

Decimal Numbers

We use a *positional number system*.

Consider 123:

the '1' resides in the 100s place

the '2' resides in the 10s place

the '3' resides in the 1s place.

The values of the places depend on *powers* of the base.

In decimal, the base is ten.

So, in decimal, we can also write 123 as $1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$.

Binary Numbers

Binary works a lot like decimal:

$$10110_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0,$$

which works out to $16 + 0 + 4 + 2 + 0 = 22$.

An Old Joke

Old joke: there are 10 types of people in the world, those who understand binary, and those who do not.

- An unlikely quiz or test question would be to ask you to explain the joke.

Hexadecimal

In principle, base 16 works the same as bases 2 and 10, but we often also use hex as a special convenience for representing binary numbers. Consider the following table:

Hex	Binary	Decimal
0	0000	0
1	0001	1
2		2
3		3
4		4
5		5
6		6
7		7
8		8
9		9
A	1010	10
B		11
C		12
D		13
E		14
F	1111	15

Every hex digit corresponds exactly to a unique group of four binary digits, and *vice versa*. Often problems expressed in hex are best solved by thinking in binary. Often binary problems are succinctly represented in hex.

Notation

If there is no indication of what base a number is, such as “123,” it is usually taken to be decimal.

One common notation is to use the base—in decimal—as a subscript:

$$14_{10}$$

$$14_{16} = 20_{10}$$

$$101_2 = 5_{10}$$

It is also very common to indicate a number is hexadecimal by preceding it with “0x”

$$0x101 = 16^2 + 1 = 257_{10}$$

Donald Knuth, a well-known computer scientist and mathematician, used to offer a hexadecimal dollar to anyone who found an error in one of his books.

- One hexadecimal dollar is **0x100** cents.

A Little Practice

0xa = 10 decimal.

0xaa = $10 \times 16 + 10 = 170$ decimal

What is it that sheep are trying to communicate?

0xbaa = $11 \times 16^2 + 10 \times 16 + 10 = 2986$ decimal

A lamb with a drawl:

0baaa = $2986 \times 16 + 10 = 47786$ decimal

Name one thing very few vegetarians eat:

0xbeef = $11 \times 16^3 + 14 \times 16^2 + 10 \times 16 + 15 = 48815$
decimal

On your own: *El día de los muertos* translates to the day of the **0xdead**. How do we express **0xdead** in decimal?

Converting from Hex to Binary is Easy

Recall the comment that each hexadecimal digit corresponds to four binary digits.

- A hex digit is four bits.

This makes things easy.

Convert `0xf3089ae4` to binary. Go digit-by-digit from the left (we could just as easily start at the right):

- `f` \Rightarrow 1111
- `3` \Rightarrow 0011
- `0` \Rightarrow 0000
- `8` \Rightarrow 1000
- `9` \Rightarrow 1001
- `a` \Rightarrow 1010
- `e` \Rightarrow 1110
- `4` \Rightarrow 0100

So `0xf3089ae4` = 11110011000010001001101011100100₂

Does anyone have a beef with that?

- `0xbeef` = 1011111011101111₂

Converting from Binary to Hex is Easy

Four bits make a hexadecimal digit. Consider:

$$1010_2 = 0\mathbf{xa}$$

Sometimes the number of bits is not divisible by four.

Example:

$$010_2 = 0\mathbf{x2}$$

What do we do with a longer number? Work from the right:

$$0100011_2 = 0\mathbf{x23}$$

Showing the steps:

- The rightmost four bits: **0011**
This is three.
- The rest of the bits: **010**
This is two.
- Put them together: **0x23**
- Leading zeroes can be dropped, just as in decimal.
- I left out the base (₂) since it was clear we were talking about bits.

Converting from Binary to Hex is Still Easy

A typical CPU word or IPv4 address is 32 bits. This is $32 \div 4 = 8$ hex digits. Convert

- $10111111100011100100110100000110_2$
- $1011\ 1111\ 1000\ 1110\ 0100\ 1101\ 0000\ 0110_2$ grouped
4 bits at a time
- $0xbf8e4d06$

Masks and And

A very important operation on numbers is *ANDing*, or finding the conjunction, conjoining, etc.

- This is oft-used in networking.

If $c = a \& b$, then c is the number with a one in every position that is one in both a and b , and a zero in every other bit position.

Examples:

- $0 \& 0 = 0, 0 \& 1 = 0, 1 \& 0 = 0, 1 \& 1 = 1$
- $0101 \& 0011 = 0001$
- $213_{10} \& 1f_{16} = 11010101_2 \& 00011111_2 = 00010101_2$

Masks are used to indicate which bits of a number one is interested in.

- If we are interested in the high 3 bits of a byte, we would use `11100000`.
- If we are interested in the high 20 bits of a 32 bit word, we use `0xffff000`. Why?

Combinatorics

Suppose I ask you how many different things you can *enumerate* with 3 decimal digits.

This would correspond to the range 0–999, which is a thousand different numbers, so the answer is one thousand.

