Homework 4 Solutions

1. "There are 10 types of people in the world. Those who understand binary and those who don't." What happened to the other 8 types of people?
   (1)

The joke lies in the fact that 10 is decimal 2 converted to binary. Therefore, the two types of people are those that understand binary (10) and those who do not.

2. Convert the following numbers to binary (unsigned) and hexadecimal. **(2)**
   a. 10
   b. 147

Decimal	**Unsigned Binary**	**Hexadecimal**
10	**1010**	**0xA**
147	**10010011**	**0x93**

3. Convert the following binary (unsigned) numbers to hexadecimal and decimal. **(2)**
   a. 101
   b. 1111011011

Unsigned Binary	**Hexadecimal**	**Decimal**
101	**0x5**	**5**
1111011011	**0x3DB**	**987**

4. What is the largest integer, smallest (most negative) integer, and the number of integers you can represent with a nibble (4 bits), when using unsigned integer, signed magnitude, one's complement, and two's complement representation? **(2)**

Notation	**Largest Integer**	**Smallest Integer**	**Number of Integers**
Unsigned Binary	**15**	**0**	**16**
Signed Magnitude	**7**	**-7**	**15**
One's Complement	**Removed**	**Removed**	**Removed**
Two's Complement	**7**	**-8**	**16**

5. Removed

6. What is the largest integer (Decimal value) that can be represented by a bit, a nibble, a byte, and a word? Assume all binary values are unsigned. Show how you came to your answer (Hint: take a look at the method for converting numbers in Chapter 4.3, using the question "How Many?"). **(2)**

Binary Width	**Largest Integer Value**
Bit	**1**
Nibble	**15**
Byte	**255**
Word	**Removed**

7. Convert 101.101 from unsigned fixed point representation to decimal. **(1)**

   5.625

   b) Convert 72.375 from decimal to unsigned binary **(1)**

   1001000.011

8. Assuming two's complement notation, perform the following : **(4)**
   a. 11000011 + 00100011
   b. 01000011 + 00100011
   c. 11000011 – 00100011
   d. 01111111 + 01111111

Assume your answer must also use only eight bits and are in two's complement notation. Do all the above operations still work correctly? If not, which ones have problems, and why?

A) 11100110 (-26)
B) 01100110 (102)
C) Convert 11000011 + 11011101 = 10100000 (-96). It has a remainder of 1, but is still -96, as we take the last remaining 8 bits. If this ended up needing a higher number, this could be dangerous!
D) 111111110 Overflow! Results in a negative number (127 + 127)

9. What would the following result in? (When x appears, assume it is an 8-bit binary number) **(4)**

a. 10010001 & 10011110
b. 10010001 | 10011110
c. ~10010001
d. x ^ x

A. 10010000
B. 10011111
C. 01101110
D. Always is 00000000, as ^ is XOR