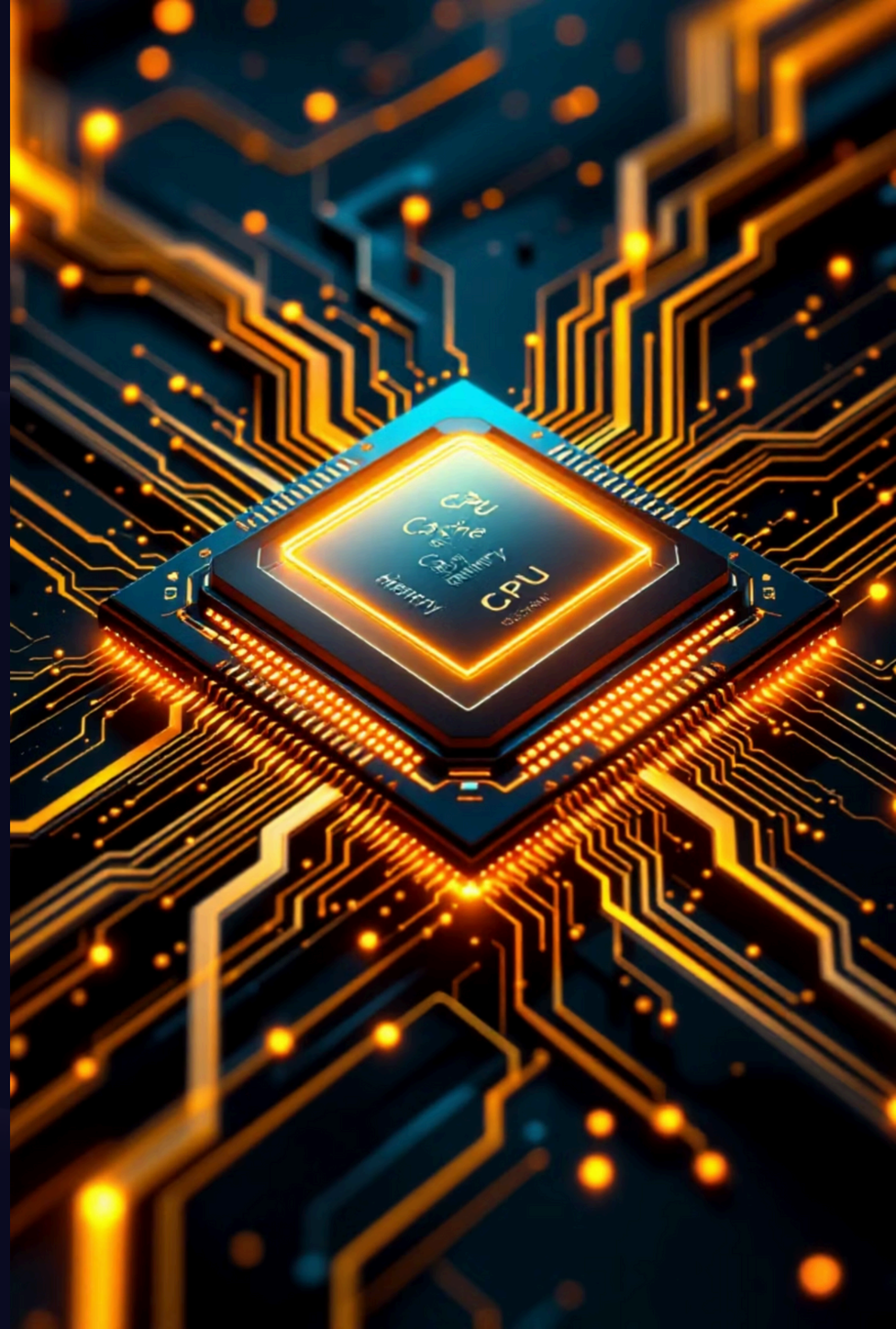
Direct-Mapped Cache Memory

Bridging the memory wall through cache for enhanced processor performance.

Presentation by Group 10:

- Garima Kamra
- Palak
- Moinak Goswami
- Harshit Sahu
- Sarthak Keshaowar



The Memory Wall Challenge

Processor Speed

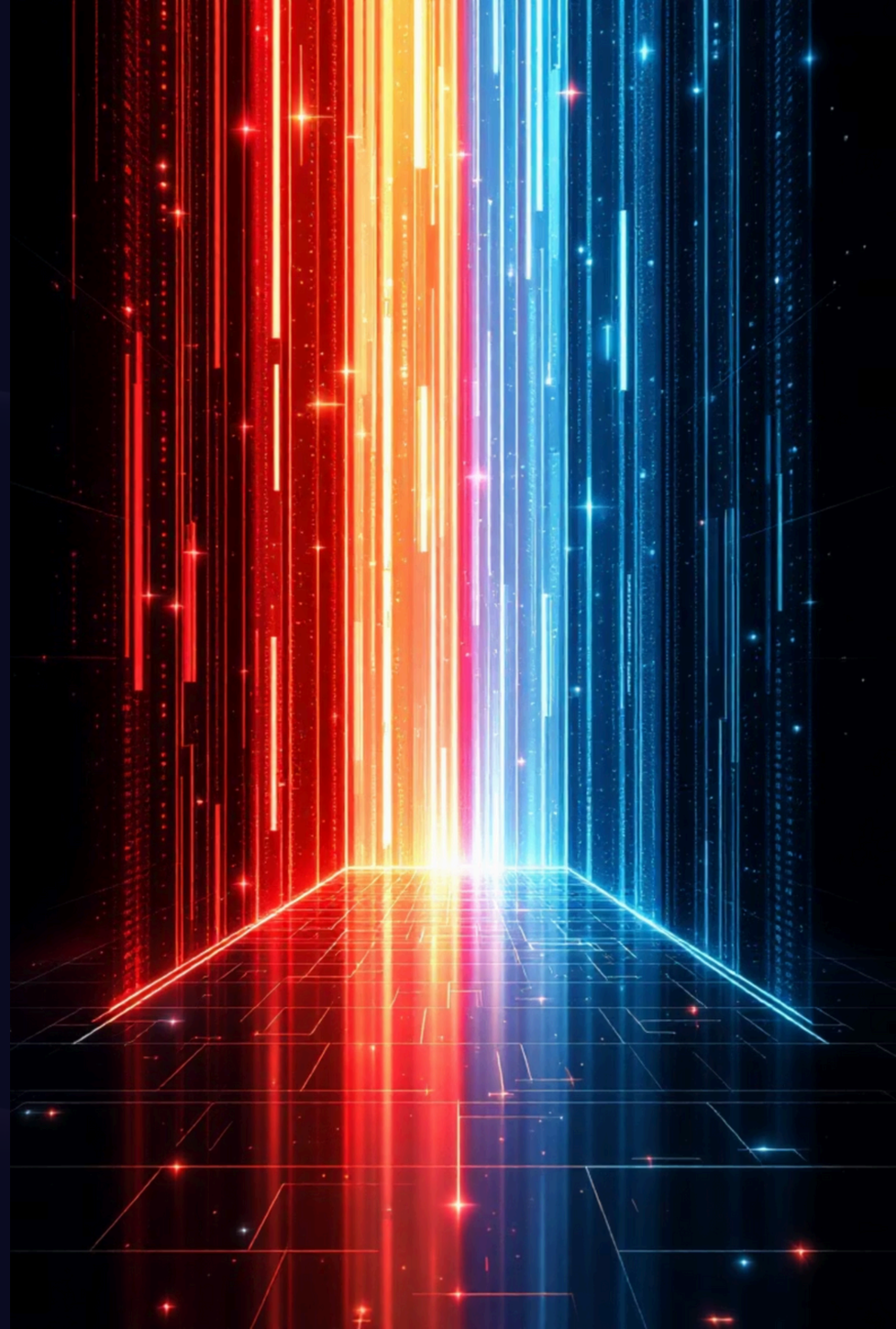
Modern CPUs operate at gigahertz speeds, demanding rapid data access.

Main Memory Latency

DRAM access takes hundreds of clock cycles, creating a significant performance bottleneck.

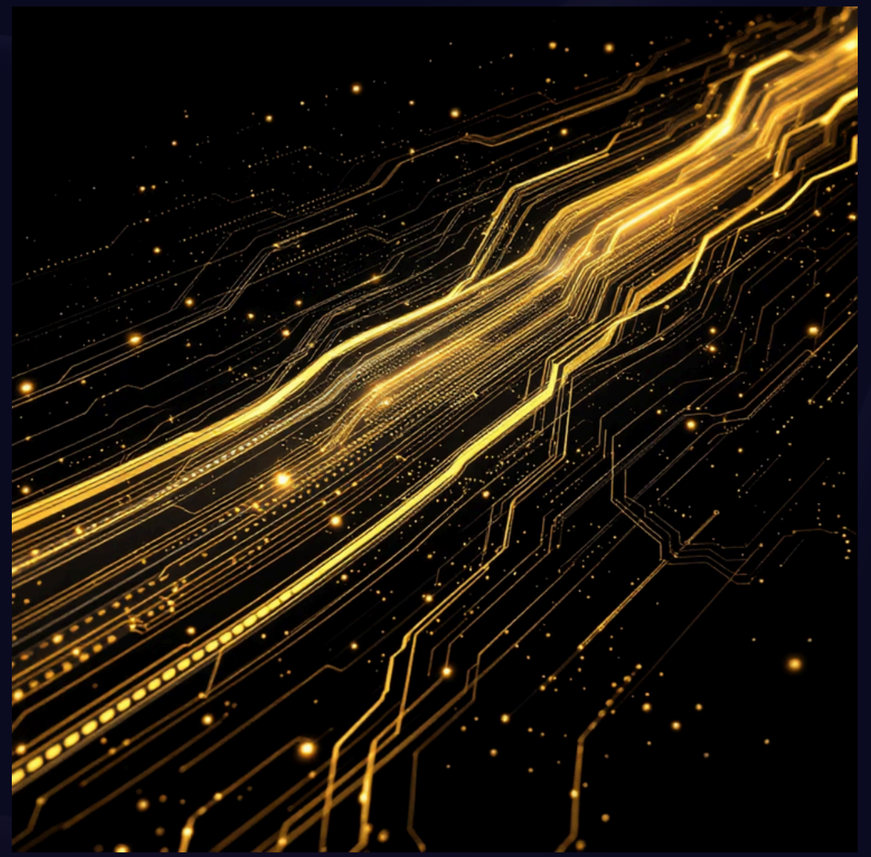
The Gap

This disparity, known as the memory wall, severely limits system efficiency.



