ENGIN 112

Intro to Electrical and Computer Engineering

Lecture 2

Number Systems

Russell Tessier
KEB 309 G
tessier@ecs.umass.edu
Overview

° The design of computers
  • It all starts with numbers
  • Building circuits
  • Building computing machines

° Digital systems

° Understanding decimal numbers

° Binary and octal numbers
  • The basis of computers!

° Conversion between different number systems
Digital Computer Systems

- Digital systems consider *discrete* amounts of data.
- **Examples**
  - 26 letters in the alphabet
  - 10 decimal digits
- **Larger quantities can be built from discrete values:**
  - Words made of letters
  - Numbers made of decimal digits (e.g. 239875.32)
- **Computers operate on *binary* values (0 and 1)**
- **Easy to represent binary values electrically**
  - Voltages and currents.
  - Can be implemented using circuits
  - Create the building blocks of modern computers
Understanding Decimal Numbers

- Decimal numbers are made of decimal digits: (0,1,2,3,4,5,6,7,8,9)
- But how many items does a decimal number represent?
  - \(8653 = 8 \times 10^3 + 6 \times 10^2 + 5 \times 10^1 + 3 \times 10^0\)
- What about fractions?
  - \(97654.35 = 9 \times 10^4 + 7 \times 10^3 + 6 \times 10^2 + 5 \times 10^1 + 4 \times 10^0 + 3 \times 10^{-1} + 5 \times 10^{-2}\)
  - In formal notation \(-\) \((97654.35)_{10}\)
- Why do we use 10 digits, anyway?
Understanding Octal Numbers

- Octal numbers are made of octal digits: (0,1,2,3,4,5,6,7)
- How many items does an octal number represent?
  - \((4536)_8 = 4 \times 8^3 + 5 \times 8^2 + 3 \times 8^1 + 6 \times 8^0 = (1362)_{10}\)
- What about fractions?
  - \((465.27)_8 = 4 \times 8^2 + 6 \times 8^1 + 5 \times 8^0 + 2 \times 8^{-1} + 7 \times 8^{-2}\)
- Octal numbers don’t use digits 8 or 9
- Who would use octal number, anyway?
Understanding Binary Numbers

- Binary numbers are made of binary digits (bits):
  - 0 and 1

- How many items does an binary number represent?
  - \((1011)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = (11)_{10}\)

- What about fractions?
  - \((110.10)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2}\)

- Groups of eight bits are called a byte
  - \((11001001)_2\)

- Groups of four bits are called a nibble.
  - \((1101)_2\)
Why Use Binary Numbers?

- Easy to represent 0 and 1 using electrical values.
- Possible to tolerate noise.
- Easy to transmit data
- Easy to build binary circuits.

![Diagram showing binary signals and an AND gate with input 1 and output 0.](image)

- Fig. 1-3 Example of binary signals.
Learn to convert between bases.

Already demonstrated how to convert from binary to decimal.

Hexadecimal described in next lecture.
Convert an Integer *from* Decimal *to* Another Base

For each digit position:

1. Divide decimal number by the base (e.g. 2)
2. The *remainder* is the lowest-order digit
3. Repeat first two steps until no *divisor* remains.

Example for \((13)_{10}\):

<table>
<thead>
<tr>
<th>Integer Quotient</th>
<th>Remainder</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>13/2 = 6</td>
<td>½</td>
<td>a₀ = 1</td>
</tr>
<tr>
<td>6/2 = 3</td>
<td>0</td>
<td>a₁ = 0</td>
</tr>
<tr>
<td>3/2 = 1</td>
<td>½</td>
<td>a₂ = 1</td>
</tr>
<tr>
<td>1/2 = 0</td>
<td>½</td>
<td>a₃ = 1</td>
</tr>
</tbody>
</table>

Answer \((13)_{10} = (a₃ a₂ a₁ a₀)₂ = (1101)₂\)
Convert an Fraction *from* Decimal *to* Another Base

For each digit position:

1. Multiply decimal number by the base (e.g. 2)
2. The *integer* is the highest-order digit
3. Repeat first two steps until fraction becomes zero.

Example for $(0.625)_10$:

<table>
<thead>
<tr>
<th>Integer</th>
<th>Fraction</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.625 $\times$ 2 =</td>
<td>1 + 0.25</td>
<td>$a_1 = 1$</td>
</tr>
<tr>
<td>0.250 $\times$ 2 =</td>
<td>0 + 0.50</td>
<td>$a_2 = 0$</td>
</tr>
<tr>
<td>0.500 $\times$ 2 =</td>
<td>1 + 0</td>
<td>$a_3 = 1$</td>
</tr>
</tbody>
</table>

Answer $(0.625)_{10} = (0.a_1 a_2 a_3)_2 = (0.101)_2$
### The Growth of Binary Numbers

<table>
<thead>
<tr>
<th>n</th>
<th>$2^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$2^0=1$</td>
</tr>
<tr>
<td>1</td>
<td>$2^1=2$</td>
</tr>
<tr>
<td>2</td>
<td>$2^2=4$</td>
</tr>
<tr>
<td>3</td>
<td>$2^3=8$</td>
</tr>
<tr>
<td>4</td>
<td>$2^4=16$</td>
</tr>
<tr>
<td>5</td>
<td>$2^5=32$</td>
</tr>
<tr>
<td>6</td>
<td>$2^6=64$</td>
</tr>
<tr>
<td>7</td>
<td>$2^7=128$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>$2^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>$2^8=256$</td>
</tr>
<tr>
<td>9</td>
<td>$2^9=512$</td>
</tr>
<tr>
<td>10</td>
<td>$2^{10}=1024$</td>
</tr>
<tr>
<td>11</td>
<td>$2^{11}=2048$</td>
</tr>
<tr>
<td>12</td>
<td>$2^{12}=4096$</td>
</tr>
<tr>
<td>20</td>
<td>$2^{20}=1M$</td>
</tr>
<tr>
<td>30</td>
<td>$2^{30}=1G$</td>
</tr>
<tr>
<td>40</td>
<td>$2^{40}=1T$</td>
</tr>
</tbody>
</table>

- Mega
- Giga
- Tera
Binary Addition

- Binary addition is very simple.
- This is best shown in an example of adding two binary numbers...

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
+ & 1 & 0 & 1 & 1 & 1 \\
\hline
1 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}
\]

\[\text{carries}\]
Binary Subtraction

° We can also perform subtraction (with borrows in place of carries).
° Let’s subtract \((10111)_2\) from \((1001101)_2\)...

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 0 & 1 & 0 & 0 & 10 \\
0 & 0 & 10 & 10 & 0 & 0 & 10 & \hline
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \hline
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 1 & 1 & 0 & \hline
\end{array}
\]
Binary Multiplication

- Binary multiplication is much the same as decimal multiplication, except that the multiplication operations are much simpler...

\[
\begin{array}{c}
1 & 0 & 1 & 1 & 1 \\
\times & 1 & 0 & 1 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 \\
\hline
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0
\end{array}
\]
Convert an Integer \textit{from} Decimal \textit{to} Octal

For each digit position:

1. \textbf{Divide} decimal number by the base (8)
2. \textbf{The remainder} is the lowest-order digit
3. \textbf{Repeat} first two steps until no \textit{divisor} remains.

Example for \((175)_{10}:

<table>
<thead>
<tr>
<th>Integer Quotient</th>
<th>Remainder</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>(175/8 = 21)</td>
<td>(7/8)</td>
<td>(a_0 = 7)</td>
</tr>
<tr>
<td>(21/8 = 2)</td>
<td>(5/8)</td>
<td>(a_1 = 5)</td>
</tr>
<tr>
<td>(2/8 = 0)</td>
<td>(2/8)</td>
<td>(a_2 = 2)</td>
</tr>
</tbody>
</table>

Answer \((175)_{10} = (a_2 a_1 a_0)_2 = (257)_{8}\)
Convert an Fraction *from* Decimal *to* Octal

For each digit position:

1. **Multiply** decimal number by the base (e.g. 8)
2. The *integer* is the highest-order digit
3. **Repeat** first two steps until fraction becomes zero.

Example for \((0.3125)_{10}\):

<table>
<thead>
<tr>
<th>Integer</th>
<th>Fraction</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3125 x 8 =</td>
<td>2 + 5</td>
<td>(a_1 = 2)</td>
</tr>
<tr>
<td>0.5000 x 8 =</td>
<td>4 + 0</td>
<td>(a_2 = 4)</td>
</tr>
</tbody>
</table>

Answer \((0.3125)_{10} = (0.24)_8\)
Summary

- Binary numbers are made of binary digits (bits)
- Binary and octal number systems
- Conversion between number systems
- Addition, subtraction, and multiplication in binary