Unicode

<table>
<thead>
<tr>
<th>Name</th>
<th>UTF-8</th>
<th>UTF-16</th>
<th>UTF-16BE</th>
<th>UTF-16LE</th>
<th>UTF-32</th>
<th>UTF-32BE</th>
<th>UTF-32LE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smallest code point</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>Largest code point</td>
<td>10FFFF</td>
<td>10FFFF</td>
<td>10FFFF</td>
<td>10FFFF</td>
<td>10FFFF</td>
<td>10FFFF</td>
<td>10FFFF</td>
</tr>
<tr>
<td>Code unit size</td>
<td>8 bits</td>
<td>16 bits</td>
<td>16 bits</td>
<td>16 bits</td>
<td>32 bits</td>
<td>32 bits</td>
<td>32 bits</td>
</tr>
<tr>
<td>Byte order</td>
<td>N/A</td>
<td>&lt;BOM&gt;</td>
<td>big-endian</td>
<td>little-endian</td>
<td>&lt;BOM&gt;</td>
<td>big-endian</td>
<td>little-endian</td>
</tr>
<tr>
<td>Fewest bytes per character</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Most bytes per character</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Normalized & Denormalized Representations

- Normalized: $1.011 \times 2^2$ \hspace{1cm} (1.F \times 2^E)
- Denormalized: 101.1
- Zero is represented by making the sign bit either 1 or 0 and all the other bits 0.
  - 1 00000000 00000000000000000000000 or 0 00000000 00000000000000000000000
- Subnormal Numbers: when exponent is zero, we consider the leading bit to be 0 \((0.F \times 2^E)\)
### Special Cases

<table>
<thead>
<tr>
<th>Exponent</th>
<th>Fraction</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>nonzero</td>
<td>subnormal</td>
</tr>
<tr>
<td>1-254</td>
<td>anything</td>
<td>normal</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>infinity</td>
</tr>
<tr>
<td>255</td>
<td>nonzero</td>
<td>NaN</td>
</tr>
</tbody>
</table>
Special Cases

- fields: `Sign`, `Fraction`, and `Exponent`
- \((-1)^S \times (1 + 0) \times 2^{0 - 127} = 0\)
- \((-1)^S \times (1 + F) \times 2^{0 - 127} = (-1)^S \times (0 + F) \times 2^{-126} = \text{subnormal}\)
- \((-1)^S \times (1 + F) \times 2^{E - 127} = \text{normal number}\)
- \((-1)^S \times (1 + 0) \times 2^{255 - 127} = \text{infinity}\)
- \((-1)^S \times (1 + F) \times 2^{255 - 127} = \text{NaN}\)
  - Not a Number (details later)
Decimal to Binary Floating Point

Convert -1313.3125 to IEEE 32-bit floating point format:

1. The integer part is \(1313_{10} = 10100100001\) \(_2\). The fractional:

\[
\begin{array}{c|c}
0.3125 & \times 2 = 0.625 \quad 0 \\
0.625 & \times 2 = 1.25 \quad 1 \\
0.25 & \times 2 = 0.5 \quad 0 \\
0.5 & \times 2 = 1.0 \quad 1 \\
\end{array}
\]

2. So \(1313.3125_{10} = 10100100001.0101 \_2\).

3. Normalize: \(10100100001.0101 \_2 = 1.01001000010101 \times 2^{10}\).

4. Mantissa is \(010010000101010000000\)

5. Exponent is \(10 + 127 = 137 = 10001001\) \(_2\), sign bit is 1.

So \(-1313.3125\) is \(1100010010100100001010000000000000 = \text{c4a42a00}_{16}\)
Arithmetic

- addition
  - align radix points
  - use normal addition
Simplified 8-Bit Model

- 3 bits exponent, bias is 3
- 4 bits fraction
Addition – Example

• $3.75 = 11.11_{\text{two}} = 1.111_{\text{two}} \times 2^1 \quad 01001110$

• $10.5 = 1010.1_{\text{two}} = 1.0101_{\text{two}} \times 2^3 \quad 01100101$

• result: $14.25 = 111.001_{\text{two}} \times 2^1$

• normalize: $1.11001_{\text{two}} \times 2^3$

• oops, 5 fraction digits, must round (up or down)

• $\Rightarrow 1.1100_{\text{two}} \times 2^3 = 14 \text{ or } 1.1101_{\text{two}} \times 2^3 = 14.5$