Cache Memory: The Solution

- Cache memory stores frequently accessed data closer to the CPU, drastically reducing access times from ~100 cycles to ~2–5 cycles.
- With typical hit rates of 90-99% , cache memory delivers 20-50x performance improvement , making it essential for modern computing.
- While direct-mapped caches have limitations like conflict misses, their simplicity, speed, and low cost make them ideal for learning fundamental caching concepts and understanding memory hierarchy design.



Direct-Mapped Cache Architecture

Simple Mapping

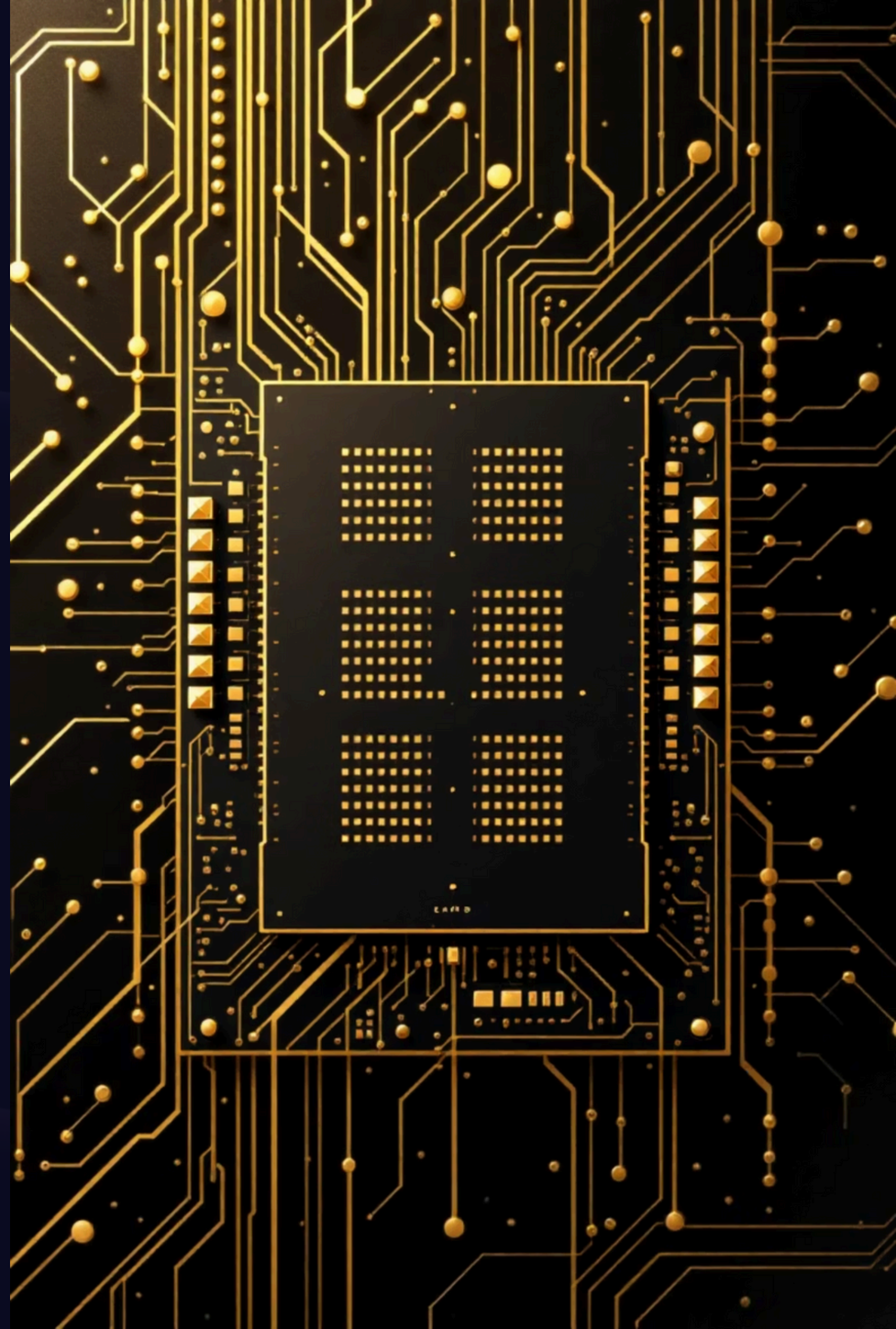
Each memory block maps to exactly one cache line.

Index Calculation

Cache Line Index = (Block number) % (Number of Cache Lines).

16-bit Address

Divided into Tag (8 bits), Index (6 bits), and Offset (2 bits).

