Fundamentals of Computer Systems
Thinking Digitally

Stephen A. Edwards

Columbia University

Summer 2016
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.
Prof. Stephen A. Edwards
sedwards@cs.columbia.edu
462 Computer Science Building

Lectures 5:30–8:40 PM, Mondays and Wednesdays
627 Mudd
May 23–June 29

<table>
<thead>
<tr>
<th>Weight</th>
<th>What</th>
<th>When</th>
</tr>
</thead>
<tbody>
<tr>
<td>40%</td>
<td>Homeworks</td>
<td>See Webpage</td>
</tr>
<tr>
<td>60%</td>
<td>Final exam</td>
<td>June 29th</td>
</tr>
</tbody>
</table>

Homework is due at the beginning of lecture.
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: Columbia Students Aren’t Cheaters

Tests will be closed-book; you may bring a single sheet of your own notes
Optional Texts: Alternative 1

No required text. One option:

- David Harris and Sarah Harris. *Digital Design and Computer Architecture*. Either 1st or 2nd ed.

Almost precisely right for the scope of this class: digital logic and computer architecture.
Optional Texts: Alternative 2

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>

DEC PDP-8/I, c. 1968

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

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

\[
= 1469_{10}
\]
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.

CAFEF00D_{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}

<table>
<thead>
<tr>
<th>C</th>
<th>A</th>
<th>F</th>
<th>E</th>
<th>F</th>
<th>0</th>
<th>0</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001010111111101111000000001101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hex

<table>
<thead>
<tr>
<th>3</th>
<th>1</th>
<th>2</th>
<th>7</th>
<th>7</th>
<th>5</th>
<th>7</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>31277570015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Binary

Octal
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?
<table>
<thead>
<tr>
<th>Jargon</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>Binary digit: 0 or 1</td>
</tr>
<tr>
<td>Byte</td>
<td>Eight bits</td>
</tr>
<tr>
<td>Word</td>
<td>Natural number of bits for the processor, e.g., 16, 32, 64</td>
</tr>
<tr>
<td>LSB</td>
<td>Least Significant Bit (&quot;rightmost&quot;)</td>
</tr>
<tr>
<td>MSB</td>
<td>Most Significant Bit (&quot;leftmost&quot;)</td>
</tr>
</tbody>
</table>
Decimal Addition Algorithm

\[
\begin{array}{cccccccccc}
& & & & & & & & & + & \\
\text{434} & + & \text{628} & & & & & & & & \text{1062} \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
4 & + & 8 & = & 12
\end{array}
\]
Decimal Addition Algorithm

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

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

<table>
<thead>
<tr>
<th>+</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>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>
</tr>
<tr>
<td>1</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>10</td>
</tr>
<tr>
<td>2</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>10</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
</tbody>
</table>
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

\[
\begin{array}{c}
1 \\
434 \\
+628 \\
\hline
062
\end{array}
\]

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

+ | 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
Decimal Addition Algorithm

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

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

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

\[
\begin{array}{c}
10011 \\
+ 11001 \\
\hline
10110
\end{array}
\]

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

1 + 1 = 10
Binary Addition Algorithm

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

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

\[
1 + 1 = 10
\]

\[
1 + 1 + 0 = 10
\]
Binary Addition Algorithm

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

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

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

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

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

1 + 1 = 10
1 + 1 + 0 = 10
1 + 0 + 0 = 01
0 + 0 + 1 = 01
Binary Addition Algorithm

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

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

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

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

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

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

\[ 1101_2 = -8 + 4 + 1 = -3 \]
\[ 1111_2 = -8 + 4 + 2 + 1 = -1 \]
\[ 0111_2 = 4 + 2 + 1 = 7 \]
\[ 1000_2 = -8 \]

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:

\[
1011.0110_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4}
\]
\[
= 8 + 2 + 1 + 0.25 + 0.125
\]
\[
= 11.375
\]
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>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>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>
Binary addition followed by a possible correction.

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

Example:

158
+242

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

First group

0110
Correction

0100
Correction

0100
Second group

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

Third group

0000

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

Result
BCD Addition

Binary addition followed by a possible correction.

