JR Changes

This document describes necessary datapath and controller changes to add jr instruction to the MIPS architecture. You will implement these changes in System Verilog by referring to this document.

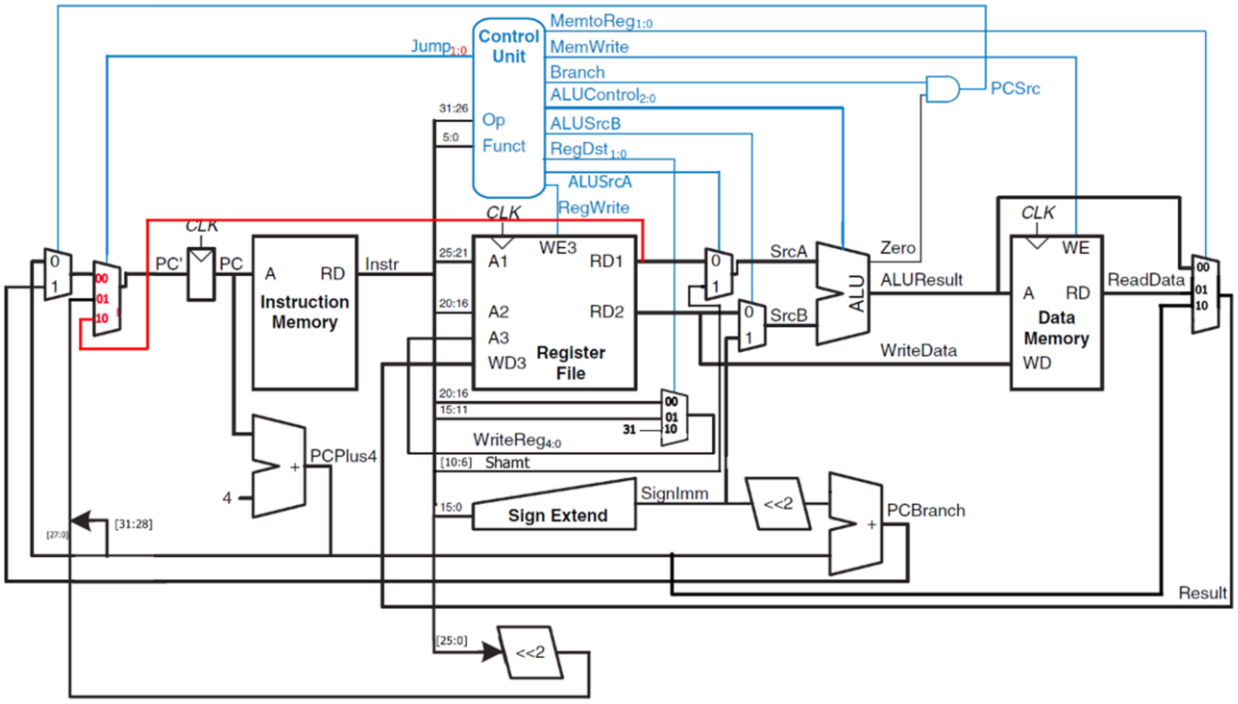

Figure 1: Datapath Changes (marked in red)

Instruction	Opcode	RegWrite	RegDst	ALUSrcA	ALUSrcB	Branch	MemWrite	MemToReg	ALUOp	Jump
R-type	000000	1	01	0	0	0	0	00	10	00
srl	000000	1	01	1	0	0	0	00	10	00
lw	100011	1	00	0	1	0	0	01	00	00
sw	101011	0	X	0	1	0	1	XX	00	00
beq	000100	0	X	0	0	1	0	01	01	00
addi	001000	1	00	0	1	0	0	00	00	00
j	000010	0	X	X	X	X	0	XX	XX	01
jal	000011	1	10	X	X	X	0	10	XX	01
jr	000000	1	01	0	0	0	0	00	10	10

Table 1: Controller Changes (marked in red)

Clarification: JR does not use X (don't cares) since it is R-Type and given this way to make it easier to implement in the given code. Please implement it according to the given specification in the document. However, the better way would be to implement it with don't cares (In such a case, we should have RegWrite = 0, Jump = 10, Memwrite = 0, ALUSrcA = ALUSrcB = Branch = MemWrite = ALUop = RegDst = X).

JR is an R-type instruction whose RTL expression is:
IM[PC], PC <= RF[RS]

Inclusion of JR instruction brings about some changes in both the datapath and controller of the MIPS architecture. First, realize that RD1 corresponds to RF[RS], therefore it must be assigned to PC. Multiplexer which selects the PC value is transformed from a MUX-2 to MUX-4 to capture that. Due to the MUX change, Jump signal becomes 2-bit value. Hence, in the datapath Jump values are all converted to two bits and for jr instruction Jump value is 10. Note that although jr is an R-type instruction, we added a new row for it in the Table 1 as its Jump value is different than the R-type instructions of the Original12. Now with these in mind you can go and code these changes up in System Verilog. As you know, you will modify “Complete MIPS model.txt” file.