南开大学

**RE Challenge2**

**（汇编语言与逆向技术实验 8）**

****

**姓名：申宗尚**

**学号：2213924**

**专业：信息安全**

1. **实验目的**

1、熟悉静态反汇编工具IDA Freeware；

2、熟悉反汇编代码的逆向分析过程；

3、掌握反汇编语言中的数学计算、数据结构、条件判断、分支结构的识别和逆向分析

1. **实验环境**

1、ida64

1. challenge.exe

**三．实验原理**

1. **task1**
2. 通过IDA Freeware得到task1.exe的反汇编代码，如图1和图2所示。

图形用户界面, 应用程序

描述已自动生成

图 1 task1.exe的反汇编代码

图示, 示意图

描述已自动生成

图 2 task1.exe反汇编代码的图形化显示

1. 对反汇编代码和计算过程、条件判断、分支结构等信息进行分析，逆向推出程序的正确输入数据，完成逆向分析挑战。

张着嘴

中度可信度描述已自动生成

图 3 逆向分析，完成task1练习

1. **task2**
2. 通过IDA Freeware得到task2.exe的反汇编代码，如图4和图5所示。

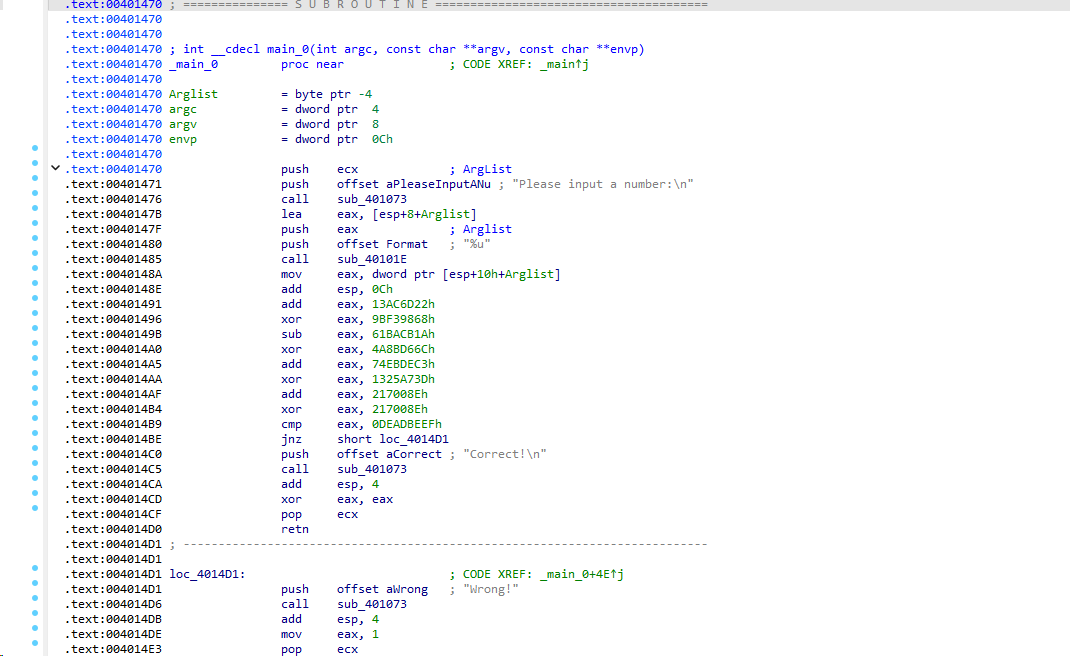


图4 task2.exe的反汇编代码

图形用户界面, 文本

描述已自动生成

