《软件安全》实验报告

姓名：邢清画　　学号：2211999　　班级：1023

实验名称：

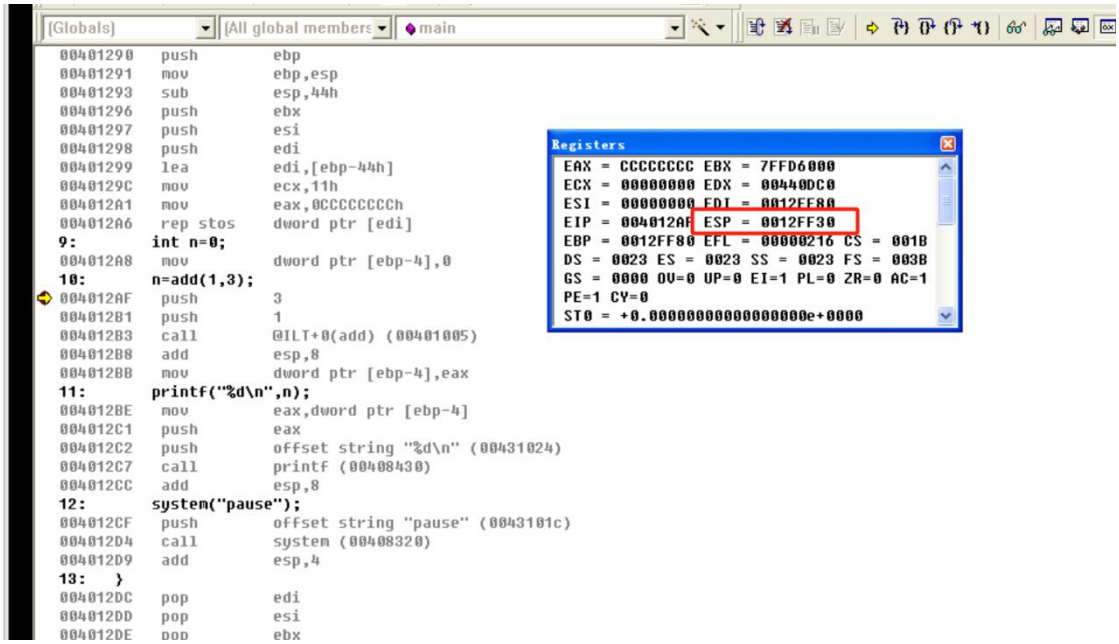
IDE 反汇编实验

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。

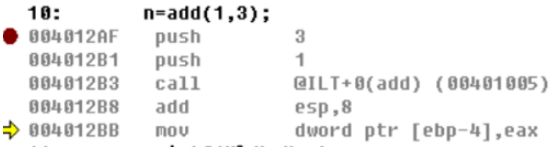
实验过程：

1. **进入 VC 反汇编**

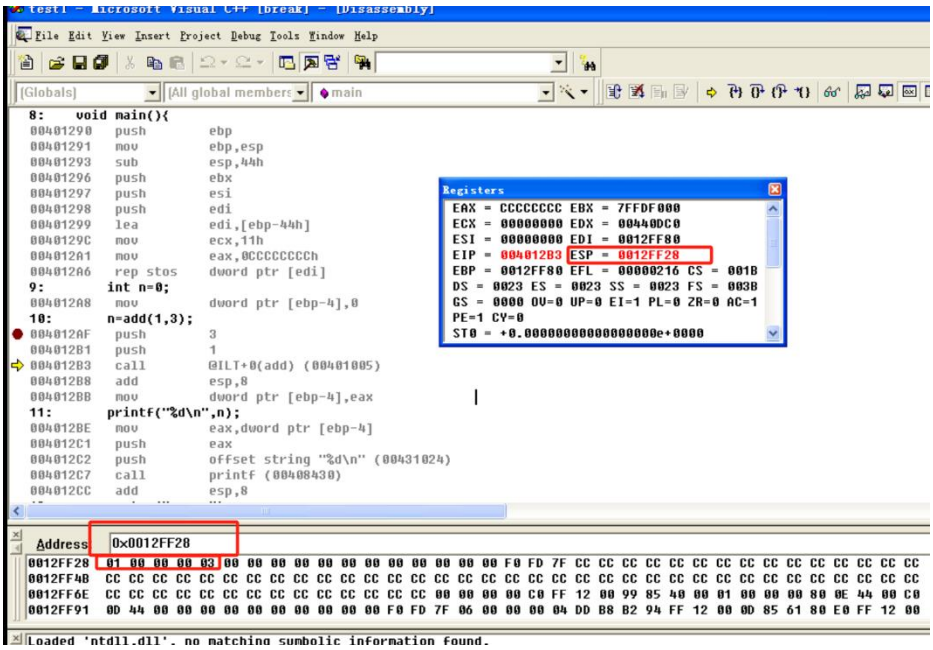


在 add 调用前，栈不断向上走，ESP 的值减小（由 0012FF30 到 0012FF2C），变为更低的地址。

