Binary Representation of Numbers

Powers

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{24}	2^{32}
16^0				16^1				16^2				16^3				16^4	16^6	16^8
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16 384	32 768	65 536	16 777 216	4 294 967 296

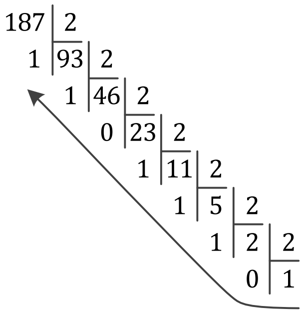
Unsigned integers (positive only)

Principle: Each digit (d_i) is multiplied by a non-negative power of the base (b), i.e. $n = d_{N-1}b^{N-1} + \cdots + d_1 b^1 + d_0 b^0$

Base 10 (decimal):	digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9	$1729_{10} = 1729 = 1 \times 10^3 + 7 \times 10^2 + 2 \times 10^1 + 9 \times 10^0$
Base 2 (binary):	digits are 0, 1 (**bi**nary digi**t** = bit)	$1101_2 = 0b1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$
Base 16 (hexadecimal):	digits are 0, 1, 2, ..., 9, A, B, C, D, E, F	$5A4_{16} = 0x5A4 = 5 \times 16^2 + 10 \times 16^1 + 4 \times 16^0 = 1444$

Conversion 10 → 2

Method 1: Successive division by 2, then reading of the last quotient and of the remainders from the last one.



$\Rightarrow 187 = 1011\ 1011_2$

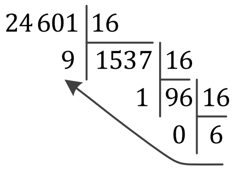
Method 2: Successive decomposition as a sum of powers of two, then have 1 for each power present, 0 otherwise (requires knowing the powers of two).

$$\begin{aligned} 187 &= 128 + 59 \\ &= 128 + 32 + 27 \\ &= 128 + 32 + 16 + 11 \\ &= 128 + 32 + 16 + 8 + 3 \\ &= 128 + 32 + 16 + 8 + 2 + 1 \\ &= 2^7 + 2^5 + 2^4 + 2^3 + 2^1 + 2^0 \end{aligned}$$

$\Rightarrow 187 = 1011\ 1011_2$

Conversion 10 → 16

Method 1: Successive division by 16, then reading of the last quotient and of the remainders from the last one.



$\Rightarrow 24\,601 = 6019_{16}$

Method 2: Successive decomposition as a sum of weighted powers of 16, then have the weight associated to each power (requires knowing the powers of 16).

$$\begin{aligned} 24\,601 &= 6 \times 4096 + 25 \\ &= 6 \times 4096 + 16 + 9 \\ &= 6 \times 16^3 + 16 \times 16^1 + 9 \times 16^0 \end{aligned}$$

$\Rightarrow 24\,601 = 6019_{16}$

Conversion 2 ↔ 16

Method: From the right digit, each hexadecimal digit is converted to four bits, or vice versa (requires knowing the numbers up to 15 in binary).

$C137_{16} = 1100\ 0001\ 0011\ 0111_2$

$1100\ 1010\ 1111\ 1110_2 = CAFE_{16}$

Information & Tips

- To make binary numbers easier to read, bits can be grouped by 4 (like decimal digits can be grouped by 3).
- The leftmost bit is called MSB, and the rightmost bit is called LSB (for most/least significant bit).
- The LSB indicates the parity of the number (0 = even, 1 = odd).
- In hardware implementations, the number of bits is most of the time a power of two: 8, 16, 32, 64, 128.
- Zeros can be padded on the left side if a specific length is wanted.
- To encode an unsigned number n, $\lfloor \log_2 n \rfloor + 1$ bits are needed.

Range

Number of bits	N	8	16	24	32	64	128	256
Minimum value	0	0	0	0	0	0	0	0
Maximum value	$2^N - 1$	255	65 535	16 777 215	$\approx 4.29 \times 10^9$	$\approx 1.84 \times 10^{19}$	$\approx 3.40 \times 10^{38}$	$\approx 1.16 \times 10^{77}$

Signed integers (two's complement)

Conversion 10 ↔ 2 for positive integers: like unsigned and left pad at least one 0

Conversion 10 → 2 for negative integers

Example: $n = -9430$ on 16 bits

- Convert $|n|$ in binary — $10\ 0100\ 1101\ 0110_2$
- Pad at least one 0 — $0010\ 0100\ 1101\ 0110_2$
- Reverse all bits ($= 2^N - 1 - |n|$) — $1101\ 1011\ 0010\ 1001_2$
- Add 1 ($= 2^N - |n|$) — $1101\ 1011\ 0010\ 1010_2$

