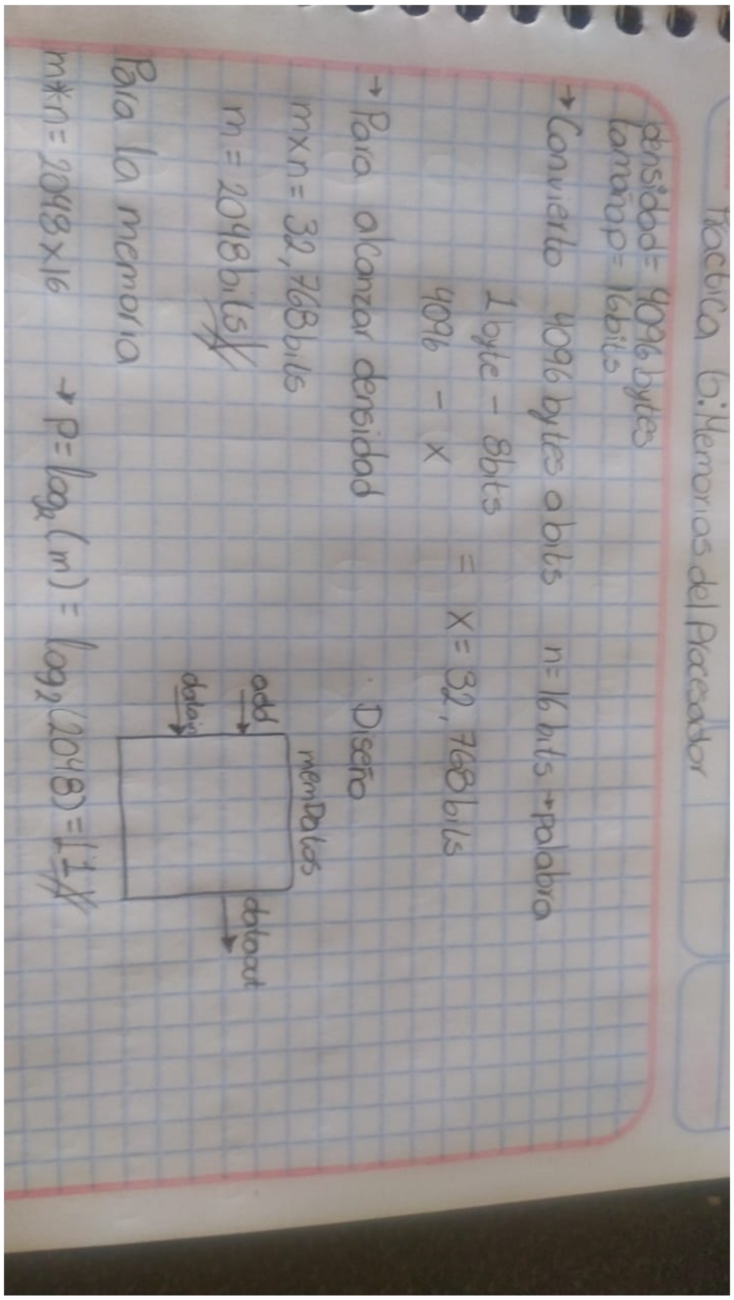
Practica 8: Memorias del procesado

Vargas Espino Carlos Hassan NL 33

Calculo necesario:



Practica 6: Memorias del Procesador

densidad = 4096 bytes
tamaño = 16 bits

→ Convierto 4096 bytes a bits $n = 16$ bits → palabra

1 byte – 8 bits
4096 – x   = $x = 32,768$ bits

→ Para alcanzar densidad

$m \times n = 32,768$ bits

$m = 2048$ bits

Para la memoria

$m * n = 2048 \times 16$ → $P = \log_2(m) = \log_2(2048) = 11$

Diseño
memDatos
add
datain
dataout

Código de implementación

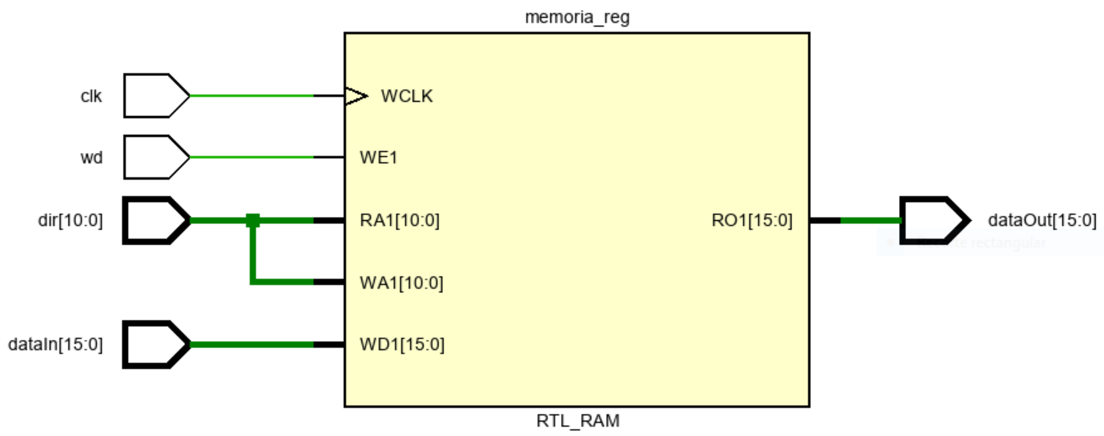
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity memoryData is
generic(
  p: integer := 11;
  n: integer := 16
);

Port (
       dir : in STD_LOGIC_VECTOR(p-1 downto 0);
        dataIn: in STD_LOGIC_VECTOR (n-1 downto 0);
        clk, wd : in STD_LOGIC;
        dataOut: out STD_LOGIC_VECTOR(n-1 downto 0));
end memoryData;

architecture Behavioral of memoryData is
type matrix is array (0 to (2**p)-1) of STD_LOGIC_VECTOR(n-1 downto 0);
signal memoria : matrix;
begin
    process(clk)
    begin
        if(rising_edge(clk)) then
            if(WD='1') then
                memoria(conv_integer(dir)) <= dataIn;
            end if;
        end if;
    end process;

    dataOut <=memoria(conv_integer(dir));

end Behavioral;
```

CÓDIGO TB

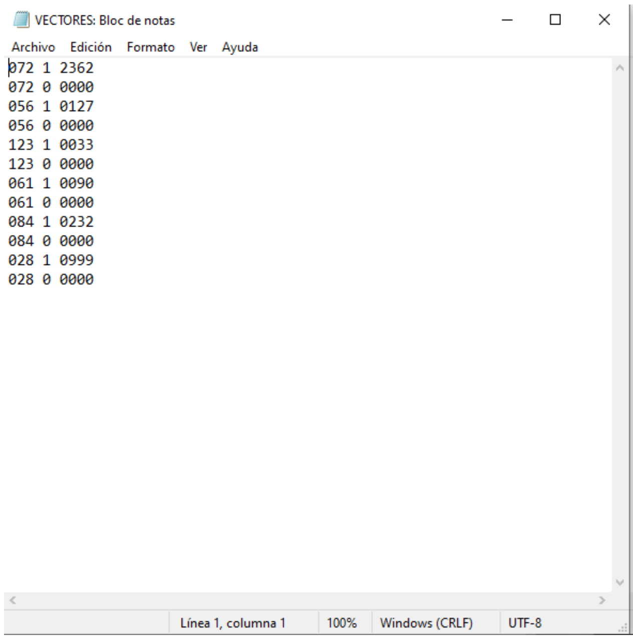
```
LIBRARY IEEE;
LIBRARY STD;
USE STD.TEXTIO.ALL;
USE IEEE.std_logic_TEXTIO.ALL; --PERMITE USAR STD_LOGIC
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_UNSIGNED.ALL;
USE IEEE.std_logic_ARITH.ALL;

entity tb_memoryData is
end tb_memoryData;

architecture Behavioral of tb_memoryData is

component memoryData is
    generic(
        p: integer := 11;
        n: integer := 16
    );
    Port (
        dir : in STD_LOGIC_VECTOR(10 downto 0);
        dataIn: in STD_LOGIC_VECTOR (15 downto 0);
        clk, wd : in STD_LOGIC;
        dataOut: out STD_LOGIC_VECTOR(15 downto 0));
end component;


