Habib University

Dhanani School of Science and Engineering

Computer Architecture

(CS-330 / EE-471)

Lab Manual

Table of Contents

About the lab manual

This lab manual has been created with the help of practical experiments, several supporting documents and presentations listed in the Bibliography section.

The creation process of this manual is started during the summer 2018 by Dr. Hasan Baig, and this manual is continuously being updated.

For questions, comments, or suggestions, please contact Dr. Hasan Baig at the following email address: hasan.baig@sse.habib.edu.pk.

About the lab exercises

These laboratory exercises have been designed to get the students acquainted with the hardware design skills. You will learn how to design hardware using hardware description language (HDL); how to simulate your design; and how to test it on a reconfigurable chip. Once you get familiar with the design flow, you will be required to develop processor peripherals in the following labs. A brief summary of all the lab exercises are given below.

In **Lab 1**, you will be introduced to the programmable logic and the Verilog HDL. Furthermore, you will learn how to design a simple hardware and verify its functional behavior using a professional simulation tool, named *ModelSim®*.

In **Lab 2a**, a hardware synthesis flow is discussed targeting the Xilinx FPGA technology. Furthermore, you will get to run your designed hardware on actual FPGA chip.

In **Lab 2b**, you will learn how to integrate the ready-made module (UART) with your custom design. Also, in this session, you will use desktop-based software, designed specifically for this course, to observe the output of on-chip hardware.

In **Lab 3**, you will be developing some intermediate modules of a processor which will be required in next labs. In particular, you will develop a multiplexer, an instruction parser, and immediate field extractor.

In **Lab 4**, In this lab, you will develop a Register File for a processor, and will simulate its behavior in ModelSim.

In **Lab 5**, you will develop a RISC arithmetic and logic unit and verify its functionality using simulation.

In **Lab 6**, you will develop modules for instruction and data memories.

In **Lab 7**, you will develop a datapath for instruction fetch cycle, followed by its verification.

In **Lab 8**, you will design a control unit of RISC processor.

In **Lab 9**, you will develop a single cycle RISC processor by integrating previously designed modules.

Conventions

The following conventions appear in this lab manual.

📖	This icon denotes a “pre-lab exercise”, which a student should complete before coming into the respective lab.
🖥	This icon denotes a “lab exercise”, which a student should complete during the lab hours.
✎	This icon denotes a “post-lab exercise”, which a student should complete outside the lab hours.
⏱	This icon indicates the expected time (in minutes) to complete the specific exercise.
💡	This icon denotes a tip, which notifies you to advisory information.
⚠	This icon denotes an alert, which notifies you to important information.
`Bold` `or` *`Italic`*	The text written in this font is used specifically for the syntax of HDL.
bold	Bold text denotes items that you must select or click or enter the value in the software, such as open file option or running the simulation button or entering the command in the transcript window. The bold text is also used to refer to the specific options in the software tools.
italic	Italic text denotes the name of a folder or a file path.
bold and italic	Bold and italic text denotes the name of a file.

Lab 9 – Design of a Single Cycle RISC Processor

Objectives

In this lab, we will integrate previously designed modules to build a single-cycle RISC-V processor.

Section	⏱
a) Introduction A brief overview of this lab.	**01**
b) Implementation In this section, you will implement a single-cycle RISC processor by integrating previously designed modules.	**60**
Exercise An exercise to verify the functionality of a single-cycle processor including some theoretical explanation.	**30**

a. Introduction

We have developed all the necessary modules of a single cycle processor. We can now combine all the pieces to make a simple datapath for the core RISC-V architecture to run specific set of instructions, which include R-Type, I-Type and Branch-Type instructions.

b. Implementation

In this lab, we will use the modules developed in the previous labs and integrate them together to construct a single-cycle datapath processor. Copy the following modules in a new file pate, named *Lab09/design*, from the locations specified below:

1. ***ImmGen.v***, ***Mux.v***, ***InsParser.v*** from *../Lab03/design* folder.
2. ***RegisterFile.v*** from *../Lab04/design* folder.
3. ***ALU_64_bit.v*** from *../Lab05/design* folder.
4. ***Data_Memory.v*** and ***Instruction_Memory.v*** from *../Lab06/design* folder.
5. ***Program_Counter.v*** and ***Adder.v*** from *../Lab07/design* folder.
6. ***ALU_Control.v*** and ***Control_Unit.v*** from *../Lab08/design* folder.

