COMP 304: Project 3
Virtual Memory Manager

Due: Jan 11th, midnight

Notes: The project can be done **individually or teams of 2**. You may discuss the problems with other teams and post questions to the OS discussion forum but the submitted work must be your own work. **Any material you use from web should be properly cited in your report. Any sort of cheating will be harshly PUNISHED.** This assignment is worth 10% of your total grade.

Responsible TA: Ismayil Ismayilov iismayilov21@ku.edu.tr, Office: ENG 230

Description

In this project, you will get more familiar with the concepts of virtual memory.

Part I

In this part, you will implement a virtual memory manager similar to the one described in Programming Projects in page 447 in Chapter 9 in 9^{th} edition of the book (in Page 458 in Chapter 9 in the online version of the book, also provided on Blackboard).

The first version of the memory manager will be implemented with the assumption that physical address space is the same size as the virtual address space. Therefore you will not implement any page-replacement policy. You are given the base source code for the virtual memory manager in the "virtmem.c" file and you will complete its implementation by filling in the missing parts marked by **TODO** comments.

The virtual memory manager will use a TLB (Translation Lookaside Buffer) and a page table. You are required to use a FIFO replacement policy for the TLB. Differently from the specification in Programming Projects, you will use 20 bits addresses instead of 16 bits. The address bits will be divided into a 10-bit page number and a 10-bit page offset, which are also specified in "virtmem.c". In order to test your implementations, "addresses.txt" and "BACKING_STORE.bin" files are provided in the project folder.

Part II

The implementation in Part I assumes that physical memory is the same size as the virtual address space. In practice, physical memory (PM) is typically smaller than virtual memory

(VM). Part II will implement the case when VM is larger than PM.

Your implementation for Part II will use 256 page frames rather than 1024 for physical memory. This change will require modifying the provided program so that it keeps track of free page frames as well as implementing a page-replacement policy. We are asking you to implement both FIFO and LRU replacement policies. Add a command line argument to select a policy such as -p 0 for FIFO and -p 1 for LRU.

Compare these two policies with the address streams provided in the project folder.

Deliverables

You are required to submit the followings packed in a zip file (named your-username(s).zip) to Blackboard :

- Each team must create a Github repository for the project and add the TA as a contributor (username is *readleyj*). Add a reference to this repo in your REPORT. We will be checking the commits during the course of the project and during the project evaluation. This is useful for you too in case if you have any issues with your OS.
- Two .c source files that implement the virtual memory manager, one for each part. The names of the files must be part1.c and part2.c. Please comment your implementation.
- Report briefly describing your implementation
- Any supplementary files for your implementation (e.g. Makefile)
- You should keep your Github repo updated from the start to the end of the project. You should not commit the project at once when you are done with it; instead make consistent commits so we can track your progress. Otherwise a penalty may apply.
- Do not submit any executable files (a.out) or object files (.o) to blackboard.
- You are required to implement the project on Linux.
- Selected submissions may be invited for a demo session. Note that team members will perform separate demos. Thus each project member is expected to be fully knowledgeable of the entire implementation. Not showing up at the demo will result in zero credits.

Grading

Part I (50 points), Part II (50 points).

GOOD LUCK.