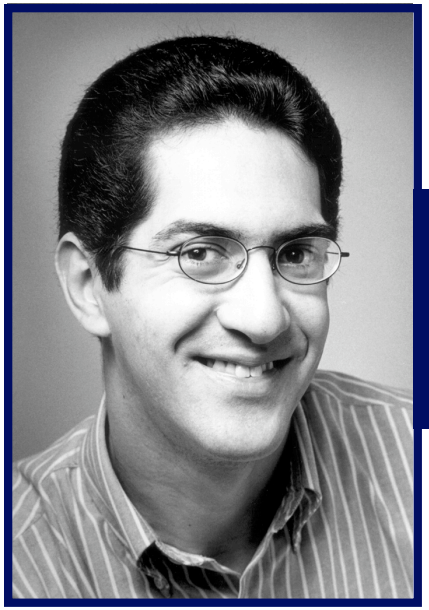
inst.eecs.berkeley.edu/~cs61c

# CS61C : Machine Structures

## Lecture 15 – Floating Point I

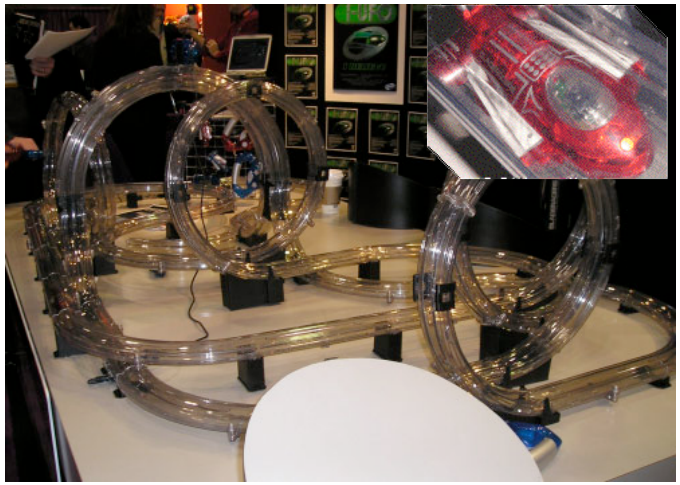
**2004-02-25**



**Lecturer PSOE Dan Garcia**

www.cs.berkeley.edu/~ddgarcia

**What is the ⇒ hottest toy for the 2004 holidays? RoboSapien? Tellmi? Blade Racers?**





Garcia, Spring 2004 © UCB

# The Flybar!! (Segway eat your heart out)

## Applications?

- Fun for seriously jaded snowboarders during the offseason
- ESPN X-Games
- Another source of painful content for RealTV
- And the “Killer” app?
- Prison breaks!
  - No walls can hold ‘em!



Garcia, Spring 2004 © UCB

# Quote of the day

"95% of the folks out there are completely clueless about floating-point."

James Gosling
Sun Fellow
Java Inventor
1998-02-28

Garcia, Spring 2004 © UCB

# Review of Numbers

- Computers are made to deal with numbers
- What can we represent in N bits?
  - Unsigned integers:

    $0$ to $2^N - 1$

  - Signed Integers (Two's Complement)

