**Họ và tên: Hà Minh Khuê MSSV: 1513079**

**Bài tập tuần 9**

**Bài 1:**

*module* cau1(SW,LEDR,HEX0,HEX1);

    input [2:0]SW;

    output [8:0]LEDR;

    output[6:0]HEX0,HEX1;

    wire [7:0]alu\_out;

    wire [7:0]A=8'b11111111;

    wire [7:0]B=8'b11111111;

    alu\_logic(A,B,SW[2:0],alu\_out,LEDR[8]);

    assign LEDR[7:0]=alu\_out;

    decoder (alu\_out[3:0],HEX0);

    decoder (alu\_out[7:4],HEX1);

*endmodule*

*module* alu\_logic(A,B,opcode,alu\_out,ovf);

    input [7:0]A,B;

    input [2:0]opcode;

    output reg[7:0]alu\_out;

    output reg ovf;

    always @(\*)

    begin

    case (opcode)

    3'b000:

        begin

        alu\_out=A+B;

        ovf=(A[7]&B[7]&~alu\_out[7])|~(A[7]&~B[7]&alu\_out[7])|(A[7]&B[7]&alu\_out[7]);

        end

    3'b001:

        begin

        alu\_out=A-B;

        ovf=0;

        end

    3'b010:

        begin

        alu\_out =A-1;

        ovf=0;

        end

    3'b011:

        begin

        alu\_out=A+1;

        ovf=(A[7]&1&~alu\_out[7])|~(A[7]&~1&alu\_out[7])|(A[7]&1&alu\_out[7]);

        end

    3'b100:

        begin

        alu\_out=A&B;

        ovf = 0;

        end

    3'b101:

        begin

        alu\_out=A|B;

        ovf=0;

        end

    default:

        begin

        alu\_out =~A;

        ovf=0;

        end

    endcase

    end

*endmodule*

*module* decoder(c,hex);

    input [7:0]c;

    output reg [6:0]hex;

    always@(c)

    case (c)

    4'b 0000: hex = 7'b 1000000;

    4'b 0001: hex = 7'b 1111001;

    4'b 0010: hex = 7'b 0100100;

    4'b 0011: hex = 7'b 0110000;

    4'b 0100: hex = 7'b 0011001;

    4'b 0101: hex = 7'b 0010010;

    4'b 0110: hex = 7'b 0000010;

    4'b 0111: hex = 7'b 1111000;

    4'b 1000: hex = 7'b 0000000;

    4'b 1001: hex = 7'b 0010000;

    4'b1010: hex = 7'b0001000; //A

    4'b1011: hex= 7'b0000011; //B

    4'b1100: hex = 7'b1000110; //C

    4'b1101: hex= 7'b0100001; //D

    4'b1110: hex = 7'b0000110; //E

    4'b1111: hex = 7'b0001110; //F

    endcase

*endmodule*

**Bài 2:**

*module* cau2(SW,LEDR,HEX0,HEX1);

    input [2:0]SW;

    output [9:0]LEDR;

    output[6:0]HEX0,HEX1;

    wire [7:0]alu\_out;

    wire [7:0]A=8'b11111111;

    wire [7:0]B=8'b11111111;

    alu\_logic(A,B,SW[2:0],alu\_out,LEDR[8],LEDR[9]);

    assign LEDR[7:0]=alu\_out;

    decoder (alu\_out[3:0],HEX0);

    decoder (alu\_out[7:4],HEX1);

*endmodule*

*module* alu\_logic(A,B,opcode,alu\_out,ovf,br);

    input [7:0]A,B;

    input [2:0]opcode;

    output reg[7:0]alu\_out;

    output reg ovf,br;

    always @(\*)

    begin

    case (opcode)

    3'b000:

        begin

        alu\_out=A+B;

        ovf=(A[7]&B[7]&~alu\_out[7])|~(A[7]&~B[7]&alu\_out[7])|(A[7]&B[7]&alu\_out[7]);

        end

    3'b001:

        begin

        alu\_out=A-B;

        ovf=0;

        end

    3'b010:

        begin

        alu\_out =A-1;

        ovf=0;

        end

    3'b011:

        begin

        alu\_out=A+1;

        ovf=(A[7]&1&~alu\_out[7])|~(A[7]&~1&alu\_out[7])|(A[7]&1&alu\_out[7]);

        end

    3'b100:

        begin

        alu\_out=A&B;

        ovf = 0;

        end

    3'b101:

        begin

        alu\_out=A|B;

        ovf=0;

        end

    3'b110:

        br=(A==B)?1:0;

