```
module IRAM(
    input clk,
    input [3:0]write,
    input [63:0] address,
    input [63:0] instr_in,
    output reg [63:0] instr_out
    );
    reg [15:0] MEM [65535:0];
    parameter FETCH = 8'd0;
    parameter NOP = 8'd3;
    parameter END = 8'd4;
    parameter CLR = 8'd5;
    parameter LOAD = 8'd8;
    parameter LOADM = 8'd14;
    parameter STAC = 8'd18;
    parameter INC = 8'd21;
    parameter INCAC = 8'd27;
    parameter JUMP = 8'd28;
    parameter MOVE = 8'd34;
    parameter ADD = 8'd39;
    parameter SUB = 8'd40;
    parameter MUL = 8'd41;
    parameter AND = 8'd42;
    parameter OR = 8'd43;
    parameter STORM = 8'd44;
    initial begin

    MEM[0] = LOAD; //load 0 BASE_A
    MEM[1] = 16'd0;
    MEM[2] = 16'd10;
    MEM[3] = LOAD; //load 1 BASE_B
    MEM[4] = 16'd1;
    MEM[5] = 16'd13;
    MEM[6] = LOAD; //load 2 BASE_C
    MEM[7] = 16'd2;
    MEM[8] = 16'd7;
    MEM[9] = LOAD; //load 3 i_ref
    MEM[10] = 16'd3;
    MEM[11] = 16'd9;
    MEM[12] = LOAD; //load 4 j_ref
    MEM[13] = 16'd4;
    MEM[14] = 16'd12;
    MEM[15] = LOAD;
    MEM[16] = 16'd31;
    MEM[17] = 16'd15;
    MEM[18] = CLR;
    MEM[19] = 16'd8;
    MEM[20] = MOVE;
    MEM[21] = 16'd23;
    MEM[22] = 16'd14;
    MEM[23] = JUMP;
    MEM[24] = 16'd2;
    MEM[25] = 16'd86;
    MEM[26] = JUMP;
```

```
    MEM[27] = 16'd6;
    MEM[28] = 16'd70;
    MEM[29] = CLR;
    MEM[30] = 16'd16;
    MEM[31] = CLR;
    MEM[32] = 16'd11;
    MEM[33] = JUMP;
    MEM[34] = 16'd4;
    MEM[35] = 16'd60;
    MEM[36] = LOADM;
    MEM[37] = 16'd21;
    MEM[38] = MOVE;
    MEM[39] = 16'd20;
    MEM[40] = 16'd17;
    MEM[41] = LOADM;
    MEM[42] = 16'd22;
    MEM[43] = MUL;
    MEM[44] = MOVE;
    MEM[45] = 16'd20;
    MEM[46] = 16'd17;
    MEM[47] = MOVE;
    MEM[48] = 16'd16;
    MEM[49] = 16'd20;
    MEM[50] = ADD;
    MEM[51] = MOVE;
    MEM[52] = 16'd20;
    MEM[53] = 16'd16;
    MEM[54] = INC;
    MEM[55] = 16'd1;
    MEM[56] = 16'd11;
    MEM[57] = JUMP;
    MEM[58] = 16'd5;
    MEM[59] = 16'd33;
    MEM[60] = MOVE;
    MEM[61] = 16'd16;
    MEM[62] = 16'd20;
    MEM[63] = STORM;
    MEM[64] = INC;
    MEM[65] = 16'd1;
    MEM[66] = 16'd14;
    MEM[67] = JUMP;
    MEM[68] = 16'd7;
    MEM[69] = 16'd26;
    MEM[70] = INC;
    MEM[71] = 16'd1;
    MEM[72] = 16'd8;
    MEM[73] = MOVE;
    MEM[74] = 16'd15;
    MEM[75] = 16'd17;
    MEM[76] = MOVE;
    MEM[77] = 16'd14;
    MEM[78] = 16'd20;
    MEM[79] = SUB;
    MEM[80] = MOVE;
```

```
    MEM[81] = 16'd20;
    MEM[82] = 16'd14;
    MEM[83] = JUMP;
    MEM[84] = 16'd3;
    MEM[85] = 16'd23;
    MEM[86] = END;

end
    always @(posedge clk)
        begin
            if (write[0:0] == 1)
            begin
                MEM[address[15:0]] <= instr_in[15:0];
            end
            if (write[1:1] == 1)
            begin
                MEM[address[31:16]] <= instr_in[31:16];
            end
            if (write[2:2] == 1)
            begin
                MEM[address[47:32]] <= instr_in[47:32];
            end
            if (write[3:3] == 1)
            begin
                MEM[address[63:48]] <= instr_in[63:48];
            end
            instr_out[15:0] <= MEM[address[15:0]];
            instr_out[31:16] <= MEM[address[31:16]];
            instr_out[47:32] <= MEM[address[47:32]];
            instr_out[63:48] <= MEM[address[63:48]];
        end
endmodule
```