《漏洞利用及渗透测试基础》实验报告

姓名：申宗尚 学号：2213924 班级：信息安全

**实验名称：**

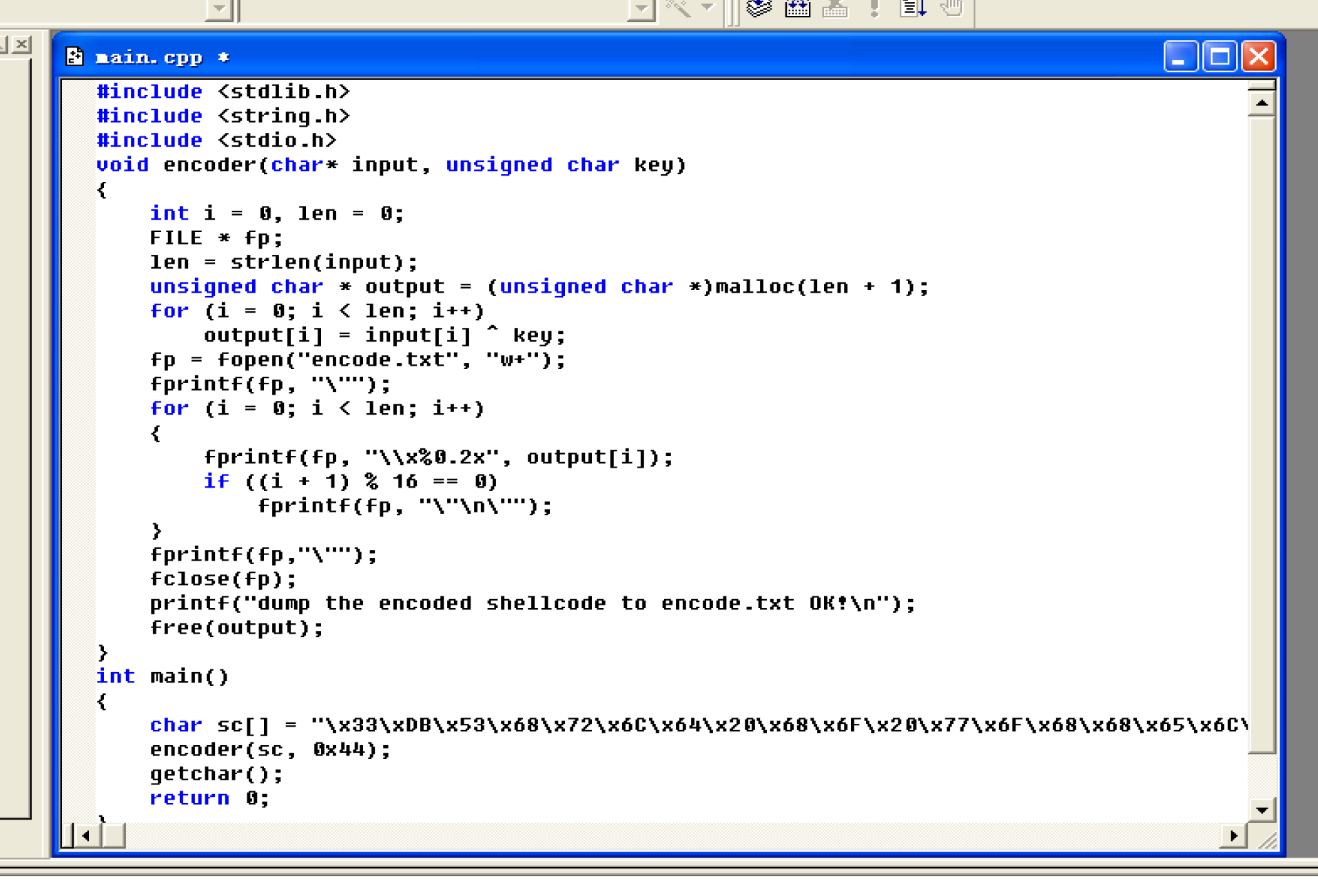
Shellcode编写及编码

**实验要求：**

复现第五章实验三，并将产生的编码后的shellcode在示例5-1中进行验证，阐述shellcode编码的原理、shellcode提取的思想。

**实验过程：**

1. 进入VC6，编写编码程序encode.cpp。



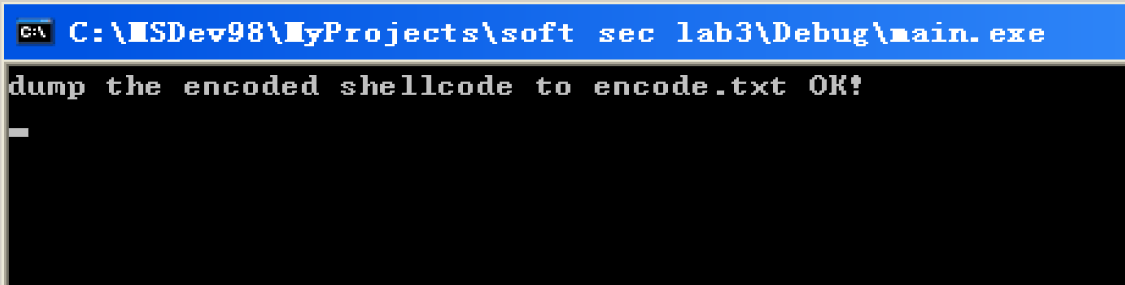
代码分析：

