Lecture 17:
How does a computer... represent information and data?

Big Idea #1: Universal Computing Device
All computers (given enough time and storage) are capable of computing exactly the same things
From Theory to Practice

Theory: Computer can *compute* anything that’s possible to compute (given enough time and storage)

Practice: *Solving problems* involves computing under constraints

- **Time**
  - weather forecast, next frame of animation, ...
- **Cost**
  - cell phone, automotive engine controller, ...
- **Power**
  - cell phone, handheld video game, ...

Big Idea #2: Abstractions Hide Complexity

Software

Application Program
- Algorithms
- Language

Hardware

Instruction Set Architecture (and I/O Interfaces)
- Microarchitecture
- Circuits
- Devices
Abstractions Hide Complexity

How do we solve a problem using a computer?
Sequence of transformations between layers of abstraction

- **Problem**
  - Stated using "natural language"

- **Algorithm**
  - Step-by-step procedure

- **Program**
  - Express algorithm w/ computer language

- **Instr Set Architecture**
  - Set of instructions computer can perform

- **Software Design**
  - Choose algorithms and data structures

- **Programming**
  - Use language to express design

- **Compiling/Interpreting**
  - Convert language to machine instructions

More and more layers...

- **Instr Set Architecture**
  - Organization of a processor implementation

- **Microarch**
  - Layout and combination of gates (and, or, not, nor, nand)

- **Circuits**
  - Properties of materials, manufacturability

- **Processor Design**
  - Choose structures to implement ISA

- **Logic/Circuit Design**
  - Gates and low-level circuits to implement components

- **Process Engineering & Fabrication**
  - Develop and manufacture lowest-level components
Example:
Many Choices at Each Level

Solve a system of equations

Red-black SOR

Gaussian elimination

Jacobi iteration

Multigrid

FORTRAN

C

C++

Java

Sun SPARC

Intel x86

Compaq Alpha

Pentium II

Pentium III

AMD Athlon

Ripple-carry adder

Carry-lookahead adder

CMOS

Bipolar

GaAs

Tradeoffs:
- Cost
- Performance
- Power
(etc.)

How do computers...

... Represent data and information?
... Represent numbers, words, pictures, and movies?
... Act so logically?

... Manipulate and remember data?
... Execute instructions?
... Access data quickly?

... Run multiple programs simultaneously?
... Store data permanently?
... Send messages?

... Find web pages?
How do computers... Represent data?

Lowest level: modern computer = electronic machine

- Works by controlling the flow of electrons

Easy to recognize two conditions:

- presence of voltage – state “1”
- absence of voltage – state “0”

More difficult to detect and control analog values

<table>
<thead>
<tr>
<th>Digital Values</th>
<th>“0”</th>
<th>Illegal</th>
<th>“1”</th>
</tr>
</thead>
</table>

| Analog Values  | 0   | 0.5     | 2.4 | 2.9 Volts |

Representing Data

Recognize this photo? Was a vote cast or not?
Modern Computers = Binary Digital Systems

Digital system (not analog)  
• finite number of symbols

Binary (base two) system:  
• has two states: 0 and 1

Basic unit of information is *binary digit*, or *bit*  
• Can be represented in any technology with two states

Bits in Computers

Transistors and wires: electrons flowing or not?

Capacitors and memory: holding a charge or not?

Optical CD-ROMs and DVDs: Reflecting or not?

Hard disk drive: Magnetized north or south?
How Can Bits represent Data?

Everything in computer is represented with 1 and 0

- All text you see or type, movies you watch, music you listen to
- Everything stored on disk, CD, or flash drive
- Everything you send between computers
  - Email, web pages
- Even instructions computer uses to run programs

What kinds of data must bits represent?

Logical: True, False

- Straight-forward: Two states
- True: 1, False: 0

Numbers

- Signed, unsigned, integers, floating point, complex, rational, irrational, ...

Text

- Characters, words, strings, ...

Images

- Pixels, colors, shapes, movies ...

Sound

Instructions
Unsigned Integers

Approach 1: Non-positional notation

- Represent a number ("5") w/ string of ones ("11111")
- Problems?

