Homework-1-Report

姓名：项 枫　　学号：2022211570

一、代码

完整代码如下（其中高亮部分为我填充的内容）：

```
1.  .arch i386
2.  .intel_syntax noprefix
3.
4.  .data
5.  str_hint_input:
6.      .asciz "Enter a string (up to 10 letters) in the input dialog win
    dow. \n\n"
7.  str_format_input:
8.      .asciz "%s"
9.  str_hint_output:
10.     .asciz "Transformed string:\n\n%s"
11.
12. str_hint_error_letter:
13.     .asciz "The string contains invalid characters.\n"
14. str_hint_error_length:
15.     .asciz "The string is too long. \n"
16.
17. .text
18.     .globl  _main
19.
20. _main:
21.     mov %ebp, %esp #for correct debugging
22.     push    ebp     # 开辟函数栈空间 （栈帧）
23.     mov ebp, esp     # 将栈顶位置对齐到栈底
24.     and esp, -16
25.     sub esp, 48
26.
27.     mov DWORD PTR [esp], OFFSET str_hint_input #{*1}
28.     call    _printf   # 打印输入字符串提示语
29.
30.     lea eax, [esp+25]
31.     mov DWORD PTR [esp+4], eax #{*2}
32.     mov DWORD PTR [esp], OFFSET str_format_input
33.     call    _scanf    # 输入字符串: 需提前在"输入"窗口填入要测试的字符
    串, 非运行时填入
34.
```

```
    lea eax, [esp+25]
    mov DWORD PTR [esp], eax
    call    _strlen   # 进行字符串长度计算

    cmp eax,  10   # {*3} 检查字符串是否超过 10
    ja  hit_error_length        # ja 为无符号大于跳转汇编指令

    mov DWORD PTR [esp+44], 1      # 该变量用于记录是否遇到非法字符，初始值置 1
    mov DWORD PTR [esp+40], 0      # 该变量用于记录当前处理的字符在字符串中的位置，初始值置 0
    jmp label_test_whether_end_of_string

label_test_char:     # 检测当前字符是否为英文字母
    lea edx, [esp+25]  # 字符串的起始地址
    mov eax, DWORD PTR [esp+40]  # 当前字符相对字符串起始地址的偏移量
    add eax, edx
    mov al, BYTE PTR [eax]  # 取出当前字符

    movsx   eax, al
    mov DWORD PTR [esp], eax
    mov eax, DWORD PTR __imp__isalpha
    call    eax    # 以当前字符为参数，调用 isalpha 函数判断是否为英文字母

    test    eax, eax
    jne label_test_nxt_char     # 是合法字符，跳转至 label_test_nxt_char

    mov DWORD PTR [esp+44], 0

    jmp label_test_end_or_invalid     # 是非法字符，[esp+44]置 0，跳转至 label_test_end_or_invalid

label_test_nxt_char:
    inc DWORD PTR [esp+40]     # {*4} 指针向后移动一位，指向下一个要检测的字符

label_test_whether_end_of_string:     # 当前字符的下一个位置是不是\0
    lea edx, [esp+25]
    mov eax, DWORD PTR [esp+40]
    add eax, edx
    mov al, BYTE PTR [eax]

