CODE, DS:DATA
START:MOV AX, DATA
      MOV DS, AX
      MOV AK, DNUM
      MOV DI, AX
      MOV DX, AX
      MOV BX, AX
      AND BX, 000FH
     AND DX, 00F0H
```

```
    MOV CL, 4
    SHR DX, CL
    AND DI, 0F00H
    MOV CL, 8
    SHR DI, CL
    AND AX, 0F000H
    MOV CL, 12
    SHR AX, CL
    SAL AX, 1
    MOV SI, AX
    MOV CL, 2
    SAL AX, CL
    ADD AX, SI
    ADD AX, DI
    SAL AX, 1
    MOV SI, AX
    MOV CL, 2
    SAL AX, CL
    ADD AX, SI
    ADD AX, DX
    SAL AX, 1
    MOV SI, AX
    MOV CL, 2
    SAL AX, CL
    ADD AX, SI
    ADD AX, DX
    MOV MNUM, AX
CODE ENDS
    END START
```

结果：[DS：0002]=223DH

实验五

```
    DATA SEGMENT
    TSF1  DB "INPUT STRING: $"
    TSF2  DB 0DH,   0AH, "SPACE CHARACTER=$"
    TSF3  DB 0DH,   0AH, "NO SPACE CHARACTER $"
    BUF    DB 51
    DB ?
```

```
BUF1  DB 50 DUP(?)
DATA ENDS
CODE  SEGMENT
      ASSUME CS: COD1E,   DS:DATA, ES:DATA
START: MOV AX, DATA
       MOV DS, AX
       MOV ES, AX
       MOV AH, 09H
       LEA DX, TSF1
       INT 21H
       MOV AH, 0AH
       LEA DX, BUF
       INT 21H
       MOV SI, OFFSET BUF1
       MOV  CL, [SI—1]
       MOV CH, 00
       MOV BX, 00
       MOV DI, OFFSET BUF1
       MOV AL, 20H
       CLD
NEXT:  REPNZ SCASB
       JNZ OUT
       INC BX
       PUSH AX
       MOV AX, BK
       AAA
       MOV BX, AX
       POP AX
       CMP CX,  00
       JNE NEXT
OUT:   CMPBX, 00
       JE DIS
       ADD BX, 3030H
       MOV AH, 09H
       LEA DX, TSF2
       INT 21H
       MOV AH, 02
       MOV DL, BH
```

```
        INT 21H
        MOV DL, BL
        INT 21H
        JMP OUT1
  DIS:  MOV AH, 09H
        LEA DX, TSF3
        INT 21H
  OUT1: MOV AH, 4CH
        INT 21H
  CODE ENDS
        END START
```

实验六

```
 DATA SEGMENT
 TSF1  DB "INPUT STRING: $"
 TSF2  DB 0DH,   0AH, "SORT STRING:$"
 BUF   DB 51
 BUF1  DB ?
 BUF2  DB 50 DUP(?)
 DATA  ENDS
 CODE SEGMENT
       ASSUME CS:CODE, DS:DATA
 START:MOV AX, DATA
       MOV DS, AX
       MOV AH, 09H
       LEA DX, TSF1
       INT 21H
       MOV AH, 0AH
       LEA DX, BUF
       INT 21H
       MOV SI, OFFSET BUF1
       MOV CL, [SI]
       MOV CH, 00
       MOV DI, CX
       MOV BUF2 [D1], " $"
CONT1: MOV BL, 00
       MOV SI, DI
       MOV CX, DI
```

```
        DEC SI
AGAIN: MOV AL, BUF2[SI]
        CMP A1, BUF2[SI-1]
        JGE NEXT
        XCHG AL, BUF2[SI-1]
        MOV BUFF2 [SI], AL
        MOV BL, -1
 NEXT: DEC SI
        LOOP AGAIN
        CMP BL, 0
        JNE CONT1
        MOV AH, 09H
        LEA DX, TSF2
        INT 21H
        LEA DX, BUF2
        IHT 21H
        MOV AH, 4CH
        INT 21H
 CODE  ENDS
        END START
```

实验七　设置光标位置实验程序清单

