Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 8 - Memoria de Programa

Unidad de aprendizaje: Arquitectura de Computadoras

Grupo: 3CV1

Alumno(a):
Ramos Diaz Enrique

Profesor(a):
Vega García Nayeli

9 de abril 2020

Índice

1. Cálculo del tamaño de los buses de datos y de direcciones

Se sabe que 2^m x n = 3200 bytes = 25600 bits

El tamaño de la palabra es el tamaño de una instrucción completa del set ESCOMips, siendo de **n = 25 bits**.

Para obtener m es necesario despejar la primera ecuación:

$m = \log^2 \frac{(3200)(8)}{n}$

$m = \log^2 \frac{25600}{25} = 10$

m = 10 bits

2. Código de implementación

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

entity MemoriaPrograma is
    generic ( m : integer := 10;
              n : integer := 25);
    Port ( pc : in STD_LOGIC_VECTOR (m-1 downto 0);
           inst : out STD_LOGIC_VECTOR (n-1 downto 0));
end MemoriaPrograma;

architecture Behavioral of MemoriaPrograma is
    --Instrucciones tipo R
    constant tipoR : STD_LOGIC_VECTOR (4 downto 0) := "00000";
    --Carga y Almacenamiento
    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";
    --Aritmticas
    constant ADDI : STD_LOGIC_VECTOR (4 downto 0) := "00101";
    constant SUBI : STD_LOGIC_VECTOR (4 downto 0) := "00110";
    --Identificador Aritmticas R
    constant ADD : STD_LOGIC_VECTOR (3 downto 0) := "0000";
    constant SUB : STD_LOGIC_VECTOR (3 downto 0) := "0001";
    --Logicas
    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";
    --Identificador Logicas R
    constant ANDR : STD_LOGIC_VECTOR (3 downto 0) := "0010";
    constant ORR : STD_LOGIC_VECTOR (3 downto 0) := "0011";
    constant XORR : STD_LOGIC_VECTOR (3 downto 0) := "0100";
    constant NANDR : STD_LOGIC_VECTOR (3 downto 0) := "0101";
    constant NORR : STD_LOGIC_VECTOR (3 downto 0) := "0110";
    constant XNORR : STD_LOGIC_VECTOR (3 downto 0) := "0111";
    constant NOTR : STD_LOGIC_VECTOR (3 downto 0) := "1000";
    --Identificador Corrimiento R
    constant SLLR : STD_LOGIC_VECTOR (3 downto 0) := "1001";
    constant SRLR : STD_LOGIC_VECTOR (3 downto 0) := "1010";
    constant OPR : STD_LOGIC_VECTOR (4 downto 0) := "00000";
    --Saltos Condicionales e Incondicionales
    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";
    constant B : STD_LOGIC_VECTOR (4 downto 0) := "10011";
    --Manejo de Subrutinas
    constant CALL : STD_LOGIC_VECTOR (4 downto 0) := "10100";
    constant RET : STD_LOGIC_VECTOR (4 downto 0) := "10101";
    --Otros
    constant NOP : STD_LOGIC_VECTOR (4 downto 0) := "10110";
    constant SU : STD_LOGIC_VECTOR (3 downto 0) := "0000"; -- sin usar
    --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";

    type banco is array (0 to (2**m)-1) of STD_LOGIC_VECTOR(n-1 downto 0);
    constant aux : banco := (
        LI & R0 & x"0000",                      --LI  R0, #0
        LI & R1 & x"0001",                      --LI  R1, #1
        LI & R2 & x"0000",                      --LI  R2, #0
        LI & R3 & x"000c",                      --LI  R3, #12
        tipoR & R4 & R0 & R1 & SU & ADD,        --et1: ADD R4, R0, R1
        SWI & R4 & x"0072",                     --SWI R4, 0x72
        ADDI & R0 & R1 & x"000",                --ADDI R0, R1, #0
        ADDI & R1 & R4 & x"000",                --ADDI R1, R4, #0
        ADDI & R2 & R2 & x"001",                --ADDI R2, R2, #1
        BNEI & R2 & R3 & x"00b",                --BNEI R2, R3, et1 (1011 = -5)
        NOP & SU & SU & SU & SU & SU,           --fin: NOP
        B & SU & x"000a",                       --B fin
        others => (others => '0')
    );
begin
    inst <= aux(conv_integer(pc));
end Behavioral;
```

3. Código de simulación

```
library ieee;
library STD;
use ieee.STD_LOGIC_1164.ALL;
use ieee.STD_LOGIC_arith.all;
use ieee.STD_LOGIC_unsigned.ALL;
use ieee.STD_LOGIC_TEXTIO.ALL;
use STD.TEXTIO.ALL;

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is
    component MemoriaPrograma is
        Port ( pc : in STD_LOGIC_VECTOR (9 downto 0);
               inst : out STD_LOGIC_VECTOR (24 downto 0));
    end component;

    signal pc : STD_LOGIC_VECTOR (9 downto 0) := "0000000000";
    signal inst : STD_LOGIC_VECTOR (24 downto 0);
begin
    mp: MemoriaPrograma Port map (
        pc => pc,
        inst => inst
    );

    process
        file arch_res : text;   --Apuntadores tipo
         ↪  txt
        variable linea_res : line;
        variable var_inst : STD_LOGIC_VECTOR (24 downto 0);
        variable cadena : string (1 to 6);
    begin
        --- PC OPCODE 19..16 15..12 11..8 7..4 3..0
        file_open(arch_res, "Resultado.txt", WRITE_MODE);
        cadena := "PC    ";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
         ↪  "PC"
        cadena := "OPCODE";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
         ↪  "OPCODE"
        cadena := "19..16";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
         ↪  "19..16"
        cadena := "15..12";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
         ↪  "15..12"
        cadena := " 11..8";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
         ↪  "11..8"
        cadena := "  7..4";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
         ↪  "7..4"
        cadena := "  3..0";
        write(linea_res, cadena, right, cadena'LENGTH+1);--ESCRIBE LA cadena
         ↪  "3..0"
        writeline(arch_res, linea_res);-- escribe la linea en el archivo

