ID: A

CSE 2021 Computer Organization Quiz #2 (Fall 2009)

Instructions

- ❑ This is a closed book, 80 minutes quiz.
- ❑ The MIPS reference sheet may be used as an aid for this test.
- ❑ An 8.5" x 11" "Cheat Sheet" may also be used as an aid for this test. MUST be original handwriting.
- ❑ This is a question/answer booklet: Write your answers in the space provided and as indicated in each question. Use the backside for scratch work. *Do not* hand in anything other than this booklet.
- ❑ Fill in your personal data in the box below *before the start of the exam* and then wait until the instructor has distributed the exams to all students. *Do not* turn this page over until the instructor has announced that you may do so.
- ❑ Keep your York photo ID (or any other acceptable photo ID) on the desk in front of you so that the instructor may inspect it without disturbing you.
- ❑ You may use ONLY those instructions that appear in the MIPS sheet. Whenever needed, assume that **the machine is little endian.**
- ❑ No questions during the quiz. Write your final answer with a pen.

LAST NAME: ________________________________

FIRST NAME: ________________________________

STUDENT NUMBER: ________________________________

PRISM LOGIN: ________________________________

Section	Points	Mark
A	20	
B	20	
TOTAL	**40**	

.

Name:___________________________

Student #:________________________

Section A <1 question = 20 points>

For each question, write your answer in the box provided.

1. Study the testbench Verilog module given below (next page).

   (a) Write your own module which is instantiated in testbench which takes the 2-bit inputs a and b and asserts its output c if the following condition is true a[1] = b[1] AND a[0] = b[0] - in other words the modules asserts its output if the inputs are found to be the same.

   module

   end module

   (b) What is the expected output from testbench when it and your module from part (a) are compiled and run using the iverilog and vvp commands?

Listing for module testbench:

```
module testbench;

        reg [1:0] test_a, test_b;
        wire test_c;

        eq2 uut_eq2(test_a, test_b, test_c);

        initial
        begin

                #200
                test_a = 2'b10;
                test_b = 2'b10;
                #1 $display($time, " a = %d, b = %d, c = %d",test_a, test_b, test_c);
                #200
                test_a = 2'b00;
                test_b = 2'b10;
                #1 $display($time, " a = %d, b = %d, c = %d",test_a, test_b, test_c);
        end
endmodule
```

Section B <1 question = 20 points>

.

2. Consider a new I-type instruction `jro` (jump to register+offset) with the following format:

opcode	rs	Not Needed	Immediate
31 26	25 21		15 0

that executes PC = RF[rs]+SignExtImm.

(a) Add the necessary circuitry to the single cycle datapath of Fig. 1 to implement the `jro` instruction.

(b). Append Fig. 2 to add the necessary control signals needed for the `jro` instruction.

