DETAILED STUDY OF CACHE SIMULATOR

BY- SIDDHANT ARORA Entry No. -2015CS50480

ANUJ DHAWAN Entry No. -2015CS10213

In this report, we will specify the result of our analysis of various cache configuration on the three sample traces provided to us namely- spice, gcc and Tex. We will achieve this objective by varying one of the cache parameters keeping all other parameters at fixed value and then analyzing its effect on cache performance in terms of hit rate or memory traffic.

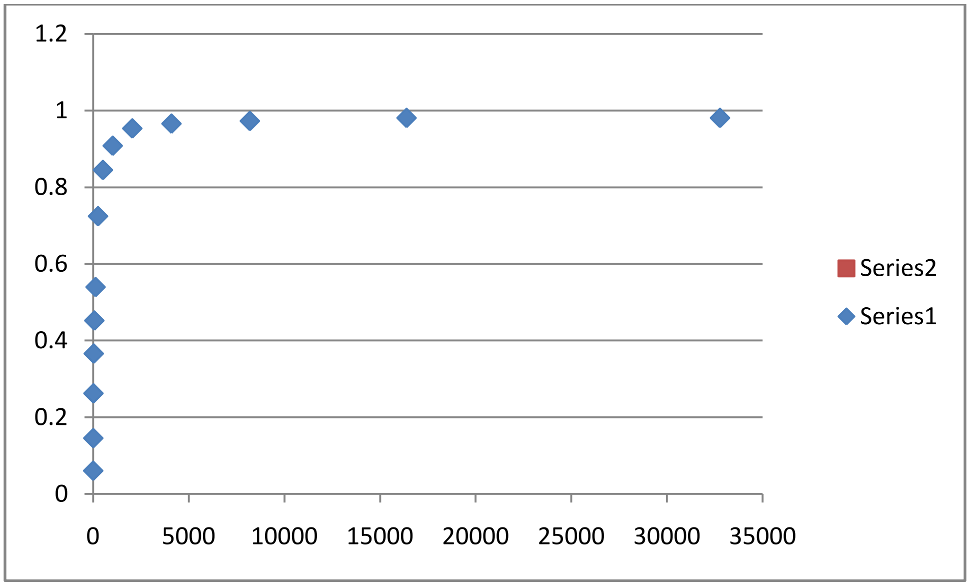
HIT RATE VS CACHE SIZE

In this analysis, we will keep block size fix to 4 bytes, use write back and write allocate and in other to make our cache fully associative we will set associativity to block size divided by cache size so that our cache is fully associative. This is to ensure that there are no conflict misses and all the misses that we see are either capacity or compulsory misses. But since compulsory misses do not change with cache size, hence what we are observing is variation of capacity misses with cache size.

Spice Trace File

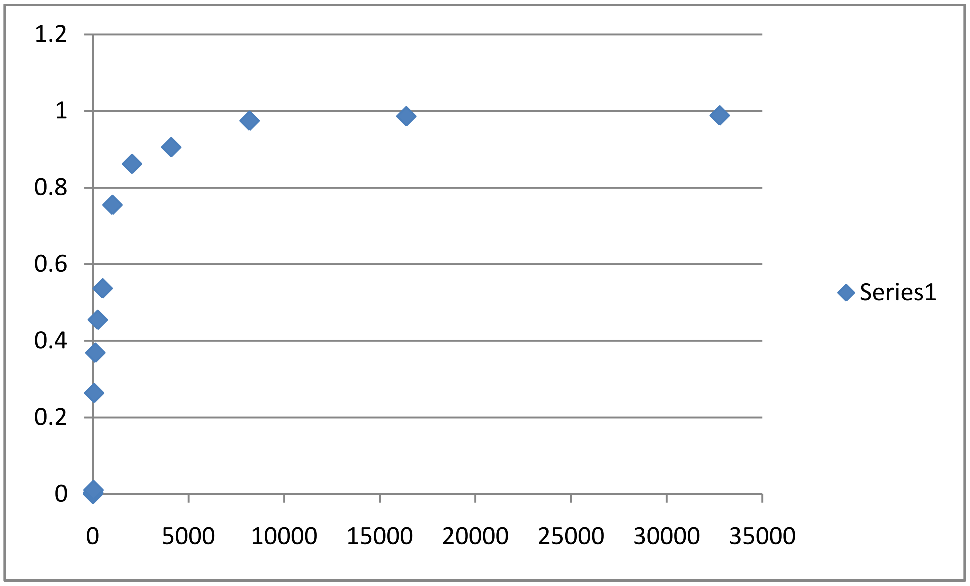
Data Cache Size	Hit rate
4	0.0604
8	0.1454
16	0.2622
32	0.3659
64	0.4521
128	0.5395
256	0.7243
512	0.8448

