**switch\_to()的理解**

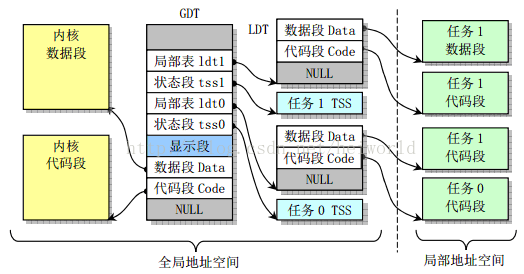
2014年05月09日 15:24:20 [heiworld](https://me.csdn.net/heiworld) 阅读数：1494

switch\_to的代码在linux-0.11\include\linux中的sched.h。它是一个宏定义，代码如下：

1. #define switch\_to(n) {\
2. struct {long a,b;} \_\_tmp; \
3. \_\_asm\_\_("cmpl %%ecx,\_current\n\t" \
4. "je 1f\n\t" \
5. "movw %%dx,%1\n\t" \
6. "xchgl %%ecx,\_current\n\t" \
7. "ljmp %0\n\t" \
8. "cmpl %%ecx,\_last\_task\_used\_math\n\t" \
9. "jne 1f\n\t" \
10. "clts\n" \
11. "1:" \
12. ::"m" (\*&\_\_tmp.a),"m" (\*&\_\_tmp.b), \
13. "d" (\_TSS(n)),"c" ((long) task[n])); \
14. }

首先讲一下末尾的

1. ::"m" (\*&\_\_tmp.a),"m" (\*&\_\_tmp.b), \
2. "d"(\_TSS(n)),"c" ((long) task[n])); \

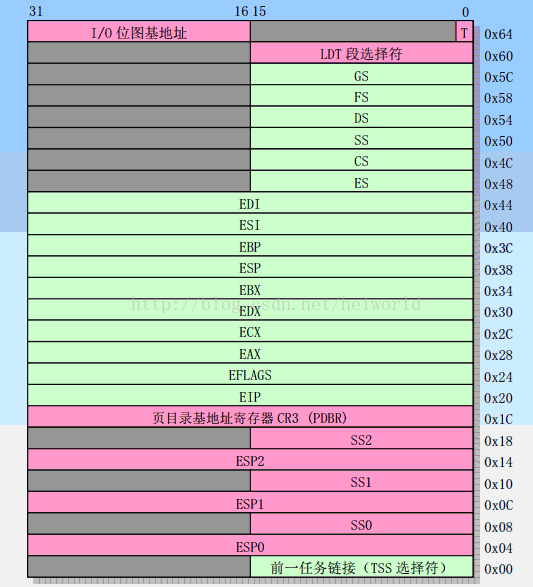
这句话吧，假如看过嵌入式汇编的应该知道这个内容，不过我还是多写点也帮助自己记忆。第一个冒号表示输出寄存器为空，第二个冒号表示输入寄存器，其中\*&\_\_tmp.a和\*&\_\_tmp.b存在任意寄存器中，m就代表任意寄存器，\_TSS(n)即该任务在GDT中的选择符，对于这个的理解需要明白GDT的构成，我截了linux内核完全注释中的一张图，如下  


再来看看\_TSS(n)这个宏定义

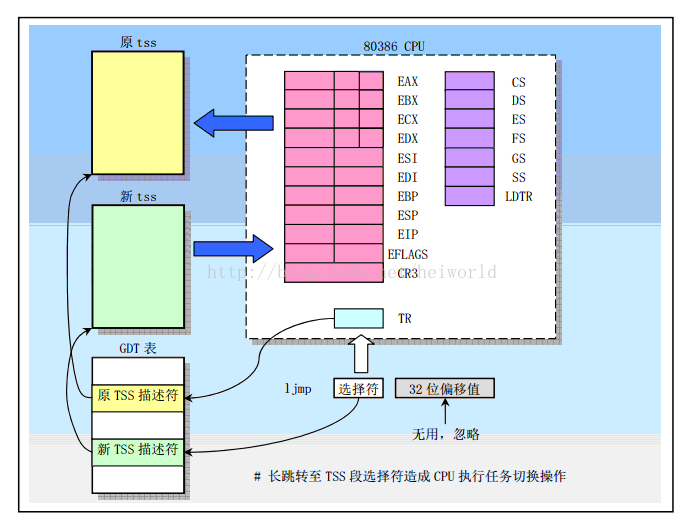
1. #define \_TSS(n) ((((unsigned long) n)<<4)+(FIRST\_TSS\_ENTRY<<3))
2. #define FIRST\_TSS\_ENTRY 4

      每个描述符占8个字节，第一个状态段是第四个，所以<<3，得到第一个任务描述符在GDT中的位置，而每个任务使用一个tss和[ldt](https://www.baidu.com/s?wd=ldt&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)，占16字节，所以<<4，两者相加得到任务n的tss在GDT中的位置。另外ecx指向要切换过去的新任务。  
     现在开始理解代码，首先声明了一个\_tmp的结构，这个结构里面包括两个long型，32位机里面long占32位，声明这个结构主要与ljmp这个长跳指令有关，这个指令有两个参数，一个参数是段选择符，另一个是[偏移地址](https://www.baidu.com/s?wd=%E5%81%8F%E7%A7%BB%E5%9C%B0%E5%9D%80&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)，所以这个\_tmp就是保存这两个参数。再比较任务n是不是当前任务，如果不是则跳转到标号1，否则交互ecx和current的内容，交换后的结果为ecx指向当前进程，current指向要切换过去的新进程，在执行长跳，%0代表输出输入寄存器列表中使用的第一个寄存器，即"m"(\*&\_\_tmp.a)，这个寄存器保存了\*&\_\_tmp.a，而\_tmp.a存放的是32位偏移，\_tmp.b存放的是新任务的tss段选择符，长跳到段选择符会造成任务切换，这个是x86的硬件原理。

TSS指向的地址里面的内容见下图



        而整个硬件切换的过程如下：



执行上述过程的操作时机主要包括：

1.      当前任务队GDT中的TSS描述符执行JMP或CALL指令；

2.      当前任务队GDT或LDT中的任务门描述符执行JMP或CALL指令；

3.      中断或异常向量执行IDT表中的任务门描述符；

4.      当EFLAG中的NT标志置位时当前任务执行IRET指令

         switch\_to中利用第一条进行切换。切换后后面的代码就不会执行了，只有等到重新切换回来的时候才会继续执行，它会判断原任务是否使用过协处理器，没有则退出，有的话要清掉cr0的的任务切换标志位TS。