The Subject of this Class
The Subjects of this Class

0 1
But let your communication be, Yea, yea; Nay, nay: for whatsoever is more than these cometh of evil.

— Matthew 5:37
Engineering Works Because of Abstraction

Application Software
Operating Systems
Architecture
Micro-Architecture
Logic
Digital Circuits
Analog Circuits
Devices
Physics
Engineering Works Because of Abstraction

Application Software  
COMS 3157, 4156, et al.

Operating Systems  
COMS W4118

Architecture  
Second Half of 3827

Micro-Architecture  
Second Half of 3827

Logic  
First Half of 3827

Digital Circuits  
First Half of 3827

Analog Circuits  
ELEN 3331

Devices  
ELEN 3106

Physics  
ELEN 3106 et al.
Boring Stuff

Mailing list: csee3827-staff@lists.cs.columbia.edu
http://www.cs.columbia.edu/~sedwards/classes/2012/3827-spring/

Prof. Stephen A. Edwards  First Half of Semester
sedwards@cs.columbia.edu
462 Computer Science Building

Prof. Martha Kim  Second Half of Semester
martha@cs.columbia.edu
469 Computer Science Building

Lectures 10:10–11:25 AM, Tue, Thur, 207 Mathematics
Sep 4–Dec 6
Holidays: Nov 6 (Election Day), Nov 22 (Thanksgiving)
# Assignments and Grading

<table>
<thead>
<tr>
<th>Weight</th>
<th>What</th>
<th>When</th>
</tr>
</thead>
<tbody>
<tr>
<td>40%</td>
<td>Six homeworks</td>
<td>See Webpage</td>
</tr>
<tr>
<td>30%</td>
<td>Midterm exam</td>
<td>October 23rd</td>
</tr>
<tr>
<td>30%</td>
<td>Final exam</td>
<td>During Finals Week (Dec 14–21)</td>
</tr>
</tbody>
</table>

Homework is due at the beginning of lecture.

We will drop the lowest of your six homework scores; you can:

- skip
- omit
- forget
- ignore
- blow off
- screw up
- feed to dog
- flake out on
- sleep through

one with no penalty.
Rules and Regulations

You may collaborate with classmates on homework.

Each assignment turned in must be unique; work must ultimately be your own.

List your collaborators on your homework.

*Don’t cheat: if you’re stupid enough to try, we’re smart enough to catch you.*

Tests will be closed-book with a one-page “cheat sheet” of your own devising.
David Harris and Sarah Harris.

*Digital Design and Computer Architecture.*


Almost precisely right for the scope of this class: digital logic and computer architecture
There are only 10 types of people in the world:
Those who understand binary and those who don't.
Which Numbering System Should We Use?
Some Older Choices:

Roman: I II III IV V VI VII VIII IX X

Mayan: base 20, Shell = 0

Babylonian: base 60
The Decimal Positional Numbering System

Ten figures: 0 1 2 3 4 5 6 7 8 9

\[7 \times 10^2 + 3 \times 10^1 + 0 \times 10^0 = 730_{10}\]

\[9 \times 10^2 + 9 \times 10^1 + 0 \times 10^0 = 990_{10}\]

Why base ten?
# Hexadecimal, Decimal, Octal, and Binary

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Oct</th>
<th>Bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>111</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>10</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>11</td>
<td>1001</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>12</td>
<td>1010</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>13</td>
<td>1011</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>14</td>
<td>1100</td>
</tr>
<tr>
<td>D</td>
<td>13</td>
<td>15</td>
<td>1101</td>
</tr>
<tr>
<td>E</td>
<td>14</td>
<td>16</td>
<td>1110</td>
</tr>
<tr>
<td>F</td>
<td>15</td>
<td>17</td>
<td>1111</td>
</tr>
</tbody>
</table>
Binary and Octal

<table>
<thead>
<tr>
<th>Oct</th>
<th>Bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
</tbody>
</table>

\[
\text{PC} = 0 \times 2^{11} + 1 \times 2^{10} + 0 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0
\]

\[= 2 \times 8^3 + 6 \times 8^2 + 7 \times 8^1 + 5 \times 8^0\]

\[= 1469_{10}\]
Hexadecimal Numbers

Base 16: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Instead of groups of 3 bits (octal), Hex uses groups of 4.

