






Lab 11 – 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) <u>Introduction</u>  A brief overview of this lab.	01
b) <u>Implementation</u>  In this section, you will implement a single-cycle RISC processor by integrating previously designed modules.	60
<u>Exercise</u>   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 path, named Lab11/design:

1. ImmGen.v, Mux.v, InsParser.v from ../Lab06 folder.
2. RegisterFile.v from ../Lab07 folder.
3. ALU_64_bit.v from ../Lab05 folder.
4. Data_Memory.v and Instruction_Memory.v from ../Lab08 folder.
5. Program_Counter.v and Adder.v from ../Lab09 folder.
6. ALU_Control.v and Control_Unit.v from ../Lab10 folder.

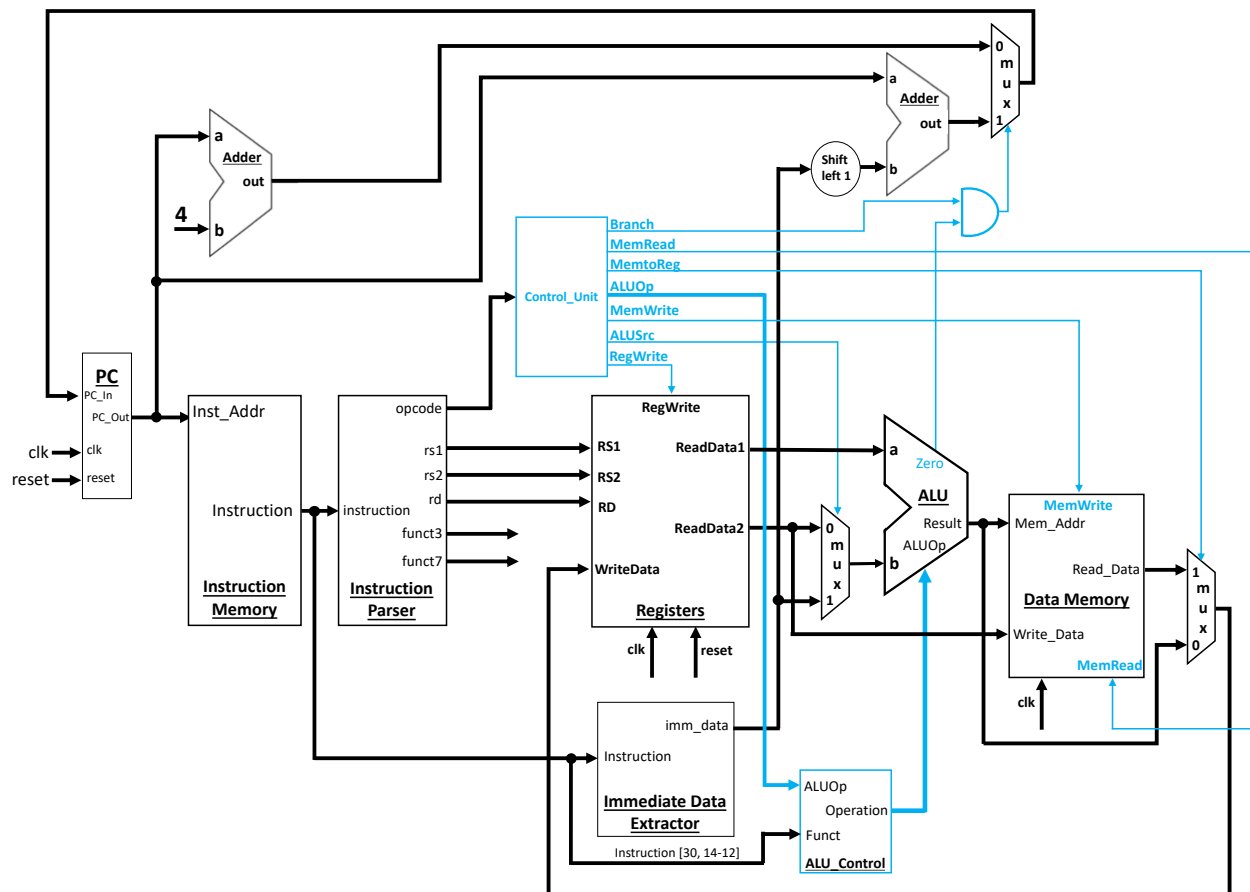


Fig. 11.1. Single cycle processor.



Now integrate all the above modules in a top module named, `RISC_V_Processor`, according to the processor diagram shown in Figure 11.1. 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 11.2 below.

00000010	15
10010101	14
00110100	13
00100011	12
00000000	11
00010100	10
10000100	9
10010011	8
00000000	7
10011010	6
10000100	5
10110011	4
00000010	3
10000101	2
00110100	1
10000011	0

1 byte

Instruction Memory

Fig. 11.2. 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. Toggle the clock signal at every 5ns.
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”.

Computer Architecture

Spring 20

Lab 11

Marks Distribution:

Task	LR2: Design/Code	LR5: Results
Task 1 (Implementation)	30 Points	-
Task 2 Exercise 1 (Test bench & Results)	10 Points	20 Points
Task 3 Exercise 2, 3	20 Points	-
Total		80 Points

Marks Obtained:

Task	LR2: Design/Code	LR5: Results
Task 1 (Implementation)		-
Task 2 Exercise 1 (Test bench & Results)		
Task 3 Exercise 2, 3		-
Total		

