```
----------------------------------------------------------------------------
---------
-- Company:
-- Engineer:
--
-- Create Date: 11/11/2020 02:35:43 PM
-- Design Name:
-- Module Name: UC - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------
---------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity UC is
    Port (Clk: in std_logic;
          Rst: in std_logic;
          Start: in std_logic;
          Q0: in std_logic;
          LoadA: out std_logic;
          ShlAQ: out std_logic;
          RstA: out std_logic;
          LoadQ: out std_logic;
          SubB: out std_logic;
          LoadB: out std_logic;
          Term: out std_logic);
end UC;

architecture Behavioral of UC is

type TIP_STARE is (idle, init, decision, add, shift, count, stop);
signal stare: TIP_STARE;
signal c: integer;

begin

process1: process(Clk)
```

```
begin
    if (rising_edge(Clk)) then
        if (Rst='1') then
            stare <= idle;
         else
            case stare is
            when idle =>
                if (Start = '1' ) then
                    stare <= init;
                else
                    stare <= idle;
                end if;
            when init =>
                c <= 8;
                stare <= decision;
            when decision =>
               case Q0 is
                when '0' => stare <= shift;
                when '1' => stare <= add;
                when others => stare <=shift;
               end case;
            when add =>
                stare <= shift;
            when shift =>
                c <= c - 1;
                stare <= count;
            when count =>
                if(c=0) then
                    stare <= stop;
                else
                    stare <= decision;
            end if;
            when stop =>
                stare <= idle;
            end case;
        end if;
   end if;
end process;

process2: process(stare)
 begin
        LoadB <='0'; LoadQ <='0'; LoadA<='0'; ShlAQ <='0'; RstA <='0';
SubB <='0'; Term <= '0';
        case Stare is
            when idle =>
                RstA <='0'; ShlAQ <='0'; LoadB <='0'; LoadQ <='0'; SubB
<='0'; Term <= '0';
            when init =>
                RstA <='1'; LoadB <='1'; LoadQ <='1';
            when add =>
                LoadA <= '1';
            when decision =>
                 LoadQ <= '0'; LoadA <= '0'; ShlAQ <= '0'; LoadB <= '0';
RstA <= '0'; SubB <= '0'; Term <= '0';
            when shift =>
                ShlAQ <= '1';
            when count =>
```

```
                LoadQ <= '0'; LoadA <= '0'; ShlAQ <= '0'; LoadB <= '0';
RstA <= '0'; SubB <= '0'; Term <= '0';
            when stop =>
                Term <= '1';
        end case;
 end process;

end Behavioral;
```