Laboratory 6

(Due date: **002**: April 11th, **003**: April 12th)

OBJECTIVES

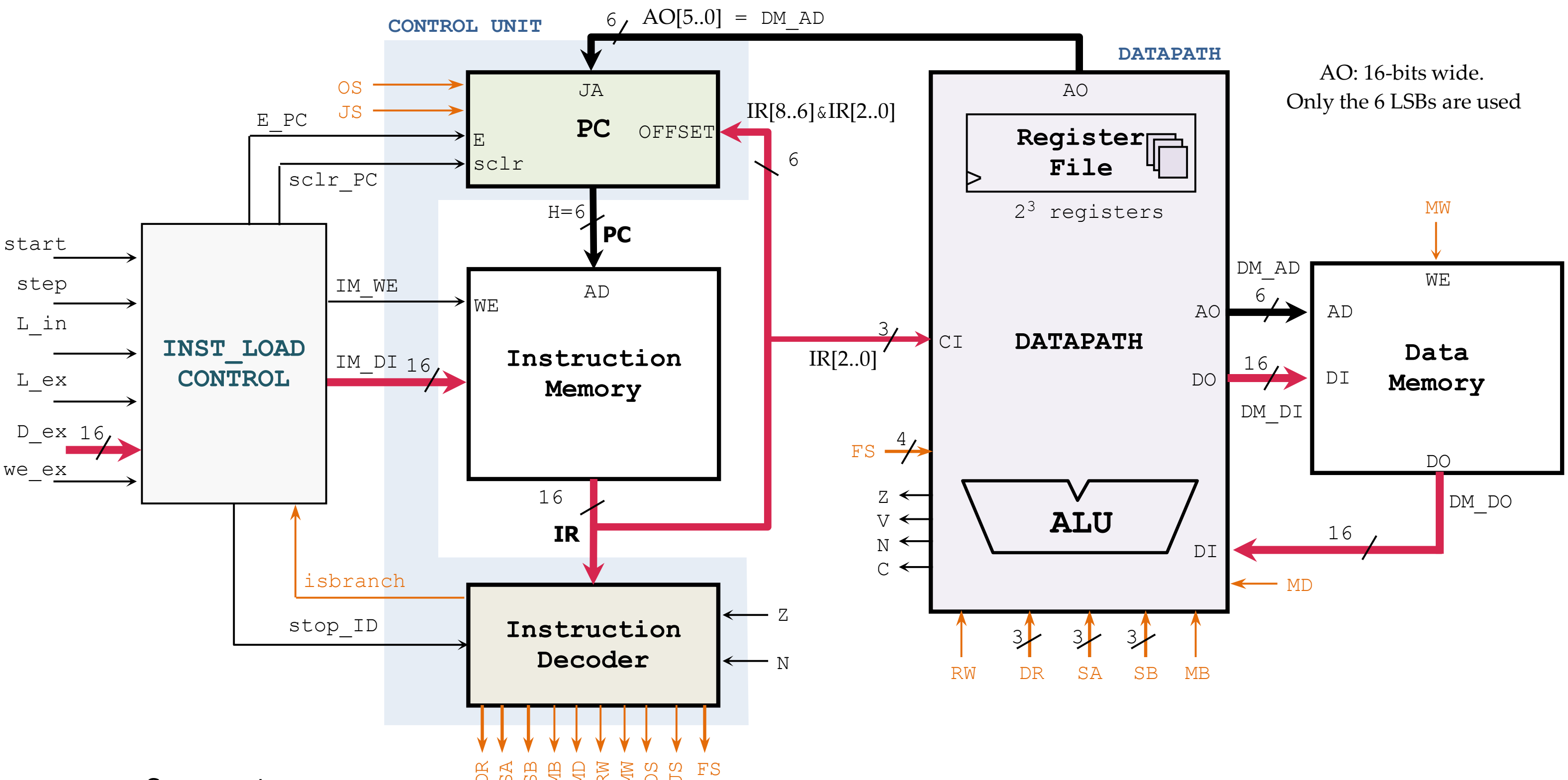
- ✓ Design a 16-bit microprocessor with Single-Cycle Hardwired Control.
- ✓ Implement an Instruction Set.

VHDL CODING

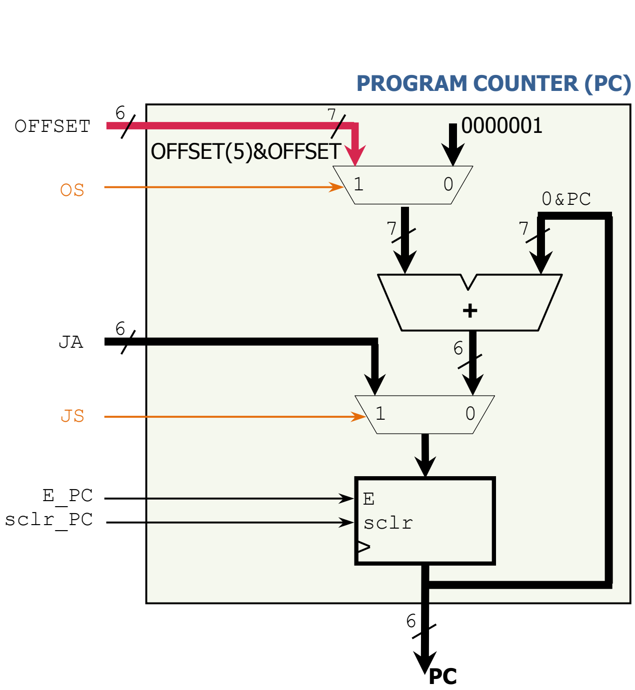
- ✓ Refer to the Tutorial: VHDL for FPGAs for a tutorial and a list of examples.

