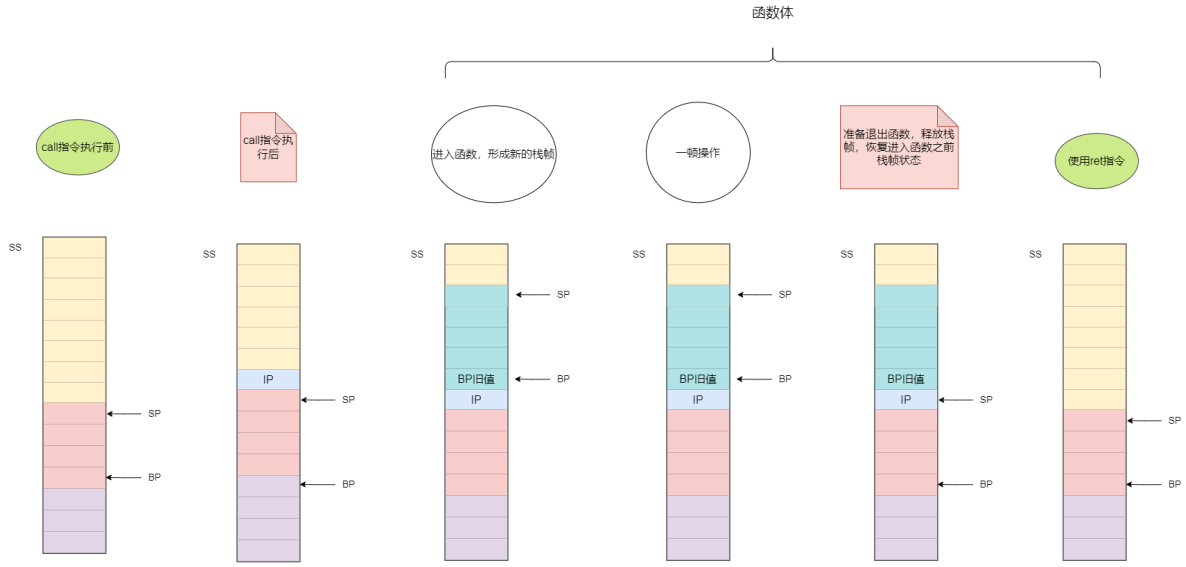
第4节 函数（模块化程序设计）



说在前面

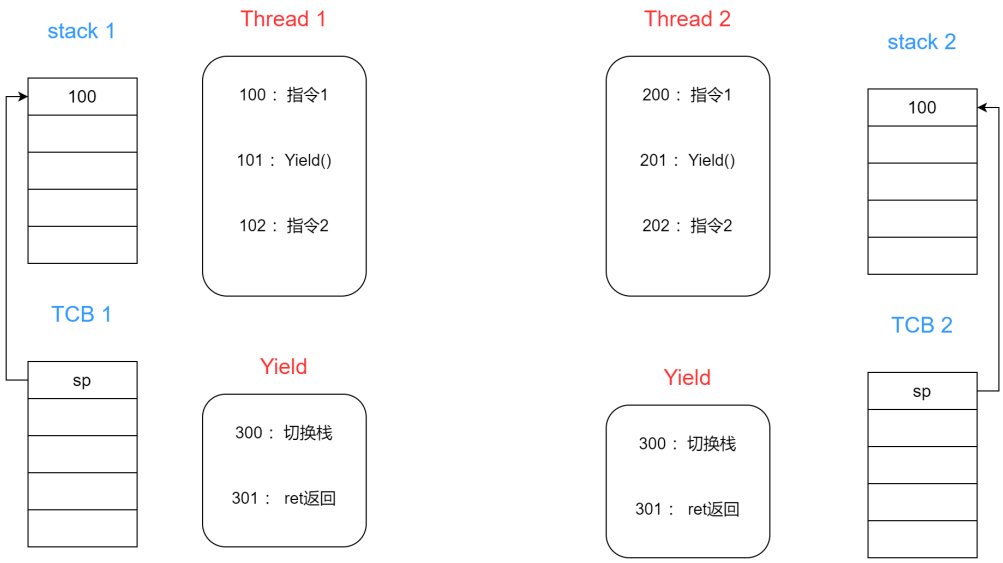
这节的目的一个是为了了解函数背后运行的机制，包括局部变量是什么，如何传参，为什么函数退出后局部变量就没了，传值和传地址有什么区别，在汇编指令级别上理解这些事情。

