COMPUTER ARCHITECTURE ASSIGNMENT 1 CACHE

In this assignment, we analyse the performance of cache and evaluate how different cache configurations affect the hit/miss rates. The cache configurations include variations in cache size, block size, and associativity. We use memory trace files to measure the impact of these parameters on cache performance.

1) **Problem a:**
   We design a cache with the following properties:
   - Associativity: 4 – way
   - Cache size: 1024Kb
   - Block size: 4 Bytes
   The address provided to us is 32 bits.
   Number of cache lines needed = [(cache size in bytes) / (block size in bytes)] / (associativity)
   = [1024*1024 / 4]/ 4
   = 65536 sets

2) **Problem b: Vary cache size**
   We design a cache with the following properties:
   - Associativity: 4 – way
   - Cache sizes: 128, 256, 512, 1024, 2048, 4096 Kilobytes
   - Block size: 4 Bytes
   The address provided to us is 32 bits.

3) **Problem c: Vary Block size**
   We design a cache with the following properties:
   - Associativity: 4 – way
   - Cache size: 1024Kb
   - Block sizes: 1, 2, 4, 8, 16, 32, 64, 128 Bytes The address provided to us is 32 bits.

4) **Problem d: Vary Associativity**
   We design a cache with the following properties:
   - Associativities: 1, 2, 4, 8, 16, 32, 64 ways
   - Cache size: 1024Kb
   - Block size: 4 Bytes
   The address provided to us is 32 bits.

We have been provided with input trace files available at: https://cseweb.ucsd.edu/classes/fa07/cse240a/proj1traces.tar.gz. These trace files are embedded in a list within our program.

We have imported the math library to calculate logarithms needed for determining byte offset and index bits. Additionally, we have imported the matplotlib library for plotting graphs.

FUNCTIONS IN OUR PROGRAM

1. initialize_cache:
   - Arguments: Cache size (in KB), block size (in bytes), associativity.
   - Description: It calculates the number of sets for the cache, initializes the cache with the calculated number of sets and ways. Each line has a tag-bits, a valid bit, and an LRU count. The LRU count is used to implement the LRU replacement algorithm. It returns the cache, number of sets, and associativity.

2. get_index_and_tag:
   - Arguments: Address (in hexadecimal), block size (in bytes), number of sets.
   - Description: It calculates the number of byte offset and index bits using the math library's log2 function. It also calculates the number of tag bits using the formula **tag_bits = 32 - index_bits - byte_offset_bits**. It converts the address to binary and extracts the tag and index. This function returns the index and tag.

3. update_lru:
   - Arguments: Cache, index, way where the data is found, associativity.
   - Description: Updates the LRU counters according to the LRU replacement algorithm.

4. replace_lru:
   - Arguments: Index, tag, cache, associativity.
   - Description: Implements the LRU replacement algorithm to add a new block by replacing the block with the lowest LRU count, using the update_lru function.

5. access_cache:
   - Arguments: Address in hexadecimal, cache, block_size in bytes, number of sets, associativity
   - Description: Invokes get_index_and_tag, checks for a cache hit or miss. If it's a cache hit, invokes update_lru. If it's a cache miss, invokes replace_lru. Returns True for a cache hit and False for a cache miss.

6. hex_to_bin:
   - Arguments: Hexadecimal string
   - Description: A helper function that converts a hexadecimal string to binary.

7. simulate_cache:
   - Arguments: Cache size (in KB), block size (in bytes), associativity, trace file.
   - Description: Reads each instruction from the input file, invokes access_cache for each load or store instruction, and uses hex_to_bin to convert addresses. Calculates the number of hits and misses based on the access_cache function. Returns the hit and miss rates.

8. plot_results:
   - Arguments: trace file, X-axis values for plotting, dictionary of miss rates, label for the X-axis, title of graph, Plotting option.
   - Description: Plots graphs for the specified problems.

9. cache_simulation_for_1024:
   - Arguments: trace files, cache_size in KB, block_size in bytes, associativity.
   - Description: Implements Problem 'a'. Calculates hit and miss rates for each file using a loop and the simulate_cache function.

10. varying_cache_size:
    - Arguments: trace files, list of cache_size in KB, block_size in bytes, associativity.
    - Description: Implements Problem 'b'. Calculates hit and miss rates for each file using a loop and the simulate_cache function. Also invokes plot_results to plot graphs for Problem 'b'.

11. varying_block_size:
    - Arguments: trace files, cache_size in KB, list of block_size in bytes, associativity.
    - Description: Implements Problem 'c'. Calculates hit and miss rates for each file using a loop and the simulate_cache function. Also invokes plot_results to plot graphs for Problem 'c'.

12. varying_associativity:
    - Arguments: trace files, cache_size in KB, block_size in byte, list of associativities.

- Description: Implements Problem ‘d’. Calculates hit and miss rates for each file using a loop and the simulate_cache function. Also invokes plot_results to plot graphs for Problem ‘d’.

13. varying_cache_size_all:
    - Arguments: trace files, list of cache_size in KB, block_size in bytes, associativity.
    - Description: Implements Problem ‘b’. It is similar to the varying_cache_size function. Instead of plotting graphs for individual trace files, we print a cumulative graph.

14. varying_block_size_all:
    - Arguments: trace files, cache_size in KB, list of block_size in bytes, associativity.
    - Description: Implements Problem ‘c’. It is similar to the varying_block_size function. Instead of plotting graphs for individual trace files, we print a cumulative graph.

15. varying_associativity_all:
    - Arguments: trace files,cache_size in KB,block_size in bytes,list of associativities.
    - Description: Implements Problem ‘d’. It is similar to the varying_associtivity function. Instead of plotting graphs for individual trace files, we print a cumulative graph.

16. plot_results_all:
    - Arguments: trace files, cache_size in KB, block_size in bytes, list of associativities.
    - Description: Similar to plot_results. It prints cumulative graph for all input files.

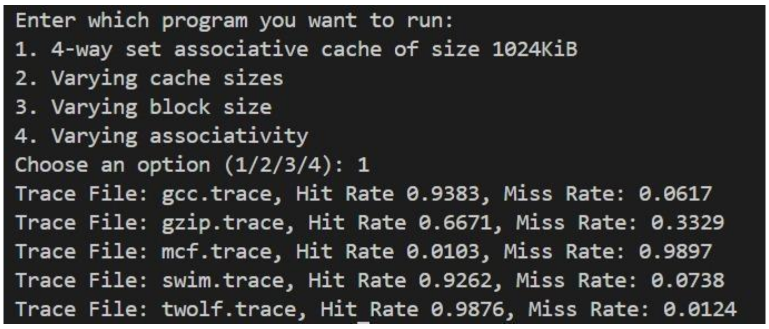
PROGRAM EXECUTION

1. The program starts by asking the user to choose which problem to run:
    - **‘1’** for Problem ‘a’ ○ **‘2’** for Problem ‘b’ ○ **‘3’** for Problem ‘c’
    - **‘4’** for Problem ‘d’
2. Based on the user's choice:
    - If ‘**1**’ is selected, the cache_simulation_for_1024 function is invoked. ○ If ‘**2**’ is selected, the varying_cache_size function is invoked. ○ If ‘**3**’ is selected, the varying_block_size function is invoked. ○ If ‘**4**’ is selected, the varying_associativity function is invoked.
3. The program prints the trace file name, hit rate, and miss rate for each trace file. It plots the graphs for the problems. Excel tables displaying these values have also been added in the result.

RESULTS

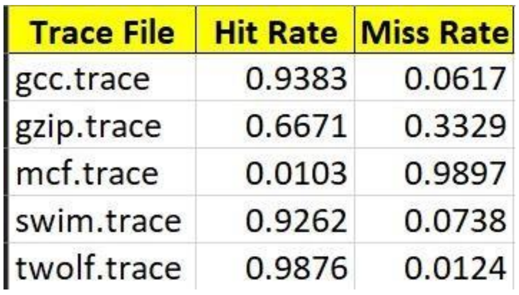
1) PROBLEM a

OUTPUT:



```
Enter which program you want to run:
1. 4-way set associative cache of size 1024KiB
2. Varying cache sizes
3. Varying block size
4. Varying associativity
Choose an option (1/2/3/4): 1
Trace File: gcc.trace, Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gzip.trace, Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: mcf.trace, Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: swim.trace, Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: twolf.trace, Hit Rate 0.9876, Miss Rate: 0.0124
```

EXCEL:



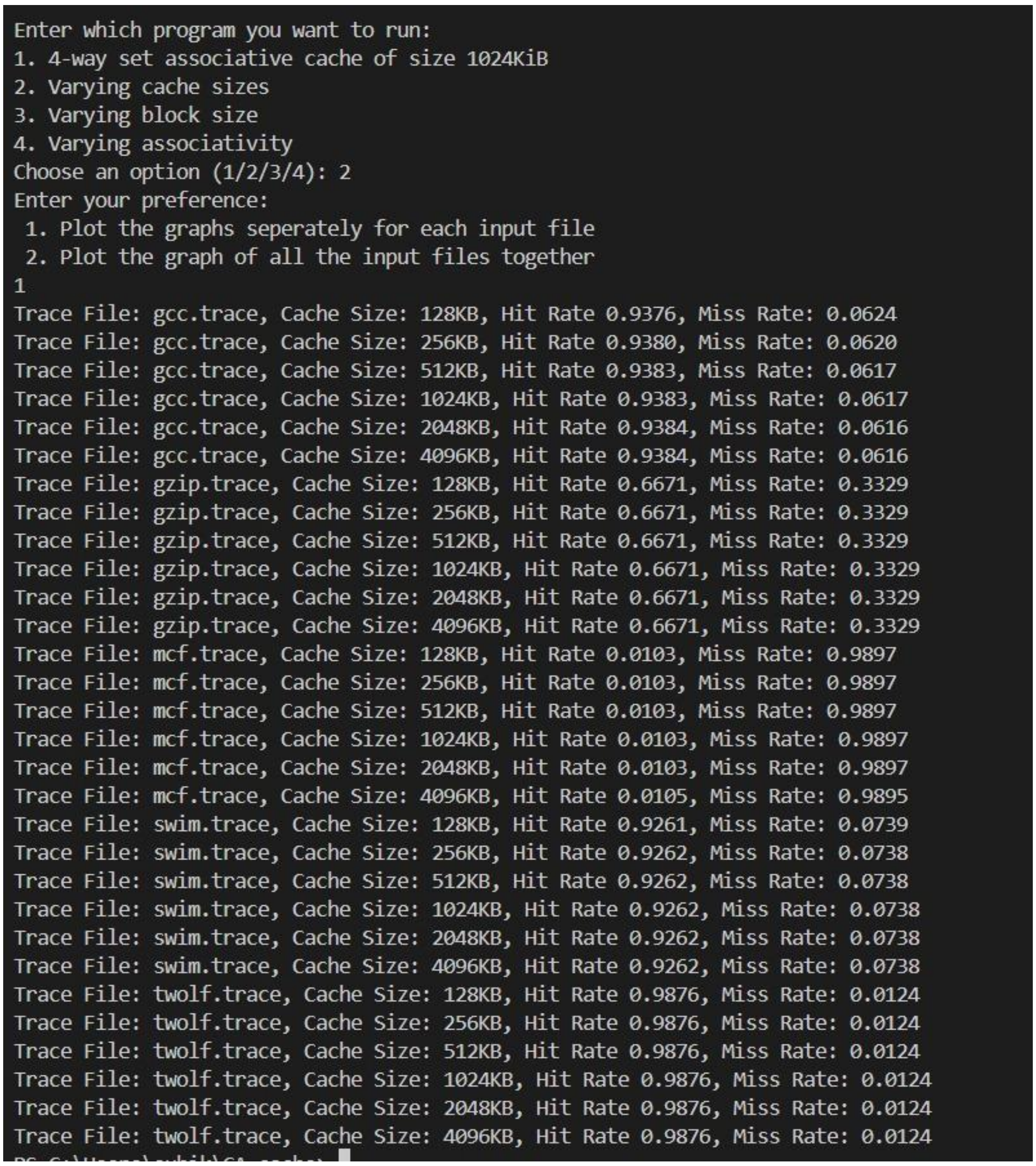
Trace File	Hit Rate	Miss Rate
gcc.trace	0.9383	0.0617
gzip.trace	0.6671	0.3329
mcf.trace	0.0103	0.9897
swim.trace	0.9262	0.0738
twolf.trace	0.9876	0.0124