    3'b111:

        br=(A>B)?1:0;

    default:

        begin

        alu\_out =~A;

        ovf=0;

        end

    endcase

    end

*endmodule*

*module* decoder(c,hex);

    input [7:0]c;

    output reg [6:0]hex;

    always@(c)

    case (c)

        4'b 0000: hex = 7'b 1000000;

        4'b 0001: hex = 7'b 1111001;

        4'b 0010: hex = 7'b 0100100;

        4'b 0011: hex = 7'b 0110000;

        4'b 0100: hex = 7'b 0011001;

        4'b 0101: hex = 7'b 0010010;

        4'b 0110: hex = 7'b 0000010;

        4'b 0111: hex = 7'b 1111000;

        4'b 1000: hex = 7'b 0000000;

        4'b 1001: hex = 7'b 0010000;

        4'b1010: hex = 7'b0001000; //A

        4'b1011: hex= 7'b0000011; //B

        4'b1100: hex = 7'b1000110; //C

        4'b1101: hex= 7'b0100001; //D

        4'b1110: hex = 7'b0000110; //E

        4'b1111: hex = 7'b0001110; //F

endcase

*endmodule*

**Bài 3:**

*module* cau3(SW,LEDR,HEX0,HEX1);

    input [3:0]SW;

    output [8:0]LEDR;

    output[6:0]HEX0,HEX1;

    wire [7:0]alu\_out;

    wire [7:0]A=8'b11111111;

    wire [7:0]B=8'b11111110;

    alu\_logic(A,B,SW[3:0],alu\_out,LEDR[8]);

    assign LEDR[7:0]=alu\_out;

    decoder (alu\_out[3:0],HEX0);

    decoder (alu\_out[7:4],HEX1);

*endmodule*

*module* alu\_logic(A,B,opcode,alu\_out,ovf);

    input [7:0]A,B;

    input [3:0]opcode;

    output reg[7:0]alu\_out;

    output reg ovf;

    always @(\*)

    begin

    case (opcode)

    4'b000:

        begin

        alu\_out=A+B;

        ovf=(A[7]&B[7]&~alu\_out[7])|~(A[7]&~B[7]&alu\_out[7])|(A[7]&B[7]&alu\_out[7]);

        end

    4'b0001:

        begin

        alu\_out=A-B;

        ovf=0;

        end

    4'b0010:

        begin

        alu\_out =A+1;

        ovf=0;

        end

    4'b0011:

        begin

        alu\_out=~B;

        ovf=0;

        end

    4'b0100:

        begin

        alu\_out=A<<1;

        ovf = 0;

        end

    4'b0101:

        begin

        alu\_out=A>>1;

        ovf=0;

        end

    4'b0110:

        begin

        alu\_out={A[6:0],A[7]};

        ovf=0;

        end

    4'b0111:

        begin

        alu\_out={A[0],A[7:1]};

        ovf=0;

        end

    4'b1000:

        begin

        alu\_out=A&B;

        ovf=0;

        end

    4'b1001:

        begin

        alu\_out=A|B;

        ovf=0;

        end

    4'b1010:

        begin

        alu\_out=A^B;

        ovf=0;

        end

    4'b1011:

        begin

        alu\_out=~(A&B);

        ovf=0;

        end

    4'b1100:

        begin

        alu\_out=~(A^B);

        ovf=0;

        end

    4'b1101:

        begin

        alu\_out=A&B;

        ovf=0;

        end

    4'b1110:

        begin

        alu\_out=(A>B)?8'b11111111:8'b00000000;

        ovf=0;

        end

    4'b1111:

        begin

        alu\_out=(A==B)?8'b11111111:8'b00000000;

        ovf=0;

        end

    endcase

    end

*endmodule*

*module* decoder(c,hex);

    input [7:0]c;

    output reg [6:0]hex;

    always@(c)

    case (c)

        4'b 0000: hex = 7'b 1000000;

        4'b 0001: hex = 7'b 1111001;

        4'b 0010: hex = 7'b 0100100;

        4'b 0011: hex = 7'b 0110000;

        4'b 0100: hex = 7'b 0011001;

        4'b 0101: hex = 7'b 0010010;

        4'b 0110: hex = 7'b 0000010;

        4'b 0111: hex = 7'b 1111000;

        4'b 1000: hex = 7'b 0000000;

        4'b 1001: hex = 7'b 0010000;

        4'b1010: hex = 7'b0001000; //A

        4'b1011: hex= 7'b0000011; //B

        4'b1100: hex = 7'b1000110; //C

        4'b1101: hex= 7'b0100001; //D

        4'b1110: hex = 7'b0000110; //E

        4'b1111: hex = 7'b0001110; //F

    endcase

*endmodule*