CSCE 2303: Computer Organization and Assembly Language Programming

Project 2

Cache Simulator

Khalid A. Mohamed 900153041

Objective:

Simulating a direct mapped cache to measure its performance and mechanisms; hit and miss ratios through variable line/block sizes, running by a variety of applications.

Apparatus:

C++ skeleton: memory reference generator, implemented by the functions memGenA(), memGenB(), memGenC(), memGenD(), memGenE() and memGenF() besides custom test cases.

Method:

Measuring the hit ratio for different addresses' generator for line sizes 4, 8, 16, 32, 64, 128.
1,000,000 memory references are used.

Output:

The source code's output produces the six line sizes simultaneously for quicker sampling in each experiment. This was implemented though an Array containing the cache TAG address and the valid bit only, neglecting the data space since data replacement will not be considered. To make the six line sizes considered, the cache array has [2 * 6] columns and 99999 rows (for simplicity). However, using nested pointers or vectors would have been more optimum. Below are screenshots for the last part in an example of the output format (for memGenA() address generator function).

0x000f4223 (Hit)	0x000f4223 (Hit)	0x000f4223 (Hit)	0x000f4223 (Hit)	0x000f4223 (Hit)	0x000f4223 (Hit)
0x000f4224 (Miss)	0x000f4224 (Hit)	0x000f4224 (Hit)	0x000f4224 (Hit)	0x000f4224 (Hit)	0x000f4224 (Hit)
0x000f4225 (Hit)	0x000f4225 (Hit)	0x000f4225 (Hit)	0x000f4225 (Hit)	0x000f4225 (Hit)	0x000f4225 (Hit)
0x000f4226 (Hit)	0x000f4226 (Hit)	0x000f4226 (Hit)	0x000f4226 (Hit)	0x000f4226 (Hit)	0x000f4226 (Hit)
0x000f4227 (Hit)	0x000f4227 (Hit)	0x000f4227 (Hit)	0x000f4227 (Hit)	0x000f4227 (Hit)	0x000f4227 (Hit)
0x000f4228 (Miss)	0x000f4228 (Miss)	0x000f4228 (Hit)	0x000f4228 (Hit)	0x000f4228 (Hit)	0x000f4228 (Hit)
0x000f4229 (Hit)	0x000f4229 (Hit)	0x000f4229 (Hit)	0x000f4229 (Hit)	0x000f4229 (Hit)	0x000f4229 (Hit)
0x000f422a (Hit)	0x000f422a (Hit)	0x000f422a (Hit)	0x000f422a (Hit)	0x000f422a (Hit)	0x000f422a (Hit)
0x000f422b (Hit)	0x000f422b (Hit)	0x000f422b (Hit)	0x000f422b (Hit)	0x000f422b (Hit)	0x000f422b (Hit)
0x000f422c (Miss)	0x000f422c (Hit)	0x000f422c (Hit)	0x000f422c (Hit)	0x000f422c (Hit)	0x000f422c (Hit)
0x000f422d (Hit)	0x000f422d (Hit)	0x000f422d (Hit)	0x000f422d (Hit)	0x000f422d (Hit)	0x000f422d (Hit)
0x000f422e (Hit)	0x000f422e (Hit)	0x000f422e (Hit)	0x000f422e (Hit)	0x000f422e (Hit)	0x000f422e (Hit)
0x000f422f (Hit)	0x000f422f (Hit)	0x000f422f (Hit)	0x000f422f (Hit)	0x000f422f (Hit)	0x000f422f (Hit)
0x000f4230 (Miss)	0x000f4230 (Miss)	0x000f4230 (Miss)	0x000f4230 (Hit)	0x000f4230 (Hit)	0x000f4230 (Hit)
0x000f4231 (Hit)	0x000f4231 (Hit)	0x000f4231 (Hit)	0x000f4231 (Hit)	0x000f4231 (Hit)	0x000f4231 (Hit)
0x000f4232 (Hit)	0x000f4232 (Hit)	0x000f4232 (Hit)	0x000f4232 (Hit)	0x000f4232 (Hit)	0x000f4232 (Hit)
0x000f4233 (Hit)	0x000f4233 (Hit)	0x000f4233 (Hit)	0x000f4233 (Hit)	0x000f4233 (Hit)	0x000f4233 (Hit)
0x000f4234 (Miss)	0x000f4234 (Hit)	0x000f4234 (Hit)	0x000f4234 (Hit)	0x000f4234 (Hit)	0x000f4234 (Hit)
0x000f4235 (Hit)	0x000f4235 (Hit)	0x000f4235 (Hit)	0x000f4235 (Hit)	0x000f4235 (Hit)	0x000f4235 (Hit)
0x000f4236 (Hit)	0x000f4236 (Hit)	0x000f4236 (Hit)	0x000f4236 (Hit)	0x000f4236 (Hit)	0x000f4236 (Hit)
0x000f4237 (Hit)	0x000f4237 (Hit)	0x000f4237 (Hit)	0x000f4237 (Hit)	0x000f4237 (Hit)	0x000f4237 (Hit)
0x000f4238 (Miss)	0x000f4238 (Miss)	0x000f4238 (Hit)	0x000f4238 (Hit)	0x000f4238 (Hit)	0x000f4238 (Hit)
0x000f4239 (Hit)	0x000f4239 (Hit)	0x000f4239 (Hit)	0x000f4239 (Hit)	0x000f4239 (Hit)	0x000f4239 (Hit)
0x000f423a (Hit)	0x000f423a (Hit)	0x000f423a (Hit)	0x000f423a (Hit)	0x000f423a (Hit)	0x000f423a (Hit)
0x000f423b (Hit)	0x000f423b (Hit)	0x000f423b (Hit)	0x000f423b (Hit)	0x000f423b (Hit)	0x000f423b (Hit)
0x000f423c (Miss)	0x000f423c (Hit)	0x000f423c (Hit)	0x000f423c (Hit)	0x000f423c (Hit)	0x000f423c (Hit)
0x000f423d (Hit)	0x000f423d (Hit)	0x000f423d (Hit)	0x000f423d (Hit)	0x000f423d (Hit)	0x000f423d (Hit)
0x000f423e (Hit)	0x000f423e (Hit)	0x000f423e (Hit)	0x000f423e (Hit)	0x000f423e (Hit)	0x000f423e (Hit)
0x000f423f (Hit)	0x000f423f (Hit)	0x000f423f (Hit)	0x000f423f (Hit)	0x000f423f (Hit)	0x000f423f (Hit)
Hit ratio = 75%	Hit ratio = 87%	Hit ratio = 93%	Hit ratio = 96%	Hit ratio = 98%	Hit ratio = 99%

