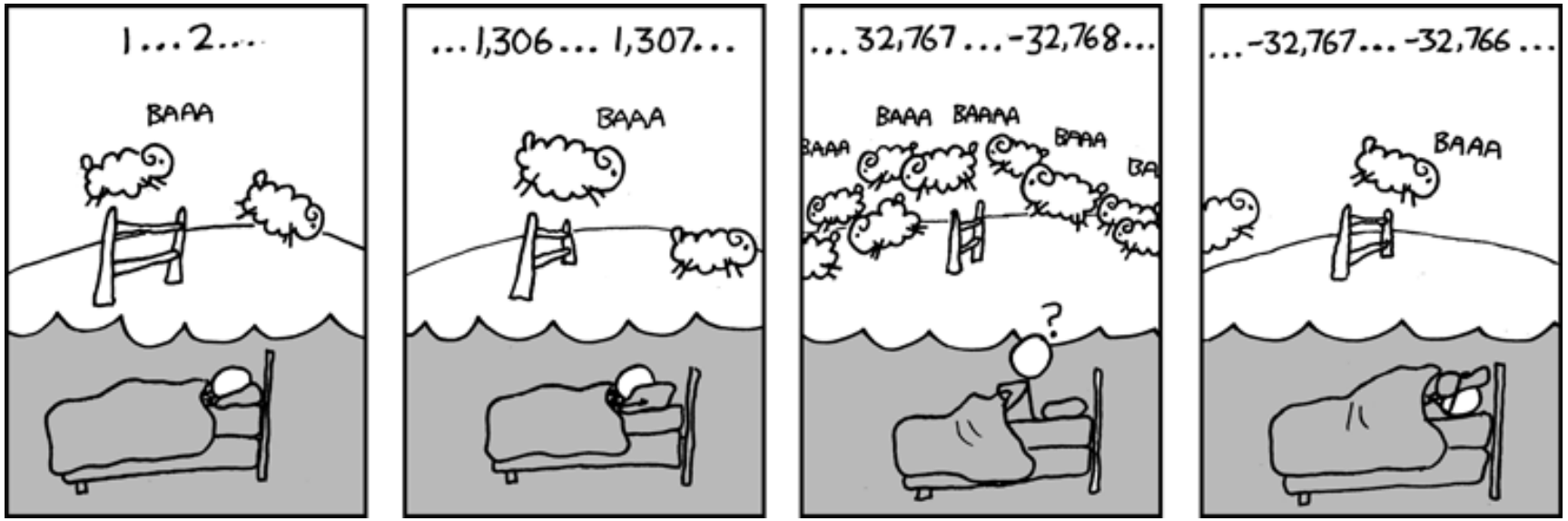
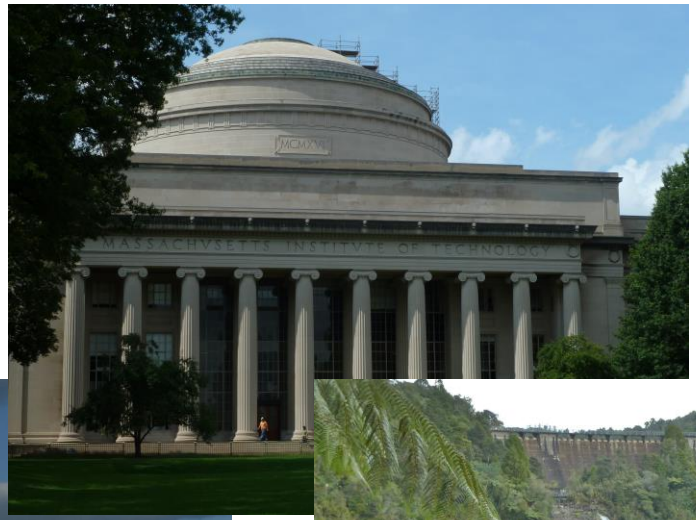
ENGSCI 233 Lecture 9.1