1024	0.9080
2048	0.9532
4096	0.9656
8192	0.9726
16384	0.9806
32768	0.9806



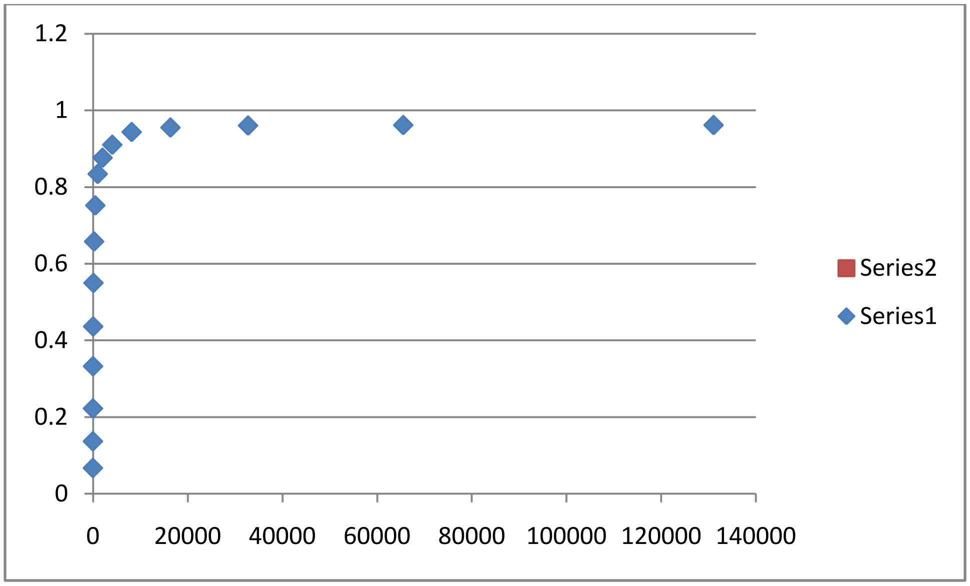
Instruction Cache Size	Hit rate
4	0
8	0
16	0.0034
32	0.0101
64	0.2636
128	0.3687
256	0.4548
512	0.5368
1024	0.7548
2048	0.8620
4096	0.9057
8192	0.9747
16384	0.9864

32768	0.9885
65536	0.9885

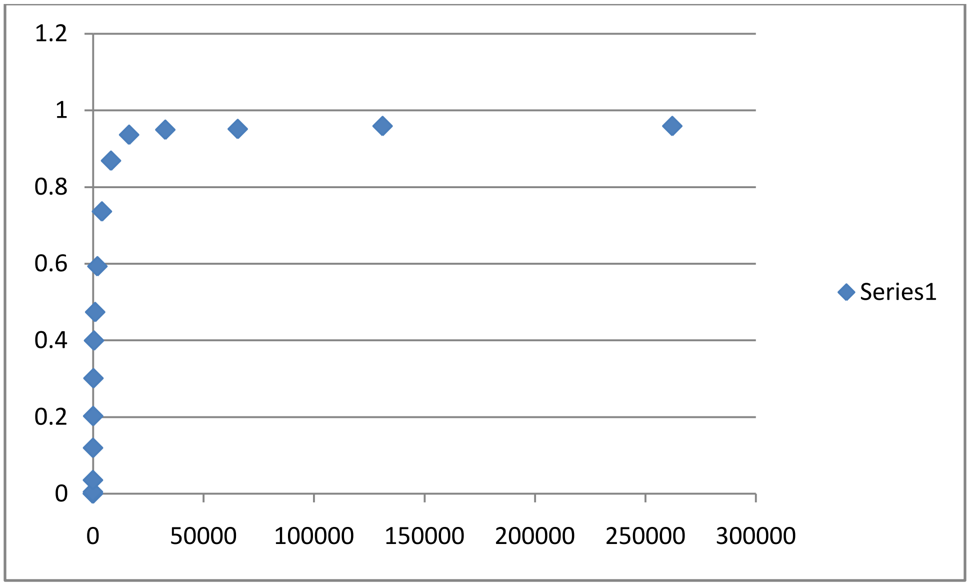


CC Trace File

Data Cache Size	Hit Rate
4	0.0676
8	0.1373
16	0.2229
32	0.3330
64	0.4366
128	0.5502
256	0.6583
512	0.7523
1024	0.8344
2048	0.8766
4096	0.9105
8192	0.9437
16384	0.9555
32768	0.9606
65536	0.9618
131072	0.9618

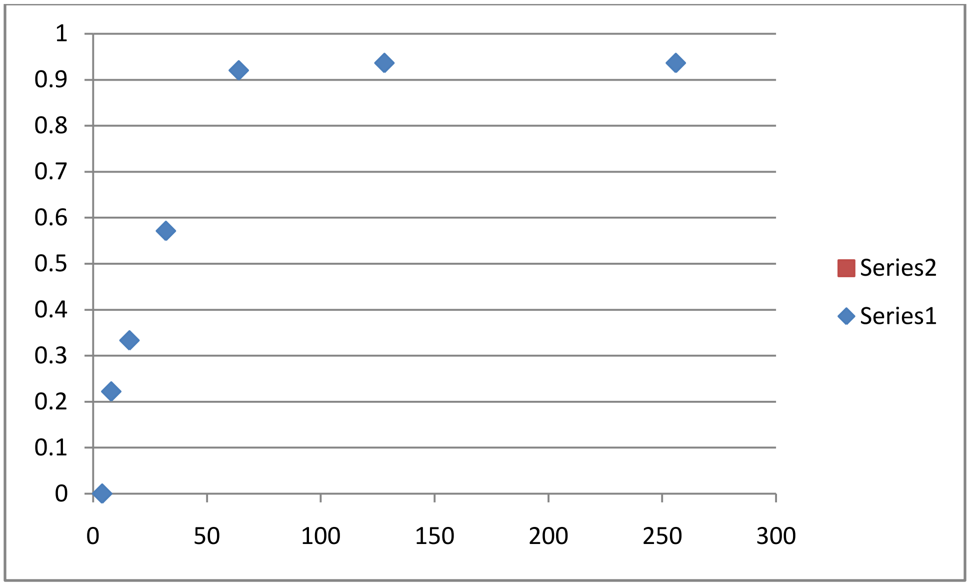


Instruction Cache Size	Hit Rate
4	0
8	0
16	0.0047
32	0.0353
64	0.1194
128	0.2023
256	0.3010
512	0.3993
1024	0.4736
2048	0.5930
4096	0.7360
8192	0.8684
16384	0.9360
32768	0.9492
65536	0.9511
131072	0.9588
262144	0.9588

