I. Introduction

In the realm of computer architecture, a cache serves as a hardware component that stores recently accessed data with the aim of expediting future requests for that same data. This small and fast type of memory is positioned between the processor and main memory and its primary objective is to reduce the time required to access data from main memory, which is typically slower than accessing data from the cache. As an essential part of modern computer architecture, cache memory has a significant impact on system performance.

As part of our project, we have created a simulator code for an arbitrary n-way set associative cache in C++. Our simulator implements two levels of caches, namely L1 and L2. By reading trace files, the simulator assigns requests to the L1 cache, which in turn sends read/write requests to the L2 cache. The L2 cache interacts with DRAM and both L1 and L2 caches keep track of their own counters such as reads, writes, misses, hits, and so on. Upon completion of the simulation, our program prints the statistics for both caches on the console, as specified in the problem statement.

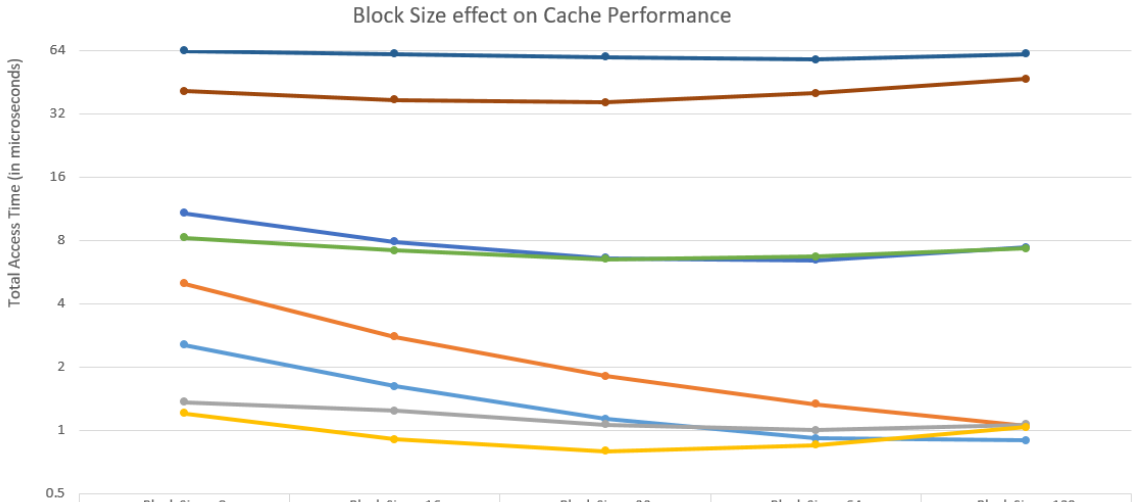
II. Configurable Parameters

The following parameters can be configured in the simulator

(i) BLOCKSIZE: Number of bytes in a block/line.
(ii) L1_Size: Total bytes of data storage for L1 Cache.
(iii) L1_Assoc: Associativity of L1 Cache.
(iv) L2_Size: Total bytes of data storage for L2 Cache.
(v) L2_Assoc: Associativity of L2 Cache.

III. Observations

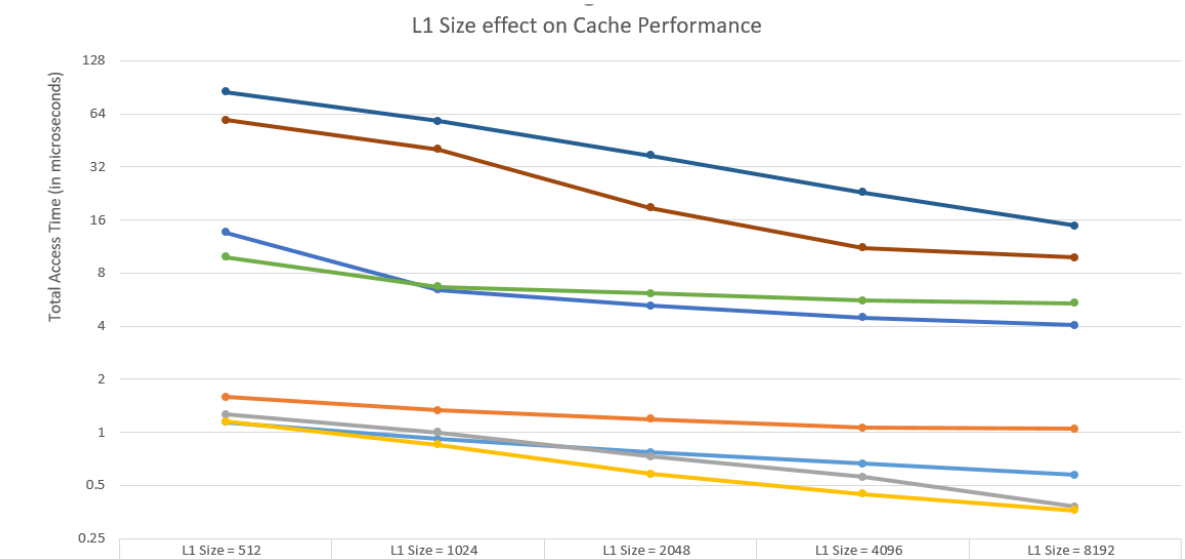
(i) Impact of varying BLOCKSIZE on cache access time



