Project 1 Report
Cache Simulator

Lakshmi Katravulapalli

EEL 4768-0003: Computer Architecture

Date: 3/24/2024

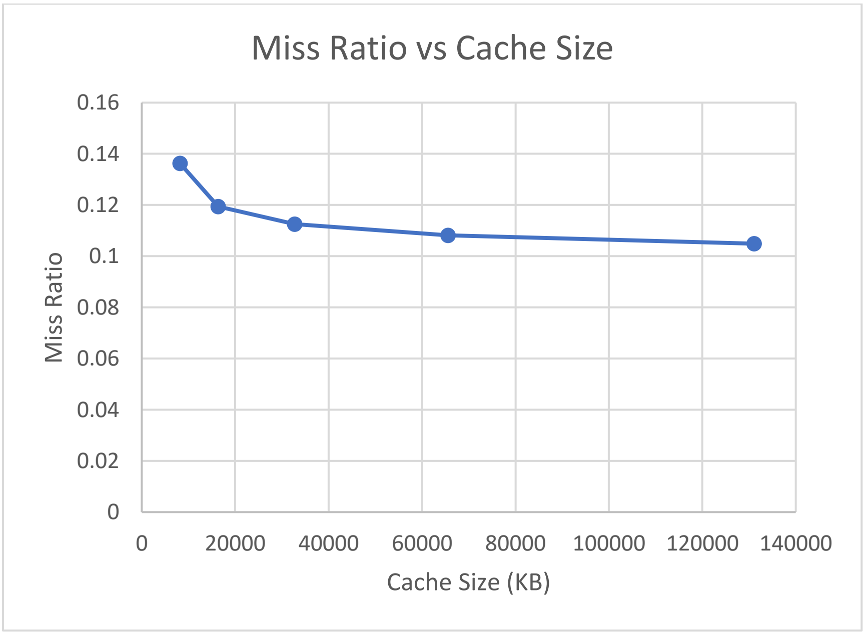
<u>Part A:</u>

In Part A, we are asked to analyze the trend of the miss ratio when we change the cache size. We are asked to use a Cache Associativity of 4, write-back policy, and the LRU replacement policy. We will vary the cache sizes from 8KB to 128KB in multiples of 2. Based on the data collected below, we noticed for the XSBench.t file that the Miss Ratio decreases as the Cache Size increases. For the MINIFE.t file, we noticed the same trend as the XSBench.t file. The Miss Ratio will decrease when the Cache Size increases because we have lower chances of conflict and therefore less chances of a Miss. A bigger cache size allows for more data storage. We notice that the XSBench.t file has a higher Miss Ratio because it needs more operations to the cache compared to the MINIFE.t File.

XSBENCH.t Table

Cache Size (KB)	Miss Ratio
8192	0.136261
16384	0.119363
32768	0.112538
65536	0.108198
131072	0.104894

