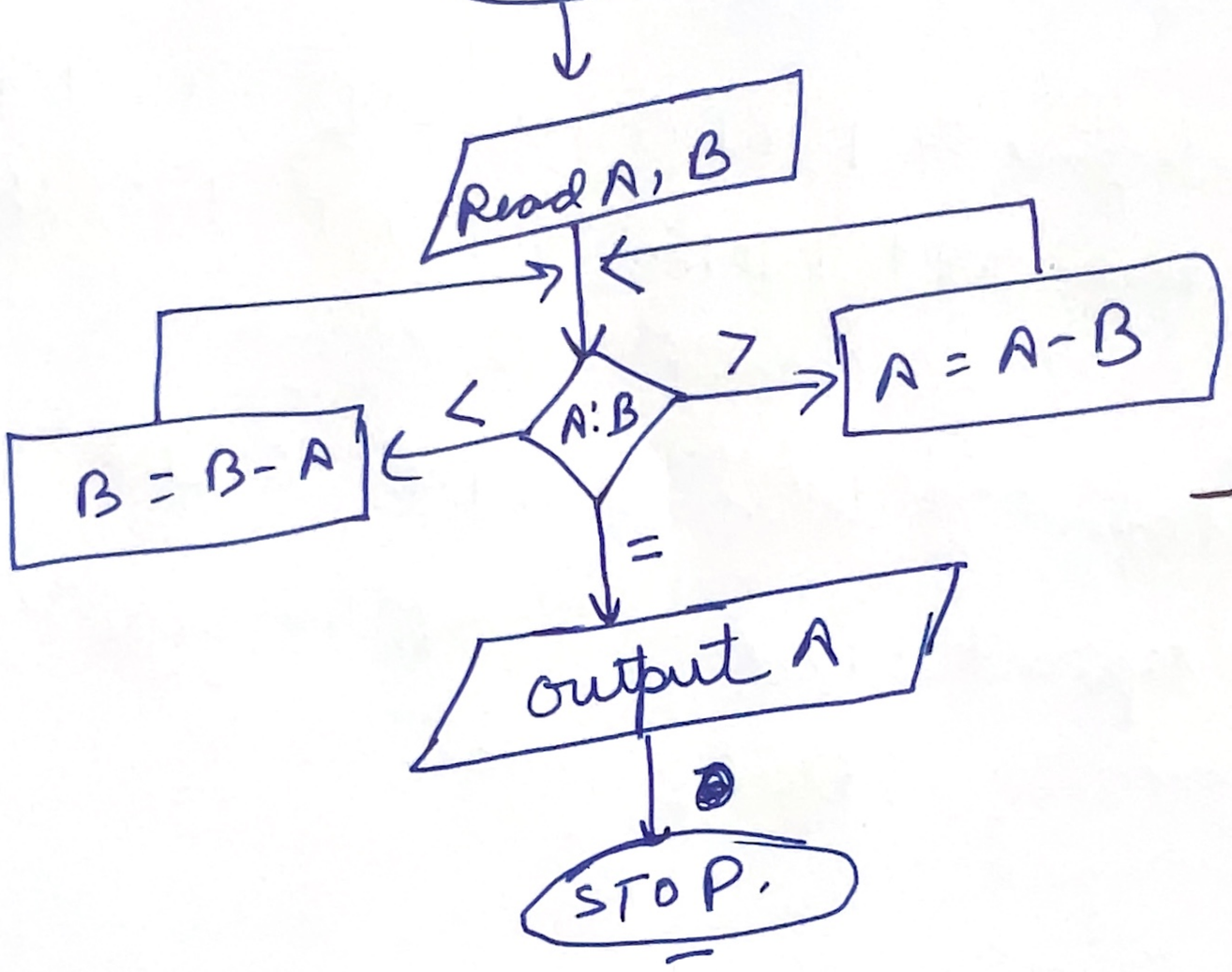
Uc 26

Computing GCD.

26 39

13.

