# Cache Simulation

Bishoy Pramanik

Professor Resch

19 November 2023

# Introduction

## Describe the purpose of the analysis.  What is the goal of your analysis?  What are you analyzing?

Cache implementations can have a large impact on the performance of a computer due to its ability speed up data transfer. Identifying the most effective cache implementations is crucial knowledge to the design of a computer. To determine the most effective cache implementation, my analysis will examine the differences between direct mapped, fully associative, and set associative caches. Additionally, I will analyze the effect of temporal and spatial locality on cache design for associative sets by looking at Least Recently Used (LRU) and First-In-First-Out (FIFO) design. To measure the performance of each cache design, I will measure the hit rate of a cache implementation with a growing cache size. This analysis into hit rates and cache size will shed light on the most effective cache implementations.

# Description of Tests

## What were the parameters for each test? Include the values of parameters: cache size, block size, associativity, replacement strategy. Why?

For all test cases, the **block/line size was set to 2^6, or 64, bytes**. My **test file was swim.trace.** My cache size started at 2^10, or 1024, and I incremented the exponent until the cache size reached 2^14, or 16,384, bytes. I stopped at this value due to the convergence of all cache implementations reaching more or less the same hit rate (~0.986) beyond this cache size. For cache implementations, I tested Direct Mapped, Fully Associative, 2-Way Set Associative, 4-Way Set Associative, and 8-Way Set Associative. For all associative sets, I treated LRU and FIFO as separate to test the effect of replacement strategy.

# Results

## What were the hit rates for the different configurations?

To run tests, I first compiled the cache\_sim.cpp into to separate files. One in LRU form by compiling it as provided, and another in FIFO form by commenting out “//cache[i+j][1]=counter;” in the .cpp file. After this was done, I created a Python script to run the .exe files with the parameters described before. For each cache implementation, the script ran at a block size of 64 bytes and cache sizes started at 1024 bytes and increased by a factor of 2 until cache size was 16,384 bytes. Hit rates were parsed from the STDOUT and written into a .csv file using the pandas library and graphed using the seaborn library.

