合肥工业大学
计算机与信息学院
实验报告

课　　程：汇编语言程序设计

专业班级：计算机科学与技术 24-4 班

学　　号：2024210858

姓　　名：傅家琪

实验一　基本汇编程序设计

一. 实验目的

1、熟悉在 PC 机上建立、汇编、连接、调试和运行 8086/8088 汇编语言程序的过程；

2、掌握基本汇编语言程序设计方法。

3、熟悉 DOSBOX 下运行 dos 程序方法；

4、熟悉 debug 调式程序的基本方法；

二. 实验内容

1、利用 DEBUG 程序中的“E”命令，将两个多字节数“12345678H”和“FEDCBA98H”分别送入起始地址为 DS:0200H 和 DS:0204H 两个单元中。

注意：阅读本书后的附录 1，熟悉掌握 DEBUG 调试程序中的 A、D、E、G、Q、R、T、U 命令的书写格式及功能；复习寻址方式和相关的数据传送指令及算术运算指令。

2、分别用直接寻址方式和寄存器间接寻址方式编写程序段，实现将 DS:0200H 单元和 DS:0204H 单元中的数据相加，并将运算结果存放在 DS:0208H 单元中。

注意：本实验的内容均在 DEBUG 下完成，实现数据的装入、修改、显示；汇编语言程序段的编辑、汇编和反汇编；程序的运行和结果检查。

3、数据段中 X、Y、Z、V 均为字变量，存放的是 16 位带符号数。编写汇编语言程序计算表达式值（V－（X*Y+Z-720））/X。要求：

（1）使用 MASM 和 LINK 生成可执行程序后，使用 Debug 装入内存；

（2）使用 U、D 命令查看代码段和数据段的内容，特别注意变量在代码段中的形式；使用 G 命令执行代码，并使用 D 命令查看结果；

（3）用 E 命令修改各个变量的内容，再次用 G 命令执行，并查看结果。

4、求一个班 50 名学生成绩的平均值、最大值和最小值,并将结果显示出来。

三. 实验过程和程序

1.

```
-E DS:200
073F:0200  78.     56.     56.34   78.12   00.98   00.BA   00.DC   00.FE
073F:0208  00.
```

2.

```
-A
073F:011C MOV AX,[0200]
073F:011F MOV DX,[0202]
073F:0123 ADD AX,[0204]
073F:0127 ADC DX,[0206]
073F:012B MOV [0208],AX
073F:012E MOV [020A],DX
073F:0132 INT 3
073F:0133
-G=011C
```

直接寻址

```
-A
073F:0133 MOV BX,0200
073F:0136 MOV AX,[BX]
073F:0138 MOV DX,[BX+2]
073F:013B ADD AX,[BX+4]
073F:013E ADC DX,[BX+6]
073F:0141 MOV [BX+8],AX
073F:0144 MOV [BX+A],DX
073F:0147 INT 3
073F:0148
-G=0133
```

寄存器间接寻址

3.

CALC.ASM

```
.model small
.stack 100h

.data
    X   dw  5       ; X = 5
    Y   dw  10      ; Y = 10
    Z   dw  500     ; Z = 500
    V   dw  1000    ; V = 1000

.code
main:
    mov ax, @data
    mov ds, ax
    mov ax, X
    imul Y
    add ax, Z
    ADC DX, 0
    sub ax, 720
    SBB DX, 0
    MOV BX,AX
    MOV CX,DX   ;CX.BX=X*Y+Z-720
    MOV AX,V
    CWD         ;DX.AX=V
    SUB AX,BX
    SBB DX,CX   ;DX.AX=V-X*Y+Z-720
    IDIV X      ;AX存商，DX存余数


    mov ah, 4Ch
    int 21h

end main

```

先写了代码，然后把这个文件、MASM、LINK、DEBUG 文件放在同一个文件夹下，重新用 dosbox 打开，运行了这个 CALC.ASM

```
C:\>MASM CALC.ASM,,,;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.


  50676 + 465868 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>LINK CALC.OBJ,,,;

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

LINK : warning L4021: no stack segment

C:\>DEBUG CALC.EXE
-t
```

```
-u
076A:0000 B86C07        MOV     AX,076C
076A:0003 8ED8          MOV     DS,AX
076A:0005 A10E00        MOV     AX,[000E]
076A:0008 F72E1000      IMUL    WORD PTR [0010]
076A:000C 03061200      ADD     AX,[0012]
076A:0010 83D200        ADC     DX,+00
076A:0013 2DD002        SUB     AX,02D0
076A:0016 83DA00        SBB     DX,+00
076A:0019 8BD8          MOV     BX,AX
076A:001B 8BCA          MOV     CX,DX
076A:001D A11400        MOV     AX,[0014]
```

