Due Date: April 9, 2025

(Worth about 30% of the Project)

Goal: You need to implement Register unit, MUX, Sign-Extension and ALU. Integrate this design with the first phase. Finally, Write a test bench to test the complete design.

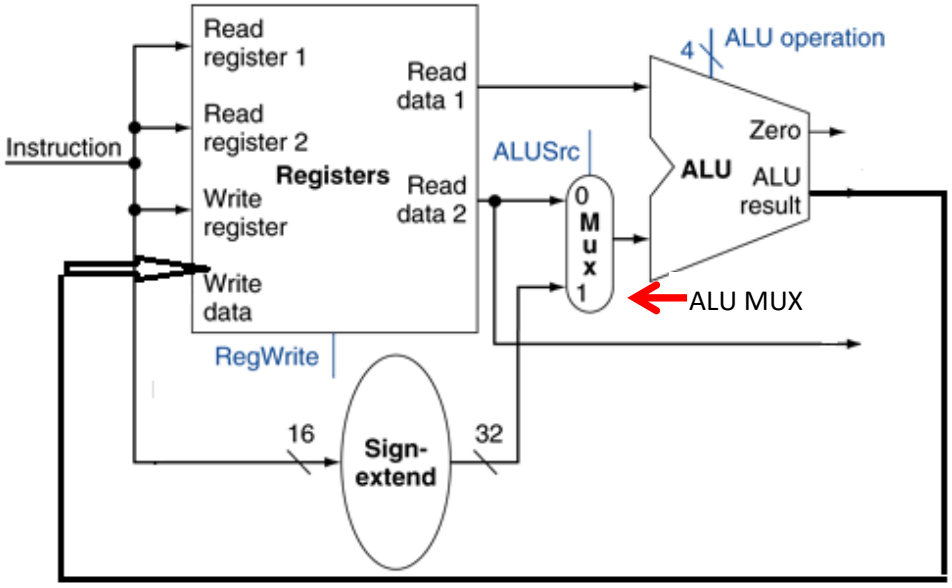

Figure 1.

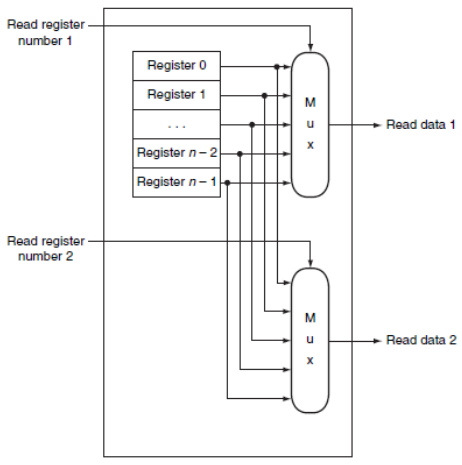

Figure 2. Read Unit

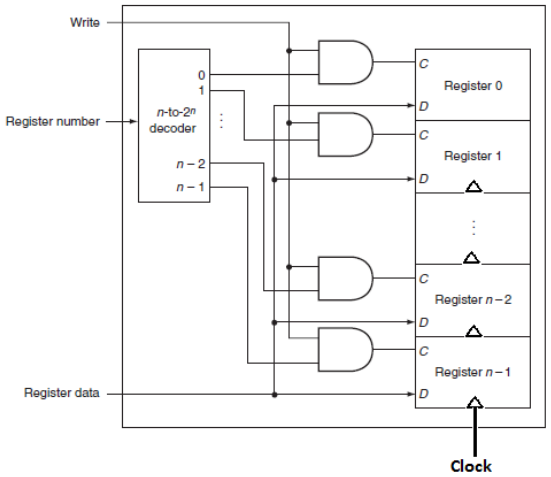

Figure 3. Write Unit

High level Description of each unit:

Registers: It should have 5 bits read/write register inputs (compliant to MIPS), a 32 bit Write data port and two **32 bit Read data outputs**.

The block diagram of ***read unit*** of register is provided in figure 2. This unit shows 32 registers of 32 bits each (Register 0 to Register (n-1), with n = 32). The select signal to the mux is the operand from the instruction memory (which you implemented in phase 1 of the project). This signal is same as Read register 1 and Read register 2 in Figure 1.

The block diagram of the ***write unit*** of the register file is provided in Figure 3. Write unit has four input signals. Clock, Write (which is same as RegWrite in Figure 1), Register Number, which is same as input connected to the write register in Figure1 and Register data, which needs to be connected to the Write data port of Figure 1. Also notice that each Register has three inputs, C, D and Clock. The C input acts as an enabling input of the register (enable the write port of the specific register), and write at the positive edge of the next Clock signal. The clock signal is the master clock at which the register writes the data. D is 32-bit input data coming from the data memory (Register Data). And n to 2^n decoder shows 5 to 32 decoder.

ALU: This should be a combinational block. Must have two 32-bit inputs (operands), 4 bit input control signals (ALU operation), 32 bit output and the result port and one bit output at zero port.

SignExtend: It replicates the 16^{th} bit 16 times to make a 32-bit input to ALU-Mux. If required you may use ieee.numeric_std.all library. The input and output of the sign extension unit should be std_logic_vector

ALU-Mux: Two 32-bit inputs (std_logic_vector) and one 32-bit output (std_logic_vector) with select signal, which will eventually come from control unit (but not in this phase of the project)

Constraints:

1. You'll be using DE10-Lite board.
2. Cannot use megafunctions or IPs for ANY block shown in Figures 1, 2, 3. **Integration of previous phase is a requirement,** and IM in phase 1 is using IP.
3. Refer to slide #s 6 and 7 in the slide set ECE4120_Chapter4_2_S_25, to implement ALU operations. ALU is expected to perform all the operations except for set-on-less-than. You use the same ALU_control value that is assigned for set-on-less-than for XOR instruction instead. Hence your ALU should be performing following six operations: AND, OR, Add, subtract, XOR, and NOR.
4. Register file unit should be implemented as shown in the Figure 2 and Figure 3 (distinctly provide the codes for decoder, multiplexer and registers).
5. For testing purposes, follow the test bench requirements provided below.
6. For testing on the board, follow the board demonstration provided below.

Test Bench Requirements:

1. Test bench entity should be empty.
2. Test bench should name the design under test as DUT.
3. First load the values to the registers, (while making sure that t3, and t4 s1 and s2 are getting the same values) with non-zero values. Make sure that you know the register numbers, which are provided in your book.
4. Generate proper control signals and perform execution of following set of MIPS instructions:
    a. add $t3, $t0, $t1
    b. sub $t4, $s0, $s1
    c. and $ t5, $t3, $t4
    d. xor $t0, $s3, $s4 – see constraint 3 for further explanation
    e. Implement not operation using NOR instruction (elaborate your working for full credit)
    f. addi $s3, $s3, 4

Testing on the Board:

1. Use a switch to select your instruction.
2. Six 7-segment display should show the hex values of all the three registers in the chosen instruction (i.e. two 7-segment units for each register value). Make sure none of your registers are holding value greater than FF_{hex} (e.g. make sure addition in instruction 1 will not result in a value greater than FF_{hex}).

Deliverables:

I. A pdf file containing following (please provide hardcopy):

1. Modified block diagram, showing the bits and additional connections (**no hand drawings and or copy and paste of any figure from this document**)
2. What are the objectives of this phase
3. Elaboration on VHDL implementation of each unit
4. Flow summary, RTL view and Technology map view (using DE 10-Lite board)
5. **Elaboration on test bench:** e.g. *how you loaded the registers*, and how you confirmed the execution of each of these instructions – (you are not allowed to use data memory block to load the registers).
6. **Elaboration on the waveforms** and snapshot of the waveforms should also be included. Show using separate waveforms to elaborate on **loading** the registers, and **writing** the result for **each** instruction. Elaboration should state whether the waveform show the successful implementation or not, with explanation on how you verified it.
7. Conclusions

Submission:

1. **A zipped VHDL Implementation of the Project**: Provide all the VHDL files and files for IP generated by Quartus, so that I can include it in the project and make sure it is running. Also add a readme.txt file explaining what each file is doing (e.g. which one is testbench, or IP, or regular .vhd file).
2. Also include a separate **readme** on the added files needed to test it on the board. Provide ***explanation on your choices of switches*** as well, and which switch/button is for resetting.

Name the submitted folder as your lastname1_lastname2_S25_phase2.