

**汇编语言与逆向技术**

**课程实验报告**

**实验九：Reverse Engineering Exercises – Advanced**

****

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. 进一步熟悉静态反汇编工具IDA Freeware；
   2. 熟悉将反汇编代码进行反编译的过程；
   3. 掌握对于反编译伪代码的逆向分析；
   4. 运用熟悉的编程语言，实现简单的脚本编写
2. **实验原理**
3. **task3**
4. 通过IDA Freeware得到task3.exe的反汇编代码，如图1和图2所示。

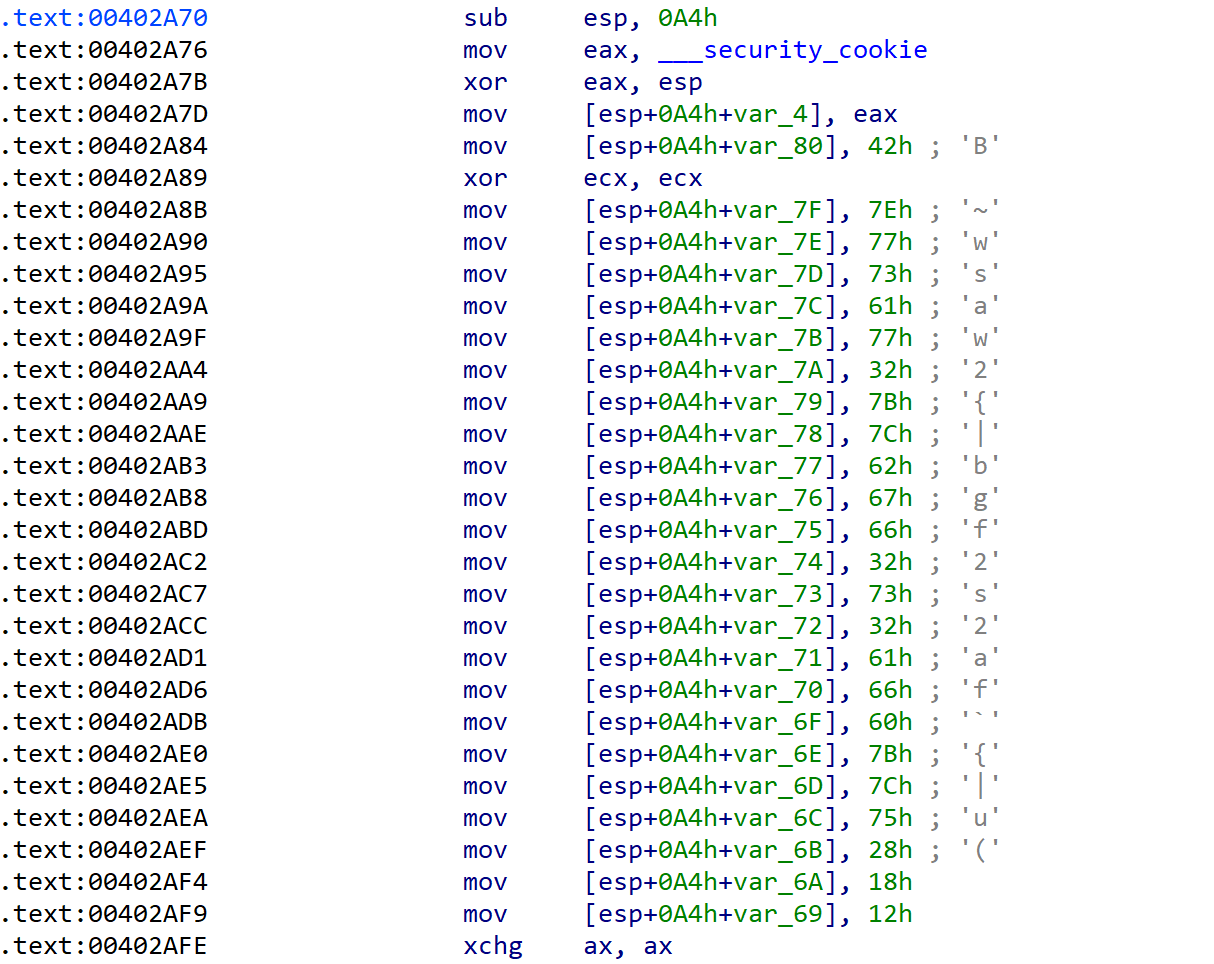


图1 task3.exe的反汇编代码

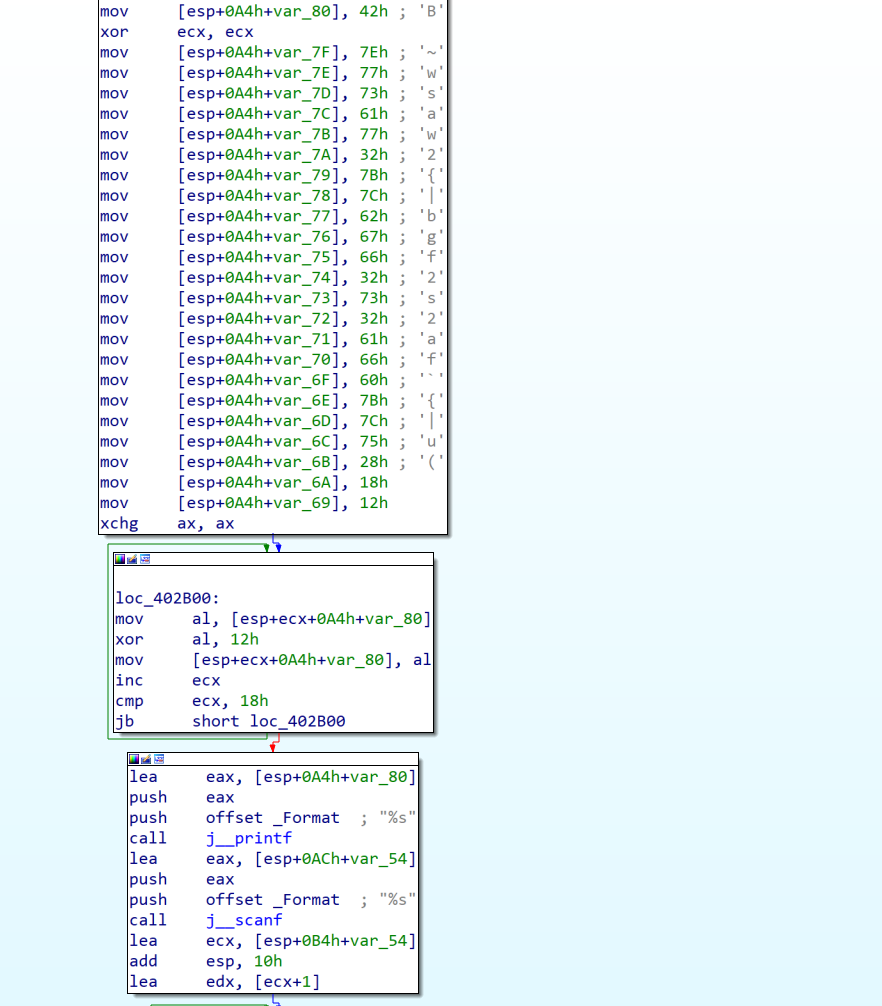


图 2 task3.exe反汇编代码的图形化显示

1. 使用IDA的反编译功能（F5快捷键）得到伪代码，如图3所示。

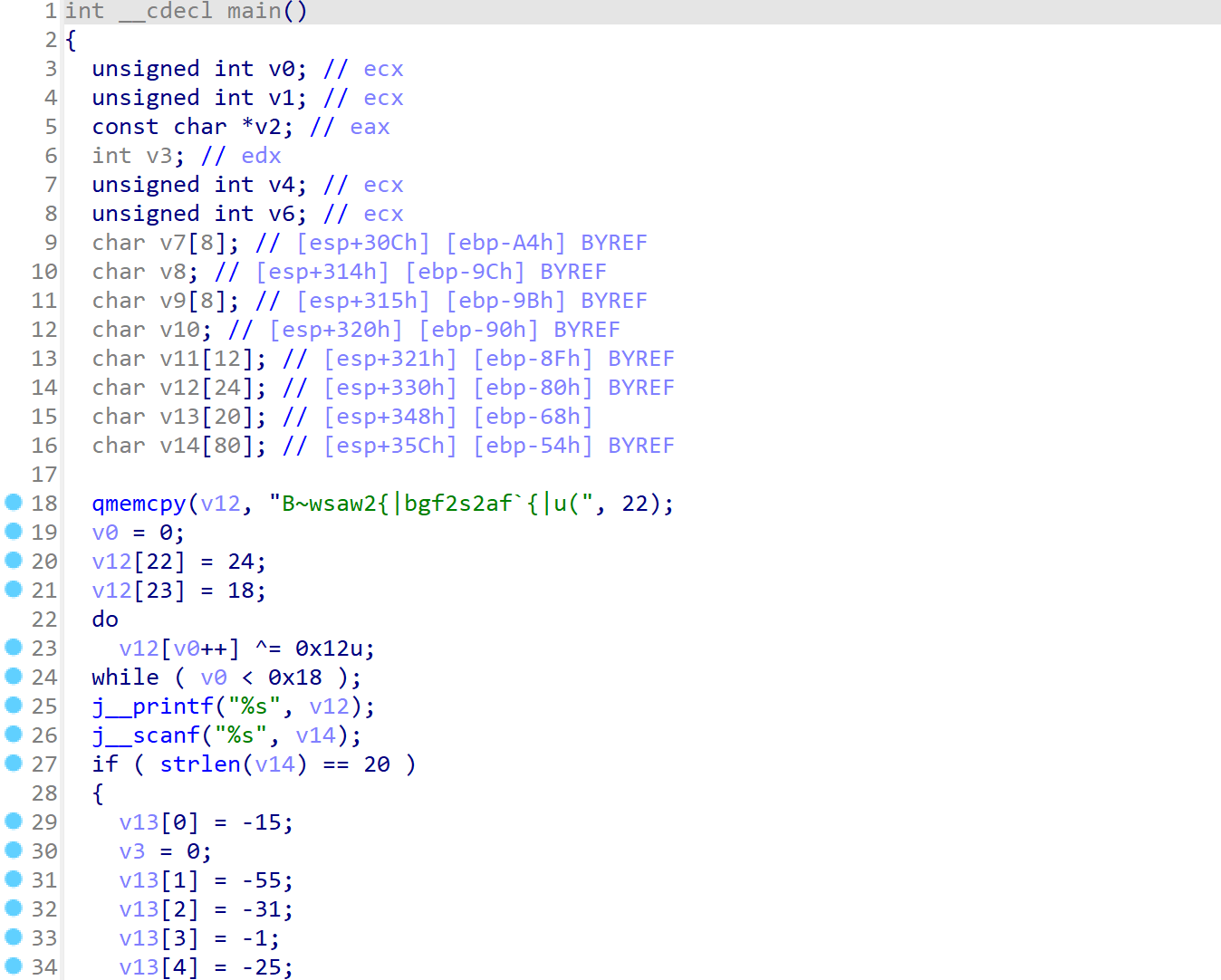


图 3 task3.exe的反编译伪代码

1. 通过对反汇编命令及反编译伪代码的分析，逆向推理出待输入字符串的计算公式
2. 使用熟悉的编程语言（C++、Java、Python等）对待输入字符串进行计算，完成逆向分析挑战。

