Alex Zaharia, Triet Luu Khoi Tran

Project 3: Virtual Memory Simulator (MemSim)

This project is a virtual memory simulator that models the TLB, page table, backing store/disk device, physical memory, frames, reference sequences, and three page replacement algorithms (fifo, lru, opt).

The TLB class manages a cache of 16 entries where each entry maps a virtual page number to a physical frame number. It makes sure entries stay within the range of 0-255 using the _validate_page_number function. The *add_entry* function adds a new mapping to the TLB, automatically removing the oldest tuple if the TLB is at capacity using the FIFO replacement policy. The *get_frame* function searches for a page number in the cache and returns the corresponding frame number if found. The *remove_frame* function deletes a tuple based on the page number, ensuring the cache does not hold outdated mappings.

The PageTable class in the Virtual Memory Simulator is a fixed table of 256 total pages, each holding a tuple with a frame number and a loaded bit (flag) for whether or not the page is currently loaded into physical memory. The class also has the *_validate_page_number* function to make sure any page number operations fall within the valid range of 0 to 255. The *add* function uses this validation and inserts or updates an entry in the table with the physical frame number and its loaded bit. The *get_frame* function retrieves the mapping for a given page number, if it exists. The *update_page_status* function updates the load status of a page and the *remove* function clears an entry in the table (sets it to None).

The PhysicalMemory class manages a fixed list of frames (depending on how many are specified in the command line), each capable of storing one page of data. The *is_full* function checks if all frames are occupied. The *load_page_into_frame* function searches for the first unoccupied frame, loads the given page data into it, marks it as occupied, and returns the frame index.

The reference sequence is initially loaded from a file and converted into a series of integers that represent the logical addresses to be translated. The backing store data is also read in its binary format and parsed. Then, the page replacement algorithms are set up: FIFO and LRU use queues, while the OPT algorithm is represented as a function that evaluates the upcoming addresses against those currently in memory and replaces the address that will be used farthest in the future.

In the main function of the simulator, the process begins by parsing command-line arguments to determine the number of memory frames and the specific page replacement algorithm to use, defaulting to 256 frames and the FIFO algorithm if these parameters are not provided. The program then proceeds to read the reference sequence and load the binary data from the backing store. During, this setup phase the TLB, page table, and physical memory are all also initialized.

Throughout the simulation, each logical address from the reference sequence is processed and translated into a physical address using the TLB and page table. The simulator also handles page faults and TLB misses based on the specified page replacement strategy. At the end of the simulation, the total number of page faults and TLB hit rates is printed to the console.