Assignment 2 - Cache Evaluation

Abstract

In this assignment, we are aiming to achieve a simulator to evaluate the effects of different structures of cache on a system's performance. Various combinations of cache size, associativity and words per block affect are considered. Other perspectives such as hit time, RAS latency, CAS latency, DRAM structure are chosen in sense of more realistic simulation of the real world situation.

Simulation specification

Cache size: 8kB, 32kB, 512kB.

Associativity: direct mapping (1 way), 4 way set associative.

Words per block: 2 or 8

Hit time is chosen according to various cache size.

Miss time: 2 clock cycles plus DRAM access latency.

Combinations as follow:

Simulation No.	Associativity	Cache size (kB)	Words per block	Hit time
1	Direct mapping	8	2	1
2	Direct mapping	32	2	2
3	Direct mapping	512	2	3
4	Direct mapping	8	8	1
5	Direct mapping	32	8	2
6	Direct mapping	512	8	3
7	4 way set associative	8	2	1
8	4 way set associative	32	2	2
9	4 way set associative	512	2	3
10	4 way set associative	8	8	1
11	4 way set associative	32	8	2
12	4 way set associative	512	8	3

Latency source

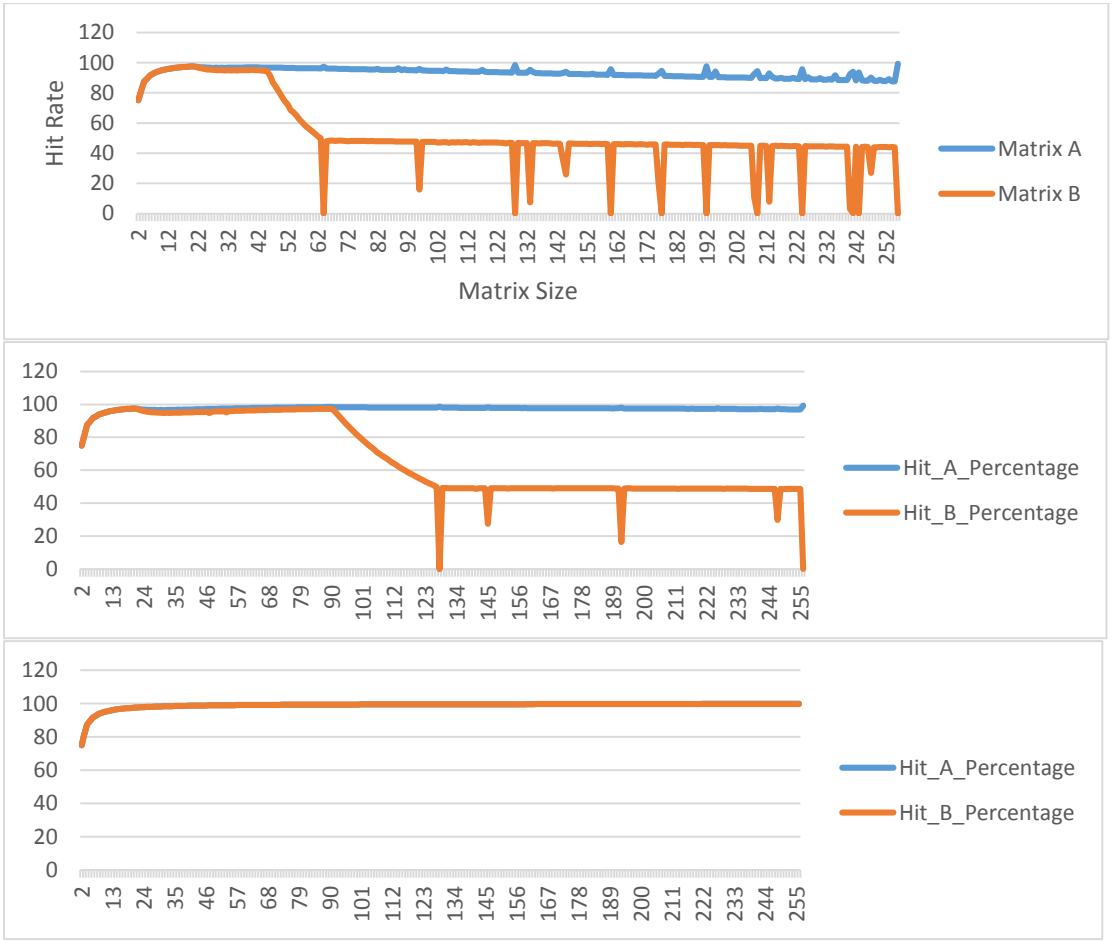
Comparing to short latency (1-3 clock cycle) by accessing data in cache, the main source of latency is access data in DRAM. There are two type of DRAM access latency, RAS and CAS. RAS latency as 72 clock cycles is much higher than a CAS latency of 24 clock cycles. The occurrence of RAS is less than CAS, because of temporal and/or spatial locality. DRAM is specified to have a 64bit wide bus, 2 words per column, and 8 columns per row. In this simulation cache write through is adapted.

Here is considering situation when cache miss, needs access to main memory. For a reading activity, if it is accessing data on same raw as previous activity, it will be a RAS only. Otherwise, if it is access data on different row, clock delay will be both one RAS and one CAS. For a writing activity, because of cache write through, two CAS miss if access same row of previous activity, otherwise two CAS plus one RAS for accessing different row.

Simulation result analysis

Systematic change in hit rate

Those are the hit rate vs matrix size plot for direct mapped with 2 word per block, where cache size varies from 8 to 32 to 512 kB.



When the matrix size is small, the hit rate remains high with small variation, when matrix size increase to a certain value, the hit rate drop to a relatively low value over a transient period. Then the hit rate remains almost constant for the remaining matrix sizes. The size of the matrix when the hit rate begins to drop increases as the cache size increases, which make the overall hit rate higher for higher cache size. Also for some matrix size, after the hit rate begins to drop, the hit rate may have a deep spike down to very low value (can be zero in direct map). Such pattern happens to all 8kB and 32kB cache structures. For the 512kB structures, the hit rate remains high with minimal variable. This is also why the hit rate is similar for all 512kB structures.

The sudden drop of hit rate for matrix B happens for matrix size 64, 96, 128 and other sizes. Here will take matrix size 64 for direct mapped 2 word per block for example. For matrix size of 64, the address

of matrix B will increment by 256, which is four times the matrix size. For each matrix A row time matrix B column, there are 64 operations. During those 64 operations, if the cache is replaced during those operations, there will be cache miss for the following matrix B read access. If the address is digital 256 with tag of 0 at the first read, after 32 read the address will be digital 8448 with tag of 1. But if we look at the lower 13 bits (top19 bit is tag bits), it is the same for address 256 and address 8448, hence index 256 will be replaced, causing zero hit rate for matrix B. The sudden drop of hit rate also happens to most of the other cache structures, but such drops are not necessarily down to zero.

Again taking the direct mapped 2word per block with 8kB cache size as example, the gradual drop of hit rate happens at matrix size of 45. When the matrix size be at or below 45 the cache is large enough to hold the entire matrix. Once the matrix size grows to 64, resulting address ranging from 0 to 8464, the cache will only hold part of the matrix elements. As matrix size further grows, such situation getting worse, resulting in smaller hit rate. In the other hand, with larger cache size, the maximum matrix sizes that can entirely fit into cache becomes larger, resulting in higher overall hit rate.

Changes in features caused by change in cache structure

There are three factors affecting hit rate: cache size, associativity and words per block.

Effect by Cache size

These are the improvement of hit rate by changing cache size. All improvement calculated based on hit rate of 8kB cache size with same situation.

Direct mapping, 2 words per block			Direct mapping, 8 words per block		
	Improvement			Improvement	
Cache Size	Hit Rate	Time	Cache Size	Hit Rate	Time
8			8		
32	1.15	0.81	32	1.37	0.49
512	1.80	0.11	512	1.65	0.12
4 ways, 2 words per block			4 ways, 8 words per block		
	Improvement			Improvement	
Cache Size	Hit Rate	Time	Cache Size	Hit Rate	Time
8			8		
32	1.11	0.85	32	1.21	0.50
512	1.71	0.13	512	1.36	0.17

The main trend is obviously larger cache size will have higher hit rate. More data are store in the cache, there are more change of cache hit. Such improvement in hit rate will decrease yet still positive when either block size or associativity increases. If we have larger block size, spatial locality boost the hit rate even cache size is small. By having 4 ways associativity, ping pong effect due to conflict misses is solved, hence boost the hit rate even cache size is small.

If we look at the time, the higher the hit rate, the less time spend to calculate the result. This is because no mater it is a RAS or CAS miss, it consume much more clock cycles to access data from main memory instead of from the cache.

Effect by block size

These are the improvement of hit rate by changing block size. All improvement calculated based on hit rate of 2 word per block with same situation.