图 5 task2.exe反汇编代码的图形化显示

1. 对反汇编代码的计算过程、条件判断、分支结构等信息进行分析，逆向推出程序的正确输入数据，完成逆向分析挑战。

文本

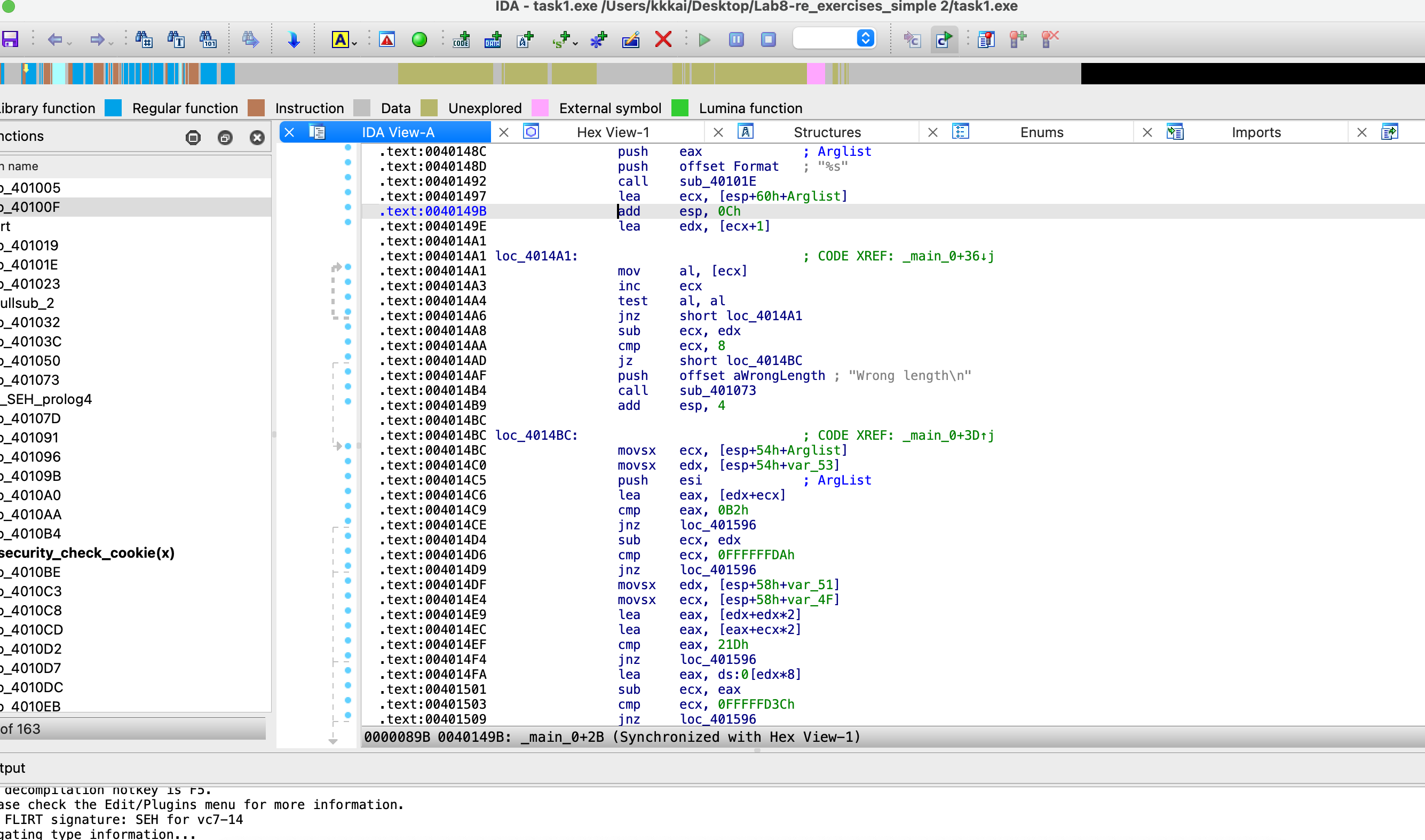
描述已自动生成

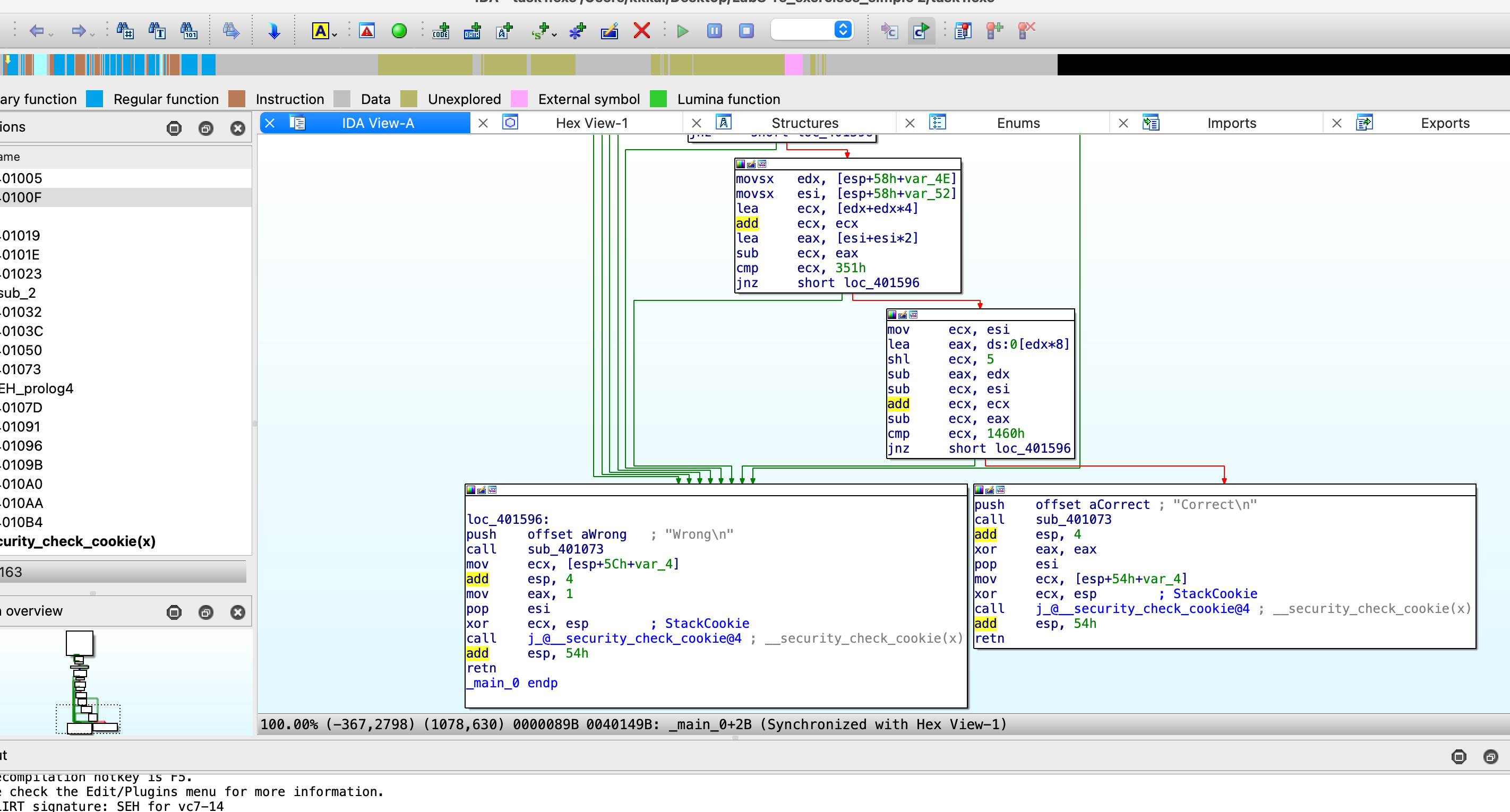
图 6 逆向分析，完成task2练习

1. **实验内容**

**（1）Task1**

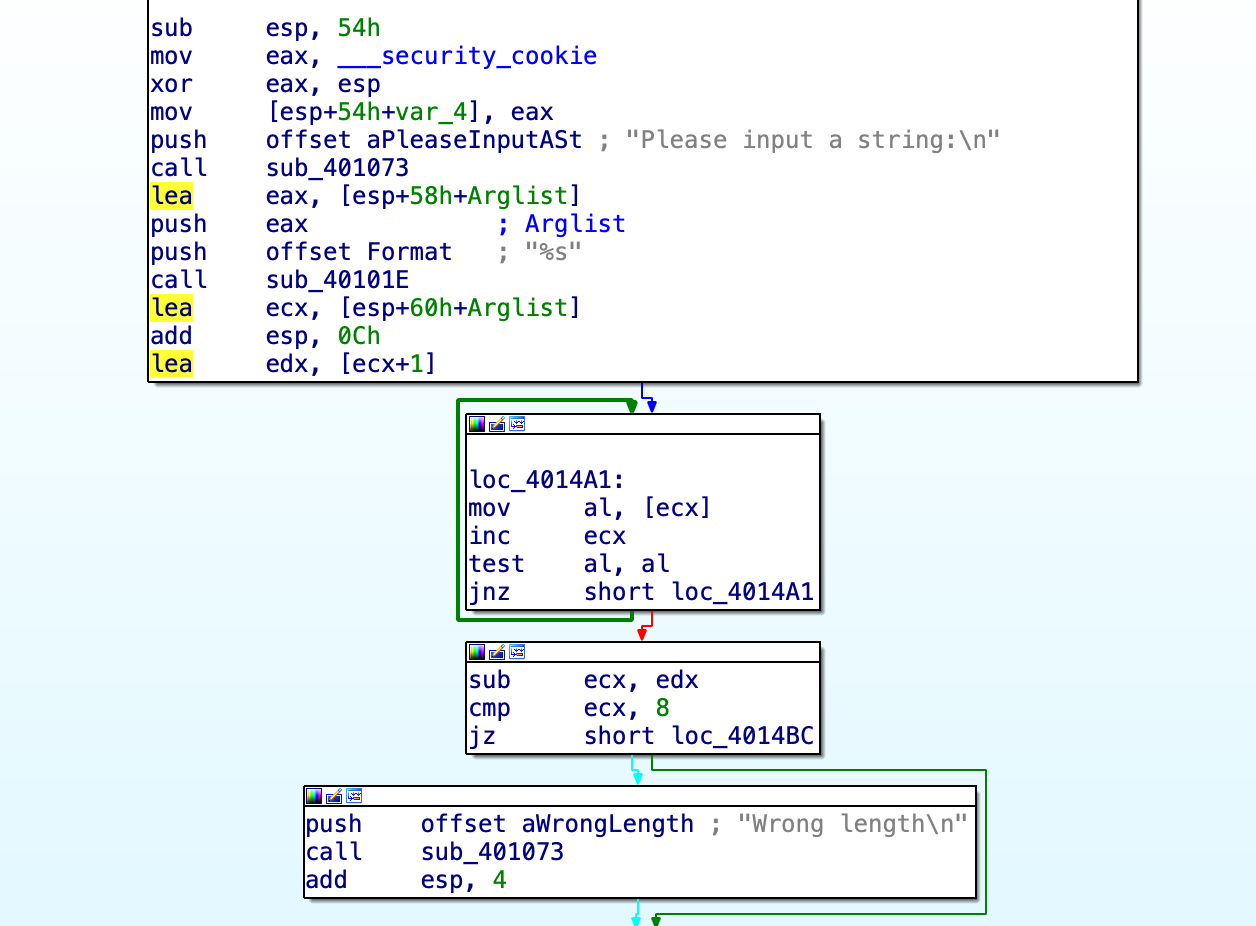
如图，将task1.exe导入ida64 Freeware中，可以得到二进制代码的反汇编代码





