X86 汇编程序设计第四次实验作业

手写程序见最后

1. 主程序构造样本串 1，然后从键盘输入一个字符串（你姓名的汉语拼音），拼接成一个长的字符串，并显示出来。其中，以下内容要编写成子程序，主程序通过堆栈传递参数调用子程序：

   （1） 从键盘输入字符串，参数为 IN_BUF 的首地址；

   （2） 拼接字符串，参数为 S1，S2 的首地址；

   （3） 显示字符串（是 ASCIIZ 串，长度未知），参数为串的首地址。

```
C:\MASM\BIN>masm 4-1.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [4-1.OBJ]: 4-1
Source listing  [NUL.LST]: 4-1
Cross-reference [NUL.CRF]: 4-1

  50532 + 466012 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\MASM\BIN>link 4-1.obj

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

Run File [4-1.EXE]:
List File [NUL.MAP]: 4-1
Libraries [.LIB]:

C:\MASM\BIN>_
```

```
C:\MASM\BIN>4-1.exe
zhaoliangxuan
17373157zhaoliangxuan
```

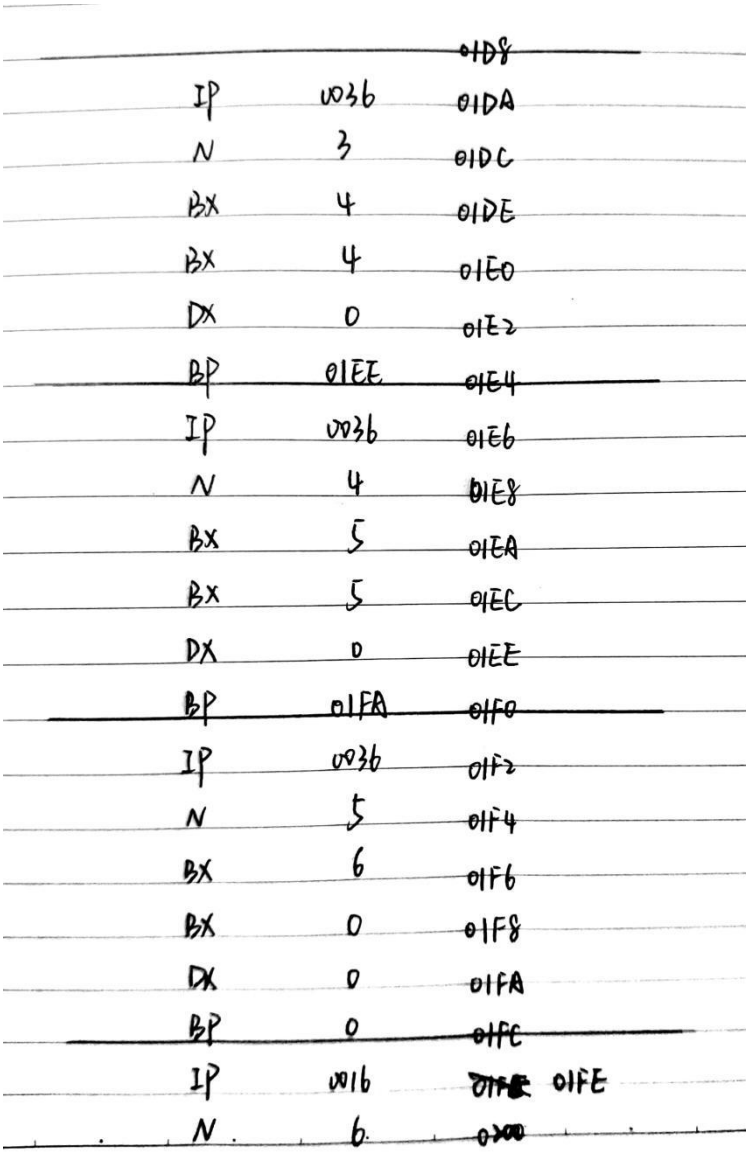
样本串 1 为 17373157，输入 zhaoliangxuan，输出拼接后的结果 17373157zhaoliangxuan