\[
\begin{align*}
\text{CAFEOF00D}_{16} &= 12 \times 16^7 + 10 \times 16^6 + 15 \times 16^5 + 14 \times 16^4 + 15 \times 16^3 + 0 \times 16^2 + 0 \times 16^1 + 13 \times 16^0 \\
&= 3,405,705,229_{10}
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>A</th>
<th>F</th>
<th>E</th>
<th>F</th>
<th>0</th>
<th>0</th>
<th>D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>110010101011111110111100000000001101</td>
<td>Hex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Computers Rarely Manipulate True Numbers

Infinite memory still very expensive

Finite-precision numbers typical

32-bit processor: naturally manipulates 32-bit numbers

64-bit processor: naturally manipulates 64-bit numbers

How many different numbers can you represent with 5

- binary
- octal
- decimal
- hexadecimal
digits?
**Jargon**

- **Bit**  
  Binary digit: 0 or 1

- **Byte**  
  Eight bits

- **Word**  
  Natural number of bits for the processor, e.g., 16, 32, 64

- **LSB**  
  Least Significant Bit ("rightmost")

- **MSB**  
  Most Significant Bit ("leftmost")
Decimal Addition Algorithm

434
+628

4 + 8 = 12
Decimal Addition Algorithm

\[
\begin{array}{c}
1 \\
434 \\
+ 628 \\
\hline
2 \\
\end{array}
\]

\[
\begin{align*}
4 + 8 &= 12 \\
1 + 3 + 2 &= 6
\end{align*}
\]
Decimal Addition Algorithm

\[\begin{array}{c}
  1 \\
  434 \\
  + 628 \\
  \hline
  62
\end{array}\]

\[\begin{array}{c}
  4 + 8 = 12 \\
  1 + 3 + 2 = 6 \\
  4 + 6 = 10
\end{array}\]

\[\begin{array}{c|cccccccccc}
  & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
  0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
  1 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
  2 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
  3 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
  4 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
  5 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
  6 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
  7 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
  8 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
  9 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\
 10 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19
\end{array}\]
Decimal Addition Algorithm

\[
\begin{array}{c}
1 & 1 \\
4 & 3 & 4 \\
+ & 6 & 2 & 8 \\
\hline
0 & 6 & 2 \\
\end{array}
\]

\[
4 + 8 = 12 \\
1 + 3 + 2 = 6 \\
4 + 6 = 10
\]

+ 
<table>
<thead>
<tr>
<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>
</tr>
</thead>
</table>
0 |
1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
4 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
5 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
6 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
7 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
8 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
9 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
Decimal Addition Algorithm

\[ \begin{array}{c}
\text{1 1} \\
\text{434} \\
\text{+628} \\
\hline
\text{1062}
\end{array} \]

\[ \begin{array}{c}
4 + 8 = 12 \\
1 + 3 + 2 = 6 \\
4 + 6 = 10
\end{array} \]

+ | 0 1 2 3 4 5 6 7 8 9 \\
---|------------------
0 | 0 1 2 3 4 5 6 7 8 9 \\
1 | 1 2 3 4 5 6 7 8 9 10 \\
2 | 2 3 4 5 6 7 8 9 10 11 \\
3 | 3 4 5 6 7 8 9 10 11 12 \\
4 | 4 5 6 7 8 9 10 11 12 13 \\
5 | 5 6 7 8 9 10 11 12 13 14 \\
6 | 6 7 8 9 10 11 12 13 14 15 \\
7 | 7 8 9 10 11 12 13 14 15 16 \\
8 | 8 9 10 11 12 13 14 15 16 17 \\
9 | 9 10 11 12 13 14 15 16 17 18 \\
10 | 10 11 12 13 14 15 16 17 18 19
Binary Addition Algorithm

$$
\begin{array}{c}
10011 \\
\underline{+11001}
\end{array}
$$

1 + 1 = 10

<table>
<thead>
<tr>
<th>+</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>
Binary Addition Algorithm

\[
\begin{array}{c}
1 \\
10011 \\
+11001 \\
\hline
0
\end{array}
\]

