PROGRAMMING PROJECT 2

Due: 18/06/2021 23:00

In this project, you are required to extend the MIPS single-cycle implementation by implementing additional instructions. You will use ModelSim simulator [1] to develop and test your code. You are required to implement your assigned **6 instructions** (selected from following 20 instructions). *Note that your set of **6 instructions** will be emailed to you or your group member.*

R-format (6): brn, brz, balrz, sll, sllv, srlv

I-format (11): xori , andi, nori, bgez, bgezal, bgtz, bltz, jm, jalm, jsp, jspal

J-format (3): bz, balz, jal

You must design the revised single-cycle datapath and revised control units which make a processor that executes your instructions as well as the instructions implemented already in the design. After designing the new enhanced processor, you will implement it in Verilog HDL.

Specifications of New Instructions

	Instr	Type	Code	Syntax	Meaning
1.	bltz	I-type	opcode=1	bltz $rs, Target	if R[rs] < 0, branches to PC-relative address (formed as beq & bne do)
2.	bgtz	I-type	opcode=38	bgtz $rs, Target	if R[rs] > 0, branch to PC-relative address (formed as beq & bne do)
3.	bgez	I-type	opcode=39	bgez $rs, Target	if R[rs] >= 0, branch to PC-relative address (formed as beq & bne do)
4.	bgezal	I-type	opcode=35	bgezal $rs, Target	if R[rs] >= 0, branch to PC-relative address (formed as beq & bne do), link address is stored in register 31.
5.	xori	I-type	opcode=14	xori $rt, $rs, imm16	Put the logical XOR of register $rs and the zero-extended immediate into register $rt.
6.	nori	I-type	opcode=13	nori $rt, $rs, imm16	Put the logical NOR of register $rs and the zero-extended immediate into register $rt.
7.	andi	I-type	opcode=12	andi $rt, $rs, imm16	Put the logical AND of register $rs and the zero-extended immediate into register $rt.
8.	brz	R-type	funct=20	brz $rs	if Status [Z] = 1, branches to address found in register $rs.

9. brn	R-type funct=21	brn $rs	if Status [N] = 1, branches to address found in register $rs.
10. balrz	R-type funct=22	balrz $rs, $rd	if Status [Z] = 1, branches to address found in register $rs link address is stored in $rd (which defaults to 31)
11. bz	J-type opcode=24	bz Target	if Status [Z] = 1, branches to pseudo-direct address (formed as j does)
12. balz	J-type opcode=26	balz Target	if Status [Z] = 1, branches to pseudo-direct address (formed as jal does), link address is stored in register 31
13. jal	J-type opcode=3	jal Target	jump to pseudo-direct address (formed as j does), link address is stored in register 31
14. jm	I-type opcode=16	jm imm16($rs)	jumps to address found in memory (indirect jump)
15. jalm	I-type opcode=17	jalm $rt, imm16($rs)	jumps to address found in memory (indirect jump), link addressis stored in $rt (which defaults to 31)
16. jsp	I-type opcode=18	jsp	jumps to address found in memory where the memory address is written in register 29 ($sp).
17. jspal	I-type opcode=19	jspal	jumps to address found in memory where the memory address is written in register 29 ($sp) and link address is stored in memory (DataMemory[Register[29]]).
18. sll	R-type func=0	sll $rd, $rt, shamt	shift register $rt to left by shift amount (shamt) and store the result in register $rd.
19. sllv	R-type func=4	sllv $rd, $rt, $rs	shift register $rt to left by the value in register $rs, and store the result in register $rd.
20. srlv	R-type func=6	srlv $rd, $rt, $rs	shift register $rt to right by the value in register $rs, and store the result in register $rd.

Status Register

Some of the conditional branches test the Z and N bits in the Status register. So the MIPS datapath will need to have a Status register, with the following 2 bits: Z (if the ALU result is zero) and N (if the ALU result is negative). The Status register will be loaded with the ALU results each clock cycle.

CSE 3038 – Computer Organization **Spring 2021**

Prof. Haluk Topcuoğlu

Requirements

- You will implement only 6 instructions. Therefore, you have to send an instruction request email lokman.altin@gmail.com with introducing your group members (3 or 4) as soon as possible.
- The deadline for the instruction request email is **04.06.2021 (Friday) 05:00pm**. Please do not send me an email later than that date.
- Your design should support all your 6 instructions without corrupting other MIPS instructions.
- You can print out single-cycle datapath and start your design on a paper. Then you can continue with the Verilog implementation.
- A demo session will be scheduled in the following week of the submission. I request you to bring these design papers within you in demo sessions.
- **You have to prepare a simulation of your instructions for the demo (including *register file*, *data memory* and *instruction memory*)!**
- You have to do your projects in Verilog with using ModelSim simulator.
- You are allowed to use only the source codes in the attached file. You cannot take any other implementations.
- You **will not** submit 6 different implementations. There should be only a single implementation supporting all 6 instructions that are assigned to you.
- You are required to submit a minimum 2-pages report explaining implementation details of your project and commented code (via Canvas). Your report should include example runs for each instruction. Your commented source code should include implementation detail of your project. All files should be submitted as a single zip file. You should use your surnames as the name of the file: *surname1_surname2_surname3_surname4_project2.zip*

General Policies

- *You are allowed to work in groups of 3 or 4 members. Group members may get different grades.*
- *It is NOT acceptable to copy (or start your) solutions from Web. In case of any forms of cheating or copying, the penalties will be severe.* ***Both Giver and Receiver are equally culpable and suffer equal penalties.***
- There will be demo session for this assignment. If you cannot answer the questions about your project details in the demo section, even if you have done all the parts completely, you will not get points!
- ***No late submission will be accepted!***

[1] https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/model-sim.html