1. (8 points) The expression $AD + (B' + C)(C' + D')$ has the minterm $A'B'CD$ and the maxterm $(A+B'+C+D)$. List four more of its minterms and four more of its maxterms.
2. (8 points) Use Karnaugh maps to derive a minimal sum of products expression and a minimal product of sums expression for the function. $F(A,B,C,D) = \Sigma m(0,1,3,5,6,10,14,15)$, $d(A,B,C,D) = \Sigma m(2,7,9,13)$. Make full use of the don’t care conditions.
3. (10 points) Draw a logic diagram that directly corresponds to the VHDL module shown below. Every signal that appears in the VHDL should be labeled in your logic diagram.

```vhdl
entity foo is port (  
  A: in std_logic;  
  B: in std_logic_vector(3 downto 0);  
  X: out std_logic_vector(3 downto 0),  
  Y: out std_logic);  
end foo;  
architecture a1 of foo is  
signal Z: std_logic_vector(4 downto 0);  
bEGIN  
  process(A, B, Z) begin  
    Z(4 downto 1) <= Z(3 downto 0) or B(3 downto 0);  
    Z(0) <= A xor B(0);  
    if B(2) < B(3) then  
      Z(4) <= '1';  
    end if;  
  end process;  
  X <= B nand Z(3 downto 0);  
  Y <= Z(4);  
end a1;
```
4. (8 points) The circuit below is a four bit 2s-complement circuit. That is, the output $Y=y_3y_2y_1y_0$ is the 2s-complement of the input $X=x_3x_2x_1x_0$. What is the worst-case propagation delay for a 16 bit version of this circuit, if every simple gate has a delay of 1 ns?

The block diagram below is for a lookahead version of a 2s-complement circuit. Fill in the sub-circuit that goes in the leftmost block.

Consider a 16 bit version of this circuit. Assuming that every gate with more than two inputs is replaced with the fastest equivalent circuit using simple gates, what is the worst-case propagation delay for the 16 bit version?
5. (12 points) Is the sequential circuit shown below, subject to internal hold time violations? If so, show how would you correct them (be specific). Assume that the flip flop setup time is 2 ns, the hold time is .5 ns and the flip flop propagation delay ranges from .75 to 2.5 ns. Also, assume that the gate delay ranges from .6 to 2 ns and that the maximum clock skew is 1 ns.

What is smallest clock period for which this circuit has no internal setup time violations?

During what time period, relative to a rising clock edge must input $A$ be stable in order to ensure that the flip flop does not become metastable?
6. (10 points). Is the state table shown below for a Mealy model sequential circuit, or a Moore model circuit? Construct a state transition diagram corresponding to the state table.

<table>
<thead>
<tr>
<th>$S_1 S_0 AB$</th>
<th>$XY$</th>
<th>$D_1 D_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 00</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>00 01</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>00 10</td>
<td>11</td>
<td>01</td>
</tr>
<tr>
<td>00 11</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>01 00</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>01 01</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>01 10</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>01 11</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>10 00</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>10 01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10 10</td>
<td>11</td>
<td>00</td>
</tr>
<tr>
<td>10 11</td>
<td>01</td>
<td>10</td>
</tr>
</tbody>
</table>
7. (15 points) The state diagram shown below is for a circuit containing a four bit counter, two four bit inputs $A$ and $B$ and a four bit output $X$. The “output” part of the edge labels specifies the current output value of $X$ (so for example, when state=green and count=15, the value of $X$ should be $B$) and changes to count should occur on the next rising clock edge.

The circuit also has a reset signal which has been omitted from the diagram. When reset is high, the circuit should go to the green state and count should be set to 8. Complete the VHDL module outlined on the next page so that it implements the sequential circuit specified by the state diagram. Include code for the reset.

\[ A \geq B, 0 < \text{count} < 15 / X = A + B, \text{ add 1 to count} \]
\[ A < B, 0 < \text{count} < 15 / X = A + B, \text{ subtract 1 from count} \]
entity rainbow is Port ( 
    clk, reset: in std_logic;
    A, B : in std_logic_vector(3 downto 0);
    X : out std_logic_vector(3 downto 0));
end rainbow;

architecture a1 of rainbow is
-- signal and type declarations here

begin
    process (clk) begin

    end process;

    X <=

    end a1;
8. (15 points) The VHDL module defined below implements a parentheses checker, which determines if the parentheses in an incoming string of ASCII characters are balanced. Complete the block diagram/schematic on the following page so that it corresponds to the circuit specified by the VHDL module. Do this by adding wires and simple gates only.

```vhdl
entity paren is
  Port (clk, reset: in std_logic;
        nxtChar : in std_logic_vector(7 downto 0);
        tooDeep, unBal : out std_logic);
end paren;
architecture a1 of paren is
  signal count: std_logic_vector(3 downto 0);
begin
  process (clk) begin
    if clk'event and clk = '1' then
      if reset = '1' then
        count <= "0000";
      else
        if count /= "1111" and nxtChar = x"28" then
          count <= count + '1'; -- x"28" is left paren
        end if;
        if count /= "0000" and nxtChar = x"29" then
          count <= count - '1'; -- x"29" is right paren
        end if;
      end if;
    end if;
  end process;
  tooDeep <= '1' when count = "1111" and nxtChar = x"28" else '0';
  unBal <= '1' when count = "0000" and nxtChar = x"29" else '0';
end a1;
```
9. (12 points) The simulation output shows a program executing on the processor from design problem 5 that includes a stack pointer. Fill in the blanks below with the information that belongs in each of the labeled blank areas in the simulation output. Note that \( \text{30}_{16} = \text{48}_{10} \).

