```
         else if (opcode == BEQ) begin
                    if (A==B) PC <= ALUOut; // branch taken--update PC
                    state = 1;
          end
          else if (opocde=J) begin
               PC = {PC[31:28], IR[25:0],2'b00}; // the jump target PC
               state = 1;
          end  //Jumps
                  else ; // other opcodes or exception for undefined instruction would go here
   end


   4: begin
       if (opcode==6'b0) begin //ALU Operation
             Regs[IR[15:11]] <= ALUOut; // write the result
             state = 1;
       end //R-type finishes
          else if (opcode == LW) begin // load instruction
              MDR <= Memory[ALUOut>>2]; // read the memory
              state = 5; // next state
          end
                 else if (opcode == LW) begin
                     Memory[ALUOut>>2] <= B; // write the memory
                     state = 1; // return to state 1
                 end //store finishes
                      else ; // other instructions go here
        end


   5: begin // LW is the only instruction still in execution
        Regs[IR[20:16]] = MDR; // write the MDR to the register
        state = 1;
      end //complete an LW instruction
   endcase
end
endmodule
```

FIGURE e4.14.5 A behavioral specification of the multicycle MIPS design. (*Continued*)