CS224

Section No.: 3

Spring 2021 Lab No.: 6

Your Full Name/Bilkent ID: Nurettin Onur Vural / 21902330

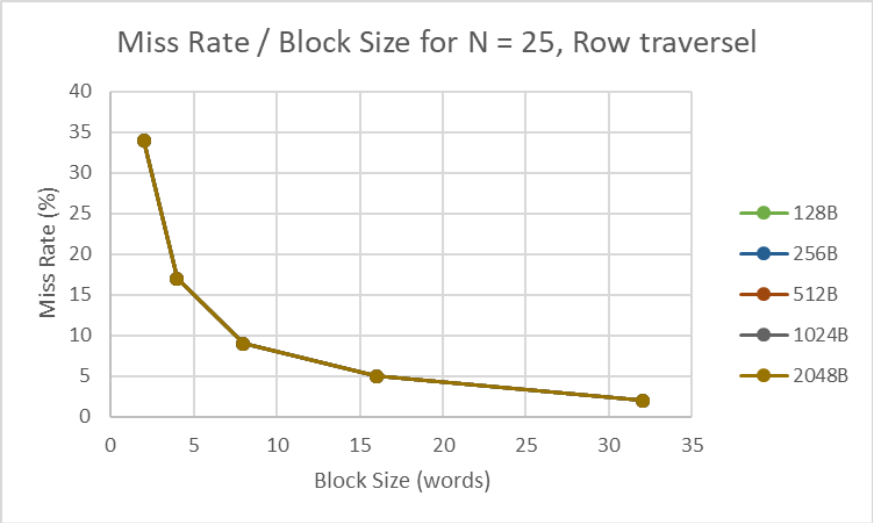
CS 224 LAB6-PART 2: Experiments with Data Cache Parameters

- **For N = 25**

a)

	Block Size (word)				
Cache Size (byte)	**2**	**4**	**8**	**16**	**32**
128 B	Cache miss count: 375 Cache miss rate: 34%	Cache miss count: 190 Cache miss rate: 17%	Cache miss count: 97 Cache miss rate: 9%	Cache miss count: 50 Cache miss rate: 5%	Cache miss count: 26 Cache miss rate: 2%
256 B	Cache miss count: 375 Cache miss rate: 34%	Cache miss count: 10033 Cache miss rate: 25%	Cache miss count: 97 Cache miss rate: 9%	Cache miss count: 50 Cache miss rate: 5%	Cache miss count: 26 Cache miss rate: 2%
512 B	Cache miss count: 375 Cache miss rate: 34%	Cache miss count: 10033 Cache miss rate: 25%	Cache miss count: 97 Cache miss rate: 9%	Cache miss count: 50 Cache miss rate: 5%	Cache miss count: 26 Cache miss rate: 2%
1024 B	Cache miss count: 375 Cache miss rate: 34%	Cache miss count: 10033 Cache miss rate: 25%	Cache miss count: 97 Cache miss rate: 9%	Cache miss count: 50 Cache miss rate: 5%	Cache miss count: 26 Cache miss rate: 2%
2048 B	Cache miss count: 20062 Cache miss rate: 50%	Cache miss count: 10033 Cache miss rate: 25%	Cache miss count: 97 Cache miss rate: 9%	Cache miss count: 50 Cache miss rate: 5%	Cache miss count: 26 Cache miss rate: 2%

