Christopher Torralba
CS 3800
Section B

Assignment 2

My hypothesis for the output is that the first-in-first out replacement algorithm is going to have the most page faults. I also believe that the clock and the least recently used replacement algorithms should be roughly the same as they update after being accessed in memory. In terms of adding another page due to prepaging, I hypothesize that bringing in an extra page would lessen the number of page faults. There is less overhead to bring pages that reside contiguously on the disk though.

The complexity of implementing FIFO was generally simple but it wasn't worth the number of page faults it produced. With FIFO the only thing that someone needs to keep track of is the location of the oldest page and since pages are being brought in sequentially this makes it even simpler as it goes around. I believe that LRU was the best in terms of complexity for the number of page faults it generated. LRU has more complexity than FIFO as there are more locations to keep a track of. When a page gets accessed it gets updated as the newest page in memory, but the algorithm still keeps the simplicity of finding the oldest page in memory. Although the clock-based implementation was the hardest out of all of them to implement, it still produced results that was similar to LRU. With clock-based replacement method there needs more things to be implemented in comparison to LRU and FIFO. The clock hand, the used bits, determining when a page was last accessed, and replacing the oldest page based on used bit was a lot to keep track of.

Prepaging had a major influence dependent on the page size. If the page size was generally more smaller, prepaging lowers the number of page faults in comparison to demand paging. Whereas the page size was larger, such as sixteen pages, it produces more page faults

than demand paging. In the simulation that I created prepaging produced, roughly, the same number of page faults as demand paging.

My results supported my hypothesis to some extent. The FIFO replacement algorithm produces more page faults than LRU and clock replacements algorithms in most cases. My hypothesis of clock and LRU replacement algorithms being similar to each other was also supported although judging from the data, as the page size grows the difference between LRU and clock replacement cycles become more evident. What I didn't expect to see from prepaging was the difference it makes with various page sizes. I only expected that prepaging lowers the number of page faults. My hypothesis for prepaging was not supported since I didn't expect to see higher page sizes producing more page faults. If the data was random then FIFO would still produce the same results, as the the data is kept in track due to time. There would be changes in LRU and clock, however, because it keeps track of the last page that was accessed.

Page Size:	1	2	4	8	16
Demand					
LRU	116886	87572	72551	65273	61705
Clock	117042	87627	72539	65327	64824
FIFO	117879	92894	80154	77382	84232
Prepaging					
LRU	62885	59406	58586	59036	104438
Clock	63001	59414	58677	59257	120072
FIFO	63656	62605	66708	79006	119921