LAB # 4
Designing Control Unit in Verilog

Objective:

- To Understand Instruction Set for RISC-V architecture.
- To design the Control Unit (Main Decoder/ALU Decoder) in Verilog for RISC-V architecture.

System Module (Hardware/Software)

- Visual Studio Code

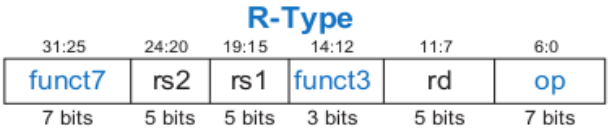
Theoretical Background:

Introduction

RISC-V uses 32-bit instructions. RISC-V consists of defining the following instruction formats: R-type, I-type, S-Type, B-Type, U-type and J-type. R-type instructions operate on three registers. I-type, S-type and B-type instructions operate on two registers and a 12-bit immediate. U-type and J-type (jump) instructions operate on one 20-bit immediate.

R-Type Instructions

The name R-type is short for register-type. R-type instructions use three registers as operands: two as sources, and one as a destination. Figure below shows the R-type machine instruction format. The 32-bit instruction has six fields: op, rs1, rs2, rd, funct3, and funct7. Each field is five or seven bits, as indicated. The operation the instruction performs is encoded in the three fields highlighted in blue: **op** (also called opcode or operation code) and **funct3** and **funct7** (also called the function). All R-type instructions have an opcode of **33**. The specific R-type operation is determined by the function fields. The operands are encoded in the three fields: **rs1**, **rs2**, and **rd**. The first two registers, rs1 and rs2, are the source registers; rd is the destination register.

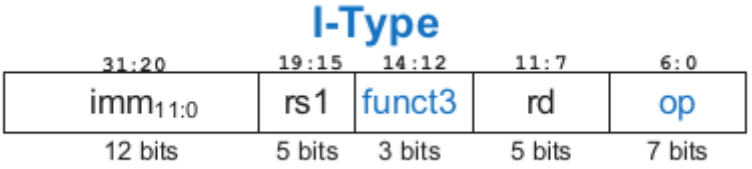


Instruction Format for some of the R-type instructions in shown below:

0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

I-Type Instructions

The name I-type is short for immediate-type. I-type instructions use two register operands and one immediate operand. Figure below shows the I-type machine instruction format. The 32-bit instruction has five fields: op, rs1, rd, funct3 and imm. The first three fields, op, rs1, and rd, are like those of R-type instructions. The imm field holds the 12-bit immediate. The funct3 holds the operation to be performed. The operation is determined by the opcode and funct3, highlighted in blue. The operands are specified in the two fields rs1, and imm. rs1 and imm are always used as source operands. rd is used as a destination.



Instruction format for some of the I-type instructions are shown below:

imm[11:0]	rs1	000	rd	0000011	LB
imm[11:0]	rs1	001	rd	0000011	LH
imm[11:0]	rs1	010	rd	0000011	LW
imm[11:0]	rs1	100	rd	0000011	LBU
imm[11:0]	rs1	101	rd	0000011	LHU
imm[11:0]	rs1	000	rd	0010011	ADDI
imm[11:0]	rs1	010	rd	0010011	SLTI
imm[11:0]	rs1	011	rd	0010011	SLTIU
imm[11:0]	rs1	100	rd	0010011	XORI
imm[11:0]	rs1	110	rd	0010011	ORI
imm[11:0]	rs1	111	rd	0010011	ANDI
0000000 shamt	rs1	001	rd	0010011	SLLI
0000000 shamt	rs1	101	rd	0010011	SRLI
0100000 shamt	rs1	101	rd	0010011	SRAI

S-Type Instructions

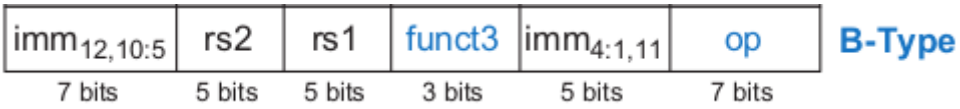
The name S-type is short for store-type. S-type instructions use two register operands and one immediate operand. Figure below shows the S-type machine instruction format. The 32-bit instruction has five fields: op, rs1, rs2, funct3 and imm. The first three fields op, rs1, and rs2 are like those of R-type instructions. The imm fields hold the 12-bit immediate. The 12-bit immediate is split into two sets as shown below. The operation is determined by the opcode and funct3, highlighted in blue. The operands are specified in the three fields rs1, rs2 and imm.

