```
1   module RAM_instrucoes (end_linha, end_coluna, clock, saida);
2
3   parameter tamanho = 30; // regula o tamanho da matriz de memoria
4   input [10:0] end_linha;
5   input [10:0] end_coluna;
6   input clock;
7   output reg [31:0] saida;
8
9   integer firstclock = 0;
10
11  reg [31:0] ram_instrucoes[tamanho-1:0][tamanho-1:0];
12
13  always @ (posedge clock) begin
14     if(firstclock == 0)begin
15
16
17
18        // teste entrada/saida de dados ok
19        //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
20        //ram_instrucoes[0][1] <= 32'b11000000010000000000000000000000;//IN ->
    reg[1]
21        //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//out
    reg[1] -> saida[0][0]
22        //ram_instrucoes[0][3] <= 32'b10010000000000000000000000000000;//JUMP para
    endereco em reg[0] = linha[0] coluna[0]
23        //ram_instrucoes[0][4] <= 32'b11011000000000000000000000000000;//HLT
24
25
26        //teste1 IN OUT, LOGICAS e STORE LOAD ok
27        //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
28        //ram_instrucoes[0][1] <= 32'b11000000010000000000000000000000;//IN ->
    reg[1]
29        //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//OUT
    reg[1] -> saida[0][0]
30        //ram_instrucoes[0][3] <= 32'b10000000010000000000000000000001;//STORE
    reg[1] = memoria[0][1]
31        //ram_instrucoes[0][4] <= 32'b01110000010000000000000000000001;//LOAD
    memoria[0][1] -> reg[2]
32        //ram_instrucoes[0][5] <= 32'b11001000010000000000000000000001;//OUT
    reg[2] -> saida[0][1]
33        //ram_instrucoes[0][6] <= 32'b00101000110001000000000000000000;//AND
    reg[3] <- reg[2] and reg[0]
34        //ram_instrucoes[0][7] <= 32'b11001000110000000000000000000010;//OUT
    reg[3] -> saida[0][2]
35        //ram_instrucoes[0][8] <= 32'b00111000110001000000000000000000;//OR reg[3]
    <- reg[2] or reg[0]
36        //ram_instrucoes[0][9] <= 32'b11001000110000000000000000000010;//OUT
    reg[3] -> saida[0][2]
37        //ram_instrucoes[0][10] <= 32'b01001000110001000000000000000000;//XOR
    reg[3] <- reg[2] xor reg[0]
38        //ram_instrucoes[0][11] <= 32'b11001000110000000000000000000010;//OUT
    reg[3] -> saida[0][2]
39        //ram_instrucoes[0][12] <= 32'b01011000111000100000000000000000;//NOT
    reb[3] <- not reg[2]
40        //ram_instrucoes[0][13] <= 32'b11001000110000000000000000000010;//OUT
    reg[3] -> saida[0][2]
41        //ram_instrucoes[0][14] <= 32'b11001000000000000000000000000000;//OUT
    reg[0] -> saida[0][0]
42        //ram_instrucoes[0][15] <= 32'b11001000000000000000000000000001;//OUT
    reg[0] -> saida[0][1]
43        //ram_instrucoes[0][16] <= 32'b11010000000000000000000000000000;//NOP
44        //ram_instrucoes[0][17] <= 32'b00110000110000000000000000000000;//ANDI
```

```
      reg[3] <- andi reg[0]
45          //ram_instrucoes[0][18] <= 32'b11001000110000000000000000000010;//OUT
      reg[3] -> saida[0][2]
46          //ram_instrucoes[0][19] <= 32'b11010000000000000000000000000000;//NOP
47          //ram_instrucoes[0][20] <= 32'b01000000110000000000000000000000;//ORI
      reg[3] <- ori reg[0]
48          //ram_instrucoes[0][21] <= 32'b11001000110000000000000000000010;//OUT
      reg[3] -> saida[0][2]
49          //ram_instrucoes[0][22] <= 32'b11010000000000000000000000000000;//NOP
