**微机原理与接口技术上机作业**

****

学生姓名： 章星宇

学 号： 19200300029

班 级： 1920031

提交日期：2021年11月20日

目录

[一、 实验要求 1](#_Toc2383)

[二、 实验环境 1](#_Toc7310)

[2.1 软件环境 1](#_Toc4635)

[2.2 硬件环境 1](#_Toc1286)

[三、 问题分析 2](#_Toc19773)

[3.1 功能一分析 2](#_Toc32420)

[3.2 功能二分析 2](#_Toc17785)

[3.3 功能三分析 2](#_Toc13387)

[3.4 功能四分析 2](#_Toc24722)

[3.5 功能五分析 3](#_Toc26998)

[四、 解决方法 3](#_Toc32208)

[4.1 整体设计和基本功能 3](#_Toc27422)

[4.1.1 实验整体设计 3](#_Toc15828)

[4.1.2 获取用户的输入 4](#_Toc26817)

[4.1.3 异常输入的检测 4](#_Toc26107)

[4.1.4 字符串的显示 4](#_Toc32444)

[4.1.5 换行显示功能 5](#_Toc9177)

[4.2 功能一：字符串小写转大写 5](#_Toc17493)

[4.3 功能二：在字符中找最大值 6](#_Toc22935)

[4.4 功能三：输入数据组的排序 8](#_Toc803)

[4.4.1 十进制数转换成十六进制数 8](#_Toc8760)

[4.4.2 输入数据的递增排序 9](#_Toc10941)

[4.5 功能四：右上角显示系统时间 11](#_Toc29970)

[4.5.1 系统时间的获取 11](#_Toc27348)

[4.5.2 时间位置的确定 11](#_Toc18941)

[4.6 功能五：结束程序的运行 11](#_Toc28450)

[五、 遇到的问题及调试 11](#_Toc27652)

[5.1 问题一：代码冗长问题 11](#_Toc21464)

[5.2 问题二：程序错误问题 12](#_Toc22177)

[六、 程序运行效果 12](#_Toc24173)

[6.1 功能一运行效果 12](#_Toc6595)

[6.2 功能二运行效果 12](#_Toc9729)

[6.3 功能三运行效果 13](#_Toc25401)

[6.4 功能四运行效果 13](#_Toc28588)

[七、 附录 13](#_Toc50)

# 实验要求

编写程序实现下列5项功能，通过从键盘输入1~5进行菜单式选择。

#### 按数字键“1”，完成将字符串中的小写字母变换成大写字母。用户输入由英文大小写字母或数字0~9组成的字符串（以回车结束），变换后按下列格式在屏幕上显示： <原字符串>：abcdgyt0092 <新字符串>：ABCDGYT0092 按任意键重做；按Esc键返回主菜单

#### 按数字键“2”，完成在字符中找最大值。用户输入由英文大小写字母或数字0~9组成的字符串（以回车结束），找出最大值后按下列格式在屏幕上显示： <原字符串>The maximum is <最大值>. 按任意键重做；按Esc键返回主菜单。

