```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.NUMERIC_STD.all;
--------------------------- Declaring the entity of ALU Circuit
--------------------------
entity ALu is
port(alu_op  : in std_logic;                         -- ALU operation control signal
reset,clk    : in std_logic;                         -- clock and reset input
 b_reg       : in  std_logic_vector(7 downto 0);     -- B register(8bit) input
 a_reg       : in  std_logic_vector(7 downto 0);     -- A register(8bit) input
 acc         : out std_logic_vector(7 downto 0);     -- accumulator register(8bit)
output
 alu_control : in  std_logic_vector(2 downto 0);     -- ALU input selector
 flag        : out std_logic);                       -- output flag status of ALU
 end ALu;

architecture Behavioral of ALu is

signal zero,overflow,carry : std_logic;              -- declaraing status_flags signa
signal next_acc: std_logic_vector(8 downto 0);       -- declaraing ALU result signal
begin
-------------------------------------------------------------------------------------
-------
-- Data transfer circuit that control signal from ALU to the Accumlator output with
-- Data transfer circuit is D flip flop with enable signal
process(clk,reset)
 begin
 if( reset = '1') then                              -- when reset is high
 acc <= x"00";                                      -- the accumlator will be
assigned as zero value
 elsif(rising_edge(clk)) then                       -- detect rising edge
 if(alu_op='1') then                                -- when control signal is
high,the accumlator will be assign as next calculated value from ALU
 acc <= next_acc(7 downto 0);
 end if;
 end if;
 end process;
 -- ALU circuits with Case Statement
process(alu_control,b_reg,a_reg,reset,alu_op)
begin
 if(reset = '1') then                              --  if reset logic is high
 next_acc <= x"00" & '0';                          -- evaluate ALU result signal as
Zero
 elsif(alu_op = '1') then                          -- if the control signal is high
 case alu_control is                               -- select the alu operation based
```

```
on alu control selector
 when "000" => next_acc <= '0'
&(std_logic_vector(to_unsigned((to_integer(unsigned(a_reg)) /
to_integer(unsigned(b_reg))),8))); -- A/B
 when "001" => next_acc <= std_logic_vector(('0' & unsigned(a_reg)) + (('0' &
unsigned(b_reg))));  -- A + B
 when "010" => next_acc <= std_logic_vector(('0' & unsigned(a_reg)) - (('0' &
unsigned(b_reg))));     -- A - B
 when "011" => next_acc <= '0' & a_reg(6 downto 0) & '0'; -- A*2
 when "100" => next_acc <= '0' & (a_reg nand b_reg);   -- A NAND B
 when "101" => next_acc <=  "00" & a_reg(7 downto 1);  -- A/2
 when "110" => next_acc <= '0' & (a_reg xor b_reg);    -- A XOR B
 when "111" => next_acc <=  '0' &
(std_logic_vector(to_unsigned((to_integer(unsigned(a_reg)) *
to_integer(unsigned(b_reg))),8))); -- A * B
 when others => next_acc <= (OTHERS => 'Z');
 end case;
 else
 next_acc <= (OTHERS => 'Z');
 end if;
end process;
-------------------------------------------------------------
zero <= '1' when next_acc = x"00" & '0' else '0';                          -- set
zero_flag when result ALU signal is zero
carry <= '1' when next_acc(8) = '1' else '0';                              -- set
carry_flag when carry is set
overflow <= '1' when a_reg = b_reg else '0';                               -- set
overflow_flag when inputs are equal
flag <= '1' when overflow = '1' or carry = '1' or zero = '1' else '0';    -- set  alu
flag when zero_flag or carry_flag or overflow_flag is set
end Behavioral;
```