UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

(Universidad del Perú, Decana de América)

FACULTAD DE INGENIERIA ELECTRÓNICA (19.1)

PROYECTO : MICROPROCESADOR

CURSO: DISEÑO DIGITAL

CICLO : VIII TURNO : Lunes de 2-6pm

PROFESOR : Ing. Alfredo Granados LI

ALUMNO : OCMIN GRANDEZ, ELVER YOEL

COD : 993908

2003

C. U. Julio 2003

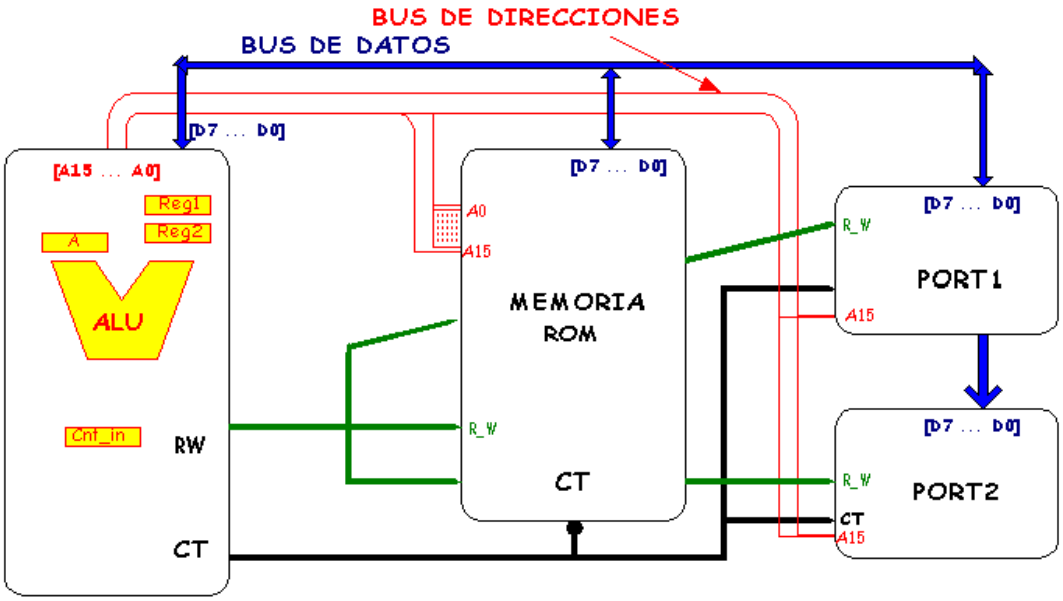
Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

UNMSM: Proyecto Final de Diseño Digital

PROYECTO:MICROPROCESADOR

CARACTERISTICAS:

Bus De Datos de 8 bits (1 byte) (D7 ... D0)
Bus de Direcciones de 16 Bits (A15.... A0)
2 posibles salidas a Puertos PORT1 y PORT2
8 Tipos de Instrucciones



INSTRUCCIONES:

Las instrucciones las he separado de acuerdo a su funcion en 8 tipos y estan diferenciadas por los 3 primeros bits (D2 ... D0) de nuestro primer bits de dato. Estos tipos son

TIPO	CODIGO (D2 D1 D0)	Abreviatura
Acceso a Memoria	000	Load
Acceso a Registros	001	Mover
Operac. Aritmeticas	010	Alu
Manejo de Bits	011	Movebits
Salto y Comparacion	100	Sal-Comp
Manejo de Puertos	101	Puertos
Operac. Logicas	110	Logicas
No operacion	111	Nop

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

DESCRIPCION de INSTRUCCIONES

1.- ACCESO a MEMORIA Load (000)

Sintaxis Ld xxx,xxx

Permite tener un contacto directo con una memoria externa, ahí se realizan 2 operaciones

- *** Ld a,DIRECCION**, Carga en el acumulador el contenido del dato que se encuentra en el lugar DIRECCION
- *** Ld DIRECCION,a** Carga en [direccion], el contenido del acumulador

Para diferenciar estas 2 operaciones se usa el bit D4, del dato de entrada, y la direccion a leer se encuentra el los siguientes dos bytes que le siguen.
Es decir que para completar el ciclo de la instrucción se debió ir primero 2 veces más a memoria

Su **OPCODE** tiene la forma:

(D23) . . . (D8)	**000(D4)**	**0000**
DIRECCION	TIPO	OPER
16bits	4 bits	4bits

Con lo que podemos apreciar que nuestro codigo ocupa 3 bytes

2.- ACCESO a REGISTROS Internos (001)

Permite mover internamente los registros del uP y las operaciones que se realizan estan diferenciadas por el bits2:

D4: Indica de donde a donde se va a mover el dato:
- '0': del acumulador a un REG
- '1': de un REG al acumulador

D6: Indica registro a usar en operacion:
- '0': REG1
- '1': REG2

Sintaxis:

- *** Mov a,REG** ; Carga en el acumulador el contenido del registro indicado
- *** Mov REG,a**; Carga en el registro inidicado, el contenido del acumulador

Su OPCODE seria:

0 (D6)	**(D5)(D4)**	**0010**
REG	TIPO	OPER
2 bits	2 bits	4bits

Con esto podemos apreciar que nuestro código ocupa 1 bytes

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

3.- OPERACIONES ARITMETICAS (011)

Este uP ha sido equipado con 4 operaciones Aritméticas, y todas estas se realizan con el acumulador, estas operaciones son indicadas por los bits (D5)(D4) y son las siguientes:

(D5)(D4)	Operación	Descripción
0 0	**ADDA R**	**Suma al acumulador el registro R**
0 1	**ADDC R**	**Suma el registro y el carry al acumulador**
1 0	**SUBA R**	**Resta al Acumulador el registro R**
1 1	**SUBC R**	**Resta al Acumulador (REG + Carry)**

Como podemos apreciar tambien vamos a usar a un registro, y este lo seleccionamos con el bit **D6**

D6: Indica registro a usar en operacion:

'0': REG1

'1': REG2

Sintaxis:

*** OPERACIÓN R**

Su OPCODE tiene la forma:

0 (D6)	(D5)(D4)	0010
REG	TIPO	OPER
2 bits	2 bits	4bits

Con esto podemos apreciar que nuestro codigo ocupa 1 bytes

4.- MANEJO de BITs (100)

Este uP puede manejar bits, es decir rotarlos y desplazarlos; y estos operaciones estas se realizan con el acumulador, estas operaciones son indicadas por los bits (D4)(D3) y son las siguientes:

(D4)(D3)	Operación	Descripción
0 0	**ROR X**	**Rota a la Derecha X bits del acumulador**
0 1	**ROL X**	**Rota a la Izquierda X bits del acumulador**
1 0	**SHR X**	**Desplaza a la Derecha X bits de a.**
1 1	**SHL X**	**Desplaza a la Izquierda X bits de a.**

Como podemos apreciar tambien vamos a usar X, que debe ser de 3 bits y se guarda en los bits **(D7)(D6)(D5)**

Sintaxis:

*** OPERACIÓN X**

Su OPCODE tiene la forma:

(D7)(D6)(D5)	(D4)(D3)	011
Veces	TIPO	OPER
3 bits	2 bits	3bits

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

Con esto podemos apreciar que nuestro codigo ocupa 1 bytes

5.- SALTOS y COMPARACION (100)

Este uP ha sido equipado con 2 registro mas (zero y carry) que nos permite realizar comparaciones y poder realizar saltos condicionales, por ser esta maquina ruimentaria, solo afectan a zero y carry la operación de comparación:

Estas operaciones son diferenciadas (SALTO de COMPARACION) con el bit **D3**:

'0' indica salto -> instrucción de 3 bytes

'1' indica comparación -> instrucción de 1 byte

caso **SALTOS (0100):**

Sintaxis:

*** SALTO DIRECCION**

Los tipos de saltos de los que disponemos son 7, estos vienen diferenciados en los bits (D6)(D5)(D4):

(D6)(D5)(D4)	Operación	Descripción
0 0 0	Jz	Salta si zero='1'
0 0 1	Jnz	Salta si zero='0'
0 1 0	Jl	Salta si es menor
0 1 1	Jg	Salta si e mayor
1 0 0	Jle	Salta si es menor o igual
1 0 1	Jge	Salta si es mayor o igual
1 1 0	Jmp	Salta siempre

Su **OPCODE** tiene la forma:

(D23)...(D8)	0(D6)(D5)(D4)	0100
DIRECCION	TIPO	OPER
16bits	4 bits	4bits

Con lo que podemos apreciar que nuestro codigo ocupa 3 bytes

caso **COMPARACION (1100):**

Sintaxis:

CMP OPE1, OPE2

Realiza lo siguiente:

Si OPE1 > OPE2 :carry =1

Si OPE1 < OPE2 :carry =0

Si OPE1 = OPE2 :carry =1

Donde OPE1 y OPE2 son escogidos de acuerdo a lo que indiquen los bits (D5)(D4):

00 => OPE1 = REG y OPE2 = a

01 => OPE1 = REG1 y OPE2 = REG2

10 => OPE1 = REG2 y OPE2 = REG1

11 => OPE1 = a y OPE2 = REG

Donde el REG depende del bit (D6):

Elver Yoel OCMIN Grandez

yoelocmin@hotmail.com

'0' => REG = REG1
'1' => REG = REG2

Su OPCODE tiene la forma:

0 (D6)	**(D5)(D4)**	**1100**
REG	TIPO	OPER
2 bits	2 bits	4bits

Con esto podemos apreciar que nuestro codigo ocupa 1 bytes

6.- MANEJO de Puertos (101)

Este uP puede manejar 2 puertos, es decir 2 perifericos externos, El dato se recibe y se envia por el acumulador, y los puertos deben encontrarse habilitados por un control CT='0', y para poder escoger entre cual usar usaremos a A(15) en la practica pero en el codigo usamos al bit **(D6)**

'0' => Usamos el PORT1
'1' => Usamos el PORT2

Existen 2 tipos de instrucciones para el manejo del puerto y esta viene determinado por el bit **(D4)**

'0' => Instrucción In
'1' => Instrucción Out

Sintaxis:

OPERACIÓN PUERTO

Su OPCODE tiene la forma:

0(D6) **Puerto**	**(0)(D4)** TIPO	**0101** OPER
2 bits	2 bits	4bits

Con esto podemos apreciar que nuestro codigo ocupa 1 bytes

7.- OPERACIONES LOGICAS (110)

Este uP ha sido realiza las 4 operaciones Lógicas basicas (And, or, not y xnor), y todas estas se realizan con el acumulador, estas operaciones son indicadas por los bits (D5)(D4) y son las siguientes:

(D5)(D4)	**Operación**	**Descripción**
0 0	**ANDA R**	**A <- A and R**
0 1	**NOTA R**	**A <- Not A**
1 0	**ORA R**	**A <- A or R**
1 1	**XORA R**	**A <- A Xor R**

Como podemos apreciar tambien vamos a usar a un registro, y este lo seleccionamos con el bit **D6**

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

D6: Indica registro a usar en operacion:
'0': REG1
'1': REG2

Sintaxis:

*** OPERACIÓN R**

Su OPCODE tiene la forma:

0 (D6)	(D5)(D4)	0010
REG	TIPO	OPER
2 bits	2 bits	4bits

Con esto podemos apreciar que nuestro codigo ocupa 1 bytes

8.- No OPERACIÓN NOP(111)

Esta instrucción no realiza nada en especial

Su sintaxis:

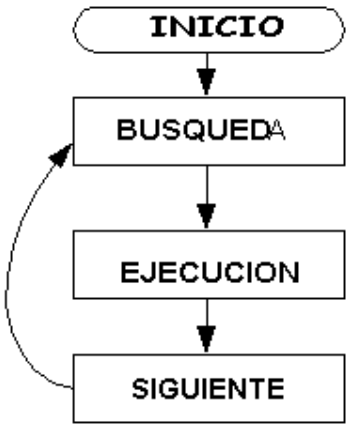
NOP

Su OPCODE tiene la forma:

XXXXX	111
Cualquier Cosa	OPER
5 bits	3bits

DIAGRAMA DE ESTADO del uP

En una forma global este uP trabaja de la siguiente manera;



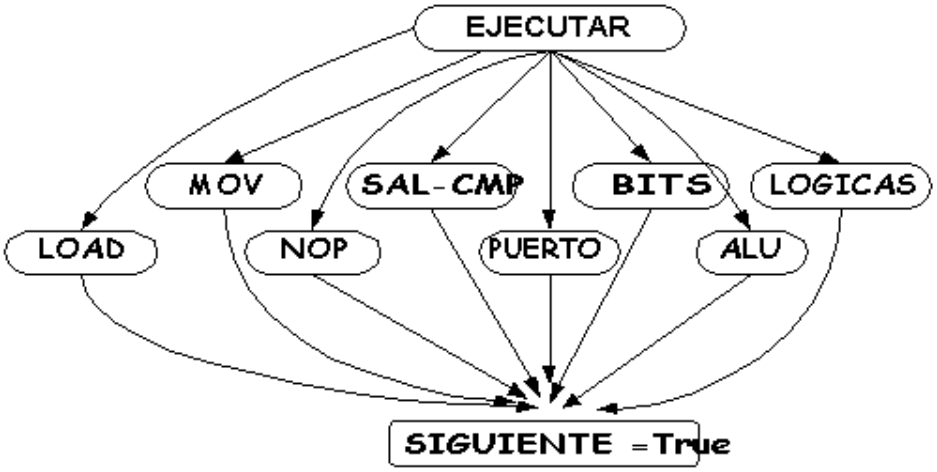
INICIO: Es la estapa inicial, cuando se le da un Reset, en esta etapa lo que hace es pasar a la siguiente etapa que es de búsqueda, aquí se resetean todos loa valores

BUSQUEDA: en esta etapa, se incrementa la direccion, el contador interno se vuelve cero, ya que no se realiza ninguna operación, y se pasa al estado de EJECUCION

EJECUCION: Es la estapa principal , aquí se decodifica el tipo de operación a realizar y dependiendo de la operación se usa o no al contador interno de instrucción cnt_in, en esta estapa se llama a las operaciones y se sale de ellas cuando siguiente (Que no es u estado, sino una señal booleana) es true(Verdadero).

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

Esta parte es representado gráficamente en el siguiente grafico:



Es decir en Ejecutar se hace que se pase al estado de operación donde de acuerdo a los estados, se realizan las operaciones y una vez terminados, hacen que siguiente sea trae y con esto se cambia al estado de búsqueda nuevamente

CODIFICACION:

Codigo en VHDL del Microprocesador:

He usado dos codificaciones:

Procesador.vhd

```
--Uso las librerias Clasicas
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--Defino mi entidad con entradas y salidas Std_logic
entity procesador is
    Port (clk,rst:in std_logic;
              CT,r_w:out std_logic;--Para el control de dispositivos
              dir:out std_logic_vector(15 downto 0);--Direccion
              dat:inout std_logic_vector(7 downto 0));--Bus de Datos
end procesador;

architecture Descripcion of procesador is
-- Creo mis estados
type estado is (inicial,busqueda,ejec);
-- Creo mis tipo de operacion
type tipoper is (load,mover,arith,movebits,salto,compa,puertos,logicos,nop);
-- Señales para el uso de los registros
signal a,reg1,reg2,copia_dat_in,carga_a:std_logic_vector(7 downto 0);
-- Señales para el uso de datos
signal dat_in,dat_out,copia_reg1:std_logic_vector(7 downto 0);
-- Señales para el direccionamiento
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

```
signal copia_dir,carga_dir:std_logic_vector(15 downto 0);
-- Señales para el control de dispositivos
signal rwaux,ctrl :std_logic;
-- Señales para registros de estado
signal zero,carry:std_logic;
signal presente: estado;
signal operacion:tipoper;
signal DATO_ENTRA:std_logic;
signal siguiente:bit;
signal cnt_in:std_logic_vector(2 downto 0);
begin
fsm:
	PROCESS(clk,rst)
variable copia_dat_in1:std_logic_vector (7 downto 0);
	BEGIN
		if rst='1' then--Si hay reset inicializo estados
			presente<=inicial;
		elsif clk='1' and clk'event then--Para cada Clk
			status: case presente is--Veo el estado presente
			when inicial =>--si es inicial
				presente<=busqueda;--Paso a estado de busqueda
			when busqueda=>--si es busqueda
				presente<=ejec;--Paso a estado de ejecucion
			when ejec=>--Copia mi dato de entrada ante posibles cambios
					if(cnt_in="000") then
					copia_dat_in<=dat_in;
					copia_dat_in1:=dat_in;
					end if;
		if siguiente='1' then
			presente<=busqueda;end if;
		ejecution: case copia_dat_in1(2 DOwNTO 0) is
		--Seleccione Tipo de operacion
		when "000"=>operacion<=load;
		when "001"=>operacion<=mover;
		when "010"=>operacion<=arith;
		when "011"=>operacion<=movebits;
		when "100"=>if(copia_dat_in1(3)='0') then--Si el tipo es salto
				operacion<=salto;
				else--Si es comparacion
				operacion<=compa;end if;
		when "101"=>operacion<=puertos;
		when "110"=>operacion<=logicos;
		when others=>operacion<=nop;
		end case ejecution;
			end case status;
	end if;--Del Rst='1'
end process fsm;

salidas:process(clk)
	variable bites:std_logic_vector(2 downto 0);
	variable tipo1:std_logic;
	variable operando1,operando2,operandoR:std_logic_vector(7 downto 0);
	variable dir_salto:std_logic_vector(15 downto 0);
begin
if(clk='1' and clk'event)then
	case presente is
		when inicial =>carga_a<=(others=>'0');
				reg2<=(others=>'0');
				reg1<=(others=>'0');
				carga_dir<=(others=>'0');
				rwaux<='1';--Inicializo en lectura
		when busqueda=>cnt_in<="000";
				--Inicializo contador interno de instruccion
		carga_dir<=carga_dir+"0000000000000001";-- Incr instrcucion
		rwaux<='1';--Habilito solo lectura
		ctrl<='1';--Desabilito puertos
		siguiente<='0';
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

```
when ejec=>--Escojo el tipo de operacion a Realizar
case operacion is--Escojo operacion
when load=>--Si es Load Durante el 1er Clock
            if(cnt_in ="000")then
                        --Incr mi direccion de lectura de datos
            copia_reg1<=reg1;
            carga_dir<=carga_dir+"0000000000000001";
            --Incr mi contador interno
            cnt_in<=cnt_in+"001";
            elsif(cnt_in ="001") then--2do clock
            reg1<=dat_in;--Guardo mi dato de entrada
            --en Reg1, dat_in=>(A7 ... A0);
            --Incr mi direccion de lectura de datos

            carga_dir<=carga_dir+"0000000000000001";
            --incr mi contador interno de instruccion
            cnt_in<=cnt_in+"001";
            elsif(cnt_in ="010") then
            copia_dir<=carga_dir;--Guardo Dire inicial
            carga_dir<=dat_in&reg1;--carga_dir <=(A15 ..A0)
                        --Configuro Modo Lectura o Escritura
            if(copia_dat_in(4)='0')then--LR
             rwaux<='1';--si es Ld a,XXX
            else carga_a<=a;--Si es WR
             rwaux<='0';--si es Ld XXX,a
            end if;
            cnt_in<=cnt_in+"001";
            elsif(cnt_in="011") then--Ejecuto Operacion
            carga_dir<=copia_dir;
            reg1<=copia_reg1;
            if(copia_dat_in(4)='0')then-- XXX->Acumulador
            carga_a<=dat_in;
            else carga_a<=a;--Acumulador -> XXX
            end if;
            siguiente<='1';
            end if;
       when mover=>--Si es Mov
         if(dat_in(4)='0') then-- Tipo: A -> REG
            if(dat_in(6)='0') then reg1<=a;else reg2<=a;end if;
         else --     REG -> Acumulador
            if(dat_in(6)='0') then carga_a<=reg1;else
            carga_a<=reg2;end if;
         end if;
            siguiente<='1';
       when arith=>--//ALU
        if copia_dat_in(5 downto 4)="00" then -- A+REG
            if copia_dat_in(6)='0' then--Escojo:
                  carga_a<=a+reg1;--Reg1
            else   carga_a<=a+reg2;--Reg2
            end if;
        elsif copia_dat_in(5 downto 4)="01" then-- A+REG+carry
            if copia_dat_in(6)='0' then--Escojo
            carga_a<=a+reg1+"0000000"&carry;--Reg1
            else  carga_a<=a+reg2+"0000000"&carry;--Reg2
            end if;
        elsif copia_dat_in(5 downto 4)="10" then-- A - REG
            if copia_dat_in(6)='0' then--Escojo
            carga_a<=a-reg1;--Reg1
            else  carga_a<=a-reg2;--Reg2
            end if;
        elsif copia_dat_in(5 downto 4)="11" then -- A-(REG+carry)
            if copia_dat_in(6)='0' then--Escojo
            carga_a<=a-(reg1+"0000000"&carry);--Reg1
            else carga_a<=a-(reg2+"0000000"&carry); --Reg2
            end if;
        end if;
       siguiente<='1';
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

```
when movebits=>--//manejo de Bits
          bites:=copia_dat_in(7 downto 5);
    case copia_dat_in(4 Downto 3) is
    when "00"=>--//rotar a la izq
                rorizq:For i in 0 to 7 loop
                if(i<=bites) then
          carga_a<=a(6)&a(5)&a(4)&a(3)&a(2)&a(1)&a(0)&a(7);
                end if;
                end loop rorizq;
    when "01"=>--//rotar a la Der
                rorder:For i in 0 to 7 loop
                if(i<=bites) then
          carga_a<=a(0)&a(7)&a(6)&a(5)&a(4)&a(3)&a(2)&a(1);
                end if;
                end loop rorder;
    when "10"=>--//Desp a la Der
                despizq:For i in 0 to 7 loop
                if(i<=bites) then
          carga_a<='0'&a(7)&a(6)&a(5)&a(4)&a(3)&a(2)&a(1);
                end if;
                end loop despizq;
    when others =>--//Desp a la Izq
          despder:For i in 0 to 7 loop
                if(i<=bites) then
          carga_a<=a(6)&a(5)&a(4)&a(3)&a(2)&a(1)&a(0)&'0';
                end if;
                end loop despder;
    end case;
    siguiente<='1';
   when compa=>if(dat_in(6)='0') then--Reg1
          operandoR:=Reg1;
          else operandoR:=Reg2; end if;
          case copia_dat_in(5 Downto 4) is--Tipo operando
          when "00" =>operando1:=operandoR;
                      operando2:=a;
          when "01" =>operando1:=reg1;
                      operando2:=reg2;
          when "10" =>operando1:=reg2;
                      operando2:=reg1;
          when others =>operando1:=a;
                      operando2:=operandoR;
          end case;
          if(operando1 > operando2) then
                     carry<='1';
                     zero<='0';
          elsif(operando1 < operando2) then
                     carry<='0';
                     zero<='0';
          else zero<='1';end if;
                     siguiente<='1';
      when salto=>--Si es salto
                     if(cnt_in="000") then--En el 1er Clk
          carga_dir<=carga_dir+"0000000000000001";
          copia_reg1<=reg1;
          cnt_in<=cnt_in+"001";
          elsif(cnt_in="001") then
          carga_dir<=carga_dir+"0000000000000001";
          reg1<=dat_in;
          cnt_in<=cnt_in+"001";
          copia_dir<=carga_dir+"0000000000000001";
          elsif(cnt_in="010") then
          reg1<=copia_reg1;
          dir_salto:=dat_in&reg1;
          casosaltos: case copia_dat_in(6 downto 4) is
   when "000"=>if(zero='1')then carga_dir<=dir_salto;--JZ
                else carga_dir<=copia_dir;--Se queda ahi
                end if;
```

```
                    when "001"=>if(zero='0')then carga_dir<=dir_salto;--JNZ
                                else carga_dir<=copia_dir;--Se queda ahi
                                end if;
                    when "010"=>if(carry ='0')then carga_dir<=dir_salto;--Jl
                                else carga_dir<=copia_dir;--Se queda ahi
                                end if;
                    when "011"=>if(carry='1')then carga_dir<=dir_salto;--JG
                                else carga_dir<=copia_dir;--Se queda ahi
                                end if;
                    when "100"=>tipo1:=(zero or carry) and carry;
                                if(tipo1='1')then
                                carga_dir<=dir_salto;--Jle
                                   else carga_dir<=copia_dir;--Se queda ahi
                                end if;
                    when "101"=>tipo1:=(zero or carry) and carry;
                                if(tipo1='0')then--Se queda ahi
                                carga_dir<=dir_salto;
                                else carga_dir<=copia_dir;
                                end if;--JgE
                    when "110"=>carga_dir<=dir_salto;--JMP
                    when others=>carga_dir<=copia_dir;--Se queda ahi
                                end case casosaltos;
                           cnt_in<=cnt_in+"001";
                    elsif(cnt_in="011") then
                           siguiente<='1';
                    end if;
                when puertos=>CTRL<='0';--Habilito puertos
               if(cnt_in="000")then--En el 1er Clock
                           copia_dir<=carga_dir;
                    if dat_in(6)='0' then-- Escojo Puerto
                           carga_dir(15)<='0';--Port1
                    else carga_dir(15)<='1';end if;--Port2
                    if(dat_in(4)='0') then--configuro Puerto
                           rwaux<='0';--In
                    else rwaux<='1';end if;--out
                    cnt_in<=cnt_in+"001";--Incr contador interno
               elsif(cnt_in<="001")then
                           if(copia_dat_in(4)='0') then--In
                           carga_a<=dat_in;
                           else carga_a<=a;end if;--Out
                                  carga_dir<=copia_dir;
                           siguiente<='1';
                           end if;
                       when logicos=>--En logicos
                                --Escojo Registro
                           if copia_dat_in(6)='0' then operandoR:=REG1;
                           elsif copia_dat_in(6)='1' then operandoR:=REG2;end if;
                                --Escojo operando1<-a;
                                operando1:=carga_a;
                                --Veo el tipo de operacion
                                case copia_dat_in(5 Downto 4) is
                                when "00"=>carga_a<=operando1 and operandoR;
                                When "01"=>carga_a<=NOT operando1;
                                when "10"=>carga_a<=operando1 or operandoR;
                                when others=>carga_a<=operando1 and operandoR;
                                end case;
                                cnt_in<=cnt_in+"001";
                           if(cnt_in="001")then
                                siguiente<='1';
                           end if;
                      when nop=>siguiente<='1';
                             end case;
       end case;
end if;
end process salidas;

regs_con_carga:
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

```
Process(clk,rst)
begin
if rst='1' then
       a<=(others=>'0');
       dir<=(others=>'0');
elsif clk='1' and clk'event then
       a<=carga_a;
       dir<=carga_dir;
end if;
end process regs_con_carga;
r_w<=rwaux;
CT<=CTRL;
--Triestado de salida;
dat_in<=dat;
dat_out<=carga_a;
DATO_ENTRA<=rwaux OR CTRL;
dat<=dat_out when DATO_ENTRA='0' else "ZZZZZZZZ";
end Descripcion;
```

Para simularlo
Cree un programa que hace lo siguiente:

```
Inicio:
; Direccion        Codigo         OPCODE
       0001:         Jmp 0004     01100100 Jmp
                                  00000100 04
                                  00000000 00
       0003:         db 0F        00001111 0F
       0004:         Ld 04        00000000 Ld
                                  00000011 03
                                  00000000 00
       0007:         nota         00010110 07
       0008:         nop          11111111 FF
       0009:         nop          11111111 FF
       000A:         Jmp 0000     01100100 Jmp
                                  00000000 00
                                  00000000 11
       000B~FFFF:nop              11111111 FF
```

Para nuestra simulacion tendriamos que ponerle en un process:
Dedir: Process DIR

```
BEGIN
       Case DIR is
       when "0000000000000000"=>Dat<="01100100";-- Jmp  --00
       when "0000000000000001"=>Dat<="00000100";--  04  --01
       when "0000000000000010"=>Dat<="00000000";--  00  --02
       when "0000000000000011"=>Dat<="00001111";--  0f  --03
       when "0000000000000100"=>Dat<="00000000";--  Ld  --04
       when "0000000000000101"=>Dat<="00000011";--  03  --05
       when "0000000000000110"=>Dat<="00000000";--  00  --06
       when "0000000000000111"=>Dat<="00010110";-- NotA --07
       when "0000000000001000"=>Dat<="11111111";--  nop --08
       when "0000000000001001"=>Dat<="11111111";--  nop --09
       when "0000000000001010"=>Dat<="01100100";--  00  --10
       when "0000000000001011"=>Dat<="00000000";--  00  --11
       when "0000000000001100"=>Dat<="00000000";--  00  --12
       end case
end process Dedir;
```

El codigo completo del archivo test Beanch es:

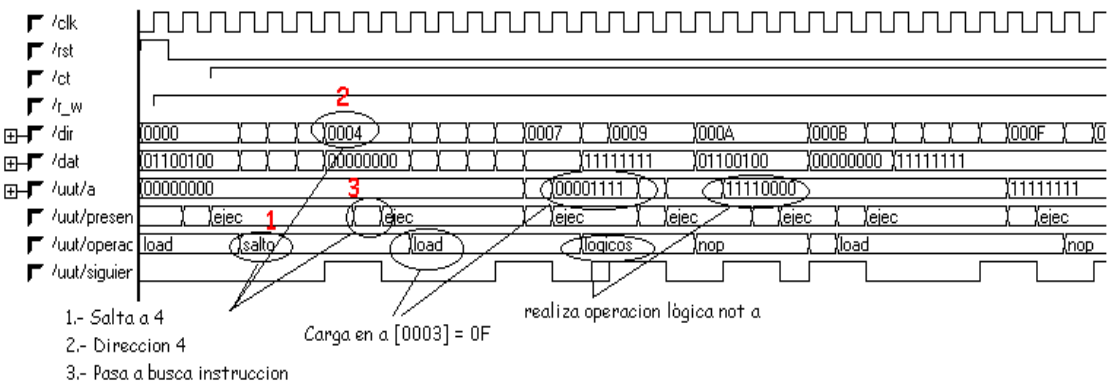
```
LIBRARY ieee;
USE ieee.numeric_std.ALL;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY testbench IS
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

```
END testbench;

ARCHITECTURE behavior OF testbench IS
	COMPONENT procesador
		PORT (
			clk : in  std_logic;
			rst : in  std_logic;
			CT : out  std_logic;
			R_W : out  std_logic;
			dir : out  std_logic_vector (15 DOWNTO 0);
			Dat : inout  std_logic_vector (7 DOWNTO 0)
		);
	END COMPONENT;

