computer science illuminated

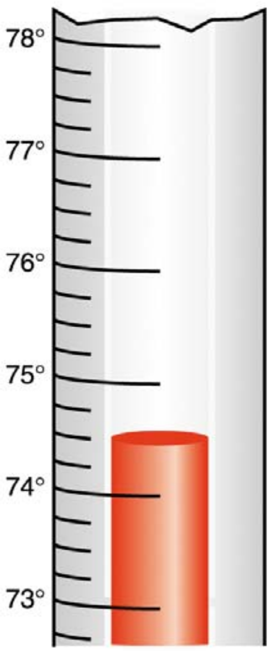
Representing Information Digitally (Number systems)

Nell Dale & John Lewis

(adapted by Michael Goldwasser)

Analog vs. Digital Information

- **Analog data** is a continuous representation, analogous to the actual information it represents. A mercury thermometer is an analog device. The mercury rises in a continuous flow in the tube in direct proportion to the temperature.



- **Digital data** is a discrete representation, breaking the information up into separate elements.

The “bit” (binary digit)

All of the historical computing devices we considered were inherently **digital**. (mechanisms had fixed number of states)

Modern computers, more specifically, are based on storing information using physical means that can be in one of **two** distinct states.

We call such a physical unit a **bit**.

Two-state Representation of Data

- "thumbs up" vs. "thumbs down"
- Flag can be raised or lowered
- dots/dashes in Morse Code
- light switch with circuit opened/closed
- "cores" (electromagnets - positive/negative field)
- "capacitor" (two small metallic plates with a small separation; can be statically charged/discharged)
- "flip/flop" (electronic circuit with output as high/low voltage signal)

Pros/Cons for Data Storage

Varying level of power usage, cost, volatility:

- Core will keep its charge even after power is shutoff (but not as quick to access)
- Flip-Flop will lose its data when power is turned off (but extremely quick/cheap)
- Capacitors use such small charges, they sometimes lose their charge while running unless recharged regularly (“dynamic memory”)

Binary and Computers

As bits have many physical realizations, they also have many symbolic representations.

Low voltage	High voltage
‘0’	‘1’
‘false’	‘true’
‘off’	‘on’
‘open’	‘closed’

Orders of Magnitude

- **One bit** can be in one of **two** distinct states (0,1)
- **Two bits** can be in one of **four** distinct states
- **Three bits** can be in one of **eight** distinct states
- **n bits** can be in one of $\mathbf{2^n}$ distinct states

1 Bit	2 Bits	3 Bits	4 Bits	5 Bits
0	00	000	0000	00000
1	01	001	0001	00001
	10	010	0010	00010
	11	011	0011	00011
		100	0100	00100
		101	0101	00101
		110	0110	00110
		111	0111	00111
			1000	01000
			1001	01001
			1010	01010
			1011	01011
			1100	01100
			1101	01101
			1110	01110
			1111	01111
				10000
				10001
				10010
				10011
				10100
				10101
				10110
				10111
				11000
				11001
				11010
				11011
				11100
				11101
				11110
				11111

Orders of Magnitude

Common Quantities of Storage

1 byte	= 8 bits	(thus 256 distinct settings)
1 Kilobyte (KB)	= 2^{10} bytes	= 1024 bytes
1 Megabyte (MB)	= 2^{10} KB	= 1048576 bytes
1 Gigabyte (GB)	= 2^{10} MB	= 1billion^{+} bytes
1 Terabyte (TB)	= 2^{10} GB	= 1trillion^{+} bytes
1 Petabyte	= 2^{10} GB	= 2^{50} bytes

Representing Numbers

Natural Numbers (0, 1, 2, 3, ….)

Integers (…, -3, -2, -1, 0, 1, 2, 3, …)

Rational Numbers (e.g., -249, -1, 0, $\frac{1}{4}$, - $\frac{1}{2}$)
An integer or the quotient of two integers

Irrational Numbers (e.g., π, $\sqrt{2}$)
Values not expressable as quotient of integers

<u>Let’s just focus on Natural Numbers for now…</u>

Natural Numbers

How many ones are there in 642?

600 + 40 + 2 ?

