

**汇编语言与逆向技术**

**课程实验报告**

**实验七：Reverse Engineering Challenge**

****

学 院 网络空间安全

专 业 信息安全

学 号 2212998

姓 名 胡博浩

班 级­­­­ 2022级信息安全

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
2. **实验原理**
3. 通过IDA Freeware可以得到二进制代码的反汇编代码，如图1和图2所示。

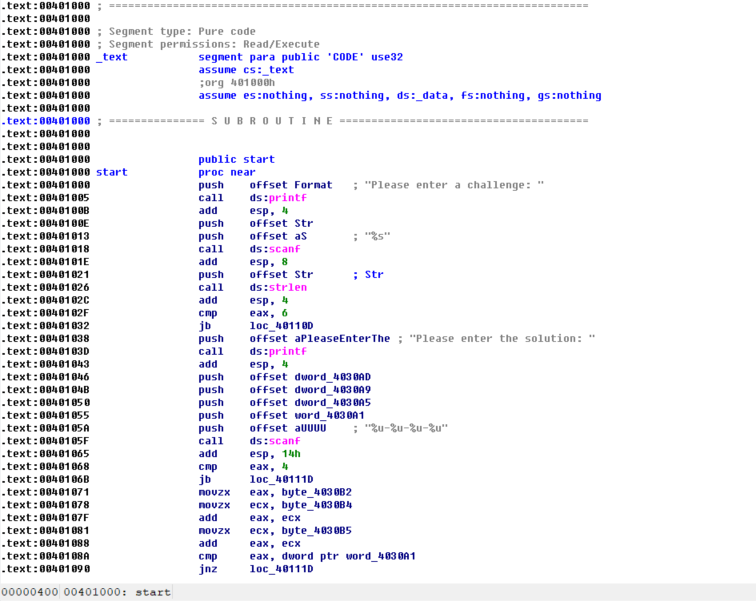


图 1 challenge.exe的反汇编代码

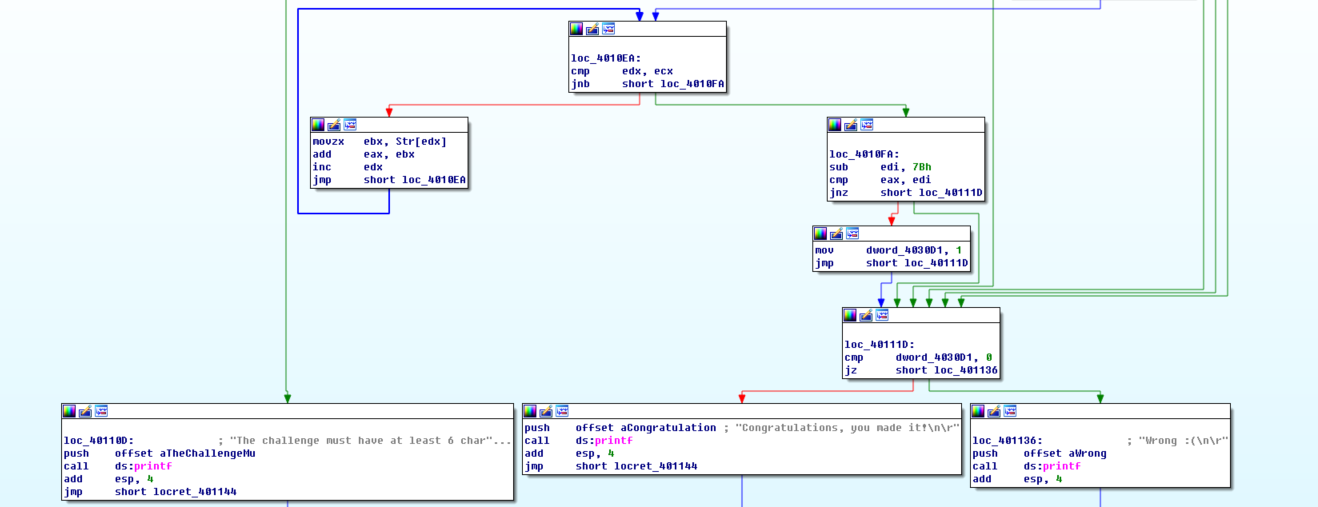


图 2 challenge.exe的反汇编代码的图形化显示

