# Chapter 8

**Representing Information Digitally**

## Symbolic Representations

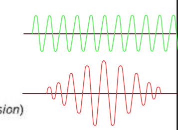
- Information: knowledge, facts and meaning
- Categories
  a. Numeric information (integers, fractions)
     - E.g. CSC101 grades, faculty salaries
  b. Textual information (keyboard characters)
     - E.g. CS16 class roster, term paper
  c. Visual information (pictures, graphics, animation)
     - E.g. digital camera picture, original corporate logo
  d. Audio information (music, speech)
     - E.g. song on CD, digitized inaugural address
  e. Instruction information (programs, recipes)
     - E.g. source code for MS Word, games, etc.

## Symbolic Representations

- Multimedia computing
  1. Seamless combination of these categories of information
  2. Note the gradual merging of technologies that deal with these different categories – TV, phone, books, mail, music, internet, etc.

## Symbolic Representation

- Common representation of data – both necessary and inevitable
- Two basic forms of information in the real world
  a. Discrete – precise, unambiguous, distinct, finite
     1) Text (and other finite symbol systems)
     2) Numbers with *finite precision*
     3) Instructions
  b. Analog – continuous, infinite
     1) Images, graphics, movies (?)
     2) Sounds
     3) The real number line (*unlimited precision*)



■**analog information is continuous, e.g., wave**

## Symbolic Representation

- Digital information – use symbols/numbers to represent <u>all</u> data
  a. A computer by nature is finite and discrete in terms of what it can store
  b. Digits are discrete, unambiguous (in theory any finite symbol set could be used)
  c. Precise, easier to store and transmit
  d. Ordering, replication, random and selective access
  e. Compression, content analysis & synthesis

■**computers process discrete or digital data**

## Symbolic Representation

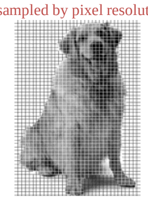
- How do we *digitize*?
  a. Discrete information – mapping of symbols
  b. Analog information – continuous & infinite, but must have a *discrete* representation
     1) Sampling – selecting a finite subset of data to represent the whole
     2) Quantizing – measuring the samples and assigning (possibly approximate) binary values for storage
     3) Precision vs. accuracy ("exactness")
     4) Possibilities for error
     5) More later…

■**the process of converting information to a binary form is called digitization**
■**both discrete and analog forms of information may be digitized**

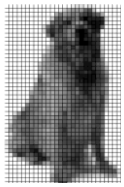
## Example: Digitizing Images

- The two step process for digitizing images:
  - sampling the continuous tone image for pixels
  - quantizing pixels

image is sampled by pixel resolution



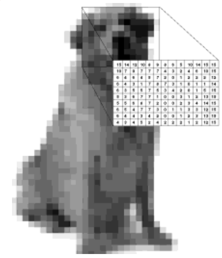
pixels samples are averaged



## Example: Digitizing Images

- The two step process for digitizing images:
  - sampling the continuous tone image for pixels
  - quantizing pixels

pixels are converted to numeric form



## Computer Representation

- Binary Technology
  - Binary System: Digits are ***0 1***
- True/False; On/Off; Yes/No; Male/Female
- 8 bits -- $2^8$ combinations
  - 256 unique Combinations
  - Limited range / overflow
- Hexadecimal (base 16)
  - Two Hex Digits: (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)
  - $16^2$ == 256 unique combinations

- binary numbering system is a base-2 positional numbering system
- binary digits are called "bits"
- bits are organized into groups, e.g., 8 = "byte"

## Problems: Representing Text on a computer

- Need: One unique binary symbol per character
  - About ≅95 characters
  - 6 bits?
  - 7 bits?
- Which binary pattern for which text symbol?
  - We must all agree
  - ASCII (*extended* to use 8-bits)
  - UNICODE
- Need some thought to "collating" sequence
- What about *formatting* codes?

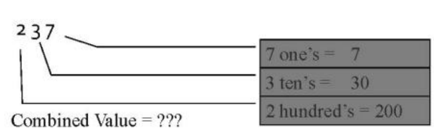
## Representing Numbers

- Integers
  - Whole numbers
  - "Counting" numbers
  - Ex: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, etc.

## Integers

- Recalling Decimal (Base 10) Numbers
  - Finite Representation (try 8 digits)
  - What is the maximum value using 8 digits?

    $10^7$ $10^6$ $10^5$ $10^4$ $10^3$ $10^2$ $10^1$ $10^0$
    - 9 9 9 9 9 9 9 9
  - Actual value is based on position of each digit



## Integers (Cont.)

- Binary (Base 2) Numbers
  - Finite Representation (8 digits)
  - What is the maximum value using 8 digits?

$2^7\ 2^6\ 2^5\ 2^4\ \ 2^3\ 2^2\ 2^1\ 2^0$

1 1 1 1 1 1 1 1

- Actual value is based on position of each digit

0000 1101

| | |
|---|---|
| 1 one = | 1 |
| 0 two's = | 0 |
| 1 four = | 4 |
| 1 eight = | 8 |

Maximum Byte Value= 128+64+32+16+8+4+2+1=255

## Integers (Cont.)

- Example Binary Numbers

$2^7\ 2^6\ 2^5\ 2^4\ \ 2^3\ 2^2\ 2^1\ 2^0$

1 1 1 0 1 1 0 1

- Powers of 2

- Actual value is based on position of each digit

0000 0011
0000 0001
0000 0010

## Integers (Cont.)

- Doing Math
  - In base 10

24
+5 *Simply add digit position-by-position*
29

1
27 *What about "carry" situations?*
+25 *7+5= 12*
5 2

Enter the first digit value and "carry" the second

## Integers in computers:

- Absolute precision (no errors)
- Limited Range
  (Range is doubled for each additional bit)
- "Easy" math
- How does a computer do this?
  - Digital circuits (more later)

•Counting 0-15 in binary…

| Binary Value | Hex Value |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | A |
| 1011 | B |
| 1100 | C |
| 1101 | D |
| 1110 | E |
| 1111 | F |

## Integers in computers:

- What about negative integers?
  - Use one "digit" position for + and - sign
  - Reduces range to provide for negatives
    - (Since 1 bit indicates + or -, range is halved)

## Signed Integers

- Just use highest bit to indicate +/- sign
- Simple solution, but ….

