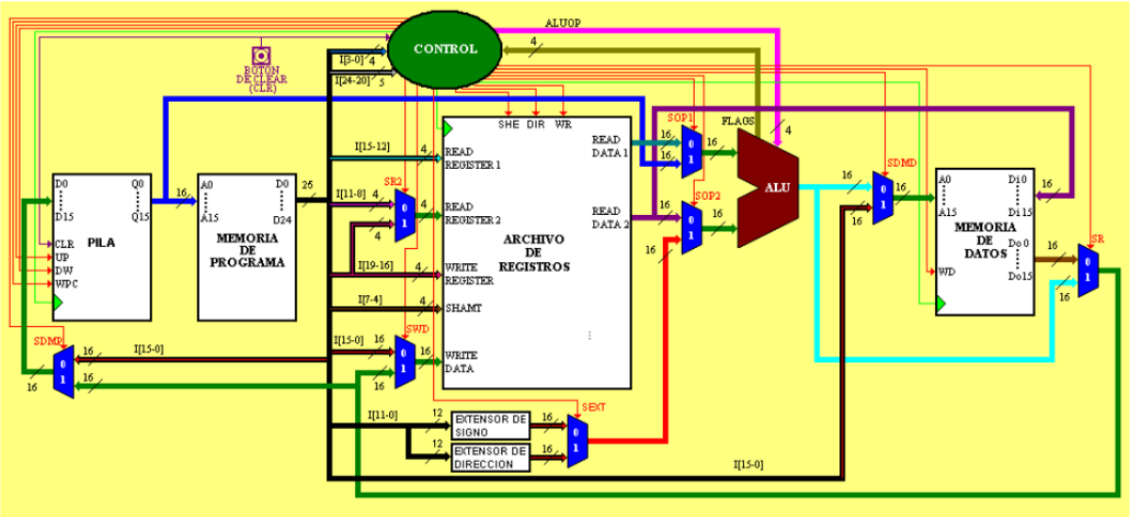
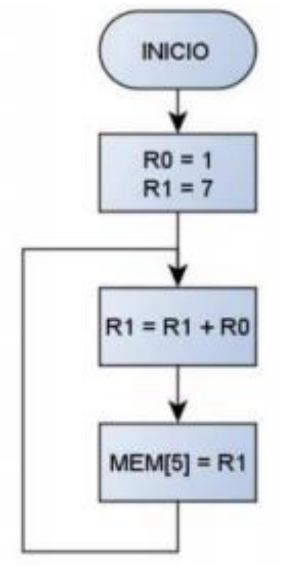
Entrega de Proyecto
ESCOMips

1. Implemente el procesador ESCOMips con las siguientes configuraciones
   a. Debe estar implementado por componentes
   b. Tamaño de los buses de la ALU: 16 bits
   c. Organización de la memoria de datos: 1024 x 16
   d. Organización de la memoria de programa: 1024 x 25



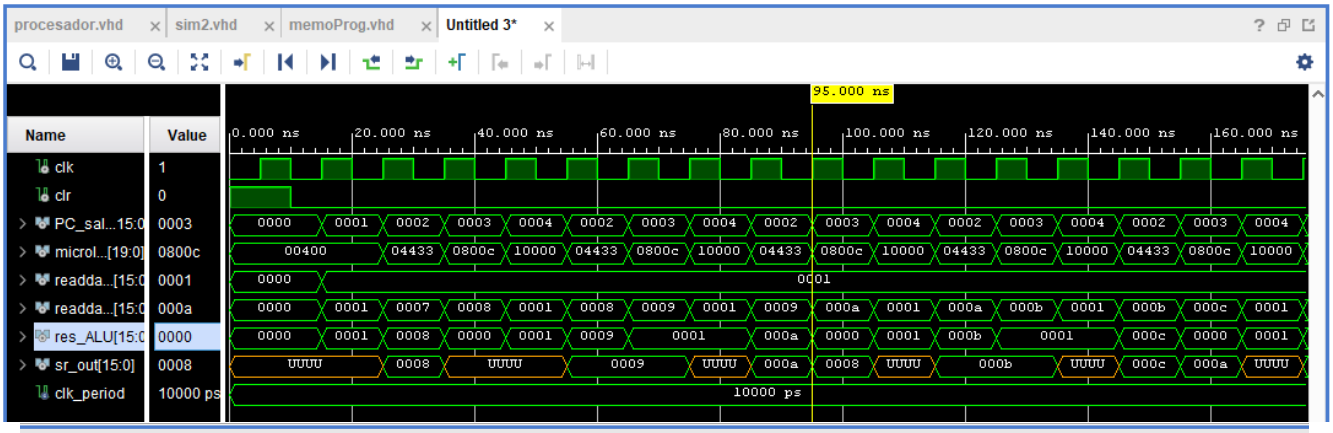
2. Cargue en la memoria de programa el código del diagrama de flujo que se muestra a continuación. Las instrucciones deben cargarse en la memoria utilizando las constantes definidas en la práctica de Memoria de programa. No se aceptará que las instrucciones sean cargadas directamente en código binario.

