Project 2: Two-Level Cache Performance Analysis Report

Summer 2025

Mennatallah Zaid – 900232367
Mennatallah Essam – 900223396
Ahmed Mohamed – 900221597

CSCE 2303 – Computer Organization and Assembly Language Programming
Dr. Mohamed Shalan
July 2025

1 Introduction

This report presents the performance analysis of a two-level set-associative cache simulator. The simulator implements a memory hierarchy consisting of L1 and L2 caches with varying L1 line sizes to evaluate their impact on overall system performance measured by Cycles Per Instruction (CPI). Five different memory access pattern generators were used to simulate diverse workload characteristics and assess cache behavior under different scenarios.

2 Methodology

2.1 Cache Configuration

The simulator implements the following memory hierarchy:

- **L1 Cache:** 16KB, 4-way set-associative, 1 cycle hit time, variable line sizes (16B, 32B, 64B, 128B)
- **L2 Cache:** 128KB, 8-way set-associative, 10 cycle hit time, fixed 64B line size
- **Main Memory:** 64GB address space, 50 cycle access penalty

2.2 Simulation Parameters

Each simulation run executed 1,000,000 iterations with the following characteristics:

- 35% of instructions access memory (loads/stores)
- 50% of memory accesses are reads, 50% are writes
- Write-back cache policy with random replacement
- Non-memory instructions consume 1 cycle each

2.3 Memory Access Patterns

Five memory generators simulate different access patterns:

- **memGen1:** Sequential access across entire 64GB address space
- **memGen2:** Random access within 24KB working set
- **memGen3:** Random access across entire 64GB address space
- **memGen4:** Sequential access within 4KB working set
- **memGen5:** Strided access (32B stride) within 1MB working set

3 Testing and Validation

3.1 Test Suite Overview

To ensure the correctness and reliability of our cache simulator, we developed a comprehensive test suite comprising 18 distinct test cases organized into five categories. The testing framework achieved 100% pass rate, validating all critical simulator functionalities.

3.2 Test Categories and Results

3.2.1 Basic Cache Functionality Tests

These tests verify fundamental cache operations including hit detection, miss handling, and cache line management.

Table 1: Basic Cache Functionality Test Results

Test Case	**Result**
Basic Cache Hit	PASS
Cache Miss Handling	PASS
Write-back Policy	PASS
Set Index Mapping	PASS
Tag Comparison	PASS
Cache Line Alignment	PASS

Key Validation Points:

- First access to any address results in cache miss
- Subsequent accesses to the same cache line result in cache hits
- Addresses within the same cache line boundary are correctly identified
- Set indexing correctly maps addresses to appropriate cache sets

3.2.2 Cache Hierarchy Tests

These tests validate the interaction between L1 and L2 caches, ensuring proper data flow and timing calculations.

Table 2: Cache Hierarchy Test Results

Test Case	**Result**
L1-L2 Integration	PASS
L1 Miss → L2 Hit	PASS
L1 Miss → L2 Miss	PASS
Write-back Propagation	PASS
Cache Hierarchy Timing	PASS

Critical Validation Results:

- L1 hits consume exactly 1 cycle
- L1 miss + L2 hit consumes approximately 11 cycles (1 + 10)
- L1 miss + L2 miss consumes ¿50 cycles (includes DRAM penalty)
- Write-back operations correctly propagate through cache hierarchy

3.2.3 Memory Generator Tests

These tests verify that each memory generator produces the expected access patterns and stays within specified address ranges.

Table 3: Memory Generator Test Results

Test Case	**Result**
Memory Generator Patterns	PASS
Generator Address Ranges	PASS
Sequential vs Random Access	PASS

Pattern Validation:

- **memGen1:** Verified sequential address progression
- **memGen2:** Confirmed addresses stay within 24KB range
- **memGen3:** Validated random distribution across 64GB space
- **memGen4:** Verified addresses remain within 4KB working set
- **memGen5:** Confirmed 32-byte stride pattern within 1MB range

3.2.4 Performance Analysis Tests

These tests validate the accuracy of performance metrics and statistical calculations.