```
 DATA SEGMENT
   COUNT  DB ?
   B1B    BD ?
 DATA  ENDS
 STACK SEGMENT STACK
    DB 200 DUP(?)
 STACK ENDS
 CODE SEGMENT
       ASSUME CS:CODE, SS:STACK, DS:DATA
DELAY  PROC NEAR
       PUSH CX
       PUSH DX
       MOV DX, 50
DL500:  MOV CX, 2801
DL10MS: LOOP DL10MS
```

```
        DEC DX
        JNZ DL500
        POP DX
        POP CX
        RET
DELAT ENDP
BEGIN: MOV AX, DATA
       MOV DS, AX
       MOV AH, 0
       MOV AL, 2            ; 设置成 80X25 黑白文本方式
       INT 10H
       MOV AH, 15
       INT 10H
       MOV DX, 0B24H        ; 获当前页号-->BH
       MOV AH, 2
       INT 10H
       MOV CX, 08
       MOV AL, 5
       MOV AH, 10
       INT 10H
NEXT:  MOV COUNT, 0
       MOV AH, 01
       MOV CX, 0BOCH
       INT 10H
       MOV AH, 02
       INT 10H
AGAIN: CALL  DELAY
       INC COOUNT
       CMP COUNT, 10
       JB AGAIN
       INC DL
       INC  BBB
       CMP BBB, 08
       JB NEXT
       MOV AH, 4CH
       INT 21H
CODE ENDS
     END  BEGIN
```

实验八　显示两个十进制数之和的程序清单

```
DATA  SEGMENT
INA    DB "INPUT DATA:$"
ASC1    DB 10
        DB ?
        DB 10 DUP(0)
BCD1    DB 8 DUP(0)
INB     DB 0DH, 0AH, '$'
OUT1    DB "OUTPUT VALUE, :" $"
ASC2    DB 20 DUP(0)
DATA  ENDS
STACK  SEGMENT PARA STACK  'STACK'
STAPN  DB 50 DUP(?)
TOP    EQU LENGTH STAPN
STACK  ENDS
CODE  SEGMENT
      ASSUME CS:CODE,   DS:DATA,   ES:DATA,   SS:STACK
START: MOV AX, DATA
       MOV DS, AX
       MOV ES, AX
       MOV AX, STACK
       MOV SS, AX
       MOV AX, TOP
       MOV SP, AX
       LEA DX, INA
       MOV AH, 09
       INT 21H
       LEA DX, ASC1
      MOV AH, 10
      INT 21H
      LEA DX, INB
      MOV AH, 09
      INT 21H
      MOV CX, 2
      LEA BX, ASC1+2
      LEA DI, BCD1
NEXT: MOV AX, [BX]
```

```
        AND AX, 0F0FH
        MOV DL, AH
        PUSH CX
        MOV CL, 4
        SHL Al, CL
        OR AL, DL
        MOV [DI], AL
        INC BX
        INC BX
        INC BK
        INC DI
        XOR AX, AX
        POP CX
        LOOP NEKT
        LEA BX, BCD1
        MOV AL, [BX]
        INC BX
        ADD AL, [BX]
        DAA
        JNC AAA
        ADD AH, 01
  AAA: INC BK
        MOV [DX], AH
        INC BX
        MOV [DX], AL
        LEA DK, OUT1
        MOV AH, 09
        INT 21H
        LEA SI, BCD1
        LEA DI, ASC2
        CLD
        MOV CX, 02
NEXT1: LODSB
        MOV DL, AL
        PUSH CX
        MOV CL, 4
        SHR AL, CL
        ADD AL, 30H
```

```
        STOSB
        MOV AL, BL
        AND AL, 0FH
        ADD AL, 30H
        STOSB
        INC DI
        POP CX
        LOOP NEXT1
        LEA DI, ASC2+2
        MOV AL, '+'
        MOV [D1], AL
      INC DI
      INC DI
      INC DI
      MOV AL, '='
      MOV [DI], AL
      LEA SI, BCD1+2
      LEA DI, ASC2+6
      CLD
      MOV CX, 2
NEXT2: LODSB
      MOV BL, AL
      PUSH CX
      MOV CL, 4
      SHR AL, CL
      ADD AL, 30H
      STOSB
      MOV AL, BL
      AND AL, 0FH
      ADD AL, 30H
      STOSB
      POP CX
      LOOP NEXT2
      MOV AL, 0DH
      MOV [DI], AL
      INC DI
      MOV AL, 0AH
      MOV [DI], AL
```

