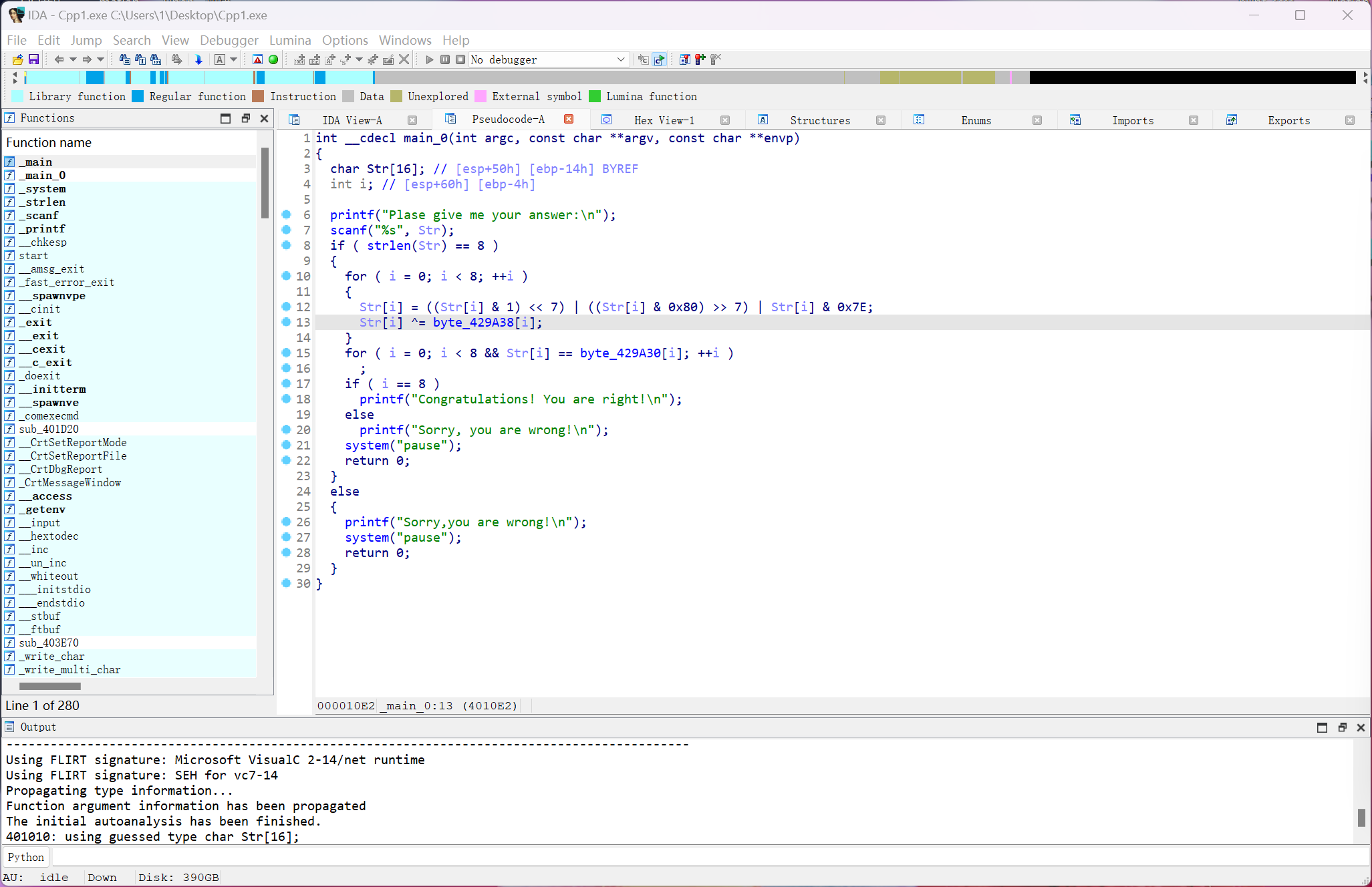
**BIT处理逆向分析**

**2023211595 李昊伦**

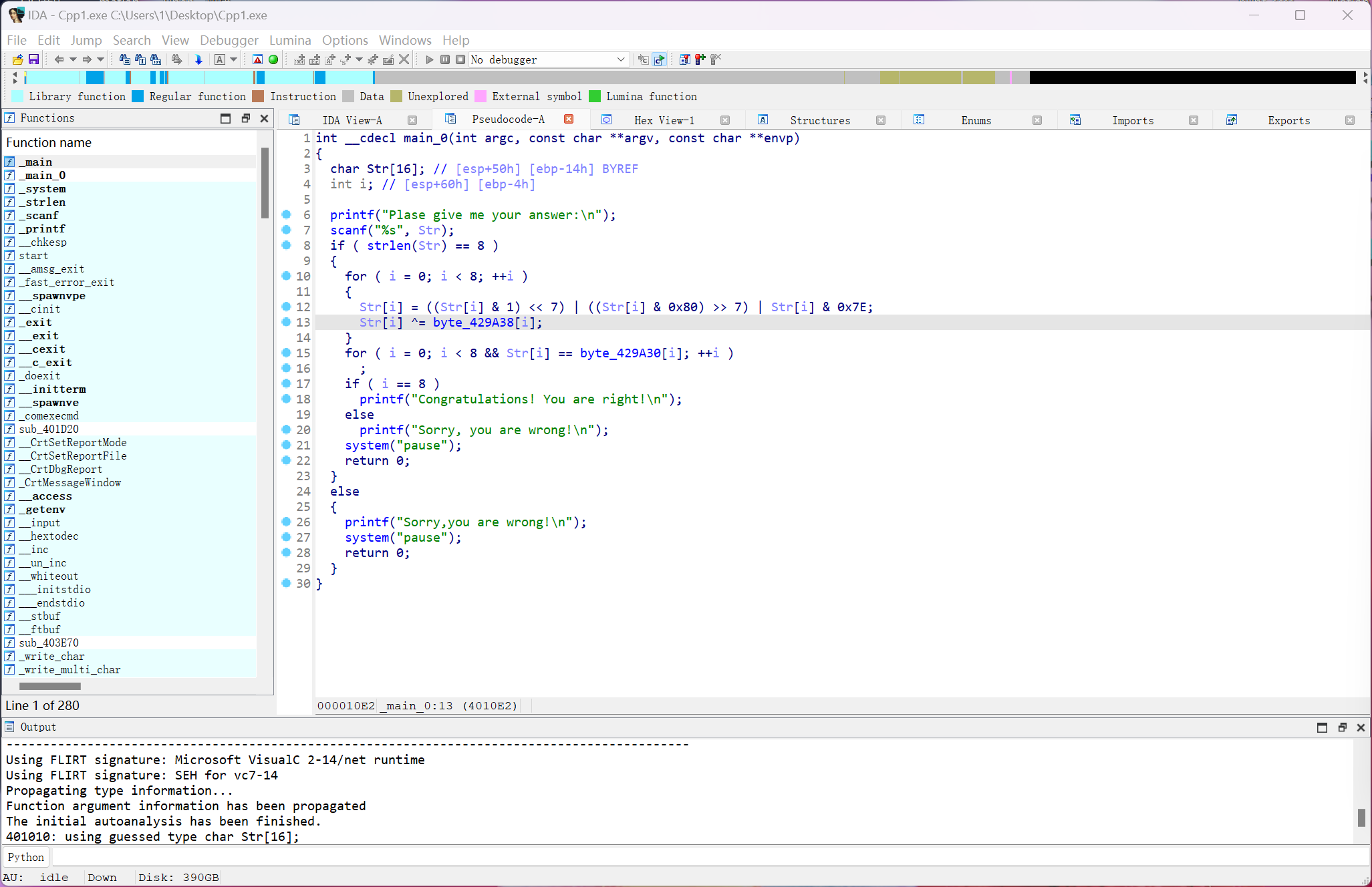
**一、常规逆向分析解出flag**

**1.函数分析**

进入main：



由此得知flag长度为8



