NC State University

Department of Electrical and Computer Engineering

ECE 406/506: Fall 2021

Project #2: Coherence Protocols

by

Cristian Hellmer

For each of the tables for each protocol, the runs which are referenced correspond to:

Run 1	Cache Size: 256 KB, Associativity: 8, Block Size: 64B
Run 2	Cache Size: 512 KB, Associativity: 8, Block Size: 64B
Run 3	Cache Size: 1 MB, Associativity: 8, Block Size: 64B
Run 4	Cache Size: 2 MB, Associativity: 8, Block Size: 64B
Run 5	Cache Size: 1 MB, Associativity: 4, Block Size: 64B
Run 6	Cache Size: 1 MB, Associativity: 8, Block Size: 64B
Run 7	Cache Size: 1 MB, Associativity: 16, Block Size: 64B
Run 8	Cache Size: 1 MB, Associativity: 8, Block Size: 64B
Run 9	Cache Size: 1 MB, Associativity: 8, Block Size: 128B
Run 10	Cache Size: 1 MB, Associativity: 8, Block Size: 256B

MSI Protocol										
Run	1	2	3	4	5	6	7	8	9	10
Reads	451857	451857	451857	451857	451857	451857	451857	451857	451857	451857
Read misses	23164	23101	23075	23069	23130	23075	23034	23075	21451	20181
Writes	48143	48143	48143	48143	48143	48143	48143	48143	48143	48143
Write misses	161	161	161	161	161	161	161	161	160	160
Average miss rate	4.67%	4.65%	4.65%	4.65%	4.66%	4.65%	4.64%	4.65%	4.33%	4.07%
Writebacks	1001	736	668	637	913	668	449	668	1077	1420
Cache-to-cache transfers	0	0	0	0	0	0	0	0	0	0
Memory transactions	27020	26692	26598	26561	26898	26598	26338	26598	25446	24610
Interventions	259	263	265	265	260	265	266	265	440	636
Invalidations	8076	8076	8076	8076	8076	8076	8076	8076	8276	8544
Flushes	413	417	419	419	414	419	420	419	594	790
BusRdX	2855	2855	2855	2855	2855	2855	2855	2855	2918	3009

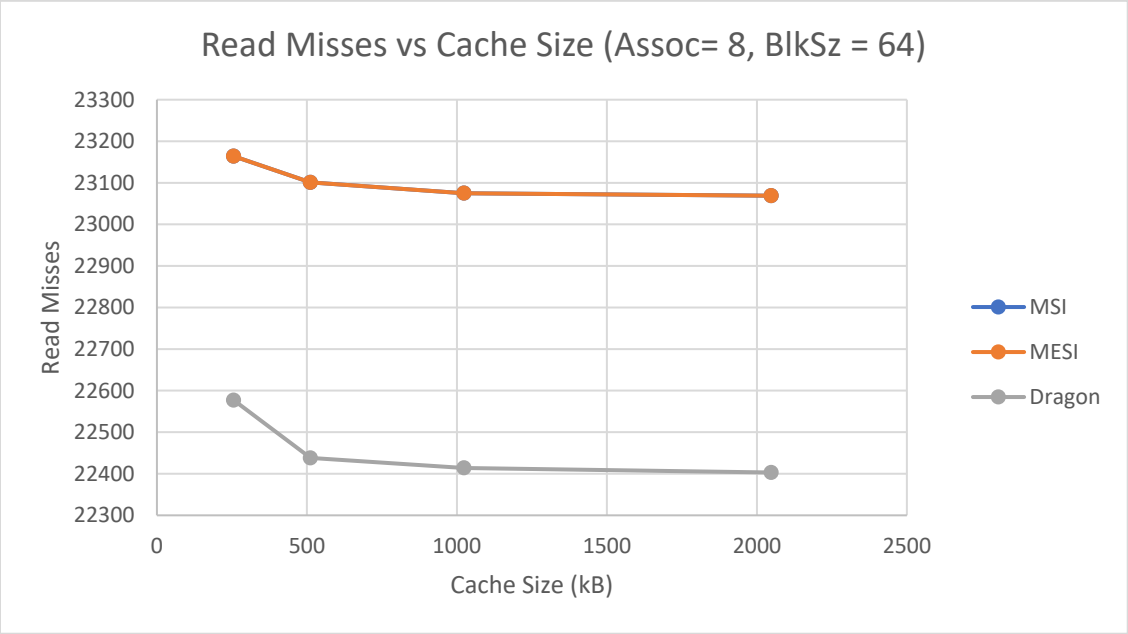
MESI Protocol										
Run	1	2	3	4	5	6	7	8	9	10
Reads	451857	451857	451857	451857	451857	451857	451857	451857	451857	451857
Read misses	23164	23101	23075	23069	23130	23075	23034	23075	21451	20181
Writes	48143	48143	48143	48143	48143	48143	48143	48143	48143	48143
Write misses	161	161	161	161	161	161	161	161	160	160
Average miss rate	4.67%	4.65%	4.65%	4.65%	4.66%	4.65%	4.64%	4.65%	4.33%	4.07%
Writebacks	1001	736	668	637	913	668	449	668	1077	1420
Cache-to-cache transfers	17663	17619	17599	17593	17638	17599	17566	17599	16519	15723
Memory transactions	6663	6379	6305	6274	6566	6305	6078	6305	6169	6038
Interventions	5848	5834	5830	5830	5841	5830	5823	5830	5466	5197
Invalidations	8076	8076	8076	8076	8076	8076	8076	8076	8276	8544
Flushes	413	417	419	419	414	419	420	419	594	790
BusRdX	161	161	161	161	161	161	161	161	160	160

Dragon Protocol										
Run	1	2	3	4	5	6	7	8	9	10
Reads	451857	451857	451857	451857	451857	451857	451857	451857	451857	451857
Read misses	22577	22438	22414	22403	22473	22414	22341	22414	20315	18531
Writes	48143	48143	48143	48143	48143	48143	48143	48143	48143	48143
Write misses	7	7	7	7	7	7	7	7	6	6
Average miss rate	4.52%	4.49%	4.48%	4.48%	4.50%	4.48%	4.47%	4.48%	4.07%	3.71%
Writebacks	935	531	445	426	597	445	209	445	599	779
Cache-to-cache transfers	0	0	0	0	0	0	0	0	0	0
Memory transactions	23519	22976	22866	22836	23077	22866	22557	22866	20920	19316
Interventions	5629	5595	5589	5586	5604	5589	5571	5589	5066	4619
Invalidations	0	0	0	0	0	0	0	0	0	0
Flushes	27	24	24	24	24	24	24	24	27	32
BusRdX	0	0	0	0	0	0	0	0	0	0

