In this project, Dylan and I learned about the interaction between physical memory and the virtual memory system. We constructed and compared two replacement algorithms to manage the limited physical memory: First-In-First-Out (FIFO) Replacement and Clock Replacement. We implemented the following three interface functions in our program.

**void mm\_init(void\* vm, int vm\_size, int n\_frames, int page\_size, int policy);**

This initializes the memory management system, where we use this call to initialize our datastructures and perform other set up operations. 'vm' denotes the pointer to the start of the virtual address space that needs to be managed, 'vm\_size' denotes the size of the virtual address space in bytes (starting from \*vm), 'n\_frames' denotes the number of physical pages available in the system, 'page\_size' denotes the size of both virtual and physical pages, and, 'policy' can take values 1 or 2 -- 1 indicates fifo replacement policy and 2 indicates clock replacement policy. The virtual memory (vm\*) is already allocated and page aligned when this call is made.

**unsigned long mm\_report\_npage\_faults();**

This function returns the cumulative number of page faults incurred since the invocation of mm\_init.

**unsigned long mm\_report\_nwrite\_backs();**

This function is similar to mm\_report\_npage\_faults(), except that it reports the number of pages that were written back to swap space upon eviction from physical memory.

Here is a description of the two page-replacement algorithms that we implemented in our signal handler.  
  
**First-In-First-Out Replacement:**

This policy evicts the oldest virtual page brought in to the physical memory. We will protect the pages that are not in physical memory to catch accesses to these page and record page faults. When we bring in a page, we mprotect() it, in Read-Only mode and when we evict a virtual page, we mprotect() it, so that any future access to it will raise a SIGSEGV, and we will record it as a page fault.

**Clock Replacement:**

In this algorithm, we maintained a circular linked list, with the head pointer pointing to the next potential candidate page for eviction. We maintained a 'Reference Bit' with each virtual page. When looking for the next candidate page for eviction the head pointer resets this bit and moves to the next element in the circular list and retries. When the head finds a candidate with 'Reference Bit' set as zero, we mprotect a future reference to this page to record a page fault. The new virtual page that is brought in, will be put in this slot in the circular list (where the head is pointing), its reference bit will be set to 1, and the head will be advanced to the next element in the circular list.

Contributions

Dylan Nguyen: mm\_init, Clock Replacement policy

Jacob Klapper: FIFO Replacement policy