

西安电子科技大学

微机系统综合实验 课程实验报告

实验名称 实验四 综合性汇编程序设计

网络与信息安全学院 2118021 班

姓名 盖乐 学号 21009200991

同作者 _____

实验日期 2023 年 5 月 10 日

成 绩

指导教师评语：

指导教师：

_____年____月____日

一、 实验要求

1. 计算 $S=1+2\times3+3\times4+4\times5+\dots+N(N+1)$ ，直到 $N(N+1)$ 项大于 200 为止。
2. 求 $N!$ 。N 为键盘输入的不大于 8 的正整数。
3. 从键盘输入一行字符（以回车结束），并按字母、数字及其它字符分类计数，最后显示出这 3 个计数结果。
4. 编写一电子钟程序，在屏幕正中按以下格式显示：

YYYY 年 MM 月 DD 日

HH: MM: SS

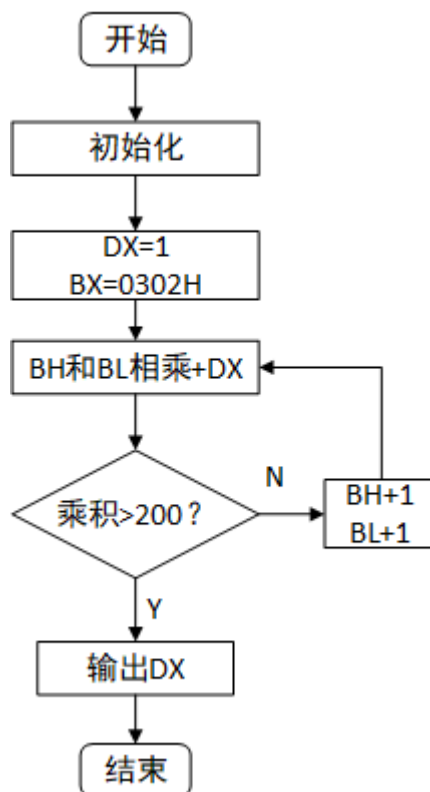
二、 实验目的

综合多种汇编程序设计方法，掌握具有人机交互界面的较复杂的汇编应用程序的编写。

一、 实验代码及实验结果

实验一：

1.实验流程图



2.实验代码

```
1. DATA SEGMENT
2.     STRING1 DB 'S= $' ; 定义字符串常量
3.
4. DATA ENDS
5. CODE SEGMENT
```

```

6.  START:
7.      MOV AX,DATA ; 将数据段地址加载到寄存器 AX 中
8.      MOV DS,AX ; 将数据段寄存器 DS 设置为 AX 中的值
9.      MOV AX,2 ; 将寄存器 AX 设置为 2
10.     MOV BX,AX ; 将寄存器 BX 设置为 AX 中的值
11.     INC BX ; 将寄存器 BX 的值加一, 此时 BX = 3
12.     MOV CX,15 ; 将寄存器 CX 设置为 15
13.     MOV DX,1 ; 将寄存器 DX 设置为 1
14.
15.  FOR:
16.     IMUL BL ; 无符号乘法, 将 AL 与 BL 相乘, 结果存放在 AX 中
17.     CMP AL,200 ; 将 AL 与 200 进行比较
18.     JA OUTSIDE ; 如果 AL 大于 200, 则跳转到 OUTSIDE 标签处
19.     ADD DX,AX ; 将 AX 的值加到 DX 中
20.     MOV AX,DX ; 将 DX 的值复制到 AX 中
21.     MOV AX,BX ; 将 BX 的值复制到 AX 中
22.     INC BX ; 将 BX 的值加一
23.
24.  LOOP FOR
25.
26.  OUTSIDE:
27.     MOV AX,DX ; 将 DX 的值复制到 AX 中
28.     PUSH AX ; 将 AX 的值压入栈中
29.     LEA DX,STRING1 ; 将字符串的偏移地址加载到寄存器 DX 中
