Memory Management Simulator

Harshit Joshi (24116038)

Introduction

This project implements a **Memory Management Simulator** that models core operating system memory subsystems in a structured and observable way. The simulator focuses on correctness, architectural clarity, and educational value.

The system includes:

- Variable-size memory allocation strategies
- Buddy memory allocation system
- Multi-level cache hierarchy
- Virtual memory with paging
- Integrated address translation pipeline
- Performance metrics and statistics

Memory Layout and Assumptions

Physical Memory Model

- Total physical memory size: **1024 bytes**
- Memory is represented as a **linear contiguous address space**
- Memory is divided into blocks with:
    - start address
    - size
    - allocation status
    - block ID (for allocated blocks)

Memory Layout Diagram

```
+----------------------------------------------------+
| Alloc 100 | Free 300 | Alloc 200 |     Free      |
+----------------------------------------------------+
0          100        400         600             1024
```

Assumptions

- Single-process model
- No segmentation
- No alignment constraints except buddy system rules
- No concurrency or parallel execution
- Memory access is simulated, not executed on hardware

Allocation Strategy Implementations

The simulator supports three classic **variable-size allocation strategies**, all operating on the same physical memory.

First Fit

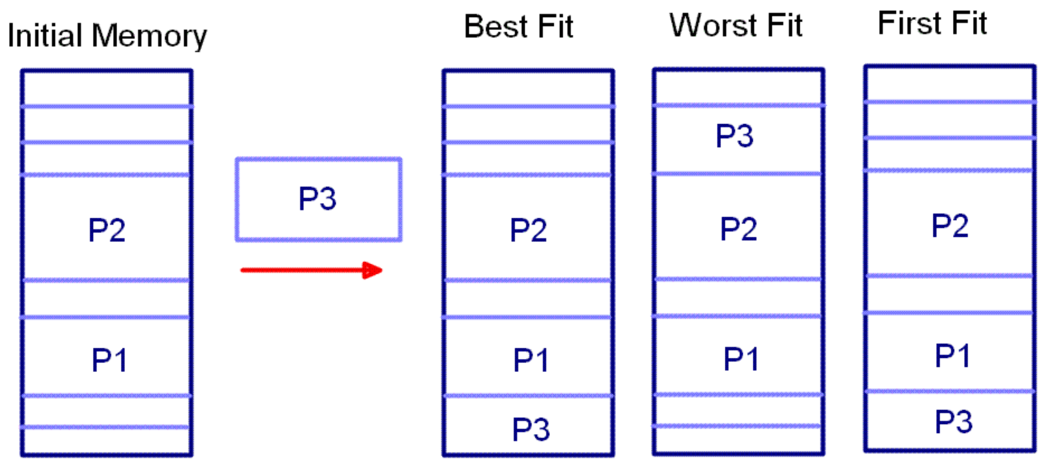
- Scans memory blocks from the beginning
- Allocates the first free block large enough
- Fast but prone to external fragmentation

Best Fit

- Searches all free blocks
- Allocates the smallest suitable block
- Reduces wasted space but increases fragmentation risk

Worst Fit

- Allocates the largest available block
- Attempts to preserve medium-sized blocks
- Often inefficient in practice



Buddy System Design

Overview

The buddy allocator manages memory in **power-of-two block sizes**, trading external fragmentation for internal fragmentation.

- Total memory: 1024 bytes (2^{10})
- Block sizes: 1, 2, 4, ..., 1024 bytes
- Blocks are always aligned to their size

Buddy Allocation Diagram

```
1024
 ├──512 (free)
 └──512
      ├──256 (free)
      └──256
           ├──128 (allocated)
```

```
                └──128 (free)
```

