COL 216 Assignment-3

Harshdeep 2020CS10346 & Pravin Kumar 2020CS10369

May 2023

Approach

Read Instruction

We have implemented a 2D array as an L1 tag array, L1 dirty bit, and L1 last use (for any block used for implementing LRU policy). Similarly, we have implemented 3 2-D arrays for a tag, dirty and last use for L2.

L1 hit

So given any read instruction, we first check it into L1 by finding its index and matching its tag to all elements of that index set, if we find then we just change the last use of that block.

L1 miss

Otherwise, this will count as L1 read miss. Now, we know that we need to bring that block in L1, so if all the entries corresponding to the index of that block are having a block, we chose the block which was least recently used and evict that position. Also if evicted block is dirty we also need to write back this block in L2 as this is a write-back cache. Now we similarly search for the block in L2 as we did in L1. If we find we change its last use value, otherwise we will count it as L2 read miss and bring the block from memory.

Write instruction

L1 hit

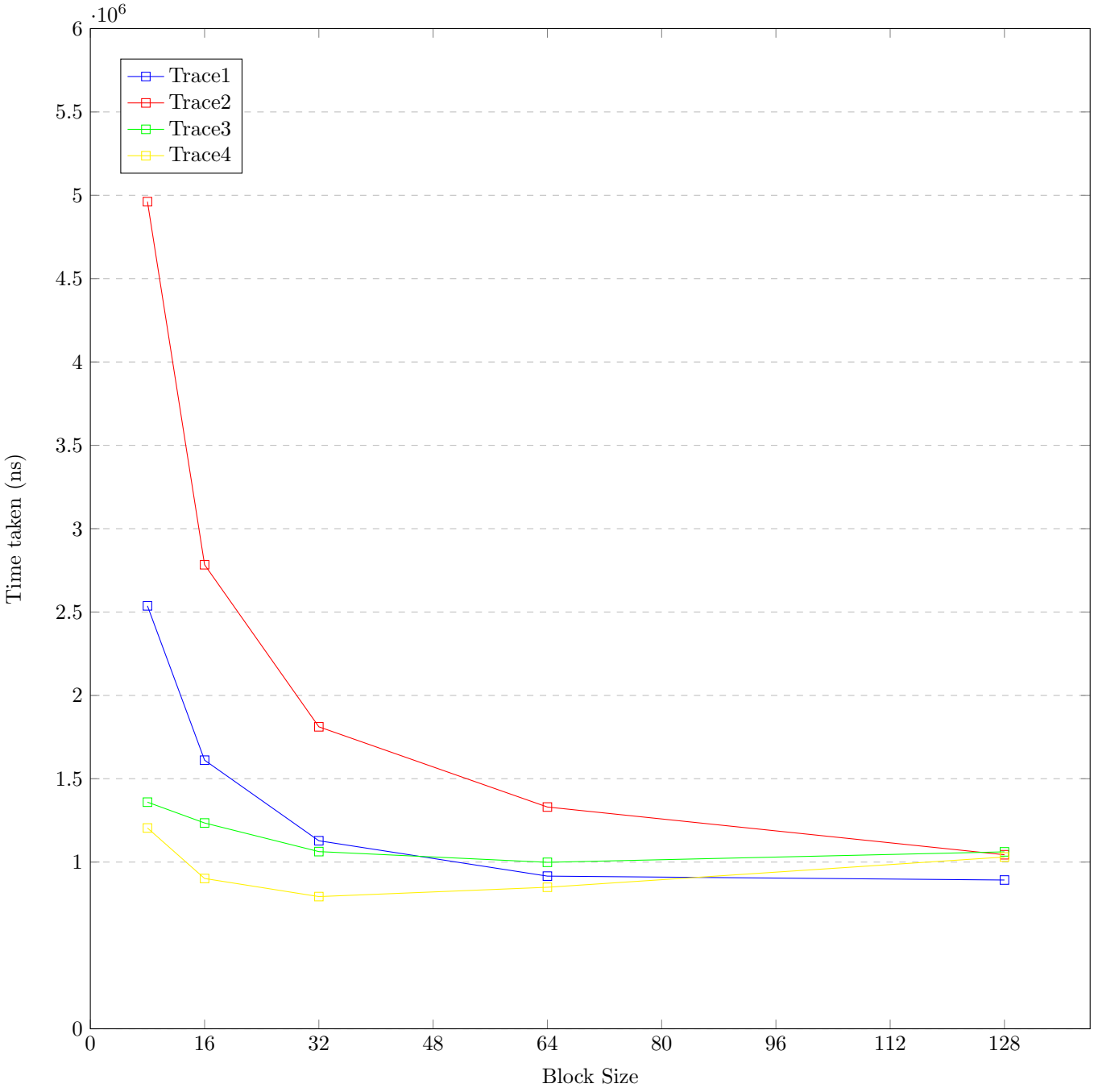
So given any write instruction, we first check it into L1 by finding its index and matching its tag to all elements of that index set. If we find then we change the last use of that block and also mark that block as a dirty block (since it is a write-back cache).

