Department of Electrical and Computer Engineering

*University of Wisconsin – Madison*

ECE 552 Introductions to Computer Architecture

Homework #4 (Due 2/20)

*For the first two questions below (verilog based), paste your entire verilog code in a single text/doc file and submit the text/doc file on canvas. Note that these verilog questions need to follow the verilog rules specified in the 552 project/verilog rules document that can be found on Canvas.*

1. ***Parallel Sub-word Add***

In this exercise you will design a 16-bit parallel sub-word addition instruction: PSA.

Its operation is described below:

The PSA instruction performs four half-byte signed additions in parallel to realize sub-word parallelism. Specifically, each of the four half bytes (4-bits) will be treated as separate numbers, but stored in whole as a single vector. When PSA is performed, the four numbers will be added separately. To be more specific, let the contents in rs and rt are aaaa\_bbbb\_cccc\_dddd, eeee\_ffff\_gggg\_hhhh respectively where a, b, c, d, e, f, g, and h ∈ {0, 1}. Then after execution of PSA, the content of rd will be {(aaaa+eeee), (bbbb+ffff), (cccc+gggg), (dddd+hhhh)}. You will need to check for overflow in each of the additions. If any of the sub-word additions result in an overflow, the PSA error flag should be set.

Submit the verilog code and a test bench. The module template is provided below.

module PSA\_16bit (Sum, Error, A, B);

input [15:0] A, B; //Input values

output [15:0] Sum; //sum output

output Error; //To indicate overflows

…...

…...

endmodule

**2. Shifter**

Design a combinational 16-bit barrel shifter to implement the following two shift instructions: SLL (Shift Left Logical), SRA (Shift Right Arithmetic). The 4-bit shift amount is an unsigned binary number.

You are required to provide a verilog design and testbench. The design interface to use is:

module Shifter (Shift\_Out, Shift\_In, Shift\_Val, Mode);

input [15:0] Shift\_In; //This is the number to perform shift operation on

input [3:0] Shift\_Val; //Shift amount (used to shift the ‘Shift\_In’)

input Mode; // To indicate SLL or SRA

output [15:0] Shift\_Out; //Shifter value

…….

…...

endmodule

*The following two questions are non-verilog. Submit this as a text doc or a snapshot.*

3. **Data Path / Control - 1**

Refer Fig. 4.4.11 (COD 4.24)

Problems in this exercise refer to a clock cycle in which the processor fetches the following instruction word:   
**10101100011000100000000000010100**

Assume that data memory is all zeros and that the processor's registers have the following values at the beginning of the cycle in which the above instruction word is fetched:

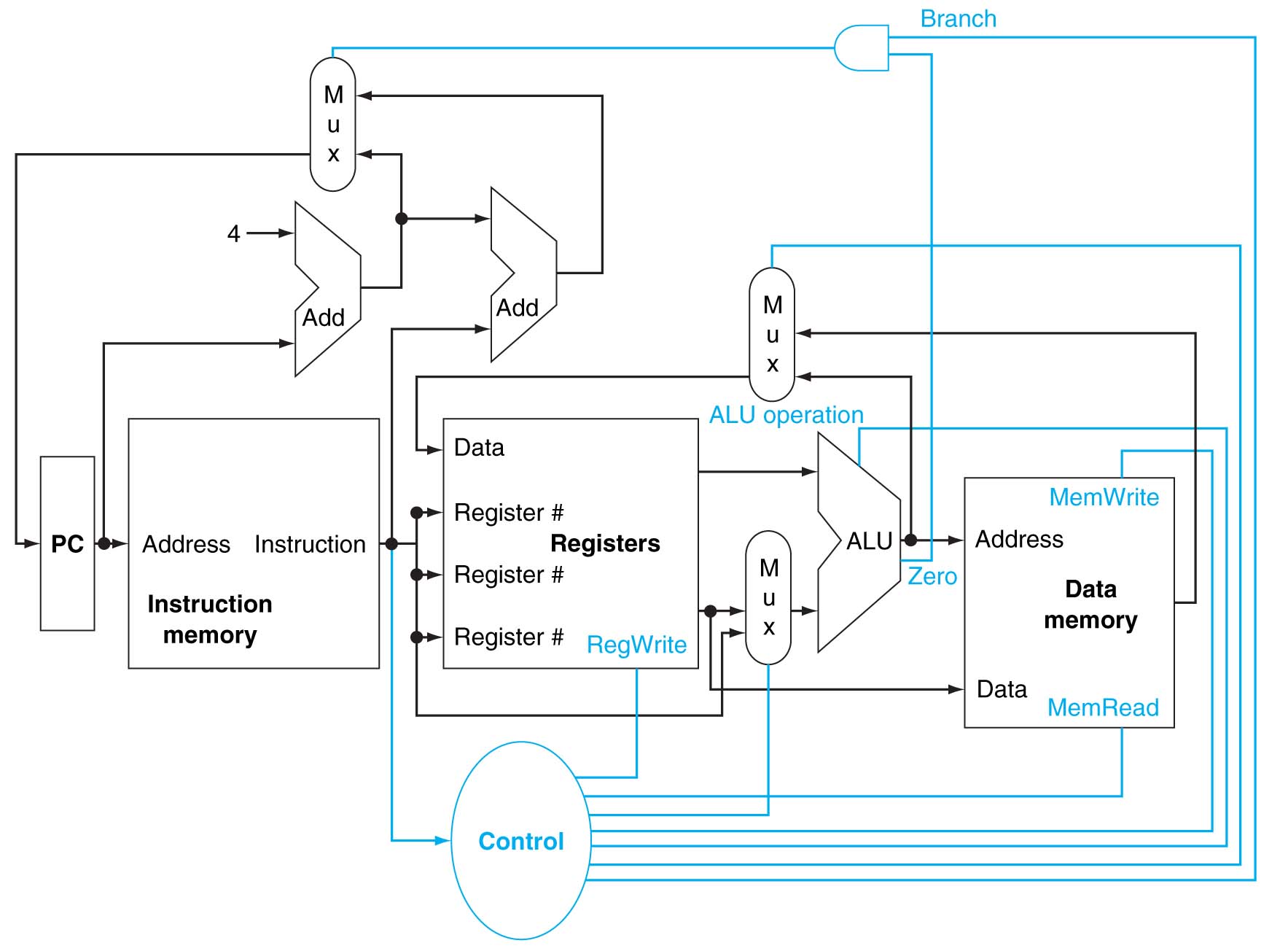
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| r0 | r1 | r2 | r3 | r4 | r5 | r6 | r8 | r12 | r31 |
| 0 | -1 | 2 | -3 | -4 | 10 | 6 | 8 | 2 | -16 |

1. What are the outputs of the sign-extend and the jump "Shift left 2" unit (near the top of COD Figure 4.24 (The simple control and datapath are extended to handle the jump instruction)) for this instruction word?
2. What are the values of the ALU control unit's inputs for this instruction?
3. What is the new PC address after this instruction is executed? Highlight the path through which this value is determined.
4. For each Mux, show the values of its data output during the execution of this instruction and these register values.
5. For the ALU and the two add units, what are their data input values?

What are the values of all inputs for the "Registers" unit?

4. **Data path / Control - 2**

Below is a simplified single-cycle MIPS data path and corresponding control signals.



0

1

1

0

0

1

If the latencies of the following blocks are: Inst. Mem (750ps), ADD (200 ps), MUX (50ps), ALU (250 ps), Registers (300 ps), Data Mem (500 ps), control (300 ps), AND gate (5 ps). What is the critical path and its total delay for evaluating the MIPS BEQ instruction?