

# Assembly: Introduction.

Yipeng Huang

Rutgers University

February 23, 2021

# Table of contents

## Announcements

## Floats: Review

- Normalized: exp field

- Normalized: frac field

- Normalized/denormalized

- Special cases

## Floats: Mastery

- Normalized number bitstring to real number it represents

- Floating point multiplication

- Properties of floating point

## Instruction set architectures

- why are instruction set architectures important

- 8-bit vs. 16-bit. vs. 32-bit vs. 64-bit

- CISC vs. RISC

# Looking ahead

## Class plan

1. Today, Tuesday, 2/23: Bits to floats. Introduction to the software-hardware interface.
2. Reading assignment for next four weeks: CS:APP Chapter 3.
3. Thursday, 2/25: Programming Assignment 3 on bits, bytes, integers, floats out.
4. Monday, 3/1: Programming Assignment 2 due. Be sure to test on ilab, "make clean". Quiz 6 on floating point trickiness out.

# Table of contents

## Announcements

## Floats: Review

- Normalized: exp field

- Normalized: frac field

- Normalized/denormalized

- Special cases

## Floats: Mastery

- Normalized number bitstring to real number it represents

- Floating point multiplication

- Properties of floating point

## Instruction set architectures

- why are instruction set architectures important

- 8-bit vs. 16-bit. vs. 32-bit vs. 64-bit

- CISC vs. RISC

# Floating point numbers

## Avogadro's number

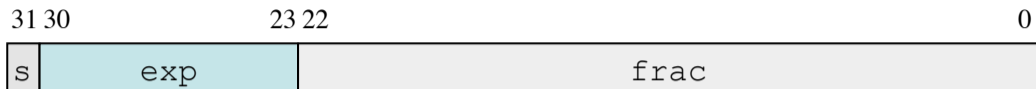
$$+6.02214 \times 10^{23} \text{ mol}^{-1}$$

## Scientific notation

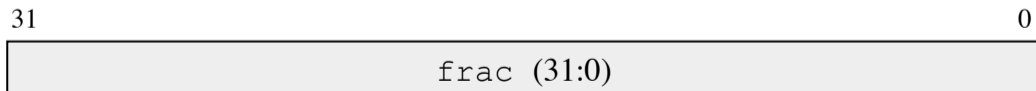
- ▶ sign
- ▶ mantissa or significand
- ▶ exponent

# Floats and doubles

Single precision



Double precision



**Figure:** The two standard formats for floating point data types. Image credit CS:APP

# Floats and doubles

property	float	double
total bits	32	64
s bit	1	1
exp bits	8	11
frac bits	23	52
C printf() format specifier	"%f"	"%lf"

Table: Properties of floats and doubles

# The IEEE 754 number line

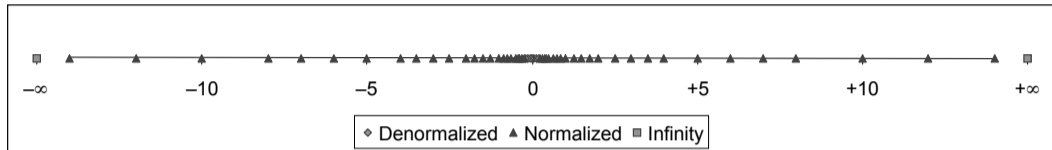


Figure: Full picture of number line for floating point values. Image credit CS:APP

$$+1.0/\text{INF} = +0; -1.0/\text{INF} = -0$$

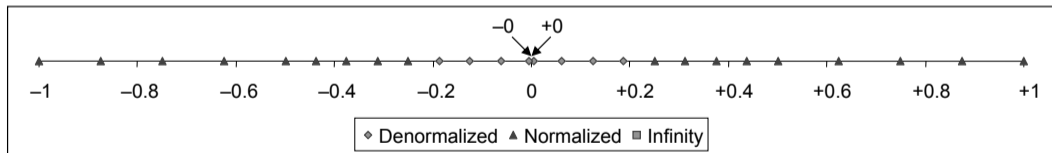


Figure: Zoomed in number line for floating point values. Image credit CS:APP

# Different cases for floating point numbers

Value of the floating point number =  $(-1)^s \times M \times 2^E$

- ▶  $E$  is encoded the exp field
- ▶  $M$  is encoded the frac field

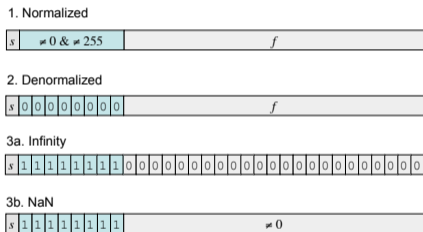


Figure: Different cases within a floating point format. Image credit CS:APP

## Normalized and denormalized numbers

Two different cases we need to consider for the encoding of  $E$ ,  $M$

## Normalized: exp field

For normalized numbers,  
 $0 < \text{exp} < 2^k - 1$

- ▶ exp is a  $k$ -bit unsigned integer

### Bias

- ▶ need a bias to represent negative exponents
- ▶  $\text{bias} = 2^{k-1} - 1$
- ▶ bias is the  $k$ -bit unsigned integer:  
011..111

