Reporte de practica 11

González Pardo Adrian

Mayo 2020

1. Código fuente:

1.1. Memoria de programa

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;

entity memPrograma is
    generic(
        m:integer:= 10;--tamanio del bus de direcciones
        n:integer:= 25--tamanio de palabra
    );
    Port(dir:in STD_LOGIC_VECTOR (m-1 downto 0);
         inst:out STD_LOGIC_VECTOR (n-1 downto 0));
end memPrograma;

architecture Behavioral of memPrograma is
        --C-OPERACION
    constant TYPER: std_logic_vector(4 downto 0):="00000";
    constant LI: std_logic_vector(4 downto 0):="00001";
    constant LWI: std_logic_vector(4 downto 0):="00010";
    constant LW: std_logic_vector(4 downto 0):="10111";
    constant SWI: std_logic_vector(4 downto 0):="00011";
    constant SW: std_logic_vector(4 downto 0):="00100";
    constant ADDI: std_logic_vector(4 downto 0):="00101";
    constant SUBI: std_logic_vector(4 downto 0):="00110";
    constant ANDI: std_logic_vector(4 downto 0):="00111";
    constant ORI: std_logic_vector(4 downto 0):="01000";
    constant XORI: std_logic_vector(4 downto 0):="01001";
    constant NANDI: std_logic_vector(4 downto 0):="01010";
    constant NORI: std_logic_vector(4 downto 0):="01011";
    constant XNORI: std_logic_vector(4 downto 0):="01100";
    constant BEQI: std_logic_vector(4 downto 0):="01101";
    constant BNEI: std_logic_vector(4 downto 0):="01110";
    constant BLTI: std_logic_vector(4 downto 0):="01111";
    constant BLETI: std_logic_vector(4 downto 0):="10000";
    constant BGTI: std_logic_vector(4 downto 0):="10001";
    constant BGETI: std_logic_vector(4 downto 0):="10010";
    constant B: std_logic_vector(4 downto 0):="10011";
    constant CALL: std_logic_vector(4 downto 0):="10100";
    constant RET: std_logic_vector(4 downto 0):="10101";
    constant NOP: std_logic_vector(4 downto 0):="10110";

        --REG
    CONSTANT R0: STD_LOGIC_VECTOR(3 DOWNTO 0):="0000";
    CONSTANT R1: STD_LOGIC_VECTOR(3 DOWNTO 0):="0001";
    CONSTANT R2: STD_LOGIC_VECTOR(3 DOWNTO 0):="0010";
    CONSTANT R3: STD_LOGIC_VECTOR(3 DOWNTO 0):="0011";
    CONSTANT R4: STD_LOGIC_VECTOR(3 DOWNTO 0):="0100";
    CONSTANT R5: STD_LOGIC_VECTOR(3 DOWNTO 0):="0101";
    CONSTANT R6: STD_LOGIC_VECTOR(3 DOWNTO 0):="0110";
    CONSTANT R7: STD_LOGIC_VECTOR(3 DOWNTO 0):="0111";
    CONSTANT R8: STD_LOGIC_VECTOR(3 DOWNTO 0):="1000";
    CONSTANT R9: STD_LOGIC_VECTOR(3 DOWNTO 0):="1001";
    CONSTANT R10: STD_LOGIC_VECTOR(3 DOWNTO 0):="1010";
    CONSTANT R11: STD_LOGIC_VECTOR(3 DOWNTO 0):="1011";
    CONSTANT R12: STD_LOGIC_VECTOR(3 DOWNTO 0):="1100";
    CONSTANT R13: STD_LOGIC_VECTOR(3 DOWNTO 0):="1101";
    CONSTANT R14: STD_LOGIC_VECTOR(3 DOWNTO 0):="1110";
    CONSTANT R15: STD_LOGIC_VECTOR(3 DOWNTO 0):="1111";

        --S/U
    CONSTANT SU: std_logic_vector(3 downto 0):="0000";
        --C-FIN-OPERACION
    CONSTANT ADD: STD_LOGIC_VECTOR(3 DOWNTO 0):="0000";
    CONSTANT SUB: STD_LOGIC_VECTOR(3 DOWNTO 0):="0001";
    CONSTANT C_AND: STD_LOGIC_VECTOR(3 DOWNTO 0):="0010";
    CONSTANT C_OR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0011";
    CONSTANT C_XOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0100";
    CONSTANT C_NAND: STD_LOGIC_VECTOR(3 DOWNTO 0):="0101";
    CONSTANT C_NOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0110";
    CONSTANT C_XNOR: STD_LOGIC_VECTOR(3 DOWNTO 0):="0111";
    CONSTANT C_NOT: STD_LOGIC_VECTOR(3 DOWNTO 0):="1000";
    CONSTANT C_SLL: STD_LOGIC_VECTOR(3 DOWNTO 0):="1001";
    CONSTANT C_SRL: STD_LOGIC_VECTOR(3 DOWNTO 0):="1010";

