Lab 3

Implement a pipelined RISCV processor.
We will start by implementing the datapath for only one instruction - addi.
5 stages, every stage takes 1 CC. This will allow us to handle the most basic hazards. No forwarding.
Entire processor works on positive edge of clock
Please add a Clock Cycle Counter to your processor that starts after reset and then increments by 1 every clock cycle. A 16 bit counter will be more than enough.
Register read is done during the ID stage.

You must print trace data in 2 files, this will help test as well debug.
In first file, print every PC being fetched in hexadecimal, 1 per line (we will need to update this when we do branches)
In a second file for data, you must print in every CC, the register number and the value being written for instructions that write to a register (either decimal or hexadecimal ok).
Later when we add Load/ Store instructions, we will print the memory address being read/ written and the data value for Load/ stores, but this is not required yet.

This must be done in 3 weeks. You are free to divide your time as you like, but by the end of week 3, we aim to have addi working with data hazards handled (stall on a hazard till value is available and then resume).
Initially, we will implement the pipeline w/o caring for the hazards and test it with the **addi_nohazard.dat.**
Then, we will update the pipeline to handle the hazards and test it with **addi_hazards.dat**

Fetch	1 CC
Decode	1 CC
Execute	1 CC
Memory access	1 CC
Write back	1 CC

Testbench, instruction memory and data memory files will be given to you. Asm files will also be given that you may use with rars to follow the code sequence and help in debug.
You only need to add the testbench, instruction memory, data memory and instruction and data memory initialization files to your project in addition to your processor.

The processor must use the following interface, name and ports.

Module name - cpu
Ports

Name	Direction	Width	Details
rst_n	Input	0:0	Active low reset
clk	Input	0:0	Clock
imem_addr	Output	31:0	Instruction memory address
imem_insn	Input	31:0	Instruction from Instruction memory
dmem_addr	Output	31:0	Data memory address
dmem_data	InOut	31:0	Data to/ from data memory
dmem_wen	Output	0:0	Write enable for data memory (1 for Store, 0 for Load)

Your cpu must **"synthesize"**. For implementation, we are using too many ports. We will later decide if we wish to put memories and cpu inside a top level design (+ disable synthesis for memories) to reduce ports (**<u>For now, you should have only your cpu in the design and all else in the testbench and set the cpu as design_top and synthesize your cpu only.</u>**)

Deliverables -
Demo - 50 marks
- (30 if pipeline works but hazards are not handled, 20 for data hazard handling)

Code submission - 50 marks
- (30 if pipeline works but hazards are not handled, 20 for data hazard handling)