·首先，在主函数中构建字符数组sc[]，储存之前转换好的以16进制数机器码编写的shellcode代码(调用messagebox,标题和正文均显示“hello world”)。

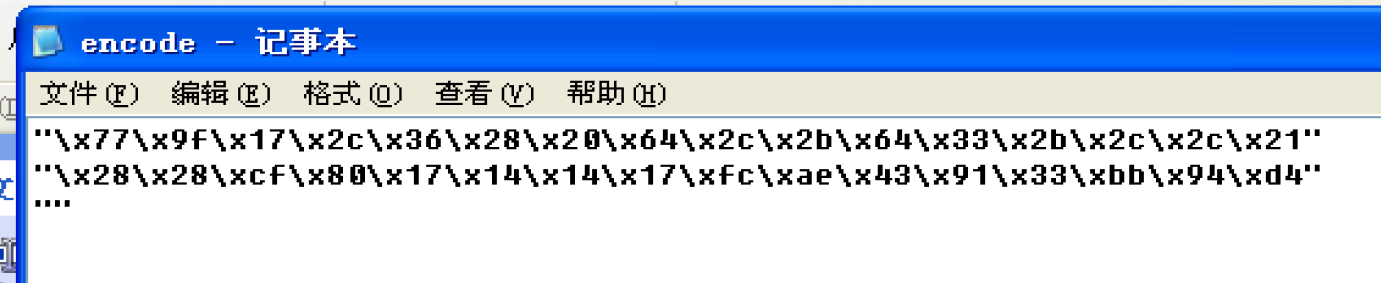
·然后编写encoder函数，使用文件读写，传入需要编码的shellcode和用于编码的key值，编码后将其输出存入encode.txt。

·最后编写encoder的核心部分，本次编码采用异或编码(将shellcode每位数与设置好的key进行异或，然后储存，每16个为一行，在每行开头和结尾均用双引号扩起)

