CS2323: Computer Artitecture, Autumn 2023

Cache Miss Simulator

Lab-6

Name: Mayank Parasramka

Roll.no: AI22BTECH11018

Introduction

The provided C++ code aims to simulate the behavior of a cache system based on the specifications outlined in the programming assignment. The simulation involves reading a configuration file (cacheconfig.txt) and an access sequence file (cacheaccess.txt). The cache parameters, access modes (read or write), and memory addresses are used to determine cache hits or misses, as well as the replacement and write policies.

Code Overview

1. Function Definitions:

1.1 powerof2():

- **Purpose:** Determines the power of 2 for a given number.
- **Parameters:** num - the input number.
- **Returns:** The power of 2 for the given number.

1.2 hextobin():

- **Purpose:** Converts a hexadecimal string to binary.
- **Parameters:** inHex - the input hexadecimal string.
- **Returns:** The binary representation of the input hexadecimal string.

1.3 bintohex():

- **Purpose:** Converts a binary string to hexadecimal.
- **Parameters:** binary - the input binary string.
- **Returns:** The hexadecimal representation of the input binary string.

1.4 allocate():

- **Purpose:** Handles cache allocation logic based on replacement policy. Used for both read and write-back with allocation.
- **Parameters:**
    - replacement_policy - the replacement policy (FIFO, LRU, RANDOM).
    - Index - the index of the cache set.
    - Tag - the tag of the memory address.
    - associativity: the cache associativity.
- **Returns:** A string indicating whether it's a hit or miss.

1.5 notallocate():

- **Purpose:** Handles cache access logic when no allocation is required. Used for write through without allocation.
- **Parameters:**
    - associativity - the cache associativity.
    - replacement_policy - the replacement policy (FIFO, LRU, RANDOM).
    - Index - the index of the cache set.
    - Tag - the tag of the memory address.
- **Returns:** A string indicating whether it's a hit or miss.

2. Report on Important Variables in the Cache Miss Simulator Code:

2.1 GLOBAL VARIABLES:

A. CacheTag Array:

- Type: 2D array of strings
- Purpose: Stores cache tags for each set and way.
- Usage: The array is indexed by set and way to keep track of the tags stored in the cache.

B. FIFO Array:

- Type: 1D array of integers
- Purpose: Keeps track of the FIFO replacement policy indices for each set.
- Usage: Used to implement FIFO replacement policy, storing the next index to replace in each set.

C. Clock Array:

- Type: 2D array of integers
- Purpose: Manages the clock values for the LRU replacement policy.
- Usage: Tracks the clock values for each way in each set to determine the least recently used way.

2.2 MAIN VARIABLES:

A. cacheSize, blockSize, associativity, replacementPolicy, writePolicy:

- Type: Integers and strings
- Purpose: Store cache configuration parameters.
- Usage: Used to initialize the cache and determine cache behavior.

B. Address, Tag, Index, Offset:

- Type: Arrays of strings
- Purpose: Store memory address components (tag, index, offset) for each memory access.
- Usage: Used to compute cache indices, tags, and offsets for simulation.

C. **IsRead:**
    - Type: Array of booleans
    - Purpose: Identifies whether each memory access is a read or write.
    - Usage: Determines the type of memory access to apply the appropriate cache policy.

D. **status:**
    - Type: Array of strings
    - Purpose: Stores the status (hit or miss) for each memory access.
    - Usage: Used to print the results of the cache simulation.

E. **FIFO, Clock:**
    - Type: Arrays of integers
    - Purpose: Implement FIFO and LRU replacement policies.
    - Usage: Tracks the next replacement index for FIFO and the clock values for LRU.

Main Idea:

- The code first takes in input from the 2 input files and then implements the cache as per the input action sequences and returning “Hit” or “Miss” accordingly.
- Firstly the hex address is converted to binary and then parsed into tag, index, offset.
- After this as per the configuration inputted the cache is filled, found and replaced accordingly.
- For the cases, read and write back with allocate the same function is called as both these operations updates the cache when “miss” is encountered. Although both are very different when it comes to real world cache implementation they bring no observable change to the given format of output, hence same function call would suffice.
- The write through with no-allocate case has a different function call has it does not update the cache when “miss” is encountered.
- The allocate() function has 3 different upgradation methods based on the replacement policy [LRU, Random, FIFO].
- For LRU replacement, a global clock array is maintained that keeps tracks of last usage of every address.

- For FIFO replacement, a simple queue data structure is implemented using arrays to keep track of last added addresses.
- For Random replacement policy, the address is generated randomly using srand() function.
- Lastly, the associativity of the cache is also taken into account and indexing of the cache is done accordingly.

Test cases:

- The code is tested on a variety of test cases and some of them are attached along with the code.

Output:

- The output is given out in the desired format on the terminal.
- It should be noted that the output for 0x00002d is given out as 0x2d.

Important Remarks:

- The defined SIZE variable is taken to be 3000 which is more than sufficient given the range of inputs in the question. Although it can be changed incase error pops up due to it.
- Please do read the README.md file if facing issues with the execution of the file.

Conclusion:

The Cache Miss Simulator employs a well-structured set of variables and functions to emulate the behavior of a cache system. From the storage of cache tags in the CacheTag array to the management of replacement policies, associativity and write policies, each variable plays a crucial role in realizing the specified cache configurations. Overall, these elements collectively contribute to a robust and functional Cache Miss Simulator, providing valuable insights into cache behavior under varying conditions.