《微机原理与汇编语言程序设计》

实验指导书

天 津 农 学 院

微 机 原 理 与 汇 编 语 言 精 品 课 程 组

前　言

本实验指导书适用于计算机专业，实验时间 24 学时，12 次上机时间。

主要学习内容为 80X86 语言实验环境配置、汇编源语言格式、输出字符、循环结构、子程序调用，以及加减乘除等指令操作、通用接口芯片的接口编程与使用。

学习结束后，要求学生能够独立编写出综合加减乘除等指令，以及循环结构、子程序调用等程序控制程序、看懂一般接口芯片电路图。

目　录

软件部分

硬件部分

软件部分

实验一　用 DEBUG 调试程序

一、实验目的

学习利用 DEBUG 调试程序的基本思想及方法

二、实验环境

1. 硬件：PC 微机
2. 软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容

利用 DEBUG 调试程序，可以将一个可执行程序（如.EXE、.COM 等）装入内存中，并接管对程序运行的控制权，通过采取如反汇编、断点运行、单步执行、寄存器内容修改等方法，对可执行程序进行跟踪、调试，以找出其中的设计错误，然后再对源程序进行相应修改，重新生成正确的可执行程序。

1. 准备被调试程序：

假定所有有关文件均在当前路径 C:\MYTEST>下，按照实验一的步骤生成一个被调试的可执行程序（如 TEST.EXE），参考程序如下：

```
DATA SEGMENT
  STR  DB  ‘Good  Morning!$’
DATA ENDS
CODE SEGMENT
    ASSUME CS：CODE，DS：DATA
    START：MOV  AX，DATA
MOV  DS，AX
MOV  DX，OFFSET STR
MOV  AH，09H
                INT  21H
                MOV  AH，4CH
                INT  21H
CODE ENDS
END START
```

2. 进入 DEBUG 环境：(其中斜体部分由键盘输入。)

C:\ MYTEST>*DEBUG TEST.EXE* ↘

-

其中，短线‘-’作为 DEBUG 环境的操作提示符，在此提示符下，可以输入各种 DEBUG 命令，对可执行程序 TEST.EXE 进行跟踪调试。

3. 主要调试命令：

1) 反汇编命令 U:将存储器中的二进制数据翻译成较有意义的助记符形式,以帮助理解。一般常用以下格式：

a) *-U* ↘：从当前 IP 处开始，对连续约 32 字节内容反汇编。如对 TEST.EXE，刚装入 DEBUG 时的 IP=0000，则在输入 U 命令后有如下显示：

```
-U ↘
12B7：0000  B8B612      MOV    AX，12B6
12B7：0003  8ED8        MOV    DS，AX
12B7：0005  BA0000      MOV    DX，0000
12B7：0008  B409        MOV    AH，09
12B7：000A  CD21        INT    21
12B7：000C  B44C        MOV    AH，4C
12B7：000E  CD21        INT    21
12B7：0010  EB51        JMP    0063
12B7：0012  8B867AF     MOV    AX，[BP+FF7A]
……          ……          ……
12B7：001F  8B4604      MOV    AX，[BP+04]
-
```

在上例中，12B7：0000 表示 CS：IP 的内容（其中 CS 的值是动态值）；B8B612 代表该处存放的二进制数据，亦即指令 MOV AX，12B6 的机器代码；当连续约 32 字节的数据反汇编完后，重新回到 DEBUG 提示符“-”下，如果再键入 U 命令，则将继续对后面的内存区反汇编。

特别应该注意的是，由于反汇编命令针对内存区的二进制数据，而被调试程序仅占内存区的某一部分，故反汇编出来的内容并非全是被调试程序的代码，如上例中的 JMP 0063 以后的部分，显然不是 TEST.EXE 的内容。另外还需注意，DEBUG 默认使用十六进制。

b) *–U 0123* ↘：从指定的 IP=0123 处开始，对连续约 32 字节内容反汇编。

c) *–U 0123 0143* ↘：从指定的 IP=0123 处开始反汇编，直至指定的 0143 处结束。

2) 显示寄存器命令 R：显示或修改寄存器的内容。一般常用以下形式：

a) *-R* ↘：显示所有寄存器当前的内容及当前将执行的指令。如对 TEST.EXE，在程序运行之前，键入 R 命令：

```
-R ↘
AX=0000  BX=0000  CX=0020  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=12A6  ES=12A6  SS=12B6  CS=12B7  IP=0000   NV UP EI PL NZ NA PO NC
12B7：0000  B8B612          MOV  AX, 12B6
-
```

在显示寄存器内容时，标志寄存器 F（或程序状态字寄存器 PSW）表示成各个分离的标志位，其意义如下表所示：

	溢出	方向	中断	符号	零	辅助进位	奇偶	进位
0	NV	UP	DI	PL	NZ	NA	PO	NC
1	OV	DN	EI	NG	ZR	AC	PE	CY

b) *-R AX* ↘：显示指定的 AX 寄存器当前的内容，并等待键入新值；如果不作修改，可直接回车。如：

-R AX ↘
AX 0000
：*1234* ↘
-

c) *-R F* ↘：显示标志寄存器 F 各个标志位的内容，并等待键入新的标志位；如果不作修改，可直接回车。如：

-R F ↘
NV UP EI PL NZ NA PO NC　*-ZR* ↘
-

3) 运行命令 G：使程序在 DEBUG 控制下运行，一般有全程、断点运行两种方式。

a) *-G* ↘：控制程序由当前 IP 处运行，直至程序结束。如果当前 IP 为初始值，其作用则相当于直接在 DOS 下运行程序，一般用于快速观察程序的运行情况。

b) *-G 0123* ↘：控制程序由当前 IP 处运行，直至指定的断点 IP=0123H 处，程序暂停，显示各个寄存器的当前值及断点处指令，然后返回 DEBUG 提示符“-”下。如对 TEST.EXE，若想观察字符串显示的入口参数是否设置好，则可以断点运行至 000A 处：

-G 000A ↘
AX=09B6　BX=0000　CX=0020　DX=0000　SP=0000　BP=0000　SI=0000　DI=0000
DS=12B6　ES=12A6　SS=12B6　CS=12B7　IP=000A　NV UP EI PL NZ NA PO NC
12B7：000A　CD21　　INT　21
-

断点一般选取在需要观察的地方，当程序停下来后，可以根据各方面的情况（如寄存器、缓冲区、标志等）来判断程序是否运行正确。

4) 单步命令 T：控制程序运行一条指令后暂停，显示各个寄存器的当前值及断点处指令，然后返回 DEBUG 提示符“-”下。如对 TEST.EXE，若当前 IP 为初始值，则键入 T 命令后有如下显示：

-T ↘
AX=12B6　BX=0000　CX=0020　DX=0000　SP=0000　BP=0000　SI=0000　DI=0000
DS=12A6　ES=12A6　SS=12B6　CS=12B7　IP=0003　NV UP EI PL NZ NA PO NC
12B7：0003　8ED8　　MOV　DS，AX
-

单步命令一般用于需对程序运行作仔细分析的地方，如判断分支转移、观察运算结果等。若能综合运用断点及单步指令，则可大大提高 DEBUG 调试的速度及效率。但应注意，当 IP 指针位于 INT 21H 一类指令处时，执行 T 命令将会使程序进入该功能调用子程序中，因此，

这种情况下最好不用 T 命令，而用断点运行命令跳过该类功能调用指令。

5) 显示内存命令 D：以十六进制及 ASCII 两种方式显示内存区的二进制数据，通常用来观察数据段内的缓冲区内容。一般常用以下二种形式：

a) *-D* ↘：从 0000 单元开始，连续显示 128 个内存单元的内容，如果继续键入 D 命令，则继续显示后 128 个单元内容。如对 TEST.EXE，若想观察字符串显示时的字符串内容是否正确，则可在程序断点运行至 000A 处，键入 D 命令：

```
-D DS：0 ↘
12B6：0000  48 65 6C 6C 6F 2C 57 6F-72 6C 64 21 24 00 00 00    Hello,World!$...
12B6：0010  B8 B6 12 8E D8 BA 00 00-B4 09 CD 21 B4 4C CD 21    ...........!.L.!
12B6：0020  EB 51 8B 86 7A FF 2B C6-40 50 8A 46 08 98 50 8B    .Q..z.+.@P.F..P.
    ……          ……
12B6：0070  83 C4 02 8B 5E 04 8A 07-2A E4 89 86 7A FF 3B C6    ....^...*...z.;.
-
```

在上例中，128 个单元分成 8 行，每行 16 个单元，每个单元的内容分别以十六进制形式和 ASCII 码形式显示。如果该单元的内容不是可显示字符，则在 ASCII 区内显示为“.”。

b) *–D 0123 0143* ↘：从指定的 0123 单元开始显示，直至指定的 0143 单元结束。

6) 汇编命令 A：用于在 DEBUG 环境下直接键入汇编语言语句、生成较简单的可执行代码而不必经过完整的汇编语言编程步骤，或者用来在调试过程中临时修改某条指令。如：

```
-A ↘
127D：0100  MOV   AH，02 ↘
127D：0102  MOV   AL，6A ↘
127D：0104  INT   21 ↘
127D：0106   ↘
-A 0102 ↘
127D：0102  MOV   DL，6A ↘
127D：0104   ↘
-
```

当键入汇编命令 A 后，将从当前 IP 或指定地址处提示输入汇编语句，每输入一条语句，DEBUG 将其汇编成机器码，并存入相应的存储单元中，然后地址自动增加，继续提示输入下一条语句；如果直接回车，则结束汇编命令。***特别应注意的是，DEBUG 默认使用十六进制，故在输入时不能使用 H。***

7) 装载命令 L：用来将被调试程序重新装载进内存中，一般用于程序运行结束后需继续调试程序时，或需从头开始调试程序时。如对 TEST.EXE：

```
-G↘
Hello,World!
Program terminated normally
-L↘-
```

8) 退出命令 Q：键入此命令，即退出 DEBUG 状态，返回 DOS。

实验二　汇编语言上机基本步骤

一、实验目的：

熟悉汇编语言的上机过程，掌握各项工具软件的使用方法

二、实验环境

1. 硬件：PC 微机
2. 软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容：

1. 前期准备：

在开始进行汇编语言上机练习之前，建立并进入自己的工作子目录，准备好相关工具软件如 MASM.EXE、LINK.EXE 等，其后所有工作均在自己的子目录中进行，以避免因路径概念不清而导致的文件存取错误，以及对系统其它部分造成影响。

1) 建立自己的工作子目录（例如 MYTEST）：
   C:\>*MD MYTEST*
2) 进入自己的工作子目录：
   C:\>*CD MYTEST*
3) 将所需工具软件从其它地方（如 C:\MASM）拷贝进自己的工作子目录：
   C:\MYTEST>*COPY C:\MASM\MASM.EXE*
   C:\MYTEST>*COPY C:\MASM\LINK.EXE*

2. 编写源程序：

原则上可以用任何文字处理软件（如 EDIT、写字板、甚至 WORD）编写源程序，但必须注意，源程序应为 ASCII 码文件（或称纯文本文件），扩展名一般为.ASM。因此，建议使用 DOS 环境下的 EDIT 软件。

1) 在 DOS 系统操作提示符下键入 EDIT 并回车，即可进入 EDIT 文本编辑环境。如果没有进入，检查当前路径下是否存在 EDIT.EXE 文件，以及文件是否完整等。

2) 从键盘输入下列程序（不必输入注释部分）：

```
CODE    SEGMENT                    ；定义一个 CODE 段
        ASSUME CS：CODE            ；定义 CODE 段为代码段
START：                            ；可执行语句起始处
        MOV   AH，02H              ；以下三条语句将显示字母 a
        MOV   DL，'a'
        INT   21H
        MOV   AH，4CH              ；以下二条语句将返回 DOS
        INT   21H
CODE    ENDS                       ；CODE 段结束
        END   START                ；整个程序结束
```

该程序的功能是仅在 CRT 屏上显示一个字母 a，完成功能的只有其中的三条语句，但其它部分则是一个完整的汇编语言源程序必不可少的部分。

3) 存盘退出 EDIT 文本编辑环境。在存盘时应将文件的扩展名确定为.ASM（如

TEST.ASM），并注意存盘的路径，最好与 EDIT、MASM、LINK 等软件相同。

4) 在 DOS 系统提示符下利用 DIR 命令检查 TEST.ASM 文件是否确实产生。

3. 汇编：

利用 MASM.EXE 宏汇编程序，将已经存盘的 ASCII 码源程序翻译成二进制目标程序，其扩展名一般为.OBJ。

1) 操作方法：假定当前工作路径为 C:\MYTEST>，且所需文件均存在于当前路径，则针对源程序 TEST.ASM 的汇编有以下三种方法（其中斜体部分由键盘输入）：

a) C:\MYTEST>*MASM.EXE* ↘

Microsoft (R) Macro Assembler Version 5.00

Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Source filename [.ASM]:*TEST* ↘

Object filename [TEST.OBJ]: ↘

Source listing [NUL.LST]: ↘

Cross-reference [NUL.CRF]: ↘

51524 + 435132 Bytes symbol space free

0 Warning Errors

0 Severe Errors

b) C:\ MYTEST >MASM TEST.ASM↘

Microsoft (R) Macro Assembler Version 5.00

Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [TEST.OBJ]: ↘

Source listing [NUL.LST]: ↘

Cross-reference [NUL.CRF]: ↘

51524 + 435132 Bytes symbol space free

0 Warning Errors

0 Severe Errors

c) C:\ MYTEST >MASM TEST； ↘

Microsoft (R) Macro Assembler Version 5.00

Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

51524 + 435132 Bytes symbol space free

0 Warning Errors

0 Severe Errors

2) 在上述三种方法中，推荐使用第三种，但是必须满足以下要求：

a) 所有文件均位于当前工作路径下。

b) 源程序扩展名为.ASM，目标程序扩展名为.OBJ。

3) 若源程序有语法错误，则汇编结束将给出提示信息，并依次列出错误出现的行号及性质。这时，应重新进入 EDIT 文本编辑环境中，根据提示对源程序进行修改，然后重新存盘、汇编。只有所有错误为 0，才能得到正确的目标文件。注意该步骤只能检查出语法错误，对设计思想上的错误，应通过调试才能检查出来。

4) 汇编结束后，检查是否产生相应目标程序 TEST.OBJ。

4. 连接：

利用 LINK.EXE 连接程序，将二进制目标程序整理成 DOS 系统下的可执行程序，其扩展名必须为.EXE。

1) 操作方法：与汇编相似，可有多种形式，一般使用：

C:\ MYTEST >*LINK TEST；* ↘

但需注意这时的 TEST 应是.TEST.OBJ 文件。

2) 在得到正确的.OBJ 文件后，该步骤一般不会出现问题，但可能会出现下列提示：

LINK：warning L4021：no stack segment

针对该提示，可不予理会。

3) 连接结束后，检查是否产生相应的可执行程序 TEST.EXE。

5. 运行：

1) 如果没有问题，生成的可执行程序 TEST.EXE 即可以象其它 DOS 外部命令一样，直接在 DOS 系统下运行，整个编程工作完成。如在本实验中：

C:\ MYTEST >*TEST* ↘

2) 一般在编写较复杂的程序时，可能出现设计上的错误。如果不能在源程序中检查出错误，则必须通过 DEBUG 调试，才能检查出错误所在，然后再回到前面的各个步骤中重复操作。在本实验中，由于程序简单，可以不必调试。

四、练习

对实验程序进行由浅及深的修改，领会上机的各个步骤及注意事项。

实验三　基本程序设计

一、实验目的

学习顺序、分支、循环三种基本结构的程序设计方法

二、实验环境

1. 硬件：PC 微机
2. 软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容

1. 编写程序，要求对键盘输入的小写英文字母用相应大写英文字母显示，如键盘输入“a”，则显示“A”。
2. 编写程序，若键盘输入小写字母，则用相应大写字母显示；反之，若键盘输入大写字母，则用相应小写字母显示。
3. 编写程序，循环实现上述程序 2 的功能，直至键盘输入任一非英文字符，程序停止。

四、练习

1. 统计某班学生的成绩等级并存放在相应变量中。等级的划分原则为：

   A：90~100　　B：80~89　　C：70~79　　D：60~69　　E：0~59

2. 编写程序，统计某个字变量中 1 的个数并存放在相应变量中。

实验四　分支程序设计（一）

一、实验目的

掌握分支结构程序的编制方法。

二、实验环境

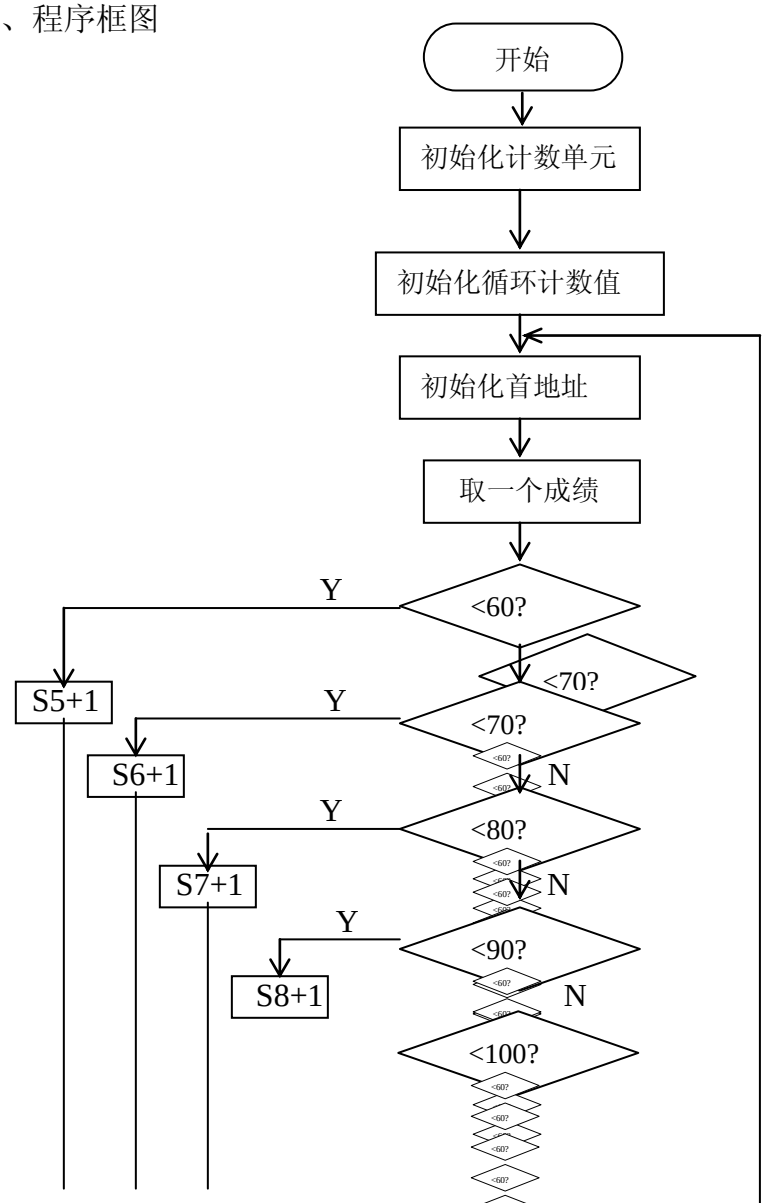
1. 硬件：PC 微机
2. 软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