#### 按数字键“3”，完成输入数据组的排序。用户输入一组十进制数值（小于255），然后变换成十六进制数，并按递增方式进行排序，按下列格式在屏幕上显示： <原数值串> <新数值串> 按任意键重做；按Esc键返回主菜单。

#### 按数字键“4”，完成时间的显示。在屏幕的右上角实时显示出时间HH:MM:SS。 按任意键重做；按Esc键返回主菜单。

#### 按数字键“5”，结束程序的运行，返回操作系统。

# 实验环境

## 软件环境

操作系统：Windows10

编辑器：emu8086

## 硬件环境

CPU：i7-10750H

RAM：32GB

# 问题分析

## 功能一分析

功能一要求完成字符串大小写的转换，同时需输出原字符串和处理后的字符串。对功能进行分解，需要完成以下几个子任务：

1. 键盘字符的读取和存储
2. 异常输入的检测
3. 字符串的显示
4. 大小写的转换

## 功能二分析

功能二要求在字符中找最大值，同时需输出原字符串和字符中的最大值。对功能进行分解，需要完成以下几个子任务：

1. 键盘字符的读取和存储
2. 异常输入的检测
3. 字符串的显示
4. 比较字符并记录字符最大值

## 功能三分析

功能三要求将输入的十进制数转换成十六进制数输出，并对这些数按递增方式进行排序输出。对功能进行分解，需要完成以下几个子任务：

1. 键盘字符的读取和存储
2. 异常输入的检测
3. 字符串的显示
4. 十进制数转换成十六进制数
5. 输入数据的递增排序

## 功能四分析

功能四要求将获取系统当前时间，并将该时间显示在屏幕的右上角。对功能进行分解，需要完成以下两个子任务：

1. 系统时间的获取
2. 时间位置的确定

## 功能五分析

功能五要求结束程序的运行，返回操作系统。该功能实现较为简单，通过DOS的中断调用即可实现。

# 解决方法

## 整体设计和基本功能

### 实验整体设计

本实验采用的是交互式的程序设计，即初始化为一个菜单，可以根据用户的输入分别选择5项功能。同时，当每一项功能结束时，用户可以选择重新尝试或者返回主菜单。

具体的流程如下图所示：

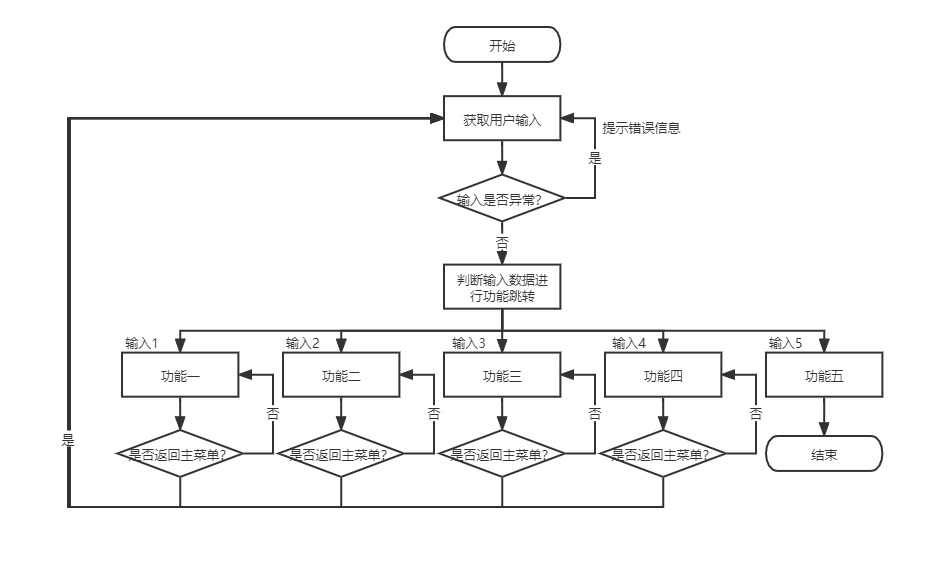


图 1总体流程图

### 获取用户的输入

总体流程中以及各个子任务中都需获取用户的输入并进行存储，这项功能的实现使用了DOS的中断调用。由于其需频繁使用，因此在程序中，将其封装成宏定义函数get\_char，其功能是获取单个输入字符并存储在AL中。

相关代码：

get\_char macro

    MOV    AH,1

**INT**    21h

endm

### 异常输入的检测

在每次获取用户输入后，都需要对输入数据进行检测，若输入异常值则需要提示错误信息并使用户重新输入。

例如，在主菜单中，用户的输入只能为1-5，输入其他值会报错，该部分可由下面的代码实现：

CMP    AL,31h

JB     main\_input\_error

CMP    AL,35h

JA     main\_input\_error

JMP    select

由于字符在计算机中使用ASCII码来进行存储。因此使用CMP指令，将输入字符的ASCII码和目标字符的ASCII码进行比较。其中31h为数字“1”的ASCII码，35h为数字“5”的ASCII码。若输入字符的ASCII码小于31h或大于35h，则跳转到错误提示代码块(main\_input\_error)，否则进入选择代码块(select)。

### 字符串的显示

在程序中，提示字符串与处理结果的字符串都需要进行显示。由于这项功能也需要被频繁使用，因此在程序中将其封装为宏定义函数print\_string，相关代码如下：

print\_string macro string

    LEA    DX,string

    MOV    AH,09h

**INT**    21h

endm

该函数的传入参数为string，通过DOS的中断调用，可以在屏幕上显示出DS：DX地址处的字符串。

### 换行显示功能

在程序运行中，为了使显示界面更为美观，需要频繁使用换行显示的功能，因此在程序中也将其封装为宏定义函数，相关代码如下：

print\_line macro

    MOV    AH,2      ;调用2号功能系统，将DL的内容显示到屏幕

    MOV    DL,0DH    ;0DH是回车符的ASCII码

**INT**    21h

    MOV    AH,2

    MOV    DL,0AH    ;0AH是换行符的ASCII码

**INT**    21h

endm

换行的本质是打印出一串空行，这里通过显示回车符和换行符的方式实现换行这一效果。

## 功能一：字符串小写转大写

在问题分析中，明确了功能一需要完成以下四个子任务：

1. 键盘字符的读取和存储
2. 异常输入的检测
3. 字符串的显示
4. 大小写的转换

其中，前三个子任务已经在宏定义中实现，因此本功能只要完成第四个任务。

考虑到字符在计算机中是以ASCII码的方式进行存储，而大小写字母之间的ASCII码相差20H，因此，只需将所有小写字母的ASCII码减去20H，即可转换成大写字母。

核心代码如下：

cmp\_data:

    ;CL用来记录剩余的处理字符数量，若CL无空值，直接跳出

    CMP    CL,00h

    JE     task1\_finish

    MOV    AL,[SI]

    ;只有小写字母才进行转换，若不是小写字母直接输出

    CMP    AL,61h

    JB     cmp\_data\_finish

    SUB    AL,20h    ;小写转大写：小写字母-20H

cmp\_data\_finish:

    ;[DI]用来存储处理后的字符数据

    MOV    [DI],AL

    INC    SI

    DEC    CL ;处理完一个字符，CL-1

    INC    DI

    ;CL不为0，代表没处理完

    JMP    cmp\_data

其中，CL用来记录剩余的处理字符数量，SI为原字符存储的变址索引，DI为处理后的字符存储的变址索引。

功能一的流程图如下所示：

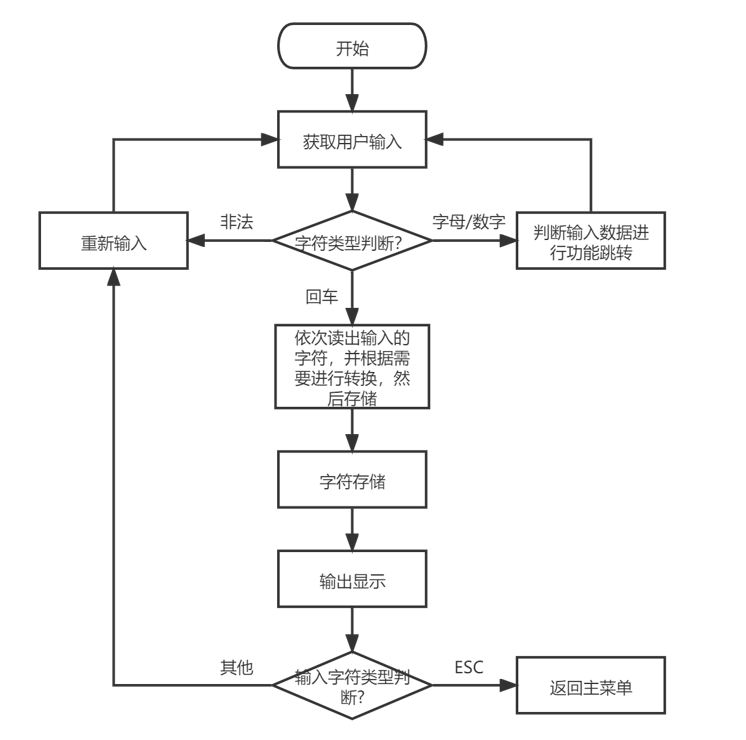


图2功能一流程图

## 功能二：在字符中找最大值

在问题分析中，明确了功能二需要完成以下四个子任务：

1. 键盘字符的读取和存储
2. 异常输入的检测
3. 字符串的显示
4. 比较字符并记录字符最大值

其中，前三个子任务已经在宏定义中实现，因此本功能只要完成第四个任务。

本程序采用遍历的思路进行比较。在获取每个字符之后，通过源变址寄存器SI对原字符串进行遍历，比较每个字符的ASCII码，初始化最大值为第一个字符，若后面的存在字符比当前最大值大，则最大值替换为该字符。最后，对源字符串和目标字符结尾加入字符串结束标记“$”，调用宏定义函数进行输出。

比较过程的核心代码如下：

func\_2\_cmp\_data:

    ;假如已经遍历完毕则进行跳出

    CMP    CL,00h

    JE     func2\_finish

    MOV    AL,[SI]

    MOV    AH,[DI]

    ;假如AL>AH，进行跳转，将AL的值存入[DI]，即最大值

    CMP    AL,AH

    JA     func\_2\_cmp\_data\_finish     ;JA:无符号大于则跳转

    INC    SI

    DEC    CL

    JMP    func\_2\_cmp\_data

功能二的流程图如下所示：

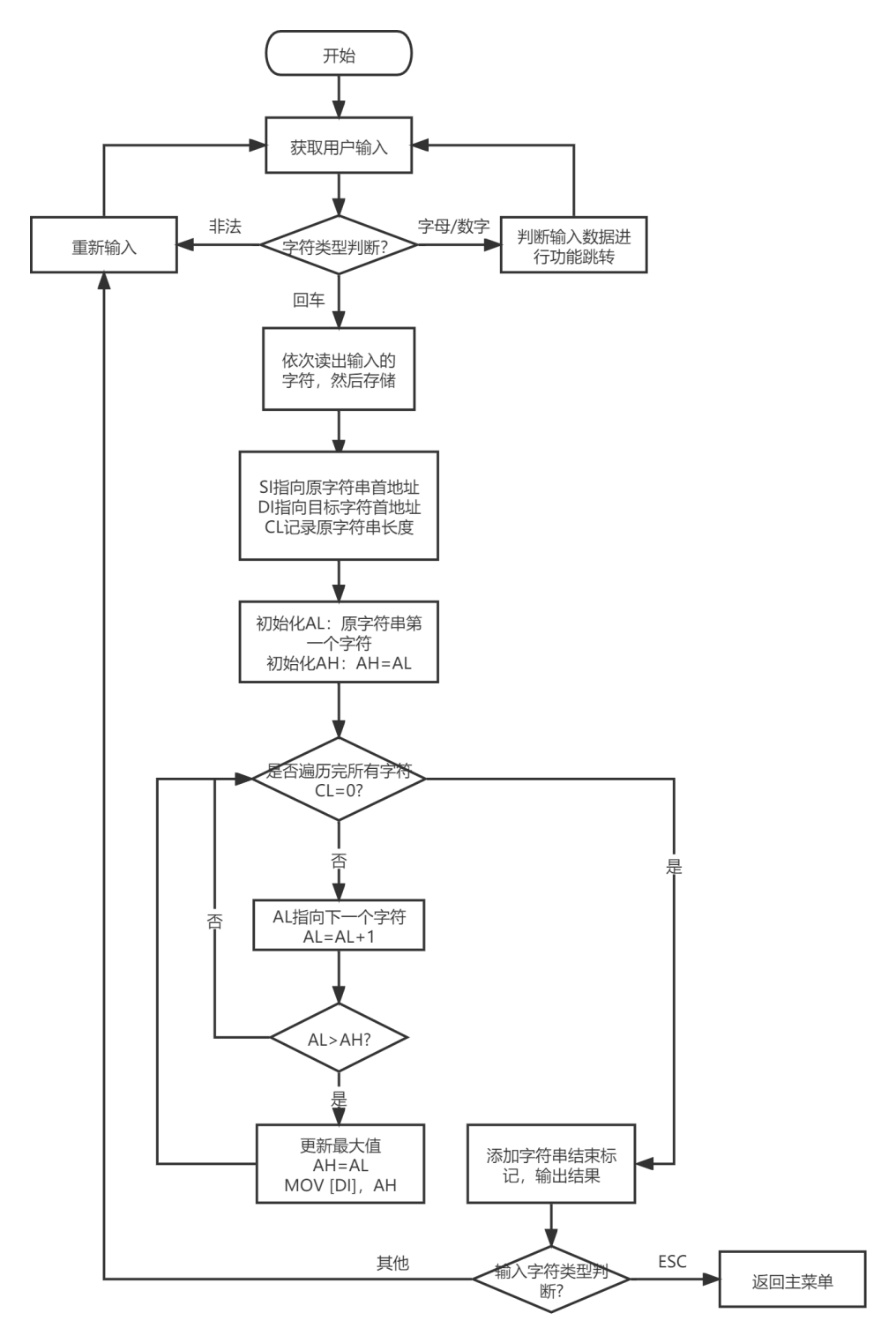


图3功能二流程图

## 功能三：输入数据组的排序

在问题分析中，明确了功能三需要完成以下五个子任务：

1. 键盘字符的读取和存储
2. 异常输入的检测
3. 字符串的显示
4. 十进制数转换成十六进制数
5. 输入数据的递增排序

其中，前三个子任务已经在宏定义中实现，因此本功能只要完成后两个任务。

### 十进制数转换成十六进制数

考虑到用户输入的数字在计算机内部存储为ASCII码，因此第一步必须将ASCII码转换成十进制数。数字“0”的ASCII码为30H，因此将输入的0-9的ASCII码-30H即可提取到对应的十进制数，也可以将0-9的ASCII码的高位清零，直接获得对应的十进制数。高位清零操作可以用当前数据和OFH取逻辑与来实现。

