# Representing data and characters

Wettl Ferenc

2017-09-25

1 Binary numbers, 2’s complement
- Binary numbers
- Hexadecimal numbers
- 1’s complement representation
- 2’s complement representation

2 Floating point representation – IEEE 754 standard
- Sign, exponent, fraction

3 ASCII
- ASCII code table
- History: ISO-8859 8-bit standards
- History: latin-2 encoding

4 Unicode and ISO/IEC 10646
- Latin encoding
- UTF – Unicode Transformation Format
- UTF-8

1 Binary numbers, 2's complement
- Binary numbers
- Hexadecimal numbers
- 1's complement representation
- 2's complement representation

2 Floating point representation – IEEE 754 standard
- Sign, exponent, fraction

3 ASCII
- ASCII code table
- History: ISO-8859 8-bit standards
- History: latin-2 encoding

4 Unicode and ISO/IEC 10646
- Latin encoding
- UTF – Unicode Transformation Format
- UTF-8

Conversion from base 2 to base 10:

$$b_n b_{n-1} \dots b_1 b_0 . b_{-1} \dots b_{-m} = \sum_{i=-m}^{n} b_i 2^i .$$

For example $110.101_2 = 6.625$

Conversion from base 10 to base 2

- for integers repeated division by 2,
- for the fractional parts repeated multiplication by 2.

For example 106 in base 2:

$$\begin{aligned}
106 &= 2 \cdot 53 + 0 \to 0 \\
53 &= 2 \cdot 26 + 1 \to 1 \\
26 &= 2 \cdot 13 + 0 \to 0 \\
13 &= 2 \cdot \phantom{0}6 + 1 \to 1 \\
6 &= 2 \cdot \phantom{0}3 + 0 \to 0 \\
3 &= 2 \cdot \phantom{0}1 + 1 \to 1 \\
1 &= 2 \cdot \phantom{0}0 + 1 \to 1
\end{aligned}$$

$$\begin{array}{r|l}
106 & 2 \\ \hline
53 & 0 \\
26 & 1 \\
13 & 0 \\
6 & 1 \\
3 & 0 \\
1 & 1 \\
0 & 1
\end{array}$$

so the binary form is 1101010.

## Example

How to convert a fractional number into binary? For example let us write the first 6 digits of 0.3 in binary!

Solution: The meaning of digits after the decimal point, $1/2$, $1/4$,..., $1/2^n$,.... For example multiplying the binary number 0.1011001 by 2 the integer part of the result in order is 1, 0, 1, 1, 0, 0, 1. Using this method:

$0.3 \cdot 2 = 0.6 \rightarrow 0$
$0.6 \cdot 2 = 1.2 \rightarrow 1$
$0.2 \cdot 2 = 0.4 \rightarrow 0$
$0.4 \cdot 2 = 0.8 \rightarrow 0$
$0.8 \cdot 2 = 1.6 \rightarrow 1$
$0.6 \cdot 2 = 1.2 \rightarrow 1$

So the binary form of 0.3 is 0.010011, we can even see that its infinite binary form is: $0.0\dot{1}00\dot{1}$.

| 0.3 | 2 |
|---|---|
| 0.6 | 0 |
| 1.2 | 1 |
| 0.4 | 0 |
| 0.8 | 0 |
| 1.6 | 1 |
| 1.2 | 1 |

Hexadecimal (base 16) numbers:

| bin | hex | bin | hex |
|---|---|---|---|
| 0000 | 0 | 1000 | 8 |
| 0001 | 1 | 1001 | 9 |
| 0010 | 2 | 1010 | A |
| 0011 | 3 | 1011 | B |
| 0100 | 4 | 1100 | C |
| 0101 | 5 | 1101 | D |
| 0110 | 6 | 1110 | E |
| 0111 | 7 | 1111 | F |

For example 0011 1100 1111 1010 = 0x3CFA.

**1's complement on *n*-bits**: the first bit is the sign. The range of representable numbers: $-2^{n-1}+1$ to $2^{n-1}-1$.
For example on 4 bits: $-7$ to $7$.
$1001 \to -1$
$1100 \to -4$
$1111 \to -7$
$1000 \to -0$
$0000 \to +0$
**Disadvantage:** There's $+0$ and $-0$.

**2's complement representation on *n*-bits**: we want a signed representation of numbers where there aren't $+0$ and $-0$.

$$\bar{x} = \begin{cases} x & \text{if } x \text{ is non-negative,} \\ 2^n - |x| & \text{if } x \text{ is negative.} \end{cases}$$

To calculate $2^n - |x|$ you can take the complement of $|x|$ and add 1: $2^n - |x| = (2^n - 1) - |x| + 1 = 11\ldots1_2 - |x| + 1.$ Since $|x| = 2^n - (2^n - |x|)$, calculating $x$ from $\bar{x}$ can be done the same way, so if the first bit is 1, then $|x| = \text{complement of } \bar{x}\ + 1.$
the form of $-1$ is $11\ldots11_2$, of $-2$ is $11\ldots10_2$, of $-3$ is $11\ldots01_2$.