Direct mapping, 8 word per block			Direct mapping, 8 word per block		
	Improvement		Average Improvement		
Cache Size	Hit Rate	Time	Hit Rate	Time	
8	1.07	0.91	1.12	0.81	
32	1.27	0.55			
512	1.00	0.97			
4 ways, 8 word per block			4 ways, 8 word per block		
	Improvement		Average Improvement		
Cache Size	Hit Rate	Time	Hit Rate	Time	
8	1.21	0.72	1.18	0.698730671	
32	1.33	0.42			
512	1.00	0.95			

By having more word per block, it load more data next to requested data, hence takes the advantage of spatial locality. The theory, when the block size becomes a significant proportion of cache size, it can drag down the hit rate. However, the block size used in simulation is small even for 8kB cache size. Result also shows improvement by increase block size is more significant when cache size is large and when set associative is used, this is because the data range within each block becomes wider.

If we look at the time, the higher the hit rate, the less time spend to calculate the result. This is because no mater it is a RAS or CAS miss, it consume much more clock cycles to access data from main memory instead of from the cache.

Effect by associative

These are the improvement of hit rate by changing block size. All improvement calculated based on hit rate of direct mapping with same situation.

4 ways, 2 word per block				4 ways, 2 word per block	
	Improvement			Average Improvement	
Cache Size	Hit Rate	Time		Hit Rate	Time
8	1.05	0.92		1.02	0.97
32	1.01	0.97			
512	1.00	1.02			
4 ways, 8 word per block				4 ways, 8 word per block	
	Improvement			Average Improvement	
Cache Size	Hit Rate	Time		Hit Rate	Time
8	1.18	0.73		1.08	0.829452477
32	1.05	0.75			
512	1.00	1.01			

The overall improvement is about 5% of hit rate comparing to direct mapping. By set associative, for each index, there is now more than one tag available in the cache. Each index can hold the data from any address inside the mapped block. With 4-way set associative, the problems (collision and loop) with direct mapping are solved. Hence improves the system performance.

As result above also shows the improvement by having higher associative is more significant when cache size is small. When cache size is small, there is more change that data in cache is replace due to cache miss. In direct mapping, all data in the index will be replace, but fore set associative, only one of them will be replace, hence remaining data can be hit in the future, increasing system performance. When cache size is larger, cache will have more lines, such improvement will be less than that for smaller cache size.

If we look at the time, the higher the hit rate, the less time spend to calculate the result. This is because no mater it is a RAS or CAS miss, it consume much more clock cycles to access data from main memory instead of from the cache.

Best combination

Average hit rate and time span for all cache size considered.

Ways	Words_Per_Block	Hit Rate	Time
1	2	79.46	4.96E+10
1	8	85.14	37705128208
4	2	80.74	46941149202
4	8	89.48	28544502218

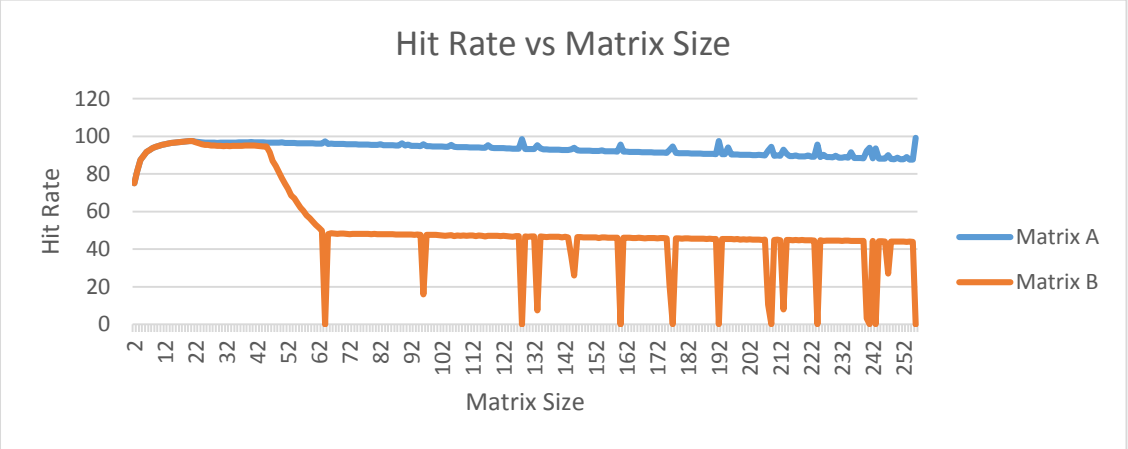
From the comparison above, it clearly shows 4 way associative with 8 words per block have both the highest hit rate and lowest time span. Hence such cache configuration works best for general purpose.

For certain matrix size 64, 128, 192 and so on, the hit rate will drop to a very low level. Increase the block size will not solve the issue because it does not increase the spatial much. However, it can be improve by increase the associative of the case. As always, increase the cache size might be the simplest solution is some situation. The best cache configuration for those matrix sizes are 4 way associative with 2words per block.

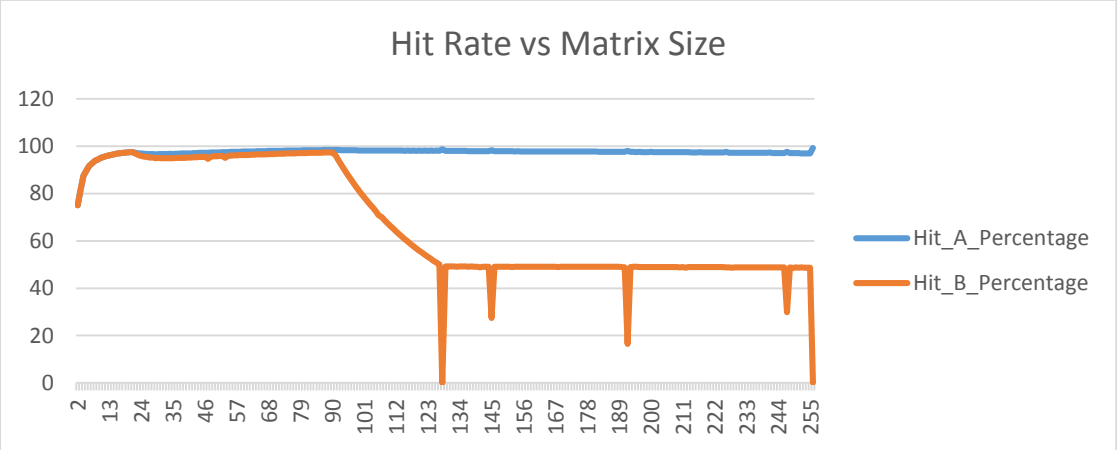
Appendix

All hit rate vs matrix size

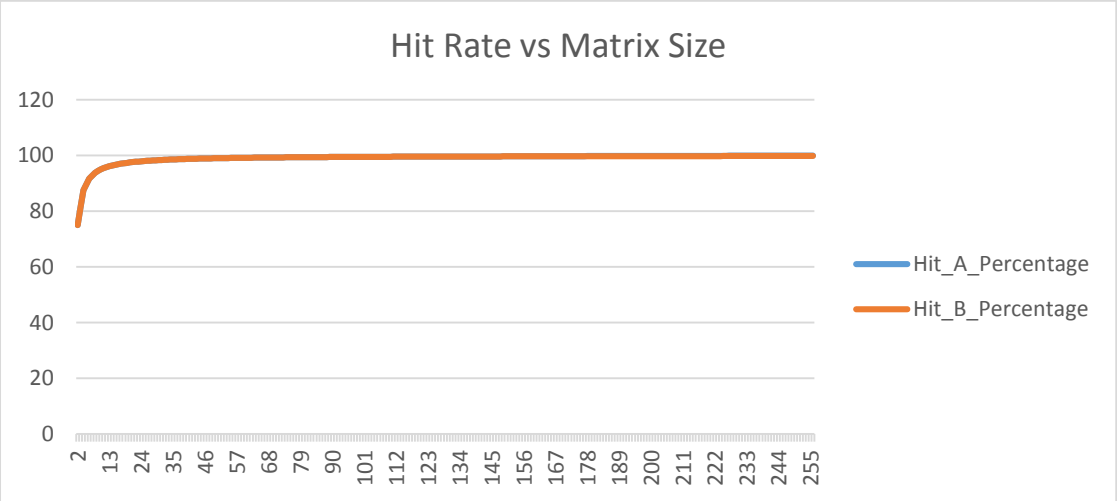
Ways	Cache Size	Words_Per_Block	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	8	2	1	1024	10	19	1



