Homework #2 - Binary Numbers

Name:

1 Intro

If we want to write good computer programs, we need to understand a little bit about the inner workings of computers. All modern computers use electronic devices to store and manipulate data and programs. To help make them simpler, these devices are built to work with only two voltages, an "ON" and an "OFF" voltage (usually 0V and 3.3V or 0V and 5V). Because of this, all data in the computer is represented using the numbers "0" and "1" (the 0 is the "OFF" state and the 1 is the "ON" state). Everything, and I mean **everything** that the computer can do is done using only these two numbers. Videos, text, pictures, and computer programs themselves are all stored inside the computer as sequences of ones and zeros. We have to understand how these numbers work (how to do math with them, compare them etc...) in order to write good computer programs. I have introduced you to binary, decimal, and hexadecimal numbers in class before. This homework tests your ability to understand and convert between these three types of numbers.

2 Binary Conversion

Convert the following numbers into binary (**hint**: use the repeated division method we talked about in class):

1. $45_{10} =$
2. $255_{10} =$
3. $0_{10} =$
4. $36_{10} =$

What would happen if you tried to convert 256 into an 8-bit binary number? Can you do this or do you need more than 8 bits?

3 Binary to Decimal Conversion

Convert the following binary numbers into decimal:

5. $101_2 =$
6. $00001111_2 =$
7. $10000000_2 =$
8. $10101010_2 =$

4 Addition for positive numbers

Add these binary numbers together:

9. $1010_2 + 0010_2 =$
10. $0010_2 + 1110_2 =$
11. $11110010_2 + 00101110_2 =$

Think about what happens if you try to add two 8-bit numbers together and the result uses more than 8-bits. What will the computer do?

5 Forming the two's complement

In the computer, positive and negative numbers are **not** stored using the "sign magnitude" form we are used to, where a number has a + or - in front to show its sign. Instead, negative numbers in binary are represented by forming the "two's complement" of a positive number with the correct magnitude (size). For instance, in binary the number "5" is represented by "101" or if our computer uses 8-bit memory, "00000101". That is the same as "+5". How do we make "-5"? We have to change all of the "1"s to "0"s and then add an additional "1" to the number. Like this:

$$00000101_2 \rightarrow 11111010_2 + 1_2 \rightarrow 11110101_2$$

So "-5" is actually represented by "11110101". You try! Convert the following **negative** decimal numbers to their negative binary equivalents:

12. $-45_{10} =$
13. $-255_{10} =$
14. $-0_{10} =$
15. $-36_{10} =$

Hint: Since these are the same decimal numbers you converted to binary at the beginning of the homework, you can use the binary conversions you already did to form the two's complement. What happens when you form the two's complement of 0? Does this seem like the right behavior?

6 Representing fractions

In decimal or base 10 numbers, a fraction like 0.255 is represented as:

$$2 \cdot 10^{-1} + 5 \cdot 10^{-2} + 5 \cdot 10^{-3}$$

You can see that it's the same system we use for representing integers (225, 34) but with negative exponents. We can do this with binary numbers as well. For instance 0000.1000 in binary would convert to the following decimal number:

$$1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-4} =$$

$$0.5 + 0 + 0 + 0 = 0.5$$

So $0.1_2 = 0.5_{10}$. Try it yourself for the following numbers:

16. $0000.1010_2 =$
17. $0000.0010_2 =$
18. $1010.0101_2 =$
19. $1100.0011_2 =$

Now try to go the other way!

20. $0.5_{10} =$
21. $0.1_{10} =$
22. $2.5_{10} =$
23. $45.25_{10} =$

Notice that some numbers do not have an exact representation when you convert them from one base to another. 0.1 in decimal has no exact representation in base 2 because it repeats forever!

7 Understanding Hexadecimal

Converting between bases is easier when they share the relationship $R_1 = R_2^K$. This is the case for base 2 and base 16 where $16 = 2^4$, and because of this base 16 is a popular way to represent numbers when talking about computer systems. Hexadecimal numbers are shorter than their binary equivalents which makes them easier to read and write, and there's a lower chance of making mistakes when writing them. In hexadecimal, there are actually 16 different symbols instead of the 10 (0 to 9) we use in decimal. In hexadecimal, the first 10 digits are represented by the numbers 0 to 9 just like in decimal, but the last 6 digits are represented by the letters A to F, where A = 10, B = 11, ...up to F = 15

Because the bases are closely related, you can easily convert to and from hexadecimal and binary almost just by looking at a number. Each hexadecimal digit represents 4 binary digits, so the number A in hex would be 1010 in binary, or 10 in decimal. The number F is decimal 15 or 1111, so FF would be 11111111, and so on... Try it yourself for the numbers below:

24. $FE_{16} =$
25. $FF_{16} =$
26. $DEAD_{16} =$
27. $ABCD_{16} =$

Now try to go the other way, converting into hexadecimal (note if a binary number does NOT break evenly into groups of 4 digits, just add zeros on the left side of the number until it does!):

28. $1010_2 =$
29. $0010_2 =$
30. $001_2 =$
31. $10101111_2 =$