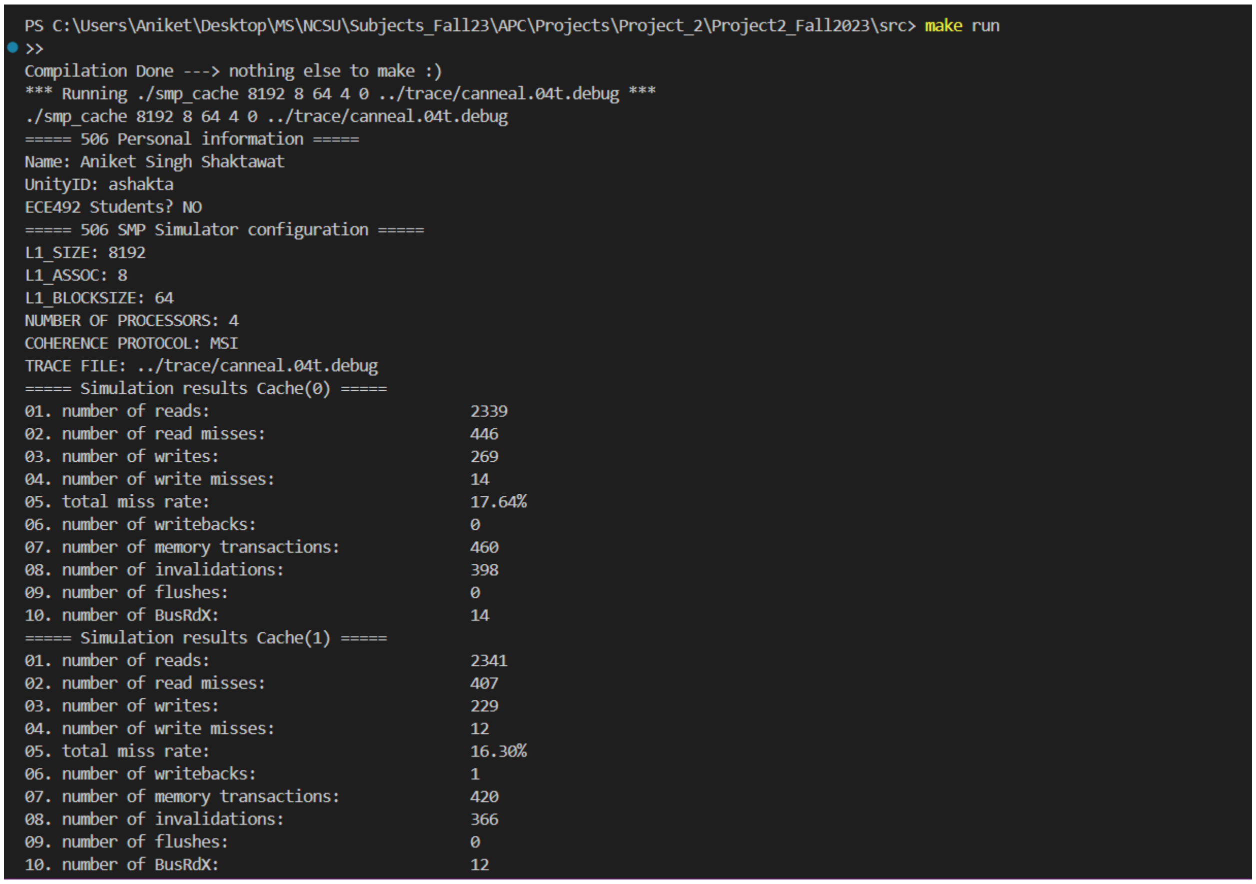
ECE/CSC 406/506: Architecture of Parallel Computers
Project 2. Coherence Protocols

