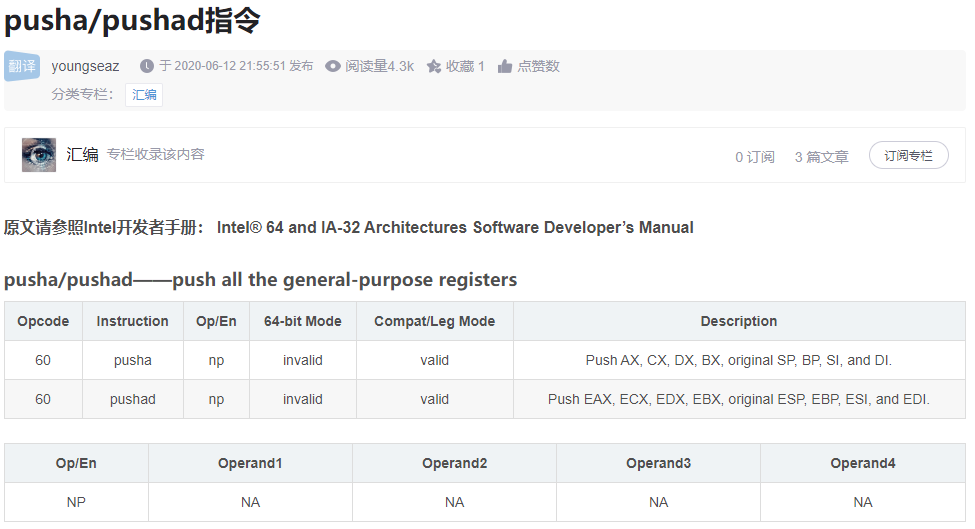
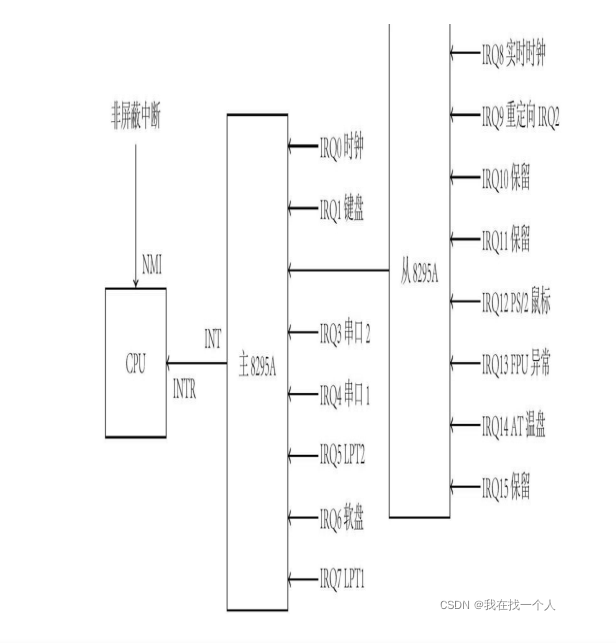
1. Pusha

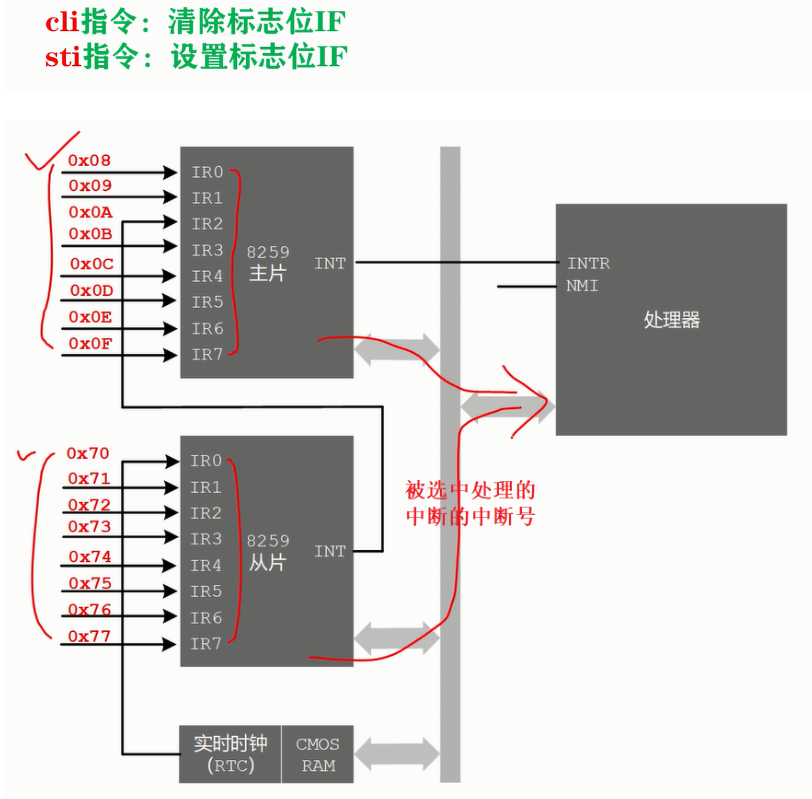
详见：https://blog.csdn.net/qq\_36963214/article/details/106717206



