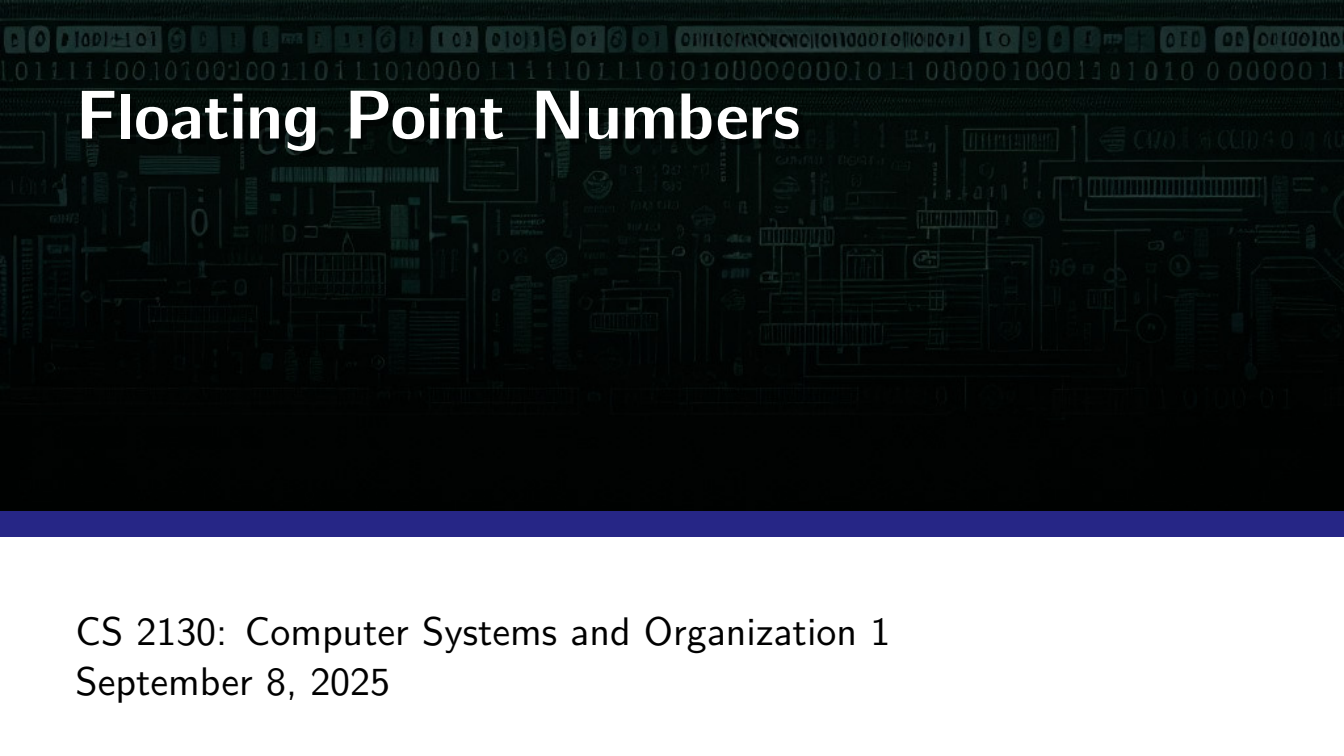




Floating Point Numbers

CS 2130: Computer Systems and Organization 1
September 8, 2025

Announcements

$$-x = \sim x + 1$$

2's complement

- Homework 1 due September 15
- Lab 2 tomorrow!

Operations

So far, we have discussed:

- Addition: $x + y$
 - Can get multiplication
- Subtraction: $x - y$
 - Can get division, but more difficult
- Unary minus (negative): $-x$
 - Flip the bits and add 1

Operations (on Integers)

Bit vector: fixed-length sequence of bits (ex: bits in an integer)

- Manipulated by bitwise operations

Bitwise operations: operate over the bits in a bit vector

- Bitwise not: $\sim x$ - flips all bits (unary)
- Bitwise and: $x \ \& \ y$ - set bit to 1 if x, y have 1 in same bit
- Bitwise or: $x \ | \ y$ - set bit to 1 if either x or y have 1
- Bitwise xor: $x \ ^ \ y$ - set bit to 1 if x, y bit differs

Operations (on Integers)

- Logical not: $!x$
 - $!0 = 1$ and $!x = 0, \forall x \neq 0$
 - Useful in C, no booleans
 - Some languages name this one differently
- Left shift: $x \ll y$ - move bits to the left
 - Effectively multiply by powers of 2
- Right shift: $x \gg y$ - move bits to the right
 - Effectively divide by powers of 2
 - Signed (extend sign bit) vs unsigned (extend 0)

Right Bit-shift Example 2

0xCA

unsigned

$$\begin{array}{r} 128 \\ + 64 \\ + 8 \\ + 2 \\ \hline 202 \end{array} \div 2^3 = 25.25$$

11001010 >> **3**

Diagram showing bit shifts for 11001010. Red arrows indicate the shift of bits 3, 4, and 5 to the right. The original bits 0, 1, and 2 are crossed out. The result is 11001.

signed

$$\begin{array}{r} 11001010 = -54 \\ 60110101 \\ + 1 \\ \hline 00110110 = 54 \\ -54/8 \approx -7 \\ -6.75 \end{array}$$

(0xCA >> 3) << 3

Diagram showing the result of the right shift (11001) shifted left by 3 bits. The result is 111.