30.     MOV AH,09H ; 设置 AH 的值为 09H, 表示显示字符串
31.     INT 21H ; 调用 DOS 功能, 显示字符串
32.     POP AX ; 将栈中的值弹出到 AX 中
33.     CALL PRINT ; 调用 PRINT 子程序
34.     MOV AX,4C00H ; 将程序的返回值设置为 4C00H
35.     INT 21H ; 调用 DOS 功能, 退出程序
36.
37.  CRLF:
38.     PUSH AX ; 将 AX 压入栈中
39.     PUSH DX ; 将 DX 压入栈中
40.     MOV DL,0AH ; 将 DL 的值设置为换行符的 ASCII 值
41.     MOV AH,2H ; 设置 AH 的值为 2H, 表示显示字符
42.     INT 21H ; 调用 DOS 功能, 显示字符
43.
44.     MOV DL,0DH ; 将 DL 的值设置为回车符的 ASCII 值
45.     MOV AH,2H ; 设置 AH 的值为 2H, 表示显示字符
46.     INT 21H ; 调用 DOS 功能, 显示字符
47.     POP DX ; 将栈中的值弹出到 DX 中
48.     POP AX ; 将栈中的值弹出到 AX 中
49.     RET ; 返回调用子程序的位置

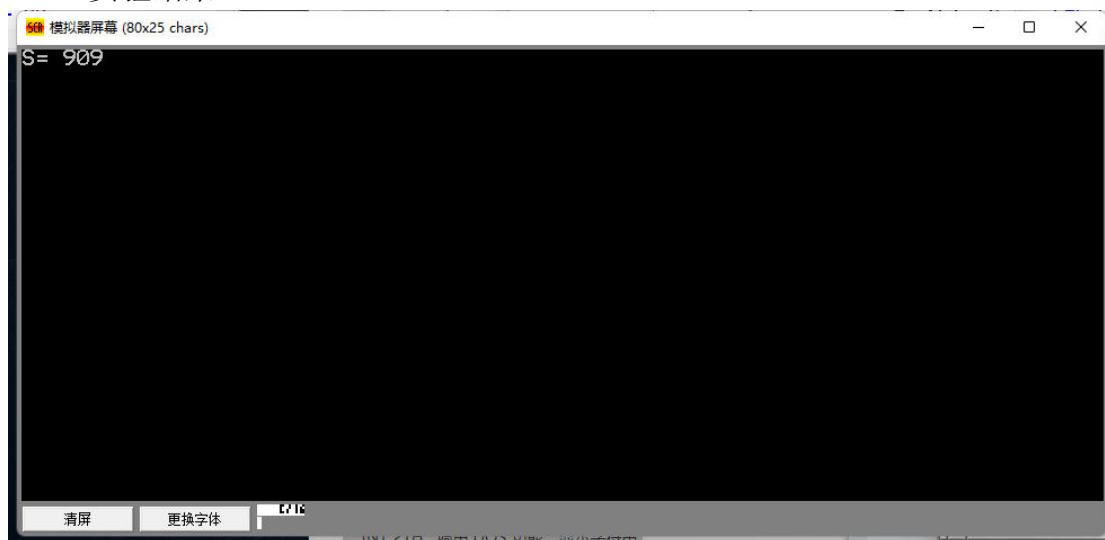
```

```

50.
51. PRINT:
52.     PUSH AX ; 将 AX 压入栈中
53.     MOV CX,0 ; 将 CX 的值设置为 0
54.     MOV BX,10 ; 将 BX 的值设置为 10
55.
56. DISP1:
57.     MOV DX,0 ; 将 DX 的值设置为 0
58.     DIV BX ; 将 AX 的值除以 BX, 商存放在 AL, 余数存放在 DX
59.     PUSH DX ; 将 DX 的值压入栈中
60.     INC CX ; 将 CX 的值加一
61.     OR AX,AX ; 将 AX 与自身进行逻辑或操作
62.     JNE DISP1 ; 如果结果不为零, 跳转到 DISP1 标签处
63.
64. DISP2:
65.     MOV AH,2 ; 设置 AH 的值为 2, 表示显示字符
66.     POP DX ; 将栈中的值弹出到 DX 中
67.     ADD DL, 30H ; 将 DL 的值加上 30H, 将其转换为 ASCII 码
68.     INT 21H ; 调用 DOS 功能, 显示字符
69.
70. LOOP DISP2
71.     POP AX ; 将栈中的值弹出到 AX 中
72.     RET ; 返回调用子程序的位置
73.
74. CODE ENDS
75. END START

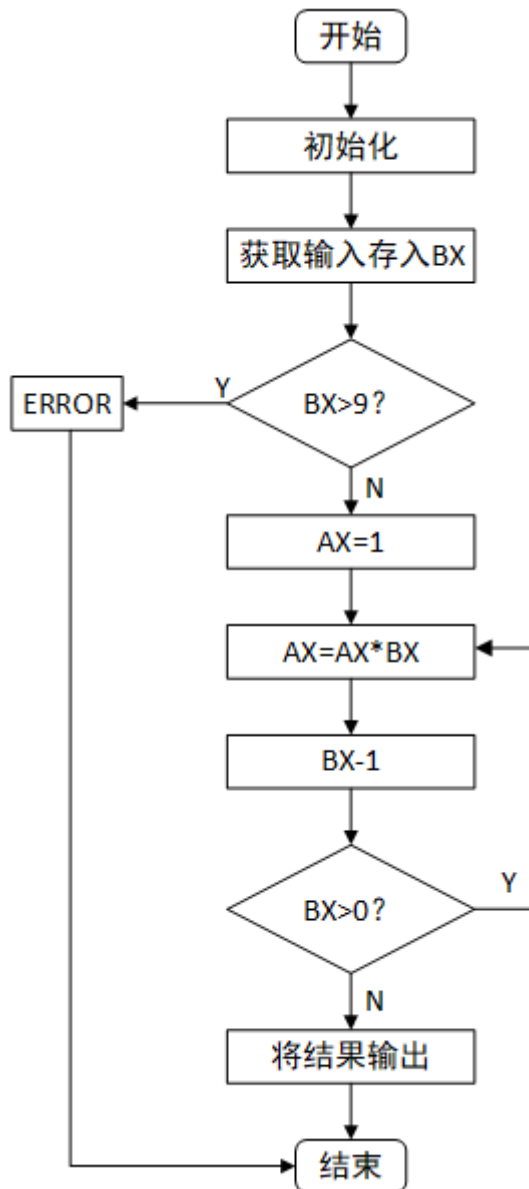
```

