

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

# 一. 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

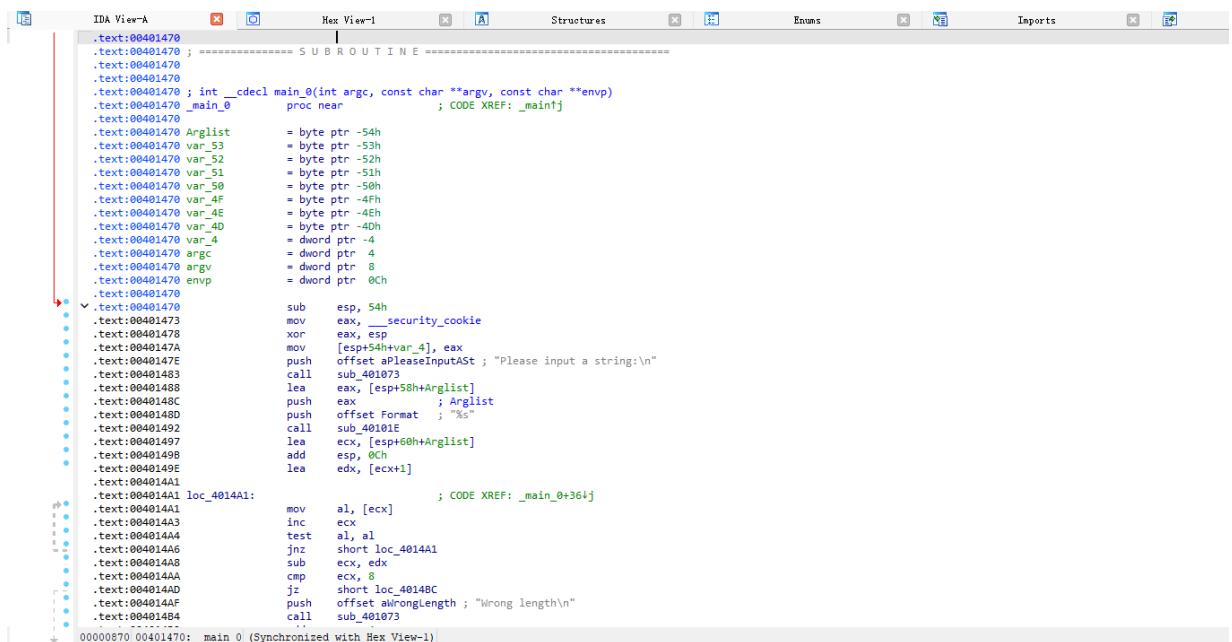
# 二. 实验环境

- 1、ida64
- 2、challenge.exe

# 三. 实验原理

## (一) task1

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



```
.text:00401470 ; ===== S U B R O U T I N E =====
.text:00401470
.text:00401470
.text:00401470
.text:00401470 ; int __cdecl main_(int argc, const char **argv, const char **envp)
.text:00401470 _main_0 proc near                ; CODE XREF: _mainf
.text:00401470
.text:00401470 Arglist      = byte ptr -54h
.text:00401470 var_33       = byte ptr -53h
.text:00401470 var_32       = byte ptr -52h
.text:00401470 var_51       = byte ptr -51h
.text:00401470 var_50       = byte ptr -50h
.text:00401470 var_4F       = byte ptr -4Fh
.text:00401470 var_4E       = byte ptr -4Eh
.text:00401470 var_4D       = byte ptr -4Dh
.text:00401470 var_4C       = dword ptr -4Ch
.text:00401470 argc        = dword ptr 4
.text:00401470 argv        = dword ptr 8
.text:00401470 envp        = dword ptr 0Ch
.text:00401470
.text:00401470
.text:00401470 sub_esp, 54h
.text:00401470 mov_eax, __security_cookie
