NC State University

Department of Electrical and Computer Engineering

ECE 463/563: Fall 2017

Project #1: Cache Design, Memory Hierarchy Design

by

<< Kevin Volkel >>

NCSU Honor Pledge: "I have neither given nor received unauthorized aid on this test or assignment."

Student's electronic signature: _____Kevin Volkel_____
(sign by typing your name)

Course number: _____563_____
(463 or 563 ?)

Abstract:

A cache simulator was constructed in order to explore various memory hierarchy configurations, and the performance that these configurations have for various traces. The simulator was specified to support simulation of a 2 level cache memory configuration that could have a victim cache at the L1 level if it were desired. The purpose of this report is to explore the design space of a 2 level cache memory hierarchy that also uses a victim cache. During the design exploration, parameters of the memory hierarchy are held constant while other parameters are varied in order to determine the effect that the varied parameters have on performance. After exploring the design space and collecting data such as miss rate and average access time (AAT), the optimal configuration was found for each trace. This document presents the results of the design space exploration and offers a discussion about important trends that can be found in the collected data.

Results:

This section presents data that was collected from various cache simulations. For all simulations, the block size was set to a constant 16 Bytes in order to condense the amount of information to be presented. Figures 1 through 4 below show the L1 cache miss rate vs. L1 cache size for various associativities. Each figure is used to show the miss rate for each trace, and the difference in miss rates for the two replacement policies. For these 4 figures, the victim cache and the L2 cache are disabled. Figure 5 below shows the impact of associativity on L1 cache miss rate for each trace using the LRU and LFU replacement policy. For Figure 5 , the L2 cache and victim cache were turned off, and the size of the L1 cache was chosen to be 4096 Bytes. This size of L1 cache was chosen so that the cache size did not mask the improvement that can be achieved with higher associativities. Figure 6 below shows the L1 cache miss rate vs the victim cache size. For these results, the associativity of the L1 cache was fixed constant to 1, and the L1 cache size was fixed to 2048 Bytes. This allows for the victim cache to show how it can improve the miss rate by giving the L1 cache some level of associativity. Figure 7 shows the impact of the L2 cache size on average access time. The L1 cache size was fixed to 2048 Bytes, and the L1 and L2 cache associativities were fixed to 4 because this associativity was found to provide a good miss rate compared to other associativity levels. The main purpose of this graph is to show the L2 cache hit time as the size is increased. The victim cache size was fixed to 1024 Bytes. Figure 8 shows the effect of L1 cache miss rate vs the lambda value for the LRFU replacement policy. The L1 and L2 caches are fixed to 2048 bytes, the victim cache is fixed to 1024 Bytes, and the associativities of L1 and L2 are fixed to 4. Table 1 shows the results of searching all possible L1, victim, and L2 combinations to find the one that yields the lowest AAT for a block size of 16 Bytes. Figure 9 below shows the effect that the write policy has on L1 miss rate for different L1 cache sizes. The associativity is fixed to 4 for this experiment,and the L2 and victim caches are disabled.