1111100 = **-7**

$$\begin{array}{r} 110 \\ + 1 \\ \hline 111 = 7 \end{array}$$

Right Bit-shift Example 2

For **signed** integers, extend the sign bit (1)

- Keeps negative value (if applicable)
- Approximates divide by powers of 2

11001010 >> 1

Quiz 1 Review

$$\begin{array}{r} -a = na + 1 \\ +a \qquad \qquad +a \\ \hline -1 \quad 0 = na + 1 + a - 1 \\ \\ -1 = na + a \end{array}$$

What about other kinds of numbers?

Non-Integer Numbers

Floating point numbers

- Decimal: 3.14159

↑
decimal

$$\begin{array}{ccccccc} & & & & 26 & 75 & 3. \\ 10^{-2} & 10^{-1} & 10^0 & 10^{-1} & 10^{-2} & 10^{-3} & \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ 1 & 3 & . & 1 & 4 & 1 & 5 \\ & & & \frac{1}{10} & \frac{1}{100} & \frac{1}{1000} & \end{array}$$

Non-Integer Numbers

Floating point numbers

- Decimal: 3.14159
- Binary: 11.10110

↑
binary point

3	2	1	0	-1	-2	-3
2	2	2	2	2	2	2
		1	1	1	0	1
				↓	↓	↓
				$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

Non-Integer Numbers

Floating point numbers

- Decimal: 3.14159
- Binary: 11.10110
- With integers, the point is always fixed after all digits
- With floating point numbers, the point can move!

Non-Integer Numbers

Floating point numbers

- Decimal: 3.14159
- Binary: 11.10110
- With integers, the point is always fixed after all digits
- With floating point numbers, the point can move!

Challenge! only 2 symbols in binary

Scientific Notation

Convert the following decimal to scientific notation:

2130

2.130×10^3

Scientific Notation

Convert the following binary to scientific notation:

101101.

$$+ \underline{1.01101} \times 2^5$$

Something to Notice

An interesting phenomenon:

- Decimal: first digit can be any number *except* 0

$$\underline{2.13} \times 10^3$$

$$\cancel{0.213} \times 10^4$$

Something to Notice

An interesting phenomenon:

- Decimal: first digit can be any number *except* 0

$$2.13 \times 10^3$$

- Binary: first digit can be any number *except* 0 **Wait!**

$$1.01101 \times 2^5$$

Something to Notice

An interesting phenomenon:

- Decimal: first digit can be any number *except* 0

$$2.13 \times 10^3$$

- Binary: first digit can be any number *except* 0 **Wait!**

$$\textcircled{+} \textcircled{1}.\textcircled{0}\textcircled{1}\textcircled{1}\textcircled{0}\textcircled{1} \times 2^{\textcircled{5}}$$

- First digit can only be 1

Floating Point in Binary

We must store 3 components

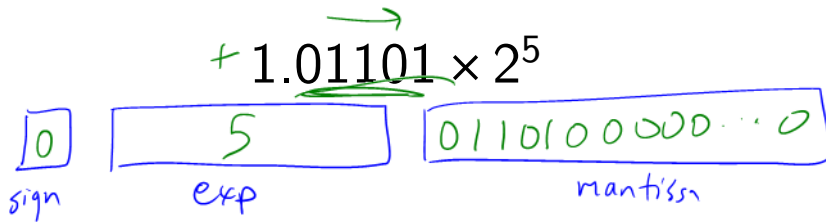
- **sign** (1-bit): 1 if negative, 0 if positive
- **fraction** or **mantissa**: (?-bits): bits after binary point
- **exponent** (?-bits): how far to move binary point

We do not need to store the value before the binary point. Why?

Floating Point in Binary

How do we store them?

- Originally many different systems
- IEEE standardized system (IEEE 754 and IEEE 854)
- Agreed-upon order, format, and number of bits for each



Example

A rough example in Decimal:

$$6.42 \times 10^3$$

0 0 3 4 2 0

$$\frac{420}{1000}$$

10^3

need to store 6!

1 ^{digit} ~~sign~~ sign
2 digit exp
3 digits frac

Exponent

How do we store the exponent?

$$\underline{2.13} \times 10^{\underline{-2}} \\ = .0213$$

- Exponents *can* be negative

$$2^{-3} = \frac{1}{2^3} = \frac{1}{8}$$

- Need positive and negative ints (but no minus sign)

Exponent

How do we store the exponent?

