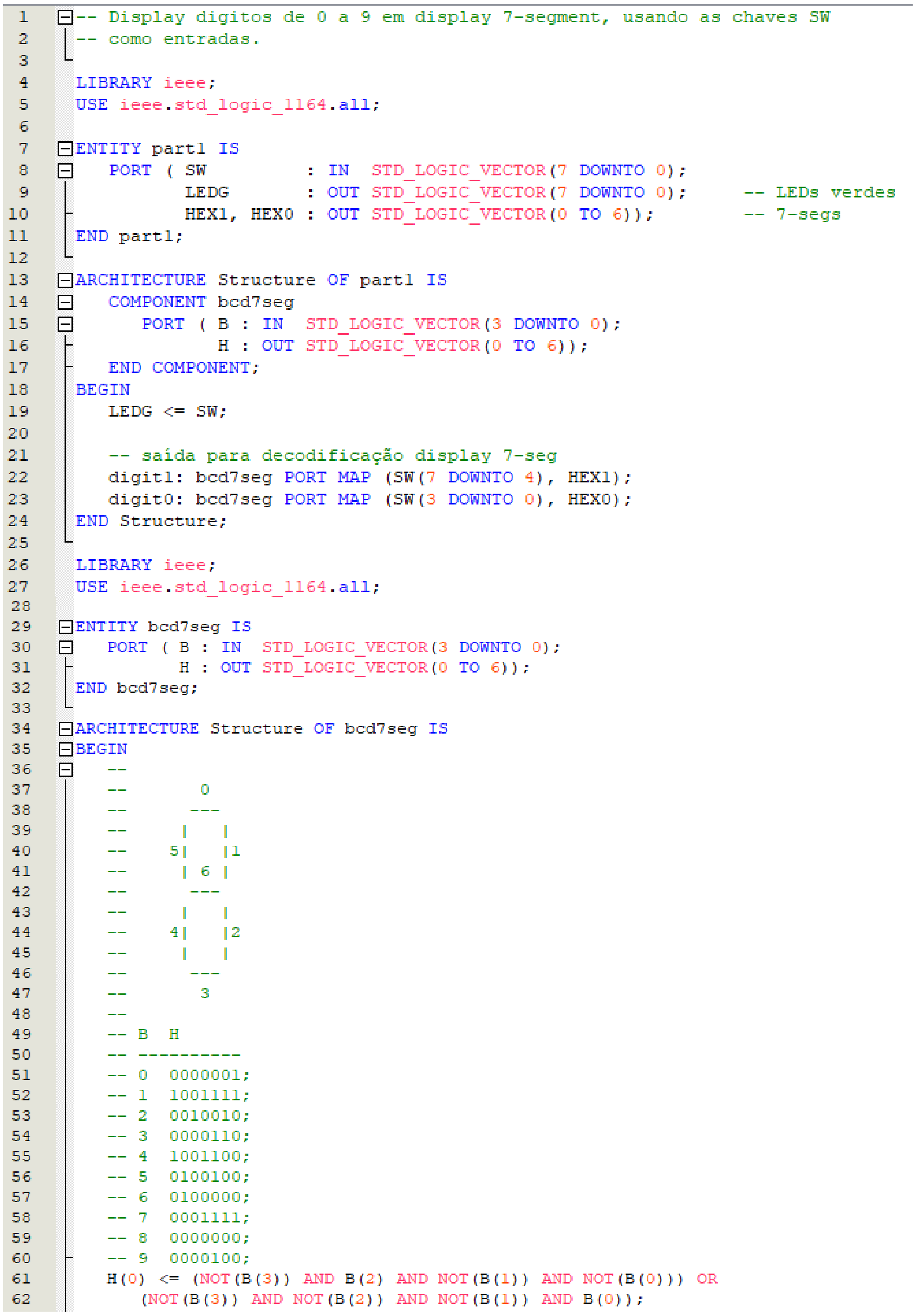
FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

Sistemas Digitais Avançados

Prof. Imbiriba

Solução - Trabalho 2

Parte 1:



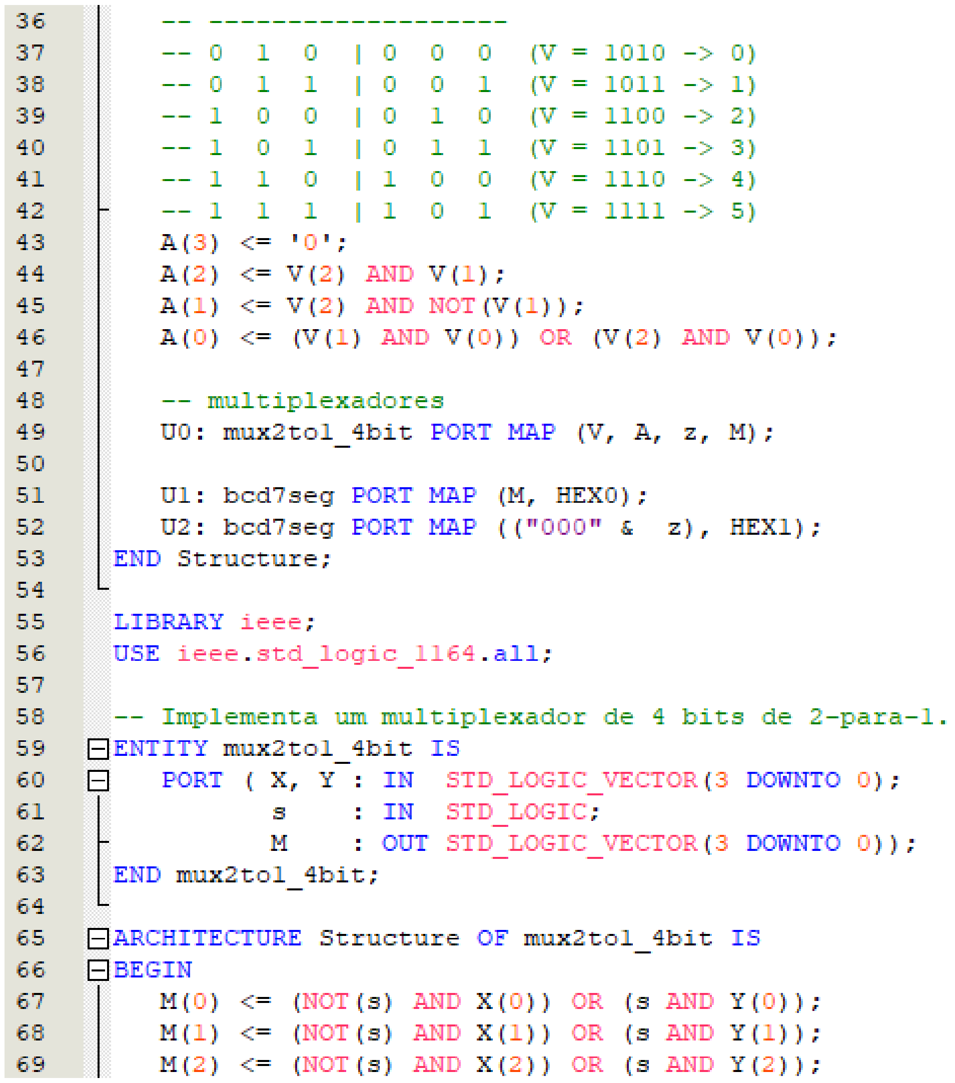
```vhdl
-- Display digitos de 0 a 9 em display 7-segment, usando as chaves SW
-- como entradas.

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY part1 IS
   PORT ( SW          : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          LEDG        : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);     -- LEDs verdes
          HEX1, HEX0  : OUT STD_LOGIC_VECTOR(0 TO 6));        -- 7-segs
END part1;

ARCHITECTURE Structure OF part1 IS
   COMPONENT bcd7seg
      PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
             H : OUT STD_LOGIC_VECTOR(0 TO 6));
   END COMPONENT;
BEGIN
   LEDG <= SW;

   -- saída para decodificação display 7-seg
   digit1: bcd7seg PORT MAP (SW(7 DOWNTO 4), HEX1);
   digit0: bcd7seg PORT MAP (SW(3 DOWNTO 0), HEX0);
END Structure;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY bcd7seg IS
   PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
          H : OUT STD_LOGIC_VECTOR(0 TO 6));
END bcd7seg;

ARCHITECTURE Structure OF bcd7seg IS
BEGIN
   --
   --        0
   --       ---
   --      |   |
   --     5|   |1
   --      | 6 |
   --       ---
   --      |   |
   --     4|   |2
   --      |   |
   --       ---
   --        3
   --
   -- B  H
   -- ----------
   -- 0  0000001;
   -- 1  1001111;
   -- 2  0010010;
   -- 3  0000110;
   -- 4  1001100;
   -- 5  0100100;
   -- 6  0100000;
   -- 7  0001111;
   -- 8  0000000;
   -- 9  0000100;
   H(0) <= (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR
      (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0));
```

```
    H(1) <= (B(2) AND NOT(B(1)) AND B(0)) OR
        (B(2) AND B(1) AND NOT(B(0)));
    H(2) <= (NOT(B(2)) AND B(1) AND NOT(B(0)));
    H(3) <= (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0)) OR
        (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR (NOT(B(3)) AND B(2) AND B(1) AND B(0));
    H(4) <= (NOT(B(1)) AND B(0)) OR (NOT(B(3)) AND B(0)) OR
        (NOT(B(3)) AND B(2) AND NOT(B(1)));
    H(5) <= (B(1) AND B(0)) OR (NOT(B(2)) AND B(1)) OR
        (NOT(B(3)) AND NOT(B(2)) AND B(0));
    H(6) <= (B(2) AND B(1) AND B(0)) OR (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)));
END Structure;
```