\[
\begin{array}{c|c|c}
+ & 0 & 1 \\
\hline
0 & 00 & 01 \\
1 & 01 & 10 \\
10 & 10 & 11 \\
\end{array}
\]

\[
1 + 1 = \boxed{10}
\]

\[
1 + 1 + 0 = \boxed{10}
\]
Binary Addition Algorithm

\[
\begin{array}{c}
\begin{array}{c}
11 \\
10011 \\
+11001 \\
\hline
00
\end{array}
\end{array}
\]

\[
\begin{array}{c|cc|c}
+ & 0 & 1 \\
0 & 00 & 01 \\
1 & 01 & 10 \\
10 & 10 & 11 \\
\end{array}
\]

\[
\begin{array}{c}
1 + 1 = 10 \\
1 + 1 + 0 = 10 \\
1 + 0 + 0 = 01
\end{array}
\]
Binary Addition Algorithm

\[
\begin{array}{c}
\begin{array}{c}
011 \\
10011 \\
+11001 \\
\hline
100
\end{array}
\end{array}
\]

\[
\begin{array}{c}
1 + 1 = 10 \\
1 + 1 + 0 = 10 \\
1 + 0 + 0 = 01 \\
0 + 0 + 1 = 01
\end{array}
\]
Binary Addition Algorithm

\[
\begin{array}{c}
0011 \\
10011 \\
+11001 \\
\hline
1100
\end{array}
\]

\[
\begin{array}{c|cc}
+ & 0 & 1 \\
\hline
0 & 00 & 01 \\
1 & 01 & 10 \\
10 & 10 & 11
\end{array}
\]

\[
\begin{align*}
1 + 1 &= 10 \\
1 + 1 + 0 &= 10 \\
1 + 0 + 0 &= 01 \\
0 + 0 + 1 &= 01 \\
0 + 1 + 1 &= 10
\end{align*}
\]
Binary Addition Algorithm

\[
\begin{array}{c}
10011 \\
10011 \\
+11001 \\
\hline
101100
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
+ & 0 & 1 \\
\hline
0 & 00 & 01 \\
1 & 01 & 10 \\
10 & 10 & 11 \\
\hline
\end{array}
\]

1 + 1 = \textbf{10}
1 + 1 + 0 = \textbf{10}
1 + 0 + 0 = \textbf{01}
0 + 0 + 1 = \textbf{01}
0 + 1 + 1 = \textbf{10}
Signed Numbers: Dealing with Negativity

How should both positive and negative numbers be represented?
Signed Magnitude Numbers

You are most familiar with this: negative numbers have a leading $-$

In binary, a leading 1 means negative:

- $0000_2 = 0$
- $0010_2 = 2$
- $1010_2 = -2$
- $1111_2 = -7$
- $1000_2 = -0$?

Can be made to work, but addition is annoying:

- If the signs match, add the magnitudes and use the same sign.
- If the signs differ, subtract the smaller number from the larger; return the sign of the larger.
One’s Complement Numbers

Like Signed Magnitude, a leading 1 indicates a negative One’s Complement number.

To negate a number, complement (flip) each bit.

\[
\begin{align*}
0000_2 &= 0 \\
0010_2 &= 2 \\
1101_2 &= -2 \\
1000_2 &= -7 \\
1111_2 &= -0? \\
\end{align*}
\]

Addition is nicer: just add the one’s complement numbers as if they were normal binary.

Really annoying having a \(-0\): two numbers are equal if their bits are the same or if one is 0 and the other is \(-0\).
NOT ALL ZEROS ARE CREATED EQUAL

ZERO CALORIES. MAXIMUM PEPSI TASTE.
Two’s Complement Numbers

Really neat trick: make the most significant bit represent a *negative* number instead of positive:

\[
\begin{align*}
1101_2 &= -8 + 4 + 1 = -3 \\
1111_2 &= -8 + 4 + 2 + 1 = -1 \\
0111_2 &= 4 + 2 + 1 = 7 \\
1000_2 &= -8
\end{align*}
\]

Easy addition: just add in binary and discard any carry.

Negation: complement each bit (as in one’s complement) then add 1.