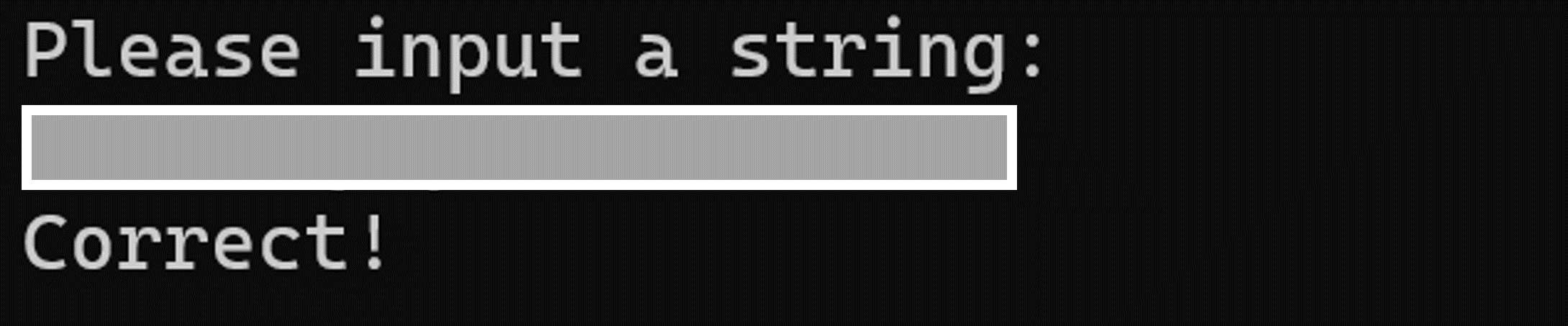


图 4 逆向分析，完成task3练习

1. **task4**
2. 通过IDA Freeware得到task4.exe的反汇编代码，如图5和图6所示。

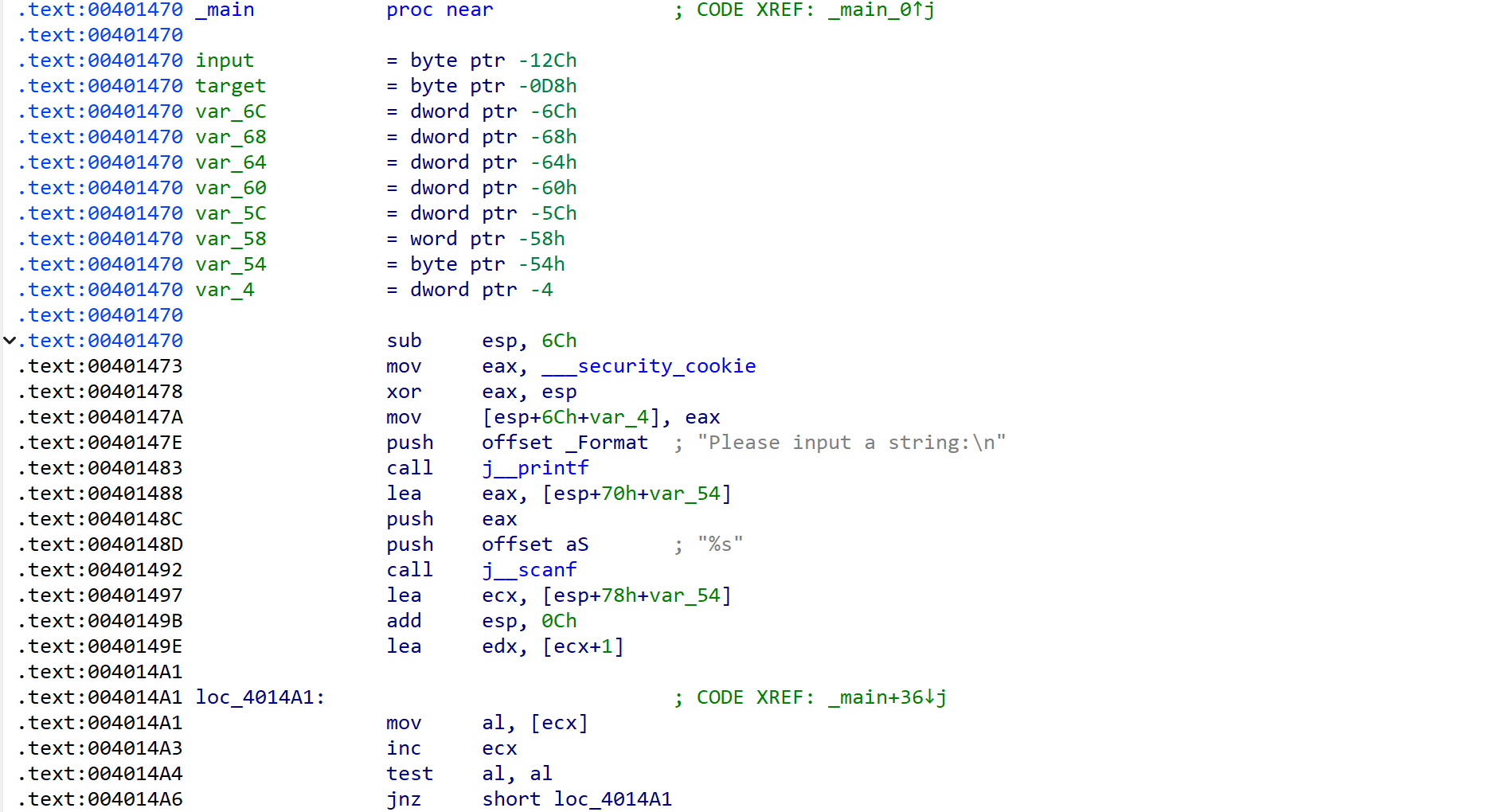


图 5 task4.exe的反汇编代码

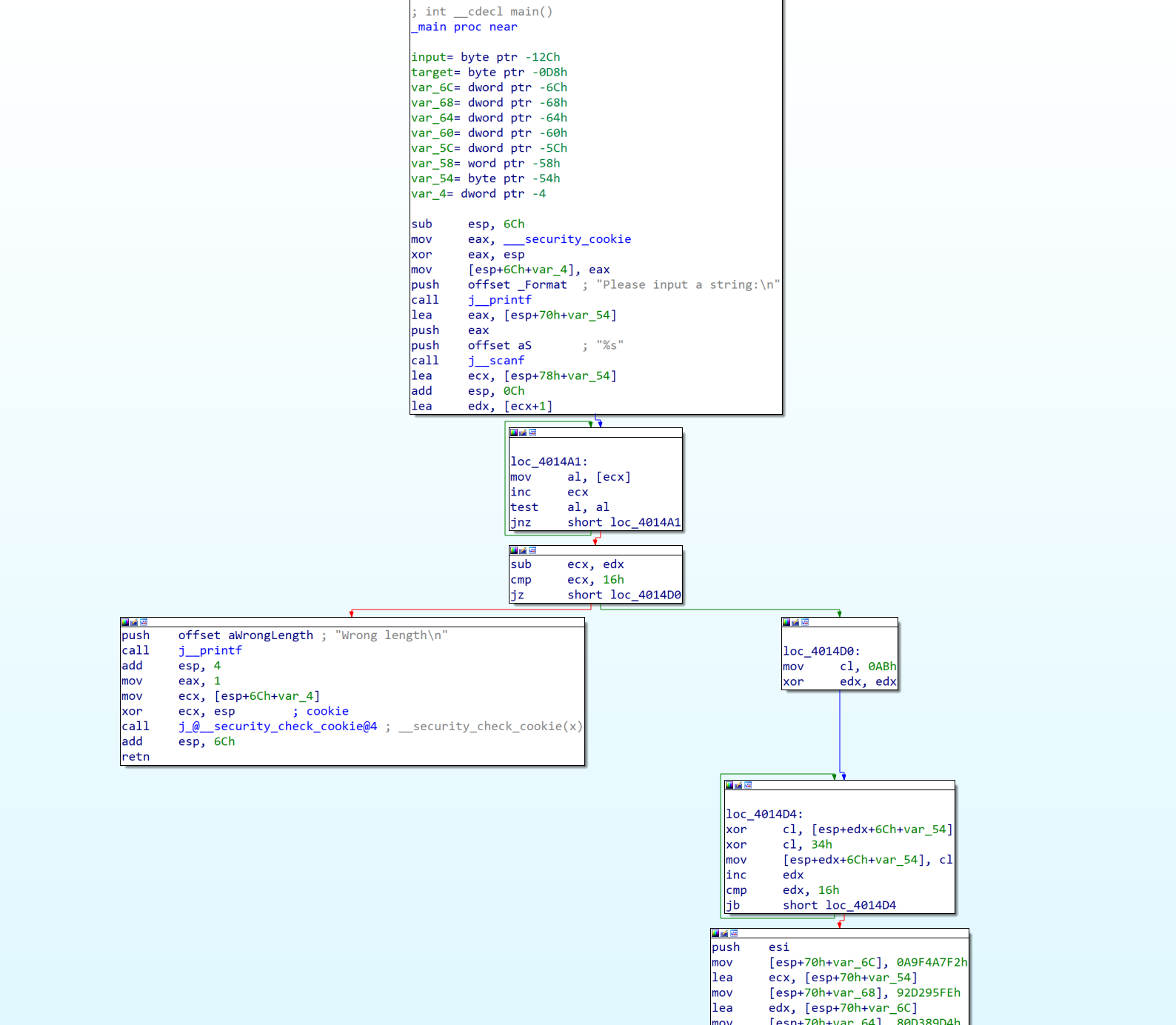


