

# **Relatório - Trabalho 4**

## **Organização e Arquitetura de Computadores - Turma C**

**Gabriel Vieira de Arimatéa, 15/0126956**

<sup>1</sup>CIC – Universidade de Brasília (UnB)

vieira.arimatea@gmail.com

### **1. Descrição do problema**

O trabalho consiste em projetar, simular e sintetizar uma versão da ULA do MIPS de 32 bits no ambiente Quartus / ModelSim-Altera.

### **2. Descrição sucinta do trabalho**

A ULA é uma peça fundamental da arquitetura de um processador. É nela que as operações são interpretadas e realizadas. Para este trabalho, foi necessário criar uma ULA utilizando linguagem VHDL. Suas entradas e saídas são:

- **A e B:** entradas de dados;
- **Z:** saída de dados;
- **Zero:** detecta valor zero na saída;
- **Overflow:** ativo quando a operação de soma ou subtração gerar resultado que ultrapasse o limite de representação em 32 bits;
- **Opcode:** entrada com o opcode da instrução. Determina a operação que deve ser realizada.

### **3. Códigos e resultados**

#### **3.1. Código da ULA**

```

1  library ieee;
2  use ieee.numeric_std.all;
3  use ieee.std_logic_1164.all;
4  use ieee.numeric_std.all;
5
6  entity ulaMIPS is
7      generic (WSIZE          : natural := 32);
8      port (
9          a, b                : in std_logic_vector (WSIZE-1 downto 0);
10         opcode              : in std_logic_vector (3 downto 0);
11         z                   : out std_logic_vector (WSIZE-1 downto 0);
12         zero, ovfl         : out std_logic
13     );
14 end ulaMIPS;
15
16 architecture Behavioral of ulaMIPS is
17     signal result: std_logic_vector (WSIZE-1 downto 0);
18     signal overflow: std_logic;
19
20
21     begin
22     ula_mips: process(opcode,a,b)
23     variable n : integer;
24     begin
25     overflow <= '0';
26     case opcode is
27     when "0000" =>                -- and
28         result <= a and b;
29
30     when "0001" =>                -- or
31         result <= a or b;
32
33     when "0010" =>                -- add
34         result <= std_logic_vector(unsigned(a) + unsigned(b));
35         overflow <= (a(WSIZE-1) and b(WSIZE-1) and not(result(WSIZE-1))) or (not(a(WSIZE

```

```

-1)) and not(b(WSIZE-1)) and result(WSIZE-1));
36
37     when "0011" =>                -- addu
38         result <= std_logic_vector(unsigned(a) + unsigned(b));
39
40     when "0100" =>                -- sub
41         result <= std_logic_vector(unsigned(a) - unsigned(b));
42         overflow <= (a(WSIZE-1) and b(WSIZE-1) and not(result(WSIZE-1))) or (not(a(WSIZE
-1)) and not(b(WSIZE-1)) and result(WSIZE-1));
43
44     when "0101" =>                -- subu
45         result <= std_logic_vector(unsigned(a) - unsigned(b));
46
47     when "0110" =>                -- slt
48         if (a<b) then
49             result <= x"00000001";
50         else
51             result <= x"00000000";
52         end if;
53
54     when "0111" =>                -- sltu
55         if (unsigned(a)<unsigned(b)) then
56             result <= x"00000001";
57         else
58             result <= x"00000000";
59         end if;
60
61     when "1000" =>                -- nor
62         result <= a nor b;
63
64     when "1001" =>                -- xor

```

```

65         result <= a xor b;
66
67     when "1010" =>                -- sll
68         result <= std_logic_vector (shift_left (unsigned (b), to_integer (unsigned (a))));
69
70     when "1011" =>                -- srl
71         result <= std_logic_vector (shift_right (unsigned (b), to_integer (unsigned (A))));
72
73     when "1100" =>                -- sra
74         result <= std_logic_vector (shift_right (signed (b), to_integer (unsigned (A))));
75
76     when "1101" =>                -- clz
77         n := 0;
78         for i in a'range loop
79             if a(i) = '0' then
80                 n := n + 1;
81             end if;
82         end loop;
83         result <= std_logic_vector (to_unsigned (n, result'length));
84
85     when "1110" =>                -- clo
86         n := 0;
87         for i in a'range loop
88             if a(i) = '1' then
89                 n := n + 1;
90             end if;
91         end loop;
92         result <= std_logic_vector (to_unsigned (n, result'length));
93
94     when others => result <= std_logic_vector (unsigned (a) + unsigned (b)); -- add
95
96 end case;
97 end process;
98 zero <= '1' when result=x"00000000" else '0';
99 z <= result;
100 ovfl <= overflow;
101 end Behavioral;

