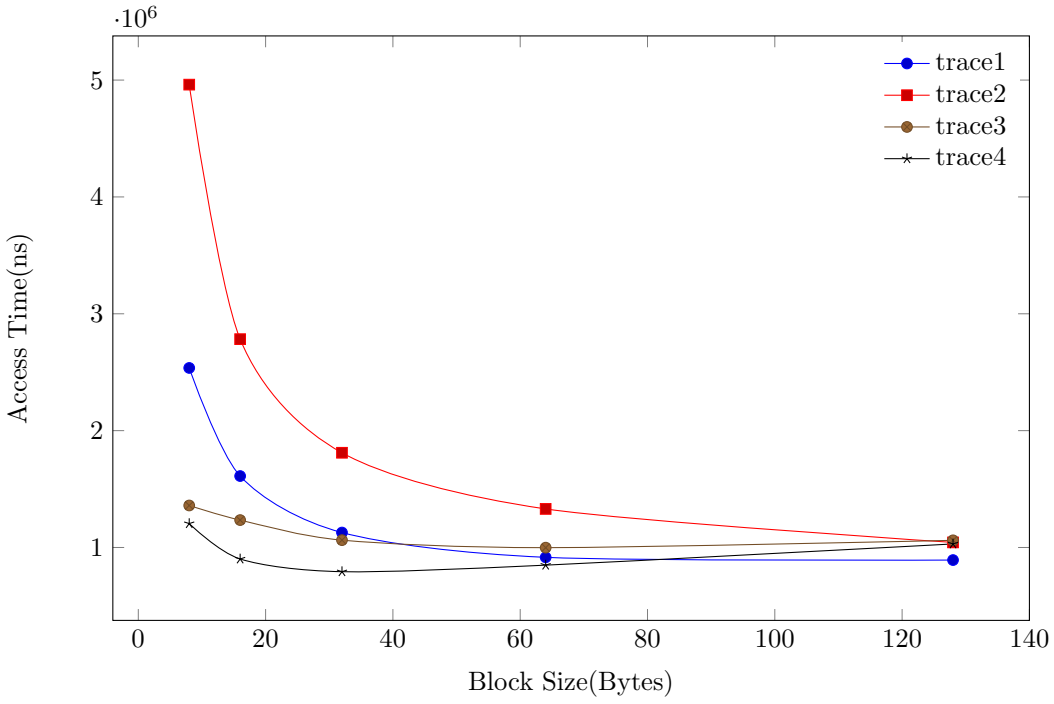
COL 216 REPORT

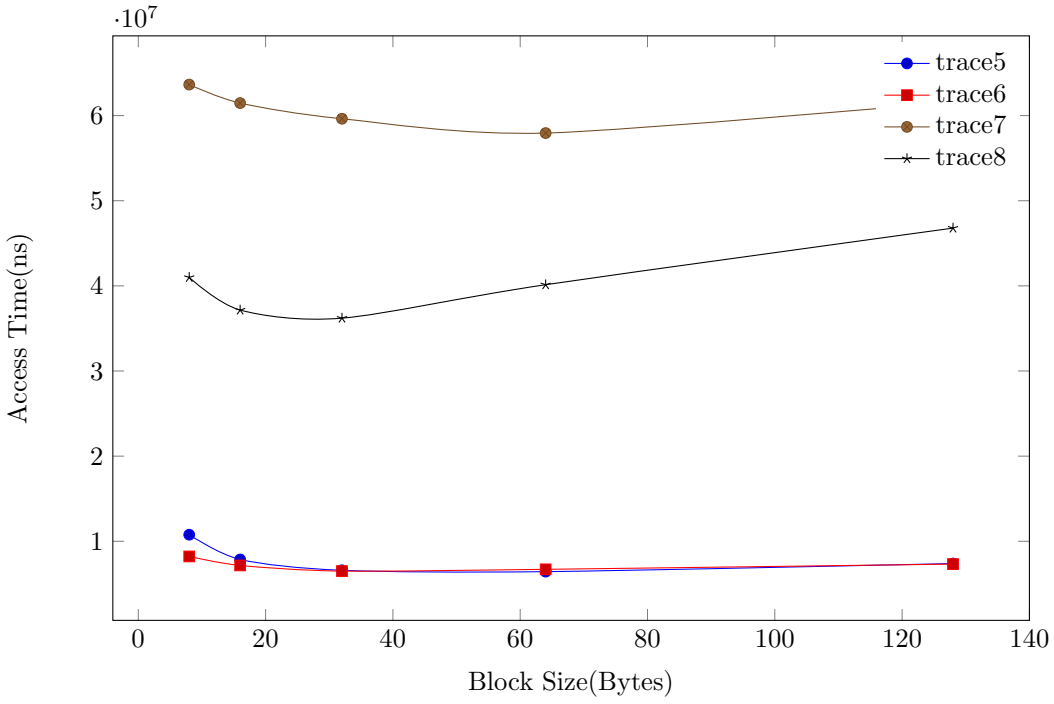
Assignment 3

2021CS10574 Gaurav Singh
2021CS10099 Sharad Kumar

May 11, 2023



(a) Trace 1,Trace 2,Trace 3,Trace 4



(b) Trace 5,Trace 6,Trace 7,Trace 8

Figure 1: BLOCKSIZE Variation

Observations

1 On varying the L1 - Blocksize

1.1 Trace 1-4

