Operating Systems, Program 3
D-shell Shared Memory

Daniel Nix

May 1, 2015

Program Description The MMU-Simulation program is a simple implementation of the memory management unit (mmu) that simulates the relocation register, paging, and paging with a Transition Look-Aside Buffer (TLB). The purpose of this programe is to simulate the different methods of mmu.

- Creates a section of memory to run in or pages for paging and the tlb
- Creates a offset to access the memory by and a page for paging and the tlb
- Accesses physical memory with the given offset or a page with given offset
- Successful if offset doesn't overstep bounds of max or frame size

1 Algorithm

1.1 Event Loop

The event loop is handled in main and uses an array of function pointers to call the relocation register, paging, and tlb functions. The event loop gets the user input with 1 being relocation register, input of 2 is paging, input of 3 is tlb, and 4 exiting the program. Any other number input causes an error message to be displayed and redisplays the prompt.

1.2 Relocation Register

To implement the relocation register a logical max is created. Then the offset for physical memory is created which will be used as the relocation register. Last a offset to access memory is created and used to see if it stays within the bounds of the physical min and max of memory.

1.3 Paging

To implement paging the user is asked to enter how many frames that are to be implemented, how many pages to implent with pages being less than or equal to the number of frames. Next page and frame size is asked to be a power of two. The page table is then displayed to the user so they know what page is referencing what frame. Once everything is set up the algorithm starts creating random pages and random offsets that are used to access physical memory. If the offset is less then the page size the frame was accessed successfully, if it wasn't then it is checked if it goes into another frame or out of physical memory. This occurs a number of iterations that the user specifies.

1.4 Transition Look-Aside Buffer

Creating the TLB requires:

1. Pages Table and Frames
2. TLB Table
3. TLB Replacement Algorithm

Pages and Frames are created using the same scheme in the Paging section except that the algorithm is not used just the creation of the pages and frames.

To create the TLB table the pages and frames are selected randomly to fill the TLB table. The Size of the TLB table should be less then the

number of entries in the page table. The associated frames of the pages stored in the TLB are also stored in the TLB frame portion.

The user is then prompted with what replacement algorithm they would like used on the TLB if there is a miss when accessing it. The choices are random, round-robin, and least recently used.

1.4.1 TLB Replacement: Random

The Random replacement algorithm for the TLB is a straightforward approach. The program will randomly select an index to replace in the TLB table with the page that was missed and then replace that entry with the page and associated frame.

1.4.2 TLB Replacement: Round-Robin

The round robin replacement algorithm is a little more involved than the random replacement with its book keeping of what pages are in the que. A que is created that will store all the pages and use the 0 index as the exit point of the que and the last index as the entry point after it is removed from the TLB. When The TLB is first created the pages that are currently in the que are put at the end of the que since every page should get equal priority of going in TLB and que to be in the TLB. If a TLB miss occurs the 0 index in the TLB will be removed and added the end of the que with everything in the TLB shifting up one and the page in the 0 index in the que is put in the last index of the TLB. The que is then shifted down and the program continues.

1.4.3 TLB Replacement: Least Recently Used

The least recently used replacement algorithm keeps track of how often a process is used. The more often it is used the more chances it has to stay in the TLB and not get removed and replaced when a miss happens. If all pages have been used the same number of times the random replacement algorithm is selected to do the TLB replacement. If a few are equally least used the first found will be selected to be replaced.

2 MMU-Simulation Testing

Unit testing for the MMU-Simulation program was done in an iterative process. Each time code was added to the shell it was tested to confirm the new code performed the desired function and did not break existing memory magement unit functions.

3 Submission Description

3.1 Compiling Instructions

3.2 External Functions