	SIGNAL clk : std_logic:='0';
	SIGNAL rst : std_logic;
	SIGNAL CT : std_logic;
	SIGNAL R_W : std_logic;
	SIGNAL dir : std_logic_vector (15 DOWNTO 0);
	SIGNAL Dat : std_logic_vector (7 DOWNTO 0);
BEGIN
-- Component Instantiation
	UUT : procesador
	PORT MAP (
		clk => clk,
		rst => rst,
		CT => CT,
		R_W => R_W,
		dir => dir,
		Dat => Dat
	);
--  Test Bench Statements
	clk<=not clk after 5 ms;
	rst<='1','0' after 10 ms;
	endir:Process(dir)
	begin
	case dir is
	when "0000000000000000"=>Dat<="01100100";-- Jmp  --00
	when "0000000000000001"=>Dat<="00000100";--  04  --01
	when "0000000000000010"=>Dat<="00000000";--  00  --02
	when "0000000000000011"=>Dat<="00001111";--  0f  --03
	when "0000000000000100"=>Dat<="00000000";--  Ld  --04
	when "0000000000000101"=>Dat<="00000011";--  03  --05
	when "0000000000000110"=>Dat<="00000000";--  00  --06
	when "0000000000000111"=>Dat<="00010110";-- NotA --07
	when "0000000000001000"=>Dat<="11111111";--  nop --08
	when "0000000000001001"=>Dat<="11111111";--  nop --09
	when "0000000000001010"=>Dat<="01100100";--  00  --10
	when "0000000000001011"=>Dat<="00000000";--  00  --11
	when "0000000000001100"=>Dat<="00000000";--  00  --12
	when others=>Dat<=(Others=>'1');
	end case;
	end process endir;

   tb : PROCESS
   BEGIN
		wait; -- will wait forever
   END PROCESS;
--  End Test Bench

END;
```

Y el resultado de la simulacion fue:

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com



Mi segunda codificaron fue:

PRO2.vhd

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

entity pro2 is
    Port (clk,rst:in std_logic;
                      CT,r_w:out std_logic;
                      dir:out std_logic_vector(15 downto 0);
                      dat:inout std_logic_vector(7 downto 0));
end pro2;

architecture Descripcion of pro2 is
type estado is
(inicial,busqueda,ejec,load,mover,arith,movebits,salto,compa,puertos,logicos,nop);
signal a,reg1,reg2,copia_dat_in,carga_a:std_logic_vector(7 downto 0);
signal dat_in,dat_out,copia_reg1:std_logic_vector(7 downto 0);
signal copia_dir,carga_dir:std_logic_vector(15 downto 0);
signal rwaux,ctrl,zero,carry:std_logic;
signal presente: estado;
signal DATO_ENTRA:std_logic;
signal siguiente:boolean;
signal cnt_in:std_logic_vector(2 downto 0);
begin
fsm:
        PROCESS(clk,rst)
variable copia_dat_in1:std_logic_vector (7 downto 0);
        BEGIN
                if rst='1' then--Si hay reset inicializo estados
                        presente<=inicial;
                elsif clk='1' and clk'event then--Para cada Clk
                        status: case presente is--Veo el estado presente
                        when inicial =>--si es inicial
                                presente<=busqueda;--Paso a estado de busqueda
                        when busqueda=>--si es busqueda
                                presente<=ejec;--Paso a estado de ejecucion
                        when ejec=>--Copia mi dato de entrada ante posibles cambios
                                        if(cnt_in="000") then
                                        copia_dat_in1:=dat;
                                        end if;
                ejecution: case copia_dat_in1(2 DOwNTO 0) is
                --Seleccione Tipo de operacion
                when "000"=>presente<=load;
                when "001"=>presente<=mover;
                when "010"=>presente<=arith;
                when "011"=>presente<=movebits;
                when "100"=>if(copia_dat_in1(3)='0') then--Si el tipo es salto
                                resente<=salto;
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

```
                        else--Si es comparacion
                                presente<=compa;
                                nd if;
                when "101"=>presente<=puertos;
                when "110"=>presente<=logicos;
                when others=>presente<=nop;
                end case ejecution;
                when others =>if siguiente then
                                presente<=busqueda;end if;
                end case status;
                end if;--Del Rst='1'
end process fsm;

salidas:process(presente,dat)
        variable bites:std_logic_vector(2 downto 0);
        variable tipo1:std_logic;
        variable operando1,operando2,operandoR:std_logic_vector(7 downto 0);
        variable dir_salto:std_logic_vector(15 downto 0);
begin
        case presente is
                when inicial =>carga_a<=(others=>'0');
                                                reg2<=(others=>'0');
                                                reg1<=(others=>'0');
                                        carga_dir<=(others=>'0');
                                                rwaux<='1';--Inicializo en lectura
                when busqueda=>cnt_in<="000";
                                --Inicializo contador interno de instruccion
                                rga_dir<=carga_dir+"0000000000000001";
                                rwaux<='1';--Habilito solo lectura
                                ctrl<='1';--Desabilito puertos
                                siguiente<=false;
                when ejec=>if (cnt_in="000")then
                                        --Guardo copia de dat_in por seguridad
                                        copia_dat_in<=dat;--_in;
                                        copia_dir<=carga_dir;end if;
                when load=>--Si es Load Durante el 1er Clock
                        if(cnt_in ="000")then
                        --guardo copia de Reg1 poque lo voy a modficar
                        copia_reg1<=reg1;
                        --Incr mi direccion de lectura de datos
                        carga_dir<=carga_dir+"0000000000000001";
                        --Incr mi contador interno
                        cnt_in<=cnt_in+"001";
                         elsif(cnt_in ="001") then--2do clock
                        reg1<=dat_in;--Guardo mi dato de entrada
                        -en Reg1, dat_in=>(A7 ... A0);
                        --Incr mi direccion de lectura de datos

                        dir=carga_dir+"0000000000000001";
                        --incr mi contador interno de instruccion

                        copia_dir<=carga_dir+"0000000000000001";--Guardo Dire inicial
                        cnt_in<=cnt_in+"001";
                        elsif(cnt_in ="010") then -- 3er Clock
                        carga_dir<=dat_in&reg1;--carga_dir <=(A15 ..A0)
                        --Configuro Modo Lectura o Escritura
                                if(copia_dat_in(4)='0')then--In
                                         rwaux<='1';--si es Ld a,XXX
                                else carga_a<=a;
                                         rwaux<='0';--si es Ld XXX,a
                                end if;
                        cnt_in<=cnt_in+"001";
                        elsif(cnt_in="011") then -- 4to clock
                        --Ejecuto operaciones
                                if(copia_dat_in(4)='0')then-- XXX->Acumulador
                                        carga_a<=dat_in;
                                else carga_a<=a;--Acumulador -> XXX
                                end if;
                        --Recupero datos cambiados al cargar
                                reg1<=copia_reg1;
                                carga_dir<=copia_dir;
                        siguiente<=true;--Habilito pasar a siguiente instruccion
                        end if;
                when mover=>--Si es Mov
                  if(dat_in(4)='0') then-- Tipo: Acumulador -> REG
                        if(dat_in(6)='0') then reg1<=a;else reg2<=a;end if;
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

```
                  else --      REG -> Acumulador
            if(dat_in(6)='0') then carga_a<=reg1;else carga_a<=reg2;end if;
       end if;
            siguiente<=true;
      when arith=>--//ALU
            if copia_dat_in(5 downto 4)="00" then -- A+REG
                  if copia_dat_in(6)='0' then--Escojo:
                        carga_a<=a+reg1;--Reg1
                  else   carga_a<=a+reg2;--Reg2
                  end if;
             elsif copia_dat_in(5 downto 4)="01" then-- A+REG+carry
                  if copia_dat_in(6)='0' then--Escojo
                        carga_a<=a+reg1+"0000000"&carry;--Reg1
                  else  carga_a<=a+reg2+"0000000"&carry;--Reg2
                                       end if;
             elsif copia_dat_in(5 downto 4)="10" then-- A - REG
                  if copia_dat_in(6)='0' then--Escojo
                        carga_a<=a-reg1;--Reg1
                  else  carga_a<=a-reg2;--Reg2
                  end if;
             elsif copia_dat_in(5 downto 4)="11" then -- A-(REG+carry)
            if copia_dat_in(6)='0' then--Escojo
            carga_a<=a-(reg1+"0000000"&carry);--Reg1
            else carga_a<=a-(reg2+"0000000"&carry); --Reg2
            end if;
            end if;
            siguiente<=true;
      when movebits=>--//manejo de Bits
             bites:=copia_dat_in(7 downto 5);
             case copia_dat_in(4 Downto 3) is
       when "00"=>--//rotar a la izq
            rorizq:For i in 0 to 7 loop
            if(i<=bites) then
            carga_a<=a(6)&a(5)&a(4)&a(3)&a(2)&a(1)&a(0)&a(7);
            end if;
            end loop rorizq;
       when "01"=>--//rotar a la Der
            rorder:For i in 0 to 7 loop
            if(i<=bites) then
            carga_a<=a(0)&a(7)&a(6)&a(5)&a(4)&a(3)&a(2)&a(1);
            end if;
            end loop rorder;
       when "10"=>--//Desp a la Der
            despizq:For i in 0 to 7 loop
            if(i<=bites) then
            carga_a<='0'&a(7)&a(6)&a(5)&a(4)&a(3)&a(2)&a(1);
            end if;
            end loop despizq;
       when others =>--//Desp a la Izq
            despder:For i in 0 to 7 loop
            if(i<=bites) then
            carga_a<=a(6)&a(5)&a(4)&a(3)&a(2)&a(1)&a(0)&'0';
            end if;
            end loop despder;
       end case;
       siguiente<=true;
  when compa=>if(dat_in(6)='0') then--Reg1
            operandoR:=Reg1;
            else operandoR:=Reg2; end if;
      case copia_dat_in(5 Downto 4) is--Tipo operando
      when "00" =>operando1:=operandoR;
                  operando2:=a;
      when "01" =>operando1:=reg1;
                  operando2:=reg2;
      when "10" =>operando1:=reg2;
                 operando2:=reg1;
      when others =>operando1:=a;
                operando2:=operandoR;
      end case;
      if(operando2 > operando1) then
                  carry<='1';
                  zero<='0';
      elsif(operando2 < operando1) then
                  carry<='0';
                  zero<='0';
      else zero<='1';end if;
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

```
                        siguiente<=true;
 when salto=>--Si es salto
       if(cnt_in="000") then--En el 1er Clk
       carga_dir<=carga_dir+"0000000000000001";
       copia_reg1<=reg1;
       cnt_in<=cnt_in+"001";

       elsif(cnt_in="001") then-- 2do clk
       carga_dir<=carga_dir+"0000000000000001";
       copia_dir<=carga_dir+"0000000000000001";
       cnt_in<=cnt_in+"001";

       elsif(cnt_in="010") then -- 3er Clk
       cnt_in<=cnt_in+"001";
       reg1<=dat_in;
       elsif(cnt_in="011")then
       cnt_in<=cnt_in+"001";

       casosaltos: case copia_dat_in(6 downto 4) is
       when "000"=>if(zero='1')then carga_dir<=dat_in&reg1;--JZ
              else carga_dir<=copia_dir;--Se queda ahi
              end if;
       when "001"=>if(zero='0')then carga_dir<=dat_in&reg1;--JNZ
              else carga_dir<=copia_dir;--Se queda ahi
              end if;
       when "010"=>if(carry ='0')then carga_dir<=dat_in&reg1;--Jl
              else carga_dir<=copia_dir;--Se queda ahi
              end if;
       when "011"=>if(carry='1')then carga_dir<=dat_in&reg1;--JG
              else carga_dir<=copia_dir;--Se queda ahi
              end if;
       when "100"=>tipo1:=(zero or carry) and carry;
              if(tipo1='1')then
              carga_dir<=dat_in&reg1;--Jle
              else carga_dir<=copia_dir;--Se queda ahi
              end if;
              when "101"=>tipo1:=(zero or carry) and carry;
              if(tipo1='0')then--Se queda ahi
              carga_dir<=dat_in&reg1;
              else carga_dir<=copia_dir;
              end if;--JgE
       when "110"=>carga_dir<=dat_in&reg1;--JMP
       when others=>carga_dir<=copia_dir;--Se queda ahi
       end case casosaltos;

       elsif(cnt_in="100") then
              reg1<=copia_reg1;
              siguiente<=true;
       cnt_in<=cnt_in+"001";
       end if;
  when puertos=>CTRL<='0';--Habilito puertos
       if(cnt_in="000")then--En el 1er Clock
       copia_dat_in<=dat_in;--Copia dato inicial
       copia_dir<=carga_dir;
       if dat_in(6)='0' then-- Escojo Puerto
       carga_dir(15)<='0';--Port1
       else carga_dir(15)<='1';end if;--Port2
       if(dat_in(4)='0') then--configuro Puerto
       rwaux<='0';--In
       else rwaux<='1';end if;--out
       cnt_in<=cnt_in+"001";--Incr contador interno
       elsif(cnt_in<="001")then
       if(copia_dat_in(4)='0') then--In
       carga_a<=dat_in;
       else carga_a<=a;end if;--Out
       carga_dir<=copia_dir;
       siguiente<=true;
       end if;
 when logicos=>--En logicos
       --Escojo Registro
       if copia_dat_in(6)='0' then operandoR:=REG1;
       elsif copia_dat_in(6)='1' then operandoR:=REG2;end if;
       --Escojo operando1<-a;
       operando1:=carga_a;
       --Veo el tipo de operacion
       case copia_dat_in(5 Downto 4) is
```

