上海交通大學

《操作系统》课程

学生实验报告

实验名称：Project8:Designing a Virtual Memory Manager

姓　　名：洪瑄锐

学　　号：517030910227

班　　级：F1703302

手　　机：15248246044

任课老师：吴晨涛

2019 年 6 月 15 日

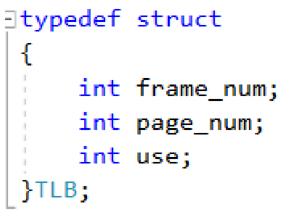
目录

一、实验要求

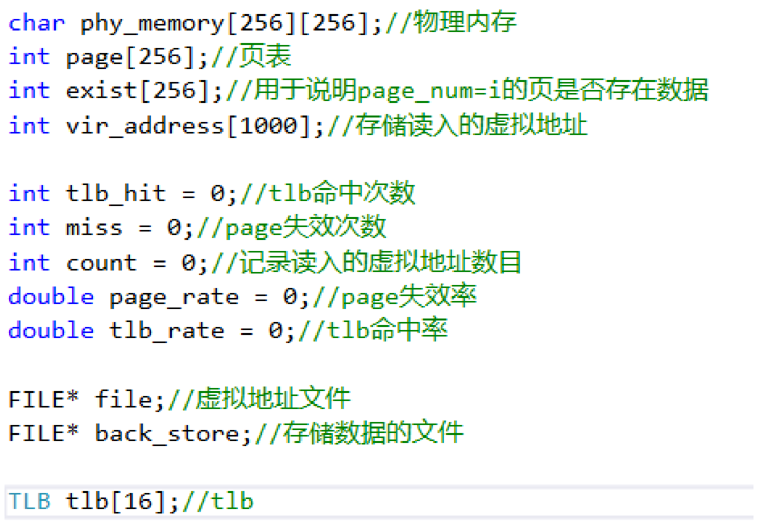
该项目包括编写一个程序，程序将大小为 2^16=65536 字节的虚拟地址空间的逻辑地址转换为物理地址。程序需要读取包含逻辑地址的文件，并使用 TLB 和页表将每个逻辑地址转换为对应的物理地址，并输出存储在转换后的物理地址中的字节值。

二、程序设计思想及代码解释

1. 基本数据结构



```
typedef struct
{
    int frame_num;
    int page_num;
    int use;
}TLB;
```

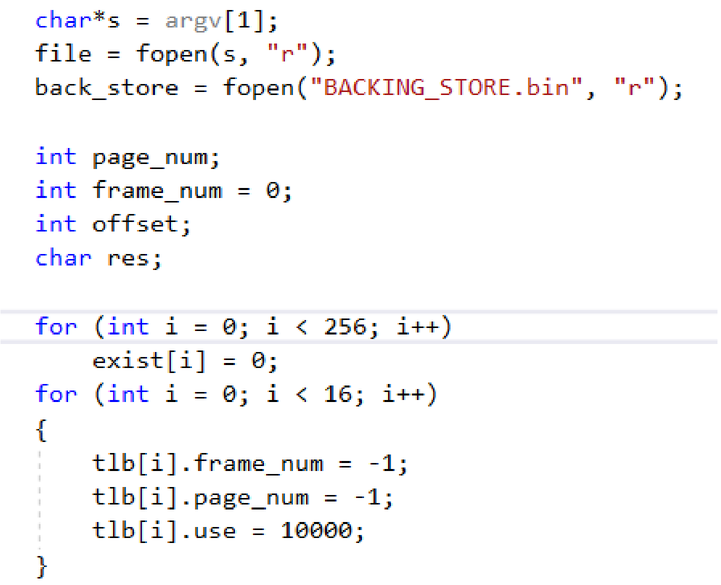


```
char phy_memory[256][256];//物理内存
int page[256];//页表
int exist[256];//用于说明page_num=i的页是否存在数据
int vir_address[1000];//存储读入的虚拟地址

int tlb_hit = 0;//tlb命中次数
int miss = 0;//page失效次数
int count = 0;//记录读入的虚拟地址数目
double page_rate = 0;//page失效率
double tlb_rate = 0;//tlb命中率

FILE* file;//虚拟地址文件
FILE* back_store;//存储数据的文件

TLB tlb[16];//tlb
```

本程序 tlb 失效替换算法采用 lru，use 用来记录每个条目的相对使用时间，初始化为 10000，当该条目被访问时修改为 0. 其余条目各加 1.

