**Code for Encryption:**

module AES\_encryption(data\_out,data\_in,key);

input [127:0] data\_in,key;

output [127:0] data\_out;

wire [127:0] key1,key2,key3,key4,key5,key6,key7,key8,key9,key10;

wire [127:0] r0,r1,r2,r3,r4,r5,r6,r7,r8,r9;

key\_expansion k1 (key , key1, 4'h1);

key\_expansion k2 (key1, key2, 4'h2);

key\_expansion k3 (key2, key3, 4'h3);

key\_expansion k4 (key3, key4, 4'h4);

key\_expansion k5 (key4, key5, 4'h5);

key\_expansion k6 (key5, key6, 4'h6);

key\_expansion k7 (key6, key7, 4'h7);

key\_expansion k8 (key7, key8, 4'h8);

key\_expansion k9 (key8, key9, 4'h9);

key\_expansion k10 (key9, key10,4'ha);

assign r0 = data\_in ^ key;

round r\_1 (r0, r1, key1);

round r\_2 (r1, r2, key2);

round r\_3 (r2, r3, key3);

round r\_4 (r3, r4, key4);

round r\_5 (r4, r5, key5);

round r\_6 (r5, r6, key6);

round r\_7 (r6, r7, key7);

round r\_8 (r7, r8, key8);

round r\_9 (r8, r9, key9);

last\_round r10 (r9, data\_out, key10);

endmodule

//Key generation module

module key\_expansion(key\_in, key\_out, round);

input [127:0] key\_in;

input [3:0] round;

output [127:0] key\_out;

wire [31:0] w1,w2,w3,w4;

wire [31:0] g\_out;

//key in words

assign w4 = key\_in[31:0];

assign w3 = key\_in[63:32];

assign w2 = key\_in[95:64];

assign w1 = key\_in[127:96];

//subkey generation

gfxn gf(w4,round,g\_out);

assign key\_out[127:96] = g\_out ^ w1;

assign key\_out[95:64] = key\_out[127:96] ^ w2;

assign key\_out[63:32] = key\_out[95:64] ^ w3;

assign key\_out[31:0] = key\_out[63:32] ^ w4;

endmodule

//g-function module

module gfxn(w,round,gout);

input [31:0] w;

input [3:0] round;

output [31:0] gout;

wire [31:0] rotword, subword;

reg [31:0] rconst;

assign rotword = {w[23:0],w[31:24]};

sbox sb1 (rotword[7:0] , subword[7:0]);

sbox sb2 (rotword[15:8] , subword[15:8]);

sbox sb3 (rotword[23:16], subword[23:16]);

sbox sb4 (rotword[31:24], subword[31:24]);

always@(round) begin

case (round)

4'h1: rconst = 32'h01000000;

4'h2: rconst = 32'h02000000;

4'h3: rconst = 32'h04000000;

4'h4: rconst = 32'h08000000;

4'h5: rconst = 32'h10000000;

4'h6: rconst = 32'h20000000;

4'h7: rconst = 32'h40000000;

4'h8: rconst = 32'h80000000;

4'h9: rconst = 32'h1b000000;

4'ha: rconst = 32'h36000000;

endcase

end

assign gout = subword ^ rconst;

endmodule

//round module

module round(data, data\_out, key\_in);

input [127:0] data, key\_in;

output [127:0] data\_out;

wire [127:0] subbyte;

wire [127:0] srow;

wire [127:0] add\_rkey;

//byte substitution

bytesub bs (data, subbyte);

//shifting rows

shiftrow sr (subbyte, srow);

//mixcolumns

mixColumns mc (srow, add\_rkey);

//add round key

assign data\_out = add\_rkey ^ key\_in;

endmodule

//last\_round modile

module last\_round(data, data\_out, key\_in);

input [127:0] data, key\_in;

output [127:0] data\_out;

wire [127:0] subbyte;

wire [127:0] srow;

//byte substitution

bytesub bs (data, subbyte);

//shifting rows

shiftrow sr (subbyte, srow);

//add round key

assign data\_out = srow ^ key\_in;

endmodule

//byte substitution

module bytesub(in, out);

input [127:0] in;

output [127:0] out;

sbox sb1 (in[127:120] , out[127:120]);

sbox sb2 (in[119:112] , out[119:112]);

sbox sb3 (in[111:104] , out[111:104]);

sbox sb4 (in[103:96] , out[103:96]);

sbox sb5 (in[95:88] , out[95:88]);

sbox sb6 (in[87:80] , out[87:80]);

sbox sb7 (in[79:72] , out[79:72]);

sbox sb8 (in[71:64] , out[71:64]);

sbox sb9 (in[63:56] , out[63:56]);

sbox sb10 (in[55:48] , out[55:48]);

sbox sb11 (in[47:40] , out[47:40]);

sbox sb12 (in[39:32] , out[39:32]);

sbox sb13 (in[31:24] , out[31:24]);

sbox sb14 (in[23:16] , out[23:16]);

sbox sb15 (in[15:8] , out[15:8]);

sbox sb16 (in[7:0] , out[7:0]);

endmodule

//shiftrows module

module shiftrow(in, out);

input [127:0] in;

output [127:0] out;

assign out[127:120] = in[127:120];

assign out[119:112] = in[87:80];

assign out[111:104] = in[47:40];

assign out[103:96] = in[7:0];

assign out[95:88] = in[95:88];

assign out[87:80] = in[55:48];

assign out[79:72] = in[15:8];

assign out[71:64] = in[103:96];

assign out[63:56] = in[63:56];

assign out[55:48] = in[23:16];

