INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

U.A: Arquitectura de Computadoras

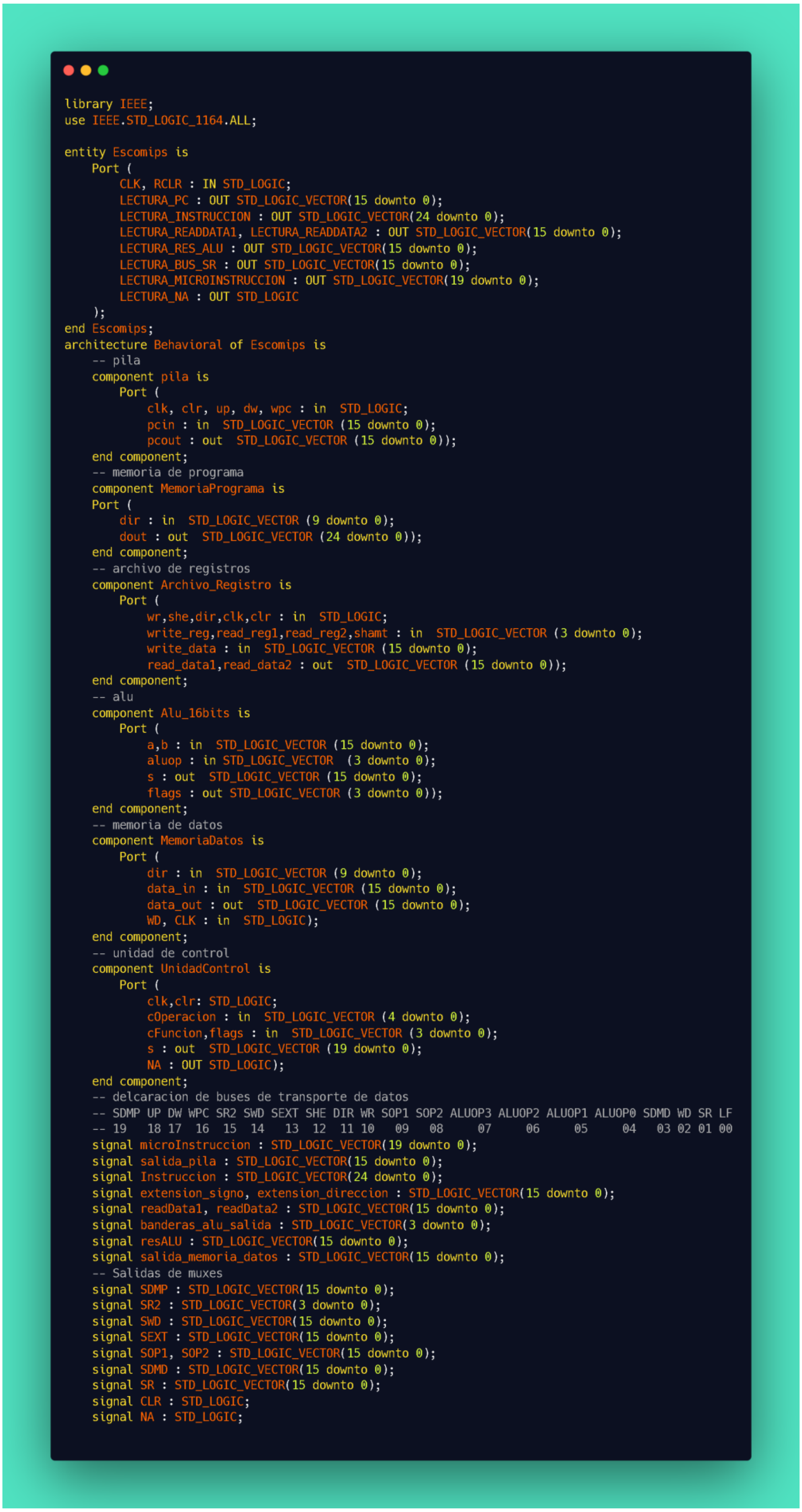
“Práctica 10: Ruta de datos del ESCOMIPS”

