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。

1. 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。

1. 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：

