Instituto Politécnico Nacional

Escuela Superior de Cómputo

Arquitectura de Computadoras

Practica 11: Pila Hardware 2

Nombre: *Sampayo Hernández Mauro*

Grupo: *3CV8*

Profesor: *Nayeli Vega García*

Fecha de entrega: *18 de abril del 2020*

Código de Implementación:

- **Pila**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;--Datos con signos y sin signo, y
operaciones aritmeticas
use IEEE.STD_LOGIC_unsigned.ALL;--Realizar operaciones sin signo para los
ST_LOGIC_VECTOR

entity Pila is
    generic ( N: integer :=3;
              M: integer :=16);
    Port ( PC_in : in  STD_LOGIC_VECTOR (M-1 downto 0);
           PC_out : out  STD_LOGIC_VECTOR (M-1 downto 0);
           clk, clr, UP, DW, WPC : in  STD_LOGIC;
           SP: out std_logic_vector(N-1 downto 0));
end Pila;

architecture Behavioral of Pila is

type banco is array (0 to (2**N)-1) of std_logic_vector(M-1 downto 0);
signal aux: banco;
signal SP1: integer range 0 to (2**N)-1;

begin
    process(clk, clr)
        variable SPout : integer range 0 to (2**N)-1;
    begin
        if(clr = '1')then
            SPout := 0;
            aux <= (others => (others => '0'));
        elsif(clk'event and clk = '1')then
            if(WPC = '0' and UP = '0' and DW = '0')then--incremento
                aux(SPout) <= aux(SPout)+1;
            elsif(WPC = '1' and UP = '1' and DW = '0')then--CALL
                SPout := SPout + 1;
                aux(SPout) <= PC_in;
            elsif(WPC = '1' and UP = '0' and DW = '0')then--JUMP
                aux(SPout) <= PC_in;
            elsif(WPC = '0' and UP = '0' and DW = '1')then--RET
                SPout := SPout - 1;
                aux(SPout) <= aux(SPout)+1;
            end if;
        end if;
        SP1 <= SPout;
    end process;

    SP <= conv_std_logic_vector(SP1, 3);
    PC_out <= aux(SP1);