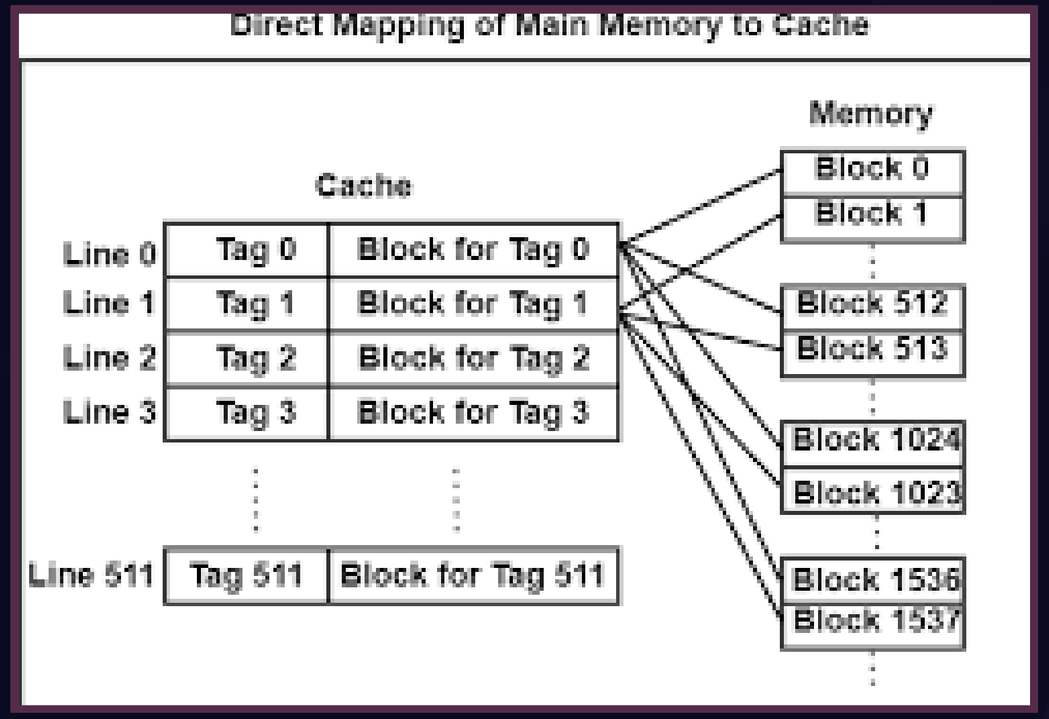


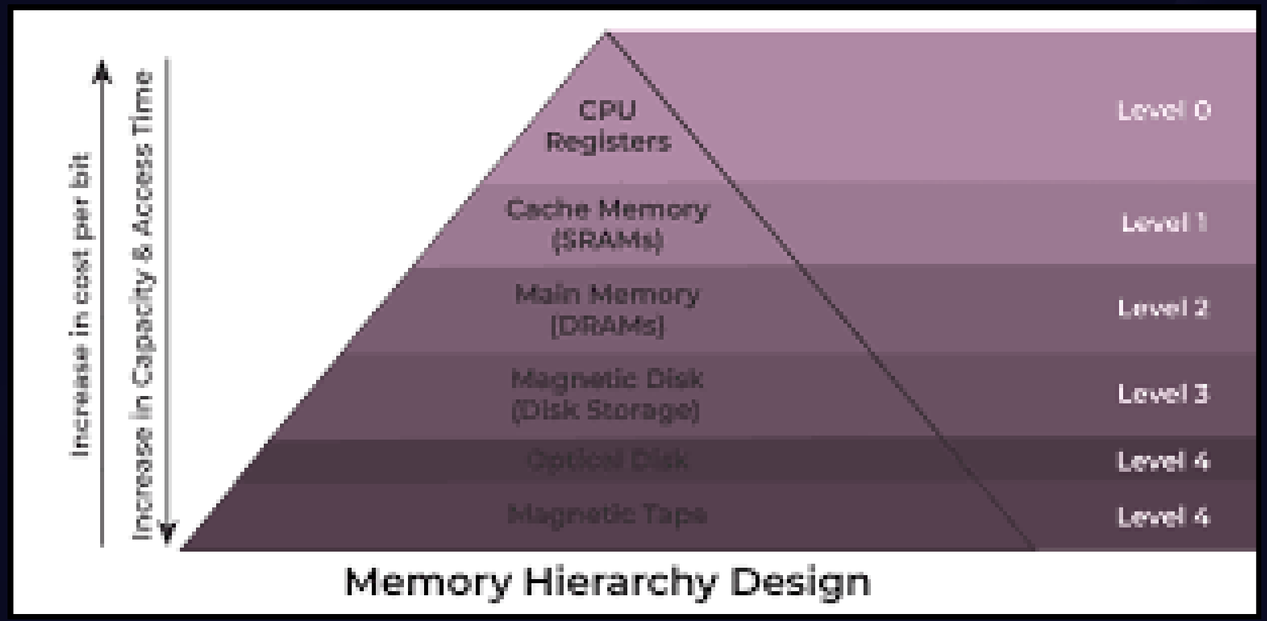
Cache Line Structure & Operations

- **Main memory blocks and Cache lines**
  1. **Main memory** - Divided into blocks which maps to only one cache line
  2. **Cache** - Contains the tag, data, valid and dirty bits
- **Read operations**

  Two cases possible -
  1. **Hit -** Directly return from Cache
  2. **Miss -** Has to go to Main memory
- **Write operations**

  Two policies used -
  1. **Write Through + No Write Allocate -**
     a. **Hit -** Updates both Cache and Memory
     b. **Miss -** Updates only the Memory
  2. **Write Back + Write Allocate -**
     a. **Hit -** Updates only the cache and marks the dirty bit = 1
     b. **Miss -** Replaces the cache line and writes to the new one.