U 指令

```
-d
076A:0000  B8 6C 07 8E D8 A1 0E 00-F7 2E 10 00 03 06 12 00
076A:0010  83 D2 00 2D D0 02 83 DA-00 8B D8 8B CA A1 14 00
076A:0020  99 2B C3 1B D1 F7 3E 0E-00 B4 4C CD 21 00 05 00
076A:0030  0A 00 F4 01 E8 03 00 00-00 00 00 00 00 00 00 00
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
-a
```

D 指令
数据从红框开始

```
-e ds:002e
075A:002E  4C.

-r ds
DS 075A
:076A
-E DS:002E
076A:002E  05.08

-E DS:002F
076A:002F  00.
076A:0030  0A.14   00.     F4.F4   01.01   E8.F4   03.01   00.54   00.01
076A:0038  00.
```

用 E 指令修改了 X，Y，Z，V 的值
再次运行指令

4.

SCORE.ASM

```
DATA SEGMENT
    SCORES DB 85, 92, 78, 64, 88, 95, 72, 81, 90, 67
           DB 83, 79, 91, 86, 74, 89, 76, 82, 94, 68
           DB 87, 93, 77, 84, 71, 96, 69, 80, 75, 98
           DB 65, 73, 97, 70, 99, 66, 99, 63, 89, 82
           DB 78, 91, 84, 76, 87, 92, 79, 84, 90, 73   ; 50个成绩

    COUNT DB 50          ; 学生人数
    MAX DB ?             ; 最大值
    MIN DB ?             ; 最小值
    AVERAGE DB ?         ; 平均值

    MSG_MAX DB 'Max: $'
    MSG_MIN DB 'Min: $'
    MSG_AVG DB 'Average: $'
    NEWLINE DB 0DH, 0AH, '$'  ; 换行
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    ; 初始化
    MOV SI, OFFSET SCORES  ; SI指向成绩数组
    MOV CL, COUNT          ; 循环计数器
    MOV AL, [SI]           ; 初始化最大值和最小值为第一个成绩
    MOV MAX, AL
    MOV MIN, AL
    XOR AX, AX             ; 清空AX用于累加和
    XOR BX, BX             ; 清空BX

CALC_LOOP:
    MOV BL, [SI]           ; 读取当前成绩

    ; 累加总分
    ADD AL, BL
    ADC AH, 0              ; 处理进位

    ; 比较最大值
    CMP BL, MAX
    JLE CHECK_MIN
    MOV MAX, BL

CHECK_MIN:
    ; 比较最小值
    CMP BL, MIN
    JGE NEXT_SCORE
    MOV MIN, BL

NEXT_SCORE:
    INC SI                 ; 指向下一个成绩
    DEC CL                 ; 计数器减1
    JNZ CALC_LOOP          ; 继续循环

    ; 计算平均值
    MOV BL, COUNT
    DIV BL                 ; AX / BL，商在AL，余数在AH
    MOV AVERAGE, AL

    ; 显示结果
    CALL DISPLAY_RESULTS

    ; 退出程序
    MOV AH, 4CH
    INT 21H

; 显示结果的子程序
DISPLAY_RESULTS PROC
    ; 显示最大值
    MOV AH, 09H
    LEA DX, MSG_MAX
    INT 21H
```

