NOTE: the exam will be on Chapters 1-4 and part of Chapter 5 (zyBooks 5.1 and 5.6). In addition to problems like those below, the exam will have true/false, fill in the blank, and multiple choice questions. You would be wise to study all the terminology we have learned in the class thus far; a set of definitions are available on Quizlet. (Link to be posted soon on Piazza.) For each FULL page completed (and brought to class on Oct 2nd), 1 point extra credit will be earned (up to 4 points possible).

1. Assume your computer uses 8-bit 2’s complement signed numbers. Find the sum of the two binary numbers below; give the result in binary and in (signed) decimal, and discuss your answer

   \[
   \begin{align*}
   &0 0 1 0 1 0 1 1 \\
   + &0 1 0 0 1 0 0 1 \\
   \end{align*}
   \]

   \[
   1 0 1 0 0 1 1 0 \\
   + 1 1 1 1 0 0 1 0
   \]

2. If our machine stores a number in 4 bits. What is the smallest and largest number we can represent
   a. in unsigned binary
   
   b. in sign/magnitude
   
   c. in 2’s compliment

3. Given the following 8-bit binary number: 1000 1001. What is its decimal representation in each of the following notations?
   a. Unsigned binary notation
   
   b. Sign/magnitude notation
   
   c. Two’s complement

4. What color is the following RGB pixel?   00000000 11111111 11111111

5. Suppose Jo’s digital camera has a storage capacity of 256MB. How many photographs could be stored in her camera if each picture is 1024 by 1024 pixels, and if each pixel requires three bytes of storage (RGB)?
6. Create a truth table for the optimized circuit shown below.

![Optimized Circuit Diagram]

7. Give a Boolean expression in the format of your choice (using only AND, OR, and NOT) that produces the truth table from problem #6.

8. Complete the table below.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0000 0010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>0001 1010</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0001 1010</td>
<td>52</td>
<td>25</td>
</tr>
</tbody>
</table>

9. Write pseudocode that computes the area and circumference of a circle given the radius $r$, if and only if the radius is greater than or equal to 1.0; if the radius is less than 1.0, the code will compute the circumference only.

10. Suppose the following sorted numbers exist in some list. Suppose you are looking for the number 1234 via the binary search algorithm. What numbers would 1234 be compared with to determine whether it is in the list or not.

    24, 1234, 3584, 4968, 7965, 9493, 10325, 15694, 20586, 54687, 78576, 496854, 558674
11. Use the algorithm for the shuffle left algorithm on the following data. What is the data list when the algorithm ends? What is the value of right when the algorithm ends?

1. Get values for n and the n data items
2. Set the value of legit to n
3. Set the value of left to 1
4. Set the value of right to 2
5. While left is less than or equal to legit do Steps 6 through 14
6. If the item at position left is not 0 then do Steps 7 and 8
7. Increase left by 1
8. Increase right by 1
9. Else (the item at position left is 0) do Steps 10 through 14
10. Reduce legit by 1
11. While right is less than or equal to n do Steps 12 and 13
12. Copy the item at position right into position (right – 1)
13. Increase right by 1
14. Set the value of right to (left + 1)
15. Stop

<table>
<thead>
<tr>
<th>0</th>
<th>14</th>
<th>27</th>
<th>0</th>
<th>41</th>
<th>0</th>
<th>9</th>
<th>12</th>
<th>52</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12. Write the output from the following pseudocode:

```plaintext
count = 0
while count is NOT 5
    count = count + 2
    print count
```

13. Suppose a program has a user input a value for n. The program then executes the following pseudocode. What is the time complexity of this algorithm?

```plaintext
1:   read in a list of n values from a file
2:   set counter to 0
3:   for i from 0 to n-1
4:       print value[i]
5:   if value[i] is greater than 100
6:       add one to counter
7:   for j from 0 to n-1
8:       divide value[j] by 2
9:       print value[j]
10:  print counter
```
14. If the average access time for RAM is 40 nsec and the average access time for cache memory is 5 nsec, what is the overall average access time if our cache hit rate is 75%?

Would it be better for a system designer to decrease RAM access time to 30 nsec, or increase the hit rate to 85%? Justify your response.

15. Assuming the standard memory cell size of 1 byte, how big can our memory be if our computer uses 16 bit addresses?

32 bit addresses?

16. Suppose Simon designed a machine language with an opcode field of 4 bits. How many different instruction types can Simon’s language contain?

Caleb designed a system with 6 bits opcode. How many instructions does his language support?

17. List and describe the parts of a machine language instruction?

18. Using the op codes described in the table below. Write a program in assembly to compute the computation \( z = \sqrt{\frac{x+y}{x-y}} \). The address of \( x, y, \) and \( z \) are M[5], M[6], and M[7]. Also, assume M[4] is available for any intermediate answer. Then convert your assembly program into machine language. Use 2 bit op codes and 3 bit addresses.

<table>
<thead>
<tr>
<th>Code</th>
<th>Assembly</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>add</td>
<td>Addition</td>
</tr>
<tr>
<td>1</td>
<td>sub</td>
<td>Subtraction</td>
</tr>
<tr>
<td>2</td>
<td>div</td>
<td>Division</td>
</tr>
<tr>
<td>3</td>
<td>sqrt</td>
<td>Square root</td>
</tr>
</tbody>
</table>

19. Here is a message in ASCII. What does it say?

```
01000111 01101111 00100000 01110011 01110100 01110101 01100100 01111001 00100001
```