# 实验一 汇编语言编程实验

## 一、实验目的

1. 掌握汇编语言的编程方法
2. 掌握DOS功能调用的使用方法
3. 掌握汇编语言程序的调试运行过程

## 二、实验内容

1. 将指定数据区的字符串数据以ASCII码形式显示在屏幕上，并通过DOS功能调用完成必要提示信息的显示。
2. 在屏幕上显示自己的学号姓名信息。
3. 循环从键盘读入字符并回显在屏幕上，然后显示出对应字符的ASCII码，直到输入”Q”或“q”时结束。
4. 自主设计输入显示信息，完成编程与调试，演示实验结果。

## 三、实验步骤

1. 运行QTHPCI软件，根据实验内容，参考程序流程图编写程序。
2. 选择“项目”菜单中的“编译”或“编译连接”对实验程序进行编译连接。
3. 选择“调试”菜单中的“进行调试”，进入Debug调试，观察调试过程中传输指令执行后各寄存器及数据区的内容。按F9连续运行。

## 四、实验过程

1. 完成此次实验，需要对汇编语言中的系统功能调用有一些了解，可能使用到的的系统功能调用如下所示。注意使用如下系统功能调用时，需要与INT 21H 一同使用。INT是interupt中断的缩写，是DOS的中断调用命令。

* 本次实验的任务二，显示学号姓名信息，就需要用到 int 21H 中断的 09H 号功能。
* 将 DX 寄存器设置为待显示的字符串偏移地址，将 AH 寄存器的内容设置为 09 调用 int 21H 中断，就可以把待显示字符串显示到屏幕上。



*图****1.****部分系统功能调用参考表*

1. 在计算机中，所有的数据均以二进制01存储，其中字符则存放其对应的ASCII码值，读取数据时，寄存器中存放的值均为ASCII码值。实验要求输出其ASCII码，而被输出的ASCII码又是以ASCII码表示的。简而言之，需要做两次关于字符与ASCII码的映射。

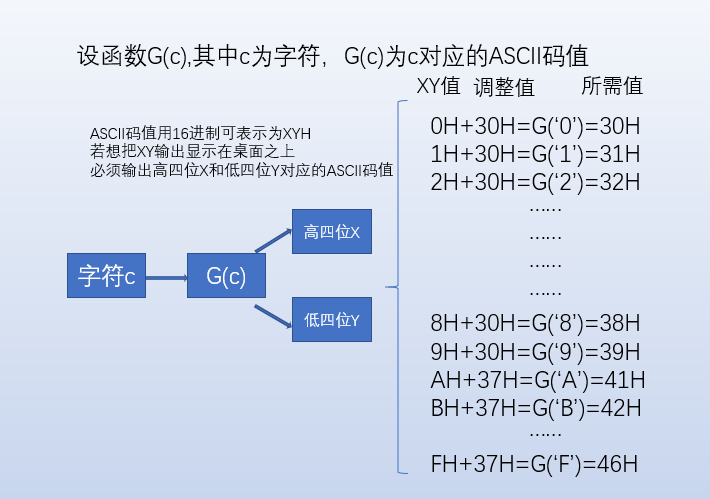
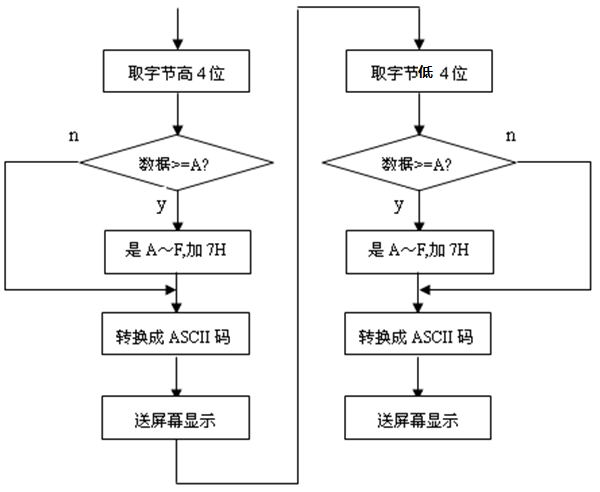


图2.所表示数据与ASCII码的映射关系

1. 当X或Y值为0~9H时，需要加上调整值30H；当X或Y值为A~FH时，需要加上调整值37H。 据此，则可将其进行转换为ASCII码值。



*图****3.****字符转换为****ASCII****码流程图*

1. 实验一的核心内容是进行 ASCII 码的转换与显示，因此，我设计了 ascii proc 子程序。调用子程序，便可将 AX 寄存器中存放的 ASCII 码值显示到屏幕上

* ascii proc   
   push ax   
   push dx   
   push cx ;保护现场   
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
  ; 显示第一位数字   
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
   PUSH AX   
   AND AL,0F0H   
   mov cl,4   
   SHR AL,cl   
   ADD AL,30H   
   MOV DL,AL   
   MOV AH,02H   
   INT 21H   
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
  ; 显示第二位数字   
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
   POP AX   
   AND AL,0FH   
   CMP AL,09H   
   JNA ascend   
   ADD AL,07H ;字母特殊处理   
  ascend:   
   add al,30h   
   mov dl,al   
   mov ah,02h   
   int 21H   
   pop cx   
   pop dx   
   pop ax   
   ret   
  ascii endp

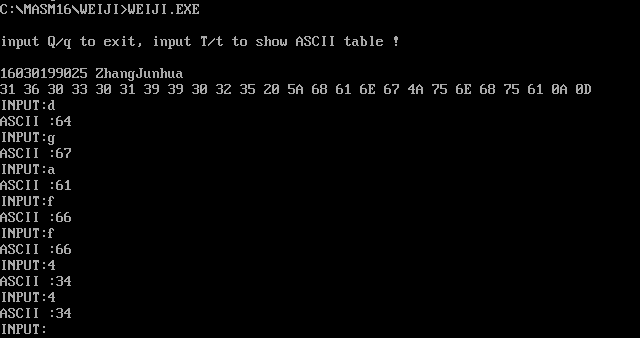
1. 要实现循环读入并按 Q 键中断的功能，需要设计一个循环，并在循环中对结束条件进行检测。在这里，我使用 JMP 命令进行无条件跳转。并在循环中将读入的字符与 ‘q’ ‘Q’ 进行比较，若相等则跳转到退出指令。