```
                when "00"=>carga_a<=operando1 and operandoR;
                When "01"=>carga_a<=NOT operando1;
                when "10"=>carga_a<=operando1 or operandoR;
                when others=>carga_a<=operando1 and operandoR;
                end case;
                cnt_in<=cnt_in+"001";
                        if(cnt_in="001")then
                                siguiente<=true;
                end if;
        when nop=>if(Cnt_in="000") then
                cnt_in<=cnt_in+"001";
                siguiente<=true;
                carga_dir<=copia_dir;
                end if;
        end case;
end process salidas;

regs_con_carga:
Process(clk,rst)
begin
if rst='1' then
        a<=(others=>'0');
        dir<=(others=>'0');
elsif clk='1' and clk'event then
        a<=carga_a;
        dir<=carga_dir;
end if;
end process regs_con_carga;
r_w<=rwaux;
CT<=CTRL;
--Triestado de salida;
dat_in<=dat;
dat_out<=carga_a;
DATO_ENTRA<=rwaux OR CTRL;
dat<=dat_out when DATO_ENTRA='0' else "ZZZZZZZZ";
end Descripcion;
```

Para la simulación de esta parte usaremos el siguiente codigo:
INICIO:

```
0001: Ld a,000F         00000000        00      00
                        00001111        0F      01
                        00000000        00      02
0003: SHL 4             10011011        9A      03
0004: Mov REG2,A        01010001        51      04
0005: Ld a,001F         00000000        00      05
                        00011111        1F      06
                        00000000        00      07
0008:SHL,2              01011011        5B      08
0009:CMP REG2,a         01001100        4C      09
000A:JZ 0000            00000100        04      10
                        00000000        00      11
                        00000000        00      12
000D:Not A              01010110        56      13
000E:Mov REG2,A         01010001        51      14
000F:Ld a,002F          00000000        00      15
                        00010111        0F      16
                        00000000        00      17
0012:ADDA REG2          01010001        51      18
0013:Mov REG2,A         01010001        51      19
0014:JMP 0000           00001100        14      20
                        00000000        00      21
                        00000000        00      22
0017~FFFF               11111111        FF      --
```

y el archivo de testBeanch completo seria:

```
LIBRARY ieee;
USE ieee.numeric_std.ALL;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY testbench IS
END testbench;

ARCHITECTURE behavior OF testbench IS
	COMPONENT pro2
		PORT (
			clk : in  std_logic;
			rst : in  std_logic;
			CT : out  std_logic;
			R_W : out  std_logic;
			dir : out  std_logic_vector (15 DOWNTO 0);
			Dat : inout  std_logic_vector (7 DOWNTO 0)
		);
	END COMPONENT;

	SIGNAL clk : std_logic:='0';
	SIGNAL rst : std_logic;
	SIGNAL CT : std_logic;
	SIGNAL R_W : std_logic;
	SIGNAL dir : std_logic_vector (15 DOWNTO 0);
	SIGNAL Dat : std_logic_vector (7 DOWNTO 0);
BEGIN

-- Component Instantiation
	UUT : pro2
	PORT MAP (
		clk => clk,
		rst => rst,
		CT => CT,
		R_W => R_W,
		dir => dir,
		Dat => Dat
	);
--  Test Bench Statements
	clk<=not clk after 5 ms;
	rst<='1','0' after 10 ms;
	endir:Process(dir)
	begin
	case dir is
	when "0000000000000000"=>Dat<="00000000";-- Ldaxx          --00 \
	when "0000000000000001"=>Dat<="00001111";-- 0F             --01  >Ld a,000F
	when "0000000000000010"=>Dat<="00000000";-- 00             --02 /
	when "0000000000000011"=>Dat<="10011011";-- SHL 4          --03
	when "0000000000000100"=>Dat<="01010001";-- Mov Reg2,a-04  --04
	when "0000000000000101"=>Dat<="00000000";-- ldaxx          --05  \
	when "0000000000000110"=>Dat<="00011111";-- 1F             --06   >Ld a,001F
	when "0000000000000111"=>Dat<="00000000";-- 00             --07  /
	when "0000000000001000"=>Dat<="01001000";-- SHL 2          --08
	when "0000000000001001"=>Dat<="01001100";-- CMP REG2,A     --09
	when "0000000000001010"=>Dat<="00000100";-- JZ             --10  \
	when "0000000000001011"=>Dat<="00000000";-- 00             --11   >Jz 0000
	when "0000000000001100"=>Dat<="00000000";-- 00             --12  /
	when "0000000000001101"=>Dat<="01010110";-- NOT A          --13
	when "0000000000001110"=>Dat<="01010001";-- Mov Reg2,a     --14
	when "0000000000001111"=>Dat<="00000000";-- Ldaxx          --15  \
	when "0000000000010000"=>Dat<="00101111";-- 2F             --16   >Ld a,002F
	when "0000000000010001"=>Dat<="00000000";-- 00             --17  /
	when "0000000000010010"=>Dat<="01000010";-- ADDA REG2      --18
	when "0000000000010011"=>Dat<="01010001";-- Mov Reg2,a     --19
	when "0000000000010100"=>Dat<="01100100";-- jmp            --20  \
	when "0000000000010101"=>Dat<="00000000";-- 00             --21   >Jmp 0000
	when "0000000000010110"=>Dat<="00000000";-- 00             --22  /
	when others=>Dat<=(Others=>'1');                           -- nop
	end case;
	end process endir;

   tb : PROCESS
   BEGIN
		wait; -- will wait forever
   END PROCESS;
--  End Test Bench

END;
```

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com

El resultado de esta simulacion es:

Queda Para que lo hagan ????

Elver Yoel OCMIN Grandez
yoelocmin@hotmail.com