    $-2^{(N-1)}$ to $2^{(N-1)} - 1$

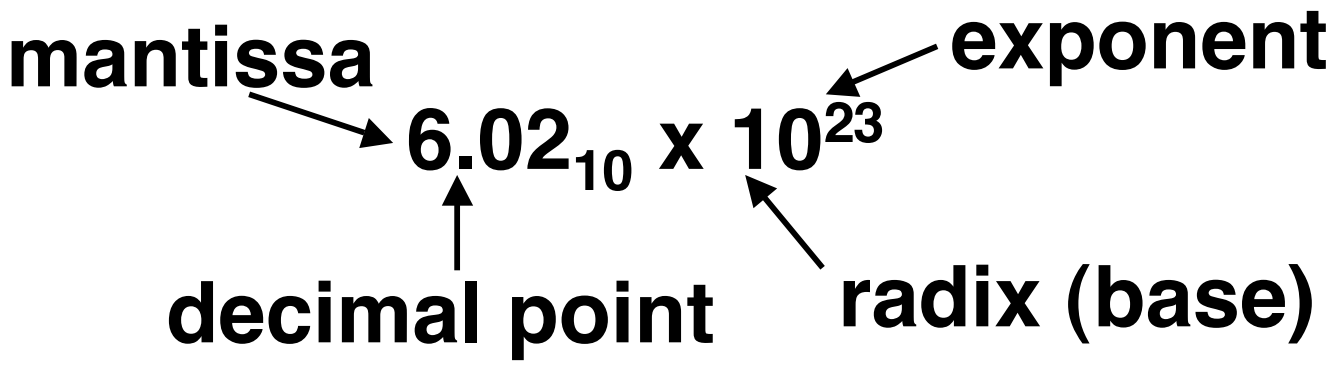
Garcia, Spring 2004 © UCB

# Other Numbers

- What about other numbers?
  - Very large numbers? (seconds/century)
    $3{,}155{,}760{,}000_{10}$ ($3.15576_{10} \times 10^{9}$)
  - Very small numbers? (atomic diameter)
    $0.00000001_{10}$ ($1.0_{10} \times 10^{-8}$)
  - Rationals (repeating pattern)
    2/3 (0.666666666. . .)
  - Irrationals
    $2^{1/2}$ (1.414213562373. . .)
  - Transcendentals
    *e* (2.718...), π (3.141...)
- All represented in scientific notation

Garcia, Spring 2004 © UCB

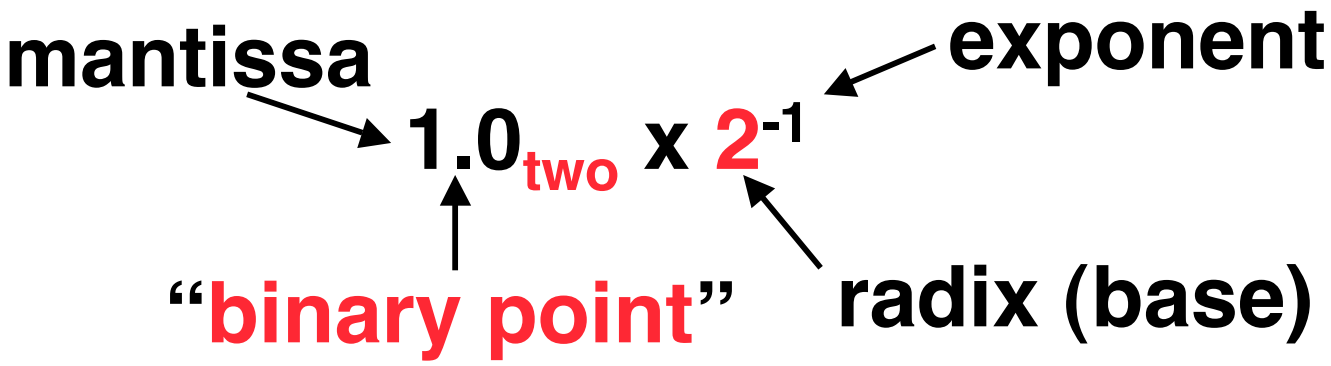
# Scientific Notation (in Decimal)




- Normalized form: no leadings 0s (exactly one digit to left of decimal point)
- Alternatives to representing 1/1,000,000,000
  - Normalized: $1.0 \times 10^{-9}$
  - Not normalized: $0.1 \times 10^{-8}$,$10.0 \times 10^{-10}$

