L1 Cache implementation on RISC-V

# Parallel computer architecture

## Spring 2024

### Garyfallia Anastasia Papadouli | 03533

### Iordana Gaisidou | 03570

### Dimitrios Voitsidis | 03480

### Kalliopi Nanou | 02934

Contents

[Introduction 3](#_Toc169839686)

[Policies 3](#_Toc169839687)

[Parameterization 3](#_Toc169839688)

[Write-allocate 3](#_Toc169839689)

[Write-back 3](#_Toc169839690)

[Bit pLRU 3](#_Toc169839691)

# Introduction

RISC-V, being provided under open-source licenses and being an open standard ISA in general has sparked the interest of many students in computer engineering - both undergraduate and postgraduate. The previous attendees of the Parallel computer architecture course were no exception, as they created their own implementation with a 5-stage pipeline design based on the RV32I ISA. The project was tested using official assembly tests and Verilator, a simulation based tool that converts Verilog code to C++. Our task has been to add a L1 cache memory to the provided processor.

Naturally, this has presented us with many new challenges. The initial step had to be the isolation of the memory modules out of the CPU and the creation of 2 SRAM memory modules to represent the raw cache memory, both instruction and data. As members of the team we also had to make a decision about the policies and the parameters that would determine the cache behavior.

# Project Structure

Εικόνα που περιέχει στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματαThe building blocks for the project have already been provided by our predecessors. We have added a multitude of additional files including modules and testing files to materialize our ideas.

# Policies

### Parameterization

Any size relative to the architecture of the cache has been written as modifiable, using macros in a separate header file. The additional complications arise in the form of for loops. Having parameterizable size and associativity in the context of cache testing is important as the behavior of the module can be observed under different circumstances.

### Write-allocate

Whenever something has to be stored in memory, it also gets stored in cache. The benefit of such systems is apparent any time subsequent reads to the stored value are solicitated. Since what is currently examined is the cache memory functionality, we thought its use as purposeful.

### Write-back

Write-back lessens the amount of accesses in the main memory, which is something we consider desired. Write-back policy mandates that any modification of data stored in cache gets written in memory only when that cache block has to be evicted.

### 

### Bit pLRU

We decided upon pLRU (pseudo LRU) as a feausible alternative to LRU as the original LRU policy is costly to hardware.

The number of bits used by LRU:

In striking contrast with the pseudo LRU that uses only one. The way bit LRU functions is by setting a status bit for every memory reference, and reverting all but the latest status bits whenever the last zero gets flipped.

## Modules

Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, παράλληλα

Περιγραφή που δημιουργήθηκε αυτόματα

Starting from the highest level of the project, the toplevel module