2. 初始化

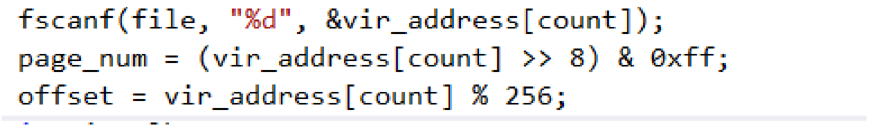
打开存储虚拟地址的文件和存储数据的文件，将 exist[i]初始化为 0，意味着目前第 i 个 page 没有内容，将 tlb[i]的 frame_num 和 page_num 初始化为-1. 意味着无效，use 初始化为 10000.



```c
char*s = argv[1];
file = fopen(s, "r");
back_store = fopen("BACKING_STORE.bin", "r");

int page_num;
int frame_num = 0;
int offset;
char res;

for (int i = 0; i < 256; i++)
    exist[i] = 0;
for (int i = 0; i < 16; i++)
{
    tlb[i].frame_num = -1;
    tlb[i].page_num = -1;
    tlb[i].use = 10000;
}
```

3.对存储虚拟地址的文件进行读取

因为页内偏移量为 2^8,所以对虚拟地址取 256 的模得到 offset,因为共有 2^8 页，所以采用右移 8 位并和掩码相与的方法得到 page_number。

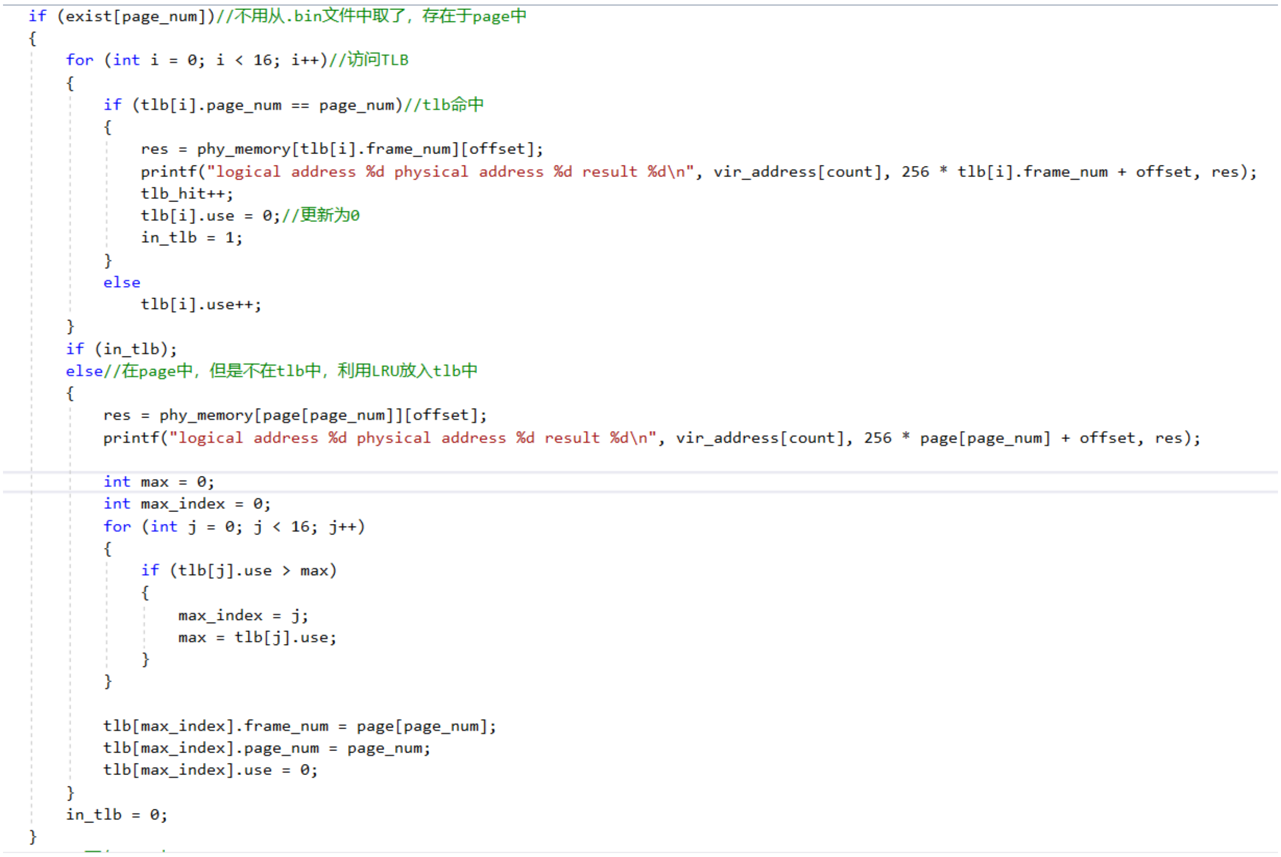


```c
fscanf(file, "%d", &vir_address[count]);
page_num = (vir_address[count] >> 8) & 0xff;
offset = vir_address[count] % 256;
```