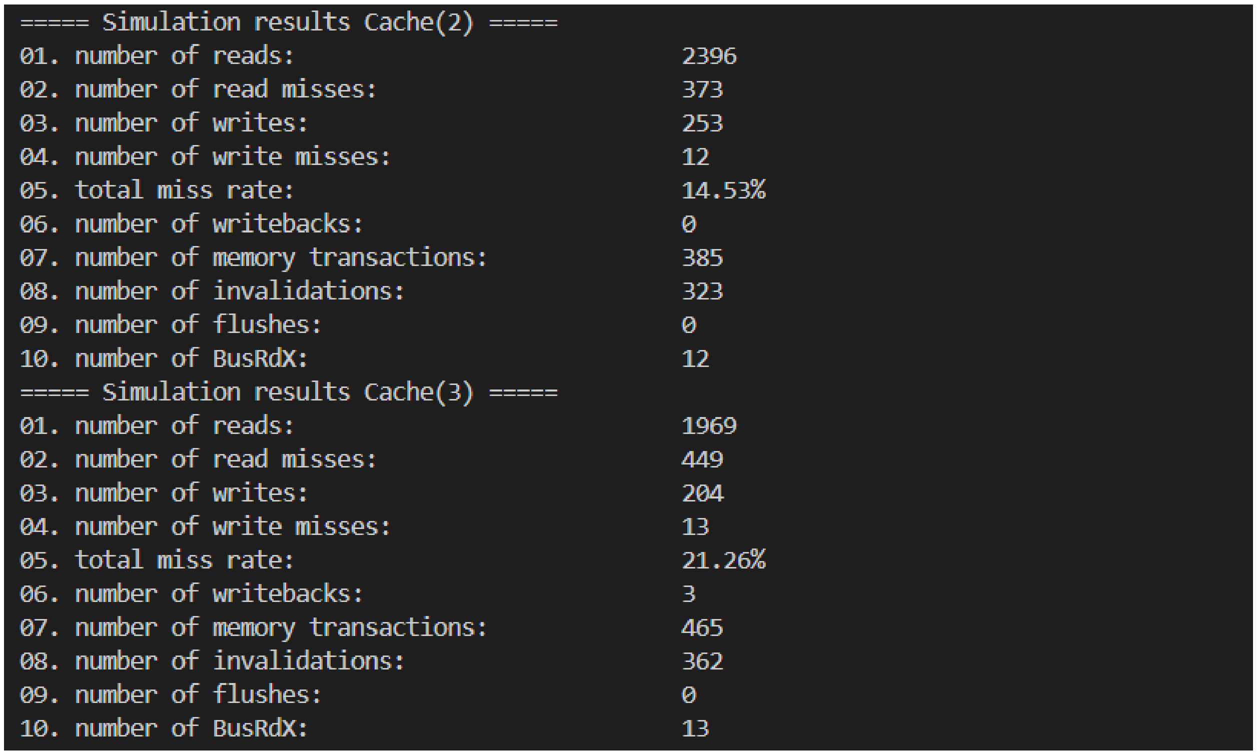
Submitted by: Aniket Singh Shaktawat
Unity ID: ashakta

Project Report

Part1 a) (40 pt) Implement Modified MSI protocol as seen in the diagram.

