Connecting with Computer Science, 2e

Chapter 7
Numbering Systems and Data Representations
Objectives

• In this chapter you will:
  – Learn why numbering systems are important to understand
  – Refresh your knowledge of powers of numbers
  – Learn how numbering systems are used to count
  – Understand the significance of positional value in a numbering system
  – Learn the differences and similarities between numbering system bases
Objectives (cont’d.)

• In this chapter you will (cont’d.):
  – Learn how to convert numbers between bases
  – Learn how to do binary and hexadecimal math
  – Learn how data is represented as binary in a computer
  – Learn how images and sounds are stored in a computer
Why You Need to Know About... Numbering Systems

• Computers store programs and data as binary digits
• Hexadecimal number system
  – Provides convenient representation
  – Written into error messages
• Helpful as students interact with computers in later computing courses and throughout their careers
Powers of Numbers: A Refresher

• Raising a number to a positive power (exponent)
  – Self-multiply number by the specified power
  – Example: \(2^3 = 2 \times 2 \times 2 = 8\) (asterisk = multiplication)
  – Special cases: 0 and 1 as powers
    • Any number raised to 0 = 1: \(10,555^0 = 1\)
    • Any number raised to 1 = itself: \(10,555^1 = 10,555\)

• Raising a number to a negative power
  – Follow same steps for positive power
  – Divide result into one
    • Example: \(2^{-3} = 1 / (2^3) = .125\)
Counting Things

• Numbers are used to count things
  – Base 10 (decimal): most familiar

• Computer uses base 2 (binary)
  – Two unique digits: 0 and 1

• Base 16 (hexadecimal) represent binary digits
  – Sixteen unique digits: 0–9, A–F

• Counting in all number systems
  – Count digits defined in number system until exhausted
  – Place 0 in ones column and carry 1 to the left
Positional Value

- Key principle of numbering systems
- Weight assigned to digit
  - Based on position in number
- Steps for base 10
  - Determine positional value of each digit by raising 10 to position within number
- Radix point
  - Divides fractional portion from the whole portion of a number
Positional Value (cont’d.)

• Positional value example: 436.95
  – 4 in position two
  – 3 in position one
  – 6 in position zero
  – To the right of the radix point:
    • 9 in position negative one
    • 5 in position negative two
Positional Value (cont’d.)

• Positional value specifies what multiplier the position gives to the overall number

• Example: 4321
  – Digit 3 multiplied by positional value of its position: 100 \( (10^2) \)
  – Sum of all digits multiplied by positional value determines total number of things being counted
Positional Value (cont’d.)

Figure 7-1, Positional values for a base 10 number
Positional Value (cont’d.)

• Positional value of the binary number $1011_2$
  – Rightmost position: positional value of the base (2) raised to the 0 power
    • Positional value: 1
    – Next position: value of 2 raised to the power of 1
    – Next position: 2 squared
    – Next position: 2 to the third

• Positional value significance
  – Gives weight each digit contributes to the number’s overall value
Positional Value (cont’d.)

Figure 7-2, Positional values for a base 2 number

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$

1011.011

$2^{-1} = .5$
$2^{-2} = .25$
$2^{-3} = .125$
How Many Things Does a Number Represent?

• Number equals sum of each digit x positional value
  – Translate with base 10
  – Example: $1001_2$
    • Equivalent to nine things
    • $(1 \times 2^0) + (0 \times 2^1) + (0 \times 2^2) + (1 \times 2^3)$
    • $1 + 0 + 0 + 8 = 9$

• General procedure (any base)
  – Calculate position value of the number by raising the base value to the power of the position
  – Multiply positional value by digit in that position
  – Add each calculated value together
Converting Numbers Between Bases

• Can represent any quantity in any base
• Counting process similar for all bases
  – Count until highest digit for base reached
  – Add 1 to next higher position to left
  – Return to 0 in current position
• Conversion: map from one base to another
  – Identities easily calculated
  – Identities obtained by table lookup
## Converting Numbers Between Bases (cont’d.)

<table>
<thead>
<tr>
<th>base 10</th>
<th>base 2</th>
<th>base 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>F</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>10001</td>
<td>11</td>
</tr>
<tr>
<td>18</td>
<td>10010</td>
<td>12</td>
</tr>
<tr>
<td>19</td>
<td>10011</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>10100</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 7-1, Counting in different bases
Converting to Base 10

• Methods:
  – Table lookup (more extensive than Table 7-1)
  – Algorithm for evaluating number in any base
    • Multiply the digit in each position by its positional value
    • Add results

• Example: $13_{16}$
  – Identify base: 16
  – Map positions to digits
  – Raise, multiply, and add:
    • $13_{16} = (3 \times 16^0) + (1 \times 16^1) = 19$
Converting from Base 10

• Method:
  – Reverse of converting to base 10
  – Follow an algorithm for converting from base 10

• Algorithm
  – Determine target base positional value nearest to or equal in value to decimal number
  – Determine how many times that positional value can be divided into the decimal number
  – Multiply number from Step 2 by the associated positional value, and subtract product from number chosen
Converting from Base 10 (cont’d.)

• Algorithm (cont’d.)
  – Use remainder from Step 3 as a new starting value, and repeat Steps 1-3 until Step 1 value is less than or equal to the maximum value in the target base
  – Converted number is the digits written down, in order from left to right
Binary and Hexadecimal Math

- Procedure for adding numbers similar in all bases
  - Difference lies in carry process
  - Value of carry equals value of base
- Procedure for subtraction
  - Similar
Binary and Hexadecimal Math (cont’d.)

Figure 7-3, Adding numbers in binary

\[
\begin{array}{c}
1 + 0 + 1 = 10 \\
\text{Bring the 0 down and carry the 1}
\end{array}
\quad
\begin{array}{c}
1 + 1 + 0 = 10 \\
\text{Bring the 0 down and carry the 1}
\end{array}
\quad
\begin{array}{c}
1 + 1 + 1 = 11 \\
\text{Bring the 1 down and carry the 1}
\end{array}
\quad
\begin{array}{c}
1 + 1 = 10 \\
\text{Bring the 0 down and carry the 1}
\end{array}
\]

\[
\begin{array}{c}
1011 \\
\text{Bring the 1 down and carry the 1}
\end{array}
\quad
1101
\quad
\begin{array}{c}
\text{Bring the 0 down and carry the 1}
\end{array}
\quad
11000
\]

Connecting with Computer Science, 2e
Binary and Hexadecimal Math (cont’d.)

Figure 7-4, Subtraction with base 2 numbers

0 is less than 1, so 10 is borrowed from the next column to the left, reducing it to 0:
10 - 1 = 1

After the borrow, the top value is 0:
0 - 0 = 0

1011
-0101
---
0110

1 - 0 = 1
1 - 1 = 0
Data Representation in Binary

• Binary values map to two states
  – On or off

• Bit
  – Each 1 and 0 (on and off) in a computer

• Byte
  – Group of 8 bits

• Word
  – Collection of bytes (typically 4 bytes)

• Nibble
  – Half a byte or 4 bits
Data Representation in Binary (cont’d.)

- Hexadecimal used as binary shorthand
  - Relate each hexadecimal digit to 4-bit binary pattern
  - Example: 1111 1010 1100 1110 =
    
    \[
    \begin{array}{cccc}
    F & A & C & E \\
    \end{array}
    \]
  - See Table 7-1 for verification
  - Example: C2D4 = 1100001011010100

- Information helps debug error messages
Representing Whole Numbers

• Whole numbers (integer numbers)
  – Stored in fixed number of bits
  – 2010 stored as 16-bit integer 0000011111011010
    • Equivalent hex value: 07DA

• Signed numbers stored with twos complement
  – Leftmost bit reserved for sign
    • 1 = negative and 0 = positive
  – If positive: leave as is
  – If negative: perform twos complement
    • Reverse bit pattern and add 1 to number using binary addition
## Representing Whole Numbers (cont’d.)

<table>
<thead>
<tr>
<th></th>
<th>52 decimal is equivalent to binary</th>
<th>52 decimal is equivalent to binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>+52</td>
<td>00110100</td>
<td>-52</td>
</tr>
<tr>
<td>-52</td>
<td>11001100</td>
<td></td>
</tr>
</tbody>
</table>

In twos complement, positive numbers are simply stored as binary values, with leading 0s to fit the field size.

Start with 00110100
Flip the bits to get 11001011
Add 1 to get 11001100

---

**Figure 7-5, Storing numbers in a twos complement 8-bit field**
Representing Fractional Numbers

• Computers store fractional numbers
  – Negative and positive
• Storage technique based on floating-point notation
  – Example: 1.345E+5
  – 1.345 = mantissa, E = exponent, + 5 moves decimal
• IEEE-754 specification
  – Uses binary mantissas and exponents
• Implementation details are part of advanced study
Representing Characters

• Computers store characters according to standards

• ASCII
  – Represents characters with 7-bit pattern
  – Provides for upper- and lowercase English letters, numeric characters, punctuation, special characters
  – Accommodates 128 ($2^7$) different characters

• Globalization places upward pressure
  – Extended ASCII: allows 8-bit patterns (256 total)
  – Unicode: defined for 16-bit patterns (34,168 total)
## Representing Characters (cont’d.)

<table>
<thead>
<tr>
<th>symbol</th>
<th>decimal representation</th>
<th>hex representation</th>
<th>binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>32</td>
<td>20</td>
<td>00100000</td>
</tr>
<tr>
<td>$</td>
<td>36</td>
<td>24</td>
<td>00100100</td>
</tr>
<tr>
<td>1</td>
<td>49</td>
<td>31</td>
<td>00110001</td>
</tr>
<tr>
<td>9</td>
<td>57</td>
<td>39</td>
<td>00111001</td>
</tr>
<tr>
<td>A</td>
<td>65</td>
<td>41</td>
<td>01000001</td>
</tr>
<tr>
<td>Z</td>
<td>90</td>
<td>5A</td>
<td>01011010</td>
</tr>
<tr>
<td>a</td>
<td>97</td>
<td>61</td>
<td>01100001</td>
</tr>
<tr>
<td>z</td>
<td>122</td>
<td>7A</td>
<td>01111010</td>
</tr>
<tr>
<td>}</td>
<td>125</td>
<td>7D</td>
<td>01111101</td>
</tr>
</tbody>
</table>

**Table 7-2, Sample standard ASCII characters**
Representing Images

• Screen image made up of small dots of colored light
  – Dot called “pixel” (picture element): smallest unit
  – Resolution: # pixels in each row and column
    • Each pixel stored in the computer as a binary pattern

• RGB encoding
  – Red, blue, and green assigned to eight of 24 bits
  – White represented with 1s, black with 0s
  – Color: amount of red, green, and blue specified in each of the 8-bit sections
Representing Images (cont’d.)

• Images are stored with pixel-based technologies
  – Compress large image files
    • JPG
    • GIF
  – Compress moving images
    • WAV
    • MP3
Representing Sounds

• Sound represented as waveform
  – Amplitude (volume)
  – Frequency (pitch)
• Computer samples sounds at fixed intervals
  – Samples given a binary value according to amplitude
  – Bits in each sample determine amplitude range
  – CD-quality audio
    • Sound must be sampled over 44,000 times a second
    • Samples must allow > 65,000 different amplitudes
Representing Sounds (cont’d.)

Figure 7-6, Digital sampling of a sound wave
One Last Thought

- To excel in the computer field:
  - Student must understand number conversions and data representations
- Everything stored in or takes place on a computer
  - Ultimately done in binary
- Computer professionals with strong understanding of chapter concepts will:
  - Perform better at whatever they specialize in
  - Become more essential in their organizations
Summary

• Knowledge of alternative number systems essential
• Machine language based on binary system
• Hexadecimal used to represent binary numbers
• Any number can be represented in any base
• Positional value: weight based on digit position
• Counting processes similar for all bases
• Conversion between bases: one-to-one mapping
• Arithmetic is defined for all bases
• Data representation: bits, nibbles, bytes, words
Summary (cont’d.)

• Twos complement is a technique for storing signed numbers
• Floating-point notation is a system used to represent fractions
• ASCII and Unicode: character set standards
• Image representation: based on binary pixels
• Sound representation: based on amplitude samples