一个虚拟内存管理器

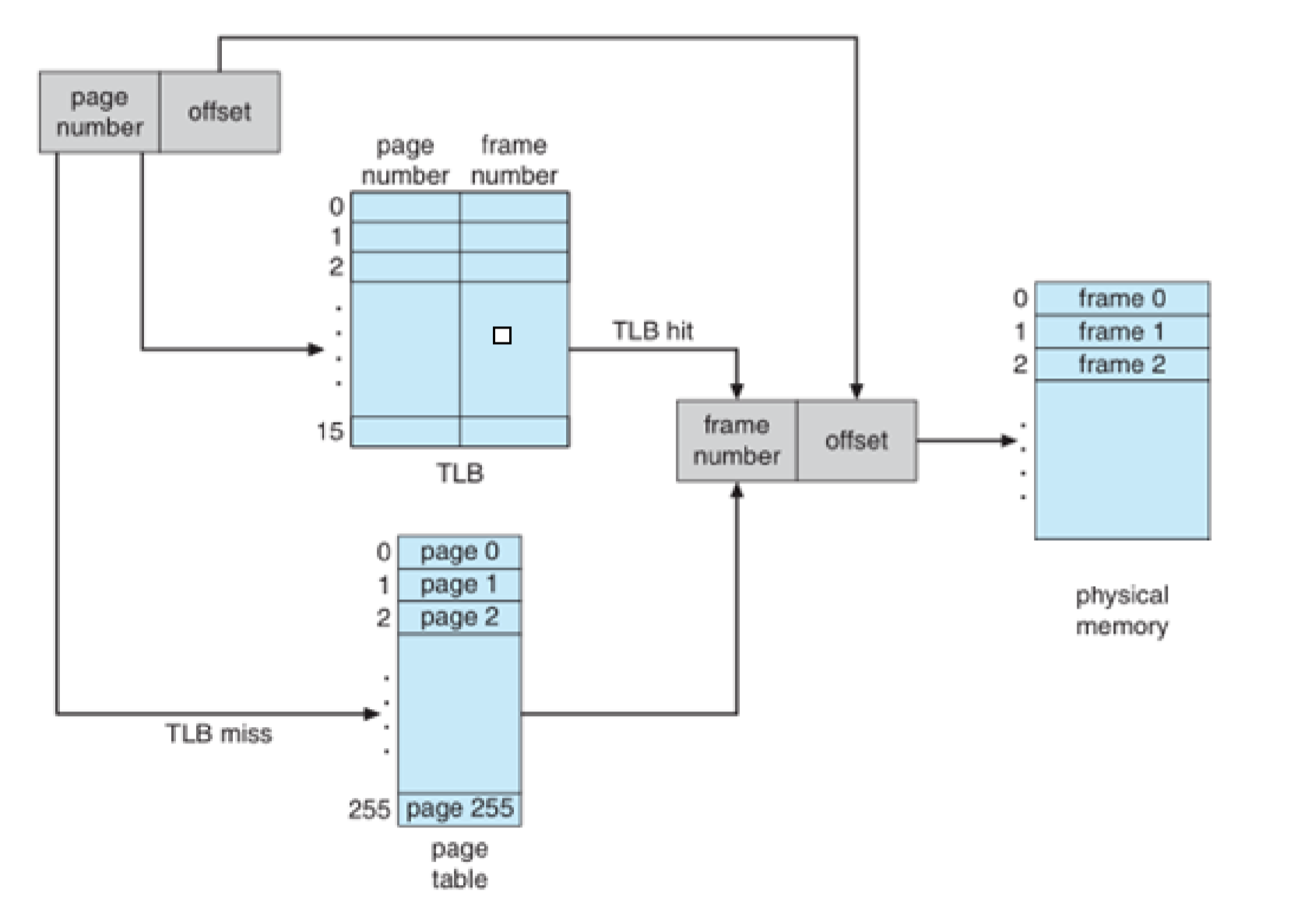
1. 实验内容

实现虚拟内存的分页系统，模拟进程调度过程中的内存管理。

1. 系统设置
   * 1. 假设系统中有 5 个进程同时运行。每个进程的虚拟地址为 32-bit virtual address，虚拟页面大小为 8KB。每个进程的大小分别为 1MB， 32MB，64MB，128MB，和 256MB，并且每个进程一半数据占据低位的虚拟地址空间，一半占据高位的虚拟地址空间。如进程 1 数据分布 的 虚 拟 地 址 范 围 为 0x00000000~0x0007FFFF 以 及

0xFFFF8000~0xFFFFFFFF。

* + 1. 物理内存大小为 64MB，并同样划分成大小为 8KB 的页框。
    2. 利用带旁路缓冲 TLB 的分页系统实现虚拟页面到物理页的映射，如下图 1 所示。TLB 包含 16 个 entry，用来保存当前进程在运行过程中最近被访问的 16 个虚拟页面到物理页面的映射。每一个进程的 page table 大小由其虚拟地址空间对应的页面数量决定。



1. 系统实现
   * 1. 五个进程轮流执行，总共执行三轮。

每个进程在第一次执行时，页表内容为空，每次执行完毕出现进程切换时，将其页表内容保存到外部文件。在该程序之后执行时，首先从外部文件导入页表内容。

TLB 用来保存当前运行进程最近的页访问信息，因此在进程切换时，需要清空 TLB 为新进程的运行做准备。

* + 1. 每个进程在一次执行过程中，利用随机函数生成 500 次虚拟地址的访问序列。例如，进程 1 在执行过程中，生成 500 个随机虚拟地址访问序列，每个虚拟地址均需落在 0x00000000~0x0007FFFF 或者

0xFFFF8000~0xFFFFFFFF 范围内。

生成随机虚拟地址序列的方式为：首先随机生成一个地址，然后以该地址为起始点生成九个连续的地址。按此方法循环 50 次，总共生成

500 次随机虚拟地址访问。

* + 1. 若访问某个虚拟地址，首先生成对应的虚拟页号（virtual page number），再查找该虚拟页对应的物理页框号（page frame number）

是否在 TLB；若不在 TLB，进一步查找是否在 page table；若也不在，则分配一个物理页框给该页面，并将虚拟页到物理页的映射信息保

存在 page table 和 TLB 中。

若 TLB 已满，按照 LRU 的置换策略置换掉最早被访问的 entry。

* + 1. 物理页面在分配的过程中，若仍然有可用的页框，按照页框编号从小到大依次分配。若没有可用页框，则利用 LRU 置换策略，将最早分配的页框收回重新分配给当前的虚拟页。被收回的页框若是已经分

配给其他进程，则需要更新该进程的 page table。

1. 系统输出
   * 1. 将每个进程 i 的 1500 次随机虚拟地址访问序列输出保存在文件 addr\_seq\_i.txt 中;
     2. 将每个进程 i 的 1500 次访问信息输出保存在 visit\_seq\_i.txt 中； 一次访问对应信息如下：

VP number PF number TLB hit/no hit PT hit/no hit；

并计算该进程在这三轮执行过程中的页缺失率（page fault rate），输出。

* + 1. 将每个进程 i 最后一次运行结束后的 page table 保存在 page\_table\_i.txt 文件中。