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
4. .data
5. str\_hint\_input:
6. .asciz "Enter a string (up to 10 letters) in the input dialog window. \n\n"
7. str\_format\_input:
8. .asciz "%s"
9. str\_hint\_output:
10. .asciz "Transformed string:\n\n%s"
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"
17. .text
18. .globl  \_main
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
27. mov DWORD PTR [esp], OFFSET str\_hint\_input #{\*1}
28. call    \_printf   # 打印输入字符串提示语
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
35. lea eax, [esp+25]
36. mov DWORD PTR [esp], eax
37. call    \_strlen   # 进行字符串长度计算
39. cmp eax,  10   # {\*3} 检查字符串是否超过 10
40. ja  hit\_error\_length        # ja 为无符号大于跳转汇编指令
42. 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
43. 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
44. jmp label\_test\_whether\_end\_of\_string
46. label\_test\_char:     # 检测当前字符是否为英文字母
47. lea edx, [esp+25]  # 字符串的起始地址
48. mov eax, DWORD PTR [esp+40]  # 当前字符相对字符串起始地址的偏移量
49. add eax, edx
50. mov al, BYTE PTR [eax]  # 取出当前字符
52. movsx   eax, al
53. mov DWORD PTR [esp], eax
54. mov eax, DWORD PTR \_\_imp\_\_isalpha
55. call    eax    # 以当前字符为参数，调用isalpha函数判断是否为英文字母
57. test    eax, eax
58. jne label\_test\_nxt\_char     # 是合法字符，跳转至label\_test\_nxt\_char
60. mov DWORD PTR [esp+44], 0
62. jmp label\_test\_end\_or\_invalid     # 是非法字符，[esp+44]置0，跳转至label\_test\_end\_or\_invalid
64. label\_test\_nxt\_char:
65. 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
67. label\_test\_whether\_end\_of\_string:     # 当前字符的下一个位置是不是\0
68. lea edx, [esp+25]
69. mov eax, DWORD PTR [esp+40]
70. add eax, edx
71. mov al, BYTE PTR [eax]
73. test al, al   # {\*5}
74. jne label\_test\_char     # 下一个字符不是\0，跳转至label\_test\_char
76. label\_test\_end\_or\_invalid:
77. cmp DWORD PTR [esp+44], 0  # 检查一下[esp+44]的值判断是否含有非法字符
78. je  label\_print\_error\_message  # 存在非法字符，跳转至label\_print\_error\_message输出错误信息
80. 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
82. jmp label\_modify\_char  # 输入的字符串合法，跳转至label\_modify\_char
84. label\_modify\_lower\_or\_upper:    # label\_modify\_lower\_or\_upper
85. lea edx, [esp+25]
86. mov eax, DWORD PTR [esp+36]
87. add eax, edx
88. mov al, BYTE PTR [eax]
90. movsx   eax, al
91. mov DWORD PTR [esp], eax
92. mov eax, DWORD PTR \_\_imp\_\_islower
93. call    eax  # 以当前字符为参数，调用islower函数判断是否为小写字母
95. test    eax, eax
96. je  label\_modify\_upper\_to\_lower  # 是大写字母，跳转至label\_modify\_upper\_to\_lower
98. # 小写字母转大写
99. lea edx, [esp+25]
100. mov eax, DWORD PTR [esp+36]
101. add eax, edx
102. mov al, BYTE PTR [eax]
104. movsx   eax, al
105. mov DWORD PTR [esp], eax
106. mov eax, DWORD PTR \_\_imp\_\_toupper
107. call    eax  # 以当前字符为参数，调用toupper函数转换为大写字母
109. mov dl, al
110. lea ecx, [esp+25]
111. 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
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
121. lea edx, [esp+25]
122. mov eax, DWORD PTR [esp+36]
123. add eax, edx
124. mov al, BYTE PTR [eax]
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
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
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
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]
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
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
157. label\_print\_error\_message:
158. mov DWORD PTR [esp], OFFSET str\_hint\_error\_letter
159. call    \_puts
160. jmp label\_end\_of\_program
162. hit\_error\_length:
163. mov DWORD PTR [esp], OFFSET str\_hint\_error\_length
164. call    \_puts
166. label\_end\_of\_program:
167. mov eax, 0
169. leave
170. ret

