《XP 环境 VC6 反汇编》实验报告

姓名：蒋薇　　学号：2110957　　班级：计科 1 班

实验名称：

IDE 反汇编实验

实验要求：

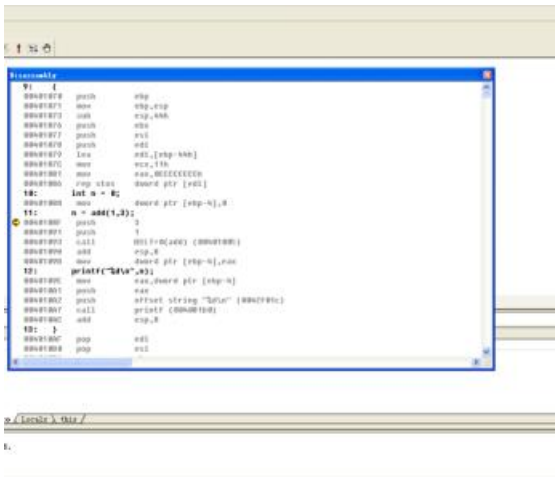
根据第二章示例 2-1，在 XP 环境下进行 VC6 反汇编调试，熟悉函数调用、栈帧切换、CALL 和 RET 指令等汇编语言实现，将 call 语句执行过程中的 EIP 变化、ESP、EBP 变化等状态进行记录，解释变化的主要原因。

实验过程：

1. 进入 VC 反汇编

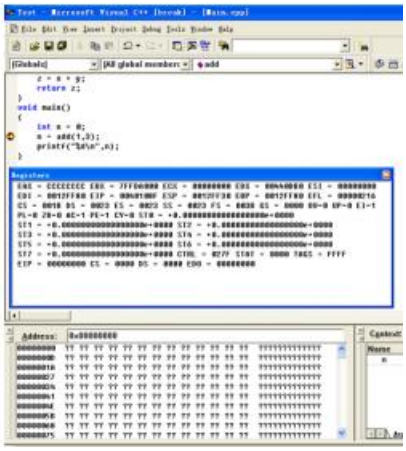
进入 Windows XP Professional-WMware Workstation 环境,打开 VC6,创建 protect Test,创建 Main.cpp,编译运行无误后，F9 设置断点，Build->start debug->go,

