Lab2 实验报告

第一部分

第一个实验比较容易，直接根据提示用递归的方式显示多级页表，到第二个实验就变得有些麻烦了。在拷贝内核页表部分与用户表部分卡了很久，之后才明白每个进程的独立内核页表应该包含内核与用户两个部分，kernel/vm.c 添加新版本的 kvminit 用于为每个进程生成内核页表，其中没有映射 CLINT，因为用户地址空间为低于 PLIC 的部分，而 CLINT 也是低于 PLIC 的，在第三个实验中，在 fork、exec、sys_sbrk 中将用户页表的内容复制到内核页表。其余的步骤在学生反复调试之后成功通过了测试。

```
== Test pte printout ==
$ make qemu-gdb
pte printout: OK (1.7s)
== Test answers-pgtbl.txt == answers-pgtbl.txt: OK
== Test count copyin ==
$ make qemu-gdb
count copyin: OK (1.1s)
== Test usertests ==
$ make qemu-gdb
(87.7s)
== Test   usertests: copyin ==
  usertests: copyin: OK
== Test   usertests: copyinstr1 ==
  usertests: copyinstr1: OK
== Test   usertests: copyinstr2 ==
  usertests: copyinstr2: OK
== Test   usertests: copyinstr3 ==
  usertests: copyinstr3: OK
== Test   usertests: sbrkmuch ==
  usertests: sbrkmuch: OK
== Test   usertests: all tests ==
  usertests: all tests: OK
== Test time ==
time: OK
Score: 66/66
chy@k8s-master:~/lab/xv6-labs-2020$
```

第二部分

遇到的问题

```
xv6 kernel is booting

hart 1 starting
hart 2 starting
panic: kvmpa
```

- 如何打印多级页表。
- 如何判断此时使用的是用户表还是内核表。

解决方案

- kvmpa()方法会在进程执行期间调用，此时需要修改为获取进程内核页表，而不是全局内核页表。
- 用递归的方式遍历多级页表。
- 看 PTE_U 来判断。

用时两天。

第三部分

- 让我对虚拟页式分配的理解有了加深。
- 为了提升用户和内核之间的复制效率。如果不复制的话每次需要调用 walk 将用户虚拟地址转为物理地址，效率低；现在是由硬件 mmu 执行 walk 函数效率高，让内核程序员更轻松。
- 内核页表里有物理地址的等值映射，cpu 还是会将 walk 得到的物理地址当成虚拟地址，通过 mmu 访问物理地址