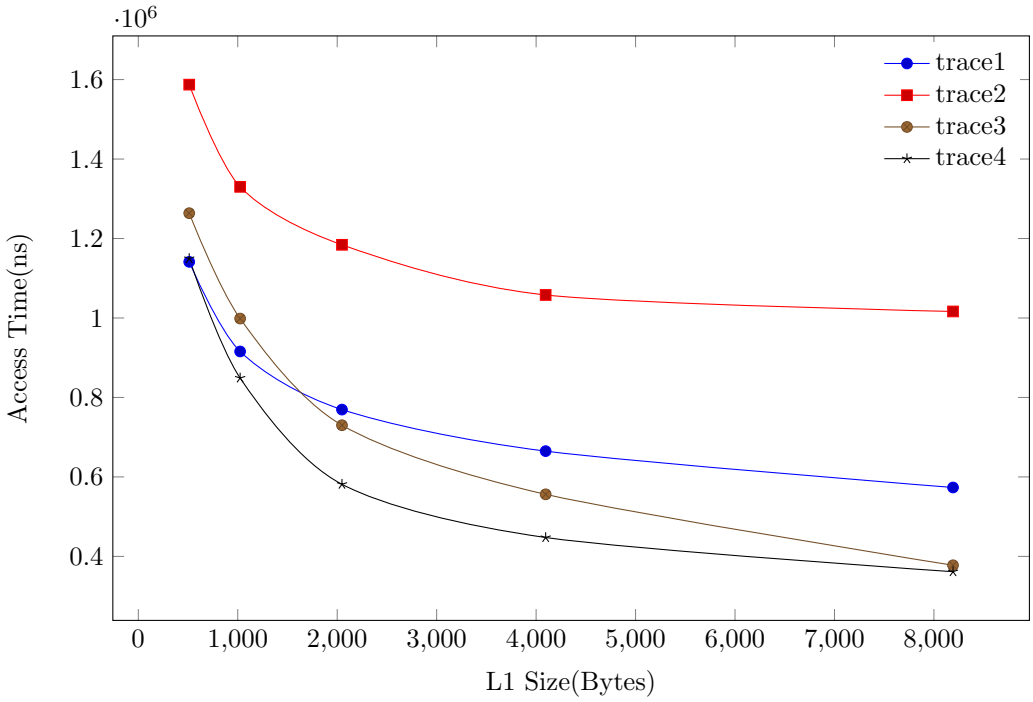
These traces have smaller size i.e. 1lakhs cache instructions.On increasing the blocksize , at first the the total access time decreases steeply as we increase the block size but on further increasing the cache size the time either remains constant (for trace 1,2,3) or started increasing gradually.