Unsigned Integers: Weighted Positional Notation

- Position and base determines value of symbol
- Example: Decimal numbers (base-ten)
  - Base-ten implies digit can be one of 10 different symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9

Base-10

\[
329 = 3 \times 10^2 + 2 \times 10^1 + 9 \times 10^0
\]

Most significant

\[
2563 = 2 \times 10^3 + 5 \times 10^2 + 6 \times 10^1 + 3 \times 10^0
\]

Least significant

Position weight
Unsigned Integers: Weighted Positional Notation

- Same properties hold for binary numbers (base-two)
- Base-two: Each digit holds two different symbols: 0 or 1

Base-10

\[
\begin{array}{c|c|c|c}
10^3 & 10^2 & 10^1 & 10^0 \\
\hline
329 & & & \\
\end{array}
\]

3\times100 + 2\times10 + 9\times1 = 329

Base-2

\[
\begin{array}{c|c|c|c}
2^3 & 2^2 & 2^1 & 2^0 \\
\hline
101 & & & \\
\end{array}
\]

1\times4 + 0\times2 + 1\times1 = 5 (decimal)

Base-10

\[
\begin{array}{c|c|c|c}
10^3 & 10^2 & 10^1 & 10^0 \\
\hline
2563 & & & \\
\end{array}
\]

2\times1000 + 5\times100 + 6\times10 + 3\times1 = 2563

Base-2

\[
\begin{array}{c|c|c|c}
2^3 & 2^2 & 2^1 & 2^0 \\
\hline
1011 & & & \\
\end{array}
\]

1\times8 + 0\times4 + 1\times2 + 1\times1 = 11 decimal

Converting Binary to Decimal

Alternate way to view binary numbers

\[
\begin{array}{c|c|c|c}
2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
\hline
16 & 8 & 4 & 2 & 1 \\
\end{array}
\]

What would 01001 be in decimal?

\[
\begin{array}{c|c|c|c|c}
0 & 1 & 0 & 0 & 1 \\
\hline
= 9 \\
\end{array}
\]

00011?

\[
2 + 1 = 3
\]

10001?

\[
16 + 1 = 17
\]

11111?

\[
16 + 8 + 4 + 2 + 1 = 31 \text{ (also, 32 - 1)}
\]
**Converting Decimal to Binary**

How would you make decimal 5 in binary?
4 + 1 → 00101
16?
16 → 10000
21?
16 + 4 + 1 → 10101

How many different numbers can you make with 5 cards?
2 * 2 * 2 * 2 * 2 = 2^5 = 32

What is the largest number you can make with 5 cards?
2^5 - 1 = 31

What is the largest number you can make with N cards?
2^N - 1

**Counting in Binary**

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>

What do you notice about bits in binary number?
Practice Available with Scratch Game

Decision Tree for 0..31

How many questions needed to find answer between 0 and 31 (32 numbers)?
If 0 represents “no” or “false” and 1 represents “yes” or “true” what do you notice about the answers?

0 = no  1 = yes

What is the height of this tree? (i.e. how many questions?)
5 questions for 32 numbers; \( \log_2(32) = 5 \)
How many bits are needed to represent number between 0..31?
5 bits
Other Useful Units

Bytes
• Collection of 8 bits: 1101 0011
• How many different values represented in a byte?
  – \(2^8 = 256\) values
• Abbreviation: Use B for Bytes vs. b for bits

Hexadecimal numbers
• Base-16
• Why might hexadecimal numbers be useful?
  – Two hex digits per byte
• What characters should we use?
  – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f
• 1101 0011 in hex is Ox1d3

Today’s Summary

Today’s topics
• All computing devices equivalent given sufficient time and storage
• Abstractions of lower layers hide complexity
• Bits: Two states (on vs. off, true vs. false)
• Represent unsigned numbers with binary numbers
  – N bits can represent \(2^N\) different values

Reading: New packet
• Ch 1, 2, 3, 6, 7, 8 of “Code: The Hidden Language of Computer Hardware and Software” (light)
• Pages 130-151 of “Invitation to Computer Science”

Announcements
– Play binary number game and secret number card trick
– Exam 1 returned soon
– Grading of Project 1 proceeding
– Homework 5 due Friday: Essay about TED talk on technology