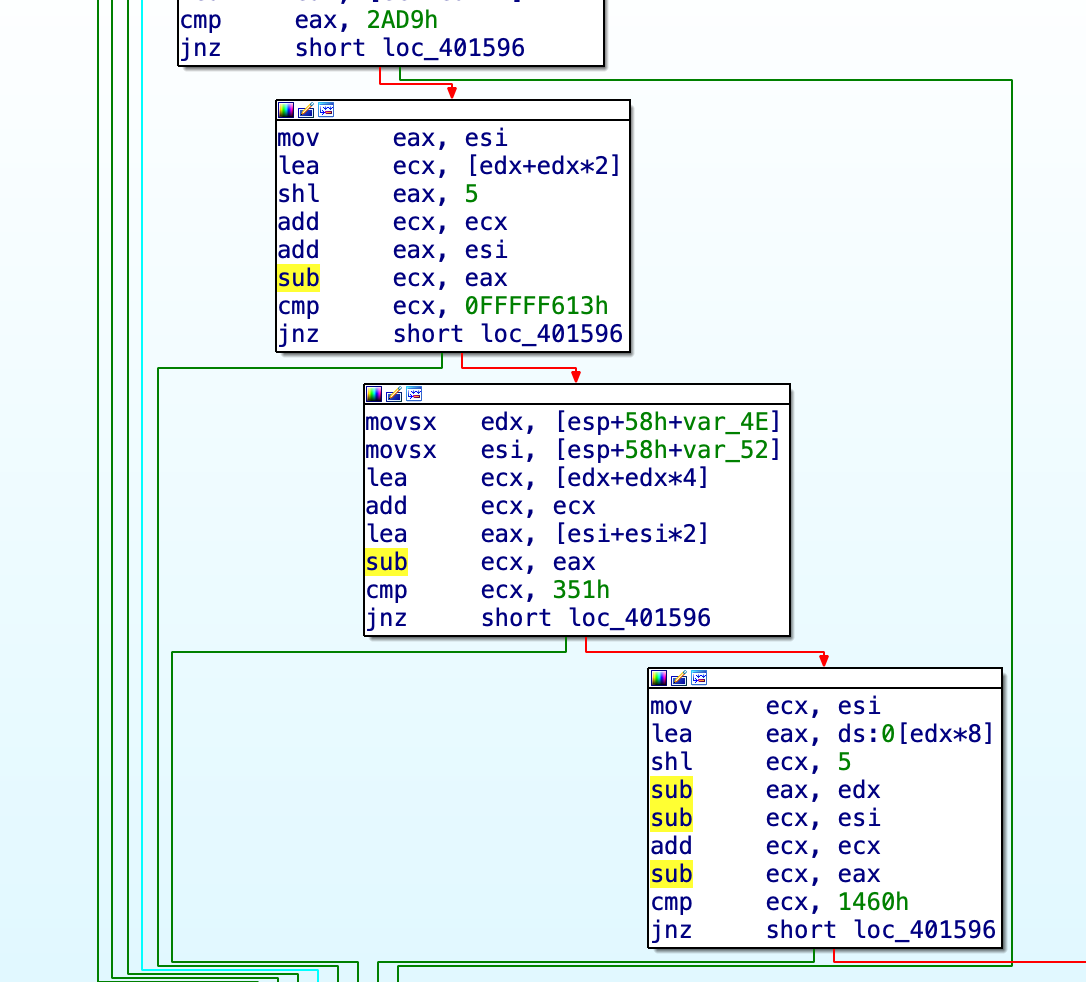
源代码见附件。

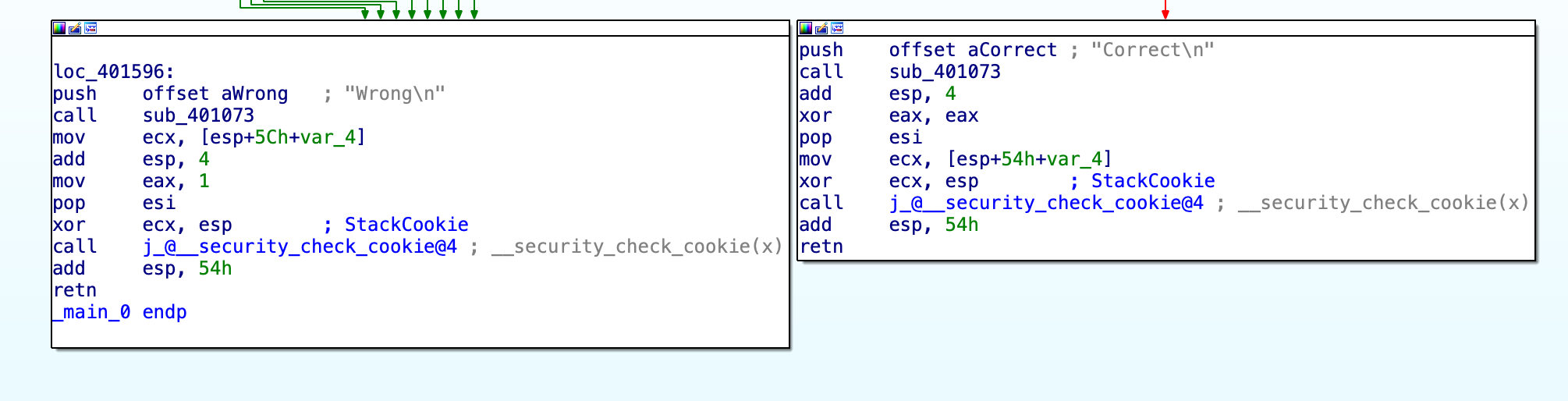
从而，我们可以通过对该反汇编代码的计算过程、数据结构、条件判断、分支结构等信息进行分析，逆向推理出程序的正确输入数据，完成逆向分析挑战。



首先，可以通过一开始的流程图推出，先进行了栈空间分配，安全性处理，和输出prompt字符串，用户输入的字符串存入Arglist中，最后将ecx+1结果存入edx中，指向字符串的下一个字符，为后续函数调用做准备。

随后对Arglist的长度进行判断，如果长度不为8，则输出“Wrong length”，由此可得，本题答案应该为长度为8的字符串

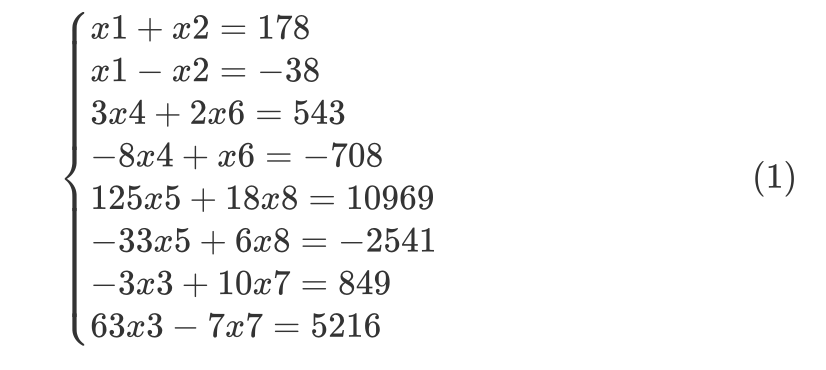


在代码的下一部分，分多个区块对字符串的逐位进行了比对，整体逻辑为：如果判断正确，运行下一个模块代码，如果错误，则直接跳转至loc\_401596，即输出：Wrong。

从而可以得知，只要判断每个区块代码对字符串每一位的判断条件，就可以解出字符串最后的每一位字符，从而得出答案，下为具体分析：

在整个程序过程中，通过位置运算，将输入字符串的8位字符分别存入了Arglist、var\_53、var\_52、var\_51、var\_50、var\_4F、var\_4E、var\_4D八个变量中，代表1、2、3、4、5、6、7、8位，然后总共列出了8个方程式：

