IC220: HW 7

Due: 3 Apr 2019

Full Name: _____ **Alpha:** _____

Circle Your Section: Aviv/1001 Aviv/2001 Aviv/4001 Choi/5001 Missler/5002

Total Points: 62

Preliminary: Carefully do the assigned reading for Chapter 4 (4.1-4.4)

---

1. **[10 points]** Fill in the needed control value (0 or 1) for each case:

Inst.	RegDest	ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUOp 1	ALUOp 2
R-Type									
lw									
sw									
beq									

2. Fill in the correct signal name by looking back at the data path diagram. Possible answers include: ALUSrc, MemtoReg, MemRead, MemWrite, Branch, RegDst, RegWrite

(a) **[1 point]** Signal Name _____

- **Effect when de-asserted (0):** The register destination number for the *write registe*r comes from the `rt` field (bits 20-26)
- **Effect when asserted (1):** The register destination number for the *write register* comes from the `rd` field (bits 15-11)

(b) **[1 point]** Signal Name _____

- **Effect when de-asserted (0):** None
- **Effect when asserted (1):** The register on the *write register* input is written with the value on the Write data input

(c) **[1 point]** Signal Name _____

- **Effect when de-asserted (0):** The second ALU operand comes from the second register file output (*Read Data 2*)
- **Effect when asserted (1):** The second ALU operand is sign-extended, lower 16 bits of the instruction

(d) **[1 point]** Signal Name _____

- **Effect when de-asserted (0):** The PC is replaced by the output of the adder that computes the value of PC+4
- **Effect when asserted (1):** The PC is replaced by the output of the adder that computes the value of branch target (privided taht *Zero* signal is true)

(e) **[1 point]** Signal Name _____

- **Effect when de-asserted (0):** None
- **Effect when asserted (1):** Data memory contents designated by the address input are put on the *Read Data* output

(f) **[1 point]** Signal Name _____

- **Effect when de-asserted (0):** None
- **Effect when asserted (1):** Data memory contents designated by the address input are replaced by the value of the *Write Data* input

(g) **[1 point]** Signal Name _____

- **Effect when de-asserted (0):** The value fed to the register *Write Data* input comes from ALU
- **Effect when asserted (1):** The value fed to the register *Write Data* input comes from the data memory

3. 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 265, **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 the 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." ALSO – is it completely broken, or only sometimes??

   (a) ALUSrc=0 :

   lw and sw would not work, because the immediate value from the instruction couldn't be provided to the ALU as needed.

   (b) **[3 points]** MemRead = 0

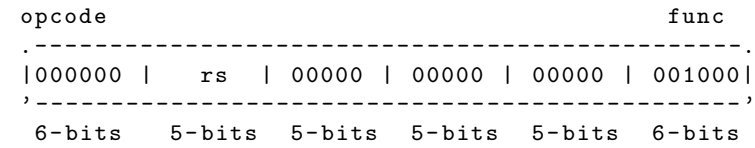
   (c) **[3 points]** MemWrite = 0

   (d) **[3 points]** ALUop1 = 0

   (e) **[3 points]** ALUop0 = 0

   (f) **[3 points]** RegWrite = 0

4. **[15 points]** Consider the `jr $rs` instruction (jump register) which is described as an R-type instruction that sets the next PC to the value in register `$rs` — not the same as the immediate `j` instruction. The `jr` has the following bit layout.



We wish to add this instruction to our single-cycle implementation. To make this happen, we