end Behavioral;
```

- **MemoriaPrograma**

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

entity MemoriaPrograma is
    generic ( d : integer := 25;
              a : integer := 16);
    Port (PC : in STD_LOGIC_VECTOR (a-1 downto 0);
          Inst : out STD_LOGIC_VECTOR (d-1 downto 0));
end MemoriaPrograma;

architecture Behavioral of MemoriaPrograma is

--INSTRUCCIONES
--Tipo I
constant LI : std_logic_vector(4 downto 0) := "00001";
constant LWI : std_logic_vector(4 downto 0) := "00010";
constant LW : std_logic_vector(4 downto 0) := "10111";
constant SWI : std_logic_vector(4 downto 0) := "00011";
constant SW : std_logic_vector(4 downto 0) := "00100";
constant ADDI : std_logic_vector(4 downto 0):= "00101";
constant SUBI : std_logic_vector(4 downto 0):= "00110";
constant ANDI: std_logic_vector(4 downto 0) := "00111";
constant ORI : std_logic_vector(4 downto 0) := "01000";
constant XORI : std_logic_vector(4 downto 0) := "01001";
constant NANDI : std_logic_vector(4 downto 0):= "01010";
constant NORI: std_logic_vector(4 downto 0) := "01011";
constant XNORI : std_logic_vector(4 downto 0):= "01100";
constant BEQI : std_logic_vector(4 downto 0):= "01101";
constant BNEI : std_logic_vector(4 downto 0):= "01110";
constant BLTI : std_logic_vector(4 downto 0):= "01111";
constant BLETI : std_logic_vector(4 downto 0):= "10000";
constant BGTI : std_logic_vector(4 downto 0):= "10001";
constant BGETI : std_logic_vector(4 downto 0):= "10010";

--Tipo R
constant TR : std_logic_vector(4 downto 0) := "00000";--Operaci�ipo R
constant ADD : std_logic_vector(3 downto 0) := "0000";
constant SUB : std_logic_vector(3 downto 0) := "0001";
constant OpAND : std_logic_vector(3 downto 0) := "0010";
constant OpOR : std_logic_vector(3 downto 0) := "0011";
constant OpXOR : std_logic_vector(3 downto 0) := "0100";
constant OpNAND: std_logic_vector(3 downto 0) := "0101";
constant OpNOR : std_logic_vector(3 downto 0) := "0110";
constant OpXNOR : std_logic_vector(3 downto 0) := "0111";
constant OpNOT : std_logic_vector(3 downto 0) := "1000";
constant OpSLL : std_logic_vector(3 downto 0) := "1001";
constant OpSRL : std_logic_vector(3 downto 0) := "1010";

--Tipo J
constant B: std_logic_vector(4 downto 0):= "10011";
constant CALL : std_logic_vector(4 downto 0):= "10100";

--Otros
```

```vhdl
constant RET : std_logic_vector(4 downto 0):= "10101";
constant NOP : std_logic_vector(4 downto 0):= "10110";

--Sin Uso
constant SU : std_logic_vector(3 downto 0) := "0000";--Sin Uso

--REGISTROS
constant R0 : std_logic_vector(3 downto 0) := "0000";
constant R1 : std_logic_vector(3 downto 0) := "0001";
constant R2 : std_logic_vector(3 downto 0) := "0010";
constant R3 : std_logic_vector(3 downto 0) := "0011";
constant R4 : std_logic_vector(3 downto 0) := "0100";
constant R5 : std_logic_vector(3 downto 0) := "0101";
constant R6 : std_logic_vector(3 downto 0) := "0110";
constant R7 : std_logic_vector(3 downto 0) := "0111";
constant R8 : std_logic_vector(3 downto 0) := "1000";
constant R9 : std_logic_vector(3 downto 0) := "1001";
constant R10 : std_logic_vector(3 downto 0) := "1010";
constant R11 : std_logic_vector(3 downto 0) := "1011";
constant R12 : std_logic_vector(3 downto 0) := "1100";
constant R13 : std_logic_vector(3 downto 0) := "1101";
constant R14 : std_logic_vector(3 downto 0) := "1110";
constant R15 : std_logic_vector(3 downto 0) := "1111";

--COMANDOS :0
type banco is array (0 to (2**10)-1) of std_logic_vector(d-1 downto 0);
constant memProg : banco := (
LI & R6 & x"0057", --1 LI R6, #87
LI & R8 & x"005a", --2 LI R8, #90
TR & R8 & R2 & R3 & SU & ADD, --3 ADD R8, R2, R3
TR & R1 & R2 & R3 & SU & SUB, --4 SUB R1, R2, R3
CALL & SU & x"0009", --5 CALL 0x09
LI & R6 & x"0057", --6 LI R6, #87
LI & R8 & x"005a", --7 LI R8, #90
CALL & SU & x"000d", --8 CALL 13
TR & R8 & R2 & R3 & SU & ADD, --9 ADD R8, R2, R3
TR & R1 & R2 & R3 & SU & SUB, --10 SUB R1, R2, R3
LI & R6 & x"0057", --11 LI R6, #87
RET & SU & SU & SU & SU & SU, --12 RET
TR & R1 & R2 & R3 & SU & SUB, --13 SUB R1, R2, R3
LI & R6 & x"0057", --14 LI R6, #87
RET & SU & SU & SU & SU & SU, --15 RET
B & SU & x"0012", --16 B 18
NOP & SU & SU & SU & SU & SU, --17 NOP
NOP & SU & SU & SU & SU & SU,--18 NOP
B & SU & x"0011", --19 B 17
others => (others => '0'));

begin
    Inst <= memProg(conv_integer(PC));

end Behavioral;
```

