

Exam 2

Name: _____

Instructor: _____

Periods: _____

Instructions:

- Write all answers on these pages. Use the back as necessary.
- Clearly indicate your final answer.
- For full credit, show your work, and document your code.
- Read the entire examination before starting, and then budget your time.

Authorized resources:

- Reference materials provided by the instructor at the time of the exam.
- Both sides of one 8½”x11” sheet of paper containing only handwritten material.

Unauthorized resources:

You are NOT permitted to use any resources other than those identified above. In particular, you may NOT use books, notes, electronic files, calculators, PDAs, or computers.

Good luck!

Problem Number	Maximum Points	Points Earned
1	30	
2	40	
3	15	
4	15	
Total	100	

Problems

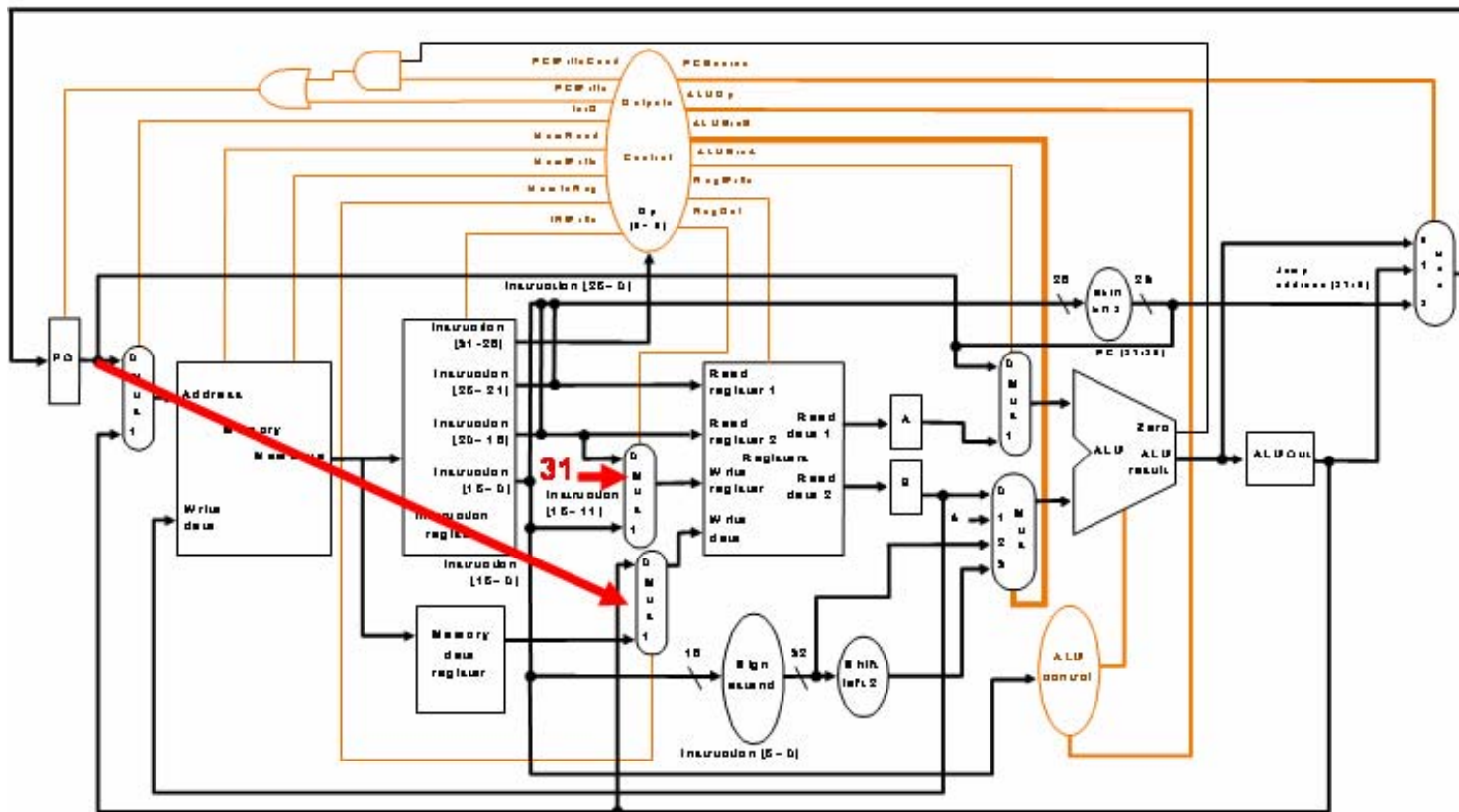
Problem 1. On the following pages, you are provided with RTL descriptions of two new MIPS instructions, as well as two modified versions of the MIPS datapath developed in the textbook, each of which supports one of the new instructions. You must specify the changes to the state transition diagram so that the control unit also supports the new instructions. The instructions are:

- `jal target` – Unconditionally jump to the instruction at `target`. Save the address of the next instruction in register `$ra`.
- `swap r1 r2` – The `swap` instruction will result in the register `r1` being assigned the contents of `r2` and register `r2` being the assigned the contents of `r1`.

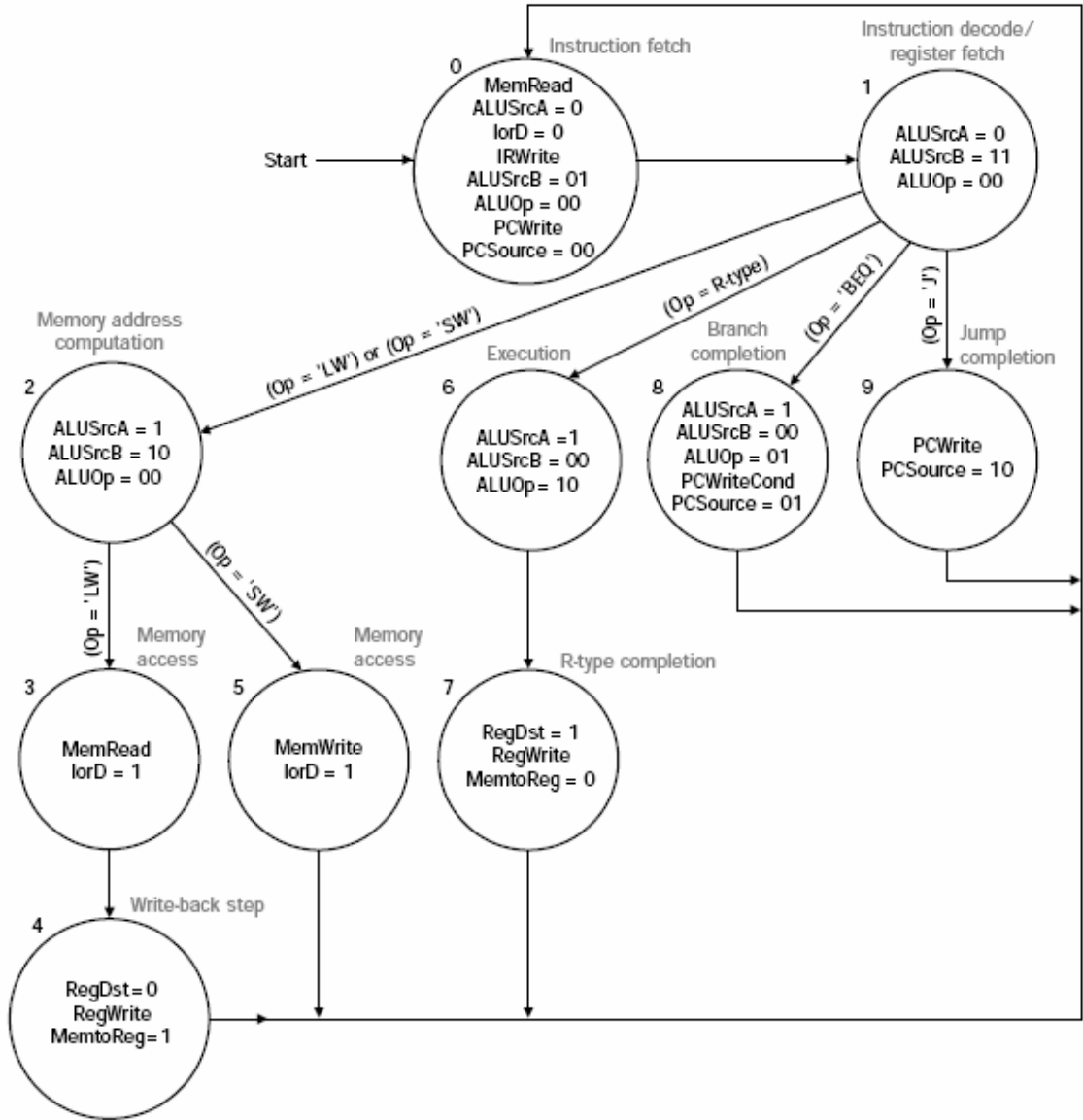
The RTL for the two new instructions is provided in the following table, along with the RTL for the `j` and R-type instructions

