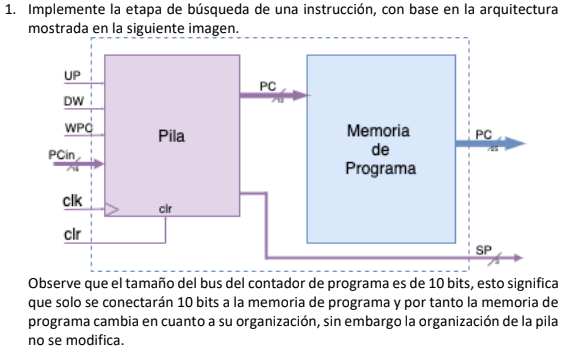
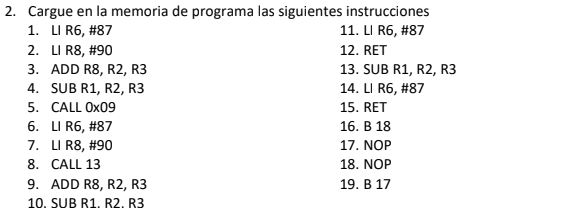
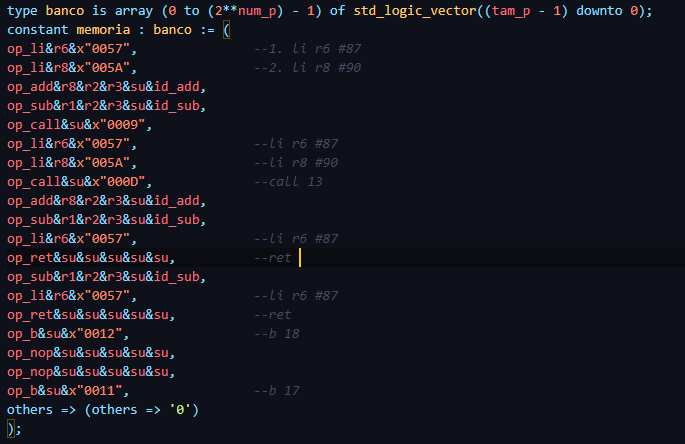
# Práctica 11.

## Pila de Memoria.



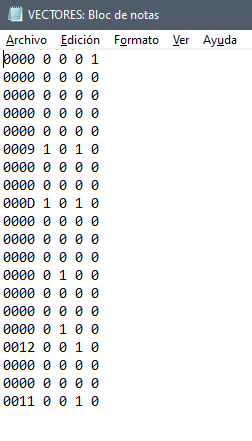


A continuación, se muestra como se han cargado todas las instrucciones a la memoria de programa.



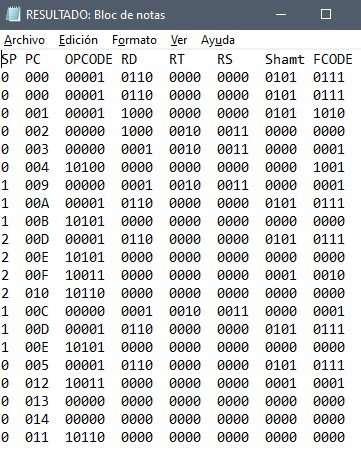


Se utilizo el siguiente archivo para la simulación del programa, en el se introducen los datos PC\_in, UP, DOWN, WPC y CLR en ese orden.





A continuación, se muestra el archivo de salida, en el se distingue que se ejecutan las instrucciones previamente cargadas en la memoria de programa.



## 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 tb\_pila\_mem IS

END tb\_pila\_mem;

ARCHITECTURE behavior OF tb\_pila\_mem IS

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

    COMPONENT main

    PORT(

         inst : OUT  std\_logic\_vector(24 downto 0);

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

         pc\_out : out  std\_logic\_vector(9 downto 0);

         sp\_out : out std\_logic\_vector(2 downto 0);

         clk : IN  std\_logic;

         clr : IN  std\_logic;

         up : IN  std\_logic;

         dw : IN  std\_logic;

         wpc : IN  std\_logic

        );

    END COMPONENT;

*--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 inst : std\_logic\_vector(24 downto 0);

   signal pc\_out : std\_logic\_vector(9 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 (

          inst => inst,

          pc\_in => pc\_in,

          pc\_out => pc\_out,

          sp\_out => sp\_out,

          clk => clk,

          clr => clr,

          up => up,

          dw => dw,

          wpc => wpc

        );

*-- 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;

    variable LINEA\_RES : line;

   variable var\_inst : std\_logic\_vector(24 downto 0);

    file ARCH\_VEC : TEXT;

    variable LINEA\_VEC : line;

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

    variable var\_pc\_out : std\_logic\_vector(9 downto 0);

    variable var\_clk,var\_clr,var\_up,var\_dw,var\_wpc: std\_logic;

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

    variable cadena2 : string(1 to 2);

    variable cadena5 : string(1 to 5);

    variable cadena6 : string(1 to 6);

   begin

        file\_open(ARCH\_RES, "D:\ESCOM\ARQUITECTURA\Practica11\pilaMem\pilaMem.srcs\sim\_1\new\RESULTADO.TXT", WRITE\_MODE);

        file\_open(ARCH\_VEC, "D:\ESCOM\ARQUITECTURA\Practica11\pilaMem\pilaMem.srcs\sim\_1\new\VECTORES.TXT", READ\_MODE);

        cadena2 := "SP";

        write(LINEA\_RES, cadena2, left, 3);

        cadena2 := "PC";

        write(LINEA\_RES, cadena2, left, 5);

        cadena6 := "OPCODE";

        write(LINEA\_RES, cadena6, left, 7);

        cadena2 := "RD";

        write(LINEA\_RES, cadena2, left, 6);

        cadena2 := "RT";

        write(LINEA\_RES, cadena2, left, 6);

        cadena2 := "RS";

        write(LINEA\_RES, cadena2, left, 6);

        cadena5 := "Shamt";

        write(LINEA\_RES, cadena5, left, 6);

        cadena5 := "FCODE";

        write(LINEA\_RES, cadena5, left, 6);

        writeline(ARCH\_RES,LINEA\_RES);*-- escribe la linea en el archivo*

        WAIT FOR 100 NS;

        FOR I IN 0 TO 20 LOOP

            readline(ARCH\_VEC,LINEA\_VEC); *-- lee una linea completa*

            Hread(LINEA\_VEC, var\_pc);

            pc\_in <= var\_pc;

            read(LINEA\_VEC, var\_up);

            up <= var\_up;

            read(LINEA\_VEC, var\_dw);

            dw <= var\_dw;

            read(LINEA\_VEC, var\_wpc);

            wpc <= var\_wpc;

            read(LINEA\_VEC, var\_clr);

            clr <= var\_clr;

            WAIT UNTIL RISING\_EDGE(CLK);    *--ESPERO AL FLANCO DE SUBIDA*

*--wait for 100 ns;*

            var\_pc\_out := pc\_out;

            var\_inst := inst;

            var\_sp\_out := sp\_out;

*--wait for 100 ns;*

            Hwrite(LINEA\_RES, var\_sp\_out, left, 3);

            Hwrite(LINEA\_RES, var\_pc\_out, left, 5); *--ESCRIBE EL CAMPO pc\_out*

            write(LINEA\_RES, var\_inst(24 downto 20), left, 7);

            write(LINEA\_RES, var\_inst(19 downto 16), left, 6);

            write(LINEA\_RES, var\_inst(15 downto 12), left, 6);

            write(LINEA\_RES, var\_inst(11 downto 8), left, 6);

            write(LINEA\_RES, var\_inst(7 downto 4), left, 6);

            write(LINEA\_RES, var\_inst(3 downto 0), left, 6);

            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 for clk\_period\*10;*

      wait;

   end process;

END;

## Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity main is

    Port ( inst : out  STD\_LOGIC\_VECTOR (24 downto 0);

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

           pc\_out: out std\_logic\_vector(9 downto 0);

           sp\_out : out std\_logic\_vector(2 downto 0);

           clk,clr,up,dw,wpc : in  STD\_LOGIC);

end main;

architecture Behavioral of main is

component pila is

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

           q : out  STD\_LOGIC\_VECTOR (9 downto 0);

           sp\_out : out std\_logic\_vector(2 downto 0);

           clk, clr, up, dw, wpc : in  STD\_LOGIC);

end component;

component mem\_prog is

    Port ( pc : in  STD\_LOGIC\_VECTOR (9 downto 0);

           inst : out  STD\_LOGIC\_VECTOR (24 downto 0));

end component;

signal pc\_out\_aux : std\_logic\_vector (9 downto 0);

begin

pila1 : pila

     port map (

        d => pc\_in,

        q => pc\_out\_aux,

        sp\_out => sp\_out,

        up => up,

        dw => dw,

        wpc => wpc,

        clk => clk,

        clr => clr

        );

mem\_prog1 : mem\_prog

     port map (

        pc => pc\_out\_aux,

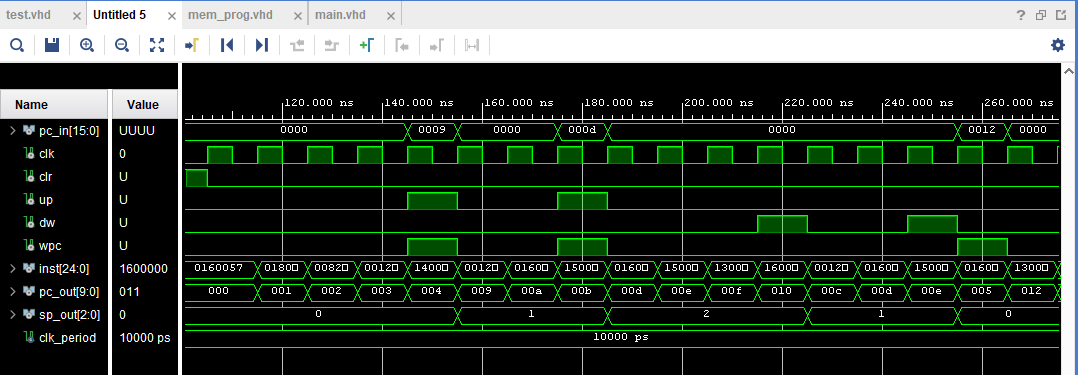
        inst => inst

        );

pc\_out <= pc\_out\_aux;

end Behavioral;

## Forma de onda de la simulación



## Diagrama RTL

