UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE ING. ELECTRONICA EAP 19.1
DISEÑO DIGITAL
LABORATORIO NUMERO 5

EL MICROPROCESADOR

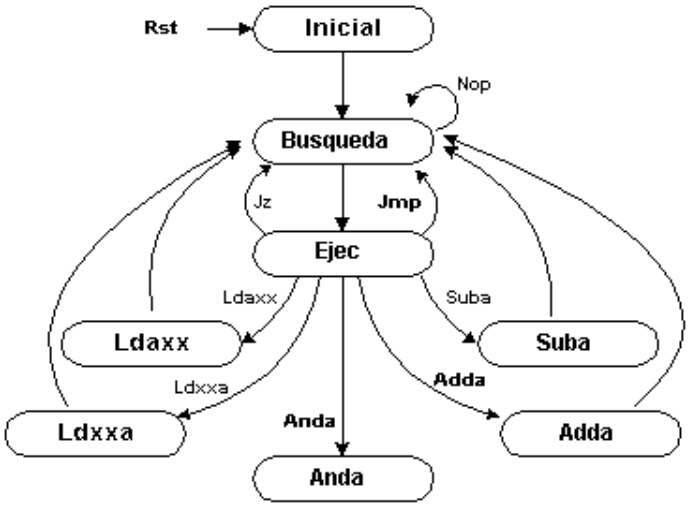
Teniendo como base el ejemplo del libro que es el siguiente:

Un uP con un bus de datos y direcciones de 8 bits

Y con las siguientes 8 instrucciones:

INSTRUCCIÓN	OPCODE	
Ld a,xx	000	:Carga a contenido de xx
Ld xx,a	001	:Carga en xx contenido de a
And a,xx	010	:Carga en a, cont. de xx And a
Add a,xx	011	:Carga en a, cont. xx + a
Sub a,xx	100	:Carga en a, cont. De xx – a
Jz xx	101	:Salta a xx si a='0'
Jmp xx	100	:Salta a xx
Nop	111	:No realiza operación

Cuyo diagrama de estados es el siguiente



Y cuyo código en VHDL es:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity procesador is
    Port (clk,rst:in std_logic;
                  r_w:out std_logic;
                  dir:out std_logic_vector(7 downto 0);
                  dat:inout std_logic_vector(7 downto 0));
end procesador;
architecture descripcion of procesador is
type estado is (inicial,busqueda,ejec,ldxxa,ldaxx,anda,adda,suba);
signal a,pc,carga_a,carga_pc:std_logic_vector(7 downto 0);
signal rdat_in,dat_in,dat_out:std_logic_vector(7 downto 0);
```

Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com

```
signal rwaux:std_logic;
signal presente: estado:= inicial;
begin
fsm:
Process(clk,rst)
begin
if rst='1'then
presente <= inicial;
elsif clk='1'and clk'event then

case presente is
when inicial =>
       presente <= busqueda ;
when busqueda =>
  if dat_in (2 downto 0 )= "111" then presente <= busqueda ;
  else  presente <= ejec;
  end if ;
when ejec =>
       case rdat_in (2 downto 0) is
       when "000"=>presente<=ldaxx;
       when "001"=>presente<=ldxxa;
       when "010"=>presente<=anda;
       when "011"=>presente<=adda;
       when "100"=>presente<=suba;
       when others=>presente<=busqueda;
       end case ;
when others=> presente<= busqueda ;
end case;
end if;
end process fsm;

salida:
Process(presente, pc,a,rdat_in,dat_in)
begin
       case presente is
       when inicial =>
              carga_pc<="00000000"; --pc a 0
              carga_a<="00000000"; --acumulador  a 0
              dir<=pc;  --Direccion a 0
              rwaux<= '1';
   when busqueda =>
              carga_pc<=pc+"00000001"; --incrementa pc
              carga_a<=a; --acumulador  a 0
              dir<=pc;  --Direccion a 0
              rwaux<= '1';
   when ejec=>
       case rdat_in (2 downto 0) is
       when "101"=>--jz
          dir<=pc;
              if a = "00000000" then carga_pc<= dat_in; --salta si A=0
              else carga_pc<= pc + "00000001";
              end if;
              rwaux<='1';
          carga_a<=a;
       when "110" => --jmp
          dir<=pc;
              carga_pc<=dat_in; -- salta siempre
              rwaux<= '1';
              carga_a <=a ;
       when others =>  --por defecto para el resto
              rwaux<= '1';
              dir <=pc ;
              carga_pc<=pc ;
          carga_a <=a ;
       end case;
   when ldaxx =>
              rwaux<= '1';
      carga_a<=dat_in ;
```

Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com

