Kevin Shah
Prof. Santosh Nagarakatte
November 30, 2018

Cache Simulator Report

Assignment 4 - Computer Architecture

For the Cache Simulator that I created, I completed **both** the required part and the ***Extra Credit*** part as well. With that being said, I used several data structures for this project. First I created two structures, one structure called "Block" and another called "QueueNode" which both served an important purpose in my simulator. I created the cache and pre-fetch cache as an "array" of Blocks chained using Linked Lists which were used as a hash-table, in a sense that I extracted the index number from the address. I used a Queue in order to apply the FIFO policy so that at any time there is a cache miss, the program enqueues the Tag and the Index of the block. I stored the Index of the block so that when I dequeue from the Queue, I can use "smart dequeueing" in a way that I only dequeue the node that matches the Index that I am working with. For the LRU policy I simply implemented an algorithm that puts the most recently used node at the beginning of the Block linked list and shifts every node one step to the right to remove the Lease Recently Used node. Finally, after careful observation, I came to the conclusion that prefetching actually increases the amount of cache hits, decreases the number of cache misses, and increases the number of memory reads. The reason why this is true is because in one miss the program is prefetching or "predicting" the future and getting another block after it as well. Doing this is efficient because I believe many of the programs written utilize loops and that usually requires getting the block of bytes adjacent to it resulting in cache hits for those two blocks. But after every two blocks in a loop when the cache encounters a miss, the program must fetch two more blocks resulting in two more memory reads. In conclusion, prefetching is not highly efficient when there are a lot of "random" accesses but in simple programs used nowadays, loops are very common and it makes prefetching efficient for most of the time.