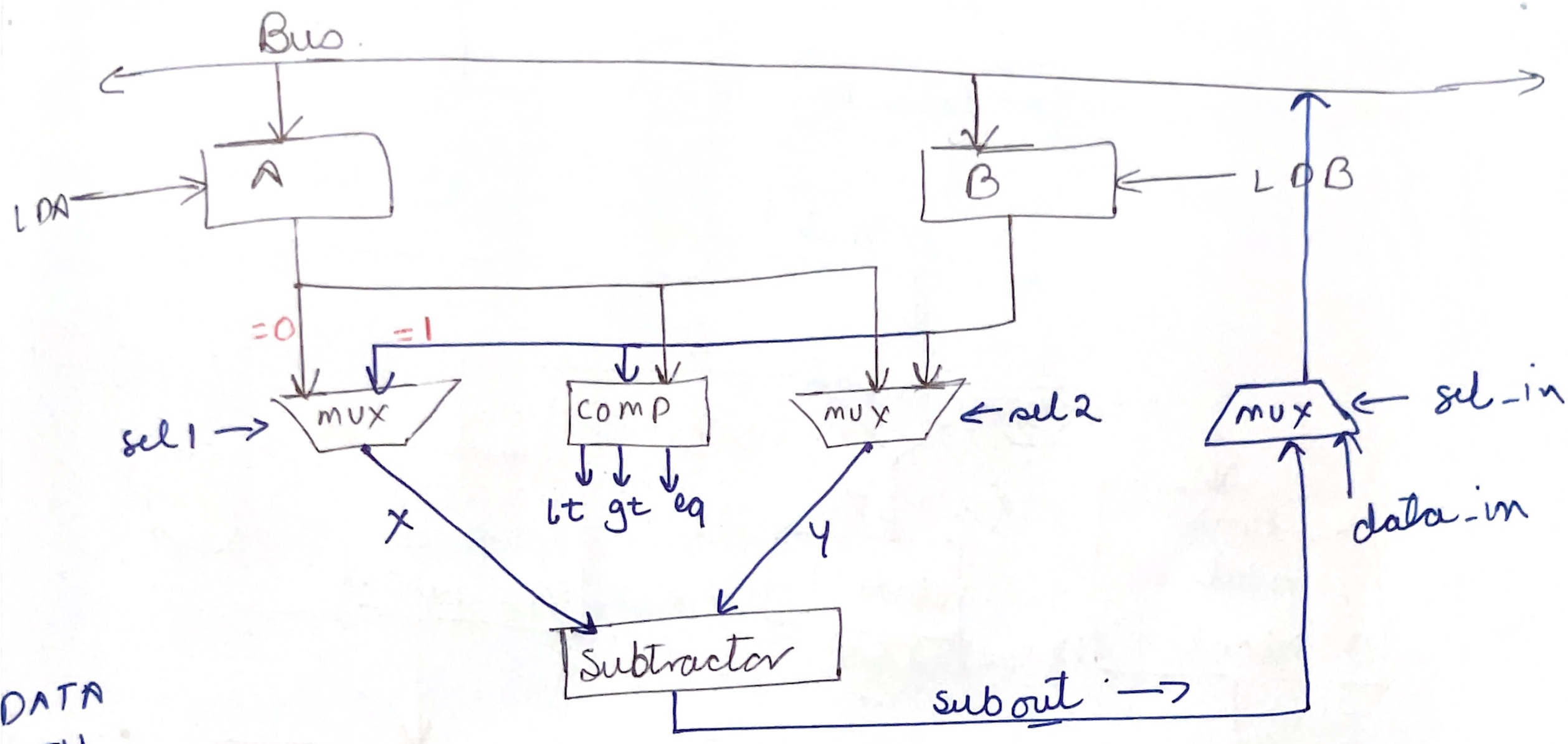


→ 2 Subtractors

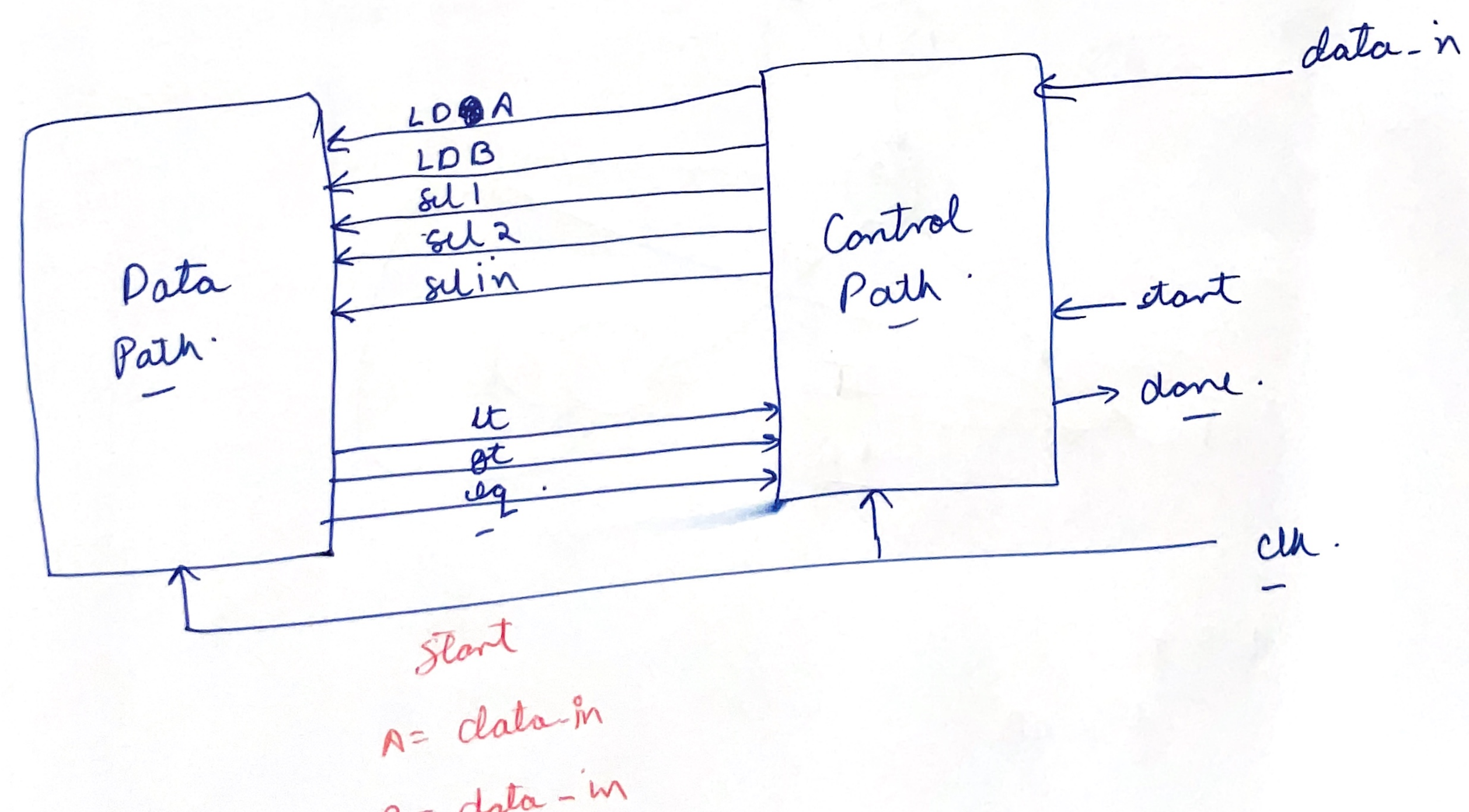