当字符串长度为8时，对字符串进行变换处理：

（1）(Str[i] & 1) << 7: 取最低位移到最高位

（2）(Str[i] & 0x80) >> 7: 取最高位移到最低位

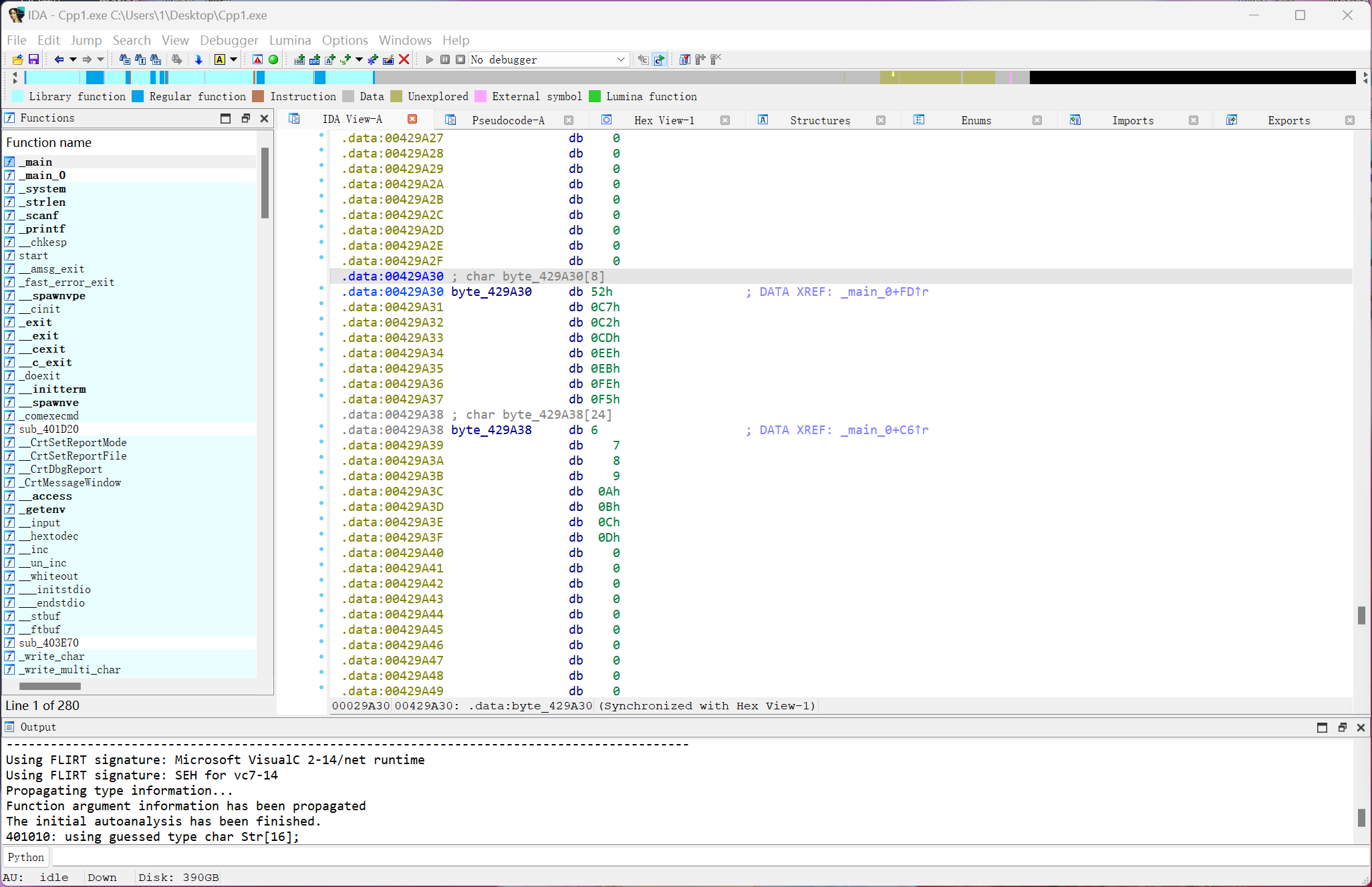
（3）Str[i] & 0x7E: 保留中间的6位。

（4）与byte\_429A38数组进行异或操作

