**PARTE 1**

module parte1(a, b, c, n, sum);

input [n:0]a, b;

input c;

output carry;

output [n-1:0]sum;

assign {carry, sum} = a + b + c;

endmodule;

**PARTE 2**

module parte2(SW, KEY, LEDR);

input [7:0] SW;

input [1:0] KEY;

output [8:0] LEDR;

wire [7:0] ss, c;

reg [7:0] a, b, sum;

reg o;

ss = 8b'00000000;

always @ (posedge KEY1 | KEY0) begin

if (KEY[1] == 0) begin

a = SW[7:0];

b = ss;

sum = ss;

o = c[7] ^ c[6];

end else if (KEY0 == 0) begin

a = 0;

b = 0;

sum = 0;

o = 0;

end

end

somador\_completo a0 (a[0], b[0], 0, ss[0], c[0]);

somador\_completo a1 (a[1], b[1], c[0], ss[1], c[1]);

somador\_completo a2 (a[2], b[2], c[1], ss[2], c[2]);

somador\_completo a3 (a[3], b[3], c[2], ss[3], c[3]);

somador\_completo a3 (a[3], b[3], c[2], ss[3], c[3]);

somador\_completo a4 (a[4], b[4], c[3], ss[4], c[4]);

somador\_completo a5 (a[5], b[5], c[4], ss[5], c[5]);

somador\_completo a6 (a[6], b[6], c[5], ss[6], c[6]);

somador\_completo a7 (a[7], b[7], c[6], ss[7], c[7]);

assign sum[7:0] = ss[7:0];

assign LEDR[7:0] = sum[7:0];

assign LEDR[8] = o;

endmodule;

**PARTE 3**

module parte3(SW, KEY, LEDR);

input [8:0] SW;

input [1:0] KEY;

output [8:0] LEDR;

wire [7:0] ss, c;

reg [7:0] a, b, sum;

reg o;

ss = 8b'00000000;

always @ (posedge KEY1 | KEY0) begin

if (KEY[1] == 0) begin

a = SW[7:0];

b = ss;

sum = ss;

o = c[7] ^ c[6];

end else if (KEY0 == 0) begin

a = 0;

b = 0;

sum = 0;

o = 0;

end

end

somador\_completo a0 (a[0], b[0] ^ SW[8], 0, ss[0], c[0]);

somador\_completo a1 (a[1], b[1] ^ SW[8], c[0], ss[1], c[1]);

somador\_completo a2 (a[2], b[2] ^ SW[8], c[1], ss[2], c[2]);

somador\_completo a3 (a[3], b[3] ^ SW[8], c[2], ss[3], c[3]);

somador\_completo a3 (a[3], b[3] ^ SW[8], c[2], ss[3], c[3]);

somador\_completo a4 (a[4], b[4] ^ SW[8], c[3], ss[4], c[4]);

somador\_completo a5 (a[5], b[5] ^ SW[8], c[4], ss[5], c[5]);

somador\_completo a6 (a[6], b[6] ^ SW[8], c[5], ss[6], c[6]);

somador\_completo a7 (a[7], b[7] ^ SW[8], c[6], ss[7], c[7]);

assign sum[7:0] = ss[7:0];

assign LEDR[7:0] = sum[7:0];

assign LEDR[8] = o;

endmodule;

module somador\_completo(a, b, ci, s, co);

input a, b, ci;

output co, s;

wire d;

assign d = a ^ b;

assign s = d ^ ci;

assign co = (b & ~d) | (d & ci);

endmodule;

**PARTE 4**

module parte4(SW, LEDR, HEX0, HEX1, HEX2, HEX3);

input [7:0]SW;

output [0:6]HEX0, HEX1, HEX2, HEX3;

wire [7:0] x;

multi m(SW[7:4], SW[3:0], x);

char\_7seg h0 (SW[7:4], HEX1);

Char\_7seg h1 (SW[3:0], HEX0);

char\_7seg h2 (x[7:4], HEX3);

char\_7seg h3 (x[3:0], HEX2);

endmodule;

module multi (A, B, x);

input [3:0]A, B;

output [7:0]x;

wire c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, s0, s1, s2, s3, s4, s5;

assign x[0] = A[0] & B[0];

somador\_completo a0(A[1] & B[0], A[0] & B[1], 0, x[1], c0);

somador\_completo a1(A[2] & B[0], A[1] & B[1], c0, s0, c1);

somador\_completo a2(A[3] & B[0], A[2] & B[1], c1, s1, c2);

somador\_completo a3(0, A[3] & B[1], c2, s2, c3);

somador\_completo a4(s0, A[0] & B[2], 0, x[2], c4);

somador\_completo a5(s1, A[1] & B[2], c4, s3, c5);

somador\_completo a6(s2, A[2] & B[2], c5, s4, c6);

somador\_completo a7(c3, A[3] & B[2], c6, s5, c7);

somador\_completo a8(s3, A[0] & B[3], 0, x[3], c8);

somador\_completo a9(s4, A[1] & B[3], c8, x[4], c9);

somador\_completo a10(s5, A[2] & B[3], c9, x[5], c10);

somador\_completo a11(c7, A[3] & B[3], c10, x[6], x[7]);

endmodule;

module somador\_completo (a, b, ci, s, co);

input a, b, ci;

output co, s;

wire d;

assign d = a ^ b;

assign s = d ^ ci;

assign co = (b & ~d) | (d & ci);

endmodule;

module char\_7seg(in, out);

output reg [6:0] out;

input [3:0] in;

always @(in)

case (in)

8'h0: out = 7'b0111111;

8'h1: out = 7'b0000110;

8'h2: out = 7'b1011011;

8'h3: out = 7'b1001111;

8'h4: out = 7'b1100110;

8'h5: out = 7'b1101101;

8'h6: out = 7'b1111101;

8'h7: out = 7'b0000111;

8'h8: out = 7'b1111111;

8'h9: out = 7'b1100111;

8'hA: out = 7'b1110111;

8'hB: out = 7'b1111100;

8'hC: out = 7'b0111001;

8'hD: out = 7'b1011110;

8'hE: out = 7'b1111001;

8'hF: out = 7'b1110001;

default: out = 7'b1111001;

endcase

endmodule;