

More on Data Representation

Computers in Engineering

Prof. Dan Thedens

Fall 2002

Signed Integers

- **Signed-magnitude format**
 - possible to include negative integers
 - 0 in the MSB (leftmost) indicates a nonnegative number
 - 1 in the MSB indicates a negative number
- **Two other methods used:**
 - 1's complement
 - 2's complement
 - Makes the hardware easier to design

1's and 2's Complement

- **Number of bits is part of the definition**
- **2's complement of an integer x :**
 $2^n - x$
- **1's complement of an integer x :**
 $2^n - 1 - x$
- **The complement of the complement returns the original value x**
 - $2^n - (2^n - x) = x$
 - $2^n - 1 - (2^n - 1 - x) = x$

1's and 2's Complement

- **Binary 4-bit complement Representations**

Integer	2's Complement	1's Complement
-8	1000	—
-7	1001	1000
-6	1010	1001
-5	1011	1010
-4	1100	1011
-3	1101	1100
-2	1110	1101
-1	1111	1110
-0	—	1111

Complement Example

- **1's complement found by complementing the individual bits**
- **Ex. if $x = 00101110$**
$$\begin{array}{r} 11111111 \\ -00101110 \\ \hline 11010001 \end{array} = \text{1's complement}$$
- **2's complement found by adding 1 to the 1's complement:**
 $11010001 + 1 = 11010010$

Overflow

- **Occurs when the result of an arithmetic operation lies outside of the format being used**
- **If no overflow occurs, the addition of 2's complement numbers can be done by:**
 - Adding the integers as binary numbers
 - Ignoring the carry from the high order bit

Range, Resolution and Error

- **Can reduce possibility of overflow by:**
 - Increasing the range by increasing n (# of bits)
 - Increase range without increasing n
 - * But this introduces problem with resolution and error
- **Resolution:** distance between consecutive numbers
- **Maximum Error:** half the distance between consecutive numbers

Scaling

- **Multiplying all of the numbers in a set by $1/s$ before storing and operating on them**
 - Equivalent to changing the units of physical quantities being operated on
- **If groups of n bits each are used to store number from a set of numbers that is evenly spaced by an amount s, then stored numbers are $1/s$ times the actual numbers**
 - E.g.

$$\frac{x}{s} + \frac{y}{s} = \frac{(x+y)}{s}$$

Chopping and Rounding

- **Methods for dealing with remainders**
- **Chopping**
 - Ignoring the remainder
 - * Biases the results
 - * Increases the average error
- **Rounding**

Real Numbers

- **Very large numbers may be needed for engineering calculations**
- **For decimal numbers, scientific notation uses multiplication by powers of 10**
 - Avoids leading and trailing 0's
 - Ex. 3, 433, 000 = 3.433×10^6
- **For different bases, can “vary the scale factor” to represent floating point numbers**

Floating Point Format

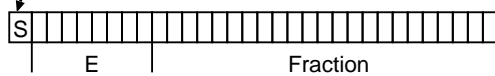
- **Includes two signed numbers:**
 - Significand, or mantissa: the multiplier
 - Exponent: the exponent of the base
 - E.g. -1.01011×2^2
- **Things to consider with floating point format:**
 - Base
 - Resolution needed to provide desired accuracy
 - Range of numbers to be represented

IEEE Floating Point Format

- **Standard format for:**
 - Single precision
 - Double precision
- **Assume that the number is represented in the form:**
 $(-1)^S \times 1.\text{fraction} \times 2^{E-N}$, where S is the sign bit, E is the exponent biased by N
 - N = 127 for single precision
 - N = 1023 for double precision

IEEE Floating Point Format Single Precision

Sign of mantissa
1 = negative
0 = positive



(excess 127)

1 bit 8 bits 23 Bits

$$\text{Number} = (-1)^S \times 1.\text{fraction} \times 2^{E-127} \quad 0 < E < 255$$

Number with $E = 0$ is reserved for 0, and $E = 255$ for numbers outside that range

IEEE Floating Point Format Double Precision

Sign of mantissa
1 = negative
0 = positive



(excess 1023)

1 bit 11 bits 52 Bits

$$\text{Number} = (-1)^S \times 1.\text{fraction} \times 2^{E-1023} \quad 0 < E < 2047$$

Number with $E = 0$ is reserved for 0, and $E = 2047$ for numbers outside that range

IEEE Floating Point Format

- (See picture)
- Example:
- $-5.375 = -101.011 = -1.01011 \times 2^2$
- For IEEE format:
 - Sign bit = 1
 - $E = 2 + 127 = 129 = 10000001$
 - Mantissa = 1.01011
 - Fraction = 01011
- Normalized so that most significant bit is 1.

IEEE Floating Point Format

- Complete representation is:
 $11000000101011000000000000000000$
 sign bit E Fraction
 1 10000001 010110000000000000000000
- Note: the 1 in the mantissa is assumed
- How would you determine the components of the number:
 $001111100110000000000000000000$
 - Sign is ?, Exponent = ?, Fraction = ?, Mantissa = ?

IEEE Floating Point Example

- Sign is positive
- Exponent = $124 - 127 = -3$
 - 01111100 is 124
- Fraction = 0.11
- Mantissa is 1.11
- Number is $1.11 \times 2^{-3} = 0.00111 = 0.21875$
 - Note: Adding 127 to exponent makes this message of storage excess-127 format which allows exponent operations be unsigned

Floating Point Arithmetic

- Addition and Subtraction
 1. Align mantissas, (or fractions)
 2. Add/Subtract mantissas
 3. Re-normalize result mantissa and adjust exponent
- Multiplication and Division
 1. Align mantissas, (or fractions)
 2. Multiply/Divide mantissas
 3. Re-normalize result mantissa and adjust exponent