将处理后的字符串与预设值比较，根据比较结果输出成功或失败信息。

2.解flag

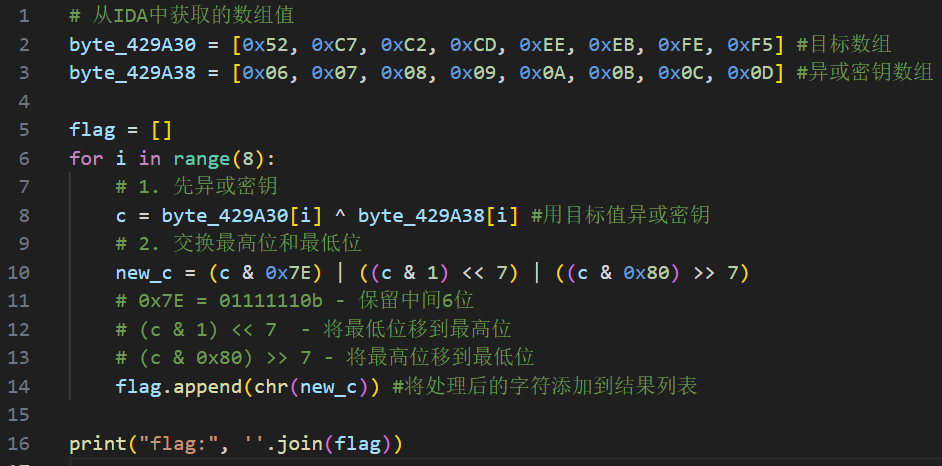
我们需要从IDA中获取byte\_429A30和byte\_429A38的实际值，即在IDA中定位byte\_429A30和byte\_429A38的内存地址。



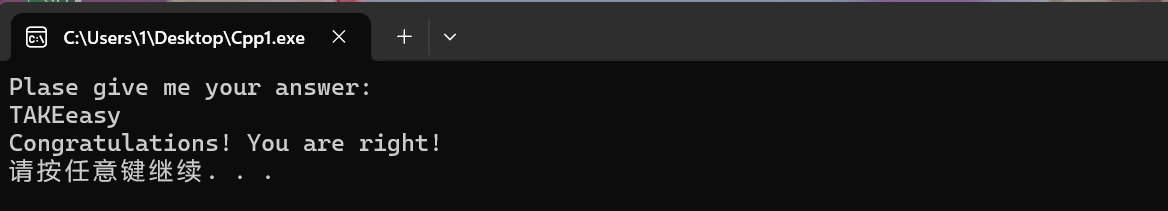
可得到两组值，即：byte\_429A30 = [0x52, 0xC7, 0xC2, 0xCD, 0xEE, 0xEB, 0xFE, 0xF5]

byte\_429A38 = [0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D]