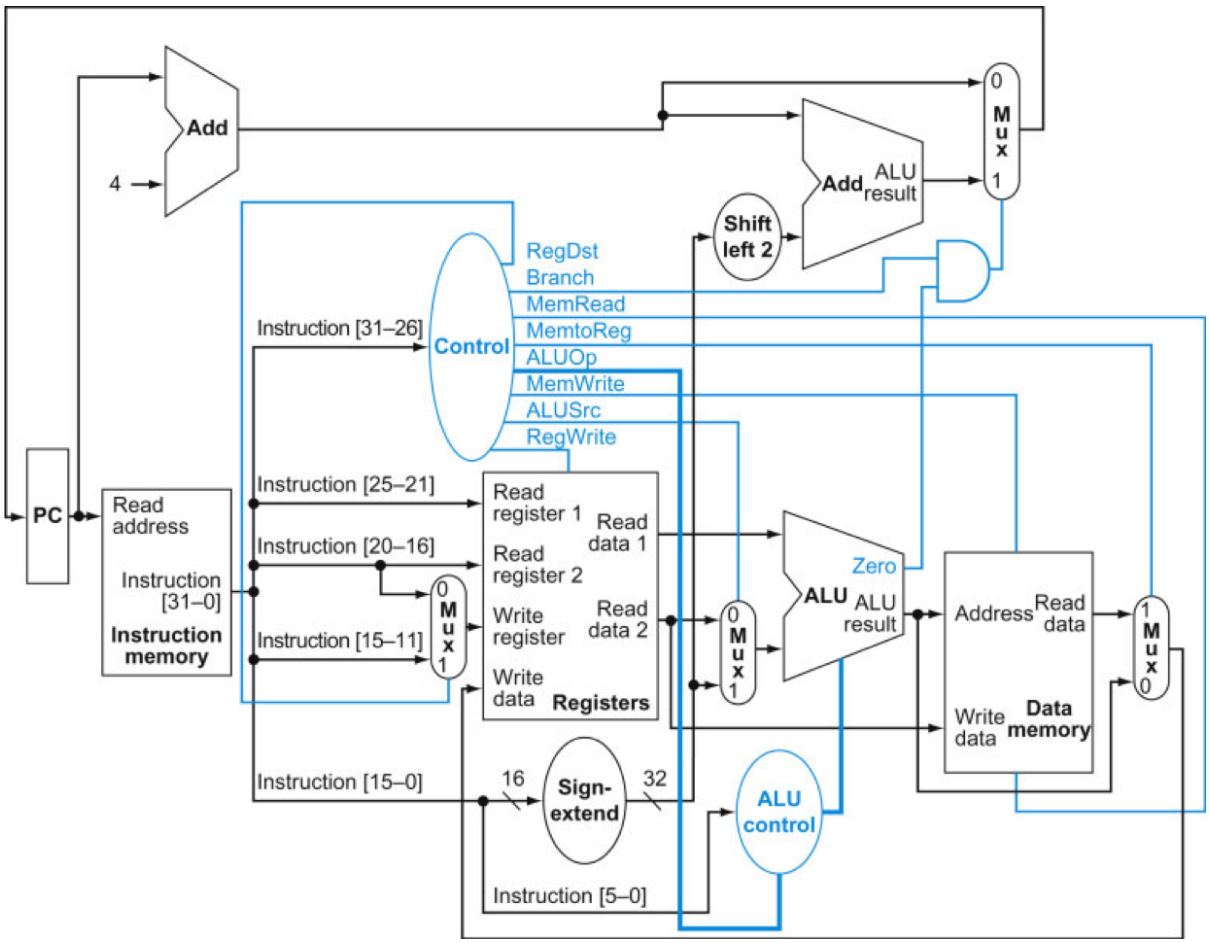
(a) modify the control chart below by adding a new row and any new signals if necessary, and

(b) necessary hardware (gates, addres, wires, etc.) to the single-cycle datapath.

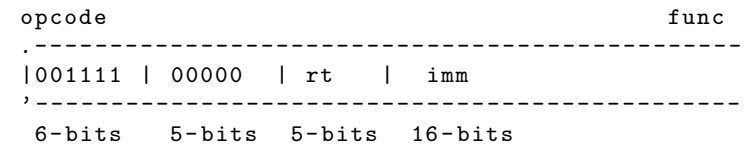
Modify the control chart below by completing the `jr` row and adding any columns for new signals you may need to complete the logic. **Hint: You might want to draw the gates first before doing this part.** Also, **note** that the values for signals for `R-Type`, `lw`, `sw`, and `beq` are left blank because they are answers to another question, but you should still be filling in the `jr` row of the table and the signal values for any new control signals you may need to add.

Inst.	RegDest	ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUOp 1	ALUOp 2
R-Type									
lw									
sw									
beq									
jr									

Add the necessary hardware (gates, addresses, wires, etc.) to the single cycle data/control path picture below:



5. [**15 points**] Do the same as the previous question for the `lui` instruction. Recall that the `lui` (load upper immediate) has the following format for `lui rt, imm`



where the lower halfword (16-bits) of the immediate vlaue `imm` is loaded into the upper halfword of register `rt`. The lower bits of the register are set to 0.

There are multiple ways to solve this problem. In addition to completing the chart, **you are required to provide some brief text explaining how your solution works below.**

Modify the control chart below by completing the `lui` row and adding any columns for new signals you may need to complete the logic. **Hint: You might want to draw the gates first before doing this part.** Also, **note** that the values for signals for `R-Type`, `lw`, `sw`, and `beq` are left blank because they are answers to another question, but you should still be filling in the `jr` row of the table and the signal values for any new control signals you may need to add.

Inst.	RegDest	ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUOp 1	ALUOp 2
R-Type									
lw									
sw									
beq									
lui									

Add the necessary hardware (gates, addresses, wires, etc.) to the single cycle data/control path picture below: