

西安电子科技大学

微机原理与系统设计 课程实验报告

实验名称 微机系统实验

计算机科学与技术学院 1603019 班

姓名 张俊华 学号 16030199025

同作者

实验日期 2018 年 11 月 20 日-12 月 11 日

实验地点 E-II-311 实验批次 第二批

成绩

指导教师评语:

指导教师:

年 月 日

实验报告内容基本要求及参考格式

- 一、实验目的
- 二、实验所用仪器（或实验环境）
- 三、实验基本原理及步骤（或方案设计及理论计算）
- 四、实验数据记录（或仿真及软件设计）
- 五、实验结果分析及回答问题（或测试环境及测试结果）

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 interrupt 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。

AH 值	功 能	调用参数	结 果
1	键盘输入并回显		AL=输出字符
2	显示单个字符(带Ctrl+Break检查)	DL=输出字符	光标在字符后面
6	显示单个字符(无Ctrl+Break检查)	DL=输出字符	光标在字符后面
8	从键盘上读一个字符		AL=字符的ASCII码
9	显示字符串	DS:DX=串地址, '\$' 为结束字符	光标跟在串后面
4CH	返回DOS系统		AL=返回码

图 1.部分系统功能调用参考表

2. 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。

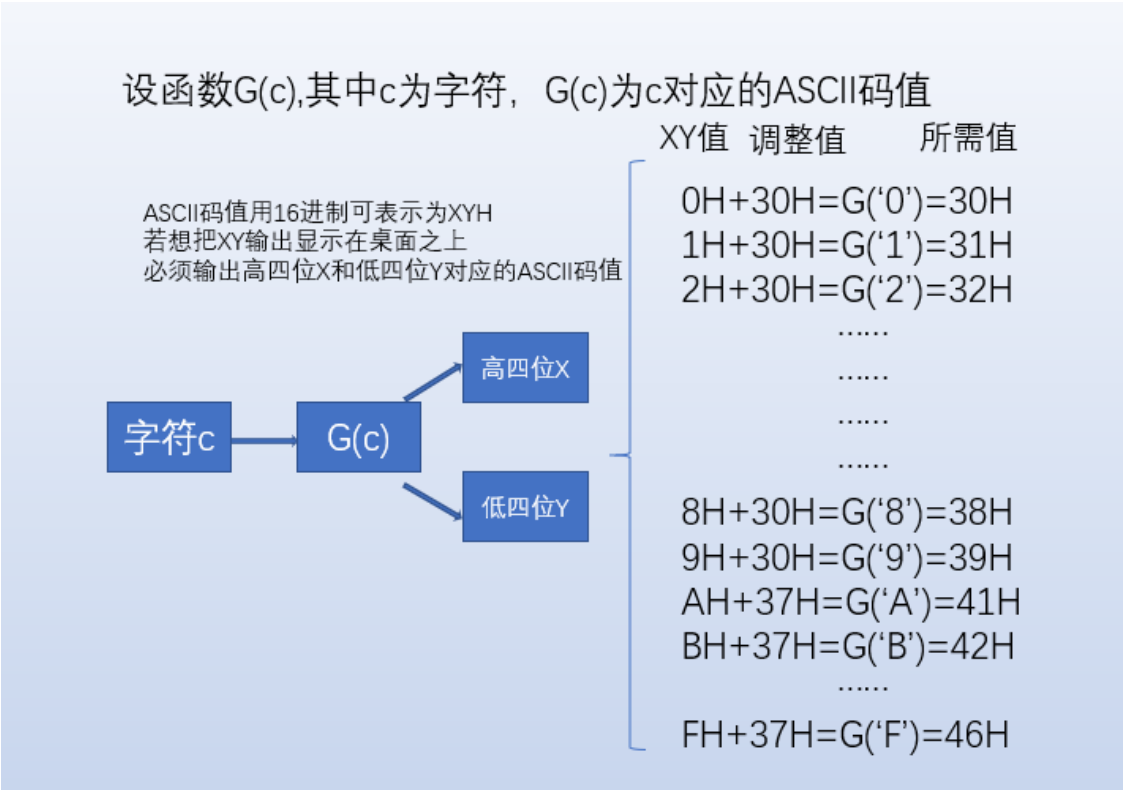


图 2.所表示数据与 ASCII 码的映射关系

3. 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。