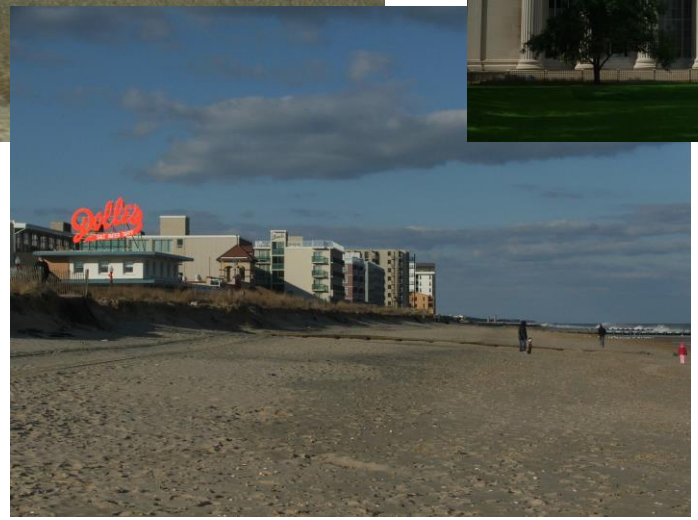
Numerical Representations



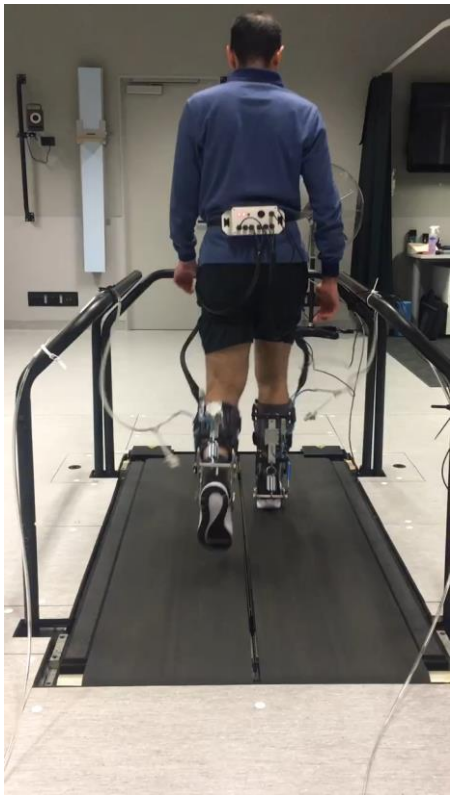
Welcome to Computer Systems

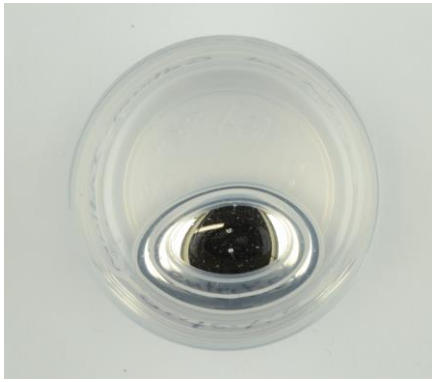
Who am I?





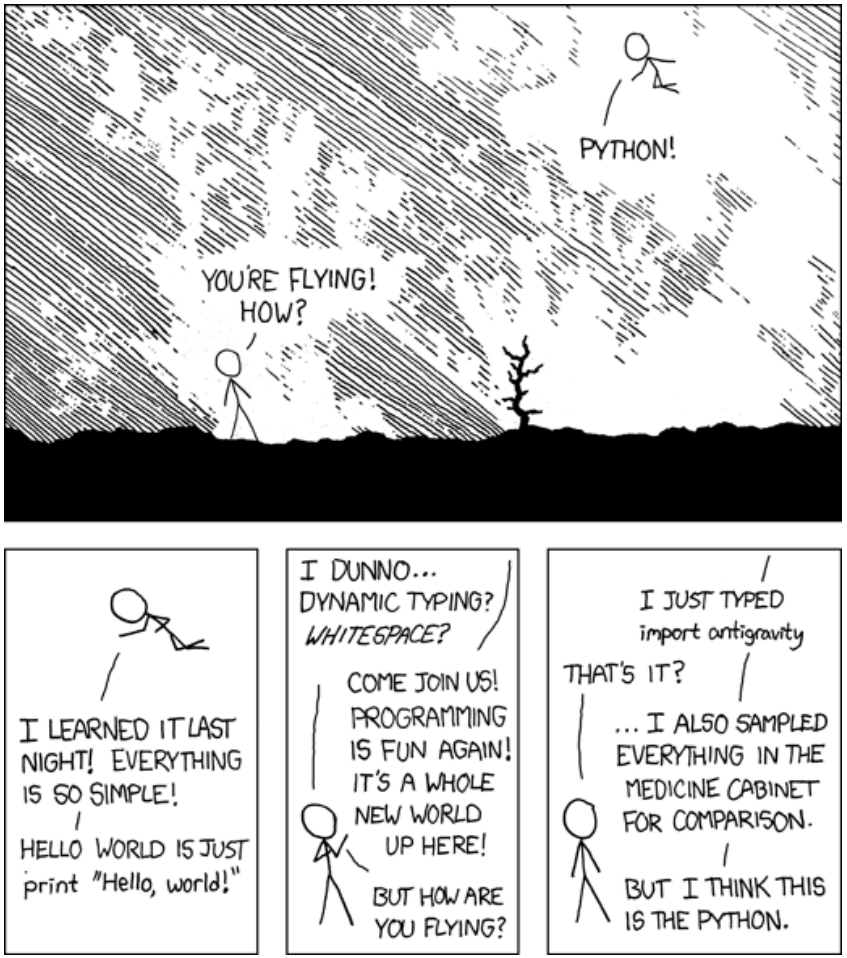
Who am I?





What does your code run on?

How can we avoid worrying about it?



Today's objectives

- Understand and convert between binary, hexadecimal, and decimal representations of integers
- Understand the impact of fixed-size integer representations
- Understand binary and hexadecimal representations of negative integers
- Understand the structure of binary floating-point representation

What is a binary number?

- Base 2
  - Digits can be 0 or 1
- *Bit*: a binary digit
- Write as 000000b
- Easy and reliable physical representation

How do different bases work again?

$$\overset{5\ 4\ 3\ 2\ 1\ 0}{101010}\text{b} =$$

$$2^5 + 2^3 + 2^1 =$$

$$32 + 8 + 2 =$$

$$42$$

Let’s practice…

- What is 37 in binary?
- What is 10001b in decimal?

Hexadecimal is more compact.

- Base 16, so each digit is exactly 4 bits
- Digits 0-9 and A-F
- Write as 0xFFFF

$$0x0 = 0000b = 0 \qquad 0x8 = 1000b = 8$$
$$0x1 = 0001b = 1 \qquad 0x9 = 1001b = 9$$
$$0x2 = 0010b = 2 \qquad 0xA = 1010b = 10$$
$$0x3 = 0011b = 3 \qquad 0xB = 1011b = 11$$
$$0x4 = 0100b = 4 \qquad 0xC = 1100b = 12$$
$$0x5 = 0101b = 5 \qquad 0xD = 1101b = 13$$
$$0x6 = 0110b = 6 \qquad 0xE = 1110b = 14$$
$$0x7 = 0111b = 7 \qquad 0xF = 1111b = 15$$

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

Let’s practice…

- What is 37 in hexadecimal?
- What is 0xAB in decimal? In binary?

How do computers use this?

- Different computers have different preferred sizes of number
- Smaller numbers are stored inefficiently
- Larger numbers are harder to work with

- 00000000b or 0x00
  - 8 bits, or 1 byte
- 0x0000
  - 16 bits (2 bytes)
- 0x00000000
  - 32 bits (4 bytes)
- 0x0000000000000000
  - 64 bits (8 bytes)

How do computers use this?

8-bit

32-bit

16-bit

64-bit

How do computers use this?

64-bit:

32-bit:

8-bit:

What if we want a negative number?

- We need a *sign bit*
- Most common form is called *2's complement*:
  - Highest bit has negative value
  - All others normal
- 2's complement lets the same process be used for arithmetic on signed and unsigned integers

- How do we represent -1 as an 8-bit signed integer?

What's the catch?

- What happens if you add one to the largest value?
- What happens if you only have part of a number?
- Have to know whether a value is meant to be signed!

- What is the largest 8-bit signed integer?
- What is the smallest?

How else can we be negative?

- *Sign-magnitude* has a literal sign bit.
  - Positive and negative zero can be confusing
- *Offset binary* subtracts a constant
  - Have to know the constant
  - Also called *biased*

- How do we represent -1 as an 8-bit signed integer?

How do you store non-integers?

- Fixed-point
  - Multiply an integer by a constant power of 2 (positive or negative)
  - No standard format
  - Exact
- Floating-point
  - Store an integer and a variable power of 2 to multiply it by
  - Standard format
  - Lossy

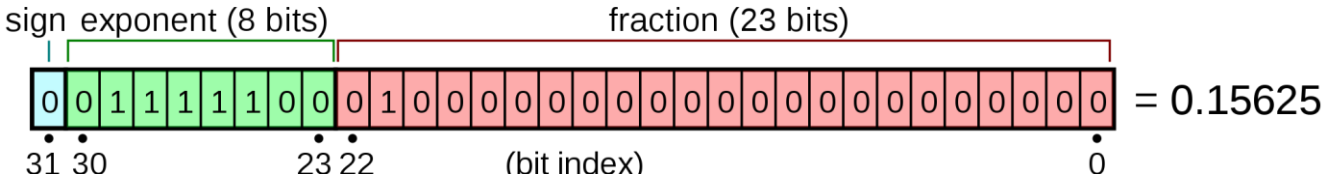
How does fixed-point work?