代码部分

```
            MOV AL, MAX
            CALL DISPLAY_NUMBER
            CALL NEW_LINE

            ; 显示最小值
            MOV AH, 09H
            LEA DX, MSG_MIN
            INT 21H

            MOV AL, MIN
            CALL DISPLAY_NUMBER
            CALL NEW_LINE

            ; 显示平均值
            MOV AH, 09H
            LEA DX, MSG_AVG
            INT 21H

            MOV AL, AVERAGE
            CALL DISPLAY_NUMBER
            CALL NEW_LINE

            RET
        DISPLAY_RESULTS ENDP

        ; 显示两位数字的子程序
        DISPLAY_NUMBER PROC
            MOV AH, 0
            MOV BL, 10
            DIV BL          ; AL = 十位, AH = 个位

            MOV BL, AH      ; 保存个位

            ; 显示十位
            MOV DL, AL
            ADD DL, '0'
            MOV AH, 02H
            INT 21H

            ; 显示个位
            MOV DL, BL
            ADD DL, '0'
            MOV AH, 02H
            INT 21H

            RET
        DISPLAY_NUMBER ENDP

        ; 换行子程序
        NEW_LINE PROC
            MOV AH, 09H
            LEA DX, NEWLINE
            INT 21H
            RET
        NEW_LINE ENDP

        CODE ENDS
        END START
```

```
LINK : warning L4021: no stack segment

C:\>DEBUG SCORE.EXE
-u
076F:0000 B86A07        MOV     AX,076A
076F:0003 8ED8          MOV     DS,AX
076F:0005 BE0000        MOV     SI,0000
076F:0008 8A0E3200      MOV     CL,[0032]
076F:000C 8A04          MOV     AL,[SI]
076F:000E A23300        MOV     [0033],AL
076F:0011 A23400        MOV     [0034],AL
076F:0014 33C0          XOR     AX,AX
076F:0016 33DB          XOR     BX,BX
076F:0018 8A1C          MOV     BL,[SI]
076F:001A 02C3          ADD     AL,BL
076F:001C 80D400        ADC     AH,00
076F:001F 3A1E3300      CMP     BL,[0033]
-q
```

四. 实验结果（包括必要的截图）

1.

```
-D 200
073F:0200  78 56 34 12 98 BA DC FE-00 00 00 00 00 00 00 00
073F:0210  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
073F:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
073F:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
073F:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
073F:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
073F:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
073F:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

2.

直接寻址

```
-G=011C

AX=1110  BX=0000  CX=0000  DX=1111  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0132   NV UP EI PL NZ AC PE CY
073F:0132 CC            INT     3
```

寄存器间接寻址

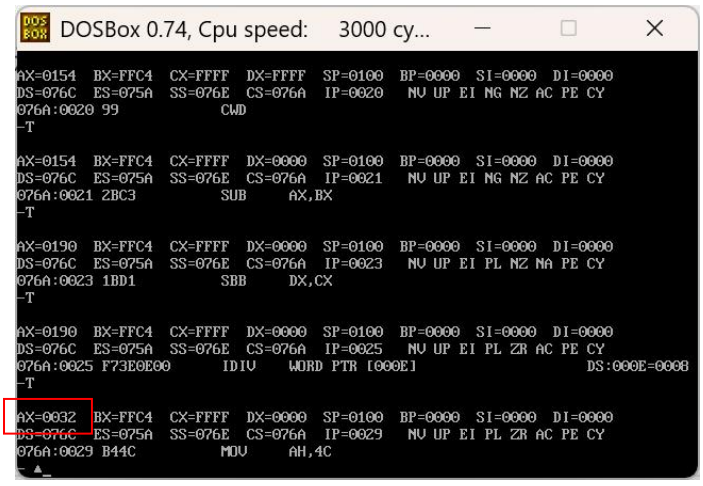
```
-G=0133

AX=1110  BX=0200  CX=0000  DX=1111  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0147   NV UP EI PL NZ AC PE CY
073F:0147 CC            INT     3
```

3.

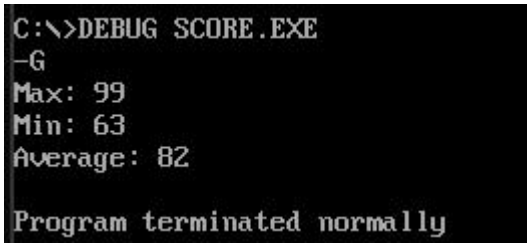
初始值下的

