COL216 Assignment-3 Report

1 Introduction

1.1 Aim

The aim of this assignment is to simulate the working of a 2-level cache consisting of L1 and L2. The simulator will read through trace files consisting of the accesses with their type(r/w) followed by the corresponding address. The trace files can be considered to be generated by the processor which are then forwarded to the L1 cache. The L1 cache then interacts with the L2 cache in case of a miss which in turn interacts with the DRAM.

The number of cycles taken for the L1, L2 cache and DRAM access is assumed to be 1, 20 and 200 nanoseconds respectively.

We follow an LRU eviction policy in both caches. Further, we use a write-back (we only write to the very next level in case of an eviction and use a dirty bit/boolean to denote that a modification has occurred) and a write-allocate (we bring the associated block all the way up to L1 and then write to it) policy.

1.2 Parameters

The simulator takes in the following 5 configurable parameters to dynamically allocate the memory associated with the L1 and L2 caches. The parameters are as follows:-

1. BLOCKSIZE :- The number of bytes present per block in both the caches.
2. L1_SIZE :- The total number of bytes that can be stored in L1.
3. L1_ASSOC :- The number of ways/lines per set in L1.
4. L2_SIZE :- The total number of bytes that can be stored in L2.
5. L2_ASSOC :- The number of ways/lines per set in L2.

The above parameters should all be powers of 2. The 5 plots for total access time generated by varying 1 of the above parameters while keeping the rest fixed can be can be found later in the report.

2 Plots for Block_Size

The total access time in nanoseconds for the given 8 trace files as a function of varying block size while keeping the remaining four parameters fixed is as follows:-

Changing Parameter	Trace_Number	8	16	32	64	128
BLOCKSIZE	Trace_1	2537100	1611560	1127900	915740	892360
	Trace_2	4965420	2785660	1811820	1330500	1044380
	Trace_3	1359520	1234560	1062900	998540	1061360
	Trace_4	1204560	901560	792920	848880	1030420
	Trace_5	10772600	7866320	6594200	6433860	7386000
	Trace_6	8220940	7164900	6500640	6706500	7321600
	Trace_7	63707360	61503140	59634940	57948460	61592220
	Trace_8	40992140	37158140	36207620	40135320	46784200