3.实验结果



实验二：

1.实验流程图



2.实验代码

```
1. DATA SEGMENT
2.     STRING_1 DB 'Please input an int(< 9):',0AH,0DH,'$' ; 提示用户
   输入一个小于 9 的整数的字符串
3.     STRING_2 DB 0AH,0DH,'ERROR: MUST < 9 !!!',0AH,0DH,'$' ; 错误提
   示字符串
4.     RESULT DB 0AH,0DH,'THE RESULT IS',0AH,0DH,0,0,0,0,0,0,'$' ; 结果
   输出字符串
5. DATA ENDS
6.
7. CODE SEGMENT
8.     ASSUME CS:CODE,DS:DATA
9.
```


```

10. START:
11.     MOV AX,DATA ; 将数据段地址加载到寄存器 AX 中
12.     MOV DS,AX ; 将数据段寄存器 DS 设置为 AX 中的值
13.
14. INPUT:
15.     MOV DX,OFFSET STRING_1 ; 将字符串的偏移地址加载到寄存器 DX 中
16.     MOV AH,09H ; 设置 AH 的值为 09H, 表示显示字符串
17.     INT 21H ; 调用 DOS 功能, 显示字符串
18.     MOV AH,01H ; 设置 AH 的值为 01H, 表示获取键盘输入
19.     INT 21H ; 调用 DOS 功能, 获取键盘输入的字符
20.     SUB AL,30H ; 将输入的 ASCII 码转换为对应的数字
21.     CMP AL,09H ; 比较输入的数字是否大于 9
22.     JA OVER_3 ; 若大于 9, 则跳转到 OVER_3 标签处
23.
24.     XOR DX,DX ; 将 DX 清零
25.     MOV BL,AL ; 将 AL 的值赋给 BL
26.     MOV AX,1 ; 将 AX 的值设置为 1
27.
28. LOOP_1:
29.     MUL BX ; 将 AX 的值乘以 BX 的值
30.     DEC BX ; 将 BX 的值减 1
31.     CMP BX,0 ; 比较 BX 的值是否为 0
32.     JE OVER_1 ; 若为 0, 则跳转到 OVER_1 标签处
33.     JMP LOOP_1 ; 若不为 0, 则跳转到 LOOP_1 标签处
34.
35. OVER_1:
36.     MOV DI,10 ; 将 10 赋给 DI
37.     MOV SI,OFFSET RESULT ; 将 RESULT 的偏移地址赋给 SI
38.     ADD SI,21 ; 将 SI 的值加 2, 指向结果字符串中的空格位置
39.
40. LOOP_2:
41.     DIV DI ; 将 AX 的值除以 DI, 商存放在 AX, 余数存放在 DX
42.     ADD DX,30H ; 将余数加上 30H, 将其转换为 ASCII 码
43.     MOV [SI],DL ; 将余数存放到结果字符串的对应位置
44.     CMP AX,0 ; 比较 AX (商) 的值是否为 0
45.     JE OVER_2 ; 若为 0, 则跳转到 OVER_2 标签处
46.     DEC SI ; 若不为 0, 则将 SI 的值减 1, 指向下一个空格位置
47.     XOR DX,DX ; 将 DX 清零
48.     LOOP LOOP_2 ; 若不为 0, 则跳转到 LOOP_2 标签处
49.
50. OVER_2:
51.     MOV DX,OFFSET RESULT ; 将 RESULT 的偏移地址赋给 DX
52.     MOV AH,09H ; 设置 AH 的值为 09H, 表示显示字符串
53.     INT 21H ; 调用 DOS 功能, 显示字符串

```

```
54.  
55.     JMP ENDD ; 跳转到 ENDD 标签处  
56.  
57. OVER_3:  
58.     MOV DX,OFFSET STRING_2 ; 将错误提示字符串的偏移地址赋给 DX  
59.     MOV AH,09H ; 设置 AH 的值为 09H, 表示显示字符串  
60.     INT 21H ; 调用 DOS 功能, 显示字符串  
61.  
62. ENDD:  
63.  
64. CODE ENDS  
65. END START
```