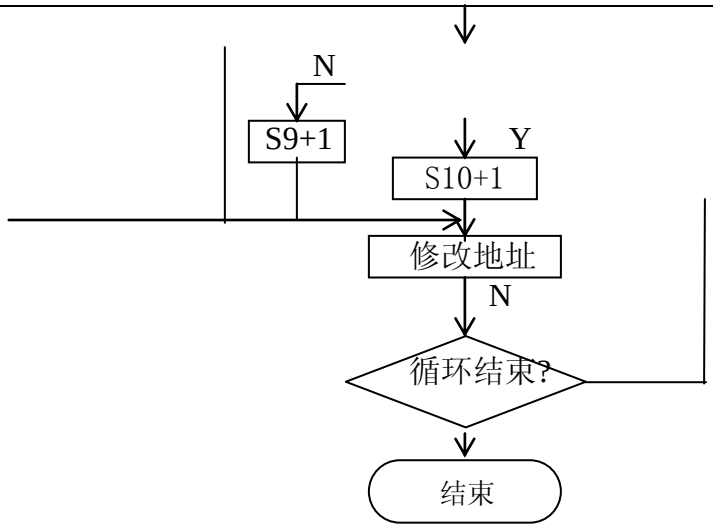
三、实验内容与要求

编制程序实现如下操作：

设有 10 个学生成绩，分别统计低于 60 分、60～69 分、70～79 分、80～89 分、90～99 分及 100 分的人数，并存放于 S5、S6、S7、S8、S9、S10 单元中。

四、程序框图





五、程序清单：

```
;          统计学生成绩
DATA       SEGMENT
GRADE   DW 95H，60H，75H，92H，71H，86H，54H，89H，83H，76H
N          EQU   ($—GRADE)/2
ORG 30H
S5         DW    0
S6         DW    0
S7         DW    0
S8         DW    0
S9         DW    0
S10        DW    0
DATA       ENDS
;
STACK        SEGMENT  STACK
STA           DB 20 DUP (0)
TOP          EQU $—STA
STACK       ENDS
;
CODE          SEGMENT
MAIN          PROC    FAR
              ASSUME  CS：CODE，  DS：DATA，  SS：STACK
START：       PUSH    DS
              SUB     AX，AX
              PUSH    AX
              MOV     AX，DATA
              MOV     DS，AX
              MOV     CX，N
              LEA     BX，GRADE          ；成绩表首地址
COMPARE：     MOV     AX，[BX]
              CMP      AX，60H           ；<60?
```

```
            JL      FIVE
            CMP     AX，70H             ；<70?
            JL      SIX
            CMP     AX，80H             ；<80?
            JL      SEVEN
            CMP     AX，90H             ；<90?
            JL      EIGHT
            CMP     AX，100H            ；=100
            JNE     NINE
            INC     S10
            JMP     CHA
NINE：      INC      S9
            JMP     CHA
EIGHT：     INC      S8
            JMP     CHA
SEVEN：     INC       S7
            JMP     CHA
SIX：       INC      S6
            JMP     CHA
FIVE：      INC      S5
            JMP     CHA
CHA：       ADD      BX，2                  ；循环学生人数
            LOOP    COMPARE
            RET
MAIN        ENDP
CODE        ENDS
            END     START
```

执行程序后，将结果分别填入下列表中

N(总人数)	S5	S6	S7	S8	S9	S10

六、思考题

1） 计算出各等级成绩百分比。分别填入下面表的 A、B、C、D、E、中
2） 增加统计平均成绩一项。

平均成绩	E	D	C	B	A	

实验五　分支程序设计（二）

一、实验目的

1. 利用测试与转移指令实现分支。

2. 利用比较与转移指令实现分支。

二、实验环境

1. 硬件：PC 微机
2. 软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容与要求

1. 利用测试与转移指令实现分支。

程序设计方法：在需要分支的地方用逻辑测试指令 TEST 进行分支条件的测试判断，再利用各种条件转移指令实现程序分支。

2. 利用比较与转移指令实现分支。

程序设计方法：在需要分支的地方用两数的比较指令 CMP，或串比较指令 CMPS 等

进行分支条件的比较判断，再利用转移指令（如 JNZ、JNC、JA、JB 等）实现程序的分支。

要求：自编程序：

1. 在 BLOCK 开始的内存单元中有若干以字节为单位的正、负数，自编程序，试统计其中正数的个数存放于 M_DATA 单元中，负数的个数存放于 P_DATA 单元中。

2. 在 BLOCK 开始的内存单元中有若干以字节为单位的奇、偶数，自编程序，试统计其中偶数的个数存放于 M_DATA 单元中，奇数的个数存放于 P_DATA 单元中。

四、部分程序清单

```
BLOCK       DB 37，—90，—32，60，—7，—120
COUNT       EQU  $—BLOCK
P_DATA      DB  COUNT  DCP (0)
M_DATA      DB  COUNT  DCP (0)

 ……
            LEA    SI，BLOOK
            LEA    DI，P_DATA
            LEA    BX，M_DATA
            MOV    CX，COUNT
RETRY：     MOV    AL，[SI]
            CMP    AL，0
```

```
            JGE     PP
            MOV     [BX]，AL
            INC     BX
            JMP     LOOP1
PP：        MOV     [DI]，AL
            INC     DI
LOOP1：     INC     SI
            LOOP    RETRY
              ……
```

五、思考题

如果将比较指令改为测试指令(加下划线处)，程序需做哪些改动？

实验六　分支程序设计（三）

一、实验目的

1） 熟悉分支程序的编写。

2） 学习 DOS 系统功能调用（INT 21H/08H，AL=ASCII 字符）从键盘接收单个字符的用法。

3） 掌握数据的输入与输出方法。

二、实验环境

1）硬件：PC 微机

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容

1） 用 DOS 系统功能调用 INT 21H 的 08H 功能接收键盘字符（AL=字符）,若是 F，显示“This is the first word string.’,若是 S，显示“This is the second word string.’,否则退出。

2） 设计一数据块间的搬移程序

四、实验要求

1） 实验前准备好汇编语言源程序，阅读实验指导书关于调试方法与步骤的内容。

2） 实验要求在 PC 机上进行。

五、编程提示

1） 用 DOS 系统功能调用 INT 21H 的 09H 功能显示字符串。

2）用 P 命令执行程序，可看出结果。

3） 显示字符串参考程序

```
data    segment
        str1 db  'this is the first word string.','$'
        str2 db  'this is the second word string.','$'
data    ends
stack   segment
        sta     db  50     dup  (?)
        top1 equ length    sta
stack   ends
code    segment
        assume  cs:code,ds:data,ss:stack
main    proc    far
start:  push    ds                ;将 DS 入栈
        mov     ax, 00h           ;AX 置零
        push    ax                ;将 0 入栈
        mov     ax, data          ;初始化 DS
        mov     ds, ax
        mov     ah, 08h               ;键盘输入一个字符
        int     21h                   ;字符在 AL 中
        cmp     al, 'f'
        jz      disp1
        cmp     al,  's'
        jz      disp2
        jmp     do
disp1:  mov     dx, offset str1       ;显示字符串
        mov     ah, 09h
        int     21h
        jmp     do
disp2:  mov     dx, offset str2
        mov     ah, 09h
        int     21h
do :    mov     ax, 4c00h             ;返回 DOS
        int     21h
main    endp
code    ends
        end  start
```

4） 设计一数据块间的搬移程序：

程序要求把内存中一数据区（称为源数据块）传送到另一存储区（成为目的数据块）。

```
STACK   SEGMENT STACK
        DW      64 DUP(?)
STACK   ENDS
CODE    SEGMENT
        ASSUME  CS:CODE
START:  MOV     CX,0010H
        MOV     SI,3100H
```

```
        MOV     DI,3200H
        CMP     SI,DI
        JA      A2
        ADD     SI,CX
        ADD     DI,CX
        DEC     SI
        DEC     DI
A1:     MOV     AL,[SI]
        MOV     [DI],AL
        DEC     SI
        DEC     DI
        DEC     CX
        JNE     A1
        JMP     A3
A2:     MOV     AL,[SI]
        MOV     [DI],AL
        INC     SI
        INC     DI
        DEC     CX
        JNE     A2
A3:     JMP     A3
CODE    ENDS
        END     START
```

实验步骤

(1)按实验流程图设计编写实验程序。（自主编程，不使用上述现成的程序）

(2) 输入程序并检查无误，经汇编、连接后产生正确的可执行文件。

(3) 用 E 命令在以 SI 寄存器内容（具体值见程序）为起址的单元中连续填入 16 个数。

(4) 运行实验程序。

(5) 用 D 命令查看 DI 寄存器内容为起址的单元中的数据是否与 SI 单元中数据相同。

(6) 试改变 SI、DI 的取值（修改程序），观察在三种不同的数据块情况下程序的运行。

六、实验报告

1） 说明程序的功能，使用方法。

2） 说明上机调试的步骤，出现的问题，对问题的分析与解决。

3） 画出程序框图，打印程序清单 与执行过程清单。

实验七　循环程序设计实验

一、 实验目的

1） 学习将一个十六位二进制数数转换成四位十六进制数显示的编程方法。

2） 熟悉循环程序的设计方法。

3） 学习乘法指令的用法。

二、实验环境

1）硬件：PC 微机

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、 实验内容

实现两个字节相乘的程序，并转换成十六进制数显示出结果。

23H×20H=？

四、实验要求

1） 分析题目，确定算法，画出程序框图。

2） 实验前准备好汇编语言源程序。

3） 实验要求在 PC 机上进行。

五、编程提示

1） 两个数据放在数据段的 DAT1 和 DAT2 中。

2） 计算结果转换成 ASCII 码显示,因此结果的高四位要拼成 3,用 INT 21H 的 02H 功能(DL=显示字符)显示结果。

3） 参考程序

```
DATA   SEGMENT
DAT1   DB    25H                    ;       25H
DAT2   DB    30H                    ;       30H
SUM1   DW    ?,?
SUM2   DB    10 DUP ('0')
TABLE  DB  '0','1','2','3','4','5','6','7','8','9'
       DB  'A','B','C','D','E','F'
DATA   ENDS
STACK  SEGMENT
ST1    DB  100  DUP  ('SA')
       TOP1   EQU  ST1  LENGTH ST1
STACK  ENDS

CODE   SEGMENT
ASSUME CS:CODE,DS:DATA,SS:STACK
MAIN   PROC   FAR
START: MOV    AX, DATA                ;初始化数据段
       MOV    DS, AX
       MOV    AX, STACK               ;初始化堆栈段
```

```
        MOV     SS, AX
        MOV     AX, 100
        MOV     SP, AX
        XOR     AX, AX
        MOV     BL, DAT1                    ;取乘数
        MOV     AL, DAT2                    ;取被乘数
        MOV     SI, OFFSET TABLE            ;取 ASCII 码表首地址
        MOV     DI, OFFSET SUM2             ;取结果地址
        MUL     BL
        MOV     SUM1,AX                     ;存乘积二进制结果
;       convert binary number in bx to hex   转换成 16 进制
        mov     BX, AX                      ;结果送 BX
        MOV     CX, 4                       ;转换 4 位数
 HEX16:PUSH     CX                          ;保护 CX
        MOV     CL, 4                       ;循环左移 4 位
        ROL     BX, CL
        MOV     AL, BL
        AND     AL, 0FH                     ;保留低四位
        PUSH    BX                          ;保护 BX
        MOV     BX, SI                      ;ASCII 码表首地址送 BX
        XLAT                                ;转换成 ASCII 码
        MOV     [DI], AL                    ;存结果的 ASCII 码
        INC     DI                          ;地址加一
        POP     BX                          ;恢复 BX 中待转换的数
        POP     CX                          ;弹出 CX 的计数值
        LOOP    HEX16                       ;未转换完，继续
;       display  results on screen
        MOV     AH, 02H                     ;调用 DOS 的 02H 功能显示
        MOV     CX, 04H                     ;显示数据的位数
        MOV     DI, OFFSET SUM2             ;送出待显示数据的首地址
DON2:   MOV     DL, [DI]                    ;显示数据送 DL
        INT     21H                         ;显示
        INC     DI                          ;显示数据所在存储单元加一
        LOOP    DON2                        ;未显示完,继续
        MOV     AH, 4CH                     ;返回 DOS
        INT     21H
        RET
MAIN    ENDP
CODE    ENDS
         END    START
```

六、实验报告

1） 说明程序结构及功能。

2） 说明入口参数与出口参数，参数的输人与输出方法。

3） 说明调试过程中遇到的问题及解决的方法。

4） 画出程序框图，打印源程序清单与执行结果。

实验八　子程序设计实验

一、实验目的

1） 学习将一个十六位二进制数数转换成四位十六进制数显示的编程方法。

2） 熟悉子程序的设计方法。

3） 学习乘法指令的用法。

二、实验环境

1）硬件：PC 微机

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容

实现多字节非组合 BCD 码相加的程序，并显示出结果。

1111111+999999=?

四、实验要求

1） 分析题目，确定算法，画出程序框图。

2） 实验前准备好汇编语言源程序。

3） 实验要求在 PC 机上进行。

五、编程提示

1） 两个数据放在数据段的 DAT1 和 DAT2 中。

2） 加法子程序：采用 BCD 码运算,用带进位位的指令 ADC,后面要跟加法校正 AAA。

3） 显示子程序：用 INT 21H 的 02H 功能(DL=显示字符)显示结果。

4） 参考程序

```
DATA    SEGMENT
DAT1    DB    01H,01H,01H,01H,01H,01H,01H,01H          ;11111111
DAT2    DB    09H,09H,09H,09H,09H,09H,09H,09H          ;99999999
SUM     DB    20  DUP  (?)
DATA    ENDS
STACK   SEGMENT
ST1     DB    100    DUP  ('SA')
TOP1    EQU   ST1  LENGTH ST1
STACK  ENDS
```

```
CODE    SEGMENT
        ASUMME  CS:CODE,DS:DATA,SS:STACK
MAIN    PROC    FAR
START:  MOV     AX, DATA                ;初始化数据段
        MOV     DS, AX
        MOV     AX, STACK               ;初始化堆栈段
        MOV     SS, AX
        MOV     AX, TOP1
        MOV     SP, AX
        MOV     CX, 8                   ;计算 8 次
        MOV     BX,  OFFSET  DAT1       ;取数据地址
        MOV     SI , OFFSET  DAT2       ;取数据地址
        MOV     DI , OFFSET SUM         ;取结果地址
        MOV     AH, 00H                 ;将暂存标志的 AH 清 0
  DON1: CALL    ADD1
        LOOP    DON1                    ;未计算完，继续
        AND     AH,     01H             ;将最高位的进位标志送 AH
        OR      AH,     30H             ;最高位的进位位拼成 ASCII 码
        MOV     [DI],   AH              ;存结果的最高位
        INC     CX                      ;显示数据的位数
        CALL    DISPLAY
        MOV     AH,4CH
        INT     21H
ADD1    PROC    NEAR
        PUSH    AX
 don1:  MOV     AL, [BX]                ; 取第一个数
        SAHF                            ;将 AH 中的标志送标志寄存器
        ADC     AL, [SI]                ;与第二个数带进位加
        AAA                             ;十进制校正
        LAHF                            ;将标志寄存器内容送 AH
        OR      AL, 30H                 ;计算值拼成 ASCII 码
        MOV     [DI], AL                ;存结果的 ASCII 码
        INC     BX                      ;指向第一个数的下一位
        INC     SI                      ;指向第二个数的下一位
        INC     DI                      ;指向结果单元的下一位
        POP     AX
        RET
ADD1    ENDP
DISPLAY  PROC  NEAR
        PUSH    AX
        PUSH    DX
DON2: MOV     AH,     02H               ;调用 DOS 的 02H 功能显示
      MOV     DL,     [DI]              ;显示数据送 DL
       INT     21H                      ;显示
```

```
        DEC     DI                          ;显示数据所在存储单元加一
        LOOP    DON2                        ;未显示完,继续
        RET
DISPLAY  ENDP
         RET
MAIN    ENDP
CODE    ENDS
        END       START
```

六、实验报告

1） 说明程序结构及功能。

2） 说明入口参数与出口参数，参数的输人与输出方法。

3） 说明调试过程中遇到的问题及解决的方法。

4） 画出程序框图，打印源程序清单与执行结果。

七、思考题

1）子程序执行完毕后要返回程序调用，它返回调用程序的什么地方，是靠什么指令、什么方法返回的？

实验九　排序程序设计实验

一、实验目的

1）掌握分支、循环、子程序调用等基本的程序结构。

2）学习综合程序的设计、编制及调试。

二、实验环境

1）硬件：PC 微机

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容

1）在数据区中存放着一组数(任意指定)，数据的个数就是数据缓冲区的长度，要求用起泡法对该数据区中数据按递增关系排序。

设计思想：

a）从最后一个数（或第一个数）开始，依次把相邻的两个数进行比较，即第 N 个数与第 N-1 个数比较，第 N-1 个数与第 N-2 个数比较等等；若第 N-1 个数大于第 N 个数，则两者交换，否则不交换，直到 N 个数的相邻两个数都比较完为止。此时，N 个数中的最小数将被排在 N 个数的最前列。

b）对剩下的N-1个数重复上步，找到N-1个数中的最小数。

c）重复第二步，直到N个数个部排序好为止。

2）学生成绩名次表

设分数为 1-100 之间的 30 个成绩保存在数据段偏移为 3000H 开始的单元中，3000H+i 表示学号为 i 的学生成绩(i 从 0 开始)。编写程序能在偏移 3100H 开始的区域排出名次表，3100H+i 为学号 i 的学生名次。

四、实验程序

```
DATA    SEGMENT
        TAB  DB '8095554'
        N = $-TAB
        OK   DB 0DH,0AH,'OK!$'
DATA    ENDS
STACK   SEGMENT
        STA     DB 20 DUP(?)
        TOP     EQU LENGTH STA
STACK   ENDS
CODE    SEGMENT
        ASSUME  CS:CODE,DS:DATA, SS:STACK
STAR:   MOV     AX,DATA
        MOV     DS,AX                   ;初始化数据段
        MOV     AX,STACK
        MOV     SS,AX
        MOV     AX,TOP
        MOV     SP,AX
        CALL    ARRAY
DO:     MOV     AH,4CH
        INT     21H                     ;返回 DOS
ARRAY   PROC    NEAR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DL,N-1                  ;置外循环次数
        MOV     DH,1                    ;设有交换标志
        XOR     BX, BX
UPPER:  OR      DH,DH                   ;
        JZ      DISP                    ;无交换,已排好序,退出
        MOV     DH,0                    ;无交换
        MOV     CX,N-1
        SUB     CX,BX                   ;CX=CX-I 内循环次数
        MOV     SI,0                    ;指向表首
INNER:  MOV     AL,TAB[SI]              ;字符送 AL
        INC     SI                      ;指向下个字符
```

```
        CMP     AL,TAB[SI]          ;比较表中相邻字符
        JBE     DON                 ;小于
        XCHG    AL,TAB[SI]          ;否则交换,大字符下
        MOV     TAB[SI-1],AL        ;小字符上浮
        MOV     DH,1                ;有交换,DH=1
DON:    LOOP    INNER               ;内循环结束?CX-1
        INC     BX                  ;一次内循环完成,加一
        DEC     DL                  ;外循环次数减一
        CMP     DL,0
        JNZ     UPPER               ;外循环次数非零,继续
DISP:   MOV     DX,OFFSET TAB
        MOV     AH,09H
        INT     21H                 ;显示排好序的字符
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
ARRAY   ENDP
CODE    ENDS
        END STAR
```