```
        INC DI
        MOV AL, '$'
        MOV [DI], AL
        LEA DX, ASC2
        MOV AH, 09
        INT 21H
        MOV AH, 4CH
        INT 21H
CODE  ENDS
      END START
INPUT DATA: 45; 75
OUTPUT VALUE:45+75=0120
```

实验九　时钟实验程序清单

```
  NAME    OUTPUT_CLOCK
  DATA SEGMENT
  BUFFER DB 11
         DB ?
         DB 10DUP(?)
  DATA  ENDS
  COSEG  SEGMENT
         ASSUME CS:COSEG,DS:DATA
  OUTCLK: MOV AX,DATA
          MOV DS,AX
          MOV DL,' :'
          MOV AH,2
          INT 21H
          MOV DX,OFFSET BUFFER
          MOV AH,10
          INT 21H
          MOV BX,OFFSET BUFFER+2
          MOV AL,[BX]
          AND AL,0FH
          MOV [BX],AL
          INC BX
          MOV AL,[BX]
          AND AL,0FH
```

```
        MOV [BX],AL
        INC BX
        INC BX
        MOV AL,[BX]
        AND AL,0FH
        MOV [BX],AL
        INC BX
        MOV AL,[BX]
        AND AL,0FH
        MOV [BX],AL
        INC BX
        INC BX
        MOV AL,[BX]
        AND AL,[BX]
        MOV [BX],AL
        INC BX
        MOV AL,[BX]
        AND AL,0FH
        MOV [BX],AL
        MOV BX,OFFSET BUFFER+2
        CALL TOBCD
        MOV CH,AL
        ADD  BX,3,
        CALL TOBCD
        MOV  DH,AL
        ADD BX,3
        CALL TOBCD
        MOV AL,AL
AGAIN:  CALL DIS
        MOV AL, DL
        ADD AL,1
        DAA
        MOV DL,AL
        CMP AL,60H
        JNE DISPY
        MOV DL,0
        MOV AL,DH
        ADD AL, 1
```

```
        AAA
        MOV AL, DH
        ADD AL, 1
        DAA
        MOV CH,AL
        CMP AL,24H
        JNE DISPY
        MOV CH, 0
DISPY: MOV BX, OFFSET BUFFER
        MOV  AL, 0DH
        MOV  [BX],AL
        INC  BX
        MOV  AL,0AH
        MOV [BX], AL
        INC  BX
        MOV  AL,CH
        CALL TRAN
        INC  BX
        MOV AL ':'
        MOV [BX},AL
        INC BX
        MOV AL,DL
        CALL TRAN
        INC BX
        MOV AL,':'
        MOV  [BX], AL
        INC  BX
        MOV AL,DL
        CALL TRAN
        INC  BX
        MOV  A1,' $'
        MOV [BX], AL
        PUSH BX
        PUSH CX
        PUSH DX
        MOV  DX, OFFSET BUFFER
        MOV  AH, 9
        INT  21H
```

```
        MOV  AH, 06
        MOV  DL,   0FFH
        INT  21H
        POP DX
        POP CX
        POP BX
        JNZ GO
        JMP AGAIN
  GO:   MOV AH,4CH
        INT 21H

 TOBCD PROC
        MOV Al, [BX]
        SHL AL, 1
        SHL AL, 1
        SHL AL, 1
        SHL AL, 1
        OR AL, [BX+1]
        RET
TOBCD ENDP
TRAN  PROC
       MOV CL, AL
       SHR AL, 1
       SHR AL, 1
       SHR AL, 1
       SHR AL, 1
       OR  AL, 30H
       MOV [BX], AL
       INC BX
       MOV AL, CL
       AND AL, 0FH
       OR AL, 30H
       MOV [BX],AL
       RET
 TRAN ENDP
 D1S PROC
       PUSH CX
       PUSH AX
```

```
        MOV CX, 0FFFFH
  GOON: DEC CX
        JNE GOON
        POP AX
        POP CX
        RET
   D1S  ENDP
  COSEG ENDS
        END OUTCLK
```

实验十　键盘中断实验程序清单