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

Example:

\[
\begin{array}{c}
000101011000 \\
+001001000010 \\
\hline
1010 \quad \text{First group correction} \\
+0110 \\
\hline
0100 \quad \text{Second group correction} \\
0000 \\
\hline
\end{array}
\]

\[
158 + 242 \quad \text{Result}
\]
BCD Addition

Binary addition followed by a possible correction.

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

Example:

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

\[
\begin{array}{c}
1 \\
000101011000 \\
+001001000010 \\
10100000
\end{array}
\]

First group correction

Second group
BCD Addition

Binary addition followed by a possible correction.

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

Example:

\[
\begin{array}{c}
1 \\
158 \\
+242 \\
\hline
0
\end{array}
\]

\[
\begin{array}{c}
1 \\
000101011000 \\
+001001000010 \\
\hline
1010 \\
+0110 \\
\hline
10100000 \\
+0110 \\
\hline
10110000
\end{array}
\]

First group Correction
Second group Correction
**BCD Addition**

Binary addition followed by a possible correction.

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

Example:

\[
\begin{array}{c}
1 \\
+1 \\
000101011000 \\
+001001000010 \\
\hline
101000000 \\
+0110 \\
010000000
\end{array}
\]

First group
Correction
Second group
Correction
Third group

\[
\begin{array}{c}
11 \\
158 \\
+242 \\
\hline
00
\end{array}
\]
Binary addition followed by a possible correction.

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

Example:

\[
\begin{array}{c}
11 \\
158 \\
+242 \\
\hline
400
\end{array}
\]
Floating-Point Numbers: “Scientific Notation”

Greater dynamic range at the expense of precision
Excellent for real-world measurements

IEEE 754 Single-Precision (32-bit)
Sign 8-bit Exponent 23-bit Fraction

\[
11000000101100000000000000000000000000
\]

implicit \times \text{“excess 127”}

\[
\begin{align*}
= & -1.0110000_2 \\
& \times 2^{1000000012-127} \\
= & -1.375 \times 2^2 \\
= & -5.5
\end{align*}
\]
### ASCII For Representing Characters and Strings

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NUL</td>
<td>DLE</td>
<td>SP</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td>‘</td>
</tr>
<tr>
<td>1</td>
<td>SOH</td>
<td>DC1</td>
<td>!</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>STX</td>
<td>DC2</td>
<td>&quot;</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>ETX</td>
<td>DC3</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>EOT</td>
<td>DC4</td>
<td>$</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>ENQ</td>
<td>NAK</td>
<td>%</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>ACK</td>
<td>SYN</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>BEL</td>
<td>ETB</td>
<td>’</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>BS</td>
<td>CAN</td>
<td>(</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td>h</td>
</tr>
<tr>
<td>9</td>
<td>HT</td>
<td>EM</td>
<td>)</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td>i</td>
</tr>
<tr>
<td>A</td>
<td>LF</td>
<td>SUB</td>
<td>*</td>
<td>:</td>
<td>J</td>
<td>Z</td>
<td>j</td>
</tr>
<tr>
<td>B</td>
<td>VT</td>
<td>ESC</td>
<td>+</td>
<td>;</td>
<td>K</td>
<td>[</td>
<td>k</td>
</tr>
<tr>
<td>C</td>
<td>FF</td>
<td>FS</td>
<td>,</td>
<td>&lt;</td>
<td>L</td>
<td>\</td>
<td>l</td>
</tr>
<tr>
<td>D</td>
<td>CR</td>
<td>GS</td>
<td>–</td>
<td>=</td>
<td>M</td>
<td>]</td>
<td>m</td>
</tr>
<tr>
<td>E</td>
<td>SO</td>
<td>RS</td>
<td>.</td>
<td>&gt;</td>
<td>N</td>
<td>^</td>
<td>n</td>
</tr>
<tr>
<td>F</td>
<td>SI</td>
<td>US</td>
<td>/</td>
<td>?</td>
<td>O</td>
<td>__</td>
<td>o</td>
</tr>
</tbody>
</table>