

**汇编语言与逆向技术课程实验报告**

**实验七：Reverse Engineering Challenge**

****

学 院 网络空间安全学院

专 业 信息安全

学 号 2211044

姓 名 陆皓喆

班 级 信息安全

**一、实验目的**

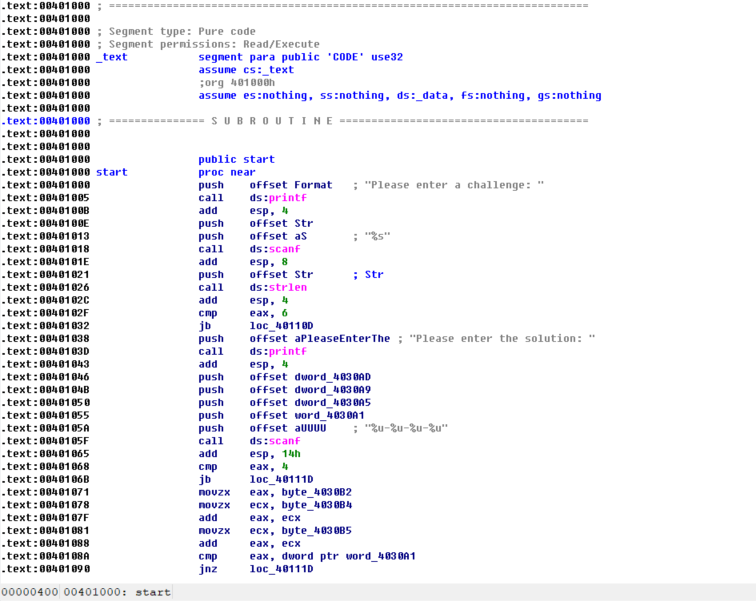
1.熟悉静态反汇编工具IDA Freeware；

2.熟悉反汇编代码的逆向分析过程；

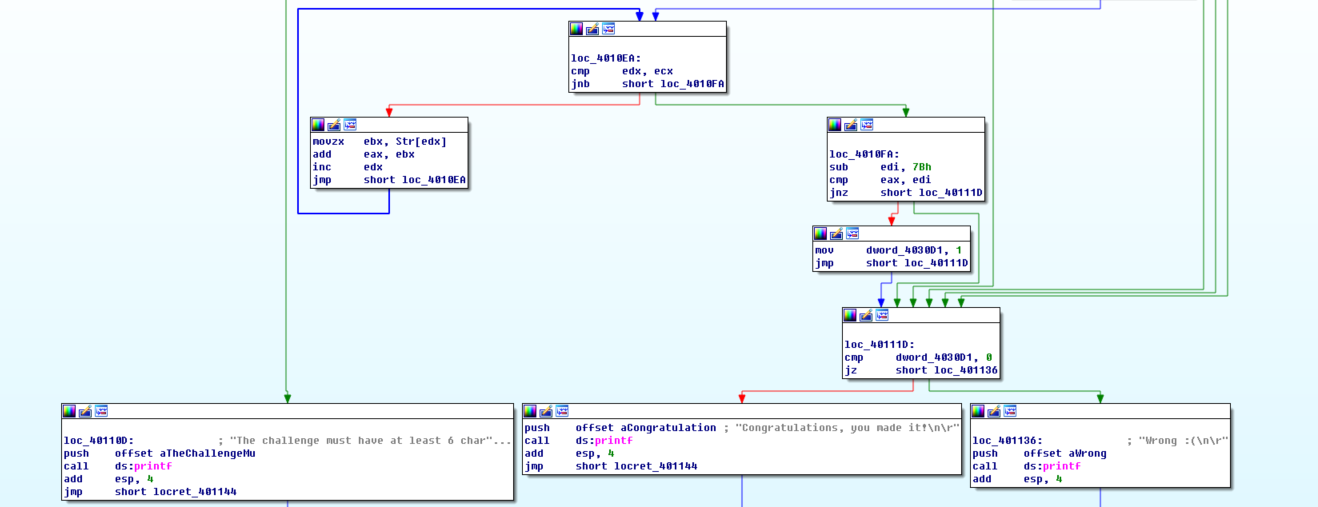
3.掌握反汇编语言中的数学计算、数据结构、条件判断、分支结构的识别和逆向分析。

