《漏洞利用及渗透测试基础》实验报告

姓名：管昀玫 学号：2013750 班级：计科一班

**实验名称：**

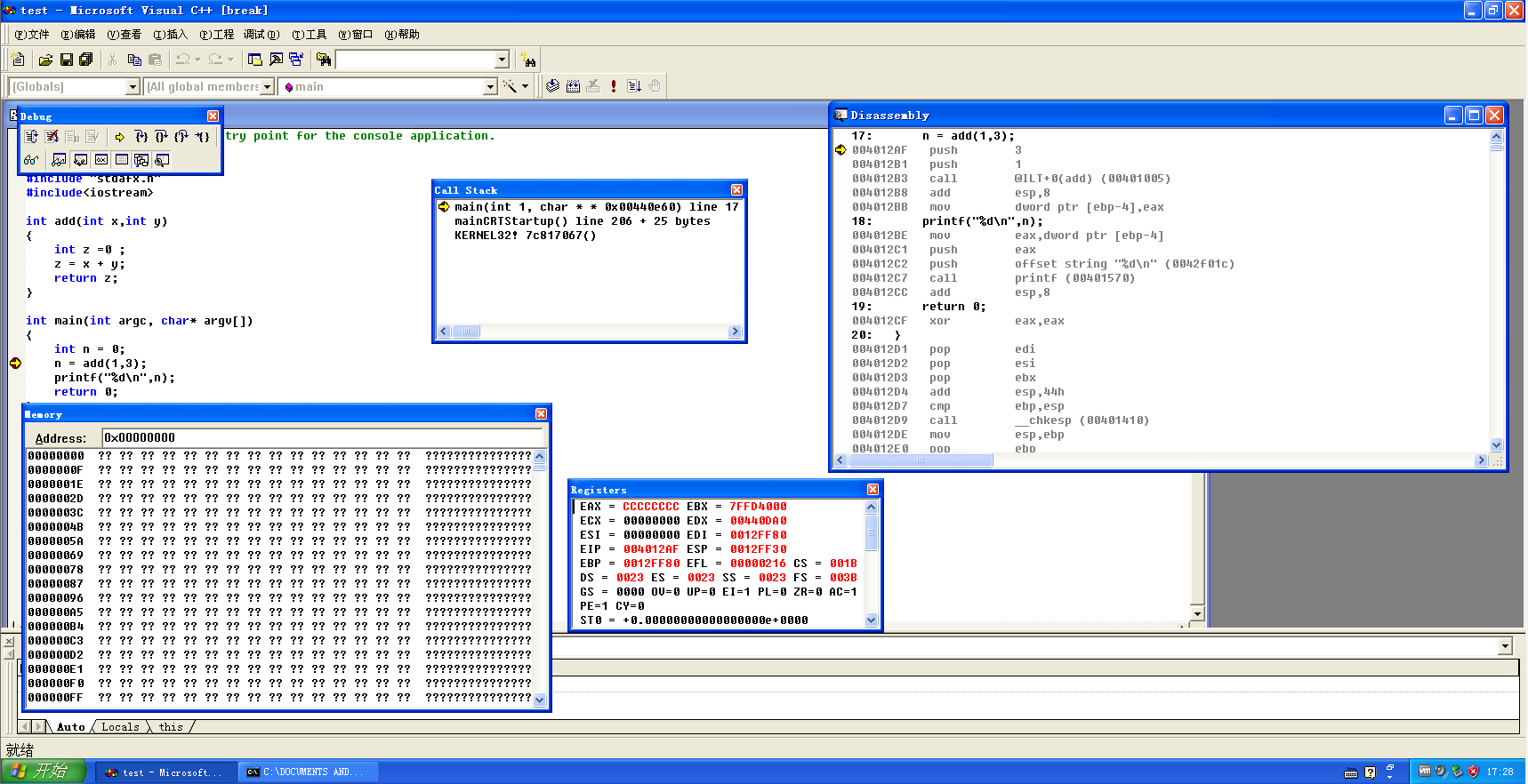
IDE反汇编实验

**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：**

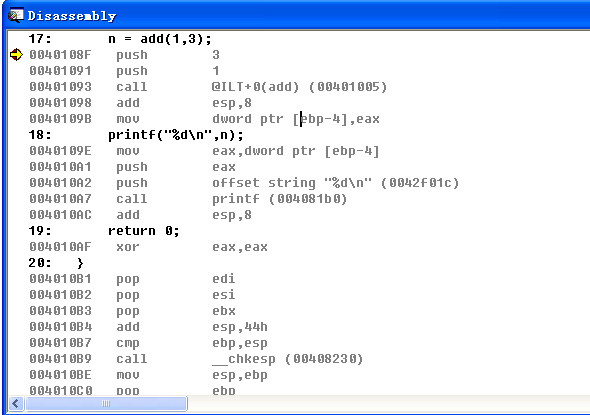
1. 在add函数处打上断点，进入VC反汇编



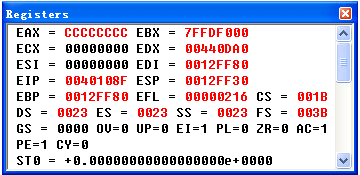
1. 观察add函数调用前反汇编语句、寄存器的状态

反汇编窗口：

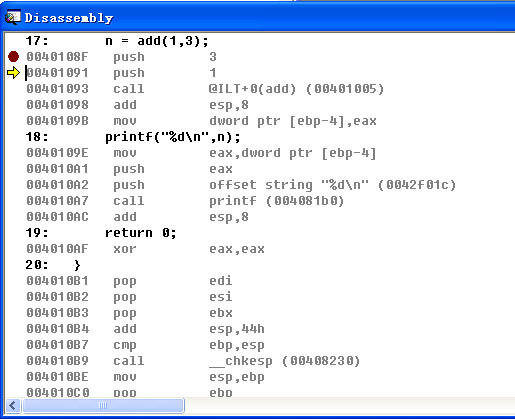
初始化n=0，此时n的地址为[ebp-4],即EBP指针抬高4字后的位置。

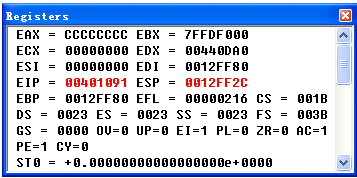


此时正要执行add函数。push语句先后push了3和1，符合参数从右向左依次入栈规则；其后为call指令，此时的寄存器状态为：

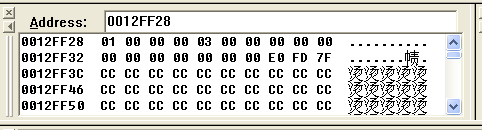


EBP寄存器中存放的地址为：0012FF80；ESP为0012FF30；EIP中存放的是0040108F，对应着push 3这条指令。





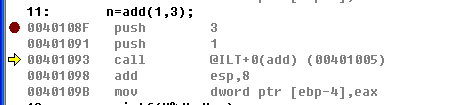
按下F11后可以看到ESP抬高了4字节，变为了0012FF2C；EIP变为push 1指令的地址，之后执行push 1也是同理。



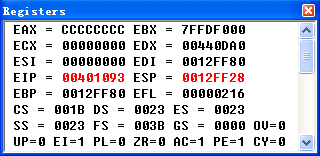
可以看见，在push 1之后，ESP的地址为0012FF28，在Address中填入该地址后可以看见push进去的3和1；

1. add函数内部栈帧切换等关键汇编代码

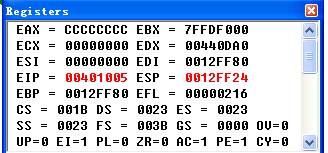
执行call指令之前：



00401093是将要执行的指令地址，而下一行的00401098是之后要返回的地址，即返回地址。

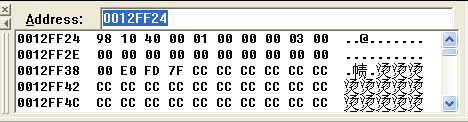


此时的EIP指向call指令：





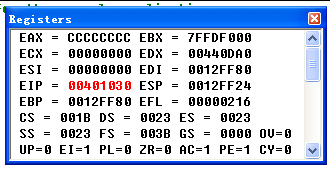
按下F11后，并没有直接跳转到add函数里，但此时栈顶发生了改变。栈顶为0012FF24，内容如图所示。



ESP所指的地址为00401098，正是之后需要返回的地址。这就是函数调用的第一条操作：返回地址入栈。

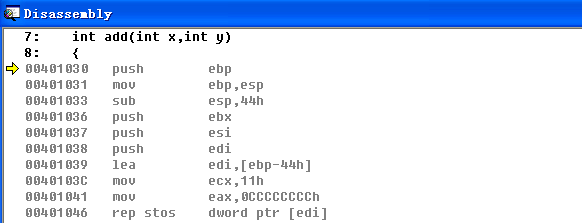
进入到call中，EIP指向了call后给的指令地址(00401005)，该地址上存放jmp指令，用于跳转到add函数地址(00401030)

