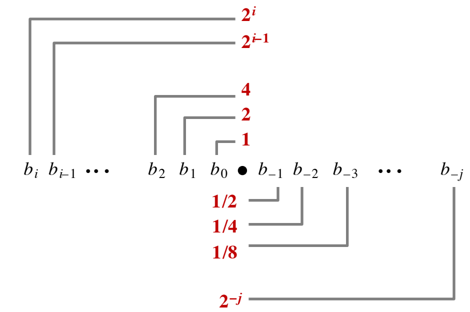
Floating-point numbers

Fractional binary numbers
IEEE floating-point standard
Floating-point operations and rounding
Lessons for programmers

Many more details we will skip (it's a 58-page standard...)
See CSAPP 2.4 for a little more

Fractional Binary Numbers



$$\sum_{k=-j}^{i} b_k \cdot 2^k$$

Fractional Binary Numbers

Value	Representation
5 and 3/4	101.11_2
2 and 7/8	10.111_2
47/64	0.101111_2

Observations

Shift left =
Shift right =
Numbers of the form $0.111111\ldots_2$ are...?

Limitations:

Exact representation possible only for numbers of the form $x * 2^y$, where x and y are integers.
Other rationals have repeating bit representations
$1/3 = 0.333333\ldots_{10} = 0.01010101[01]\ldots_2$

Fixed-Point Representation

Implied binary point. Example:

b_7 b_6 b_5 b_4 b_3 [.] b_2 b_1 b_0

Same hardware as for integer arithmetic.

b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 [.]

Fixed point = fixed *range* and fixed *precision*

range: difference between largest and smallest representable numbers
precision: smallest difference between any two representable numbers

IEEE Floating Point

Analogous to scientific notation

12 000 000	1.2×10^{7}	1.2e7
0.000 001 2	1.2×10^{-6}	1.2e-6

IEEE Standard 754 used by all major CPUs today

IEEE = Institute of Electrical and Electronics Engineers

Driven by numerical concerns

Rounding, overflow, underflow

Numerically well-behaved, but hard to make fast in hardware

5

Floating Point Representation

Numerical form:

$$V_{10} = (-1)^{S} * M * 2^{E}$$

Sign bit ***s*** determines whether number is negative or positive

Significand (mantissa) ***M*** normally a fractional value in range [1.0,2.0)

Exponent ***E*** weights value by a (possibly negative) power of two

Representation:

MSB `s` = sign bit *s*

`exp` field encodes ***E*** (but is *not equal* to E)

`frac` field encodes ***M*** (but is *not equal* to M)

s	exp	frac

6

Precisions

Single precision `(float)`**: 32 bits**

s	exp	frac
1 bit	8 bits	23 bits

Double precision `(double)`**: 64 bits**

s	exp	frac
1 bit	11 bits	52 bits

Finite representation of infinite range:

Not all values can be represented exactly.

Some are approximated.

7

Normalization and Special Values

$V = (-1)^{S} * M * 2^{E}$

s	exp	frac

"Normalized" = M has the form 1.xxxxx

As in scientific notation

$0.011 \times 2^{5} = 1.1 \times 2^{3}$, latter is more compact

Do not store the (guaranteed) leading 1.

Special values: **(How do we represent 0.0? 1.0/0.0?)**

zero:	`s` == 0	`exp` == 00...0	`frac` == 00...0
+inf, -inf:		`exp` == 11...1	`frac` == 00...0

1.0/0.0 = –1.0/–0.0 = +inf, 1.0/–0.0 = –1.0/0.0 = -inf

NaN ("Not a Number"):	`exp` == 11...1	`frac` != 00...0

sqrt(-1), $\infty - \infty$, $\infty * 0$, etc.

Denormalized/subnormal values (near 0.0) not covered here.

8

Floating Point Arithmetic

$V = (-1)^{S} * M * 2^{E}$

s	exp	frac

```
double  x = ..., y = ...;
double  z = x + y;
```

1. **Compute exact result.**
2. **Round,** to fit:

   Overflow exponent if it is too wide for **`exp`**.

   Drop LSBs of significand if it is too wide for **`frac`**.

   Underflow if nearest representable value is 0.

   ...

9

Lessons for programmers

$V = (-1)^{S} * M * 2^{E}$

s	exp	frac

`float` ≠ **real number** ≠ `double`

Rounding breaks associativity and other properties.

```
double  a = ..., b = ...;
...
if (a == b) ...

if (abs(a - b) < epsilon) ...
```

More shortly...

10