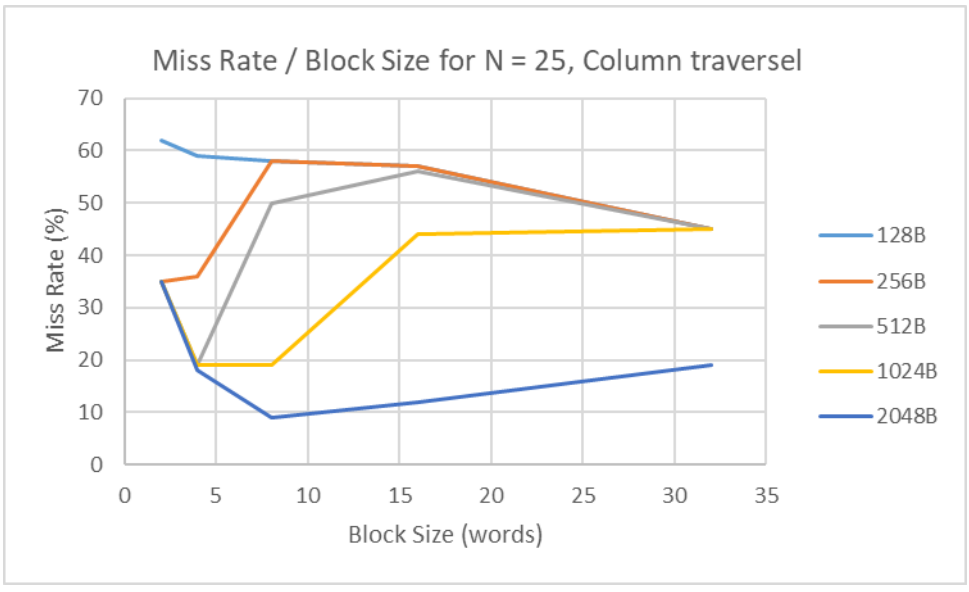
Table 1: Miss rates according to varied cache size and block size for row-major traversel



*Graph 1: Relation between block size and miss rate of 25*25 dimension matrix for row-major averaging*

	Block Size (word)				
Cache Size (byte)	**2**	**4**	**8**	**16**	**32**
128 B	Cache miss count: 687 Cache miss rate: 62%	Cache miss count: 657 Cache miss rate: 59%	Cache miss count: 643 Cache miss rate: 58%	Cache miss count: 635 Cache miss rate: 57%	Cache miss count: 503 Cache miss rate: 45%
256 B	Cache miss count: 387 Cache miss rate: 35%	Cache miss count: 399 Cache miss rate: 36%	Cache miss count: 643 Cache miss rate: 58%	Cache miss count: 635 Cache miss rate: 57%	Cache miss count: 503 Cache miss rate: 45%
512 B	Cache miss count: 387 Cache miss rate: 35%	Cache miss count: 207 Cache miss rate: 19%	Cache miss count: 553 Cache miss rate: 50%	Cache miss count: 626 Cache miss rate: 56%	Cache miss count: 503 Cache miss rate: 45%
1024 B	Cache miss count: 387 Cache miss rate: 35%	Cache miss count: 207 Cache miss rate: 19%	Cache miss count: 208 Cache miss rate: 19%	Cache miss count: 493 Cache miss rate: 44%	Cache miss count: 503 Cache miss rate: 45%
2048 B	Cache miss count: 387 Cache miss rate: 35%	Cache miss count: 195 Cache miss rate: 18%	Cache miss count: 104 Cache miss rate: 9%	Cache miss count: 138 Cache miss rate: 12%	Cache miss count: 211 Cache miss rate: 19%

Table 2: Miss rates according to varied cache size and block size for column-major traversel

