CS 2340 - Computer Architecture

13 Floating Point Arithmetic
Dr. Alice Wang

A programmer walks into a bar and orders 1.000000119 root beers. The bartender says, "I'm gonna have to charge you extra; that's a root beer float". And the programmer says, "Well in that case make it a double".

Term Project – Grading

Sign up for a 12-minute interview slot with a grader during the week of Mon Oct 27- Fri Oct 31 by signing up for one slot in this Sign-up Genius.

During this interview, you will be asked to download and run your game as submitted, explain its functionality, and answer questions related to it. Your grade will be based on

- 20% documentation
- 20% demonstration
- 60% functionality (determined during the interview)

Review

Last Lecture

- Binary Arithmetic: Add, Sub, Mult, Division
- Fixed point

Today

- Floating Point

Pros/Cons with Fixed-point

- **Pros:**
  - Simple to understand
  - Simple to implement in circuits - fast, low power, low cost
- **Cons:**
  - Limited in bit precision
  - Limited in range

Floating point standard

- We demand more precision and range! Floating point standard emerged
- Defined by IEEE Std 754-1985
  - Developed in response to divergence of representations
  - Now almost universally adopted
- Two representations
  - Single precision (32-bit) - “float” in C
  - Double precision (64-bit) - “double” in C and “float” in Python

Floating point Numbers – Decimal

- Floating point is similar to decimal scientific notation
- For example, write 273_{10} in scientific notation:

$$\mathbf{273 = 2.73 \times 10^2}$$

- In general, a number is written in scientific notation as:

$$\pm \mathbf{M \times B^E}$$

  - **M** = mantissa (decimal point is always after the first non-zero digit)
  - **B** = base
  - **E** = exponent
  - In the decimal example, M = 2.73, B = 10, and E = 2

Floating point Numbers – Binary

- For binary floating point numbers, the Base is 2

  $$\pm \mathbf{M} \times \mathbf{B}^{\mathbf{E}}$$

  - **M** = mantissa (binary point is always after the first non-zero digit)
  - **B** = base = 2
  - **E** = exponent
- Examples of binary scientific notation
  - $1.01101110001 \times 2^{23}$
  - -1.01×2^{3}

Single-precision Floating point Numbers

Single-precision is 32-bit

We will use this example to explain three floating point versions

– Version 3 is called the **IEEE 754 floating-point standard**

Example: represent the value 228_{10} using a 32-bit floating point representation

Floating point Representation – Version 1

1. Convert decimal to binary:

$$228_{10} = 11100100_2$$

2. Write the number in “binary scientific notation”:

$$11100100_2 = 1.11001_2 \times 2^7$$

3. Fill in each field of the 32-bit floating point number:
   - The sign bit is positive (0)
   - The 8 exponent bits represent the value 7
   - The remaining 23 bits are the mantissa

1 bit	8 bits	23 bits
0	00000111	11 1001 0000 0000 0000 0000
Sign	**Exponent**	**Mantissa**

Floating point Representation – Version 2

Because the first bit of the mantissa is always 1:

$$228_{10} = 11100100_2 = \mathbf{1.11001} \times 2^7$$

There is no need to store it ⇒ *implicit leading 1*

Store only the fraction bits in 23-bit field & get 1 extra bit of precision

1 bit	8 bits	23 bits
0	00000111	110 0100 0000 0000 0000 0000
Sign	**Exponent**	**Fraction**

Floating point Representation – Version 3

Biased exponent: bias = 127 (01111111_2)

- Biased exponent = bias + exponent → Gives us the ability to store both pos and neg exponents
- Exponent of 7 is stored as:

$$127 + 7 = 134 = 0x10000110_2$$

This is the **IEEE 754 32-bit floating-point representation** of 228_{10} in hexadecimal: **0x43640000**

1 bit	8 bits	23 bits
0	10000110	110 0100 0000 0000 0000 0000
Sign	**Biased Exponent**	**Fraction**

