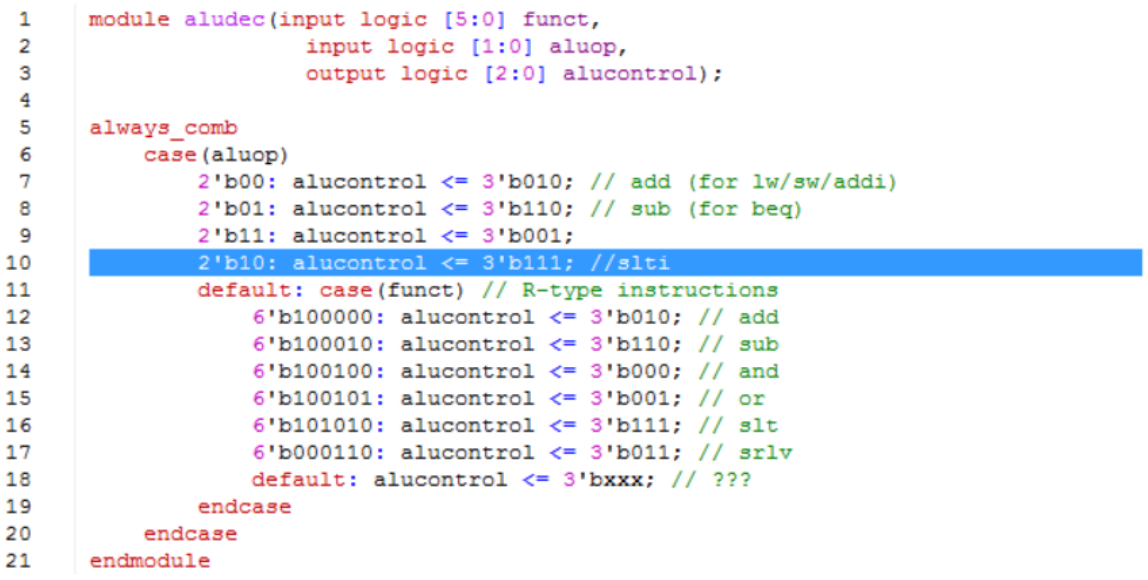
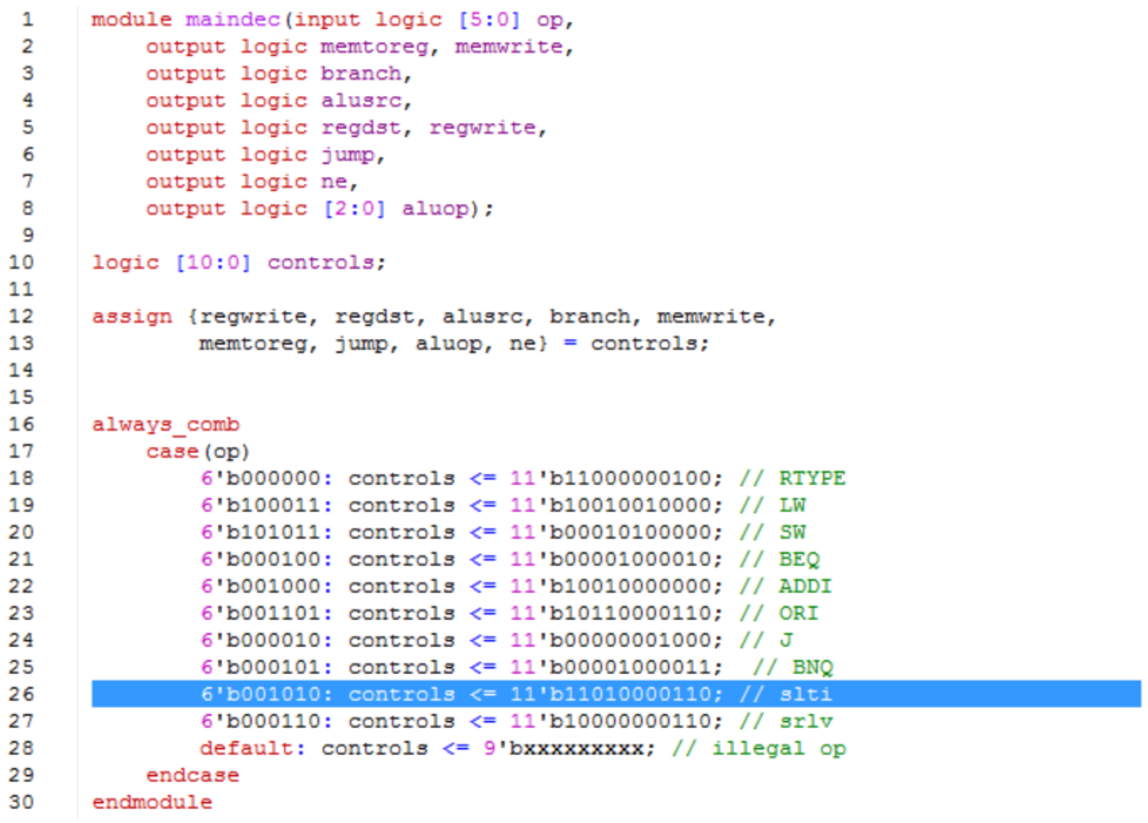
slti