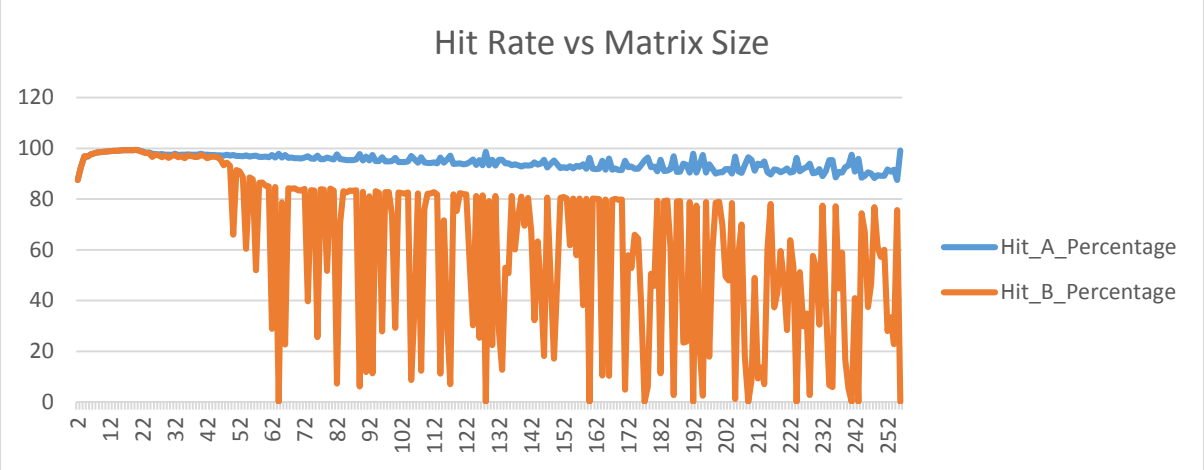
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	32	2	2	4096	12	17	2



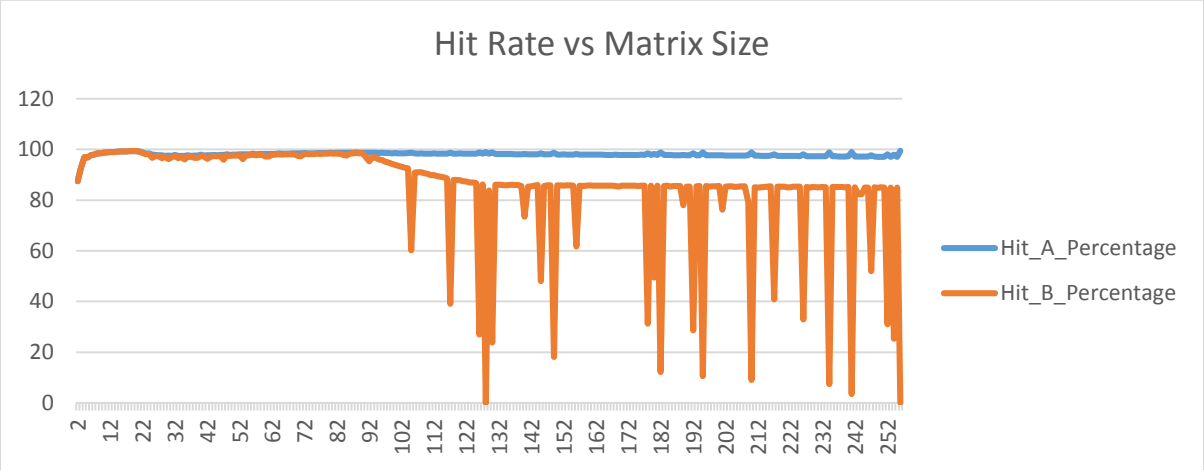
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	512	2	3	65536	16	13	3



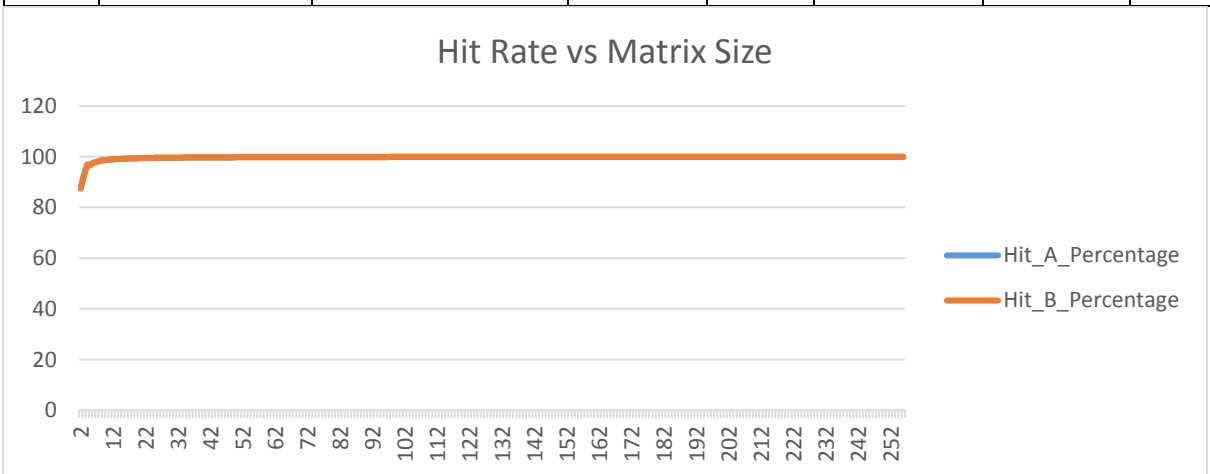
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	8	8	1	256	8	19	1



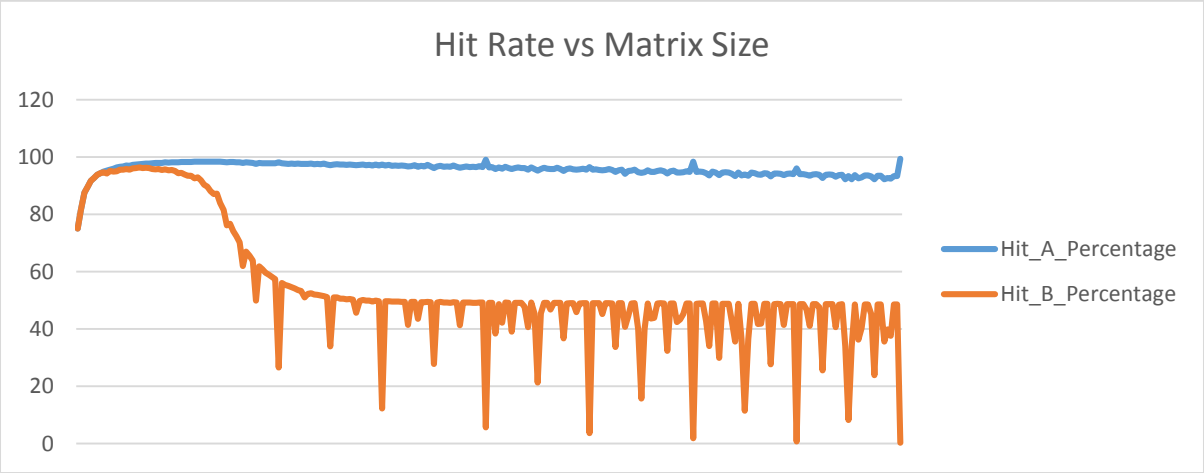
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	32	8	2	1024	10	17	2



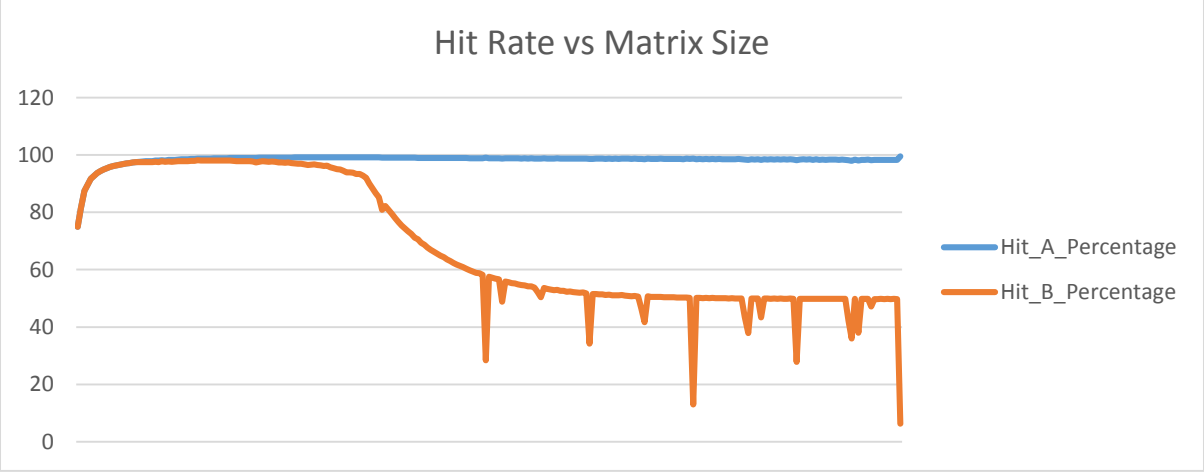
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	512	8	3	16384	14	13	3



