COMPSCI 105 – Lecture #2
Numbers and the Computer

(The First of Three Math-Heavy Lectures)

Definition: BIT

- Binary Digit
- Smallest possible unit of information
- Two values only: 0 or 1
- Represent a single Yes or No question
- Can encode any two-valued system
  - Yes/No, True/False, Up/Down, On/Off, In/Out, etc.
- Easy to build hardware to encode bits.

Bits and Patterns

- 1 Bit gives 2^1 = 2 patterns: 0 or 1
- 2 Bits gives 2^2 = 4 patterns: 00, 01, 10, 11
- 3 Bits gives 2^3 = 8 patterns: 000, 001, 010, 011, 100, 101, 110, 111
- Each new bit doubles the number of patterns
- Therefore: N Bits gives 2^N Distinct patterns.
- Those patterns can be interpreted as either unsigned or signed decimal numbers.

What About 8 Bits?

Unsigned	Signed
00000000 = 0	00000000 = +0 (zero is positive!)
00000001 = 1	00000001 = +1
00000010 = 2	00000010 = +2
00000011 = 3	00000011 = +3
00000100 = 4	00000100 = +4
…	… (all positives start with 0)
01111110 = 126	01111110 = +126
01111111 = 127	01111111 = +127 (biggest positive)
10000000 = 128	10000000 = -128 (smallest negative)
10000001 = 129	10000001 = -127
…	… (all negatives start with 1)
11111100 = 252	11111100 = -4
11111101 = 253	11111101 = -3
11111110 = 254	11111110 = -2
11111111 = 255	11111111 = -1

Definition: Byte

- Packet of 8 Bits (French word is "octet")
- Typical unit of computer memory / storage
- Used to represent one standard character
- Values range from 00000000 … 11111111
- 2^8=256 Distinct patterns
- Can encode any integer from 0 through 255
- Can also encode any integer from -128 through +127 (and zero is treated as positive)

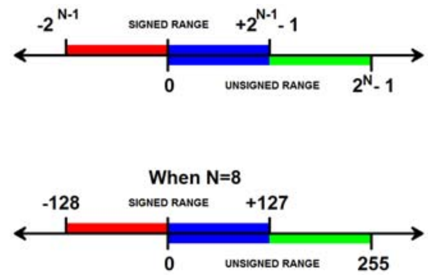
Unsigned Integers

- Pick storage size of N bits (8, 16, 32, 64, etc.)…
- …therefore 2^N distinct patterns are available.
- Smallest value is all zeroes (decimal value 0),
- Largest value is therefore 2^N-1.
- Results less than zero are "underflow" errors,
- Results greater than max are "overflow" errors
- Each computer architecture has a fixed N.

Signed Integers

- Pick N, there are still 2^N patterns.
- Consider half the patterns to be negative.
  (Half of 2^N is $2^N/2 = 2^{N-1}$)
- The other half of the patterns are zero and above.
- Zero is considered to be positive.
- Signed range is therefore -2^{N-1} … $+2^{N-1}-1$.

Signed vs. Unsigned Integers



Example for N=8

- 2^8 = 256 patterns
- Unsigned Range
  - Minimum: 0
  - Maximum: $2^8-1 = 255$
- Signed Range
  - Minimum: $-2^{8-1} = -2^7 = -128$
  - Maximum: $+2^{8-1}-1 = +2^7-1 = +128-1 = +127$

Example for N=16

- 2^{16} = 65536 patterns
- Unsigned Range
  - Minimum: 0
  - Maximum: $2^{16}-1 = 65535$
- Signed Range
  - Minimum: $-2^{16-1} = -2^{15} = -32768$
  - Maximum: $+2^{16-1}-1 = +2^{15}-1 = +32768-1 = +32767$

Example for N=32

- 2^{32} = 4,294,967,296 patterns
- Unsigned Range
  - Minimum: 0
  - Maximum: $2^{32}-1$ = 4,294,967,295
- Signed Range
  - Minimum: $-2^{32-1} = -2^{31}$ = -2,147,483,648
  - Maximum: $+2^{32-1}-1 = +2^{31}-1$ = +2,147,483,647
  - Nine (and a little more) significant digits

What about Real numbers?

- Approaches:
  - Rational (ratio of integers)
  - Fixed-Point
  - Floating-Point
- All require re-interpreting how bits are used.
- All have both good and bad attributes.
- All have been successfully used in real tools.
- Floating-Point is dominant today.

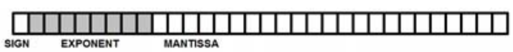
Rational Numbers

- For N bits, divide into two $^{N}/_{2}$ bit sections:
  - First section is numerator
  - Second section is denominator
- Numbers like $^{1}/_{3}$, $^{1}/_{2}$, $^{3}/_{7}$, $^{1}/_{10}$, $^{355}/_{113}$ are easy
- Reduce to lowest form (e.g., $^{2}/_{4}$ and $^{3}/_{6}$ go to $^{1}/_{2}$)
- Not very efficient:
  - low information density (many redundant patterns)
  - Not efficient use of bits (N has to be *very* big)

Fixed-Point Numbers

- Set virtual decimal point to middle of bits:
  - Half the bits are integer
  - Half the bits are fraction
- All bit patterns are useful
- Easy to add, subtract, multiply, divide in binary
- Trades off range of values for fraction support.
  - For N=32 (16-bit signed integer, 16-bit fraction), maximum signed value is only +32767.9999847412…
- Still not an efficient use of bits