2. 运行程序，获取成功提示信息：

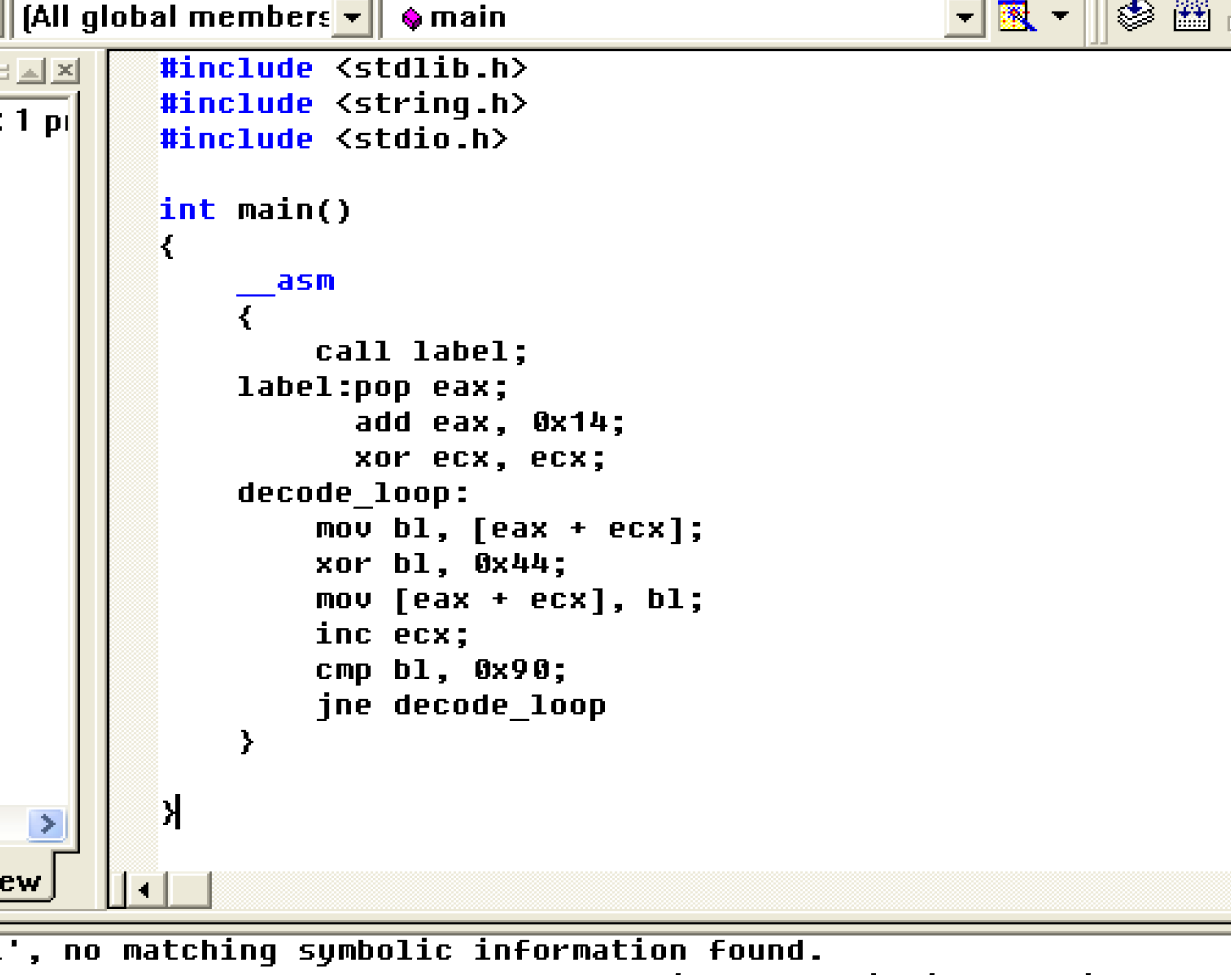


打开encode.txt，观察编码后的结果如下：

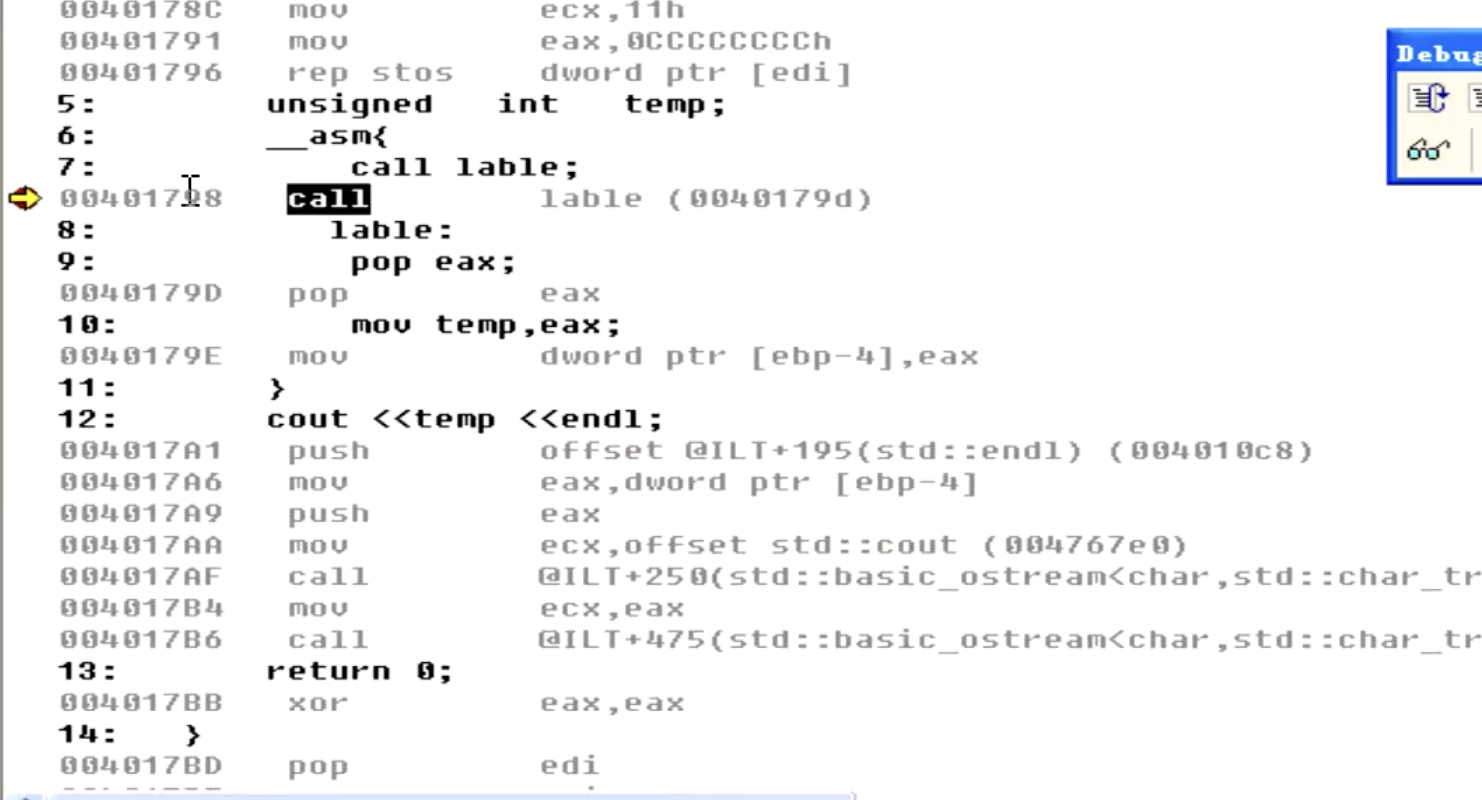


·以第一个\x33为例，0x33与key0x44进行异或，其值为0x77，因此储存\x77，以此类推……每隔16个进行换行。

3.随后，进行解码部分代码decode.cpp的编写如下：



代码分析：  
 ·首先通过call label，对下一行label标签进行调用，使call指令的下一条指令(pop eax)的地址压入函数栈，然后EIP变成下一条指令的地址，此条指令为pop eax,从而将函数顶部的值弹入eax中，即完成将EIP的值存入eax中。