1.2 Trace 5-8

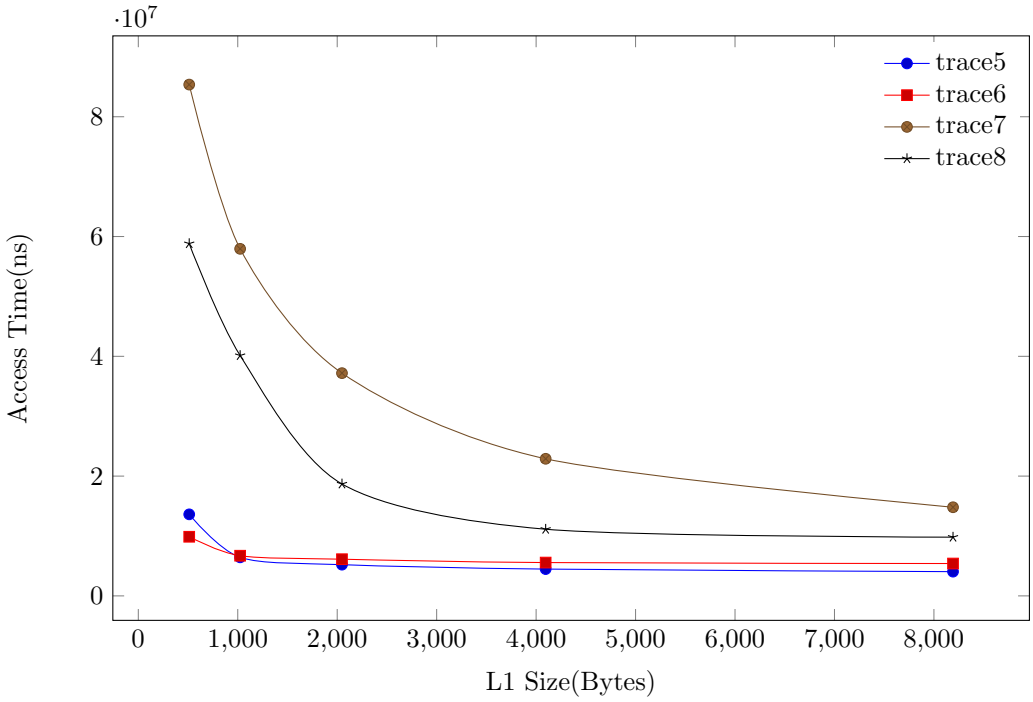
These traces are larger in size i.e (4-12)million instructions. On increasing the blocksize initially , time decreases but started increasing gradually on further increasing the size.

1.3 Explanation

By varying the block size and keeping all other cache parameters fixed, we can observe how the cache hit rate changes as you increase or decrease the block size. Typically, you would expect to see a curve that starts high for small block sizes (due to poor spatial locality) and gradually increases as the block size increases, before plateauing and then rising again (this happens due to reduced no. of sets and thus the range of data stored in cache that have been hit in the past has been reduced). The optimal block size would be the one that corresponds to the trough of the curve.



(a) Trace 1,Trace 2,Trace 3,Trace 4



(b) Trace 5,Trace 6,Trace 7,Trace 8

Figure 2: L1_SIZE Variation

2 On varying the L1-size

2.1 Trace 1-4

These traces have smaller size i.e. 1lakhs cache instructions.On increasing the cache size the total time decreases for all traces but the decrease is not uniform.We observe significant decrease, reason being the increase in no. of sets reduces the cache conflicts and this reduction reaches a saturation on further increase in L1 size and thus there is no more decrease in total access time.

2.2 Trace 5-8

Even when we increase the size of the trace files ,the trend remains similar .Increasing the size of L1 cache resulted in the decrease in total access time for trace 7 and 8 but for trace 5 and trace 6 the decrease in negligible which implies that the conflict for the same index is very less.

