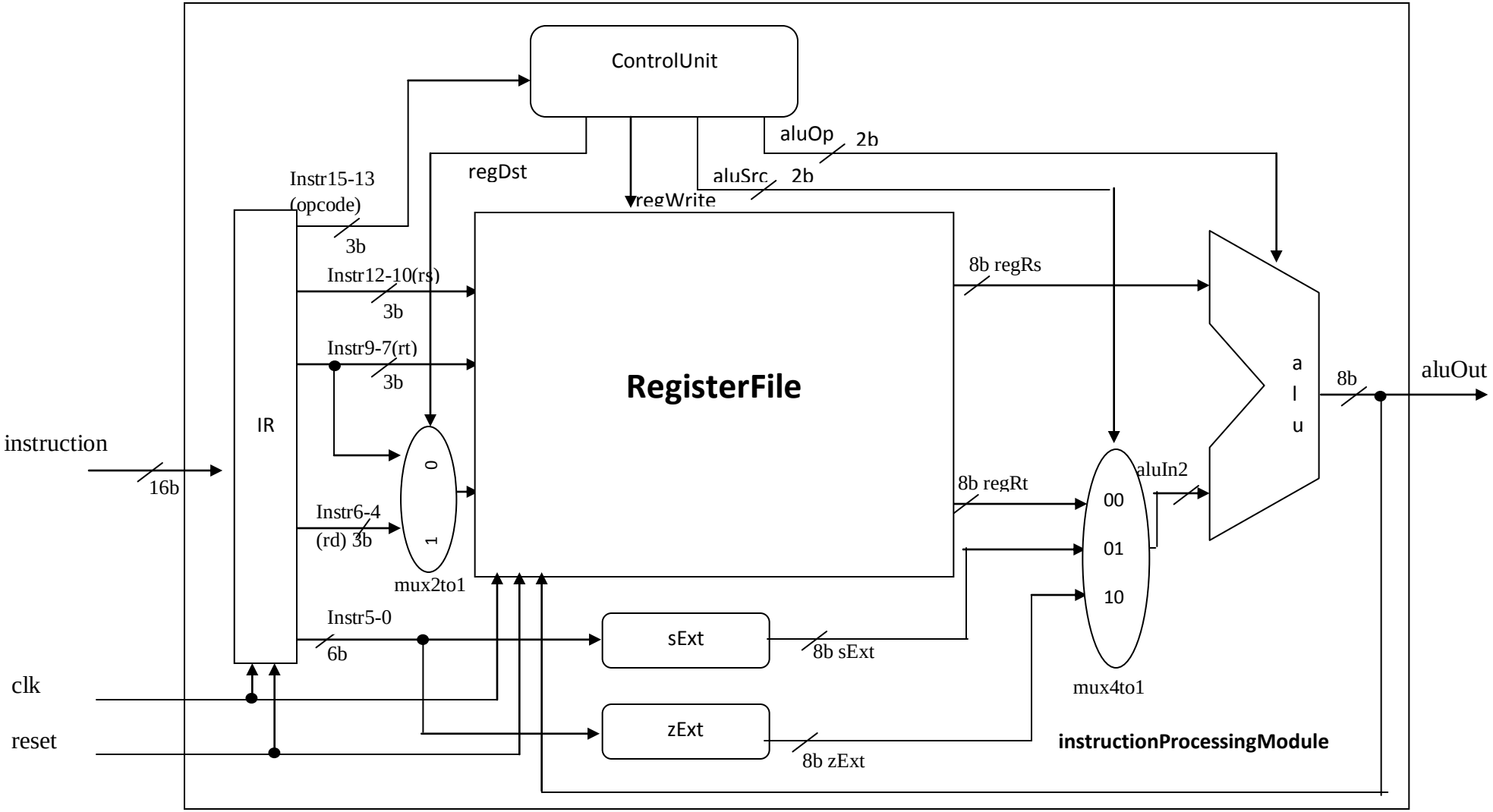
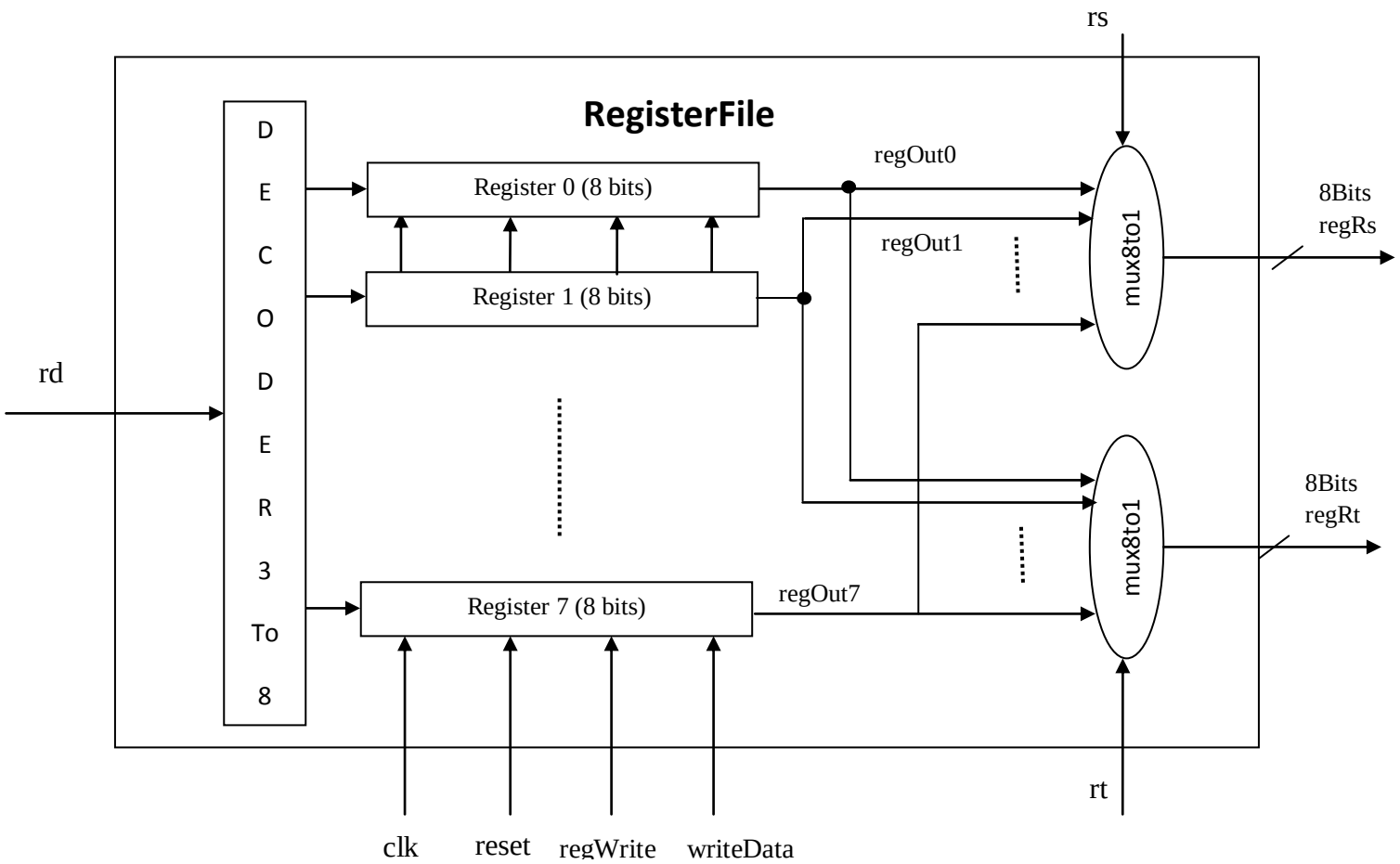
BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI K. K. BIRLA GOA CAMPUS
Second Semester 2013-2014
COMPUTER ORGANIZATION (CS C342, IS C351, IS F242)

