## Código de implementación:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity registro is
  Port (
     writeData: in std_logic_vector (15 downto 0);
     writeReg: in std logic vector (3 downto 0);
     readReg1: in std_logic_vector ( 3 downto 0 );
     readReg2: in std logic vector ( 3 downto 0 );
     shamt: in std logic vector (3 downto 0);
     WR: in std_logic;
     DIR: in std logic;
     SHE: in std logic;
     clr: in std logic;
     clk: in std logic;
     readData1: out std logic vector (15 downto 0);
     readData2: out std_logic_vector ( 15 downto 0 )
  );
end registro;
architecture Behavioral of registro is
type arreglo is array (0 to 15) of std_logic_vector( 15 downto 0 );
signal registros: arreglo;
begin

    Lectura del archivo de registros

  readData1 <= registros( conv integer( readReg1 ) );
  readData2 <= registros( conv integer( readReg2 ) );</pre>
  -- Carga en el archivo de registros
  process (clk, clr)
  variable shift: bit vector (15 downto 0);
  variable auxq: std_logic_vector ( 15 downto 0 );
  begin
     shift := to_bitvector( registros ( conv_integer( readReg2 ) ) );
     if ( clr = '1' )then
       registros <= (others => (others => '0'));
     elsif (rising edge(clk)) then -- Señales síncronas
       if (WR = '1' and SHE = '0') then -- Carga
          registros(conv integer( writeReg )) <= writeData;
       elsif (WR = '1' and SHE = '1') then -- Corrimiento
          if (DIR = '0') then -- Corrimiento derecha
            shift := shift srl conv_integer( shamt );
          else -- Corimiento izquierda
```

```
shift := shift sll conv_integer( shamt );
    end if;
    registros(conv_integer( writeReg )) <= to_stdlogicvector( shift );
    end if;
    end if;
    end process;
end Behavioral;</pre>
```

### Código de simulación:

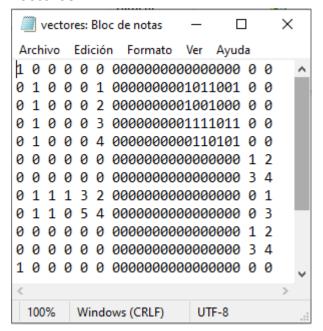
```
LIBRARY IEEE;
LIBRARY STD;
USE STD.TEXTIO.ALL;
USE ieee.std_logic_TEXTIO.ALL;
USE ieee.std logic 1164.ALL;
USE ieee.std_logic_UNSIGNED.ALL;
USE ieee.std_logic_ARITH.ALL;
entity fileReg TB is
-- Port ();
end fileReg TB;
architecture Behavioral of fileReg TB is
component registro is
  Port (
    writeData: in std_logic_vector (15 downto 0);
    writeReg: in std logic vector ( 3 downto 0 );
    readReg1: in std_logic_vector ( 3 downto 0 );
    readReg2: in std logic vector ( 3 downto 0 );
    shamt: in std logic vector ( 3 downto 0 );
    WR: in std logic;
    DIR: in std logic;
    SHE: in std logic;
    clr: in std logic;
    clk: in std logic;
    readData1: out std logic vector (15 downto 0);
    readData2: out std_logic_vector ( 15 downto 0 )
  );
end component;
signal writeData: std logic vector (15 downto 0):=(others=>'0');
signal writeReg: std_logic_vector ( 3 downto 0 ):=(others=>'0');
signal readReg1: std logic vector ( 3 downto 0 ):=(others=>'0');
signal readReg2: std_logic_vector ( 3 downto 0 ):=(others=>'0');
signal shamt: std_logic_vector ( 3 downto 0 ):=(others=>'0');
signal WR: std logic:='0';
```

```
signal DIR: std_logic:='0';
signal SHE: std_logic:='0';
signal clk: std logic:='0';
signal clr: std_logic:='0';
signal readData1: std logic vector (15 downto 0);
signal readData2: std_logic_vector ( 15 downto 0 );
begin
asignacion: registro
  Port map(
    writeData => writeData,
    writeReg => writeReg,
    readReg1 => readReg1,
    readReg2 => readReg2,
    shamt => shamt.
    WR => WR
    DIR => DIR,
    SHE => SHE,
    clk => clk,
    clr => clr,
    readData1 => readData1,
    readData2 => readData2
  );
  process
    begin
       clk <= '0';
       wait for 15 ns;
       clk <= '1';
       wait for 15 ns;
     end process;
  stim proc: process
       file ARCH_RES : TEXT;
       variable LINEA_RES: line;
       -- Variables para las entradas
       variable var_writeReg, var_shamt : STD_LOGIC_VECTOR (3 downto 0);
  variable var_writeData : STD_LOGIC_VECTOR (15 downto 0);
  variable var_readReg1 : STD_LOGIC_VECTOR (3 downto 0);
  variable var_readReg2 : STD_LOGIC_VECTOR (3 downto 0);
  variable var_WR, var_dir, var_SHE, var_clr: STD_LOGIC;
file ARCH_VEC : TEXT;
       variable LINEA VEC: line;
       VARIABLE CADENA: STRING(1 TO 7);
       -- Variables para los puertos de salida
       variable var readData1: STD LOGIC VECTOR (15 downto 0);
```

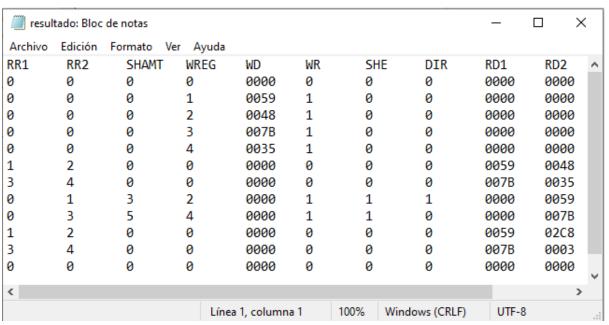
```
variable var_readData2 : STD_LOGIC_VECTOR (15 downto 0);
 begin
             file_open(ARCH_VEC,
"C:\Users\sebas\Desktop\PracticasArg\Practica5\vectores.txt", READ MODE);
             file open(ARCH RES,
"C:\Users\sebas\Desktop\PracticasArq\Practica5\resultado.txt", WRITE MODE);
             -- write ( linea , valor a escribir, lado a escribir, tamaño)
             CADENA := "RR1 ":
             write(LINEA RES, CADENA, left, CADENA'LENGTH+1);
    CADENA := "RR2 ";
             write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
             CADENA := "SHAMT ";
             write(LINEA RES, CADENA, left, CADENA'LENGTH+1);
             CADENA := "WREG ";
             write(LINEA RES, CADENA, left, CADENA'LENGTH+1);
             CADENA := "WD
             write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
             CADENA := "WR
             write(LINEA RES, CADENA, left, CADENA'LENGTH+1);
             CADENA := "SHE ";
             write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
             CADENA := "DIR ";
             write(LINEA_RES, CADENA, left, CADENA'LENGTH+1);
             CADENA := "RD1 ";
             write(LINEA RES, CADENA, left, CADENA'LENGTH+1);
             CADENA := "RD2 ";
             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_clr);
      clr <= var clr;
      read(LINEA_VEC, var_WR);
      WR <= var WR;
      read(LINEA_VEC, var_SHE);
      SHE <= var_SHE;
      read(LINEA_VEC, var_dir);
      dir <= var dir;
      hread(LINEA_VEC, var_shamt);
      shamt <= var shamt;
      hread(LINEA_VEC, var_writeReg);
      writeReg <= var_writeReg;</pre>
      read(LINEA VEC, var writeData);
```

```
writeData <= var_writeData;</pre>
       hread(LINEA_VEC, var_readReg1);
       readReg1 <= var readReg1;</pre>
       hread(LINEA_VEC, var_readReg2);
       readReg2 <= var readReg2;</pre>
                     WAIT UNTIL RISING_EDGE(CLK); --ESPERO AL FLANCO DE
SUBIDA
       var readData1 := readData1;
       var_readData2 := readData2;
       hwrite(LINEA_RES, var_readReg1, left, 8);
       hwrite(LINEA_RES, var_readReg2, left, 8);
       hwrite(LINEA RES, var shamt, left, 8);
       hwrite(LINEA_RES, var_writeReg, left, 8);
       hwrite(LINEA_RES, var_writeData, left, 8);
       write(LINEA_RES, var_WR, left, 8);
       write(LINEA_RES, var_SHE, left, 8);
       write(LINEA_RES, var_dir, left, 8);
       hwrite(LINEA RES, var readData1, left, 8);
       hwrite(LINEA_RES, var_readData2, left, 18);
                     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;
```

#### **Vectores:**



#### Resultados:



# Diagrama RTL:

