Lab2 补充指导书

在阅读本指导书前，请确保你已经仔细读过了pmap.c， pmap.h， mmu.h这三个文件的代码。与CO、OO等课程不同，OS实验课程要求大家读的代码比写的代码更多，在自己动手写代码的时候会用到已经给出的的函数或宏，如果只是关注要填写的exercise部分的代码，必然会无从下手。

关于虚拟内存和物理内存

操作系统是运行在虚拟地址空间还是物理地址空间的？我们的操作系统是完全运行在虚拟地址空间中的，无论是取指令还是取数据，CPU所操作的地址都是虚拟地址。尽管将虚拟地址转换为物理地址的过程不需要操作系统完成，但是操作系统需要**建立虚拟地址与物理地址之间的映射关系。**在MIPS体系结构中，如果我们使用的地址在kseg0(0x80000000-0x9fffffff)区域，只需要将最高位置零就可以得到物理地址，因此操作系统此时不需要再单独为这一部分填写页表。可是，如果我们使用的地址位于kuseg或是kseg2，就必须填写页表来实现虚拟地址与物理地址的映射。

关于Page结构体

当我们发现某一个虚拟页面没有对应的物理页面与之形成映射时，就需要找到一块空闲的物理页面完成这样的映射。问题在于，用什么方法来找到一块空闲的物理页面呢？一个物理页面，可能有多个虚拟页面映射到这里，我们又怎么知道有多少个虚拟页面映射到这里了呢？在MOS操作系统中，我们建立了Page结构体。并且将代表空闲页面的Page结构体用链表组织起来，这样，当想找空闲物理页面时，我们只需要从这个链表中找一块就可以了。

那么，pages数组的**地址**和相应物理页面的地址有什么关系呢？一点关系也没有！他们唯一的关系是，pages[0]代表着第0号物理页面，有这种序号上的对应关系罢了。那么怎么实现page和对应物理页面的起始地址的转换呢？

> 在阅读本指导书前，请确保你已经仔细读过了pmap.c， pmap.h， mmu.h这三个文件的代码。：）

关于页表与页目录

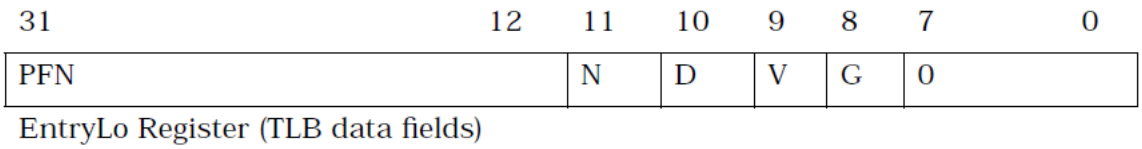
笔者认为这一部分在指导书中的图片已经相当清楚地展示了如何查找页目录和页表了。可能唯一的问题在于，页目录在哪里？在`mips_init_vm()`中，我们为页目录申请了一块页面

```
/* Step 1: Allocate a page for page directory(first level page table). */
    pgdir = alloc(BY2PG, BY2PG, 1);
```

你可以打印这个地址，看看它具体的位置。页目录在申请后还是空的，也没有二级页表。我们怎么让这个页表变得充实起来呢？在MOS中，主要是通过`boot_pgdir_walk`与`pgdir_walk`这两个函数填充页表。这两个函数的目标其实只是为了获得`va`所对应的二级页表的页表项而已。有的时候我们希望在二级页表不存在的情况下创建相应的二级页表，这时就应该申请一页新的物理页面作为二级页表。

关于页表项

一个页表项大概长这样：



EntryLo Register (TLB data fields)

Figure 6.2. EntryHi and EntryLo register fields

其中，0-7位硬件没有给出定义，但是操作系统作为软件可以自定义一些权限位，我们可以暂时不管它。

其他的N, D, V, G，分别代表MOS中的PTE_UC, PTE_R, PTE_V, PTE_G，他们的定义，在此摘录《R3000手册》的相关内容：

> • N: “noncacheable”; 0 to make the access cacheable, 1 for uncacheable.
> • D: “dirty”, but really a write-enable bit. 1 to allow writes, 0 and any store using this translation will be trapped.
> • V: “valid”, if 0 any address matching this entry will cause an exception.
> • G: “global”. When the G bit in a TLB entry is set, that TLB entry will match solely on the VPN field, regardless of whether the TLB entry’s ASID field matches the value in EntryHi.

这份补充指导书可能来得有点晚了，不过还是希望可以帮到大家。如果有错误，烦请您联系我指正。

AUTHOR : 王宇康

wechat：wyk3703