Instituto Politécnico Nacional

Escuela Superior de Cómputo

Arquitectura de Computadoras

Practica 10: Pila Hardware 2

Nombre: *Sampayo Hernández Mauro*

Grupo: *3CV8*

Profesor: *Nayeli Vega García*

Fecha de entrega: *17 de abril del 2020*

Código de Implementación:

- **Pila**

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;--Datos con signos y sin signo, y
operaciones aritmeticas
use IEEE.STD_LOGIC_unsigned.ALL;--Realizar operaciones sin signo para los
ST_LOGIC_VECTOR

entity Pila is
    generic ( N: integer :=3;
              M: integer :=16);
    Port ( PC_in : in  STD_LOGIC_VECTOR (M-1 downto 0);
           PC_out : out  STD_LOGIC_VECTOR (M-1 downto 0);
           clk, clr, UP, DW, WPC : in  STD_LOGIC;
           SP: out std_logic_vector(N-1 downto 0));
end Pila;

architecture Behavioral of Pila is

type banco is array (0 to (2**N)-1) of std_logic_vector(M-1 downto 0);
signal aux: banco;
signal SP1: integer range 0 to (2**N)-1;

begin
    process(clk, clr)
        variable SPout : integer range 0 to (2**N)-1;
    begin
        if(clr = '1')then
            SPout := 0;
            aux <= (others => (others => '0'));
        elsif(clk'event and clk = '1')then
            if(WPC = '0' and UP = '0' and DW = '0')then--incremento
                aux(SPout) <= aux(SPout)+1;
            elsif(WPC = '1' and UP = '1' and DW = '0')then--CALL
                SPout := SPout + 1;
                aux(SPout) <= PC_in;
            elsif(WPC = '1' and UP = '0' and DW = '0')then--JUMP
                aux(SPout) <= PC_in;
            elsif(WPC = '0' and UP = '0' and DW = '1')then--RET
                SPout := SPout - 1;
                aux(SPout) <= aux(SPout)+1;
            end if;
        end if;
        SP1 <= SPout;
    end process;

    SP <= conv_std_logic_vector(SP1, 3);
    PC_out <= aux(SP1);

end Behavioral;
```

- **StackPointer**

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

entity StackPointer is
    Port ( clk, clr, UP, DW : in  STD_LOGIC;
           SP: inout STD_LOGIC_VECTOR(2 downto 0));
end StackPointer;

architecture Behavioral of StackPointer is

begin
    process(clk, clr)
    begin
        if(clr = '1') then
            SP <= "000";
        elsif(RISING_EDGE(clk))then
            if(UP = '1')then
                SP <= SP + 1;
            elsif(DW = '1')then
                SP <= SP - 1;
            end if;
        end if;
    end process;

end Behavioral;
```

- **Demux**

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

entity Demux is
    Port ( WPC : in  STD_LOGIC;
           SP: in STD_LOGIC_VECTOR(2 downto 0);
           dex: out STD_LOGIC_VECTOR(7 downto 0));
end Demux;

architecture Behavioral of Demux is

begin
    dex(0) <= WPC when SP = "000" else '0';
    dex(1) <= WPC when SP = "001" else '0';
    dex(2) <= WPC when SP = "010" else '0';
    dex(3) <= WPC when SP = "011" else '0';
    dex(4) <= WPC when SP = "100" else '0';
    dex(5) <= WPC when SP = "101" else '0';
    dex(6) <= WPC when SP = "110" else '0';
    dex(7) <= WPC when SP = "111" else '0';

end Behavioral;
```

- **Mux**

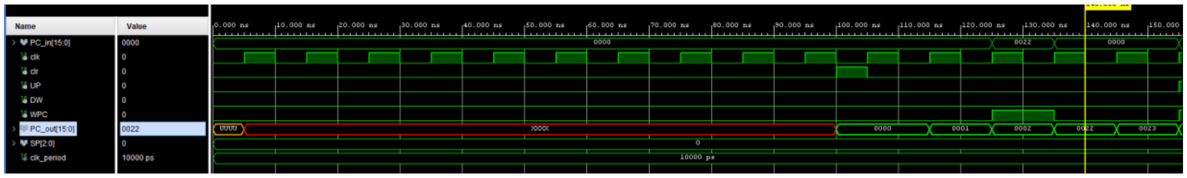
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Mux is
    Port( PCin0: in std_logic_vector(15 downto 0);
          PCin1: in std_logic_vector(15 downto 0);
          PCin2: in std_logic_vector(15 downto 0);
          PCin3: in std_logic_vector(15 downto 0);
          PCin4: in std_logic_vector(15 downto 0);
          PCin5: in std_logic_vector(15 downto 0);
          PCin6: in std_logic_vector(15 downto 0);
          PCin7: in std_logic_vector(15 downto 0);
          SP : in  STD_LOGIC_VECTOR (2 downto 0);
          PC_out: out  STD_LOGIC_VECTOR (15 downto 0));
end Mux;

architecture Behavioral of Mux is

begin
    PC_out <= PCin0 when SP = "000" else
              PCin1 when SP = "001" else
              PCin2 when SP = "010" else
              PCin3 when SP = "011" else
              PCin4 when SP = "100" else
              PCin5 when SP = "101" else
              PCin6 when SP = "110" else
              PCin7;

end Behavioral;
```

