Computer Architecture - CS2323
Lab-6 (Cache Miss Simulator)

Waghmare Aditya Abhaykumar
CS22BTECH11061

November 22, 2023

Coding Approach

The program is implemented in C++ for Cache Hit and Miss Simulation and consists of several Classes and functions for this purpose. Here is an overview of the coding approach:

I) `main()` function

`main()` function reads the .config file and creates an instance of `Class Cache` with those configurations. Later while reading the .access file, we use the created Cache Class instance to simulate cache with its `read_access(string Address)` and `write_access(string Address)` functions. This functions Give appropriate Output for each access.

II) `Class Cache`

This is a cpp class to implement read and write functionality of Cache. It consists of Cache Configuration attributes as well as `read_access(string Address)` and `write_access(string Address)` functions to implement Cache functionality. To store blocks it has an array of `Class Row` of size equal to number of sets/indexes in Cache.

a. `read_access(string Address)`

This function maps given Address to Index and calls `read_access()` function of respective `Class Row` instance from the array. After all this it prints output for present read access. This functions also updates hits and misses counter.

b. `write_access(string Address)`

This function maps given Address to Index and calls `write_access()` function of respective `Class Row` instance from the array. After all this it prints output for present write access. This functions also updates hits and misses counter.

III) `Class Row`

This Class has a `vector<string>` attribute to store tags and functions like `read_access()` ,`write_access()` and `fetch_block()` to simulate cache.

a. `vector<string> row`

It is used to store tags. For LRU most recently used tag is placed last, Least recently used is placed first. For FIFO most recently fetched block is placed last and least recently fetched is placed first.

b. `read_access()`

This function searches for the tag. If it is a hit, for LRU the hit tag is deleted and placed last and for FIFO do nothing. If miss, then fetch block using `fetch_block()` function.

c. `write_access()`

This function searches for the tag. If miss and Write Policy is Write Back, then fetch block using `fetch_block()` function(so that it cache block data can be upadated).

d. `fetch_block()`

If number of tags present in vector is less than associativity then fetch block directly to vector. If not then need to replace block. For LRU, least recently used block is placed first in vector so delete it. For FIFO, least recently fetched block is placed first in vector so delete it. For RANDOM, random number generation is used to delete random block. Once deleted fetch new block.

Conclusion

All parts i.e. parts 1, 2, 3 and 4 are supported in this program. The program successfully Simulates Cache hits and misses for different configurations. This program is tested for correctness with test input files cache.config and test.access provided in the zip file of which test.access file was genereted with python script.