Lab2实验报告

150616-14051131-陈登博

思考题

1. Thinking 2.1
   这种写法的好处是以后可以通过修改`while()`括号中的条件方便地改变循环的次数，使得程序具有更好的可扩展性。

2. Thinking 2.2
   物理内存页具体在0x80000000到0x84000000这64MB空间中，4KB为一页，共计0x4000页。对于一个Page结构体p，我们可以通过`page2kva(&p)`获取它所对应的物理内存页的虚拟地址。观察page2kva函数的实现可知，首先通过`(&p)-pages`得到该Page页的索引，然后将其向左移动12位，就得到了该Page页对应物理页的物理地址，再将该物理地址加ULIM，即得到了相应的虚拟地址。

3. Thinking 2.3
   这个b指针是一个虚拟地址。从bzero函数的实现即可看出，其中使用了`*(int*)b=0`等语句。而在整个实验中，物理地址都是需要转换为虚拟地址(+ULIM)之后才可以寻址的，因此b指针是一个虚拟地址。

4. Thinking 2.4
   其页目录的起始地址为0xC0000000+(0xC0000000>>10) = 0xc0300000。

5. Thinking 2.5

   tlb_out函数跳转到NOFOUND的流程为

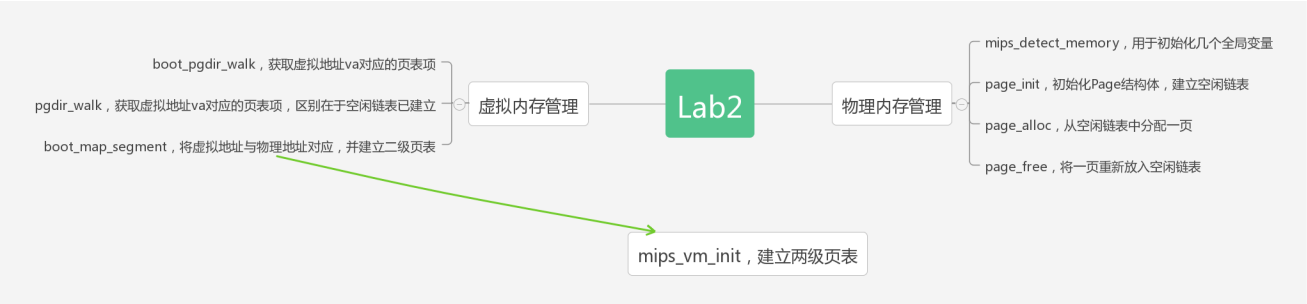
```
mfc0 k1,CP0_ENTRYHI     #先从CP0_ENTRYHI寄存器中取出值保存在k1寄存器中
mtc0 a0,CP0_ENTRYHI     #再把参数a0寄存器中保存的值存入CP0_ENTRYHI寄存器
nop
tlbp                    #然后在TLB的表项中查找是否有与CP0_ENTRYHI寄存器匹配的，若有则把匹配项Index保
                        #存到CP0_INDEX寄存器中,没有匹配则置CP0_INDEX的最高位为1
...
mfc0 k0,CP0_INDEX       #取出CP0_INDEX寄存器
bltz k0,NOFOUND         #检测CP0_INDEX第一位是否为1(是负数就为1)，为1则跳转到NOFOUND
nop
...
NOFOUND:
mtc0 k1,CP0_ENTRYHI     #没找到匹配项，将刚刚保存的ENTRYHI寄存器的值重新赋给它
j ra                    #函数返回
nop
```

6. Thinking 2.6
   这是因为我们先执行`page_insert`函数，其中会调用`tlb_invalidate`函数，而`tlb_invalidate`函数又会去

调用`tlb_out`函数，也就是说在插入页的时候我们就将va对应的页项从tlb中删除了，所以在向*va处写入数据的时候，我们会触发tlb miss，引起异常，没有重填机制的lab2就会在这里陷入死循环。

实验难点

在本次实验中，最难的莫过于理解整个页表的分配和管理机制了。示意图如下



首先让人感到困惑的是，什么是pa，什么是kva，二者的关系是什么。通过查阅宏函数对这两种地址的转换可知，pa是kva-0x80000000得到的。因此通过联系mips_detect_memory函数可知，这个pa是物理地址，kva是内核虚拟地址，由于物理空间只有从0x80000000-0x84000000的64MB，因此将内核虚拟地址转换为物理地址只需要减去一个0x80000000即可。

其次的难点在于pgdir_walk和boot_pgdir_walk，从注释的意思可以看出这两个函数都是要获取虚拟地址对应的页表项，同时兼具创建页表的功能。一开始这两个函数让我感到很困惑，在了解了二级页表的机制之后才慢慢明白其含义。因为这两个函数都会给定页目录地址，因此我们通过PDX(va)和页目录地址的配合就能找到页目录项，也就是对应页表的首地址，再通过PTX(va)就能找到对应的页表项，这也符合我们二级页表寻址过程。而create功能无非是其中一个点缀，其作用是创建一个页表，然后将地址传回到页目录项中，这就建立起页目录中项和一个页表的联系，这显然与我们的二级页表思想相吻合。而boot_pgdir_walk和pgdir_walk的区别在于空闲链表是否被初始化，从boot的名字上也可以看出是在启动时分配页表，因此空闲链表还未被初始化。明白了二者的区别，就可以发现二者其实挺相似的，因此解决了一个另一个也变得容易了。

而lab2还有一个难点在于页目录的自映射。这可以用数学中的不动点定理加一证明，简而言之，就是页目录跟它所指向的一个页表重合了，因此产生了只要4MB空间即可存储页目录和页表的现象。理解了自映射机制，那么可以得出页目录起始地址计算公式。

$$PGDIR_{init\_addr} = PGTABLE_{init\_addr} + PGTABLE_{init\_addr} >> 10$$

体会和感想

在lab2中我大量的时间花在了理解二级页表机制上，其实真正需要完成的函数不多，而且重要的page_insert函数也没有要求我们实现。从lab2的经历中，我发现阅读那些已经为我们写好的宏函数是非常有用的。因为它们透露出设计者的意图或者整个工程的思想，这对于完成整个任务是很有帮助的。此外，善用宏函数能够让我们需要写的代码非常简洁。

在lab2中，仍未解决的问题就是TLB重填函数的未实现问题，这导致了无法正确再现之前思考题2.6所提到的现象，程序会在TLB缺失之后陷入死循环。因此我对TLB重填函数的实现机制也不甚了解，这些可能都得在lab3或者更以后的实验中才能解决。