Parte 2:

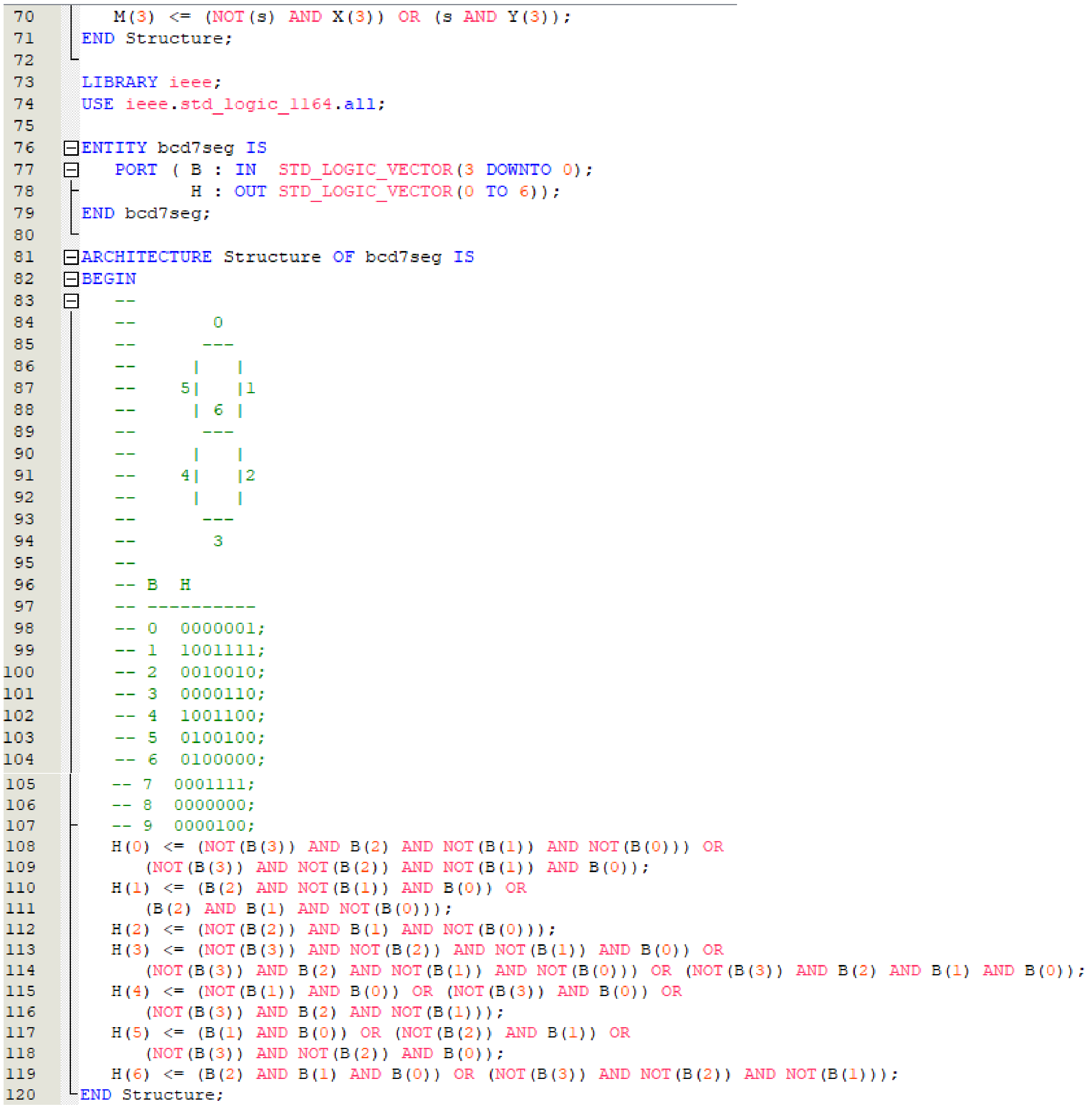
```
-- conversor bcd-para-decimal

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY part2 IS
   PORT ( SW           : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
          HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6));  -- 7-segs
END part2;

ARCHITECTURE Structure OF part2 IS
   COMPONENT mux2to1_4bit
      PORT ( X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
             s    : IN  STD_LOGIC;
             M    : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
   END COMPONENT;
   COMPONENT bcd7seg
      PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
             H : OUT STD_LOGIC_VECTOR(0 TO 6));
   END COMPONENT;
   SIGNAL V, M : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL A : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL z : STD_LOGIC;
BEGIN
   V <= SW;

   -- circuito comparador para V > 9
   z <= (V(3) AND V(2)) OR (V(3) AND V(1));

   -- Circuito A: when V > 9, este circuito permite o display0 - d0 mostrar os valores
   -- de 0 - 5 (para V = 10 até V = 15). Note que V3 = 1 para todos esses valores
   -- e V3 não é necessário para o circuito A. O circuito implementa a seguinte tabela
   -- verdade:
   --
   -- V2 V1 V0 | A2 A1 A0
```

Continua ...



