REPORT
ECE 506 – Machine Problem 2

By – Ujan Sengupta
Unity ID – usengup

Test Cases and Counters used:

As mentioned in the project spec, the cache configurations that were simulated were:

Cache size: vary from 256KB, 512KB, 1MB, 2MB while keeping the cache

associativity at 8 and block size at 64B.

- Cache associativity: vary from 4-way, 8-way, and 16-way while keeping the cache

size at 1MB and block size at 64B.

- Cache block size: vary from 64B, 128B, and 256B, while keeping the cache size at

1MB and cache associativity at 8 way.

- Cache policies: use write back and write allocate policy, and use LRU replacement

policy for all cases. (both are already implemented in the provided code)

The counters that I decided to focus on were:

1) Number of writebacks
2) Number of Memory Transactions

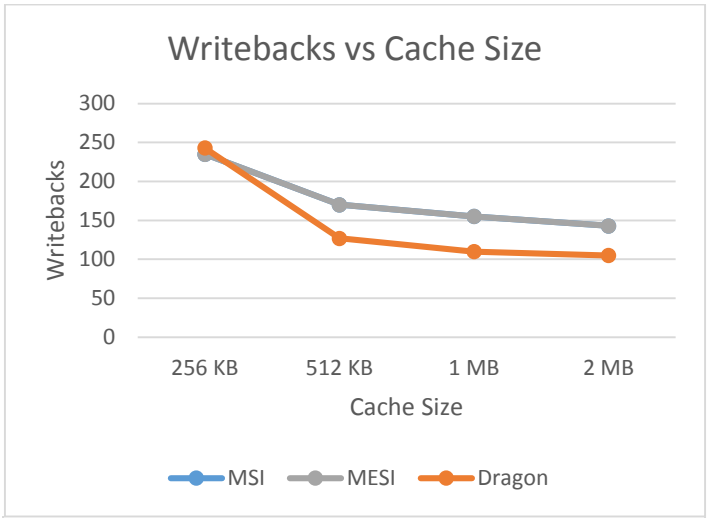
Observations:

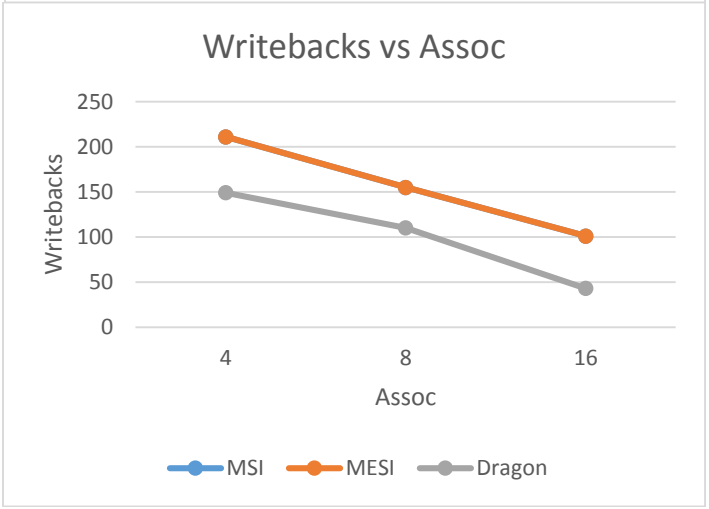
WRITEBACKS –

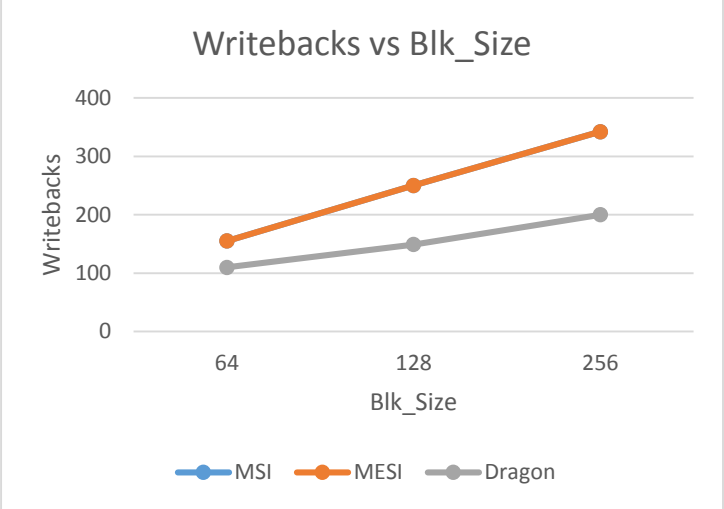
Size	MSI	MESI	Dragon
256 KB	235	235	243
512 KB	170	170	127
1 MB	155	155	110
2 MB	143	143	105

Assoc	MSI	MESI	Dragon
4	211	211	149
8	155	155	110
16	101	101	43

Blk_Size	MSI	MESI	Dragon
64	155	155	110
128	250	250	149
256	342	342	200







INFERENCE –

1) MSI and MESI have identical counts for writebacks in all three cases. This is because dirty sharing is not allowed in MSI and MESI, unlike in the Dragon protocol. Hence, for the MSI and MESI protocol, whenever a dirty block is evicted, it has to be written back to the memory. In the Dragon protocol, a dirty block needn't be written back to the memory (unless it's the owner) when it is evicted. This is also the reason that the Dragon protocol has a lesser number of writebacks as compared to MSI and MESI.

2) With an increase in Cache Size and Associativity, the number of writebacks decrease gradually. As the Cache size increases, the number of sets increases. Resultantly, more and more blocks can be fit into a single cache and that reduces the number of blocks that have to be evicted.
Also, as the associativity increases, the cache sets become more flexible in terms of accommodating blocks. Ergo, with an increased flexibility, less number of blocks need to be evicted because there is more place to accommodate newer blocks. Hence, the number of writebacks decrease.

