# 《微机原理与接口技术》实验报告

姓 名:	
学 号:	
专业班级:	
实验名称:	第一次实验
实验日期:	2023年10月17日

## 备注:

- (1)请将报告电子版发到邮箱 MrSuInterfaceWork@163.com, 文件名: 姓名-学号-班级-微机原理-第 X 次实验. docx。
- (2) 提交的内容: 文档,实验源代码(有几个任务就提交几个源代码)
  - (3) 邮件的主题和文件名同名。
  - (4) 文档排版统一为小四仿宋, 行间距离 1.5 倍行距。
  - (5) 提交日期: 下一次实验之前

# 一、实验目的

- 1)熟悉汇编程序开发(和调试)环境;
- 2)熟悉汇编程序基本结构和分段概念;
- 3)熟悉8086指令系统常见指令;
- 4)熟悉8086寻址方式和应用;

# 二、实验内容

- 1)实验1.1 寄存器读写(和调试)实验
- 2)实验1.2 一维数值数组处理实验
- 3)实验1.3 字符串处理实验
- 4)实验1.4 二维数组处理实验
- 5)实验1.5 位处理实验

# 三、实验过程和结果

# 3.1 寄存器读写(和调试)实验

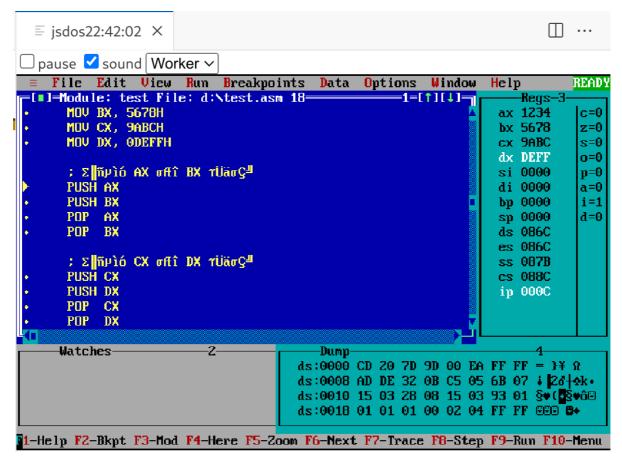
#### 3.1.1 源代码

```
1 DATA SEGMENT
2
3
   DATA ENDS
   STACK SEGMENT
6
           DW 128 DUP(0)
7
   STACK ENDS
8
9
    CODE SEGMENT
10
            ASSUME CS:CODE, DS:DATA, SS:STACK
11
       START:
12
             MOV
                  AX, 1234H
13
             MOV
                   BX, 5678H
14
                    CX, 9ABCH
             MOV
                   DX, ODEFFH
15
             MOV
16
17
        ; 交换 AX 和 BX 的值
             PUSH
18
                   AX
19
             PUSH
20
             POP
                    AX
21
             POP
                    BX
22
23
        ; 交换 CX 和 DX 的值
24
             PUSH
                   CX
25
             PUSH
                    DX
26
             POP
                    CX
27
             POP
                    DX
28
        ; 交换 AX 的高 8 位 (AH) 和 DX 的低 8 位 (DL)
29
30
             PUSH AX
```

```
31
                 POP
                         \mathsf{CX}
32
                 MOV
                         AL, AH
                         AH, DL
33
                 MOV
34
                 MOV
                         DL, AL
35
                 MOV
                         AX, 4C00H
36
                 INT
                         21H
37
     CODE ENDS
38
     END START
39
```