Or is it

384 + 32 + 2 ?

Or maybe…

1536 + 64 + 2 ?

Natural Numbers

Aha!

642 is 600 + 40 + 2 in **BASE 10**

The **base** of a number determines the number of digits and the value of digit positions

(Why was base 10 chosen by humans?)

Positional Notation

Continuing with our example…

642 in base 10 *positional notation* is:

$$
\begin{aligned}
& 6 \times 10^2 = 6 \times 100 = 600 \\
+ & 4 \times 10^1 = 4 \times 10 = 40 \\
+ & 2 \times 10^0 = 2 \times 1 = 2 = 642 \text{ in base 10}
\end{aligned}
$$

This number is in base 10

The power indicates the position of the number

Positional Notation

What if 642 has the base of 13?

$$
\begin{aligned}
&+ 6 \times 13^2 = 6 \times 169 = 1014 \\
&+ 4 \times 13^1 = 4 \times 13 = 52 \\
&+ 2 \times 13^0 = 2 \times 1 = 2 \\
&= 1068 \text{ in base } 10
\end{aligned}
$$

642 in base 13 is equivalent to 1068 in base 10

Binary

Decimal is base 10 and has 10 digits: 0,1,2,3,4,5,6,7,8,9

Binary is base 2 and has 2 digits: 0,1

For a number to exist in a given number system, the number system must include those digits. For example:
The number 284 only exists in base 9 and higher.

Converting Binary to Decimal

What is the decimal equivalent of the binary number 1101100?

$$
\begin{aligned}
& 1 \times 2^6 = 1 \times 64 = 64 \\
+ & 1 \times 2^5 = 1 \times 32 = 32 \\
+ & 0 \times 2^4 = 0 \times 16 = 0 \\
+ & 1 \times 2^3 = 1 \times 8 = 8 \\
+ & 1 \times 2^2 = 1 \times 4 = 4 \\
+ & 0 \times 2^1 = 0 \times 2 = 0 \\
+ & 0 \times 2^0 = 0 \times 1 = 0 \\
& = 108 \text{ in base } 10
\end{aligned}
$$

Bases Higher than 10

How are digits in bases higher than 10 represented?

Base 16:
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, and F

Converting Hexadecimal to Decimal

What is the decimal equivalent of the hexadecimal number DEF?

$$
\begin{aligned}
& D \times 16^2 &= 13 \times 256 &= 3328 \\
+\ & E \times 16^1 &= 14 \times 16 &= 224 \\
+\ & F \times 16^0 &= 15 \times 1 &= 15 \\
& & &= 3567 \text{ in base 10}
\end{aligned}
$$

Remember, base 16 is
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Converting Octal to Decimal

What is the decimal equivalent of the octal number 642?

$$
\begin{aligned}
6 \times 8^2 &= 6 \times 64 &= 384 \\
+\ 4 \times 8^1 &= 4 \times 8 &= 32 \\
+\ 2 \times 8^0 &= 2 \times 1 &= 2 \\
& &= 418 \text{ in base } 10
\end{aligned}
$$

Power of 2 Number System

Binary	Octal	Decimal
0000	00	00
0001	01	01
0010	02	02
0011	03	03
0100	04	04
0101	05	05
0110	06	06
0111	07	07
1000	10	08
1001	11	09
1010	12	10