```

---

## 3.2. Código do Test Bench

```
1  -- Copyright (C) 1991-2013 Altera Corporation
2  -- Your use of Altera Corporation's design tools, logic functions
3  -- and other software and tools, and its AMPP partner logic
4  -- functions, and any output files from any of the foregoing
5  -- (including device programming or simulation files), and any
6  -- associated documentation or information are expressly subject
7  -- to the terms and conditions of the Altera Program License
8  -- Subscription Agreement, Altera MegaCore Function License
9  -- Agreement, or other applicable license agreement, including,
10 -- without limitation, that your use is for the sole purpose of
11 -- programming logic devices manufactured by Altera and sold by
12 -- Altera or its authorized distributors. Please refer to the
13 -- applicable agreement for further details.
14
15 -- *****
16 -- This file contains a Vhdl test bench template that is freely editable to
17 -- suit user's needs .Comments are provided in each section to help the user
18 -- fill out necessary details.
19 -- *****
20 -- Generated on "11/15/2018 22:13:40"
21
22 -- Vhdl Test Bench template for design :  ulaMIPS
23 --
24 -- Simulation tool : ModelSim-Altera (VHDL)
25 --
26
27 LIBRARY ieee;
28 USE ieee.std_logic_1164.all;
29
30 ENTITY ulaMIPS_vhd_tst IS
31 END ulaMIPS_vhd_tst;
32 ARCHITECTURE ulaMIPS_arch OF ulaMIPS_vhd_tst IS
33 -- constants
34 -- signals
35 SIGNAL a : STD_LOGIC_VECTOR (31 DOWNT0 0);
```

```

36 SIGNAL b : STD_LOGIC_VECTOR (31 DOWNT0 0);
37 SIGNAL opcode : STD_LOGIC_VECTOR (3 DOWNT0 0);
38 SIGNAL ovfl : STD_LOGIC;
39 SIGNAL z : STD_LOGIC_VECTOR (31 DOWNT0 0);
40 SIGNAL zero : STD_LOGIC;
41 COMPONENT ulaMIPS
42     PORT (
43         a : IN STD_LOGIC_VECTOR (31 DOWNT0 0);
44         b : IN STD_LOGIC_VECTOR (31 DOWNT0 0);
45         opcode : IN STD_LOGIC_VECTOR (3 DOWNT0 0);
46         ovfl : OUT STD_LOGIC;
47         z : OUT STD_LOGIC_VECTOR (31 DOWNT0 0);
48         zero : OUT STD_LOGIC
49     );
50 END COMPONENT;
51 BEGIN
52     il : ulaMIPS
53     PORT MAP (
54         -- list connections between master ports and signals
55         a => a,
56         b => b,
57         opcode => opcode,
58         ovfl => ovfl,
59         z => z,
60         zero => zero
61     );
62     init : PROCESS
63         -- variable declarations
64     BEGIN
65         -- code that executes only once
66         -- Teste AND

```

```
67     opcode <= "0000";
68
69     a <= x"FF00FF00";
70     b <= x"F00FF00F";
71     wait for 5 ns;
72
73     -- Teste OR
74     opcode <= "0001";
75
76     a <= x"FF00FF00";
77     b <= x"F00FF00F";
78     wait for 5 ns;
79
80     -- Teste ADD
81     opcode <= "0010";
82
83     a <= x"0000000A";
84     b <= x"00000001";
85     wait for 5 ns;
86
87     a <= x"F0000000";
88     b <= x"0FFFFFFF";
89     wait for 5 ns;
90
91     a <= x"5FFFFFFF";
92     b <= x"5FFFFFFF";
93     wait for 5 ns;    -- overflow
94
95     a <= x"80000000";
96     b <= x"80000000";
97     wait for 5 ns;    -- overflow
98
99     a <= x"00000000";
100    b <= x"00000000";
101    wait for 5 ns;    -- zero
102
103    -- Teste ADDU
```