1. 不修改二进制代码，分析汇编代码的计算过程、条件判断、分支结构等信息，逆向推理出程序的正确输入数据，完成逆向分析挑战。

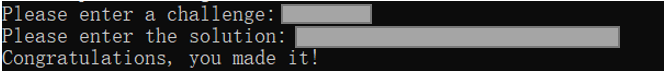
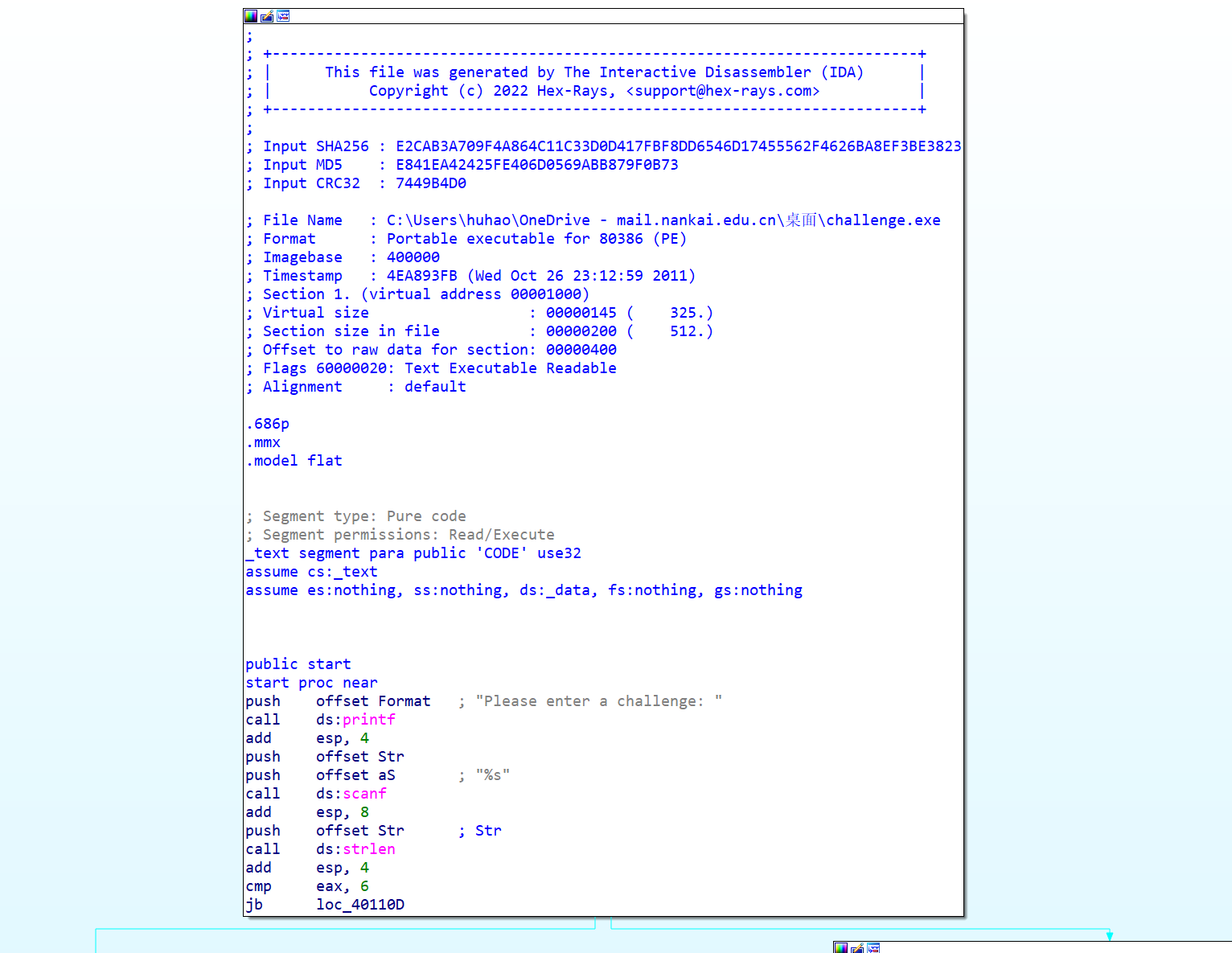
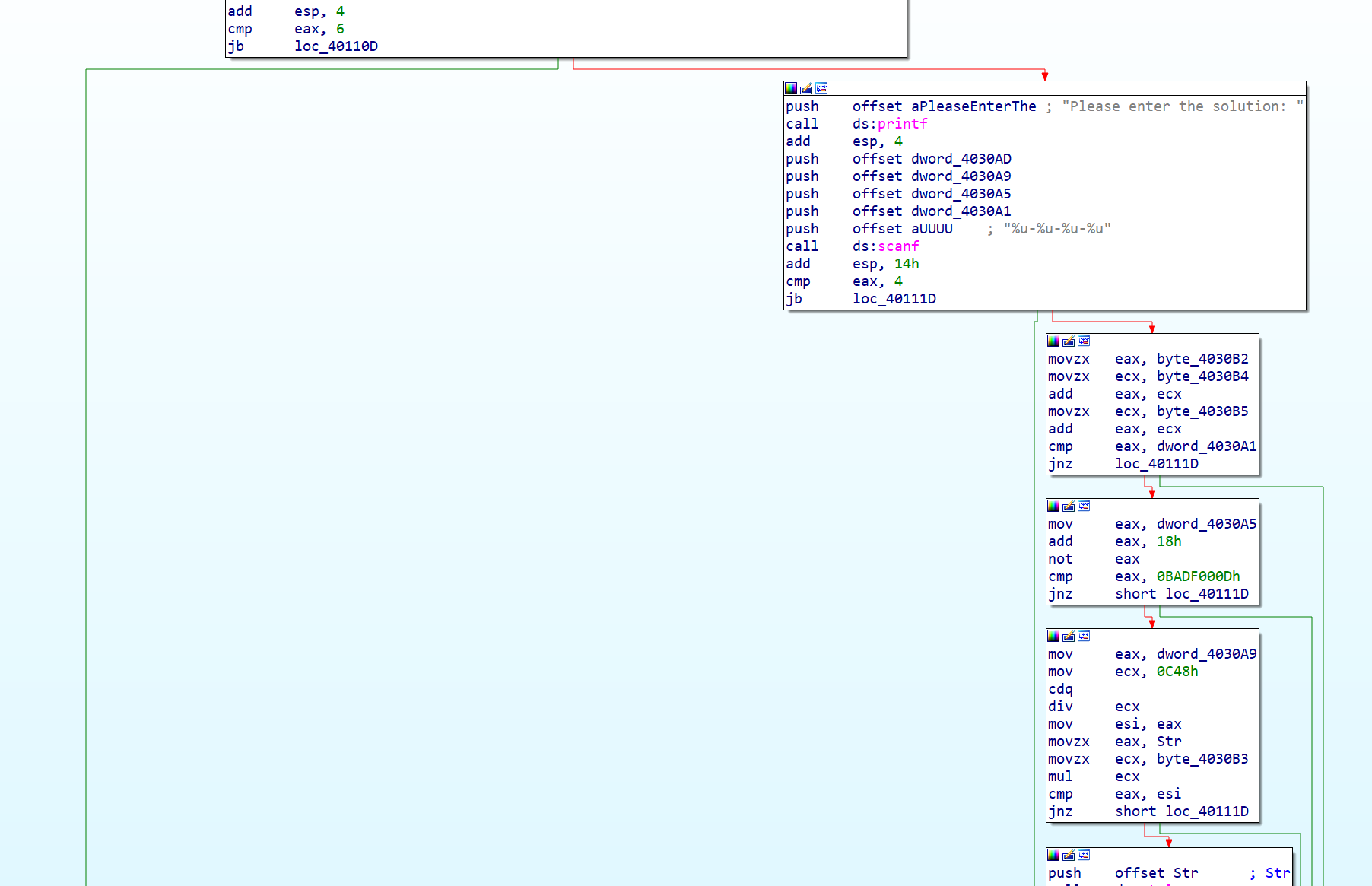
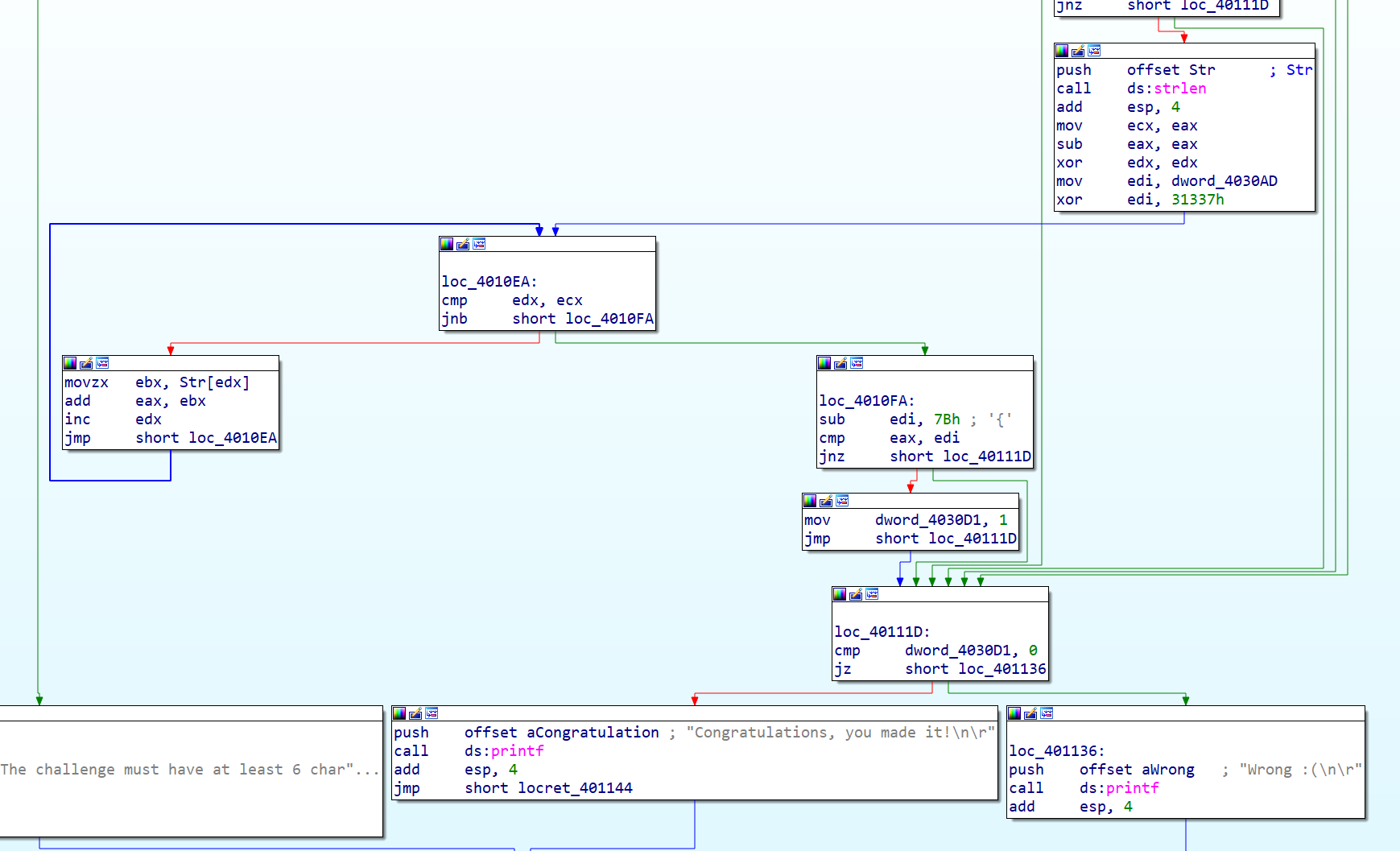


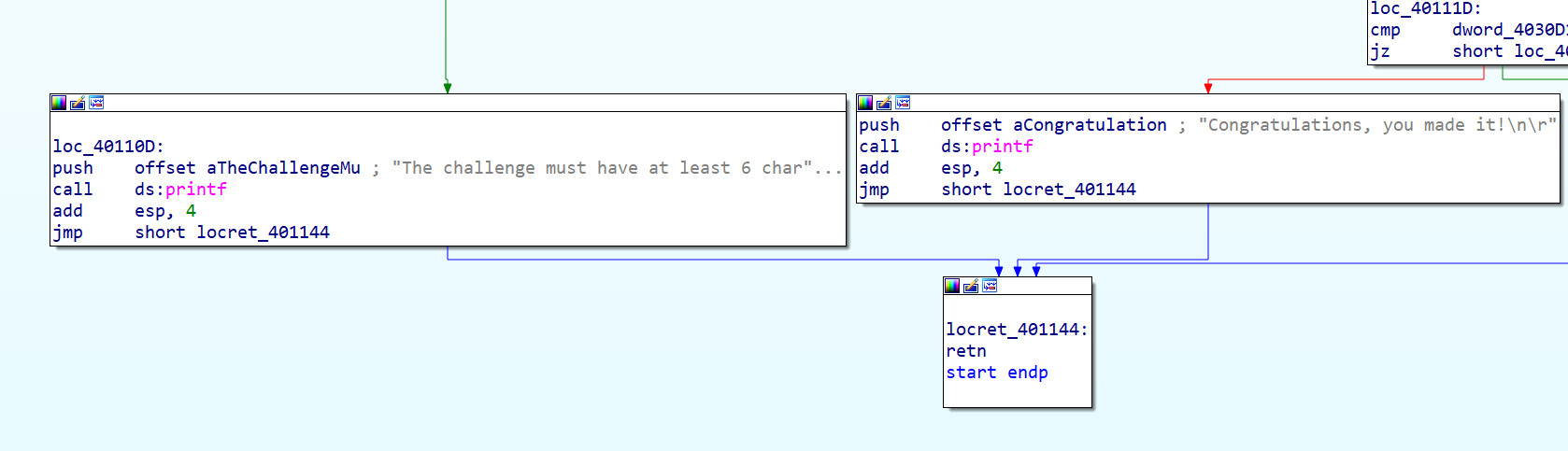
图 3 逆向分析，完成挑战

1. **实验步骤**
2. 使用IDA Freeware，获得二进制代码的反汇编代码，提供截图。









1. 逆向分析二进制代码的计算过程、数据结构、条件判断、分支结构等信息，在实验报告中记录逆向分析的详细过程。

①首先查看数据段，发现如下信息：

1）需要输入一个挑战，且至少6个字母

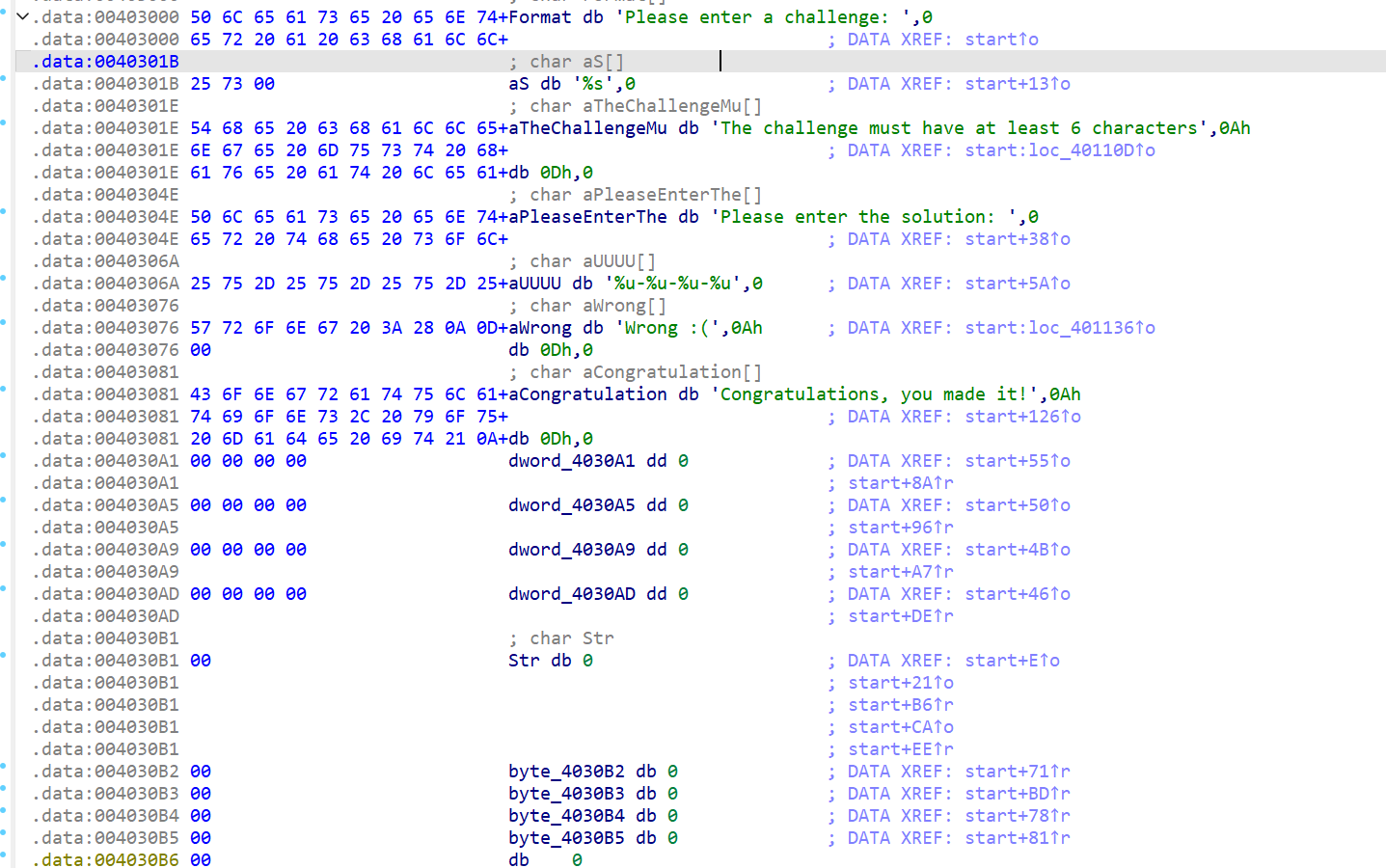
2）需要输入解决方法，并且格式是'%u-%u-%u-%u'

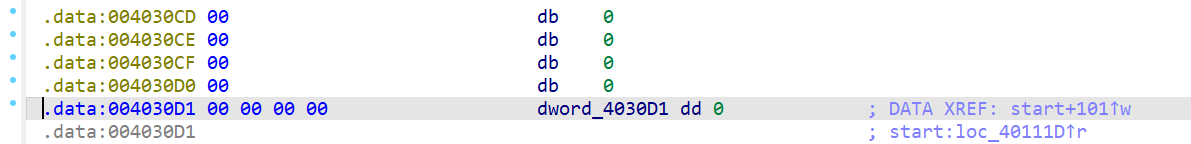
3）似乎经过了一个判断过程，猜测是判断解决方法是否正确。之后弹出成功或者报错信息

4）dword\_4030A1到A5存储了四个双字类型数据，应该是“solution”

5）str，byte\_4030B2到B5存储了五个字节类型数据，应该是“挑战”

6）dword\_4030D1暂时不知道有什么用

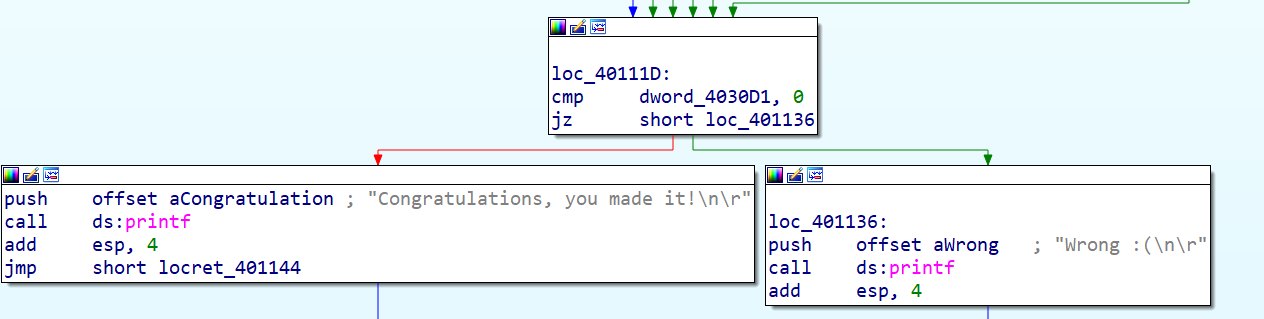




②从后往前，查看怎样才能输出“成功信息”

1）从下图可知，条件为dword\_4030D1 ！= 0

（那个之前“没有用”的数据dword\_4030D1出现了）

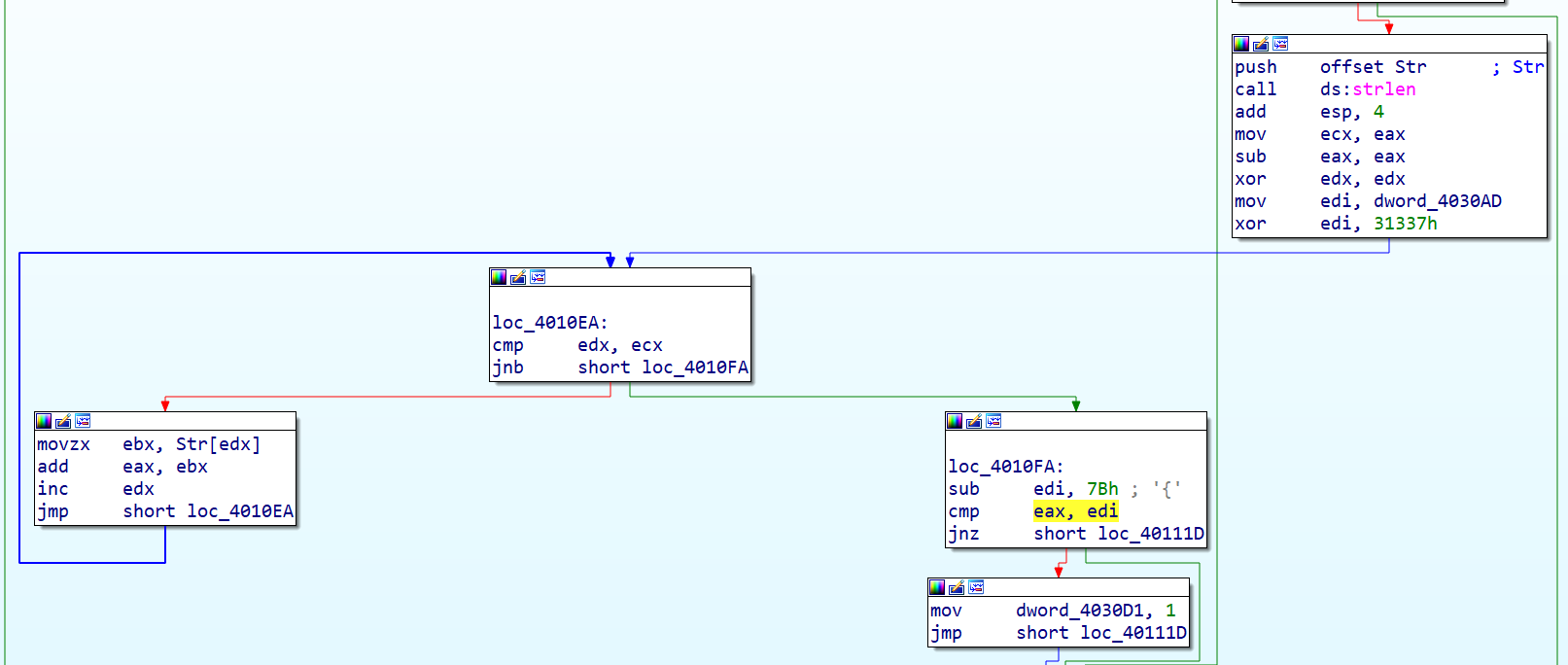


2）查找dword\_4030D1，发现它出现在此处

条件为eax = edi

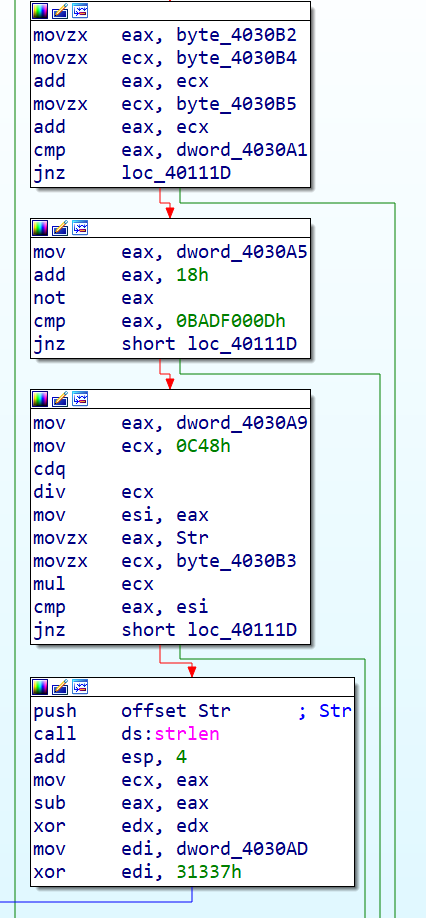
此处经过了一个判断过程，最后判断eax = edi，具体之后再讨论



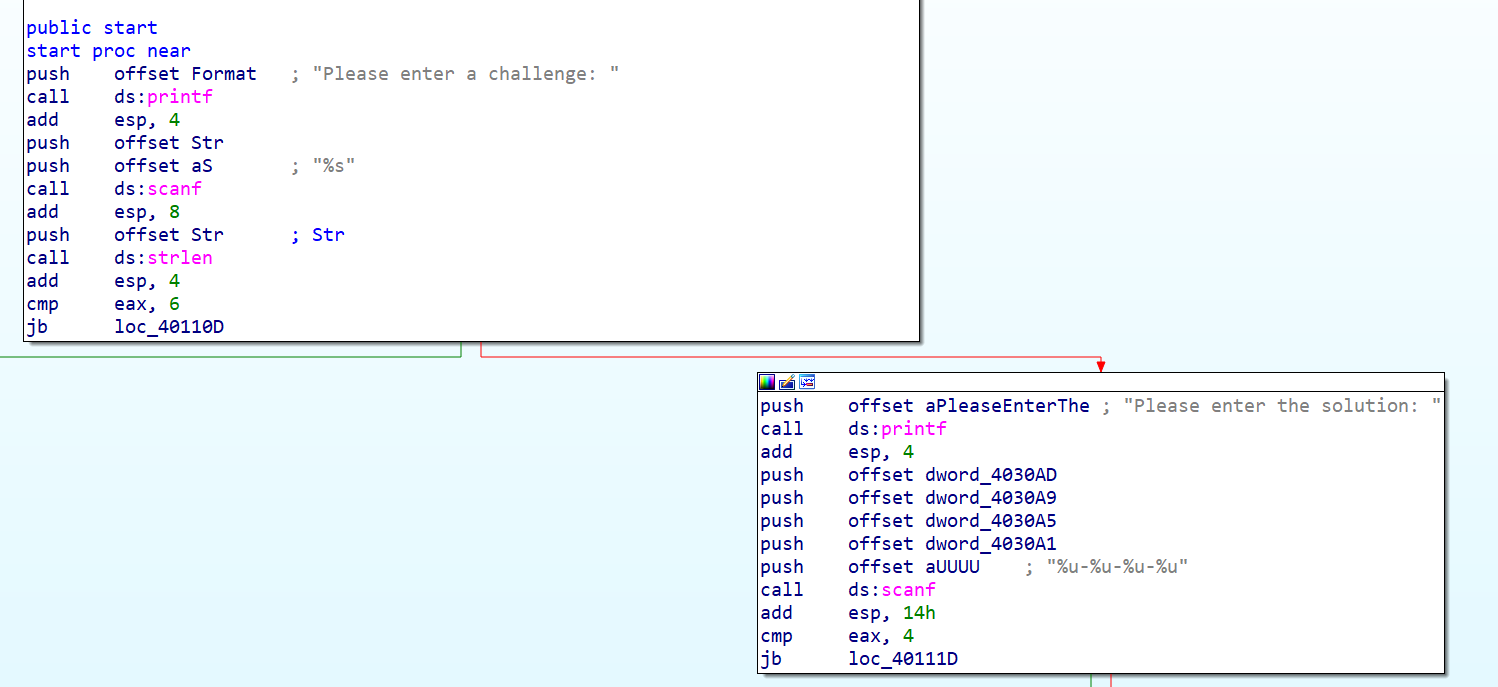


3）再往上是三个判断过程，成功条件分别为：

eax = esi eax = 0BADF000Dh eax = dword\_4030A1



4）到了最前面，发现成功信息的条件和之前获得的一样，即“challenge”至少6个字母，“solution”至少4个字母段



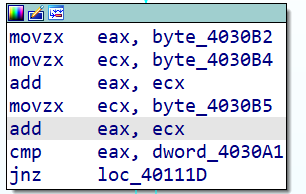
③到此程序的大概脉络已经清晰

接下来具体查看4个条件的判断，根据前面获得的成功条件进行梳理

1）成功条件：eax = dword\_4030A1

即dword\_4030A1 = byte\_4030B2+ byte\_4030B4+ byte\_4030B5

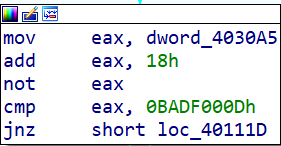
即Str[1] + Str[3] + Str[4] = solution1



