

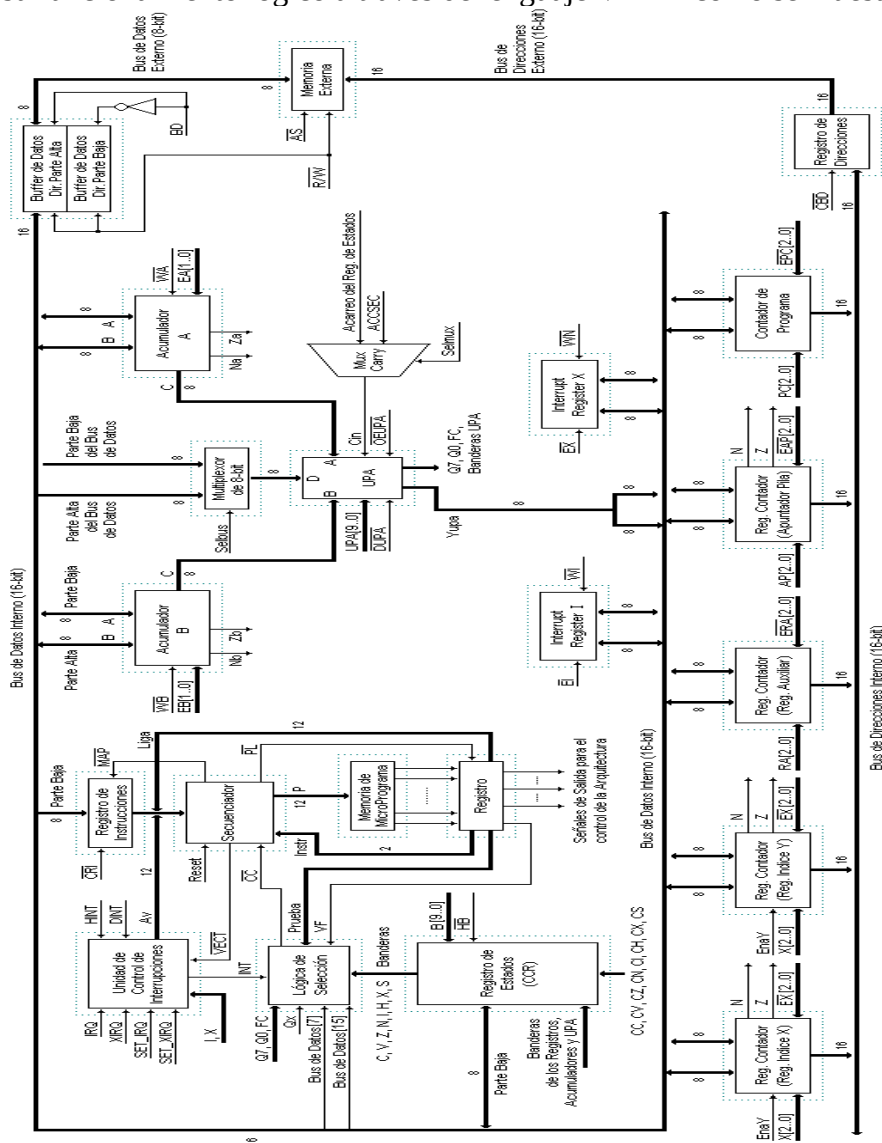
## Practica No. 6 Diseño de un Procesador CISC de 8 Bits

**Objetivo:** Diseñar un microprocesador CISC de 8 bits, específicamente un 'clón' del microprocesador 6811 de Motorola®

**Desarrollo:** Para cada uno de los siguientes apartados, realizar los diseños electrónicos que se piden.

**Duración:** 3 semanas.

1.- La figura 1 muestra el 'clon' del procesador 6811 como el que se vio en la teoría, instrumente la descripción de este Hardware utilizando el lenguaje VHDL. No es necesario instrumentar cada uno de los componentes como se muestran en esta figura, sino teniendo su funcionamiento lógico a través de lenguaje VHDL como se muestra en el apéndice A.



2.- Pruebe su arquitectura con el siguiente código de lenguaje del 6811:

```
LDAA #$FF
LDAB #$01
LDX  #$0010
ABA
BNE  ET1
STAA 0,X
BRA  ET2
ET1: STAB 0,X
ET2: LDAA #$07
LDAB #$02
MUL
STAA 1,X
STAB 2,X
FIN: BRA  FIN
```

Los valores iniciales de los acumuladores A y B pueden ser modificados de acuerdo a lo que le solicite el instructor del laboratorio. Muestre el resultado conectando las direcciones \$10, \$11 y \$12 a los 8 leds de la tarjeta, seleccione una de estas direcciones con los switches que contiene ésta.

3.- Ejecute los siguientes DRIVERS cuando ocurra una interrupción IRQ o XIRQ, estas son generadas por dos botones de la tarjeta de desarrollo.

```
DRIVER_X: LDX  #$0020
LDAA $0030
STAA 0,X
RTI
```

```
DRIVER_Y: LDX  #$0030
LDAB $0020
STAB 0,X
RTI
```

---

## A P E N D I C E   A

---

entity micro68HC11 is

```
Port (  
    clk: in STD_LOGIC;  
    reset: in STD_LOGIC;  
    nIRQ: in STD_LOGIC;  
    nXIRQ: in STD_LOGIC;  
    Data_in: in unsigned(7 downto 0);  
    Data_out: out unsigned(7 downto 0); -- Bus de datos de 8 bits  
    Dir: out unsigned(15 downto 0); -- Bis de direcciones de 16 bits  
    nRW: out STD_LOGIC:= '1'; -- Señal para escribir en memoria  
  
    -- Debug  
  
    PC_low_out: out unsigned(7 downto 0);  
    e_presente_out: out unsigned(7 downto 0);  
    A_out: out unsigned (7 downto 0);  
    B_out: out unsigned (7 downto 0);  
    X_low_out: out unsigned(7 downto 0);  
    X_high_out: out unsigned(7 downto 0);  
    Y_low_out: out unsigned(7 downto 0);  
    flags: out STD_LOGIC_VECTOR(7 downto 0)  
    -- S X H I N Z V C  
);
```

end micro68HC11;

