

Práctica 7

Memoria de datos

1. Implementar la memoria de datos para el ESCOMips. Tomar en cuenta que esta memoria debe estar parametrizada y que para esta práctica, la densidad de la memoria será de 4096 bytes
 - a. Calcular el tamaño de los buses de datos y de direcciones

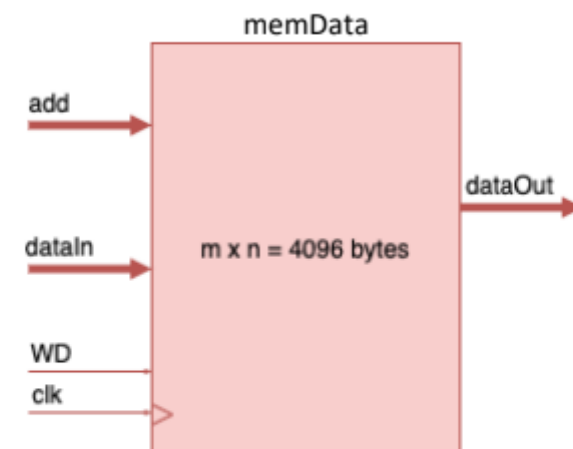
Dado que se trata de una densidad de 4096 bytes tenemos que la densidad convertida a bits es de $4096 \times 8 = 32768 \text{ bits}$.

Ahora que conocemos la densidad en bits, y dado que también conocemos la longitud de la literal de la instrucción, la cual es de 16 bits, lo siguiente será dividir la densidad entre la longitud de palabra.

Así tenemos que $\frac{32768}{16} = 2048$; por lo cual ahora conocemos el valor tanto de m como de n , siendo $m = 16$ y $n = 2048$.

Cabe aclarar que, dado que se poseerá un número de palabras igual a 2048, se requiere de un bus de direcciones de tamaño $\log_2 2048 = 11 \text{ bits}$.

- b. Implementar la memoria con los tamaños calculados en el inciso a



clk	WD	Operación
↑	1	memData[add]=dataIn
X	X	dataOut = memData[add]

a) Código de implementación

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;

entity Arreglo is
    Port (
        addr: 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 Arreglo;

architecture Behavioral of Arreglo is
    type banco is array (0 to 2047) of std_logic_vector(15 downto 0);
    signal aux : banco;
    begin

        process(clk)
        begin
            if (rising_edge(clk)) then
                if (wd = '1') then
                    aux(conv_integer(addr)) <= dataIn;
                end if;
            end if;
        end process;
        dataOut <= aux(conv_integer(addr));

    end Behavioral;

```

b) 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_Arreglo is
end tb_Arreglo;

architecture tb of tb_Arreglo is

    component Arreglo
        port (
            addr : in std_logic_vector (10 downto 0);
            dataIn : in std_logic_vector (15 downto 0);

