

重 庆 大 学

学 生 实 验 报 告

实验课程名称 汇编语言程序设计

开课实验室 DS1502 机房

学 院 大数据与软件学院 年级 2023 级 专业 软件工程 01 班

学 生 姓 名 学 号

开 课 时 间 2022 至 2023 学年第 1 学期

总 成 绩	
教师签名	陈蜀宇

大数据与软件工程学院 制

《汇编语言程序设计》实验报告

开课实验室： DS1502 机房

2023 年 10 月 24 日

学院	软件学院	年级、专业、班	23 级软工 01 班	姓名		成绩	
课程名称	汇编语言程序设计		实验项目名称	实验四：循环、分支结构及子程序.doc		指导教师	陈蜀宇
教师评语	教师签名：陈蜀宇 2022 年 10 月 日						

一【实验目的】

1. 掌握循环、分支结构及子程序基本结构。
2. 熟练运用相关持续结构。

二【实验环境】

- Pc及dos或 Windows 操作系统；
- masm.exe, link.exe, debug.exe 或宏汇编集成环境。

(如果是 64 位机器需要安装 dosbox 将 64 位 win 虚拟为 32 位的环境,安装好 DOSBOX 后启动,用 mount 虚拟的空间 win 中真实的空间,例如 mount : f:\masm 就将本机磁盘上的 f:\masm 虚拟成 f:,这时进入 f:就相当于进入了 f:\masm ,且此时是虚拟成 32 位的环境)。

- dosbox.exe (64 位 Windows 操作系统需要)

Dosbox.exe 的使用方法

- 1.安装 。
- 2.运行 dosbos.exe 程序 。
- 3.在输入框 z:\>状态下输入:

```
z:\> mount c d:\masm
```

说明：“z:\>”这是提示符，“c”作为虚拟 c 盘，“d:\masm”是要虚拟的文件夹位置；简洁的讲，把要虚拟的文件夹位置换掉上面的 d:\masm 。
- 4.在刚才的提示符下输入 c:，这样就切换到虚拟的 c 盘，也就是你要的目录。
- 5.按照 dos 操作系统环境要求运行 c: 目录下的 masm.exe, link.exe, debug.exe 等软件。
如：

```
c:\> masm 123.asm  
c:\> LINK 123.obj  
c:\> debug 123.exe
```

三【实验要求】

1. 阅读数据传送指令、算术运算指令、逻辑指令、数据串传送程序和数据串传送指令 MOVS、STOS 及重复前缀 REP 的内容、串操作、逻辑指令、控制转移指令等；
2. 理解循环、分支结构及子程序相关知识。

四【预备知识】

1.汇编语言程序格式，指令，伪操作，表达式，循环、分支结构及子程序框架。

2.输入和显示字符。

```
DATAS SEGMENT
```

```
CHAR DW ? ; 定义char为db类型 但是不赋值
```

```
DATAS ENDS
```

```
STACKS SEGMENT
```

```
DB 256 DUP(?) ;申请 256 个字节空间
```

```
STACKS ENDS
```

```
CODES SEGMENT
```

```
ASSUME CS:CODES,DS:DATAS,SS:STACKS
```

```
START: MOV AX,DATAS
```

```
MOV DS,AX
```

```
;代码初始化
```

```
MOV AH,01H ; 输入一个字符保存在AL并回显到屏幕上
```

```
INT 21H
```

```
MOV [CHAR], AL
```

```
MOV DL,[CHAR] ; 显示一个字符，该字符在DL中
```

```
MOV AH,02H;
```

```
INT 21H
```

```
MOV AH,07H;
```

```
INT 21H
```

```
MOV DL,AL;
```

```
MOV AH,02H;
```

```
INT 21H;
```

```
MOV AH,4CH
```

```
INT 21H
```

```
CODES ENDS
```

```
END START
```

3. 字符串的显示输出。

;hello world这个程序是每们语言入门级新手的必经之路，下面就其汇编形式给出简短的源代码和粗略注释，关于21号中断的作用可自行网上查阅。

;作用：hello world! 字符串的输出

;寄存器关联：代码段code与代码段寄存器CS关联，

;数据段data与数据段寄存器DS关联。

assume CS:code,DS:data ; 含义：assume是伪指令，在扫描编译时不翻译

;data数据段定义。具体代码如下：

```
data segment
```

```
string db 'Hello world!','$' ;切记串结束符$
```

```
data ends ;代码段定义
```

```
code segment ;程序开始
```

```
start: mov ax,data ;将数据段段地址装入AX寄存器
```

```
mov ds,ax ;将数据段段地址通过通用寄存器AX装入DS
```

```
mov dx,offset string ;将串的段内地址装入DX
```

```

        mov ah,09h      ;调用DOS的09H号功能，传入参数DS:DX=串地址，'$'结束字符串
        int 21h
        mov ah,4ch      ;调用DOS的4CH号功能，带返回码结束，返回码存在于AL
        int 21h
code ends      ;代码段定义结束
end start      ;程序结束