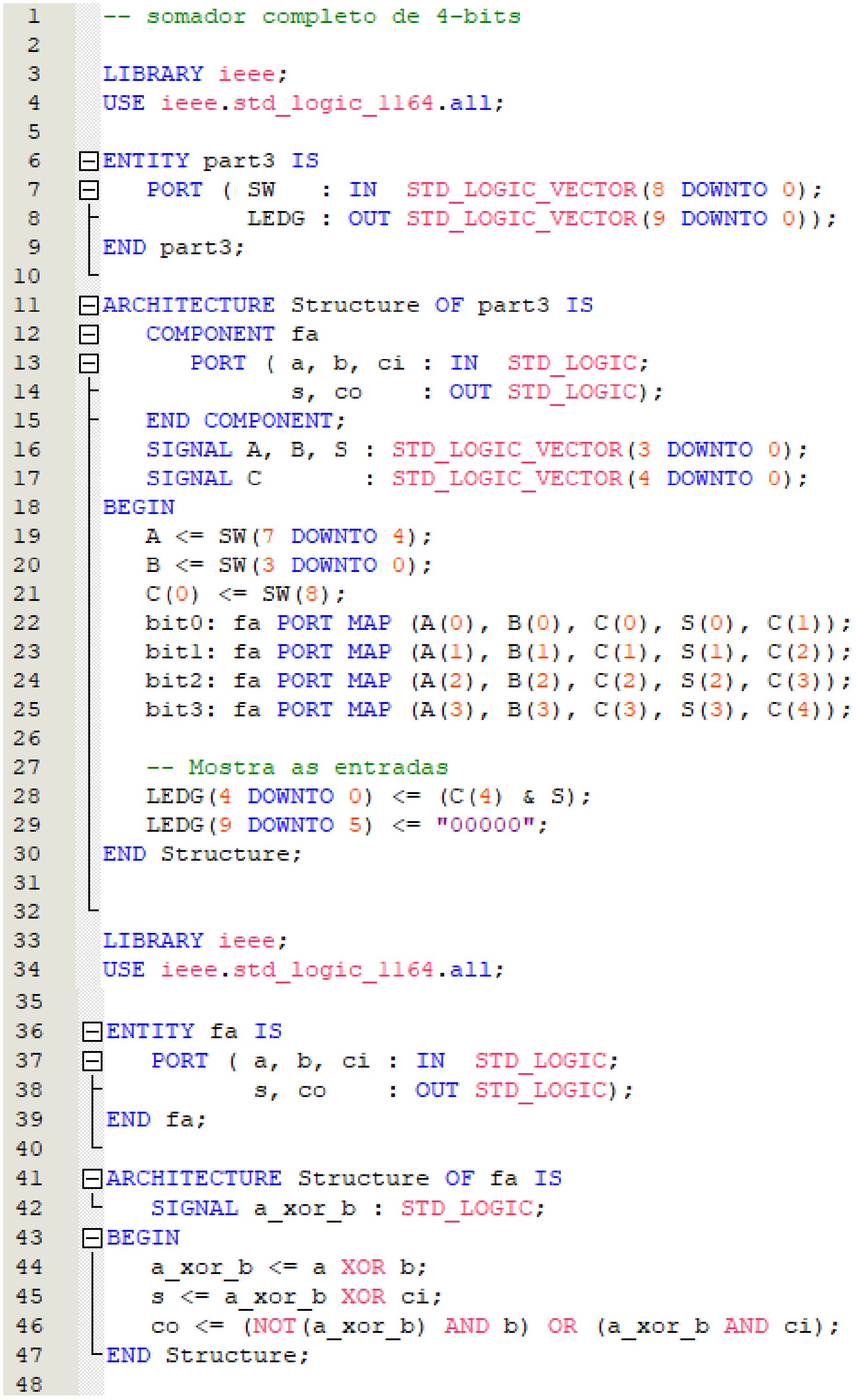
```vhdl
   -- -------------------
   -- 0  1  0  | 0  0  0  (V = 1010 -> 0)
   -- 0  1  1  | 0  0  1  (V = 1011 -> 1)
   -- 1  0  0  | 0  1  0  (V = 1100 -> 2)
   -- 1  0  1  | 0  1  1  (V = 1101 -> 3)
   -- 1  1  0  | 1  0  0  (V = 1110 -> 4)
   -- 1  1  1  | 1  0  1  (V = 1111 -> 5)
   A(3) <= '0';
   A(2) <= V(2) AND V(1);
   A(1) <= V(2) AND NOT(V(1));
   A(0) <= (V(1) AND V(0)) OR (V(2) AND V(0));

   -- multiplexadores
   U0: mux2to1_4bit PORT MAP (V, A, z, M);

   U1: bcd7seg PORT MAP (M, HEX0);
   U2: bcd7seg PORT MAP (("000" &  z), HEX1);
END Structure;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Implementa um multiplexador de 4 bits de 2-para-1.
ENTITY mux2to1_4bit IS
   PORT ( X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
          s    : IN  STD_LOGIC;
          M    : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END mux2to1_4bit;

ARCHITECTURE Structure OF mux2to1_4bit IS
BEGIN
   M(0) <= (NOT(s) AND X(0)) OR (s AND Y(0));
   M(1) <= (NOT(s) AND X(1)) OR (s AND Y(1));
   M(2) <= (NOT(s) AND X(2)) OR (s AND Y(2));
```

Continua ...



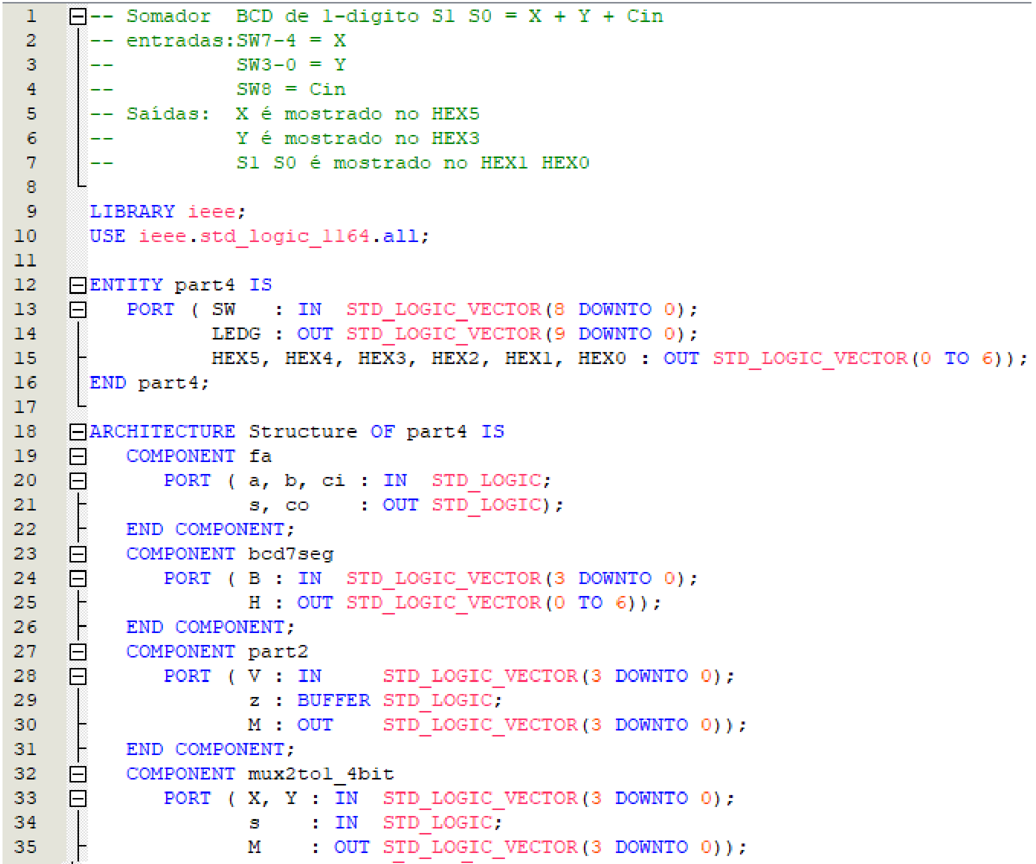
```
    M(3) <= (NOT(s) AND X(3)) OR (s AND Y(3));
END Structure;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY bcd7seg IS
    PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
           H : OUT STD_LOGIC_VECTOR(0 TO 6));
END bcd7seg;

ARCHITECTURE Structure OF bcd7seg IS
BEGIN
    --
    --         0
    --        ---
    --       |   |
    --      5|   |1
    --       | 6 |
    --        ---
    --       |   |
    --      4|   |2
    --       |   |
    --        ---
    --         3
    --
    -- B  H
    -- ----------
    -- 0  0000001;
    -- 1  1001111;
    -- 2  0010010;
    -- 3  0000110;
    -- 4  1001100;
    -- 5  0100100;
    -- 6  0100000;
    -- 7  0001111;
    -- 8  0000000;
    -- 9  0000100;
    H(0) <= (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR
        (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0));
    H(1) <= (B(2) AND NOT(B(1)) AND B(0)) OR
        (B(2) AND B(1) AND NOT(B(0)));
    H(2) <= (NOT(B(2)) AND B(1) AND NOT(B(0)));
    H(3) <= (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0)) OR
        (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR (NOT(B(3)) AND B(2) AND B(1) AND B(0));
    H(4) <= (NOT(B(1)) AND B(0)) OR (NOT(B(3)) AND B(0)) OR
        (NOT(B(3)) AND B(2) AND NOT(B(1)));
    H(5) <= (B(1) AND B(0)) OR (NOT(B(2)) AND B(1)) OR
        (NOT(B(3)) AND NOT(B(2)) AND B(0));
    H(6) <= (B(2) AND B(1) AND B(0)) OR (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)));
END Structure;
```

Parte 3:

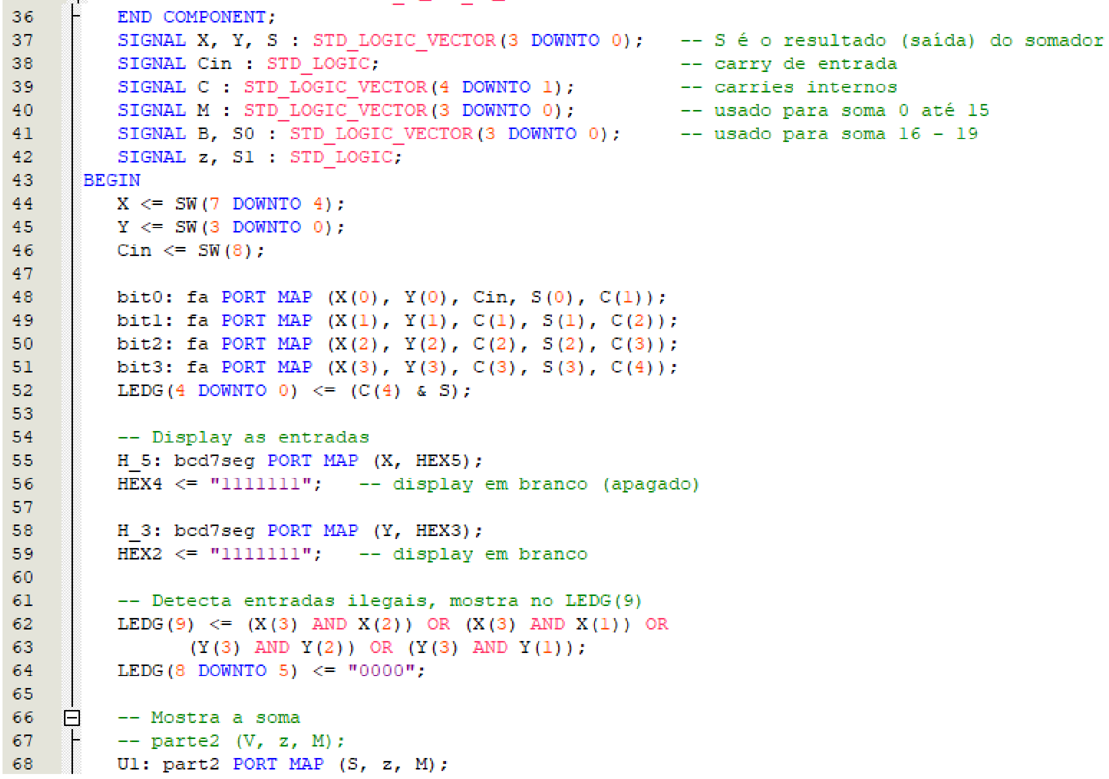


```vhdl
-- somador completo de 4-bits

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY part3 IS
   PORT ( SW   : IN  STD_LOGIC_VECTOR(8 DOWNTO 0);
          LEDG : OUT STD_LOGIC_VECTOR(9 DOWNTO 0));
END part3;

ARCHITECTURE Structure OF part3 IS
   COMPONENT fa
      PORT ( a, b, ci : IN  STD_LOGIC;
             s, co    : OUT STD_LOGIC);
   END COMPONENT;
   SIGNAL A, B, S : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL C       : STD_LOGIC_VECTOR(4 DOWNTO 0);
BEGIN
   A <= SW(7 DOWNTO 4);
   B <= SW(3 DOWNTO 0);
   C(0)  <= SW(8);
   bit0: fa PORT MAP (A(0), B(0), C(0), S(0), C(1));
   bit1: fa PORT MAP (A(1), B(1), C(1), S(1), C(2));
   bit2: fa PORT MAP (A(2), B(2), C(2), S(2), C(3));
   bit3: fa PORT MAP (A(3), B(3), C(3), S(3), C(4));

   -- Mostra as entradas
   LEDG(4 DOWNTO 0) <= (C(4) & S);
   LEDG(9 DOWNTO 5) <= "00000";
END Structure;


LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY fa IS
   PORT ( a, b, ci : IN  STD_LOGIC;
          s, co    : OUT STD_LOGIC);
END fa;

ARCHITECTURE Structure OF fa IS
   SIGNAL a_xor_b : STD_LOGIC;
BEGIN
   a_xor_b <= a XOR b;
   s <= a_xor_b XOR ci;
   co <= (NOT(a_xor_b) AND b) OR (a_xor_b AND ci);
END Structure;
```

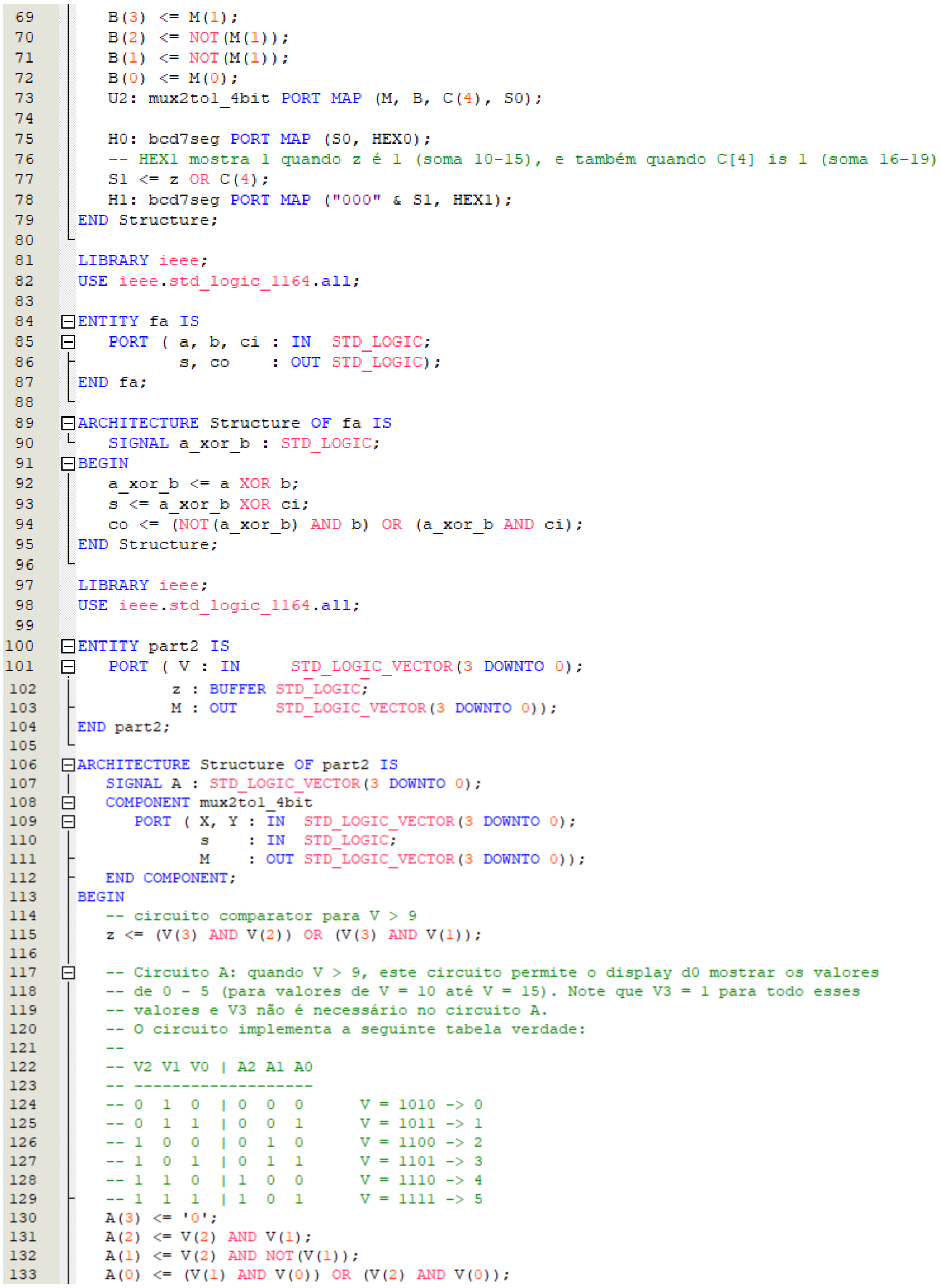
Parte4:



```vhdl
-- Somador  BCD de 1-digito S1 S0 = X + Y + Cin
-- entradas:SW7-4 = X
--          SW3-0 = Y
--          SW8 = Cin
-- Saídas:  X é mostrado no HEX5
--          Y é mostrado no HEX3
--          S1 S0 é mostrado no HEX1 HEX0

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY part4 IS
   PORT ( SW    : IN  STD_LOGIC_VECTOR(8 DOWNTO 0);
          LEDG : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
          HEX5, HEX4, HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6));
END part4;

ARCHITECTURE Structure OF part4 IS
   COMPONENT fa
      PORT ( a, b, ci : IN  STD_LOGIC;
             s, co     : OUT STD_LOGIC);
   END COMPONENT;
   COMPONENT bcd7seg
      PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
             H : OUT STD_LOGIC_VECTOR(0 TO 6));
   END COMPONENT;
   COMPONENT part2
      PORT ( V : IN     STD_LOGIC_VECTOR(3 DOWNTO 0);
             z : BUFFER STD_LOGIC;
             M : OUT    STD_LOGIC_VECTOR(3 DOWNTO 0));
   END COMPONENT;
   COMPONENT mux2to1_4bit
      PORT ( X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
             s     : IN  STD_LOGIC;
             M     : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
```

