《漏洞利用及渗透测试基础》实验报告

姓名：赵悦蛟 学号：2313650 班级： 1071

**实验名称：**

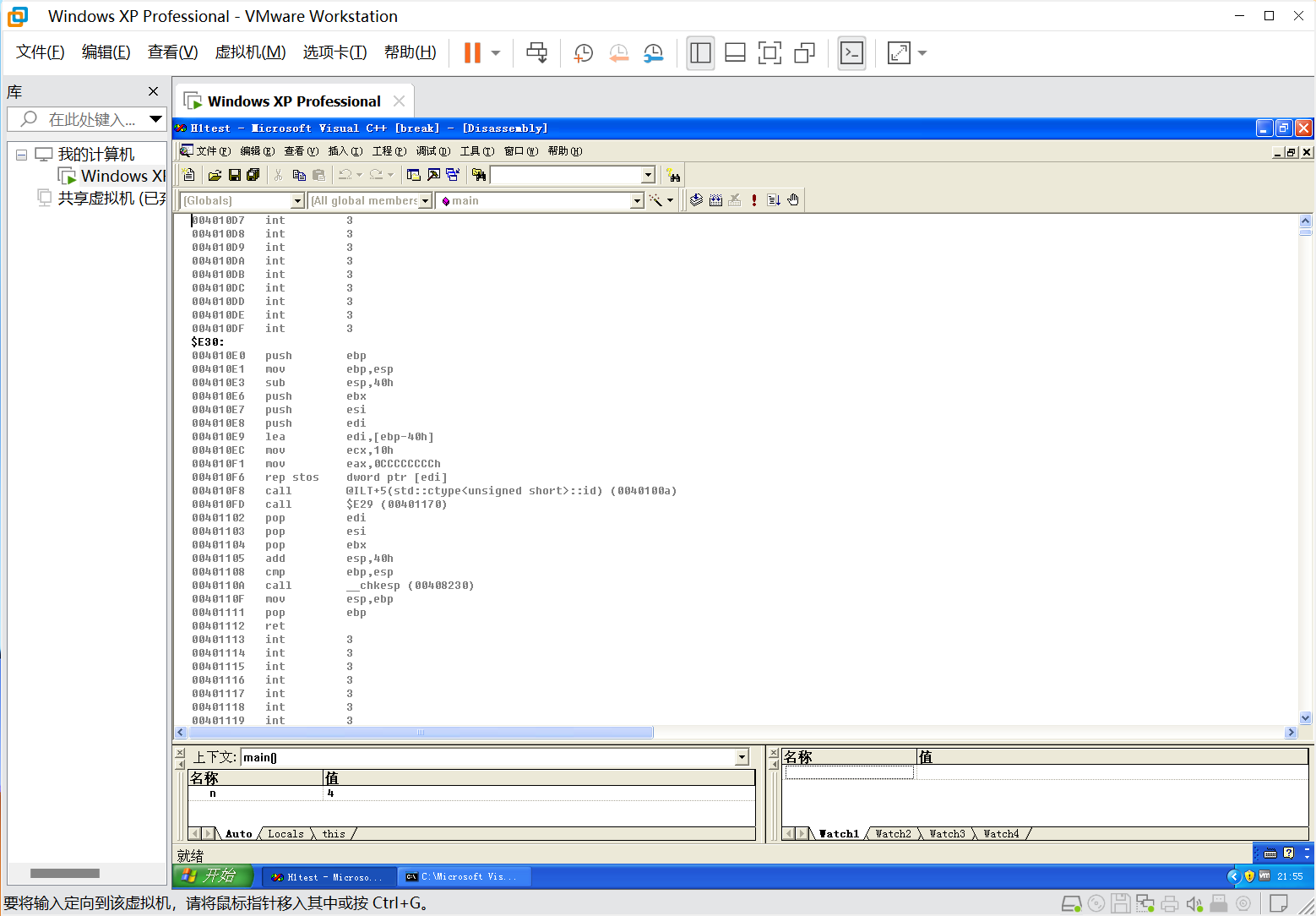
IDE反汇编实验

**实验要求：**

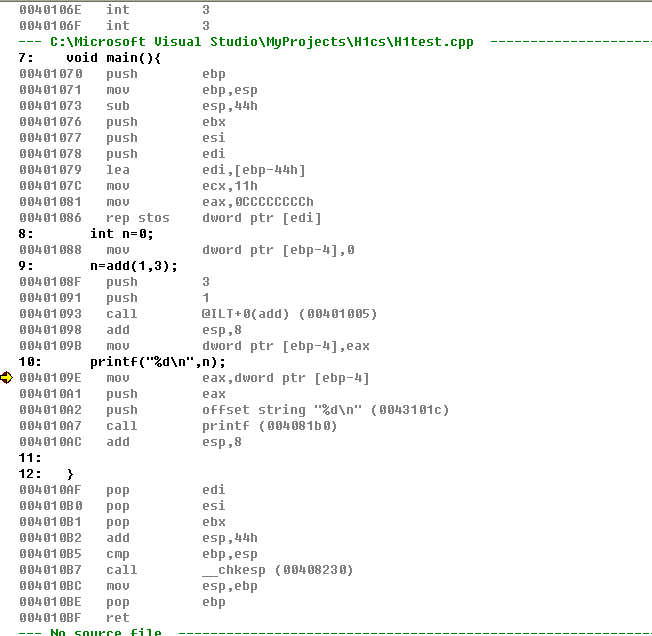
根据第二章示例2-1，在XP环境下进行VC6反汇编调试，熟悉函数调用、栈帧切换、CALL和RET指令等汇编语言实现，将call语句执行过程中的EIP变化、ESP、EBP变化等状态进行记录，解释变化的主要原因。