    type banco is array (0 to (2**m)-1) of std_logic_vector(n-1 downto 0);

    constant aux:banco:=(
        LI & R6 & x"0057",                      -- 1 LI R6, #87
        LI & R8 & x"005A",                      -- 2 LI R8, #90
        TYPER & R8 & R2 & R3 & SU & ADD ,       -- 3 ADD R8,R2,R3
        TYPER & R1 & R2 & R3 & SU & SUB ,       -- 4 SUB R1,R2,R3
        CALL & SU & x"0009",                    -- 5 CALL 0x09
        LI & R6 & x"0057",                      -- 6 LI R6,#87
        LI & R8 & x"005A",                      -- 7 LI R8, #90
        CALL & SU & x"000D",                    -- 8 CALL 13
        TYPER & R8 & R2 & R3 & SU & ADD ,       -- 9 ADD R8,R2,R3
        TYPER & R1 & R2 & R3 & SU & SUB ,       --10 SUB R1,R2,R3
        LI & R6 & x"0057",                      --11 LI R6,#87
        RET & SU & SU & SU & SU & SU,           --12 RET
        TYPER & R1 & R2 & R3 & SU & SUB ,       --13 SUB R1,R2,R3
        LI & R6 & x"0057",                      --14 LI R6,#87
        RET & SU & SU & SU & SU & SU,           --15 RET
        B&SU & x"0012",                         --16 B 18
        NOP & SU & SU & SU & SU & SU,           --17 NOP
        NOP & SU & SU & SU & SU & SU,           --18 NOP
        B & SU & x"0011",                       --19 B 17
        others=>(others=>'0')
    );
begin
    inst <= aux(conv_integer(dir));
end Behavioral;
```

1.2. Stack

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity pila is
    generic(
        N:integer:=16;
        B:integer:=3
    );
    Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
         DW, UP, WPC, CLK,CLR:in STD_LOGIC;
         PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
         SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
end pila;

architecture Behavioral of pila is
    type banco is array (0 to 7) of std_logic_vector(N-1 downto 0);
    signal pila: banco;
begin
    process(CLK,CLR,pila)
        variable sp: integer range 0 to 7;
    begin
        if(CLR = '1') then
            sp := 0;
            pila <= (others =>(others => '0'));
        elsif(CLK'event and clk = '1') then
            if(WPC = '0' and UP = '0' and DW = '0') then
                pila(sp) <= pila(sp)+1;
            elsif(WPC = '1' AND UP = '0' AND DW = '0') then
                pila(sp) <= PCin;
            elsif(WPC = '1' and UP = '1' and DW = '0')then
                sp := sp + 1;
                pila(sp) <= PCin;
            elsif(WPC = '0' and UP = '0' and DW = '1') then
                sp:=sp-1;
                pila(sp)<=pila(sp)-1;
            end if;
        end if;
        SP_out <= conv_std_logic_vector(sp,3);
        PCout <= pila(sp);
    end process;
end Behavioral;
```

1.3. Unión para la arquitectura

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity pilapc is
    generic(
        N:integer:=16;
        B:integer:=3;
        m:integer:= 10;--tamanio del bus de direcciones
        s:integer:= 25--tamanio de palabra
    );
    Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
         DW, UP, WPC, CLK,CLR:in STD_LOGIC;
         PCout_h:out STD_LOGIC_VECTOR(s-1 downto 0);
         PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
         SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
end pilapc;

architecture Behavioral of pilapc is
    component memPrograma is
        Port(dir:in STD_LOGIC_VECTOR (m-1 downto 0);
             inst:out STD_LOGIC_VECTOR (s-1 downto 0));
    end component;
    component pila is
        Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
             DW, UP, WPC, CLK,CLR:in STD_LOGIC;
             PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
             SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
    end component;
```

```vhdl
    signal SPCout: STD_LOGIC_VECTOR(N-1 downto 0);
begin
    stack:pila port map(
        PCin=>PCin,
        DW=>DW,
        UP=>UP,
        WPC=>WPC,
        CLK=>CLK,
        CLR=>CLR,
        SP_out=>SP_out,
        PCout=>SPCout
    );
    memoria:memPrograma port map(
        dir=>SPCout(m-1 downto 0),
        inst=>PCout_h
    );
    process (SPCout)
    begin
        PCout<=SPCout;
    end process;
end Behavioral;
```

2. Test-Bench:

```vhdl
LIBRARY ieee;
LIBRARY STD;
USE STD.TEXTIO.ALL;
USE ieee.std_logic_TEXTIO.ALL;  --PERMITE USAR STD_LOGIC
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_UNSIGNED.ALL;
USE ieee.std_logic_ARITH.ALL;

entity pila_TB is
    generic(
        N:integer:=16;
        B:integer:=3;
        s:integer:= 25
    );
end pila_TB;

architecture Behavioral of pila_TB is
    component pilapc
    Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
         DW, UP, WPC, CLK,CLR:in STD_LOGIC;
         PCout_h:out STD_LOGIC_VECTOR(s-1 downto 0);
         PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
         SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
    end component;
    signal PCin:STD_LOGIC_VECTOR(N-1 downto 0);
    signal DW, UP, WPC, CLK,CLR:STD_LOGIC;
    signal PCout_h:STD_LOGIC_VECTOR(s-1 downto 0);
    signal PCout:STD_LOGIC_VECTOR(N-1 downto 0);
    signal SP_out:STD_LOGIC_VECTOR(B-1 downto 0);
    constant CLK_P:time:=10ns;
begin
    pilaH:pilapc port map(
        PCin=>PCin,
        DW=>DW,
        UP=>UP,
        WPC=>WPC,
        CLK=>CLK,
        CLR=>CLR,
        PCout_h=>PCout_h,
        PCout=>PCout,
        SP_out=>SP_out
    );

    CLK_Process:process
    begin
        CLK<='0';
        wait for CLK_P/2;
        CLK<='1';
        wait for CLK_P/2;
    end process;

    tbPila:process
        file ARCH_ENT:TEXT;
        variable LINEA_ENT:line;
        variable VAR_PCin:STD_LOGIC_VECTOR (15 downto 0);
        variable VAR_DW:STD_LOGIC;
        variable VAR_UP:STD_LOGIC;
        variable VAR_WPC:STD_LOGIC;
        variable VAR_CLR:STD_LOGIC;
        VARIABLE CADENA:STRING(1 TO 8);

        file ARCH_SAL:TEXT;
        variable LINEA_SAL:line;
        variable VAR_PCout:STD_LOGIC_VECTOR(15 downto 0);
        variable VAR_Ins:STD_LOGIC_VECTOR(24 downto 0);
        variable VAR_SPout:STD_LOGIC_VECTOR(2 downto 0);
        variable VAR_OP:STD_LOGIC_VECTOR(4 downto 0);
        variable VAR_Rd:STD_LOGIC_VECTOR(3 downto 0);
        variable VAR_Rt:STD_LOGIC_VECTOR(3 downto 0);
        variable VAR_Rs:STD_LOGIC_VECTOR(3 downto 0);
        variable VAR_Shamt:STD_LOGIC_VECTOR(3 downto 0);
        variable VAR_FuncCode:STD_LOGIC_VECTOR(3 downto 0);
    begin
        file_open(ARCH_ENT, "/run/media/d3vcr4ck/externData/materias-Sem20_2/arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/pilaHardware_2/in.txt", READ_MODE);
        file_open(ARCH_SAL, "/run/media/d3vcr4ck/externData/materias-Sem20_2/arquitecturaDeComputadoras/arquitecturaDeComputadoras/practicasVivado/pilaHardware_2/out.txt", WRITE_MODE);
        CADENA:="   SP   ";
        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
        CADENA:="   PC   ";
        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
        CADENA:="   OPC  ";
        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
        CADENA:="   Rd   ";
        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
        CADENA:="   Rt   ";
        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
        CADENA:="   Rs   ";
        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
        CADENA:="  Shamt ";
        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
        CADENA:="   FC   ";
        write(LINEA_SAL, CADENA, right, CADENA'LENGTH+1);
        writeline(ARCH_SAL,LINEA_SAL);
        for i in 0 to 31 loop
            readline(ARCH_ENT,LINEA_ENT);
            read(LINEA_ENT,VAR_UP);
            UP <= VAR_UP;
            read(LINEA_ENT,VAR_DW);
            DW <= VAR_DW;
            read(LINEA_ENT,VAR_WPC);
            WPC <= VAR_WPC;
            Hread(LINEA_ENT,VAR_PCin);
            PCin <= VAR_PCin;
            read(LINEA_ENT,VAR_CLR);
            CLR <= VAR_CLR;
            wait until rising_edge(CLK);

