Reporte ESCOMips Parte 1

González Pardo Adrian

Junio 2020

1. Código de modulos de hardware

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

package modulosHardware is
    -- Elementos de la ALU 16 Bits
    component sumador is
    Port ( a,b,cin : in STD_LOGIC;
           s,cout : out STD_LOGIC);
    end component;

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

    --Fin elementos de la ALU

    --Archivo de registros
    component demuxG is
    Port ( input : in STD_LOGIC;
           selector : in STD_LOGIC_VECTOR (3 downto 0);
           output : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    component registro is
    Port ( d : in STD_LOGIC_VECTOR (15 downto 0);
           clr,clk,l : in STD_LOGIC;
           q : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    component muxGral is
    Port ( chanel_0,chanel_1,chanel_2,chanel_3,chanel_4,chanel_5,chanel_6,chanel_7,chanel_8
    ,chanel_9,chanel_10,chanel_11,chanel_12,chanel_13,chanel_14,chanel_15 : in
    STD_LOGIC_VECTOR (15 downto 0);
           selectMux : in STD_LOGIC_VECTOR (3 downto 0);
           outMux : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    component barrel is
    Port ( dato : in STD_LOGIC_VECTOR (15 downto 0);
           s : in STD_LOGIC_VECTOR (3 downto 0);
           dir : std_logic;
           res : out STD_LOGIC_VECTOR (15 downto 0));
    end component;

    component mux2 is
    Port ( p,s : in STD_LOGIC_VECTOR (15 downto 0);
           sel : in STD_LOGIC;
           sout : out STD_LOGIC_VECTOR (15 downto 0));
    end component;
```

```
50
51      --Fin de archivo de registros
52
53      --Unidad de control
54      component regControl is
55      Port ( D : in STD_LOGIC_VECTOR (3 downto 0);
56             Q : out STD_LOGIC_VECTOR (3 downto 0);
57             CLR,CLK,L : in STD_LOGIC);
58      end component;
59
60      component condition is
61      Port ( Q : in STD_LOGIC_VECTOR (3 downto 0);
62             EQ,NE,LT,LE,GT,GE : out STD_LOGIC);
63      end component;
64
65      component decInstrucciones is
66      Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
67             TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI : out STD_LOGIC);
68      end component;
69
70      component level is
71      Port ( CLR,CLK : in STD_LOGIC;
72             NA : out STD_LOGIC);
73      end component;
74
75      component mopcode is
76      Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
77             micro : out STD_LOGIC_VECTOR (19 downto 0));
78      end component;
79
80      component mfuncode is
81      Port ( funCode : in STD_LOGIC_VECTOR (3 downto 0);
82             micro : out STD_LOGIC_VECTOR (19 downto 0));
83      end component;
84
85      component unControl is
86      Port ( TIPOR,BEQI,BNEQI,BLTI,BLETI,BGTI,BGETI,NA,EQ,NE,LT,LE,GT,GE : in STD_LOGIC;
87             SM,SDOPC : out STD_LOGIC);
88      end component;
89
90      --Fin Unidad de control
91
92
93
94
95
96
97      --Componentes ESCOMips
98      component aluN is
99      generic(m:integer:=16);
100     Port ( a, b : in STD_LOGIC_VECTOR (m-1 downto 0);
101            aluop : in STD_LOGIC_VECTOR (3 downto 0);
102            res : out STD_LOGIC_VECTOR (m-1 downto 0);
103            n,z,ov,co : out STD_LOGIC
104            );
105     end component;
106
107     --Memoria de datos para almacenamiento
108     component memoriaDatos is
109     generic (
110         m : integer := 16; --tamanio del bus de direcciones
111         n : integer := 16 --tamanio de palabra (dato)
112     );
113     Port(add : in STD_LOGIC_VECTOR (m-1 downto 0);
114          dataIn : in STD_LOGIC_VECTOR (n-1 downto 0);
115          wd, clk : in STD_LOGIC;
116          dataOut : out STD_LOGIC_VECTOR (n-1 downto 0));
```

```
117     end component;
118     --Fin memoria de datos
119     component archRegistros is
120     Port ( writeData : in STD_LOGIC_VECTOR (15 downto 0);
121            writeReg,readReg1,readReg2,shamt : in STD_LOGIC_VECTOR (3 downto 0);
122            WR,CLK,SHE,DIR,clr : in STD_LOGIC;
123            readData1 : inout STD_LOGIC_VECTOR (15 downto 0);
124            readData2 : out STD_LOGIC_VECTOR (15 downto 0));
125     end component;
126     --Memoria de Programa (Contador de programa)
127     component memoriaPrograma is
128     generic(
129         m:integer:= 10;--tamanio del bus de direcciones
130         n:integer:= 25--tamanio de palabra
131     );
132     Port(dir:in STD_LOGIC_VECTOR (m-1 downto 0);
133          inst:out STD_LOGIC_VECTOR (n-1 downto 0));
134     end component;
135     --Fin Memoria de Programa
136     --Pila
137     component pila is
138     generic(
139         N:integer:=16;
140         B:integer:=3
141     );
142     Port(PCin:in STD_LOGIC_VECTOR(N-1 downto 0);
143          DW, UP, WPC, CLK,CLR:in STD_LOGIC;
144          PCout:out STD_LOGIC_VECTOR(N-1 downto 0);
145          SP_out:out STD_LOGIC_VECTOR(B-1 downto 0));
146     end component;
147     --Fin Pila
148     component unidadControl is
149     Port ( funCode,banderas : in STD_LOGIC_VECTOR (3 downto 0);
150            CLK,CLR : in STD_LOGIC;
151            opCode : in STD_LOGIC_VECTOR (4 downto 0);
152            microinstruccion : out STD_LOGIC_VECTOR (19 downto 0));
153     end component;
154 
155     component extendDir is
156     Port ( entrada : in STD_LOGIC_VECTOR (11 downto 0);
157            salida : out STD_LOGIC_VECTOR (15 downto 0));
158     end component;
159 
160     component extendSig is
161     Port ( entrada : in STD_LOGIC_VECTOR (11 downto 0);
162            salida : out STD_LOGIC_VECTOR (15 downto 0));
163     end component;
164 
165     component mux4bits is
166     Port ( entrada,entrada1 : in STD_LOGIC_VECTOR (3 downto 0);
167            flag : in STD_LOGIC;
168            salida : out STD_LOGIC_VECTOR (3 downto 0));
169     end component;
170 
171     component mux16bits is
172     Port ( entrada,entrada1 : in STD_LOGIC_VECTOR(15 downto 0);
173         flag : in STD_LOGIC;
174         salida : out STD_LOGIC_VECTOR(15 downto 0));
175     end component;
176 
177     --Fin ESCOMips
178 
179     --ESCOMips
180 
181     component escomips is
182     Port ( clk,rclr : in STD_LOGIC;
183         PC : out STD_LOGIC_VECTOR(15 downto 0);
```

```
        instruccion : out STD_LOGIC_VECTOR(24 downto 0);
        readData1,readData2,aluOut,busSR : out STD_LOGIC_VECTOR(15 downto 0)
        );
    end component;
    --Fin

end modulosHardware;
```

2. Código de la arquitectura ESCOMips

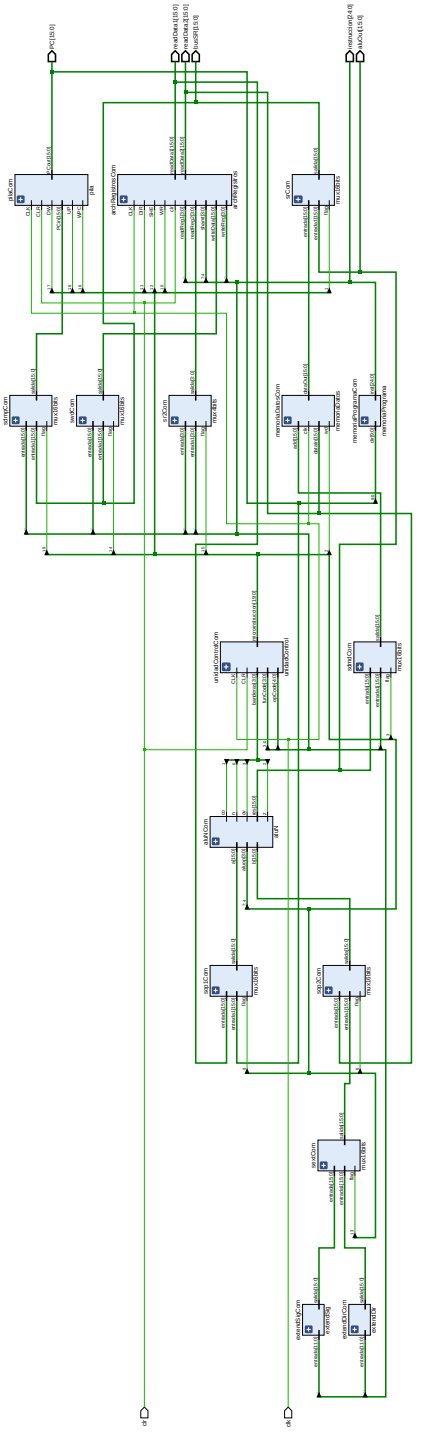
```
library IEEE;
library WORK;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.modulosHardware.pila;
use WORK.modulosHardware.memoriaPrograma;
use WORK.modulosHardware.mux4bits;
use WORK.modulosHardware.mux16bits;
use WORK.modulosHardware.extendDir;
use WORK.modulosHardware.extendSig;
use WORK.modulosHardware.archRegistros;
use WORK.modulosHardware.aluN;
use WORK.modulosHardware.memoriaDatos;
use WORK.modulosHardware.unidadControl;

entity escomips is
    Port ( clk,rclr : in STD_LOGIC;
        PC : out STD_LOGIC_VECTOR(15 downto 0);
        instruccion : out STD_LOGIC_VECTOR(24 downto 0);
        readData1,readData2,aluOut,busSR : out STD_LOGIC_VECTOR(15 downto 0)
        );
end escomips;

architecture Behavioral of escomips is
    signal sr2Out,flagsALU : STD_LOGIC_VECTOR(3 downto 0);
    signal sdmpOut,swdOut,sop1Out,sop2Out,sdmdOut,srOut,sextOut,eSig,eDir :
    STD_LOGIC_VECTOR(15 downto 0);
    signal outPila : STD_LOGIC_VECTOR(15 downto 0);
    signal SDMP,DW,UP,WPC,SR2,SWD,SEXT,SHE,DIR,WR,SOP1,SOP2,SDMD,WD,SR,CLR : STD_LOGIC;
    signal sp : STD_LOGIC_VECTOR(2 downto 0);
    signal outMemoria : STD_LOGIC_VECTOR(24 downto 0);
    signal microinstruccion : STD_LOGIC_VECTOR (19 downto 0);
    signal outReadData1,outReadData2,outALU,outMemDatos : STD_LOGIC_VECTOR(15 downto 0);
begin

    process(SDMP,DW,UP,WPC,SR2,SWD,SEXT,SHE,DIR,WR,SOP1,SOP2,SDMD,WD,SR,microinstruccion)
    begin
        SDMP<=microinstruccion(19);
        UP<=microinstruccion(18);
        DW<=microinstruccion(17);
        WPC<=microinstruccion(16);
        SR2<=microinstruccion(15);
        SWD<=microinstruccion(14);
        SEXT<=microinstruccion(13);
        SHE<=microinstruccion(12);
        DIR<=microinstruccion(11);
        WR<=microinstruccion(10);
        SOP1<=microinstruccion(9);
        SOP2<=microinstruccion(8);
        SDMD<=microinstruccion(3);
        WD<=microinstruccion(2);
        SR<=microinstruccion(1);
    end process;