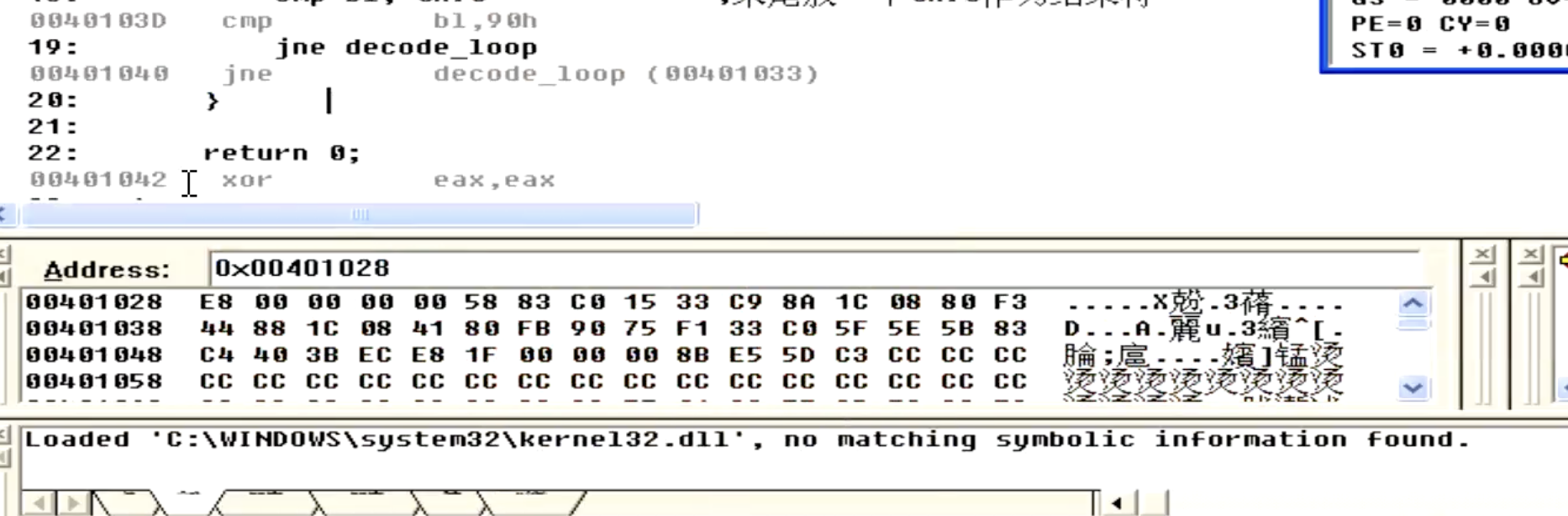
如图，在0x00401798处使用call指令，下一条pop指令地址为0x0040179D，此时从而将0x0040179D压入栈顶，随后执行下一条指令，pop eax，将栈顶值（即0x0040179D）弹入eax，存下当前的EIP。

·然后，将eax加上0x14，这里是由于decode.cpp的代码从当前位置到结尾大小刚好为0x14，从而如果将decode部分代码放在编码后的shellcode前面，此时EIP的值+0x14，就会刚好是shellcode开始部分的地址，便于后续进行解码。

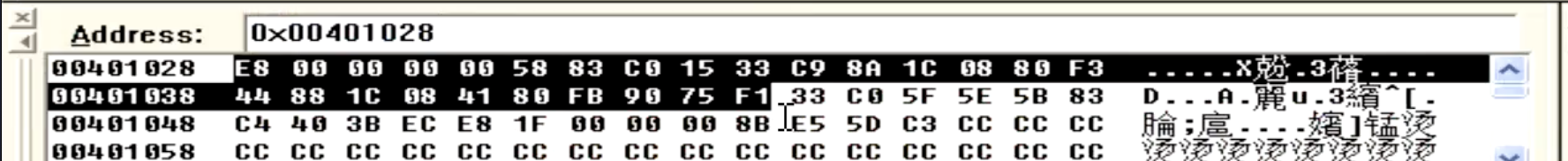
·随后进入解码循环，由于a^key^key=a，只要将每一位再与key进行异或，即可恢复到原来的值，从而完成解码