OBSERVATION:

Hit rate is maximum for twolf.trace and minimum for mcf.trace.

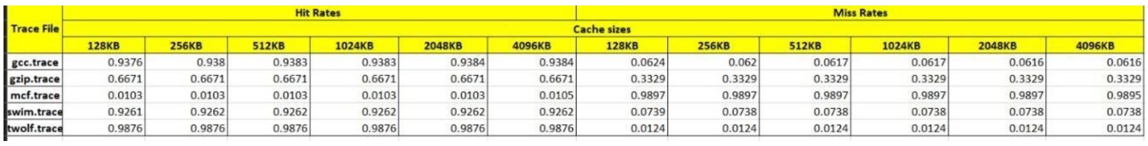
2) PROBLEM b

OUTPUT:



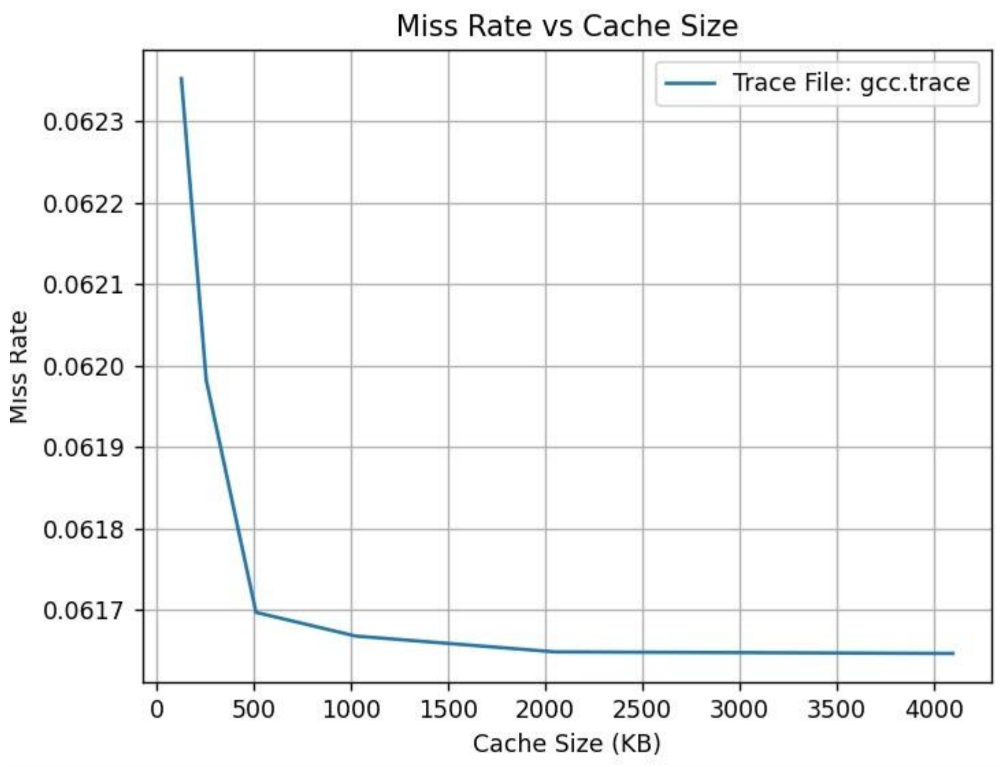
```
Enter which program you want to run:
1. 4-way set associative cache of size 1024KiB
2. Varying cache sizes
3. Varying block size
4. Varying associativity
Choose an option (1/2/3/4): 2
Enter your preference:
 1. Plot the graphs seperately for each input file
 2. Plot the graph of all the input files together
1
Trace File: gcc.trace, Cache Size: 128KB, Hit Rate 0.9376, Miss Rate: 0.0624
Trace File: gcc.trace, Cache Size: 256KB, Hit Rate 0.9380, Miss Rate: 0.0620
Trace File: gcc.trace, Cache Size: 512KB, Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gcc.trace, Cache Size: 1024KB, Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gcc.trace, Cache Size: 2048KB, Hit Rate 0.9384, Miss Rate: 0.0616
Trace File: gcc.trace, Cache Size: 4096KB, Hit Rate 0.9384, Miss Rate: 0.0616
Trace File: gzip.trace, Cache Size: 128KB, Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Cache Size: 256KB, Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Cache Size: 512KB, Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Cache Size: 1024KB, Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Cache Size: 2048KB, Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Cache Size: 4096KB, Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: mcf.trace, Cache Size: 128KB, Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Cache Size: 256KB, Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Cache Size: 512KB, Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Cache Size: 1024KB, Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Cache Size: 2048KB, Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Cache Size: 4096KB, Hit Rate 0.0105, Miss Rate: 0.9895
Trace File: swim.trace, Cache Size: 128KB, Hit Rate 0.9261, Miss Rate: 0.0739
Trace File: swim.trace, Cache Size: 256KB, Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Cache Size: 512KB, Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Cache Size: 1024KB, Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Cache Size: 2048KB, Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Cache Size: 4096KB, Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: twolf.trace, Cache Size: 128KB, Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Cache Size: 256KB, Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Cache Size: 512KB, Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Cache Size: 1024KB, Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Cache Size: 2048KB, Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Cache Size: 4096KB, Hit Rate 0.9876, Miss Rate: 0.0124
```

EXCEL:

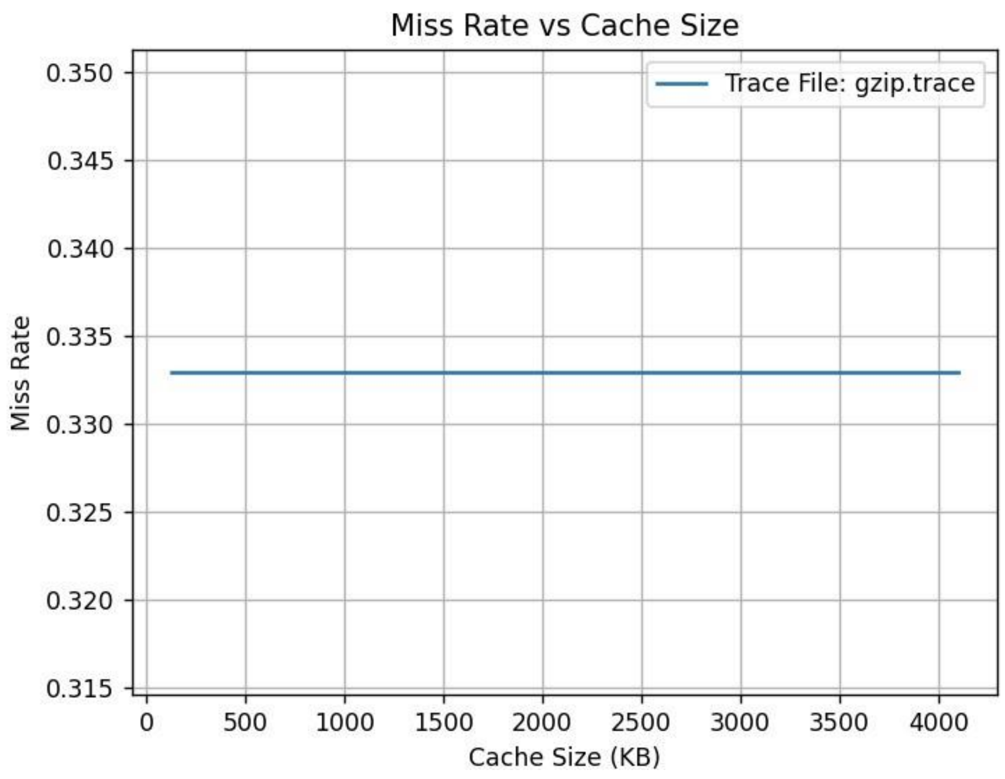


Trace File	Hit Rates						Miss Rates					
	Cache sizes											
	128KB	256KB	512KB	1024KB	2048KB	4096KB	128KB	256KB	512KB	1024KB	2048KB	4096KB
gcc.trace	0.9376	0.938	0.9383	0.9383	0.9384	0.9384	0.0624	0.062	0.0617	0.0617	0.0616	0.0616
gzip.trace	0.6671	0.6671	0.6671	0.6671	0.6671	0.6671	0.3329	0.3329	0.3329	0.3329	0.3329	0.3329
mcf.trace	0.0103	0.0103	0.0103	0.0103	0.0103	0.0105	0.9897	0.9897	0.9897	0.9897	0.9897	0.9895
swim.trace	0.9261	0.9262	0.9262	0.9262	0.9262	0.9262	0.0739	0.0738	0.0738	0.0738	0.0738	0.0738
twolf.trace	0.9876	0.9876	0.9876	0.9876	0.9876	0.9876	0.0124	0.0124	0.0124	0.0124	0.0124	0.0124