Garcia, Spring 2004 © UCB

# Scientific Notation (in Binary)

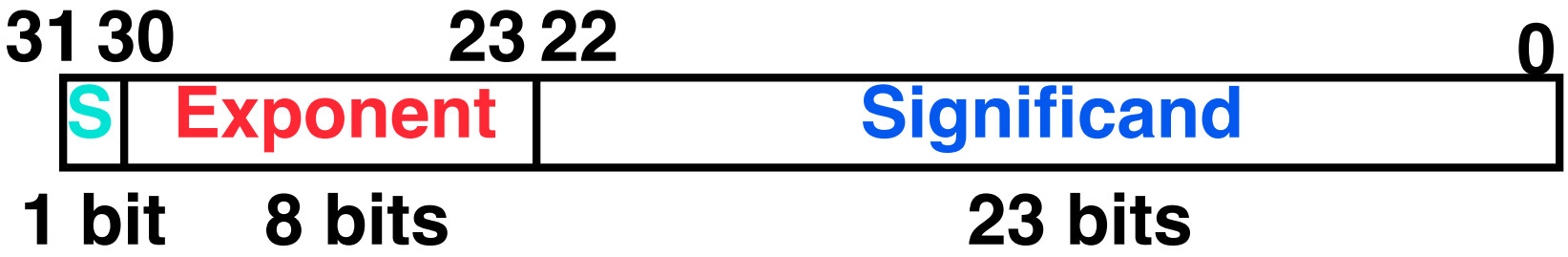



- Computer arithmetic that supports it called floating point, because it represents numbers where binary point is not fixed, as it is for integers
  - Declare such variable in C as `float`

Garcia, Spring 2004 © UCB

# Floating Point Representation (1/2)

- Normal format: $+1.xxxxxxxxxx_{two} * 2^{yyyy}{}_{two}$
- Multiple of Word Size (32 bits)




- S represents Sign
  Exponent represents y's
  Significand represents x's
- Represent numbers as small as $2.0 \times 10^{-38}$ to as large as $2.0 \times 10^{38}$

Garcia, Spring 2004 © UCB

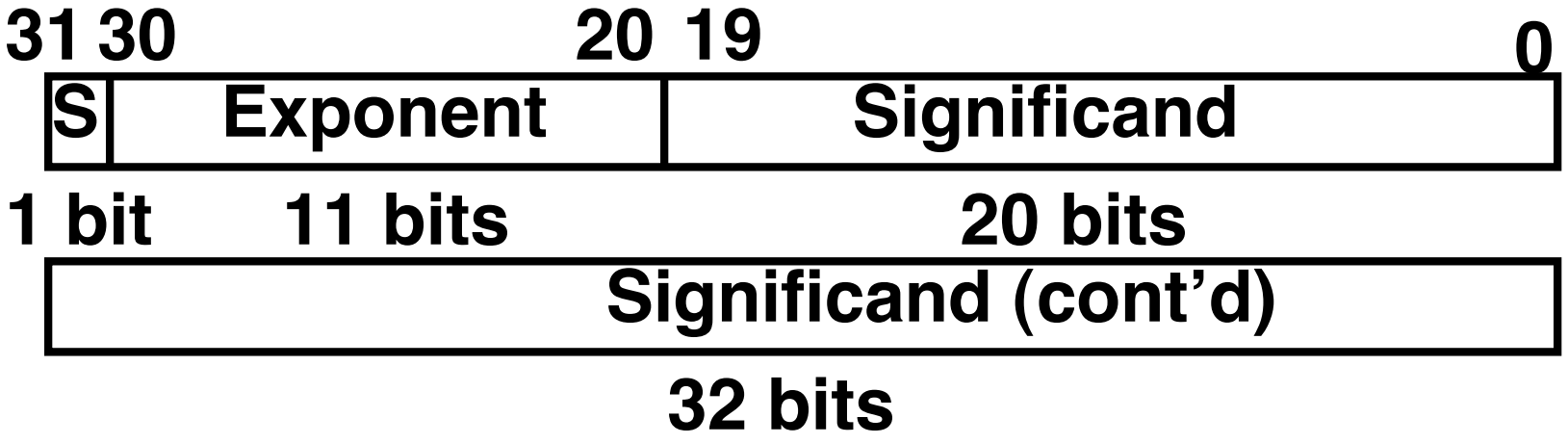
# Floating Point Representation (2/2)

- What if result too large? (> $2.0 \times 10^{38}$ )
  - Overflow!
  - Overflow ⇒ Exponent larger than represented in 8-bit Exponent field
- What if result too small? (>0, < $2.0 \times 10^{-38}$ )
  - Underflow!
  - Underflow ⇒ Negative exponent larger than represented in 8-bit Exponent field
- How to reduce chances of overflow or underflow?

