CS431 Operating Systems: Fall 2017 **Due: 11/12/2017 by 11:59PM**

Project 2
Virtual Memory Simulator
(100 points)

Introduction:
For project 2 you will implement a virtual memory simulator. This simulator consists of several important parts: the CPU which contains the memory management unit and the TLB cache, the virtual page table, physical memory, and the operating system. This system must be modular, that is, the aforementioned parts must be represented by at least one class each. **You are allowed to use Java, C++, or C# to implement this simulator.**

Implementation:
Data Structures:
You must use fixed-size arrays to simulate the page table, the TLB, the page, and physical memory. Physical memory will be a two-dimensional array to simulate the page-frame # and the page of byte-addressable,byte-sized data. The index of the elements of the page table will be the virtual page index (V-Page#). The virtual page index will be a field in the TLB. The TLB is small and must be scanned on every lookup. The arrays used to implement the page table and TLB will be arrays of data structures that represent the tables' entries. We will simulate the pages of data that must be loaded from disk with text files that will be provided with this project.

Table entry for the TLB:

V-Page#	V	R	D	PageFrame#

Table entry for the virtual page table:

V	R	D	PageFrame#

The CPU address width is 16 bits, physical memory's address width is 12 bits. The page offset is 8 bits. Assume the TLB contains 8 entries.

Note: you are not permitted to use any containers or data structures Provided by your chosen language other than simple arrays (not arraylists). You must implement the clock replacement algorithm's data structure as a linked list but you cannot use the Linked List structure provided by Java or C#.

Execution:
Your program will accept as a commandline argument. a file path to a text file populated with virtual memory addresses (in hex and also provided with this project) that are used to simulate memory accesses by a process. Your CPU will read them, hand them to the MMU for fetching or writing. If the address is preceded by a zero, then the MMU should only read and output the value. If the address is preceded by a one, then the address will be followed by a decimal value that needs to be written to physical memory. In the latter case, the page containing the data must have it's dirty bit set by the mmu in both the TLB and the page table. This page must be written back to disk if your page replacement algorithm decides to replace the file.

For each test file that you run, you should use an original copy of the page files that are available with the project. You do not want altered page files of a previous runs to be used. Keep the originals in a safe place and overwrite the working set on each simulation. This should be done programmatically from within your simulator, not manually. You should use a file copy facility of your chosen language, not an operating system call. You cannot assume that the operating system of the grader is the same as yours.

Output:
Your program will output the following information in a CSV file (with appropriate headers):

The address called for
Read or write (0 = read, 1 = write)
Soft miss (0 = no, 1 = yes)
Hard miss (0 = no, 1 = yes)
A hit (0 = no, 1 = yes)
If a page fault occurred, was the dirty bit set on the page that was evicted (0 = no, 1 = yes)
The value fetched if a read, or the value written if a write

Each output file should contain a total of the number of disk accesses, soft misses, hard misses, hits, hard disk reads and hard disk writes. You are not limited to the fields above, you can add more to make the calculation of the totals possible in a spreadsheet. There should be one output file for each input file.

Page Replacement:
We will keep the page replacement algorithm simple. Please implement the clock algorithm for page replacement. When a hard miss occurs, it's the job of the OS to replace the page and write the evicted page back to the drive if the dirty bit is set. The CPU should trap to the OS when these events occur.

The MMU is responsible to set the r-bit and the d-bit; the OS is responsible for unsetting them. The MMU should set the r-bit on a read or write of data. It should set the dirty bit on a write. The OS should unset the r-bits of all table entries after the CPU processes five instructions; it should reset the d-bit after a page has been written back to the disk by the OS.

When the MMU encounters a soft miss it will need to overwrite a record in the TLB (if it is full) with the entry of the page being called for. You can use the FIFO replacement algorithm for this.

Teams:
You are allowed to work in teams of up to three students. **Each student must submit a copy of the project.** The project must contain a statement listing the individuals on your team and whether or not they contributed equally to the project (a word document will be provided). A word of advice, choose your teammates wisely. If you cannot find someone that you trust to treat this project with the same enthusiasm as yourself then work on the project by yourself. It sounds like a lot but it's not that difficult if you approach the problem in an organized fashion and start work on it early.

If you choose to work on a team, your first order of business should be to meet and discuss the design of your program and the responsibilities of the members. You should set milestones for your work so that you hold each other accountable and you do not fall behind. Once you form teams, or if you will be working solo, one member of each team should email to me a list of your team members.

Turn in (via Blackboard):
1. Your source code; include a README file to explain how to compile and run your program. You cannot assume that the grader will use your IDE so give instructions on how to compile from the command line.
2. Your output files
3. Compress your submission into a single file.
4. If you are on team, include a completed "Team Member Scorecard" form.

Grading:
You will be graded on the modularity of your design and its correctness.The grader will test your program for correctness with a separate input file. Do not alter the provided page files to accommodate your program. Your design must demonstrate the responsibilities of the various components of this system and the modules similar to the real system discussed in class.