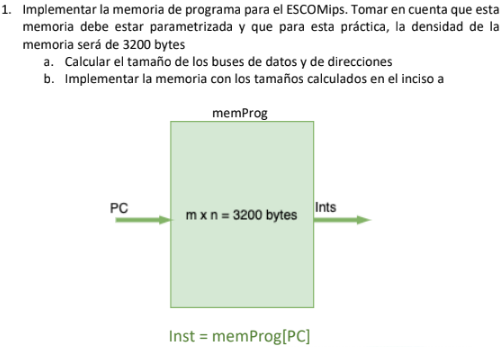
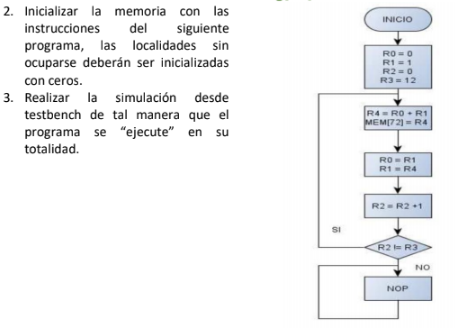
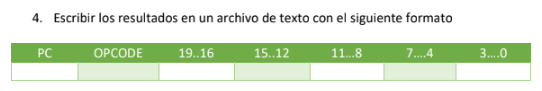
# Práctica 8

# Memoria de Programa

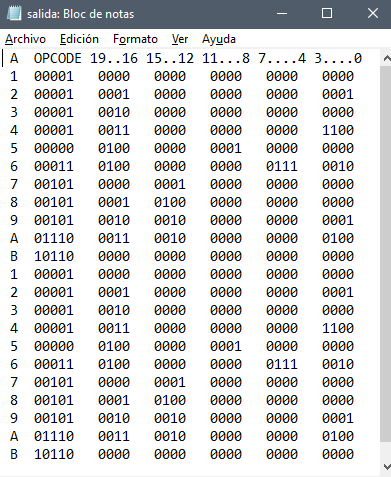


Dado que tenemos una densidad expresada en bytes es necesario calcular la densidad en bits, lo cual es muy sencillo pues es el resultado de . Posteriormente, y dado que se conoce que la longitud de cada instrucción es de 25 bits, se realiza la división siendo ese el número de palabras a almacenar. Ahora, para poder calcular el tamaño del bus de datos se requiere hacer el uso del logaritmo base 2, es decir: . Por lo cual obtenemos que el **tamaño del bus de direcciones es de 10bits, y el de datos es de 25 bits**.





Al realizar la simulación se obtienen los siguientes resultados.



**Código de implementación**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_arith.ALL;

use IEEE.STD\_LOGIC\_unsigned.ALL;

entity MemoPrograma is

    generic (

        m : integer := 10; *--tama�o del bus de direcciones*

        n : integer := 25 *--tama�o de palabra*

    );

    Port ( dir : in STD\_LOGIC\_VECTOR (m-1 downto 0);

           inst : out STD\_LOGIC\_VECTOR (n-1 downto 0));

end MemoPrograma;

architecture Behavioral of MemoPrograma is

*---------------------Tipo R------------------------*

constant tipoR : std\_logic\_vector(4 downto 0) := "00000";

*----------Instrucciones de Carga y Almacenamiento-------*

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";

*---------Aritmeticas con registros----------------*

constant add\_app : std\_logic\_vector(3 downto 0) := "0000";

constant sub\_app : std\_logic\_vector(3 downto 0) := "0001";

*---------Aritmeticas--------------------------------*

constant addi : std\_logic\_vector (4 downto 0)  := "00101";

constant subi : std\_logic\_vector (4 downto 0)  := "00110";

*--------.Logicas-----------------------------------*

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";

*---------Logicas con Registros--------------------*

constant andr : std\_logic\_vector (3 downto 0) := "0010";

constant orr  : std\_logic\_vector (3 downto 0) := "0011";

constant xorr : std\_logic\_vector (3 downto 0) := "0100";

constant nandr: std\_logic\_vector (3 downto 0) := "0101";

constant norr : std\_logic\_vector (3 downto 0) := "0110";

constant xnorr: std\_logic\_vector (3 downto 0) := "0111";

constant notr : std\_logic\_vector (3 downto 0) := "1000";

*--------Identificador Corrimiento R------------------------*

constant sllr : std\_logic\_vector (3 downto 0) := "1001";

constant srlr : std\_logic\_vector (3 downto 0) := "1010";

*--------Saltos Condicionales e Incondicionales-------------*

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";

*---------------Manejo de Subrutinas--------------------------------*

constant call : std\_logic\_vector (4 downto 0):= "10100";

constant ret : std\_logic\_vector (4 downto 0) := "10101";

*--------------- Otros ---------------------------------------------*

constant nop : std\_logic\_vector (4 downto 0) := "10110";

constant su  : std\_logic\_vector (3 downto 0) := "0000";

*-----------------  Registros --------------------------------------*

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";

type banco is array (0 to (2\*\*m)-1) of std\_logic\_vector(n-1 downto 0);

constant aux : banco := (

    LI&R0&x"0000",

    LI&R1&x"0001",

    LI&R2&x"0000",

    LI&R3&x"000C",

    tipoR&R4&R0&R1&SU&ADD\_APP,

    SWI&R4&x"0072",

    ADDI&R0&R1&x"000",

    ADDI&R1&R4&x"000",

    ADDI&R2&R2&x"001",

    BNEI&R3&R2&x"004",

    NOP&SU&SU&SU&SU&SU,

    others => (others => '0')

);

begin

    inst <= aux(conv\_integer(dir));

end Behavioral;

**Código de simulación**

LIBRARY STD;

LIBRARY ieee;

USE STD.TEXTIO.ALL;

USE ieee.std\_logic\_TEXTIO.ALL;

USE ieee.std\_logic\_1164.ALL;

USE ieee.std\_logic\_UNSIGNED.ALL;

USE ieee.std\_logic\_ARITH.ALL;

entity tb\_MemoPrograma is

end tb\_MemoPrograma;

architecture tb of tb\_MemoPrograma is

    component MemoPrograma

        port (dir  : in std\_logic\_vector (9 downto 0);

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

    end component;

    signal dir  : std\_logic\_vector (9 downto 0) := (others => '0');

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

begin

    dut : MemoPrograma

    port map (dir  => dir,

              inst => inst);

    stim\_proc: process

   file ARCH\_SAL : TEXT;

    variable LINEA\_SAL : line;

    VARIABLE CADENA : STRING(1 TO 6);

    VARIABLE CADENA2 : STRING(1 TO 2);

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

    variable var\_operacion: std\_logic\_vector(4 downto 0);

    variable var\_Parte1 : std\_logic\_vector(3 downto 0);

    variable var\_Parte2 : std\_logic\_vector(3 downto 0);

    variable var\_Parte3 : std\_logic\_vector(3 downto 0);

    variable var\_Parte4 : std\_logic\_vector(3 downto 0);

    variable var\_Parte5 : std\_logic\_vector(3 downto 0);

    variable ID : std\_logic\_vector(3 downto 0);

    begin

      file\_open(ARCH\_SAL, "D:\ESCOM\ARQUITECTURA\MemoriaPrograma\MemoriaPrograma.srcs\sim\_1\new\salida.txt", WRITE\_MODE);

*--para los encabezados*

    CADENA2:="A ";

    write(LINEA\_SAL, CADENA2, right, CADENA2'LENGTH+1);

    CADENA:="OPCODE";

    write(LINEA\_SAL, CADENA, right, CADENA'LENGTH+1);

    CADENA:="19..16";

    write(LINEA\_SAL, CADENA, right, CADENA'LENGTH+1);

    CADENA:="15..12";

    write(LINEA\_SAL, CADENA, right, CADENA'LENGTH+1);

    CADENA:="11...8";

    write(LINEA\_SAL, CADENA, right, CADENA'LENGTH+1);

    CADENA:="7....4";

    write(LINEA\_SAL, CADENA, right, CADENA'LENGTH+1);

    CADENA:="3....0";

    write(LINEA\_SAL, CADENA, right, CADENA'LENGTH+1);

    writeline(ARCH\_SAL,LINEA\_SAL);

    for veces in 0 to 1 loop

        for i in 0 to 10 loop

                dir <= conv\_std\_logic\_vector(i, 10);

                wait for 10 ps;

                var\_Instruccion:= inst;

                for j in 24 downto 20 loop

                    var\_operacion(j-20):=var\_Instruccion(j);

                end loop;

                for j in 19 downto 16 loop

                    var\_Parte1(j-16):=var\_Instruccion(j);

                end loop;

                for j in 15 downto 12 loop

                    var\_Parte2(j-12):=var\_Instruccion(j);

                end loop;

                for j in 11 downto 8 loop

                    var\_Parte3(j-8):=var\_Instruccion(j);

                end loop;

                for j in 7 downto 4 loop

                    var\_Parte4(j-4):=var\_Instruccion(j);

                end loop;

                for j in 3 downto 0 loop

                    var\_Parte5(j):=var\_Instruccion(j);

                end loop;

                ID:=conv\_std\_logic\_vector(i+1,4);

                Hwrite (LINEA\_SAL,ID,RIGHT,2);

                write(LINEA\_SAL,var\_operacion,right,7);

                write(LINEA\_SAL,var\_Parte1,right,7);

                write(LINEA\_SAL,var\_Parte2,right,7);

                write(LINEA\_SAL,var\_Parte3,right,7);

                write(LINEA\_SAL,var\_Parte4,right,7);

                write(LINEA\_SAL,var\_Parte5,right,7);

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

        end loop;

    end loop;

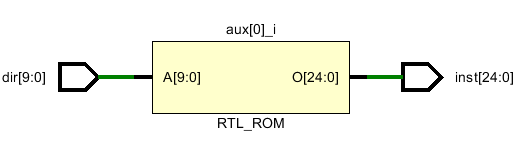
            file\_close(ARCH\_SAL);  *-- cierra el archivo*

        wait;

    end process;

end tb;

**Diagrama RTL**



**Forma de onda de salida**