```
AX=00EA  BX=FF56  CX=FFFF  DX=0000  SP=0100  BP=0000  SI=0000  DI=0000
DS=076C  ES=075A  SS=076E  CS=076A  IP=0029   NV UP EI PL ZR AC PE CY
076A:0029 B44C          MOV     AH,4C
-a
```

修改完 XYZV 的值后运行



4.



五. 实验体会

打开 dosbox 的 debug 功能就已经费了好大功夫，发现在 mount 后面要跟文件路径，而后学习了一下 A,E,D,G 等指令。

第一个任务是赋值，赋值有单个单个赋值和一长串赋值，但要注意题目给的和输进去的应该要低字节对低位；

第二个任务是分别使用直接寻址和寄存器间接寻址来实现加法，两种方法都需要 ADD 和 ADC，区别是直接寻址是直接找地址，而寄存器间接寻址是把首地址先赋给寄存器 BX，然后用 BX+2,+4……来表示数据；

第三和第四个任务都是先写好 ASM 文件，再用 MASM 和 LINK 打开，用 debug 功能运行，代码老师上课有讲过类似的，第四个考的是循环结构和子程序，控制循环次数，先把最大最小值都认为是第一个数据，然后再通过指针的移动，不断让后面的数据与最大最小值比较、更新。

第三个任务中，我发现直接 G 指令运行的话，AX，DX 寄存器没有变化，要按 T 才会逐步变化，另外在修改 XYZV 的值的时候一直不对！费了好久时间才发现是传入数据的数据段的地址跟 R 指令下的 DS 不一样，要修改地址让 DS 一样，才能成功传数据进去，最后成功了。

实验二　高级汇编程序设计

一. 实验目的

1、 掌握中断服务子程序的编写。
2、 汇编语言与高级语言的混编。

二. 实验内容

1、BL 中的只有一位为 1。编写程序测试并输出提示信息“The X Bit is 1”，要求：

（1）使用地址表实现分支功能，在分支中输出提示信息；

（2）程序直接执行，不得再使用 debug 相关命令执行。

2、编写一个子程序计算 z=f(x,y)=x*y+x-y（x,y,z 有符号数：字），要求：

（1）输入参数通过堆栈传送，输出参数通过 AX 传送（假设结果可以用 AX 表示得下）；

（2）结果需要按照 10 进制的格式输出到屏幕上。

3、挂接 1CH 中断，正计时 90 秒后退出。要求屏幕显示 0-89 的秒数。

4、从键盘读入一个字符串，以 Enter 结束，字符串不超过 50 个字符，并打印该字符串(查找中间是否有‘asm’子串。如果有，输出‘Yes’；否则，输出‘No’)。

三. 实验过程和程序

1.

```
ex1.asm*
1   data segment
2       msg0 db 'The 0 Bit is 1$'
3       msg1 db 'The 1 Bit is 1$'
4       msg2 db 'The 2 Bit is 1$'
5       msg3 db 'The 3 Bit is 1$'
6       msg4 db 'The 4 Bit is 1$'
7       msg5 db 'The 5 Bit is 1$'
8       msg6 db 'The 6 Bit is 1$'
9       msg7 db 'The 7 Bit is 1$'
10      jmp_table dw offset m0, offset m1, offset m2, offset m3
11                dw offset m4, offset m5, offset m6, offset m7
12  data ends
13
14  code segment
15  assume cs:code, ds:data
16  start:
17      mov ax, data
18      mov ds, ax
19
20      mov bl, 00000100b   ;
21
22      mov bh, 0
23      mov cx, 8
24  find_bit:
25      shr bl, 1
26      jc found
27      inc bh
28      loop find_bit
29      jmp exit
30
31  found:
32      mov al, bh
33      mov ah, 0
34      shl ax, 1           ; 乘2（因为表项是字）
35      mov si, ax
36      jmp jmp_table[si]
38      m0: mov dx, offset msg0
39          jmp print
40      m1: mov dx, offset msg1
41          jmp print
42      m2: mov dx, offset msg2
43          jmp print
44      m3: mov dx, offset msg3
45          jmp print
46      m4: mov dx, offset msg4
47          jmp print
48      m5: mov dx, offset msg5
49          jmp print
50      m6: mov dx, offset msg6
51          jmp print
52      m7: mov dx, offset msg7
53
54      print:
55          mov ah, 9
56          int 21h
57
58      exit:
59          mov ah, 4ch
60          int 21h
61
62      code ends
63      end start
```

这里可以设置要检测哪一位，我这里是第二位，所以后面会输出 The 2Bit is 1

2.

ex2.asm