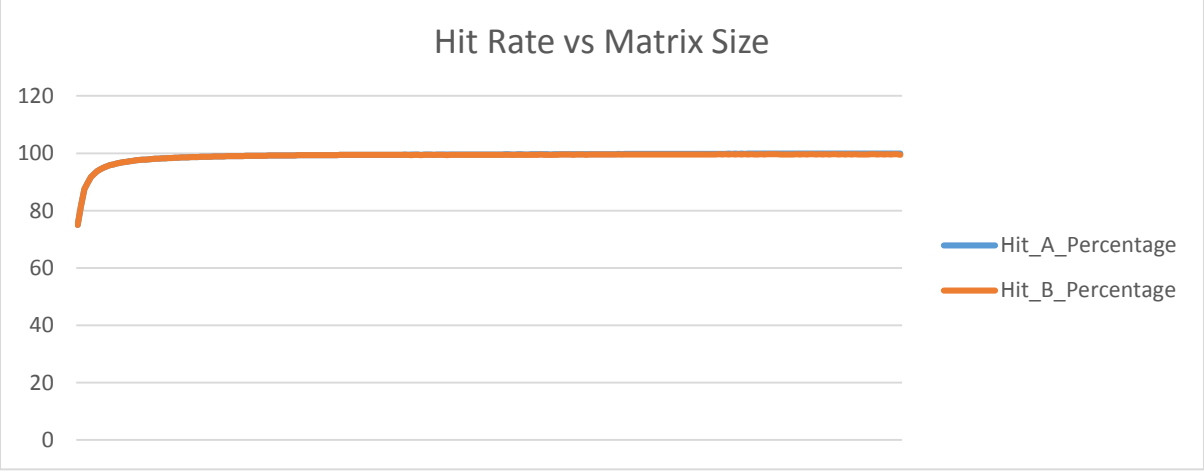
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	8	2	1	256	8	21	1



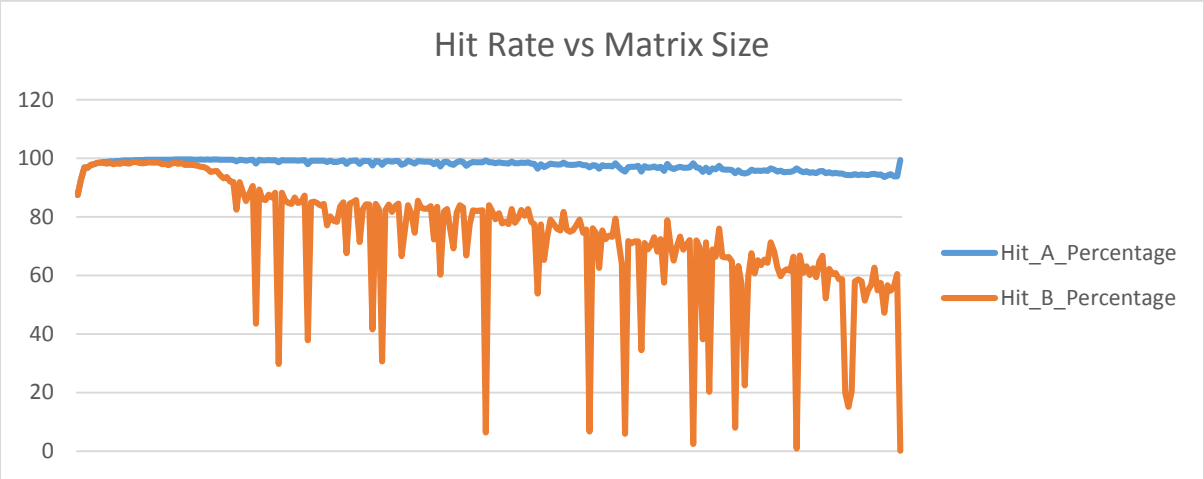
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	32	2	2	1024	10	19	2



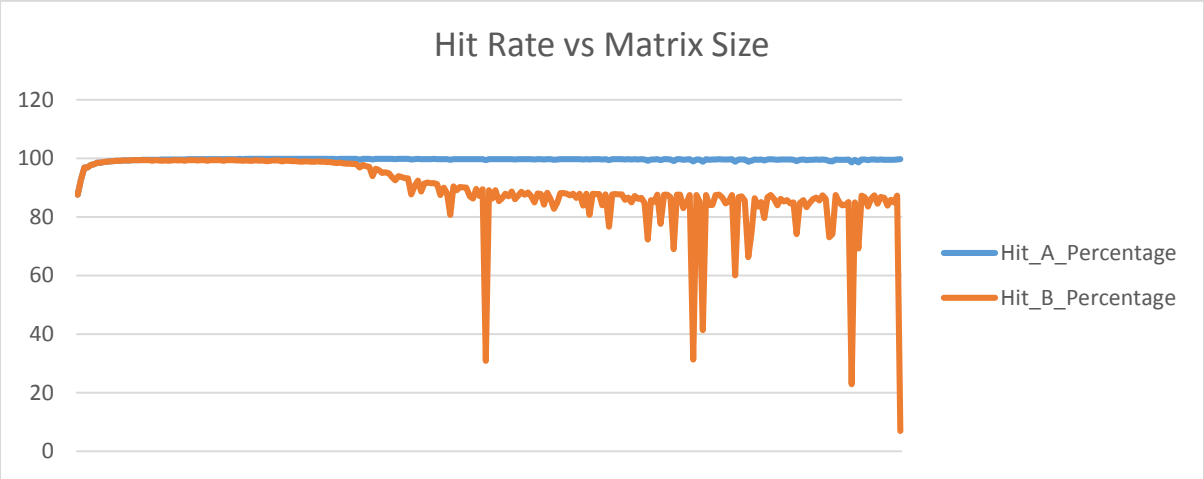
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	512	2	3	16384	14	15	3



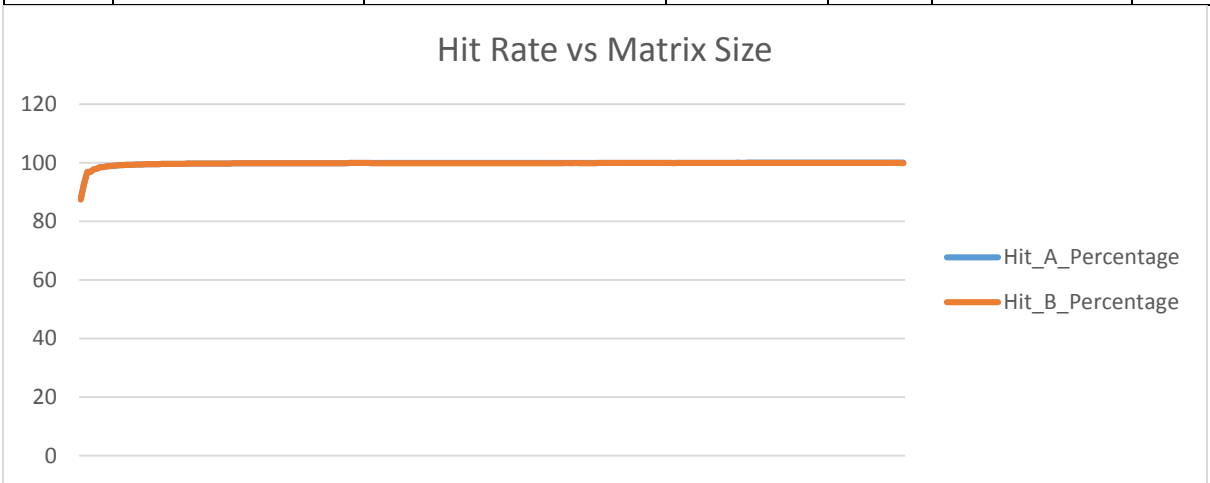
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	8	8	1	64	6	21	1



Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	32	8	2	256	8	19	2

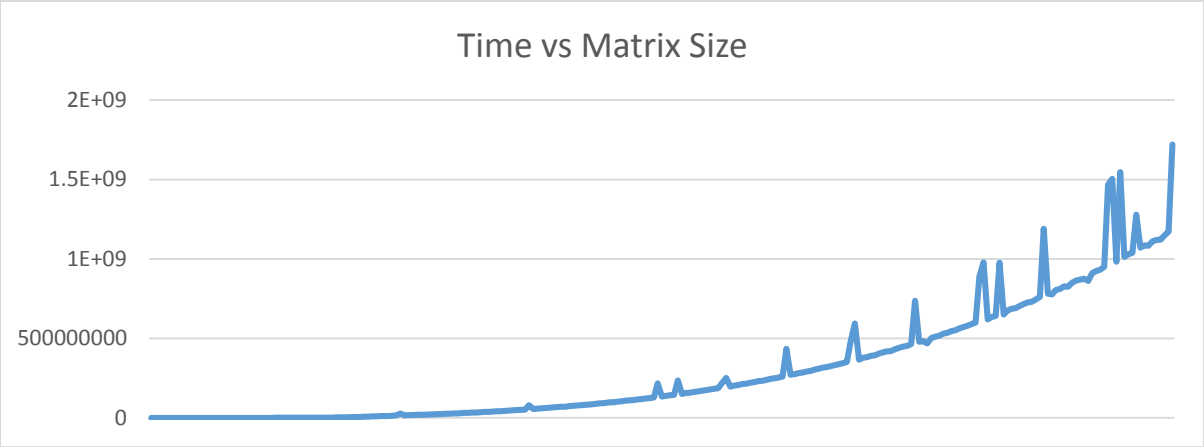


Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size
4	512	8	3	4096	12	15

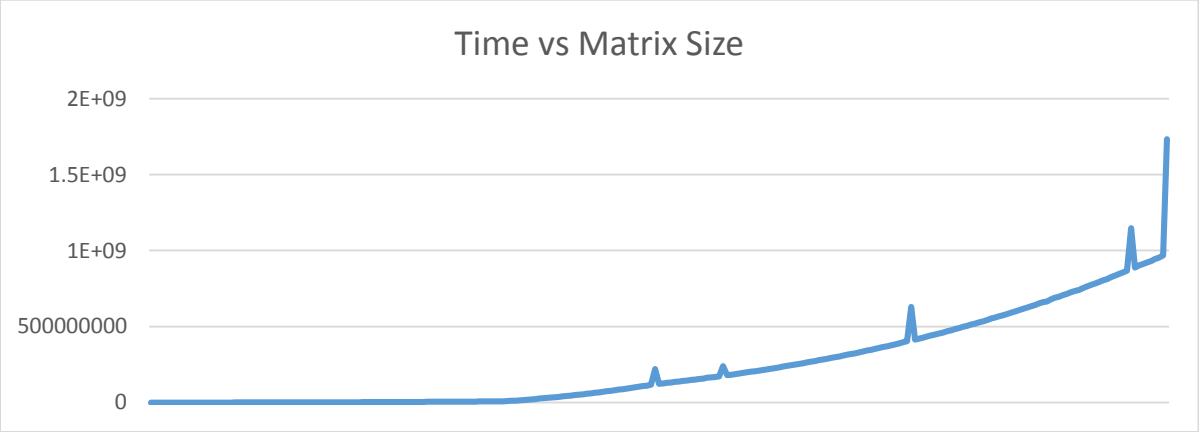


All time vs matrix size

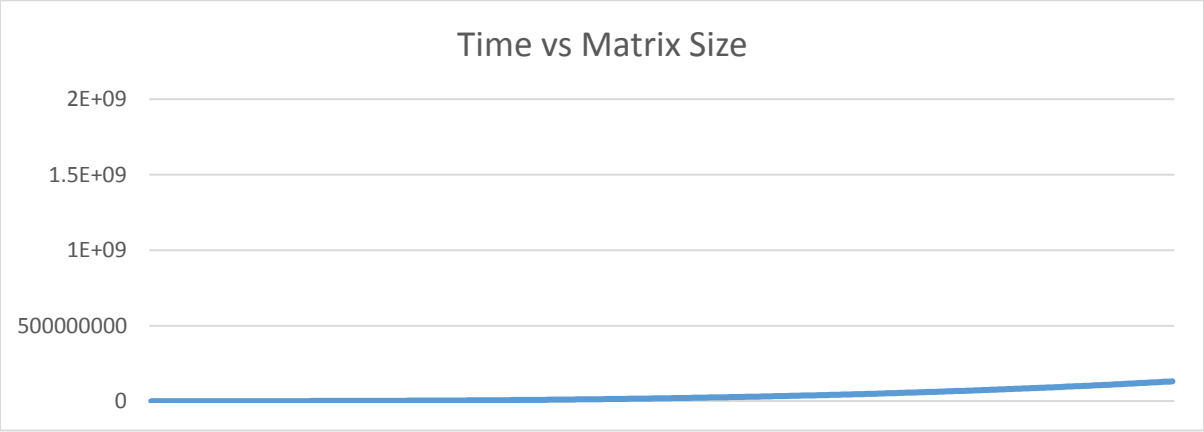
Ways	Cache Size	Words_Per_Block	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	8	2	1	1024	10	19	1



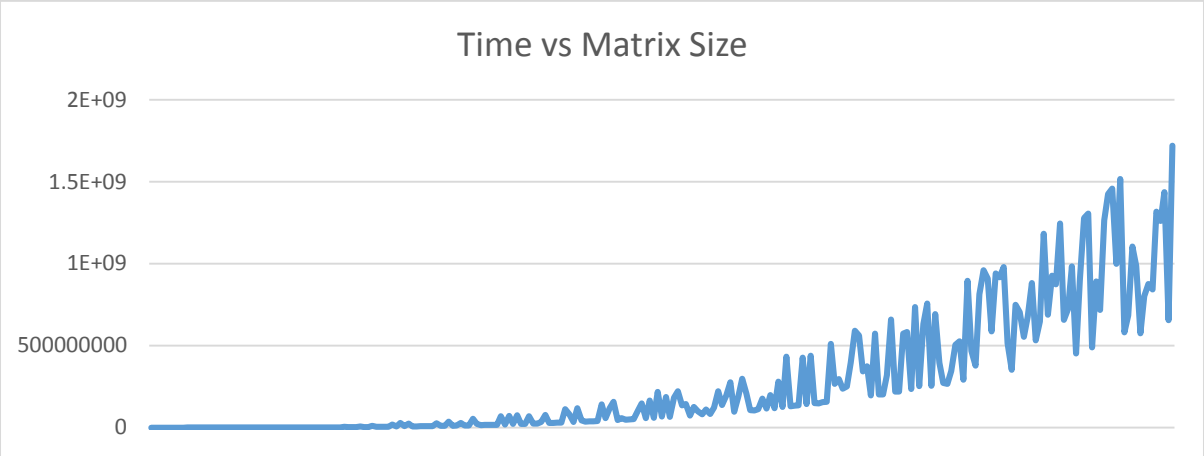
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	32	2	2	4096	12	17	2



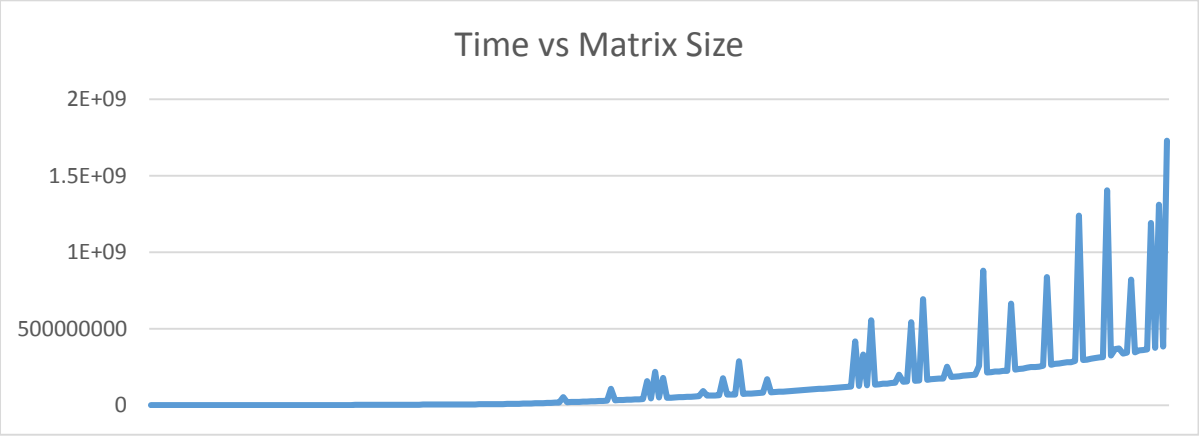
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	512	2	3	65536	16	13	3



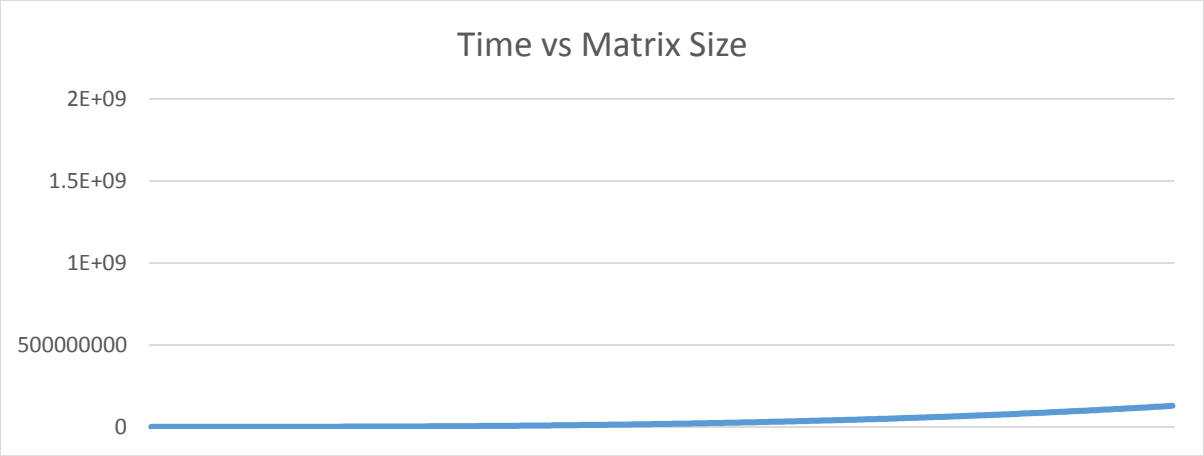
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	8	8	1	256	8	19	1