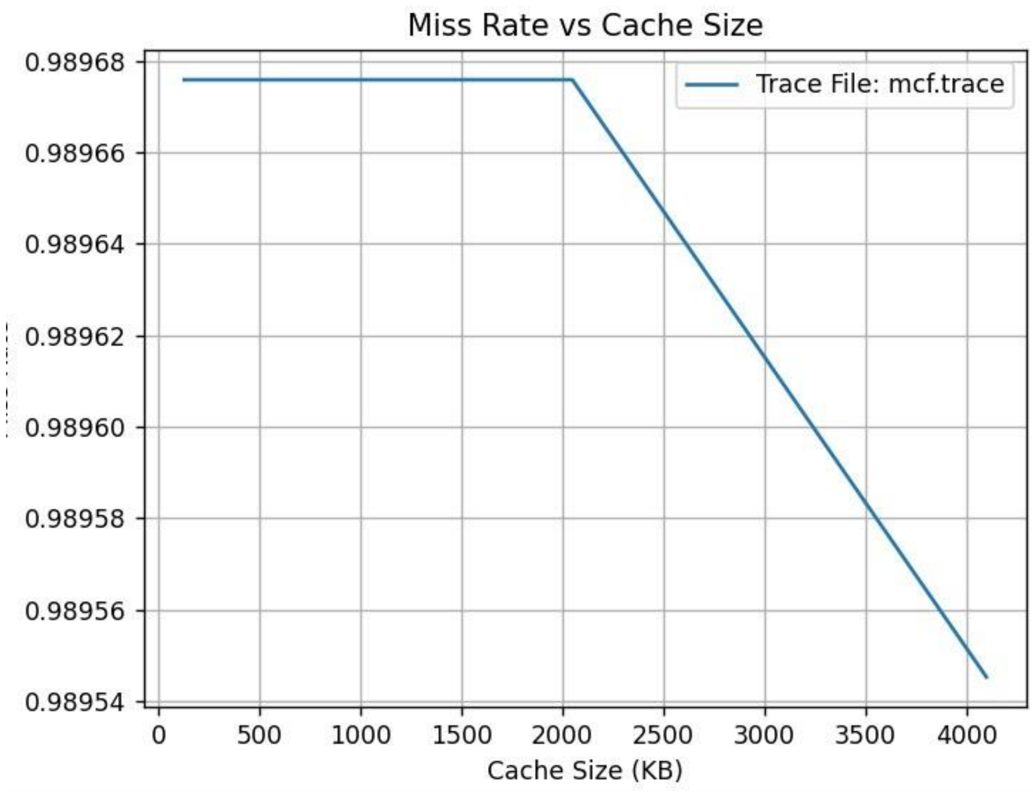
GRAPH FOR GCC:



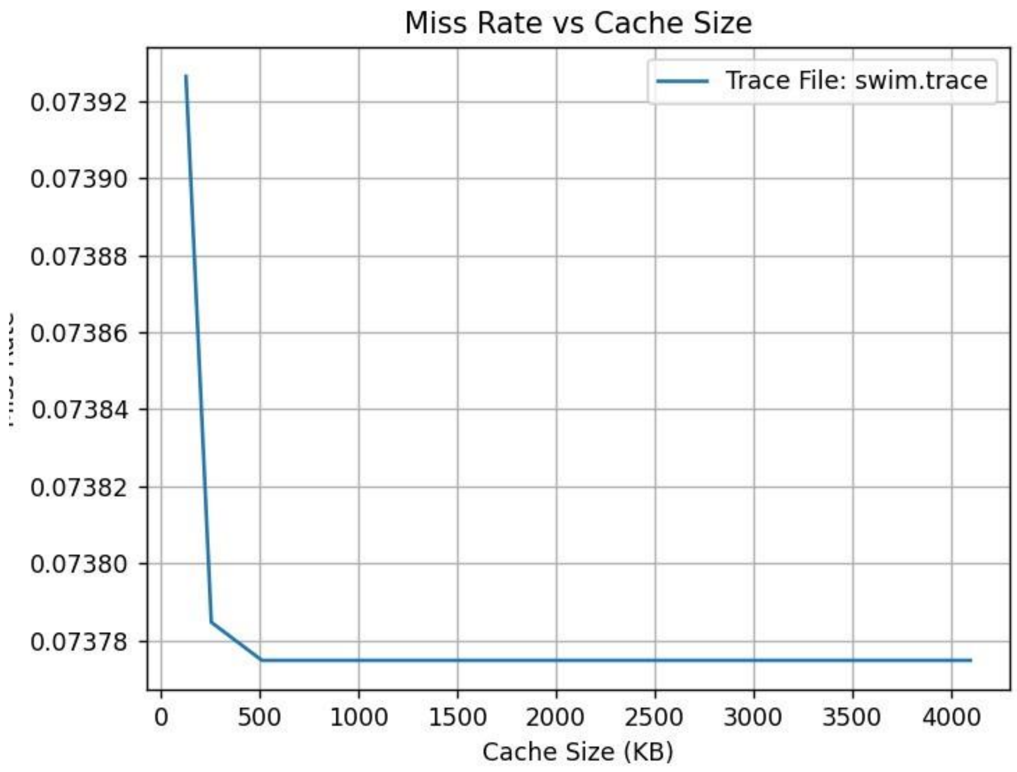
GRAPH FOR GZIP:



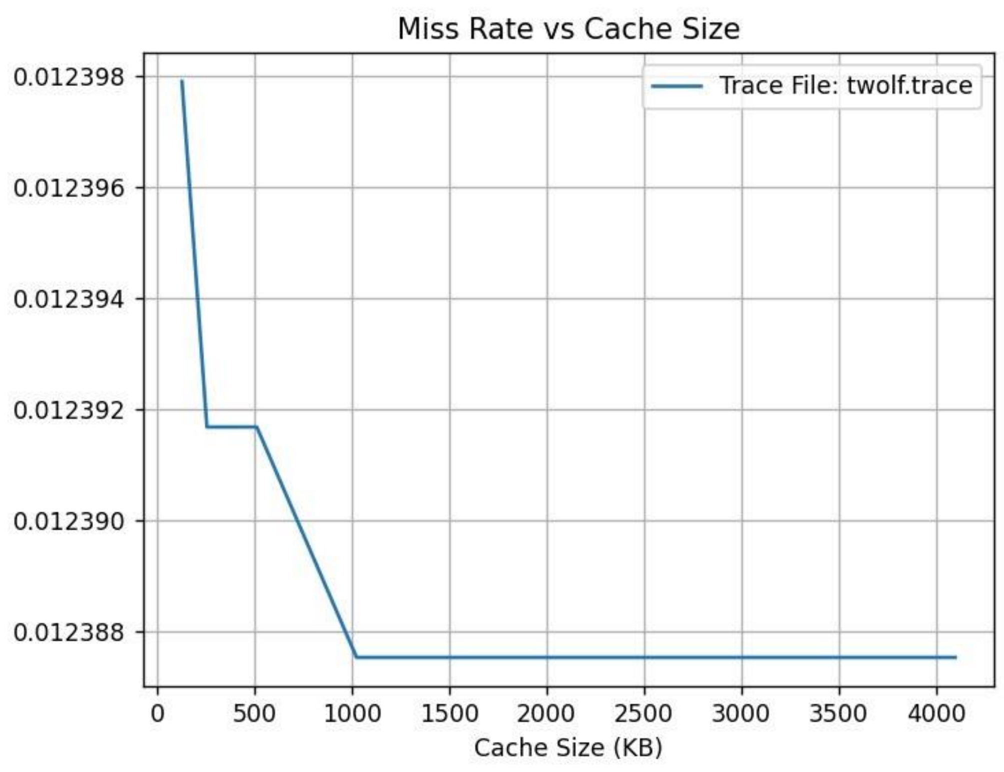
GRAPH FOR MCF:



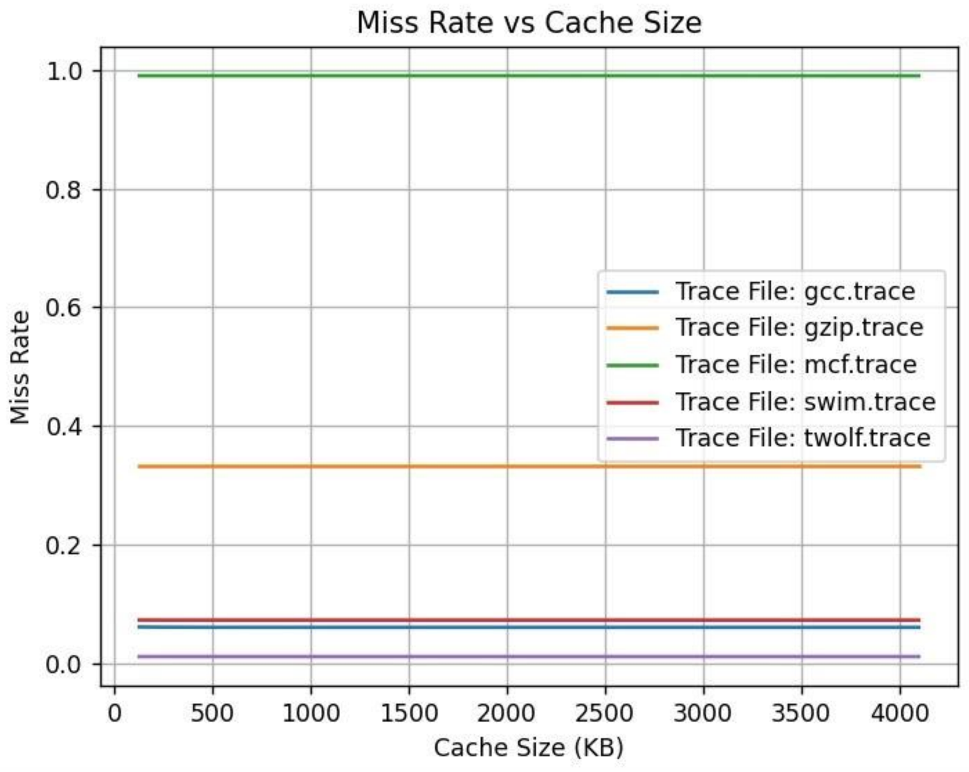
GRAPH FOR SWIM:



GRAPH FOR TWOLF:



CUMULATIVE GRAPH:

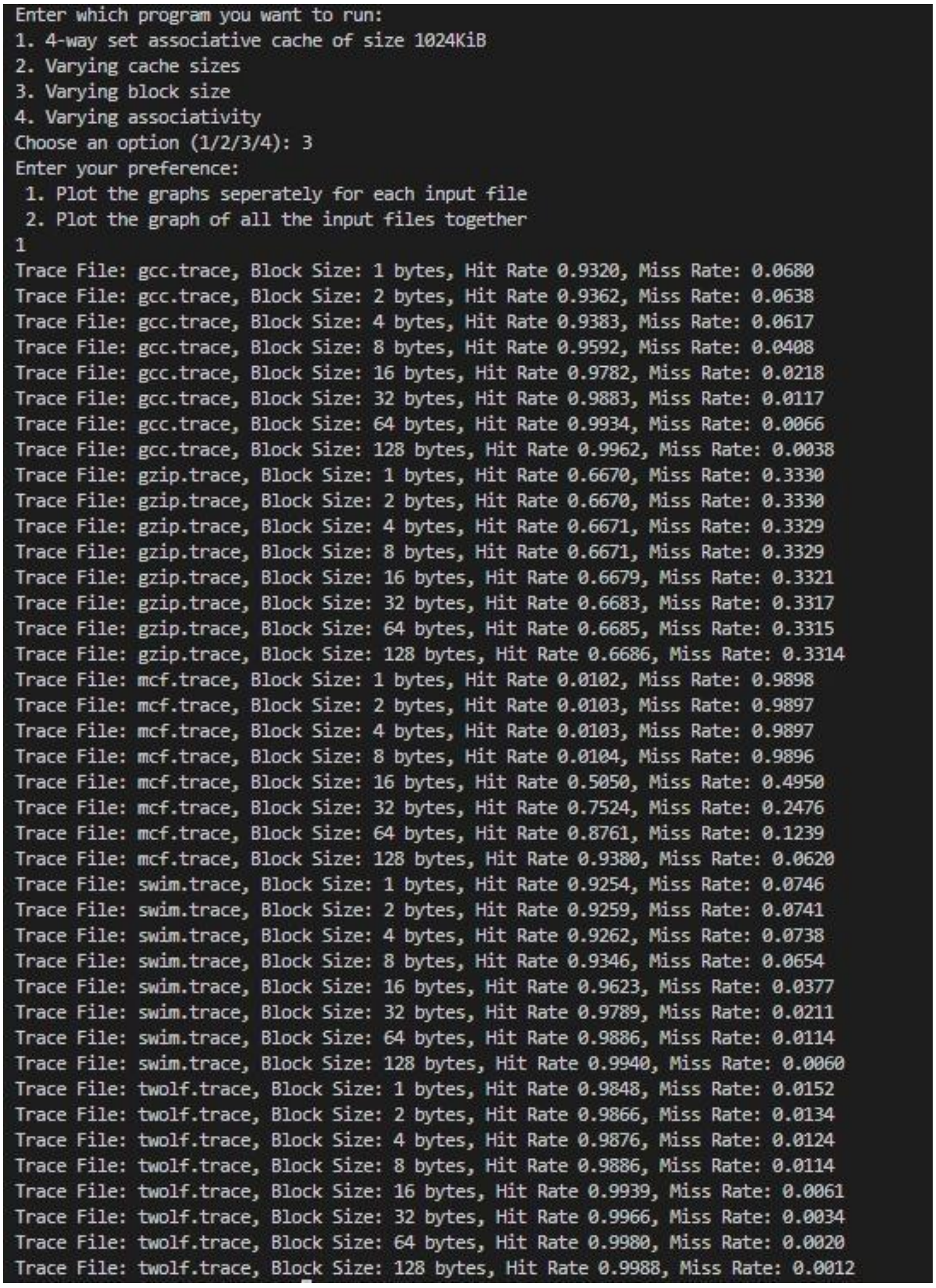


OBSERVATION:

- We can see that all traces behave in a similar way.
- As the cache size increases, the miss rate tends to decrease because larger caches can store more data, reducing the chances of cache misses.
- For small cache sizes, the miss rate is higher due to frequent evictions caused by insufficient space.
- Beyond a certain cache size, the reduction in miss rate becomes less significant, clearly seen in the graphs. - mcf.trace has the maximum miss rate and twolf.trace has the minimum miss rate.

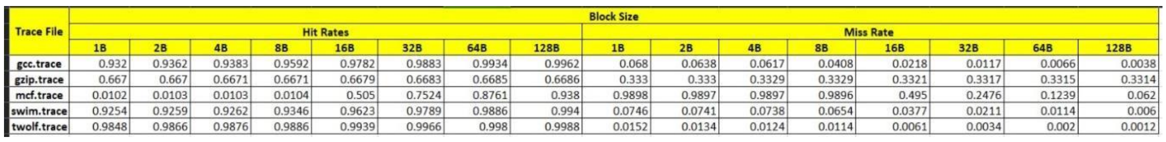
3) PROBLEM c

OUTPUT:



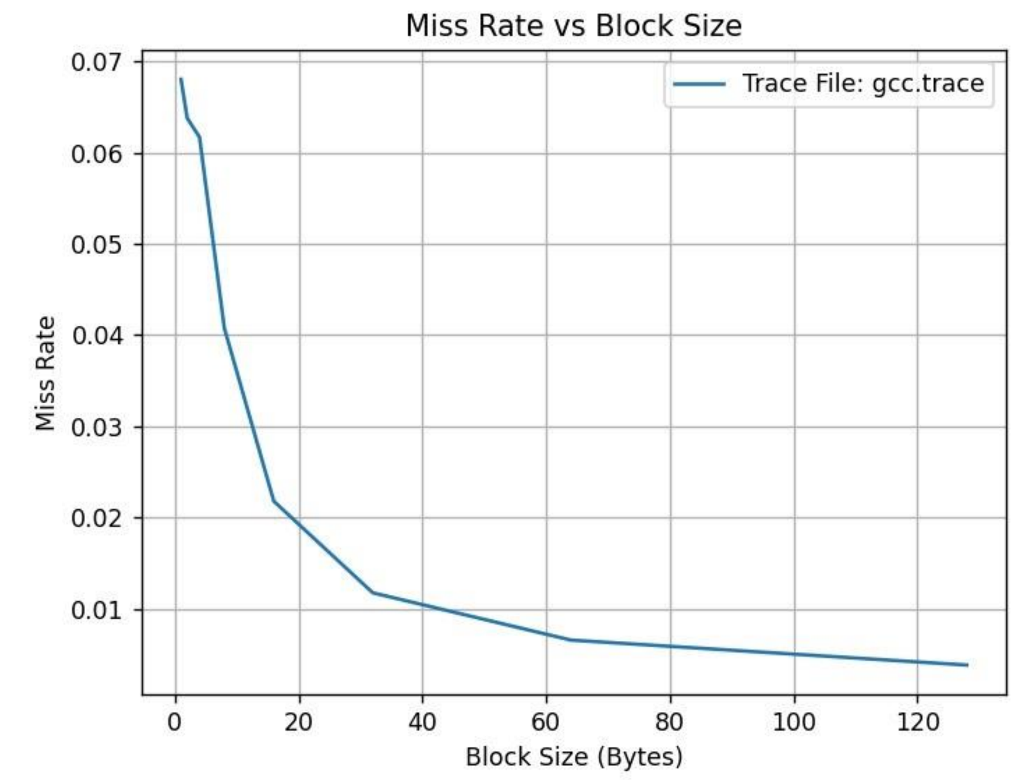
```
Enter which program you want to run:
1. 4-way set associative cache of size 1024KiB
2. Varying cache sizes
3. Varying block size
4. Varying associativity
Choose an option (1/2/3/4): 3
Enter your preference:
 1. Plot the graphs seperately for each input file
 2. Plot the graph of all the input files together
1
Trace File: gcc.trace, Block Size: 1 bytes, Hit Rate 0.9320, Miss Rate: 0.0680
Trace File: gcc.trace, Block Size: 2 bytes, Hit Rate 0.9362, Miss Rate: 0.0638
Trace File: gcc.trace, Block Size: 4 bytes, Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gcc.trace, Block Size: 8 bytes, Hit Rate 0.9592, Miss Rate: 0.0408
Trace File: gcc.trace, Block Size: 16 bytes, Hit Rate 0.9782, Miss Rate: 0.0218
Trace File: gcc.trace, Block Size: 32 bytes, Hit Rate 0.9883, Miss Rate: 0.0117
Trace File: gcc.trace, Block Size: 64 bytes, Hit Rate 0.9934, Miss Rate: 0.0066
Trace File: gcc.trace, Block Size: 128 bytes, Hit Rate 0.9962, Miss Rate: 0.0038
Trace File: gzip.trace, Block Size: 1 bytes, Hit Rate 0.6670, Miss Rate: 0.3330
Trace File: gzip.trace, Block Size: 2 bytes, Hit Rate 0.6670, Miss Rate: 0.3330
Trace File: gzip.trace, Block Size: 4 bytes, Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Block Size: 8 bytes, Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Block Size: 16 bytes, Hit Rate 0.6679, Miss Rate: 0.3321
Trace File: gzip.trace, Block Size: 32 bytes, Hit Rate 0.6683, Miss Rate: 0.3317
Trace File: gzip.trace, Block Size: 64 bytes, Hit Rate 0.6685, Miss Rate: 0.3315
Trace File: gzip.trace, Block Size: 128 bytes, Hit Rate 0.6686, Miss Rate: 0.3314
Trace File: mcf.trace, Block Size: 1 bytes, Hit Rate 0.0102, Miss Rate: 0.9898
Trace File: mcf.trace, Block Size: 2 bytes, Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Block Size: 4 bytes, Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Block Size: 8 bytes, Hit Rate 0.0104, Miss Rate: 0.9896
Trace File: mcf.trace, Block Size: 16 bytes, Hit Rate 0.5050, Miss Rate: 0.4950
Trace File: mcf.trace, Block Size: 32 bytes, Hit Rate 0.7524, Miss Rate: 0.2476
Trace File: mcf.trace, Block Size: 64 bytes, Hit Rate 0.8761, Miss Rate: 0.1239
Trace File: mcf.trace, Block Size: 128 bytes, Hit Rate 0.9380, Miss Rate: 0.0620
Trace File: swim.trace, Block Size: 1 bytes, Hit Rate 0.9254, Miss Rate: 0.0746
Trace File: swim.trace, Block Size: 2 bytes, Hit Rate 0.9259, Miss Rate: 0.0741
Trace File: swim.trace, Block Size: 4 bytes, Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Block Size: 8 bytes, Hit Rate 0.9346, Miss Rate: 0.0654
Trace File: swim.trace, Block Size: 16 bytes, Hit Rate 0.9623, Miss Rate: 0.0377
Trace File: swim.trace, Block Size: 32 bytes, Hit Rate 0.9789, Miss Rate: 0.0211
Trace File: swim.trace, Block Size: 64 bytes, Hit Rate 0.9886, Miss Rate: 0.0114
Trace File: swim.trace, Block Size: 128 bytes, Hit Rate 0.9940, Miss Rate: 0.0060
Trace File: twolf.trace, Block Size: 1 bytes, Hit Rate 0.9848, Miss Rate: 0.0152
Trace File: twolf.trace, Block Size: 2 bytes, Hit Rate 0.9866, Miss Rate: 0.0134
Trace File: twolf.trace, Block Size: 4 bytes, Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Block Size: 8 bytes, Hit Rate 0.9886, Miss Rate: 0.0114
Trace File: twolf.trace, Block Size: 16 bytes, Hit Rate 0.9939, Miss Rate: 0.0061
Trace File: twolf.trace, Block Size: 32 bytes, Hit Rate 0.9966, Miss Rate: 0.0034
Trace File: twolf.trace, Block Size: 64 bytes, Hit Rate 0.9980, Miss Rate: 0.0020
Trace File: twolf.trace, Block Size: 128 bytes, Hit Rate 0.9988, Miss Rate: 0.0012
```