```
  STACK  SEGMENT STACK
  DW 200H DUP(?)
  STACK ENDS
  DATA SEGMENT
  KEY DB ?
  BUF DB "OK!"
  DATA ENDS
  CODE SEGMENT
       ASSUME CS:CODE, SS:STACK, DS:DATA
  DELYP PROC
        PUSH CX
        PUSH DX
        MOV DX, 20
DL500: MOV CX, 2801
DL10MS:LOOP DL10MS
         DEC DX
         JNZ DL500
         POP DX
         POP CX
         RET
   DELAY ENDP
   DISP1 PROC FAR
       PUSH AX
       PUSH BX
       PUSH CX
       PUSH DX
```

```
        MOV AH,15
        INT 10H
        MOV AH, 0
        INT 10H
        MOV CX, 1
        MOV DX, 0
REPT:   MOV AH, 2
        INT 10H
        MOV AL, 0FH
        MOV AH, 10
        INT 10H
        CALL DELAY
        SUB AL, AL
        MOV AH, 0
        INT 10H
        INC DH
        ADD DL, 2
        CMP DH, 25
        JNE REPT
        POP DX
        POP CX
        POP BX
        POP AX
        RET
DISP1   ENDP
DISP2   PROC FAR
        PUSH CX
        PUSH BX
        PUSH AX
        MOV CX, 3
        MOV BX,01
        CALL DELAY
        LOOP NEXTC
        POP AX
        POP BX
        POP CX
        RET
DISP2   ENDP
```

```
KEYINT  PROC FAR
        PUSH AX
        PUSH SI
        STI
        IN AL, 60H
        MOV AH, AL
        IN AL, 61H
        OR AL, 80H
        OUT 61H, AL
        AND AL, 7FH
        OUT 61H, AL
        TEST AH, 80H
        JNE GO
        STI
        INC KEY
        MOV SI,   OFFSET BUF
        CALL DISP2
  GO:   MOV AL, 20H
        OUT 20H, AL
        POP SI
        POP AX
        IRET
KEYINS ENDP

STAR:   MOV AX, STACK
        MOV SS, AX
        MOV AX, DATA
        MOV DS, AX
        MOV AX, 0
        MOV ES, AX
        MOV AX, ES:[24H]
        PUSH AX
        MOV AX, ES:[26H]
        PUSH AX
        CLI
        MOV AX, OFFSET KEY1NT
        MOV ES: [24H], AX
        MOV AX, SEG KEYINTS
```

```
        MOV ES: [26H], AX
        STI
        MOV  KEY, 0
AGAIN: CAll DISPl
        CMP KEY, 10
        JB AGAIN
        CLI
        POP AX
        MOV ES: [26H], AX
        POP AX
        MOV  ES:[24H], AX
        STI
        MOV AH, 4CH
        INT 21H
CODE   ENDS
        END START
```

实验十一　定时中断实验程序清单

```
STACK SEGMENT STACK
    DW 200H DUP(?)
STACK ENDS
DATA SEGMENT
COUNT DB ?
LOS DB ?
BUF DB ”SUN”
DATA ENDS
CODE SEGMENT
    ASSUME CS: CODE, ES: STACK, DS: DATA
DELAY  PROC
        PUSH CX
        PUSH DX
        MOV DX, 20
DL500: MOV CX,   2801
DL10MS: LOOP DL10MS
        DEC DX
        JNZ DL500
        POP DX
```

```
        POP CX
        RET
DELAY ENDP
DISP1  PROC FAR
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        MOV AH, 15
        INT 10H
        MOV AH, 0
        INT 10H
        MOV CX, 1
        MOV DX, 0
 REPT: MOV AH, 2
      INT 10H
      MOV AL, 0FH
      MOV AH, 10
      INT 10H
      CALL DE1AY
      SUB AL, AL
      MOV AH, 10
      INT 10H
      INC DH
      ADD DL, 2
      CMP DH, 25
      JNE REPT
      POP DX
      POP CX
      POP BX
      POP AX
      RET
  DISP1 ENDP
  DISP2 PROC FAR
       PUSH CX
       PUSH BX
       PUSH AX
       MOV CX, 3
```