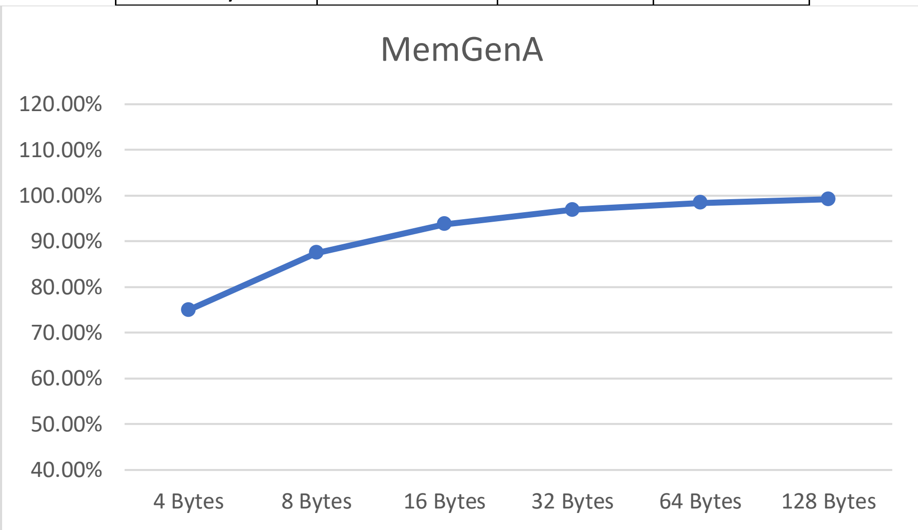
```
Line Size = 32 Bytes
    Hit ratio = 96%
    Miss ratio = 4%
    Hits = 968750
    Misses = 31250
    Compulsory Misses = 2048
    Capcity Misses = 29202
    Conflict = 0

Line Size = 64 Bytes
    Hit ratio = 98%
    Miss ratio = 2%
    Hits = 984375
    Misses = 15625
    Compulsory Misses = 1024
    Capcity Misses = 14601
    Conflict = 0

Line Size = 128 Bytes
    Hit ratio = 99%
    Miss ratio = 1%
    Hits = 992187
    Misses = 7813
    Compulsory Misses = 512
    Capcity Misses = 7301
    Conflict = 0
```

```
Line Size = 4 Bytes
    Hit ratio = 75%
    Miss ratio = 25%
    Hits = 750000
    Misses = 250000
    Compulsory Misses = 16384
    Capcity Misses = 233616
    Conflict = 0

Line Size = 8 Bytes
    Hit ratio = 87%
    Miss ratio = 13%
    Hits = 875000
    Misses = 125000
    Compulsory Misses = 8192
    Capcity Misses = 116808
    Conflict = 0

Line Size = 16 Bytes
    Hit ratio = 93%
    Miss ratio = 7%
    Hits = 937500
    Misses = 62500
    Compulsory Misses = 4096
    Capcity Misses = 58404
    Conflict = 0
```