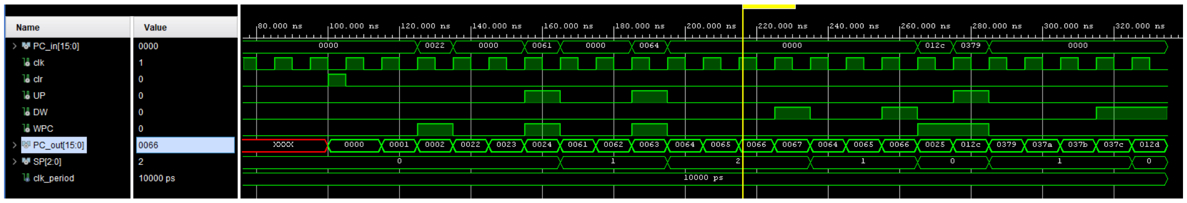
<u>Código de Simulación:</u>

```
library IEEE;
LIBRARY STD;
USE STD.TEXTIO.ALL;
use IEEE.STD_LOGIC_1164.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
end Pila_tb;

architecture Behavioral of Pila_tb is

component Pila
    Port ( PC_in : in  STD_LOGIC_VECTOR (15 downto 0);
           PC_out : out  STD_LOGIC_VECTOR (15 downto 0);
           clk, clr, UP, DW, WPC : in  STD_LOGIC;
           SP: out STD_LOGIC_VECTOR (2 downto 0));
end component;
```

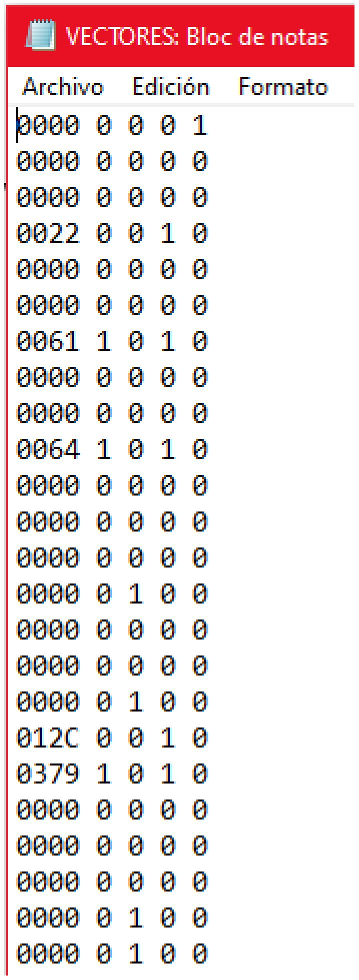
```vhdl
--Inputs
signal PC_in : STD_LOGIC_VECTOR (15 downto 0) := (others => '0');
signal clk : STD_LOGIC := '0';
signal clr : STD_LOGIC := '0';
signal UP : STD_LOGIC := '0';
signal DW : STD_LOGIC := '0';
signal WPC : STD_LOGIC := '0';

--Outputs
signal PC_out : STD_LOGIC_VECTOR (15 downto 0);
signal SP: STD_LOGIC_VECTOR (2 downto 0);

-- Clock period definitions
constant clk_period : time := 10 ns;

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: pila PORT MAP (
        PC_in => PC_in,
        clk => clk,
        clr => clr,
        UP => UP,
        DW => DW,
        WPC => WPC,
        PC_out => PC_out,
        SP => SP
        );

    -- Clock process definitions
    clk_process :process
    begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;

    -- Stimulus process
    stim_proc: process
        file ARCH_RES : TEXT;--archivo de resultados
        variable LINEA_RES : line;--linea de resultado
        file ARCH_VEC : TEXT;--archivo de vectores
        variable LINEA_VEC : line;--linea de vectores

        --Variables
        variable V_PC_in, V_PC_out: STD_LOGIC_VECTOR(15 DOWNTO 0);
        variable v_SP : std_logic_vector(3 downto 0);
        variable V_clr, V_UP, V_DW, V_WPC: STD_LOGIC;
        --Cadena
        variable CADENA : STRING(1 TO 4);
    begin
        file_open(ARCH_VEC, "VECTORES.txt", READ_MODE);
        file_open(ARCH_RES, "RESULTADO.txt", WRITE_MODE);

        --Impresion de Cadenas
        CADENA := "  IN";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := "  UP";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := "  DW";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := " WPC";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := " CLR";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := "  SP";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := "  PC";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        writeline(ARCH_RES,LINEA_RES);--Escribe la linea en el archivo

        --Impresión de Resultados
        wait for 100 ns;
        for i in 0 to 25 loop
           --Lectura de cadenas de VECTORES.txt
            readline(ARCH_VEC, LINEA_VEC);--Lee una linea completa
            hread(LINEA_VEC, V_PC_in);
            PC_in <= V_PC_in;
            read(LINEA_VEC, V_UP);
            UP <= V_UP;
            read(LINEA_VEC, V_DW);
            DW <= V_DW;
            read(LINEA_VEC, V_WPC);
            WPC <= V_WPC;
            read(LINEA_VEC, V_clr);
            clr <= V_clr;

            wait until RISING_EDGE(clk);
            V_PC_out := PC_out; -- asignando salida
            V_SP := '0' & SP;
            --Escribiendo Resultados
            Hwrite(LINEA_RES, V_PC_in, right, 5);
            write(LINEA_RES, V_UP, right, 5);
            write(LINEA_RES, V_DW, right, 5);
            write(LINEA_RES, V_WPC, right, 5);
            write(LINEA_RES, V_clr, right, 5);
            Hwrite(LINEA_RES, V_SP, right, 5);
            Hwrite(LINEA_RES, V_PC_out, right, 5);

            writeline(ARCH_RES,LINEA_RES);--Escribe la linea en el
archivo
        end loop;
        file_close(ARCH_VEC);--Cierra el archivo
        file_close(ARCH_RES);--Cierra el archivo
      wait;
   end process;--Stimulus process

end Behavioral;
```

