1. 任务切换

1140310606 张茗帅

**①中断引发的任务切换：**

NT位，在EFLAGS寄存器中第14位，表示当前任务是否嵌套在其他任务中，1表示有嵌套，0没有。因为中断引发任务切换时，取决于旧任务是否嵌套于其他任务内。

旧任务TSS的忙状态（B位）在任务切换后同样保持不变。

新任务，B位置1，新任务NT为置1。

旧任务的TSS选择子将会填写到新任务TSS的任务链接域。

当iret指令时，必须先查询NT位，若NT位为1，进行返回，任务切换到被中断的那个任务中，同时将当前任务NT置0，TSS的B置0，保存当前任务TSS之后，用之前被中断任务的TSS恢复现场；若NT位为0，一般中断，仍在原来的任务内。

**②JMP和CALL引发的任务切换：**

JMP和CALL操作数为TSS选择子或者任务门时，可以引发任务切换。32位偏移地址忽略，从TSS中获取所有寄存器的值。（任务门描述符可以安装在中断描述符表中，也可以在GDT和LDT表中）

（1）CALL引发的中断是嵌套的，类似于中断引发：

旧任务B位仍然为1，NT位不变。

新任务B位置1，NT位置1.

旧任务TSS选择子放入新任务TSS任务链接域中。

返回时可以用iret，B和NT都置0.

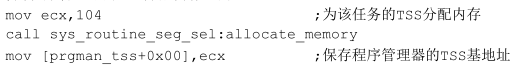
（2）JMP引发的任务切换不涉及嵌套，不主动修改NT位。

旧任务B位置0.

新任务B位置1.

任务不允许重入，处理器用B位来检测。只要任务切换，都会将B位置1.

**③主引导程序加载内核等基本与之前相同，从以下分配TSS空间开始：**



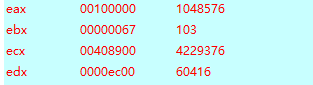
call sys\_routine\_seg\_sel:allocate\_memory函数返回值ecx为TSS基地址