**二、实验原理**

1.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。

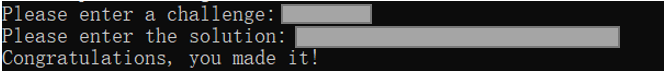


**图 1 challenge.exe的反汇编代码**



**图 2 challenge.exe的反汇编代码的图形化显示**

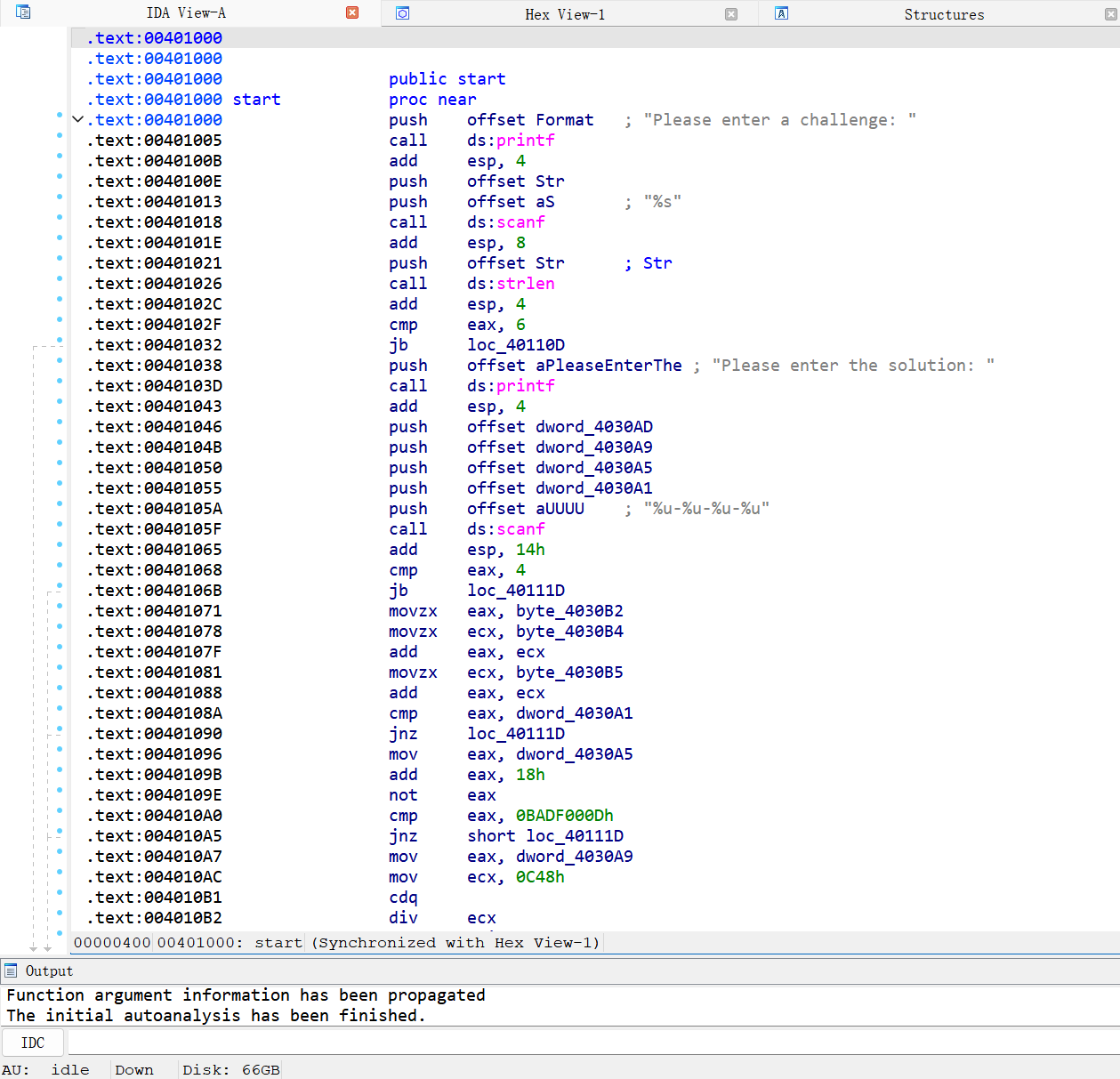
2.不修改二进制代码，分析汇编代码的计算过程、条件判断、分支结构等信息，逆向推理出程序的正确输入数据，完成逆向分析挑战。

