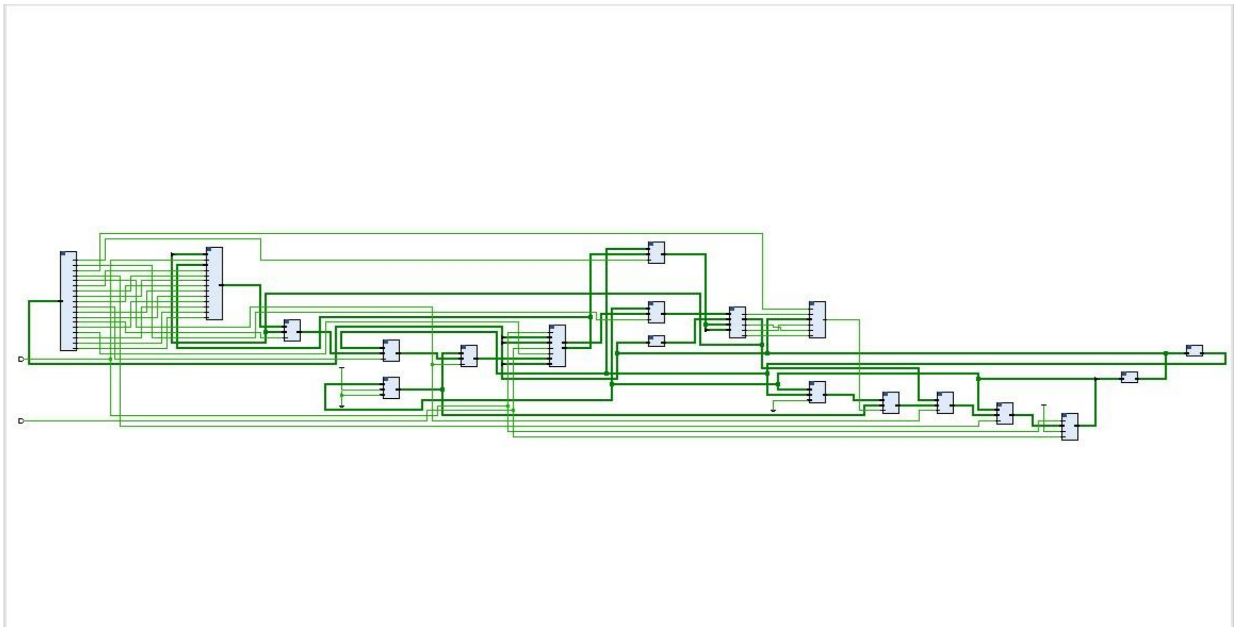
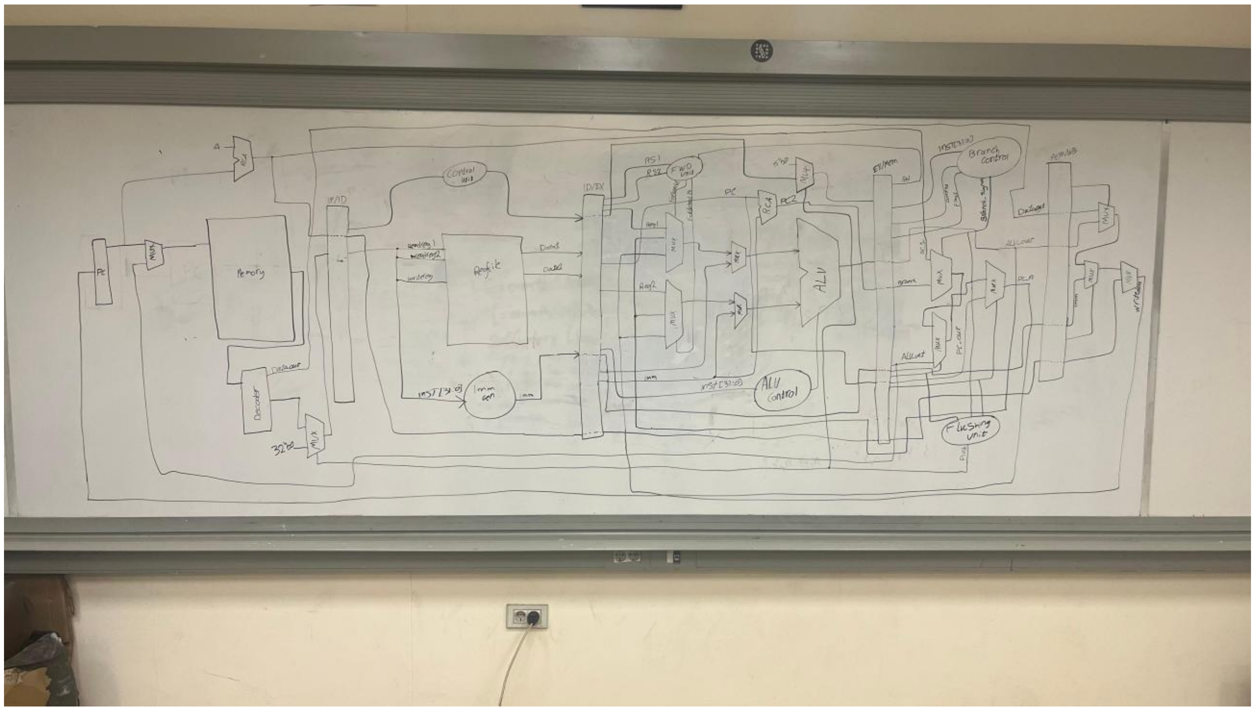
Project Milestone 2 Report

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

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).

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

Data_memory: Implements data RAM for load/store instructions. Supports byte, half-word, and word operations with sign or zero extension. This memory also acts as our instruction memory and it contains our instructions, this is a byte addressable memory.

Forwarding unit: this unit takes inputs from the rs1 and rs2 in the decoded instruction with the rd and the reg write of the previous two instructions and based on the conditions it either forwards or not.

Flushing unit: it takes the branch, jump, and halt signals and checks the need for flushing and creates a new signal called flush. This signal is used to flush the unnecessary instructions.

Clock divider: divides the clock by two.

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 underscores to separate part of the instructions to prevent errors.
- Shifting instructions in r type were incorrect because the input Shamt to the ALU was incorrect.
    - Changed it to the output of the multiplexor between the rs2 and the immediate.
- The memory was not working properly, and this caused the whole program to not work.
    - After removing the initialization that we had included and adding a default case the memory worked correctly

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. our reference was the outputs from the screenshots of milestone 2 as these are the outputs of the single cycle implementation and we compared the new results to these, and the results were identical to the old ones.