

University of South Florida

CDA 4205L Computer Architecture Lab

Lab Report

Semester: Spring 2025		
Experiment	<i>Number:</i>	12
	<i>Date:</i>	04/09/2025
Lab	<i>Section:</i>	04
	<i>Lab TA:</i>	Rupal Agarwal
Report	<i>Due Date:</i>	04/16/2025
Group	<i>Member #1 name:</i>	Claude Watson
	<i>Member #2 name:</i>	Anirudh Kasyap Sesham

RUBRIC SUMMARY

[100%] Final submission/report

ABSTRACT:

As our journey into the realm of computer architecture deepened, we found ourselves standing at the crossroads of memory hierarchies and performance optimization. Lab 12 beckoned us to explore the intricate world of cache configurations through the lens of a matrix multiplication program. We ventured through landscapes of direct-mapped caches with varying dimensions, navigated the layered territories of n-way set associative structures, and finally traversed the boundless domains of fully associative caches. Our expedition yielded treasures of empirical data that illuminated the delicate balance between miss rates and hardware complexity. The narrative that unfolded from our graphs and tables revealed a compelling truth: while larger caches with greater associativity often chart paths to better performance, the wisdom of optimal design lies in finding harmony between hit rates and implementation costs. As our adventure concluded, we emerged with newfound appreciation for the elegant compromises embodied in modern cache architectures.

INTRODUCTION:

Cache memory serves as a critical component in modern computer architectures, addressing the performance gap between processor speeds and main memory access times. This lab explores how different cache configurations impact system performance by simulating various cache designs and measuring their effectiveness when running a matrix multiplication algorithm. We investigate three primary cache organizations: direct-mapped, n-way set associative, and fully associative, along with multiple cache sizes and block sizes for each configuration. Through this exploration, we aim to understand the principles of temporal and spatial locality that underpin effective cache design, and how various design parameters create tradeoffs between performance, hardware complexity, and power consumption. This hands-on experience provides practical insight into the design space exploration process used by computer architects when optimizing memory hierarchies for specific applications.

METHOD:

Software Tools Used:

For this lab, we utilized the RISC-V simulator provided by our course, which includes an integrated Data Cache Simulator tool. This simulator allows configuration of various cache parameters including cache size, block size, associativity, and replacement policies. We also used spreadsheet software to record our experimental data and generate plots showing the relationships between different cache configurations and their corresponding miss rates. Additionally, we employed a text editor to modify the matrix multiplication assembly code as needed and document our findings.

Files:

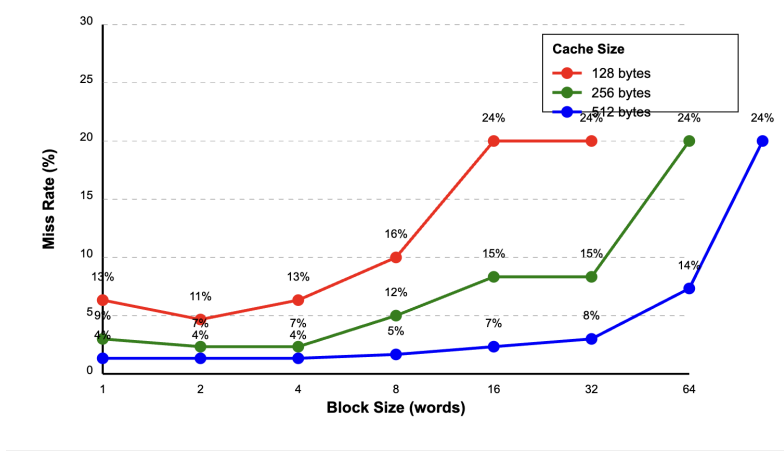
The primary file used in this lab was matmul.asm, a RISC-V assembly implementation of matrix multiplication that we downloaded from Canvas. This file contains code for multiplying two 10×10 matrices stored in the data segment.