1. 外部中断

详见：http://lihuaxi.xjx100.cn/news/1417635.html?action=onClick

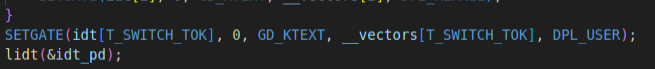


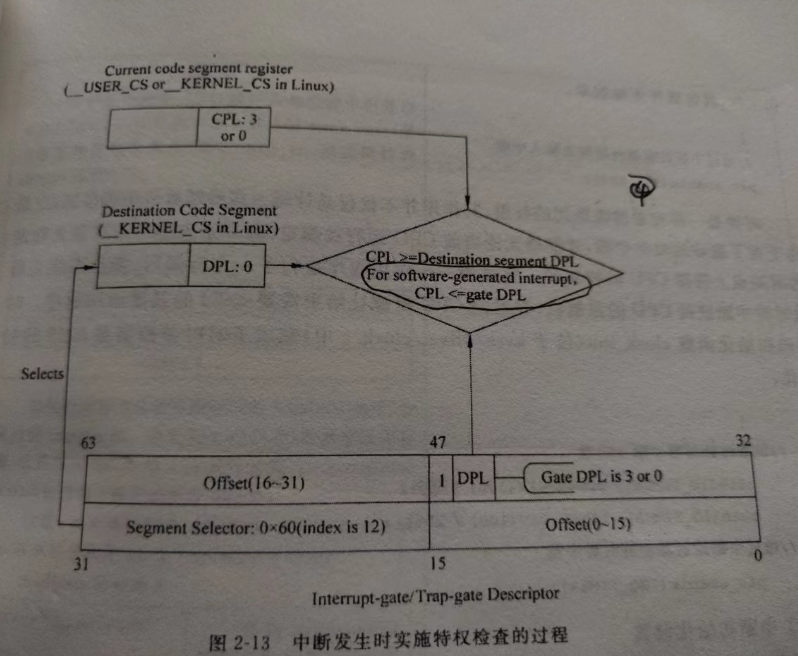


1. 内核态<->用户态之间的切换

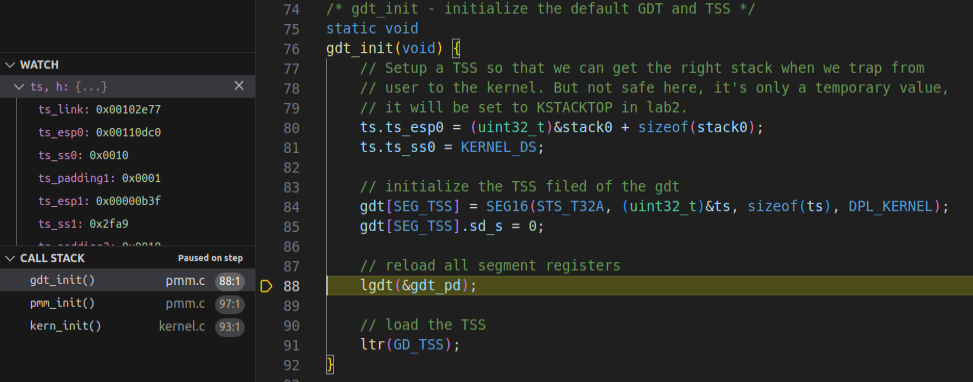
下面以当时运行的值或者说例子说明。

1. T\_SWITCH\_TOK即0x79中断的DPL被设置成DPL\_USER（Ring3），而段选择子是GD\_KTEXT内核态。根据下面的图，只有Ring3的CPL可以调用该中断，或者说该陷入门。

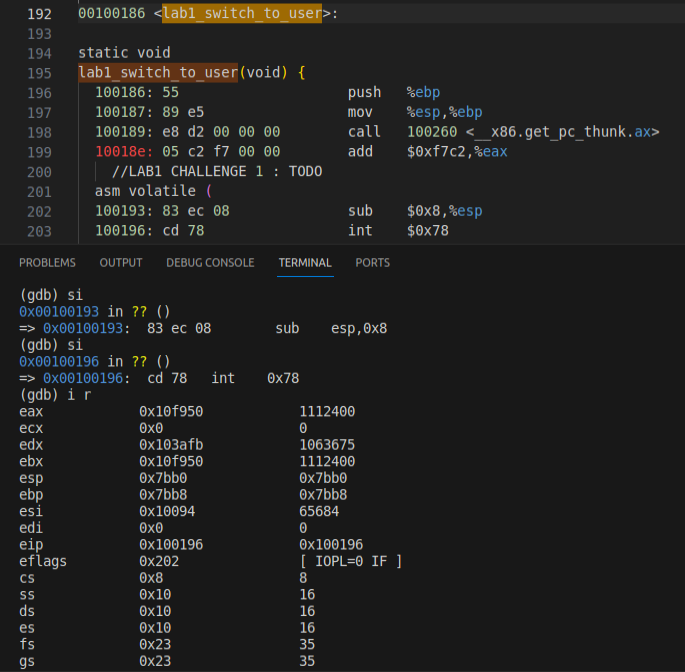




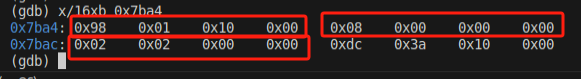
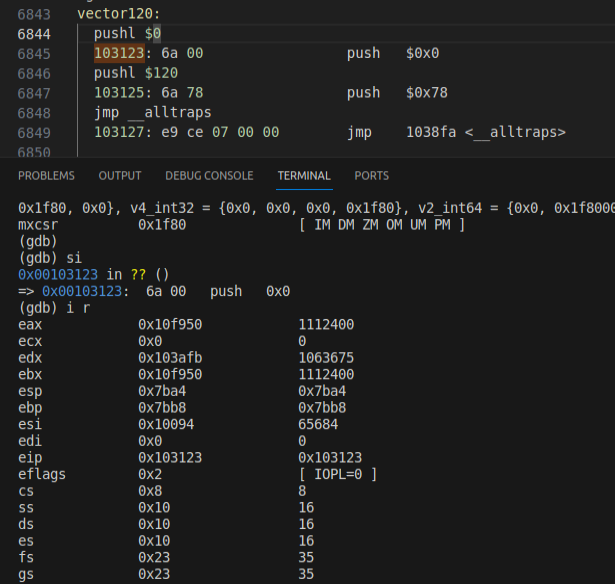
1. 在pmm\_init中，对TSS的 Ring0 的SS和SP分别做了设置，对应ESP0和SS0。这里ESP0设置为了0x00110dc0，而SS0设置成了0x0010。