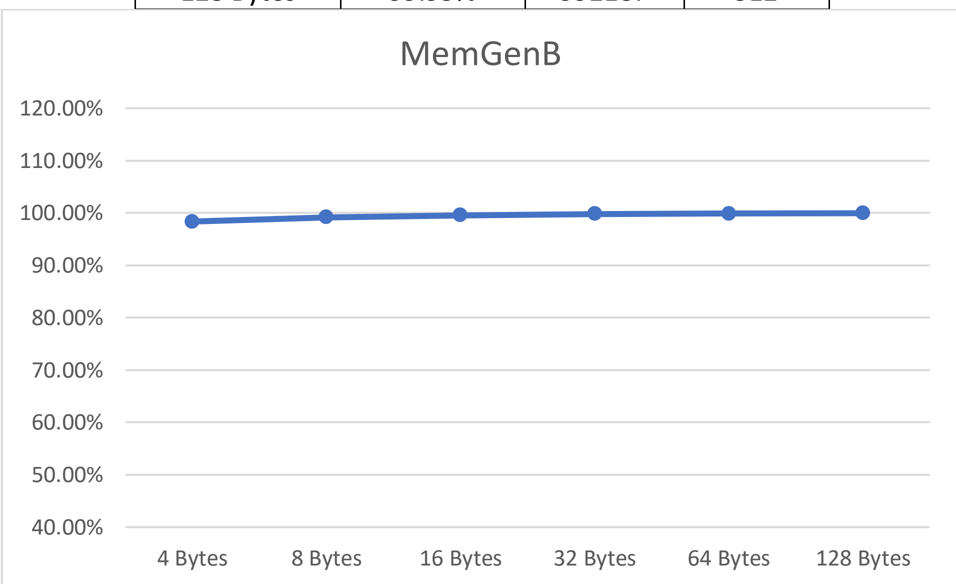
Furthermore, the types of misses are also calculated amongst the three C's: Compulsory, Capacity and Conflict misses. As shown above, this method particularly aids in measuring the performance of the six line sizes in a single run. Leading to easier observation of trends and effects of changing the line/block size.

Results

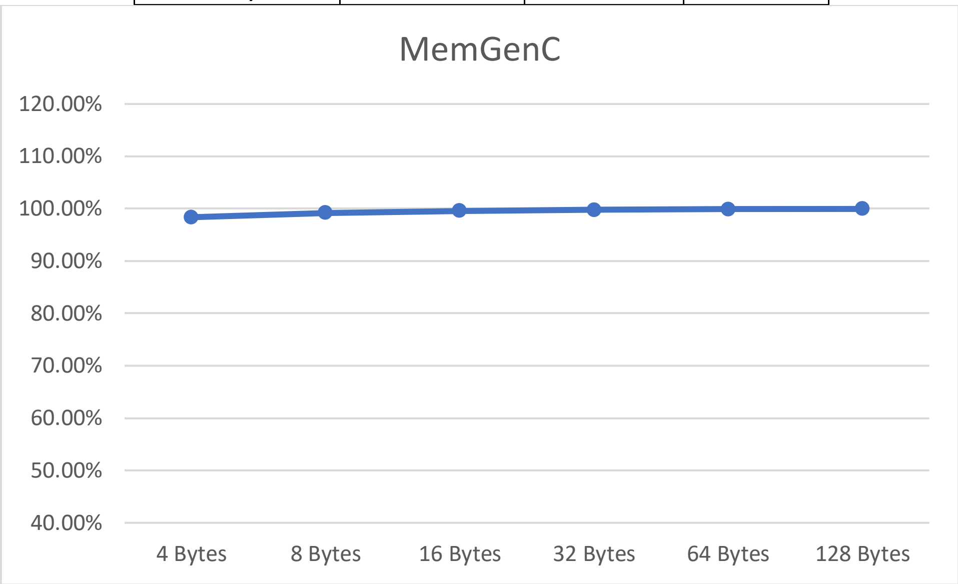
MemGenA			
Line Size	Hit Ratio	Hits	Misses
4 Bytes	75.00%	750000	250000
8 Bytes	87.50%	875000	125000
16 Bytes	93.75%	937500	62500
32 Bytes	96.88%	968750	31250
64 Bytes	98.44%	984375	15625
128 Bytes	99.22%	992187	7813



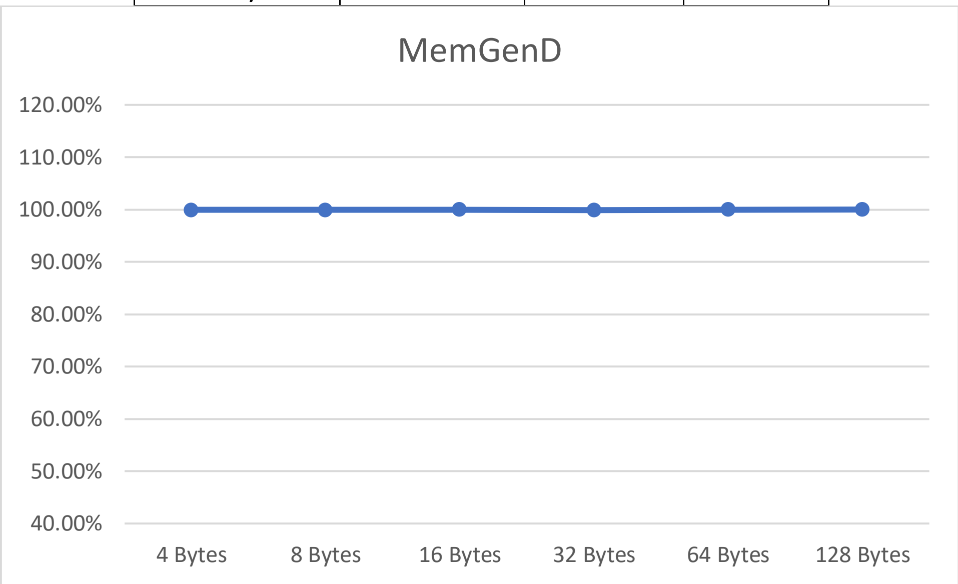
MemGenB			
Line Size	Hit Ratio	Hits	Misses
4 Bytes	98.36%	983616	16384
8 Bytes	99.18%	991808	8192
16 Bytes	99.59%	995904	4096
32 Bytes	99.80%	997952	2048
64 Bytes	99.90%	998976	1024
128 Bytes	99.95%	992187	512