实际上，寄存器是以十六进制来进行数据的存储，因此当数字仅有一位时，十六进制和十进制完全一样。因此只需考虑多位数据的情况。这一功能中，不同的数据采用空格进行分隔，在检测到空格之前，每一个数据先进行存储，当存在下一位数据时，之间的数据\*10再加上下一位数据，这样就实现了十进制数向十六进制数的转换。

相关代码如下所示：

MOV    DL,0

MOV    DH,10

task\_3\_real\_store:

    get\_char

    CMP    AL,0DH   ;判断是否按下回车键

    JE     task3\_next\_step

    CMP    AL,20H   ;判断是否按下空格键

    JE     task3\_save\_num

    ;小于0或大于9跳转报错

    CMP    AL,30H ;30H为"0"的ASCII码

    JB     task3\_error\_input

    CMP    AL,39H ;39H为"9"的ASCII码

    JA     task3\_error\_input

    MOV    BH,0

    ;这里首先需要将ASCII码转换成十进制数，0的ASCII码为30H，因此可以用ASCII码-30H或者将高位清零

    AND    AL,0FH ;高4位清零

    MOV    BL,AL

    MOV    AL,DL

    MUL    DH     ;每一个高位\*10

    ADD    AX,BX

    MOV    DL,AL

    JMP    task\_3\_real\_store

### 输入数据的递增排序

为了将输入数据进行递增排序，本程序运用了双层循环的选择排序方法。

相关代码如下：

sort:

    LEA    SI,string\_task3\_count

    MOV    CL,[SI]

    LEA    SI,string\_task3

;外循环

fori:

    LEA    DI,string\_task3\_count

    MOV    CH,CL

    MOV    DI,SI

    INC    DI

;内循环

forj:

    MOV    BL,[SI]

    MOV    BH,[DI]

    CMP    BL,BH

    JBE    afterswap

    MOV    [SI],BH

    MOV    [DI],BL

afterswap:

    DEC    CH

    INC    DI

    CMP    CH,1

    JA     forj

    INC    SI

    DEC    CL

    CMP    CL,1

    JA     fori

第i轮外循环，BL指向原字符串第i个字符，外循环的次数由CH来确定。第j轮内循环，BH指向原字符串第j个字符，j初始值为i+1，内循环的次数由CL来确定。每一次内循环中，第i个字符分别与其后面的字符一一比较，若BL>BH，则进行两数交换。第一次外循环确定第一位最小值，第二次外循环确定第二位最小值，直到所有位置的最小都确定，从而实现递增排序。