Table 1: Total Access Time for various block sizes

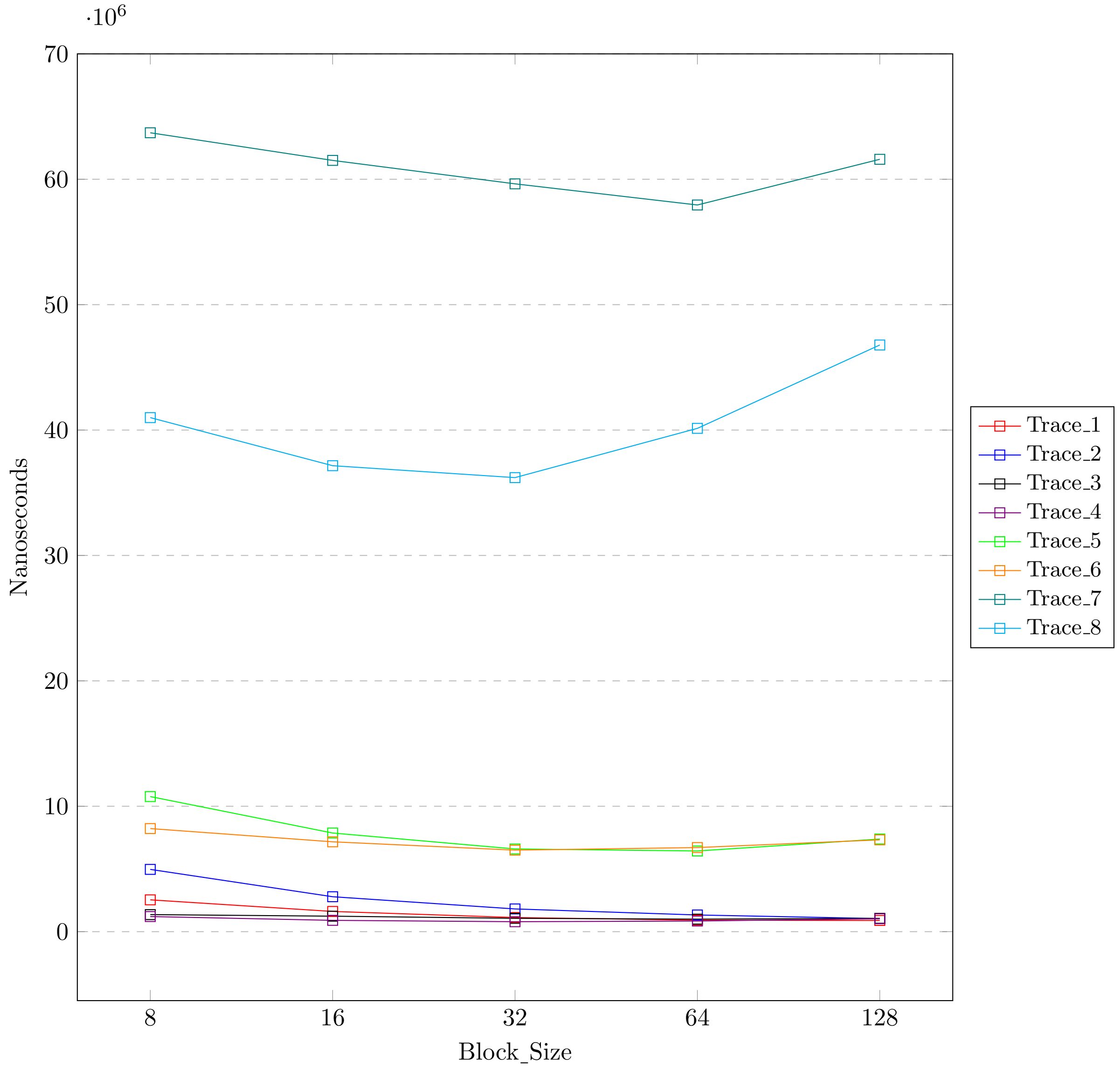

Figure 1: Total Access Time versus Block Size

3 Plots for L1_Size

The total access time in nanoseconds for the given 8 trace files as a function of varying size of L1 while keeping the remaining four parameters fixed is as follows:-

Changing Parameter	Trace_Number	512	1024	2048	4096	8192
L1_SIZE	Trace_1	1141280	915740	769680	665300	573820
	Trace_2	1587680	1330500	1184620	1058120	1046840
	Trace_3	1263760	998540	730120	556180	377680
	Trace_4	1149600	848880	581480	447540	361900
	Trace_5	13614080	6433860	5212780	4473920	4046280
	Trace_6	9872940	6706500	6124480	5562900	5402980
	Trace_7	85373840	57948460	37192640	22896880	14806620
	Trace_8	58819680	40135320	18687360	11130400	9805480