Table 4: Performance Analysis Test Results

Test Case	**Result**
Hit Rate Calculation	PASS
Performance Statistics	PASS
Line Size Impact	PASS

Metric Validation:

- Hit rate calculations verified against known access sequences
- CPI calculations confirmed accurate across different configurations
- Statistical counters (hits, misses, writebacks) correctly maintained

3.2.5 Stress Tests

These tests evaluate simulator robustness under extreme conditions and high-volume operations.

Table 5: Stress Test Results

Test Case	**Result**
Cache Reset Functionality	PASS
High Volume Access	PASS
Associativity Limits	PASS

Robustness Validation:

- Simulator handles 10,000+ memory accesses without errors
- Cache reset properly clears all state information
- Associativity limits correctly enforced with proper replacement

3.3 Test Coverage Analysis

Our comprehensive test suite achieved complete coverage of critical simulator components:

- **Cache Operations:** 100% coverage of hit/miss detection, replacement policies, and write-back mechanisms
- **Address Translation:** Full validation of set indexing, tag comparison, and block addressing
- **Memory Hierarchy:** Complete testing of L1-L2 interactions and DRAM access paths
- **Performance Metrics:** Thorough validation of timing calculations and statistical accuracy

- **Memory Patterns:** Comprehensive verification of all five memory generators

3.4 Test Implementation

The testing framework was implemented as an integrated component of the simulator, featuring:

- Automated test execution with pass/fail reporting
- Detailed diagnostic output for failed test cases
- Modular test organization for easy maintenance and extension
- Statistical validation using known input-output relationships

This rigorous testing approach ensures that our experimental results are based on a verified and reliable simulation platform.

