《软件安全》实验一实验报告

姓名：李雅帆　　学号：2213041　　班级：信安班

一、实验名称：

熟悉 IDE 反汇编及汇编语言

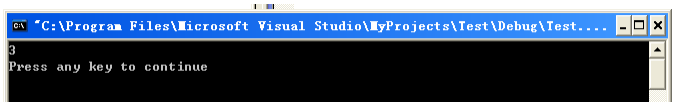
二、实验要求：

请根据第二章示例 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。

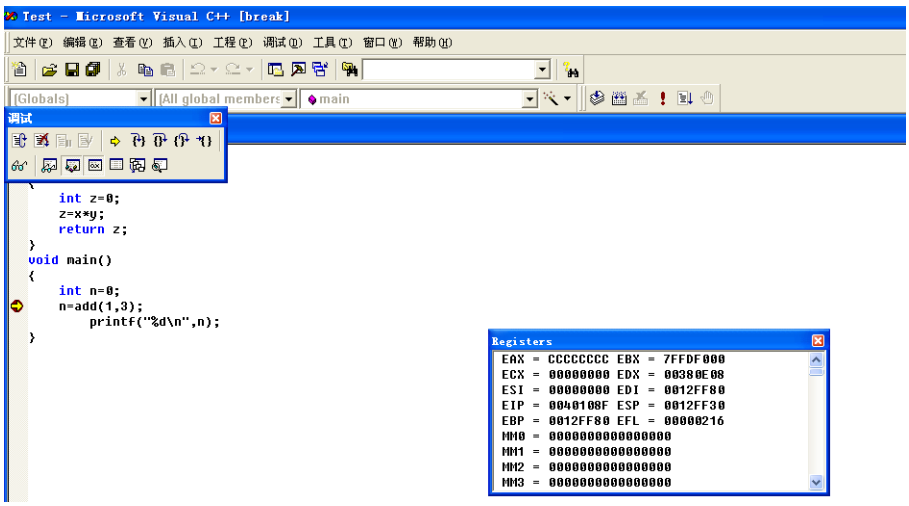
三、实验过程：

1.创建项目

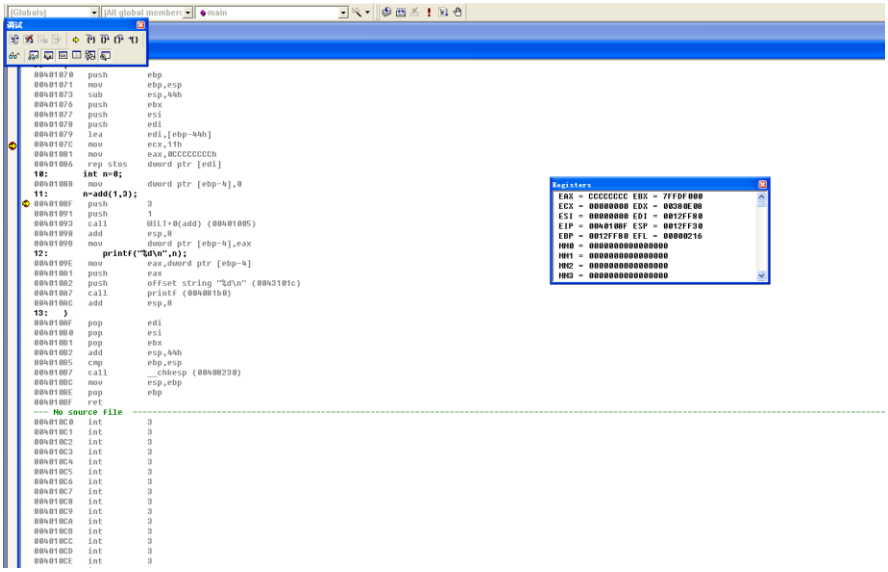
2. 运行项目



3.F9 加断点，F5 进入调试状态



4. 进入 VC 反汇编模式，观察每一个语句对应的汇编语言语句



5. 观察 add 函数调用前后语句

```
11:       n=add(1,3);
0040107F   push        3
00401081   push        1
00401083   call        @ILT+0(add) (00401005)
00401088   add         esp,8
0040108B   mov         dword ptr [ebp-4],eax
```