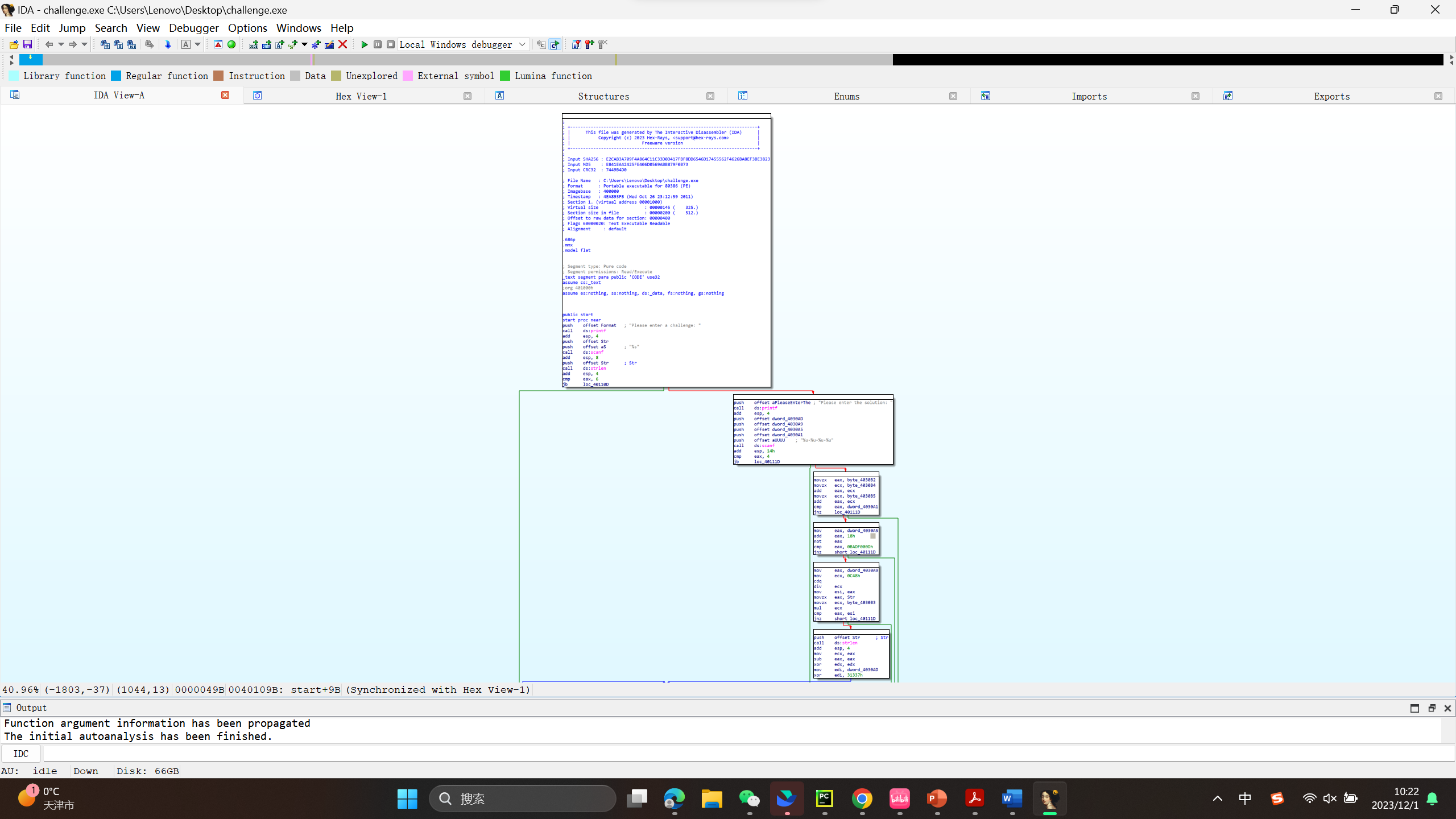


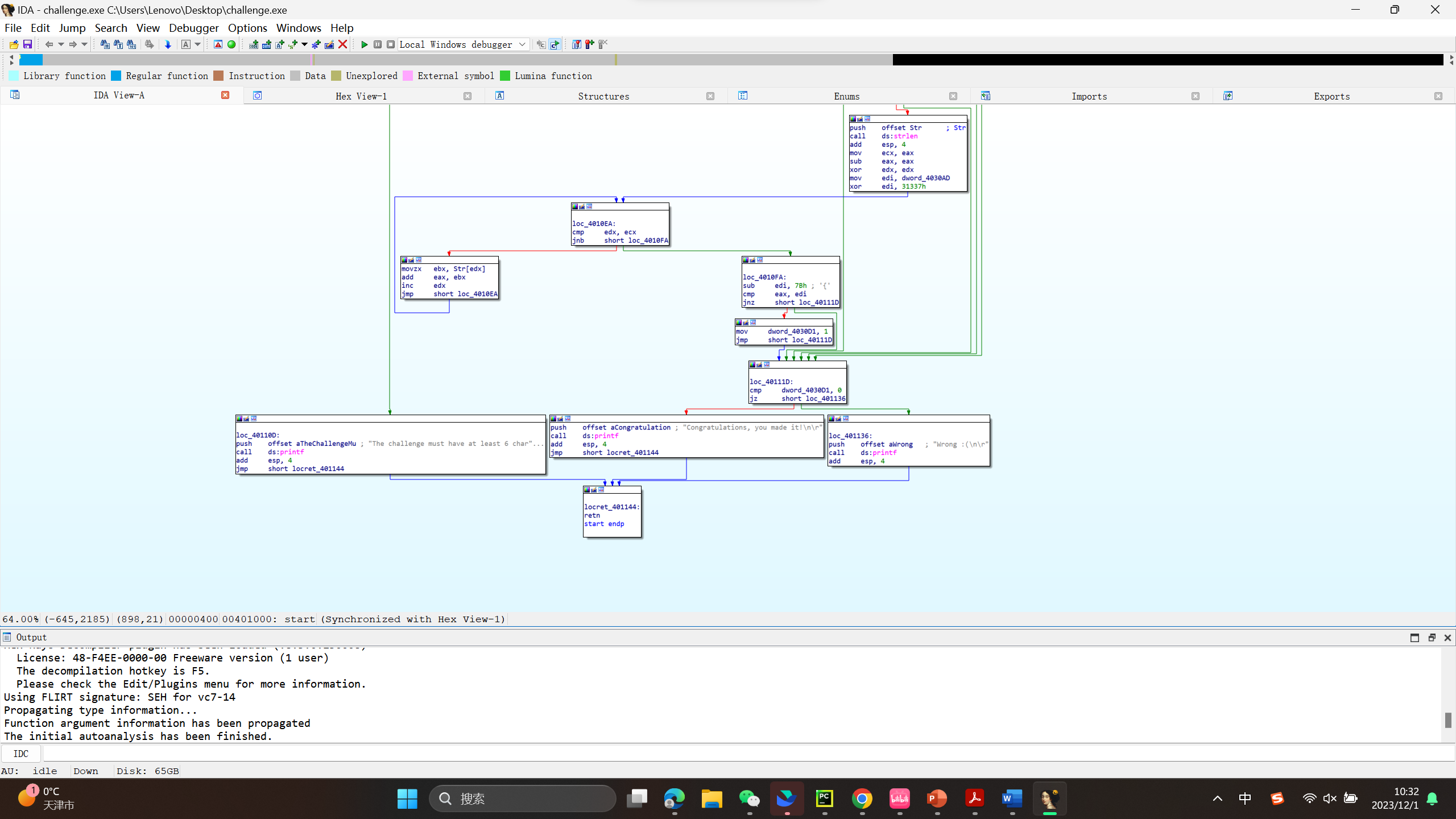
**图 3 逆向分析，完成挑战**

**三、实验报告**

**3.1 使用IDA Freeware，获得二进制代码的反汇编代码，提供截图**

****

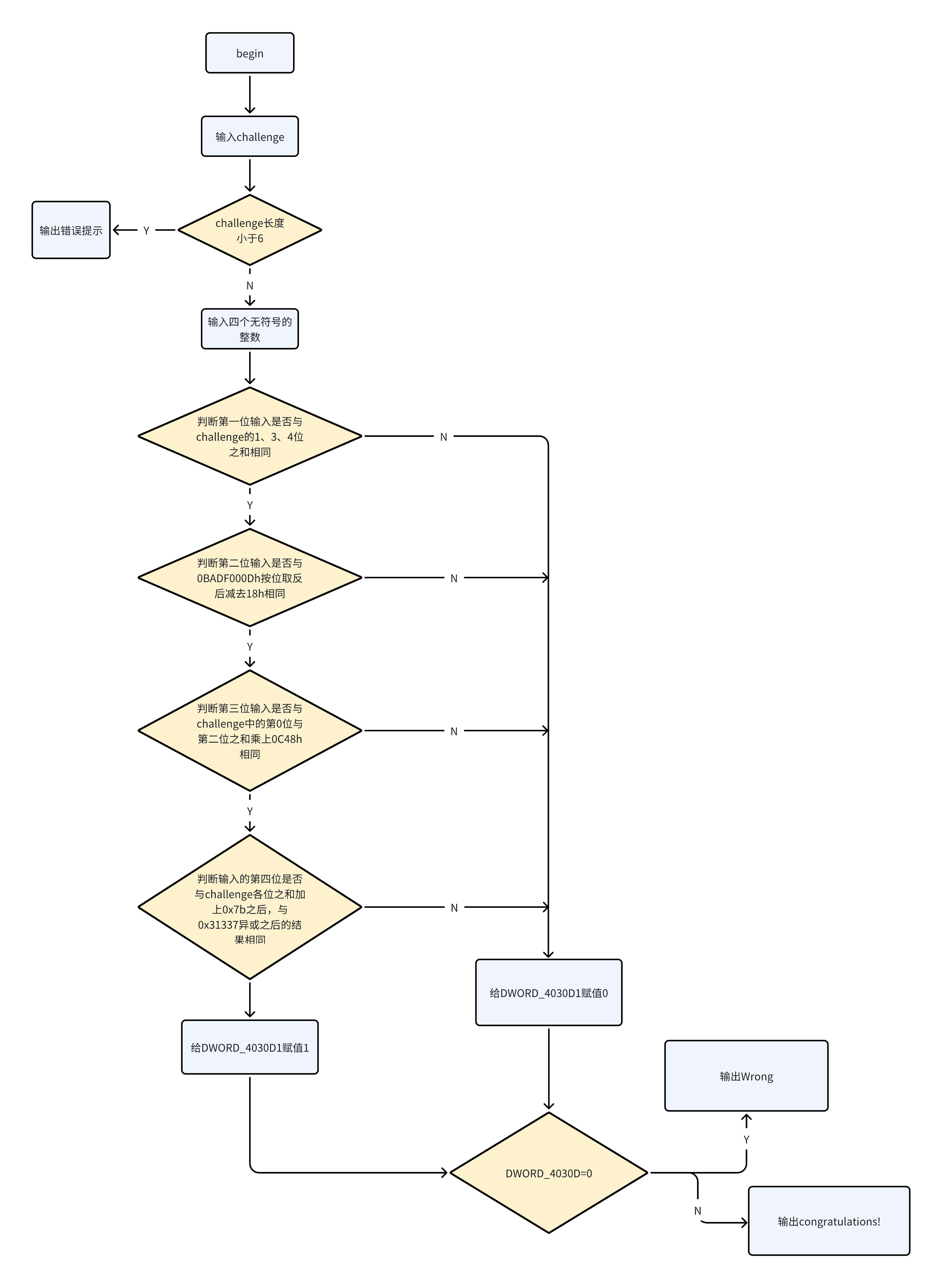




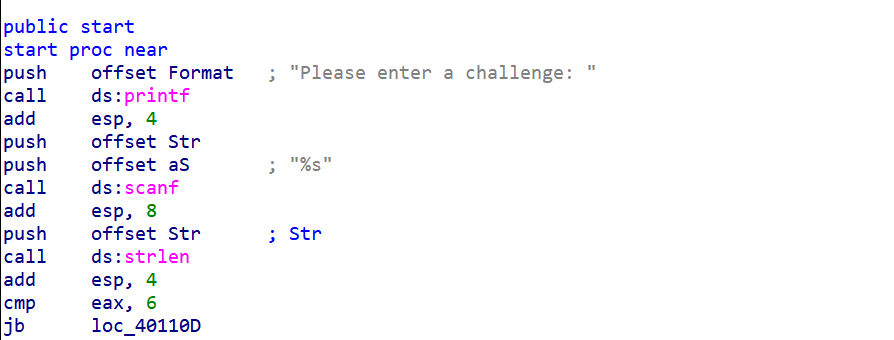
上图就是提供的二进制文件的**反汇编代码**。

**3.2****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**

先根据IDA提供的信息，画出流程表。