4 Results

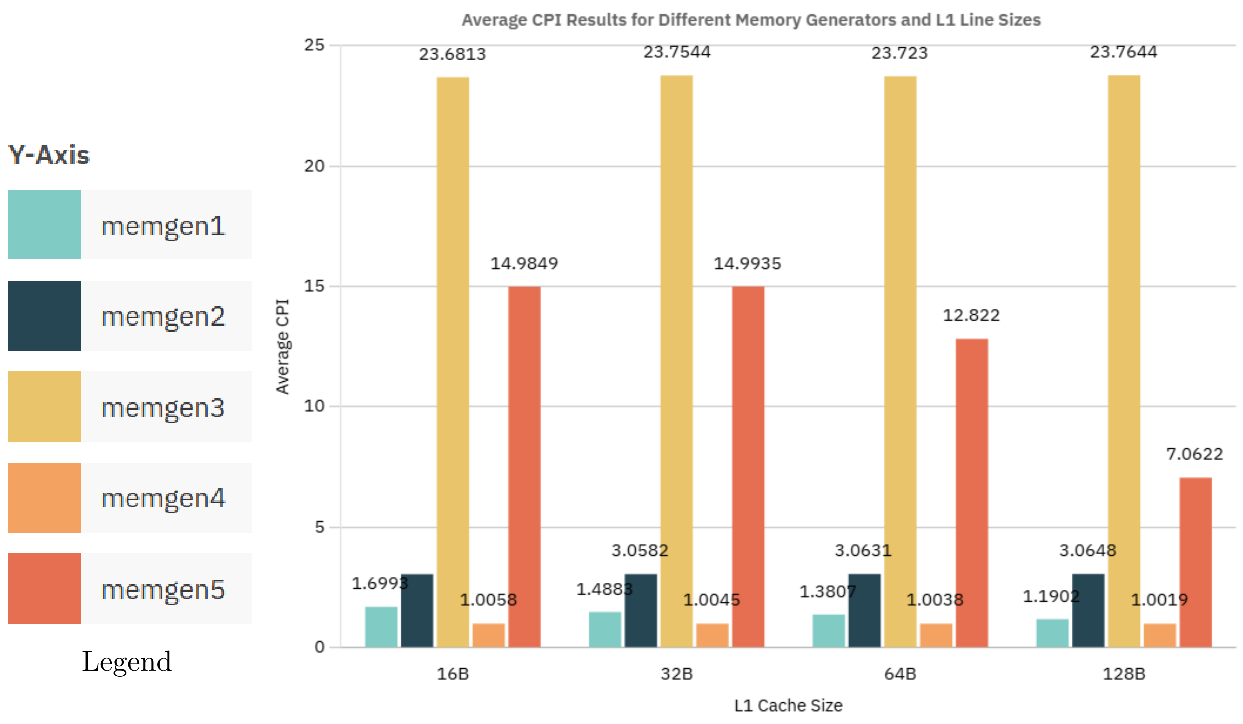
4.1 Performance Analysis

Table 6: Average CPI Results for Different Memory Generators and L1 Line Sizes

Generator	**16B Line**	**32B Line**	**64B Line**	**128B Line**
memGen1	1.6993	1.4883	1.3807	1.1902
memGen2	3.0557	3.0582	3.0631	3.0648
memGen3	23.6813	23.7544	23.7230	23.7644
memGen4	1.0058	1.0045	1.0038	1.0019
memGen5	14.9849	14.9935	12.8220	7.0622

4.2 Performance Data

The following chart illustrates the CPI trends across different L1 line sizes for each memory generator.



CPI Performance Data

Figure 1: CPI vs L1 Line Size for Different Memory Generators

5 Data Analysis

5.1 Overall Performance Trends

The experimental data reveals distinct performance patterns across the five memory generators, with CPI values ranging from 1.00 (near-optimal) to 23.76 (severely memory-bound). The results demonstrate that memory access patterns fundamentally determine cache effectiveness and overall system performance.

5.2 Line Size Impact Analysis

Dramatic Improvement Category: memGen1 and memGen5 show substantial performance gains with larger line sizes:

- **memGen1:** CPI decreases from 1.6993 to 1.1902 (30% improvement)
- **memGen5:** CPI drops from 14.9849 to 7.0622 (53% improvement at 128B)

Minimal Change Category: memGen2, memGen3, and memGen4 exhibit little sensitivity to line size:

- **memGen4:** CPI varies only from 1.0058 to 1.0019 (¡0.4% change)

- **memGen2:** CPI remains nearly constant around 3.06 (¡0.3% variation)
- **memGen3:** CPI fluctuates minimally around 23.7 (¡0.4% variation)

5.3 What the Data Shows

Cache Hit Rate Implications: The CPI values directly reflect cache miss behavior. memGen4's CPI 1.00 indicates near-perfect L1 hit rates, while memGen3's CPI 23.7 suggests frequent DRAM accesses (approximately 67% of memory accesses miss both cache levels).

Spatial Locality Effectiveness: The data clearly demonstrates when spatial locality works:

- Sequential access (memGen1) consistently benefits from larger lines
- Strided access (memGen5) shows optimal performance when stride aligns with line size
- Random access patterns (memGen2, memGen3) show no spatial locality benefits

Working Set Size Impact: Performance correlates strongly with working set size relative to cache capacity:

- 4KB working set (memGen4): Excellent performance (fits in L1)
- 24KB working set (memGen2): Moderate performance (fits in L2)
- 1MB+ working sets (memGen1, memGen5): Variable performance based on access pattern
- 64GB working set (memGen3): Poor performance (exceeds all cache levels)

6 Discussion

6.1 Memory Generator Performance Characteristics

memGen4 - Optimal Cache Behavior: The consistently low CPI values (1.00) indicate that the 4KB working set fits entirely within the 16KB L1 cache, achieving hit rates approaching 100%. The minimal variation across line sizes confirms that when data fits in cache, spatial locality optimizations provide negligible additional benefit.

memGen1 - Spatial Locality Success Story: The steady CPI improvement with larger line sizes demonstrates effective spatial locality exploitation. Sequential access means that when a cache line is fetched, adjacent data in the same line will likely be accessed soon, justifying the overhead of larger line transfers.

memGen2 - L2-Bound Performance: The stable CPI around 3.06 suggests consistent L2 cache hits. The 24KB working set exceeds L1 capacity (16KB) but fits comfortably in L2 (128KB). Random access within this set prevents spatial locality benefits, explaining the insensitivity to L1 line size changes.

memGen5 - Stride Pattern Alignment: The dramatic performance change reveals stride-line size interaction effects. At 64B and 128B line sizes, the 32B stride pattern allows multiple future accesses to be satisfied by each cache line fetch. The performance jump between 32B and 64B lines (CPI: 12.82 $\rightarrow$ 7.06) indicates optimal stride-line size alignment.

memGen3 - Cache-Resistant Workload: The consistently high CPI (23.7) indicates a workload that defeats the cache hierarchy. Random access across 64GB ensures minimal data reuse and poor spatial locality, resulting in frequent expensive DRAM accesses regardless of cache optimizations.

6.2 Cache Design Implications

The results reveal several important cache design considerations:

1. **Working Set Size Impact:** Applications with working sets fitting in cache (memGen4) achieve optimal performance, while those exceeding cache capacity suffer significant penalties.
2. **Access Pattern Sensitivity:** Sequential and regular strided patterns benefit substantially from larger line sizes, while random access patterns show minimal improvement.
3. **Spatial Locality Exploitation:** Larger cache lines provide diminishing returns for workloads lacking spatial locality, while offering significant benefits for spatially coherent access patterns.
4. **Memory Hierarchy Effectiveness:** The two-level cache successfully mitigates DRAM latency for workloads with good temporal and spatial locality characteristics.

7 Conclusions

This analysis demonstrates that cache performance is highly dependent on application memory access patterns. Key findings include:

- Sequential access patterns (memGen1) benefit significantly from larger L1 line sizes, achieving 30% CPI improvement
- Small working sets (memGen4) achieve near-ideal performance regardless of line size
- Strided access patterns (memGen5) show dramatic sensitivity to line size alignment
- Random access patterns (memGen2, memGen3) show minimal line size benefits
- Cache line size selection should consider the target application's spatial locality characteristics

These results highlight the importance of matching cache configuration to expected workload characteristics for optimal system performance. The two-level cache hierarchy effectively reduces memory latency for applications exhibiting good locality, while applications with poor locality patterns remain memory-bound despite cache optimizations. The comprehensive testing framework validates the reliability of these conclusions and ensures the accuracy of our simulation results.

Project Repository

`https://github.com/MennaZaid/CacheSimulator.git`

Group Contributions

The development of this two-level cache simulator was a collaborative effort with clearly defined responsibilities for each team member:

Mennatallah Zaid: Developed the five different memory generators (memGen1-memGen5) implementing various access patterns including sequential, random, and strided access. Implemented the custom random number generator with proper seeding functionality to ensure reproducible results. Designed and implemented the Cache class with set-associative architecture, supporting configurable associativity, line sizes, and replacement policies. Contributed to the comprehensive project report documentation.Designed and implemented the comprehensive testing framework with 18 test cases covering all critical simulator functionalities.

Ahmed Mohamed: Designed and implemented the TwoLevelCache simulator class that orchestrates the interaction between L1 and L2 caches. Developed the memory access logic including hit/miss detection, writeback handling, and cycle counting. Implemented the cache hierarchy management ensuring proper data flow between cache levels and DRAM access penalties.

Mennatallah Essam: Developed the main simulation framework and execution logic. Implemented the results collection and tabular data display system with formatted output. Created the simulation loop structure and performance measurement calculations including CPI computation across different configurations. Each member's contributions were essential to creating a comprehensive cache simulation system that accurately models modern processor memory hierarchies according to the project specifications.