功能三的流程图如下所示：

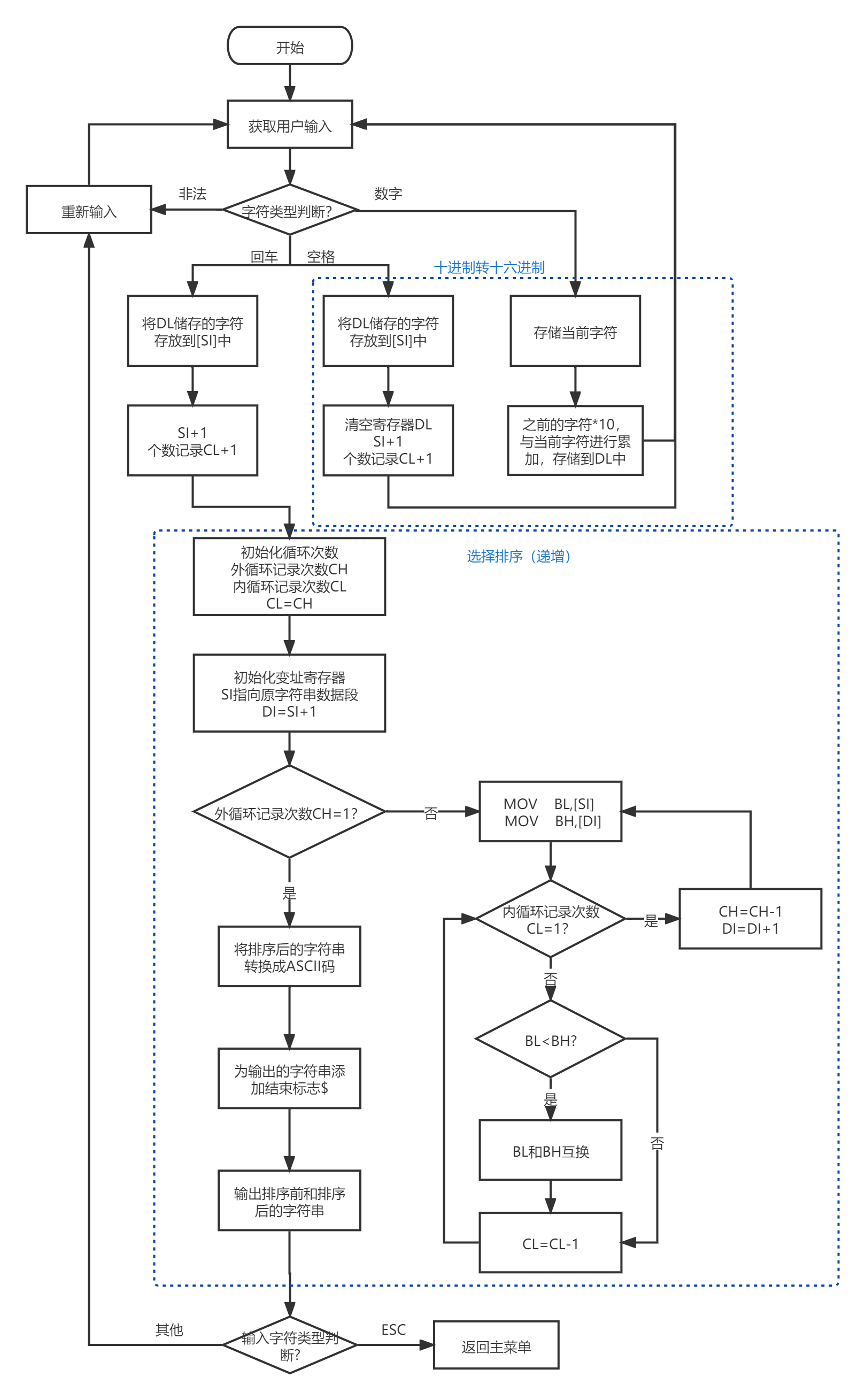


图 4功能3流程图

## 功能四：右上角显示系统时间

在问题分析中，明确了功能四需要完成以下两个子任务：

1. 系统时间的获取
2. 时间位置的确定

### 系统时间的获取

本程序使用DOS中断INT 21H的2CH号功能来获取当前系统时间，小时值会存放在CH中，分值会存放在CL中，秒值会存放在DH中。

相关代码：

MOV AH,2Ch     ;CH=时,CL=分,DH=秒

**INT** 21h

### 时间位置的确定

本程序使用DOS中断INT 10H的2H号功能来设置光标显示在右上角。其中，DH代表行号，DL代表列号。

相关代码：

MOV DH,0  ;行号

MOV DL,72  ;列号

MOV BH,0

MOV AH,2

**INT** 10h

## 功能五：结束程序的运行

本程序使用DOS中断INT 21H的4CH号功能来实现结束程序运行。

相关代码：

MOV AH, 4CH

**INT** 21h

# 遇到的问题及调试

## 问题一：代码冗长问题

在编程中，遇到重复的代码多次使用导致整体代码冗长的问题，例如字符获取、字符显示、换行等操作，需使用重复的代码。

解决方法：通过宏定义的方式将重复用到的代码进行封装，从而大大减小代码冗余量。

## 问题二：程序错误问题

在编写功能三的时候，初次运行时，递增后的字符和输入的字符不相同。

解决方法：通过调试，发现错误原因是在功能三中，使用了DL寄存输入的字符数据，而在中途使用换行语句时，调用了INT 21H的2号功能，导致DL发生变化。因此，在换行前后通过入栈出栈的方式来保护DL数值，从而使问题得到解决。

