Digital Information

INFO/CSE 100, Spring 2005
Fluency in Information Technology

http://www.cs.washington.edu/100
Readings and References

• Reading
  » Fluency with Information Technology
    • Chapter 11, Representing Multimedia Digitally

• Wikipedia - The Free Encyclopedia
  » Arabic numerals, ASCII
    • http://en.wikipedia.org/wiki/Arabic_numerals
    • http://en.wikipedia.org/wiki/Ascii

• Cyrillic Text
  • http://www.dimka.com/ru/cyrillic/
Info Representation

• Adult humans have 32 teeth
  » sometimes a tooth or two is missing!

• How can we represent a set of teeth?
  » How many different items of information?
    • 2 items - tooth or no tooth
  » How many "digits" or positions to use?
    • 32 positions - one per tooth socket
  » Choose a set of symbols
    no tooth: 0  tooth: 1
What's your tooth number?

incisors: 0 0 0 0 0 0 0 0
canines: 0 0 0 0 0 0 0 0
pre-molars: 0 0 0 0 0 0 0 0
molars: 0 0 0 0 0 0 0 0

no teeth ↔ 0000 0000 0000 0000 0000 0000 0000 0000

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

no molars ↔ 1111 1111 1111 1111 1111 0000 0000 0000

How many possible combinations? \(2 \times 2 \times 2 \times ... \times 2 = 2^{32} \approx 4 \text{ Billion}\)
Color monitors combine light from Red, Green, and Blue phosphors to show us colors.

How can we represent a particular color?

- How many different items of information?
  - 256 items - *distinguish 256 levels of brightness*

- How many "digits" or positions to use?
  - 3 positions - *one Red, one Green, one Blue*

- Choose a set of symbols
  - brightness level represented by the numbers 0 to 255
What is the pixel's color?

How many possible combinations?
\[256 \times 256 \times 256 = 256^3 \approx 16 \text{ Million}\]

16 M colors is often called "True Color"
How can we store numbers?

- We want to store numbers
  - 0 to 255 for color brightness
  - 0 to 4B for tooth configuration
  - 0 to 255 for ASCII character codes

- What do we have available in memory?
  - *Binary digits*
    - 0 or 1
    - on or off
    - clockwise or counter-clockwise
The hardware is binary

0 and 1 are the only symbols the computer can actually store directly in memory
  a single bit is either off or on

How many numbers can we represent with 0 and 1?
  How many different items of information?
    2 items - off or on
  How many "digits" or positions to use?
    let's think about that on the next slide
  Choose a set of symbols
    already chosen: 0 and 1
How many positions should we use?

It depends: how many numbers do we need?

<table>
<thead>
<tr>
<th>One position</th>
<th>Two positions</th>
<th>Three positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>001</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>010</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101</td>
</tr>
<tr>
<td></td>
<td></td>
<td>110</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111</td>
</tr>
</tbody>
</table>
The sky's the limit

• We can get as many numbers as we need by allocating enough positions
  » each additional position means that we get twice as many values because we can represent two numbers in each position
  » these are base 2 or binary numbers
    • each position can represent two different values

• How many different numbers can we represent in base 2 using 4 positions?
How can we read binary numbers?

Let's look at the equivalent *decimal* (ie, *base 10*) numbers.

<table>
<thead>
<tr>
<th>binary base 2</th>
<th>decimal base 10</th>
<th>binary base 2</th>
<th>decimal base 10</th>
<th>binary base 2</th>
<th>decimal base 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>00</td>
<td>0</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>01</td>
<td>1</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>2</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>3</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>

$111_2$ represents *exactly the same quantity* as $7_{10}$

They are just different ways of representing the same number.
Position matters!

<table>
<thead>
<tr>
<th>Position represents base 10</th>
<th>binary base 2</th>
<th>decimal base 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>32</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>64</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>128</td>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>256</td>
<td>1000</td>
<td>8</td>
</tr>
</tbody>
</table>

1000 1011

1 0 1 0

1 0 1 0

base 2

base 10
What do the positions represent?

Each position represents one more multiplication by the base value. For binary numbers, the base value is 2, so each new column represents a multiplication by 2.