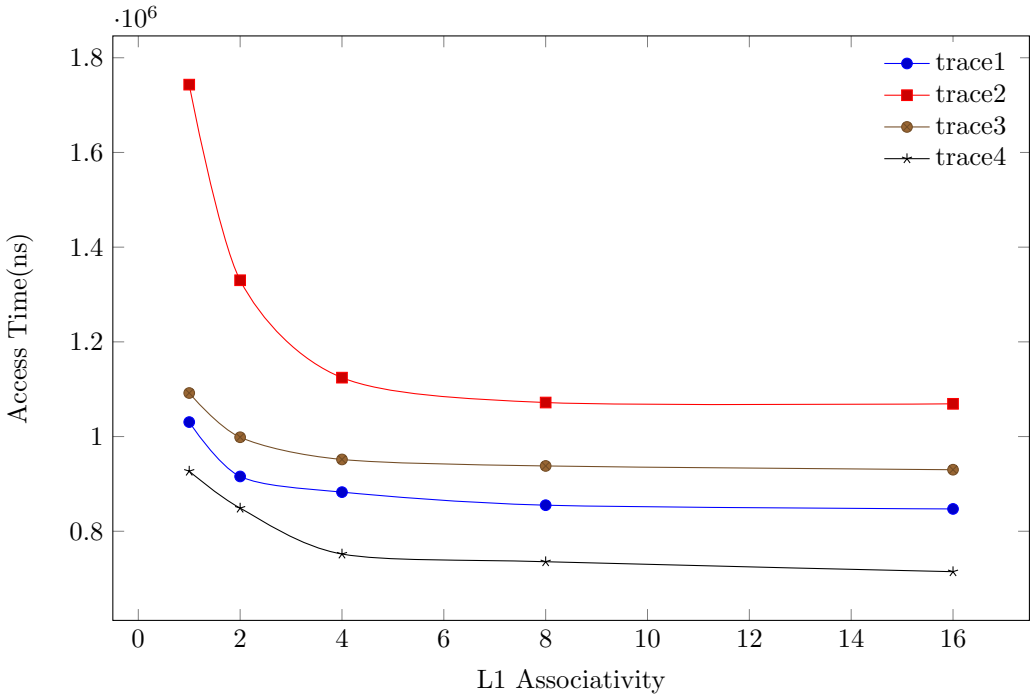
2.3 Explanation

Ideally, Specifically, increasing the L1 size from 512 to 1024, and then to 2048, is likely to result in improved performance. This is because the larger cache can store more data and reduce the number of times the CPU needs to access main memory, which can result in faster execution times.

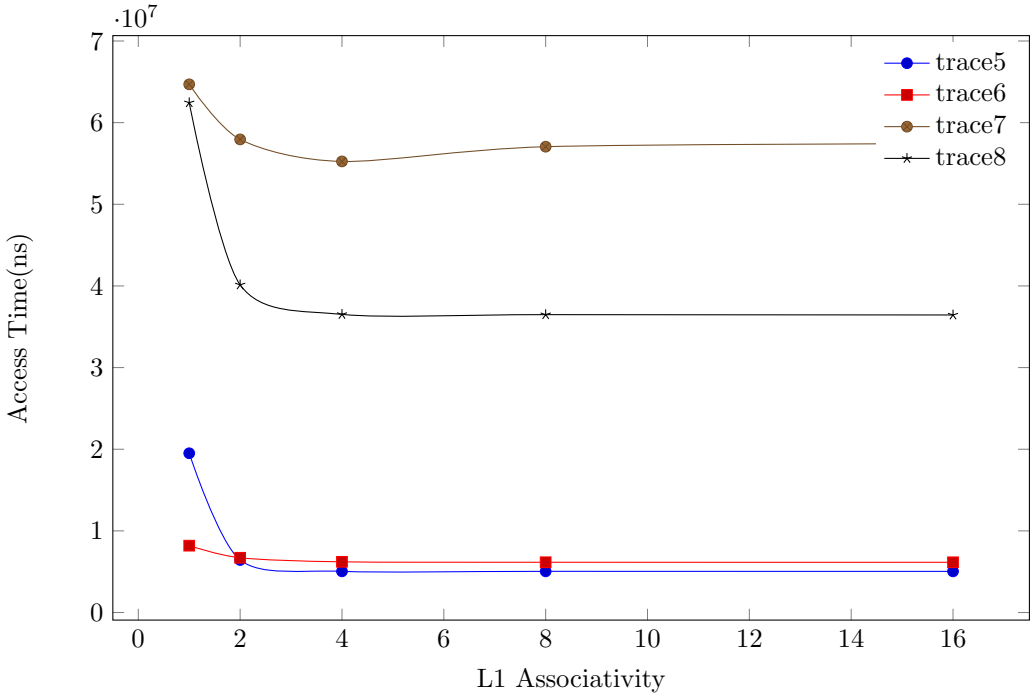
However, increasing the L1 size further beyond 2048 may result in diminishing returns, or even decreased performance, due to the additional latency and power consumption associated with larger caches. This is because larger caches have longer access times, which can slow down the processor if the cache is not being fully utilized.

But in reality , we have kept the L1 access time same not matter what is the size of the cache , so the total access time decreases on increasing the the size but the rate of decrement slows down after increasing to 4096 bytes.This is because the total access time not only depends on L1 cache(whose access time has saturated) but also on L2 cache .