50          //ram_instrucoes[0][23] <= 32'b01010000110000000000000000000000;//XORI
      reg[3] <- ori reg[0]
51          //ram_instrucoes[0][24] <= 32'b11001000110000000000000000000010;//OUT
      reg[3] -> saida[0][2]
52          //ram_instrucoes[0][25] <= 32'b11011000000000000000000000000000;//HLT
53
54          //teste2 SL SR MOVE ok
55          //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
56          //// com o numero 2 como entrada
57          //ram_instrucoes[0][1] <= 32'b11000000010000000000000000000000;//IN ->
      reg[1]
58          //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000001;//OUT
      reg[1] -> saida[0][1]
59          //ram_instrucoes[0][3] <= 32'b10001000100000100000000000000000;//MOVE
      reg[2] <- reg[1]
60          //ram_instrucoes[0][4] <= 32'b11001000100000000000000000000010;//OUT
      reg[2] -> saida[0][2]
61          //ram_instrucoes[0][5] <= 32'b01101000110001000000000000000000;//SL reg[3]
      <- sl reg[2]
62          //ram_instrucoes[0][6] <= 32'b11001000110000000000000000000010;//OUT
      reg[3] -> saida[0][2]
63          //ram_instrucoes[0][7] <= 32'b01100000100001100000000000000000;//SR reg[2]
      <- sr reg[3]
64          //ram_instrucoes[0][8] <= 32'b11001000100000000000000000000010;//OUT
      reg[2] -> saida[0][2]
65          //ram_instrucoes[0][9] <= 32'b10010000000011000000000000000000;//JUMP
      PC[0][5] (reg[6])
66          //ram_instrucoes[0][10] <= 32'b11011000000000000000000000000000;//HLT
67
68          //teste3 ADD ok
69          //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
70          //ram_instrucoes[0][1] <= 32'b11000000010000000000000000000000;//IN ->
      reg[1]
71          //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//OUT
      reg[1] -> saida[0][0]
72          //ram_instrucoes[0][3] <= 32'b11010000000000000000000000000000;//NOP
73          //ram_instrucoes[0][4] <= 32'b11000000010000000000000000000000;//IN ->
      reg[2]
74          //ram_instrucoes[0][5] <= 32'b11001000100000000000000000000001;//OUT
      reg[2] -> saida[0][1]
75          //ram_instrucoes[0][6] <= 32'b00000000110000100010000000000000;//ADD
      reg[3] <- reg[1]+reg[2]
76          //ram_instrucoes[0][7] <= 32'b11001000110000000000000000000010;//OUT
      reg[3] -> saida[0][2]
77          //ram_instrucoes[0][8] <= 32'b10010000000000000000000000000000;//JUMP
      PC[0][0](reg[0])
78
79          //teste SUB ok
80          //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
81          //ram_instrucoes[0][1] <= 32'b11000000010000000000000000000000;//IN ->
      reg[1]
82          //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//OUT
      reg[1] -> saida[0][0]
```

```
83          //ram_instrucoes[0][3] <= 32'b11010000000000000000000000000000;//NOP
84          //ram_instrucoes[0][4] <= 32'b11000000100000000000000000000000;//IN ->
    reg[2]
85          //ram_instrucoes[0][5] <= 32'b11001000100000000000000000000001;//OUT
    reg[2] -> saida[0][1]
86          //ram_instrucoes[0][6] <= 32'b00010000110000100010000000000000;//SUB
    reg[3] <- reg[1]-reg[2]
87          //ram_instrucoes[0][7] <= 32'b11001000110000000000000000000010;//OUT
    reg[3] -> saida[0][2]
88          //ram_instrucoes[0][8] <= 32'b10010000000000000000000000000000;//JUMP
    PC[0][0](reg[0])
89
