Lab 10 – Designing a Control Unit of a RISC Processor

Objectives

In this lab, we will develop an instruction-fetch datapath and verify its functionality.

Section	
a) Introduction A brief overview of this lab.	**10**
b) Implementation In this section, you will implement a control unit of a RISC-V processor.	**90**

a. Introduction

The last module we are left with is the control unit, which we have to design before we proceed further to integrate the already-designed processor components. In this lab, we will develop a module for generating control signals for specific instructions. These control signals are used to control the data flow and enabling or disabling the modules which are not needed for specific instructions.

b. Implementation

Besides control unit, we also have to develop ALU Control unit to set the control signals of ALU.

i. Lab Task 01

As shown in Fig. 10.1, Write a module, named`Control_Unit`, which takes a 7-bit wide input, named `Opcode`, and generate 7 output signals. Out of these seven outputs, one is `ALUOp` which is 2-bits wide, and the remaining six are 1-bit wide, which are `Branch`, `MemRead`, `MemtoReg`, `MemWrite`, `ALUSrc`, and `RegWrite`.

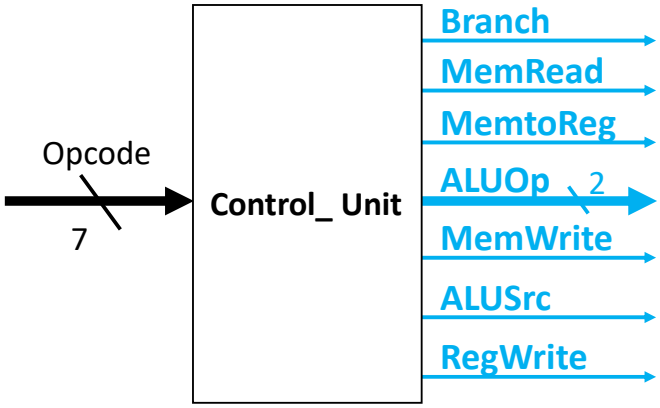

Fig. 10.1. I/O diagram of Control Unit.

The behavior of Control_Unit module should be designed according to the Table 10.1. The outputs depend only on the input, `Opcode`. Here, 1'bx means don't care.

Table. 10.1. This table shows how the control signals are set based on the input values of Opcode.

Instruction Type	Opcode	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp [1:0]
R-Type	0110011	0	0	1	0	0	0	10
I-Type (ld)	0000011	1	1	1	1	0	0	00
I-Type (sd)	0100011	1	X	0	0	1	0	00
SB-Type (Beq)	1100011	0	X	0	0	0	1	01

Interpret these control signals, relate these with the given instructions and identify their impact on data-flow.
Now, add another case for addi instruction by including its opcode and corresponding output signals.
You can initialize all these control signals with 0 or add a default case for this in module.

ii. Lab Task 02

Write a module, named `ALU_Control`, which takes a 2-bit input, named `ALUOp` and a 4-bit input, named `Funct`, and produces a 4-bit output, named `Operation`, as shown in Figure. 10.2.

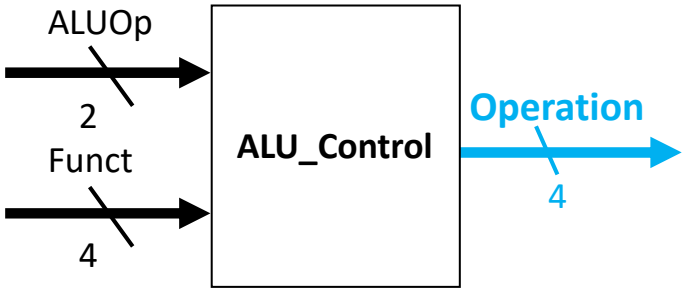

Fig. 10.2. I/O diagram of ALU_Control module.

The values of the output, `Operation`, should be set based on the input signals, `ALUOp`, as shown in Table 10.2.

Table. 10.2. The values of the output, Operation, based on the corresponding input, ALUOp, signals.

Instruction Type	ALUOp [1:0]	Funct	Operation
I/S-Type (ld, sd)	00	xxxx	0010
SB-Type (Beq)	01	xxxx	0110
R-Type	10	0000	0010
		1000	0110
		0111	0000
		0110	0001

If ALUOp is 2'b00 or 2b'01, then Operation can be decided directly without checking bits of Funct.
If ALUOp is 2'b10, then Operation is decided according to the value of Funct bits.

iii. Lab Task 03

Integrate above two modules in a top module, named `top_control`. Write a testbench and perform verification.

The top module has 2 inputs; Opcode, and Funct and outputs 7 outputs; Branch, MemRead, MemtoReg, MemWrite ALUSrc, RegWrite, and Operation.

Assessment Rubric
Computer Architecture Lab
Lab 10
Designing a Control Unit for RISC V Processor

Name:	**Student ID:**

Points Distribution

Task No.	**LR 2** **Code**	**LR 5** **Results**	**AR 7** **Report Submission**
Task 1 **Control Unit**	/20	-	/20
Task 2 **ALU Control**	/20	-	
Task 3 **Top Control**	/20	/20	
Total Points	/100 Points		
CLO Mapped	CLO 1		

For description of different levels of the mapped rubrics, please refer to the provided Lab Evaluation Assessment Rubrics.

#	Assessment Elements	Level 1: Unsatisfactory Points 0-1	Level 2: Developing Points 2	Level 3: Good Points 3	Level 4: Exemplary Points 4
LR2	**Program/Code/ Simulation Model/ Network Model**	Program/code/simulation model/network model does not implement the required functionality and has several errors. The student is not able to utilize even the basic tools of the software.	Program/code/simulation model/network model has some errors and does not produce completely accurate results. Student has limited command on the basic tools of the software.	Program/code/simulation model/network model gives correct output but not efficiently implemented or implemented by computationally complex routine.	Program/code/simulation /network model is efficiently implemented and gives correct output. Student has full command on the basic tools of the software.
LR5	**Results & Plots**	Figures/ graphs / tables are not developed or are poorly constructed with erroneous results. Titles, captions, units are not mentioned. Data is presented in an obscure manner.	Figures, graphs and tables are drawn but contain errors. Titles, captions, units are not accurate. Data presentation is not too clear.	All figures, graphs, tables are correctly drawn but contain minor errors or some of the details are missing.	Figures / graphs / tables are correctly drawn and appropriate titles/captions and proper units are mentioned. Data presentation is systematic.
AR7	**Report Content/Code Comments**	Most of the questions are not answered / figures are not labelled/ titles are not mentioned / units are not mentioned. No comments are present in the code.	Some of the questions are answered, figures are labelled, titles are mentioned and units are mentioned. Few comments are stated in the code.	Majority of the questions are answered, figures are labelled, titles are mentioned and units are mentioned. Comments are stated in the code.	All the questions are answered, figures are labelled, titles are mentioned and units are properly mentioned. Proper comments are stated in the code.