Ceng 111 – Fall 2015
Week 2b

Digital Computation and
Binary Representation

Credit: Some slides are from the “Invitation to Computer Science” book by G. M. Schneider, J. L. Gersting and some from the “Digital Design” book by M. M. Mano and M. D. Ciletti.
An example problem: Water Tank

### Truth Table Representation

<table>
<thead>
<tr>
<th>HI</th>
<th>LO</th>
<th>Pump</th>
<th>Drain</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

- **Tank level is OK**
- **Low level, pump more in**
- **High level, drain some out**
- **Inputs cannot occur**

### Schematic Representation

![Diagram of Water Tank Schematic]
Boolean Logic/Algebra

Pump = HI’.LO
Drain = HI.LO’

Boolean formula describing the circuit.
The binary addition

Question (Binary notation): \[ 111010 + 11011 = ? \]
1-bit Half-adder

<table>
<thead>
<tr>
<th>Ai</th>
<th>Bi</th>
<th>Sum</th>
<th>Carry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
1-bit full-adder

\[
\begin{array}{c}
A \\ B \\ \text{Cin}
\end{array}
\begin{array}{c}
0 \\ 0 \\ 1
\end{array}
\begin{array}{c}
\text{Cout}
\end{array}
\begin{array}{c}
\text{S}
\end{array}
\begin{array}{c}
0 \\ 1 \\ 0 \\ 1
\end{array}
\begin{array}{c}
0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1
\end{array}

\begin{array}{c|c|c}
A & B & \text{Cin} & \text{S} & \text{Cout}
\hline
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1
\end{array}

Previously on CENG 111!
Today

- Quiz
- Binary representation of data
A computer

Devices

Gates

Transistors
Decoders

<table>
<thead>
<tr>
<th>S1</th>
<th>S0</th>
<th>Q0</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Decoders: An example

$C(X,Y,Z) = \Sigma m(3,5,6,7)$

$S(X,Y,Z) = \Sigma m(1,2,4,7)$

0 + 1 + 1 = 10

1 + 1 + 1 = 11
Decoders: An example

\[ C(X, Y, Z) = \Sigma m(3, 5, 6, 7) \]

\[ S(X, Y, Z) = \Sigma m(1, 2, 4, 7) \]
Multiplexers

\[ Q = S' D_0 + S D_1 \]
Multiplexers: An example

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 4.19
Diagram of a Typical Computer Circuit
BINARY REPRESENTATION

There are 10 types of people in the world: those who understand binary and those who don’t.
Data Representation

- Based on 1s and 0s
  - So, everything is represented as a set of binary numbers
- We will now see how we can represent:
  - Integers: 3, 1234435, -12945 etc.
  - Floating point numbers: 4.5, 124.3458, -1334.234 etc.
  - Characters: /, &, +, -, A, a, ^, 1, etc.
  - ...
Binary Representation of Numeric Information

- Decimal numbering system
  - Base-10
  - Each position is a power of 10
    \[ 3052 = 3 \times 10^3 + 0 \times 10^2 + 5 \times 10^1 + 2 \times 10^0 \]

- Binary numbering system
  - Base-2
  - Uses ones and zeros
  - Each position is a power of 2
    \[ 1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \]
Decimal-to-binary Conversion

Divide the number until zero:

- $35 / 2 = 17 \times 2 + 1$
- $17 / 2 = 8 \times 2 + 1$
- $8 / 2 = 4 \times 2 + 0$
- $4 / 2 = 2 \times 2 + 0$
- $2 / 2 = 1 \times 2 + 0$

Therefore, 35 has the binary representation: $100011$
Binary Representation of Numeric Information (continued)

- Representing integers
  - Decimal integers are converted to binary integers
  - **Question:** given \( k \) bits, what is the value of the largest integer that can be represented?
    - \( 2^k - 1 \)
      - Ex: given 4 bits, the largest is \( 2^4 - 1 = 15 \)

- Signed integers must also represent the sign (positive or negative) - **Sign/Magnitude notation**
Binary Representation of Numeric Information (continued)

- **Sign/magnitude notation**
  - 1 101 = -5
  - 0 101 = +5

- **Problems:**
  - Two different representations for 0:
    - 1 000 = -0
    - 0 000 = +0
  - Addition & subtraction require a watch for the sign! Otherwise, you get wrong results:
    - 0 010 (+2) + 1 010 (-2) = 1 100 (-4)
Binary Representation of Numeric Information (continued)

- **Two’s complement** instead of sign-magnitude representation
  - Positive numbers have a leading 0.
    - 5 => 0101
  - The representation for negative numbers is found by subtracting the absolute value from $2^N$ for an N-bit system:
    - -5 => $2^4 - 5 = 16 - 5 = 11_{10} => 1011_2$

- **Advantages:**
  - 0 has a single representation: +0 = 0000, -0 = 0000
  - Arithmetic works fine without checking the sign bit:
    - 1011 (-5) + 0110 (6) = 0001 (1)
    - 1011 (-5) + 0011 (3) = 1110 (-2)
Binary Representation of Numeric Information (continued)

- Shortcut to convert from “two’s complement”:
  - If the leading bit is zero, no need to convert.
  - If the leading bit is one, invert the number and add 1.

