X86 汇编程序设计第三次实验作业

手写代码见最后

1. 编写一道完整汇编程序，实现以下要求（提示：参考讲义例题修改）：

（1） 编写子程序 Memmove，将处于同一数据段的串 String1 复制到 String2。子程序入口参数为：DS:SI 指向 String1,， ES:DI 指向 String2。

（2）数据段先定义 String1 为你的姓名的汉语拼音，在此串前后各留一点空余串。构造 String2 的首地址分别为(1)与 String1 不重叠；(2) String2 地址在 String1 前，但有重叠；(3) String2 地址在 String1 后，但有重叠。

（3）主程序分三次调用 Memmove，每次调用前显示 String1，调用后显示 String1，String2。

```
C:\MASM\BIN>masm 3-1.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [3-1.OBJ]:
Source listing  [NUL.LST]: 3-1
Cross-reference [NUL.CRF]: 3-1

  50610 + 465934 Bytes symbol space free

       0 Warning Errors
       0 Severe  Errors

C:\MASM\BIN>link 3-1.obj

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

Run File [3-1.EXE]:
List File [NUL.MAP]: 3-1
Libraries [.LIB]:

C:\MASM\BIN>
```

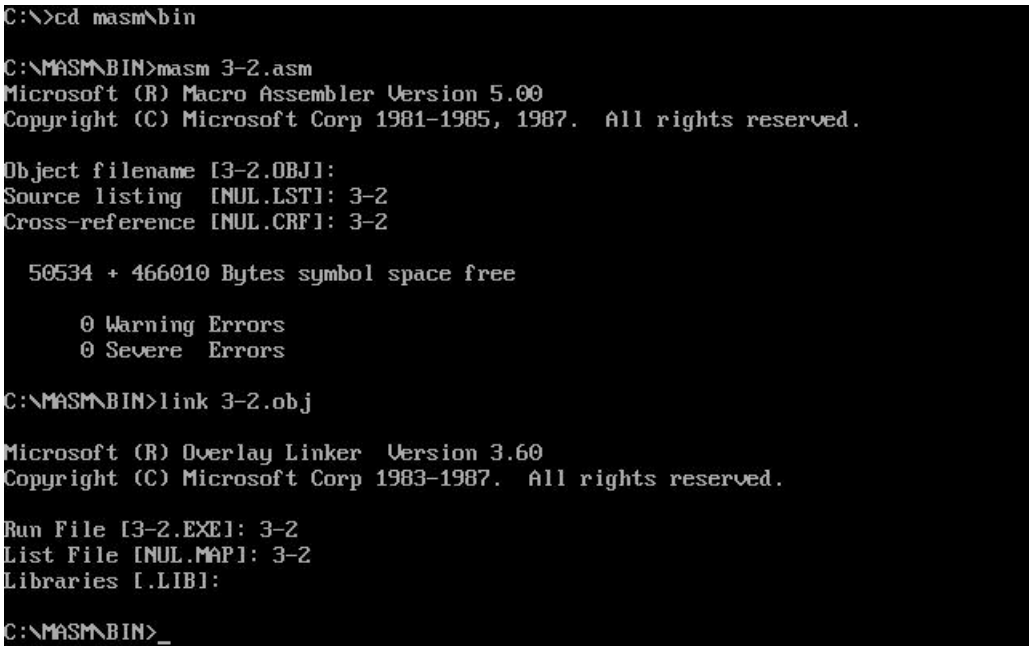
```
C:\MASM\BIN>3-1.exe
zhaoliangxuan zhaoliangxuan zhaoliangxuan
zhaoliangxuan ngxuan zhaoliangxuan
ngxuan ngxuan ngxuan

C:\MASM\BIN>
```

（1）常规的 strcmp

（2）String2 在 String1 前，故 String2 的尾覆盖了 String1 的头

（3）在（2）的基础上，String2 地址在 String1 后，只复制了“残存”的 String1

2. 编写一道完整汇编程序，实现：从键盘输入一个字符串 String2（测试时输入你的姓名的汉语拼音），与内存中的已有字符串 String1 比较。根据比较结果，显示“XXXX>=<YYYY”。（提示：输入、比较、显示编写为子程序或宏；显示结果时，分段显示 XXXX，字符=（或<，>），YYYY）。