Another way to get the answer is to realize that there are 10^3 different combinations of 3 decimal digits.

How about if I ask you how many things you can enumerate with 5 decimal digits? With one decimal digit?

In Binary

So, now I ask how many things you can enumerate with 3 Binary dIgiTs, i.e., 3 bits. How many?

With one?

With 5?

With 12 bits?

There are a few powers of 2 and 10 worth remembering:

Abbrev	Decimal	Binary
k	$10^3 = 1000$	$2^{10} = 1024$
M	$10^6 = 1,000,000$	$2^{20} = 1,048,576$
G	$10^9 = 1,000,000,000$	$2^{30} = 1,073,741,824$

Note that, in addition to using the function given earlier to calculate exponents, we can count on our fingers. This is what I usually do.

How high can I count on the fingers of one hand? On both hands? If I toss in my toes?

Note: the binary powers are also sometimes expressed as *mebi-* (Mi), *kibi-* (Ki), etc.

An Identity

Identity: $2^{a+b} = 2^a \times 2^b$

So, what is 2^{32} ?

$$\begin{aligned} 2^{32} &= 2^{30+2} \\ &= 2^{30} \times 2^2 \\ &= 1k \times 1k \times 1k \times 4 \\ &= 1G \times 4 \\ &= 4G \end{aligned}$$

Why do we care?

- In 32 bits we can enumerate 2^{32} different things.
- This is 4G different things.

Converting from Decimal to Binary

Suppose we want to convert 212 from decimal to binary.

Consider the structure of a binary number: the rightmost digit is the ones place, then the twos, then the 4s, and so on.

212 is even, so the ones place will be zero, so we know the binary will look like $b \dots b0$.

- $212 \div 2 = 106$, which is even, so there is a zero in the twos place: $d \dots d00$.
- $106 \div 2 = 53$, which is odd, so the fours place is one: $d \dots d100$.
- $53 \div 2 = 26$, which is even, so the eights place is zero: $d \dots d0100$.
- Continuing: $13 \Rightarrow d \dots d10100$
- $6 \Rightarrow d \dots d010100$
- $3 \Rightarrow d \dots d1010100$
- $1 \Rightarrow 11010100$

Double-checking the calculation is smart:

$$2^2 + 2^4 + 2^6 + 2^7 = 4 + 16 + 64 + 128 = 212$$

Converting from Decimal to Hexadecimal

Recall that a hex digit is just a grouping of four binary digits. So, to convert 212_{10} to hex, we can convert it to binary and then group the bits:

1101 0100

We know that $1101_2 = d_{16}$ and that $0100_2 = 4_{16}$, so the answer is $d4_{16}$, which is often written $0xd4$.

$$d4_{16} = 13 \times 16 + 4 = 212$$

Another way to do the calculation is by repeated division by 16:

- $212 \div 16 = 13$ with remainder 4
- $13 \div 16 = 0$ with remainder 13.

This is really the same algorithm we used to convert from decimal to binary in the first place, just with a different denominator.

More Conversions: Binary to Hex, Hex to Binary

Going Backwards

How many things can we enumerate in n bits? 2^n

How many bits do we need to enumerate n things?

A simple way to answer this is to ask how many bits we need to represent the number $n - 1$.

Example: how many bits do we need to enumerate 100 things?

- $n = 100 \Rightarrow n - 1 = 99$
- $99_{10} = 1100011_2$, which is 7 bits.

Another way to figure out how many bits we need to enumerate n things is to figure out the first power of two at least as big as $n - 1$:

- $n - 1 = 99$ (as above)
- $2^6 = 64$, $2^7 = 128$
- So we need 7 bits.

Logarithms

What we were really doing in figuring the smallest adequate power of two was taking a logarithm to the base 2.

Really, the number of bits we need to enumerate n things is $\lceil \log_2 n \rceil$ (assuming $n \geq 2$).

Properties of Logarithms

First, the most basic definition: $\log_a a^x = x$.

Next, three useful identities:

$$\log_a(x_1x_2) = \log_a x_1 + \log_a x_2$$

$$\log_a x^b = b \log_a x$$

$$\log_a x = \frac{\log_b x}{\log_b a} = (\log_b x)(\log_a b)$$

So, specifically,

$$\log_2 x = \frac{\log_{10} x}{\log_{10} 2}$$

This is useful because many calculators will do \log_{10} , but not \log_2 .

An Observation on Growth

When do you use a log, and when do you use exponentiation?

First, note that log is a very slow-growing function, but exponentiation grows quickly (note the oft-misused phrase “exponential growth”). E.g.,

- $\log_2 16 = 4$, $\log_2 32 = 5$
- $2^{16} = 65536$, $2^{17} = 131072$

Note the difference between:

- log of 1, 2, 3, 4, 5, 6, 7, 8
- 2 raised to the 1, 2, 3, 4, 5, 6, 7, 8

Average, Mean, Median

Usually when people speak of the average, they mean the mean.

Sometimes we're really interested in the median. Why?