ECS404: Computer Systems and Networks

Digital Representation

Week 1 Pt 4: Binary

Aims

- To get you used to the idea of binary, and some of its basic properties.
- To teach you how about the use of different bases for arithmetic, that binary is base 2, and hence how positive whole numbers are represented in binary.

Week 1: digital representation

By the end of this week you should:

1. understand that a bit is a single binary digit, and understand the difference between bits and bytes;

2. understand the correspondence between bit sequences and unsigned integers and in particular:

   a) be able to translate numbers from decimal to binary and vice versa

Computers use binary

- **Binary** means things are represented by codes made up out of 0’s and 1’s.
- Each 0 or 1 is called a **bit**

Binary: examples

- 0000 0101 represents the number 5
- 1111 1011 can represent the number -5
- 0111 1000 represents the letter x

Binary: examples

- 0000 0101 represents the number 5
- 1111 1011 can represent the number -5
- 0111 1000 represents the letter x
- these examples all use 8 bits

Bits and bytes

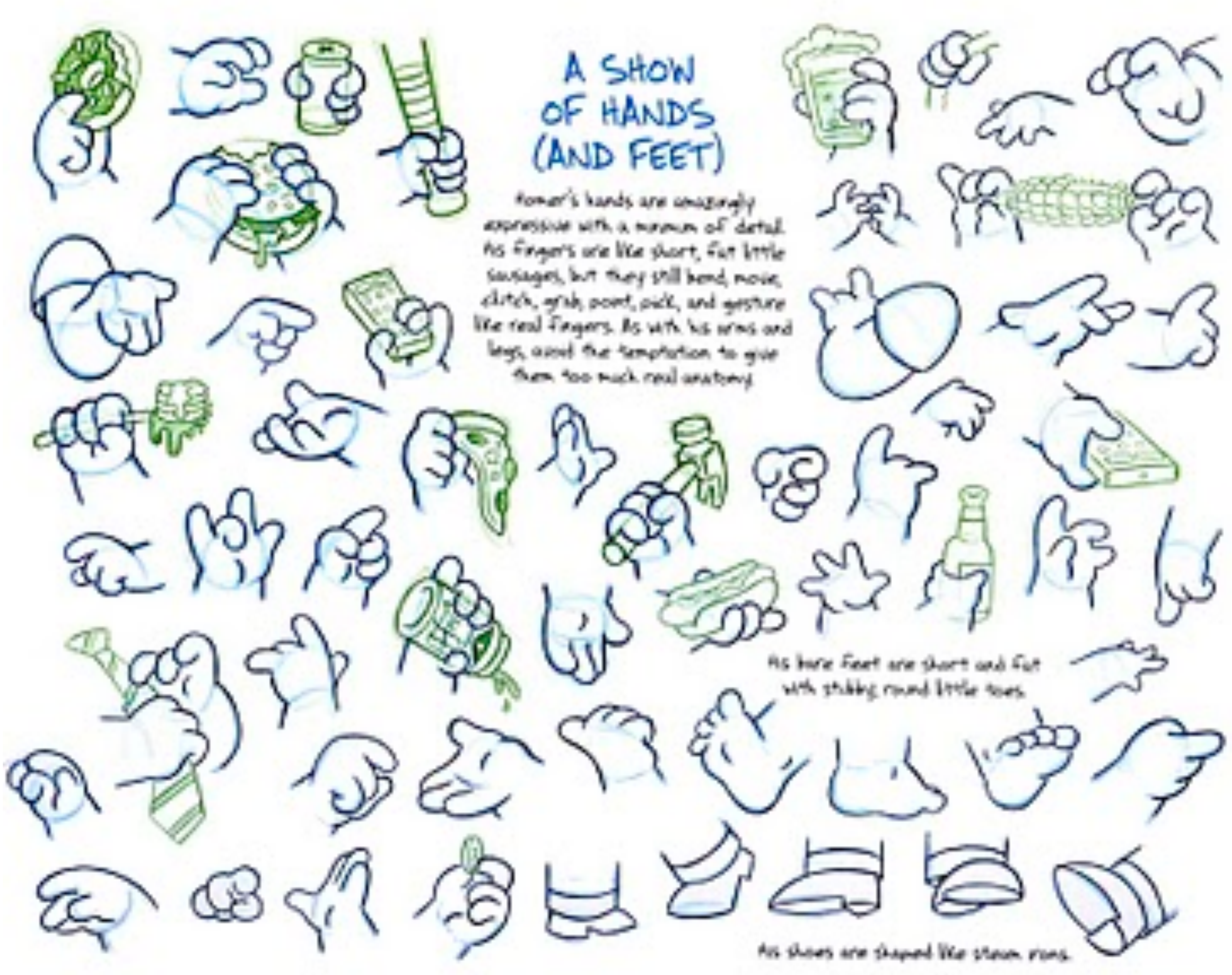
- 8 bits is called a **byte**
- At one time memory would deliver information in byte-sized units.
- We still use multiples of bytes for storage: 2 bytes = 16 bits, 4 bytes = 32 bits, 8 bytes = 64 bits.