五、思考题

1）请指出程序1中哪部分属于外循环，哪部分属于内循环？

2）修改程序2，使其能将3100开始的区域中的名次表显示出来。

实验十　运算类指令编程实验

一、实验目的

1） 掌握使用运算类指令编程及调试方法。

2） 掌握运算类指令对各状态标志位的影响及其测试方法。

二、实验环境

1）硬件：PC 微机

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容

8086/8088指令系统提供了实现加、减、乘、除运算的基本指令，可对如表所示的数据类型进行算术运算。

表 1　数据类型算术运算表

数制	二进制		BCD 码	
	带符号	无符号	组合	非组合
运算符	+、−、×、÷		+、−	+、−、×、÷
操作数	字节、字、多精度		字节(二位数字)	字节(一位数字)

四、实验要求

实验前准备好汇编语言源程序，阅读实验指导书关于调试方法与步骤的内容。

五、编程提示

用T命令执行程序，可看出结果。

1、二进制双精度加法运算

计算X+Y=Z，将结果Z存入某存储单元。

1）实验程序如下：

```
STACK   SEGMENT STACK
        DW      64 DUP(?)
STACK   ENDS
DATA    SEGMENT
        XL      DW      ?           ;请在此处给 X 低位赋值
        XH      DW      ?           ;请在此处给 X 高位赋值
        YL      DW      ?           ;请在此处给 Y 低位赋值
        YH      DW      ?           ;请在此处给 Y 高位赋值
        ZL      DW      ?
        ZH      DW      ?
DATA    ENDS
CODE    SEGMENT
        ASSUME  CS:CODE,DS:DATA
START:  MOV     AX,DATA
        MOV     DS,AX
        MOV     AX,XL       ;X 低位送 AX
        ADD     AX,YL       ;X 低位加 Y 低位
        MOV     ZL,AX       ;存低位和
        MOV     AX,XH       ;X 高位送 AX
        ADC     AX,YH       ;X 高位加 Y 高位
        MOV     ZH,AX
A1:     JMP     A1
CODE    ENDS
        END START
```

本实验程序是双精度（2个16位，既32位）运算，利用累加器AX，先求低十六位和，并存入低址存储单元，后求高16位和，再存入高址存储单元。由于低位和可能向高位有进位，因而高位字相加语句需用ADC指令，则低位相加有进位时，CF=1，高位字相加时，同时加上CF中的1。

2）实验步骤

(1) 输入程序并检查无误，经汇编、连接后产生正确的可执行文件。

(2) 用 U CS:0000 查看 MOV AX,XXXX(DATA)语句，即得到数据段段地址 DS=XXXX，用 E 命令 E XXXX:0000↙给 XL，XH，YL 赋值存入二进制数 A0 65 15 00 和 9E B7 21 00。

(3) 单步运行以上程序到最后一条指令。

(4) 用 D 命令 D XXXX:0008↙，显示计算结果：3E 1D 37 00 CC…

(5) 反复试几组数，考察程序的正确性。

2、 十进制数的 BCD 码减法运算

计算 X-Y=Z，其中 X、Y、Z 为 BCD 码。实验程序及流程如下：

```
STACK   SEGMENT  STACK
        DW       64 DUP(?)
STACK   ENDS
DATA    SEGMENT
        X        DW       ?          ;请在此处给 X 赋值
        Y        DW       ?          ;请在此处给 Y 赋值
        Z        DW       ?
DATA    ENDS
CODE    SEGMENT
        ASSUME CS:CODE,DS:DATA
START:  MOV      AX,DATA
        MOV      DS,AX
        MOV      AH,00H
        SAHF
        MOV      CX,0002H
        LEA      SI, X
        LEA      DI, Z
A1:     MOV      AL,[SI]
        SBB      AL,[SI+02H]
        DAS
        PUSHF
        AND      AL,0FH
        POPF
        MOV      [DI],AL
        INC      DI
        INC      SI
        LOOPA1
A2:     JMP      A2
CODE    ENDS
        END START
```

实验步骤

(1) 输入程序并检查无误，经汇编、连接后产生正确的可执行文件。

(2) 用 U CS:0000 查看 MOV AX,XXXX(DATA)语句，即得到数据段段地址 DS=XXXX，用 E 命令 E XXXX：0000↙给 X、Y 赋值存入 40 和 22 的 BCD 码：00 04 02 02。

(3) 单步运行以上程序到最后一条指令。

(4) 用 D 命令 D XXXX:0004↙，显示计算结果：08 01 CC…

(5) 反复试几组数，考察程序的正确性。

3．乘法运算

本实验实现十进制数的乘法，被乘数、乘数和乘积均以 BCD 码形式存放在内存中，实验程序及流程如下：

```
STACK   SEGMENT     STACK
        DW      64 DUP(?)
STACK   ENDS

DATA    SEGMENT
        DATA1   DB      5 DUP(?)
        DATA2   DB      ?
        RESULT  DB      6 DUP(?)
DATA    ENDS
CODE    SEGMENT
        ASSUME CS:CODE,DS:DATA
START:  MOV     AX,DATA
        MOV     DS,AX
        CALL    INIT
        MOV     SI,OFFSET DATA2
        MOV     BL,[SI]
        AND     BL,0FH
        CMP     BL,09H
        JNC     ERROR
        MOV     SI,OFFSET DATA1
        MOV     DI,OFFSET RESULT
        MOV     CX,0005H
A1:     MOV     AL,[SI+04H]
        AND     AL,0FH
        CMP     AL,09H
        JNC     ERROR
        DEC     SI
        MUL     BL
        AAM
        ADD     AL,[DI+05H]
        AAA
        MOV     [DI+05H],AL
        DEC     DI
        MOV     [DI+05H],AH
```

```
        LOOP    A1
        MOV     CX,06H
        MOV     SI,OFFSET RESULT
DISPLAY:MOV     AH,01H
        MOV     AL,[SI]
        ADD     AL,30H
        MOV     [SI],AL
        INC     SI
        LOOP    DISPLAY
A2:     JMP     A2
INIT:   MOV     SI,OFFSET RESULT
        MOV     CX,0003H
        MOV     AX,0000H
A3:     MOV     [SI],AX
        INC     SI
        INC     SI
        LOOP    A3
        RET
ERROR:  MOV     AX,0145H
        JMP     A2
CODE    ENDS
        END   START
```

实验步骤:

(1) 输入程序并检查无误，经汇编、连接后产生正确的可执行文件。

(2) 用 U CS:0000 查看 MOV AX,XXXX(DATA)语句，即得到数据段段地址 DS=XXXX。

(3) 用 E 命令 E XXXX：0000↙在对应数据段填入乘数与被乘数。

(4) 单步运行程序到最后一条指令。检查结果。

(5) 反复试几组，考察程序的正确性。

六、思考题

1、编写有符号数 A1B1+A2B2 的程序，A1、A2、B1、B2 均为符号数。

2、编写两个数值长度不等的 BCD 码相加程序。

硬件部分

实验十一　即插即用配置资源的获取实验

一、实验目的和内容

掌握 PCI 总线设备的资源的方法

（注：做本试验前将实验箱上的所有连线撤除。）

此程序不需连线.

二、实验环境

1）硬件：PC 微机、SXL-100 B+型微机接口实验开发系统

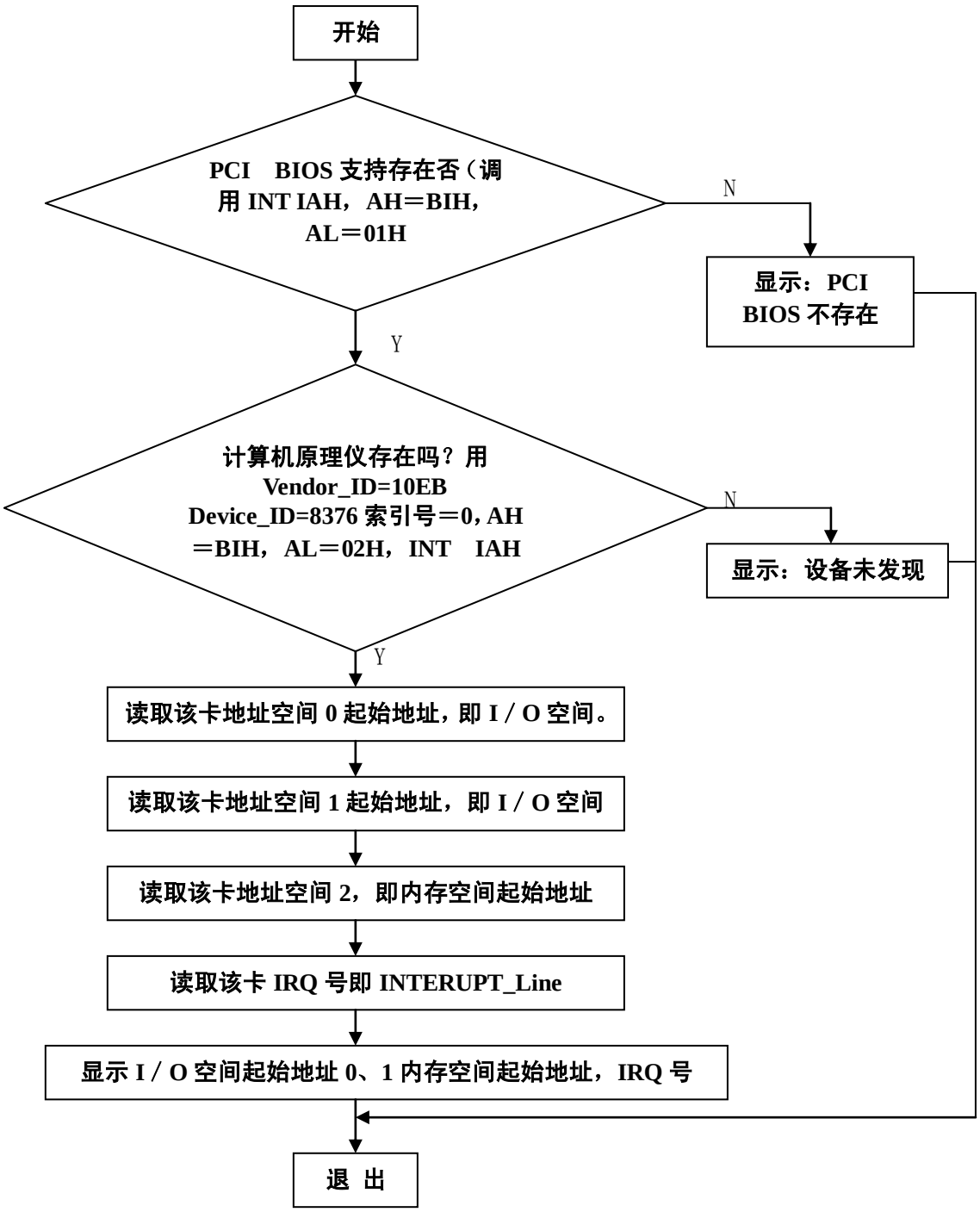
2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、编程提示

1、 PCI 设备资源寄存器地址分布

```
#define  PCI_CS_VENDOR_ID                0x00
#define  PCI_CS_DEVICE_ID                0x02
#define  PCI_CS_COMMAND                  0x04
#define  PCI_CS_STATUS                   0x06
#define  PCI_CS_REVISION_ID              0x08
#define  PCI_CS_CLASS_CODE               0x09
#define  PCI_CS_CACHE_LINE_SIZE          0x0c
#define  PCI_CS_MASTER_LATENCY           0x0d
#define  PCI_CS_HEADER_TYPE              0x0e
#define  PCI_CS_BIST                     0x0f
#define  PCI_CS_BASE_ADDRESS_0           0x10
#define  PCI_CS_BASE_ADDRESS_1           0x14
#define  PCI_CS_BASE_ADDRESS_2           0x18
#define  PCI_CS_BASE_ADDRESS_3           0x1c
#define  PCI_CS_BASE_ADDRESS_4           0x20
#define  PCI_CS_BASE_ADDRESS_5           0x24
#define  PCI_CS_EXPANSION_ROM            0x30
#define  PCI_CS_INTERRUPT_LINE           0x3c
#define  PCI_CS_INTERRUPT_PIN            0x3d
#define  PCI_CS_MIN_GNT                  0x3e
#define  PCI_CS_MAX_LAT                  0x3f
```

四、程序框图


开始
PCI BIOS 支持存在否（调用 INT IAH，AH＝BIH，AL＝01H
N
显示：PCI BIOS 不存在
Y
计算机原理仪存在吗？用 Vendor_ID=10EB Device_ID=8376 索引号＝0，AH＝BIH，AL＝02H，INT IAH
N
显示：设备未发现
Y
读取该卡地址空间 0 起始地址，即 I／O 空间。
读取该卡地址空间 1 起始地址，即 I／O 空间
读取该卡地址空间 2，即内存空间起始地址
读取该卡 IRQ 号即 INTERUPT_Line
显示 I／O 空间起始地址 0、1 内存空间起始地址，IRQ 号
退 出

五、程序代码　\ASM\9052_P~1\PCI-CO.asm

```
.MODEL SMALL
.DATA
DIS0 DB 'PCI CONFIG INFORMATION!$';提示信息
DIS1 DB 'PCI ADDRESS 0 $';
DIS2 DB 'PCI ADDRESS 1 $'
DIS3 DB 'PCI MEMORY ADDRESS $'
DIS4 DB 'PCI INTERRUPT LINE $'
DIS5 DB 'BIOS NOT SUPPER!$'
DIS6 DB 'READ PCI BOARD FAIL!$'
;-------PCI Configuration Space Registers-----------------------
PCI_CS_VENDOR_ID EQU 0
PCI_CS_DEVICE_ID EQU 2
PCI_CS_COMMAND    EQU 4
PCI_CS_STATUS     EQU 6
PCI_CS_REVISION   EQU 8
PCI_CS_CLASS_CODE  EQU 9
PCI_CS_CACHE_LINE_SIZE EQU 0CH
PCI_CS_MASTER_LATENCY EQU  0DH
PCI_CS_HEADER_TYPE EQU 0EH
PCI_CS_BIST       EQU 0FH
PCI_CS_BASE_ADDRESS_0 EQU 10H
PCI_CS_BASE_ADDRESS_1 EQU 14H
PCI_CS_BASE_ADDRESS_2 EQU 18H
PCI_CS_BASE_ADDRESS_3 EQU 1CH
PCI_CS_BASE_ADDRESS_4 EQU 20H
PCI_CS_BASE_ADDRESS_5 EQU 24H
PCI_CS_EXPANSION EQU 30H
PCI_CS_INTERRUPT_LINE EQU 3CH
PCI_CS_INTERRUPT_PIN EQU 3DH
PCI_CS_MIN_GNT EQU 3EH
PCI_CS_MAX_LAT EQU 3FH
;---------------------END-------------------------
ADDRESS_IO_0 DW ?
ADDRESS_IO_1 DW ?
ADDRESS_MEM_L  DW ?
ADDRESS_MEM_H  DW ?
INTERRUPT_LINE DB ?
.STACK 100H
.CODE
.STARTUP
MOV AL,3
```

```
MOV AH,0
INT 10H                          ;置显示方式
MOV DX,OFFSET DIS0               ;显示提示
MOV AH,9
INT 21H
MOV AH,2H
MOV BH,0
MOV DH,1
MOV DL,0
INT 10H                          ;置游标位置
MOV AH,0B1H
MOV AL,1H
INT 1AH
.IF AH!=00
MOV DX,OFFSET DIS5               ;显示 BIOS NOT SUPPER!
JMP m_EXIT
.ENDIF
MOV AH,0B1H
MOV AL,02H
MOV CX,8376H                     ; 设备 ID
MOV DX,10EBH                     ; 厂商 ID
MOV SI,0
INT 1AH                          ; 计算机接口仪存在??
JNC AA
MOV DX,OFFSET DIS1
MOV AH,9
INT 21H
JMP m_EXIT
AA:      MOV AH,0B1H
MOV AL,09H
MOV DI,PCI_CS_BASE_ADDRESS_1
INT 1AH                              ;读取该卡地址空间,0起始地址
.IF AH!=0
MOV DX,OFFSET DIS1
MOV AH,9
INT 21H
JMP m_EXIT
.ENDIF
AND CX,0FFFCH
MOV AX,CX
MOV ADDRESS_IO_0,AX
MOV AH,0B1H
MOV AL,09H
```

```
MOV DI,PCI_CS_BASE_ADDRESS_3       ;读取该卡地址空间,1起始地址
INT 1AH
.IF AH!=0
MOV DX,OFFSET DIS1
MOV AH,9
INT 21H
JMP m_EXIT
.ENDIF
AND CX,0FFFCH
MOV AX,CX
MOV ADDRESS_IO_1,AX
MOV AH,0B1H
MOV AL,09H
MOV DI,PCI_CS_BASE_ADDRESS_2    ;读取该卡地址空间,内存空间起始地址
INT 1AH
.IF AH!=0
MOV DX,OFFSET DIS1
MOV AH,9
INT 21H
JMP m_EXIT
.ENDIF
AND CX,0FFFCH
MOV AX,CX
MOV ADDRESS_MEM_L,AX
MOV AH,0B1H
MOV AL,09H
MOV DI,PCI_CS_BASE_ADDRESS_2
ADD DI,2
INT 1AH
.IF AH!=0
MOV DX,OFFSET DIS1
MOV AH,9
INT 21H
JMP m_EXIT
.ENDIF
MOV AX,CX
MOV ADDRESS_MEM_H,AX
MOV AH,0B1H
MOV AL,09H
MOV DI,PCI_CS_INTERRUPT_LINE          ;读取该卡 IRQ 号
INT 1AH

.IF AH!=0
```

```
MOV DX,OFFSET DIS1
MOV AH,9
INT 21H
JMP m_EXIT
.ENDIF
MOV AX,CX
MOV INTERRUPT_LINE,AL
MOV DX,OFFSET DIS1
MOV AH,9
INT 21H
MOV AX, ADDRESS_IO_0                    ;显示信息
CALL DISPLAY
CALL HDIS
MOV AH,2H
MOV BH,0
MOV DL,0
MOV DH,2
INT 10H
MOV DX,OFFSET DIS2
MOV AH,9
INT 21H
MOV AX, ADDRESS_IO_1                ;显示实验箱基地址
CALL DISPLAY
CALL HDIS
MOV AH,2H
MOV BH,0
MOV DL,0
MOV DH,3
INT 10H
MOV DX,OFFSET DIS3
MOV AH,9
INT 21H
MOV AX,ADDRESS_MEM_H                ; 显示内存空间起始地址高位(字)
CALL DISPLAY
MOV AX,ADDRESS_MEM_L                ; 显示内存空间起始地址低位(字)
CALL DISPLAY
CALL HDIS
MOV AH,2H
MOV BH,0
MOV DL,0
MOV DH,4
INT 10H
MOV DX,OFFSET DIS4
```

```
MOV AH,9
INT 21H
MOV AH,0
MOV AL,INTERRUPT_LINE                ; 显示中断号
CALL DISPLAY
CALL HDIS
JMP m_EXIT
DISPLAY PROC NEAR
PUSH CX
PUSH BX
MOV CL,4
MOV CH,4
DISPH1:                              ; 数字转换为 ASCII 码输出
ROL AX,CL
PUSH AX
AND AL,0FH
ADD AL,30H
CMP AL,'9'
JBE DISPH2
ADD AL,7
DISPH2:                              ; 显示输出
MOV AH,2
MOV DL,AL
INT 21H
POP AX
DEC CH
JNZ DISPH1
POP CX
POP BX
RET
DISPLAY ENDP
HDIS    PROC NEAR                    ; 如果显示 16 进制地址，在后面加 H
MOV DL, 'H'
MOV AH,2
INT 21H
RET
HDIS  ENDP
m_EXIT:                              ; 退出
MOV AX,4C00H
INT 21H
END
```

PCI 卡资源也可由以下方法获得:在 Windows98 中,进入系统属性—〉设备管理

器—〉计算机原理—〉属性—〉资源—〉内存范围（0、2），输入/输出（1、3）.
PCI address 0—9052 MEM
PCI address 1—9052 I/O 方式
PCI address 2—用户自定义 用户存储器
PCI address 3—用户自定义 用户 I/O 地址

程序的调试过程：

把程序\9052ASM\9052-P~1\PCI-CO.asm 及其他文件拷贝到硬盘上.并把只读属性去掉.

