**在汇编程序中调用C函数**

2018年03月10日 21:30:57 [ARM的程序员敲着诗歌的梦](https://me.csdn.net/u013490896) 阅读数：2887

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/u013490896/article/details/79511747

**一、参数如何传递**

汇编程序调用C函数时，函数的入口参数使用栈来传送,参数的传递顺序是从右到左。即函数最后（最右边的）一个参数先入栈，而最左边的第一个参数最后入栈，然后执行 CALL 指令去调用C函数。

**二、参数的清除**

在C函数返回后，汇编程序需要把先前压入栈中的函数参数清除掉，即调用者负责清除参数占用的栈空间。

比如要调用的C函数和要传递的参数是printSomething (arg1, arg2, arg3, arg4);   
那么在汇编程序中应该这样写：

pushl arg4

pushl arg3

pushl arg2

pushl arg1

call printSomething

addl $0x10, %esp

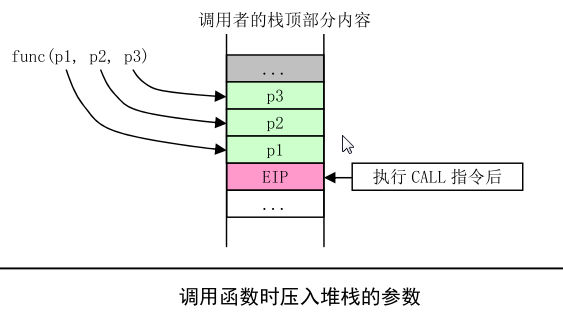
* 1
* 2
* 3
* 4
* 5
* 6

**三、返回值的传递**

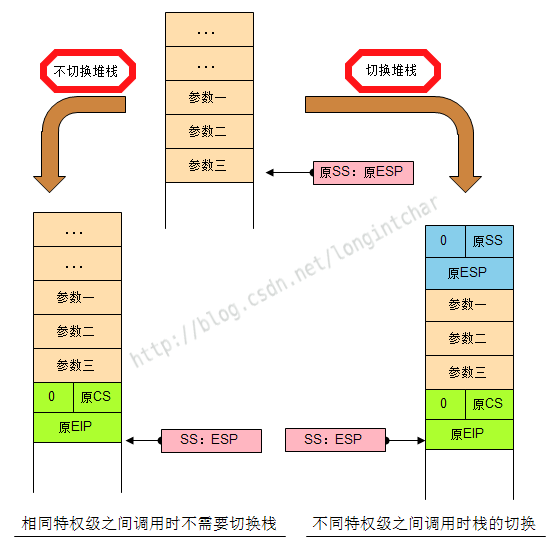
C函数的返回值如果是32位整数，则保存在eax寄存器；如果是[64位](https://www.baidu.com/s?wd=64%E4%BD%8D&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)整数，则保存在edx:eax寄存器。

**四、关于**[**返回地址**](https://www.baidu.com/s?wd=%E8%BF%94%E5%9B%9E%E5%9C%B0%E5%9D%80&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)

在执行CALL指令时，CPU会把CALL指令的下一条指令的地址（返回地址）压入栈中（见图中 EIP）。



**五、关于栈的切换**

如果调用还涉及到代码特权级变化，那 么CPU还会进行栈切换，这个过程就比较复杂了。可以参考我的博文：[通过调用门进行控制转移](http://blog.csdn.net/u013490896/article/details/51367377)   


**注意**：Linux内核中只使用中断门和陷阱门方式处理特权级变化时的调用情况，并没有使用CALL指令来处理特权级变化的情况。

**六、用JMP指令代替CALL指令**

我们可以不用CALL指令而采用JMP指令来同样达到调用C函数的目的。方法是：在所有参数入栈后，人工把下一条要执行的指令的地址（返回地址）压入栈中，然后直接使用JMP指令跳转到被调用函数的入口地址去执行。当被调用函数执行其最末尾的ret指令时，就会把我们人工压入栈中的返回地址弹出到EIP寄存器中，使执行流程返回到主函数。