pr5 - Pipelined Processor Simulator

1 Lab Assignment 4

In this lab, you will simulate a pipelined RISC-V processor and count the number of clock cycles a simple 5-stage pipeline takes to execute a code. Follow the instructions given below to achieve the same.

Instructions

1. Merge the code from the `'lab4'` branch of your respective git repositories into the `'main'` branch **if required**. It is recommended to use the code given in the `'lab4'` branch, only if you could not solve the previous lab yourselves. Note that the code given in the `'lab4'` branch may be different from your code. That is okay, and you need not worry about it.

2. Implement a new command-line switch `proc`. This command-line switch can take two different values: `SingleCycleProcessor` or `PipelinedProcessor`. You will have to instantiate the appropriate processor depending on the command-line switch passed. See the `TODO` tags in `simulate.py`.

3. Modify the member functions such as `fetch()`, `decode()`, `execute()`, etc. of the base class `Processor` to make them stateless (these functions should not set any member variables as far as possible). These functions should take the necessary inputs as arguments and explicitly return the computed values. This will ensure that these member functions can be re-used across several derived classes while the code remains readable.

4. Modify `SingleCycleProcessor` to honor the changes in the interfaces made above.

5. Modify `SingleCycleProcessor` to increment the clock cycles. This will ensure that `stats.json` will record the number of clock cycles taken to execute the given number of instructions.

6. Modify the run() function of `PipelinedProcessor`. You should implement the `simple` 5-stage pipeline discussed in the lectures (detect data and control hazards and stall). Assume that there would be three wrong-path instructions in the case of a control hazard. Note that you should increment the clock cycles of the `PipelinedProcessor` also appropriately.

7. Push your changes to the `'main'` branch at regular intervals.