Announcements

- Labs come in two forms: exercises and free work; Thursday’s lab is some of each
How we represent bits, numbers, letters?

Communicating in the Blink of an Eye

Lawrence Snyder
University of Washington, Seattle
Key principle: *Information is the presence or absence of a phenomenon at given place/time*

Turn signal is an example

- Phenom: Flashing light
- Present: Flashing
- Absent: Off
- Info: Present == intention to turn in specific direction
- Place (side of car)
- Time: now
The **Presence and Absence** of a phenomenon at a specific place and time abbreviated: PandA

Phenomena: light, magnetism, charge, mass, color, current, ...

Detecting depends on phenomenon – but the result must be discrete: was it detected or not; there is no option for “sorta there”

Place and time apply, but usually default to “obvious” values; not so important to us

Many alternatives ...
Subtleties In Interpreting Bits

- Pay attention to the meaning of the bits
Subtleties In Interpreting Bits

- Pay attention to the meaning of the bits
Subtleties In Interpreting Bits

- Pay attention to the meaning of the bits
A Curious Story... That You Know

The Diving Bell and the Butterfly
Jean-Dominique Bauby
Asking Yes/No Questions

- A protocol for Yes/No questions
  - One blink == Yes
  - Two blinks == No
- PandA implies that this is not the fewest number of blinks ... really?
- “Presence and absence” is too long, use 0, 1
- At the coffee shop ... record passersby:

  - == going right
  - == wearing purple

- In multi-state cases, pick one for present, all others are absent
PandA is a binary representation because it uses 2 patterns

Bit – it’s a contraction for “binary digit”

Information exists even if the phenom is absent

Sherlock Holmes’ s Mystery of Silver Blaze -- a popular example where “absent” gives information … the dog didn’t bark, that is the phenomenon wasn’t detected

Memory -- a position in space/time capable of being set and detected in 2 patterns
Bytes

- A byte is eight bits treated as a unit
  - Adopted by IBM in 1960s
  - A standard measure ever since
  - Bytes encode the Latin alphabet using ASCII -- the American Standard Code for Information Interchange
A byte has 256 configurations (symbols), good enough for usual American and Western European languages.
UTF-8

Uniform Transformation Format for bytes (UTF-8) is universal ... all characters have a place: 1,2,3,4 B
UTF-8

Uniform Transformation Format for bytes (UTF-8) is universal ... all characters have a place: 1, 2, 3, 4 B
- 100,000 characters

can you read this?
Bits and bytes encode the information, but that’s not all

- Tags encode format and some structure in word processors
- Tags encode format and some structure in HTML
- In the *Oxford English Dictionary* tags encode structure and some formatting
- Tags are one form of meta-data: *meta-data* is information about information
**OED Entry For Byte -- Metadata**

**byte** (balt). *Computers.* [Arbitrary, prob. influenced by *bit* sb.\(^4\) and *bite* sb.] A group of eight consecutive bits operated on as a unit in a computer. *1964 Blaauw & Brooks* in *IBM Systems Jrnl.* III. 122 An 8-bit unit of information is fundamental to most of the formats [of the System/360]. A consecutive group of \(n\) such units constitutes a field of length \(n\). Fixed-length fields of length one, two, four, and eight are termed bytes, halfwords, words, and double words respectively. *1964 IBM Jrnl. Res. & Develpm.* VIII. 97/1 When a byte of data appears from an I/O device, the CPU is seized, dumped, used and restored. *1967 P. A. Stark* *Digital Computer Programming* xix. 351 The normal operations in fixed point are done on four bytes at a time. *1968 Dataweek* 24 Jan. 1/1 Tape reading and writing is at from 34,160 to 192,000 bytes per second.