Instruction format for some of the S-type instructions are shown below:

imm[11:5]	rs2	rs1	000	imm[4:0]	0100011 SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011 SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011 SW

B-Type Instructions

The name B-type is short for branch-type. This format is used only with branch instructions. B-type instructions use two register operands and one immediate operand. Figure below shows the B-type machine instruction format. The 32-bit instruction has five fields: op, rs1, rs2, funct3 and imm. The first three fields op, rs1, and rs2 are like those of R-type instructions. The imm fields hold the 12-bit immediate. The 12-bit immediate is split into two sets as shown below. The operation is determined by the opcode and funct3, highlighted in blue. The operands are specified in the three fields rs1, rs2 and imm.



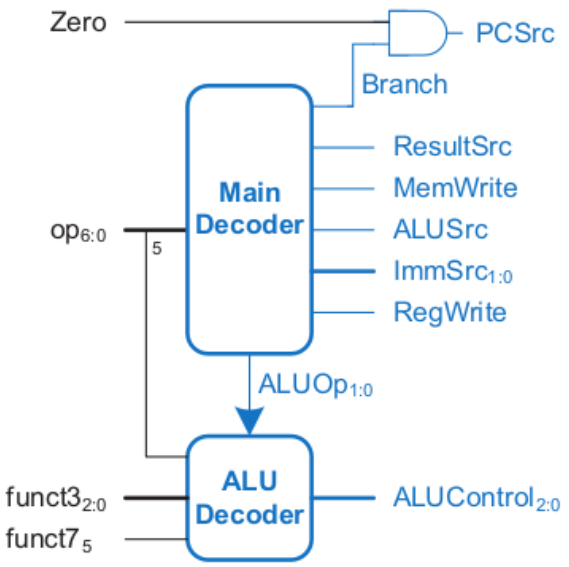
Instruction format for some of the B-type instructions are shown below:

imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU

To interpret machine language, one must decipher the fields of each 32-bit instruction word. Different instructions use different formats, but all formats start with a 7-bit opcode field. Thus, the best place to begin is to look at the opcode.

Control Unit

The control unit computes the control signals based on the opcode and funct fields of the instruction, Instr[31:25], Instr[14:12] and Instr[6:0]. Most of the control information comes from the opcode, but for further operations function fields are used. Thus, we will simplify our design by factoring the control unit into two blocks of combinational logic, as shown in Figure below



MAIN Decoder:

Table below is a truth table for the main decoder that summarizes the control signals as a function of the opcode. All R-type instructions use the same main decoder values; they differ only in the ALU decoder output. Recall that, for instructions that do not write to the register file (e.g., S-type and B-type), the ResultSrc control signals is don't care (X); the address and data to the register write port do not matter because RegWrite is not asserted. The logic for the decoder can be designed using your favorite techniques for combinational logic design.

Instruction	Op	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp
lw	0000011	1	00	1	0	1	0	00
sw	0100011	0	01	1	1	x	0	00
R-type	0110011	1	xx	0	0	0	0	10
beq	1100011	0	10	0	0	x	1	01

ALU Decoder:

The main decoder computes most of the outputs from the opcode. It also determines a 2-bit ALUOp signal. The ALU decoder uses this ALUOp signal in conjunction with the funct field and opcode bit to compute ALUControl. The meaning of the ALUOp signal is given in Table below

ALUOp	Meaning
00	add
01	subtract
10	look at funct fields and opcode bit
11	N/A

Table below is a truth table for the ALU decoder. The logic of ALUControl was covered in lab #2. When ALUOp is 00 or 01, the ALU should add or subtract, respectively. When ALUOp is 10, the decoder examines the function fields and operand bit to determine the ALUControl. The control signals for each instruction were described as we built the datapath.

ALUOp	funct3	$\{op_5, funct7_5\}$	ALUControl	Instruction
00	x	x	000 (add)	`lw, sw`
01	x	x	001 (subtract)	`beq`
10	000	00, 01, 10	000 (add)	`add`
	000	11	001 (subtract)	`sub`
	010	x	101 (set less than)	`slt`
	110	x	011 (or)	`or`
	111	x	010 (and)	`and`

Procedure:

Open Visual Studio Code and perform the procedure mentioned in lab #01 in order to make a Verilog module and test bench.

Lab Tasks:

1. **Write a Verilog Code for the RISC-V Control Unit by using the provided truth tables. Make two different modules for the Main decoder and ALU decoder. Use appropriate inputs and outputs for the modules. By making a test bench for each module check whether your design is correct or not. Attach the codes, test benches and the waveforms.**

Conclusion:

What have you learnt from this lab?

Learning Outcomes:

Upon successful completion of the lab, students will be able to:

LO1: Differentiate between R, I, S and B type instructions.
LO2: Design the control unit for RISC-V.