```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE work.cpu_pack.ALL;

-------------------------------------------------
-- ENTITY: INPUT And OUTPUT decLARATIONS.
-------------------------------------------------

ENTITY fsm IS
 PORT(
    state                       : BUFFER state_type;
    LU_OP                       : OUT    luc_type;
    C_in                        : BUFFER std_logic;
    C_out                       : IN     std_logic;
    Counter_Reset               : OUT    std_logic;
    OP_Code                     : IN     opcode_type;
    Counter_state               : IN     count_type;
    start, RESET, CLK, alu_out0 : IN     std_logic
    );
END fsm;

---------------------------------------------------
-- THE ARCHITECTURAL BODY - BEHAVIORAL DESCRIPTION.
---------------------------------------------------


ARCHITECTURE fsm_arch OF fsm IS

 SIGNAL next_C_in  : std_logic;
 SIGNAL next_state : state_type;

BEGIN

-------------------------------------------------
-- BEGIN FINITE state MACHINE PROCESS
-------------------------------------------------
 FSM_Comb : PROCESS (start, OP_Code, C_out, alu_out0, Counter_state)
 BEGIN
    LU_OP                <= norm_h;
    Counter_Reset        <= '0';
    next_C_in            <= '0';
    next_state           <= state;
    CASE OP_Code IS
-------------------------------------------------------------------------------
    WHEN load   =>
-------------------------------------------------------------------------------
    CASE state IS
        WHEN wait0  =>
            IF (start = '1') THEN
                next_state  <= wait1;
            ELSE
                next_state  <= wait0;
            END IF;
        WHEN wait1  =>
            Counter_Reset <= '1';
            IF(start = '0') THEN
                next_state  <= load8;
```

```
                next_C_in   <= '0';
            ELSE
                next_state  <= wait1;
                next_C_in   <= C_in;
            END IF;
        WHEN load8  =>
            next_state    <= wait0;
            LU_OP         <= norm_h;
            next_C_in     <= '0';
            WHEN OTHERS =>
                next_state    <= wait1;
    END CASE;

-------------------------------------------------------------------------------
    WHEN inc    =>
-------------------------------------------------------------------------------
    CASE state IS
        WHEN wait0 =>
            IF (start = '1') THEN
                next_state <= wait1;
            ELSE
                next_state <= wait0;
            END IF;
        WHEN wait1 =>
            IF (start = '0') THEN
                next_state <= add8;
                next_c_in <= '1';
            ELSE
                next_state <= wait1;
                next_c_in <= c_in;
            END IF;
        WHEN add8 =>
            LU_OP <= zero_h;
            next_c_in <= '0';
            next_state <= wait0;
        WHEN OTHERS =>
            next_state <= wait1;
    END CASE;

-------------------------------------------------------------------------------
    WHEN inc4   =>
-------------------------------------------------------------------------------
    CASE state IS
        WHEN wait0 =>
            IF (start = '1') THEN
                next_state <= wait1;
            ELSE
                next_state <= wait0;
            END IF;
        WHEN wait1 =>
            IF (start = '0') THEN
                next_state <= add8;
                next_c_in <= '1';
                counter_reset <= '1';
            ELSE
                next_state <= wait1;
                next_c_in <= c_in;
```

```
            END IF;
        WHEN add8 =>
            counter_reset <= '0';
            lu_op <= zero_h;
            next_C_in <= '0';
            if (counter_state < 3) THEN
                next_state <= add8;
                next_c_in <= '1';
            else
                next_state <= wait0;
            END IF;
        WHEN OTHERS =>
            next_state <= wait1;
    END CASE;


--------------------------------------------------------------------------------
     WHEN sub   =>
--------------------------------------------------------------------------------

    CASE state IS
        WHEN wait0 =>
            IF (start = '1') THEN
                next_state <= wait1;
            ELSE
                next_state <= wait0;
            END IF;
        WHEN wait1 =>
            Counter_Reset <= '1';
            IF(start = '0') THEN
                next_state <= add8;
                next_C_in <= '1';
            ELSE
                next_state <= wait1;
            END IF;
        WHEN add8 =>
            next_state <= wait0;
            LU_OP <= invert_h;
        WHEN OTHERS =>
            next_state <= wait1;
    END CASE;

--------------------------------------------------------------------------------
    WHEN cmp    =>
--------------------------------------------------------------------------------
    CASE state IS
        WHEN wait0 =>
            IF (start = '1') THEN
                next_state <= wait1;
            ELSE
                next_state <= wait0;
            END IF;
        WHEN wait1 =>
            next_c_in <= '0';
            IF(start = '0') THEN
                next_state <= load8;
            ELSE
```

```
                next_state <= wait1;
            END IF;
        WHEN load8 =>
            next_state <= add8;
            LU_OP <= xor_h;
            next_c_in <= '0';
        WHEN add8 =>
            next_state <= wait0;
            LU_OP <= one_h;
        WHEN OTHERS =>
            next_state <= wait1;
    END CASE;

-------------------------------------------------------------------------------
    WHEN OTHERS     =>
-------------------------------------------------------------------------------
       next_state        <= wait0;
    END CASE;
 END PROCESS FSM_Comb;

-------------------------------------------------
-- BEGIN OUTPUT PROCESS
-------------------------------------------------


 FSM_Reg : PROCESS(clk, reset)
 BEGIN
    IF RESET = '1' THEN
        state <= wait0;
        C_in  <= '0';
    ELSIF(CLK'event AND CLK = '1') THEN
        state <= next_state;
        C_in  <= next_c_in;
    END IF;
 END PROCESS FSM_Reg;

END fsm_arch;
```