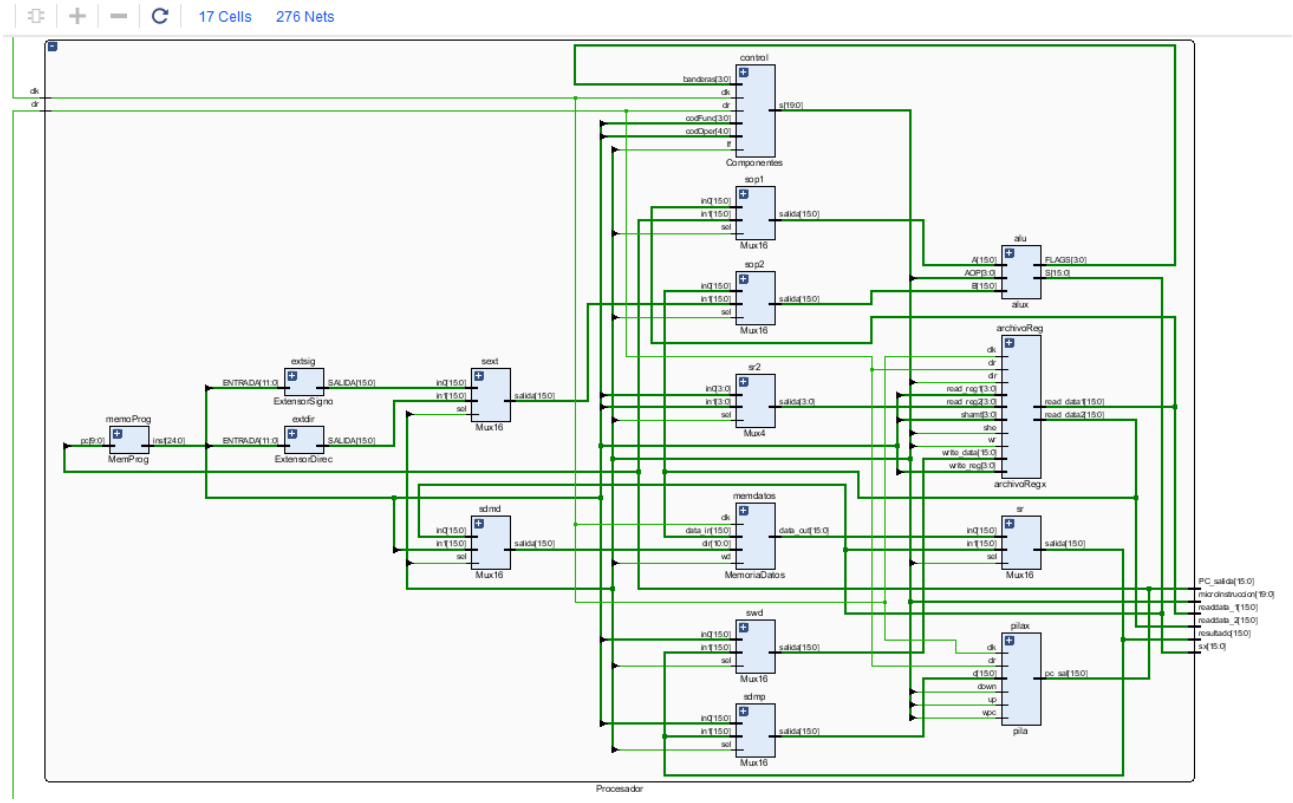



```
LI & R0 & X"0001",
LI & R1 & X"0007",
TIPO_R & R1 & R0 & R1 & SU & ADD,
SWI & R1 & X"0005",
B & SU & X"0002"
```

3. Realice la simulación del procesador
4. Compruebe el correcto funcionamiento del procesador y llene la siguiente tabla

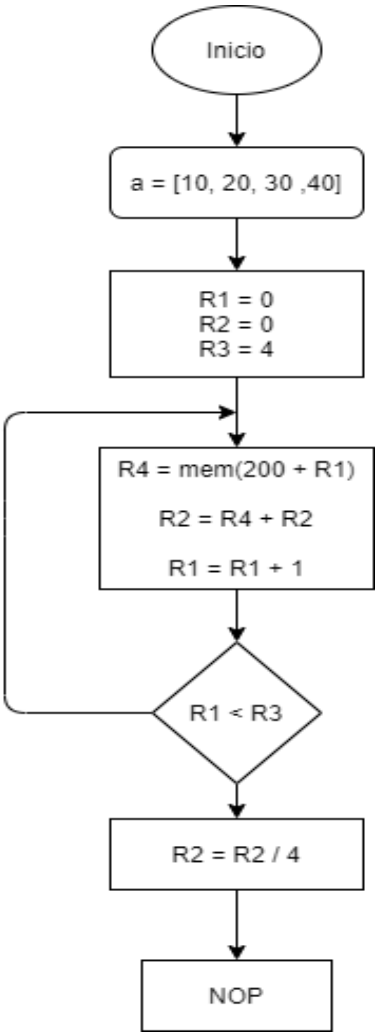
Bus	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
PC	0001	0002	0003	0004	0002	0003	0004	0002	0003	0004	0002
Instrucción	00400	04433	0800C	10000	04433	0800C	10000	04433	0800C	10000	04433
ReadData1	0001										
ReadData2	0001	0007	0008	0001	0008	0009	0001	0009	000A	0001	0001
ResAlu	0001	0008	UUUU	0001	0009	0009	0001	000A	0000	UUUU	0001
BusSR	UUUU	0008	UUUU	UUUU	0009	0009	UUUU	000A	0008	UUUU	UUUU





5. Cargue en la memoria de programa el código correspondiente al programa asignado para cada uno de ustedes.

Realizar un programa que permita calcular el promedio de un arreglo de 4 números.



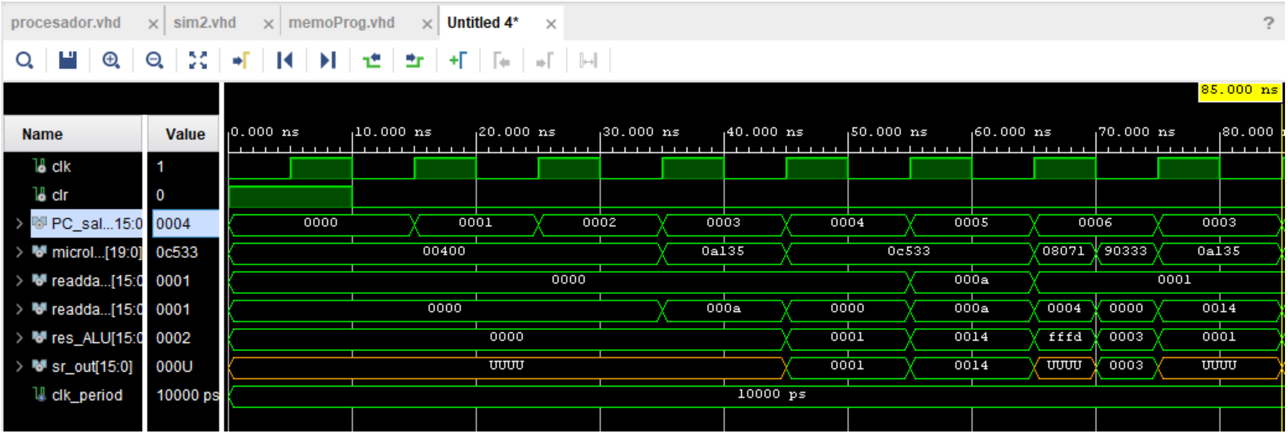
En primer lugar, requerimos crear el arreglo, por lo cual creamos una subrutina que se encarga de llenarlo.

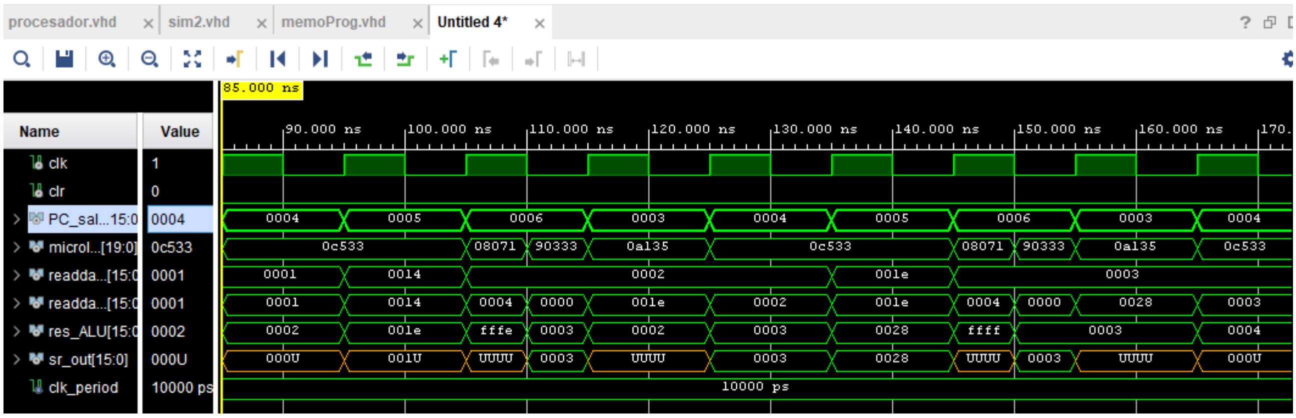
```
LI & R0 & X"0001",
LI & R1 & X"000A",
LI & R2 & X"0004",
SW & R1 & R0 & X"000"
ADDI & R0 & R0 & X"001"
ADDI & R1 & R1 & X"00A"
BLTI & R2 & R0 & X"FFD"
RET & SU & SU & SU & SU & SU
```

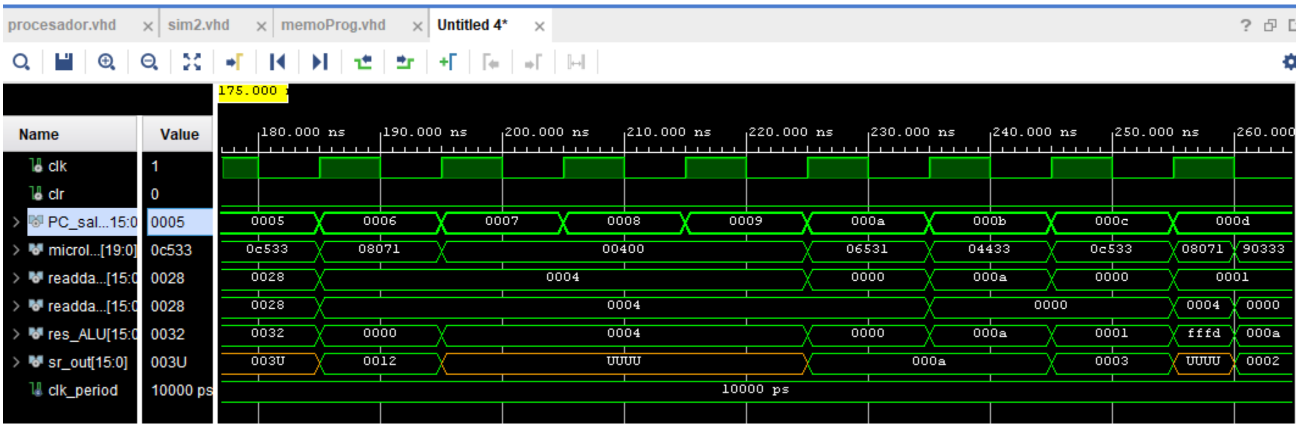
```
LI & R1 & X"0000",                    --7
LI & R2 & X"0000",                    --8
LI & R3 & X"0004",                    --9
LW & R4 & R1 & X"000",                --10
TIPO_R & R2 & R4 & R2 & SU & ADD,     --11
ADDI & R1 & R1 & X"001",              --12
BLTI & R3 & R1 & X"FFD",              --13
TIPO_R & R2 & R2 & SU & X"2" & SRLR,      --14
SWI & R2 & X"0900",                       --15
LWI & R2 & X"0900",                        --16
NOP & SU & SU & SU & SU & SU,               --17
B & SU & X"0011",
```