EXCEL:

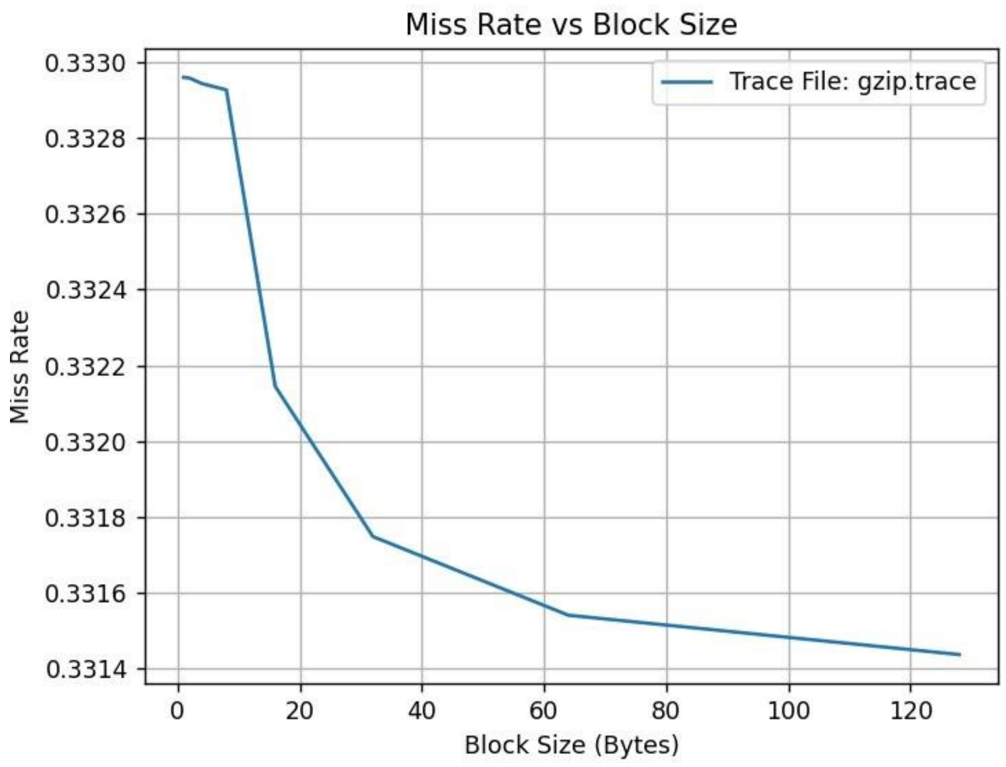


Trace File	Block Size															
	Hit Rates								Miss Rate							
	1B	2B	4B	8B	16B	32B	64B	128B	1B	2B	4B	8B	16B	32B	64B	128B
gcc.trace	0.932	0.9362	0.9383	0.9592	0.9782	0.9883	0.9934	0.9962	0.068	0.0638	0.0617	0.0408	0.0218	0.0117	0.0066	0.0038
gzip.trace	0.667	0.667	0.6671	0.6671	0.6679	0.6683	0.6685	0.6686	0.333	0.333	0.3329	0.3329	0.3321	0.3317	0.3315	0.3314
mcf.trace	0.0102	0.0103	0.0103	0.0104	0.505	0.7524	0.8761	0.938	0.9898	0.9897	0.9897	0.9896	0.495	0.2476	0.1239	0.062
swim.trace	0.9254	0.9259	0.9262	0.9346	0.9623	0.9789	0.9886	0.994	0.0746	0.0741	0.0738	0.0654	0.0377	0.0211	0.0114	0.006
twolf.trace	0.9848	0.9866	0.9876	0.9886	0.9939	0.9966	0.998	0.9988	0.0152	0.0134	0.0124	0.0114	0.0061	0.0034	0.002	0.0012

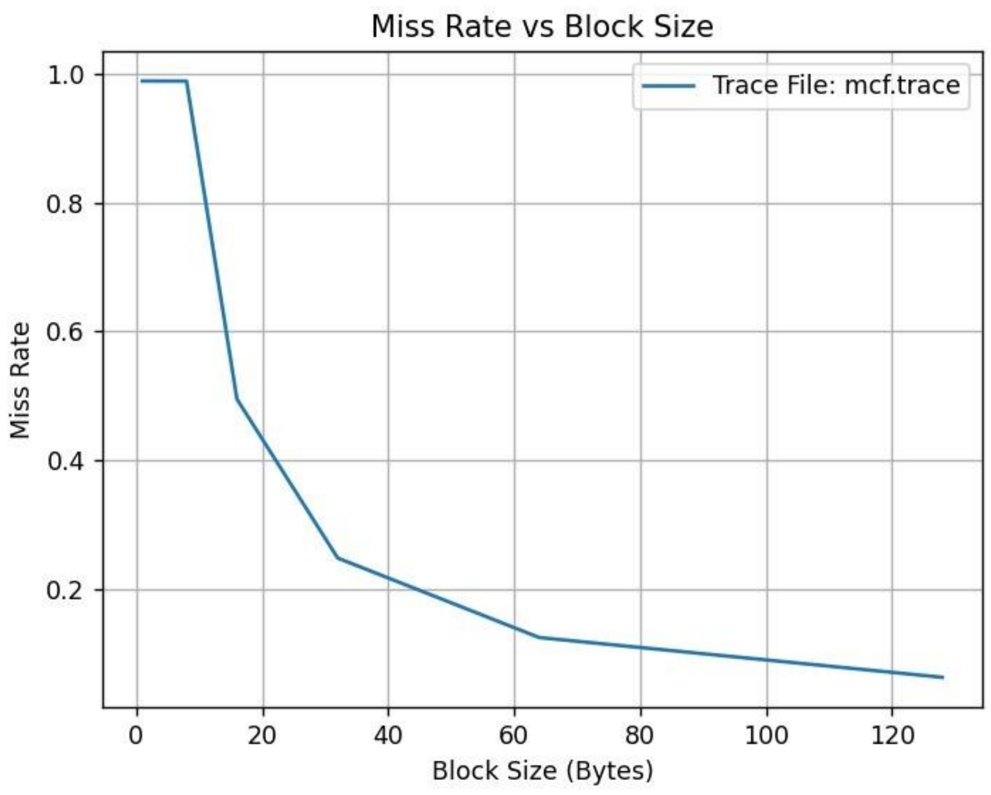
GRAPH FOR GCC:



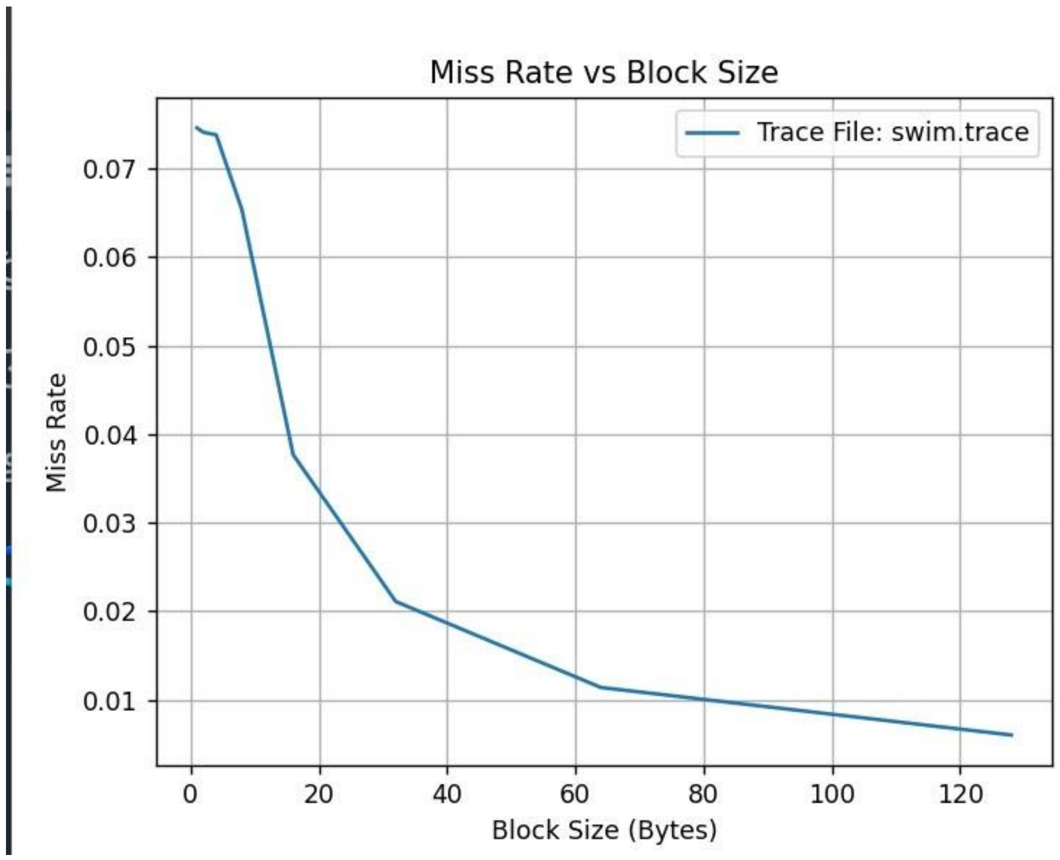
GRAPH FOR GZIP:



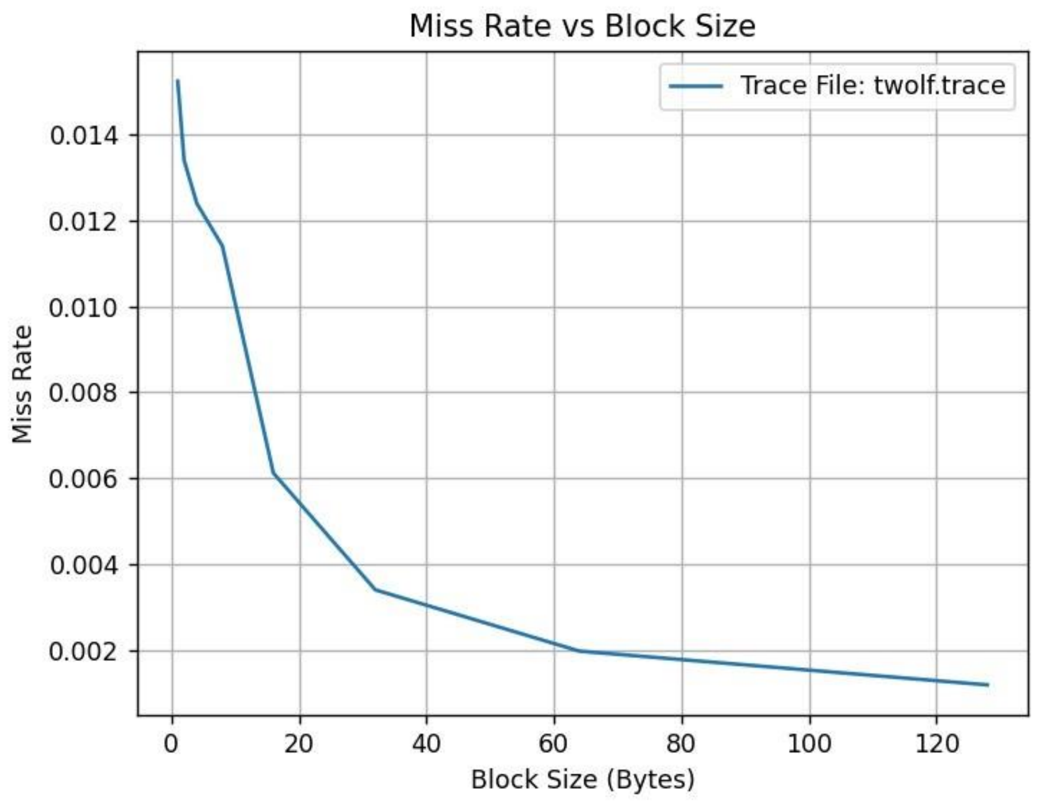
GRAPH FOR MCF:



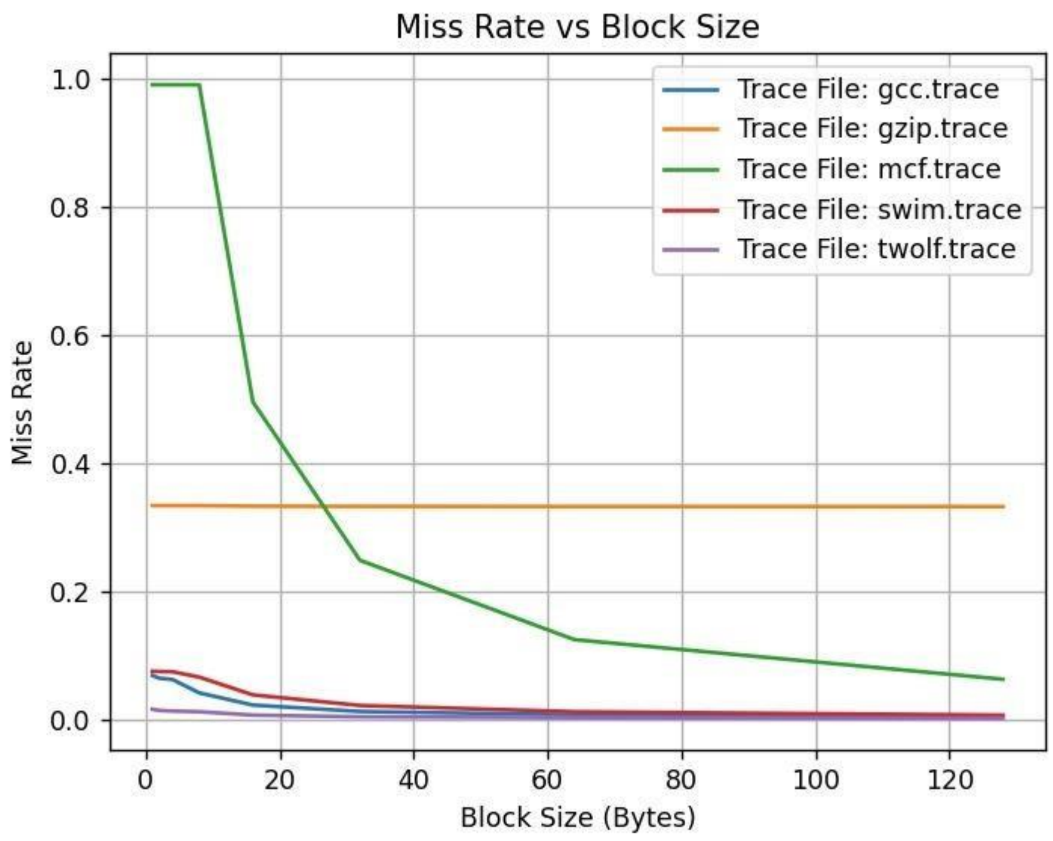
GRAPH FOR SWIM:



GRAPH FOR TWOLF:



CUMULATIVE GRAPH:



OBSERVATION:

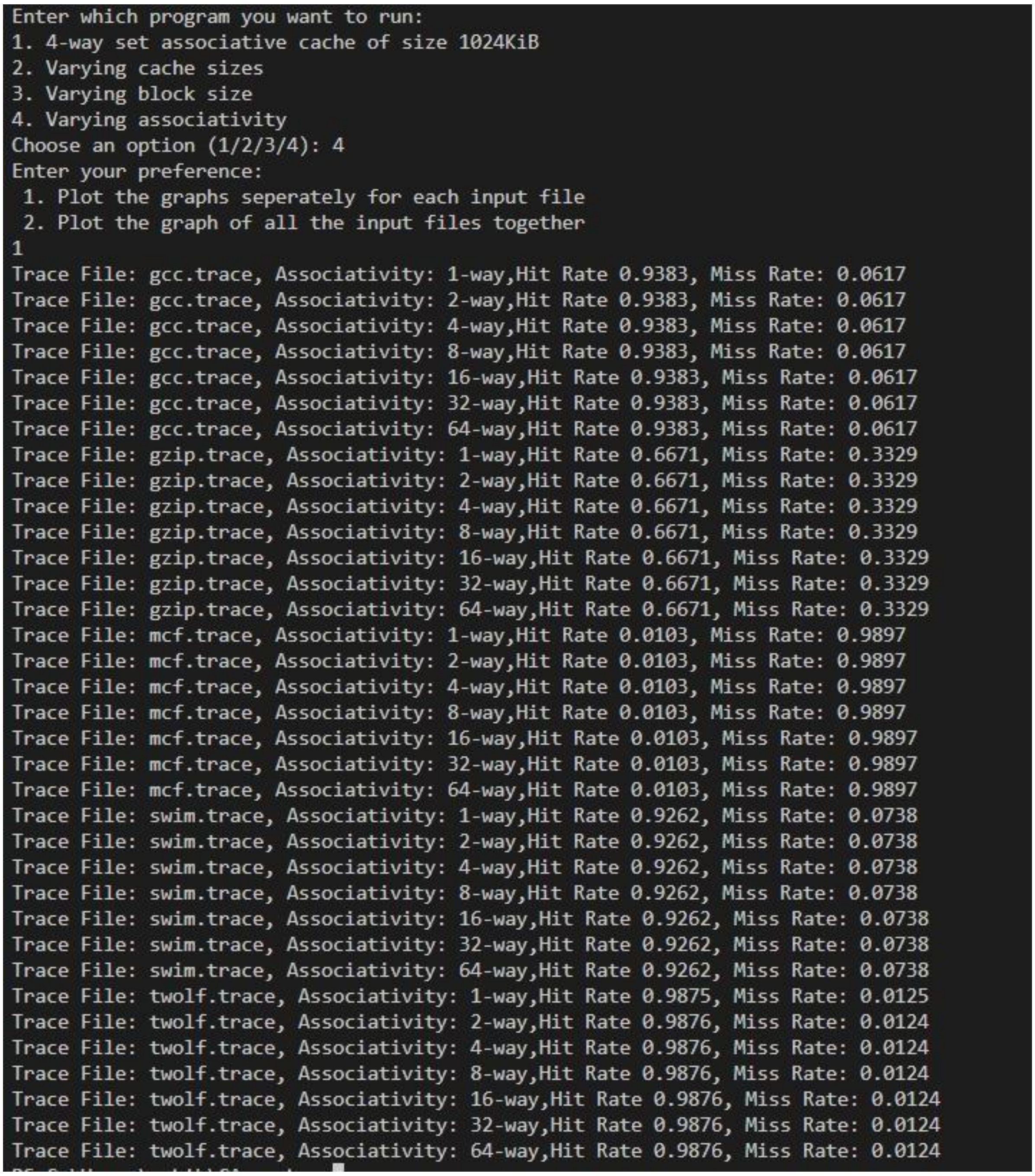
- We can see that all traces behave in a similar way.

- Hit rates increase with increase in block size(from 1 byte to 32 bytes) in the start due to spatial locality.

- After a certain block size (64 bytes or more), miss rates start to increase slightly due to cache inefficiency. - mcf.trace shows the most significant variation in miss rate vs block size.