L1 miss

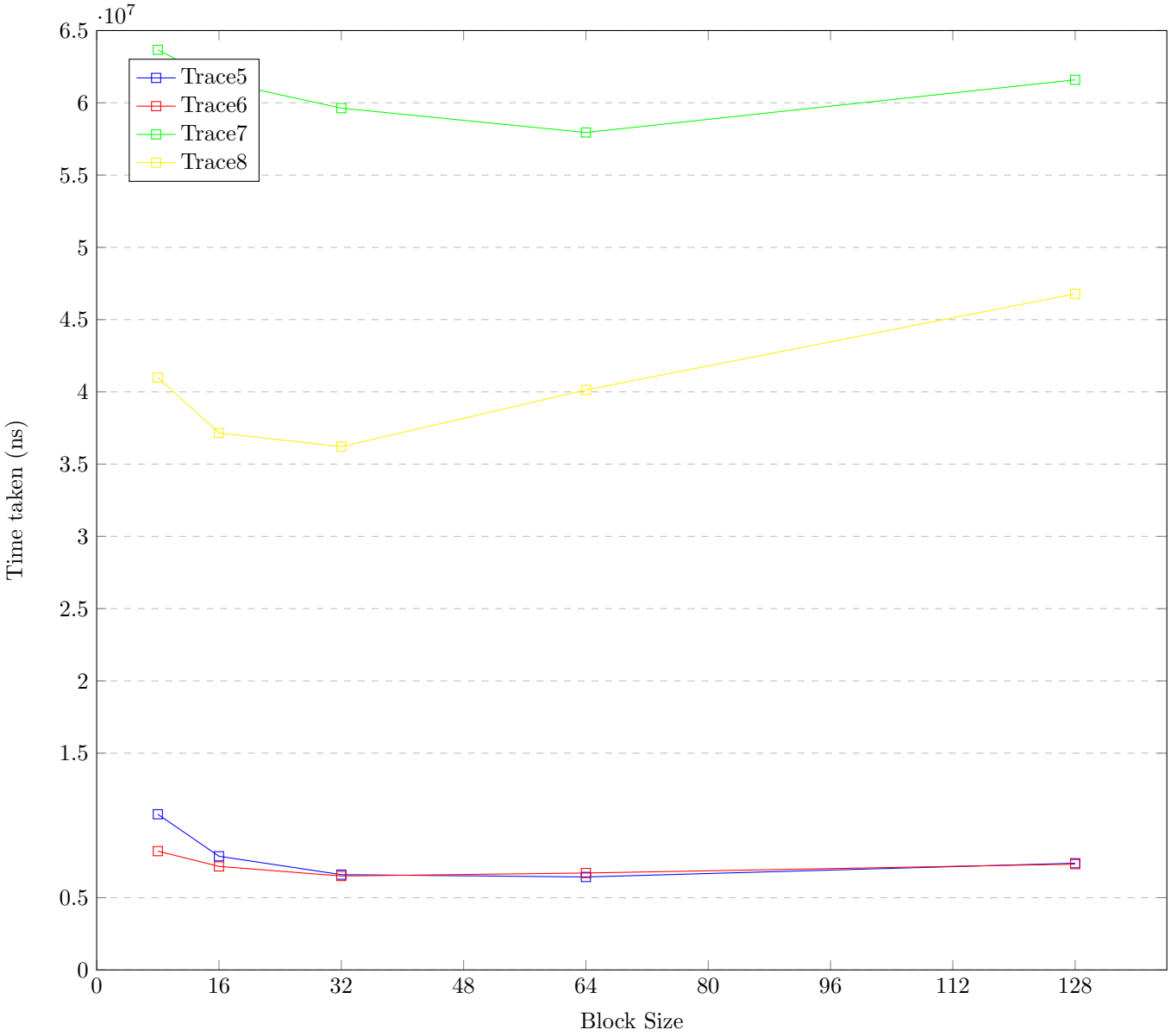
Otherwise, this will count as an L1 write-miss. Now it will be similar to the read case. We know that we need to bring that block in L1, so if all the entries corresponding to the index of that block are having a block, we chose the block which was least recently used and evict that position. Also if evicted block is dirty we also need to write back this block in L2 as this is a write-back cache. Now we similarly search for the block in L2 as we did in L1. If we find we change its last use value, otherwise we will count it as L2 read miss and bring the block from memory. Similarly, if there is L2 read miss and all the entries corresponding to the index of that block are non-empty, then we need to evict that block from L2. And if this block is dirty we need to write back this block in memory.

We have also maintained the inclusivity by evicting the block from L1 (if present) whenever we evict from L1.

Varying Block Size(Trace 1 to Trace 4)