90
91          //teste MUL ok
92          //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
93          //ram_instrucoes[0][1] <= 32'b11000000100000000000000000000000;//IN ->
    reg[1]
94          //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//OUT
    reg[1] -> saida[0][0]
95          //ram_instrucoes[0][3] <= 32'b11010000000000000000000000000000;//NOP
96          //ram_instrucoes[0][4] <= 32'b11000000100000000000000000000000;//IN ->
    reg[2]
97          //ram_instrucoes[0][5] <= 32'b11001000100000000000000000000001;//OUT
    reg[2] -> saida[0][1]
98          //ram_instrucoes[0][6] <= 32'b00100000110000100010000000000000;//MUL
    reg[3] <- reg[1]x reg[2]
99          //ram_instrucoes[0][7] <= 32'b11001000110000000000000000000010;//OUT
    reg[3] -> saida[0][2]
100         //ram_instrucoes[0][8] <= 32'b10010000000000000000000000000000;//JUMP
    PC[0][0](reg[0])
101
102         //teste ADDI SUBI ok
103         //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
104         //ram_instrucoes[0][1] <= 32'b11000000100000000000000000000000;//IN ->
    reg[1]
105         //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000010;//OUT
    reg[1] -> saida[0][2]
106         //ram_instrucoes[0][3] <= 32'b11010000000000000000000000000000;//NOP
107         //ram_instrucoes[0][4] <= 32'b00001000100000100000000000000000;//ADDI
    reg[2] <- addi reg[1]
108         //ram_instrucoes[0][5] <= 32'b11001000100000000000000000000010;//OUT
    reg[2] -> saida[0][2]
109         //ram_instrucoes[0][6] <= 32'b11010000000000000000000000000000;//NOP
110         //ram_instrucoes[0][7] <= 32'b00011000010001000000000000000000;//SUBI
    reg[1] <- subi reg[2]
111         //ram_instrucoes[0][8] <= 32'b11001000010000000000000000000010;//OUT
    reg[1] -> saida[0][2]
112         //ram_instrucoes[0][9] <= 32'b10010000000010100000000000000000;//JUMP
    PC[0][4](reg[5])
113
114         //teste JUMPZ ok
115         //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
116         //ram_instrucoes[0][1] <= 32'b11000000100000000000000000000000;//IN ->
    reg[1]
117         //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//OUT
    reg[1] -> saida[0][0]
118         //ram_instrucoes[0][3] <= 32'b11010000000000000000000000000000;//NOP
119         //ram_instrucoes[0][4] <= 32'b00011000100000100000000000000000;//SUBI
    reg[2] <- subi reg[1]
120         //ram_instrucoes[0][5] <= 32'b10110000000000000000000000000000;//JUMPZ
    PC[0][0](reg[0])
121         //ram_instrucoes[0][6] <= 32'b11001000010000000000000000000001;//OUT
```

```
      reg[1] -> saida[0][0]
122         //ram_instrucoes[0][7] <= 32'b11011000000000000000000000000000;//HLT
123
124
125         //teste JUMPN ok
126         //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
127         //ram_instrucoes[0][1] <= 32'b11000000010000000000000000000000;//IN ->
      reg[1]
128         //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//OUT
      reg[1] -> saida[0][0]
129         //ram_instrucoes[0][3] <= 32'b11010000000000000000000000000000;//NOP
130         //ram_instrucoes[0][4] <= 32'b00011000100000100000000000000000;//SUBI
      reg[2] <- subi reg[1]
131         //ram_instrucoes[0][5] <= 32'b10100000000000000000000000000000;//JUMPN
      PC[0][0](reg[0])