- What is our range?
  - With 2’s complement we can represent numbers from 
    \(-2^{N-1}\) to \(2^{N-1} - 1\) using \(N\) bits.
  - 8 bits: -128 to +127.
  - 16 bits: -32,768 to +32,767.
  - 32 bits: -2,147,483,648 to +2,147,483,647.

<table>
<thead>
<tr>
<th>Binary Number</th>
<th>Decimal Value</th>
<th>Value in Two’s Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>-8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>-7</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>-6</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>-5</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
<td>-4</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
<td>-3</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>-2</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>-1</td>
</tr>
</tbody>
</table>
Example:

- We want to compute: $12 - 6$
- $12 \Rightarrow 01100$
- $-6 \Rightarrow -(00110) \Rightarrow (11001)+1 \Rightarrow (11010)$

$12 - 6 =$

\[
\begin{array}{c}
01100 \\
+11010 \\
\hline
00110
\end{array}
\]

So, addition and subtraction operations are simpler in the Two’s Complement representation.
Due to its advantages, two’s complement is the most common way to represent integers on computers.
Why does Two’s Complement work?

1st Perspective:
We divide the range of unsigned numbers using N bits into two halves:
- 0 to $2^{N-1} - 1$ => Positive numbers
- $2^{N-1}$ to $2^N - 1$ => Negative numbers ($-2^{N-1}$ to -1)

Why?
- $i - j \mod 2^N = i + (2^N - j) \mod 2^N$

Example:
- Consider X and Y are positive numbers.
- $X + (-Y) = X + (2^N - Y)$
  $= 2^N - (Y - X) = -(Y - X) = X - Y$

<table>
<thead>
<tr>
<th>Binary Number</th>
<th>Decimal Value</th>
<th>Value in Two’s Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>-8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>-7</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>-6</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>-5</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
<td>-4</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
<td>-3</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>-2</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>-1</td>
</tr>
</tbody>
</table>
Why does Two’s Complement work?

A 2\textsuperscript{nd} perspective:

- Inversion and addition of a 1-bit correspond effectively to subtraction from 0 – i.e., negative of a number.
- Negative of a binary number X: 00...00\textsubscript{2} - X\textsubscript{2}
- Note that 00...00\textsubscript{2} = 11...11\textsubscript{2} + 1\textsubscript{2}
- In other words:
  - 00...00\textsubscript{2} - X\textsubscript{2} = (11...11\textsubscript{2}) - X\textsubscript{2} + 1.

\textit{(i.e., how we find two’s complement)}
Why does Two’s Complement work?

A smart trick used in mechanical calculators

- To subtract $b$ from $a$, invert $b$ and add that to $a$. Then discard the most significant digit.

Now

- Binary representation of real numbers
Binary Representation of Real Numbers

Conversion of the digits after the dot into binary:

1\textsuperscript{st} Way:
- \(0.375\rightarrow 0\times\frac{1}{2} + 1\times\frac{1}{4} + 1\times\frac{1}{8}\rightarrow 011\)

2\textsuperscript{nd} Way:
- Multiply by 2 and get the integer part until we get ‘0’ after the dot:
  - \(0.375 \times 2 = 0.750 = 0 + 0.750\)
  - \(0.750 \times 2 = 1.500 = 1 + 0.500\)
  - \(0.500 \times 2 = 1.000 = 1 + 0.000\)
Binary Representation of Real Numbers

- **Approach 1:** Use fixed-point
  - Similar to integers, except that there is a decimal point.
  - E.g: using 8 bits:
    \[
    \begin{array}{cccccccc}
    1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
    \end{array}
    \]
    Assumed decimal point
    \[
    = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 +
    1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}
    = 15.9375
    \]
Binary Representation of Real Numbers

Location of the decimal point changes the value of the number.

• E.g.: using 8 bits:

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

= \(1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0\)

\(1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}\) = 31.875

Assumed decimal point
Binary Representation of Real Numbers

- Problems with fixed-point:
  - Limited in the maximum and minimum values that can be represented.
  - For instance, using 32-bits, reserving 1-bit for the sign and putting the decimal point after 16 bits from the right, the maximum positive value that can be stored is slightly less than, $2^{15}$.
  - Allowing larger values gives away from the precision (the decimal part).
Binary Representation of Real Numbers

Solution: Use scientific notation: $a \times 2^b$ (or $\pm M \times B^{\pm E}$)

- Example: 5.75
  - 5 $\rightarrow$ 101
  - 0.75 $\rightarrow$ $\frac{1}{2} + \frac{1}{4} \rightarrow 2^{-1} + 2^{-2} \rightarrow 11$
  - 5.75 $\rightarrow$ 101.11 $\times$ 2^0

- Number is then normalized so that the first significant digit is immediately to the left of the binary point
  - Example: 1.0111 $\times$ 2^2

- We take and store the mantissa and the exponent.
This needs some standardization for:

- where to put the decimal point
- how to represent negative numbers
- how to represent numbers less than 1