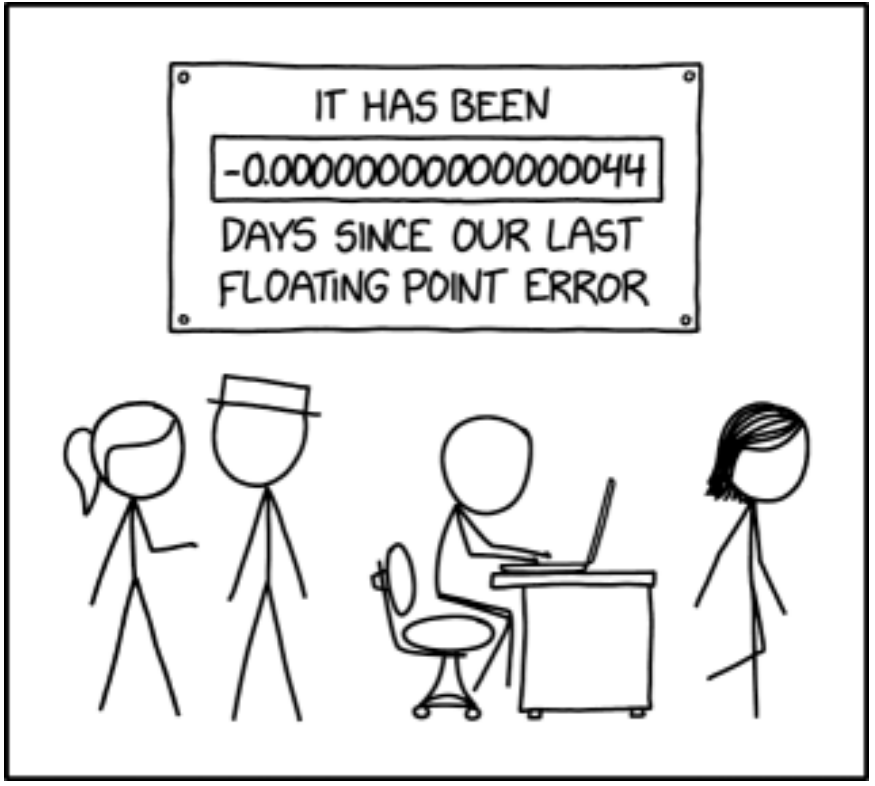
CSCI 2330


IT HAS BEEN
-0.00000000000000044
DAYS SINCE OUR LAST
FLOATING POINT ERROR

TODO

- Assignment 0 is assigned (due September 11)
- Assignment 1 is assigned (due September 24)

Overview for today

- Lab 1 prep
- Introduction to floating point

Mini bit puzzles (assume 32-bit int)

- Write an expression that evaluates to -1
- Allowed constants: 0 through 255
- Allowed operators:
  `+  &  |  ~  ^  <<  >>`
- Notably, no - allowed, so no -1
- ???

Mini bit puzzles (assume 32-bit int)

- Write an expression that evaluates to -1
- Allowed constants: 0 through 255
- Allowed operators:
  `+  &  |  ~  ^  <<  >>`
- Notably, no - allowed, so no -1
- `~0`

Mini bit puzzles (assume 32-bit int)

- Write an expression that evaluates to INT_MIN a.k.a. T_{min}
- Allowed constants: 0 through 255
- Allowed operators:
  `+  &  |  ~  ^  <<  >>`
- Notably, still no -
- ???

Mini bit puzzles (assume 32-bit int)

- Write an expression that evaluates to INT_MIN a.k.a. T_{min}
- Allowed constants: 0 through 255
- Allowed operators:
  `+  &  |  ~  ^  <<  >>`
- Notably, still no -
- `1 << 31`

Mini bit puzzles (assume 32-bit int)

- `int x = ...;`
- Write an expression that keeps only the lowest 4 bits of x and zeros out the others
- Allowed constants: 0 through 255
- Allowed operators:
  `+  &  |  ~  ^  <<  >>`
- `???`

Mini bit puzzles (assume 32-bit int)

- `int x = ...;`
- Write an expression that keeps only the lowest 4 bits of x and zeros out the others
- Allowed constants: 0 through 255
- Allowed operators:
  `+  &  |  ~  ^  <<  >>`
- `x & 0xf // 0xf is called an “and mask”`
  `// and we are using it to “mask” x`

Mini bit puzzles (assume 32-bit int)

- `int x = 0b???…???XXXX;`
- `int y = 0b???…???YYYY;`
- Write an expression that returns `0b000…000XXXXYYYY`
- Allowed operators:
  `+  &  |  ~  ^  <<  >>`
- `???`

Mini bit puzzles (assume 32-bit int)

- `int x = 0b???…???XXXX;`
- `int y = 0b???…???YYYY;`
- Write an expression that returns `0b000…000XXXXYYYY`
- Allowed operators:
  `+  &  |  ~  ^  <<  >>`
- `((x & 0xf) << 4) | (y & 0xf)`

Bit Puzzle Exercises

Lab 1

- Lab 1 is more of these puzzles
- Some are of comparable difficulty
- Most are even more difficult
- We have not covered the material for some of the floating point puzzles
- But you can still start work on all of the others!