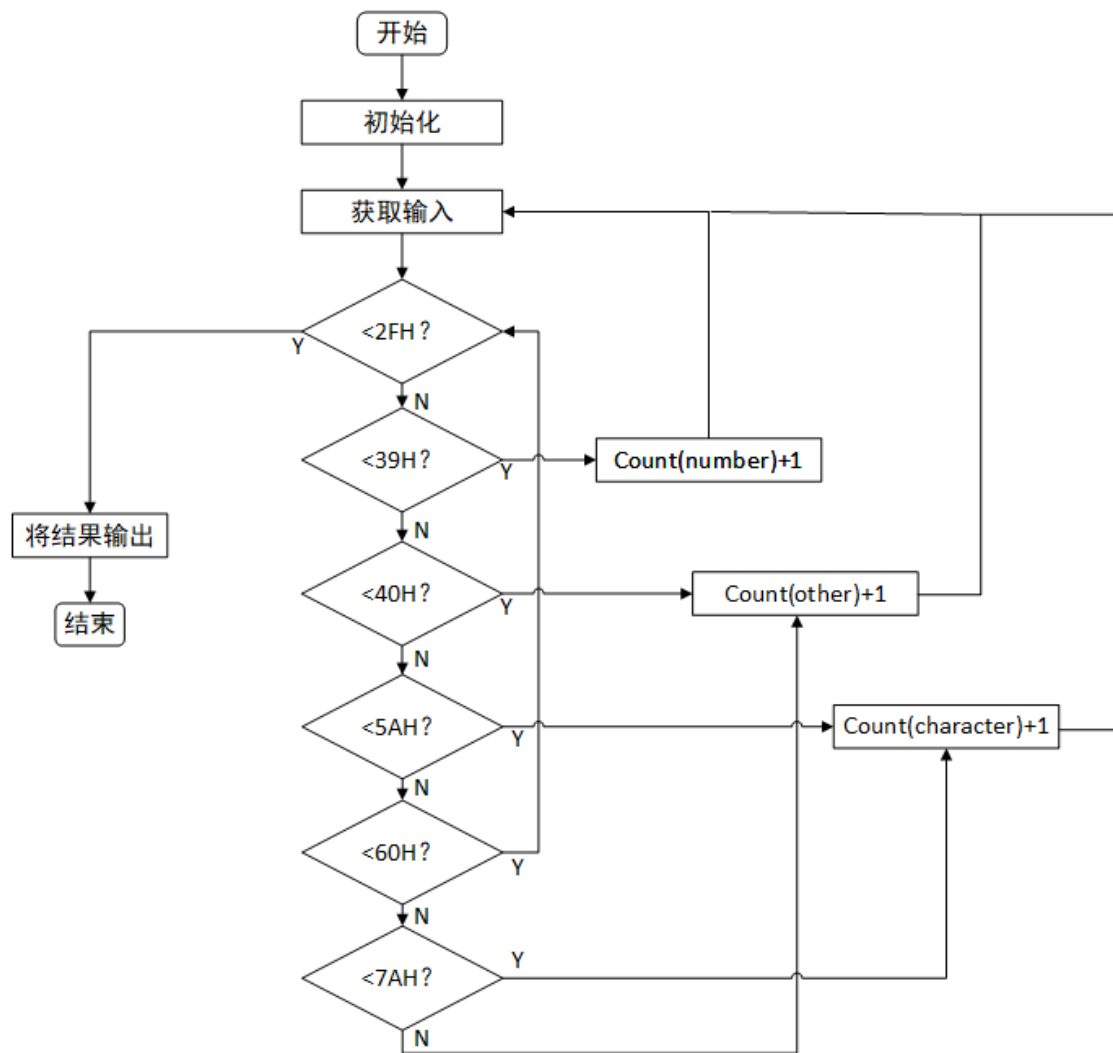
3.实验结果



```
模拟屏幕 (80x25 chars)  
Please input an int(< 9):  
6  
THE RESULT IS  
720  
清屏 更换字体
```