signal dir: STD_LOGIC_VECTOR(10 downto 0);
signal dataIn:  STD_LOGIC_VECTOR (15 downto 0);
signal clk,wd: STD_LOGIC;


signal dataOut: STD_LOGIC_VECTOR (15 downto 0);


constant CLK_period : time := 60 ns;
```

```
begin

    memory:memoryData PORT MAP
        ( dir => dir,
          dataIn => dataIn,
          clk => clk,
          wd => wd,
          dataOut => dataOut
        );

    CLK_PROCESS: process
    begin
        clk <= '0';
        wait for CLK_period/2;
        clk <= '1';
        wait for CLK_period/2;
    end process;

    Stim_Process: process
        file ARCH_RES : TEXT;
        variable LINEA_RES : line;
        file ARCH_VEC : TEXT;
        variable LINEA_VEC : line;

        variable cadena: string(1 to 7);

        variable VAR_DIR: STD_LOGIC_VECTOR(10 downto 0);
        variable VAR_DATAIN,VAR_DATAOUT: STD_LOGIC_VECTOR (15 downto 0);
        variable VAR_WD: STD_LOGIC;

    begin
        file_open(ARCH_VEC,"C:\Users\Hassan\Desktop\Practicasarqui\Practica8\Resulvec\VECTORES.txt", READ_MODE);
        file_open(ARCH_RES,"C:\Users\Hassan\Desktop\Practicasarqui\Practica8\Resulvec\RESULTADOS.txt", WRITE_MODE);

        CADENA :="    add";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA :="     WD";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA :=" dataIn";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA :="dataOut";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);

        writeline(ARCH_RES,LINEA_RES);
        wait for 100 ns;

        for i in 0 to 11 loop
            readline(ARCH_VEC, LINEA_VEC);
            Hread(LINEA_VEC, VAR_DIR);
            dir<=VAR_DIR;
            read(LINEA_VEC,VAR_WD);
            WD<=VAR_WD;
            Hread(LINEA_VEC,VAR_DATAIN);
            dataIn<=VAR_DATAIN;

            wait until rising_edge(clk);
            VAR_DATAOUT := dataOut;

            Hwrite(LINEA_RES, VAR_DIR, right, 8);
            write(LINEA_RES, VAR_WD, right, 8);
            Hwrite(LINEA_RES, VAR_DATAIN, right, 8);
            Hwrite(LINEA_RES, VAR_DATAOUT, right, 8);

            writeline(ARCH_RES,LINEA_RES);
        end loop;
        file_close(ARCH_VEC);
        file_close(ARCH_RES);
        wait;
    end process;
end Behavioral;
```

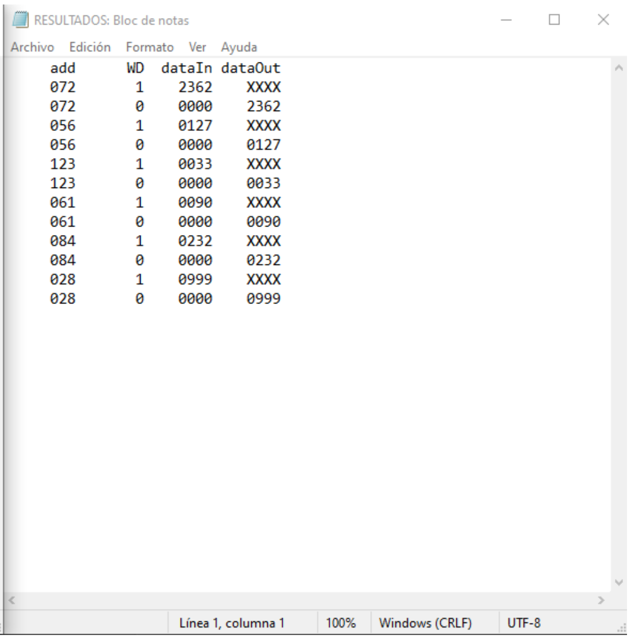
DISEÑO RTL



Resultados

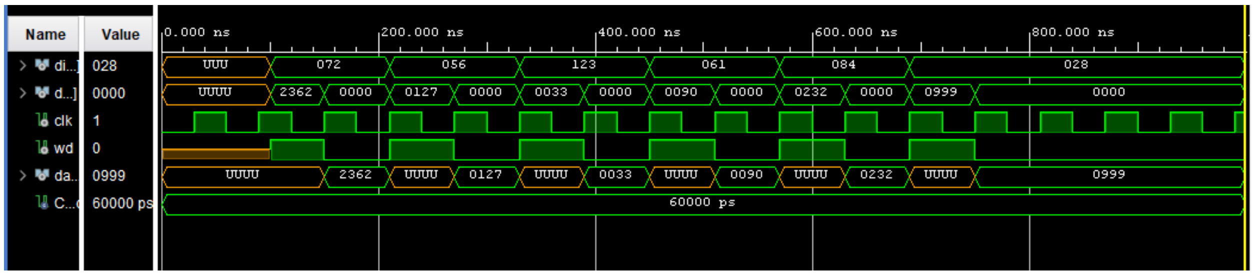

VECTORES: Bloc de notas
Archivo Edición Formato Ver Ayuda
Línea 1, columna 1 100% Windows (CRLF) UTF-8