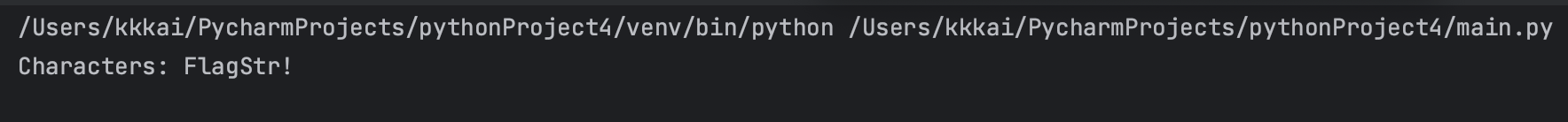
（为方便表示，将Arglist…var\_4D分别命名为x1…x8）



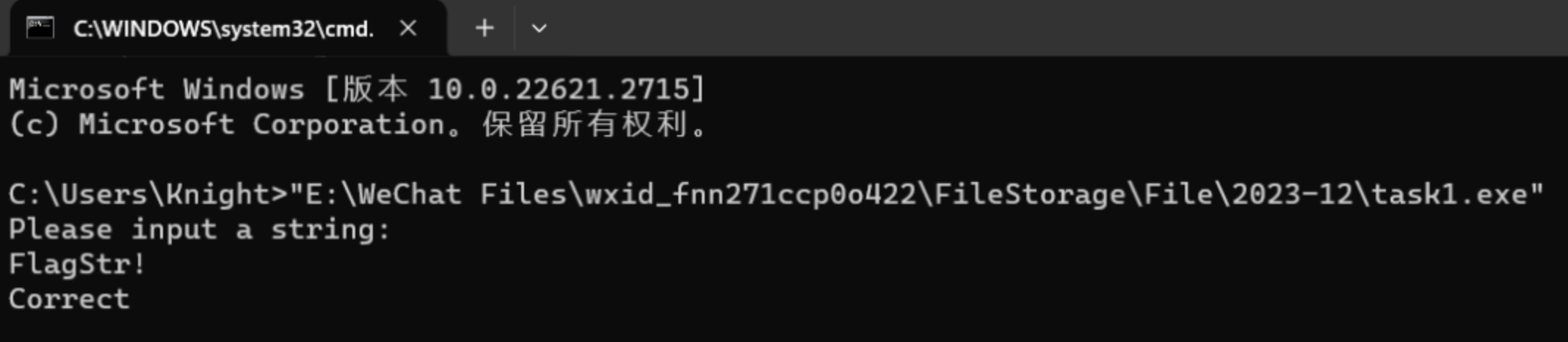
从而，解本题答案变成了解八元线性方程组，从而可以列写python程序解线性方程组矩阵，并将其输出为对应的ASCII码值，代码如下：



从而运行程序，得到答案：

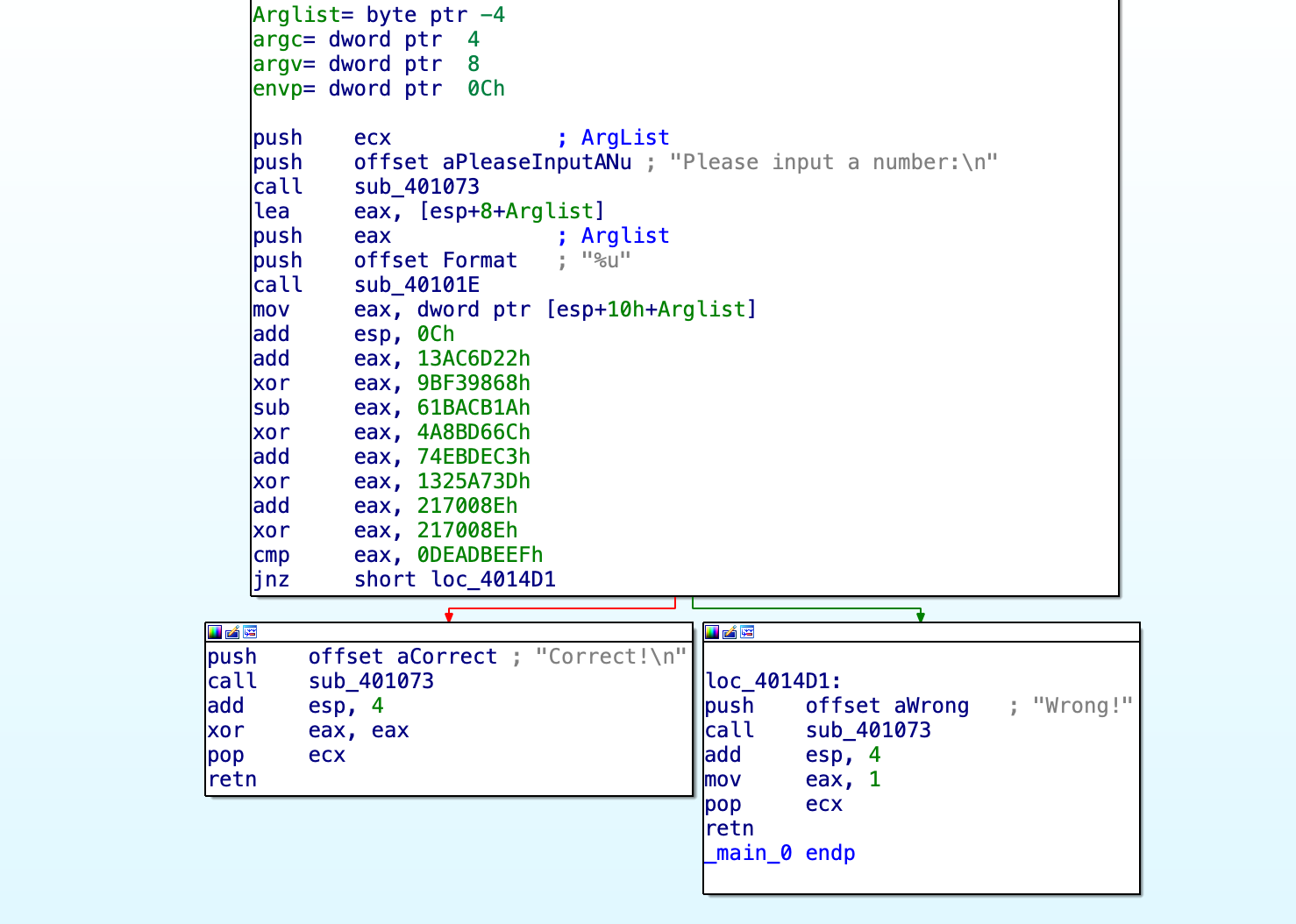


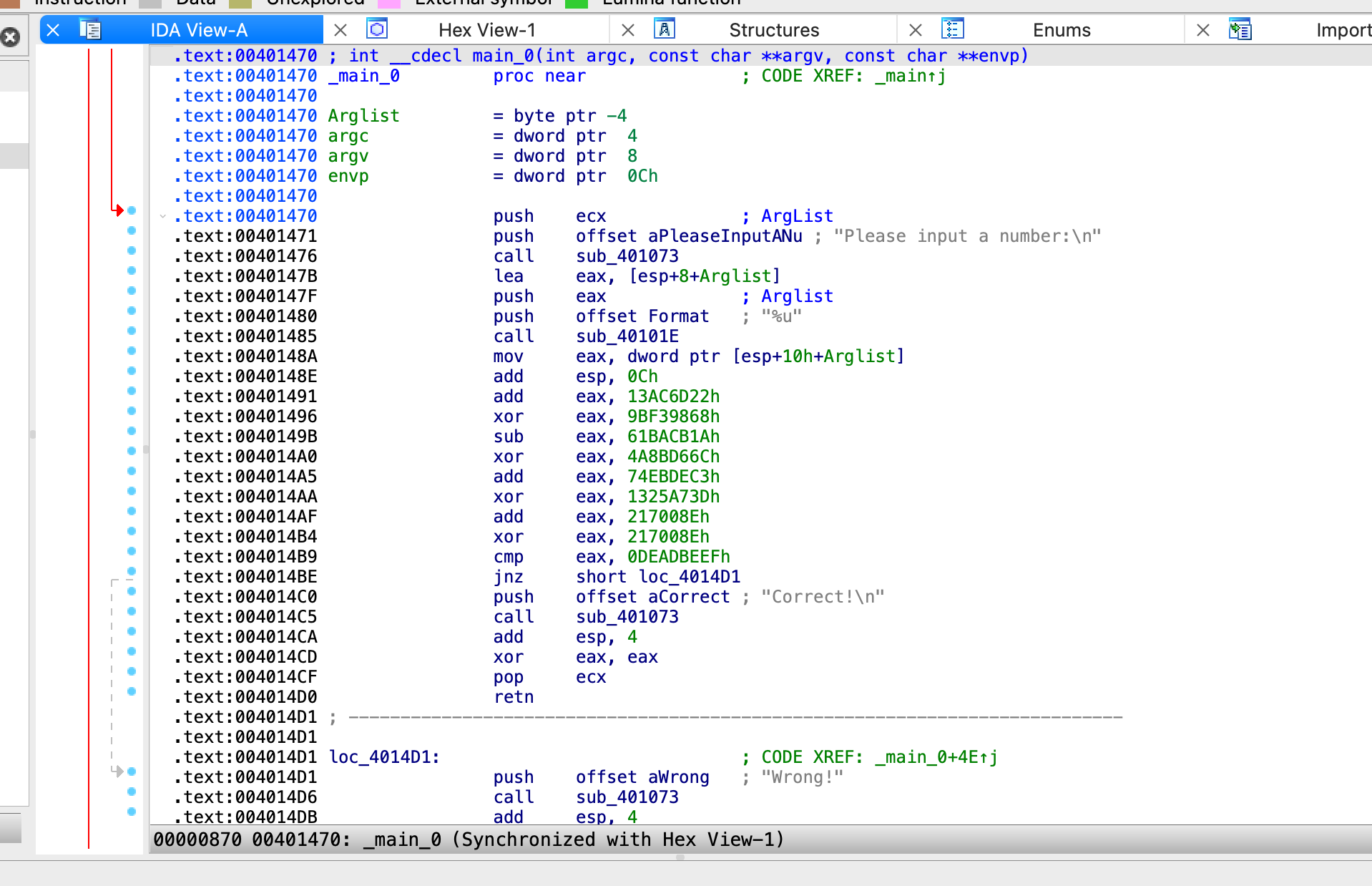
在电脑上测试，得到答案：



**（1）Task2**

如图，将task2.exe导入ida64 Freeware中，可以得到二进制代码的反汇编代码





分析代码可知，该程序主要对用户输入的数字进行一系列的异或、加、减运算，最后与0DEADBEEFh相比较，如果相等则输出“Correct！”，不相等则输出“Wrong！”，从而可以对每一步运算进行逆运算，由0DEADBEEFh反推出输入的正确数字。