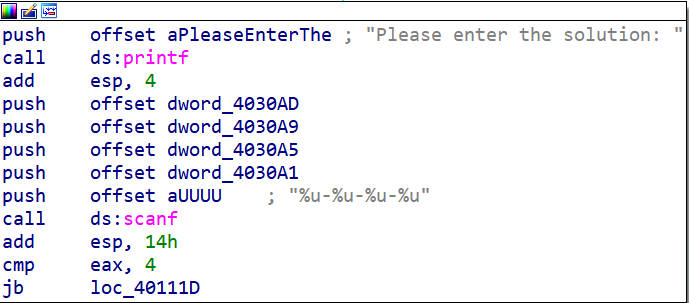


然后再对每一步逆向的结果进行详细的分析。



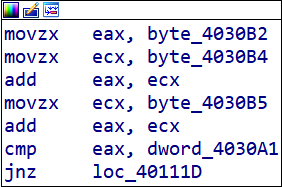
该段代码的功能是：

**输入challenge并判断其长度是否小于6，如果小于6的话就实行跳转到错误信息。**



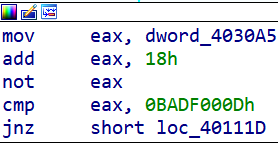
该段代码的功能是：

**依次输入4个数，并且比较输入的数量是否等于4，如果不等于4的话，就跳转到错误信息。**



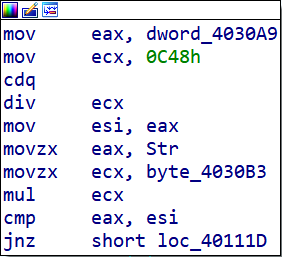
