# slti

#### aludec

```
1
      module aludec(input logic [5:0] funct,
 2
                      input logic [1:0] aluop,
 3
                      output logic [2:0] alucontrol);
 4
 5
     always comb
 6
          case (aluop)
 7
              2'b00: alucontrol <= 3'b010; // add (for lw/sw/addi)
              2'b01: alucontrol <= 3'b110; // sub (for beq)
 8
              2'b11: alucontrol <= 3'b001;
 9
             2'b10: alucontrol <= 3'b111; //slti
10
11
              default: case(funct) // R-type instructions
12
                  6'b100000: alucontrol <= 3'b010; // add
13
                  6'b100010: alucontrol <= 3'b110; // sub
14
                  6'b100100: alucontrol <= 3'b000; // and
                  6'b100101: alucontrol <= 3'b001; // or
15
16
                  6'b101010: alucontrol <= 3'b111; // slt
17
                  6'b000110: alucontrol <= 3'b011; // srlv
18
                  default: alucontrol <= 3'bxxx; // ???</pre>
19
              endcase
20
          endcase
21
    endmodule
```

### **maindec**

```
module maindec(input logic [5:0] op,
1
2
        output logic memtoreg, memwrite,
 3
          output logic branch,
 4
          output logic alusro,
 5
          output logic regdst, regwrite,
 6
          output logic jump,
          output logic ne,
          output logic [2:0] aluop);
8
9
     logic [10:0] controls;
10
11
12
     assign {regwrite, regdst, alusrc, branch, memwrite,
13
              memtoreg, jump, aluop, ne} = controls;
14
15
16
     always comb
17
          case (op)
              6'b000000: controls <= 11'b11000000100; // RTYPE
18
19
              6'b100011: controls <= 11'b10010010000; // LW
              6'b101011: controls <= 11'b00010100000; // SW
20
              6'b000100: controls <= 11'b00001000010: // BEO
21
              6'b001000: controls <= 11'b10010000000; // ADDI
22
23
              6'b001101: controls <= 11'b10110000110; // ORI
24
              6'b000010: controls <= 11'b00000001000; // J
              6'b000101: controls <= 11'b00001000011; // BNQ
25
26
              6'b001010: controls <= 11'b11010000110:
              6'b000110: controls <= 11'b10000000110: // srlv
27
              default: controls <= 9'bxxxxxxxxx; // illegal op</pre>
28
29
          endcase
30
      endmodule
```

# srlv

## aludec

```
module aludec(input logic [5:0] funct,
                        input logic [1:0] aluop,
                        output logic [2:0] alucontrol);
 5
      always comb
 6
          case (aluop)
               2'b00: alucontrol <= 3'b010; // add (for lw/sw/addi)
2'b01: alucontrol <= 3'b110; // sub (for beq)
 8
               2'b11: alucontrol <= 3'b001;
 9
10
               2'b10: alucontrol <= 3'b111; //slti
               default: case(funct) // R-type instructions
                    6'b100000: alucontrol <= 3'b010; // add
12
                    6'b100010: alucontrol <= 3'b110; // sub
13
14
                    6'b100100: alucontrol <= 3'b000; // and
                    6'b100101: alucontrol <= 3'b001; // or
15
                    6'b101010: alucontrol <= 3'b111; // slt
16
17
18
                    default: alucontrol <= 3'bxxx; // ???</pre>
19
           endcase
20
      endmodule
```

## <u>mainde</u>c

```
module maindec(input logic [5:0] op,
            output logic memtoreg, memwrite,
            output logic branch,
            output logic alusro,
            output logic regdst, regwrite,
            output logic jump,
            output logic ne,
            output logic [2:0] aluop);
       logic [10:0] controls;
11
       assign {regwrite, regdst, alusrc, branch, memwrite,
12
                 memtoreg, jump, aluop, ne} = controls;
14
15
16
       always comb
17
                 6'b000000: controls <= 11'b11000000100; // RTYPE
6'b100011: controls <= 11'b10010010000; // LW
18
19
                 6'b101011: controls <= 11'b00010100000; // SW
                 6'b000100: controls <= 11'b00001000010; // BEQ
21
22
                 6'b001000: controls <= 11'b10010000000; // ADDI
                 6'b001101: controls <= 11'b10110000110; // ORI
                 6'b000010: controls <= 11'b0000001000; // J
6'b000101: controls <= 11'b00001000011; // BNQ
6'b001010: controls <= 11'b11010000110; // slti
24
25
26
27
                 default: controls <= 9'bxxxxxxxxx; // illegal op</pre>
28
29
            endcase
        endmodule
```

## <u>alu</u>

```
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;
10
                3'b001: y=a|b;
11
                3'b010: y=a+b;
12
13
                3'b100: y=a+(~b);
                3'b101: y=a|(~b);
14
15
                3'b110: y=a-b;
16
                3'b111:
17
                begin
18
                         y=a-b;
                         y=y[31]?'b1:'b0;
20
                default: y=0;
21
22
                endcase
23
25
                assign zero =~(|y);
26
       endmodule
```