## Create plots to show your results.

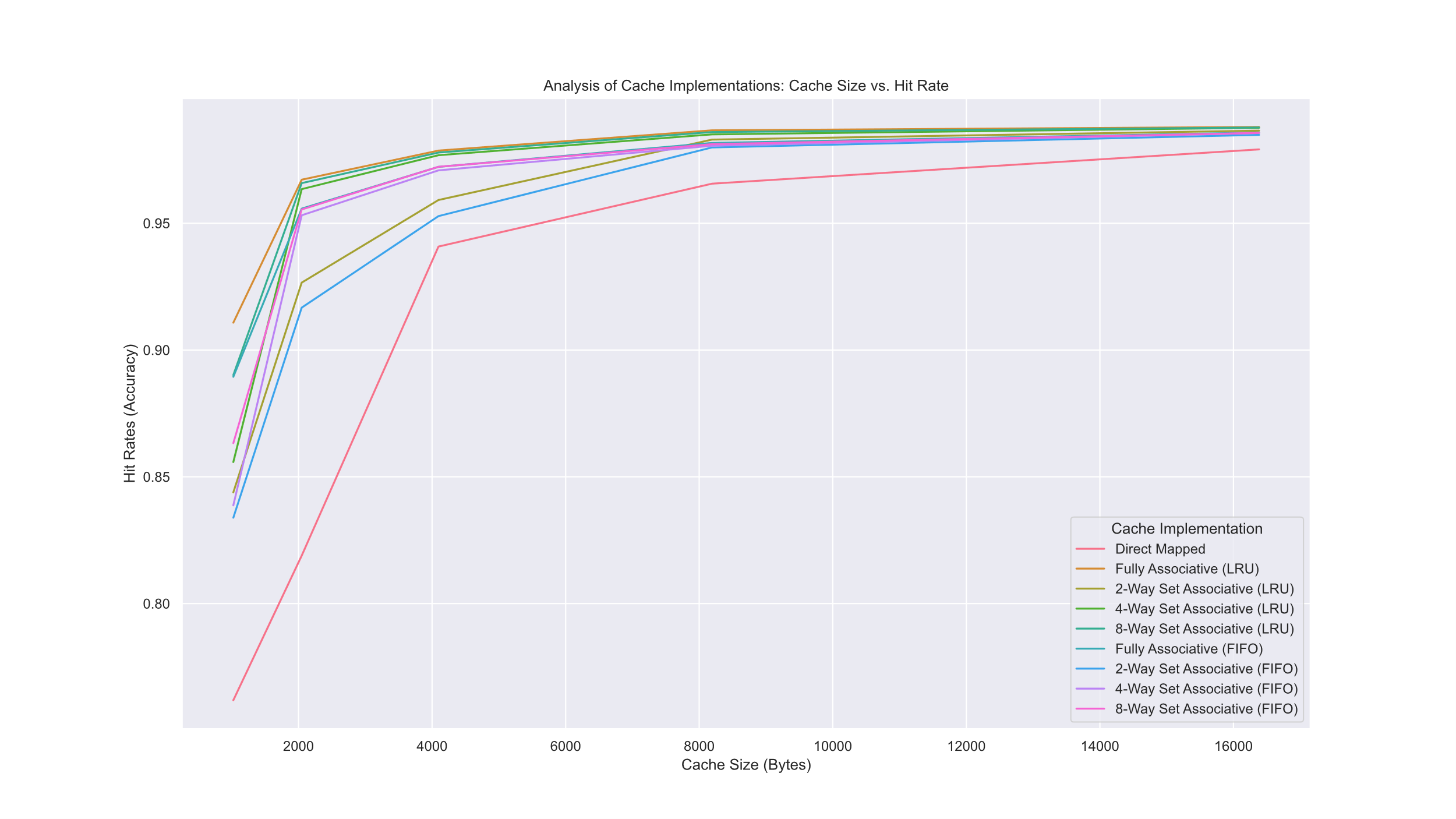


Figure : Cache Size vs. Hit Rate visualized with seaborn

# Conclusions

## How does cache design (direct mapped/set associative/fully associative) affect hit rate?

The direct mapped design was the slowest by far, while fully associative performed the best. The second fastest was the 8-way set associative design. Set Associative design performed better as the lines per set increased. This suggests that a fully associative design would be the most effective cache design, if possible.