assign out[47:40] = in[111:104];

assign out[39:32] = in[71:64];

assign out[31:24] = in[31:24];

assign out[23:16] = in[119:112];

assign out[15:8] = in[79:72];

assign out[7:0] = in[39:32];

endmodule

//mix column module

module mixColumns(in,out);

input [127:0] in;

output reg [127:0] out;

function [7:0] mul\_2; //multiply by 2

input [7:0] a;

begin

if(a[7] == 1) mul\_2 = ((a << 1) ^ 8'h1b);

else mul\_2 = a << 1;

end

endfunction

function [7:0] mul\_3; //multiply by 3

input [7:0] b;

begin

mul\_3 = mul\_2(b) ^ b;

end

endfunction

genvar i;

generate

for(i=3; i>=0 ; i=i-1) begin

assign out[(i\*32 + 24)+:8]= mul\_2(in[(i\*32 + 24)+:8]) ^ mul\_3(in[(i\*32 + 16)+:8]) ^ in[(i\*32 + 8)+:8] ^ in[i\*32+:8];

assign out[(i\*32 + 16)+:8]= in[(i\*32 + 24)+:8] ^ mul\_2(in[(i\*32 + 16)+:8]) ^ mul\_3(in[(i\*32 + 8)+:8]) ^ in[i\*32+:8];

assign out[(i\*32 + 8)+:8]= in[(i\*32 + 24)+:8] ^ in[(i\*32 + 16)+:8] ^ mul\_2(in[(i\*32 + 8)+:8]) ^ mul\_3(in[i\*32+:8]);

assign out[i\*32+:8]= mul\_3(in[(i\*32 + 24)+:8]) ^ in[(i\*32 + 16)+:8] ^ in[(i\*32 + 8)+:8] ^ mul\_2(in[i\*32+:8]);

end

endgenerate

endmodule

//s-box module

module sbox(in,c);

input [7:0] in;

output reg [7:0] c;

always@(in) begin

case (in)

8'h00: c=8'h63;

8'h01: c=8'h7c;

8'h02: c=8'h77;

8'h03: c=8'h7b;

8'h04: c=8'hf2;

8'h05: c=8'h6b;

8'h06: c=8'h6f;

8'h07: c=8'hc5;

8'h08: c=8'h30;

8'h09: c=8'h01;

8'h0a: c=8'h67;

8'h0b: c=8'h2b;

8'h0c: c=8'hfe;

8'h0d: c=8'hd7;

8'h0e: c=8'hab;

8'h0f: c=8'h76;

8'h10: c=8'hca;

8'h11: c=8'h82;

8'h12: c=8'hc9;

8'h13: c=8'h7d;

8'h14: c=8'hfa;

8'h15: c=8'h59;

8'h16: c=8'h47;

8'h17: c=8'hf0;

8'h18: c=8'had;

8'h19: c=8'hd4;

8'h1a: c=8'ha2;

8'h1b: c=8'haf;

8'h1c: c=8'h9c;

8'h1d: c=8'ha4;

8'h1e: c=8'h72;

8'h1f: c=8'hc0;

8'h20: c=8'hb7;

8'h21: c=8'hfd;

8'h22: c=8'h93;

8'h23: c=8'h26;

8'h24: c=8'h36;

8'h25: c=8'h3f;

8'h26: c=8'hf7;

8'h27: c=8'hcc;

8'h28: c=8'h34;

8'h29: c=8'ha5;

8'h2a: c=8'he5;

8'h2b: c=8'hf1;

8'h2c: c=8'h71;

8'h2d: c=8'hd8;

8'h2e: c=8'h31;

8'h2f: c=8'h15;

8'h30: c=8'h04;

8'h31: c=8'hc7;

8'h32: c=8'h23;

8'h33: c=8'hc3;

8'h34: c=8'h18;

8'h35: c=8'h96;

8'h36: c=8'h05;

8'h37: c=8'h9a;

8'h38: c=8'h07;

8'h39: c=8'h12;

8'h3a: c=8'h80;

8'h3b: c=8'he2;

8'h3c: c=8'heb;

8'h3d: c=8'h27;

8'h3e: c=8'hb2;

8'h3f: c=8'h75;

8'h40: c=8'h09;

8'h41: c=8'h83;

8'h42: c=8'h2c;

8'h43: c=8'h1a;

8'h44: c=8'h1b;

8'h45: c=8'h6e;

8'h46: c=8'h5a;

8'h47: c=8'ha0;

8'h48: c=8'h52;

8'h49: c=8'h3b;

8'h4a: c=8'hd6;

8'h4b: c=8'hb3;

8'h4c: c=8'h29;

8'h4d: c=8'he3;

8'h4e: c=8'h2f;

8'h4f: c=8'h84;

8'h50: c=8'h53;

8'h51: c=8'hd1;

8'h52: c=8'h00;

8'h53: c=8'hed;

8'h54: c=8'h20;

8'h55: c=8'hfc;

8'h56: c=8'hb1;

8'h57: c=8'h5b;

8'h58: c=8'h6a;

8'h59: c=8'hcb;

8'h5a: c=8'hbe;

8'h5b: c=8'h39;

8'h5c: c=8'h4a;

8'h5d: c=8'h4c;

8'h5e: c=8'h58;

8'h5f: c=8'hcf;

8'h60: c=8'hd0;

8'h61: c=8'hef;

8'h62: c=8'haa;

8'h63: c=8'hfb;

8'h64: c=8'h43;

8'h65: c=8'h4d;

8'h66: c=8'h33;

8'h67: c=8'h85;

8'h68: c=8'h45;