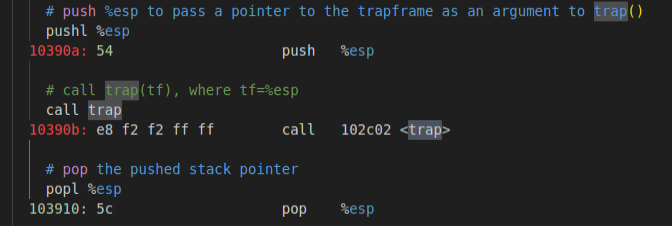
1. a). 在lab1\_switch\_to\_user里，会调用int 0x78（中断的DPL是0），在int 0x78之前，ESP = 0x7bb0，EBP = 0x7bb8。

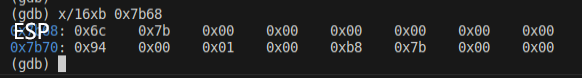


b). 执行完 INT 指令后，可以看到，ESP=0x7ba4，即压入了EF/CS/IP，并没有压入SS/ESP，因为没有发生特权切换。

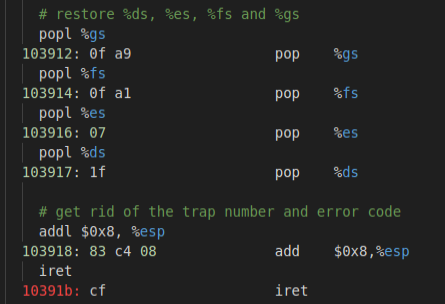


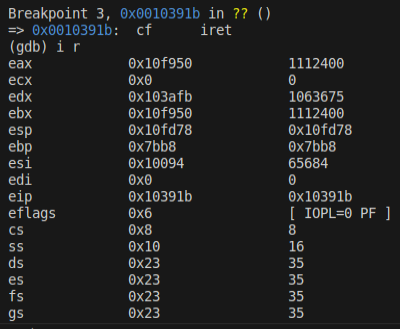
c). 这里将ESP压入栈，这样ESP+4=0x7b68。而原ESP压入栈中，即地址为0x7b68的内容为0x7b6c。然后在trap函数里改了这个0x7b6c，改成了某变量switchk2u的地址，这样在call完trap函数后，pop %esp，就会把ESP的值设置成switchk2u的地址。

