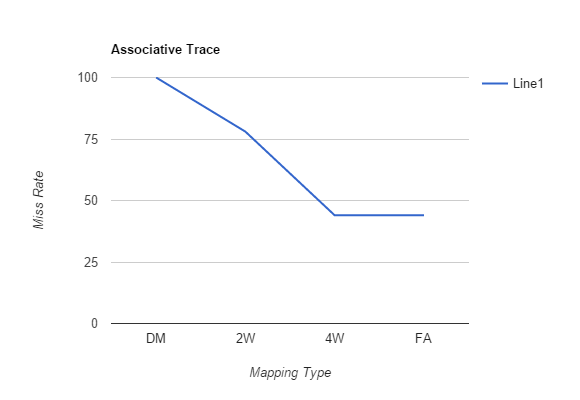
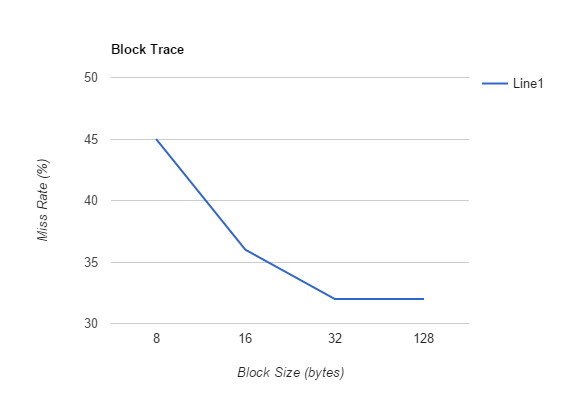
For the implementation of this project I used 3 files to create my cache simulation. cache.h and cache.c were used to creates the necessary data structures for my simulation. The simulation.cpp file was used to handle getting data from the input file to drive the cache simulation. It also outputted the results of that simulation. Within the cache.h and cache.c files, I created 3 classes. One class represented an individual cache line or block, another class represented the sets in the block, and the last classes represented the cache itself. In the line and set class, they only contained data and no functions. The cache class contains information regarding the cache data structure and the functions that will simulate the cache.

Within the cache class I created specific functions to handle the reads and writes for different mapping type. This was mainly used to help debug my function, but for the most part the read and write algorithms for the different mapping types are similar. The basic algorithm for the read functions is that it determines if the tag of the given address “hits” with any other tag in the set. The set that is used to determine the hits is obtained from the given address. Next if the address “hits” with any of the lines if the set, it counts as a hit and increments the hit variable. However, if there are no hits the simulation replaces the appropriate line in the cache and updates the variables for determining what line can or cannot be replaced in the set. The write algorithm is similar. The difference between the two is how the various variables to determine if a line is empty or not and can be replaced or not are updates. However, there is an error in my simulation I could not resolve. When determining the number of times the cache was transferred to memory, this value gave me the wrong result. I unfortunately cannot determine why this was the case.

For this simulation I created two traces to show how different mapping types and block sizes can affect the simulation’s miss rate. Below shows the associativity results for the associative trace:



From this chart it is shown that as the greater the associativity the smaller the miss rate. However, it seems effect of associativity flattens out at some point. Next is the chart the shows the relationship between block size and miss rate.



Similar to the other trace, the block size also decreases the miss rate and also flattens out at some point.