Código de implementación de cada uno de los elementos

FunCode:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity MfunCode is
   generic (
      instruccion : natural := 20;
      tamMemoria : natural := 16
   );
   Port ( funCode : in STD_LOGIC_VECTOR (3 downto 0);
   outMFCode: out STD_LOGIC_VECTOR (19 downto 0));
end MfunCode;
architecture Behavioral of MfunCode is
type memoria is array (tamMemoria-1 downto 0) of
STD_LOGIC_VECTOR(instruccion-1 downto 0);
constant mem : memoria :=(
   0 => "00000100010000110011",
   1 => "00000100010001110011",
   2 => "00000100010000000011",
   3 => "00000100010000010011",
   4 => "00000100010000100011",
   5 => "00000100010011010011",
   6 => "00000100010011000011",
   7 => "00000100010010100011",
   8 => "00000100010011010011",
   9 => "00000001110000000000",
   10 => "00000001010000000000",
   others => (others=>'0')
   );
begin
   outMFCode <= mem(TO_INTEGER(UNSIGNED(funcode)));
end Behavioral;
```

Decodificador de instrucción:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity decodificador is
   Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
          tipoR, BEQI, BNEI, BLTI, BLETI, BGTI, BGETI : out STD_LOGIC);
end decodificador;
architecture Behavioral of decodificador is
begin
   process(opCode) is
   begin
      if(opCode = "00000") then
         tipoR <= '1';
         BEQI <= '0';
         BNEI <= '0';
         BLTI <= '0';
         BLETI <= '0';
         BGTI <= '0';
         BGETI <= '0';
      elsif(opCode = "01101") then
         tipoR <= '0';
```

```
            BEQI <= '1';
            BNEI <= '0';
            BLTI <= '0';
            BLETI <= '0';
            BGTI <= '0';
            BGETI <= '0';
        elsif(opCode = "01110") then
            tipoR <= '0';
            BEQI <= '0';
            BNEI <= '1';
            BLTI <= '0';
            BLETI <= '0';
            BGTI <= '0';
            BGETI <= '0';
        elsif(opCode = "01111") then
            tipoR <= '0';
            BEQI <= '0';
            BNEI <= '0';
            BLTI <= '1';
            BLETI <= '0';
            BGTI <= '0';
            BGETI <= '0';
        elsif(opCode = "10000") then
            tipoR <= '0';
            BEQI <= '0';
            BNEI <= '0';
            BLTI <= '0';
            BLETI <= '1';
            BGTI <= '0';
            BGETI <= '0';
        elsif(opCode = "10001") then
            tipoR <= '0';
            BEQI <= '0';
            BNEI <= '0';
            BLTI <= '0';
            BLETI <= '0';
            BGTI <= '1';
            BGETI <= '0';
        elsif(opCode = "10010") then
            tipoR <= '0';
            BEQI <= '0';
            BNEI <= '0';
            BLTI <= '0';
            BLETI <= '0';
            BGTI <= '0';
            BGETI <= '1';
            else
            tipoR <= '0';
            BEQI <= '0';
            BNEI <= '0';
            BLTI <= '0';
            BLETI <= '0';
            BGTI <= '0';
            BGETI <= '0';
        end if;
end process;
```

MUX SDOPC:

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

entity sdopcMux is
   Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
          SDOPC : in STD_LOGIC;
          muxUOut : out STD_LOGIC_VECTOR (4 downto 0));
end sdopcMux;

architecture Behavioral of sdopcMux is

begin
   muxUOut <= "00000" when SDOPC ='0'else
   opCode;

end Behavioral;
```

OPCODE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
entity MopCode is
   Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
          outOpCode : out STD_LOGIC_VECTOR (19 downto 0));
end MopCode;

architecture Behavioral of MopCode is
type memoria is array (31 downto 0) of
STD_LOGIC_VECTOR(19 downto 0);
constant mem : memoria :=(
   0 => "00001000000001110001",
   1 => "00000000010000000000",
   2 => "00000100010000001000",
   3 => "00001000000000001100",
   4 => "00001010000100110101",
   5 => "00000100010100110011",
   6 => "00000100010101110011",
   7 => "00000100010100000011",
   8 => "00000100010100010011",
   9 => "00000100010100100011",
   10 => "00000100010111010011",
   11 => "00000100000111000011",
   12 => "00000100000110100011",
   13 => "10010000001100110011",
   14 => "10010000001100110011",
   15 => "10010000001100110011",
   16 => "10010000001100110011",
   17 => "10010000001100110011",
   18 => "10010000001100110011",
   19 => "00010000000000000000",
   20 => "01010000000000000000",
   21 => "00100000000000000000",
   22 => "00000000000000000000",
   23 => "00000110010100110001",
   others => (others => '0')
   );
```

```
begin
   outOpCode <= mem(TO_INTEGER(UNSIGNED(opCode)));
end Behavioral;
```

MUX SM:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity smMux is
   Port ( MfunCode : in STD_LOGIC_VECTOR (19 downto 0);
         MopCode : in STD_LOGIC_VECTOR (19 downto 0);
         microinstruccion : out STD_LOGIC_VECTOR (19 downto 0);
         SM : in STD_LOGIC);
end smMux;

architecture Behavioral of smMux is

begin
   microinstruccion <= MfunCode when SM ='0'else
   MopCode;
end Behavioral;
```

Bloque de nivel:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity bloqueNivel is
   Port ( clk,clr : in STD_LOGIC;
         NA : out STD_LOGIC);
end bloqueNivel;

architecture Behavioral of bloqueNivel is
SIGNAL pclk : STD_LOGIC;
SIGNAL nclk : STD_LOGIC;
begin

ALTO : process (clk, clr)
begin
   if(clr = '1') then
      pclk <= '0';
   elsif(rising_edge(clk)) then
      pclk <= NOT (pclk);
   end if;
end process;

BAJO : process(clk ,clr)
begin
   if(clr = '1') then
      nclk <= '0';
   elsif(falling_edge(clk)) then
      nclk <= NOT (nclk);
   end if;
end process;

NA <= nclk XOR pclk;
end Behavioral;
```

Registro de banderas:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity capturaBanderas is
    Port ( LF, clk, clr : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (3 downto 0);
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end capturaBanderas;

architecture Behavioral of capturaBanderas is

begin
process (clk,clr)
begin
    if(clr = '1') then
        Q <= "0000";
    elsif(falling_edge(clk)) then
        Q <= D;
    end if;
end process;

end Behavioral;
```

Bloque condición:

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

entity bloqueCondicion is
    Port ( q : in STD_LOGIC_VECTOR (3 downto 0);
           EQ, NE, LT, LE, GT, GE : out STD_LOGIC);
end bloqueCondicion;

architecture Behavioral of bloqueCondicion is

begin

    EQ <= q(2);
    NE <= not(q(2));
    LT <= not(q(1));
    LE <= q(2) OR NOT(q(1));
    GT <= NOT(q(2))AND(q(1));
    GE <= q(1);

end Behavioral;
```

Unidad de control:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity unidadControl is
    Port ( clk, clr,tipoR, BEQI, BNEI, BLTI,BLETI,BGTI, BGETI, EQ, NE, LT, LE, GT, GE, NA : in
STD_LOGIC;
           SDOPC, SM : out STD_LOGIC);
end unidadControl;

architecture Behavioral of unidadControl is
```

```vhdl
type estados is (e0);
SIGNAL edo_act, edo_sig : estados;
begin

process(clr,clk)
begin
    if(clr = '1') then
        edo_act <= e0;
    elsif(rising_edge(clk)) then
        edo_act <= edo_sig;
    end if;
end process;

UC : process(edo_act,tipoR, BEQI, BNEI, BLTI,BLETI,BGTI, BGETI, EQ, NE, LT, LE, GT, GE, NA)
begin
    SDOPC <= '0';
    SM <= '0';

    if(tipoR = '1') then
    SDOPC <= '0';
    SM <= '0';
    edo_sig <= e0;
    --**BLOQUE 1
    --
    elsif(BEQI = '1') then
        if (NA = '1') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
        elsif(EQ = '0') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
        else
            SDOPC <= '1';
            SM <= '1';
            edo_sig <= e0;
        end if;
    --
    elsif(BNEI = '1') then
        if(NA = '1') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
        elsif(NE = '0') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
            else
                SDOPC <= '1';
                SM <= '1';
                edo_sig <= e0;
        end if;
    --
    --
    elsif(BLTI = '1') then
        if(NA = '1') then
        SDOPC <= '0';
        SM <= '1';
        edo_sig <= e0;
        elsif(LT = '0') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
            else
                SDOPC <= '1';
                SM <= '1';
                edo_sig <= e0;
        end if;
    --
    --
    elsif(BLETI = '1') then
        if(NA = '1') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
         elsif(LE = '0') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
            else
                SDOPC <= '1';
                SM <= '1';
                edo_sig <= e0;
        end if;

    elsif(BGTI = '1') then
        if(NA = '1') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
        elsif(GT = '0') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
            else
                SDOPC <= '1';
                SM <= '1';
                edo_sig <= e0;
        end if;

    elsif(BGETI = '1') then
        if(NA = '1') then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
        elsif(GE = '0')then
            SDOPC <= '0';
            SM <= '1';
            edo_sig <= e0;
            else
                SDOPC <= '1';
                SM <= '1';
                edo_sig <= e0;
         end if;

    else
            SDOPC <= '1';
            SM <= '1';
            edo_sig <= e0;
    end if;
end process;
end Behavioral;
```

Código de implementación de la arquitectura completa

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use work.paquete.all;
entity ucComp is
 Port (funCode, banderas : in STD_LOGIC_VECTOR (3 downto 0 );
 opCode : in STD_LOGIC_VECTOR (4 downto 0);
 clk , clr, lf : in STD_LOGIC;
 microinstruccion : out STD_LOGIC_VECTOR (19 downto 0)
 );
end ucComp;

architecture Behavioral of ucComp is

SIGNAL outMfunCode : STD_LOGIC_VECTOR (19 downto 0);
--**MfunCode

SIGNAL outTipoR, outBEQI, outBNEI, outBLTI, outBLETI, outBGTI, outBGETI : STD_LOGIC;
--**DECODIFICADOR

SIGNAL outSDOPC,outSM : STD_LOGIC;
--**UNIDAD DE CONTROL

SIGNAL outMuxSDOPC : std_logic_VECTOR (4 downto 0);
--**MUXES

SIGNAL outMopCode : std_logic_VECTOR (19 downto 0);
--**MOPCODE

SIGNAL outNivel : std_logic;
--**BLOQUE DE NIVEL

SIGNAL outEQ , outNE, outLT, outLE, outGT, outGE : std_logic;

SIGNAL inLF : std_logic;
SIGNAL outMicroinstruccion : std_logic_vector (19 downto 0);
SIGNAL outQ : std_logic_vector (3 downto 0);
--**LECTOR BANDERAS
begin
MfunCod : MfunCode
    Port map (
    funCode => funCode,
    outMFCode => outMfunCode
    );

decodificadorInstruccion : decodificador
    Port map (
        opCode => opCode,
        tipoR => outTipoR,
        BEQI => outBEQI,
        BNEI => outBNEI,
        BLTI => outBLTI,
        BLETI => outBLETI,
        BGTI => outBGTI,
        BGETI => outBGETI
    );

muxSDOPC : sdopcMux
    Port map(
        opCode => opCode,
        SDOPC => outSDOPC,
        muxUOut => outMuxSDOPC
    );

MopCod : MopCode
    Port map(
       opCode => outMuxSDOPC,
       outOpCode => outMopCode
    );

muxSM : smMux
    Port map(
        mFunCode => outMfunCode,
        MopCode => outMopCode,
        SM => outSM,
        microinstruccion => outMicroinstruccion
    );
    microinstruccion <= outMicroinstruccion;

bloqueNive : bloqueNivel
    Port map(
        clk => clk,
        clr => clr,
        NA => outNivel
    );

registroBanderas : capturaBanderas
    Port map (
        clk => clk,
        clr => clr,
        D => banderas,
        LF => lf,
        Q => outQ
    );

bloqueCondicio : bloqueCondicion
    Port map(
        q => outQ,
        EQ => outEQ,
        NE => outNE,
        LT => outLT,
        LE => outLE,
        GT => outGT,
        GE => outGE
    );

unidadDeControl : unidadControl
    Port map(
        clk => clk,
        clr => clr,
        tipoR => outTipoR,
        BEQI => outBEQI,
        BNEI => outBNEI,
        BLTI => outBLTI,
        BLETI => outBLETI,
        BGTI => outBGTI,
        BGETI => outBGETI,
        EQ => outEQ,
        NE => outNE,
        LT => outLT,
        LE => outLE,
        GT => outGT,
        GE => outGE,
        NA => outNivel,
        SDOPC => outSDOPC,
        SM => outSM
    );

end Behavioral;
```

Código de simulación de cada uno de los elementos

Fun code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_MfunCode is
--  Port ( );
end TB_MfunCode;

architecture Behavioral of TB_MfunCode is
component MfunCode is
    Port ( funCode : in STD_LOGIC_VECTOR (3 downto 0);
    outMFCode: out STD_LOGIC_VECTOR (19 downto 0));
end component;
SIGNAL funCode : STD_LOGIC_VECTOR (3 downto 0);
SIGNAL outMFCode : STD_LOGIC_VECTOR (19 downto 0);
begin

u1 : MfunCode
    Port Map(
        funCode => funCode,
        outMFCode => outMFCode
    );

stimulus : process
begin
funCode <= "0000";
wait for 10 ns;
funCode <= "0001";
wait for 10 ns;
funCode <= "0010";
wait for 10 ns;
funCode <= "0011";
wait for 10 ns;
funCode <= "0100";
wait for 10 ns;
funCode <= "0101";
wait for 10 ns;
funCode <= "0110";
```

```
wait for 10 ns;
funCode <= "0111";
wait for 10 ns;
funCode <= "1000";
wait for 10 ns;
funCode <= "1001";
wait for 10 ns;
funCode <= "1010";
wait for 10 ns;

wait;
end process;

end Behavioral;
```

Op code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_MopCode is
--  Port ( );
end TB_MopCode;

architecture Behavioral of TB_MopCode is
component MopCode is
   Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
          outOpCode : out STD_LOGIC_VECTOR (19 downto 0));
end component;
SIGNAL opCode : STD_LOGIC_VECTOR (4 downto 0);
SIGNAL outOpCode : STD_LOGIC_VECTOR (19 downto 0);
begin
u2 : MopCode
   Port map(
   opCode => opCode,
   outOpCode => outOpCode
   );

stimulus : process
begin
opCode <= "00001";
wait for 10 ns;
opCode <= "00010";
wait for 10 ns;
opCode <= "00011";
wait for 10 ns;
opCode <= "00100";
wait for 10 ns;
opCode <= "00101";
wait for 10 ns;
opCode <= "00110";
wait for 10 ns;
opCode <= "00111";
wait for 10 ns;
opCode <= "01000";
wait for 10 ns;
opCode <= "01001";
wait for 10 ns;
opCode <= "01010";
wait for 10 ns;
```

```
opCode <= "01011";
wait for 10 ns;
opCode <= "01100";
wait for 10 ns;
opCode <= "01101";
wait for 10 ns;
opCode <= "01111";
wait for 10 ns;
opCode <= "10000";
wait for 10 ns;
opCode <= "10001";
wait for 10 ns;
opCode <= "10010";
wait for 10 ns;
opCode <= "10011";
wait for 10 ns;
opCode <= "10100";
wait for 10 ns;
opCode <= "10101";
wait for 10 ns;
opCode <= "10110";
wait for 10 ns;
opCode <= "10111";
wait for 10 ns;
opCode <= "11000";
wait for 10 ns;
wait;
end process;

end Behavioral;
```

Bloque de condición:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_bloqueCondicion is
--  Port ( );
end TB_bloqueCondicion;

architecture Behavioral of TB_bloqueCondicion is
component bloqueCondicion is
   Port ( q : in STD_LOGIC_VECTOR (3 downto 0);
          EQ, NE, LT, LE, GT, GE : out STD_LOGIC);
end component;
SIGNAL q : STD_LOGIC_VECTOR (3 downto 0);
SIGNAL EQ, NE, LT, LE, GT, GE :  STD_LOGIC;
begin
u4 : bloqueCondicion
   Port map(
      q => q,
      EQ => EQ,
      NE => NE,
      LT => LT,
      LE => LE,
      GT => GT,
      GE => GE
   );
stimulus : process
begin
q <= "0100";
```

```
wait for 10 ns;
q <= "1011";
wait for 10 ns;
q <= "1111";
wait for 10 ns;
wait;
end process;
end Behavioral;
```

Bloque de nivel:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_bloqueNivel is
--  Port ( );
end TB_bloqueNivel;

architecture Behavioral of TB_bloqueNivel is
component bloqueNivel is
    Port ( clk,clr : in STD_LOGIC;
           NA : out STD_LOGIC);
end component;
SIGNAL clk, clr :  STD_LOGIC;
SIGNAL NA : STD_LOGIC;
constant CLK_period : time := 10ns;

begin
u5 : bloqueNivel
    Port map(
        clk => clk,
        clr => clr,
        NA => NA
    );
CLK_process : process
begin
    CLK <= '0';
    wait for CLK_period/2;
    CLK <= '1';
    wait for CLK_period/2;
end process;

stimulus : process
begin
    wait until rising_edge(clk);
    clr <= '1';
    wait for 10 ns;
    clr <= '0';
    wait;
end process;

end Behavioral;
```

Decodificador:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_decodificador is
--  Port ( );
end TB_decodificador;

architecture Behavioral of TB_decodificador is
component decodificador is
   Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
          tipoR, BEQI, BNEI, BLTI, BLETI, BGTI, BGETI : out STD_LOGIC);
end component;
SIGNAL opCode : STD_LOGIC_VECTOR (4 downto 0);
SIGNAL tipoR, BEQI, BNEI, BLTI, BLETI, BGTI, BGETI : STD_LOGIC;
begin
u3 : decodificador
   Port map(
   opCode => opCode,
   tipoR => tipoR,
   BEQI => BEQI,
   BNEI => BNEI,
   BLTI => BLTI,
   BLETI => BLETI,
   BGTI => BGTI,
   BGETI => BGETI
   );

stimulus : process
begin
   opCode <= "00000";
   wait for 10 ns;
   opCode <= "01101";
   wait for 10 ns;
   opCode <= "01110";
   wait for 10 ns;
   opCode <= "01111";
   wait for 10 ns;
   opCode <= "10000";
   wait for 10 ns;
   opCode <= "10001";
   wait for 10 ns;
   opCode <= "10010";
   wait for 10 ns;
   opCode <= "00001";
   wait for 10 ns;
   wait;
end process;

end Behavioral;
```

Unidad de control:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_unidadControl is
--  Port ( );
end TB_unidadControl;

architecture Behavioral of TB_unidadControl is
```

```
component unidadControl is
   Port ( clk, clr,tipoR, BEQI, BNEI, BLTI,BLETI,BGTI, BGETI, EQ, NE, LT, LE, GT, GE, NA : in
STD_LOGIC;
         SDOPC, SM : out STD_LOGIC);
end component;

signal clk, clr,tipoR, BEQI, BNEI, BLTI,BLETI,BGTI, BGETI, EQ, NE, LT, LE, GT, GE, NA :
STD_LOGIC := '0';
SIGNAL SDOPC, SM : STD_LOGIC;
constant CLK_period : time := 10ns;
begin
u6 : unidadControl
   Port map(
       clk => clk,
       clr => clr,
       tipoR => tipoR,
       BEQI => BEQI,
       BNEI => BNEI,
       BLTI => BLTI,
       BLETI => BLETI,
       BGTI => BGTI,
       BGETI => BGETI,
       EQ => EQ,
       NE => NE,
       LT => LT,
       LE => LE,
       GT => GT,
       NA => NA,
       GE => GE,
       SDOPC => SDOPC,
       SM => SM
       );

CLK_process : process
begin
   CLK <= '0';
   wait for CLK_period/2;
   CLK <= '1';
   wait for CLK_period/2;
end process;
stimulus : process
begin
wait until rising_edge(clk);
   clr <= '1';
wait for 10 ns;
   clr <= '0';
wait for 10 ns;
   BEQI <= '0';
   EQ <= '0';
      tipoR <= '0';
       BNEI <= '1';
       BLTI <= '0';
       BLETI <= '0';
       BGTI <= '0';
       BGETI <= '0';
       NE <= '1';
       LT <= '0';
       LE <= '0';
       GT <= '0';
       NA <= '0';
```

```
        GE <= '0';
wait for 10 ns;
            BEQI <= '0';
    EQ <= '0';
       tipoR <= '0';
        BNEI <= '1';
        BLTI <= '0';
        BLETI <= '0';
        BGTI <= '0';
        BGETI <= '0';
        NE <= '1';
        LT <= '0';
        LE <= '0';
        GT <= '0';
        NA <= '1';
        GE <= '0';
wait for 10 ns;
wait;
end process;

end Behavioral;
```

Código de simulación de la arquitectura completa

Unidad de control:

```
LIBRARY IEEE;
LIBRARY STD;
USE STD.TEXTIO.ALL;
USE ieee.std_logic_TEXTIO.ALL;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_UNSIGNED.ALL;
USE ieee.std_logic_ARITH.ALL;
entity TB_UCCompleta is
end TB_UCCompleta;

architecture Behavioral of TB_UCCompleta is
component ucComp is
 Port (funCode, banderas : in STD_LOGIC_VECTOR (3 downto 0 );
 opCode : in STD_LOGIC_VECTOR (4 downto 0);
 clk , clr, lf : in STD_LOGIC;
 microinstruccion : out STD_LOGIC_VECTOR (19 downto 0)
 );
end component;

SIGNAL funCode, banderas : STD_LOGIC_VECTOR (3 downto 0 ):="0000";
SIGNAL opCode : STD_LOGIC_VECTOR (4 downto 0):="00000";
SIGNAL clk , clr, lf : STD_LOGIC := '0';
SIGNAL microinstruccion : STD_LOGIC_VECTOR (19 downto 0):="00000000000000000000";
constant CLK_period : time := 100 ns;
begin
UnidadControlr : ucComp
    Port map(
        funCode => funCode,
        banderas => banderas,
        opCode => opCode,
        clk => clk,
        clr => clr,
        lf => lf,
        microinstruccion => microinstruccion
    );

process
begin
    clk <= '0';
    wait for 5 ns;
    clk <= '1';
    wait for 5 ns;
end process;
    stim_process : process
    file ARCH_RES : TEXT;
    variable LINEA_RES : line;
    variable var_funCode, var_banderas : STD_LOGIC_VECTOR (3 downto 0);
    variable  var_opCode : STD_LOGIC_VECTOR (4 downto 0);
    variable var_clr, var_lf : STD_LOGIC;

    file ARCH_VEC : TEXT;
    variable LINEA_VEC : line;
    VARIABLE CADENA : STRING(1 TO 10);
            VARIABLE CADENA2 : STRING(1 TO 16);
            VARIABLE NIVEL : STRING(1 to 4);

    variable var_microinstruccion : STD_LOGIC_VECTOR (19 downto 0);
 begin
                    file_open(ARCH_VEC,
"C:\Users\sebas\Downloads\PracticasArq\unidadControl\vectores.txt", READ_MODE);
                    file_open(ARCH_RES,
"C:\Users\sebas\Downloads\PracticasArq\unidadControl\resultado.txt", WRITE_MODE);
        CADENA:="OP_CODE   ";
        write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
        CADENA:="FUN_CODE  ";
        write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
        CADENA:="BANDERAS  ";
        write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
        CADENA:="CLR       ";
        write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
        CADENA:="LF        ";
        write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
        CADENA2:="MICROINSTRUCCION";
        write(LINEA_RES, CADENA2, left, CADENA2'LENGTH+1);
        CADENA:="   NIVEL  ";
        write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
        writeline(ARCH_RES,LINEA_RES);
        wait for 100ns;
    --**Encabezado archivo de salida
                    WAIT FOR 100 NS;
                    FOR I IN 0 TO 51 LOOP
                            readline(ARCH_VEC,LINEA_VEC);
                            read(LINEA_VEC, var_opCode);
            opCode <= var_opCode;
            read(LINEA_VEC, var_funCode);
            funCode <= var_funCode;
            read(LINEA_VEC, var_banderas);
            banderas <= var_banderas;
            read(LINEA_VEC, var_clr);
            clr <= var_clr;
            read(LINEA_VEC, var_lf);
            lf <= var_lf;
```

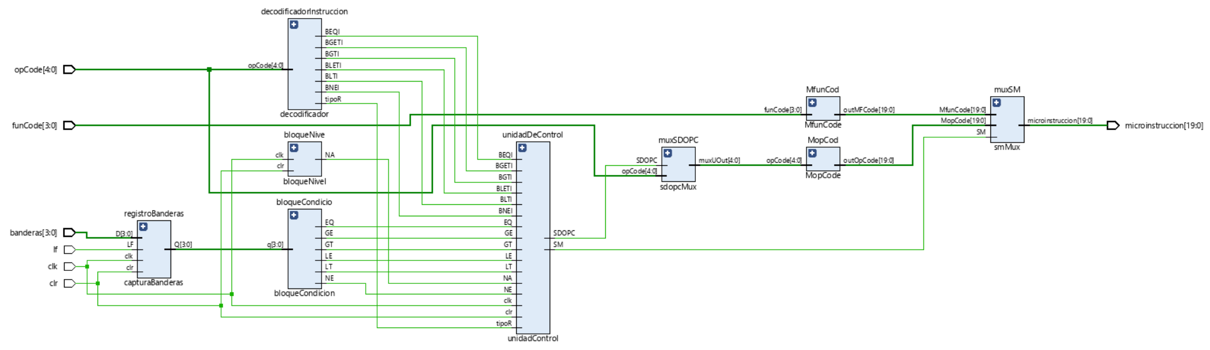
```
                              WAIT UNTIL RISING_EDGE(CLK);
            var_microinstruccion := microinstruccion;
            write(LINEA_RES, var_opCode, left, 11);
            write(LINEA_RES, var_funCode, left, 11);
            write(LINEA_RES, var_banderas, left, 11);
            write(LINEA_RES, var_clr, left, 11);
            write(LINEA_RES, var_lf, left, 9);
            write(LINEA_RES, var_microinstruccion, left, 22);
            NIVEL := "ALTO";
            write(LINEA_RES, NIVEL, left, 8);
                              writeline(ARCH_RES,LINEA_RES);
                              wait until falling_edge(clk);
                              var_microinstruccion := microinstruccion;
            write(LINEA_RES, var_opCode, left, 11);
            write(LINEA_RES, var_funCode, left, 11);
            write(LINEA_RES, var_banderas, left, 11);
            write(LINEA_RES, var_clr, left, 11);
            write(LINEA_RES, var_lf, left, 9);
            write(LINEA_RES, var_microinstruccion, left, 22);
            NIVEL := "BAJO";
            write(LINEA_RES, NIVEL, left, 8);
                              writeline(ARCH_RES,LINEA_RES);

                        end loop;
                        file_close(ARCH_VEC);
                        file_close(ARCH_RES);
      wait;
   end process;
end Behavioral;
```

Paquete:

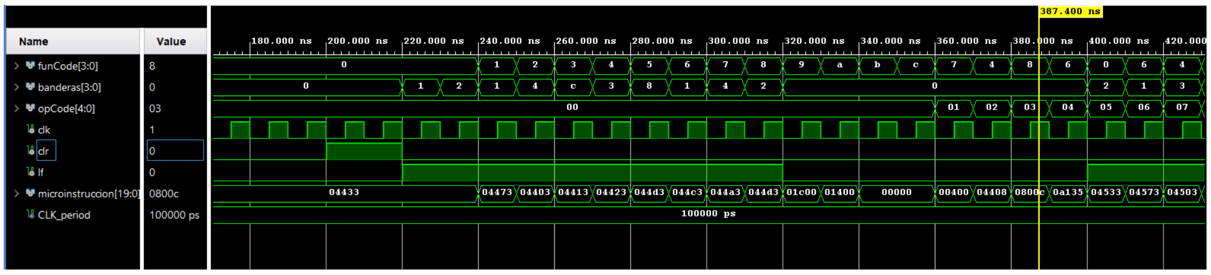
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
package paquete is

component bloqueNivel is
   Port ( clk,clr : in STD_LOGIC;
          NA : out STD_LOGIC);
end component;

component MfunCode is
   Port ( funCode : in STD_LOGIC_VECTOR (3 downto 0);
   outMFCode: out STD_LOGIC_VECTOR (19 downto 0));
end component;

component MopCode is
   Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
          outOpCode : out STD_LOGIC_VECTOR (19 downto 0));
end component;

component bloqueCondicion is
   Port ( q : in STD_LOGIC_VECTOR (3 downto 0);
          EQ, NE, LT, LE, GT, GE : out STD_LOGIC);
end component;

component decodificador is
   Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
          tipoR, BEQI, BNEI, BLTI, BLETI, BGTI, BGETI : out STD_LOGIC);
end component;
```

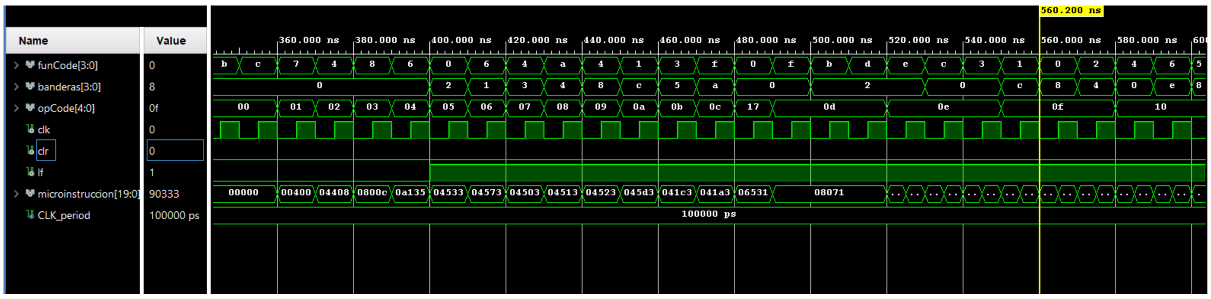
```
component sdopcMux is
    Port ( opCode : in STD_LOGIC_VECTOR (4 downto 0);
           SDOPC : in STD_LOGIC;
           muxUOut : out STD_LOGIC_VECTOR (4 downto 0));
end component;

component smMux is
    Port ( MfunCode : in STD_LOGIC_VECTOR (19 downto 0);
           MopCode : in STD_LOGIC_VECTOR (19 downto 0);
           microinstruccion : out STD_LOGIC_VECTOR (19 downto 0);
           SM : in STD_LOGIC);
end component;
component unidadControl is
    Port ( clk, clr,tipoR, BEQI, BNEI, BLTI,BLETI,BGTI, BGETI, EQ, NE, LT, LE, GT, GE, NA : in
STD_LOGIC;
           SDOPC, SM : out STD_LOGIC);
end component;
component capturaBanderas is
    Port ( LF, clk, clr : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR (3 downto 0);
           Q : out STD_LOGIC_VECTOR (3 downto 0));
end component;
end package;
```

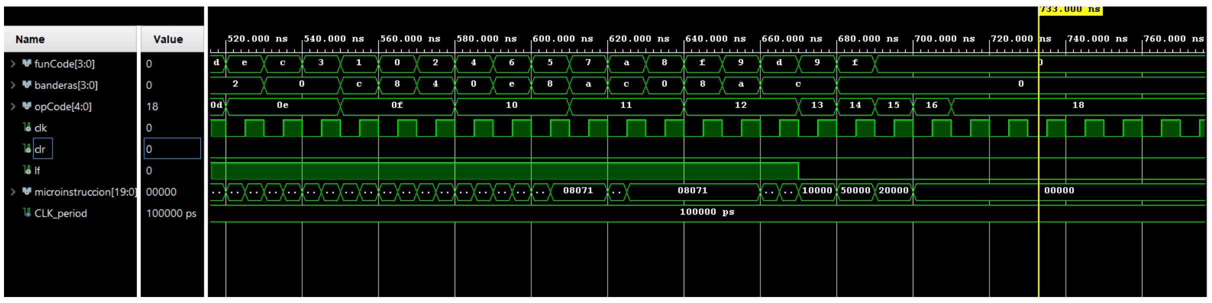
Diagrama RTL:



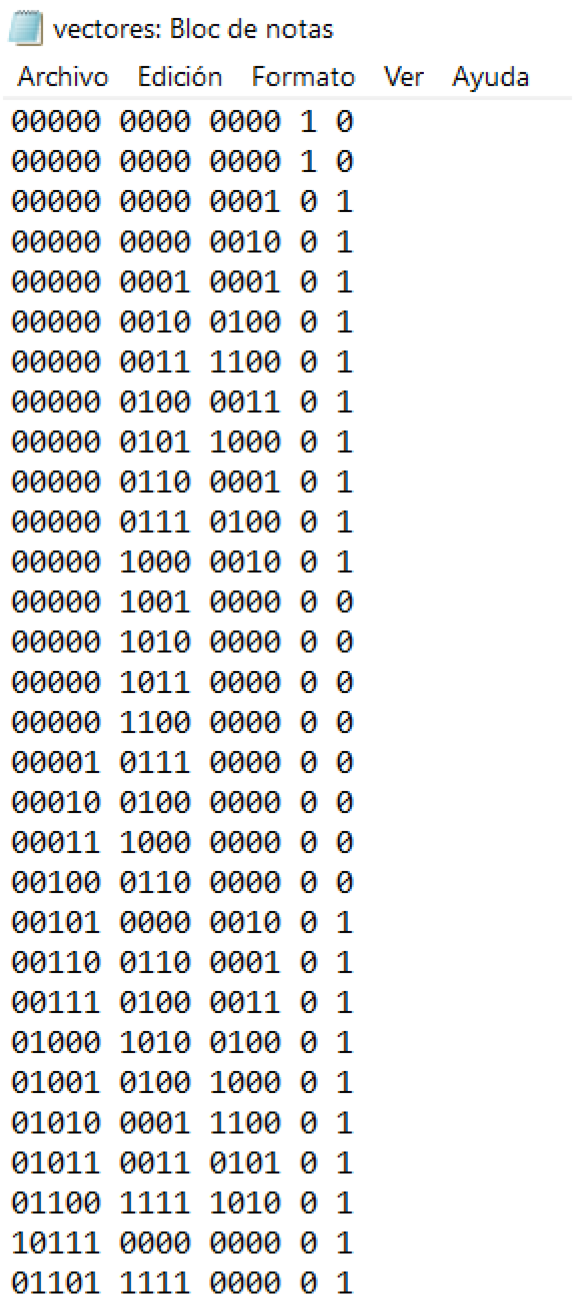
Forma de onda de la simulación completa:





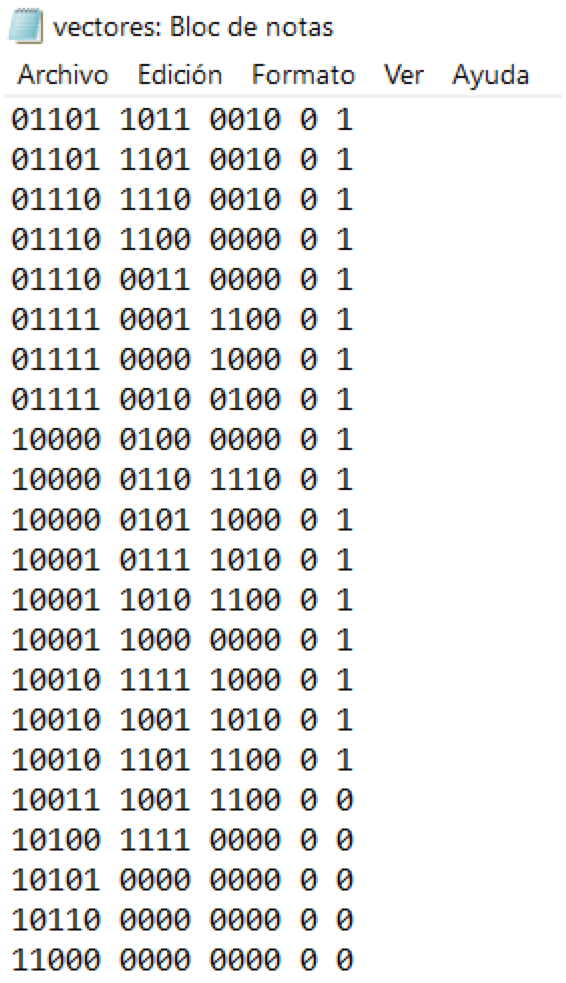


Impresión de pantalla del archivo de entradas:



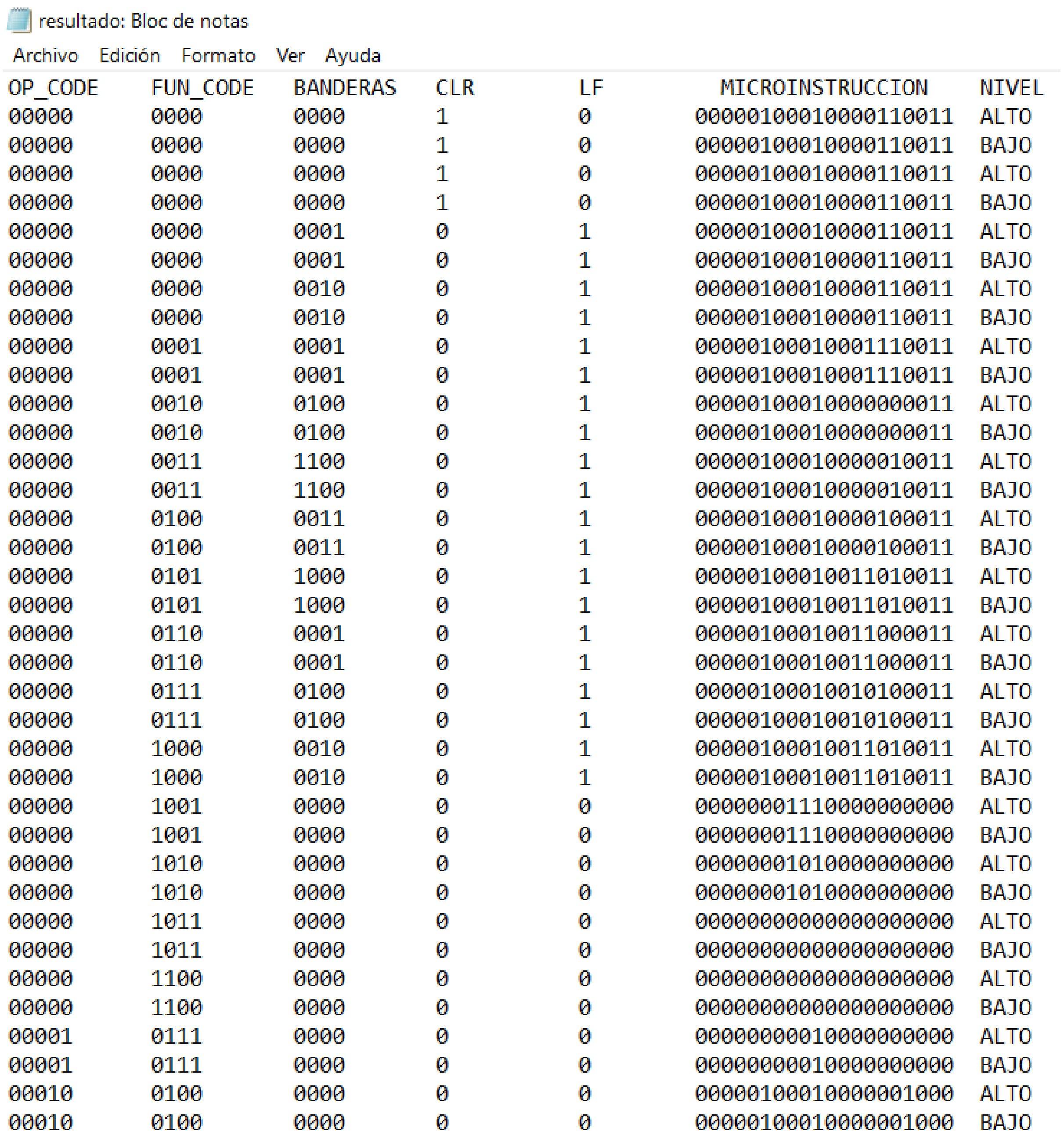
```
00000 0000 0000 1 0
00000 0000 0000 1 0
00000 0000 0001 0 1
00000 0000 0010 0 1
00000 0001 0001 0 1
00000 0010 0100 0 1
00000 0011 1100 0 1
00000 0100 0011 0 1
00000 0101 1000 0 1
00000 0110 0001 0 1
00000 0111 0100 0 1
00000 1000 0010 0 1
00000 1001 0000 0 0
00000 1010 0000 0 0
00000 1011 0000 0 0
00000 1100 0000 0 0
00001 0111 0000 0 0
00010 0100 0000 0 0
00011 1000 0000 0 0
00100 0110 0000 0 0
00101 0000 0010 0 1
00110 0110 0001 0 1
00111 0100 0011 0 1
01000 1010 0100 0 1
01001 0100 1000 0 1
01010 0001 1100 0 1
01011 0011 0101 0 1
01100 1111 1010 0 1
10111 0000 0000 0 1
01101 1111 0000 0 1
```

Sebastián Cipriano Damián P11



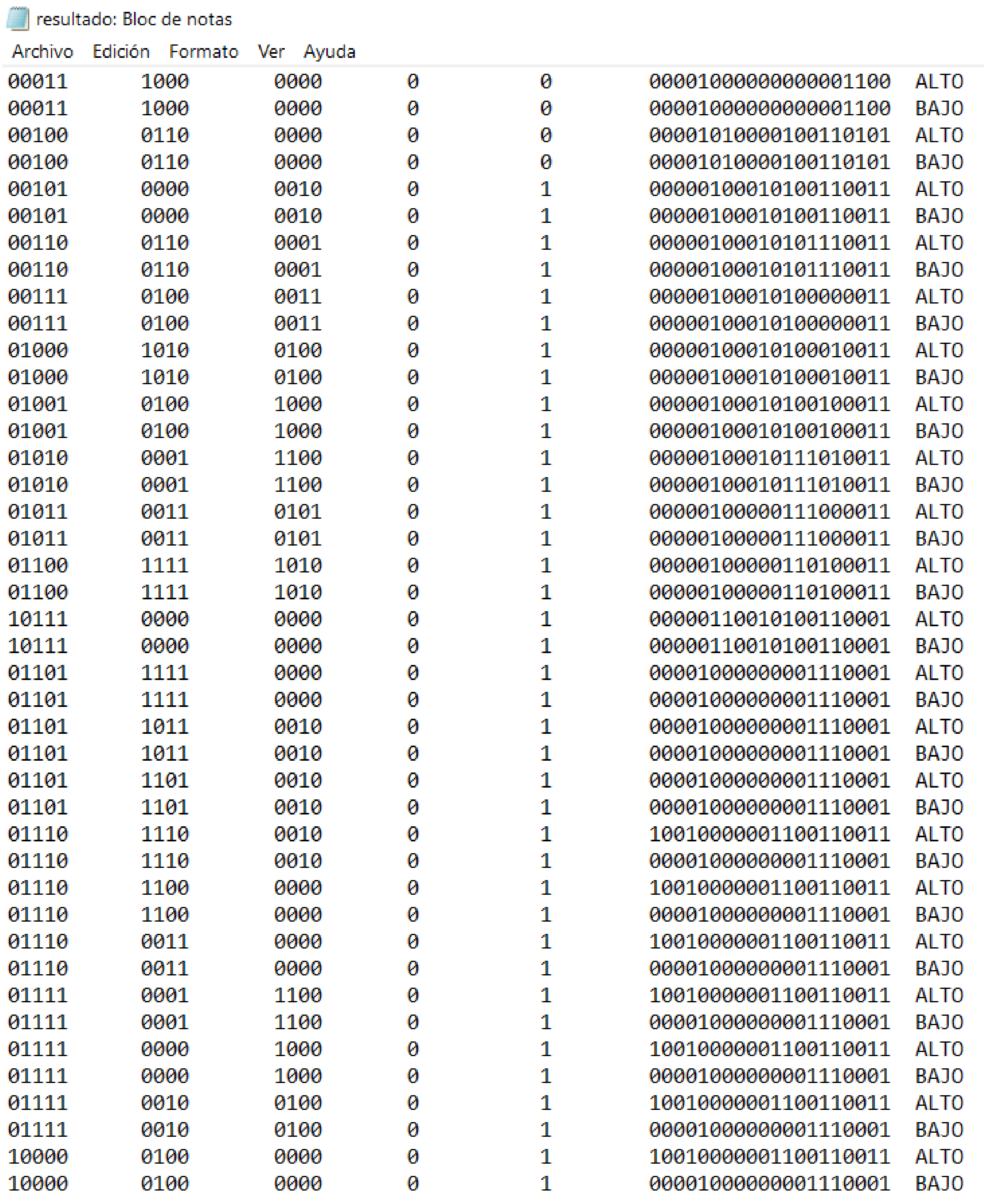
```
01101 1011 0010 0 1
01101 1101 0010 0 1
01110 1110 0010 0 1
01110 1100 0000 0 1
01110 0011 0000 0 1
01111 0001 1100 0 1
01111 0000 1000 0 1
01111 0010 0100 0 1
10000 0100 0000 0 1
10000 0110 1110 0 1
10000 0101 1000 0 1
10001 0111 1010 0 1
10001 1010 1100 0 1
10001 1000 0000 0 1
10010 1111 1000 0 1
10010 1001 1010 0 1
10010 1101 1100 0 1
10011 1001 1100 0 0
10100 1111 0000 0 0
10101 0000 0000 0 0
10110 0000 0000 0 0
11000 0000 0000 0 0
```

Impresión de pantalla del archivo de salida:



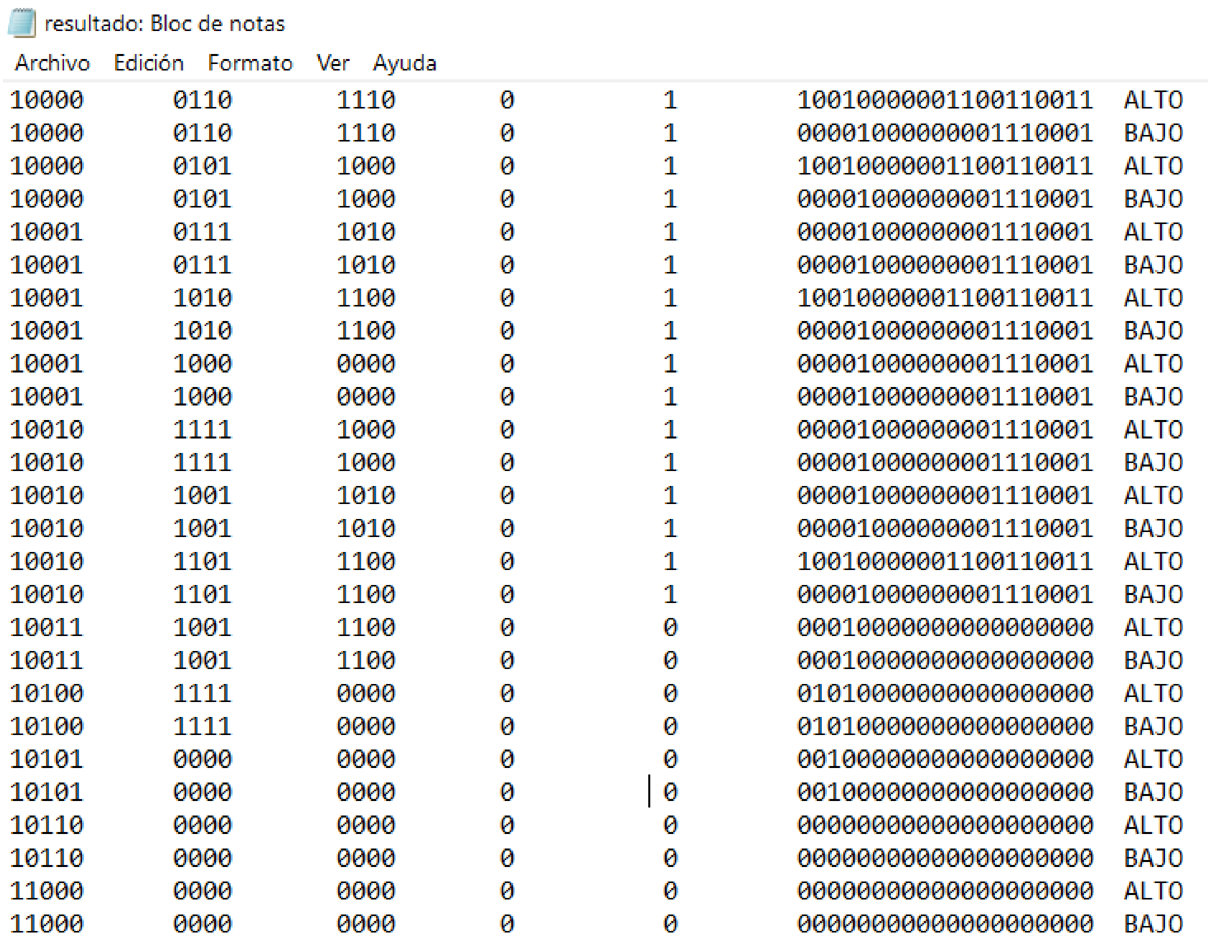
```
OP_CODE    FUN_CODE   BANDERAS   CLR        LF          MICROINSTRUCCION    NIVEL
00000      0000       0000       1          0         00000100010000110011  ALTO
00000      0000       0000       1          0         00000100010000110011  BAJO
00000      0000       0000       1          0         00000100010000110011  ALTO
00000      0000       0000       1          0         00000100010000110011  BAJO
00000      0000       0001       0          1         00000100010000110011  ALTO
00000      0000       0001       0          1         00000100010000110011  BAJO
00000      0000       0010       0          1         00000100010000110011  ALTO
00000      0000       0010       0          1         00000100010000110011  BAJO
00000      0001       0001       0          1         00000100010001110011  ALTO
00000      0001       0001       0          1         00000100010001110011  BAJO
00000      0010       0100       0          1         00000100010000000011  ALTO
00000      0010       0100       0          1         00000100010000000011  BAJO
00000      0011       1100       0          1         00000100010000010011  ALTO
00000      0011       1100       0          1         00000100010000010011  BAJO
00000      0100       0011       0          1         00000100010000100011  ALTO
00000      0100       0011       0          1         00000100010000100011  BAJO
00000      0101       1000       0          1         00000100010011010011  ALTO
00000      0101       1000       0          1         00000100010011010011  BAJO
00000      0110       0001       0          1         00000100010011000011  ALTO
00000      0110       0001       0          1         00000100010011000011  BAJO
00000      0111       0100       0          1         00000100010010100011  ALTO
00000      0111       0100       0          1         00000100010010100011  BAJO
00000      1000       0010       0          1         00000100010011010011  ALTO
00000      1000       0010       0          1         00000100010011010011  BAJO
00000      1001       0000       0          0         00000001110000000000  ALTO
00000      1001       0000       0          0         00000001110000000000  BAJO
00000      1010       0000       0          0         00000001010000000000  ALTO
00000      1010       0000       0          0         00000001010000000000  BAJO
00000      1011       0000       0          0         00000000000000000000  ALTO
00000      1011       0000       0          0         00000000000000000000  BAJO
00000      1100       0000       0          0         00000000000000000000  ALTO
00000      1100       0000       0          0         00000000000000000000  BAJO
00001      0111       0000       0          0         00000000010000000000  ALTO
00001      0111       0000       0          0         00000000010000000000  BAJO
00010      0100       0000       0          0         00000100010000001000  ALTO
00010      0100       0000       0          0         00000100010000001000  BAJO
```

Sebastián Cipriano Damián P11