进入 DOS 环境,输入 mouse,即在 DOS 环境中使用鼠标.

输入 PWB 进入汇编集成环境.

在 PWB 环境中的菜单中的 options 选项下选择 Build options…弹出 Build options 对话框,选择 Build 框中的 debug。点击 Set Initial Build Options …进入 Set Initial Build Option 对话框,选中 DOS EXE,按 OK 退出 Set Initial Build Option 对话框,再按 OK 退出 Build options 对话框。

环境已经设置完毕,可以调试程序

操作如下：

1. 打开 PCI _CO.asm 文件

2. 点 Make 菜单选 Rebuild All 项（重新编译所有文件）

3. 如程序有误，则会弹出编译结果窗口，显示错误提示，全部改正完毕以后，再选

Make->Rebuild All 项，直至程序无错，这样，重建结束后，会弹出一个对话框，内有 5 个选项，这里选<Run Program>即可执行程序。

实验十二　简单 I / O 端口实验

一、实验目的和内容

1、掌握三态门，锁存器构成简单 I / O 端口的原理及应用

2、完成流水灯及其控制的编程实验

注：做试验前将前一试验的连线撤作除。

二、实验环境

1）硬件：PC 微机、SXL-100 B+型微机接口实验开发系统

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验电路及说明

实验电路图 1、2、3 所示，74ALS273 锁存器的输出控制 LED 发光管的亮暗（1：亮 0：暗）74ALS273 的 CLK 被一些位址线所控制，它的位址为 60H(16 进制)

ADD7, ADD6, ADD5, ADD4, ADD3, ADD2, ADD1, ADD0

0 1 1 0 0 0 0 0

只有这时 7430 与非门才输出低电平(八段 LED 是阴极型)

在 DOS 下，它的地址为 PCI 接口板的 I / O 地址加上 60H 为真实地址。如 PCI 的 I / O 起始地址 0 为 E400H , I / O 起始地址 1 为 E800H。 74ALS273 的地址为 E800H + 60H。

注意：PCI 板卡分配了 2 块 I / O 空间，I / O 空间 0 是分配给 PCI9052 内部寄存器使用的，即 PCI9052 专用，空间大小 128 BYTE(这由 9052 自身决定)。I / O 空间 1 是分配给 PCI 板卡用户电路使用的，其大小为 256 BYTE(这由实验箱的设计者决定)。

流水灯原理如下：

K1、K2 是八位拔动开关（位于区域 F）的从左起第 1，2 位。

K1＝K2＝H 发光管从上到下移位

K1＝L，K2＝H 发光管从下到上移位

KI＝H，K2＝L 发光管全部闪烁

K1＝K2＝L－退出 注：程序开始运行时，开关不能处于此位

注：ON 方向为 L，OFF 方向为 H

K1、K2 为主板上 J64 数码拔动开关的 1、2 位。

其中 I/O 基地址 O：为 9052 各控制寄存器的基地址，寻址各个 9052 的寄存器要以此基地址加上各个控制寄存器的偏移地址，控制寄存器的偏移地址请参看 9052db-20.pdf

I/O 地址 1：为接口仪实验箱上的元器件的基地址，寻址实验箱上各个器件，要以此基地址加上各个器件的偏移地址，各个器件的偏移地址见电路图中 main.sch。

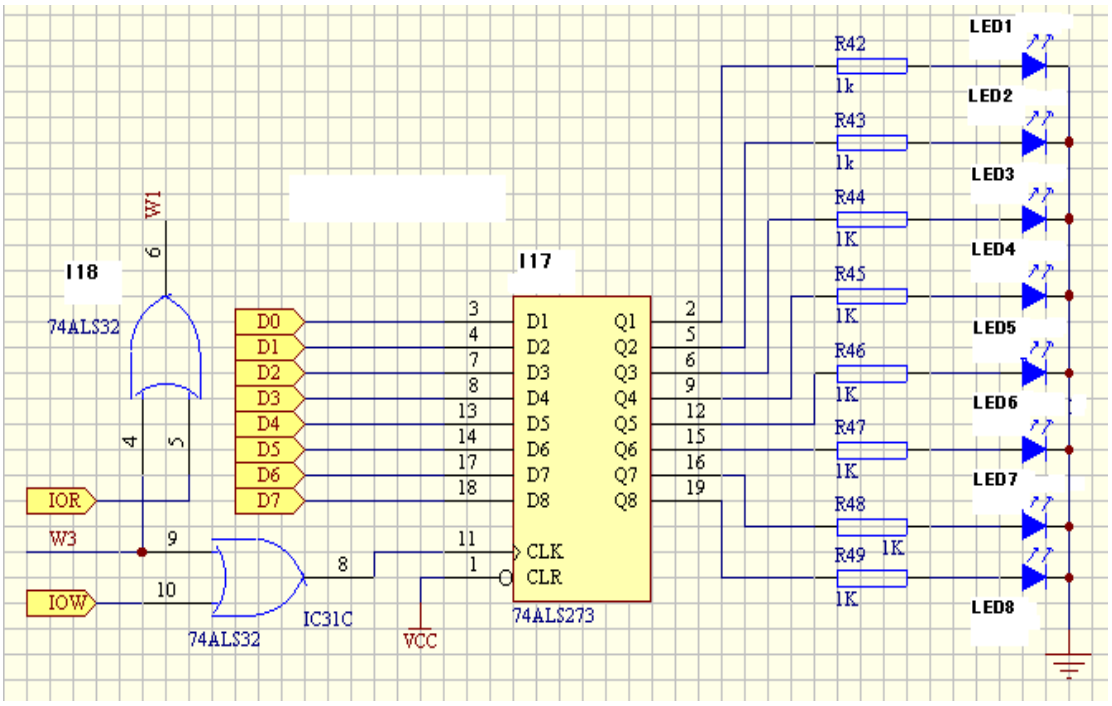

I18
74ALS32
W1
IOR
W3
IOW
74ALS32
IC31C
I17
D0
D1
D2
D3
D4
D5
D6
D7
D1
D2
D3
D4
D5
D6
D7
D8
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
CLK
CLR
74ALS273
VCC
R42
R43
R44
R45
R46
R47
R48
R49
1k
1K
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8

图 1

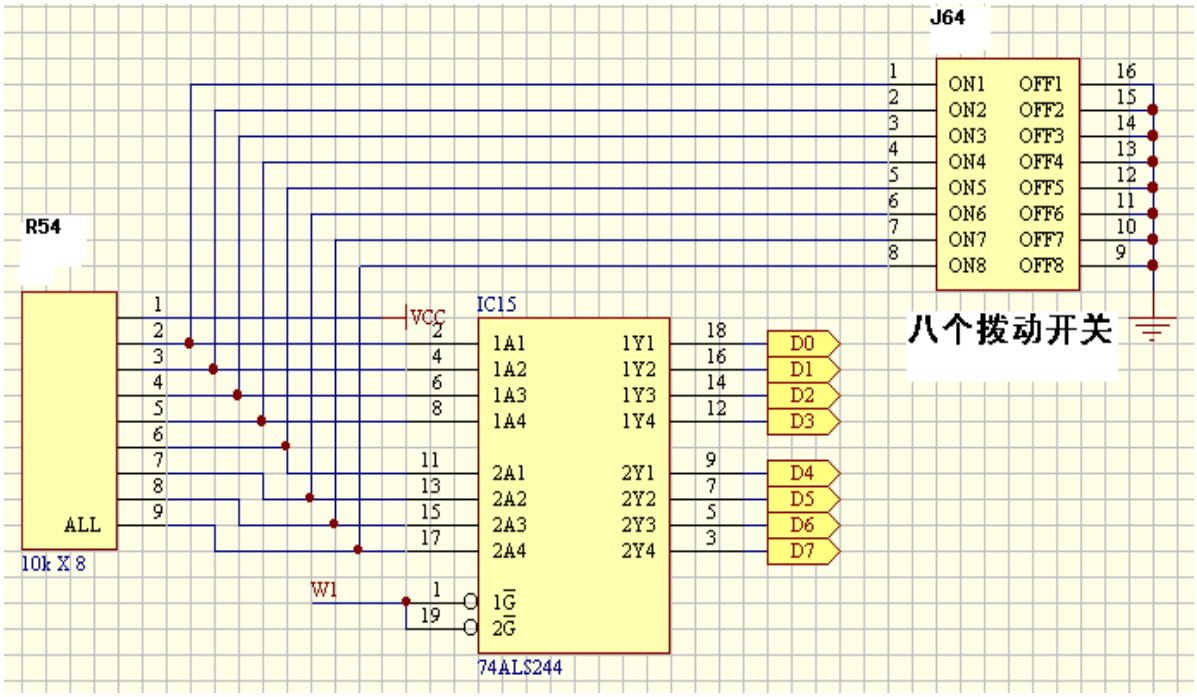

J64
ON1
ON2
ON3
ON4
ON5
ON6
ON7
ON8
OFF1
OFF2
OFF3
OFF4
OFF5
OFF6
OFF7
OFF8
八个拨动开关
R54
ALL
10k X 8
IC15
VCC
W1
1A1
1A2
1A3
1A4
2A1
2A2
2A3
2A4
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
2Y3
2Y4
1G
2G
74ALS244
D0
D1
D2
D3
D4
D5
D6
D7

图 2

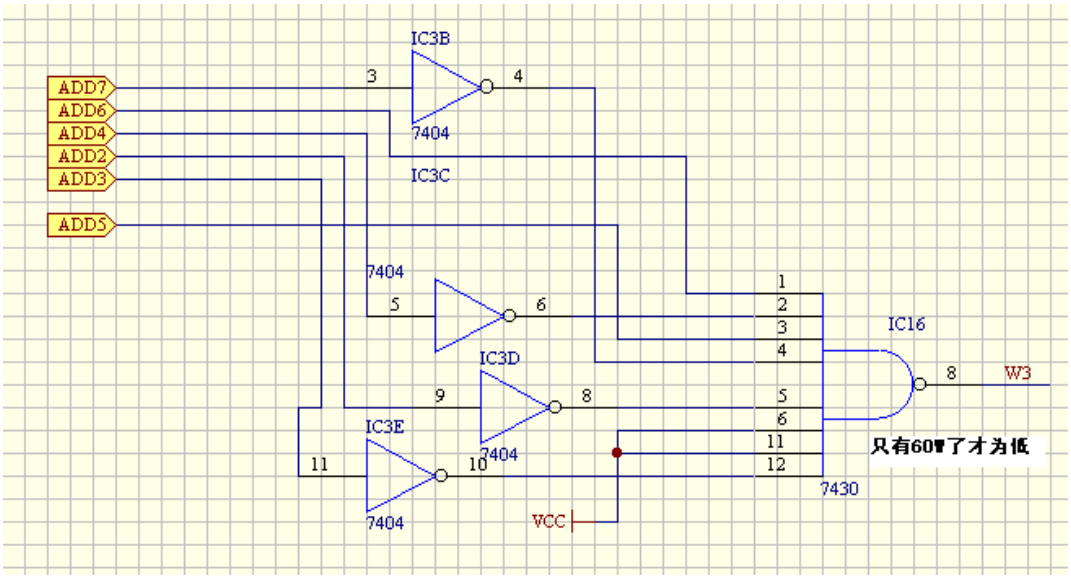
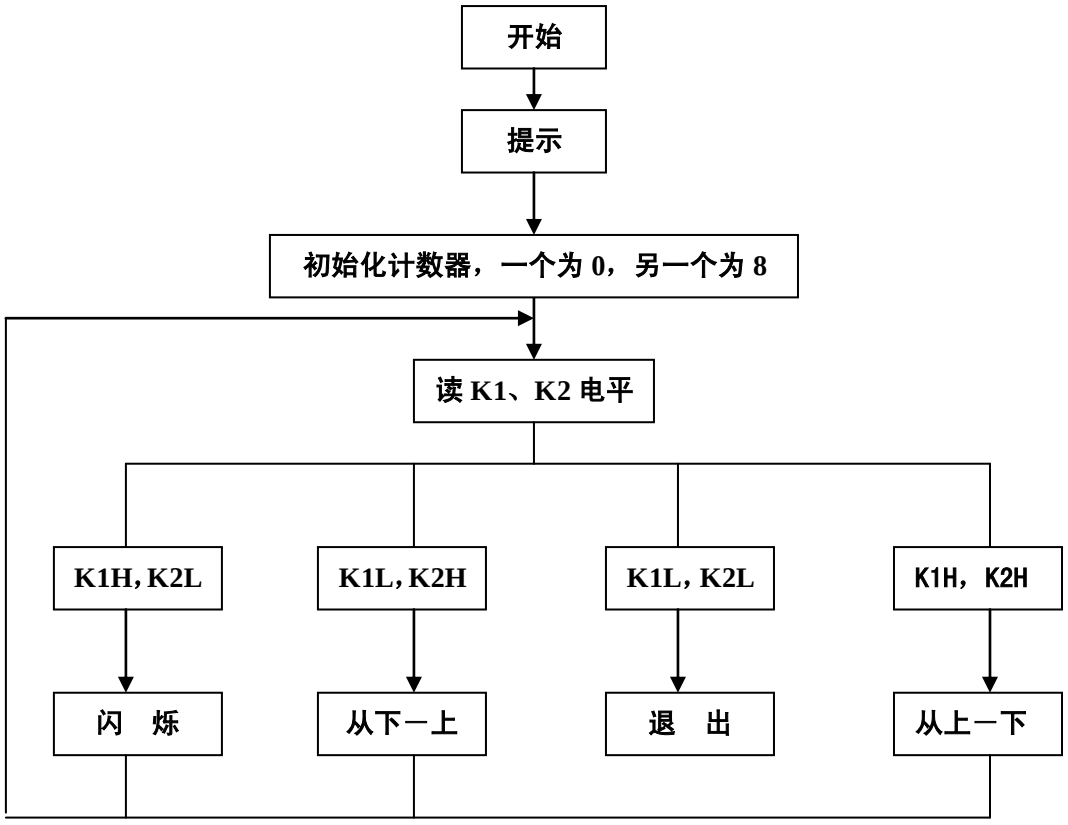
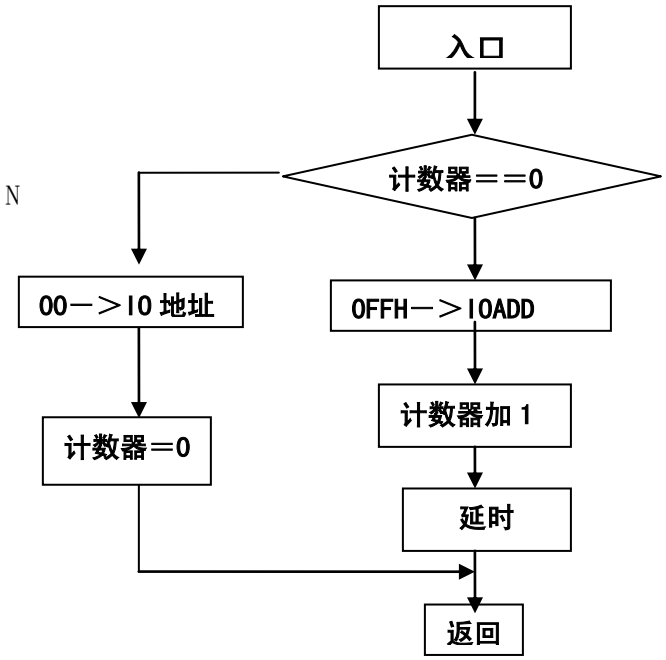

IC3B
3
4
7404
ADD7
ADD6
ADD4
ADD2
ADD3
IC3C
ADD5
7404
5
6
1
2
3
4
IC16
8
W3
IC3D
9
8
5
6
IC3E
7404
11
11
12
10
7404
7430
VCC
只有60T了才为低

图 3

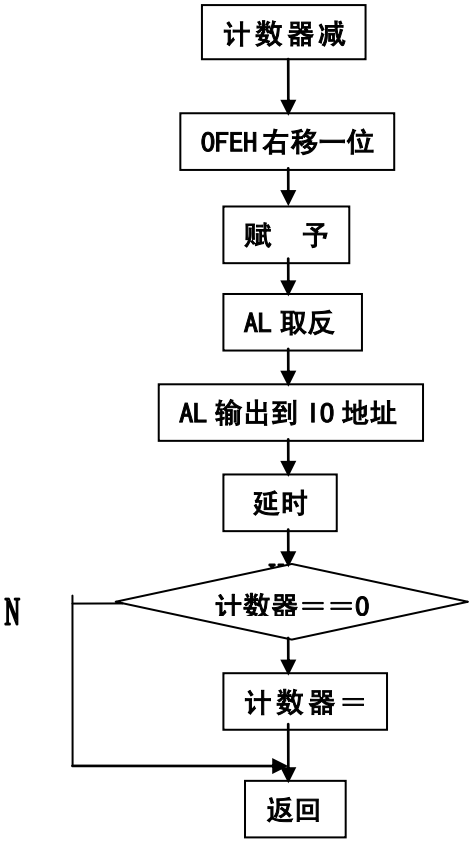
四、程序方框图


开始
提示
初始化计数器，一个为 0，另一个为 8
读 K1、K2 电平
K1H，K2L
K1L，K2H
K1L，K2L
K1H，K2H
闪　烁
从下一上
退　出
从上一下

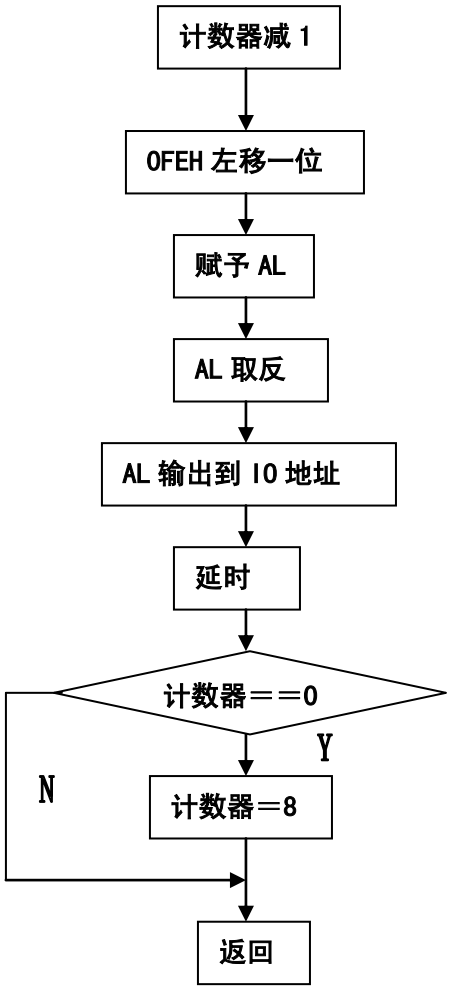
闪烁程序框图


入口
计数器==0
N
00->IO地址
OFFH->IOADD
计数器=0
计数器加1
延时
返回

从上->下


计数器减
OFEH右移一位
赋 予
AL取反
AL输出到IO地址
延时
计数器==0
N
计数器=
返回

从下－＞上



五、程序代码　ASM\IO\IO。asm

```
.MODEL SMALL
.stack 100h
.DATA
       DIS1 DB 'PCI ADDRESS 0 $';
       DIS2 DB 'PCI ADDRESS 1 $'
       DIS3 DB 'PCI MEMORY ADDRESS $'
       DIS4 DB 'PCI INTERRUPT LINE $'
       DIS5 DB 'BIOS NOT SUPPER!$'
       DIS6 DB 'READ PCI BOARD FAIL!$'
;-------PCI Configuration Space Registers-----------------
PCI_CS_VENDOR_ID EQU 0
PCI_CS_DEVICE_ID EQU 2
PCI_CS_COMMAND    EQU 4
PCI_CS_STATUS     EQU 6
```

```
PCI_CS_REVISION  EQU 8
PCI_CS_CLASS_CODE  EQU 9
PCI_CS_CACHE_LINE_SIZE EQU 0CH
PCI_CS_MASTER_LATENCY EQU  0DH
PCI_CS_HEADER_TYPE EQU 0EH
PCI_CS_BIST       EQU 0FH
PCI_CS_BASE_ADDRESS_0 EQU 10H
PCI_CS_BASE_ADDRESS_1 EQU 14H
PCI_CS_BASE_ADDRESS_2 EQU 18H
PCI_CS_BASE_ADDRESS_3 EQU 1CH
PCI_CS_BASE_ADDRESS_4 EQU 20H
PCI_CS_BASE_ADDRESS_5 EQU 24H
PCI_CS_EXPANSION EQU 30H
PCI_CS_INTERRUPT_LINE EQU 3CH
PCI_CS_INTERRUPT_PIN EQU 3DH
PCI_CS_MIN_GNT EQU 3EH
PCI_CS_MAX_LAT EQU 3FH
;--------------------END-------------------------
ADDRESS_IO_0 DW ?
ADDRESS_IO_1 DW ?
ADDRESS_MEM_L  DW ?
ADDRESS_MEM_H  DW ?
INTERRUPT_LINE DB ?
IOAD       DW   60H            ;定义 I/O 器件的地址的偏移量
M_BIT    DB       8
DD1    DB  0FEH
DD2    DB  07FH
MES  DB  'PRESS K1 AND K2 TO LOW FOR QUIT $'
D1    DB  0
.CODE
.STARTUP
START:
MOV      AX,DATA
MOV      DS,AX
MOV      AX,DATA
MOV      ES,AX
MOV      AX,SSREG
MOV     AH,0B1H                      ;看 PCI BIOS 是否存在
MOV     AL,1H
INT     1AH
.IF      AH!=00
JMP     m_EXIT
.ENDIF
```

```
MOV AH,0B1H                          ;查找 PCI 卡的位置
MOV AL,02H
MOV CX,8376H
MOV DX,10EBH
MOV SI,0
INT 1AH
JNC AA
JMP m_EXIT
AA:      MOV AH,0B1H                 ;读取配置寄存器的板卡基地址
MOV AL,09H
MOV DI,PCI_CS_BASE_ADDRESS_1
INT 1AH
.IF AH!=0
JMP m_EXIT
.ENDIF
AND CX,0FFFCH
MOV AX,CX
MOV ADDRESS_IO_0,AX
MOV AH,0B1H                          ;读取配置寄存器的用户基地址
MOV AL,09H
MOV DI,PCI_CS_BASE_ADDRESS_3
INT 1AH
.IF AH!=0
JMP m_EXIT
.ENDIF
AND CX,0FFFCH
MOV AX,CX
MOV ADDRESS_IO_1,AX
ADD IOADD,AX                         ; 清屏
MOV AL,3
MOV AH,0
INT 10H
MOV AH,9
MOV DX,OFFSET MES
INT 21H
SCAN:
MOV     AX,0                         ;扫描输入信号 —开关值
MOV     DX,IOADD
IN      AL,DX
AND     AL,3H                        ;保留低两位
.IF     AL==0                        ;判断低两位的值，调用相应的子程序
CALL   M_K0
.ENDIF
```

```
.IF     AL==1
CALL     M_K1
.ENDIF
.IF     AL==2
CALL     M_K2
.ENDIF
.IF     AL==3
CALL     M_K3
.ENDIF
JMP     SCAN
M_K0    PROC   FAR                    ;使 LED 全亮后退出程序
MOV   DX,IOADD
MOV   AL,0FFH
OUT   DX,AL
MOV AX,4C00H
INT 21H
RET
M_K0   ENDP
M_K1    PROC    FAR
.IF   D1==0                    ;若标志位 D1 为 0，则使发光管全亮并延时
MOV       DX,IOADD
MOV       AL,0FFH
OUT       DX,AL
CALL      TIME
MOV       D1,1
.ELSE                         ;若标志位 D1 非 0，则使发光管全灭并延时
MOV       DX,IOADD
MOV       AL,0H
OUT       DX,AL
CALL      TIME
MOV       D1,0
.ENDIF
RET
M_K1    ENDP
M_K2    PROC    FAR
DEC       M_BIT               ;使 DD1 资料左移一位，并用发光管显示
ROL       DD1,1
MOV       AL,DD1
NOT       AL
MOV       DX,IOADD
OUT       DX,AL
CALL      TIME
.IF       M_BIT==0            ;若已移动 8 位，则从头开始移位元并显示
```

```
MOV        DD1,0FEH
MOV        M_BIT,8
.ENDIF
RET
M_K2     ENDP
M_K3     PROC    FAR
DEC      M_BIT              ;使 DD1 资料右移一位，并用发光管显示
ROR      DD1,1
MOV      AL,DD1
MOV      DX,IOADD
NOT      AL
OUT      DX,AL
CALL     TIME
.IF      M_BIT==0           ;若已移动 8 位，则从头开始移位元并显示
MOV      DD1,0FEH
MOV      M_BIT,8
.ENDIF
RET
M_K3     ENDP
TIME     PROC NEAR          ;延时子程序
MOV AL,0FFH
T2:    DEC AL
MOV BX,0FFFFH
T1:    DEC BX
JNZ T1
.IF AL!=0
JMP T2
.ENDIF
RET
TIME ENDP
m_EXIT:                     ;退出程序
MOV AX,4C00H
INT 21H
END
```

实验十三　可编程中断控制器 8259A 实验（一）

中断是计算机处理外部事件和内部异常的一种重要机制，有关的控制逻辑是当今高档微机中不可缺少的硬件支持，8259A 是 PC 系列微机和单片机系统中应用最为广泛的可编程中断控制器之一。

一、实验目的

1．掌握 8259A 中断控制器的工作原理，熟悉实验中涉及的中断屏蔽寄存器 IMR 和中断服务寄存器 ISR 等的使用方法；

2．学会中断处理程序的编写。

二、实验环境

1）硬件：PC 微机、SXL-100 B+型微机接口实验开发系统

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容与要求

用 8253A 的输出作为中断申请信号使 8259A 产生中断，每中断一次在 CRT 上显示一串提示字符。

1．使用 PC 机内的 8259A 芯片，IRQ2 输入，中断类型号为 0AH；

2．使用装置中的 8253A 为中断源，每隔 1S 中断一次；

3．要求主机响应外部中断 IRQ2 时，显示字符串“THIS IS A 8259A INTERRUPT!”，且中断十次后，程序退出；

按图 4 接线：

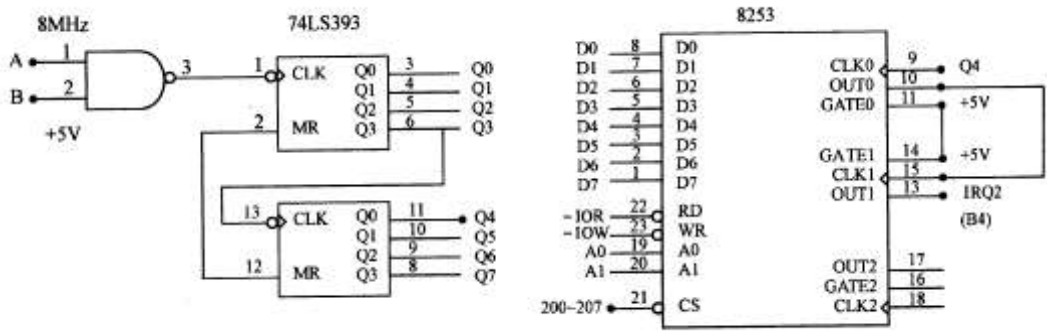


图 4　8259A 中断控制器连线图

四、编程提示

1．机内 8259A 初始化为普通中断结束方式，因此，外中断结束时，必须使用中断结束命令来清除中断服务寄存器 ISR 中的对应位。

2．中断十次后，程序退出前，关闭 IRQ2 中断，即给 IMR 中相应位置“1”禁止中断。

3．机内 8259A 地址：偶地址为 20H，奇地址为 21H。

4．中断处理子程序框图如图 5：

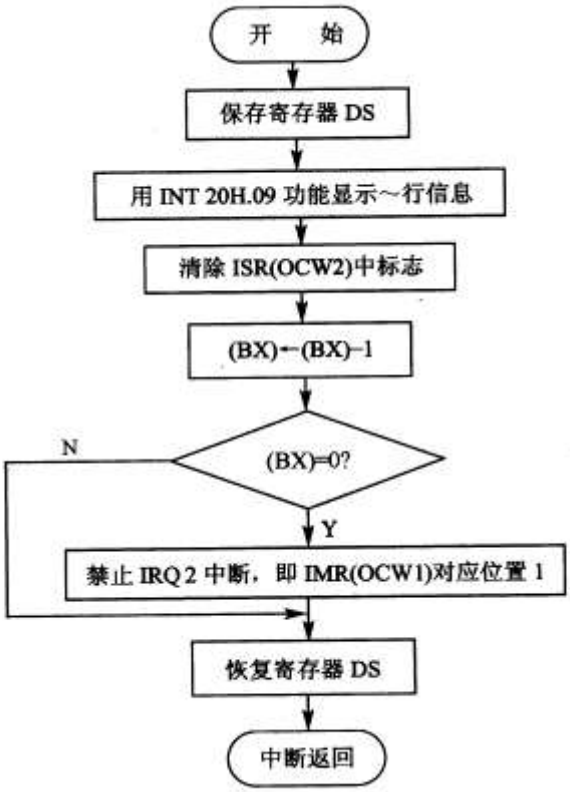


图 5　中断处理子程序框图

五、程序清单

```
INTA00          EQU     20H
INTA01          EQU     21H
TIM_CTL         EQU     203H
TIMER0          EQU     200H
TIMER1          EQU     201H
MODE03          EQU     36H
MODE12          EQU     54H

DATA            SEGMENT
MESS            DB      ‘THIS IS A 8259A INTERRUPT!’，0AH，0DH，’$’
FLAG            DB      0
INTMASK         DB      0
CSREG           DW      ?
IPREG           DW      ?
DATA            ENDS

STACK           SEGMENT
STA             DB      50 DUP (?)
TOP             EQU     LENGTH  STA
STACK           ENDS
```

```
CODE            SEGMENT
ASSUME          CS：CODE，  DS：DATA，  SS：STACK

START：         CLI
                MOV         AX，  DATA
                MOV         DS，  AX
                MOV         DX，    TIM_CTL
                MOV         AL，    MODE03
                OUT         DX，    AL
                MOV         DX，    TIMER0
                MOV         AL，    00H
                OUT         DX，    AL
                MOV         AL，    02H
                OUT         DX，    AL
                MOV         DX，    TIM_CTL
                MOV         AL，    MODE12
                OUT         DX，    AL
                MOV         DX，    TIMER1
                MOV         AL，    0AH
                OUT         DX，    AL
;
                MOV         AX，    STACK
                MOV         SS，    AX
                MOV         SP，    TOP
                MOV         AX，    350AH
                INT         21H
                MOV         AX，    ES
                MOV         CSREG，AX
                MOV         IPREG，  BX
                PUSH        DS
;
                MOV         AX，    CS
                MOV         DS，    AX
                MOV         DX，    OFFSET INT_PROC
                MOV         AX，    250AH
                INT         21H
; ▼
                POP         DS
                MOV         DX，    INTA01
                IN          AL，    DX
                MOV         INTMASK，  AL
                AND         AL，    0FBH
                OUT         DX，    AL
```

```
;
                MOV     BX，    10
                STI
;

LL:             MOV     AL，    FLAG
                CMP     AL，    01H
                JNZ     LL
;
                CLI
                MOV     AL，  INTMASK
                MOV     DX，  INTA01
                OUT     DX，  AL
                MOV     DX，  IPREG
                MOV     AX，  CSREG
                MOV     DS，  AX
                MOV     AX，  250AH
                INT     21H
                STI
                MOV     AX，  4C00H
                INT     21H
;
INT_PROC:       PUSH    DS
                MOV     AX，  DATA
                MOV     DS，  AX
                MOV     DX，  OFFSET MESS
                MOV     AH，  09
                INT     21H
                MOV     DX，  INTA00
                MOV     AL，  20H
                OUT     DX，  AL
                DEC     BX
                JNZ     NEXT
                MOV     AL，  01
                MOV     FLAG，  AL
                MOV     DX，  INTA01
                IN      AL，  DX
                OR      AL，  04H
                OUT     DX，  AL
NEXT:           POP     DS
                IRET
CODE            ENDS
END     START
```

六、思考题

1．在编程过程中，用到了哪些操作命令字？请在程序中注明。

2．程序执行到▼处，检查 0：28H～0：2BH 的内容，并记录。

3．说明变量 FLAG 的作用。

4．如将中断源改为外部脉冲输入方式，应如何改接连线？如何修改程序?

实验十四　可编程中断控制器 8259A 实验（二）

一、实验目的

a．掌握中断的基本概念

b．了解 8259 的工作原理以及连接方法

注：做试验前将前一试验的连线撤除

二、实验环境

1）硬件：PC 微机、SXL-100 B+型微机接口实验开发系统

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

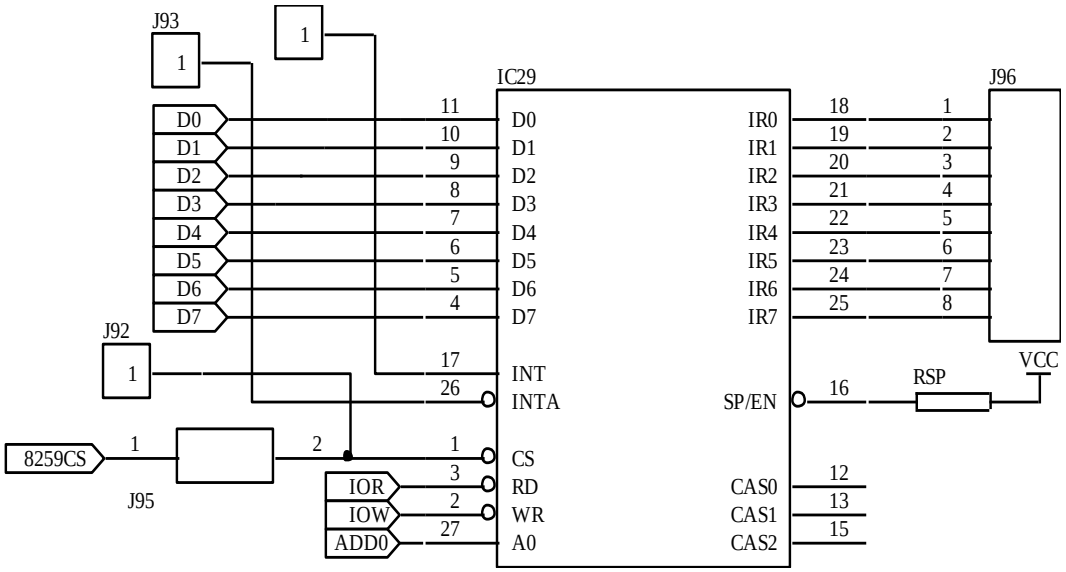
三、实验内容

8259 SP 经电阻接高电平，为单片工作方式，INTA 信号由 138 产生（0xA0），中断服务程序内，对 0xA0 地址读操作，产生 80X86 CPU 的中断响应信号，在 8259 设置为 8085 方式时，须产生 3 个 INTA 信号，第一个脉冲使得 8259 送出 CALL 的机器码(CDH),其后两个使 8259 送出中断服务程序的地址，应用程序收到此地址后，既可转至相应的中断处理程序。

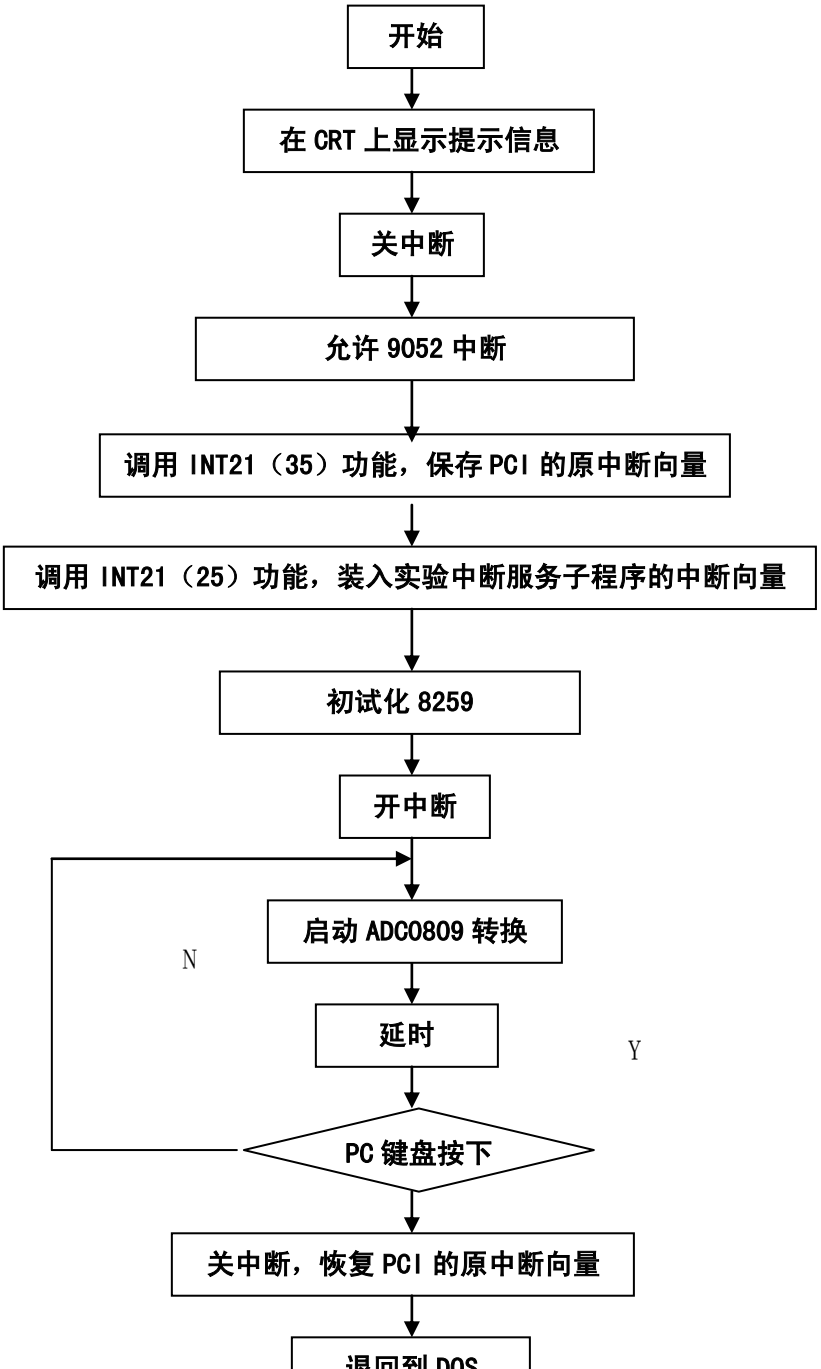
本实验主要通过 8259 控制 AD 转换的中断申请。先通过电位器将 VCC 分压后，送入 0809 转换成数字量，转换完成时会使 0809 的 EOC 端产生中断请求信号，通过 INT0 送入 8259 的 IR0 端，由 8259 向 PCI 发送中断申请，由 PCI 允许后向计算机发出申请。由计算机执行中断，并显示转换结果。