Table 2: Total Access Time for various L1 sizes

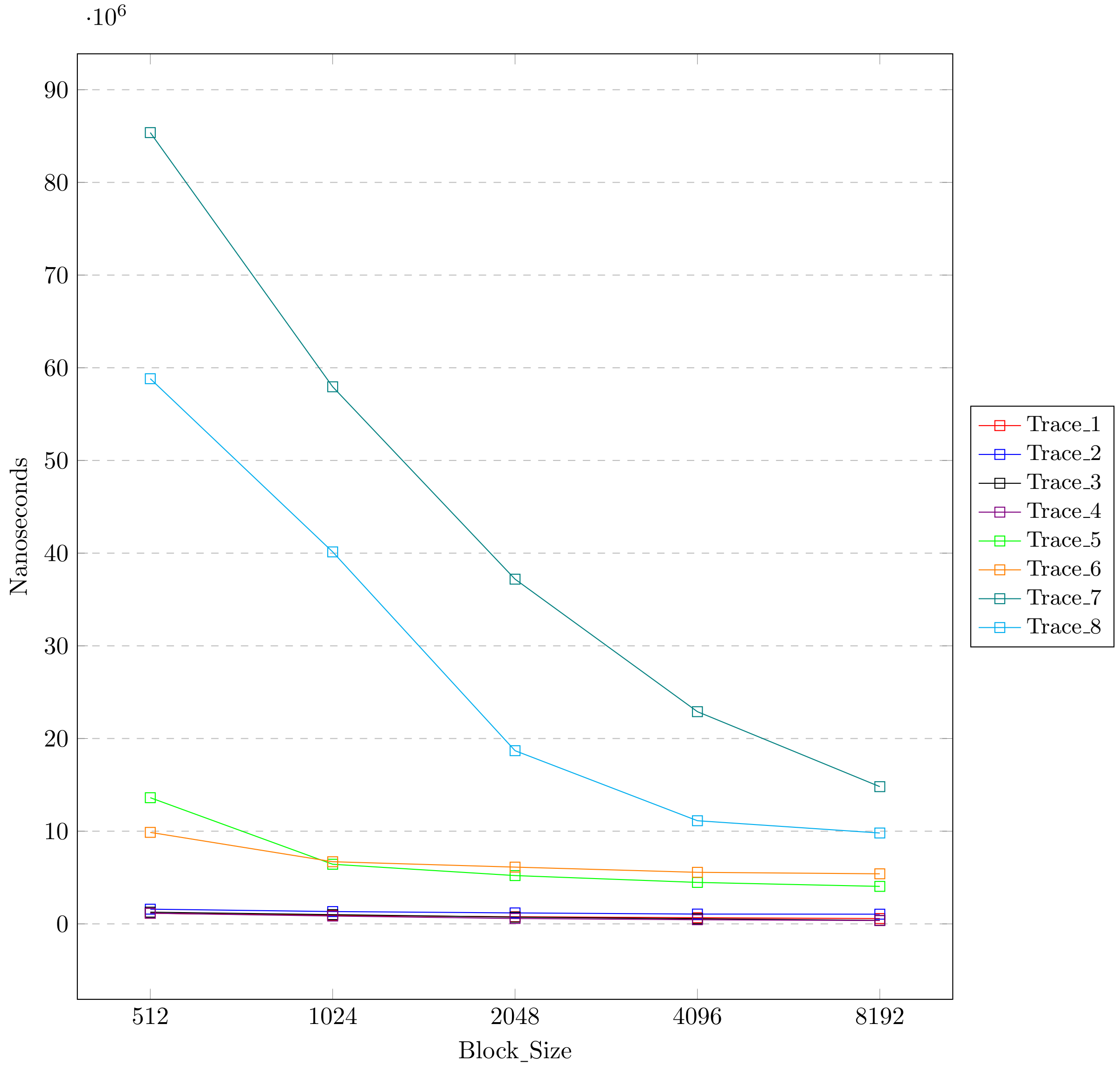

Figure 2: Total Access Time versus L1 size

4 Plots for L1_Assoc

The total access time in nanoseconds for the given 8 trace files as a function of varying associativity of L1 while keeping the remaining four parameters fixed is as follows:-

Changing Parameter	Trace_Number	1	2	4	8	16
L1_ASSOC	Trace_1	1030580	915740	882620	855040	847140
	Trace_2	1743240	1330500	1124600	1072060	1069140
	Trace_3	1091940	998540	951640	937920	929960
	Trace_4	926920	848880	751880	735640	714600
	Trace_5	19505720	6433860	5041900	5044060	5039460
	Trace_6	8189160	6706500	6210360	6155800	6153680
	Trace_7	64675680	57948460	55249260	57053000	57434380
	Trace_8	62442600	40135320	36516340	36483580	36443420

