**Cache Simulation Paper**

**Introduction**

The purpose of this cache simulation analysis is to evaluate the performance of different cache configurations. Specifically focusing on the different cache sizes, block sizes, associativity, and replacement policy and how that may affect cache performance/effectiveness (hit rate). The cache simulator simulates three different types of cache designs: direct mapped, n-way set associative, and fully associative. The analysis uses a cache simulator to observe how different configurations influence memory access efficiency. Conclusions from this analysis can help computer engineers optimize cache architecture depending on performance and cost. By varying parameters such as cache size, block size, and associativity, and by comparing different replacement policies like FIFO and LRU, the configurations that yield optimal performance will be clear.

**Description of Tests**

Cache size vs Hit rate

To determine the effect of cache size on hit rate, 7 different cache sizes were picked. Each cache size was chosen to ensure it was a power of two, this is because when calculating the number of bytes in the tag or offset field, the log base two will be taken and the byte count must be a whole number. Therefore, it would make sense to have cache sizes that are powers of two. The specific caches size (in bytes) chosen for this analysis are 256 (28), 512 (29), 1024 (210), 2048 (211), 4096 (212), 8192 (213), 16384 (214). Similar to the cache size, the block size was also chosen to be a power of two for the same reason that when taking log base 2, the end result should be a whole number. The block size was chosen to a fixed number of 32 bytes (25) since there were other variables that were going to be changed, the block size was meant to act as a fixed variable across the trials.

There were two replacement strategies that were analyzed, FIFO (first in first out) and LRU (least recently used). FIFO evicts the oldest block in the set whenever a new block needs to be loaded into the cache. LRU evicts the block that has not been accessed for the longest time. Choosing two different replacement strategies will highlight which replacement strategy optimizes cache performance.

There were five different levels of associativity: direct mapped (1-way; each set contains a single cache line), fully associative (all lines belong to a single set), 2-way associative (each set contains 2 lines), 4-way associative (each set contains 4 lines), 8-way associative (each set contains 8 lines).

In Figure 1, a screenshot of some of the test cases used to create the graph of cache size vs hit rate is displayed. The first parameter is the cache size, the second parameter is the block/line size, the third parameter is the associativity, and the last parameter is the replacement policy.

A screenshot of a computer

Description automatically generated

Figure - Screenshot of sample test cases for different cache sizes

Block size vs Hit rate

Similar to the cache size vs hit rate section the same associativity and replacement policies were used. However, in this section the block sizes vary while the cache size remains the same. The constant cache size was chosen to 4096 bytes since the requirement of the size being a power of two remained and 4096 was one of the cache sizes chosen in the previous section. The chosen block sizes (in bytes) are: 8 (23), 16 (24), 32 (25), 64 (26), 128 (27), 256 (28). The block sizes are powers of two for the same reasons mentioned earlier.

In Figure 2, a screenshot of some of the test cases used to create the graph of block size vs hit rate is displayed. The first parameter is the cache size, the second parameter is the block/line size, the third parameter is the associativity, and the last parameter is the replacement policy.

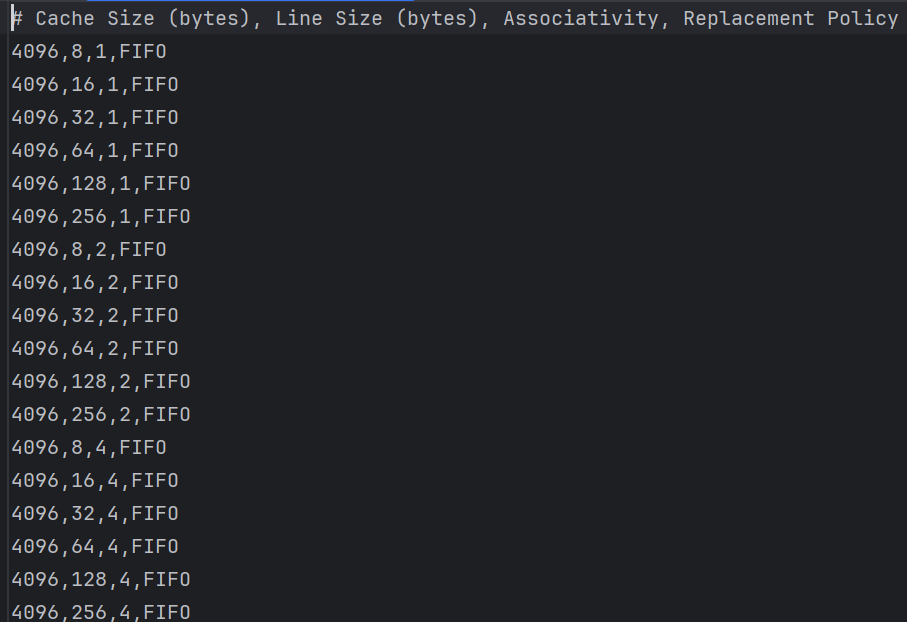


Figure - Screenshot of sample test cases for different block/line sizes

**Results**

A graph of different colored lines

Description automatically generated

Figure - A plot of cache size's effect on hit rate

In Figure 3, the hit rate generally increases across all configurations as the cache size increases. This is expected since as the cache increases in size, the amount of data it can store can also increase, which reduces the likelihood of conflict misses. The direct mapped cache consistently shows a lower hit rate compared to the other caches because direct mapped caches are more susceptible to conflict misses since each memory block can only map to a single in the cache. The 2-way, 4-way, and 8-way associative caches resulted in a better performance than the direct mapped cache. The fully associative cache appears to perform better or the same as the n-way associative caches since they allow any block to be placed in any line, which minimizes conflict misses. LRU configurations appear to have a slightly better hit rate than the FIFO configurations. This is most likely because the LRU configurations retain the most frequently accessed data blocks, which leads to a higher likelihood of cache hits. Towards 16,384 bytes, all the configurations tend to converge, which suggests that at such a large cache size, most of the configurations are able to store most of the data needed.

A graph of different colored lines

Description automatically generated

Figure - A plot of block size's effect on hit rate

From Figure 4, it is noticeable that increasing the block size from 8 to 32 bytes results in an improvement in the hit rate across all the configurations. This makes sense larger blocks fetch more adjacent data, which increases the likelihood of cache hits. However, as the block size continues to increase beyond 32 bytes, then the cache hits decrease for most configurations. This could be possibly attributed to the fact that unnecessary data may be brought into the cache, which in turn reduces the effective use of cache space. The direct mapped cache shows the lowest hit rate across all the block sizes. This makes sense since direct mapped caches are more susceptible to conflict misses because there is a strict mapping of memory blocks to cache lines. The fully associative cache (both LRU and FIFO) displays the highest hit rate for most block sizes, since a block can be placed in any cache line, resulting in a lower number of conflicts. LRU configurations appear to have a slightly better hit rate than the FIFO configurations for this plot as well. This is most likely because the LRU configurations retain the most frequently accessed data blocks, which leads to a higher likelihood of cache hits. The optimal block size for maximizing hit rate appears to be 32 bytes since any larger than 32 bytes, the hit rate starts to decrease for most cache configurations.

To create the graphs seen in Figure 3 and Figure 4, the results from the main.cpp file were entered into a .csv file. Then I wrote a simple Python program, that took the data in the .csv file and created two plots. A sample of the Python code used to generate the plots can be seen in Figure 5.

A screen shot of a computer screen

Description automatically generated

Figure - Sample of Python code used to generate the plots in Figure 3 and Figure 4

**Conclusion**

In conclusion, the aim of this analysis was to observe the performance (hit rate) of three cache designs (direct mapped, fully associative, and n-way associative) when the following parameters were changed or kept the same: cache size, associativity, replacement policy, and block size.

From Figure 3 and Figure 4, it was seen that fully associative caches consistently achieved a high hit rate across different cache and block sizes compared to other cache configurations since fully associative caches have an increased flexibility due to placing blocks anywhere in the cache, thus reducing conflict misses. Also, the 2-way, 4-way, 8-way associative caches provided a higher hit rate than direct mapped caches since they allowed more lines per set, which in turn reduced conflicts. Direct mapped caches generally exhibited the lowest hit rate since each block could only mapped to one specific cache line, resulting in more conflict misses.

The LRU policy generally performed slightly better than the FIFO policy, which makes sense since the LRU policy would retain the blocks that are most likely to be accessed again while FIFO got rid of the oldest block regardless of how often it was accessed, resulting in conflict misses. However, the difference in hit rates between replacement policies seemed to decrease as cache size increased, suggesting that the replacement policy is not as important when the cache has enough space to store frequently accessed blocks.

Increasing cache size significantly improved the hit rate across all configurations, which suggests that the larger the cache, the more information that can be stored, and the more likely the information needing to be accessed will be present (increased hit rate). However, an increased block size only appears to positively impact hit rate until a block size of 32 bytes as seen in Figure 4. Block size increasing from 8 to 32 bytes improves hit rate, which could be attributed to the fact larger blocks exploit spatial locality by fetching more consecutive data. However, past a block size of 32 bytes, there may be blocks that are filled with data that may not be reused leading to an ineffective cache utilization, resulting in lower hit rates.