4.如果页号为 page_num 的页有内容即 exist[page_num]=1,则遍历 tlb 检查 tlb 中是否存在该页号到 frame_num 的对应。

如果存在,直接从物理内存中读取 phy_memory[tlb[i].frame_num][offset],其物理地址为 tlb[i].frame_num*256+offset，并且 tlb_hit++。

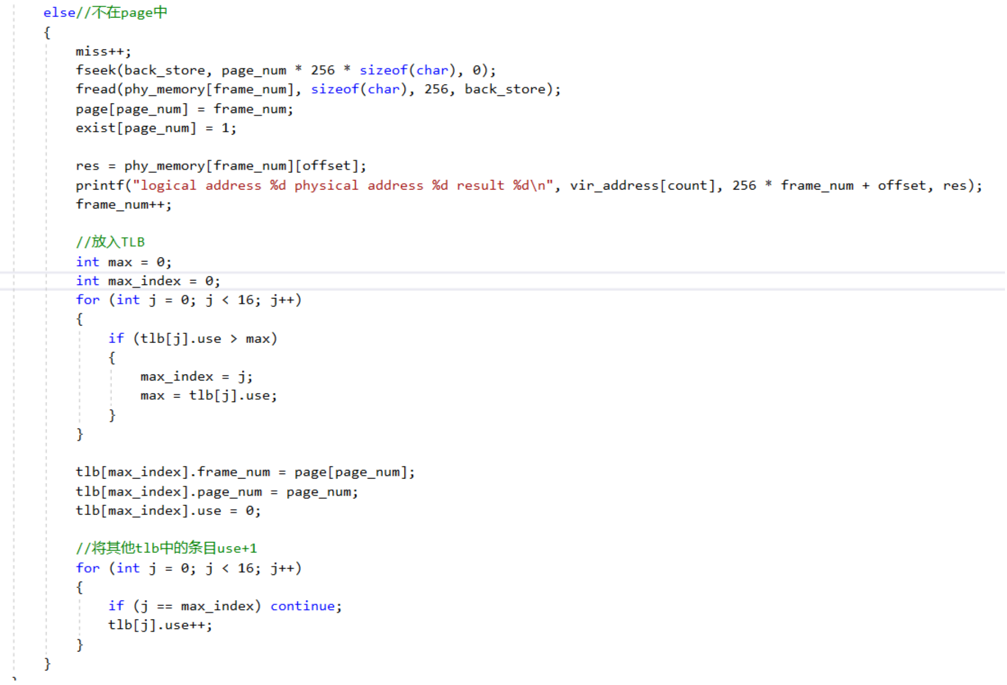
如果不存在，则要采用 lru 算法，将该 page_num 和 frame_num 的对应存入 tlb 中。遍历整个 tlb，找到 use 最大的条目即最近最久未使用的 victim，修改其 page_num,frame_num 以及 use 为 0.同时将其他条目的 use+1.此时其物理地址为 page[page_num]*256+offset.



```
if (exist[page_num])//不用从.bin文件中取了，存在于page中
{
    for (int i = 0; i < 16; i++)//访问TLB
    {
        if (tlb[i].page_num == page_num)//tlb命中
        {
            res = phy_memory[tlb[i].frame_num][offset];
            printf("logical address %d physical address %d result %d\n", vir_address[count], 256 * tlb[i].frame_num + offset, res);
            tlb_hit++;
            tlb[i].use = 0;//更新为0
            in_tlb = 1;
        }
        else
            tlb[i].use++;
    }
    if (in_tlb);
    else//在page中，但是不在tlb中，利用LRU放入tlb中
    {
        res = phy_memory[page[page_num]][offset];
        printf("logical address %d physical address %d result %d\n", vir_address[count], 256 * page[page_num] + offset, res);

        int max = 0;
        int max_index = 0;
        for (int j = 0; j < 16; j++)
        {
            if (tlb[j].use > max)
            {
                max_index = j;
                max = tlb[j].use;
            }
        }

        tlb[max_index].frame_num = page[page_num];
        tlb[max_index].page_num = page_num;
        tlb[max_index].use = 0;
    }
    in_tlb = 0;
}
```