For normalized numbers,  
 $E = \text{exp} - \text{bias}$

In other words,  $\text{exp} = E + \text{bias}$

property	float	double
k	8	11
bias	127	1023
smallest E (greatest precision)	-126	-1022
largest E (greatest range)	127	1023

Table: Summary of normalized exp field

Normalized: frac field

$$M = 1.\text{frac}$$

# Floats: Summary

	normalized	denormalized
value of number	$(-1)^s \times M \times 2^E$	$(-1)^s \times M \times 2^E$
E	$E = \text{exp} - \text{bias}$	$E = -\text{bias} + 1$
bias	$2^{k-1} - 1$	$2^{k-1} - 1$
exp	$0 < \text{exp} < (2^k - 1)$	$\text{exp} = 0$
M	$M = 1.\text{frac}$	$M = 0.\text{frac}$
	M has implied leading 1	M has leading 0
	greater range	greater precision
	large magnitude numbers	small magnitude numbers
	denser near origin	evenly spaced

**Table:** Summary of normalized and denormalized numbers

## Floats: Special cases

number class	when it arises	exp field	frac field
+0 / -0		0	0
+infinity / -infinity	overflow or division by 0	$2^k - 1$	0
NaN not-a-number	illegal ops. such as $\sqrt{-1}$ , inf-inf, inf*0	$2^k - 1$	non-0

Table: Summary of special cases

# Table of contents

## Announcements

## Floats: Review

- Normalized: exp field

- Normalized: frac field

- Normalized/denormalized

- Special cases

## Floats: Mastery

- Normalized number bitstring to real number it represents

- Floating point multiplication

- Properties of floating point

## Instruction set architectures

- why are instruction set architectures important

- 8-bit vs. 16-bit. vs. 32-bit vs. 64-bit

- CISC vs. RISC

# Normalized number bitstring to real number it represents

- ▶ Tiny FP: 1-bit s, 4-bit exp, 3-bit frac
- ▶ What does this encode: 1\_1001\_101
- ▶  $s=1$ , so positive? negative? **Negative number**
- ▶  $\text{exp} = 1001_2 = 8+1 = 9$
- ▶  $\text{bias} = 2^{k-1} - 1 = 2^{4-1} - 1 = 7$
- ▶  $E = \text{exp} - \text{bias} = 9 - 7 = 2$
- ▶  $\text{frac} = 101_2$
- ▶  $M = 1.\text{frac} = 1.101_2$

# How to multiply scientific notation?

Avagadro number squared?

$$\begin{aligned} (+6.02 \times 10^{23})^2 &= (+6.02 \times 10^{23}) \times (+6.02 \times 10^{23}) \\ &= (+6.02 \times 6.02) \times 10^{46} \\ &= 36.2404 \times 10^{46} \\ &= 3.62404 \times 10^{47} \end{aligned}$$

Recall:  $\log(x \times y) = \log(x) + \log(y)$

## FP Multiplication

- $(-1)^{s1} M1 2^{E1} \times (-1)^{s2} M2 2^{E2}$
- Exact Result:  $(-1)^s M 2^E$ 
  - Sign s:  $s1 \wedge s2 = s1 \text{ XOR } s2 = s1 + s2 \pmod{2}$
  - Significand M:  $M1 \times M2$
  - Exponent E:  $E1 + E2$
- Fixing
  - If  $M \geq 2$ , shift M right, increment E
  - If E out of range, overflow
  - Round M to fit **frac** precision
- Implementation
  - Biggest chore is multiplying significands

## Mathematical Properties of FP Add

### ■ Compare to those of Abelian Group

- Closed under addition? **Yes**
  - But may generate infinity or NaN
- Commutative? **Yes**
- Associative? **No**
  - Overflow and inexactness of rounding
  - $(3.14+1e10)-1e10 = 0$ ,  $3.14+(1e10-1e10) = 3.14$
- 0 is additive identity?
- Every element has additive inverse? **Yes**
  - Yes, except for infinities & NaNs **Almost**

### ■ Monotonicity

- $a \geq b \Rightarrow a+c \geq b+c$  **Almost**
  - Except for infinities & NaNs