```
072 1 2362
072 0 0000
056 1 0127
056 0 0000
123 1 0033
123 0 0000
061 1 0090
061 0 0000
084 1 0232
084 0 0000
028 1 0999
028 0 0000
```


RESULTADOS: Bloc de notas
Archivo Edición Formato Ver Ayuda
Línea 1, columna 1 100% Windows (CRLF) UTF-8

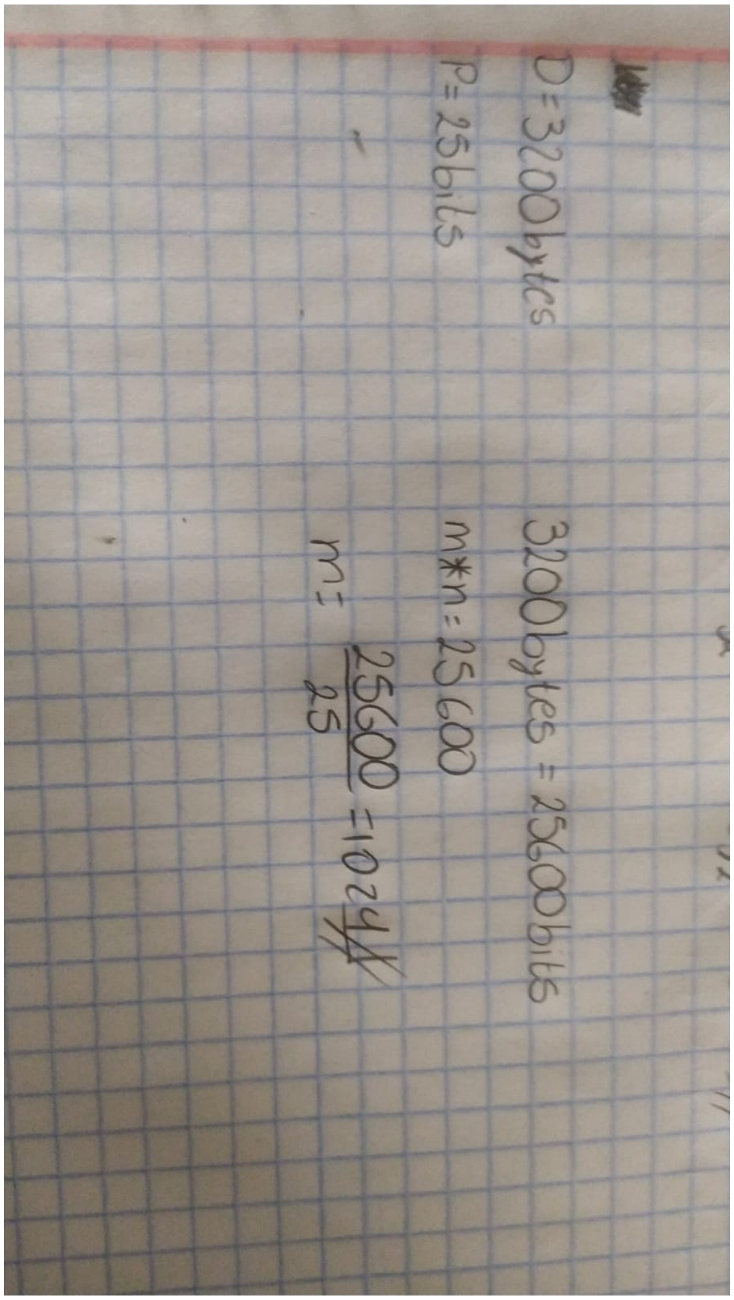
```
     add      WD  dataIn  dataOut
     072       1    2362     XXXX
     072       0    0000     2362
     056       1    0127     XXXX
     056       0    0000     0127
     123       1    0033     XXXX
     123       0    0000     0033
     061       1    0090     XXXX
     061       0    0000     0090
     084       1    0232     XXXX
     084       0    0000     0232
     028       1    0999     XXXX
     028       0    0000     0999
```

Salida



Implementación memoria de programa

Calculo necesario



D=3200bytes

P=25bits

3200bytes = 25600bits

m*n=25600

$m = \frac{25600}{25} = 1024$

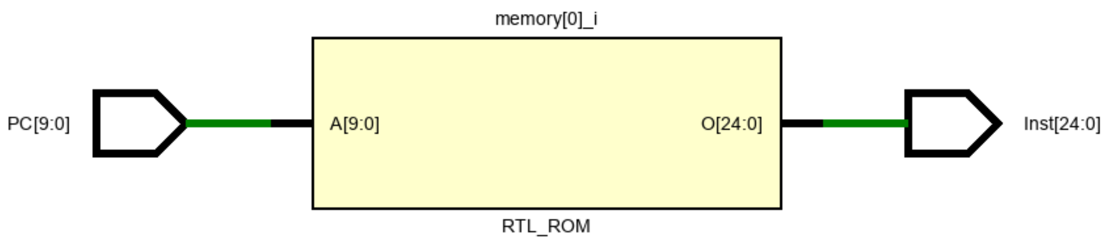
Programa

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

entity memoryProgram is
    generic(
        p: integer := 10; -- p=log_2(m) = log_2(2048) = 11
        n: integer := 25
    );
    Port (
        PC : in STD_LOGIC_VECTOR(p-1 downto 0);
        Inst: out STD_LOGIC_VECTOR(n-1 downto 0));
end memoryProgram;

architecture Behavioral of memoryProgram is

constant OP_R : STD_LOGIC_VECTOR(4 downto 0):= "00000"; --OPCODE INSTRUCCIONES R
constant SU: STD_LOGIC_VECTOR(3 downto 0):= "0000";


constant F_ADD: STD_LOGIC_VECTOR(3 downto 0):= "0000";
constant F_SUB: STD_LOGIC_VECTOR(3 downto 0):= "0001";
constant F_AND: STD_LOGIC_VECTOR(3 downto 0):= "0010";
