```
//MEMORIA DE DADOS

module memory(clock,endereco,controle_escrita,controle_leitura,dado_entrada,dado_saida);

    input clock,controle_escrita,controle_leitura;
    input [7:0]endereco,dado_entrada;
    output [7:0]dado_saida;

    reg [7:0]line;
    reg[7:0]memdata[255:0];

    assign dado_saida=line;

    always @(negedge clock) begin  //leitura
        if(controle_leitura)
            line = memdata[endereco];
    end

    always @(posedge clock) begin  //escrita
        if(controle_escrita)
            memdata[endereco]=dado_entrada;
    end
endmodule




//PC

module PC(address,clock,saida_pc);

    input clock;
    input [7:0]address;             //tamanho do salto
    output [7:0]saida_pc;

    reg [7:0] valor;            //valor atual de PC

    assign saida_pc = valor;

    initial begin
        valor = 0;
    end

    always @(posedge clock) begin
        valor = address;
    end

endmodule




//MEMORIA DE PROGRAMA

module memory_program(PC,code,clock);
    input clock;
    input [7:0]PC;
    output [7:0]code;

    reg [7:0]linha;
    reg[7:0]memprogram[65:0];
    assign code=linha;

    initial begin
        $readmemb("C:/Users/Thepe/Documents/VERILOG/project.dat",memprogram);
    end
```

```
70
71      always @(negedge clock) begin //caso PC não mude(em modo de halt desligado), o codigo
            não executa
72          linha = memprogram[PC];
73      end
74  endmodule
75
76
77
78
79
80  //CENTRAL DE CONTROLE
81
82  module control(line,saida_controle);
83
84      input [7:0]line;
85      output [12:0] saida_controle;
86
87      reg [11:0] controle;
88      // controle[12]=controle ULA GT - controle[11:10]=ControleA - controle[9]=controleB -
            controle[8]=controleUla - controle[7]=controleEscritaReg -
            controle[6]=controleEscritaMemo - controle[5]=controleLeituraMemo -
89      // controle[4]=controleJump - controle[3]=controleGoTo - controle[2:1]=controleInputUla
            - controleHalt[0]=controleHalt
90
91      assign saida_controle = controle;
92
93      always @(line) begin
94          case ( line[7:4] )
95              4'b0000: controle = 13'b0011010000011; //movfw
96              4'b0001: controle = 13'b0100100010001; //jogw
97              4'b0010: controle = 13'b0101010000011; //addw
98              4'b0011: controle = 13'b0000010000001; //clearw
99              4'b0100: controle = 13'b0100101000101; //mtm
100             4'b0101: controle = 13'b0100010100101; //mfm
101             4'b0110: controle = 13'b0100100010001; //jog
102             4'b0111: controle = 13'b0100100010001; //joe
103             4'b1000: controle = 13'b0100100010001; //jol
104             4'b1001: controle = 13'b0011010000101; //move
105             4'b1010: controle = 13'b0101100001101; //gt
106             4'b1011: controle = 13'b0100000000000; //end
107             4'b1010: controle = 13'b1101100001101; //gtb
108             default: controle = 13'b0100000000000; //end
109         endcase
110     end
111 endmodule
112
113
114
115
116
117 //       OUTROS COMPONENTES
118
119 // MUX GT
120 module mux_gt(controle_gt,soma_1_mux_gt,soma_2_mux_gt,saida_mux_gt);
121
122     input controle_gt;
123     input [7:0]soma_1_mux_gt,soma_2_mux_gt;
124     output [7:0]saida_mux_gt;  //saida para o PC
125     reg [7:0]temp;
126
127     assign saida_mux_gt = temp;
128
129     always @(controle_gt,soma_1_mux_gt,soma_2_mux_gt)begin
130         if(controle_gt)
131             temp = soma_2_mux_gt; // saida do somador_gt
132         else
133             temp = soma_1_mux_gt; // saida do somador_jump
134     end
```

```
135 endmodule
136
137 // MUX JUMP
138 module mux_jump(bloco_controle,saida_mux_jump);
139
140     input bloco_controle;
141     output [7:0]saida_mux_jump;
142     reg [7:0]temp;
143
144     assign saida_mux_jump = temp;
145
146     always @(bloco_controle)begin
147         if(bloco_controle)
148             temp = 8'b00000010;
149         else
150             temp = 8'b00000001;
151     end
152 endmodule
153
154 // MUX A
155 module mux_a(controle_a,entrada_1_mux_a,entrada_2_mux_a,entrada_3_mux_a,saida_mux_a);
156
157     input [1:0] controle_a;
158     input [3:0] entrada_1_mux_a;
159     input [7:0] entrada_2_mux_a,entrada_3_mux_a;
160     output [7:0] saida_mux_a;
161     reg [7:0]temp;
162
163     assign saida_mux_a = temp;
164
165     always @(controle_a,entrada_1_mux_a,entrada_2_mux_a,entrada_3_mux_a) begin
166         case(controle_a)
167             2'b00 : temp = entrada_1_mux_a;  // valor vindo da instrução
168             2'b01 : temp = entrada_2_mux_a;  // valor vindo da ULA
169             2'b10 : temp = entrada_3_mux_a;  // valor vindo da Memoria
170         endcase
171     end
172 endmodule
173
174 // DEMUX B
175 module demux_b(controle_b,entrada_demux_b,saida_1_demux_b,saida_2_demux_b);
176
177     input controle_b;
178     input [7:0] entrada_demux_b;
179     output [7:0] saida_1_demux_b,saida_2_demux_b;
180     reg [7:0]temp_1,temp_2;
181
182     assign saida_1_demux_b = temp_1;
183     assign saida_2_demux_b = temp_2;
184
185     always@(controle_b,entrada_demux_b)begin
186         if(controle_b)
187             temp_1 = entrada_demux_b;  // valor para banco de registradores
188         else
189             temp_2 = entrada_demux_b;  // valor para a Memoria
190     end
191 endmodule
192
193 // MUX ULA
194 module mux_ula(controle_input_ula,entrada_1_demux_ula,entrada_2_demux_ula,saida_demux_ula);
195
196     input [1:0]controle_input_ula;
197     input [7:0] entrada_1_demux_ula;
198     input [3:0] entrada_2_demux_ula;
199     output [7:0] saida_demux_ula;
200     reg [7:0] temp;
201
202     assign saida_demux_ula = temp;
203
```

```
204       always@(controle_input_ula,entrada_1_demux_ula,entrada_2_demux_ula)begin
205           case(controle_input_ula)
206               2'b00 : temp = entrada_1_demux_ula; // valor vindo do banco de resgistrador
207               2'b01 : temp = entrada_2_demux_ula; // valor vindo da instrução
208               2'b10 : temp = 0;
209           endcase
210       end
211   endmodule
212
213   // DEMUX HALT
214   module demux_halt(controle_halt,entrada_demux_halt,saida_1_demux_halt,saida_2_demux_halt);
215
216       input controle_halt;
217       input [7:0] entrada_demux_halt;
218       output [7:0] saida_1_demux_halt,saida_2_demux_halt;
219       reg [7:0]temp_1,temp_2;
220
221       assign saida_1_demux_halt = temp_1;
222       assign saida_2_demux_halt = temp_2;
223
224
225       always@(controle_halt,entrada_demux_halt) begin
226           if(controle_halt)
227               temp_2 = entrada_demux_halt; //halt ligado
228           else
229               temp_1 = 0;   // halt desligado
230       end
231   endmodule
232
233   // ULA GT
234   module ULA_GT(controle,valor1_somador_gt,valor2_somador_gt,saida_somador_gt);
235       input controle;
236       input [7:0]valor1_somador_gt;
237       input [3:0]valor2_somador_gt;
238       output [7:0]saida_somador_gt;
239       reg [7:0]temp;
240
241       assign saida_somador_gt = temp;
242
243       always@(valor1_somador_gt,valor2_somador_gt) begin
244           if(controle)
245               temp = valor1_somador_gt - valor2_somador_gt;  //sub
246           else
247               temp = valor1_somador_gt + valor2_somador_gt;  //add
248       end
249   endmodule
250
251   // SOMADOR JUMP
252   module somador_jump(valor1_somador_jump,valor2_somador_jump,saida_somador_jump);
253
254       input [7:0]valor1_somador_jump,valor2_somador_jump;
255       output [7:0]saida_somador_jump;
256       reg [7:0] temp;
257
258       assign saida_somador_jump = temp;
259
260       always @(valor1_somador_jump,valor2_somador_jump)begin
261           temp = valor1_somador_jump + valor2_somador_jump;
262       end
263   endmodule
264
265   // BLOCO DE CONTROLE
266   module bloco_controle(zero,neg,controle_jumps,saida_bloco_controle);
267
268       input zero,neg,controle_jumps;
269       output saida_bloco_controle;
270       reg temp;
271
272       assign saida_bloco_controle = temp;
```

```
273
274         always@(zero or neg or controle_jumps) begin
275             temp = controle_jumps & (~zero | ~neg);
276         end
277
278     endmodule
```