	Block Size = 8	Block Size = 16	Block Size = 32	Block Size = 64	Block Size = 128
Trace File 1	2.5371	1.61156	1.1279	0.91574	0.89236
Trace File 2	4.96542	2.78566	1.81182	1.3305	1.04438
Trace File 3	1.35952	1.23456	1.0629	0.99854	1.06136
Trace File 4	1.20456	0.90156	0.79292	0.84888	1.03042
Trace File 5	10.7726	7.86632	6.5942	6.43386	7.386
Trace File 6	8.22094	7.1649	6.50064	6.7065	7.3216
Trace File 7	63.7074	61.5031	59.6349	57.9485	61.5922
Trace File 8	40.9921	37.1581	36.2076	40.1353	46.7842

- Increasing block size decreases the number of blocks in cache and reduces compulsory misses.
- However, if block size becomes too large, cache capacity may become a limiting factor.
- As block size increases, cache access time may also increase due to longer transfer times for larger blocks.
- Increasing block size can improve cache hit rate by reducing the number of blocks that must be searched for a given address, but may also increase cache miss rate if there is more data in a block than necessary.
- Smaller block sizes tend to reduce cache miss rate by allowing more fine-grained data placement, but may increase cache overhead due to more frequent tag lookups.

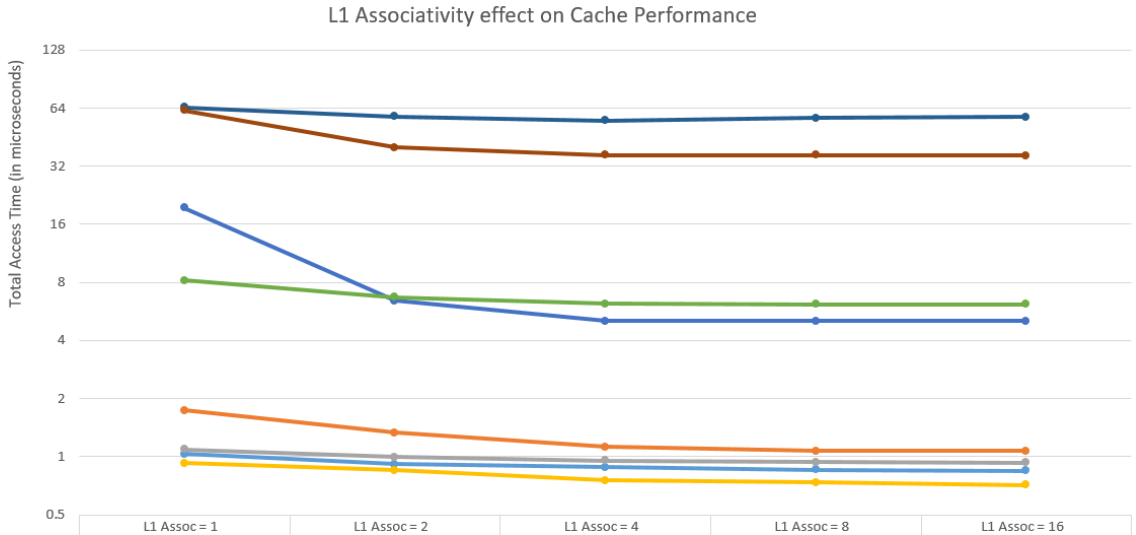
(ii) Impact of varying L1_Size on cache access time



	L1 Size = 512	L1 Size = 1024	L1 Size = 2048	L1 Size = 4096	L1 Size = 8192
Trace File 1	1.14128	0.91574	0.76968	0.6653	0.57382
Trace File 2	1.58768	1.3305	1.18462	1.05812	1.04684
Trace File 3	1.26376	0.99854	0.73012	0.55618	0.37768
Trace File 4	1.1496	0.84888	0.58148	0.44754	0.3619
Trace File 5	13.6141	6.43386	5.21278	4.47392	4.04628
Trace File 6	9.87294	6.7065	6.12448	5.5629	5.40298
Trace File 7	85.3738	57.9485	37.1926	22.8969	14.8066
Trace File 8	58.8197	40.1353	18.6874	11.1304	9.80548