Very good property: no −0

Two’s complement numbers are equal if all their bits are the same.
## Number Representations Compared

<table>
<thead>
<tr>
<th>Bits</th>
<th>Binary</th>
<th>Signed Mag.</th>
<th>One’s Comp.</th>
<th>Two’s Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>-0</td>
<td>-7</td>
<td>-8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>-1</td>
<td>-6</td>
<td>-7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>-6</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>-7</td>
<td>-0</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Smallest number**: 0
**Largest number**: 15
Fixed-point Numbers

How to represent fractional numbers? In decimal, we continue with negative powers of 10:

\[ 31.4159 = 3 \times 10^1 + 1 \times 10^0 + 4 \times 10^{-1} + 1 \times 10^{-2} + 5 \times 10^{-3} + 9 \times 10^{-4} \]

The same trick works in binary:

\[ 1011.0110_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} \]
\[ = 8 + 2 + 1 + 0.25 + 0.125 \]
\[ = 11.375 \]
Interesting

The ancient Egyptians used binary fractions:

The Eye of Horus
Humans prefer reading decimal numbers; computers prefer binary. BCD is a compromise: every four bits represents a decimal digit.

<table>
<thead>
<tr>
<th>Dec</th>
<th>BCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000 0000</td>
</tr>
<tr>
<td>1</td>
<td>0000 0001</td>
</tr>
<tr>
<td>2</td>
<td>0000 0010</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0000 1000</td>
</tr>
<tr>
<td>9</td>
<td>0000 1001</td>
</tr>
<tr>
<td>10</td>
<td>0001 0000</td>
</tr>
<tr>
<td>11</td>
<td>0001 0001</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0001 1000</td>
</tr>
<tr>
<td>19</td>
<td>0001 1001</td>
</tr>
<tr>
<td>20</td>
<td>0010 0000</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

158
+242

\[
\begin{array}{c}
000101011000 \\
+001001000010 \\
\hline
1010 \text{ First group} \\
\end{array}
\]

\[
\begin{array}{c}
0110 \\
\hline
0100 \\
\text{Correction} \\
\end{array}
\]

\[
\begin{array}{c}
0100 \\
\hline
0000 \\
\text{Second group} \\
\end{array}
\]

\[
\begin{array}{c}
0100 \\
\hline
0000 \\
\text{Third group (No correction)} \\
\end{array}
\]

Result
BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

\[
\begin{align*}
158 & \quad +242 \\
\quad & \quad +242
\end{align*}
\]

\[
\begin{array}{c}
\begin{array}{c}
000101011000 \\
+001001000010
\end{array}
\hline
1010
\end{array}
\]

First group correction

\[
\begin{array}{c}
\begin{array}{c}
0110 \\
+0110
\end{array}
\hline
0000
\end{array}
\]

Result
BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

\[
\begin{array}{c}
1 \\
0001 \\
+0010 \\
0110 \\
1010 \\
\end{array}
\]

First group

\[
\begin{array}{c}
1010 \\
+0110 \\
1010 \\
\end{array}
\]

Correction

Second group

\[
\begin{array}{c}
0000 \\
0000 \\
0000 \\
\end{array}
\]

Result

\[
\begin{array}{c}
1 \quad 158 \\
+242 \\
0 \\
\end{array}
\]
BCD Addition

Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

```
1
158
+242
0
```

```
000101011000
+001001000010
1010 First group
0110 Correction
1010 0000
+ 0110 Correction
0100 0000
```

First group
Correction
Second group
Correction

Result
Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

\[
\begin{array}{c}
  11 \\
  158 \\
  +242 \\
  \hline
  00
\end{array}
\]

\[
\begin{array}{c}
  11 \\
  000101011000 \\
  +001001000010 \\
  \hline
  1010 \\
  +0110 \\
  10100000 \\
  +0110 \\
  \hline
  01000000
\end{array}
\]
Binary addition followed by a possible correction.

Any four-bit group greater than 9 must have 6 added to it.

Example:

```
  11  
+ 158  
+ 242  
   400
```

```
1  1
0001 0101 1100
+ 0010 0100 0010
1010 1010
+ 0110
1010 0000
+ 0110
0100 0000
```

First group
Correction
Second group
Correction
Third group
(No correction)
Result

```
0100 0000 0000
```