MIPS32_RISC

This **Verilog module** simulates a **5-stage pipelined MIPS32 processor**, a classic RISC design. It models key features such as:

- Instruction and data memory
- Register bank
- ALU
- Pipeline registers between stages
- Hazard control (basic branch handling)
- Two-phase clocking (`clk1`, `clk2`) for alternating pipeline stages

---

MIPS32 Instruction Set and Architecture

- **32-bit processor**: Each instruction and register is 32 bits wide.
- **Registers**: 32 general-purpose registers (R0–R31)
    - `R0` is set to **0**
- **Memory**: Word-addressable (each address points to a 32-bit word)
- **Instruction Types**:
    - **R-type**: All operands are registers (e.g., `ADD`, `SUB`)
    - **I-type**: Uses immediate values or memory addresses (e.g., `LW`, `SW`, `ADDI`)
    - **J-type**: Used for jumps (e.g., `J`)
- **No flags**: Condition evaluation is done through registers

---

Instruction Set in Code :

Type	Instructions
R-type	`ADD`, `SUB`, `AND`, `OR`, `MUL`, `SLT`
I-type	`ADDI`, `SUBI`, `SLTI`, `LW`, `SW`
Branch	`BEQZ`, `BNEQZ`
Jump	J (not implemented explicitly in code)
Misc.	`HLT`

Instruction Cycle (Pipeline Stages)

Each instruction flows through **five pipeline stages**, each performing a distinct task:

---

1. IF – Instruction Fetch (Clock: `clk1`)

- Fetches the instruction from memory using the current **PC** (Program Counter)
- Increments PC by 1 (next instruction address)
- Saves the instruction and next PC in pipeline registers: `IF_ID_IR`, `IF_ID_NPC`

Branch Handling:
If a branch is taken (determined in MEM stage), PC is updated to the target address, and next instruction is fetched from that address.

---

2. ID – Instruction Decode / Register Fetch (Clock: `clk2`)

- Decodes the opcode and instruction fields (`rs`, `rt`, `rd`, `immediate`)
- Fetches operands from the **register file**:

    - `rs` → `ID_EX_A`
    - `rt` → `ID_EX_B`
- Sign-extends 16-bit immediate to 32 bits → `ID_EX_Imm`
- Classifies instruction type (`RR_ALU`, `RM_ALU`, `LOAD`, etc.) using opcode

Stores decoded info in the `ID_EX_*` pipeline registers.

---

3. EX – Execute / Effective Address Calculation (Clock: `clk1`)

- Performs the actual computation:
    - **R-type**: ALU operates on `ID_EX_A` and `ID_EX_B`
    - **I-type**: ALU operates on `ID_EX_A` and `ID_EX_Imm`
    - **LOAD/STORE**: Computes memory address: `base + offset`
    - **BRANCH**: Computes target PC address (`NPC + offset`) and condition

Results and control signals are saved into `EX_MEM_*` pipeline registers.

---

4. MEM – Memory Access / Branch Completion (Clock: `clk2`)

- **LOAD**: Reads data from memory → `MEM_WB_LMD`
- **STORE**: Writes data to memory (`Mem[address] = EX_MEM_B`) — **only if branch not taken**
- For branches, if condition is true, the processor updates PC and sets `TAKEN_BRANCH` to cancel following instructions.

Results go into `MEM_WB_*` registers.

---

5. WB – Register Write-Back (Clock: `clk1`)

- Writes back results to registers
- Result may or may not come from ALU/ Memory system.

Write-back is **cancelled** if a branch was just taken (`TAKEN_BRANCH == 1`) to prevent invalid writes.

Features :

Two-Phase Clocking

- Alternates pipeline stages:
    - `clk1`: IF, EX, WB
    - `clk2`: ID, MEM
- Allows the stages to operate efficiently without overlapping too much.

Branch Handling (`TAKEN_BRANCH`)

- Simple branch prediction: only resolves branch in MEM stage.
- If a branch is taken, the next few instructions are flushed or ignored to avoid incorrect execution.

HALT Instruction

- Stops pipeline after reaching WB stage
- `HALTED` flag is set, and no further instructions are fetched or processed.
