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