

**CSE1301**  
**Computer Programming**  
**Lecture 30:**  
**Real Number Representation**

1

**Topics**

- Terminology
- IEEE standard for floating-point representation
- Floating point arithmetic
- Limitations

2

**Terminology**

- All digits in a number following any leading zeros are *significant digits*:

12.345  
-0.12345  
0.00012345

3

**Terminology (cont)**

- The scientific notation for real numbers is:

$mantissa \times base^{exponent}$

In C, the expression: **12.456e-2**  
means:  $12.456 \times 10^{-2}$

4

**Terminology (cont)**

- The mantissa is always *normalized* between 1 and the base (i.e., exactly one significant digit before the point)

<u>Unnormalized</u>	<u>Normalized</u>
$2997.9 \times 10^5$	$2.9979 \times 10^8$
$B1.39FC \times 16^{11}$	$B.139FC \times 16^{12}$
$0.010110110101 \times 2^{-1}$	$1.0110110101 \times 2^{-3}$

5

**Terminology (cont)**

- The precision of a number is how many digits (or bits) we use to represent it
- For example:  
3  
3.14  
3.1415926  
3.1415926535897932384626433832795028

6

### Representing Numbers

- A real number  $n$  is represented by a floating-point approximation  $n^*$
- The computer uses 32 bits (or more) to store each approximation
- It needs to store
  - the mantissa
  - the sign of the mantissa
  - the exponent (with its sign)

7

### Representing Numbers (cont)

- The standard way to allocate 32 bits (specified by IEEE Standard 754) is:
  - 23 bits for the mantissa
  - 1 bit for the mantissa's sign
  - 8 bits for the exponent

8

### Representing Numbers (cont)

- 23 bits for the mantissa
- 1 bit for the mantissa's sign
- 8 bits for the exponent

9

### Representing Numbers (cont)

- 23 bits for the mantissa
- 1 bit for the mantissa's sign
- 8 bits for the exponent

10

### Representing Numbers (cont)

- 23 bits for the mantissa
- 1 bit for the mantissa's sign
- 8 bits for the exponent

11

### Representing the Mantissa

- The *mantissa* has to be in the range  $1 \leq \text{mantissa} < \text{base}$
- Therefore
  - If we use base 2, the digit before the point must be a 1
  - So we don't have to worry about storing it
    - ➔ We get 24 bits of precision using 23 bits

12

### Representing the Mantissa (cont)

- 24 bits of precision are equivalent to a little over 7 decimal digits:

$$\frac{24}{\log_2 10} \approx 7.2$$

13

### Representing the Mantissa (cont)

- Suppose we want to represent  $\pi$ :  
3.1415926535897932384626433832795.....
- That means that we can only represent it as:  
3.141592 (if we truncate)  
3.141593 (if we round)

14

### Representing the Exponent

- The exponent is represented as *excess-127*. E.g.,

Actual Exponent		Stored Value
-127	↔	00000000
-126	↔	00000001
		...
0	↔	01111111
+1	↔	10000000
		...
<i>i</i>	↔	$(i+127)_2$
		...
+128	↔	11111111

15

### Representing the Exponent (cont)

- The IEEE standard restricts exponents to the range:  
 $-126 \leq \text{exponent} \leq +127$
- The exponents  $-127$  and  $+128$  have special meanings:
  - If exponent =  $-127$ , the stored value is 0
  - If exponent =  $128$ , the stored value is  $\infty$

16

### Representing Numbers -- Example 1

What is 01011011 (8-bit machine) ?