将该值存到程序管理器TSS中。

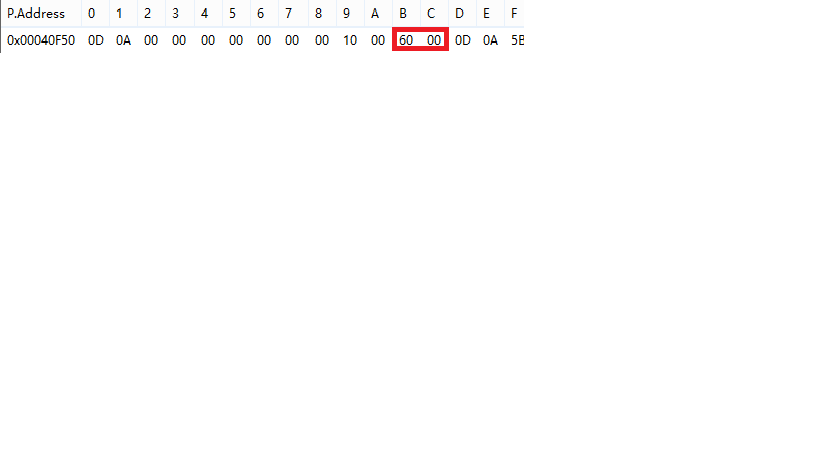


创建描述符，添加到GDT表中。





TSS选择子存到程序管理器TSS中。



加载用户程序与之前类似。

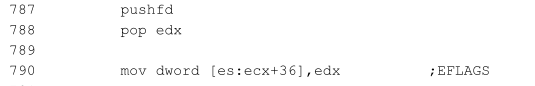
得到LDT描述符



得到TSS基地址，存在TCB中。



**787行对EFLAGS的处理：**



Pushf将EFLAGs的值压栈



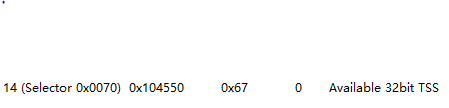
pop到Edx中



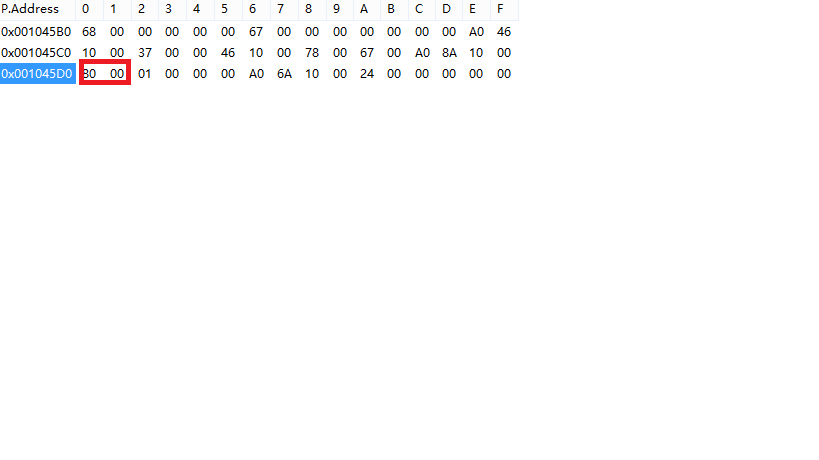
加载到TSS中：



TSS描述符加载到GDT表中



TSS选择子添加到TCB中。

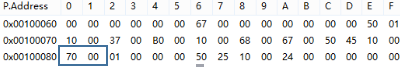


**④接下来进入核心的任务切换。**

**（1）CALL方式，**操作数是TCB的第0x14项



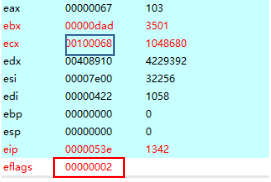
由TCB结构我们知道，那里存放的是TSS基地址，以及下一位的TSS选择子0x0070



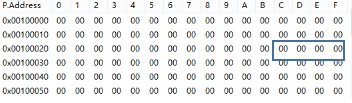
则改跳转将完成一次任务切换。我们来看看细节过程。

在GDT表中找到到0x70号描述符，得到TSS基地址为0x104550，跳转过程中，先进行了旧任务寄存器的保存，存到任务管理器TSS中，然后将新任务TSS中的值添加到各个寄存器中。

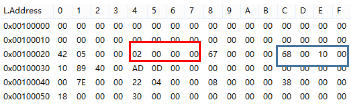
跳转前，各个寄存器的值



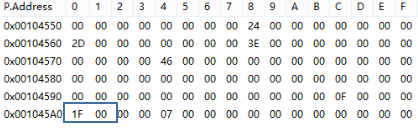
任务切换前，程序管理器TSS：



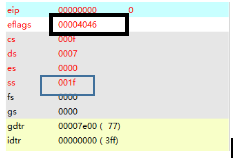
任务切换后，程序管理器TSS（可以看到，TSS中存入了切换前各个寄存器的值，图中蓝色框以ECX为例）：



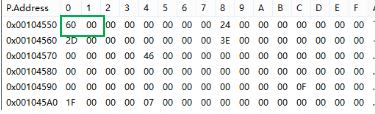
切换前，新任务（用户程序）的TSS：



切换后，部分寄存器的值（可以看到，与新任务TSS的值相对应，图中以ss举例）

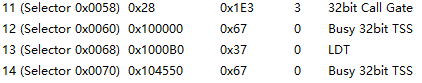


切换后，新任务的TSS：



由于我们本次采用的是CALL任务切换，所以任务切换后，程序管理器TSS描述符B位依然是1不变，标志寄存器的NT位不变（上面图中的红色框证实），新任务TSS的B位置1，NT位置1（黑框标出），任务链接域指向前一个任务TSS选择子（上图绿色框）。

