```
module ALU (SW, KEY, LEDR, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5);
   input [7:0] SW;
   input [2:0] KEY;
   output [7:0] LEDR;
   output [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
   // For ease of readability
   wire [3:0] A, B;
   wire [2:0] K;
   wire [4:0] addOut;

   assign A = SW[7:4];
   assign B = SW[3:0];
   assign K = ~KEY;
   // Calculates sum for the ALU
   FARip4b a1(SW[7:0],
              addOut[4:0]);
   // Stores our ouput value
   reg [7:0] ALUout;

   always @(*)
   begin
      case (K)
         // Case 0
         3'b000: ALUout = addOut;
         // Case 1
         3'b001: ALUout = A + B;
         // Case 2
         3'b010: begin
            ALUout[7:4] = ~(A & B);
            ALUout[3:0] = ~(A ^ B);
            end
         // Case 3
         3'b011: begin
            if (|{A, B})
               ALUout = 8'b00001111;
            else
               ALUout = 8'b0;
            end
         // Case 4
         3'b100: begin
            if ((A && !(A & (A - 1))) && (|B && ~(^B) && ~(&B)))
               ALUout = 8'b1110000;
            else
               ALUout = 8'b0;
            end
         // Case 5
         3'b101: ALUout = {A, ~B};
         // Default case
         default: ALUout = 8'b00000000;
      endcase
   end
   // Assign results to appropraite displays/LEDs
   assign LEDR[7:0] = ALUout;
   HexDecoder h0 (SW[3:0], HEX0);
   HexDecoder h1 (SW[7:4], HEX2);
   HexDecoder h2 (4'b0000, HEX1);
   HexDecoder h3 (4'b0000, HEX3);
   HexDecoder h4 (ALUout[3:0], HEX4);
   HexDecoder h5 (ALUout[7:4], HEX5);
endmodule

module FARip4b (In, Out);

   input [7:0] In;
   output [4:0] Out;
   wire w1, w2, w3;

   FA u0 (.in1(In[4]), .in2(In[0]), .cin(1'b0),
          .s(Out[0]), .cout(w1) );
   FA u1 (.in1(In[5]), .in2(In[1]), .cin(w1),
          .s(Out[1]), .cout(w2) );
   FA u2 (.in1(In[6]), .in2(In[2]), .cin(w2),
          .s(Out[2]), .cout(w3) );
   FA u3 (.in1(In[7]), .in2(In[3]), .cin(w3),
          .s(Out[3]), .cout(Out[4]) );
endmodule
```

```

module FA (input in1, in2, cin,
           output s, cout);
   assign s = in1 ^ in2 ^ cin;
   assign cout = (in1 & in2) | (in1 & cin) | (in2 & cin);
endmodule

module HexDecoder (In, HEX);
   input [3:0] In;
   output [6:0] HEX;
   // For ease of readability
   assign c0 = In[0];
   assign c1 = In[1];
   assign c2 = In[2];
   assign c3 = In[3];

   assign HEX[0] = (~c3 & ~c2 & ~c1 & c0) + (~c3 & c2 & ~c1 & ~c0) + (c3 & ~c2 & c1 & c0) + (c3 & c2 & ~c1 & c0);
   assign HEX[1] = (~c3 & c2 & ~c1 & c0) + (~c3 & c2 & c1 & ~c0) + (c3 & ~c2 & c1 & c0) + (c3 & c2 & ~c1 & ~c0) + (c3 & c2 & c1 & ~c0) + (c3 & c2 & c1 & c0);
   assign HEX[2] = (~c3 & ~c2 & c1 & ~c0) + (c3 & c2 & ~c1 & ~c0) + (c3 & c2 & c1 & ~c0) + (c3 & c2 & c1 & c0);
   assign HEX[3] = (~c3 & ~c2 & ~c1 & c0) + (~c3 & c2 & ~c1 & ~c0) + (~c3 & c2 & c1 & c0) + (c3 & ~c2 & ~c1 & c0) + (c3 & ~c2 & c1 & ~c0) + (c3 & c2 & c1 & c0);
   assign HEX[4] = (~c3 & ~c2 & ~c1 & c0) + (~c3 & ~c2 & c1 & c0) + (~c3 & c2 & ~c1 & ~c0) + (~c3 & c2 & ~c1 & c0) + (~c3 & c2 & c1 & c0) + (c3 & ~c2 & ~c1 & c0);
   assign HEX[5] = (~c3 & ~c2 & ~c1 & c0) + (~c3 & ~c2 & c1 & ~c0) + (~c3 & ~c2 & c1 & c0) + (~c3 & c2 & c1 & c0) + (c3 & c2 & ~c1 & c0);
   assign HEX[6] = (~c3 & ~c2 & ~c1 & ~c0) + (~c3 & ~c2 & ~c1 & c0) + (~c3 & c2 & c1 & c0) + (c3 & c2 & ~c1 & ~c0);
endmodule


```