Binary math

This worksheet and all related files are licensed under the Creative Commons Attribution License, version 1.0. To view a copy of this license, visit http://creativecommons.org/licenses/by/1.0/, or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA. The terms and conditions of this license allow for free copying, distribution, and/or modification of all licensed works by the general public.

Resources and methods for learning about these subjects (list a few here, in preparation for your research):

Question 1

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

	Binary	Octal	Hex
Zero			
One			
Two			
Three			
Four			
Five			
Six			
Seven			
Eight			
Nine			
Ten			
Eleven			
Twelve			
Thirteen			
Fourteen			
Fifteen			

	Binary	Octal	Hex
Sixteen			
Seventeen			
Eighteen			
Nineteen			
Twenty			
Twenty one			
Twenty two			
Twenty three			
Twenty four			
Twenty five			
Twenty six			
Twenty seven			
Twenty eight			
Twenty nine			
Thirty			
Thirty one			

file 01221

Answer 1

No answers given here – compare with your classmates!

Notes 1

In order to familiarize students with these "strange" numeration systems, I like to begin each day of digital circuit instruction with counting practice. Students need to be *fluent* in these numeration systems by the time they are finished studying digital circuits!

One suggestion I give to students to help them see patterns in the count sequences is "pad" the numbers with leading zeroes so that all numbers have the same number of characters. For example, instead of writing "10" for the binary number two, write "00010". This way, the patterns of character cycling (especially binary, where each successively higher-valued bit has half the frequency of the one before it) become more evident to see.

Question 2

Add the following binary numbers:

```
  10010          1011101          10011
+ 1100         + 1000000      + 1111101

  10011001       11000011       1001100
+   100111     +   101111     + 1100101
```

file 01220

Answer 2

```
  10010          1011101          10011
+ 1100         + 1000000      + 1111101
  11110          10011101       10010000

  10011001       11000011       1001100
+   100111     +   101111     + 1100101
  11000000       11110010       10110001
```

Notes 2

Ask your students to describe what differences exist between manually adding binary numbers and manually adding decimal numbers, if any.

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

Answer 3

No. The form of numeration used to represent numbers has no bearing on the outcome of mathematical operations.

Notes 3

Although this may seem like a trivial question, I've met electronics technicians who actually believed that the form of numeration affected the outcome of certain mathematical operations. In particular, I met one fellow who believed the number π was fundamentally different in binary form than it was in decimal form: that a binary "pi" was not the same quantity as a decimal "pi". I challenged his belief by applying some Socratic irony:

Me: How do you use a hand calculator to determine the circumference of a circle, given its diameter? For example, a circle with a diameter of 5 feet has a circumference of . . .

Him: By multiplying the diameter times "pi". 5 feet times "pi" is a little over 15 feet.

Me: Does a calculator give you the correct answer?

Him: Of course it does.

Me: Does an electronic calculator use decimal numbers, internally, to do math?

Him: No, it uses binary numbers, because its circuitry is made up of logic gates . . . *(long pause)* . . . Oh, now I see! If the type of number system mattered in doing math, digital computers and calculators would arrive at different answers for arithmetic problems than we would doing the math by hand!

Of course, those familiar with computer programming and numerical analysis understand that digital computers can introduce "artifacts" into computed results that are not mathematically correct. However, this is not due to their use of binary numeration so much as it is limited word-widths (leading to overflow conditions), algorithmic problems converting floating-point to integer and visa-versa, and such.

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

Answer 4

- 10001010_2: One's complement = 01110101_2
- 11010111_2: One's complement = 00101000_2
- 11110011_2: One's complement = 00001100_2
- 11111111_2: One's complement = 00000000_2
- 11111_2: One's complement = 00000_2
- 00000000_2: One's complement = 11111111_2
- 00000_2: One's complement = 11111_2

Follow-up question: is the one's complement 11111111_2 identical to the one's complement of 11111_2? How about the one's complements of 00000000_2 and 00000_2? Explain.