**实验过程：**

1. 进入VC反汇编

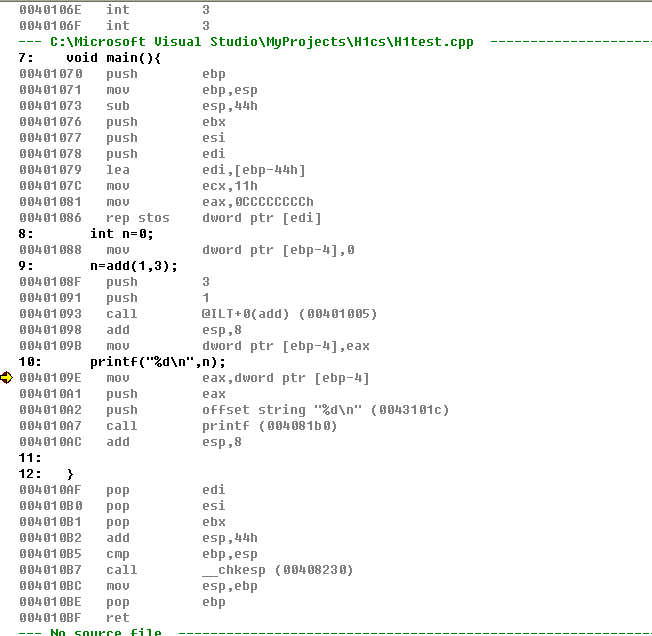


在vs6.0中写好代码，再printf行添加断点，进入调试状态，之后再通过alt+8快捷键进入反汇编模式。



2. 观察add函数调用前后语句

Add函数调用前：



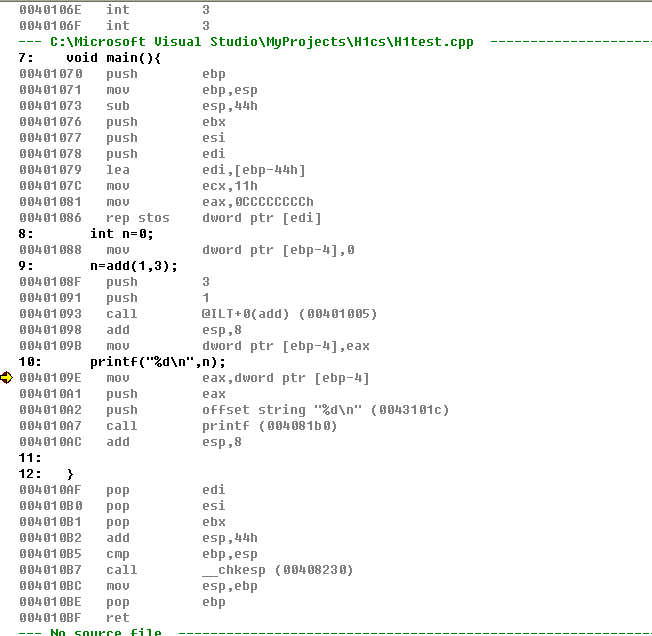
ebp表示基址指针寄存器，在函数开头先push ebp,用于保存调用函数的基址指针。接着建立了新的栈帧。

mov ebp,esp语句将当前栈指针（esp）的值复制到基址指针寄存器（ebp）中，从而建立新的栈帧，使ebp指向当前函数的栈帧基址。

sub语句为局部变量分配了栈空间。接下来的push语句是将寄存器的值压入栈中。

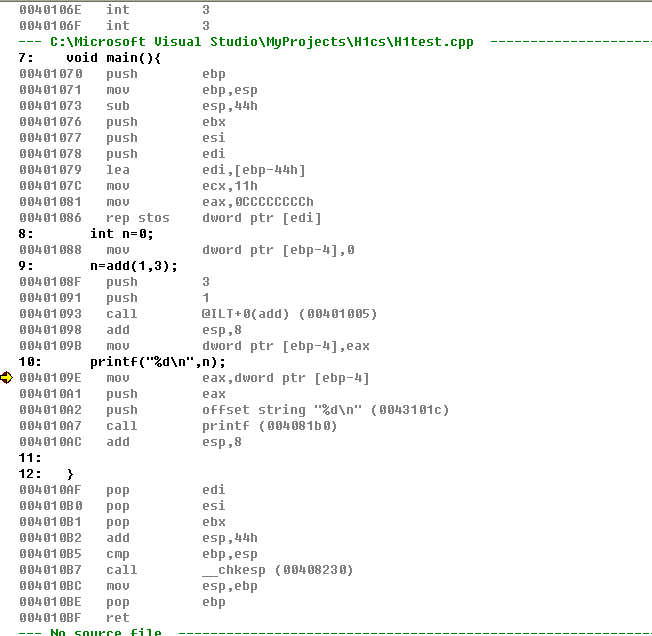
lea edi,[ebp-44h]将效地址[ebp-44h]加载到EDI寄存器中，进行了一个初始化。然后mov语句将ecx寄存器设置为11h,eax寄存器设置为0CCCCCCCCh，mov dword ptr [ebp-4], 0是将0存储到[ebp-4]指向的内存位置。对应int n = 0。

Add函数调用后：



这是add函数调用完之后的语句。在这个过程中：分别将[ebp-4]指向的内存位置的值加载到EAX寄存器中，将这个值压入栈中，将“%d/n”的值压入栈中。然后call printf调用printf,输出n的值。再将栈指针寄存器（ESP）增加8字节，清理栈上为printf函数参数分配的空间，从栈中弹出之前保存的EDI、ESI和EBX寄存器的值。再将基址指针寄存器（EBP）的值复制到栈指针寄存器（ESP）中，从栈中弹出之前保存的EBP寄存器的值。ret从当前函数返回。

3.add函数内部栈帧切换等关键汇编代码



将3，1按顺序压入栈中，call语句调用add函数。add esp, 8: 将栈指针寄存器（ESP）增加了8字节。mov dword ptr [ebp-4], eax: 将EAX寄存器的值存储到[ebp-4]指向的内存位置，这是将add函数的返回值赋值给变量n。

**心得体会：**

通过实验，掌握了RET指令的用法：ret指令是从当前函数返回到调用它的地方。当call指令调用函数是会产生一个返回值，把返回地址压入栈中，ret指令就会从栈中弹出这个地址，把eip设置为这个地址。

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。