.text:00401470 xor_eax, esp
.text:00401470 mov [esp+54h+var_4], eax
.text:00401470 push offset aPleaseInputSt ; "Please input a string:\n"
.text:00401470 call sub_401073
.text:00401470
.text:00401470 lea_eax, [esp+58h+Arglist]
.text:00401470 push_eax
.text:00401470 offset Format ; "%s"
.text:00401470 push sub_40101E
.text:00401470 call ecx, [esp+60h+Arglist]
.text:00401470 add_esp, 0Ch
.text:00401470 lea_edx, [ecx+1]
.text:00401470
.text:00401470 loc_4014A1:           ; CODE XREF: _main_0+36+j
.text:00401470     mov al, [ecx]
.text:00401470     inc ecx
.text:00401470     test al, al
.text:00401470     jnz short loc_4014A1
.text:00401470     sub ecx, edx
.text:00401470     cmp ecx, 8
.text:00401470     jz short loc_4014BC
.text:00401470     push offset aWrongLength ; "Wrong length\n"
.text:00401470     call sub_401073
```
00000870 00401470: _main_0 (Synchronized with Hex View-1)
```

图 1 task1.exe 的反汇编代码

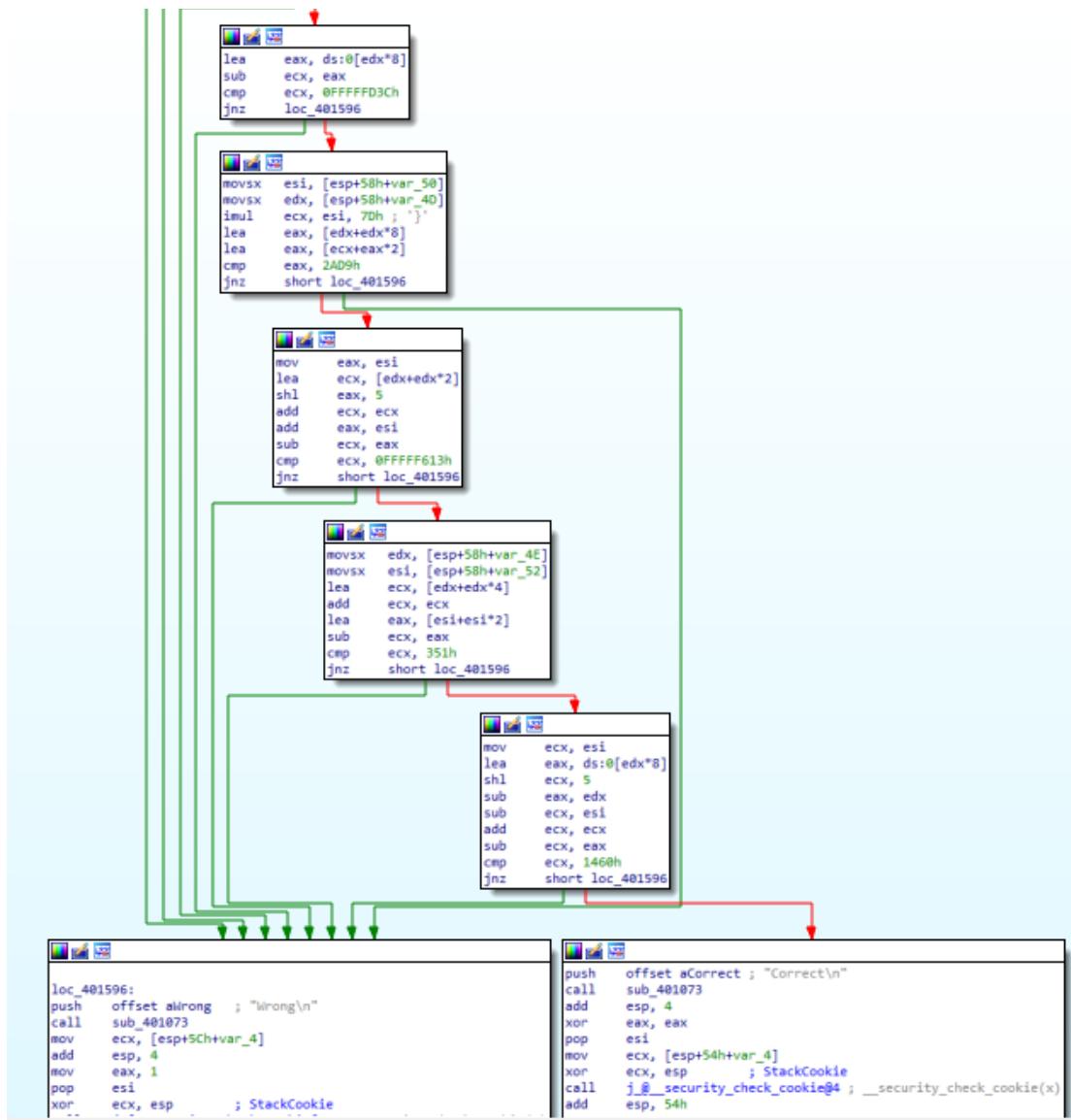


图 2 task1.exe 反汇编代码的图形化显示

2. 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。

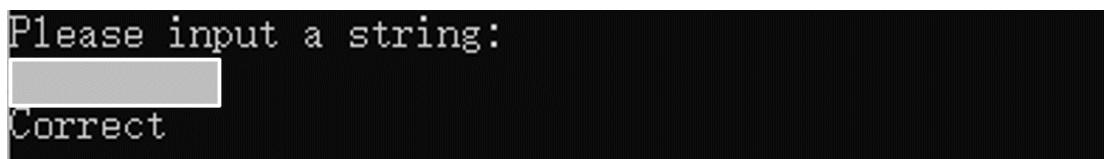


图 3 逆向分析，完成 task1 练习

## (二) task2

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