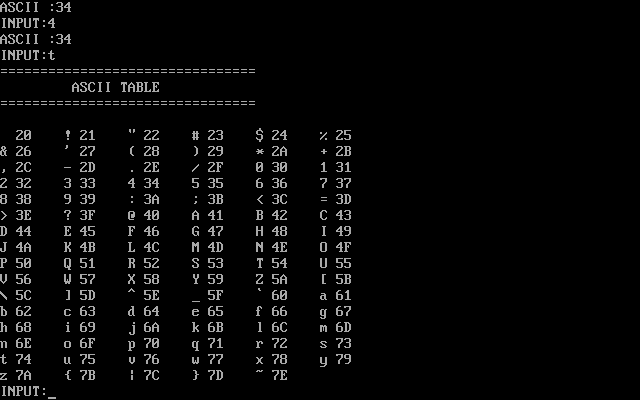
* MAIN:   
   MOV DX,OFFSET string2   
   MOV AH,09H   
   INT 21H   
   MOV AH,01H   
   INT 21H   
   push ax   
   CMP AL,'Q'   
   JZ EXIT   
   CMP AL,'q'   
   JZ EXIT   
   cmp al,'T'   
   jz ASCTB   
   cmp al,'t'   
   jz ASCTB   
     
   MOV DX,OFFSET RESULT   
   MOV AH,09H   
   INT 21H   
   pop ax   
   call ascii   
   jmp MAIN   
  EXIT:   
   MOV AX,4C00H   
   INT 21H

1. 在基础实验的要求上，我又设计实现了 ASCII 码表的输出，当检测到键盘输入 t 时，调用 ascii 码表输出子程序，通过循环操作对 ASCII 码表进行输出

* ASCTB:   
   push ax   
   push cx   
   push dx   
     
   mov bl, 20H   
   call newlne   
   MOV DX,OFFSET ASCTT   
   MOV AH,09H   
   INT 21H   
   TABLE:   
     
   ;;;;;;;;;;;;   
   ; 换行   
   ;;;;;;;;;;;;   
   mov ah,02H   
   mov dl,0AH   
   int 21H   
   mov dl,0DH   
   int 21H   
   mov cx,6   
   row:   
   ;;;;;;;;;;;;;;;;;;;   
   ; 显示字符   
   ;;;;;;;;;;;;;;;;;   
   mov dl,bl   
   mov ah,02H   
   int 21H   
   mov dl,20H   
   int 21H   
   ;;;;;;;;;;;;;;;   
   ; 显示ascii   
   ;;;;;;;;;;;;;   
   mov al,bl   
   call ascii   
   mov dl,09   
   int 21H   
     
   inc bl   
   cmp bl,7FH   
   jnb TABLEEND   
     
     
   loop row   
   jmp TABLE   
     
   TABLEEND:   
   pop dx   
   pop cx   
   pop ax   
  jmp MAIN

## 五、实验结果





### 完整源代码：

DATA SEGMENT   
Sno DB 'this is a massage',0AH,0DH,'$'   
string1 DB 0AH,0DH,'input Q/q to exit, input T/t to show ASCII table !',0AH,0DH,0AH,0DH,'$'   
string2 DB 0AH,0DH,'INPUT:$'   
RESULT DB 0AH,0DH,'ASCII :$'   
ASCTT DB '================================',0AH,0DH   
 DB ' ASCII TABLE ',0AH,0DH   
 DB '================================',0AH,0DH,'$'   
DATA ENDS   
   
STACK SEGMENT   
DB 100 DUP(0)   
STACK ENDS   
   
CODE SEGMENT   
ASSUME CS:CODE,DS:DATA,SS:STACK   
   
START:   
 MOV AX,STACK   
 MOV SS,AX   
 MOV AX,DATA   
 MOV DS,AX   
 MOV DX,OFFSET string1   
 MOV AH,09H   
 INT 21H   
 MOV DX,OFFSET Sno   
 INT 21H   
 MOV BX,0   
   
l1:   
 MOV DL,[BX]   
 CMP DL,'$'   
 JZ MAIN   
 MOV AL,DL   
 call ascii   
 MOV DL,' '   
 MOV AH,02H   
 INT 21H   
 INC BX   
 JMP l1   
   
MAIN:   
 MOV DX,OFFSET string2   
 MOV AH,09H   
 INT 21H   
 MOV AH,01H   
 INT 21H   
 push ax   
 CMP AL,'Q'   
 JZ EXIT   
 CMP AL,'q'   
 JZ EXIT   
 cmp al,'T'   
 jz ASCTB   
 cmp al,'t'   
 jz ASCTB   
   
 MOV DX,OFFSET RESULT   
 MOV AH,09H   
 INT 21H   
 pop ax   
 call ascii   
 jmp MAIN   
EXIT:   
 MOV AX,4C00H   
 INT 21H   
   
ASCTB:   
 push ax   
 push cx   
 push dx   
   
 mov bl, 20H   
 call newlne   
 MOV DX,OFFSET ASCTT   
 MOV AH,09H   
 INT 21H   
 TABLE:   
   