# 程序运行效果

下面是各个功能的运行效果截图。

## 功能一运行效果

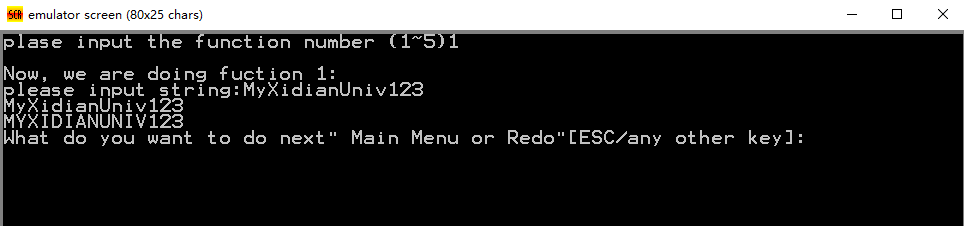


图 5功能1效果图

## 功能二运行效果

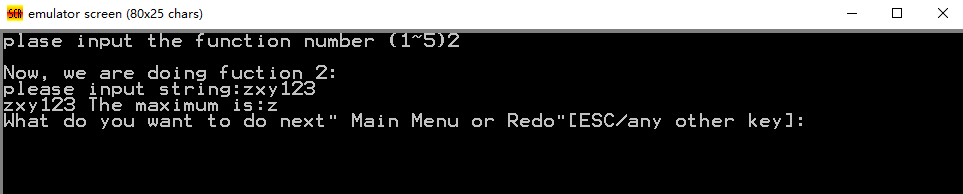


图 6功能2效果图

## 功能三运行效果

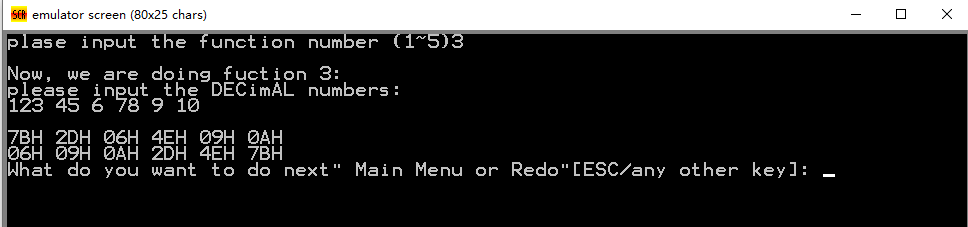


图 7功能3效果图

## 功能四运行效果

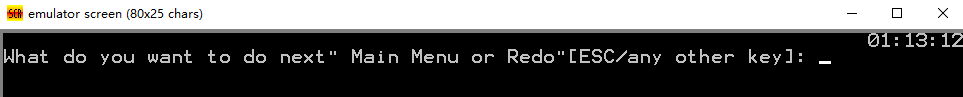


图 8功能4效果图

# 附录

1. DATAS SEGMENT
2. ;此处输入数据段代码
3. string\_main db 'plase input the function number (1~5)$'
4. tip DB 'please input string:','$'
5. string\_task2\_result2 db 'The maximum is :$'
6. string\_error db 'Wrong number, please input again. $'
7. string\_task3\_info db 'please input the DECimAL numbers:',0DH,0AH,'$'
8. again\_or\_return db 'What do you want to do next" Main Menu or Redo"[ESC/any other key]: $'
9. task1\_info db 'Now, we are doing fuction 1: $'
10. task2\_info db 'Now, we are doing fuction 2: $'
11. task3\_info db 'Now, we are doing fuction 3: $'
12. ; emu8086不支持dup(?) 因此用0来初始化
13. string\_task1 db 100 dup(0)
14. string\_task1\_result db 100 dup(0)
15. string\_task2 db 100 dup(0)
16. string\_task2\_result db 10 dup(0)
17. string\_task3 db 100 dup(0)
18. string\_task3\_count db 10 dup(0)
19. string\_task3\_result db 100 dup(0)
20. hour db 0
21. minute db 0
22. second db 0
23. time db "00:00:00$"
24. len equ $-time
25. num db 0
27. DATAS ENDS
29. STACKS SEGMENT
30. TOP LABEL **WORD**
31. ;此处输入堆栈段代码
32. DW      100H    DUP(0)
33. STACKS ENDS
35. ;换行
36. print\_line macro
37. MOV    AH,2      ;调用2号功能系统，将DL的内容显示到屏幕
38. MOV    DL,0DH    ;0DH是回车符的ASCII码
39. **INT**    21h
40. MOV    AH,2
41. MOV    DL,0AH    ;0AH是换行符的ASCII码
42. **INT**    21h
43. endm
45. ;调用DOS功能，该功能为显示DS：DX地址处的字符
46. print\_string macro string
47. LEA    DX,string
48. MOV    AH,09h
49. **INT**    21h
50. endm
52. ;用来获取单个输入字符
53. get\_char macro
54. MOV    AH,1
55. **INT**    21h
56. endm
58. CODES SEGMENT
59. ASSUME CS:CODES,DS:DATAS,SS:STACKS
60. START:
61. MOV AX,DATAS
62. MOV DS,AX
64. menu:
65. print\_string string\_main
66. get\_char
67. ;只能输入1-5
68. CMP    AL,31h
69. JB     main\_input\_error
70. CMP    AL,35h
71. JA     main\_input\_error
72. JMP    select
74. main\_input\_error:
75. print\_line
76. print\_string string\_error
77. print\_line
78. JMP    menu
80. select:
81. CMP   AL,31h
82. JE    Task1
83. CMP   AL,32h
84. JE    Task2
85. CMP   AL,33h
86. JE    Task3
87. CMP   AL,34h
88. JE    Task4
89. CMP   AL,35h
90. JE    Task5
92. return\_menu:
93. print\_line
94. JMP    menu
96. ;任务一：实现小写转化成大写
97. Task1 PROC
98. print\_line
99. print\_line
100. print\_string task1\_info
101. print\_line
102. print\_string tip
103. LEA    SI,string\_task1
104. MOV    CL,00H;CL用来存储需要处理字符的数量，初始化置0
106. restore\_input:
107. get\_char
108. MOV    BL,AL  ;BL用来存储输入的字符信息，保护数据
109. ;假如当前字符输入是回车(0DH)，则不进行存储，直接跳出
110. CMP    AL,0DH
111. JE     deal\_first
112. ;对空格字符进行存储，单独判断
113. CMP    AL,20H ;20H为"空格"的ASCII码
114. JE     real\_store
115. ;设置输入字符的范围，只能存储数字、大小写字母，否则报错
116. CMP    AL,30H ;30H为"0"的ASCII码
117. JB     input\_error
118. CMP    AL,39H ;39H为"9"的ASCII码
119. JBE    real\_store
120. CMP    AL,41H ;41H为"A"的ASCII码
121. JB     input\_error
122. CMP    AL,5AH ;5AH为"Z"的ASCII码
123. JBE    real\_store
124. CMP    AL,61H ;61H为"a"的ASCII码
125. JB     input\_error
126. CMP    AL,7AH ;7AH为"z"的ASCII码
127. JBE    real\_store
129. ;错误跳转信息，并重新提示输入
130. input\_error:
131. print\_line
132. print\_string string\_error
133. print\_line
134. print\_string tip
135. LEA    SI,string\_task1
136. MOV    CL,00H;CL用来存储需要处理字符的数量，初始化置0
137. JMP    restore\_input
139. ;存储当前字符
140. real\_store:
141. MOV    AL,BL ;将BL存储的字符重新赋值给AL
142. MOV    [SI],AL ;将AL存储的字符传输到[SI]进行存储
143. INC    SI
144. INC    CL
145. INC    DI
146. ;未检测到结束标志“回车”，返回上一步继续存储下一个字符
147. JMP    restore\_input
149. ;重新回到存储的首地址位置，对每一个字符进行处理
150. deal\_first:
151. LEA    SI,string\_task1
152. LEA    DI,string\_task1\_result
154. cmp\_data:
155. ;CL用来记录剩余的处理字符数量，若CL无空值，直接跳出
156. CMP    CL,00h
157. JE     task1\_finish
158. MOV    AL,[SI]
159. ;只有小写字母才进行转换，若不是小写字母直接输出
160. CMP    AL,61h
161. JB     cmp\_data\_finish
162. SUB    AL,20h    ;小写转大写：小写字母-20H
164. cmp\_data\_finish:
165. ;[DI]用来存储处理后的字符数据
166. MOV    [DI],AL
167. INC    SI
168. DEC    CL ;处理完一个字符，CL-1
169. INC    DI
170. ;CL不为0，代表没处理完
171. JMP    cmp\_data
173. task1\_finish:
174. print\_line
175. ;为输入和输出字符串添加结束标志
176. MOV    [SI],'$'
177. MOV    [DI],'$'
178. ;这里显示输入
179. print\_string string\_task1
180. print\_line
181. ;这里显示输出结果
182. print\_string string\_task1\_result
183. print\_line
185. ;判断返回菜单或者重试
186. print\_string again\_or\_return
187. get\_char
188. CMP    AL,1BH   ;判断是否按下esc
189. JE     return\_menu
191. JMP    Task1
193. Task1 ENDP
195. ;任务二：在字符中找最大值
196. Task2 PROC
197. print\_line
198. print\_line
199. print\_string task2\_info
200. print\_line
201. print\_string tip
202. LEA    SI,string\_task2
203. MOV    CL,00H;CL用来存储需要处理字符的数量，初始化置0
205. task\_2\_restore\_input:
206. get\_char
207. MOV    BL,AL  ;BL用来存储输入的字符信息，保护数据
208. ;假如当前字符输入是回车(0DH)，则不进行存储，直接跳出
209. CMP    AL,0DH
210. JE     task2\_dealfirst
212. ;设置输入字符的范围，只能存储数字、大小写字母，否则报错
213. CMP    AL,30H ;30H为"0"的ASCII码
214. JB     task2\_input\_error
215. CMP    AL,39H ;39H为"9"的ASCII码
216. JBE    task2\_store
217. CMP    AL,41H ;41H为"A"的ASCII码
218. JB     task2\_input\_error
219. CMP    AL,5AH ;5AH为"Z"的ASCII码
220. JBE    task2\_store
221. CMP    AL,61H ;61H为"a"的ASCII码
222. JB     task2\_input\_error
223. CMP    AL,7AH ;7AH为"z"的ASCII码
224. JBE    task2\_store
226. task2\_store:
227. MOV    AL,BL
228. MOV    [SI],AL
229. INC    SI
230. INC    CL
231. JMP    task\_2\_restore\_input
233. ;错误跳转信息，并重新提示输入
234. task2\_input\_error:
235. print\_line