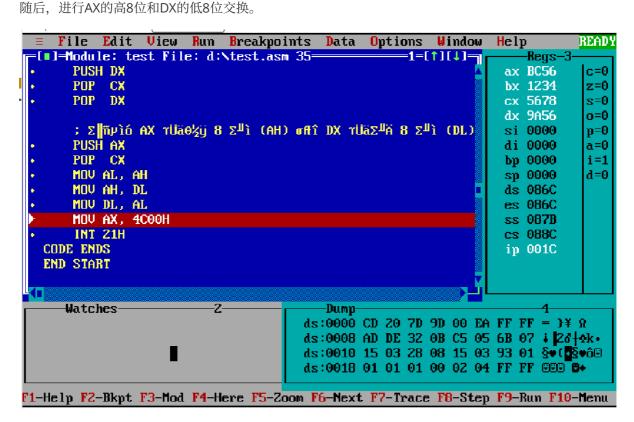
#### 3.1.2 观察到的现象

运行程序前:



运行程序后: 首先交换AX和BX的值, CX和DX的值, 在TD右侧的窗口可以看到具体的值:

```
File Edit View Run Breakpoints Data Options Window
                                                                              READY
  ■l=Module: test File: d:Ntest.asm 30=
                                                                      -Regs-3
       PUSH DX
                                                                               c=0
                                                                    ax 5678
       POP CX
                                                                    bx 1234
                                                                               z=0
       POP DX
                                                                   cx DEFF
                                                                               s=0
                                                                   dx 9ABC
                                                                               o=0
       ; Σ<mark>|</mark>ñμìό ΑΧ τὐäθዿÿ 8 Σμὶ (ΑΗ) σflî DX τὐäΣμὰ 8 Σμὶ (DL)
                                                                   si 0000
                                                                               p=0
       PUSH AX
                                                                   di 0000
                                                                               a=0
       POP CX
                                                                    bp 0000
                                                                               i=1
      MOV AL, AH
                                                                    sp 0000
                                                                               d=0
      MOV AH, DL
                                                                    ds 086C
      MOV DL, AL
                                                                    es 086C
      MOV AX, 4COOH
                                                                    ss 087B
       INT 21H
                                                                    cs 0880
  CODE ENDS
                                                                    ip 0014
  END START
 4:
     Watches
                           -2-
                                          -Dump
                                       ds:0000 CD 20 7D 9D 00 EA FF FF = Y
                                       ds:0008 AD DE 32 0B C5 05 6B 07 4 28+0k.
                                       ds:0010 15 03 28 08 15 03 93 01 §♥(•§♥ô©
                                       ds:0018 01 01 01 00 02 04 FF FF @@@ B+
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```



### 3.2 一维数值数组处理实验

#### 3.2.1 源代码