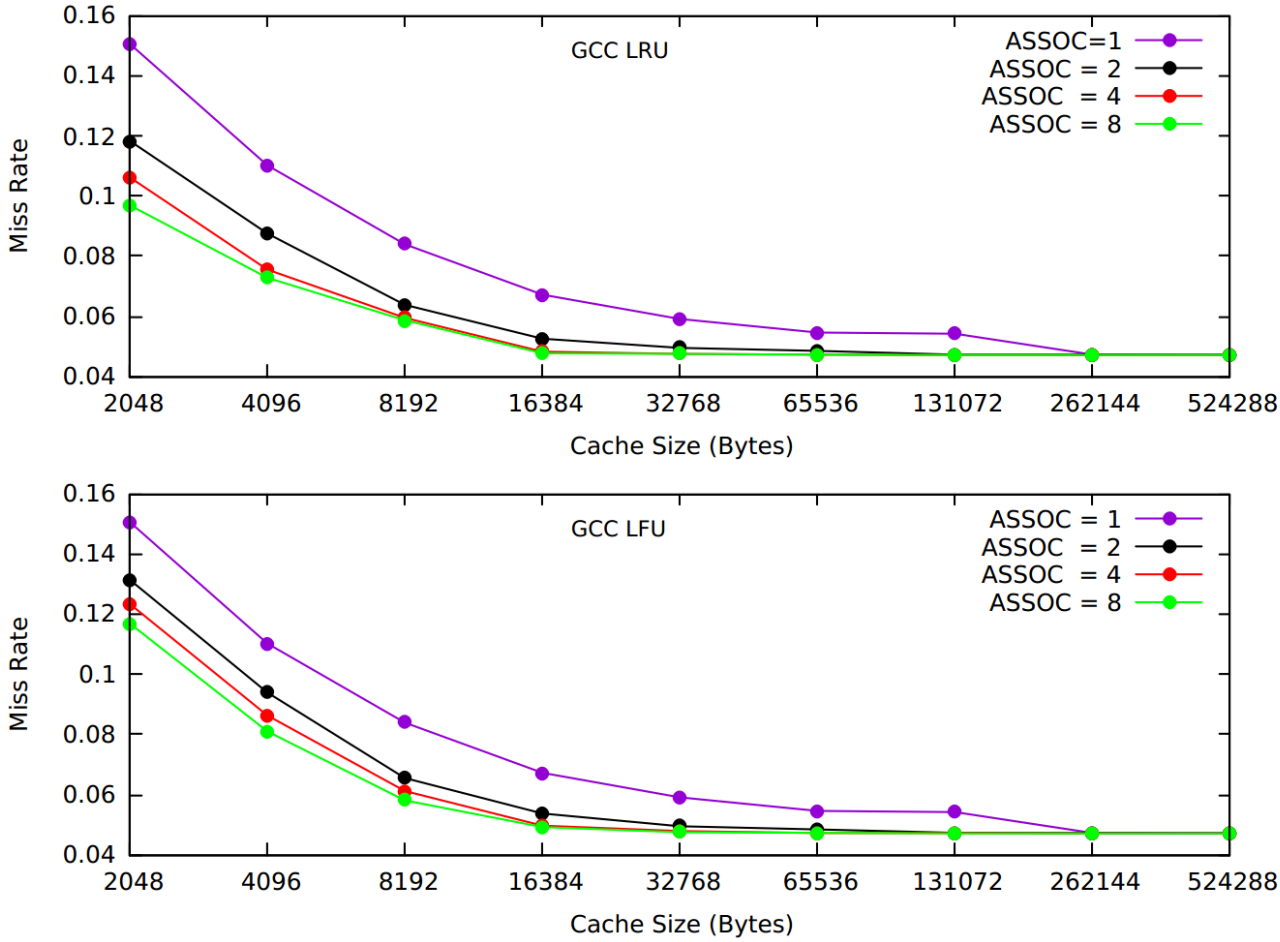

Figure 1: L1 Miss Rate vs. L1 Cache Size for different associativities and different replacement policies for the GCC trace.

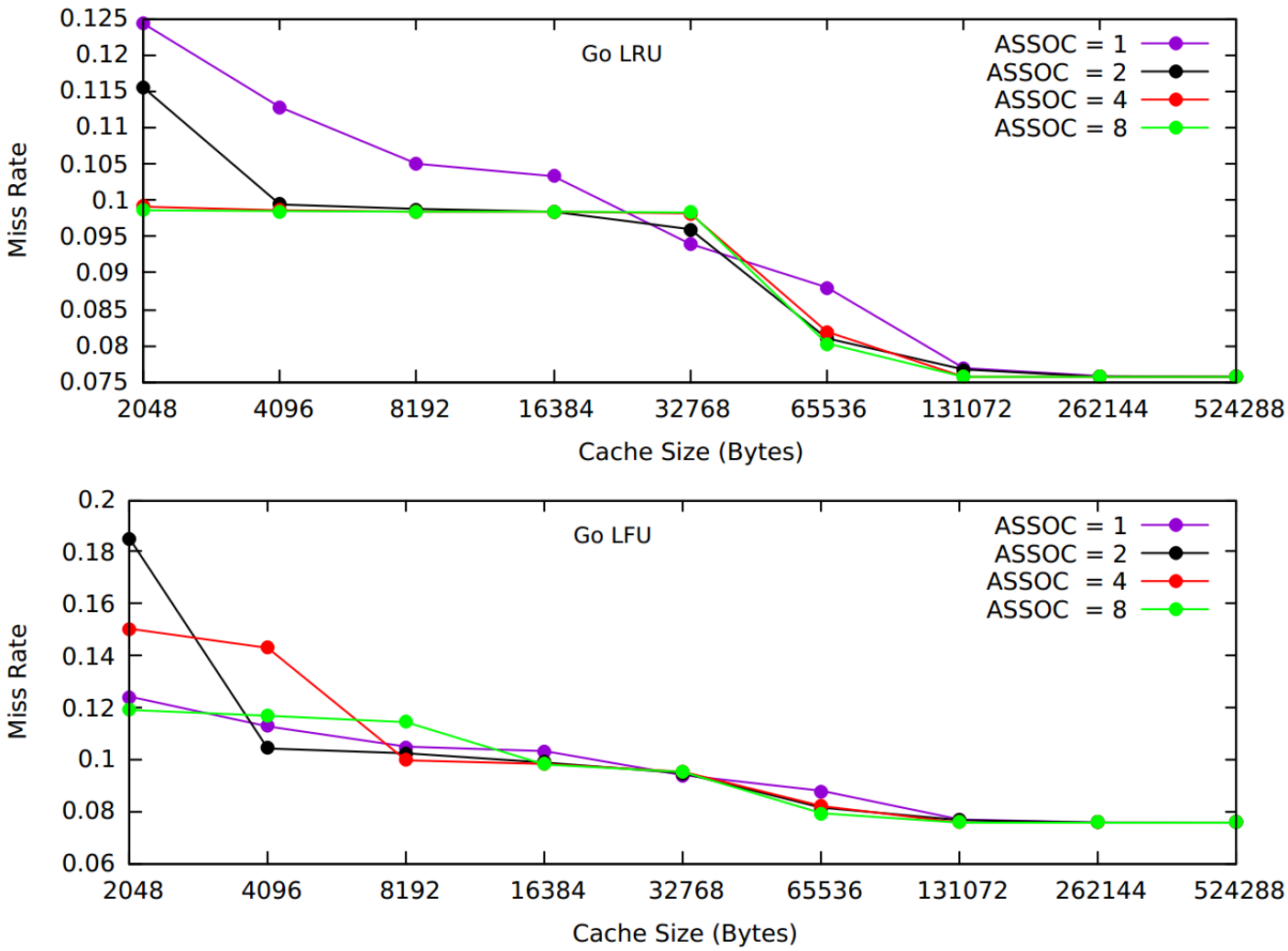

Figure 2: L1 Miss Rate vs. L1 Cache Size for different associativities and different replacement policies for the Go trace.

