微机原理实验报告

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、心得

通过实验，深入掌握了汇编语言的基础编程方法及其在DOS功能调用中的应用。尤其是在调试过程中，对寄存器和数据区的观察帮助更好地理解指令执行的逻辑。此外，通过实现字符串显示和字符ASCII码的回显，强化了对程序流程的控制能力。实验全面展示了汇编语言的基本操作和调试过程，增强了对底层编程的认识，同时也感受到其逻辑严谨性和复杂性。

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16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：Dn*10n+Dn-1*10n-1+...+D0*100=S Di*10i

其中Di代表十进制数1、2、3、...、9、0。

上式可以转换为：S Di*10i= (((Dn*10+Dn-1) *10+ Dn-2)* 10+...+ D1) *10+ D0

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。

四、代码

```
_STACK SEGMENT PARA STACK'_STACK'
           DB 128 DUP(0)
_STACK ENDS
DATA SEGMENT
    hello    DB 'Input a number or an instruction!!Q OR q:
EXIT,s:SEARCH',0AH,0DH,'$'
    wrong    DB 0AH,0DH,'Wrong Input(only numbers!)',0AH,0DH,'$'
    endofhex DB 0AH,0DH,'Binary:',0AH,0DH,'$'
    finish   DB 0AH,0DH,'Finished',0AH,0DH,'$'
    hello2   DB 0AH,0DH,'Search number inyour string. Space to end
input',0AH,0DH,'Input string:',0AH,0DH,'$'
    finish2  DB 0AH,0DH,'FINISHED!!!',0AH,0DH,'THERE ARE ','$'
    finish3  DB ' numbers',0AH,0DH,'$'
    got      DB 5 DUP(0)
    huanhang DB 0AH,0DH,'$'
DATA ENDS

CODE SEGMENT
                      assume cs:CODE,ds:DATA,ss:_STACK
    START:
    beginofread:
                      mov    ax,DATA
                      mov    ds,ax
                      mov    dx,offset hello
                      mov    ah,09H
                      int    21H
                      mov    bx,0H
                      mov    di,offset got
                      mov    cx,0H
    readchar:
                      mov    ah,01H
                      mov    al,00h
                      int    21H
                      cmp    bx,0H
                      jne    notfirst
                      cmp    al,'Q'
                      je     exit
                      cmp    al,'q'
                      je     exit
                      cmp    al,'s'
                      je     next
    notfirst:
                      mov    bx,01H
```

```
                    call    legalcheck
                    cmp     bx,02H
                    je      beginofread
                    cmp     bx,04H
                    je      endofinput
                    jmp     loadinmemory
    loadinmemory:
                    mov     [di],al
                    inc     cx
                    inc     di
                    jmp     readchar
    endofinput:
                    mov     dx,0H
                    mov     di,offset got
    beginofhandle:
                    mov     bx,0H
                    mov     bl,[di]
                    sub     bx,30H
                    add     dx,bx
                    cmp     cx,1H
                    je      endofhandle
                    call    mulAHdxtodx
                    dec     cx
                    inc     di
                    jmp     beginofhandle
    next:
                    jmp     counterofnumber
    endofhandle:
                    call    binaryoutput
                    jmp     beginofread
    binaryoutput:
                    mov     bx,dx
                    mov     dx,0H
                    mov     cx,10H
    beginofoutputloop:
                    shl     bx,1
                    jnc     out0
                    mov     dl,'1'
                    jmp     outputdl
    exit:
                    mov     ah,4CH
                    int     21H
    out0:
                    mov     dl,'0'
    outputdl:
                    mov     ah,02H
                    int     21H
                    dec     cx
                    cmp     cx,0H
                    jne     beginofoutputloop
                    mov     dx,offset finish
                    mov     ah,09H
                    int     21H
                    ret
    legalcheck:
                    cmp     al,00H
```

```
                    je      endlegalnextline
                    cmp     al,30H
                    jb      endlegalfalse
                    cmp     al,39H
                    ja      endlegalfalse
    endlegaltrue:
                    mov     bx,03H
                    ret
    endlegalnextline:
                    mov     bx,04H
                    mov     dx,offset huanhang
                    mov     ah,09h
                    int     21h
                    ret
    endlegalfalse:
                    mov     dx,offset wrong
                    mov     ah,09H
                    int     21H
                    mov     bx,02H
                    ret
    mulAHdxtodx:
                    mov     bx,0H
                    mov     ax,0H
    loopofmul:
                    add     ax,dx
                    inc     bx
                    cmp     bx,0AH
                    jb      loopofmul
                    mov     dx,ax
                    ret
    counterofnumber:
                    mov     dx,offset hello2
                    mov     ah,09H
                    int     21H
                    mov     cx,0H
    beginofcount:
                    mov     ah,01H
                    mov     al,00h
                    int     21H
                    cmp     al,20H
                    je      endofcount
                    cmp     al,30H
                    jb      notnum
                    cmp     al,39H
                    ja      notnum
    isnum:
                    inc     cx
                    jmp     beginofcount
    notnum:
                    jmp     beginofcount
    endofcount:
                    add     cx,30H
                    mov     dx,offset finish2
                    mov     ah,09H
                    int     21H
                    mov     dx,0H
```

```
                    mov     dl,cl
                    mov     ah,02H
                    int     21H
                    mov     dx,offset finish3
                    mov     ah,09H
                    int     21H
                    jmp     beginofread
CODE ENDS
END START
```