```
DATA SEGMENT

NUMBERS DW 0001H, 0002H, 0003H, 0004H, 0005H, 0006H, 0007H, 0008H, 0009H, 000AH, 000BH, 000CH, 000DH, 000FH, 0010H

SUM DW 0FFFFH

DATA ENDS

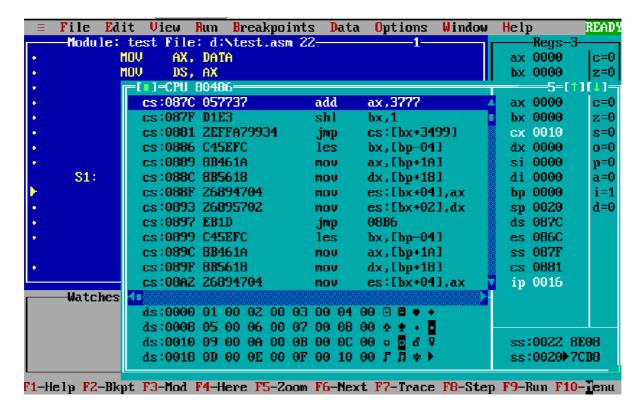
STACK SEGMENT

DW 16 DUP(0)
```

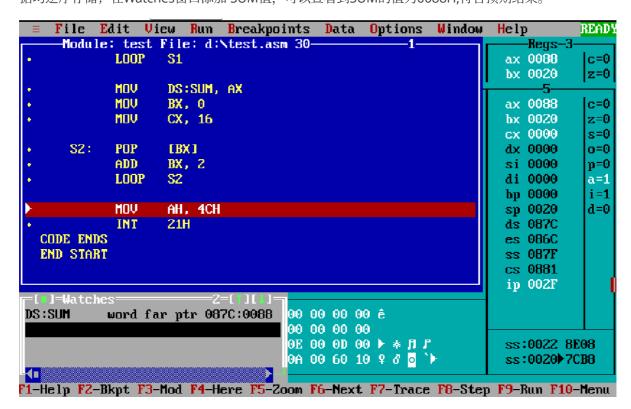
```
8 STACK ENDS
  9
 10 CODE SEGMENT
 11
                ASSUME CS:CODE, DS:DATA, SS:STACK
 12
           START:
 13
                 MOV
                        AX, DATA
 14
                        DS, AX
                 MOV
 15
                       AX, STACK
                 MOV
 16
                 MOV
                        SS, AX
 17
                       SP, 20H
                 MOV
                       AX, 0
 18
                 MOV
                        BX, 0
 19
                 MOV
                        CX, 16
 20
                 MOV
 21
           s1:
 22
                       AX, [BX]
                 ADD
 23
                 PUSH
                       [BX]
 24
                 ADD
                        BX, 2
 25
                        S1
                 LOOP
 26
 27
                 MOV
                        DS:SUM, AX
 28
                        BX, 0
                 MOV
 29
                        CX, 16
                 MOV
 30
 31
         S2:
                 POP
                       [BX]
 32
                        BX, 2
                 ADD
 33
                        S2
                 LOOP
 34
 35
                 MOV
                       AH, 4CH
 36
                        21H
                 INT
 37 CODE ENDS
 38 END START
 39
```

### 3.2.2 观察到的现象

运行程序,初始化结束后可以在TD.exe窗口输入DS并回车,查看代码段的数据:



按F2设置断点之后,按F9执行整个程序,会自动暂停,此时再次通过监视窗口查看,可以发现DS段的数据均逆序存储;在Watches窗口添加 SUM值,可以查看到SUM的值为0088H,符合预期结果。



### 3.3 字符串处理实验

#### 3.3.1 源代码

```
1
   DATA SEGMENT
2
                         DB 100
           MaxLength
3
           ActualLength DB ?
4
           STRING
                         DB 100 DUP(?)
5
                         DB ODH, OAH, '$'
           ChangeRow
6
           UpperCase
                        DB 100 DUP(?)
   DATA ENDS
```

```
8
 9
10
    STACK SEGMENT
11
            DW 128 DUP(0)
12
    STACK ENDS
13
14
    CODE SEGMENT
15
                          ASSUME CS: CODE, DS: DATA, SS: STACK
16
             MAIN:
17
                          MOV
                                  AX, DATA
18
                          MOV
                                  DS, AX
19
             INPUT:
20
                          MOV
                                  DX,OFFSET MaxLength
21
                          MOV
                                  AH, 10
                                  21H
22
                          INT
23
24
             ;转换为大写
25
                          MOV
                                  BX, OFFSET STRING
26
                          MOV
                                  DI,OFFSET UpperCase
27
                          MOV
                                  CH, 0
28
                          MOV
                                  CL, Actual Length
29
             ;CX=ACTUALLENGTH
30
             TO_UPPER:
31
                          MOV
                                  AL, [BX]
32
                          CALL
                                  TurnCapital
33
                          MOV
                                  [DI],AL
34
                           INC
                                  вх
35
                           INC
                                  DI
36
                          L00P
                                  TO_UPPER
37
38
             ; 添加终止字符到UpperCase
39
                          MOV
                                  BYTE PTR [DI], 24H
40
41
             ; 输出大写字符串
42
             OUTPUT_UPPER:
43
                          MOV
                                  DX, OFFSET ChangeRow
44
                          MOV
                                  AH, 9
45
                          INT
                                  21H
46
47
                          MOV
                                  DX,OFFSET UpperCase
48
                          MOV
                                  AH,9
                                  21H
49
                          INT
50
51
             ;转换为小写,CX=ACTUALLENGTH
52
                          MOV
                                  BX, OFFSET STRING
53
                          MOV
                                  CH, 0
54
                          MOV
                                  CL, Actual Length
55
             TO_LOWER:
56
                          MOV
                                  AL, [BX]
57
                           CALL
                                  TurnSmall
58
                          MOV
                                  [BX],AL
59
                          INC
                                  вх
60
                          L00P
                                  TO_LOWER
61
62
             ; 添加终止字符到STRING
63
                          MOV
                                  BYTE PTR [BX], 24H
```

