Memory Management Simulator

Luvpreet Singh

December 31, 2025

GitHub Repository: `github.com/Luvpreet042004/Memory-Management-Simulator`

Contents

1 Overview

This document describes the design and implementation of the backend of the Memory Simulator project. The simulator models physical memory allocation, cache hierarchy behavior, and access timing. The goal is educational: to demonstrate memory management and caching concepts rather than provide a real operating system implementation.

2 Memory Layout and Assumptions

2.1 Physical Memory Model

Physical memory is modeled as a contiguous, byte-addressable array of fixed size N. Memory is divided into variable-sized blocks on allocation. Each block has a start address, size, and unique identifier.

2.2 Data Structures

- `vector<uint8_t>` for raw memory storage
- `list<MemoryBlock>` for the free block list
- `map<size_t, AllocatedBlock>` for allocated blocks

2.3 Assumptions

- Flat physical memory (no segmentation or paging)
- No memory protection or access permissions
- All addresses are byte-based
- Fragmentation is allowed and handled via coalescing

3 Allocation Strategy Implementations

The allocator supports three strategies.

3.1 First Fit

Selects the first free block large enough to satisfy the request.

3.2 Best Fit

Selects the smallest free block that can satisfy the request.

3.3 Worst Fit

Selects the largest available free block.

3.4 Allocation Process

1. Locate suitable block using strategy
2. Split block if larger than needed
3. Register allocation in allocated map
4. Update free list

4 Freeing and Coalescing

When a block is freed, it is returned to the free list and adjacent free blocks are merged to reduce fragmentation.

5 Buddy System Design

A buddy system was considered but not implemented due to complexity and project scope limitations.

6 Cache Hierarchy

Two cache levels are modeled:

- L1 cache (small and fast)
- L2 cache (larger and slower)

Each cache is parameterized by size, block size, and associativity.

7 Cache Replacement Policy

FIFO replacement is used. The oldest entry in a set is evicted on a miss when no empty line is available.

8 Cache Access Flow

Address translation:

$$\text{Address} \rightarrow \text{Block Address} \rightarrow \text{Set Index} \rightarrow \text{Tag Comparison}$$

9 Cache Statistics

Each cache tracks hit and miss counts. Total cycle cost is accumulated using:

- L1 hit: 1 cycle
- L2 hit: 10 cycles
- Memory access: 100 cycles

10 Virtual Memory Model

Virtual memory is not implemented. No page tables, TLBs, or page faults are simulated.

11 Address Translation Flow

The system directly translates block ID and offset into a physical address.

12 Limitations and Simplifications

- Single-threaded execution
- No memory protection
- No paging or swapping
- FIFO replacement only
- No cache coherence

13 Conclusion

The simulator models physical memory allocation and caching behavior for educational purposes. It intentionally simplifies real-world systems to maintain clarity and focus.