- **Pila_MemoriaPrograma**

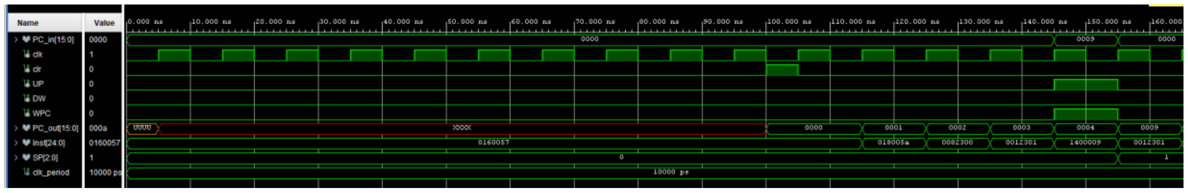
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Pila_MemoriaPrograma is
    generic ( d : integer := 25;
              a : integer := 16;
              s : integer := 3);
    Port( PC_in : in  STD_LOGIC_VECTOR (a-1 downto 0);
          clk, clr, UP, DW, WPC : in  STD_LOGIC;
          PC_out : out  STD_LOGIC_VECTOR (a-1 downto 0);
          Inst : out  STD_LOGIC_VECTOR (d-1 downto 0);
          SP : out STD_LOGIC_VECTOR(s-1 downto 0));
end Pila_MemoriaPrograma;

architecture Behavioral of Pila_MemoriaPrograma is

component Pila is
    Port ( PC_in : in  STD_LOGIC_VECTOR (a-1 downto 0);
           PC_out : out  STD_LOGIC_VECTOR (a-1 downto 0);
           clk, clr, UP, DW, WPC : in  STD_LOGIC;
           SP: out std_logic_vector(S-1 downto 0));
end component;

component MemoriaPrograma is
    Port (PC : in STD_LOGIC_VECTOR (a-1 downto 0);
          Inst : out STD_LOGIC_VECTOR (d-1 downto 0));
end component;

signal PC: STD_LOGIC_VECTOR (a-1 downto 0);
signal SP_out: STD_LOGIC_VECTOR (a-1 downto 0);

begin

    Stack: pila
    Port map( PC_in => PC_in,
              clk => clk,
              clr => clr,
              UP => UP,
              DW => DW,
              WPC => WPC,
              PC_out => PC,
              SP => SP);

    Memory: MemoriaPrograma
    Port map( PC => PC,--el segundo PC es la se:v
              Inst => Inst);

    PC_out <= PC;

end Behavioral;
```

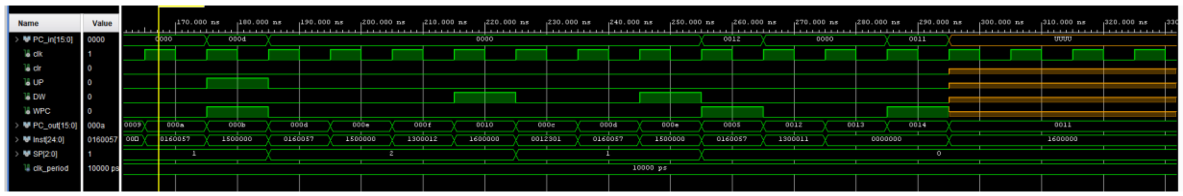
Código de Simulación:

```
library IEEE;
LIBRARY STD;
USE STD.TEXTIO.ALL;
USE IEEE.STD_LOGIC_TEXTIO.ALL;--PERMITE USAR STD_LOGIC
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;

entity Pila_MemoriaPrograma_tb is
end Pila_MemoriaPrograma_tb;

architecture Behavioral of Pila_MemoriaPrograma_tb is

component Pila_MemoriaPrograma is
    Port( PC_in : in  STD_LOGIC_VECTOR (15 downto 0);
          clk, clr, UP, DW, WPC : in  STD_LOGIC;
          PC_out : out  STD_LOGIC_VECTOR (15 downto 0);
          Inst : out  STD_LOGIC_VECTOR (24 downto 0);
          SP : out STD_LOGIC_VECTOR(2 downto 0));
end component;

--Inputs
signal PC_in : STD_LOGIC_VECTOR (15 downto 0) := (others => '0');
signal clk : STD_LOGIC := '0';
signal clr : STD_LOGIC := '0';
signal UP : STD_LOGIC := '0';
signal DW : STD_LOGIC := '0';
signal WPC : STD_LOGIC := '0';

--Outputs
signal PC_out : STD_LOGIC_VECTOR (15 downto 0);
signal Inst : STD_LOGIC_VECTOR (24 downto 0);
signal SP : STD_LOGIC_VECTOR(2 downto 0);

-- Clock period definitions
constant clk_period : time := 10 ns;

begin

    -- Instantiate the Unit Under Test (UUT)
    uut: Pila_MemoriaPrograma PORT MAP (
        PC_in => PC_in,
        clk => clk,
        clr => clr,
        UP => UP,
        DW => DW,
        WPC => WPC,
        Inst => Inst,
        PC_out => PC_out,
        SP => SP
        );

    -- Clock process definitions
    clk_process :process
    begin
```

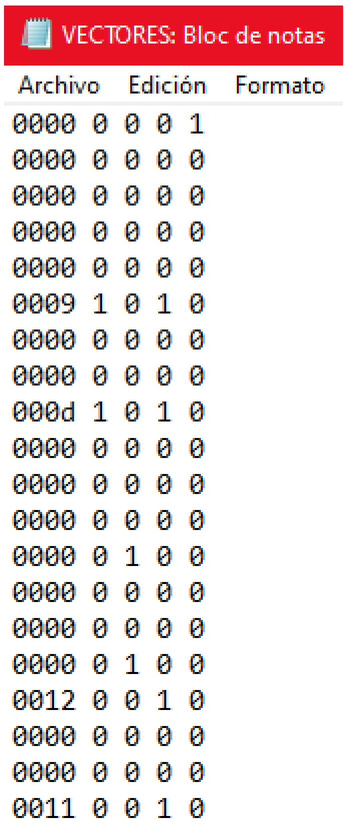
```vhdl
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
        wait for clk_period/2;
    end process;

    -- Stimulus process
    stim_proc: process
    file ARCH_RES : TEXT;--Archivo de resultados
    variable LINEA_RES : line;--linea de resultado
    file ARCH_VEC : TEXT;--Archivo de vectores
    variable LINEA_VEC : line;--Linea de vectores

    --Variables
    variable v_PC_in: STD_LOGIC_VECTOR(15 DOWNTO 0);
    variable v_SP : STD_LOGIC_VECTOR (2 downto 0);
    variable v_PC_out : std_logic_vector(15 downto 0);
    variable v_OP_CODE: STD_LOGIC_VECTOR(4 DOWNTO 0);
    variable v_Rd, v_Rt, v_Rs, v_shamt, v_FUNC_CODE: STD_LOGIC_VECTOR(3
DOWNTO 0);
    variable v_clr, v_UP, v_DW, v_WPC: STD_LOGIC;
    --Cadena
    variable CADENA : STRING(1 TO 6);

    begin
        file_open(ARCH_VEC, "VECTORES.txt", READ_MODE);
        file_open(ARCH_RES, "RESULTADO.txt", WRITE_MODE);
        --Impresi□e Cadenas
        CADENA := "    SP";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := "    PC";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+2);
        CADENA := "OPCODE";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := "    Rd";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := "    Rt";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := "    Rs";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := " Shamt";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        CADENA := "F_CODE";
        write(LINEA_RES, CADENA, right, CADENA'LENGTH+1);
        writeline(ARCH_RES,LINEA_RES);--Escribe la linea en el archivo

        --Impresion de Resultados
        wait for 100 ns;
        for i in 0 to 25 loop
           --Lectura de cadenas de VECTORES.txt
            readline(ARCH_VEC, LINEA_VEC);--Lee una linea completa
            hread(LINEA_VEC, V_PC_in);
            PC_in <= V_PC_in;
            read(LINEA_VEC, V_UP);
            UP <= V_UP;
            read(LINEA_VEC, V_DW);
            DW <= V_DW;
            read(LINEA_VEC, V_WPC);
            WPC <= V_WPC;
            read(LINEA_VEC, V_clr);
            clr <= V_clr;

            wait until RISING_EDGE(CLK);


