Quiz 2

Name:____________________________________  uname:_______________ lab section:_______

This is a closed-book, closed-note, closed-everything (including calculators) quiz. You will have 20 minutes. Be aware that there are different versions of the quiz.

1. Convert the following representations. If there is no representation for a given value, simply draw an “X” in the box. [25, -3 per blank or wrong answer]

<table>
<thead>
<tr>
<th>Number</th>
<th>4-bit unsigned</th>
<th>4-bit signed-magnitude</th>
<th>4-bit 2’s complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>X</td>
<td>1001</td>
<td>1111</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>0110</td>
<td>0110</td>
</tr>
<tr>
<td>-8</td>
<td>X</td>
<td>X</td>
<td>1000</td>
</tr>
</tbody>
</table>

2. What is printed by the following C++ program? [20]

```cpp
#include <iostream>
using namespace std;

class bob
{
public:
  int local;
  bob();
  int cool(int a);
};

bob::bob()
{
  local=-4;
}
int bob::cool(int a)
{
  local=local+a+1;
  return(local*2);
}

main()
{
  bob a, b;
  int i=6;
  b.local=1;
  a.local=b.cool(i);
  cout << a.local << " " << b.local << endl;
}
```

3. 147

**Comments**: default constructor sets a.local and b.local to be -4. b.local is then made 1. In b.cool local (which is the same a b.local) is set to 1*6+1 or 7. The function returns 14. a.local is then set to 14.
Matlab: [12]
Show the matrix “x” which results from the following instructions:
1. \( x=7:2:12 \)
\[
\begin{bmatrix}
7 & 9 & 11
\end{bmatrix}
\]

2. \( y=[2 \ 5 \ 7 \ 1] \)
\( x=y([2 \ 3]) \)
\[
\begin{bmatrix}
5 & 7
\end{bmatrix}
\]

4. Write a Matlab program which finds the sum of all values of a 2 by 2 matrix named bob. [13]

\[
\begin{align*}
\text{sum}(\text{sum}(\text{bob})) \\
\text{or} \\
\text{bob}(1,1)+\text{bob}(1,2)+\text{bob}(2,1)+\text{bob}(2,2) \\
\text{or} \\
\text{lots of other ways, but the above two are probably the best.}
\end{align*}
\]

Recall the following:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>00</td>
<td>memA</td>
<td>memB</td>
</tr>
<tr>
<td>addi</td>
<td>01</td>
<td>memA</td>
<td>immediate</td>
</tr>
<tr>
<td>beq</td>
<td>10</td>
<td>memA</td>
<td>target</td>
</tr>
<tr>
<td>print</td>
<td>11</td>
<td>memA</td>
<td>unused</td>
</tr>
</tbody>
</table>

**add**: \( \text{Mem}[\text{memA}] = \text{Mem}[\text{memA}] + \text{Mem}[\text{memB}] \)
**addi**: \( \text{Mem}[\text{memA}] = \text{Mem}[\text{memA}] + \text{immediate} \)
**beq**: if(\( \text{Mem}[\text{memA}] == 0 \)) \( \text{PC} = \text{target} \)
**print**: print \( \text{Mem}[\text{memA}] \) and halt.

So \( 0000 \ 0000 \ 0000 \ 0000 \) says to add the byte at memory location 0 to itself and store the result in memory location zero.

5. Translate the following assembly program into machine code. Leave blank any unused space. [30]

```
add a, b  
add a, 1  
print a  
a: 5      
b: -4
```

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00 001110</td>
</tr>
<tr>
<td>1</td>
<td>00000111</td>
</tr>
<tr>
<td>2</td>
<td>01 001110</td>
</tr>
<tr>
<td>3</td>
<td>00000001</td>
</tr>
<tr>
<td>4</td>
<td>11 000110</td>
</tr>
<tr>
<td>5</td>
<td>anything or nothing could go here</td>
</tr>
<tr>
<td>6</td>
<td>00000101</td>
</tr>
<tr>
<td>7</td>
<td>11111100</td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

**Comments**
add in bytes 0 and 1.
add in bytes 2 and 3
print in 4 and 5
a in 6
b in 7

* negative numbers in 2’s complement
* execution starts at location 0
*