Binary Number System and Boolean Logic

Instructor: Nihshanka Debroy
(Slides borrowed from Tammy Bailey)
Bits

- Computers represent information as patterns of bits
- A **bit** (binary digit) is either 0 or 1
  - binary → “two states”
    - **true** and **false**, **on** and **off**, **open** and **closed**
- Storing a bit within a machine requires a binary device
- Binary devices can be in one of two possible states
  - a light switch is a binary device
    - holds one bit of information: **on** or **off**
  - A light dimmer is not a binary device
    - can be **on**, **off**, or some state in-between
Why Binary?

- Theoretically, no reason to prefer binary numbers!
- Practically, binary numbers more reliable - easier to represent only two values rather than multiple values (On vs. Off, High vs. Low, Left vs. Right, Positive vs. Negative, Clockwise vs. Counterclockwise)
- By having only two possible values it's easier to keep those values separated so there's no confusion.
- Voltages in an electric circuit, magnetic field directions
- Binary system gives greatest separation between values and so the greatest chance for reliability
Transistors in Computers

- 1955-75: magnetic cores to represent computer memories (0's and 1's represented by direction of magnetic field on the core – clockwise vs. anti-clockwise)

- Today – elementary block for computers is the transistor
  - **Transistor**: switch that can be electrically turned on/off
    - On – electricity allowed to pass through
    - Off – electricity not allowed to pass through

- Current/On : 1, No current/Off : 0
- Integrated circuit: large numbers of transistors on it
Boolean Logic

- AND, OR, NOT, NOR, NAND, XOR
- Each operator has a set of rules for combining two binary inputs
  - These rules are defined in a Truth Table
  - (This term is from the field of Logic)
- Each implemented in an electronic device called a gate
  - Gates operate on inputs of 0’s and 1’s
  - These are more basic than operations like addition
  - Gates are used to build up circuits that
    - Compute addition, subtraction, etc
    - Store values to be used later
    - Translate values from one format to another
Boolean operations

- Bits are manipulated using **Boolean operations**
  - AND, OR, XOR, NOT
A gate is a binary device that produces the output of a Boolean operation given the operation’s input values.

- **AND**
  - input
  - input
  - output

- **OR**
  - input
  - input
  - output

- **XOR**
  - input
  - input
  - output

- **NOT**
  - input
  - output
Truth Tables

### AND gate

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

### OR gate

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

### NOT gate

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit patterns

- Bits can be used to represent patterns.
- Specifically, any system or set of symbols can be translated into bit patterns:
  - patterns of ones and zeros
  - 10100001101
- Example: characters from any language alphabet.
- Require enough bits so that all symbols have a unique bit pattern to represent them:
  - How many bits are needed to represent the English alphabet?
How many bits?

• A bit pattern consisting of a single bit can represent at most two symbols
  – possible patterns are 0 and 1
• A bit pattern consisting of two bits can represent at most four symbols
  – possible patterns are 00, 01, 10 and 11
• In general, a bit pattern consisting of n bits can represent at most \(2^n\) symbols
• How many bits are needed to represent the English alphabet?
  – we can represent 26 symbols using 5 bits \((2^5=32)\)
  – 4 bits is not enough \((2^4=16)\)
Decimal (base 10) representation

- We commonly represent numbers in decimal (base 10).
- Numbers are represented using patterns of the digits \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}
- Position of each digit represents a power of ten.
- Example: Consider the decimal representation 2307

\[
2307 = 2 \times 10^3 + 3 \times 10^2 + 0 \times 10^1 + 7 \times 10^0
\]
A base $n$ system contains $n$ distinct symbols, the digits 0 through $n - 1$.

Numeric values greater than $n - 1$ are represented by a pattern of the $n$ symbols.

The value of any symbol in the string is found by multiplying that symbol by $n^p$, where $p$ is the distance from the rightmost symbol in the pattern.

Computers represent information using bit patterns, or binary (base 2) representation.

Numbers represented in base 2 are usually called binary numbers.

Overflow: when a value is too big to be represented (Example: a byte variable, to store the answer of $01111111 + 01111111$).
The binary representation contains two symbols: \{0, 1\}

Position of each symbol represents a power of two

What is the value of the binary representation 111?

\[
\begin{array}{ccc}
1 & 1 & 1 \\
\uparrow & \uparrow & \uparrow \\
\text{position:} & 2 & 1 & 0 \\
\end{array}
\]

\[
111 = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
= 1 \times 4 + 1 \times 2 + 1 \times 1 \\
= 4 + 2 + 1 = 7
\]
Binary (Base 2) Numbers

- Each digit in a binary number is chosen from two symbols: \{ 0, 1 \}
- The position (right to left) of each digit represents a power of two.
- **Example:** Convert binary number 1101 to decimal

\[
\begin{array}{cccc}
1 & 1 & 0 & 1 \\
\uparrow & \uparrow & \uparrow & \uparrow \\
\text{position:} & 3 & 2 & 1 & 0 \\
\end{array}
\]

\[
1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
= 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 8 + 4 + 1 = 13
\]
What is the value of the binary representation 1010?