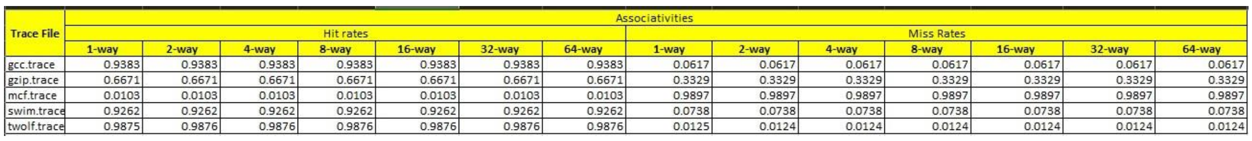
4) PROBLEM d

OUTPUT:



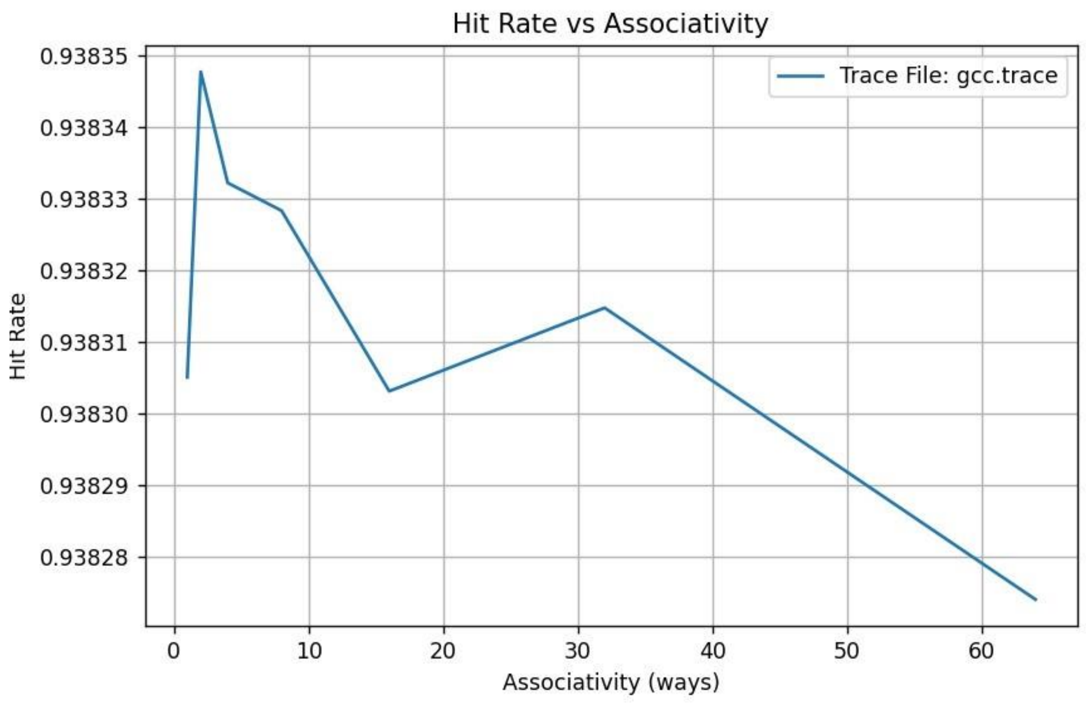
```
Enter which program you want to run:
1. 4-way set associative cache of size 1024KiB
2. Varying cache sizes
3. Varying block size
4. Varying associativity
Choose an option (1/2/3/4): 4
Enter your preference:
 1. Plot the graphs seperately for each input file
 2. Plot the graph of all the input files together
1
Trace File: gcc.trace, Associativity: 1-way,Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gcc.trace, Associativity: 2-way,Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gcc.trace, Associativity: 4-way,Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gcc.trace, Associativity: 8-way,Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gcc.trace, Associativity: 16-way,Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gcc.trace, Associativity: 32-way,Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gcc.trace, Associativity: 64-way,Hit Rate 0.9383, Miss Rate: 0.0617
Trace File: gzip.trace, Associativity: 1-way,Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Associativity: 2-way,Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Associativity: 4-way,Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Associativity: 8-way,Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Associativity: 16-way,Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Associativity: 32-way,Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: gzip.trace, Associativity: 64-way,Hit Rate 0.6671, Miss Rate: 0.3329
Trace File: mcf.trace, Associativity: 1-way,Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Associativity: 2-way,Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Associativity: 4-way,Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Associativity: 8-way,Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Associativity: 16-way,Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Associativity: 32-way,Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: mcf.trace, Associativity: 64-way,Hit Rate 0.0103, Miss Rate: 0.9897
Trace File: swim.trace, Associativity: 1-way,Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Associativity: 2-way,Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Associativity: 4-way,Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Associativity: 8-way,Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Associativity: 16-way,Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Associativity: 32-way,Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: swim.trace, Associativity: 64-way,Hit Rate 0.9262, Miss Rate: 0.0738
Trace File: twolf.trace, Associativity: 1-way,Hit Rate 0.9875, Miss Rate: 0.0125
Trace File: twolf.trace, Associativity: 2-way,Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Associativity: 4-way,Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Associativity: 8-way,Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Associativity: 16-way,Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Associativity: 32-way,Hit Rate 0.9876, Miss Rate: 0.0124
Trace File: twolf.trace, Associativity: 64-way,Hit Rate 0.9876, Miss Rate: 0.0124
```

EXCEL:

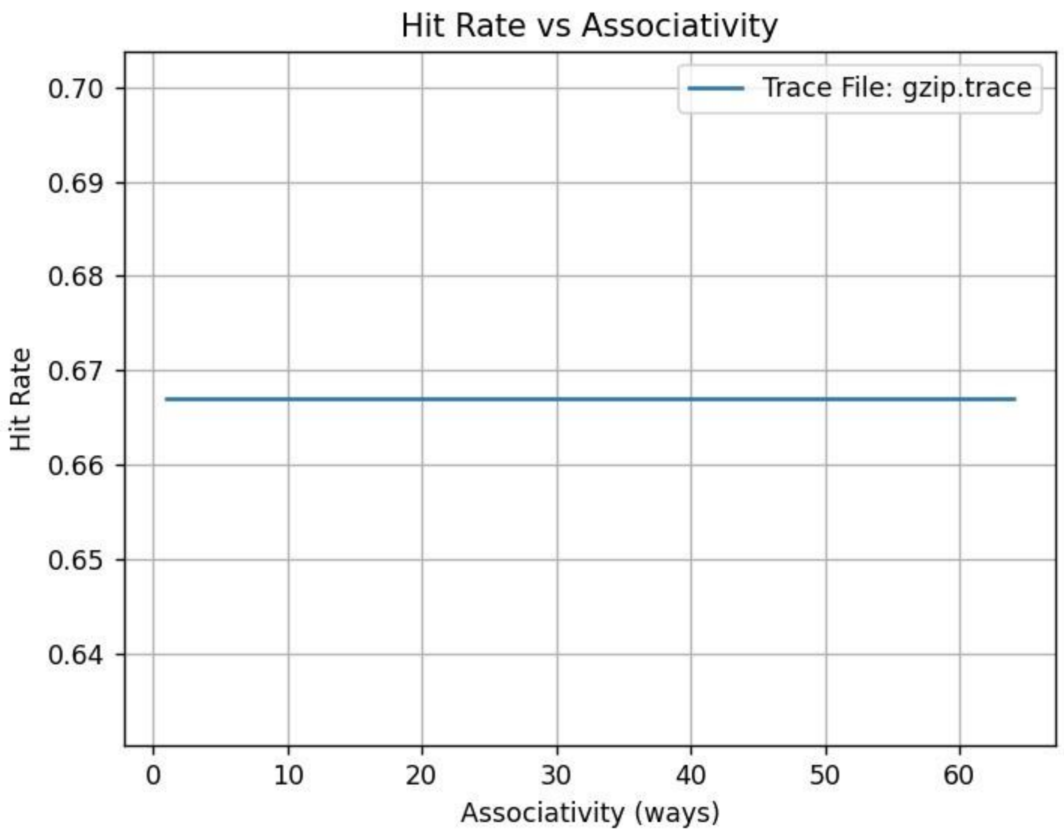


Trace File	Associativities													
	Hit rates							Miss Rates						
	1-way	2-way	4-way	8-way	16-way	32-way	64-way	1-way	2-way	4-way	8-way	16-way	32-way	64-way
gcc.trace	0.9383	0.9383	0.9383	0.9383	0.9383	0.9383	0.9383	0.0617	0.0617	0.0617	0.0617	0.0617	0.0617	0.0617
gzip.trace	0.6671	0.6671	0.6671	0.6671	0.6671	0.6671	0.6671	0.3329	0.3329	0.3329	0.3329	0.3329	0.3329	0.3329
mcf.trace	0.0103	0.0103	0.0103	0.0103	0.0103	0.0103	0.0103	0.9897	0.9897	0.9897	0.9897	0.9897	0.9897	0.9897
swim.trace	0.9262	0.9262	0.9262	0.9262	0.9262	0.9262	0.9262	0.0738	0.0738	0.0738	0.0738	0.0738	0.0738	0.0738
twolf.trace	0.9875	0.9876	0.9876	0.9876	0.9876	0.9876	0.9876	0.0125	0.0124	0.0124	0.0124	0.0124	0.0124	0.0124

