Código de paquete

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
package Paquete is

component memoriaPrograma is
--Parametrizado
    generic(
    --NUMERO DE BITS DEL BUS DE DIRECCIONES
        p : integer := 10;
        d : integer := 25
    );
    Port ( PC : in STD_LOGIC_VECTOR (p-1 downto 0);
           Inst : out STD_LOGIC_VECTOR (d-1 downto 0));
end component;

component Pila is
    Port ( clr,clk,wpc,up,pw : in STD_LOGIC;
           PCin : in STD_LOGIC_VECTOR (15 downto 0);
           PCout : out STD_LOGIC_VECTOR (15 downto 0);
           SP : out STD_LOGIC_VECTOR (2 downto 0));
end component;

component registro is
    Port (
        writeData: in std_logic_vector (15 downto 0);
        writeReg: in std_logic_vector ( 3 downto 0 );
        readReg1: in std_logic_vector ( 3 downto 0 );
        readReg2: in std_logic_vector ( 3 downto 0 );
        shamt: in std_logic_vector ( 3 downto 0 );
        WR: in std_logic;
        DIR: in std_logic;
        SHE: in std_logic;
        clr: in std_logic;
        clk: in std_logic;
        readData1: out std_logic_vector ( 15 downto 0 );
        readData2: out std_logic_vector ( 15 downto 0 )
    );
end component;

component ALU4bitss is
 Port ( sela,selb,cin : in STD_LOGIC;
           a,b : in STD_LOGIC_VECTOR(15 downto 0);
           res : out STD_LOGIC_VECTOR(15 downto 0);
           cout,ov,z,N : out STD_LOGIC;
           op : in STD_LOGIC_VECTOR (1 downto 0));
end component;

component memoriaDatos is
--Parametrizado
```

```
    generic(
    --NUMERO DE BITS DEL BUS DE DIRECCIONES
        p : integer := 16;
        d : integer := 16
    );
    Port ( add : in STD_LOGIC_VECTOR (p-1 downto 0);
           dataIn : in STD_LOGIC_VECTOR(d-1 downto 0);
           clk, wd : in STD_LOGIC;
           dataOut : out STD_LOGIC_VECTOR (d-1 downto 0));
end component;
```

Código de implementación

```
library IEEE;
library work; --PAQUETES DEFINIDOS POR EL USUARIO
use work.Paquete.all;
use IEEE.STD_LOGIC_1164.ALL;
entity fetch is
    Port ( clk,clr,up,Down,wpc : in STD_LOGIC
           );
end fetch;

architecture Behavioral of fetch is
signal spaux : std_logic_vector(2 downto 0);
signal pc : std_logic_vector(15 downto 0);
--SEÑALES PARA EL ARCHIVO DE REGSTROS
signal auxReg : std_logic_vector(24 downto 0);
signal bandera : std_logic;
signal extensorSigno : std_logic_vector(15 downto 0);
signal extensorDireccion : std_logic_vector(15 downto 0);
--PRIMER MUX
signal selector1 : std_logic;
signal resultado1 : std_logic_vector(3 downto 0);

--SEGUNDO MUX
signal selector2 : std_logic;
signal resultado2 : std_logic_vector(15 downto 0);

--TERCER MUX
signal selector3 : std_logic;
signal resultado3 : std_logic_vector(15 downto 0);

--SEÑALES DEL CONTROLADOR QUE NO TENGO
signal SHE : std_logic;
signal DIR : std_logic;
signal WR : std_logic;
signal WD : std_logic;
--**SEÑALES PARA EL CONTROLADOR
```

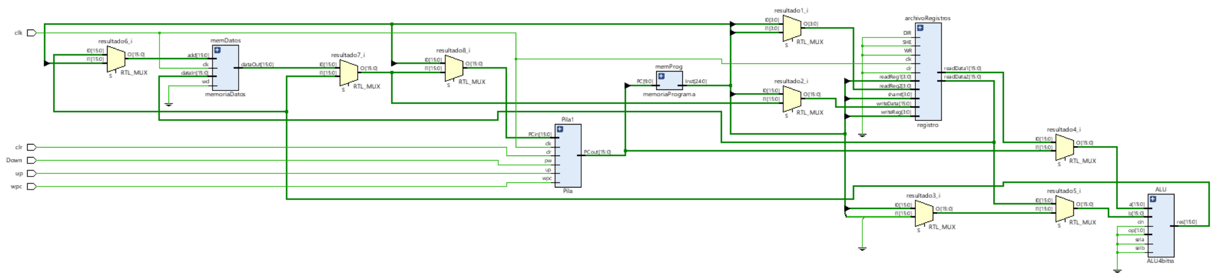
```

--SEÑALES PARA LA ALU
signal ALUOP : std_logic_vector(3 downto 0); --VIENE DEL CONTROLADOR
--CUARTO MUX
signal selector4 : std_logic;
signal resultado4 : std_logic_vector(15 downto 0);
signal readData1 : std_logic_vector(15 downto 0);

--QUINTO MUX
signal selector5 : std_logic;
signal resultado5 : std_logic_vector(15 downto 0);
signal readData2 : std_logic_vector(15 downto 0);
--**SEÑALES PARA LA ALU

--SEÑALES PARA LA MEMORIA DE DATOS
signal res : STD_LOGIC_VECTOR(15 downto 0);
signal dataOut : STD_LOGIC_VECTOR(15 downto 0);
--SEXTO MUX
signal selector6 : std_logic;
signal resultado6 : std_logic_vector(15 downto 0);
--SEPTIMO MUX
signal selector7 : std_logic;
signal resultado7 : std_logic_vector(15 downto 0);
--Octavo MUX
signal selector8 : std_logic;
signal resultado8 : std_logic_vector(15 downto 0);
--**SEÑALES PARA LA MEMORIA DE DATOS