**Example**

let $n = 4$, $x = -5$: $-5 \to \bar{x} = 16 - 5 = 11 = 1011_2$
with bit operations:
$x = -5 \to |x| = 5 \to 0101_2 \to \bar{x} = 1010_2 + 1_2 = 1011_2$
the reverse: $\bar{x} = 1011_2 \to x = 0100_2 + 1_2 = 0101_2 = 5.$

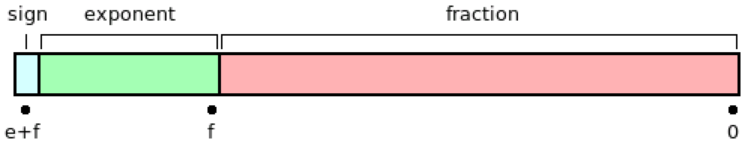
1 Binary numbers, 2's complement
- Binary numbers
- Hexadecimal numbers
- 1's complement representation
- 2's complement representation

2 Floating point representation – IEEE 754 standard
- Sign, exponent, fraction

3 ASCII
- ASCII code table
- History: ISO-8859 8-bit standards
- History: latin-2 encoding

4 Unicode and ISO/IEC 10646
- Latin encoding
- UTF – Unicode Transformation Format
- UTF-8

## IEEE 754-2008, ISO/IEC/IEEE 60559:2011

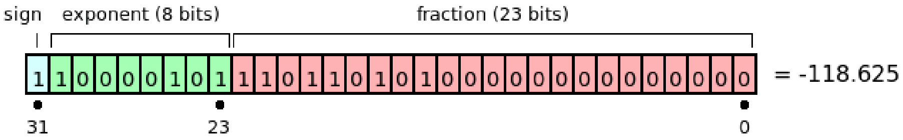


| | $s$=sign | $e$=exponent | fraction | all | bias |
|---|---|---|---|---|---|
| simple | 1 | 8 | 23 | 32 | 127 (01111111) |
| double | 1 | 11 | 52 | 64 | 1023 (01111111111) |

simple: $(-1)^s(1.b_{22}b_{21}\dots b_0)_2 \cdot 2^{e-127} = \left(1+\sum_{i=1}^{23} b_{23-i}2^{-i}\right)\cdot 2^{e-127}$

double: $(-1)^s(1.b_{51}b_{50}\dots b_0)_2 \cdot 2^{e-1023} = \left(1+\sum_{i=1}^{52} b_{52-i}2^{-i}\right)\cdot 2^{e-1023}$

For example using double precision, between $2^{52} = 4\,503\,599\,627\,370\,496$ and $2^{53} = 9\,007\,199\,254\,740\,992$ only integers are represented. between $2^{53}$ and $2^{54}$ only even integers...



sign 1 $\rightarrow$ negative
exponent $10000101_2 - 01111111_2 = 00000110_2$, so 6
fraction (1.significand) $1.110110101_2$,
the number $-1110110.101_2$, which is $-118.625$

## Example

Convert $-14.3$ into IEEE 754 standard on 32 bits.

Solution: $14 = 1110_2$, $0.3 = 0.01001\ldots_2$, so the first 24 digit of the fractional part is 1110.01001100110011001100, the exponent is 3, omitting the first 1 the 23 bits of the *fraction* is 11001001100110011001100. Add 127 to the exponent: $11 + 01111111 = 10000010$, thus the representation of the number after **rounding**: 11000001011001001100110011001101

Rounding: if the first omitted bit is 1, then we add 1 to the last bit of the fraction

There are many sites on the internet that provide converters like this or this.

1 Binary numbers, 2's complement
- Binary numbers
- Hexadecimal numbers
- 1's complement representation
- 2's complement representation

2 Floating point representation – IEEE 754 standard
- Sign, exponent, fraction

3 ASCII
- ASCII code table
- History: ISO-8859 8-bit standards
- History: latin-2 encoding

4 Unicode and ISO/IEC 10646
- Latin encoding
- UTF – Unicode Transformation Format
- UTF-8

## ASCII – American Standard Code for Information Interchange

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 00 | | <control> | 59 | 3B | ; | SEMICOLON |
| ... | | | | 60 | 3C | < | LESS-THAN SIGN |
| 31 | 1F | | <control> | 61 | 3D | = | EQUALS SIGN |
| **32** | **20** | | **SPACE** | 62 | 3E | > | GREATER-THAN SIGN |
| 33 | 21 | ! | EXCLAMATION MARK | 63 | 3F | ? | QUESTION MARK |
| 34 | 22 | " | QUOTATION MARK | 64 | 40 | @ | COMMERCIAL AT |
| 35 | 23 | # | NUMBER SIGN | **65** | **41** | **A** | **LATIN CAPITAL LETTER A** |
| 36 | 24 | $ | DOLLAR SIGN | ... | | | |
| 37 | 25 | % | PERCENT SIGN | 90 | 5A | Z | LATIN CAPITAL LETTER Z |
| 38 | 26 | & | AMPERSAND | 91 | 5B | [ | LEFT SQUARE BRACKET |
| 39 | 27 | ' | APOSTROPHE | 92 | 5C | \ | REVERSE SOLIDUS |
| 40 | 28 | ( | LEFT PARENTHESIS | 93 | 5D | ] | RIGHT SQUARE BRACKET |
| 41 | 29 | ) | RIGHT PARENTHESIS | 94 | 5E | ^ | CIRCUMFLEX ACCENT |
| 42 | 2A | * | ASTERISK | 95 | 5F | _ | LOW LINE |
| 43 | 2B | + | PLUS SIGN | 96 | 60 | ` | GRAVE ACCENT |
| 44 | 2C | , | COMMA | **97** | **61** | **a** | **LATIN SMALL LETTER A** |
| 45 | 2D | - | HYPHEN-MINUS | ... | | | |
| 46 | 2E | . | FULL STOP | 122 | 7A | z | LATIN SMALL LETTER Z |
| 47 | 2F | / | SOLIDUS | 123 | 7B | { | LEFT CURLY BRACKET |
| **48** | **30** | **0** | **DIGIT ZERO** | 124 | 7C | \| | VERTICAL LINE |
| ... | | | | 125 | 7D | } | RIGHT CURLY BRACKET |
| 57 | 39 | 9 | DIGIT NINE | 126 | 7E | ~ | TILDE |
| 58 | 3A | : | COLON | 127 | 7F | | <control> |

## These are nearly history

1. ISO-8859-1 Latin1 (West European)
2. ISO-8859-2 Latin2 (East European)
3. ISO-8859-3 Latin3 (South European)
4. ISO-8859-4 Latin4 (North European)
5. ISO-8859-5 Cyrillic
6. ISO-8859-6 Arabic
7. ISO-8859-7 Greek
8. ISO-8859-8 Hebrew
9. ISO-8859-9 Latin5 (Turkish)
10. ISO-8859-10 Latin6 (Nordic)

## These are nearly history

ISO-8859-2, Microsoft CP1250 (Windows Latin2), CP852 (DOSLatin2)

| | | | | |
|---|---|---|---|---|
| ISO-8859-1 | C1 | Á | U+00C1 | LATIN CAPITAL LETTER A WITH ACUTE |
| ISO-8859-1 | E1 | á | U+00E1 | LATIN SMALL LETTER A WITH ACUTE |
| ISO-8859-1 | D5 | Õ | U+00D5 | LATIN CAPITAL LETTER O WITH TILDE |
| ISO-8859-1 | DB | Û | U+00DB | LATIN CAPITAL LETTER U WITH CIRCUMFLEX |
| ISO-8859-1 | F5 | õ | U+00F5 | LATIN SMALL LETTER O WITH TILDE |
| ISO-8859-1 | FB | û | U+00FB | LATIN SMALL LETTER U WITH CIRCUMFLEX |
| ISO-8859-2 | D5 | Ő | U+0150 | LATIN CAPITAL LETTER O WITH DOUBLE ACUTE |
| ISO-8859-2 | DB | Ű | U+0170 | LATIN CAPITAL LETTER U WITH DOUBLE ACUTE |
| ISO-8859-2 | F5 | ő | U+0151 | LATIN SMALL LETTER O WITH DOUBLE ACUTE |
| ISO-8859-2 | FB | ű | U+0171 | LATIN SMALL LETTER U WITH DOUBLE ACUTE |
| CP1250 | 82 | ‚ | U+201A | SINGLE LOW-9 QUOTATION MARK |
| CP1250 | 84 | „ | U+201E | DOUBLE LOW-9 QUOTATION MARK |
| CP1250 | 85 | … | U+2026 | HORIZONTAL ELLIPSIS |
| CP1250 | 91 | ‘ | U+2018 | LEFT SINGLE QUOTATION MARK |
| CP1250 | 92 | ’ | U+2019 | RIGHT SINGLE QUOTATION MARK |
| CP1250 | 93 | “ | U+201C | LEFT DOUBLE QUOTATION MARK |
| CP1250 | 94 | ” | U+201D | RIGHT DOUBLE QUOTATION MARK |
| CP1250 | 96 | – | U+2013 | EN DASH |
| CP1250 | 97 | — | U+2014 | EM DASH |