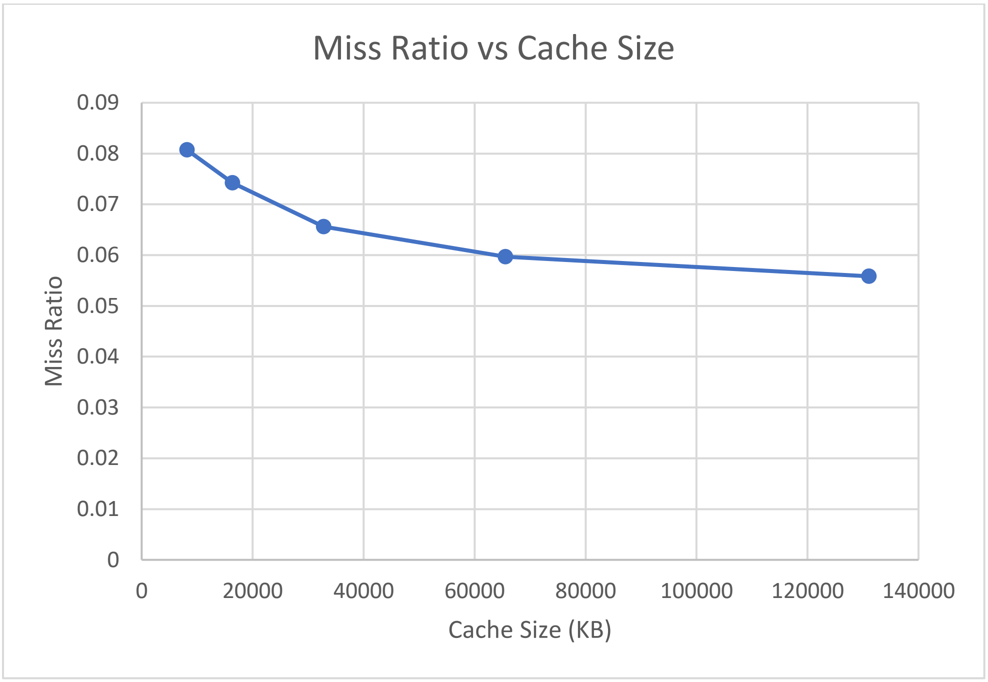
XSBENCH.t Graph



MINIFE.t Table

Cache Size (KB)	Miss Ratio
8192	0.080703
16384	0.074231
32768	0.065595
65536	0.059669
131072	0.055828

MINIFE.t graph



Part B:

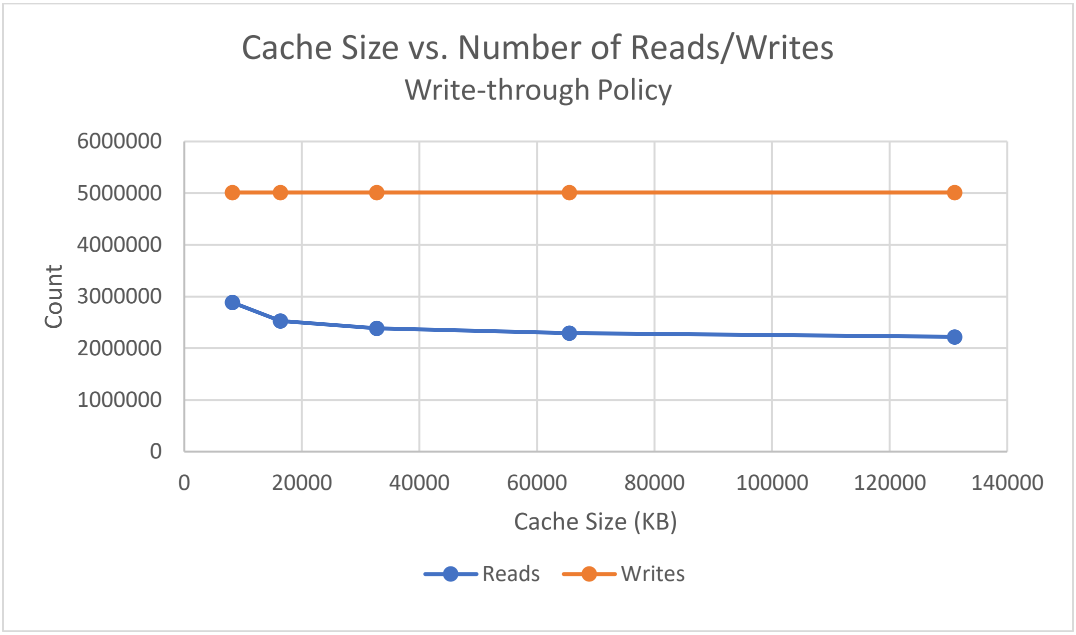
In Part B, we are asked to repeat Part A, but compare the policies of Write-Back and Write-Through while changing the Cache Sizes. We will compare the number of Read and Write operations for each policy. We will vary the Cache Size from 8K to 128KB by multiples of 2.