Allocation

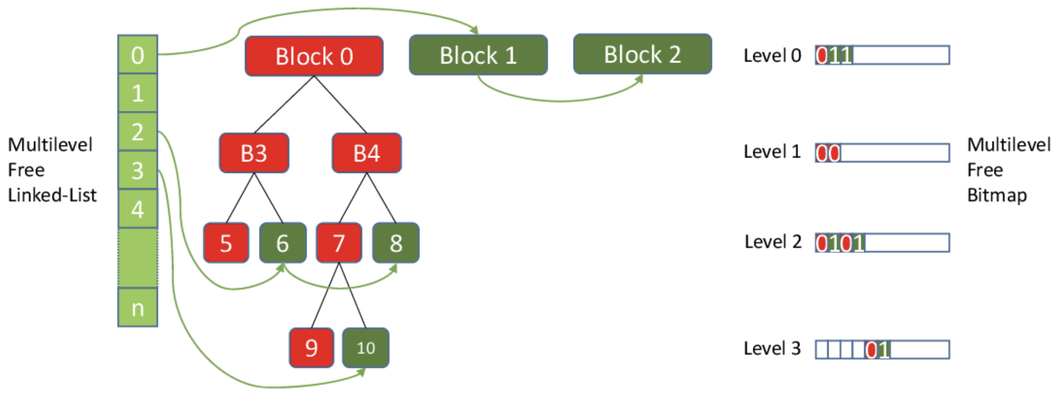
1. Requested size is rounded up to nearest power of two
2. Smallest suitable block is found
3. Larger blocks are recursively split
4. Final block is allocated

De-allocation

1. Block is freed
2. Buddy address is computed using XOR
3. If buddy is free, blocks are merged
4. Merging continues recursively

Fragmentation Characteristics

- External fragmentation: **negligible**
- Internal fragmentation: **possible and tracked**



Cache Hierarchy and Replacement Policy

Cache Structure

The simulator models a **two-level cache hierarchy**.Cache

- Size: 64 bytes
- Block size: 8 bytes
- Associativity: 2-way
- Replacement policy: LRU Cache
- Size: 256 bytes
- Block size: 8 bytes
- Associativity: 4-way
- Replacement policy: FIFO Cache Hierarchy Diagram

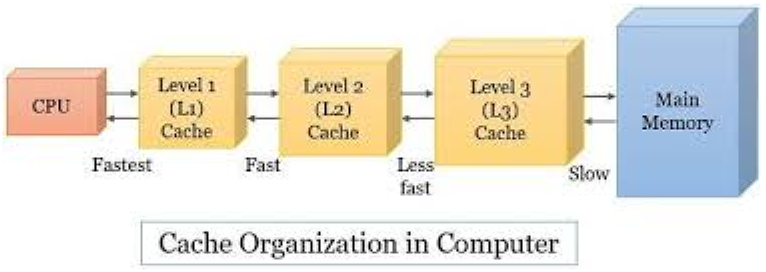
```
           +--------+
Address -> |  L1    |  Hit?
           +--------+
                |
              Miss
                v
           +--------+
           |  L2    |  Hit?
           +--------+
                |
              Miss
                v
           Main Memory
```

Each cache is set-associative and consists of:

Cache Organization

- Sets
- Cache lines per set
- Each line contains:

    - Valid bit
    - Tag
    - Replacement metadata



Virtual Memory Model

Paging System

- Page size: 64 bytes
- Physical frames: 8
- Virtual address space is conceptually unlimited

