Binary Number Representations

Table of contents

Bits, bytes, and words

- **Bit** = binary digit (0 or 1).
- **Byte** = 8 bits, e.g. `01010111`.
- Modern computers deal with **words**, usually a power-of-2 number of bytes: 1, 2, 4, or 8 bytes = 8, 16, 32, 64 bits.

Representing whole (unsigned) numbers

Each bit position has a value — a power of 2, increasing from right (least significant) to left (most significant):

Bit position	9	8	7	6	5	4	3	2	1	0
Value	512	256	128	64	32	16	8	4	2	1

Binary → decimal: add the values of each position where the bit is 1. E.g. `10010001` = $128 + 16 + 1 = 145$.

- **Least significant bit (LSB)** — the bit position worth the least ($2^0 = 1$).
- **Most significant bit (MSB)** — the bit position worth the most. For an n-bit unsigned word, the MSB is worth 2^{n-1}.

Converting decimal to binary

Two equivalent methods (example: convert 53 to binary):

- **Method 1**: rewrite n as a sum of powers of 2, by repeatedly subtracting the largest power of 2 not greater than n. Assemble the binary number from 1's in the bit positions corresponding to those powers of 2, 0's elsewhere.
- **Method 2** (build up from the right/LSB): divide n by 2; the remainder (0 or 1) is the next bit; repeat with $n =$ the quotient, until $n = 0$.

Number range (unsigned)

- Smallest representable value: all 0's $\rightarrow$ 0.
- Largest representable value: all 1's $\rightarrow$ for an n-bit word, $2^n - 1$ (e.g. 255 for 8 bits).

Other radices

Radix = number system base. A radix-k number system has k distinct symbols for digits 0 to $k-1$, and the value of each digit (from the right) is $k^0, k^1, k^2, ...$

- **Octal** (radix-8): symbols `0`–`7`. One octal digit corresponds to exactly 3 bits.
- **Hexadecimal** (radix-16): symbols `0`–`9`, `A`–`F`. One hex digit corresponds to exactly 4 bits — very convenient for grouping binary.

Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Oct	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	20	21
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11

Radix notation conventions

Since a bare number like `101` or `747` is ambiguous, a radix indicator is needed. A subscript works generically (e.g. 101_2, 101_{16}); in code/assembly, conventions vary:

Radix	Convention	Example	Where used
Hex	leading `0x`	`0x101`	C, Atmel AVR
Hex	trailing `h`	`101h`	some assembly languages
Hex	leading `$`	`$747`	Atmel AVR assembly
Octal	leading `0`	`0101`	C, Atmel AVR
Octal	trailing `q`	`101q`	some assembly languages

Radix	Convention	Example	Where used
Octal	leading `@`	`@747`	some assembly languages
Binary	leading `0b`	`0b101`	Atmel AVR assembly, some C
Binary	trailing `b`	`101b`	some assembly languages
Binary	leading `%`	`%101`	some assembly languages