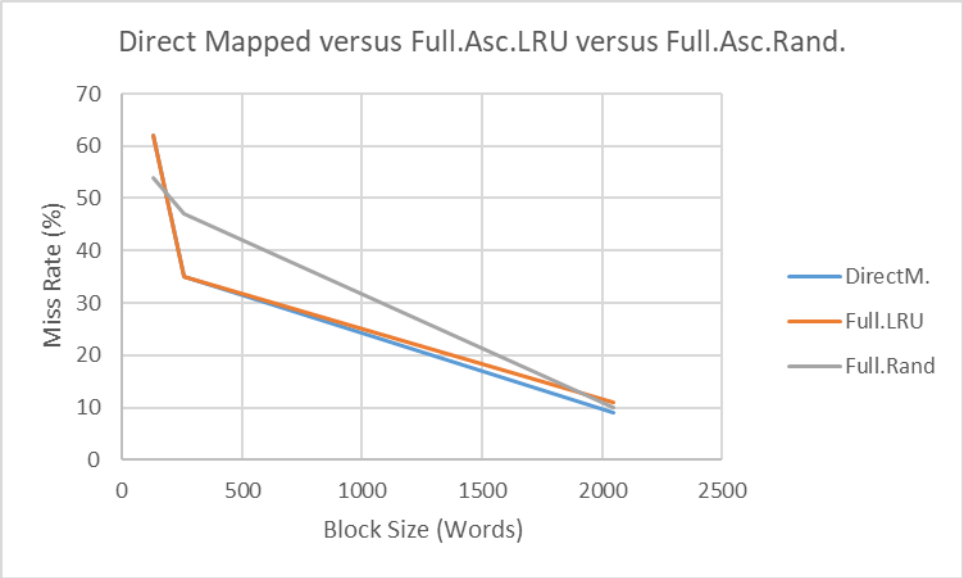


*Graph 2: Relation between block size and miss rate of 25*25 dimension matrix for column-major averaging*

b)

Hit quality / Charactheristics		Cache Architecture		
		Direct Mapped	**Fully Associative (LRU)**	**Fully Associative (Random)**
Poor	**Block Size: 2 words** **Cache Size: 128 bytes**	687 miss count 62% miss rate	687 miss count 62% miss rate	595 miss count 54% miss rate
Medium	**Block Size: 2 words** **Cache Size: 256 bytes**	387 miss count 35% miss rate	387 miss count 35% miss rate	518 miss count 47% miss rate
Good	**Block Size: 8 words** **Cache Size: 2048 bytes**	104 miss count 9% miss rate	118 miss count 11% miss rate	108 miss count 10% miss rate

*Table 3: Miss rates of varied hit qualities of 25*25 matrix column traversel according to cache architecture properties*



*Graph 3: Relation between block size and miss rate of 25*25 dimension matrix for column-major averaging with three different cache architectures*

It can be observed from the relation that Fully Asc. LRU and Direct Mapped architectures show parellelism overall and this is due to the fact that for column-major averaging technique, bringing neighbor items do not provide any extra advantage over Direct Mapping since in column traversel instead of the next items of the array, a much different strategy is playing out where the columnwise addresses are calculated. In this respect, as column traversel is used for our case, spatial locality principle does not contribute to a huge difference in terms of hit rate improvement. When it comes to random block replacement policy, it can also be noticed that the values are not much different but since it repleces the items in a random fashion, in some parts it provides efficency whereas other parts does not.

c)

Medium Hit Rate Configuration: 387 miss count & 35% miss rate (2 words, 256 bytes)

N (Number of Ways)	2	4	8	16
Hit Rate	58%	65%	65%	65%
Miss Rate	42%	35%	35%	35%
Number of Misses	471	387	387	387

Table 4: Data of medium hit rate for varied set number values

Good Hit Rate Configuration: 104 miss count & 9% miss rate (8 words, 2048 bytes)

N (Number of Ways)	2	4	8	16
Hit Rate	90%	90%	89%	89%
Miss Rate	10%	10%	11%	11%
Number of Misses	108	114	118	118

Table 5: Data of good hit rate for varied set number values

Poor Hit Rate Configuration: 687 miss count & 62% miss rate (2 words, 128 bytes)

N (Number of Ways)	2	4	8	16
Hit Rate	38%	38%	38%	38%
Miss Rate	62%	62%	62%	62%
Number of Misses	687	687	687	687