Problems:

- 10000000 => Negative Zero??
- Can I subtract using addition?? -19 + 5 or -6 + -12

  10010011 (-19)    10000110 (-6)
  +00000101 (+5)    +10001100 (-12)
  10011000 (-24) ←Errors!!→ 00010010 (+18)

## Two's Complement
*for negative integers*

- Technique for representing negative integers
  - Step A: Complement bits (invert)
  - Step B: Add 1 to the complemented value
- *Interpret* the result as the negative of the original

To make a -5, start with (+5) 0000 0101

Compliment bits: 1111 1010
Add 1 to compliment: + 1

Final result (-5): 1111 1011

## Two's Complement
*for negative integers*

- Sign bit still used
- Single representation for zero
  - Try to make a negative 0
  - Result is still 0
- Math works!
  - Try (-19) + (5) example from before
- "Signed" Integers: Summary
  - If sign bit is 0 (positive), number is positional
  - If sign bit is 1 *(interpret as negative!)*, so……. *Take 2's complement – then find positive value*

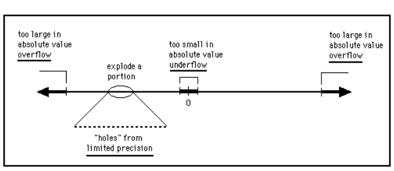
## Real (Decimal) Number Storage

- Real numbers are stored in floating point representation
  - a sign
  - an exponent
  - a mantissa (normalized decimal fraction)
    - no digits to the left of the decimal
    - first digit to the right of the decimal is nonzero
- Limited precision because most real numbers have an infinite decimal expansion (this holds no matter what number base is used in the representation)

## Real Number Storage
Limited Range and Precision

- There are three categories of numbers left out when floating point representation is used
  - numbers out of range because their absolute value is too large (similar to integer overflow)
  - numbers out of range because their absolute value is too small (numbers too near zero to be stored given the precision available
  - numbers whose binary representations require either an infinite number of binary digits or more binary digits than the bits available

## Real Number Storage
Limited Range and Precision Illustrated




## Limited Range and Precision
Some Consequences

- Limited range will invalidate certain calculations
  - If integers are involved, this can often be avoided by switching to real numbers
  - For real number calculations, this problem arises infrequently and in those cases can sometimes be handled by special methods. It is not a common occurrence in non-scientific work.
- Limited precision for real numbers is ***very pervasive***
  - Assume that most decimal calculations will, in fact, be in error!
  - Evaluate and use computer calculations with this in mind

## Risks in Numerical Computing

- Almost all computer calculations involve roundoff error (limited precision error)
- If not monitored and planned for carefully, such errors can lead to unexpected and catastrophic results
  - Ariane 5 Rocket Failure
  - Patriot Missile Failure during Gulf War

## The Explosion of the Ariane 5

- On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off.
- The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million.
- It turned out that the cause of the failure was a software error in the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer. The number was larger than 32,767, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.
- Back

## Patriot Missile Failure during Gulf War

- During the Gulf War, an American Patriot Missile battery in Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people.
- The General Accounting office reported on the cause of the failure. It turns out that the cause was an inaccurate calculation due to computer arithmetic errors.
- The time in tenths of second as measured by the system's internal clock was multiplied by 1/10 to produce the time in seconds.
- The value 1/10, which has a non-terminating binary expansion, was chopped at 24 bits. The small chopping error, when multiplied by the large number giving the time in tenths of a second, led to a significant error. Indeed, the Patriot battery had been up around 100 hours, and an easy calculation shows that the resulting time error due to the magnified chopping error was about 0.34 seconds. (The number 1/10 equals $1/2^4+1/2^5+1/2^8+1/2^9+1/2^{12}+1/2^{13}+\ldots$.
- A Scud travels at about 1,676 meters per second, and so travels more than half a kilometer in this time. This was far enough that the incoming Scud was outside the "range gate" that the Patriot tracked.
- Back

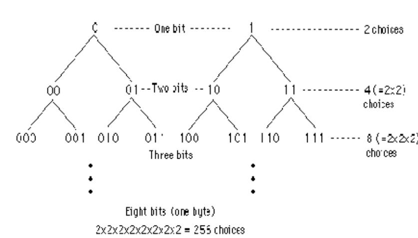
## Digital Representation

- All data is digitized into some pattern of symbols
- Meaning of the pattern depends on how we *interpret* the representation
- What does 0110 0001 0010 0101 1010 1001 … represent?
  - Could be text: a%©
  - Could be three ***unsigned*** integers: 97, 37, 169
  - Could be three ***signed*** integers: 97, 37, -77
  - Could be colors for one pixel: R:97 G:37: B169 = ■
  - Could be ???

## Digital Representation - Text

- Text (letters punctuation, invisible formatting characters)
- HTML (already discussed need for visual deign)

## Text - One Byte per Character Suffices



## English Character Set

- All uppercase and lowercase letters
- Punctuation symbols like ! . , ? : ; “ ‘ etc.
- Digits 0, …, 9
- Arithmetic symbols + = - / < >
- Assorted special symbols like # @ $ % ^ & * ( ) { } [ ] etc.
- Invisible formatting characters

## Using ASCII