b

Design Document : Memory Management Simulator

Name : Dara Zenas Zephaniah

Enrollment : 23321010

① Memory Layout and Assumptions - The Simulator models the computer's physical memory as a contiguous block of raw bytes, managed dynamically at runtime.

Physical memory size : 1024 Bytes (configurable).

Addressing : Byte-addressable simulation.

Data Structure : The memory is not tracked by a bitmap but by a Double Linked List of MemorySegment* structures.

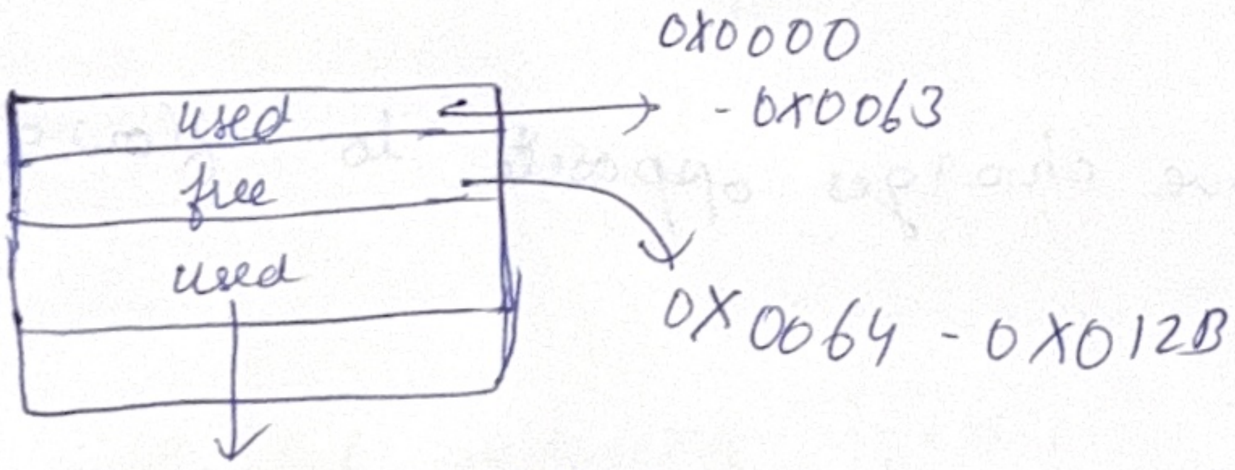
Metadata : Each segment header stores

start_addr : Physical starting address

size : Size of the block in bytes

is_allocated : Boolean flag (True = Used, False = Free).

id : Unique identifier for tracking allocation (0 for holes)



② Allocation Strategy Implementation

Algorithm: First Fit

The Simulator implements the first Fit allocation strategy, prioritized for its speed and simplicity.

a) Search : The allocator traverses the linked list from the head (start_addr: 0)

b) Selection : It selects the first free block that satisfies

block.size >= requested_size

c) Splitting : If the selected block is larger than required, it is split into two:

Block A (Allocated) : Shrinks to the requested size.

Block B (Free) : Created from the remaining space and inserted into the list immediately after block A.

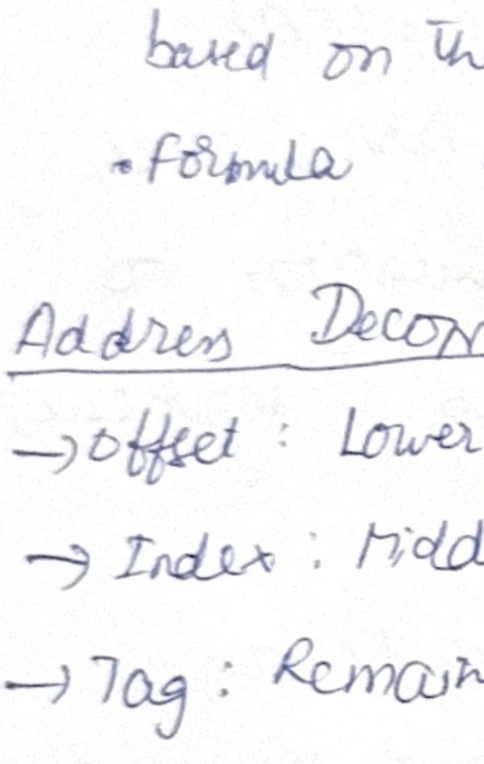
d) Coalescing (Deallocation): When free(id) is called.
→ The block is marked as free.
→ The system checks the next and prev pointers.
→ If a neighbor is also free, they are merged into a single larger block to mitigate external fragmentation.

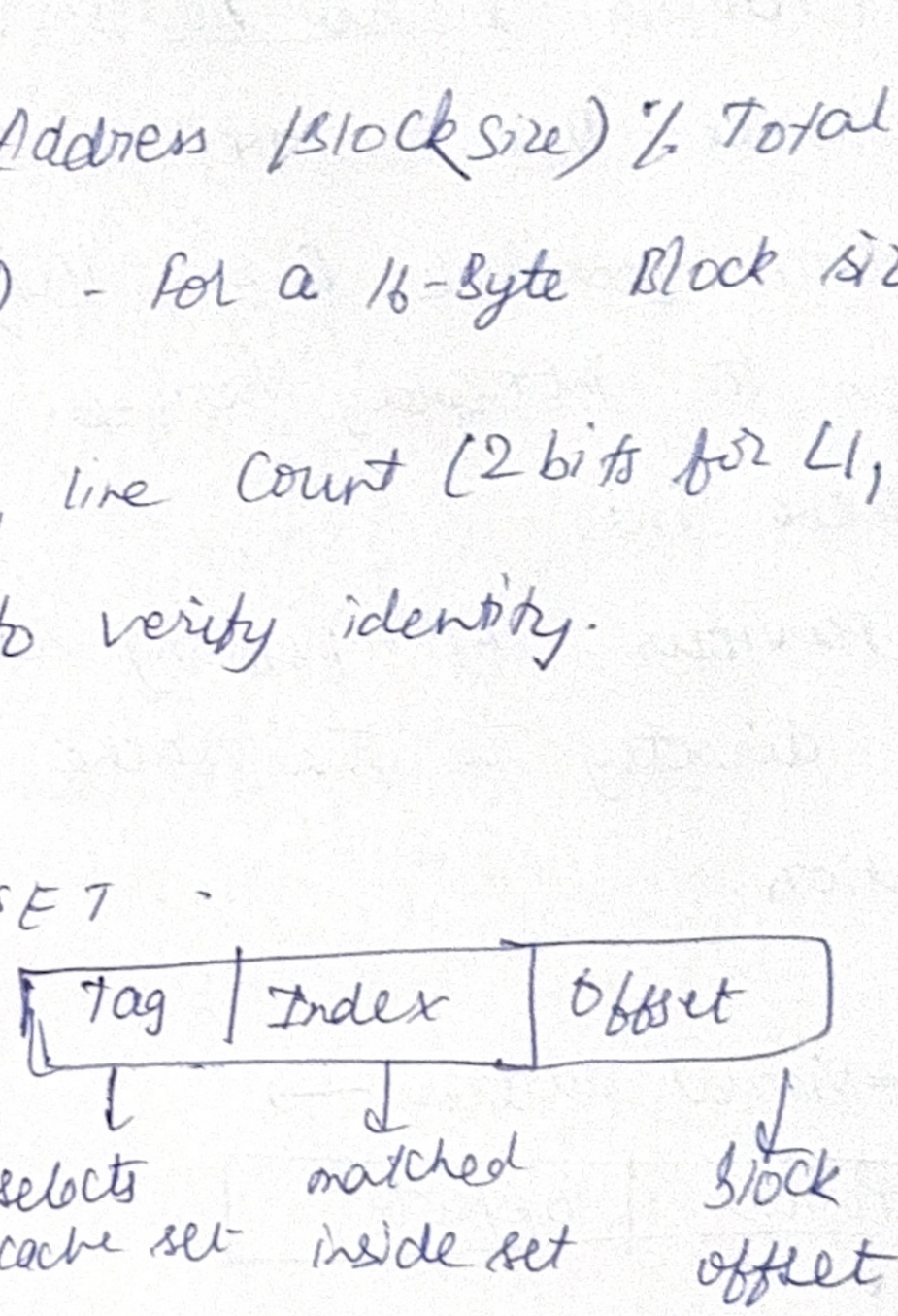
③ Cache Hierarchy and Replacement Policy - The system simulates a Two-level (L1/L2) Inclusive cache Hierarchy to reduce access latency.

Cache specifications
→ L1 cache: 64 Bytes (4 lines x 16 Byte Blocks). High speed, small capacity.
→ L2 cache: 256 Bytes (16 lines x 16 Byte Blocks). Slower speed, larger capacity.
→ Mapping Policy: Direct Mapped.
- Each memory address map to exactly one specific line in the cache based on the index bits.
- Formula Cache Line Index = (Address / Block size) % Total Lines

Address Decomposition (Bitwise Logic) - for a 16-Byte Block size:
→ offset: Lower 4 bits ($\log_2 16$).
→ Index: Middle bits determined by line count (2 bits for L1, 4 bits for L2)
→ Tag: Remaining Upper bits used to verify identity.

Address format: TAG | INDEX | OFFSET.





Access Flow

1) Check L1: Extract L1 Index/Tag. If valid bit is set and tags match → L1 HIT (1 cycle)

2) Check L2: If L1 Miss, extract L2 Index/Tag. If L2 Hit → L2 HIT (+10 cycles). Promote block to L1.

3) Fetch RAM: If L2 Miss → Access Main Memory (+1000 cycles). Load block into L2, then into L1.