Table 6: Data of poor hit rate for varied set number values

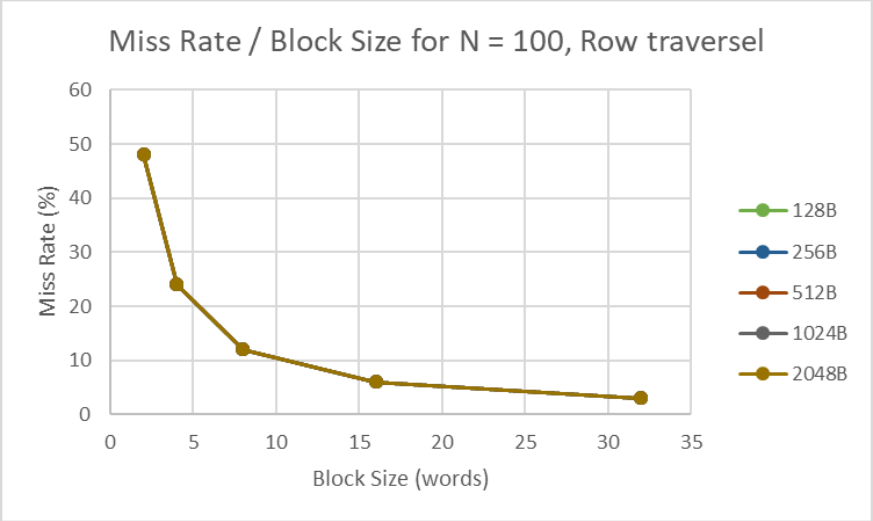
From the results it can be observed that; although it lowers the hardware cost, changing the number of sets in N-way set associative architecture does not provide any significant change for all three hit rate configurations in terms of improving the hit rate efficiency in this example. Medium hit rate operates on its expected value with the ste numbers bigger than 2, good hit rate produces the best results with set number 2, poor hit rate yields same result overall. Although there are such differences, almost all the possible set numbers in the tables produce the same result nonetheless and there may be two possible reasons behind this result. Firstly, as stated in part b, because of the nature of column-major addition the spatial locality principles hold same and regardless of the set number, the data is obtained by column address calculation which indicates that neighboring data values (next items in the array) are not the values to be used actually. Another possible reason is connected with the matrix size itself. Although the matrix size is considerably large, with it becoming larger and larger, approaching to infinity, the change for set number to make a difference in terms of improving the overall hit rate increases since it is much more important for cache memory to hold much larger amout of items within.

- **For N = 100**

a)

	Block Size (word)				
Cache Size (byte)	**2**	**4**	**8**	**16**	**32**
128 B	Cache miss count: 5062 Cache miss rate: 48%	Cache miss count: 2533 Cache miss rate: 24%	Cache miss count: 1268 Cache miss rate: 12%	Cache miss count: 635 Cache miss rate: 6%	Cache miss count: 319 Cache miss rate: 3%
256 B	Cache miss count: 5062 Cache miss rate: 48%	Cache miss count: 2533 Cache miss rate: 24%	Cache miss count: 1268 Cache miss rate: 12%	Cache miss count: 635 Cache miss rate: 6%	Cache miss count: 319 Cache miss rate: 3%
512 B	Cache miss count: 5062 Cache miss rate: 48%	Cache miss count: 2533 Cache miss rate: 24%	Cache miss count: 1268 Cache miss rate: 12%	Cache miss count: 635 Cache miss rate: 6%	Cache miss count: 319 Cache miss rate: 3%
1024 B	Cache miss count: 5062 Cache miss rate: 48%	Cache miss count: 2533 Cache miss rate: 24%	Cache miss count: 1268 Cache miss rate: 12%	Cache miss count: 635 Cache miss rate: 6%	Cache miss count: 319 Cache miss rate: 3%
2048 B	Cache miss count: 5062 Cache miss rate: 48%	Cache miss count: 2533 Cache miss rate: 24%	Cache miss count: 1268 Cache miss rate: 12%	Cache miss count: 635 Cache miss rate: 6%	Cache miss count: 319 Cache miss rate: 3%

