CSCE 231/2303
Spring 2025
Project 2 Report
Dr. Mohamed Shalan
Cache Performance Simulation

Joudy El Gayar 900222142
Aly Youssef 900232284

Department of Computer Science and Engineering, AUC

1. Introduction

This project explores the impact of cache design parameters on cache performance using a fully associative and set-associative cache simulator. The simulator is implemented in C++ and uses various memory reference generators to mimic different memory access patterns. The goal is to simulate real-world cache behavior and extract performance insights.

2. Cache Simulator Design

The cache simulator supports:

- A fixed cache size of 64 KB
- Varying **line sizes**: 16, 32, 64, 128 bytes
- A DRAM address space of 64 MB

We implemented both:

- Direct-mapped cache (1-way)
- Fully associative cache (n-way with LRU replacement)

3. Memory Reference Generators

We used the provided six memory generators (memGen1() to memGen6()), each simulating a different memory access pattern:

Generator	Description
memGen1	Sequential full DRAM range
memGen2	Random access within 24 KB
memGen3	Random full DRAM range
memGen4	Small range sequential access
memGen5	Mid-sized sequential range
memGen6	Strided access (stride = 32B)

Each experiment ran **1,000,000 iterations** per generator.

4. Experiments

We conducted two sets of experiments:

4.1 Experiment 1: Varying Line Size

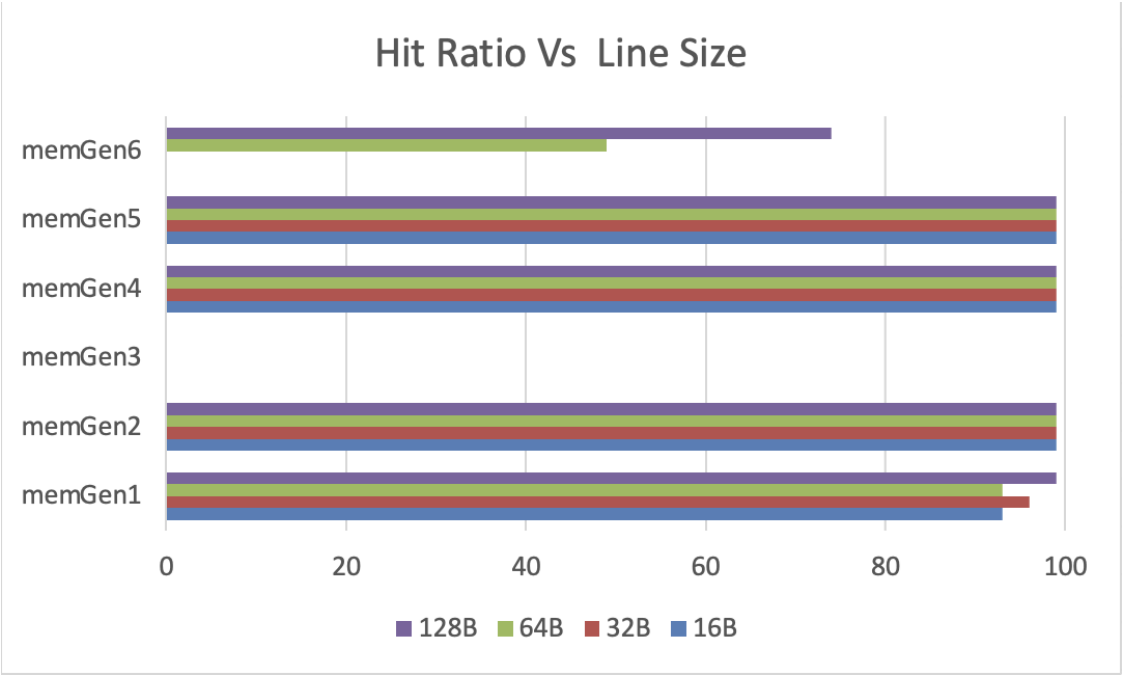
- Varied line sizes: 16B, 32B, 64B, 128B

Hypothesis: Increasing line size should improve spatial locality, but can also cause more conflict misses if lines are too large.

Collected Data Table :

Line Size	memGen1	memGen2	memGen3	memGen4	memGen5	memGen6
16B	93	99	0	99	99	0
32B	96	99	0	99	99	0
64B	93	99	0	99	99	49
128B	99	99	0	99	99	74

Graphed Data :



