Class 03: Representing Other Types of Data

A. Why?

In addition to integer data, we need to manipulate floating-point numbers and text. Since long binary numbers are tedious to use, we'll often using octal and hexadecimal numbers.

B. Outcomes

By the end of the class you should

- Be able to say why we have floating-point numbers.
- Know in general how floating-point numbers are represented.
- Know how textual characters are represented.
- Know the critical piece of information you need to know if someone hands you some random bits and asks you "What does this represent?"

C. Non-Whole Numbers

- In decimal, non-whole numbers are represented using a decimal point and a fractional part that sums up negative powers of 10.
- In binary, non-whole numbers use a binary point and negative powers of 2.

D. Floating-Point Numbers

- True real numbers can take an infinite number of digits.
- We approximate them with a finite number of digits.
- Range vs precision of a set of numbers.
- Floating-point numbers trade range for precision.
- Based on scientific notation. (E.g. 6.02×10^{23})

E. IEEE Floating-Point Number Standard

- IEEE = Institute of Electrical & Electronics Engineers.
- For 32-bit floating-point numbers, 1 bit for sign, 8 bits for exponent E, 23 for fraction.
- If $1 \leq E < 255$, then the bit string represents $N = (-1)^S \times 1.\text{Fraction}_2 \times 2^{(E-127)}$
- If E = 0 and Fraction = 0 we have +0 or –0.
- If E = 0 and Fraction $\neq$ 0 we have $N = (-1)^S \times 0.\text{Fraction}_2 \times 2^{-126}$
- E = 255 used for $+\infty$, $-\infty$, and NaN (Not a Number).

F. Octal and Hexadecimal

- Bases 8 and 16 used to abbreviate bit strings.
- Octal → Binary: Replace each octal digit (0–7) by its 3-bit representation.
- Binary → Octal: Going right-to-left, replace each 3-bit sequence by equivalent octal digit. Add extra zeros if necessary.

- Hexadecimal uses 0–9 and A–F as digits: $A_{16} = 1100_2 = 10_{10}$, …, $F_{15} = 1111_2 = 15_{10}$.
- Hexadecimal → Binary: Replace each hex digit by its 4-bit representation.
- Binary → Hexadecimal: Going right-to-left, replace 4-bit sequences by equivalent hex digit. Add extra zeros if necessary.
- Decimal → and ← Octal or Hex: Similar to Decimal → or ← Binary. Multiply or divide by 8 or 16 instead of 2.
- Abbreviating 8, 16, or 32 bits easy with hex (no leftover bits).
  - Trickier with octal: leftmost octal digit won't be 5–7

G. Characters and Strings

- A character is represented by a bit string
  - ASCII and Unicode schemes
- Strings are sequences of characters (null-terminated or length-specified).

———— End of Mon, Aug 31, 2009 ————

H. Logical Operations

- Logical NOT, AND, OR, XOR, NAND, NOR
  - A XOR B = 1 iff A≠B [one is 1, the other is 0]
  - A NAND B = NOT(A AND B)
  - A NOR B = NOT (A OR B)
- Bitwise operations on bit string(s)
  - Bitwise NOT = 1's complement negative
  - 2's complement negative(X) = (NOT X) + 1
- Mask bits
  - X = X AND Y clears selected bits of X
  - X = X OR Y sets selected bits of X
  - Y is the mask

I. What Do Bits Represent?

- It depends on the context.