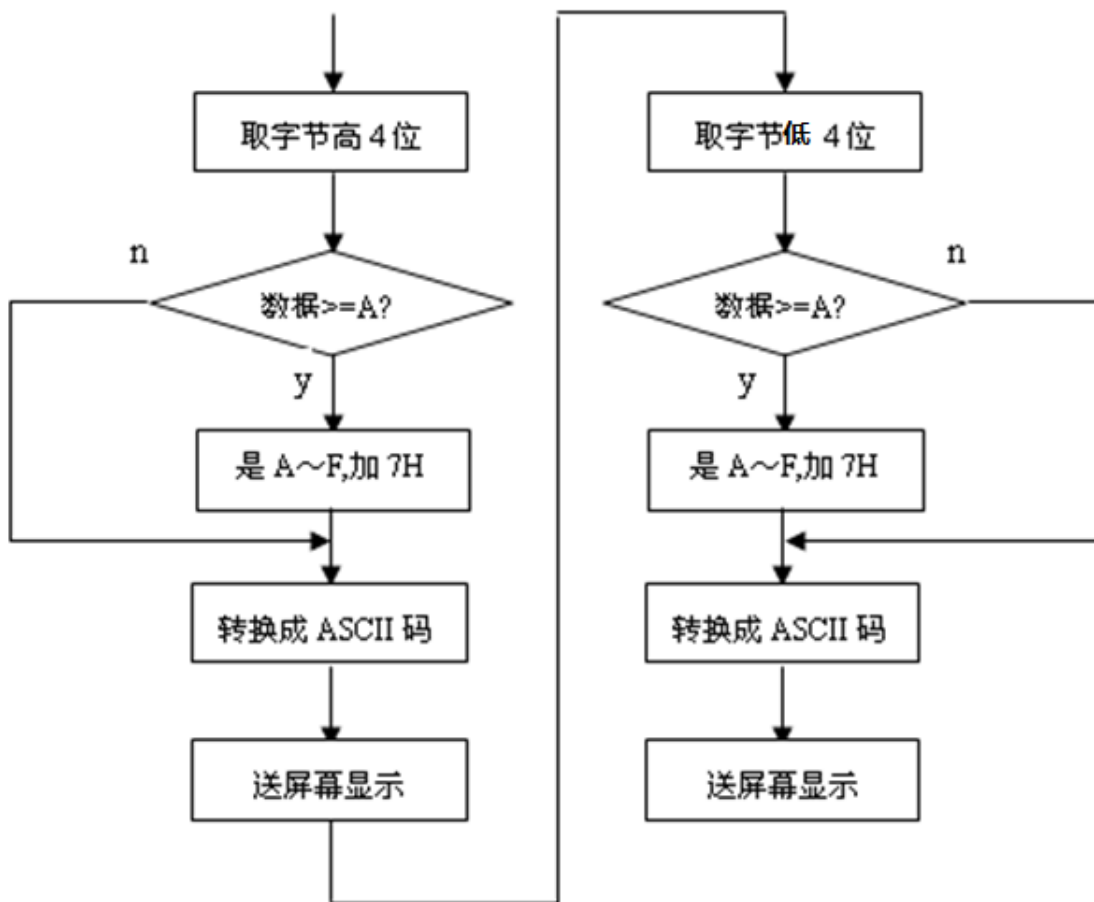


图3. 字符转换为ASCII 码流程图

4. 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

```

5. 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

```

6. 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

```
C:\MASM16\WEIJI>WEIJI.EXE
```

```
input Q/q to exit, input T/t to show ASCII table !
```

```
16030199025 ZhangJunhua
```

```
31 36 30 33 30 31 39 39 30 32 35 20 5A 68 61 6E 67 4A 75 6E 68 75 61 0A 0D
```

```
INPUT:d
```

```
ASCII :64
```

```
INPUT:g
```

```
ASCII :67
```

```
INPUT:a
```

```
ASCII :61
```

```
INPUT:f
```

```
ASCII :66
```

```
INPUT:f
```

```
ASCII :66
```

```
INPUT:4
```

```
ASCII :34
```

```
INPUT:4
```

```
ASCII :34
```

```
INPUT:
```

```
ASCII :34
```

```
INPUT:4
```

```
ASCII :34
```

```
INPUT:t
```

```
=====
          ASCII TABLE
=====
```

