Introduction to Computer Science
Lecture 1: Data Storage

Instructor: Tian-Li Yu
Taiwan Evolutionary Intelligence Laboratory (TEIL)
Department of Electrical Engineering
National Taiwan University
tianliyu@cc.ee.ntu.edu.tw

Slides made by Tian-Li Yu, Jie-Wei Wu, and Chu-Yu Hsu
Binary World

- **Bit**: binary digit (0/1)
- Simple, logical, and unambiguous
- Boolean operations & gates

<table>
<thead>
<tr>
<th>Inputs</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>

<table>
<thead>
<tr>
<th>Inputs</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>
Logical Gates

**XOR**

<table>
<thead>
<tr>
<th>Inputs</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>0</td>
</tr>
</tbody>
</table>

**NOT**

<table>
<thead>
<tr>
<th>Inputs</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>

- Logical vs. real world
  - To be or not to be → always **TRUE**.
Flip-Flop

- **Purpose**: to keep the state of output until the next excitement.
- **SR Flip-Flop**
  - Has two input lines: set and reset.
  - One input sets its stored value to 1.
  - The other input sets its stored value to 0.
  - While both inputs are 0, the most recently stored value is preserved.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>unchanged</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>undefined</td>
</tr>
</tbody>
</table>

Instructor: Tian-Li Yu
A Simple SR Flip-Flop Circuit
Another SR Flip-Flop Circuit
Hexadecimal Coding (Hex)

<table>
<thead>
<tr>
<th>Bit pattern</th>
<th>Hexadecimal representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>A</td>
</tr>
<tr>
<td>1011</td>
<td>B</td>
</tr>
<tr>
<td>1100</td>
<td>C</td>
</tr>
<tr>
<td>1101</td>
<td>D</td>
</tr>
<tr>
<td>1110</td>
<td>E</td>
</tr>
<tr>
<td>1111</td>
<td>F</td>
</tr>
</tbody>
</table>

- Binary is usually too long for human to remember.
- Binary to Hex is straightforward.
- $0010111010110101 \rightarrow 2EB5$. 
Main Memory Cells

- **Cell**: A unit of main memory (typically 8 bits which is one byte)

- **Diagram**:
  - High-order end: 0
  - Low-order end: 1
  - Most significant bit: 0
  - Least significant bit: 1

Instructor: Tian-Li Yu
Main Memory and Address

- One dimensional.
- Random accessible.
- Access the content by the address (practically, also in binary).
- Recall the pointer in C/C++.
Memory Techniques

- **Random Access Memory (RAM):** Memory in which individual cells can be easily accessed in any order.
  - **Static Memory (SRAM):** like flip-flop.
  - **Dynamic Memory (DRAM):** Tiny capacitors replenished regularly by refresh circuit.
  - **Synchronous DRAM (SDRAM)**
  - **Double Data Rate (DDR)**
  - **Dual/Triple channel**

- **Capacity**
  - **Kilobyte:** \(2^{10}\) bytes = 1,024 bytes \(\approx 10^3\) bytes.
  - **Megabyte:** \(2^{20}\) bytes = 1,048,576 bytes \(\approx 10^6\) bytes.
  - **Gigabyte:** \(2^{30}\) bytes = 1,073,741,824 bytes \(\approx 10^9\) bytes.
Mass Storage

- Properties (compared with main memory)
  - Larger capacity
  - Less volatility
  - Slower
  - On-line or off-line

- Types
  - Magnetic systems (hard disk, tape)
  - Optical systems (CD, DVD)
  - Flash drives
Magnetic Disk Storage System

- Head, track, sector, cylinder
- Access time = seek time + rotation delay / latency time.
- Transfer rate (SATA 1.5/3/6, etc.)
Optical Storage

Data recorded on a single track, consisting of individual sectors, that spirals toward the outer edge.

CD

Disk motion
Physical vs. Logical Records

- Logical records correspond to natural divisions within the data.
- Physical records correspond to the size of a sector.

- Files and file systems
- Fragmentation problem
- We talk about this later in OS.
Buffer

- **Purpose:** To synchronize (or to make compatible) different R/W mechanisms and rates.
- A memory area used for the temporary storage of data (usually as a step in transferring the data).
- Blocks of data compatible with physical records can be transferred between buffers and the mass storage system.
- Data in buffer can be referenced in terms of logical records.
Representing Text

- **ASCII** (American standard code for information interchange by ANSI): 7 bits (or 8 bits with a leading 0).
- **Unicode**: 16 bits.
- **ISO standard** (international organization of standardization): 32 bits.
ASCII Example

ASCII Code Chart

<table>
<thead>
<tr>
<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>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>SOH</td>
<td>STX</td>
<td>ETX</td>
<td>EOT</td>
<td>ENQ</td>
<td>ACK</td>
<td>BEL</td>
<td>BS</td>
<td>HT</td>
<td>LF</td>
<td>VT</td>
<td>FF</td>
<td>CR</td>
<td>SO</td>
<td>SI</td>
</tr>
<tr>
<td>DLE</td>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>DC4</td>
<td>NAK</td>
<td>SYN</td>
<td>ETB</td>
<td>CAN</td>
<td>EM</td>
<td>SUB</td>
<td>ESC</td>
<td>FS</td>
<td>GS</td>
<td>RS</td>
<td>US</td>
</tr>
<tr>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>(</td>
<td>)</td>
<td>*</td>
<td>+</td>
<td>,</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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>
<td>;</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
</tr>
<tr>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
</tr>
<tr>
<td>\</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

01001000 01100101 01101100 01101100 01101111 00101110
H e l l o .
Representing Numeric Values

**Base ten system**

<table>
<thead>
<tr>
<th>9</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

$10^2$, $10^1$, $10^0$

- Representation
- Position’s quantity

**Base two system**

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

$2^3$, $2^2$, $2^1$, $2^0$

- Representation
- Position’s quantity
### Representing Numeric Values

#### From Binary to Decimal

Instructor: Tian-Li Yu

<table>
<thead>
<tr>
<th>Base two system</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>\times</td>
<td>\times</td>
<td>\times</td>
<td>\times</td>
<td></td>
</tr>
<tr>
<td>\space{2}{3} = \space{2}{2} = \space{2}{1} = \space{2}{0}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- **Binary pattern**: 1011
- **Bit’s value**: \space{2}{3}, \space{2}{2}, \space{2}{1}, \space{2}{0}
- **Position’s quantity**: 11

Total 11
Representing Numeric Values

From Decimal to Binary

- Just as in decimal, keep dividing the number by 2 and record the remainders.
- Be careful about the order.

\[
2 \Big| \begin{array}{c}
11 \\
2 \\
5 \\
2 \\
2 \\
2 \\
1 \\
0 \\
0
\end{array} \quad \cdots \quad 1
\]

\[
11 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0
\]
Representing Images

- **Bit map techniques**
  - **Pixel**: picture element.
  - Colors: RGB, HSV, etc.
  - LCD, scanner, digital cameras, etc.

- **Vector techniques**
  - Scalable
  - TrueType, Postscript, SVG (scalable vector graphics), etc.
  - CAD, printers.
Representing Sounds

- Sampling
  - Sampling rate
  - Bit resolution
  - Bit rate (sampling rate $\times$ bit resolution)
- MIDI (synthesis)
Binary System Revisited

- **Addition**

\[
\begin{array}{c}
0 \\
+ 0 \\
\hline
0
\end{array} \\
\begin{array}{c}
0 \\
+ 1 \\
\hline
1
\end{array} \\
\begin{array}{c}
1 \\
+ 0 \\
\hline
1
\end{array} \\
\begin{array}{c}
1 \\
+ 1 \\
\hline
10
\end{array}
\]

- **Subtraction?**
  - Let’s first define negative numbers.
Two’s Complement Notation

- Range: $-2^{n-1} \sim 2^{n-1} - 1$

<table>
<thead>
<tr>
<th>Bit pattern</th>
<th>Value represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>-1</td>
</tr>
<tr>
<td>110</td>
<td>-2</td>
</tr>
<tr>
<td>101</td>
<td>-3</td>
</tr>
<tr>
<td>100</td>
<td>-4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit pattern</th>
<th>Value represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>1111</td>
<td>-1</td>
</tr>
<tr>
<td>1110</td>
<td>-2</td>
</tr>
<tr>
<td>1101</td>
<td>-3</td>
</tr>
<tr>
<td>1100</td>
<td>-4</td>
</tr>
<tr>
<td>1011</td>
<td>-5</td>
</tr>
<tr>
<td>1010</td>
<td>-6</td>
</tr>
<tr>
<td>1001</td>
<td>-7</td>
</tr>
<tr>
<td>1000</td>
<td>-8</td>
</tr>
</tbody>
</table>
Two’s Complement Encoding

- **Textbook’s way**
  - For positive $x$, $x \rightarrow$ binary encoding of $x$.
  - $-x \rightarrow$ binary encoding of $(2^n - x)$.

- **My way**
  - Copy the bits from right to left until a 1 has been copied.
  - Complement the remaining bits.
Subtraction in 2’s Complement

- Do it as usual in binary.

\[
\begin{array}{c@{}c@{}c@{}c@{}c}
3 & + & 2 & \Rightarrow & 0011 \\
\hline
? & \Rightarrow & 0010 & \Rightarrow & 5 \\
\end{array}
\]

\[
\begin{array}{c@{}c@{}c@{}c@{}c}
-3 & + & -2 & \Rightarrow & 1101 \\
\hline
? & \Rightarrow & 1110 & \Rightarrow & -5 \\
\end{array}
\]

\[
\begin{array}{c@{}c@{}c@{}c@{}c}
7 & + & -5 & \Rightarrow & 0111 \\
\hline
? & \Rightarrow & 1011 & \Rightarrow & 2 \\
\end{array}
\]

Instructor: Tian-Li Yu
### Excess Notation

<table>
<thead>
<tr>
<th>Bit pattern</th>
<th>Value represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>3</td>
</tr>
<tr>
<td>110</td>
<td>2</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>-1</td>
</tr>
<tr>
<td>010</td>
<td>-2</td>
</tr>
<tr>
<td>001</td>
<td>-3</td>
</tr>
<tr>
<td>000</td>
<td>-4</td>
</tr>
</tbody>
</table>

- **Conversion**
  \[ x \rightarrow (2^{n-1} + x) \mod 2^n \]

- **Addition**
  \[
  x + y \rightarrow \\
  (2^{n-1} + (2^{n-1} + x) + (2^{n-1} + y)) \mod 2^n \\
  = (2^{n-1} + x + y) \mod 2^n
  \]
Overflow

- **Overflow** occurs when the arithmetic result is out of the range of representation.

- Addition of two positive numbers
  - \(2 + 3 = 5 \rightarrow -3 \pmod{8}\)

- Addition of two negative numbers
  - \((-2) + (-3) = -5 \rightarrow 3 \pmod{8}\)
Fraction in Binary (Fixed-Point)

Base two system

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>1</th>
<th>.</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$2^2$</td>
<td>$2^1$</td>
<td>$2^0$</td>
<td>$2^{-1}$</td>
<td>$2^{-2}$</td>
<td>$2^{-3}$</td>
<td></td>
</tr>
</tbody>
</table>

| 4 | 0 | 1 | $\frac{1}{2}$ | 0 | $\frac{1}{8}$ |

Total $\frac{5}{8}$

- Binary pattern
- Bit’s value
- Position’s quantity
Float-Point Notation

- Why? (How to represent 0.000000000000001?)

- On most current 64-bit computers, the exponent takes 11 bits, and the mantissa takes 52 bits (IEEE 754 standard).
Decoding Floating-Point

- $01101011$
  \[\rightarrow (0)(110)(1011)\]
  \[\rightarrow (+)(+2)(1011)\]
  \[.1011 \rightarrow 10.11 \rightarrow 2 + \frac{1}{2} + \frac{1}{4} = 2\frac{3}{4}\]

- $10010011$
  \[\rightarrow (1)(001)(0011)\]
  \[\rightarrow (-)(-3)(0011)\]
  \[-.0011 \rightarrow -.0000011 \rightarrow -\left(\frac{1}{64} + \frac{1}{128}\right) = -\frac{3}{128}\]
Truncation Errors

- Required precision is beyond the limitation of the mantissa.

Original: $\frac{5}{8} = 2^{\frac{1}{2}}$

Base two: 10.101

Raw bit pattern: 10101

Sign bit | Exponent | Mantissa

The computer can only represent it as $2^{\frac{1}{2}}$. 
Normalized Form

- The most significant bit of mantissa is 1.
- 0’s floating-point representation is all zero.

Normalization

\[ 01100011 \rightarrow (0)(110)(0011) \rightarrow .0011 \times 2^2 \]
\[ \rightarrow .1100 \times 2^0 \rightarrow (0)(100)(1100) \rightarrow 01001100 \]

IEEE standard

- The left-most bit in mantissa is always 1 → omit it.
- An IEEE standard normalized form is \((s)(eee)(mmm)\)
  \[ \rightarrow (-1)^s \times 1.mmmm \times 2^{(eee-4)} \]
- \[01100011 \rightarrow (0)(110)(0011) \rightarrow 1.0011 \times 2^{(6-4)}\]
Loss of Digits

- $4 + \frac{1}{4} + \frac{1}{4}$

  \[
  \begin{align*}
  &4 + \frac{1}{4} + \frac{1}{4} \\
  &= 01111000 + 00111000 + 00111000 \\
  &= 01111000 + 01110000 + 01110000 \\
  &= 01111000 = 4 \text{!!!}
  \end{align*}
  \]

- $4 + \left(\frac{1}{4} + \frac{1}{4}\right)$

  \[
  \begin{align*}
  &4 + \left(\frac{1}{4} + \frac{1}{4}\right) \\
  &= 01111000 + (00111000 + 00111000) \\
  &= 01111000 + 01001000 \\
  &= 01111000 + 01110001 \\
  &= 01111001 = 4\frac{1}{2} \text{!!!}
  \end{align*}
  \]

- Just like when you use a calculator to do $10^{99} + 0.123 - 10^{99}$. 

Instructor: Tian-Li Yu

Data Storage
Data Compression

- Lossy vs. lossless
- Run-length encoding
- Frequency-dependent encoding
  - Huffman encoding
- Relative encoding / difference encoding
- Dictionary encoding
  - Adaptive dictionary encoding
  - LZW encoding
Huffman Encoding

- **Tradition encoding**
  - A → 00; B → 01; C → 10; D → 11.
  - 00000001010100001100000000111 (32 bits).

- **Huffman encoding**
  - Count occurrences: A(9); B(5); C(1); D(1).
  - Build a Huffman tree.

  ![Huffman Tree Diagram]

  - A → 0; B → 10; C → 110; D → 111.
  - 00010101001011000010111 (25 bits)
LZW Encoding

- A dictionary encoding which does not need to store the dictionary.

- `xyx` `xyx` `xyx` `xyx`
- 1
- 12
- 121
- `1213` → (knowing `xyx` forms a word).
- 12134
- 121343434

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>2</td>
</tr>
<tr>
<td>space</td>
<td>3</td>
</tr>
<tr>
<td>xyx</td>
<td>4</td>
</tr>
</tbody>
</table>

In reality, simply use ASCII code. So no addition dictionary is needed.