Reason for very little decrease in trace 5 and 6 : if increasing the L1 cache size does not result in a significant decrease in total access time, it may indicate a low cache conflicts or a high cache hit rate in the trace file and therefore increasing cache size doesn't have a significant impact on total access time.



(a) Trace 1,Trace 2,Trace 3,Trace 4



(b) Trace 5,Trace 6,Trace 7,Trace 8

Figure 3: L1_ASSOC Variation

3 On varying the L1- Associativity

3.1 Trace 1-4

These traces have smaller size i.e. 1 lakhs cache instructions. For all traces the total access time decreases if we increase the associativity upto 4 but on further increasing the associativity ,there is no significant change. For trace 2 the graph is the steepest upto 4.

3.2 Trace 5-8

These observations are interesting and differ from the trend followed by trace 1-4. trace 5 and trace 8 show similar trends as above but for trace 6 the total access time decrease is very little , this shows that the effect of increase in conflict due to reduced no. of indexes has been cancelled out by the increase in associativity and hence reduced conflicts. For trace 7 the access time decreases upto 4 but further increases marginally on increasing the associativity .

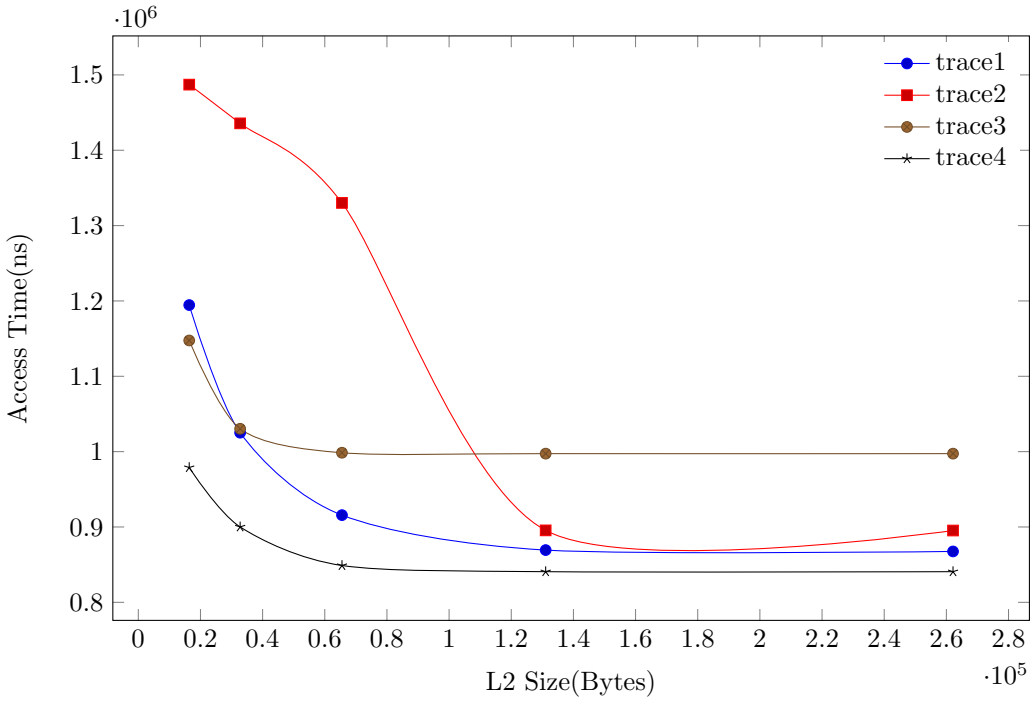
3.3 Explanation

If we increase the associativity ,we reduce the number of sets as the total size of cache is constant and thus the number of cache lines in each set will increase, and the probability of cache conflicts will decrease. This means that the cache hit rate will increase, and the total access time may improve.