Conversion 2 → 10 for negative integers

Example: $1101\ 1011\ 0010\ 1010_2$

- Reverse all bits ($= |n| - 1$) — $0010\ 0100\ 1101\ 0101_2$
- Add 1 ($= |n|$) — $0010\ 0100\ 1101\ 0110_2$
- Convert to decimal — $2 + 4 + \cdots + 8192 = 9430$
- Put minus sign ($= -|n|$) — $n = -9430$

Direct method:

- Convert each bit like unsigned, except the MSB where a minus sign is applied — $2 + 8 + \cdots + 16\,384 - 32\,768 = -9430$

Information & Tips

- We must know if we are dealing with unsigned or signed integers, it cannot be guessed from the bits.
- Faster method than reverse and add 1: From the LSB, keep all the bits up to the first 1, then reverse.
- The MSB shows the sign (0: +, 1: −).
- Bits with the same value as the MSB can be padded on the left side if a specific length is wanted.
- To encode a signed number n, $n > 0$: $\lfloor \log_2 n \rfloor + 2$ bits are needed $n < 0$: $\lceil \log_2 |n| \rceil + 1$ bits are needed.

Range

Number of bits	N	8	16	24	32	64	128	256
Minimum value	-2^{N-1}	−128	−32 768	−8 388 608	$\approx -2.15 \times 10^9$	$\approx -9.22 \times 10^{18}$	$\approx -1.70 \times 10^{38}$	$\approx -5.79 \times 10^{76}$
Maximum value	$2^{N-1} - 1$	127	32 767	8 388 607	$\approx 2.15 \times 10^9$	$\approx 9.22 \times 10^{18}$	$\approx 1.70 \times 10^{38}$	$\approx 5.79 \times 10^{76}$

Tricks

Fast product ($n \times 2^K$): Left shift the bits K times, and right pad 0s K times. $13 \times 16 = 1101_2 \ll 4 = 11010000_2 = 208$

Fast division ($\lfloor n/2^K \rfloor$): Right shift the bits K times, and left pad 0s if unsigned or MSBs if signed K times ($\lfloor\ \rfloor$ rounds towards $-\infty$). $\lfloor 43/8 \rfloor = 101011_2 \gg 3 = 000101_2 = 5$ $\lfloor -27/4 \rfloor = 100101_2 \gg 2 = 111001_2 = -7$

Negative Powers

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
			16^{-1}				16^{-2}
0.5	0.25	0.125	0.0625	0.031 25	0.015 625	0.007 812 5	0.003 906 25

Fixed-point numbers

Principle: Each digit is multiplied by an integer power of the base, $n = d_{N_I-1}b^{N_I-1} + \cdots + d_0 b^0 + d_{-1}b^{-1} + \cdots + d_{-N_F}b^{-N_F}$

Base 10 (decimal): $23.10344 = 2 \times 10^1 + 3 \times 10^0 + 1 \times 10^{-1} + 0 \times 10^{-2} + 3 \times 10^{-3} + 4 \times 10^{-4} + 4 \times 10^{-5}$

Base 2 (binary): $10.10111_2 = 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} = 2.718\,75$

Base 16 (hexadecimal): $3.244_{16} = 3 \times 16^0 + 2 \times 16^{-1} + 4 \times 16^{-2} + 4 \times 16^{-3} = 3.141\,601\,562\,5$

Conversion 10 → 2

Method for the integer part: like integers.

Method for the fractional part:
Successive multiplication by 2 of the fractional part (if it reaches 0 ⇒ exact conversion, else ∞ number of bits), then reading of the units from the first one.

Example 1: $13.1875 = 1101.0011_2$

$13 = 8 + 4 + 1 \Rightarrow 13 = 1101_2$

0.1875 × 2 = 0.375	0
0.375 × 2 = 0.75	0
0.75 × 2 = 1.5	1
0.5 × 2 = 1	1

$\Rightarrow 0.1875 = 0.0011_2$

Example 2: $7.4 = 111.0110\,0110..._2$

$7 = 4 + 2 + 1 \Rightarrow 7 = 111_2$

0.4 × 2 = 0.8	0
0.8 × 2 = 1.6	1
0.6 × 2 = 1.2	1
0.2 × 2 = 0.4	0

$\Rightarrow 0.4 = 0.0110\,0110..._2 = 0.\overline{0110}_2$

Fixed-point numbers are fractions whose denominator is a power of the base (equivalent to manipulate integers and apply a factor). In base 2:

$$\sum_{i=-N_F}^{N_I-1} d_i 2^i = \frac{d_{N_I-1}2^{N_I+N_F-1} + \cdots + d_0 2^{N_F} + \cdots + d_{-(N_F-1)}2 + d_{-N_F}}{2^{N_F}}$$

Examples: $2.718\,75 = \frac{87}{32}$; $3.141\,601\,562\,5 = \frac{3217}{1024}$; $13.1875 = \frac{211}{16}$

Whereas: $7.4 = \frac{37}{5} \neq \frac{q}{2^p}, \forall(q,p) \in \mathbb{N}^2$; $7.45 = \frac{149}{20} \neq \frac{q}{2^p}, \forall(q,p) \in \mathbb{N}^2$

Tips:

- Every non-integer fixed-point binary number has 5 as last digit in decimal (but not all numbers with 5 as last digit in decimal can be encoded in a fixed-point binary number).
- With N_F bits, there are at least $D = \lfloor N_F / \log_2(10) \rfloor \approx \lfloor 0.3\, N_F \rfloor$ correct decimals.
- To have at least D correct decimals, $N_F = \lceil D \log_2(10) \rceil \approx \lceil D / 0.3 \rceil$ bits are required.