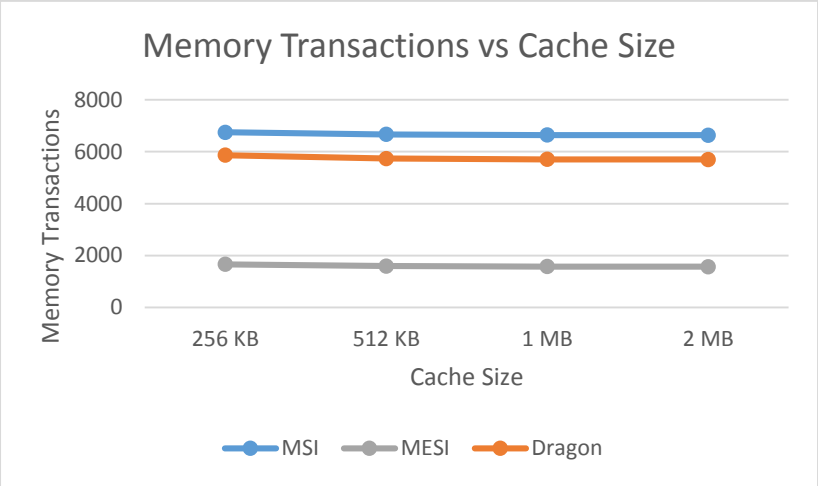
3) However, with an increase in Block Size, the number of writebacks increase. This behavior is expected because we view the cache at the block level granularity. So, the more bytes that a block contains, the more it's chances of being written to and getting classified as dirty. And since the number of sets also gradually decreases with an increase in Block Size (#Sets = Size/(Assoc. * BlockSize)), we have an increase in the number of blocks getting evicted as the Block size increases.

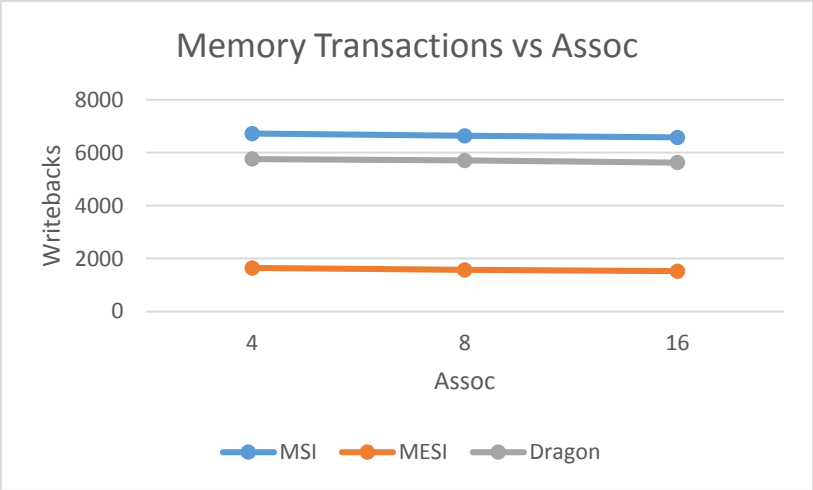
MEMORY TRANSACTIONS –

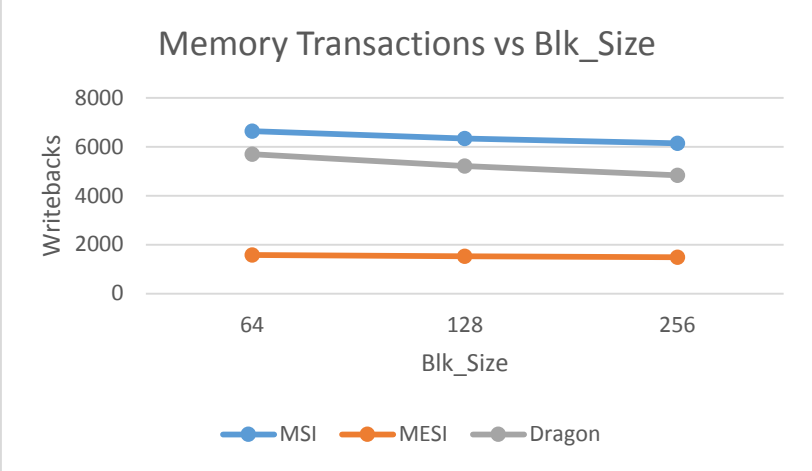
Size	MSI	MESI	Dragon
256 KB	6745	1663	5864
512 KB	6663	1594	5732
1 MB	6638	1572	5705
2 MB	6632	1568	5696

Assoc	MSI	MESI	Dragon
4	6723	1644	5760
8	6638	1572	5705
16	6577	1521	5626

Blk_Size	MSI	MESI	Dragon
64	6638	1572	5705
128	6344	1530	5217
256	6137	1487	4829







INFERENCE –

There are two distinct inferences that we can make looking at the above graphs:

1) The first inference is that for an increase in Cache Size, Associativity and Block size, the number of memory transactions decrease very gradually. The reason for this decrease is that as the caches get larger in size and more flexible in associativity, the number of times that a block has to be written back to memory decreases. Also, as the number of evictions decrease, more blocks are present in the cache at any given time and the miss rate decreases proportionately. Hence, the number of memory transactions decrease as we need to fetch less blocks from the memory.

2) The second marked inference is that the number of memory transactions incurred by the MSI and Dragon protocols are significantly higher than for the MESI protocol. This is because the MESI protocol has cache-to-cache transfers enabled, whereas, for the purpose of MP2, MSI and Dragon protocols do not. As a result, many of the transfers that would've incurred a bus transaction for the MESI protocol are handled by cache-to-cache transfers.