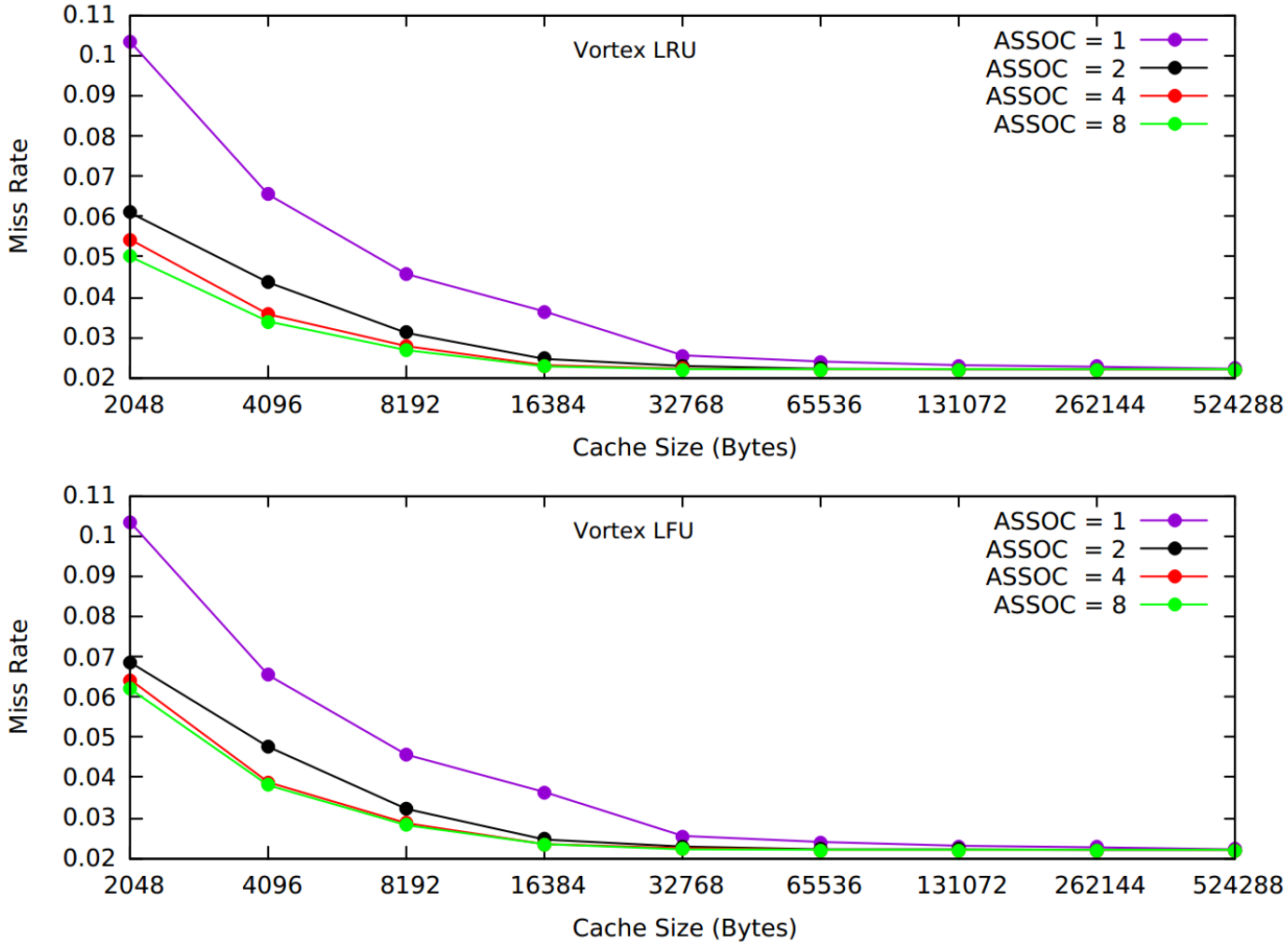

Figure 3: L1 Miss Rate vs. L1 Cache Size for different associativities and different replacement policies for the Vortex trace.