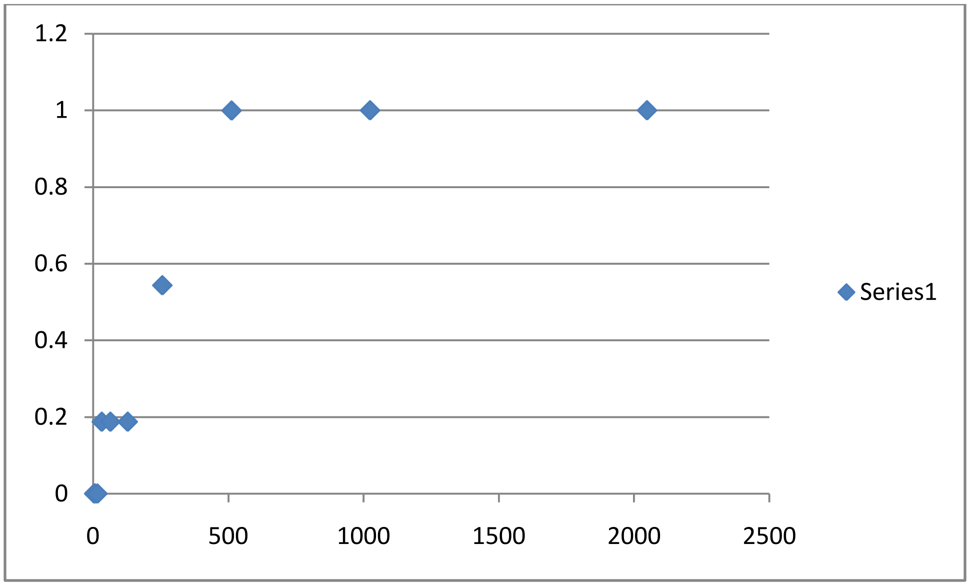


Tex Trace File

Data Cache Size	Hit Rate
4	0.0001
8	0.2222
16	0.3333
32	0.5713
64	0.9204
128	0.9365
256	0.9365



Instruction Cache Size	Hit Rate
4	0
8	0
16	0
32	0.1874
64	0.1874
128	0.1874
256	0.5433
512	0.9991
1024	0.9997
2048	0.9997



From all the above tables we can see that as we increase the cache size initially from 4 bytes the hit rate for all the caches increases. This is because capacity misses occur because blocks are being discarded from cache as cache cannot contain all blocks needed for program execution .This is observed when program working set is much larger than cache capacity. So as cache size increase cache will easily be able to accommodate more blocks and hence hit rate will increase.

But as we keep on increasing cache size, we will observe that after certain size of cache further increase in size will not lead to increase in hit rate. This is because cache is big enough to accommodate all blocks needed for program execution. This particular size of cache is called working set size of cache. From the above table we can easily observe the total instruction working set size and data working set size for each of the three sample traces as follows:-

1. SPICE TRACE
    a. Instruction working set size- 65536
    b. Data working set size- 32768
2. CC TRACE
    a. Instruction working set size-262144
    b. Data working set size-131072

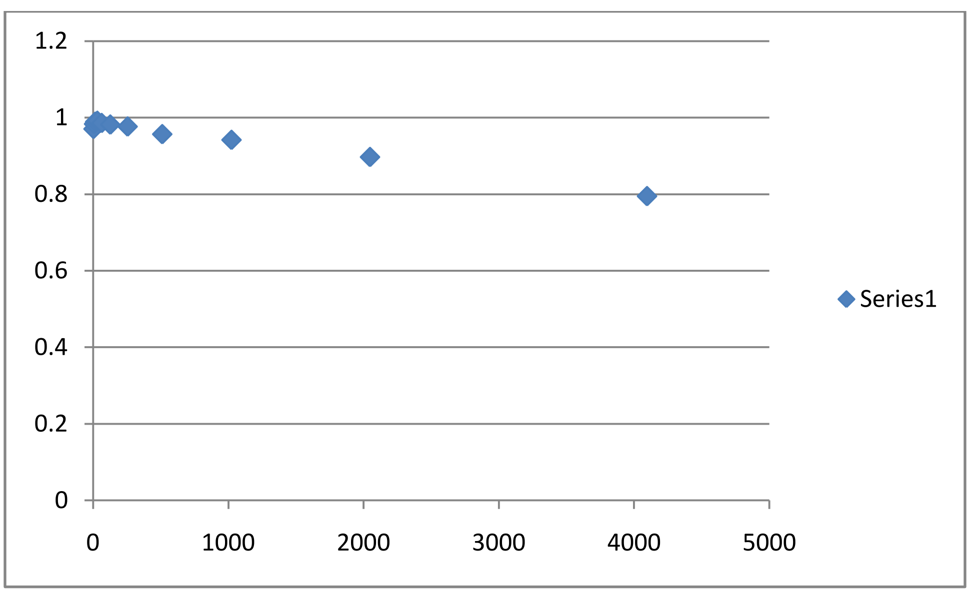
3. TEX TRACE
    a. Instruction working set size-2048
    b. Data working set size-256

