**[C/C++函数调用过程分析](http://www.cnblogs.com/biyeymyhjob/archive/2012/07/20/2601204.html)**

这里以一个简单的C语言代码为例，来分析函数调用过程

代码：

[复制代码](javascript:void(0);)

1 #include <stdio.h>

2

3 int func(int param1 ,int param2,int param3)

4 {

5 int var1 = param1;

6 int var2 = param2;

7 int var3 = param3;

8

9 printf("var1=%d,var2=%d,var3=%d",var1,var2,var3);

10 return var1;

11 }

12

13 int main(int argc, char\* argv[])

14 {

15 int result = func(1,2,3);

16

17 return 0;

18 }

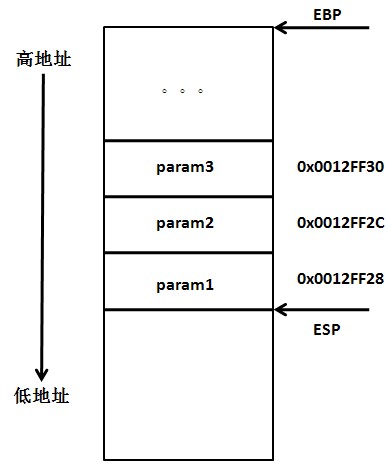
[复制代码](javascript:void(0);)

首先说明，在堆栈中变量分布是从高地址到低地址分布，EBP是指向栈底的指针，在过程调用中不变，又称为帧指针。ESP指向栈顶，程序执行时移动，ESP减小分配空间，ESP增大释放空间，ESP又称为栈指针。

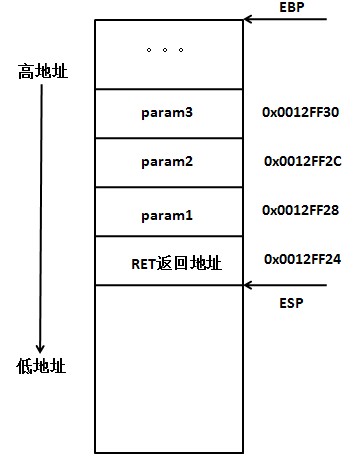
下面来逐步分析函数的调用过程

**1.函数main执行，main各个参数从右向左逐步压入栈中，最后压入返回地址**

**2.执行第15行，3个参数以从左向右的顺序压入堆栈，及从param3到param1，栈内分布如下图：**



**3.然后是返回地址入栈：此时的栈内分布如下：**

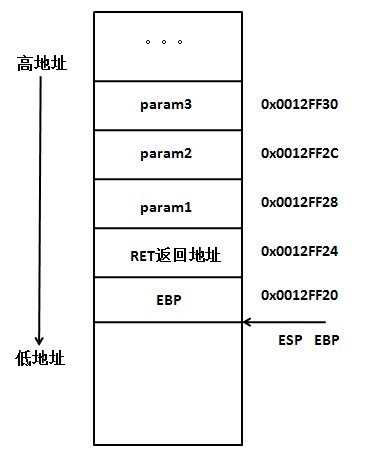


**4.第3行函数调用时，通过跳转指令进入函数后，函数地址入栈后，EBP入栈，然后把当前ESP的值给EBP，对应的汇编指令**：

push ebp

mov ebp esp

   此时栈顶和栈底指向同一位置，栈内分布如下：

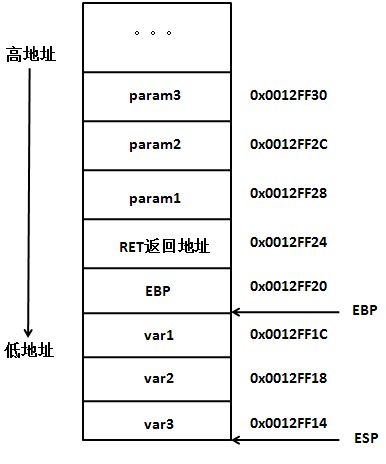


**5.第5行开始执行， int var1 = param1; int var2 = param2; int var3 = param3;按申明顺序依次存储**。对应的汇编：

mov 0x8(%ebp),%eax

mov %eax,-0x4(%ebp)

  其中将[EBP+0x8]地址里的内容赋给EAX，即把param的值赋给EAX，然后把EAX的中的值放到[EBP-4]这个地址里，即把EAX值赋给var1，完成C代码 int var1 = param1，其他变量雷同。



**6.第9行，输出结果，第10行执行 对应的汇编代码：**

mov -0x4(%ebp),%eax

 最后通过eax寄存器保存函数的返回值；

**7.调用执行函数完毕，局部变量var3，var2，var1一次出栈，EBP恢复原值，返回地址出栈，找到原执行地址，param1，param2，param3依次出栈，函数调用执行完毕。图略**