```

```
        test al, al   # {*5}
    jne label_test_char     # 下一个字符不是\0，跳转至 label_test_char

label_test_end_or_invalid:
    cmp DWORD PTR [esp+44], 0  # 检查一下[esp+44]的值判断是否含有非法字符
    je  label_print_error_message  # 存在非法字符，跳转至 label_print_error_message 输出错误信息

    mov DWORD PTR [esp+36], 0  # 新的变量指示当前字符在字符串中的位置，初始值置 0，准备重新遍历字符串

    jmp label_modify_char  # 输入的字符串合法，跳转至 label_modify_char

label_modify_lower_or_upper:    # label_modify_lower_or_upper
    lea edx, [esp+25]
    mov eax, DWORD PTR [esp+36]
    add eax, edx
    mov al, BYTE PTR [eax]

    movsx   eax, al
    mov DWORD PTR [esp], eax
    mov eax, DWORD PTR __imp__islower
    call    eax  # 以当前字符为参数，调用 islower 函数判断是否为小写字母

    test    eax, eax
    je  label_modify_upper_to_lower  # 是大写字母，跳转至 label_modify_upper_to_lower

        # 小写字母转大写
    lea edx, [esp+25]
       mov eax, DWORD PTR [esp+36]
       add eax, edx
       mov al, BYTE PTR [eax]

       movsx   eax, al
       mov DWORD PTR [esp], eax
       mov eax, DWORD PTR __imp__toupper
       call    eax  # 以当前字符为参数，调用 toupper 函数转换为大写字母

       mov dl, al
       lea ecx, [esp+25]
       mov eax, DWORD PTR [esp+36]
```

```
112.        add eax, ecx
113.        mov BYTE PTR [eax], dl  # 将转换后的大写字母放入新构建的字符串的对应位置
114.        jmp label_modify_nxt_char
115.
116.    label_modify_upper_to_lower:  # 大写字母转小写 (*** 请完成这段功能代码)
117.        # 提示: 可如下调用 __imp__tolower 函数
118.        # mov   eax, DWORD PTR __imp__tolower
119.        # call  eax  # 以当前字符为参数，调用 tolower 函数转换为小写字母
120.
121.        lea edx, [esp+25]
122.        mov eax, DWORD PTR [esp+36]
123.        add eax, edx
124.        mov al, BYTE PTR [eax]
125.
126.        movsx   eax, al
127.        mov DWORD PTR [esp], eax
128.        mov eax, DWORD PTR __imp__tolower
129.        call    eax  # 以当前字符为参数，调用 tolower 函数转换为小写字母
130.
131.        mov dl, al
132.        lea ecx, [esp+25]
133.        mov eax, DWORD PTR [esp+36]
134.        add eax, ecx
135.        mov BYTE PTR [eax], dl  # 将转换后的小写字母放入新构建的字符串的对应位置
136.        jmp label_modify_nxt_char
137.
138.    label_modify_nxt_char:
139.        inc DWORD PTR [esp+36]  # 指针向后移动一位，指向下一个要转换的字符
140.
141.    label_modify_char:
142.        lea edx, [esp+25]  # 构建的新字符串的起始地址
143.        mov eax, DWORD PTR [esp+36]  # 当前处理的字符相对起始地址的偏移量
144.        add eax, edx
145.        mov al, BYTE PTR [eax]
146.
147.        test    al, al  # 下一个字符不是\0，跳转至 label_modify_lower_or_upper
148.        jne label_modify_lower_or_upper
149.
```

```
150.          # 将构建的新字符串（起始地址[esp+25]）和输出提示信息作为参数调
    用 printf 函数进行打印
151.      lea eax, [esp+25]
152.      mov DWORD PTR [esp+4], eax
153.      mov DWORD PTR [esp], OFFSET str_hint_output
154.      call    _printf
155.      jmp label_end_of_program
156.
157.  label_print_error_message:
158.      mov DWORD PTR [esp], OFFSET str_hint_error_letter
159.      call    _puts
160.      jmp label_end_of_program
161.
162.  hit_error_length:
163.      mov DWORD PTR [esp], OFFSET str_hint_error_length
164.      call    _puts
165.
166.  label_end_of_program:
167.      mov eax, 0
168.
169.      leave
170.      ret
```

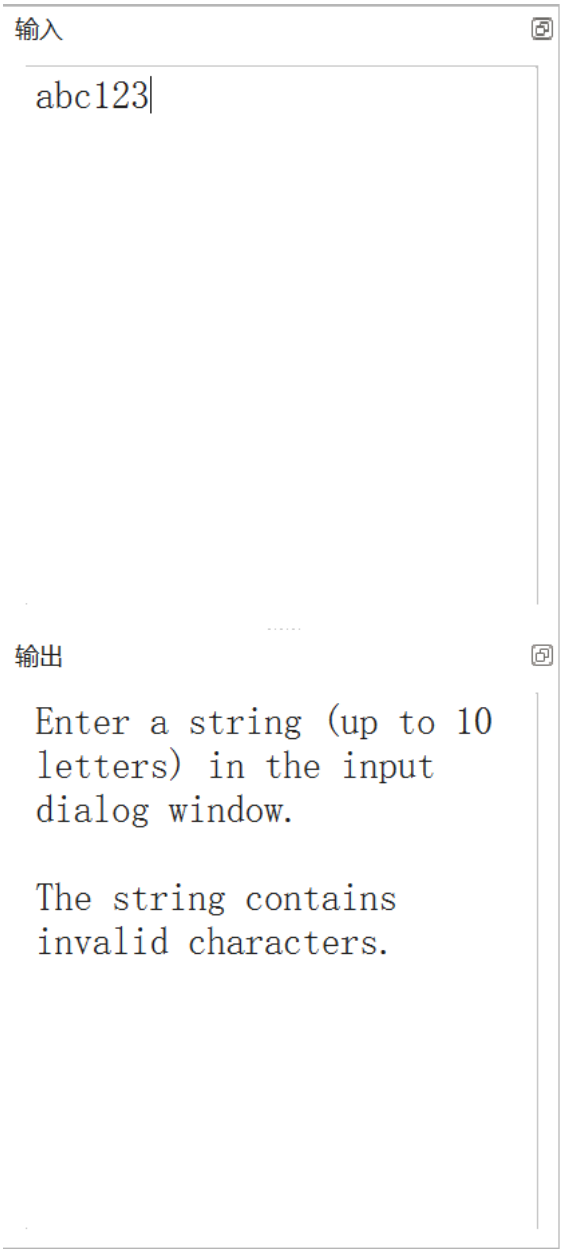
(.asm 文件及.exe 程序请见附件)

