**Mid-Sem (A Group Mini Project): Address Spaces, Address Translation, and Paging in C**

Jonathan Kuug

Randy E. Fiatsi

Dawud Swallieu

Department of Computer Science, Ashesi University

Dr. Kavu Tatenda

March 1, 2024

### **Memory Management System Report**

#### *Introduction*

This report details the design, implementation, analysis, and optimization of a simulated memory management system. The system simulates key aspects of virtual memory management, including multi-level paging with a Translation Lookaside Buffer (TLB), page table management (hierarchical 2 level) , and physical memory frames (dram), in a controlled environment using C programming language.

#### *Design*

The design of the simulated memory management system incorporates several key concepts of operating systems, focusing on the efficient management of virtual and physical memory. The system is designed to demonstrate the principles of virtual memory management, including multi-level paging, address translation, page allocation and deallocation, handling page faults, and the use of a Translation Lookaside Buffer (TLB) for fast address translation. The design considerations are outlined below:

*Virtual and Physical Memory Spaces*

* **Size Configuration:** The system defines both virtual and physical memory spaces as 2^16 KB, aligning with the common architecture of a 16-bit address space. This size supports a comprehensive simulation environment.
* **Page and Frame Sizes:** Set at 256 bytes, this configuration facilitates the division of memory into pages (virtual memory) and frames (physical memory), allowing for a detailed simulation of paging mechanisms.

*Multi-level Paging System*

* **Page Directory and Page Tables:** A two-level paging hierarchy is utilized, where the page directory contains pointers to page tables, and page tables, in turn, map to physical memory frames. This design is pivotal in managing large address spaces efficiently. Page tables are persisted to disk (file binaries) when not in use to simulate memory efficiency and optimization.
* **Address Decomposition:** Virtual addresses are decomposed into multiple parts: a page directory index, a page table index, and an offset. This structure is crucial for navigating the multi-level page table system during address translation.

*Translation Lookaside Buffer (TLB)*

* Purpose: The TLB serves as a fast-cache memory that stores recent translations of virtual page numbers to physical frame numbers, significantly speeding up the address translation process.
* Replacement Policies: The system implements FIFO (First-In-First-Out) and LRU (Least Recently Used) policies to manage TLB entries, demonstrating their impact on the system's performance.

*Physical Memory Simulation (DRAM)*

* Representation: Physical memory frames are simulated using a dynamically allocated two-dimensional array, where each row represents a frame and columns represent bytes within a frame.
* Functionality: The simulation includes mechanisms for loading data into frames from a simulated backing store, tracking allocated and free frames, and mapping virtual pages to physical frames.

*Address Translation Mechanism*

The cornerstone of the system is the address translation mechanism, which converts virtual addresses into physical addresses. This process involves:

* **TLB Lookup:** Initially, the system checks the TLB for a quick translation. If the translation is found (TLB hit), it proceeds directly to access the physical memory. If not (TLB miss), the system looks up the page directory and page tables.
* **Page Directory and Page Table Lookup:** On a TLB miss, the system uses the page directory index to find the corresponding page table and then uses the page table index to find the frame number. This step may trigger a page fault if the page is not in memory.
* **Handling Page Faults:** When a page fault occurs, the system loads the requested page from the simulated disk (backing store) into a free frame in physical memory and updates the page table and TLB accordingly.

*Memory Management Operations*

* **Page Allocation and Deallocation:** While explicit functions for page allocation and deallocation are not detailed in the initial implementation, the system is designed to allocate pages dynamically during page faults and could be extended to include explicit deallocation as part of process termination or memory management optimization.
* **Simulation of Memory Access Patterns:** The design includes the simulation of processes accessing memory through randomly generated virtual addresses, demonstrating how the system handles address translation, page faults, and TLB updates in a dynamic environment.

*Implementation*

The implementation part of the memory management system simulation involves several critical components, each designed to mimic aspects of an operating system's handling of virtual memory. Here, we delve deeper into the specifics of how each component is implemented and interacts within the system.

*Data Structures*

* **Translation Lookaside Buffer (TLB) (`vmTable\_t`):** The TLB is a cache that stores a limited number of mappings from virtual pages to physical frames. It is implemented as a structure containing arrays for page numbers, frame numbers, and possibly entry ages (for the LRU policy). It speeds up the translation process by avoiding the need to walk the page table for frequently accessed memory pages.

* **Page Directory and Page Tables (`vmTable\_t`):** The page directory is the top-level structure in the multi-level paging system, containing pointers to page tables. Each entry in the page directory corresponds to a page table, which in turn contains mappings from virtual pages to physical frame numbers. This hierarchical structure allows the system to manage a large virtual address space efficiently.
* **Physical Memory (DRAM) (`int dram`):** Simulated as a two-dimensional dynamic array, where each row represents a physical frame and each column represents a byte within that frame. This structure stores the actual "data" that would be present in physical memory.

*Address Translation Process*

* **Virtual Address Decomposition**: Each virtual address is decomposed into several components: the page directory index, the page table index, and the page offset. These components are extracted using bitwise operations based on predefined mask and shift values.