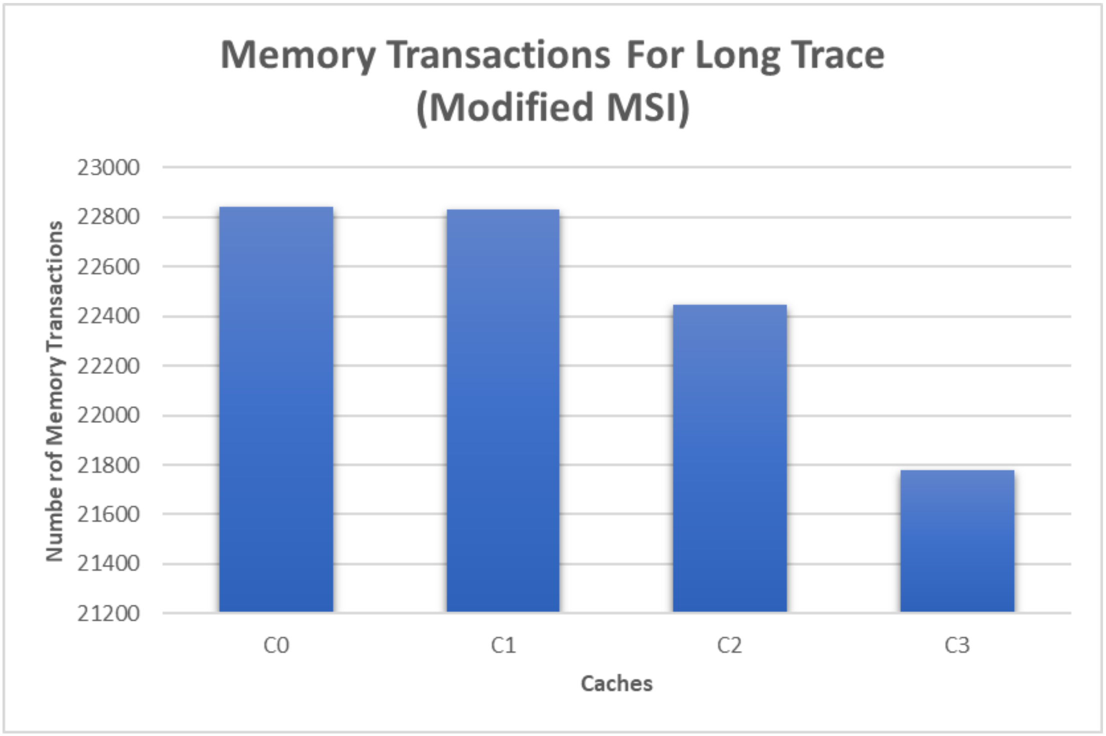


```
PS C:\Users\Aniket\Desktop\MS\NCSU\Subjects_Fall23\APC\Projects\Project_2\Project2_Fall2023\src> make run
>>
Compilation Done ---> nothing else to make :)
*** Running ./smp_cache 8192 8 64 4 0 ../trace/canneal.04t.debug ***
./smp_cache 8192 8 64 4 0 ../trace/canneal.04t.debug
===== 506 Personal information =====
Name: Aniket Singh Shaktawat
UnityID: ashakta
ECE492 Students? NO
===== 506 SMP Simulator configuration =====
L1_SIZE: 8192
L1_ASSOC: 8
L1_BLOCKSIZE: 64
NUMBER OF PROCESSORS: 4
COHERENCE PROTOCOL: MSI
TRACE FILE: ../trace/canneal.04t.debug
===== Simulation results Cache(0) =====
01. number of reads:                            2339
02. number of read misses:                      446
03. number of writes:                           269
04. number of write misses:                     14
05. total miss rate:                            17.64%
06. number of writebacks:                       0
07. number of memory transactions:              460
08. number of invalidations:                    398
09. number of flushes:                          0
10. number of BusRdX:                           14
===== Simulation results Cache(1) =====
01. number of reads:                            2341
02. number of read misses:                      407
03. number of writes:                           229
04. number of write misses:                     12
05. total miss rate:                            16.30%
06. number of writebacks:                       1
07. number of memory transactions:              420
08. number of invalidations:                    366
09. number of flushes:                          0
10. number of BusRdX:                           12
```

