Zach Martin

Cenek

CSCE A320

December 10, 2017

Analysis of VM Program

To test the functionality and performance of my virtual memory simulator I created three test scripts:

The first one: had a bus size of 16 bits, had 4 bits to represent the virtual page, and 2 bits to represent the number of physical page frames present in the MMU table at one time. In this example, there were 16 possible virtual pages, but only 4 physical page frames. As a result, once the table was full, there is a 75% chance that any given request would result in a page fault. I tested to make sure that once the table was full, the page that would be evicted was the first present page starting from the bottom because I had each request as a read.

For the second test, I used a bus size of 6 bits and used 3 bits to represent both the virtual and physical pages. Since I used the 3 upper most bits of the request to represent the virtual page, I had a total of 8 possible virtual pages. In addition, I have 8 possible physical pages due to 3 bits being used to represent the physical page number in the MMU table. Because these two are equal, there is enough room in the MMU for there to be a mapping from each virtual page to a physical page. As a result, the table will only have 8 page faults, one for each virtual page initially, but then there will be no more page faults. This was confirmed by testing each possible request allowed by a 6 bit bus (0-63).

The performance of the virtual memory simulator is tied to the ratio of bits used for virtual memory page to the bits used for physical pages. Assuming a full MMU table, a difference of 1 bit between virtual pages and physical pages leads to a ½ chance of a page fault, each additional bit difference multiplies this by another ½. This geometric series continues with each additional bit difference between the two pages. So performance takes a bit hit initially and then decreases slower, but there will always be a considerably high chance for a page fault given any number of difference. No difference would imply that there is no need for virtual memory in the first place, so this implementation will guarantee at least a 50% page fault chance given that each request is random. In practice, this number should be much lower as data is usually accesses in sequence in chunks rather than purple random addresses.

My final test script makes sure that the replacement algorithm ignores pages that have been written to on the first pass through, but then chooses the first page if all have been written to.