            --Asignaci□e Salidas :0
            v_SP := SP;
            v_PC_out := PC_out;

            v_OP_CODE(4) := Inst(24);
            v_OP_CODE(3) := Inst(23);
            v_OP_CODE(2) := Inst(22);
            v_OP_CODE(1) := Inst(21);
            v_OP_CODE(0) := Inst(20);

            v_Rd(3) := Inst(19);
            v_Rd(2) := Inst(18);
            v_Rd(1) := Inst(17);
            v_Rd(0) := Inst(16);

            v_Rt(3) := Inst(15);
            v_Rt(2) := Inst(14);
            v_Rt(1) := Inst(13);
            v_Rt(0) := Inst(12);

            v_Rs(3) := Inst(11);
            v_Rs(2) := Inst(10);
            v_Rs(1) := Inst(9);
            v_Rs(0) := Inst(8);

            v_shamt(3) := Inst(7);
            v_shamt(2) := Inst(6);
            v_shamt(1) := Inst(5);
            v_shamt(0) := Inst(4);

            v_FUNC_CODE(3) := Inst(3);
            v_FUNC_CODE(2) := Inst(2);
            v_FUNC_CODE(1) := Inst(1);
            v_FUNC_CODE(0) := Inst(0);

            --Escritura de Resultados
            Hwrite(LINEA_RES, v_SP, right, 7);
            Hwrite(LINEA_RES, v_PC_out, right, 7);
            write(LINEA_RES, v_OP_CODE, right, 7);
            write(LINEA_RES, v_Rd, right, 7);
            write(LINEA_RES, v_Rt, right, 7);
            write(LINEA_RES, v_Rs, right, 7);
            write(LINEA_RES, v_shamt, right, 7);
            write(LINEA_RES, v_FUNC_CODE, right, 7);

            writeline(ARCH_RES,LINEA_RES);--Escribe la linea en el
archivo
        end loop;
        file_close(ARCH_VEC);--Cierra el archivo
        file_close(ARCH_RES);--Cierra el archivo
      wait;
   end process;-- Stimulus process

end Behavioral;
```

Simulación:





Archivo de entrada:

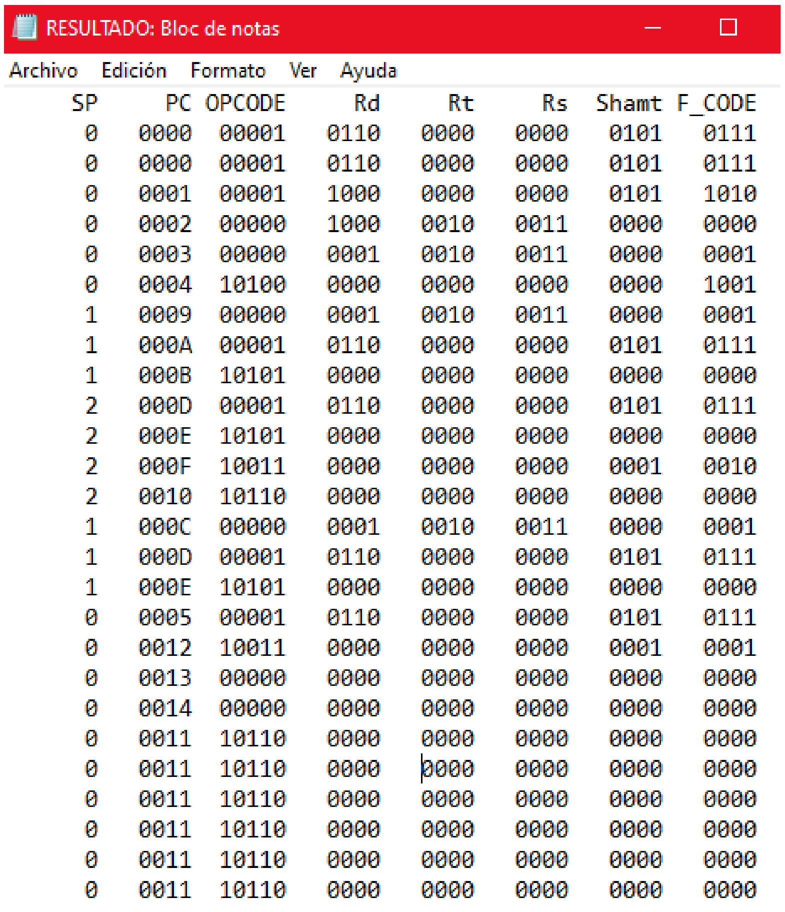


VECTORES: Bloc de notas

Archivo Edición Formato

```
0000 0 0 0 1
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0009 1 0 1 0
0000 0 0 0 0
0000 0 0 0 0
000d 1 0 1 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 1 0 0
0000 0 0 0 0
0000 0 0 0 0
0000 0 1 0 0
0012 0 0 1 0
0000 0 0 0 0
0000 0 0 0 0
0011 0 0 1 0
```

Archivo de salida:

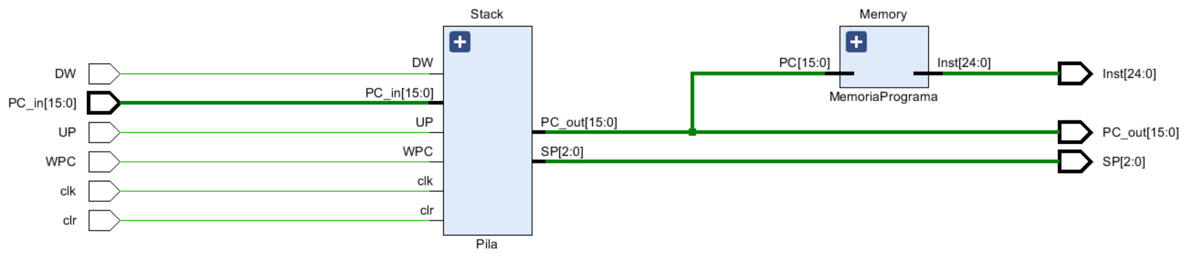


RESULTADO: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
   SP    PC  OPCODE    Rd     Rt     Rs   Shamt F_CODE
    0  0000  00001   0110   0000   0000   0101   0111
    0  0000  00001   0110   0000   0000   0101   0111
    0  0001  00001   1000   0000   0000   0101   1010
    0  0002  00000   1000   0010   0011   0000   0000
    0  0003  00000   0001   0010   0011   0000   0001
    0  0004  10100   0000   0000   0000   0000   1001
    1  0009  00000   0001   0010   0011   0000   0001
    1  000A  00001   0110   0000   0000   0101   0111
    1  000B  10101   0000   0000   0000   0000   0000
    2  000D  00001   0110   0000   0000   0101   0111
    2  000E  10101   0000   0000   0000   0000   0000
    2  000F  10011   0000   0000   0000   0001   0010
    2  0010  10110   0000   0000   0000   0000   0000
    1  000C  00000   0001   0010   0011   0000   0001
    1  000D  00001   0110   0000   0000   0101   0111
    1  000E  10101   0000   0000   0000   0000   0000
    0  0005  00001   0110   0000   0000   0101   0111
    0  0012  10011   0000   0000   0000   0001   0001
    0  0013  00000   0000   0000   0000   0000   0000
    0  0014  00000   0000   0000   0000   0000   0000
    0  0011  10110   0000   0000   0000   0000   0000
    0  0011  10110   0000   0000   0000   0000   0000
    0  0011  10110   0000   0000   0000   0000   0000
    0  0011  10110   0000   0000   0000   0000   0000
    0  0011  10110   0000   0000   0000   0000   0000
    0  0011  10110   0000   0000   0000   0000   0000
```

Diagrama RTL:


Stack
Memory
DW
PC_in[15:0]
UP
WPC
clk
clr
DW
PC_in[15:0]
UP
WPC
clk
clr
Pila
PC_out[15:0]
SP[2:0]
PC[15:0]
Inst[24:0]
MemoriaPrograma
Inst[24:0]
PC_out[15:0]
SP[2:0]