Name:____________________________

Student #:_________________________

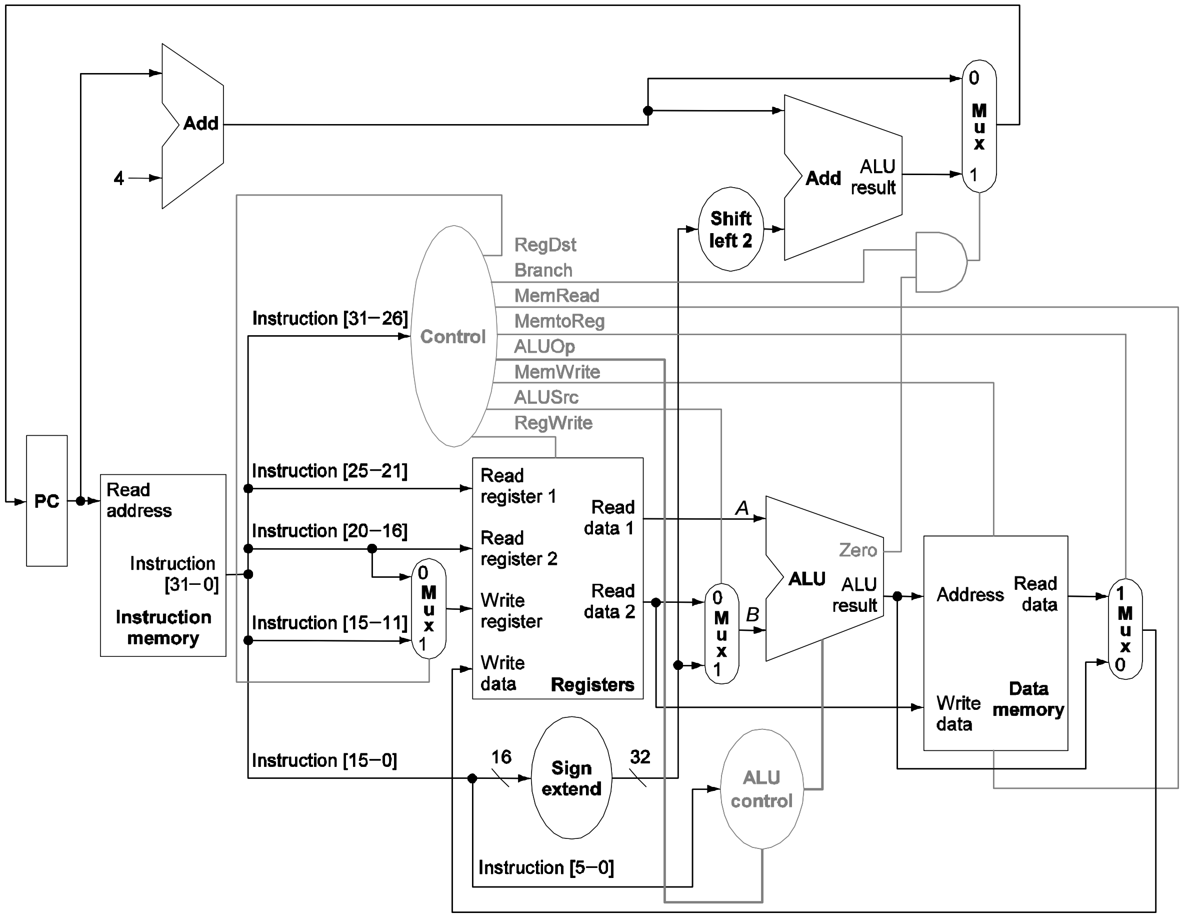


Fig .1: Datapath for R-type, lw, sw, and beq instructions. Add additional hardware, if required, to implement the datapath for the instruction specified in the problem statement. Also, highlight the datapath for the instruction.

Instruction	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp2	
`jro`										

Fig. 2: Control Signals for `jro` in a single cycle implementation. Add extra columns to add any additional control signals, if needed. The control signals that are not being used by the `jro` instruction should be marked as don't care (X) condition.

CSE 2021 Computer Organization Quiz #2 (Fall 2009)

Answer Section

SHORT ANSWER

1. ANS:
(a) Here is the listing for module eq2:

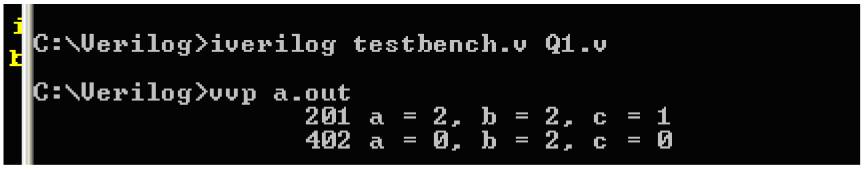
```
module eq2(a, b, c);
        input [1:0] a, b; // 2-bit signals
        output c;

        wire p0, p1, p2, p3;

        assign c = p0 | p1 | p2 | p3;
        assign p0 = (~a[1] & ~b[1]) & (~a[0] & ~b[0]);
        assign p1 = (~a[1] & ~b[1]) & (a[0] & b[0]);
        assign p2 = (a[1] & b[1]) & (~a[0] & ~b[0]);
        assign p3 = (a[1] & b[1]) & (a[0] & b[0]);

endmodule
```