→ Comparator as well

A		B
26	<	65
26	<	39 (65-26)
26	>	13 (39-26)
13	=	13

13 Required GCD.



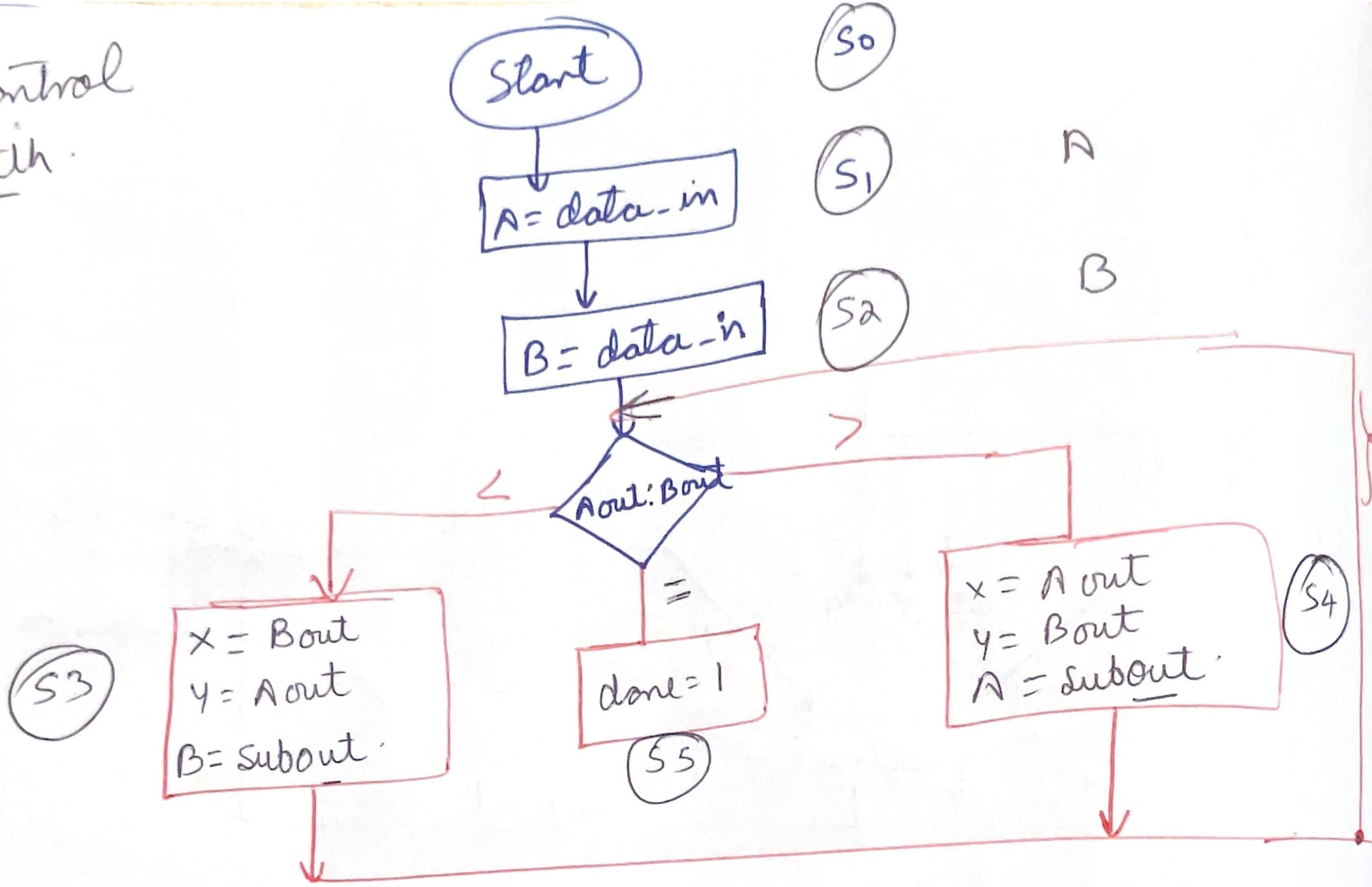
DATA PATH.

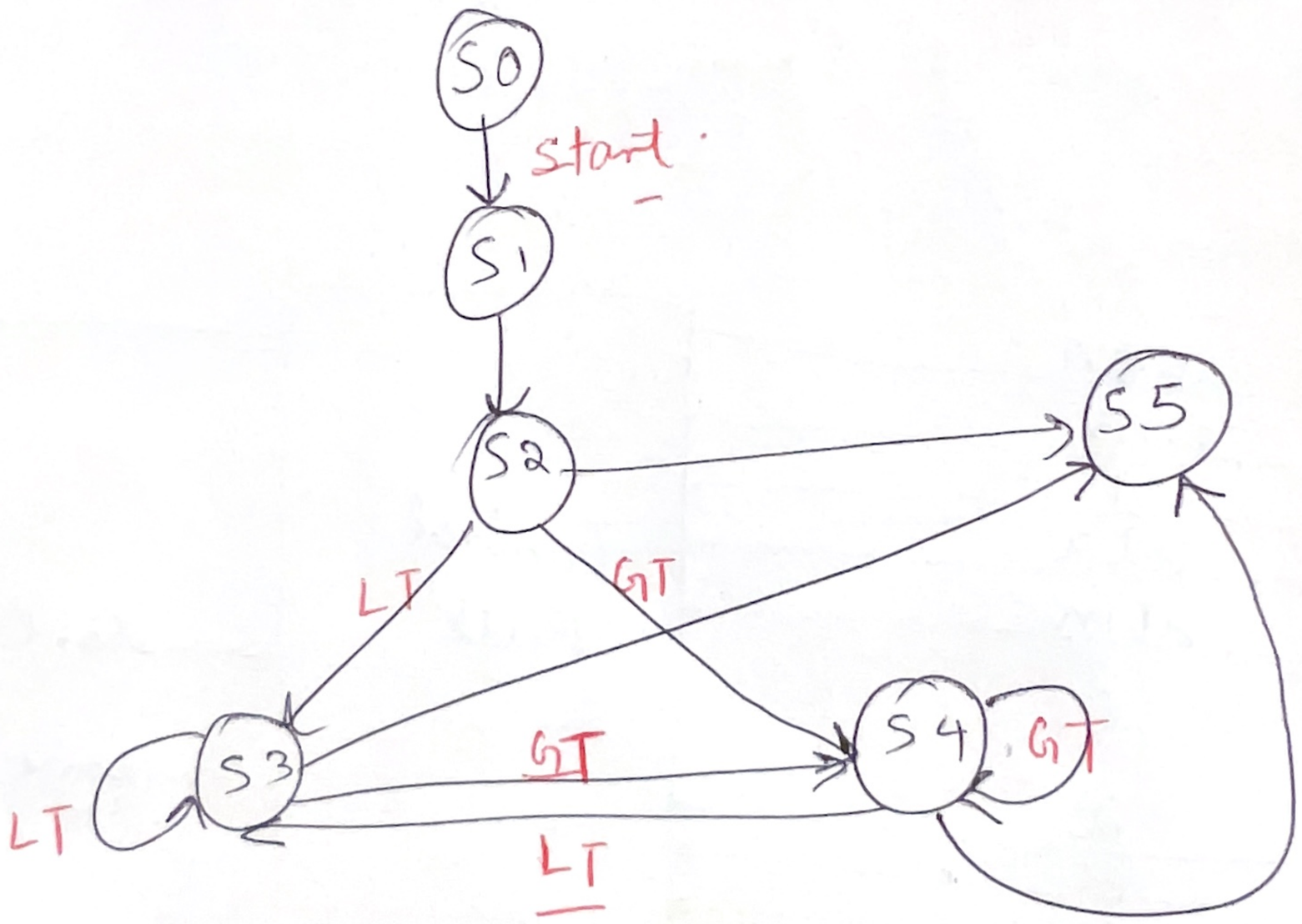


clk Missing

4

Control Path.

