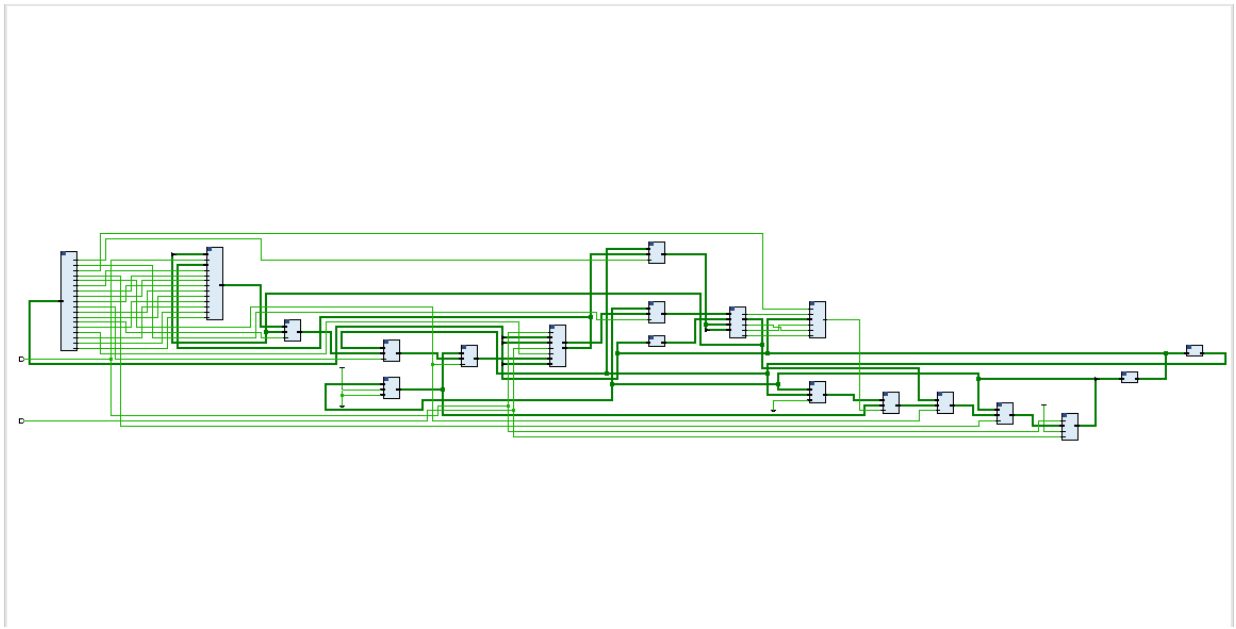
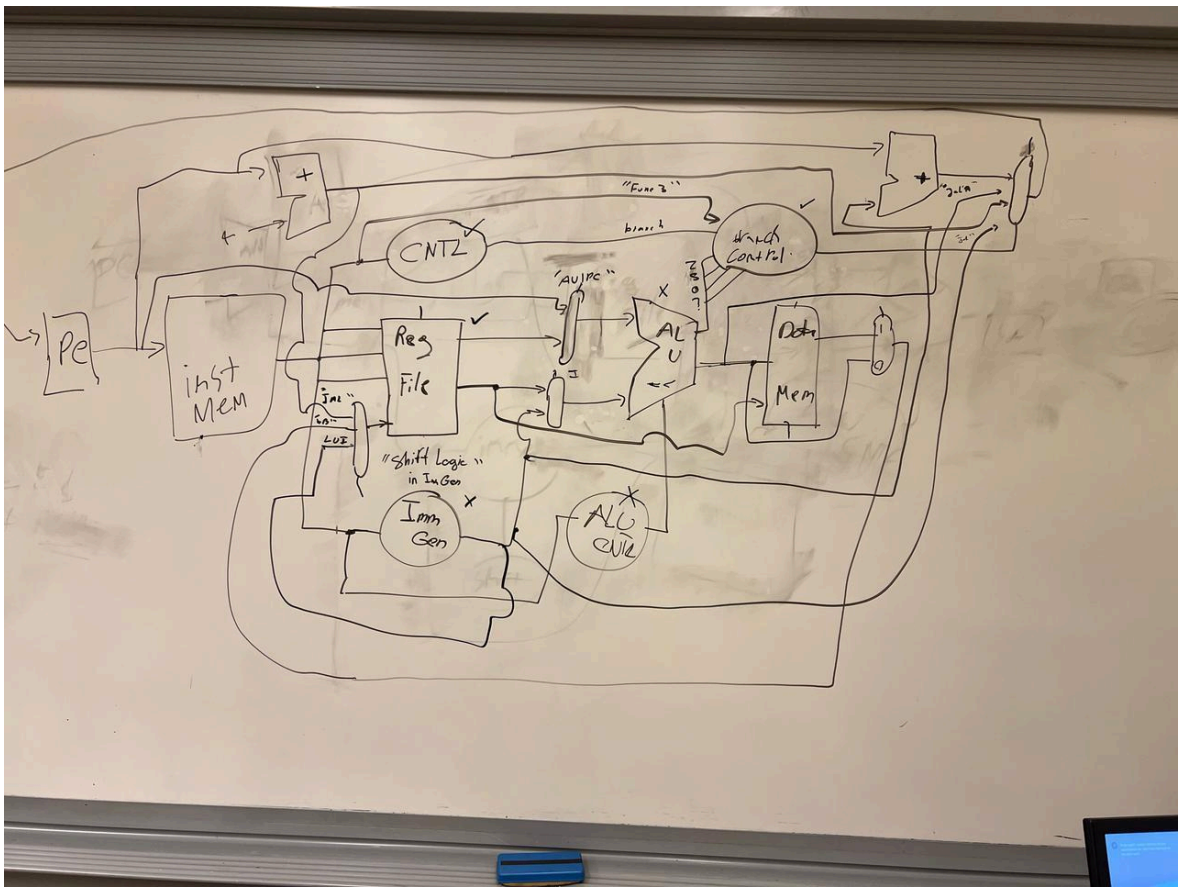
# Project Milestone 2 Report

