《漏洞利用及渗透测试基础》实验报告

姓名： 许健 学号： 2013018 班级： 信安班

**实验名称：**

IDE反汇编实验

**实验要求：**

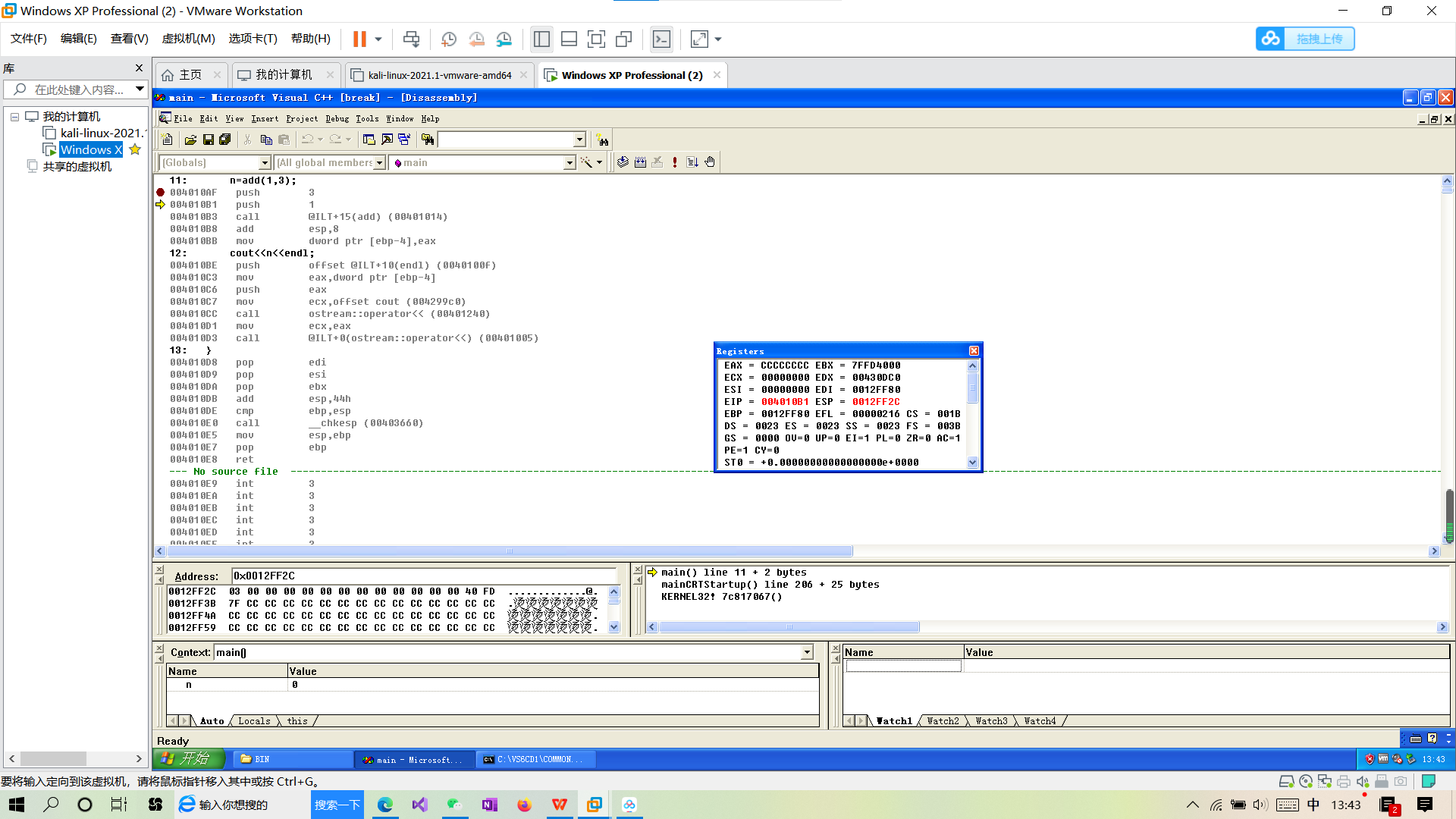
根据第二章示例2-1，在XP环境下进行VC6反汇编调试，熟悉函数调用、栈帧切换、CALL和RET指令等汇编语言实现，将call语句执行过程中的EIP变化、ESP、EBP变化等状态进行记录，解释变化的主要原因。