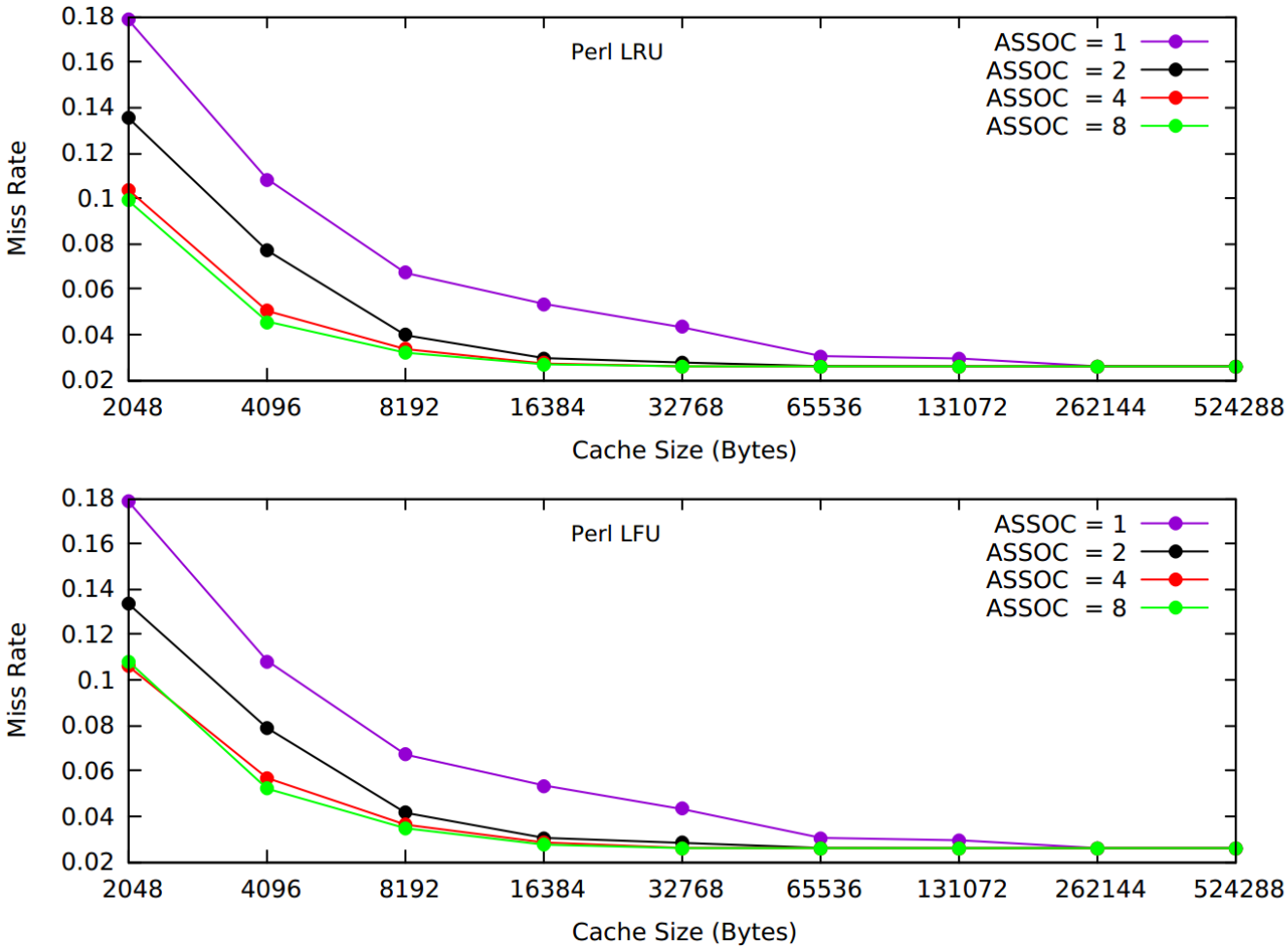

Figure 4: L1 Miss Rate vs. L1 Cache Size for different associativities and different replacement policies for the Perl trace.

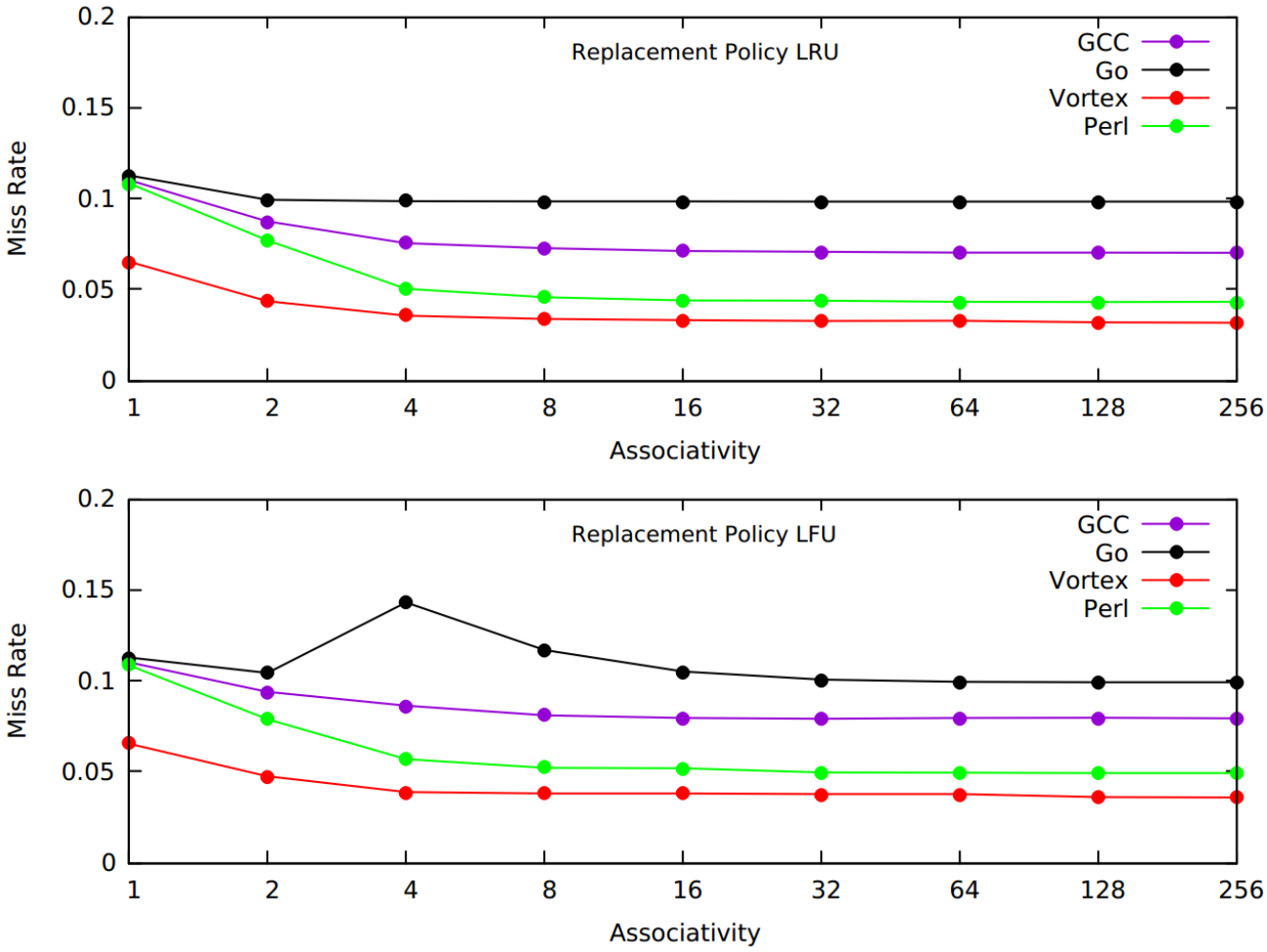

Figure 5: L1 Miss Rate vs. L1 Associativity for different replacement policies and for each trace. L1 fixed to a size of 4096 Bytes.