- Exponents *can* be negative

$$2^{-3} = \frac{1}{2^3} = \frac{1}{8}$$

- Need positive and negative ints (but no minus sign)
- *Don't we always use Two's Complement?*

Exponent

How do we store the exponent?

- Exponents *can* be negative

$$2^{-3} = \frac{1}{2^3} = \frac{1}{8}$$

- Need positive and negative ints (but no minus sign)
- *Don't we always use Two's Complement?* Unfortunately Not

Exponent

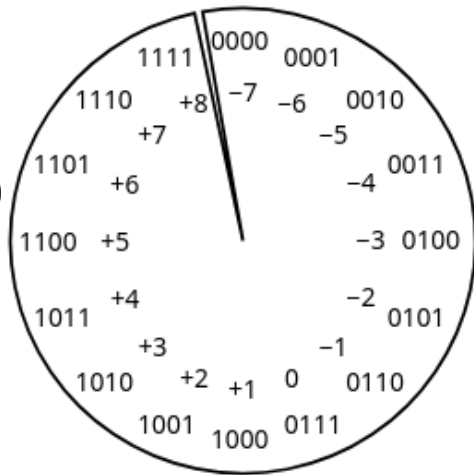
How do we store the exponent?

- Biased integers
 - Make comparison operations run more smoothly
 - Hardware more efficient to build
 - Other valid reasons

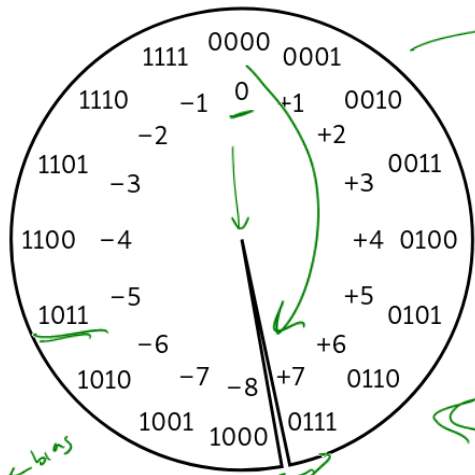
Biased Integers

Similar to Two's Complement, but add **bias**

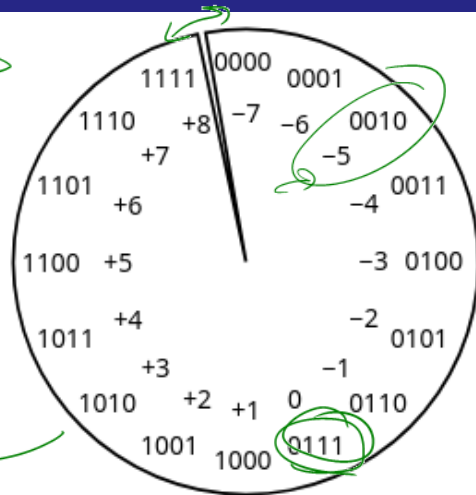
- **Two's Complement:** Define 0 as $00\dots0$
- **Biased:** Define 0 as $0111\dots1$
- Biased wraps from $000\dots0$ to $111\dots1$



Biased Integers



Two's Complement



Biased

1111
1011
+0111

0010

+0111

~~0111~~
-0111

Biased Integers Example

Calculate value of biased integers (4-bit example)

$$\begin{array}{r} \begin{array}{cccc} 1 & 1 & 1 & 1 \\ | & | & | & | \\ 0 & 0 & 1 & 0 \end{array} = 2 \\ - \begin{array}{cccc} 0 & 1 & 1 & 1 \end{array} = 7 \\ \hline \begin{array}{cccc} 1 & 0 & 1 & 1 \end{array} = -5 \\ \text{2's complement} \rightarrow \end{array}$$
$$\begin{array}{r} \sim \quad 0100 \\ +1 \quad 1 \\ \hline 0101 = 5 \end{array}$$

Biased Integers

Floating Point Example

101.011_2

Floating Point Example

101.011_2

1 sign
4 exp
3 mantissa

Floating Point Example

What does the following encode?

1	001110	1010101
---	--------	---------

Floating Point Example

What does the following encode?

1	001110	1010101
---	--------	---------



What about 0?

Floating Point Numbers

Four cases:

- **Normalized:** What we have seen today

$$s \text{ } eeee \text{ } ffff = \pm 1.ffff \times 2^{eeee - \text{bias}}$$

- **Denormalized:** Exponent bits all 0

$$s \text{ } eeee \text{ } ffff = \pm 0.ffff \times 2^{1 - \text{bias}}$$

- **Infinity:** Exponent bits all 1, fraction bits all 0 (i.e., $\pm\infty$)
- **Not a Number (NaN):** Exponent bits all 1, fraction bits not all 0