

## (5 pts) Exercise 4-1 (Single-cyc impl.)

---

Fill in the needed control value (0 or 1) for each case

Inst.	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp 1	ALUOp 0
R-format									
lw									
sw									
beq									

## (5 pts) Exercise 4-2 (Single-cyc impl.)

---

Fill in the correct signal name by looking back at the datapath diagram.

Possibilities: **ALUSrc**, **MemtoReg**, **MemRead**, **MemWrite**, **PCSrc**, **RegDst**, **RegWrite**

Signal Name	Effect when deasserted	Effect when asserted
	The register destination number for the write register comes from the rt field (bits 20-26)	The register destination number for the write register comes from the rd field (bits 15-11)
	None	The register on the Write register input is written with the value on the Write data input
	The second ALU operand comes from the second register file output (Read data 2)	The second ALU operand is sign-extended, lower 16 bits of the instruction
	The PC is replaced by the output of the adder that computes the value of PC+4	The PC is replaced by the output of the adder that computes the value of branch target
	None	Data memory contents designated by the address input are put on the Read Data output
	None	Data memory contents designated by the address input are replaced by the value on the Write Data input
	The value fed to the register Write data input comes from ALU	The value fed to the register Write data input comes from the data memory

## (15 pts) Exercise 4-6

---

A “stuck-at-0” fault is a defect that can occur during manufacturing, where a particular signal becomes hardwired to zero. Considering the single-cycle implementation shown in Figure 4.17 on page 322, describe the effect that a stuck-at-0 fault would have for each of the following signals. Which instructions, if any, will not work correctly? Explain why. The first is done for you as an example. Consider these instructions: R-type, lw, sw, beq.

NOTE – explain specifically what goes wrong in each case – do NOT just say “MemRead is supposed to one for that instruction.”

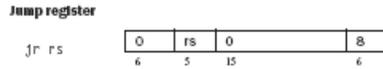
- ALUSrc = 0. *lw and sw would not work, because the immediate value from the instruction couldn't be provided to the ALU as needed.*
- MemRead = 0
  
- MemWrite = 0
  
- ALUop1 = 0
  
- ALUop0 = 0
  
- RegWrite = 0

(blank space)

---

## (15 pts) Exercise 4-7

•Consider the jr instruction (jump register), which is described as follows:

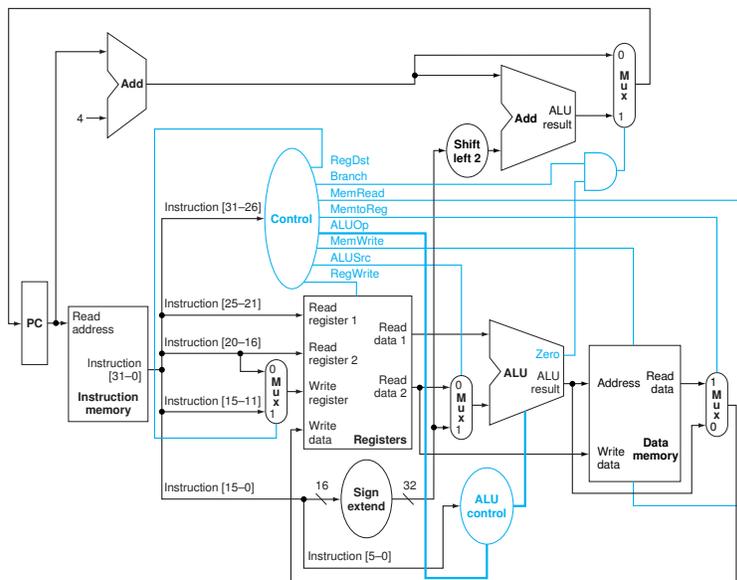


Unconditionally jump to the instruction whose address is in register rs.

We wish to add this instruction to our single-cycle implementation. To make this happen, 1.) add any necessary hardware (gates, adders, wires, etc.) to the single-cycle datapath shown below and 2.) modify the control chart below (add a new row and any new signals, if necessary).

**Note:** 'jr \$s0' states that the next PC value should come from register \$s0. It is **NOT** the same as instructions like 'j Loop', whose datapath and control is described on page 328 (though reading about that may help with the general idea)

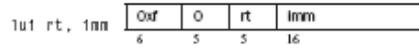
Instr	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1



## (15 pts) Exercise 4-8

•Do the same as previous exercise, but for the lui instruction:

Load upper immediate



Load the lower halfword of the immediate *imm* into the upper halfword of register *rt*. The lower bits of the register are set to 0.

You can find more info on this instruction in Section 2.10.

Again, table/figure below for you to modify. There are multiple ways to solve this problem; provide some brief text explaining how your solution works. Make sure that the other instructions continue to work.

Instr	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

