LAB -6 REPORT

CS22BTECH11006

I did my assignment in the C language .I did part 4(all parts).

Coding Approach

1. File Parsing

- The code reads cache configuration details from a configuration file (`cache.config`). It extracts parameters such as cache size, block size, associativity, replacement policy, and write-back policy.

2. Cache Initialization

- Initializes a 2D array to represent the cache, where each entry is initialized to -1.

3. Cache Simulation

- The code simulates cache accesses based on a trace file (`cache.access`). It reads each line, extracts the operation (read or write) and memory address, and processes the cache accordingly.

4. Cache Address Decoding

- Utilizes bit manipulation to decode memory addresses into tag, index, and offset parts.

5. Replacement Policies

- Supports three replacement policies: FIFO, LRU, and RANDOM.

6. Write Policies

- Supports two write policies: Write-Back (WB) and Write-Through (WT).

7. Random Number Generation

- Utilizes the `rand()` function for random replacement and random cache block selection in certain scenarios.

Testing Approach

1. Input Files

- Assumes the existence of two input files: `cache.config` for cache configuration and `cache.access` for memory access traces.

2. File Reading

- The code checks for errors while opening and reading from the configuration and trace files.
- Cache size: max. 1 MB, always power of 2
  Block size: max. 64 Bytes, always power of 2
  Associativity: max. 16, always power of 2
  Input address: input address which is always hexadecimal ,it must be always <=8 bits hexadecimal i.e input address can be 8 bit hexadecimal address or 7 bit hexadecimal address or 6 bit hexadecimal etc.

3. Configuration Parsing

- Parses configuration parameters using `fgets` and `sscanf`.

4. Cache Initialization

- Initializes the cache with default values (-1) for each entry.

5. Cache Simulation

- Simulates cache accesses based on the trace file, checking for hits and misses.

6. Address Decoding

- Utilizes bit manipulation to decode memory addresses into tag, index, and offset parts.

7. Replacement Policy Handling

- Implements FIFO, LRU, and RANDOM replacement policies as specified in the configuration file.

8. Write Policy Handling

- Implements Write-Back (WB) and Write-Through (WT) policies based on the configuration file.

9. Random Number Generation

- Uses `rand()` for random replacement and block selection.

10. Output

- Prints relevant information for each memory access, including address, set, hit/miss, and tag.