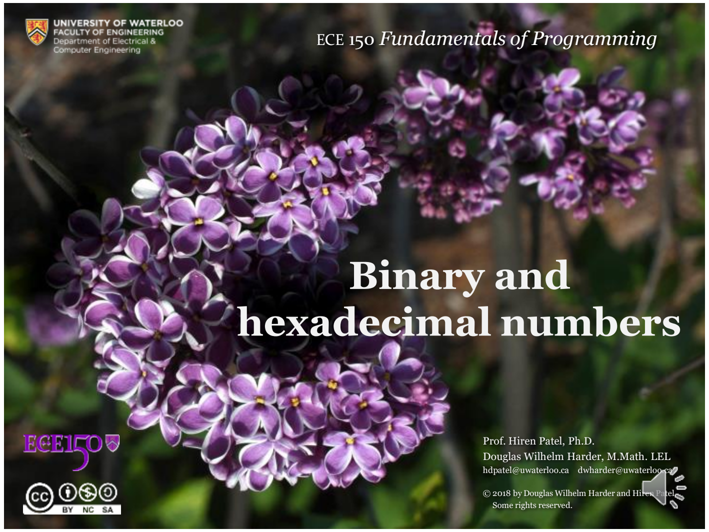

Outline

- In this lesson, we will:
  - Review counting
  - Consider what happens if we had only eight fingers
  - Introduce binary numbers
  - Look at addition of binary numbers
  - Introduce hexadecimal numbers
  - See how hexadecimal numbers an represent binary numbers

Counting

- In our counting system, we have ten different digits
- Our numbering system is *positional*
  - The significance of a number depends on its position

דוד

1942

$1 \times 10^3 = 1000$

$9 \times 10^2 = 900$

$4 \times 10^1 = 40$

$2 \times 10^0 = 2$

  - The number 1942 represents the total of these numbers

- Note, if "9" represents XXXXXXXXX objects,
  then "10" represents XXXXXXXXXX objects

Base 8

- Base 10 is great for humans: we have a total of 10 fingers
- Suppose humans had eight fingers, so we only and eight digits:

  0, 1, 2, 3, 4, 5, 6, 7
- Thus, the numbers would look like:

  0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, …

```
   1  111  1
   43573532
 + 61546073
  125341625
```

1 + 0 + 0 = 1

4 + 6 = 12

1 + 3 + 1 = 5

1 + 5 + 5 = 13

1 + 7 + 4 = 14

3 + 6 = 11

1 + 5 + 0 = 6

3 + 7 = 12

2 + 3 = 5

Base 8

- Therefore, each of these base-8 numbers would represent a certain number of "things":

0	zero	
1	one	X
2	two	XX
3	three	XXX
4	four	XXXX
5	five	XXXXX
6	six	XXXXXX
7	seven	XXXXXXX
10	eight	XXXXXXXX
11	nine	XXXXXXXXX
12	ten	XXXXXXXXXX
13	eleven	XXXXXXXXXXX
14	twelve	XXXXXXXXXXXX

Base 8

- To convert a base-8 number to base 10, we calculate:

4173

$3 \times 8^0 = 3$

$4 \times 8^3 = 2048$

$7 \times 8^1 = 56$

$1 \times 8^2 = 64$

$2048 + 64 + 56 + 3 = 2171$

Base 2

- In a computer,
  a number must be stored as a voltage
- Having ten different voltages is difficult to design and maintain
  - Some of the earliest computers did use base 10
    - Back in the 1940s
- Instead, it is easiest to have only two voltages:
  - 0 is represented by 0 V
  - 1 is represented by 5 V
- That is, we only have two digits, 0 and 1
  - We will call these *bits* from *b*inary dig*its*

Counting

- This means we have only two digits now:
  0, 1
- Thus, the number after "1" is "10",
  so 1 + 1 = 10
  also, 10 + 1 = 11
  thus, 11 + 1 = 100

Counting

- Therefore, each of these base-2 numbers would represent a certain number of "things":