- Increasing L1 cache size reduces miss rates, improving cache access time.
- However, if L1 cache size becomes too large, access time may increase due to longer search times.
- Larger L1 cache sizes tend to improve hit rate and reduce miss rate, but may increase cache access time due to longer search times and tag lookup times.
- Smaller L1 cache sizes tend to have lower hit rates and higher miss rates, but may have faster access times due to smaller search and tag lookup times.
- The optimal L1 size depends on the specific workload and access patterns of the application being run.

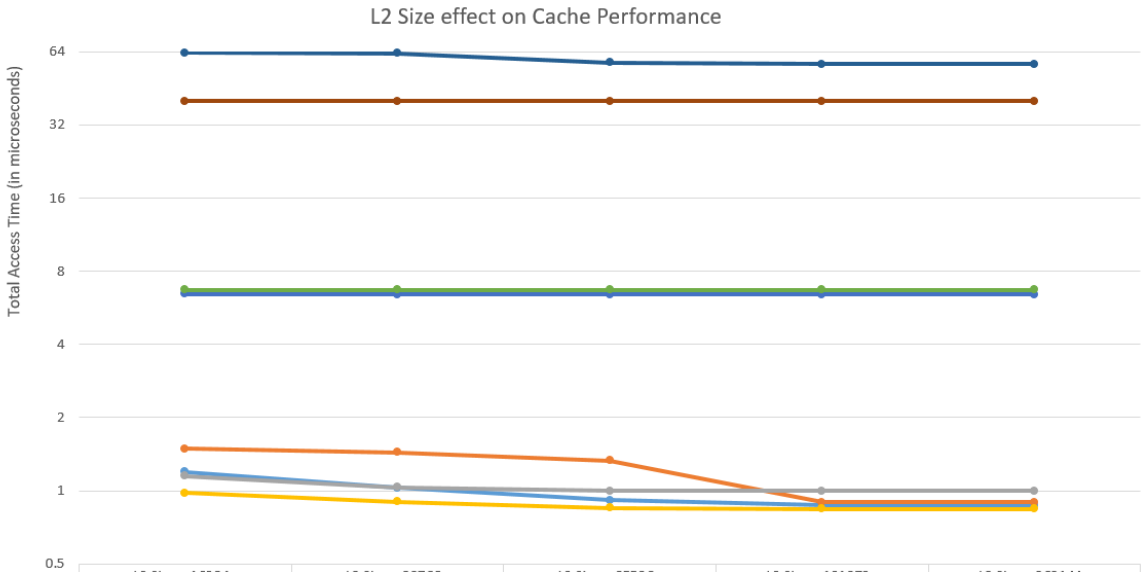
(iii) Impact of varying L1_Assoc on cache access time



	L1 Assoc = 1	L1 Assoc = 2	L1 Assoc = 4	L1 Assoc = 8	L1 Assoc = 16
Trace File 1	1.03058	0.91574	0.88262	0.85504	0.84714
Trace File 2	1.74324	1.3305	1.1246	1.07206	1.06914
Trace File 3	1.09194	0.99854	0.95164	0.93792	0.92996
Trace File 4	0.92692	0.84888	0.75188	0.73564	0.7146
Trace File 5	19.5057	6.43386	5.0419	5.04406	5.03946
Trace File 6	8.18916	6.7065	6.21036	6.1558	6.15368
Trace File 7	64.6757	57.9485	55.2493	57.053	57.4344
Trace File 8	62.4426	40.1353	36.5163	36.4836	36.4434

- Increasing L1 associativity reduces conflict misses and improves cache access time.
- However, higher associativity also increases search time.
- Higher L1 associativity tends to improve hit rate by reducing conflict misses, but may also increase cache access time due to longer search and tag lookup times.
- Lower L1 associativity tends to increase conflict misses, but may have faster access times due to shorter search and tag lookup times.