## Mathematical Properties of FP Mult

### ■ Compare to Commutative Ring

- Closed under multiplication? **Yes**
  - But may generate infinity or NaN
- Multiplication Commutative? **Yes**
- Multiplication is Associative? **No**
  - Possibility of overflow, inexactness of rounding
  - Ex:  $(1e20 * 1e20) * 1e-20 = \text{inf}$ ,  $1e20 * (1e20 * 1e-20) = 1e20$
- 1 is multiplicative identity? **Yes**
- Multiplication distributes over addition? **No**
  - Possibility of overflow, inexactness of rounding
  - $1e20 * (1e20 - 1e20) = 0.0$ ,  $1e20 * 1e20 - 1e20 * 1e20 = \text{NaN}$

### ■ Monotonicity

- $a \geq b \ \& \ c \geq 0 \Rightarrow a * c \geq b * c$ ? **Almost**
  - Except for infinities & NaNs

# Table of contents

## Announcements

## Floats: Review

- Normalized: exp field

- Normalized: frac field

- Normalized/denormalized

- Special cases

## Floats: Mastery

- Normalized number bitstring to real number it represents

- Floating point multiplication

- Properties of floating point

## Instruction set architectures

- why are instruction set architectures important

- 8-bit vs. 16-bit. vs. 32-bit vs. 64-bit

- CISC vs. RISC

# Computer organization

## Layer cake

- ▶ Society
- ▶ Human beings
- ▶ Applications
- ▶ Algorithms
- ▶ High-level programming languages    Python, Java
- ▶ Interpreters
- ▶ Low-level programming languages
- ▶ Compilers
- ▶ Architectures
- ▶ Microarchitectures
- ▶ Sequential/combinational logic
- ▶ Transistors
- ▶ Semiconductors
- ▶ Materials science

# why are instruction set architectures important

Interface between computer science and electrical and computer engineering

- ▶ Software is varied, changes
- ▶ Hardware is standardized, static

## Computer architect Fred Brooks and the IBM 360

- ▶ IBM was selling computers with different capacities,
- ▶ Compile once, and can run software on all IBM machines.
- ▶ Backward compatibility.
- ▶ An influential idea.

# CISC vs. RISC

## Complex instruction set computer

- ▶ Intel and AMD
- ▶ Have an extensive and complex set of instructions
- ▶ For example: x86's extensions: x87, IA-32, x86-64, MMX, 3DNow!, SSE, SSE2, SSE3, SSSE3, SSE4, SSE4.2, SSE5, AES-NI, CLMUL, RDRAND, SHA, MPX, SGX, XOP, F16C, ADX, BMI, FMA, AVX, AVX2, AVX512, VT-x, VT-d, AMD-V, AMD-Vi, TSX, ASF
- ▶ Can license Intel's compilers to extract performance
- ▶ Secret: inside the processor, they break it down to more elementary instructions

# CISC vs. RISC

## Reduced instruction set computer

- ▶ MIPS, ARM, RISC-V (can find Patterson and Hennessy Computer Organization and Design textbook in each of these versions), an PowerPC
- ▶ Have a relatively simple set of instructions
- ▶ For example: ARM's extensions: SVE;SVE2;TME; All mandatory: Thumb-2, Neon, VFPv4-D16, VFPv4 Obsolete: Jazelle
- ▶ ARM: smartphones, Apple ARM M1 Mac

$$[1, 2, 4, 5, \dots 9] * 2 = [2, 4\dots]$$

# Into the future: Post-ISA world

## Post-ISA world

- ▶ Increasingly, the CPU is not the only character
- ▶ It orchestrates among many pieces of hardware
- ▶ Smartphone die shot
- ▶ GPU, TPU, FPGA, ASIC

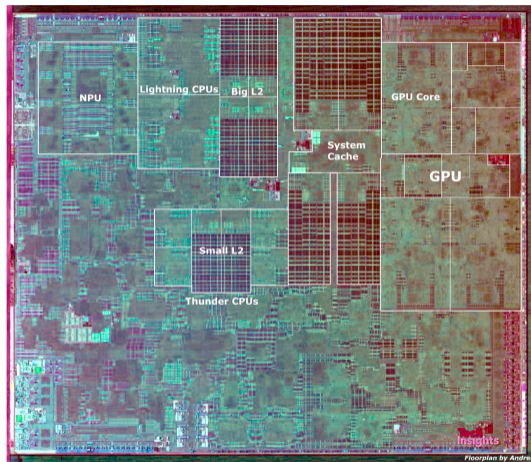


Figure: Apple A13 (2019 Apple iPhone 11 CPU). Image credit AnandTech