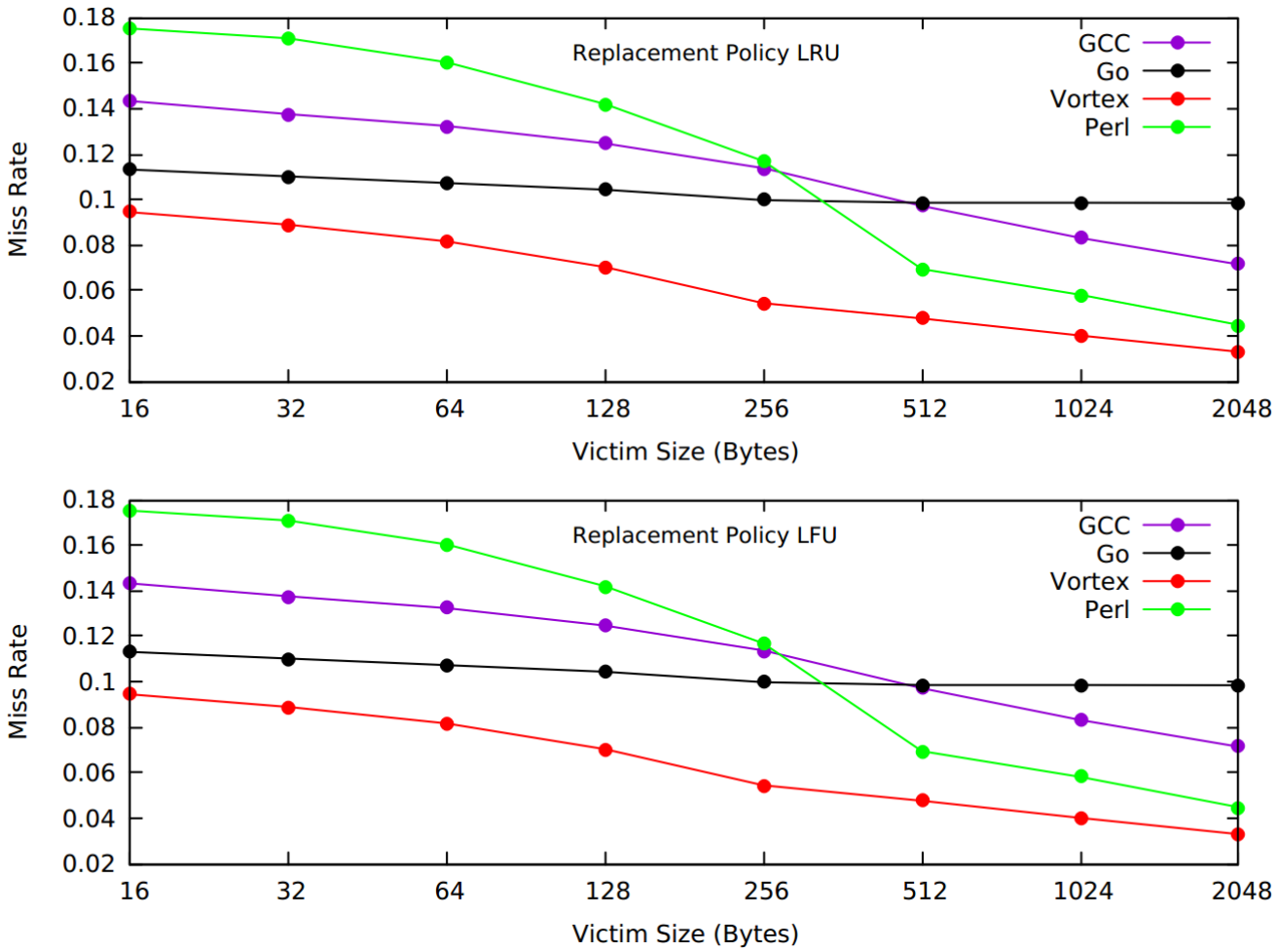

Figure 6: L1 Miss Rate vs. victim cache size for each replacement policy and trace. The L2 cache is disabled, the L1 cache associativity is fixed to 1, and the L1 size is fixed to 2048 Bytes.