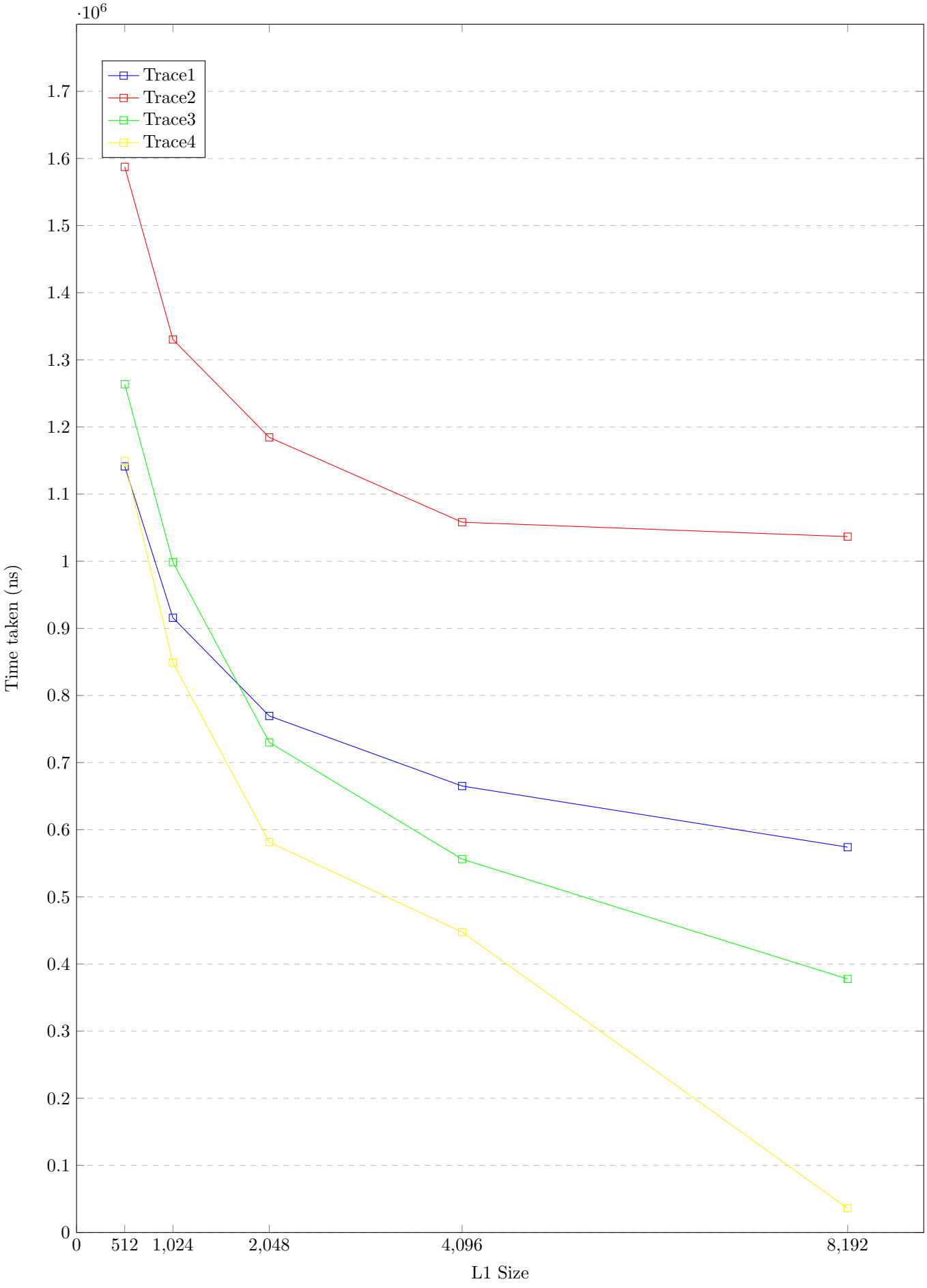


Varying Block Size(Trace 5 to Trace 8)