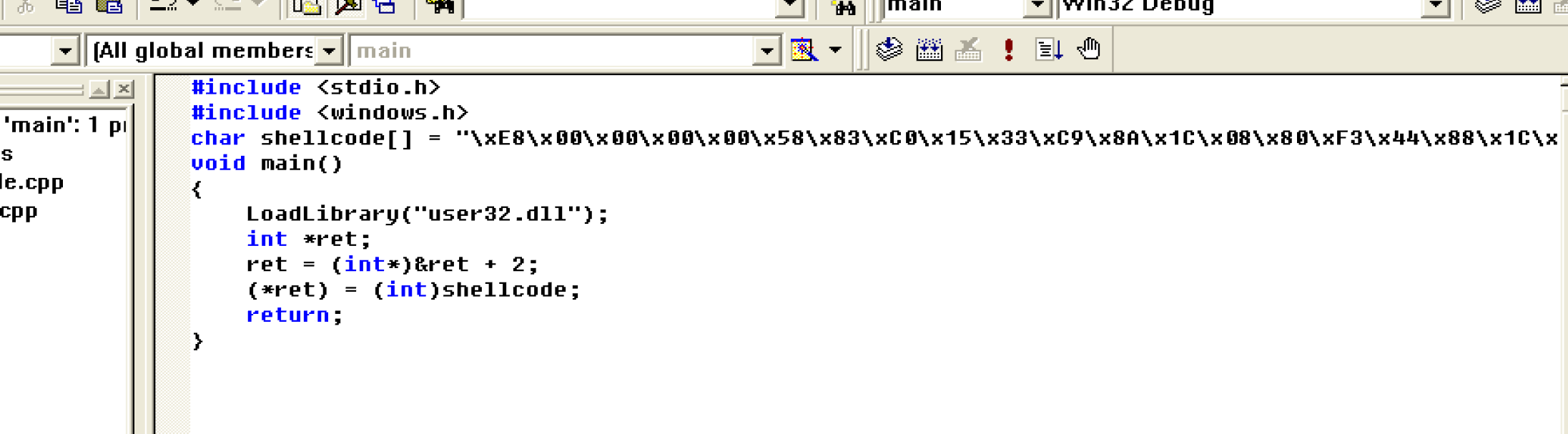
·由于shellcode的最后一位为0x90，通过cmp进行判断，当读到0x90时，终止解码循环，完成解码。

4. 接着，需要获取decode代码部分的机器码，我们通过VC6的汇编模式，可以看到代码开始的地址和结束（return 0）部分的地址，然后通过下面的地址值表，可以查看对应的机器码，如图，代码从0x00401028开始，到0x00401042结束