\[
1010 = 1 	imes 2^3 + 0 	imes 2^2 + 1 	imes 2^1 + 0 	imes 2^0
\]

\[
= 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1
\]

\[
= 8 + 0 + 2 + 0 = 10
\]
## Powers of 2

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Power of 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$2^0$</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>$2^1$</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>$2^2$</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>$2^3$</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
<td>$2^4$</td>
</tr>
<tr>
<td>32</td>
<td>100000</td>
<td>$2^5$</td>
</tr>
<tr>
<td>64</td>
<td>1000000</td>
<td>$2^6$</td>
</tr>
<tr>
<td>128</td>
<td>10000000</td>
<td>$2^7$</td>
</tr>
</tbody>
</table>
## Famous Powers of 2

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
<th>Value Description</th>
<th>Representation</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kilobyte (KB)</td>
<td>1024</td>
<td>or $2^{10}$ bytes</td>
<td>1,024 bytes</td>
<td>Thousands of bytes</td>
</tr>
<tr>
<td>Megabyte (MB)</td>
<td>$1024^2$</td>
<td>or $2^{20}$ bytes</td>
<td>1,048,576 bytes</td>
<td>Millions of bytes</td>
</tr>
<tr>
<td>Gigabyte (GB)</td>
<td>$1024^3$</td>
<td>or $2^{30}$ bytes</td>
<td>1,073,741,824 bytes</td>
<td>Billions of bytes</td>
</tr>
<tr>
<td>Terabyte (TB)</td>
<td>$1024^4$</td>
<td>or $2^{40}$ bytes</td>
<td>1,099,511,627,776 bytes</td>
<td>Trillions of bytes</td>
</tr>
</tbody>
</table>

### Other Number Systems

<table>
<thead>
<tr>
<th>Binary</th>
<th>Octal</th>
<th>Decimal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>11</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>12</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>1011</td>
<td>13</td>
<td>11</td>
<td>B</td>
</tr>
<tr>
<td>1100</td>
<td>14</td>
<td>12</td>
<td>C</td>
</tr>
<tr>
<td>1101</td>
<td>15</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>1110</td>
<td>16</td>
<td>14</td>
<td>E</td>
</tr>
<tr>
<td>1111</td>
<td>17</td>
<td>15</td>
<td>F</td>
</tr>
</tbody>
</table>

Binary Addition

Also: $1 + 1 + 1 = 1$ with a carry of 1

Binary addition

- Represent sum of binary numbers as a binary number

**Decimal addition**

\[
\begin{align*}
1+1 &= 2 \\
1+1+1 &= 3
\end{align*}
\]

**Binary addition**

\[
\begin{align*}
1+1 &= 10 \\
1+1+1 &= 10+1 = 11
\end{align*}
\]
Adding binary numbers

\[
\begin{array}{c}
101 \\
+ \ 10 \\
\hline
111
\end{array}
\]

\[
\begin{array}{c}
101 \\
+ \ 11 \\
\hline
1000
\end{array}
\]

\[
\begin{array}{c}
111 \\
+ \ 110 \\
\hline
1101
\end{array}
\]

\[
\begin{array}{c}
10101010111 \\
+ \ 110000110 \\
\hline
11011011101
\end{array}
\]
## Converting decimal to binary

<table>
<thead>
<tr>
<th>Decimal</th>
<th>→ → conversion → →</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$0 \times 2^0$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$1 \times 2^0$</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$1 \times 2^1 + 0 \times 2^0$</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>$1 \times 2^1 + 1 \times 2^0$</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>$1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>$1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>$1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$</td>
<td>111</td>
</tr>
<tr>
<td>8</td>
<td>$1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$</td>
<td>1000</td>
</tr>
</tbody>
</table>
Converting decimal to binary

- Repeated division by two until the quotient is zero
- What is the binary representation of 30?

\[
\begin{array}{c}
\frac{30}{2} = 15 \quad \text{remainder } 0 \\
\frac{15}{2} = 7 \quad \text{remainder } 1 \\
\frac{7}{2} = 3 \quad \text{remainder } 1 \\
\frac{3}{2} = 1 \quad \text{remainder } 1 \\
\frac{1}{2} = 0 \quad \text{remainder } 1
\end{array}
\]

The binary representation of 30 is 11110.
Converting decimal to binary

- Repeated division by two until the quotient is zero
- What is the binary representation of 47?

\[
\begin{array}{c}
47 \\
2 \overline{)1} \\
\downarrow \\
1 \\
2 \overline{)2} \\
\downarrow \\
2 \\
2 \overline{)5} \\
\downarrow \\
5 \\
2 \overline{)11} \\
\downarrow \\
11 \\
2 \overline{)23} \\
\downarrow \\
23 \\
2 \overline{)47} \\
\end{array}
\]

remainder 1
remainder 0
remainder 1
remainder 1
remainder 1
remainder 1
remainder 1

101111
Problems

• Convert 1011000 to decimal representation

• Add the binary numbers 1011001 and 10101 and express their sum in binary representation

• Convert 77 to binary representation
Solutions

- Convert 1011000 to decimal representation

\[ 1011000 = 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \]
\[ = 64 + 16 + 8 = 88 \]

- Add the binary numbers 1011001 and 10101 and express their sum in binary representation

\[ \begin{array}{c}
1011001 \\
+ 10101 \\
\hline
1101110
\end{array} \]

- Convert 77 to binary representation: 1001101