aludec



```
module aludec(input logic [5:0] funct,
                input logic [1:0] aluop,
                output logic [2:0] alucontrol);

always_comb
    case(aluop)
        2'b00: alucontrol <= 3'b010; // add (for lw/sw/addi)
        2'b01: alucontrol <= 3'b110; // sub (for beq)
        2'b11: alucontrol <= 3'b001;
        2'b10: alucontrol <= 3'b111; //slti
        default: case(funct) // R-type instructions
            6'b100000: alucontrol <= 3'b010; // add
            6'b100010: alucontrol <= 3'b110; // sub
            6'b100100: alucontrol <= 3'b000; // and
            6'b100101: alucontrol <= 3'b001; // or
            6'b101010: alucontrol <= 3'b111; // slt
            6'b000110: alucontrol <= 3'b011; // srlv
            default: alucontrol <= 3'bxxx; // ???
        endcase
    endcase
endmodule
```

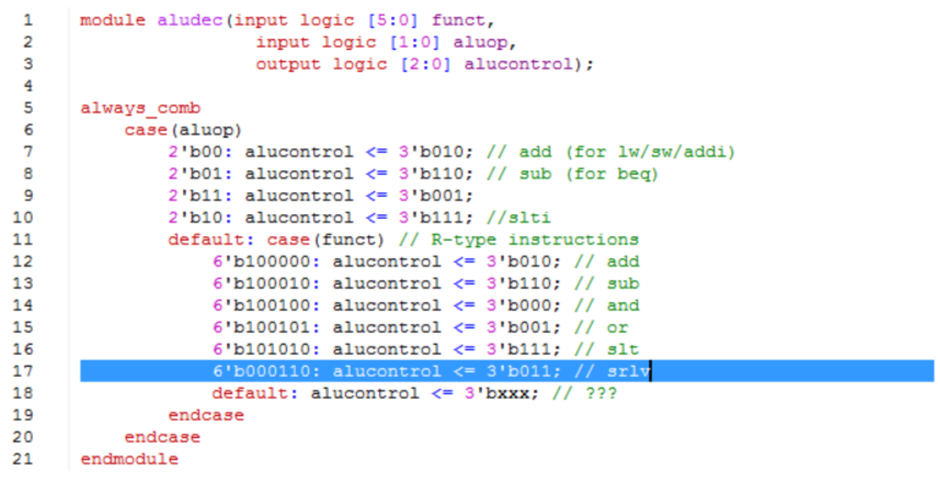
maindec



```
module maindec(input logic [5:0] op,
    output logic memtoreg, memwrite,
    output logic branch,
    output logic alusrc,
    output logic regdst, regwrite,
    output logic jump,
    output logic ne,
    output logic [2:0] aluop);

logic [10:0] controls;

assign {regwrite, regdst, alusrc, branch, memwrite,
        memtoreg, jump, aluop, ne} = controls;


always_comb
    case(op)
        6'b000000: controls <= 11'b11000000100; // RTYPE
        6'b100011: controls <= 11'b10010010000; // LW
        6'b101011: controls <= 11'b00010100000; // SW
        6'b000100: controls <= 11'b00001000010; // BEQ
        6'b001000: controls <= 11'b10010000000; // ADDI
        6'b001101: controls <= 11'b10110000110; // ORI
        6'b000010: controls <= 11'b00000001000; // J
        6'b000101: controls <= 11'b00001000011;  // BNQ
        6'b001010: controls <= 11'b11010000110; // slti
        6'b000110: controls <= 11'b10000000110; // srlv
        default: controls <= 9'bxxxxxxxxx; // illegal op
    endcase
endmodule
```

