Digital Data

- Binary and hexadecimal representations
- Different types of numbers: natural numbers, integers, real numbers
- ASCII code and UNICODE
- Sound: Sampling, and Quantitizing
- Images
We are used to counting in base 10:

1000  100  10  1
<br>

Example:

1 7 3 2
<br>

1x1000+7x100+3x10+2x1 = 1732

Computers use a different system: base 2:

1024  512  256  128  64  32  16  8  4  2  1
<br>

Example:

1 1 0 1 1 0 0 0 1 0 0
<br>

2^10 + 2^8 + 2^6 + 2^0 = 1024 + 512 + 256 + 1 = 1732

<table>
<thead>
<tr>
<th>Base 10</th>
<th>Base 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
</tr>
<tr>
<td>17</td>
<td>10001</td>
</tr>
<tr>
<td>18</td>
<td>10010</td>
</tr>
<tr>
<td>19</td>
<td>10011</td>
</tr>
<tr>
<td>20</td>
<td>10100</td>
</tr>
<tr>
<td>21</td>
<td>10101</td>
</tr>
<tr>
<td>22</td>
<td>10110</td>
</tr>
<tr>
<td>23</td>
<td>10111</td>
</tr>
<tr>
<td>24</td>
<td>11000</td>
</tr>
<tr>
<td>25</td>
<td>11001</td>
</tr>
<tr>
<td>26</td>
<td>11010</td>
</tr>
<tr>
<td>27</td>
<td>11011</td>
</tr>
<tr>
<td>28</td>
<td>11100</td>
</tr>
<tr>
<td>29</td>
<td>11101</td>
</tr>
<tr>
<td>30</td>
<td>11110</td>
</tr>
<tr>
<td>31</td>
<td>11111</td>
</tr>
</tbody>
</table>
**Conversion**

From base 2 to base 10:

```
1 1 1 0 1 0 1 1 1 0 0
```

1 x 1024 + 1 x 512 + 1 x 256 + 1 x 128 + 0 x 64 + 1 x 32 + 0 x 16 + 1 x 8 + 0 x 4 + 0 x 2 + 1 x 1 = 1876

From base 10 to base 2:

```
1877  52  =  938  Remainder 1
938  52  =  469  Remainder 0
469  52  =  234  Remainder 1
234  52  =  117  Remainder 0
117  52  =  58  Remainder 1
58  52  =  29  Remainder 0
29  52  =  14  Remainder 1
14  52  =  7  Remainder 1
7  52  =  3  Remainder 1
3  52  =  1  Remainder 1
1  52  =  0  Remainder 1
```

1877  (base 10) = 11101010101  (base 2)

**Facts about Binary Numbers**

- Each “digit” of a binary number (each 0 or 1) is called a bit
- 1 byte = 8 bits
- 1 KB = 1 kilobyte = 2^10 bytes (=1 thousand bytes)
- 1 MB = 1 Megabyte = 2^20 bytes (=1 million bytes)
- 1 GB = 1 Gigabyte = 2^30 bytes (=1 billion bytes)
- 1 TB = 1 Terabyte = 2^40 bytes (=1 trillion bytes)
- A byte can represent numbers up to 255: 11111111 (base 2) = 255 (base 10)
- The largest number represented by a binary number of size N is 2^N - 1

**Big Data: Volume**

- 1000 bytes = 1 KB
- 1000 MB = 1 GB
- 1000 GB = 1 TB
- 1000 TB = 1 PB
- 1000 PB = 1 EB
- 1000 EB = 1 ZB
- 1000 ZB = 1 YB
Hexadecimal numbers

While base 10 and base 2 are the most common bases used to represent numbers, others are also possible: base 16 is another popular one, corresponding to hexadecimal numbers.

<table>
<thead>
<tr>
<th>Octal</th>
<th>Decimal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

The “digits” are: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Example:

\[ 3A_{16} = 3\times16^1 + 10\times16^0 = 48 + 10 = 58 \]
Hexadecimal numbers

Everything we have learned in base 10 should be studied again in other bases!!

