Floating Point I

CSE 351 Winter 2021

Instructor:	Teaching Assistants:		
Mark Wyse	Kyrie Dowling	Catherine Gueva	ra Ian Hsiao
	Jim Limprasert	Armin Magness	Allie Pfleger
	Cosmo Wang	Ronald Widjaja	



Administrivia

- hw5 due Wednesday, hw6 due Friday
- Lab 1a due tonight at 11:59 pm
 - Submit pointer.c and lab1Areflect.txt
 - Make sure you submit *something* to Gradescope before the deadline and that the file names are correct
 - Can use late day tokens to submit up until Mon 11:59 pm
- Lab 1b due next Friday (1/22)
 - Submitaisle_manager.c, store_client.c, and lab1Breflect.txt

Reading Review

- Terminology:
 - normalized scientific binary notation
 - trailing zeros
 - sign, mantissa, exponent ↔ bit fields S, M, and E
 - float, double
 - biased notation (exponent), implicit leading one (mantissa)
 - rounding errors
- Floating Point Simulator
 - https://www.h-schmidt.net/FloatConverter/IEEE754.html
- Questions from the Reading?

Review Questions

- Convert 11.375₁₀ to normalized binary scientific notation
- What is the correct value encoded by the following floating point number?

- bias = 2^{w-1}-1
- exponent = E bias
- mantissa = 1.M

Number Representation Revisited

- What can we represent in one word?
 - Signed and Unsigned Integers
 - Characters (ASCII)
 - Addresses
- How do we encode the following:
 - Real numbers (*e.g.*, 3.14159)
 - Very large numbers (*e.g.*, 6.02×10²³)
 - Very small numbers (*e.g.*, 6.626×10⁻³⁴)
 - Special numbers (e.g., ∞, NaN)

Floating Point Topics

- * Fractional binary numbers
- * IEEE floating-point standard
- Floating-point operations and rounding
- Floating-point in C

- There are many more details that we won't cover
 - It's a 58-page standard...

Representation of Fractions

 "Binary Point," like decimal point, signifies boundary between integer and fractional parts:



* Example: $10.1010_2 = 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-3} = 2.625_{10}$

Representation of Fractions

 "Binary Point," like decimal point, signifies boundary between integer and fractional parts:



- In this 6-bit representation:
 - What is the encoding and value of the smallest (most negative) number?
 - What is the encoding and value of the largest (most positive) number?
 - What is the smallest number greater than 2 that we can represent?

Scientific Notation (Binary)



- Normalized form: exactly one digit (non-zero) to left of binary point
- Computer arithmetic that supports this called floating point due to the "floating" of the binary point
 - Declare such variable in C as float (or double)

IEEE Floating Point

- IEEE 754 (established in 1985)
 - Standard to make numerically-sensitive programs portable
 - Specifies two things: representation scheme and result of floating point operations
 - Supported by all major CPUs
- Driven by numerical concerns
 - Scientists/numerical analysts want them to be as real as possible
 - Engineers want them to be easy to implement and fast
 - Scientists mostly won out:
 - Nice standards for rounding, overflow, underflow, but...
 - Hard to make fast in hardware
 - Float operations can be an order of magnitude slower than integer ops

Floating Point Encoding

- Use normalized, base 2 scientific notation:
 - Value: ±1 × Mantissa × 2^{Exponent}
 - Bit Fields: $(-1)^{S} \times 1.M \times 2^{(E-bias)}$
- Representation Scheme:
 - Sign bit (0 is positive, 1 is negative)
 - Mantissa (a.k.a. significand) is the fractional part of the number in normalized form and encoded in bit vector M
 - Exponent weights the value by a (possibly negative) power of 2 and encoded in the bit vector E



The Exponent Field

- Use biased notation
 - Read exponent as unsigned, but with bias of 2^{w-1}-1 = 127
 - Representable exponents roughly ½ positive and ½ negative
 - $Exp = E bias \leftrightarrow E = Exp + bias$
 - Exponent 0 (Exp = 0) is represented as E = 0b 0111 1111

- Why biased?
 - Makes floating point arithmetic easier
 - Makes somewhat compatible with two's complement hardware

The Mantissa (Fraction) Field



Note the implicit leading 1 in front of the M bit vector

- Gives us an extra bit of precision
- Mantissa "limits"
 - Low values near M = 0b0...0 are close to 2^{Exp}
 - High values near M = 0b1...1 are close to 2^{Exp+1}

Normalized Floating Point Conversions

- ♦ FP → Decimal
 - Append the bits of M to implicit leading 1 to form the mantissa.
 - 2. Multiply the mantissa by 2^{E-bias} .
 - 3. Multiply the sign $(-1)^{S}$.
 - Multiply out the exponent by shifting the binary point.
 - 5. Convert from binary to decimal.

♦ Decimal → FP

- Convert decimal to binary.
- Convert binary to normalized scientific notation.
- 3. Encode sign as S (0/1).
- Add the bias to exponent and encode E as unsigned.
- 5. The first bits after the leading 1 that fit are encoded into M.

Practice Question

Convert the decimal number -7.375 into floating point representation

Challenge Question

 Find the sum of the following binary numbers in normalized scientific binary notation:

 $1.01_2 \times 2^0 + 1.11_2 \times 2^2$

Precision and Accuracy

- Precision is a count of the number of bits in a computer word used to represent a value
 - Capacity for accuracy
- Accuracy is a measure of the difference between the actual value of a number and its computer representation
 - High precision permits high accuracy but doesn't guarantee it. It is possible to have high precision but low accuracy.
 - Example: float pi = 3.14;
 - pi will be represented using all 24 bits of the mantissa (highly precise), but is only an approximation (not accurate)

Need Greater Precision?

Double Precision (vs. Single Precision) in 64 bits



- C variable declared as double
- Exponent bias is now 2¹⁰-1 = 1023
- Advantages: greater precision (larger mantissa), greater range (larger exponent)
- Disadvantages: more bits used, slower to manipulate

Current Limitations

- Largest magnitude we can represent?
- Smallest magnitude we can represent?
 - Limited range due to width of E field
- What happens if we try to represent $2^0 + 2^{-30}$?
 - Rounding due to limited *precision*: stores 2⁰
- There is a need for *special cases*
 - How do we represent the value zero?
 - What about ∞ and NaN?

Summary

Floating point approximates real numbers:



- Handles large numbers, small numbers, special numbers
- Exponent in biased notation (bias = 2^{w-1} 1)
 - Size of exponent field determines our representable range
 - Outside of representable exponents is overflow and underflow
- Mantissa approximates fractional portion of binary point
 - Size of mantissa field determines our representable *precision*
 - Implicit leading 1 (normalized) except in special cases
 - Exceeding length causes *rounding*