五、心得

实验帮助理解了不同进制之间的转换原理，尤其是十进制到二进制的递归计算方法。编码和调试环节中，学会了设计输入校验和循环处理的逻辑。通过统计字符串中的数字字符，进一步熟悉了字符判断和计数的实现。实验结合实际运算需求，培养了进制转换的编程技巧，体会到循环程序设计的严密性和实际意义。

实验三：基本I口扩展实验

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.自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、代码

```
IO244 EQU 0230H
IO273 EQU 0230H
mystack segment stack
    db 100 DUP(0)
mystack ends

data segment
```

```
data ends

code segment
     assume cs:code,ds:data,ss:mystack
start:
     mov ax,data
     mov ds,ax

input:
     mov dx,IO244
     in ax,dx
     cmp ax,0FFFFH
     jz q1  ;从右向左依次点亮
     cmp ax,0
     jz q2  ;从左向右依次点亮

     mov dx,IO273
     not ax
     out dx,ax
     jmp input
   q1:
     mov ax,7FFFH
     mov dx,IO273
   r2l:
     call delay
     out dx,ax
     rol ax,1
     cmp ax,7FFFH
     jnz r2l
     jz input
   q2:
     mov ax,0FFFEH
     mov dx,IO273
   l2r:
     call delay
     out dx,ax
     ror ax,1
     cmp ax,0FFFEH
     jnz l2r
     jmp input

 delay proc near
     xor cx,cx
     delay1:loop delay1
     ret
     delay endp

code ends
     end start
```

五，心得

通过使用TTL芯片，掌握了I/O口的扩展方法和基本操作。利用LED灯的状态显示，直观理解了输入输出的对应关系。设计了多种控制模式，通过调试逐步优化了程序逻辑。实验加深了对硬件控制和数据交互的理解，为后续硬件设计和嵌入式开发打下了基础。

实验四 可编程并行接口实验

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

使二极管从中间往两边依次点亮。

三、实验原理

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、代码

```
COM_ADD EQU 0273H
PA_ADD  EQU 0270H
PB_ADD  EQU 0271H
PC_ADD  EQU 0272H

_STACK  SEGMENT STACK
        DW      100 DUP(?)
_STACK  ENDS

_DATA   SEGMENT WORD PUBLIC 'DATA'
LED_Data  DB 11100111B
          DB 11011011B
          DB 10111101B
          DB 01111110B
_DATA   ENDS

CODE    SEGMENT
START   PROC NEAR
        ASSUME CS:CODE,DS:_DATA,SS:_STACK
        MOV AX,_DATA
        MOV DS,AX
        NOP
        MOV DX,COM_ADD
        MOV AL,80H
        OUT DX,AL
        MOV DX,PA_ADD
        MOV AL,0FFH
        OUT DX,AL
        LEA BX,LED_Data
START1: MOV AL,0
```

```
        XLAT
        OUT DX,AL
        CALL DL100ms
        MOV CX,0
        MOV AL,1
        XLAT
        OUT DX,AL
        CALL DL100ms
        MOV AL,2
        XLAT
        OUT DX,AL
        CALL DL100ms
        MOV AL,3
        XLAT
        OUT DX,AL
        CALL DL100ms
        MOV CX,0
        MOV AL,4
        JMP START1

DL100ms PROC NEAR
        PUSH CX
        MOV CX,60000
DL100ms1:   LOOP DL100ms1
            POP CX
            RET
DL100ms     ENDP

START       ENDP
CODE        ENDS
            END START
```

五、心得

实验探索了8255芯片的功能和应用，重点学习了并行接口的工作方式及初始化编程。通过编程实现二极管从中间向两侧的点亮，增强了对并行接口原理的感性认知。实验理论结合实际操作，让我更加直观地理解了芯片接口的控制原理，拓宽了对嵌入式系统开发的认识。