Decimal - base 10 place values

and so on...	10000	1000	100	10	1

so… 156_{10} = (1 x 100) + (5 x 10) + (6 x 1)

Binary - base 2 place values
*** 8 bits - 1 byte of data**

and so on...	128	64	32	16	8	4	2	1

Hexadecimal - base 16 place values

and so on...	65536	4096	256	16	1

0x - a prefix you might see sometimes to indicate the number is in hexadecimal rather than in some other base. The C programming language, and others, use it to tell the compiler "this is hex, not some meaningless letters and numbers."

One way (and probably the easiest way) to convert a number from decimal to hex is to convert it to binary first

Subscripts: 2, 8, 10, 16 - The base subscripts are used to indicate the type of number systems (binary, octal, decimal, hexadecimal)

BASE 10	BASE 16
0	**0**
1	**1**
2	**2**
3	**3**
4	**4**
5	**5**
6	**6**
7	**7**
8	**8**
9	**9**
10	**A**
11	**B**
12	**C**
13	**D**
14	**E**
15	**F**

It always helps to draw out a chart for the number system when working with conversions

Binary (base 2)
0, 1

128	64	32	16	8	4	2	1	(place values)
0	0	0	0	0	0	0	0	

Decimal (base 10)
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

10000000	10000000	1000000	10000	1000	100	10	1	(place values)
0	0	0	0	0	0	0	0	

Hexadecimal (base 16)
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

65536	4096	256	16	1	(place values)
0	0	0	0	0	

Decimal to Binary:

work left to right and deduct the decimal amounts as you go - turn the values “on” for the correct places

so…	**156_{10} =**	**128 + 16 + 8 + 4 =**	**$1001\ 1100_2$**
	14_{10} =	**8 + 4 + 2 =**	**$0000\ 1110_2$**
	28_{10}	**16 + 8 + 4 =**	**$0001\ 1100_2$**

Binary to Hex:

Let's take those same binary numbers and turn them into hex
- **one very neat thing about hex is that you can take each set of 4 bits and convert to the hex value**
- remember that 4 bits can only hold up to a 15_{10}

so… **156_{10} = 1001 1100$_2$ = 9 + C = 9C$_{16}$**

14_{10} = 0000 1110$_2$ = 0 + E = 0E$_{16}$

28_{10} = 0001 1100$_2$ 1 + C = 1C$_{16}$

Hex to Binary:

Each Hex digit is equal to 4 bits - use the values for the lowest set of bits (8, 4, 2, 1)

so…

6D$_{16}$ = 0110 1101$_2$ = 109$_{10}$

let's test this out:

hex: (6 * 16) + (13 * 1) = 96 + 13 = **109_{10}**
binary: 64 + 32 + 8 + 4 + 1 = **109_{10}**

C5$_{16}$ = 1100 0101$_2$ = 197$_{10}$

let's test this out:

hex: (12 * 16) + (5 * 1) = (192 + 5) = **197_{10}**
binary: 128 + 64 + 4 + 1 = **197_{10}**

56$_{16}$ = 0101 0110 = 86_{10}

let's test this out:

hex: (5 * 16) + (6 * 1) = (80 + 6) = **86_{10}**
binary: 64 + 16 + 4 + 2 = **86_{10}**

Do

1011 1111 1101 0101

B F D 5

Chris:

General:

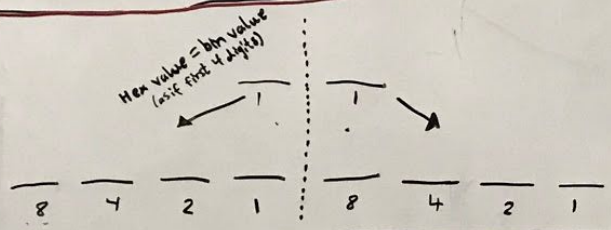
bin: 128 64 32 16 8 4 2 1 — X_2 ← number in binary

hex: 16 1 — 1 2 3 4 5 6 7 8 9 A B C D E F ← Digits — X_{16} ← number in hexadecimal
(10)(11)(12)(13)(14)(15)

X_{10} ← number in decimal

Ex. $100_{10} = 01100100_2 = 64_{16}$

HEX ⇌ BIN (Quick Method)



Ex. $10 = A\ 4 = 4$ → $A4 = 164$

$1010\ 0100 = 4$ → $10100100 = 164$

$10 = 8\ 4\ 2\ 1$

$128\ 64\ 32\ 16\ 8\ 4\ 2\ 1$

2

MINE