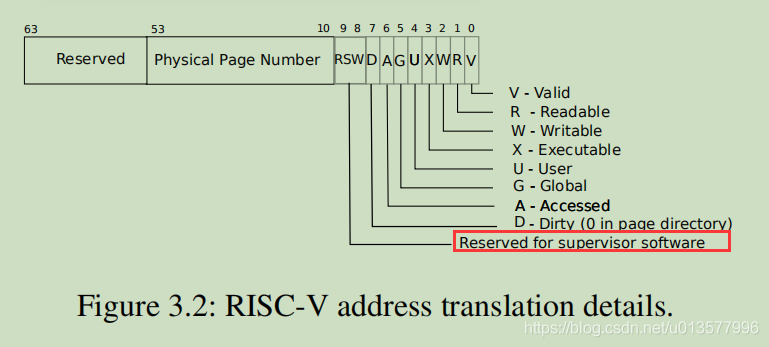
# Lab5实验报告

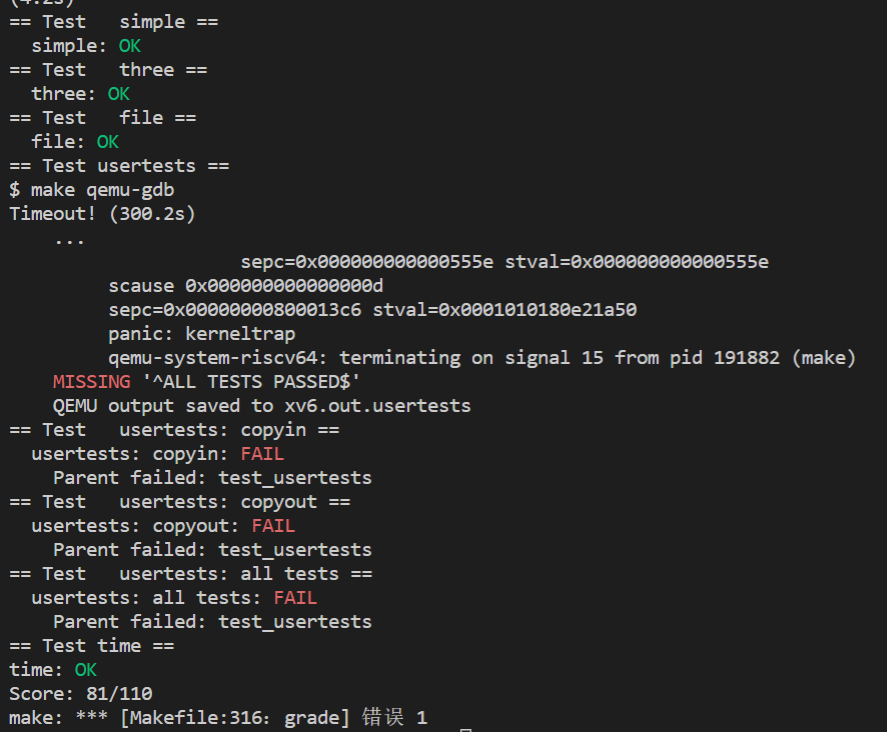
### 第一部分

这次的实验断断续续做了好久，还是没法通过所有测试，所以把我做实验的具体步骤简单说一下：

1. 修改kernel/vm.c，新增int refNum[32768];来记录关联物理页的页表数量。32768是根据(PHYSTOP-KERNBASE)/PGSIZE得出。
2. 修改kernel/riscv.h，新增PTE\_COW标志位，参考riscv对PTE标志位定义，第9-10位为预留标志位。



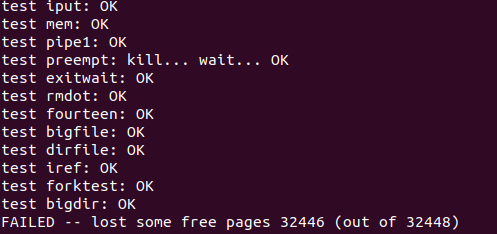
1. 修改kernel/vm.c的uvmcopy()，让进程fork时，不赋值物理页，而是child进程页表指向parent进程的物理页，但标记要将parent和child的PTE都清除PTE\_W标志位，并添加COW标志位，表明两个PTE指向一个物理页。
2. 修改kernel/vm.c的mappages()，在页表与物理页绑定时，增加refNum对应元素计数。
3. 修改kernel/vm.c的uvmunmap()，在页表与物理页解绑时，减少refNum对应元素计数，当refNum==1即仅kernel pagetable持有时，释放内存。
4. 将kernel/vm.c中walk()定义在defs.h中。
5. 修改kernel/trap.c的usertrap()，引入refNum，在发生page fault时，若该虚拟地址关联的PTE，表明关联的物理页是一个COW页，则新申请一个物理页，让此虚拟地址指向新物理页，并修改refNum计数。
6. 修改kernel/vm.c的copyout()，同kernel/trap.c的usertrap。

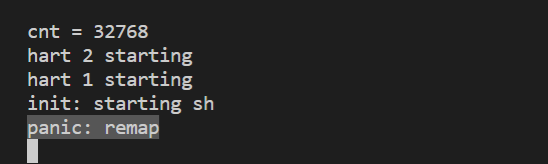
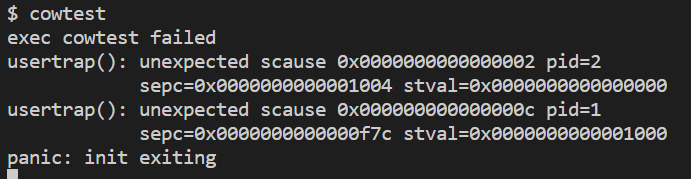


### 第二部分

*遇到的问题*

* 有一些内存块没有回收





*解决方案*

* 设计带锁的结构体来保存对物理内存的引用数

用时三天完成。

### 第三部分

* 这次的实验加强了我对于异常处理的熟练程度。
* 让我发觉自己对操作系统的知识还不够了解，准备再看看手册，看看能不能解决这次没解决的问题。