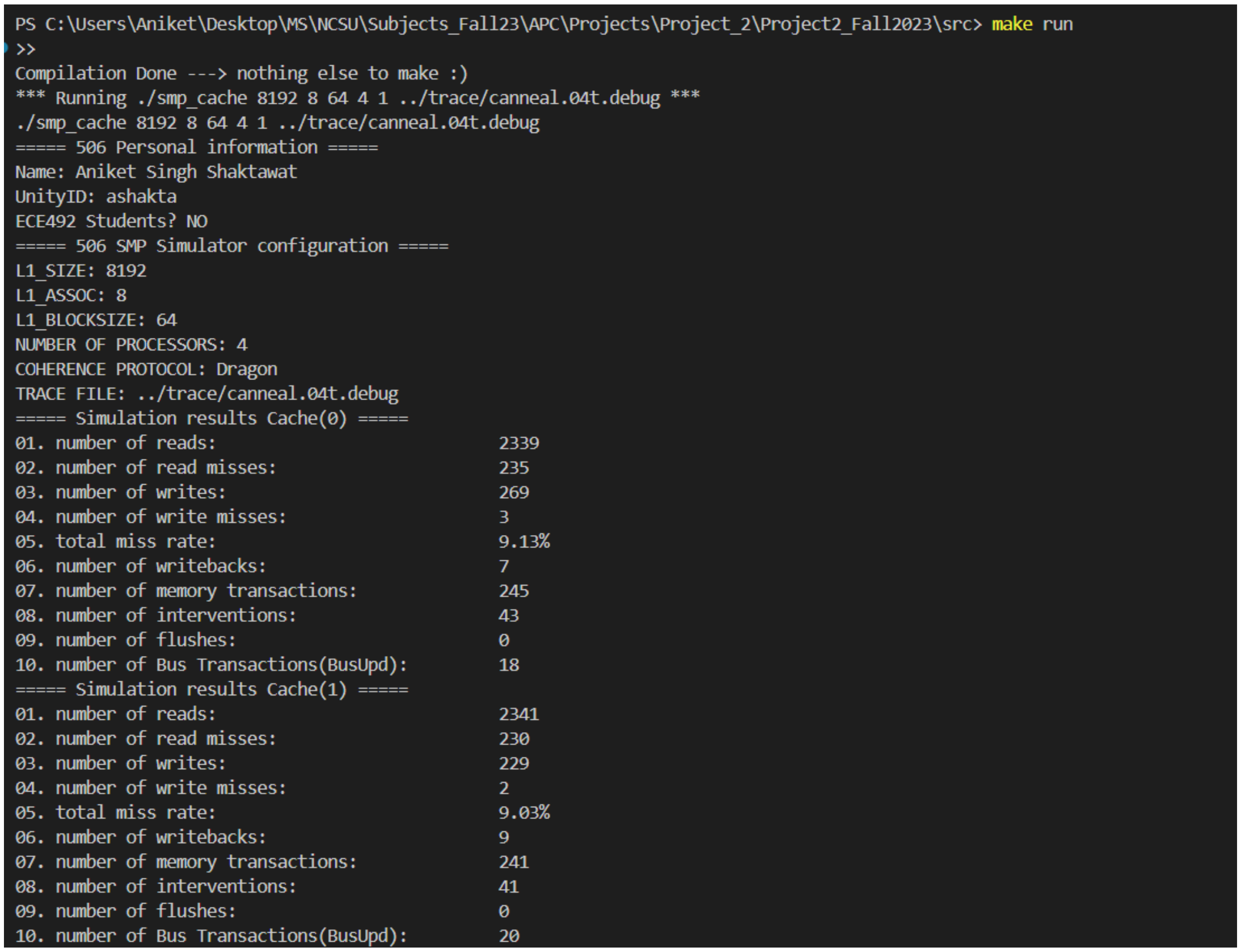


```
===== Simulation results Cache(2) =====
01. number of reads:                        2396
02. number of read misses:                  373
03. number of writes:                       253
04. number of write misses:                 12
05. total miss rate:                        14.53%
06. number of writebacks:                   0
07. number of memory transactions:          385
08. number of invalidations:                323
09. number of flushes:                      0
10. number of BusRdX:                       12
===== Simulation results Cache(3) =====
01. number of reads:                        1969
02. number of read misses:                  449
03. number of writes:                       204
04. number of write misses:                 13
05. total miss rate:                        21.26%
06. number of writebacks:                   3
07. number of memory transactions:          465
08. number of invalidations:                362
09. number of flushes:                      0
10. number of BusRdX:                       13
```

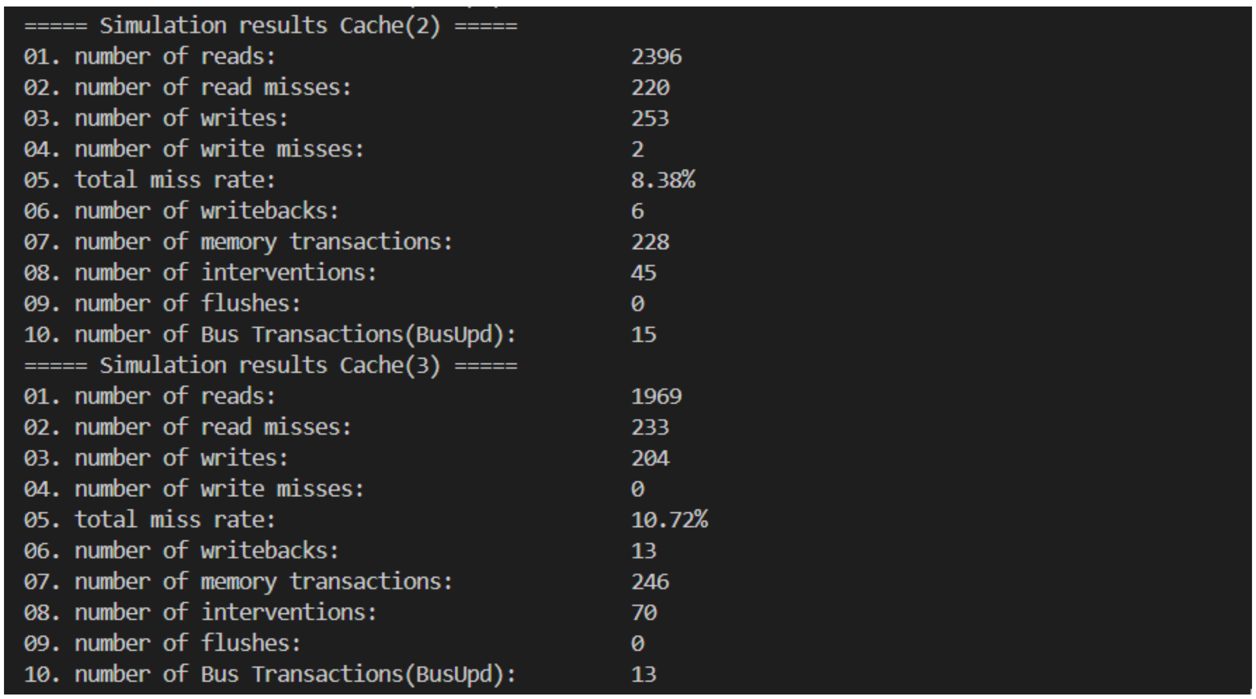
b) (10 pt) Plot the number of memory transactions for the long trace.