图6 task4.exe反汇编代码的图形化显示

1. 使用IDA的反编译功能（F5快捷键）得到伪代码，如图7所示。右键点击数字对象可实现数制转换。

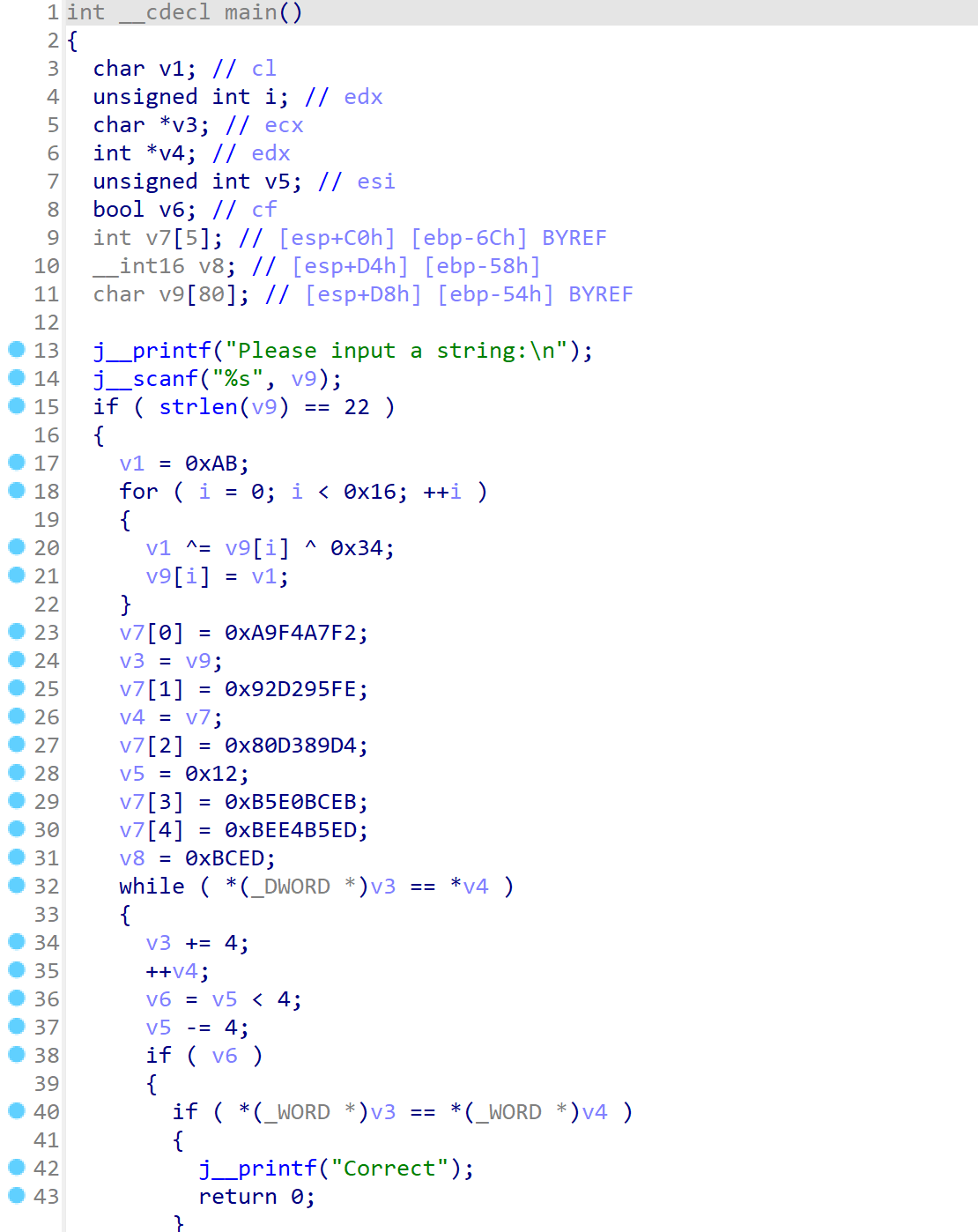


图7 task4.exe的反编译伪代码

1. 通过对反汇编命令及反编译伪代码的分析，逆向推理出待输入字符串的计算公式
2. 使用熟悉的编程语言（C++、Java、Python等）对待输入字符串进行计算，完成逆向分析挑战。