GDT表中，两个任务TSS描述符都为Busy



现在我们来到了用户程序

执行完72行，计算当前CPL之后，显示：



**第74行采用调用门，将控制权转移给内核**



Fs=0x0007，查询LDT表第一项。下图，为0x 0040f310 01500327



则段基地址为0x00100150，加偏移量0x128为0x00100278



则调用门选择符为0x005B，偏移量为0x1e3（忽略无关，虽然他和最终用到的偏移量数值上相等，但是用的并不是他），在GDT中找到0x0058项即可（因为设置调用门时，将后两位置成了11以保证用户程序特权级能够使用，所以58变成了5B）

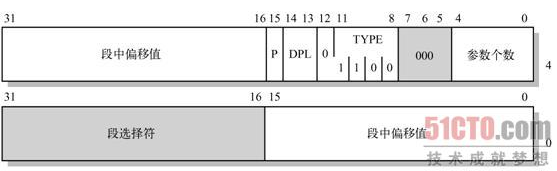
GDT表中对应的调用门描述符



在GDT对应的内存中找到的详细的调用门描述符0x0000ec00 002801e3



根据调用门描述符的结构，段选择子为0x28，偏移值为0x01e3，参数个数为0



0x28在GDT表中找到段基地址0x40018



加上偏移量0x1e3等于0x401fb



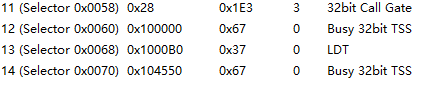
跳到了终止当前任务的公共例程，检查NT位，发现是有嵌套的，直接跳到b1（371）行，准备iret。

1. **iret后，进行任务切换**

切换到程序管理器任务，切换后旧任务（用户程序）EFLAGS中NT位应该改写成0，并写入到它的TSS中，B位应该由1改写成0.

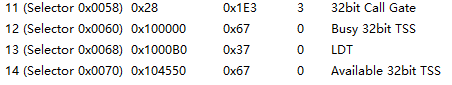
切换前





切换后



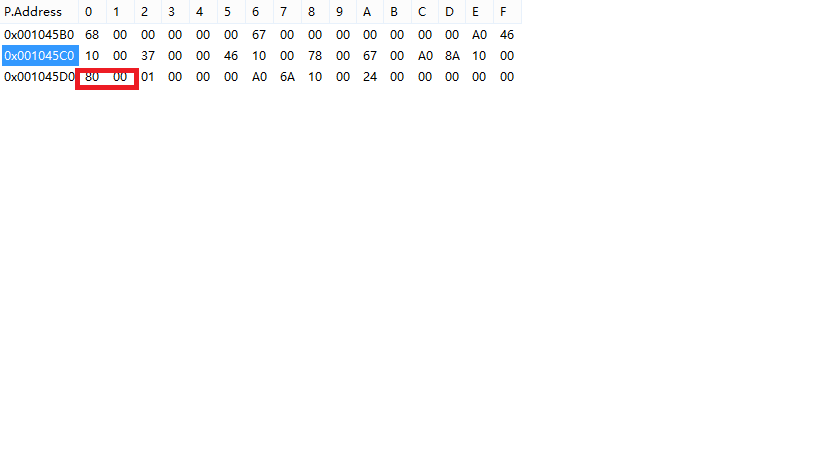


回到程序管理器任务后，从切换走的下一条开始执行。这次程序管理器再次用同样的方法加载了用户程序，在切换到用户程序的时候，这次它使用JMP的方式。

**（3）JMP方式任务切换**



由TCB结构我们知道，那里存放的是TSS基地址，以及下一位的TSS选择子0x0080



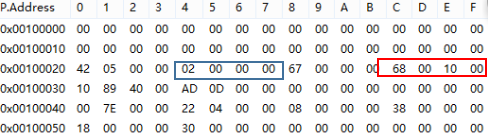
则改跳转将完成一次任务切换。我们来看看细节过程。

在GDT表中找到到0x80号描述符，得到TSS基地址为0x108aa0，跳转过程中，先进行了旧任务寄存器的保存，存到任务管理器TSS中，然后将新任务TSS中的值添加到各个寄存器中。

