```
// EXERCICIOS 1


`include "Code_control.v"
`include "code_ULA.v"
`include "Code_MemoryData.v"
`include "Code_Outhers.v"
`include "Code_PC.v"
`include "Code_REG.v"

module nRisc(clock,code,address);

    input clock;
    input [7:0]code;                    // linha de codigo a ser executada
    output [7:0]address;                    // ultimo endereço na memoria de programa

    wire zero,negativo,saida_bloco_controle;
    wire [12:0] controle;
    wire [7:0] saida1_reg,  saida2_reg;             // saidas de banco de registradores
    wire [7:0] saida_mux_a;                 // mux A
    wire [7:0] saida_1_demux_b, saida_2_demux_b;        // demux b
    wire [7:0] saida_ula;                   //ULA
    wire [7:0] saida_memo;                  // saida memoria
    wire [7:0] saida_demux_ula;                 //demux ula
    wire [7:0] saida_1_demux_halt,  saida_2_demux_halt;  //demux HALT
    wire [7:0] saida_mux_gt;                //Mux gt
    wire [7:0] saida_mux_jump;              // mux jump
    wire [7:0] saida_somador_jump;              //somador jump
    wire [7:0] saida_somador_gt;                // somador gt
    wire [7:0] saida_pc;                    // PC

    assign address = saida_pc;


    always @(clock) begin end

    // PC
    PC              modulo1(  saida_2_demux_halt, clock, saida_pc);


    // Central de Controle
    control         modulo2(  code, controle);


    //Banco de Regsitradores
    banco_reg       modulo3(  code,  saida_mux_a,  controle[7],  clock,  saida1_reg,
        saida2_reg);


    // ULA
    ULA             modulo4(  controle[8],  saida1_reg,  saida_demux_ula,  saida_ula,  zero,
        negativo);


    // Memória de Dados
    memory          modulo5(  clock,  saida_1_demux_b,  controle[6],  controle[5],
        saida2_reg,  saida_memo);


    //DEMAIS COMPONENTES
    ULA_GT          modulo6( controle[12] , saida_pc,  code[3:0],  saida_somador_gt);  //
        Somador gt


    somador_jump    modulo7(  saida_mux_jump,  saida_pc,  saida_somador_jump);   // Somador GT


    mux_gt          modulo8(  controle[3],  saida_somador_jump,  saida_somador_gt,
        saida_mux_gt);  // MUX gt
```

```
65
66
67      mux_jump       modulo9(  saida_bloco_controle,  saida_mux_jump);  // MUX jump
68
69
70      demux_halt     modulo10(  controle[0],  saida_mux_gt,  saida_1_demux_halt,
            saida_2_demux_halt);  //Demux Halt
71
72
73      mux_a          modulo11(  controle[11:10],  code[3:0],  saida_2_demux_b,  saida_memo,
            saida_mux_a);  // Mux A
74
75
76      demux_b        modulo12(  controle[9],  saida_ula,  saida_1_demux_b,  saida_2_demux_b);
            //Demux B
77
78
79      mux_ula        modulo13(  controle[2:1],  saida2_reg, code[3:0],  saida_demux_ula);  //
            Demux Ula
80
81
82      bloco_controle modulo14(  zero,  negativo,  controle[4],  saida_bloco_controle);   //
            Bloco de Seleção de Jump
83
84
85  endmodule
```