CS 1550 – Introduction to Operating Systems

Project 3 – Virtual Memory

Due Date – December 13, 2013 (11:59pm)

Project Description

Virtual memory abstracts the physical memory into an extremely large, uniform storage array, separating logical memory as viewed by the user from the physical memory as seen by the processor. In particular, virtual memory conceals the fact that the physical memory is limited and shared among multiple processes. It creates the illusion that a process has one or more contiguous private namespaces, each beginning at address zero. The main objective of this project is to simulate the steps involved in translating logical to physical addresses. As part of a project you are required to:

1. Write a program that reads from a file containing logical addresses and, using a TLB as well as a page table, translates each logical address to its corresponding physical address and output the value of the byte stored at the translated physical address.
2. The program must also report the page-fault rare and TLB hit rate.

Project Specific

In this project, we consider a virtual spaced of size 2^{16} = 65,536 bytes. The program will read a file containing several 32-bit integer numbers that represent logical addresses. However, you need to 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) 8-bit page number and (2) 8-bit page offset. Hence, the addresses are structured as shown in Figure 1.

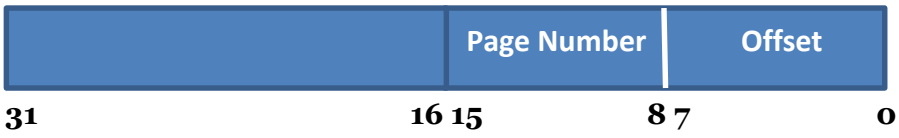

Figure 1 – Address Structure

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, and
- Physical memory of 2^{16} = 65,536 bytes = 256 frames of 256 bytes each.