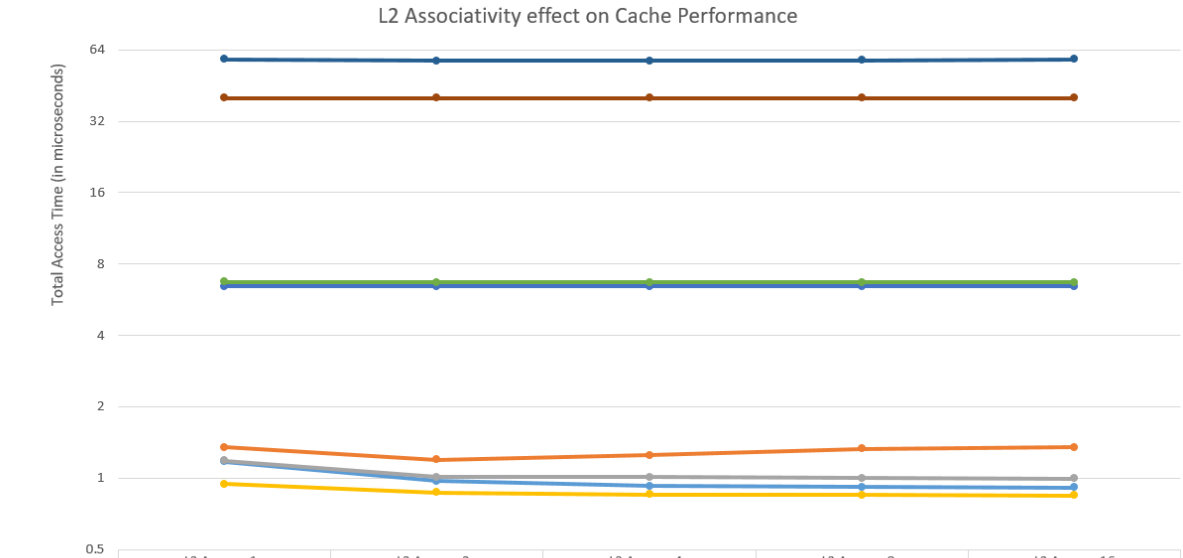
(iv) Impact of varying L2_Size on cache access time



	L2 Size = 16384	L2 Size = 32768	L2 Size = 65536	L2 Size = 131072	L2 Size = 262144
Trace File 1	1.19614	1.02514	0.91574	0.86934	0.86754
Trace File 2	1.4871	1.4357	1.3305	0.8955	0.8953
Trace File 3	1.14754	1.03014	0.99854	0.99734	0.99734
Trace File 4	0.97888	0.90008	0.84888	0.84068	0.84068
Trace File 5	6.43666	6.43386	6.43386	6.43386	6.43386
Trace File 6	6.7167	6.7065	6.7065	6.7065	6.7065
Trace File 7	63.3273	63.1437	57.9485	56.9719	56.9109
Trace File 8	40.1433	40.1353	40.1353	40.1353	40.1353

- Increasing L2 cache size reduces miss rates, improving cache access time.
- However, if L2 cache size becomes too large, access time may increase due to longer search times.
- Larger L2 cache sizes tend to reduce miss rate by providing a larger secondary cache, but may increase cache access time due to longer search and tag lookup times.
- Smaller L2 cache sizes tend to increase miss rate but may have faster access times due to shorter search and tag lookup times.
- The optimal L2 size depends on the specific workload and access patterns of the application being run.

(v) Impact of varying L2_Assoc on cache access time



	L2 Assoc = 1	L2 Assoc = 2	L2 Assoc = 4	L2 Assoc = 8	L2 Assoc = 16
Trace File 1	1.17514	0.97074	0.92694	0.91574	0.91414
Trace File 2	1.3467	1.1983	1.2505	1.3305	1.3467
Trace File 3	1.18694	1.01234	1.00814	0.99854	0.99734
Trace File 4	0.94608	0.86868	0.85508	0.84888	0.84408
Trace File 5	6.43986	6.43626	6.43386	6.43386	6.43386
Trace File 6	6.7183	6.7071	6.7065	6.7065	6.7065
Trace File 7	58.3867	57.5899	57.6435	57.9485	58.4571
Trace File 8	40.1425	40.1353	40.1353	40.1353	40.1353

- Increasing L2 associativity reduces conflict misses and improves cache access time.
- However, higher associativity also increases search time.
- Higher L2 associativity tends to improve hit rate by reducing conflict misses, but may also increase cache access time due to longer search and tag lookup times.
- Lower L2 associativity tends to increase conflict misses, but may have faster access times due to shorter search and tag lookup times.

IV. References

(i) Computer Organization and Architecture by William Stallings
(ii) Cache memory – GeeksforGeeks
(iii) Introduction to Cache Memory – tutorialspoint
(iv) Computer Architecture - Carnegie Mellon University
(v) Cache hierarchy - University of Maryland