该段代码的功能是：

**将challenge的1、3、4位取出加和，并判断第一位输入的数字是否和其相等。如果不相等，跳转到loc\_40111D。**

****

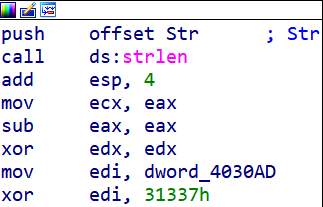
该段代码的功能是：

**将第二位数字加上18h然后按位取反，和BADF000Dh进行比较，如果不相等，跳转到loc\_40111D。**

****

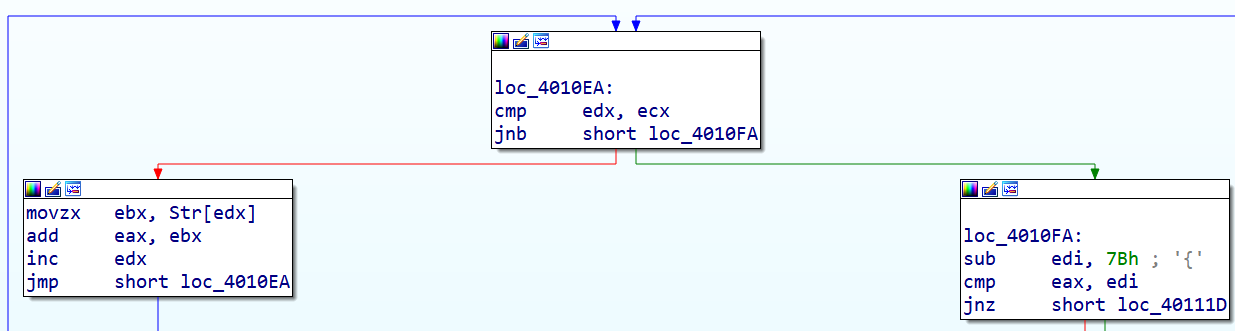
该段代码的功能是：

