```
module CPU (clock);

  // Instruction opcodes
   parameter LW = 6'b100011, SW = 6'b101011, BEQ = 6'b000100, no-op = 32'b00000_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; // Access register fields
   wire [5:0] EXMEMop, MEMWBop, IDEXop; // Access opcodes
wire [31:0] Ain, Bin; // the ALU inputs
// These assignments define fields from the pipeline registers
   assign IDEXrs = IDEXIR[25:21];   // rs field
   assign IDEXrt = IDEXIR[20:16];   // rt field
   assign EXMEMrd = EXMEMIR[15:11]; // rd field
   assign MEMWBrd = MEMWBIR[15:11]; //rd field
   assign MEMWBrt = MEMWBIR[20:16]; //rt field--used for loads
   assign EXMEMop = EXMEMIR[31:26]; // the opcode
   assign MEMWBop = MEMWBIR[31:26]; // the opcode
   assign IDEXop = IDEXIR[31:26];   // the opcode
   // Inputs to the ALU come directly from the ID/EX pipeline registers
   assign Ain = IDEXA;
   assign Bin = IDEXB;
   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
   // Remember that ALL these actions happen every pipe stage and with the use of <= they happen in parallel!
   // first instruction  in the pipeline is being fetched
         IFIDIR <= IMemory[PC>>2];
         PC <= PC + 4;
      end // Fetch & increment PC
      // second instruction in pipeline is fetching registers
         IDEXA <= Regs[IFIDIR[25:21]]; IDEXB <= Regs[IFIDIR[20:16]]; // get two registers
         IDEXIR <= IFIDIR;  //pass along IR--can happen anywhere, since this affects next stage only!
      // third instruction is doing address calculation or ALU operation
      if ((IDEXop==LW) |(IDEXop==SW))  // address calculation
          EXMEMALUOut <= IDEXA +{{16{IDEXIR[15]}}, IDEXIR[15:0]};
         else if (IDEXop==ALUop) case (IDEXIR[5:0]) //case for the various R-type instructions
              32: EXMEMALUOut <= Ain + Bin;  //add operation
              default: ; //other R-type operations: subtract, SLT, etc.
         endcase
```

FIGURE e4.14.1 A Verilog behavorial model for the MIPS five-stage pipeline, ignoring branch and data hazards. As in the design earlier in Chapter 4, we use separate instruction and data memories, which would be implemented using separate caches as we describe in Chapter 5.