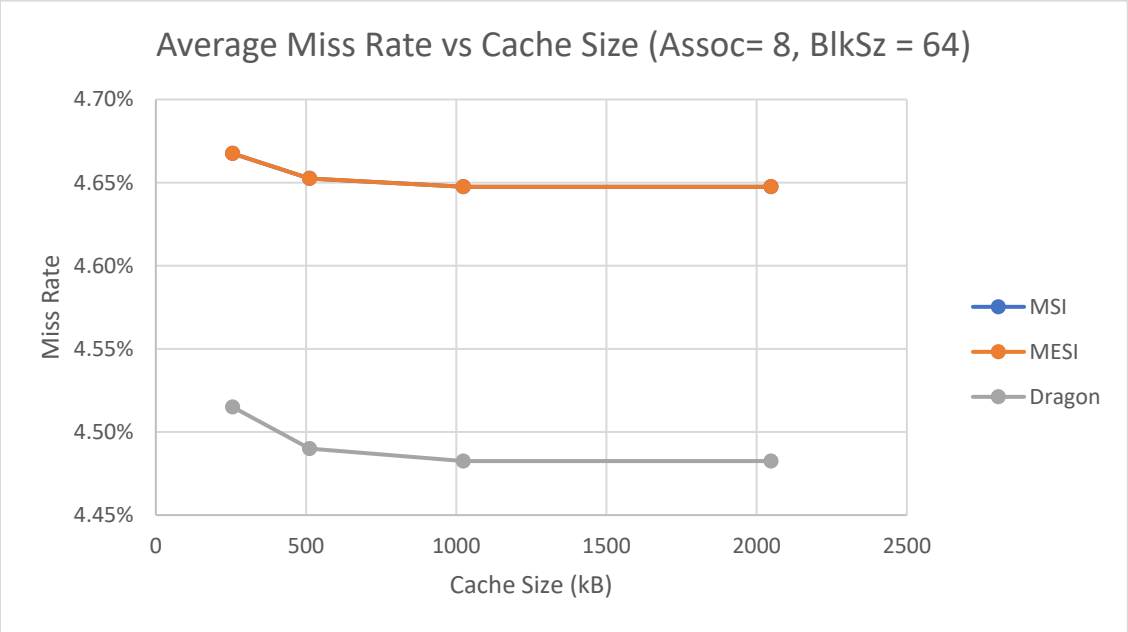
Below are the graphs for the different configurations which were tested. Some of the data was not graphed due to its lack of significance. The reasons that data was left off include the following.

- No change in the reads for any protocols.
- No change in the writes for any protocols.
- No/minimal change in the write misses for the protocols.
- No cache-to-cache transfers for the MSI and Dragon protocols.
- No change in invalidations for MSI and MESI protocols for varying cache size and associativity.
- No invalidations or BusRdX for Dragon protocol.
- No change in BusRdX for MSI and MESI Protocols.

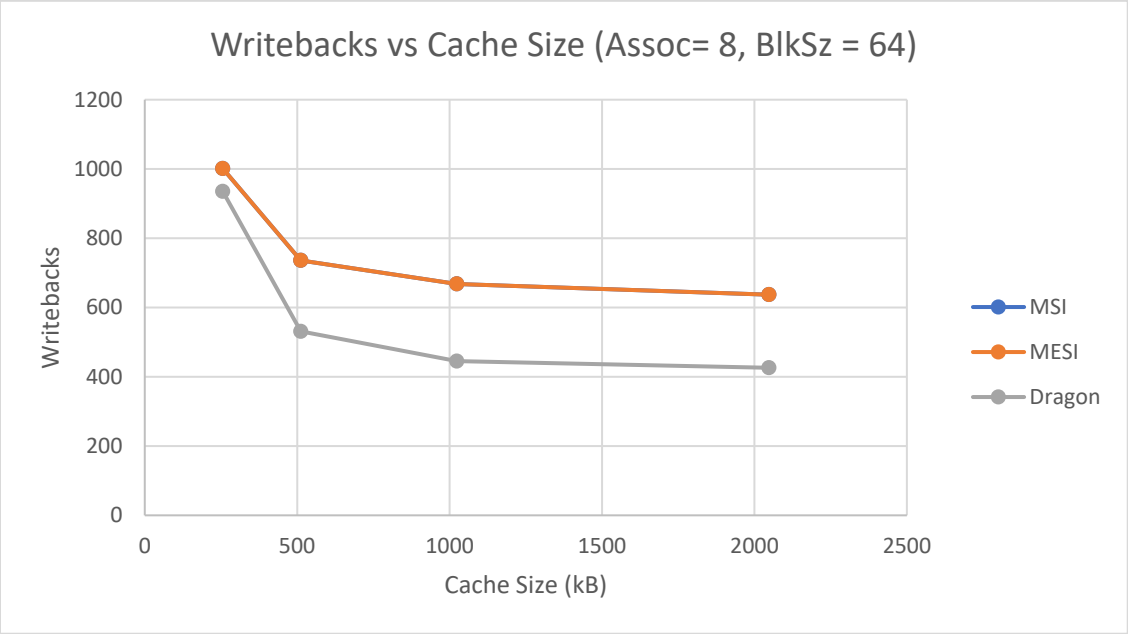
GRAPH #1



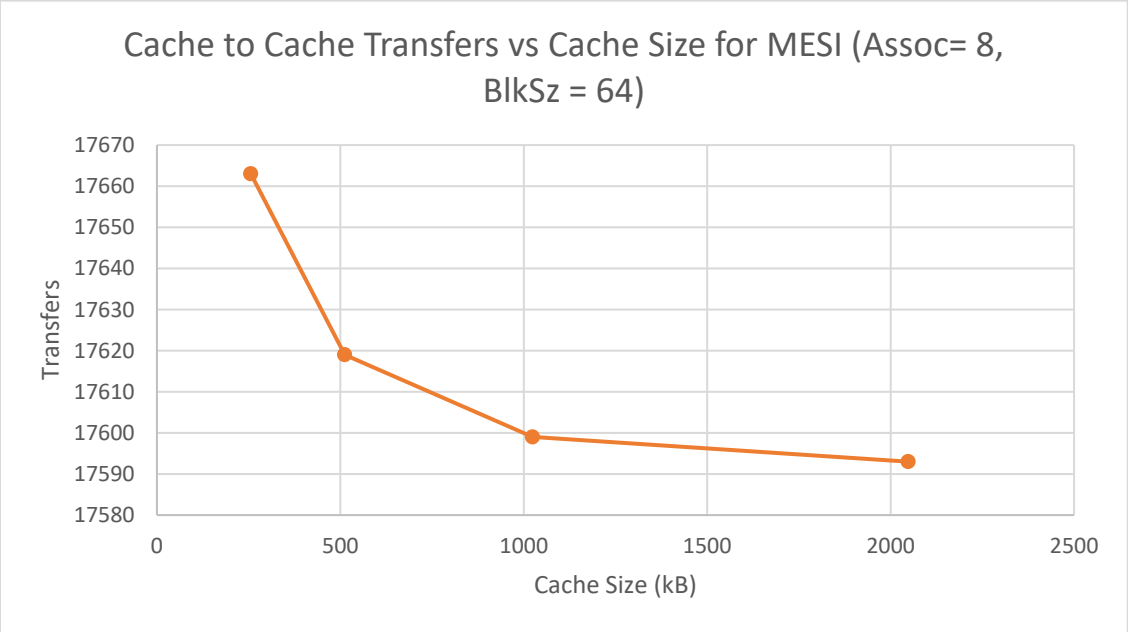
GRAPH #2



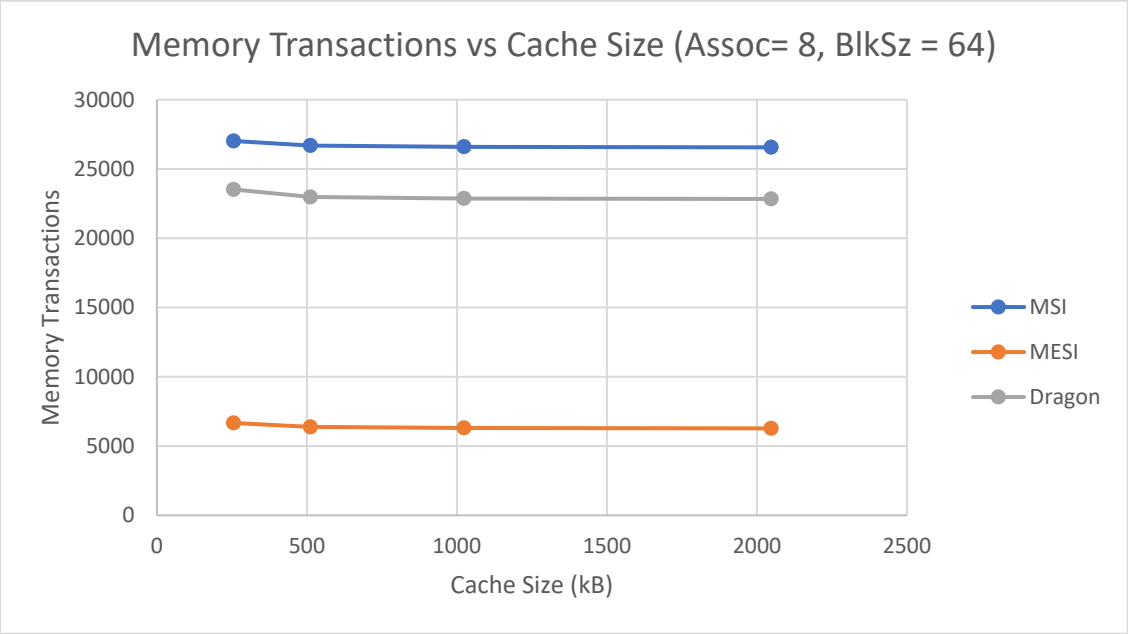
GRAPH #3



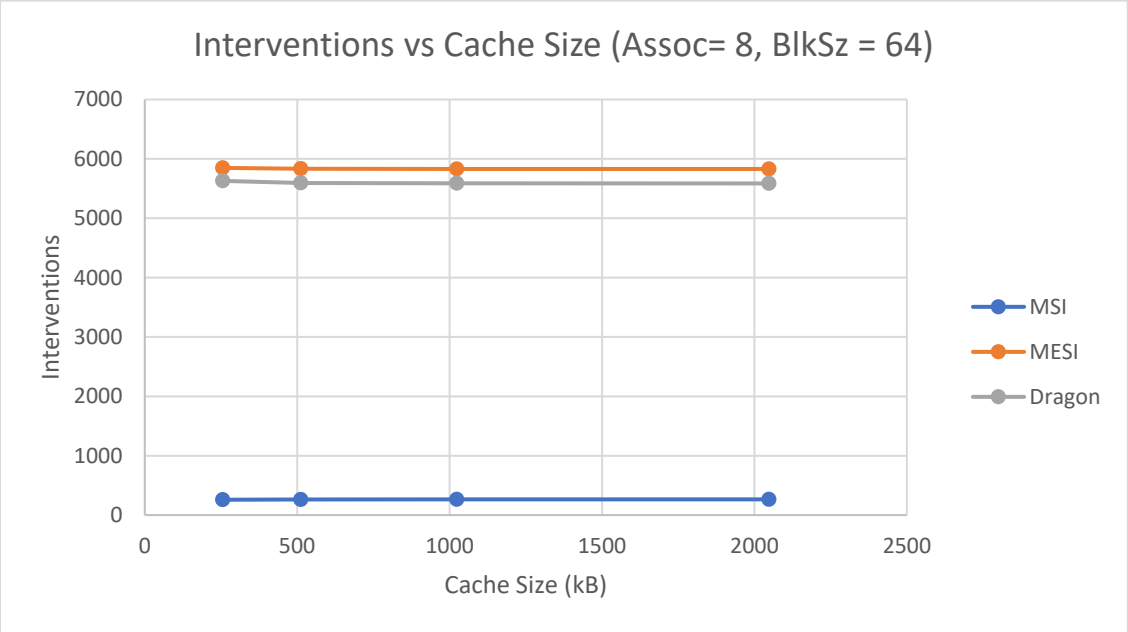
GRAPH #4



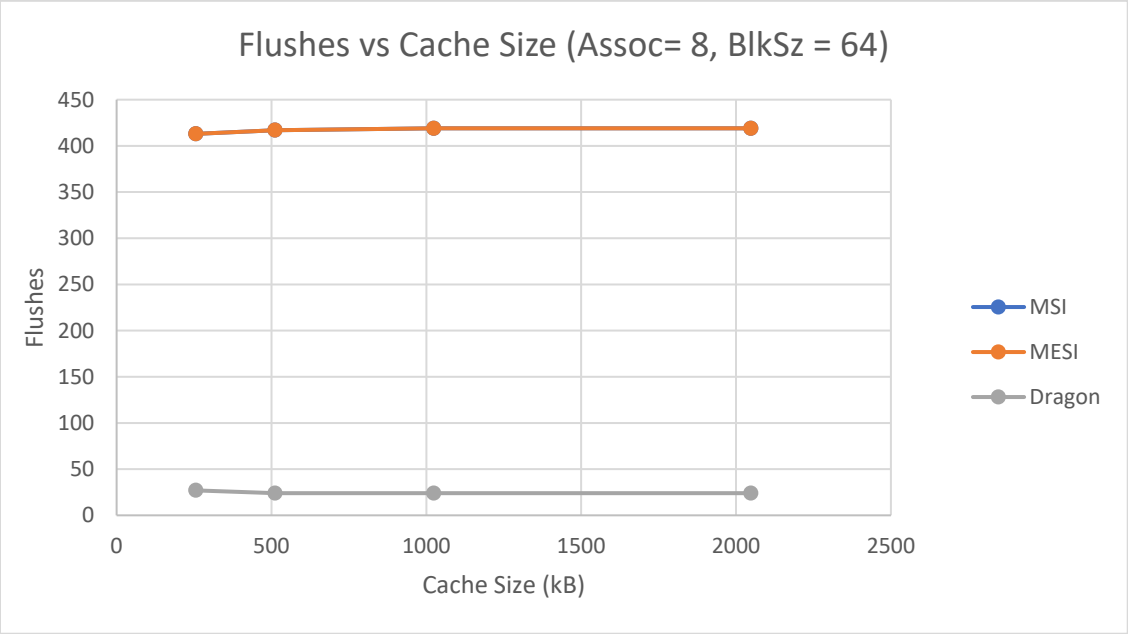
GRAPH #5



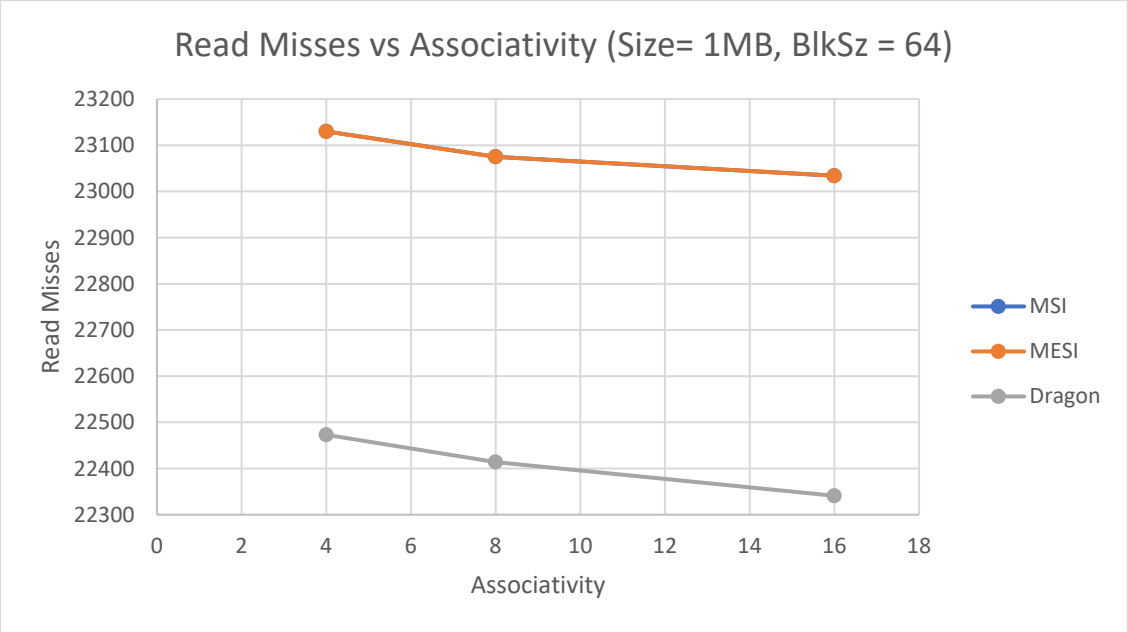
GRAPH #6



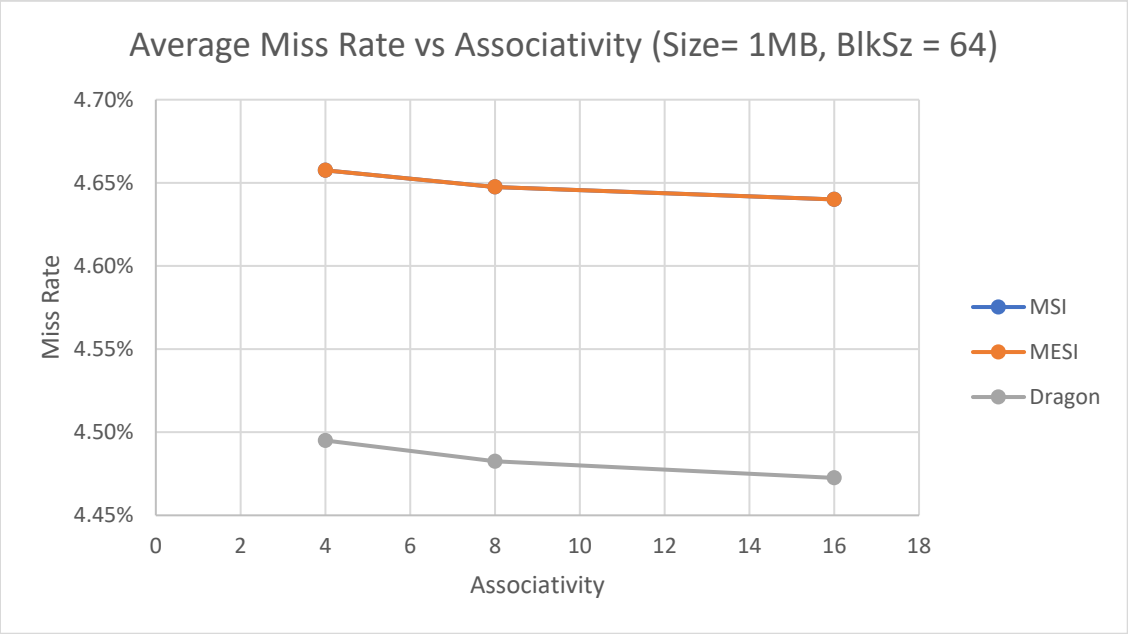
GRAPH #7



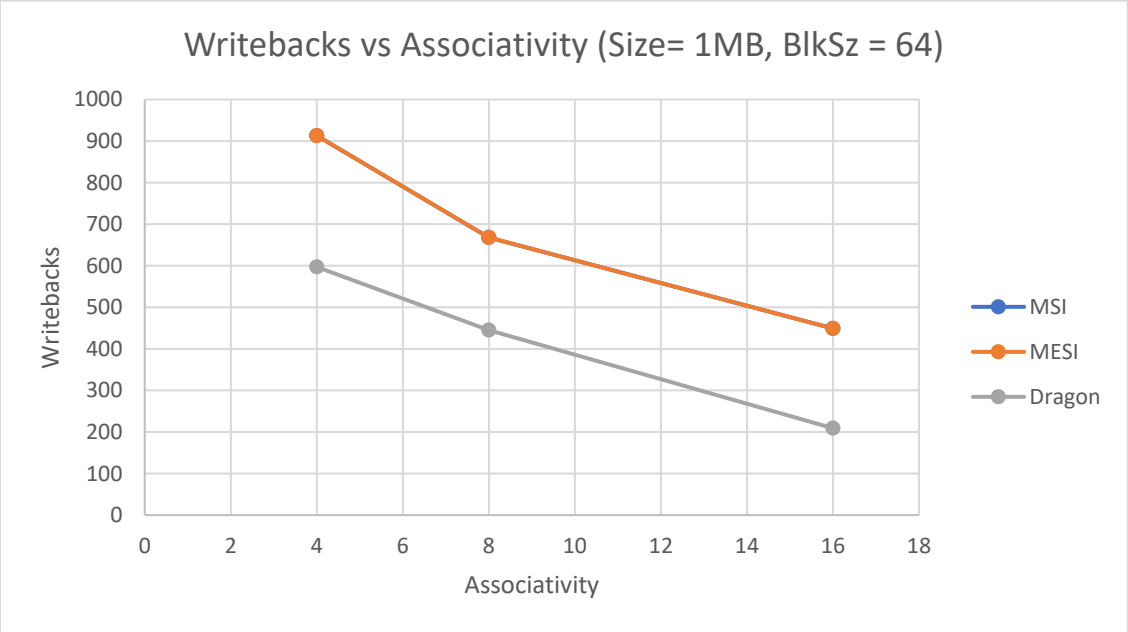
GRAPH #8



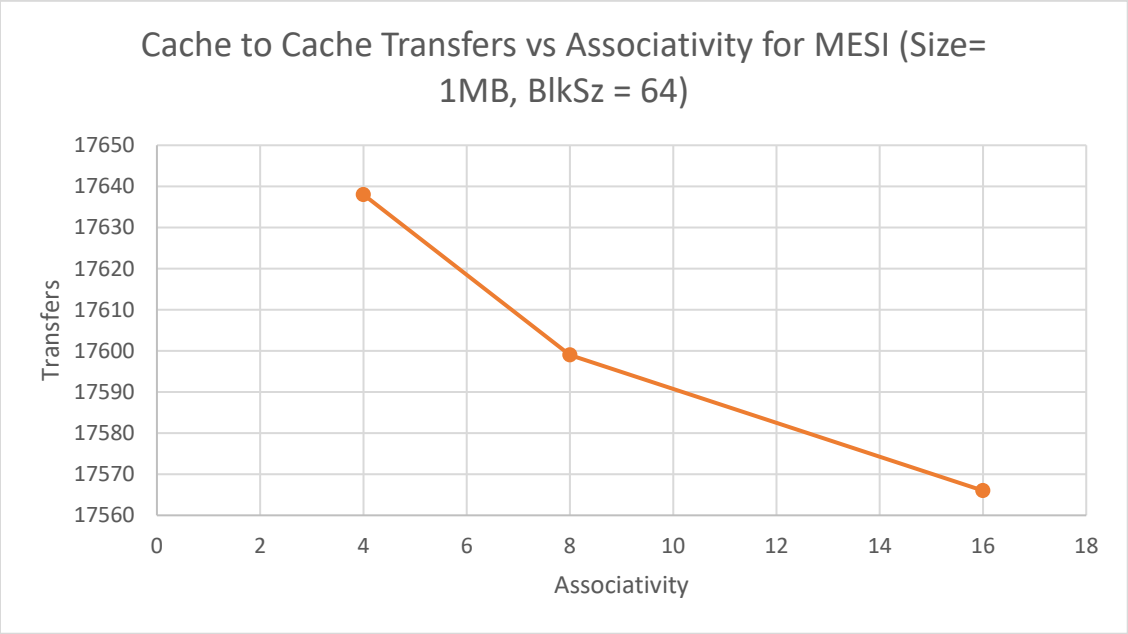
GRAPH #9



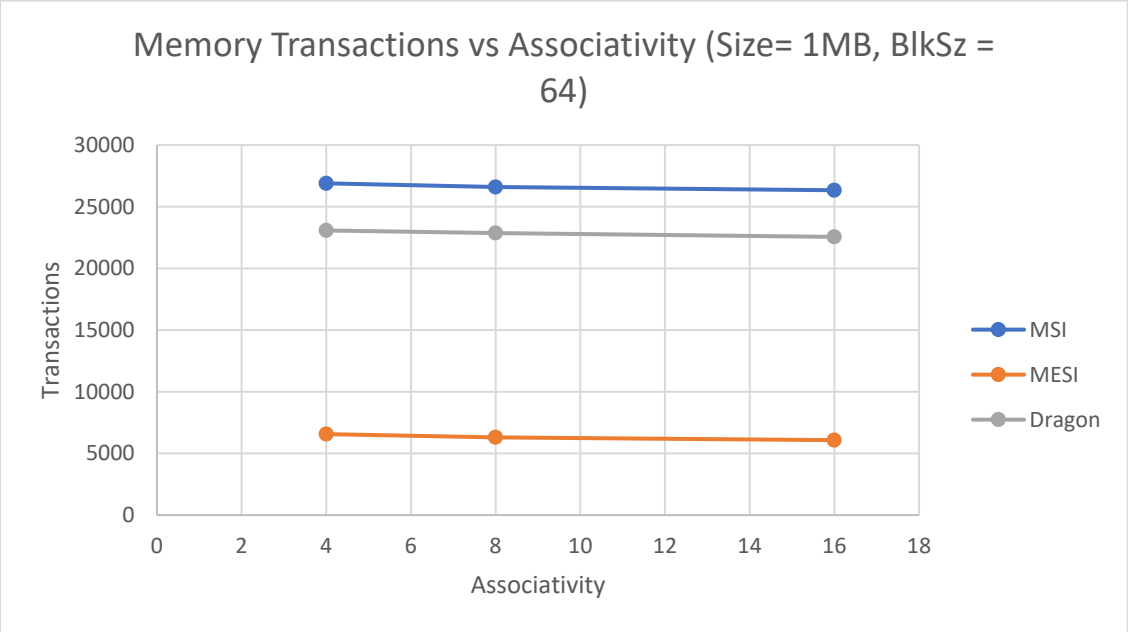
GRAPH #10



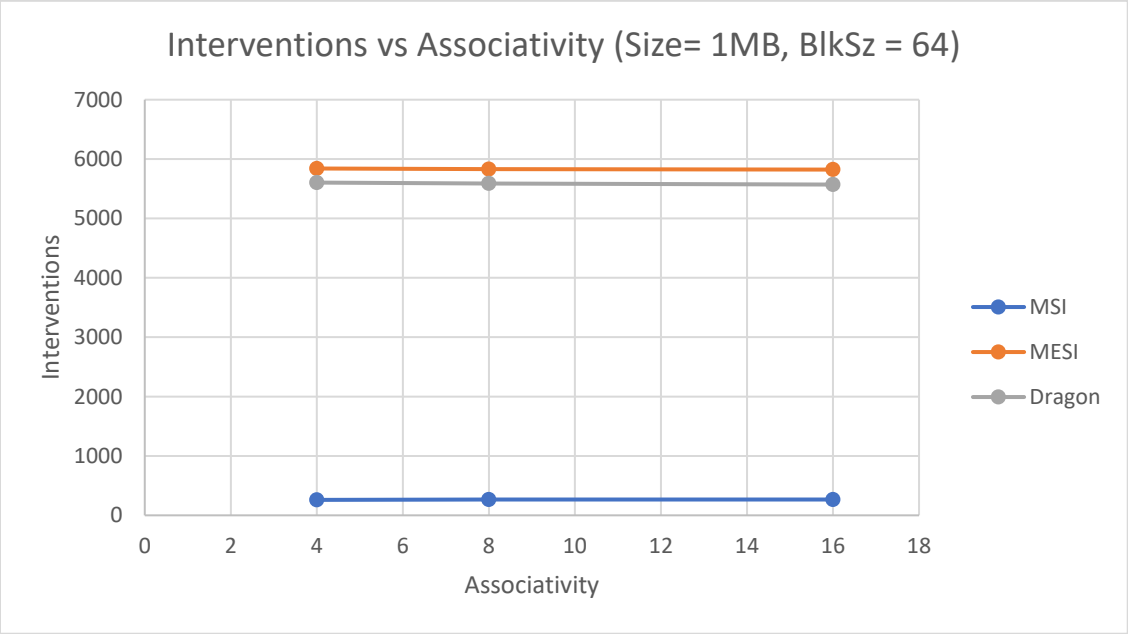
GRAPH #11



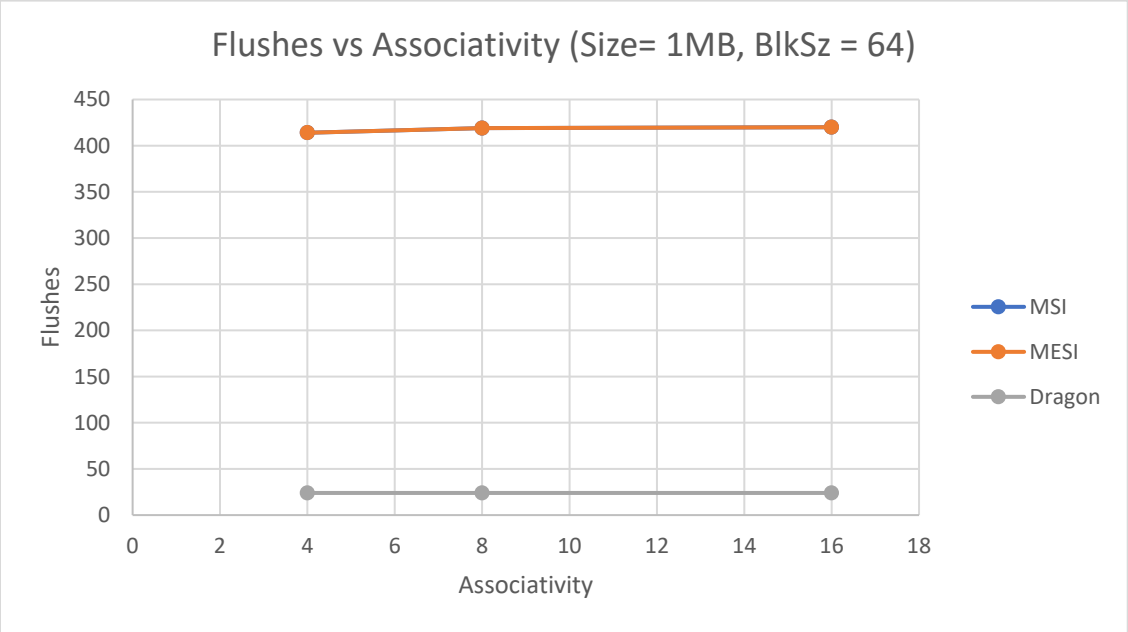
GRAPH #12



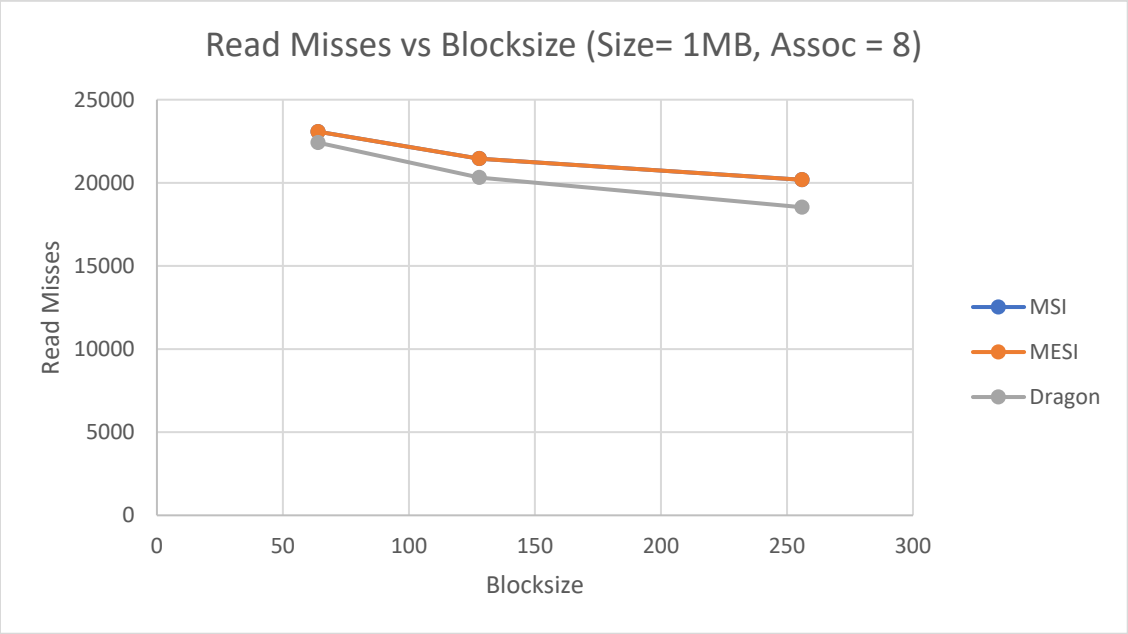
GRAPH #13



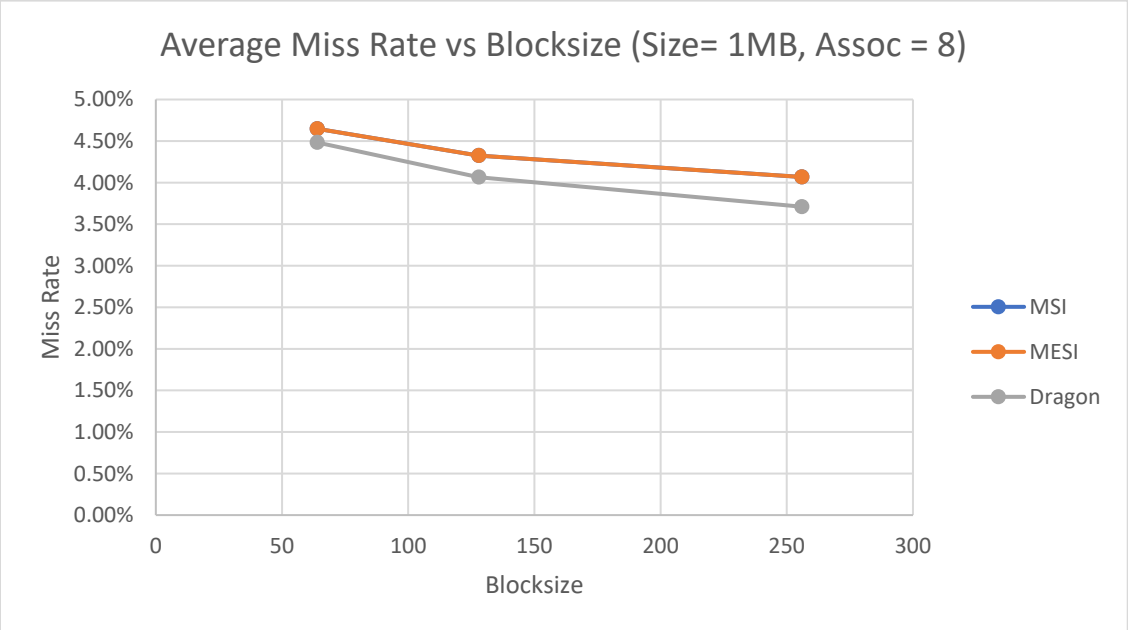
GRAPH #14



