**PARTE 1:**

module parte1 (SW, LEDR);

input [17:0]SW;

output [17:0]LEDR;

assign LEDR[17:0] = SW[17:10];

endmodule

**PARTE 2:**

module parte2 (SW, LEDR, LEDG);

input [17:0]SW;

ouput [17:0]LEDR;

output [7:0]LEDG;

wire S;

wire [7:0] X, Y, M;

assign S = SW[17];

assign X = SW[7:0];

assign Y = SW[15:8];

assign LEDR[17:0] = SW[17:0];

assign LEDG[7:0] = M;

assign M[0] = (~S & X[0]) | (S & Y[0]);

assign M[1] = (~S & X[1]) | (S & Y[1]);

assign M[2] = (~S & X[2]) | (S & Y[2]);

assign M[3] = (~S & X[3]) | (S & Y[3]);

assign M[4] = (~S & X[4]) | (S & Y[4]);

assign M[5] = (~S & X[5]) | (S & Y[5]);

assign M[6] = (~S & X[6]) | (S & Y[6]);

assign M[7] = (~S & X[7]) | (S & Y[7]);

endmodule

**PARTE 3:**

module parte3 (SW, LEDR, LEDG);

input [17:0]SW;

output [17:0]LEDR;

output [2:0]LEDG;

wire [2:0] M0, M1, M2, M, U, V, W, X, Y;

wire S0, S1, S2;

assign S0 = SW[17];

assign S1 = SW[16];

assign S2 = SW[15];

assign U = SW[14:12];

assign V = SW[11:9];

assign W = SW[8:6];

assign X = SW[5:3];

assign Y = SW[2:0];

assign LEDR[17:0] = SW[17:0];

assign LEDG[2:0] = M;

//divide os multiplexadores por partes

assign M0[0] = (~S0 & U[0]) | (S0 & V[0]);

assign M0[1] = (~S0 & U[1]) | (S0 & V[1]);

assign M0[2] = (~S0 & U[2]) | (S0 & V[2]);

assign M1[0] = (~S0 & W[0]) | (S0 & X[0]);

assign M1[1] = (~S0 & W[1]) | (S0 & X[1]);

assign M1[2] = (~S0 & W[2]) | (S0 & X[2]);

assign M2[0] = (~S1 & M0[0]) | (S1 & M1[0]);

assign M2[1] = (~S1 & M0[1]) | (S1 & M1[1]);

assign M2[2] = (~S1 & M0[2]) | (S1 & M1[2]);

assign M[0] = (~S2 & M2[0]) | (S2 & Y[0]);

assign M[1] = (~S2 & M2[1]) | (S2 & Y[1]);

assign M[2] = (~S2 & M2[2]) | (S2 & Y[2]);

endmodule

**PARTE 4:**

module parte4 (SW, HEX0);

input [2:0]SW;

output [6:0]HEX0;

wire c0, c1, c2;

assign c0 = SW[0];

assign c1 = SW[1];

assign c2 = SW[2];

assign L = (~c2 & ~c1 & ~c0);

assign O = (~c2 & ~c1 & c0);

assign G = (~c2 & c1 & ~c0);

assign I = (~c2 & c1 & c0);

assign C = (c2 & ~c1 & ~c0);

assign blank = (c2 & ~c1 & c0) | (c2 & c1 & ~c0) | (c2 & c1 & c0);

assign HEX0[5:3] = L;

assign HEX0[6:0] = O;

assign HEX0[6:2] = G;

assign HEX0[0] = G;

assign HEX0[2:1] = I;

assign HEX0[5:3] = C;

assign HEX0[0] = C;

assign ~HEX[6:0] = blank

endmodule

**PARTE 5**

module parte5 (SW, HEX0);

input [17:0] SW; //chaves

output [0:6] HEX0; // display de 7 seg

wire [2:0] M;

mux\_3bit\_5to1 M (SW[17:15], SW[14:12], SW[11:9], SW[8:6], SW[5:3], SW[2:0], M);

char 7seg H0 (M, HEX0);

endmodule

module mux\_3bit\_5to1 (S, U, V, W, X, Y, M);

input [2:0] S, U, V, W, X, Y;

output [2:0] M ;

wire [2:0] M0, M1, M2;

assign M0[0] = (~S[0] & U[0]) | (S[0] & V[0]);

assign M0[1] = (~S[0] & U[1]) | (S[0] & V[1]);

assign M0[2] = (~S[0] & U[2]) | (S[0] & V[2]);

assign M1[0] = (~S[0] & W[0]) | (S[0] & X[0]);

assign M1[1] = (~S[0] & W[1]) | (S[0] & X[1]);

assign M1[2] = (~S[0] & W[2]) | (S[0] & X[2]);

assign M2[0] = (~S[1] & M0[0]) | (S[1] & M1[0]);

assign M2[1] = (~S[1] & M0[1]) | (S[1] & M1[1]);

assign M2[2] = (~S[1] & M0[2]) | (S[1] & M1[2]);

assign M[0] = (~S[2] & M2[0]) | (S[2] & Y[0]);