Garcia, Spring 2004 © UCB

# Double Precision Fl. Pt. Representation

- Next Multiple of Word Size (64 bits)




- Double Precision (vs. Single Precision)
  - C variable declared as `double`
  - Represent numbers almost as small as $2.0 \times 10^{-308}$ to almost as large as $2.0 \times 10^{308}$
  - But primary advantage is greater accuracy due to larger significand

Garcia, Spring 2004 © UCB

# QUAD Precision Fl. Pt. Representation

- Next Multiple of Word Size (128 bits)
- Unbelievable range of numbers
- Unbelievable precision (accuracy)
- This is currently being worked on

Garcia, Spring 2004 © UCB

# IEEE 754 Floating Point Standard (1/4)

- Single Precision, DP similar
- Sign bit: 1 means negative
  0 means positive
- Significand:
  - To pack more bits, leading 1 implicit for normalized numbers
  - 1 + 23 bits single, 1 + 52 bits double
  - always true: Significand < 1
    (for normalized numbers)
- Note: 0 has no leading 1, so reserve exponent value 0 just for number 0

Garcia, Spring 2004 © UCB

# IEEE 754 Floating Point Standard (2/4)

- Kahan wanted FP numbers to be used even if no FP hardware; e.g., sort records with FP numbers using integer compares
- Could break FP number into 3 parts: compare signs, then compare exponents, then compare significands
- Wanted it to be faster, single compare if possible, especially if positive numbers
- Then want order:
  - Highest order bit is sign ( negative < positive)
  - Exponent next, so big exponent => bigger #
  - Significand last: exponents same => bigger #

Garcia, Spring 2004 © UCB

# IEEE 754 Floating Point Standard (3/4)

- **Negative Exponent?**
  - 2's comp? $1.0 \times 2^{-1}$ v. $1.0 \times 2^{+1}$ (1/2 v. 2)

| | | | |
|---|---|---|---|
| 1/2 | 0 | 1111 1111 | 000 0000 0000 0000 0000 0000 |
| 2 | 0 | 0000 0001 | 000 0000 0000 0000 0000 0000 |

  - This notation using integer compare of 1/2 v. 2 makes 1/2 > 2!
- **Instead, pick notation 0000 0001 is most negative, and 1111 1111 is most positive**
  - $1.0 \times 2^{-1}$ v. $1.0 \times 2^{+1}$ (1/2 v. 2)

| | | | |
|---|---|---|---|
| 1/2 | 0 | 0111 1110 | 000 0000 0000 0000 0000 0000 |
| 2 | 0 | 1000 0000 | 000 0000 0000 0000 0000 0000 |

Garcia, Spring 2004 © UCB

# IEEE 754 Floating Point Standard (4/4)

- Called Biased Notation, where bias is number subtract to get real number
  - IEEE 754 uses bias of 127 for single prec.
  - Subtract 127 from Exponent field to get actual value for exponent
  - 1023 is bias for double precision
- Summary (single precision):

- $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$
  - Double precision identical, except with exponent bias of 1023

Garcia, Spring 2004 © UCB

# "Father" of the Floating point standard

**IEEE Standard 754 for Binary Floating-Point Arithmetic.**

**Prof. Kahan**

**1989 ACM Turing Award Winner!**

`www.cs.berkeley.edu/~wkahan/`
`…/ieee754status/754story.html`

Garcia, Spring 2004 © UCB

# Administrivia

- HW4 due today @ 23:59
- You've taken (& reviewed) your Faux exam…
  - Enter your score in lab
- Midterm 155 Dwinelle Mon 03-08 @ 7-10pm
  - Conflicts? Email Head TA Jeremy, cc Dan
  - DSP exam? Email Cindy `cindyp@cs`, cc Dan
- How should we study for the midterm?
  - Attend the review session (time/date TBA)
  - Look over HW, Labs, Projects
  - Write up your 1-page study sheet
  - Go over old exams – HKN office has put them online (link from 61C home page)

Garcia, Spring 2004 © UCB

# Upcoming Calendar

| Week # | Mon | Wed | Thurs Lab | Fri |
|---|---|---|---|---|
| **#6 This week** | Dan **MIPS Inst Format II** | Dan **Floating Pt I** | **Floating Pt** (No Dan OH) | Dan **Floating Pt II** |
| **#7 Next week** | Dan **MIPS Inst Format III** | Alexandre **Running Program** (No Dan OH) | **Running Program** (No Dan OH) | Roy **Running Program** |
| **#8 Midterm week** | Dan **Caches** **Midterm @ 7pm** | Dan **Caches** | **Caches** | Dan **Caches** |