Based on the data presented below, we notice that for the Write-Through Policy the number of read operations decreases as the Cache Size increases. On the other hand, the number of Write Operations stays the same for every Cache Size. For the Write-Back policy, we notice the Read and Write operations decrease as the Cache Size increases. This is the case for both the XSBench.t File and the MINIFE.t file. For the XSBench.t file we notice that it has a steeper change in the Number of Write-operations for Write-Back compared to the MINIFE.t file. The XSBench.t also has more Read operations because it is a bigger file thus needing more operations. Overall, the miss ratio stays the same and both policies have similar trends except for the write operations in Write-through which stay constant.

XSBench.t table (Write-Through Policy)

Cache Size (KB)	Reads	Writes
8192	2885331	5013495
16384	2527505	5013495
32768	2383001	5013495
65536	2291098	5013495
131072	2221125	5013495

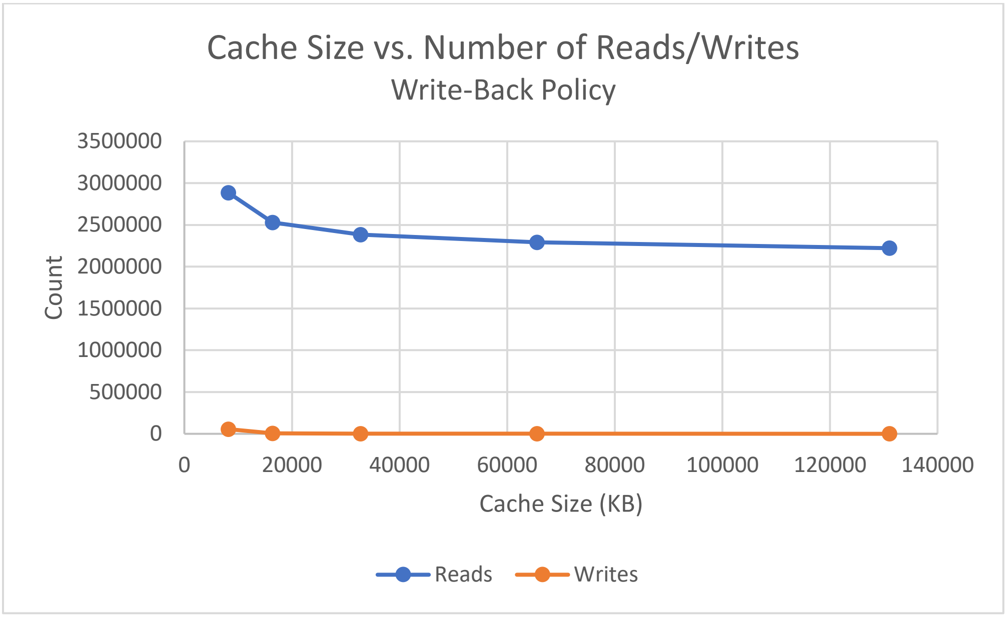
XSBench.t graph (Write-Through Policy)



XSBench.t table (Write-Back Policy)

Cache Size (KB)	Reads	Writes
8192	2885331	55680
16384	2527505	7318
32768	2383001	600
65536	2291098	69
131072	2221125	36