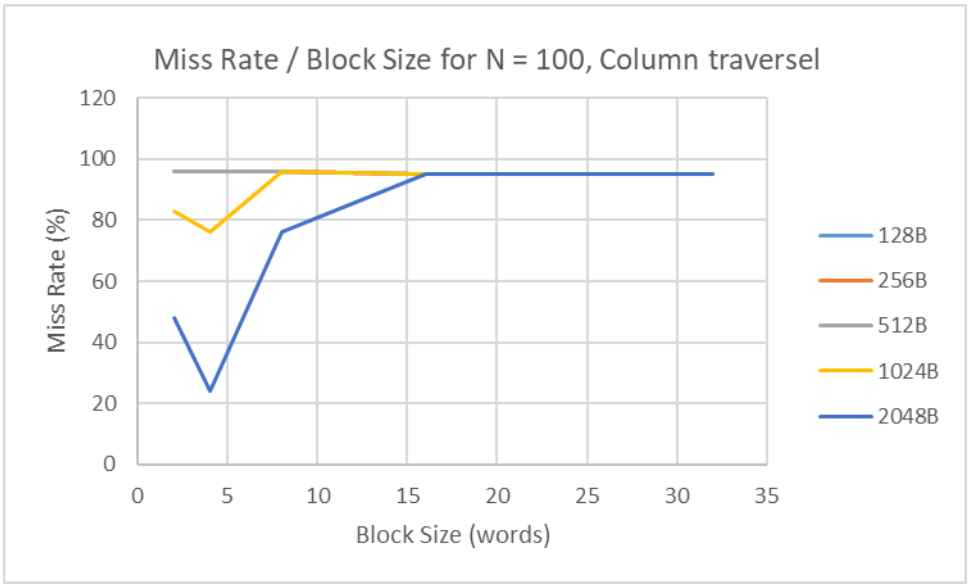
Table 7: Miss rates according to varied cache size and block size for row-major traversel



*Graph 4: Relation between block size and miss rate of 100*100 dimension matrix for row-major averaging*

	Block Size (word)				
Cache Size (byte)	**2**	**4**	**8**	**16**	**32**
128 B	Cache miss count: 10062 Cache miss rate: 96%	Cache miss count: 10032 Cache miss rate: 96%	Cache miss count: 10018 Cache miss rate: 96%	Cache miss count: 10010 Cache miss rate: 95%	Cache miss count: 10007 Cache miss rate: 95%
256 B	Cache miss count: 10062 Cache miss rate: 96%	Cache miss count: 10032 Cache miss rate: 96%	Cache miss count: 10018 Cache miss rate: 96%	Cache miss count: 10010 Cache miss rate: 95%	Cache miss count: 10007 Cache miss rate: 95%
512 B	Cache miss count: 10062 Cache miss rate: 96%	Cache miss count: 10032 Cache miss rate: 96%	Cache miss count: 10018 Cache miss rate: 96%	Cache miss count: 10010 Cache miss rate: 95%	Cache miss count: 10007 Cache miss rate: 95%
1024 B	Cache miss count: 8662 Cache miss rate: 83%	Cache miss count: 7932 Cache miss rate: 76%	Cache miss count: 10018 Cache miss rate: 96%	Cache miss count: 10010 Cache miss rate: 95%	Cache miss count: 10007 Cache miss rate: 95%
2048 B	Cache miss count: 5062 Cache miss rate: 48%	Cache miss count: 2532 Cache miss rate: 24%	Cache miss count: 7918 Cache miss rate: 76 %	Cache miss count: 10010 Cache miss rate: 95%	Cache miss count: 10007 Cache miss rate: 95%

Table 8: Miss rates according to varied cache size and block size for column-major traversel

