**《软件安全》实验报告**

**姓名：郭子涵 学号：2312145 班级：信息安全、法学双学位班**

**实验名称：**

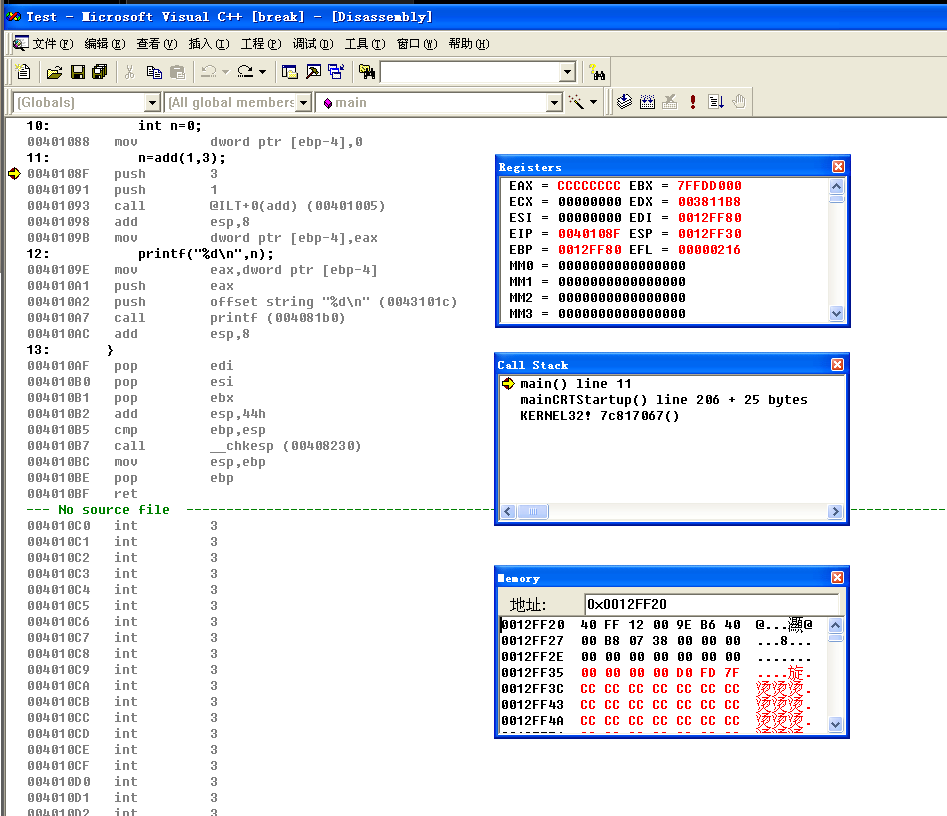
IDE反汇编实验

**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，观察汇编语言：**



1. **观察add函数调用前后语句**

**2.1调用前**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08 | **10:           int n=0;**  00401088   mov         dword ptr [ebp-4],0  *//ebp当前主函数栈帧的基地址，-4：栈的增长方向由高地址向低地址*  **11:           n=add(1,3);**  0040108F   push        3  00401091   push        1*//参数由右向左入栈*  00401093   call        @ILT+0(add) (00401005)  *//首先esp改变里面存的00401098及返回地址；*  *第二条操作：代码区跳转，其起始地址为虚拟地址；ebp入栈，esp的值赋值给ebp（此时esp=ebp）* |

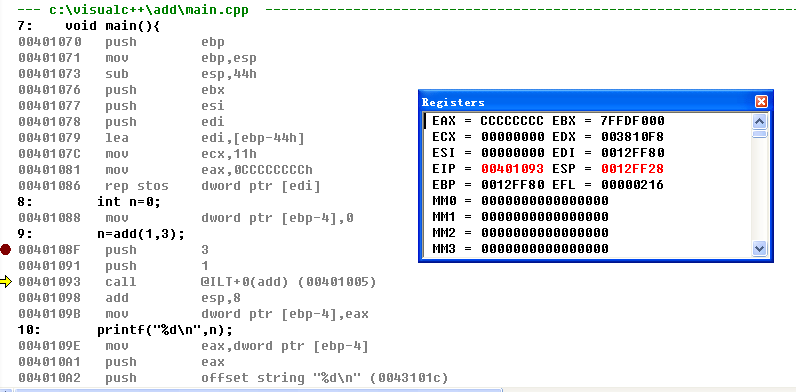
**2.2调用后**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18 | 00401098   add         esp,8*//ESP回归调用函数之前*  0040109B   mov         dword ptr [ebp-4],eax*//eax寄存器中存储为add函数返回的值，将计算的结果赋值给局部变量n*  12:           printf("%d\n",n);*//输出函数*  0040109E   mov         eax,dword ptr [ebp-4]  004010A1   push        eax  004010A2   push        offset string "%d\n" (0043101c)  004010A7   call        printf (004081b0)  004010AC   add         esp,8  13:       }  004010AF   pop         edi  004010B0   pop         esi  004010B1   pop         ebx  004010B2   add         esp,44h  004010B5   cmp         ebp,esp  004010B7   call        \_\_chkesp (00408230)  004010BC   mov         esp,ebp  004010BE   pop         ebp  004010BF   ret |