(b)



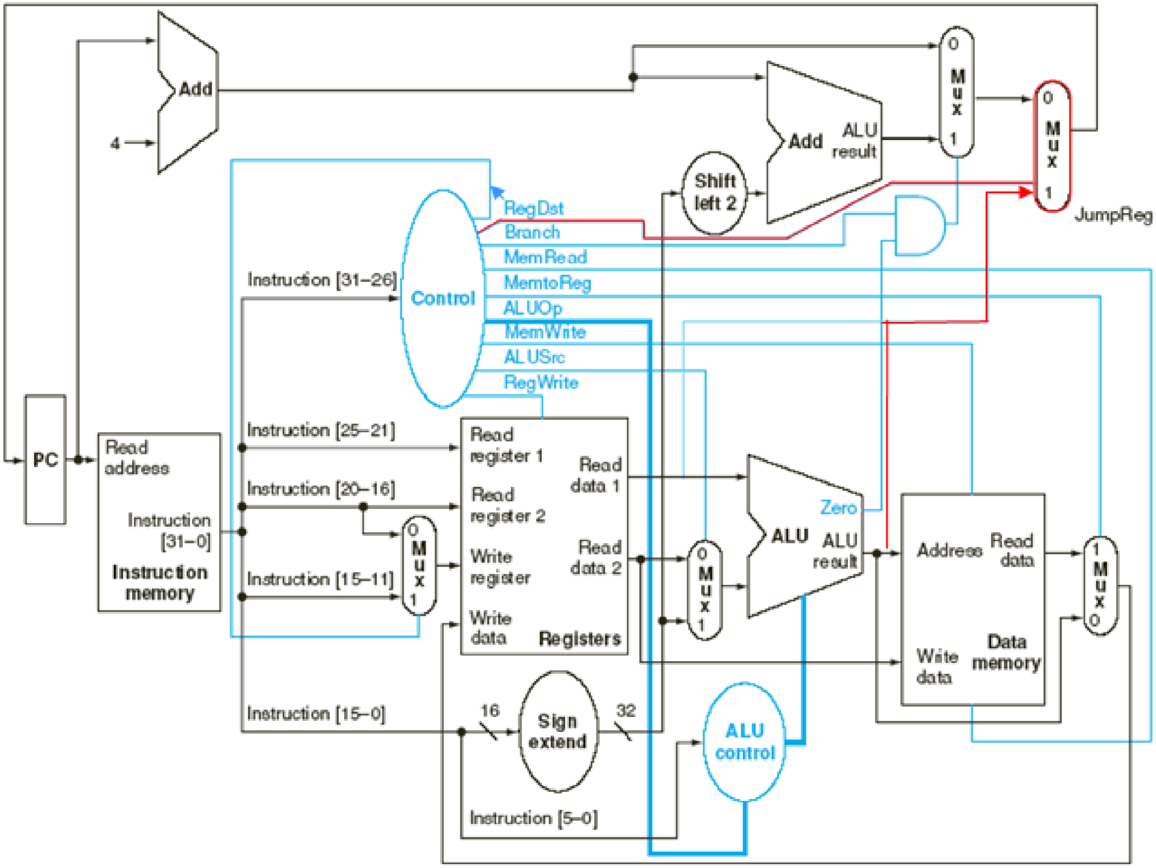
```
C:\Verilog>iverilog testbench.v Q1.v

C:\Verilog>vvp a.out
                201 a = 2, b = 2, c = 1
                402 a = 0, b = 2, c = 0
```

PTS: 1

PROBLEM

2. ANS:
(a) The modified data path is shown in red. A JumpReg control is required to control the extra mux required for Question (b)



(b)

Instruction	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	JumpReg	ALUOp1	ALUOp0
R-format	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**
lw	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**
sw	**X**	**1**	**X**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
beq	**X**	**0**	**X**	**0**	**0**	**0**	**1**	**0**	**0**	**1**
jr	**X**	**1**	**X**	**0**	**0**	**0**	**X**	**1**	**0**	**0**

PTS: 1