```

.text:00401470 ; ===== S U B R O U T I N E =====
.text:00401470
.text:00401470
.text:00401470 ; int __cdecl main_0(int argc, const char **argv, const char **envp)
.text:00401470 _main_0 proc near ; CODE XREF: _mainfj
.text:00401470
.text:00401470 Arglist = byte ptr -4
.text:00401470 argc = dword ptr 4
.text:00401470 argv = dword ptr 8
.text:00401470 envp = dword ptr 0Ch
.text:00401470
.text:00401470 push ecx ; ArgList
.text:00401471 push offset aPleaseInputANu ; "Please input a number:\n"
.text:00401476 call sub_401073
.text:00401478 lea eax, [esp+8+Arglist]
.text:0040147F push eax ; Arglist
.text:00401480 push offset Format ; "%u"
.text:00401485 call sub_40101E
.text:0040148A mov eax, dword ptr [esp+10h+Arglist]
.text:0040148E add esp, 0Ch
.text:00401491 add eax, 13AC6D22h
.text:00401496 xor eax, 9BF39868h
.text:0040149B sub eax, 61BACB1Ah
.text:004014A0 xor eax, 4A8BD66Ch
.text:004014A5 add eax, 74EBDEC3h
.text:004014A4 xor eax, 1325A73Dh
.text:004014AF add eax, 217008Eh
.text:004014B4 xor eax, 217008Eh
.text:004014B9 cmp eax, 0DEADBEFFh
.text:004014BE jnz short loc_4014D1
.text:004014C0 push offset aCorrect ; "Correct!\n"
.text:004014C5 call sub_401073
.text:004014CA add esp, 4
.text:004014CD xor eax, eax
.text:004014CF pop ecx
.text:004014D0 retn
.text:004014D1 ;
.text:004014D1 loc_4014D1: ; CODE XREF: _main_0+4E!j
.text:004014D1 push offset aWrong ; "Wrong!"
.text:004014D6 call sub_401073
.text:004014D8 add esp, 4
.text:004014D9 mov eax, 1
.pop ecx

```

图 4 task2.exe 的反汇编代码

```

; int __cdecl main_0(int argc, const char **argv, const char **envp)
_main_0 proc near

Arglist= byte ptr -4
argc= dword ptr 4
argv= dword ptr 8
envp= dword ptr 0Ch

push ecx ; ArgList
push offset aPleaseInputANu ; "Please input a number:\n"
call sub_401073
lea eax, [esp+8+Arglist]
push eax ; Arglist
push offset Format ; "%u"
call sub_40101E
mov eax, dword ptr [esp+10h+Arglist]
add esp, 0Ch
add eax, 13AC6D22h
xor eax, 9BF39868h
sub eax, 61BACB1Ah
xor eax, 4A8BD66Ch
add eax, 74EBDEC3h
xor eax, 1325A73Dh
add eax, 217008Eh
xor eax, 217008Eh
cmp eax, 0DEADBEFFh
jnz short loc_4014D1

loc_4014D1:
push offset aCorrect ; "Correct!\n"
call sub_401073
add esp, 4
xor eax, eax
pop ecx
retn

