Milestone 2

Computer architecture

Youssef attalah	900221247
Youssef Nader	900226170

1. Introduction

This project involves the implementation of a single-cycle processor that supports all 42 base instructions from the RV32I RISC-V Instruction Set Architecture. The processor is designed and implemented using Verilog HDL and tested using waveform simulations.

The processor supports all 42 instructions categorized under R, I, S, B, U, and J types. These include arithmetic, logical, memory access, control transfer, and immediate instructions.

3. Proposed Datapath Design

The datapath is designed to execute all supported RV32I instructions within a single clock cycle, ensuring high-speed operation at the cost of hardware complexity and resource usage.

The datapath consists of the following major components:

- **Program Counter (PC):** Holds the address of the current instruction. After each instruction fetch, it is updated either sequentially (PC + 4) or by a branch/jump target.
- **Instruction Memory:** Supplies the instruction corresponding to the current PC value.
- **Register File:** Contains 32 general-purpose registers. It supports two simultaneous reads and one write per cycle, allowing efficient operand access and result storage.

- **ALU (Arithmetic Logic Unit):** Performs arithmetic (addition, subtraction) and logical operations (AND, OR, XOR, etc.) based on the ALU control signals. It also outputs a Zero flag used for conditional branches.
- **ALU Control Unit:** Determines the ALU operation based on the `funct3`, `funct7`, and opcode fields of the instruction.
- **Immediate Generator:** Extracts and sign-extends immediate values from I-type, S-type, B-type, U-type, and J-type instructions, allowing immediate values to be used in arithmetic operations or address calculations.
- **Control Unit:** Decodes the instruction's opcode to generate all necessary control signals. This includes signals to select ALU inputs, memory read/write, register write-back, and branching logic.
- **Branch Control Unit:** Evaluates branching conditions using register values and determines whether the PC should be updated with a branch target address.
- **Shifter:** Handles shift left logical (SLL), shift right logical (SRL), and shift right arithmetic (SRA) instructions.
- **Data Memory:** Used for load (`lw`) and store (`sw`) operations. Controlled via MemRead, MemWrite, and MemToReg signals from the control unit.
- **Multiplexers (MUXes):** Used to select between different input options at key points in the datapath, such as:
    1. Between register data and immediate values for ALU input
    2. Between ALU result and memory data for write-back
    3. Between sequential PC and branch/jump target
- **Instruction Flow:**
    1. Fetch: Instruction is fetched from memory using the PC.
    2. Decode: Instruction is decoded to generate control signals and read operands from the register file.
    3. Execute: The ALU performs the required computation. Branch decisions are made if needed.

4. Memory: For `lw` and `sw`, the data memory is accessed.
5. Write-back: Results are written back to the register file.

This datapath allows complete instruction execution in one clock cycle by connecting all components through a combinational logic structure, controlled by carefully designed control signals.

4. Modules Implemented

ALU.v

Performs arithmetic and logical operations based on control signals.

ALUcntrl.v

Decodes the funct3 and funct7 fields to select the ALU operation.

Branchcntrl.v

Handles branch condition evaluation.

control_unit.v

Generates control signals based on the instruction opcode.

ImmGen.v

Generates sign-extended immediate values for instructions.

RegFile.v

Implements the register file with two read ports and one write port.

Shifter.v

Handles logical and arithmetic shift operations.

Datapath.v

Integrates all components to execute instructions in a single cycle.

defines.v

Contains constants and macros for opcodes and control signals.

InstrMem.v

Implements the instruction memory that supplies instructions to the datapath based on the current PC.

DataMem.v

Implements the data memory for load (`lw`) and store (`sw`) instructions, controlled by memory control signals.