 ;;;;;;;;;;;;   
 ; 换行   
 ;;;;;;;;;;;;   
 mov ah,02H   
 mov dl,0AH   
 int 21H   
 mov dl,0DH   
 int 21H   
 mov cx,6   
 row:   
 ;;;;;;;;;;;;;;;;;;;   
 ; 显示字符   
 ;;;;;;;;;;;;;;;;;   
 mov dl,bl   
 mov ah,02H   
 int 21H   
 mov dl,20H   
 int 21H   
 ;;;;;;;;;;;;;;;   
 ; 显示ascii   
 ;;;;;;;;;;;;;   
 mov al,bl   
 call ascii   
 mov dl,09   
 int 21H   
   
 inc bl   
 cmp bl,7FH   
 jnb TABLEEND   
   
   
 loop row   
 jmp TABLE   
   
 TABLEEND:   
 pop dx   
 pop cx   
 pop ax   
jmp MAIN   
   
newlne proc   
push ax   
push dx   
mov ah,02H   
mov dl,0AH   
int 21H   
mov dl,0DH   
int 21H   
pop dx   
pop ax   
ret   
newlne endp   
   
ascii proc   
 push ax   
 push dx   
 push cx ;保护现场   
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
; 显示第一位数字   
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
 PUSH AX   
 AND AL,0F0H   
 mov cl,4   
 SHR AL,cl   
 ADD AL,30H   
 MOV DL,AL   
 MOV AH,02H   
 INT 21H   
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
; 显示第二位数字   
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
 POP AX   
 AND AL,0FH   
 CMP AL,09H   
 JNA ascend   
 ADD AL,07H ;字母特殊处理   
ascend:   
 add al,30h   
 mov dl,al   
 mov ah,02h   
 int 21H   
 pop cx   
 pop dx   
 pop ax   
 ret   
ascii endp   
 CODE ENDS   
 END START

# 实验二 数码转换实验

## 一、实验目的

1. 掌握不同进制数及编码相互转换的程序设计方法。
2. 掌握运算类指令编程及调试方法。
3. 掌握循环程序的设计方法。

## 二、实验内容

1. 重复从键盘输入不超过5位的十进制数，按回车键结束输入；
2. 将该十进制数转换成二进制数；结果以2进制数的形式显示在屏幕上；
3. 如果输入非数字字符，则报告出错信息，重新输入；
4. 直到输入“Q”或‘q’时程序运行结束。
5. 键盘输入一字符串，以空格结束，统计其中数字字符的个数，在屏幕显示。

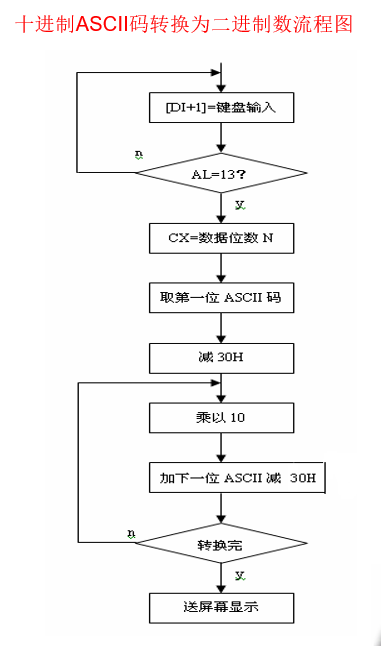
## 三、实验原理

十进制数可以表示为：

其中表十进制数1、2、3、…、9、0。

上式可以转换为：

由上式可归纳出十进制数转换为二进制数的方法：从十进制数的最高位Dn开始做乘10加次位的操作，依此类推，则可求出二进制数结果。转换过程可参考图2.1十进制ASCII码转换为二进制数流程图





## 四、实验过程

1. 通过 int 21H 中断的 0AH 号功能实现字符串的读入。读入的内容会以 ASCII 码的形式存放在 dx 寄存器所指向的内存单元中。因此需要提前获取足够大小的空间来存放读入的字符串

* mov dx, offset INPUT   
  mov ah, 0AH   
  int 21H   
  mov bx,offset INPUT+1   
  mov cx,[bx]   
  mov ch,0   
  mov dx, offset INPUT+2
* 这样操作后，cx 寄存器存放了读取的字符串长度，dx 寄存器存放了字符串的起始地址

1. 编写十进制转二进制子程序，根据实验原理，通过移位和乘法运算，实现到二进制的转换

* Binary proc   
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
  ; 10 进制转二进制子程序   
  ; ax: 返回的8位二进制数   
  ; dx：ASCII 形式的 10 进制字符串位置   
  ; cx: 10 进制字符串长度   
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
   push bx   
   push cx   
   push dx ;保护现场   
   push si   
   push dx   
     
   mov bx,0   
   mov si,dx   
   mov ah, 0   
   mov al,[si]   
   sub ax,30H   
   cmp cx,1H   
   je BinEnd   
   mov dx,10   
   dec cx   
  Binlop:   
   inc si   
   mov dx,10   
   mul dx   
   mov bl,[si]   
   sub bl,30H   
   cmp bl,9   
   ja BinErr   
   add ax,bx   
   loop Binlop   
   jmp BinEnd   
  BinErr:   
   mov dx, offset ERRINF   
   call newlne   
   pop dx   
   call CNTNUM   
   call newlne   
   mov ah, 09H   
   int 21H   
  BinEnd:   
   pop si   
   pop dx   
   pop cx   
   pop bx   
     
   ret   
  Binary endp

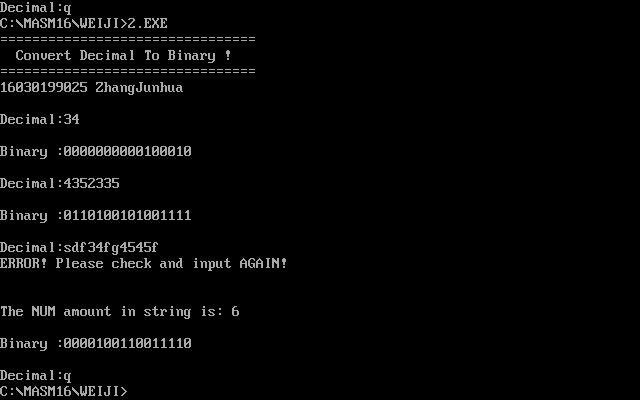
1. 编写二进制显示子程序。

* 通过二进制转换子程序，转换得到的二进制存放在 ax 寄存器中，通过移位运算，检查溢出标志位，判断是否溢出，来确定输出 0 或 1
* DispB proc   
  ;;;;;;;;;;;;;;;;;;;;;;;   
  ; 二进制显示子程序   
  ; ax: 待显示的二进制数据   
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
   push ax   
   push bx   
   push cx   
   push dx   
   mov bx,ax   
   mov cx,16   
  s: mov dl,'0'   
   rol bx,1   
   jnc s1   
   mov dl,'1'   
  s1: mov ah,02h   
   int 21h   
   loop s   
     
   call newlne   
   pop dx   
   pop cx   
   pop bx   
   pop ax   
   ret   
  DispB endp