---

```

104     opcode <= "0011";
105
106     a <= x"0000000A";
107     b <= x"00000001";
108     wait for 5 ns;
109
110     a <= x"F0000000";
111     b <= x"0FFFFFFF";
112     wait for 5 ns;
113
114     a <= x"5FFFFFFF";
115     b <= x"5FFFFFFF";
116     wait for 5 ns;    -- overflow
117
118     a <= x"80000000";
119     b <= x"80000000";
120     wait for 5 ns;    -- overflow
121
122     a <= x"00000000";
123     b <= x"00000000";
124     wait for 5 ns;    -- zero
125
126     -- Teste SUB
127     opcode <= "0100";
128
129     a <= x"00000000";
130     b <= x"A0000000";
131     wait for 5 ns;
132

```

```

133     a <= x"A0000000";
134     b <= x"60000000";
135     wait for 5 ns;
136
137     a <= x"80000000";
138     b <= x"80000000";
139     wait for 5 ns;    -- overflow
140
141     a <= x"5FFFFFFF";
142     b <= x"5FFFFFFF";
143     wait for 5 ns;    -- overflow
144
145     a <= x"A0000000";
146     b <= x"A0000000";
147     wait for 5 ns;    --zero
148
149     -- Teste SUBU
150     opcode <= "0101";
151
152     a <= x"00000000";
153     b <= x"A0000000";
154     wait for 5 ns;
155
156     a <= x"A0000000";
157     b <= x"60000000";
158     wait for 5 ns;
159
160     a <= x"80000000";
161     b <= x"80000000";
162     wait for 5 ns;    -- overflow
163
164     a <= x"5FFFFFFF";
165     b <= x"5FFFFFFF";
166     wait for 5 ns;    -- overflow
167
168     a <= x"A0000000";
169     b <= x"A0000000";

```



```

170         wait for 5 ns;      --zero
171
172     -- teste SLT
173     opcode <= "0110";
174
175     a <= x"000000C0";
176     b <= x"000000A0";
177     wait for 5 ns;
178
179     a <= x"000000A0";
180     b <= x"000000C0";
181     wait for 5 ns;
182
183     a <= x"000000C0";
184     b <= x"000000C0";
185     wait for 5 ns;
186
187     a <= x"F00000C0";
188     b <= x"000000A0";
189     wait for 5 ns;
190
191     a <= x"000000C0";
192     b <= x"F00000A0";
193     wait for 5 ns;
194
195     a <= x"F00000C0";
196     b <= x"F00000A0";
197     wait for 5 ns;
198

```

---

```

199     -- teste SLTU
200     opcode <= "0111";
201
202     a <= x"000000C0";
203     b <= x"000000A0";
204     wait for 5 ns;
205
206     a <= x"000000A0";
207     b <= x"000000C0";
208     wait for 5 ns;
209
210     a <= x"000000C0";
211     b <= x"000000C0";
212     wait for 5 ns;
213
214     a <= x"F00000C0";
215     b <= x"000000A0";
216     wait for 5 ns;
217
218     a <= x"000000C0";
219     b <= x"F00000A0";
220     wait for 5 ns;
221
222     a <= x"F00000C0";
223     b <= x"F00000A0";
224     wait for 5 ns;
225
226     -- Teste NOR
227     opcode <= "1000";
228
229     a <= x"FF00FF00";
230     b <= x"F00FF00F";
231     wait for 5 ns;
232
233     a <= x"00000000";
234     b <= x"00000000";
235     wait for 5 ns;
236

```

```

236
237     -- Teste XOR
238     opcode <= "1001";
239
240     a <= x"FF00FF00";
241     b <= x"00000000";
242     wait for 5 ns;
243
244     a <= x"FF00FF00";
245     b <= x"F00FF00F";
246     wait for 5 ns;
247
248     -- Teste SLL
249     opcode <= "1010";
250
251     a <= x"00000100";
252     b <= x"FFFFFFFF";
253     wait for 5 ns;
254
255     a <= x"00000010";
256     b <= x"FFFFFFFF";
257     wait for 5 ns;
258
259     -- Teste SRL
260     opcode <= "1011";
261
262     a <= x"00000100";
263     b <= x"FFFFFFFF";
264     wait for 5 ns;

```

```

--
302     a <= x"FFFFFFF0";
303     wait for 5 ns;
304
305 WAIT;
306 END PROCESS init;
307 always : PROCESS
308 -- optional sensitivity list
309 -- (
310 -- variable declarations
311 BEGIN
312     -- code executes for every event on sensitivity list
313 WAIT;
314 END PROCESS always;
315 END ulaMIPS_arch;
316

```

3.3. Resultado simulação (ModelSim)

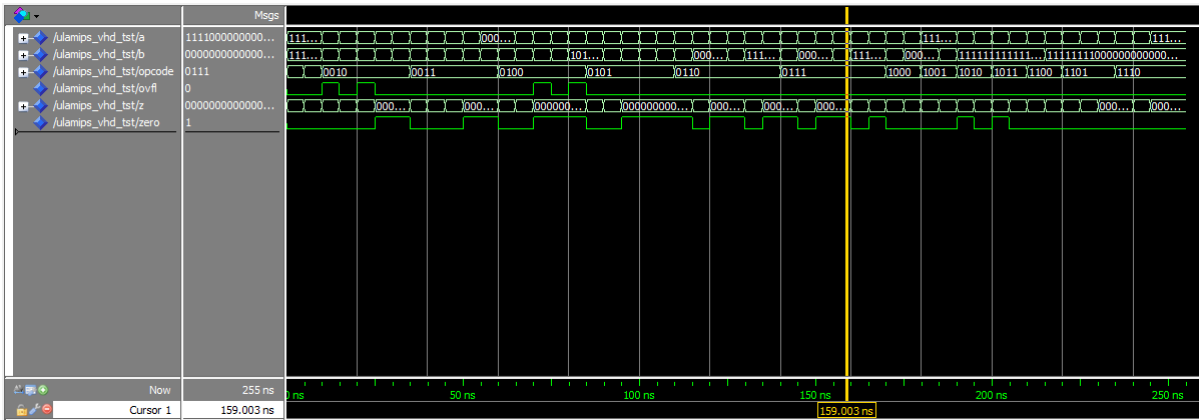


Figura 1. Resultado final tota

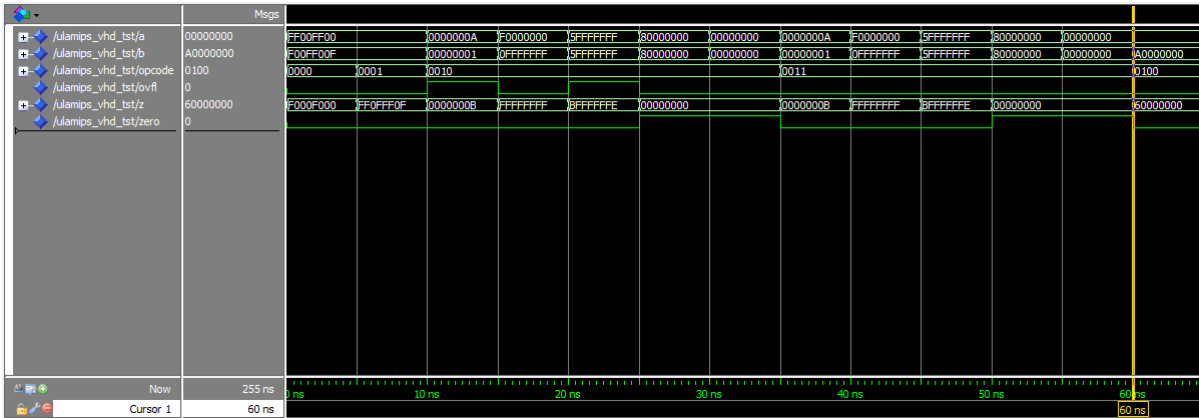


Figura 2. Parte 1



#### **4. Observação**

O trabalho foi entregue com uma semana de atraso com a permissão do professor. Expliquei para ele que estava em uma competição e que não seria possível dedicar tempo para este trabalho na semana que era para o mesmo ser realizado.