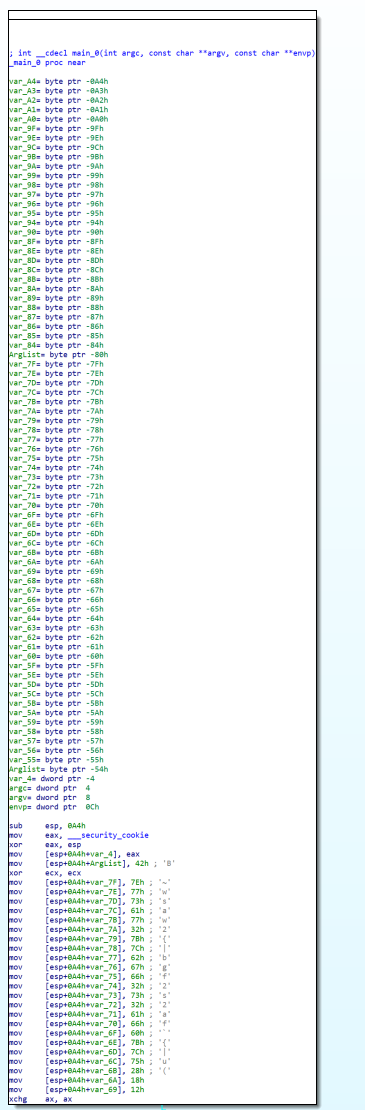
图8 逆向分析，完成task4练习

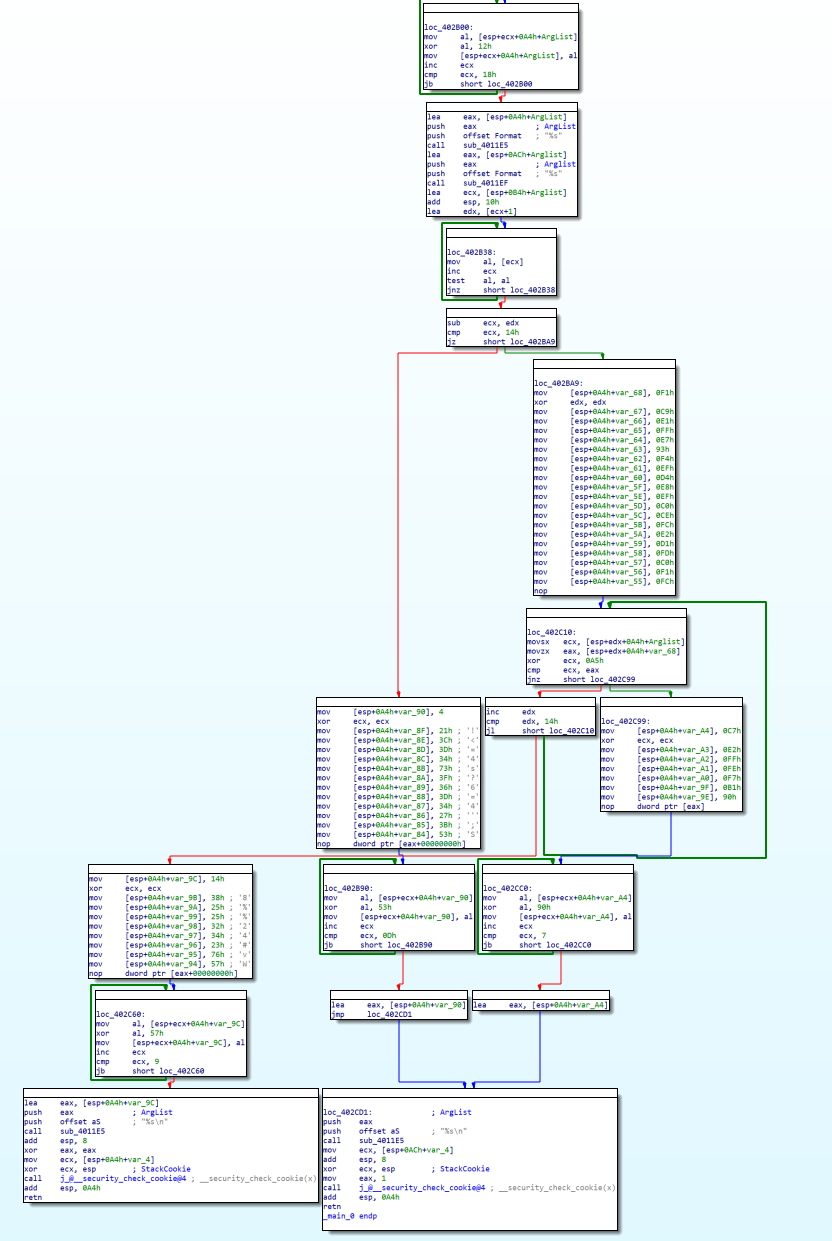
1. **实验步骤**

**（一）task3**

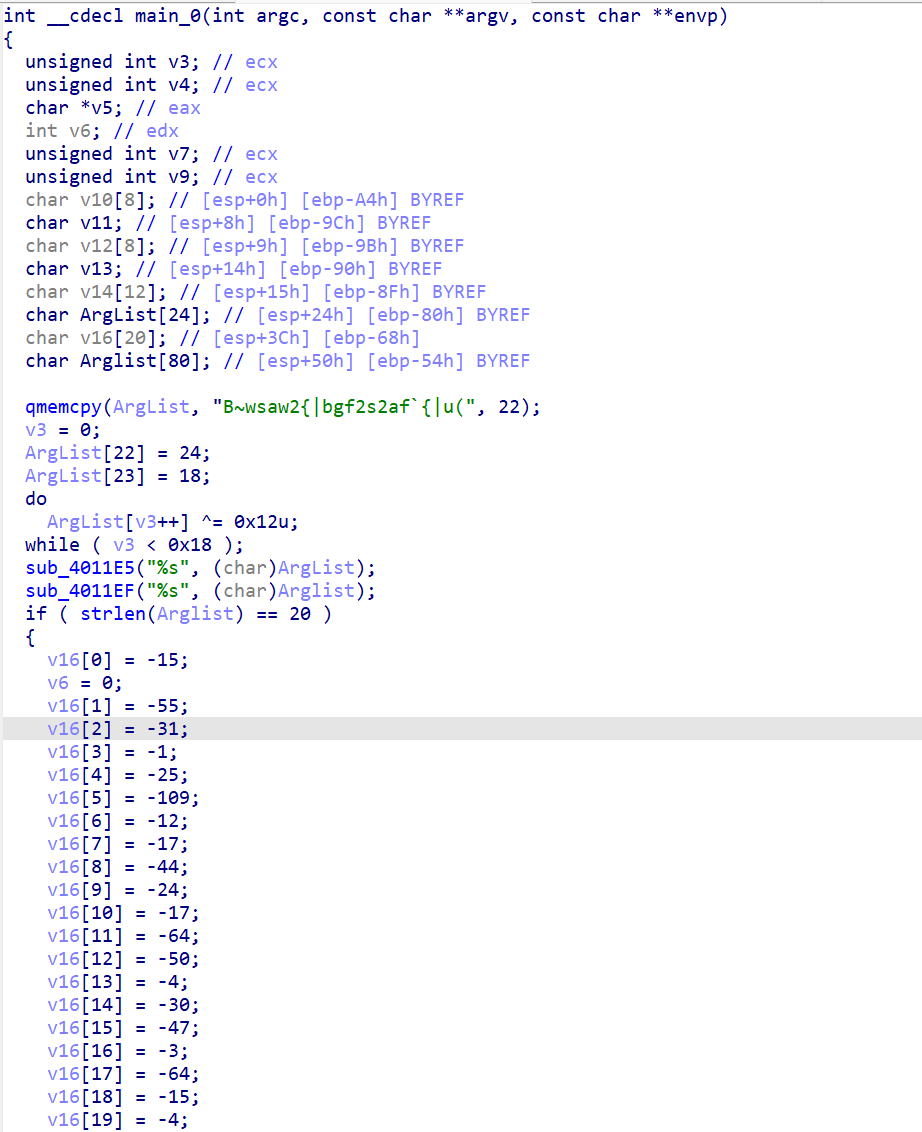
1. 使用IDA Freeware，获得可执行文件的反汇编代码及反编译伪代码，提供截图。

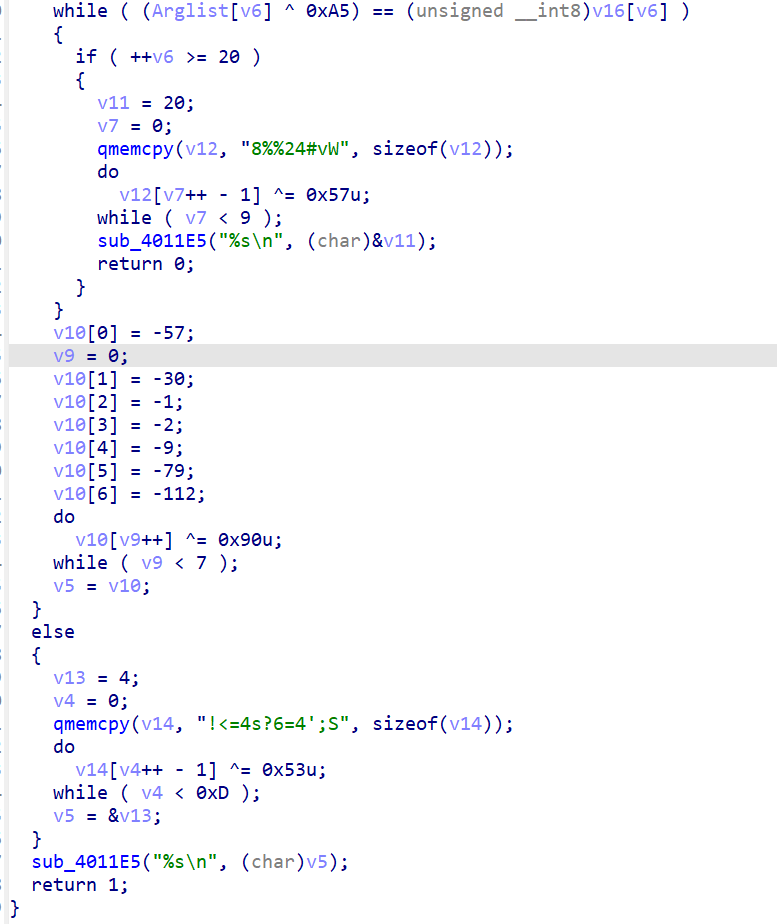
（1）反汇编代码截图





（二）反编译伪代码截图





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，列出正确输入字符串的计算公式。

当我们分析这段代码时，我们可以按照主要功能和流程来理解它。

1. 初始化:

ArgList被初始化为经过异或操作的字符串 "please input a string",用来提示用户输入字符串。

2. 用户输入和长度检查：

用户输入的字符串被存储在Arglist中，检查Arglist的长度是否等于20。

3. 条件判断和循环：

如果输入的字符串长度为20，进入条件判断块，初始化数组v16的每个元素为一组特定的值。

通过循环，将输入字符串的每个字符与0xA5进行异或运算，然后与v16中对应位置的值进行比较。

如果所有字符都匹配，输出 "Correct!"，如果有字符不匹配，输出 "Wrong!"。

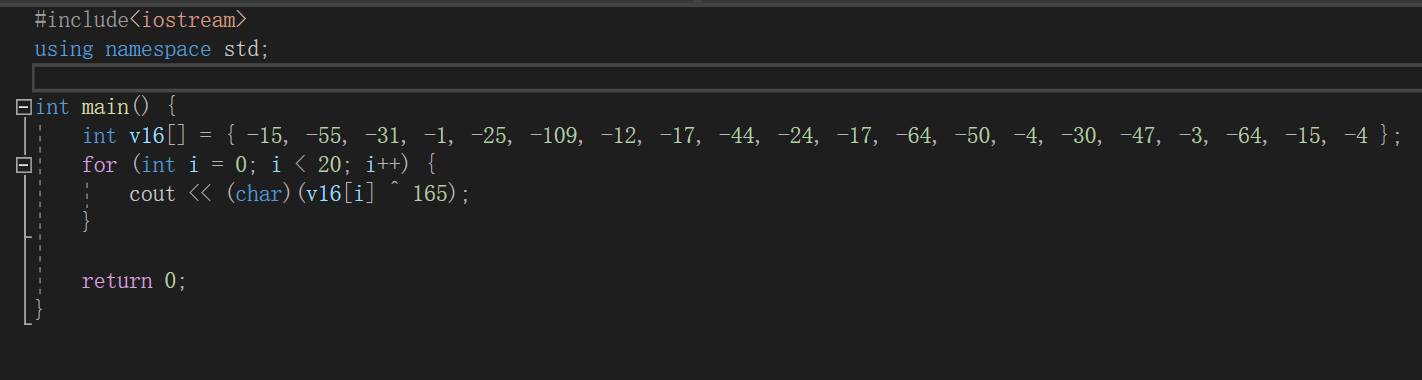
通过以上分析，我们可以得知这段代码的主要目的是根据用户输入的字符串进行一系列处理，并输出相应的消息，包括 "Correct!"、"Wrong!"、"correct!" 或 "wrong length"。

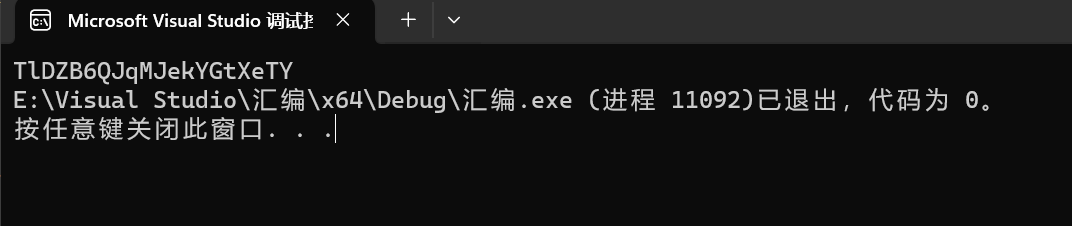
所以，要得到正确的字符串，只需要将v16数组中的每个数分别与165异或一下就行，具体的计算公式为：

Arglist[i]=v16[i]^165;