Converting Binary to Octal

- Groups of Three (from right)
- Convert each group

10101011	**10**	**101**	**011**
	2	**5**	**3**

10101011 is 253 in base 8

Converting Binary to Hexadecimal

- Groups of Four (from right)
- Convert each group

10101011 **<u>1010</u>** **<u>1011</u>**

A **B**

10101011 is 89 in base 16

Converting Decimal to Other Bases

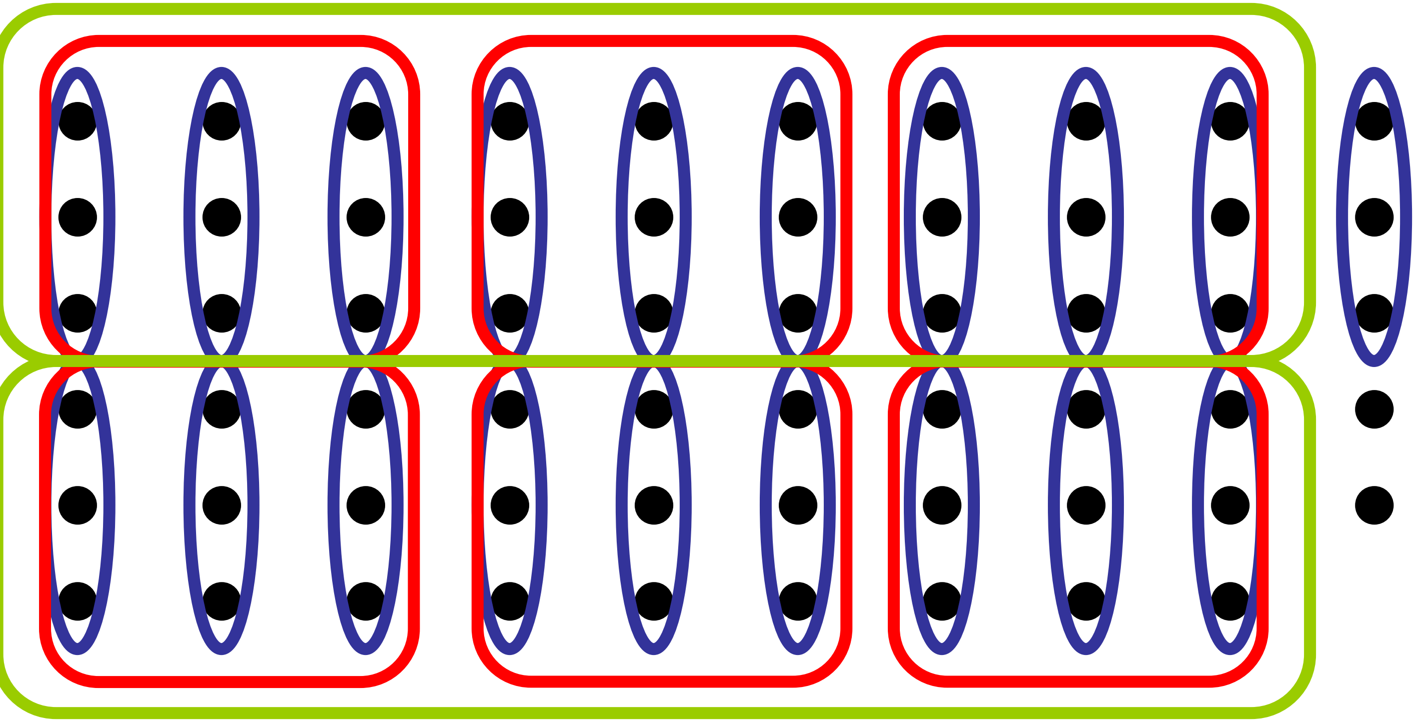
Algorithm for converting base 10 to other bases:

While the quotient is ***not*** zero:

1. Divide the decimal number by the new base
2. Make the remainder the next digit to the left in the answer
3. Replace the decimal number with the quotient

Why does this method work?

Let's convert 59 (base 10) to base 3:



$$3\,\overline{)\,59}\quad 19 \text{ R } 2$$

$$3\,\overline{)\,19}\quad 6 \text{ R } 1$$

$$3\,\overline{)\,6}\quad 2 \text{ R } 0$$

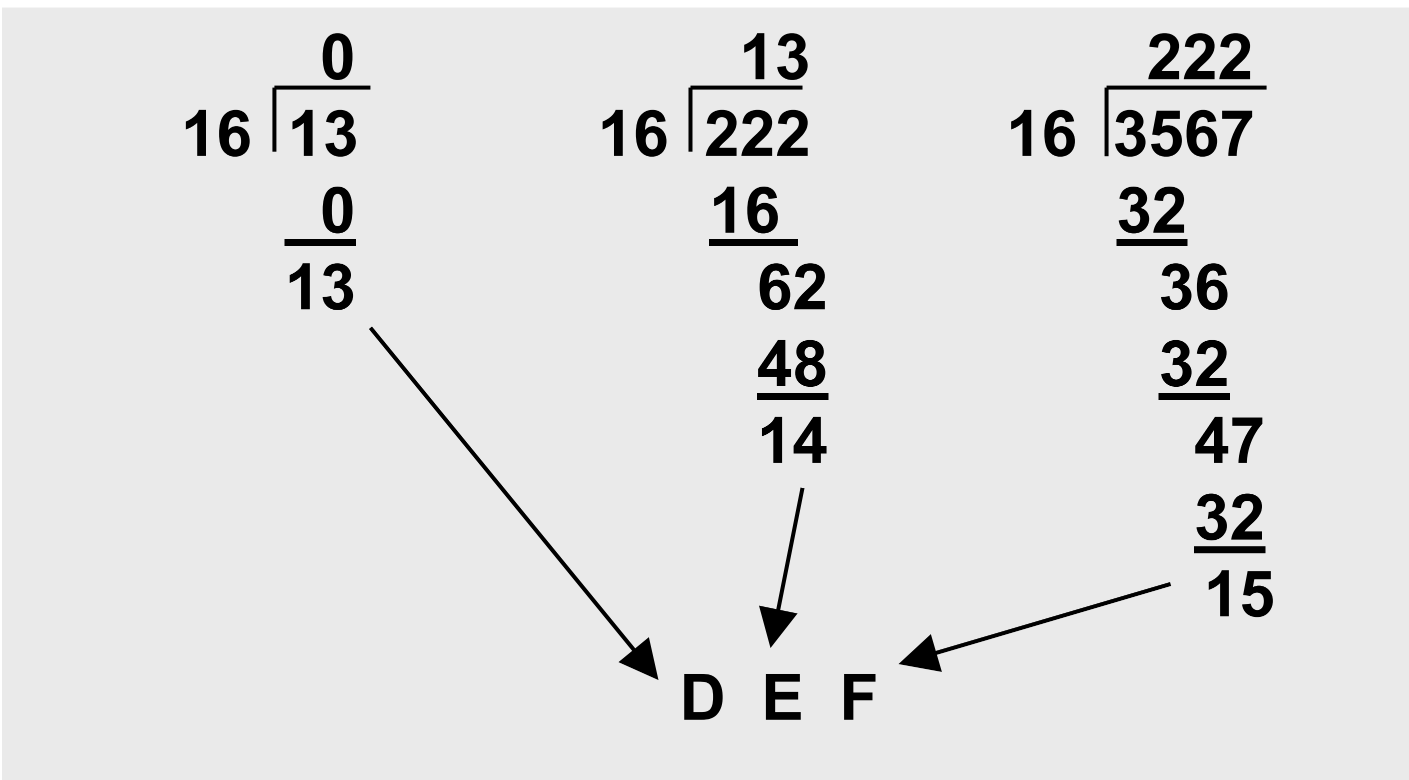
$$3\,\overline{)\,2}\quad 0 \text{ R } 2$$

2 0 1 2

Converting Decimal to Hexadecimal

Try another Conversion:

3567 (base 10) is what number in base 16?



Converting Decimal to Other Bases

Try another Conversion:

The base 10 number 108

is what number in base 5?

Arithmetic in Decimal

Let's start with base 10:

```
   1     1 1
   3 5 7 2 5 7
 +   6 2 5 4 5
 -------------
   4 1 9 8 0 2
```

Carry Values

Arithmetic in Other Bases

What if this is in base 8?

```
  1  1 1 1 1        <- Carry Values
   3 5 7 2 5 7
 +   7 2 5 4 5
 -------------
  4  5 2 0  2 4
```

Arithmetic in Binary

Remember: there are only 2 digits in binary:
0 and 1

Position is key, carry values are used:

```
 1   1 1 1 1 1      <---- Carry Values
   1 0 1 0 1 1 1
  +1 0 0 1 0 1 1
  --------------
 1 0 1 0 0 0 1 0
```

Subtracting Binary Numbers

Remember borrowing? Apply that concept here:

```
    1 2
  0 2 0 2
  1 0 1 0 1 1 1
-   1 1 1 0 1 1
  -------------
  0 0 1 1 1 0 0
```