1. 编写异常处理程序，当输入非数字型字符时，给出异常信息提示，并统计字符串中数字的数量

* CNTNUM proc   
   push ax   
   push bx   
   push cx   
   push dx   
     
   mov si, dx   
   mov ax,0   
  CHECK:   
   mov bl,[si]   
   inc si   
   cmp bl,30H   
   jb NOTNUM   
   cmp bl,39H   
   ja NOTNUM   
   inc ax   
  NOTNUM:   
   loop CHECK   
     
   add ax,30H   
   mov dx,ax   
   mov ah,02H   
   int 21h   
     
   pop dx   
   pop cx   
   pop bx   
   pop ax   
     
   ret   
  CNTNUM endp

## 五、实验结果



可见实验内容 1至5 均已完成

### 完整实验代码：

DATA SEGMENT   
Sno DB '16030199025 ZhangJunhua',0AH,0DH,'$'   
string1 DB 0AH,0DH,'Convert Decimal To Binary !',0AH,0DH,0AH,0DH,'$'   
string2 DB 0AH,0DH,'Decimal:$'   
INPUT DB 20H   
DB 100 DUP('$')   
RESULT DB 0AH,0DH,'Binary :$'   
ERRINF DB 'ERROR! Please check and input AGAIN!',0AH,0DH,0AH,0DH,'$'   
INFO DB '================================',0AH,0D   
 DB ' Convert Decimal To Binary ! ',0AH,0DH   
 DB '================================',0AH,0DH,'$'   
DATA ENDS   
   
STACK SEGMENT   
DB 100 DUP(0)   
STACK ENDS   
   
CODE SEGMENT   
ASSUME CS:CODE,DS:DATA,SS:STACK   
   
START:   
 MOV AX,STACK   
 MOV SS,AX   
 MOV AX,DATA   
 MOV DS,AX   
 MOV DX,OFFSET INFO   
 MOV AH,09H   
 INT 21H   
 MOV DX,OFFSET Sno   
 INT 21H   
 MOV BX,0   
SCAN:   
 mov dx, offset string2   
 mov ah, 09H   
 int 21H   
   
 mov dx, offset INPUT   
 mov ah, 0AH   
 int 21H   
   
 mov bx,offset INPUT+1   
 mov cx,[bx]   
 mov ch,0   
 mov dx, offset INPUT+2   
   
 ;;;;;;;;; 检查退出逻辑   
 mov al,[bx+1]   
 cmp al,'q'   
 je exit   
 cmp al,'Q'   
 je exit   
   
 ;;;;;;;;; 执行主程序   
 call newlne   
 call Binary   
 ;打印提示信息   
 push ax   
 mov dx, offset RESULT   
 mov ah, 09H   
 int 21H   
 pop ax   
 ;显示二进制   
 call DispB   
   
 jmp SCAN   
   
exit:   
 mov ah, 4cH   
 int 21H   
 int 21H   
   
Binary proc   
;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
; 10 进制转二进制子程序   
; ax: 返回的8位二进制数   
; dx：ASCII 形式的 10 进制字符串位置   
; cx: 10 进制字符串长度   
;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
 push bx   
 push cx   
 push dx ;保护现场   
 push si   
 push dx   
   
 mov bx,0   
 mov si,dx   
 mov ah, 0   
 mov al,[si]   
 sub ax,30H   
 cmp cx,1H   
 je BinEnd   
 mov dx,10   
 dec cx   
Binlop:   
 inc si   
 mov dx,10   
 mul dx   
 mov bl,[si]   
 sub bl,30H   
 cmp bl,9   
 ja BinErr   
 add ax,bx   
 loop Binlop   
 jmp BinEnd   
BinErr:   
 mov dx, offset ERRINF   
 call newlne   
 pop dx   
 call CNTNUM   
 call newlne   
 mov ah, 09H   
 int 21H   
BinEnd:   
 pop si   
 pop dx   
 pop cx   
 pop bx   
   
 ret   
Binary endp   
   
CNTNUM proc   
 push ax   
 push bx   
 push cx   
 push dx   
   
 mov si, dx   
 mov ax,0   
CHECK:   
 mov bl,[si]   
 inc si   
 cmp bl,30H   
 jb NOTNUM   
 cmp bl,39H   
 ja NOTNUM   
 inc ax   
NOTNUM:   
 loop CHECK   
   
 add ax,30H   
 mov dx,ax   
 mov ah,02H   
 int 21h   
   
 pop dx   
 pop cx   
 pop bx   
 pop ax   
   
 ret   
CNTNUM endp   
   
DispB proc   
;;;;;;;;;;;;;;;;;;;;;;;   
; 二进制显示子程序   
; ax: 待显示的二进制数据   
;;;;;;;;;;;;;;;;;;;;;;;;;;;;   
 push ax   
 push bx   
 push cx   
 push dx   
 mov bx,ax   
 mov cx,16   
s: mov dl,'0'   
 rol bx,1   
 jnc s1   
 mov dl,'1'   
s1: mov ah,02h   
 int 21h   
 loop s   
   
 call newlne   
 pop dx   
 pop cx   
 pop bx   
 pop ax   
 ret   
DispB endp   
   
newlne proc   
push ax   
push dx   
mov ah,02H   
mov dl,0AH   
int 21H   
mov dl,0DH   
int 21H   
pop dx   
pop ax   
ret   
newlne endp   
   