20	!	21	"	22	#	23	\$	24	%	25
& 26	'	27	(28)	29	*	2A	+	2B
, 2C	-	2D	.	2E	/	2F	0	30	1	31
2 32	3	33	4	34	5	35	6	36	7	37
8 38	9	39	:	3A	;	3B	<	3C	=	3D
> 3E	?	3F	@	40	A	41	B	42	C	43
D 44	E	45	F	46	G	47	H	48	I	49
J 4A	K	4B	L	4C	M	4D	N	4E	O	4F
P 50	Q	51	R	52	S	53	T	54	U	55
V 56	W	57	X	58	Y	59	Z	5A	[5B
\ 5C] 5D	^ 5E	_ 5F	` 60	a	61				
b 62	c	63	d	64	e	65	f	66	g	67
h 68	i	69	j	6A	k	6B	l	6C	m	6D
n 6E	o	6F	p	70	q	71	r	72	s	73
t 74	u	75	v	76	w	77	x	78	y	79
z 7A	{	7B		7C	}	7D	~	7E		

```
INPUT:_
```

完整源代码:

```
DATA SEGMENT
```

```
Sno DB '16030199025 ZhangJunhua',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 '=====','$'
```

```
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
```


- 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

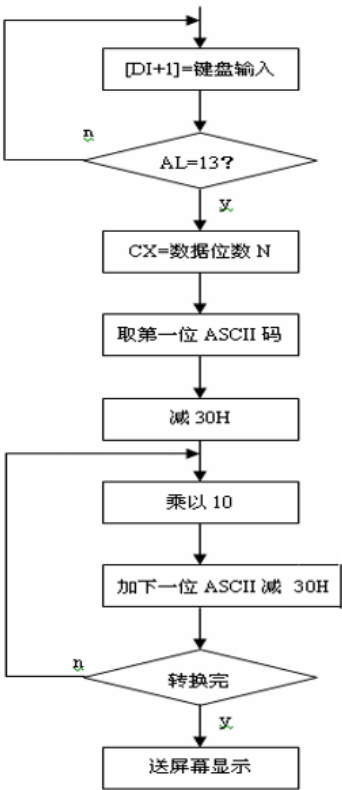
十进制数可以表示为： $D_n * 10^n + D_{n-1} * 10^{n-1} + \dots + D_0 * 10^0 = \sum D_i * 10^i$

其中 D_i 表十进制数 1、2、3、...、9、0。

上式可以转换为： $\sum D_i * 10^i = (((D_n * 10 + D_{n-1}) * 10 + D_{n-2}) * 10 + \dots + D_1) * 10 + D_0$

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 D_n 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

十进制ASCII码转换为二进制数流程图




```
    ret
DispB endp
```

4. 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

```
Decimal:q
C:\MASM16\WEIJI>2.EXE
=====
    Convert Decimal To Binary !
=====
16030199025 ZhangJunhua

Decimal:34

Binary :000000000100010

Decimal:4352335

Binary :011010010100111

Decimal:sdf34fg4545f
ERROR! Please check and input AGAIN!

The NUM amount in string is: 6

Binary :0000100110011110

Decimal:q
C:\MASM16\WEIJI>
```

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 '=====','$'
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

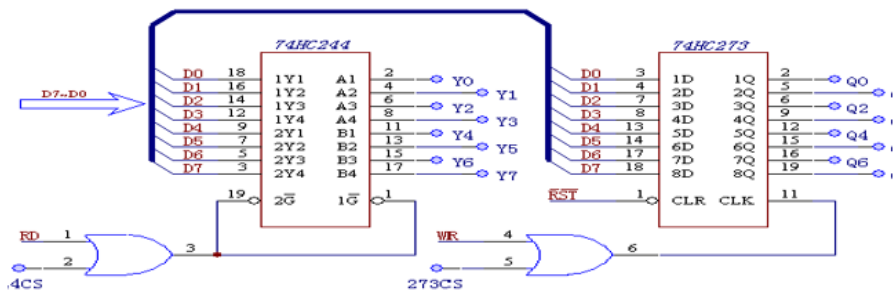
本实验要求用 74LS244 作为输入口，读取开关状态，并将此状态通过 74LS273 连到发光二极管显示。具体实验内容如下：

