Pipelined Processor Report

17th March, 2024

1. Introduction

This report documents the development of a pipelined processor simulator for ToyRISC ISA. The simulator is implemented in Java and adheres to the specifications provided in the description. The primary goal of this part of the project is to simulate the execution of ToyRISC programs and generate statistics regarding the number of instructions executed, cycles taken and number of stalled instruction.

2. Implementation Details

2.1 Simulation of the program

The simulation of the program involves five stages of the processor, which are pipelined. Between each pair of stages, there is a latch, and each latch type is described by a separate class. The implemented stages include:

1. IF (Instruction Fetch) Stage: This stage has been implemented and is responsible for fetching instructions from memory. The instructions are then passed to the next stage through the IF-OF latch.
2. OF (Operand Fetch) Stage: This stage is responsible for decoding the instructions and preparing them for execution. The contents of the ID latch are determined by the specific instruction being processed.
3. EX (Execution) Stage: In this stage, arithmetic and logic operations are performed based on the decoded instructions. The contents of the EX latch depend on the operation being executed.
4. MA (Memory Access) Stage: This stage handles memory read and write operations, including load and store instructions. The contents of the MEM latch are determined by the memory access

The simulation continues until an “end” instruction passes through the WB stage, at which point the simulation is marked as complete using setSimulationComplete().

2.2 Interlocks

The pipelined ToyRISC processor is an in-order pipeline implying the presence of Read after Write(RAW) Hazard and Branch Hazard which will be resolved by their respective Interlocks.

2.2.1 Data Interlock

RAW hazard is encountered when the source registers for the next coming instruction conflict with the destination register of the currently processing instruction. In the case of RAW hazard, the pipeline is stalled by passing two nop instructions. In a ToyRISC processor, RAW can be encountered for all instructions that use source registers, i.e. for all instructions except jmp instruction.

2.2.2 Branch Interlock

Branch hazard is encountered when the ”branch” instructions result in the ”isBranchTaken” to be true i.e. the instructions already in the IF and OF stages have to be nullified and new instructions are to be fetched. Branch hazard is possible in any of the conditional branch instructions i.e. **beq, blt, bgt, bne** and will definitely exist in **jmp** instruction.

3. Test Case Performance

The following table summarises the performance of the simulator for

each test case:

Test Case	Instructions	Executed Instructions	No. of nop	No. of wrong
Fibonacci	21	153	42	16
Descending	21	658	211	88
Even or Odd	9	13	6	0
Prime	16	64	25	5
Palindrome	16	107	43	7

4. Conclusion

In conclusion, the ToyRISC pipelined processor simulator has been successfully implemented. The simulator can load programs, execute instructions through the five processor stages, and record relevant statistics.