Lecture 5: Floating point

Philipp Koehn, David Hovemeyer

January 31, 2024

601.229 Computer Systems Fundamentals

Floating point numbers

Numbers

- So far, we only dealt with integers
- But there are other types of numbers

Numbers

- So far, we only dealt with integers
- But there are other types of numbers
- Rational numbers (from ratio $\simeq$ fraction)
  - $3/4 = 0.75$
  - $10/3 = 3.33333333....$

Numbers

- So far, we only dealt with integers
- But there are other types of numbers
- Rational numbers (from ratio $\simeq$ fraction)
  - $3/4 = 0.75$
  - $10/3 = 3.33333333....$
- Real numbers
  - $\pi = 3.14159265...$
  - $e = 2.71828182...$

Very Large Numbers

- Distance of sun and earth

$$150,000,000,000 \text{ meters}$$

- Scientific notation

$$1.5 \times 10^{11} \text{ meters}$$

- Another example: number of atoms in 12 gram of carbon-12 (1 mol)

$$6.022140857 \times 10^{23}$$

Binary Numbers in Scientific Notation

- Example binary number (π again)

$$11.0010010001$$

- Scientific notation

$$1.10010010001 \times 2^1$$

- General form

$$1.x \times 2^y$$

Representation

- IEEE 754 floating point standard
- Uses 4 bytes

31	30	29	...	24	23	22	21	...	1	0
s	exponent					fraction				

1 bit — 8 bits — 23 bits

- Exponent is offset with a bias of 127

  e.g. $2^{-6} \rightarrow$ exponent $=$ -6 $+$ 127 $=$ 121

Conversion into Binary

- $\pi = 3.14159265$
- Number before period: $3_{10} = 11_2$
- Conversion of fraction .14159265

Conversion into Binary

- $\pi = 3.14159265$
- Number before period: $3_{10} = 11_2$
- Conversion of fraction .14159265

Digit	**Calculation**
	$0.14159265 \times 2 \downarrow$

Conversion into Binary

- $\pi = 3.14159265$
- Number before period: $3_{10} = 11_2$
- Conversion of fraction .14159265

Digit	**Calculation**
	$0.14159265 \times 2 \downarrow$
0	0.2831853

Conversion into Binary

- $\pi = 3.14159265$
- Number before period: $3_{10} = 11_2$
- Conversion of fraction .14159265

Digit	**Calculation**
	$0.14159265 \times 2 \downarrow$
0	$0.2831853 \times 2 \downarrow$
0	0.5663706

Conversion into Binary

- $\pi = 3.14159265$
- Number before period: $3_{10} = 11_2$
- Conversion of fraction .14159265

Digit	**Calculation**
	$0.14159265 \times 2 \downarrow$
0	$0.2831853 \times 2 \downarrow$
0	$0.5663706 \times 2 \downarrow$
1	0.1327412

Conversion into Binary

- $\pi = 3.14159265$
- Number before period: $3_{10} = 11_2$
- Conversion of fraction .14159265

Digit	Calculation	Digit	Calculation
	$0.14159265 \times 2 \downarrow$	1	$0.9817472 \times 2 \downarrow$
0	$0.2831853 \times 2 \downarrow$	1	$0.9634944 \times 2 \downarrow$
0	$0.5663706 \times 2 \downarrow$	1	$0.9269888 \times 2 \downarrow$
1	$0.1327412 \times 2 \downarrow$	1	$0.8539776 \times 2 \downarrow$
0	$0.2654824 \times 2 \downarrow$	1	$0.7079552 \times 2 \downarrow$
0	$0.5309648 \times 2 \downarrow$	1	$0.4159104 \times 2 \downarrow$
1	$0.0619296 \times 2 \downarrow$	0	$0.8318208 \times 2 \downarrow$
0	$0.1238592 \times 2 \downarrow$	1	$0.6636416 \times 2 \downarrow$
0	$0.2477184 \times 2 \downarrow$	1	$0.3272832 \times 2 \downarrow$
0	$0.4954368 \times 2 \downarrow$	0	$0.6545664 \times 2 \downarrow$
0	$0.9908736 \times 2 \rightarrow$	1	0.3091328×2

- Binary: 11.0010010000111111101101

Encoding into Representation

- π

$$1.10010010000111111101101 \times 2^1$$

- Encoding

Sign	Exponent	Fraction
0	10000000	10010010000111111101101

- Note: leading 1 in fraction is omitted

Clicker quiz!