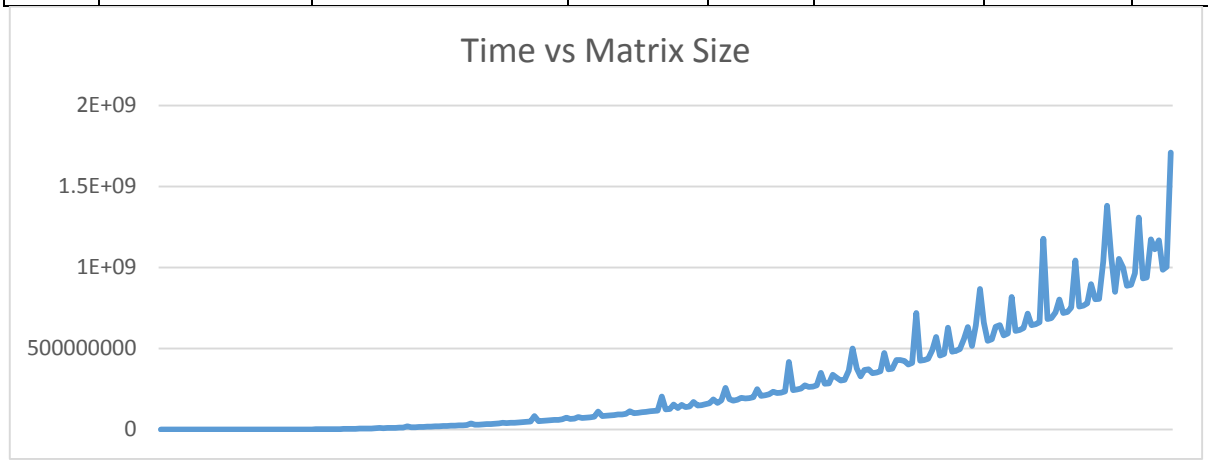
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	32	8	2	1024	10	17	2



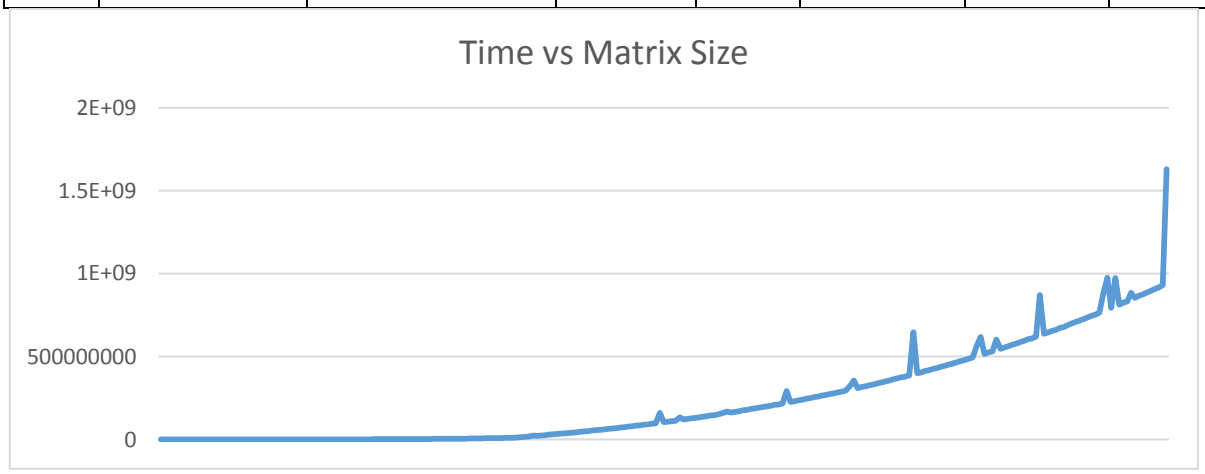
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
1	512	8	3	16384	14	13	3



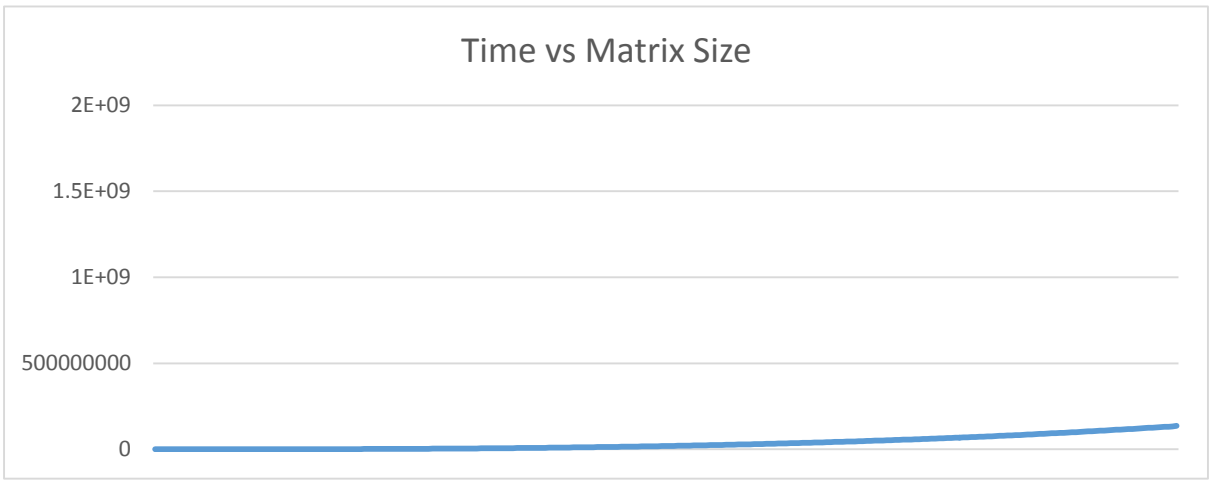
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	8	2	1	256	8	21	1



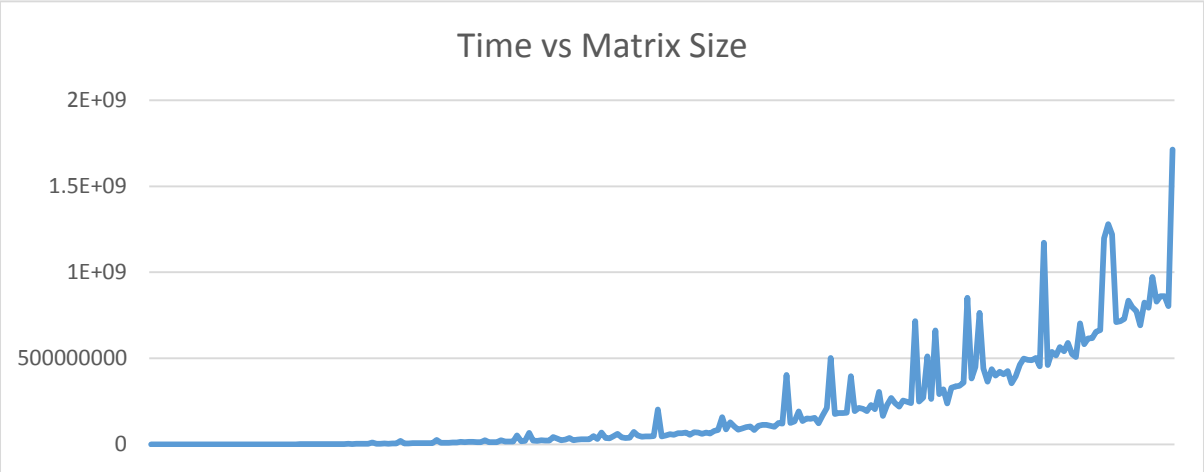
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	32	2	2	1024	10	19	2



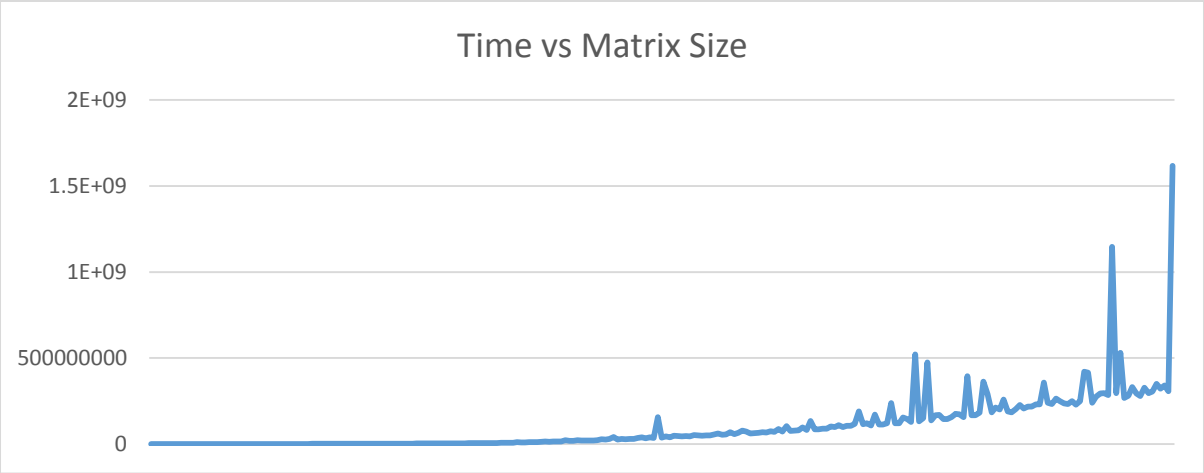
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	512	2	3	16384	14	15	3