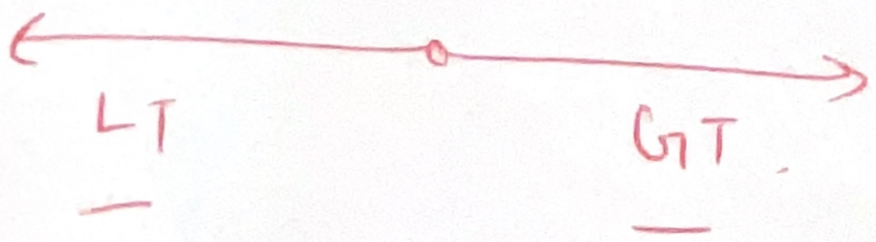




at S4

if A<B

x=B



```
module GCD-datapath( gt, lt, eq, ldA, ldB, sel1, sel2, sel-in,
                     data_in , clk) ;

input ldA, ldB, sel1, sel2, sel-in, clk;
input [15: 0] data_in;
output gt, lt, eq;
wire [15: 0] Aout, Bout, x, y, Bus, subout;   // intermediate wires

PIPO A   (Aout, Bus, ldA, clk);
PIPO B   (Bout, Bus, ldB, clk);
mux      mux_in1( x, Aout, Bout, sel1);
mux      mux_in2( y, Aout, Bout, sel2);
mux      mux_load (Bus, subout, data-in, sel_in);
SUB      SB (subout, x, y);
COMPARE  cmp (lt, gt, eq, Aout, Bout);

endmodule.
```

— // its wiring O/P -> I/P.

Cont -> 12

```
module PIPO (data_out, data_in, load, clk);
    input [15:0] data_in;
    input load, clk;
    output reg [15:0] data_out;
    always @(posedge clk)
        if(load) data_out <= data_in;
```

↑ //A/Bout ↑ //Bus

endmodule.

—

```
module SUB (out, in1, in2);
    input [15:0] in1, in2;
    output reg [15:0] out;
    always @(*)
        out = in1 - in2;
```

endmodule.

—

```
module COMPARE (lt, ge, eq, data1, data2);
    input [15:0] data1, data2;
    output lt, ge, eq;
    assign lt = data1 < data2;
    assign ge = data1 > data2;
    assign eq = data1 = data2;
```

endmodule.

```
module MUX (out, in0, in1, sel);
    input [15:0] in0, in1;
    input sel;
    output [15:0] out;
    assign out = sel ? in1 : in0;
```

endmodule.

↓

true false.

4

```
module controller (ldA, ldB, sel1, sel2, sel_in, done, clk,
                   lt, gt, eq, start);

input clk, lt, gt, eq, start;
output reg ldA, ldB, sel1, sel2, sel_in, done;
reg [2:0] state;
Parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010,
          S3 = 3'b011, S4 = 3'b100, S5 = 3'b101

always @(posedge clk)
    begin
        case (state)
            S0: if (start) state <= S1;
            S1: state <= S2;
            S2: #2 if (eq) state <= S5;
                   else if (lt) state <= S3;
                   else if (gt) state <= S4;

            S3: #2 if (eq)

            S4:

            S5: state <= S5;

            default : state <= S0;

        endcase

    end
```

always @(state)
begin
case (state)
S0: begin sel_in = 1;
ldA = 1;
ldB = 0; done = 0; end

sel_in = 1
gives
data

← I have shown

S1: begin sel_in = 1; ldA = 0; ldB = 1; end
// in S0 I have shown done = 0. like
// S2 S1'

S2: if (eq) done = 1;
else if (lt) begin
sel1 = 1; sel2 = 0; sel_in = 0;
// ldA = 0; ldB = 1;

end

//

else if (gt) begin // a > b
sel1 = 0; sel2 = 1; sel_in = 0;
// ldA = 1; ldB = 0;

end

S3: if (eq) done = 1; A < B
else if (lt) begin
sel1 = 1; sel2 = 0; sel_in = 0;
// ldA = 0; ldB = 1;

end
else if (gt) begin
sel1 = 0; sel2 = 1; sel_in = 0;
// ldA = 1; ldB = 0;

end

Result will go to A.

S4 : ____

```
S5: begin
      clear = 1; sel1 = 0; sel2 = 0; idA = 0;
      idB = 0;
    end
default : begin idA = 0; idB = 0; end

endcase
end
endmodule.
```

TB @ 21:0c

2 alway block Running in 11