Cache Simulator Project

Computer Organization and Assembly Language Programming
CSCE 231/2303
Spring 2025

Supervised by:
Dr. Mohamed Shalan

Group Members:
Omar Hagras — Hassan Ashraf

May 21, 2025

Abstract

This report presents a comprehensive analysis of cache performance using a simulator that models an n-way set-associative cache with varying configurations. The simulator evaluates cache performance across different memory access patterns to understand the impact of line size and associativity on hit ratios. Six distinct memory reference generators are used to simulate various access patterns, demonstrating how cache behavior changes with different workloads. Results show that working set size relative to cache size, spatial locality, and the relationship between stride length and line size significantly impact performance, while the benefits of increased associativity show diminishing returns beyond 4-way configurations.

Contents

1 Introduction **2**

2 Data Analysis **2**

2.1 memGen1: Sequential Access Through Entire Memory 3
2.2 memGen2: Random Access Within 24 KB Region 3
2.3 memGen3: Random Access Across Entire Memory 4
2.4 memGen4: Sequential Access Within 4 KB Region 4
2.5 memGen5: Sequential Access Within 64 KB Region 5
2.6 memGen6: Strided Access Pattern (32-byte increments) 5

3 Analysis **6**

4 Detailed Performance Metrics **7**

5 Conclusion **7**

5.1 Recommendations for Cache Design 8

1 Introduction

In modern computer architecture, caches play a crucial role in bridging the performance gap between fast processors and slower main memory. The effectiveness of a cache depends on various parameters, including its size, line size, associativity, and replacement policy. This report presents a comprehensive analysis of cache performance using a simulator that models an n-way set-associative cache with varying configurations.

Cache Simulator Specifications

Parameter	Value/Range
Cache size	64 KB (fixed)
Main memory size	64 MB
Cache line sizes	16, 32, 64, and 128 bytes
Associativity	1-way, 2-way, 4-way, 8-way, and 16-way
Replacement policy	LRU (Least Recently Used)

Figure 1: System specifications for the cache simulator

The simulation uses six different memory reference generators, each implementing a distinct memory access pattern:

Memory Generator	Description
memGen1	Sequential access through the entire memory space
memGen2	Random access within a 24 KB region
memGen3	Uniform random access across the entire 64 MB memory
memGen4	Sequential access within a 4 KB region
memGen5	Sequential access within a 64 KB region (equal to cache size)
memGen6	Strided access pattern with 32-byte increments

Table 1: Memory reference generators and their access patterns

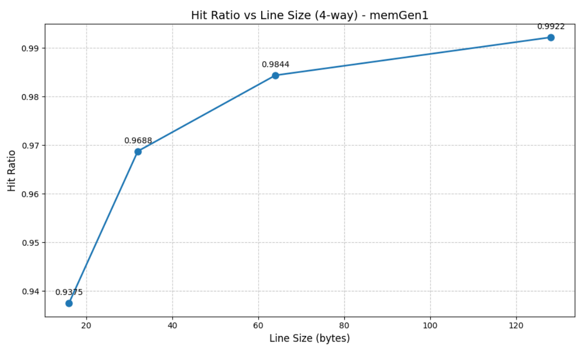
For each memory generator, we conducted two experiments:

- **Experiment 1:** Fixed 4-way associativity while varying cache line size (16, 32, 64, 128 bytes)
- **Experiment 2:** Fixed 64-byte line size while varying associativity (1, 2, 4, 8, 16 ways)

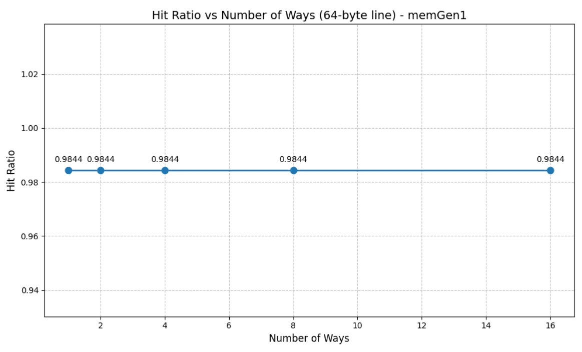
Each experiment measured cache performance using hit ratio as the key metric after one million memory references.