Example: multiplication table in base 16:

<table>
<thead>
<tr>
<th></th>
<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>0</td>
<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>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>1</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>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>1A</td>
<td>1C</td>
<td>1E</td>
<td>20</td>
<td>22</td>
<td>24</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
<td>24</td>
<td>27</td>
<td>2A</td>
<td>2D</td>
<td>2G</td>
<td>30</td>
<td>33</td>
<td>36</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>36</td>
<td>40</td>
<td>44</td>
<td>48</td>
<td>52</td>
<td>56</td>
<td>60</td>
<td>64</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>60</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
<td>48</td>
<td>54</td>
<td>60</td>
<td>66</td>
<td>72</td>
<td>78</td>
<td>84</td>
<td>90</td>
<td>96</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
<td>56</td>
<td>63</td>
<td>70</td>
<td>77</td>
<td>84</td>
<td>91</td>
<td>98</td>
<td>A5</td>
<td>A2</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>48</td>
<td>56</td>
<td>64</td>
<td>72</td>
<td>80</td>
<td>88</td>
<td>96</td>
<td>A4</td>
<td>B2</td>
<td>C0</td>
<td>C8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>72</td>
<td>81</td>
<td>90</td>
<td>99</td>
<td>A8</td>
<td>B7</td>
<td>C6</td>
<td>D5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>A0</td>
<td>B0</td>
<td>C0</td>
<td>D0</td>
<td>E0</td>
<td>F0</td>
<td>10</td>
</tr>
</tbody>
</table>

Conversion: From base 2 to base 16, and back

This is in fact easy?

- From base 2 to base 16:
  
  Example: DEFHED
  
  Step 1: break into groups of 4 (starting from the right)
  
  DEFHED
  
  Step 2: pad with 0, if needed
  
  DEFHED
  
  Step 3: convert each group of 4, using table
  
  11   9   13   14
  
  Step 4: regroup
  
  D914
  
  11011000100 (base 2) = D914 (base 16)
Conversion: From base 10 to base 2.

Example: 4FD

Step 1: split
4 F D

Step 2: convert each “digit”, using table:

<table>
<thead>
<tr>
<th>4F</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1111</td>
</tr>
</tbody>
</table>

Step 3: Remove leading 0, if needed
100 1111 1101

Step 4: regroup:
10011111101

4FD (base 10) = 10011111101 (base 2)

Digital Data

Binary and hexadecimal representations

Different types of numbers: natural numbers, integers, real numbers

ASCII code and UNICODE

Sound: Sampling, and Quantitizing

Images

The different set of numbers

<table>
<thead>
<tr>
<th>Set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Natural numbers 1, 2, 3, 4, ...</td>
</tr>
<tr>
<td>Z</td>
<td>Integers ..., -4, -3, -2, -1, 0, 1, 2, 3, 4, ...</td>
</tr>
<tr>
<td>Q</td>
<td>Rational numbers ( \frac{a}{b} ) where ( a ) and ( b ) are integers and ( b ) is not zero</td>
</tr>
<tr>
<td>R</td>
<td>Real numbers The limit of a convergent sequence of rational numbers</td>
</tr>
<tr>
<td>C</td>
<td>Complex numbers ( a + ib ) where ( a ) and ( b ) are real numbers and ( i ) is the square root of (-1)</td>
</tr>
</tbody>
</table>
Representing Integers

Unsigned integers (natural numbers):

- **Sizes**
  - Char: 1 bit
  - Unsigned short: 16 bits (2 bytes)
  - Unsigned int: 32 bits (4 bytes)

Signed integers

- **S:**
  - sign bit: 0 means positive, 1 means negative
- **Num:**
  - If $s = 0$, direct representation of the number in binary form
  - If $s = 1$, two's complement of the number

**Sizes**
- Char: 1 bit
- Short: 16 bits (2 bytes)
- Int: 32 bits (4 bytes)

Representing Integers: two’s complement

The two's complement of an $N$-bit number is defined as its complement with respect to $2^N$.