2）成功条件：eax = 0BADF000Dh

即dword\_4030A5+18h = 0BADF000Dh

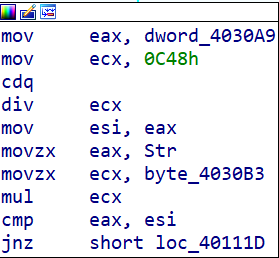
即solution2 = 1159790554



3）成功条件：eax = esi

即Str \* byte\_4030B3 = dword\_4030A9 / 0C48h

即Str[0] \* Str[2] = solution3 / 0C48h



4）成功条件：eax = edi

eax：循环累加字符串字符

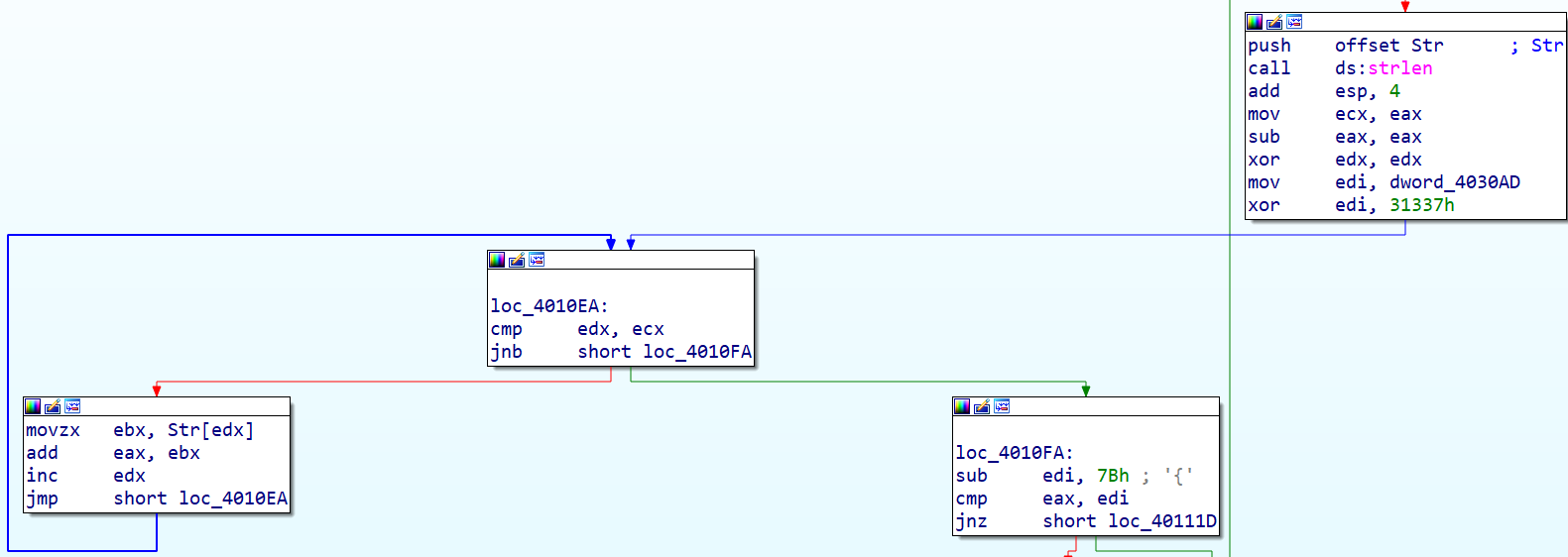
循环体： 将字符串中的每个字符累加到 eax 中。

条件判断： cmp edx, 6，如果 edx < 6，跳转到循环体。

最终 eax 存储了 Str[0] + Str[1] + Str[2] + Str[3] + Str[4] + Str[5]。

edi：(solution4 XOR 31337h) - 7Bh

即xor(solution4, 31337h) - 7Bh = Str[0] + Str[1] + Str[2] + Str[3] + Str[4] + Str[5]



④总结整个程序框架

1）第一部分：输入挑战字符串

条件判断： cmp eax, 6 比较字符串长度和6。

如果字符串长度小于6，跳转到 loc\_40110D，输出错误信息，否则继续执行。

2）第二部分：输入解决方案

条件判断： cmp eax, 4检查成功读入的数据项数是否为4。

如果不是4，跳转到 loc\_40111D，输出错误信息，否则继续执行。

3）第三部分：解决方案的各个部分验证

验证 solution1： Str[1] + Str[3] + Str[4] == solution1

验证 solution2： solution2 == 1159790554

验证 solution3： Str[0] \* Str[2] == solution3 / 0C48h

验证 solution4： xor(solution4, 31337h) - 7Bh == Str[0] + Str[1] + Str[2] + Str[3] + Str[4] + Str[5]（注意：此处str中的数据为字符，所以每个都得加上48）

4）总结

逆向分析的目标是找到满足上述各个条件的输入，使得程序成功执行到最后的成功信息输出分支。

每个条件的验证都需要细致分析，了解字符串处理、数学运算和逻辑运算的细节。

使用逆向工具，跟踪寄存器、内存的变化，可以帮助理解程序执行的过程。

1. 运行程序，根据提示输入字符串和逆向挑战的结果，获得“Congratulations，you made it！”输出，将成功的截图复制到实验报告中。

测试用例为123456

根据四个条件计算出结果155-1159790554-7856869-201351

运行程序，输出成功信息，表明实验成功