**将第三位数和0C48h相除，并用edx进行高位拓展，eax存商，edx存余数，将商存到esi中。将challenge 的0，2位相乘。比较esi和eax是否相等，如果不相等，跳转到loc\_40111D。**

****

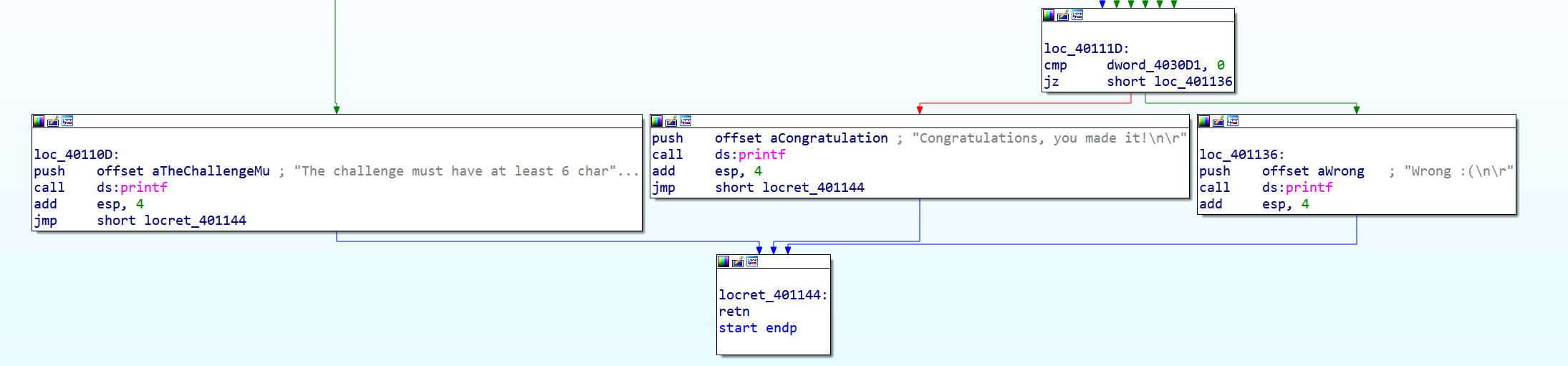
该段代码的功能是:

**ecx中存放challenge的长度，eax和edx全部置零，将第四位数字和31337h进行异或并保存到edi中。**



该段代码的功能是：

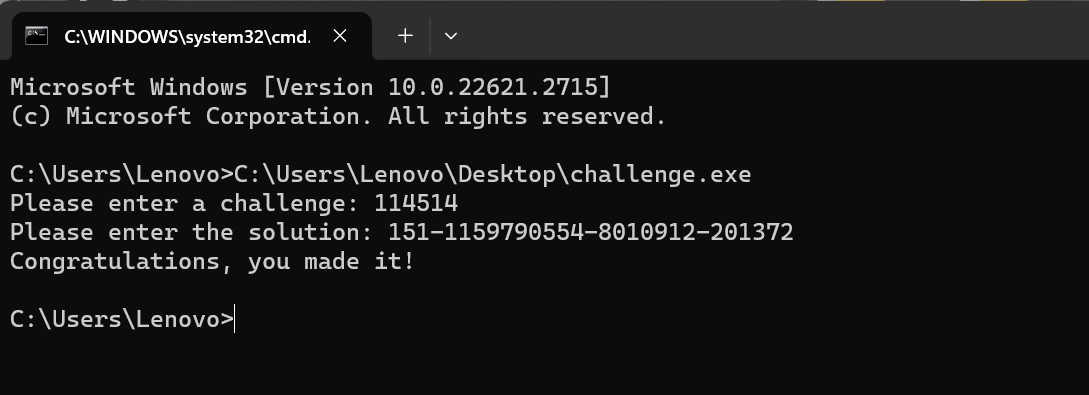
**用eax循环累加challenge 的每一位，并将edi减去7Bh后比较eax和bdi是否相等，如果不相等跳转到loc\_40111D。**

