X86 汇编程序设计第二次实验作业

第二次实验，共三道编程题。没有讲解录播。请在本周末前按时提交。

手写程序见文档最后的附件

1. 编写一道完整汇编程序，实现冒泡排序，并显示排序前后的结果。

   要求（提示：参考讲义例题修改）：

   （1） 建立样本数据区，其中包含两个字（分开，分别由学生本人的 8 位学号的 16 进制字组成：XXXXh,YYYYh）。排序后，这两个字可以分开。

   （2） 要显示排序前及排序后的字表，每个字中间空一格。

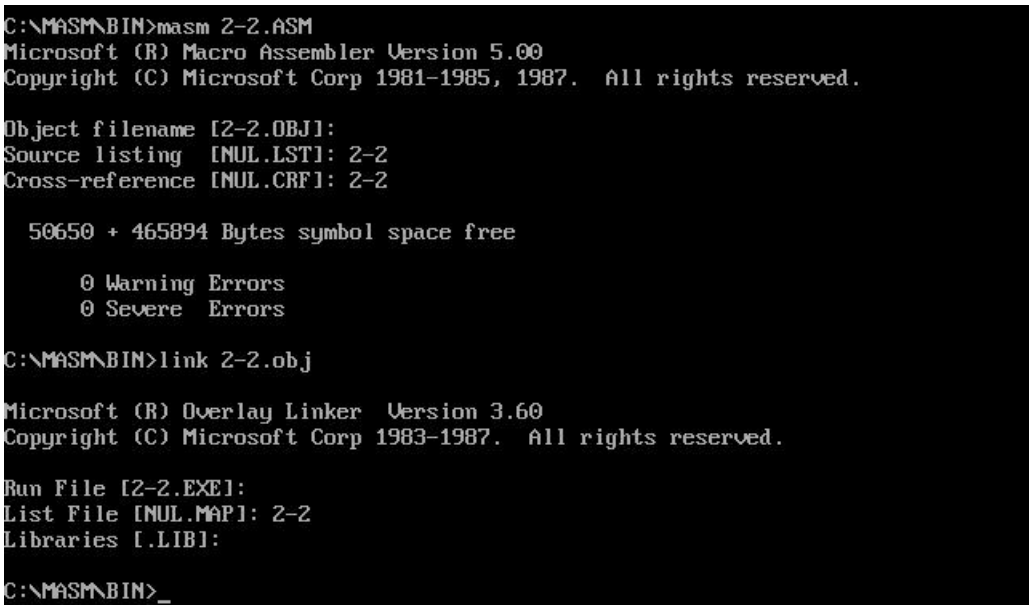
   （3） 要求将排序、显示内存中的字（十六进制至十进制 ASCII 码）、显示字符、显示字符串等程序块改编为子程序或宏。

```
C:\MASM\BIN>masm 2-1.ASM
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [2-1.OBJ]:
Source listing  [NUL.LST]: 2-1
Cross-reference [NUL.CRF]: 2-1

  50534 + 466010 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\MASM\BIN>link 2-1.obj

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

Run File [2-1.EXE]: 2-1
List File [NUL.MAP]: 2-1
Libraries [.LIB]:

C:\MASM\BIN>
```

```
C:\MASM\BIN>2-1.exe
12631 4660 22136 5943
4660 5943 12631 22136

C:\MASM\BIN>
```

可以看到，输出了冒泡排序前后的数组（学号水印：5943=1737H, 12631=3157H），且均没有显示前导 0。

2. 编程实现：从键盘输入一个两位及三位的十进制数，做乘法（假定乘积小于 65535，不考虑溢出），并显示乘法结果的十进制 ASCII 码。



17 * 157 = 2669

3. 编程实现 32 位无符号数乘法。

（1）在内存中定义一个无符号数双字 XX，YY，做乘法，得到一个 64 位的乘积。

（2） 显示该乘积的 16 进制 ASCII 码。

（3） （选做）：显示该乘积的十进制 ASCII 码。

提示：双字 XX 可拆分成两个字 XXH，XXL；双字 YY 可拆分成两个字 YYH，YYL；双字或 64 位结果可用 DD 定义，也可以用 DW 定义，也可以定义为数组（间接寻址）。（XXH，XXL）*（YYH，YYL）=4 个字；乘法列竖式，注意到处都有进位！进位加法用 ADC 指令。

```
C:\MASM\BIN>masm 2-3.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [2-3.OBJ]:
Source listing  [NUL.LST]: 2-3
Cross-reference [NUL.CRF]: 2-3

  50458 + 466086 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\MASM\BIN>link 2-3.obj

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

Run File [2-3.EXE]:
List File [NUL.MAP]: 2-3
Libraries [.LIB]:

C:\MASM\BIN>_
```

```
C:\MASM\BIN>2-3.exe
21682D079C3F90A
150451469471185162
C:\MASM\BIN>_
```

XX=17373157H，YY=17061706H，首先显示乘积的 16 进制，然后显示 10 进制，均与使用 Python 计算的结果（下图）一致。

```
>>> x = 0x17061706
>>> y = 0x17373157
>>> x
386275078
>>> y
389493079
>>> x * y
150451469471185162
>>> print('%#x'%(x*y))
0x21682d079c3f90a
```