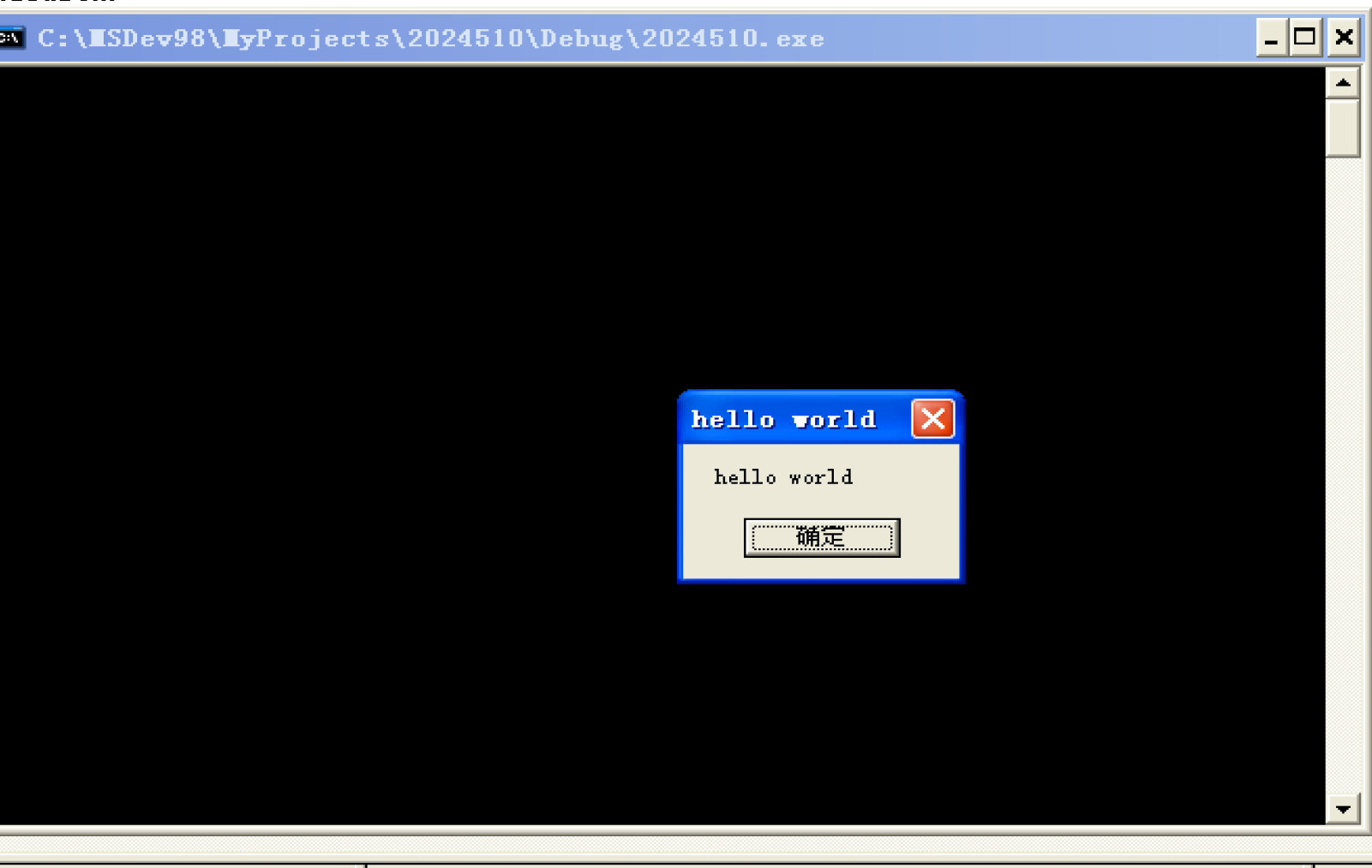
获取机器码如下：



5.然后将decode的机器码在前，encode.txt中的编码后shellcode机器码在后，二者连接放入示例5-1代码中的shellcode[]数组中，如下:  


·这段代码通过将ret设置为(\*ret) = (int)shellcode这行指令的下一行，使得机器将shellcode中的指令全部运行，完成exploit。

6. 执行这段代码如下：



弹出hello world的messagebox，实验成功。

**Shellcode编码的原理：**

Shellcode编码是渗透测试和漏洞利用中常用的技术之一，旨在绕过防御机制和检测规则。其基本原理是对原始的Shellcode进行转换或混淆，使其在传输和执行过程中难以被识别和阻止。

在本实验中，采用了异或编码作为示例。异或编码的原理是将原始Shellcode的每个字节与一个预先定义的密钥进行异或运算。这样可以产生新的字节序列，使得原始Shellcode的结构和内容被混淆。解码时，再次使用相同的密钥对编码后的字节进行异或运算，即可还原成原始的Shellcode。

异或编码的优点在于简单易实现，且不需要额外的存储空间。然而，由于异或运算是可逆的，因此其安全性相对较低，容易被高级的检测技术所绕过。

**Shellcode提取的思想：**

Shellcode提取是指从编码后的Shellcode中还原出原始的二进制数据。这个过程通常需要了解编码时所采用的算法和密钥，并编写相应的解码程序。

在本实验中，提取编码后的Shellcode的思想是通过解码程序逆向编码过程，将编码后的字节序列还原成原始的Shellcode。解码程序根据编码时所使用的密钥，对每个字节进行逆向的异或运算，以还原出原始的字节序列。

提取Shellcode的过程需要仔细分析编码和解码的算法，确保解码过程的正确性和完整性。同时，需要注意处理特殊情况，如编码后的Shellcode中可能包含控制字符或其他特殊字符。

**心得体会：**

通过这次实验，我深入理解了Shellcode编码和提取的原理及实现方式。在编写编码和解码程序的过程中，我学会了如何分析和处理二进制数据，以及如何设计有效的编码方案。实验过程中的调试和验证让我更加熟悉了Shellcode的运行机制，提高了我的安全意识和技术能力。