Part2 a) (40 pt) Implement Dragon protocol as seen in the diagram.



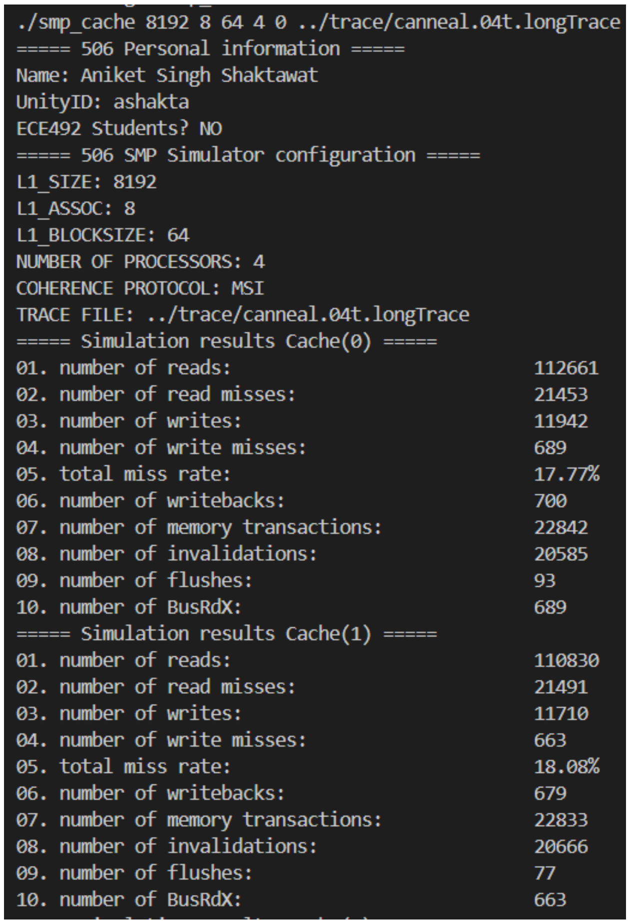
```
PS C:\Users\Aniket\Desktop\MS\NCSU\Subjects_Fall23\APC\Projects\Project_2\Project2_Fall2023\src> make run
>>
Compilation Done ---> nothing else to make :)
*** Running ./smp_cache 8192 8 64 4 1 ../trace/canneal.04t.debug ***
./smp_cache 8192 8 64 4 1 ../trace/canneal.04t.debug
===== 506 Personal information =====
Name: Aniket Singh Shaktawat
UnityID: ashakta
ECE492 Students? NO
===== 506 SMP Simulator configuration =====
L1_SIZE: 8192
L1_ASSOC: 8
L1_BLOCKSIZE: 64
NUMBER OF PROCESSORS: 4
COHERENCE PROTOCOL: Dragon
TRACE FILE: ../trace/canneal.04t.debug
===== Simulation results Cache(0) =====
01. number of reads:                            2339
02. number of read misses:                      235
03. number of writes:                           269
04. number of write misses:                     3
05. total miss rate:                            9.13%
06. number of writebacks:                       7
07. number of memory transactions:              245
08. number of interventions:                    43
09. number of flushes:                          0
10. number of Bus Transactions(BusUpd):         18
===== Simulation results Cache(1) =====
01. number of reads:                            2341
02. number of read misses:                      230
03. number of writes:                           229
04. number of write misses:                     2
05. total miss rate:                            9.03%
06. number of writebacks:                       9
07. number of memory transactions:              241
08. number of interventions:                    41
09. number of flushes:                          0
10. number of Bus Transactions(BusUpd):         20
```



```
===== Simulation results Cache(2) =====
01. number of reads:                      2396
02. number of read misses:                220
03. number of writes:                     253
04. number of write misses:               2
05. total miss rate:                      8.38%
06. number of writebacks:                 6
07. number of memory transactions:        228
08. number of interventions:              45
09. number of flushes:                    0
10. number of Bus Transactions(BusUpd):   15
===== Simulation results Cache(3) =====
01. number of reads:                      1969
02. number of read misses:                233
03. number of writes:                     204
04. number of write misses:               0
05. total miss rate:                      10.72%
06. number of writebacks:                 13
07. number of memory transactions:        246
08. number of interventions:              70
09. number of flushes:                    0
10. number of Bus Transactions(BusUpd):   13
```