**实验过程：**

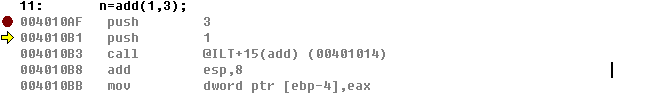
1. **进入VC反汇编**

使用vmware运行xp环境，与主机共享文件夹VC6，创建源文件并进入反汇编界面，

调出汇编代码、registers的值以及内存地址等如图：



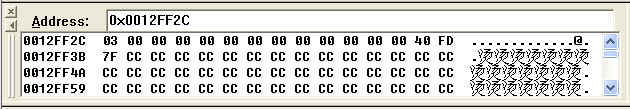
1. **观察add函数调用前后语句**

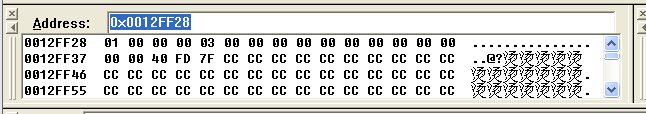


Add函数的调用：

首先将参数按照从右向左的顺序压入栈中，观察此时esp寄存器的变化

我们使用快捷键F11单步执行，观察esp存储Address变化情况：

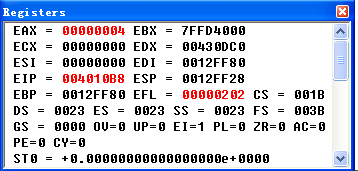




可以看到，esp存储地址值向低地址方向扩展，

并且栈顶的值由3变成1，可以看到参数被压入栈中。

此时我们再使用快捷键F10暂时跳过call指令具体执行



可以看到EIP存储的值始终是下一条执行语句的地址！

**Add esp,8**这条指令则是将栈顶指针向高地址移动8个字节，也就是两个参数占用地址，当函数执行完后，参数被弹出栈。

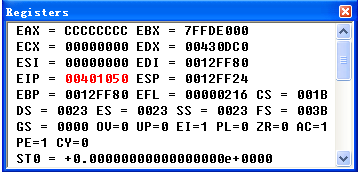
**mov dword ptr [ebp-4],eax**

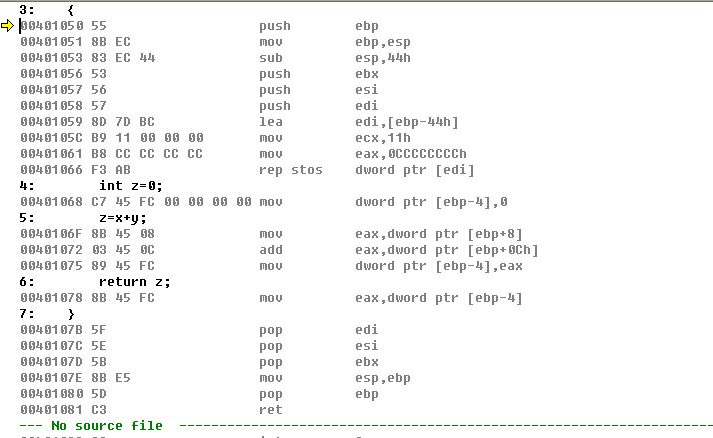
这条指令则是将存储在eax中的函数返回值赋值给局部变量n

1. **add函数内部栈帧切换等关键汇编代码**



F11单步进入call指令，首先执行一个jmp指令，看到EIP的值变化





关键代码：

1.将ebp的地址压入栈，保存原函数栈帧的基地址

2.再将esp的值赋给ebp，ebp现在指向新栈帧的基地址

3.将esp的值-44h，给新栈帧预留空间

4.将ebx、esi、edi寄存器压入栈中

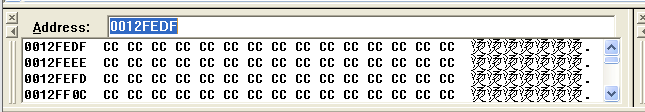
5.将ebp-44h的地址赋给edi,也就是压入寄存器前esp的地址

6.ecx存的是循环次数

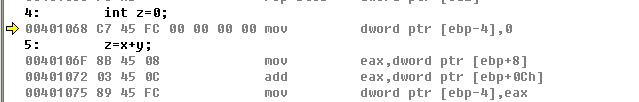
7.eax存的是要初始化的值

8.rep指令重复前缀指令集，将栈帧内部地址都初始为0CCCCCCCCh

注意到每一次循环，ecx的值都会减1，不再演示。



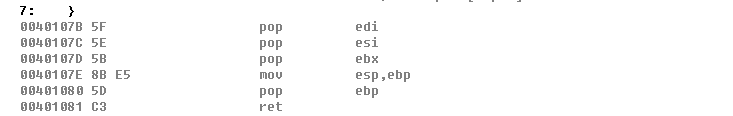
可以看到最终的效果



[ebp-4]也就是n的地址，可以看到z=x+y实际上就是将值赋给n



Return语句将函数返回值传给eax



最后将寄存器弹出栈，恢复esp、ebp的值，ret指令实际上是执行了Pop EIP，

函数调用的过程结束。

**心得体会：**

通过一个简单的C++函数，利用VC6反汇编，看到函数调用前后内部栈帧的变化情况，理解寄存器尤其是指令寄存器EIP的作用，同时熟练了汇编语言的使用，包括Call指令、Ret指令；理解了函数参数、返回地址、局部变量压入栈中顺序，和esp、ebp栈寄存器变化情况。