# 西安电子科技大学

## 微机原理与系统设计 课程实验报告

**实验名称 微机原理实验**

人工智能 学院 23200XX 班

姓名 X X X 学号 2300920XXXX

同作者 无

实验日期 2025 年 12 月 7 日

| 成 绩 |
| --- |
| |

## 一、实验目的

通过子程序调用实现以下 5 个功能，通过从键盘输入 1-5 实现功能选择

(1)功能 1：把存储器空间的一串字符串实现小写转大写

EG: Xidian University 2024

XIDIAN UNIVERSITY 2024

(2)功能 2：在存储器空间中一串字符串中找出最大值显示出来

显示结果：'Xidian University 2024', 0F1H, 25H,0DH

The maxmum number is : 0F1H

(3)功能 3： 将存储器中的一段数组排序

显示结果：05H.81H,32H,44H

81H,44H,32H,05H

(4)功能 4：显示系统时间： 时：分：秒

(5)功能 5：回到功能选择

## 二、实验所用仪器（或实验环境）

操作系统：Windows 11

编程工具：emu8086（8086 模拟器）

## 三、实验基本原理及步骤（或方案设计及理论计算）

1. **字符操作与子程序调用：**该实验通过编写 8086 汇编程序，演示了如何通过子程序对存储器中的字符串进行各种操作。通过键盘输入数字（1-5）来选择不同的功能，每个功能都会调用一个相应的子程序来处理输入字符串或进行其它操作(如显示系统时间)。实验中使用了子程序的调用与返回机制，体现了 8086 汇编语言中的栈操作和寄存器操作。

2. **功能实现原理：**

（1） **功能 1：实现小写字母转大写字母。**通过遍历字符串中的每个字符，如果字符是小写字母（在 ASCII 码范围内），就将其转为大写字母。转换规则是通过减少 32 的方式将小写字母转换为大写字母（因为小写字母和大写字母的 ASCII 码差值是 32）。

（2） **功能 2：寻找字符串中的最大字符。**通过遍历字符串中的每个字符，比较其 ASCII 值，最终找到并显示出最大值。

（3） **功能 3：对存储在字符串中的数据进行排序。**该实验通过实现简单的冒泡排序算法，对数据进行升序排列。排序过程使用了寄存器存储和交换数据的方法。

（4） **功能 4：显示系统时间。**通过调用 BIOS 的时间中断（INT 1AH）来获取当前的系统时间，并将其格式化为小时:分钟:秒的格式显示出来。

（5） **功能 5：**返回到功能选择界面，允许用户重新选择功能。

3. **实验步骤：**

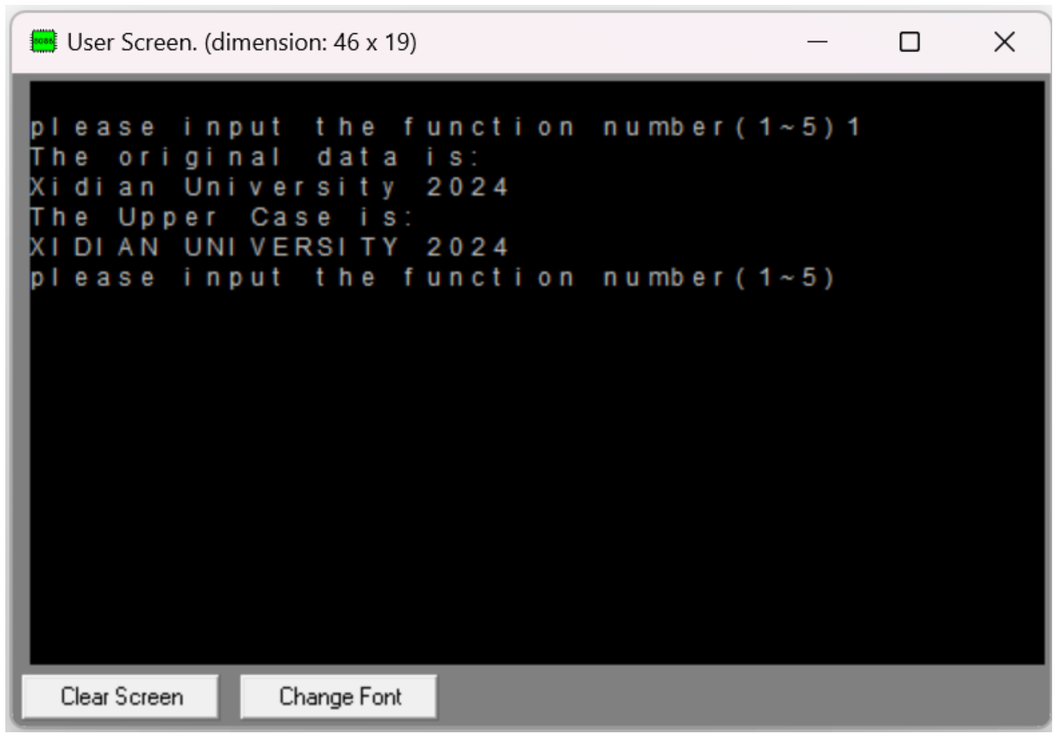
- 设置数据段（DSEG）和代码段（CSEG），并初始化堆栈段（SSEG）以进行寄存器的保存与恢复。
- 实现主功能选择界面，通过键盘输入 1-5，调用相应的子程序执行功能。
- 每个功能的实现都通过子程序来完成，使用循环、条件判断等基本控制结构。