            VAR_PCout:=PCout;
            VAR_SPout:=SP_out;
            VAR_Ins:=PCout_h;
            VAR_OP:=VAR_Ins(24 downto 20);
            VAR_Rd:=VAR_Ins(19 downto 16);
            VAR_Rt:=VAR_Ins(15 downto 12);
            VAR_Rs:=VAR_Ins(11 downto 8);
            VAR_Shamt:=VAR_Ins(7 downto 4);
            VAR_FuncCode:=VAR_Ins(3 downto 0);
            HWRITE(LINEA_SAL,VAR_SPout, right, 6);
            HWRITE(LINEA_SAL,VAR_PCout, right, 10);
            WRITE(LINEA_SAL,VAR_OP, right, 10);
            WRITE(LINEA_SAL,VAR_Rd, right, 8);
            WRITE(LINEA_SAL,VAR_Rt, right, 10);
            WRITE(LINEA_SAL,VAR_Rs, right, 8);
            WRITE(LINEA_SAL,VAR_Shamt, right, 10);
            WRITE(LINEA_SAL,VAR_FuncCode,right, 8);
            writeline(ARCH_SAL,LINEA_SAL);
        end loop;
        wait for 100 ns;
        file_close(ARCH_ENT);
        file_close(ARCH_SAL);
        wait;
    end process;
end Behavioral;
```

3. Archivos de entrada y salida

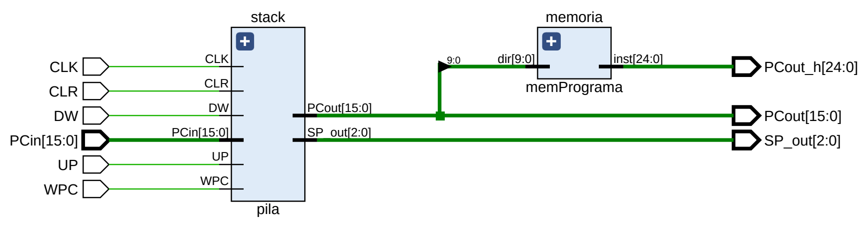
3.1. Entrada

```
0 0 0 0000 1
0 0 0 0000 0
0 0 1 0001 0
0 0 1 0002 0
0 0 1 0003 0
0 0 1 0004 0
0 0 1 0005 0
1 0 1 0008 0
0 0 1 0009 0
0 0 1 000A 0
0 0 1 000B 0
0 1 0 0000 0
0 0 1 0005 0
0 0 1 0006 0
0 0 1 0007 0
0 0 1 0008 0
1 0 1 000C 0
0 0 1 000D 0
0 1 0 0000 0
0 0 1 0008 0
0 0 1 0009 0
0 0 1 000A 0
0 0 1 000B 0
0 0 1 000C 0
0 0 1 000D 0
0 0 1 000E 0
0 0 1 000F 0
0 0 1 0011 0
0 0 1 0012 0
0 0 1 0010 0
0 0 1 0011 0
0 0 1 0012 0
UP DW WPC PCin CLR
```

3.2. Salida

```
    SP        PC        OPC       Rd         Rt        Rs        Shamt     FC
     0       0000      00001     0110       0000      0000        0101     0111
     0       0000      00001     0110       0000      0000        0101     0111
     0       0001      00001     1000       0000      0000        0101     1010
     0       0001      00001     1000       0000      0000        0101     1010
     0       0002      00000     1000       0010      0011        0000     0000
     0       0003      00000     0001       0010      0011        0000     0001
     0       0004      10100     0000       0000      0000        0000     1001
     0       0005      00001     0110       0000      0000        0101     0111
     1       0008      00000     1000       0010      0011        0000     0000
     1       0009      00000     0001       0010      0011        0000     0001
     1       000A      00001     0110       0000      0000        0101     0111
     1       000B      10101     0000       0000      0000        0000     0000
     0       0004      10100     0000       0000      0000        0000     1001
     0       0005      00001     0110       0000      0000        0101     0111
     0       0006      00001     1000       0000      0000        0101     1010
     0       0007      10100     0000       0000      0000        0000     1101
     0       0008      00000     1000       0010      0011        0000     0000
     1       000C      00000     0001       0010      0011        0000     0001
     1       000D      00001     0110       0000      0000        0101     0111
     0       0007      10100     0000       0000      0000        0000     1101
     0       0008      00000     1000       0010      0011        0000     0000
     0       0009      00000     0001       0010      0011        0000     0001
     0       000A      00001     0110       0000      0000        0101     0111
     0       000B      10101     0000       0000      0000        0000     0000
     0       000C      00000     0001       0010      0011        0000     0001
     0       000D      00001     0110       0000      0000        0101     0111
     0       000E      10101     0000       0000      0000        0000     0000
     0       000F      10011     0000       0000      0000        0001     0010
     0       0011      10110     0000       0000      0000        0000     0000
     0       0012      10011     0000       0000      0000        0001     0001
     0       0010      10110     0000       0000      0000        0000     0000
     0       0011      10110     0000       0000      0000        0000     0000
```

4. Diagrama RTL


stack
CLK
CLR
DW
PCin[15:0]
UP
WPC
CLK
CLR
DW
PCin[15:0]
UP
WPC
pila
PCout[15:0]
SP_out[2:0]
9:0
memoria
dir[9:0]
inst[24:0]
memPrograma
PCout_h[24:0]
PCout[15:0]
SP_out[2:0]