Table 3: Total Access Time for various L1 associativity values

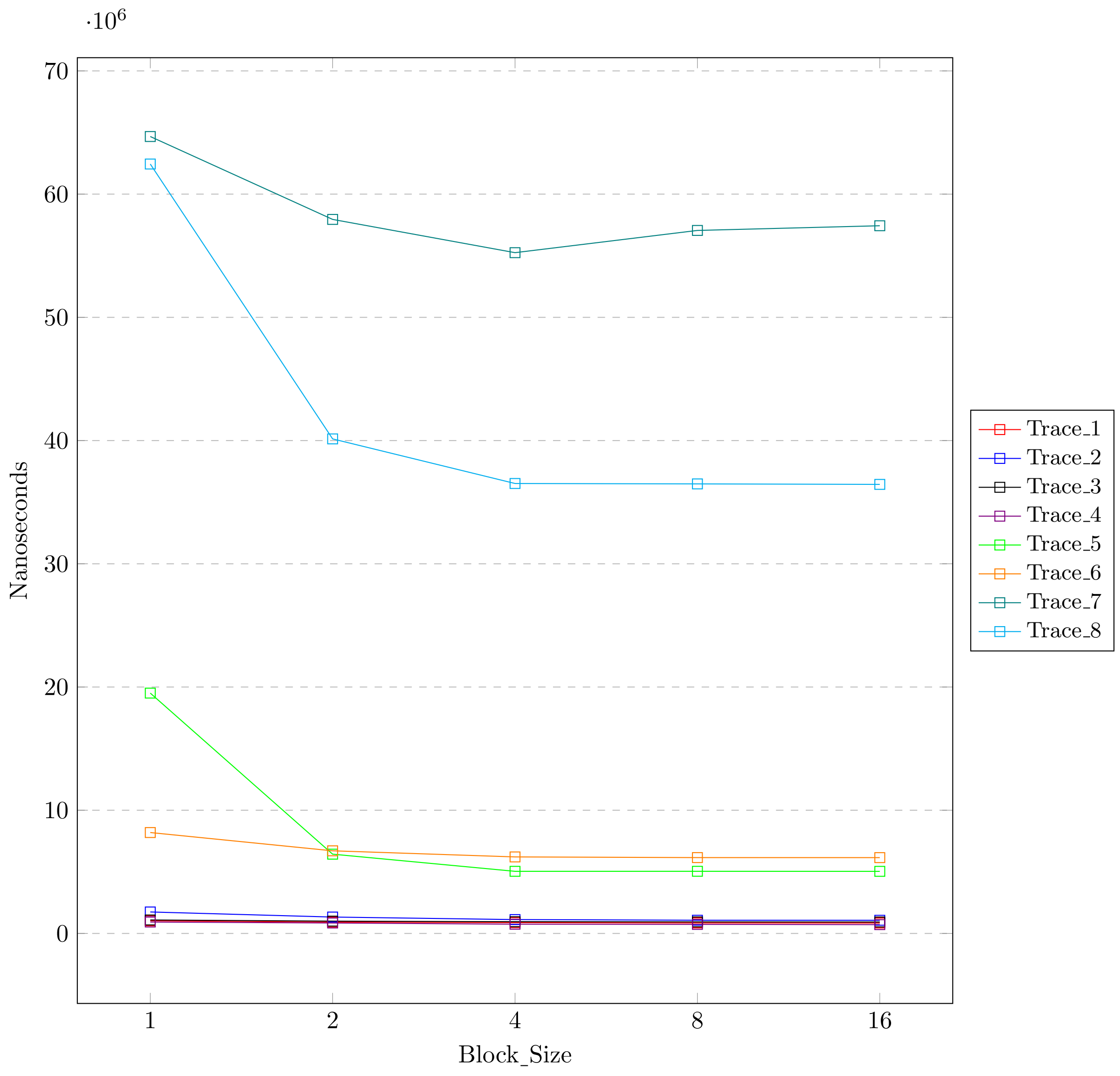

Figure 3: Total Access Time versus L1 associativity

5 Plots for L2_Size

The total access time in nanoseconds for the given 8 trace files as a function of varying size of L2 while keeping the remaining four parameters fixed is as follows:-

Changing Parameter	Trace_Number	16384	32768	65536	131072	262144
L2_SIZE	Trace_1	1196140	1025140	915740	869340	867540
	Trace_2	1487100	1435700	1330500	895500	895300
	Trace_3	1147540	1030140	998540	997340	997340
	Trace_4	978880	900080	848880	840680	840680
	Trace_5	6436660	6433860	6433860	6433860	6433860
	Trace_6	6716700	6706500	6706500	6706500	6706500
	Trace_7	63327260	63143660	57948460	56971860	56910860
	Trace_8	40143320	40135320	40135320	40135320	40135320

Table 4: Total Access Time for various L2 sizes

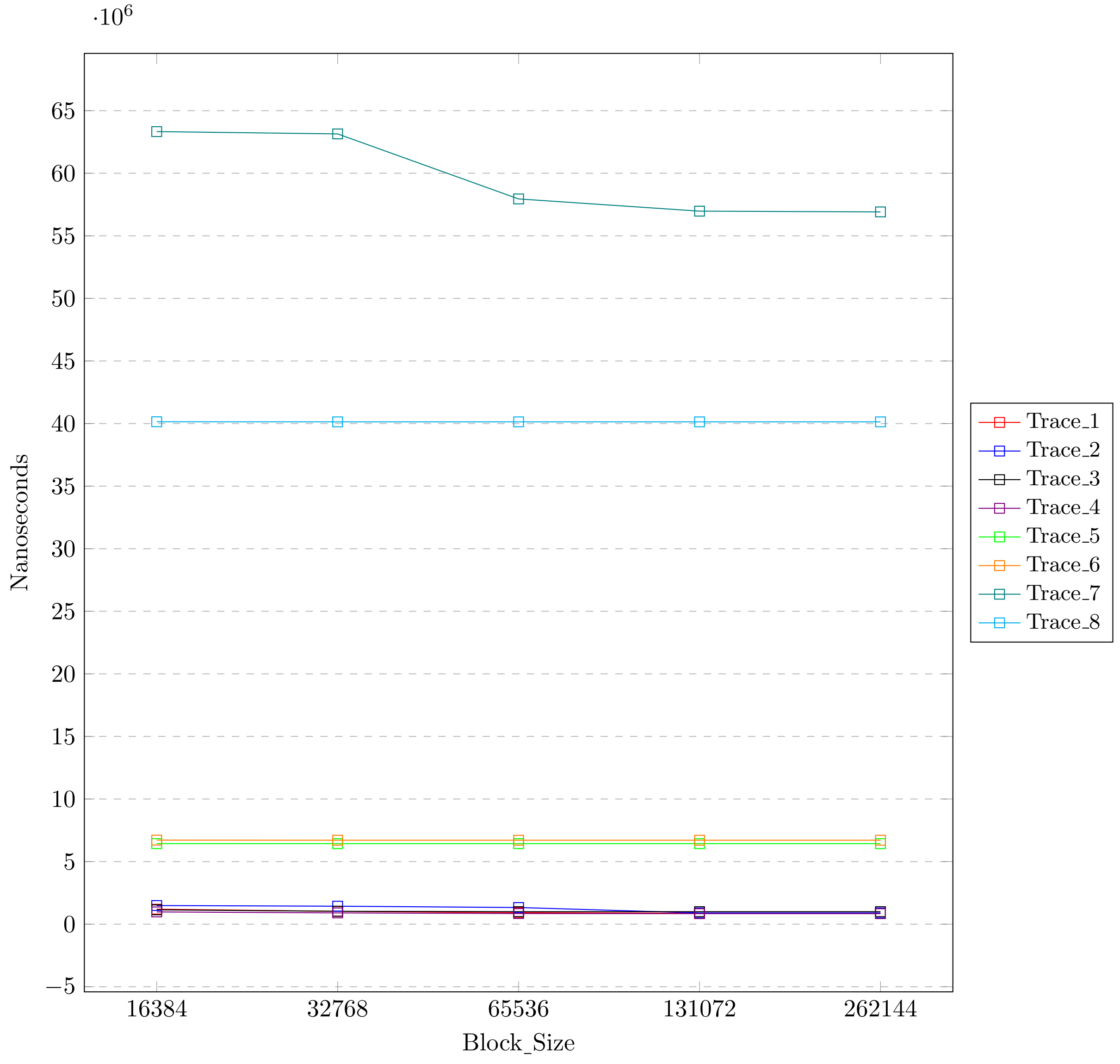

Figure 4: Total Access Time versus L2 size

6 Plots for L2_Assoc

The total access time in nanoseconds for the given 8 trace files as a function of varying associativity of L2 while keeping the remaining four parameters fixed is as follows:-

Changing Parameter	Trace_Number	1	2	4	8	16
L2_ASSOC	Trace_1	1176140	970740	926940	915740	914140
	Trace_2	1346700	1198300	1250500	1330500	1346700
	Trace_3	1186940	1012340	1008140	998540	997340
	Trace_4	946080	868680	855080	848880	844080
	Trace_5	6439860	6436260	6433860	6433860	6433860
	Trace_6	6718300	6707100	6706500	6706500	6706500
	Trace_7	58387660	57590060	57643460	57948460	58457060
	Trace_8	40142520	40135320	40135320	40135320	40135320

Table 5: Total Access Time for various L2 associativity values

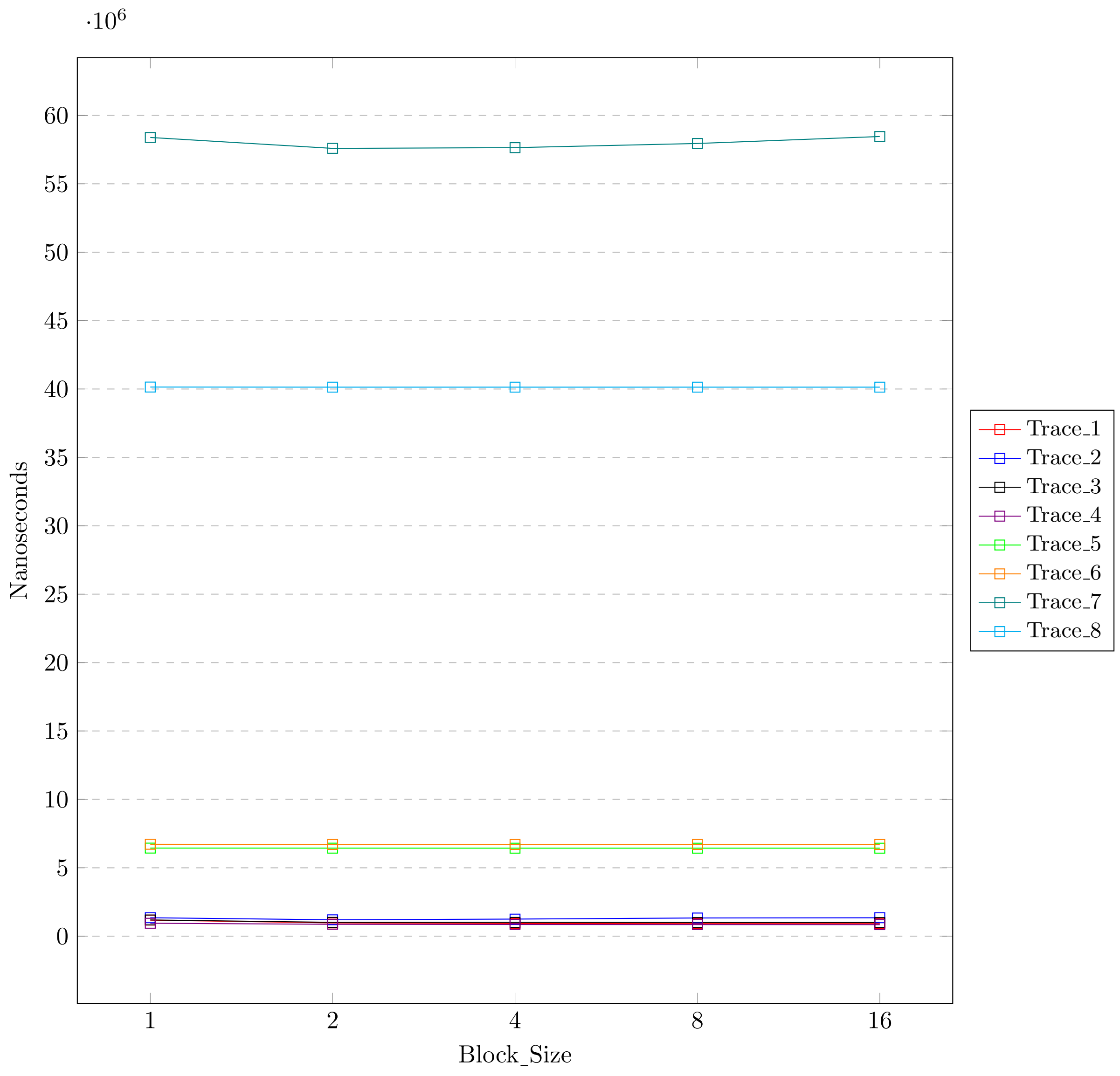

Figure 5: Total Access Time versus L2 associativity

7 Plot Conclusions

Let us now look at the key observations based on the above plots which corelate the effects of each varying parameter on the the Total Access Time.

7.1 Varying Block Size

The below points show the two processes which compete with each other to either ultimately affect the miss rate and thereby also the total access time.

1. **Block Size**: The increase in the block size causes the miss rate to decrease initially in the graph because of increase in number of bytes that are brought to the caches from lower levels (larger storage) thereby allowing us to carry out many more accesses without having to go to the lower levels.

2. **Number of Sets**: The number of sets decreases as we increase the block size because we keep the remaining 4 parameters constant. When the block size becomes a significant fraction of the cache size, the number of collisions causes the block to be evicted without many accesses corresponding to it. This will result in going to the lower levels several times and increase the total access time significantly.

The trade-off between the above two points eventually decides the total access time. We also observe that the graph reaches the positive slope area only in the case of larger files (Trace-3 to 8) because of the higher dependence on the number of sets in these cases. Note that in general, the increase in a block size significantly increases the cost of a miss due to a larger bus traffic. This factor does not affect our graph because of the constant time assumption.