--SEÑALES PARA E/S QUE YA NO SE USAN
signal clrReg : std_logic;


--BANDERAS DE LA ALU AL CONTROLADOR ¡CUIDAR ORDEN!
signal banderasALU : std_logic_vector(3 downto 0);

begin
--PILA
Pila1 : Pila
    Port map (
        clr => clr,
        clk => clk,
        wpc => wpc,
        up => up,
        pw => down,
        sp => spaux,
        PCin => resultado8,
        PCout => pc
        );
--*PILA

--MEMORIA DE PROGRAMA
memProg : memoriaPrograma
```

```
    Port map(
        PC => PC(9 downto 0),
        Inst => auxReg
    );
--*MEMORIA DE PROGRAMA

--ARCHIVO DE REGISTROS

--PRIMER MUX
    PROCESS (selector1) IS
        BEGIN
         CASE selector1 IS
           WHEN '0' => resultado1 <= auxReg(11 downto 8);
           WHEN '1' => resultado1 <= auxReg(19 downto 16);
         END CASE;
        END PROCESS;
--**PRIMER MUX

--SEGUNDO MUX
    PROCESS (selector2) IS
        BEGIN
         CASE selector2 IS
           WHEN '0' => resultado2 <= auxReg(15 downto 0);
           WHEN '1' => resultado2 <= resultado7;
         END CASE;
        END PROCESS;
--**SEGUNDO MUX

archivoRegistros : registro
    Port map(
        readReg1 => auxReg(15 downto 12),
        readReg2 => resultado1,
        writeReg => auxReg(19 downto 16),
        shamt =>  auxReg(7 downto 4),
        writeData => resultado2,
        WR => WR,
        DIR => DIR,
        SHE => SHE,
        clr => clrReg,
        clk => clk,
        readData1 => readData1,
        readData2 =>readData2
    );

--EXTENSOR DE SIGNO
bandera <= auxReg(11);
extensorSigno <= bandera & bandera & bandera & bandera & auxReg(11
downto 0);
--**EXTENSOR DE SIGNO
--EXTENSOR DE DIRECCION
extensorDireccion <= "0000" & auxReg(11 downto 0);
--**EXTENSOR DE DIRECCION
```

```vhdl

--TERCER MUX
    PROCESS (selector3) IS
        BEGIN
         CASE selector3 IS
           WHEN '0' => resultado3 <= extensorSigno;
           WHEN '1' => resultado3 <= extensorDireccion;
         END CASE;
        END PROCESS;
--**TERCER MUX
--*ARCHIVO DE REGISTROS

--ALU

--CUARTO MUX
    PROCESS (selector4) IS
        BEGIN
         CASE selector4 IS
           WHEN '0' => resultado4 <= readData1;
           WHEN '1' => resultado4 <= pc;
         END CASE;
        END PROCESS;
--**CUARTO MUX

--QUINTO MUX
    PROCESS (selector5) IS
        BEGIN
         CASE selector5 IS
           WHEN '0' => resultado5 <= readData2;
           WHEN '1' => resultado5 <= resultado3;
         END CASE;
        END PROCESS;
--**QUINTO MUX

ALU : ALU4bitss
 Port map(
        sela => ALUOP(3),
        selb => ALUOP(2),
        cin => ALUOP(2),
        OP => ALUOP (1 downto 0),
        a => resultado4,
        b => resultado5,
        res => res,
        cout => banderasAlu(0),
        ov => banderasAlu(1),
        z => banderasAlu(2),
        N => banderasAlu(3)
        );

--**ALU

--MEMORIA DE DATOS
```

```
--SEXTO MUX
    PROCESS (selector6) IS
        BEGIN
         CASE selector6 IS
           WHEN '0' => resultado6 <= res;
           WHEN '1' => resultado6 <= auxReg(15 downto 0);
         END CASE;
        END PROCESS;
--**SEXTO MUX

memDatos : memoriaDatos
    Port map(
           add => resultado6,
           dataIn => readData2,
           clk => clk,
           wd => wd,
           dataOut => dataOut
           );

--SEPTIMO MUX
    PROCESS (selector7) IS
        BEGIN
         CASE selector7 IS
           WHEN '0' => resultado7 <= dataOut;
           WHEN '1' => resultado7 <= res;
         END CASE;
        END PROCESS;
--**SEPTIMO MUX

--OCTAVO MUX
    PROCESS (selector8) IS
        BEGIN
         CASE selector8 IS
           WHEN '0' => resultado8 <= auxReg(15 downto 0);
           WHEN '1' => resultado8 <= resultado7;
         END CASE;
        END PROCESS;
--**OCTAVO MUX
--**MEMORIA DE DATOS

end Behavioral;
```

Diagrama RTL:


clk
clr
Down
up
wpc
resultado6_i
I0[15:0]
I1[15:0]
O[15:0]
S
RTL_MUX
memDatos
add[15:0]
clk
dataIn[15:0]
wd
dataOut[15:0]
memoriaDatos
resultado7_i
resultado8_i
Pila1
PCin[15:0]
clk
clr
pw
up
wpc
PCout[15:0]
Pila
memProg
PC[9:0]
Inst[24:0]
memoriaPrograma
resultado1_i
I0[3:0]
I1[3:0]
O[3:0]
resultado2_i
archivoRegistros
DIR
SHE
WR
clk
clr
readReg1[3:0]
readReg2[3:0]
shamt[3:0]
writeData[15:0]
writeReg[3:0]
readData1[15:0]
readData2[15:0]
registro
resultado4_i
resultado3_i
resultado5_i
ALU
a[15:0]
b[15:0]
cin
op[1:0]
sela
selb
res[15:0]
ALU4bitss