Lab 1

- The puzzles become progressively more difficult
- Do not assume that the ones at the end will be as easy as the ones in the beginning
- Please avail yourself of office and LA hours
- Consider working in a group!

Lab 1

- MSB/LSB: most/least significant bit
  - 00000101 (5)
  - MSB synonyms: high order bit, sign bit (signed #s)
- T_{min}: smallest two's complement # (`INT_MIN`)
- T_{max}: largest two's complement # (`INT_MAX`)
- U_{min}: smallest unsigned # (always zero)
- U_{max}: largest unsigned # (`UINT_MAX`)

Structure of course

- First third (Exam 1):
  - Data Representation ← here
  - Memory
- Second third (Exam 2):
  - Machine Code
  - Memory hierarchy
- Final third (Final exam):
  - Processes
  - Other miscellaneous topics

Data Representation roadmap

- ~~Integers~~
- Floating point numbers ← here
- Strings

Floating point numbers

- Bad news: not two’s complement
- Dedicated part of the CPU to handle them (some CPUs cannot)

Decimal points

- 837.42 → what does this mean???

Decimal points

- 837.42 →
- $8 * 10^{2} + 3 * 10^{1} + 7 * 10^{0} + 4 * 10^{-1} + 2 * 10^{-2}$

Binary points

- 1101.1010 → ???

Binary points

- 1101.1010 →
- $1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3}$

Binary points

- 1101.1010 →
- $1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3}$

Binary points

- 1101.1010 →
- $2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-3}$

Fixed point

- Example: 8 bits, 4 on the left, 4 on the right
- 1011.1010

Fixed point

- Example: 8 bits, 6 on the left, 2 on the right
- 001011.10

Fixed point

- Example: 8 bits, 2 on the left, 6 on the right
- 11.101011

Fixed point

- What is the tradeoff exactly?
- More bits to the left of the binary point gets us what?
- More bits to the right of the binary point gets us what?

Fixed point

- What is the tradeoff exactly?
- More bits to the left of the binary point gets us what? Larger numbers
- More bits to the right of the binary point gets us what? More precision

Fixed point

- Problem:
- You have to ***fix*** the point’s position when you are designing the hardware

Problem

- Sometimes we need to work with very large numbers or very small numbers
- How can our system be flexible enough to support both?

How do we normally do this?

- In *science*, if we have numbers like…
- 437,000,000,000,000
- or
- 0.0000000000859
- Is there a kind of *notation* for this?

Scientific notation

- Written in scientific notation:
- $4.37 * 10^{14}$
- or
- $8.59 * 10^{-11}$

Scientific notation

- Written in scientific notation:
- $4.37 * 10^{14}$
- or
- $8.59 * 10^{-11}$
- (The 10 is the same every time)

Scientific notation

- Written in scientific notation:
- $4.37 * 10^{14}$
- or
- $8.59 * 10^{-11}$
- The digit before the decimal point is always >= 1 and < 10 (exclude exactly 0 for now)

Floating point

- Written in binary:
- 1.10001 * 2^{48}
- or
- 1.01111 * 2^{-34}
- The bit part before the decimal point is always >= 1 and < 2 (exclude exactly 0 for now)

Floating point

- Written in binary:
- $1.10001 * 2^{48}$
- or
- $1.01111 * 2^{-34}$
- All we need to encode in binary (plus the sign)

IEEE 754 floating point

- *Floating* point standard used across many machines
- Released in 1985
- 70 page document! 🥲
- Let’s cover the basics…