已有字符串 String1 为 liangxuanxuan，输入字符串为 zhaoliangxuan。

按照字典序，有 liangxuanxuan < zhaoliangxuan。

3. 编程实现（讲义 149 页第 6 题）：

6. 编写一个有主程序及子程序结构的程序模块。主程序从键盘接收一串字符建立串 STRING，然后将 STRING 的首地址、串长度作为参数通过堆栈传递给段内调用的子程序，同时在 AL 中放入要查找的字符。子程序 FIND 查找 AL 中的字符在 STRING 中出现的次数，并将出现次数放入 AX 返回。

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

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

  50572 + 465972 Bytes symbol space free

       0 Warning Errors
       0 Severe  Errors

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

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

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

C:\MASM\BIN>
```

```
C:\MASM\BIN>3-3.exe
zhaoliangxuan
3
C:\MASM\BIN>_
```

设置的要查找的字符为'a'，输入的串 STRING 为'zhaoliangxuan'，故输出 3

4. 选做题：先构造一个不等长的单词表（区分大小写），然后进行字典式排序，再输入一个单词（你的姓名的汉语拼音），插入到此单词表中。要求输出显示排序前后的单词表。（此题不用手写，只需提交运行结果及源程序）。

```
C:\MASM\BIN>masm 3-4.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [3-4.OBJ]:
Source listing  [NUL.LST]: 3-4
Cross-reference [NUL.CRF]: 3-4

  50496 + 449664 Bytes symbol space free

       0 Warning Errors
       0 Severe  Errors

C:\MASM\BIN>link 3-4.obj

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

Run File [3-4.EXE]:
List File [NUL.MAP]: 3-4
Libraries [.LIB]:

C:\MASM\BIN>
```

```
beihang
SCSE
X86
2020
zuoye3
xuanzuoti

2020
SCSE
X86
beihang
xuanzuoti
zuoye3

zhaoliangxuan

2020
SCSE
X86
beihang
xuanzuoti
zhaoliangxuan
zuoye3

C:\MASM\BIN>
```

单词表为：beihang, SCSE, X86, 2020, zuoye3, xuanzuoti

排序后为：2020, SCSE, X86, beihang, xuanzuoti, zuoye3

插入 zhaoliangxuan 再排序后为：2020, SCSE, X86, beihang, xuanzuoti, zhaoliangxuan, zuoye3

手写代码：

3-1：

```
3-1 MemMove.
STACK       SEGMENT     PARA. STACK
STACK-AREA  DW          100h DUP(?)
STACK_TOP   EQU         $-STACK_AREA
STACK       ENDS

DATA        SEGMENT     PARA
STR1        DB          'zhaoliangxuan', '$'
STR2        DB          30, DUP(?)
SPACE       DB          20H, '$'
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
            MOV         ES, AX
            MOV         BX, OFFSET STR2
            CALL        DISP. STR
            MOV         BX, OFFSET STR1-7
            CALL        DISP-STR
            MOV         BX, OFFSET STR1+8
            CALL        DISP-STR
            MOV         AX, 4C00H
            INT         21H
MAIN        ENDP

DISP-STR    PROC
            MOV         DX, OFFSET STR1
            MOV         AH, 09H
            INT         21H
            MOV         DX, OFFSET SPACE
            MOV         AH, 09H
            INT         21H
            MOV         SI, OFFSET STR1
            MOV         DI, BX
            CALL        MEM. MOVE
            MOV         DX, OFFSET STR1
            MOV         AH, 09H
            INT         21H
            MOV         DX, OFFSET SPACE
            MOV         AH, 09H
            INT         21H
            MOV         DX, BX
            MOV         AH, 09H
            INT         21H
            MOV         DX, OFFSET NEW-LINE
            MOV         AH, 09H
            INT         21H
DISP_STR    ENDP

MEM_MOVE    PROC
            CLD
MM_LOOP0    LODSB
            STOSB
            CMP. AL, '$'
            JNZ MM_LOOP
            RET
MEM_MOVE    ENDP
CODE        ENDS
            END MAIN
```

3-2：

```
3-2
STACK       SEGMENT     PARA.STACK
STACK_AREA  DW          100H. DUP(?)
STACK_TOP   EQU         $- STACK_AREA
STACK       ENDS