```

```
        clk      : in std_logic;
        wd       : in std_logic;
        dataOut  : out std_logic_vector (15 downto 0));
end component;

signal addr     : std_logic_vector (10 downto 0);
signal dataIn   : std_logic_vector (15 downto 0);
signal clk      : std_logic;
signal wd       : std_logic;
signal dataOut  : std_logic_vector (15 downto 0);

constant CLK_period : time := 10 ns;

begin

    dut : Arreglo
    port map (addr     => addr,
              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;

    stimuli : process
    file ARCH_RES : TEXT;
    variable LINEA_RES : LINE;
    VARIABLE VAR_DATAOUT : STD_LOGIC_VECTOR(15 DOWNT0 0);

    file ARCH_VEC : TEXT;
    variable LINEA_VEC : line;

    VARIABLE VAR_RDATAIN : STD_LOGIC_VECTOR(15 DOWNT0 0);
    VARIABLE VAR_RADDRS : STD_LOGIC_VECTOR(10 DOWNT0 0);
    VARIABLE VAR_RWD : STD_LOGIC;

    VARIABLE CADENA : STRING(1 TO 4);
    VARIABLE CADENA_I : STRING(1 TO 6);
    VARIABLE CADENA_X : STRING(1 TO 5);
    VARIABLE CADENA_W : STRING(1 TO 7);

    begin
```

```
file_open(ARCH_RES, "D:\ESCOM\ARQUITECTURA\MemoriaDatos\MemoriDatos.
srcs\sim_1\new\resultado.txt", WRITE_MODE);
file_open(ARCH_VEC, "D:\ESCOM\ARQUITECTURA\MemoriaDatos\MemoriaDatos.
srcs\sim_1\new\entradas.txt", READ_MODE);

CADENA_X := " ADD ";
write(LINEA_RES, CADENA_X, right, CADENA_X'LENGTH+1);
CADENA := " WD ";
write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
CADENA_W := "dataIN ";
write(LINEA_RES, CADENA_W, right, CADENA_W'LENGTH+1);
CADENA_W := "dataOUT";
write(LINEA_RES, CADENA_W, right, CADENA_W'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_RADDRS);
    addr <= VAR_RADDRS;

    read(LINEA_VEC, VAR_RWD);
    wd <= VAR_RWD;

    Hread(LINEA_VEC, VAR_RDATAIN);
    dataIn <= VAR_RDATAIN;

    WAIT UNTIL RISING_EDGE(CLK);

    VAR_Dataout := dataOut;

    Hwrite(LINEA_RES, VAR_RADDRS, right, 5);
    write(LINEA_RES, VAR_RWD, right, 5);
    Hwrite (LINEA_RES, VAR_RDATAIN, right, 7);
    Hwrite (LINEA_RES, VAR_DATAOUT, right, 8);

    writeline(ARCH_RES, LINEA_RES);

end loop;

file_close(ARCH_VEC);
file_close(ARCH_RES);

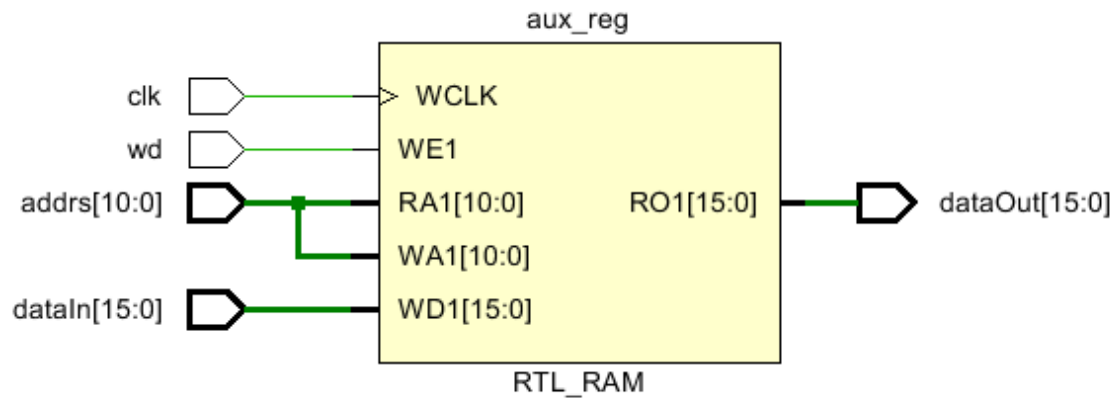
-- EDIT Add stimuli here
wait for CLK_period*10;

-- Stop the clock and hence terminate the simulation
wait;
```

