反汇编实验

姓名：汤清云 学号：2013536

实验步骤：

- 在 XP 环境下进行 VC6 反汇编调试
- 熟悉栈帧调用、函数切换的汇编语言实现
- 熟悉 CALL 指令、RET 指令的汇编语言实现

实验报告：

1. Call 指令执行中的 EIP 变化及原因。

   Push 3：EIP 永远指向下一条等待执行指令地址。在执行完此条后由 0040108F（push 3 的地址）指向下一条 00401091（push 1 的地址）

   Push 1：EIP 永远指向下一条等待执行指令地址。在执行完此条后由 00401091（push 1 的地址）指向下一条 00401093（call 的地址）

   Call 指令：EIP 值为 00401005，即接下来将进行的操作为对代码区的调整，一直持续到 00401046，均为 int add（）函数的编译过程，顺序进行，之后进行 11h 次循环，将 add 函数的栈帧空白处均初始化为 CC。之后顺序进行至 00401061.

   Ret 后 EIP 变为 00401098，返回到原函数 call 的下一条指令。之后顺序执行。

2. Call 指令执行中的 ESP 变化及原因。

Push 3：将变量 3 由右向左入栈，ESP 由 0012FF30 变为 0012FF2C；esp 向低地址扩展，故而 esp 实际值-4；

Push 1：将变量 1 由右向左入栈，ESP 由 0012FF2C 变为 0012FF28；ESP 抬高，故而 ESP 实际值-4；

Call 指令：ESP 变为 0012FF24：将返回地址 00401098（add 指令的地址）压入栈。

ESP 变为 0012FF20：将 EBP 地址入栈（主函数栈帧地址），发生栈帧调整。

ESP 变为 0012FEDC：为 add 函数开辟栈帧，将栈顶设置在 0012FEDC 处。

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

ESP 变为 0012FEDC：将 edi，esi，ebx 依次出栈。

ESP 变为 0012FF20：调整栈帧。栈顶指向 00401098

ESP 变为 0012FF24：pop EBP。

ESP 变为 0012FF30：将参数 3，1 出栈。

3. Call 指令执行中的 EBP 变化及原因。

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。

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

附图：

XP 环境：

反汇编过程：