Clicker quiz omitted from public slides

See the representation of a `float`

```
#include <stdio.h>

int main(void) {
  float x;
  scanf("%f", &x);
  unsigned *p = (unsigned *) &x;

  for (int i = 31; i >= 0; i--) {
    printf("%c", (*p & (1 << i)) ? '1' : '0');
    if (i == 31 || i == 23) { printf(" "); }
  }
  printf("\n");

  return 0;
}
```

See the representation of a `float`

```
$ gcc explain.c
$ echo '-18.8203125' | ./a.out
1 10000011 00101101001000000000000
```

Special Cases

- Zero

Special Cases

- Zero
- Infinity (1/0)
- Negative infinity (-1/0)

Special Cases

- Zero
- Infinity (1/0)
- Negative infinity (-1/0)
- Not a number (0/0 or $\infty - \infty$)

Encoding

Exponent	**Fraction**	**Object**
0	0	zero
0	$>$0	denormalized number
1-254	anything	floating point number
255	0	infinity
255	$>$0	NaN (not a number)

(denormalized number: $0.x \times 2^{-126}$)

Clicker quiz!

Clicker quiz omitted from public slides

Double Precision

- Single precision $=$ 4 bytes

Sign	**Exponent**	**Fraction**
1 bit	8 bits	23 bits

- Double precision $=$ 8 bytes

Sign	**Exponent**	**Fraction**
1 bit	11 bits	52 bits

Addition

Addition with Scientific Notation

- Decimal example, with 4 significant digits in encoding
- Example

$$0.1610 + 99.99$$

- In scientific notation

$$1.610 \times 10^{-1} + 9.999 \times 10^{1}$$

Addition with Scientific Notation

- Decimal example, with 4 significant digits in encoding
- Example

$$0.1610 + 99.99$$

- In scientific notation

$$1.610 \times 10^{-1} + 9.999 \times 10^{1}$$

- Bring lower number on same exponent as higher number

$$0.01610 \times 10^{1} + 9.999 \times 10^{1}$$

Addition with Scientific Notation

- Round to 4 significant digits

$$0.016 \times 10^1 + 9.999 \times 10^1$$

Addition with Scientific Notation

- Round to 4 significant digits

$$0.016 \times 10^1 + 9.999 \times 10^1$$

- Add fractions

$$0.016 + 9.999 = 10.015$$

Addition with Scientific Notation

- Round to 4 significant digits

$$0.016 \times 10^1 + 9.999 \times 10^1$$

- Add fractions

$$0.016 + 9.999 = 10.015$$

- Adjust exponent

$$10.015 \times 10^1 = 1.0015 \times 10^2$$

Addition with Scientific Notation

- Round to 4 significant digits
$$0.016 \times 10^1 + 9.999 \times 10^1$$
- Add fractions
$$0.016 + 9.999 = 10.015$$
- Adjust exponent
$$10.015 \times 10^1 = 1.0015 \times 10^2$$
- Round to 4 significant digits
$$1.002 \times 10^2$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10}$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10} = \frac{1}{2^1}_{10}$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10} = \frac{1}{2^1}_{10} = 0.1_2$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10} = \frac{1}{2^1}_{10} = 0.1_2 = 1.000_2 \times 2^{-1}$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10} = \frac{1}{2^1}_{10} = 0.1_2 = 1.000_2 \times 2^{-1}$$

$$-0.4375_{10} = -\frac{7}{16}_{10}$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10} = \frac{1}{2^1}_{10} = 0.1_2 = 1.000_2 \times 2^{-1}$$

$$-0.4375_{10} = -\frac{7}{16}_{10} = -\frac{7}{2^4}_{10} = 0.0111_2$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10} = \frac{1}{2^1}_{10} = 0.1_2 = 1.000_2 \times 2^{-1}$$
$$-0.4375_{10} = -\frac{7}{16}_{10} = -\frac{7}{2^4}_{10} = 0.0111_2 = -1.110_2 \times 2^{-2}$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10} = \frac{1}{2^1}_{10} = 0.1_2 = 1.000_2 \times 2^{-1}$$

$$-0.4375_{10} = -\frac{7}{16}_{10} = -\frac{7}{2^4}_{10} = 0.0111_2 = -1.110_2 \times 2^{-2}$$

- Bring lower number on same exponent as higher number

$$-1.110 \times 2^{-2} = -0.111 \times 2^{-1}$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10} = \frac{1}{2^1}_{10} = 0.1_2 = 1.000_2 \times 2^{-1}$$
$$-0.4375_{10} = -\frac{7}{16}_{10} = -\frac{7}{2^4}_{10} = 0.0111_2 = -1.110_2 \times 2^{-2}$$

- Bring lower number on same exponent as higher number

$$-1.110 \times 2^{-2} = -0.111 \times 2^{-1}$$

- Add the fractions

$$1.000_2 \times 2^{-1} + (-0.111 \times 2^{-1}) = 0.001 \times 2^{-1}$$

Binary Floating Point Addition

- Numbers

$$0.5_{10} = \frac{1}{2}_{10} = \frac{1}{2^1}_{10} = 0.1_2 = 1.000_2 \times 2^{-1}$$

$$-0.4375_{10} = -\frac{7}{16}_{10} = -\frac{7}{2^4}_{10} = 0.0111_2 = -1.110_2 \times 2^{-2}$$

- Bring lower number on same exponent as higher number

$$-1.110 \times 2^{-2} = -0.111 \times 2^{-1}$$

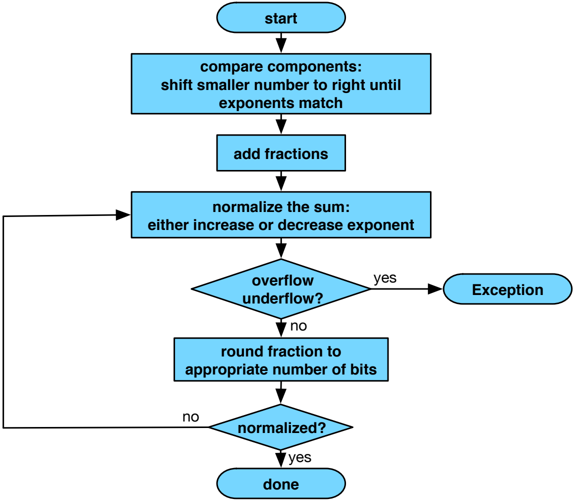
- Add the fractions

$$1.000_2 \times 2^{-1} + (-0.111 \times 2^{-1}) = 0.001 \times 2^{-1}$$

- Adjust exponent

$$0.001 \times 2^{-1} = 1.000 \times 2^{-4}$$

Flowchart


start
compare components:
shift smaller number to right until
exponents match
add fractions
normalize the sum:
either increase or decrease exponent
overflow
underflow?
yes
Exception
no
round fraction to
appropriate number of bits
no
normalized?
yes
done