8'h69: c=8'hf9;

8'h6a: c=8'h02;

8'h6b: c=8'h7f;

8'h6c: c=8'h50;

8'h6d: c=8'h3c;

8'h6e: c=8'h9f;

8'h6f: c=8'ha8;

8'h70: c=8'h51;

8'h71: c=8'ha3;

8'h72: c=8'h40;

8'h73: c=8'h8f;

8'h74: c=8'h92;

8'h75: c=8'h9d;

8'h76: c=8'h38;

8'h77: c=8'hf5;

8'h78: c=8'hbc;

8'h79: c=8'hb6;

8'h7a: c=8'hda;

8'h7b: c=8'h21;

8'h7c: c=8'h10;

8'h7d: c=8'hff;

8'h7e: c=8'hf3;

8'h7f: c=8'hd2;

8'h80: c=8'hcd;

8'h81: c=8'h0c;

8'h82: c=8'h13;

8'h83: c=8'hec;

8'h84: c=8'h5f;

8'h85: c=8'h97;

8'h86: c=8'h44;

8'h87: c=8'h17;

8'h88: c=8'hc4;

8'h89: c=8'ha7;

8'h8a: c=8'h7e;

8'h8b: c=8'h3d;

8'h8c: c=8'h64;

8'h8d: c=8'h5d;

8'h8e: c=8'h19;

8'h8f: c=8'h73;

8'h90: c=8'h60;

8'h91: c=8'h81;

8'h92: c=8'h4f;

8'h93: c=8'hdc;

8'h94: c=8'h22;

8'h95: c=8'h2a;

8'h96: c=8'h90;

8'h97: c=8'h88;

8'h98: c=8'h46;

8'h99: c=8'hee;

8'h9a: c=8'hb8;

8'h9b: c=8'h14;

8'h9c: c=8'hde;

8'h9d: c=8'h5e;

8'h9e: c=8'h0b;

8'h9f: c=8'hdb;

8'ha0: c=8'he0;

8'ha1: c=8'h32;

8'ha2: c=8'h3a;

8'ha3: c=8'h0a;

8'ha4: c=8'h49;

8'ha5: c=8'h06;

8'ha6: c=8'h24;

8'ha7: c=8'h5c;

8'ha8: c=8'hc2;

8'ha9: c=8'hd3;

8'haa: c=8'hac;

8'hab: c=8'h62;

8'hac: c=8'h91;

8'had: c=8'h95;

8'hae: c=8'he4;

8'haf: c=8'h79;

8'hb0: c=8'he7;

8'hb1: c=8'hc8;

8'hb2: c=8'h37;

8'hb3: c=8'h6d;

8'hb4: c=8'h8d;

8'hb5: c=8'hd5;

8'hb6: c=8'h4e;

8'hb7: c=8'ha9;

8'hb8: c=8'h6c;

8'hb9: c=8'h56;

8'hba: c=8'hf4;

8'hbb: c=8'hea;

8'hbc: c=8'h65;

8'hbd: c=8'h7a;

8'hbe: c=8'hae;

8'hbf: c=8'h08;

8'hc0: c=8'hba;

8'hc1: c=8'h78;

8'hc2: c=8'h25;

8'hc3: c=8'h2e;

8'hc4: c=8'h1c;

8'hc5: c=8'ha6;

8'hc6: c=8'hb4;

8'hc7: c=8'hc6;

8'hc8: c=8'he8;

8'hc9: c=8'hdd;

8'hca: c=8'h74;

8'hcb: c=8'h1f;

8'hcc: c=8'h4b;

8'hcd: c=8'hbd;

8'hce: c=8'h8b;

8'hcf: c=8'h8a;

8'hd0: c=8'h70;

8'hd1: c=8'h3e;

8'hd2: c=8'hb5;

8'hd3: c=8'h66;

8'hd4: c=8'h48;

8'hd5: c=8'h03;

8'hd6: c=8'hf6;

8'hd7: c=8'h0e;

8'hd8: c=8'h61;

8'hd9: c=8'h35;

8'hda: c=8'h57;

8'hdb: c=8'hb9;

8'hdc: c=8'h86;

8'hdd: c=8'hc1;

8'hde: c=8'h1d;

8'hdf: c=8'h9e;

8'he0: c=8'he1;

8'he1: c=8'hf8;

8'he2: c=8'h98;

8'he3: c=8'h11;

8'he4: c=8'h69;

8'he5: c=8'hd9;

8'he6: c=8'h8e;

8'he7: c=8'h94;

8'he8: c=8'h9b;

8'he9: c=8'h1e;

8'hea: c=8'h87;

8'heb: c=8'he9;

8'hec: c=8'hce;

8'hed: c=8'h55;

8'hee: c=8'h28;

8'hef: c=8'hdf;

8'hf0: c=8'h8c;

8'hf1: c=8'ha1;

8'hf2: c=8'h89;

8'hf3: c=8'h0d;

8'hf4: c=8'hbf;

8'hf5: c=8'he6;

8'hf6: c=8'h42;

8'hf7: c=8'h68;

8'hf8: c=8'h41;

8'hf9: c=8'h99;

8'hfa: c=8'h2d;

8'hfb: c=8'h0f;

8'hfc: c=8'hb0;

8'hfd: c=8'h54;

8'hfe: c=8'hbb;

8'hff: c=8'h16;

endcase

end

Endmodule

**Code for Decryption:**

module AES\_decryption(data\_out, data\_in, key);

input [127:0] data\_in, key;

output [127:0] data\_out;

wire [127:0] key1, key2, key3, key4, key5, key6, key7, key8, key9, key10;

wire [127:0] r0, r1, r2, r3, r4, r5, r6, r7, r8, r9;

key\_expansion k1 (key , key1, 4'h1);

key\_expansion k2 (key1, key2, 4'h2);

key\_expansion k3 (key2, key3, 4'h3);

key\_expansion k4 (key3, key4, 4'h4);

key\_expansion k5 (key4, key5, 4'h5);

key\_expansion k6 (key5, key6, 4'h6);

key\_expansion k7 (key6, key7, 4'h7);

key\_expansion k8 (key7, key8, 4'h8);

key\_expansion k9 (key8, key9, 4'h9);

key\_expansion k10 (key9, key10,4'ha);

assign r0 = data\_in ^ key10;

round r\_1 (r0, r1, key9);

round r\_2 (r1, r2, key8);

round r\_3 (r2, r3, key7);

round r\_4 (r3, r4, key6);

round r\_5 (r4, r5, key5);

round r\_6 (r5, r6, key4);

round r\_7 (r6, r7, key3);

round r\_8 (r7, r8, key2);

round r\_9 (r8, r9, key1);

last\_round r\_10 (r9, data\_out, key);

endmodule

//keyexpansion same as encrytion

module key\_expansion(key\_in, key\_out, round);

input [127:0] key\_in;

input [3:0] round;

output [127:0] key\_out;

wire [31:0] w1,w2,w3,w4;

wire [31:0] g\_out;

//key in words

assign w4 = key\_in[31:0];

assign w3 = key\_in[63:32];

assign w2 = key\_in[95:64];

assign w1 = key\_in[127:96];

//subkey generation

gfxn gf(w4,round,g\_out);

assign key\_out[127:96] = g\_out ^ w1;

assign key\_out[95:64] = key\_out[127:96] ^ w2;

assign key\_out[63:32] = key\_out[95:64] ^ w3;

assign key\_out[31:0] = key\_out[63:32] ^ w4;

endmodule

module gfxn(w,round,gout);

input [31:0] w;

input [3:0] round;

output [31:0] gout;

wire [31:0] rotword, subword;

reg [31:0] rconst;

assign rotword = {w[23:0],w[31:24]};

sbox sb1 (rotword[7:0] , subword[7:0]);

sbox sb2 (rotword[15:8] , subword[15:8]);

sbox sb3 (rotword[23:16], subword[23:16]);

sbox sb4 (rotword[31:24], subword[31:24]);

always@(round) begin

case (round)

4'h1: rconst = 32'h01000000;

4'h2: rconst = 32'h02000000;

4'h3: rconst = 32'h04000000;

4'h4: rconst = 32'h08000000;

4'h5: rconst = 32'h10000000;

4'h6: rconst = 32'h20000000;

4'h7: rconst = 32'h40000000;

4'h8: rconst = 32'h80000000;

4'h9: rconst = 32'h1b000000;

4'ha: rconst = 32'h36000000;

endcase

end

assign gout = subword ^ rconst;

endmodule

module addroundkey(out,data,key);

input [127:0] data;

input [127:0] key;

output [127:0] out;

assign out = key ^ data;

endmodule

module round(data, data\_out, key\_in);

input [127:0] data, key\_in;

output [127:0] data\_out;

wire [127:0] subbyte;

wire [127:0] srow;

wire [127:0] add\_rkey;

inv\_shiftrow sr (data, srow);

inv\_bytesub bs (srow, subbyte);

addroundkey ak (add\_rkey,subbyte,key\_in);

inv\_mixColumns mc (add\_rkey,data\_out);

endmodule

module last\_round(data, data\_out,key\_in);

input [127:0] data, key\_in;

output [127:0] data\_out;

wire [127:0] isrow;

wire [127:0] isubbyte;

inv\_shiftrow isr (data, isrow);

inv\_bytesub ibs (isrow, isubbyte);

addroundkey ak1 (data\_out,isubbyte,key\_in);

endmodule

module inv\_bytesub(in, out);

input [127:0] in;

output [127:0] out;

sbox\_inv sb1 (in[127:120] , out[127:120]);

sbox\_inv sb2 (in[119:112] , out[119:112]);

sbox\_inv sb3 (in[111:104] , out[111:104]);

sbox\_inv sb4 (in[103:96] , out[103:96]);

sbox\_inv sb5 (in[95:88] , out[95:88]);

sbox\_inv sb6 (in[87:80] , out[87:80]);

sbox\_inv sb7 (in[79:72] , out[79:72]);

sbox\_inv sb8 (in[71:64] , out[71:64]);

sbox\_inv sb9 (in[63:56] , out[63:56]);

sbox\_inv sb10 (in[55:48] , out[55:48]);

sbox\_inv sb11 (in[47:40] , out[47:40]);

sbox\_inv sb12 (in[39:32] , out[39:32]);

sbox\_inv sb13 (in[31:24] , out[31:24]);

sbox\_inv sb14 (in[23:16] , out[23:16]);

sbox\_inv sb15 (in[15:8] , out[15:8]);

sbox\_inv sb16 (in[7:0] , out[7:0]);

endmodule

// Inverse Shift Rows

module inv\_shiftrow(in,out);

input [0:127] in;

output [0:127] out;

assign out[0:7] = in[0:7];

assign out[32:39] = in[32:39];

assign out[64:71] = in[64:71];

assign out[96:103] = in[96:103];

assign out[8:15] = in[104:111];

assign out[40:47] = in[8:15];

assign out[72:79] = in[40:47];

assign out[104:111] = in[72:79];