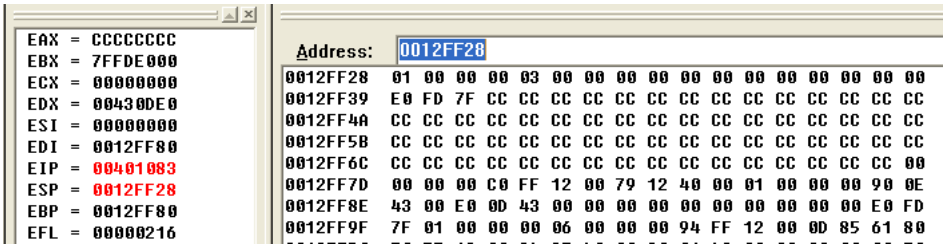
在 add 函数调用前，首先将 add 函数所需要的两个参数从右向左依次压入栈中，然后通过 call 指令调用 add 函数，此时隐含做了两件事情，将 EIP 中的下一条指令的地址入栈，然后跳转到 add 函数所在的代码块中。

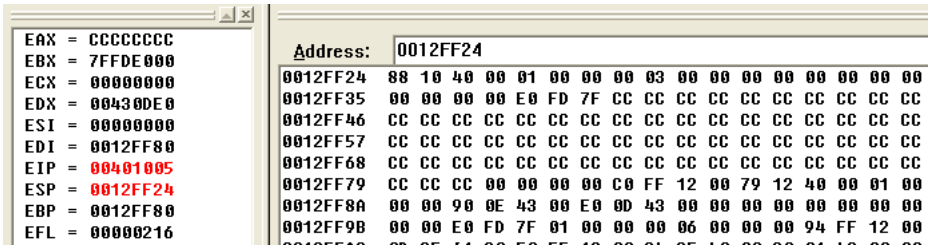
在 add 函数调用后，进行栈平衡操作，将 ESP 恢复到调用 add 函数之前的状态，然后将函数返回值从 EAX 寄存器中转移到局部变量 n 所在的地址处。

6. add 函数内部栈帧切换等关键汇编代码

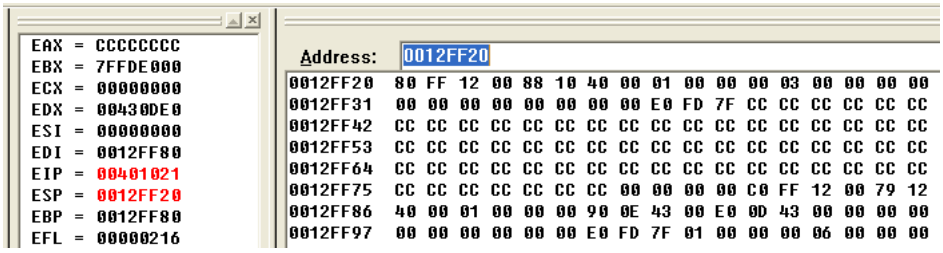
```
00401020   push        ebp
00401021   mov         ebp,esp
00401023   sub         esp,44h
00401026   push        ebx
00401027   push        esi
00401028   push        edi
00401029   lea         edi,[ebp-44h]
0040102C   mov         ecx,11h
00401031   mov         eax,0CCCCCCCCh
00401036   rep stos    dword ptr [edi]
```

