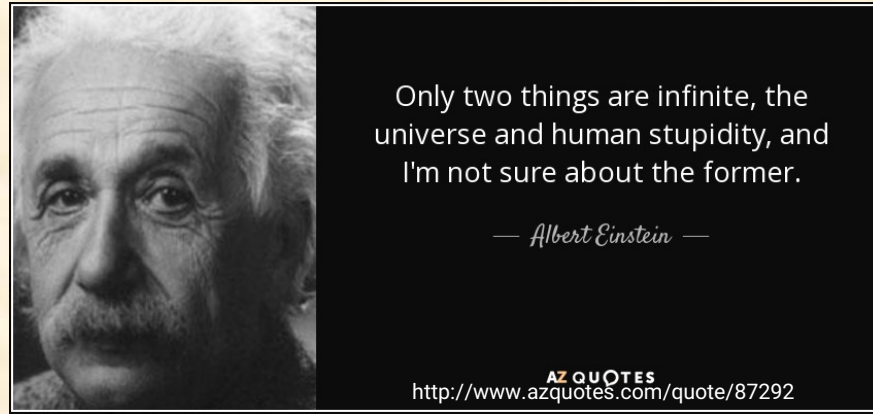


# 18-642: Floating Point Blues

10/18/2018



Carnegie  
Mellon  
University

# Floating Point Math

## ■ Anti-Patterns:

- **Not accounting for roundoff errors**
  - Tests for floating point equality
- **Not handling special values**
- **Float used if integer does the job**
  - **Not always good for “big” numbers**

## ■ Floating Point Math:

- Exponent + Mantissa representation
  - 32-bit, 64-bit, others on some systems
- Roundoff errors due to finite number of mantissa bits
- Special values:  
Infinity, Not A Number (NaN), denorms, signed zero

IEEE Floating Point Format  
Single Precision: 32 bits total



Value = (+/-) 1.Mantissa \*  $2^{(\text{Exponent}-127)}$

**Sign:** 0=positive; 1=negative

**Exponent:** 127 bias, radix 2  
value is EXPONENT - 127

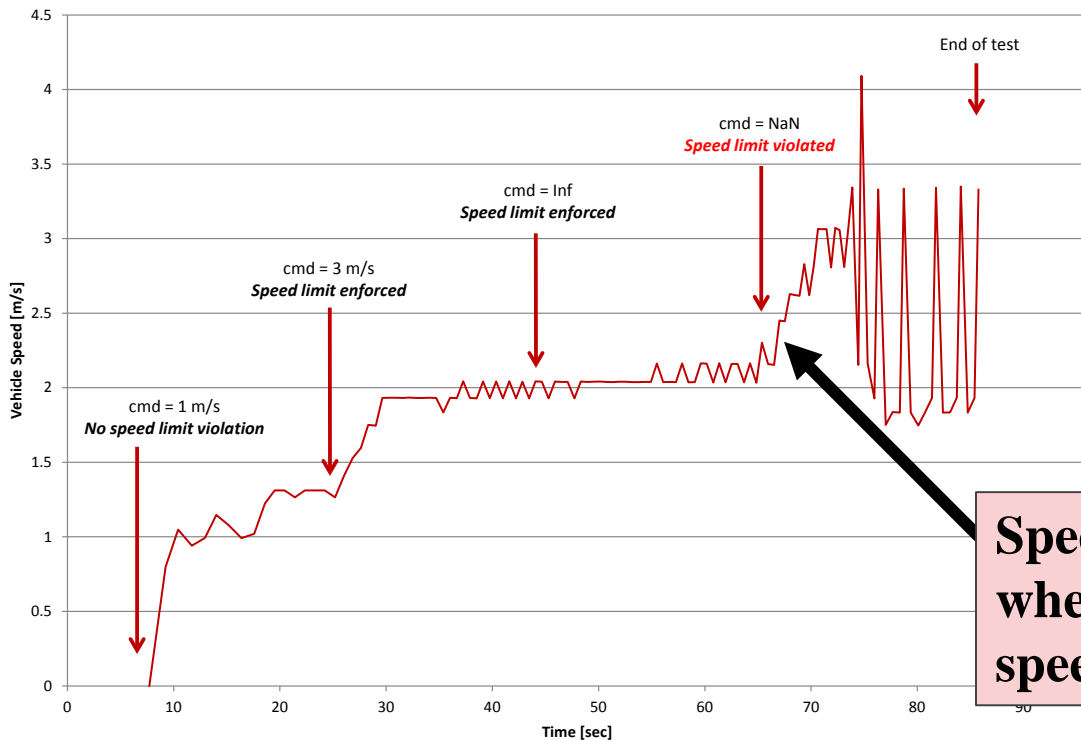
**Mantissa:** implicit 1.  
value is 1.MANTISSA (binary)

