pr5 - CPU Caches

1 Lab Assignment 6

In this lab, you will introduce a memory hierarchy into the pipelined RISC-V processor implemented so far. Follow the instructions given below to achieve the same.

Instructions

1. Extend the simulator to accept the simulation parameters and processor configurations through a config file. The path to the config file should be passed into the simulator using a command-line argument `--config`. If no argument is specified, then the default configuration file is `config.ini`. The simulation parameters and processor configurations include items such as the start PC of the simulation, the Class-name of the simulated processor (single cycle, or pipelined processor), the size of the caches, levels of caches are present, and the replacement policy of each cache. A sample `config.ini` is given in the `lab6` branch.

2. Integrate the given `config_reader.py` into the simulator. The arguments given in the command-line should supercede the arguments in the config file, in case of conflicts. Modify `simulate.py` to achieve this.

3. Create a new folder named `memory/` (just like `core/`). Move the `ram.py` into the newly created `memory/` folder. Modify your existing code to reflect this change in folder structure.

4. Create a new Class called `Cache`. This should take in the cache parameters given in the config file as input and instantiate a cache as specified. If the config file sets the parameter `valid` to `False`, then that cache should not be instantiated. The processor simulation should behave as if that particular cache is completely absent from the system.

5. Modify the `Processor` class to incorporate a memory hierarchy. The requests to instruction memory should go to `I1_Cache`, and the requests to the data memory should go to the `L1_Cache`. If the requested data is not found in any of these caches, the request is then sent to the `L2_Cache`. Note that there is no `valid` parameter associated with the `RAM`, and hence if all the caches are absent, then the data comes from `RAM` with the latency (in clock cycles) associated with the RAM.

6. Update the clock cycles appropriately (per memory access), and write it to statistics file as earlier.