236. print\_string string\_error
237. print\_line
238. print\_string tip
239. JMP    Task2
241. task2\_dealfirst:
242. LEA    SI,string\_task2
243. LEA    DI,string\_task2\_result
244. MOV    AL,[SI]
245. MOV    [DI],AL   ;[DI]用来存储输入的数据
246. INC    SI
247. DEC    CL
249. ;遍历比较
250. func\_2\_cmp\_data:
251. ;假如已经遍历完毕则进行跳出
252. CMP    CL,00h
253. JE     func2\_finish
254. MOV    AL,[SI]
255. MOV    AH,[DI]
256. ;假如AL>AH，进行跳转，将AL的值存入[DI]，即最大值
257. CMP    AL,AH
258. JA     func\_2\_cmp\_data\_finish     ;JA:无符号大于则跳转
259. INC    SI
260. DEC    CL
261. JMP    func\_2\_cmp\_data
263. func\_2\_cmp\_data\_finish:
264. MOV    [DI],AL
265. INC    SI
266. DEC    CL
267. JMP    func\_2\_cmp\_data
269. func2\_finish:
270. print\_line
271. MOV    [SI],'$'
272. INC    DI
273. MOV    [DI],'$'
274. ;先输出原字符串
275. print\_string string\_task2
276. ;补一个空格进行分隔
277. MOV    DL,20h
278. MOV    AH,2
279. **INT**    21H
280. print\_string string\_task2\_result2 ;the maximum is :$
281. print\_string string\_task2\_result
282. print\_line
284. ;判断返回菜单或者重试
285. print\_string again\_or\_return
286. get\_char
287. CMP    AL,1BH   ;判断是否按下esc
288. JE     return\_menu
290. JMP    Task2
292. Task2 ENDP
294. ;任务三：输入数据组的排序
295. Task3 PROC
296. print\_line
297. print\_line
298. print\_string task3\_info
299. print\_line
300. print\_string string\_task3\_info
301. LEA    SI,string\_task3
302. MOV    CL,0
303. MOV    BH,0
304. MOV    DI,0
306. task3\_store:
307. MOV    DL,0
308. MOV    DH,10
310. task\_3\_real\_store:
311. get\_char
312. CMP    AL,0DH   ;判断是否按下回车键
313. JE     task3\_next\_step
314. CMP    AL,20H   ;判断是否按下空格键
315. JE     task3\_save\_num
316. ;小于0或大于9跳转报错
317. CMP    AL,30H ;30H为"0"的ASCII码
318. JB     task3\_error\_input
319. CMP    AL,39H ;39H为"9"的ASCII码
320. JA     task3\_error\_input
321. MOV    BH,0
322. ;这里首先需要将ASCII码转换成十进制数，0的ASCII码为30H，因此可以用ASCII码-30H或者将高位清零
323. AND    AL,0FH ;高4位清零
324. MOV    BL,AL
325. MOV    AL,DL
326. MUL    DH     ;每一个高位\*10
327. ADD    AX,BX
328. MOV    DL,AL
329. JMP    task\_3\_real\_store
331. task3\_error\_input:
332. print\_line
333. print\_string string\_error
334. JMP    Task3
336. ;检测到空格则进行一次数值保存
337. task3\_save\_num:
338. CMP    BH,0
339. JNZ    task3\_before\_store
340. MOV    [SI],DL
341. INC    SI
342. INC    CL
344. task3\_before\_store:
345. MOV    BH,1
346. JMP    task3\_store
348. ;输出回车之后
349. task3\_next\_step:
350. ;注意，换行会改变DL数值，因此必须用堆栈保护DL，栈仅能对字节进行操作
351. PUSH   DX
352. print\_line
353. POP    DX
354. MOV    [SI],DL
355. INC    SI
356. INC    CL
357. LEA    DI,string\_task3\_count
358. MOV    [DI],CL
360. task3\_finish:
361. LEA    DI,string\_task3\_count
362. MOV    CL,[DI]
364. LEA    DI,string\_task3
365. PUSH   DX
366. print\_line
367. POP    DX
369. ;将十六进制数转换成ASCII码显示输出
370. task3\_write:
371. MOV    BH,[DI]
372. MOV    DL,BH
373. AND    DL,0F0H ;低位清零
374. SHR    DL,4    ;逻辑右移，将DL右移4位
375. CMP    DL,0AH
376. JGE    task3\_writehighchar
377. ADD    DL,30H  ;数值0-9要进行显示需转换成ASCII码，需+30H
378. JMP    task3\_writehigh
380. ;用来写当前十六进制数的高位
381. task3\_writehighchar:
382. ADD    DL,37H  ;数值A-F要进行显示需转换成ASCII码，需+37H
384. task3\_writehigh:
385. MOV    AH,2
386. **INT**    21H
387. MOV    DL,BH
388. AND    DL,00FH ;高位清零
389. CMP    DL,0AH
390. JGE    task3\_writelowchar    ;JGE大于等于
391. ADD    DL,30H
392. JMP    task3\_writelow
394. ;写当前十六进制数的低位
395. task3\_writelowchar:
396. ADD    DL,37H
398. task3\_writelow:
400. MOV    AH,2
401. **INT**    21H
402. ;每输出一个十六进制数在后面添加H标识
403. MOV    DL,48H
404. MOV    AH,2
405. **INT**    21H
406. ;每输出一个十六进制数显示空格进行分隔
407. MOV    DL,20h
408. MOV    AH,2
409. **INT**    21H
411. INC    DI
412. DEC    CL
413. CMP    CL,0   ;CL记录剩余处理的字符数量，假如不为0，继续跳转运行
414. JNE    task3\_write
416. ;采用选择排序实现十六进制从小到大输出
417. ;BL用来存储第i位，BH用来存储第i+1,i+2,i+3...位，每次外循环确定第i位最小值
418. sort:
419. LEA    SI,string\_task3\_count
420. MOV    CL,[SI]
421. LEA    SI,string\_task3
423. ;外循环
424. fori:
425. LEA    DI,string\_task3\_count
426. MOV    CH,CL
427. MOV    DI,SI
428. INC    DI
429. ;内循环
430. forj:
431. MOV    BL,[SI]
432. MOV    BH,[DI]
433. CMP    BL,BH
434. JBE    afterswap
435. MOV    [SI],BH
436. MOV    [DI],BL
438. afterswap:
439. DEC    CH
440. INC    DI
441. CMP    CH,1
442. JA     forj
444. INC    SI
445. DEC    CL
446. CMP    CL,1
447. JA     fori
449. task3\_finish2:
450. LEA    DI,string\_task3\_count
451. MOV    CL,[DI]
452. LEA    DI,string\_task3
453. PUSH   DX
454. print\_line
455. POP    DX
457. ;输出排序后的结果
458. task3\_write2:
459. MOV    BH,[DI]
460. MOV    DL,BH
461. AND    DL,0F0H ;低位清零
462. SHR    DL,4    ;逻辑右移，将DL右移4位
463. CMP    DL,0AH
464. JGE    task3\_writehighchar2
465. ADD    DL,30H  ;数值0-9要进行显示需转换成ASCII码，需+30H
466. JMP    task3\_writehigh2
468. task3\_writehighchar2:
469. ADD    DL,37H  ;数值A-F要进行显示需转换成ASCII码，需+37H
471. task3\_writehigh2:
472. MOV    AH,2
473. **INT**    21H
474. MOV    DL,BH
475. AND    DL,00FH ;高位清零
476. CMP    DL,0AH
477. JGE    task3\_writelowchar2    ;JGE大于等于
478. ADD    DL,30H
479. JMP    task3\_writelow2
481. ;写当前十六进制数的低位
482. task3\_writelowchar2:
483. ADD    DL,37H
485. task3\_writelow2:
486. MOV    AH,2
487. **INT**    21H
488. ;每输出一个十六进制数在后面添加H标识
489. MOV    DL,48H
490. MOV    AH,2
491. **INT**    21H
492. ;每输出一个十六进制数显示空格进行分隔
493. MOV    DL,20h
494. MOV    AH,2
495. **INT**    21H
496. INC    DI
497. DEC    CL
498. CMP    CL,0             ;CL记录剩余处理的字符数量，假如不为0，继续跳转运行
499. JNE    task3\_write2
501. PUSH   DX
502. print\_line
503. POP    DX
504. ;判断返回菜单或者重试
505. print\_string again\_or\_return
506. get\_char
507. CMP    AL,1BH   ;判断是否按下esc
508. JE     return\_menu
510. JMP    Task3
512. Task3 ENDP

