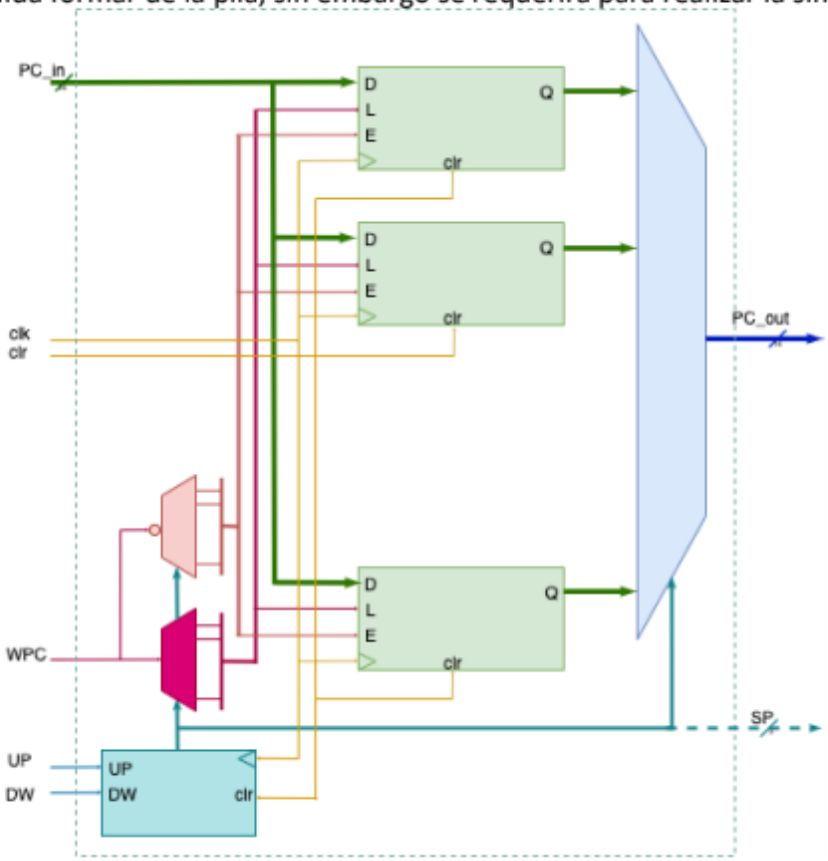
Práctica 10

Pila Hardware 2

1. Implemente la arquitectura de la pila en hardware, que se muestra en la siguiente figura, de forma comportamental, es decir, usando el mas alto nivel de abstracción en VHDL.
   Como puede observar, el SP está punteado en la flecha, esto es porque como tal no es una salida formar de la pila, sin embargo se requerirá para realizar la simulación.

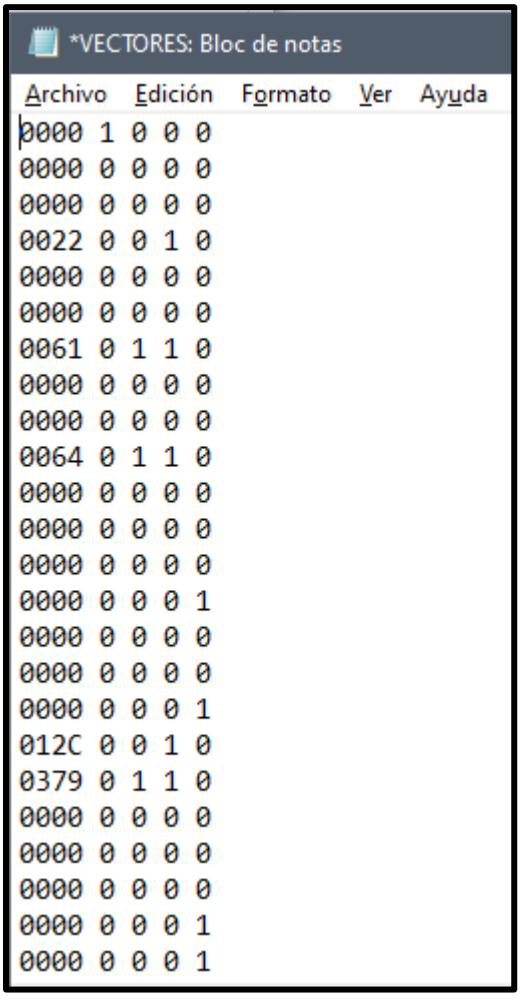


2. Simule el funcionamiento completo de la pila con los siguientes estímulos. Estos deben ser leídos desde un archivo de texto y el resultado debe ser escrito en un archivo también.
   1. LI R6, #87
   2. LI R8, #90
   3. B 34
   4. ADD R8, R2, R3
   5. SUB R1, R2, R3
   6. CALL 0x61
   7. LI R6, #87
   8. LI R8, #90
   9. CALL 100
   10. ADD R8, R2, R3
   11. SUB R1, R2, R3
   12. LI R6, #87
   13. RET
   14. SUB R1, R2, R3
   15. LI R6, #87
   16. RET

17. B 300
18. CALL 889
19. ADD R8, R2, R3
20. SUB R1, R2, R3
21. LI R6, #87
22. RET
23. RET

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



Escriba el resultado en el archivo en dos columnas en formato hexadecimal

SP	PC

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.

RESULTADO: Bloc de notas

Archivo Edición Formato Ver Ayuda

SP	PC_out
000	0000
000	0000
000	0001
000	0002
000	0002
000	0003
000	0004
001	0061
001	0062
001	0063
010	0064
010	0065
010	0066
010	0067
001	0064
001	0065
001	0066
000	0005
000	0005
001	0379
001	037A
001	037B
001	037C
000	0006
111	0001

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.s
rcs\sim_1\new\VECTORES.txt", READ_MODE);
        file_open(ARCH_RES, "D:\ESCOM\ARQUITECTURA\Practica10\pila\pila.s
rcs\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