Cycle	Action for R-type instruction	Action for <code>j</code>	Action for <code>jal</code>	Action for <code>swap</code>
1.	$\text{IR} = \text{Memory}[\text{PC}]$ $\text{PC} = \text{PC} + 4$			
2.	$A = \text{Reg}[\text{IR}[25-21]]$ $B = \text{Reg}[\text{IR}[20-16]]$ $\text{ALUOut} = \text{PC} + \text{sign-extend}(\text{IR}[15-0] \ll 2)$			
3.	$\text{ALUOut} = A \text{ op } B$	$\text{PC} = \text{PC}[31-28] \parallel (\text{IR}[25-0] \ll 2)$	$\text{PC} = \text{PC}[31-28] \parallel (\text{IR}[25-0] \ll 2)$ $\text{Reg}[31] = \text{PC}$	$\text{Reg}[\text{IR}[20-16]] = A;$ $B = B$
4.	$\text{Reg}[\text{IR}[15-11]] = \text{ALUOut}$			$\text{Reg}[\text{IR}[25-21]] = B$

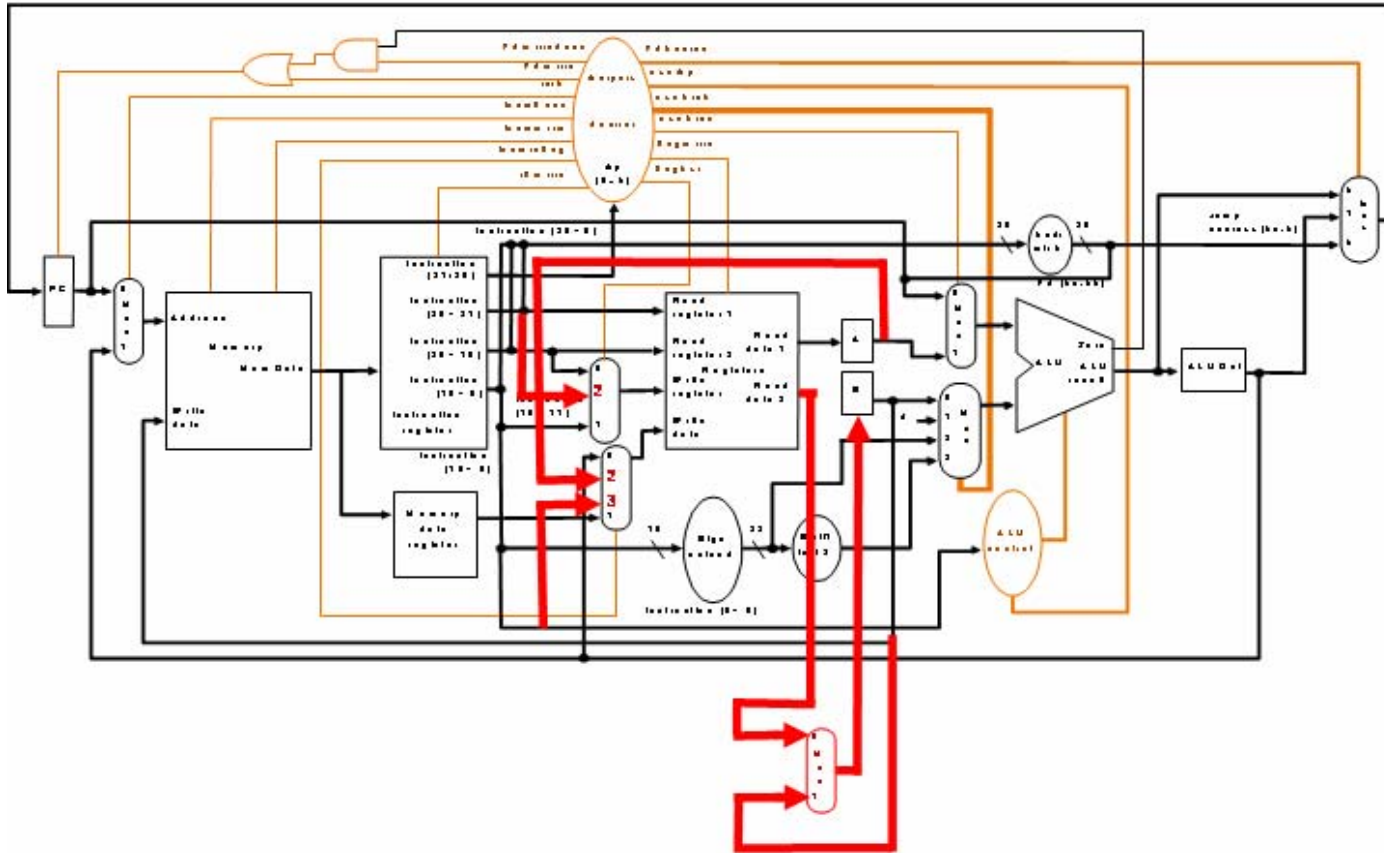
- a. [15 points] The modifications to the datapath necessary to implement jal are shown below. On the following page, show the modifications to the state transition diagram necessary to implement the jal instruction.