5. Issues faced and solutions

Debugging

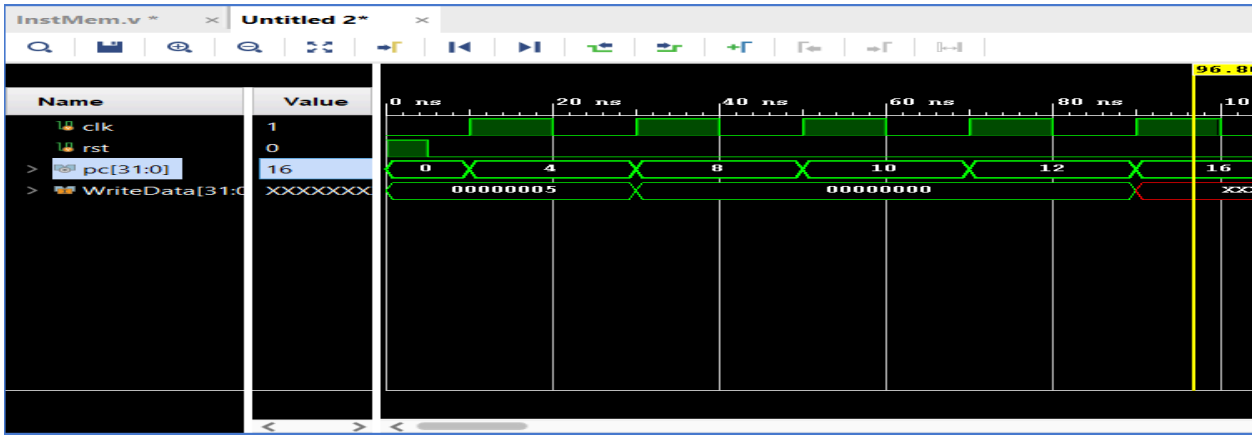
Debugging took a full day of trying different test cases