0	zero	
1	one	X
10	two	XX
11	three	XXX
100	four	XXXX
101	five	XXXXX
110	six	XXXXXX
111	seven	XXXXXXX
1000	eight	XXXXXXXX
1001	nine	XXXXXXXXX
1010	ten	XXXXXXXXXX
1011	eleven	XXXXXXXXXXX
1100	twelve	XXXXXXXXXXXX

Counting

- It is useful to recognize very specific values:

1	one	2^0
10	two	2^1
100	four	2^2
1000	eight	2^3
10000	sixteen	2^4
100000	thirty two	2^5
1000000	sixty four	2^6
10000000	128	2^7
100000000	256	2^8
1000000000	512	2^9
10000000000	1024	2^{10}

Adding powers of two

- Adding numbers in base 10 is something you have learned
  - In base 10, adding powers of 10 is easy: 10 + 100 + 10000 = 10110
- Adding powers of two (10_2) in base 2 is also quite easy:

```
   13
      2                 10
      8               1000
     64            1000000
    128           10000000
    256          100000000
 + 1024      + 10000000000
   1482        10111001010
```


Representation

- In base 10, every digit is associated with a power of 10:

  123456789

- We will say that:
  - The "9" is 9×10^0, so we will say that nine is in the *zeroeth* position
  - The "8" is 8×10^1, so we will say that eight is in the *first* position
  - The "1" is 1×10^8, so we will say that one is in the *eighth* position

Representation

- To figure out what a binary number represents in decimal, we have

```
100010011101
```

$2^0 = 1$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^7 = 128$
$2^{11} = 2048$

  – Thus 1 + 4 + 8 + 16 + 128 + 2048 = 2205

Adding two binary numbers

- To add two binary numbers, the process is the same as adding two decimal numbers
  – However, you only need to remember that:
    1 + 1 = 10
    1 + 1 + 1 = 11

```
  1 1  1 1 1  1 1
   110001101101
 +  10011100100
 1000101010001
```

- We can even check our answer:
  1 + 4 + 8 + 32 + 64 + 1024 + 2048 = 3181
  4 + 32 + 64 + 128 + 1024 = 1252
  1 + 16 + 64 + 256 + 4096 = 4433 = 3181 + 1252

Counting in binary

- Question: Is 100 equal to 10^2 or 4?
  – We will usually:
    - Prefix binary numbers with "`0b`"
    - Use the monospaced typeface `Consolas`
  – Thus:
    - 100011010 is a large decimal number
    - `0b11110110010` is binary for 1970
  – To start, we will gray-out the "`0b`"

Addition

- Just like addition with decimal numbers, you can do the same with binary, you only have to remember:
  `1 + 1 = 10` and `1 + 1 + 1 = 11`

```
  1 1 1              1 1 1 1
   3725                1010
 + 8982             +   111
  12707               10001

    1 1  1  1 1 1      1 1               1 1 1 1 1   1   1 1    1 1
          9275948135782                      1110010001100111
 + 5032925823853222553                   + 1101111011011000110
   5033018583334583335                     10001011011100101101
```

Counting in binary

There are only 10 types of people in the world.

Those who understand binary, and those who do not.

Verbosity

- It seems it takes more bits to represent a number in binary than it does in decimal
  - It's not that bad:
    it only takes approximately $\log_2(10) \approx 3.3$ times as many bits
  - For example, 8357 requires four decimal digits,
    it would require approximately $4 \times 3.3 = 13.2$
    - In fact, it requires fourteen bits: `0b10000010100101`
- The computer doesn't care,
  but it's more frustrating for a human to deal in binary

Base 16: hexadecimal

- Suppose instead, we had 16 digits, and not 10:

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f

- In base 16,

a	ten
b	eleven
c	twelve
d	thirteen
e	fourteen
f	fifteen
10	sixteen

Counting in hexadecimal

- Question: Is 5923 equal to 5923 or something else in base 16?
  - We will usually:
    - Prefix hexadecimal numbers with "`0x`"
    - Use the monospaced typeface `Consolas`
  - Thus:
    - 5923 is a decimal number
    - `0x5923` is a hexadecimal number for 22819
  - To start, we will gray-out the "`0x`"

Verbosity

- It seems it takes more fewer hexadecimal digits to represent a number in hexadecimal than it does in decimal
  - It's only slightly better:
    it takes approximately $\log_{16}(10) \approx 0.83$ times as many hexadecimal digits
  - For example, 8357 requires four decimal digits,
    it would require approximately $4 \times 0.83 = 3.32$ hexadecimal digits
    - In fact, it still four: `0x20a5`

Why hexadecimal?