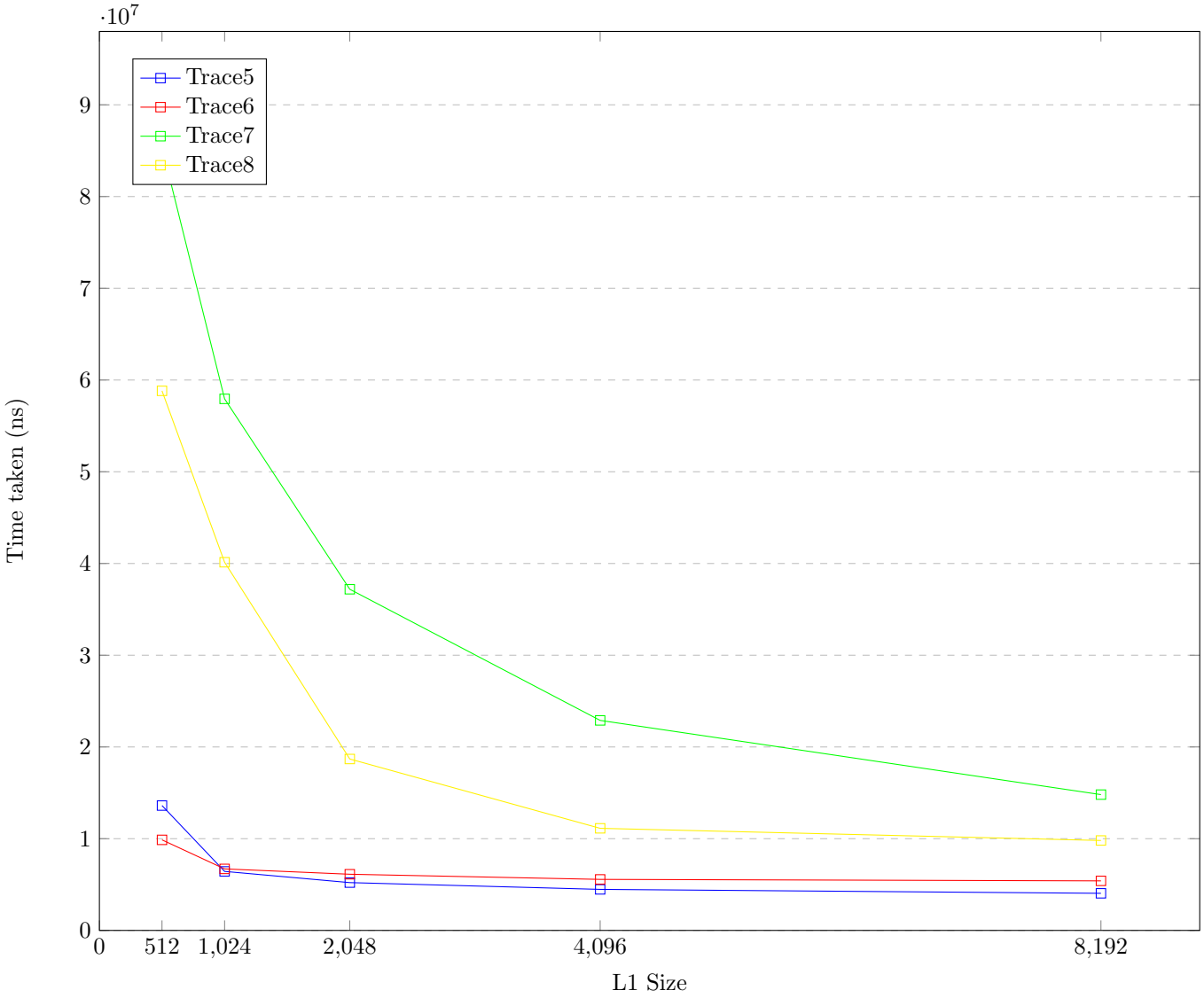


When we increase the block size, initially the total time taken increases, but after 64 blocksize it is almost constant, or the time increases in some cases. This may be because increasing the block size intially allows us to fit more data into each cache block. This can improve spatial locality by allowing the cache to store more data that is likely to be accessed together. As a result, the cache hit rate may increase, which can lead to better overall performance.But after some time it either remains constant or performance is decreasing.This is because now each cache block can store more data, we need fewer blocks to store the same amount of data. As a result, the total number of blocks in the cache decreases, which reduces the cache capacity. This means that there is a higher likelihood of cache conflicts, where multiple memory locations map to the same cache block.

Varying L1 Size(Trace 1 to Trace 4)



Varying L1 Size(Trace 5 to Trace 8)



When we vary L1 size, then we can see that the time is decreasing and performance is increasing,due to the following reasons.

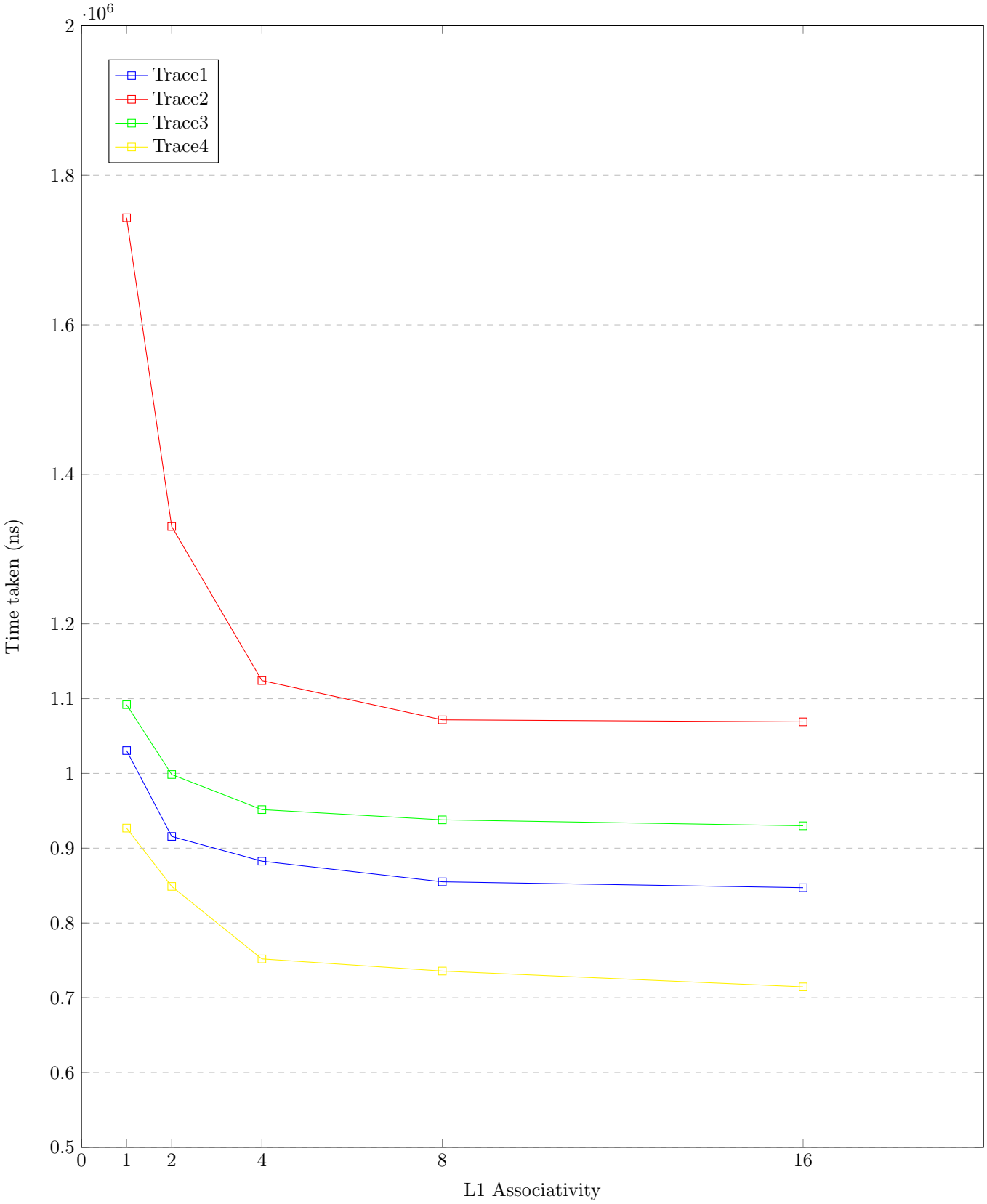
i)Increased hit rate: With a larger L1 cache, there is more room to store frequently accessed data, which can lead to a higher cache hit rate. This means that the processor can access data more quickly, improving performance.

ii)Reduced capacity misses: With a larger L1 cache, there is less likelihood of capacity misses, where data is evicted from the cache to make room for new data. This is because more data can be stored in the cache, reducing the frequency with which data must be evicted.

