1. For the sample computer architecture shown, give the Register Transfer Notation description of the following:

   a. (4) The instruction fetch steps for any instruction. Assume all instructions fit into a single memory word.
      
      PC→MAR;
      Memory Read;
      Memory→IR;

   b. (4) The instruction decode and execution for the instruction SUBTRACT_FROM_AC.
      
      1. If IR<opcode> = SUBTRACT_FROM_AC then
      2. IR<addr> → MAR
      3. Memory Read
      4. ALU SUBTRACT
      5. ALU result → AC

   c. (2) The housekeeping details to complete the cycle.
      
      PC+1 → PC;
2. Complete the state diagram below for a Mealy machine that implements a full adder. The state machine has two inputs X and Y, representing the two addends respectively, an output S representing the sum, and a state variable C representing the carry (10 points).

![State Diagram]

Ans:
3. Based on the above state diagram, complete the following VHDL implementation of the adder Mealy machine by replacing the questions marks with the correct numbers or words (10 points).

library IEEE;
use IEEE.std_logic_1164.all;

entity adder is
  port (X, Y: in std_logic;
        CLOCK: in std_logic;
        S: out std_logic);
end;

architecture adder_arch OF adder IS
  type State_type is (S0, S1);
  signal C : State_type;
begin
  process(CLOCK)--state machine states and transitions
  begin
    if CLOCK'event and CLOCK='1' then --assume positive edge FF
      CASE C IS
        When S0 => if X='0' and Y='0' then C<=S0; S<='0';
                   elsif X='0' and Y='1' then C<=s0; S<='1';
                   elsif X='1' and Y='0' then C<=S0; S<='1';
                   elsif X='1' and Y='1' then C<=S1; S<='0'; end if;
        When S1 => if X='0' and Y='0' then C<=S0; S<='1';
                   elsif X='0' and Y='1' then C<=S1; S<='0';
                   elsif X='1' and Y='0' then C<=S1; S<='0';
                   elsif X='1' and Y='1' then C<=S1; S<='1'; end if;
        when others =>   C <= S0; S<='0';
      end case;
    end if;
  end process;
end adder_arch