**Assignment 2 Report**

**Hypothesis** FIFO is simple to implement but does not take advantage of the impact coming from the frequency a particular page is referenced thus should be result in the most page faults. LRU keeps track of the least referenced page and keep frequently referenced pages in memory for fast access resulting in much less page faults. CLOCK method is a compromise between the two, it utilizes page reference at the same time make each page a fair chance to get replaced, resulting in also less page faults.

**Effects of pre-paging** Pre-paging in most real life applications may result in great performance increases for any page replacement method utilized in the application, it greatly exploits the principle of locality in typical program applications.

**Results** The following graph shows the result of page size vs. page faults for each page replacement algorithm/paging combination:

As the graph demonstrates, as page size increases, the page faults of all algorithm/paging combination decrease, and in general enable pre-paging decreases page faults in any replacement algorithm.

**Complexity vs Performance** FIFO is easy to implement but requires extra hardware/software resource to buffer timestamp for each frame. LRU is also easy to implement, also requires timestamp buffer as well as search algorithm. CLOCK is more difficult to implement than the previous two methods but requires less resource, only needs a 1-bit buffer for each frame compared to larger timestamp buffer. Pre-paging is easy to implement and provide great performance gain/complexity.

**Random Memory Access** In case of random memory access trace, pre-paging would have much less/no significant impact on performance, performance gap between FIFO and LRU/CLOCK would decrease since no apparent page access pattern would exist, resulting less performance gain from FIFO to LRU/CLOCK. The performance impact of page size would likely to remain the same. LRU/CLOCK performance may decrease for applications with less loop structure and more linear process due to the randomness downgrade the potential performance gain from these replacement methods in programs with those properties.