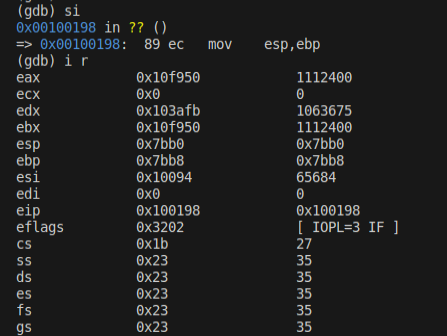




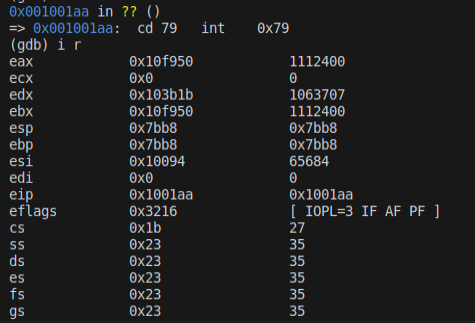
d). switchk2u的CS在trap里被置为了USER\_CS=0x1B。虽然在iret之前，通过pop设置了gs/fs/es/ds寄存器，但CS是没改的。可以看到在iret前，ESP=0x10fd78，还是个与switchk2u有关的地址。然而iret后，ESP=0x7bb0，这是因为IRET会pop CS，这时会发现有特权级切换，由内核态Ring0到用户态Ring3。在特权切换过程中，IRET会pop IP/CS/EF/ESP/SS。因为ESP/SS也得以改变。

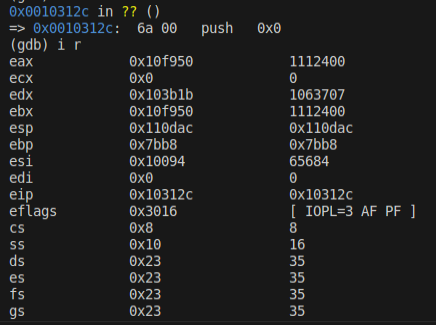


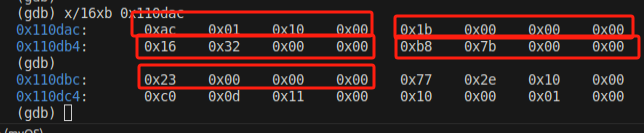




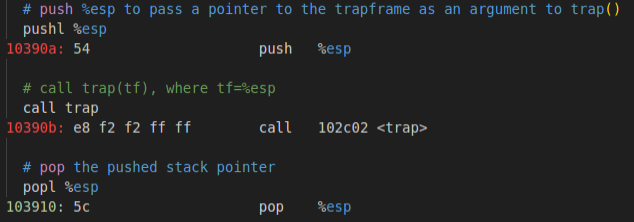
d). 在lab1\_switch\_to\_kernel中，可以看到在int 0x79执行中断前，ESP=0x7bb8，SS=0x23，CS=0x1b。一旦到中断向量，CS变成0x8，SS变成了0x10，ESP变成了0x110dac，即1中TSS里设置的值。 这是因为当前CPL为Ring3用户态，而0x79中断的选择子对应的段是Ring0的内核态，这里需要发生特权切换切换到GD\_KTEXT内核态。这里发生了权限切换，所以堆栈按TSS切了。这里的INT。SS/ESP/EF/CS/IP先后入TSS指定的栈。进入内核态。







e). push %esp，在trap里改%%esp，导致pop %esp时ESP改为想要的值switchu2k，同c)。TSS指定的栈会把寄存器都压进去，在trap中，switchu2k复制了TSS的指定的栈，但不复制ESP0/SS0部分，并且让switchu2k的地址正好是，switchu2k的ESP0/SS0对应INT前的栈顶的值，即进lab1\_switch\_to\_kernel压入的EBP=0x7bc8。



f). 在IRET时，注意这里和d)不一样，这里CS已经是0x8，Ring0，内核态，不涉及特权切换，因此只要pop IP/CS/EF，而不用pop SP/SS，因此不会把SP置到0x7bc8去。

h). 总之，当发生特权切换时，int/iret需要push/pop SS/ESP，否则不用。