Two basic types

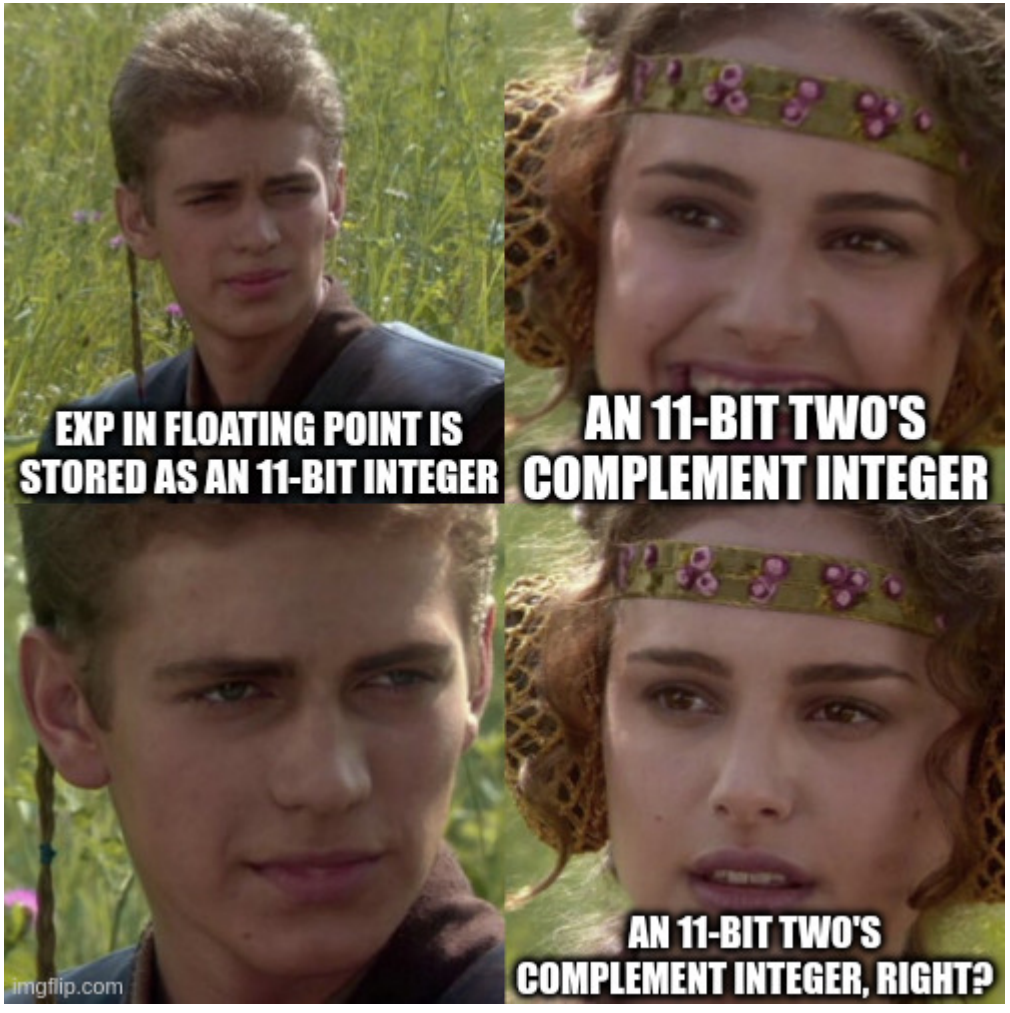
- float: “single precision”: 32 bits
- double: “double precision”: 64 bits

Terms

- $(-1)^s * M * 2^E = (-1)^s * 1.\text{fraction} * 2^E$
- s is sign bit
- M is mantissa
- M = 1.fraction
- E is exponent

But there's a catch

- How do we encode s, fraction, E in binary?
- s is straightforward…
  - 0 is 0, 1 is 1
- frac is also straightforward…
  - 1011 means mantissa is 1.1011 = $2^0 + 2^{-1} + 2^{-3} + 2^{-4}$
- But E…


EXP IN FLOATING POINT IS STORED AS AN 11-BIT INTEGER
AN 11-BIT TWO'S COMPLEMENT INTEGER
AN 11-BIT TWO'S COMPLEMENT INTEGER, RIGHT?
imgflip.com

Interpretation of exp

- $(-1)^s * M * 2^E = (-1)^s * 1.\text{fraction} * 2^{\text{exp} - 2^{k-1} + 1}$
- s is sign bit
- M is mantissa (1.fraction)
- E = exp (as an unsigned number) - bias
- bias = $2^{k-1} - 1$, where *k* is # of bits of exp

len: 1 **k** = 8 or 11 23 or 52

How do we encode 1 in 32-bit float?

- $(-1)^s * M * 2^E = (-1)^s * 1.\text{fraction} * 2^{\text{exp} - 2^{k-1} + 1}$

How do we encode 1 in 32-bit float?

- $(-1)^s * M * 2^E = (-1)^s * 1.\text{fraction} * 2^{\text{exp} - 2^{k-1} + 1}$
- Want s = 0
- E = 0
- M = 1.00000…
- $(-1)^0 * 1.00000\ldots * 2^0 = 1$

- We want s = 0, so set s = 0
- We want M = 1.00000…
  - Since M = 1.frac, set frac = 00000…
- We want E = 0… what is exp?

len: 1 **k** = 8 or 11 23 or 52

- How do we encode E = 0 in a 32-bit float?
- E = exp - bias
- => 0 = exp - 2^{k-1} + 1
- => 0 = exp - 2^{7} + 1
- => exp = 127
- exp is an unsigned integer, so encode 127 as:
- 01111111

len: 1 **k** = 8 or 11 23 or 52

- Combining everything, 1.0 is encoded as:
- 0 01111111 00000000000000000000000

len: 1 **k** = 8 or 11 23 or 52

- Combining everything, 1.0 is encoded as:
- 0 01111111 00000000000000000000000
- What would we have to change to encode 2.0?
- $(-1)^s * M * 2^E = (-1)^s * 1.\text{fraction} * 2^{\text{exp} - 2^{k-1} + 1}$

- Combining everything, 1.0 is encoded as:
- 0 01111111 00000000000000000000000
- What would we have to change to encode 2.0?
- $(-1)^s * M * 2^E = (-1)^s * 1.\text{fraction} * 2^{\text{exp} - 2^{k-1} + 1}$
- Combining everything, 2.0 is encoded as:
- 0 10000000 00000000000000000000000

TODO

- Assignment 0 is assigned (due September 11)
- Assignment 1 is assigned (due September 24)