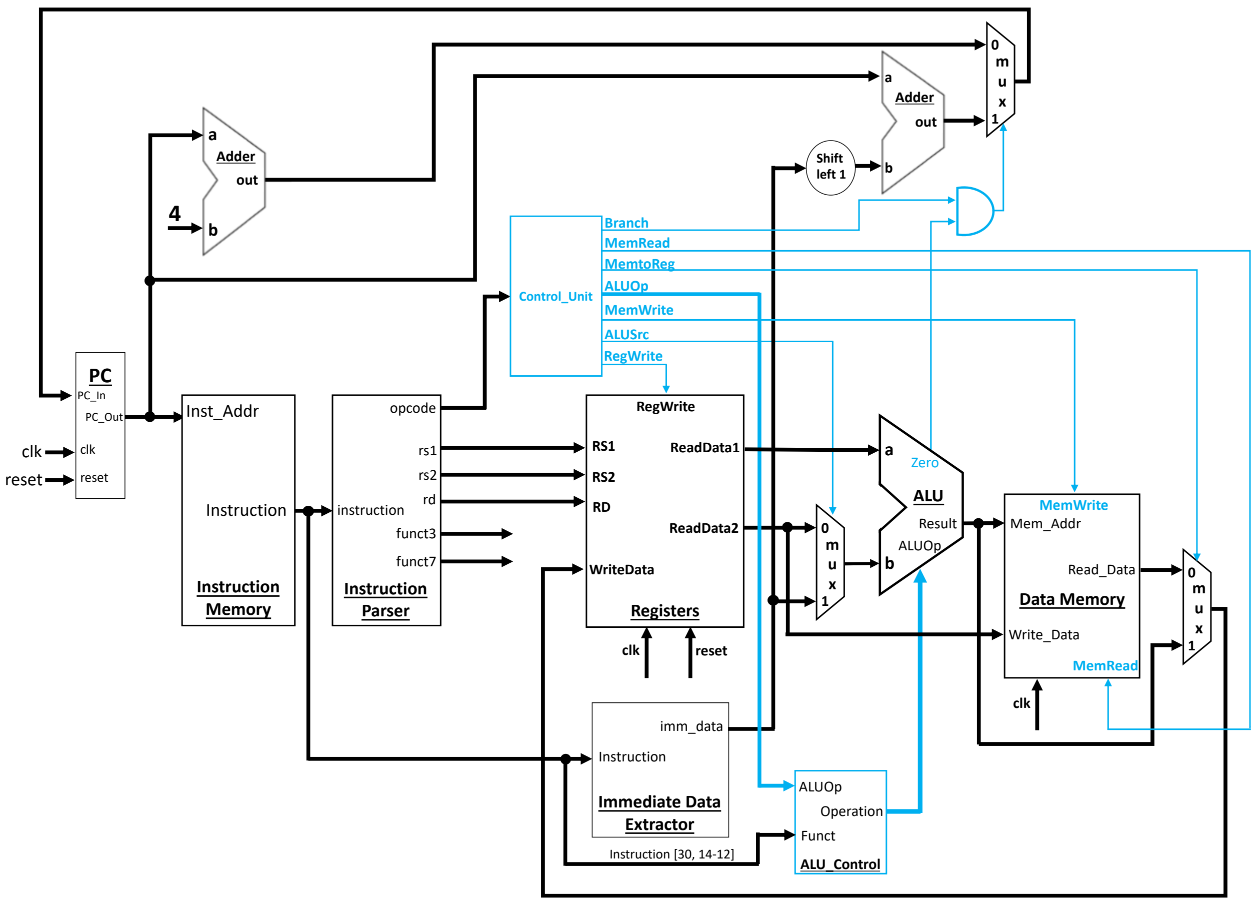

Fig. 9.1. Single cycle processor.

Now integrate all the above modules in a top module named, `RISC_V_Processor`, according to the processor diagram shown in Figure 9.1. The inputs to the top-level module are `clk` and `reset`, and there is no output.

Exercise

Load instruction memory with the contents shown in Figure 9.2 below.

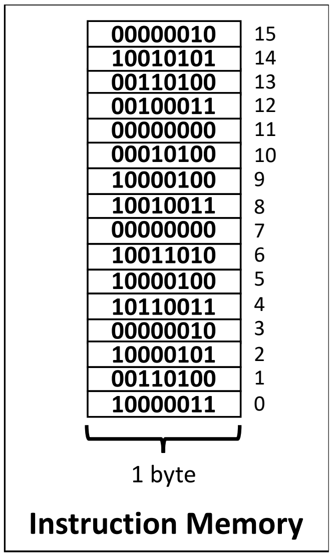

Fig. 9.2. Instruction memory contents for exercise.

Make sure that you already have appropriate values initialized in the Register File and Data Memory. Also make sure that there are 32 registers in the Register File. Perform the following tasks.

1. Write a testbench. Initialize the simulation by first resetting the module under test for, say, 10ns.

2. Decode the instructions placed in instruction memory, and mention their assembly code below.

3. Write down its corresponding C code below. Assume that the relevant register used in this exercise is given a variable name "h" and the memory array is given a name "A".