Phase 1 of the Project – ECE 4120/5120 (Spring 2025)

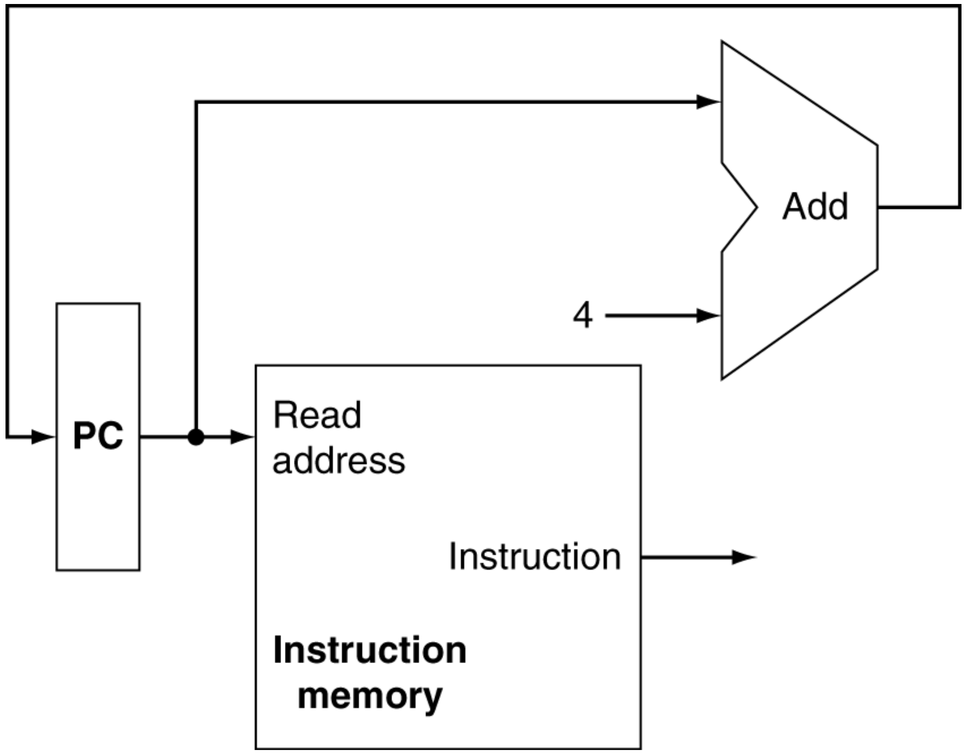
Posted: 02/24/25 Due Date: March 14th, 2025

(Worth 20 % of the Project)

This is a Group Project – maximum of two students per group are allowed. Inform me about your group by next class (x/x/24)

Goal: You need to implement instruction fetch unit of the MIPS processor. The block diagram is provided below, your code needs to follow the given description and constrains:

PC and Instruction Memory both are getting the same clock signal



High level Description of each unit:

PC: It should be synchronous block. It should have **32 bits input** and **32 bit output**.

Instruction Memory: This should be a synchronous block (PC and Instruction Memory should have the same clock signal). Read address should be 8 **bits** wide (8 least significant bits from the PC output) and the instruction output should be **32 bits.**

Add block: It is a purely combinational block. It is a 32 bit adder, with one of the input is a constant number '4' (you may need to modify this number based on your instruction memory constraint, if so then you need to provide the justification).

Constraints:

Phase 1 of the Project – ECE 4120/5120 (Spring 2025)

1. Cannot use Megafunction or IP blocks for any block other than the instruction memory
2. MIPS Instructions need to be stored through **test bench or .mif file**, in the first 5 locations of instruction memory.
3. Read address of the instruction memory should be connected to 8 least significant bits of PC.
4. Top module should be used to port map all the components.
5. Your testbed entity should be empty.
6. ***Special attention should be given to make sure your design uses least number of output pins.***
7. Your simulation results should include following signals: input and output of the PC, constant input to the adder, read address to the instruction memory, wren of the memory (if using testbench to populate the instruction memory), input to the instruction memory (if supplying instructions using testbench), output of the instruction of the memory, and clock & reset signals.
8. Instruction memory should store five MIPS instructions in any combination, instructions should be (add, sub, srl, lw, sw) – chose your own "MIPS' legitimate" operands.

Deliverables:

I. A pdf file containing following:

1. Modified block diagram, showing the ***bits and additional connections*** (for example how are you connecting the input data to instruction memory, how are you connecting constant value to the input of the adder)
2. Elaboration on VHDL implementation – It is not the comments on the code, it is about how you developed the code. Include the rationale of design choices from digital hardware design perspective.
3. Use the same device/board for synthesis that you used in ECE 4110/3140 and show the snapshot of that device selection.
4. Flow summary, RTL view and Technology map view
5. **Elaboration on the test bench & .mif (if used)** (along with the testbench and line by line explanation): e.g. how many instructions are loaded (explain .mif file if used), how many are read, how you enabled the add block etc.
6. **Elaboration on the waveform** should include following:

   Both writing to the memory (if you are not using .mif) and reading from the memory should be clearly shown in the waveform – special emphasis on whether you are getting data out from the **correct address** location at the **right time.** So, you need to state whether the waveforms show the successful implementation or not. Explain what data you have written to the memory and what data you are expecting at the output and at what time instance. If using .mif file then show what is the value you provided to the particular address and whether you read back the same value or not (at the correct time instance).

7. Conclusion(s)

II. **A zipped VHDL Implementation of the Project**: A folder containing all the files generated by Quartus, including all the .vhd files, should be submitted to the drop box via ilearn. Name the submitted folder as lastname1_lastname2_S24.