二、代码功能测试方案及结果

1、输入：abc123

输出：The string contains invalid characters.

如下图：

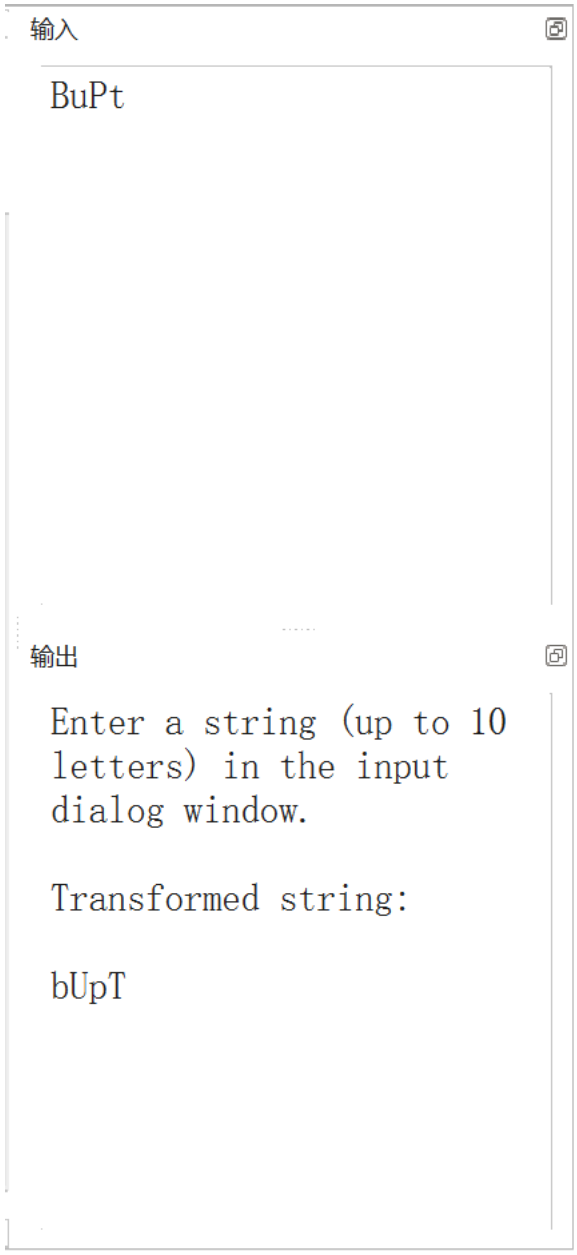


2、输入：BuPt

输出：Transformed string:

bUpT

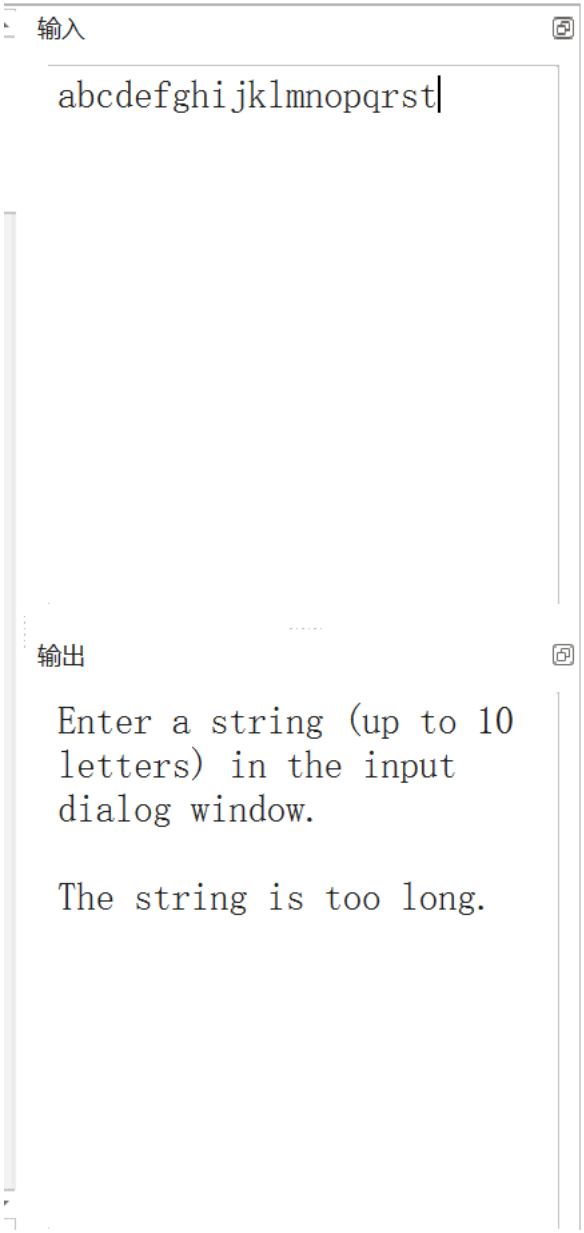
如下图：



3、输入：abcdefghijklmnopqrst

输出：The string is too long.

如下图：



三、解题逻辑思路

{*1}：此处代码段功能为打印输入字符串提示语，_printf 函数需要一个参数[esp]，而上一步（第 27 行）需用 mov 指令将 str_hint_input 传送至[esp]中，以达到输出文本"Enter a string (up to 10 letters) in the input dialog window."的目的。

{*2}：此处代码段功能为输入字符串，_scanf 函数需要两个参数，参数存放在与 scanf 的函数栈相邻的位置上，一个是[esp]，那么另外一个是[esp+4]（32 位程序，字符串地址占 4 字节），故第 31 行需用 mov 指令将 eax 传送至[esp+4]。

{*3}：第 39 行 cmp 指令执行从目的操作数中减去源操作数的隐含减法操作，并且不修改任何操作数。标志位 当实际的减法发生时，CMP 指令按照计算结果修改溢出、符号、零、进位、辅助进位和奇偶标志位。如果比较的是两个无符号数，则零标志位和进位标志位表示的两个操作数之间的关系如下表所示：