Grupo: 3CV11

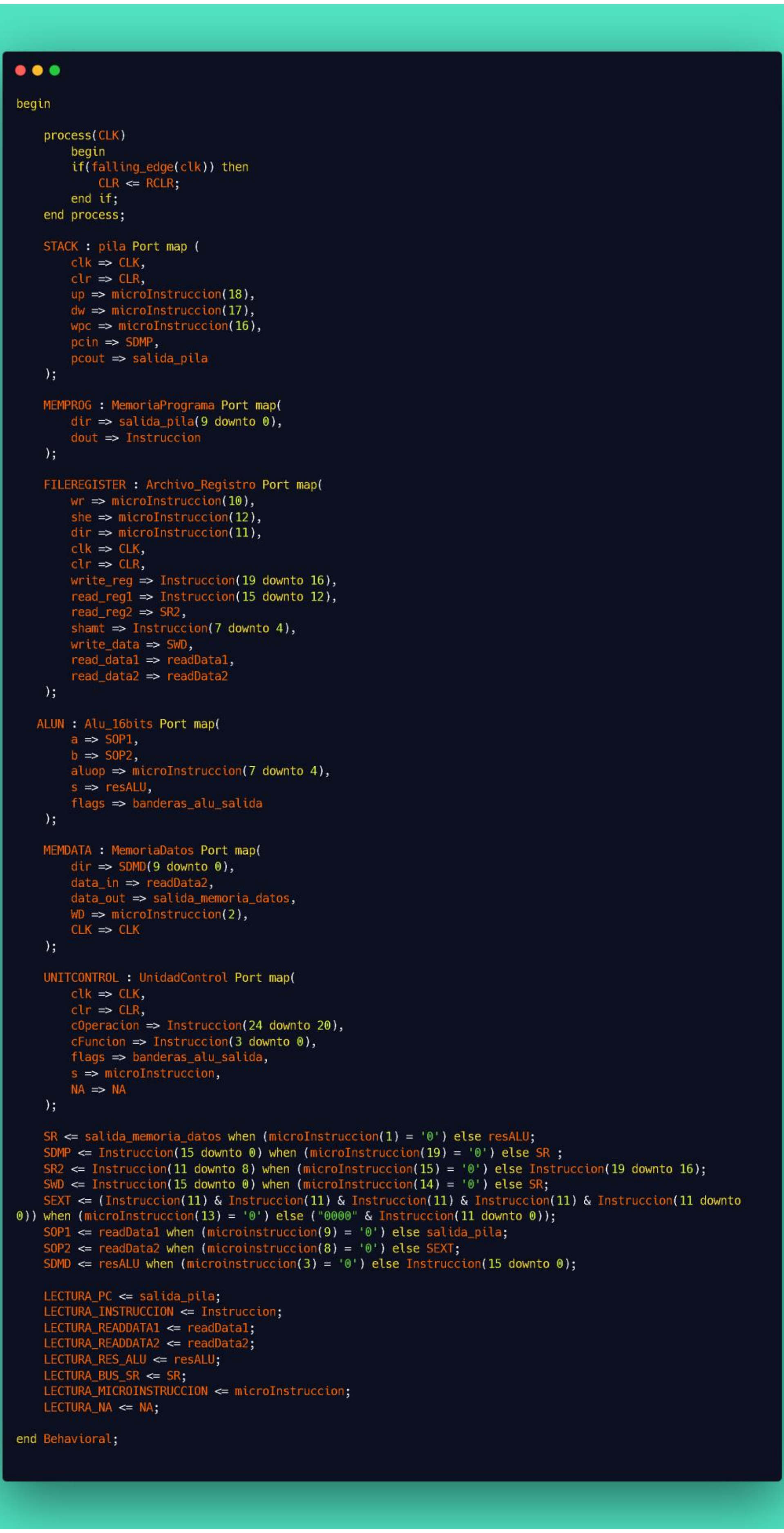
Profesor: Vega García Nayeli

Sánchez Becerra Ernesto Daniel

Código de implementación

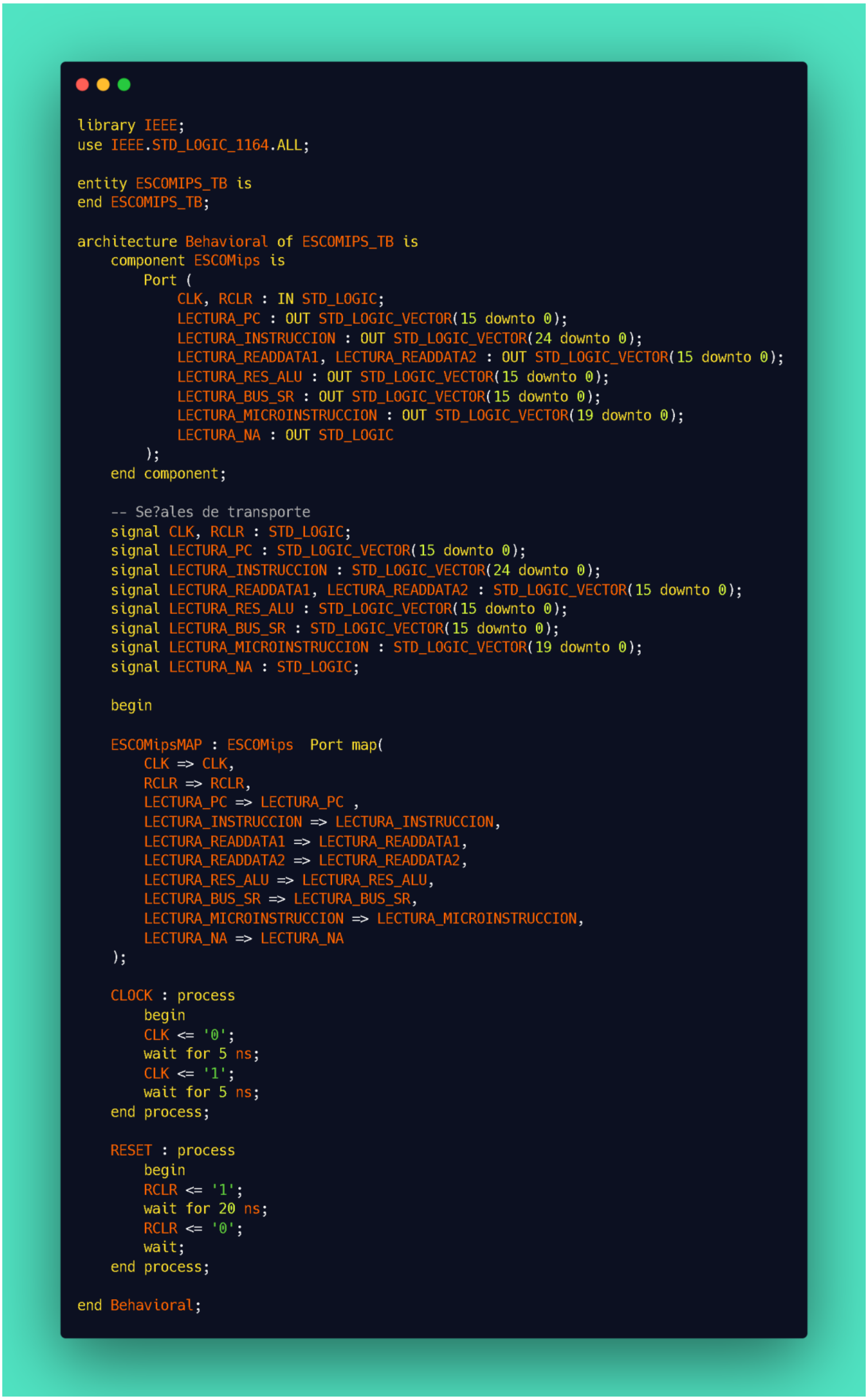


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Escomips is
    Port (
        CLK, RCLR : IN STD_LOGIC;
        LECTURA_PC : OUT STD_LOGIC_VECTOR(15 downto 0);
        LECTURA_INSTRUCCION : OUT STD_LOGIC_VECTOR(24 downto 0);
        LECTURA_READDATA1, LECTURA_READDATA2 : OUT STD_LOGIC_VECTOR(15 downto 0);
        LECTURA_RES_ALU : OUT STD_LOGIC_VECTOR(15 downto 0);
        LECTURA_BUS_SR : OUT STD_LOGIC_VECTOR(15 downto 0);
        LECTURA_MICROINSTRUCCION : OUT STD_LOGIC_VECTOR(19 downto 0);
        LECTURA_NA : OUT STD_LOGIC
    );
end Escomips;
architecture Behavioral of Escomips is
    -- pila
    component pila is
        Port (
            clk, clr, up, dw, wpc : in  STD_LOGIC;
            pcin : in  STD_LOGIC_VECTOR (15 downto 0);
            pcout : out  STD_LOGIC_VECTOR (15 downto 0));
    end component;
    -- memoria de programa
    component MemoriaPrograma is
    Port (
        dir : in  STD_LOGIC_VECTOR (9 downto 0);
        dout : out  STD_LOGIC_VECTOR (24 downto 0));
    end component;
    -- archivo de registros
    component Archivo_Registro is
        Port (
            wr,she,dir,clk,clr : in  STD_LOGIC;
            write_reg,read_reg1,read_reg2,shamt : in  STD_LOGIC_VECTOR (3 downto 0);
            write_data : in  STD_LOGIC_VECTOR (15 downto 0);
            read_data1,read_data2 : out  STD_LOGIC_VECTOR (15 downto 0));
    end component;
    -- alu
    component Alu_16bits is
        Port (
            a,b : in  STD_LOGIC_VECTOR (15 downto 0);
            aluop : in STD_LOGIC_VECTOR  (3 downto 0);
            s : out  STD_LOGIC_VECTOR (15 downto 0);
            flags : out STD_LOGIC_VECTOR (3 downto 0));
    end component;
    -- memoria de datos
    component MemoriaDatos is
        Port (
            dir : in  STD_LOGIC_VECTOR (9 downto 0);
            data_in : in  STD_LOGIC_VECTOR (15 downto 0);
            data_out : out  STD_LOGIC_VECTOR (15 downto 0);
            WD, CLK : in  STD_LOGIC);
    end component;
    -- unidad de control
    component UnidadControl is
        Port (
            clk,clr: STD_LOGIC;
            cOperacion : in  STD_LOGIC_VECTOR (4 downto 0);
            cFuncion,flags : in  STD_LOGIC_VECTOR (3 downto 0);
            s : out  STD_LOGIC_VECTOR (19 downto 0);
            NA : OUT STD_LOGIC);
    end component;
    -- delcaracion de buses de transporte de datos
    -- SDMP UP DW WPC SR2 SWD SEXT SHE DIR WR SOP1 SOP2 ALUOP3 ALUOP2 ALUOP1 ALUOP0 SDMD WD SR LF
    -- 19   18 17  16  15  14   13  12  11 10   09   08     07     06     05     04   03 02 01 00
    signal microInstruccion : STD_LOGIC_VECTOR(19 downto 0);