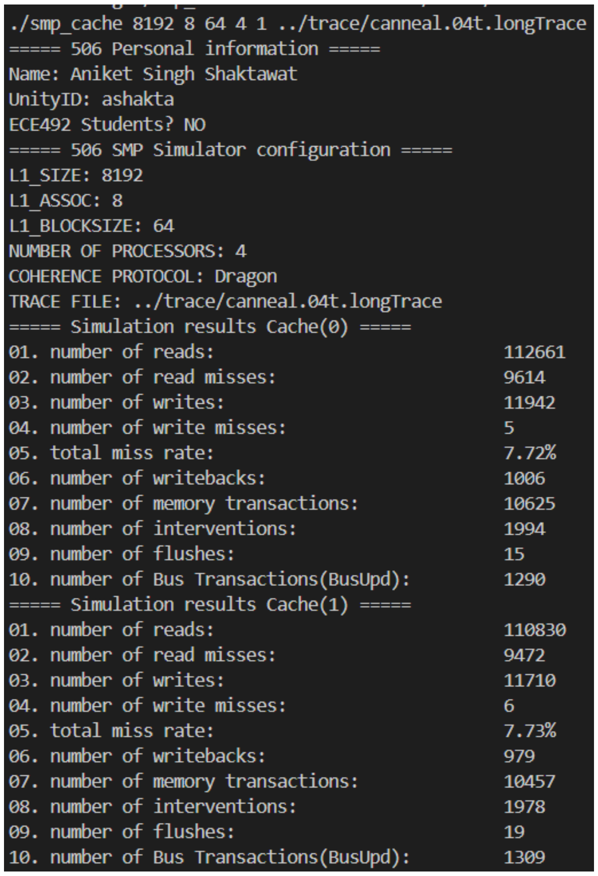
Part2 b) (10 pt) Compare Modified MSI optimization and Dragon protocols. Explain the results briefly.

