Project Guidelines

EE 371L/CS 330L/CE 321L Computer Architecture Lab

Maximum Members per Group	3
Weightage	28 % of the lab grade
Submission Deadline	Due in the last week of classes
Evaluation	Viva-based

Objective:

To **build a 5-stage pipelined processor capable of executing any one array sorting algorithm.**

Description:

Basically, you will be converting your single cycle processor to a pipelined one. Normally the instructions you have already implemented should enable you to execute a sorting algorithm program with small additions i.e., you might need to implement the **bgt or blt** instruction, or something similar, so that you know when you'd need to swap two values. This would require small modifications to the circuit.

Here's the recommended course of action:

Part 1. Choose a suitable sorting algorithm. Convert its pseudocode to assembly using RISC-V instruction set and test its working on the assembly code simulator. Now, modify the existing lab 11 single-cycle processor architecture to run the sorting algorithm code on it. Test and verify that it is doing the sorting correctly.

Part 2. Modify this processor to make it a pipelined one (5 stages). Test and run each instruction separately to verify that the pipelined processor version can at least execute one instruction correctly in isolation. Test different types of instructions. Test cases matter!

Part 3. Introduce circuitry to detect hazards (data, control, and structural if needed) and try to handle them in hardware i.e., by forwarding, stalling and flushing the pipeline. If this is done correctly, then your array sorting code should be able to function in its entirety

Part 4. Compare the performance of running the array sorting program on Single Cycle Processor with RISC-V Pipelined Processor in terms of execution time.

The book Chapter 4 till end of section 4.8, covered in the theory classes, can be referred for implementing the above step by step.

Submission and Evaluation:

1. You will submit all your codes (the Verilog files for modules as well as test benches) task wise
2. You will submit a PDF report containing explanations of how you implemented your processor, test cases and results, performance comparison (part 4), any difficulties you had and how you

overcame them, and any deficiencies in your projects if there are any. Also, add all the codes in appendices.

3. Any marks are conditional on your being able to explain each line of your code to the examiner during a viva. Each member of the group will be evaluated through the viva.

Part-wise Weightage:

Part		Weightage
1	Sorting Algorithm on Single Cycle Processor	25 %
2	Pipelined Processor	30 %
3	Hazard Detection & Sorting Algorithm on Pipelined Processor	30 %
4	Performance comparison	5%
5	Viva	10%
	Total	100 %

Implementation Help for the Pipelined Processor and its Forwarding Unit

We have studied pipeline implementation of a RISC-V processor with data forwarding techniques to overcome data hazards. Implement the pipeline version of RISC-V processor shown in Figure 1. Initialize all the pipeline registers to an appropriate size. The control values for the forwarding multiplexers are shown in Table 1. **For each of the pipeline registers, you can create a separate module.**

Table 1: The control values for forwarding multiplexers

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

Refer to the lecture slides in order to understand the behavior of forwarding unit. Verify the functionality of forwarding by introducing data dependencies in R-format instructions. Do not check the dependency of a load instruction result on the next instruction, as the architecture shown in Figure 1 does not support stalling to overcome certain type of data hazard.

For Task 3, you will be modifying this to include stalling and pipeline flushing. After these modifications, you will be able to test the sorting algorithm on pipelined processor.

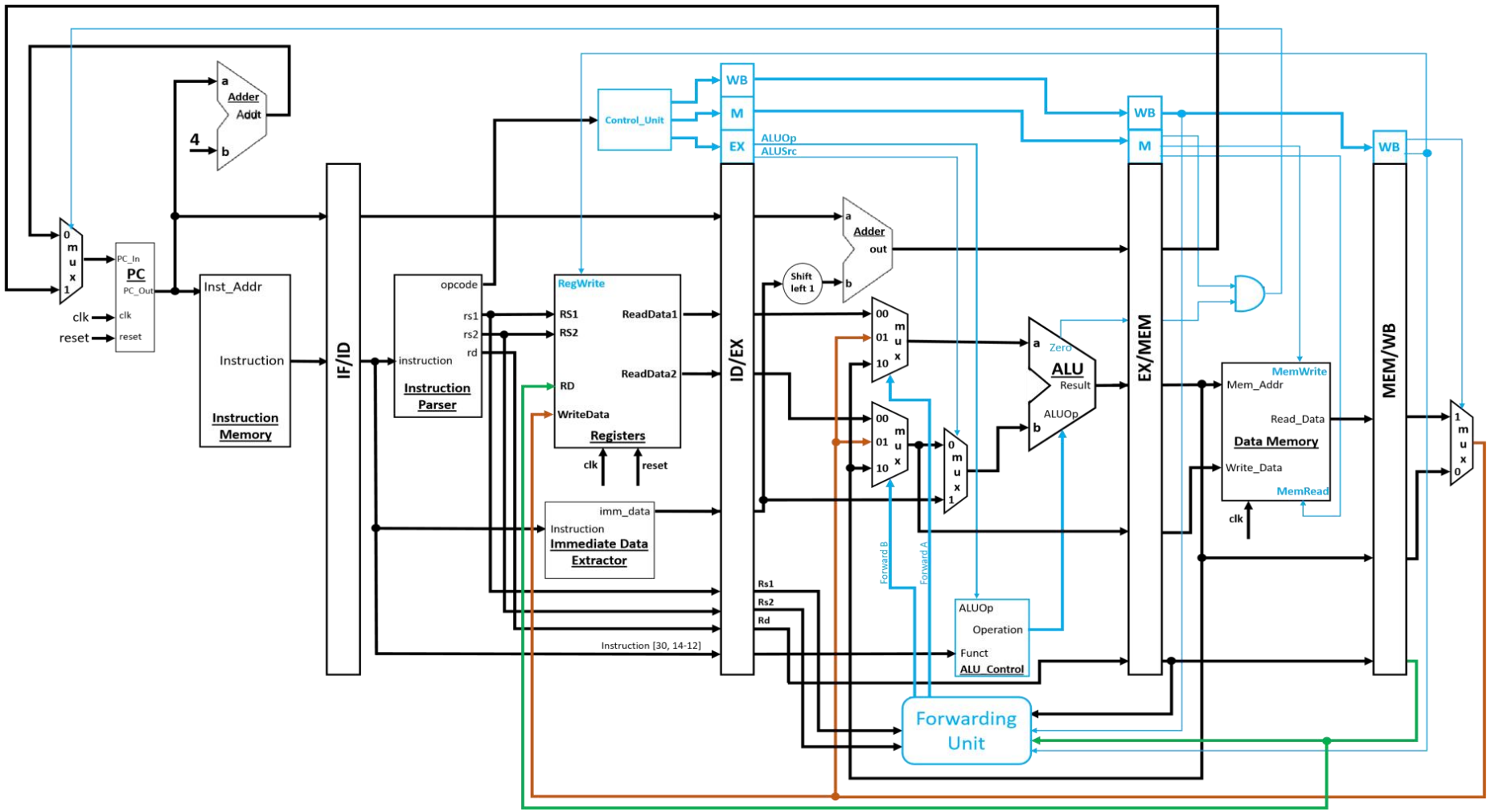

Figure 1: A subset of Pipeline RISC-V processor with forwarding functionality only. Some wires are shown in color just to help a user to distinguish wires easily (specifically between IF/ID and ID/EX stages).