Notes 4

The principle of a "one's complement" is very, very simple. Don't give your students any hints at all concerning the technique for finding a one's complement. Rather, let them research it and present it to you on their own!

Be sure to discuss the follow-up question, concerning the one's complement of different-width binary numbers. There is a very important lesson to be learned here!

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

Answer 5

The two's complement of 01100101 is 10011011.

The two's complement of five is 1011 in the four-bit system. It is 11111011 in the eight-bit system.

Notes 5

The point about word-length is extremely important. One cannot arrive at a definite two's complement for any number unless the word length is first known!

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

Answer 6

- $01000101_2 = 69_{10}$
- $01110000_2 = 112_{10}$
- $11000001_2 = -63_{10}$
- $10010111_2 = -105_{10}$
- $01010101_2 = 85_{10}$
- $10101010_2 = -86_{10}$
- $01100101_2 = 101_{10}$

Notes 6

Students accustomed to checking their conversions with calculators may find difficulty with these examples, given the negative place weight! Two's complement notation may seem unusual at first, but it possesses decided advantages in binary arithmetic.

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

Answer 7

Largest (most positive): $01111111_2 = 127_{10}$

Smallest (most negative): $10000000_2 = -128_{10}$

Notes 7

The most important concept in this question is that of *range*: what are the limits of the representable quantities, given a certain number of bits. Two's complement just makes the concept a bit more interesting.

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:

```
  00110101        01110110        00111101
+ 00001100      + 00000010      + 11111011
----------      ----------      ----------

  00001010        11111110        11111110
+ 10010101      + 11011101      + 11111101
----------      ----------      ----------
```

file 01227

Answer 8

```
  00110101  53        01110110  118       00111101  61
+ 00001100  12      + 00000010   2      + 11111011  -5
----------          ----------          ----------
  01000001  65        01111000  120       00111000  56

  00001010  10        11111110  -2        11111110  -2
+ 10010101 -107     + 11011101  -35     + 11111101  -3
----------          ----------          ----------
  10011111 -97        11011011  -37       11111011  -5
```

Notes 8

Have your students do some of these problems on the board, in front of class for all to see. Ask students what happens to the left-most "carry" bit, if it exists in any of these problems. Ask them *why* we do what we do with that bit, when we would usually place it in our answer.

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:

```
  10110111        00111101        11111011
+ 01110110      + 00111011      + 11111011

  10000001        01111011        01111111
+ 10010001      + 00111101      + 10000001
```

file 01230

Answer 9

```
  10110111  -73        00111101  61         11111011  -5
+ 01110110  118      + 00111011  59       + 11111011  -5
  00101101  45         01111000  120        11110110  -10

  10000001  -127       01111011  123        01111111  127
+ 10010001  -111      + 00111101   61      + 10000001 -127
  00010010   18         10111000  -72        00000000   0
```

Follow-up question: Why are some of these answers incorrect? Hint: perform the additions in *decimal* form rather than binary form, and then explain why those answers are not represented in the binary answers.

Notes 9

This question introduces students to the phenomenon of *overflow*. This is a very important principle to understand, because real computer systems must deal with this condition properly, so as not to output incorrect answers!

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

Answer 10

Check the sign bit of the answer, and compare it to the sign bits of the *addend* and *augend*.

Challenge question: under what condition(s) is overflow impossible? When can we add two binary numbers together and know with certainty that the answer will be correct?

Notes 10

Later, this concept of overflow checking should be applied to a real circuit, with students designing logic gate arrays to detect the presence of overflow. First, though, they must learn to recognize its presence analytically.

Question 11

What is a *floating-point* number in a digital system?
file 01232

Answer 11

"Floating-point" numbers are the binary equivalent of scientific notation: certain bits are used to represent the mantissa, another collection of bits represents the exponent, and (usually) there is a single bit representing sign. Unfortunately, there are several different "standards" for representing floating-point numbers.

Notes 11

Ask your students why computer systems would have need for floating point numbers. What's wrong with the standard forms of binary numbers that we've explored thus far?