System Components

Cache Memory Array

Stores data, tags, valid and dirty bits

Comparator Logic

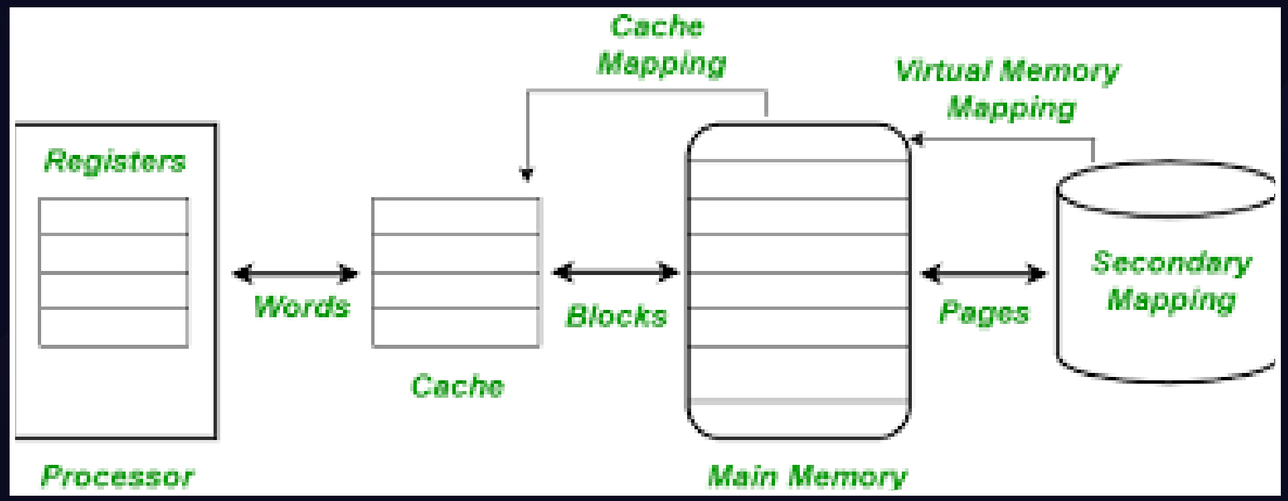
Compares incoming address tag with stored tag.

Main memory Module

Simulates cache fills/evictions, data fetch and write through operations

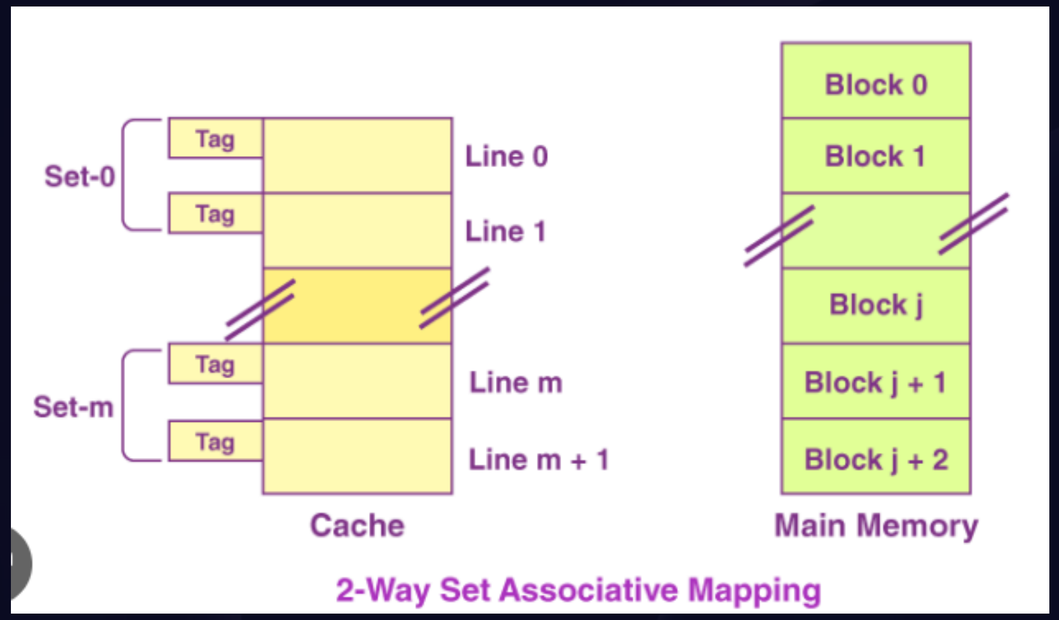
FSM Control Logic

Manages read/write operations, hit/miss detection,



Cache Configuration

Each memory block maps to exactly one cache line.