constant F_OR: STD_LOGIC_VECTOR(3 downto 0):= "0011";
constant F_XOR: STD_LOGIC_VECTOR(3 downto 0):= "0100";
constant F_NAND: STD_LOGIC_VECTOR(3 downto 0):= "0101";
constant F_NOR: STD_LOGIC_VECTOR(3 downto 0):= "0110";
constant F_XNOR: STD_LOGIC_VECTOR(3 downto 0):= "0111";
constant F_NOT: STD_LOGIC_VECTOR(3 downto 0):= "1000";


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";
constant ADDI: STD_LOGIC_VECTOR(4 downto 0):= "00101";
constant SUBI: STD_LOGIC_VECTOR(4 downto 0):= "00110";
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";
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";

--CONSTANTES DE INSTRUCCIONES TIPO J
constant B: STD_LOGIC_VECTOR(4 downto 0):= "10011";
constant CALL: STD_LOGIC_VECTOR(4 downto 0):= "10100";

--CONSTANTES DE OTRAS INSTRUCCIONES
constant RET: std_logic_vector(24 downto 0):="1010100000000000000000000";
constant NOP: std_logic_vector(24 downto 0):="1011000000000000000000000";

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 matrix is array (0 to (2**p)-1) of STD_LOGIC_VECTOR(n-1 downto 0);
constant memory : matrix := (
    LI & R0 & x"0000", --LI R0, 0
    LI & R1 & x"0001", --LI R1, 1
    LI & R2 & x"0000", --LI R2, 0
    LI & R3 & x"000C", --LI R3, 12
    OP_R & R4 & R0 & R1 & SU & f_ADD,--ADD R4,R0,R1
    SWI & R4 & x"0048",--SWI R4, 72
    ADDI & R0 & R1 & x"000",-- ADDI R0, R1, #0
    ADDI & R1 & R4 & x"000",-- ADDI R1, R4, #0
    ADDI & R2 & R2 & x"001",-- ADDI R2, R2, #1
    BNEI & R3 & R2 & x"FFB",-- BNEI R3, R2, -5
    NOP, --NOP
    B & SU & x"000A", --B 10
    others => (others=> '0')
);
begin

    Inst <=memory(conv_integer(PC));

end Behavioral;
```

