Task 3

Processor Design (Verilog):

```
module pipelined_processor (
    input wire clk, reset
);

    // Instruction format: [7:6] Opcode, [5:3] Rs, [2:0] Rt
    reg [7:0] instruction_memory [0:15];
    reg [7:0] register_file [0:7];
    reg [7:0] data_memory [0:15];
   reg [7:0] IF_ID;
    reg [7:0] ID_EX_A, ID_EX_B;
    reg [7:0] ID_EX_Imm;
    reg [1:0] ID_EX_Opcode;
    reg [2:0] ID_EX_Dest;
   reg [7:0] EX_WB_Result;
    reg [2:0] EX_WB_Dest;
    reg EX_WB_WriteEnable
```

```
    reg [3:0] pc = 0;
always @(posedge clk or posedge reset) begin
        if (reset) begin
            pc <= 0;
        end else begin
            IF_ID <= instruction_memory[pc];
            pc <= pc + 1;
        end
    end
    always @(posedge clk) begin
        ID_EX_Opcode <= IF_ID[7:6];
        ID_EX_Dest <= IF_ID[5:3];
        ID_EX_Imm <= IF_ID[2:0];
        ID_EX_A <= register_file[IF_ID[5:3]];
        ID_EX_B <= register_file[IF_ID[2:0]];
    end
    reg [7:0] ALU_result;

    always @(*) begin
```

```
 case (ID_EX_Opcode)
        2'b00: ALU_result = ID_EX_A + ID_EX_B; // ADD
        2'b01: ALU_result = ID_EX_A - ID_EX_B; // SUB
        2'b10: ALU_result = data_memory[ID_EX_Imm]; //
LOAD
        default: ALU_result = 8'b00000000;
      endcase
   end
   always @(posedge clk) begin
      EX_WB_Result <= ALU_result;
      EX_WB_Dest <= ID_EX_Dest;
      EX_WB_WriteEnable <= (ID_EX_Opcode != 2'b10
   end
   always @(posedge clk) begin
      if (EX_WB_WriteEnable)
         register_file[EX_WB_Dest] <= EX_WB_Result;
   end

endmodule
```

Testbench (Verilog)

```
module tb_pipelined_processor;
    reg clk, reset;
    pipelined_processor uut (.clk(clk), .reset(reset));
    always #5 clk = ~clk;
  initial begin
        clk = 0;
        reset = 1;
        #10 reset = 0;
        uut.instruction_memory[0] = 8'b00001001;
        uut.instruction_memory[1] = 8'b01001110;
        uut.instruction_memory[2] = 8'b10010100;
        uut.register_file[0] = 8'h05;
        uut.register_file[2] = 8'h08;
        #50 $finish;
    end
```

```
    initial begin
        $monitor("Time = %0t | PC = %0d | R1 = %h | R3 = %h | R4 = %h",
                 $time, uut.pc, uut.register_file[1], uut.register_file[3], uut.register_file[4]);
    end

endmodule
```