# PROJECT REPORT - 2025-2026
# COMPUTER ORGANIZATION AND ARCHITECTURE

**STUDENTS IN A TEAM :-**

**Ram Rastogi - 23517093**
**Saurav Garg- 23517109**
**Akshit Chauhan - 23517084**

**TOPIC:**

**SIMULATE DIRECT MAPPED CACHE OR ASSOCIATIVE CACHE USING PYTHON**

**SUBMITTED TO - DR. NAVEEN CHAUHAN**

## ABSTRACT

This project implements an **interactive cache memory simulator** to visualize the behavior of **Direct-Mapped and Associative caches**.

It demonstrates how a CPU accesses memory and how cache hits and misses occur.

The simulator displays **cache lines, tags, valid and dirty bits, and data blocks** for clear understanding.

Step-by-step **animations and visual feedback** help users follow each memory access operation.

Users can configure **cache size and block size** to observe performance differences.

This tool serves as an **educational aid for learning cache concepts** and memory optimization techniques.

## INTRODUCTION

This project is an **interactive cache memory simulator** that helps visualize how a CPU accesses memory using **Direct-Mapped and Associative caches**.
It shows **hits, misses, valid/dirty bits, and data blocks** with step-by-step animations.
Users can also **change cache configurations** to see how it affects performance, making it an effective educational tool for understanding cache behavior.

## **Objectives:**

1. **To simulate Direct-Mapped and Associative cache memory for educational purposes.**

2. **To visualize cache operations including hits, misses, and memory writes.**

3. **To demonstrate the roles of valid bits, dirty bits, tags, and data blocks in cache.**

4. **To allow users to configure cache size and block size and observe performance changes.**

5. **To provide interactive animations and visual feedback for better understanding of cache behavior.**

6. **To serve as a teaching and learning tool for students studying computer architecture.**
7. **To help students visualize the effect of different cache replacement policies (like LRU, FIFO) on performance.**

8. **To provide real-time logs and feedback for each memory access to enhance understanding.**

9. **To compare Direct-Mapped and Associative cache performance and demonstrate how configuration impacts hit/miss ratios.**

## TOOLS AND TECHNOLOGIES:

**Tools and Technologies:**

1. **Programming Language:** Python (for CLI simulation and backend logic)

2. **Web Technologies:** HTML, CSS, JavaScript (for interactive web interface)
3. **Libraries/Frameworks:**

    ○ Python: None required (or use standard libraries like `time` for delays)

