```
// Reset with KEY[0]. SW[9] is Run.
// The processor executes the instructions in the file inst_mem.mif
module part3 (KEY, SW, CLOCK_50, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0, LEDR);
    input [0:0] KEY;
    input [9:0] SW;
    input CLOCK_50;
    output [6:0] HEX5, HEX4, HEX3, HEX2, HEX1, HEX0;
    output [9:0] LEDR;

    wire [15:0] DOUT, ADDR;
    wire Done;
    reg [15:0] DIN;
    wire W, Sync, Run;
    wire inst_mem_cs, SW_cs, LED_reg_cs, seven_seg_cs;
    wire [15:0] inst_mem_q;
    wire [8:0] LED_reg, SW_reg; // LED[9] and SW[9] are used for Run

    // synchronize the Run input
    flipflop U1 (SW[9], KEY[0], CLOCK_50, Sync);
    flipflop U2 (Sync, KEY[0], CLOCK_50, Run);

    // module proc(DIN, Resetn, Clock, Run, DOUT, ADDR, W, Done);
    proc U3 (DIN, KEY[0], CLOCK_50, Run, DOUT, ADDR, W, Done);

    assign inst_mem_cs = (ADDR[15:12] == 4'h0);
    assign LED_reg_cs = (ADDR[15:12] == 4'h1);
    assign seven_seg_cs = (ADDR[15:12] == 4'h2);
    assign SW_cs = (ADDR[15:12] == 4'h3);
    // module inst_mem (data, wren, address, clock, q);
    inst_mem U4 (ADDR[7:0], CLOCK_50, DOUT, inst_mem_cs & W, inst_mem_q);

    always @ (*)
    if (inst_mem_cs == 1'b1)
        DIN = inst_mem_q;
    else if (SW_cs == 1'b1)
        DIN = {7'b0000000, SW_reg};
    else
        DIN = 16'bxxxxxxxxxxxxxxxx;

    // module regn(R, Rin, Clock, Q);
    regn #(.n(9)) U5 (DOUT[8:0], LED_reg_cs & W, CLOCK_50, LED_reg);
    assign LEDR[8:0] = LED_reg;
    assign LEDR[9] = Run;

    // module regn(R, Rin, Clock, Q);
    regn #(.n(9)) U7 (SW[8:0], 1'b1, CLOCK_50, SW_reg); // SW[9] is used for Run

    // 7 Segment display
    // module seg7_scroll (Data, Addr, Sel, Resetn, Clock, H5, H4, H3, H2, H1, H0);
    seg7_scroll HEX(DOUT[6:0], ADDR[2:0], seven_seg_cs & W, KEY[0], CLOCK_50, HEX5,
    HEX4, HEX3, HEX2, HEX1, HEX0);
endmodule
```