4. **理论计算：**

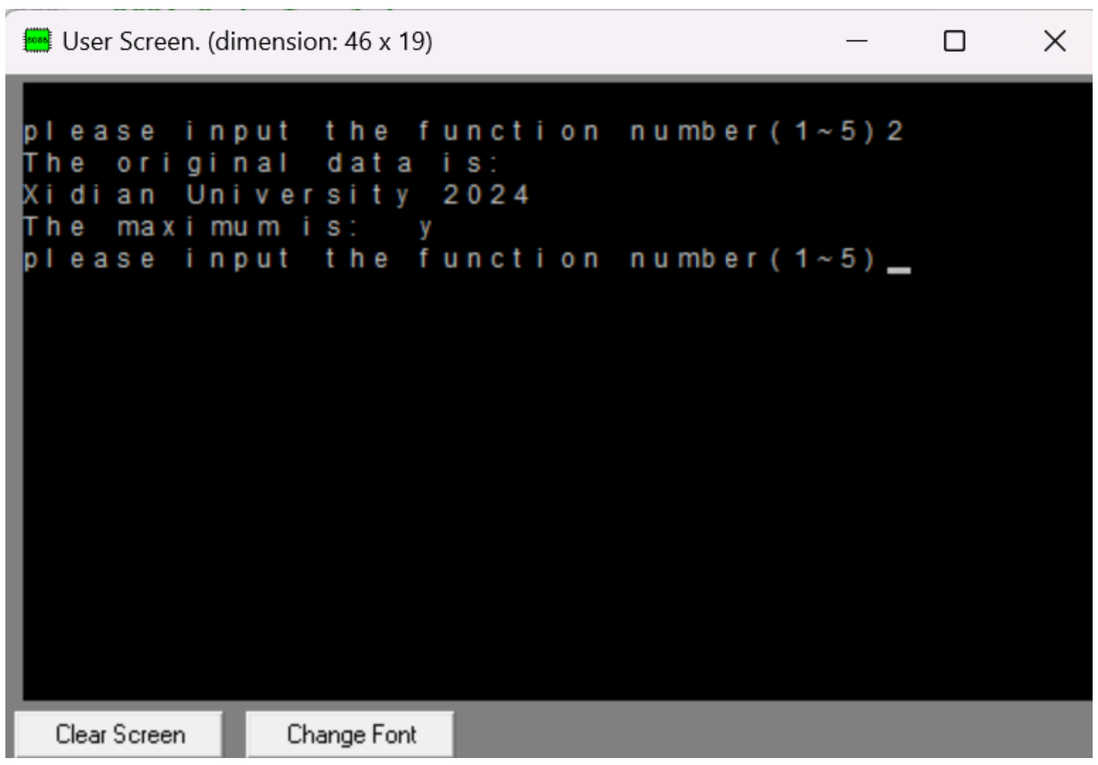
- **小写转大写：**根据 ASCII 码表，小写字母与大写字母之间的差值为 32（即 a - A = 32），通过将小写字母减去 32 即可得到对应的大写字母。
- **排序算法：**实验中使用的是冒泡排序算法，复杂度为 $O(n^2)$，适用于小规模数据的排序。
- **系统时间获取：**通过 BIOS 中断 INT 1AH 获取系统时间，返回的时间以 24 小时制表示，包含小时、分钟和秒。

