The American University in Cairo

Computer Science and Engineering Department

Cache Simulator Project Report

Gehad Salem 900205068
Mario Mamdouh 900202178
Ahmed Asaad 900201621

❖ Brief Description of Implementation

The goal of this project is to implement a read-only cache simulator to keep track of the cache entry valid bit and tag bits in addition to the total number of accesses, hit ratio, miss ratio, and the average Memory Access Time (AMAT) of the memory hierarchy (in cycles). Our program has 2 separate caches: one for data and the other for instructions. Additionally, the simulator supports direct mapping, set-associative, and full associative caching systems. The implementation language used in this simulator is C++. Moving on to the sequence of the implementation, we validate the input: cache size, line size, associativity level, cache organization, and the number of cycles needed to access the cache. Cache size and line size are validated to be positive integers that are powers of 2, and the line size is validated to divide the cache size. The associativity level is validated to be a positive integer, a power of 2, and less than or equal to the number of lines in the cache. Furthermore, the program checks if the cache organization is not any of the three inputs specified for each organization. Lastly, we validate the number of cycles needed to access the cache is an integer between 1 and 10 clock cycles. Here comes the last part in the brief description of the implementation, which is the logic used to update the cache, and data structures used. Since the program supports two separate caches, we designed an array of vector of vector of pair to represent the cache. The array size is 2: index 0 is for the instruction cache, and index 1 is for the data cache. The vector of vectors is to represent the available positions inside different sets. Moreover, the pair first is of type boolean for the valid bit, and the pair second holds the tag value. Finally, we keep track of the number of hits, misses and accesses, which are used later in the output to calculate total accesses, hit ratio, miss ratio, and Average Memory Access Time (AMAT) of the memory hierarchy (in cycles).

❖ **Design Decisions:**

When implementing the bonus of the 2 separated caches, we chose to design it as a one-data structure: an array of vector of vector of pair to represent the cache. The array size is 2: index 0 is for the instruction cache, and index 1 is for the data cache. The vector of vectors is to represent the available positions inside different sets. For the pair data structure, it was implemented as a <bool, int> to hold the valid bit, and the tag value.

❖ **Assumptions**

- ➢ The input file should have a specific format by entering first a number corresponding to whether the access is for data or instructions and then the memory address.
- ➢ The label (data or instruction) and memory address have to be on the same line (it doesn't matter the number of spaces).
- ➢ The memory addresses are assumed to be written correctly in the file with no negative values and can fit into a 32-bit integer.

❖ **Known Bugs or Issues**

After running the program dozens of times, we have found the following bugs/issues:

- ➢ An error occurs if the memory address is a number greater than (2^32 - 1), as overflow will occur, and this doesn't fit in the 32-bit memory.
- ➢ An error occurs if the memory address is negative, as this doesn't make sense.
- ➢ An error occurs if the label (instruction or data) and memory access aren't in the same line.

❖ User Guide

The Cache simulator is capable of tracing how the cache works and how the processor and cache communicate.

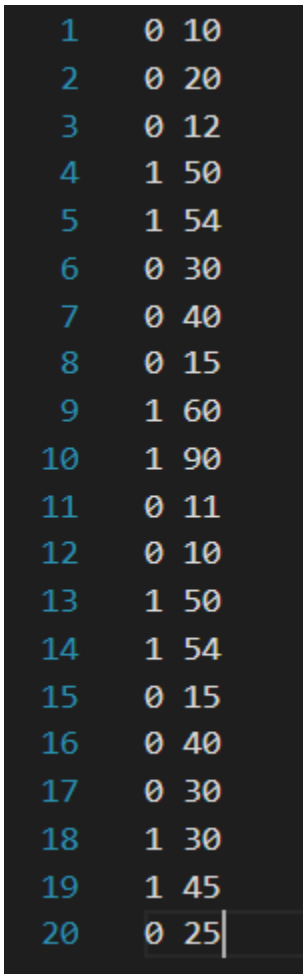
This user guide will show you how to compile and run this Cache simulator.

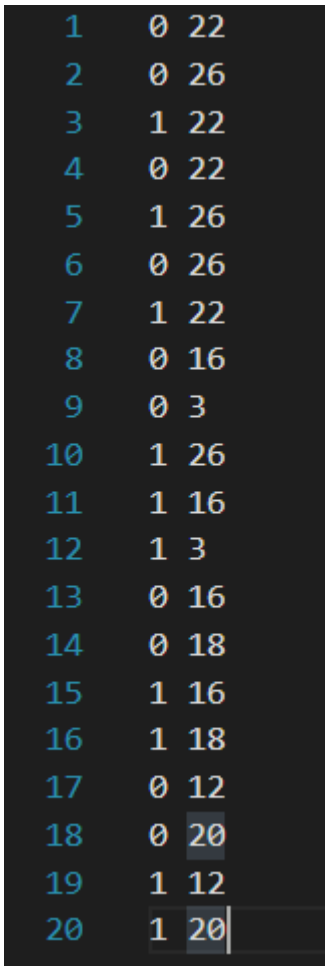
First, you should download and open the program with all its files on your respective Integrated Development Environment (IDE).

```
gehadsalemfekry@W10CNFLQG3:~/projects/Cache_Simulator$ g++ main.cpp
gehadsalemfekry@W10CNFLQG3:~/projects/Cache_Simulator$ ./a.out
Please, enter the cache total size in bytes, and the cache line size in bytes: 8 1
Please choose a cache organization:
D: direct mapping
F: full associativity
S: set associativity
D
Enter the number of cycles needed to access the cache (an integer between 1 and 10  clock cycles): 5
Please enter the file containing the memory addresses (in bytes) you want to access using this simulator.
program2.txt
Please enter the file you want to have the output in.
output2.txt
```

Upon running the program, the user will be asked to enter the cache size and line size with all the necessary validations upon input (that the numbers have to be positive and some power of 2). Then, the user should choose which cache organization he wants to simulate and the clock cycles for accessing the cache. After that, he will be asked to enter the name of the file containing the memory addresses with their labels corresponding to data or instructions. For this example, the user entered “program2.txt”. Additionally, the user will be asked to enter the name of the output file; here, he entered “output2.txt”.

Here are the data in the input files:





Then, the program will process all this input and simulate how it works. After each memory access, the content of the cache of both instructions and data is outputted along with the total accesses made to the cache, the hit ratio, the miss ratio, and the average memory access time (AMAT) of the memory hierarchy, allowing the user to see each change taking place step by step, with the tag displayed in binary and the cache entry index. The output will then be saved and placed in a file named “output2.txt”. Below is a screenshot of the contents of this file:

```
--------------------------------------------------------------------------------
Data Access with Address 26
---------------------- Instructions Cache Information: -------------------------
Valid Bit   Tag     Index
0           0       0
0           0       1
1           11      2
0           0       3
0           0       4
0           0       5
1           10      6
0           0       7
------------------------ Data Cache Information: ------------------------------
Valid Bit   Tag     Index
0           0       0
0           0       1
0           0       2
0           0       3
0           0       4
0           0       5
0           0       6
0           0       7
--------------------------------------------------------------------------------
Total Accesses are: 2
The Hit ratio is: 0
The Miss Ratio is: 1
The Average Memory Access Time (AMAT) of the memory hierarchy (in cycles) is: 105
--------------------------------------------------------------------------------
```