---

*etym* Arbitrary, prob. influenced by *<xr>*<x>*bit* <ps>n.</ps>* and *<xr>*<x>*bite* <ps>n.</ps>*  A group of eight consecutive bits operated on as a unit in a computer. *1964 IBM Jrnl. Res. & Develpm.* VIII. 97/1

*<quanit>4</quanit>* A group of eight consecutive bits operated on as a unit in a computer. *1964 IBM Jrnl. Res. & Develpm.* VIII. 97/1

*<quanit>122</quanit>* An 8-bit unit of information is fundamental to most of the formats [of the System/360]. A consecutive group of \(i\) such units constitutes a field of length \(i\). Fixed-length fields of length one, two, four, and eight are termed bytes, halfwords, words, and double words respectively. *1967 P. A. Stark* *Digital Computer Programming* xix. 351

*<quanit>351</quanit>* The normal operations in fixed point are done on four bytes at a time. *1968 Dataweek* 24 Jan. 1/1

*<quanit>34,160</quanit>* Tape reading and writing is at from 34,160 to 192,000 bytes per second.
Binary numbers, like decimal numbers, use *place notation*

\[ 1101 = 1 \times 1000 + 1 \times 100 + 0 \times 10 + 1 \times 1 \]

\[ = 1 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 10^0 \]

except that the base is 2 not 10

\[ 1101 = 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \]

\[ = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \]

1101 in binary is 13 in decimal
Positional Notation Logic

Binary is just like decimal except that it uses base 2 rather than base 10 ...
Representing 160 in Binary

Given a binary number, add up the powers of 2 corresponding to its 1s

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

\[= 160\]
Representing 76 In Binary

Given a binary number, add up the powers of 2 corresponding to 1s

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

= 76
So Husky purple is (160, 76, 230) which is

\[
1010 \ 0000 \ 0100 \ 1100 \ 1110 \ 0110
\]

160  76  230
So Husky purple is (160,76,230) which is

1010 0000 0100 1100 1110 0110
160 76 230

Suppose you decide it’s not “red” enough

- Increase the red by 16 = 1 0000

1010 0000
+ 1 0000
1011 0000

Adding in binary is pretty much like adding in decimal
Adding In Binary ... like Decimal

- Increase by 16 more

\[
\begin{array}{c}
00110 \ 000 \\
1011 \ 0010 \\
+ \ 1 \ 0100 \\
1100 \ 0110 \\
\end{array}
\]

Carries

The rule: When the “place sum” equals the radix or more, subtract radix & carry

Check it out online: searching hits 19M times, and all of the p.1 hits are good explanations
What is 230? Fill in the Table:

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
<th>102</th>
<th>38</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Subtract</td>
<td>102</td>
<td>38</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

© 2010 Larry Snyder, CSE
Find Binary From Decimal

Place number to be converted into the table; fill place value row with decimal powers of 2

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
</tr>
<tr>
<td>Subtract</td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td></td>
</tr>
</tbody>
</table>
Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
</tr>
<tr>
<td>Subtract</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
<th>102</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Subtract</td>
<td>102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
<th>102</th>
<th>38</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>Subtract</td>
<td>102</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
<th>102</th>
<th>38</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>Subtract</td>
<td>102</td>
<td>38</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
<th>102</th>
<th>38</th>
<th>6</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Subtract</td>
<td>102</td>
<td>38</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

© 2010 Larry Snyder, CSE
Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
<th>102</th>
<th>38</th>
<th>6</th>
<th>6</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Subtract</td>
<td>102</td>
<td>38</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

© 2010 Larry Snyder, CSE
**Find Binary From Decimal**

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
<th>102</th>
<th>38</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Subtract</td>
<td>102</td>
<td>38</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

© 2010 Larry Snyder, CSE
Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
<th>102</th>
<th>38</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Subtract</td>
<td>102</td>
<td>38</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

<table>
<thead>
<tr>
<th>Num Being Converted</th>
<th>230</th>
<th>230</th>
<th>102</th>
<th>38</th>
<th>6</th>
<th>6</th>
<th>6</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Subtract</td>
<td>102</td>
<td>38</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Read off the result: 0 1110 0110
Final Fact: Bits Are IT

- We ALL KNOW computers represent data by binary numbers
- NOT QUITE TRUE
- Computers represent information by bits
  - ASCII, numbers (yes, in binary), metadata + computer instructions, color, sound, video, etc.
- Fundamental Fact –
  - Bits can represent ALL information
  - Bits have no inherent meaning ... you don’t know what 1100 0100 1010 1110 means ... it could be anything