132         //ram_instrucoes[0][6] <= 32'b11001000010000000000000000000001;//OUT
      reg[1] -> saida[0][0]
133         //ram_instrucoes[0][7] <= 32'b11011000000000000000000000000000;//HLT
134
135
136         //teste LOADI JUMPI ok
137         //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
138         //ram_instrucoes[0][1] <= 32'b11000000010000000000000000000000;//IN ->
      reg[1]
139         //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//OUT
      reg[1] -> saida[0][0]
140         //ram_instrucoes[0][3] <= 32'b10000000010000000000000000000001;//STORE
      reg[1] = memoria[0][1]
141         ////com o numero 1 como entrada
142         //ram_instrucoes[0][4] <= 32'b11010000000000000000000000000000;//NOP
143         //ram_instrucoes[0][5] <= 32'b01111000100000000000000000000000;//LOADI
      memoria[0][1] -> reg[2]
144         //ram_instrucoes[0][6] <= 32'b11001000100000000000000000000001;//OUT
      reg[2] -> saida[0][1]
145         //ram_instrucoes[0][7] <= 32'b11010000000000000000000000000000;//NOP
146         //ram_instrucoes[0][8] <= 32'b10011000000000000000000000000000;//JUMPI
147
148
149         //teste JUMPNI JUMPZI ok
150         //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
151         //ram_instrucoes[0][1] <= 32'b11000000010000000000000000000000;//IN ->
      reg[1]
152         //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//OUT
      reg[1] -> saida[0][0]
153         //ram_instrucoes[0][3] <= 32'b11010000000000000000000000000000;//NOP
154         //ram_instrucoes[0][4] <= 32'b00011000100000100000000000000000;//SUBI
      reg[2] <- subi reg[1]
155         ////com o numero 10 como entrada
156         //ram_instrucoes[0][5] <= 32'b11010000000000000000000000000000;//NOP
157         //ram_instrucoes[0][6] <= 32'b10101000000000000000000000000000;//JUMPNI
158         ////com o numero 12 como entrada
159         //ram_instrucoes[0][7] <= 32'b11010000000000000000000000000000;//NOP
160         //ram_instrucoes[0][8] <= 32'b10111000000000000000000000000000;//JUMPZI
161         //ram_instrucoes[0][9] <= 32'b11011000000000000000000000000000;//HLT
162         //ram_instrucoes[0][10] <= 32'b11001000010000000000000000000010;//OUT
      reg[1] -> saida[0][2]
163         //ram_instrucoes[0][11] <= 32'b11011000000000000000000000000000;//HLT
164         //ram_instrucoes[0][12] <= 32'b11001000000000000000000000000010;//OUT
      reg[0] -> saida[0][2]
165         //ram_instrucoes[0][13] <= 32'b11011000000000000000000000000000;//HLT
166
167
```

```
168         //teste pre fibonacci ok
169         //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
170         //ram_instrucoes[0][1] <= 32'b11000000010000000000000000000000;//IN ->
    reg[1]
171         //ram_instrucoes[0][2] <= 32'b11001000010000000000000000000000;//OUT
    reg[1] -> saida[0][0]
172         //ram_instrucoes[0][3] <= 32'b11010000000000000000000000000000;//NOP
173         //ram_instrucoes[0][4] <= 32'b11000000010000000000000000000000;//IN ->
    reg[2]
174         //ram_instrucoes[0][5] <= 32'b11001000010000000000000000000001;//OUT
    reg[2] -> saida[0][1]
175         //ram_instrucoes[0][6] <= 32'b00010000010000100010000000000000;//SUB
    reg[1] <- reg[1]-reg[2]
176         //ram_instrucoes[0][7] <= 32'b11001000010000000000000000000010;//OUT
    reg[1] -> saida[0][2]
177         //ram_instrucoes[0][8] <= 32'b00010000010000100010000000000000;//SUB
    reg[1] <- reg[1]-reg[2]
178         //ram_instrucoes[0][9] <= 32'b11001000010000000000000000000010;//OUT
    reg[1] -> saida[0][2]
179         //ram_instrucoes[0][10] <= 32'b11011000000000000000000000000000;//HLT
180
181
182
183
184
185         // teste Fibonacci --
186         //ram_instrucoes[0][0] <= 32'b11010000000000000000000000000000;//NOP
187         //ram_instrucoes[0][1] <= 32'b11010000000000000000000000000000;//NOP
188         //ram_instrucoes[0][2] <= 32'b11000000010000000000000000000000;//IN ->
    reg[1]
189         //ram_instrucoes[0][3] <= 32'b11001000010000000000000000000000;//OUT
    reg[1] -> saida[0][0]
190         //ram_instrucoes[0][4] <= 32'b10001000010000100000000000000000;//MOVE
    reg[1] <- reg[1]
191         //ram_instrucoes[0][5] <= 32'b10110000000011110000000000000000;//JZ -> PC
    [0][25] (reg[15])
192         //ram_instrucoes[0][6] <= 32'b00010000010000100111000000000000;//SUB
    reg[1] <- reg[1] - 2 (reg[7])
193         //ram_instrucoes[0][7] <= 32'b10110000000011000000000000000000;//JZ -> PC
    [0][19] (reg[12])
194         //ram_instrucoes[0][8] <= 32'b10100000000011000000000000000000;//JN -> PC
    [0][19] (reg[12])
195         //ram_instrucoes[0][9] <= 32'b00000010110101001001000000000000;//ADD
    reg[11] <- reg[10]+reg[9]
196         //ram_instrucoes[0][10] <= 32'b00010000010000101000000000000000;//SUB
    reg[1] <- reg[1]-reg[8]
197         //ram_instrucoes[0][11] <= 32'b10110000000011100000000000000000;//JZ -> PC
    [0][23] (reg[14])
198         //ram_instrucoes[0][12] <= 32'b00000010010101001011000000000000;//ADD
    reg[9] <- reg[10]+reg[11]
199         //ram_instrucoes[0][13] <= 32'b00010000010000101000000000000000;//SUB
    reg[1] <- reg[1]-reg[8]
200         //ram_instrucoes[0][14] <= 32'b10110000000011000000000000000000;//JZ -> PC
    [0][19] (reg[12])
201         //ram_instrucoes[0][15] <= 32'b00000010100101101001000000000000;//ADD
    reg[10] <- reg[11]+reg[9]
202         //ram_instrucoes[0][16] <= 32'b00010000010000101000000000000000;//SUB
    reg[1] <- reg[1]-reg[8]
203         //ram_instrucoes[0][17] <= 32'b10110000000011010000000000000000;//JZ -> PC
    [0][21] (reg[13])
204         //ram_instrucoes[0][18] <= 32'b10010000001000000000000000000000;//JUMP PC
    <- [0][9] (reg[16])
```

```
205         //ram_instrucoes[0][19] <= 32'b11001010010000000000000000000010;//OUT
    reg[9] -> saida[0][2]
206         //ram_instrucoes[0][20] <= 32'b11011000000000000000000000000000;//HLT
207         //ram_instrucoes[0][21] <= 32'b11001010100000000000000000000010;//OUT
    reg[10] -> saida[0][2]
208         //ram_instrucoes[0][22] <= 32'b11011000000000000000000000000000;//HLT
209         //ram_instrucoes[0][23] <= 32'b11001010110000000000000000000010;//OUT
    reg[11] -> saida[0][2]
210         //ram_instrucoes[0][24] <= 32'b11011000000000000000000000000000;//HLT
211         //ram_instrucoes[0][25] <= 32'b11001000000000000000000000000010;//OUT
    reg[0] -> saida[0][2]
212         //ram_instrucoes[0][26] <= 32'b11011000000000000000000000000000;//HLT
213
214
215         firstclock <= 1;
216      end
217   end
218
219   always @(end_linha or end_coluna) begin
220      saida <= ram_instrucoes[end_linha][end_coluna];
221   end
222   endmodule
```