cmp 结果	ZF	CF
目的操作数 < 源操作数	0	1
目的操作数 = 源操作数	0	0
目的操作数 > 源操作数	1	0

第 40 行 ja 指令：当 CF=0 且 ZF=0，跳转。而此处代码段功能为检查字符串是否超过 10，故应该 cmp eax, 10。

{*4}：此处代码段功能为指针向后移动一位，指向下一个要检测的字符，inc 指令为自增+1 功能，由第 44 行可知，变量[esp+40] 用

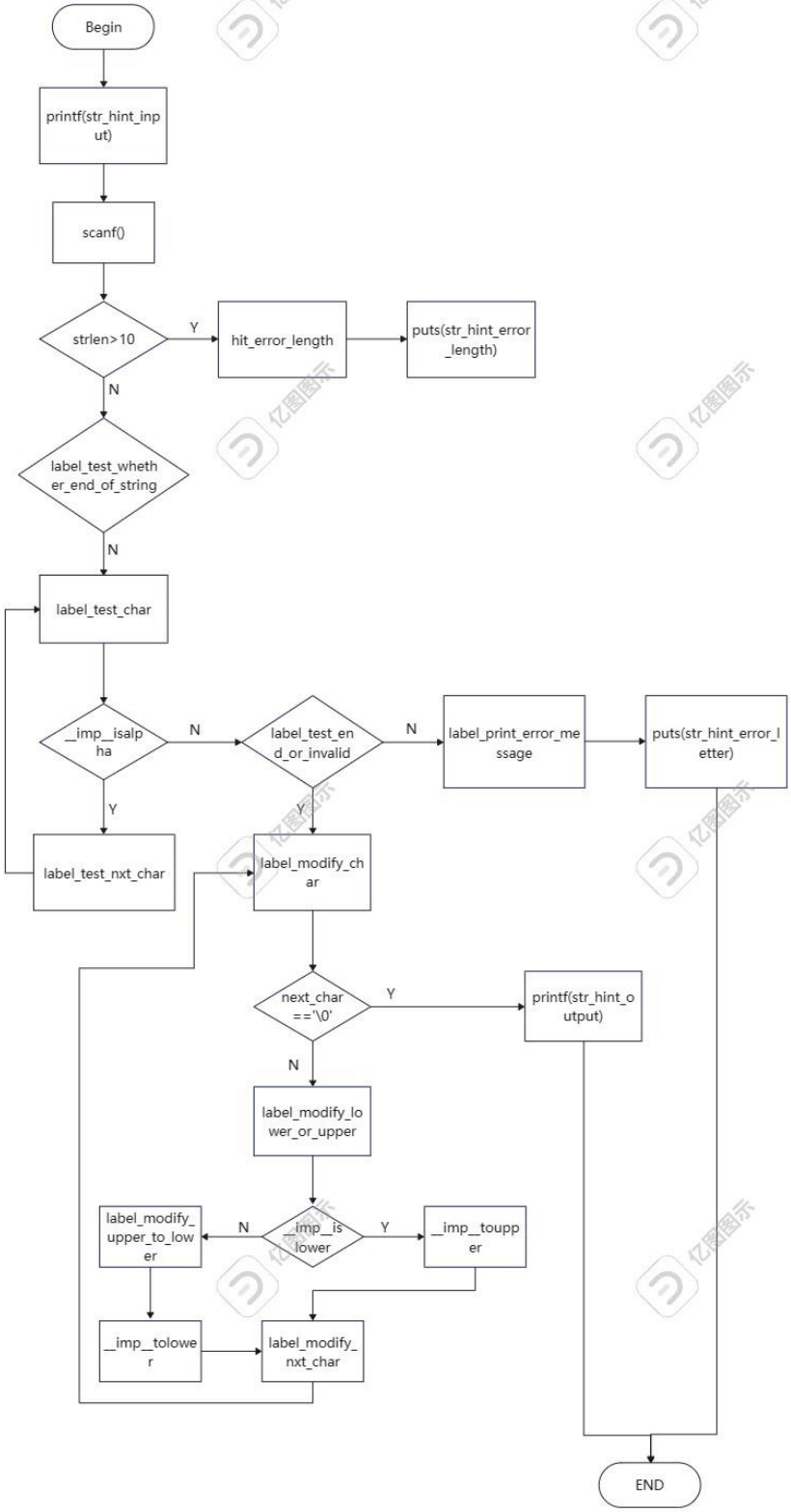
于记录当前处理的字符在字符串中的位置，故此处应填[esp+40]。

{*5}：第 74 行 jne 跳转：当 ZF=0，则跳转。此处代码段功能为检测当前字符的下一个位置是不是\0，若不是\0，跳转至 label_test_char。就需在第 73 行进行 test（异或）操作，不是\0 即 ZF=0，跳转。

