Martínez Coronel Brayan Yosafat

# Código Paquete

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

package PAQUETE\_DATOS is

component archivo\_registro is

Port ( readReg1, readReg2, writeReg : in STD\_LOGIC\_VECTOR (3 downto 0);

shamt : in STD\_LOGIC\_VECTOR (3 downto 0);

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

clk, clear, dir, SHE, WR : in STD\_LOGIC;

readData1, readData2 : out STD\_LOGIC\_VECTOR(15 downto 0)

);

end component;

component alu is

generic (

m : integer := 16

);

Port ( A, B : in STD\_LOGIC\_VECTOR (m-1 downto 0);

op : in STD\_LOGIC\_VECTOR (3 downto 0);

OV, Z, C, N : out STD\_LOGIC;

R : inout STD\_LOGIC\_VECTOR (m-1 downto 0)

);

end component;

component memoria\_dato is

generic (

m : integer := 10;

n : integer := 16

);

Port ( add : in STD\_LOGIC\_VECTOR (m-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 component;

component memoria\_programa is

generic (

m : integer := 10;

n : integer := 25

);

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

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

);

end component;

component pila is

generic (

pc\_size : integer := 16;

n : integer := 3

);

Port ( PC\_in : in STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

clk, clr, WPC, UP, DW : in STD\_LOGIC;

PC\_out : out STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

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

);

end component;

end package;

# Código de implementación

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use WORK.PAQUETE\_DATOS.ALL;

entity ruta is

Port (

clk, clr, WPC, UP, DW, SHE, DIR, WR, WD, SR2, SWD, SR, SDMP, SDMD, SOP1, SOP2, SEXT : in STD\_LOGIC;

alu\_op : in STD\_LOGIC\_VECTOR (3 downto 0);

prefix : out STD\_LOGIC\_VECTOR (4 downto 0);

flags, sufix : out STD\_LOGIC\_VECTOR (3 downto 0)

);

end ruta;

architecture Behavioral of ruta is

constant pc\_size : integer := 16;

constant stack\_pointer\_bits : integer := 3;

constant instruction\_length : integer := 25;

constant data\_pointer\_bits : integer := 10;

constant data\_length : integer := 16;

signal PC\_in : STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

signal PC\_out : STD\_LOGIC\_VECTOR (pc\_size-1 downto 0);

signal aux\_sp : STD\_LOGIC\_VECTOR (stack\_pointer\_bits-1 downto 0);

signal instruction : STD\_LOGIC\_VECTOR (instruction\_length-1 downto 0);

signal readReg1, readReg2, writeReg, shamt : STD\_LOGIC\_VECTOR (3 downto 0);

signal writeData, readData1, readData2, aux\_data, aux\_B, aux\_sign : STD\_LOGIC\_VECTOR(data\_length-1 downto 0);

signal A, B, R : STD\_LOGIC\_VECTOR (data\_length-1 downto 0);

signal address : STD\_LOGIC\_VECTOR (data\_pointer\_bits-1 downto 0);

signal dataIn, dataOut : STD\_LOGIC\_VECTOR (data\_length-1 downto 0);

signal OV, Z, C, N : STD\_LOGIC;

begin

c\_pila : pila port map (

PC\_in => PC\_in,

clk => clk,

clr => clr,

WPC => WPC,

UP => UP,

DW => DW,

PC\_out => PC\_out,

SP => aux\_sp

);

c\_programa : memoria\_programa port map (

pc => PC\_out (9 downto 0),

inst => instruction

);

c\_archivo : archivo\_registro port map (

readReg1 => readReg1,

readReg2 => readReg2,

writeReg => writeReg,

shamt => shamt,

writeData => writeData,

clk => clk,

clear => clr,

dir => DIR,

SHE => SHE,

WR => WR,

readData1 => readData1,

readData2 => readData2

);

c\_alu : alu port map (

A => A,

B => B,

op => alu\_op,

OV => OV,

Z => Z,

C => C,

N => N,

R => R

);

c\_dato : memoria\_dato port map (

add => address,

dataIn => dataIn,

clk => clk,

WD => WD,

dataOut => dataOut

);

--Auxiliares

with SR select

aux\_data <= dataOut when '0',

R when others;

with instruction (11) select

aux\_sign <= "0000" & instruction (11 downto 0) when '0',

"1111" & instruction (11 downto 0) when others;

with SEXT select

aux\_B <= aux\_sign when '0',

"0000" & instruction (11 downto 0) when others;

--Pila

with SDMP select

PC\_in <= instruction (15 downto 0) when '0',

aux\_data when others;

--Archivo de Registros

readReg1 <= instruction (15 downto 12);

with SR2 select

readReg2 <= instruction (11 downto 8) when '0',

instruction (19 downto 16) when others;

writeReg <= instruction (19 downto 16);

shamt <= instruction (7 downto 4);

with SWD select

writeData <= instruction (15 downto 0) when '0',

aux\_data when others;

--ALU

with SOP1 select

A <= readData1 when '0',

PC\_out when others;

with SOP2 select

B <= readData2 when '0',

aux\_B when others;

--Memoria de Datos

with SDMD select

address <= R (9 downto 0) when '0',

instruction (9 downto 0) when others;

dataIn <= readData2;

--Salidas

flags <= OV & N & Z & C;

prefix <= instruction (24 downto 20);

sufix <= instruction (3 downto 0);

end Behavioral;

# Diagrama RTL

