EGRE 426 LAB 5 REPORT

Title: Cache Simulator

Date: 12/05/2023

Author: Bruno da Silva

Pledge: On my honor, I have neither given nor received unauthorized aid on this assignment.

I. Introduction

The goal of this project is to implement a cache simulator that allows configurable parameters such as block size, number of blocks, associativity, and replacement policy. The simulator reads a series of memory addresses and computes the hit and miss rate for the given cache configuration. The main objective is to observe how changes in block size and cache size influence cache hit rates.

II. Lab Content

Lab Summary:

This cache simulator is designed to simulate the behavior of a cache memory system given a series of memory addresses. It supports both direct-mapped and set-associative caches with options for configuring block size, number of blocks, associativity, replacement policy (random or LRU), hit time, and miss time. The primary functionality is to compute the hit/miss rate and total access time for a given set of memory accesses.

Testing:

These testing cases were created to cover the given scenarios on the lab sheet in order to ensure the correctness and reliability of the simulator.

1. Direct-Mapped Cache:
   - The “DirectMappedCache” class represents a direct-mapped cache with a specified size.
   - For each memory access in the sequence (“memory_accesses” list), the code calculates the cache index based on the address.
   - If the accessed address is already in the cache at the calculated index, it is a hit. Otherwise, it is a miss, and the address is stored in the cache at the calculated index.

- At the end of the simulation, the code prints the number of hits, misses, and the contents of the cache.

2. Two-Way Set-Associative Cache with LRU Replacement:
- The “SetAssociativeCache” class represents a two-way set-associative cache with a specified number of sets and cache size.
- For each memory access in the sequence, the code calculates the set index based on the address.
- If the accessed address is already in the cache in the calculated set, it is a hit. Otherwise, it is a miss.
- If the set is not full, the address is added to the cache in the set. If the set is full, the Least Recently Used (LRU) replacement policy is applied to make room for the new address.
- At the end of the simulation, the code prints the number of hits, misses, and the contents of each set in the cache.

Test Case 1: Direct-Mapped Cache Expected Output

- Hits: 2
- Misses: 7
- Cache Contents:
  - 0: 16
  - 3: 3
  - 5: 21
  - 11: 11

Test Case 2: Two-Way Set-Associative Cache Expected Output

- Hits: 3
- Misses: 6
- Cache Contents:
  - 16: 1
  - 48: 1

- 3: 0
- 11: 1
- 21: 0

How it works:

The core component is the “Cache” class, which represents the cache memory. The class contains methods to simulate memory accesses (access_memory), calculate and print results (print_results), and handle cache organization based on specified parameters. For each memory address, the simulator calculates the block address and determines cache hits or misses based on the cache organization. The cache is organized in sets, and eviction strategies (LRU) are applied when needed. The simulator uses command line arguments to receive input parameters, making it configurable for different cache setups. The Extra Credit portion was done and is an option on the command line. Users can choose random and LRU policies if they need it for a desired cache setup.

Command Line Options:

--block-size: Specifies the block size in words.
--num-blocks: Specifies the number of blocks in the cache.
--associativity: Specifies the cache associativity (1 for direct, 2 for two-way set associative).
--hit-time: Specifies the cache hit time in cycles.
--miss-time: Specifies the cache miss time in cycles.
--replacement-policy: Specifies the replacement policy (random or LRU).
--datafile: Specifies the path to the given input file containing hex addresses.

III. Conclusion

The cache simulator successfully demonstrated the impact of cache configuration on hit and miss rates. The results demonstrate that an increased number of cache size and block size

helps reduce the cache miss rate, as expected. The cache simulator provides a flexible and functional tool for understanding and analyzing the performance of different cache memory configurations. This report summarizes the cache simulator assignment, describing its functionality, testing procedures, and overall process. The simulator proves to be a valuable tool for studying the impact of cache configurations on memory access performance.