操作系统 Lab2 实验报告

17373452 单彦博

一． 思考题

1. **请思考 cache 用虚拟地址来查询的可能性，并且给出这种方式对访存带来的好处和坏处。另外，你能否能根据前一个问题的解答来得出用物理地址来查询的优势**

可以将 cache 改为用虚拟地址来查询，这种方式相当于把查询 cache 放在 TLB 和页表前面。好处是：如果 cache 命中，则访存速度会极大地加快；坏处是：如果 cache 不命中，则访存速度会比之前还要慢，因为这种情况是必然会进行主存访问的。用物理地址来查询，优势是访存时间相对稳定，既不会在 cache 命中的时候过快，也不会在 cache 未命中的时候过慢。

2. **请查阅相关资料，针对我们提出的疑问，给出一个上述流程的优化版本，新的版本需要有更快的访存效率。（提示：考虑并行执行某些步骤）**

将 cache 改为用虚拟地址查询，在 cache 查询的同时通过 MMU 将虚拟地址转化为物理地址，可以有效加快方寸效率。

3. **在我们的实验中，有许多对虚拟地址或者物理地址操作的宏函数(详见 include/mmu.h),那么我们在调用这些宏的时候需要弄清楚需要操作的地址是物理地址还是虚拟地址，阅读下面的代码，指出 x 是一个物理地址还是虚拟地址**

```
int x;
char *value = return_a_pointer();
*value = 10;
x = (int) value;
```

x 是虚拟地址。

4. **我们注意到我们把宏函数的函数体写成了 |do /* ... */ while(0)| 的形式，而不是仅仅写成形如 | /* ... */ | 的语句块，这样的写法好处是什么?**

将宏函数变成一个整体，防止引用的时候产生逻辑错误。同时，使用宏函数时，一般程序员的习惯会在行尾加分号，用{//……},b 编译就会报错，而使用 do { // ... } while(0)，则不会。

5. **注意，我们定义的 Page 结构体只是一个信息的载体，它只代表了相应物理内存页的信息，它本身并不是物理内存页。 那我们的物理内存页究竟在哪呢？Page 结构体又是通过怎样的方式找到它代表的物理内存页的地址呢？ 请你阅读 include/pmap.h 与 mm/pmap.c 中相关代码，给出你的想法**

pmap.c 中的 mips_vm_init 给物理内存页分派了虚拟地址，在通过虚拟地址与物理地址的转换实现物理内存页的存储；Pages 则是通过 pmap.h 中的 page2pa 函数指向实际的物理内存页。

6. **请阅读 include/queue.h 以及 include/pmap.h, 将 Page_list 的结构梳理清楚,选择正确的展开结构(请注意指针)**

答案是 C。

```
struct Page_list{
    struct {
        struct {
            struct Page *le_next;
            struct Page **le_prev;
        } pp_link;
        u_short pp_ref;
    }* lh_first;
}
```

7. **在 mmu.h 中定义了|bzero(void *b, size_t)|这样一个函数,请你思考，此处的 b 指针是一个物理地址， 还是一个虚拟地址呢?**

   b 是个虚拟地址。

8. **了解了二级页表页目录自映射的原理之后，我们知道，Win2k 内核的虚存管理也是采用了二级页表的形式，其页表所占的 4M 空间对应的虚存起始地址为 0xC0000000，那么，它的页目录的起始地址是多少呢?**

   0xC0000000 + （0xC0000000 >> 12） << 2 = 0xC0300000

9. **思考一下 tlb_out 汇编函数，结合代码阐述一下跳转到 NOFOUND 的流程?从 MIPS 手册中查找 tlbp 和 tlbwi 指令，明确其用途，并解释为何第 10 行处指令后有 4 条 nop 指令。**

   NOFOUND 是在 TLB 入口没有找到时所返回的错误信息，即在 tlb_out 汇编函数中，我们首先需要寻找 TLB 入口，没有找到就会返回 NOFOUND。

   在 MIPS 手册中，两条指令的定义如下。

   tlbp:To find a matching entry in the TLB.即寻找 TLB 中对应的页表项。

   tlbwi:To write a TLB entry indexed by the Index register.即向 TLB 中写入一个页表项信息。

   4 行 nop 是由于在 CPU 没有转发机制时，需要保证其中一条指令执行完毕后才能开始下一条指令的执行，所以需要 4 个 nop 保证指令操作的正确性。

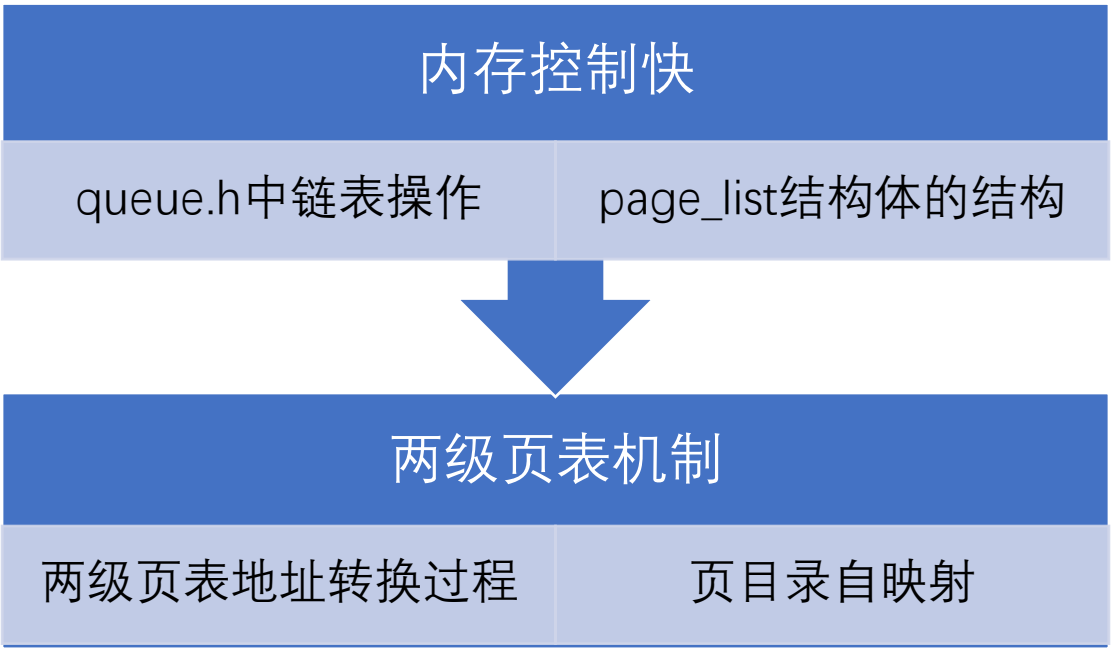
10. **显然，运行后结果与我们预期的不符，va 值为 0x88888，相应的 pa 中的值为 0。这说明我们的代码中存在问题，请你仔细思考我们的访存模型，指出问题所在。**

    虚拟地址更新后的数据进入缓冲区，但是由于物理地址位于 kuseg，且在 Start.S 中 kernel mode cache 被禁止了，所以出现了错误。

11. **在 X86 体系结构下的操作系统，有一个特殊的寄存器 CR4，在其中有一个 PSE 位，当该位设为 1 时将开启 4MB 大物理页面模式，请查阅相关资料，说明当 PSE 开启时的页表组织形式与我们当前的页表组织形式的区别。**

CR4 的 PSE 位在置 1 时，会指向一个 4M 的页面，而不仅仅指向一个 4K 的页面，这样的好处是由于页面的变大，所需要使用的页表项也就可以减小，减少占用的空间。

二． 实验难点

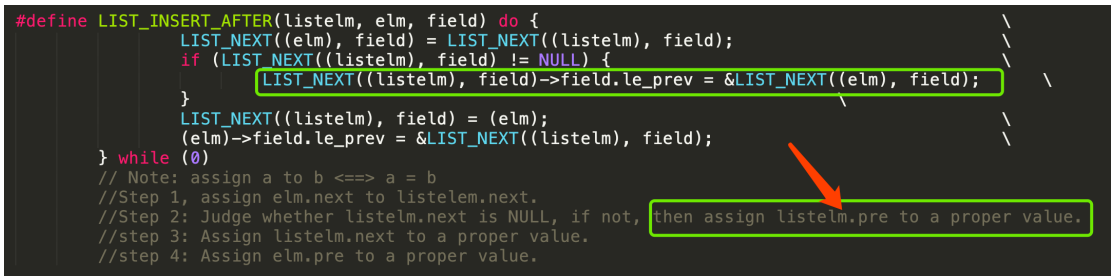


三． 体会与感想

本次实验难度比较大，主要表现是对 Page_list 结构体和链表操作的理解上，我在理解这几个东西的时候花了非常多的时间。难度其次的是对两级页表机制，特别是对自映射过程的理解，这个地方我在理解的时候，结合了理论课的讲述，更明白了两级页表查询的实现过程。本次实验我花费的时间大约在 10 小时左右，因为难度较大而且比较重要，需要深入理解的地方也很多。但是，个人认为我自己对实验代码的理解还不够到位，在接下来的学习过程中，我会不断强化自己的理解，花更多的时间去阅读代码，做到理解透彻，灵活运用。

四． 指导书反馈

1. 在指导书第 76 页，“从虚拟地址到物理地址的转换只需要清掉最高位的零即可”，我认为其中的“零”应该是“一”。
2. 对于 queue.h 实验代码中，我认为 LIST_INSERT_AFTER 的代码提示有误。



对于箭头所指的位置，按同学们的理解，应该是 listelm.next.pre。