assign out[16:23] = in[80:87];

assign out[48:55] = in[112:119];

assign out[80:87] = in[16:23];

assign out[112:119] = in[48:55];

assign out[24:31] = in[56:63];

assign out[56:63] = in[88:95];

assign out[88:95] = in[120:127];

assign out[120:127] = in[24:31];

endmodule

//inverse mix columns

module inv\_mixColumns(state\_in,state\_out);

input [127:0] state\_in;

output [127:0] state\_out;

function[7:0] multiply(input [7:0]a,input integer n);

integer i;

begin

for(i=0;i<n;i=i+1)begin

if(a[7] == 1) a = ((a<< 1) ^ 8'h1b);

else a = a << 1;

end

multiply=a;

end

endfunction

function [7:0] mul\_0e;

input [7:0] a;

begin

mul\_0e=multiply(a,3) ^ multiply(a,2)^ multiply(a,1);

end

endfunction

function [7:0] mul\_0d;

input [7:0] a;

begin

mul\_0d=multiply(a,3) ^ multiply(a,2)^a;

end

endfunction

function [7:0] mul\_0b;

input [7:0] a;

begin

mul\_0b=multiply(a,3) ^ multiply(a,1)^a;

end

endfunction

function [7:0] mul\_09;

input [7:0] a;

begin

mul\_09=multiply(a,3) ^a;

end

endfunction

genvar i;

generate

for(i=0;i< 4;i=i+1) begin

assign state\_out[(i\*32 + 24)+:8]= mul\_0e(state\_in[(i\*32 + 24)+:8]) ^ mul\_0b(state\_in[(i\*32 + 16)+:8]) ^ mul\_0d(state\_in[(i\*32 + 8)+:8]) ^ mul\_09(state\_in[i\*32+:8]);

assign state\_out[(i\*32 + 16)+:8]= mul\_09(state\_in[(i\*32 + 24)+:8]) ^ mul\_0e(state\_in[(i\*32 + 16)+:8]) ^ mul\_0b(state\_in[(i\*32 + 8)+:8]) ^ mul\_0d(state\_in[i\*32+:8]);

assign state\_out[(i\*32 + 8)+:8]= mul\_0d(state\_in[(i\*32 + 24)+:8]) ^ mul\_09(state\_in[(i\*32 + 16)+:8]) ^ mul\_0e(state\_in[(i\*32 + 8)+:8]) ^ mul\_0b(state\_in[i\*32+:8]);

assign state\_out[i\*32+:8]= mul\_0b(state\_in[(i\*32 + 24)+:8]) ^ mul\_0d(state\_in[(i\*32 + 16)+:8]) ^ mul\_09(state\_in[(i\*32 + 8)+:8]) ^ mul\_0e(state\_in[i\*32+:8]);

end

endgenerate

endmodule

module sbox\_inv(in,sbout);

input [7:0] in;

output reg [7:0] sbout;

always@(in) begin

case (in)

8'h00:sbout =8'h52;

8'h01:sbout =8'h09;

8'h02:sbout =8'h6a;

8'h03:sbout =8'hd5;

8'h04:sbout =8'h30;

8'h05:sbout =8'h36;

8'h06:sbout =8'ha5;

8'h07:sbout =8'h38;

8'h08:sbout =8'hbf;

8'h09:sbout =8'h40;

8'h0a:sbout =8'ha3;

8'h0b:sbout =8'h9e;

8'h0c:sbout =8'h81;

8'h0d:sbout =8'hf3;

8'h0e:sbout =8'hd7;

8'h0f:sbout =8'hfb;

8'h10:sbout =8'h7c;

8'h11:sbout =8'he3;

8'h12:sbout =8'h39;

8'h13:sbout =8'h82;

8'h14:sbout =8'h9b;

8'h15:sbout =8'h2f;

8'h16:sbout =8'hff;

8'h17:sbout =8'h87;

8'h18:sbout =8'h34;

8'h19:sbout =8'h8e;

8'h1a:sbout =8'h43;

8'h1b:sbout =8'h44;

8'h1c:sbout =8'hc4;

8'h1d:sbout =8'hde;

8'h1e:sbout =8'he9;

8'h1f:sbout =8'hcb;

8'h20:sbout =8'h54;

8'h21:sbout =8'h7b;

8'h22:sbout =8'h94;

8'h23:sbout =8'h32;

8'h24:sbout =8'ha6;

8'h25:sbout =8'hc2;

8'h26:sbout =8'h23;

8'h27:sbout =8'h3d;

8'h28:sbout =8'hee;

8'h29:sbout =8'h4c;

8'h2a:sbout =8'h95;

8'h2b:sbout =8'h0b;

8'h2c:sbout =8'h42;

8'h2d:sbout =8'hfa;

8'h2e:sbout =8'hc3;

8'h2f:sbout =8'h4e;

8'h30:sbout =8'h08;

8'h31:sbout =8'h2e;

8'h32:sbout =8'ha1;

8'h33:sbout =8'h66;

8'h34:sbout =8'h28;

8'h35:sbout =8'hd9;

8'h36:sbout =8'h24;

8'h37:sbout =8'hb2;

8'h38:sbout =8'h76;

8'h39:sbout =8'h5b;

8'h3a:sbout =8'ha2;

8'h3b:sbout =8'h49;

8'h3c:sbout =8'h6d;

8'h3d:sbout =8'h8b;

8'h3e:sbout =8'hd1;

8'h3f:sbout =8'h25;

8'h40:sbout =8'h72;

8'h41:sbout =8'hf8;

8'h42:sbout =8'hf6;