_main_0 endp

```

图 5 task2.exe 反汇编代码的图形化显示

2. 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。

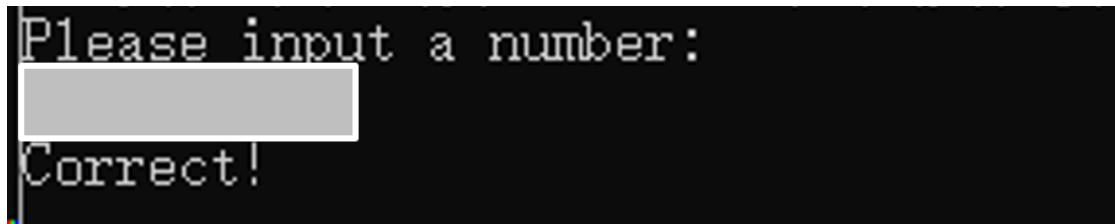


图 6 逆向分析，完成 task2 练习

### 三. 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

```

IDA - task1.exe /Users/kkai/Desktop/Lab8-re_exercises_simple 2/task1.exe

library function Regular function Instruction Data Unexplored External symbol Lumina function
actions
    .text:0040148C push eax ; Arglist
    .text:0040148D push offset Format ; "%s"
    .text:0040148E add esp, 4
    .text:0040148F lea esp, [esp+60h+Arglist]
    .text:00401490 bld esp, 0Ch
    .text:00401491 lea edx, [ecx+1]
    .text:00401492 mov al, [ecx]
    .text:00401493 inc ecx
    .text:00401494 test al, al
    .text:00401495 jnz short loc_4014A1
    .text:00401496 sub edx, edx
    .text:00401497 cmp ecx, 8
    .text:00401498 jz short loc_4014BC
    .text:00401499 push offset aWrongLength ; "Wrong length\n"
    .text:0040149A call sub_401073
    .text:0040149B add esp, 4
    .text:0040149C movsx ecx, [esp+54h+Arglist]
    .text:0040149D movsx edx, [esp+54h+var_53] ; ArgList
    .text:0040149E push esi
    .text:0040149F lea eax, [edx+ecx]
    .text:004014A0 cmp eax, 082h
    .text:004014A1 jnz loc_401596
    .text:004014A2 sub ecx, 1
    .text:004014A3 cmp ecx, 0FFFDDAh
    .text:004014A4 jnz loc_401596
    .text:004014A5 movsx edx, [esp+58h+var_51]
    .text:004014A6 movsx ecx, [esp+58h+var_4F]
    .text:004014A7 lea eax, [edx+dx*2]
    .text:004014A8 sub eax, [eax+ecx*2]
    .text:004014A9 lea eax, 210h
    .text:004014A9 cmp eax, 210h
    .text:004014A9 jnz loc_401596
    .text:004014A9 sub ecx, ds:[edx*8]
    .text:004014A9 sub ecx, eax
    .text:004014A9 cmp ecx, 0FFFFFD3Ch
    .text:004014A9 jnz loc_401596

loc_4014BC:
    .text:0040149B push offset aCorrect ; "Correct\n"
    .text:0040149C call sub_401073
    .text:0040149D add esp, 4
    .text:0040149E mov eax, 1
    .text:0040149F xor eax, eax
    .text:0040149G add esp, 4
    .text:0040149H xor ecx, esp ; StackCookie
    .text:0040149I call j_E_security_check_cookie@4 ; __security_check_cookie(x)
    .text:0040149J add esp, 54h
    .text:0040149K ret

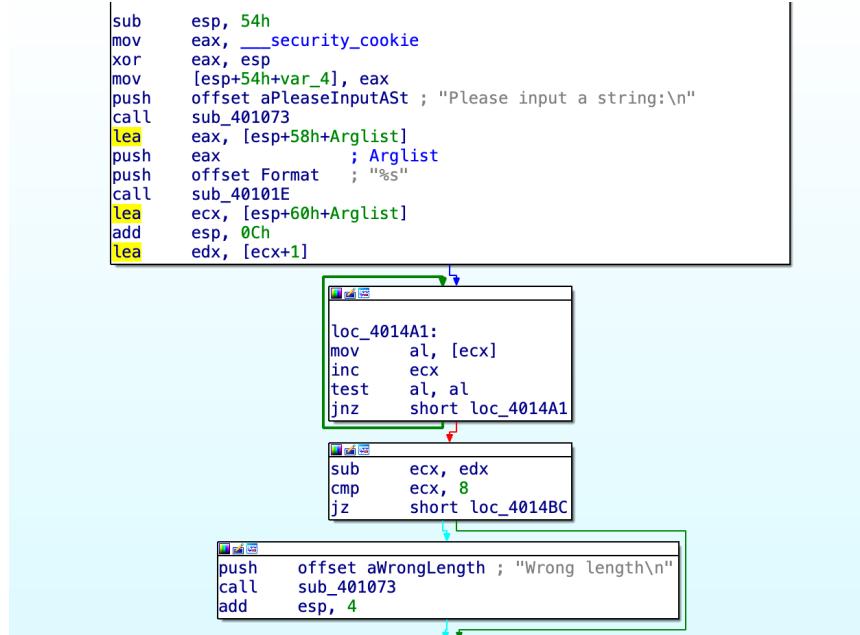
_main_0+2B: _main_0+2B (Synchronized with Hex View-1)

