CSC369 Assignment 2 - Page Tables and Replacement Algorithms

June 2, 2020

1 Introduction

For this assignment, we're going back to the realm of user mode programming. Specifically, you will have to simulate the operation of page tables and page replacement. As I keep saying: the way to gain a solid understanding of the theory is by applying it in practice.

You have two tasks in this assignment, which will be based on a virtual memory simulator. The first task is to implement virtual-to-physical address translation and hldemand paging using a two-level page table. The second task is to implement four different page replacement algorithms: FIFO, Clock, exact LRU, and OPT.

Before you start work, you should complete the set of readings about memory, if you haven't done so already: link

2 Requirements

2.1 Setup

You will find the starter code HERE. It is your responsibility this time to add the code in your repository and make sure that you submit all the necessary files!
Note that you may be generating some large trace files and must NOT commit any of the trace files that you generate to your repository or you will run into serious problems with disk quota. Most of the trace programs should be familiar to you from the exercise in week 6. We have added a blocked version of matrix multiply, *blocked.c* which should exhibit fewer page faults under at least some of the page replacement algorithms. The Makefile shows you exactly how to compile and run the traces. Note that it takes quite a while to run the trace collection.

Compile the trace programs and generate the traces.

You may have noticed while doing the Exercise that the traces generated by Valgrind are enormous since they contain every memory reference from the entire execution. We have provided a program, *fastslim.py* to reduce the traces by removing repeated references to the same page that occur within a small window of each other while preserving the important characteristics for virtual memory simulation. (For example, a sequence of references to pages A and B such as "ABABABABAB...AB" are reduced to just "AB".) The runit script pipes the output of valgrind through this program to create the reduced trace. If you wish, you can experiment with fastslim.py to try omitting the instruction references from the trace or using a smaller or larger window (*fastslim.py –help*). You may also want to create traces from other programs, and you will definitely want to create small manual traces for testing.

2.2 Task 1 - Address Translation and Paging

2.3 Task 2

2.4 Write up