

Reporte de practica 11

González Pardo Adrian

Mayo 2020

1. Código fuente:

1.1. Memoria de programa

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_arith.ALL;
4 use IEEE.STD_LOGIC_unsigned.ALL;
5
6 entity memPrograma is
7     generic(
8         m:integer:= 10;--tamano del bus de direcciones
9         n:integer:= 25--tamano de palabra
10    );
11    Port(dir:in STD_LOGIC_VECTOR (m-1 downto 0);
12          inst:out STD_LOGIC_VECTOR (n-1 downto 0));
13 end memPrograma;
14
15 architecture Behavioral of memPrograma is
16     --C-OPERACION
17     constant TYPER: std_logic_vector(4 downto 0):="00000";
18     constant LI: std_logic_vector(4 downto 0):="00001";
19     constant LWI: std_logic_vector(4 downto 0):="00010";
20     constant LW: std_logic_vector(4 downto 0):="10111";
21     constant SWI: std_logic_vector(4 downto 0):="00011";
22     constant SW: std_logic_vector(4 downto 0):="00100";
23     constant ADDI: std_logic_vector(4 downto 0):="00101";
24     constant SUBI: std_logic_vector(4 downto 0):="00110";
25     constant ANDI: std_logic_vector(4 downto 0):="00111";
26     constant ORI: std_logic_vector(4 downto 0):="01000";
27     constant XORI: std_logic_vector(4 downto 0):="01001";
28     constant NANDI: std_logic_vector(4 downto 0):="01010";
29     constant NORI: std_logic_vector(4 downto 0):="01011";
30     constant XNORI: std_logic_vector(4 downto 0):="01100";
31     constant BEQI: std_logic_vector(4 downto 0):="01101";
32     constant BNEI: std_logic_vector(4 downto 0):="01110";
33     constant BLTI: std_logic_vector(4 downto 0):="01111";
34     constant BLETI: std_logic_vector(4 downto 0):="10000";
35     constant BGTI: std_logic_vector(4 downto 0):="10001";
36     constant BGETI: std_logic_vector(4 downto 0):="10010";
37     constant B: std_logic_vector(4 downto 0):="10011";
38     constant CALL: std_logic_vector(4 downto 0):="10100";
39     constant RET: std_logic_vector(4 downto 0):="10101";
40     constant NOP: std_logic_vector(4 downto 0):="10110";
41
42     --REG
43     CONSTANT R0: STD_LOGIC_VECTOR(3 DOWNT0 0):="0000";
44     CONSTANT R1: STD_LOGIC_VECTOR(3 DOWNT0 0):="0001";
45     CONSTANT R2: STD_LOGIC_VECTOR(3 DOWNT0 0):="0010";
46     CONSTANT R3: STD_LOGIC_VECTOR(3 DOWNT0 0):="0011";
47     CONSTANT R4: STD_LOGIC_VECTOR(3 DOWNT0 0):="0100";
48     CONSTANT R5: STD_LOGIC_VECTOR(3 DOWNT0 0):="0101";
```

```

49  CONSTANT R6: STD_LOGIC_VECTOR(3 DOWNTO 0):="0110";
50  CONSTANT R7: STD_LOGIC_VECTOR(3 DOWNTO 0):="0111";
51  CONSTANT R8: STD_LOGIC_VECTOR(3 DOWNTO 0):="1000";
52  CONSTANT R9: STD_LOGIC_VECTOR(3 DOWNTO 0):="1001";
53  CONSTANT R10: STD_LOGIC_VECTOR(3 DOWNTO 0):="1010";
54  CONSTANT R11: STD_LOGIC_VECTOR(3 DOWNTO 0):="1011";
55  CONSTANT R12: STD_LOGIC_VECTOR(3 DOWNTO 0):="1100";
56  CONSTANT R13: STD_LOGIC_VECTOR(3 DOWNTO 0):="1101";
57  CONSTANT R14: STD_LOGIC_VECTOR(3 DOWNTO 0):="1110";
58  CONSTANT R15: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111";
59
60  --S/U
61  CONSTANT SU: std_logic_vector(3 downto 0):="0000";
62  --C-FIN-OPERACION
63  CONSTANT ADD: STD_LOGIC_VECTOR(3 DOWNTO 0):="0000";
64  CONSTANT SUB: STD_LOGIC_VECTOR(3 DOWNTO 0):="0001";
65  CONSTANT C_AND: STD_LOGIC_VECTOR(3 DOWNTO 0):="0010";
66  CONSTANT C_OR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0011";
67  CONSTANT C_XOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0100";
68  CONSTANT C_NAND: STD_LOGIC_VECTOR(3 DOWNTO 0):="0101";
69  CONSTANT C_NOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0110";
70  CONSTANT C_XNOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0111";
71  CONSTANT C_NOT: STD_LOGIC_VECTOR(3 DOWNTO 0):="1000";
72  CONSTANT C_SLL: STD_LOGIC_VECTOR(3 DOWNTO 0):="1001";
73  CONSTANT C_SRL: STD_LOGIC_VECTOR(3 DOWNTO 0):="1010";
74
75  type banco is array (0 to (2**m)-1) of std_logic_vector(n-1 downto 0);
76
77  constant aux:banco:=(
78      LI & R6 & x"0057",          -- 1 LI R6, #87
79      LI & R8 & x"005A",          -- 2 LI R8, #90
80      TYPED & R8 & R2 & R3 & SU & ADD, -- 3 ADD R8,R2,R3
81      TYPED & R1 & R2 & R3 & SU & SUB, -- 4 SUB R1,R2,R3
82      CALL & SU & x"0009",        -- 5 CALL 0x09
83      LI & R6 & x"0057",          -- 6 LI R6, #87
84      LI & R8 & x"005A",          -- 7 LI R8, #90
85      CALL & SU & x"000D",        -- 8 CALL 13
86      TYPED & R8 & R2 & R3 & SU & ADD, -- 9 ADD R8,R2,R3
87      TYPED & R1 & R2 & R3 & SU & SUB, --10 SUB R1,R2,R3
88      LI & R6 & x"0057",          --11 LI R6, #87
89      RET & SU & SU & SU & SU & SU, --12 RET
90      TYPED & R1 & R2 & R3 & SU & SUB, --13 SUB R1,R2,R3
91      LI & R6 & x"0057",          --14 LI R6, #87
92      RET & SU & SU & SU & SU & SU, --15 RET
93      B&SU & x"0012",             --16 B 18
94      NOP & SU & SU & SU & SU & SU, --17 NOP
95      NOP & SU & SU & SU & SU & SU, --18 NOP
96      B & SU & x"0011",           --19 B 17
97      others=>(others=>'0')
98  );
99  begin
100  inst <= aux(conv_integer(dir));
101  end Behavioral;

```

1.2. Stack

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5
6  entity pila is
7      generic(
8          N:integer:=16;
9          B:integer:=3
10     );

```

```

11     Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
12         DW, UP, WPC, CLK,CLR:in STD_LOGIC;
13         PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
14         SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
15 end pila;
16
17 architecture Behavioral of pila is
18     type banco is array (0 to 7) of std_logic_vector(N-1 downto 0);
19     signal pila: banco;
20 begin
21     process(CLK,CLR,pila)
22         variable sp: integer range 0 to 7;
23     begin
24         if(CLR = '1') then
25             sp := 0;
26             pila <= (others =>(others => '0'));
27         elsif(CLK'event and clk = '1') then
28             if(WPC = '0' and UP = '0' and DW = '0') then
29                 pila(sp) <= pila(sp)+1;
30             elsif(WPC = '1' AND UP = '0' AND DW = '0') then
31                 pila(sp) <= PCin;
32             elsif(WPC = '1' and UP = '1' and DW = '0') then
33                 sp := sp + 1;
34                 pila(sp) <= PCin;
35             elsif(WPC = '0' and UP = '0' and DW = '1') then
36                 sp:=sp-1;
37                 pila(sp)<=pila(sp)-1;
38             end if;
39         end if;
40         SP_out <= conv_std_logic_vector(sp,3);
41         PCout <= pila(sp);
42     end process;
43 end Behavioral;

```

1.3. Unión para la arquitectura

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_UNSIGNED.ALL;
4 use IEEE.STD_LOGIC_ARITH.ALL;
5
6 entity pilapc is
7     generic(
8         N:integer:=16;
9         B:integer:=3;
10        m:integer:= 10;--tamaño del bus de direcciones
11        s:integer:= 25--tamaño de palabra
12    );
13    Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
14        DW, UP, WPC, CLK,CLR:in STD_LOGIC;
15        PCout_h:out STD_LOGIC_VECTOR(s-1 downto 0);
16        PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
17        SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
18 end pilapc;
19
20 architecture Behavioral of pilapc is
21     component memPrograma is
22         Port(dir:in STD_LOGIC_VECTOR (m-1 downto 0);
23             inst:out STD_LOGIC_VECTOR (s-1 downto 0));
24     end component;
25     component pila is
26         Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
27             DW, UP, WPC, CLK,CLR:in STD_LOGIC;
28             PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
29             SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
30     end component;

```

```

31     signal SPCout: STD_LOGIC_VECTOR(N-1 downto 0);
32 begin
33     stack:pila port map(
34         PCin=>PCin,
35         DW=>DW,
36         UP=>UP,
37         WPC=>WPC,
38         CLK=>CLK,
39         CLR=>CLR,
40         SP_out=>SP_out,
41         PCout=>SPCout
42     );
43     memoria:memPrograma port map(
44         dir=>SPCout(m-1 downto 0),
45         inst=>PCout_h
46     );
47     process (SPCout)
48     begin
49         PCout<=SPCout;
50     end process;
51 end Behavioral;

```

2. Test-Bench:

```

1  LIBRARY ieee;
2  LIBRARY STD;
3  USE STD.TEXTIO.ALL;
4  USE ieee.std_logic_TEXTIO.ALL;  --PERMITE USAR STD_LOGIC
5  USE ieee.std_logic_1164.ALL;
6  USE ieee.std_logic_UNSIGNED.ALL;
7  USE ieee.std_logic_ARITH.ALL;
8
9  entity pila_TB is
10     generic(
11         N:integer:=16;
12         B:integer:=3;
13         s:integer:= 25
14     );
15 end pila_TB;
16
17 architecture Behavioral of pila_TB is
18     component pilapc
19     Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
20         DW, UP, WPC, CLK,CLR:in STD_LOGIC;
21         PCout_h:out STD_LOGIC_VECTOR(s-1 downto 0);
22         PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
23         SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
24     end component;
25     signal PCin:STD_LOGIC_VECTOR(N-1 downto 0);
26     signal DW, UP, WPC, CLK,CLR:STD_LOGIC;
27     signal PCout_h:STD_LOGIC_VECTOR(s-1 downto 0);
28     signal PCout:STD_LOGIC_VECTOR(N-1 downto 0);
29     signal SP_out:STD_LOGIC_VECTOR(B-1 downto 0);
30     constant CLK_P:time:=10ns;
31 begin
32     pilaH:pilapc port map(
33         PCin=>PCin,
34         DW=>DW,
35         UP=>UP,
36         WPC=>WPC,
37         CLK=>CLK,
38         CLR=>CLR,
39         PCout_h=>PCout_h,
40         PCout=>PCout,
41         SP_out=>SP_out

```

```

42 );
43
44 CLK_Process:process
45 begin
46     CLK<='0';
47     wait for CLK_P/2;
48     CLK<='1';
49     wait for CLK_P/2;
50 end process;
51
52 tbPila:process
53     file ARCH_ENT:TEXT;
54     variable LINEA_ENT:line;
55     variable VAR_PCin:STD_LOGIC_VECTOR (15 downto 0);
56     variable VAR_DW:STD_LOGIC;
57     variable VAR_UP:STD_LOGIC;
58     variable VAR_WPC:STD_LOGIC;
59     variable VAR_CLR:STD_LOGIC;
60     VARIABLE CADENA:STRING(1 TO 8);
61
62     file ARCH_SAL:TEXT;
63     variable LINEA_SAL:line;
64     variable VAR_PCout:STD_LOGIC_VECTOR(15 downto 0);
65     variable VAR_Ins:STD_LOGIC_VECTOR(24 downto 0);
66     variable VAR_SPout:STD_LOGIC_VECTOR(2 downto 0);
67     variable VAR_OP:STD_LOGIC_VECTOR(4 downto 0);
68     variable VAR_Rd:STD_LOGIC_VECTOR(3 downto 0);
69     variable VAR_Rt:STD_LOGIC_VECTOR(3 downto 0);
70     variable VAR_Rs:STD_LOGIC_VECTOR(3 downto 0);
71     variable VAR_Shamt:STD_LOGIC_VECTOR(3 downto 0);
72     variable VAR_FuncCode:STD_LOGIC_VECTOR(3 downto 0);
73 begin
74     file_open(ARCH_ENT, "/run/media/d3vcr4ck/externData/materias-Sem20_2/
arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/pilaHardware_2/in
.txt", READ_MODE);
75     file_open(ARCH_SAL, "/run/media/d3vcr4ck/externData/materias-Sem20_2/
arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/pilaHardware_2/
out.txt", WRITE_MODE);
76     CADENA:="  SP  ";
77     write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
78     CADENA:="  PC  ";
79     write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
80     CADENA:="  OPC ";
81     write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
82     CADENA:="  Rd  ";
83     write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
84     CADENA:="  Rt  ";
85     write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
86     CADENA:="  Rs  ";
87     write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
88     CADENA:="  Shamt ";
89     write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
90     CADENA:="  FC  ";
91     write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
92     writeline(ARCH_SAL,LINEA_SAL);
93     for i in 0 to 31 loop
94         readline(ARCH_ENT,LINEA_ENT);
95         read(LINEA_ENT,VAR_UP);
96         UP <= VAR_UP;
97         read(LINEA_ENT,VAR_DW);
98         DW <= VAR_DW;
99         read(LINEA_ENT,VAR_WPC);
100        WPC <= VAR_WPC;
101        Hread(LINEA_ENT,VAR_PCin);
102        PCin <= VAR_PCin;
103        read(LINEA_ENT,VAR_CLR);
104        CLR <= VAR_CLR;

```

```

105         wait until rising_edge(CLK);
106
107         VAR_PCnt:=PCout;
108         VAR_SPout:=SP_out;
109         VAR_Ins:=PCout_h;
110         VAR_OP:=VAR_Ins(24 downto 20);
111         VAR_Rd:=VAR_Ins(19 downto 16);
112         VAR_Rt:=VAR_Ins(15 downto 12);
113         VAR_Rs:=VAR_Ins(11 downto 8);
114         VAR_Shamt:=VAR_Ins(7 downto 4);
115         VAR_FuncCode:=VAR_Ins(3 downto 0);
116         HWRITE(LINEA_SAL,VAR_SPout, right, 6);
117         HWRITE(LINEA_SAL,VAR_PCnt, right, 10);
118         WRITE(LINEA_SAL,VAR_OP, right, 10);
119         WRITE(LINEA_SAL,VAR_Rd, right, 8);
120         WRITE(LINEA_SAL,VAR_Rt, right, 10);
121         WRITE(LINEA_SAL,VAR_Rs, right, 8);
122         WRITE(LINEA_SAL,VAR_Shamt, right, 10);
123         WRITE(LINEA_SAL,VAR_FuncCode,right, 8);
124         writeline(ARCH_SAL,LINEA_SAL);
125     end loop;
126     wait for 100 ns;
127     file_close(ARCH_ENT);
128     file_close(ARCH_SAL);
129     wait;
130 end process;
131 end Behavioral;

```

3. Archivos de entrada y salida

3.1. Entrada

```

1 0 0 0 0000 1
2 0 0 0 0000 0
3 0 0 1 0001 0
4 0 0 1 0002 0
5 0 0 1 0003 0
6 0 0 1 0004 0
7 0 0 1 0005 0
8 1 0 1 0008 0
9 0 0 1 0009 0
10 0 0 1 000A 0
11 0 0 1 000B 0
12 0 1 0 0000 0
13 0 0 1 0005 0
14 0 0 1 0006 0
15 0 0 1 0007 0
16 0 0 1 0008 0
17 1 0 1 000C 0
18 0 0 1 000D 0
19 0 1 0 0000 0
20 0 0 1 0008 0
21 0 0 1 0009 0
22 0 0 1 000A 0
23 0 0 1 000B 0
24 0 0 1 000C 0
25 0 0 1 000D 0
26 0 0 1 000E 0
27 0 0 1 000F 0
28 0 0 1 0011 0
29 0 0 1 0012 0
30 0 0 1 0010 0
31 0 0 1 0011 0
32 0 0 1 0012 0
33 UP DW WPC PCin CLR

```

3.2. Salida

1	SP	PC	OPC	Rd	Rt	Rs	Shamt	FC
2	0	0000	00001	0110	0000	0000	0101	0111
3	0	0000	00001	0110	0000	0000	0101	0111
4	0	0001	00001	1000	0000	0000	0101	1010
5	0	0001	00001	1000	0000	0000	0101	1010
6	0	0002	00000	1000	0010	0011	0000	0000
7	0	0003	00000	0001	0010	0011	0000	0001
8	0	0004	10100	0000	0000	0000	0000	1001
9	0	0005	00001	0110	0000	0000	0101	0111
10	1	0008	00000	1000	0010	0011	0000	0000
11	1	0009	00000	0001	0010	0011	0000	0001
12	1	000A	00001	0110	0000	0000	0101	0111
13	1	000B	10101	0000	0000	0000	0000	0000
14	0	0004	10100	0000	0000	0000	0000	1001
15	0	0005	00001	0110	0000	0000	0101	0111
16	0	0006	00001	1000	0000	0000	0101	1010
17	0	0007	10100	0000	0000	0000	0000	1101
18	0	0008	00000	1000	0010	0011	0000	0000
19	1	000C	00000	0001	0010	0011	0000	0001
20	1	000D	00001	0110	0000	0000	0101	0111
21	0	0007	10100	0000	0000	0000	0000	1101
22	0	0008	00000	1000	0010	0011	0000	0000
23	0	0009	00000	0001	0010	0011	0000	0001
24	0	000A	00001	0110	0000	0000	0101	0111
25	0	000B	10101	0000	0000	0000	0000	0000
26	0	000C	00000	0001	0010	0011	0000	0001
27	0	000D	00001	0110	0000	0000	0101	0111
28	0	000E	10101	0000	0000	0000	0000	0000
29	0	000F	10011	0000	0000	0000	0001	0010
30	0	0011	10110	0000	0000	0000	0000	0000
31	0	0012	10011	0000	0000	0000	0001	0001
32	0	0010	10110	0000	0000	0000	0000	0000
33	0	0011	10110	0000	0000	0000	0000	0000

4. Diagrama RTL

