Project 3: Memory Simulator Report

Authors: Brian Nguyen and Charles Snead

Summary

For the memory simulator project, we implemented logical-to-physical memory address translation, memory access with TLB first and page table lookup, and page replacement using FIFO, LRU, and optimal algorithms. These algorithms came into use when the user inputted number of frames was less than the number of logical addresses, which was always 256. All of this was implemented in Python, making use of a LogicalAddr class, and multiple list-like data structures to represent the TLB, page table, and physical frames.

Objects

LogicalAddr Class

Contains two attributes: the page number and the page offset. These attributes are populated by reading in from the addresses.txt file and doing the necessary bit masking and shifting operations to find the page number and offset from the logical address. The lower 8 bits were found with a bit mask to determine the address offset. The upper 8 bits were determined using bit shifting by 8 bits to the left to find the page number.

Functional Units

TLB

The TLB uses a two-dimensional list where the inner list contains two values which are the page number and the frame number. Both values are initialized to -1 at the beginning. The outer list contains 16 of these inner lists, representing the 16 total entries of the TLB. The TLB serves as the first lookup table when reading in addresses from addresses.txt and if the page number is found in the TLB and the page is located in the physical frame, we consider this a TLB hit. If it is not found or loaded, we resort to the page table to determine if a page fault has occurred. If a page fault has occurred, the TLB is updated using the FIFO algorithm to insert or replace a new page number and frame number pair based on the PRA currently in use.

Page Table

The page table is a two-dimensional list where the inner list contains two values which are the loaded bit and the frame number. The loaded bits are initialized to 0 and the frame numbers are initialized to -1. The index into the outer represents the page number. The outer list contains 256 of the inner lists, which represents 256 total pages. The page table serves as the second lookup table when reading in addresses from the addresses.txt file. When indexing into the page table, if the loaded bit is not set to 1, we consider this a page fault. If a page fault has occurred, the page number in the page table is updated to have a new frame number based on the PRA currently in use, and the loaded bit is set to 1. Given the specifications, we assume the page table will always contain all loaded and unloaded page entries.

Physical Frame Table

The physical frame table is a one-dimensional list containing a user-inputted number of 256-byte chunks. If the user does not specify the number of frames, the default number is set to 256 frames. The 256-byte chunks are represented in Python using a *bytearray* data structure to store binary strings as the page content. The page content is being read in from the BACKING_STORE.bin file every time a page fault occurs and loaded into a physical frame. If the size of the physical frame is smaller than the page table, we use the user-inputted page-replacement algorithm to replace old page content with new page content when a page fault occurs.

PRA Implementation

FIFO

The FIFO logic was implemented using modulo arithmetic to ensure that the index resets to 0 once the maximum number of frames has been reached.

LRU

The LRU logic makes use of a global list to keep track of previously accessed page numbers. As we read in page numbers, if the page number has already been accessed, it gets appended to the back of the list (highest index) so that when a page fault occurs, it pulls from the 0^{th} index to get the least recently used page number. This least recently used page number is what is unloaded from the physical frame table, page table, and TLB so that the new page content can be loaded into the physical frame table and the page number and frame number can be updated in the page table and TLB.

OPT

The OPT logic makes use of a global list to keep track of "future" page numbers to be accessed. Since we have already read in all the logical addresses before the simulation begins, we can populate the global list with all the page numbers that will be accessed when the simulation begins. As we access pages and a page fault occurs, we look into the global list and find the furthest out page number that can be replaced within our physical frame table. This furthest away page number is what is unloaded from the physical frame table, page table, and TLB so that the new page content can be loaded into the physical frame table and the page number and frame number can be updated in the page table and TLB. If a page fault does not occur, we still pop off the 0^{th} index of the global list to ensure that the future values to be accessed are up to date.