BLOCK SIZE

SPICE TRACE

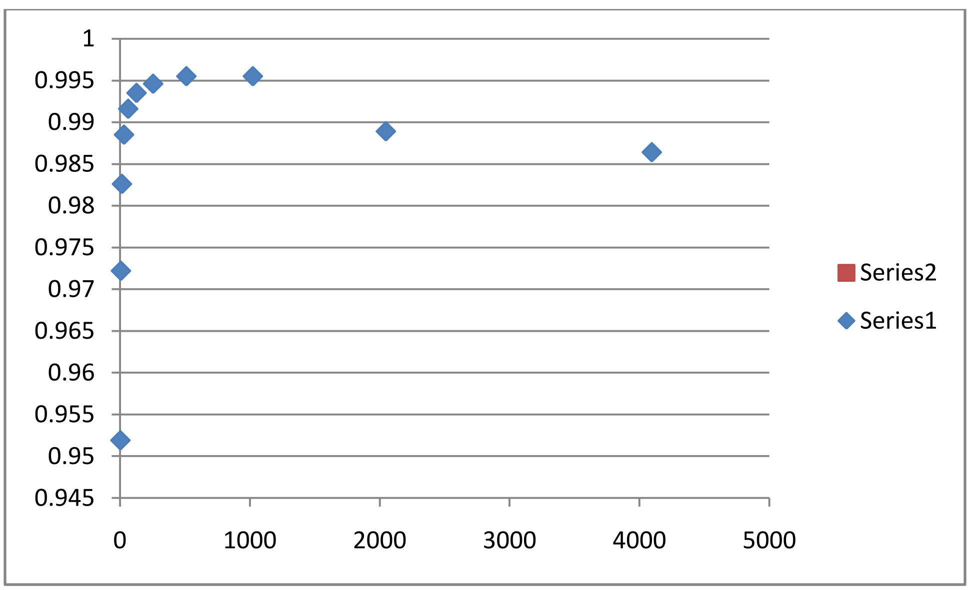
DATA CACHE

BLOCK SIZE	HIT RATE
4	0.9701
8	0.9830
16	0.9886
32	0.9912
64	0.9855
128	0.9814
256	0.9763
512	0.9564
1024	0.9415
2048	0.8968
4096	0.7947



INSTRUCTION CACHE

BLOCK SIZE	HIT RATE
4	0.9519
8	0.9722
16	0.9826
32	0.9885
64	0.9916
128	0.9935
256	0.9946
512	0.9955
1024	0.9955
2048	0.9889
4096	0.9864

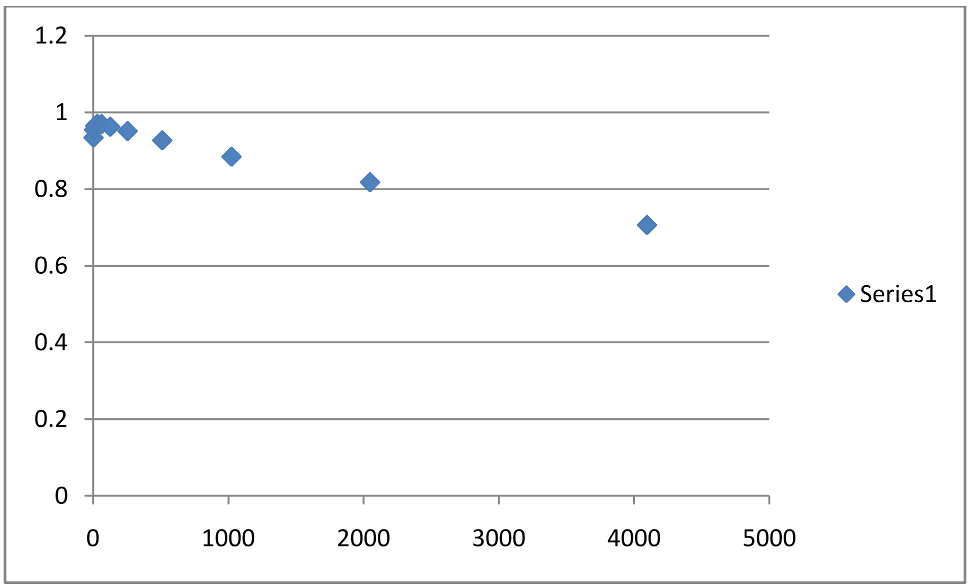


CC TRACE

DATA CACHE

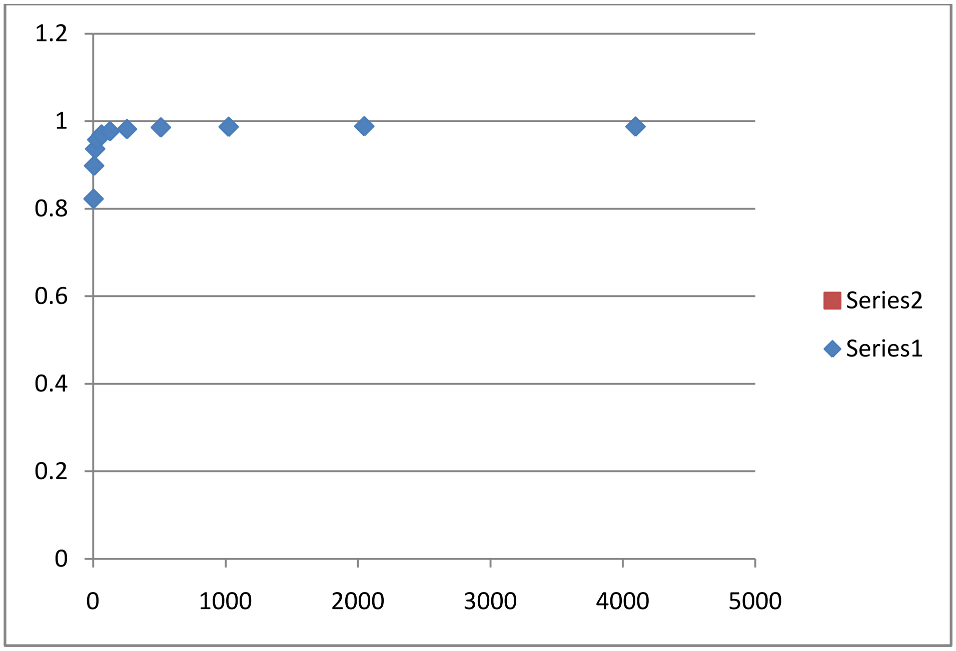
Block Size	HIT RATE
4	0.9339
8	0.9544
16	0.9646