```
64
 65
             ; 输出小写字符串
 66
             OUTPUT_LOWER:
 67
                                 DX,OFFSET ChangeRow
                          MOV
 68
                          MOV
                                 AH, 9
 69
                          INT
                                 21H
 70
                                 DX, OFFSET STRING
 71
                          MOV
 72
                          MOV
                                 AH, 9
 73
                                 21H
                          INT
 74
 75
                                 AH,4CH
             END_MAIN:
                          MOV
 76
                          INT
                                 21H
 77
 78
     TurnCapital PROC
 79
             ;AL中的字符转为大写
 80
 81
                                AL,'a'
                          CMP
 82
                          JL
                                 DONE_CAP
 83
                          CMP
                                 AL, 'z'
 84
                                 DONE_CAP
                          JG
 85
                                 AL,20H
                          SUB
             ; 大写=小写-20H
 86
 87
             DONE_CAP:
 88
                          RET
 89
     TurnCapital ENDP
 90
 91
     TurnSmall PROC
 92
             ;AL中的字符作大小写转换
 93
                                 AL,'A'
                          CMP
 94
                          JL
                                 DONE_SMALL
 95
                          CMP
                                 AL, 'Z'
 96
                          JG
                                 DONE_SMALL
 97
                          ADD
                                 AL,20H
98
             DONE_SMALL:
99
                          RET
100
     TurnSmall ENDP
101
102
     CSEG ENDS
103
             END MAIN
104
```

#### 3.3.2 观察到的现象

输入长度不大于100的字符串,可以在屏幕上输出全部大写和全部小写的版本,完成了实验任务的加强 版。

```
D:N>set PATH=C:NTASM
D:N>TASM D:Ntest.asm
Turbo Assembler  Version 4.1  Copyright (c) 1988, 1996 Borland International
Assembling file:
                   D:\test.asm to test.OBJ
Warning* D:\test.asm(10) Reserved word used as symbol: STACK
*Warning* D:\test.asm(102) Unmatched ENDS: CSEG
                  None
Error messages:
Warning messages:
                   Z
Passes:
Remaining memory:
                  466k
D:N>TLINK D:Ntest
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International
Warning: No stack
D:N>D:Ntest
HeLlo
HELLO
hello
D: N>
```

### 3.4 二维数组处理实验

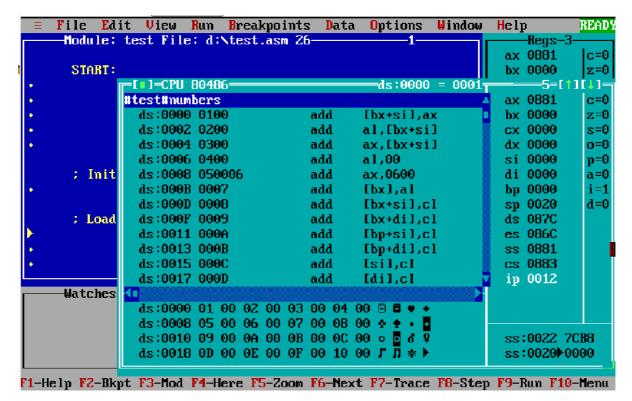
#### 3.4.1 源代码