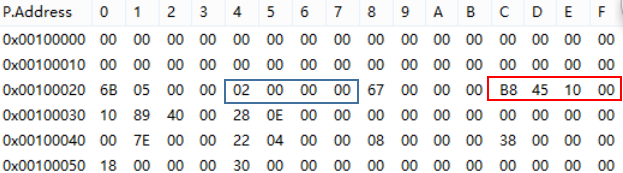
跳转前，各个寄存器的值



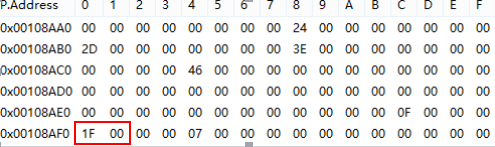
任务切换前，程序管理器TSS：



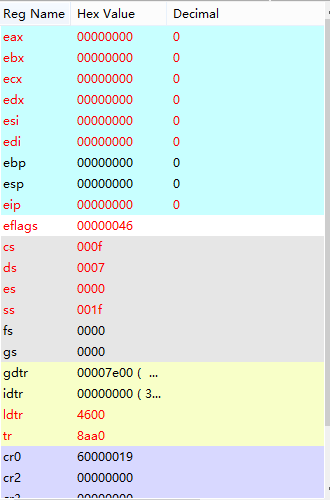
任务切换后，程序管理器TSS（可以看到，TSS中存入了切换前各个寄存器的值，图中红色框以ECX为例）：



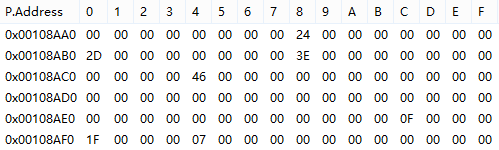
切换前，新任务（用户程序）的TSS：



切换后，部分寄存器的值（可以看到，与新任务TSS的值相对应，图中以ss举例）

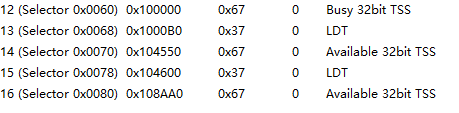


切换后，新任务的TSS：

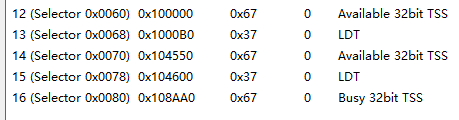


由于我们本次采用的是JMP任务切换，不涉及嵌套，所以任务切换后，程序管理器TSS描述符B位依然是1不变，标志寄存器的NT位不变（上面图中的蓝色框证实），新任务TSS的B位置1，NT位不变，任务链接域也不改变。

切换前GDT



切换后GDT，说明程序管理器（旧任务）B位置0，用户程序（新任务）B位置1.





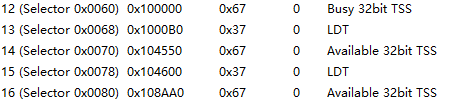
现在我们又来到了用户程序。

当再次采用调用门来到程序终止公共例程的时候，检查NT位，没有置1，没有嵌套，采用JMP方式转换到程序管理器任务。



**（4）同样是一次JMP切换，只不过是返回被切换的那个任务**

Jmp far的参数指向的是程序管理器tss基地址，下一位就是其TSS的选择子，启动任务切换，按照段选择子0x60在GDT中找到段描述符，回到程序管理器任务中。Jmp实现任务切换，所以旧任务（用户程序）的B位置0，新任务B位置1.



至此完成了全部任务切换。