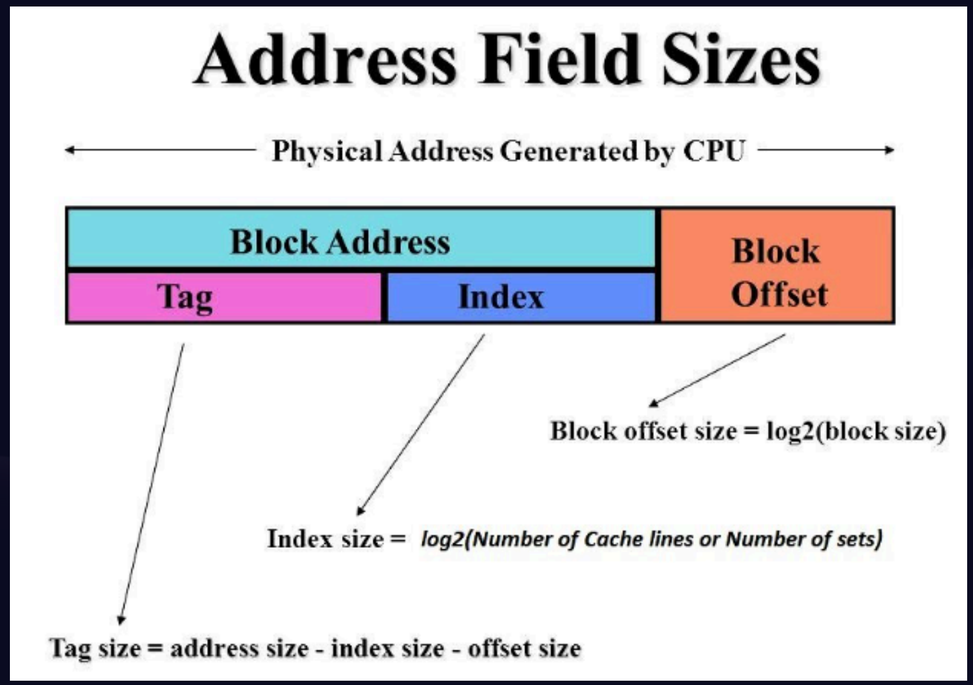
2-Way Set Associative Mapping

The whole cache and memory system is divided into blocks(the smallest addressable unit is called word), the cache is divided into cache lines

TAG : It distinguishes which memory block is stored in the cache line

INDEX : Index points to the cache line

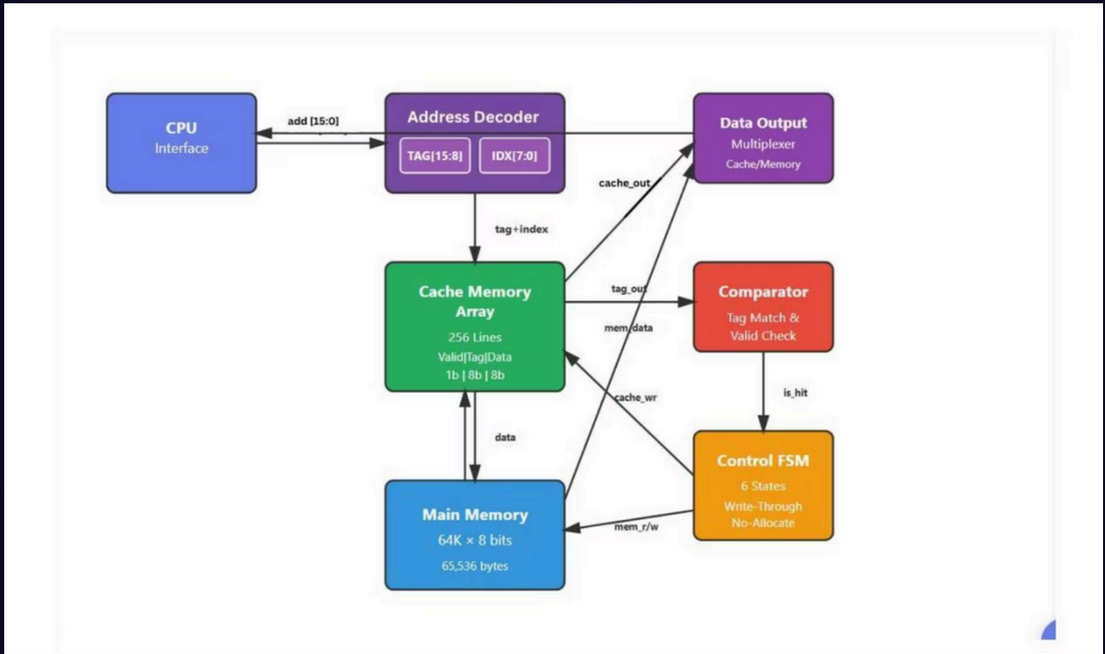
OFFSET : Identifies the specific byte within the cache block



Visualizing the structural and control flow of the direct-mapped cache.