另一个重要的事情是了解从一个指令序列切换到另一个指令序列的方式，除了这里利用栈来实现指令序列切换，还可以利用相关数据结构实现指令序列切换。这两种指令切换在后面的操作系统课程中是非常非常非常重要，如何实现用户态和内核态直接的切换，如何实现进程之间的切换，如何实现线程之间的切换，基本都是本节提到的两种切换思想。

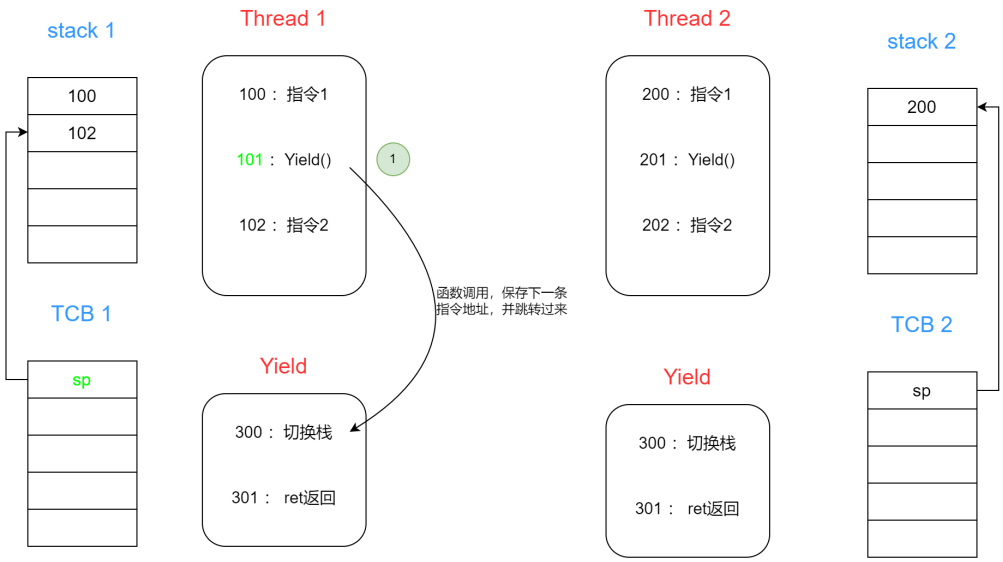
1. 利用栈实现指令序列切换

简单来说就是进程或者线程1在切换到线程2时，把记录点信息全部压入栈中，然后到了线程2后想切回来时，先把栈切过来，然后把栈里面的内容弹出到cpu中即可

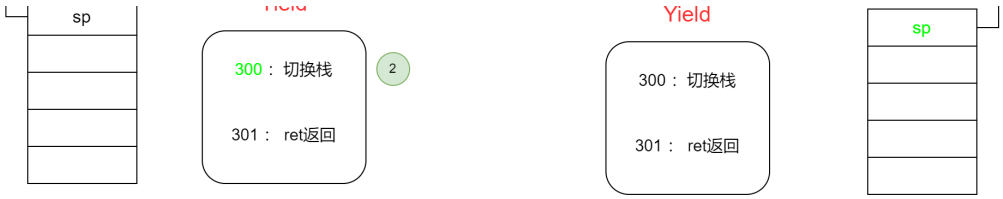
用户级线程切换原理



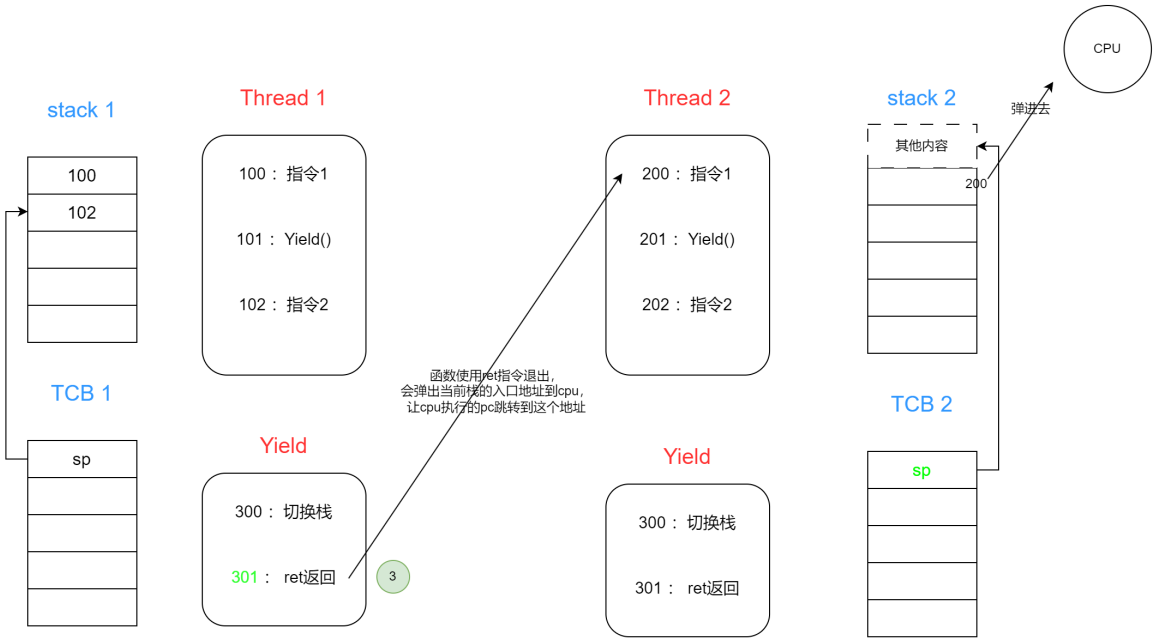
初始：为每个线程建立1个TCB和1个栈，栈初始只有一个元素，就是线程的起始地址
TCB就是描述线程的cpu状态信息，主要是当前进程执行到哪里了，也就是执行入口地址。
其他东西我们这里暂时并不关心，因此上面图中栈里面只记录入口地址。



第1步：刚执行完101行的时候，会自动把102压入栈中，注意此时的栈被标绿了



第2步：刚执行完300行的时候，也就是将当前工作的栈切换到线程2工作的栈，注意sp颜色变化