```
1
    DATA SEGMENT
 2
        NUMBERS
                  DW 0001H, 0002H, 0003H, 0004H, 0005H, 0006H, 0007H, 0008H,
    0009H, 000AH, 000BH, 000CH, 000DH, 000EH, 000FH, 0010H,0011H, 0012H, 0013H,
    0014H, 0015H, 0016H, 0017H, 0018H,0019H, 001AH, 001BH, 001CH, 001DH, 001EH,
    001FH, 0020H
 3
        ChangeRow DB ODH, OAH, '$'
 4
        MAXVAL
                  DW ?
 5
    DATA ENDS
 6
 7
    STACK SEGMENT
              DW 16 DUP(0)
 8
9
    STACK ENDS
10
    CODE SEGMENT
11
12
                      ASSUME CS:CODE, DS:DATA, SS:STACK
13
        MINVAL
                      DW
                                                             ; Store MINVAL at the
    beginning of the code segment
14
15
        START:
16
                             AX, DATA
                      MOV
17
                             DS, AX
                      MOV
18
                             AX, STACK
                      MOV
19
                             SS, AX
                      MOV
20
                      MOV
                             SP, 20H
21
22
        ; Initialize BX to point to the start of the array
23
                      MOV
                             BX, OFFSET NUMBERS
24
25
        ; Load the first number into AX to initialize MAXVAL and MINVAL
26
                      MOV
                             AX, [BX]
```

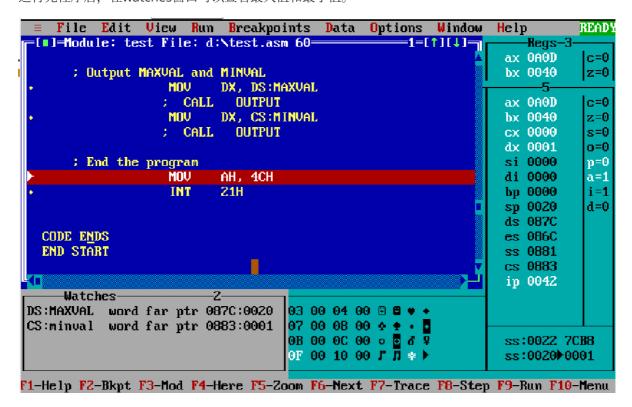
```
27
                    MOV DS:MAXVAL, AX
28
                    MOV
                           CS:MINVAL, AX
29
        ; Set CX to 32 (4*8) for loop count
30
31
                    MOV CX, 32
32
33
       FIND_MIN_MAX:
34
        ; Compare current number with MAXVAL
35
                    CMP
                         AX, DS:MAXVAL
36
                    JLE NOT_MAX
                    MOV
37
                           DS:MAXVAL, AX
38
       NOT_MAX:
39
        ; Compare current number with MINVAL
40
                    CMP
                         AX, CS:MINVAL
                        NOT_MIN
41
                    JGE
42
                    MOV CS:MINVAL, AX
43
       NOT_MIN:
        ; Move to the next number in the array
44
45
                    ADD
                           BX, 2
46
                    MOV AX, [BX]
47
                    LOOP FIND_MIN_MAX
48
49
        ; Output MAXVAL and MINVAL
50
                    MOV
                         DX, DS:MAXVAL
51
                   ; CALL OUTPUT
52
                    MOV DX, CS:MINVAL
53
                   ; CALL OUTPUT
54
55
        ; End the program
56
                    MOV
                         AH, 4CH
57
                    INT
                           21H
58
59
60
   CODE ENDS
61
    END START
62
```

#### 3.4.2 观察到的现象

初始化结束后,可以在DS段观察到代码段的数据。



运行完程序后,在watches窗口可以查看最大值和最小值。



# 3.5 位处理实验

#### 3.5.1 源代码

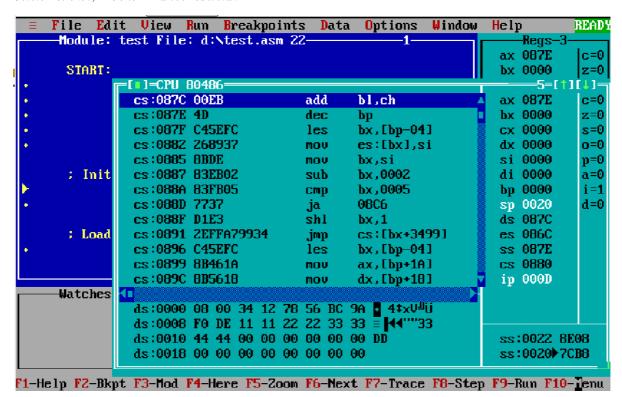
```
DATA SEGMENT
COUNT DW 8
8 words follow
NUMBERS DW 1234H, 5678H, 9ABCH, ODEFOH, 1111H, 2222H, 3333H, 4444H
ENDNUM DW ?
Placeholder for the end of the numbers
DATA ENDS

