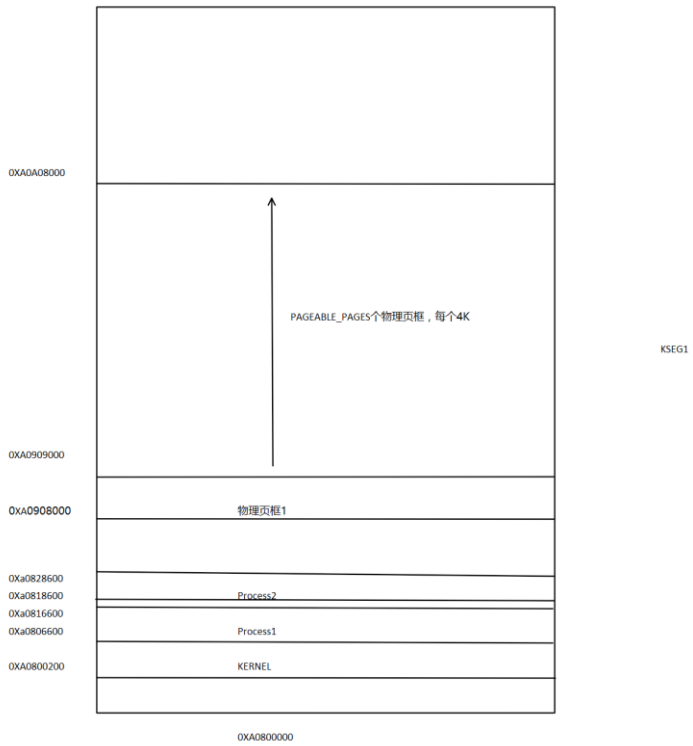
Project 5 Virtual Memory 设计文档

中国科学院大学
张旭
2017/12/20

1. 用户态进程内存管理设计

（1） 测试的用户态进程虚存布局是怎样的？
如下图所示，物理磁盘区域为 0xa0806600 – 0xa0828600



（2） 你设计的页表项结构是怎样的，包含哪些标记位？
页表项结构如下：

```
typedef struct {
    // design here
    uint32_t    entrylo_0;
    uint32_t    entrylo_1;
    uint32_t    entryhi;
    uint32_t    daddr;

} PTE;
```

daddr 记录页表在磁盘中的地址。

（3） 任务 1 中用户态进程页表初始化做了哪些操作？使用了多少个页表项（PTE），以及使用了多少个物理页保存页表？

任务一：

1. 为页表分配页框。
2. 初始化页表项
3. 为每个页表项分配两个物理页框
4. 将进程信息拷贝到物理页框
5. 更新页表项

在我的设计中，使用了 32 个页表项，用两个物理页保存页表。

（4） 任务 2 中用户态进程页表初始化做了哪些操作？使用了多少个页表项（PTE），以及使用了多少个物理页保存页表？

1. 为页表分配页框
2. 初始化页表项

我使用了 32 个页表项，用两个物理页保存页表。

（5） 物理内存使用什么数据结构进行管理，描述物理内存的元数据信息有哪些，各有什么用途？此处的物理页分配策略是什么？

使用 page_map_entry_t 结构管理物理内存：

```
typedef struct {
    // design here
    int         num;
    uint32_t    paddr;
    uint32_t    vaddr;      //映射到内核
    uint8_t     pin;
    uint8_t     used;
    struct page_map_entry_t *next;
    PTE     *pte;
} page_map_entry_t;
```

num 记录页框序号，paddr 保存页框的物理地址，vaddr 保存页框映射到内存的地址，pin 记录该页框是否被钉住，uesd 记录该页框是否被使用，next 记录由被使用页框组成的链表的下一个页框地址，pte 记录该页框被分配给的页表项的地址。

（6） TLB miss 何时发生？你处理 TLB miss 的流程是怎样的？

当 CPU 访问需要映射的地址空间时，若在 TLB 中没有找到对应项，发生 TLB miss。流程：查询进程的页表，若找到页表项且数据有效，则把该页表项填到 TLB 中，若数据无效，则触发 page fault 例外。

2. 缺页中断与 swap 处理设计

（1） 任务 1 和任务 2 中是否有缺页中断？若有，何时发生缺页中断？你设计的缺页中断处理流程是怎样的？

任务 2 中会出现缺页中断。缺页中断就是要访问的页不在主存，需要操作系统将其调入主存后再进行访问。

流程： 选择一个空闲物理页框分配给该地址，若无空闲页框，则从没有被钉住的页框中根据先进先出原则进行替换，并刷新 TLB。然后将磁盘中的页拷贝到该物理页框中，更新页表项和 TLB。

（2） 你设计中哪些页属于 pinning pages？你实现的页替换策略是怎样的？

保存页表的页框属于 pining pages，实现页替换的代码如下：

```
while(1){
    if(head->pin == 0){
        P = head->pte;
        tlb_flush(P->entryhi);
        if((P->entrylo_0 >> 6) == (page_paddr(head->num)>>12)){
            bcopy((char*)page_vaddr(head->num),(char*)P->daddr,PAGE_SIZE);
            P->entrylo_0 &= (~PE_V) ;

        }
        else{
            bcopy((char*)page_vaddr(head->num),(char*)(P->daddr+PAGE_SIZE),PAGE_SIZE);
            P->entrylo_1 &= (~PE_V) ;
        }
        head -> pin = pinned;
        head -> pte = pte;
        tail -> next = head;
        head = head -> next;
        tail = tail->next;
        tail->next = NULL;
        return tail->num;
    }
    tail -> next = head;
    head = head -> next;
    tail = tail->next;
    tail->next = NULL;


}
```

即先进先出原则。

（3） 由于在实验二中，我把页框数改为 3，故导致发生 TLB miss 的次数和 PAGE fault 的次数一样。

参考文献

[1] See MIPS Run