CODE ENDS   
end START

# 实验三 基本IO拓展实验

## 一、实验目的

1. 了解 TTL 芯片扩展简单 I/O 口的方法。
2. 掌握数据输入输出程序编制的方法。

## 二、实验内容说明

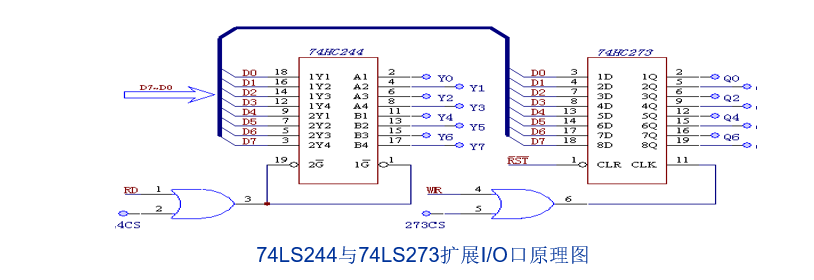
本实验要求用74LS244作为输入口，读取开关状态，并将此状态通过74LS273连到发光二极管显示。具体实验内容如下：

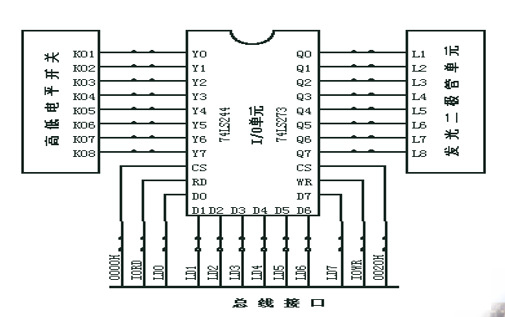
1. 开关Yi为低电平时对应的发光二极管亮，Yi为高电平时对应的发光二极管灭。
2. 当开关Yi全为高电平时，发光二极管Qi从左至右轮流点亮。
3. 当开关Yi全为低电平时，发光二极管Qi从右至左轮流点亮。
4. 主设计控制及显示模式，完成编程调试，演示实验结果。

## 三、实验原理

74LS244是一种三态输出的8总线缓冲驱动器，无锁存功能，当G为低电平，Ai信号传送到Yi，当为高电平时，Yi处于禁止高阻状态;

74LS273是一种带清除功能的8D触发器， 1D～8D为数据输入端，1Q～8Q为数据输出端，正脉冲触发，低电平清除，常用作8位地址锁存器。





## 四、实验步骤

1. 按照实验连线图连接：
   * 244的CS接到ISA总线接口模块的0000H，Y7—Y0——开关K1—K8。
   * 273的CS接到ISA总线接口模块的0020H，Q7—Q0——发光二极管L1—L8。
   * 该模块的WR、RD分别连到ISA总线接口模块的IOWR、IORD。
   * 该模块的数据（AD0～AD7）连到ISA总线接口模块的数据（LD0～LD7）。
2. 编写实验程序，编译链接，运行程序
3. 拨动开关，观察发光二极管的变化。

## 五、实验结果

编译链接项目后，改变开关Yi可观察到

开关Yi为低电平时对应的发光二极管亮，Yi为高电平时对应的发光二极管灭。

当开关Yi全为高电平时，发光二极管Qi从左至右轮流点亮。

当开关Yi全为低电平时，发光二极管Qi从右至左轮流点亮。

### 实验源代码：

MY\_STACK SEGMENT PARA 'STACK'   
 DB 100 DUP(?)   
MY\_STACK ENDS   
   
MY\_DATA SEGMENT PARA 'DATA'   
IO\_9054base\_address DB 4 DUP(0) ;PCI卡9054芯片I/O基地址暂存空间   
IO\_base\_address DB 4 DUP(0) ;PCI卡I/O基地址暂存空间   
pcicardnotfind DB 0DH,0AH,'pci card not find or address/interrupt error !!!',0DH,0AH,'$'   
GOOD DB 0DH,0AH,'The Program is Executing !',0DH,0AH,'$'   
LS244 DW 00000H   
LS273 DW 00020H   
;   
;   
DELAY\_SET EQU 0FFFH ;延时常数   
   
MY\_DATA ENDs   
   
MY\_CODE SEGMENT PARA 'CODE'   
   
MY\_PROC PROC FAR   
 ASSUME CS:MY\_CODE, DS:MY\_DATA, SS:MY\_STACK   
MAIN:   
.386 ;386模式编译   
 MOV AX,MY\_DATA   
 MOV DS,AX   
 MOV ES,AX   
 MOV AX,MY\_STACK   
 MOV SS,AX   
 CALL FINDPCI ;自动查找PCI卡资源及IO口基址   
 MOV CX,word ptr IO\_base\_address   
; MOV CX,0E800H ;直接加入(E800:本机PCI卡IO口基址)   
   
 ADD LS244,CX ;PCI卡IO基址+偏移   
 ADD LS273,CX   
   
; 插入功能实现代码   
START:   
 MOV DX, LS244   
 IN AL, DX   
 CMP AL, 11111101B   
 JE HIGH   
 CMP AL, 00H   
 JE LOW   
 MOV DX, LS273   
 OUT DX, AL   
 call DELAY   
 JMP START   
   
HIGH:   
 MOV CX, 8   
 MOV AL, 7FH   
 MOV DX, LS273   
 ROUND1:   
 OUT DX, AL   
 RCL AL,1   
 CALL DELAY   
 LOOP ROUND1   
 MOV AL,00H   
 OUT DX,AL   
 CALL DELAY   
 OUT DX,AL   
 CALL DELAY   
   
 JMP START   
LOW:   
 MOV CX, 8   
 MOV AL, 7FH   
 MOV DX, LS273   
 ROUND2:   
 OUT DX, AL   
 RCR AL,1   
 CALL DELAY   
 LOOP ROUND2   
 MOV AL,00H   
 OUT DX,AL   
 CALL DELAY   
 OUT DX,AL   
 CALL DELAY   
   
 JMP START   
MY\_PROC ENDp   
   
   
   