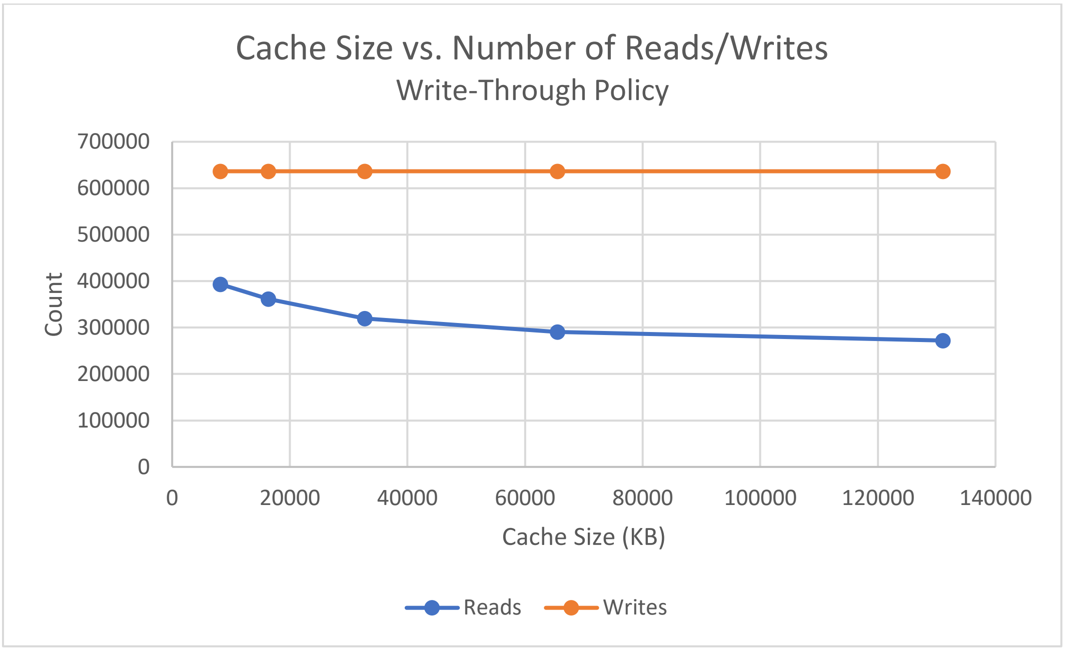
XSBench.t Graph (Write-Back Policy)



MINIFE.t table (Write-Through Policy)

Cache Size (KB)	Reads	Writes
8192	393352	636483
16384	361807	636483
32768	319713	636483
65536	290830	636483
131072	272113	636483

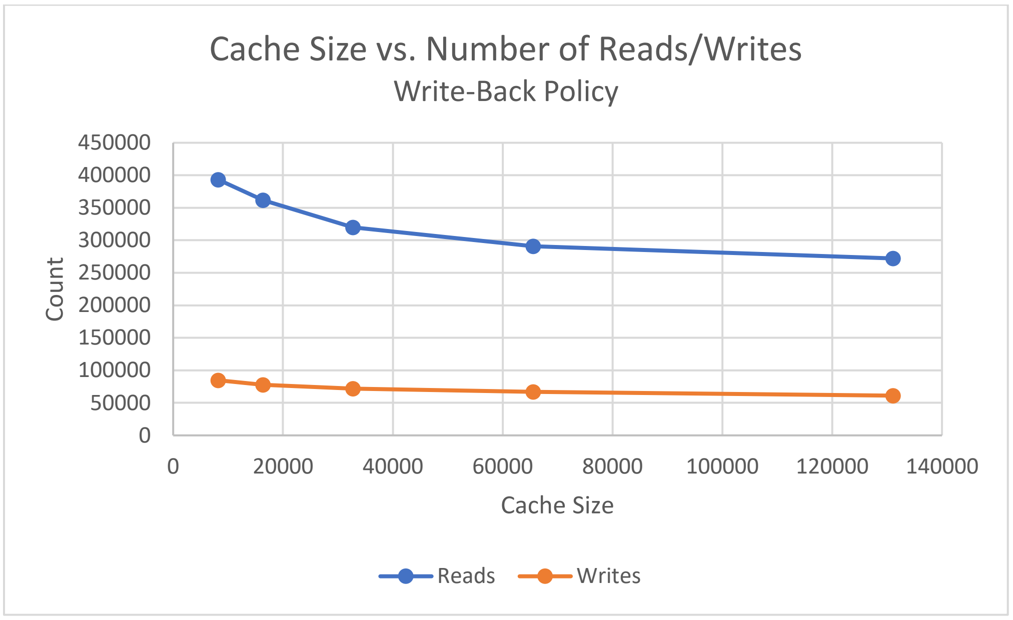
MINIFE.t graph (Write-Through Policy)



MINIFE.t table (Write-Back Policy)

Cache Size (KB)	Reads	Writes
8192	393352	84760
16384	361807	77552
32768	319713	71942
65536	290830	66927
131072	272113	61142

MINIFE.t graph (Write-Back Policy)



Part C:

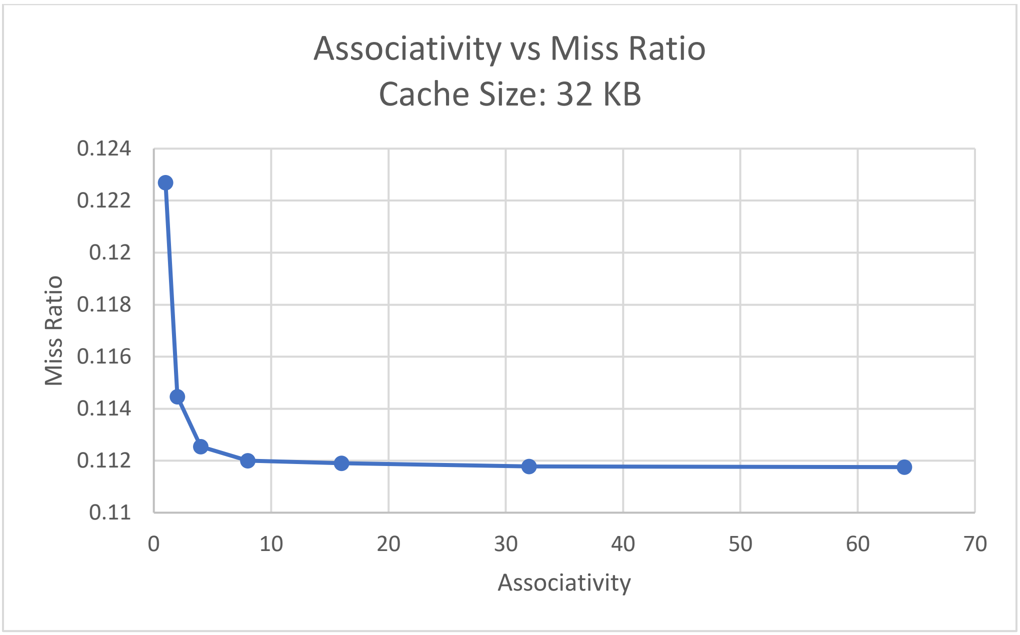
In Part C, we are asked to change the associativity, and keep the replacement policy as LRU, and cache size as 32KB. We will change the associativity from 1 to 64 in multiples of 2 and compare the behavior of the Miss Rate. Based on the data below for the XSBench.t file, we notice that when we increase the associativity the Miss Ratio decreases. Although, we notice that as we increase the size of the associativity the Miss Ratio decreases at a slower rate. On the other hand, when we look at the MINIFE.t file we notice that the Miss Ratio decreases when the associativity increases from 1 to 8, but we see that it starts increasing when the associativity changes from 8 to 64. A higher-level associative cache can reduce the Miss Rate because each set can have more blocks which reduces the chances for a conflict, but this might not always be the case.

XSBENCH.t table:

Associativity	Miss Ratio
1	0.122697
2	0.114457
4	0.112539
8	0.112007
16	0.111905
32	0.111784

64	0.111753

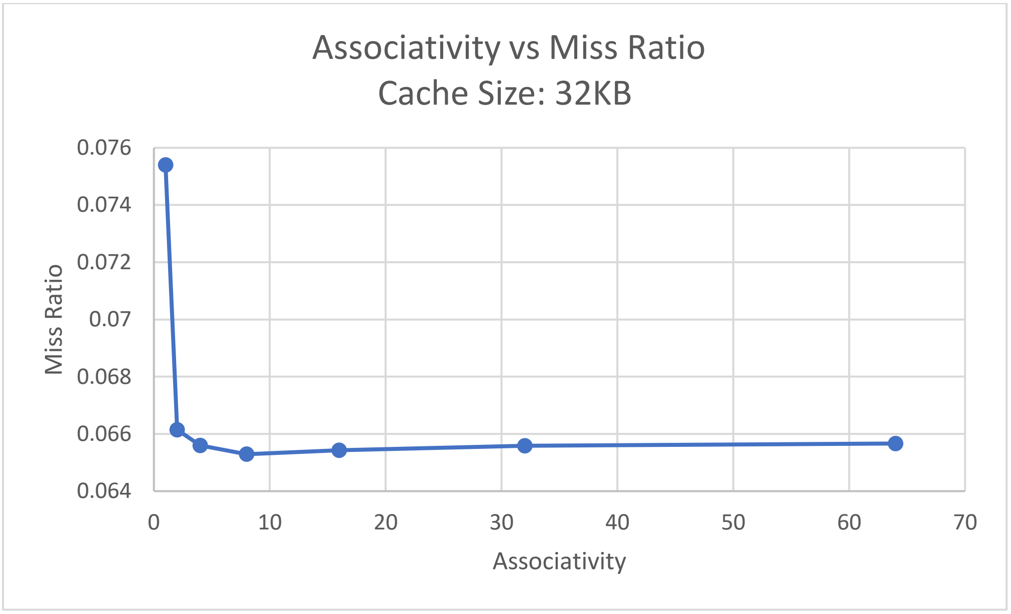
XEBENCH.t graph:



MINIFE.t table:

Associativity	Miss Ratio
1	0.075405
2	0.066151
4	0.065594
8	0.065291
16	0.065429
32	0.065583
64	0.065665

MINIFE.t graph:



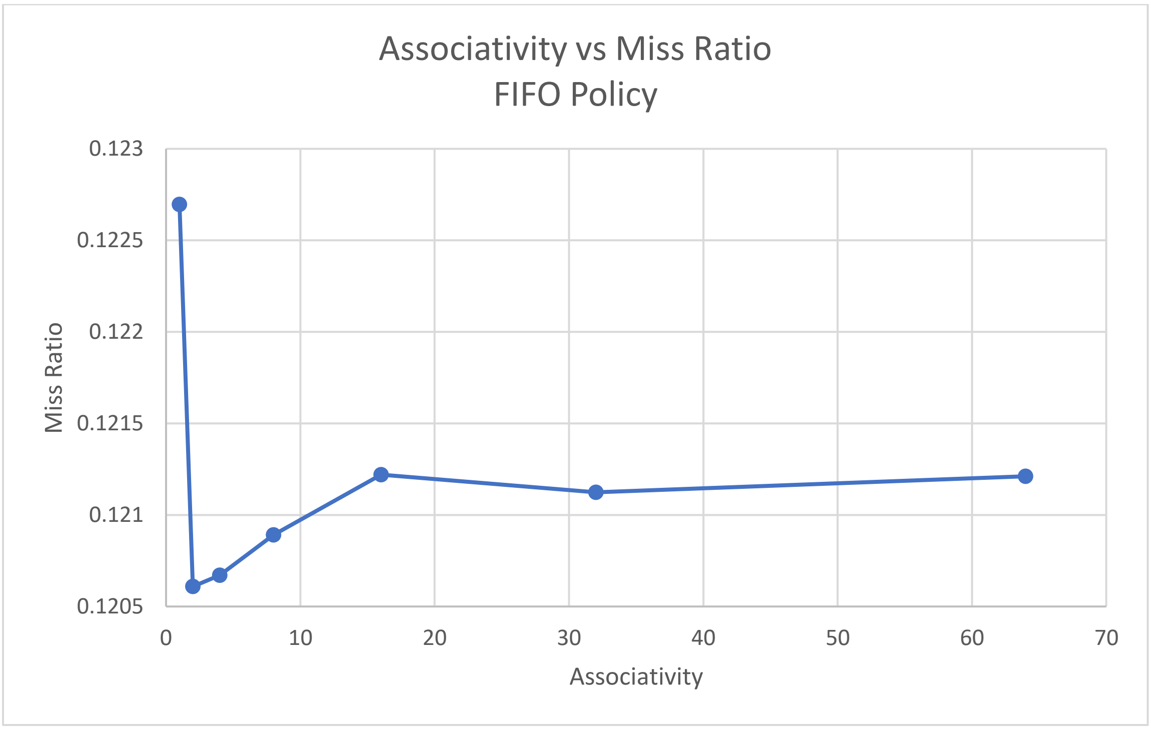
Part D:

In Part D, we are asked to repeat Part C but using the FIFO replacement policy. We are asked to analyze the results for Part C and Part D and understand how the replacement policy changes with associativity. Based on the data, we can conclude that the LIFO and FIFO policies have similar outcomes for Miss Ratio. For the XSBench.t file we see that after the associativity changes from 1 to 2, the miss ratio increases as the associativity increases. For the MINIFE.t file, we see a similar trend to the LRU policy where the Miss Ratio decreases from associativity 1-8 but increases from 8-64. Another observation based on the graphs, is how the LRU policy has a steadier rate change than FIFO. This is because LRU is said to have better accuracy and performance than FIFO. Overall, the only difference we noticed was for the XSBench.t file where the Miss Ratio decreases at a constant rate for the LRU policy, but increases in Miss Ratio for the FIFO policy.

XSBench.t Table

Associativity	Miss Ratio
1	0.122696
2	0.120609
4	0.120671
8	0.120891
16	0.12122
32	0.121124
64	0.121212

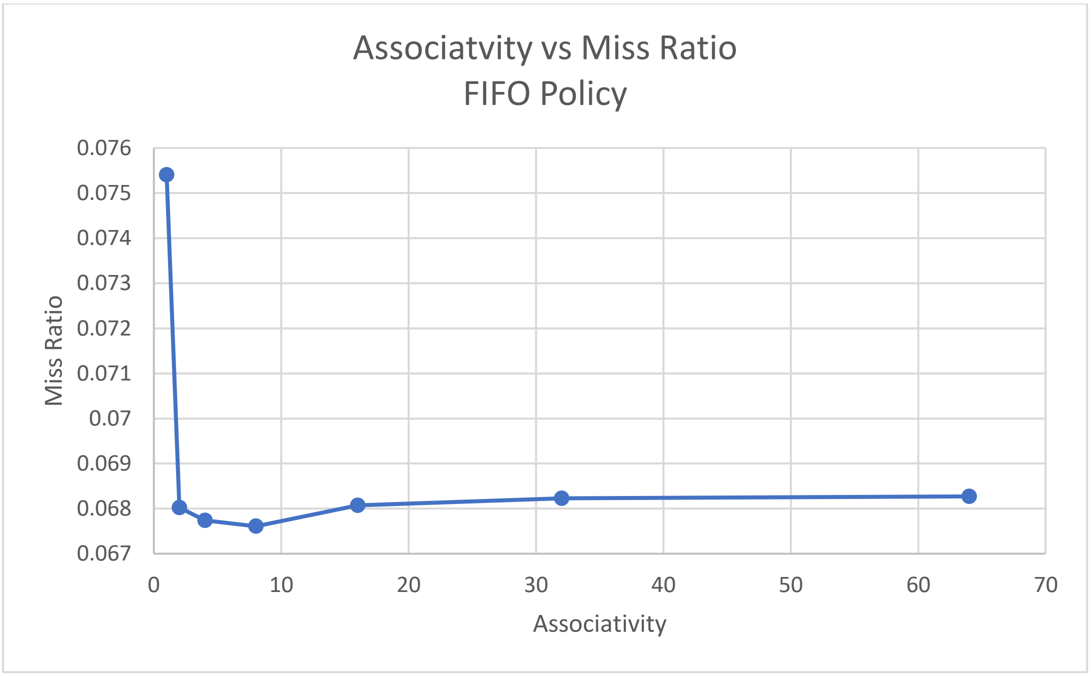
XSBench.t Graph



MINIFE.t Table

Associativity	Miss Ratio
1	0.075405
2	0.068023
4	0.067742
8	0.067609
16	0.06807
32	0.068226
64	0.068274

MINIFE.t graph



Notes:

For the Bonus Opportunity, I implemented the LIFO policy.

The compilation command used is python3
 Example: python3 sim.py 32768 4 1 1 XSBench.t