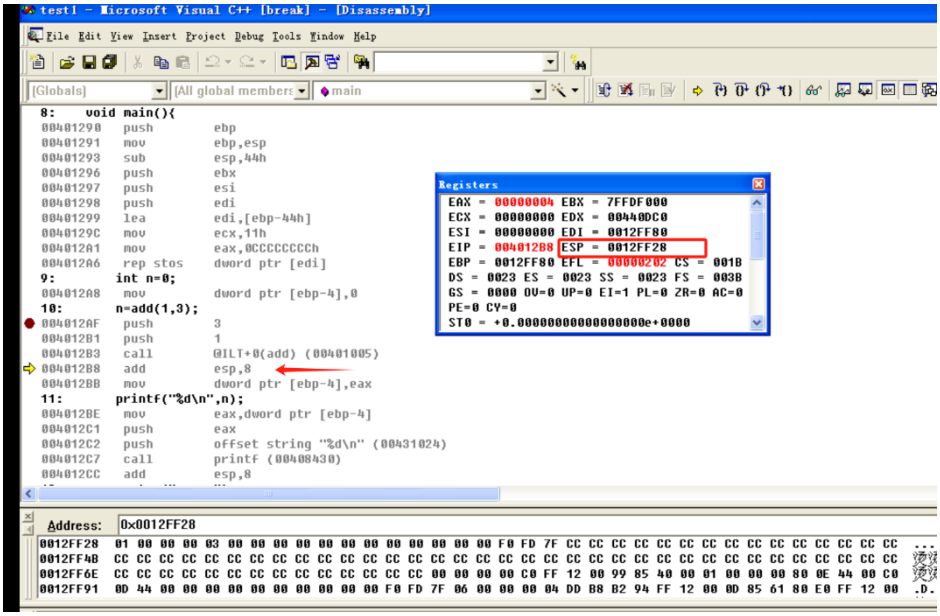
2. **观察 add 函数调用前后语句**



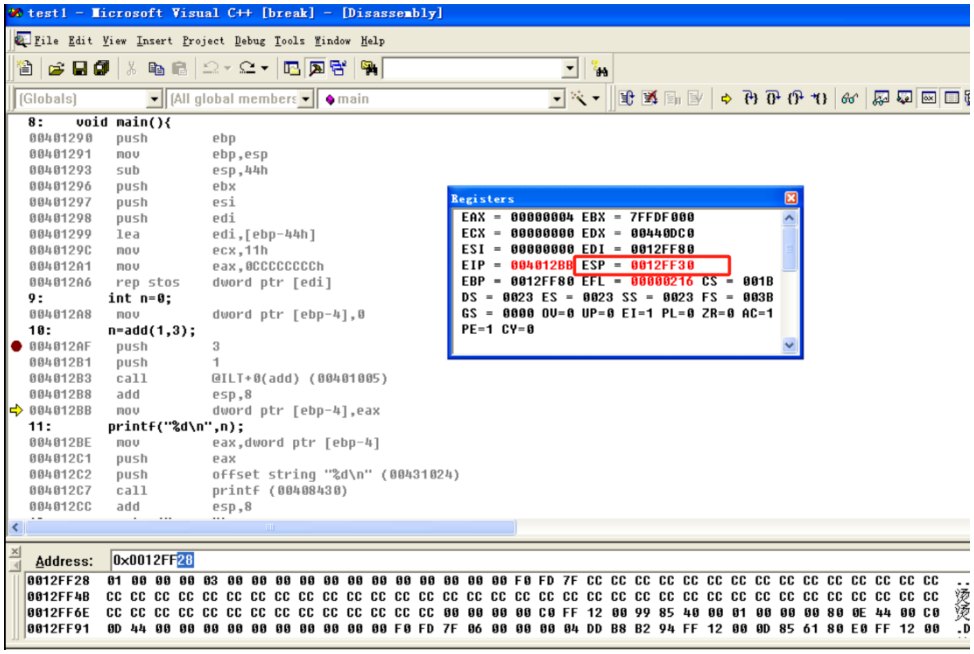
进入 add 函数后，参数由右向左依次入栈；EIP 寄存器存放的下一条指令（下一条步的地址）数值递增；ESP 的值不断减小（由高地址向低地址变化）。进行到 call 语句时，进行了两个关键步骤，1.返回地址入栈（EIP 中的值入栈），ESP 的值改变（由 0012FF28 变为 0012FF24）；2.代码区跳转，EIP 寄存器中的值是 call 指令的参数，指向 add 函数入口的地址，进入 add 函数内部。（详细图片记录如下）