DATA        SEGMENT     PARA
RESULT      DW          ?
STR1        DB          'zhaoliangxuan'. '$'
LEN         EQU         121
IN_BUFF     DB          LEN-1
            DB          ?
            DB          LEN DUP(?)
ABOVE       DB          '>', '$'
BELOW       DB          '<'. '$'
EQUAL       DB          '='. '$'
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
            MOV         ES. AX .
            ~~CALL        INPUT_STR~~
            ~~MOV         SI. OFFSET IN_BUF+2~~
            ~~MOV         CL, IN_BUF+1~~
            ~~XOR         CH. CH~~
            ~~PUSH        SI~~
            ~~PUSH        CX~~
            ~~MOV         AL, 'a'~~
            CALL        INPUT_STR
            MOV         SI. OFFSET STR1
            MOV         DI, OFFSET IN_BUF+2
            CALL        STRCMP
            MOV         DX, OFFSET. STR1
            CALL        DIS. STR
            MOV         AX. RESULT
            CMP         AX. 1
            JA          MAIN_A
            JZ          MAIN_Z
            MOV         DX. OFFSET BELOW
            JMP         PRINT
MAIN_A:     MOV         DX. OFFSET ABOVE
            JMP         PRINT
MAIN_Z:     MOV         DX. OFFSET EQUAL
PRINT:      CALL        DIS_STR
            MOV         DX. OFFSET IN_BUF+2
            CALL        DIS_STR
            MOV         AX. 4C00H
            INT         21H
MAIN        ENDP

INPUT_STR:  PROC
            MOV         AH. 0AH
            MOV         DX. OFFSET IN_BUF
            INT         21H
            MOV         CL. IN_BUF+1
            XOR         CH. CH
            MOV         SI. OFFSET IN_BUF+2
            ADD         SI. CX
            MOV         BYTE PTR [SI]. '$'
INPUT_STR   ENDP

STRCMP      PROC
            PUSH        CX
            PUSH        BX
            PUSH        AX
LP3:        CMP         BYTE PTR [SI]. '$'
            JZ          LP3-1
            CMP         BYTE PTR [DI]. '$'
            JZ          A
            JMP         LP3-2
LP3-1:      CMP         BYTE. PTR [DI]. '$'
            JZ          Z
            JMP         B
```

```
LP3_2:  MOV     AL,BYTE PTR[SI]
        CMP     AL,BYTE PTR[DI]
        JA      A
        JB      B
        INC     SI
        INC     DI
        JMP     LP3
B:      MOV     RESULT,0
        JMP     RETURN
A:      MOV     RESULT,2
        JMP     RETURN
E:      MOV     RESULT,1
RETURN: POP     AX
        POP     BX
        POP     CX
        RET
STRCMP  ENDP

DIS_STR PROC
        MOV     AH,09H
        INT     21H
        RET
DIS_STR ENDP

CODE    ENDS
        END     MAIN
```

3-3：

```
3-3
STACK       SEGMENT     PARA. STACK
STACK_AREA  DW          100H. DUP(?)
STACK_TOP   EQU         $-STACK_AREA
STACK       ENDS
DATA        SEGMENT     PARA
~~STR1      DB          'Enter a char: $'~~
~~STR2      DB          30. DUP(?)~~
LEN         EQU         121
IN_BUFF     DB          LEN-1
            DB          ?
            DB          LEN. DUP(?)
NEW_LINE    DB          0DH. 0AH. '$'
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
            MOV         ES. AX.
            CALL        INPUT_STR
            MOV         SI, OFFSET IN_BUF+2
            MOV         CL, IN_BUF+1
            XOR         CH. CH
            PUSH        SI
            PUSH        CX
            MOV         AL. 'a'
            CALL        FIND
            PUSH        AX
            MOV         DX, OFFSET NEW_LINE
            MOV         AX. 09H
            INT         21H
            POP         AX
            MOV         DX. AX
            OR          DL, 30H
            MOV         AH, 2
            INT         21H
            MOV         AX, 4C00H
            INT         21H
MAIN        ENDP
FIND        PROC
            MOV         BP. SP
            MOV         CX. [BP+2]
            MOV         SI. [BP+4]
            PUSH        DX
            PUSH        BX
            XOR         BX. BX
            MOV         DL. AL
            CLD
LP1:        LODSB
            CMP         AL. '$'
            JZ          LP1_1
            CMP         AL. DL
            JNZ         LP1
            INC         BX
            JMP         LP1
LP1_1       MOV         AX. BX
            POP         BX
            POP         DX
            RET
FIND        ENDP
INPUT_STR   PROC
            MOV         AH. 0AH
            MOV         DX. OFFSET IN_BUF
            INT         21H
            MOV         CL. IN_BUF+1
            XOR         CH. CH
            MOV         SI. OFFSET IN_BUF+2
            ADD         SI. CX
            MOV         BYTE PTR[SI]. '$'
            RET
INPUT_STR   ENDP
CODE        ENDS
            END         MAIN
```