4. **Development Environment:** Replit IDE / VS Code

5. **Browser:** Any modern web browser to run the interactive simulator

6. **Design Tools:** CSS for animations, gradients, and responsive design

## System Design / Flow

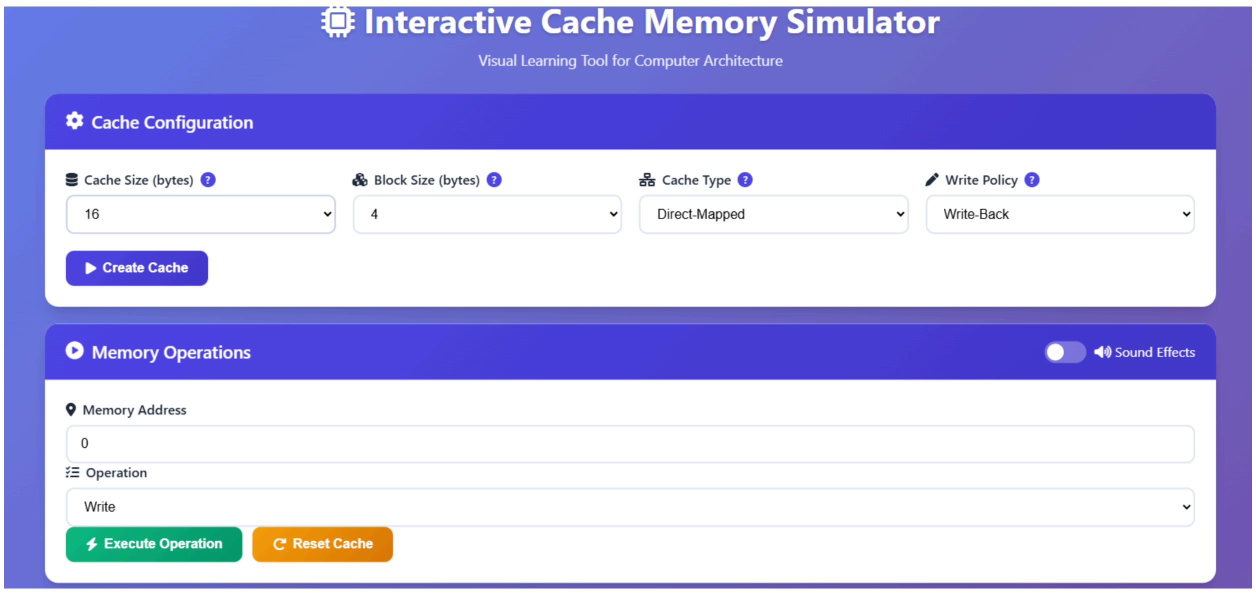
**System Flow:**

1) **Input: User enters memory addresses and cache configuration.**

2) **Processing: Simulator checks cache for hits or misses, updates cache lines, tags, valid and dirty bits.**

3) **Output: Visual feedback with animations showing hits, misses, memory writes, and step-by-step progress.**

4) **Flow: Fetch → Compare Tag → Hit/Miss → Update Cache → Update Memory (if needed)**