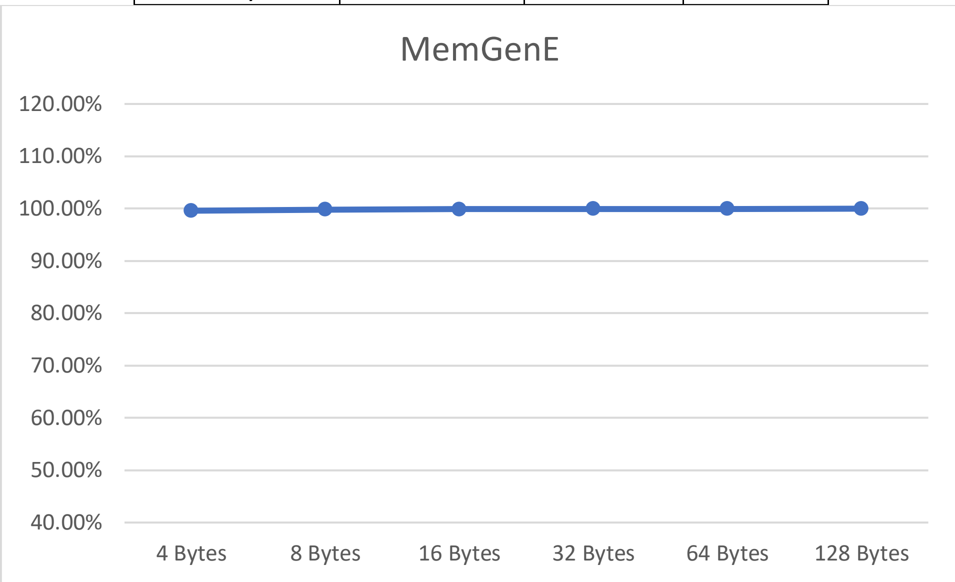
MemGenC			
Line Size	Hit Ratio	Hits	Misses
4 Bytes	98.36%	983616	16384
8 Bytes	99.18%	991808	8192
16 Bytes	99.59%	995904	4096
32 Bytes	99.80%	997952	2048
64 Bytes	99.90%	998976	1024
128 Bytes	99.95%	992187	512



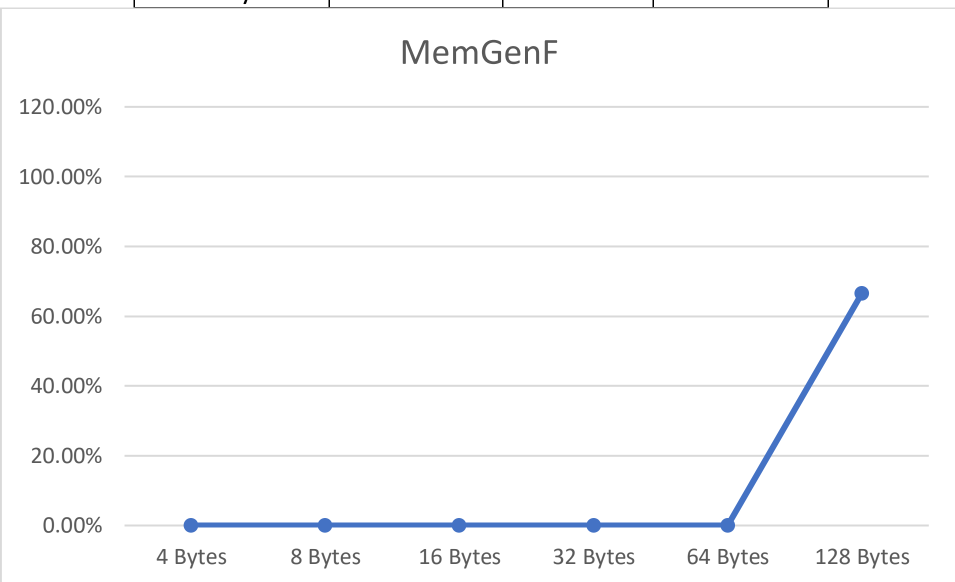
MemGenD			
Line Size	Hit Ratio	Hits	Misses
4 Bytes	99.90%	998976	1024
8 Bytes	99.95%	999488	512
16 Bytes	99.97%	999744	256
32 Bytes	99.89%	998872	1128
64 Bytes	99.99%	999936	64
128 Bytes	100.00%	992187	32



