[**switch\_to 理解**](https://www.cnblogs.com/elnino/p/4342603.html)

 看到任务切换函数switch\_to，感觉很晦涩，于是在网上查了一些资料，现在终于有些眉目，特记录于此，以方便大家参考，有什么错误或不足之处，还请大家指出~

switch\_to源码

/\*  
 \*    switch\_to(n) should switch tasks to task nr n, first  
 \* checking that n isn't the current task, in which case it does nothing.  
 \* This also clears the TS-flag if the task we switched to has used  
 \* tha math co-processor latest.  
 \*/  
#define switch\_to(n) {\  
struct {long a,b;} \_\_tmp; \  
\_\_asm\_\_("cmpl %%ecx,\_current\n\t" \  
    "je 1f\n\t" \  
    "movw %%dx,%1\n\t" \  
    "xchgl %%ecx,\_current\n\t" \  
    "ljmp %0\n\t" \  
    "cmpl %%ecx,\_last\_task\_used\_math\n\t" \  
    "jne 1f\n\t" \  
    "clts\n" \  
    "1:" \  
    ::"m" (\*&\_\_tmp.a),"m" (\*&\_\_tmp.b), \  
    "d" (\_TSS(n)),"c" ((long) task[n])); \  
}

大部分代码都很容易看懂，主要是：判断当前任务是否是要切换的任务，是则跳到标号1，即不做任何事；交换；调整等。。。

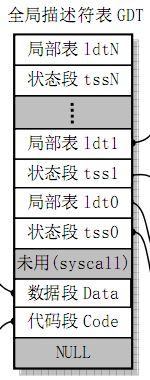
这里重点强调\_TSS(n) 和ljmp %0；

（2）\_TSS(n)，作用是生成TSS的段选择符

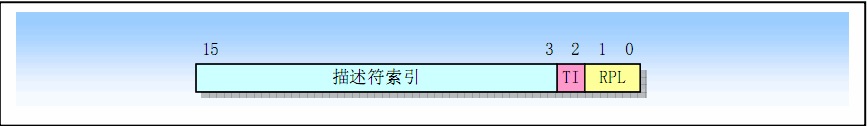
#define FIRST\_TSS\_ENTRY 4

#define \_TSS(n) ((((unsigned long) n)<<4)+(FIRST\_TSS\_ENTRY<<3))

当通过以上两个代码是不足以明白\_TSS(n)的机制，需要结合以下知识；



上图描述linux内核中GDT的布局；0-nul, 1-cs, 2-ds, 3-syscall, 4-TSS0, 5-LDT0, 6-TSS1 等。。。



上图是段选择符，TI=0表示在GDT（全局描述符表）中，1表示在LDT（局部描述符表）中，RPL表示优先级；描述符索引就是在GDT中的索引；

通过上面两张图，下面分析代码，从图1可以看出第一个TSS位于索引为4的位置，于是#define FIRST\_TSS\_ENTRY 4；而FIRST\_TSS\_ENTRY<<3表示左移3位，因为TI和RPL总共占3为；((unsigned long) n)<<4为什么要左移4位呢？从图1可以看出TSS索引都是偶数，于是TI（1位）+RPL（2位）+偶数位（1）=4；通过上述组合就可以得到TSS选择子；

（）ljmp %0或（ljmp \*%0）

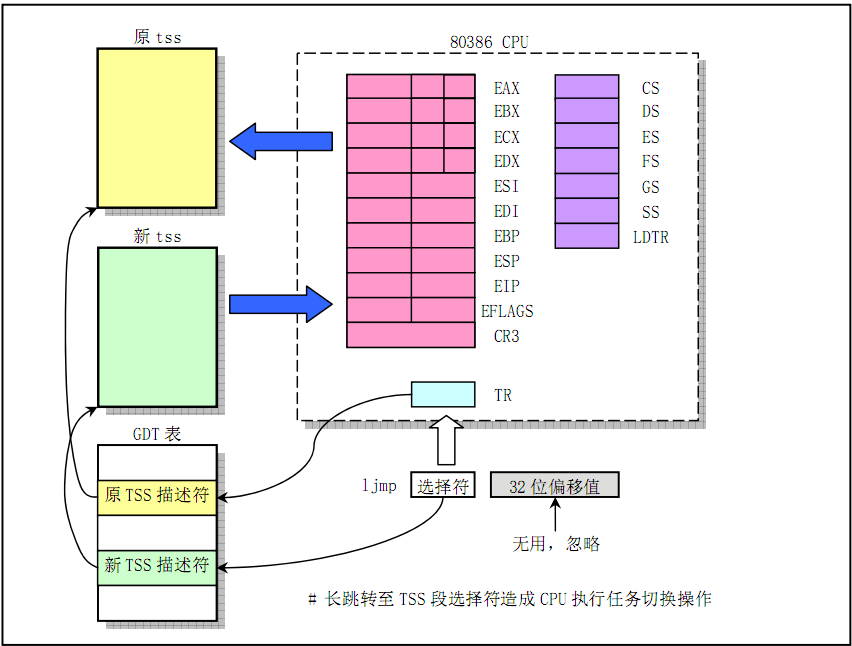
首先是为什么要加\*？这是gas语法，表示绝对跳转（与C中的\*是不同的），若程序没有加\*，则编译器会自己加上\*，可以在linux中测试；

ljmp用法说明：（很重要）

按AS手册，ljmp指令存在两种形式，即：  
   一、直接操作数跳转，此时操作数即为目标逻辑地址（选择子，偏移），即形如：ljmp $seg\_selector, $offset的方式；  
   二、使用内存操作数，这时候，AS手册规定，内存操作数必须用“\*”作前缀，即形如：ljmp \*mem48，其中内存位置mem48处存放目标逻辑地址: 高16bit存放的是seg\_selector，低32bit存放的是offset。注意：这条指令里的“\*”只是表示间接跳转的意思，与C语言里的“\*”作用完全不同。

回到源码上，ljmp %0用的ljmp的第二种用法，“ljmp \*%0”这条语句展开后相当于“ljmp \*\_\_tmp.a”，也就是跳转到地址&\_\_tmp.a中包含的48bit逻辑地址处。而按struct \_tmp的定义，这也就意味着\_\_tmp.a即为该逻辑地址的offset部分，\_\_tmp.b的低16bit为seg\_selector(高16bit无用)部分。由于在"ljmp %0"之前，"movw %%dx,%1"这条语句已经把状态段选择子"\_\_TSS(n)"的值赋给了\_\_tmp.b的低16bit。至于为什么要用\*&\_\_tmp.a，目前还不清楚，其实\*&\_\_tmp.a和\_\_tmp.a是一样的，通过汇编也可以看出；这里就先不用关心它了；

通过以上说明，可以知道了ljmp将跳转到选择子指定的地方，大致过程是，ljmp判断选择子为TSS类型，于是就告诉硬件要切换任务，硬件首先它要将当前的PC,esp,eax等现场信息保存在当前自己的TSS段描述符中,然后再将目标TSS段描述符中的pc,esp,eax的值拷贝至对应的寄存器中.当这些过程全部做完以后内核就实现了内核的切换；可以参考下图：



总结：

通过以上内容，可以大致了解到任务切换的流程，switch\_to中关键是ljmp %0；