Informatics 1 3rd lecture: Representing numbers and characters

Using Ferenc Wettl's presentation

Budapest University of Technology and Economics

2019-09-30

Base 2 format

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:

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

so the binary form is 1101010.

106	2
53	0
26	1
13	0
6	1
3	0
1	1
0	1

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, $1/2$, $1/4, \ldots, 1/2^n, \ldots$. 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 numbers

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 representation

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.

$1001 \rightarrow -1$

$1100 \rightarrow -4$

$1111 \rightarrow -7$

$1000 \rightarrow -0$

$0000 \rightarrow +0$

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.

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

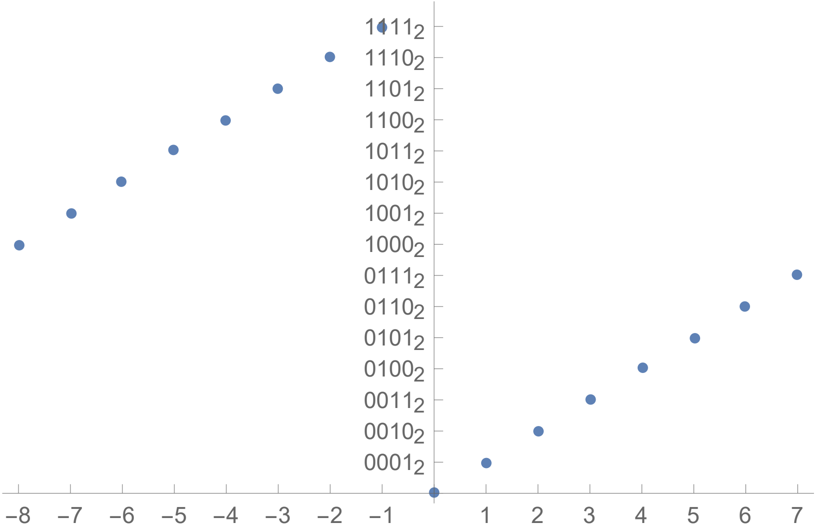
To calculate $2^n - |x|$ you can take the complement of $|x|$ and add 1: $2^n - |x| = (2^n - 1) - |x| + 1 = 11\dots1_2 - |x| + 1$. Since $|x| = 2^n - (2^n - |x|)$, calculating x from $\bar{x}$ can be done the same way, so if the first bit is 1, then $|x| =$ complement of $\bar{x}\ + 1$.
the form of -1 is $11\dots11_2$, of -2 is $11\dots10_2$, of -3 is $11\dots01_2$.

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: $\bar{x} = 1011_2 \to x = 0100_2 + 1_2 = 0101_2 = 5$.

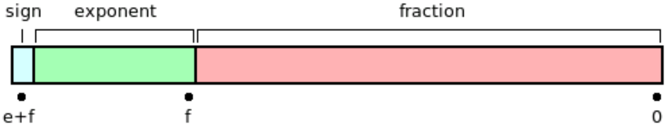
2's complement representation

On 4 bits:



Floating point representation

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:

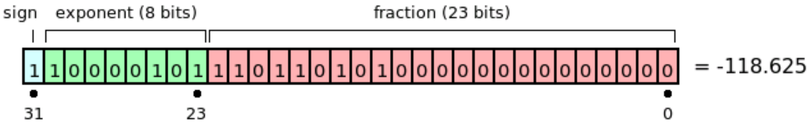
$$(-1)^s(1.b_{22}b_{21}\dots b_0)_2\cdot 2^{e-127}=\left(1+\sum_{i=1}^{23}b_{23-i}2^{-i}\right)\cdot 2^{e-127}$$

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

Example 1



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 2

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.

