INSTITUTO SUPERIOR TÉCNICO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

COMPUTER ORGANIZATION

LEIC-A, LEIC-T

First Lab Assignment: Simple Cache Simulator

Version 1.1*

2024/2025

*Rev. AMC, Sep. 2024.

1 Introduction

Multilevel cache hierarchies are common in today computers with the first levels of caches integrated in the processor chip.

The goal of this assignment is the development of a memory hierarchy composed of up to two levels of caches. To achieve this, students will complete the base code provided by the faculty to implement L1 (first level) and L2 (second level) caches from scratch. Then students must integrate these components to form a complete cache 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` ...).

The simulator must be built with `make` or `make all` without any warnings. For testing purposes, it must also run on 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 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 the number indicated for each task. On each task students

must complete and provide a fully functional cache hierarchy (L1 + eventually L2) to be tested.

All caches, whenever possible, have write-back policy and a LRU block replacement.

4.1 Direct-Mapped L1 Cache

In this task, you must develop a memory hierarchy with only one level with a (L1) **direct mapped cache with several lines** with the parameters provided in the constants file.

Students must copy `SimpleCache.c` and `SimpleCache.h` to `L1Cache.c` and `L1Cache.h`, respectively, and modify the code to simulate this cache.

4.2 Direct-Mapped L2 Cache

In this task, you must develop an **direct mapped L2 cache** with the parameters specified in the constants file.

In the resulting two-level cache hierarchy you must use the Direct-Mapped L1 Cache developed in the previous task.

4.3 2-Way Set Associative L2 Cache

In this task, you must change the organization and control of the L2 cache developed in the previous task to implement a **two way set-associative L2 cache**. Note that, the other parameters remain the same, in particular the `L2Size` value.

In the resulting two-level cache hierarchy you must use the Direct-Mapped L1 Cache developed in the first task.

5 Testing the Simulator

The faculty will provide some test traces for you to test your implementations. Nevertheless, you are strongly encouraged to design and develop your own tests.

6 Report and Evaluation

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

Be prepared to demonstrate your work with test patterns that show your simulator is functionally correct, or to run test patterns provided, or suggested, by the teacher.

It is strongly recommended to develop the requested tasks in the proposed sequence and complete and test each task, making sure it works properly, before moving to the next task.