Simulación:





Archivo de entrada:

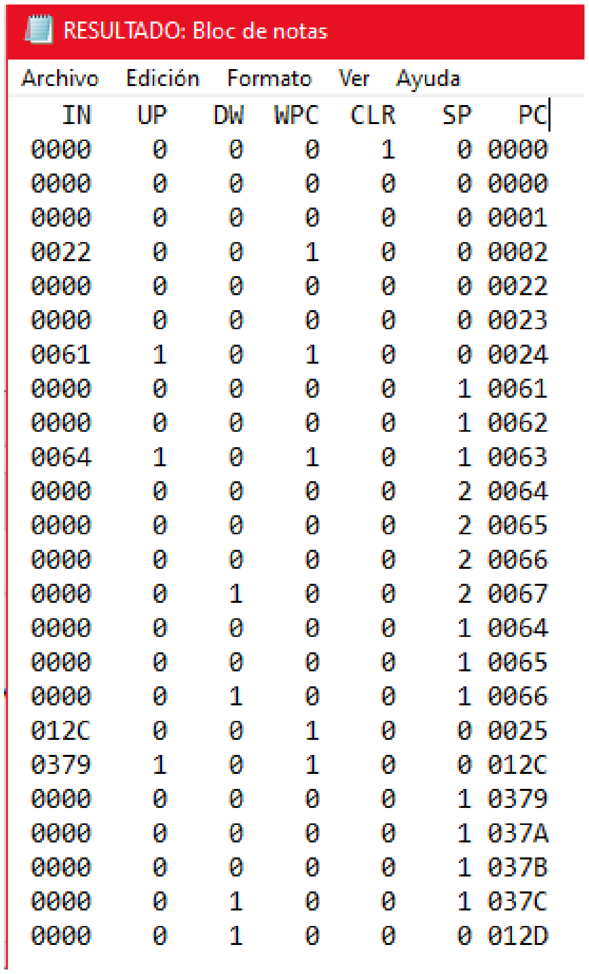


VECTORES: Bloc de notas

Archivo Edición Formato

```
0000 0 0 0 1
0000 0 0 0 0
0000 0 0 0 0
0022 0 0 1 0
0000 0 0 0 0
0000 0 0 0 0
0061 1 0 1 0
0000 0 0 0 0
0000 0 0 0 0
0064 1 0 1 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 1 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 1 0 0
012C 0 0 1 0
0379 1 0 1 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 1 0 0
0000 0 1 0 0
```

Archivo de salida:



RESULTADO: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
  IN    UP   DW  WPC  CLR   SP  PC
0000     0    0    0    1    0 0000
0000     0    0    0    0    0 0000
0000     0    0    0    0    0 0001
0022     0    0    1    0    0 0002
0000     0    0    0    0    0 0022
0000     0    0    0    0    0 0023
0061     1    0    1    0    0 0024
0000     0    0    0    0    1 0061
0000     0    0    0    0    1 0062
0064     1    0    1    0    1 0063
0000     0    0    0    0    2 0064
0000     0    0    0    0    2 0065
0000     0    0    0    0    2 0066
0000     0    1    0    0    2 0067
0000     0    0    0    0    1 0064
0000     0    0    0    0    1 0065
0000     0    1    0    0    1 0066
012C     0    0    1    0    0 0025
0379     1    0    1    0    0 012C
0000     0    0    0    0    1 0379
0000     0    0    0    0    1 037A
0000     0    0    0    0    1 037B
0000     0    1    0    0    1 037C
0000     0    1    0    0    0 012D
```

Diagrama RTL: