CSCE2303 – Computer Organization and Assembly Language Programming
Spring 2024

Project 2: Memory Hierarchy Simulator

Project Overview: The goal of this project is to implement a simple simulator for the memory caching system. This document details the requirements of the simulator.

Implementation language: Any general-purpose programming language. The resulting application can either be a console application or a graphical user interface (GUI) application (desktop, web-based, or mobile app) as a bonus feature.

Team Size: 2 or 3 students

The memory hierarchy: The basic requirements for the project assume a simplified ***read-only one-level*** caching system that uses ***direct mapping***. The underlying memory address space is assumed to be a 30-bit byte-addressable memory (1 GB). Accessing the memory is assumed to take 150 clock cycles.

Simulator inputs:

1. *Cache Information:* The total cache size ***S*** (in bytes), the cache line size ***L*** (in bytes), and the number of cycles needed to access the cache (e.g., an integer between 1 and 10 clock cycles).
2. *Access Sequence:* A sequence of memory addresses that are accessed by a certain program. Addresses should be given in bytes. This sequence should be provided as a text file containing a comma separated list of byte addresses.

Simulation and simulation outputs: The simulator should trace the cache behavior given the input sequence of memory accesses by keeping track of the cache contents after each access (by updating the valid bits and tags associated with each cache line. The actual cache content is irrelevant). The cache is assumed to be initially empty. The simulator should also keep track of the number of accesses, the number of hits, and the number of misses. After each memory access and until the sequence ends, the program should **output the following:**

- Valid bits and tags of all entries
- Total number of accesses
- The hit and miss ratios
- The Average Memory Access Time (AMAT) of the memory hierarchy (in cycles)

***Bonus* features:**

To get bonus marks, you can implement up to 2 bonus features from the list below:

1. Building the application as a GUI application (either desktop, web-based, mobile app).
2. Supporting set and full associativity. In this case, in addition to the cache information listed above, the user will need to specify the associativity level.
3. Supporting multi-level cache hierarchies. In this case, the user of the simulator should specify the number of cache levels to simulate. For each cache level, the user will also need to specify: The total cache size S, the cache line size L, and the number of cycles needed to access the cache.
4. Supporting separate caches for instructions and data. In this case, the user will provide 2 access sequences: One for instructions and one for data.
5. Supporting a read-write cache instead of just a read only cache. In this case, the user will need to specify the writing policy used (both in case of hit and in case of miss). Also, each of the memory addresses in the provided sequence must be labelled as either a read or write access.

General Guidelines

- Every group member must log all her/his activities in a journal (a text file). A journal entry may look like the following:

```
May 9, 1:55PM: implemented a function to compare the tag of a memory
       address with that of a cache entry.
```

- Only one member of each group should submit on **Canvas**. The submission should consist of a single compressed folder (**zip** or **rar**) which must include the following:
    - A journal folder that contains the journal file of each team member.
    - A source code folder that contains the code you wrote using your chosen programming language.
    - A test folder that contains all input files (sequence files) used for testing.
    - A **PDF** report that containing the following
        - Students' names and IDs
        - A brief description of your implementation including any bonus features included.
        - Any design decisions and/or assumptions you made.
        - Any known bugs or issues in your simulator.
        - A user guide showing how to compile and run your simulator including a full simulation example step-by-step with screenshots.
        - A list of sequences you simulated along with corresponding outputs. You should at least provide two 20-access sequences.
        - Optionally, you can include a section about your experience working on this project. This section will NOT affect your grade in any way.

Important Plagiarism notice: You must write your own code from scratch. Submissions based on others code will receive a grade of **zero** in the entire project and will be reported as an academic integrity violation (even if you understand every code line and even if the code is heavily re-factored/modified). Examples of such sources include (but is not limited to) code coming from the following sources: other teams, previous course offerings, open-source software, tutors, Internet, etc.

Deadline: You are required to submit the full project on **Thursday May 16, 2024 (11:59 pm)**. **Demos will be scheduled on May 19 and May 20.**

Grading Criteria

- This project is equally weighted with project1. So, it will account for **15%** of the course marks.
- **Bonuses**: Each bonus feature will count for 5% with a maximum of 2 bonuses worth 10%. Please do not be tempted to implement more than 2 bonus features, as this will cost you too much time.
- Deductions:
    - **-5%** for not following the required submission directory structure.
    - **-5%** per day for late submission (two days maximum).
    - **-100%** for plagiarized submissions.
- Group members may receive different grades based on their contribution to the project.