515. ;任务四：输出系统时间到屏幕右上角
516. Task4 PROC
517. ;首先对窗口清屏
518. clear:
519. MOV AX,CS
520. MOV SS,AX
521. MOV SP,0x7c00  ;set stack and SP
523. MOV AH,0x06
524. MOV AL,0
526. MOV CH,0  ;(0,0)
527. MOV CL,0
528. MOV DH,24  ;(24,79)
529. MOV DL,79
530. MOV BH,0x07 ;黑底白字
531. **INT** 0x10
533. ;设置光标位置
534. setshow:
535. MOV DH,0  ;行号
536. MOV DL,72  ;列号
537. MOV BH,0
538. ;置光标位置
539. MOV AH,2
540. **INT** 10h
542. ;获取系统时间
543. get\_system\_time:
544. ;获取时间
545. MOV AH,2Ch     ;CH=时,CL=分,DH=秒
546. **INT** 21h
547. MOV hour,CH
548. MOV minute,CL
549. MOV second,DH
550. ;时间显示到字符串
551. MOV AX,0
552. MOV AL,hour     ;商:AL,余数:AH
553. MOV num,10
554. DIV num
555. ADD AL,30h
556. MOV time[0],AL
557. ADD AH,30h
558. MOV time[1],AH
560. MOV AX,0
561. MOV AL,minute
562. MOV num,10
563. DIV num
564. ADD AL,30h
565. MOV time[3],AL
566. ADD AH,30h
567. MOV time[4],AH
569. MOV AX,0
570. MOV AL,second
571. MOV num,10
572. DIV num
573. ADD AL,30h
574. MOV time[6],AL
575. ADD AH,30h
576. MOV time[7],AH
578. ;输出系统时间
579. LEA DX, time
580. MOV AH, 9
581. **INT** 21h
583. ;判断返回菜单或者重试
584. print\_string again\_or\_return
585. get\_char
586. CMP    AL,1BH   ;判断是否按下esc
587. JE     return\_menu
589. JMP    setshow
591. Task4 ENDP

594. ;任务五：退出
595. Task5 PROC
596. MOV AH, 4CH
597. **INT** 21h
598. Task5 ENDP
600. CODES ENDS
601. END START