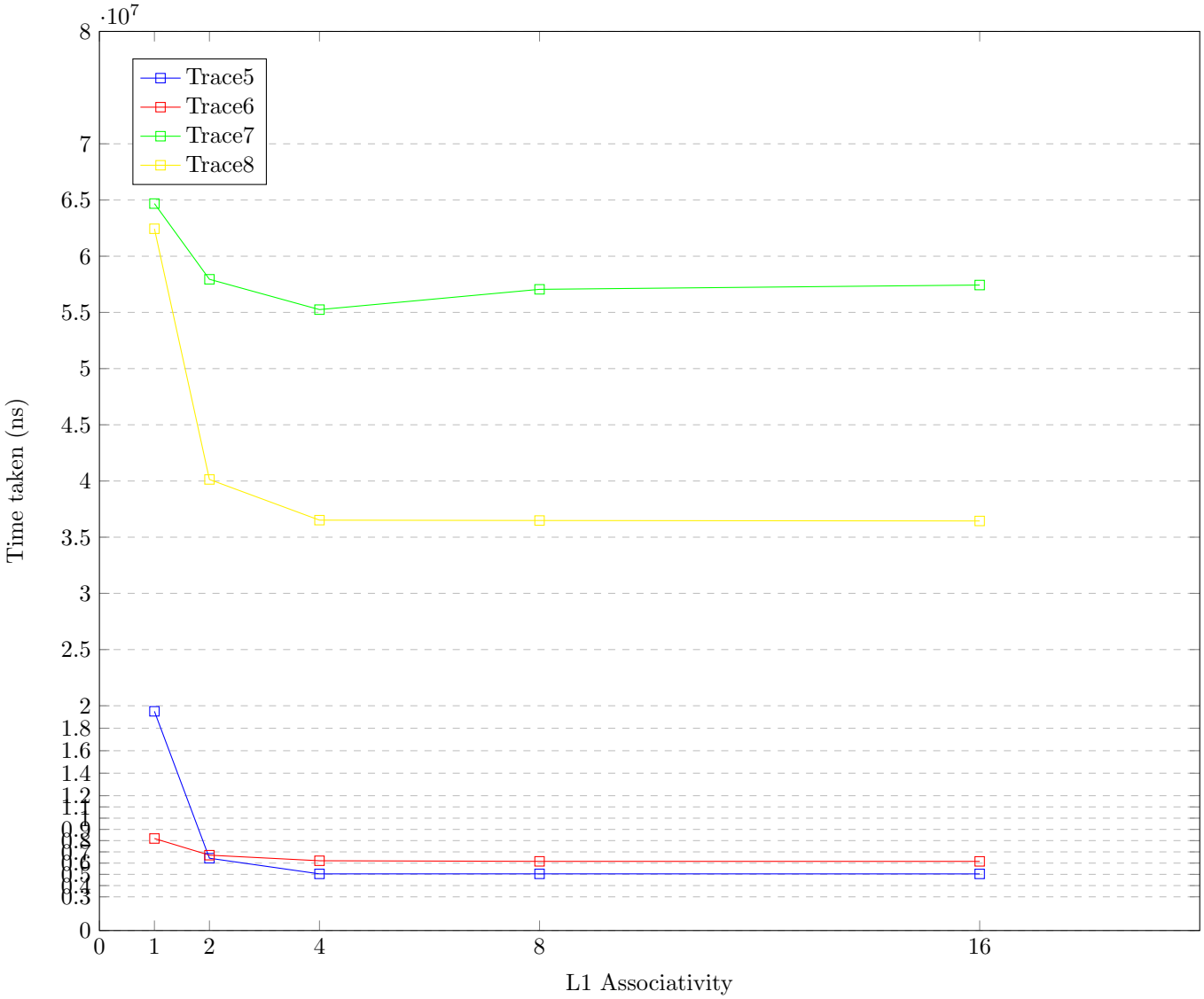
iii)Reduced conflict misses: With a larger L1 cache, there is also less likelihood of conflict misses, where multiple memory locations map to the same cache block. This is because there is more room to store data, reducing the chance of cache blocks being reused frequently.

Overall, increasing the L1 cache size can have a positive impact on cache performance by reducing cache misses and increasing hit rate. However, it is important to consider the trade-offs, such as increased energy consumption and potential latency increases, when deciding on the optimal cache size for a particular application or system.

Varying L1 Associativity(Trace 1 to Trace 4)

