Computer Organization – Project – Phase II

Email: iustCompOrg+4012@gmail.com

In this phase of the project you are going to design the control unit of your CPU and branch related circuits (stage 2). `Branch_Unit.v` and `Control_Unit.v` are included in `RISCV_CPU.zip`.

1) Design the control unit of your RISC-V core based on microprogramming approach. Decode and extract different fields of IR fetched by the previous modules you designed.

31	0
IR	

a. Determine instruction type → R / I / S / B / U / J.

IR[4:2] / IR[6:5]	000	001	010	011	100	101	110	111
00	I	I	-	-	I	U	I	-
01	S	S	-	R	R	U	R	-
10	-	-	-	-	R	-	-	-
11	B	I	-	J	-	-	-	-

Using the table above create signals indicating the type of the instruction.

b. Extract the immediate value contained in IR.

31	30 25	24 21	20	19 15	14 12	11 8	7	6 0	
funct7		rs2		rs1	funct3	rd		opcode	R-type
imm[11:0]				rs1	funct3	rd		opcode	I-type
imm[11:5]		rs2		rs1	funct3	imm[4:0]		opcode	S-type
imm[12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm[11]	opcode	B-type
imm[31:12]						rd		opcode	U-type
imm[20]	imm[10:1]		imm[11]	imm[19:12]		rd		opcode	J-type

RISC-V base instruction formats showing immediate variants.

Each of the base instruction formats produce a different immediate value.

31	30 20	19 12	11	10 5	4 1	0	
-IR[31]-				IR[30:25]	IR[24:21]	IR[20]	I-immediate
-IR[31]-				IR[30:25]	IR[11:8]	IR[7]	S-immediate
IR[31]			IR[7]	IR[30:25]	IR[11:8]	0	B-immediate
IR[31]	IR[30:20]	IR[19:12]	-0-				U-immediate
IR[31]		IR[19:12]	IR[20]	IR[30:25]	IR[24:21]	0	J-immediate

Types of immediate produced by RISC-V instructions. The fields are labeled with IR bits used to construct their value.

Spring 2023

c. Extract other fields of the instruction register such as:

`funct7, funct3, rs1, rs2, rd, opcode`

Determine the validity of the extracted fields (especially `funct7`) based on the instruction type.

d. Determine the operation that is going to be taken by the ALU based on the fields of `opcode, funct3, funct7`.

31-25	24-20	19-15	14-12	11-7	6-0	
0000000	rs2	rs1	000	rd	0110011	ADD
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	100	rd	0000011	LBU
0000001	rs2	rs1	000	rd	0110011	MUL
0000000	rs2	rs1	rm	rd	1010011	FADD
0101100	00000	rs1	rm	rd	1010011	FSQRT
1101000	00001	rs1	rm	rd	1010011	FCVT.S.WU

Bonus point) Adding hardware support for additional instructions will be rewarded. For the ALU these instructions include arithmetic and logical instructions such as:

`SUB, SLLI, SRLI, OR, ORI, AND, ANDI, XOR, XORI, …`

2) Design the circuit needed for branch support:

31-25	24-20	19-15	14-12	11-7	6-0	
imm[12 | 10:5]	rs2	rs1	100	imm[4:1 | 11]	1100011	BLT

- **a.** Determine the taken branch signal as a conditional expression based on instruction type (i.e. if instruction is of type B).
  [Hint: The taken branch signal must default to zero]
- **b.** Compute the branch target to be taken by adding the immediate value of the instruction to the current value of PC.
- **c.** Modify the PC multiplexer expression (your program counter module must have this multiplexer expression).
  Modify the PC multiplexer to use the previous branch target when the previous instruction produced the taken branch signal.

Your Branch Unit includes the following inputs:

`Stage / branch_instruction / bus_rs1 (32 bits) / bus_rs2 (32 bits)`

And outputs:

`branch_enable`

On the second stage (`stage == 1`), check the RS1 < RS2 condition and assign it to a `reg` named `branch_condition`. Then check for `branch_condition` state, if it was equal to 1 (High), set `branch_enable` to `1'b1` and else, to `1b'0`.

Bonus point) Including jump instructions and other types of branches such as:

`JAL, BEQ, BNE, BGE, BLTU, BGEU`

3) Connect all your modules from this and the previous phase and simulate your code.

a. Monitor the PC counter and check if it changes correctly when a branch instruction is decoded.
b. Debug your code as needed.

Notes:

Send all your assignment related files (*Top module* and *Testbench* [.v] and [.vvp] and [.vcd] files) along with a detailed report in [.pdf] format all in one zip file to the email address of the class.

For this assignment you can work in groups of two, each participant must submit the assignment individually with their respective name and student number.

If you have any questions regarding this assignment, feel free to contact us.

Please submit your final project in the following format:

Name_StudentNumber_Project (BillGates_12345678_Project)

Good Luck!