Project 8

秦啸涵　**521021910604**

实现虚拟内存管理器

从address.txt读取逻辑地址

通过 `main()` 函数参数传递实现

```
int main(int argc, char*argv[])
{
    addresses = fopen(argv[1], "r");            //./memory address.txt
    fscanf(addresses, "%u", &address);
    ...
}
```

使用TLB和页表进行地址翻译得到物理地址并读取对应字节

定义TLB结构和页表结构

```
typedef struct TLB_ITEM
{
    int used_time;
    int frame_id;
    int page_id;
} tlb_item;
typedef struct PAGE_TABLE_ITEM
{
    int valid;
    int frame_id;
} page_table_item;
```

通过地址得到页号和页内偏移

```c
/* get the page num, given address*/
int get_page(int address)
{
    address = address>>8;
    return address;
}


/*get the offset, given address*/
int get_offset(int address)
{
    return address - (get_page(address)<<8);
}
```

之后查找TLB(miss则查找页表)得到物理地址页号，最后与偏移量拼接得到真实物理地址，并从文件中找到对应数据

缺页时从BACKING_STORE.bin中加载页面并更新TLB和页表

这里使用LRU算法进行TLB的更新

```c
// search in TLB
int tlb_find=0;
for(int i=0; i<TLB_ENTRY_NUM; i++)
{
    if(page_id==TLB[i].page_id)
    {
        tlb_hit ++;
        tlb_find = 1;
        frame_id = TLB[i].frame_id;
        memory[frame_id].used_time = time;
        TLB[i].used_time = time;
        break;
    }
}

// not find in TLB ,search in page table
```

```
int page_find=0;
if(!tlb_find)
{
    // valid = 1, find page
    if(page_table[page_id].valid==1)
    {
        page_find = 1;
        frame_id = page_table[page_id].frame_id;
        memory[frame_id].used_time = time;
        TLB_LRU_Replacement(page_id, frame_id, time); // Update TLB

    }
    else // valid = -1, not find, page fault , do demand paging
    {
        page_fault++;

        // demand paging
        frame_id = memory_LRU_Replacement(page_id, time);

        //update page table
        page_table[page_id].frame_id = frame_id;
        page_table[page_id].valid = 1;// set valid

        //update TLB
        TLB_LRU_Replacement(page_id, frame_id, time);

    }
}
```

当发生 `page_fault` 时，执行 `memory_LRU_Replacement()` ，在该函数中使用文件指针读取 `BACKING_STORE.bin` 来加载页面

```
int memory_LRU_Replacement(int page_id, int time)
{
    int min_time=time;
    int min_idx = 0;
    // find the least recently used
    for(int i=0; i<FRAME_NUM; i++)

```

```c
    {
        if(memory[i].used_time<min_time)
        {
            min_time = memory[i].used_time;
            min_idx = i;
        }
    }
    memory[min_idx].used_time = time;

    //find the old page id, and set invalid
    for(int i=0; i<PAGE_TABLE_SIZE; i++)
    {
        if(page_table[i].frame_id==min_idx)
        {
            page_table[i].valid = -1;
        }
    }

    // seek data
    fseek(backing_store, page_id*PAGE_SIZE, SEEK_SET);
    fread(memory[min_idx].data, sizeof(char), FRAME_SIZE, backing_store);

    return min_idx;
}
```

统计并报告Page-fault rate和TLB hit rate

程序运行结果如下:

```
jianke@ubuntu:~/Desktop/final-src-osc10e/ch10$ gcc memory.c -o memory
jianke@ubuntu:~/Desktop/final-src-osc10e/ch10$ ./memory addresses.txt
Initialize Finish.
Execution Finish.
-----------------------------
Frame Num = 256:
TLB Hit Rate: 0.055000
Page Fault Rate: 0.244000
```

(这里我采用的是LRU算法，不同算法实现得到的TLB hit rate和Page fault rate会不一样)

标答:

ch10 > correct.txt

```
Virtual address: 16916 Physical address: 20 Value: 0
Virtual address: 62493 Physical address: 285 Value: 0
Virtual address: 30198 Physical address: 758 Value: 29
Virtual address: 53683 Physical address: 947 Value: 108
Virtual address: 40185 Physical address: 1273 Value: 0
Virtual address: 28781 Physical address: 1389 Value: 0
Virtual address: 24462 Physical address: 1678 Value: 23
Virtual address: 48399 Physical address: 1807 Value: 67
Virtual address: 64815 Physical address: 2095 Value: 75
Virtual address: 18295 Physical address: 2423 Value: -35
Virtual address: 12218 Physical address: 2746 Value: 11
Virtual address: 22760 Physical address: 3048 Value: 0
Virtual address: 57982 Physical address: 3198 Value: 56
Virtual address: 27966 Physical address: 3390 Value: 27
Virtual address: 54894 Physical address: 3694 Value: 53
```

我的输出：

ch10 > result.txt

```
Virtual address: 16916 Physical address: 20 Value: 0
Virtual address: 62493 Physical address: 285 Value: 0
Virtual address: 30198 Physical address: 758 Value: 29
Virtual address: 53683 Physical address: 947 Value: 108
Virtual address: 40185 Physical address: 1273 Value: 0
Virtual address: 28781 Physical address: 1389 Value: 0
Virtual address: 24462 Physical address: 1678 Value: 23
Virtual address: 48399 Physical address: 1807 Value: 67
Virtual address: 64815 Physical address: 2095 Value: 75
Virtual address: 18295 Physical address: 2423 Value: -35
Virtual address: 12218 Physical address: 2746 Value: 11
Virtual address: 22760 Physical address: 3048 Value: 0
Virtual address: 57982 Physical address: 3198 Value: 56
Virtual address: 27966 Physical address: 3390 Value: 27
```

内容一致

bonus:物理内存128个页帧

实现方式与之前基本一致，只不过由于虚拟内存大于物理内存，需要解决页面替换的问题，而我已经通过LRU实现。因此只需要更改 `memory` 的size即可

```
#define FRAME_NUM 128
memory_item memory[FRAME_NUM];
//之后逻辑与前文一致，发生page fault时使用LRU替换即可
```

实现结果

```
jianke@ubuntu:~/Desktop/final-src-osc10e/ch10$ gcc memory.c -o memory
jianke@ubuntu:~/Desktop/final-src-osc10e/ch10$ ./memory addresses.txt
Initialize Finish.
Execution Finish.
-----------------------------
Frame Num = 128:
TLB Hit Rate: 0.055000
Page Fault Rate: 0.539000
```

Frame Num = 128, 即物理内存128个页帧的情况，page fault rate相比256页帧明显上升