Floating-Point Numbers



- Binary version of Scientific Notation
  - Decimal: -3.4024×10^{15} or $+1.732\times10^{-6}$ or $3\times10^{0} = 3$
  - Binary: $+1.00101001\times2^{101}$ (which in decimal is +37⅛)
- Use one bit for *sign* (0=plus, 1=minus)
- Use some of the N bits for *exponent*
- Use remaining bits for *mantissa* (significand)
- Trades off <u>precision</u> for <u>dynamic range</u>
- Efficient use of bits (very few unused patterns)
- Most software today uses floating-point

Floating-Point Precision

- **Single Precision**
  - N=32 bits (1 sign, 8 exponent, 23 mantissa)
  - Dynamic Range: $\pm10^{\pm38}$
  - Significant Figures: only 5-6 Decimal Digits (Remember that 32 bit integers have about 9 sig. figs.)
- **Double Precision** (Used by Excel & Python)
  - N=64 bits (1 sign, 11 exponent, 52 mantissa)
  - Dynamic Range $\pm10^{\pm308}$
  - Significant Figures: 15-16 Decimal Digits

But long fractions get rounded off:

- Expected loss of precision:
  - Numbers with naturally long but finite fractions,
  - Rationals that repeat forever (⅓ = 0.333333333…),
  - Irrationals (e, π, ϕ, √2, √3, √5, etc.).
- <u>Unexpected</u> loss of precision: Many well-behaved fractions in decimal are ill-behaved in binary ($^{1}/_{10}$ = 0.00011001100110011…)

Aside: Proof that √2 is Irrational

- √2 = 1.414213562…
- Remember:
  - Even × Even = Even (4 × 6 = 24)
  - Even × Odd = Even (4 × 7 = 28)
  - Odd × Odd = Odd (5 × 7 = 35)
- Assume √2 is Rational: √2 = $^{P}/_{Q}$
- Assume Lowest Form: P, Q aren't <u>both</u> even
  - (if both were even, we can repeatedly divide both P and Q by 2 until at least one is odd)

Aside: Proof that $\sqrt{2}$ is Irrational

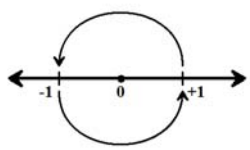
- Starting assumption: $\sqrt{2} = {}^{P}\!/_{Q}$ (P, Q not both even)
- Square both sides: $2 = P^2 / Q^2$
- Multiply by Q^2: $2Q^2 = P^2$
- Conclusion #1: P^2 is even, thus P is even
- Divide by 2: $Q^2 = P^2 / 2 = P \times {}^{P}\!/_{2}$
- Conclusion #2: Q^2 is even, thus Q is even
- Contradiction:
  - Initial assertion was P, Q aren't both even, proof says both are even, thus assumption that $\sqrt{2} = {}^{P}\!/_{Q}$ is false. No such rational number exists.

The Biggest Dirty Secret of Computing

- Most of the interesting numbers in the Universe are irrational,
- Numbers on computers have a fixed and finite number of bits,
- Therefore, most values get rounded off.
- **Most numerical results are approximations.**
- More bits means more precision, but only forestalls and does not eliminate the problem.

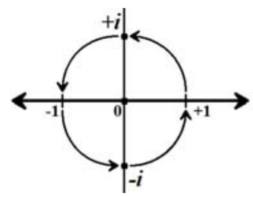
Complex Numbers

- The Real number line extends from -∞ to +∞,
- Use of space above and below the line gives us more computational expressive power.
- Negation then becomes a rotation of 180°:



Complex Numbers

- Rotation of +1 by 90° leaves it in space above the zero center. Call that number *i*:



Complex Numbers

- Multiplying a number by *i* twice equals negation (two rotations of 90°),
- Thus $i \times i = i^2 = -1$, and therefore $i = \sqrt{-1}$
- You can't take the square root of negative numbers, right?
- Well, here we can, sort of, but to do this…
- … *i* is called "imaginary"

Complex Numbers

- A complex number is then a pair of numbers:
  - A value along the Real axis,
  - A value along the Imaginary axis.
  - Written with the Real part first, then Imaginary.
- Examples:
  - $2+3i$, $5-7i$, $-3+2i$, $-4-6i$, $6.7+5.9i$, etc.
  - Just 7 by itself (the same as $7+0i$)

Complex Math

- Add/Subtract: treat components separately:
  - $2+6i\ +\ 5-2i\ =\ (2+5)+(6-2)i\ =\ 7+4i$
  - $2+6i\ -\ 5-2i\ =\ (2-5)+(6--2)i\ =\ -3+8i$
- Multiplication uses FOIL method:
  - $2+6i\ \times\ 5-2i\ =$
  - $(2\times 5)+(2\times -2i)+(6i\ \times\ 5)+(6i\ \times\ -2i)\ =$
  - $(10)\ +\ (-4i)\ +\ (30i)\ +\ (-12i^2)\ =$
  - $(10+12)\ +\ (-4+30)i\ =\ 22+26i$

Complex Math

- Division uses <u>two</u> complex multiplications to eliminate imaginary component in denominator,
- Multiply both numerator and denominator by <u>complex conjugate</u> of denominator (the same number with opposite sign on imaginary part).
- Example:
  - $22+26i\ \div\ 2+6i\ =$
  - Numerator: $22+26i\ \times\ 2-6i\ =\ 200-80i$
  - Denominator: $2+6i\ \times\ 2-6i\ =\ 4+36\ =\ 40$
  - $200-80i\ \div\ 40\ =\ 5-2i$

Complex Math

- Used in math, engineering, physics, etc.
- Supported by early language FORTRAN,
- Supported by modern language Python,
- Supported (badly) by Excel 2007 and later.
- Mostly Double-Precision Floats,
- Subject to same round-off errors as other floating-point numbers.