1. 8-bit Arithmetic Logic Unit (ALU)

```
module ALU (
    input [7:0] A,       // 8-bit input operand A
    input [7:0] B,       // 8-bit input operand B
    input [2:0] OpCode,  // 3-bit opcode to specify the operation
    output reg [7:0] Result, // 8-bit output for the result
    output reg ZeroFlag       // Zero flag to indicate if Result is zero
);

always @(*) begin
    case (OpCode)
        3'b000: Result = A + B;      // ADD
        3'b001: Result = A - B;      // SUBTRACT
        3'b010: Result = A & B;      // AND
        3'b011: Result = A | B;      // OR
        3'b100: Result = A ^ B;      // XOR
        3'b101: Result = ~A;         // NOT
        default: Result = 8'b00000000; // Default to 0 for undefined opcodes
    endcase

    // Set the ZeroFlag if the result is zero
    ZeroFlag = (Result == 8'b00000000) ? 1 : 0;
end

Endmodule
```

2. Register Module

```
module Register (
    input clk,            // Clock input
    input reset,          // Reset input
    input [7:0] D,        // 8-bit data input
    output reg [7:0] Q    // 8-bit data output
);

always @(posedge clk or posedge reset) begin
    if (reset)
        Q <= 8'b00000000;  // Reset the register to 0
    else
        Q <= D;           // Load data D into register Q on clock edge
end
```

```
endmodule
```

Testing in Simulation

```
module ALU_tb;

reg [7:0] A, B;
reg [2:0] OpCode;
wire [7:0] Result;
wire ZeroFlag;

// Instantiate the ALU module
ALU uut (
   .A(A),
   .B(B),
   .OpCode(OpCode),
   .Result(Result),
   .ZeroFlag(ZeroFlag)
);

initial begin
   // Test addition
   A = 8'b00000101; // 5
   B = 8'b00000011; // 3
   OpCode = 3'b000; // ADD
   #10;

   // Test subtraction
   A = 8'b00000101; // 5
   B = 8'b00000011; // 3
   OpCode = 3'b001; // SUB
   #10;

   // Test AND
   A = 8'b00000101; // 5
   B = 8'b00000011; // 3
   OpCode = 3'b010; // AND
   #10;

   // Test OR
   A = 8'b00000101; // 5
   B = 8'b00000011; // 3
```

```
    OpCode = 3'b011; // OR
    #10;

    // Test NOT
    A = 8'b00000101; // 5
    OpCode = 3'b101; // NOT
    #10;
end

endmodule
```