```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity subtractorTB is
end entity;

architecture behavior of subtractorTB is
  constant TIME_DELAY : time := 20 ns;
  constant NUM_VALS : integer := 6;


  type A_array is array(0 to (NUM_VALS - 1)) of std_logic_vector(15 downto 0);
  type B_array is array(0 to (NUM_VALS - 1)) of std_logic_vector(15 downto 0);
  type C_array is array(0 to (NUM_VALS - 1)) of std_logic_vector(15 downto 0);
  type mode_array is array(0 to (NUM_VALS - 1)) of std_logic_vector(2 downto 0);
  type Zero_array is array(0 to (NUM_VALS - 1)) of std_logic;
  type OE_array is array(0 to (NUM_VALS - 1)) of std_logic;
  type Cout_array is array(0 to (NUM_VALS - 1)) of std_logic;

  -- Expected input and output data.
  -- Full Zeros
  -- a negative and a positive
  -- a negative and a negative
  -- a positive and negative in which the difference is zero
  -- overflows in the negative and positive direvtion
  -- a positive and a positive


  constant A_vals : A_array := (B"0000_0000_0000_0000",
                                B"1000_1000_0000_0000",  -- -30720
                                B"1111_0000_0000_0000", -- -4096
                                B"1111_1111_1111_1111", -- -1
                                B"1000_0000_0000_0000", -- -32768
                                B"0000_0000_0000_1000"); -- 8



  constant B_vals : B_array := (B"0000_0000_0000_0000",
                                B"0000_0001_0001_0000", -- 272
                                B"1000_0101_0000_0000", -- -31488
                                B"1111_1111_1111_1111", -- -1
                                B"1000_0000_0000_0001", -- -32767
                                B"0000_0000_0000_0010"); -- 2



  constant mode_vals : mode_array := (B"001",
                                      B"001",
                                      B"001",
                                      B"001",
                                      B"001",
                                      B"001");

  constant Zero_vals : Zero_array := ('1','0','0', '1', '0', '0');

  constant OE_vals : OE_array := ('1','1','1','1', '1', '1');

  constant Cout_vals : Cout_array := ('0','0','0', '0', '1', '0');

  constant C_vals : C_array := (B"0000_0000_0000_0000",
                                B"1000_0110_1111_0000", -- -30720 - 272 =
                                B"0110_1011_0000_0000", -- -4096 - -31488
                                B"0000_0000_0000_0000", -- -1 - -1 = -2
                                B"1111_1111_1111_1111", -- -32768 - -32767 = -1
                                B"0000_0000_0000_0110"); -- 8 - 2 = 6


  signal A_sig : std_logic_vector(15 downto 0);
  signal B_sig : std_logic_vector(15 downto 0);
  signal C_sig : std_logic_vector(15 downto 0);
  signal mode_sig : std_logic_vector(2 downto 0);
  signal Zero_sig : std_logic;
  signal OE_sig : std_logic;
  signal Cout_sig : std_logic;

begin

  DUT : entity work.ALU(behavioral)
    port map(A => A_sig,
             B => B_sig,
             C => C_sig,
             Mode => mode_sig,
             Zero => Zero_sig,
             OE => OE_sig,
             Cout => Cout_sig);

  stimulus : process
  begin
    for i in 0 to (NUM_VALS - 1) loop
      A_sig <= A_vals(i);
      B_sig <= B_vals(i);
      C_sig <= C_vals(i);
      mode_sig <= mode_vals(i);
      OE_sig <= OE_vals(i);
      wait for TIME_DELAY;
    end loop;
    wait;
  end process stimulus;

  monitor : process
    variable i : integer := 0;
  begin
    wait for TIME_DELAY/4;
    while (i < NUM_VALS) loop
      assert C_sig = C_vals(i)
        report "C value is incorrect."
        severity error;

      assert Zero_sig = Zero_vals(i)
        report "Zero value is incorrect."
        severity error;

      wait for TIME_DELAY/2;

      assert Cout_sig = Cout_vals(i)
        report "Cout value is incorrect."
        severity error;

      i := i + 1;
      wait for TIME_DELAY/2;
    end loop;
    wait;
  end process monitor;

end behavior;
```