其函数处理流程如下：  
（1）输入数字a

（2）a + 0x13AC6D22

（3）a ^ 0x9BF39868

（4）a – 0x61BACB1A

（5）a ^ 0x4A88D66C

（6）a + 0x74EBDEC3

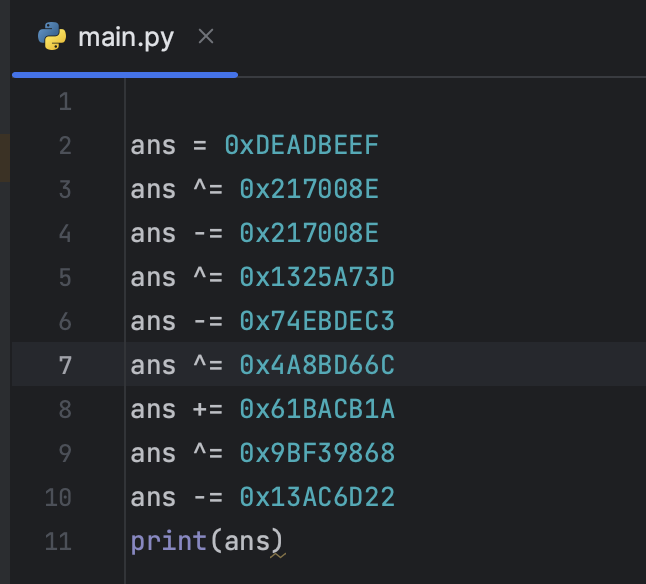
（7）a ^ 0x1325A73D

（8）a + 0x217008E

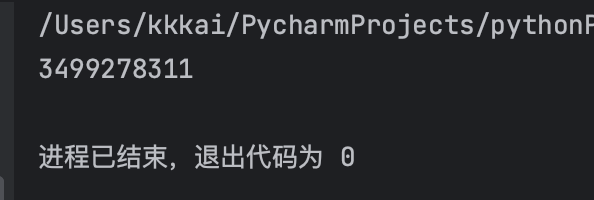
（9）a ^ 0x217008E

比较最后数字与0DEADBEEFh

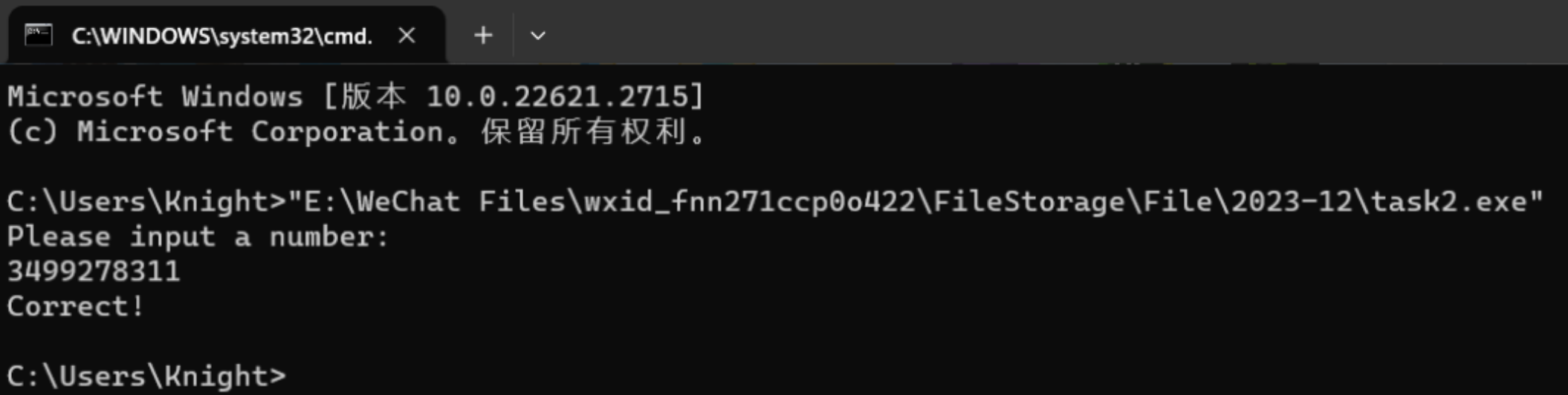
从而，又+逆运算为-，-逆运算为+，异或^逆为异或^，从而可以编写python程序