7.2 Varying L1 Size

The below points show the two processes which compete with each other to either ultimately affect the total access time.

1. **Hit Rate**: The increase in size of L1 is same as increasing the number of sets in L1 while keeping other parameters fixed. This leads to a much better hit rate as the number of collisions decreases significantly. Thus, the total access time decreases in all cases with increase in L1 size.

2. **L1 Hit Time**: In general, a larger cache will have a longer hit time as we have to manage a larger number of sets and correspondingly a larger storage itself. This would in general cause in increase in access time.

However, we assume that the L1 hit time is constant for the simulator with respect to the varying parameters. Thus, the factor in point-2 does not come into play in the above graph which is why the total access time decreases in all cases for an increasing L1 cache size.

7.3 Varying L1 Assoc

The below points show the two processes which compete with each other to either ultimately affect the total access time.

1. **Miss Rate**: The increase in the associativity of the L1 Cache causes the miss rate to decrease as the number of lines per set increases by a significant factor. Although the number of blocks that map to the same set increase, this is overshadowed by the increased lines per set atleast initially. This causes the access time to decrease.

2. **Number of sets**: As the number of sets continue to decrease and the number of blocks mapping to the same also increase, this may eventually take effect and cause a larger access time despite increased number of lines as we can see in Trace-7.

