Project 2 Report

Assembly

Ahmed Jaheen - 900212943
Adham Samy - 900213258
Kyrollos Zakaria - 900203309

Submitted to Dr. Nourhan Zayed
This assignment is prepared for CSCE2303, section 2

Computer Science and Engineering Department
The American University in Cairo
19/05/2023

Implementation Brief Illustration

The implemented C++ code simulates a cache memory system using a direct-mapped cache organization. It takes input parameters such as cache size, line size, and access time from the user. The code reads a memory access sequence from a file and simulates cache hits and misses based on the given sequence. It tracks the number of accesses, hits, misses, and calculates hit ratio, miss ratio, and average memory access time (AMAT) for each step of the simulation. In details:

(i) **Data Structures:**

- **CacheLine Struct:** Represents a cache line and consists of two members:
  (a) **valid:** A boolean value indicating whether the cache line is valid or not.
  (b) **tag:** An unsigned integer representing the tag of the memory block stored in the cache line.

(ii) **Function: Cache_Simulation**

- **Parameters:**
  (a) **filename:** A constant reference to a string that specifies the filename containing the memory access sequence.
  (b) **S:** An unsigned integer representing the cache size in bytes.
  (c) **L:** An unsigned integer representing the line size in bytes.
  (d) **Time:** An unsigned integer representing the access time in clock cycles.
- **Description:**
  (a) Calculates the number of cache lines based on the given cache size and line size.
  (b) Creates a vector of struct called **cache** to represent the cache memory with the calculated number of cache lines.
  (c) Initializes variables to track the number of accesses, hits, and misses.
  (d) Opens the input file specified by **filename** to read the memory access sequence.
  (e) Enters a loop to process each memory address in the sequence which increment number of access, increment hits if the condition satisfied and increment miss otherwise with outputing each of the valid bits and tags of all cache entries, the total number of accesses, hit ratio, miss ratio, and AMAT (Average Memory Access Time) based on each memory statistics.
  (f) Closes the input file.

(iii) **Main Function:**

- Prompts the user to enter the cache size, line size, and access time.
- Validates the input for the access time to ensure it falls within the range of 1 to 10.
- Calls the **Cache_Simulation** function with the provided input parameters.

Design Decisions and Assumptions

1. The cache is implemented as a vector of CacheLine structures, representing the cache lines.
2. Each cache line has a valid bit indicating if the cache line is valid or not, and a tag to identify the memory block stored in the cache line.
3. The cache is direct-mapped, which means each memory block maps to exactly one cache line determined by the cache index.
4. The cache index is calculated as the memory address divided by the line size, modulo the number of cache lines.
5. The cache tag is calculated as the memory address divided by the product of the cache lines and the line size.
6. The cache simulation assumes that cache lines are initially empty (valid bit set to false) before any memory access.
7. The cache is updated with the new memory block when a cache miss occurs.

User Guide

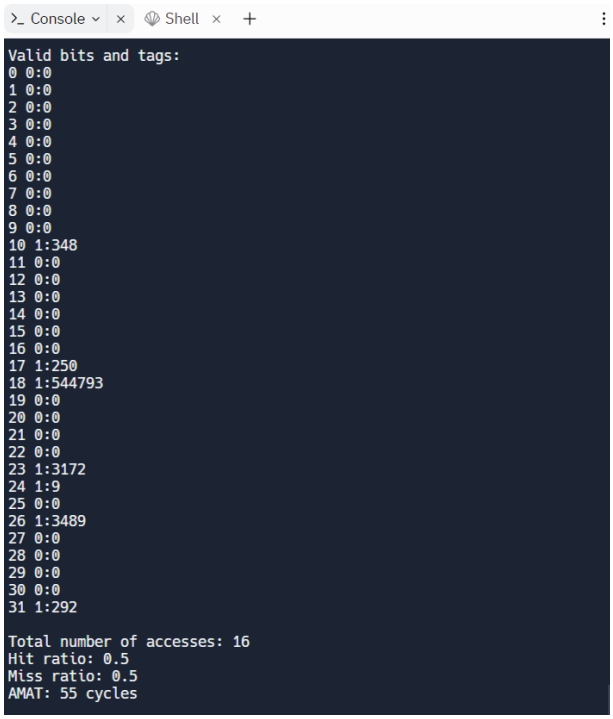
The project is written in $C++$, allowing you to compile it using any C++ compiler. Once the compilation process is complete, you have to edit **access_sequence.txt** with your memory addressing file where you have to make sure that the **access_sequence.txt** is present in the same directiory as the .cpp file.
To initiate the program, follow these steps:

1. Firstly, the user enters the cache total size in bytes.

2. Secondly, the user enters the size of each line (block) in bytes.

3. Thirdly, the user enters the clock cycles needed to access the cache.

4. Finally, the user will see the memory caching process step by step and the final picture of the cache, which will look like this.



List of sequences simulated

You can find in the full submitted project, the list of test cases' files with 20-access sequences.