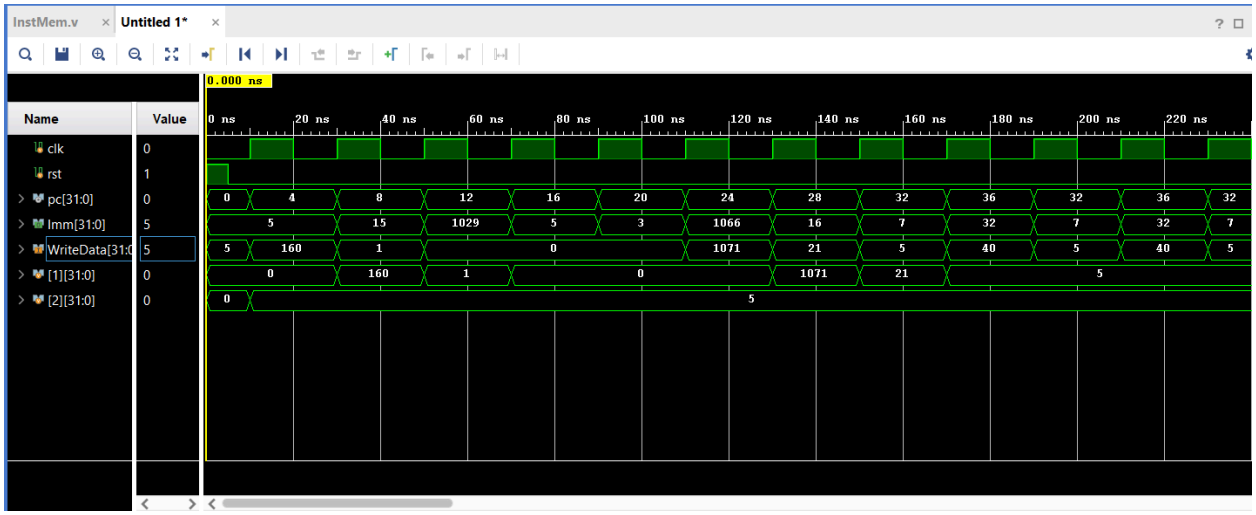
Tracing

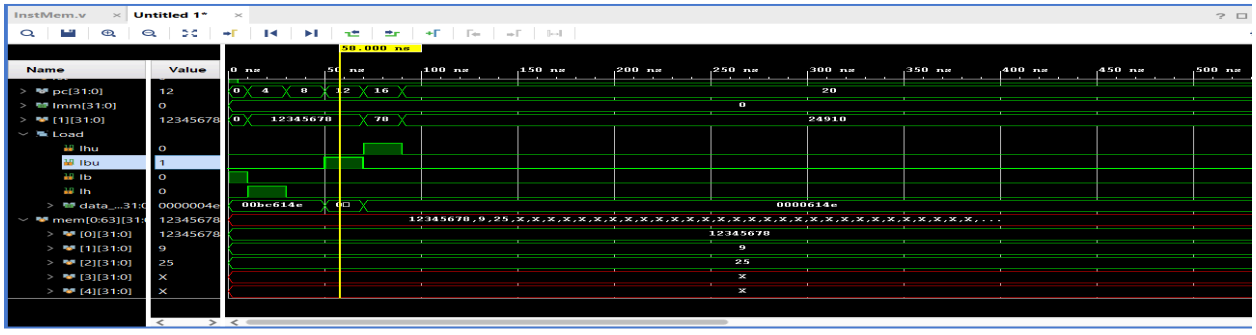
Tracing was difficult across many modules