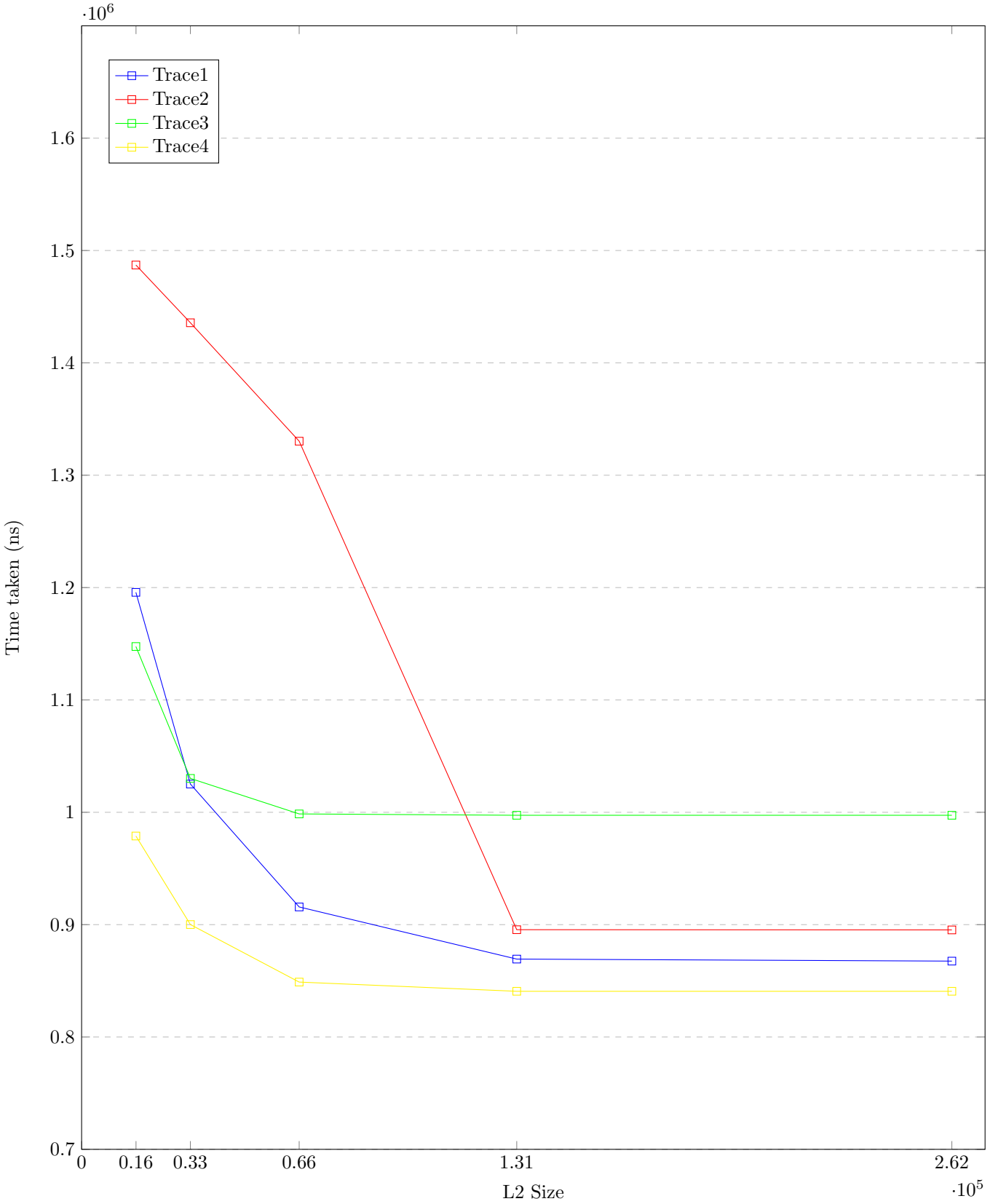


Varying L1 Associativity(Trace 5 to Trace 8)