****

该段代码的功能是：

**当以上条件均满足的时候，会正常进行 mov dword\_4030D1,1的操作，此时loc\_40111D中的判断不能满足，正常输出Congratulations, you made it!。如果又任何一步不满足，就会直接跳转到dword\_40111D ，而此时dword\_3040D1 的值为0，导致判断满足条件，跳转到loc\_401136 输出Wrong:(。**

**3.3** **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**

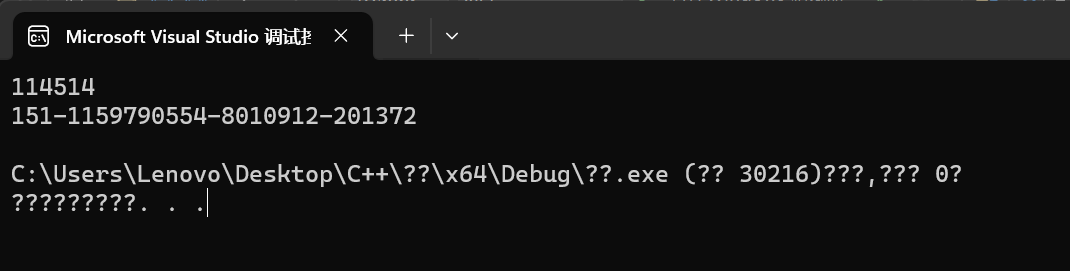
****

为了更好的解出结果，我简单地编写了一个C++程序来解出结果，相当于一个反方向的过程。

C++代码如下所示：

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. string str;
6. cin >> str;
7. int len = str.length();
8. unsigned int num1 = 0, num2 = 0, num3 = 0, num4 = 0;
9. num1 = str[1] + str[3] + str[4];
10. num2 = ((~0xBADF000D) - 0x18);
11. num3 = (str[0] \* str[2]) \* 0x0c48;
12. for (int i = 0; i < len; i++)
13. num4 += str[i];
14. num4 = (num4 + 0x7b) ^ 0x31337;
15. cout << num1 << "-" << num2 << "-" << num3 << "-" << num4 << endl;
16. return 0;
17. }

用该程序就可以解出最后的结果，使其输出“Congratulations,you made it!”

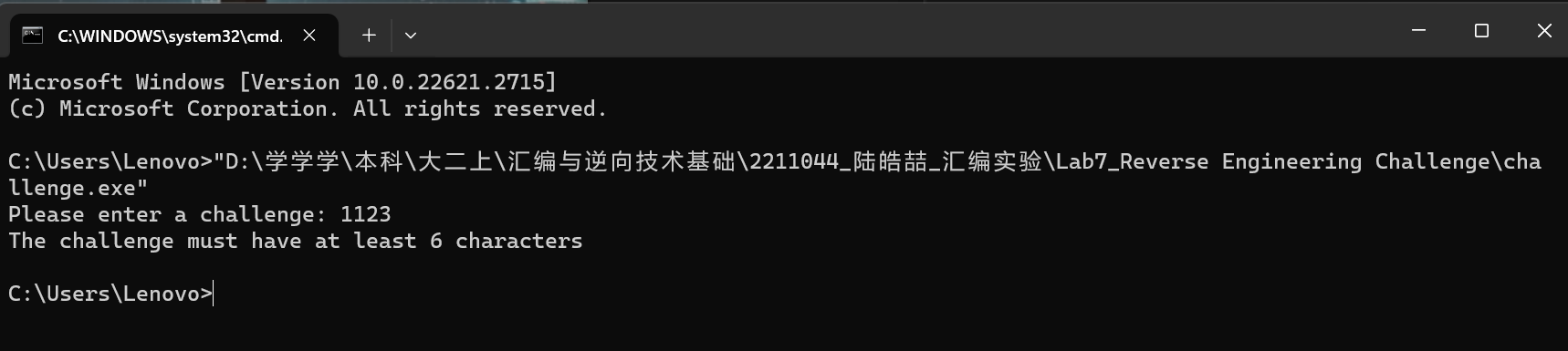


**（如上图所示，输入114514即可输出我们想要得到的结果）**

**四、自我思考**

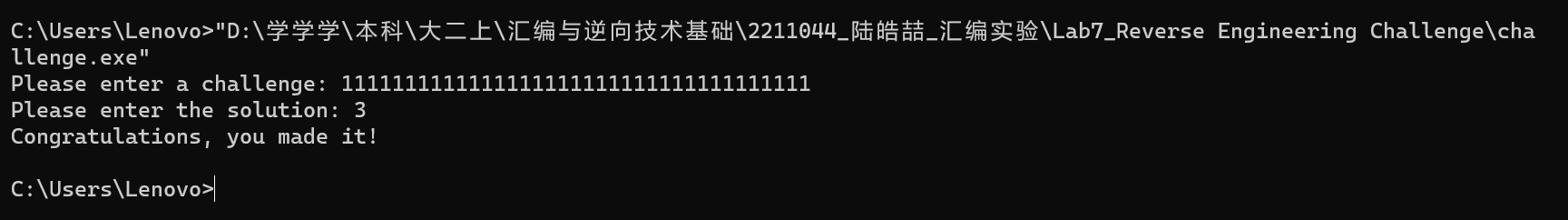
今天的实验课，我通过静态逆向分析软件IDA完成了对老师提供的exe文件的逆向分析，并且得出了最后的正确的结果。