8'h43:sbout =8'h64;

8'h44:sbout =8'h86;

8'h45:sbout =8'h68;

8'h46:sbout =8'h98;

8'h47:sbout =8'h16;

8'h48:sbout =8'hd4;

8'h49:sbout =8'ha4;

8'h4a:sbout =8'h5c;

8'h4b:sbout =8'hcc;

8'h4c:sbout =8'h5d;

8'h4d:sbout =8'h65;

8'h4e:sbout =8'hb6;

8'h4f:sbout =8'h92;

8'h50:sbout =8'h6c;

8'h51:sbout =8'h70;

8'h52:sbout =8'h48;

8'h53:sbout =8'h50;

8'h54:sbout =8'hfd;

8'h55:sbout =8'hed;

8'h56:sbout =8'hb9;

8'h57:sbout =8'hda;

8'h58:sbout =8'h5e;

8'h59:sbout =8'h15;

8'h5a:sbout =8'h46;

8'h5b:sbout =8'h57;

8'h5c:sbout =8'ha7;

8'h5d:sbout =8'h8d;

8'h5e:sbout =8'h9d;

8'h5f:sbout =8'h84;

8'h60:sbout =8'h90;

8'h61:sbout =8'hd8;

8'h62:sbout =8'hab;

8'h63:sbout =8'h00;

8'h64:sbout =8'h8c;

8'h65:sbout =8'hbc;

8'h66:sbout =8'hd3;

8'h67:sbout =8'h0a;

8'h68:sbout =8'hf7;

8'h69:sbout =8'he4;

8'h6a:sbout =8'h58;

8'h6b:sbout =8'h05;

8'h6c:sbout =8'hb8;

8'h6d:sbout =8'hb3;

8'h6e:sbout =8'h45;

8'h6f:sbout =8'h06;

8'h70:sbout =8'hd0;

8'h71:sbout =8'h2c;

8'h72:sbout =8'h1e;

8'h73:sbout =8'h8f;

8'h74:sbout =8'hca;

8'h75:sbout =8'h3f;

8'h76:sbout =8'h0f;

8'h77:sbout =8'h02;

8'h78:sbout =8'hc1;

8'h79:sbout =8'haf;

8'h7a:sbout =8'hbd;

8'h7b:sbout =8'h03;

8'h7c:sbout =8'h01;

8'h7d:sbout =8'h13;

8'h7e:sbout =8'h8a;

8'h7f:sbout =8'h6b;

8'h80:sbout =8'h3a;

8'h81:sbout =8'h91;

8'h82:sbout =8'h11;

8'h83:sbout =8'h41;

8'h84:sbout =8'h4f;

8'h85:sbout =8'h67;

8'h86:sbout =8'hdc;

8'h87:sbout =8'hea;

8'h88:sbout =8'h97;

8'h89:sbout =8'hf2;

8'h8a:sbout =8'hcf;

8'h8b:sbout =8'hce;

8'h8c:sbout =8'hf0;

8'h8d:sbout =8'hb4;

8'h8e:sbout =8'he6;

8'h8f:sbout =8'h73;

8'h90:sbout =8'h96;

8'h91:sbout =8'hac;

8'h92:sbout =8'h74;

8'h93:sbout =8'h22;

8'h94:sbout =8'he7;

8'h95:sbout =8'had;

8'h96:sbout =8'h35;

8'h97:sbout =8'h85;

8'h98:sbout =8'he2;

8'h99:sbout =8'hf9;

8'h9a:sbout =8'h37;

8'h9b:sbout =8'he8;

8'h9c:sbout =8'h1c;

8'h9d:sbout =8'h75;

8'h9e:sbout =8'hdf;

8'h9f:sbout =8'h6e;

8'ha0:sbout =8'h47;

8'ha1:sbout =8'hf1;

8'ha2:sbout =8'h1a;

8'ha3:sbout =8'h71;

8'ha4:sbout =8'h1d;

8'ha5:sbout =8'h29;

8'ha6:sbout =8'hc5;

8'ha7:sbout =8'h89;

8'ha8:sbout =8'h6f;

8'ha9:sbout =8'hb7;

8'haa:sbout =8'h62;

8'hab:sbout =8'h0e;

8'hac:sbout =8'haa;

8'had:sbout =8'h18;

8'hae:sbout =8'hbe;

8'haf:sbout =8'h1b;

8'hb0:sbout =8'hfc;

8'hb1:sbout =8'h56;

8'hb2:sbout =8'h3e;

8'hb3:sbout =8'h4b;

8'hb4:sbout =8'hc6;

8'hb5:sbout =8'hd2;

8'hb6:sbout =8'h79;

8'hb7:sbout =8'h20;

8'hb8:sbout =8'h9a;

8'hb9:sbout =8'hdb;

8'hba:sbout =8'hc0;

8'hbb:sbout =8'hfe;

8'hbc:sbout =8'h78;

8'hbd:sbout =8'hcd;

8'hbe:sbout =8'h5a;

8'hbf:sbout =8'hf4;

8'hc0:sbout =8'h1f;

8'hc1:sbout =8'hdd;

8'hc2:sbout =8'ha8;

8'hc3:sbout =8'h33;

8'hc4:sbout =8'h88;

8'hc5:sbout =8'h07;

8'hc6:sbout =8'hc7;

8'hc7:sbout =8'h31;

8'hc8:sbout =8'hb1;

8'hc9:sbout =8'h12;

8'hca:sbout =8'h10;

8'hcb:sbout =8'h59;

8'hcc:sbout =8'h27;

8'hcd:sbout =8'h80;