Modified MSI for Long Trace

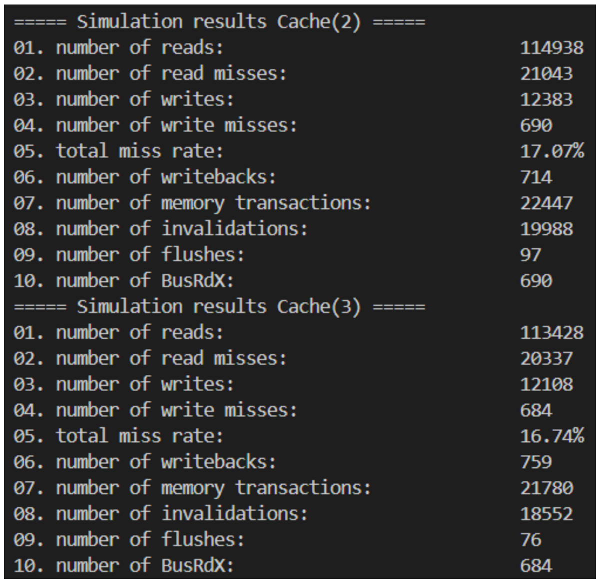


```
./smp_cache 8192 8 64 4 0 ../trace/canneal.04t.longTrace
===== 506 Personal information =====
Name: Aniket Singh Shaktawat
UnityID: ashakta
ECE492 Students? NO
===== 506 SMP Simulator configuration =====
L1_SIZE: 8192
L1_ASSOC: 8
L1_BLOCKSIZE: 64
NUMBER OF PROCESSORS: 4
COHERENCE PROTOCOL: MSI
TRACE FILE: ../trace/canneal.04t.longTrace
===== Simulation results Cache(0) =====
01. number of reads:                            112661
02. number of read misses:                      21453
03. number of writes:                           11942
04. number of write misses:                     689
05. total miss rate:                            17.77%
06. number of writebacks:                       700
07. number of memory transactions:              22842
08. number of invalidations:                    20585
09. number of flushes:                          93
10. number of BusRdX:                           689
===== Simulation results Cache(1) =====
01. number of reads:                            110830
02. number of read misses:                      21491
03. number of writes:                           11710
04. number of write misses:                     663
05. total miss rate:                            18.08%
06. number of writebacks:                       679
07. number of memory transactions:              22833
08. number of invalidations:                    20666
09. number of flushes:                          77
10. number of BusRdX:                           663
```

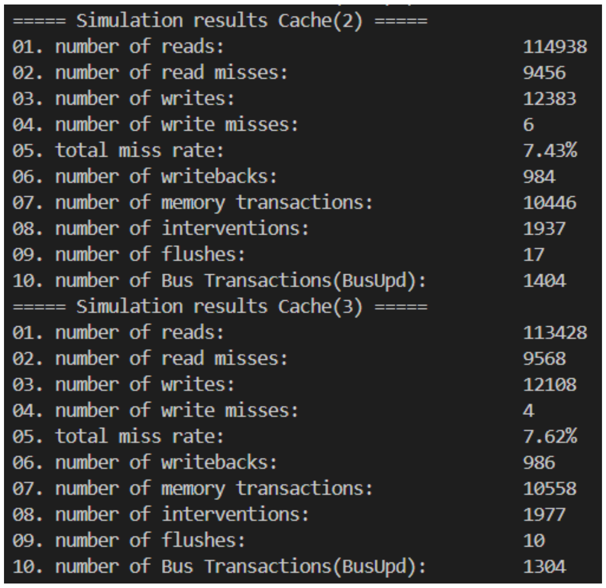
Dragon for Long Trace



```
./smp_cache 8192 8 64 4 1 ../trace/canneal.04t.longTrace
===== 506 Personal information =====
Name: Aniket Singh Shaktawat
UnityID: ashakta
ECE492 Students? NO
===== 506 SMP Simulator configuration =====
L1_SIZE: 8192
L1_ASSOC: 8
L1_BLOCKSIZE: 64
NUMBER OF PROCESSORS: 4
COHERENCE PROTOCOL: Dragon
TRACE FILE: ../trace/canneal.04t.longTrace
===== Simulation results Cache(0) =====
01. number of reads:                            112661
02. number of read misses:                      9614
03. number of writes:                           11942
04. number of write misses:                     5
05. total miss rate:                            7.72%
06. number of writebacks:                       1006
07. number of memory transactions:              10625
08. number of interventions:                    1994
09. number of flushes:                          15
10. number of Bus Transactions(BusUpd):         1290
===== Simulation results Cache(1) =====
01. number of reads:                            110830
02. number of read misses:                      9472
03. number of writes:                           11710
04. number of write misses:                     6
05. total miss rate:                            7.73%
06. number of writebacks:                       979
07. number of memory transactions:              10457
08. number of interventions:                    1978
09. number of flushes:                          19
10. number of Bus Transactions(BusUpd):         1309
```



```
===== Simulation results Cache(2) =====
01. number of reads:                            114938
02. number of read misses:                      21043
03. number of writes:                           12383
04. number of write misses:                     690
05. total miss rate:                            17.07%
06. number of writebacks:                       714
07. number of memory transactions:              22447
08. number of invalidations:                    19988
09. number of flushes:                          97
10. number of BusRdX:                           690
===== Simulation results Cache(3) =====
01. number of reads:                            113428
02. number of read misses:                      20337
03. number of writes:                           12108
04. number of write misses:                     684
05. total miss rate:                            16.74%
06. number of writebacks:                       759
07. number of memory transactions:              21780
08. number of invalidations:                    18552
09. number of flushes:                          76
10. number of BusRdX:                           684
```



```
===== Simulation results Cache(2) =====
01. number of reads:                            114938
02. number of read misses:                      9456
03. number of writes:                           12383
04. number of write misses:                     6
05. total miss rate:                            7.43%
06. number of writebacks:                       984
07. number of memory transactions:              10446
08. number of interventions:                    1937
09. number of flushes:                          17
10. number of Bus Transactions(BusUpd):         1404
===== Simulation results Cache(3) =====
01. number of reads:                            113428
02. number of read misses:                      9568
03. number of writes:                           12108
04. number of write misses:                     4
05. total miss rate:                            7.62%
06. number of writebacks:                       986
07. number of memory transactions:              10558
08. number of interventions:                    1977
09. number of flushes:                          10
10. number of Bus Transactions(BusUpd):         1304
```