Garcia, Spring 2004 © UCB

# Understanding the Significand (1/2)

- Method 1 (Fractions):
  - In decimal: $0.340_{10}$ => $340_{10}/1000_{10}$ => $34_{10}/100_{10}$
  - In binary: $0.110_2$ => $110_2/1000_2 = 6_{10}/8_{10}$ => $11_2/100_2 = 3_{10}/4_{10}$
  - Advantage: less purely numerical, more thought oriented; this method usually helps people understand the meaning of the significand better

Garcia, Spring 2004 © UCB

# Understanding the Significand (2/2)

- Method 2 (Place Values):
  - Convert from scientific notation
  - In decimal: $1.6732 = (1 \times 10^{0}) + (6 \times 10^{-1}) + (7 \times 10^{-2}) + (3 \times 10^{-3}) + (2 \times 10^{-4})$
  - In binary: $1.1001 = (1 \times 2^{0}) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4})$
  - Interpretation of value in each position extends beyond the decimal/binary point
  - Advantage: good for quickly calculating significand value; use this method for translating FP numbers

Garcia, Spring 2004 © UCB

# Example: Converting Binary FP to Decimal

- Sign: 0 => positive
- Exponent:
  - $0110\ 1000_{two} = 104_{ten}$
  - Bias adjustment: 104 - 127 = -23
- Significand:
  - $1 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} + 0x2^{-4} + 1x2^{-5} + \ldots$
    $= 1 + 2^{-1} + 2^{-3} + 2^{-5} + 2^{-7} + 2^{-9} + 2^{-14} + 2^{-15} + 2^{-17} + 2^{-22}$
    $= 1.0_{ten} + 0.666115_{ten}$
- Represents: $1.666115_{ten} * 2^{-23} \sim 1.986 * 10^{-7}$ (about 2/10,000,000)

Garcia, Spring 2004 © UCB

# Converting Decimal to FP (1/3)

- Simple Case: If denominator is an exponent of 2 (2, 4, 8, 16, etc.), then it's easy.
- Show MIPS representation of -0.75
  - -0.75 = -3/4
  - $-11_{two}/100_{two} = -0.11_{two}$
  - Normalized to $-1.1_{two} \times 2^{-1}$
  - $(-1)^{S} \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$
  - $(-1)^{1} \times (1 + .100\ 0000\ \ldots\ 0000) \times 2^{(126-127)}$

| 1 | 0111 1110 | 100 0000 0000 0000 0000 0000 |
|---|---|---|

Garcia, Spring 2004 © UCB

# Converting Decimal to FP (2/3)

- Not So Simple Case: If denominator is not an exponent of 2.
  - Then we can't represent number precisely, but that's why we have so many bits in significand: for precision
  - Once we have significand, normalizing a number to get the exponent is easy.
  - So how do we get the significand of a neverending number?

Garcia, Spring 2004 © UCB

# Converting Decimal to FP (3/3)

- Fact: All rational numbers have a repeating pattern when written out in decimal.
- Fact: This still applies in binary.
- To finish conversion:
  - Write out binary number with repeating pattern.
  - Cut it off after correct number of bits (different for single v. double precision).
  - Derive Sign, Exponent and Significand fields.

Garcia, Spring 2004 © UCB

# Peer Instruction

| 1 | 1000 0001 | 111 0000 0000 0000 0000 0000 |
|---|---|---|

**What is the decimal equivalent of the floating pt # above?**

```
1: -1.75
2: -3.5
3: -3.75
4: -7
5: -7.5
6: -15
7: -7 * 2^129
8: -129 * 2^7
```

Garcia, Spring 2004 © UCB

# "And in conclusion…"

- Floating Point numbers approximate values that we want to use.
- IEEE 754 Floating Point Standard is most widely accepted attempt to standardize interpretation of such numbers
  - Every desktop or server computer sold since ~1997 follows these conventions
- Summary (single precision):

- $(-1)^S \times (1 + \text{Significand}) \times 2^{(\text{Exponent}-127)}$
- Double precision identical, bias of 1023


Garcia, Spring 2004 © UCB