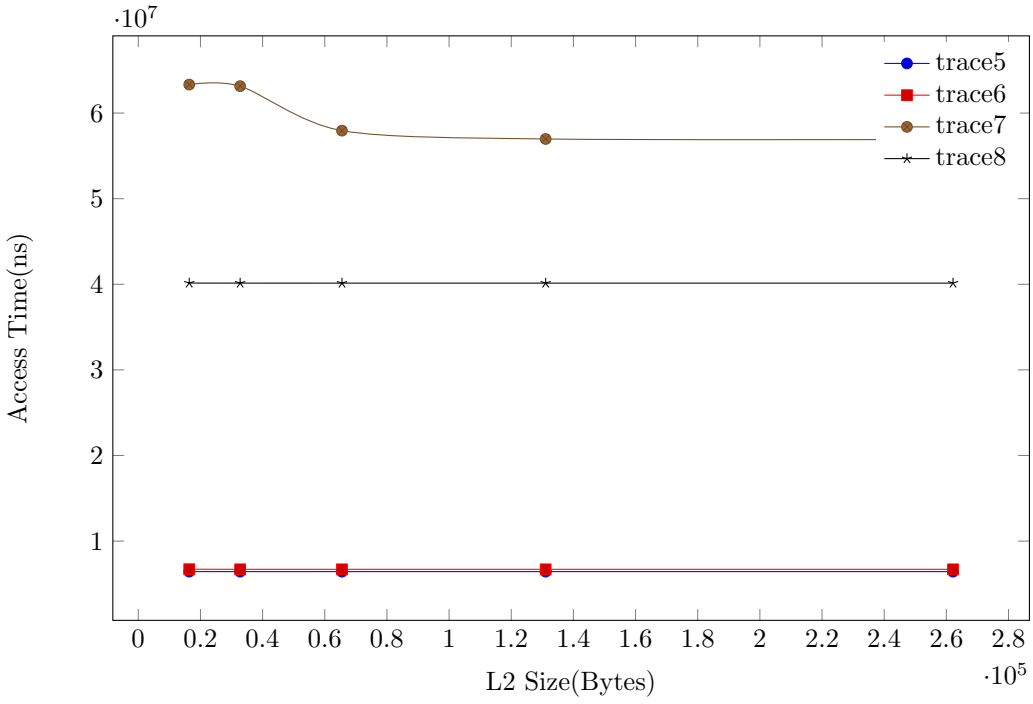
However, reducing the number of sets also means that the number of cache lines in the cache will decrease. This may lead to more cache misses, as the cache may not be able to store all the required cache lines.

So we have to find an optimal associativity which gives optimal time.

For trace 7 the increase in the time on increasing the associativity to 8 may be explained as : It can be a case, on reducing set size further , many more addresses that were referring to different indexes earlier may now refer to the same index and effect of increase in the associativity is dominated by the effect of decreased no. of indexes ,misses are higher than before and thus there is an increase in access time.



(a) Trace 1,Trace 2,Trace 3,Trace 4



(b) Trace 5,Trace 6,Trace 7,Trace 8

Figure 4: L2_SIZE Variation

4 On varying the L2-size

4.1 Trace 1-4

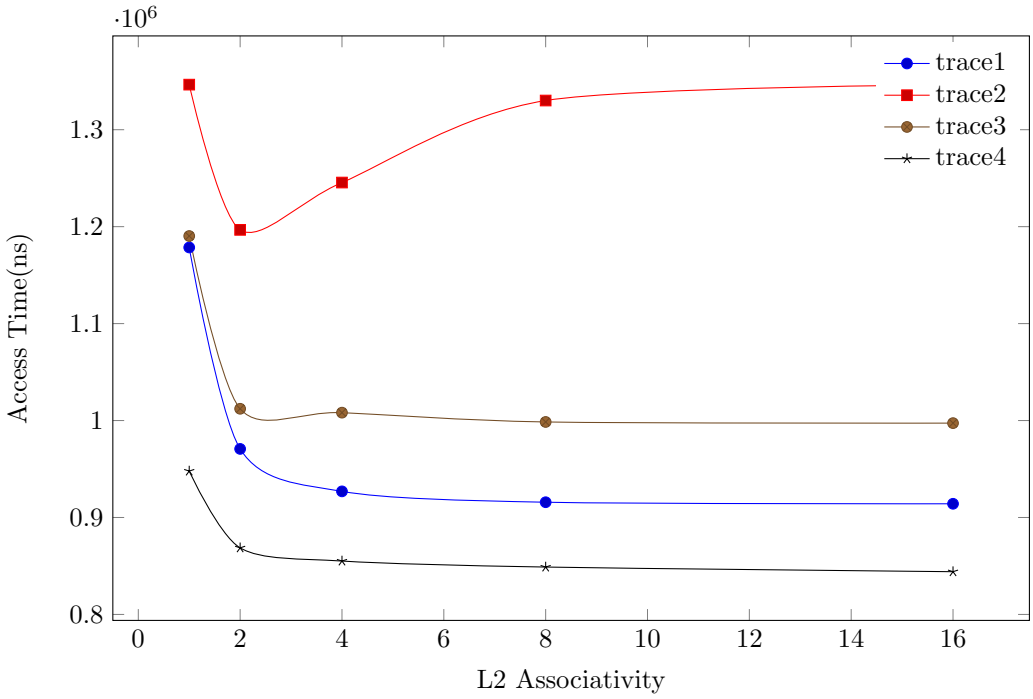
These traces have smaller size i.e. 1lakhs cache instructions.On increasing the cache size the total time decreases for all traces but the decrease is not uniform. We observe that the curve is steepest for trace 2 which implies that there are large number of conflicts in case of less no. of sets and thus on increasing the cache size resulting in increase in no. of sets reduced the cache conflicts improving the total access time.

