Data Representation

Interpreting bits to give them meaning

Part 2: Hexadecimal and Practical Issues

Class Reminders

Class/Assignments:
● Assignment 2 will be handed out today - start planning!

Lab Exercises:
● Review Lab 4 solutions (in Blackboard) - important for HW 2!

Blown to Bits:
● Chapter 2 discussion - contribute before Wednesday (10:00am)

From Last Time...

Key points from “Data Representation, Part 1”:
● A number is an abstract idea
● Anything you can point at or write down is a representation of a number
● Lots of different representations for the same number:
  ○ Written in decimal notation (what we’re most familiar with)
  ○ Written in roman numerals (e.g., 6 is the same as VI)
  ○ Written as a set of “tick marks” (e.g., 6 is the same as IIIIII)
  ○ Written in binary (e.g., 6 is the same as 1102)
  ○ As a sequence of voltages on wires
● Computers work with binary because switches are off or on (0 or 1)
● Converting between number bases doesn’t change the number, just chooses a different representation
Hexadecimal - another useful base

Hexadecimal is base 16. How do we get 16 different digits? Use letters! 

Hexadecimal digits (or "hex digits" for short):
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Counting - now our odometer has 16 digits:

<table>
<thead>
<tr>
<th>0₆ (≡ 0₁₀)</th>
<th>1₆ (≡ 1₁₀)</th>
<th>2₆ (≡ 2₁₀)</th>
<th>3₆ (≡ 3₁₀)</th>
<th>4₆ (≡ 4₁₀)</th>
<th>5₆ (≡ 5₁₀)</th>
<th>6₆ (≡ 6₁₀)</th>
<th>7₆ (≡ 7₁₀)</th>
<th>C₆ (≡ 12₁₀)</th>
<th>D₆ (≡ 13₁₀)</th>
<th>E₆ (≡ 14₁₀)</th>
<th>F₆ (≡ 15₁₀)</th>
<th>...</th>
</tr>
</thead>
</table>

Hexadecimal/Decimal Conversions

Conversion process is like binary, but base is 16

Problem 1: Convert 423₁₀ to hexadecimal:
423/16 = quotient 26, remainder 7 (=7₁₆)
26/16 = quotient 1, remainder 10 (=A₁₆)
1/16 = quotient 0, remainder 1 (=1₁₆)
• Reading digits bottom-up: 423₁₀ = 1A7₁₆

Problem 2: Convert 9C3₁₆ to decimal:
Start with first digit, 9
9*16 + 12 = 156
156*16 + 3 = 2499
• Therefore, 9C3₁₆ = 2499₁₀

Hexadecimal/Decimal Conversions

Conversion process is like binary, but base is 16

Problem 1: Convert 423₁₀ to hexadecimal:
423/16 = quotient 26, remainder 7 (=7₁₆)
26/16 = quotient 1, remainder 10 (=A₁₆)
1/16 = quotient 0, remainder 1 (=1₁₆)
• Reading digits bottom-up: 423₁₀ = 1A7₁₆

Problem 2: Convert 9C3₁₆ to decimal:
Start with first digit, 9
9*16 + 12 = 156
156*16 + 3 = 2499
• Therefore, 9C3₁₆ = 2499₁₀

Your turn! Convert:

<table>
<thead>
<tr>
<th>103₂₆</th>
<th>247₂₆</th>
<th>95₂₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>3C₂₆</td>
<td>89₂₆</td>
<td>357₁₆</td>
</tr>
</tbody>
</table>

Hex Digit List

<table>
<thead>
<tr>
<th>0₁₆</th>
<th>1₁₆</th>
<th>2₁₆</th>
<th>3₁₆</th>
<th>4₁₆</th>
<th>5₁₆</th>
<th>6₁₆</th>
<th>7₁₆</th>
<th>C₁₆</th>
<th>D₁₆</th>
<th>E₁₆</th>
<th>F₁₆</th>
<th>1₀₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0₁₀</td>
<td>1₁₀</td>
<td>2₁₀</td>
<td>3₁₀</td>
<td>4₁₀</td>
<td>5₁₀</td>
<td>6₁₀</td>
<td>7₁₀</td>
<td>12₁₀</td>
<td>13₁₀</td>
<td>14₁₀</td>
<td>15₁₀</td>
<td>0₁₀</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8₁₀</th>
<th>9₁₀</th>
<th>A₁₀</th>
<th>B₁₀</th>
<th>C₁₀</th>
<th>D₁₀</th>
<th>E₁₀</th>
<th>F₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>10₁₀</td>
<td>11₁₀</td>
<td>12₁₀</td>
<td>13₁₀</td>
<td>14₁₀</td>
<td>15₁₀</td>
<td>16₁₀</td>
<td></td>
</tr>
</tbody>
</table>
Hexadecimal/Binary Conversions

Exactly 16 hex digits, and exactly 16 4-bit binary numbers

Converting between hex and binary is easy - 4 bits at a time:

<table>
<thead>
<tr>
<th>Hex Digit List</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>

Example:

<table>
<thead>
<tr>
<th>Hex Digit List</th>
<th>0110</th>
<th>1010</th>
<th>0110</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>A</td>
<td>6</td>
</tr>
</tbody>
</table>

Answer: 3A6

Problem 1: Convert 01110100110₂ to hexadecimal

011 = 3
1010 = A
0110 = 6

Answer: 3A6₁₆

Problem 2: Convert D49₁₆ to binary

D = 1101
4 = 100
9 = 1001

Answer: 110101001001₂

Use of hexadecimal in file dumps

Binary is a very long format (8 bits per byte), but often data files only make sense as binary data. Hexadecimal is great for this - simple one-to-one correspondence with binary, and more compact.

