Cache Simulation Code Report

Gunnam Sri Satya Koushik
Roll.No : cs22btech11026

November 23, 2023

Introduction

In this assignment, I wrote a C program to simulate cache behavior, aiming to provide a comprehensive understanding of how different cache configurations impact system performance.This report presents details of implementation and testing of this program.

The primary objectives of this simulation are to explore the impact of cache size, block size, associativity, replacement policy, and write policy on the cache's performance. By simulating different scenarios and access patterns, we aim to gain insights into the dynamics of the cache and evaluate the effectiveness of different configurations.

Coding Approach

The cache simulation code is implemented in C, reading configuration parameters from a file (`cache.config`) and memory access sequences from another file (`cache.access`). The coding approach includes the initialization of the cache based on specified parameters, simulation of memory accesses, and the implementation of replacement and write policies.

1. The program extracts cache parameters from `cache.config`, storing them in an array for easy access during simulation.
2. The cache is represented as a 3D array, 2D array for cache and another dimension for storing latest access time for each block(which is used for implementing LRU).There is another 1D array which stored number of filled block in each set(useful for quickly determining whether a set is full or not).
3. The code processes each address line by line from `cache.access` and it extracts the address into a unsignened 32 bit int variable and stores whether its a read statement or a write statement in another variable
4. It extracts tag , setindex from address using bit manupulations.
5. Then it checks if that tag is present in the set , if its present it prints hit and continues.If its a miss , if the set if full replacement is done or else it is added in the array directly.
6. Three replacement policies (FIFO, LRU, RANDOM) are implemented to handle cache misses.In FIFO first entry in set is deleted and entire array is shifted left and new entry is added at the end. In LRU latest acces time stored in 3rd dimension is used. In RANDOM a random number is picked from [0,no.of ways - 1] and is replaced with new entry.
7. The program supports two write policies (WriteBack and WriteThrough), adhering to specified allocation policies.
8. Output is printed as specified in problem statement , additionally number of hits and misses are also printed

Testing Approach

The testing approach involves testing with a number of randomised inputs.By using large and randomised inputs , i am making sure all possible cases are covered.I have uploaded the test cases also.

Sample input and output

1 cache.config :

```
8192
256
16
RANDOM
WT
```

2 cache.access :

```
R: 0xAABB
W: 0xCCDD
R: 0xEEFF
W: 0x1122
R: 0xAABB
R: 0x3344
W: 0xCCDD
R: 0xAABB
W: 0xEEFF
R: 0x1122
R: 0x3344
```

3 output :

```
Address: 0x0000aabb, Set: 0x0, Miss, Tag: 0x55
Address: 0x0000ccdd, Set: 0x0, Miss, Tag: 0x66
Address: 0x0000eeff, Set: 0x0, Miss, Tag: 0x77
Address: 0x00001122, Set: 0x1, Miss, Tag: 0x8
Address: 0x0000aabb, Set: 0x0, Hit, Tag: 0x55
Address: 0x00003344, Set: 0x1, Miss, Tag: 0x19
Address: 0x0000ccdd, Set: 0x0, Miss, Tag: 0x66
Address: 0x0000aabb, Set: 0x0, Hit, Tag: 0x55
Address: 0x0000eeff, Set: 0x0, Hit, Tag: 0x77
Address: 0x00001122, Set: 0x1, Miss, Tag: 0x8
Address: 0x00003344, Set: 0x1, Hit, Tag: 0x19
4 hits
7 misses
```

A sample output from the program is provided, showcasing the information presented for each memory access.

Conclusion

The cache simulation code has been rigorously tested, providing valuable insights into the behavior of different cache configurations.