Name(s)_____ Period _____ Date _____

Activity Guide - Encoding Hexadecimal Numbers

Hexadecimal: a number system comprised of the familiar ten arabic numerals (0, 1, 2 … 9) as well as the first six English letters (A, B, C, D, E, F). Also referred to as a "base 16" number system, hexadecimal is used in the world of computing to help humans read and talk about large binary numbers. Since there are no symbols reserved for the numbers 10 through 15 in our familiar base 10 number system, the characters A through F are used to represent them.

Here are the first 16 numbers, in both binary and hexadecimal:

Number	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

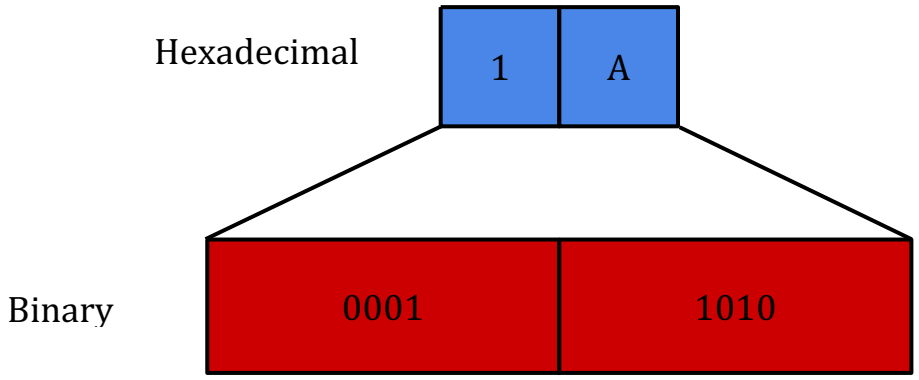
Converting from hexadecimal to binary is quite easy. One hexadecimal digit can represent any of 16 values. To do the same in binary you would need 4 bits, or 4 binary digits. Therefore, every hexadecimal digit can be replaced with its four bit equivalent in binary.

Converting Hexadecimal to Binary

1. Split the hexadecimal number up into two digits.
2. Convert each digit to decimal.

Example: Converting 1A in hexadecimal to binary

The first digit contains the number 1. The 4-bit binary representation of 1 is 0001.
The second digit contains “A” or 10. The 4-bit binary representation of 10 is 1010.
Therefore 1A in hexadecimal can be converted to 00011010 in binary.

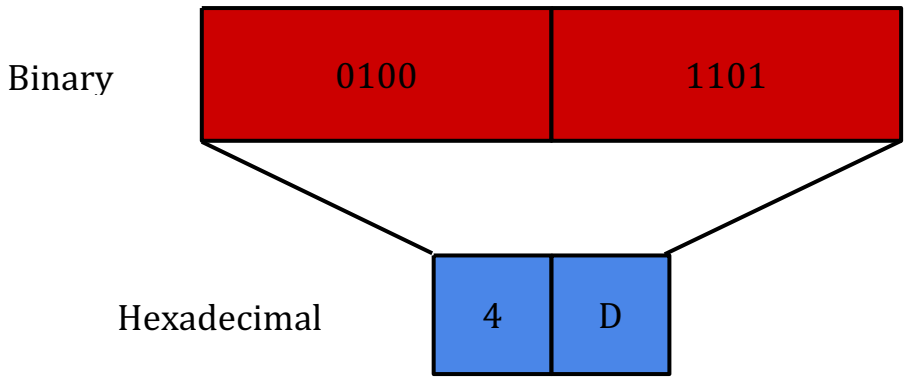


Convert: Convert the following hexadecimal numbers to their binary equivalents:

Hexadecimal	Binary
3	
A	
1C	
89	
DD0	
ABC7	

Example: Converting 01001101 in binary to hexadecimal

The first four bits are 0100 or “4,” which is just the character “4” in hexadecimal.
The second four bits are 1101 or “13,” which is just the character “D” in hexadecimal.
Therefore 01001101 in binary can be converted to 4D in hexadecimal.



Binary	Hexadecimal
0111	
1110	
01010001	
11000111	
000101011111	
101001011110	

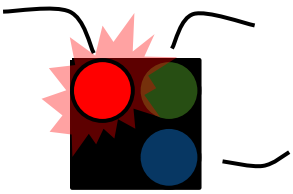
Hexadecimal numbers are used because this conversion allows you to quickly communicate information about large chunks of binary digits. Perhaps the most common place one will see this is in naming colors to be displayed on a screen.

Representing Colors

While monitors typically can represent thousands, if not millions of colors, each individual pixel contains only three tiny lights, one each of red, green, and blue. Every color you see on a screen is created by setting different brightnesses for each of these little lights. Amazing!

Typically each of these little lights (red, green, blue) has its brightness dictated by a single byte. The largest number that can be written using one byte is 255, which indicates that a light should be turned to full brightness. 0 indicates that a light should be turned off.

> ***Example:*** *If we wanted to make the reddest color our pixel could make, we would turn the red light all the way on, and the other two all the way off. This would be represented as*
>
> *red: 11111111 or 255*
> *green: 00000000 or 0*
> *blue: 00000000 or 0*
>
> *But since these will usually be read as one 24-bit chunk, we display them below in the order they would be received, first the red byte, then the green byte, then the blue byte*

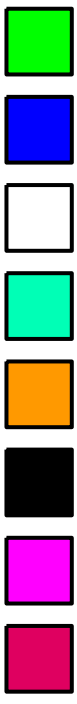


Reddest Red*: 111111110000000000000000*

While this system makes a lot of sense for a computer to read, it can be tricky for a human to keep track of all those digits. This is where hexadecimal becomes really handy. The 24-bit sequence representing each pixel is 6 nibbles long, and so it can be represented as 6 hexadecimal digits.

> ***Example:*** *Our reddest red can be written in hexadecimal as "#FF0000". To do the conversion yourself, just read the bits above in 4-bit nibbles and convert from binary to hexadecimal using the table above.*

Match the color to its hexadecimal equivalent:



#df006

#fffff

#00ffb

#ff00f

#00000

#0000f

#ff990

#00ff0