    pilaCom:pila port map(
        PCin=>sdmpOut,
```

```
        DW=>DW,
        UP=>UP,
        WPC=>WPC,
        CLK=>CLK,
        CLR=>CLR,
        PCout=>outPila,
        SP_out=>sp
    );

    memoriaProgramaCom:memoriaPrograma port map(
        dir=>outPila(9 downto 0),
        inst=>outMemoria
    );

    sr2Com:mux4bits port map(
        entrada=>outMemoria(11 downto 8),
        entrada1=>outMemoria(19 downto 16),
        flag=>SR2,
        salida=>sr2Out
    );

    swdCom:mux16bits port map(
        entrada=>outMemoria(15 downto 0),
        entrada1=>srOut,
        flag=>SWD,
        salida=>swdOut
    );

    sdmpCom:mux16bits port map(
        entrada=>outMemoria(15 downto 0),
        entrada1=>srOut,
        flag=>SDMP,
        salida=>sdmpOut
    );

    extendDirCom:extendDir port map(
        entrada=>outMemoria(11 downto 0),
        salida=>eDir
    );

    extendSigCom:extendSig port map(
        entrada=>outMemoria(11 downto 0),
        salida=>eSig
    );

    sextCom:mux16bits port map(
        entrada=>eSig,
        entrada1=>eDir,
        flag=>SEXT,
        salida=>sextOut
    );

    archRegistrosCom:archRegistros port map(
        writeData=>swdOut,
        writeReg=>outMemoria(19 downto 16),
        readReg1=>outMemoria(15 downto 12),
        readReg2=>sr2Out,
        shamt=>outMemoria(7 downto 4),
        WR=>WR,
        CLK=>CLK,
        SHE=>SHE,
        DIR=>DIR,
        clr=>CLR,
        readData1=>outReadData1,
        readData2=>outReadData2
    );

```

```
    sop1Com:mux16bits port map(
        entrada=>outReadData1,
        entrada1=>outPila,
        flag=>SOP1,
        salida=>sop1Out
    );

    sop2Com:mux16bits port map(
        entrada=>outReadData2,
        entrada1=>sextOut,
        flag=>SOP2,
        salida=>sop2Out
    );

    aluNCom:aluN port map(
        a=>sop1Out,
        b=>sop2Out,
        aluop=>microinstruccion(7 downto 4),
        res=>outALU,
        n=>flagsALU(0),
        z=>flagsALU(2),
        ov=>flagsALU(3),
        co=>flagsALU(1)
    );

    sdmdCom:mux16bits port map(
        entrada=>outALU,
        entrada1=>outMemoria(15 downto 0),
        flag=>SDMD,
        salida=>sdmdOut
    );

    memoriaDatosCom:memoriaDatos port map(
        add=>sdmdOut,
        dataIn=>outReadData2,
        WD=>WD,
        CLK=>CLK,
        dataOut=>outMemDatos
    );

    srCom:mux16bits port map(
        entrada=>outMemDatos,
        entrada1=>outALU,
        flag=>SR,
        salida=>srOut
    );

    unidadControlCom:unidadControl port map(
        funCode=>outMemoria(3 downto 0),
        banderas=>flagsALU,
        CLK=>CLK,
        CLR=>CLR,
        opCode=>outMemoria(24 downto 20),
        microinstruccion=>microinstruccion
    );

    process(outMemoria,outReadData1,outReadData2,outAlu,srOut)
    begin
        PC<=outPila;
        instruccion<=outMemoria;
        readData1<=outReadData1;
        readData2<=outReadData2;
        aluOut<=outAlu;
        busSR<=srOut;
    end process;

    process(CLK)
```

```
189     begin
190         if falling_edge(CLK) then
191             CLR<=RCLR;
192         end if;
193     end process;
194 
195 
196 end Behavioral;
```

3. RTL de la arquitectura



4. Código del primer programa en Memoria de programa