        for i in 0 to 11 loop
            wait for 10 ns;
```

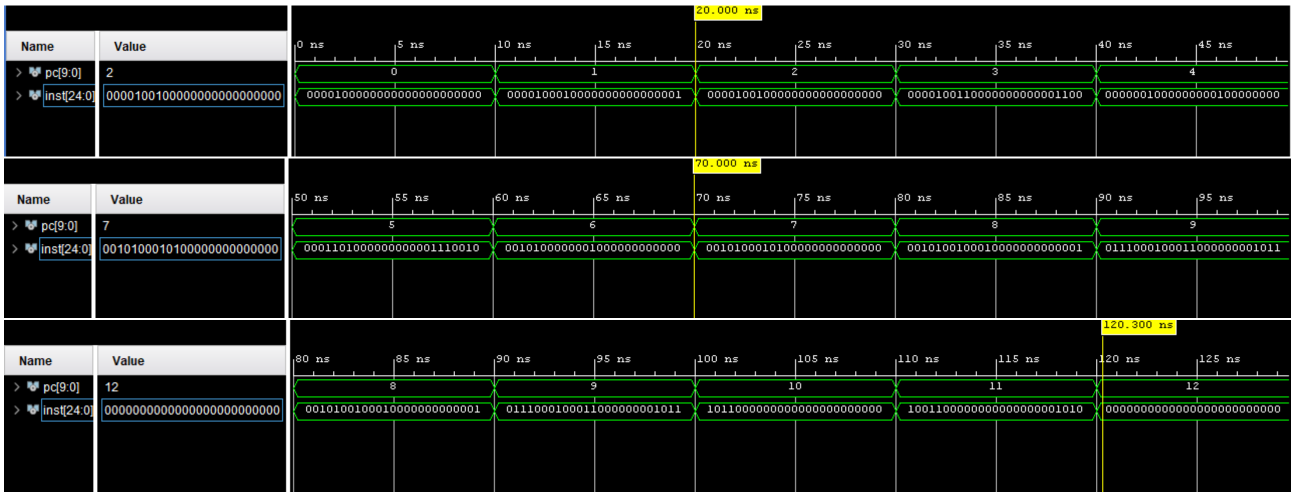
```vhdl
            var_inst := inst;
            Hwrite(linea_res, pc, left, 9);--ESCRIBE EL CAMPO PC
            write(linea_res, var_inst(24 downto 20), left, 8);--ESCRIBE EL CAMPO OPCODE
            write(linea_res, var_inst(19 downto 16), left, 7);--ESCRIBE EL CAMPO 19..16
            write(linea_res, var_inst(15 downto 12), left, 7);--ESCRIBE EL CAMPO 15..12
            write(linea_res, var_inst(11 downto 8), left, 7);--ESCRIBE EL CAMPO 11..8
            write(linea_res, var_inst(7 downto 4), left, 7);--ESCRIBE EL CAMPO 7 .. 4
            write(linea_res, var_inst(3 downto 0), left, 7);--ESCRIBE EL CAMPO 3 .. 0
            writeline(arch_res, linea_res);
            pc <= pc + 1;
        end loop;
        file_close(arch_res);  -- cierra el archivo
        wait;
    end process;
end Behavioral;
```

4. Simulación

4.1. Programa a "ejecutar"

```
     LI     R0, #0
     LI     R1, #1
     LI     R2, #0
     LI     R3, #12
et1: ADDI   R4, R0, R1
     SWI    R4, 0x72
     ADDI   R0, R1, #0
     ADDI   R1, R4, #0
     ADDI   R2, R2, #1
     BNEI   R2, R3, et1
fin: NOP
     B      fin
```

4.2. Forma de onda de simulación

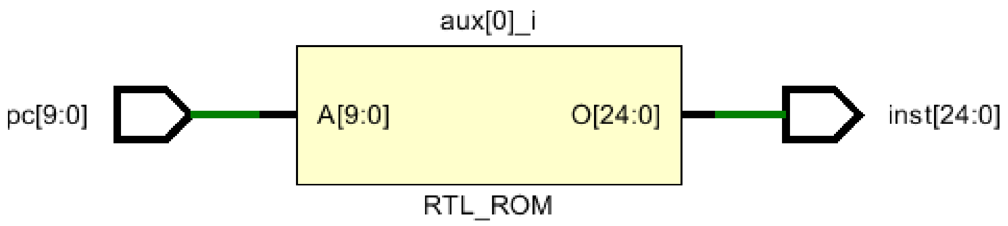


4.3. Archivo salida: Resultado.txt

```
 PC      OPCODE 19..16 15..12  11..8   7..4   3..0
000        00001   0000   0000   0000   0000   0000
001        00001   0001   0000   0000   0000   0001
002        00001   0010   0000   0000   0000   0000
003        00001   0011   0000   0000   0000   1100
004        00000   0100   0000   0001   0000   0000
005        00011   0100   0000   0000   0111   0010
006        00101   0000   0001   0000   0000   0000
007        00101   0001   0100   0000   0000   0000
008        00101   0010   0010   0000   0000   0001
009        01110   0010   0011   0000   0000   1011
00A        10110   0000   0000   0000   0000   0000
00B        10011   0000   0000   0000   0000   1010
```

5. Diagramas RTL

5.1. Análisis RTL



5.2. Synthesis