## Design





**Implementation**

We implemented the schematic as the one attached exactly, the only difference was that instead of using a single multiplexor we used three multiplexors to check for the program counter and 4 multiplexors to check the write data that would be written to the register file.

**Modules**

Instruction_memory: Stores the instruction set for the processor. Given a program counter value, it outputs the corresponding 32-bit instruction.

rv32_ImmGen: Extracts and sign-extends immediate values from the instruction, supporting all RISC-V formats (I, S, B, U, J).

REG_file: Implements the RISC-V register file with 32 general-purpose registers. It allows reading two registers and writing to one per cycle.

Control_unit: Decodes the instruction opcode and function bits to generate control signals for ALU, memory, branching, halting, branching and register file operations

ALU_control: Generates a specific ALU operation code based on the instruction's funct3, funct7, and the general ALUop signal.

prv32_ALU: Performs arithmetic and logical operations like addition, subtraction, AND, OR, shifts, etc. Also sets status flags (zero, carry, sign, overflow).

branchCTRL: Evaluates whether a branch should be taken based on instruction type and ALU flags (zero, sign, carry, overflow).

Data_memory: Implements data RAM for load/store instructions. Supports byte, half-word, and word operations with sign or zero extension.

RCA: A 32-bit Ripple Carry Adder used for address calculations like PC + 4 or PC + immediate for branching and jumping.

**Issues Encountered**

- Typos in the code caused PC_out to not be updated correctly
    - Typos Fixed.
- ALU_control was outputting wrong ALU selects which caused sub operations to add
    - Fixed ALU select output from the ALU_control to match that in the ALU.
- Instruction memory encoding of test cases contained errors.
    - Added underscored to separate part of the instructions to prevent errors.
- Shifting instructions in r type were incorrect because the input Shamt to the LAU was incorrect.
    - Changed it to the output of the multiplexor between the rs2 and the immediate.

**Screenshots of test cases attached in the submission github repository**

We used multiple programs to test different instructions and we separated these programs in the github repository, we checked that each instruction outputted the correct values whether it was a register value or a value in the memory or a jump and a branch. We concluded that all the results are correct and the screenshots are attached.