    signal salida_pila : STD_LOGIC_VECTOR(15 downto 0);
    signal Instruccion : STD_LOGIC_VECTOR(24 downto 0);
    signal extension_signo, extension_direccion : STD_LOGIC_VECTOR(15 downto 0);
    signal readData1, readData2 : STD_LOGIC_VECTOR(15 downto 0);
    signal banderas_alu_salida : STD_LOGIC_VECTOR(3 downto 0);
    signal resALU : STD_LOGIC_VECTOR(15 downto 0);
    signal salida_memoria_datos : STD_LOGIC_VECTOR(15 downto 0);
    -- Salidas de muxes
    signal SDMP : STD_LOGIC_VECTOR(15 downto 0);
    signal SR2 : STD_LOGIC_VECTOR(3 downto 0);
    signal SWD : STD_LOGIC_VECTOR(15 downto 0);
    signal SEXT : STD_LOGIC_VECTOR(15 downto 0);
    signal SOP1, SOP2 : STD_LOGIC_VECTOR(15 downto 0);
    signal SDMD : STD_LOGIC_VECTOR(15 downto 0);
    signal SR : STD_LOGIC_VECTOR(15 downto 0);
    signal CLR : STD_LOGIC;
    signal NA : STD_LOGIC;
```



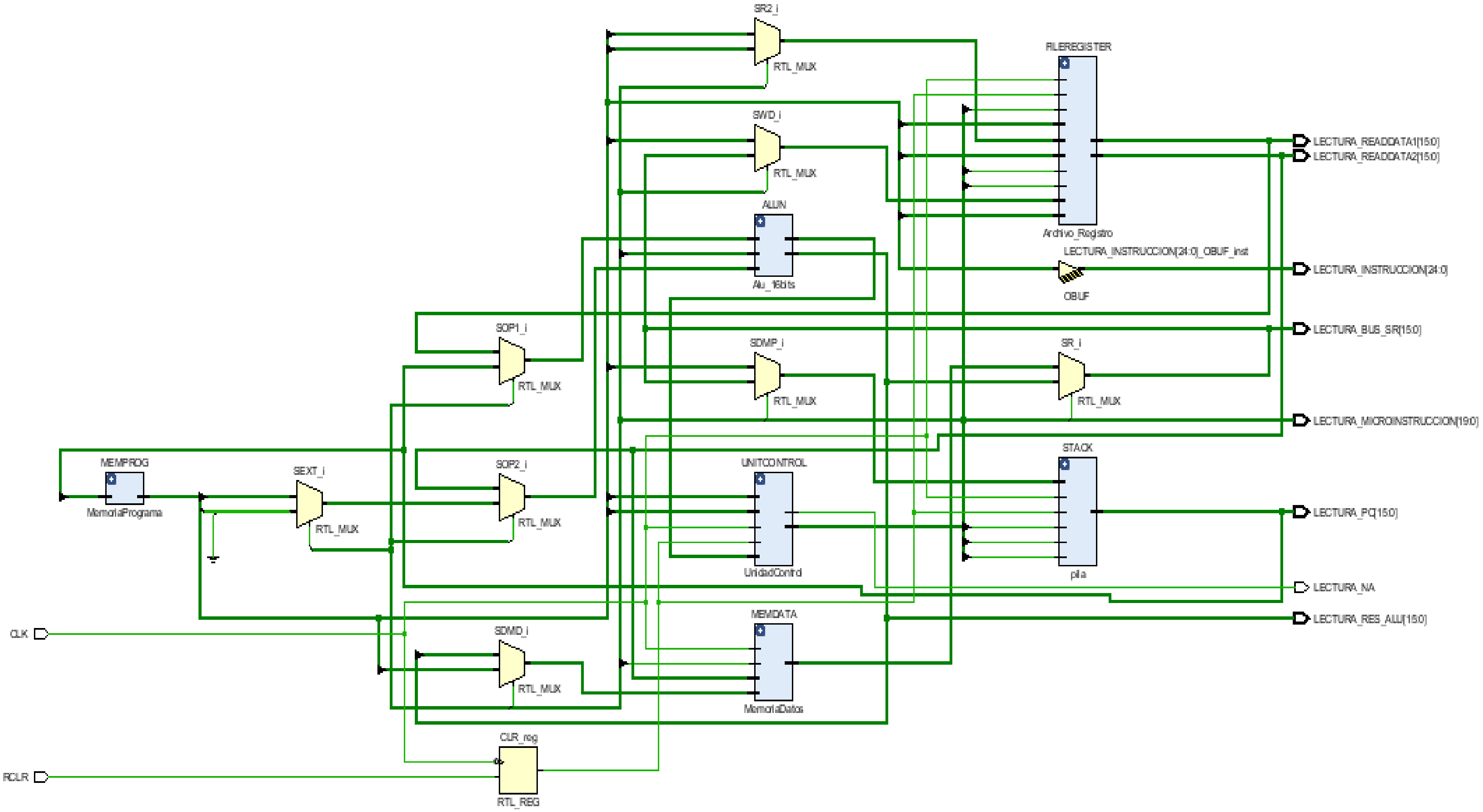
```vhdl
begin

