Memory Hierarchy Simulator

Fall 2024

The American University in Cairo

Aabed Elghadban 900223106

Abdulaziz Alhaidary 900225979

Omar Mokhtar 900223343

Introduction:

Memory caching is fundamental in computer architecture, significantly enhancing system performance by reducing data access times. This project focuses on developing a simple simulator for a memory caching system to analyze cache behavior and performance metrics.

The primary objective is to implement a simulator for a simplified read-only one-level direct-mapped cache within a 32-bit byte-addressable memory space (4 GB). Each memory access is assumed to take 100 clock cycles. The simulator tracks cache contents, including valid bits and tags, and records the number of accesses, hits, and misses to calculate the Average Memory Access Time (AMAT).

We utilized C++ as the main programming language due to its efficiency and control over system resources, which are essential for performance-critical applications like a caching simulator. C++'s robust standard library and support for object-oriented programming also facilitate a modular and maintainable codebase.

The simulator takes two main inputs: cache information (total cache size, cache line size, and cache access cycles) and an access sequence of memory addresses provided through a text file. By processing these inputs, the simulator emulates cache behavior and outputs the valid bits and tags of all cache entries, total number of accesses, hit and miss ratios, and the AMAT after each memory access.

Implementation Description

We developed a memory caching simulator in C++ that models a simplified read-only one-level direct-mapped cache within a 32-bit byte-addressable memory space (4 GB). The simulator handles separate instruction and data caches, allowing users to input cache size, line size, and access time. It processes memory access sequences from a text file, updating cache valid bits and tags, and tracking accesses, hits, misses, and calculating the Average Memory Access Time (AMAT). The application is console-based, with potential for future GUI enhancements as a bonus feature.

Design Decisions and Assumptions

- **Language Choice:** Utilized C++ for its performance and control over system resources.
- **Cache Architecture:** Implemented a direct-mapped, read-only one-level cache for simplicity.
- **Memory Addressing:** Assumed a 32-bit address space, ensuring addresses fit within 4 GB.

- **Input Handling:** Parsed memory accesses from a text file, distinguishing between instruction (I) and data (D) accesses.
- **Validation:** Ensured cache and line sizes are powers of two and validated cache access time between 1 and 10 cycles.
- **Initialization:** Started with empty caches, reflecting a cold cache state.

Known Bugs and Issues

- **Error Handling:** Abruptly exits on invalid access types without allowing recovery or user correction.
- **Scalability:** May experience performance issues with very large access sequences due to sequential processing.
- **Limited Functionality:** Supports only read operations, lacking write policies like write-through or write-back.

Bonus Features added:

- Web-based GUI for a more accessible and easy-to-use user experience.
- Supporting separate caches for instructions and data.

Simulator Usage Guide:

Navigate to the SourceCodes folder and run the file Run.cpp. Input the Instruction Cache Size (bytes), the Data Cache Size (bytes), the Cache Line Size (bytes), the Cache Access Time (cycles), and the Access Sequence file name (ending in .txt). Click Run Simulation and it should run like so:

Testing:

Test1:

Input:

00000000000000000000000000000000I

00000000000000000000000000000001I

00000000000000000000000000000010I

00000000000000000000000000000011I

00000000000000000000000000000100D

00000000000000000000000000000101D

00000000000000000000000000000110D

Instruction Cache Size: 16 bytes

Data Cache Size: 32 bytes

Cache Block Size: 8 bytes

Cache Access Latency: 2 cycles

Output:

```
Final Data Cache Status:
Cache Status Report:
Block 0: Active = 1, Tag = 000000000000000000000000000
Cache Details:
Cache Size: 32 bytes
Block Size: 8 bytes
Total Blocks: 4
Access Time per Block: 2 cycles
Memory Latency: 100 cycles
Total Requests: 3
Successful Hits: 2
Misses: 1
Hit Rate: 0.67
Miss Rate: 0.33
Average Memory Access Time: 35.33 cycles
```

Test2:

Input:

00000000000000000000000000000101I

00000000000000000000000000001010I

00000000000000000000000000001111I

00000000000000000000000000000101D

00000000000000000000000000010001D

00000000000000000000000000001111D

00000000000000000000000000010101I

00000000000000000000000000011010I

00000000000000000000000000010101D

00000000000000000000000000011010D

00000000000000000000000000011111I

Instruction Cache Size: 64 bytes

Data Cache Size: 128 bytes

Cache Block Size: 16 bytes

Cache Access Latency: 4 cycles

Output:

```
Final Data Cache Status:
Cache Status Report:
Block 0: Active = 1, Tag = 0000000000000000000000000
Cache Details:
Cache Size: 128 bytes
Block Size: 16 bytes
Total Blocks: 8
Access Time per Block: 4 cycles
Memory Latency: 100 cycles
Total Requests: 5
Successful Hits: 4
Misses: 1
Hit Rate: 0.80
Miss Rate: 0.20
Average Memory Access Time: 24.00 cycles
```

Test3:

Input:

00000000000000000000000000000000I

00000000000000000000000000000001I

00000000000000000000000000000010I

00000000000000000000000000000011I

00000000000000000000000000000100D

00000000000000000000000000000101D

00000000000000000000000000000110D

00000000000000000000000000010000I

00000000000000000000000000100000D

00000000000000000000000001000000I

00000000000000000000000010000000D

00000000000000000000000100000000I

00000000000000000000001000000000D

00000000000000000000010000000000I

00000000000000000000100000000000D

00000000000000000001000000000000I

00000000000000000010000000000000D

00000000000000000100000000000000I

00000000000000001000000000000000D

00000000000000010000000000000000I

00000000000000100000000000000000D

00000000000001000000000000000000I

00000000000010000000000000000000D

00000000000100000000000000000000I

00000000001000000000000000000000D

00000000010000000000000000000000I

0000000010000000000000000000000D

instruction Cache Size: 512 bytes

Data Cache Size: 1 KB

Cache Block Size: 64 bytes

Cache Access Latency: 5 cycles

Output:

```
Final Data Cache Status:
Cache Status Report:
Block 0: Active = 1, Tag = 0000000001000000000000
Block 1: Active = 1, Tag = 0000000000000000000000
Block 4: Active = 1, Tag = 0000000000000000000000
Cache Details:
Cache Size: 1024 bytes
Block Size: 64 bytes
Total Blocks: 16
Access Time per Block: 5 cycles
Memory Latency: 100 cycles
Total Requests: 13
Successful Hits: 3
Misses: 10
Hit Rate: 0.23
Miss Rate: 0.77
Average Memory Access Time: 81.92 cycles
```