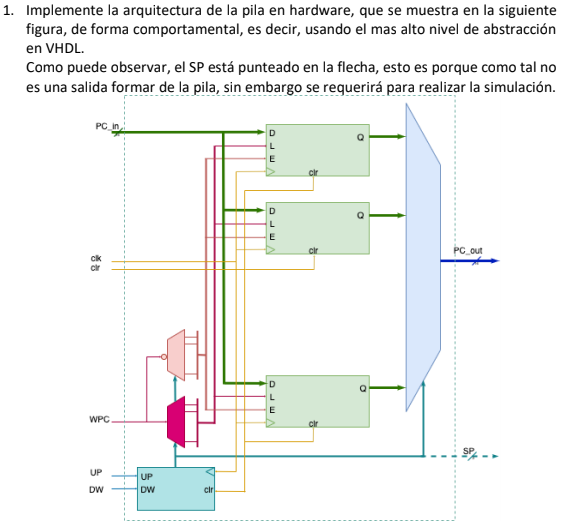
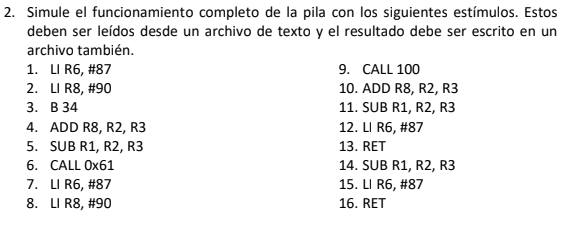
# Práctica 10

## Pila Hardware 2

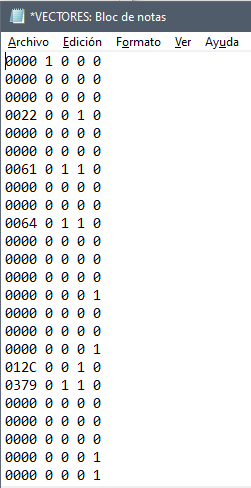






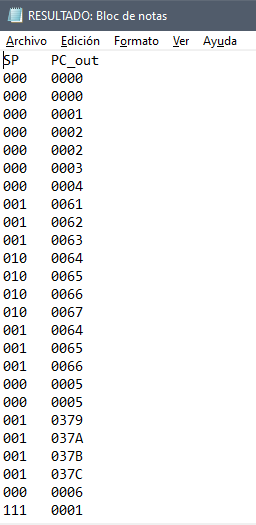
Escribiremos cada una de las instrucciones en un archivo de vectores de entrada al que nombraremos como “VECTORES.txt”, además de estas instrucciones agregaremos una instrucción de borrado para asegurar el correcto funcionamiento de la pila.

A continuación, se muestra el contenido del archivo “VECTORES.txt





Al leer el archivo con los vectores de entrada se obtuvo el siguiente archivo, el cual tiene el formato indicado, además, al final del archivo es posible observar el desbordamiento de la pila que se esperaba por el número de *instrucciones* RET en la lista.



### Forma de onda de la simulación.



### Código de implementación.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity main is

    Port ( pc\_in : in  STD\_LOGIC\_VECTOR (15 downto 0);

           up : in  STD\_LOGIC;

           dw : in  STD\_LOGIC;

           wpc : in  STD\_LOGIC;

           clk : in  STD\_LOGIC;

           clr : in STD\_LOGIC;

           pc\_out : out  STD\_LOGIC\_VECTOR (15 downto 0);

           sp\_out  : out STD\_LOGIC\_VECTOR(2 downto 0));

end main;

architecture Behavioral of main is

type nivel is array(0 to 7)of std\_logic\_vector(15 downto 0);

signal pila : nivel;

begin

process(clk, clr, pila)

variable sp : integer range 0 to 7:=0;

begin

    if(clr = '1')then           *--RESET*

        pila <= (others => (others => ('0')));

        sp := 0;

    elsif(rising\_edge(clk))then

            if(up = '0' and dw = '0' and wpc = '0')then     *--RETENCI�N INCREMENTO PC*

                sp := sp;

                pila(sp) <= pila(sp) + 1;

            elsif(up = '0' and dw = '0' and wpc = '1')then  *--SALTOS*

                pila(sp) <= pc\_in;

            elsif(up = '1' and dw = '0' and wpc = '1')then  *--CALL*

                sp := sp + 1;

                pila(sp) <= pc\_in;

            elsif(up = '0' and dw = '1' and wpc = '0')then  *--DECREMENTO*

                sp := sp - 1;

                if(sp < 0) then

                    sp := 7;

                end if;

                pila(sp) <= pila(sp) + 1;

            end if;

    end if;

    pc\_out <= pila(sp);

    sp\_out <= std\_logic\_vector(to\_unsigned(sp, sp\_out'length));

end process;

end Behavioral;