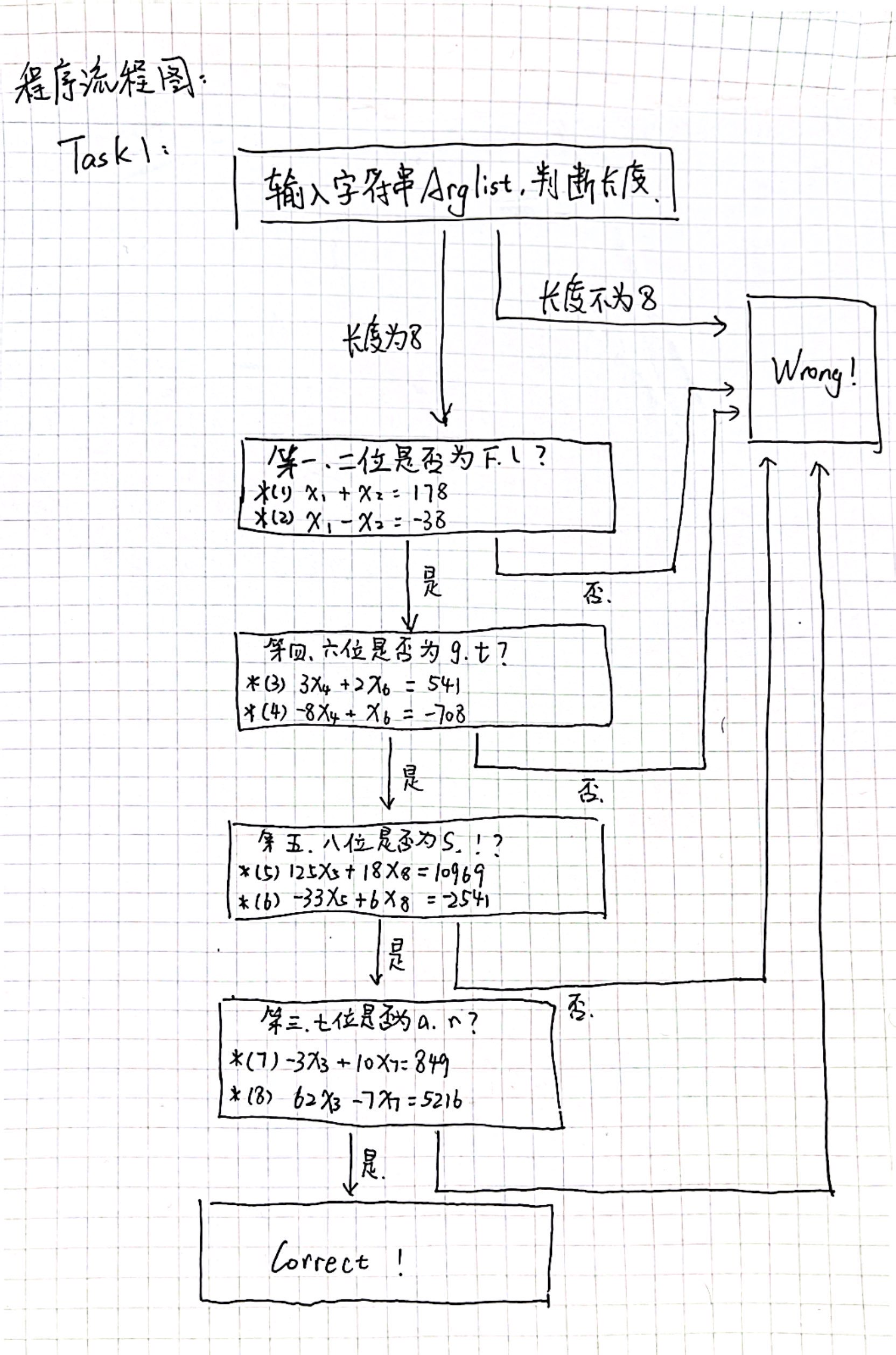
运行程序，得出答案

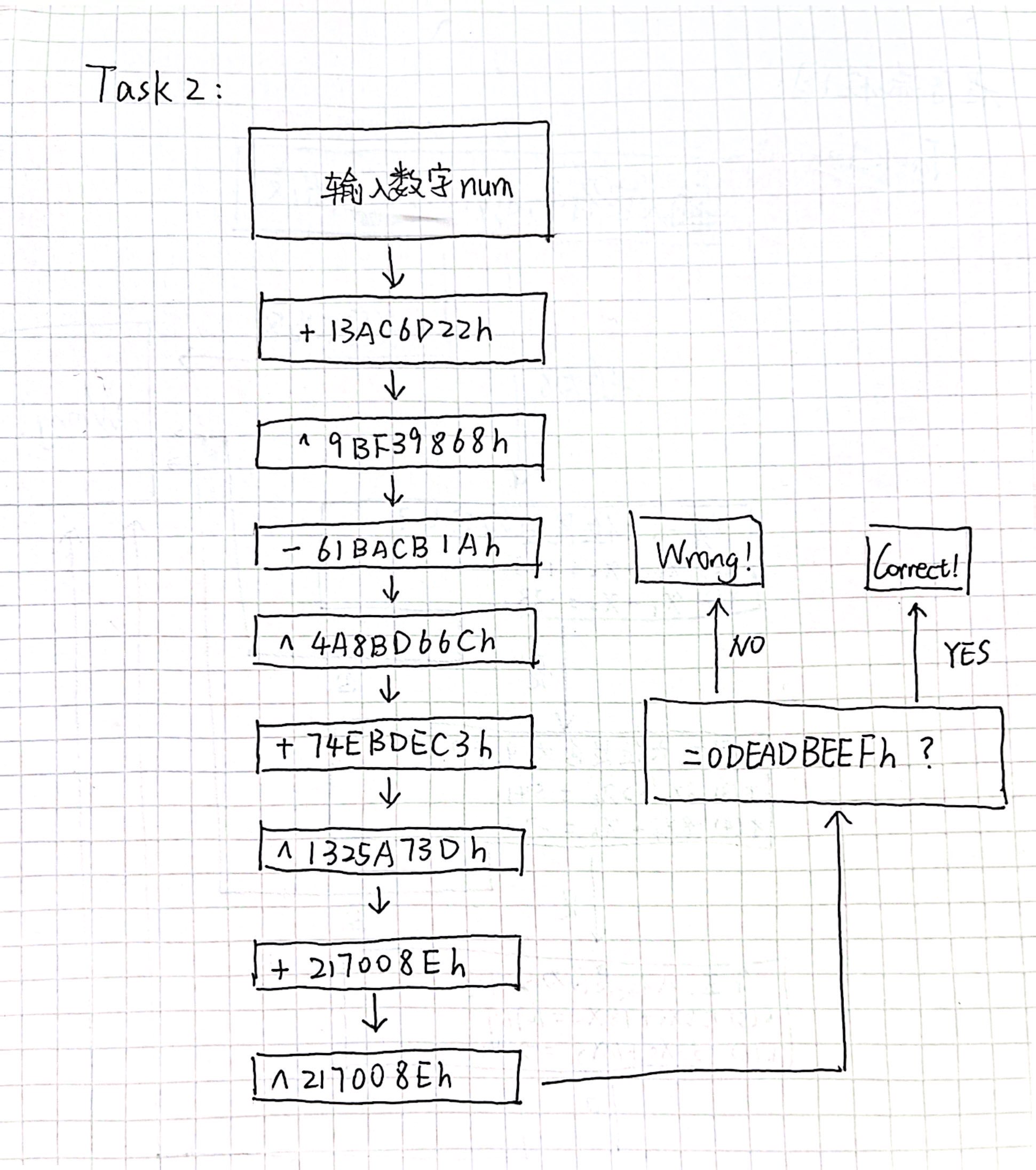


运行task2.exe程序，输入得到最终结果：



**四.流程图**





**五.附件（python代码）**

**Task1:**

import numpy as np  
  
A = np.array([[1, 1, 0, 0, 0, 0, 0, 0],  
 [1, -1, 0, 0, 0, 0, 0, 0,],  
 [0, 0, 0, 3, 0, 2, 0, 0],  
 [0, 0, 0, -8, 0, 1, 0, 0],  
 [0, 0, 0, 0, 125, 0, 0, 18],  
 [0, 0, 0, 0, -33, 0, 0, 6],  
 [0, 0, -3, 0, 0, 0, 10, 0],  
 [0, 0, 62, 0, 0, 0, -7, 0],]) # 创建矩阵A  
b = np.array([[178],[-38],[541],[-708],[10969],[-2541],[849],[5216]]) # 矩阵b  
C = np.append(A, b, axis=1) # 增广矩阵  
  
x = np.linalg.solve(A, b) # 使用 solve 函数求解线性方程组  
  
characters = ''.join([chr(int(round(i))) for i in x.flatten()]) #将答案 x 中的值转化为字符  
print(f"Characters: {characters}")

**Task2:**

ans = 0xDEADBEEF  
ans ^= 0x217008E  
ans -= 0x217008E  
ans ^= 0x1325A73D  
ans -= 0x74EBDEC3  
ans ^= 0x4A8BD66C  
ans += 0x61BACB1A  
ans ^= 0x9BF39868  
ans -= 0x13AC6D22  
print(ans)