The explanation of the results is as follows:

- As the input file is the same for both protocols, the number of reads and writes are the same in both cases.

- **Read Misses:** Read Misses are the number of read operations that result in cache misses. It reflects how often the processor had to fetch data from main memory or other caches(In the case of dragon protocol) due to cache misses. The number of read misses is significantly less in the Dragon protocol because the Dragon protocol incurs fewer invalidations when a processor writes to a shared data block as it is an Update-based Protocol. This means that other caches are less frequently forced to invalidate their copies, reducing the chances of read misses.

- **Write Misses:** It refers to the number of write operations requested by a processor when a cache miss happens. The number of Write Misses is significantly less in Dragon Protocol because The Dragon Protocol has optimized mechanisms for propagating write updates to other caches. This reduces the latency associated with write operations and minimizes the likelihood of other caches experiencing write misses. Dragon employs advanced write-back strategies, ensuring that modified data is efficiently communicated to other caches. This reduces the need for subsequent write misses when another processor accesses the same data.

- **Total Miss Rate:** It is less in Dragon protocol due to the following reason
  Total Miss Rate = (Read Misses+Write Misses)*100 / (Reads+Writes))

  The Dragon protocol's coherence mechanisms are designed to minimize contentions and conflicts between caches. By efficiently coordinating access to shared data, the protocol reduces the likelihood of read and write misses, contributing to a lower total miss rate.

- **Number of Writebacks to main memory:** The number of writebacks is more in the Dragon protocol because the Dragon protocol has a more aggressive or proactive write propagation mechanism. It is designed to promptly update other caches and memory with modified data, leading to more writebacks when a cache line is evicted or when the data is no longer needed in the current cache, and whenever a cacheline is updated, it sends a bus update signal to update other caches and the memory.

- **Number of Memory Transactions:** The number of memory transactions is less in the Dragon protocol because it allows cache-to-cache transfer and also it has optimized mechanisms for propagating write updates to other caches. Efficient write propagation reduces the need for additional memory transactions, as modified data is communicated effectively through the coherence protocol. The Dragon protocol minimizes the number of invalidations required when a cache line is modified. By efficiently managing the coherence state and updating other caches through writebacks instead of invalidations, the protocol reduces the overall number of memory transactions.

- **Number of Flushes to the Main Memory:** The number of flushes to the main memory is significantly less in the Dragon Protocol because it proactively manages coherence by updating other caches through writebacks and interventions rather than relying on frequent flushes. This proactive approach helps maintain consistency without unnecessary writebacks to the main memory.

- **Number of BUS transactions:** These are significantly more in Dragon protocol because its proactive approach to maintaining coherence involves more frequent checking and updating of cache states through bus transactions. This contributes to a higher count of transactions on the system bus and it sends a bus update signal every time any cache updates or performs a write operation to a cacheline so that coherence can be maintained at every point in time.

Determining which protocol is better depends on the specific goals and criteria of the system and workload. Both the Modified MSI protocol and the Dragon protocol have their strengths and weaknesses, and the choice between them should align with the desired characteristics of the system.
If the goal is to minimize overall memory transactions and improve scalability, the Dragon protocol might be preferred and sometimes The Modified MSI protocol might be simpler to implement due to its less intricate coherence mechanisms and less number of bus transactions.