4.2 Trace 5-8

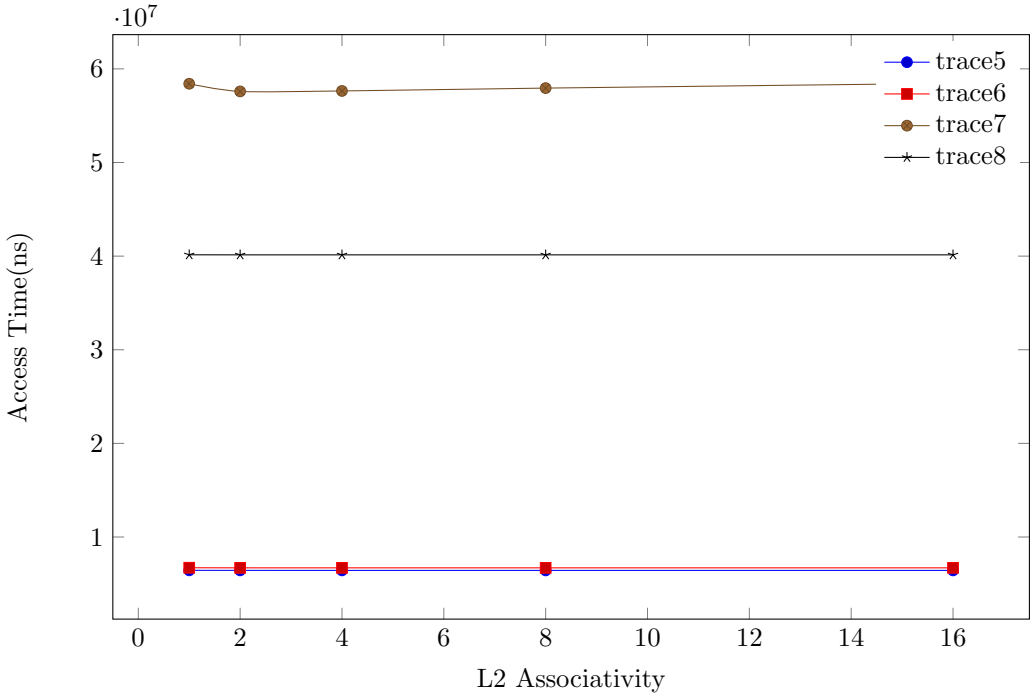
Only trace 7 shows a little decrease in total access time . The time for traces 5,6 and 8 almost remains constant . This signifies that there were very less conflicts for the same index for these traces as the number of read misses are very less and thus increasing size further does not have any significant effect on these traces.

4.3 Explanation

General explanation same as L1 cache. The increase in L2-size definitely increases the chances of cache hit . But if there are less conflicts in the cache for the same index, increasing the index size doesn't help much after some extent as little to no conflicts remain . This was seen in case of trace 1,3,4, and 7. For trace 2 there may be a large number of conflicts for some index and thus total access time decreased steeply .



(a) Trace 1,Trace 2,Trace 3,Trace 4



(b) Trace 5,Trace 6,Trace 7,Trace 8

Figure 5: L2_ASSOC Variation

5 On varying the L2-Associativity

5.1 Trace 1-4

These traces have smaller size i.e. 1 lakhs cache instructions. For trace 2 there may be a lot of conflicts for some indexes and initially on increasing the associativity the conflicts resolve but on further increasing the effect of decrease in no. of sets dominates the effect of increased associativity . Thus on initially increasing the the associativity the time decreases but eventually reaches saturation on further increment due to equal effects of both no. of sets and associativity factor.

