Herbert Wertheim College of Engineering
Department of Electrical & Computer Engineering
UNIVERSITY of FLORIDA

Dr. Christophe Bobda
Omar Elbasri
EEL3701C - Fall 2025

Lab 6: CPU Design

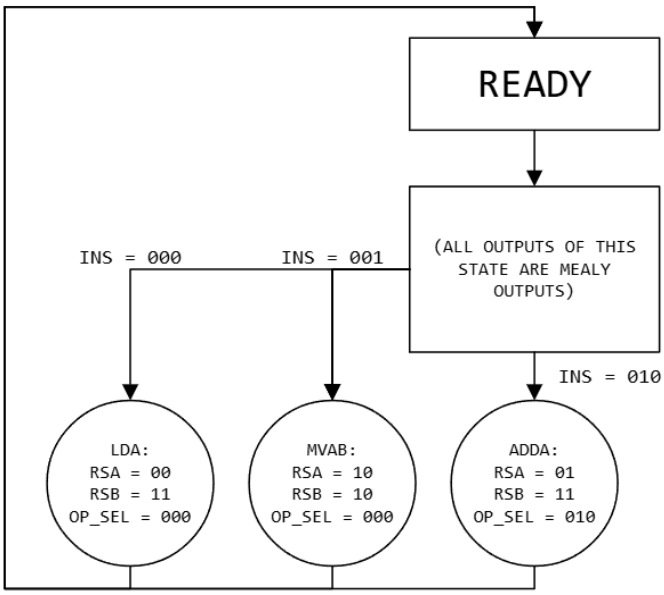
Overview

While an ALU is the heart of the CPU, the CPU is the heart of the computer. The CPU executes instructions and is responsible for most of the operations that your computer performs. In Part 1 of this lab, a controller will be designed that allows for better control of the Registered ALU from Lab 4 by controlling its inputs. In Part 2 of this lab, you will connect your controller to the ALU and add an instruction register to store incoming instructions. In Part 3 of this lab, you will use the DE10-Lite board to run programs on your CPU.

If you do not have a working Lab 4 RALU, one is provided at this link: https://eel3701.ece.ufl.edu/assets/misc/lab6_ralu.vhd. As done with previous VHDL files, produce a symbol file for use in upcoming parts.

Pre-Lab Procedure: Part 1 - The Controller

In this part, a controller will be created for the Registered ALU. A controller is a state machine that takes in an instruction and outputs the corresponding control signals for the RALU. A flowchart with some of the instructions omitted is below. You will have to submit a hand-drawn or computer generated flowchart with the additional 5 instructions that are not displayed (see Table 1).



Herbert Wertheim College of Engineering
Department of Electrical & Computer Engineering
UNIVERSITY of FLORIDA

Dr. Christophe Bobda
Omar Elbasri
EEL3701C - Fall 2025

Figure 1: Incomplete Controller Flowchart

There are two main states: the READY state, and the EXECUTE state. In the READY state, the controller is indicating it is ready to receive an instruction, and the READY output should be true. In this state, the controller should output the appropriate signals to ensure the values of the registers don't change. Then, based on the instruction, the controller will output appropriate select lines to the RALU.

INSTRUCTION	CODE	DESCRIPTION
LDA	000	Loads REG A with input bus.
MVAB	001	Loads REG B with REG A.
ADDA	010	A ← A + B
ORA	011	A ← A OR B
NOTA	100	A ← NOT A
SRA	101	A ← A shifted right 1 bit
NANDA	110	A ← NOT (A AND B)
NOP	111	Preserve registers.

Table 1: Supported CPU Operations

For example, for LDA, RSA = 00 so REGA will be loaded with the input bus, RSB = 11 so whatever is in REGB will be preserved, and OP_SEL = 000 because it isn't being used for anything.

Similarly, for MVAB, RSA = 10 to preserve REGA, RSB = 10 to get the contents of REGA, and OP_SEL = 000 because it isn't used for anything.

Finally, for ADDA, RSA = 01 to use the output of the ALU for REGA, RSB = 11 to preserve REGB, and OP_SEL = 010 for addition.

The table of all instructions and their explanations is above, and the OP_SEL and RSA/RSB tables from Lab 4 are located in Appendix A. The outputs for the other instructions are to be done by you. For NANDA, which requires two operations, consider how multiple operations could be performed in sequence.

Unlike previous labs, any VHDL structure is allowed when making the controller. Process blocks and if statements are allowed, as well as external flip-flops and combinational logic in VHDL. The controller as a whole should have inputs of CLK, INS[2:0], RST for the flip-flops, and outputs of READY, RSA[1:0], RSB[1:0], and OP_SEL[2:0] for use with the controller.

Create a simulation **testing every single instruction**, with annotations showing the outputs are as expected. The simulation should show all the inputs and outputs of the controller.