实验联线方式：C1 区 J70（EOC）端与 H 区 J96 的 INT0 端相连，H 区 J94（8259INT）与 A 区 J19 的 IRQ 端相连。

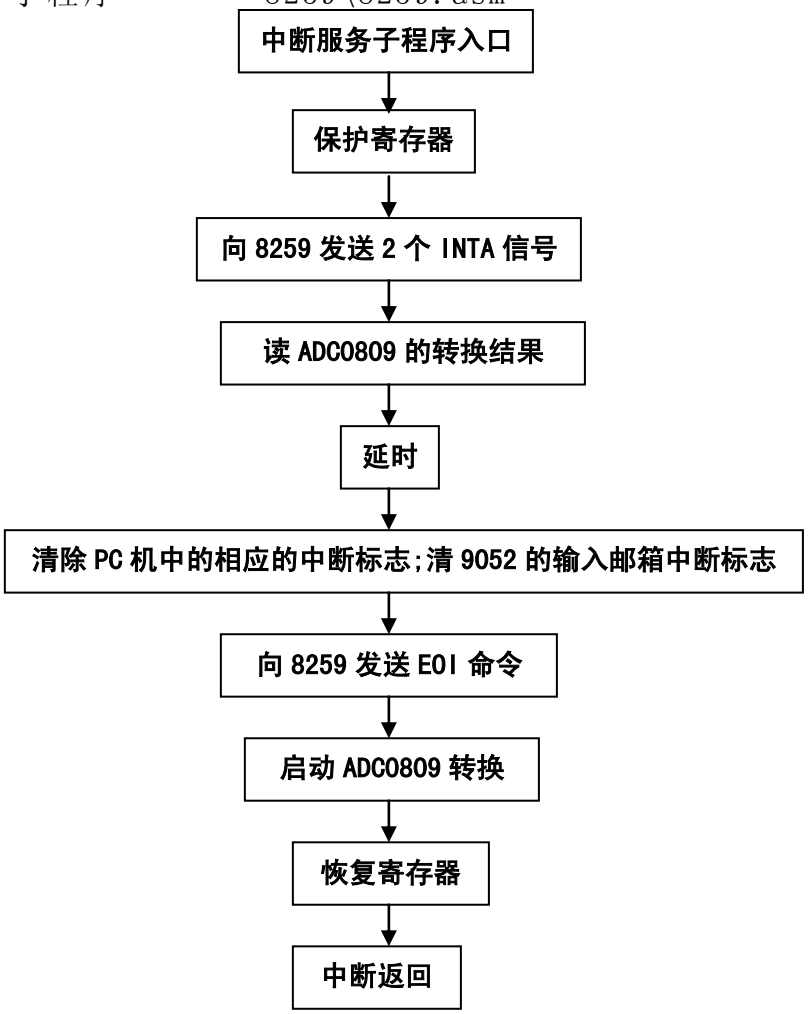
电路图如下所示：


J93
1
1
IC29
J96
D0
D1
D2
D3
D4
D5
D6
D7
11
10
9
8
7
6
5
4
D0
D1
D2
D3
D4
D5
D6
D7
IR0
IR1
IR2
IR3
IR4
IR5
IR6
IR7
18
19
20
21
22
23
24
25
1
2
3
4
5
6
7
8
J92
1
17
26
INT
INTA
SP/EN
16
RSP
VCC
8259CS
1
J95
2
1
CS
IOR
IOW
ADD0
3
2
27
RD
WR
A0
CAS0
CAS1
CAS2
12
13
15

四、程序框图


开始
在 CRT 上显示提示信息
关中断
允许 9052 中断
调用 INT21（35）功能，保存 PCI 的原中断向量
调用 INT21（25）功能，装入实验中断服务子程序的中断向量
初试化 8259
开中断
启动 ADC0809 转换
N
延时
Y
PC 键盘按下
关中断，恢复 PCI 的原中断向量

中断服务子程序　　8259\8259.asm



程序源代码如下：

```
DATA SEGMENT
CSREG    DW       ?                ;原中断保护地址
IPREG  DW  ?
```

```
    mmDB ?                              ;中断有无产生标志
      P8259A0    EQU   0090h            ;8259 地址
      COUNT  DB    0
      P8259B0   EQU    0a0H
    DIS0 DB '0809 interrupt  PRESS ANY OF THE KEYS TO QUIT!$';提示信息
      DIS1 DB 'INTERRUT OVER!!!$'
     IC_ADC0809    DW    80H             ;0809 地址
     m_d1  db ?                          ;0809 值存放地址
     m_x       DB 0                      ;游标位置
     m_y       DB 35
;-------PCI Configuration Space Registers-----------------
   PCI_CS_BASE_ADDRESS_1 EQU 14H         ;配置寄存器的板卡基地址
   PCI_CS_BASE_ADDRESS_3 EQU 1CH         ;配置寄存器的用户基地址
   PCI_CS_INTERRUPT_LINE EQU 3CH         ;配置寄存器的中断线
   ADDRESS_IO_0 DW ?                     ;板卡基地址
   ADDRESS_IO_1 DW ?                     ;用户基地址
   INTERRUPT_LINE DB ?                   ;中断线
   INTMASK DB        ?
   INTCSR_9052 DW 4cH                    ;9052 中断寄存器地址
   INT_VECT         DB      0            ;存放中断号
DATA ENDS
SSREG    SEGMENT STACK
QSTA      DB     100DUP(?)
TOP      EQU      LENGTH STA
SSREG ENDS
CODE     SEGMENT 'CODE'
   ASSUME CS:CODE,DS:DATA,SS:SSREG
START:  CLI
   MOV       AX,DATA
   MOV       DS,AX
   MOV       AX,DATA
   MOV       ES,AX
   MOV       AX,SSREG
   MOV AH,0B1H                          ;看 PCI BIOS 是否存在
   MOV AL,1H
   INT 1AH
   .IF AH!=00
   JMP m_EXIT
         .ENDIF
         MOV AH,0B1H                    ;查找 PCI 卡的位置
         MOV AL,02H
         MOV CX,8376H
         MOV DX,10EBH
```

```
        MOV SI,0
        INT 1AH
   JNC AA
        JMP m_EXIT
AA:     MOV AH,0B1H                ;读取配置寄存器的板卡基地址
        MOV AL,09H
   MOV DI,PCI_CS_BASE_ADDRESS_1
        INT 1AH
   .IF AH!=0
        JMP m_EXIT
        .ENDIF
   AND CX,0FFFCH
   MOV AX,CX
   MOV ADDRESS_IO_0,AX
        MOV AH,0B1H       ;读取配置寄存器的用户基地址
        MOV AL,09H
   MOV DI,PCI_CS_BASE_ADDRESS_3
        INT 1AH
   .IF AH!=0
        JMP m_EXIT
        .ENDIF
   AND CX,0FFFCH
   MOV AX,CX
   MOV ADDRESS_IO_1,AX

   MOV AH,0B1H               ;读取配置寄存器的中断号
   MOV AL,09H
        MOV DI,PCI_CS_INTERRUPT_LINE
        INT 1AH
   .IF AH!=0
        JMP m_EXIT
        .ENDIF
   MOV AX,CX
   MOV INTERRUPT_LINE,AL
        MOV     AL,3       ;清屏
        MOV     AH,0
        INT     10H
        MOV     DX,OFFSET DIS0
   MOV     AH,9
   INT 21H
        MOV     AH,3     ;读游标位置
   MOV     BH,0
   INT     10H
```

```
MOV       m_x,DL
MOV m_y,DH
MOV BL,INTERRUPT_LINE   ;中断处理(>8 从片  8--f,<8 主片  70--77)
        .IF       BL<8
        MOV       AL,INTERRUPT_LINE
        ADD       AL,8H
        MOV       INT_VECT,AL
        .ELSE
        MOV       AL,INTERRUPT_LINE
        SUB       AL,8H
        ADD       AL,70H
        MOV       INT_VECT,AL
.ENDIF
MOV  BL,INT_VECT        ;保护原中断
MOV  AH,35H
MOV  AL,BL
INT  21H
MOV  AX,ES
MOV  CSREG,AX
MOV  IPREG,BX
 MOV       BL,INT_VECT       ;产生要处理中断的向量
PUSH      DS
MOV       AX,CS
MOV       DS,AX
 MOV       DX,OFFSET  INT_PROC
MOV       AH,25H
MOV       AL,BL
INT       21H
        POP       DS
        MOV BL,INTERRUPT_LINE   ;判断中断为主,从片然后送屏蔽字
        .IF       BL<8
        IN        AL,21H
        MOV       CL,BL
        MOV       BL,1
        SHL       BL,CL
        NOT       BL
        AND       AL,BL
        MOV       DX,21H
        OUT       DX,AL
        .ELSE
IN   AL,0A1H
MOV  INTMASK,AL
        SUB       BL,8H
```

```
          MOV       CL,BL
          MOV       BL,1
          SHL       BL,CL
          NOT       BL
          AND       AL,BL
          MOV       DX,0A1H
          OUT       DX,AL
   .ENDIF
   STI
LP:    MOV AX,ADDRESS_IO_1      ;初始化 8259
        MOV       DX,AX
        MOV   AX,P8259A0
        ADD   DX,AX
     MOV       al,17h
        OUT   DX,AL
        MOV   AX,ADDRESS_IO_1
        MOV       DX,AX
        MOV   AX,P8259A1
        ADD   DX,AX
     mov       al,10H
        out   dx,al
     MOV       AL,03h
        OUT   DX,AL
     MOV       AL,0fch
        OUT   DX,AL
TT1:   MOV AX,ADDRESS_IO_1      ;启动 0809 转换
        MOV       DX,AX
        MOV       AX,IC_ADC0809
     ADD       DX,AX
     mov       al,0
        OUT   DX,AL
        MOV   mm,0H
TT2:   CALL   TIME              ;等待中断
          .IF       mm==0
          CALL      PCKEY
   JMP TT2
          .ENDIF
          CALL      DISNULL
          MOV       AH,0
          MOV       AL,m_d1
          CALL      DISPLAY
          CALL      PCKEY
        JMP   TT1
```

```
        MOV       AX,4C00H
        INT       21H
     JMP       INT_PROC
PCKEY    PROC     NEAR                ;按任意键退出
   MOV       AH,6
   MOV       DL,0FFH
   INT       21H
   JE  P1
   CLI
   MOV AL,INTMASK
   MOV DX,0A1H
   OUT DX,AL
   MOV DX,IPREG
   MOV AX,CSREG
   MOV DS,AX
   MOV AX,250DH
   INT 21H
   STI
   MOV       AX,4C00H
   INT       21H
P1:      RET
PCKEY    ENDP
TIME      PROC NEAR                  ;延时子程序
       MOV Ax,04FFH
T2:    DEC Ax
       MOV BX,0FFFFH
T1:    DEC BX
       JNZ T1
       .IF Ax!=0
       JMP T2
       .ENDIF
       RET
TIME ENDP
DISPLAY PROC NEAR                 ;显示子程序
   PUSH DX
         PUSH CX
         PUSH BX
         MOV CX,0
         MOV BX,10
DISPX1:
         MOV DX,0
         DIV BX
         PUSH DX
```

```
          INC CX
          OR AX,AX
          JNZ DISPX1
DISPX2:
          POP DX
          MOV AH,6
          ADD DL,30H
          INT 21H
          LOOP DISPX2
          POP BX
          POP CX
          POP DX
      RET
DISPLAY ENDP
DISNULL PROC FAR                ;使不显示的位元为空格
   MOV AH,2
   MOV      BH,0
   MOV      DL,m_x
   MOV      DH,m_y
   INT      10H
   MOV BL,4
D1:   DEC BL
   .IF BL!=0
   MOV AH,6
   MOV DL,' '
   INT 21H
   JMP D1
         .ENDIF
   MOV AH,2
   MOV      BH,0
   MOV      DL,m_x
   MOV      DH,m_y
   INT      10H
         RET
DISNULL  ENDP

INT_PROC:                       ;中断子程序
   PUSH   DS
   MOV DX,ADDRESS_IO_0
   MOV AX,INTCSR_9052           ;屏蔽 9052 板卡中断
   ADD DX,AX
   ADD DX,1
   IN  AL,DX
```

```
    MOV BL,0cH
    OR  AL,BL
    OUT DX,AL
    ;-------------------------
    MOV AX,ADDRESS_IO_1            ;启动 0809 转换
    MOV       DX,AX
           MOV       AX,P8259b0
    ADD DX,AX
              in        DX,AL
           in        dx,al
           in        dx,al
    ;-------------------------
MOV    AL,60H
MOV       DX,0A0H
    MOV  BL,INTERRUPT_LINE          ;送 EOI
    .IF  BL<8
    AND BL,7H
    OR   BL,60H
    MOV  AL,BL
    MOV  DX,20H
    OUT  DX,AL
    .ELSE
    SUB  BL,8
    MOV  AL,BL
    AND  AL,7H
    mov dx,0a0h
    OR  AL,60H
    OUT  DX,AL
    MOV       DX,20H
    MOV AL,62H
    OUT       DX,AL
    .ENDIF
    POP       DS
    MOV mm,10                                ;中断标志
MOV       AX,ADDRESS_IO_1                    ;读 0809 转换值
    MOV       DX,AX
    MOV       AX,IC_ADC0809
    ADD       DX,AX
    IN        AL,DX
    MOV m_d1,AL

    MOV AH,2
    MOV BH,0
```

```
    MOV DL,m_x
    MOV       DH,m_y
    INT 10H
    IRET
m_EXIT:
    MOV AX,4C00H
    INT 21H
    CODE      ENDS
    END       START
```

实验十五　可编程计数器/定时器 8253（一）

可编程计数器/定时器（8253）既可作为计数器，又可作为定时器。它有 3 个独立编程的计数器 0、计数器 1、计数器 2，它们均可独立地作为计数器和定时器。每个计数器都有 6 种工作方式，每种工作方式都是靠方式字来控制，从而产生不同方式的输出信号。

一、实验目的

1．加深对 8253 基本原理及工作方式的了解；

2．掌握 8253 初始化编程的方法；

3．学会通过 62 芯总线驱动器、译码器等线路，在微机外部扩展新的芯片。

二、实验环境

1）硬件：PC 微机、SXL-100 B+型微机接口实验开发系统

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容与要求

1、使用 8253 的 CNT0 构成一个方波发生器，CLK0=250KHZ ，OUT0=500HZ，再将该信号通过 CNT1 进行 10 分频，得到频率为 50HZ ，脉宽为 2ms 的波形。

2、按下图完成下述连接：

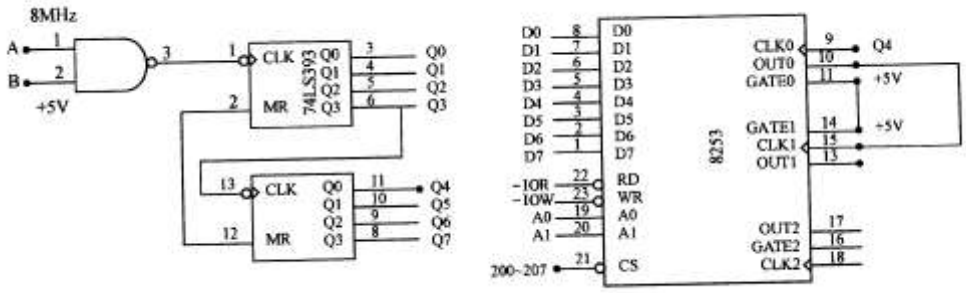


图 6　8253 计数器/定时器连线图

1）将 B 点连接+5V，A 点连接 8MHZ ；

2）将 8253 的 CLK0 与 Q4 相连，OUT0 与 CLK1 相连；

3）将 GATE0、GATE1 与+5V 相连；

4）连接 8253 片选信号 $\overline{CS}$；

5）接通电源。

3、按要求将下列程序补充完整：

1）CNT0 设为方式 3（方波发生器），计算初值；

2）CNT1 设为方式 2（10 分频）；

3）改变方波宽度和分频信号周期，在示波器中观察计数器 0 和计数器 1 的输出波形及其关系。

四、程序清单

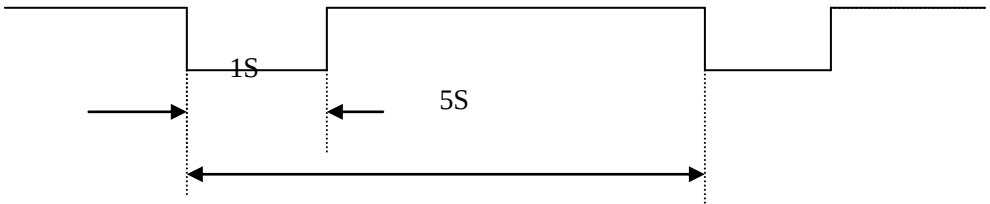
```
TIM_CTL   EQU    203H
TIMER0    EQU    200H
TIMER1    EQU    201H

DATA        SEGMENT
MESS   DB '8253A TIMER0 IN MODE3! COUNT=01F4H', 0AH, 0DH
       DB '8253A TIMER1 IN MODE2! COUNT=0AH',  0AH,  0DH , '$'
DATA   ENDS

CODE       SEGMENT
MAIN       PROC   FAR
ASSUME     CS: CODE ,  DS: DATA
START:     PUSH     DS
           MOV      AX, 0
           PUSH     AX
           MOV      AX , DATA
           MOV      DS , AX
           CLI
           MOV      DX , TIM_CTL
           MOV      AL ,  _____          ; 设置 0 通道方式字
           OUT      DX , AL
           MOV      DX , TIMER0
           MOV      AL ,  _____          ; 设置计数初值
           OUT      DX , AL
           MOV      AL ,  _____
           OUT      DX , AL
           MOV      DX , TIM_CTL
           MOV      AL ,  _____          ; 设置 1 通道方式字
           OUT      DX , AL
```

```
        MOV     DX ， TIMER1
        MOV     AL ， 0AH
        OUT      DX ， AL
        STI
        MOV     DX ， OFFSET MESS
        MOV     AH ， 09
        INT      21H
        RET
MAIN    ENDP
CODE    ENDS
END     START
```

