(missed start)

## Counting in Binary

If we have n bits in a binary system, we have $2^n$ possibilities or things. Each state can correspond to a different thing. These can be numbers, colours, whatever. There can't be any more than $2^n$ states.

We can assign some bits to denote the decimal place and others to denote the exponent for a floating point number.
But how many bits to each thing? Increase accuracy by representing more places, and we run out of range because we don't have enough bits for the exponent. Also vice-versa.

We are limited by the physical capabilities of the machine.
A physical machine is limited in the number of bits it can hold & directly manipulate.

These limitations can be mitigated but not completely eliminated.

Example: The max addressible memory in a 32-bit machine is equal to $2^{32}$ different locations. 4 gigalocations. Each location typically holds 1 byte, so 4 gigabytes of memory is the max in a 32-bit machine.

## Base 8, Base 16

Going between bases that are not a perfect power of 2 requires some work, as seen before.
For bases that are a perfect power of 2, we can make very quick comparisons.

Each digit in that base will have a corresponding unique bit pattern using the number of bits in the associated base, with no bit patterns left over.

For base 8 ($2^3$, a.k.a. octal), we can represent each digit as a unique combination of 3 bits:

| | |
|---|---|
| 000 | 0 |

| | |
|---|---|
| 001 | 1 |
| 010 | 2 |
| 011 | 3 |
| 100 | 4 |
| 101 | 5 |
| 110 | 6 |
| 111 | 7 |

For base 16 ($2^4$, a.k.a. hexadecimal):

| | |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | A |
| 1011 | B |
| 1100 | C |
| 1101 | D |
| 1110 | E |
| 1111 | F |

## Simplifying Conversions Between Binary and Bases that Are a Perfect Power of 2

To convert between binary and octal, substitute the corresponding 3 bits for each octal digit & vice-versa.

For hex and binary, it's the same but with 4 bits.

Example 1: Convert $1247_8$ to binary

1→001

2→010

4→100

7→111

So the answer is 001010100111.

To go the other way, break the binary number into groups of three, starting at the least significant side, and substitute the octal digits appropriately.

Example 2: Convert $1101101011_2$ to octal

Break the sequence into groups of 3, starting from the right-hand side, appending 0s to the left-hand side if needed to make groups of 3.

Thus: 001 101 101 011

011→3

101→5

101→5

001→1

So the answer is $1553_8$

Example 3: Convert $FE10A_{16}$ → binary

F→1111

E→1110

1→0001

0→0000

A→1010

So the answer is $1111111000010000101 0_2$

Example 4: Convert $110111001_2$ to hex.

Start on the RHS and break into groups of 4, adding 0s to the LHS if needed.

0001→1

1011→B

1001→9

So the answer is $1B9_{16}$

Converting from binary to octal/hexadecimal represents a second level of abstraction in our journey from the machine to higher levels of information representation/manipulation.

## Bytes

Bits are almost always bundled together into 8-bit collections called bytes. 1 byte can be represented by 2 hex digits (because each hex digit corresponds to 4 bits). A hex digit is a nibble.

The lab allocation is on the website now, though labs won't start for another week or two.