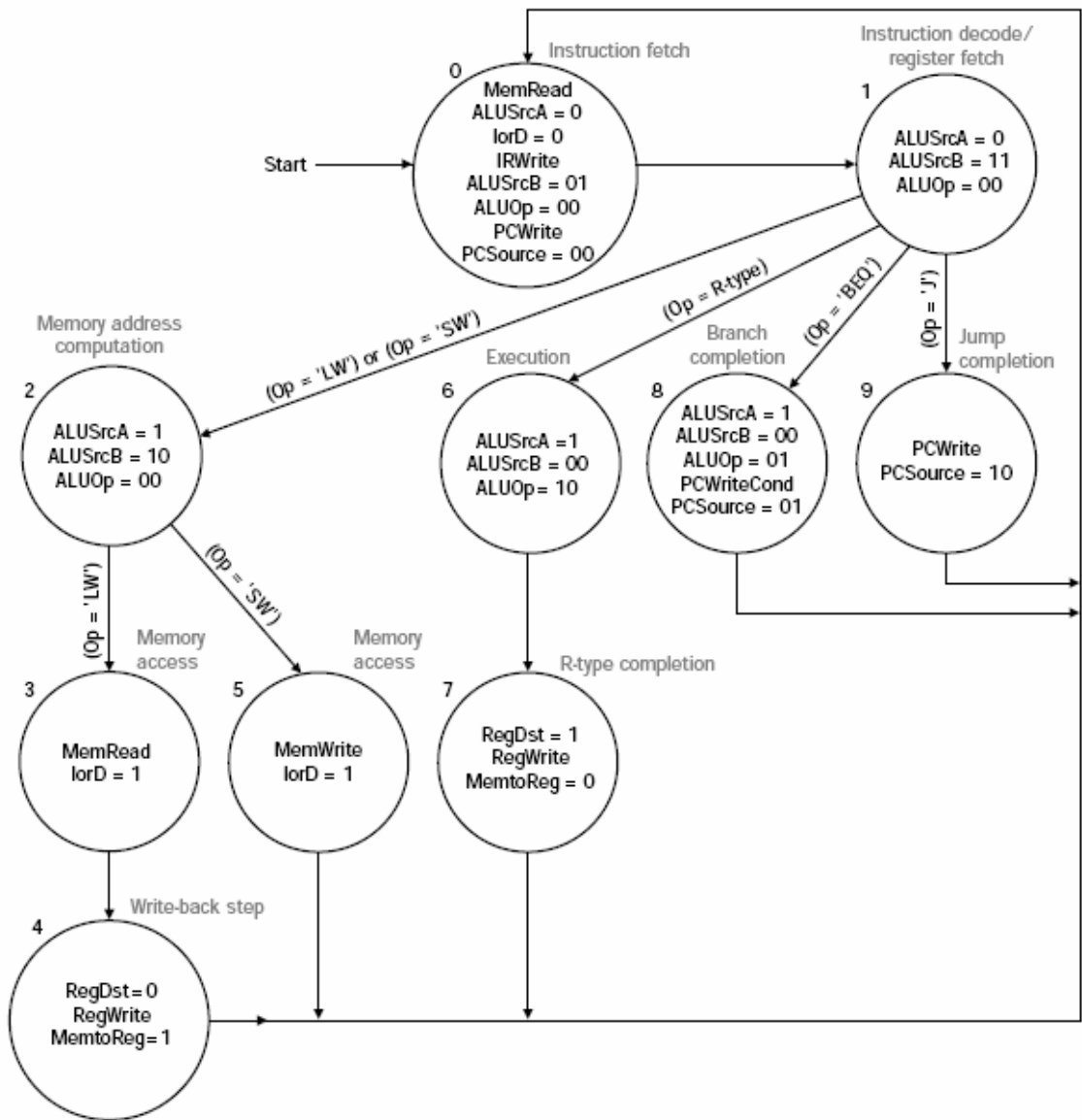
Note: MemToReg and RegDst must each be increased to two bits.