```

五 【实验内容】

1. 编写一程序，要求比较两个字符串 STRING1 和 STRING2 所含字符是否相同，若相同则显示 'MATCH' ,若不相同则显示 'N'。字符串的内容在程序中指定。

源代码：

```

        data segment
string1 db 'Hello world!','$'
len1 equ $- string1-1
string2 db 'Hellq world!','$'////在这里设置字符串
len2 equ $- string2-1
symbol1 db 'match','$'
symbol2 db 'not match','$'

data ends

code segment
        assume cs:code,ds:data
        start:
mov ax,data
mov ds,ax
mov ax,len1
cmp ax,len2
jne different////如果长度不等，直接判断为不一致
mov cx,len1
search:////用于比较
        lea si,string1
        add si,cx
        mov al,byte ptr [si]
        lea si,string2
        add si,cx
        mov bl,byte ptr [si]
        cmp al,bl
        jne different
        pop cx
        cmp cx,0
        je same
        dec cx
        jmp search

same:////输出 match

```

```

mov dx,offset symbol1
mov ah,09h
int 21h
mov ah,4ch
int 21h

different:///输出 not match
mov dx,offset symbol2
mov ah,09h
int 21h
mov ah,4ch
int 21h

code ends
end start

```

运行截图：

当相同字符时：

data segment	076C:003A B409	MOV	AH,09
string1 db 'world!','\$'	076C:003C CD21	INT	21
len1 equ \$- string1-1	076C:003E B44C	MOV	AH,4C
string2 db 'world!','\$'	-u		
len2 equ \$- string2-1	076C:0040 CD21	INT	21
symbol1 db 'match!','\$'	076C:0042 50	PUSH	AX
symbol2 db 'not match!','\$'	076C:0043 8BC3	MOV	AX,BX
	076C:0045 8CC2	MOV	DX,ES
	076C:0047 050C00	ADD	AX,0000
data ends	076C:004A 52	PUSH	DX
	076C:004B 50	PUSH	AX
code segment	076C:004C E8C148	CALL	4910
assume cs:code,ds:data	076C:004F 83C404	ADD	SP,+04
start:	076C:0052 50	PUSH	AX
mov ax,data	076C:0053 8D86FAFE	LEA	AX,[BP+00000006]
mov ds,ax	076C:0057 50	PUSH	AX
mov ax,len1	076C:0058 E81773	CALL	7372
cmp ax,len2	076C:005B 83C406	ADD	SP,+06
jne different	076C:005E 8BB6FAFE	MOV	SI,[BP+00000006]
mov cx,len1	-g 0042		
search:	match		
lea si,string1	Program terminated normally		
add si,cx	-		
mov al,byte ptr [si]			
lea si,string2			
add si,cx			

当为不同长度时：

```
data segment
    string1 db 'hello world!','$'
    len1 equ $- string1-1
    string2 db 'world!','$'
    len2 equ $- string2-1
    symbol1 db 'match!','$'
    symbol2 db 'not match!','$'
```

```
data ends
```

```
code segment
    assume cs:code,ds:data
start:
    mov ax,data
    mov ds,ax
    mov ax,len1
    mov ax,len2
    cmp ax,len2
    jne different
    mov cx,len1
search:
    lea si,string1
    add si,cx
    mov al,byte ptr [si]
    lea si,string2
    add si,cx
```

```
Object filename LAB42.OBJ:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
```

```
51602 + 464942 Bytes symbol space free
```

```
0 Warning Errors
0 Severe Errors
```

```
C:\>link lab42
```

```
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
```

```
Run File [LAB42.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
```

```
C:\>debug lab42.exe
-g 0042
not match
Program terminated normally
```

的顺序排列并显示出来

当相同长度但是有些字符不同时：

```
data segment
    string1 db 'hello world!','$'
    len1 equ $- string1-1
    string2 db 'heeee world!','$'
    len2 equ $- string2-1
    symbol1 db 'match!','$'
    symbol2 db 'not match!','$'
```

```
data ends
```

```
code segment
    assume cs:code,ds:data
start:
    mov ax,data
    mov ds,ax
    mov ax,len1
    mov ax,len2
    cmp ax,len2
    jne different
    mov cx,len1
search:
    lea si,string1
    add si,cx
    mov al,byte ptr [si]
    lea si,string2
    add si,cx
```

```
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
```

```
51602 + 464942 Bytes symbol space free
```

```
0 Warning Errors
0 Severe Errors
```

```
C:\>link lab42
```

```
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
```

```
Run File [LAB42.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
```

```
C:\>debug lab42.exe
-g 0042
not match
Program terminated normally
```