实验三：

1.实验流程图



2.实验代码

1. DATA SEGMENT
2. STRING0 DB 'PLEASE INPUT A STRING: \$' ; 提示用户输入一个字符串的字符串
3. STRING1 DB 'NUMBER OF ALPHABET: \$' ; 字母数量输出字符串
4. STRING2 DB 'NUMBER OF NUMBER: \$' ; 数字数量输出字符串
5. STRING3 DB 'NUMBER OF OTHER CHARACTER: \$' ; 其他字符数量输出字符串
6. NUM_ALPH DB 0 ; 字母数量变量
7. NUM_NUM DB 0 ; 数字数量变量
8. NUM_CHAR DB 0 ; 其他字符数量变量
9. ENDS
- 10.
11. CODE SEGMENT
12. ASSUME CS:CODE,DS:DATA
- 13.


```

14. START:
15.     MOV AX, DATA ; 将数据段地址加载到寄存器 AX 中
16.     MOV DS, AX ; 将数据段寄存器 DS 设置为 AX 中的值
17.
18.     LEA SI, NUM_ALPH ; 将 NUM_ALPH 变量的地址加载到寄存器 SI 中
19.     LEA DX, STRING0 ; 将字符串 0 的偏移地址加载到寄存器 DX 中
20.     MOV AH, 09H ; 设置 AH 的值为 09H, 表示显示字符串
21.     INT 21H ; 调用 DOS 功能, 显示字符串
22.
23.     MOV CX, 40 ; 设置 CX 的值为 40, 表示最多输入 40 个字符
24.
25. INPUT:
26.     MOV AH, 01H ; 设置 AH 的值为 01H, 表示获取键盘输入
27.     INT 21H ; 调用 DOS 功能, 获取键盘输入的字符
28.
29.     CMP AL, 0DH ; 比较输入的字符是否为回车符
30.     JE PRINTF_NUM ; 若为回车符, 则跳转到 PRINTF_NUM 标签处
31.
32.     CMP AL, 48 ; 比较输入的字符与 ASCII 码 48 ('0') 的大小
33.     JB ADD_CHAR ; 若小于 '0', 则跳转到 ADD_CHAR 标签处
34.
35.     CMP AL, 58 ; 比较输入的字符与 ASCII 码 58 ('9'+1) 的大小
36.     JB ADD_NUM ; 若小于 '9'+1, 则跳转到 ADD_NUM 标签处
37.
38.     CMP AL, 65 ; 比较输入的字符与 ASCII 码 65 ('A') 的大小
39.     JB ADD_CHAR ; 若小于 'A', 则跳转到 ADD_CHAR 标签处
40.
41.     CMP AL, 91 ; 比较输入的字符与 ASCII 码 91 ('Z'+1) 的大小
42.     JB ADD_ALPH ; 若小于 'Z'+1, 则跳转到 ADD_ALPH 标签处
43.
44.     CMP AL, 97 ; 比较输入的字符与 ASCII 码 97 ('a') 的大小
45.     JB ADD_CHAR ; 若小于 'a', 则跳转到 ADD_CHAR 标签处
46.
47.     CMP AL, 123 ; 比较输入的字符与 ASCII 码 123 ('z'+1) 的大小
48.     JB ADD_ALPH ; 若小于 'z'+1, 则跳转到 ADD_ALPH 标签处
49.
50.     JMP ADD_CHAR ; 若以上条件都不满足, 则跳转到 ADD_CHAR 标签处
51.
52. INSIDE:
53.     LOOP INPUT ; 循环读取字符, 直到 CX 的值为 0
54.
55. PRINTF_NUM:
56.     CALL CRLF ; 调用换行函数, 显示空行
57.     CALL CRLF ; 调用换行函数, 显示空行