32	0.9695
64	0.9690
128	0.9620
256	0.9508
512	0.9267
1024	0.8842
2048	0.8175
4096	0.7062



INSTRUCTION CACHE

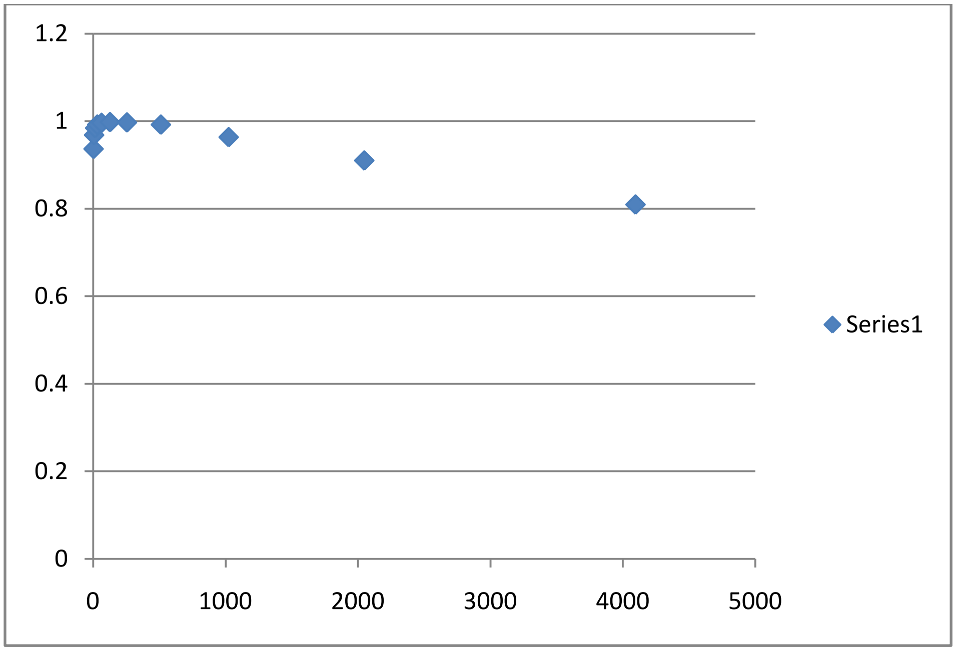
Block Size	Hit Rate
4	0.8225
8	0.8982
16	0.9368
32	0.9578
64	0.9696
128	0.9772
256	0.9820
512	0.9859
1024	0.9872
2048	0.9885
4096	0.9877



TEX TRACE

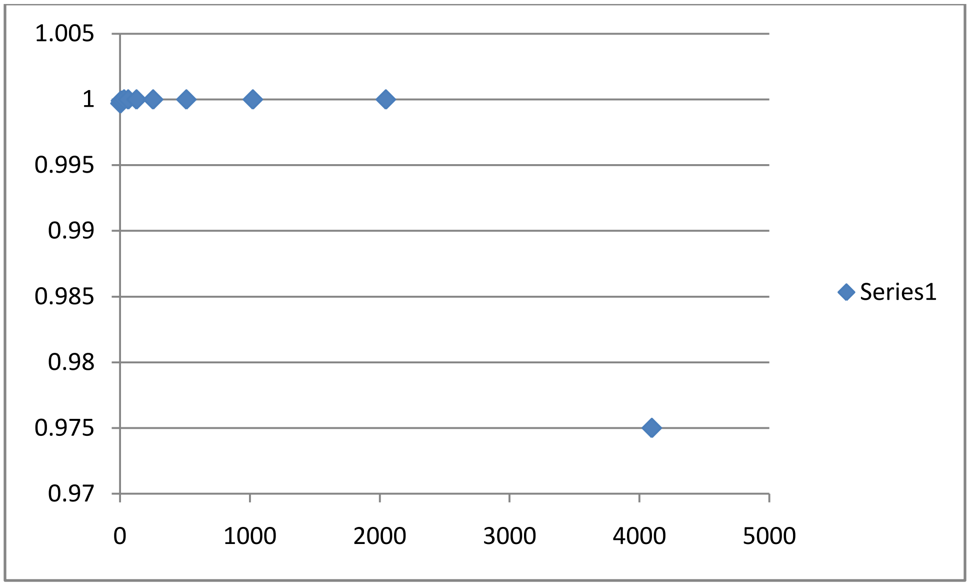
DATA CACHE

Block Size	HIT RATE
4	0.9365
8	0.9682
16	0.9841
32	0.9920
64	0.9959
128	0.9976
256	0.9969
512	0.9919
1024	0.9633
2048	0.9100
4096	0.8095



INSTRUCTION CACHE

Block Size	Hit Rate
4	0.9997
8	0.9999
16	0.9999
32	1.0000
64	1.0000
128	1.0000
256	1.0000
512	1.0000
1024	1.0000
2048	1.0000
4096	0.9750



Hit Rate initially increases initially as block size increases. This can be accounted for by two factors. Larger block size will reduce the initial number of compulsory misses. Moreover in most of computer programs the theory of spatial locality holds. Spatial locality is concept that states that likelihood of referencing a resource is higher if a resource near it was just referenced. The spatial locality is very important if our program has loops or if we are using array. Larger blocks take advantage of spatial locality. But after a particular block size the hit rate starts decreasing. This is because the larger the block size, the less the number of entries in the cache, and the more the competition between program data for these entries.