   
;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
; /\*延时程序\*/   
;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
;   
DELAY PROC NEAR ;延时程序   
 PUSHF   
 PUSH DX   
 PUSH CX   
 MOV DX,DELAY\_SET   
D1: MOV CX,-1   
D2: DEC CX   
 JNZ D2   
 DEC DX   
 JNZ D1   
 POP CX   
 POP DX   
 POPF   
 RET   
DELAY ENDp   
;   
;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
; /\* 找卡子程序 \*/   
;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
;   
;FUNCTION CODE   
IO\_port\_addre EQU 0CF8H ;32位配置地址端口   
IO\_port\_data EQU 0CFCH ;32位配置数据端口   
IO\_PLX\_ID EQU 200810B5H ;PCI卡设备及厂商ID   
BADR0 = 10H ;基地址寄存器0   
BADR1 = 14H ;基地址寄存器1   
BADR2 = 18H ;基地址寄存器2   
BADR3 = 1CH ;基地址寄存器3   
   
FINDPCI PROC NEAR ;查找PCI卡资源并显示   
 PUSHAD   
 PUSHFD   
 MOV EBX,080000000H   
FINDPCI\_next:   
 ADD EBX,100H   
 CMP EBX,081000000H   
 JNZ findpci\_continue   
 MOV DX,offset pcicardnotfind ;显示未找到PCI卡提示信息   
 MOV AH,09H   
 INT 21H   
 MOV AH,4CH   
 INT 21H ;退出   
findpci\_continue:   
 MOV DX,IO\_port\_addre   
 MOV EAX,EBX   
 OUT DX,EAX ;写地址口   
 MOV DX,IO\_port\_data   
 IN EAX,DX ;读数据口   
 CMP EAX,IO\_PLX\_ID   
 JNZ findpci\_next ;检查是否发现PCI卡   
   
 MOV DX,IO\_port\_addre   
 MOV EAX,EBX   
 ADD EAX,BADR1   
 OUT DX,EAX ;写地址口   
 MOV DX,IO\_port\_data   
 IN EAX,DX ;读数据口   
 MOV dword ptr IO\_9054base\_address,EAX   
 AND EAX,1   
 JZ findPCI\_next ;检查是否为i/o基址信息   
 MOV EAX,dword ptr IO\_9054base\_address   
 AND EAX,0fffffffeh   
 MOV dword ptr IO\_9054base\_address,EAX ;去除i/o指示位并保存   
   
 MOV DX,IO\_port\_addre   
 MOV EAX,EBX   
 ADD EAX,BADR2   
 OUT DX,EAX ;写地址口   
 MOV DX,IO\_port\_data   
 IN EAX,DX ;读数据口   
 MOV dword ptr IO\_base\_address,EAX   
 AND EAX,1   
 JZ findPCI\_next ;检查是否为i/o基址信息   
 MOV EAX,dword ptr IO\_base\_address   
 AND EAX,0fffffffeh   
 MOV dword ptr IO\_base\_address,EAX ;去除i/o指示位并保存   
 MOV DX,offset good ;显示开始执行程序信息   
 MOV AH,09H   
 INT 21H   
 POPfd   
 POPad   
 RET   
findPCI ENDP   
   
MY\_CODE ENDS   
   
 END MAIN

# 实验四 可编程并行接口 8255 实验

## 一、实验目的

1. 了解可编程并行接口8255的内部结构
2. 掌握工作方式、初始化编程及应用。

## 二、实验内容

1. 流水灯实验：利用8255的A口、B口循环点亮发光二极管。
2. 交通灯实验：利用8255的A口模拟交通信号灯。
3. I/O输入输出实验：利用8255的A口读取开关状态，8255的B口把状态送发光二极管显示。
4. 在完成(1)基础上，增加通过读取开关控制流水灯的循环方向和循环方式。
5. 在完成(2)基础上，增加通过读取开关控制交通红绿灯的亮灭时间。

## 三、实验原理

1. 8255A的内部结构

（1）数据总线缓冲器：这是一个双向三态的8位数据缓冲器，它是8255A与微机系统数据总线的接口。输入输出的数据、CPU输出的控制字以及CPU输入的状态信息都是通过这个缓冲器传送的。

（2）三个端口A，B和C：A端口包含一个8位数据输出锁存器和缓冲器，一个8位数据输入锁存器。B端口包含一个8位数据输入/输出锁存器和缓冲器，一个8位数据输入缓冲器。C端口包含一个8位数据输出锁存器及缓冲器，一个8位数据输入缓冲器（输入没有锁存器）。

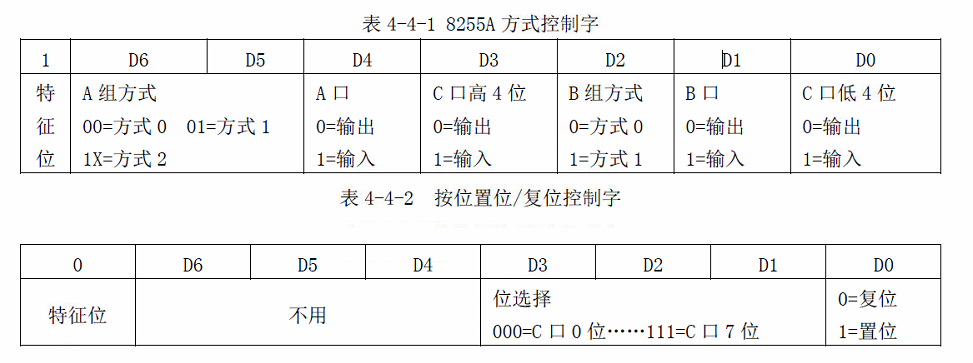
（3）A组和B组控制电路：这是两组根据CPU输出的控制字控制8255工作方式的电路，它们对于CPU而言，共用一个端口地址相同的控制字寄存器，接收CPU输出的一字节方式控制字或对C口按位复位字命令。方式控制字的高5位决定A组工作方式，低3位决定B组的工作方式。对C口按位复位命令字可对C口的每一位实现置位或复位。A组控制电路控制A口和C口上半部，B组控制电路控制B口和C口下半部。