**Special zero value:**  
zero = 0x00000000



# NaN and the Robot Apocalypse

## RECbot Speed Limit Tests



**Speed-limit violation occurred when exceptional input sent as speed command**

# Roundoff Errors

Iteratively Add 0.1	
iteration 1	value=0.100000
iteration 2	value=0.200000
iteration 3	value=0.300000
iteration 4	value=0.400000
iteration 5	value=0.500000
iteration 6	value=0.600000
iteration 7	value=0.700000
iteration 8	value=0.800000
iteration 9	value=0.900000
iteration 10	value=1.000000
iteration 11	value=1.100000
iteration 12	value=1.200000
iteration 13	value=1.300000
iteration 14	value=1.400000
iteration 15	value=1.500000
iteration 16	value=1.600000
iteration 17	value=1.700000
iteration 18	value=1.800000
iteration 19	value=1.900000
iteration 20	value=2.000000
iteration 21	value=2.100000
iteration 22	value=2.200000
iteration 23	value=2.300000
iteration 24	value=2.400000
iteration 25	value=2.500000
iteration 26	value=2.600000
iteration 27	value=2.700000
iteration 28	value=2.799999
iteration 29	value=2.899999
iteration 30	value=2.999999
iteration 31	value=3.099999
iteration 32	value=3.199999
iteration 33	value=3.299999
iteration 34	value=3.399999
iteration 35	value=3.499999

## ■ Rounding error due to limited bits

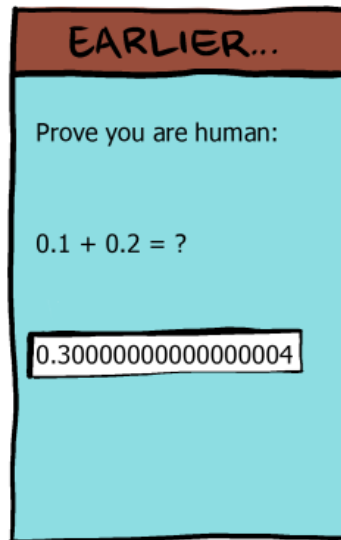
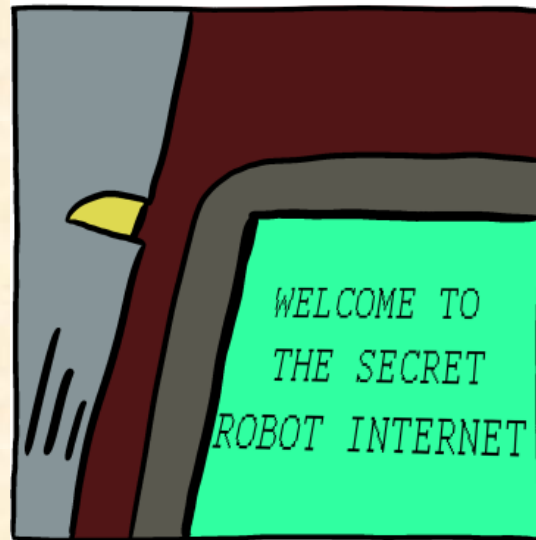
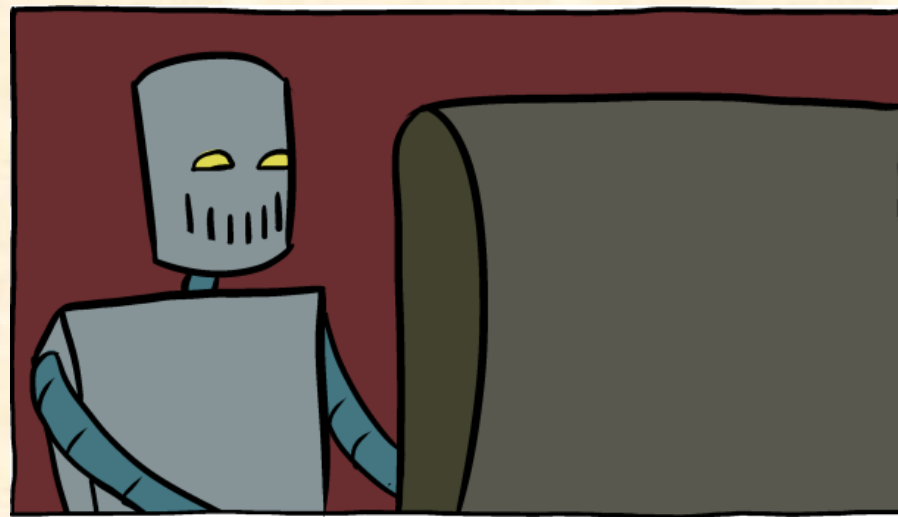
- Mantissa: 24 bits (implicit leading one)
  - E.g.: all zero mantissa bits  $\rightarrow 1.000000000000000000000000_2$
- More than 24 bits of value won't fit
  - Converting int to float to int to float in a chain gives:  
`0x72345673`  $\rightarrow$  1916032640.0  $\rightarrow$  `0x72345680`  $\rightarrow$  1916032640.0