DATA ENDS
```

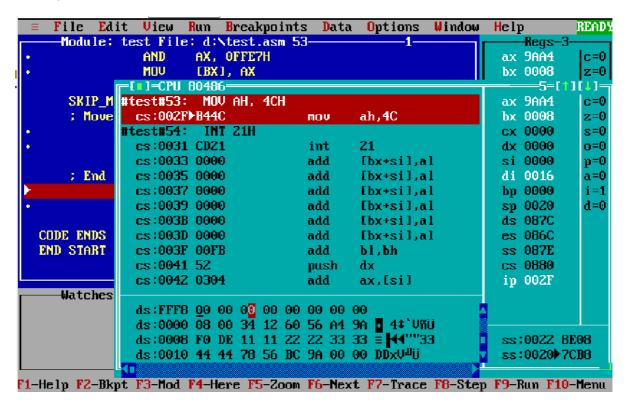
```
7 STACK SEGMENT
              DW 16 DUP(0)
  9
     STACK ENDS
 10
 11
     CODE SEGMENT
 12
                   ASSUME CS:CODE, DS:DATA, SS:STACK
 13
 14
         START:
 15
                   MOV
                          AX, DATA
                          DS, AX
 16
                   MOV
 17
                          AX, STACK
                   MOV
 18
                    MOV
                          SS, AX
 19
                   MOV
                          SP, 20H
 20
          ; Initialize BX to point to the start of the array
 21
 22
                   MOV
                        BX, OFFSET NUMBERS
 23
                   ADD
                          BX, 2
                                                       ; Skip the count
 24
          ; Load the count into CX
 25
 26
                   MOV
                        CX, [OFFSET NUMBERS]
 27
 28
          ; Initialize DI to point to the end of the array
 29
                   LEA DI, ENDNUM
 30
          PROCESS:
 31
 32
          ; Load the current number into AX
 33
                   MOV
                        AX, [BX]
 34
          ; Check if D3 and D4 bits are set
 35
 36
                   TEST AX, 0018H
                                                       ; D3 and D4 bits
 37
                   JΖ
                          SKIP_MOVE
                                                        ; If not set, skip
 38
          ; Move the number to the end
 39
 40
                   MOV
                          [DI], AX
 41
                          DI, 2
                   ADD
 42
 43
          ; Clear D3 and D4 bits
 44
                   AND
                        AX, OFFE7H
 45
                          [BX], AX
                   MOV
 46
 47
         SKIP_MOVE:
 48
          ; Move to the next number in the array
 49
                   ADD
                          BX, 2
 50
                   L00P
                          PROCESS
 51
 52
          ; End the program
                          AH, 4CH
 53
                   MOV
 54
                   INT
                        21H
 55
 56
     CODE ENDS
 57
     END START
 58
```

#### 5.5.4 观察到的现象

初始化结束后,可以在DS段看到初始信息:



运行结束后,可以发现 $D_3$ 、 $D_4$ 位为0的数据已经完成了修改,并且复制到了代码段的最后(5678H、9ABCH):



# 四、收获

通过微机原理实验一,我深入理解了汇编程序的基本结构和分段概念,特别是如何在实际应用中使用 8086指令系统的常见指令。此外,我还掌握了8086的寻址方式,尤其是在处理一维数值数组、字符串、二维数组和位操作时的应用。这次实验加深了我对汇编语言编程的认识,为我后续的学习打下了坚实的基础。