In general, the main disadvantage of increasing the associativity is that the hit time increases because we have to search through several lines for a single block. We have however assumed that this time is constant. Therefore, this factor does not come into play in the graph.

7.4 Varying L2 Size

The below points show the two processes which compete with each other to either ultimately affect the total access time.

1. **Hit Rate**: The increase in size of L2 which is nothing but increasing the number of sets in L2 causes the number of collisions to decrease. This can lead to a better hit rate as the number of collisions decreases. Thus, the total access time decreases in all cases with increase in L2 size.

2. **L2 and Large Size**: The design of the L2 caches is such that it decreases the effect of an L1 miss by a decent factor which would otherwise result in a costly DRAM access and thereby a longer penalty. This is possible only if the L2 cache itself is of sufficient size. Since, we already take the L2 cache to be of 16384 bytes, which is large enough, a further increase will not cause much effect on the hit rates. This is the reason for the access time being roughly equal even on changing the L2 size.

7.5 Varying L2 Assoc

The below points show the two processes which compete with each other to either ultimately affect the total access time.

1. **Miss Rate**: The increase in the associativity of the L2 Cache causes the miss rate to decrease as the number of lines per set increases. Although the number of blocks that map to the same set increase, this is overshadowed by the increased lines per set. This causes the access time to decrease slightly.

2. **L2 and Large Size**: The main focus of the L2 cache is decreasing the miss rate as compared to the L1 cache where decreasing hit time is important. Note that in the case of larger files, the decrease in the set size and the increase in lines per set roughly cancel out thereby causing almost same access time.

8 Design Comparison with Inclusivity

The above sections and plots assume that the L1 and L2 are independent caches which implies that blocks can be evicted from L1 as well as L2 irrespective of each other (with proper writebacks). We now compare the above design decision with inclusivity.

Inclusivity: Inclusivity or inclusion implies that all caches at a lower level (L2 in our case) contain all the blocks present in a higher level (namely L1). This means that the higher levels are always subsets of the lower levels.
Inclusive Implementation: An inclusive implementation of the above sections involves changing the eviction process of L2. The eviction of a block from L2 will now involve the removal the block from the L1 cache as well if it is present thereby ensuring that L1 remains a subset of L2. We have implemented the code for the same and added the source file to the submission as inclusive.cpp.

8.1 Observations

1. **Positives**: An inclusive implementation of cache hierarchy implies that there is no need to check if a block is present in L2 and other lower levels if present because L1 will always be a subset of them. This is not the case in the original implementation in which you may have to go further in very rare cases when evictions occur in lower levels as well because they are full. Thus, this policy prevents sudden bus traffic as the element will definitely be there in the immediately lower level. Thus, we can simply write to it and mark the bit dirty.

2. **Negatives**: An eviction in a lower level (such as L2 in our case) involves the inherent complexity of having to invalidate this block from all upper levels (such as L1 in our case) which may take significant time if there are many levels.

8.2 Differences in Time

Cache	8(original)	8(Inclusive)	16(original)	16(Inclusive)
Trace_7 with varying block sizes and remaining original parameters	63707360	63714080	61503140	61504460

Table 6: Original vs Inclusive Comparison for Trace-7 text file

We observe differences between the original and the inclusive implementation for the block size of 8 and 16 in trace7. In these cases, the original implementation clearly runs faster than the inclusive case because keeping the subset condition of inclusivity removes some blocks from the higher level irrespective of how recent they are there. This causes an access to a lower level if this block is required later.

9 Contributions

The folder containing the cpp file with the cache struct and makefile to run the same along with the report PDF files has been zipped and submitted in the specified format. The Team consists of:

- Santhosh Rishi Deshineni(2021CS10564)
- Pothamsetty Chethan Manogna Sai(2021CS10561)

Contributions	**Santhosh**	**Chethan**
Total	**50%**	**50%**

Table 7: Contributions