全部计算出来后，得到正确字符串：TlDZB6QJqMJekYGtXeTY。

3. 解方程组，得到字符串

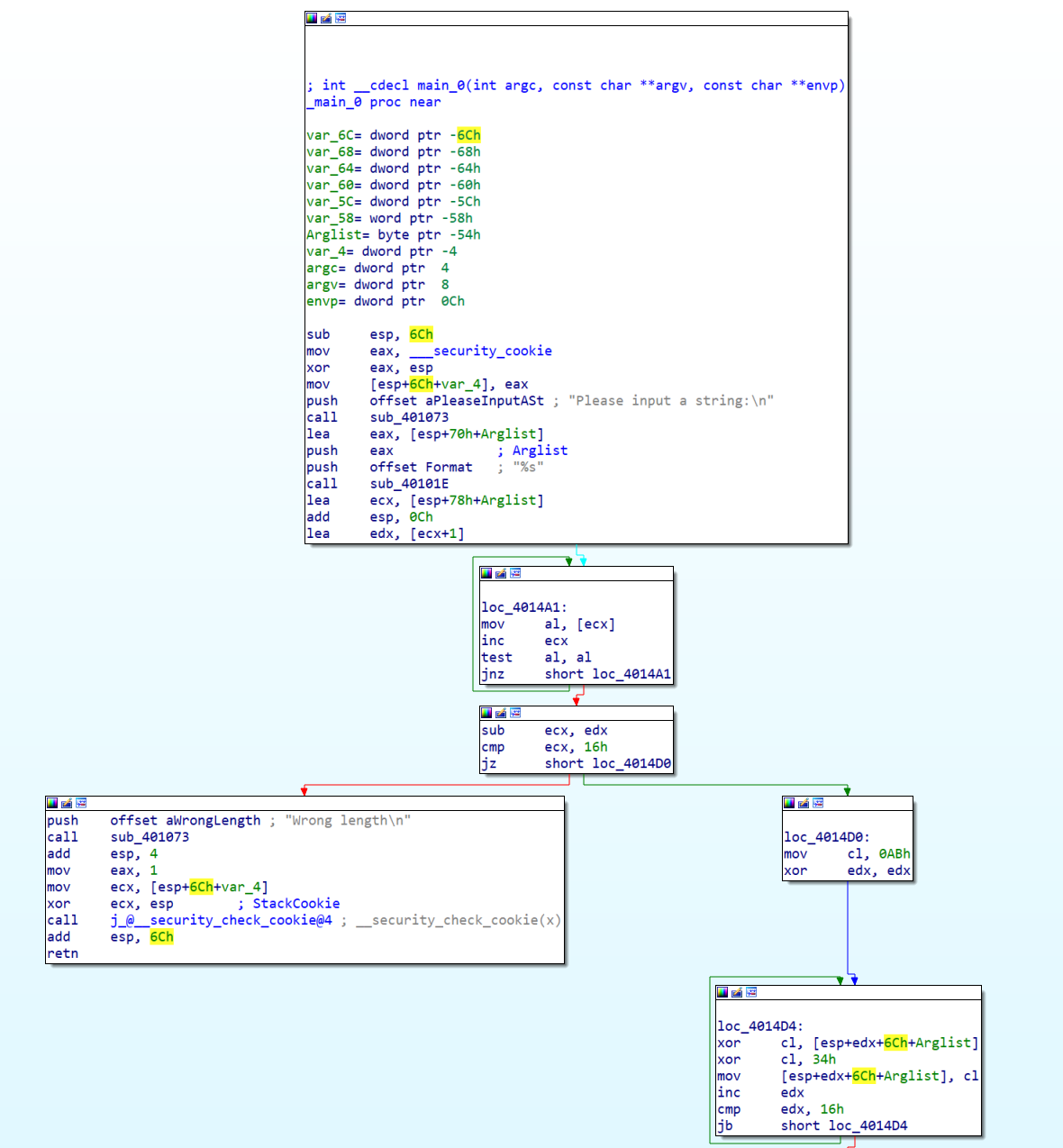


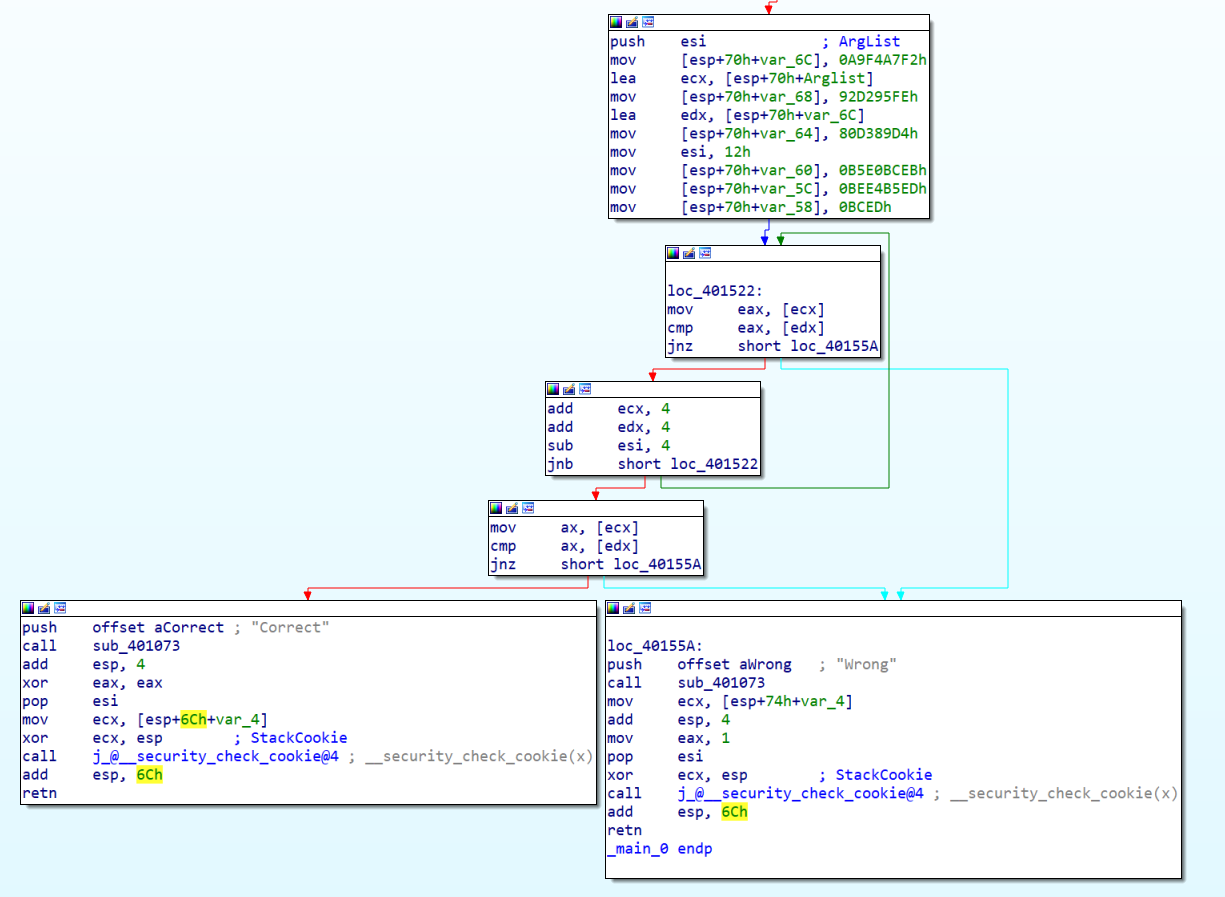


**（二）task4**

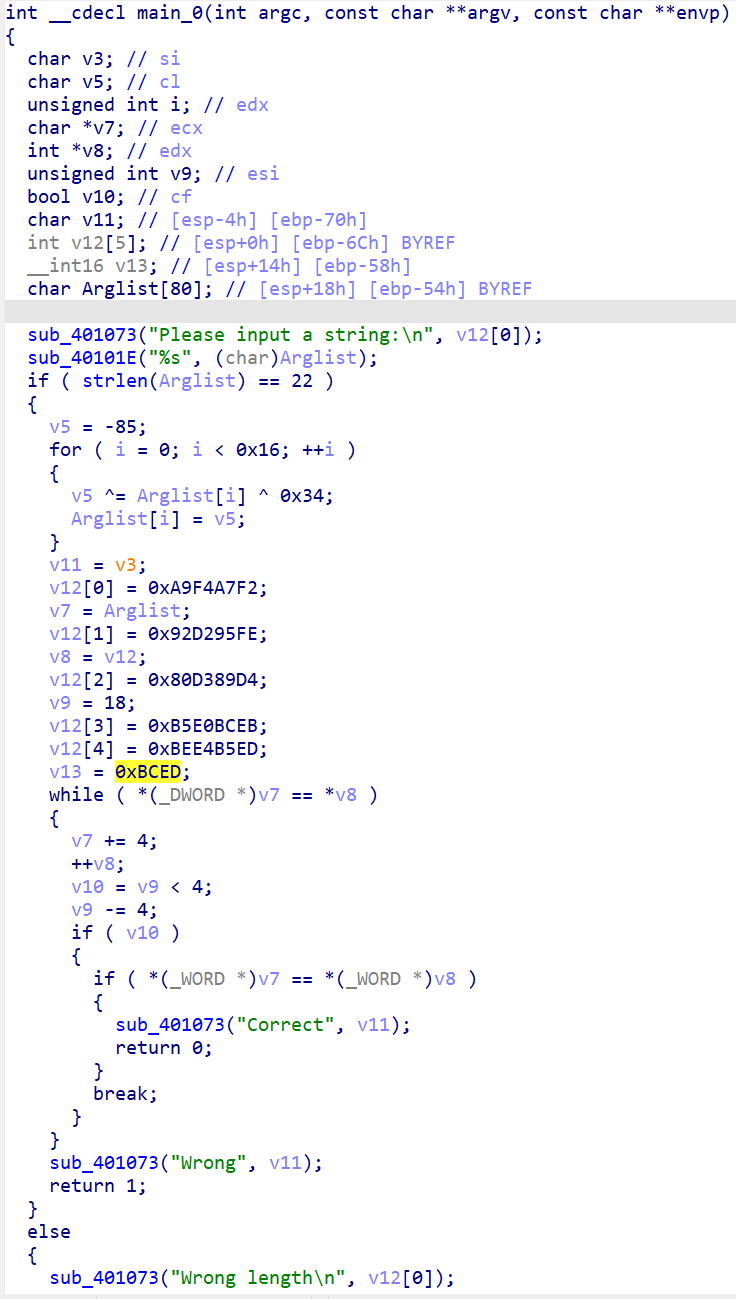
1. 使用IDA Freeware，获得可执行文件的反汇编代码及反编译伪代码，提供截图。