## 四、实验结果与分析（图表等）

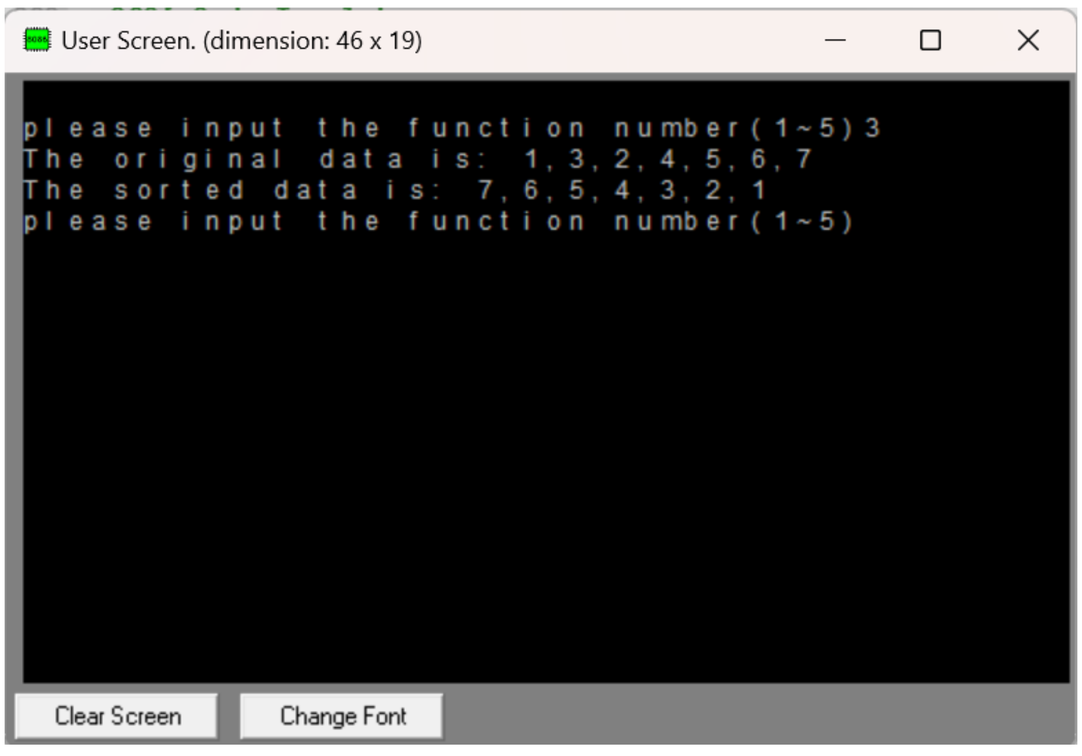
（1） 选择功能 1，把存储器空间的一串字符串实现小写转大写，结果如下：



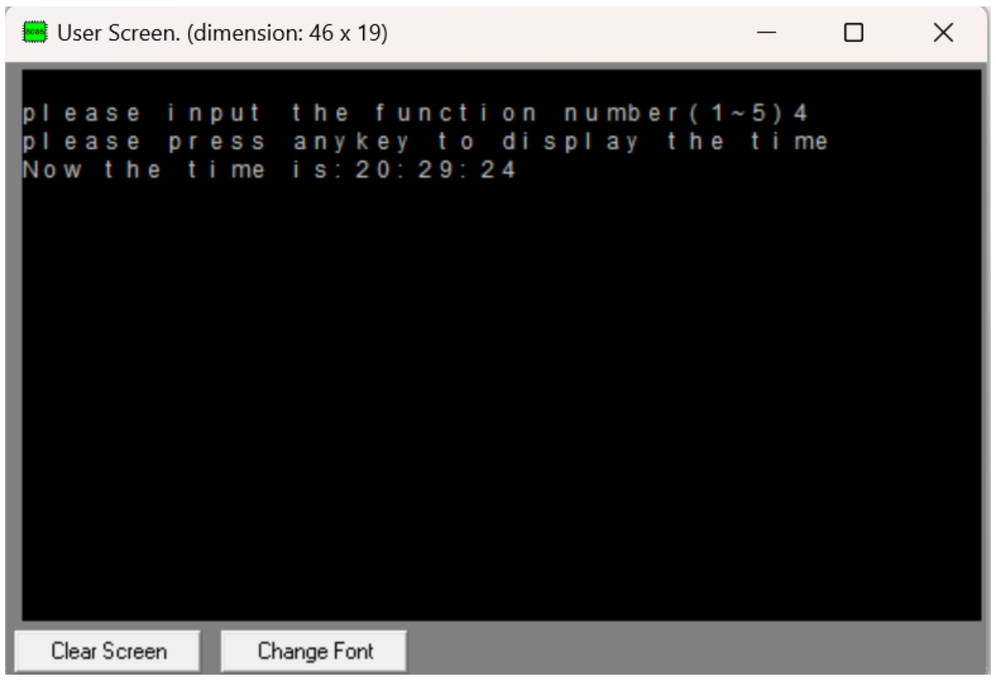

（2） 选择功能 2，寻找字符串中的最大字符，结果如下：



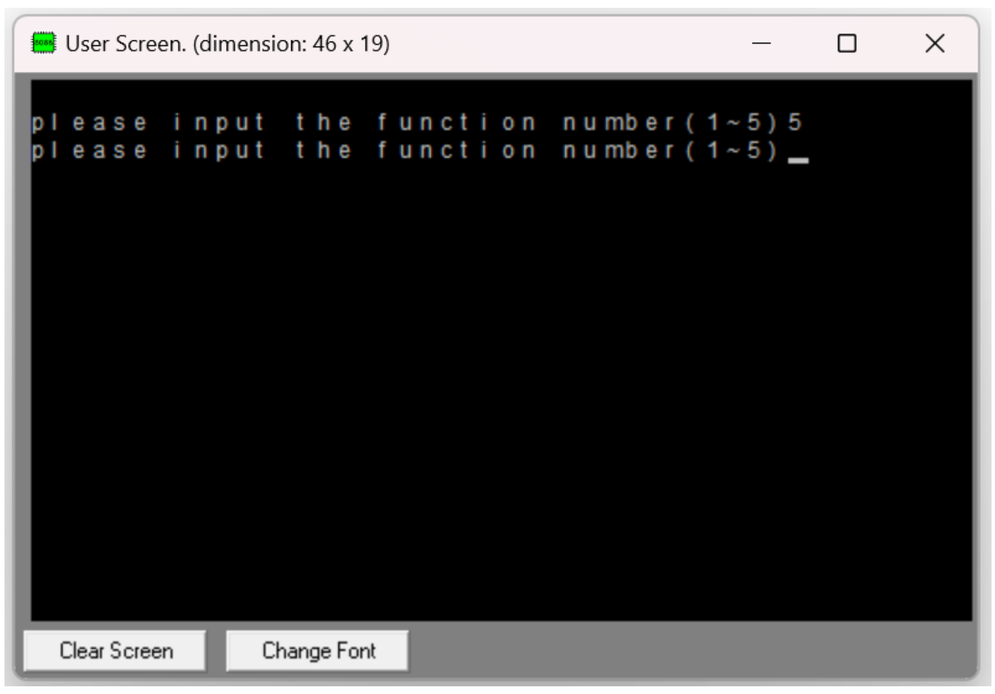

（3） 选择功能 3，对存储在字符串中的数据进行排序，结果如下：



（4） 选择功能 4，显示系统时间，结果如下：




（5） 选择功能 5，回到功能选择，结果如下：




## 五、实验的收获及心得体会

通过本次实验，我对微机原理课程中使用的编码程序软件有了基本的了解，并且掌握了 emu8086 汇编软件的操作方式。通过逐步解决实验题目，我从简单到复杂地掌握了汇编语言的基本语法。在编程过程中，我遇到了各种挑战，包括程序无法执行、结果错误等问题，这些问题让我深刻理解了寄存器之间的关系和数据流的连通性。

通过实验，我能够使用 8086 汇编语言编写简单的程序，并且利用这个软件加深了我对课本知识的理解。在字符处理、数据存储和循环控制等领域的实践，让我对计算机的底层工作原理有了更清晰的认识。通过手动实现字符串操作和系统时间获取，我对中断机制和寄存器操作有了更深的掌握。在实现不同功能的过程中，我对调试和优化汇编代码的技能提出了更高的要求，特别是在数据段和堆栈段的管理上。这些经验不仅提升了我的编程能力，也加深了我对计算机科学的理解。