在 call 指令执行时，首先将 EIP 中的返回地址入栈，ESP-4，然后设置 EIP 的值，实现从 main 函数到 add 函数的跳转。

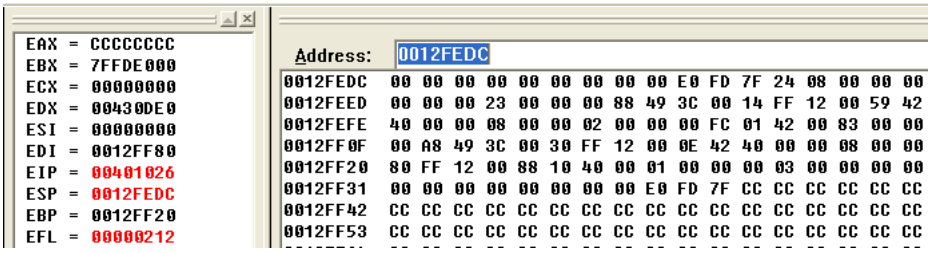




进入 add 函数内部后，首先将原始的 EBP 入栈，以便在函数结束后返回时恢复栈帧。



然后将 EBP 栈底指针上提到 ESP 处，并将 ESP-44h，开辟 add 函数的栈帧。



再将原始的寄存器信息入栈保存，将 EBX，ESI，EDI 入栈。

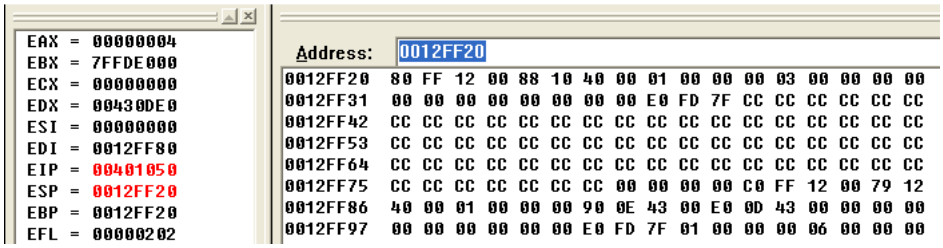
最后对局部变量进行初始化。完成栈帧的转化。

```
6:        return z;
00401048   mov         eax,dword ptr [ebp-4]
7:    }
0040104B   pop         edi
0040104C   pop         esi
0040104D   pop         ebx
0040104E   mov         esp,ebp
00401050   pop         ebp
00401051   ret
```

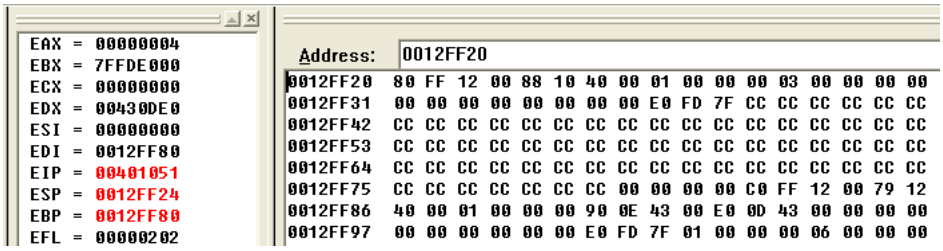
在 add 函数执行完成之后，return z；这时将函数的返回值从局部变量中转移到寄存器 EAX 中，实现函数结果的返回。

随后要清理函数开辟的栈帧，将原始的寄存器信息出栈，恢复状态。

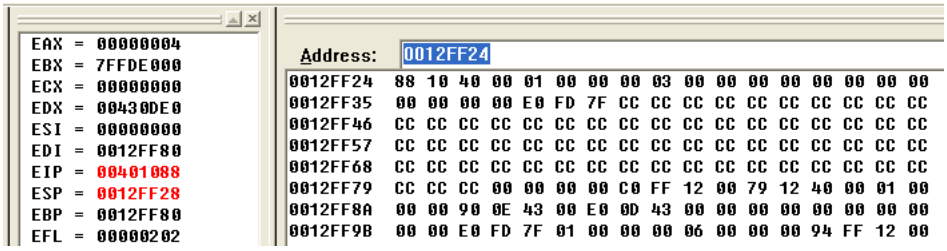
通过 mov esp,ebp 清除函数开辟的栈帧，此时 esp 和 ebp 的值相同，都指向原始栈帧的栈底 EBP。



然后将原始栈帧的栈底 EBP 的值出栈，恢复了原始的 EBP 状态



ret 将当前的栈顶 ESP 中保存原始 EIP 的值出栈到 EIP 中，然后跳回到调用函数的代码块中。



四、心得体会：

这个实验的主要目的是通过反汇编调试来熟悉 IDE 的使用，以及了解函数调用、栈帧切换、CALL 和 RET 指令等汇编语言的实现过程。在这个实验中，我们使用了 VC6 作为开发环境，并根据第二章实验视频 2-1 进行调试。

在实验过程中，我们首先需要了解函数调用的基本原理。当一个函数被调用时，会先将当前函数的返回地址（即下一条指令的地址）保存到栈上，然后跳转到被调用函数的入口地址。被调用函数执行完毕后，会通过 RET 指令将返回地址弹出栈，并跳转回调用函数的下一条指令。

通过调试器，我观察到了在函数调用过程中，EIP（指令指针寄存器）、ESP（栈指针寄存器）和 EBP（基址指针寄存器）等寄存器的变化。在调用函数之前，EIP 指向调用函数的下一条指令地址；在 CALL 指令执行后，EIP 被设置为被调用函数的入口地址；同时，ESP 会向下移动，为被调用函数的局部变量和参数预留空间；EBP 则被设置为当前栈帧的基址，用于访问局部变量和参数。

在被调用函数执行完毕后，通过 RET 指令，返回地址会从栈中弹出，恢复到

调用函数的下一条指令，同时 ESP 和 EBP 也会恢复到调用函数的栈帧。这样就完成了函数的调用和返回过程。

通过这个实验，我深入理解了函数调用的原理和汇编语言中的栈帧切换过程。同时，也加深了对汇编语言中的 CALL 和 RET 指令的理解。 我对 IDE 反汇编调试和汇编语言有了更深入的认识，对函数调用和栈帧切换等汇编语言的实现有了更清晰的理解。这对于进一步学习和应用汇编语言非常有帮助。