Tutorial 1: Number Representations

ECSE 322: Computer Engineering

Winter 2008

TA: Robert Rolnick
E-mail: Robert.Rolnick@mail.mcgill.ca

Tutorial 1: Number Representations

1. Number Systems
2. USASI
3. IEEE 754

What is a Number System?

- A number system maps number (integers, reals) to bits.
- In Ecse 221 (ICE) you saw how to represent integers. (E.g. $010110_2 = 22_{10}$)
- How do you represent real numbers using bits? (E.g ¾, 1.232, $1.78 * 10^{23}$, etc.)

Representing Real Numbers

- Balancing the precision and range of real numbers is difficult.
- In scientific notation precision is the number of significant figures in your calculation, and range would be the possible exponents.
- Also, we need a bit to represent the sign of your number.
- The solution is to divide a number into 3 pieces: sign, exponent, and mantissa.
- The sign needs only 1 bit, but the number of bits used for the exponent and mantissa will determine the precision, accuracy, and range of your representation.
- In this course, we will be typically dealing with 32-bit representations, although virtually all consumer PCs on the market today have shifted to 64-bit processors.

The USASI Representation

- 1-bit used for sign (0 = positive, 1 = negative.)
- 7-bits used for exponent, which is stored with a bias of +64.
- 16 is the base of the exponent.
- 24-bits used for the mantissa.
- Hidden bit normalization is **<u>NOT</u>** used.

32-bits		
Sign (1-bit)	Exponent (7-bits)	Mantissa (24-bits)

An Example of USASI Conversion

Convert: -204.5625 into its USASI form

- Step 1: Check sign. It is negative, so the sign bit is 1.
- Step 2: Convert your number to binary. $204.5625_{10} = 11001100.1001_2$
- Step 3: Shift (four bits at a time) until all bits are left of the decimal point.
  $0.110011001001 \ * \ 16^2$
- Step 4: Pad your result, to get the 24-bit mantissa. In our example we get:
  110011001001000000000000
- Step 5: Get exponent by adding the bias (+64) to the power of 16. In our example we get: 2 + 64 = 66.
- Step 6: Convert your exponent to binary ensure that it is 7 bits. $66_{10} = 1000010_2$
- Step 7: Combine to get our final result

1	1000010	110011001001000000000000

Key Points of the USASI Representation

- No way to represent NaN or infinity.
- Zero is represented by setting all the mantissa bits to zero. (Exponent can be anything.)
- The mantissa can start with up to 3 zeroes.
- Largest possible number: $01111111111111111111111111111111_2$.
  This is equivalent to: $(1-2^{-24}) * 16^{(127-64)} \approx 0.99999994 * 16^{63} \approx 7.237 * 10^{75}$
- Smallest possible number: $00000000000100000000000000000000_2$.
  This is equivalent to: $(1-2^{-4}) * 16^{(0-64)} = 0.9375 * 16^{-64} \approx 8.0964 * 10^{-78}$
- Maximum relative error: $2^{-25} / (2^{-4} + 2^{-25}) \approx 2^{-21}$
- Dynamic Range: $16^{63} / (16^{-1} * 16^{-64}) = 16^{63} / 16^{-65} = 16^{128} = (2^4)^{128} = 2^{512}$

The IEEE 754 Representation

- 1-bit used for sign (0 = positive, 1 = negative.)
- 8-bits used for exponent, which is stored with a bias of +127.
- 2 is the base of the exponent.
- 23-bits used for the mantissa.
- Hidden bit normalization is used.

32-bits		
Sign (1-bit)	Exponent (8-bits)	Mantissa (23-bits)

An Example of IEEE 754 Conversion

Convert: $1100\ 0010\ 1100\ 1100\ 1001\ 0000\ 0000\ 0000_2$ from IEEE 754 into its base-10 form

- Step 1: Regroup terms into sign, exponent, and mantissa.
  Sign: 1 (so negative)
  Exponent: 10000101
  Mantissa: 10011001001000000000000

- Step 2: Convert exponent to base 10, and remove the bias.
  $10000101_2 = 133_{10}$
  $133 - 127 = 6$

- Step 3: Compute mantissa in base 10, and add 1.
  $1.10011001001 = 1 + 2^{-1} + 2^{-4} + 2^{-5} + 2^{-8} + 2^{-11} = 1.59814453125_{10}$

- Step 4: Combine the above to get $-1.59814453125 * 2^6 = -102.28125$

- *NOTE: The binary value provided is the same one that was calculated in the USASI example! Notice how, using IEEE 754, it converts back to a different decimal value!*

Key Points of the IEEE 754 Representation

- NaN is represented by an exponent of all 1s, and a non-zero mantissa.
- ±∞ is represented by an exponent of all 1s, a mantissa of all 0s, and sign acts normally.
- Zero is represented by both exponent and mantissa being all 0s. Sign doesn't matter. Consequently, there are two distinct zeroes that compare as equal.
- Denormalized numbers (numbers without the hidden bit) are used when the exponent is set to all-zeroes. The mantissa can be anything. Thus zero is just a special case of a denormalized number.
- Largest possible number: $01111111011111111111111111111111_2$.
  This is equivalent to: $(2-2^{-23}) * 2^{(254-127)} = 1.999999881 * 2^{127} \approx 3.4028 * 10^{38}$
- Smallest possible number: $00000000100000000000000000000000_2$.
  This is equivalent to: $1 * 2^{(1-127)} = 2^{-126} \approx 1.1755 * 10^{-38}$
- Maximum relative error: $2^{-25} / (2^{-4} + 2^{-25}) \approx 2^{-21}$
- Dynamic range: $2^{127}/2^{-126} = 2^{254}$