

**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	
<b>TOTAL</b>	<b>40</b>	

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  $a[1] = \text{NOT}(b[1])$  AND  $a[0] = \text{NOT}(b[0])$  - in other words your module will assert its output if the inputs are the two's complement of each other.

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?

Name: \_\_\_\_\_

Student #: \_\_\_\_\_



Listing for module testbench:

```
module testbench;

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

    neg2 uut_neg2(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'b01;
        test_b = 2'b10;
        #1 $display($time, " a = %d, b = %d, c = %d",test_a, test_b, test_c);

    end
endmodule
```

Name: \_\_\_\_\_

Student #: \_\_\_\_\_



**Section B** <1 question = 20 points>

2. Consider the instruction `sll` (shift left logical) with the following format:

opcode		Not needed		rt		rd		sa		Function	
31	26	25	21	20	16	15	11	10	6	5	0

that executes  $RF[rd] = RF[rt] \ll sa$ . Assume that the ALU is capable of shifting the operand applied at the upper input  $A$  by the number of bits specified at the lower input  $B$  when its control input is set to 11.

- Add the necessary circuitry to the single cycle datapath of Fig. 1 to implement the `sll` instruction.
- Append Fig. 2 to add the necessary control signals needed for the `sll` instruction.

Name: \_\_\_\_\_

Student #: \_\_\_\_\_

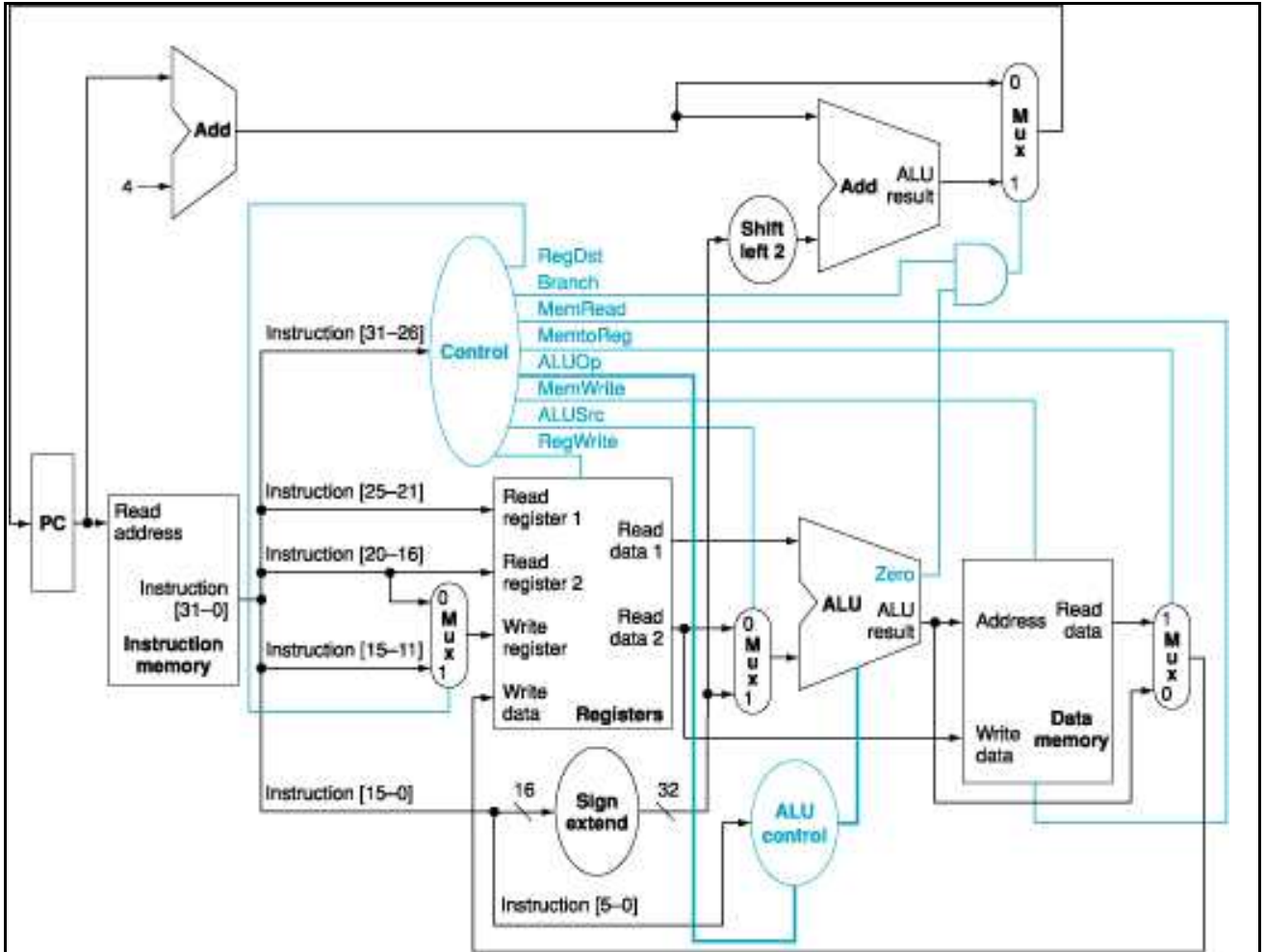


Fig .1: Datapath for R-type, lw, sw, and beq instructions. Add additional hardware, if required, to implement the datapath for the instruction specified in the problem statement. Also, highlight the datapath for the instruction.