## 六、实验代码

```
TITLE   8086 Code Template (for EXE file)


;       AUTHOR          WXT
;       DATE            2024.9.11
;       VERSION         1.00
;        FILE           ?.ASM

; 8086 Code Template

; Directive to make EXE output:
      #MAKE_EXE#

DSEG    SEGMENT 'DATA'
; TODO: add your data here!!!!


string_fun  db '1324657' ; 'Xidian University 2024'
            db '$'

string_origin  db 'The original data is: $'


string_main  db 'please input the function number(1~5)$'
string_error db 'Wrong number, please input again:  $'
string_broken db 'The pro is broken, Ple. run again $'



string_fun1_result  db 100 dup(?)
```

```
                db '$'
string_fun1_disresult   db  'The Upper Case is:  $'



string_fun2_result  db 'The maximum is:  $'


string_fun3_disp_result     db 'The sorted data is: $'
string_fun3_hex      db '15243'
                     db '$'
string_fun3_temp     dw  ?
                     db '$'
string_fun3_result   db  100 dup(?)
                     db  '$'


string_fun4_info     db 'please press anykey to display the time $'
string_fun4  db  'Now the time is:$'
string_time       db 2 dup(0)
           db ':'
           db 2 dup(0)
           db ':'
           db 2 dup(0)
           db '$'


;string_input_character db 'please input character $'
;string_input_error     db 'error please input 0~9 a~z A~Z $'
```

```
;string_fun3_info     db 'please input the decimal number $'
;string_fun3_error    db 'input error,please input 0~255 $'
;string_fun3_hex      db 100 dup(?)
;                     db '$'
;string_fun3_temp     dw  ?
;                     db '$'
;string_fun3_result   db  100 dup(?)
;                     db  '$'
;string_hexcopy       db  100 dup(?)
;                     db  '$'
;string_fun4_info     db 'please press anykey to display the time $'

DSEG    ENDS

SSEG    SEGMENT STACK   'STACK'
        DW      100h    DUP(?)
TOP     LABEL WORD
SSEG    ENDS



CSEG    SEGMENT 'CODE'
        ASSUME  CS:CSEG, DS:DSEG, ES:DSEG, SS:SSEG
;*****************************************



START:
```

```
; set segment registers:
    MOV     AX, DSEG
    MOV     DS, AX
    MOV     ES, AX
    MOV     AX, SSEG
    MOV     SS, AX
    LEA     SP, TOP


; TODO: add your code here!!!!
  MAIN_FUNCTION:
       MOV     AH, 02H
       MOV     DL, 0DH     ; 0d:回车  0a: 换行换到下面位置
       INT     21H
       MOV     AH, 02H
       MOV     DL, 0AH
       INT     21H

       LEA     DX, string_main  ;   显示一段字符串  地址:   DS: DX
       MOV     AH, 09H
       INT     21H

       MOV     AH,  01H    ;       输入一个数字（其 ascii 码会输入到 AL），
选择子程序
            INT     21H

            CMP     AL, 31H     ;      分析是不是 1-5，如果不是，报错；如果是，
调用响应子函数
```

```
        JB      DISP_INPUT_ERR
        CMP     AL, 35H
        JA      DISP_INPUT_ERR
        JMP     FUNC_SEL
DISP_INPUT_ERR:
        MOV     AH,  02H    ;   回车
        MOV     DL,  0DH
        INT     21H
        MOV     AH, 02H
        MOV     DL, 0AH
        INT     21H
        LEA     DX,  string_error
        MOV     AH,  09H
        INT     21H
        JMP      MAIN_FUNCTION


FUNC_SEL:
        CMP  AL,  32H
        JB   FUNC_1
        JE   FUNC_2
        JMP  FUNC_345