Trace 1,3,4 show similar trends as L1-cache .

5.2 Trace 5-8

All the traces show negligible (trace 7 shows little decrease initially) change on changing the associativity. This may be because there may be very little to no cache-conflicts in the L2 caches for these traces or the effects of increase and decrease of associativity and sets respectively cancels out and thus increasing associativity doesn't have any significant impact.

5.3 Explanation

Increasing L2 associativity can reduce the number of cache conflicts in L2 cache, and thus increase the cache hit rate. This can lead to lower total access times. However, increasing associativity also leads to an decrease in the number of sets of the cache, which can lead to higher misses in some case and thus higher access times

Therefore, increasing the L2 associativity from 1 to 2 or 4 may lead to a reduction in total access time, as the cache hit rate improves. However, further increasing the L2 associativity beyond 4 may lead to diminishing returns and longer access times due to decreasing number of sets.

For larger trace files (5,6,7,8) the number of L2 misses is very less which implies the number of cache conflicts will also be very less, and hence increase in associativity plays very little role.

6 Value of a particular parameter for Optimal Condition keeping the other parameters constant.

6.1 Trace1

- Blocksize - 128
- L1 size - 8192
- L1 Associativity -16
- L2 size -131072
- L2 Associativity -16

6.2 Trace2

- Blocksize - 128
- L1 size - 8192
- L1 Associativity - 16
- L2 size - 262144
- L2 Associativity - 2

6.3 Trace3

- Blocksize - 64
- L1 size - 8192
- L1 Associativity - 16
- L2 size - 262144
- L2 Associativity - 16

6.4 Trace4

- Blocksize - 32
- L1 size - 8192
- L1 Associativity - 16
- L2 size - 131072
- L2 Associativity - 16

6.5 Trace5

- Blocksize - 64
- L1 size - 8192
- L1 Associativity - 16
- L2 size - 262144
- L2 Associativity - 16

6.6 Trace6

- Blocksize- 32
- L1 size - 8192
- L1 Associativity - 16
- L2 size - 262144
- L2 Associativity - 16

6.7 Trace7

- Blocksize - 64
- L1 size - 8192
- L1 Associativity - 4
- L2 size - 262144
- L2 Associativity - 2

6.8 Trace8

- Blocksize - 32
- L1 size - 8192
- L1 Associativity -16
- L2 size -262144
- L2 Associativity -16

7 Design Decisions

7.1 L1 Reads

L1_read++

If tag is found: Read hit

If tag not found: L1_read_miss++

read request to L2(fetch the cache from L2) → Now the fetched block is to be stored in L1 cache → two situations!

I) If cache line is empty → insert the cache → dirty bit =0 and update LRU for that index.

II) If any of the cache lines are not empty → evict the LRU cache and put the new cache block there →if the evicted cache has dirty bit 1 → write back to L2 else do nothing → update LRU for that index and set dirty bit =0.

7.2 L1 Writes

L1_write++

If tag is found: write hit → change dirty bit to 1

If tag not found: L1_write_miss ++

read request to L2(fetch the cache from L2) → Now the fetched block is to be stored in L1 cache → two situations!

I) If a cache line is empty → insert the cache → dirty bit =1 and update LRU for that index.

II) If any of the cache lines are not empty → evict the LRU cache and put the new cache block there →if the evicted cache has dirty bit 1 → write back to L2 else do nothing → update LRU for that index → set the dirty bit of new cache as 1

7.3 L2 Reads

L2_read ++

If tag is found: L2_read_hit ++

If tag is not found: L2_read_miss ++

Fetch cache block from memory → two situations

I) If a cache line is empty → insert the block → dirty bit =0 and update LRU for that index.

II) If any of the cache lines are not empty→ evict the LRU cache and put the new cache block there →if the evicted cache has dirty bit 1 → memory_wrtbk++ -→ update LRU for that index and dirty bit =0

7.4 L2 writes

L2_write ++

If tag is found : L2_wrt_hit ++

If tag is not found : L2_wrt_miss ++

Fetch cache block from memory → two situations

I) If a cache line is empty → insert the block → dirty bit =1 and update LRU for that index.

II) If any of the cache lines are not empty → evict the LRU cache and put the new cache there →if the evicted cache block has dirty bit 1 → memory_wrtbk++ -→ update LRU for that index and set dirty bit for block to 1.