put
decompilation noisy is r3.
use check the Edit/Plugins menu for more information.
FLIRT signature: SEH for vc7-14
waiting type information...

```

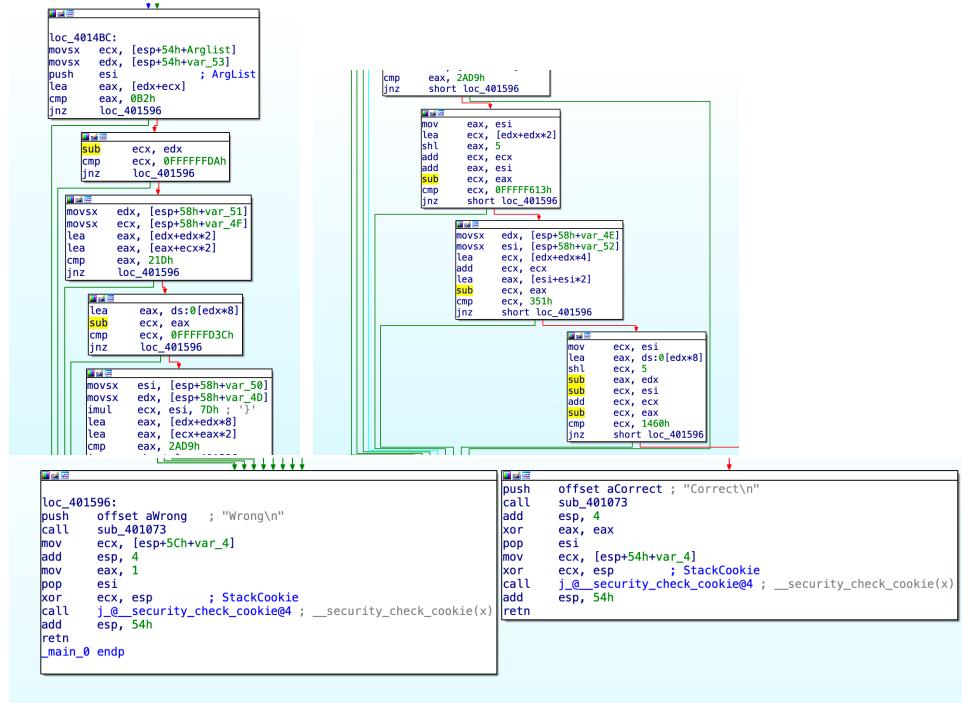
源代码见附件。

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）

$$\begin{cases} x1 + x2 = 178 \\ x1 - x2 = -38 \\ 3x4 + 2x6 = 543 \\ -8x4 + x6 = -708 \\ 125x5 + 18x8 = 10969 \\ -33x5 + 6x8 = -2541 \\ -3x3 + 10x7 = 849 \\ 63x3 - 7x7 = 5216 \end{cases} \quad (1)$$

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：

```
1 import numpy as np
2
3 A = np.array([[1, 1, 0, 0, 0, 0, 0, 0],
4               [1, -1, 0, 0, 0, 0, 0, 0],
5               [0, 0, 0, 3, 0, 2, 0, 0],
6               [0, 0, 0, -8, 0, 1, 0, 0],
7               [0, 0, 0, 125, 0, 0, 18],
8               [0, 0, 0, -33, 0, 0, 6],
9               [0, 0, -3, 0, 0, 0, 10, 0],
10              [0, 0, 62, 0, 0, 0, -7, 0],]) # 创建矩阵A
11 b = np.array([[178], [-38], [541], [-708], [10969], [-2541], [849], [5216]]) # 矩阵b
12 C = np.append(A, b, axis=1) # 增广矩阵
13
14 x = np.linalg.solve(A, b) # 使用 solve 函数求解线性方程组
15
16 characters = ''.join([chr(int(round(i))) for i in x.flatten()]) # 将答案x中的值转化为字符
17 print(f"Characters: {characters}")
18 |
```

从而运行程序，得到答案：