2. 编写递归程序实现求 N!（讲义例题），N 的初值为 6。在 DEBUG 下，画出 N=3 时刚进入子程序时的堆栈图：**左边标注出堆栈中从初始栈顶至当前栈顶的每个字的含义(如 …，BP，IP，N)，右边为 SP 的值(如…，01FA，01FC，01FE)，中间单元格内是每个字单元的 16 进制值**。先在 DEBUG 下跟踪执行至 N=3 时，并进入子程序，然后截取堆栈数据区，标注出当前栈顶至初始栈顶的区域（画出下划线），然后手工画堆栈图，并标注，拍照。

```
C:\MASM\BIN>masm 4-2.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [4-2.OBJ]:
Source listing  [NUL.LST]: 4-2
Cross-reference [NUL.CRF]: 4-2

  50534 + 466010 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\MASM\BIN>link 4-2.obj

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

Run File [4-2.EXE]:
List File [NUL.MAP]: 4-2
Libraries [.LIB]:

C:\MASM\BIN>4-2.exe
720
C:\MASM\BIN>
```

6! = 720，答案正确

```
-g 001e

AX=0006  BX=0003  CX=0282  DX=0000  SP=01D8  BP=01E2  SI=0000  DI=0000
DS=078A  ES=078A  SS=076A  CS=078B  IP=001E   NV UP EI PL NZ NA PE NC
078B:001E 55            PUSH    BP
-d ss:1a0
076A:01A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:01B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:01C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:01D0  E2 01 1E 00 8B 07 A3 01-36 00 03 00 04 00 04 00   ........6.......
076A:01E0  00 00 EE 01 36 00 04 00-05 00 05 00 00 00 FA 01   ....6...........
076A:01F0  36 00 05 00 06 00 00 00-00 00 00 00 16 00 06 00   6...............
076A:0200  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0210  B8 6A 07 8E D0 BC 00 02-B8 8A 07 8E D8 8E C0 B8   .j..............
```



3. （选做题）编写一道完整汇编程序，实现以下要求：

（1） 编写 5 个子程序，分别完成字符串输入、在串中查找字符、比较两个字符串、将串 1 复制到串 2、显示字符串。

（2）主程序完成样本串定义（其中有一个串是你姓名的汉语拼音），构造子程序地址数组（函数指针数组）；构造一个无限循环，从键盘输入 1-5，作为索引，根据索引值 0-4，使用段内间接转移方式调用相应的子程序（如 CALL BX，BX=0-4）。如果从键盘输入 0，则循环结束；如果从键盘输入了其他字符，忽略掉，并继续循环。

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

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

  50572 + 449588 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

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

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

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

C:\MASM\BIN>_
```

```
C:\MASM\BIN>4-3.exe
5
zhaoliangxuan x86
1
buaa
buaa
2
a:3
3
zhaoliangxuan > x86
4

5
zhaoliangxuan zhaoliangxuan
3
zhaoliangxuan = zhaoliangxuan
0

C:\MASM\BIN>_
```

两个预设的字符串分别是 zhaoliangxuan 和 x86。

1. 输入 5：显示 zhaoliangxuan 和 x86 两个字符串
2. 输入 1：输入一个字符串 buaa，然后回显输入的字符串 buaa
3. 输入 2：输入字符 a，统计字符串 zhaoliangxuan 中出现字符 a 的次数，显示 3
4. 输入 3：按照字典序比较两个字符串，结果为 zhaoliangxuan > x86
5. 输入 4：将字符串 1 复制到字符串 2 处
6. 输入 5：再次显示两个字符串，此时均为 zhaoliangxuan
7. 输入 3：再次比较两个字符串，此时 zhaoliangxuan = zhaoliangxuan

4-1:

4-1 文字列結合

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

DATA        SEGMENT   PARA
STR1        DB        '123456', 0
IN_LEN      EQU       12
IN_BUF      DB        IN_LEN-1
            DB        ?
            DB        IN_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
            MOV       SI, OFFSET IN_BUF
            PUSH      SI
            CALL      READ_STR
            MOV       DX, OFFSET NEWLINE
            MOV       AH, 9
            INT       21H
            MOV       SI, OFFSET IN_BUF+2
            PUSH      SI
            MOV       SI, OFFSET STR1
            PUSH      SI
            CALL      CONCAT_STR
            MOV       SI, OFFSET STR1
            PUSH      SI
            CALL      DISP_STR
            MOV       AX, 4C00H
            INT       21H
MAIN        ENDP

READ_STR:   PROC
            PUSH      BP
            MOV       BP, SP
            PUSH      AX
            PUSH      DX
            MOV       AH, 0AH
            MOV       DX, [BP+4]
            INT       21H
            POP       DX
            POP       AX
            POP       BP
            RET       2
READ_STR    ENDP

CONCAT_STR: PROC
            PUSH      BP
            MOV       BP, SP
            PUSH      DI
            PUSH      SI
            PUSH      AX
            MOV       SI, [BP+4]
            MOV       DI, [BP+6]
            CLD
CAT_LP1:    LODSB
            CMP       AL, 0
            JNZ       CAT_LP1
            DEC       SI
CAT_LP2:    MOV       AL, BYTE PTR [DI]
            MOV       BYTE PTR [SI], AL
            INC       SI
            INC       DI
            CMP       AL, 0
            JNZ       CAT_LP2

            POP       AX
            POP       SI
            POP       DI
            POP       BP
            RET       4
CONCAT_STR  ENDP
```

```
DISP_STR:   PROC
            PUSH    BP
            MOV     BP. SP
            PUSH    SI
            PUSH    AX
            PUSH    DX
            MOV     SI. [BP+4]
            CLD
DISP1:      LODSB
            CMP     AL. 0
            JZ      DISP_EXIT
            MOV     DL. AL
            MOV     AH. 2
            INT     21H
            JMP     DISP1
DISP_EXIT:  MOV     DL. 0DH
            MOV     AH. 2
            INT     21H
            MOV     DL. 0AH
            MOV     AH. 2
            INT     21H
            POP     DX
            POP     AX
            POP     SI
            POP     BP
            RET     2
DISP_STR    ENDP
CODE        ENDS
            END     MAIN
```

4-2:

4-2 递归求N!

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

DATA        SEGMENT     PARA
DATA        ENDS

CODE        SEGMENT     ~~PARA~~
            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         AX, 6
            PUSH        AX
            CALL        CALCULATE
            CALL        DISP_VALUE
            MOV         AX, 4C00H
            INT         21H
MAIN        ENDP

CALCULATE   PROC
            PUSH        BP
            MOV         BP, SP
            PUSH        DX
            PUSH        BX
            MOV         BX, [BP+4]
            CMP         BX, 1
            JNZ         CAL1
            MOV         AX, 1
            JMP         CAL2
CAL1:       PUSH        BX
            DEC         BX
            PUSH        BX
            CALL        CACULATE
            POP         BX
            MUL         BX
CAL2:       POP         BX
            POP         DX
            POP         BP
            RET         2
CACULATE    ENDP

DISP_VALUE: PROC
            PUSH        DX
            PUSH        CX
            PUSH        BX
            PUSH        AX
            MOV         CX, 5
            MOV         BX, 10
DLP1:       XOR         DX, DX
            DIV         BX
            PUSH        DX          ;余
            LOOP        DLP1
            MOV         CX, 5
            MOV         BX, 0
DLP2:       POP         DX
            CMP         DL, 0
            JNZ         DLP2-1
            CMP         BX, 0
            JZ          DLP2-2
DLP2-1:     MOV         BX, 1
            OR          DL, 30H
            MOV         AH, 2
            INT         21H
DLP2-2:     LOOP        DLP2
            POP         AX
            POP         BX
            POP         CX
            POP         DX
DISP_VALUE  ENDP

CODE        ENDS
            END         MAIN
```