Number systems

- Last lecture
  - Course overview
  - The Digital Age

- Today’s lecture
  - Binary numbers
  - Base conversion
  - Number systems
    - Twos-complement
  - A/D and D/A conversion
Digital

- Digital = discrete
  - Binary codes (example: BCD)
  - Decimal digits 0-9
  - DNA nucleotides

- Binary codes
  - Represent symbols using binary digits (bits)

- Digital computers:
  - I/O is digital
    - ASCII, decimal, etc.
  - Internal representation is binary
    - Process information in bits

<table>
<thead>
<tr>
<th>Decimal Symbols</th>
<th>BCD Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
</tbody>
</table>
The basics: Binary numbers

- **Bases we will use**
  - Binary: Base 2
  - Octal: Base 8
  - Hexadecimal: Base 16

- **Positional number system**
  - \(101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0\)
  - \(63_8 =\)
  - \(A1_{16} =\)

- **Addition and subtraction**

  \[
  \begin{array}{c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c}
    & & 1 & 0 & 1 & 1 \\
  + & & 1 & 0 & 1 & 0 \\
  \hline
  & & 1 & 1 & 0 & 0 & 1
  \end{array}
  \quad
  \begin{array}{c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c}
    & & 1 & 0 & 1 & 1 \\
  & & & & & & & & & & \text{---} \\
  \hline
  & & 0 & 1 & 1 & 0
  \end{array}
  \quad
  \begin{array}{c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c}
    & & 1 & 0 & 1 & 1 \\
  - & & 0 & 1 & 1 & 0 \\
  \hline
  & & 1 & 0 & 0 & 1
  \end{array}
  \quad
  \begin{array}{c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c@{}c}
    & & 1 & 0 & 1 & 1 \\
  & & & & & & & & & & \text{---} \\
  \hline
  & & 0 & 1 & 1 & 0
  \end{array}
  \]
Binary → hex/decimal/octal conversion

- Conversion from binary to octal/hex
  - Binary: 1001110001
  - Octal:
  - Hex:

- Conversion from binary to decimal
  - \(101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5_{10}\)
  - \(63.4_8 =\)
  - \(A1_{16} =\)
Decimal → binary/octet/hex conversion

<table>
<thead>
<tr>
<th>Binary</th>
<th></th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quotient</td>
<td>Remainder</td>
<td>Quotient</td>
</tr>
<tr>
<td>56 ÷ 2 =</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>28 ÷ 2 =</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>14 ÷ 2 =</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>7 ÷ 2 =</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3 ÷ 2 =</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 ÷ 2 =</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

56₁₀ = 111000₂
56₁₀ = 70₈

◆ Why does this work?
  ■ N = 56₁₀ = 111000₂
  ■ Q = N/2 = 56/2 = 111000/2 = 11100 remainder 0

◆ Each successive divide liberates an LSB
Number systems

- How do we write negative binary numbers?

- Historically: 3 approaches
  - Sign-and-magnitude
  - Ones-complement
  - Twos-complement

- For all 3, the most-significant bit (msb) is the sign digit
  - 0 \equiv \text{positive}
  - 1 \equiv \text{negative}

- Learn twos-complement
  - Simplifies arithmetic
  - Used almost universally
Sign-and-magnitude

- The most-significant bit (msb) is the sign digit
  - 0 \equiv \text{positive}
  - 1 \equiv \text{negative}

- The remaining bits are the number’s magnitude

- Problem 1: Two representations for zero
  - 0 = 0000 and also –0 = 1000

- Problem 2: Arithmetic is cumbersome

<table>
<thead>
<tr>
<th>Add</th>
<th>Subtract</th>
<th>Compare and subtract</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0100</td>
<td>4 0100 0100 –4 1100 1100</td>
</tr>
<tr>
<td>+ 3</td>
<td>+ 0011</td>
<td>–3 + 1011 –0011 +3 + 0011 –0011</td>
</tr>
</tbody>
</table>
Ones-complement

- **Negative number**: Bitwise complement positive number
  - 0011 \( \equiv 3_{10} \)
  - 1100 \( \equiv -3_{10} \)

- **Solves the arithmetic problem**

- **Remaining problem**: Two representations for zero
  - 0 = 0000 and also \(-0 = 1111\)
Twos-complement

- Negative number: Bitwise complement plus one
  - 0011 $\equiv 3_{10}$
  - 1101 $\equiv -3_{10}$

- Number wheel

- Only one zero!

- msb is the sign digit
  - 0 $\equiv$ positive
  - 1 $\equiv$ negative
Twos-complement (con’t)

- Complementing a complement → the original number
- Arithmetic is easy
  - Subtraction = negation and addition
  - Easy to implement in hardware

<table>
<thead>
<tr>
<th>Add</th>
<th>Invert and add</th>
<th>Invert and add</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>+ 3</td>
<td>+ 0011</td>
<td>− 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>− 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 3</td>
</tr>
</tbody>
</table>
Miscellaneous

- Twos-complement of non-integers
  - $1.6875_{10} = 01.1011_2$
  - $-1.6875_{10} = 10.0101_2$

- Sign extension
  - Write $+6$ and $-6$ as twos complement
    - $0110$ and $1010$
  - Sign extend to 8-bit bytes
    - $00000110$ and $11111010$

- Can’t infer a representation from a number
  - $11001$ is $25$ (unsigned)
  - $11001$ is $-9$ (sign magnitude)
  - $11001$ is $-6$ (ones complement)
  - $11001$ is $-7$ (twos complement)
Twos-complement overflow

- Summing two positive numbers gives a negative result
- Summing two negative numbers gives a positive result
Twos-complement overflow (cont’d)

- **Correct results**
  
  \[
  \begin{array}{ccc}
  1111 & -1 & 0011 & +3 \\
  +1010 & -6 & +0010 & +2 \\
  \end{array}
  \]

- **Incorrect results**
  
  \[
  \begin{array}{ccc}
  0110 & +6 & 1001 & -7 \\
  +0100 & +4 & +1010 & -6 \\
  \end{array}
  \]

- **Overflow condition**
  
  Carry from 2sb-msb and carry from msb-Cout are different

<table>
<thead>
<tr>
<th>2sb-msb</th>
<th>msb-Cout</th>
<th>Overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Gray and BCD codes

<table>
<thead>
<tr>
<th>Decimal Symbols</th>
<th>Gray Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0011</td>
</tr>
<tr>
<td>3</td>
<td>0010</td>
</tr>
<tr>
<td>4</td>
<td>0110</td>
</tr>
<tr>
<td>5</td>
<td>0111</td>
</tr>
<tr>
<td>6</td>
<td>0101</td>
</tr>
<tr>
<td>7</td>
<td>0100</td>
</tr>
<tr>
<td>8</td>
<td>1100</td>
</tr>
<tr>
<td>9</td>
<td>1101</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decimal Symbols</th>
<th>BCD Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
</tbody>
</table>
The physical world is analog

- Digital systems need to
  - Measure analog quantities
    - Speech waveforms, etc
  - Control analog systems
    - Drive motors, etc

- How do we connect the analog and digital domains?
  - Analog-to-digital converter (ADC or A/D)
    - Example: CD recording
  - Digital-to-analog converter (DAC or D/A)
    - Example: CD playback
Sampling

- Quantization
  - Conversion from analog to discrete values

- Quantizing a signal
  - We sample it

Datel Data Acquisition and Conversion Handbook
Conversion

- **Encoding**
  - Assigning a digital word to each discrete value
- **Encoding a quantized signal**
  - Encode the samples
  - Typically Gray or binary codes

Datel Data Acquisition and Conversion Handbook