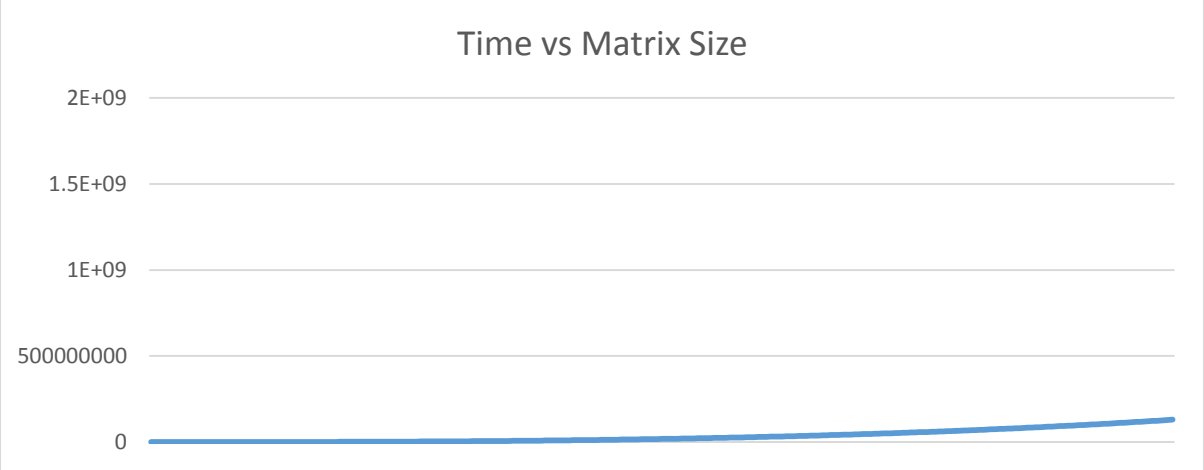
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	8	8	1	64	6	21	1



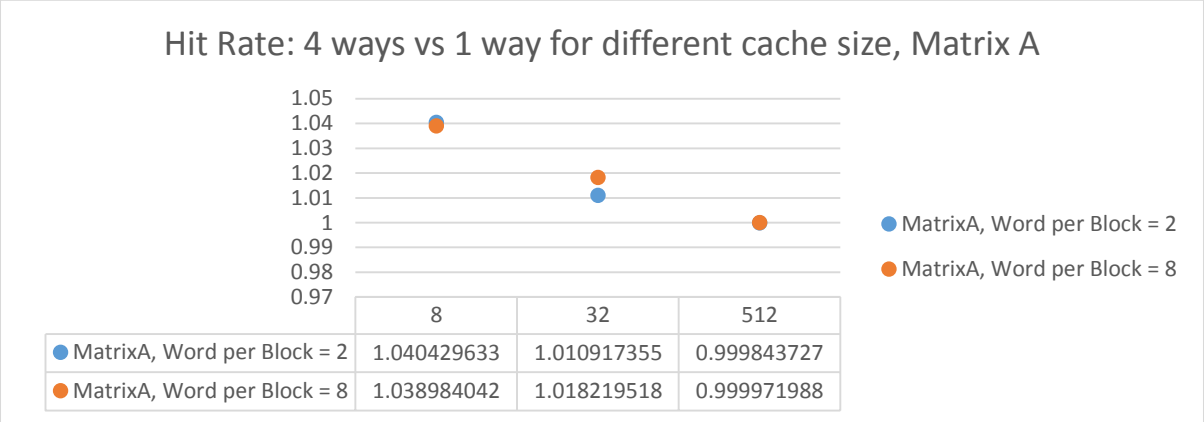
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size	Hit_Time
4	32	8	2	256	8	19	2



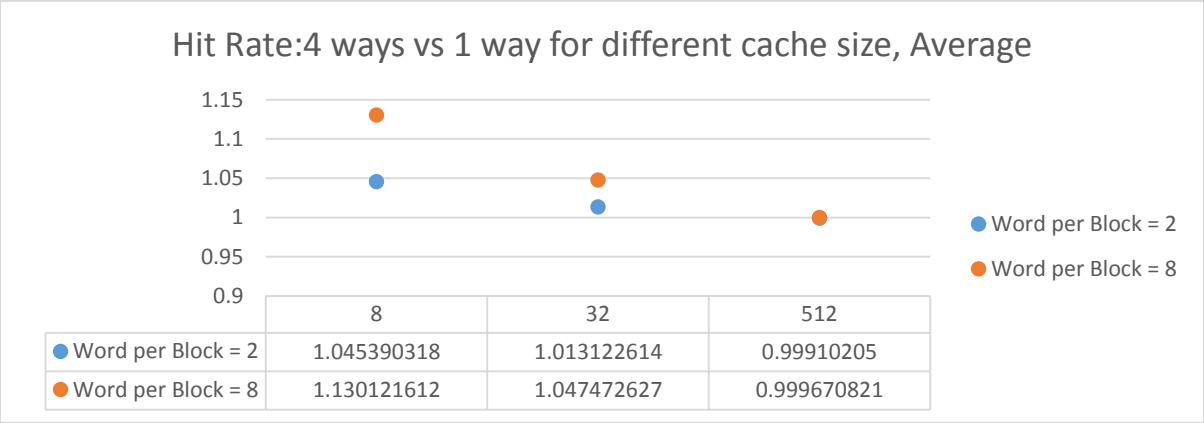
Ways	Cache Size	Words_Per_Bock	Hit_Time	Sets	Index_Size	Tag_Size
4	512	8	3	4096	12	15



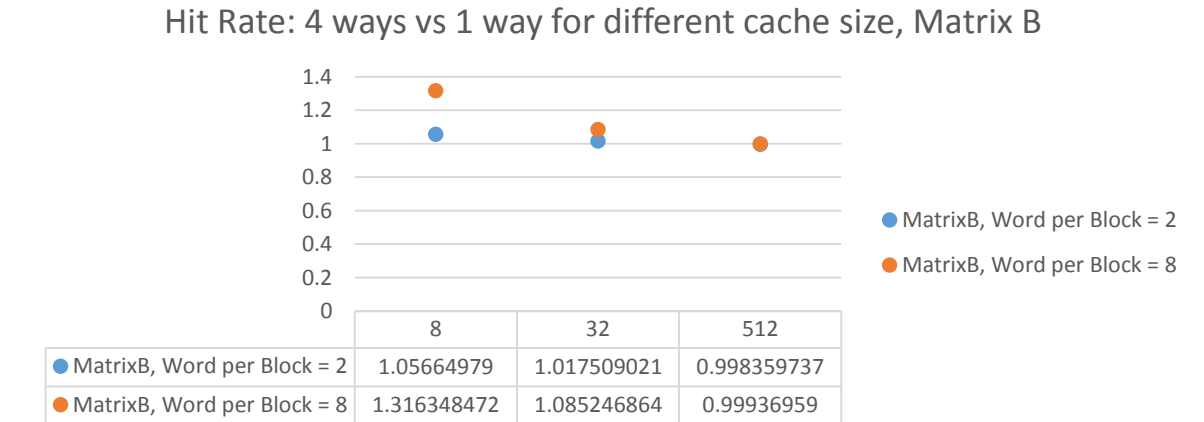
Effect by associative



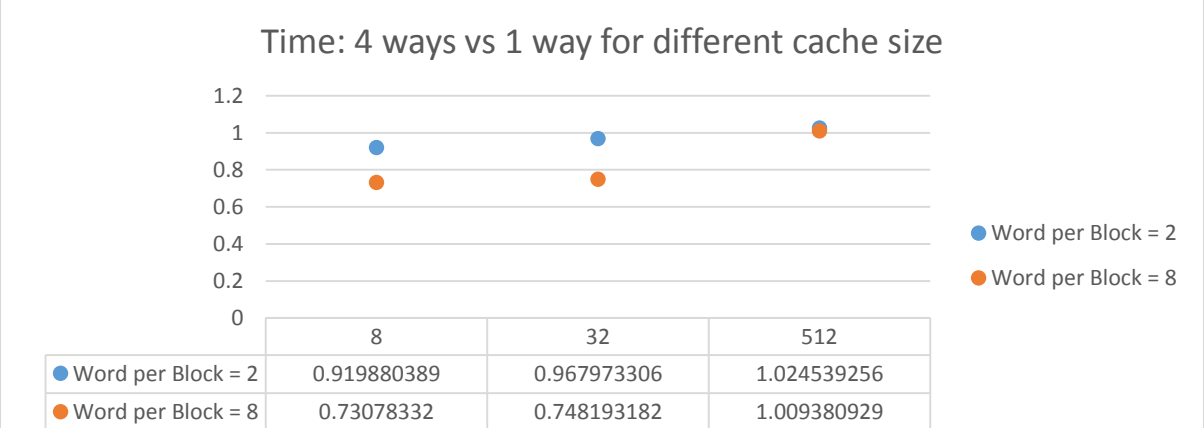
	8	32	512
MatrixA, Word per Block = 2	1.040429633	1.010917355	0.999843727
MatrixA, Word per Block = 8	1.038984042	1.018219518	0.999971988



