**PARTE 1**

1) output Q = 0.

2) output Q = 0,retendo o valor anterior.

3) output Q = 0, retendo o valor anterior.

4) Q = 1, os valores anteriores resetados.

5) Q = 1.

**PARTE 2**

D latch flip flop

SET RESET D CK Q Q`

0 1 - - 1 0

1 0 - - 0 1

0 0 - - 1 1

1 1 1 H 1 0

1 1 0 H 0 1

module parte2 (Clk, R, S, Q);

input Clk R, S;

output Q;

wire R\_g, S\_g, Qa, Qb;

assign R\_g = R & Clk;

assign S\_g = S & Clk;

assign Qa = ~(R\_g | Qb);

assign Qb = ~(S\_g | Qa);

assign Q = Qa;

endmodule

**PARTE 3**

1) 74LS00/03, 74LS04/05

2)

PINOS PARA NAND:

1: S

2: Clk in

3: S\_g

4: Clk in

5: R

6: R\_g

13: S\_g

12: Qb (do pino 8)

11: Qa

10: R\_g

9: Qa (do pino 11)

8: Qb

**PARTE 3**

module parte3 (D, Clk, Q);

input Clk R, S;

output Q;

wire S, R, R\_g, S\_g, Qa, Qb ;

assign R = ~D;

assign R\_g = ~(R & Clk);

assign S\_g = ~(S & Clk);

assign Qa = ~(S\_g & Qb);

assign Qb = ~(R\_g & Qa);

assign Q = Qa;

endmodule

**PARTE 4**

module parte4 (SW, LEDR);

input [1:0]SW;

output [0]LEDR;

assign D = SW[0];

assign Clk = SW[1];

wire S, R, R\_g, S\_g, Qa, Qb;

assign R = ~D;

assign R\_g = ~(R & Clk);

assign S\_g = ~(S & Clk);

assign Qa = ~(S\_g & Qb);

assign Qb = ~(R\_g & Qa);

assign Q = Qa;

assign LEDR[0] = Q;

endmodule

**PARTE 5**

module parte5 (SW, LEDR);

input [1:0]SW;

output [0]LEDR;

assign D = SW[0];

assign Clk = SW[1];

wire Q, S, R, R\_g, S\_g, Qa, Qb

assign R = ~D;

assign R\_g = ~(R & Clk);

assign S\_g = ~(S & Clk);

assign Qa = ~(S\_g & Qb);

assign Qb = ~(R\_g & Qa);

escravo S0 (Qa, Qb, Q);

assign LEDR[0] = Q;

endmodule

module escravo(D, Clk, Q);

input Qa, Qb;

output Q;

wire Q, S, R, R\_g, S\_g, Qa, Qb ;

assign R = ~D;

assign R\_g = ~(R & Clk);

assign S\_g = ~(S & Clk);

assign Qa = ~(S\_g & Qb);

assign Qb = ~(R\_g & Qa);

assign Q = Qa;

endmodule;

**PARTE 6**

module D\_latch(D, Clk, Q);

input D, Clk;

output reg Qa;

always @ (D, Clk);

if (Clk) begin

Qa = D;

end

endmodule

module flip\_flop\_latches(D, Clk, Q);

input D, Clk;

output reg Q;

wire Q\_m;

D\_latch positive (D, Clk, Q\_m);

D\_latch negative (Q\_m, ~Clk, Q);

endmodule

**PARTE 7**

module parte7 (SW, KEY, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7);

input [15:0]SW;

input [1:0]KEY;

output [6:0]HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7;

wire[15:0] Q, Q\_p;

flip\_flip\_latches(SW[0], ~KEY[1], KEY[0], Q[0]);

flip\_flip\_latches(SW[1], ~KEY[1], KEY[0], Q[1]);

flip\_flip\_latches(SW[2], ~KEY[1], KEY[0], Q[2]);

flip\_flip\_latches(SW[3], ~KEY[1], KEY[0], Q[3]);

flip\_flip\_latches(SW[4], ~KEY[1], KEY[0], Q[4]);

flip\_flip\_latches(SW[5], ~KEY[1], KEY[0], Q[5]);

flip\_flip\_latches(SW[6], ~KEY[1], KEY[0], Q[6]);

flip\_flip\_latches(SW[7], ~KEY[1], KEY[0], Q[7]);

flip\_flip\_latches(SW[8], ~KEY[1], KEY[0], Q[8]);

flip\_flip\_latches(SW[9], ~KEY[1], KEY[0], Q[9]);

flip\_flip\_latches(SW[10], ~KEY[1], KEY[0], Q[10]);

flip\_flip\_latches(SW[11], ~KEY[1], KEY[0], Q[11]);

flip\_flip\_latches(SW[12], ~KEY[1], KEY[0], Q[12]);

flip\_flip\_latches(SW[13], ~KEY[1], KEY[0], Q[13]);

flip\_flip\_latches(SW[14], ~KEY[1], KEY[0], Q[14]);

flip\_flip\_latches(SW[15], ~KEY[1], KEY[0], Q[15]);

Q\_p = Q;

char\_7seg d0 (SW[3:0], HEX0);

char\_7seg d1 (SW[7:4], HEX1);

char\_7seg d2 (SW[11:8], HEX2);

char\_7seg d3 (SW[15:12], HEX3);

char\_7seg d4 (Q\_p[3:0], HEX4);

char\_7seg d5 (Q\_p[7:4], HEX5);

char\_7seg d6 (Q\_p[11:8], HEX6);

char\_7seg d7 (Q\_p[15:12], HEX7);

module D\_latch(D, Clk, Res, Q);

input D, Clk, Res;

output reg Qa;

always @ (D, Clk);

if (Clk) begin

Qa = D;

end else if (Res) begin

Qa = 0;

end

endmodule

module flip\_flop\_latches(D, Clk, Q, Res);

input D, Clk, Res;

output reg Q;

wire Q\_m;

D\_latch positive (D, Clk, Res, Q\_m);

D\_latch negative (Q\_m, ~Clk, Res, Q);

endmodule

module char\_7seg(in, out);

output reg [6:0] out;

input [3:0] in;

always @(in)

case (in)

4'h0: out = 7'b0111111;

4'h1: out = 7'b0000110;

4'h2: out = 7'b1011011;

4'h3: out = 7'b1001111;

4'h4: out = 7'b1100110;

4'h5: out = 7'b1101101;

4'h6: out = 7'b1111101;

4'h7: out = 7'b0000111;

4'h8: out = 7'b1111111;

4'h9: out = 7'b1100111;

4'hA: out = 7'b1110111;

4'hB: out = 7'b1111100;

4'hC: out = 7'b0111001;

4'hD: out = 7'b1011110;

4'hE: out = 7'b1111001;

4'hF: out = 7'b1110001;

default: out = 7'b1111001;

endcase

endmodule