COMPUTER ARCHITECTURE
LAB 6: CACHE MISS SIMULATOR
REPORT:

ai22btech11023

Coding Approach :

A 2-D array has been used as the cache. The first argument is the block index and he second is the associativity.

This takes care of all cases, since direct mapped cache is just a 1-way associative cache, and fully associative cache is a cache with all blocks in the same set.

tags[index][associativity], valid[index][associativity], dirty[index][associativity] are used to store the tags, validity, and dirty bit.

There is also a recency[index][associativity] array which stores the latest time at which that part of the cache has been last accessed.

The input is taken as strings. The first character is separated to check if it's a read or write, and then the hexadecimal part is extracted from the string.

After this, this is converted into decimal for easier operating. The decimal value is right shifted by the offset, since these offset bits do not matter for locating the cache block. Then the set is identified by doing dec_val%num_of_sets(index) and tag is identified by doing dec_val/num_of_sets(index)

Then the cache is updated based on the replacement policies.

1) For FIFO: A queue was used, which was updated based on when a block came in
2) For LRU: The recency array was used for what had the least time value for access.
3) For RANDOM: srand() was used to generate random values then rand() to get them.

For WB, the dirty bit is updated when there is a write. If we are replacing a cache block that has an active dirty bit, you initially update the dirty bit to 0, then update it based on whether a read or write instruction was performed.

For WT, if there was a miss, then the cache is not updated. All the other implementation is the same.