	8	32	512
Word per Block = 2	1.045390318	1.013122614	0.99910205
Word per Block = 8	1.130121612	1.047472627	0.999670821

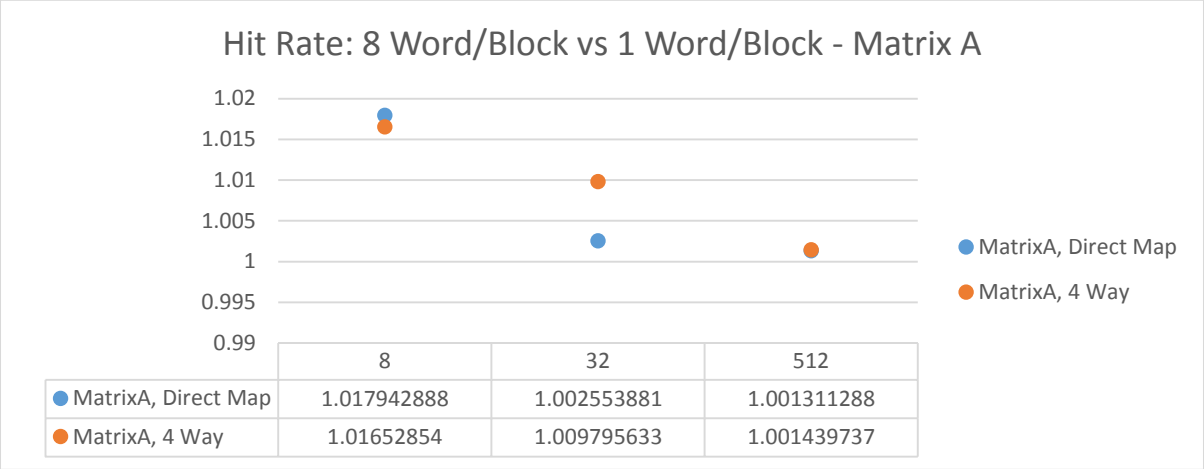


	8	32	512
MatrixB, Word per Block = 2	1.05664979	1.017509021	0.998359737
MatrixB, Word per Block = 8	1.316348472	1.085246864	0.99936959

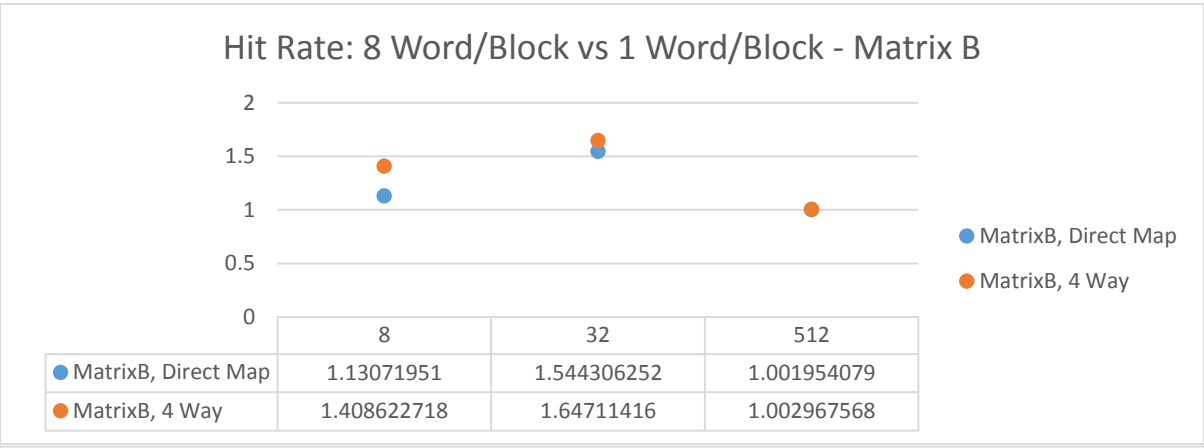


	8	32	512
Word per Block = 2	0.919880389	0.967973306	1.024539256
Word per Block = 8	0.73078332	0.748193182	1.009380929

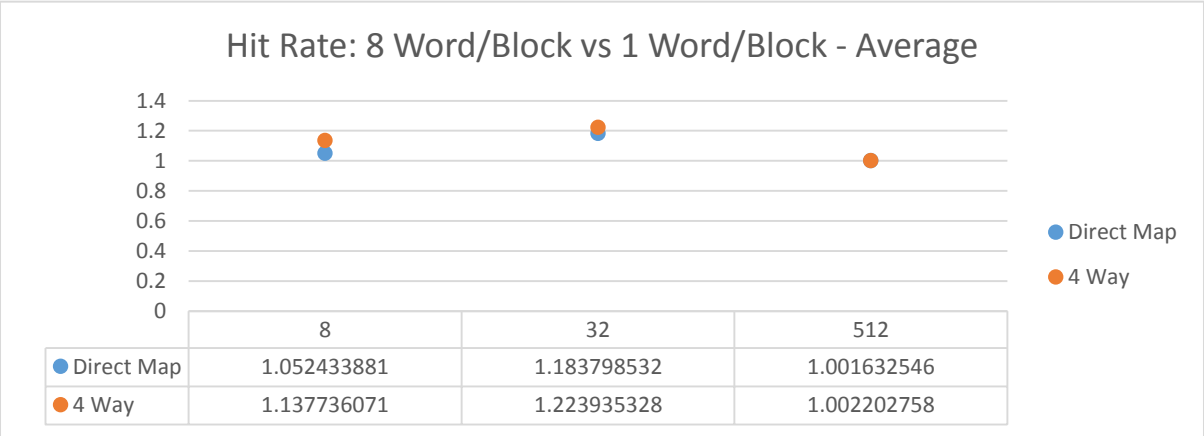
Effect by block size



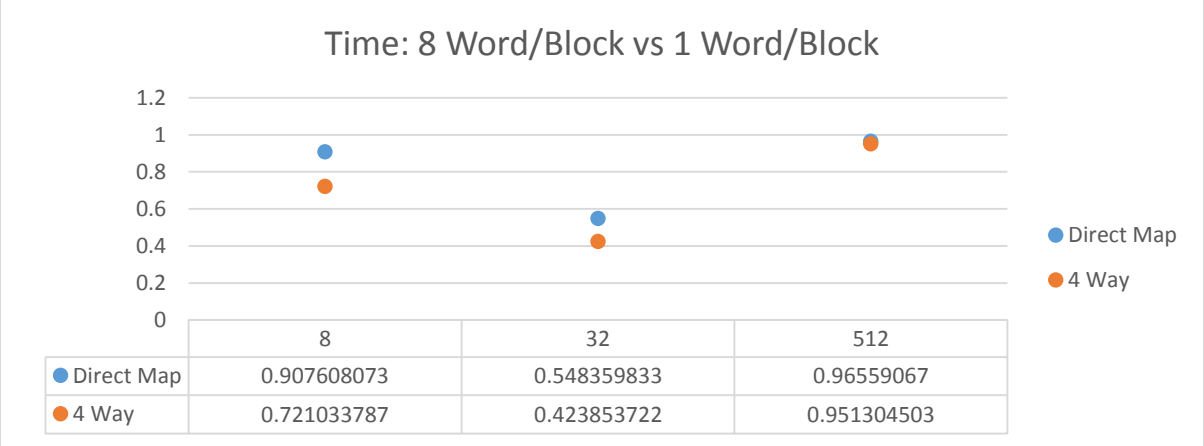
	8	32	512
MatrixA, Direct Map	1.017942888	1.002553881	1.001311288
MatrixA, 4 Way	1.01652854	1.009795633	1.001439737



	8	32	512
MatrixB, Direct Map	1.13071951	1.544306252	1.001954079
MatrixB, 4 Way	1.408622718	1.64711416	1.002967568



	8	32	512
Direct Map	1.052433881	1.183798532	1.001632546
4 Way	1.137736071	1.223935328	1.002202758



	8	32	512
Direct Map	0.907608073	0.548359833	0.96559067
4 Way	0.721033787	0.423853722	0.951304503

Detailed output for each matrix size iteration for each matrix size

Please refers to Report.xlsx