What base 10 decimal value is equivalent to the base 2 binary value 10001010₂ shown above?
Some Examples

<table>
<thead>
<tr>
<th>2^7 = 128</th>
<th>2^6 = 64</th>
<th>2^5 = 32</th>
<th>2^4 = 16</th>
<th>2^3 = 8</th>
<th>2^2 = 4</th>
<th>2^1 = 2</th>
<th>2^0 = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

base 2

\[ 10_2 = 2_{10} \]
\[ 100_2 = 4_{10} \]
\[ 110_2 = 4_{10} + 2_{10} = 6_{10} \]
\[ 111_2 = 4_{10} + 2_{10} + 1_{10} = 7_{10} \]
\[ 1000_2 = 8_{10} \]
\[ 1001_2 = 8_{10} + 1_{10} = 9_{10} \]
Recall: What do number positions represent?

Each position represents one more multiplication by the base value.

For binary numbers, the base value is 2, so each new column represents a multiplication by 2.

\[
\begin{align*}
2^7 &= 128 \\
2^6 &= 64 \\
2^5 &= 32 \\
2^4 &= 16 \\
2^3 &= 8 \\
2^2 &= 4 \\
2^1 &= 2 \\
2^0 &= 1 \\
\end{align*}
\]

\[
\begin{array}{ccccccc}
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

\[
1 \cdot 128 + 1 \cdot 8 + 1 \cdot 2 = 138_{10}
\]
Use the base, Luke

- Each position represents one more multiplication by the base value
  - The base value can be 2 - *binary numbers*
    - Two symbols: 0 and 1
    - Each column represents a multiplication by two
  - The base value can be 10 - *decimal numbers*
    - Ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    - Each column represents a multiplication by ten

\[
\begin{array}{cccc}
10 \times 10 \times 10 & 10 \times 10 & 10 & 1 \\
10^3 = 1000 & 10^2 = 100 & 10^1 = 10 & 10^0 = 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 3 & 8 \\
\end{array}
\]

\[
1 \cdot 100 + 3 \cdot 10 + 8 \cdot 1 = 138_{10}
\]
Base 16 Hexadecimal

• The base value can be **16** - *hexadecimal numbers*
  » Sixteen symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  » Each column represents a multiplication by sixteen
  » Hex is easier to use than binary because the numbers are shorter even though *they represent the same value*

\[
\begin{array}{cccc}
16 \times 16 \times 16 & 16 \times 16 & 16 & 1 \\
16^3 = 4096 & 16^2 = 256 & 16^1 = 16 & 16^0 = 1
\end{array}
\]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>8</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>base 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
8 \cdot 16 + 10 \cdot 1 = 138_{10}
\]
## Four binary bits ⇔ One hex digit

<table>
<thead>
<tr>
<th>Binary (base 2)</th>
<th>Hexdecimal (base 16)</th>
<th>Decimal (base 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Binary (base 2)</th>
<th>Hexdecimal (base 16)</th>
<th>Decimal (base 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>A</td>
<td>10</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>B</td>
<td>11</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>C</td>
<td>12</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>D</td>
<td>13</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>E</td>
<td>14</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>F</td>
<td>15</td>
</tr>
</tbody>
</table>
Binary to Hex examples

1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1

base 2

8 2 0 7 A 1 0 F

base 16

10000010000001111010000100001111

= 8207A10F

10000011010001010110100110111110

base 2

1000011010001010110100110111110

= ______________________

16
Whew! We are now official geeks ...
Recall: The hardware is binary

• How many numbers can we represent with 0 and 1?
  » As many as we want, it just takes a little more space to get a bigger range

• So what can we represent with these numbers?
  » Anything that has a numeric value or can be associated with a numeric value
  » Number of people, index into a list, account balance, ...
  » Alphabetic characters, punctuation marks, display tags
  » Any signal that can be converted into numeric values
    • colors, sounds, water level, blood pressure, temperature
  » Computer instructions
Represent numbers

- How many bit positions to allocate?
  » Depends on the desired range
  » 8 bits → 0 to 255
    • or -128 to +127
  » 16 bits → 0 to 65535
    • or -32768 to +32767
  » 32 bits → 0 to 4294967296
    • or -2B to +2B
Represent Text - ASCII

• Assign a unique number to each character
  » 7-bit ASCII
    • Range is 0 to 127 giving 128 possible values
    • There are 95 printable characters
    • There are 33 control codes like tab and carriage return
Hello world!!!
Aa
Bb
Cc
Dd

0: 48 65 6C 6C 6F 20 77 6F 72 6C 64 21 21 21 0D 0A Hello world!!!
10: 41 61 0D 0A 42 62 0D 0A 43 63 0D 0A 44 64 0D 0A Aa..Bb..Cc..Dd..
Represent Text - Unicode

• The goal of Unicode is to provide the means to encode the text of every document people want to store in computers
• Unicode aims to provide a unique number for each letter, without regard to typographic variations used by printers
• Unicode encodes each character in a number
  » the number can be 7, 8, 16, or 32 bits long
  » 16-bit encoding is common today
This page was created in order to test support for different encodings that allows to present Cyrillic correctly.