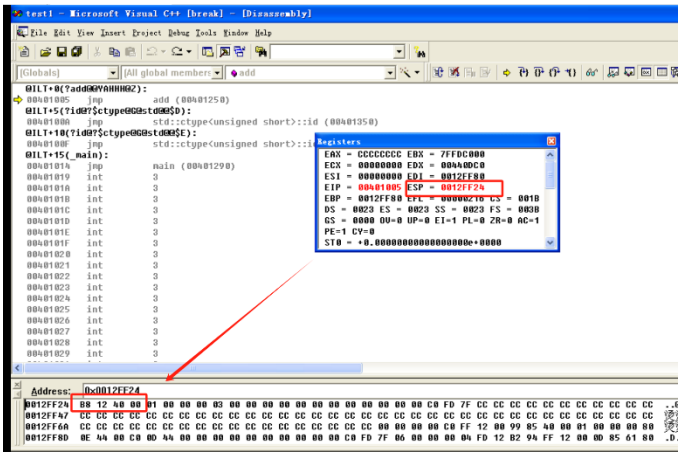
如上图变化所示，3 和 1 依次入栈，ESP 不断减小（至 0012FF28）。

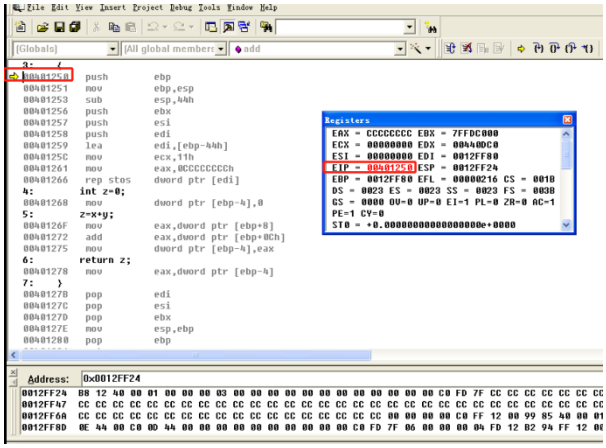


运行完 call 之后,ESP 与之前相同（0012FF28）



add 函数调用完,ESP 恢复到调用之前的状态（0012FF30），EAX=4，赋值给 n,完成整个调用。



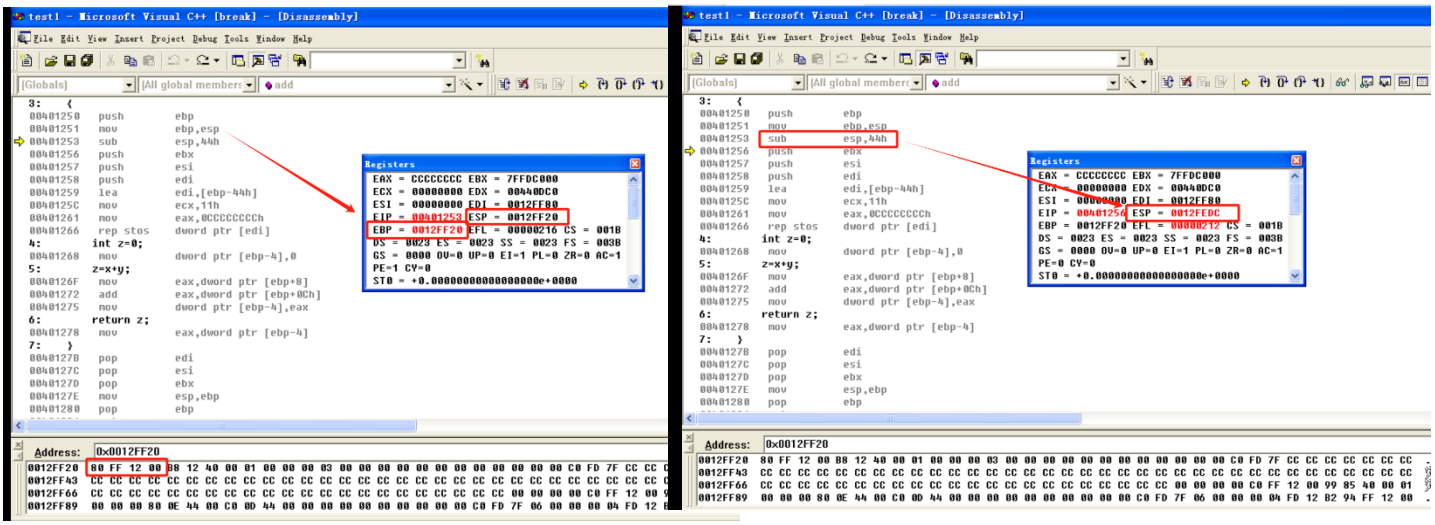


恰好是 add 函数运行完要返回的地址，进行 call 的返回地址入栈和代码区跳转。

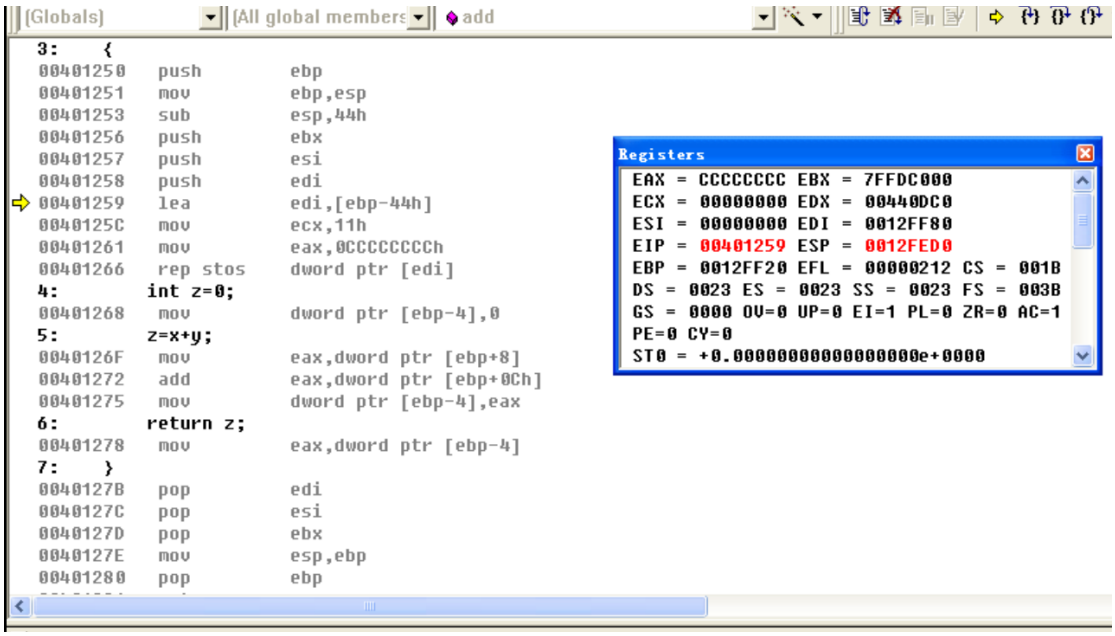
3. **add 函数内部栈帧切换等关键汇编代码**

```
3:    {
00401250  push        ebp
00401251  mov         ebp,esp
00401253  sub         esp,44h
00401256  push        ebx
00401257  push        esi
00401258  push        edi
00401259  lea         edi,[ebp-44h]
0040125C  mov         ecx,11h
00401261  mov         eax,0CCCCCCCCh
00401266  rep stos    dword ptr [edi]
```