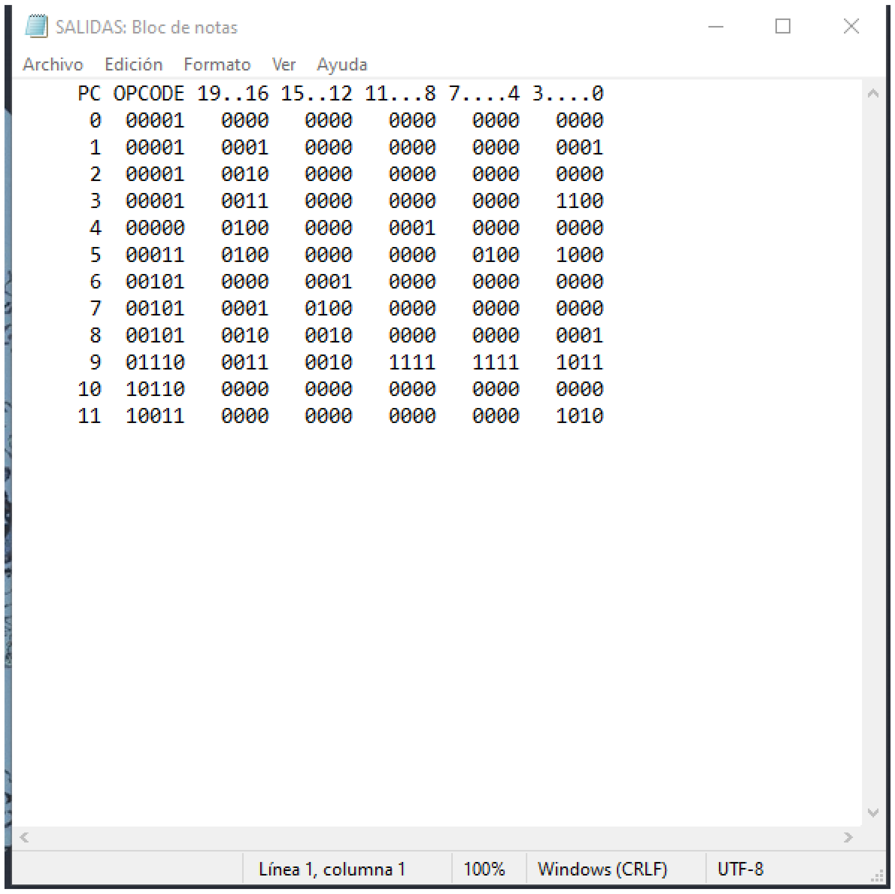
Programa tb

```
LIBRARY IEEE;
LIBRARY STD;
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;
use IEEE.numeric_std.all;

entity tb_memoryProgram is
--  Port ( );
end tb_memoryProgram;

architecture Behavioral of tb_memoryProgram is

component memoryProgram is
    Port (
        PC : in STD_LOGIC_VECTOR(9 downto 0);
        Inst: out STD_LOGIC_VECTOR(24 downto 0));
end component;

signal PC: STD_LOGIC_VECTOR(9 downto 0);
signal Inst: STD_LOGIC_VECTOR(24 downto 0);

begin

    memory: memoryProgram PORT MAP
        (
            PC => PC,
            Inst => Inst
        );
    Stim_Process: process
        file ARCH_RES : TEXT;
        variable LINEA_RES : line;
        file ARCH_VEC : TEXT;
        variable LINEA_VEC : line;

        variable cadena: string(1 to 6);

    begin
        file_open(ARCH_RES,"C:\Users\Hassan\Desktop\Practicasarqui\Practica8\memoriaprograma\ENTRADAS\SALIDAS", WRITE_MODE);
        cadena :="    PC";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        cadena :="OPCODE";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        cadena :="19..16";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        cadena :="15..12";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        cadena :="11...8";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        cadena :="7....4";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        cadena :="3....0";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);

        writeline(ARCH_RES,LINEA_RES);
        wait for 100 ns;

        for i in 0 to 11 loop

        PC <= std_logic_vector(to_unsigned(i,10));
        wait for 60 ns;
        write(LINEA_RES, i, right, 7);
        write(LINEA_RES, Inst(24 downto 20), right, 7);
        write(LINEA_RES, Inst(19 downto 16), right, 7);
        write(LINEA_RES, Inst(15 downto 12), right, 7);
        write(LINEA_RES, Inst(11 downto 8), right, 7);
        write(LINEA_RES, Inst(7 downto 4), right, 7);
        write(LINEA_RES, Inst(3 downto 0), right, 7);

        writeline(ARCH_RES,LINEA_RES);


        end loop;

        file_close(ARCH_VEC);
        file_close(ARCH_RES);
        wait;

    end process;

end Behavioral;
```