第3步：刚执行完301行的时候，会将当前栈的pc（或者说ip，程序计数器）弹出，由于当前栈已经被切换了，所以弹出的pc是200，下一次cpu会从入口地址为200的地方执行

第4步：刚执行完201行的时候，将入口地址202压入栈中，并跳到Yield的地方第300行执行

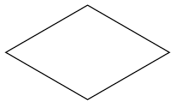
从第5步开始，后面就不画图了，和前面类似

第5步：刚执行完300行的时候，当前栈切换回stack1

第6步：刚执行完301行的时候，会将stack1的pc弹出给cpu，也就是102，表明马上就要从102

行开始执行

第7步：这步就是从102行开始执行，回顾前面，整个流程顺利完成了切走再切回来，
就是先执行100行，然后从101行切走，然后顺利切回102行，对于线程1来说，
成功完成了切走再切回来，上面这种通过切换栈的工作模式，就是线程切换的核心思想



2. 利用数据结构实现指令序列切换

简单来说，就是把记录点信息保存到tss结构体中，想切回来时，直接利用ljmp指令，硬件实现切回来。

注意，这个ljmp指令需要配合80386cpu的TR寄存器，和tss结构体一起才能发挥作用，这个是早期切换方式，

速度特别慢，现在的os都不用这种切换方式了。