<table>
<thead>
<tr>
<th>( \text{mem_enx} )</th>
<th>( \text{mem_nwx} )</th>
<th>( \text{busx} )</th>
<th>( \text{busx} )</th>
<th>( \text{clk} )</th>
<th>( \text{cpu/state} )</th>
<th>( \text{cpu/tck} )</th>
<th>( \text{regx} )</th>
<th>( \text{spx} )</th>
<th>( \text{axx} )</th>
<th>( \text{acx} )</th>
<th>( \text{alux} )</th>
<th>( \text{mar/mem} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0000</td>
<td>2222</td>
<td>0</td>
<td>halt</td>
<td>t0</td>
<td>0006</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>00000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00014C</td>
<td>0006</td>
<td>0006</td>
<td></td>
<td></td>
<td>019</td>
<td>000A</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0008</td>
<td>0019</td>
<td>00019</td>
<td></td>
<td></td>
<td></td>
<td>007A</td>
<td>0019</td>
<td>00036</td>
<td>0013</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0010</td>
<td></td>
<td>0006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00010</td>
</tr>
</tbody>
</table>

A. ____________ B. __________________ C. __________________

D. ____________ E. __________________ F. __________________

0000 halt = halt execution
0001 negate - \( \text{ACC} := -\text{ACC} \)
0002 read SP, \( \text{ACC} := \text{SP} \).
0003 write SP, \( \text{SP} := \text{ACC} \).
0004 push, \( M[\text{SP}] := \text{ACC}; \text{SP} := \text{SP} - 1 \).
0005 pop, \( \text{ACC} := M[\text{SP+1}]; \text{SP} := \text{SP} + 1 \).
0006 return, \( \text{PC} := M[\text{SP+1}]; \text{SP} := \text{SP} + 1 \).

1xxx immediate load - if sign bit of xxx is 0 then \( \text{ACC} := 0 \text{xxx} \) else \( \text{ACC} := f\text{xxx} \)
2xxx direct load - \( \text{ACC} := M[0\text{xxx}] \)
3xxx indirect load - \( \text{ACC} := M[M[0\text{xxx}]] \)
4xxx direct store - \( M[0\text{xxx}] := \text{ACC} \)
5xxx indirect store - \( M[M[0\text{xxx}]] := \text{ACC} \)
6xxx branch - \( \text{PC} := 0\text{xxx} \)
7xxx branch if zero - if \( \text{ACC} = 0 \) then \( \text{PC} := 0\text{xxx} \)
8xxx branch if positive - if \( \text{ACC} > 0 \) then \( \text{PC} := 0\text{xxx} \)
9xxx branch if negative - if \( \text{ACC} < 0 \) then \( \text{PC} := 0\text{xxx} \)
axxx add - \( \text{ACC} := \text{ACC} + M[0\text{xxx}] \)
bxxx call, \( M[\text{sp}] := \text{PC}; \text{PC} := 0\text{xxx}; \text{SP} := \text{SP} - 1 \).
10. (12 points) Show how to configure the PAL shown below to implement the logic equations.

\[ X = A'D + BD' + AC' \quad Y = A'B' + BC + A'D \quad Z = C'D + BCD' + AC' \]

Do not simplify the equations first.

If you were to implement the same equation using a PLA, what is the smallest number of product terms that would be required for the PLA (again, assuming that the equations are not simplified first)?
11. (12 points) The table shown below represents a 2-way set associative cache (combined instruction and data cache). The ‘<’ and ‘>’ symbols in the center point toward the least recently used of the two entries in a given row. So for example, the right entry in row 4 has been used more recently than the left entry.

<table>
<thead>
<tr>
<th>tag</th>
<th>data</th>
<th>tag</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0a4</td>
<td>401f</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>b23</td>
<td>1001</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0a4</td>
<td>a01d</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0a4</td>
<td>c012</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>20c</td>
<td>1025</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>20c</td>
<td>0001</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>302</td>
<td>a01d</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>c25</td>
<td>6c5d</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>b23</td>
<td>4738</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>acb</td>
<td>c012</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>c45</td>
<td>23fb</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>03f</td>
<td>1003</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>0a3</td>
<td>2eec</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>0c2</td>
<td>25c5</td>
<td>e</td>
</tr>
<tr>
<td>e</td>
<td>20c</td>
<td>1001</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>0a4</td>
<td>a01d</td>
<td></td>
</tr>
</tbody>
</table>

Show how the state of the cache changes, when the processor fetches and executes the two instructions shown below, assuming that a new value replaces the least recently used entry. Assume that cache entries are selected using the low order 4 bits of the memory address and that the tag consists of the high order 12 bits. Be sure to update the ‘<’ and ‘>’ symbols where appropriate.

<table>
<thead>
<tr>
<th>address</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>c45a</td>
<td>23fb</td>
</tr>
<tr>
<td>c45b</td>
<td>ab24</td>
</tr>
</tbody>
</table>

What is the value in the accumulator, after the above two instruction complete?

What is the value of the PC after the next instruction is executed?