FUNC_345:
        CMP  AL,  34H
        JB   FUNC_3
        JE   FUNC_4
        JMP  FUNC_5
```

```
FUNC_1:
        CALL   FUNCTION_1
        JMP    MAIN_FUNCTION

FUNC_2:
        CALL   FUNCTION_2
        JMP    MAIN_FUNCTION

FUNC_3:
        CALL   FUNCTION_3
        JMP    MAIN_FUNCTION

FUNC_4:
        CALL   FUNCTION_4
        JMP    MAIN_FUNCTION

FUNC_5:
        CALL   FUNCTION_5
        JMP    MAIN_FUNCTION

 ;    INPUT YOUR FUNCTION 1   HERE
   FUNCTION_1 PROC NEAR
   LEA SI, string_fun        ; 源字符串的起始地址
   LEA DI, string_fun1_result ; 转换后字符串的存储地址

COUNTER:
    LODSB
    ; 检查是否到达字符串末尾（$符号）
    CMP AL, '$'
```

```
    JE DONE


    ; 判断字符是否为小写字母
    CMP AL, 'a'
    JB NO_CHANGE ; 如果小于'a', 则不是小写字母
    CMP AL, 'z'
    JA NO_CHANGE ; 如果大于'z', 则不是小写字母

    ; 小写字母转换为大写字母
    SUB AL, 32

NO_CHANGE:
    STOSB
    JMP COUNTER ; 处理下一字符

DONE:
    ; 在目标字符串末尾添加结束字符'$'
    MOV [DI], '$'

    ; 换行
    MOV     AH, 02H
    MOV     DL, 0DH
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0AH
    INT     21H

    ; 输出提示信息并换行
    LEA     DX, string_origin
```

```
    MOV     AH, 09H
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0DH
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0AH
    INT     21H

    ; 输出原字符串并换行
    LEA     DX, string_fun
    MOV     AH, 09H
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0DH
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0AH
    INT     21H

    ; 输出转换提示信息并换行
    LEA DX, string_fun1_disresult
    MOV AH, 09H
    INT 21H
    MOV     AH, 02H
    MOV     DL, 0DH
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0AH
```

```
    INT     21H


    ; 输出转换后字符串
    LEA DX, string_fun1_result
    MOV AH, 09H
    INT 21H


    ; 清除寄存器信息
    MOV AX,0
    MOV BX,0
    MOV CX,0
    MOV DX,0
    ; 子程序返回
    RET
FUNCTION_1 ENDP


 ;    INPUT YOUR FUNCTION 2 HERE
    FUNCTION_2 PROC NEAR


   LEA SI, string_fun
   MOV CX, 0


LENGTH:
   ; 检查当前字符是否为终止字符 '$'
   LODSB
   CMP AL, '$'
   JE DONE_CALCULATE ; 如果相等, 则跳转到 DONE_CALCULATING


   ; 否则增加 CX 寄存器的值
```

```
    INC CX
    JMP LENGTH ;循环记数

DONE_CALCULATE:
    LEA SI, string_fun
    MOV DL, [SI]
    INC SI
    DEC CX
LCMP:
    CMP DL, [SI]
    JA NOCHNG
    MOV DL, [SI]

NOCHNG:
    INC SI
    LOOP LCMP  ; 找出字符串中最大字符，放入 DL
    MOV BL,DL
DISPLAY:
    ; 换行
    MOV     AH, 02H
    MOV     DL, 0DH
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0AH
    INT     21H

    ; 输出提示信息并换行
    LEA     DX, string_origin
    MOV     AH, 09H
```

```
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0DH
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0AH
    INT     21H

    ; 输出原字符串并换行
    LEA     DX, string_fun
    MOV     AH, 09H
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0DH
    INT     21H
    MOV     AH, 02H
    MOV     DL, 0AH
    INT     21H

    ; 输出字符串中的最大值提示信息
    LEA     DX, string_fun2_result
    MOV     AH, 09H
    INT     21H

    ; 输出字符串中的最大值
    MOV DL, BL
    MOV AH, 02H
    INT 21H
```

```
    ; 清除寄存器信息
        MOV AX,0
    MOV BX,0
    MOV CX,0
        MOV DX,0


    RET
    FUNCTION_2      ENDP

;    INPUT YOUR FUNCTION 3 HERE
    FUNCTION_3 PROC NEAR

        ; 换行输出提示信息
         MOV      AH,2