0 101 1011  
sign exp mantissa

- Mantissa: 1.1011
- Exponent (excess-3 format):  $5-3=2$

$$1.1011 \times 2^2 \Rightarrow 110.11$$

$$110.11_2 = 2^2 + 2^1 + 2^{-1} + 2^{-2}$$

$$= 4 + 2 + 0.5 + 0.25 = 6.75$$

17

### Representing Numbers -- Example 2

Represent -10.375 (32-bit machine)

$$10.375_{10} = 10 + 0.25 + 0.125$$

$$= 2^3 + 2^1 + 2^{-2} + 2^{-3}$$

$$= 1010.011_2 \Rightarrow 1.010011_2 \times 2^3$$

- Sign: 1
- Mantissa: 010011
- Exponent (excess-127 format):  
 $3+127 = 130_{10} = 10000010_2$

1 10000010 010011000000000000000000

18

### Floating Point Overflow

- Floating point representations can overflow, e.g.,

$$\begin{array}{r} 1.111111 \times 2^{127} \\ + 1.111111 \times 2^{127} \\ \hline \end{array}$$

$$11.111110 \times 2^{127}$$

$$1.1111110 \times 2^{128} = \infty$$

19

### Floating Point Underflow

- Floating point numbers can also get too *small*, e.g.,

$$\begin{array}{r} 10.010000 \times 2^{-126} \\ \div 11.000000 \times 2^0 \\ \hline \end{array}$$

$$0.110000 \times 2^{-126}$$

$$1.100000 \times 2^{-127} = 0$$

20

### Floating Point Addition

Five steps to add two floating point numbers:

- Express the numbers with the same exponent (*denormalize*)
- Add the mantissas
- Adjust the mantissa to one digit/bit before the point (*renormalize*)
- Round or truncate to required precision
- Check for overflow/underflow

21

### Floating Point Addition -- Example 1 (Assume precision 4 decimal digits)

$$x = 9.876 \times 10^7$$

$$y = 1.357 \times 10^6$$

22

### Floating Point Addition -- Example 1 (cont) (Assume precision 4 decimal digits)

- Use the same exponents:

$$x = 9.876 \times 10^7$$

$$y = 0.1357 \times 10^7$$

23

### Floating Point Addition -- Example 1 (cont) (Assume precision 4 decimal digits)

- Add the mantissas:

$$x = 9.876 \times 10^7$$

$$y = 0.136 \times 10^7$$

$$\begin{array}{r} x+y = 10.012 \times 10^7 \end{array}$$

24

Floating Point Addition -- Example 1 (cont)  
(Assume precision 4 decimal digits)

3. Renormalize the sum:

$$x = 9.876 \times 10^7$$

$$y = 0.136 \times 10^7$$

---

$$x+y = 1.0012 \times 10^8$$

25

Floating Point Addition -- Example 1 (cont)  
(Assume precision 4 decimal digits)

4. Truncate or round:

$$x = 9.876 \times 10^7$$

$$y = 0.136 \times 10^7$$

---

$$x+y = 1.001 \times 10^8$$

26

Floating Point Addition -- Example 1 (cont)  
(Assume precision 4 decimal digits)

5. Check overflow and underflow:

$$x = 9.876 \times 10^7$$

$$y = 0.136 \times 10^7$$

---

$$x+y = 1.001 \times 10^8$$



27

Floating Point Addition -- Example 2  
(Assume precision 4 decimal digits)

$$x = 3.506 \times 10^{-5}$$

$$y = -3.497 \times 10^{-5}$$

28

Floating Point Addition -- Example 2 (cont)  
(Assume precision 4 decimal digits)

1. Use the same exponents:

$$x = 3.506 \times 10^{-5}$$

$$y = -3.497 \times 10^{-5}$$

29

Floating Point Addition -- Example 2 (cont)  
(Assume precision 4 decimal digits)

2. Add the mantissas:

$$x = 3.506 \times 10^{-5}$$

$$y = -3.497 \times 10^{-5}$$

---

$$x+y = 0.009 \times 10^{-5}$$

30

Floating Point Addition -- Example 2 (cont)  
 (Assume precision 4 decimal digits)