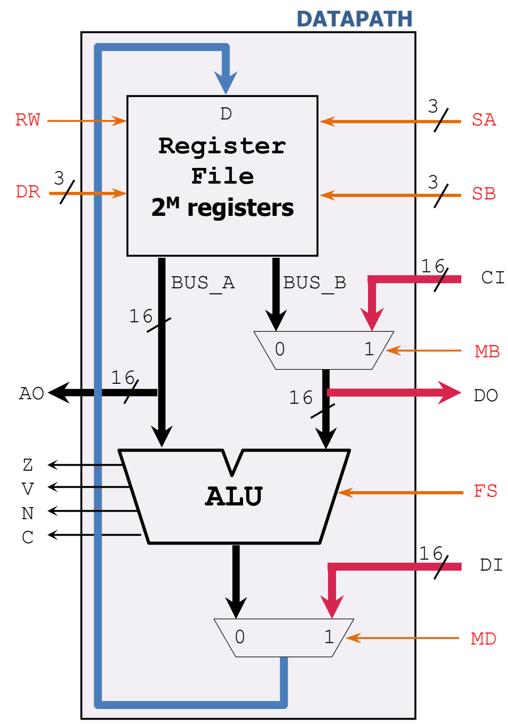
FIRST ACTIVITY: 16-BIT MICROPROCESSOR DESIGN AND SIMULATION (100/100)

- Implement the **Simple Computer** (see Notes – Unit 6): uP with 6-bit IM/DM address, 16-bit instructions, and 16-bit data.



- Components:
  - ✓ DM, IM: 64 words, 16 bits per word. Use the files RAM_emul.vhd, my_rege.vhd. (set the proper parameters).
  - ✓ Datapath:
    - Register File: 8 registers (R0 – R7) are included. See Notes – Unit 6 for an example with 4 registers.
    - ALU: Use the files: alu.vhd, alu_arith.vhd, alu_logic.vhd, super_addsub.vhd, fulladd.vhd.
  - ✓ PC: Note that OFFSET is a 6-bit signed number. The adder uses 7 bits, from which we only retrieve the 6 LBSs.
  - ✓ Instruction Decoder (ID): This is a large combinational circuit. The outputs depend directly on the inputs.
    - The outputs are generated based on the instructions on IR (Instruction Register).
    - Instruction Set: For the list of instructions, refer to Notes – Unit 6. The Instruction Set does not include instructions that read the V and C bits. Thus, the ID does not consider these two bits.
    - stop_ID: This input signal causes all the ID outputs to be '0' if stop_ID=1.
    - isbranch: If the instruction in IR is a branch of jump instruction, this signal is set to '1'.
  - ✓ Instruction Load Control: This component is required in order to write instructions on the IM, and then to trigger program execution. Use the file instload_ctrl.vhd. This circuit is a FSM that works as follows:
    - To store instructions on IM from an external port, assert L_ex and then use the inputs D_ex and we_ex.
    - To store instructions on IM using pre-stored hardwired data, assert L_in.
    - Once instructions are written on the IM, program execution is started by asserting start for a clock cycle. The step signal controls whether to enable program execution (step=1) or disable it (step=0).





SIMULATION

- We will execute the following pre-stored program (counter from 2 to 13): (see instload_ctrl.vhd). Note that the count appears on R0.

Address	Assembly Program	VHDL code snippet
000000	start: LDI R3,4	CD (0) <= "1001100011---100";
000001	LDI R0,2	CD (1) <= "1001100000---010";
000010	LDI R1,7	CD (2) <= "1001100001---111";
000011	ADI R1,R1,4	CD (3) <= "1000010001001100";
000100	loop: ADI R0,R0,1	CD (4) <= "1000010000000001";
000101	DEC R1,R1	CD (5) <= "0000110001001---";
000110	BRZ R1,-5	CD (6) <= "1100000111001011";
000111	JMP R3	CD (7) <= "1110000---011---";
001000		…
...		

- Tesbench:
  - ✓ Set L_in=1 for a clock cycle. Then wait about 70 cycles for the program to be written on the Instruction Memory.
  - ✓ Set start=1 for a clock cycle. Make sure that step = 1 during the execution of the program.
    - ▫ Circuit verification: To see if the instructions are processed in the right order, take a look at PC and IR. Then, to check the outputs, observe the R0-R7 values and then focus on other signals such as the ID outputs.

- Design Flow and verification:
  - ✓ Write the VHDL for the given circuit. Synthesize your circuit. (Run Synthesis).
  - ✓ Perform Functional Simulation (Run Simulation → Run Behavioral Simulation). **Demonstrate this to your TA.**

- Submit (as a .zip file) the generated files: VHDL code, and VHDL testbench, and XDC file to Moodle (an assignment will be created). DO NOT submit the whole Vivado Project.

You can work in teams of up to two (2) students. Only one Moodle submission per team.

TA signature: ______________________________ **Date:** ___________________________