```
type banco is array (0 to (2**m)-1) of std_logic_vector(n-1 downto 0);
constant aux:banco:=(
  -- Primer programa proyecto
  LI & R0 & x"0001" ,                  -- 0 LI R0, #1
  LI & R1 & x"0007" ,                  -- 1 LI R1, #7
  TYPER & R1 & R1 & R0 & SU & ADD ,    -- 2 ADD R1,R1,R0
  SWI & R1 & x"0005" ,                 -- 3 SWI R1,#5
  B& SU & x"0002" ,                    -- 4 B 2
  others=>(others=>'0')
  );
```

5. Simulación imagenes

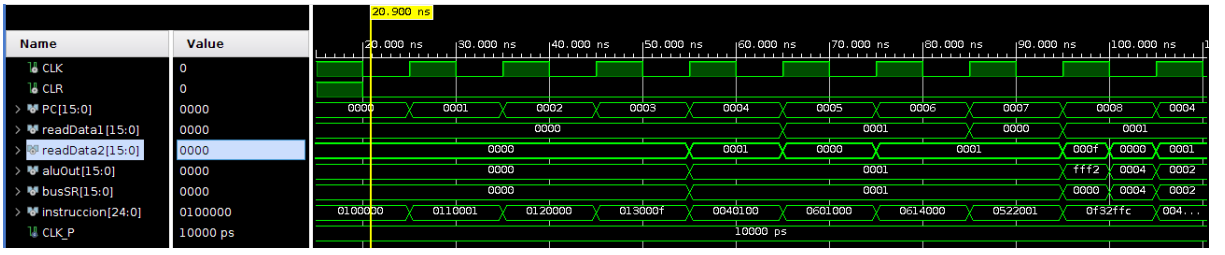
6. Tabla del programa 1

Bus	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
PC	0000	0001	0002	0003	0004	0002	0003	0004	0002	0003	0004
Instrucción	0100001	0110007	0011000	0310005	1300002	0011000	0310005	1300002	0011000	0310005	1300002
Read Data 1	0000	0001	0007	0001	0001	0008	0001	0001	0009	0001	0001
Read Data 2	0000	0001	0001	0008	0001	0001	0009	0001	0001	000a	0001
Res ALU	0000	0001	0008	0000	0001	0009	0001	0001	000a	0000	0001
Bus SR	0000	0000	0008	0000	0000	0009	0008	0000	000a	0009	0000

7. Código del segundo programa en Memoria de programa

```
type banco is array (0 to (2**m)-1) of std_logic_vector(n-1 downto 0);

constant aux:banco:=(
    -- Segundo programa proyecto Fibonacci
    LI & R0 & x"0000" ,                 -- 0 LI R0, #0
    LI & R1 & x"0001" ,                 -- 1 LI R1, #1
    LI & R2 & x"0000" ,                 -- 2 LI R2, #0
    LI & R3 & x"000F" ,                 -- 3 LI R3, #15
    TYPER & R4 & R0 & R1 & SU & ADD ,   -- 4 ADD R1,R1,R0
    SUBI & R0 & R1 & x"000" ,           -- 5 SUBI R0,R1,#0
    SUBI & R1 & R4 & x"000" ,           -- 6 SUBI R1,R4,#0
    ADDI & R2 & R2 & x"001" ,           -- 7 ADDI R2,R2,#1
    BLTI & R3 & R2 & x"FFC" ,           -- 8 BLTI R3,R2,-4
    NOP & SU & SU & SU & SU & SU ,      -- 9 NOP
    B & SU & x"0009" ,                  --10 B 9
    others=>(others=>'0')
);
```

8. Simulación imagenes



9. Tabla del programa 2

Bus	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
PC	0000	0001	0002	0003	0004	0005	0006	0007	0008	0004	0005
Instrucción	0100000	0110001	0120000	013000f	0040100	0601000	0614000	0522001	0f32ffc	0040100	0601000
Read Data 1	0000	0000	0000	0000	0000	0001	0001	0000	0001	0001	0001
Read Data 2	0000	0000	0000	0000	0001	0000	0001	0001	000F&0000	0001	0001
Res ALU	0000	0000	0000	0000	0001	0001	0001	0001	FFF2&0004	0002	0001
Bus SR	0000	0000	0000	0000	0001	0001	0001	0001	0000&0004	0002	0001