```

```
58.
59.     LEA DX, STRING1 ; 将字符串 1 的偏移地址加载到寄存器 DX 中
60.     MOV AH, 09H ; 设置 AH 的值为 09H, 表示显示字符串
61.     INT 21H ; 调用 DOS 功能, 显示字符串
62.
63.     MOV AL, [SI] ; 将 NUM_ALPH 变量的值加载到寄存器 AL 中
64.     MOV AH, 0 ; 清零寄存器 AH
65.     CALL PRINT ; 调用打印函数, 显示 AL 的值 (字母数量)
66.     CALL CRLF ; 调用换行函数, 显示空行
67.     CALL CRLF ; 调用换行函数, 显示空行
68.
69.     LEA DX, STRING2 ; 将字符串 2 的偏移地址加载到寄存器 DX 中
70.     MOV AH, 09H ; 设置 AH 的值为 09H, 表示显示字符串
71.     INT 21H ; 调用 DOS 功能, 显示字符串
72.
73.     MOV AL, [SI+1] ; 将 NUM_NUM 变量的值加载到寄存器 AL 中
74.     MOV AH, 0 ; 清零寄存器 AH
75.     CALL PRINT ; 调用打印函数, 显示 AL 的值 (数字数量)
76.     CALL CRLF ; 调用换行函数, 显示空行
77.     CALL CRLF ; 调用换行函数, 显示空行
78.
79.     LEA DX, STRING3 ; 将字符串 3 的偏移地址加载到寄存器 DX 中
80.     MOV AH, 09H ; 设置 AH 的值为 09H, 表示显示字符串
81.     INT 21H ; 调用 DOS 功能, 显示字符串
82.
83.     MOV AL, [SI+2] ; 将 NUM_CHAR 变量的值加载到寄存器 AL 中
84.     MOV AH, 0 ; 清零寄存器 AH
85.     CALL PRINT ; 调用打印函数, 显示 AL 的值 (其他字符数量)
86.
87. TAIL:
88.     MOV AX, 4C00H ; 设置 AX 的值为 4C00H, 表示程序正常退出
89.     INT 21H ; 调用 DOS 功能, 退出程序
90.
91. CRLF:
92.     PUSH AX ; 保存 AX 的值
93.     PUSH DX ; 保存 DX 的值
94.     MOV DL, 0AH ; 将换行符的 ASCII 码加载到寄存器 DL 中
95.     MOV AH, 2H ; 设置 AH 的值为 2H, 表示显示字符
96.     INT 21H ; 调用 DOS 功能, 显示字符
97.
98.     MOV DL, 0DH ; 将回车符的 ASCII 码加载到寄存器 DL 中
99.     MOV AH, 2H ; 设置 AH 的值为 2H, 表示显示字符
100.    INT 21H ; 调用 DOS 功能, 显示字符
101.
```

```

102.     POP DX ; 恢复 DX 的值
103.     POP AX ; 恢复 AX 的值
104.     RET ; 返回调用函数的地址
105.
106. PRINT:
107.     PUSH AX ; 保存 AX 的值
108.     MOV CX, 0 ; 清零寄存器 CX, 用于记录位数
109.     MOV BX, 10 ; 将 10 (用于除法) 加载到寄存器 BX 中
110.
111. DISP1:
112.     MOV DX, 0 ; 清零寄存器 DX
113.     DIV BX ; 除以 10, 商保存在 AX 中, 余数保存在 DX 中
114.     PUSH DX ; 将余数压栈
115.     INC CX ; CX 加 1, 表示位数增加了 1
116.     OR AX, AX ; 检查商的值是否为 0
117.     JNE DISP1 ; 若不为 0, 则继续循环
118.
119. DISP2:
120.     MOV AH, 2 ; 设置 AH 的值为 2H, 表示显示字符
121.     POP DX ; 弹出栈顶元素, 即余数
122.     ADD DL, 30H ; 将余数转换为 ASCII 码
123.     INT 21H ; 调用 DOS 功能, 显示字符
124.
125.     LOOP DISP2 ; 继续循环, 显示下一位数字
126.
127.     POP AX ; 恢复 AX 的值
128.     RET ; 返回调用函数的地址
129.
130. ADD_ALPH:
131.     PUSH DX ; 保存 DX 的值
132.     MOV DL, [SI] ; 将 NUM_ALPH 变量的值加载到寄存器 DL 中
133.     INC DL ; 将 DL 的值加 1, 表示字母数量增加了 1
134.     MOV [SI], DL ; 将修改后的值存回 NUM_ALPH 变量
135.     POP DX ; 恢复 DX 的值
136.     JMP INSIDE ; 跳转到 INSIDE 标签处
137.
138. ADD_NUM:
139.     PUSH DX ; 保存 DX 的值
140.     MOV DL, [SI+1] ; 将 NUM_NUM 变量的值加载到寄存器 DL 中
141.     INC DL ; 将 DL 的值加 1, 表示数字数量增加了 1
142.     MOV [SI+1], DL ; 将修改后的值存回 NUM_NUM 变量
143.     POP DX ; 恢复 DX 的值
144.     JMP INSIDE ; 跳转到 INSIDE 标签处
145.

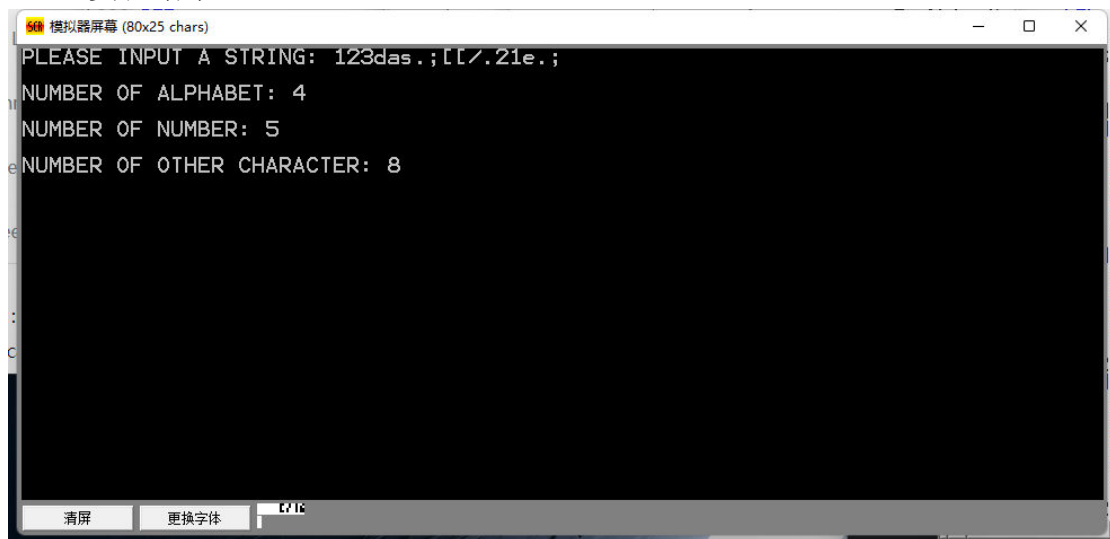
```

```

146. ADD_CHAR:
147.     PUSH DX ; 保存 DX 的值
148.     MOV DL, [SI+2] ; 将 NUM_CHAR 变量的值加载到寄存器 DL 中
149.     INC DL ; 将 DL 的值加 1, 表示其他字符数量增加了 1
150.     MOV [SI+2], DL ; 将修改后的值存回 NUM_CHAR 变量
151.     POP DX ; 恢复 DX 的值
152.     JMP INSIDE ; 跳转到 INSIDE 标签处
153.
154. ENDS
155. END START

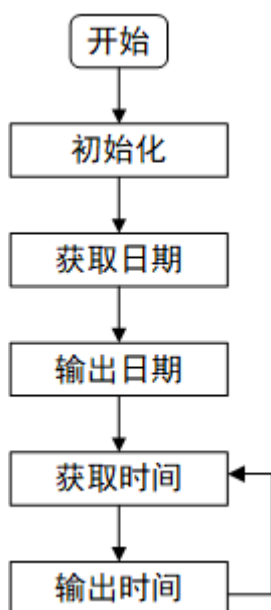
```

3.实验结果



实验四:

1.实验流程图



2.实验代码

1. DATA SEGMENT; 数据段定义开始
2. YEAR DB 0AH,0AH,0AH,0AH,0AH,0AH,0,0,0,0,0,0,0,0,'/',0,0,'/',0,0,
0AH,0DH,'\$'; 年份数据
3. TIME DB 0,0,0,0,0,0,0,':',0,0,':',0,0,0DH,'\$'; 时间数据
4. DATA ENDS; 数据段定义结束
- 5.
6. STACK SEGMENT; 栈段定义开始
7. STA DB DUP(10); 定义名为 STA 的栈，大小为 10 字节
8. STACK ENDS; 栈段定义结束
- 9.
10. CODE SEGMENT; 代码段定义开始
11. ASSUME CS:CODE,DS:DATA; 段寄存器的关联
- 12.
13. START;; 程序入口
- 14.
15. MOV AX,DATA; 将数据段的基地址放入 AX 寄存器
16. MOV DS,AX ; 将 DS 寄存器设置为数据段基地址
- 17.
18. MOV BL,10; 将寄存器 BL 设置为 10，用于除法操作
- 19.
20. LOOP_1;; 循环 1 标签
- 21.
22. MOV AH,2AH; 设置功能号为 2AH，用于获取日期
23. INT 21H; 调用 21H 中断获取日期
- 24.
25. MOV SI,OFFSET YEAR; 将 YEAR 的偏移地址赋给 SI 寄存器
26. ADD SI,14; 将 SI 偏移 14 个字节，指向年份的最后一个字符
- 27.
28. MOV AX,CX; 将 CX 寄存器的值赋给 AX 寄存器
29. CALL CHANGE_4; 调用 CHANGE_4 过程，将 AX 中的数字转换为 4 位十进制
- 30.
31. ADD SI,7; 将 SI 偏移 7 个字节，指向月份的最后一个字符
32. MOV AL,DH; 将 DH 寄存器的值赋给 AL 寄存器
33. CALL CHANGE_2; 调用 CHANGE_2 过程，将 AL 中的数字转换为 2 位十进制
- 34.
35. ADD SI,5; 将 SI 偏移 5 个字节，指向日期的最后一个字符
36. MOV AL,DL; 将 DL 寄存器的值赋给 AL 寄存器
37. CALL CHANGE_2; 调用 CHANGE_2 过程，将 AL 中的数字转换为 2 位十进制
- 38.
39. MOV AH,09H; 设置功能号为 09H，用于显示字符串
40. MOV DX,OFFSET YEAR; 将 YEAR 的偏移地址放入 DX 寄存器
41. INT 21H; 调用 21H 中断显示年份
- 42.

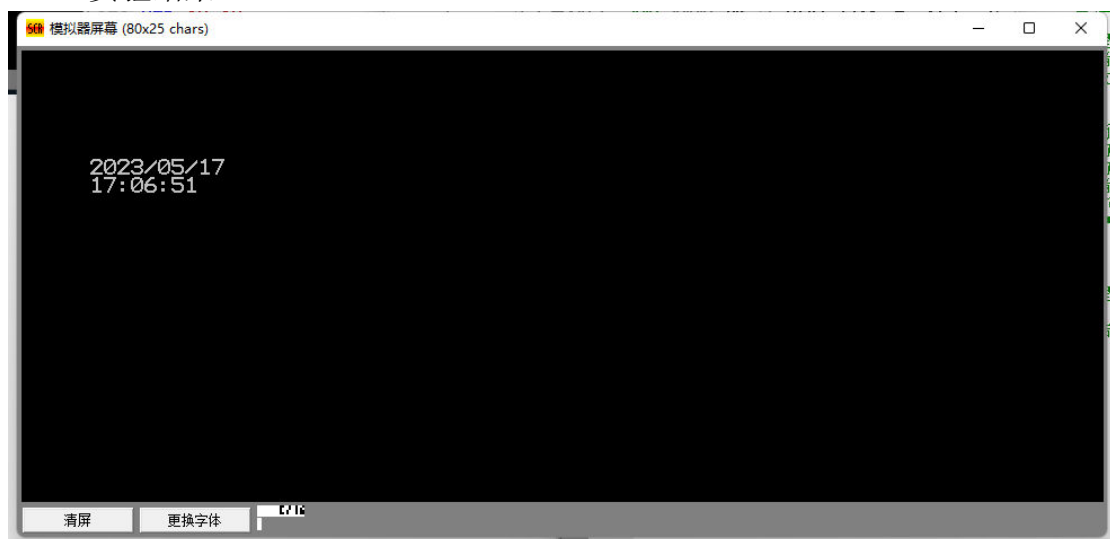
```
43. LOOP_4:; 循环 4 标签
44.
45. MOV AH,2CH; 设置功能号为 2CH, 用于获取时间
46. INT 21H; 调用 21H 中断获取时间
47.
48. MOV SI,OFFSET TIME; 将 TIME 的偏移地址赋给 SI 寄存器
49. PUSH CX; 保存 CX 寄存器的值
50. ADD SI,6; 将 SI 偏移 6 个字节, 指向时钟的最后一个字符
51. MOV AL,CH; 将 CH 寄存器的值赋给 AL 寄存器
52. CALL CHANGE_2; 调用 CHANGE_2 过程, 将 AL 中的数字转换为 2 位十进制
53.
54. ADD SI,5; 将 SI 偏移 5 个字节, 指向分钟的最后一个字符
55. POP CX; 恢复 CX 寄存器的值
56. MOV AL,CL; 将 CL 寄存器的值赋给 AL 寄存器
57. CALL CHANGE_2; 调用 CHANGE_2 过程, 将 AL 中的数字转换为 2 位十进制
58.
59. ADD SI,5; 将 SI 偏移 5 个字节, 指向秒的最后一个字符
60. MOV AL,DH; 将 DH 寄存器的值赋给 AL 寄存器
61. CALL CHANGE_2; 调用 CHANGE_2 过程, 将 AL 中的数字转换为 2 位十进制
62.
63. MOV AH,09H; 设置功能号为 09H, 用于显示字符串
64. MOV DX,OFFSET TIME; 将 TIME 的偏移地址放入 DX 寄存器
65. INT 21H; 调用 21H 中断显示时间
66.
67. JMP LOOP_4; 无条件跳转到循环 4 标签
68.
69. CHANGE_4 PROC; CHANGE_4 过程开始
70. MOV CX,4; 将 CX 寄存器设置为 4
71.
72. LOOP_2:; 循环 2 标签
73. DIV BL; 除法操作, 商存入 AL 寄存器, 余数存入 AH 寄存器
74. ADD AH,30H ; 将 AH 寄存器的值加上 30H, 转换为 ASCII 码
75. MOV [SI],AH; 将 AH 寄存器的值存入 SI 所指向的内存单元
76. XOR AH,AH; 将 AH 寄存器清零
77. DEC SI; 将 SI 减 1, 指向前一个字符
78. LOOP LOOP_2; 循环, 如果 CX 不为零则继续执行
79.
80. RET
81. CHANGE_4 ENDP; CHANGE_4 过程结束
82.
83. CHANGE_2 PROC; CHANGE_2 过程开始
84. XOR AH,AH; 将 AH 寄存器清零
85. MOV CX,2 ; 将 CX 寄存器设置为 2
86.
```

```

87. LOOP_3:; 循环 3 标签
88. DIV BL; 除法操作, 商存入 AL 寄存器, 余数存入 AH 寄存器
89. ADD AH,30H ; 将 AH 寄存器的值加上 30H, 转换为 ASCII 码
90. MOV [SI],AH; 将 AH 寄存器的值存入 SI 所指向的内存单元
91. XOR AH,AH ; 将 AH 寄存器清零
92. DEC SI; 将 SI 减 1, 指向前一个字符
93. LOOP LOOP_3; 循环, 如果 CX 不为零则继续执行
94.
95. RET
96.
97. CHANGE_2 ENDP; CHANGE_2 过程结束
98.
99. CODE ENDS ; 代码段定义结束
100.
101. END START; 程序结束

```

3.实验结果



四、 实验总结

通过本次综合性汇编程序设计实验,我在编写汇编语言方面取得了显著进步。在第四个实验中,我遇到了一个问题:输出指针无法返回上一行,导致只能刷新时间而无法刷新日期。我尝试了多种方法,但由于刷新页面会导致输出框闪烁,频繁刷新页面效果不理想。为了解决这个问题,我采取了一种每次循环获取日期进行检查的方法。如果发现日期发生改变,就重新刷新页面进行输出。这样可以避免频繁刷新页面,但会增加程序的开销。另外,我还考虑过每日重启程序的方

式，这样可以减小程序的开销。在解决问题的过程中，我积累了许多新的知识和经验，对汇编语言的编写能力有了很大的提高。