INSTITUTO SUPERIOR TÉCNICO

Departamento de Engenharia Informática

Computer Organization

LEIC-A, LEIC-T

Second Lab Assignment: Cache Simulator

Version 1.0

2023/2024

1 Introduction

The goal of this assignment is the development of a complete memory hierarchy. To achieve this, the students will complete the base code provided by the faculty to implement the L1 cache and L2 cache from scratch. Finally, the students will integrate these components to form a complete memory hierarchy.

2 Development Environment

The simulator is to be developed in C and target x86-64 Linux. The only dependencies allowed are those provided by `glibc` (`stdio.h`, `stdlib.h`...).
It should be built with `make` or `make all` without any warnings. For testing purposes, it should also run in the lab computers (in Linux).
Students are free to modify the `Makefile` and add more build targets during the development of the simulator, but the `CFLAGS` must remain the same.

3 Code Provided

The source code provided is composed of 4 files: `SimpleCache.c`, `SimpleCache.h`, `Cache.h` and `SimpleProgram.c`.
`Cache.h` defines constants for the simulator.
`SimpleCache.h` defines the base data structures for the implementation of a 1-line directly mapped L1 cache and the overall interface through which the simulator will be tested (below).

```
void resetTime();

unsigned int getTime();

void initCache();

void read(int, unsigned char *);

void write(int, unsigned char *);
```

`SimpleCache.c` implements the logic for the functions declared in `SimpleCache.h`
`SimpleProgram.h` is a simple test case for the 1-line L1 cache provided.

4 Student Tasks

All the programs developed should be configurable via the `Cache.h` header, meaning the students cannot delete any of the constants defined in the file.
Students must deliver a header (.h) and source code (.c) file per task (described in the following subsections) on a separate directory, with number indicated for each task. On each task the students must provide a fully functional memory hierarchy to be tested. The memory hierarchy uses a write back policy and a LRU block replacement approach.

4.1 Directly-Mapped L1 Cache

In this task, you must develop a memory hierarchy with an **L1 direct mapped cache with multiple lines** with the parameters provided in the constant file.
The students must copy `SimpleCache.c`, `SimpleCache.h` to `L1Cache.c`, `L1Cache.h` (respectively), and modify the code to support multiple lines.

4.2 Directly-Mapped L2 Cache

In this task, you must develop an **direct mapped L2 cache** with the parameters specified in the constant file. In the resulting memory hierarchy of this task you must use the Directly-Mapped L1 Cache developed in the previous task (4.1).

4.3 2-Way L2 Cache

In this task, you must change the L2 cache developed in the previous task and modify it to a two way set-associate cache. Note that, the other parameters remain the same, in particular the *L2Size* value.

In the resulting memory hierarchy of this task you must use the Directly-Mapped L1 Cache developed in task (4.1).

5 Testing the Simulator

The faculty will provide some tests for you to test the resulting implementations. Nevertheless, students are strongly encouraged to develop their own tests.

6 Report and Evaluation

The students must deliver a two page report describing their implementation along with the code for the three tasks.

Students are strongly recommended to develop the requested tasks in order and complete each task in order, making sure they work properly before moving to the next task.