（4）读写控制逻辑：用来控制把CPU输出的控制字或数据送至相应端口，也由它来控制把状态信息或输入数据通过相应的端口送到CPU。

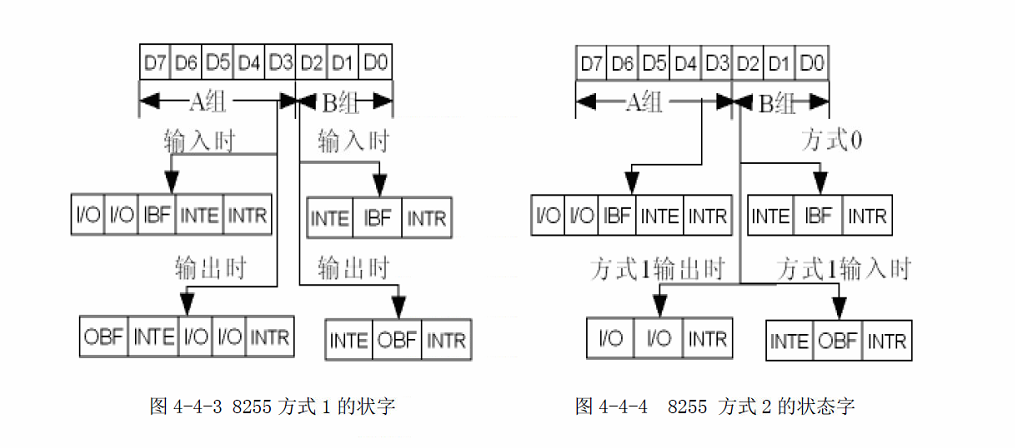
2. 8255A的工作方式

方式0—基本输入输出方式；方式1—选通输入输出方式；方式2—双向选通输入输出方式。

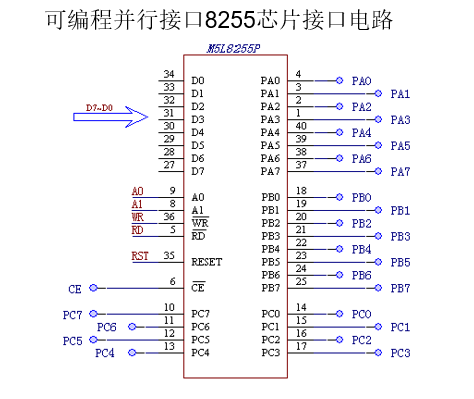
3. 8255A的控制字



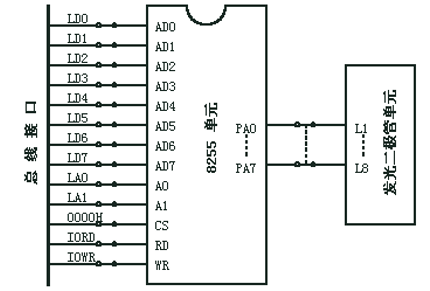
4. 8255A的状态字



8255是一个通用可编程并行接口电路。它具有A、B、C三个8位并行口。其中C口也可用作A、B口的联络信号及中断申请信号。通过编程，它可以被设置为基本输入输出、选通输入输出以及双向传送方式。对于C口还具有按位置0、1的功能。



## 四、实验步骤



* 模块的WR、RD分别连到ISA总线接口模块的IOWR、IORD。
* 模块的数据（AD0～AD7）、地址线（A0～A7）分别连到ISA总线接口模块的数据（LD0～LD7）、地址线（LA0～LA7）。
* 8255模块选通线CE连到ISA总线接口模块的0000H。
* 8255的PA0～PA7连到发光二极管的L0～L7；8255的PB0～PB7连到发光二极管的L8～L15。
* 编写 8255 驱动程序
* 运行程序，观察发光二极管。

## 五、实验结果

全速运行程序后，可观察到发光二极管被循环点亮

### 完成程序源代码：

MY\_STACK SEGMENT PARA 'STACK'   
 DB 100 DUP(?)   
MY\_STACK ENDS   
   
MY\_DATA SEGMENT PARA 'DATA'   
IO\_9054base\_address DB 4 DUP(0) ;PCI卡9054芯片I/O基地址暂存空间   
IO\_base\_address DB 4 DUP(0) ;PCI卡I/O基地址暂存空间   
pcicardnotfind DB 0DH,0AH,'pci card not find or address/interrupt error !!!',0DH,0AH,'$'   
GOOD DB 0DH,0AH,'The Program is Executing !',0DH,0AH,'$'   
LS244 DW 00000H   
LS273 DW 00020H   
LS8255 DW 00000H   
LS8255W DW 00003H   
RA DB ?   
LB DB ?   
;   
;   
DELAY\_SET EQU 0FFFH ;延时常数   
   
MY\_DATA ENDs   
   
MY\_CODE SEGMENT PARA 'CODE'   
   
MY\_PROC PROC FAR   
 ASSUME CS:MY\_CODE, DS:MY\_DATA, SS:MY\_STACK   
MAIN:   
.386 ;386模式编译   
 MOV AX,MY\_DATA   
 MOV DS,AX   
 MOV ES,AX   
 MOV AX,MY\_STACK   
 MOV SS,AX   
 CALL FINDPCI ;自动查找PCI卡资源及IO口基址   
 MOV CX,word ptr IO\_base\_address   
; MOV CX,0E800H ;直接加入(E800:本机PCI卡IO口基址)   
   
 ;PCI卡IO基址+偏移   
   
 ADD LS8255,CX   
 ADD LS8255W,CX   
; 插入功能实现代码   
MOV RA,7FH   
MOV LB,0FEH   
   
READ1:MOV AX,80H   
 MOV DX,LS8255W   
 OUT DX,AX   
 MOV DX,LS8255   
 MOV AL,0E7H   
 OUT DX,AL   
 CALL DELAY   
 CALL BREAK   
 MOV AL,0DBH   
 MOV DX,LS8255   
 OUT DX,AL   
 CALL DELAY   
 CALL BREAK   
 MOV AL,0BDH   
 MOV DX,LS8255   
 OUT DX,AL   
 CALL DELAY   
 CALL BREAK   
 MOV AL,07EH   
 MOV DX,LS8255   
 OUT DX,AL   
 CALL DELAY   
 CALL BREAK   
 MOV DX,LS8255W   
 OUT DX,AX   
 MOV DX,LS8255   
 MOV AL,0E7H   
 OUT DX,AL   
 CALL DELAY   
 CALL BREAK   
 MOV AL,0DBH   
 MOV DX,LS8255   
 OUT DX,AL   
 CALL DELAY   
 CALL BREAK   
 MOV AL,0BDH   
 MOV DX,LS8255   
 OUT DX,AL   
 CALL DELAY   
 CALL BREAK   
 MOV AL,07EH   
 MOV DX,LS8255   
 OUT DX,AL   
 CALL DELAY   
 CALL BREAK   
 JMP READ1   
   