Herbert Wertheim College of Engineering
Department of Electrical & Computer Engineering
UNIVERSITY of FLORIDA

Dr. Christophe Bobda
Omar Elbasri
EEL3701C - Fall 2025

Part 1 Summary:

1. Submit the completed flowchart by adding in the 5 missing instructions and their RSA, RSB, and OP_SEL outputs.
2. Design a controller in Quartus using your flowchart.
3. Simulate every instruction in your controller, then annotate the simulation to show that the RSA, RSB, and OP_SEL outputs are correct.
4. Include images or text of your design, taking a screenshot of either your .bdf or .vhd file.
5. After thoroughly testing your controller, make it into a .bsf file as done in previous labs for use in the upcoming parts.

Pre-Lab Procedure: Part 2 - Instruction Register and Connections

In this part, you will create an instruction register to hold the instruction currently being executed, and connect the controller to your RALU to execute instructions. The instruction register will be made of 3 flip-flops with 2-to-1 muxes on their inputs. The muxes will have two inputs: the output of the flip-flop, and the new value to load into the flip-flop. Depending on if we want a new value to be loaded into the flip-flop or not, we can change the input of the mux's select line. New values should only be loaded in when the controller is not executing an instruction, so consider what signal could be used for this purpose.

You will then connect the output of the instruction register to the controller, and the output of the controller to the ALU. Then create a simulation of at least 4 different instructions in sequence, showing that it correctly modifies the values within A and B. The simulation should show at minimum CLK, INS[2:0], READY, and the outputs of REGA and REGB.

Part 2 Summary:

1. Create a block diagram for the instruction register, showing the 3 flip-flops, their corresponding muxes, and the names of signals going in and out.
2. Design an instruction register in Quartus using a .bdf or .vhd file, then connect the instruction register, controller, and ALU together in a file named lab6_cpu.bdf.
3. Simulate at least 4 different instructions in sequence, and submit annotated screenshots.
4. Submit screenshots of your design lab6_cpu.bdf design, being sure to show the connections between the controller, instruction register, and ALU.

UF Herbert Wertheim College of Engineering
Department of Electrical & Computer Engineering
UNIVERSITY of FLORIDA

Dr. Christophe Bobda
Omar Elbasri
EEL3701C - Fall 2025

Pre-Lab Procedure: Part 3 - Programming

In this final part of the lab, you will be creating a small assembly program to run on your CPU. It will be done with the same method as Lab 4, using the switches and buttons of the DE10 to send the input bus and instruction to the design from Part 2. Take into account instructions that may require multiple clock cycles, and toggle the clock multiple times accordingly.

The program to implement will calculate the following equation: **REGA = 7 + (3 AND (5 OR C)).** Consider what operations must be done with combinations of multiple instructions. Fill out a table with the same structure as Table 2 for the values of inputs as the program executes. The values of REGA and REGB should be after the instruction executes (right after the rising clock edge), and the first two rows are filled out as an example.

Instruction	**Input Bus**	**REGA**	**REGB**
LDA (000)	0101	5	?
MVAB (001)	0000	5	5

Table 2: Example table structure for program

After filling out the table, run the sequence on the DE10 Lite and ensure it functions properly. Ensure the clock speed is slow enough to be able to see every step individually. You will demonstrate the sequence with the proper end result to your PI in-lab. The values of REGA and REGB are required to appear on two seven-segment displays. Any other information you find helpful in debugging, such as the value within the instruction register is optional but recommended.

Part 3 Summary:

1. Create an assembly program for the equation REGA = 7 + (3 AND (5 OR C)), then fill out the corresponding table with the values of REGA and REGB after every instruction.
2. Make sure the sequence runs on the DE10 board after finishing the program.

When you are done with the lab, archive the Quartus project and submit the qar file, along with the pre-lab report, on GitHub Classroom. The qar file submitted on Github Classroom is your final submission, and it cannot be changed after the deadline. This file must contain a working design that is ready to demo. **“Ready to demo” implies that you have already run your design on your DE10-Lite and have assigned all the pins. You will be given no extra time in-lab to assign pins or otherwise change the project file.**

UF Herbert Wertheim College of Engineering
Department of Electrical & Computer Engineering
UNIVERSITY of FLORIDA

Dr. Christophe Bobda
Omar Elbasri
EEL3701C - Fall 2025

Pre-Lab Questions

1. What is the advantage of making the controller a state machine, rather than combinational logic?
2. Write a small program to calculate the following equation: REGB = 6 + (5 NOR (E / 2)). Use the same table format as Table 2, and consider what instruction could be used for division. This sequence is not required to be simulated or shown on the DE10 board.

In-Lab Procedure

1. Demonstrate the program from Part 3 to your PI, ensuring the final values are correct.
2. Complete the in-lab quiz as specified by your PI.

UF Herbert Wertheim College of Engineering
Department of Electrical & Computer Engineering
UNIVERSITY of FLORIDA

Dr. Christophe Bobda
Omar Elbasri
EEL3701C - Fall 2025

Appendix A

Summary of Lab 4 RALU Tables:

RSA, RSB	Data Source
00	Input Bus
01	Output Bus (ALU Out)
10	REGA Output
11	REGB Output

Operation	OP_SEL
Bitwise AND of A and B	000
Bitwise OR of A and B	001
Sum of A and B with carry	010
Bitwise Negation of A	011
A left shifted by 1 bit	100
A right shifted by 1 bit	101
Output A	110
Output B	111