- So why do computers use binary instead of decimal?
- Come to that why do we use decimal?

A thought experiment

- Why do we count in units of 10?

Here’s a picture to help



and here's another

A thought experiment

- Why do we count in units of ten?
- What would happen if we had eight fingers?
- What would happen if we had twelve fingers?
- Is 10 a good choice, mathematically?

How do we represent (positive whole) numbers?

- Only signs used are digits: 0,1,…,9
- Value of each digit depends on its place in the number.

Place notation

2017

Thousands	Hundreds	Tens	Units
1000	100	10	1
10^3	10^2	10^1	10^0

10 is not special

2017

512’s	64’s	Eights	Units
512	64	8	1
8^3	8^2	8^1	8^0

This is called working in a different **base**

2017

512’s	64’s	Eights	Units
512	64	8	1
8^3	8^2	8^1	8^0

What is 2017 in base 8?

2017

512’s	64’s	Eights	Units
512	64	8	1
8^3	8^2	8^1	8^0

2*512 + 0*64 + 1*8 + 7*1 = 1039

2017_8

2017_8

$= 2*512 + 0*64 + 1*8 + 7*1$

$= 1024+8+7$

$= 1039_{10}$

You can do this with whatever base you like, eg 16

2017

4096's	256's	16's	Units
4096	256	16	1
16^3	16^2	16^1	16^0

2017_{16}

2017_{16}

$= 2*4096 + 0*256 + 1*16 +7$

$= 8192 + 16 + 7$

$= 8215$

Binary

is base 2

Binary

- Equivalent of 10, 100, 1000 etc is powers of 2
- These should become familiar friends:

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

binary/decimal conversion

Converting from binary to decimal

digits	1	0	1	1
powers	8	4	2	1
worth	8	0	2	1
sums	8		10	11

$1011_2 = 11_{10}$

base

Converting from binary to decimal

digits	1	0	1	0	1	0	1	0
powers	128	64	32	16	8	4	2	1
worth	128	0	32	0	8	0	2	0
sums	128	128	160	160	168	168	170	170

$10101010_2 = 170_{10}$

Converting from decimal to binary

sums		13	5		1
powers	16	8	4	2	1
diff	0	5	1	2	0
binary		1	1	0	1

$13_{10}=1101_2$

Converting from decimal to binary

sums	174		46	0	14	6	2	
powers	128	64	32	16	8	4	2	1
diff	46	0	14	0	6	2	0	
binary	1	0	1	0	1	1	1	0

$174_{10}=10101110_2$

Summary

- Computers use binary which is base 2
- But the way we use positional notation to write numbers in binary is similar to the way we write numbers in decimal.
- We have seen basic methods to convert between binary and decimal and vice versa.
- These work for small numbers, but there are other methods that work better with large ones.