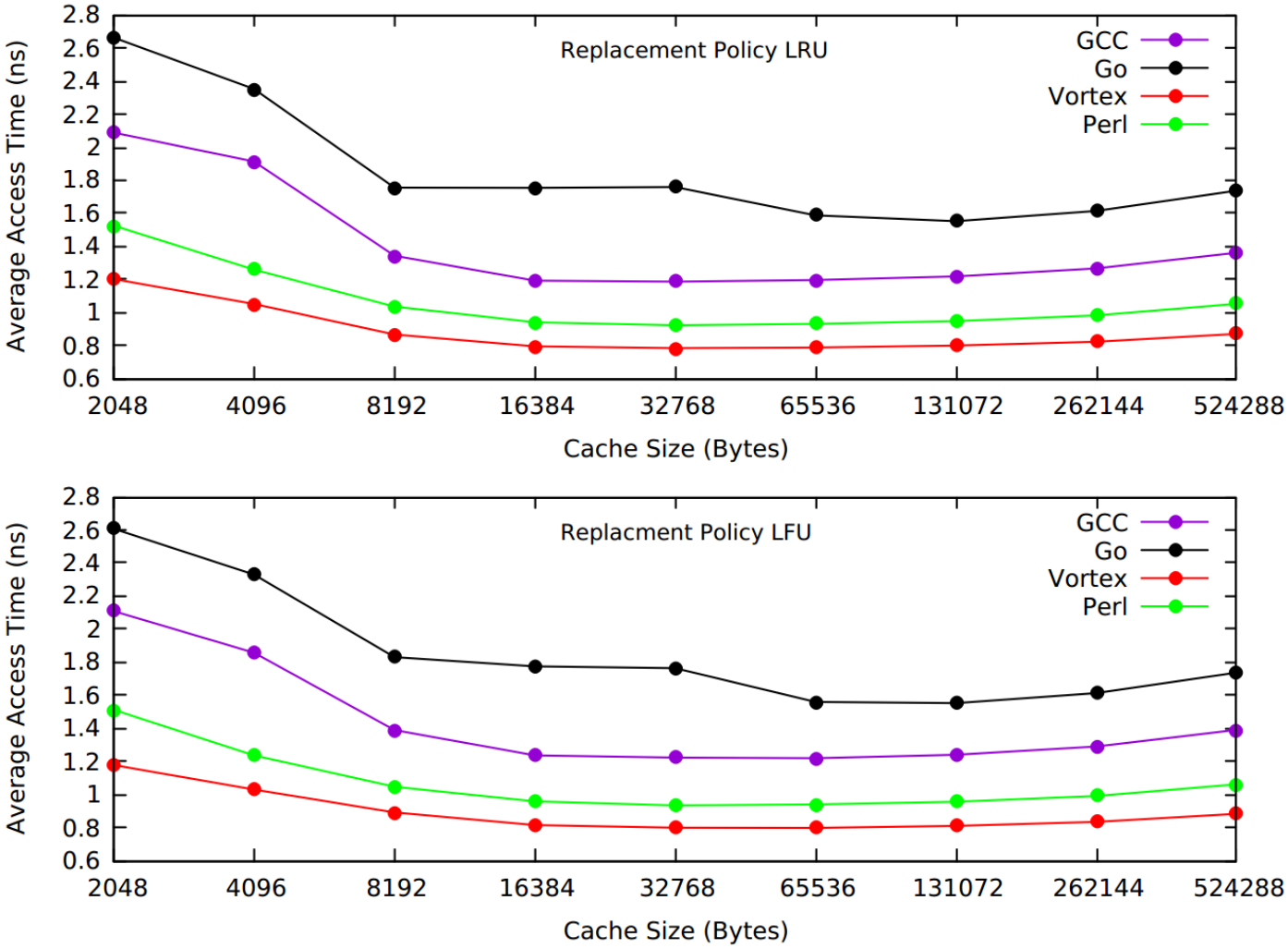

Figure 7: Average Access Time vs. L2 Cache Size for each replacement policy and trace. L1 and L2 are fixed to associativities of 4. L1 is fixed to 2048 Bytes, victim cache was fixed to 1024 bytes.

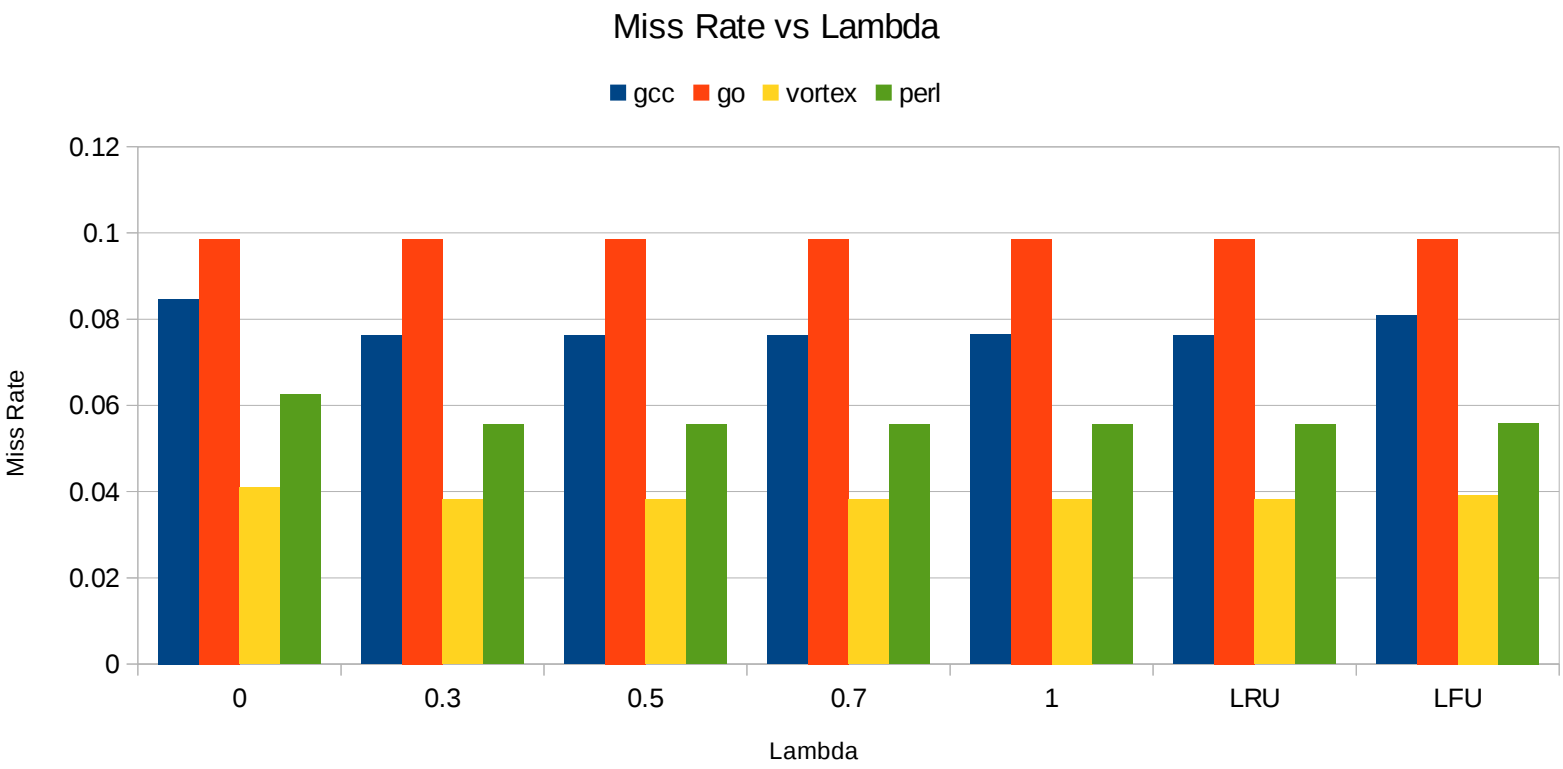

Figure 8: Miss rate of L1 cache vs the lambda value of the LRFU policy.

Table 1: The best configuration for each trace with a block size of 16 B.