（1）反汇编代码截图





（2）反编译伪代码截图





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，列出正确输入字符串的计算公式。

以下是对代码的分析：

1. 用户输入：

使用sub\_401073函数输出提示信息，要求用户输入一个字符串。使用sub\_40101E函数读取用户输入的字符串到Arglist中。

2. 长度检查：

检查Arglist的长度是否等于22。

3. 字符串处理：

如果长度为22，通过循环对Arglist中的每个字符进行处理：

当 i = 0 时，Arglist[i] = Arglist[i] ^ 52 ^ (-85)。

当 i > 0 时，Arglist[i] = Arglist[i] ^ 52 ^ Arglist[i-1]。

4. 比较和输出：

将一组预定义的整数值存储在数组v12中，并将一个短整数值存储在v13中。通过循环，将Arglist与数组v12中的整数和v16中的短整数逐个比较。如果全部相等，输出 "Correct"；否则，输出 "Wrong"。

5. 错误处理：

如果字符串长度不等于22，输出 "Wrong length"。

要求出正确的字符串，首先需要进行逆运算，还原出原始的字符串。在你提供的逆运算过程中，我们已经知道了处理方式。以下是步骤：

1.拆分16进制数：

将 v12 中的八位16进制数和 v13 中的四位16进制数拆分成两位的16进制数。

2.逆运算：

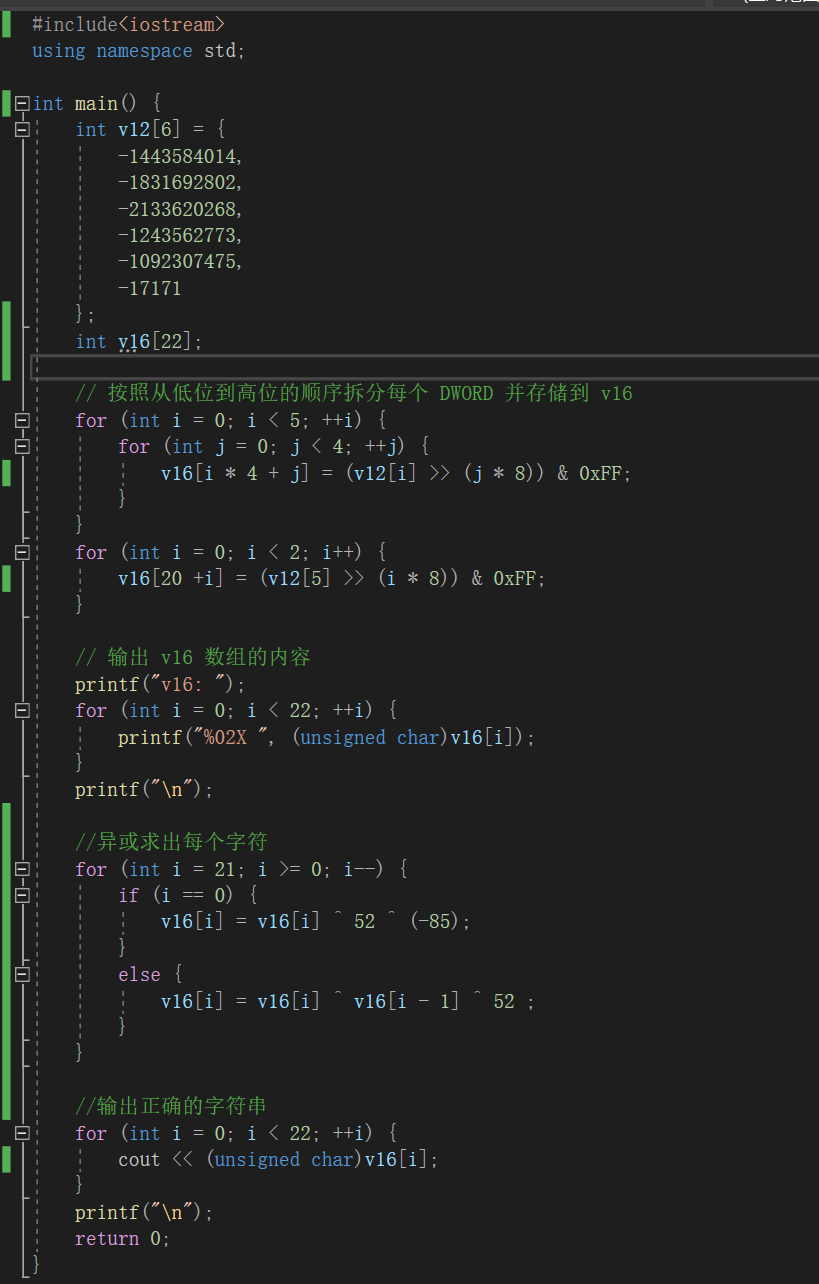
对于每个字符 A[i]，使用逆运算还原原始字符：

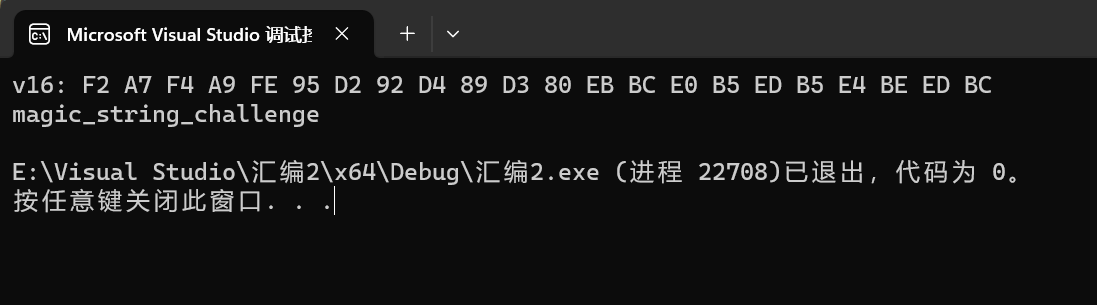
当 i = 0 时，A[i] = A[i] ^ 52 ^ (-85)。

当 i > 0 时，A[i] = A[i] ^ 52 ^ A[i-1]。

逐步进行逆运算，可以得到原始字符串。

3. 解方程，得到数字





1. **实验结果**

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

Task3: TlDZB6QJqMJekYGtXeTY

Task2: magic\_string\_challenge

运行程序，输出成功信息，表明实验成功