Multiplication

Multiplication with Scientific Notation

- Example: multiply 1.110×10^{10} and 9.200×10^{-5}

Multiplication with Scientific Notation

- Example: multiply 1.110×10^{10} and 9.200×10^{-5}

$$1.110 \times 10^{10} \;\times\; 9.200 \times 10^{-5}$$

Multiplication with Scientific Notation

- Example: multiply 1.110×10^{10} and 9.200×10^{-5}

$$1.110 \times 10^{10} \; \times \; 9.200 \times 10^{-5}$$

$$1.110 \times 9.200 \times 10^{-5} \times 10^{10}$$

Multiplication with Scientific Notation

- Example: multiply 1.110×10^{10} and 9.200×10^{-5}

$$1.110 \times 10^{10} \times 9.200 \times 10^{-5}$$

$$1.110 \times 9.200 \times 10^{-5} \times 10^{10}$$

$$1.110 \times 9.200 \times 10^{-5+10}$$

Multiplication with Scientific Notation

- Example: multiply 1.110×10^{10} and 9.200×10^{-5}

$$1.110 \times 10^{10} \ \times \ 9.200 \times 10^{-5}$$

$$1.110 \times 9.200 \times 10^{-5} \times 10^{10}$$

$$1.110 \times 9.200 \times 10^{-5+10}$$

- Add exponents

$$-5 + 10 = 5$$

Multiplication with Scientific Notation

- Example: multiply 1.110×10^{10} and 9.200×10^{-5}

$$1.110 \times 10^{10} \;\times\; 9.200 \times 10^{-5}$$

$$1.110 \times 9.200 \times 10^{-5} \times 10^{10}$$

$$1.110 \times 9.200 \times 10^{-5+10}$$

- Add exponents

$$-5 + 10 = 5$$

- Multiply fractions

$$1.110 \times 9.200 = 10.212$$

Multiplication with Scientific Notation

- Example: multiply 1.110×10^{10} and 9.200×10^{-5}

$$1.110 \times 10^{10} \ \times \ 9.200 \times 10^{-5}$$

$$1.110 \times 9.200 \times 10^{-5} \times 10^{10}$$

$$1.110 \times 9.200 \times 10^{-5+10}$$

- Add exponents

$$-5 + 10 = 5$$

- Multiply fractions

$$1.110 \times 9.200 = 10.212$$

- Adjust exponent

$$10.212 \times 10^{5} = 1.0212 \times 10^{6}$$

Binary Floating Point Multiplication

- Example

$$1.000 \times 2^{-1} \times -1.110 \times 2^{-2}$$

Binary Floating Point Multiplication

- Example

$$1.000 \times 2^{-1} \ \times \ -1.110 \times 2^{-2}$$

- Add exponents

$$-1 + (-2) = -3$$

Binary Floating Point Multiplication

- Example

$$1.000 \times 2^{-1} \ \times \ -1.110 \times 2^{-2}$$

- Add exponents

$$-1 + (-2) = -3$$

- Multiply fractions

$$1.000 \times -1.110 = -1.110$$

Binary Floating Point Multiplication

- Example

$$1.000 \times 2^{-1} \;\times\; -1.110 \times 2^{-2}$$

- Add exponents

$$-1 + (-2) = -3$$

- Multiply fractions

$$1.000 \times -1.110 = -1.110$$

$$1000 \times 1110 = 1110000$$

Binary Floating Point Multiplication

- Example

$$1.000 \times 2^{-1} \;\times\; -1.110 \times 2^{-2}$$

- Add exponents

$$-1 + (-2) = -3$$

- Multiply fractions

$$1.000 \times -1.110 = -1.110$$

$$1000 \times 1110 = 1110000$$

$$-1.110000$$

Binary Floating Point Multiplication

- Example

$$1.000 \times 2^{-1} \;\times\; -1.110 \times 2^{-2}$$

- Add exponents

$$-1 + (-2) = -3$$

- Multiply fractions

$$1.000 \times -1.110 = -1.110$$

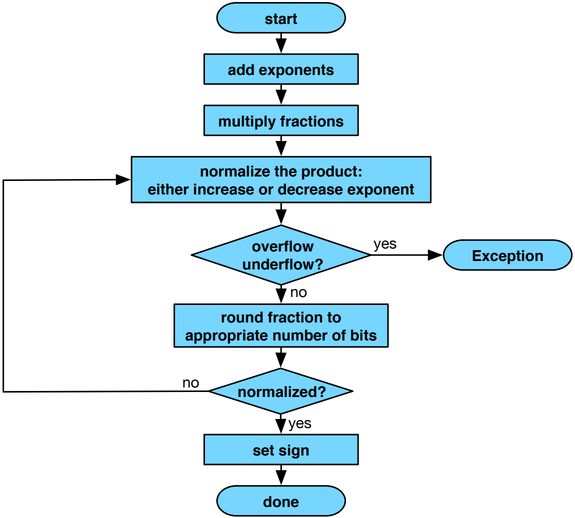
$$1000 \times 1110 = 1110000$$

$$-1.110000$$

- Adjust exponent (not needed)

$$-1.110 \times 2^{-3}$$

Flowchart



start
add exponents
multiply fractions
normalize the product:
either increase or decrease exponent
overflow
underflow?
yes
Exception
no
round fraction to
appropriate number of bits
no
normalized?
yes
set sign
done