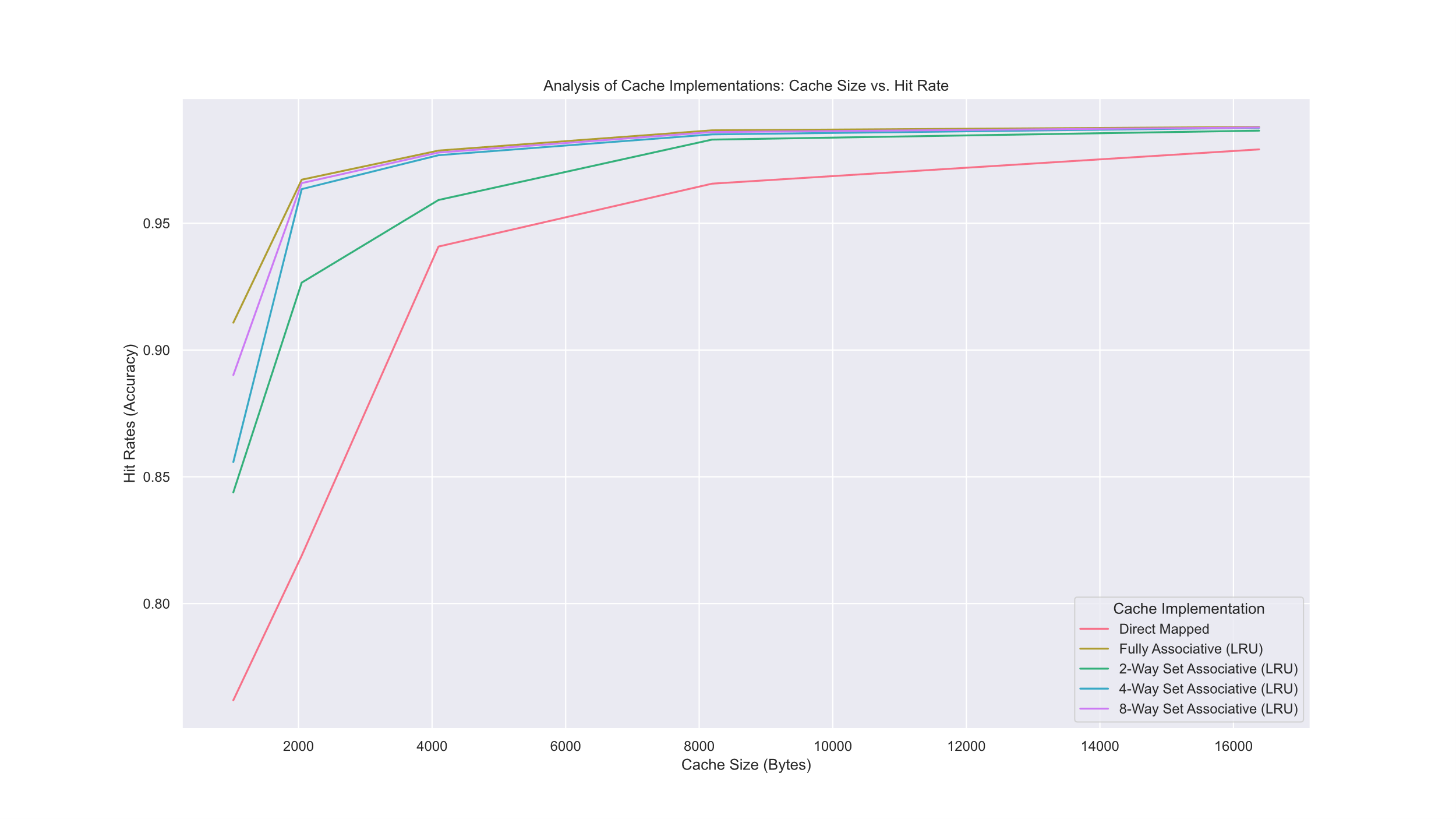


Figure : Cache Analysis: LRU only

## How does hit rate change with the two replacement policies?

LRU generally performed better than the FIFO design in associative cache implementation.