Interactive Cache Memory Simulator

Visual Learning Tool for Computer Architecture

Cache Configuration

Cache Size (bytes): 16

Block Size (bytes): 4

Cache Type: Direct-Mapped

Write Policy: Write-Back

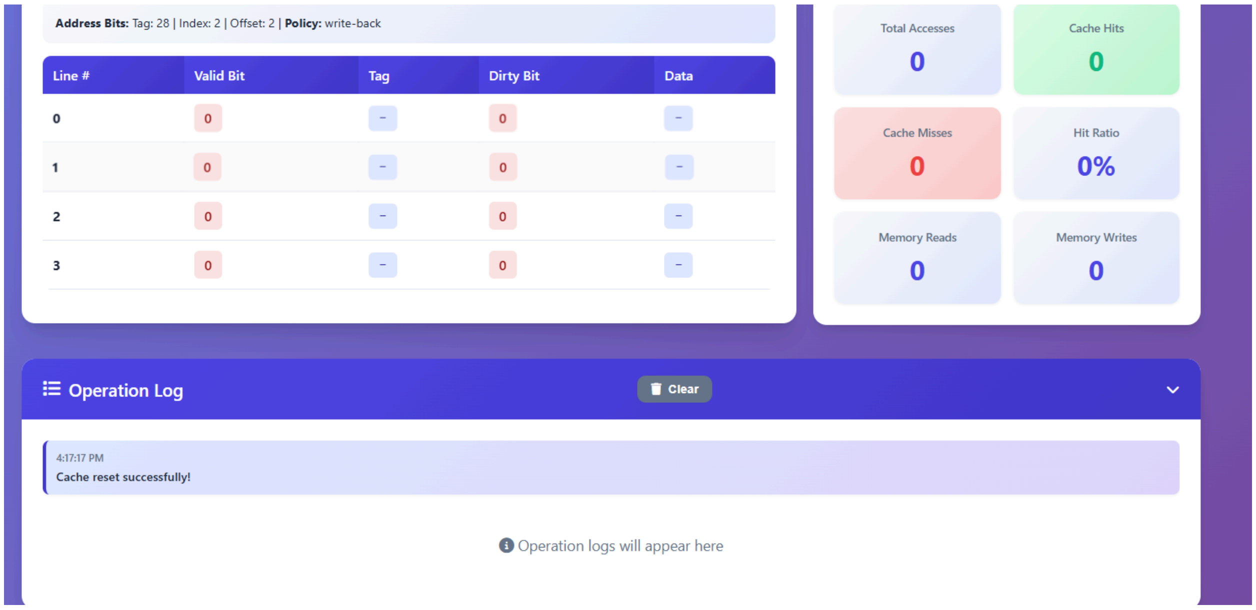
Create Cache

Memory Operations — Sound Effects

Memory Address: 0

Operation: Write

Execute Operation | Reset Cache



**Address Bits:** Tag: 28 | Index: 2 | Offset: 2 | **Policy:** write-back

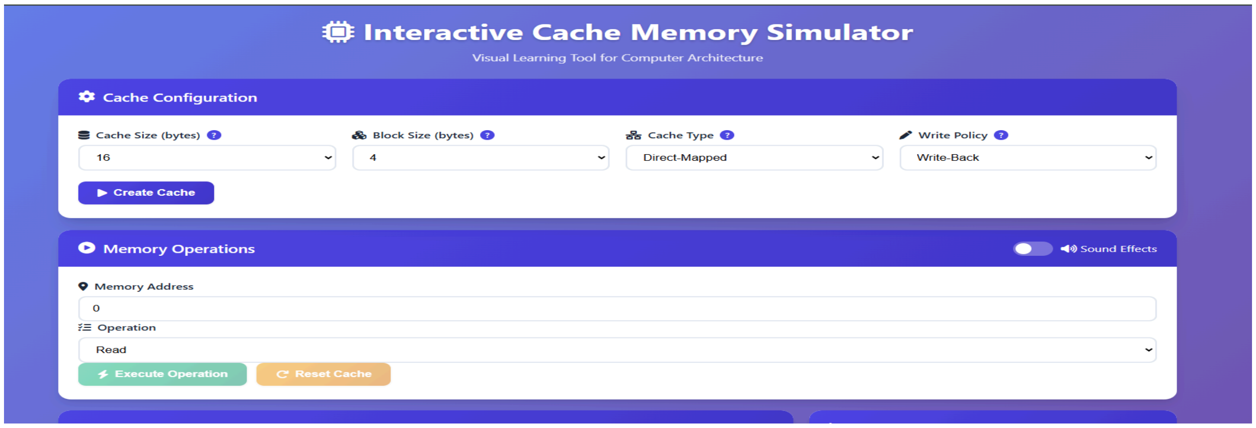
| Line # | Valid Bit | Tag | Dirty Bit | Data |
|---|---|---|---|---|
| 0 | 0 | – | 0 | – |
| 1 | 0 | – | 0 | – |
| 2 | 0 | – | 0 | – |
| 3 | 0 | – | 0 | – |

| | |
|---|---|
| Total Accesses: 0 | Cache Hits: 0 |
| Cache Misses: 0 | Hit Ratio: 0% |
| Memory Reads: 0 | Memory Writes: 0 |

Operation Log — Clear

```
4:17:17 PM
Cache reset successfully!
```

Operation logs will appear here



Interactive Cache Memory Simulator

Visual Learning Tool for Computer Architecture

Cache Configuration

Cache Size (bytes): 16

Block Size (bytes): 4

Cache Type: Direct-Mapped

Write Policy: Write-Back

Create Cache

Memory Operations — Sound Effects

Memory Address: 0

Operation: Read

Execute Operation | Reset Cache

**Implementation**

**Member 1 – UI Design:** Interactive cache panels, step indicators, hover tooltips, and CSS animations for hits/misses.

**Member 2 – Simulator Logic:** Cache operations including Direct-Mapped and Associative mapping, hit/miss detection, valid/dirty bits, and memory writes.

**Member 3 – Testing:** Sample memory access sequences, verifying hit/miss ratios, dirty bit updates, and ensuring correct memory write behavior.

## Challenges, Conclusion & References

**Challenges & Solutions:**

- Implementing **accurate cache behavior** including hits, misses, valid/dirty bits, and memory write-back logic.

- Designing a **clear and interactive UI** with animations and tooltips to visually explain cache operations.

### Conclusion:

The Cache Memory Simulator effectively demonstrates **Direct-Mapped and Associative cache behavior**.It provides **visual feedback for hits, misses, and memory writes**, making learning easier.Users can experiment with **different cache configurations** to observe performance changes.

Overall, it serves as a **practical educational tool** for understanding cache memory concepts in computer architecture.

## References:

1. **Computer Organization and Design: The Hardware/Software Interface — David A. Patterson & John L. Hennessy. This is a classic textbook for understanding memory hierarchy, cache design, and system architecture. O'Reilly Media**

2. **The Cache Memory Book (2nd Edition) — Jim Handy. A deep dive into cache architectures, mapping, replacement policies, and real-world design practices. Elsevier Shop+1**

3. **O'Reilly –** ***Computer Architecture, 5th Edition*** **by Hennessy & Patterson: includes a detailed review of memory hierarchy and cache performance optimization. O'Reilly Media+1**

4. ***The Essentials of Computer Organization & Architecture*** **— Linda Null & Julia Lobur: provides a clear, layered explanation of cache memory and organization. O'Reilly Media+1**

5. ***Code: The Hidden Language of Computer Hardware and Software*** **— Charles Petzold: helps in understanding how low-level hardware (bits, memory) works in a very accessible way.**