    process(CLK)
        begin
        if(falling_edge(clk)) then
            CLR <= RCLR;
        end if;
    end process;

    STACK : pila Port map (
        clk => CLK,
        clr => CLR,
        up => microInstruccion(18),
        dw => microInstruccion(17),
        wpc => microInstruccion(16),
        pcin => SDMP,
        pcout => salida_pila
    );

    MEMPROG : MemoriaPrograma Port map(
        dir => salida_pila(9 downto 0),
        dout => Instruccion
    );

    FILEREGISTER : Archivo_Registro Port map(
        wr => microInstruccion(10),
        she => microInstruccion(12),
        dir => microInstruccion(11),
        clk => CLK,
        clr => CLR,
        write_reg => Instruccion(19 downto 16),
        read_reg1 => Instruccion(15 downto 12),
        read_reg2 => SR2,
        shamt => Instruccion(7 downto 4),
        write_data => SWD,
        read_data1 => readData1,
        read_data2 => readData2
    );

   ALUN : Alu_16bits Port map(
        a => SOP1,
        b => SOP2,
        aluop => microInstruccion(7 downto 4),
        s => resALU,
        flags => banderas_alu_salida
    );

    MEMDATA : MemoriaDatos Port map(
        dir => SDMD(9 downto 0),
        data_in => readData2,
        data_out => salida_memoria_datos,
        WD => microInstruccion(2),
        CLK => CLK
    );

    UNITCONTROL : UnidadControl Port map(
        clk => CLK,
        clr => CLR,
        cOperacion => Instruccion(24 downto 20),
        cFuncion => Instruccion(3 downto 0),
        flags => banderas_alu_salida,
        s => microInstruccion,
        NA => NA
    );

    SR <= salida_memoria_datos when (microInstruccion(1) = '0') else resALU;
    SDMP <= Instruccion(15 downto 0) when (microInstruccion(19) = '0') else SR ;
    SR2 <= Instruccion(11 downto 8) when (microInstruccion(15) = '0') else Instruccion(19 downto 16);
    SWD <= Instruccion(15 downto 0) when (microInstruccion(14) = '0') else SR;
    SEXT <= (Instruccion(11) & Instruccion(11) & Instruccion(11) & Instruccion(11) & Instruccion(11 downto
0)) when (microInstruccion(13) = '0') else ("0000" & Instruccion(11 downto 0));
    SOP1 <= readData1 when (microinstruccion(9) = '0') else salida_pila;
    SOP2 <= readData2 when (microinstruccion(8) = '0') else SEXT;
    SDMD <= resALU when (microinstruccion(3) = '0') else Instruccion(15 downto 0);