* **TLB Lookup:** The system first checks the TLB for an existing mapping of the virtual page to a physical frame. If a mapping is found (TLB hit), the translation is quickly completed. If not (TLB miss), the system proceeds to the page directory.

* **Page Directory and Page Table Lookup:** The page directory index is used to locate the corresponding page table. Then, the page table index is used to find the frame number. If the page table or the mapping does not exist, a page fault is triggered.

* **Handling Page Faults:** On a page fault, the system simulates reading the requested page from a backing store (disk) into physical memory. A free frame is allocated for this page, and the page table (and possibly the page directory) is updated with the new mapping. This step may involve allocating a new page table if the required one does not exist.

* **Updating the TLB**: After resolving a TLB miss (whether directly from the page table or after handling a page fault), the new mapping is inserted into the TLB using the chosen replacement policy (FIFO or LRU).

*Page Allocation and Deallocation*

* **Allocation**: Pages are allocated on demand. When a page fault occurs because a requested page is not in memory, the system simulates loading the page from disk into a free frame in physical memory. The page table and TLB are then updated to reflect the new mapping.
* **Deallocation**: While not explicitly detailed in the initial implementation, deallocation would involve removing a page's mapping from the page table and TLB, then marking its frame in physical memory as free. This process is crucial for long-running systems to prevent memory leaks and ensure efficient memory usage.

*Memory Access Simulation*

The simulation involves generating random virtual addresses and translating them into physical addresses using the system's address translation mechanism. This process simulates the memory access patterns of processes, allowing for the analysis of the system's performance in terms of TLB hit rate, page fault rate, and overall memory utilization.

*Analysis*

The system's performance is analyzed based on memory utilization, page fault rate, and TLB hit rate under various simulated conditions.

* **Memory Utilization** - Memory utilization measures the percentage of physical memory frames that are in use at any given time. It's a critical metric for understanding how efficiently the memory management system is using the available physical memory. High memory utilization indicates that most of the physical memory is being used, which could lead to fewer page faults if most accessed pages are already loaded into memory. However, it could also lead to thrashing if there isn't enough free memory to accommodate the working set of the active processes. Analyzing memory utilization involves tracking the allocation and deallocation of frames to see how the system's memory usage changes over time.
* **Page Fault Rate** - The page fault rate is the ratio of access operations that result in a page fault to the total number of memory access operations. A page fault occurs when a process attempts to access a page that is not currently loaded into physical memory, necessitating a fetch from the simulated disk (backing store). The page fault rate is a direct indicator of the effectiveness of the memory management strategies employed by the system. High page fault rates may indicate inadequate memory allocation, poor prediction of page usage, or simply an access pattern that doesn't align well with the system's paging strategy. Reducing the page fault rate is often a key goal in optimizing memory management systems.
* **TLB Hit Rate** - The TLB hit rate measures the effectiveness of the Translation Lookaside Buffer in speeding up the address translation process. It is calculated as the ratio of address translations that were resolved by the TLB to the total number of address translation requests. A high TLB hit rate suggests that the TLB is effectively caching frequently accessed page translations, significantly reducing the time required for address translation. Conversely, a low TLB hit rate indicates that the system is often required to traverse the multi-level page table structure to resolve address translations, which is more time-consuming.

*Analysis Methods*

* **Workload Variation** - The system's behavior is analyzed under different workloads to understand how various types of memory access patterns affect performance. Workloads can be varied by changing the sequence of memory accesses, the working set size of the processes, and the degree of locality of reference. This variation helps identify how well the system adapts to different operational scenarios.
* **Configuration Testing** - The impact of different system configurations on performance is analyzed by varying parameters such as the size of the TLB, the page size, and the replacement policies used for the TLB and page frames. This analysis helps identify optimal configurations for different types of workloads and operational conditions.
* **Statistical Analysis** - Performance metrics are collected and statistically analyzed to provide insights into the system's behavior. This includes calculating average rates (e.g., average page fault rate, average TLB hit rate) and analyzing variance in performance metrics under different conditions. The goal is to identify trends, patterns, and outliers that can inform decisions about system design and optimization

#### *Optimization*

The system's optimization focuses on reducing page faults and improving TLB effectiveness.

* **TLB Replacement Policies**: Comparing FIFO and LRU policies to determine their impact on the TLB hit rate.
* **Page Size Adjustment**: Exploring the effects of different page sizes on memory utilization and page fault rate.
* **Pre-Paging and Demand Paging**: Considering strategies for loading pages into memory before they are needed to reduce page faults.

#### **Conclusion**

The memory management system simulation provides a detailed look into the workings of virtual memory systems, highlighting the challenges and considerations in managing memory effectively. Through the design and implementation of this system, we gain insights into the trade-offs between memory utilization, access speed, and system complexity. The analysis and optimization steps underscore the importance of carefully chosen algorithms and policies in achieving efficient memory management. This project not only serves as an educational tool but also as a foundation for further exploration and development in the field of operating system design.