## 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*)

## 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

## 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…

## 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

## 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

2 3 7

| | |
|---|---|
| 7 one's = | 7 |
| 3 ten's = | 30 |
| 2 hundred's = | 200 |

Combined Value = ???

## 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*
52

Enter the first digit value and "carry" the second

## Integers (Cont.)

- Converting from decimal (base 10) to binary (base 2)
  - Simple algorithm based on division by 2

## Integers (Cont.)

- Doing Math in Binary – base 2

```
 00001001  (9)
+00000100  (4)    Simply add digits position-by-position
 00001101  (13)
```

```
      1
 00000111    What about "carry" situations?
+00000101    1 + 1 = 2??  (Binary for 2 is 10)
       0
```

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*

## 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 ???

## Use-able Representation of Information

- Text (ASCII and/or UNICODE)
  - Examples: Text files, HTML, etc.
  - Already discussed need for effective visual design
- Lacking *Semantic* Structure
  - Text – even HTML formatted text – still does not give us a structure for determining *meaning* of the text
  - HTML provides "visual structure" only

  …need more.
- Finding the meaning of 'set' in the O.E.D. (using text *Find*)
- Must know semantic structure
  - added METADATA tags, et al.
  - *More later…*