1 alu.vhd

```
library ieee;
use     ieee.std_logic_1164.all;                        -- library import
use     ieee.std_logic_signed.all;
use     ieee.numeric_std.all;

entity alu is
    port(a,b:   in  std_logic_vector(31 downto 0 ); --inputs of alu
         opcode:in  std_logic_vector(5 downto 0 );  --opcode of alu
         result:out std_logic_vector(31 downto 0);  --output of alu
         clk:   in  std_logic                        --clock excitation
    );
end entity alu;

architecture sample of alu is
    signal int_a:integer;
    --store interger input numbers
    signal int_b:integer;

    begin
        process (clk)
        --different functions of alu unit
        begin
            int_a<=conv_integer(a);
            --convert Binary number into Decimal number
            int_b<=conv_integer(b);
            case opcode is
            when "000100" =>
            result<= std_logic_vector(to_signed(int_a+int_b, 32));
            -- plus
            when "001101" =>
            result<= std_logic_vector(to_signed(int_a-int_b, 32));
            --  subtraction
            when "001000" =>
                if (int_a<0) then

                    result<= std_logic_vector(to_signed(0-int_a, 32));
                else

                    -- Absolute value of input a
                    result<= std_logic_vector(to_signed(int_a, 32));
                end if;
            when "000111" =>
                result<= std_logic_vector(to_signed(0-int_a, 32));
                -- Opposite number of input a
            when "001010" =>
                if (int_b<0) then
                result<= std_logic_vector(to_signed(0-int_b, 32));
                -- Absolute value of input b
                else
                result<= std_logic_vector(to_signed(int_b, 32));
                end if;
            when "000101" =>
                result<= std_logic_vector(to_signed(0-int_b, 32));
            when "001011" =>
   -- Opposite number of input b
                result<=a or b;                        -- a or b
            when "001100" =>
                result<=not a;                         -- not a
            when "001110" =>
                result<=not b;                         -- not b
            when "000011" =>
                result<=a and b;                    -- result=a and b
            when "001111" =>
                result<=a xor b;                    -- result=a xor b
            when others =>
            end case;
        end process;
end architecture sample;
```