```
          carga_pc<=pc + "00000001" ;
                    dir <=rdat_in ;
       when ldxxa =>
              rwaux<= '0';
      carga_a<=a ;
          carga_pc<=pc + "00000001" ;
                    dir <=rdat_in ;
       when anda =>
              rwaux<= '1';
      carga_a<=a and dat_in  ;
          carga_pc<=pc + "00000001" ;
                    dir <=rdat_in ;
       when adda =>
              rwaux<= '1';
      carga_a<=a + dat_in ;
          carga_pc<=pc + "00000001" ;
                    dir <=rdat_in ;
       when suba =>
              rwaux<= '1';
      carga_a<=a - dat_in ;
          carga_pc<=pc + "00000001" ;
                    dir <=rdat_in ;
       end case;
       end process salida;

registro_de_entrada:
process(clk,rst)
begin
   if rst = '1' then rdat_in <= "00000000";
       elsif clk='1' and clk'event then rdat_in<= dat_in;
       end if;
       end process registro_de_entrada;

regs_con_carga:
process(clk,rst)
begin
   if rst ='1' then
       a<= "00000000";
       pc<= "00000000";
       elsif clk='1' and clk'event then
       a<= carga_a ;
       pc<=carga_pc;
       end if;
       end process regs_con_carga;
       r_w<= rwaux;---para leer salida
       --triestado de salida:
       dat_in<= dat;
       dat_out<=a;
       dat<=dat_out when rwaux ='0' else "ZZZZZZZZ";
       end descripcion;
```

y teniendo esto realizar:

CUESTIONARIO

1.-Hacer un archivo de test_beanch donde:

Se guarde en las direcciones

91H → 01
92H → 02
93H → 03
94H → 04

Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com

Para hacer el archivo en Test_beanch voy a crear el programa donde sus opcodes son:

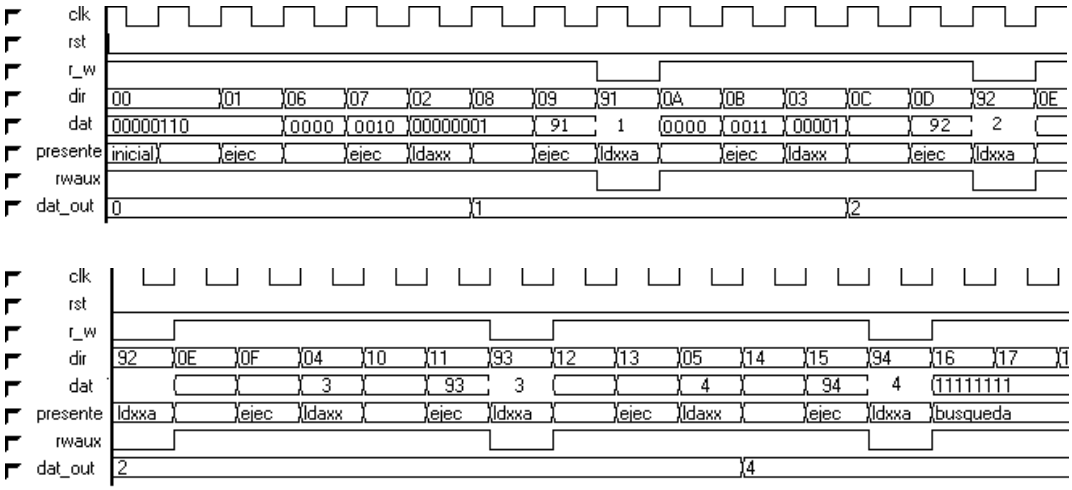
00:	Jmp 06	00000110
		00000110
02:	db 01	00000001
03:	db 02	00000010
04:	db 03	00000011
05:	db 04	00000100
06:	Ldaxx 02	00000000
		00000010
08:	Ldxxa 91	00000001
		10010001
10:	Ldaxx 03	00000000
		00000011
12:	Ldxxa 92	00000001
		10010010
14:	Ldaxx 04	00000000
		00000100
16:	Ldxxa 93	00000001
		10010011
18:	Ldaxx 05	00000000
		00000101
20:	Ldxxa 94	00000001
		10010100

y el código completo en test Beanch es:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
ENTITY testbench IS
END testbench;
ARCHITECTURE behavior OF testbench IS
	COMPONENT procesador
	PORT(
		clk : IN std_logic;
		rst : IN std_logic;
		dat : INOUT std_logic_vector(7 downto 0);
		r_w : OUT std_logic;
		dir : OUT std_logic_vector(7 downto 0)
		);
	END COMPONENT;
	SIGNAL clk :  std_logic:='0';
	SIGNAL rst :  std_logic;
	SIGNAL r_w :  std_logic;
	SIGNAL dir :  std_logic_vector(7 downto 0);
	SIGNAL dat :  std_logic_vector(7 downto 0);

BEGIN

	uut: procesador PORT MAP(
		clk => clk,
		rst => rst,
		r_w => r_w,
		dir => dir,
		dat => dat
	);


-- *** Test Bench - User Defined Section ***
	clk<=not clk after 50 ns;
	rst<='1','0' After 70 ns;
   tb : PROCESS (Dir)
   BEGIN
```

```
        case dir is
        when "00000000"=>dat<="00000110";-- 00 JMP
        when "00000001"=>dat<="00000110";-- 01 06
        when "00000010"=>dat<="00000001";-- 02  01
        when "00000011"=>dat<="00000010";-- 03 02
        when "00000100"=>dat<="00000010";-- 04 03
        when "00000101"=>dat<="00000100";-- 05 04
        when "00000110"=>dat<="00000000";-- 06 Ldaxx
        when "00000111"=>dat<="00000010";-- 07 02
        when "00001000"=>dat<="00000001";-- 08 Ldxxa
        when "00001001"=>dat<="10010001";-- 09 91
        when "00001010"=>dat<="00000000";-- 10 Ldaxx
