Adam Gincel

Operating Systems

I pledge my Honor that I have abided by the Stevens Honor System.

Homework 2 Analysis

Throughout this assignment I implemented 3 algorithms for handling Page Faults, emulating Virtual Memory Management. These algorithms were First-In-First-Out (FIFO), Least-Recently-Used (LRU), and Clock. In addition, I implemented Demand Paging and Pre-Paging for each algorithm. Below is a graph of the number of page faults across different page sizes using each algorithm, with and without prepaging. Here are some conclusions I drew about each algorithm:

FIFO:

First In, First Out is a very simple algorithm that is also pretty inefficient. It assumes the newest page is the most relevant one, and unloads the oldest one. However, this can frequently unload a commonly used page, which increases the number of page faults significantly over the long term. Prepaged FIFO seems to do a bit better, especially on smaller page sizes, but still performs relatively poorly compared to the other algorithms. FIFO was probably the easiest of these to program.

LRU:

Least Recently Used is a significant improvement over FIFO. The only real difference between this algorithm and FIFO is that while FIFO removes a Page based on when it was loaded, LRU removes a page when it was used last; effectively updating its timestamp every

time it's accessed, instead of only on load. This keeps commonly used pages in memory longer, reducing page faults. It still can unload commonly used pages if enough pages are between uses to the most popular ones, so it isn't perfect. LRU was only a slightly modified FIFO, so it wasn't terribly difficult to implement once FIFO worked. For some reason my LRU Prepaged gets much lower numbers than my other algorithms, with or without prepaging, I'm inclined to believe this is due to some sort of error on my part, unless LRU Prepaged is simply amazingly efficient.

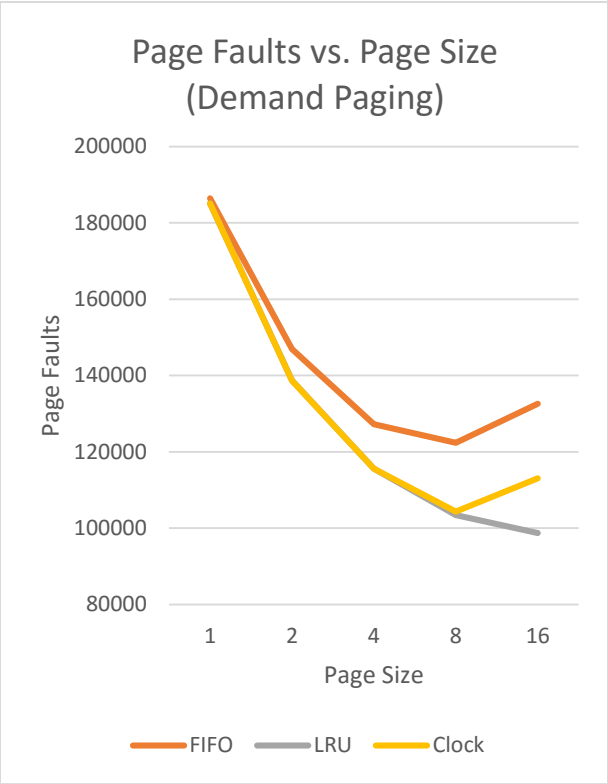
Clock:

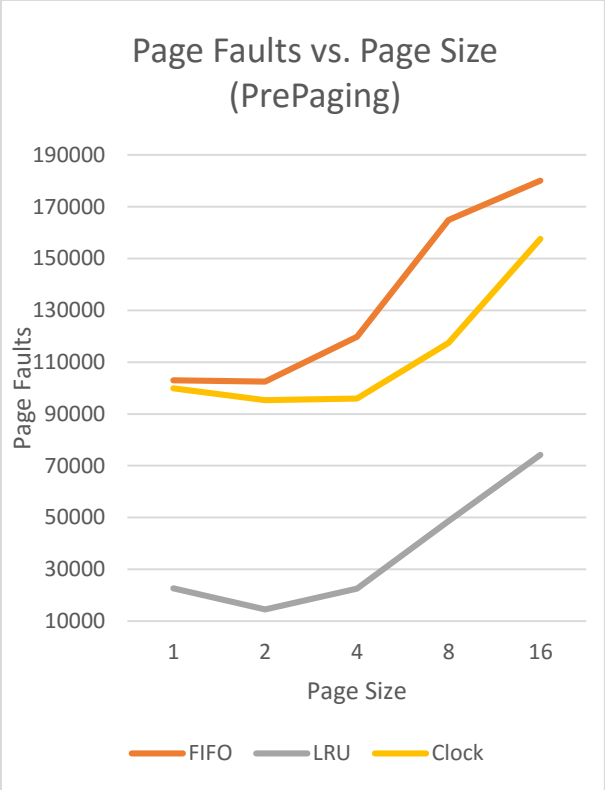
The clock algorithm attempts to improve on both FIFO and LRU by adding an extra "clean or dirty bit" R which effectively gives each page a "second chance" before being removed. Second Chance is an algorithm on its own, but Clock improves complexity by having a hand modularly go through the Page array, instead of constantly reorganizing the Pages. Clock was implemented entirely differently than FIFO or LRU, and was conceptually difficult to understand at first, but it actually has the shortest implementation by lines of code of the three algorithms.

If the Trace was random:

All of these algorithms attempt to achieve greater efficiency and minimize page faults by using the assumption that there are specific pages that will be needed more often than others. If the trace was completely random, that would no longer be true, and each algorithm would have an effectively random, but most likely high, number of page faults, and none would any tangible benefit over the others.

Found on the next page are the graphs of each algorithm, with and without prepaging.


Page Faults vs. Page Size
(Demand Paging)
200000
180000
160000
140000
120000
100000
80000
Page Faults
1
2
4
8
16
Page Size
FIFO
LRU
Clock


Page Faults vs. Page Size
(PrePaging)
190000
170000
150000
130000
110000
90000
70000
50000
30000
10000
Page Faults
1
2
4
8
16
Page Size
FIFO
LRU
Clock