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

Address Translation

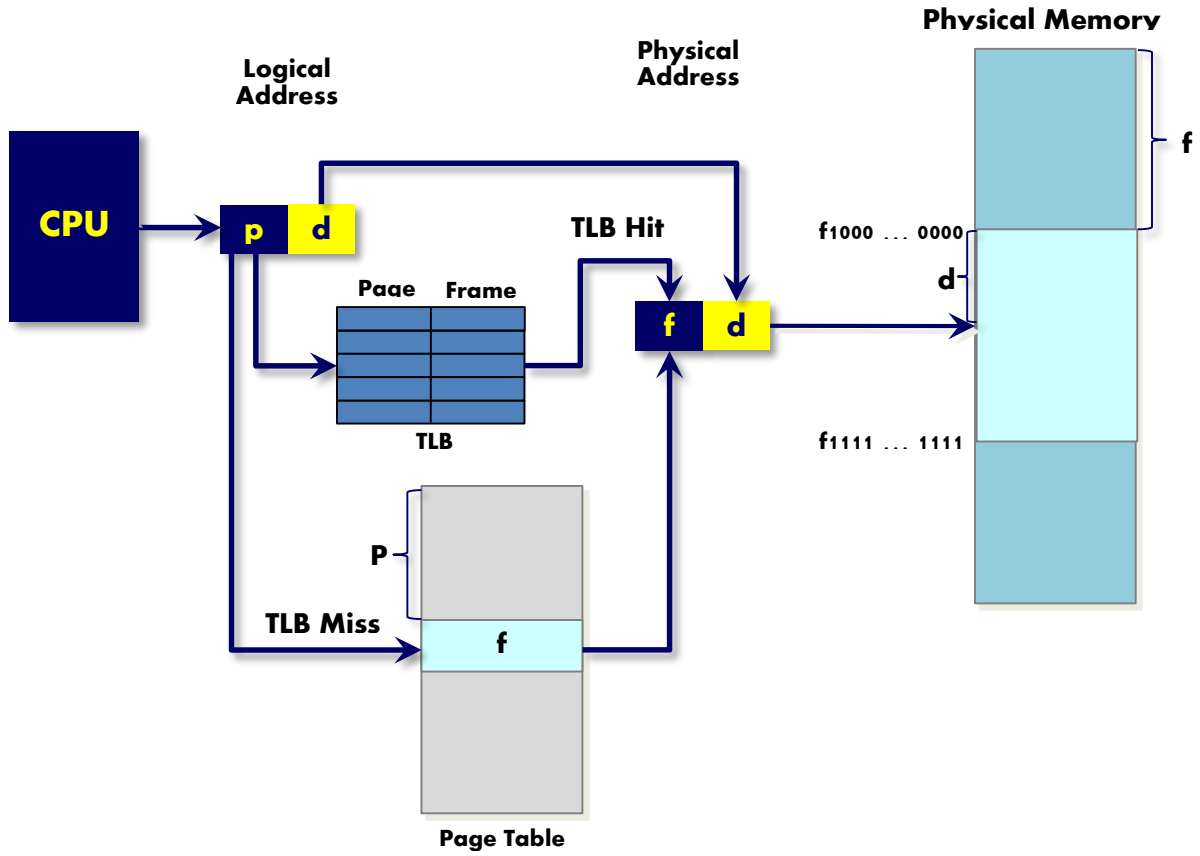

Figure 2 – Address Translation Process

You program will translate logical to physical addresses using a TLB and a page table. 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 virtual representation of address-translation process appears in Figure 2.

Handling Page Faults

Your program will implement demand paging. The backing store is presented by the file BACKIN_STORE.bin, as 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.bin and store it in an available page frame in physical memory. For example, if a logical address with page number 15 resulted in page fault, your program would read in page 15 from BACKING_STORE.bin (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 sot that you can randomly seek to certain positions of the file for reading. It is suggested that you use 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, namely 2^{16} = 65,536 bytes. Consequently, **you do not need to be concerned about page replacements during a page fault.**

Test File

A file, referred to as addresses.txt, will be provided. The file contains integer values representing addresses ranging from 0 to 65,535, 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. It then outputs the value of the signed byte at the physical address.

Suggested Steps toward the completion of the project

1. **Step 1.** Write a simple program that extracts the page number and offset, based on Figure 1, from the integer numbers:
    - This can be easily achieved by using the operator for bit-masking and bit-shifting. Once the simple program correctly establishes the page number and offset from an integer number, you are ready to move the next step.
2. **Step 2**. Bypass the TLB and use only a page table. Remember, address translation can work without a TLB. The TLB just makes it faster. Recall that since the physical

memory is the same size as the virtual address space, no page replacement algorithm is needed.

3. **Step 3**. Integrate the TLB once your page table is working properly. When you are ready to implement the TLB, recall that it has only 16 entries, so you will need to use a replacement strategy when you update a full TLB. In this project, you are will use the **FIFO policy** for updating the TLB.

How to Run Your Program

Your program should run as follows:

```
./VirtMemSim BACKING_STORE addresses.txt
```

Your program will read in the file addresses.txt, which contains 1,000 logical addresses ranging from 0 to 65,535. Your program is to translate each logical address to a physical address and determine the contents of the physical byte stored at the correct physical address. In the C language, the char data type occupies a byte of storage, it is recommended using char values.

Your program should output the following values:

- The logical address being translated (the integer valued being read from addresses.txt).
- The corresponding physical address (what your program translates the logical address to).
- The signed byte value stored at the translated physical address.

A file, referred to as **correct.txt**, is provided. The file contains the correct output values for the files **addresses.txt**. You should use this file to determine if you program is correctly translating logical to physical addresses. Note, however, that other test files will be used to verify the correctness of your program.

Statistics

After completion, your program is to report the following statistics:

- Page-Fault rate – The percentage of address references that resulted in page faults.
- TLB hit rate – The percentage of address references that were resolved in the TLB.

Since the logical addresses in addresses.txt were generated randomly and not reflect any memory access locality, it is unlikely that TLB hit rate is high.

Requirements and Submission

For a full credit, the following must be achieved by the due date:

- Make a tar.gz file, named USERNAME-PrjPartII.tar.gz, and copy it to **~znati/submit/1550Prj3** by the deadline.
- In the tar file, you need to include **well-commented** file(s) of the source code developed for this project.