RESULTS:

Task 1:
table for Direct mapping we have used data we have collected from RARS

Direct Mapping table					
#	Cache Size (bytes)	#	Block Size (words)	#	Number of Blocks
	128		1		32
	128		2		16
	128		4		8
	128		8		4
	128		16		2
	128		32		1
	256		1		64
	256		2		32
	256		4		16
	256		8		8
	256		16		4
	256		32		2
	256		64		1
	512		1		128
	512		2		64
	512		4		32
	512		8		16
	512		16		8
	512		32		4
	512		64		2
	512		128		1

Direct Mapped Cache Miss Rate vs Block Size




Task 2:

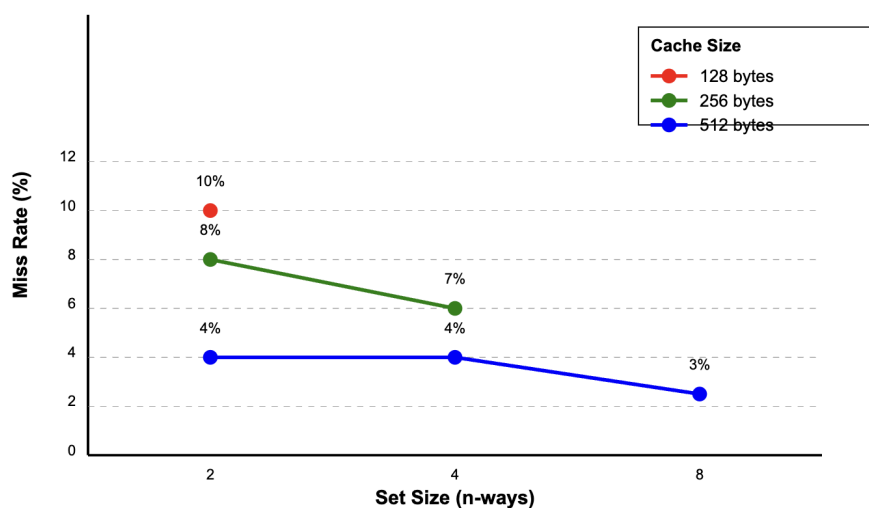
The block size significantly impacts miss rates for direct mapped caches due to the tradeoff between spatial locality and cache utilization. For smaller cache sizes (128 bytes), the miss rate is minimized at 2-word blocks (11%), while larger caches can maintain low miss rates up to 4-8 word blocks. As block size increases, more memory addresses map to fewer cache locations, increasing conflict misses. Additionally, larger blocks fetch more data than needed, wasting cache space when only a small portion of the block is used. This explains why very large block sizes (32-128 words) result in high miss rates regardless of cache size. The optimal block size balances exploiting spatial locality (accessing nearby data) with minimizing conflict misses.

Task 3:

table for N-way set associative cache we have used data we have collected from RARS

Nway set associative map... 						
Cache Size (bytes)	Number of blocks	Block Size (words)	Set Size (n-ways)	Hit Rate (%)	Miss Rate (%)	
128	4	8	2	90	10	
256	8	8	2	92	8	
256	8	8	4	93	7	
512	16	8	2	96	4	
512	16	8	4	96	4	
512	16	8	8	97	3	

N-Way Set Associative Cache Miss Rate vs Set Size



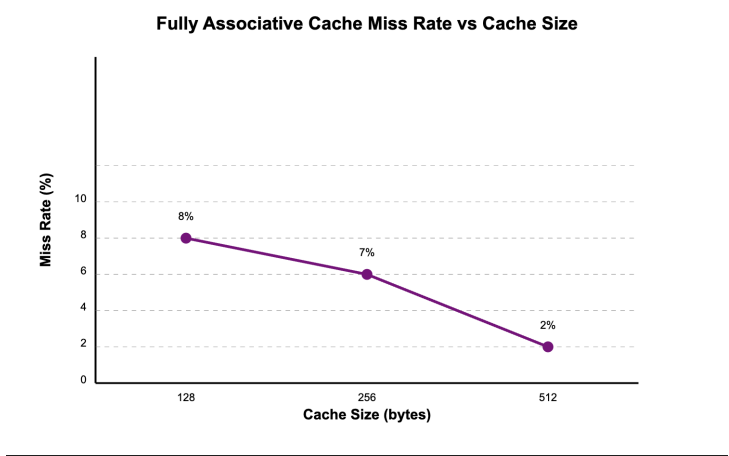
Task 4:

Considering hardware requirements and performance benefits, the optimal n-way set associative configuration appears to be 4-way for the 256-byte and 512-byte caches. While 8-way associativity for the 512-byte cache provides a marginal improvement (3% vs 4% miss rate), it requires significantly more hardware complexity. Each increase in associativity requires additional comparators for tag matching, more complex replacement logic, and higher power consumption. The 4-way configuration offers a good balance between performance and hardware cost, particularly for the 256-byte cache where it achieves a reasonable 7% miss rate. Beyond 4-way, the law of diminishing returns applies, with minimal performance gains despite substantial increases in hardware complexity.