Example (Decimal to Float) – My Turn

Write decimal -18.125 in single-precision floating point (IEEE 754)

1. Convert decimal to binary:

   18.125 = ______________

2. Write in binary scientific notation:

   ____

3. Fill in fields:

1bit	8bits	23bits
Sign	Biased Exp	Fraction

Sign bit: ______________

Exponent: ____________________

Biased exponent (8 bits): ______________

Mantissa: ______________________________

Fraction (23 bits): ______________________________

in hex: ________________

Example (Decimal to Float) – Your Turn

Write decimal 14.5625 in single-precision floating point (IEEE 754)

1. Convert decimal to binary:

   14.5625 = ______________________

2. Write in binary scientific notation:

   ____

3. Fill in fields:

1bit	8bits	23bits
Sign	Biased Exp	Fraction

in hex: ______________________

Sign bit: __________________

Exponent: ________________________

Biased exponent (8 bits): __________________

Mantissa: ______________________________________

Fraction (23 bits): ______________________________________

Example (Float pt to Dec) – My Turn

What is IEEE 754 floating point 0x44050000 in decimal?

1. Convert hexadecimal to binary:
2. Fill in fields:

Sign bit: ______

Biased exponent bits: ____________________

Fraction bits: ____________________________

Exponent: ____________________

Mantissa: ____________________

Binary Scientific Notation: ____________________

Binary: _________________ **Decimal :** ____________________

Example (Float pt to Dec) – Your Turn

What is IEEE 754 floating point 0x41340000 in decimal?

1. Convert hexadecimal to binary:
2. Fill in fields:

Sign bit: ______

Biased exponent bits: ____________________

Fraction bits: ____________________________

Exponent: ____________________

Mantissa: ____________________

Binary Scientific Notation: ____________________

Binary: _________________ **Decimal :** ____________________

Floating Point Special Cases

Number	Sign	Exponent	Fraction
0	X	00000000	00000000000000000000000
∞	0	11111111	00000000000000000000000
- ∞	1	11111111	00000000000000000000000
NaN	X	11111111	non-zero

Single vs Double precision

	Single	Double
# of bits	**32**	**64**
# Exponent bits	**8**	**11**
# of Fractional bits	**23**	**52**
Reserved Exponents	**0000_0000** **1111_1111**	**000_0000_0000** **111_1111_1111**
Smallest Value	Exponent: 0000_0001 Fraction: 000…00⇒ Mantissa 1.0 $\pm 1.0 \times 2^{-126} \approx \pm 1.2 \times 10^{-38}$	Exponent: 000_0000_0001 Fraction: 000…00⇒ Mantissa 1.0 $\pm 1.0 \times 2^{-1022} \approx \pm 2.2 \times 10^{-308}$
Largest Value	Exponent: 1111_1110 Fraction: 111…11⇒ Mantissa 2.0 $\pm 2.0 \times 2^{+127} \approx \pm 3.4 \times 10^{+38}$	Exponent: 111_1111_1110 Fraction: 111…11⇒ Mantissa 2.0 $\pm 2.0 \times 2^{+1023} \approx \pm 1.8 \times 10^{+308}$

Floating point Arithmetic – Addition

- Floating point addition is more complex than fixed-point and integer addition!
- Show floating point addition in decimal first, then in floating point
- Key point: You can only add in scientific notation if the exponents are the same

Example in decimal:

Add 3.5×10^2 by 5.5×10^4

- Adjust one of the addends so that the exponents are the same: 550×10^2
- Add 3.5 and 550 = 553.5
- Exponent : 2
- Adjust the decimal point of the result (if needed) → 5.535×10^4

Floating point Arithmetic – Addition

- Floating point addition is more complex than fixed-point and integer addition!
- Show floating point addition in decimal first, then in floating point
- Key point: You can only add in scientific notation if the exponents are the same

Example in decimal:

Add 3.5×10^2 by 5.5×10^4

- Adjust one of the addends so that the exponents are the same: 550×10^2
- Add 3.5 and 550 = 553.5
- Exponent : 2
- Adjust the decimal point of the result (if needed) → 5.535×10^4