LABTEST – 1 **(Marks : 25)**

Design and implement the following using verilog HDL in **Xilinx software** with the following specifications.




RegisterFile
rs
rd
D
E
C
O
D
E
R
3
To
8
Register 0 (8 bits)
Register 1 (8 bits)
Register 7 (8 bits)
regOut0
regOut1
regOut7
mux8to1
mux8to1
8Bits
regRs
8Bits
regRt
rt
clk
reset
regWrite
writeData

Modules

1. **module D_ff(input clk,input reset,input regWrite,input decOut,input d,output reg q);**

```
always @ (negedge clk)
begin
if(reset==1)
        q=0;
else
        if(regWrite == 1 && decOut==1)
        begin
                q=d;
        end
end
```

   endmodule

2. **module IR(input clk,input reset,input [15:0] InstructionIn,output reg [15:0] InstructionOut);**
   On negedge of clock if reset is 1 then InstructionOut is 16'b0 else on negedge of clock InstructionOut is InstructionIn.
3. **module controlUnit(input [2:0] opcode,output reg regDst,output reg regWrite,output reg [1:0] aluSrc,output reg [1:0] aluOp);**
   if opcode is 3'b000 then regDst = 1, regWrite = 1, aluSrc = 0 and aluOp = 0.
   if opcode is 3'b001 then regDst = 1, regWrite = 1, aluSrc = 0 and aluOp = 1.
   if opcode is 3'b010 then regDst = 0, regWrite = 1, aluSrc = 1 and aluOp = 0.
   if opcode is 3'b011 then regDst = 0, regWrite = 1, aluSrc = 2 and aluOp = 2.
   if opcode is 3'b100 then regDst = 0, regWrite = 1, aluSrc = 2 and aluOp = 3.
4. **module regFile(input clk,input reset,input [2:0] rs,input [2:0] rt,input [2:0] destinationRegister,input regWrite,input [7:0] writeData,output [7:0] regRs,output [7:0] regRt);**
5. **module decoder3to8(input [2:0] decIn,output reg [7:0] decOut);**
6. **module reg8bits(input clk,input reset,input regWrite,input decOut,input [7:0] regIn,output [7:0] regOut);**

7. **module mux8to1(input [7:0] regOut0,input [7:0] regOut1,input [7:0] regOut2,input [7:0] regOut3,input [7:0] regOut4,input [7:0] regOut5,input [7:0] regOut6,input [7:0] regOut7,input [2:0] sel,output reg [7:0] outBus);**
8. **module sExt(input [5:0] instr5_0,output reg [7:0] sExt);**
   Module sign extents 6 bit input to 8 bit output.
9. **module zExt(input [5:0] instr5_0,output reg [7:0] zExt);**
   Module zero extents 6 bit input to 8 bit output.
10. **module mux2to1(input [2:0] rt,input [2:0] rd,input regDst,output reg [2:0] destinationReg);**
11. **module mux4to1(input [7:0] regRt,input [7:0] sExt,input [7:0] zExt,input [1:0] aluSrc,output reg [7:0] aluIn2);**
12. **module alu(input [7:0] aluIn1,input [7:0] aluIn2,input [1:0] aluOp, output reg [7:0] aluOut);**
    if aluOp = 0 then aluOut = aluIn1 + aluIn2.
    if aluOp = 1 then aluOut = aluIn1 - aluIn2.
    if aluOp = 2 then aluOut = aluIn1 & aluIn2.
    if aluOp = 3 then aluOut = aluIn1 | aluIn2.
13. **module instructionProcessing(input clk,input reset,input [15:0] instruction,output [7:0] aluOut);**
14. **module testBench;**

```
        // Inputs
        reg clk;
        reg reset;
        reg [15:0] instruction;

    // Outputs
        wire [7:0] aluOut;

        // Instantiate the Unit Under Test (UUT)
        instructionProcessing uut (
                .clk(clk),
                .reset(reset),
```

```
        .instruction(instruction),
        .aluOut(aluOut)
);

always
#5 clk = ~clk;
initial begin

//$monitor(" instruction %h aluOut %d", instruction,aluOut);
        $monitor("aluOut %d",aluOut);
// Initialize Inputs
        clk = 0;
        reset = 1;
        instruction = 0;
#5
  reset = 0;
  instruction = 16'h4006; //R0<- 6
 #10
  instruction = 16'h40BF; //R1<- 5
 #10
  instruction = 16'h8904; //R2<- 4
 #10
  instruction = 16'h8BBF; //R7<- 63 <-0011 1111
 #10
  instruction = 16'h20D0; //R5 <- 1
 #10
  instruction = 16'h16C0; //R4 <- 2
 #10
  instruction = 16'h6183; //R3 <- 2
```

```
        #10
         instruction = 16'h4D81; //R3 <- 3
        #10
         instruction = 16'h7F3B; //R6 <- 59 <- 0011 1011
    #20;
                $finish;
       end

endmodule
```