《软件安全》实验报告

姓名：辛浩然　　学号：2112514　　班级： 信息安全、法学

实验名称：

IDE 反汇编实验

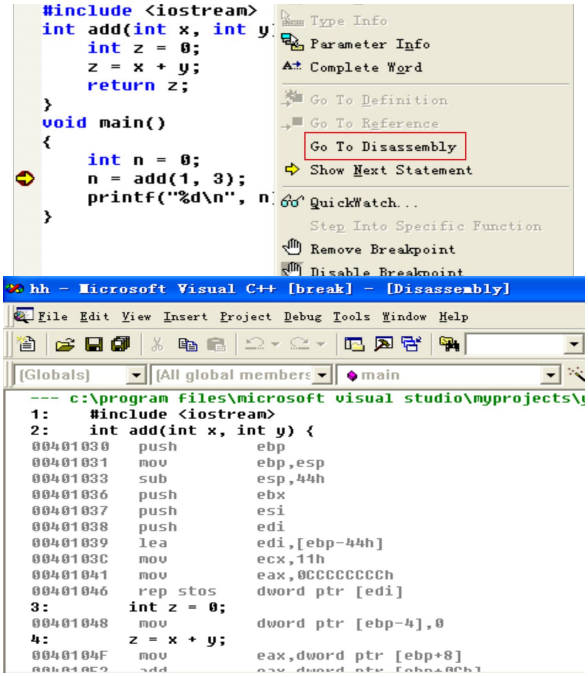
实验要求：

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

在主函数中设置一个断点，按 F5 进入调试状态，右击，在弹出的快捷菜单中选择“转到反汇编”。



2. 观察 add 函数调用前后语句

语句截图：

```
10:        n = add(1, 3);
0040108F   push          3
00401091   push          1
00401093   call          @ILT+0(add) (00401005)
00401098   add           esp,8
0040109B   mov           dword ptr [ebp-4],eax
```

代码分析：

（1）参数自右向左入栈。

0040108F　push　3

00401091　push　1

（2）调用函数，返回地址入栈。

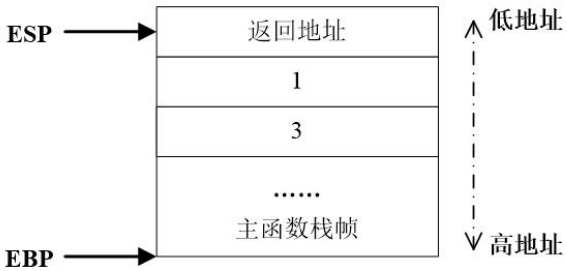
00401093　call　@ILT+0(add) (00401005)

（3）调用后：

由调用者清栈，将 eax 中存放的函数返回值赋值给 main 函数中的变量。

00401098　add　esp,8

0040109B　mov　dword ptr [ebp-4],eax



3. add 函数内部栈帧切换等关键汇编代码

代码截图：

```
1:     #include <iostream>
2:
3:     int add(int x, int y) {
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
```

代码分析：

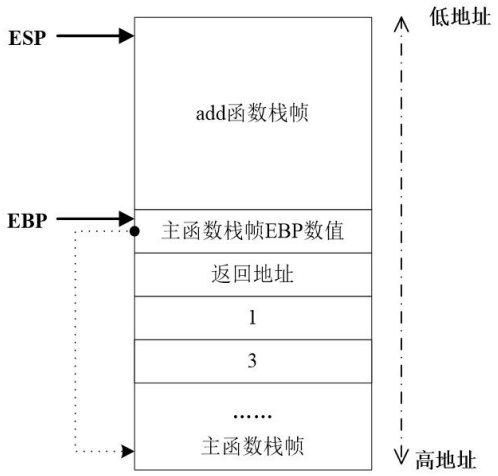
（1）栈帧切换

00401030　push　　ebp

00401031　mov　　ebp,esp

00401033　sub　　esp,44h

首先将 **ebp 的值入栈**；然后，**将 esp 的值赋值给 ebp**；然后，**将 esp 抬高**，得到栈大小为 44h，为 add 函数分配空间。



（2）函数状态保存

00401036　push　　ebx

用于保存现场， ebx 作为内存偏移指针使用。

00401037　push　　esi

用于保存现场，esi 是源地址指针寄存器。

00401038 push edi

用于保存现场，edi 是目的地址指针寄存器。

00401039 lea edi,[ebp-44h]

将 ebp-44h 地址装入 EDI

0040103C mov ecx,11h

设置计数器数值，即将 ECX 寄存器赋值为 11h

00401041 mov eax,0CCCCCCCCh

向寄存器 EAX 赋值

00401046 rep stos dword ptr [edi]

重复 ecx 次，将 eax 的内容存储到 edi 指向的位置，**每次将 edi(取决于方向标志)递增或递减 4 个字节**。循环将栈区数据都初始化为 CCh。

（3）执行函数体

00401048 mov dword ptr [ebp-4],0

将 z 初始化为 0

0040104F mov eax,dword ptr [ebp+8]

将寄存器 eax 的值设置为形参 x 的值

00401052 add eax,dword ptr [ebp+0Ch]

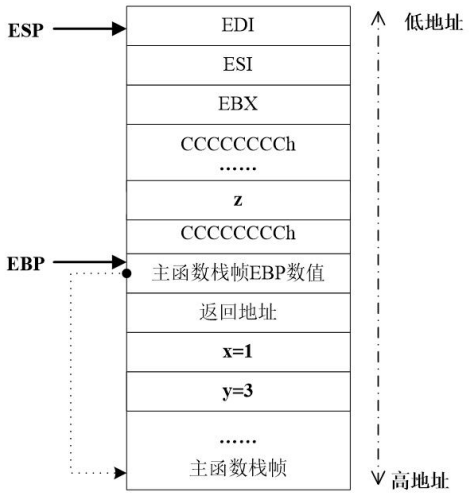
将寄存器 eax 累加形参 y 的值

00401055 mov dword ptr [ebp-4],eax

将 eax 的值赋值给 z

00401058 mov eax,dword ptr [ebp-4]

将 z 的值储存到 eax 寄存器中



（4）恢复状态

0040105B　pop　edi

0040105C　pop　esi

0040105D　pop　ebx

0040105E　mov　esp,ebp

00401060　pop　ebp

恢复寄存器的值，**恢复 esp 和 ebp 的值**

00401061　ret

根据返回地址**恢复 eip 值，相当于 pop eip**

心得体会：

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。

熟悉了函数调用、栈帧切换、CALL 和 RET 指令等汇编语言实现，熟悉了 call 语句执行过程中的 EIP、ESP、EBP 等的变化。

此外，通过本实验，掌握了多个汇编语言的用法。