Parameter	GCC	Go	Vortex	Perl
L1 Size (Bytes)	2048	2048	2048	2048
L1 Associativity	1	1	1	1
L2 Size (Byes)	32768	131072	32768	32768
L2 Associativity	4	4	4	4
Victim Size	1024	512	1024	1024
Replacement Policy	LRU	LRU	LRU	LRU
AAT (ns)	1.139	1.477	0.7165	0.8558

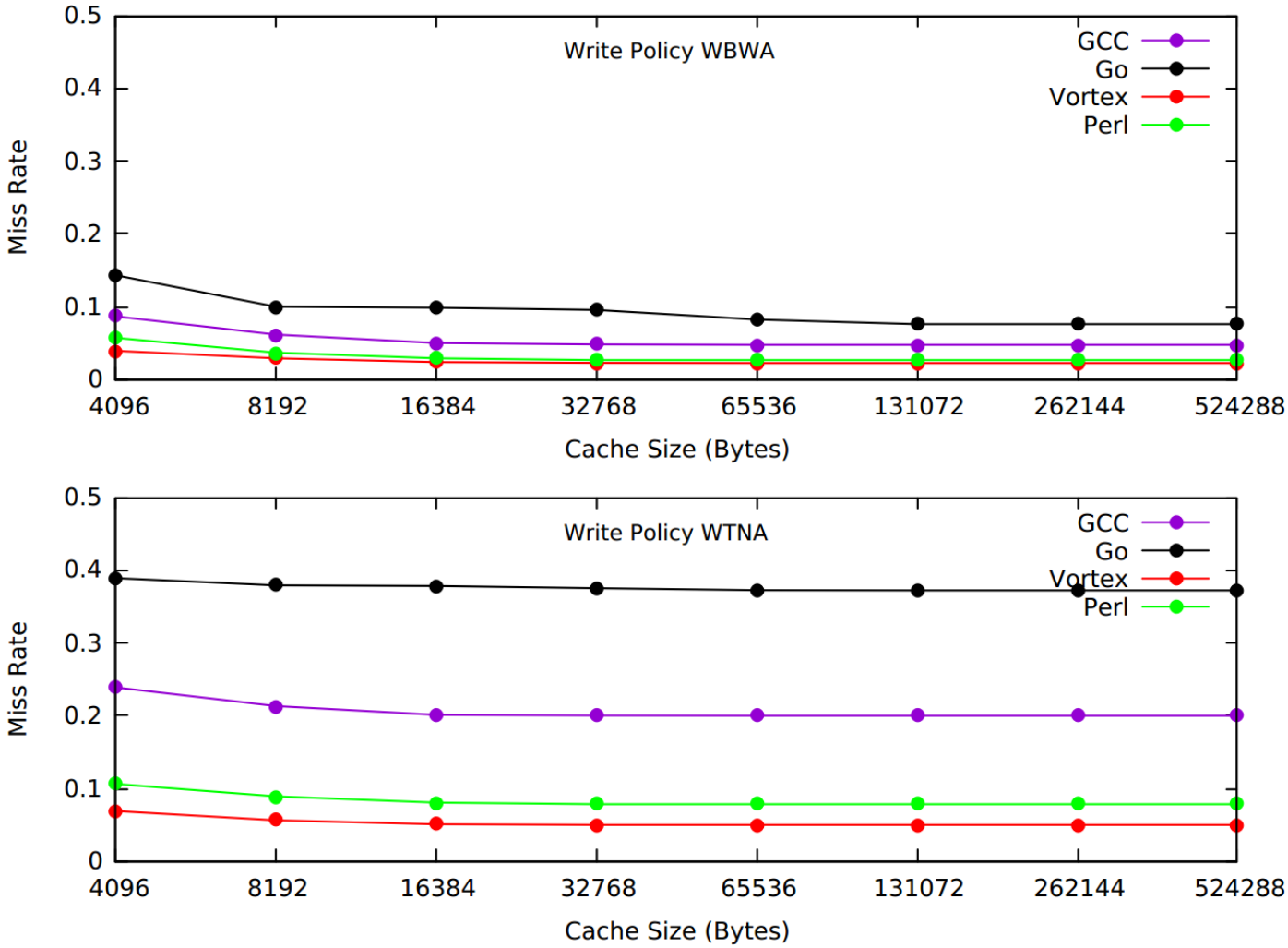

Figure 9: Miss rate vs L1 cache size with a fixed associativity of 4 for the two different write policies. LRU replacement policy was used for both write policies.

Discussion:

This section will discuss the results of the experiments that were conducted with the simulator. The discussion will begin with Figures 1-4, which show the L1 miss rate versus the L1 cache size. From the figures it can be seen that for every trace and associativity, as the size of the cache increases, the miss rate decreases and eventually becomes a constant value which is a result of the misses that cannot be removed (compulsory misses). Also, from these graphs, it can be seen that for most of the traces as the associativity of the cache increases, the miss rate decreases. Although this seems to be the trend, for the Go trace when the associativity is 4 at L1 size 4096 Bytes and with a replacement policy of LFU, the miss rate jumps up well beyond all of the other associativities. Although seemingly counter intuitive, this may be caused by the nature of the address trace. That is, the address trace may have addresses that map to separate blocks with perhaps a low degree of conflict misses occurring when the associativity is 1, and when the associativity changes to 4, these addresses that were mapped to separate blocks could now be mapped to the same set. This could result in the addresses that were just conflicting amongst themselves at the same block to conflict with other new addresses that are mapped to the same new set that is created from the increase in associativity. This problem can also be made more pronounced by the LFU policy because 1 address may be monopolizing a block in the set for a period of time. This would cause the conflicting to become worse because the other addresses would be competing for even less blocks. Although the LFU policy tries to protect against this, it can still happen for a short period of time before another block counter catches up to the block counter that has a high value. This is also seen in the bottom graph of Figure 5 which illustrates that when the associativity increases to 4 for the Go trace, the miss rate spikes to a high value.

