**通过调用门进行控制转移 ——《x86汇编语言：从实模式到保护模式》读书笔记29**

2016年05月10日 22:30:14 [ARM的程序员敲着诗歌的梦](https://me.csdn.net/u013490896) 阅读数：1450更多

所属专栏： [《x86汇编语言：从实模式到保护模式》读书笔记](https://blog.csdn.net/column/details/protectedmode.html)

 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/51367377

**通过调用门进行控制转移**

**1. 关于堆栈切换**

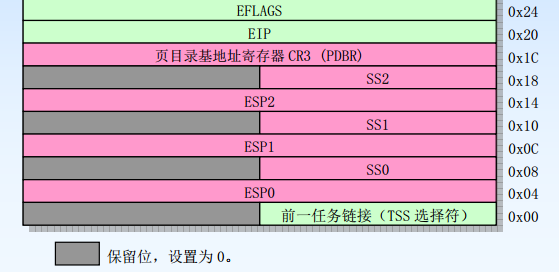
如果通过调用门把控制转移到了更高特权级的非一致代码段中，那么CPL就会被设置为目标代码段的DPL值，并且会引起堆栈切换。为什么要切换堆栈呢？原因有以下几点：   
1. 因为栈段的特权级必须同CPL保持一致；   
2. 防止高特权级程序由于栈空间不足而崩溃；   
3. 防止低特权级程序通过共享的栈有意或无意地干扰高特权级程序。

为了切换栈，每个任务除了自己的固有栈之外，还必须额外定义一套或多套栈，具体是多少取决于任务的特权级别。   
0特权级的任务不需要额外的栈，因为除了从调用高特权级的例程（通常是操作系统例程）返回外，不允许将控制从特权级高的代码段转移到特权级低的代码段——操作系统不会引用可靠性比自己低的代码；1特权级的任务需要额外定义一个DPL为0的栈，以便将控制转移到0特权级时使用；2特权级的任务需要额外定义两个栈，其DPL分别为0和1；3特权级的任务最多额外定义三个栈，其DPL分别为0、1、2.

以下文字摘自《Intel Architecture Software Developer’s Manual Volume 3:System Programming》的4.8.5节——Stack Switching.

Each task must define up to 4 stacks: one for applications code (running at privilege level 3) and one for each of the privilege levels 2, 1, and 0 that are used. (If only two privilege levels are used[3 and 0], then only two stacks must be defined.)

操作系统负责为任务用到的所有特权级分配栈空间和创建栈段描述符，并且在任务的TSS中填写栈段的选择子和ESP的初始值（下图是TSS的一部分）。



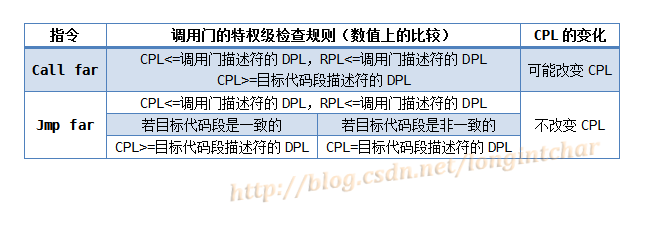
有几点需要注意：   
1. 每个栈必须可读可写，并且具有足够的空间来存放以下信息：   
(1)调用过程的SS、ESP、CS和EIP寄存器的内容   
(2)被调用过程的参数和临时变量所需使用的空间   
(3)当隐含调用一个异常或者中断过程时标志寄存器EFLAGS和出错码使用的空间   
2. 由于一个过程可以调用其他过程，因此每个栈必须有足够的空间来容纳多帧信息   
3. TSS中的中的SSx、ESPx（x=0,1,2）字段是静态的，除非软件进行修改，处理器从来不会改变它们。举例来说，假设操作系统为一个用户任务的TSS填写了ESP0，其值为0x800；当这个任务通过调用门进入0特权级的代码段时，会切换到0特权级堆栈，堆栈指针ESP的初始值就是0x800；返回时，假设ESP变成了0x808，处理器并不会把0x808更新到TSS中的ESP0域；下次再通过调用门进入0特权级代码段时，使用的还是当初设置的静态值0x800。

**2. 通过调用门进行控制转移和返回的具体过程**

**2.1 转移的过程**

首先，通过调用门进行控制转移，可以使用jmp far或者call far指令。指令执行时，段选择子必须指向调用门，32偏移量可以是任意值（会被[CPU](https://www.baidu.com/s?wd=CPU&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)忽略）。