        when "00001011"=>dat<="00000011";-- 11 03
        when "00001100"=>dat<="00000001";-- 12 Ldxxa
        when "00001101"=>dat<="10010010";-- 13 92
        when "00001110"=>dat<="00000000";-- 14 Ldaxx
        when "00001111"=>dat<="00000100";-- 15 04
        when "00010000"=>dat<="00000001";-- 16 Ldxxa
        When "00010001"=>dat<="10010011";-- 17 93
        when "00010010"=>dat<="00000000";-- 18 Ldaxx
        when "00010011"=>dat<="00000101";-- 19 05
        when "00010100"=>dat<="00000001";-- 20 Ldxxa
        when "00010101"=>dat<="10010100";-- 21 94
        when others=>dat<="11111111";
        end case;
--       wait; -- will wait forever
    END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

Los resultados fueron:



2.- Incrementar Mas instrucciones de este procesador:

Sol:

He incrementado 4 instrucciones:

INSTRUCCIÓN	OPCODE	
Or a,xx	1000	:Carga en a, cont. de xx Or a
Xor a,xx	1001	:Carga en a, cont. de xx Xor a
Not a	1010	:Carga en a, Not a
Inc a	1011	:Carga en a, a + 1
Dec a	1100	:Carga en a, a – 1

Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com

Y la nueva codificacion es:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity procmas is
    Port (clk,rst:in std_logic;
                  r_w:out std_logic;
                  dir:out std_logic_vector(7 downto 0);
                  dat:inout std_logic_vector(7 downto 0));
end procmas;
architecture descripcion of procmas is
type estado is
(inicial,busqueda,ejec,ldxxa,ldaxx,anda,adda,suba,ora,xora,nota,inca,deca);
signal a,pc,carga_a,carga_pc:std_logic_vector(7 downto 0);
signal rdat_in,dat_in,dat_out:std_logic_vector(7 downto 0);
signal rwaux:std_logic;
signal presente: estado:= inicial;
begin
fsm: Process(clk,rst)
begin
if rst='1'then
presente <= inicial;
elsif clk='1'and clk'event then
case presente is
when inicial =>     presente <= busqueda ;
when busqueda =>   if dat_in (2 downto 0 )= "111" then presente <= busqueda ;
  else  presente <= ejec;
  end if ;
when ejec =>  case rdat_in (3 downto 0) is
       when "0000"=>presente<=ldaxx;
       when "0001"=>presente<=ldxxa;
       when "0010"=>presente<=anda;
       when "0011"=>presente<=adda;
       when "0100"=>presente<=suba;
       when "1000"=>presente<=ora;
       when "1001"=>presente<=xora;
       when "1010"=>presente<=nota;
       when "1011"=>presente<=inca;
       when "1100"=>presente<=deca;
       when others=>presente<=busqueda;
       end case ;
when others=> presente<= busqueda ;
end case;
end if;
end process fsm;
salida:
Process(presente, pc,a,rdat_in,dat_in)
begin
       case presente is
       when inicial =>     carga_pc<="00000000"; --pc a 0
                           carga_a<="00000000"; --acumulador  a 0
                           dir<=pc;  --Direccion a 0
                           rwaux<= '1';
       when busqueda =>    carga_pc<=pc+"00000001"; --incrementa pc
                           carga_a<=a; --acumulador  a 0
                           dir<=pc;  --Direccion a 0
                           rwaux<= '1';
       when ejec=>         case rdat_in (2 downto 0) is
                     when "101"=>--jz
                                dir<=pc;
                     if a = "00000000" then carga_pc<= dat_in; --salta si A=0
                     else carga_pc<= pc + "00000001";
                     end if;
                     rwaux<='1';
                     carga_a<=a;
                     when "110" => --jmp
         dir<=pc;
```

Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com

```
             carga_pc<=dat_in; -- salta siempre
             rwaux<= '1';
             carga_a <=a ;
      when others =>  --por defecto para el resto
             rwaux<= '1';
             dir <=pc ;
             carga_pc<=pc ;
          carga_a <=a ;
      end case;
when ldaxx =>  rwaux<= '1';
                  carga_a<=dat_in ;
                carga_pc<=pc + "00000001" ;
                   dir <=rdat_in ;
when ldxxa =>  rwaux<= '0';
                 carga_a<=a ;
                carga_pc<=pc + "00000001" ;
                   dir <=rdat_in ;
when anda =>  rwaux<= '1';
             carga_a<=a and dat_in  ;
             carga_pc<=pc + "00000001" ;
             dir <=rdat_in ;
when adda =>  rwaux<= '1';
              carga_a<=a + dat_in ;
              carga_pc<=pc + "00000001" ;
              dir <=rdat_in ;
when suba =>  rwaux<= '1';
              carga_a<=a - dat_in ;
              carga_pc<=pc + "00000001" ;
              dir <=rdat_in ;
when ora =>  rwaux<= '1';
             carga_a<=a or dat_in ;
             carga_pc<=pc + "00000001" ;
             dir <=rdat_in ;
when xora =>  rwaux<= '1';
             carga_a<=a xor dat_in ;
             carga_pc<=pc + "00000001" ;
             dir <=rdat_in ;
when nota =>  rwaux<= '1';
             carga_a<=not a;
             when inca =>
             rwaux<= '1';
             carga_a<=a + "00000001" ;
when deca =>  rwaux<= '1';
             carga_a<=a  - "00000001" ;
end case;
end process salida;
registro_de_entrada:
process(clk,rst)
begin
   if rst = '1' then rdat_in <= "00000000";
       elsif clk='1' and clk'event then rdat_in<= dat_in;
       end if;
       end process registro_de_entrada;
regs_con_carga:
process(clk,rst)
begin
   if rst ='1' then
       a<= "00000000";
       pc<= "00000000";
       elsif clk='1' and clk'event then
       a<= carga_a ;
       pc<=carga_pc;
       end if;
       end process regs_con_carga;
       r_w<= rwaux;---para leer salida
       --triestado de salida:
       dat_in<= dat;
       dat_out<=a;
```

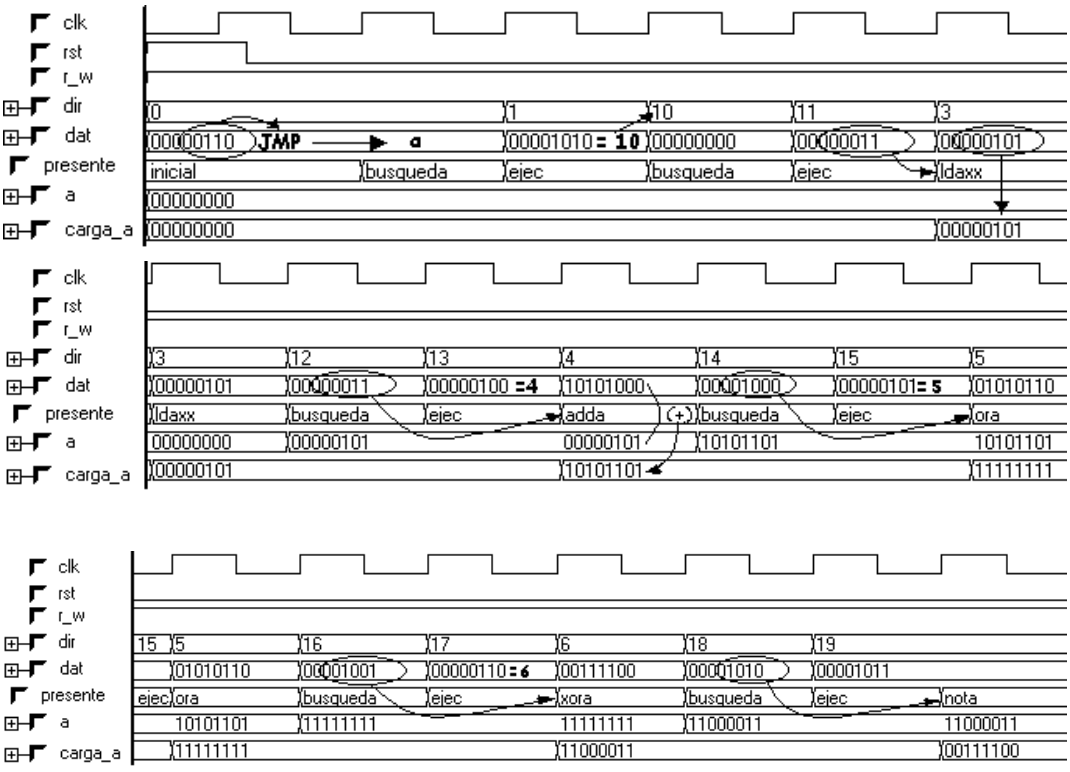
Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com

```
        dat<=dat_out when rwaux ='0' else "ZZZZZZZZ";
        end descripcion;
```

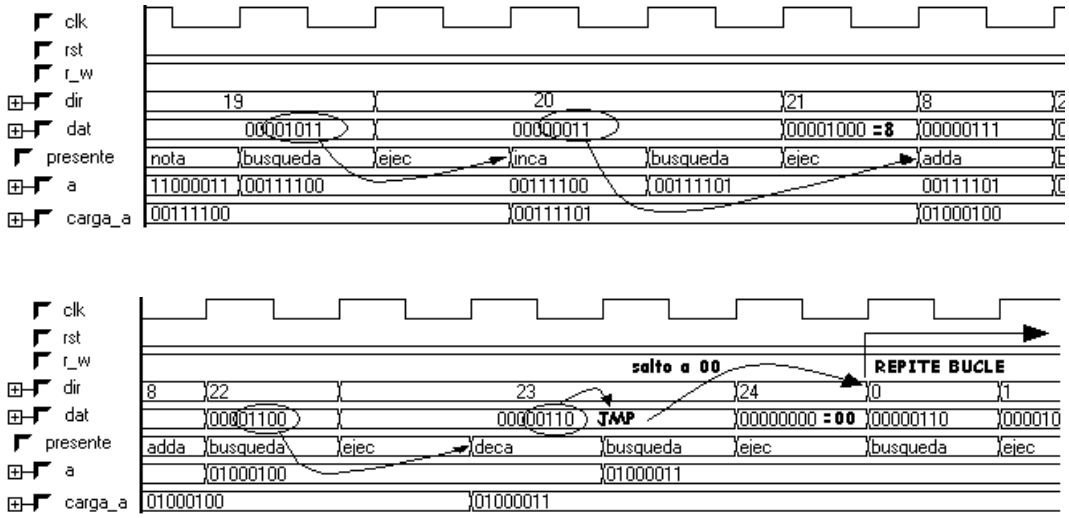
Y para su simulación usamos el siguiten case en el process:

```
        if (r_w='1') then
        case dir is
        when "00000000"=>dat<="00000110";-- 00 JMP
        when "00000001"=>dat<="00001010";-- 01 10
        when "00000010"=>dat<="00000000";-- 02
        when "00000011"=>dat<="00000101";-- 03
        when "00000100"=>dat<="10101000";-- 04
        when "00000101"=>dat<="01010110";-- 05
        when "00000110"=>dat<="00111100";-- 06
        when "00000111"=>dat<="00000000";-- 07
        when "00001000"=>dat<="00000111";-- 08
        when "00001001"=>dat<="00000000";-- 09
        when "00001010"=>dat<="00000000";-- 10 Ldaxx
        when "00001011"=>dat<="00000011";-- 11 03
        when "00001100"=>dat<="00000011";-- 12 Add
        when "00001101"=>dat<="00000100";-- 13 04
        when "00001110"=>dat<="00001000";-- 14 Ora
        when "00001111"=>dat<="00000101";-- 15 05
        when "00010000"=>dat<="00001001";-- 16 Xor
        When "00010001"=>dat<="00000110";-- 17 06
        when "00010010"=>dat<="00001010";-- 18 NotA
        when "00010011"=>dat<="00001011";-- 19 IncA
        when "00010100"=>dat<="00000011";-- 20 Add
        when "00010101"=>dat<="00001000";-- 21 08
        when "00010110"=>dat<="00001100";-- 22 DecA
        when "00010111"=>dat<="00000110";-- 23 Jmp
        when "00011000"=>dat<="00000000";-- 24 00
        when others=>dat<="11111111";
        end case;
end if;
```

Y los resultados de la simulacion son:



Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com



3.- Añadir al up 2 registros de trabajo R1,R2 de 8 bits c/u las operaciones pueden ser solo entre el acumulador y los registros R1,R2

Mov A,X
Mov X,A
Mov A,Y

Agregamos otra funcion mas : la función de opcode XX1101 y de variables:XX
Cuando:XX=(D5)(D4)

XX=00=> Mov a,R2
XX=01=> Mov R2,a
XX=10=> Mov a,R1
XX=11=> Mov R1,a

Para eso aumento:
En mis tipos de estado uno nuevo: mov
Y en mi arquitectura: otro caso:

```
when mov =>  rwaux<= '1';
                    case dat_in (5 downto 4) is
                        when "00"=>carga_a<=R2;
                        when "01"=>R2<=carga_a;
                        when "10"=>carga_a<=R1;
                        when "11"=>R1<=carga_a;
                        when others=>a<=a;
                    end case;
```

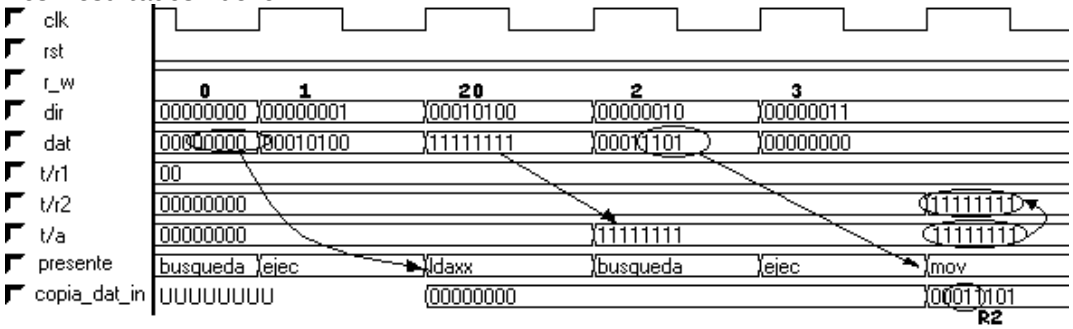
Creamos el programa:
Ld a,20 –-Cargamos a a el contenido de 20 =[20]=”11111111”
Mov R2,a -–Movemos este dato al R2
Ld a,21 -- Cargamos en a el contenido de 21=[21]= “11111111“
Add a,22 –- Le sumamos [22]=”11111111” =>a=”11111110”
Mov R1,a -- Movemos el resultado en R1