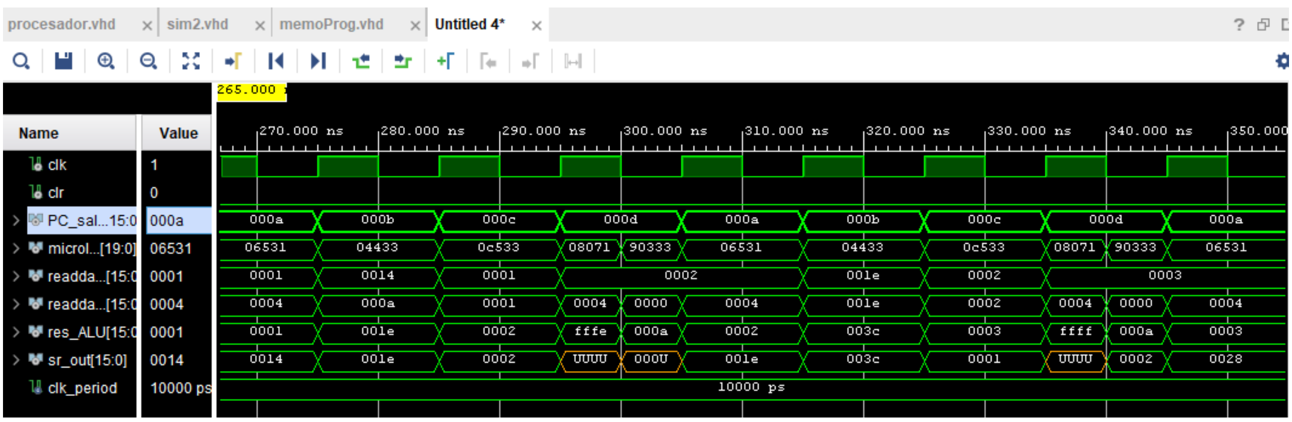
Tabla de llenado de arreglo

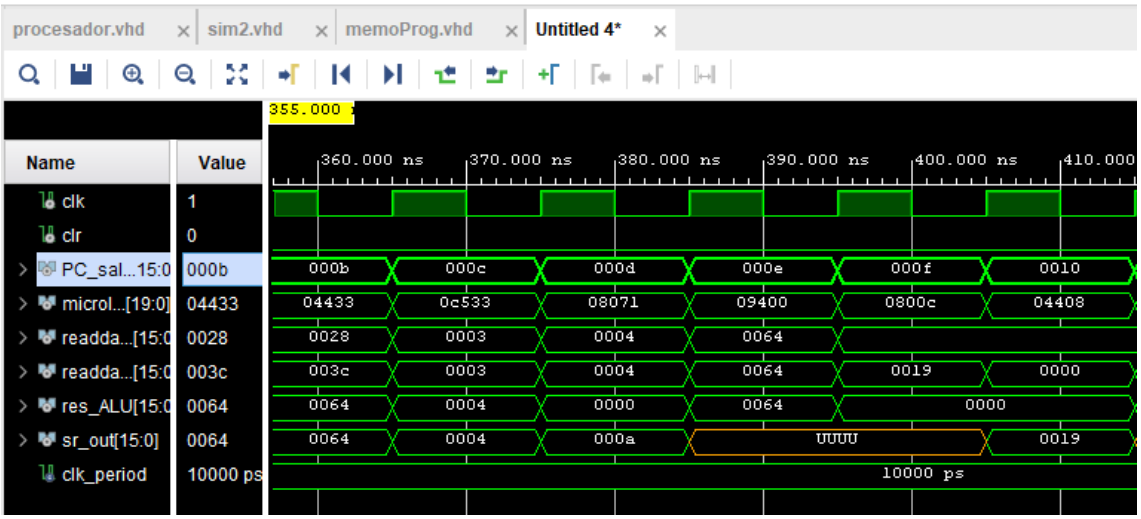
Bus	T1	T2	T3	T4	T5	T6		T7	T8	T9	T10	
PC	0001	0002	0003	0004	0005	0006		0003	0004	0005	0006	
Instrucción	00400	00400	0A135	0C533	0C533	08071	90333	0A135	0C533	0C533	08071	90333
ReadData1	0000	0000	0000	0000	000A	0001		0001	0001	0014	0002	
ReadData2	0000	0000	000A	0000	000A	0004	0000	0014	0001	0014	0004	0000
ResAlu	0000	0000	0000	0001	0014	FFFD	0003	0001	0002	001E	FFFE	0003
BusSR	UUUU	UUUU	UUUU	0001	00014	UUUU	0003	UUUU	UUUU	001E	UUUU	0003
Bus	**T11**	**T12**	**T13**	**T14**		**T15**	**T16**	**T17**	**T18**	**T19**	**T20**	
PC	0003	0004	0005	0006		0003	0004	0005	0006	0005	000A	000B
Instrucción	0A135	0C533	0C533	08071	90333	0A135	0C533	0C533	08071	0C533	06531	90333
ReadData1	0002	0002	001E	0003				0028	0004	0000	0000	000A
ReadData2	001E	0002	001E	0004	0000	0028	0003	0028	0004	0004	0004	0000
ResAlu	0002	0003	0028	FFFF	0003	0003	0004	0032	0000	0000	000A	000A
BusSR	UUUU	0003	0028	UUUU	0003	UUUU	UUUU	0032	0012	000A	0000A	


procesador.vhd
sim2.vhd
memoProg.vhd
Untitled 4*
85.000 ns
Name
Value
clk 1
clr 0
PC_sal...15:0 0004