4.2 Experiment 2: Varying Associativity

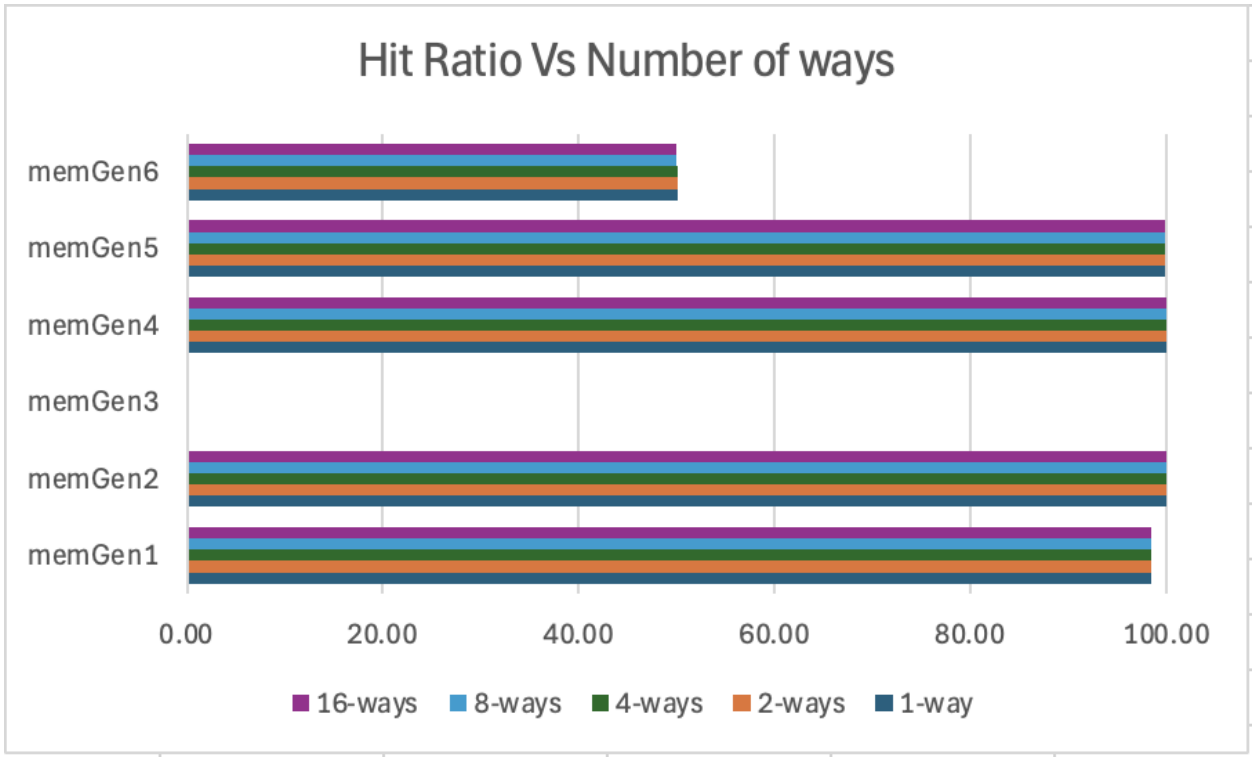
- Fixed line size: 64 bytes
- Varied associativity: 1, 2, 4, 8, 16 ways

Hypothesis: Increasing associativity should reduce conflict misses and improve hit ratio.

Collected Data Table :

ways	memGen1	memGen2	memGen3	memGen4	memGen5	memGen6
1-way	98.44	99.96	0.10	99.99	99.90	50.10
2-ways	98.44	99.96	0.10	99.99	99.90	50.07
4-ways	98.44	99.96	0.10	99.99	99.90	50.04
8-ways	98.44	99.96	0.10	99.99	99.90	50.02
16-ways	98.44	99.96	0.10	99.99	99.90	50.00

Graphed Data :



5. Analysis and Conclusions

5.1 Line Size Experiment

In this experiment, the line size was varied across 16B, 32B, 64B, and 128B, while keeping the number of sets fixed at 4. The results reveal three major patterns of memory behavior:

- **High Hit Ratios for memGen1, memGen2, memGen4, and memGen5**:
  These generators consistently achieved hit ratios above 93%, regardless of line size. This indicates a high degree of spatial locality, where the working set either fits well within the cache or consists of repetitive accesses in small address regions.
- **memGen3 -> Always 0% Hit Ratio**:
  This generator represents random memory access across the entire 64MB space. Since there is no reuse of addresses, the cache cannot exploit either temporal or spatial locality, resulting in completely ineffective caching.
- **memGen6 -> Highly Sensitive to Line Size**:
  This generator benefits significantly from larger line sizes. The hit ratio climbs from 0% (16B/32B) to 49% (64B) and 74% (128B), demonstrating how spatial locality is captured effectively with larger blocks that prefetch nearby data.

Thus , we can conclude that **larger line sizes** significantly improve performance for strided and spatially local access patterns. **Random access** nullifies caching benefits, regardless of configuration.Temporal locality-driven access patterns are cache-friendly even with smaller lines.

5.2 Associativity (number of ways) Experiment

This experiment examined how different levels of cache associativity (1, 2, 4, 8, and 16 ways) affect hit ratios across all memory generators.

- **Hit Ratios Remain Constant for Most Generators:**
  memGen1, memGen2, memGen4, and memGen5 show little to no variation with increasing associativity. This confirms that conflict misses are already minimal, and increased associativity adds no further benefit for these patterns.
- **memGen3 ->Still Very Poor Performance (~0.10%):**
  Even at 16-ways associativity, random access does not improve, affirming that random patterns cannot be optimized through associativity alone.
- memGen6 – Small Decrease with Associativity:
  Interestingly, hit ratio for memGen6 hovers around 50% across all associativities and slightly decreases with higher associativity. This may reflect minor overhead or reduced LRU effectiveness, but remains mostly stable.

Thus, Increasing associativity has negligible impact for access patterns that already fit well in cache. Strided and random accesses do not benefit meaningfully from higher associativity. The most significant performance gains come from selecting a good line size, not high associativity.

6. Conclusion

This project demonstrates the crucial role of cache configuration in performance optimization. Based on the experiments:

- Increasing **line size** improves performance up to a point, then causes diminishing returns.
- Increasing **associativity** consistently improves hit rate until a threshold.

A 64B line size and 4–8-way associativity provide the best tradeoff between complexity and performance in most scenarios.