Floating Point Representation

September 13, 2000

CSC201 Section 002

Fall, 2000

Floating Point

- The computer representation for real numbers
- Hardware support on most processors (special instructions)
- Advantages
  - much greater range
  - uniform precision

Representing (Unsigned) Fractions in Binary

- Converting binary to decimal
  - each binary digit is weighted by a (negative) power of two

2^i	As a ratio	As a decimal fraction
2^{-1}	1/2	.5
2^{-2}	1/4	.25
2^{-3}	1/8	.125
2^{-4}	1/16	.0625

Example:

$.1011_2 =$

$1 * 2^{-1} + 0*2^{-2} + 1*2^{-3} + 1*2^{-4} =$

$.5 + .125 + .0625 =$

$.6875$

Binary Fractions (cont.)

- Converting decimal fraction to binary fraction
  - multiply the fractional part by two repeatedly
  - the integer portion at each step is the next binary digit, from msb to lsb
  - stop when the fraction = 0

Example:

convert .3125_10 to binary

.3125 * 2	=	0	+	.625
.625 * 2	=	1	+	.25
.25 * 2	=	0	+	.5
.5 * 2	=	1	+	0

(Binary) Scientific Notation

- A number is represented as $(-1)^s * m * b^e$, where…
  - s = sign bit
  - m = mantissa
  - b = base
  - e=exponent
- In the IEEE floating point standard…
  - b = 2 (implicit, not stored)
  - s = sign (1 bit)
  - m = signed magnitude binary representation (23 bits + implicit 24th)
  - e = biased-127 binary representation (8 bits), allowable range is -126 <= e <= +127

Examples (FPS)

S	E	M	Value (decimal)
1	-3	1.0110..0	$-1.375 * 2^{-3}$
0	6	1.0000..0	$+1.0 * 2^{6}$
0	-100	1.1000..0	$+1.5 * 2^{-100}$

Precision, Significant Digits, and Accuracy

- Precision = accuracy of a measurement
- Leading zeros don't count as digits of precision
  - Trailing zeros may or may not be digits of precision
- Goal for arithmetic: maintain maximum precision at all times
- Single and double precision

How many digits of (decimal) precision?

198

19.8

001980

1980.00

Normalization

- Mantissa is (almost) always stored so that it is in the range 1.0 <= m < 2
  - reason: preserve maximum precision at all times!
  - Normalized: 1.110, 1.000, 1.111, 1.001
  - Not normalized: 0.110, 11.10, 10.01
- "Normalizing" means shifting mantissa to be in this form
  - for each shift right, add 1 to the exponent value
  - for each shift left, subtract 1 from the exponent value

Ex.: normalize $.00101_2 * 2^5$

$= 0.0101 * 2^4$

$= 0.101 * 2^3$

$= 1.01 * 2^2$

Normalizing (cont.)

Example: normalize $111.01_2 * 2^{-1}$

$= 11.101 * 2^{0}$

$= 1.1101 * 2^{1}$

- Exception: the value zero can never be normalized
- Since the leftmost bit is (almost) always 1, not explicitly stored