CSE 3038 – Computer Organization **Spring 2020**
Prof. Haluk Topcuoğlu

PROGRAMMING PROJECT 2
Due: 29/05/2020 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 28 instructions). *Note that your set of **6 instructions** will be emailed to you or your group member.*

R-format (11): balrn, balrz, sll, srl, sllv, srlv, jmadd, jmor, jmsub, jalr, jr

I-format (13): xori , andi, ori, bneal, bgez, bgezal, bgtz, blez, bltzal, bltz, jrs, jrsal, jpc

J-format (4): baln, balz, balv, 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.	xori	I-type	opcode=14	xori $rt, $rs, Label	Put the logical XOR of register $rs and the zero-extended immediate into register $rt.
2.	andi	I-type	opcode=12	andi $rt, $rs, Label	Put the logical AND of register $rs and the zero-extended immediate into register $rt.
3.	ori	I-type	opcode=13	ori $rt, $rs, Label	Put the logical OR of register $rs and the zero-extended immediate into register $rt.
4.	bneal	I-type	opcode=45	bneal $rs, $rt, Target	if R[rs] != R[rt], branches to PC-relative address (formed as beq & bne do), link address is stored in register 31
5.	bgez	I-type	opcode=39	bgez $rs, Label	if R[rs] >= 0, branch to PC-relative address (formed as beq & bne do)
6.	bgezal	I-type	opcode=35	bgezal $rs, Label	if R[rs] >= 0, branch to PC-relative address (formed as beq & bne do), link address is stored in register 31.
7.	bgtz	I-type	opcode=38	bgtz $rs, Label	if R[rs] > 0, branch to PC-relative address (formed as beq & bne do)
8.	blez	I-type	opcode =6	blez $rs, Label	if R[rs] <= 0, branch to PC-relative address (formed as beq & bne do)
9.	bltzal	I-type	opcode=34	bltzal $rs, Label	if R[rs] < 0, branch to to PC-relative address (formed as beq & bne do), link address is stored in register 31.
10.	bltz	I-type	opcode=1	bltz $rs, $rt, Target	if R[rs] < 0, branches to PC-relative address (formed as beq & bne do)

11. 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)
12. balrn	R-type funct=23	balrn $rs, $rd	if Status [N] = 1, branches to address found in register $rs link address is stored in $rd (which defaults to 31)
13. jmadd	R-type funct=32	jmadd $rs,$rt	jumps to address found in memory [$rs+$rt], link address is stored in $31
14. jmor	R-type funct=37	jmor $rs,$rt	jumps to address found in memory [$rs\|$rt], link address is stored in $31
15. jmsub	R-type funct=34	jmsub $rs,$rt	jumps to address found in memory [$rs-$rt], link address is stored in $31
16. jalr	R-type func=9	jalr $rs, $rd	Unconditionally jump to the address found in register $rs, link address is stored in $rd.
17. jr	R-type func=8	jr $rs	Unconditionally jump to the address found in register $rs
18. balv	J-type opcode=33	balv Target	if Status [V] = 1, branches to pseudo-direct address (formed as jal does), link address is stored in register 31
19. 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
20. baln	J-type opcode=27	baln Target	if Status [N] = 1, branches to pseudo-direct address (formed as jal does), link address is stored in register 31
21. jal	J-type opcode=3	jal Target	jump to pseudo-direct address (formed as j does), link address is stored in register 31.
22. jpc	I-type opcode=30	jpc Target	jumps to PC-relative address (formed as beq and bne do) link address is stored in $rt (which defaults to 31)
23. jrs	I-type opcode=18	jrs $rs	jumps to address found in memory where the memory address is written in register $rs.
24. jrsal	I-type opcode=19	jrsal $rs	jumps to address found in memory where the memory address is written in register $rs and link address is stored in memory (DataMemory[$rs]).
25. 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.
26. srl	R-type func=2	srl $rd, $rt, shamt	shift register $rt to right by shift amount (shamt) and store the result in register $rd.
27. 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.
28. 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 3 bits: Z (if the ALU result is zero) , N (if the ALU result is negative) or V (if the ALU result causes overflow). The Status register will be loaded with the ALU results each clock cycle.

CSE 3038 – Computer Organization **Spring 2020**
Prof. Haluk Topcuoğlu

Requirements

- You will implement only 6 instructions. Therefore, you have to send a request email from canvas with introducing your group member.
- The deadline for the instruction request is **15.05.2020 (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. You cannot take any other implementations.
- You **will not** submit 6 different implementations. There should be only a single implementation supporting all your instructions.
- 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 implementation detail should be provided in the source code comment. All the files should be submitted as a single zip file. You should use your surnames as the name of the file: surname1_surname2_project2.zip

General Policies

- *You are allowed to work in groups of 2 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.mentor.com/company/higher_ed/modelsim-student-edition