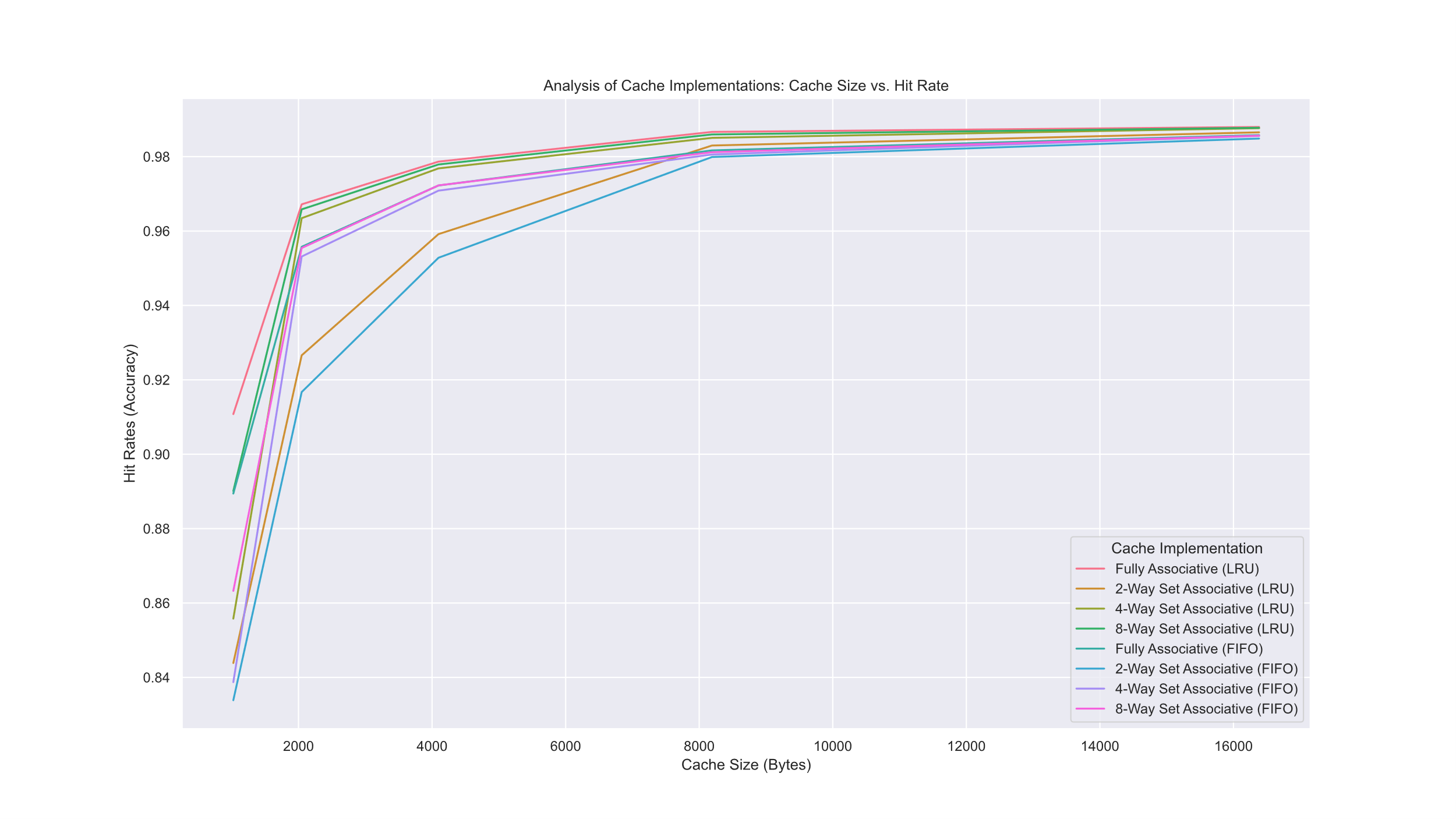


Figure : Cache Analysis: Replacement Strategies Compared for All Associative Designs

To visualize this more clearly, I plotted the results of Fully Associative Designs in LRU and FIFO designs to compare the replacement policies.

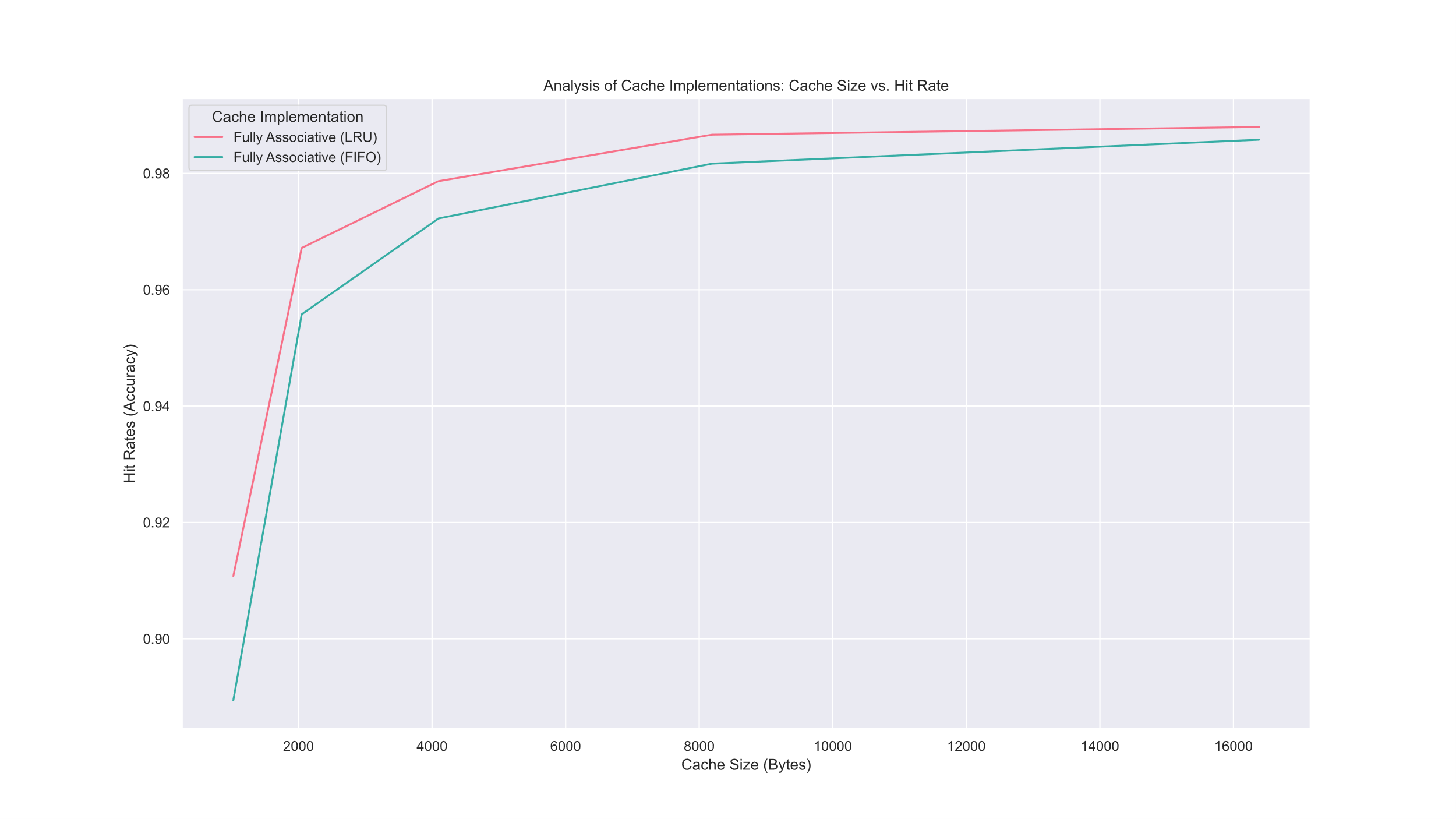


Figure : Cache Analysis: Replacement Strategies Compared for only Fully Associative Designs