microI...[19:0] 0c533
readda...[15:0 0001
readda...[15:0 0001
res_ALU[15:0 0002
sr_out[15:0] 000U
clk_period 10000 ps
0.000 ns
10.000 ns
20.000 ns
30.000 ns
40.000 ns
50.000 ns
60.000 ns
70.000 ns
80.000


procesador.vhd
sim2.vhd
memoProg.vhd
Untitled 4*
85.000 ns
Name
Value
clk 1
clr 0
PC_sal...15:0 0004
microI...[19:0] 0c533
readda...[15:0 0001
readda...[15:0 0001
res_ALU[15:0 0002
sr_out[15:0] 000U
clk_period 10000 ps
90.000 ns
100.000 ns
110.000 ns
120.000 ns
130.000 ns
140.000 ns
150.000 ns
160.000 ns


procesador.vhd
sim2.vhd
memoProg.vhd
Untitled 4*
175.000
Name
Value
clk 1
clr 0
PC_sal...15:0 0005
microI...[19:0] 0c533
readda...[15:0 0028
readda...[15:0 0028
res_ALU[15:0 0032
sr_out[15:0] 003U
clk_period 10000 ps
180.000 ns
190.000 ns
200.000 ns
210.000 ns
220.000 ns
230.000 ns
240.000 ns
250.000 ns


procesador.vhd
sim2.vhd
memoProg.vhd
Untitled 4*
265.000
Name
Value
clk 1
clr 0
PC_sal...15:0 000a
microI...[19:0] 06531
readda...[15:0 0001
readda...[15:0 0004
res_ALU[15:0 0001
sr_out[15:0] 0014
clk_period 10000 ps
270.000 ns
280.000 ns
290.000 ns
300.000 ns
310.000 ns
320.000 ns
330.000 ns
340.000 ns