8'hce:sbout =8'hec;

8'hcf:sbout =8'h5f;

8'hd0:sbout =8'h60;

8'hd1:sbout =8'h51;

8'hd2:sbout =8'h7f;

8'hd3:sbout =8'ha9;

8'hd4:sbout =8'h19;

8'hd5:sbout =8'hb5;

8'hd6:sbout =8'h4a;

8'hd7:sbout =8'h0d;

8'hd8:sbout =8'h2d;

8'hd9:sbout =8'he5;

8'hda:sbout =8'h7a;

8'hdb:sbout =8'h9f;

8'hdc:sbout =8'h93;

8'hdd:sbout =8'hc9;

8'hde:sbout =8'h9c;

8'hdf:sbout =8'hef;

8'he0:sbout =8'ha0;

8'he1:sbout =8'he0;

8'he2:sbout =8'h3b;

8'he3:sbout =8'h4d;

8'he4:sbout =8'hae;

8'he5:sbout =8'h2a;

8'he6:sbout =8'hf5;

8'he7:sbout =8'hb0;

8'he8:sbout =8'hc8;

8'he9:sbout =8'heb;

8'hea:sbout =8'hbb;

8'heb:sbout =8'h3c;

8'hec:sbout =8'h83;

8'hed:sbout =8'h53;

8'hee:sbout =8'h99;

8'hef:sbout =8'h61;

8'hf0:sbout =8'h17;

8'hf1:sbout =8'h2b;

8'hf2:sbout =8'h04;

8'hf3:sbout =8'h7e;

8'hf4:sbout =8'hba;

8'hf5:sbout =8'h77;

8'hf6:sbout =8'hd6;

8'hf7:sbout =8'h26;

8'hf8:sbout =8'he1;

8'hf9:sbout =8'h69;

8'hfa:sbout =8'h14;

8'hfb:sbout =8'h63;

8'hfc:sbout =8'h55;

8'hfd:sbout =8'h21;

8'hfe:sbout =8'h0c;

8'hff:sbout =8'h7d;

endcase

end

endmodule

//for gfunction

module sbox(in,c);

input [7:0] in;

output reg [7:0] c;

always@(in) begin

case (in)

8'h00: c=8'h63;

8'h01: c=8'h7c;

8'h02: c=8'h77;

8'h03: c=8'h7b;

8'h04: c=8'hf2;

8'h05: c=8'h6b;

8'h06: c=8'h6f;

8'h07: c=8'hc5;

8'h08: c=8'h30;

8'h09: c=8'h01;

8'h0a: c=8'h67;

8'h0b: c=8'h2b;

8'h0c: c=8'hfe;

8'h0d: c=8'hd7;

8'h0e: c=8'hab;

8'h0f: c=8'h76;

8'h10: c=8'hca;

8'h11: c=8'h82;

8'h12: c=8'hc9;

8'h13: c=8'h7d;

8'h14: c=8'hfa;

8'h15: c=8'h59;

8'h16: c=8'h47;

8'h17: c=8'hf0;

8'h18: c=8'had;

8'h19: c=8'hd4;

8'h1a: c=8'ha2;

8'h1b: c=8'haf;

8'h1c: c=8'h9c;

8'h1d: c=8'ha4;

8'h1e: c=8'h72;

8'h1f: c=8'hc0;

8'h20: c=8'hb7;

8'h21: c=8'hfd;

8'h22: c=8'h93;

8'h23: c=8'h26;

8'h24: c=8'h36;

8'h25: c=8'h3f;

8'h26: c=8'hf7;

8'h27: c=8'hcc;

8'h28: c=8'h34;

8'h29: c=8'ha5;

8'h2a: c=8'he5;

8'h2b: c=8'hf1;

8'h2c: c=8'h71;

8'h2d: c=8'hd8;

8'h2e: c=8'h31;

8'h2f: c=8'h15;

8'h30: c=8'h04;

8'h31: c=8'hc7;

8'h32: c=8'h23;

8'h33: c=8'hc3;

8'h34: c=8'h18;

8'h35: c=8'h96;

8'h36: c=8'h05;

8'h37: c=8'h9a;

8'h38: c=8'h07;

8'h39: c=8'h12;

8'h3a: c=8'h80;

8'h3b: c=8'he2;

8'h3c: c=8'heb;

8'h3d: c=8'h27;

8'h3e: c=8'hb2;

8'h3f: c=8'h75;

8'h40: c=8'h09;

8'h41: c=8'h83;

8'h42: c=8'h2c;

8'h43: c=8'h1a;

8'h44: c=8'h1b;

8'h45: c=8'h6e;

8'h46: c=8'h5a;

8'h47: c=8'ha0;

8'h48: c=8'h52;

8'h49: c=8'h3b;

8'h4a: c=8'hd6;

8'h4b: c=8'hb3;

8'h4c: c=8'h29;

8'h4d: c=8'he3;

8'h4e: c=8'h2f;

8'h4f: c=8'h84;

8'h50: c=8'h53;

8'h51: c=8'hd1;

8'h52: c=8'h00;

8'h53: c=8'hed;

8'h54: c=8'h20;

8'h55: c=8'hfc;

8'h56: c=8'hb1;

8'h57: c=8'h5b;

8'h58: c=8'h6a;

8'h59: c=8'hcb;

8'h5a: c=8'hbe;

8'h5b: c=8'h39;

8'h5c: c=8'h4a;

8'h5d: c=8'h4c;

8'h5e: c=8'h58;

8'h5f: c=8'hcf;

8'h60: c=8'hd0;

8'h61: c=8'hef;

8'h62: c=8'haa;

8'h63: c=8'hfb;

8'h64: c=8'h43;

8'h65: c=8'h4d;

8'h66: c=8'h33;

8'h67: c=8'h85;

8'h68: c=8'h45;

8'h69: c=8'hf9;

8'h6a: c=8'h02;

8'h6b: c=8'h7f;

8'h6c: c=8'h50;

8'h6d: c=8'h3c;

8'h6e: c=8'h9f;

8'h6f: c=8'ha8;

8'h70: c=8'h51;

8'h71: c=8'ha3;

8'h72: c=8'h40;

8'h73: c=8'h8f;

8'h74: c=8'h92;

8'h75: c=8'h9d;

8'h76: c=8'h38;

8'h77: c=8'hf5;

8'h78: c=8'hbc;

8'h79: c=8'hb6;

8'h7a: c=8'hda;

8'h7b: c=8'h21;

8'h7c: c=8'h10;

8'h7d: c=8'hff;

8'h7e: c=8'hf3;

8'h7f: c=8'hd2;

8'h80: c=8'hcd;

8'h81: c=8'h0c;

8'h82: c=8'h13;

8'h83: c=8'hec;

8'h84: c=8'h5f;

8'h85: c=8'h97;

8'h86: c=8'h44;

8'h87: c=8'h17;

8'h88: c=8'hc4;

8'h89: c=8'ha7;

8'h8a: c=8'h7e;

8'h8b: c=8'h3d;

8'h8c: c=8'h64;

8'h8d: c=8'h5d;

8'h8e: c=8'h19;

8'h8f: c=8'h73;

8'h90: c=8'h60;

8'h91: c=8'h81;

8'h92: c=8'h4f;

8'h93: c=8'hdc;

8'h94: c=8'h22;

8'h95: c=8'h2a;

8'h96: c=8'h90;

8'h97: c=8'h88;

8'h98: c=8'h46;

8'h99: c=8'hee;

8'h9a: c=8'hb8;

8'h9b: c=8'h14;

8'h9c: c=8'hde;

8'h9d: c=8'h5e;

8'h9e: c=8'h0b;

8'h9f: c=8'hdb;

8'ha0: c=8'he0;

8'ha1: c=8'h32;

8'ha2: c=8'h3a;

8'ha3: c=8'h0a;

8'ha4: c=8'h49;

8'ha5: c=8'h06;

8'ha6: c=8'h24;

8'ha7: c=8'h5c;

8'ha8: c=8'hc2;

8'ha9: c=8'hd3;

8'haa: c=8'hac;

8'hab: c=8'h62;

8'hac: c=8'h91;

8'had: c=8'h95;

8'hae: c=8'he4;

8'haf: c=8'h79;

8'hb0: c=8'he7;

8'hb1: c=8'hc8;

8'hb2: c=8'h37;

8'hb3: c=8'h6d;

8'hb4: c=8'h8d;

8'hb5: c=8'hd5;

8'hb6: c=8'h4e;

8'hb7: c=8'ha9;

8'hb8: c=8'h6c;

8'hb9: c=8'h56;

8'hba: c=8'hf4;

8'hbb: c=8'hea;

8'hbc: c=8'h65;

8'hbd: c=8'h7a;

8'hbe: c=8'hae;

8'hbf: c=8'h08;

8'hc0: c=8'hba;

8'hc1: c=8'h78;

8'hc2: c=8'h25;

8'hc3: c=8'h2e;

8'hc4: c=8'h1c;

8'hc5: c=8'ha6;

8'hc6: c=8'hb4;

8'hc7: c=8'hc6;

8'hc8: c=8'he8;

8'hc9: c=8'hdd;

8'hca: c=8'h74;

8'hcb: c=8'h1f;

8'hcc: c=8'h4b;

8'hcd: c=8'hbd;

8'hce: c=8'h8b;

8'hcf: c=8'h8a;

8'hd0: c=8'h70;

8'hd1: c=8'h3e;

8'hd2: c=8'hb5;

8'hd3: c=8'h66;

8'hd4: c=8'h48;

8'hd5: c=8'h03;

8'hd6: c=8'hf6;

8'hd7: c=8'h0e;

8'hd8: c=8'h61;

8'hd9: c=8'h35;

8'hda: c=8'h57;

8'hdb: c=8'hb9;

8'hdc: c=8'h86;

8'hdd: c=8'hc1;

8'hde: c=8'h1d;

8'hdf: c=8'h9e;

8'he0: c=8'he1;

8'he1: c=8'hf8;

8'he2: c=8'h98;

8'he3: c=8'h11;

8'he4: c=8'h69;

8'he5: c=8'hd9;

8'he6: c=8'h8e;

8'he7: c=8'h94;

8'he8: c=8'h9b;

8'he9: c=8'h1e;

8'hea: c=8'h87;

8'heb: c=8'he9;

8'hec: c=8'hce;

8'hed: c=8'h55;

8'hee: c=8'h28;

8'hef: c=8'hdf;

8'hf0: c=8'h8c;

8'hf1: c=8'ha1;

8'hf2: c=8'h89;

8'hf3: c=8'h0d;

8'hf4: c=8'hbf;

8'hf5: c=8'he6;

8'hf6: c=8'h42;

8'hf7: c=8'h68;

8'hf8: c=8'h41;

8'hf9: c=8'h99;

8'hfa: c=8'h2d;

8'hfb: c=8'h0f;

8'hfc: c=8'hb0;

8'hfd: c=8'h54;

8'hfe: c=8'hbb;

8'hff: c=8'h16;

endcase

end

endmodule