D correct decimals	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
N_F bits required	4	7	10	14	17	20	24	27	30	34	37	40	44	47	50	54	57	60	64	67

Floating-point numbers

Principle: Express a number as the product of a fixed-point number (significand) and a power of the base (exponent).
$n = \pm m \times b^e$. To have a unique representation, the integer part of the significand is limited to: $1 \leq \lfloor m \rfloor < b$.
This representation allows the encoding and manipulation of very large and very small numbers.
Like for fixed-point numbers, only fractions whose denominator is a power of the base can be represented exactly.

Binary format ($b = 2$) according to IEEE 754 standard:

- Sign encoded with one bit.
- Exponent encoded as an unsigned to which an offset is applied.
- Significand encoded without the integer part, since it is always 1.

$n = -1^s\, 2^{e-e_0}\,(1 + m)$, with $0 \leq m < 1$
e_0: exponent offset

s = sign	e = exponent	m = significand
1 bit	E bits	M bits

Exceptions:

- e full of 1 & $m = 0$: ±Inf (infinity)
- e full of 1 & $m \neq 0$: NaN (not a number)
- $e = 0$: subnormal number (integer part of significand is 0)
- $e = 0$ and $m = 0$: zero

Expression for subnormal numbers:
$n = -1^s\, 2^{e-e_0+1}\,(0 + m)$, with $0 \leq m < 1$

- Allows encoding zero.
- Allows encoding smaller numbers.
- Have less accuracy than normal numbers.

Format in the IEEE 754 standard	*Storage (bit)* 1 | E | M	*Exponent offset* e_0	*Maximum* $2^{2^E-2-e_0} \times (2-2^{-M})$	*Minimum normal* 2^{1-e_0}	*Minimum subnormal* 2^{1-e_0-M}	*All integers encoded exactly up to* $\pm 2^{M+1}$
16 bits, binary16 half-precision	1 | 5 | 10	15	$2^{15}\times(2-2^{-10})$ = 65 504	2^{-14} ≈ 6.10×10^{-5}	2^{-24} ≈ 5.96×10^{-8}	$\pm 2^{11}$ = ±2048
32 bits, binary32 float/single precision	1 | 8 | 23	127	$2^{127}\times(2-2^{-23})$ ≈ 3.40×10^{38}	2^{-126} ≈ 1.18×10^{-38}	2^{-149} ≈ 1.40×10^{-45}	$\pm 2^{24}$ = ±16 777 216
64 bits, binary64 double/double precision	1 | 11 | 52	1023	$2^{1023}\times(2-2^{-52})$ ≈ 1.80×10^{308}	2^{-1022} ≈ 2.23×10^{-308}	2^{-1074} ≈ 4.94×10^{-324}	$\pm 2^{53}$ ≈ $\pm 9.01\times10^{15}$
80 bits (no implicit bit) extended double	1 | 15 | 1+63	16 383	$2^{16383}(2-2^{-63})$ ≈ 1.19×10^{4932}	2^{-16382} ≈ 3.36×10^{-4932}	2^{-16445} ≈ 3.65×10^{-4951}	$\pm 2^{64}$ ≈ $\pm 1.84\times10^{19}$
128 bits, binary128 quadruple precision	1 | 15 | 112	16 383	$2^{16383}(2-2^{-112})$ ≈ 1.19×10^{4932}	2^{-16382} ≈ 3.36×10^{-4932}	2^{-16494} ≈ 6.48×10^{-4966}	$\pm 2^{113}$ ≈ $\pm 1.04\times10^{34}$

Conversion 10 → 2 for normal numbers

- Exponent computation: $2^{e-e_0} \leq |n| < 2^{e-e_0+1} \Rightarrow e - e_0 = \lfloor \log_2 |n| \rfloor$
- Significand computation: $2^{e-e_0}(1+m) = |n| \Rightarrow m = |n|/2^{e-e_0} - 1$
- Convert the exponent and the significand in binary
- Put things together according to the format

Example: 8000.5 on 32 bits = 0x45FA0400

- $e - e_0 = \lfloor \log_2(8000.5) \rfloor = 12 \Longrightarrow e = 139$
- $m = 8000.5/2^{12} - 1 = 0.953\,247\,070\,3125$
- $e = 10001011_2$; $m = 0.1111\,0100\,0000\,1_2$
- 0 | 10001011 | 11110100000010000000000