Figure 6 presents the effect that the victim cache size has on the L1 cache miss rate. For these simulations, the L2 cache was disabled as it does not affect L1 miss rate. The L1 cache was fixed to have an associativity of 1 in order to observe the associativity benefits that the victim cache offers. For all of the traces besides the Go trace, the L1 cache miss rate drops sharply as the victim cache size increases. This is likely due to the reduction in conflict misses that occur with a direct mapped cache. The fact that the miss rate does not decrease much for the Go trace indicates that this trace suffers mostly from compulsory and capacity misses.

Figure 7 shows the average access time versus the size of the L2 cache. The purpose of this graph is to illustrate how the L2 cache size affects the access time for a fixed L1 cache size and victim cache size. From the graph it can be seen that for every trace the average access time decreases when the L2 cache size increases. This is because the L2 cache miss rate is decreasing as well, resulting in less main memory traffic. Although the AAT decreases, it starts to increase once the L2 cache size reaches a certain point. This happens because most of the conflict and capacity misses are resolved at the L2 level once a certain size is reached, and the minimal subsequent miss rate decreases due to a higher cache size no longer out weigh the overhead of the increased hit time of the large cache. This point is reached at different L2 sizes for each trace, and occurs the latest for the Go trace at 131072 Bytes. This result also points to the idea that the Go trace suffers from mostly capacity misses because a large L2 cache is needed to hold all of the blocks that are referenced by the trace.

Figure 8 shows the miss rate of the L1 cache versus the change in lambda for the LRFU policy. Although not many changes are observed for the miss rate for different lambda values, the miss rates for lambda equal to 0 and 1 match closely to the LFU and LRU results respectively. This is expected because LRFU changes to LFU as lambda approaches 0, and LRFU changes to LRU as lambda approaches 1. The LRFU miss rate does not exactly match the LFU rate because the LFU that results from LRFU is not the same as the dynamic aging LFU. This is because the LFU that results from

LRFU does not include dynamic aging, due to the fact that when a new block is brought in its CRF value is reset to 1.

Figure 9 shows the miss rate versus L1 cache size for the two different write policies. It can be seen that the miss rate is higher for WTNA for every trace. This indicates that not allocating space in the cache when a write is requested does not help the miss rate in these cases. This also means that when a block is allocated for a write with the WBWA policy, that block will be requested again soon after.

The last results to be discussed are in Table 1. This table shows the result of searching for the best configuration for each trace. This search was performed by finding the lowest AAT value from simulating every L1, and L2 cache size and associativity combination for each replacement policy of LRU and LFU, and for each victim cache size. For easy calculations, and for the simulation to work properly, only associativities and sizes that were powers of 2 were used. This ensures that the number of sets calculates to be a power of 2. The max victim cache size allowable is kept to a realistic max of 1024 Bytes. Table 1 shows that for the L1 cache, the optimal configuration for all traces is a size of 2048 Bytes with an associativity of 1. This may be due to the fact that a high associativity is not needed in the L1 level due to the presence of the victim cache. This result also shows that increasing the L1 cache size to a large value does not lead to the optimal design as the extra size increases the hit time to a value that out weighs the benefit of having a larger cache. The L2 cache is the same size for all traces except for the Go trace which has a L2 cache size much larger than the other traces. The associativity of the L2 cache is the same for all of the traces and has a value of 4. The L2 cache size was found to be much larger than the L1 level. This is because at the L2 level, having a larger cache with a larger hit time that has a minimal miss rate is very important because if a L2 cache read or write results in a miss, then main memory needs to be accessed. The Go trace needs a L2 cache size of 131072 Bytes to achieve this, and this result is actually visualized in Figure 7 where the lowest average access time occurs when the second level of cache is 131072 Bytes for the Go trace. This observation about the Go trace also indicates that this trace suffers a great deal from capacity misses as a large L2 cache size is needed to hold the many different blocks referenced by the address trace. The victim cache size is 1024 Bytes for all of the traces except for the Go trace. This indicates that all of the traces benefit from having the maximum victim cache size to reduce conflict misses, except for the Go trace which experiences no performance improvement when the victim cache is increased from 512 to 1024 Bytes. This is yet again a result of the Go trace suffering mostly from compulsory and capacity misses.

The types of misses for each trace can also be observed from the graphs. For example, the Go trace has the highest degree of compulsory misses compared to the other 3. This can be found by observing the miss rate for the largest L1 cache size in Figures 1-4. At this cache size, the miss rate is not decreasing anymore for the traces, which indicates that the only misses left are compulsory. When comparing the miss rates due to compulsory misses among the traces, the Go trace has the highest miss rate. The degree of conflict misses can be observed through the associativity variation graphs. The traces that achieve better miss rates with higher associativities indicate that they can resolve misses by increasing associativity for a given size. This is most easily observed in Figure 5. For each trace (except for Go) the miss rate decreases sharply with an increase in associativity. For the Go trace, a low decrease in miss rate for increase in associativity indicates a low degree of conflict misses. For the traces vortex, gcc, and perl it can be concluded that there is a high degree of conflict misses. The degree of capacity misses can also be determined from Figures 1-4. By observing the point where the capacity increase does not decrease miss rate for the highest associativity is the point where capacity misses are completely or mostly resolved. This point indicates what size of cache is needed in order to remove capacity misses, and a large size indicates that the trace exhibits a large degree of capacity misses due to a large size needed to hold the large amount of different blocks referenced in the address

stream. The Go trace has the highest degree of capacity misses because this point does not occur until a L1 size of 131072 Bytes, much larger than the other traces that have this point occur at about 16384 Bytes.

Conclusion

During the course of this project a simulator was built in order to study many different cache memory configurations. The structural configurations that the cache simulator supported was 1 level of cache, 2 levels of cache, and 2 levels of cache with 1 victim cache available to the L1 cache. Using this simulator, the design space was explored for each trace and the optimal configuration was obtain. On top of obtaining the optimal configurations, important characteristics such as the types of misses were discussed and observed from the reported results.