         MOV      DL,0DH
         INT      21H
         MOV      AH,2
         MOV      DL,0AH
         INT      21H
    LEA DX, string_origin
    MOV AH,09H
    INT 21H

     ; 计算数组串长度
        LEA SI, string_fun
        MOV CX, 0

    CALCULATE:
        LODSB
```

```
        CMP AL, '$'
        JE END
        INC CX
        JMP CALCULATE

    END:
        LEA SI, string_fun
        PUSH CX
        MOV BX,CX
        MOV AX,0

    ORIGION:
       MOV AH,02H ; 打印原字符串中字符
       MOV DL,[SI]
       INT 21H
       INC SI
       CMP CX,1
       JE PRE
       MOV DL,','
       INT 21H
       LOOP ORIGION

    ; 排序准备
    PRE:
       POP CX
        DEC CX
        LEA SI, string_fun
        ADD SI, CX
        ; SI 指向末尾字符
```

```
    ; 冒泡排序
LP1:
    PUSH CX
    PUSH SI
LP2:
    MOV AL,[SI]
    CMP [SI-1],AL
    JAE NOXCHG
    XCHG AL,[SI-1]
    MOV [SI],AL
NOXCHG:
    DEC SI
    LOOP LP2
    POP SI
    POP CX
    LOOP LP1

    MOV AX,0
    ; 显示提示信息
    MOV     AH,2
    MOV     DL,0DH
    INT     21H
    MOV     AH,2
     MOV     DL,0AH
     INT     21H
    LEA DX, string_fun3_disp_result
    MOV AH,09H
     INT 21H
```

```
    LEA SI,string_fun
     MOV CX,BX
     MOV AX,0


RESULT:
    MOV AH,02H ; 分别打印排序后字符
    MOV DL,[SI]
     INT 21H
     INC SI
    CMP CX,1
    JE EXIT
    MOV DL,','
     INT 21H
    LOOP RESULT


 EXIT:
MOV AX,0
    MOV BX,0
    MOV CX,0
    MOV DX,0


        RET
 FUNCTION_3      ENDP


;    INPUT YOUR FUNCTION 4 HERE
    FUNCTION_4 PROC NEAR


    ; 换行显示提示信息
```

```
        MOV     AH,2
         MOV     DL,0DH
         INT     21H
         MOV     AH,2
         MOV     DL,0AH
         INT     21H
        LEA DX, string_fun4_info
        MOV AH,09H
        INT 21H

        ; 输入任意字符后获取系统时间
         MOV     AH,01H
         INT     21H

        ; 获取系统时间 CL:小时 DX:分和秒的时钟计数总数
        MOV AH,00H
        INT 1AH

        PUSH DX
        push CX

         ; 换行显示提示信息
        MOV     AH,2
         MOV     DL,0DH
         INT     21H
         MOV     AH,2
         MOV     DL,0AH
         INT     21H
```

```
LEA DX, string_fun4
MOV AH,09H
INT 21H

POP CX
; 将CL中的内容转化为十进制显示在屏幕
MOV AH,0
MOV AL,CL
MOV BL,10
DIV BL ; AH(余数)中存放小时数低位，AL(商)中存放小时数高位
MOV BL,AH
ADD AL,30H
ADD BL,30H
MOV AH,02H ; 显示小时数和冒号
MOV DL,AL
INT 21H
MOV AH,02H
MOV DL,BL
INT 21H
MOV AH,02H
MOV DL,':'
INT 21H

; 取出时钟数
POP DX

; 根据时钟数获取分钟秒钟
MOV CX,DX
MOV DX,0
```

```
MOV AX,0
MOV AX,CX
MOV BX,18
DIV BX ; AX 当中保留分钟时钟总秒数
MOV DX,0 ; 去除余数
MOV BL,60
DIV BL ; AL 中保留分钟数, AH 中保留秒钟数

; 显示分钟数
MOV CX,0
MOV CH,AH ; CH 中存放秒钟数
MOV CL,AL
MOV AH,0
MOV AL,CL
MOV BL,10 ;将分钟数转为十进制
DIV BL
 MOV BL,AH
ADD AL,30H
ADD BL,30H

MOV AH,02H ; 显示
MOV DL,AL
INT 21H
MOV AH,02H
MOV DL,BL
INT 21H
MOV AH,02H
MOV DL,':'
INT 21H
```

```
; 显示秒钟数
MOV DX,0
MOV AX,0
MOV BX,0
MOV BL,10 ;将秒钟数转为十进制
MOV AL,CH
DIV BL ;
 MOV BL,AH
ADD AL,30H
ADD BL,30H

MOV AH,02H ; 显示
MOV DL,AL
INT 21H
MOV AH,02H
MOV DL,BL
INT 21H


MOV AX,0
MOV BX,0
MOV CX,0
MOV DX,0
    RET
FUNCTION_4      ENDP
```

```
;    INPUT YOUR FUNCTION 5HERE
    FUNCTION_5 PROC NEAR

          JMP  MAIN_FUNCTION:

        RET
FUNCTION_5      ENDP


    HLT
    CSEG    ENDS

    END    START    ; set entry point.
```