1. (6 points) What is the dual of the expression \((A + C)'D + (B' + D)(A + C') + AB'\)?

\[ ((AC)' + D)(B'D + AC')(A + B') \]

What is its complement?

\[ ((A'C')' + D')(BD' + A'C)(A' + B) = (A + C + D')(BD' + A'C)(A' + B) \]

2. (6 points) The expression \(AB' + (A' + B)C\) has the minterm \(A'BC\) and the maxterm \((A + B + C)\). List all its other minterms and maxterms.

<table>
<thead>
<tr>
<th>(ABC)</th>
<th>(X)</th>
<th>minterms: (A'B'C), (AB'C), (AB'C), (ABC)</th>
<th>maxterms: ((A + B' + C)), ((A' + B' + C))</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. (12 points) Suppose that the value in the accumulator is 1abc before the first instruction in the group of (partially-specified) instructions shown below is executed and that after two instructions have been executed, the value is e545. Show how the missing hex digits in the first two instructions can be completed in a way that is consistent with this result (assume that all instructions are well-formed). The list of instructions is given below.

<table>
<thead>
<tr>
<th>addr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0130</td>
<td>0001</td>
</tr>
<tr>
<td>0131</td>
<td>a130</td>
</tr>
<tr>
<td>0132</td>
<td>9130</td>
</tr>
<tr>
<td>0133</td>
<td>a131</td>
</tr>
</tbody>
</table>

Suppose that after four instructions are executed, the value of the program counter is 0131. Show how to complete the third and fourth instructions in a way that is consistent with this result.

0000 halt – halt execution
0001 negate – \( ACC := -ACC \)
1xxx immediate load – if sign bit of xxx is 0 then \( ACC := 0xxx \) else \( ACC := fxxx \)
2xxx direct load – \( ACC := M[0xxx] \)
3xxx indirect load – \( ACC := M[M[0xxx]] \)
4xxx direct store – \( M[0xxx] := ACC \)
5xxx indirect store – \( M[M[0xxx]] := ACC \)
6xxx branch – \( PC := 0xxx \)
7xxx branch if zero – if \( ACC = 0 \) then \( PC := 0xxx \)
8xxx branch if positive – if \( ACC > 0 \) then \( PC := 0xxx \)
9xxx branch if negative – if \( ACC < 0 \) then \( PC := 0xxx \)
axxx add – \( ACC := ACC + M[0xxx] \)
4. (12 points) Draw a logic circuit using the smallest possible number of simple gates (AND, OR and inverters, only) for the logic expression $UX' + X(V + Z') + (V' + U)X'Z'$.

How many transistors are required by a CMOS version of this circuit? Show how to improve it by using NAND and NOR gates. How many transistors does this version require?

*The circuit above requires 36 transistors if implemented directly in CMOS. The circuit shown below uses just 24 transistors.*
5. (12 points) Use a Karnaugh map to find the simplest **sum-of-products** expression for
\[ F(X,Y,Z) = \Sigma m(1,2,4), \quad d(X,Y,Z) = \Sigma m(3,6) \]

Use a Karnaugh map to find the simplest **product-of-sums** expression for
\[ F(A,B,C,D) = \Sigma m(1,3,6,7,8,10,15), \quad d(A,B,C,D) = \Sigma m(2,4,5,11) \]
6. (10 points) Draw a logic diagram that directly corresponds to the VHDL module shown below. Use simple gates only.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity foobar is port (
    X: in std_logic_vector(3 downto 0);
    Y: in std_logic;
    Z: out std_logic_vector(2 downto 0));
end foobar;

architecture a1 of foobar is
begin
    process(A, X, Y) begin
        A(2) <= X(3) xor Y;
        for i in 0 to 1 loop
            A(i) <= X(i) or A(i+1);
        end loop;
    end process;
    Z(0) <= A(2);
    Z(2 downto 1) <= A(1 downto 0);
end a1;
```

![Logic Diagram](image-url)
7. (15 points) The circuit shown below implements a ternary (base 3) half-adder. The pair of input bits \((A_i, B_i)\) represents a single ternary digit (the bit pair 00 represent the ternary digit 0, the bit pair 01 represents the ternary digit 1 and the bit pair 10 represents the ternary digit 2). Similarly for the outputs \((X_i, Y_i)\). We can build a ternary increment circuit by combining these ternary half-adder circuits together, in the same way as with a binary ripple-carry increment circuit. Suppose that the input presented to a ternary increment circuit with four ternary digits corresponds to the ternary value 2102. What are the values of the nine output bits? Fill in your answer below the output signals listed below.

\[
\begin{array}{cccccccc}
\text{Cout, (X}_3, Y_3), (X}_2, Y_2), (X}_1, Y_1), (X}_0, Y_0) \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
\end{array}
\]
The partial VHDL module below implements a ternary increment circuit with 8 ternary digits. Complete the missing parts. Your VHDL should be complete and syntactically correct.

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

entity ternaryInc is
  Port (
    A, B : in std_logic_vector(7 downto 0);
    Cin : in std_logic;
    X, Y : out std_logic_vector(7 downto 0);
    Cout : out std_logic
  );
end ternaryInc;

architecture arch1 of ternaryInc is
  signal C: std_logic_vector(8 downto 0);
begin
  process(A, B, Cin, C) begin
    C(0) <= Cin;
    for i in 0 to 7 loop
      X(i) <= (A(i) and (not C(i))) or (B(i) and C(i));
      Y(i) <= (B(i) and (not C(i))) or
              ((not A(i)) and (not B(i)) and C(i));
      C(i+1) <= A(i) and C(i);
    end loop;
    Cout <= C(8);
  end process;
end arch1;
```
8. (8 points) For the sequential circuit shown below, complete the timing diagram to show when each of the two inputs must be stable. Label the start and end of each stable period relative to the rising clock edge (so write $-3$ to indicate 3 ns before the clock edge, for example). Assume that the flip flop setup time is 1.5 ns, the hold time is 1 ns and that all gates have delays that vary between .3 ns and 1.2 ns.