```
data segment
    x dw 5
    y dw 3
    buffer db 6 dup('$')
data ends

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

    push x
    push y
    call func
    add sp, 4

    ; AX 中为结果，输出十进制
    call print_decimal

    mov ah, 4ch
    int 21h

func proc
    push bp
    mov bp, sp
    mov ax, [bp+6]  ; x
    mov bx, [bp+4]  ; y
    imul bx         ; dx:ax = x*y
    add ax, [bp+6]  ; +x
    sub ax, bx      ; -y
    pop bp
    ret
func endp

print_decimal proc
    ; 输入 AX = 有符号数
    mov bx, 10
    mov cx, 0
    test ax, ax
    jns positive
    neg ax
    push ax
    mov dl, '-'
    mov ah, 2
    int 21h
    pop ax
positive:
    xor dx, dx
    div bx
    push dx
    inc cx
    cmp ax, 0
    jnz positive
print_loop:
    pop dx
    add dl, '0'
    mov ah, 2
    int 21h
    loop print_loop
    ret
print_decimal endp

code ends
end start
```

3.

ex3.asm

```
assume cs:code,ds:data
data segment
oldisr dw ?,?
timer db 0
count db 0
new db 0dh,0ah,"$"
data ends
code segment
start:mov ax,data
    mov ds,ax
    mov ax,0
    mov es,ax
    mov ax,es:[1ch*4]
    mov oldisr[0],ax
    mov ax,es:[1ch*4+2]
    mov oldisr[2],ax     ;保存原中断
    mov word ptr es:[1ch*4],offset isr
    mov word ptr es:[1ch*4+2],seg isr      ;设置新中断
again:cmp timer, 89
    ja exit
    jmp again
exit:mov ax,oldisr[0]
    mov es:[1ch*4],ax
    mov ax,oldisr[2]
    mov es:[1ch*4+2],ax
    mov ah,4ch
    int 21h
isr proc far
    push ax
    push bx
    push cx
    push dx
    sti
    inc count
    cmp count,1000/55
    jb s
    inc timer
    mov count,0
    mov al,timer
    mov ah,0
    mov bh,10
    div bh
    mov dh,ah
    mov dl,al
    add dl,30h
    mov ah,2
    int 21h
    mov dl,dh
    add dl,30h
    mov ah,2
    int 21h
    lea dx,new
    mov ah,9
    int 21h
s:pushf
    call dword ptr oldisr
    pop dx
    pop cx
    pop bx
    pop ax
    iret

isr endp
code ends
end start
```

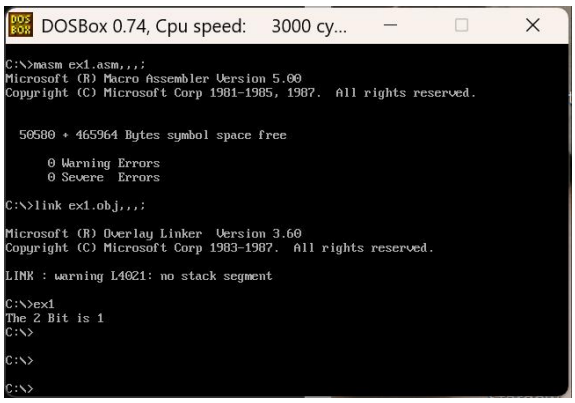
4.

ex4.asm

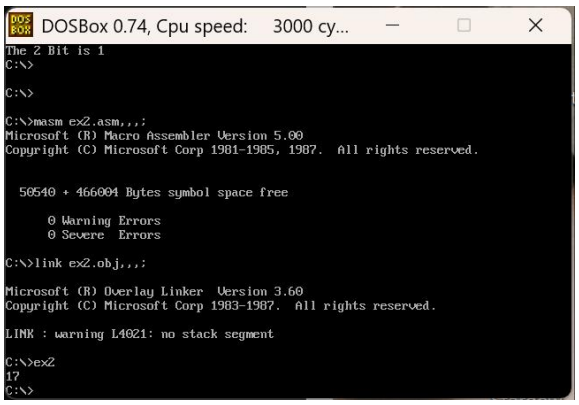
```
data segment
    prompt db 'Input string: $'
    yes_msg db 0dh, 0ah, 'Yes$'
    no_msg db 0dh, 0ah, 'No$'
    buffer db 50, 0, 50 dup('$')
data ends