```
00011	1000	0000	0	0	00001000000000001100	ALTO
00011	1000	0000	0	0	00001000000000001100	BAJO
00100	0110	0000	0	0	00001010000100110101	ALTO
00100	0110	0000	0	0	00001010000100110101	BAJO
00101	0000	0010	0	1	00001000101001100110011	ALTO
00101	0000	0010	0	1	00001000101001100110011	BAJO
```

Sebastián Cipriano Damián P11



```
10000      0110      1110      0       1       10010000001100110011  ALTO
10000      0110      1110      0       1       00001000000001110001  BAJO
10000      0101      1000      0       1       10010000001100110011  ALTO
10000      0101      1000      0       1       00001000000001110001  BAJO
10001      0111      1010      0       1       00001000000001110001  ALTO
10001      0111      1010      0       1       00001000000001110001  BAJO
10001      1010      1100      0       1       10010000001100110011  ALTO
10001      1010      1100      0       1       00001000000001110001  BAJO
10001      1000      0000      0       1       00001000000001110001  ALTO
10001      1000      0000      0       1       00001000000001110001  BAJO
10010      1111      1000      0       1       00001000000001110001  ALTO
10010      1111      1000      0       1       00001000000001110001  BAJO
10010      1001      1010      0       1       00001000000001110001  ALTO
10010      1001      1010      0       1       00001000000001110001  BAJO
10010      1101      1100      0       1       10010000001100110011  ALTO
10010      1101      1100      0       1       00001000000001110001  BAJO
10011      1001      1100      0       0       00010000000000000000  ALTO
10011      1001      1100      0       0       00010000000000000000  BAJO
10100      1111      0000      0       0       01010000000000000000  ALTO
10100      1111      0000      0       0       01010000000000000000  BAJO
10101      0000      0000      0       0       00100000000000000000  ALTO
10101      0000      0000      0       0       00100000000000000000  BAJO
10110      0000      0000      0       0       00000000000000000000  ALTO
10110      0000      0000      0       0       00000000000000000000  BAJO
11000      0000      0000      0       0       00000000000000000000  ALTO
11000      0000      0000      0       0       00000000000000000000  BAJO
```