

Decimal = Base 10 = 10 digits (0 thru 9)

First 4 place values:

1000	100	10	1
10^3	10^2	10^1	10^0

Binary = Base 2 = 2 digits (0 thru 1)

First 4 place values:

8	4	2	1
2^3	2^2	2^1	2^0

Hexadecimal = Base 16 = 16 digits (0 thru 15, written as 0 thru F)

First 4 place values:

4096	256	16	1
16^3	16^2	16^1	16^0

Notice 2 rules from above, true for all number systems:

$X^0 = 1$ "Anything to the zero power equals 1"

$X^1 = X$ "Anything to the first power equals itself"

For hexadecimal, remember:

The digits above 9 are: A=10, B=11, C=12, D=13, E=14, F=15

So, how much (in decimal) is the Binary value 1110 worth?

Answer: $(1 \times 8) + (1 \times 4) + (1 \times 2) + (0 \times 1) = 14$ in decimal.

And, how much (in decimal) is the Hexadecimal value 20AF worth?

Answer: $(2 \times 4096) + (0 \times 256) + (10 \times 16) + (15 \times 1) = 8,367$ in decimal.

(Remember, the digit A is worth 10, the digit F is worth 15).

In the 2 examples above, notice we simply multiplied each DIGIT with the proper PLACE VALUE. If a digit is zero, of course, zero x place value is zero. Then we added them all up.

HEXADECIMAL IS A NATURAL SHORTHAND FOR BINARY.

There are exactly 16 possible combinations of 4 bits (a "nybble", or half byte):

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

So, we can show the value of any whole byte (8 bits) as simply 2 hexadecimal digits. Just split the byte IN HALF, and work each half AS A SEPARATE NUMBER:

1001 1100 in Binary:

Left side 1001 = 9, written as 9 in hexadecimal

Right side 1100 = 12, written as C in hexadecimal

So, binary 1001 1100 is simply 9C in hexadecimal.

You can go the other way as well, again splitting the byte:

A5 in hexadecimal:

Left side A = value 10 = 1010 in binary (one 8 plus one 2 gives 10)

Right side 5 = value 5 = 0101 in binary (one 4 plus one 1 gives 5)

So, hexadecimal A5 expands into 1010 0101 in binary.

Interesting facts:

You may know there are 1,024 bytes in a Kilobyte. Where did that come from?

1024	512	256	128	64	32	16	8	4	2	1
2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

So, 1024 came from 2 to the tenth power.

You will notice that the values in the above table show up a lot when you look at things like memory sticks, USB jump drives, etc.

Incidentally, there are exactly 256 combinations of 8 bits in a byte.

Smallest value = 0000 0000 = 00 in hex = value 0 in decimal

Largest value = 1111 1111 = FF in hex = value 255 in decimal*

*Do the math: FF is 2 digits in hexadecimal, so $(15 \times 16) + (15 \times 1) = 240 + 15 = 255$.

So, you can show the values 0 thru 255 in 8 binary digits, giving a total of 256 combinations. ASCII takes advantage of this to use certain combinations to represent uppercase letters, lowercase letters, digits, special characters, and even unprintable characters (such as blank and tab).

Note: when you buy hard drives, it looks like you lost some space when Windows shows you the total space on the drive. What happened?

Well, for better or worse (I think for worse), the hard drive manufacturers use 1,000 bytes = 1 KB, instead of 1,024 bytes per kilobyte. It adds up... for example, if you see on the box that it is a "1 Terabyte (TB)" hard drive, then it has 1,000,000,000,000 bytes (based on an even 1,000 bytes per kilobyte).

But Windows (and any self-respecting Geek), knows that we abide by 1,024 bytes per KB, 1,024 KB per MB, 1,024 MB per GB, 1,024 GB per TB. So, start dividing:

$1,000,000,000,000$ bytes (total actually bought) / 1024 bytes per KB = 976,562,500 KB

$976,562,500$ KB / 1024 KB per MB = 953,674 MB (rounded)

$953,674$ MB / 1024 MB per GB = 931 (rounded)

Yikes! It appears as if we lost nearly 70 GB!

Nope, you just bought 1,000,000,000,000 bytes instead of 1,099,511,627,776 bytes. Get over it.