Computer Architecture - CS2323. Autumn 2023
Lab-6 (Cache Miss Simulator)

Name:- Syed Abrar
Roll Number:- CS22BTECH11058

Cache Simulation Report:-

Introduction:-
The code implements a cache simulator in C, to replicate the behavior of a set-associative cache system. It allows configuration of cache size, block size, associativity, and replacement/write policies to imitate cache.

Coding Approach:-

Code Structure:-
The code consists of various functions. It uses various approaches such as initialization, address parsing, cache simulation, and memory management.

Structural Representation: C structs are used to define cache blocks, set-associative caches, and the cache structure. This approach is for related data and operations .

Initialization:-
Cache Initialization: This was used set up the cache system based on parameters. These functions allocate memory for cache blocks, handle associativity, and configure the cache's structure according to the specified settings.

Dynamic Memory Allocation: Ensuring proper memory allocation for cache blocks, error handling for memory allocation failures, and freeing allocated memory at program termination to prevent memory leaks.

Cache Simulation Logic:-
Address Parsing and Decoding:
Implemented functions to extract tag and index information from memory addresses. Utilizing bit manipulation and logical operations to parse addresses based on cache configuration parameters such as size, block size, and associativity.

Read and Write Operations: Implemented read and write operations of cache systems. Considering cache hit and miss scenarios, updating cache blocks upon accesses, and managing cache misses by employing replacement policies (LRU, FIFO, RANDOM).

Testing Approach:-

Test Cases:-

Diverse Configurations: Developed multiple test cases consisting of diverse configurations. This included varying cache sizes, block sizes, associativity levels, replacement policies (LRU, FIFO, RANDOM), and write policies (Write-Through, Write-Back).

Edge Cases Coverage: Ensured test cases covered edge scenarios, such as fully associative caches, direct-mapped caches, and different associativity levels and checked the output for each test case, covering a range of read and write operations to thoroughly evaluate cache behavior.

Testing Execution

Verification Process: Executed the code against each test case, examining the output against expected results. Checked the cache hits/misses , replacement logic, and write policies to confirm it.

Debugging: Identified some mistakes between actual and expected outputs, debugged my code to rectify issues