```
  NEXTC: LODSB
         MOV AH, 0EH
         MOV BX, 01
         INT 10H
         CALL DELAY
         LOOP NEXTC
         POP AX
         POP BX
         POP CX
         RET
  DISP2  ENDP
  LOSINT PROC
         PUSH AX
         PUSH SI
         STI
         INC COUNT
         CMP COUNT, 30
         JNE GO
         MOV COUNT, 0
         MOV SI, OFFSET BUF
         CALL DISP2
         INC LOS
  GO:    POP SI
         POP AX
         IRET
LOSINT   ENDP
START:   MOV AX, STACK
         MOV SS, AX
         MOV AX, DATA
         MOV DS, AX
         MOV AX, 0
         MOV ES, AX
         MOV AX, ES: [70H]
         PUSH AX
         MOV AX, ES: [72H]
         PUSH AX
         CLI
         MOV AX, OFFSET LOSINT
```

```
        MOV ES:[70H], AX
        MOV AX, SEG LOS1NT
        MOV ES:[72H], AX
        STI
        MOV LOS, 0
        MOV COUNT, 0
AGAIN:  CALL DISP1
        CMP LOS, 10
        JB AGAIN
        CLI
        POP AX
        MOV ES:[72H], AX
        POP AX
        MOV ES:[70H], AX
        STI
        MOV AH, 4CH
        INT 21H
  CODE ENDS
       END START
```

实验十二　演奏音乐程序清单

```
  NAME    SING OF FOUR SEASONS
  STACK    SEGMENT PARA STACK  'STACK'
         DW 200 DUP(?)
  STACK    ENDS
  DATA    SEGMENT
  BG    DB  0AH,   0DH,"SING OFFOURSEASONS  :   $"
  FREQ  Dw 660, 660, 588, 524,, 588, 524, 494, 440 440, 440
        DW  698, 698, 660, 588, 524, 588, 698, 660
        DW  698, 698, 660, 588, 588, 698, 660, 660, 524, 440, 524
        DW  494, 660, 588, 524, 494, 524, 440, 0
TIME    DW  100, 50, 50, 50, 50, 50, 50, 100, 100, 200
        DW  100, 50, 50, 50, 50, 50, 50, 400
        DW  100, 50, 50, 100, 50, 50, 100, 50, 50, 100, 100
        DW  100, 100, 50, 50, 50, 50, 400
DATA   ENDS
CODE   SEGMENT
       ASSUME CS: CODE, DS: DATA, SS: STACK
```

```
SEASON: MOV AX, DATA
        MOV DS, AX
        MOV AX, STACK
        MOV  SS, AX
        MOV  DX,   OFFSET BG
        MOV  AH, 09
        INT  21H
        MOV  SI,   OFFSET FREQ
        MOV  BP,   OFFSET TIME
        CALL SING
        MOV AH, 4CH
        INT 21H
SING  PROC  NEAR
        PUSH  DI
        PUSH  SI
        PUSH  BP
        PUSH  BX
REPT:  MOV  DI, [SI]
        CMP  DI, 0
        JE END_SING
        MOV BX, DS:[BP]
        CALL  SOUND
        ADD  SI, 2
        ADD  BP, 2
        JMP  REPT
END_SING: POP BX
        POP  BP
        POP  SI
        POP  DI
        RET
 SING ENP
 SOUND PROC NEAR
      PUSH  AX
      PUSH  BX
      PUSH  CX
      PUSH  DX
      PUSH  DI
      MOV AL, 0B6H
```

```
        OUT 43H, AL
        MOV DX, 12H
        MOV  AX, 34DCH
        DIV  DI
        OUT  42H, AL
        MOV  AL, AH
        OUT  42H, AL
        IN   AL, 61H
        MOV  AH, AL
        OR   AL, 3
        OUT  61H, AL
DELAY: MOV CX, 2801
DL10MS: LOOP  DL10MS
      DEC  BX
      JNZ  DE1AY
      MOV AL, AH
      OUT 61H, AL
      POP  DI
      POP  DX
      POP  CX
      POP  BX
      POP  AX
      RET
  SOUND ENDP
  CODE ENDS
      END SEASON

  SING OF FOUR SEASONS:
```