当然在做的时候也发现了不少问题，比如说在输入challenge的时候，我发现输入6个以下的字符的时候，会报错，输出“The challenge must have at least 6 characters”。

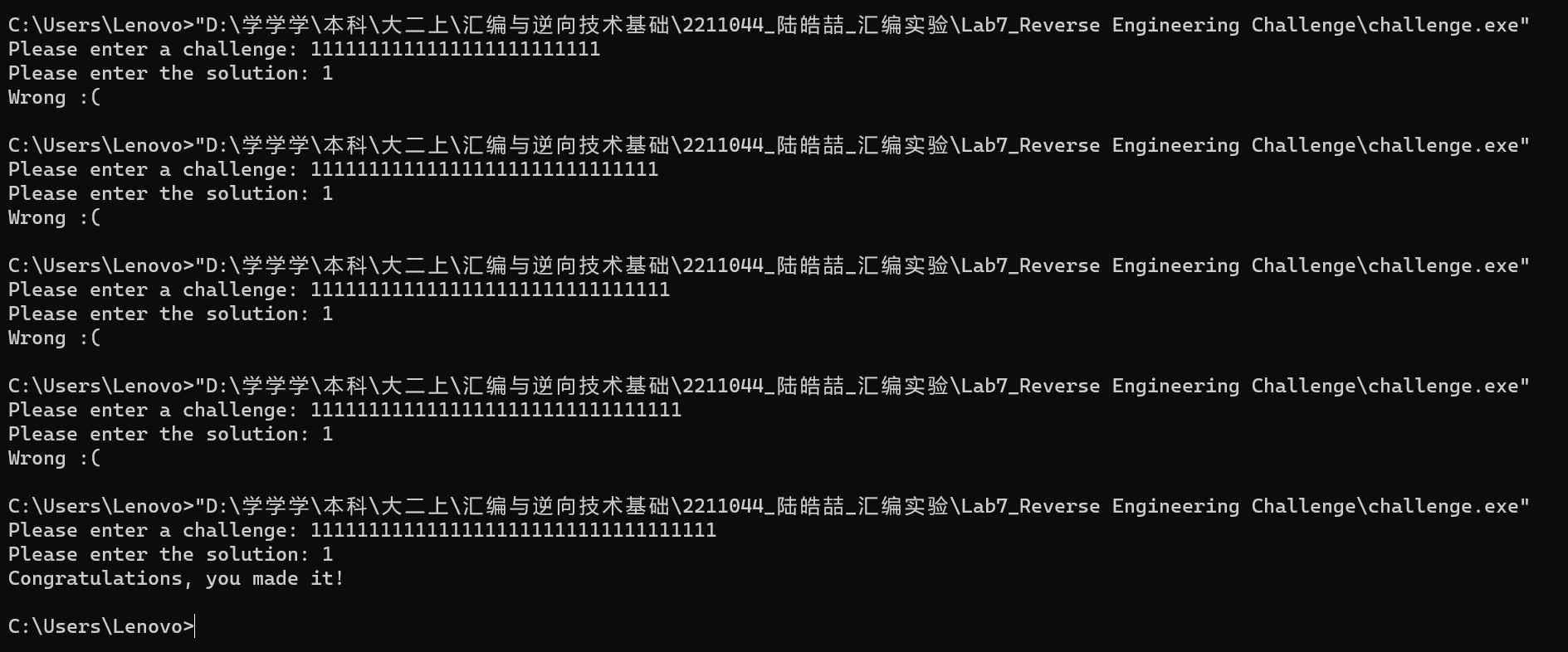


只能是输入6个字符，才能让程序启动。

但是当我输入许多个字符的时候，不管输入什么solution，都会输出“Congratulations, you made it!”（如下图）。



我经过几次测试，发现当输入的challenge长度为36或者以上时，都会输出正确的结果。



经过查看内存空间，我发现：当challenge的长度大于等于36时，会使得输出结果的内存空间被覆盖，导致其直接输出1，使最后的结果不管怎么样都会输出正确的“Congratulations, you made it!”。这其实是该程序的一个漏洞，可以通过限制其输入长度或者改变输出结果内存存储位置去解决这个问题。



**（如图，查看输出结果的内存空间）**

希望在接下来的逆向分析课程中，能够继续深入学习反汇编的一些知识。