### Código de simulación

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 test IS

END test;

ARCHITECTURE behavior OF test IS

*-- Component Declaration for the Unit Under Test (UUT)*

    COMPONENT main

    PORT(

         pc\_in : IN  std\_logic\_vector(15 downto 0);

         pc\_out : OUT  std\_logic\_vector(15 downto 0);

         up : IN  std\_logic;

         dw : IN  std\_logic;

         wpc : IN  std\_logic;

         clk : IN  std\_logic;

         clr : IN  std\_logic;

            sp\_out  : out STD\_LOGIC\_VECTOR(2 downto 0)

        );

    END COMPONENT;

*--Inputs*

   signal pc\_in : std\_logic\_vector(15 downto 0) := (others => '0');

   signal up : std\_logic := '0';

   signal dw : std\_logic := '0';

   signal wpc : std\_logic := '0';

   signal clk : std\_logic := '0';

   signal clr : std\_logic := '0';

*--Outputs*

   signal pc\_out : std\_logic\_vector(15 downto 0);

    signal sp\_out  : STD\_LOGIC\_VECTOR(2 downto 0);

*-- Clock period definitions*

   constant clk\_period : time := 10 ns;

BEGIN

*-- Instantiate the Unit Under Test (UUT)*

   uut: main PORT MAP (

          pc\_in => pc\_in,

          pc\_out => pc\_out,

          up => up,

          dw => dw,

          wpc => wpc,

          clk => clk,

          clr => clr,

             sp\_out => sp\_out

        );

*-- 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

*--Variables para el manejo del archivo*

        file ARCH\_RES : TEXT;

        file ARCH\_VEC : TEXT;

        variable LINEA\_RES : line;

        variable LINEA\_VEC : line;

*--Variables de la entidad*

        variable var\_up: std\_logic;

        variable var\_dw: std\_logic;

        variable var\_wpc: std\_logic;

        variable var\_clr: std\_logic;

        variable CADENA: string(1 to 6);

        variable var\_pc\_in : std\_logic\_vector(15 downto 0);

        variable var\_pc\_out: std\_logic\_vector(15 downto 0);

        variable var\_sp\_out: std\_logic\_vector(2 downto 0);

   begin

        file\_open(ARCH\_VEC, "D:\ESCOM\ARQUITECTURA\Practica10\pila\pila.srcs\sim\_1\new\VECTORES.txt", READ\_MODE);

        file\_open(ARCH\_RES, "D:\ESCOM\ARQUITECTURA\Practica10\pila\pila.srcs\sim\_1\new\RESULTADO.txt", WRITE\_MODE);

        CADENA :="SP    ";

        write(LINEA\_RES, CADENA, left, CADENA'LENGTH);

        CADENA :="PC\_out";

        write(LINEA\_RES, CADENA, left, CADENA'LENGTH+2);

        writeline(ARCH\_RES, LINEA\_RES);

*-- hold reset state for 100 ns.*

      wait for 2 ps;

            FOR I IN 0 TO 24 LOOP

                readline(ARCH\_VEC,LINEA\_VEC);

                Hread(LINEA\_VEC, var\_pc\_in);

                pc\_in <= var\_pc\_in;

                read(LINEA\_VEC, var\_clr);

                clr <= var\_clr;

                read(LINEA\_VEC, var\_wpc);

                wpc <= var\_wpc;

                read(LINEA\_VEC, var\_up);

                up <= var\_up;

                read(LINEA\_VEC, var\_dw);

                dw <= var\_dw;

                WAIT UNTIL RISING\_EDGE(CLK);

                var\_pc\_out := pc\_out;

                var\_pc\_in  := pc\_in;

                var\_up     :=up;

                var\_dw    :=dw;

                var\_wpc    :=wpc;

                var\_clr    :=clr;

                var\_sp\_out :=sp\_out;

                write(LINEA\_RES, var\_sp\_out,left, 6);

                Hwrite(LINEA\_RES, var\_pc\_out, left, 8);

                writeline(ARCH\_RES, LINEA\_RES);

            end loop;

            file\_close(ARCH\_VEC);  *-- cierra el archivo*

            file\_close(ARCH\_RES);  *-- cierra el archivo*

      wait for clk\_period\*10;

      wait;

   end process;

END;

### Diagrama RTL

### 