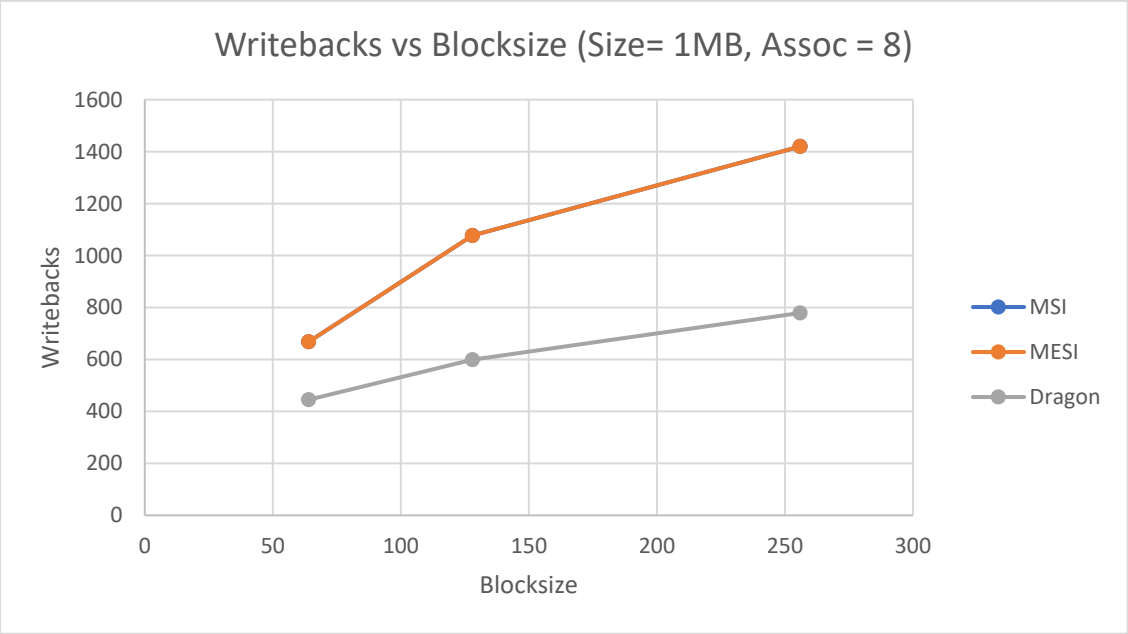
GRAPH #15



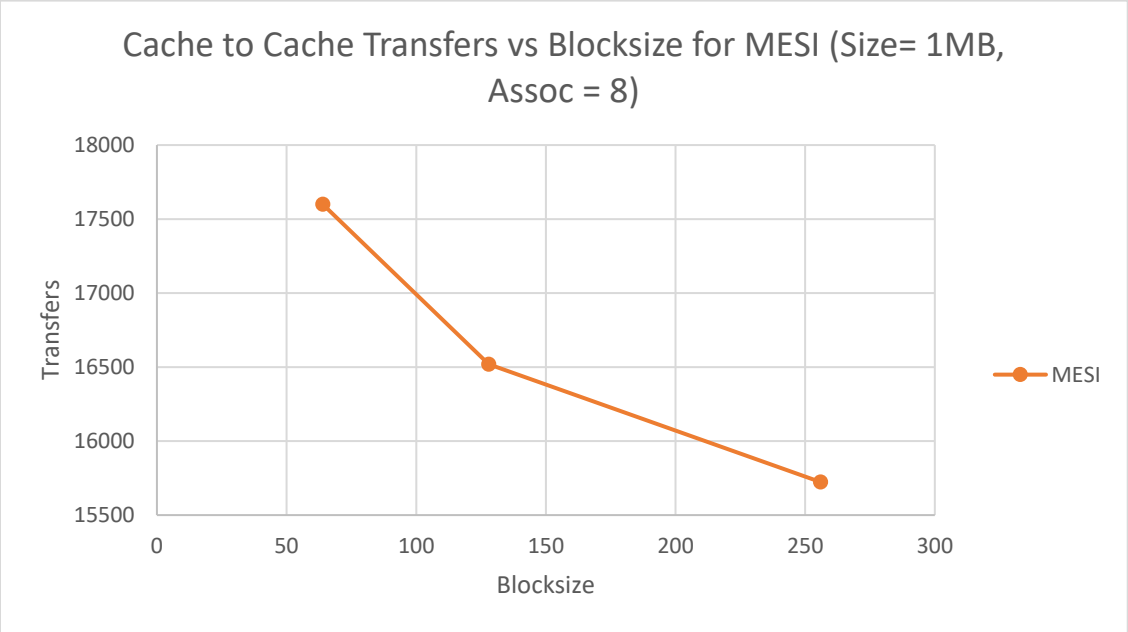
GRAPH #16



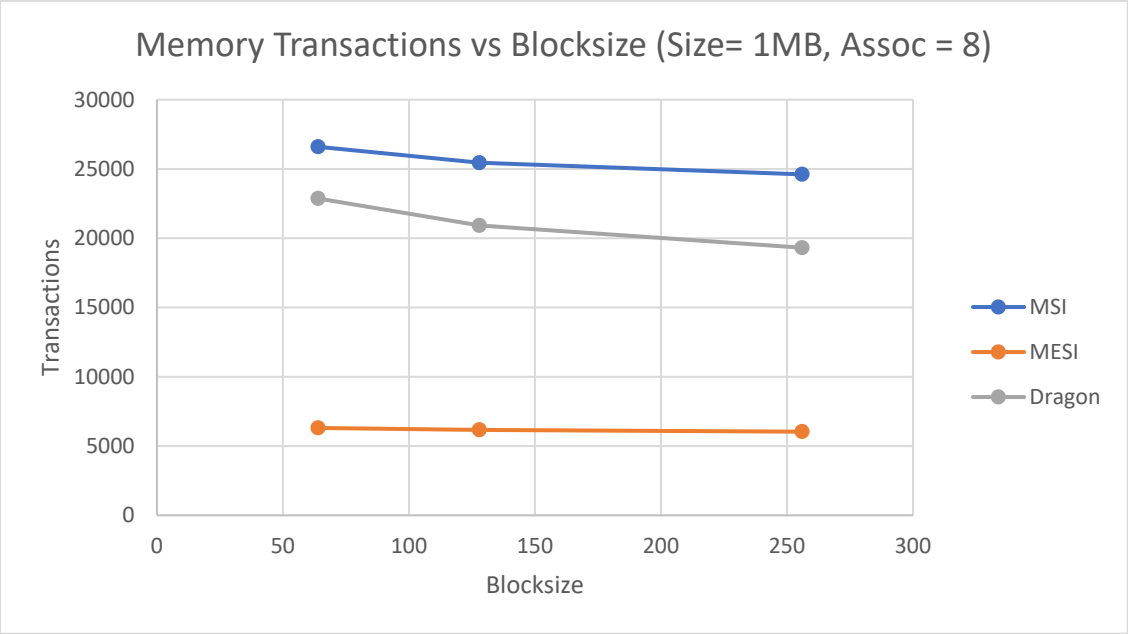
GRAPH #17



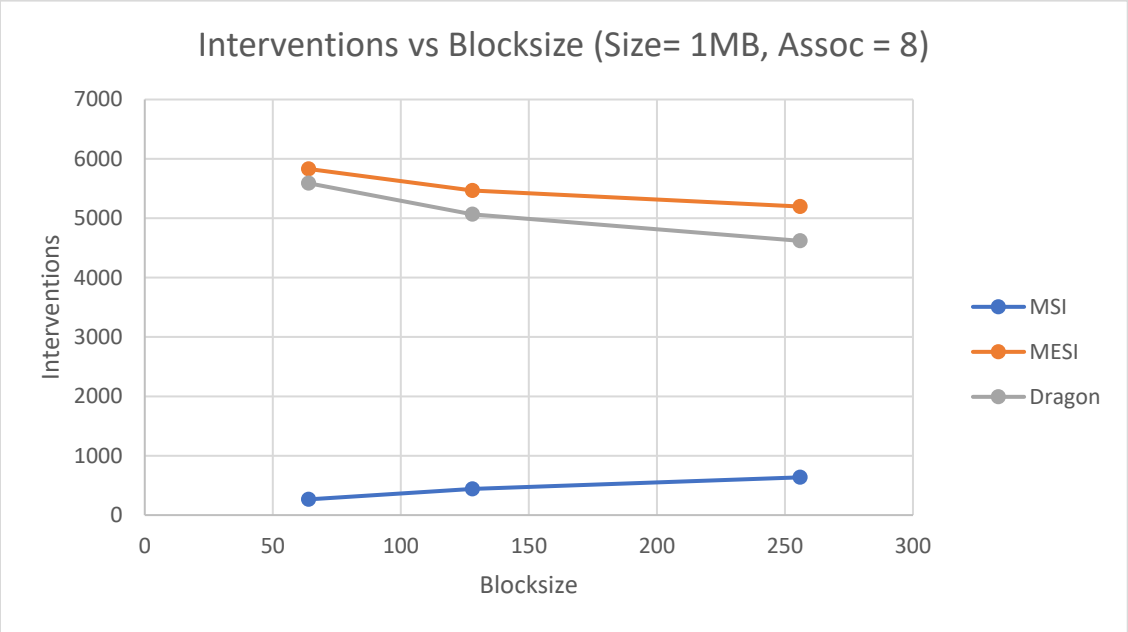
GRAPH #18



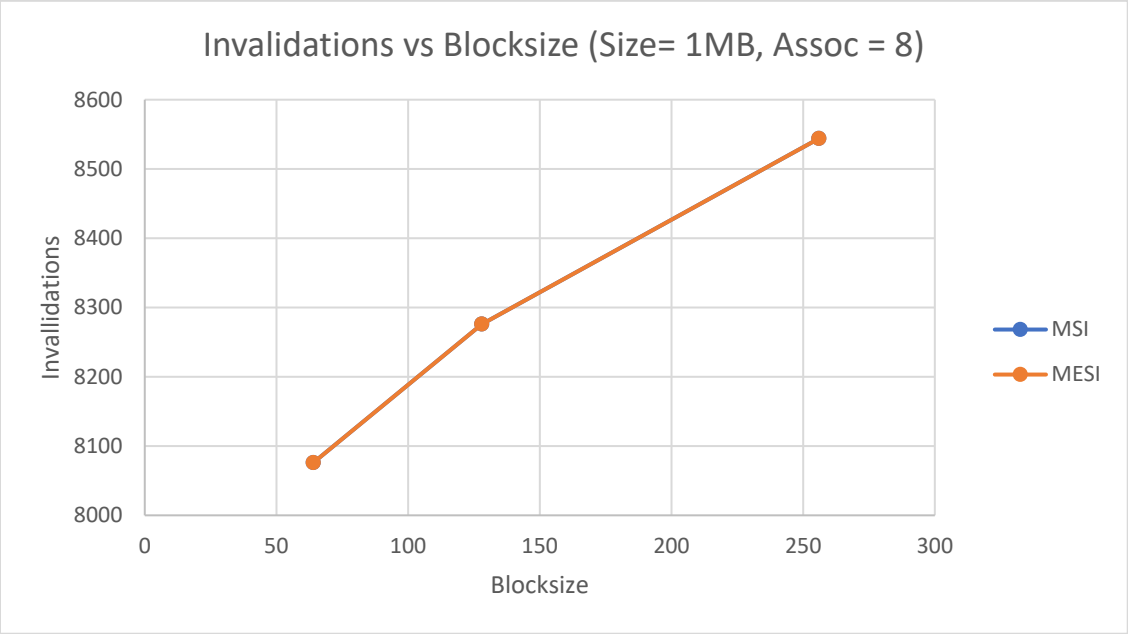
GRAPH #19



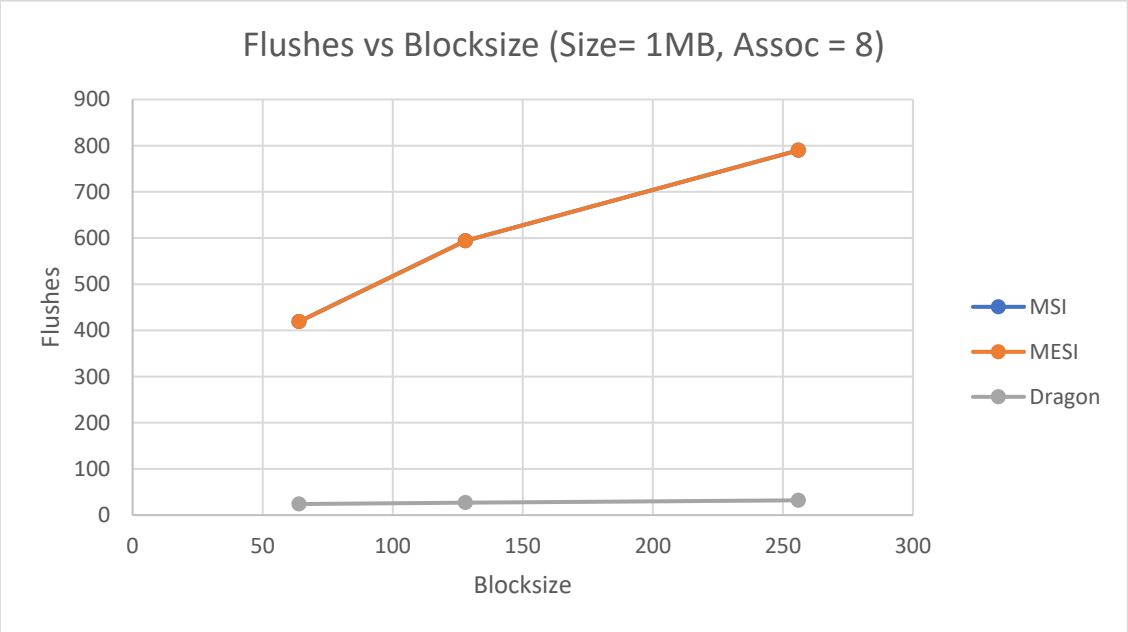
GRAPH #20



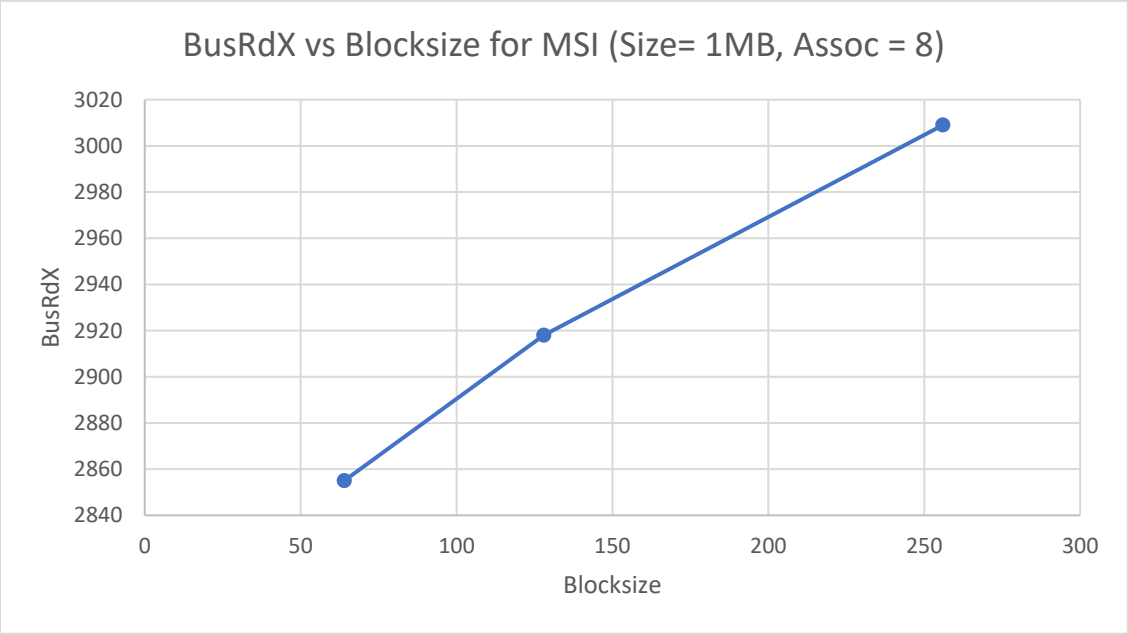
GRAPH #21



GRAPH #22



GRAPH #23


BusRdX vs Blocksize for MSI (Size= 1MB, Assoc = 8)
BusRdX
3020
3000
2980
2960
2940
2920
2900
2880
2860
2840
0
50
100
150
200
250
300
Blocksize

Conclusions:

For all of the protocols, increasing the cache size caused the miss rate to decrease. The reason for this is that increasing the cache size causes fewer evictions as the cache has more space to hold the memory blocks which are put into the cache. It's hard to say if this decrease in miss rate increases the overall performance of the cache since I am unsure how the increased size affects the access time for the cache. The miss rate also decreased with increases to the block size and associativity, but it is also unclear what the overall effect this has on the overall average access time for this simulation.

For MSI, increasing the cache size also decreased the writebacks and memory transactions. This is probably because of the decrease in evictions that should be needed for each cache. With less evictions means less reasons to need to go to the memory. Interestingly enough, the number of flushes and interventions increased as a result of increasing the cache size. I imagine this is because of the fact that the extra space leads to more caches being able to hold memory blocks at the same time, so if one cache needs to write to the value, that needs to intervene with another copy. When increasing the associativity, there was a strong increase in the amount of writebacks, interventions, flushes, and BusRdX. This is probably due to the fact that with a larger block size, there are less unique blocks that the cache can hold. With less blocks means that repeated access to memory locations which are far away from one another could lead to more evictions to keep up with the program jumping around the memory.

For all three protocols, increasing the block size seemed to have the more pronounced effect on the miss rate. This leads me to believe that the trace which was testing for this report had a lot of spatial locality that could be exploited as a result of bigger block size. If there wasn't much spatial locality to take advantage of, then increasing the block size could actually have a negative effect on the miss rate. This is called cache pollution.

For the MESI protocol, there were many less memory transactions than either the Dragon or MSI protocol. This is probably a result of the fact that MESI has a state dedicated to indicate that a memory block is exclusive that one cache only. This exclusive state allows for caches which have an exclusive block to update it as necessary without checking to see if there are other caches which hold this block. Also, MESI introducing the flushopt signal which sends data from cache to cache instead of through the memory. This also explains the lower number of memory transactions than other protocols.

The Dragon protocol had a lower number of misses than MSI and MESI protocols. This is likely due to the fact that Dragon is an update-based protocol in comparison to MSI and MESI which are invalidation-based protocols. With an update-based protocol, caches which write to a block do not invalidate other copies of that block in other caches. Instead, that write is used to update the other copies held in other caches, thus keeping the block inside the cache. This means that the trace which was used to write this report utilized these blocks being kept in the cache later on by access those blocks.