You may use one page of notes. No calculators. Write in the spaces provided. Be neat.

1. (10 points) Write the machine language instructions needed to implement each of the lines of pseudo-code shown below. The list of machine language instructions that you are to use is given at the bottom of the page. Assume that the computer uses 16 bit words and represents negative numbers in 2s complement. Include comments with your code to show clearly how it corresponds with the pseudo-code.

```c
int a, b;
    a = 1;
    b = 3;
    while (b != 0) {
        b = b - a;
    }
```

<table>
<thead>
<tr>
<th>Address</th>
<th>Instructions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0000</td>
<td>location for a</td>
</tr>
<tr>
<td>0001</td>
<td>0000</td>
<td>location for b</td>
</tr>
<tr>
<td>0002</td>
<td>8001</td>
<td>a = 1;</td>
</tr>
<tr>
<td>0003</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>8003</td>
<td>b = 3;</td>
</tr>
<tr>
<td>0005</td>
<td>1001</td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>0001</td>
<td>while b != 0 {</td>
</tr>
<tr>
<td>0007</td>
<td>500c</td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>0000</td>
<td>b = b-a</td>
</tr>
<tr>
<td>0009</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>000a</td>
<td>2001</td>
<td></td>
</tr>
<tr>
<td>000b</td>
<td>1001</td>
<td></td>
</tr>
<tr>
<td>000c</td>
<td>4006</td>
<td>}</td>
</tr>
<tr>
<td>000d</td>
<td>3001</td>
<td></td>
</tr>
</tbody>
</table>

0 *** load the accumulator with value stored in memory word ***
1 *** store the value in the accumulator into memory word ***
2 *** add the value in memory word *** to the value in the accumulator
3 000 negate the value in the accumulator
3 001 halt
4 *** change the value of the PC to ***
5 *** if the value in the accumulator is zero, change PC value to ***
6 *** load the accumulator with value whose address is stored in word ***
7 *** store the accumulator value into the word whose address is in word ***
8 *** change the accumulator value to ***
9 *** add *** to the value in the accumulator
2. (5 points) How many simple gates (including inverters) are needed to directly implement the logic expression \( ABC' + A(C+B'D') + (B+C')(A'+D) \)? If the circuit is implemented with CMOS, how many transistors does it use?

There are 4 inverters, 5 OR gates and 5 AND gates, so 14 gates altogether.

The number of transistors is \(8 + 30 + 30\) or 68.

3. (5 points) One of the following timing diagrams is for a J-K master-slave flip flop and the other is for a J-K negative edge-triggered flip flop. Which is which? Why?

The left hand timing diagram is for the Master-Slave flip flop. To see this, note that in the edge-triggered flip flop, the output responds to the inputs only at the falling clock edge, so the edge-triggered flip flop must be high after the first falling clock edge. The M/S flip flop, on the other hand is cleared, since the master latch is cleared when K is high while the clock is high and it stays cleared because the feedback from the slave latch effectively disables the J input.
4. (10 points) A 4 bit *parallel* comparison circuit has inputs \(A = a_3a_2a_1a_0\), \(B = b_3b_2b_1b_0\) and outputs \(A > B\) and \(B > A\). The first output is high if the \(A\) inputs represent a value that is numerically larger than the \(B\) inputs. Similarly, the second output is high if the \(B\) inputs represent a value that is numerically larger than the \(A\) inputs. The figure below shows a 4 bit comparison circuit that is made up of 4 copies of a common building block. Design a combinational circuit that implements this common building block. It should be implemented with AND gates, OR gates and inverters, only. Draw a schematic diagram of your circuit. Assume \(a_3\) and \(b_3\) are the most significant bits and that the input values are unsigned binary integers.
5. (15 points) Draw a schematic diagram for the simplest two level circuit you can find that implements these two equations.

\[
X = BCD' + (AB' + BC)D' + C'(A'B + D)
Y = C'(B' + A'B)D' + A'B' + D + A(B'C'D + CD')
\]

Do this by constructing Karnaugh maps for the two logic equations and finding the most efficient covering. Remember to look for coverings that allow you to share terms in the circuits for the two functions.
6. (10 points) A schematic for a combinational circuit is given below. Write a complete and syntactically correct VHDL entity and architecture that is equivalent, using dataflow VHDL.

