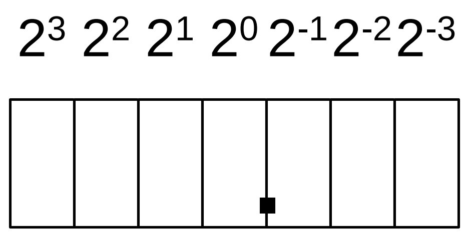
Extra Lecture Notes
Floating Point Numbers

Floating Point (a brief look)

- We need a way to represent
  - numbers with fractions, e.g., 3.14159265358979323846 2642
  - very small numbers, e.g., .000000001
  - very large numbers, e.g., 3.15576×10^9
- Numbers with fractions:



Big and small numbers

- Solution for decimal - scientific notation
  - $2.5 \times 10^{15} = 2{,}500{,}000{,}000{,}000{,}000$
- Can do the same for binary:
  - $2\text{MB} = 2 \times 2^{20}$ or $10_2 \times 10_2^{10100}$
  - This is called a *floating-point number*
  - In general, composed of sign, exponent, significand:

    $(-1)^{\text{sign}} \times \text{ significand } \times 2^{\text{exponent}}$
  - more bits for significand gives more accuracy
  - more bits for exponent increases range
- IEEE 754 floating point standard:
  - single precision: 8 bit exponent, 23 bit significand
  - double precision: 11 bit exponent, 52 bit significand

IEEE 754 floating-point standard

- Leading “1” bit of significand is implicit
- Exponent is “biased” to make sorting easier
  - Biasing is a way of representing negative numbers
  - All 0s is smallest exponent all 1s is largest
  - bias of 127 for single precision and 1023 for double precision
  - summary: $(-1)^{\text{sign}} \times (1+\text{significand}) \times 2^{\text{exponent} - \text{bias}}$
- Example:
  - decimal: $-.75 = -3/4 = -3/2^2$
  - binary: $-.11 = -1.1 \times 2^{-1}$
  - floating point: exponent = 126 = 01111110
  - IEEE single precision: 10111111010000000000000000000000

Examples of Floating Point Numbers

Show the IEEE 754 binary representation for the number 20.0 in single and double precision:

$$20 = 10100 \text{ x } 2^0 \text{ or } 1.0100 \text{ x } 2^4$$

Single Precision:

The exponent is 127+4 = 131 = 128 + 3 = 10000011

The entire number is

0 1000 0011 0100 0000 0000 0000 0000 000

Double Precision:

The exponent is 1023+4 =1027=1024 + 3 =10000000011

The entire number is:

0 1000 0000 011 0100 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000

Examples of Floating Point Numbers

Show the IEEE 754 binary representation for the number 20.5_{10} in single and double precision:

$20.0 = 10100 \times 2^0$, $0.5 = 0.1 \times 2^0$ together

1.01001×2^4

Single Precision:

The exponent is 127+4 = 131 = 128 + 3 = 10000011

The entire number is

0 1000 0011 0100 1000 0000 0000 0000 000

Double Precision:

The exponent is 1023+4 =1027=1024 + 3 =10000000011

The entire number is:

0 1000 0000 011 0100 1000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000

Examples of Floating Point Numbers

Show the IEEE 754 binary representation for the number -0.1_{10} in single and double precision:

$0.1 = 0.0\underline{0011} \times 2^{0}$ or $1.1\underline{0011} \times 2^{-4}$

(the $\underline{0011}$ pattern is repeated)

Single Precision:

The exponent is 127-4 = 123 = 0111 1011

The entire number is

1 0111 1011 1001 1001 1001 1001 1001 100

Double Precision:

The exponent is 1023-4 = 1019 = 01111111011

The entire number is:

1 0111 1111 011 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001

Floating Point Complexities

- Operations are somewhat more complicated (see text)
- In addition to overflow we can have “underflow”
  - Result of two adding two very small values becomes zero
- Accuracy can be a big problem
  - 1/(1/3) should = 3
  - IEEE 754 keeps two extra bits, guard and round
  - four rounding modes
  - positive divided by zero yields “infinity”
  - zero divide by zero yields “not a number” (NaN)
- Implementing the standard can be tricky
- Not using the standard can be even worse