The sum of a number and its two's complement is $2^N$.

For instance, for the three-bit number 010, the two's complement is 110, because

- $010 + 110 = 1000$ (where $2^3 = 8$).

The two's complement is calculated by inverting the bits and adding one.
IEEE Standard 754

- Established in 1985 as a uniform standard for floating point arithmetic
- Before that, many idiosyncratic formats
- Supported by all major CPUs

Driven by Numerical Concerns

- Nice standards for rounding, overflow, underflow
- Hard to make go fast
- Numerical analysts predominated over hardware types in defining standard

IEEE Floating Point Representation

Numerical Form

\[ (-1)^s \times M \times 2^E \]

- Sign bit \( s \) determines whether number is negative or positive
- Significand \( M \) normally a fractional value in range \([1.0, 2.0)\)
- Exponent \( E \) weighs value by power of two

Encoding

- MSB is sign bit
- \( \text{exp} \) field encodes \( E \)
- \( \text{frac} \) field encodes \( M \)
Encoding
➢ MSB is sign bit
➢ exp field encodes $E$
➢ frac field encodes $M$

Sizes
➢ Single precision: 8 exp bits, 23 frac bits
   (32 bits total)
➢ Double precision: 11 exp bits, 52 frac bits
   (64 bits total)
➢ Extended precision: 15 exp bits, 63 frac bits
   • Only found in Intel-compatible machines
   • Stored in 80 bits (1 bit wasted)

IEEE Floating Point Representation

Special value:
➢ exp = 011...1
    ➢ exp = 011...1, frac = 000...0
        • Represents value = ($\infty$)
        • Operation that overflows
        • Both positive and negative
          ➢ e.g. $1.000\ldots0 = -1.000\ldots0 = +\infty, 1.000\ldots0 = -\infty$
➢ exp = 011...1, frac ≠ 000...0
    • Not-a-Number (NaN)
    • Represents case when no numeric value can be determined
      ➢ e.g. $\sqrt{-1}$, $\infty$ - $\infty$

Floating Point Operations

Conceptual View
• First compute exact result
• Make it fit into desired precision
  ➢ Possibly overflow if exponent too large
  ➢ Possibly round to fit into frac

Rounding Modes (illustrate with $\$ rounding)

<table>
<thead>
<tr>
<th></th>
<th>$1.40$</th>
<th>$1.60$</th>
<th>$1.50$</th>
<th>$2.50$</th>
<th>$-1.50$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round down (-)</td>
<td>$1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$2$</td>
<td>$-2$</td>
</tr>
<tr>
<td>Round up (+)</td>
<td>$1$</td>
<td>$2$</td>
<td>$2$</td>
<td>$2$</td>
<td>$2$</td>
</tr>
<tr>
<td>Nearest Even</td>
<td>$1$</td>
<td>$2$</td>
<td>$2$</td>
<td>$2$</td>
<td>$-2$</td>
</tr>
</tbody>
</table>

Note:
1. Round down: rounded result is close to but no greater than true result.
2. Round up: rounded result is close to but no less than true result.
Computers encounter noise!

The Ariane 5 tragedy: On June 1996, the first Ariane 5 was launched and exploded after 37 seconds. The failure of the Ariane 501 was caused by the complete loss of guidance and altitude information 37 seconds after start... due to a numerical error.

Unwanted noise

Digital Data

Binary and hexadecimal representations

Different types of numbers: natural numbers, integers, real numbers

ASCII code and UNICODE

Sound: Sampling, and Quantitizing

Images

ASCII

American Standard Code for Information Interchange

So far, we have seen how computers can handle numbers. What about letters / characters?