执行到jmp指令时，EIP中存放add函数地址

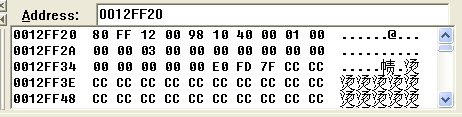


再按下F11后，便跳转进入add函数的地址。

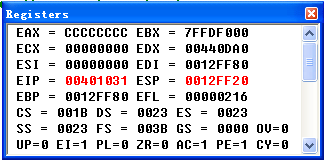
进入到add函数，首先是栈帧的切换：



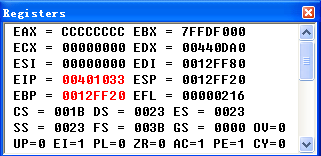
首先是将EBP中存放的地址压入栈中。Push之后，EBP-4，向高地址调整，变成0012FF80， EBP是刚才主函数的地址。



执行mov ebp,esp之前：

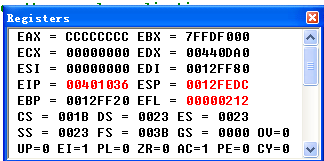


执行mov ebp,esp之后：

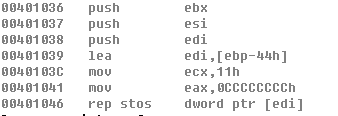


此时EBP上移，和ESP达到同一个位置。Mov指令就是把ESP的值赋给EBP。意味着为add函数设置了栈的地址。

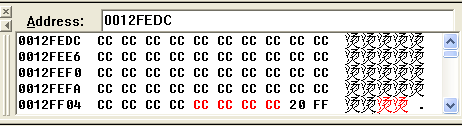
之后执行sub esp, 44h，将ESP上移，开辟空间；



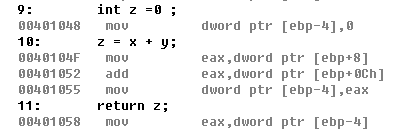
再将需要用到的EBX, ESI, EDI入栈，并完成初始化。每一次push，ESP的值都会抬高。



Mov ecx, 11h，说明执行了11h次的循环，利用0CCCCCCCCh将44h的空间初始化。



执行函数体，并将需要返回的值存在eax中

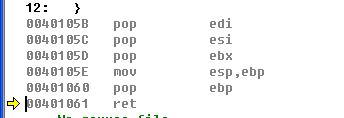


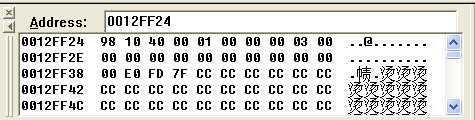
EAX先被赋值为1，而EBP+0Ch为3，因此EAX的最终结果为4，

,可以看到EAX中已经存放了结果4：



函数体执行完毕，系统将栈中已经用过的空间进行释放。首先是将EBP-4这个地址赋值给EAX，然后对应先前3个push，此处有三个pop，释放空间；将EBP赋值给ESP之后，弹出EBP；

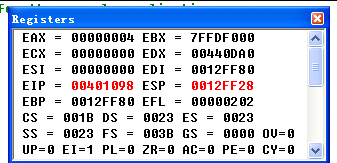




此时EBP的地址中存放了00401098，即主函数的地址。

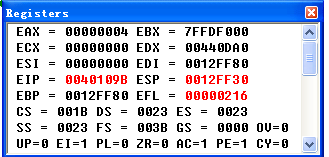
执行ret指令后：





可以看到此时的EIP回到了call指令的后一条指令地址。

执行add esp, 8之后的寄存器状态：



add esp,8的指令便是将之前压入栈中的参数3和1，一共8字节的释放；返回值用EAX来保存，用mov指令将EAX赋值给EBP-4，即把结果4赋值给n，至此栈帧调整完毕（此时的EBP已经回到了call add之前的状态），继续执行main函数中剩下的指令直到结束。

**心得体会：**

通过本次实验，我更加直观清晰地了解了函数调用时栈帧、地址、EIP、ESP、EBP等指针存放状态的变化，熟悉了反汇编、断点、单步执行等操作。我理解了栈帧调整的具体步骤，即：返回地址入栈，参数从右向左入栈，开辟函数空间，之后再相对应地释放函数空间，读取返回地址，最终返回到主函数。这为日后的汇编学习与软件安全的学习打下基础。