```
/Users/kkkai/PycharmProjects/pythonProject4/venv/bin/python /Users/kkkai/PycharmProjects/pythonProject4/main.py
Characters: FlagStr!
```

在电脑上测试，得到答案：

```
C:\WINDOWS\system32\cmd. x + v

Microsoft Windows [版本 10.0.22621.2715]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Knight>"E:\WeChat Files\wxid_fnn271ccp0o422\FileStorage\File\2023-12\task1.exe"
Please input a string:
FlagStr!
Correct
```

## (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

```
Arglist= byte ptr -4
argc= dword ptr 4
argv= dword ptr 8
envp= dword ptr 0Ch

push    ecx          ; ArgList
push    offset aPleaseInputANu ; "Please input a number:\n"
call    sub_401073
lea     eax, [esp+8+Arglist]
push    eax          ; Arglist
push    offset Format  ; "%u"
call    sub_40101E
mov     eax, dword ptr [esp+10h+Arglist]
add     esp, 0Ch
add     eax, 13AC6D22h
xor    eax, 9BF39868h
sub     eax, 61BACB1Ah
xor    eax, 4A8BD66Ch
add     eax, 74EBDEC3h
xor    eax, 1325A73Dh
add     eax, 217008Eh
xor    eax, 217008Eh
cmp    eax, 0DEADBEEFh
jnz    short loc_4014D1

push    offset aCorrect ; "Correct!\n"
call    sub_401073
add    esp, 4
xor    eax, eax
pop    ecx
retn

loc_4014D1:
push    offset aWrong   ; "Wrong!"
call    sub_401073
add    esp, 4
mov    eax, 1
pop    ecx
retn
_main_0 endp

.text:00401470 ; int _cdecl main_0(int argc, const char **argv, const char **envp)
.text:00401470 _main_0      proc near             ; CODE XREF: _main!j
.text:00401470
.text:00401470 Arglist      = byte ptr -4
.text:00401470 argc        = dword ptr 4
.text:00401470 argv        = dword ptr 8
.text:00401470 envp        = dword ptr 0Ch
.text:00401470
.text:00401470 push    ecx          ; ArgList
.text:00401470 push    offset aPleaseInputANu ; "Please input a number:\n"
.text:00401470 call    sub_401073
.text:00401470 lea     eax, [esp+8+Arglist]
.text:00401470 push    eax          ; Arglist
.text:00401470 push    offset Format  ; "%u"
.text:00401470 call    sub_40101E
.text:00401470 mov     eax, dword ptr [esp+10h+Arglist]
.text:00401470 add     esp, 0Ch
.text:00401470 add     eax, 13AC6D22h
.text:00401470 xor    eax, 9BF39868h
.text:00401470 sub     eax, 61BACB1Ah
.text:00401470 xor    eax, 4A8BD66Ch
.text:00401470 add     eax, 74EBDEC3h
.text:00401470 xor    eax, 1325A73Dh
.text:00401470 add     eax, 217008Eh
.text:00401470 xor    eax, 217008Eh
.text:00401470 cmp    eax, 0DEADBEEFh
.text:00401470 jnz    short loc_4014D1
.text:00401470 push    offset aCorrect ; "Correct!\n"
.text:00401470 call    sub_401073
.text:00401470 add    esp, 4
.text:00401470 xor    eax, eax
.text:00401470 pop    ecx
.text:00401470 retn

loc_401401:
push    offset aWrong   ; "Wrong!"
call    sub_401073
add    esp, 4
mov    eax, 1
pop    ecx
retn
_main_0 endp