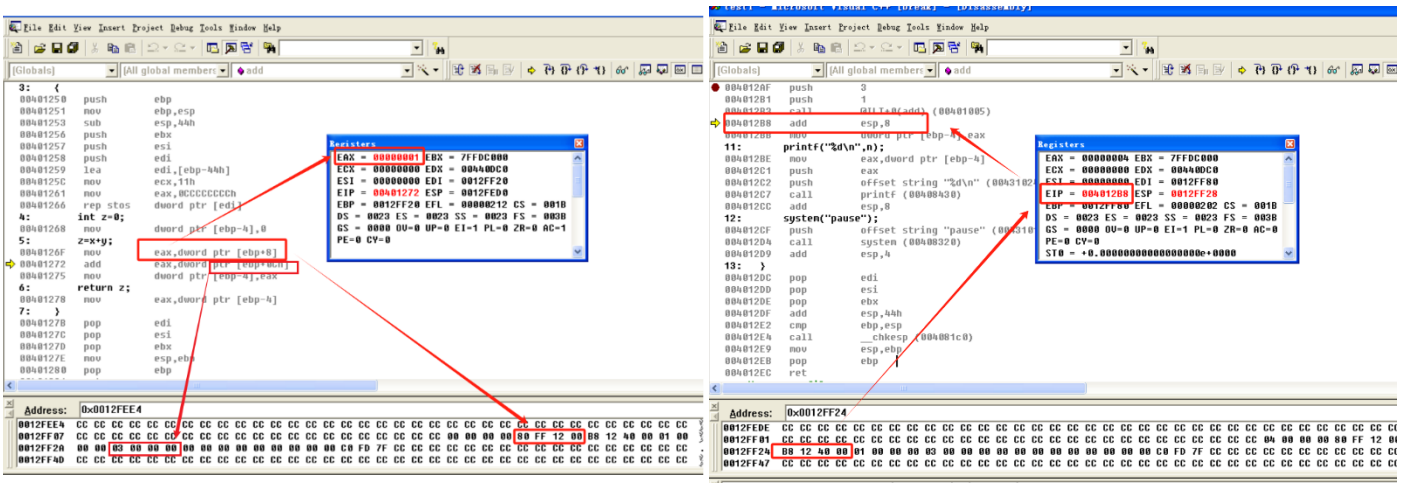
栈帧切换部分，EBP 寄存器的值入栈用来储存当前函数的栈帧起点，

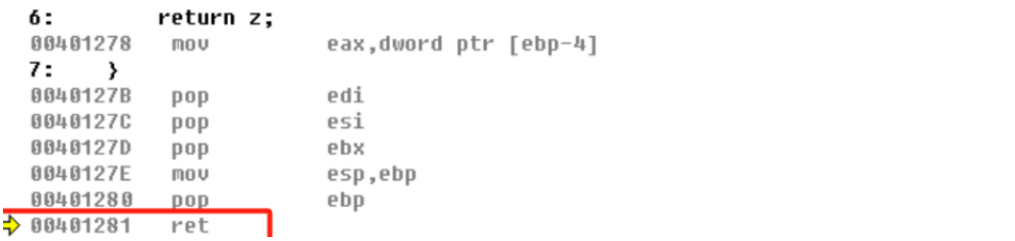


Mov 把 ESP 的值赋给 EBP，栈顶向上移，然后将 ESP 寄存器的值减去 44h 完成栈帧切换。

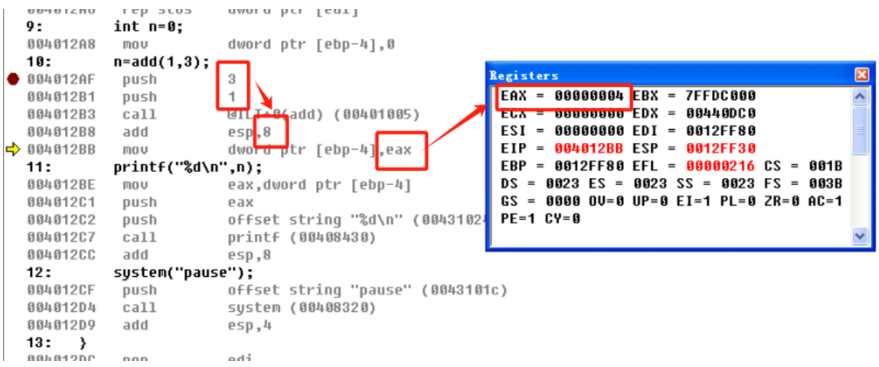


ESP 的值不断减小，栈顶不断提高，提高了 44h。下面四行代码是进行初始化的操作，ecx 计数寄存器，进行循环操作 11 次。将 0 赋给 ebp-4





栈顶指向主函数返回地址，返回值放入 EAX，之前压入栈的数据出栈，将 EBP 的值赋给 ESP，将 add 函数的栈帧清空， EBP 出栈，执行 ret 指令，返回地址放入 EIP 寄存器完成跳转，返回到 main 函数中将栈顶指针加 8，恢复到之前的状态，完成调用。



心得体会：

1.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

2. add 指令表示加法，sub 指令表示减法，mov 完成将后面的值赋值给前面，ebp+和参数有关，ebp-和局部变量有关，call 指令进行函数调用。

3. EIP 指令的值变化，进行程序的跳转（跳转到 EIP 的目标地址），从而执行目标地址中的命令，函数的调用和返回，即保存当前的 EIP 中的指令地址，返回时重新将该地址放入 EIP 寄存器。

4.ecx 计数寄存器通常意味着要进行循环。