Эта страница создана для тестирования поддержки различных кодировок позволяющих корректно показывать кириллицу.
Represent Text - Postscript

- Postscript is a page description language somewhat like HTML
  » The file is mostly text and can be looked at with a regular text editor
  » programs that know what it is can interpret the embedded commands
  » Programs *and printers* that understand Postscript format can display complex text and graphical images in a standard fashion
This page was created in order to test support for different entities that allows to present Cyrillic correctly.

Эта страница создана для тестирования поддержки различных кодировок и корректно показывать кириллицу.
Represent Text - PDF

• PDF is another page description language based on Postscript
• The file is mostly text
  » can be looked at with a regular text editor
  » programs that know what it is can interpret the embedded commands
  » just like Postscript and HTML in that respect
This page was created in allows to present Cyrillic.

Эта страница создана для корректно показывать кириллицу.
Represent Color - Bit Map

- Numbers can represent anything we want
- Recall that we can represent colors with three values
  » Red, Green, Blue brightness values
- There are *numerous* formats for image files
  » All of them store some sort of numeric representation of the brightness of each color at each pixel of the image
  » commonly use 0 to 255 range (or 0 to FF₁₆)
What about "continuous" signals?

- Color and sound are natural quantities that don't come in nice discrete numeric quantities
- But we can “make it so!”
Digitized image contains color data
And much, much more!

<table>
<thead>
<tr>
<th>EXIF Tag</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>C:\home\finson\cse100\slides\19-digital\uw-quad.jpg</td>
</tr>
<tr>
<td>Make</td>
<td>Canon</td>
</tr>
<tr>
<td>Model</td>
<td>Canon PowerShot SD100</td>
</tr>
<tr>
<td>Orientation</td>
<td>Top left</td>
</tr>
<tr>
<td>XResolution</td>
<td>190</td>
</tr>
<tr>
<td>YResolution</td>
<td>190</td>
</tr>
<tr>
<td>ResolutionUnit</td>
<td>Inch</td>
</tr>
<tr>
<td>DateTime</td>
<td>2004:11:16 19:31:08</td>
</tr>
<tr>
<td>YCbCrPositioning</td>
<td>Centered</td>
</tr>
<tr>
<td>ExifOffset</td>
<td>137</td>
</tr>
<tr>
<td>ExposureTime</td>
<td>1/250 seconds</td>
</tr>
<tr>
<td>FNumber</td>
<td>3.50</td>
</tr>
<tr>
<td>ExifVersion</td>
<td>220</td>
</tr>
<tr>
<td>DateTimeOriginal</td>
<td>2004:10:23 12:58:17</td>
</tr>
<tr>
<td>DateTimeDigitized</td>
<td>2004:10:23 12:58:17</td>
</tr>
<tr>
<td>ComponentsConfiguration</td>
<td>YCbCr</td>
</tr>
<tr>
<td>CompressedBitsPerPixel</td>
<td>5 (bits/pixel)</td>
</tr>
<tr>
<td>ShutterSpeedValue</td>
<td>1/251 seconds</td>
</tr>
<tr>
<td>ApertureValue</td>
<td>F 3.51</td>
</tr>
<tr>
<td>ExposureBiasValue</td>
<td>0.00</td>
</tr>
<tr>
<td>MaxApertureValue</td>
<td>F 3.51</td>
</tr>
<tr>
<td>MeteringMode</td>
<td>Multi-segment</td>
</tr>
<tr>
<td>Flash</td>
<td>Not fired, auto mode</td>
</tr>
<tr>
<td>FocalLength</td>
<td>919 mm</td>
</tr>
<tr>
<td>UserComment</td>
<td></td>
</tr>
<tr>
<td>FlashPixVersion</td>
<td></td>
</tr>
</tbody>
</table>
Summary

• Bits can represent any information
  » Discrete information is directly encoded using binary
  » Continuous information is made discrete

• We can look at the bits in different ways
  » The format guides us in how to interpret it
  » Different interpretations let us work with the data in different ways