Instruction	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp2	
sll										

Fig. 2: Control Signals for sll in a single cycle implementation. Use the extra column to add any additional control signal, if needed.

CSE 2021 Computer Organization Quiz #2 (Fall 2009)Answer Section

## SHORT ANSWER

1. ANS:

(a) Here is the listing for module neg2:

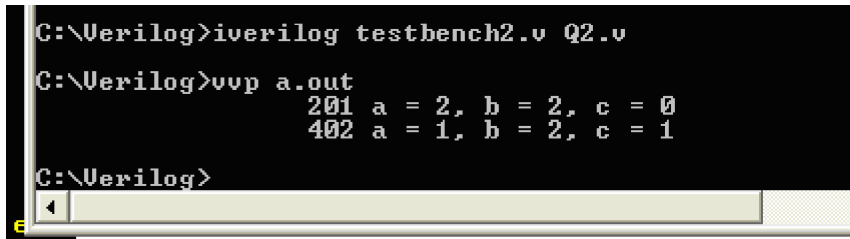
```
module neg2(a, b, c);
    input [1:0] a, b; // 2-bit signals
    output c;

    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)



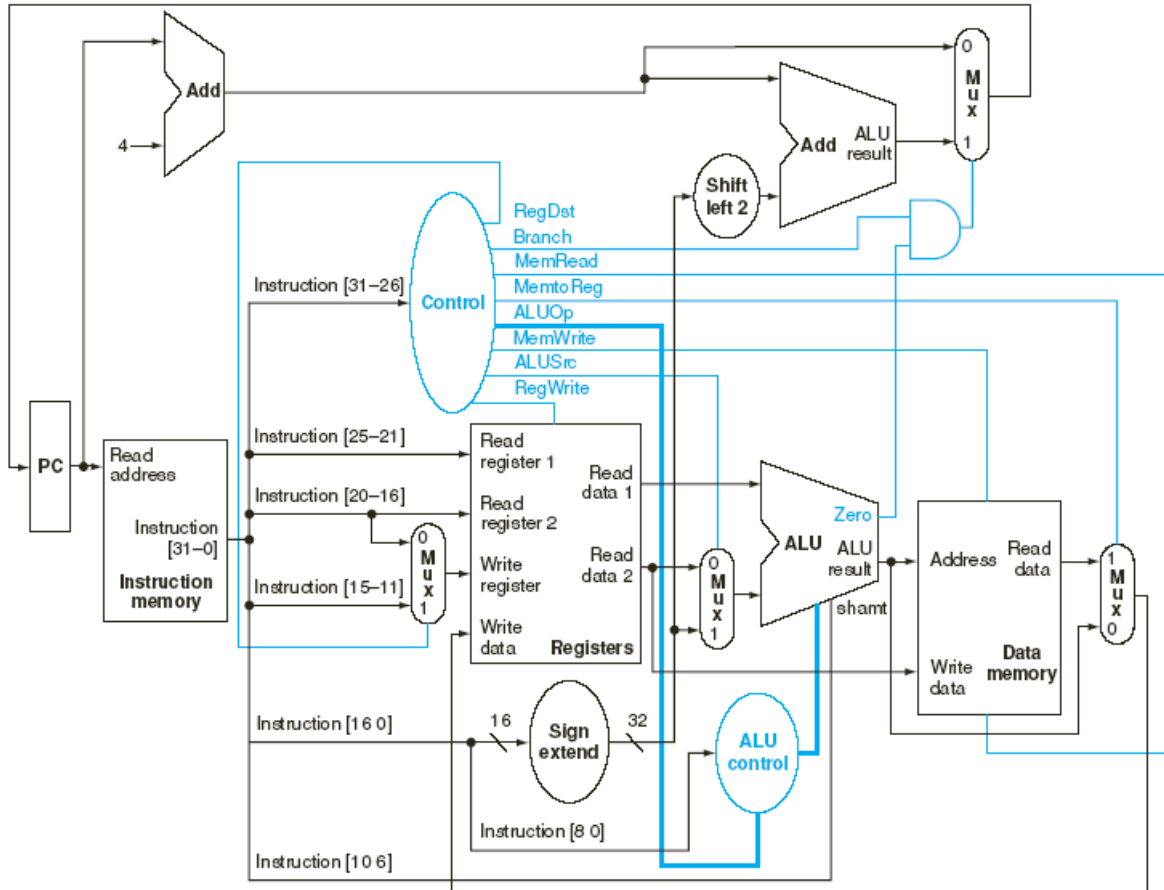
```
C:\Verilog>iverilog testbench2.v Q2.v
C:\Verilog>vvp a.out
      201 a = 2, b = 2, c = 0
      402 a = 1, b = 2, c = 1
C:\Verilog>
```

PTS: 1

**PROBLEM**

2. ANS:

(a) The updated single cycle datapath should look something like as shown below - “shamt” field (instruction [10:6]) from the instruction has been added directly as a control to the ALU to determine the shift amount.



(b) The

instruction is in R-format and is controlled according to the first line in the table below. No changes in the table are needed.

Instruction	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	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

**Alternative answer** - The ALU will identify the sll operation from the ALUOp field.

The ALU will identify the sll operation by the ALUOp field, which is not specified in our design (Fig. 4.13). Figure 4.13 on page 318 (4th ed) should be modified to recognize the opcode of sll: the third line should be changed to 1X1X0000 0010 (to discriminate the add and ssl functions), and a new line, inserted, for example, 1X0X0000 0011 (to define sll by the 0011 operation code).

PTS: 1