```
--Cronometro experimento 7
--Pedro Lucas
--Gabriel Diniz
--Joaquim José
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity cronometro is
    Port (
        clk        : in STD_LOGIC;
        reset      : in STD_LOGIC;      -- ativo em 0 (botão físico)
        start_stop : in STD_LOGIC;      -- ativo em 0 (botão físico)
        seg_anodo  : out STD_LOGIC_VECTOR (3 downto 0); -- lógica inversa
        seg_catodo : out STD_LOGIC_VECTOR (6 downto 0)  -- lógica inversa
    );
end cronometro;

architecture Behavioral of cronometro is

    function display_decoder(bcd : STD_LOGIC_VECTOR(3 downto 0)) return STD_LOGIC_VECTOR is
        variable seg : STD_LOGIC_VECTOR(6 downto 0);
    begin
        case bcd is
            when "0000" => seg := "1000000"; -- 0
            when "0001" => seg := "1111001"; -- 1
            when "0010" => seg := "0100100"; -- 2
            when "0011" => seg := "0110000"; -- 3
            when "0100" => seg := "0011001"; -- 4
            when "0101" => seg := "0010010"; -- 5
            when "0110" => seg := "0000010"; -- 6
            when "0111" => seg := "1111000"; -- 7
            when "1000" => seg := "0000000"; -- 8
            when "1001" => seg := "0010000"; -- 9
            when others => seg := "1111111"; -- off
        end case;
        return seg;
    end function;

    signal clk_div     : STD_LOGIC := '0';
    signal count_1hz   : INTEGER := 0;
    constant MAX_COUNT : INTEGER := 25_000_000;

    signal seconds  : INTEGER range 0 to 59 := 0;
    signal minutes  : INTEGER range 0 to 9 := 0;

    signal running      : STD_LOGIC := '0';
    signal last_btn     : STD_LOGIC := '0';

    signal reset_n      : STD_LOGIC;
    signal start_stop_n : STD_LOGIC;

    signal bcd_0, bcd_1, bcd_2, bcd_3 : STD_LOGIC_VECTOR(3 downto 0);
    signal display_sel     : INTEGER range 0 to 3 := 0;
```

```
    signal refresh_counter : INTEGER range 0 to 99999 := 0;

begin

    reset_n      <= not reset;
    start_stop_n <= not start_stop;

    process(clk)
    begin
        if rising_edge(clk) then
            if count_1hz = MAX_COUNT then
                count_1hz <= 0;
                clk_div <= not clk_div;
            else
                count_1hz <= count_1hz + 1;
            end if;
        end if;
    end process;

    process(clk)
    begin
        if rising_edge(clk) then
            if start_stop_n = '1' and last_btn = '0' then
                running <= not running;
            end if;
            last_btn <= start_stop_n;
        end if;
    end process;

    process(clk_div)
    begin
        if rising_edge(clk_div) then
            if reset_n = '1' then
                seconds <= 0;
                minutes <= 0;
            elsif running = '1' then
                if seconds = 59 then
                    seconds <= 0;
                    if minutes < 9 then
                        minutes <= minutes + 1;
                    else
                        minutes <= 0;
                    end if;
                else
                    seconds <= seconds + 1;
                end if;
            end if;
        end if;
    end process;

    bcd_0 <= std_logic_vector(to_unsigned(seconds mod 10, 4));
    bcd_1 <= std_logic_vector(to_unsigned(seconds / 10, 4));
    bcd_2 <= std_logic_vector(to_unsigned(minutes mod 10, 4));
    bcd_3 <= "0000";
```

```
    process(clk)
    begin
        if rising_edge(clk) then
            if refresh_counter = 50000 then
                refresh_counter <= 0;
                display_sel <= (display_sel + 1) mod 4;
            else
                refresh_counter <= refresh_counter + 1;
            end if;
        end if;
    end process;

    process(display_sel, bcd_0, bcd_1, bcd_2, bcd_3)
    begin
        case display_sel is
            when 0 =>
                seg_anodo  <= "1110";
                seg_catodo <= display_decoder(bcd_0);
            when 1 =>
                seg_anodo  <= "1101";
                seg_catodo <= display_decoder(bcd_1);
            when 2 =>
                seg_anodo  <= "1011";
                seg_catodo <= display_decoder(bcd_2);
            when 3 =>
                seg_anodo  <= "0111";
                seg_catodo <= display_decoder(bcd_3);
        end case;
    end process;

end Behavioral;
```