Solutions - Midterm Exam (Online Section)
(Due Date: October 9th @ 11:00 am)
Clarity is very important! Show your procedure!

**Problem 1 (1.5 pts)**
- Complete the following table. Use the fewest number of bits on each case.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Sign-and-magnitude</th>
<th>1's complement</th>
<th>2's complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>-256</td>
<td>1100000000</td>
<td>1011111111</td>
<td>100000000</td>
</tr>
<tr>
<td>-154</td>
<td>110011010</td>
<td>101100101</td>
<td>101100110</td>
</tr>
<tr>
<td>-107</td>
<td>11101101111</td>
<td>10010100</td>
<td>10010101</td>
</tr>
<tr>
<td>135</td>
<td>0100001111</td>
<td>010000111</td>
<td>010000111</td>
</tr>
</tbody>
</table>

**Problem 2 (1.5 pts)**
- a) Convert the decimal number 136.6875 to i) binary, and ii) hexadecimal.  
  136.6875 = (10001000.1011)₂ = 0x88.B
- b) Represent the decimal number 136 in i) BCD, and ii) Reflective Gray Code

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary (unsigned)</th>
<th>BCD</th>
<th>Reflective Gray Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>136</td>
<td>00001000</td>
<td>0001 0111 0110</td>
<td>11001100</td>
</tr>
</tbody>
</table>

- c) What is the minimum number of bits to represent numbers between 12,000 and 13,024?  
  Numbers between 12000 and 13024: 13024 – 12000 + 1 = 1025  
  Minimum number of bits: \( \log_2 1025 \) = 11 bits

**Problem 3 (1.5 pts)**
Design a circuit (simplify your circuit) that verifies the logical operation of a 3-input NAND gate. \( f = '1' \) (LED ON) if the NAND gate does NOT work properly. Assumption: when the NAND gate is not working, it generates 1's instead of 0's and vice versa.

\[
f = \overline{b} \overline{x} + \overline{a} \overline{x} + \overline{c} \overline{x} + abc \overline{x}
\]

\[
f = x(ab) + x(abc) = x\Theta(abc)
\]
**Problem 4 (20 pts)**

- Implement the following function using: i) a decoder (and gates) and ii) a multiplexer:
  \[
  F_a(X, Y, Z) = \prod(M_3, M_4, M_5, M_6) \quad \text{and} \quad F_b = X \oplus Y \oplus Z
  \]

- Complete the timing diagram of the circuit shown below:
PROBLEM 5 (15 PTS)
- Perform the following operations using the 2’s complement arithmetic. For each case, provide the summands and the result in 2’s complement representation. Use the minimum number of bits to represent the summands and the result so that overflow is avoided.

\[
\begin{align*}
\text{n = 10 bits} & \\
-128 &= 1110000000 + \\
+453 &= 0111000101 \\
+325 &= 0101000101 \\
\text{overflow} &= \bar{c}_9 \bar{c}_8 = 0 \rightarrow \text{no overflow} \\
+325 &\in [-2^8, 2^8-1] \rightarrow \text{no overflow}
\end{align*}
\]

\[
\begin{align*}
\text{n = 9 bits} & \\
-255 &= 100000001 + \\
-37 &= 111011011 \\
011011100 \\
\text{overflow} &= c_9 \bar{c}_8 = 1 \rightarrow \text{overflow!} \\
-255-37 =-292 &\notin [-2^8, 2^8-1] \rightarrow \text{overflow!}
\end{align*}
\]

To avoid overflow:
\[
\begin{align*}
\text{n = 10 bits (sign-extension)}: & \\
-255 &= 1100000001 + \\
-37 &= 1111011011 \\
-292 &= 1011011100 \\
\text{overflow} &= c_9 \bar{c}_8 = 0 \rightarrow \text{no overflow} \\
-292 &\in [-2^9, 2^9-1] \rightarrow \text{no overflow}
\end{align*}
\]

PROBLEM 6 (10 PTS)
- The following VHDL code corresponds to the shaded circuit. Complete the timing diagram:

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

entity tst is
  port (b, c: in std_logic;
        z: out std_logic_vector(1 downto 0));
end tst;

architecture bhv of tst is
  signal x, y: std_logic;
begin
  process (b,c)
  begin
    z <= c & b;
    if c = '0' then
      z <= "10";
    end if;
  end process;
end bhv;
```

architecture bhv of tst is
signal x, y: std_logic;
begin
  process (b,c)
  begin
    z <= c & b;
    if c = '0' then
      z <= "10";
    end if;
  end process;
end bhv;

```

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

library ieee;
use ieee.std_logic_1164.all;

entity tst is
  port (b, c: in std_logic;
        z: out std_logic_vector(1 downto 0));
end tst;

architecture bhv of tst is
  signal x, y: std_logic;
begin
  process (b,c)
  begin
    z <= c & b;
    if c = '0' then
      z <= "10";
    end if;
  end process;
end bhv;
```

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

architecture bhv of tst is
signal x, y: std_logic;
begin
  process (b,c)
  begin
    z <= c & b;
    if c = '0' then
      z <= "10";
    end if;
  end process;
end bhv;
```

<table>
<thead>
<tr>
<th>z</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

library ieee;
use ieee.std_logic_1164.all;

entity tst is
  port (b, c: in std_logic;
        z: out std_logic_vector(1 downto 0));
end tst;

architecture bhv of tst is
  signal x, y: std_logic;
begin
  process (b,c)
  begin
    z <= c & b;
    if c = '0' then
      z <= "10";
    end if;
  end process;
end bhv;
```

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

architecture bhv of tst is
signal x, y: std_logic;
begin
  process (b,c)
  begin
    z <= c & b;
    if c = '0' then
      z <= "10";
    end if;
  end process;
end bhv;
```

<table>
<thead>
<tr>
<th>z</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

```
PROBLEM 7 (10 pts)
- Complete the timing diagram of the circuit shown below:

BONUS PROBLEM (+5 pts)
- Using ONLY 4-to-1 MUXs, implement an 8-to-1 MUX.
- A microprocessor has a 24-bit address line. We connect a memory chip to the microprocessor. The memory chip addresses are assigned the range 0x800000 to 0xBFFFFFF. What is the minimum number of bits required to represent addresses in that individual memory chip?

8-to-1 MUX using ONLY 4-to-1 MUXs:
- If we convert the numbers to binary, we notice that the addresses in the range 0x800000 to 0xBFFFFFF require 24 bits.
- However, we notice that all the addresses in the range 0x800000 to 0xBFFFFFF share the same first two MSBs: 10. Therefore, if we were to address data ONLY in the individual memory chip, we do not need those two bits ⇒ we need 22 bits.
- Another way to put it: We only need 22 bits for the address line of that individual memory device.