- 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。

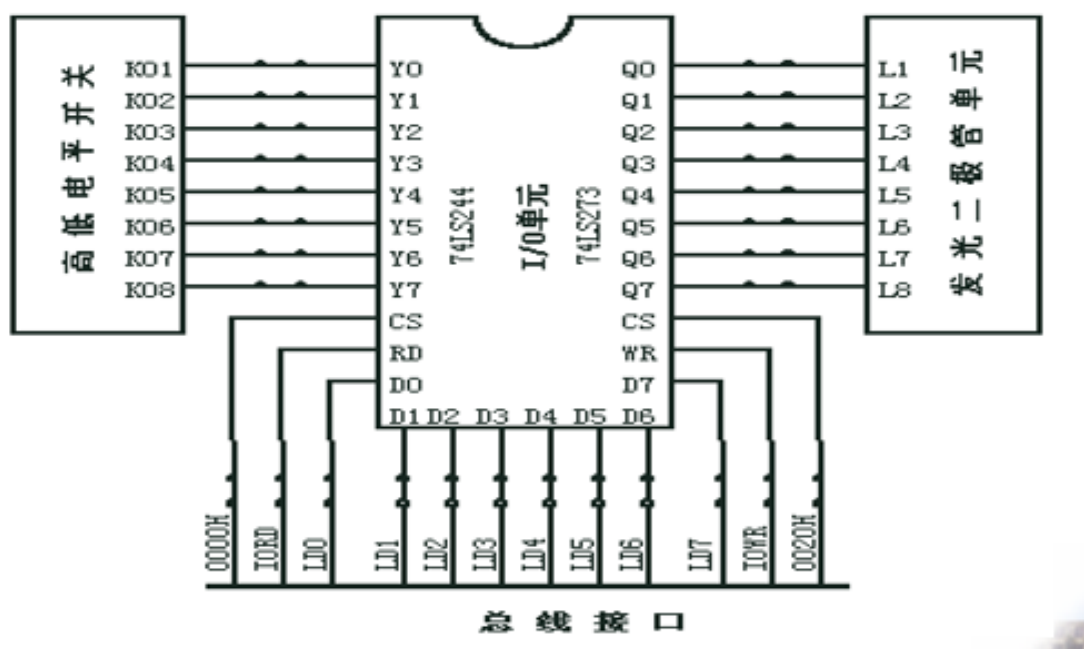
三、实验原理

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。



74LS244与74LS273扩展I/O口原理图



四、实验步骤

- 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）。
- 编写实验程序，编译链接，运行程序
- 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/
0 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
D1:        MOV     DX,DELAY_SET
D2:        MOV     CX,-1
          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,0fffffffh
    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,0fffffffh
                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

表 4-4-1 8255A 方式控制字

1	D6	D5	D4	D3	D2	D1	D0
特征位	A 组方式		A 口	C 口高 4 位	B 组方式	B 口	C 口低 4 位
	00=方式 0 01=方式 1		0=输出	0=输出	0=方式 0	0=输出	0=输出
	1X=方式 2		1=输入	1=输入	1=方式 1	1=输入	1=输入

