Cache Miss Simulator

CS2323 Computer Architecture

Lab Report

Edward Nathan Varghese - CS22BTECH11020

November 24, 2023

1 The Task

For this lab assignment, I implemented a basic set-associative cache in C/C++ with different Replacement Policies (such as `LRU`, `RANDOM`, and `FIFO`).

2 Approach

I chose C++ for this assignment and followed these steps:

1) **Reading from the file:** I used the `<fstream>` header and the `ifstream` function to read content from the `cache.config` and `cache.access` files. Each line was processed individually.

2) **Getting tag, index, and offset:** I converted the hex address to binary and calculated the tag, index, and offset values using the provided formulae for the given data.

3) **Check read/write:** Based on R/W, I passed the respective data to their corresponding functions and returned 1 for a cache hit and 0 for a cache miss.

4) **Implementing LRU, RANDOM, FIFO:** `LRU` and `FIFO` were implemented using a global counter (`timecounter`) tracking the last accessed cache item. Random replacement was achieved using `srand()` with `time(NULL)` as the seed.

5) **Writes:** `Write Back` and `Write Through` policies were implemented in the `cache_write` function, updating cache data in the tags table and last access tables, respectively.

6) **Validation and error handling:** After implementing the program's logic, I handled errors, modularized the code, and added comments to enhance readability.

3 Testing and Validation

I used a set of hex addresses and a random cache configuration to validate my code, verifying its correctness against concepts learned in class. Additionally, I created test cases and used peer test cases to ensure accuracy across various inputs.

4 Learning Outcome

Through this assignment, I learned to simulate a simple cache and about various replacement policies in caches using C/C++, improving my programming skills. It deepened my understanding of computer architecture and organization, providing hands-on experience in file handling, data processing, and cache miss simulation with a modular and efficient code structure.