《漏洞利用及渗透测试基础》实验报告

姓名：张洋　　学号：2111460　　班级：信安二班

实验名称：

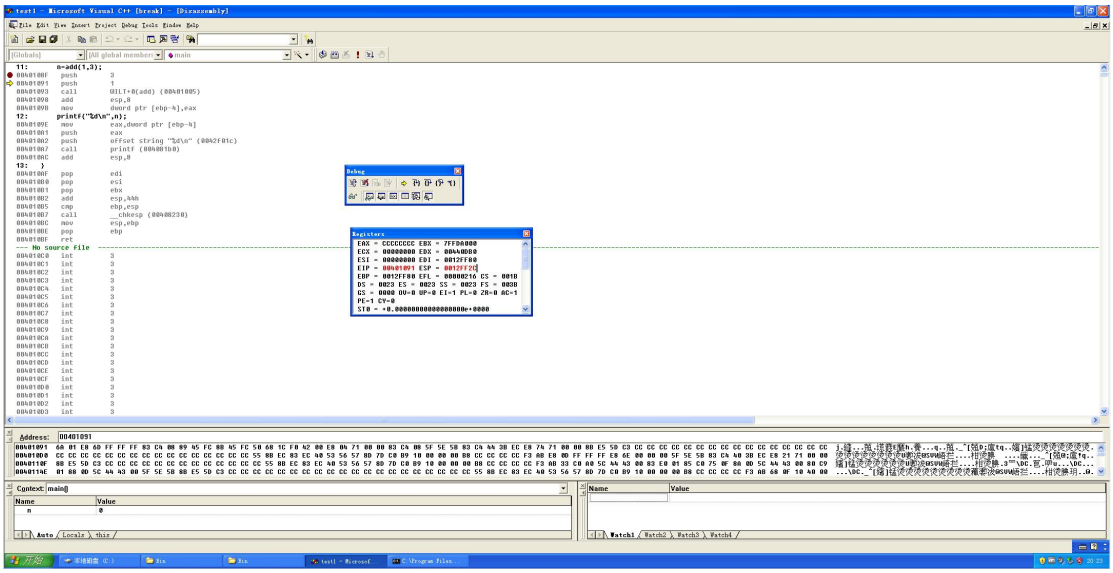
IDE 反汇编实验

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



2. 观察 add 函数调用前后语句

调用 add 函数前，实现参数从右到左入栈。

push 3：ESP 向低地址扩展，由 0012FF30 变为 0012FF2C。

push 5：ESP 向低地址扩展，由 0012FF2C 变为 0012FF28。

call 指令调用 add 函数后执行 add esp,8，执行完这条指令后可以发现 esp 的地址回到执行第一条指令（push 3）之前。

3. add 函数内部栈帧切换等关键汇编代码

```
2:    int add(int x,int y)
3:    {
00401030   push        ebp
00401031   mov         ebp,esp
00401033   sub         esp,44h
00401036   push        ebx
00401037   push        esi
00401038   push        edi
00401039   lea         edi,[ebp-44h]
0040103C   mov         ecx,11h
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]
00401052   add         eax,dword ptr [ebp+0Ch]
00401055   mov         dword ptr [ebp-4],eax
6:        return z;
00401058   mov         eax,dword ptr [ebp-4]
7:    }
0040105B   pop         edi
0040105C   pop         esi
0040105D   pop         ebx
0040105E   mov         esp,ebp
00401060   pop         ebp
00401061   ret
```

CALL 指令执行进入 add 函数内部。ESP 变为 0012FF24 并将返回地址 00401098（add 指令的地址）压入栈。此时 EIP 值为 00401005，接下来将进行的操作为对代码区的调整。进行以下操作：

(1)将 EBP 地址入栈（主函数栈帧地址），发生栈帧调整。ESP 变为 00401098。

(2)为 add 函数开辟栈帧，EBP 由 0012FF80（原主函数栈帧地址）变为 0012FF20（ESP 的值），为 add 函数设置了基址 0012FF20，把栈顶 esp 设置在 0012FEDC 处。

(3)将 ebx，esi，edi（主函数可能用到的寄存器的值）依次入栈。ESP 变为 0012FED0

(4)进行 11h 次循环，将 add 函数的栈帧空白处均初始化为 CC

(5)将 edi，esi，ebx 依次出栈。ESP 变为 0012FEDC

(6)调整栈帧，栈顶指向 00401098。RET 后 EIP 变为 00401098，返回到原函数 call 的下一条指令。

(7)add 函数结束后调整栈帧，回到主函数，栈帧的基底地址 EBP 变为 0012FF80。

心得体会：

通过实验，我掌握了 CALL 指令和 RET 指令的用法：CALL 指令调用一个过程，指挥处理器从新的内存地址开始执行，使用 RET 指令将处理器转回到该过程被调用的程序点上。CALL 指令将其返回地址压入堆栈，再把被调用过程的地址复制到指令指针寄存器。当过程准备返回时，它的 RET 指令从堆栈把返回地址弹回到指令指针寄存器。RET 指令实际就是执行了 pop EIP。

此外，通过本实验，我还掌握了多个汇编语言的用法：

（1） 进栈指令 PUSH 和出栈指令 POP

指令的基本功能：PUSH 指令在程序中常用来暂存某些数据，而 POP 指令又可将这些数据恢复。

（2） 加法指令 ADD 和减法指令 SUB

指令支持的寻址方式：他们两个操作数不能同时为存储器寻址。即为除源操作数为立即数的情况外，源操作数和目地操作数必须有一个寄存器寻址方式。

（3） 循环指令 LOOP

指令的基本功能：① (CX)←(CX)－1 ② 若(CX)≠0，则(IP)←(IP)当前＋位移量，否则循环结束。