表 4-4-2 按位置位/复位控制字

0	D6	D5	D4	D3	D2	D1	D0
特征位	不用			位选择 000=C 口 0 位……111=C 口 7 位			0=复位 1=置位

4. 8255A 的状态字

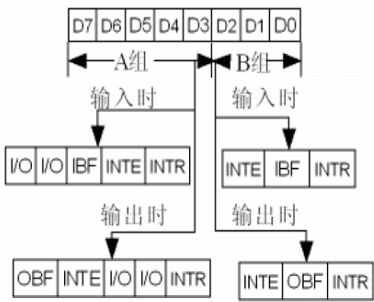


图 4-4-3 8255 方式 1 的状态

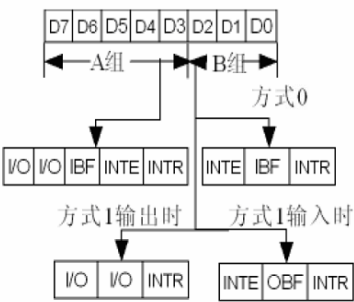
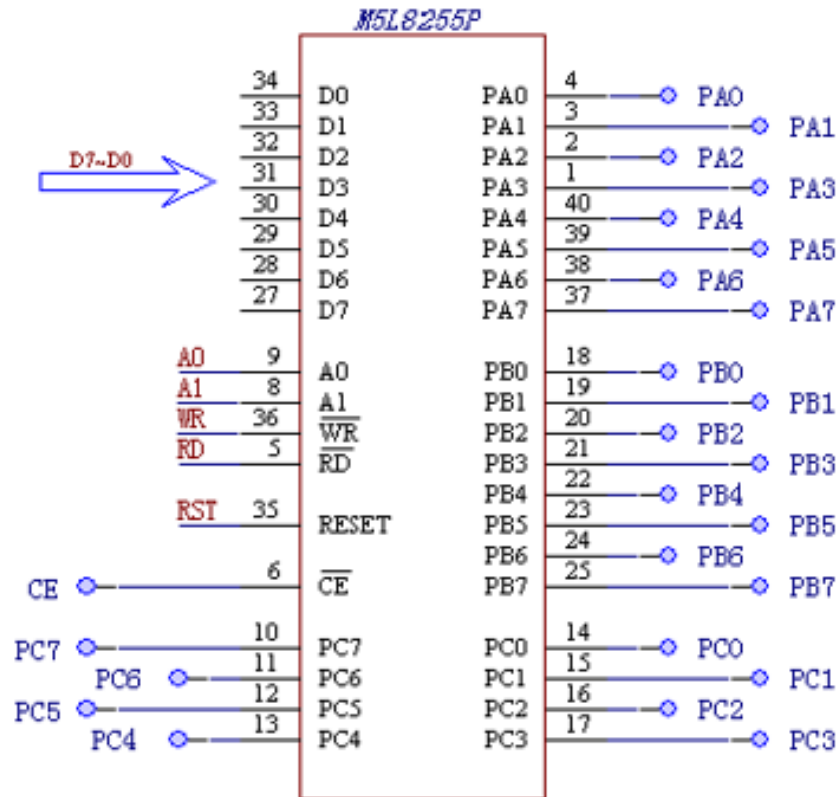


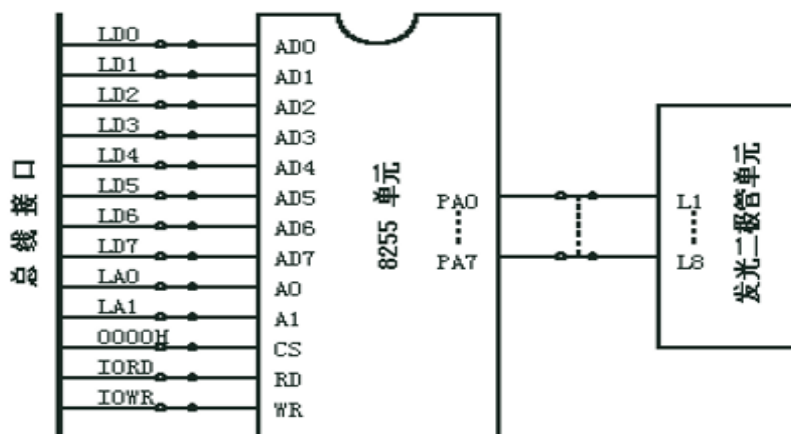
图 4-4-4 8255 方式 2 的状态字

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。

可编程并行接口8255芯片接口电路



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/interr
upt 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(E8
00: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

```

址口

```

MOV     DX,IO_port_data
IN      EAX,DX ;读数据口

```

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,0fffffffh
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,0fffffffh
        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匪浅。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，这毕竟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。