Diagrama

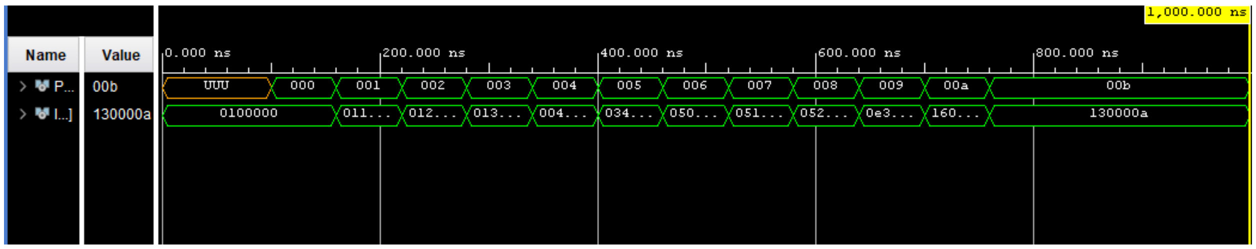


SALIDA


SALIDAS: Bloc de notas
Archivo Edición Formato Ver Ayuda
Línea 1, columna 1 100% Windows (CRLF) UTF-8

```
   PC OPCODE 19..16 15..12 11...8 7....4 3....0
    0  00001  0000   0000   0000   0000   0000
    1  00001  0001   0000   0000   0000   0001
    2  00001  0010   0000   0000   0000   0000
    3  00001  0011   0000   0000   0000   1100
    4  00000  0100   0000   0001   0000   0000
    5  00011  0100   0000   0000   0100   1000
    6  00101  0000   0001   0000   0000   0000
    7  00101  0001   0100   0000   0000   0000
    8  00101  0010   0010   0000   0000   0001
    9  01110  0011   0010   1111   1111   1011
   10  10110  0000   0000   0000   0000   0000
   11  10011  0000   0000   0000   0000   1010
```

SALIDA DE ONDA


1,000.000 ns
Name
Value
P...
00b
I...]
130000a
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
UUU
000
001
002
003
004
005
006
007
008
009
00a
00b
0100000
011...
012...
013...
004...
034...
050...
051...
052...
0e3...
160...
130000a