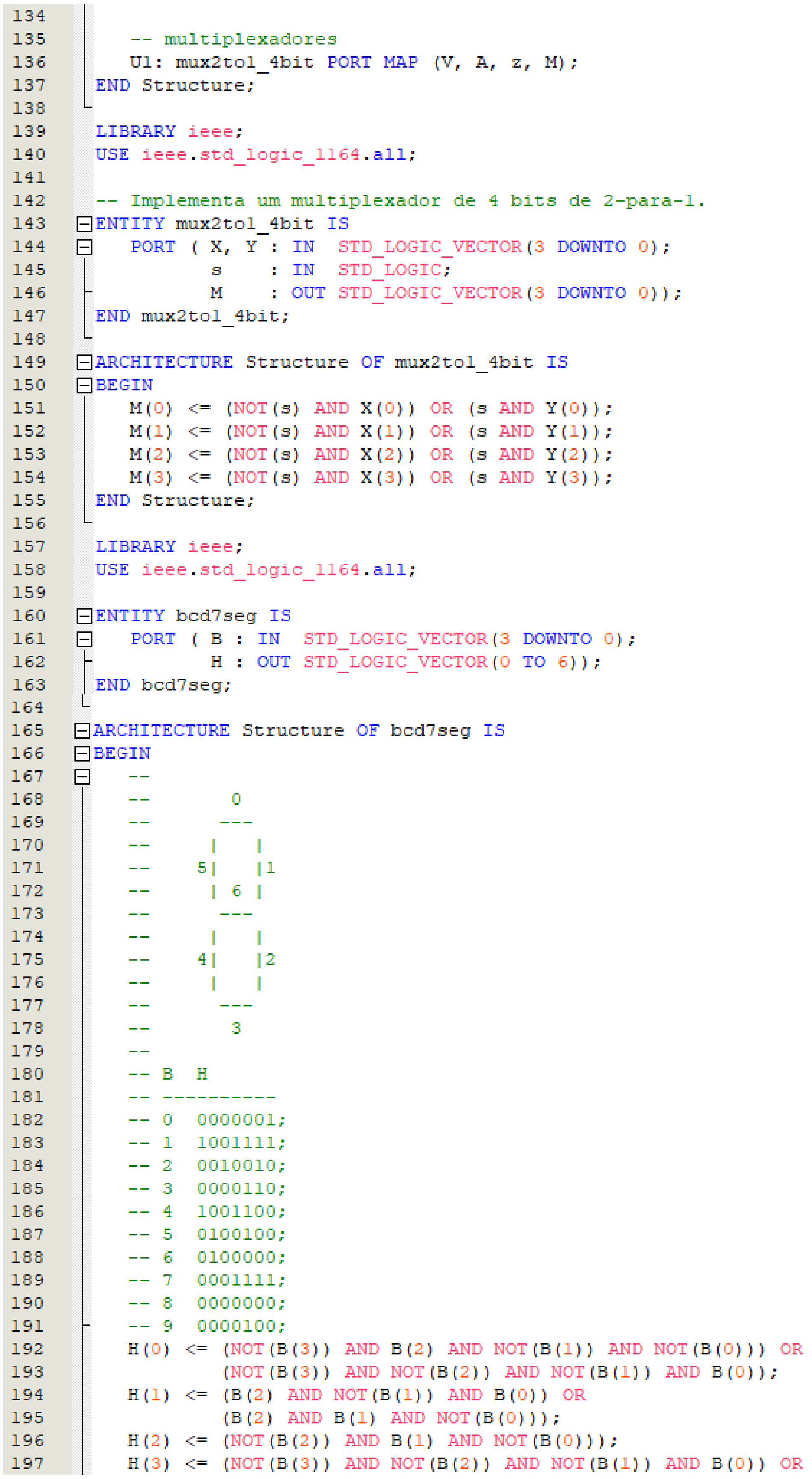


```vhdl
   END COMPONENT;
   SIGNAL X, Y, S : STD_LOGIC_VECTOR(3 DOWNTO 0);   -- S é o resultado (saída) do somador
   SIGNAL Cin : STD_LOGIC;                           -- carry de entrada
   SIGNAL C : STD_LOGIC_VECTOR(4 DOWNTO 1);          -- carries internos
   SIGNAL M : STD_LOGIC_VECTOR(3 DOWNTO 0);          -- usado para soma 0 até 15
   SIGNAL B, S0 : STD_LOGIC_VECTOR(3 DOWNTO 0);      -- usado para soma 16 - 19
   SIGNAL z, S1 : STD_LOGIC;
BEGIN
   X <= SW(7 DOWNTO 4);
   Y <= SW(3 DOWNTO 0);
   Cin <= SW(8);

   bit0: fa PORT MAP (X(0), Y(0), Cin, S(0), C(1));
   bit1: fa PORT MAP (X(1), Y(1), C(1), S(1), C(2));
   bit2: fa PORT MAP (X(2), Y(2), C(2), S(2), C(3));
   bit3: fa PORT MAP (X(3), Y(3), C(3), S(3), C(4));
   LEDG(4 DOWNTO 0) <= (C(4) & S);

   -- Display as entradas
   H_5: bcd7seg PORT MAP (X, HEX5);
   HEX4 <= "1111111";   -- display em branco (apagado)

   H_3: bcd7seg PORT MAP (Y, HEX3);
   HEX2 <= "1111111";   -- display em branco

   -- Detecta entradas ilegais, mostra no LEDG(9)
   LEDG(9) <= (X(3) AND X(2)) OR (X(3) AND X(1)) OR
          (Y(3) AND Y(2)) OR (Y(3) AND Y(1));
   LEDG(8 DOWNTO 5) <= "0000";

   -- Mostra a soma
   -- parte2 (V, z, M);
   U1: part2 PORT MAP (S, z, M);
```