00000870 00401470: _main_0 (Synchronized with Hex View-1)
```

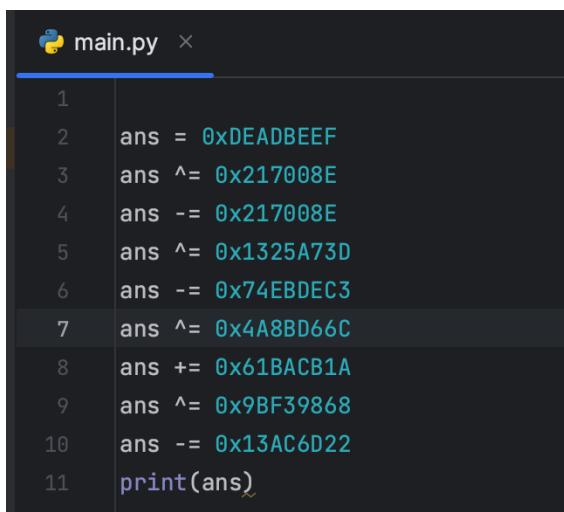
分析代码可知，该程序主要对用户输入的数字进行一系列的异或、加、减运算，最后与 0xDEADBEEFh 相比较，如果相等则输出“Correct！”，不相等则输出“Wrong！”，从而可以对每一步运算进行逆运算，由 0xDEADBEEFh 反推出输入的正确数字。

其函数处理流程如下：

- (1) 输入数字 a
- (2)  $a + 0x13AC6D22$
- (3)  $a \wedge 0x9BF39868$
- (4)  $a - 0x61BACB1A$
- (5)  $a \wedge 0x4A88D66C$
- (6)  $a + 0x74EBDEC3$
- (7)  $a \wedge 0x1325A73D$
- (8)  $a + 0x217008E$
- (9)  $a \wedge 0x217008E$

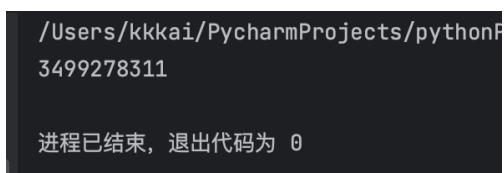
比较最后数字与 0xDEADBEEFh

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



```
main.py
1
2     ans = 0xDEADBEEF
3     ans ^= 0x217008E
4     ans -= 0x217008E
5     ans ^= 0x1325A73D
6     ans -= 0x74EBDEC3
7     ans ^= 0x4A8BD66C
8     ans += 0x61BACB1A
9     ans ^= 0x9BF39868
10    ans -= 0x13AC6D22
11    print(ans)
```

运行程序，得出答案