- Hexadecimal is an easy way to represent a binary number:

0	0b0	0x0	0b0000
1	0b1	0x1	0b0001
2	0b10	0x2	0b0010
3	0b11	0x3	0b0011
4	0b100	0x4	0b0100
5	0b101	0x5	0b0101
6	0b110	0x6	0b0110
7	0b111	0x7	0b0111
8	0b1000	0x8	0b1000
9	0b1001	0x9	0b1001
10	0b1010	0xa	0b1010
11	0b1011	0xb	0b1011
12	0b1100	0xc	0b1100
13	0b1101	0xd	0b1101
14	0b1110	0xe	0b1110
15	0b1111	0xf	0b1111

Converting binary to hexadecimal

- We will only use hexadecimal to easily represent a binary number:
  - To convert a binary number to hexadecimal:
    - Split the binary number into groups of four starting at the least-significant bit
      - Pad with zeros to the left if necessary

0001	0101	1100	0110	1000	1101	0010	1110	1010
1	5	c	6	8	d	2	e	a

    - Replace each group of four bits with the corresponding hexadecimal digit

- Thus, in hexadecimal, this binary number is
  `0x15c68d2ea`

0x0	0b0000
0x1	0b0001
0x2	0b0010
0x3	0b0011
0x4	0b0100
0x5	0b0101
0x6	0b0110
0x7	0b0111
0x8	0b1000
0x9	0b1001
0xa	0b1010
0xb	0b1011
0xc	0b1100
0xd	0b1101
0xe	0b1110
0xf	0b1111

Converting hexadecimal to binary

- We can easily convert a hexadecimal number back to binary:
  - To convert a hexadecimal number to binary:
    - Replace each hexadecimal digit with its corresponding four bits

`0xffa307b8`

`11111111101000110000011110111000`

- Thus, in binary, this hexadecimal number is
  `0b11111111101000110000011110111000`

0x0	0b0000
0x1	0b0001
0x2	0b0010
0x3	0b0011
0x4	0b0100
0x5	0b0101
0x6	0b0110
0x7	0b0111
0x8	0b1000
0x9	0b1001
0xa	0b1010
0xb	0b1011
0xc	0b1100
0xd	0b1101
0xe	0b1110
0xf	0b1111

Counting in hexadecimal

- In this course,
  you will not be required to add two hexadecimal numbers
- You must, however, understand that if you have the hexadecimal number

0xff3a04	0xff3a05	0xff3a06
0x5afe	0x5aff	0x5b00
0xa520f	0xa5210	0xa5211

Beyond the scope of this course

- Now, you could create addition and multiplication tables for both binary and hexadecimal numbers
  - You could define binary and hexadecimal multiplication and division
  - You could do everything you do with decimal numbers in binary or in hexadecimal
- However, you don't care for this course
  - What is covered here is all you will really need:
    - Converting between binary and decimal
    - Adding two binary numbers
    - Converting between binary and hexadecimal
    - Understanding the relative order of both binary and hexadecimal numbers

Summary

- Following this lesson, you now
  - Understand that computer use binary numbers
  - Know that the digits `0` and `1` are called bits
    - Binary numbers are prefixed by `"0b"`
  - See that binary addition mirrors decimal addition
  - You know that the hexadecimal digits are `0` through `9` and `a b c d e f`
  - Understand that binary numbers are verbose
    and hexadecimal representations are more compact
    - Hexadecimal numbers are prefixed by `"0x"`
  - Know how to translate between binary and hexadecimal and back
    - You don't care what decimal value a hexadecimal number is…

References

[1] Wikipedia:
https://en.wikipedia.org/wiki/Binary_number
https://en.wikipedia.org/wiki/Hexadecimal
https://simple.wikipedia.org/wiki/Hexadecimal_numeral_system

Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.

Disclaimer