Jason Valenzuela

CS 4414 Project 3 Write-Up

This project is comprised of 3 data structures (TLB, page table, physical memory), and reads input from the addresses.txt file and stores them in memory. If there is a page fault, the program will read from the BACKING_STORE.bin file and allocate all necessary information into physical memory. The three main components of this virtual memory manager are the translation look-aside buffer (TLB), page table, and physical memory.

Translation Look-aside Buffer:

The TLB is the first thing that the virtual memory searches when an address is loaded. The TLB is an array of 4 tlbBlock structs, which contains two int fields: page and frame. The current address is searched within the TLB, and if it is found, the corresponding frame table is returned and displayed to the user. If not, the page table is searched with the current address.

TLB entries are placed into the data structure using the first-fit algorithm, and this is done by checking a boolean array which maps to the free entries of the TLB. If an index of the array returns true, then that index in the TLB is free and an entry can be placed there. If it returns -1, a page must be replaced using the FIFO replacement algorithm. This is done with an integer pointing to a victim entry in the table. If a replacement needs to occur, the victim entry is removed from the list and the new entry is added to the TLB. The pointing integer is incremented to point to the next victim entry, and set to zero if it is pointing to the last element of the TLB.

Page Table:

The page table is an array that consists of 16 pageBlock structs. Each pageBlock struct contains 4 fields: index integer (gives the current index of the block), page integer (holds the corresponding frame number), inmem boolean (indicates whether or not the page is in memory), and counter unsigned integer (a “clock” that counts the number of times a page has gotten a hit).

The placement strategy is somewhat similar to the one used for the TLB. Once an address has been hashed into the page table according to its page number, it checks the inmem boolean to check if that page is in memory. If the page is in memory, the corresponding frame number is returned and displayed to the user, and the counter field is increased with a boolean OR, and sets the most significant bit of the integer to 1. Before the next address is loaded into the program, the counter field is shifted left by one.

If the page is not in memory, the first-fit algorithm is used to place the page into physical memory. This is done by checking the free frame list, which is an boolean array which maps to the physical memory frames. If an index of the array returns true, the page is entered into the corresponding index, the page and frame are entered into the TLB, and the next address is loaded into the program. If all indices of the free frame array return false, then there are no free frames in physical memory and a page must be replaced. This is done by using a variation of the least recently used (LRU) algorithm. The program checks the counter fields of each page that is currently in memory for the smallest counter field. Once the smallest one has been found, it is cleared from memory, and the current address is added to memory. The TLB is cleared from the page table entry that was evicted, and the new value is added to the array.

Physical Memory:

The implementation for physical memory was relatively simple: it was an integer array of integer arrays. One axis of the data structure mapped to the 8 frames that are available, and the other axis was the data portion of the physical memory, where all the data retrieved from the backing store was placed.