Announcements

Project 1B due Today at 11:00 PM
Midterm Friday, in class
More Digital Representation

Discrete information is represented in binary (PandA), and "continuous" information is made discrete.
Images are constructed from picture elements (pixels); color uses RGB light

The RGB color intensities are specified by 3 numbers in the range [0, 255], ie 1 byte each

Black = [0, 0, 0] 0000 0000 0000 0000 0000 0000
Gray = [128, 128, 128] 1000 0000 1000 0000 1000 0000
White = [255, 255, 255] 1111 1111 1111 1111 1111 1111

White-gray-black all have same values for RGB
Colors use different combinations of RGB

- **Husky Purple**
  - Red = 160
  - Green = 76
  - Blue = 230
Positional Notation

The RGB intensities are binary numbers.

Binary numbers, like decimal numbers, use place notation except that the base is 2 not 10.

\[ 1101 = 1 \times 1000 + 1 \times 100 + 0 \times 10 + 1 \times 1 \]
\[ = 1 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 10^0 \]

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

1010 0000

1x2⁷ = 1x128 = 128
0x2⁶ = 0x64 = 0
1x2⁵ = 1x32 = 32
0x2⁴ = 0x16 = 0
0x2³ = 0x8 = 0
0x2² = 0x4 = 0
0x2¹ = 0x2 = 0
0x2⁰ = 0x1 = 0

= 160
Binary Numbers

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

```
0x2^7  = 0x128  = 0
1x2^6  = 1x64   = 64
0x2^5  = 0x32   = 0
0x2^4  = 0x16   = 0
1x2^3  = 1x8    = 8
1x2^2  = 1x4    = 4
0x2^1  = 0x2    = 0
0x2^0  = 0x1    = 0
```

0100 1100  = 76
Given a binary number, add up the powers of 2 corresponding to 1s

- \(1 \times 2^7 = 1 \times 128 = 128\)
- \(1 \times 2^6 = 1 \times 64 = 64\)
- \(1 \times 2^5 = 1 \times 32 = 32\)
- \(0 \times 2^4 = 0 \times 16 = 0\)
- \(0 \times 2^3 = 0 \times 8 = 0\)
- \(1 \times 2^2 = 1 \times 4 = 4\)
- \(1 \times 2^1 = 1 \times 2 = 2\)
- \(0 \times 2^0 = 0 \times 1 = 0\)

\[\text{Total} = 128 + 64 + 32 + 0 + 0 + 4 + 2 + 0 = 230\]
Recall that Husky purple is $(160, 76, 230)$ which in binary is:

$$10100000 \ 01001100 \ 11100110$$

$160 \ 76 \ 230$

Suppose you decide it’s not “red” enough:

- **Increase the red by $16 = 10000$**

  $$10100000$$

  $$+ \ \ 10000$$

  $$\underline{+ \ \ 10000}$$

  $$10110000$$

Adding in binary is pretty much like adding in decimal.
A Redder Purple

Increase by 16 more

\[ \begin{array}{c}
00110 000 \\
1011 0000 \\
\hline
+ 1 0000 \\
\hline
1100 0000 \\
\end{array} \]

Carries

The rule: When the “place sum” equals the radix or more, subtract radix & carry.
Find Binary From Decimal

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>2</td>
<td>0</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>
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>256 128 64 32 16 8 4 2 1</td>
</tr>
<tr>
<td>Subtract Binary Num</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 → 230</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place Value</td>
<td>256 128 64 32 16</td>
</tr>
<tr>
<td>Subtract</td>
<td>0</td>
</tr>
<tr>
<td>Binary Num</td>
<td>0</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>
**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>
</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>
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></th>
<th></th>
<th></th>
<th></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></td>
<td></td>
<td></td>
<td></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>
<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>
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>
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>2</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></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>
<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>2</td>
<td>0</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>
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>
<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>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</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
“Continuous” information like light and sound must be made “discrete”

Digital audio uses 44,100 samples per second of 16 bits on two channels, or 10,584,000 B/min.
Compression

Compression: use fewer bits

* Lossless - Recover the data
* Lossy - Lose the original data

JPEG

Original

Over compressed
Run-Length Compression

Give number of 1s, number of 0s, etc.

1111111111... (270 1s)
1111111111... (270 1s)
1100000000... (2 1s)(266 0s)(2 1s)
1100000000... (2 1s)(266 0s)(2 1s)

Forget row encoding ... alternate

[Size: 270x200](542)(266)(4)(266)(4)(266)(4)(266) ...
Bits represent information, but their interpretation gives bits meaning.

- Could be a number, color, instruction, ASCII, sound samples, IP address, ...

**Bias-free Universal Medium Principle:** Bits can represent all discrete information; bits have no inherent meaning.
Summary

Bits can represent any information

* Discrete information is directly encoded using binary
* Continuous information is made discrete
* Bias-free Universal Medium Principle