- b. [15 points The modifications to the datapath necessary to implement swap are shown below. On the following page, show the modifications to the state transition diagram necessary to implement the swap instruction.



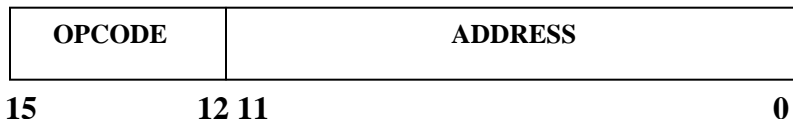
Note: MemToReg and RegDst must each be increased to two bits. Also, a new control signal called BSRc will control the new MUX.



Problem 2. [40 points] MARIE, A Machine Architecture that is Really Intuitive and Easy, is a simple architecture consisting of memory, eight registers and an ALU. Five of the registers in MARIE are listed below:

- **AC: Accumulator** – This is the only general purpose register. It holds 16-bit data that the CPU needs to process.
- **MAR: Memory Address Register** – Holds the 12-bit memory address of the data/instruction being referenced.
- **MDR: Memory Data Register** – Holds the 16-bit data that was read from memory, or the data that is to be written to memory.
- **PC: Program counter** – Holds the 12-bit address of the next instruction.
- **IR: Instruction Register** – Holds the 16-bit instruction that is to be executed.

The MARIE instruction format is as shown below:



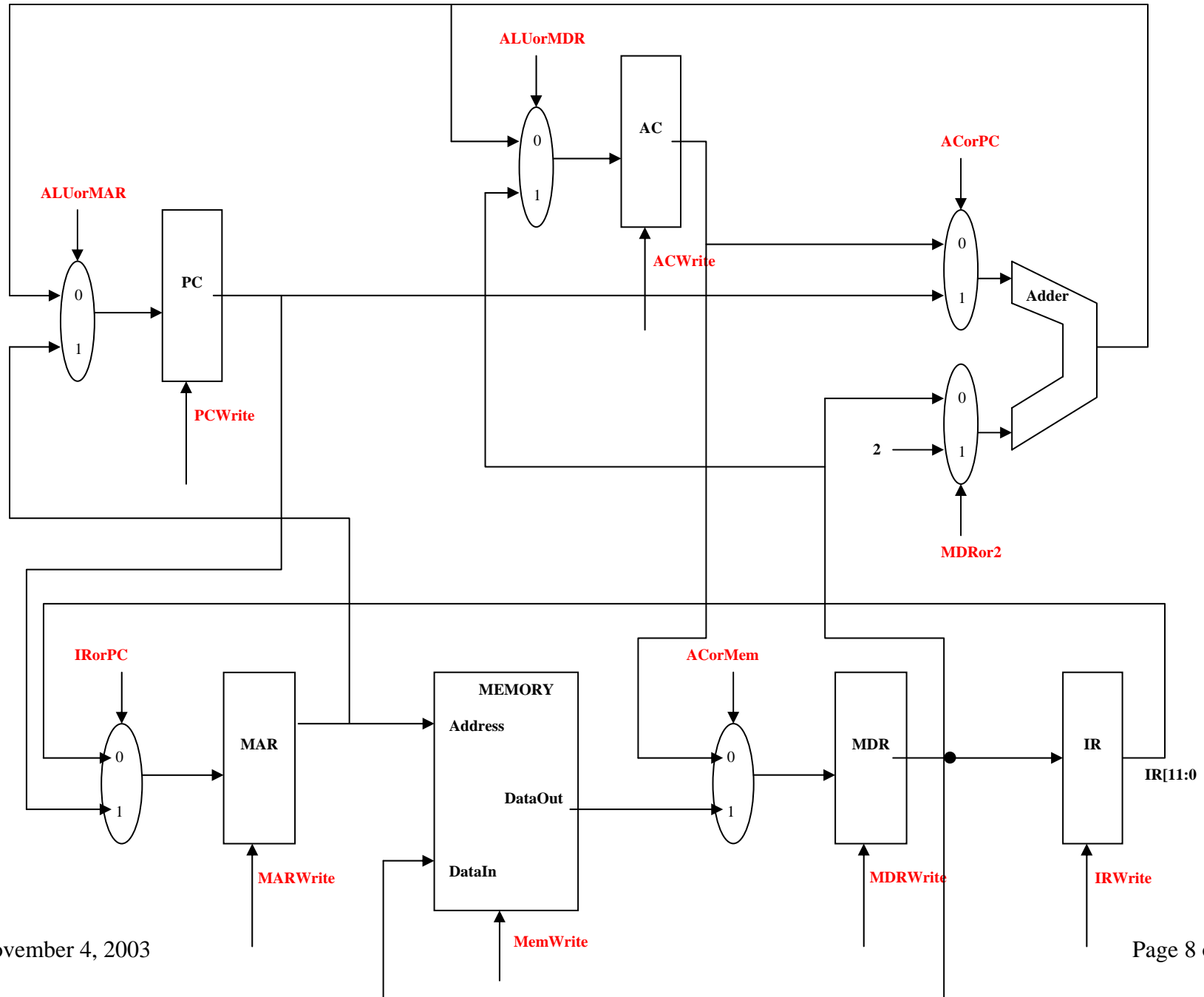
Four of the instructions in MARIE are listed below:

- **LOAD X** – Load the contents of address X into AC.
- **STORE X** – Store the contents of AC at address X.
- **ADD X** – Add the contents of address X to the contents of AC and store the result in AC.
- **JUMP X** – Load the value of X in PC.

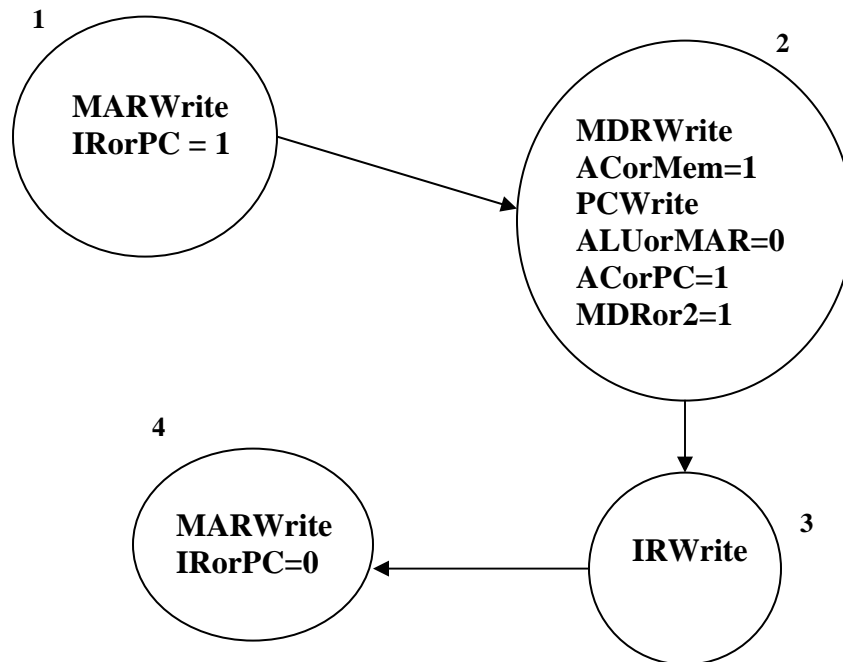
The RTL for four MARIE Assembly language instructions is listed in the table:

Cycle	Action for LOAD	Action for STORE	Action for ADD	Action for JUMP
1.			MAR = PC	
2.			MDR = Mem[MAR] PC = PC + 2	
3.			IR = MDR	
4.			MAR = IR[11:0] If (IR[15:12] == ZZZZ) then	
5.	MDR = Mem[MAR]	MDR = AC	MDR = Mem[MAR]	PC = MAR
6.	AC = MDR	Mem[MAR] = MDR	AC = AC + MDR	

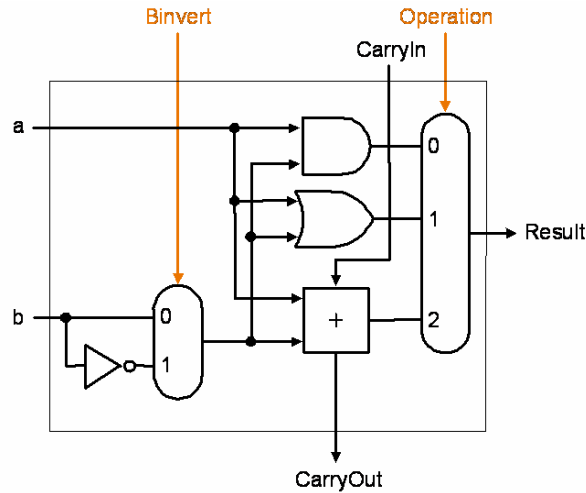
The complete datapath for MARIE is shown below. You may not change it.



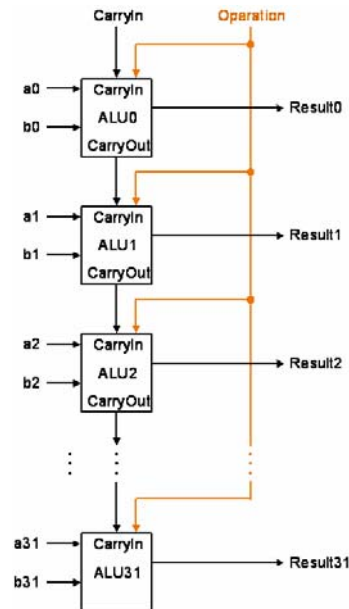
States 1, 2, 3 and 4 of the state transition diagram for the finite state machine that can be used to implement the control unit for MARIE are provided below. They correspond to steps 1, 2, 3 and 4 of the RTL description, respectively. Complete the state transition diagram to implement the LOAD, STORE, ADD and JUMP instructions.



Problem 3. [15 points] The figure below shows a 1-bit ALU that performs the bitwise AND, bitwise OR, addition, and subtraction operations.



The next figure shows a 32-bit ALU constructed from 32 1-bit ALUs.



Show the modifications necessary and state how to set the control signals so that the ALU performs the bitwise NOT operation.

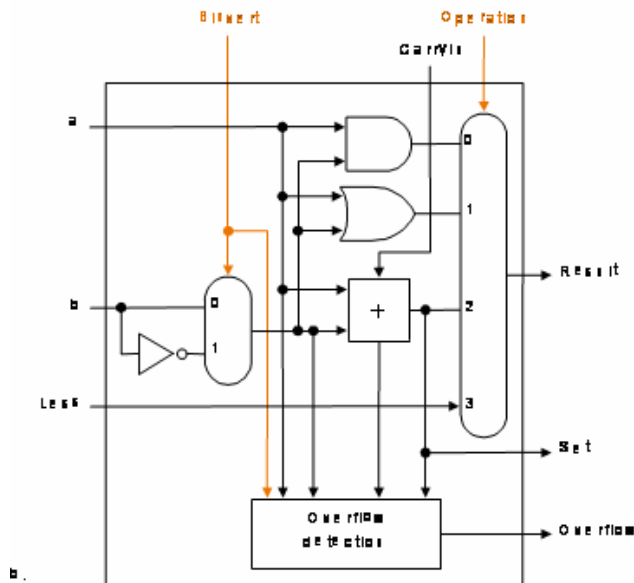
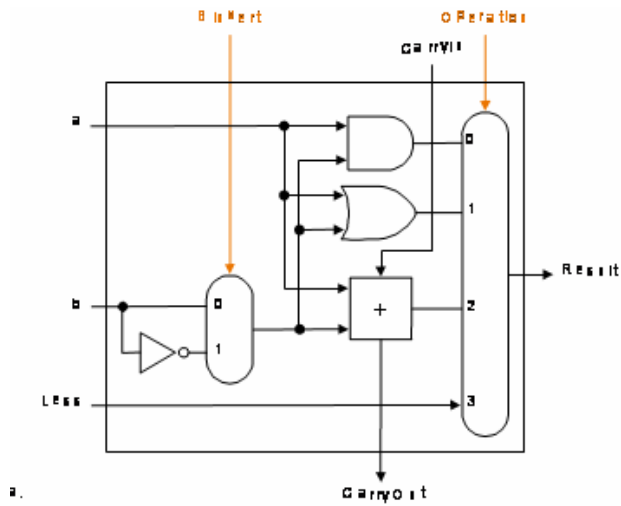
Note: the bitwise NOT of