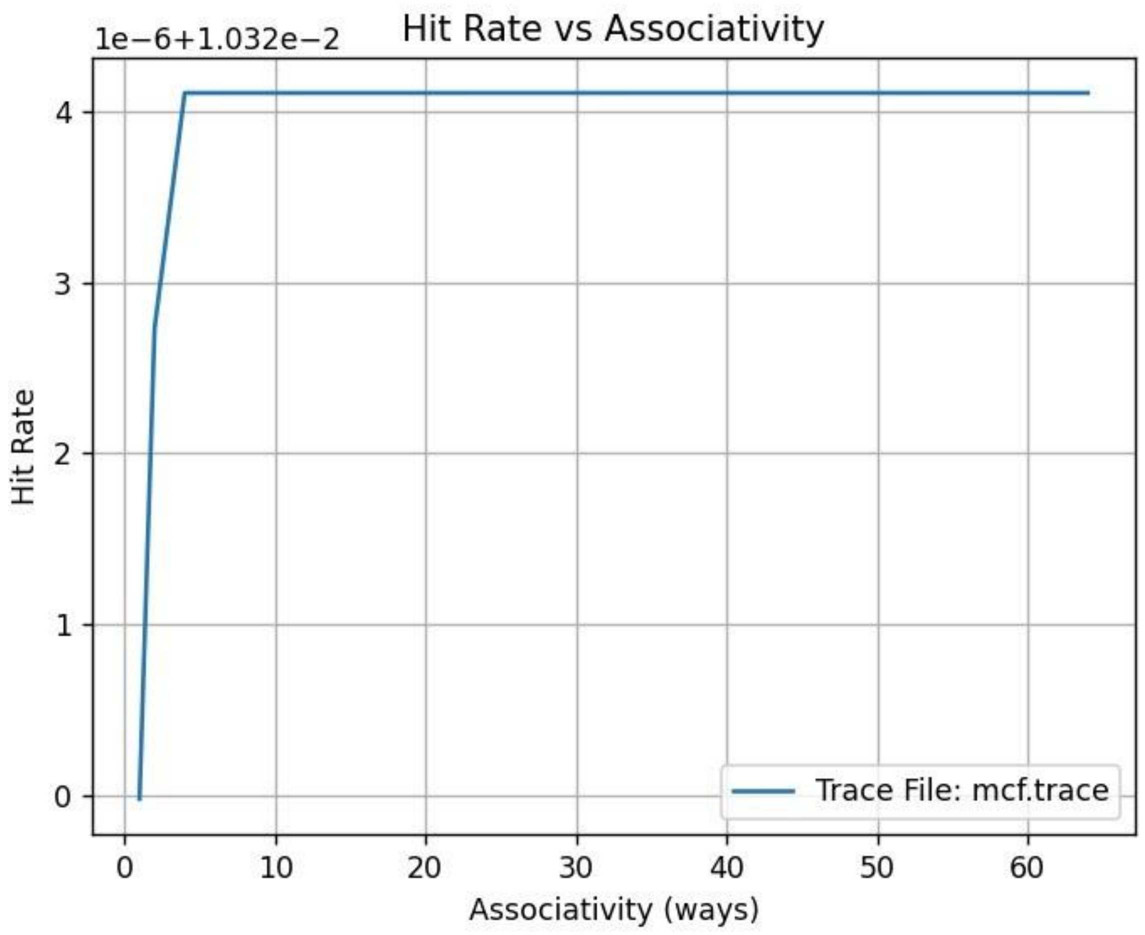
GRAPH FOR GCC:



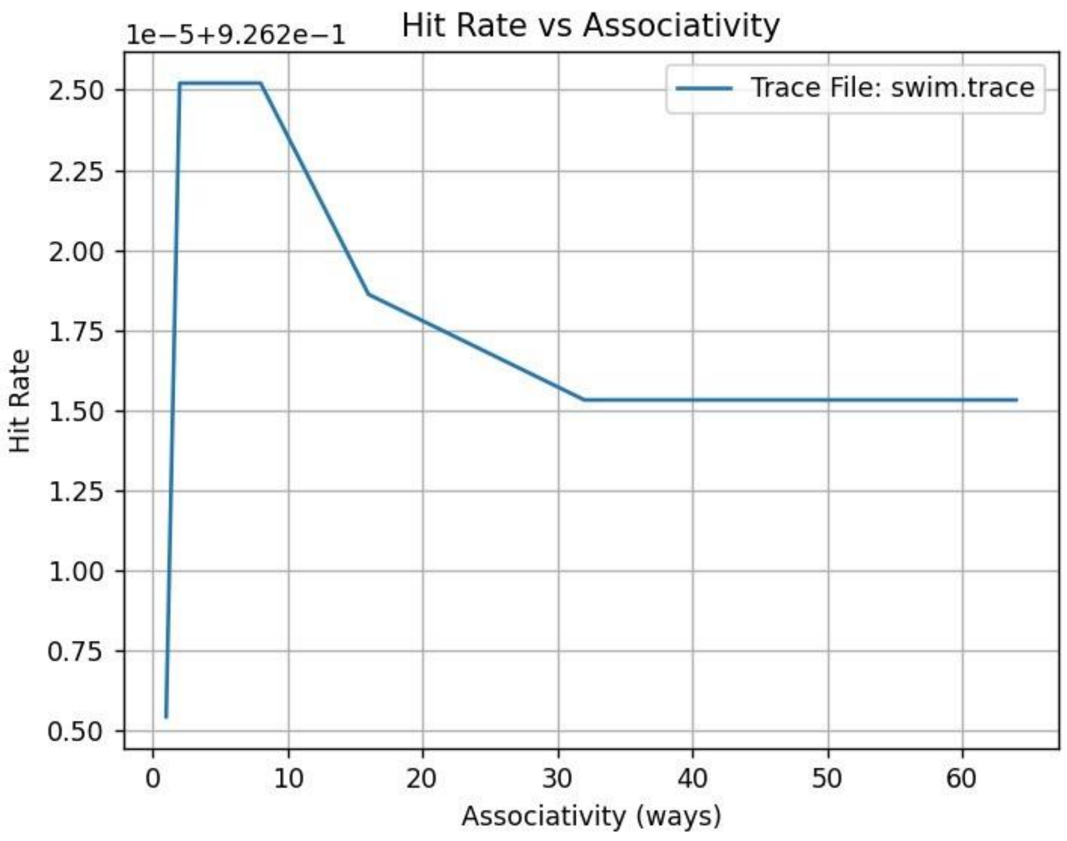
GRAPH FOR GZIP:



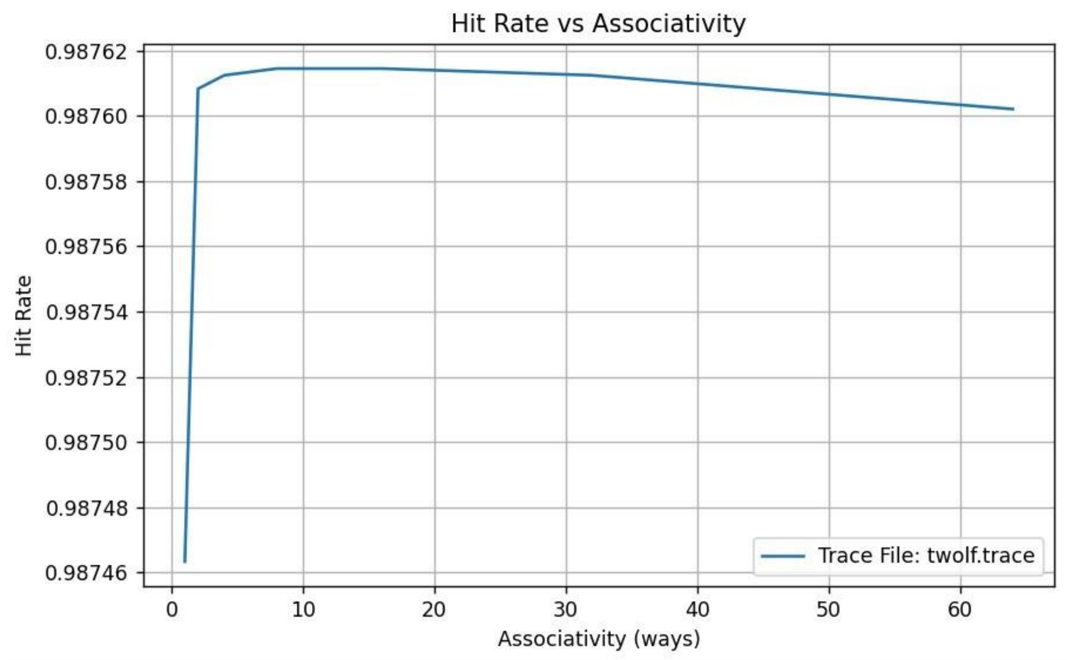
GRAPH FOR MCF:



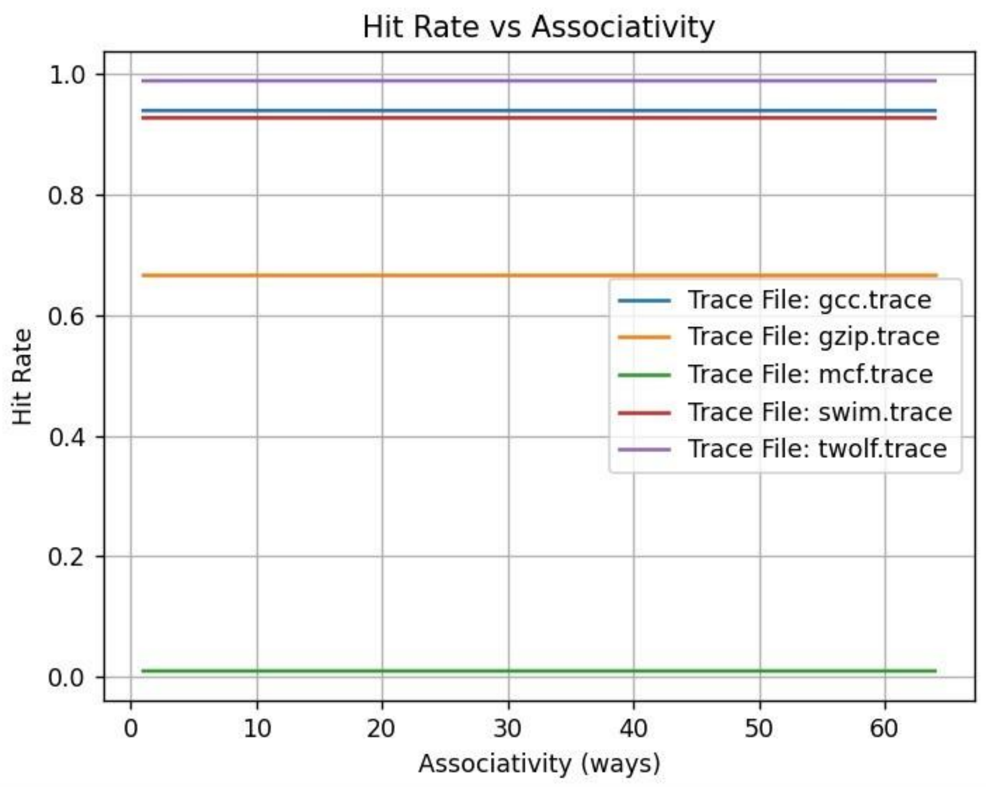
GRAPH FOR SWIM:



GRAPH FOR TWOLF:



CUMULATIVE GRAPH:



OBSERVATION:

- We can see that all traces behave in a similar way.
- Lower associativity generally results in higher miss rates due to increased conflict misses.
- As associativity increases, the miss rate decreases because more cache lines are availaible, reducing conflicts and improving cache utilization.
- Beyond a certain level , the reduction in miss rate becomes minimal.
- twolf.trace has the maximum hit rate and mcf.trace has the minimum hit rate.