Translate to binary:

Add 1.1×2^2 by 1.101×2^4

- Adjust one of the addends so that the exponents are the same: 110.1×2^2
- Add 1.1and 110.1 = 1000.0
- Exponent : 2
- Adjust the binary point (if needed) → 1.0×2^5

Floating point Addition Example

Add the two floating-point numbers and give the result in IEEE 754 floating-point

0x3fc00000 + 0x41200000 [1.5 + 10 = 11.5]

1. Extract exponent and fraction bits. Get into binary scientific notation	0x3FC00000: Sign = Biased Exponent = Exponent = Fraction = Mantissa = Binary Scientific =	0x41200000: Sign = Biased Exponent = Exponent = Fraction = Mantissa = Binary Scientific =

Floating point Addition Example

Add 2 binary scientific numbers: $(1.1 \times 2^0) + (1.01 \times 2^3)$

Sign	Mantissa	Exponents
Check the sign to determine if you need to apply 2's complement before adding	Shift Mantissa of the larger Exponent so that the Exponents are the same before applying binary addition.	Larger Exponent shifted so that Exponents of both numbers are the same.
Sign1 = + Sign2 = + Apply 2's complement?	Mantissa1 = 1.1 Mantissa2 = 1.01 → Shift Mantissa1 = Apply binary addition +____________________	Exponent1 = 0 Exponent2 = 3 →

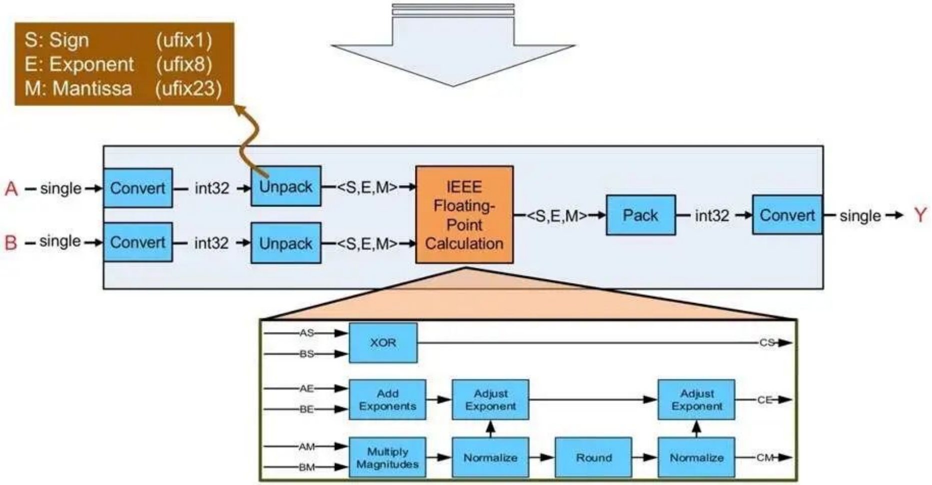
Floating point Addition Example

Adjust the binary point (if needed)	
Assemble exponent and fraction back into IEEE 754 floating point format	Sign = Exponent = Biased Exponent = Mantissa = Fraction =

Hex: ____________________

Fixed-point vs Floating-point addition

$$1.5 + 10 = 11.5$$

Fixed point	Floating point
$\begin{array}{r} 0001.1 \\ \underline{+1010.0} \\ 1011.1 \end{array}$	0x3fc00000 + 0x41200000 

Fixed-point vs Floating-point

Fixed point

- Very fast
- No complex logic, reuse integer logic
- Fixed accuracy
- Can only represent small number set

Floating point

- Slower
- Dedicated complex logic required
- Accuracy varies
- Represent large data sets

Summary

- How do we represent Fractions: Fixed point and Floating point
- Fixed-point unsigned, signed, arithmetic
- Floating-point : IEEE 754 32-bit floating-point representation

Next time: MIPS floating-point instructions

Next lecture

Floating point instructions