Código de implementación del procesador

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library work;
use work.main.all;

entity Procesador is
    Port ( clk, clr : in  STD_LOGIC;
           microInstruccion : out  STD_LOGIC_VECTOR (19 downto 0);
           PC_salida : out  STD_LOGIC_VECTOR (15 downto 0);
              readdata_1 : out  STD_LOGIC_VECTOR (15 downto 0);
              readdata_2 : out  STD_LOGIC_VECTOR (15 downto 0);
           sr_out : out  STD_LOGIC_VECTOR (15 downto 0);
              res_ALU : out  STD_LOGIC_VECTOR (15 downto 0));
end Procesador;

architecture Behavioral of Procesador is

    signal instruccion : std_logic_vector(24 downto 0);
    signal pc : std_logic_vector(15 downto 0);
    signal banderas : std_logic_vector(3 downto 0);
    signal microinstr : std_logic_vector(19 downto 0);
    signal sdmpOut : std_logic_vector(15 downto 0);
    signal sr2out : std_logic_vector(3 downto 0);
    signal swdout : std_logic_vector(15 downto 0);
    signal readata1 : std_logic_vector(15 downto 0);
    signal readata2 : std_logic_vector(15 downto 0);
    signal srout : std_logic_vector(15 downto 0);
    signal extsigout : std_logic_vector(15 downto 0);
    signal extdirout : std_logic_vector(15 downto 0);
```

```
    signal sextout : std_logic_vector(15 downto 0);
    signal sop1out : std_logic_vector(15 downto 0);
    signal sop2out : std_logic_vector(15 downto 0);
    signal aluout : std_logic_vector(15 downto 0);
    signal sdmdout : std_logic_vector(15 downto 0);
    signal memdataout : std_logic_vector(15 downto 0);
    signal mysr : std_logic;

begin
    control : Componentes port map (instruccion(3 downto 0), instruccion(
24 downto 20), banderas, clk, clr, microinstr(0), microinstr );
    memoProg : MemProg port map (pc(9 downto 0), instruccion);
    pilax : pila port map(sdmpOut, microinstr(19), microinstr(17), microi
nstr(16), clk, clr, pc);
    archivoReg : archivoRegx port map (clk, clr, microinstr(10), microins
tr(12), microinstr(11), instruccion(19 downto 16), instruccion(15 downto
12), sr2out, instruccion(7 downto 4), swdout, readata1, readata2);
    sr2 : Mux4 port map(instruccion(11 downto 8), instruccion(19 downto 1
6), sr2out, microinstr(15));
    swd : Mux16 port map(instruccion(15 downto 0), srout, swdout, microin
str(14));
    sdmp : Mux16 port map(instruccion(15 downto 0), srout, sdmpOut, micro
instr(19));
    extsig : ExtensorSigno port map(instruccion(11 downto 0), extsigout);
    extdir : ExtensorDirec port map(instruccion(11 downto 0), extdirout);
    sext : Mux16 port map(extsigout, extdirout, sextout, microinstr(13));
    sop1 : Mux16 port map(readata1, pc, sop1out, microinstr(9));
    sop2 : Mux16 port map(readata2, sextout, sop2out, microinstr(8));
    alu : alux port map(sop1out, sop2out, microinstr(7 downto 4), bandera
s, aluout);
    sdmd : Mux16 port map(aluout, instruccion(15 downto 0), sdmdout, micr
oinstr(3));
    memdatos : MemoriaDatos port map(sdmdout(10 downto 0), readata2, micr
oinstr(2), clk, memdataout);
    sr : Mux16 port map(memdataout, aluout, srout, microinstr(1));

    res_ALU <= aluout;
    microInstruccion <= microinstr;
    PC_salida <= pc;
    sr_out <= srout;
    readdata_1 <= readata1;
    readdata_2 <= readata2;
    mysr <= microinstr(1);

end Behavioral;
```