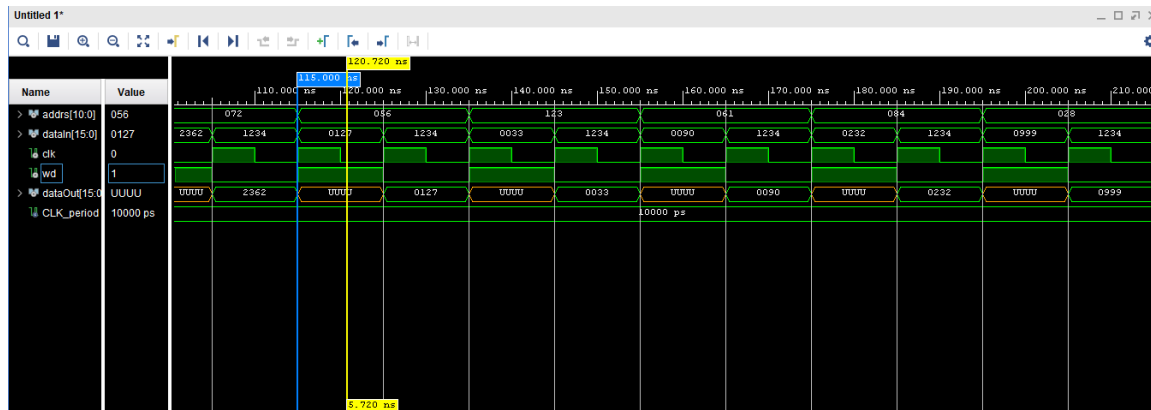
```
end process;

end tb;
```

c) Diagrama RTL



d) Forma de onda de simulación



2. Simular el funcionamiento de la memoria de datos utilizando archivos de texto

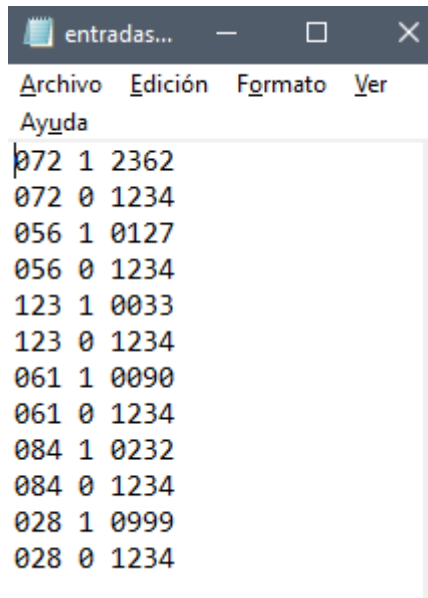
a. Construir el archivo de entrada con los siguientes estímulos

1. Escritura en la localidad x72 con el valor x2362
2. Lectura de la localidad x72
3. Escritura en la localidad x56 con el valor x127
4. Lectura de la localidad x56
5. Escritura en la localidad x123 con el valor x33
6. Lectura de la localidad x123
7. Escritura en la localidad x61 con el valor x90
8. Lectura de la localidad x61
9. Escritura en la localidad x84 con el valor x232
10. Lectura de la localidad x84
11. Escritura en la localidad x28 con el valor x999
12. Lectura de la localidad x28

b. Escribir el resultado en un archivo de texto con el siguiente formato

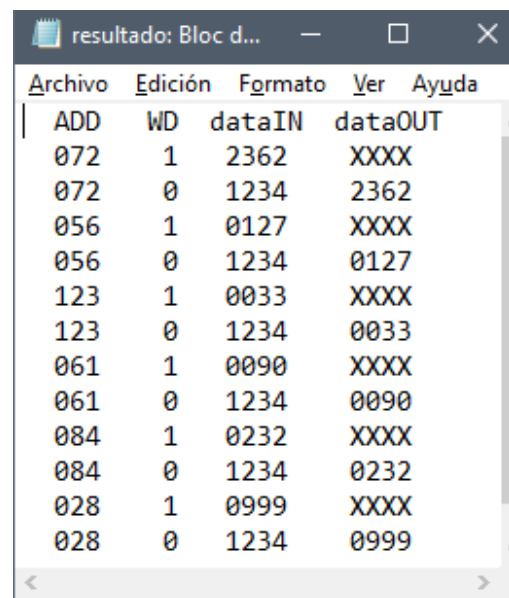
add	WD	dataIn	dataOut
hex	bin	hex	hex

Al introducir las instrucciones en el archivo **entradas.txt**, obtenemos los siguientes resultados en el archivo de salida **resultados.txt**.



```

Archivo  Edición  Formato  Ver
Ayuda
072 1 2362
072 0 1234
056 1 0127
056 0 1234
123 1 0033
123 0 1234
061 1 0090
061 0 1234
084 1 0232
084 0 1234
028 1 0999
028 0 1234
  
```



```

Archivo  Edición  Formato  Ver  Ayuda
ADD      WD      dataIN   dataOUT
072      1      2362     XXXX
072      0      1234     2362
056      1      0127     XXXX
056      0      1234     0127
123      1      0033     XXXX
123      0      1234     0033
061      1      0090     XXXX
061      0      1234     0090
084      1      0232     XXXX
084      0      1234     0232
028      1      0999     XXXX
028      0      1234     0999
  
```