Block Diagram & FSM

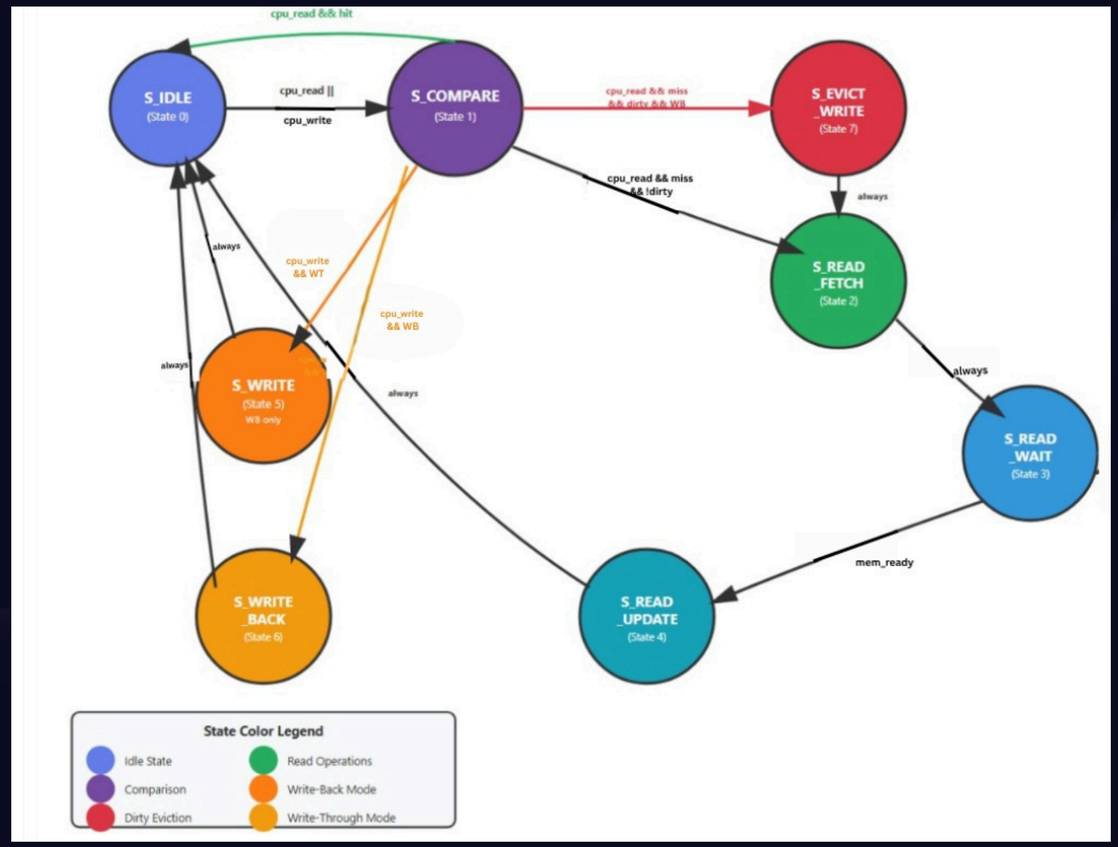
Block Diagram



A direct-mapped cache is a computer memory architecture where every main memory block has exactly one possible location in the cache to be stored

Other types of cache mapping are fully associative cache (a memory block can go anywhere) and set-associative cache (a memory block can go anywhere within a specific "set" of cache lines).

