CS2323: Extending the RISC-V Simulator to Support Pipelining and Hazard Handling

Devansh Tripathi - CS24BTECH11022

Contents

0.1 Objective

To enhance the existing single-cycle RISC-V simulator by implementing an instruction pipeline (IF, ID, EX, MEM, WB) and supporting data and control hazard handling mechanisms.

The goal is to demonstrate the impact of pipelining on performance and compare various pipeline configurations.

0.2 Motivation

The current in-house simulator executes one instruction per cycle (single-cycle), which leads to higher cyle counts. This makes it inefficient and does not reflect the behaviour of current processors.

Pipelining is an architectural optimization that improves the cycle count by overlapping multiple instructions.

This project aims to modify the current simulator by introducing configurable pipelining functionality to it, which can demonstrate:

- Pipeline performance improvements.
- Different types of hazards and their effects.
- The impact of hazard mitigation techniques like forwarding and branch prediction.

0.3 Proposed Enhancements

The enhanced simulator will support the following configurable execution modes:

Mode	Description
0	Single-cycle
1	Pipelined with no hazard handling
2	Pipelined with hazard detection (stalling only)
3	Pipelined with hazard detection and data forwarding
4	Pipelined with hazard + static branch prediction
5	Pipelined with hazard + dynamic 1-bit branch prediction

These modes can be toggled with a configuration file or a command line flag.

0.4 Functional Scope

1. **Pipeline Stage Simulation**
    - Implement the five stages (IF, ID, EX, MEM, WB)

- Introduce pipeline registers between stages (IF/ID, ID/EX, EX/MEM, MEM/WB)

2. **Hazard Handling**
   - Detect data hazards, implement stall insertion, data forwarding logic.
   - Detect branch instructions and manage flushing and stalling, add static and dynamic branch prediction.
3. **Performance Evaluation**
   - Log cycle counts, number of stalls, correctness.
   - Compare across configurations.

0.5 Testing and Validation

- **Benchmark Programs**
  - Simple arithmetic
  - Data hazard prone programs
  - Branch-heavy programs
  - Memory intensive programs
- Each of the above will be run across all pipeline modes.
- Results will be logged as:
  - Execution cycles
  - Number of stalls
  - Correctness

0.6 Deliverables

- **Code**: Enhanced simulator source code with pipelining and hazard handling logic.
- **Documentation**: Design report with benchmark program results and comparison.
- **Benchmark programs**: Test programs.

0.7 Team Members

- Devansh Tripathi - CS24BTECH11022