3. Renormalize the sum:

$$\begin{array}{r} x = 3.506 \times 10^{-5} \\ y = -3.497 \times 10^{-5} \\ \hline x+y = 9.000 \times 10^{-8} \end{array}$$

31

Floating Point Addition -- Example 2 (cont)  
 (Assume precision 4 decimal digits)


4. Truncate or round:

$$\begin{array}{r} x = 3.506 \times 10^{-5} \\ y = -3.497 \times 10^{-5} \\ \hline x+y = 9.000 \times 10^{-8} \text{ (no change)} \end{array}$$

32

Floating Point Addition -- Example 2 (cont)  
 (Assume precision 4 decimal digits)

5. Check overflow and underflow:

$$\begin{array}{r} x = 3.506 \times 10^{-5} \\ y = -3.497 \times 10^{-5} \\ \hline x+y = 9.000 \times 10^{-8} \end{array}$$


33

### Limitations

- Floating-point representations only approximate real numbers
- The normal laws of arithmetic don't always hold, e.g., associativity is *not* guaranteed

34

### Limitations -- Example

(Assume precision 4 decimal digits)

$$x = 3.002 \times 10^3$$

$$y = -3.000 \times 10^3$$

$$z = 6.531 \times 10^0$$

35

### Limitations -- Example (cont)

(Assume precision 4 decimal digits)

$$x = 3.002 \times 10^3$$

$$x+y = 2.000 \times 10^0$$

$$y = -3.000 \times 10^3$$

$$z = 6.531 \times 10^0$$

36

**Limitations -- Example (cont)**  
 (Assume precision 4 decimal digits)

$$x = 3.002 \times 10^3$$

$$x+y = 2.000 \times 10^0$$

$$y = -3.000 \times 10^3$$

$$(x+y)+z = 8.531 \times 10^0$$

$$z = 6.531 \times 10^0$$

37

**Limitations -- Example (cont)**  
 (Assume precision 4 decimal digits)

$$x = 3.002 \times 10^3$$

$$y = -3.000 \times 10^3$$

$$z = 6.531 \times 10^0$$

38

**Limitations -- Example (cont)**  
 (Assume precision 4 decimal digits)

$$x = 3.002 \times 10^3$$

$$y = -3.000 \times 10^3$$

$$y+z = -2.993 \times 10^3$$

$$z = 6.531 \times 10^0$$

39

**Limitations -- Example (cont)**  
 (Assume precision 4 decimal digits)

$$x = 3.002 \times 10^3$$

$$x+(y+z) = 0.009 \times 10^3$$

$$y = -3.000 \times 10^3$$

$$y+z = -2.993 \times 10^3$$

$$z = 6.531 \times 10^0$$

40

**Limitations -- Example (cont)**  
 (Assume precision 4 decimal digits)

$$x = 3.002 \times 10^3$$

$$x+(y+z) = 9.000 \times 10^0$$

$$y = -3.000 \times 10^3$$

$$y+z = -2.993 \times 10^3$$

$$z = 6.531 \times 10^0$$

41

**Limitations -- Example (cont)**  
 (Assume precision 4 decimal digits)

$$x = 3.002 \times 10^3$$

$$x+(y+z) = 9.000 \times 10^0$$

$$y = -3.000 \times 10^3$$

$$(x+y)+z = 8.531 \times 10^0$$

$$z = 6.531 \times 10^0$$

42

### Limitations -- Exercise Laws of Arithmetic

- Consider the laws of arithmetic:
  - Commutativity (additive and multiplicative)
  - Associativity
  - Distributivity
  - Identity (additive and multiplicative)
- Try to work out which ones *always* hold for floating-point numbers

43

### Reading (for the *Very Keen*)

- Goldberg, D., *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, ACM Computing Surveys, Vol.23, No.1, March 1991
- Knuth, D.E., *The Art of Computer Programming* (Vol 2) -- Seminumerical Algorithms, Section 4.4, pp. 319-329 (ed 3)

44