Santa Clara University

Computer Science & Engineering Department

CSEN 283: Operating Systems,

Winter 2024

Programming Project 4

Designing a Virtual Memory Manager

This project consists of writing a program that translates logical to physical addresses for a virtual address space of size 216 = 65,536 bytes. Your program will read from a file containing logical addresses and, using a TLB as well as 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. The goal behind this project is to simulate the steps involved in translating logical to physical addresses.

SPECIFICS

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) 8-bit page offset

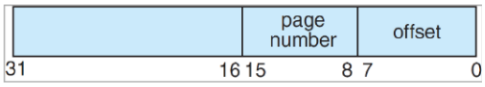

Figure 1. Address structure.

Other specifics include the following:
• 28 entries on the page table
• Page size of 28 bytes
• 16 entries in the TLB
• Frame size of 28 bytes
• 256 frames
• Physical memory of 65,536 bytes (256 frames x 256-byte frame size)

Additionally, your program need only be concerned with reading logical address 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 address using a TLB and page table as outlined in the

course's textbook Section 7.4. 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.

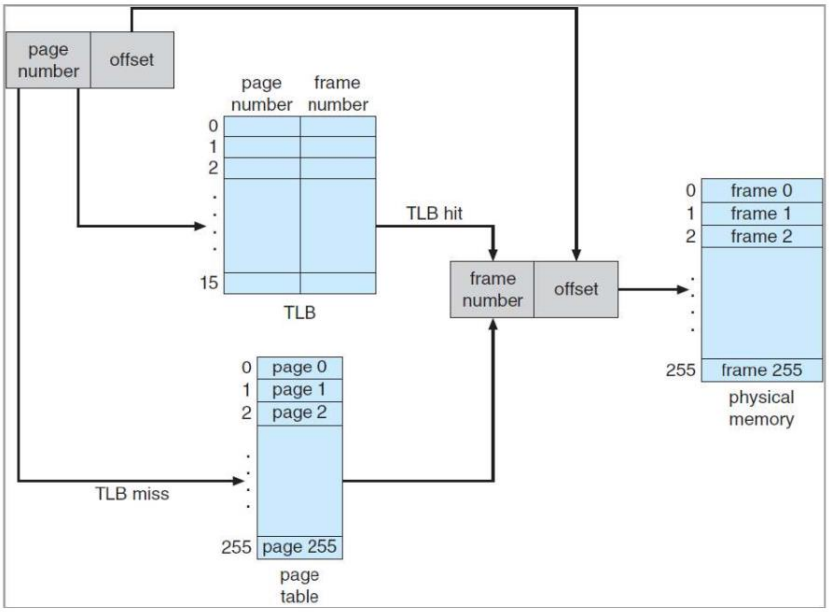

Figure 2 – A representation of the address-translation process

Test File:

You are provided the `addresses.txt` file, which contains integer values representing logical addresses ranging from 0 – 65,536 (the size of the virtual address space). You 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.

How to Begin:

First, write a simple program that extracts the page number and offset (based on Figure 1) from the following integer numbers:

1, 256, 32768, 32769, 128, 64434, 33153

Perhaps 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 that you bypass the TLB and use only a page table**. You can integrate the TLB once your page table is working properly (optional). 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 will need to use a replacement strategy when you update a full TLB. You may use either a FIFO or a LRU policy for updating you 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 1000 logical addresses ranging from 0 – 65535.
Your program is to translate each logical address to a physical address and determine the contents of the signed byte stored at the correct physical address. (Hint: in the C language, the char data type occupies a byte of storage, so we suggest using char values.)

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 stored at the translated physical address

What to submit

1. Submit source code file as project4.c (you will get points off if file names are incorrect)
2. Submit the output results and explanations of your code in a report file. You can copy paste your code in the file if you want**.**