```
module CPU (clock);
   parameter LW = 6'b100011, SW = 6'b101011, BEQ = 6'b000100, J = 6'd2; //constants
   input clock; reg [2:0] state;
   wire [1:0] ALUOp, ALUSrcB, PCSource; wire [5:0] opcode;
   wire RegDst, MemRead, MemWrite, IorD, RegWrite, IRWrite, PCWrite, PCWriteCond,
        ALUSrcA, MemoryOp, IRWwrite, Mem2Reg;
// Create an instance of the MIPS datapath, the inputs are the control signals; opcode is only output
Datapath MIPSDP (ALUOp,RegDst,Mem2Reg, MemRead, MemWrite, IorD, RegWrite,
   IRWrite, PCWrite, PCWriteCond, ALUSrcA, ALUSrcB, PCSource, opcode, clock);
 initial begin state = 1; end // start the state machine in state 1
// These are the definitions of the control signals
assign IRWrite = (state==1);
assign Mem2Reg = ~ RegDst;
assign MemoryOp = (opcode==LW)|(opcode==SW); // a memory operation
assign ALUOp = ((state==1)|(state==2)|((state==3)&MemoryOp)) ? 2'b00 : // add
       ((state==3)&(opcode==BEQ)) ? 2'b01 : 2'b10; // subtract or use function code
   assign RegDst = ((state==4)&(opcode==0)) ? 1 : 0;
   assign MemRead = (state==1) | ((state==4)&(opcode==LW));
   assign MemWrite = (state==4)&(opcode==SW);
   assign IorD = (state==1) ? 0 : (state==4) ? 1 : X;
   assign RegWrite = (state==5) | ((state==4) &(opcode==0));
   assign PCWrite = (state==1) | ((state==3)&(opcode==J));
   assign PCWriteCond = (state==3)&(opcode==BEQ);
   assign ALUSrcA = ((state==1)|(state==2)) ? 0 :1;
   assign ALUSrcB = ((state==1) | ((state==3)&(opcode==BEQ))) ? 2'b01 : (state==2) ? 2'b11 :
          ((state==3)&MemoryOp) ? 2'b10 : 2'b00; // memory operation or other
   assign PCSource = (state==1) ? 2'b00 : ((opcode==BEQ) ? 2'b01 : 2'b10);
// Here is the state machine, which only has to sequence states

   always @(posedge clock) begin // all state updates on a positive clock edge
      case (state)
      1: state = 2;  //unconditional next state
      2: state = 3;  //unconditional next state
      3: // third step: jumps and branches complete
         state = ((opcode==BEQ) | (opcode==J)) ? 1 : 4;// branch or jump go back else next state
      4: state = (opcode==LW) ? 5 : 1; //R-type and SW finish
      5: state = 1; // go back
       endcase
end
  endmodule
```

FIGURE e4.14.7 The MIPS CPU using the datapath from Figure E4.14.6.