The ASCII code was designed for that: it assigns a number to each character:

A-Z: 65-90
a-z: 97-122
0-9: 48-57
ASCII
American Standard Code for Information Interchange

UNICODE

ASCII only contains 127 characters (though an extended version exists with 257 characters).

This is by far not enough as it is too restrictive to the English language.

UNICODE was developed to alleviate this problem: the latest version, UNICODE 14.0 (September 2021) contains more than 140,000 characters, covering most existing languages.

For more information, see:
http://www.unicode.org/versions/Unicode14.0.0/

Digital Data

Binary and hexadecimal representations

Different types of numbers: natural numbers, integers, real numbers

ASCII code and UNICODE

Sound: Sampling, and Quantizing

Images
Sound is produced by the vibration of a medium like air or water. Audio refers to the sound within the range of human hearing. Naturally, a sound signal is analog, i.e. continuous in both time and amplitude.

To store and process sound information in a computer or to transmit it through a computer network, we must first convert the analog signal to digital form using an analog-to-digital converter (ADC); the conversion involves two steps:

1. Sampling,
2. Quantization.

### Digital Sound

![Digital Sound Diagram]

Sound is produced by the vibration of a medium like air or water. Audio refers to the sound within the range of human hearing. Naturally, a sound signal is analog, i.e. continuous in both time and amplitude.

To store and process sound information in a computer or to transmit it through a computer network, we must first convert the analog signal to digital form using an analog-to-digital converter (ADC); the conversion involves two steps:

1. Sampling,
2. Quantization.

### Sampling

**Sampling** is the process of examining the value of a continuous function at regular intervals.

Sampling usually occurs at uniform intervals, which are referred to as sampling intervals. The reciprocal of sampling interval is referred to as the sampling frequency or sampling rate.

If the sampling is done in the time domain, the unit of sampling interval is second and the unit of sampling rate is Hz, which means cycles per second.

Note that choosing the sampling rate is not innocent:

**A higher sampling rate usually allows for a better representation of the original sound wave.** However, when the sampling rate is set to twice the highest frequency in the signal, the original sound wave can be reconstructed without loss from the samples. This is known as the Nyquist theorem.
Quantization

Quantization is the process of limiting the value of a sample of a continuous function to one of a predetermined number of allowed values, which can then be represented by a finite number of bits.

The number of bits used to store each intensity defines the accuracy of the digital sound:

Adding one bit makes the sample twice as accurate.

Audio Sound

Sampling:
The human ear can hear sound up to 20,000 Hz; a sampling rate of 40,000 Hz is therefore sufficient. The standard for digital audio is 44,100 Hz.

Quantization:
The current standard for the digital representation of audio sound is to use 16 bits (i.e. 65,536 levels, half positive and half negative)
How much space do we need to store one minute of music?

- 60 seconds
- 44,100 samples
- 16 bits (2 bytes) per sample
- 2 channels (stereo)

\[ S = 60 \times 44,100 \times 2 \times 2 = 10,934,000 \text{ bytes} = 10 \text{ MB} \]

1 hour of music would be more than 600 MB!
DIGITAL RECORDING

Advantages of digital recording:
- Faithful
- Can make multiple identical copies
- Can be processed
- Compression (e.g., MP3)

Digital Data

Binary and hexadecimal representations

Different types of numbers: natural numbers, integers, real numbers

ASCII code and Unicode

Sound: Sampling and Quantizing

Images

Digital Images
Digital Images

Sampling:
Images are broken down into little squares: pixels
Resolution: Number of squares along each direction

Quantization:
Each pixel is characterized either as

- A binary number (0 or 1) to indicate black or white
- A natural number between 0 and 255, to indicate a gray scale
- A set of three numbers, each between 0 and 255, to indicate the amount of Red (R), Green (G), and Blue (B)

"True Color": a pixel is represented by 24 bits, corresponding to 16,777,216 possible colors
Digital Images

The CMYK color model (used by color printers)