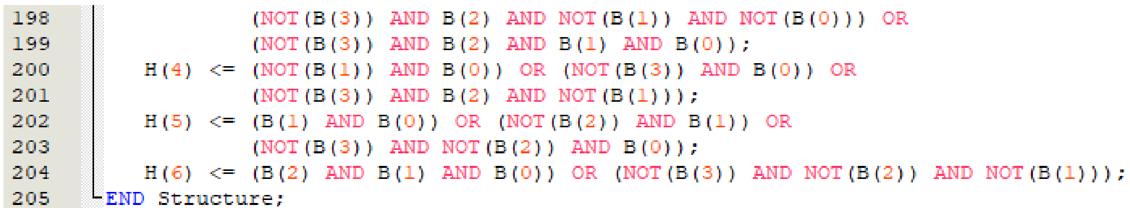
```
    B(3) <= M(1);
    B(2) <= NOT(M(1));
    B(1) <= NOT(M(1));
    B(0) <= M(0);
    U2: mux2to1_4bit PORT MAP (M, B, C(4), S0);

    H0: bcd7seg PORT MAP (S0, HEX0);
    -- HEX1 mostra 1 quando z é 1 (soma 10-15), e também quando C[4] is 1 (soma 16-19)
    S1 <= z OR C(4);
    H1: bcd7seg PORT MAP ("000" & S1, HEX1);
END Structure;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY fa IS
    PORT ( a, b, ci : IN  STD_LOGIC;
           s, co    : OUT STD_LOGIC);
END fa;

ARCHITECTURE Structure OF fa IS
    SIGNAL a_xor_b : STD_LOGIC;
BEGIN
    a_xor_b <= a XOR b;
    s <= a_xor_b XOR ci;
    co <= (NOT(a_xor_b) AND b) OR (a_xor_b AND ci);
END Structure;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY part2 IS
    PORT ( V : IN     STD_LOGIC_VECTOR(3 DOWNTO 0);
           z : BUFFER STD_LOGIC;
           M : OUT    STD_LOGIC_VECTOR(3 DOWNTO 0));
END part2;

ARCHITECTURE Structure OF part2 IS
    SIGNAL A : STD_LOGIC_VECTOR(3 DOWNTO 0);
    COMPONENT mux2to1_4bit
        PORT ( X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
               s    : IN  STD_LOGIC;
               M    : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
    END COMPONENT;
BEGIN
    -- circuito comparator para V > 9
    z <= (V(3) AND V(2)) OR (V(3) AND V(1));

    -- Circuito A: quando V > 9, este circuito permite o display d0 mostrar os valores
    -- de 0 - 5 (para valores de V = 10 até V = 15). Note que V3 = 1 para todo esses
    -- valores e V3 não é necessário no circuito A.
    -- O circuito implementa a seguinte tabela verdade:
    --
    -- V2 V1 V0 | A2 A1 A0
    -- -------------------
    -- 0  1  0  | 0  0  0       V = 1010 -> 0
    -- 0  1  1  | 0  0  1       V = 1011 -> 1
    -- 1  0  0  | 0  1  0       V = 1100 -> 2
    -- 1  0  1  | 0  1  1       V = 1101 -> 3
    -- 1  1  0  | 1  0  0       V = 1110 -> 4
    -- 1  1  1  | 1  0  1       V = 1111 -> 5
    A(3) <= '0';
    A(2) <= V(2) AND V(1);
    A(1) <= V(2) AND NOT(V(1));
    A(0) <= (V(1) AND V(0)) OR (V(2) AND V(0));
```

Continua ...



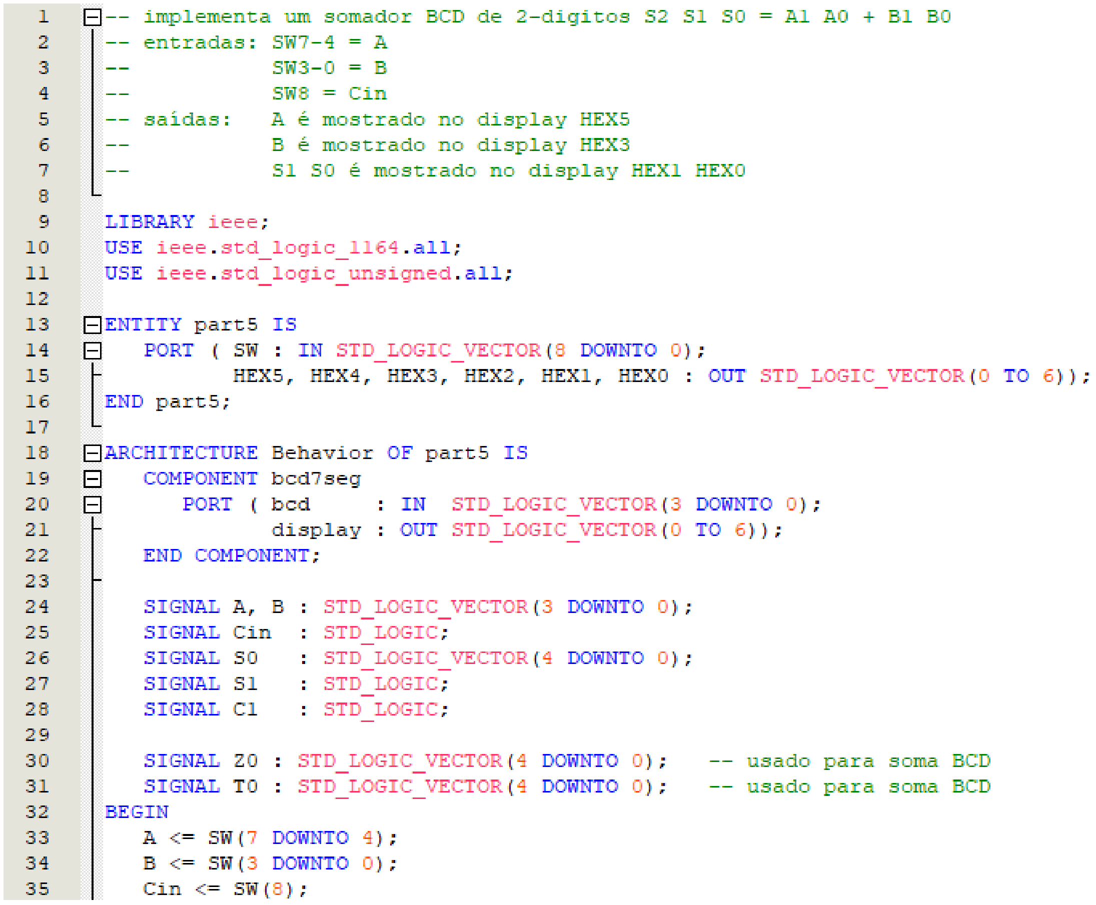
```
   -- multiplexadores
   U1: mux2to1_4bit PORT MAP (V, A, z, M);
END Structure;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Implementa um multiplexador de 4 bits de 2-para-1.
ENTITY mux2to1_4bit IS
   PORT ( X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
          s    : IN  STD_LOGIC;
          M    : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END mux2to1_4bit;

ARCHITECTURE Structure OF mux2to1_4bit IS
BEGIN
   M(0) <= (NOT(s) AND X(0)) OR (s AND Y(0));
   M(1) <= (NOT(s) AND X(1)) OR (s AND Y(1));
   M(2) <= (NOT(s) AND X(2)) OR (s AND Y(2));
   M(3) <= (NOT(s) AND X(3)) OR (s AND Y(3));
END Structure;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY bcd7seg IS
   PORT ( B : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
          H : OUT STD_LOGIC_VECTOR(0 TO 6));
END bcd7seg;

ARCHITECTURE Structure OF bcd7seg IS
BEGIN
   --
   --       0
   --      ---
   --     |   |
   --    5|   |1
   --     | 6 |
   --      ---
   --     |   |
   --    4|   |2
   --     |   |
   --      ---
   --       3
   --
   -- B  H
   -- ----------
   -- 0  0000001;
   -- 1  1001111;
   -- 2  0010010;
   -- 3  0000110;
   -- 4  1001100;
   -- 5  0100100;
   -- 6  0100000;
   -- 7  0001111;
   -- 8  0000000;
   -- 9  0000100;
   H(0) <= (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR
           (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0));
   H(1) <= (B(2) AND NOT(B(1)) AND B(0)) OR
           (B(2) AND B(1) AND NOT(B(0)));
   H(2) <= (NOT(B(2)) AND B(1) AND NOT(B(0)));
   H(3) <= (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)) AND B(0)) OR
```

Continua ...