architecture Behavioral of micro68HC11 is

```
signal e_presente: unsigned(11 downto 0) := X"000";  
signal e_siguiete: unsigned(11 downto 0);  
  
signal PC: unsigned (15 downto 0):= X"0014";  
  
signal estados: STD_LOGIC_VECTOR (7 downto 0):= X"FF";  
signal A: unsigned (7 downto 0);  
signal B: unsigned (7 downto 0);  
signal Q: unsigned (7 downto 0);  
signal Yupa: unsigned (7 downto 0);  
signal XH: unsigned (7 downto 0);  
signal XL: unsigned (7 downto 0);  
signal YH: unsigned (7 downto 0);  
signal YL: unsigned (7 downto 0);  
signal AuxH: unsigned (7 downto 0);  
signal AuxL: unsigned (7 downto 0);  
signal Aux: unsigned (15 downto 0);  
signal PCH: unsigned (7 downto 0) := X"00";  
signal PCL: unsigned (7 downto 0) := X"14";  
signal SPH: unsigned (7 downto 0) := X"FF"; -- Definir en qué lugar poner el stack...  
signal SPL: unsigned (7 downto 0) := X"FF"; -- de qué tamaño es la memoria y ponerlo  
    -- en la última dirección  
  
signal microI: unsigned (11 downto 0) := X"333"; -- Direccion del driver de I := X""
```

```
signal microX: unsigned (11 downto 0) := X"444" ; -- Direccion del driver de I := X""
```

```
signal IntRI: unsigned (15 downto 0);  
signal IntrRX: unsigned (15 downto 0);
```

```
signal IRQ: STD_LOGIC := '0';  
signal XIRQ: STD_LOGIC := '0';
```

```
signal startMUL: STD_LOGIC := '0';  
constant ZERO : unsigned (7 downto 0) := "00000000" ;  
signal D: unsigned (15 downto 0);  
signal varRW: STD_LOGIC := '1';  
signal indY: STD_LOGIC := '0';
```

```
begin
```

```
process(clk, reset,e_presente,e_siguiete)  
begin
```

```
if (reset = '0') then  
    e_siguiete <= X"000";  
    PC <= X"0014";  
    IRQ <= '0';  
    XIRQ <= '0';  
    indY <= '0';
```

```
else
```

```
if (rising_edge(clk)) then  
    case e_presente is
```

```
        when X"000" =>  
            Dir <= PC;  
            e_siguiete <= X"001";
```

```
        when X"001" =>  
            PC <= PC + 1;  
            e_siguiete <= e_presente + 1;
```

```
        when X"002" =>  
            e_siguiete <= (Data_in & ZERO(3 downto 0));
```

```
-----  
        when X"860" => -- LDAA IMM  
            Dir <= PC;  
            e_siguiete <= e_presente + 1;
```

```
        when X"861" => -- LDAA  
            PC <= PC + 1;  
            e_siguiete <= e_presente + 1;
```

```
        when X"862" => -- LDAA  
            A <= Data_in;  
            -- Actualiza N  
            estados(3) <= Data_in(7);
```

```
            -- Actualiza Z  
            if(Data_in = ZERO) then
```

```
                estados(2) <= '1';
```

```
            else  
                estados(2) <= '0';
```

```
            end if;
```

```
            -- Actualiza V  
            estados(1) <= '0';
```

```
if (XIRQ = '1') then
    e_siguiete <= microX;
else
    if (IRQ = '1') then
        e_siguiete <= microI;
    else
        Dir <= PC;
        e_siguiete <= X"001";
    end if;
end if;
```

---

```
when X"C60" => -- LDAB
    Dir <= PC;
    e_siguiete <= e_presente + 1;
when X"C61" => -- LDAB
    PC <= PC + 1;

    e_siguiete <= e_presente + 1;

when X"C62" => -- LDAB
    B <= Data_in;

    -- Actualiza N
    estados(3) <= Data_in(7);

    -- Actualiza Z
    if(Data_in = ZERO) then
        estados(2) <= '1';
    else
        estados(2) <= '0';
    end if;

    -- Actualiza V
    estados(1) <= '0';

    if (XIRQ = '1') then
        e_siguiete <= microX;
    else
        if (IRQ = '1') then
            e_siguiete <= microI;
        else
            Dir <= PC;
            e_siguiete <= X"001";
        end if;
    end if;
```

---

```
-- Código de la instrucción de acceso relativo BNE
when X"260" =>
    Dir <= PC;
    e_siguiete <= e_presente + 1;
when X"261" =>
    PC <= PC + 1;
    e_siguiete <= e_presente + 1;
when X"262" =>
    if(estados(2)='0') then
        if (Data_in(7) = '1') then
```

```

PC <= PC - unsigned(not(Data_in-1));

else
    PC <= PC + Data_in;
end if;
end if;
e_siguiete <= e_presente + 1;
when X"263" =>
    if (XIRQ = '1') then
        e_siguiete <= microX;
    else
        if (IRQ = '1') then
            e_siguiete <= microI;
        else
            Dir <= PC;
            e_siguiete <= X"001";
        end if;
    end if;
end if;

-----
--          .
--          .
--          .
-----

when others =>
    e_siguiete <= X"000";
    PC <= X"0000";

end case;

end if;
end if;
e_presente <= e_siguiete;

-- debug vals
A_out<=A;
B_out<=B;
e_presente_out<=e_presente(11 downto 4);
PC_low_out <= PC(7 downto 0);
X_low_out <= XL;
X_high_out <= XH;
Y_low_out <= YL;
flags <= estados;
end process;

```