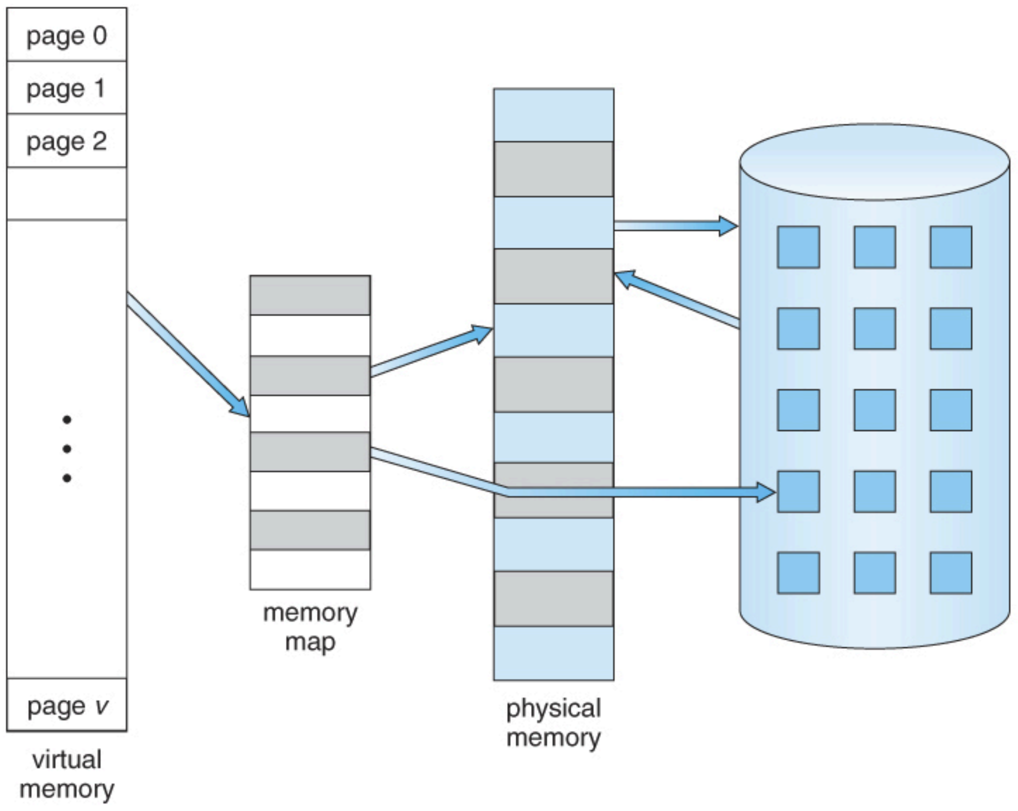
Page Table

Each page table entry contains:

- Valid bit
- Frame number (if valid)

Page Replacement

- FIFO policy is used
- Page faults occur on first access or eviction



Address Translation Flow

Translation Pipeline

The simulator strictly follows the OS-standard order:

```
Virtual Address
→ Page Table
→ Physical Address
→ Cache (L1 → L2)
→ Main Memory
```

Address Translation Diagram

```
Virtual Address
+-------------------+
| PageNo |Offset  |
+------------------+
       |
       v
   PageTable
       |
       v
Physical Address
       |
       v
Cache
       |
       v
   Main Memory
```

Design Rationale

- Cache operates only on physical addresses
- Virtual memory is upstream of cache
- Matches real MMU + cache architecture

Metrics and Statistics

The simulator computes and reports the following metrics.

Internal Fragmentation

- Applies to buddy allocator
- Calculated as:

```
Allocated block size − Requested size

External Fragmentation
```

- Applies to variable-size allocation strategies

- Calculated as:

```
1 − (LargestFree Block/ TotalFree Memory)
```

Allocation Success / Failure Rate

Tracks:

- Total allocation requests
- Successful allocations
- Failed allocations

Memory Utilization

```
UsedMemory/TotalMemory
```

User Interface Design

Visualization Features

- Physical memory bar (free vs allocated blocks)
- Buddy allocator block visualization
- Cache set and line visualization
- Integrated VM → Cache access trace
- Live statistics panel

Design Philosophy

- UI reflects actual internal state
- No inferred or artificial visualization
- Visuals complement textual output

Limitations and Simplifications

To maintain clarity and focus, the simulator intentionally omits:

- Multi-process isolation
- TLB simulation
- Dirty bits
- Write-back / write-through cache distinction
- Concurrency
- Hardware timing simulation

These simplifications allow emphasis on **core memory management principles**.

Conclusion

This simulator provides a faithful and observable model of modern memory management. By integrating allocation strategies, paging, cache hierarchy, and performance metrics, it serves as a strong educational and experimental tool for understanding operating system memory behavior.