Sample "file dump":

```
0000000: ffd8 ffe1 35fe 4578 6966 0000 4949 2a00  ....5.Exif..II*
0000010: 0800 0000 0b00 0e01 0200 2000 0000 9200  .......... ......
0000020: 0000 0f01 0200 0600 0000 b200 0000 1001  ................
0000030: 0200 1900 0000 b800 0000 1201 0300 0100  ................
0000040: 0000 0600 0000 1a01 0500 0100 0000 d800  ................
0000050: 0000 1b01 0500 0100 0000 e000 0000 2801  ..............(.
0000060: 0300 0100 0000 0200 0000 3201 0200 1400  ..........2....
0000070: 0000 e800 0000 1302 0300 0100 0000 0200  ................
0000080: 0000 6987 0400 0100 0000 fc00 0000 2588  ..i...........%.
0000090: 0400 0100 0000 2413 0000 f213 0000 2020  ......$.......
00000c0: 6e6f 6e00 4361 6e6f 6e20 506f 7765 7253  non.Canon PowerS
00000d0: 686f 7420 5358 3233 3020 4853 0000 0000  hot SX230 HS....
00000e0: 0000 b400 0000 0100 0000 b400 0000 0100  ................
00000f0: 0100 0000 3230 3131 3a30 373a 3134 2031  ....2011:07:14 1
0000100: 353a 3039 3a32 3700 2100 9a82 0500 0100  5:09:27.!.......
0000110: 0000 8e02 0000 9d82 0500 0100 0000 9602  ................
0000120: 0000 2788 0300 0100 0000 6400 0000 3088  ..'.......d...0.
```

The same data, showing character representation

```
Remember....

Don't get lost in the details and manipulations:

*Any base is a representation of an abstract number*

We are interested in working with the number, and computations are not "in a base" - the base is only useful for having it make sense to us or the computer

Practice!

*You should be able to convert from one base to another.*

Lots of ways to practice:
- By hand: Pick a random number convert to binary and convert back - did you get the same value?
  - This isn't foolproof: You could have made two mistakes!
- With a calculator: Many calculators (physical and software) do base conversion - check your randomly selected conversions.
- With a web site: Several web sites provide says to practice
  - For example, see [http://cs.iupui.edu/~aharris/230binPractice.html](http://cs.iupui.edu/~aharris/230binPractice.html)

Practical Issues with Numbers

*Finite Length Integers*

Question (a little contrived):

If a CPU has 4 single-bit storage locations for each number, what happens when you add:

\[ 1111_2 + 0001_2 = \_\_\_\_\_2 \]
Practical Issues with Numbers
Finite Length Integers

Question (a little contrived):
If a CPU has 4 single-bit storage locations for each number, what happens when you add:

$1111_2 + 0001_2 = \_\_\_\_\_ \_2$

Answer Part 1: If you did this on paper, you’d get $10000_2$
Which leads to another question:
How do we store 5 bits when there are only storage locations for 4 bits?

Answer Part 2: What CPUs do is throw out the 5th bit, storing $0000_2$
Which means: To a 4-bit computer, $15 + 1 = 0$

On real computers:
- This happens, but with 32-bit numbers or 64-bit numbers instead of 4.
- When things “wrap around” it actually goes to negative values...
  On a 32-bit CPU: $2,147,483,647 + 1 = -2,147,483,648$

However: Some programming languages/systems support numbers larger than the hardware, by using multiple memory locations.

Let’s try this!
Practical Issues with Numbers

Finite Length Integers

In C:
```c
int val=1000*1000*1000*1000;
printf("%d\n", val);
```

Outputs:
```
-727379968
```

In Java:
```java
int val = 1000*1000*1000*1000;
System.out.println(val);
```

Outputs:
```
-727379968
```

In Python:
```python
x = 1000*1000*1000*1000
print x
```

Outputs:
```
1000000000000
```

First thought: Python is cool!

Second thought: Don’t expect something for nothing…

Let’s do something pretty useless (that takes a lot of integer operations)

Problem: Compute the last 6 digits of the billionth Fibonacci number

In C:
```
3.5 seconds
```

In Java:
```
3.4 seconds
```

In Python:
```
3 minutes, 56.2 seconds
```

Times on my laptop: Intel i7-3740QM (2.7GHz)
Practical Issues with Numbers

Finite Precision Floating Point

Question: How do you write out 1/3 in decimal?
Answer: 0.33333333333….

Observation: Impossible to write out exactly with a finite number of digits

The same holds in binary!

<table>
<thead>
<tr>
<th>Can be written exactly</th>
<th>Cannot be written exactly</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 = 0.1₂</td>
<td>1/3 = 0.01010101010…₂</td>
</tr>
<tr>
<td>0.25 = 0.01₁</td>
<td>1/5 = 0.001100110011…₂</td>
</tr>
<tr>
<td>0.375 = 0.011₂</td>
<td>1/10 = 0.0001100110011…₂</td>
</tr>
</tbody>
</table>

Imagine: How hard is it to write banking software when there is no finite representation of a dime (0.10 dollars)?

Solutions people came up with:
- Work with cents (integers!) or special codings (BCD=Binary Coded Decimal)

Bottom Line:
There are a lot of subtle problems with numbers that go beyond the level of study in CSC 100.

These issues usually don’t come up.
But… when they matter, they can matter a LOT.

For now: Be aware what the issues are.

For a later class: Understand the details.

Solutions people came up with:
- Work with cents (integers!) or special codings (BCD=Binary Coded Decimal)

Still More Data Representation for Later

Now we know all about representing numbers

But computers also deal with text, web pages, pictures, sound/music, video, ...

How does that work?