```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity controllore is
  port ( CLK   : in std_logic;
         RESET  : in std_logic;
         REQ    : in std_logic;
         CODE   : in unsigned(7 downto 0);
         DATA_IN: in unsigned(7 downto 0);
         RESULT : out unsigned(15 downto 0);
         BUSY   : out std_logic;
         CID    : out unsigned(1 downto 0)
         );
end controllore;


architecture A of controllore is

  type stato is (idle,read_code,wait4,read_dato,wait4_bis,fine);
  signal ns,cs : stato;
  signal code_en, code_reset : std_logic;
  signal code_ck : unsigned(7 downto 0);
  signal op : std_logic;
  signal c_id : unsigned(1 downto 0);
  signal num_dati : unsigned(4 downto 0);
  signal cicli4, c4_reset, c4_en : std_logic;
  signal c4_in,c4_out : unsigned(1 downto 0);
  signal cdata_in,cdata_out : unsigned(4 downto 0);
  signal cdata_reset : std_logic;
  signal campiona, all_data, campiona_1, campiona_2: std_logic;
  signal result_en : std_logic;
  signal dato_1, dato_2 : unsigned(7 downto 0);
  signal prodotto, acc_out, acc_in, result_tmp : unsigned(15 downto 0);


begin


  -----------------------------------------------------------------------------
  -- 1 parte
  -----------------------------------------------------------------------------

  -- registro per il segnale code
  process(CLK)
  begin
    if CLK'event and CLK='1' then
      if (RESET='1' or code_reset = '1') then
        code_ck <= (others => '0');
      elsif code_en = '1' then
        code_ck <= code;
      end if;
    end if;
  end process;

  op <= code_ck(7);
  cid <= code_ck(6 downto 5);
  num_dati <= code_ck(4 downto 0);


  -- contatore 4 cicli
  process(CLK)
  begin
    if CLK'event and CLK='1' then
      if (RESET='1' or c4_reset = '1') then
        c4_out <= (others => '0');
      elsif c4_en = '1' then
        c4_out <= c4_in;
      end if;
    end if;
  end process;

  c4_in <= c4_out +1;
  cicli4 <= '1' when c4_out = 3 else '0';

  -- contatore dati letti
  process(CLK)
  begin
    if CLK'event and CLK='1' then
      if (RESET='1' or cdata_reset = '1') then
        cdata_out <= (others => '0');
      elsif campiona = '1' then
        cdata_out <= cdata_in;
      end if;
    end if;
  end process;

  cdata_in <= cdata_out +1;
  all_data <= '1' when cdata_out = num_dati else '0';




  -- registro di stato
  process(CLK)
  begin
    if CLK'event and CLK='1' then
      if RESET='1' then
        cs <= idle;
      else
        cs <= ns;
      end if;
    end if;
  end process;
```

```
-- fsm combinatoria
fsm: process (cs,REQ,cicli4,all_data)
begin  -- process fsm
  busy <= '1';
  code_en <= '0';
  c4_en <= '0';
  c4_reset <= '1';
  code_reset <= '0';
  campiona <= '0';
  cdata_reset <= '0';
  result_en <= '0';

  case cs is
    when idle => code_reset <= '1';
                 cdata_reset <= '1';
                 busy <= '0';
                 if req = '1' then
                   ns <= read_code;
                 else
                   ns <= cs;
                 end if;

    when read_code => code_en <= '1';
                      if req = '0' then
                        ns <= wait4;
                      else
                        ns <= cs;
                      end if;

    when wait4 => c4_en <= '1';
                  c4_reset <= '0';
                  if cicli4 = '1' then
                    ns <= read_dato;
                  else
                    ns <= cs;
                  end if;

    when read_dato => c4_en <= '0';
                      campiona <= '1';
                      ns <= wait4_bis;

    when wait4_bis => c4_en <= '1';
                      c4_reset <= '0';


                      if all_data = '1' then
                        ns <= fine;
                      else
                        if cicli4 = '1'  then
                          ns <= wait4;  -- non ho bisogno di reesettare il
                                        -- contatore perchè è a due bit
                        else
                          ns <= cs;
                        end if;

                      end if;

    when fine => result_en <= '1';
                 ns <= idle;

    when others => null;
  end case;

end process fsm;

----------------------------------------------------------------------------
-- 2 parte
----------------------------------------------------------------------------

-- calcolo prodotto
process(CLK)
begin
  if CLK'event and CLK='1' then
    if (RESET='1' or op = '0') then
      dato_1 <= (others => '0');
    elsif campiona_1 = '1' then
      dato_1 <= data_in;
    end if;
  end if;
end process;

process(CLK)
begin
  if CLK'event and CLK='1' then
    if (RESET='1'or op = '0') then
      dato_2 <= (others => '0');
    elsif campiona_2 = '1' then
      dato_2 <= data_in;
    end if;
  end if;
end process;

prodotto <= dato_1 * dato_2;
campiona_1 <= '1' when (campiona = '1' and cdata_out = conv_unsigned(0,5)) else '0';
campiona_2 <= '1' when (campiona = '1' and cdata_out = conv_unsigned(1,5)) else '0';

-- calcolo media
process(CLK)
begin
  if CLK'event and CLK='1' then
    if (RESET='1' or op ='1' or result_en = '1') then
      acc_out <= (others => '0');
    elsif campiona = '1' then
      acc_out <= acc_in;
    end if;
  end if;
```

```
  end process;

  acc_in <= acc_out + data_in;

  -- registro finale
  process(CLK)
  begin
    if CLK'event and CLK='1' then
      if (RESET='1') then
        result <= (others => '0');
      elsif result_en = '1' then
        result <= result_tmp;
      end if;
    end if;
  end process;

  result_tmp <= '0'&acc_out(15 downto 1) when (op = '0' and num_dati = 2 )
              else "00"&acc_out(15 downto 2) when (op = '0' and num_dati = 4 )
              else "000"&acc_out(15 downto 3) when (op = '0' and num_dati = 8 )
              else "0000"&acc_out(15 downto 4) when (op = '0' and num_dati = 16 )
              else "00000"&acc_out(15 downto 5) when (op = '0' and num_dati = 32 )
              else prodotto;


end A;
```