```
/Users/kkkai/PycharmProjects/pythonProject
3499278311

进程已结束，退出代码为 0
```

运行 task2.exe 程序，输入得到最终结果：

```

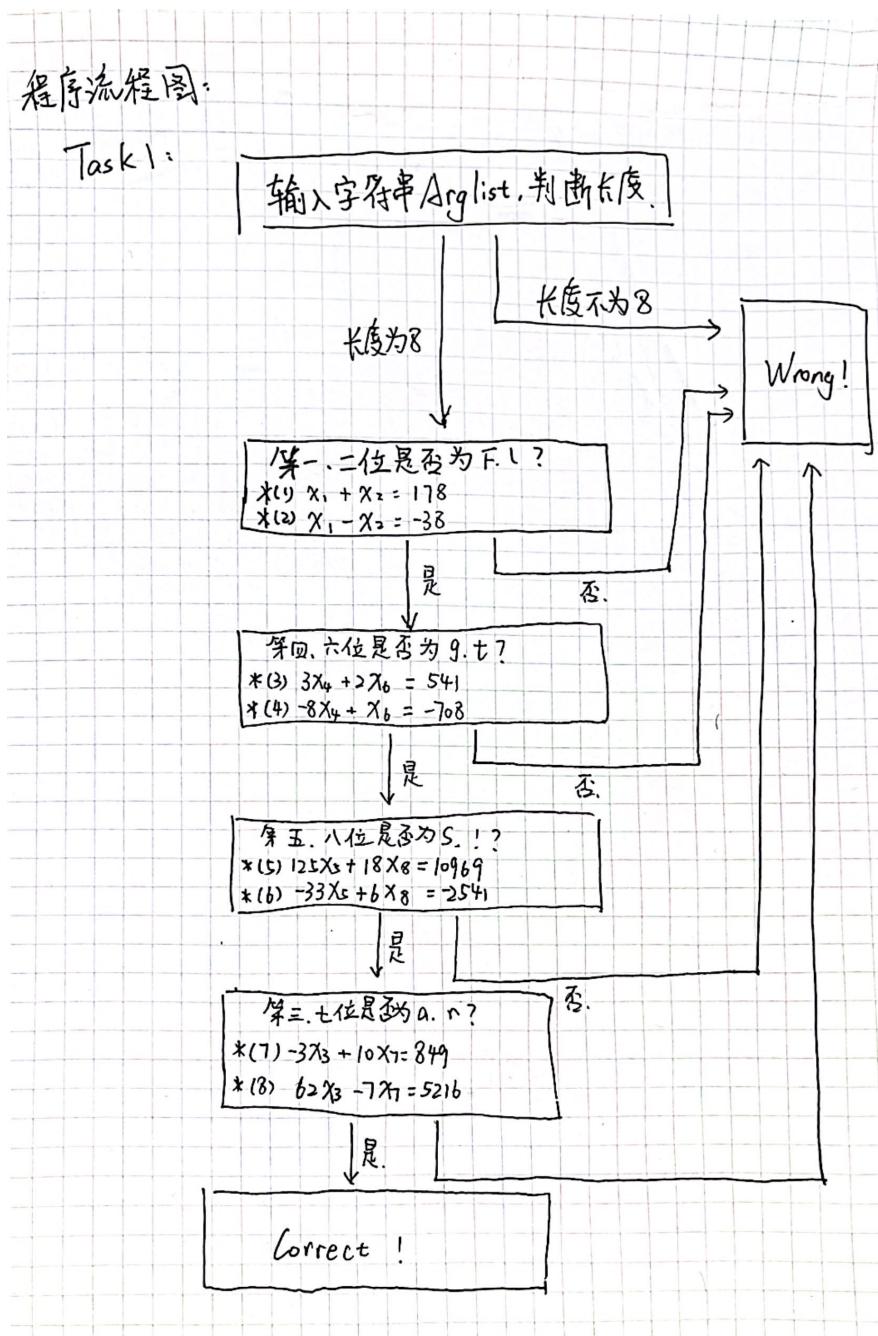
C:\WINDOWS\system32\cmd. × + v
Microsoft Windows [版本 10.0.22621.2715]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Knight>"E:\WeChat Files\wxid_fnn271ccp0o422\FileStorage\File\2023-12\task2.exe"
Please input a number:
3499278311
Correct!

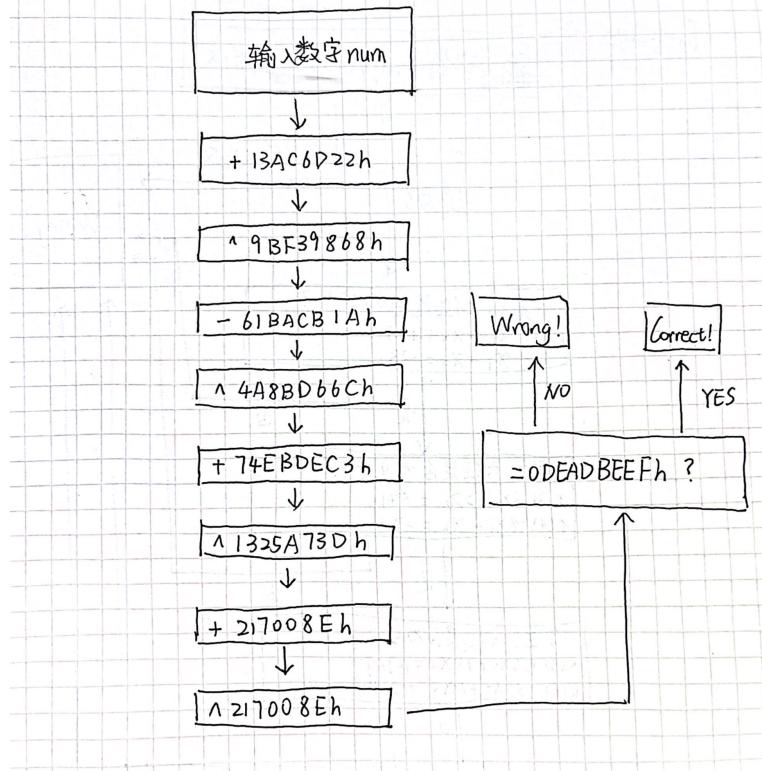
C:\Users\Knight>

```

#### 四. 流程图



Task 2:



## 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)
```