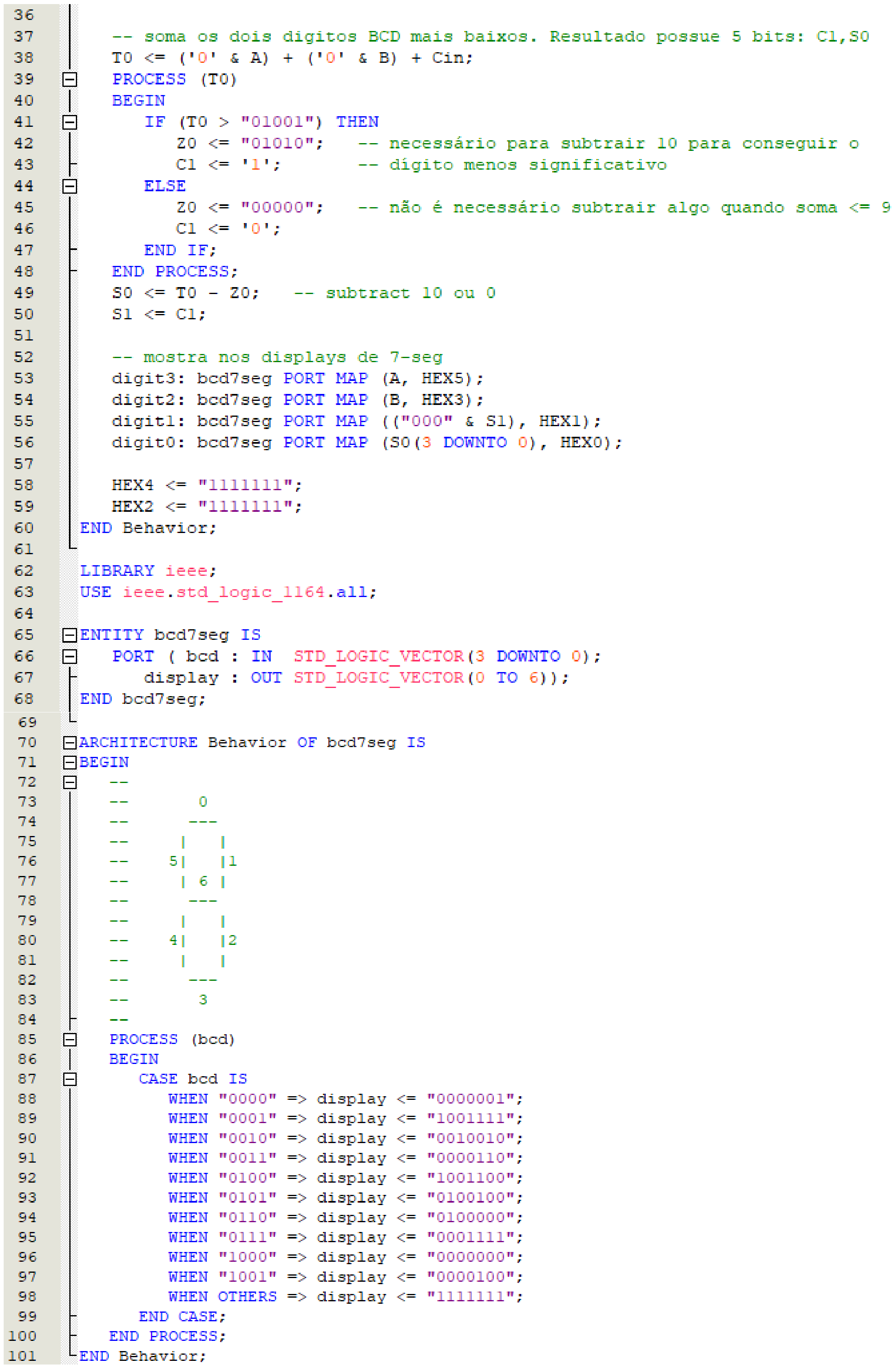
```
           (NOT(B(3)) AND B(2) AND NOT(B(1)) AND NOT(B(0))) OR
           (NOT(B(3)) AND B(2) AND B(1) AND B(0));
   H(4) <= (NOT(B(1)) AND B(0)) OR (NOT(B(3)) AND B(0)) OR
           (NOT(B(3)) AND B(2) AND NOT(B(1)));
   H(5) <= (B(1) AND B(0)) OR (NOT(B(2)) AND B(1)) OR
           (NOT(B(3)) AND NOT(B(2)) AND B(0));
   H(6) <= (B(2) AND B(1) AND B(0)) OR (NOT(B(3)) AND NOT(B(2)) AND NOT(B(1)));
END Structure;
```

Parte5:



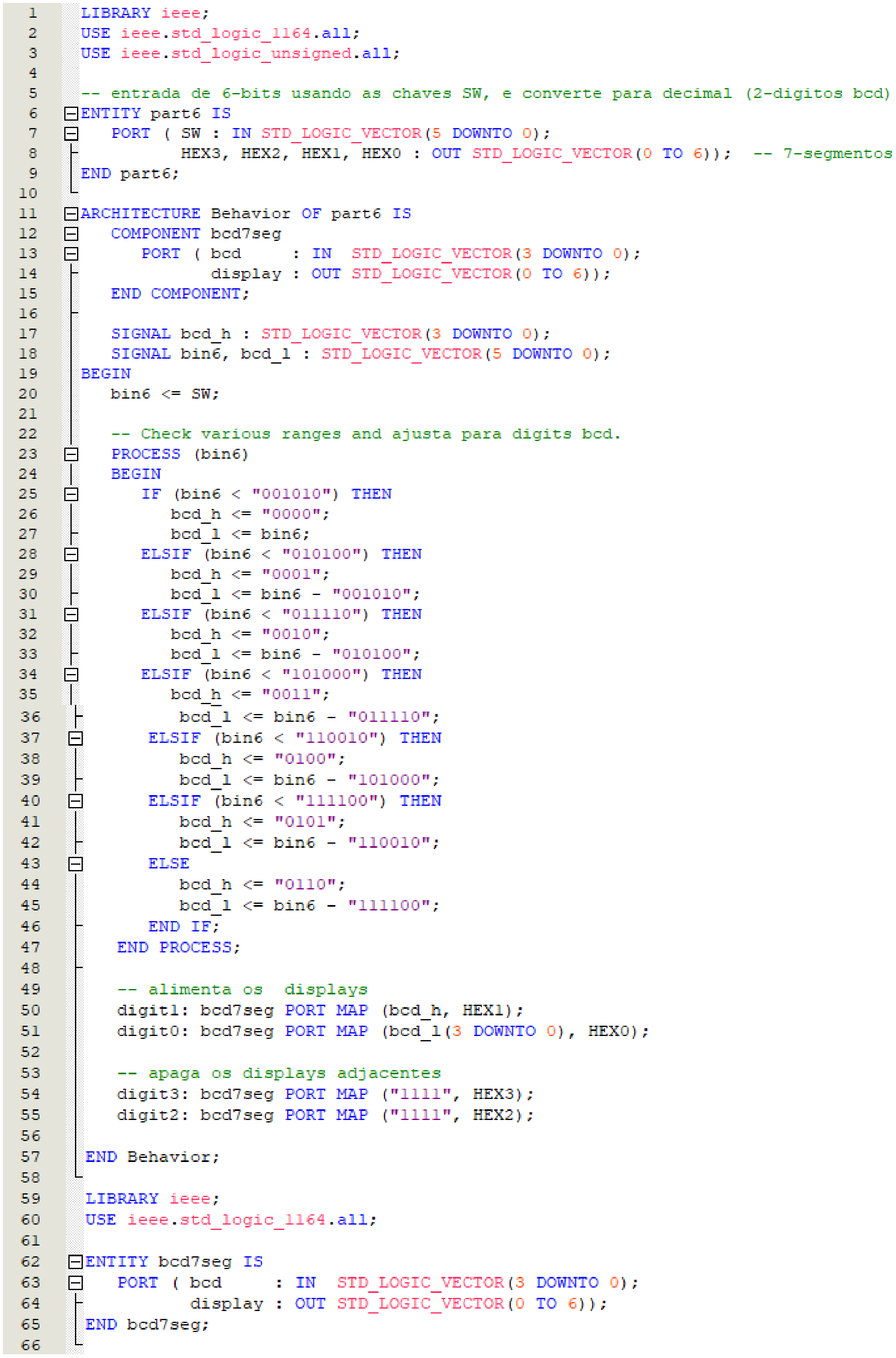
```
-- implementa um somador BCD de 2-digitos S2 S1 S0 = A1 A0 + B1 B0
-- entradas: SW7-4 = A
--           SW3-0 = B
--           SW8 = Cin
-- saídas:   A é mostrado no display HEX5
--           B é mostrado no display HEX3
--           S1 S0 é mostrado no display HEX1 HEX0

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY part5 IS
   PORT ( SW : IN STD_LOGIC_VECTOR(8 DOWNTO 0);
          HEX5, HEX4, HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6));
END part5;

ARCHITECTURE Behavior OF part5 IS
   COMPONENT bcd7seg
      PORT ( bcd     : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
             display : OUT STD_LOGIC_VECTOR(0 TO 6));
   END COMPONENT;

   SIGNAL A, B : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL Cin  : STD_LOGIC;
   SIGNAL S0   : STD_LOGIC_VECTOR(4 DOWNTO 0);
   SIGNAL S1   : STD_LOGIC;
   SIGNAL C1   : STD_LOGIC;

   SIGNAL Z0 : STD_LOGIC_VECTOR(4 DOWNTO 0);   -- usado para soma BCD
   SIGNAL T0 : STD_LOGIC_VECTOR(4 DOWNTO 0);   -- usado para soma BCD
BEGIN
   A <= SW(7 DOWNTO 4);
   B <= SW(3 DOWNTO 0);
   Cin <= SW(8);
```

Continua ...