1. **add函数内部栈帧切换等关键汇编代码**

**3.1从断点出按F10调试**

此时esp=0012FF28,EIP=00401093

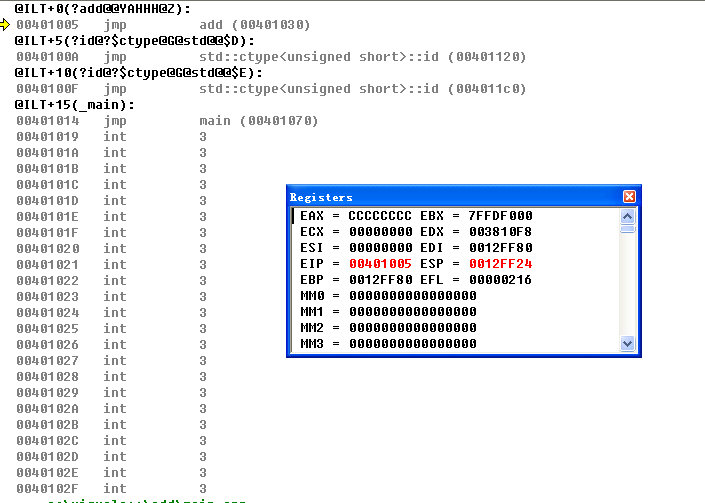


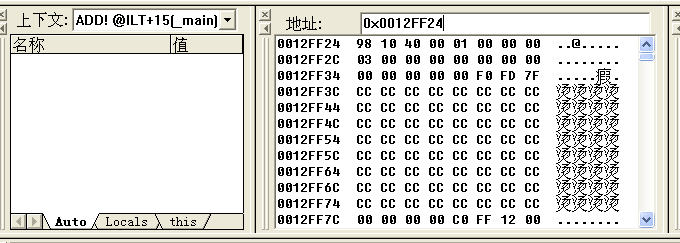
**3.2 点击step into(F11)并不会直接跳转到代码区**

ESP地址减小4变为0012FF24，里面存为00401098（即函数的返回地址）;

EIP地址为00401005，即图中jump指令，将跳转到add函数。

**总结：call指令分两步：第一步是将调用的下一条指令入栈，作为返回地址；第二步是修改EIP的值**





**3.3再次点击调试（step into）跳转到add函数的代码区**

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | 3:        { *//代码区跳转，其起始地址为虚拟地址*00401030   push        ebp*//ebp入栈，esp寄存器的值减少4*  00401031   mov         ebp,esp*//ebp=esp,变成了以前栈顶的值，为add函数设置栈的基址*  00401033   sub         esp,44h*//为add函数开辟栈帧，esp变为0012FEDC*    00401036   push        ebx  00401037   push        esi  00401038   push        edi*//主函数可能用到的值压入栈中*  00401039   lea         edi,[ebp-44h]*//*  0040103C   mov         ecx,11h*//计数寄存器赋值，循环11h次，把44h的空间进行初始化为0CCCCCCCCh，如下图查看地址所示：*  00401041   mov         eax,0CCCCCCCCh    00401046   rep stos    dword ptr [edi]  4:            int z=0;  00401048   mov         dword ptr [ebp-4],0  5:            z=x+y;  0040104F   mov         eax,dword ptr [ebp+8]*//ebp加是参数相关，减是局部变量相关*  00401052   add         eax,dword ptr [ebp+0Ch]  00401055   mov         dword ptr [ebp-4],eax  6:            **return** z;  00401058   mov         eax,dword ptr [ebp-4]*//eax承载函数的返回值（z）*  7:        }  0040105B   pop         edi  0040105C   pop         esi  0040105D   pop         ebx*//三个寄存器依次弹出,esp变为0012FEDC*    0040105E   mov         esp,ebp*//ebp赋值给esp,esp指向主函数的返回地址*  00401060   pop         ebp*//恢复栈帧,此时esp所致内容即为函数的返回地址*      00401061   ret*//返回主函数的下一条指令，eip的值即为00401098* |

**心得体会：**

通过实验，在实际操作中掌握了很多知识点。

1. 通过本次实验学会了如何再虚拟机上安装操作系统，并且在xp操作系统上使用VC6,编写程序，进行反汇编，设置断点，查找地址的内容等等。
2. 明晰了在调用函数时的原理，得出RET指令实际就是执行了Pop EIP，将eip寄存器弹出，并且变为返回函数的地址；本次实验对函数调用过程有了更深入的理解，call指令实际分为两步：第一步是将调用的下一条指令入栈，作为返回地址；第二步是修改EIP的值
3. 此外，通过本次实验，掌握了多个汇编语言的用法，本次上级实验也应用到一些上学期学习的《汇编语言与逆向技术》的知识，通过设置断点，一步一步调试，在本实验中比较难理解的最后循环初始变量的代码部分，让人一目了然，受益匪浅。