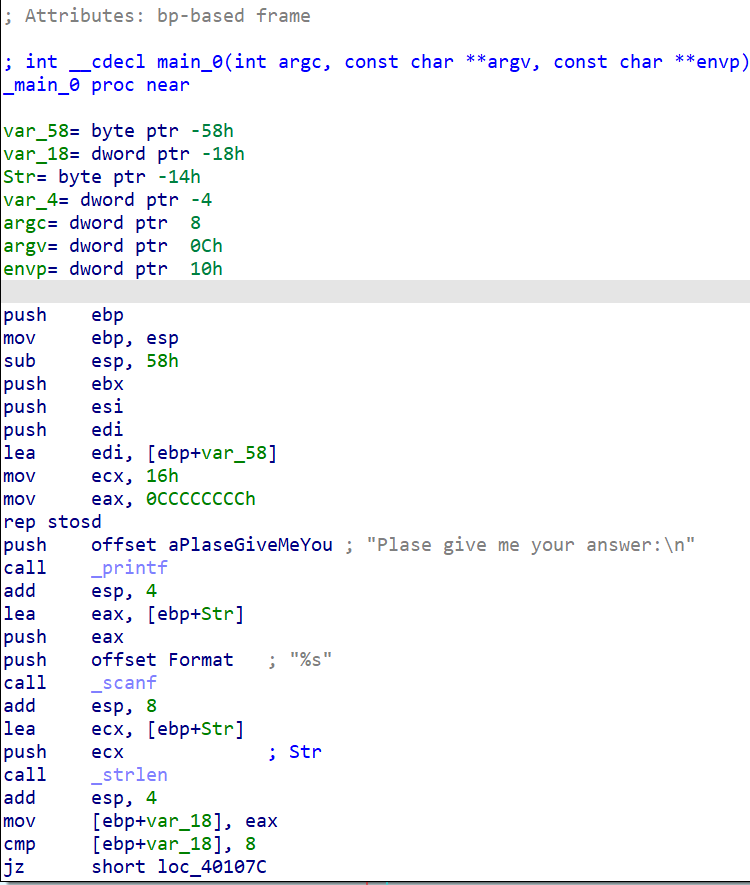
下面进行编程，逆向解出flag



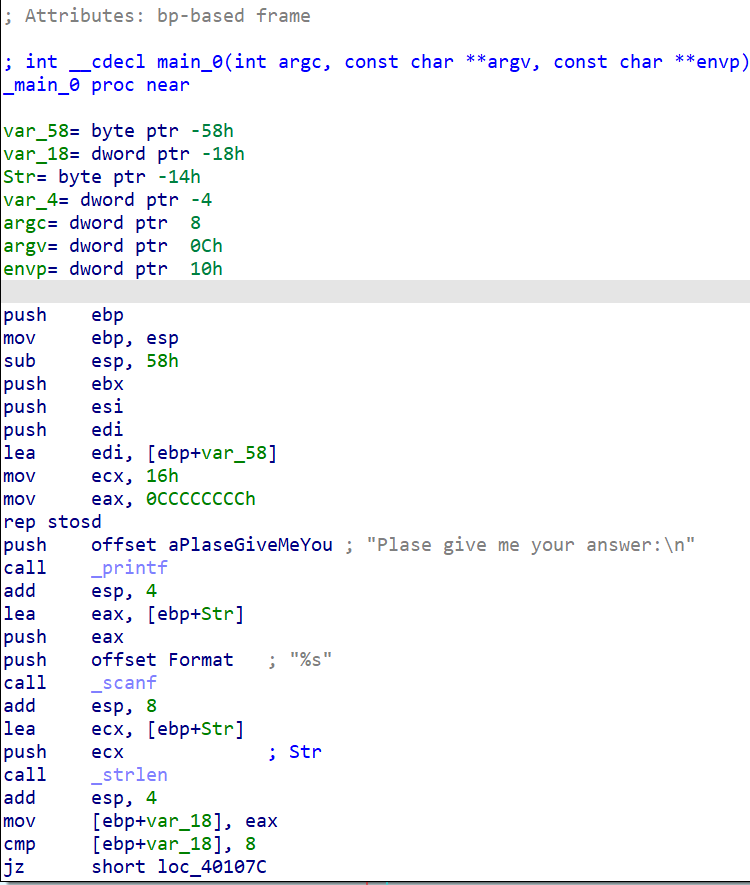
验证可得flag正确



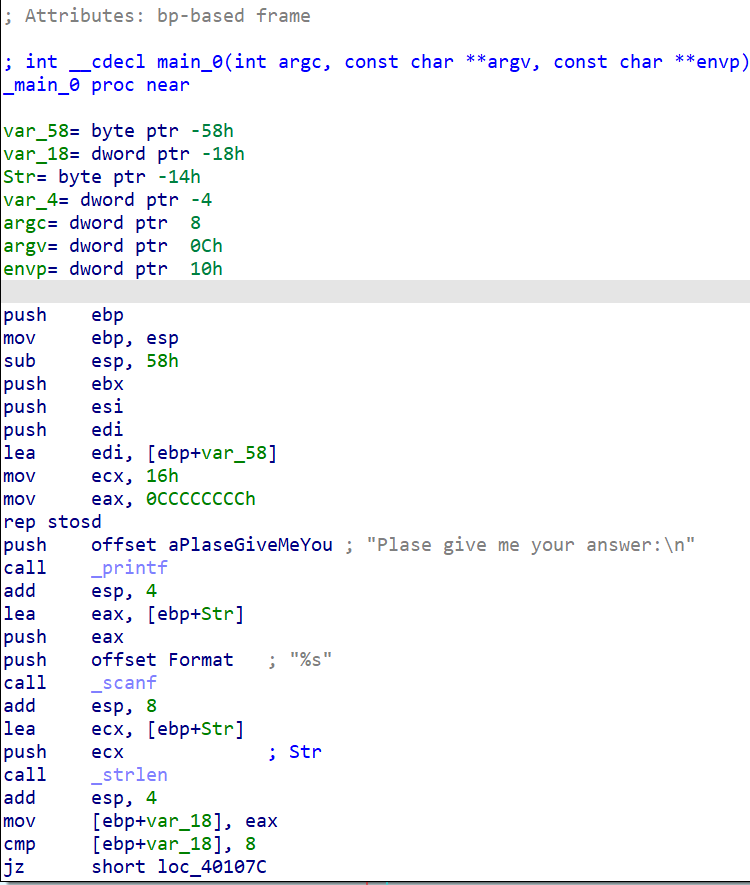