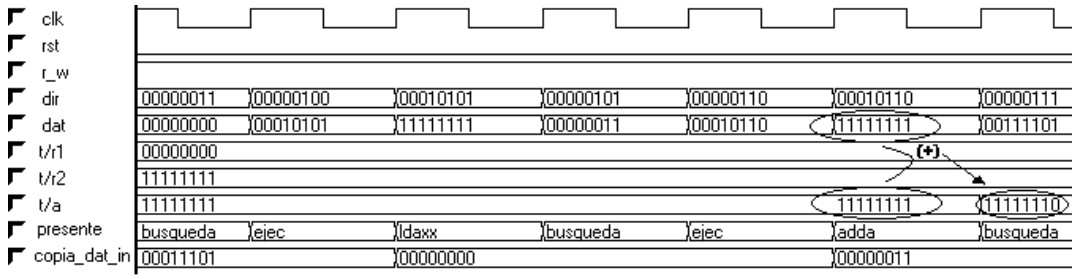
Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com

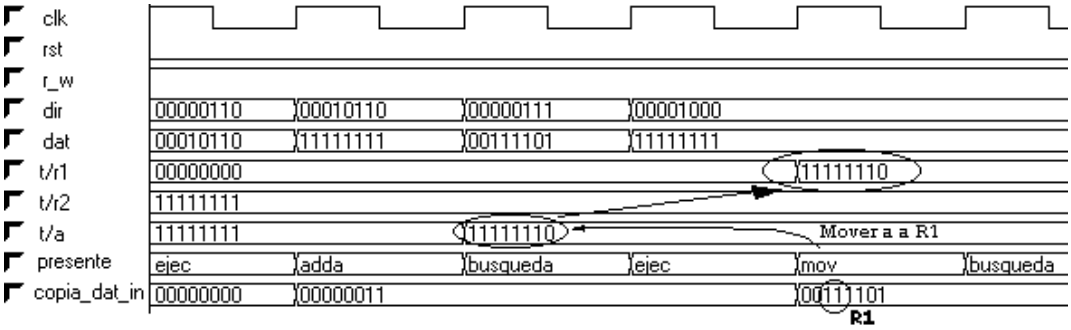
El Case Dir que usare en el archivo de Test Beanch seria:

```
case dir is
when "00000000"=>dat<="00000000";     -- 00 Ldaxx
when "00000001"=>dat<="00010100";     -- 01 20
when "00000010"=>dat<="00011101";     -- 02    Mov R2,a
when "00000011"=>dat<="00000000";     -- 03 Ldaxx
when "00000100"=>dat<="00010101";     -- 04 21
when "00000101"=>dat<="00000011";     -- 05 Adda
when "00000110"=>dat<="00010110";     -- 06 22
when "00000111"=>dat<="00111101";     -- 07 Mov R1,a
when others=>dat<="11111111";
end case;
```

Los Resultados fueron:







Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

(Universidad del Perú, Decana de América)

FACULTAD DE INGENIERIA ELECTRÓNICA (19.1)

LABORATORIO 5 : MICROPROCESADOR

CURSO: DISEÑO DIGITAL

CICLO : VIII TURNO : Lunes de 2-6pm

PROFESOR : Ing. Alfredo Granados LY

ALUMNO : OCMIN GRANDEZ, ELVER YOEL

COD : 993908

2003

Elver Yoel OCMIN GRANDEZ
yoelocmin@hotmail.com