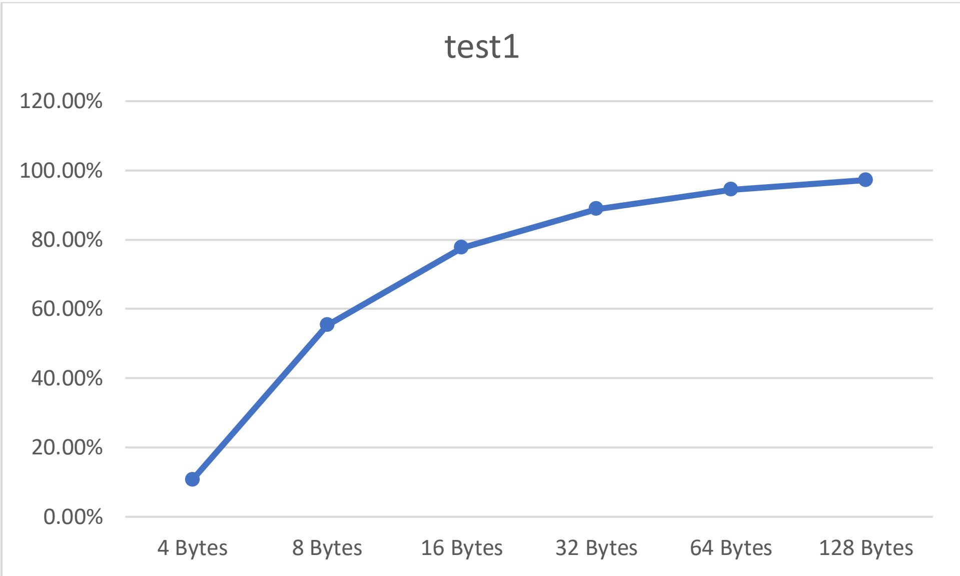
MemGenE			
Line Size	Hit Ratio	Hits	Misses
4 Bytes	99.59%	995904	4096
8 Bytes	99.80%	997952	2048
16 Bytes	99.90%	998976	1024
32 Bytes	99.95%	999488	512
64 Bytes	99.97%	999744	256
128 Bytes	99.99%	992187	128



MemGenF			
Line Size	Hit Ratio	Hits	Misses
4 Bytes	0.00%	0	1000000
8 Bytes	0.00%	0	1000000
16 Bytes	0.00%	0	1000000
32 Bytes	0.00%	0	1000000
64 Bytes	0.00%	0	1000000
128 Bytes	66.49%	992187	500001



test1			
Line Size	Hit Ratio	Hits	Misses
4 Bytes	10.71%	107142	892858
8 Bytes	55.36%	553571	446429
16 Bytes	77.68%	776785	223215
32 Bytes	88.84%	888392	111608
64 Bytes	94.42%	944196	55804
128 Bytes	97.21%	972098	27902



Overall Performance Chart

Analysis and Conclusion:

As observed through the different line sizes, increasing the line sizes caused a significant increase in memGenA() and test1()'s hit ratios, besides a slight increase in it in memGenB(), memGenC(), memGenD() and memGen(E). Regarding memGenF(), the presence of hits only appeared in the maximum block size (128 bytes). Which implies that increasing the line size is a valid solution for decreasing the misses' rates. However, keeping in mind that increasing the line size in direct mapped caches decreases the overhead values, it results in reducing the amount of lines in fixed cache sizes. Hence, raising the number of misses caused by conflicts.