   
MY\_PROC ENDp   
   
BREAK PROC NEAR   
PUSHF   
PUSH AX   
PUSH DX   
MOV AH,06H   
MOV DL,0FFH   
INT 21H   
JE RETURN   
MOV AX,4C00H   
INT 21H   
RETURN:   
 POP DX   
 POP AX   
 POPF   
 RET   
 BREAK ENDP   
 DELAY PROC NEAR   
   
   
;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
; /\*延时程序\*/   
;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
;   
DELAY PROC NEAR ;延时程序   
 PUSHF   
 PUSH DX   
 PUSH CX   
 MOV DX,DELAY\_SET   
D1: MOV CX,-1   
D2: DEC CX   
 JNZ D2   
 DEC DX   
 JNZ D1   
 POP CX   
 POP DX   
 POPF   
 RET   
DELAY ENDp   
;   
;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
; /\* 找卡子程序 \*/   
;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
;   
;FUNCTION CODE   
IO\_port\_addre EQU 0CF8H ;32位配置地址端口   
IO\_port\_data EQU 0CFCH ;32位配置数据端口   
IO\_PLX\_ID EQU 200810B5H ;PCI卡设备及厂商ID   
BADR0 = 10H ;基地址寄存器0   
BADR1 = 14H ;基地址寄存器1   
BADR2 = 18H ;基地址寄存器2   
BADR3 = 1CH ;基地址寄存器3   
   
FINDPCI PROC NEAR ;查找PCI卡资源并显示   
 PUSHAD   
 PUSHFD   
 MOV EBX,080000000H   
FINDPCI\_next:   
 ADD EBX,100H   
 CMP EBX,081000000H   
 JNZ findpci\_continue   
 MOV DX,offset pcicardnotfind ;显示未找到PCI卡提示信息   
 MOV AH,09H   
 INT 21H   
 MOV AH,4CH   
 INT 21H ;退出   
findpci\_continue:   
 MOV DX,IO\_port\_addre   
 MOV EAX,EBX   
 OUT DX,EAX ;写地址口   
 MOV DX,IO\_port\_data   
 IN EAX,DX ;读数据口   
 CMP EAX,IO\_PLX\_ID   
 JNZ findpci\_next ;检查是否发现PCI卡   
   
 MOV DX,IO\_port\_addre   
 MOV EAX,EBX   
 ADD EAX,BADR1   
 OUT DX,EAX ;写地址口   
 MOV DX,IO\_port\_data   
 IN EAX,DX ;读数据口   
 MOV dword ptr IO\_9054base\_address,EAX   
 AND EAX,1   
 JZ findPCI\_next ;检查是否为i/o基址信息   
 MOV EAX,dword ptr IO\_9054base\_address   
 AND EAX,0fffffffeh   
 MOV dword ptr IO\_9054base\_address,EAX ;去除i/o指示位并保存   
   
 MOV DX,IO\_port\_addre   
 MOV EAX,EBX   
 ADD EAX,BADR2   
 OUT DX,EAX ;写地址口   
 MOV DX,IO\_port\_data   
 IN EAX,DX ;读数据口   
 MOV dword ptr IO\_base\_address,EAX   
 AND EAX,1   
 JZ findPCI\_next ;检查是否为i/o基址信息   
 MOV EAX,dword ptr IO\_base\_address   
 AND EAX,0fffffffeh   
 MOV dword ptr IO\_base\_address,EAX ;去除i/o指示位并保存   
 MOV DX,offset good ;显示开始执行程序信息   
 MOV AH,09H   
 INT 21H   
 POPfd   
 POPad   
 RET   
findPCI ENDP   
   
   
MY\_CODE ENDS   
   
 END MAIN

## 实验心得

这学期的计算机组成原理课程设计让我受益菲浅。这次的微机原理课程，我详细的了解了 74LS244、74LS273、可编程并行接口8255的电路设计，驱动程序的编写方式，并通过自己的亲自实践，使用编写的代码完成了对 LED 发光二极管的控制。看着试验台上闪亮的光点，获得了极大的成就感。

没有接触过计算机或者对计算机不是特别了解的人可能觉得计算机特别神秘而且不知道为什么它可以实现那么复杂的功能，而就我们而言越是深入学习越是渴望了解其工作原理。很幸运这学期我们开设了微机原理实验，《微机原理与接口》这门课程是我们计算机专业一门很重要的专业课。这学期的微机原理实验，更是让我对课堂上刚刚学过的知识有了亲身的应用和体验，通过自己的亲自操作让我对计算机的基本结构，底层硬件语言，基本组成与结构原理有了更加深入的了解，特别是前两次汇编实验，极大的提升了我的汇编代码编写能力，实验中遇到的问题和 BUG，提高了我动手调试的能力，后两次的接口实验，让我对软硬件的结合工作方式有了进一步的了解和认识，课本上的实例也不再那么空泛而变得鲜活起来。但是由于课程及实验时间的限制，我想我们学到的东西还是太少了，不过没关系，这毕竞为我们以后的学习打下了基础。

总之，这次微机原理给我提供了动手实验的机会，使我对计算机组成原理的相关知识有了更深的印象和认识。微机原理与接口是计算机专业的基础课。这门课对于使我们了解现代计算机的各个组成部分及其工作原理具有重要作用，对于我们后续课程的学习无疑也具有积极的意义。计算机专业是一个很渊博的专业，我们现在有很好的机会站在巨人的肩膀上学习，虽然通过这学期的课程设计学到了很多知识，但那只是计算机知识海洋中的一滴，我将继续努力对计算机原理方面进行深入的研究，了解更多计算机方面的知识，为以后打下坚实的基础。