

Questions

Question 1

Counting practice: count from zero to thirty-one in binary, octal, and hexadecimal:

	Binary	Octal	Hex		Binary	Octal	Hex
Zero				Sixteen			
One				Seventeen			
Two				Eighteen			
Three				Nineteen			
Four				Twenty			
Five				Twenty one			
Six				Twenty two			
Seven				Twenty three			
Eight				Twenty four			
Nine				Twenty five			
Ten				Twenty six			
Eleven				Twenty seven			
Twelve				Twenty eight			
Thirteen				Twenty nine			
Fourteen				Thirty			
Fifteen				Thirty one			

file 01221

Question 2

Add the following binary numbers:

$$\begin{array}{r} 10010 \\ + 1100 \\ \hline \end{array}$$

$$\begin{array}{r} 1011101 \\ + 1000000 \\ \hline \end{array}$$

$$\begin{array}{r} 10011 \\ + 1111101 \\ \hline \end{array}$$

$$\begin{array}{r} 10011001 \\ + 100111 \\ \hline \end{array}$$

$$\begin{array}{r} 11000011 \\ + 101111 \\ \hline \end{array}$$

$$\begin{array}{r} 1001100 \\ + 1100101 \\ \hline \end{array}$$

file 01220

Question 3

If the numbers sixteen and nine are added in binary form, will the answer be any different than if the same quantities are added in decimal form? Explain.

file 01229

 Question 4

What is the *one's complement* of a binary number? If you had to describe this principle to someone who just learned what binary numbers are, what would you say?

Determine the one's complement for the following binary numbers:

- 10001010_2
- 11010111_2
- 11110011_2
- 11111111_2
- 11111_2
- 00000000_2
- 00000_2

[file 01222](#)

Question 5

Determine the *two's complement* of the binary number 01100101_2 . Explain how you did the conversion, step by step.

Next, determine the two's complement representation of the quantity *five* for a digital system where all numbers are represented by four bits, and also for a digital system where all numbers are represented by eight bits (one *byte*). Identify the difference that "word length" (the number of bits allocated to represent quantities in a particular digital system) makes in determining the two's complement of any number.

[file 01224](#)

Question 6

In a computer system that represents all integer quantities using two's complement form, the most significant bit has a negative place-weight. For an eight-bit system, the place weights are as follows:

$$\overline{-2^7} \quad \overline{2^6} \quad \overline{2^5} \quad \overline{2^4} \quad \overline{2^3} \quad \overline{2^2} \quad \overline{2^1} \quad \overline{2^0}$$

Given this place-weighting, convert the following eight-bit two's complement binary numbers into decimal form:

- $01000101_2 =$
- $01110000_2 =$
- $11000001_2 =$
- $10010111_2 =$
- $01010101_2 =$
- $10101010_2 =$
- $01100101_2 =$

[file 01225](#)

Question 7

In an eight-bit digital system, where all numbers are represented in two's complement form, what is the largest (most positive) quantity that may be represented with those eight bits? What is the smallest (most negative) quantity that may be represented? Express your answers in both binary (two's complement) and decimal form.

[file 01226](#)

Question 8

Two's complement notation really shows its value in binary addition, where positive and negative quantities may be handled with equal ease. Add the following byte-long (8 bit) two's complement numbers together, and then convert all binary quantities into decimal form to verify the accuracy of the addition:

$$\begin{array}{r} 00110101 \\ + 00001100 \\ \hline \end{array}$$

$$\begin{array}{r} 01110110 \\ + 00000010 \\ \hline \end{array}$$

$$\begin{array}{r} 00111101 \\ + 11111011 \\ \hline \end{array}$$

$$\begin{array}{r} 00001010 \\ + 10010101 \\ \hline \end{array}$$

$$\begin{array}{r} 11111110 \\ + 11011101 \\ \hline \end{array}$$

$$\begin{array}{r} 11111110 \\ + 11111101 \\ \hline \end{array}$$

file 01227

Question 9

Add the following eight-bit two's complement numbers together, and then convert all binary quantities into decimal form to verify the accuracy of the addition:

$$\begin{array}{r} 10110111 \\ + 01110110 \\ \hline \end{array}$$

$$\begin{array}{r} 00111101 \\ + 00111011 \\ \hline \end{array}$$

$$\begin{array}{r} 11111011 \\ + 11111011 \\ \hline \end{array}$$

$$\begin{array}{r} 10000001 \\ + 10010001 \\ \hline \end{array}$$

$$\begin{array}{r} 01111011 \\ + 00111101 \\ \hline \end{array}$$

$$\begin{array}{r} 01111111 \\ + 10000001 \\ \hline \end{array}$$

file 01230

Question 10

How is it possible to tell that *overflow* has occurred in the addition of binary numbers, without converting the binary sums to decimal form and having a human being verify the answers?

file 01231

Question 11

What is a *floating-point* number in a digital system?

file 01232