ASCII – American Standard Code for Information Interchange

0	00		<control>
...			
31	1F		<control>
32	20		SPACE
33	21	!	EXCLAMATION MARK
34	22	"	QUOTATION MARK
35	23	#	NUMBER SIGN
36	24	$	DOLLAR SIGN
37	25	%	PERCENT SIGN
38	26	&	AMPERSAND
39	27	'	APOSTROPHE
40	28	(	LEFT PARENTHESIS
41	29	)	RIGHT PARENTHESIS
42	2A	*	ASTERISK
43	2B	+	PLUS SIGN
44	2C	,	COMMA
45	2D	-	HYPHEN-MINUS
46	2E	.	FULL STOP
47	2F	/	SOLIDUS
48	30	0	DIGIT ZERO
...			
57	39	9	DIGIT NINE
58	3A	:	COLON

<table>
<tr><td>59</td><td>3B</td><td>;</td><td>SEMICOLON</td></tr>
<tr><td>60</td><td>3C</td><td><</td><td>LESS-THAN SIGN</td></tr>
<tr><td>61</td><td>3D</td><td>=</td><td>EQUALS SIGN</td></tr>
<tr><td>62</td><td>3E</td><td>></td><td>GREATER-THAN SIGN</td></tr>
<tr><td>63</td><td>3F</td><td>?</td><td>QUESTION MARK</td></tr>
<tr><td>64</td><td>40</td><td>@</td><td>COMMERCIAL AT</td></tr>
<tr><td>65</td><td>41</td><td>A</td><td>LATIN CAPITAL LETTER A</td></tr>
<tr><td>...</td><td></td><td></td><td></td></tr>
<tr><td>90</td><td>5A</td><td>Z</td><td>LATIN CAPITAL LETTER Z</td></tr>
<tr><td>91</td><td>5B</td><td>[</td><td>LEFT SQUARE BRACKET</td></tr>
<tr><td>92</td><td>5C</td><td>\</td><td>REVERSE SOLIDUS</td></tr>
<tr><td>93</td><td>5D</td><td>]</td><td>RIGHT SQUARE BRACKET</td></tr>
<tr><td>94</td><td>5E</td><td>^</td><td>CIRCUMFLEX ACCENT</td></tr>
<tr><td>95</td><td>5F</td><td>_</td><td>LOW LINE</td></tr>
<tr><td>96</td><td>60</td><td>`</td><td>GRAVE ACCENT</td></tr>
<tr><td>97</td><td>61</td><td>a</td><td>LATIN SMALL LETTER A</td></tr>
<tr><td>...</td><td></td><td></td><td></td></tr>
<tr><td>122</td><td>7A</td><td>z</td><td>LATIN SMALL LETTER Z</td></tr>
<tr><td>123</td><td>7B</td><td>{</td><td>LEFT CURLY BRACKET</td></tr>
<tr><td>124</td><td>7C</td><td>|</td><td>VERTICAL LINE</td></tr>
<tr><td>125</td><td>7D</td><td>}</td><td>RIGHT CURLY BRACKET</td></tr>
<tr><td>126</td><td>7E</td><td>~</td><td>TILDE</td></tr>
<tr><td>127</td><td>7F</td><td></td><td><control></td></tr>
</table>

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 ACU
ISO-8859-2	DB	Ű	U+0170	LATIN CAPITAL LETTER U WITH DOUBLE ACU
ISO-8859-2	F5	ő	U+0151	LATIN SMALL LETTER O WITH DOUBLE ACUT
ISO-8859-2	FB	ű	U+0171	LATIN SMALL LETTER U WITH DOUBLE ACUT
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

Unicode

- 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 – Unicode Transformation Format

- 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.

UTF-8

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

UTF-8

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 10zzz

Á 00C1→1100 0001→00011 000001→11000011 10000001→C3 81
Õ 00D5→1101 0101→00011 010101→11000011 10010101→C3 95
Ő 0150→0001 0101 0000→00101 010000→11000101 10010000→C5 90
Byte Order Mark FEFF→11111110 11111111→ 11101111 10111011 10111111→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!