Task 5:

table for fully associative cache we have used data we have collected from RARS

Fully Associative					
Cache Size (bytes)	Number of blocks	Block Size (words)	Hit Rate (%)	Miss Rate (%)	
128	4	8	92	8	
256	8	8	93	7	
512	16	8	98	2	



Task 6:

In a fully associative cache, the set size equals the number of blocks because there is effectively only one set containing all blocks. Unlike direct mapped or n-way set associative caches that divide the cache into multiple sets based on address mapping, a fully associative cache treats the entire cache as a single set. This means all blocks in the cache can be used for any memory address, maximizing flexibility but requiring comparison of all tags simultaneously, which increases hardware complexity significantly.

Task 7:

The goal of a memory hierarchy is to provide the illusion of a memory that is both large and fast by exploiting locality of reference. Memory hierarchies are necessary because of fundamental technological and economic constraints: fast memory technologies (like SRAM used in cache) are expensive and consume more power per bit than slower memory technologies (like DRAM used in main memory). By using multiple levels with increasing size but decreasing speed, the memory hierarchy achieves a better compromise between performance, cost, and power consumption than would be possible with a single memory type. Caches exploit temporal locality (recently accessed data likely to be accessed again) and spatial locality (data near recently accessed locations likely to be accessed) to maintain frequently used data in faster levels, reducing average memory access time.

Task 8:

For the optimal 4-way set associative cache configuration with 256 bytes total size, we need to calculate the storage overhead percentage. The cache contains 8 blocks with 8 words per block (32 bytes per block), organized into 2 sets (8 blocks ÷ 4 ways). When analyzing the address structure in a 32-bit RISC-V system, we have 2 bits for byte offset (4 bytes per word), 3 bits for word offset (8 words per block), and 1 bit for index (2 sets), leaving 26 bits for the tag. Each block requires storage for this 26-bit tag plus a valid bit, totaling 27 bits of overhead per block. Across all 8 blocks, this equals 216 bits or 27 bytes of overhead. Therefore, the storage overhead percentage for this cache configuration is 10.55% ($27 \text{ bytes} \div 256 \text{ bytes} \times 100\%$), representing a reasonable balance between the additional hardware required and the performance benefits gained from the 4-way set associative structure.

DISCUSSION:

Our experimental results highlight several important insights about cache configuration design. The most notable observation is the relationship between block size and miss rate across different cache sizes. For direct-mapped caches, our data reveals an optimal block size range (approximately 2-4 words) that minimizes miss rates for all cache sizes tested. Beyond this range, miss rates increase substantially, with all cache sizes converging toward high miss rates at very large block sizes. This demonstrates the fundamental tradeoff between exploiting spatial locality with larger blocks and maintaining efficient cache utilization.

The comparison between direct-mapped, n-way set associative, and fully associative configurations further emphasizes the performance benefits of higher associativity. Our 4-way set associative configuration emerged as a particularly effective compromise, offering substantial performance improvements over direct-mapped caches while avoiding the prohibitive hardware complexity of fully associative designs. This finding aligns with industry practices, where 4-way and 8-way set associative caches are commonly implemented in modern processors. The diminishing returns observed beyond 4-way associativity suggest that architects must carefully weigh marginal performance gains against increased implementation costs. Future work could explore additional dimensions such as different replacement policies, varied memory access patterns from diverse applications, or the impact of multi-level cache hierarchies. These explorations would provide a more comprehensive understanding of cache behavior under real-world conditions and potentially reveal optimization opportunities beyond those identified in our current experiments.

CONCLUSION:

As our exploration of cache configurations draws to a close, we find ourselves with a richer understanding of the invisible architecture that shapes computing performance. This laboratory experience illuminated how subtle adjustments to cache parameters can dramatically affect system behavior—a dance of numbers that translates directly to the user experience in real-world computing. Through our careful measurements and graphical representations, we witnessed the classic engineering narrative unfold: every gain comes with a cost, every optimization creates new constraints. The larger caches reduced miss rates but demanded more chip area; higher associativity improved hit rates but required more complex comparison circuitry; larger block sizes enhanced spatial locality until they began to waste valuable cache space. Such is the nature of engineering—not the pursuit of perfection, but the art of balanced compromise. Our journey through these tradeoffs provided not just technical knowledge, but a deeper appreciation for the thoughtful decisions embedded in the memory hierarchies we rely on daily. The humble cache, often overlooked in discussions of computing power, stands revealed as a masterpiece of architectural design, where science meets art in the service of performance.