可右键点击 Go To Disassembly，进入反汇编 Disassembly



点击 View ->Debug Windows->Memory 可得

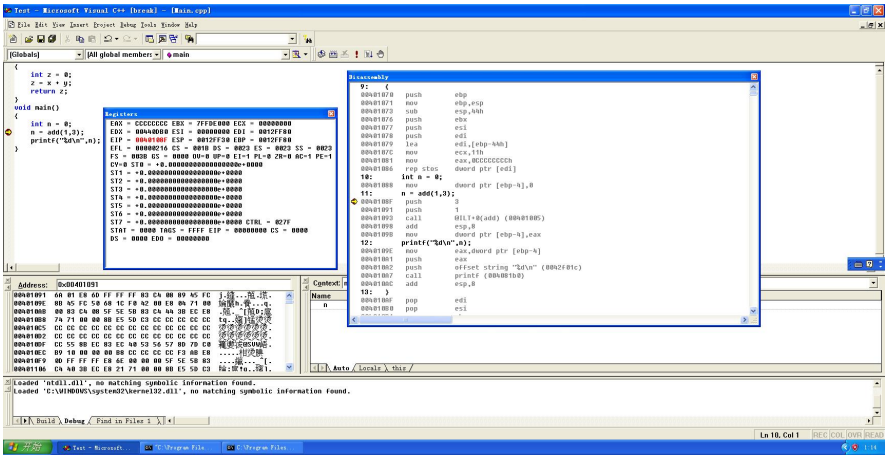
点击 View->Debug Windows->Registers 可得



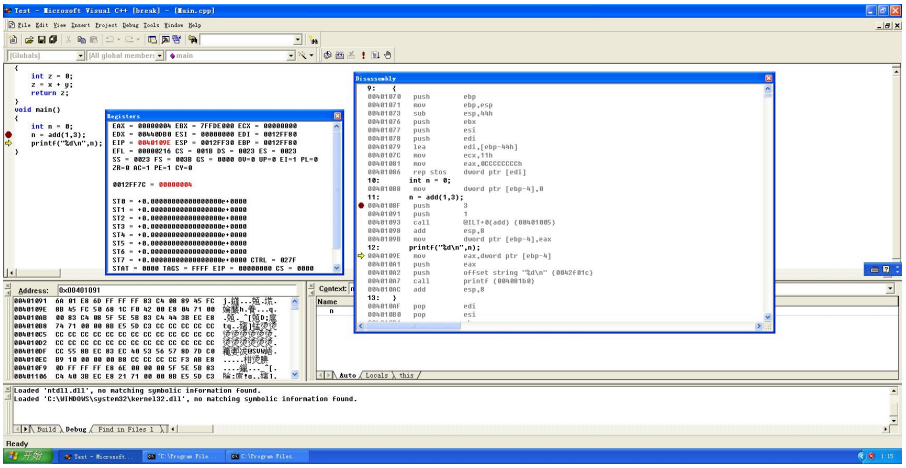
2. 观察 add 函数调用前后语句

[ebp - 4]为 n 的地址

调用一个函数前需，参数入栈，返回地址入栈，转到 add 函数代码处执行，栈帧调整，如图



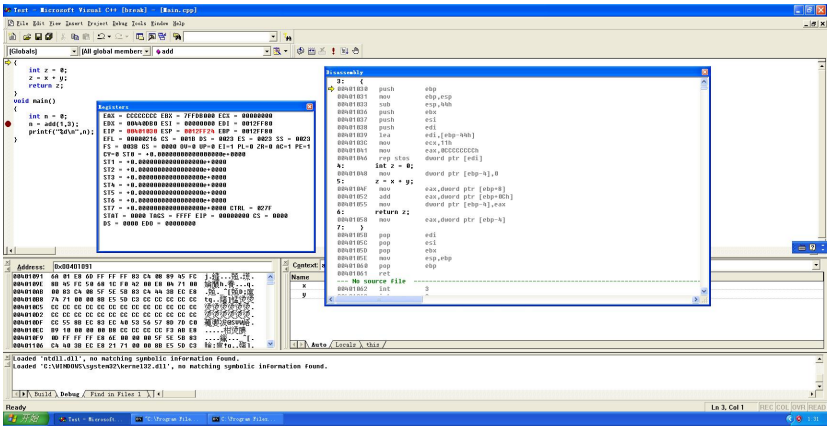
后需恢复栈帧，返回地址出栈，如图



3. add 函数内部栈帧切换等关键汇编代码

函数参数入栈，参数从右向左，

遇到 call 指令，F11，返回地址入栈，F11 代码区跳转 add 函数虚拟地址 00401030



eip 存放下一条要操作的指令

push ebp,esp 变化，esp-4,总是向低地址增长，[0012FF20]存的是 0012FF80,是主函数栈帧底部；

mov ebp,esp，ebp 的值变化，为栈顶值，即为 add 函数设置一个基址

sub esp,44h 向低地址走，即为 add 函数开辟栈帧

push ebx, push esi,push edi, 为可能用到的寄存器的值入栈，每次 push,esp 向上走，即 esp 值变小

lea edi,[edp -44h] mov ecx,11h mov eax,0cccccccch 初始化，edi 抬高 44h 后开辟起始地址,ecx 计数寄存器，往往使用循环，rep 循环前缀指令，循环 11h 次，利用 cch 把 44h 空间初始化,直到 ecx 为 0，循环结束；

(int z = 0)mov [ebp -4],0 , [ebp - 4]局部变量 z 的地址，赋值为 0

(z = x + y) [ebp + 8],ebp 加和参数相关，ebp 减和局部变量相关，mov eax,[ebp + 8] 即 eax 被赋值为 1,add eax,[ebp + 0ch],eax 加 3，mov [epb - 4],eax,即最后的结果存于[ebp - 4]处

(return z) mov eax,[epb - 4],用 eax 接受返回的 z 值,并未真正完成

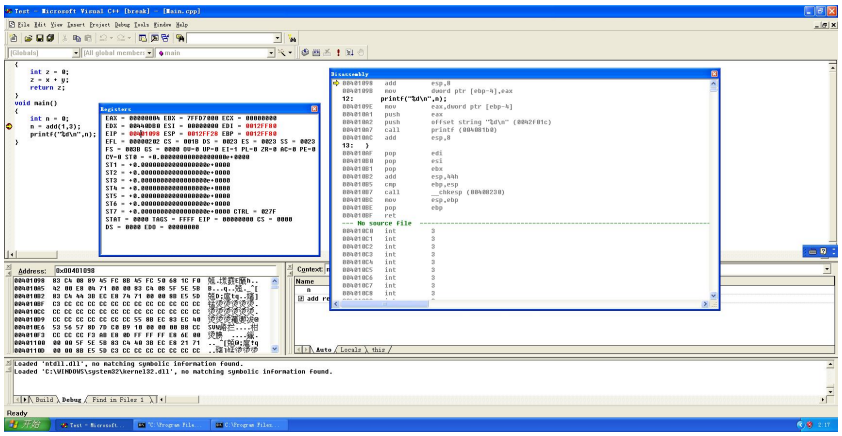
保存栈帧，恢复栈帧，pop edi,pop esi,pop ebx,对应 3 个 push

mov esp,ebp pop ebp 与之前操作对应

ret eip= 00401061,esp = 0012FF24,[esp] =00401098,指向主函数返回地址,ebp =0012FF80，

F11 eip = 00401098,返回主函数

add esp,8 mov [ebp - 4],eax 即 n = 4.



心得体会：

函数被调用时，系统栈会为这个函数开辟一个新的栈帧，并把它压入栈中。每个栈帧对应一个未运行完的函数，栈帧中保存该函数的返回地址和局部变量。函数返回时，系统栈弹出函数对应栈帧。

eip 指向下一条等待执行的指令地址，esp 指向栈顶，ebp 指向栈底，

通过实验，掌握了 RET 指令的用法；

RET 指令实际就是执行了 Pop EIP

此外，通过本实验，掌握了多个汇编语言的用法