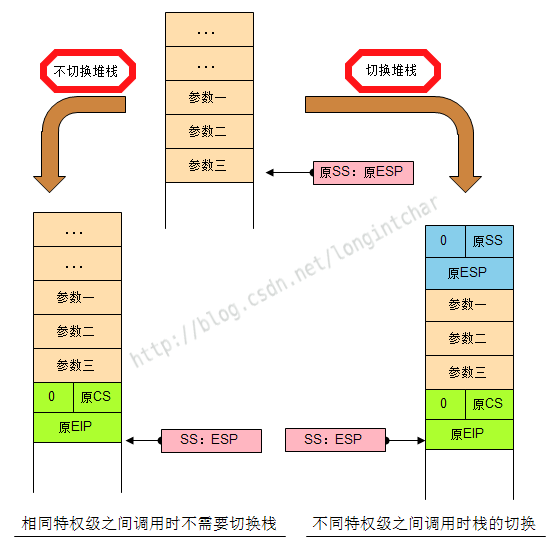
其次，必须符合下表的特权级检查规则。



再次，当使用call far指令通过调用门转移控制时，如果改变了CPL，则必须切换栈，即从当前任务的固有栈切换到与目标代码段特权级相同的栈上。栈的切换是由处理器固件自动进行的。

当前栈是由SS和ESP的当前内容所指示的。要切换到的新栈的相关信息位于当前任务的TSS中，处理器知道如何找到它。栈切换过程如下：   
1. 根据目标代码段的DPL（也就是新的CPL）到当前任务的TSS中读取新栈的选择子和栈指针。在读取栈选择子、栈指针或者栈段描述符的过程中，任何违反段界限的错误都将导致产生一个无效TSS异常。   
2. 检查栈段描述符的特权级和类型是否有效，若无效同样产生一个无效TSS异常。   
3. 临时保存SS和ESP的当前值，把新栈的选择子和栈指针加载到SS和ESP中。然后把临时保存的SS和ESP的内容压入新栈中。   
4. 根据调用门描述符中“参数个数”字段，把旧栈中的所有参数复制到新栈中。如果参数个数为0，则不复制参数。   
5. 将当前CS和EIP的内容压入新栈。通过调用门实施的控制转移一定是远转移，所以要压入CS和EIP。   
6. 从调用门描述符中把目标代码段的选择子和段内偏移值传送到CS和EIP中，开始执行被调用过程。

相反，如果没有改变特权级别，则不切换栈，继续使用调用者当前的栈，只是在原来的基础上压入当前CS和EIP的内容。



另外，如果通过调用门的控制转移是jmp far指令发起的，结果就是“肉包子打狗——有去无回”，且没有特权级变化，也不需要切换栈。相反，如果是call far指令发起的，则可以使用远返回指令retf把控制返回到调用者。

**2.2. 返回的过程**

对于相同特权级的返回，CPU从堆栈中弹出EIP和CS；会发生特权级改变的远返回仅允许返回到低特权级程序中，即返回到的代码段的DPL在数值上要大于CPL。返回的全部过程如下：

1. 检测被调用者栈中CS寄存器的RPL字段值，以确定在返回时特权级是否发生改变。
2. 弹出并使用被调用过程栈上的值加载EIP和CS寄存器。在此过程中会对代码段描述符和代码段选择子的RPL进行特权级与类型检查。
3. 如果远返回指令是带参数的，则将参数和ESP寄存器的当前值相加，以跳过被调用者栈中的参数部分，最后的结果是ESP寄存器指向调用者SS和ESP的压栈值。注意，retf指令的参数必须等于调用门中所有参数的总字节数之和。
4. 如果返回时需要改变特权级，则从栈中将ESP和SS弹出，并把值代入寄存器ESP和SS，切换到调用者的栈。
5. 如果远返回指令是带参数的，则将参数和ESP寄存器的当前值相加，以跳过调用者栈中的参数部分，最后的结果是调用者的栈恢复平衡。
6. 如果返回时需要改变特权级，则检查DS,ES,FS和GS的内容，如果段选择子指向数据段或者非一致代码段且段描述符的DPL在数值上小于返回后的新CPL，那么就把数值0传送到该段寄存器。