![Circuit Diagram]

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity foo is
  port(A, B, C, D: in std_logic;
       X, Y: out std_logic)
end foo;

architecture bar of foo begin
  X <= (not (A or (not D)) and (not D)) and
       ((B and C) or (not D));
  Y <= (A or (not D)) and ((B and C) or (not D));
end bar;
```
(20 points) The block diagram and schematic shown below is for a four bit multiplier. In the schematic, the blocks labelled with an ‘F’ are full adders. If every gate has a propagation delay of 250 ps, what is the worst-case delay through the circuit?

The worst-case delay path is through the second AND gate from the right, the rightmost two full-adders in the middle row and all the full-adders at the bottom. There are 4 gate delays in the full-adders in the second row and 12 in the bottom row for a total of 17 gate delays or 4.25 ns.
The block diagram below is for a 16 bit multiplier using the same design approach. What would the maximum delay be for this circuit? In a processor with a 500 MHz clock, how many clock ticks would it take to complete a multiplication operation using such a multiplier?

The worst-case delay path passes through the rightmost AND block and the rightmost Adder block in each row. The worst-case delay path includes 4 gate delays in the second row, 6 gate delays in the third row, 10 gate delays in the third row and 48 in the bottom row. This gives a total of 69 gate delays or 17.25 ns. This means that with a 500 MHz clock, it will take 9 clock ticks to complete a multiply operation.
8. (5 points) What is the binary representation of 87? What is the hexadecimal representation of 87? What is the binary representation of −87 in 2s complement (assume 8 bit words)? What is the binary representation of −87 in sign-magnitude form?

\[ 87 = 64 + 15 + 4 + 2 + 1, \text{ so the binary representations is 01010111 and the hexadecimal representation is } 57. \]

In 2s complement, −87 is 10101001. In sign-magnitude, −87 is 11010111.

9. (10 points) If a PROM is used to implement the following set of logic equations, how many words must it have and how many bits per word? If a PLA with 4 inputs is used, how many AND gates must it have? If a PAL with two AND gates per output is used, how many inputs and outputs must it have?

\[
\begin{align*}
x_1 &= a_0 \oplus a_1 \\
x_2 &= a_0 \oplus a_1 \oplus a_2 \\
x_3 &= a_0 \oplus a_1 \oplus a_2 \oplus a_3
\end{align*}
\]

The PROM needs 16 words with 3 bits each.

The PLA needs 2 AND gates for \(x_1\), 4 AND gates for \(x_2\) and 8 AND gates for \(x_3\). So 14, altogether.

If the PAL has 2 gates per output, we need to feedback some intermediate values to get \(x_2\) and \(x_3\). If there is no delay constraint, we can do this most easily by feeding \(x_1\) back through and using it to produce \(x_2\), then feeding \(x_2\) back through and using it to produce \(x_3\). This requires 6 inputs and 3 outputs.
10. (10 points) Show how the following logic expression can be implemented using a single CLB (diagram shown below). Label all CLB inputs and outputs you use with the signal names in the expression. Show how the multiplexors are configured by drawing heavy lines on the diagram to show the connections. Specify the contents of each of the LUTs by filling in the tables shown below. Assume that unused inputs are connected to ground.

\[ X = A(CD' + E) + B(C + FG')H \]
11. (10 points) The VHDL shown below implements a sequential circuit. Label the arrows in the state diagram appropriately. What are the logic equations for the next-state variables?

library IEEE;
use IEEE.std_logic_1164.all;

entity foo is
  port (     
a, b, clk: in STD_LOGIC;
  x, y: out STD_LOGIC
  );
end foo;

architecture bar of foo is
signal s1, s0: STD_LOGIC;
begin
  process(clk) begin
    if clk'event and clk = '1' then
      if a = '1' then
        s1 <= '0'; s0 <= '0';
      elsif s1 = '0' and s0 = '0' and b = '0' then
        s1 <= '1'; s0 <= '0';
      elsif s1 = '0' and s0 = '1' and b = '1' then
        s1 <= '1'; s0 <= '1';
      elsif s1 = '1' and s0 = '0' and b = '0' then
        s1 <= '0'; s0 <= '1';
      elsif s1 = '1' and s0 = '1' and b = '0' then
        s1 <= '0'; s0 <= '0';
      end if;
    end if;
  end process;
  x <= '1' when s1 = '0' and s0 = '1' else '0';
  y <= '1' when s1 = '1' and (s0 = '1' or a = b) else '0';
end bar;

\[
s_1 = a'(s_1s_0'b + s_1's_0'b' + s_1's_0b + s_1s_0b)
\]
\[
s_0 = a'(s_1's_0'b' + s_1s_0'b + s_1's_0b + s_1s_0b)
\]
12. (10 points) Suppose we add a small direct-mapped cache to the simple processor that was presented in section 7 of the notes, as shown below (the cache is used for both instructions and data). If the processors starts executing instructions at location 0, what will be the contents of the cache when the processor halts. Write your answer directly on the diagram. For simplicity, assume that the memory addresses are just four bits long. (Note, that the instruction coding can be found in problem 1.)