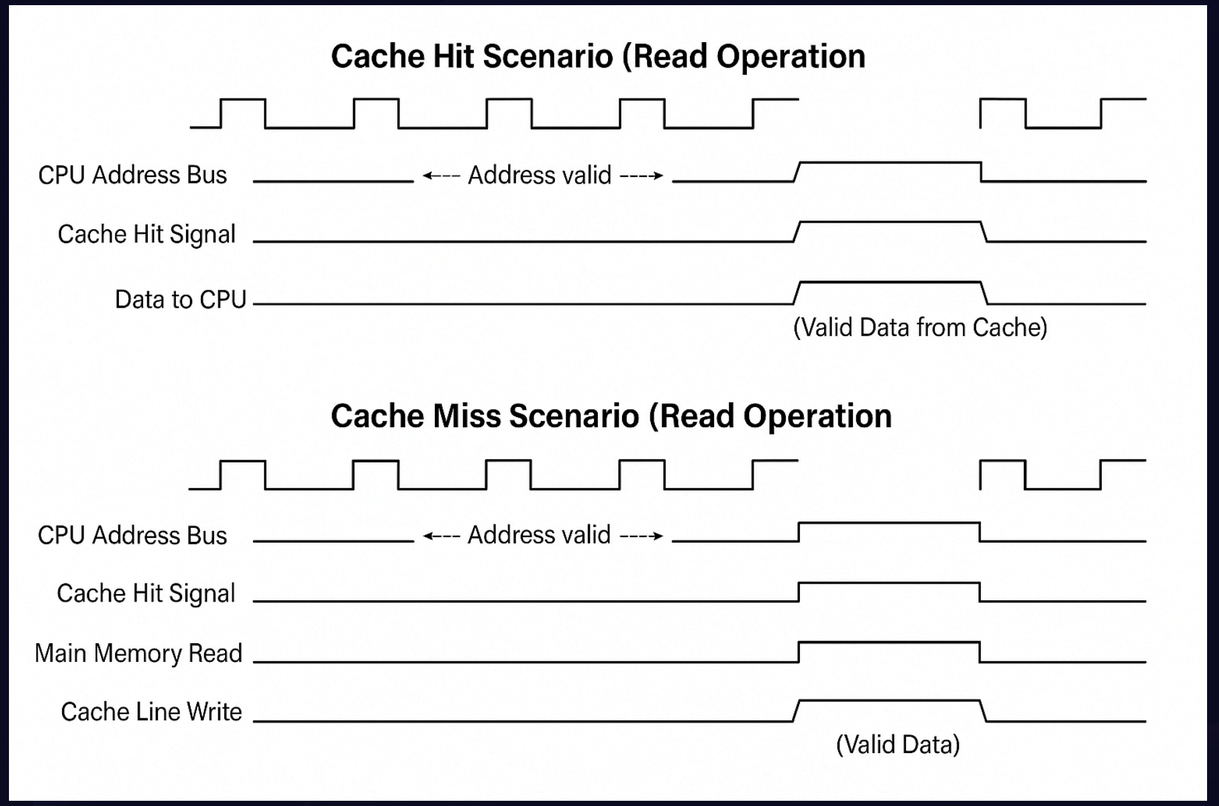
FSM Diagram



TIMING ANALYSIS

Timing Diagram

Illustrates critical signal timings during cache operations.

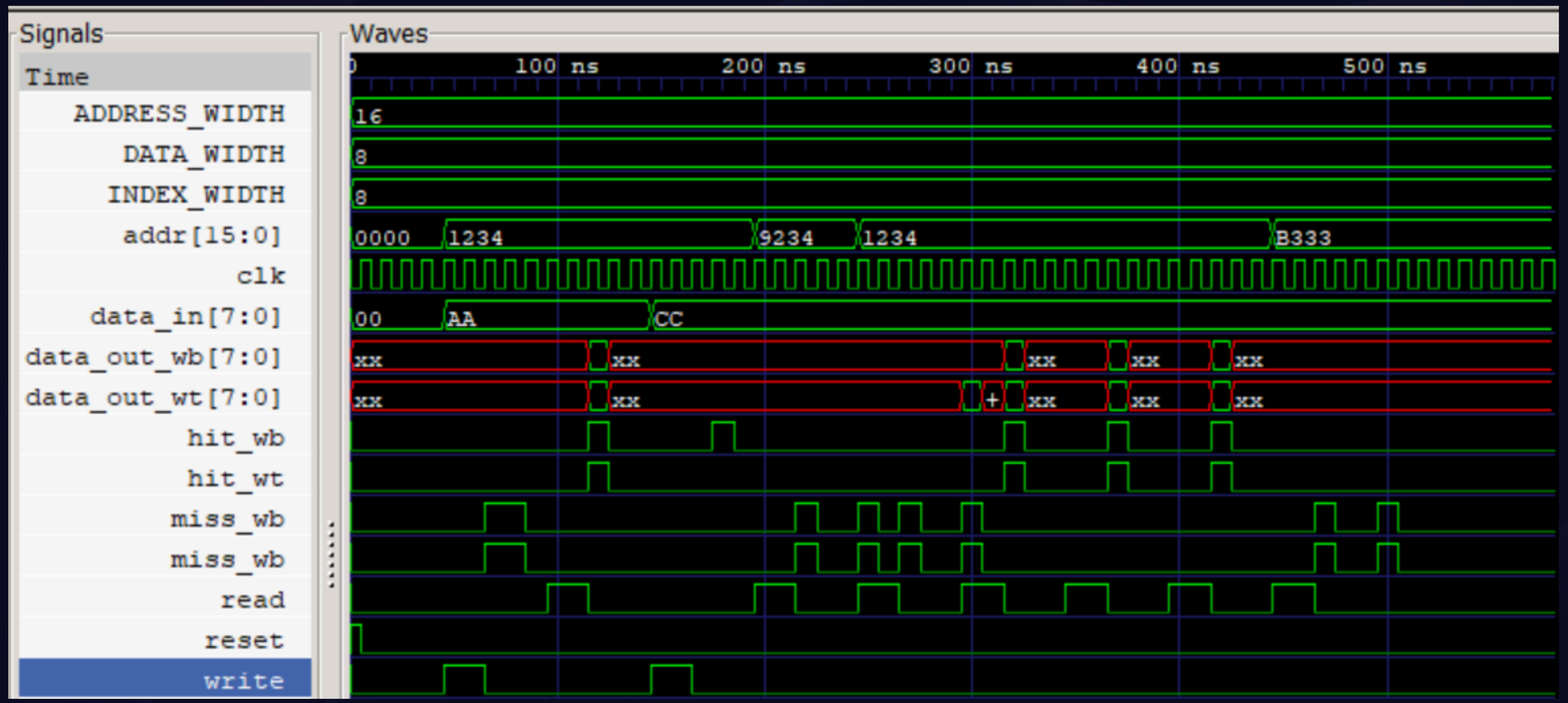


WAVEFORM ANALYSIS

*Actual waveform generated by our code

Waveform

Detailed view of signal changes over time, confirming functionality.



Conclusion

This direct-mapped cache design effectively reduces access latency and significantly enhances processor performance, offering a foundational understanding of memory hierarchy.