    LECTURA_PC <= salida_pila;
    LECTURA_INSTRUCCION <= Instruccion;
    LECTURA_READDATA1 <= readData1;
    LECTURA_READDATA2 <= readData2;
    LECTURA_RES_ALU <= resALU;
    LECTURA_BUS_SR <= SR;
    LECTURA_MICROINSTRUCCION <= microInstruccion;
    LECTURA_NA <= NA;

end Behavioral;
```

Código de simulación



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

entity ESCOMIPS_TB is
end ESCOMIPS_TB;

architecture Behavioral of ESCOMIPS_TB is
    component ESCOMips is
        Port (
            CLK, RCLR : IN STD_LOGIC;
            LECTURA_PC : OUT STD_LOGIC_VECTOR(15 downto 0);
            LECTURA_INSTRUCCION : OUT STD_LOGIC_VECTOR(24 downto 0);
            LECTURA_READDATA1, LECTURA_READDATA2 : OUT STD_LOGIC_VECTOR(15 downto 0);
            LECTURA_RES_ALU : OUT STD_LOGIC_VECTOR(15 downto 0);
            LECTURA_BUS_SR : OUT STD_LOGIC_VECTOR(15 downto 0);
            LECTURA_MICROINSTRUCCION : OUT STD_LOGIC_VECTOR(19 downto 0);
            LECTURA_NA : OUT STD_LOGIC
        );
    end component;

    -- Se?ales de transporte
    signal CLK, RCLR : STD_LOGIC;
    signal LECTURA_PC : STD_LOGIC_VECTOR(15 downto 0);
    signal LECTURA_INSTRUCCION : STD_LOGIC_VECTOR(24 downto 0);
    signal LECTURA_READDATA1, LECTURA_READDATA2 : STD_LOGIC_VECTOR(15 downto 0);
    signal LECTURA_RES_ALU : STD_LOGIC_VECTOR(15 downto 0);
    signal LECTURA_BUS_SR : STD_LOGIC_VECTOR(15 downto 0);
    signal LECTURA_MICROINSTRUCCION : STD_LOGIC_VECTOR(19 downto 0);
    signal LECTURA_NA : STD_LOGIC;

    begin

    ESCOMipsMAP : ESCOMips  Port map(
        CLK => CLK,
        RCLR => RCLR,
        LECTURA_PC => LECTURA_PC ,
        LECTURA_INSTRUCCION => LECTURA_INSTRUCCION,
        LECTURA_READDATA1 => LECTURA_READDATA1,
        LECTURA_READDATA2 => LECTURA_READDATA2,
        LECTURA_RES_ALU => LECTURA_RES_ALU,
        LECTURA_BUS_SR => LECTURA_BUS_SR,
        LECTURA_MICROINSTRUCCION => LECTURA_MICROINSTRUCCION,
        LECTURA_NA => LECTURA_NA
    );

    CLOCK : process
        begin
        CLK <= '0';
        wait for 5 ns;
        CLK <= '1';
        wait for 5 ns;
    end process;

    RESET : process
        begin
        RCLR <= '1';
        wait for 20 ns;
        RCLR <= '0';
        wait;
    end process;

end Behavioral;
```

Diagrama RTL



SR2_i
RTL_MUX
FILEREGISTER
Archivo_Registro
SWD_i
RTL_MUX
ALUN
Alu_16bits
LECTURA_INSTRUCCION[24:0]_OBUF_inst
OBUF
LECTURA_READDATA1[15:0]
LECTURA_READDATA2[15:0]
LECTURA_INSTRUCCION[24:0]
LECTURA_BUS_SR[15:0]
SOP1_i
RTL_MUX
SDMP_i
RTL_MUX
SR_i
RTL_MUX
LECTURA_MICROINSTRUCCION[19:0]
STACK
pila
MEMPROG
MemoriaPrograma
SEXT_i
RTL_MUX
SOP2_i
RTL_MUX
UNITCONTROL
UnidadControl
LECTURA_PC[15:0]
LECTURA_NA
LECTURA_RES_ALU[15:0]
CLK
MEMDATA
MemoriaDatos
SDMD_i
RTL_MUX
CLR_reg
RTL_REG
RCLR