**二、汇编分析**

打印提示信息，即系统告诉用户输入答案。

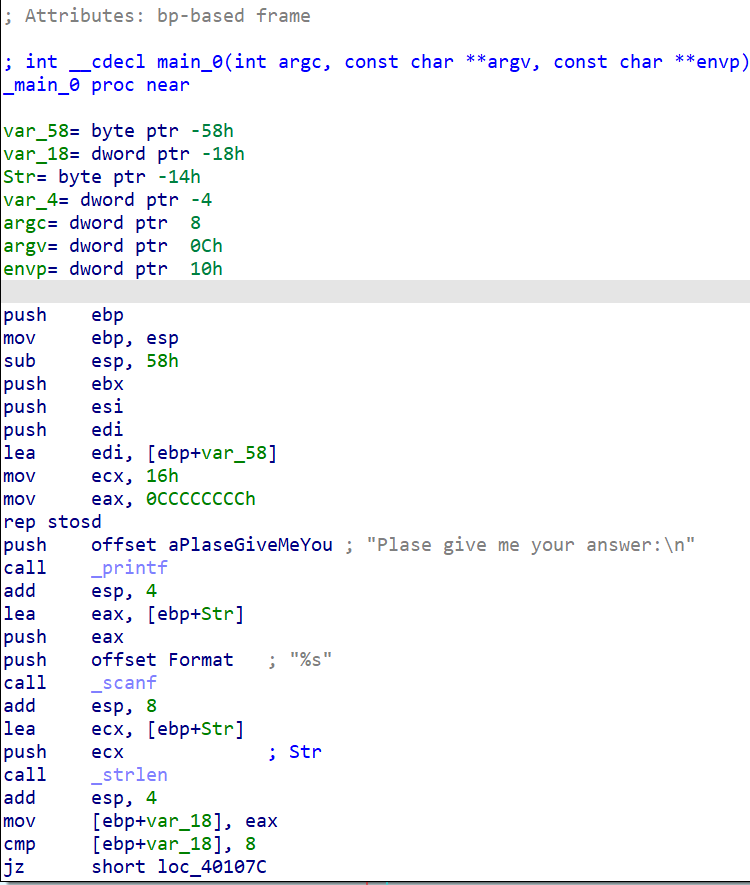
读取用户输入信息。第一行先获取Str缓冲区的地址，第二行作为参数压栈，第三行为格式字符串，第四行调用scanf读取字符串，第五行为清理栈



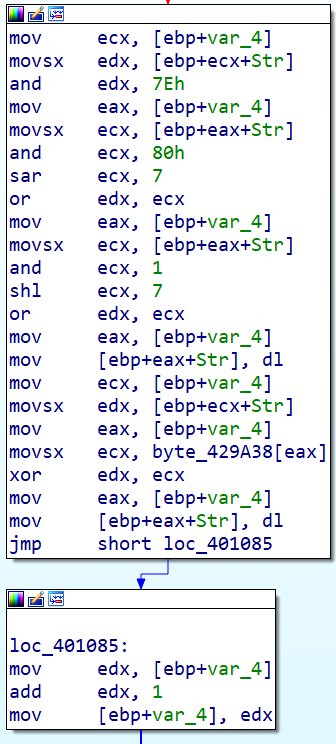
接下来计算输入字符串长度。获取Str地址，调用strlen，清理栈，并保存长度到var\_18



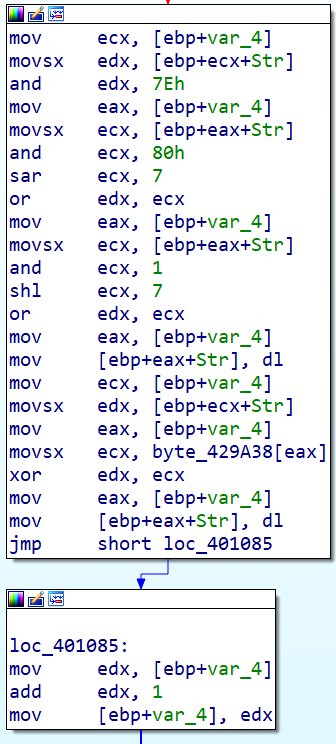
此时检查字符串长度，比较长度是否为8，若等于8则跳转



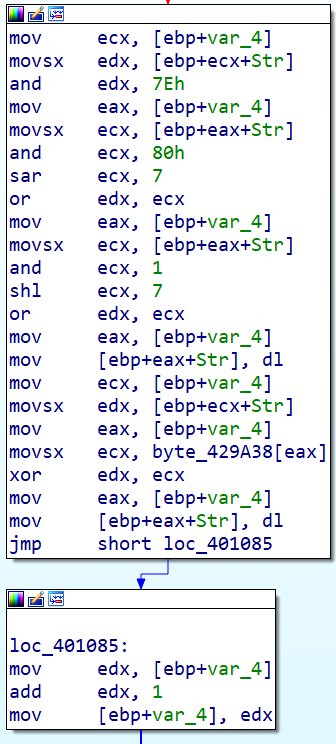
获取当前字符并进行位运算。取出input[i]，并屏蔽掉最高位（bit7），保留低7位。



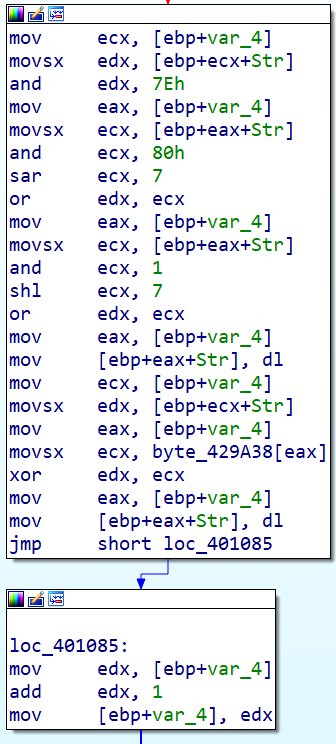
提取 input[i]的bit7，并右移7位，使其变成0或1。



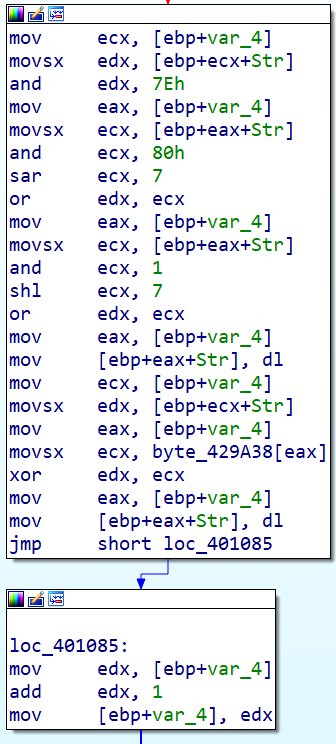
把bit7的值放到bit0的位置。



提取bit0并左移到bit7

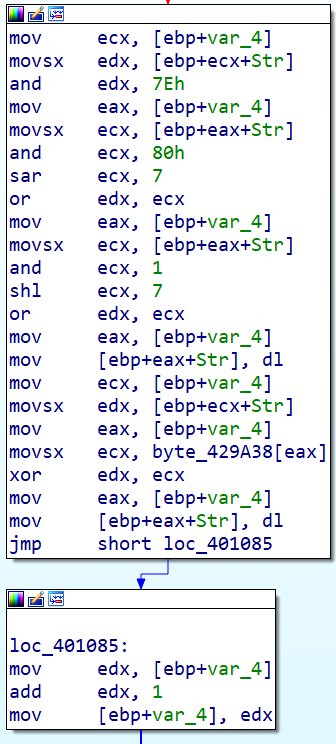


把经过位运算后的字符写回原字符串。

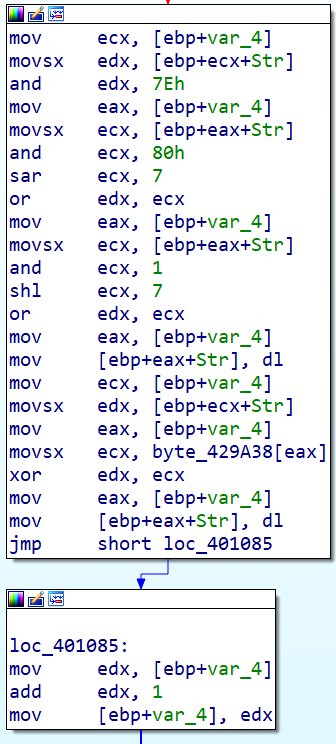


byte\_429A38是一个全局字节数组，可能是密钥表。

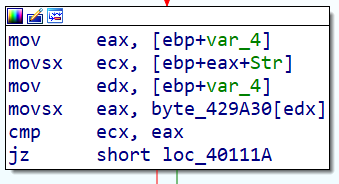
把input[i]和byte\_429A38[i]进行XOR运算，得到加密后的字符。



继续循环处理下一个字符。



这段为循环体内部的字符校验部分，它检查经过加密处理后的输入字符串是否与某个预设的密钥byte\_429A30匹配。



点击列表里的byte\_429A38与byte\_429A30可得到相应数组内存地址，进而解出。