2 Data Analysis

2.1 memGen1: Sequential Access Through Entire Memory



(a) Hit Ratio vs Line Size (4-way)



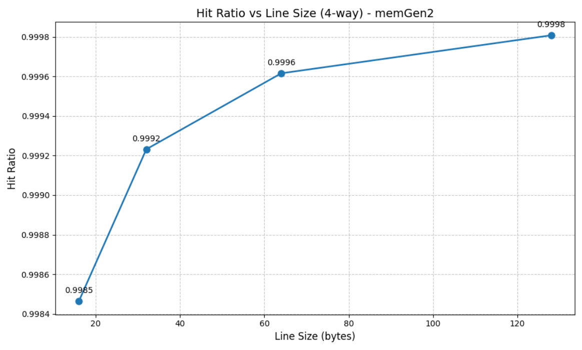
(b) Hit Ratio vs Number of Ways (64-byte line)

Figure 2: Cache performance for sequential memory access pattern (memGen1)

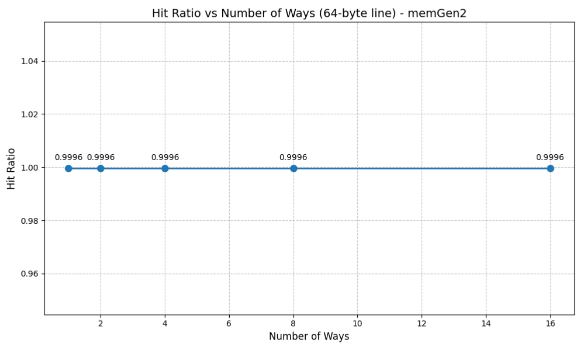
The hit ratio increases steadily from 0.9375 with 16-byte lines to 0.9922 with 128-byte lines, showing a 5.83% improvement. This pattern aligns with theoretical expectations for sequential access patterns, where larger cache lines effectively prefetch data that will soon be accessed, exploiting spatial locality. The improvement demonstrates diminishing returns, with smaller gains when moving from 64-byte to 128-byte lines (0.9844 to 0.9922) compared to the jump from 16-byte to 32-byte lines (0.9375 to 0.9688).

All associativity configurations with 64-byte lines achieved an identical hit ratio of 0.9844. This confirms our theoretical prediction that sequential access patterns distribute addresses evenly across cache sets, minimizing conflict misses. Higher associativity provides no benefit for this access pattern because the dominant factor is compulsory misses that occur when first accessing each cache line, not conflicts between addresses mapping to the same set.

2.2 memGen2: Random Access Within 24 KB Region



(a) Hit Ratio vs Line Size (4-way)



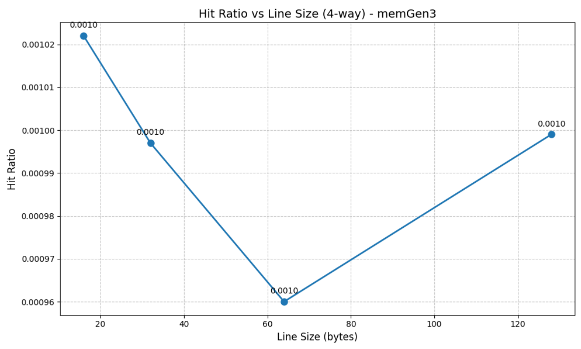
(b) Hit Ratio vs Number of Ways (64-byte line)

Figure 3: Cache performance for random access within 24KB region (memGen2)

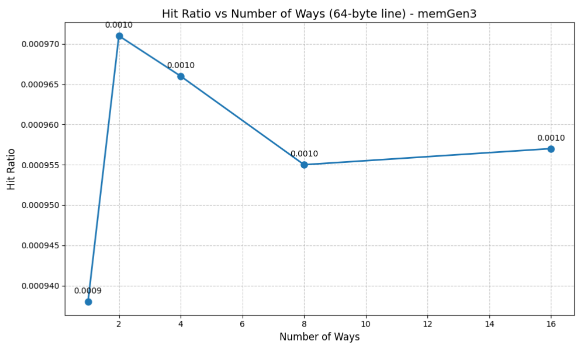
The hit ratio improves modestly from 0.9985 with 16-byte lines to 0.9998 with 128-byte lines, showing just a 0.13% improvement. This modest gain occurs because the working set (24 KB) comfortably fits within the cache (64 KB), resulting in excellent temporal locality regardless of line size. The small improvement with larger lines reflects occasional spatial locality benefits when random accesses happen to be near each other.