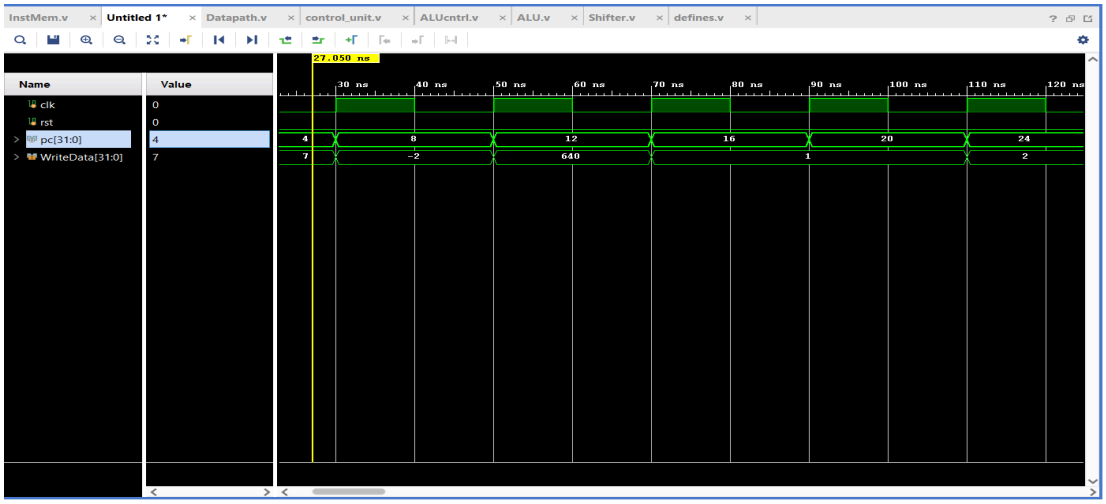
Figuring out the Full Data Path

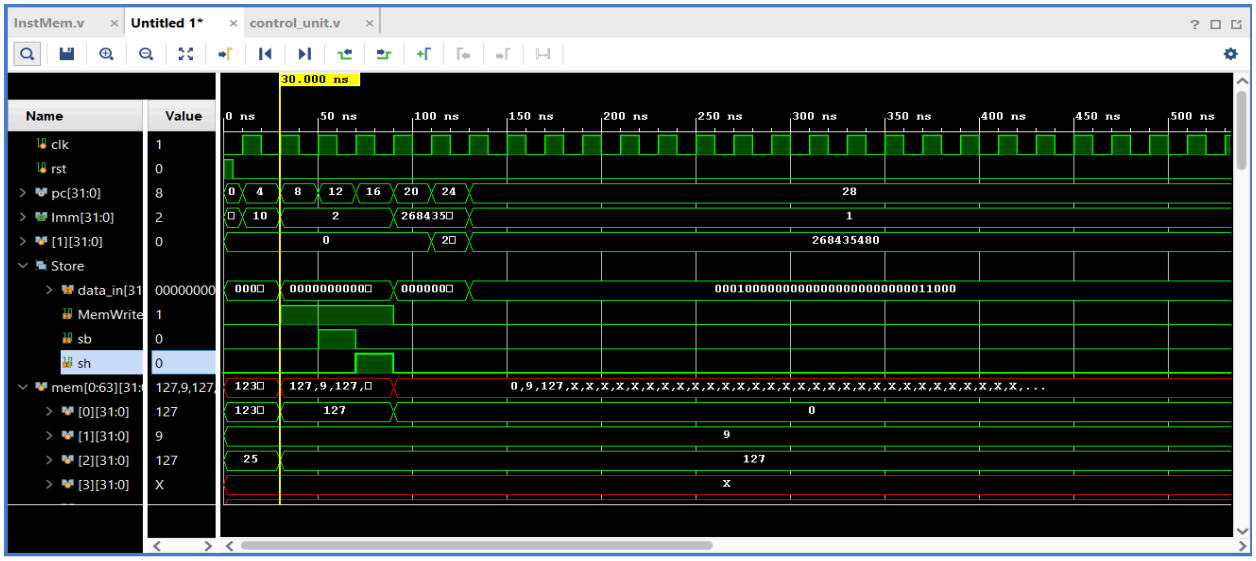
Screenshots and Schematic Report

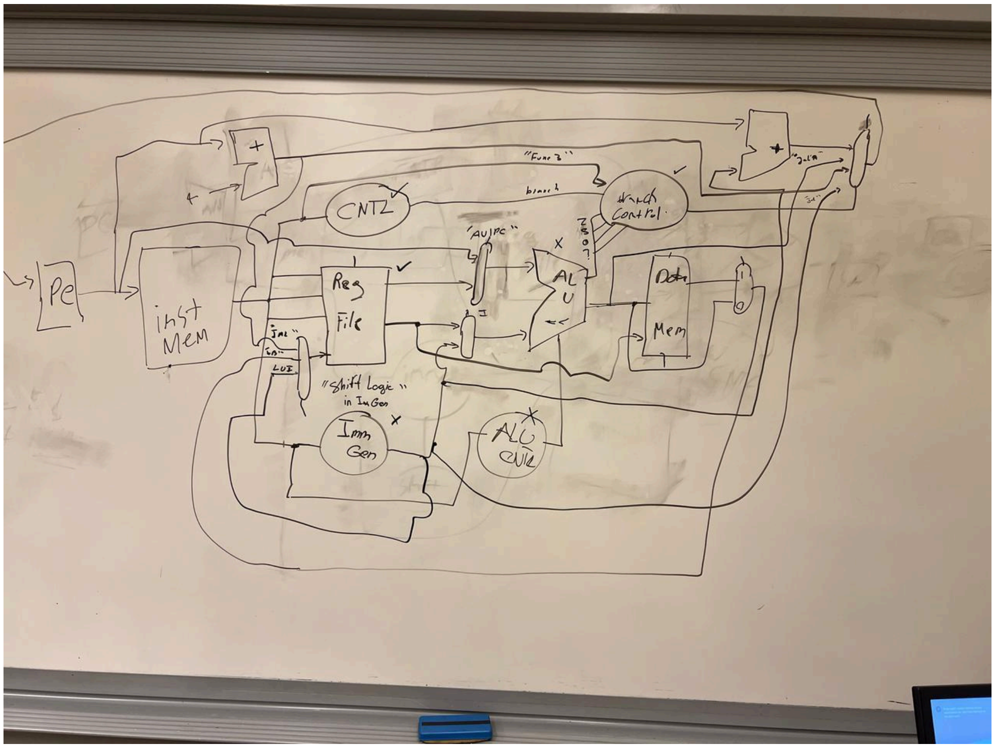







InstMem.v
Untitled 1*
Datapath.v
control_unit.v
ALUcntrl.v
ALU.v
Shifter.v
defines.v
Name
Value
clk
rst
pc[31:0]
WriteData[31:0]
27.050 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
120 ns


InstMem.v
Untitled 1*
control_unit.v
Name
Value
30.000 ns
clk
rst
pc[31:0]
Imm[31:0]
[1][31:0]
Store
data_in[31
MemWrite
sb
sh
mem[0:63][31:
[0][31:0]
[1][31:0]
[2][31:0]
[3][31:0]
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
400 ns
450 ns
500 ns


PC
inst Mem
CNTZ
Reg File
ALU
Data Mem
Imm Gen
ALU cntl
Branch Control
"Shift Logic" in ImmGen
"AUIPC"
"Funct3"
branch
"JAL"
LUI