Lab 7: Virtual Memory

Total points: 100

Assignment 1 (30 points) –Programing Problem 10.44 (page P-51)

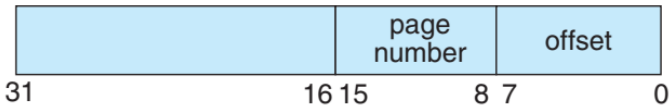
Write a program that implements the FIFO, LRU, and optimal (OPT) page-replacement algorithms presented in Section 10.4. Have your program initially generate a random page-reference string where page numbers range from 0 to 9. Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm. Pass the number of page frames to the program at startup. You may implement this program in any programming language of your choice.

Assignment 2 (70 points) - Programming Projects – **Designing a Virtual Memory Manager** (page P-51)

This project consists of writing a program that translates logical to physical addresses for a virtual address space of size $2^{16} = 65,536$ bytes. Your program will read from a file containing logical addresses and, using a TLB and a page table, will translate each logical address to its corresponding physical address and output the value of the byte stored at the translated physical address. Your learning goal is to use simulation to understand the steps involved in translating logical to physical addresses. This will include resolving page faults using demand paging, managing a TLB, and implementing a page-replacement algorithm.

Specific

Your program will read a file containing several 32-bit integer numbers that represent logical addresses. However, you need only be concerned with 16-bit addresses, so you must mask the rightmost 16 bits of each logical address. These 16 bits are divided into (1) an 8-bit page number and (2) an 8-bit page offset. Hence, the addresses are structured as shown as:



Other specifics include the following:

• 2^8 entries in the page table

• Page size of 2^8 bytes

• 16 entries in the TLB

• Frame size of 2^8 bytes

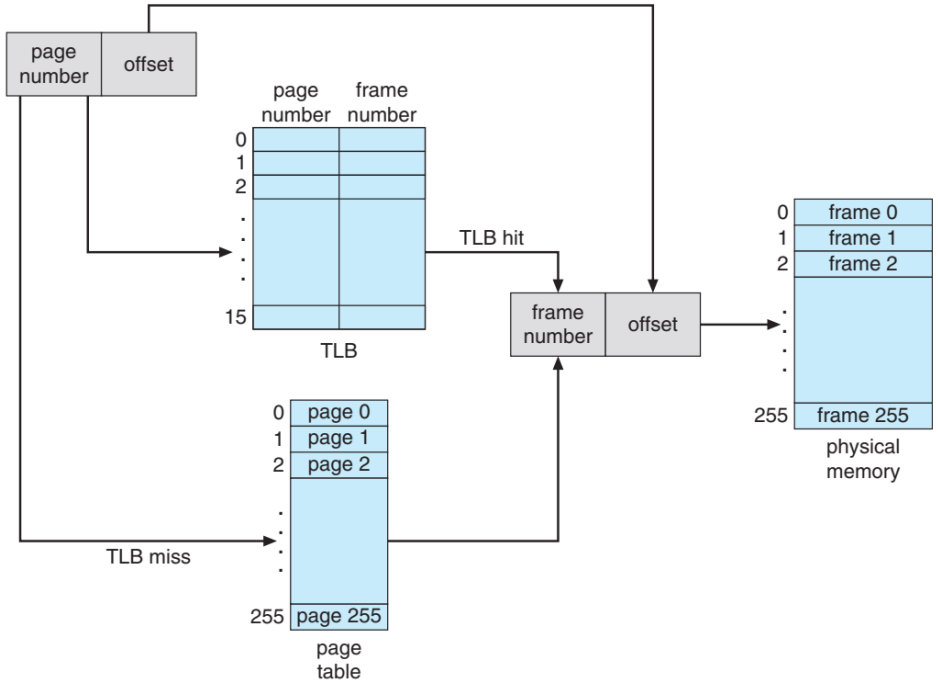
• 256 frames

• Physical memory of 65,536 bytes (256 frames × 256-byte frame size)

Additionally, your program need only be concerned with reading logical addresses and translating them to their corresponding physical addresses. You do not need to support writing to the logical address space.

Address Translation

Your program will translate logical to physical addresses using a TLB and page table as outlined in Section 9.3. First, the page number is extracted from the logical address, and the TLB is consulted. In the case of a TLB hit, the frame number is obtained from the TLB. In the case of a TLB miss, the page table must be consulted. In the latter case, either the frame number is obtained from the page table, or a page fault occurs. A visual representation of the address translation process is:



Handling Page Faults

Your program will implement demand paging as described in Section 10.2. The backing store is represented by the file `BACKING_STORE.bin`, a binary file of size 65,536 bytes. When a page fault occurs, you will read in a 256-byte page from the file BACKING STORE and store it in an available page frame in physical memory. For example, if a logical address with page number 15 resulted in a page fault, your program would read in page 15 from BACKING STORE (remember that pages begin at 0 and are 256 bytes in size) and store it in a page frame in physical memory. Once this frame is stored (and the page table and TLB are updated), subsequent accesses to page 15 will be resolved by either the TLB or the page table.

You will need to treat `BACKING_STORE.bin` as a random-access file so that you can randomly seek to certain positions of the file for reading. We suggest using the standard C library functions for performing I/O, including fopen(), fread(), fseek(), and fclose().

The size of physical memory is the same as the size of the virtual address space—65,536 bytes—so you do not need to be concerned about page replacements during a page fault. Later, we describe a modification to this project using a smaller amount of physical memory; at that point, a page-replacement strategy will be required.

Test File

We provide the file `addresses.txt`, which contains integer values representing logical addresses ranging from 0to65535 (the size of the virtual address space). Your program will open this file, read each logical

address and translate it to its corresponding physical address, and output the value of the signed byte at the physical address.