Operating Systems - Lab Assignment 3

Deadline: Nov. 11, 2023

This task 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. This project aims to simulate the steps involved in translating logical to physical addresses.

Specifics
Your program will read the file (`addresses.txt`) containing several 32-bit integer numbers representing 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 (i) an 8-bit page number and (ii) an 8-bit page offset. Other specifics include the following:

- 16 entries in the TLB
- 256 entries in the page table
- Page size of 256 bytes
- 256 frames in the physical memory
- Frame size of 256 bytes
- Physical memory of 65,536 bytes (256 frames $\times$ 256-byte frame size)

Address Translation
Using a TLB and page table, your program will translate logically to physical addresses. 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, the frame number is obtained from the page table, or a page fault occurs.

Handling Page Faults
Your program will implement demand paging. The backing store (e.g., your

disk) 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.bin` 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 on 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 so that you can randomly seek 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.

Test File
We provide the file `addresses.txt`, which contains integer values representing logical addresses ranging from 0 - 65,535 (the size of the virtual address space). Your program will open this file, read each logical address, translate it to its corresponding physical address, and output the value of the signed byte at the physical address.

How to Begin
First, write a simple program that extracts the page number and offset (an 8-bit page number and 8-bit page offset) from the following integer numbers:
1, 256, 32768, 32769, 128, 65534, 33153
The easiest way to do this is by using the operators for bit-masking and bit-shifting. Once you can correctly establish the page number and offset from an integer number, you are ready to begin. Initially, we suggest bypassing the TLB and using only a page table. You can integrate the TLB once your page table is working properly. Remember, address translation can work without a TLB; the TLB just makes it faster. When you are ready to implement the TLB, recall that it has only 16 entries, so you must use a replacement strategy when you update a full TLB. You can use FIFO for updating your TLB.

How to Run Your Program
Your program should run as follows:
```
./a.out 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 translates each logical address to a physical address and determines the contents of the signed byte stored at the correct physical address. Your program is to output the following values:

1. The logical address being translated (the integer value being read from `addresses.txt`).

2. The corresponding physical address (what your program translates the logical address to).
3. The signed byte value, which is stored at the translated physical address.

We also provide the file `correct.txt` that contains the correct output values for the file `addresses.txt`. You should use this file to determine if your program is correctly translating logical to physical addresses. Statistics After completion, your program is to report the following statistics:

1. `Page-fault rate`: the percentage of address references that resulted in page faults.
2. `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 do not reflect any memory access locality, do not expect to have a high TLB hit rate.