Datapath and Control Signals:

1. Instruction Fetch Stage: Using instruction address from PC module, fetch instruction from instruction cache, and load into IF/ID Reg.
    a. Control Signals in this stage
        i. load_PC: Load signal for PC register, when set high, register loads value into register.
        ii. Instr_cache_read:
        iii. Alu_sel: Comes from Control rom, if jalr, set alu_sel to 1, otherwise 0.
    b. Data Signals
        i. Cache_Instr_address:
        ii. Cache_Instr_rdata:
        iii. Branch Address: Selects between PC incremented by 4, or conditional/unconditional branch.

2. Instruction Decode Stage: Split instruction from IF/ID Register into fields going into regfile and control_ROM. Control Rom will send the control word to all registers between the pipeline stages. The 32 bit values for rs1 and rs2 are also stored inside the ID/EX register.
    a. Control Signals
    b. Data Signals
        i. Inputs
            1. opcode
            2. funct3
            3. funct7
        ii. Outputs
            1. rs1
            2. rs2
            3. rs1_out
            4. rs2_out
            5. Control Word packed struct
                a. Typedef struct packed {
                    i. **Logic Alumux1_sel**: selects between i_imm and rs2_out to send to CMP.
                    ii. **Logic [2:0] Alumux2_sel**: Selects between all immediate values and rs2_out to pass to the ALU.
                    iii. **Logic Load_regfile**: Tells the reg file to load the value regfilemux_out into register rd.
                    iv. **Logic [3:0] Regfilemux_sel**: Selects between branch enable, u_imm, ALU output and mem_rdata to load into regfile.
                    v. **Logic [2:0] aluop**: Indicates which ALU operation to compute inside the ALU.

vi. **Logic [2:0] cmpop**: Indicates which compare operation to perform in the CMP.
vii. **Logic Instruction/data cache_read**: Read request to send to either the Instruction or data cache.
viii. **Logic Data cache_write**: Write request to the data cache. Instruction cache is ROM.
ix. **Logic cmpmux_sel**: Selects between i_imm and rs2_out.
x. **Logic pcmux_sel:** Selects between output of PC register added with 4, alu_out, and alu_out with LSB set to 0. This select signal is br_en if the corresponding pipeline stage is working on the branch instruction. Otherwise, it has to be set to pcmux::alu_mod2 or pcmux::alu_out if its on jalr or jal instruction, respectively. The **Default** value for this signal is 0 or pc_plus4.
xi. **Load_pc:** Where does this signal come from.

b. } control_word

3. Execution Stage: Where comparisons and arithmetic/logical operations take place. Outputs of these operations are stored in the EX_MEM register.
    a. Control Signals
        i. Inputs
            1. Alumux1_sel
            2. Alumux2_sel
            3. Cmpmux_sel
        ii. Outputs
    b. Data Signals
        i. Inputs
            1. rs1_out
            2. rs2_out
            3. i_imm
            4. u_imm
            5. b_imm
            6. s_imm
            7. J_imm
            8. aluop
        ii. Intermediate Data Path Signals
            1. Alumux1_out
            2. Alumux2_out
            3. cmpmux_out
        iii. Outputs
            1. Alu_out:

            2. Br_en:

4. Memory Access Stage: Retrieve data from cache using alu_out. Load data into MEM_WB register.
    a. Control Signals:
        i. datacache_read
        ii. datacache_write
        iii. mdr_load
    b. Datapath Signals:
        i. Outputs
            1. Alu_out
            2. mdr_out

5. Register Write Back Stage: Choose between datapath input signals to load into regfile using regfilemux_sel.
    a. Control Signals
        i. Regfilemux_sel:
        ii. Load_regfile:
    b. Datapath Signals
        i. Inputs
            1. br_en
            2. u_imm
            3. alu_out
            4. Mem_rdata
        ii. Outputs
            1. regfilemux_out