第一题手写程序：

2-1 冒泡排序（前后显示结果）

```
STACK       SEGMENT  PARA STACK
STACK_AREA  DW       100H DUP(?)
STACK_TOP   EQU      $ - STACK_AREA
STACK       ENDS

DATA        SEGMENT  PARA
TABLE       DW       3157H, 1234H, 5678H, 1357H
TABLE_LEN   EQU      4
SPACE       DB       20H, '$'
NEW_LINE    DB       0DH, 0AH, '$'
DATA        ENDS

CODE        SEGMENT
            ASSUME CS:CODE, DS:DATA, SS:STACK

MAIN        PROC     FAR
            MOV      AX, STACK
            MOV      SS, AX
            MOV      SP, STACK_TOP
            MOV      AX, DATA
            MOV      DS, AX
            CALL     DISP_ARRAY
            CALL     SORT
            CALL     DISP_ARRAY
            MOV      AX, 4C00H
            INT      21H
MAIN        ENDP

DISP_VALUE  PROC
            PUSH DX
            PUSH CX
            PUSH BX
            PUSH AX
            MOV  CX, 5
            MOV  BX, 10
DVLP1:      XOR  DX, DX
            DIV  BX
            PUSH DX
            LOOP DVLP1
            MOV  BX, 0
            MOV  CX, 5
DVLP2:      POP  DX
            CMP  DL, 0
            JNZ  DVLP2-1
            CMP  BX, 0
            JZ   DVLP2-2
DVLP2-1:    MOV  BX, 1
            OR   DL, 30H
            MOV  AH, 2
            INT  21H
DVLP2-2:    LOOP DVLP2
            POP  AX
            POP  BX
            POP  CX
            POP  DX
            RET
DISP_VALUE  ENDP
DISP_ARRAY  PROC
            PUSH DX
            PUSH CX
            PUSH BX
            PUSH AX
            PUSH SI
            MOV  CX, TABLE_LEN
            MOV  SI, OFFSET TABLE
DALP:       MOV  AX, [SI]
            CALL DISP_VALUE
            MOV  DX, OFFSET SPACE
            MOV  AH, 9
            INT  21H
            INC  SI
            INC  SI
            LOOP DALP
            MOV  DX, OFFSET NEWLINE
            MOV  AH, 9
            INT  21H
            POP  SI
            POP  AX
            POP  BX
            POP  CX
            POP  DX
DISP_ARRAY  ENDP
```

2-1

```
SORT    PROC
        PUSH    DX
        PUSH    CX
        PUSH    BX
        PUSH    AX
        PUSH    SI
PART2:  MOV     CX, TABLE-LEN
        DEC     CX
LP2:    MOV     BX, 1
        MOV     SI, OFFSET TABLE
        PUSH    CX
LP2-1:  MOV     AX, [SI]
        CMP     AX, [SI+2]
        JBE     CONTINUE
        XCHG    AX, [SI+2]
        MOV     [SI], AX
        MOV     BX, 0
CONTINUE: ADD   SI, 2
        LOOP    LP2-1
        POP     CX
        DEC     CX
        CMP     BX, 1
        JZ      EXIT
        JMP     SHORT LP2
EXIT:   POP     SI
        POP     AX
        POP     BX
        POP     CX
        POP     DX
SORT    ENDP
CODE    ENDS
        END     MAIN
```

②

第二题手写程序：

```
2-2  乘法1
STACK        SEGMENT     PARA . STACK
STACK-AREA   DW          100H . DUP(?)
STACK-TOP    EQU         $-STACK-AREA
STACK        ENDS
DATA         SEGMENT     PARA
DATA         ENDS
CODE         SEGMENT
             ASSUME CS:CODE, DS:DATA, SS:STACK
MAIN         PROC        FAR
             MOV         AX, STACK
             MOV         SS, AX
             MOV         SP, STACK-TOP
             MOV         AX, DATA
             MOV         DS, AX
             CALL        GETNUM
             MOV         BX, AX
             CALL        GETNUM
             MUL         BX
             CALL        DISP-VALUE
             MOV         AX, 4C00H
             INT         21H
MAIN         ENDP
GETNUM       PROC
             PUSH        SI
             PUSH        DX
             PUSH        BX
             MOV         SI, 0
             MOV         BX, 10
             XOR         AX, AX
LP1:         MOV         AH, 1
             INT         21H
             CMP         AL, 0DH
             JE          RETURN
             CMP         AL, 30H
             JB          LP1
             CMP         AL, 39H
             JA          LP1
             AND         AL, 0FH
             XOR         AH, AH
             PUSH        AX
             MOV         AX, SI
             MUL         BX
             MOV         ~~AX, SI~~ SI, AX
             POP         AX
             ADD         SI, AX
             LOOP        LP1
RETURN:      MOV         AX, SI
             POP         BX
             POP         DX
             POP         SI
GETNUM       ENDP
DISP-VALUE   PROC
             PUSH        DX
             PUSH        CX
             PUSH        BX
             PUSH        AX
             MOV         CX, 5
             MOV         BX, 10
DVLP1:       XOR         DX, DX
             DIV         BX
             PUSH        DX
             LOOP        DVLP1
             MOV         BX, 0
             MOV         CX, 5
DVLP2:       POP         DX
             CMP         DL, 0
             JNZ         DVLP2-1
             CMP         BX, 0
             JZ          DVLP2-2
DVLP2-1:     MOV         BX, 1
             OR          DL, 30H
             MOV         AH, 2
             INT         21H
DVLP2-2:     LOOP        DVLP2
             POP         AX
             POP         BX
             POP         CX
             POP         DX
DISP-VALUE   ENDP
CODE         ENDS
             END         MAIN
```

③

第三题手写程序：

```
2-3        乘法2 (32位×32位)

STACK       SEGMENT   PARA.STACK
STACK_AREA  DW        100H DUP(?)
STACK_TOP   EQU       $-STACK_AREA
STACK       ENDS

DATA        SEGMENT   PARA
XX          DD        17313157H
YY          DD        17061706H
RESULT      DW        4 DUP(0H)
NEWLINE     DB        0DH,0AH,'$'
DATA        ENDS

CODE        SEGMENT
            ASSUME CS:CODE,DS:DATA,SS:STACK
MAIN        PROC      FAR
            MOV       AX,STACK
            MOV       SS,AX
            MOV       SP,STACK_TOP
            MOV       AX,DATA
            MOV       DS,AX
            CALL      MUL_32
            CALL      DISP_HEX
            MOV       DX,OFFSET NEWLINE
            MOV       AH,9
            INT       21H
            CALL      DISP_DEC
            MOV       AX,4C00H
            INT       21H
MAIN        ENDP
MUL-32      PROC
            PUSH      SI
            PUSH      DI
            PUSH      AX
            PUSH      BX
            PUSH      DX
            MOV       SI,OFFSET XX
            MOV       DI,OFFSET YY
            MOV       AX,WORD PTR[SI]
            MOV       BX,WORD PTR[DI]
            MUL       BX
            MOV       [RESULT],AX
            MOV       [RESULT+2],DX
            MOV       AX,WORD PTR[SI+2]
            MOV       BX,WORD PTR[DI]
            MUL       BX
            ADD       [RESULT+2],AX
            ADC       [RESULT+4],DX
            ADC       [RESULT+6],0
            MOV       AX,WORD PTR[SI]
            MOV       BX,WORD PTR[DI+2]
            MUL       BX
            ADD       [RESULT+2],AX
            ADC       [RESULT+4],DX
            ADC       [RESULT+6],0
            MOV       AX,WORD PTR[SI+2]
            MOV       BX,WORD PTR[DI+2]
            MUL       BX
            ADD       [RESULT+4],AX
            ADC       [RESULT+6],DX
            POP       DX
            POP       CX      (struck out)
            POP       BX
            POP       AX
            POP       DI
            POP       SI
            RET
MUL-32      ENDP
DISP_VALUE  PROC
            PUSH      DX
            PUSH      CX
            PUSH      BX
            PUSH      AX
            MOV       CX,4
            MOV       BX,16
DLP1:       XOR       DX,DX
            DIV       BX
            ADD       DL,30H
            CMP       DL,39H
            JBE       DIGIT
            ADD       DL,'A'-'9'-1
DIGIT       PUSH      DX
            LOOP      DLP1
```

④

2-3

```
        MOV     CX, 4
DLP2:   POP     DX
        MOV     AH, 2
        INT     21H
        LOOP    DLP2
        POP     AX
        POP     BX
        POP     CX
        POP     DX
DISP_VALUE  ENDP
DISP_HEX    PROC
        PUSH    DX
        PUSH    CX
        PUSH    AX
        PUSH    SI
        MOV     CX, 4
        MOV     SI, OFFSET RESULT+6
L1:     MOV     AX, [SI]
        CALL    DISP_VALUE
        DEC     SI
        DEC     SI
        LOOP    L1
        POP     SI
        POP     AX
        POP     CX
        POP     DX
DISP_HEX    ENDP
DISP_DEC    PROC
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        MOV     CX, 20
        MOV     BX, 10
        MOV     SI, OFFSET RESULT
DECLP1: XOR     DX, DX
        MOV     AX, [SI+6]
        DIV     BX
        MOV     [SI+6], AX
        MOV     AX, [SI+4]
        DIV     BX
        MOV     [SI+4], AX
        MOV     AX, [SI+2]
        DIV     BX
        MOV     [SI+2], AX
        MOV     AX, [SI]
        DIV     BX
        MOV     [SI], AX
        PUSH    DX
        LOOP    DECLP1
        MOV     BX, 0
        MOV     CX, 20
DECLP2: POP     DX
        CMP     DL, 0
        JNZ     DECLP2-1
        CMP     BX, 0
        JZ      DECLP2-2
DECLP2-1:   MOV     BX, 1
        OR      DL, 30H
        MOV     AH, 2
        INT     21H
DECLP2-2:   LOOP    DECLP2
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
DISP_DEC    ENDP
CODE    ENDS
        END     MAIN
```