This is a solution that was provided by the teacher.
It uses 2 processes instead of 1.
In my opinion it was unnecessary, but it can be useful to show how to use multiple processes.

```
-- This solution splits the problem into 2 processes (pro1, pro2)
-- The first defines the functioning states.
-- The second determines the output and/or the state of the internal signals

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity converter1to2x8 is
      port (ck, sdin, en: in STD_LOGIC;
            OutByte: out STD_LOGIC_VECTOR (7 downto 0) );
end converter1to2x8;

architecture Behavioral of converter1to2x8 is
      -- 4 possible states
      type stato is (S1, S2, S3, S4);
      signal CS: stato;
      -- instantiation of 2 8-bits registers to memorize input data
      signal reg8MSByte, reg8LSByte: STD_LOGIC_VECTOR(7 downto 0);
      signal conta: INTEGER range 0 to 31;

begin
-- 2 distinct processes:
-- The first determines how to pass from a state to the others
-- The second decides what to do in each state

pro1: process(ck,en)
begin
      if (ck'event and ck='1') then
            if en ='1' then
                  conta <= conta+1;
                  if conta <= 7 then
                        CS <= S1;
                  elsif conta > 7 and conta < 15 then
                        CS <=S2;
                  elsif conta = 16 then
                        CS <=S3;
                  elsif conta > 16 then
                        CS <=S4;
                        conta <= 0;
                  end if;
            end if;
      end if;
end process;

pro2: process (CS,conta)
begin

case CS is
      when S1 =>
            -- let the LSBs flow into reg8LSByte
            reg8LSbyte <= sdin & reg8LSbyte (7 downto 1);
            -- output is in high impedance in the meantime
            OutByte <= "ZZZZZZZZ";
```

```
      when S2 =>
            -- let the MSBs flow into reg8LSByte
            reg8MSbyte <= sdin & reg8MSbyte (7 downto 1);
            OutByte <= "ZZZZZZZZ";
      when S3 =>
            -- in S3 the MSBs are transferred to the output
            OutByte(7 downto 0)<= reg8MSbyte(7 downto 0);
      when S4 =>
            -- in S4 the LSBs are transferred to the output
            OutByte(7 downto 0)<= reg8LSbyte(7 downto 0);
      end case;

end process;
end Behavioral;
```