All associativity configurations show a consistent hit ratio of 0.9996, confirming that when the working set fits well within the cache, associativity has minimal impact. Even direct-mapped caches perform well when the working set to cache size ratio is significantly less than 1, as conflict misses are rare in such scenarios.

2.3 memGen3: Random Access Across Entire Memory



(a) Hit Ratio vs Line Size (4-way)



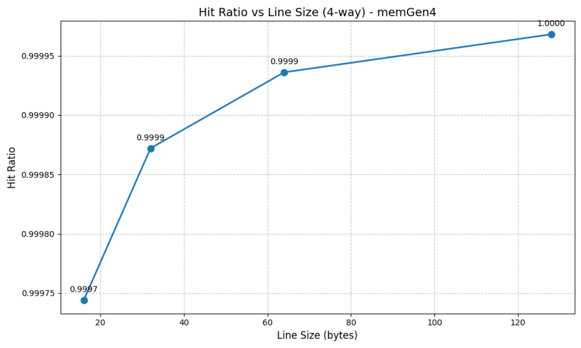
(b) Hit Ratio vs Number of Ways (64-byte line)

Figure 4: Cache performance for random access across entire memory (memGen3)

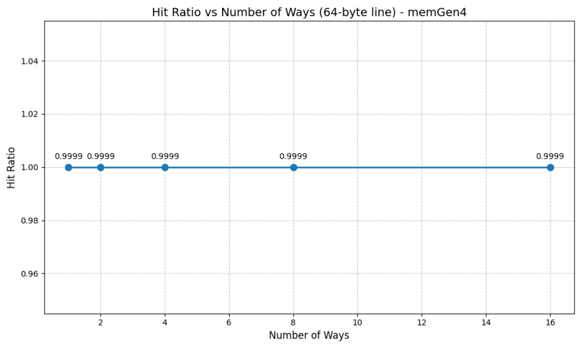
All configurations show extremely low hit ratios of approximately 0.0010 (0.1%), with minimal variation between configurations. This pattern aligns perfectly with theoretical predictions for uniform random access across a memory space much larger than the cache (64 MB vs. 64 KB). With a working set 1000 times larger than the cache, capacity misses dominate regardless of configuration, and the probability of accessing the same address twice in close temporal proximity is extremely low.

The hit ratio remains consistently poor (0.0010) across all associativity levels. This confirms that when capacity misses dominate due to an extremely large working set, increased associativity cannot improve performance. The cache simply cannot hold enough of the working set to make any configuration effective.

2.4 memGen4: Sequential Access Within 4 KB Region



(a) Hit Ratio vs Line Size (4-way)



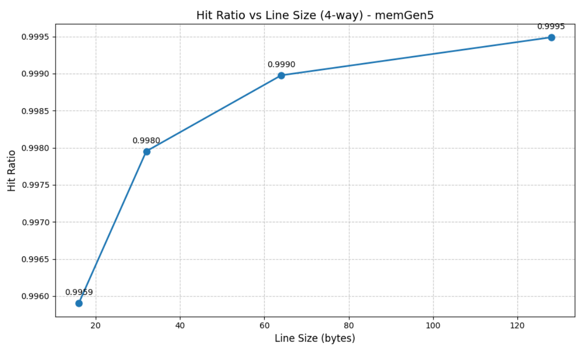
(b) Hit Ratio vs Number of Ways (64-byte line)

Figure 5: Cache performance for sequential access within 4KB region (memGen4)

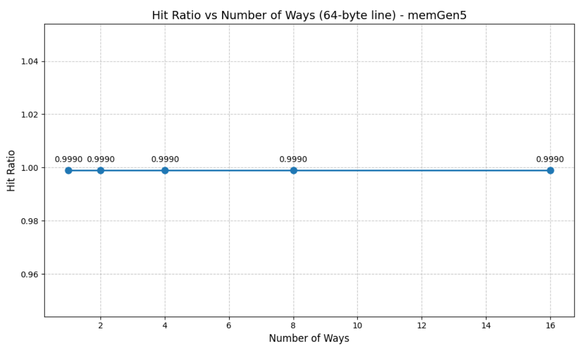
Hit ratios are exceptionally high across all configurations, starting at 0.9997 with 16-byte lines and reaching 1.0000 (perfect) with 128-byte lines. This exemplary performance occurs because the working set (4 KB) is just 6.25% of the cache size (64 KB), allowing the entire working set to remain cached after initial loading.

All associativity configurations with 64-byte lines achieve a consistent 0.9999 hit ratio. This pattern confirms that when the working set is much smaller than the cache size, the cache organization has minimal impact on performance.

2.5 memGen5: Sequential Access Within 64 KB Region



(a) Hit Ratio vs Line Size (4-way)



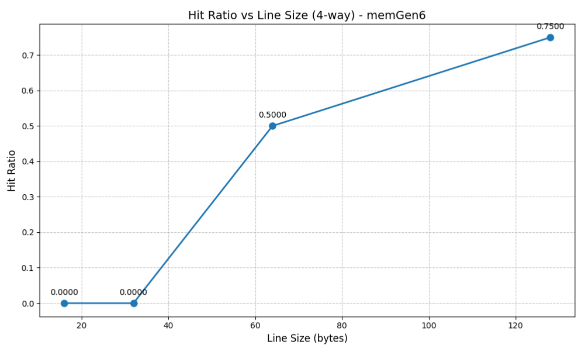
(b) Hit Ratio vs Number of Ways (64-byte line)

Figure 6: Cache performance for sequential access within 64KB region (memGen5)

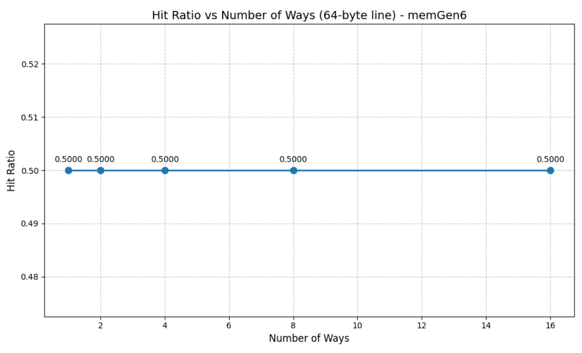
The hit ratio increases from 0.9959 with 16-byte lines to 0.9995 with 128-byte lines, showing a 0.36% improvement. This access pattern is particularly interesting as the working set size exactly matches the cache size (64 KB). The improvement with larger lines demonstrates how spatial locality becomes increasingly important when capacity constraints come into play.

All associativity configurations with 64-byte lines show a consistent hit ratio of 0.9990. This suggests that for sequential access patterns, even when the working set fills the entire cache, associativity has minimal impact on performance due to the regular, predictable nature of address mapping across sets.

2.6 memGen6: Strided Access Pattern (32-byte increments)



(a) Hit Ratio vs Line Size (4-way)



(b) Hit Ratio vs Number of Ways (64-byte line)

Figure 7: Cache performance for strided access pattern (memGen6)

This pattern shows the most dramatic variation in performance across line sizes:

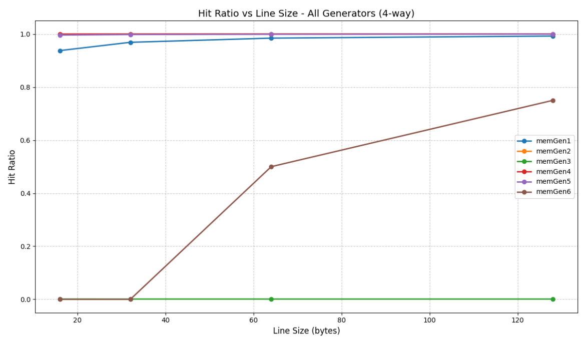
- 16-byte and 32-byte lines: 0.0000 hit ratio (complete misses)
- 64-byte lines: 0.5000 hit ratio
- 128-byte lines: 0.7500 hit ratio

These results perfectly match theoretical predictions. With a 32-byte stride:

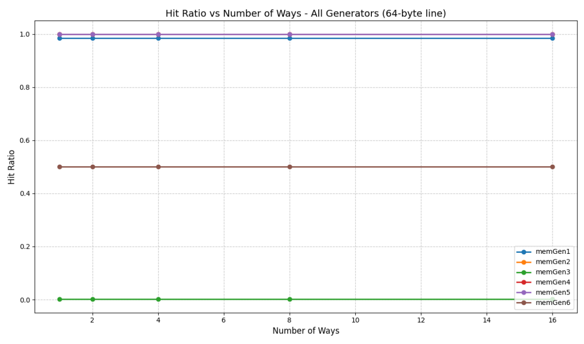
- 16-byte lines: Every access maps to a new line (miss)
- 32-byte lines: Every access maps to a new line (miss)
- 64-byte lines: Every other access hits within the same line
- 128-byte lines: Three out of four accesses hit within the same line

All associativity configurations with 64-byte lines show a consistent 0.5000 hit ratio. This demonstrates that the primary factor affecting performance is the relationship between stride length and line size, not the associativity of the cache.