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

    ; 显示提示
    mov dx, offset prompt
    mov ah, 9
    int 21h

    ; 读字符串
    mov dx, offset buffer
    mov ah, 0ah
    int 21h

    ; 查找子串
    mov si, offset buffer+2
search:
    mov al, [si]
    cmp al, 0dh          ; 遇到回车结束
    je not_found
    cmp al, 'a'
    jne next_char
    mov al, [si+1]
    cmp al, 's'
    jne next_char
    mov al, [si+2]
    cmp al, 'm'
    je found
next_char:
    inc si
    jmp search

found:
    mov dx, offset yes_msg
    jmp output

not_found:
    mov dx, offset no_msg

output:
    mov ah, 9
    int 21h

    ; 再输出一个换行使光标到下一行
    mov dl, 0dh
    mov ah, 2
    int 21h
    mov dl, 0ah
    mov ah, 2
    int 21h

    mov ah, 4ch
    int 21h

code ends
end start
```

四、实验结果（包括必要的截图）

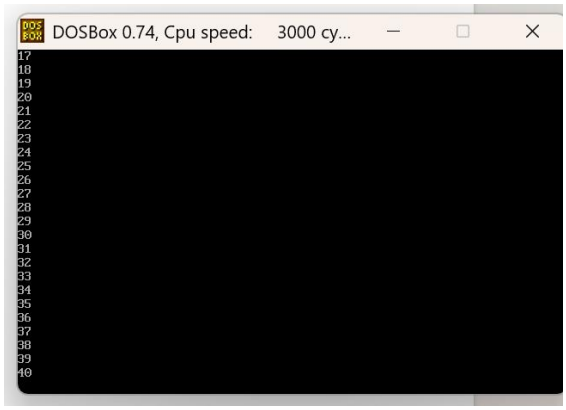
1.



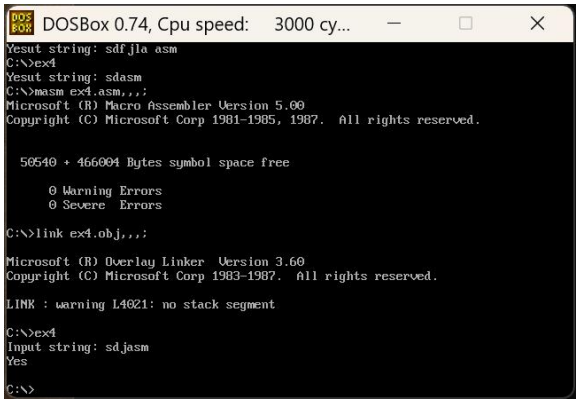
2.



3.



4.



五、实验体会

第一个任务把 bh 作为位数计数器，cx 作为循环次数，通过移位指令 shr，将 bl 中存的数据一位一位读取，判断是不是为 1，如果是 1，就跳转，如果是 0，就 bh+1，如果没找到就退出循环，如果找到了就把 bh 的值赋给 al，ax 要乘 2，因为跳转表里面每个地址占两个字节，然后就打印就可以了。

第二个任务首先是计算表达式的值，我这里设置的是 X=5，Y=3，最后答案是 17，使用堆栈保存数据，sp 始终指向栈底，bp 用来寻址，计算完结果之后要判断结果是正是负，通过最高位符号位来判断，然后就是转换成 10 进制的过程，不断除 10 求余数，最后要倒序输出。这里采用栈所以先进后出就自动倒序了。

第三个任务是跟中断有关，整个流程就是存储原中断，存新中断，设置新中断程序，调用新中断，恢复原中断的过程，注意这个返回要用 iret，而且要记住保护寄存器，一开始要给 DS 指位置。

第四个任务主要其实就是靠 SI 这个指针寄存器，不断的移动，找到 A 了再找 S，找到 S 了再找 M，最后记得回车换行再输出就可以了，同样这要调用 dos 的中断子程序 int21h，AH=9，DS:DX 指向以'$'结尾的字符串。

表. 实验成绩评定表

	评价内容		权重	得分
验收	实验原理是否理解；程序能否运行；实验结果是否正确；任务是否全部完成。		0.5	
实验报告	1	报告格式是否规范，语言使用是否规范，行文是否流畅，是否图文并茂；	0.2	
	2	实验原理、实验步骤描述是否正确、详实； 程序流程图是否规范，代码实现是否正确； 实验数据记录是否完整，实验结果是否正确； 实验结果的分析、对比是否充分；	0.2	
	3	实验体会是否正确，是否提出了自己独到见解。	0.1	
合计				
指导教师（签章）：_____ 202_年___月___日				