- Example: 4.4 format, 0x5A (01011010b)

- The decimal point location is not stored
- Addition/subtraction follow standard rules
- Multiplication requires rescaling
- Can have underflow and overflow
- Common in control systems

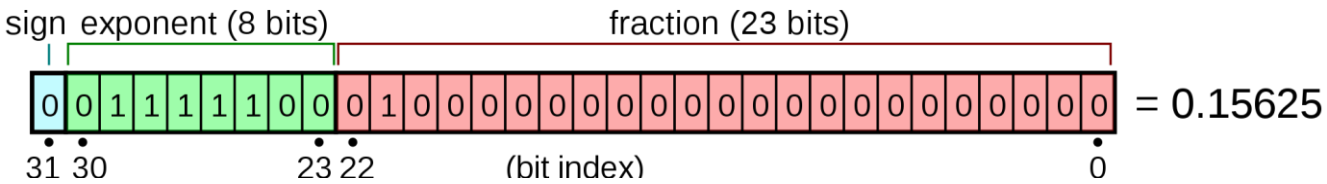
How is floating point represented?

- An international standard, IEEE 754, governs floating point representations.
- Single-precision (4-byte):



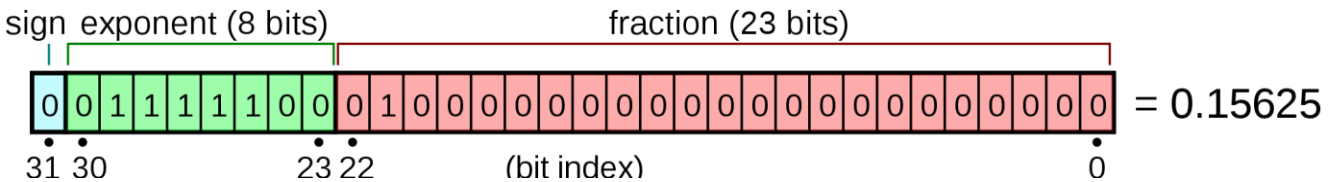
- Double-precision (8-byte):
  - 11 bit exponent, 52-bit significand fraction
- Extended precision (10-byte) – only on PCs
- Quadruple precision (16-byte) – only in IBM supercomputers

How is floating point represented?



- Exponent is *biased*: mid-range is zero
  - 127 for single-precision
- First digit of significand is assumed 1
  - Fractional part gives digits after the leading 1
- The exponent can have special values:
  - All zeroes – makes the first digit of significand zero
    - Actual exponent is still the same as given by “1”
    - Called *denormalized* if significand is nonzero
  - All ones – infinity if significand zero, NaN otherwise

How is floating point represented?



Next time: More complex data

Image References

Slide 1: *Can't Sleep*, by Randall Munroe, from https://xkcd.com/571/ (CC BY-NC 2.5)
Slide 4 (left): Video courtesy of Mahsa Momtahan
Slide 4 (centre left): Image courtesy of Soroosh Haji Hossennejad
Slides 5 (top left) and 15 (left): by LG전자, from https://commons.wikimedia.org/wiki/File:Inductive_charging_of_LG_smartphone_(2).jpg (CC BY 2.0)
Slides 5 (top centre) and 15 (centre): by Raimond Spekking, from https://commons.wikimedia.org/wiki/File:Lenovo_G500s_laptop-2905.jpg (CC BY-SA 4.0)
Slides 5 (top right) and 15 (top right): by Gareth Halfacree, from https://www.flickr.com/photos/120586634@N05/26212930836 (CC BY-SA 2.0)
Slide 5 (bottom): from https://www.nesi.org.nz/services/high-performance-computing/platforms (UoA copyright)
Slide 6: *Python*, by Randall Munroe, from https://xkcd.com/353/ (CC BY-NC 2.5)
Slide 8: by Christiaan Colen, from https://www.flickr.com/photos/christiaancolen/20607150556 (CC BY-SA 2.0)
Slide 11: by Mwtoews, from https://commons.wikimedia.org/wiki/File:Wikipedia_favicon_hexdump.svg (CC BY-SA 3.0)
Slide 14 (top left): by Evan-Amos, from https://commons.wikimedia.org/wiki/File:NES-Console-Set.jpg (Public domain)
Slide 14 (top right): by Evan-Amos, from https://commons.wikimedia.org/wiki/File:PSX-Console-wController.jpg (Public domain)
Slide 14 (bottom left): by Evan-Amos, from https://commons.wikimedia.org/wiki/File:SNES-Mod1-Console-Set.jpg (CC BY-SA 3.0)
Slide 14 (bottom right): by Evan-Amos, from https://commons.wikimedia.org/wiki/File:Nintendo-64-wController-L.jpg (Public domain)
Slide 15 (bottom right): by Snootlab, from https://www.flickr.com/photos/snootlab/6052455554 (CC BY 2.0)
Slides 21-23: by Stannerd, from https://commons.wikimedia.org/wiki/File:Float_example.svg (CC BY-SA 3.0)