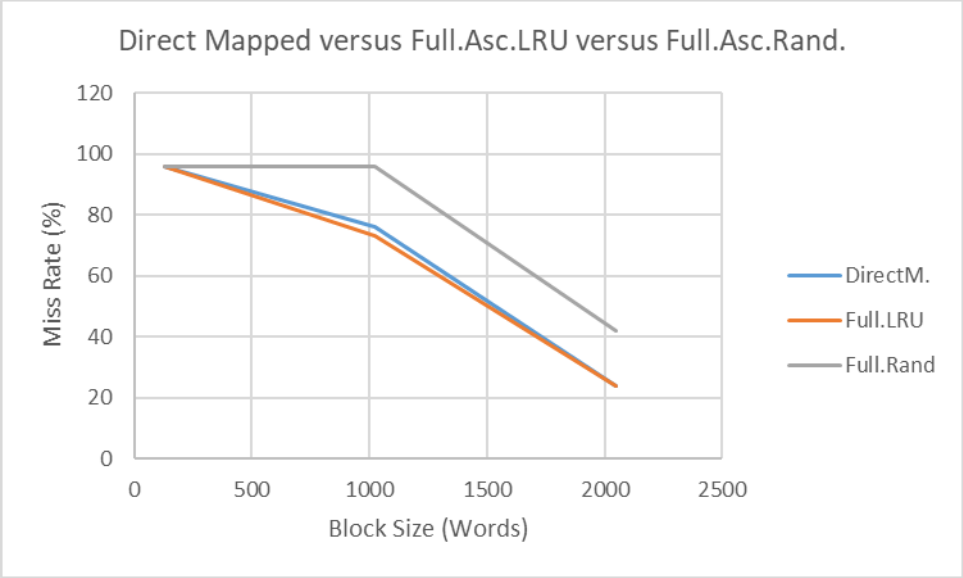


*Graph 5: Relation between block size and miss rate of 100*100 dimension matrix for column-major averaging*

b)

Hit quality / Charactheristics		Cache Architecture		
		Direct Mapped	**Fully Associative (LRU)**	**Fully Associative (Random)**
Poor	**Block Size: 2 words** **Cache Size: 128 bytes**	10062 miss count 96% miss rate	10062 miss count 96% miss rate	10054 miss count 96% miss rate
Medium	**Block Size: 4 words** **Cache Size: 1024 bytes**	7932 miss count 76% miss rate	10032 miss count 73% miss rate	7663 miss count 96% miss rate
Good	**Block Size: 4 words** **Cache Size: 2048 bytes**	2532 miss count 24% miss rate	2532 miss count 24% miss rate	4449 miss count 42% miss rate

*Table 9: Miss rates of varied hit qualities of 100*100 matrix column traversel according to cache architecture properties*



*Graph 6: Relation between block size and miss rate of 100*100 dimension matrix for column-major averaging with three different cache architectures*

Similar to Graph 3, again there is parallelity between Fully Asc. LRU and Direct Mapped architectures stemming from column traversal technique. Column traversel does not proceed by obtaining the data values next to each other but proceeds by checking columnwise addresses instead. Keeping nearby items, in other words next values in the array cannot provide any remarkable advantage in this case. When it comes to random block replacement policy, it can also be noticed that the values are not much different but since it repleces the items in a random fashion, in some parts it provides efficency whereas other parts does not. But for the specific data values in the graph, it can be observed that random block replacement policy mostly increases the miss rate isted of contributing to a larger hit rate.

c)

Medium Hit Rate Configuration: 7932 miss count & 76% miss rate (4 words, 1024 bytes)

N (Number of Ways)	**2**	**4**	**8**	**16**
Hit Rate	4%	4%	4%	4%
Miss Rate	96%	96%	96%	96%
Number of Misses	10032	10032	10032	10032

Table 10: Data of medium hit rate for varied set number values

Good Hit Rate Configuration: 2532 miss count & 24% miss rate (4 words, 2048 bytes)

N (Number of Ways)	**2**	**4**	**8**	**16**
Hit Rate	76%	76%	76%	76%
Miss Rate	24%	24%	24%	24%
Number of Misses	2532	2532	2532	2532

Table 11: Data of good hit rate for varied set number values

Poor Hit Rate Configuration: 10062 miss count & 96% miss rate (2 words, 128 bytes)

N (Number of Ways)	**2**	**4**	**8**	**16**
Hit Rate	4%	4%	4%	4%
Miss Rate	96%	96%	96%	96%
Number of Misses	10062	10062	10062	10062

Table 12: Data of poor hit rate for varied set number values

From the results it can be observed that; although it lowers the hardware cost, changing the number of sets in N-way set associative architecture does not provide any significant change for all three hit rate configurations in terms of improving the hit rate efficiency in this example. All the possible set numbers in the tables produce the same result and there may be two possible reasons behind this result. Firstly, as stated in part b, because of the nature of column-major addition the spatial locality principles hold same and regardless of the set number, the data is obtained by column address calculation which indicates that neighboring data values (next items in the array) are not the values to be used actually. Another possible reason is connected with the matrix size itself. Although the matrix size is considerably large, with it becoming larger and larger, approaching to infinity, the change for set number to make a difference in terms of improving the overall hit rate increases since it is much more important for cache memory to hold much larger amout of items within.