assign M[1] = (~S[2] & M2[1]) | (S[2] & Y[1]);

assign M[2] = (~S[2] & M2[2]) | (S[2] & Y[2]);

endmodule

module char\_7seg (C, Display);

input [2:0] C;

output [6:0] Display;

assign L = (~C[2] & ~C[1] & ~C[0]);

assign O = (~C[2] & ~C[1] & C[0]);

assign G = (~C[2] & C[1] & ~C[0]);

assign I = (~C[2] & C[1] & C[0]);

assign c = (C[2] & ~C[1] & ~C[0]);

assign blank = (C[2] & ~C[1] & C[0]) | (C[2] & C[1] & ~C[0] | (C[2] & C[1] & C[0]);

assign Display[5:3] = L;

assign Display[6:0] = O;

assign Display[6:2] = G;

assign Display[0] = G;

assign Display[2:1] = I;

assign Display[5:3] = c;

assign Display[0] = c;

assign ~Display[6:0] = blank;

endmodule

**PARTE 6**

module parte6 (SW, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7);

input [17:0] SW;

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

wire[2:0]M0, M1, M2, M3, M4, M5, M6, M7, BL;

assign BL[0] = SW[14]

assign BL[1] = SW[11]

assign BL[2] = SW[8]

mux\_3bit\_8to1 C0(SW[17:15], BL, BL, BL, SW[14:12], SW[11:9], SW[8:6], SW[5:3], SW[2:0], M7);

mux\_3bit\_8to1 C1(SW[17:15], BL, BL, SW[14:12], SW[11:9], SW[8:6], SW[5:3], SW[2:0], BL, M6);

mux\_3bit\_8to1 C2(SW[17:15], BL, SW[14:12], SW[11:9], SW[8:6], SW[5:3], SW[2:0], BL, BL, M5);

mux\_3bit\_8to1 C3(SW[17:15], SW[14:12], SW[11:9], SW[8:6], SW[5:3], SW[2:0], BL, BL, BL, M4);

mux\_3bit\_8to1 C4(SW[17:15], SW[11:9], SW[8:6], SW[5:3], SW[2:0], BL, BL, BL, SW[14:12], M3);

mux\_3bit\_8to1 C5(SW[17:15], SW[8:6], SW[5:3], SW[2:0], BL, BL, BL, SW[14:12], SW[11:9], M2);

mux\_3bit\_8to1 C6(SW[17:15], SW[5:3], SW[2:0], BL, BL, BL, SW[14:12], SW[11:9], SW[8:6], M1);

mux\_3bit\_8to1 C7(SW[17:15], SW[2:0], BL, BL, BL, SW[14:12], SW[11:9], SW[8:6], SW[5:3], M0);

char\_7seg H7(M7, HEX7);

char\_7seg H6(M6, HEX6);

char\_7seg H5(M5, HEX5);

char\_7seg H4(M4, HEX4);

char\_7seg H3(M3, HEX3);

char\_7seg H2(M2, HEX2);

char\_7seg H1(M1, HEX1);

char\_7seg H0(M0, HEX0);

endmodule

module mux\_3bit\_8to1 (S, A, B, C, D, E, F, G, H, O);

input [2:0] S, A, B, C, D, E, F, G, H;

output [2:0] O;

wire [2:0]o0, o1, o2, o3, o4, o5, o6, o7;

assign o0[0] = (~S[0] & A[0]) | (S[0] & B[0]);

assign o0[1] = (~S[1] & A[1]) | (S[1] & B[1]);

assign o0[2] = (~S[2] & A[2]) | (S[2] & B[2]);

assign o1[0] = (~S[0] & C[0]) | (S[0] & D[0]);

assign o1[1] = (~S[1] & C[1]) | (S[1] & D[1]);

assign o1[2] = (~S[2] & C[2]) | (S[2] & D[2]);

assign o2[0] = (~S[0] & E[0]) | (S[0] & F[0]);

assign o2[1] = (~S[1] & E[1]) | (S[1] & F[1]);

assign o2[2] = (~S[2] & E[2]) | (S[2] & F[2]);

assign o3[0] = (~S[0] & G[0]) | (S[0] & H[0]);

assign o3[1] = (~S[1] & G[1]) | (S[1] & H[1]);

assign o3[2] = (~S[2] & G[2]) | (S[2] & H[2]);

assign o4[0] = (~S[0] & o0[0]) | (S[0] & o1[0]);

assign o4[1] = (~S[1] & o0[1]) | (S[1] & o1[1]);

assign o4[2] = (~S[2] & o0[2]) | (S[2] & o1[2]);

assign o5[0] = (~S[0] & o2[0]) | (S[0] & o3[0]);

assign o5[1] = (~S[1] & o2[1]) | (S[1] & o3[1]);

assign o5[2] = (~S[2] & o2[2]) | (S[2] & o3[2]);

assign O[0] = (~S[0] & o4[0]) | (S[0] & o5[0]);

assign O[1] = (~S[1] & o4[1]) | (S[1] & o5[1]);

assign O[2] = (~S[2] & o4[2]) | (S[2] & o5[2]);

endmodule