```vhdl
--------------------------------------------------------------------------------
--
-- Title       : R4_2_tb
-- Design      : ALU
-- Author      : Aaron Lin and Hang Chen
-- Company     : Stony Brook University
--
--------------------------------------------------------------------------------
--
-- File        : C:\my_designs\ALU\ALU\src\R4_2_tb.vhd
-- Generated   : Tue Apr 21 22:51:52 2020
-- From        : interface description file
-- By          : Itf2Vhdl ver. 1.22
--
--------------------------------------------------------------------------------
--
-- Description : Testbench for the R4 010 Instruction. Checks if overflows and underflows are saturated properly, and checks if the normal case
--      is calculated correctly. Each of the 4 32 bit fields are tested.
--
--------------------------------------------------------------------------------
library work;
use work.all;
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.NUMERIC_STD.all;

entity R4_2_tb is
end R4_2_tb;

--}} End of automatically maintained section

architecture R4_2_tb of R4_2_tb is
signal ALUop : std_logic_vector(19 downto 0);
signal Res, regA_Data, regB_Data, regC_Data : std_logic_vector(127 downto 0);
constant period : time := 20ns;
begin

    uut : entity ALU
        port map (ALUop => ALUop, Res => Res, regA_Data => regA_Data, regB_Data => regB_Data, regC_Data => regC_Data);

    tb : process
    begin
        wait for period/2;

        ALUop <= "10010XXXXXXXXXXXXXXX";

        --Right most 32 bits
        --Testing max negative - (-1 * -1)
        regB_Data <= (31 downto 0 => '1', others => '0');
        regC_Data <= (31 downto 0 => '1', others => '0');
        regA_Data <= (31 => '1', others => '0');
        wait for period;

        --Testing max positive - (1 * -1)
        regB_Data <= (31 downto 0 => '1', others => '0');
        regC_Data <= (0 => '1', others => '0');
        regA_Data <= (30 downto 0 => '1', others => '0');
        wait for period;

        --Testing 0 - (3 * 3)
        regB_Data <= (1 downto 0 => '1', others => '0');
        regC_Data <= (1 downto 0 => '1', others => '0');
        regA_Data <= (others => '0');
        wait for period;

        --Testing 0 - (-3 * 3)
        regB_Data <= (1 downto 0 => '1', others => '0');
        regC_Data <= (31 downto 2 => '1', 1 => '0', 0 => '1', others => '0');
        regA_Data <= (others => '0');
        wait for period;

        --Next 32 bits
        --Testing max negative - (-1 * -1)
        regB_Data <= (63 downto 32 => '1', others => '0');
        regC_Data <= (63 downto 32 => '1', others => '0');
        regA_Data <= (63 => '1', others => '0');
        wait for period;

        --Testing max positive - (1 * -1)
        regB_Data <= (63 downto 32 => '1', others => '0');
        regC_Data <= (32 => '1', others => '0');
        regA_Data <= (62 downto 32 => '1', others => '0');
        wait for period;

        --Testing 0 - (3 * 3)
        regB_Data <= (33 downto 32 => '1', others => '0');
        regC_Data <= (33 downto 32 => '1', others => '0');
        regA_Data <= (others => '0');
        wait for period;

        --Testing 0 - (-3 * 3)
        regB_Data <= (33 downto 32 => '1', others => '0');
        regC_Data <= (63 downto 34 => '1', 33 => '0', 32 => '1', others => '0');
        regA_Data <= (others => '0');
        wait for period;

        --Next 32 bits
        --Testing max negative - (-1 * -1)
        regB_Data <= (95 downto 64 => '1', others => '0');
        regC_Data <= (95 downto 64 => '1', others => '0');
        regA_Data <= (95 => '1', others => '0');
        wait for period;

        --Testing max positive - (1 * -1)
        regB_Data <= (95 downto 64 => '1', others => '0');
        regC_Data <= (64 => '1', others => '0');
        regA_Data <= (94 downto 64 => '1', others => '0');
        wait for period;

        --Testing 0 - (3 * 3)
        regB_Data <= (65 downto 64 => '1', others => '0');
        regC_Data <= (65 downto 64 => '1', others => '0');
        regA_Data <= (others => '0');
        wait for period;

        --Testing 0 - (-3 * 3)
        regB_Data <= (65 downto 64 => '1', others => '0');
        regC_Data <= (95 downto 66 => '1', 65 => '0', 64 => '1', others => '0');
        regA_Data <= (others => '0');
        wait for period;

        --Left most 32 bits
        --Testing max negative - (-1 * -1)
        regB_Data <= (127 downto 96 => '1', others => '0');
        regC_Data <= (127 downto 96 => '1', others => '0');
        regA_Data <= (127 => '1', others => '0');
        wait for period;

        --Testing max positive - (1 * -1)
        regB_Data <= (127 downto 96 => '1', others => '0');
        regC_Data <= (96 => '1', others => '0');
        regA_Data <= (126 downto 96 => '1', others => '0');
        wait for period;

        --Testing 0 - (3 * 3)
        regB_Data <= (97 downto 96 => '1', others => '0');
        regC_Data <= (97 downto 96 => '1', others => '0');
        regA_Data <= (others => '0');
        wait for period;

        --Testing 0 - (-3 * 3)
        regB_Data <= (97 downto 96 => '1', others => '0');
        regC_Data <= (127 downto 98 => '1', 97 => '0', 96 => '1', others => '0');
        regA_Data <= (others => '0');
        wait for period;

        wait;
    end process;

end R4_2_tb;
```