五、思考题

1．将计数器 1 改为方式 1（方式 4，方式 5），用外接脉冲控制计数，输出 OUT1 接至发光二极管，观察其点亮情况。要求点亮时间为 1S，问如何接线？如何编程序？

2．组成一定时系统，输出如图所示的连续波形。



3．简单模拟一个计件流水线：当流水线通过 5 个部件后（可将开关产生的脉冲信号接入 CLK 信号来模拟有部件通过），则点亮一次发光二极管，采用方式 1 进行计数。编写实验初始化程序并设计硬件连接图。

实验十六　可编程计数器/定时器 8253（二）

一、实验目的和内容

1．掌握 8253 定时器/计数器芯片的工作原理

2．完成 8253 定时器 / 计数器的编程实验

注：做试验前将前一试验的连线撤除

二、实验环境

1）硬件：PC 微机、SXL-100 B+型微机接口实验开发系统、示波器

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验电路及说明

实验电路如图 7 所示

地址为 IOH

8253 的三个计数器全部开放，J53 上有 OUT0、GATE0CLK0、OUT1、GATE1、CLK1、OUT2、GATE2、CLEK2。它们可与任何 I/O 相连。在我公司提供的实验箱上的 D 部分有各种频率时钟信号。下面提供一例，供参考，学生可根据要求自行调换频率。

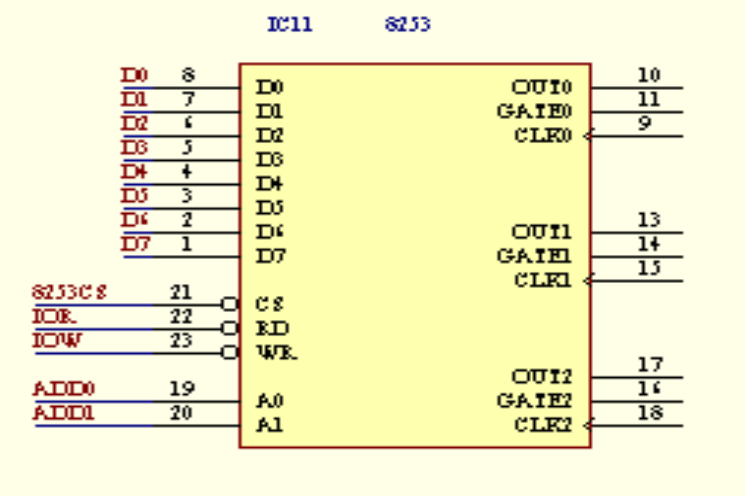


图 7

8253的控制字

向控制寄存器写入方式控制字，以选择计数通道、工作方式、计数值读写方式等。8253的控制字格式如下表所示：

D7 D6	D5 D4	D3D2D1（方式）	D0
00 通道	00 当前计数值	000 方式 0	清 0 时为二进制计数
01 通道	01 读/写低 8 位、	001 方式 1	置 1 时为十进制计数
10 通道	高 8 位清 0	010 方式 2	
11 无效	10 读/写高 8 位、	011 方式 3	
	低 8 位清 0	100 方式 4	
	11 读/写双字节	101 方式 5	

本实验中 T0 控制字为 3EH，即为二进制计数，工作方式 3，选择 00 通道，计数值先读低 8 位、再读高 8 位。T1 控制字为 5EH，与 T0 控制字的区别仅是 D7D6 位选择了 01 通道。

四、硬件实验步骤

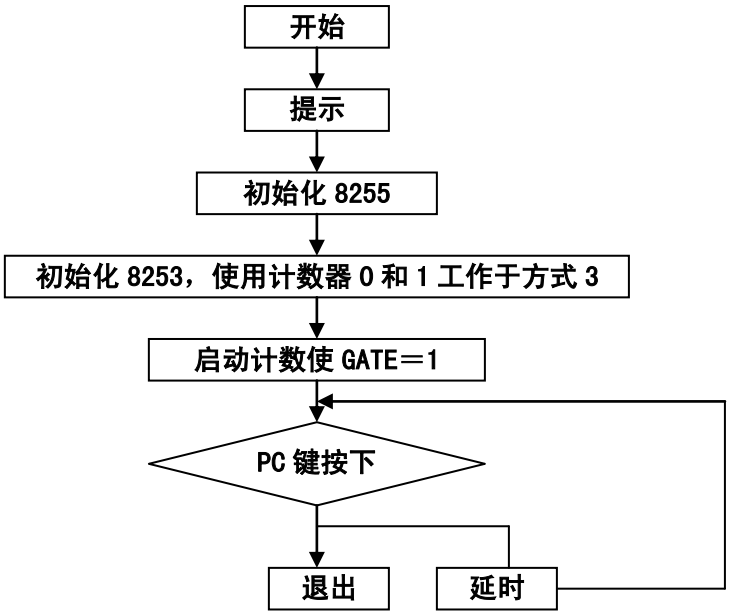
本实验在板上由 G 与 H 部分组成，由短路块结合跳线方式进行实验。

1 用跳线把 G 区 J7 的 31.25KHZ 端与 H 区 J53 的 CLK0 相连

2 用跳线把 J 区 J101(GATE)和 H 区 GATE0、GATE1 端相连

3 H 区 OUT0 端和 H 区 CLK1 端相连，保持 8255 的各短路块不动

五、程序框图



六、程序代码　82533\82533.asm

```
.MODEL SMALL
.DATA
    IOADD DW 0060H                ;输出口地址
    I8253ADD0 DW 0020H            ;8253T0 地址
    I8253ADD1 DW 0021H            ;8253T1 地址
    I8253TYPE DW 0023H            ;8253 控制字位址
    FLAG DB 00H                    ;读数标志
       DIS0       DB 'PRESS ANY OF THE KEYS TO QUIT!$';提示信息
    I8255TYPE DW 0003H             ;8255 控制字的位址
    I8255addA DW 0000H             ;8255 的 A 口地址
    I8255addB DW 0001H             ;8255 的 B 口地址
       DLYC1          EQU      3
    DDBB       DB      4 DUP(0)
    COUNT1     DB 00H              ;计脉冲次数
     DISC      DB   0c0H,0f9H,0a4H,0b0H,99H,92H,82H,0f8H,80H,98H
    m_A         db   16             ;A 口资料位
    m_B    db   4                  ;B 口扫描次数
    DIS1 DB 'PCI ADDRESS 0 $';
       DIS2 DB 'PCI ADDRESS 1 $'
       DIS3 DB 'PCI MEMORY ADDRESS $'
                              DIS4 DB 'PCI INTERRUPT LINE $'
       DIS5 DB 'BIOS NOT SUPPER!$'
```

```
    DIS6 DB 'READ PCI BOARD FAIL!$'
      PCI_CS_VENDOR_ID EQU 0
      PCI_CS_DEVICE_ID EQU 2
      PCI_CS_COMMAND   EQU 4
      PCI_CS_STATUS    EQU 6
      PCI_CS_REVISION  EQU 8
      PCI_CS_CLASS_CODE  EQU 9

      PCI_CS_BASE_ADDRESS_0 EQU 10H
      PCI_CS_BASE_ADDRESS_1 EQU 14H
      PCI_CS_BASE_ADDRESS_2 EQU 18H
      PCI_CS_BASE_ADDRESS_3 EQU 1CH
      PCI_CS_BASE_ADDRESS_4 EQU 20H
      PCI_CS_BASE_ADDRESS_5 EQU 24H
      PCI_CS_EXPANSION EQU 30H
      PCI_CS_INTERRUPT_LINE EQU 3CH
       PCI_CS_INTERRUPT_PIN EQU 3DH
      PCI_CS_MIN_GNT EQU 3EH
      PCI_CS_MAX_LAT EQU 3FH
ADDRESS_IO_0 DW ?
ADDRESS_IO_1 DW ?
.stack 100H
 .CODE
 .STARTUP
MOV      AX,DATA
MOV      DS,AX
MOV      AX,DATA
MOV      ES,AX
MOV      AX,SSREG
MOV AH,0B1H                    ;看 PCI BIOS 是否存在
MOV AL,1H
INT 1AH
.IF AH!=00
JMP m_EXIT
      .ENDIF
      MOV AH,0B1H              ;查找 PCI 卡的位置
      MOV AL,02H
      MOV CX,8376H
      MOV DX,10EBH
      MOV SI,0
      INT 1AH
JNC AA
      JMP m_EXIT
```

```
AA:       MOV AH,0B1H                ;读取配置寄存器的板卡基地址
          MOV AL,09H
    MOV DI,PCI_CS_BASE_ADDRESS_1
          INT 1AH
    .IF AH!=0
          JMP m_EXIT
          .ENDIF
    AND CX,0FFFCH
    MOV AX,CX
    MOV ADDRESS_IO_0,AX
          MOV AH,0B1H                ;读取配置寄存器的用户基地址
          MOV AL,09H
    MOV DI,PCI_CS_BASE_ADDRESS_3
          INT 1AH
    .IF AH!=0
          JMP m_EXIT
          .ENDIF
    AND CX,0FFFCH
    MOV AX,CX
    MOV ADDRESS_IO_1,AX
    ADD I8255TYPE,AX
    ADD I8255addA,AX
    ADD I8255addB,AX
    ADD I8253TYPE,AX
    ADD I8253ADD0,AX
    ADD I8253ADD1,AX
           MOV AL,3
           MOV AH,0
           INT 10H
           MOV DX,OFFSET DIS0          ;显示提示
           MOV AH,9
           INT 21H
           MOV DX,I8255TYPE            ;初始 8255 的 A，B 口输出。
    MOV AL, 80H
           OUT DX,AL
    MOV DX,I8255addA
    MOV AL,0FFH
    OUT DX,AL
    MOV DX,I8255addB                   ;关控制埠
    MOV AL,0FFH
    OUT DX,AL
    MOV DX,I8253TYPE                    ;写 8253T0 控制字
    MOV AL,3EH
```

```
    OUT DX,AL
    MOV DX,I8253ADD0                    ;写 T0 的计数值
    MOV AL,0FFH
    OUT DX,AL
    MOV AL,0FH
    OUT DX,AL
    MOV DX,I8253TYPE                    ;写 8253T1 控制字
    MOV AL,5EH
    OUT DX,AL
    MOV DX,I8253ADD1                    ;写 T1 计数值
    MOV AL,08H
    OUT DX,AL
CYCLE: MOV DX,I8253ADD1   ; CONG 8253 T1 DU QU JI SHU ZHI DAO CX ZHONG
    IN AL,DX
    MOV CL,AL
    CMP CL,2                            ;判断计数值是否为 2
    JNZ CYCLE1
    CMP FLAG,0                          ;判断是否已计过 2
    JNZ CCP
    MOV FLAG,1
    INC COUNT1
    CMP COUNT1,2
    JNZ CCP
    MOV COUNT1,0
    CALL PLUS1
    JMP CCP
CYCLE1:    MOV FLAG,0
CCP:   CALL   DISPLAY
    CALL   PCKEY
    JMP CYCLE
DISPLAY PROC     NEAR                   ;显示子程序
          MOV      SI,OFFSET DDBB
           MOV BH,0                     ;取个位
    MOV DX,I8255addA
    MOV BL,[SI]
    MOV DI,BX
    MOV AL,[DI+DISC]
    OUT DX,AL
    MOV DX,I8255addB
    MOV AL,0FeH                         ; 显示个位
    OUT DX,AL
          CALL     TIME
        MOV AL,0FFH
```

```
        OUT DX,AL
        INC SI
        MOV DX,I8255addA
        MOV BL,[SI]
        MOV DI,BX
        MOV AL,[DI+DISC]
        OUT DX,AL
        MOV DX,I8255addB
        MOV AL,0FdH                    ;显示十位
        OUT DX,AL
         CALL TIME
        MOV AL,0FFH
        OUT DX,AL
        INC SI
        MOV DX,I8255addA
        MOV BL,[SI]                    ;取百位数
        MOV DI,BX
        MOV AL,[DI+DISC]               ;取显示段码
        OUT DX,AL
        INC DX
        MOV AL,0FbH                     ;显示百位
        OUT DX,AL
         CALL TIME
        MOV AL,0FFH
        OUT DX,AL
        INC SI
        MOV DX,I8255addA
        MOV BL,[SI]                    ;取千位数
        MOV DI,BX
        MOV AL,[DI+DISC]                ;取显示段码
        OUT DX,AL
        MOV DX,I8255addB
        MOV AL,0F7H                     ;显示千位
        OUT DX,AL
         CALL TIME
        MOV AL,0FFH
        OUT DX,AL
        RET
DISPLAY     ENDP
PCKEY PROC NEAR                         ;扫描键盘子程序
    MOV AH,6
           MOV DL,0FFH
           INT 21H
```

```
            JE P1
            MOV DX,I8255addA
    MOV AL,0FFH
    OUT DX,AL
            MOV AX,4C00H
            INT 21H
P1:     RET
PCKEY  ENDP
TIME PROC NEAR                          ;延时子程序(2 秒~5 秒)
    MOV AX,000FH
T2:     DEC AX
    MOV BX,0FFH
T1:     DEC BX
            JNZ T1
            .IF AX!=0
            JMP T2
            .ENDIF
            RET
TIME   ENDP
PLUS1 PROC NEAR                         ;四位十进制加 1 子程序
    PUSH CX
D1:MOV SI,OFFSET DDBB                    ; 显示
    MOV CX,0004H
D0:MOV AL,[SI]
    INC AL
    CMP AL,0AH
    JNE DD1
    MOV AL,0
    MOV [SI],AL
    INC SI
    LOOP   D0
DD1:MOV    [SI],AL
    POP CX
    RET
PLUS1 ENDP
m_EXIT:    MOV AX,4C00H
    INT 21H
    END
```

实验十七　PC 机发声及音乐程序设计

一、实验目的

1. 掌握 PC 机的发声原理
2. 掌握演奏程序设计方法

二、实验环境

1）硬件：PC 微机

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

三、实验内容

1） 软件定时控制发声频率

假设计算机的主频为 1.6GHz/S（千兆赫/秒），执行一条 LOOP 指令约需 2 个总线周期，若使用软件定时控制发声频率，编写程序，使计算机扬声器发出固定频率为 200HZ/S 的声音，发声持续 5 秒。

2） 硬件(8253)定时控制发声频率

已知 8253 的时钟频率 CLK=1.19MHZ/S（兆赫/秒），使用 8253 的定时器/计数器 2 定时控制发声频率，编写程序，使计算机的扬声器依次按 200、400、300、450、350、250、300、200、150、300、200 的频率各发声 100ms。假设计算机主频为 1.6G,执行一条 loop 指令用两个总线周期。

3） 音乐程序设计

编写程序用计算机扬声器演奏歌曲亚洲雄风

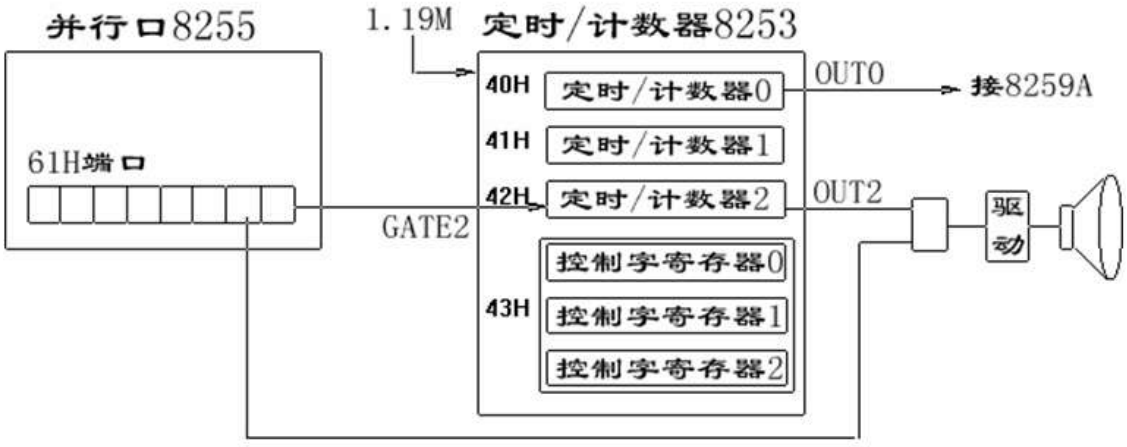
四、实验要求

1） 实验前准备好汇编语言源程序。

2） 实验要求在 PC 机上进行。

五、实验电路及说明

实验电路如图所示。



PC 机上装有一个小扬声器,连接到 8255 的 61H 输出口和 8253 定时/计数器 2 的输出端 OUT2，如上图所示。可以看出，扬声器的振荡发声是由 8255 输出寄存器（61H）中的最低两位来控制的。一种情况是使 61H 的第 0 位为 0，8254 定时器不工作，OUT2 输出恒为 1，扬声器前的与门打开，让 61H 的第 1 位 0/1 交替变化，使扬声器交替地接通与断开，从而推动扬声器的纸盆振

动发声。另一种情况是让第 1 位为 1，扬声器前的与门打开，当输出寄存器的第 0 位为 1 时，控制 8254 定时器工作，OUT2 产生 0/1 交替变化，驱动扬声器发声。

六、编程提示

编写乐曲程序可分为四个步骤：

1）为演奏的乐曲定义一个频率表和一个节拍时间表,在程序数据段中分别由freq和time所指向的数组;

2）分别将两个表的偏移地址放入SI和BP

```
mov si,offset freq      ;si指向发声频率表
mov bp,offset time      ;bp指向节拍时间表;
```

3）从表中取出音符的频率送DI,取出音符的持续时间(实际上是10ms的倍数)送BX

```
mov di,[si]             ;取此次发声频率到di
mov bx,ds:[bp]          ;取此次发声持续时间(10ms的倍数)到bx
```

4）调用sounding子程序发声。

所求程序如下:

```
stack segment stack
   dw 100 dup(?)
stack ends
data  segment
  disp db 0ah,0dh,"亚洲雄风:$"
  freq dw 262,196,262,294,330,262,262,392,262,330,294,262,294
       dw 262,196,262,294,330,262,262,440,262,330,294,262,294,392
       dw 262,196,262,294,330,262,262,392,262,330,294,262,294
       dw 262,196,262,294,330,262,220,262,440,392,392,392
       dw 392,262,392,392,349,349,330,262,392,262,294,330
       dw 392,523,392,392,440,440,392,349,330,262,294,294,262
       dw 392,392,392,262,440,392,392,349,392,330,262,392,262,294
       dw 392,392,392,262,392,440,440,440,392,349,330,262,294,294,262
       dw 0
  time dw 50,25,25,18,7,25,25,25,25,25,18,7,100
       dw 50,25,25,18,7,50,25,25,25,25,18,7,25,75
       dw 50,25,25,18,7,50,25,25,25,25,18,7,100
       dw 50,25,25,18,7,75,25,25,13,25,12,100
       dw 50,25,25,13,12,75,25,25,25,18,7,100
       dw 50,25,25,13,12,50,13,12,25,25,25,13,112
       dw 25,25,13,37,13,25,62,25,25,25,25,25,13,62
       dw 25,25,13,37,13,25,12,25,13,12,25,25,25,13,112
data  ends
code segment
     assume cs:code,ds:data
start: mov ax,data
       mov ds,ax
       mov dx,offset disp
```

```
       mov ah,09
       int 21h
       mov si,offset freq  ;si指向第一个发声频率数据
       mov bp,offset time  ;bp指向第一个频率发声长度(10ms的倍数)
       rept1: mov di,[si]  ;取此次发声频率到di
       cmp di,0            ;看是否应结束
       je ending
       mov bx,ds:[bp]      ;取此次发声长度(10ms的倍数)到bx
       call sounding       ;调用发声子程序
       add si,2            ;si指向下一个发声频率数据
       add bp,2            ;bp指向下一个频率发声长度
       jmp rept1
ending:mov ah,4ch
       int 21h
sounding proc near
      push ax
      push bx
      push cx
      push dx
      push di
      mov al,0b6h     ;定时器2按方式3工作,二进制计数,16位初值
      out 43h,al      ;控制字送定时器2的控制字寄存器
      mov dx,12h      ;1.19M=1234dcH, 做被除数
      mov ax,34dch
      div di          ;此次发声频率做除数,求定时器2的计数初值
      out 42h,al      ;计数初值分两次送计定时器2的数初值寄存器
      mov al,ah
      out 42h,al
      in al,61h       ;读入61H端口内容到al
      mov ah,al       ;将该内容先送ah保存
      or al,3         ;将al最低两位置1,其他位不变
      out 61h,al      ;送回61H端口,OUT2按此次频率交替送出0/1
delay: mov cx,1000    ;软件延时(bx)*10ms
dl10:  mov di,cx
       mov cx,2000
dl001: loop dl001
       mov cx,di
       loop dl10
       dec bx
       jnz delay
       mov al,ah     ;将开始存在ah中的61H端口原内容送回
       out 61h,al
       pop di
       pop dx
       pop cx
       pop bx
       pop ax
       ret
sounding endp
code ends
     end start
```

实验十八　主控 DMA8237 实验

一、实验目的和内容

1. 掌握 8237 的工作原理。
2. 利用 8237 实现 DMA 传送。

注：做试验前将前一试验的连线撤除

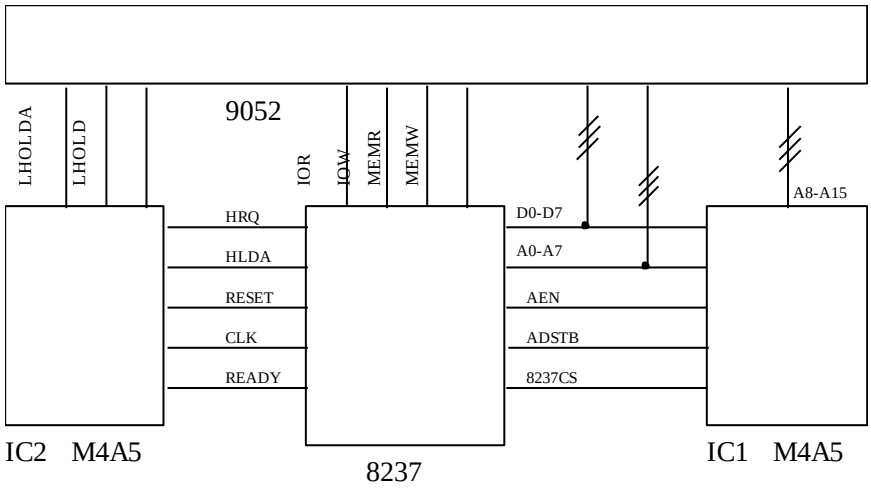
二、实验环境

1）硬件：PC 微机、SXL-100 B+型微机接口实验开发系统

2）软件：DOS 系统、EDIT.EXE、MASM.EXE、LINK.EXE、DEBUG.EXE

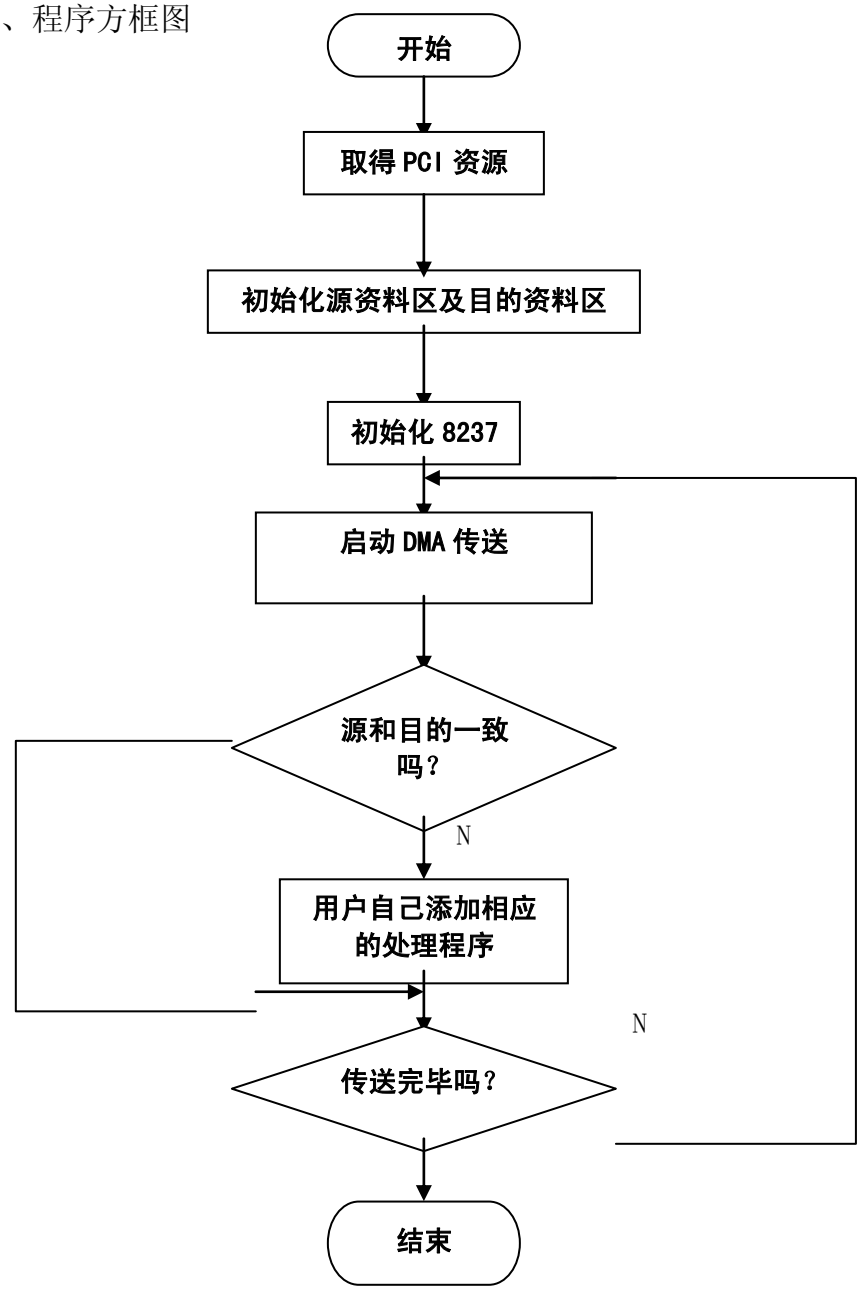
三、实验电路及说明

实验电路见图。8237 的地址范围为 70~7F，其 HRQ 接 9052 的 LHOLD，HLDA 接 9052 的 LHOLDA，CLK 经 LCLK（8MHz）2 分频得到（8237A-5 的频率最高为 5MHz），READY 接高电平，IC1（M4A5）完成 8237 的地址译码，及 8237 进行 DMA 传送时的高 8 位地址的锁存，IC2（M4A5）完成 CLK 的分频，RESET、READY、HRQ、HLDA 的连接，并完成 RAM 6116 的片选译码



8237 命令字格式，通道地址等。

四、程序方框图



五、程序代码

```
.MODEL SMALL
.DATA
      DIS0 DB 'PCI CONFIG INFORMATION!$';提示信息
      DIS1 DB 'PCI ADDRESS 0 $';
      DIS2 DB 'PCI ADDRESS 1 $'
      DIS3 DB 'PCI MEMORY ADDRESS $'
      DIS4 DB 'PCI INTERRUPT LINE $'
      DIS5 DB 'BIOS NOT SUPPER!$'
      DIS6 DB 'READ PCI BOARD FAIL!$'
      mode9052 db ?
      iobase0 dw ?
```

```
      iobase1 dw ?
      membase dw ?
      str  db 128 DUP(0AAH)
      deststr db 128 DUP(55H)
      ch0_a dw 70h
      ch1_a dw 72h
      ch1_c dw 73h
      statue dw 78h ;for read
      cmd dw 78h    ;for write
      req dw 79h
      maskr dw 7ah
      mode dw 7bh
      clear_f dw 7ch
      reset dw 7dh  ;for write
      temp dw 7dh   ;for read
      masks dw 7fh
;-------PCI Configuration Space Registers-----------------------
        PCI_CS_VENDOR_ID EQU 0
        PCI_CS_DEVICE_ID EQU 2
        PCI_CS_COMMAND   EQU 4
        PCI_CS_STATUS    EQU 6
        PCI_CS_REVISION  EQU 8
        PCI_CS_CLASS_CODE  EQU 9
        PCI_CS_CACHE_LINE_SIZE EQU 0CH
        PCI_CS_MASTER_LATENCY EQU  0DH
        PCI_CS_HEADER_TYPE EQU 0EH
        PCI_CS_BIST      EQU 0FH
        PCI_CS_BASE_ADDRESS_0 EQU 10H
        PCI_CS_BASE_ADDRESS_1 EQU 14H
        PCI_CS_BASE_ADDRESS_2 EQU 18H
        PCI_CS_BASE_ADDRESS_3 EQU 1CH
        PCI_CS_BASE_ADDRESS_4 EQU 20H
        PCI_CS_BASE_ADDRESS_5 EQU 24H
        PCI_CS_EXPANSION EQU 30H
        PCI_CS_INTERRUPT_LINE EQU 3CH
         PCI_CS_INTERRUPT_PIN EQU 3DH
        PCI_CS_MIN_GNT EQU 3EH
        PCI_CS_MAX_LAT EQU 3FH
;--------------------END-------------------------
    ADDRESS_IO_0 DW ?
    ADDRESS_IO_1 DW ?
    ADDRESS_MEM_L  DW ?
    ADDRESS_MEM_H  DW ?
    INTERRUPT_LINE DB ?
.STACK 100H
.CODE
.STARTUP
         MOV AL,3
     MOV AH,0
     INT 10H
     MOV DX,OFFSET DIS0      ;显示提示
```

```
        MOV AH,9
       INT 21H
           MOV AH,2H
           MOV BH,0
           MOV DH,1
           MOV DL,0
           INT 10H
           MOV AH,0B1H
           MOV AL,1H
           INT 1AH
           .IF AH!=00
           MOV DX,OFFSET DIS5
           JMP m_EXIT
           .ENDIF
           MOV AH,0B1H
           MOV AL,02H
           MOV CX,8376H
           MOV DX,10EBH
           MOV SI,0
           INT 1AH
           JNC AA
           MOV DX,OFFSET DIS1
           MOV AH,9
           INT 21H
           JMP m_EXIT
AA:        MOV AH,0B1H
           MOV AL,09H
       MOV DI,PCI_CS_BASE_ADDRESS_1
           INT 1AH
           .IF AH!=0
            MOV DX,OFFSET DIS1
           MOV AH,9
           INT 21H
           JMP m_EXIT
           .ENDIF
       AND CX,0FFFCH
       MOV AX,CX
       MOV ADDRESS_IO_0,AX
       mov iobase0,ax
           MOV AH,0B1H
           MOV AL,09H
           MOV DI,PCI_CS_BASE_ADDRESS_3
           INT 1AH
           .IF AH!=0
            MOV DX,OFFSET DIS1
           MOV AH,9
           INT 21H
           JMP m_EXIT
           .ENDIF
       AND CX,0FFFCH
       MOV AX,CX
```

```
MOV ADDRESS_IO_1,AX
mov iobase1,ax
    MOV AH,0B1H
    MOV AL,09H
    MOV DI,PCI_CS_BASE_ADDRESS_2
    INT 1AH
    .IF AH!=0
     MOV DX,OFFSET DIS1
    MOV AH,9
    INT 21H
    JMP m_EXIT
    .ENDIF
AND CX,0FFFCH
MOV AX,CX
MOV ADDRESS_MEM_L,AX
    MOV AH,0B1H
    MOV AL,09H
    MOV DI,PCI_CS_BASE_ADDRESS_2
ADD DI,2
    INT 1AH
    .IF AH!=0
     MOV DX,OFFSET DIS1
    MOV AH,9
    INT 21H
    JMP m_EXIT
    .ENDIF
MOV AX,CX
MOV ADDRESS_MEM_H,AX        ; 读 PCI 上的内存地址
mov ax,ADDRESS_MEM_L
mov dx,ADDRESS_MEM_H
mov cx,10h
div cx
mov membase,ax
    MOV AH,0B1H        ; 读取配置寄存器的中断号
    MOV AL,09H
    MOV DI,PCI_CS_INTERRUPT_LINE
    INT 1AH
    .IF AH!=0
     MOV DX,OFFSET DIS1
    MOV AH,9
    INT 21H
    JMP m_EXIT
    .ENDIF
MOV AX,CX
MOV INTERRUPT_LINE,AL     ; 显示“PCI ADDRESS 0”
     MOV DX,OFFSET DIS1
 MOV AH,9
     INT 21H
     MOV AX, ADDRESS_IO_0           ; 显示 PCI 卡上的基地址
     CALL DISPLAY
     CALL HDIS
```

```
    MOV AH,2H
    MOV BH,0
    MOV DL,0
    MOV DH,2
    INT 10H
    MOV DX,OFFSET DIS2
MOV AH,9
    INT 21H
    MOV AX, ADDRESS_IO_1          ; 显示大板子上的基地址
    CALL DISPLAY
    CALL HDIS
    MOV AH,2H
    MOV BH,0
    MOV DL,0
    MOV DH,3
    INT 10H
    MOV DX,OFFSET DIS3
MOV AH,9
    INT 21H
MOV AX,ADDRESS_MEM_H
CALL DISPLAY
    MOV AX,ADDRESS_MEM_L           ; 显示 PCI 内存地址
    CALL DISPLAY
    CALL HDIS
    MOV AH,2H
    MOV BH,0
    MOV DL,0
    MOV DH,4
    INT 10H
    MOV DX,OFFSET DIS4
MOV AH,9
    INT 21H
    MOV AH,0
    MOV AL,INTERRUPT_LINE          ; 显示 PCI 卡中断号
    CALL DISPLAY
CALL HDIS
mov ah,0
mov al,2ah
add ax,iobase0
mov dx,ax
in al,dx
mov mode9052,al          ;存 9052 模式
mov al,0
out dx,al               ;置 8 位模式
mov ax,51h
add ax,iobase0
mov dx,ax
mov al,43h
out dx,al               ;禁止 74LS245
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mov di,0                ; 向偏移量为 0 的附加段写入 STR 处的 128 个字符
```

```
     mov es,membase
     mov cx,128
     mov si,offset str
lp1:     mov al,[si]
     mov es:[di],al
     inc di
     inc si
     loop lp1
     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     mov di,128          ; 向偏移量为 128 的附加段写入 DESTSRT 处的 128 个字符
     mov es,membase
     mov cx,128
     mov si,offset deststr
lp2:     mov al,[si]
     mov es:[di],al
     inc di
     inc si
     loop lp2
     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     mov ax,iobase1           ; 获取大板上 8237 的各项地址
     add ch0_a,ax
     add ch1_a,ax
     add ch1_c,ax
     add cmd,ax
     add statue,ax
     add req,ax
     add mode,ax
     add reset,ax
     add temp,ax
     add clear_f,ax
     add masks,ax
     add maskr,ax
     mov dx,temp
     in al,dx              ; 从 8237 读入
     mov cx,120
     mov bl,0
     mov bh,80h
lp3:     mov dx,masks      ; 初始化 8237 并启动 DMA 传送
     mov al,0ch
     out dx,al
     mov dx,ch0_a
     mov al,bl
     out dx,al
     mov al,0
     out dx,al
     mov dx,ch1_a
     mov al,bh
     out dx,al
     mov al,0
     out dx,al
     mov dx,ch1_c
```

```
        mov al,0
        out dx,al
        out dx,al
        mov dx,req
        mov al,4
        out dx,al
        call delay
        mov al,bl
        mov ah,0
        mov si,ax
        mov di,offset deststr
        add ax,di
        mov di,ax
        mov dx,temp        ; 将 DMA 传送数据送 AL
        in al,dx
        mov [di],al
        mov ah,es:[si]     ; 比较源数据和传送数据是否一致；若不一致则微处理（由用
户自己加程序）
        .IF AL!=AH
        .ENDIF
        inc bl
        inc bh
        loop lp3
        mov di,128
        mov es,membase
        mov cx,128
        mov si,offset deststr
    lp4:     mov al,[si]    ; 将 DESTSTR 处的 128 个数据送到偏移量为 128 的地址
        mov es:[di],al
        inc di
        inc si
        loop lp4
        mov ax,51h
        add ax,iobase0
        mov dx,ax
        mov al,42h
        out dx,al         ;使能 74LS245
        mov ax,2ah
        add ax,iobase0
        mov dx,ax
        mov al,mode9052     ;restore 9052's mode
        out dx,al
            JMP m_EXIT
    DISPLAY PROC NEAR   ; 显示子程序
          PUSH CX
         PUSH BX
         MOV CL,4
         MOV CH,4
    DISPH1:ROL AX,CL
         PUSH AX
         AND AL,0FH
```

```
      ADD AL,30H     ; 获取 AL 中数据的 ASCⅡ码
      CMP AL,'9'     ; 与 9 的 ASCⅡ码相减比较
      JBE DISPH2     ; 若小于等于 9 则转到 DISPH2
      ADD AL,7       ; 显示 AL 中的字符
DISPH2: MOV AH,2
      MOV DL,AL
      INT 21H
      POP AX
      DEC CH
      JNZ DISPH1
      POP CX
      POP BX
      RET
DISPLAY ENDP
HDIS   PROC NEAR      ; 显示 H 字符子程序
 MOV DL, 'H'
      MOV AH,2
      INT 21H
       RET
HDIS  ENDP
delay  proc near
       push ax
       push bx
       MOV AL,0FH
T2:   DEC AL
      MOV BX,0FFFFH
T1:   DEC BX
      JNZ T1
      .IF AL!=0
      JMP T2
      .ENDIF
       pop bx
       pop ax
       ret
delay  endp
m_EXIT:
        MOV AX,4C00H
    INT 21H
        END
```