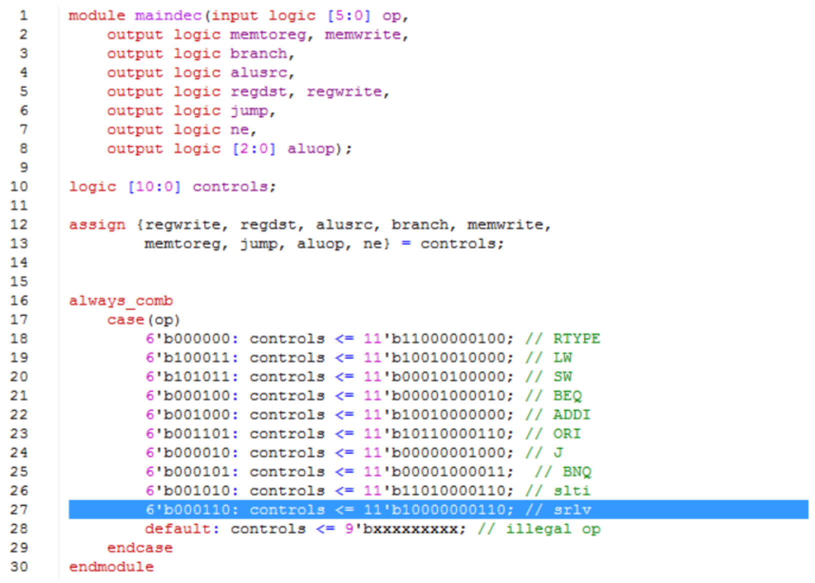
srlv

aludec



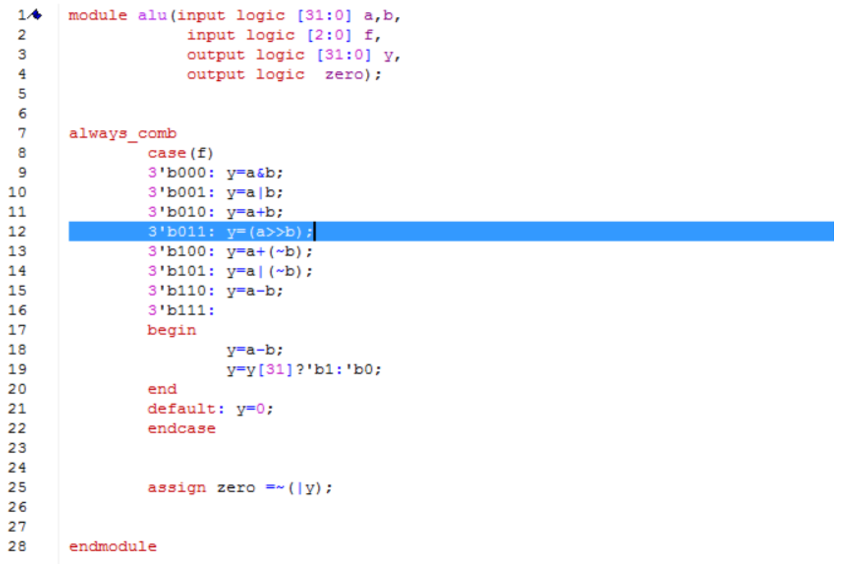
```
module aludec(input logic [5:0] funct,
                input logic [1:0] aluop,
                output logic [2:0] alucontrol);

always_comb
    case(aluop)
        2'b00: alucontrol <= 3'b010; // add (for lw/sw/addi)
        2'b01: alucontrol <= 3'b110; // sub (for beq)
        2'b11: alucontrol <= 3'b001;
        2'b10: alucontrol <= 3'b111; //slti
        default: case(funct) // R-type instructions
            6'b100000: alucontrol <= 3'b010; // add
            6'b100010: alucontrol <= 3'b110; // sub
            6'b100100: alucontrol <= 3'b000; // and
            6'b100101: alucontrol <= 3'b001; // or
            6'b101010: alucontrol <= 3'b111; // slt
            6'b000110: alucontrol <= 3'b011; // srlv
            default: alucontrol <= 3'bxxx; // ???
        endcase
    endcase
endmodule
```

maindec



```
module maindec(input logic [5:0] op,
    output logic memtoreg, memwrite,
    output logic branch,
    output logic alusrc,
    output logic regdst, regwrite,
    output logic jump,
    output logic ne,
    output logic [2:0] aluop);

logic [10:0] controls;

assign {regwrite, regdst, alusrc, branch, memwrite,
        memtoreg, jump, aluop, ne} = controls;


always_comb
    case(op)
        6'b000000: controls <= 11'b11000000100; // RTYPE
        6'b100011: controls <= 11'b10010010000; // LW
        6'b101011: controls <= 11'b00010100000; // SW
        6'b000100: controls <= 11'b00001000010; // BEQ
        6'b001000: controls <= 11'b10010000000; // ADDI
        6'b001101: controls <= 11'b10110000110; // ORI
        6'b000010: controls <= 11'b00000001000; // J
        6'b000101: controls <= 11'b00001000011;  // BNQ
        6'b001010: controls <= 11'b11010000110; // slti
        6'b000110: controls <= 11'b10000000110; // srlv
        default: controls <= 9'bxxxxxxxxx; // illegal op
    endcase
endmodule
```

alu



```
module alu(input logic [31:0] a,b,
           input logic [2:0] f,
           output logic [31:0] y,
           output logic  zero);


always_comb
        case(f)
        3'b000: y=a&b;
        3'b001: y=a|b;
        3'b010: y=a+b;
        3'b011: y=(a>>b);
        3'b100: y=a+(~b);
        3'b101: y=a|(~b);
        3'b110: y=a-b;
        3'b111:
        begin
                y=a-b;
                y=y[31]?'b1:'b0;
        end
        default: y=0;
        endcase


        assign zero =~(|y);


endmodule
```