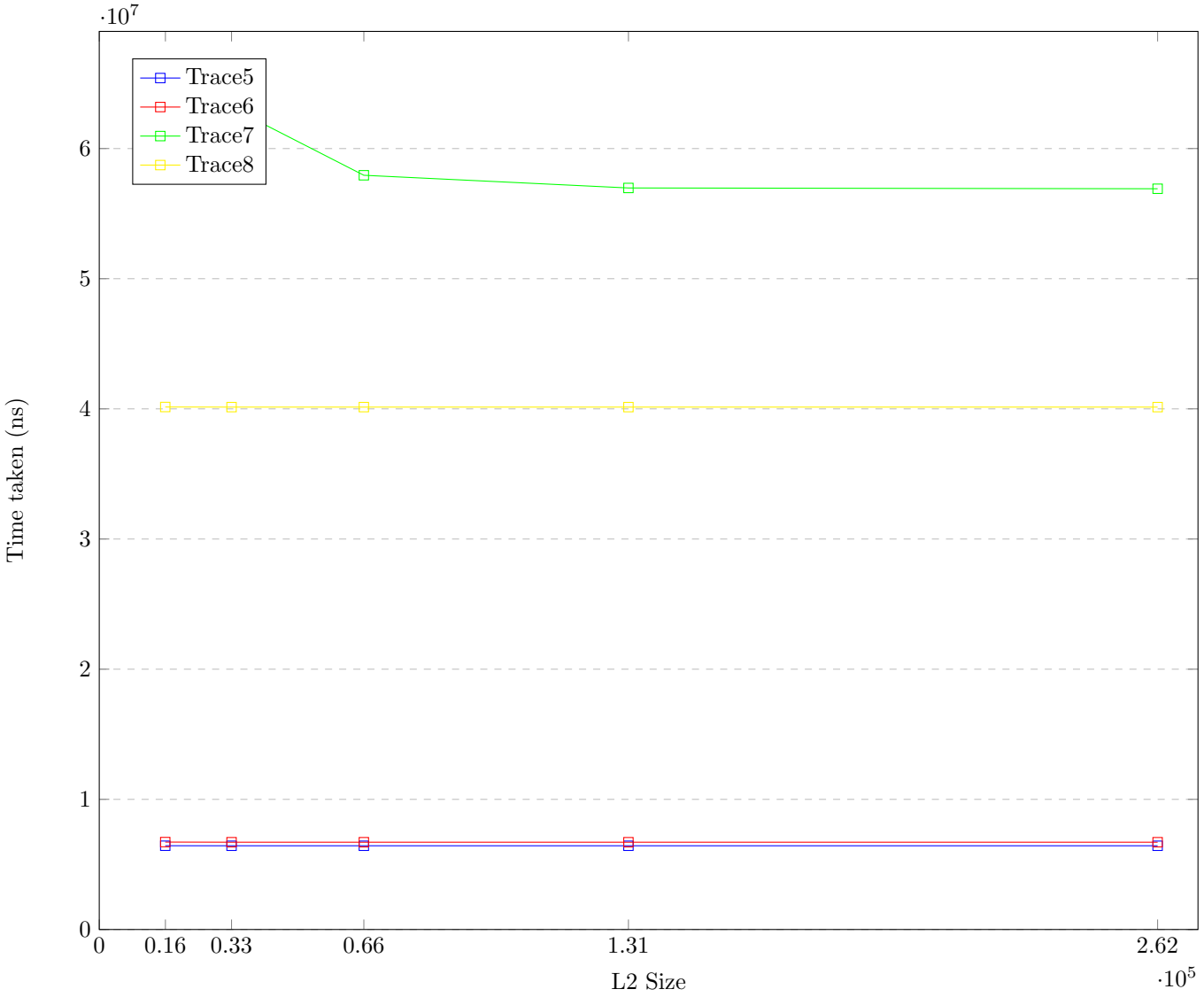


When we increase L1 associativity, then it decreases the total access time or it remains same.This is because L1 cache associativity determines the number of cache lines that can map to a particular set within the cache. By increasing the associativity, the number of sets in the cache increases, which can reduce the likelihood of conflict misses where multiple memory locations map to the same cache set. Also, With increased associativity, there is a higher likelihood of finding the desired data within the cache, which can improve the hit rate and overall performance.
We don't see many improvements after increasing associativity till 8 because as the associativity of the L1 cache is increased, the benefits of additional sets begin to diminish. This is because the likelihood of finding data within the cache improves less as the associativity becomes larger, resulting in smaller improvements in the hit rate for each additional set. Also in real caches, as L1 cache associativity is increased, the time required to search through the cache structure for a particular piece of data increases. This can result in longer access times and slower overall performance.Therefore there is also a tradeoff.

Varying L2 Size (Trace 1 to Trace 4)



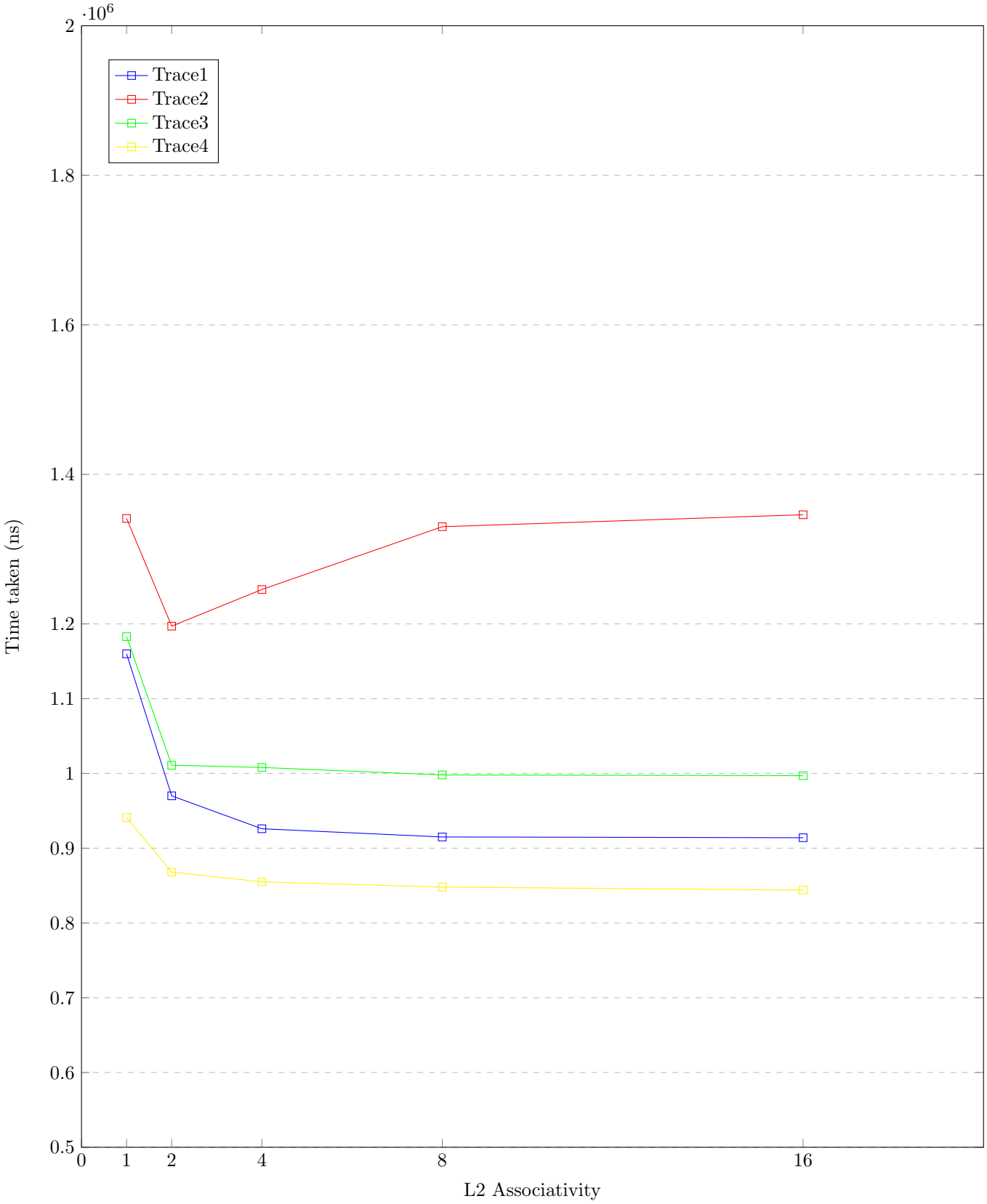
Varying L2 Size (Trace 5 to Trace 8)