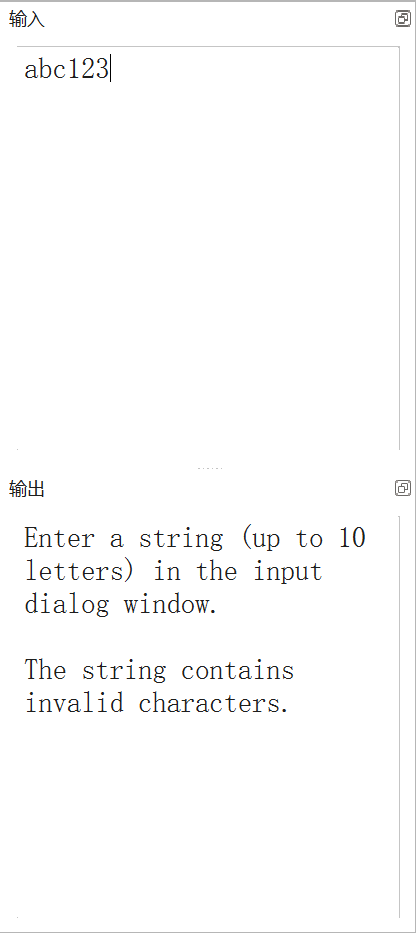
(.asm文件及.exe程序请见附件)

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：

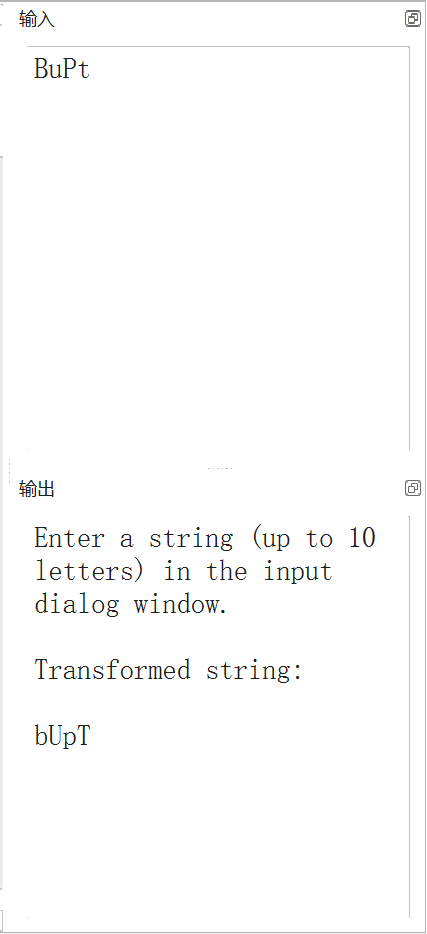


**2、**输入：BuPt

输出：Transformed string:

bUpT

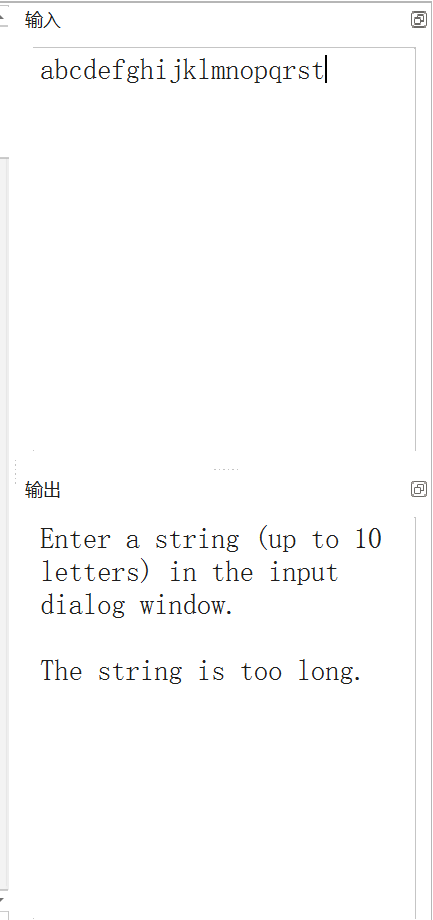
如下图：



**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。

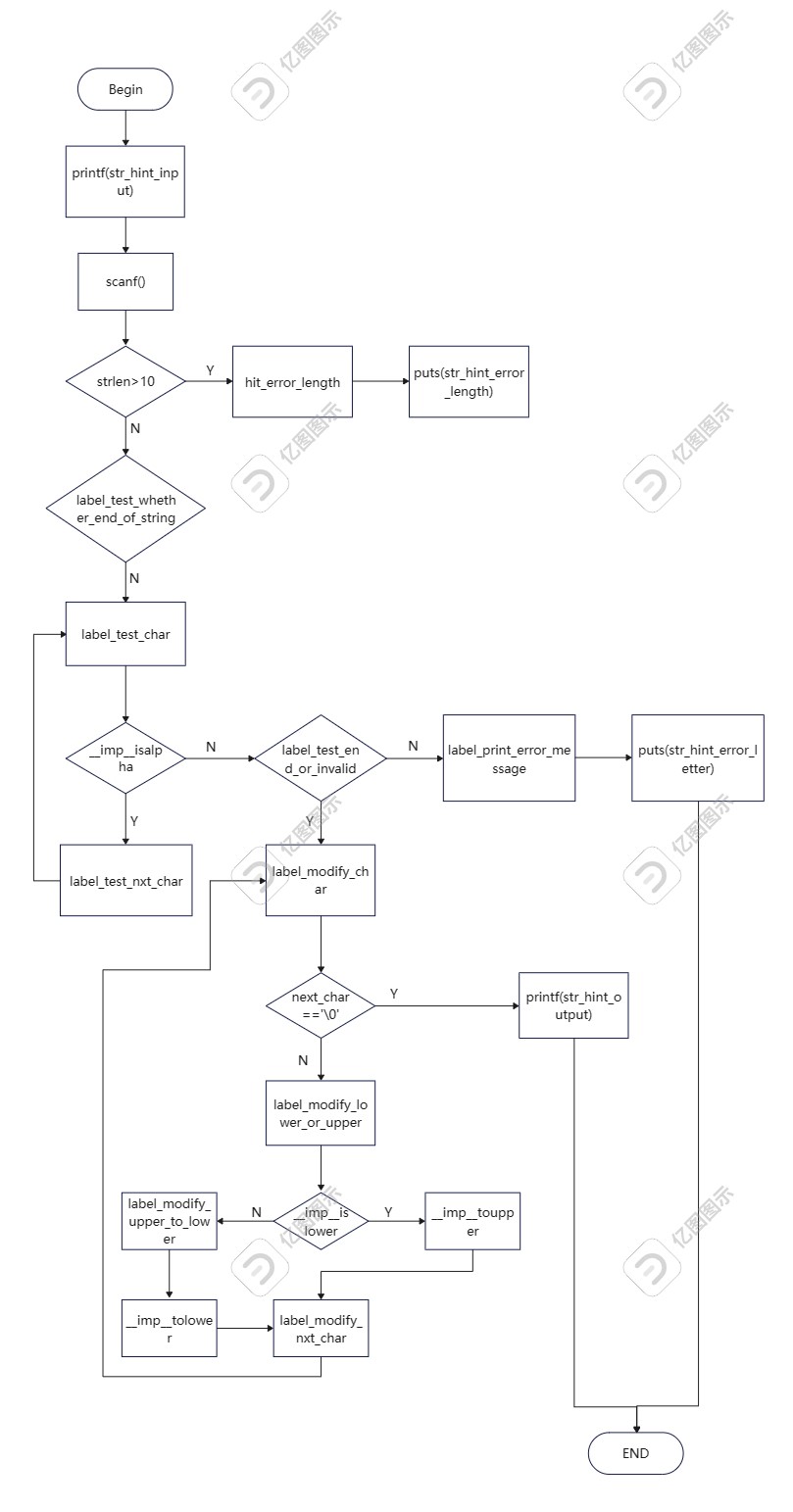
**{\*4}：**此处代码段功能为指针向后移动一位，指向下一个要检测的字符，inc指令为自增+1功能，由第44行可知，变量[esp+40] 用于记录当前处理的字符在字符串中的位置，故此处应填[esp+40]。