5.如果页号为 page_num 的页无内容，则需要从 backing store 文件中读取 256 个数据放入 phy_memory 中，并且 page miss++。注意确定 frame_num 的方法，frame_num 是一个全局变量，初始化为 0。当从二进制文件中读取数据时，page_num 对应的 frame_num 为当前的 frame_num,然后将 frame_num+1 供下一次读取二进制文件使用。要将 exist[page_num]设置为 1.

此外，需要更新 tlb，遍历整个 tlb 找到 use 最大的条目，修改其 page_num,frame_num 和 use。



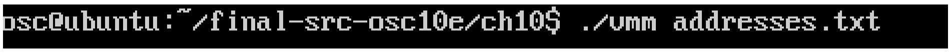
```
else//不在page中
{
    miss++;
    fseek(back_store, page_num * 256 * sizeof(char), 0);
    fread(phy_memory[frame_num], sizeof(char), 256, back_store);
    page[page_num] = frame_num;
    exist[page_num] = 1;

    res = phy_memory[frame_num][offset];
    printf("logical address %d physical address %d result %d\n", vir_address[count], 256 * frame_num + offset, res);
    frame_num++;

    //放入TLB
    int max = 0;
    int max_index = 0;
    for (int j = 0; j < 16; j++)
    {
        if (tlb[j].use > max)
        {
            max_index = j;
            max = tlb[j].use;
        }
    }

    tlb[max_index].frame_num = page[page_num];
    tlb[max_index].page_num = page_num;
    tlb[max_index].use = 0;

    //将其他tlb中的条目use+1
    for (int j = 0; j < 16; j++)
    {
        if (j == max_index) continue;
        tlb[j].use++;
    }
}
```

6.最后计算 tlb 命中率和 page 失效率并输出。

```
tlb_rate = tlb_hit / 1000.0;
page_rate = miss / 1000.0;
printf("Page miss rate %f , TLB hit rate %f\n", page_rate, tlb_rate);
```

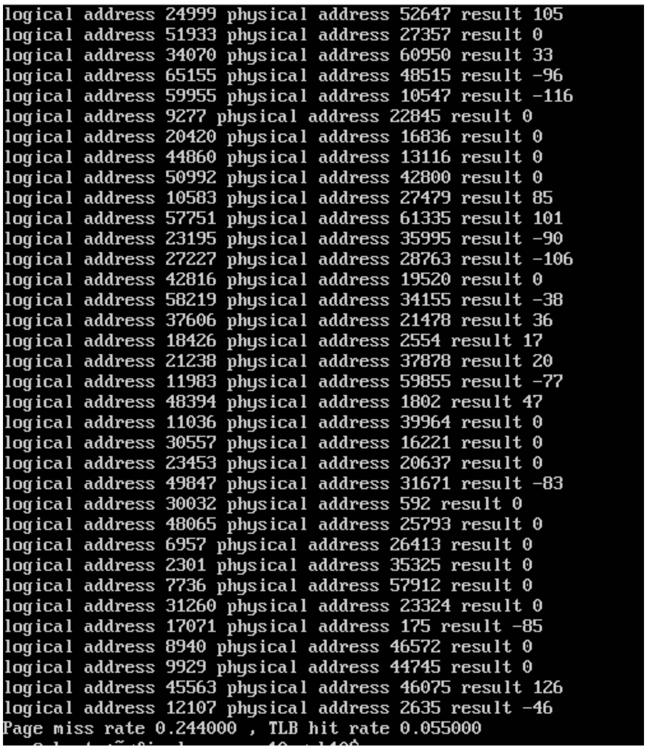
三、运行结果截图

输入命令：



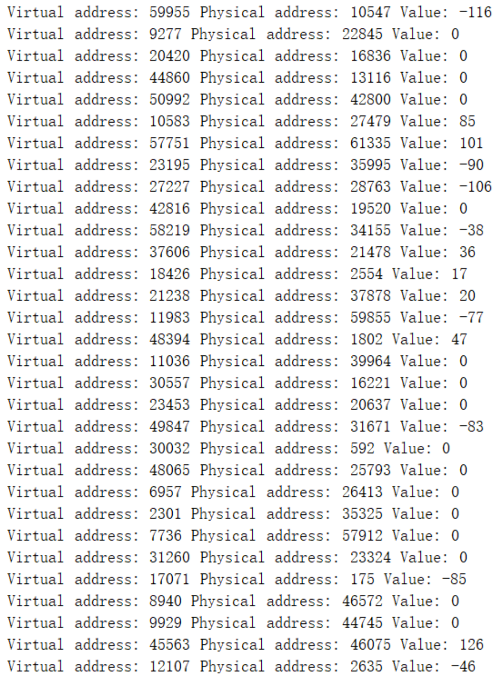
```
osc@ubuntu:~/final-src-osc10e/ch10$ ./vmm addresses.txt
```

得到结果：（部分截图）



```
logical address 24999 physical address 52647 result 105
logical address 51933 physical address 27357 result 0
logical address 34070 physical address 60950 result 33
logical address 65155 physical address 48515 result -96
logical address 59955 physical address 10547 result -116
logical address 9277 physical address 22845 result 0
logical address 20420 physical address 16836 result 0
logical address 44860 physical address 13116 result 0
logical address 50992 physical address 42800 result 0
logical address 10583 physical address 27479 result 85
logical address 57751 physical address 61335 result 101
logical address 23195 physical address 35995 result -90
logical address 27227 physical address 28763 result -106
logical address 42816 physical address 19520 result 0
logical address 58219 physical address 34155 result -38
logical address 37606 physical address 21478 result 36
logical address 18426 physical address 2554 result 17
logical address 21238 physical address 37878 result 20
logical address 11983 physical address 59855 result -77
logical address 48394 physical address 1802 result 47
logical address 11036 physical address 39964 result 0
logical address 30557 physical address 16221 result 0
logical address 23453 physical address 20637 result 0
logical address 49847 physical address 31671 result -83
logical address 30032 physical address 592 result 0
logical address 48065 physical address 25793 result 0
logical address 6957 physical address 26413 result 0
logical address 2301 physical address 35325 result 0
logical address 7736 physical address 57912 result 0
logical address 31260 physical address 23324 result 0
logical address 17071 physical address 175 result -85
logical address 8940 physical address 46572 result 0
logical address 9929 physical address 44745 result 0
logical address 45563 physical address 46075 result 126
logical address 12107 physical address 2635 result -46
Page miss rate 0.244000 , TLB hit rate 0.055000
```

附答案对比（部分截图）：



```
Virtual address: 59955 Physical address: 10547 Value: -116
Virtual address: 9277 Physical address: 22845 Value: 0
Virtual address: 20420 Physical address: 16836 Value: 0
Virtual address: 44860 Physical address: 13116 Value: 0
Virtual address: 50992 Physical address: 42800 Value: 0
Virtual address: 10583 Physical address: 27479 Value: 85
Virtual address: 57751 Physical address: 61335 Value: 101
Virtual address: 23195 Physical address: 35995 Value: -90
Virtual address: 27227 Physical address: 28763 Value: -106
Virtual address: 42816 Physical address: 19520 Value: 0
Virtual address: 58219 Physical address: 34155 Value: -38
Virtual address: 37606 Physical address: 21478 Value: 36
Virtual address: 18426 Physical address: 2554 Value: 17
Virtual address: 21238 Physical address: 37878 Value: 20
Virtual address: 11983 Physical address: 59855 Value: -77
Virtual address: 48394 Physical address: 1802 Value: 47
Virtual address: 11036 Physical address: 39964 Value: 0
Virtual address: 30557 Physical address: 16221 Value: 0
Virtual address: 23453 Physical address: 20637 Value: 0
Virtual address: 49847 Physical address: 31671 Value: -83
Virtual address: 30032 Physical address: 592 Value: 0
Virtual address: 48065 Physical address: 25793 Value: 0
Virtual address: 6957 Physical address: 26413 Value: 0
Virtual address: 2301 Physical address: 35325 Value: 0
Virtual address: 7736 Physical address: 57912 Value: 0
Virtual address: 31260 Physical address: 23324 Value: 0
Virtual address: 17071 Physical address: 175 Value: -85
Virtual address: 8940 Physical address: 46572 Value: 0
Virtual address: 9929 Physical address: 44745 Value: 0
Virtual address: 45563 Physical address: 46075 Value: 126
Virtual address: 12107 Physical address: 2635 Value: -46
```