01001100011100001111000001111101

is

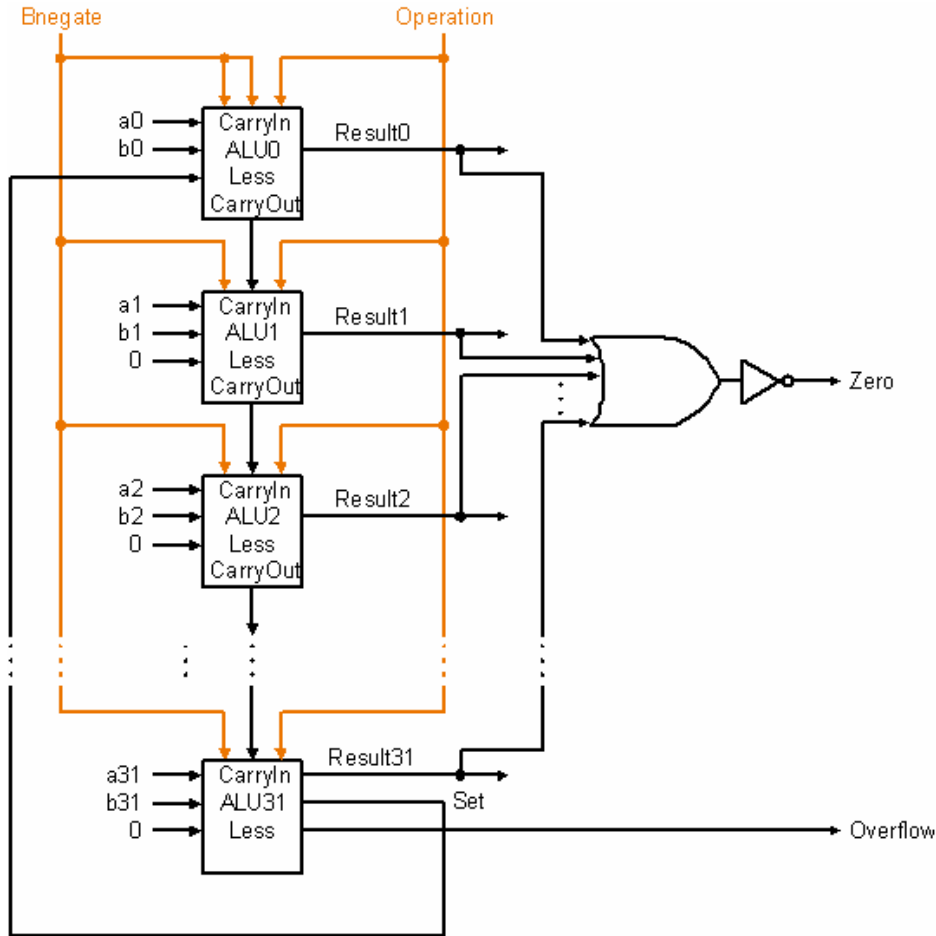
10110011100011110000111110000010

Problem 4. [15 points] The two figures below show a 1-bit ALU that performs the bitwise AND, bitwise OR, addition, subtraction, and “set less than” operations and a corresponding 1-bit ALU for the most significant bit.



This problem is continued on the next page.

The next figure shows a 32-bit ALU constructed from those 1-bit ALUs.



Show the modifications necessary and state how to set the control signals so that the ALU performs the equivalence operation. Specifically, the output of the ALU should be:

000000000000000000000000000001, if the two operands are identical, and

000000000000000000000000000000, otherwise.