The optimal block size for each of the cache set is value at which hit rate is maximum. From the above table we can easily observe the optimal instruction block size and data block size for each of the three sample traces as follows:-

1. SPICE TRACE
    a. Instruction block size- 512/1024
    b. Data block size- 32
2. CC TRACE
    a. Instruction block size-2048
    b. Data block size-32

3. TEX TRACE
    a. Instruction working set size-(32-2048)
    b. Data working set size-128

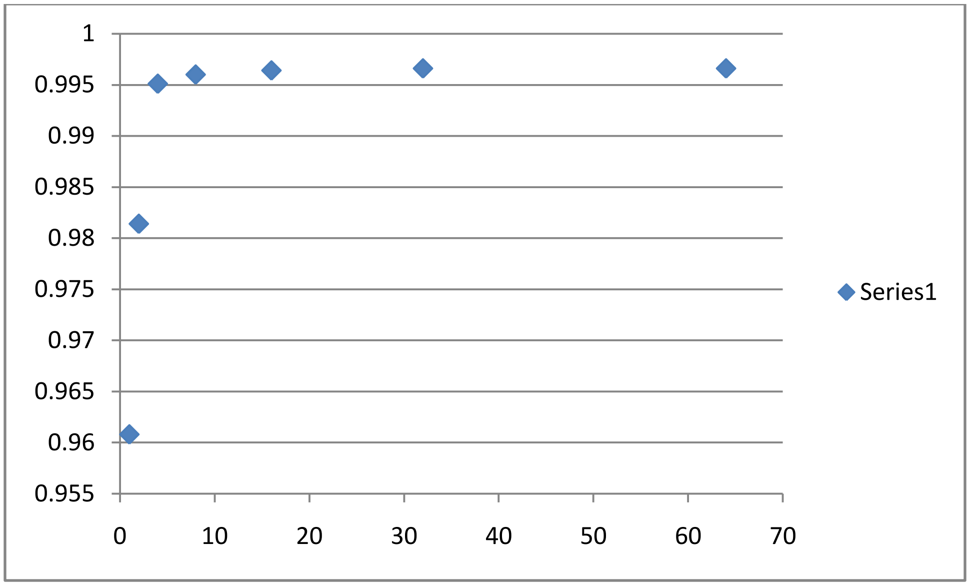
We can see that optimal block size for instruction references is higher than that for data references. Hence the instruction references have higher spatial locality than data references.

ASSOCIATIVITY

SPICE

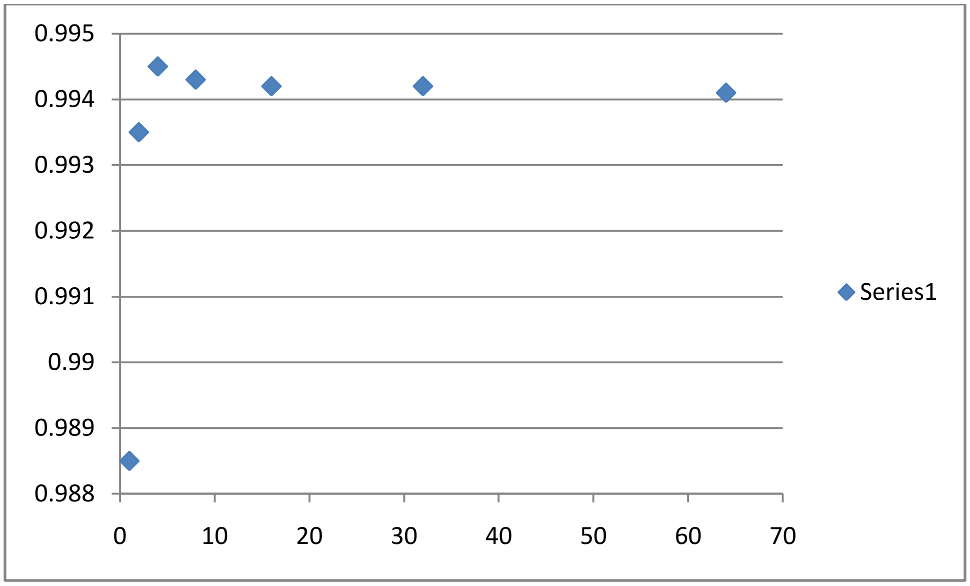
DS

ASSOCIATIVITY	HIT RATE
1	0.9608
2	0.9814
4	0.9951
8	0.9960
16	0.9964
32	0.9966
64	0.9966



IS

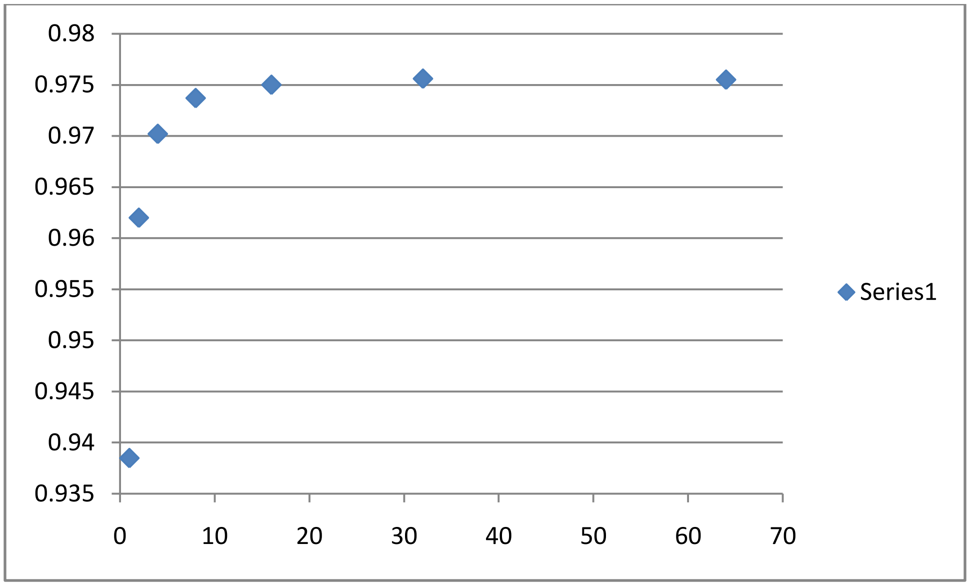
ASSOCIATIVITY	Hit Rate
1	0.9885
2	0.9935
4	0.9945
8	0.9943
16	0.9942
32	0.9942
64	0.9941



CC

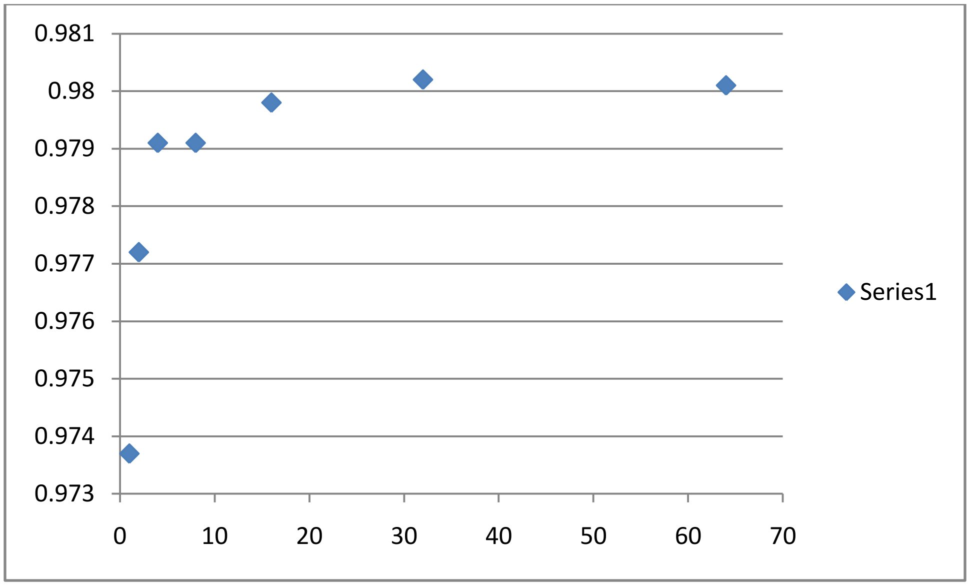
DS

ASSOCIATIVITY	HIT RATE
1	0.9385
2	0.9620
4	0.9702
8	0.9737
16	0.9750
32	0.9756
64	0.9755



IS

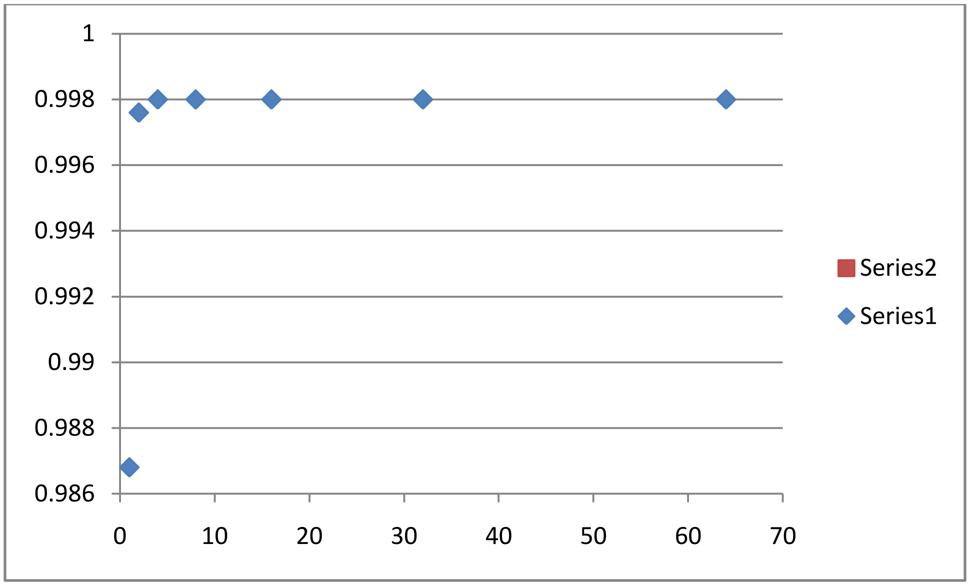
ASSOCIATIVE	Hit Rate
1	0.9737
2	0.9772
4	0.9791
8	0.9791
16	0.9798
32	0.9802
64	0.9801



TEX

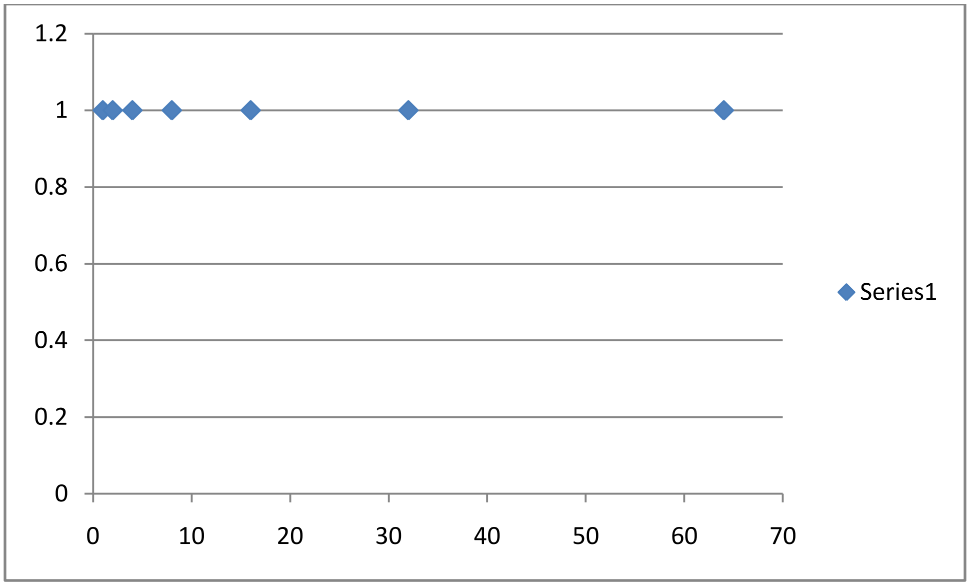
DS

ASSOCIATIVE	Hit Rate
1	0.9868
2	0.9976
4	0.9980
8	0.9980
16	0.9980
32	0.9980
64	0.9980



IS

ASSOCIATIVE	Hit Rate
1	1.0000
2	1.0000
4	1.0000
8	1.0000
16	1.0000
32	1.0000
64	1.0000



If the block-placement strategy is set associative or direct mapped, conflict misses (in addition to compulsory and capacity misses) will occur because a block can be discarded and later retrieved if too many blocks map to its set. These are also called collision misses or interference misses. As we increase the associativity hit rate increases as conflict misses decreases. However after a certain value of associativity hit rate decreases due to increase in capacity misses since larger the associativity, lesser the number of entries in the cache, and the more the competition between program data for these entries. Impact of associativity if there is no spatial locality. Hence its impact on data set is more which means that we are using the data that are located far aways in data cache as compared to instruction cache.

MEMORY BANDWITH

COMPARISON OF WRITE THROUGH AND WRITE BACK

SPICE TRACE

WRITE THROUGH

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY Traffic
64	8192	2	151104	66538	217642
128	8192	2	280768	66538	347306
64	8192	4	107296	66538	173834
64	16384	2	57008	66538	123546
128	16384	2	93632	66538	160170

WRITE BACK

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY TRAFFIC
64	8192	2	151104	13624	164728
128	8192	2	280768	32287	313055
64	8192	4	107296	9219	116515
64	16384	2	57008	8252	65260
128	16384	2	93632	9880	103512

CC TRACE

WRITE THROUGH

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY Traffic
64	8192	2	474256	83030	557286
128	8192	2	824192	83030	907222
64	8192	4	423104	83030	506134
64	16384	2	252096	83030	335126
128	16384	2	440768	83030	523798

WRITE BACK

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY TRAFFIC
64	8192	2	474256	39426	513682
128	8192	2	824192	78872	903064
64	8192	4	423104	32102	455206
64	16384	2	252096	24048	276144
128	16384	2	440768	40538	481306

TEX TRACE

WRITE THROUGH

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY Traffic
64	8192	2	2880	104513	107393
128	8192	2	3488	104513	108001
64	8192	4	3312	104513	107825
64	16384	2	2464	104513	106977
128	16384	2	2656	104513	107169

WRITE BACK

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY TRAFFIC
64	8192	2	2880	29903	32783
128	8192	2	3488	29935	33423
64	8192	4	3312	29903	33215
64	16384	2	2464	29909	32373
128	16384	2	2656	29957	32613

We can see that write back cache has less memory traffic than write through cache. This is mainly because in write back cache the number of copy backs is lesser than in write through. This is observed since in write back during memory write instruction the information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.

The answer to this question will flip when there are lot of misses and hence the copy backs will be very large in write back cache since the dirty bit will require the memory to be constantly updated.

COMPARISON OF WRITE ALLOCATE AND NO WRITE ALLOCATE

SPICE.TRACE

WRITE ALLOCATE

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY TRAFFIC
64	8192	2	156000	18880	174880
128	8192	2	291584	36256	327840
64	8192	4	110400	7296	117696
64	16384	2	59856	6208	66064
128	16384	2	99008	9088	108096

NO WRITE ALLOCATE

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY TRAFFIC
64	8192	2	151104	13624	164728
128	8192	2	280768	32287	313055
64	8192	4	107296	9219	116515
64	16384	2	57008	8252	65260
128	16384	2	93632	9880	103512

CC TRACE

WRITE ALLOCATE

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY TRAFFIC
64	8192	2	488976	41840	530816
128	8192	2	848832	91744	940576
64	8192	4	435408	33216	468624
64	16384	2	261248	23584	284832
128	16384	2	455840	46048	501888

NO WRITE ALLOCATE

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY TRAFFIC
64	8192	2	474256	39426	513682
128	8192	2	824192	78872	903064
64	8192	4	423104	32102	455206
64	16384	2	252096	24048	276144
128	16384	2	440768	40538	481306

TEX TRACE

WRITE ALLOCATE

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY TRAFFIC
64	8192	2	15696	7664	23360
128	8192	2	18528	8608	27136
64	8192	4	15408	7568	22976
64	16384	2	11696	7616	19312
128	16384	2	13696	8256	21952

NO WRITE ALLOCATE

BLOCK SIZE	CACHE SIZE	ASSOCIATIVITY	DEMAND FETCH	COPIES BACK	MEMORY TRAFFIC
64	8192	2	2880	29903	32783
128	8192	2	3488	29935	33423
64	8192	4	3312	29903	33215
64	16384	2	2464	29909	32373
128	16384	2	2656	29957	32613

From the table we observe that we cannot clearly say that either one of write allocate or not write allocate is better since write not allocate is better for spice and cc trace file and write allocate is better for tex trace file. However, theoretically we expect write allocate to perform better with write back cache hoping that subsequent writes to that block will be captured by the cache.

The answer will be flipped if it is a write through cache since subsequent writes to that block will still have to go to memory.