1 Binary numbers, 2's complement
- Binary numbers
- Hexadecimal numbers
- 1's complement representation
- 2's complement representation

2 Floating point representation – IEEE 754 standard
- Sign, exponent, fraction

3 ASCII
- ASCII code table
- History: ISO-8859 8-bit standards
- History: latin-2 encoding

4 Unicode and ISO/IEC 10646
- Latin encoding
- UTF – Unicode Transformation Format
- UTF-8

- U+0000 - U+007F ASCII
- U+0080 - U+00FF Latin-1
- U+0100 - U+017F Latin Extended-A (latin1, hungarian ő, ű)
- U+0180 - U+024F Latin Extended-B
- U+1E00 - U+1EFF Latin Extended Additional

- UTF-8 every character is represented on 8, 16, 24 or 32-bits.
- UTF-16 every character is represented on 16 or 32-bits.
- UTF-32 every character is represented on 32-bits.

| Unicode | | UTF-8 | a official name of the character |
|---|---|---|---|
| U+0020 | | 20 | SPACE |
| U+0030 | 0 | 30 | DIGIT ZERO |
| U+0040 | @ | 40 | COMMERCIAL AT |
| U+0041 | A | 41 | LATIN CAPITAL LETTER A |
| U+0061 | a | 61 | LATIN SMALL LETTER A |
| U+00C1 | Á | c3 81 | LATIN CAPITAL LETTER A WITH ACUTE |
| U+00C9 | É | c3 89 | LATIN CAPITAL LETTER E WITH ACUTE |
| U+00CD | Í | c3 8d | LATIN CAPITAL LETTER I WITH ACUTE |
| U+00D3 | Ó | c3 93 | LATIN CAPITAL LETTER O WITH ACUTE |
| U+00D6 | Ö | c3 96 | LATIN CAPITAL LETTER O WITH DIAERESIS |
| U+00DA | Ú | c3 9a | LATIN CAPITAL LETTER U WITH ACUTE |
| U+00DC | Ü | c3 9c | LATIN CAPITAL LETTER U WITH DIAERESIS |
| U+00E1 | á | c3 a1 | LATIN SMALL LETTER A WITH ACUTE |
| U+00E9 | é | c3 a9 | LATIN SMALL LETTER E WITH ACUTE |
| U+00ED | í | c3 ad | LATIN SMALL LETTER I WITH ACUTE |
| U+00F3 | ó | c3 b3 | LATIN SMALL LETTER O WITH ACUTE |
| U+00F6 | ö | c3 b6 | LATIN SMALL LETTER O WITH DIAERESIS |
| U+00FA | ú | c3 ba | LATIN SMALL LETTER U WITH ACUTE |
| U+00FC | ü | c3 bc | LATIN SMALL LETTER U WITH DIAERESIS |
| U+0150 | Ő | c5 90 | LATIN CAPITAL LETTER O WITH DOUBLE ACUTE |
| U+0151 | ő | c5 91 | LATIN SMALL LETTER O WITH DOUBLE ACUTE |
| U+0170 | Ű | c5 b0 | LATIN CAPITAL LETTER U WITH DOUBLE ACUTE |
| U+0171 | ű | c5 b1 | LATIN SMALL LETTER U WITH DOUBLE ACUTE |

| Range (number) | binary form | UTF-8 |
|---|---|---|
| 000000-00007F (128) | 0zzzzzzz | 0zzzzzzz |
| 000080-0007FF (1920) | 00000yyy yyzzzzzz | 110yyyyy 10zzzzzz |
| 000800-00FFFF (63488) | xxxxyyyy yyzzzzzz | 1110xxxx 10yyyyyy 10zzzzzz |
| 010000-10FFFF (1048576) | 000wwwxx xxxxyyyy yyzzzzzz | 11110www 10xxxxxx 10yyyyyy 10zzzzzz |

Á 00C1→1100 0001→00011 000001→**110**00011 **10**000001→C3 81
Õ 00D5→1101 0101→00011 010101→**110**00011 **10**010101→C3 95
Ő 0150→0001 0101 0000→00101 010000→**110**00101 **10**010000→C5 90
Byte Order Mark FEFF→11111110 11111111→
**1110**1111 **10**111011 **10**111111→EF BB BF (ï»¿ When viewing files written in UTF-8 formats on windows and reading with a latin-1 encoder)

## Test questions

1. Calculate the binary form of 13.4 and −12.6!
2. Calculate the 2's complement form of −23 and −24 on 8 bits.
3. What is the value of the number represented on 2's complement by 10101001?
4. What is the IEEE 754 standard form of −23.4 and −12.6 on 32-bits?
5. What is the value of this 32-bits floating point number 11000001110101100000000000000000?
6. How to convert a character from unicode to utf-8?
7. The unicode of the character € is U+20AC. Calculate its UTF-8 code in binary and hexadecimal!