

```

module CPU (clock);
parameter LW = 6'b100011, SW = 6'b101011, BEQ = 6'b000100, no-op = 32'b000000_100000, ALUop = 6'b0;
input clock;
    reg[31:0] PC, Regs[0:31], IMemory[0:1023], DMemory[0:1023], // separate memories
        IFIDIR, IDEXA, IDEXB, IDEXIR, EXMEMIR, EXMEMB, // pipeline registers
        EXMEMALUOut, MEMWBValue, MEMWBIR; // pipeline registers
    wire [4:0] IDEXrs, IDEXrt, EXMEMrd, MEMWBrd, MEMWBrt; //hold register fields
    wire [5:0] EXMEMop, MEMWBop, IDEXop; Hold opcodes
    wire [31:0] Ain, Bin;

// declare the bypass signals
    wire stall, bypassAfromMEM, bypassAfromALUinWB, bypassBfromMEM, bypassBfromALUinWB,
        bypassAfromLWinWB, bypassBfromLWinWB;

    assign IDEXrs = IDEXIR[25:21];    assign IDEXrt = IDEXIR[15:11];    assign EXMEMrd = EXMEMIR[15:11];
    assign MEMWBrd = MEMWBIR[20:16]; assign EXMEMop = EXMEMIR[31:26];
    assign MEMWBrt = MEMWBIR[25:20];
    assign MEMWBop = MEMWBIR[31:26]; assign IDEXop = IDEXIR[31:26];
    // The bypass to input A from the MEM stage for an ALU operation
    assign bypassAfromMEM = (IDEXrs == EXMEMrd) & (IDEXrs!=0) & (EXMEMop==ALUop); // yes, bypass
    // The bypass to input B from the MEM stage for an ALU operation
    assign bypassBfromMEM = (IDEXrt == EXMEMrd) & (IDEXrt!=0) & (EXMEMop==ALUop); // yes, bypass
    // The bypass to input A from the WB stage for an ALU operation
    assign bypassAfromALUinWB = (IDEXrs == MEMWBrd) & (IDEXrs!=0) & (MEMWBop==ALUop);
    // The bypass to input B from the WB stage for an ALU operation
    assign bypassBfromALUinWB = (IDEXrt == MEMWBrd) & (IDEXrt!=0) & (MEMWBop==ALUop); /
    // The bypass to input A from the WB stage for an LW operation
    assign bypassAfromLWinWB = (IDEXrs == MEMWBIR[20:16]) & (IDEXrs!=0) & (MEMWBop==LW);
    // The bypass to input B from the WB stage for an LW operation
    assign bypassBfromLWinWB = (IDEXrt == MEMWBIR[20:16]) & (IDEXrt!=0) & (MEMWBop==LW);
    // The A input to the ALU is bypassed from MEM if there is a bypass there,
    // Otherwise from WB if there is a bypass there, and otherwise comes from the IDEX register
    assign Ain = bypassAfromMEM? EXMEMALUOut :
        (bypassAfromALUinWB | bypassAfromLWinWB)? MEMWBValue : IDEXA;
    // The B input to the ALU is bypassed from MEM if there is a bypass there,
    // Otherwise from WB if there is a bypass there, and otherwise comes from the IDEX register
    assign Bin = bypassBfromMEM? EXMEMALUOut :
        (bypassBfromALUinWB | bypassBfromLWinWB)? MEMWBValue : IDEXB;

    // The signal for detecting a stall based on the use of a result from LW
    assign stall = (MEMWBIR[31:26]==LW) && // source instruction is a load
        (((IDEXop==LW)|(IDEXop==SW)) && (IDEXrs==MEMWBrd)) | // stall for address calc
        ((IDEXop==ALUop) && ((IDEXrs==MEMWBrd)|(IDEXrt==MEMWBrd))); // ALU use

    reg [5:0] i; //used to initialize registers

    initial begin
        PC = 0;
        IFIDIR = no-op; IDEXIR = no-op; EXMEMIR = no-op; MEMWBIR = no-op; // put no-ops in pipeline registers
        for (i = 0; i<=31; i = i+1) Regs[i] = i; //initialize registers--just so they aren't cares
    end

    always @ (posedge clock) begin
        if (~stall) begin // the first three pipeline stages stall if there is a load hazard

```

FIGURE e4.14.3 A behavioral definition of the five-stage MIPS pipeline with stalls for loads when the destination is an ALU instruction or effective address calculation. The changes from Figure e4.14.2 are highlighted.