Assignment 1

1. [5 marks] In this assignment, you will explore the /proc filesystem in linux. The /proc filesystem provides a
means to get and set various information about the kernel and about particular processes.

You have to first write a C program SystemInfo.c that will read the /proc file system and print out the following (with an appropriate message in each case):
1. The number of CPUs in your machine and their clock speed, number of cores.
2. The version of Linux kernel running on your system
3. The time in day:hr:min:sec when the system was last booted
4. The average load on the system in the last 15 minutes
5. The total usable and currently free memory in the system
6. The total swap space and the currently used swap space in the system
7. The swap partitions and their sizes
8. The time the CPU spent (over all processes) in the user mode and kernel mode
9. The number of context switches made by the system so far
10. The number of interrupts handled by the system so far

2. [5 marks] In this assignment, you simulate some page-replacement algorithms and compare their performances. Your simulations should be able to take in the required parameters from a data file in the following format:
•The first line of the file contains two integers in this order: the number of pages in the reference string, and the number of page frames.
•The rest of the file contains page reference string as a sequence of integers that are the page numbers (in virtual address space) accessed by the process in sequence.

The simulator will simulate the behavior of the following page-replacement algorithms on the reference string, andreport the number of page faults generated for each algorithm:
•FIFO
•LFU
•LRU

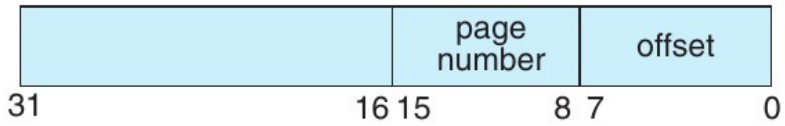
All the algorithms above are described in your text book. The simulator should take the following command-linearguments (in this sequence): the name of the data file, a sequence of strings (maximum three) from the following sets: FF, LF, LR (meaning the above three algorithms respectively). The simulator simulates only the algorithms specified inthe command line. If no algorithm is specified, all three are simulated.Submit the file page.c containing a main function, and the implementation of the above three algorithms. Each of the algorithms should be implemented as a separate function that is called from the main function.

Submit your programs by email to operatingsystemisi@gmail.com with the subject line as “MTechRollNumber_Assignment1” for MTech CS students and “Name_Assignment1” for JRF students by the beginning of the next lab session.

3. [Extra Credit] Designing a Virtual Memory Manager [Programming Project, Galvin Ch. 10]

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 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:



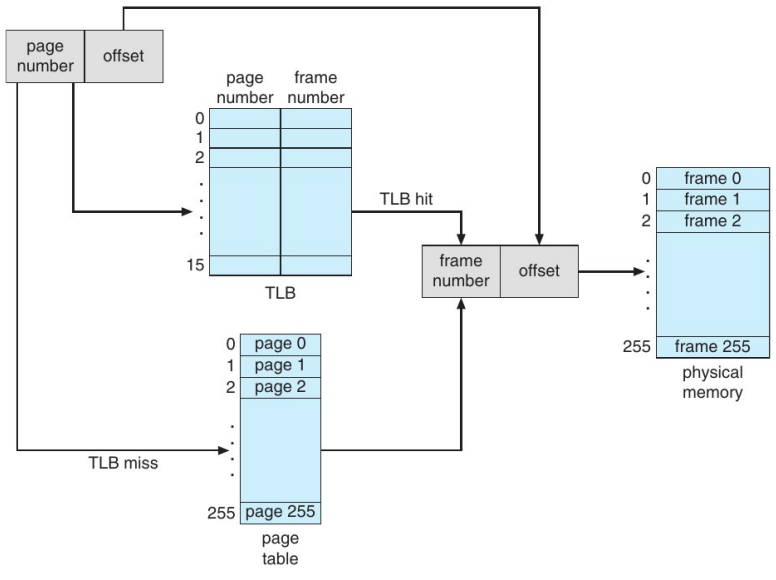
Other specifics include the following:

- 28 entries in 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 × 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:



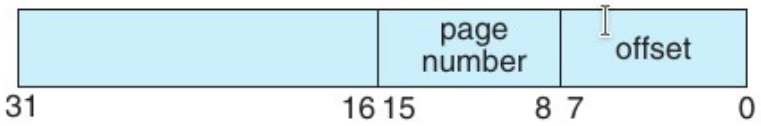
Handling Page Faults

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.

How to Begin

First, write a simple program that extracts the page number and offset based on :



from the following integer numbers: 1, 256, 32768, 32769, 128, 65534, 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. 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 an LRU policy for updating your TLB.

How to Run Your Program

Your program should run as follows:

./vmgr addresses.txt

Your program will read in the file addresses.txt, which contains 1,000 logical addresses ranging from 0 to 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. (Recall that 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.

We also provide the file correct.txt, which 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.