## ■ Rounding error due to imprecise representation

- IEEE 754 is radix 2, so decimal fractions can be inexact
  - Repeatedly add 0.1 to a 32-bit float and you get....  
0.1, 0.2, ..., 2.799999, ..., 49.999809, ..., 99.999046

## ■ Floating point comparison pitfall:

- if (`fa == fb`) might not match due to rounding error
  - Use a comparison interval, e.g.,:  
if (`fabs(result - expectedResult) < 0.00001`)



# Don't Use Floats for Time!

## ■ Patriot Missile mishap

- 1991: Scud kills 28 American (Desert Storm)
- <http://www.fas.org/spp/starwars/gao/im92026.htm>  
“after about 20 hours, the inaccurate time calculation becomes sufficiently large to cause the radar to look in the wrong place”
  - “Range gate” used to look where target is predicted to be next
  - Target track is lost if range gate is wrong, resulting in a miss
  - The incident happened 100 hours after the last system reset

## ■ What was the root cause?

- Patriot designed for aircraft and frequent mobile relocations
  - Scud missiles travel at Mach 5 (3750 mph); Patriot deployed in fixed location
- Even a small round-off error matters when computing distance = velocity \* time
  - Large accumulated base time and high velocity leads to a failure







# Best Practices for Floating Point

## ■ Use integer math if you can

- Scaled integer (e.g., 10ths of a second)
- Binary Coded Decimal (BCD) + radix point
- Fixed point (e.g., value \*256)

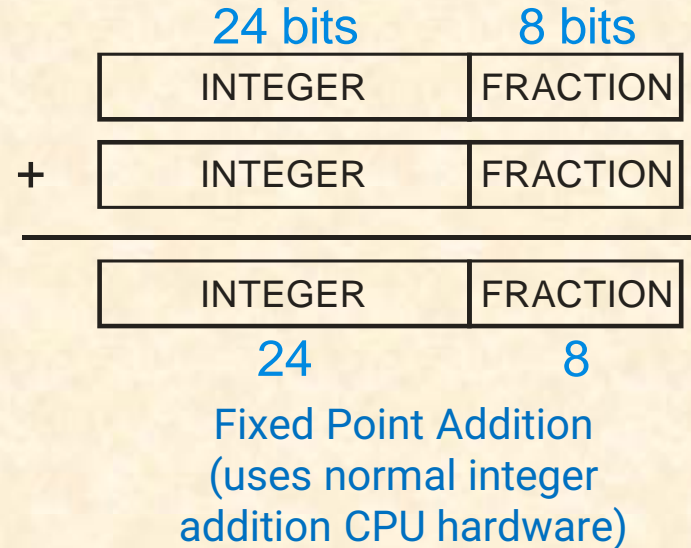
## ■ Handle special values

- NaN is especially tricky to get right

## ■ Manage and handle roundoff error

- Doubles give more bits to work with (53-bit mantissa)
  - But fundamentally, all problems are still there
- Don't use floating point as an iterator, including time!

## ■ Comparisons are especially problematic (NaN, roundoff)



HEY, CHECK IT OUT:  $e^\pi - \pi$  IS 19.999099979. THAT'S WEIRD.

YEAH. THAT'S HOW I GOT KICKED OUT OF THE ACM IN COLLEGE.

... WHAT?



DURING A COMPETITION, I TOLD THE PROGRAMMERS ON OUR TEAM THAT  $e^\pi - \pi$  WAS A STANDARD TEST OF FLOATING-POINT HANDLERS -- IT WOULD COME OUT TO 20 UNLESS THEY HAD ROUNDING ERRORS.



THAT'S AWFUL.

YEAH, THEY DUG THROUGH HALF THEIR ALGORITHMS LOOKING FOR THE BUG BEFORE THEY FIGURED IT OUT.



<https://xkcd.com/217/>