④ Virtual Memory Model (Simplification)

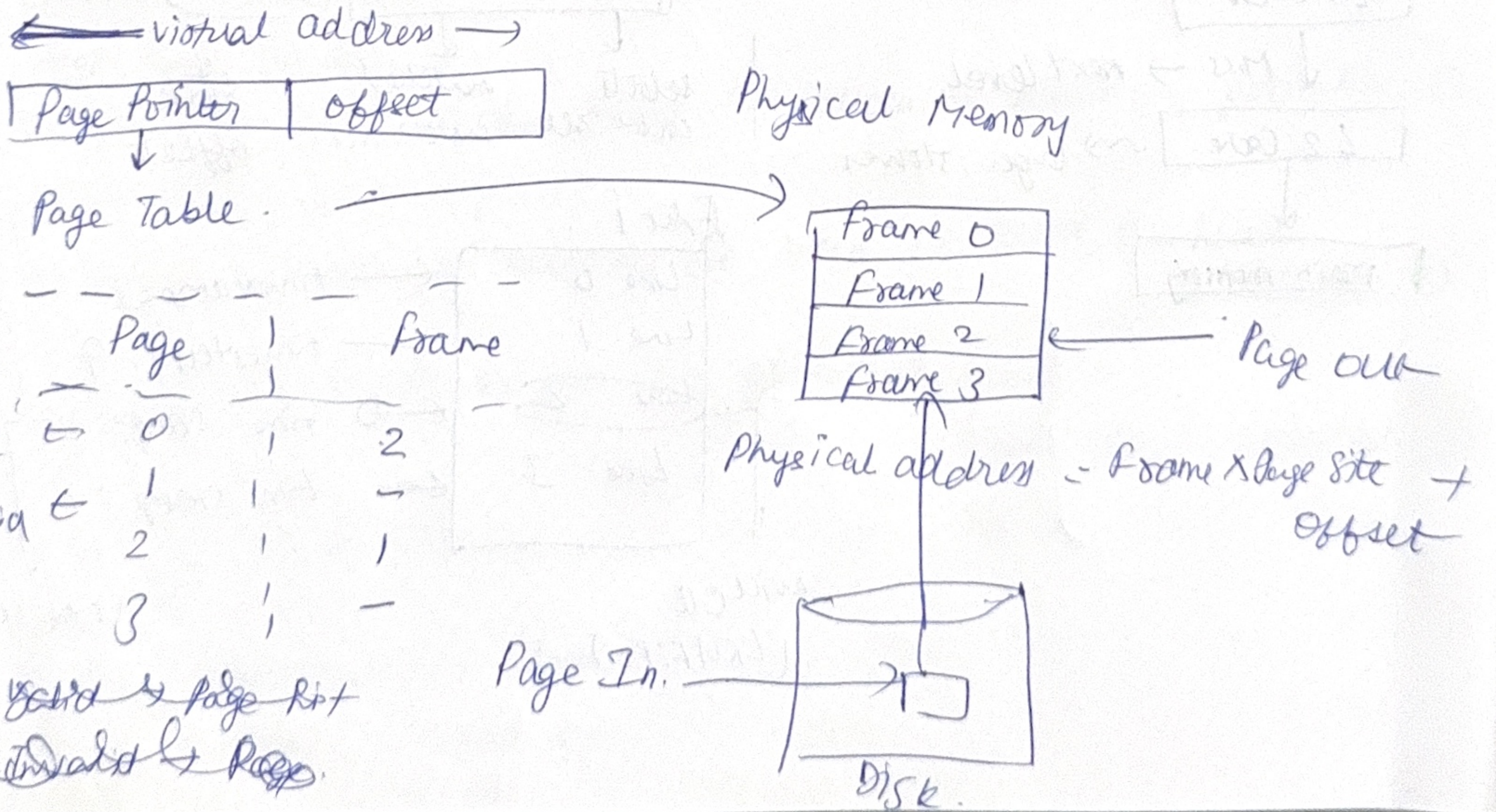
Design Choice: Identity Mapping

To prioritize the fidelity of the Cache and Allocation subsystems, this version of the simulator implements Identity Mapping for virtual memory.

→ Mechanism: Virtual Address (VA) == Physical Address (PA)

→ Rationale: This removes the overhead of Page Tables and Translation Lookaside Buffers (TLB), allowing the simulation to focus on the interplay between variable-size memory segments and cache lines.

→ Behaviour: All memory requests from the shell are passed directly to the Cache Hierarchy logic without intermediate translation.



⑤ Address Translation Flow - since identity mapping is used, the flow is streamlined

a) User request: malloc(16) returns Virtual Address 0

b) Translation: VA 0 maps to physical address 0

c) Cache Lookup:

→ Address 0 is binary ... 0000 0000.

→ L1 Index 00 (Line 0). Check Line 0.

→ L2 Index 0000 (Line 0). Check Line 0.

d) Memory Access: If both caches miss, data is fetched from Physical RAM at index 0.

⑥ Limitations and Simplications

a) No Disk Swap - The system does not simulate hard disk storage or Page Faults (swapping Pages in/out). It assumes all memory is resident.

b) Single process - The Simulator manages a single global memory space, rather than seperate ~~proces~~ spaces for multiple processes.

c) Direct Mapped Cache: While simple, this strategy suffers from conflict misses compared to ~~set~~ set-Associath caches.

d) Buddy System - A Buddy allocator was considered but First Fit was chosen to allow for arbitary allocation sizes without strict power-of-two alignment constraints, reducing Internal Fragmentation.