Decoding is similar.
Images, Audios, and Videos

- GIF: 256 colors, dictionary encoding
- JPEG
  - Lossy or lossless.
  - Discrete cosine transform.
  - Discard high-frequency information that is insensitive to human eyes.
- MP3
  - Temporal masking
  - Frequency masking
- MPEG
  - Relative encoding & other techniques.
Communication Errors

- Compression
  - Remove redundancy.

- Error detection & correction
  - Add redundancy to prevent errors.

- Error detection: Check code
  - Cannot correct errors, but can check if errors occur.
  - ID numbers
  - ISBN
  - Parity code

- Error correcting
  - Can correct errors (to some degree).
Taiwan ID

\[ Ca_1a_2a_3a_4a_5a_6a_7a_8a_9 \]

1. Convert the English letter \( C \) into a number \( xy \):

|   | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 4 | 8 | 9 | 0 | 1 | 2 | 5 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 3 |

2. \( d_1 = x + 9y \)

3. \( d_2 = \sum_{i=1}^{8} (9 - i) \cdot a_i = 8 \cdot a_1 + 7 \cdot a_2 + \ldots + 1 \cdot a_8 \)

4. Check code \( a_9 = 10 - ((d_1 + d_2) \mod 10) \)
The first 9 digits of ISBN-10 of the textbook is 0-273-75139

1. Compute \( S = 0 \cdot 10 + 2 \cdot 9 + 7 \cdot 8 + 3 \cdot 7 + 7 \cdot 6 + 5 \cdot 5 + 1 \cdot 4 + 3 \cdot 3 + 9 \cdot 2 = 193 \)

2. \( M = S \mod 11 = 6 \)

3. \( N = 11 - M = 5 \)
   - If \( N = 10 \), the check code is \( X \).
   - If \( N = 11 \), the check code is \( 0 \).
   - Otherwise, the check code is the number \( N \)

Parity Bits

- Add an additional bit to make the whole odd number of 1s.
- Communication
- RAID (redundant array of independent disks) techniques

Diagram:
- ASCII A containing an even number of 1s
  - Total pattern has an odd number of 1s
- ASCII A containing an odd number of 1s
  - Total pattern has an odd number of 1s
An Error-Correcting Code (ECC)

- (3,1)-repetition code (can correct 1-bit errors)

<table>
<thead>
<tr>
<th>Triplet received</th>
<th>Interpret as</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0 (error free)</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>1 (error free)</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
</tr>
</tbody>
</table>
Another Error-Correcting Code (ECC)

- Maximized Hamming distances among symbols (at least 3).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>000000</td>
</tr>
<tr>
<td>B</td>
<td>001111</td>
</tr>
<tr>
<td>C</td>
<td>010011</td>
</tr>
<tr>
<td>D</td>
<td>011100</td>
</tr>
<tr>
<td>E</td>
<td>100110</td>
</tr>
<tr>
<td>F</td>
<td>101001</td>
</tr>
<tr>
<td>G</td>
<td>110101</td>
</tr>
<tr>
<td>H</td>
<td>111010</td>
</tr>
</tbody>
</table>

- Received 010100.

- 010100 → D.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>H</td>
<td>4</td>
</tr>
</tbody>
</table>
## License

<table>
<thead>
<tr>
<th>Page</th>
<th>File</th>
<th>Licensing</th>
<th>Source/ author</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td><img src="http://commons.wikimedia.org/wiki/File:ASCII_Code_Chart.svg" alt="ASCII Code Chart" /></td>
<td><img src="http://creativecommons.org/licenses/publicdomain/" alt="License" /></td>
<td>Wikimedia., Author: Anomie, Source: <a href="http://commons.wikimedia.org/wiki/File:ASCII_Code_Chart.svg">http://commons.wikimedia.org/wiki/File:ASCII_Code_Chart.svg</a>, Date: 2012/02/05, This file is ineligible for copyright and therefore in the public domain</td>
</tr>
</tbody>
</table>