**{\*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。

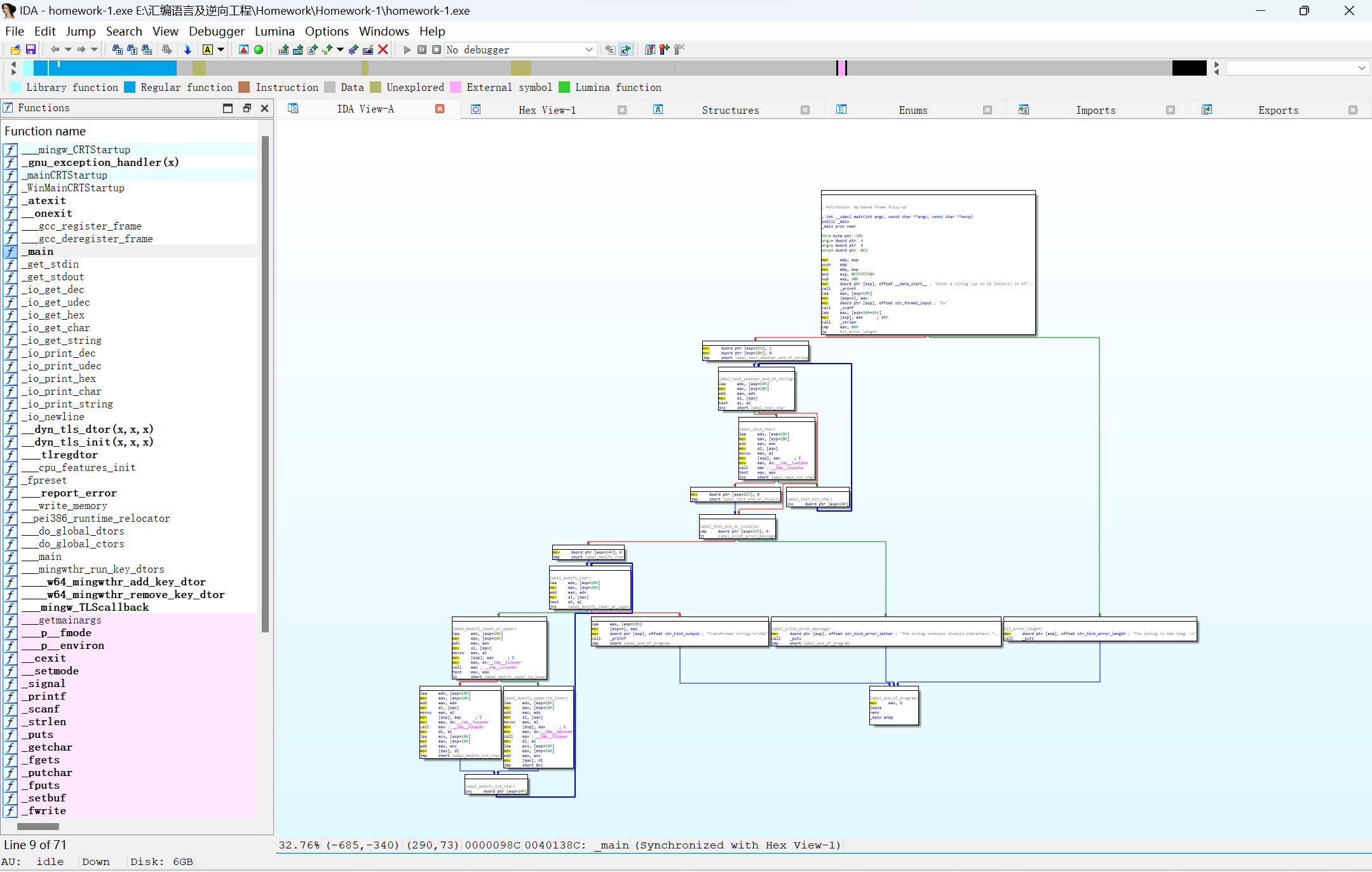
**大写字母转小写功能代码：**此处与上方小写转大写类似，只需注意\_\_imp\_\_tolower 函数调用即可。

四、伪代码流程图

**1、**程序流程图

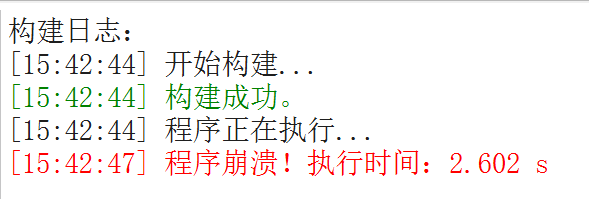


**2、**IDA Pro分析截图



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。