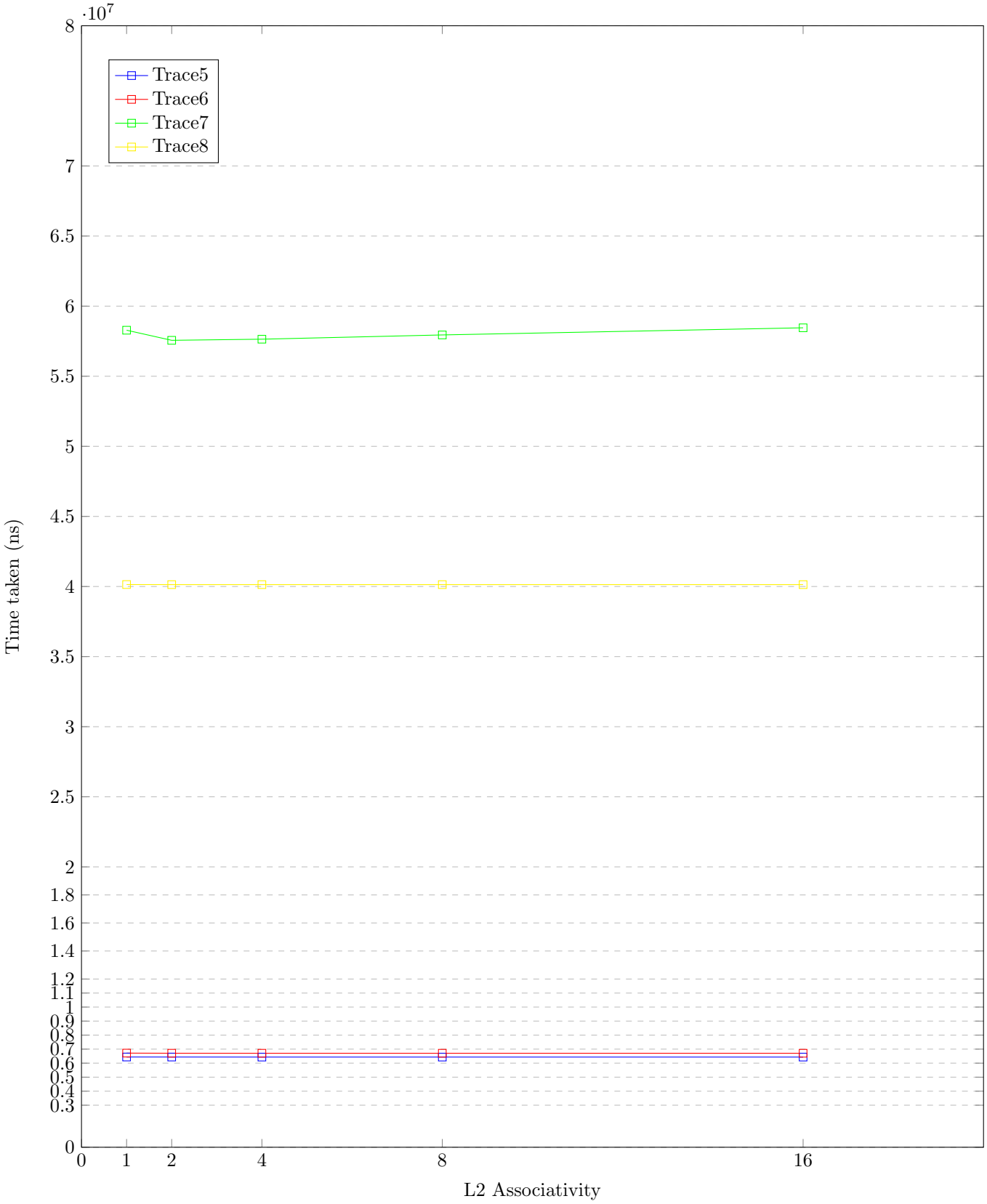
Increasing L2 size, while keeping other parameters at a constant value, do increases performance but the increase is not as significant as it was in the case of increasing the L1 cache. Generally increasing L1 cache size can have a more significant impact on cache performance than increasing L2 cache size, due to the closer proximity of the L1 cache to the processor. This can result in lower access latencies and higher hit rates compared to the L2 cache. Additionally, the L1 cache is typically smaller than the L2 cache, which means that it may be more likely to experience capacity misses. Increasing the size of the L1 cache can help to reduce the frequency of these misses and improve cache performance.

However, there may be situations where increasing L2 cache size is more beneficial. For example, if the application has a high degree of spatial locality, where nearby memory locations are accessed frequently, increasing the L2 cache size may provide a larger improvement in cache hit rate compared to increasing the L1 cache size.

Varying L2 Associativity(Trace 1 to Trace 4)



Varying L2 Associativity(Trace 5 to Trace 8)



Similar to the case of increasing L1 associativity, we can see that increasing L2 associativity increases performance initially, but it may also decrease the performance sometimes asthe likelihood of finding data within the cache improves less as the associativity becomes larger, resulting in smaller improvements in the hit rate for each additional set.

Distrubtion

We have worked almost equally on this assignment.

Conclusion

We can see that for all the parameters, initially when we increase it then the performance is increasing for all the trace files, but after some point, it starts to decrease for some or all test cases.Also, there are various tradeoffs in increasing any parameters. For example, when we increase the cache associativity, then it may increase hit rates but the time it takes to go through one set increases. Similarly when we increase blocksize, then it may increase the hit rate, but the amount it takes for bringing a block from memory to cache also depends on the block size. So we need to take care of that also. Overall we can say that increasing blocksize has impact than increasing L1 cache and L1 cache improving has more impact than improving L2 cache on overall performance.

Thank You