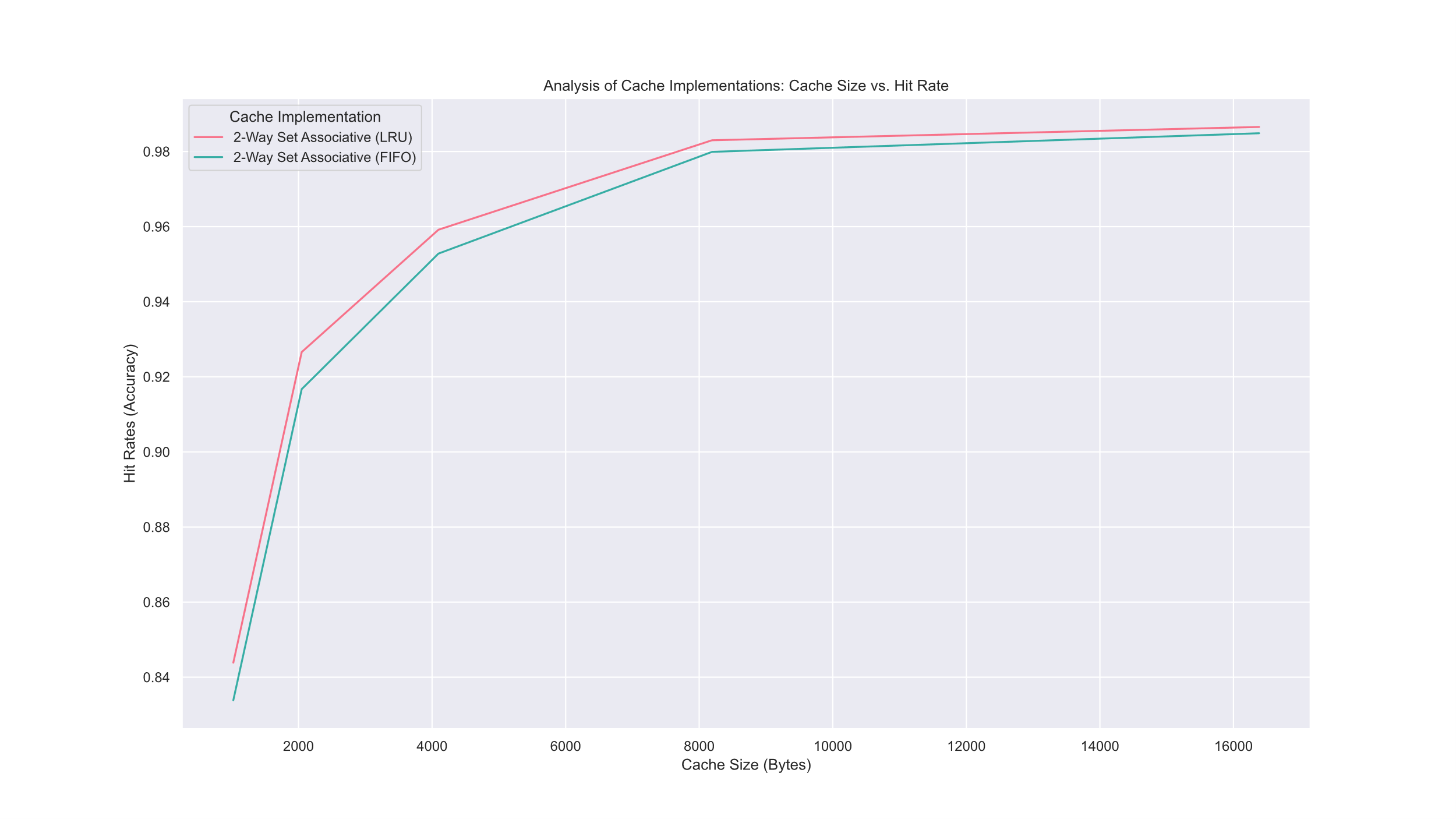


Figure : Cache Analysis: Replacement Strategy Compared for 2-Way Set Associative Designs

For fully associative designs, we can clearly see that LRU performs with a higher hit rate than FIFO. This suggests that for associative designs, taking advantage of temporal locality through LRU is more effective than spatial locality through FIFO designs.

## How does cache size affect hit rate?

At lower cache sizes, cache size led to a larger disparity in hit rates; however, as cache size increased, performance increased at a decreasing rate, levelling off for all associative designs around 98.7% accuracy, while direct mapping levels off at 98.0% accuracy.

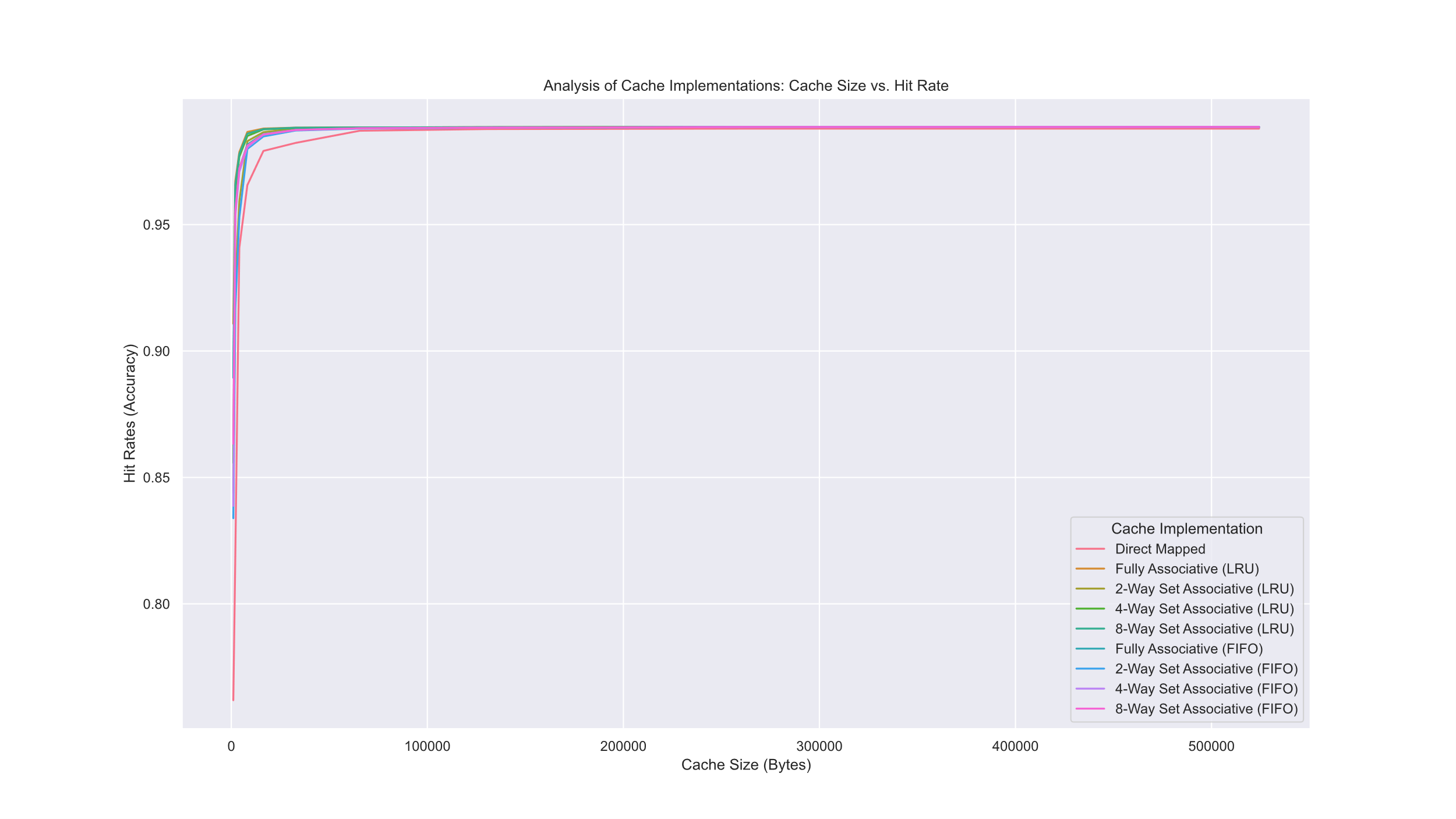


Figure : Cache Size vs. Hit Rate for All Cache Implementations.

In Figure 6, we can see that all cache implementations begin to approach the same hit rate. Note that the x-axis now extends past 2^14 bytes, all the way to 2^19 bytes. This clearly shows the asymptotic behavior of the hit rate with respect to cache size.