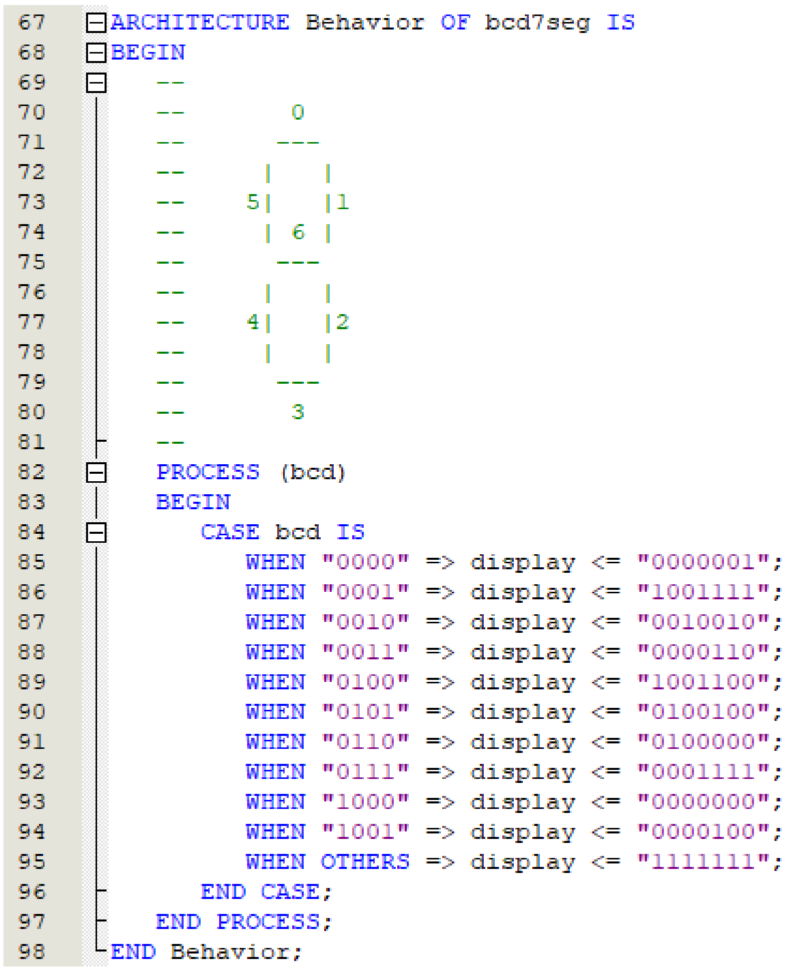
```vhdl

   -- soma os dois digitos BCD mais baixos. Resultado possue 5 bits: C1,S0
   T0 <= ('0' & A) + ('0' & B) + Cin;
   PROCESS (T0)
   BEGIN
      IF (T0 > "01001") THEN
         Z0 <= "01010";   -- necessário para subtrair 10 para conseguir o
         C1 <= '1';       -- dígito menos significativo
      ELSE
         Z0 <= "00000";   -- não é necessário subtrair algo quando soma <= 9
         C1 <= '0';
      END IF;
   END PROCESS;
   S0 <= T0 - Z0;   -- subtract 10 ou 0
   S1 <= C1;

   -- mostra nos displays de 7-seg
   digit3: bcd7seg PORT MAP (A, HEX5);
   digit2: bcd7seg PORT MAP (B, HEX3);
   digit1: bcd7seg PORT MAP (("000" & S1), HEX1);
   digit0: bcd7seg PORT MAP (S0(3 DOWNTO 0), HEX0);

   HEX4 <= "1111111";
   HEX2 <= "1111111";
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY bcd7seg IS
   PORT ( bcd : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
      display : OUT STD_LOGIC_VECTOR(0 TO 6));
END bcd7seg;

ARCHITECTURE Behavior OF bcd7seg IS
BEGIN
   --
   --       0
   --      ---
   --     |   |
   --    5|   |1
   --     | 6 |
   --      ---
   --     |   |
   --    4|   |2
   --     |   |
   --      ---
   --       3
   --
   PROCESS (bcd)
   BEGIN
      CASE bcd IS
         WHEN "0000" => display <= "0000001";
         WHEN "0001" => display <= "1001111";
         WHEN "0010" => display <= "0010010";
         WHEN "0011" => display <= "0000110";
         WHEN "0100" => display <= "1001100";
         WHEN "0101" => display <= "0100100";
         WHEN "0110" => display <= "0100000";
         WHEN "0111" => display <= "0001111";
         WHEN "1000" => display <= "0000000";
         WHEN "1001" => display <= "0000100";
         WHEN OTHERS => display <= "1111111";
      END CASE;
   END PROCESS;
END Behavior;
```

Parte6:



```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

-- entrada de 6-bits usando as chaves SW, e converte para decimal (2-digitos bcd)
ENTITY part6 IS
   PORT ( SW : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
          HEX3, HEX2, HEX1, HEX0 : OUT STD_LOGIC_VECTOR(0 TO 6));  -- 7-segmentos
END part6;

ARCHITECTURE Behavior OF part6 IS
   COMPONENT bcd7seg
      PORT ( bcd     : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
             display : OUT STD_LOGIC_VECTOR(0 TO 6));
   END COMPONENT;

   SIGNAL bcd_h : STD_LOGIC_VECTOR(3 DOWNTO 0);
   SIGNAL bin6, bcd_l : STD_LOGIC_VECTOR(5 DOWNTO 0);
BEGIN
   bin6 <= SW;

   -- Check various ranges and ajusta para digits bcd.
   PROCESS (bin6)
   BEGIN
      IF (bin6 < "001010") THEN
         bcd_h <= "0000";
         bcd_l <= bin6;
      ELSIF (bin6 < "010100") THEN
         bcd_h <= "0001";
         bcd_l <= bin6 - "001010";
      ELSIF (bin6 < "011110") THEN
         bcd_h <= "0010";
         bcd_l <= bin6 - "010100";
      ELSIF (bin6 < "101000") THEN
         bcd_h <= "0011";
         bcd_l <= bin6 - "011110";
      ELSIF (bin6 < "110010") THEN
         bcd_h <= "0100";
         bcd_l <= bin6 - "101000";
      ELSIF (bin6 < "111100") THEN
         bcd_h <= "0101";
         bcd_l <= bin6 - "110010";
      ELSE
         bcd_h <= "0110";
         bcd_l <= bin6 - "111100";
      END IF;
   END PROCESS;

   -- alimenta os  displays
   digit1: bcd7seg PORT MAP (bcd_h, HEX1);
   digit0: bcd7seg PORT MAP (bcd_l(3 DOWNTO 0), HEX0);

   -- apaga os displays adjacentes
   digit3: bcd7seg PORT MAP ("1111", HEX3);
   digit2: bcd7seg PORT MAP ("1111", HEX2);

END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY bcd7seg IS
   PORT ( bcd     : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
          display : OUT STD_LOGIC_VECTOR(0 TO 6));
END bcd7seg;

```

Continua ...



```vhdl
ARCHITECTURE Behavior OF bcd7seg IS
BEGIN
   --
   --        0
   --       ---
   --      |   |
   --     5|   |1
   --      | 6 |
   --       ---
   --      |   |
   --     4|   |2
   --      |   |
   --       ---
   --        3
   --
   PROCESS (bcd)
   BEGIN
      CASE bcd IS
         WHEN "0000" => display <= "0000001";
         WHEN "0001" => display <= "1001111";
         WHEN "0010" => display <= "0010010";
         WHEN "0011" => display <= "0000110";
         WHEN "0100" => display <= "1001100";
         WHEN "0101" => display <= "0100100";
         WHEN "0110" => display <= "0100000";
         WHEN "0111" => display <= "0001111";
         WHEN "1000" => display <= "0000000";
         WHEN "1001" => display <= "0000100";
         WHEN OTHERS => display <= "1111111";
      END CASE;
   END PROCESS;
END Behavior;
```