Código de implementación:

Memoria de programa:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.std_logic_arith.ALL;
entity memoriaPrograma is
--Parametrizado
   generic(
   --NUMERO DE BITS DEL BUS DE DIRECCIONES
      p : integer := 10;
      d : integer := 25
   );
   Port ( PC : in STD_LOGIC_VECTOR (p-1 downto 0);
         Inst : out STD_LOGIC_VECTOR (d-1 downto 0));
end memoriaPrograma;

architecture Behavioral of memoriaPrograma is
constant OPLI : std_logic_vector(4 downto 0) := "00001";
constant OPR : std_logic_vector(4 downto 0) := "00000";
constant OPLWI : std_logic_vector(4 downto 0) := "00010";
constant OPLW : std_logic_vector(4 downto 0) := "10111";
constant OPSWI : std_logic_vector(4 downto 0) := "00011";
constant OPSW : std_logic_vector(4 downto 0) := "00100";
constant OPADDI : std_logic_vector(4 downto 0) := "00101";
constant OPSUBI : std_logic_vector(4 downto 0) := "00110";
constant OPANDI : std_logic_vector(4 downto 0) := "00111";
constant OPORI : std_logic_vector(4 downto 0) := "01000";
constant OPXORI : std_logic_vector(4 downto 0) := "01001";
constant OPNANDI : std_logic_vector(4 downto 0) := "01010";
constant OPNORI : std_logic_vector(4 downto 0) := "01011";
constant OPXNORI : std_logic_vector(4 downto 0) := "01100";
constant OPBEQI : std_logic_vector(4 downto 0) := "01101";
constant OPBNEI : std_logic_vector(4 downto 0) := "01110";
constant OPBLTI : std_logic_vector(4 downto 0) := "01111";
constant OPBLETI : std_logic_vector(4 downto 0) := "10000";
constant OPBGTI : std_logic_vector(4 downto 0) := "10001";
constant OPBGETI : std_logic_vector(4 downto 0) := "10010";
constant OPB : std_logic_vector(4 downto 0) := "10011";
constant OPCALL : std_logic_vector(4 downto 0) := "10100";
constant OPRET : std_logic_vector(4 downto 0) := "10101";
constant OPNOT : std_logic_vector(4 downto 0) := "10110";
constant SU : std_logic_vector(3 downto 0) := "0000";
type aux is array(0 to (2**p)-1) of std_logic_vector(d-1 downto 0);
constant caja : aux := (
```

```
    OPLI & SU & SU & SU & SU & SU,
    OPLI & "0001" & SU & SU & SU & "0001",
    OPLI & "0010" & SU & SU & SU & SU,
    OPLI & "0011" & SU & SU & SU & "1100",
    OPR & "0100" & SU & "0001" & SU & SU,
    OPSWI & "0100" & SU & SU & "0100" & "1000",
    OPADDI & SU & "0001" & SU & SU & SU,
    OPADDI & "0001" & "0100" & SU & SU & SU,
    OPADDI & "0010" & "0010" & SU & SU & "0001",
    OPBNEI & "0011" & "0010" & "1111" & "1111" & "1011",
    OPNOT & SU & SU & SU & SU & SU,
    OPB & SU & SU & SU & SU & "1010",
    others => (others => '0')
);

begin

Inst <= caja(conv_integer(PC(9 downto 0)));

end Behavioral;
```

Memoria de datos:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.std_logic_arith.ALL;
entity memoriaDatos is
--Parametrizado
    generic(
    --NUMERO DE BITS DEL BUS DE DIRECCIONES
        p : integer := 11;
        d : integer := 16
    );
    Port ( add : in STD_LOGIC_VECTOR (p-1 downto 0);
           dataIn : in STD_LOGIC_VECTOR(d-1 downto 0);
           clk, wd : in STD_LOGIC;
           dataOut : out STD_LOGIC_VECTOR (d-1 downto 0));
end memoriaDatos;

architecture Behavioral of memoriaDatos is

type aux is array(0 to (2**p)-1) of std_logic_vector(d-1 downto 0);
signal caja : aux;

begin
process(clk)
begin
```

```
    if (rising_edge(clk)) then
        if(wd = '1')then
            caja(conv_integer(add))<=dataIn;
       -- else
        end if;
    end if;
end process;
dataOut <= caja(conv_integer(add));
end Behavioral;
```

Código de simulación:

Memoria de programa:

```
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 memoriaPrograma_TB is
    generic(
    --NUMERO DE BITS DEL BUS DE DIRECCIONES
        p : integer := 10;
        d : integer := 25
    );
end memoriaPrograma_TB;

architecture Behavioral of memoriaPrograma_TB is
component memoriaPrograma is
--Parametrizado
    generic(
    --NUMERO DE BITS DEL BUS DE DIRECCIONES
        p : integer := 10;
        d : integer := 25
    );
    Port ( PC : in STD_LOGIC_VECTOR (p-1 downto 0);
           Inst : out STD_LOGIC_VECTOR (d-1 downto 0));
end component;
signal Inst :  STD_LOGIC_VECTOR (d-1 downto 0);
signal PC : STD_LOGIC_VECTOR (p-1 downto 0);
signal CLK : STD_LOGIC;

constant CLK_period : time := 10 ns;
begin
```

```
u1: memoriaPrograma
    Port map(
    Inst => Inst,
    PC => PC
    );
   CLK_process :process
begin
    CLK <= '0';
    wait for CLK_period/2;
    CLK <= '1';
    wait for CLK_period/2;
end process;
stim_process : process
file ARCH_RES : TEXT;
variable LINEA_RES : line;
--VARIABLES PARA LAS ENTRADAS
variable var_PC : STD_LOGIC_VECTOR (p-1 downto 0);

file ARCH_VEC : TEXT;
variable LINEA_VEC : line;
        VARIABLE var_OPCODE : STD_LOGIC_VECTOR(4 DOWNTO 0);
        VARIABLE var_1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
        VARIABLE var_2 : STD_LOGIC_VECTOR(3 DOWNTO 0);
        VARIABLE var_3 : STD_LOGIC_VECTOR(3 DOWNTO 0);
        VARIABLE var_4 : STD_LOGIC_VECTOR(3 DOWNTO 0);
        VARIABLE var_5 : STD_LOGIC_VECTOR(3 DOWNTO 0);
variable CADENA : STRING(1 to 7);
--VARIABLES PARA LOS PUERTOS DE SALIDA
variable var_Inst : STD_LOGIC_VECTOR (d-1 downto 0);
begin
                file_open(ARCH_VEC, "\vectores.txt", READ_MODE);
                file_open(ARCH_RES, "\resultado.txt", WRITE_MODE);

                -- write (   linea  , valor a escribir, lado a escribir, tamaño)
                CADENA := "PC     ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
    CADENA := "OPCODE ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "19..16 ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "15..12 ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
            CADENA := "11..08 ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "07..04 ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
            CADENA := "03..00 ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
```

```
                writeline(ARCH_RES,LINEA_RES);-- escribe la linea en el archivo


                WAIT FOR 100 NS;
                FOR I IN 0 TO 11 LOOP
                   WAIT UNTIL RISING_EDGE(CLK);
                        readline(ARCH_VEC,LINEA_VEC); -- lee una linea completa
                        read(LINEA_VEC, var_PC);
        PC <= var_PC;

        var_OPCODE := inst(24 downto 20);
        var_1 := inst(19 downto 16);
        var_2 := inst(15 downto 12);
        var_3 := inst(11 downto 8);
        var_4 := inst(7 downto 4);
        var_5 := inst(3 downto 0);
        Hwrite(LINEA_RES, var_PC, left, 8);
        write(linea_res,var_OPCODE, left, 9);
                        write(linea_res,var_1, left, 8);
                        write(linea_res,var_2, left, 8);
                        write(linea_res,var_3, left, 8);
                        write(linea_res,var_4, left, 8);
                        write(linea_res,var_5, left, 8);
                        writeline(ARCH_RES,LINEA_RES);-- escribe la linea en el archivo
                end loop;
                file_close(ARCH_VEC);  -- cierra el archivo
                file_close(ARCH_RES);  -- cierra el archivo
    wait;
  end process;
end Behavioral;
```

Memoria de datos:

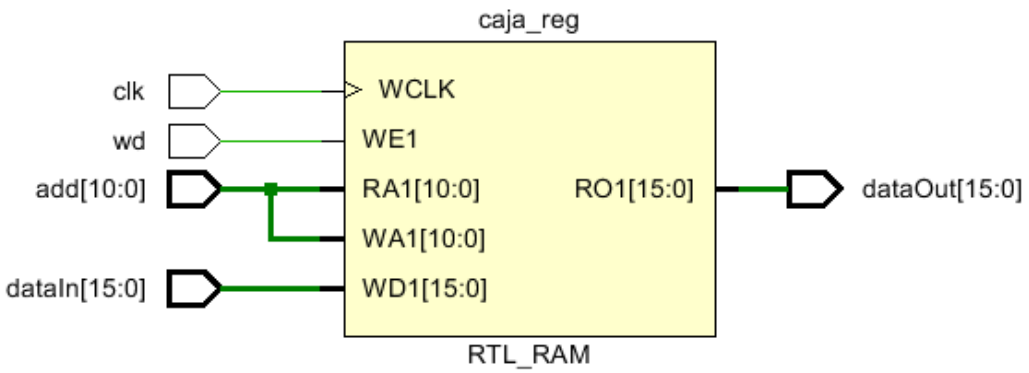
```
LIBRARY IEEE;
LIBRARY STD;
USE STD.TEXTIO.ALL;
USE ieee.std_logic_TEXTIO.ALL;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_UNSIGNED.ALL;
USE ieee.std_logic_ARITH.ALL;
entity memoriaDatos_TB is
generic(
   --NUMERO DE BITS DEL BUS DE DIRECCIONES
      p : integer := 11;
      d : integer := 16
   );
end memoriaDatos_TB;
```

```
architecture Behavioral of memoriaDatos_TB is
component memoriaDatos is
--Parametrizado
   generic(
   --NUMERO DE BITS DEL BUS DE DIRECCIONES
      p : integer := 11;
      d : integer := 16
   );
   Port ( add : in STD_LOGIC_VECTOR (p-1 downto 0);
         dataIn : in STD_LOGIC_VECTOR(d-1 downto 0);
         clk, wd : in STD_LOGIC;
         dataOut : out STD_LOGIC_VECTOR (d-1 downto 0));
end component;
signal add : STD_LOGIC_VECTOR (p-1 downto 0);
signal dataIn : STD_LOGIC_VECTOR(d-1 downto 0);
signal clk, wd : STD_LOGIC;
signal dataOut : STD_LOGIC_VECTOR (d-1 downto 0);
begin
u1 : memoriaDatos
   Port map(
      add => add,
      dataIn => dataIn,
      clk => clk,
      wd => wd,
      dataOut => dataOut
   );

process
begin
   clk <= '0';
   wait for 10 ns;
   clk <= '1';
   wait for 10 ns;
end process;

   stim_process : process
   file ARCH_RES : TEXT;
   variable LINEA_RES : line;
   --VARIABLES PARA LAS ENTRADAS
   variable var_add : STD_LOGIC_VECTOR (p-1 downto 0);
   variable  var_dataIn : STD_LOGIC_VECTOR (d-1 downto 0);
   variable var_wd : STD_LOGIC;

   file ARCH_VEC : TEXT;
   variable LINEA_VEC : line;
   variable CADENA : STRING(1 to 7);
   --VARIABLES PARA LOS PUERTOS DE SALIDA
   variable var_dataOut : STD_LOGIC_VECTOR (d-1 downto 0);
```

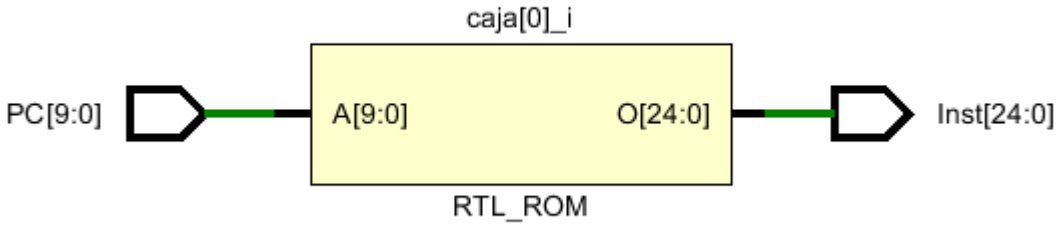
```
 begin
                file_open(ARCH_VEC,
"C:\Users\sebas\Desktop\PracticasArq\memoria1\vectores.txt", READ_MODE);
                file_open(ARCH_RES,
"C:\Users\sebas\Desktop\PracticasArq\memoria1\resultado.txt", WRITE_MODE);

                -- write (   linea  , valor a escribir, lado a escribir, tamaño)
                CADENA := "add    ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
      CADENA := "WD     ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "dataIn ";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                CADENA := "dataOut";
                write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
                writeline(ARCH_RES,LINEA_RES);-- escribe la linea en el archivo


                WAIT FOR 100 NS;
                FOR I IN 0 TO 11 LOOP
                        readline(ARCH_VEC,LINEA_VEC); -- lee una linea completa
                        read(LINEA_VEC, var_WD);
        WD <= var_WD;
        read(LINEA_VEC, var_add);
        add <= var_add;
        read(LINEA_VEC, var_dataIn);
        dataIn <= var_dataIn;

                        WAIT UNTIL RISING_EDGE(CLK);   --ESPERO AL FLANCO  DE
SUBIDA
        var_dataOut := dataOut;

        hwrite(LINEA_RES, var_add, left, 8);
        write(LINEA_RES, var_WD, left, 8);
        hwrite(LINEA_RES, var_dataIn, left, 8);
        hwrite(LINEA_RES, var_dataOut, left, 8);
                        writeline(ARCH_RES,LINEA_RES);-- escribe la linea en el archivo
                end loop;
                file_close(ARCH_VEC);  -- cierra el archivo
                file_close(ARCH_RES);  -- cierra el archivo
    wait;
  end process;
end Behavioral;
```

Diagrama RTL:

Memoria de datos:

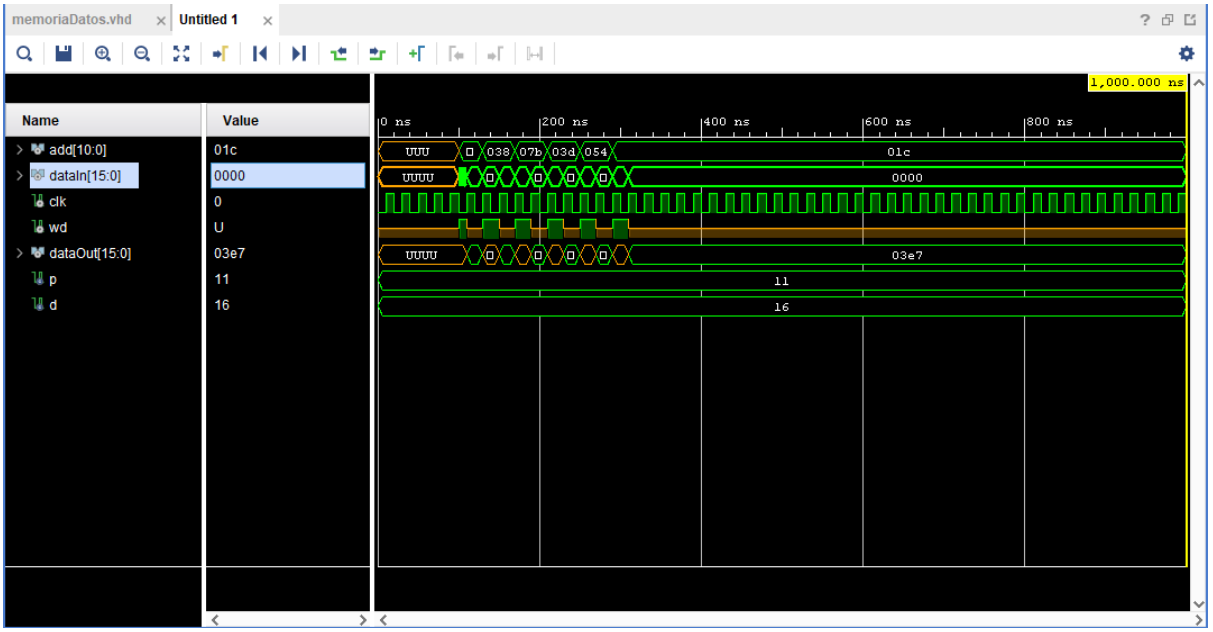


Memoria de programa:



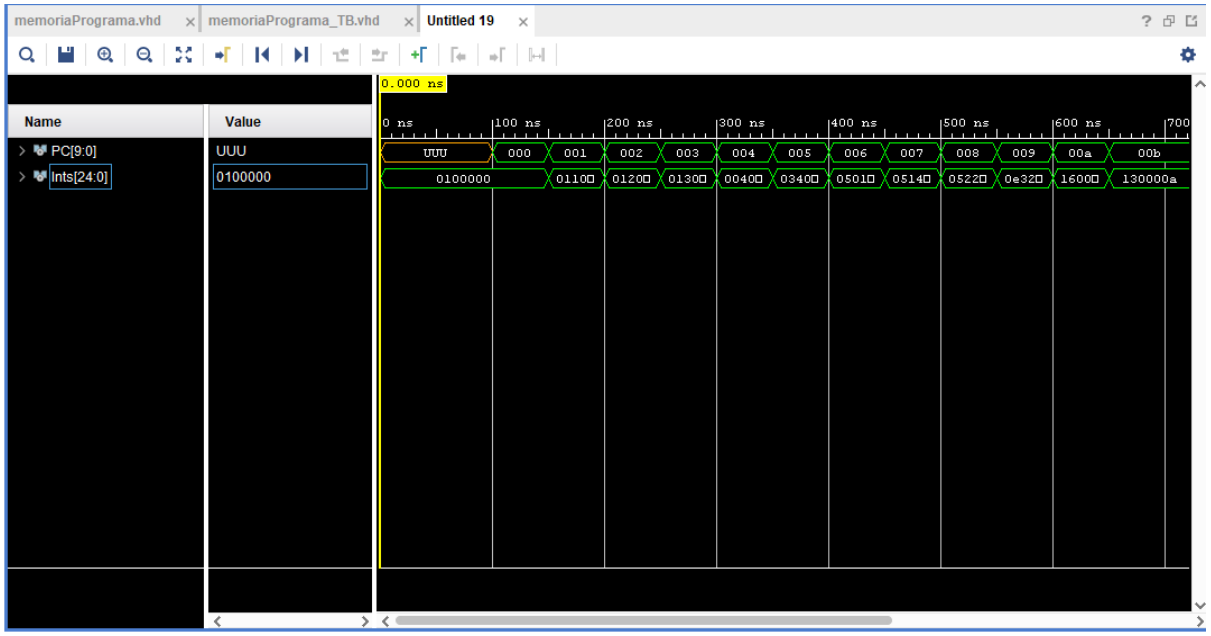
Forma de onda de simulación:

Memoria de datos:



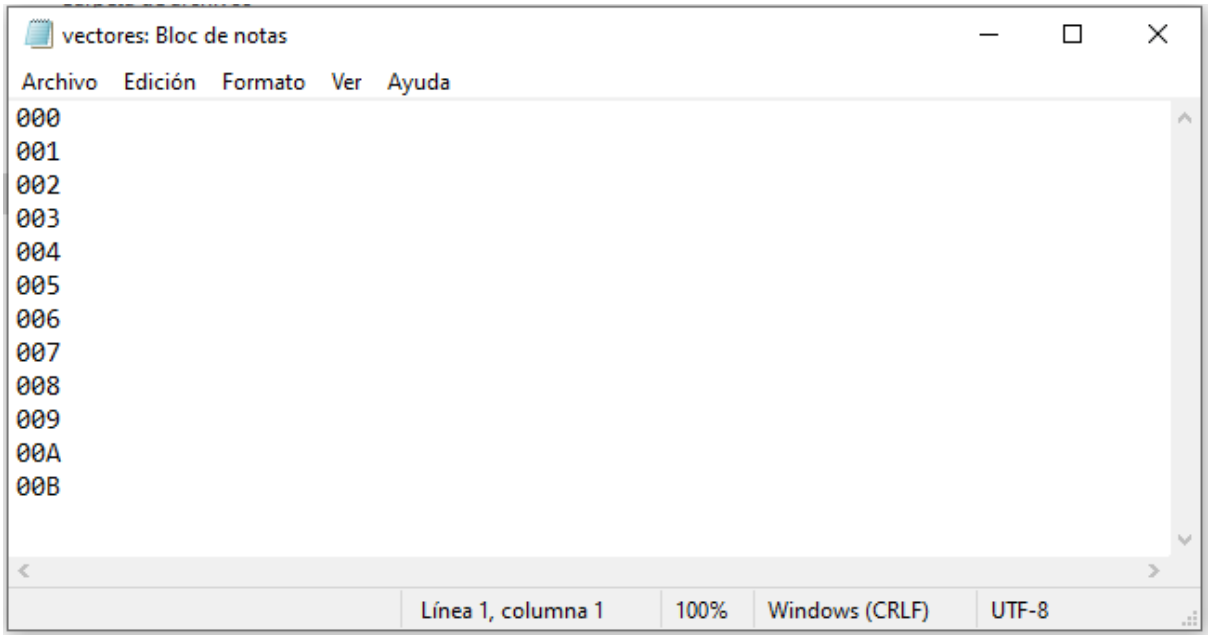
Practica 6 Sebastián Cipriano Damián 3CV12

Memoria de programa:



Archivos de estímulos:

Memoria de datos:



Memoria de programa:

vectores: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
1 00001001000 0000100100111010
x 00001001000 0000000000000000
1 00000111000 0000000001111111
x 00000111000 0000000000000000
1 00001111011 0000000000100001
x 00001111011 0000000000000000
1 00000111101 0000000001011010
x 00000111101 0000000000000000
1 00001010100 0000000011101000
x 00001010100 0000000000000000
1 00000011100 0000001111100111
x 00000011100 0000000000000000
```

Línea 1, columna 1 100% Windows (CRLF) UTF-8

Archivos de salida:

Memoria de datos:

resultado: Bloc de notas

Archivo Edición Formato Ver Ayuda

add	WD	dataIn	dataOut
048	1	093A	XXXX
048	U	0000	093A
038	1	007F	XXXX
038	U	0000	007F
07B	1	0021	XXXX
07B	U	0000	0021
03D	1	005A	XXXX
03D	U	0000	005A
054	1	00E8	XXXX
054	U	0000	00E8
01C	1	03E7	XXXX
01C	U	0000	03E7

Línea 1, columna 1 100% Windows (CRLF) UTF-8

Memoria de programa:

resultado: Bloc de notas

Archivo Edición Formato Ver Ayuda

PC	OPCODE	19..16	15..12	11...8	7....4	3....0
000	00001	0000	0000	0000	0000	0000
001	00001	0001	0000	0000	0000	0001
002	00001	0010	0000	0000	0000	0000
003	00001	0011	0000	0000	0000	1100
004	00000	0100	0000	0001	0000	0000
005	00011	0100	0000	0000	0111	0010
006	00101	0000	0001	0000	0000	0000
007	00101	0001	0100	0000	0000	0000
008	00101	0010	0010	0000	0000	0001
009	01110	0011	0010	1111	1111	1011
00A	10110	0000	0000	0000	0000	0000
00B	10011	0000	0000	0000	0000	1010

Línea 1, columna 1 100% Windows (CRLF) UTF-8