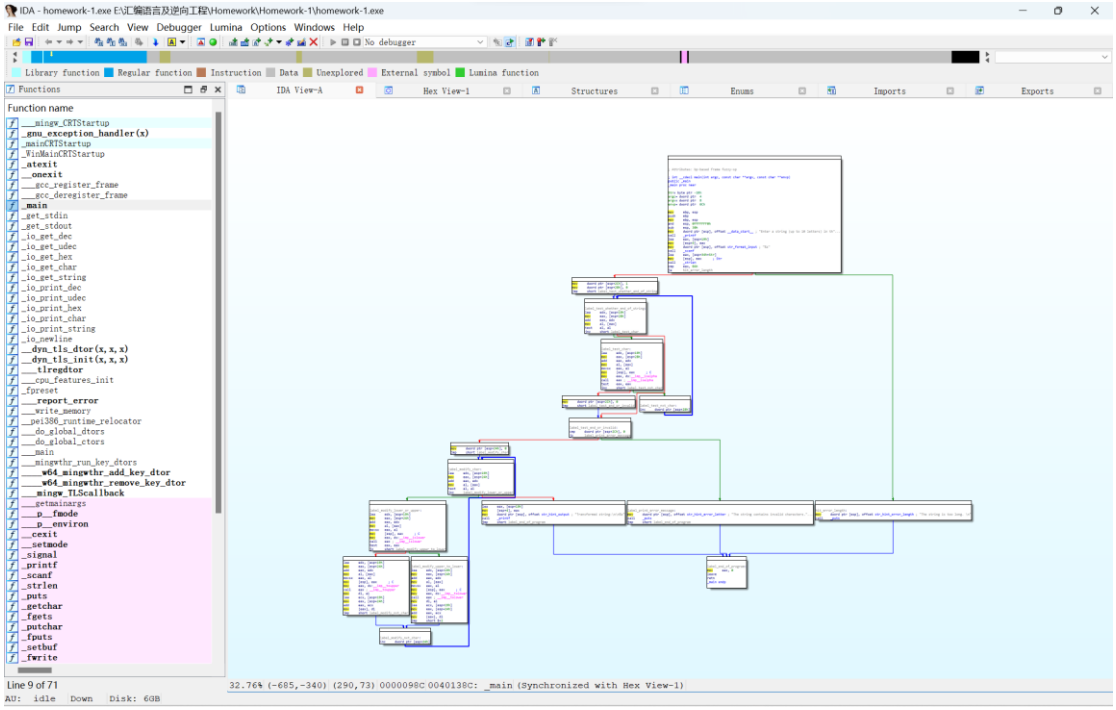
大写字母转小写功能代码：此处与上方小写转大写类似，只需注意 __imp__tolower 函数调用即可。

四、伪代码流程图

1、程序流程图

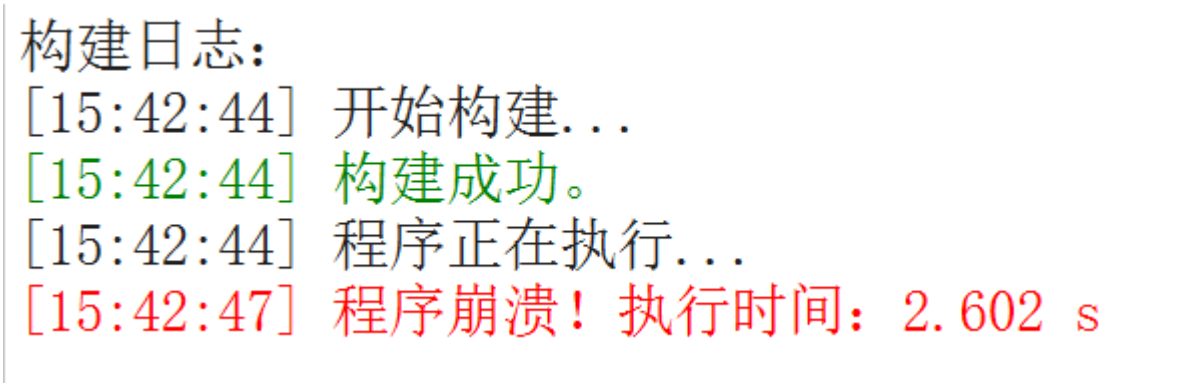


2、IDA Pro 分析截图



五、遇到的问题及解决方案

问题：最开始尝试填了一下，但是一直在提示“程序崩溃!”（如下图），我百思不得其解，查阅课件、百度、CSDN、ChatGPT…都没有好的解决方案，但是我发现当无输入的时候，程序会正常显示“The string contains invalid characters.”，而只要有输入就会提示“程序崩溃!”，所以我认为应该是{*2}处代码填写有误，导致字符串输入有误以至于运行失败。



解决方案：最后的最后，实在找不出解决方案，我只能去向董英杰学长寻求帮助，学长很耐心的解答，我也知道了问题所在，scanf 函数需要两个参数，参数存放在与 scanf 的函数栈相邻的位置上，一个是[esp]，那么另外一个是[esp+4]。确实是{*2}处的问题，我将代码进行更改后，终于“程序正常完成。”

六、体会

路漫漫其修远兮，吾将上下而求索。在《汇编语言与逆向工程》这门课程上我只是一个刚入门（可能还在门外）的小白，我不会的知识还有很多，在今后的学习生涯中我要更加努力。

最后，感谢潘老师课堂上的认真教学以及董学长的热心解答。