**Project 8: Designing a Virtual Memory Manager**

（一）问题分析

该项目需要模拟一个虚拟内存管理器，实现以下功能：

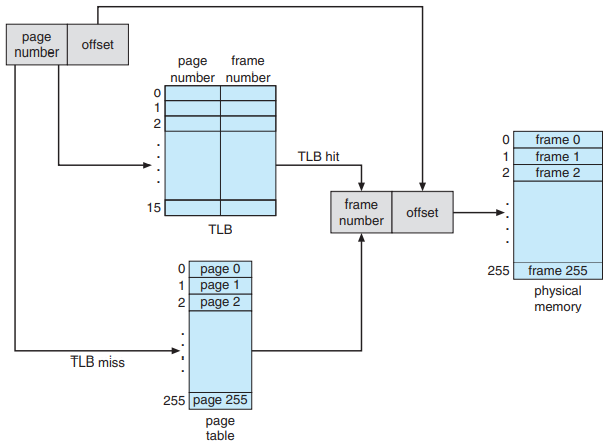
1、维护一个基于分页的虚拟内存空间。

2、通过TLB或页表将逻辑地址转换为物理地址。

3、请求调页（发生缺页错误时进行磁盘与物理内存间的页面置换）。

4、通过物理地址访问物理内存。

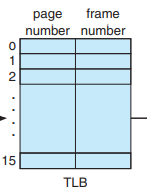
5、计算缺页错误率和TLB命中率。

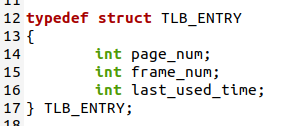


（二）实现细节

一、数据结构

1、TLB条目



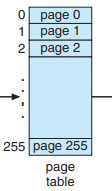


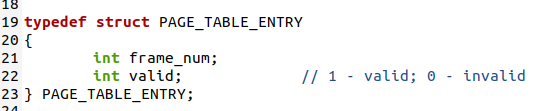
（1）page\_num: 对应页表中的页码。

（2）frame\_num: 对应物理内存中的帧码。

（3）last\_used\_time: 该条目最后一次被访问的时间。

2、页表条目

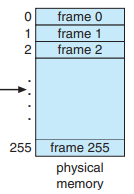


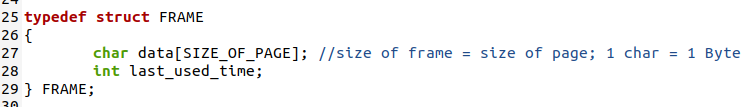


（1）frame\_num: 对应物理内存中的帧码。

（2）valid: 有效-无效位。记录该页是否在物理内存空间内。1表示存在，0表示不存在或已被换出。

3、物理内存帧





（1）data[]: 物理内存的每帧存储256个1 Byte的数据，在c语言中char类型刚为1字节。页面大小为256，帧大小等于页面大小。

（2）last\_used\_time: 该帧最后一次被访问的时间。

二、全局变量

1、TLB[]: 转换表缓冲区。

2、page\_table[]: 页表。

3、physical\_memory[]: 物理内存。

4、pc: 记录内存引用次数。

5、page\_fault: 记录缺页错误次数。

6、TLB\_hit: 记录TLB命中次数。

7、time: 记录当前时间。

三、函数

1、init(): 初始化所有全局变量。

2、page\_number\_in(): 通过移位操作，从16位逻辑地址中取出前8位，前8位为页表中的页码。

3、offset\_in(): 通过并运算，从16位逻辑地址中取出后8位，后8位为offset。

4、TLB\_replacement(): 采用LRU算法。

（1）遍历TLB，查找最久没有被访问的条目，即最后一次被访问时间最早的条目。因为初始化时空条目的最后访问时间被初始化为-1，所以存在空条目时会选中空条目。

（2）用本次被访问的页码和帧码更新该条目。

5、page\_replacement(): 采用LRU算法。

（1）遍历物理内存，查找最久没有被访问的帧，即最后一次被访问时间最早的帧。因为初始化时空闲帧的最后访问时间被初始化为-1，所以存在空闲帧时会选中空闲帧。

（2）将磁盘（“BACKING\_STORE.bin”）中与请求调入内存的页面对应的存储区域的数据写入该帧（利用fseek()和fread()）。因为主函数对内存数据只做读操作，所以换入物理内存的数据不会被修改，在页面置换时不需要将物理内存中的数据换出。

（3）更新页表。页表中帧码为（1）中选中帧的条目失效，因为该帧已换入新页面。

（4）更新TLB。TLB中帧码为（1）中选中帧的条目被移除，即设置为空条目。

（5）返回选中帧的帧码。

6、get\_frame\_num()

（1）首先遍历TLB，查找是否存在目标页码对应的条目，若存在，TLB命中次数增加1，更新该条目的最后访问时间，返回该条目的帧码。

（2）如果遍历结束，没有退出函数，说明TLB未命中。接着在页表中查找。

（3）如果目标页码对应的条目失效，缺页错误次数增加1，进行页面置换。用page\_replacement()的返回值，即置换后访问的物理内存区域的帧码更新该条目的帧码，将有效-无效位设为有效。

（4）用目标页码和该条目的帧码更新TLB。

（5）返回该条目的帧码。

7、access\_memory()

（1）更新目标帧码对应的物理内存帧的最后访问时间。

（2）返回该帧的offset位置存储的数据。

四、主函数

1、执行内存引用

（1）初始化。

（2）循环读入addresses.txt中的逻辑地址。

（3）通过并运算取出有用的位数，即后16位。

（4）调用page\_number\_in()函数得到页码。

（5）调用offset\_in()函数得到offset。

（6）调用get\_frame\_num()函数得到帧码。

（7）调用access\_memory()函数访问物理内存得到读取的数据。

（8）每次循环都更新内存引用次数和当前时间。

（9）每次访问内存都输出逻辑地址、物理地址和读到的数据到output.txt文件。

（10）循环结束后，在控制台输出物理内存帧的数量。

（11）计算缺页错误率和TLB命中率，并在控制台输出。

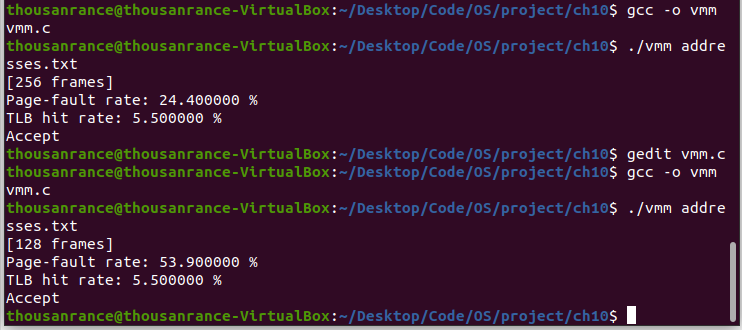
2、检查输出结果

（1）循环读入correct.txt和output.txt中的数据，检查是否一致。

（2）如果不一致，输出错误出现的行数。

（3）循环结束后，输出检查结果。

（三）运行结果



可以看出，当物理内存变大时，缺页错误率会变低。