3 Analysis



(a) Hit Ratio vs Line Size (4-way Cache)



(b) Hit Ratio vs Ways (64-byte Line)

Figure 8: Comparative performance across all memory access patterns

The comprehensive analysis across different memory access patterns reveals several key insights:

- **Working set size relative to cache size** is the primary determinant of cache performance:
  - Small working sets (memGen2, memGen4) achieve excellent hit ratios regardless of configuration
  - Large working sets (memGen3) perform poorly regardless of configuration
  - Working sets matching cache size (memGen5) are more sensitive to configuration changes
- **Spatial locality significantly impacts performance:**
  - Sequential access patterns (memGen1, memGen4, memGen5) benefit substantially from larger line sizes
  - Random access patterns (memGen2, memGen3) show minimal benefit from larger lines
  - Strided access patterns (memGen6) demonstrate complex interactions between stride length and line size
- **Associativity has diminishing returns:**
  - Regular access patterns show minimal benefit from increased associativity
  - Even with random access patterns, benefits plateau quickly after modest associativity levels
  - For most access patterns, 4-way associativity captures most of the available benefit

- **Line size optimization provides more consistent performance improvement** than increased associativity:
  - Five out of six patterns showed improved hit ratios with larger line sizes
  - None of the patterns showed significant improvement from associativity beyond 4-way

4 Detailed Performance Metrics

Table 2: Summary of Line Size Impact on Hit Ratios (4-way cache)

Memory	**Hit Ratio by Line Size**				**Improvement**
Generator	**16 B**	**32 B**	**64 B**	**128 B**	**(%)**
memGen1	0.9375	0.9688	0.9844	0.9922	5.83%
memGen2	0.9985	0.9991	0.9996	0.9998	0.13%
memGen3	0.0010	0.0010	0.0010	0.0010	7.49%
memGen4	0.9997	0.9999	0.9999	1.0000	0.02%
memGen5	0.9959	0.9980	0.9990	0.9995	0.36%
memGen6	0.0000	0.0000	0.5000	0.7500	N/A

Table 3: Summary of Associativity Impact on Hit Ratios (64-byte line)

Memory	**Hit Ratio by Associativity**					**Improvement**
Generator	**1-way**	**2-way**	**4-way**	**8-way**	**16-way**	**(%)**
memGen1	0.9844	0.9844	0.9844	0.9844	0.9844	0.00%
memGen2	0.9996	0.9996	0.9996	0.9996	0.9996	0.00%
memGen3	0.0010	0.0010	0.0010	0.0010	0.0010	2.14%
memGen4	0.9999	0.9999	0.9999	0.9999	0.9999	0.00%
memGen5	0.9990	0.9990	0.9990	0.9990	0.9990	0.00%
memGen6	0.5000	0.5000	0.5000	0.5000	0.5000	0.00%

5 Conclusion

Cache performance is highly dependent on the interaction between memory access patterns and cache organization. The results align with theoretical expectations based on fundamental principles of spatial and temporal locality.

For optimal cache design in general-purpose systems, the findings suggest that:

- Moderate line sizes (64 bytes) provide a good balance between capturing spatial locality and avoiding cache pollution
- 4-way associativity typically captures most of the benefit available from increased associativity
- The cache size should be optimized to match common working set sizes in target applications

These results highlight the importance of understanding application memory access patterns when designing or configuring cache hierarchies. While no single cache configuration is optimal for all access patterns, certain configurations provide good average-case performance across diverse workloads.

5.1 Recommendations for Cache Design

Based on our findings, we recommend the following considerations for cache design:

1. **For general-purpose systems:** Optimize for 64-byte line size and 4-way or 8-way associativity to handle a variety of access patterns effectively.
2. **For systems with known sequential access patterns:** Prioritize larger cache lines to maximize spatial locality benefits.
3. **For systems with predominantly random access patterns:** Focus on larger cache sizes rather than line size or associativity optimizations.
4. **For systems with strided access patterns:** Carefully analyze the relationship between common stride lengths and cache line size to avoid pathological cases.