的顺序排列并显

要求：

注意在用 equ 时需要紧跟着定义的变量，避免后续定义冲散了标记使得返回的长度出现问题

2. 在缓冲区中输入 20 字符，然后按 ASCII 值从大到小的顺序排列并显示出来。

要求：

- (1) 显示时每个字符间隔一个空格；
- (2) 换行后再将其中的数字按序显示出来。

源代码：

```
data segment
    buffer db 20 dup(?)
data ends
```

```

stacks segment
    db 21 dup(?)
stacks ends////定义20个缓冲位置

code segment
    assume cs:code,ds:data,ss:stacks
    start:
    mov ax,data
    mov ds,ax
    mov cx,20
    mov si,0
input:
    lea dx,buffer[si]
    mov ah,01h
    int 21h
    mov buffer[si],al
    inc si
    loop input
    mov cx,20
    mov si,0////输入到定义中的缓冲区域

    mov dl,10
    mov ah,02h
    int 21h////换行

    mov cx,19 ////冒泡排序
    outer_loop:
        mov si,0
        mov dx,cx
        inner_loop:
            mov al,buffer[si]
            cmp al,buffer[si + 1]
            jle no_swap
            xchg al,buffer[si+1]
            mov buffer[si],al
        no_swap:
            inc si
            dec dx
            jnz inner_loop
        loop outer_loop

    mov cx,20
    mov si,0

```

display:///显示全部

```
mov dl,buffer[si]
mov ah,02h
int 21h
mov dl,32
mov ah,02h
int 21h
inc si
loop display
mov cx,20
mov si,0
```

```
mov dl,10
mov ah,02h
int 21h///换行
```

display_number:///筛选数字字符并显示

```
mov al,buffer[si]
cmp al,48
jl not_digit
cmp al,57
jg not_digit
mov dl,al
mov ah,02h
int 21h
mov dl,32
mov ah,02h
int 21h
jmp next
not_digit:
next:
inc si
loop display_number
mov ah,4ch
int 21h
code ends
end start
运行结果:
```



```

Libraries [LIB]:
LINK : warning L4021: no stack segment

C:\>lab43.exe
asdfghb231764vgdaajs
1 2 3 4 6 7 a a a b d d f g g h j s s v
1 2 3 4 6 7
C:\>

```

3. 定义 10 个字节的键盘缓冲区，然后键盘输入字符填满该缓冲区，做如下工作：

- (1) 分别将输入键盘缓冲区的字符按数字，小写字母，大写字母，其他字符进行计数。
- (2) 分别将这些计数值显示出来。

源代码：

data segment

```

    buffer db 10 dup(?)
    upper_num db 0
    lower_num db 0
    other_num db 0
    digit_num db 0
data ends

```

code segment

```

    assume cs:code,ds:data
    start:
    mov upper_num,0
    mov lower_num,0
    mov other_num,0
    mov digit_num,0////初始化为零
    mov ax,data
    mov ds,ax
    mov cx,10
    mov si,0

input_count:
    cmp cx,0
    je display
    lea dx,buffer[si]
    mov ah,01h
    int 21h
    mov buffer[si],al////通过01h输入

    cmp al,48
    jl not_digit
    cmp al,57
    jg not_digit
    inc digit_num

```

```
jmp next///判断是否为数字
```

```
not_digit:  
cmp al,65  
jl not_upper  
cmp al,90  
jg not_upper  
inc upper_num  
jmp next///判断是否为大写字母
```

```
not_upper:  
cmp al,97  
jl not_lower  
cmp al,122  
jg not_lower  
inc lower_num  
jmp next///判断是否为小写字母  
not_lower:  
inc other_num///都不是则为其  
next:  
inc si  
dec cx  
jmp input_count
```

```
display:  
mov dl,10  
mov ah,02h  
int 21h  
add digit_num,48///加上48使得变成数字  
mov dl,digit_num  
mov ah,02h  
int 21h  
mov dl,32  
mov ah,02h  
int 21h
```

```
add lower_num,48  
mov dl,lower_num  
mov ah,02h  
int 21h  
mov dl,32  
mov ah,02h  
int 21h
```

```
add upper_num,48  
mov dl,upper_num
```

```

mov ah,02h
int 21h
mov dl,32
mov ah,02h
int 21h

add other_num,48
mov dl,other_num
mov ah,02h
int 21h
mov dl,32
mov ah,02h
int 21h
mov ah,4ch
    int 21h
code ends
end start
运行:

```

The screenshot shows a DOS command prompt window. On the left, assembly code is being entered line by line. On the right, the output of the linker and the execution of the program is shown. The linker output includes the version of the Microsoft Overlay Linker and a warning about a missing stack segment. The program execution shows the output of the assembly code, which is a series of numbers: 2 2 3 3 _.

```

C:\>LINK lab44
Microsoft (R) Overlay Linker  Vers
Copyright (C) Microsoft Corp 1983-

Run File [LAB44.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack seg

C:\>lab44
asDFS12*&*
2 2 3 3 _

```

需要注意的是在初始化计数是初始化为0，后续要加上48变成数字，所以不能计数超过10，或者在加上48后判断是否为58，是的话就直接输出10

五【实验步骤】

六【实验结果及分析】