# Hexadecimal

In mathematics and computing, the **hexadecimal** (also **base-16** or simply **hex**) numeral system is a positional numeral system that represents numbers using a radix (base) of 16. Unlike the decimal system representing numbers using 10 symbols, hexadecimal uses 16 distinct symbols, most often the symbols "0"–"9" to represent values 0 to 9, and "A"–"F" (or alternatively "a"–"f") to represent values from 10 to 15.

Software developers and system designers widely use hexadecimal numbers because they provide a human-friendly representation of binary-coded values. Each hexadecimal digit represents four bits (binary digits), also known as a nibble (or nybble).[1] For example, an 8-bit byte can have values ranging from 00000000 to 11111111 in binary form, which can be conveniently represented as 00 to FF in hexadecimal.

In mathematics, a subscript is typically used to specify the base. For example, the decimal value 65,226 would be expressed in hexadecimal as $FECA_{16}$. In programming, a number of notations are used to denote hexadecimal numbers, usually involving a prefix. The prefix `0x` is used in C, which would denote this value as `0xFECA`.

Hexadecimal is used in the transfer encoding **Base16**, in which each byte of the plaintext is broken into two 4-bit values and represented by two hexadecimal digits.

## Contents

# Representation

## Written representation

In most current use cases, the letters A–F or a–f represent the values 10–15, while the numerals 0–9 are used to represent their decimal values.

There is no universal convention to use lowercase or uppercase, so each is prevalent or preferred in particular environments by community standards or convention; even mixed case is used. Seven-segment displays use mixed-case AbCdEF to make digits that can be distinguished from each other.

There is some standardization of using spaces (rather than commas or another punctuation mark) to separate hex values in a long list. For instance, in the following hex dump, each 8-bit byte is a 2-digit hex number, with spaces between them, while the 32-bit offset at the start is an 8-digit hex number.

```
00000000  57 69 6b 69 70 65 64 69  61 2c 20 74 68 65 20 66
00000010  72 65 65 20 65 6e 63 79  63 6c 6f 70 65 64 69 61
00000020  20 74 68 61 74 20 61 6e  79 6f 6e 65 20 63 61 6e
00000030  20 65 64 69 74 0a
```

### Distinguishing from decimal

In contexts where the base is not clear, hexadecimal numbers can be ambiguous and confused with numbers expressed in other bases. There are several conventions for expressing values unambiguously. A numerical subscript (itself written in decimal) can give the base explicitly: $159_{10}$ is decimal 159; $159_{16}$ is hexadecimal 159, which equals $345_{10}$. Some authors prefer a text subscript, such as $159_{\text{decimal}}$ and $159_{\text{hex}}$, or $159_{\text{d}}$ and $159_{\text{h}}$.

Donald Knuth introduced the use of a particular typeface to represent a particular radix in his book *The TeXbook*.[2] Hexadecimal representations are written there in a typewriter typeface: `5A3`

In linear text systems, such as those used in most computer programming environments, a variety of methods have arisen:

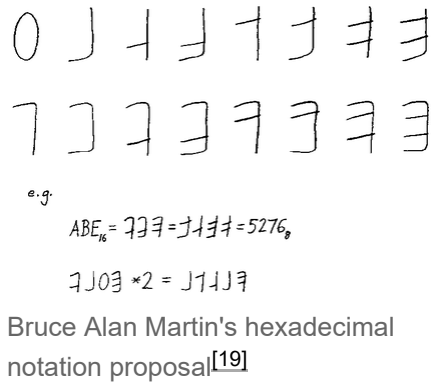
- Unix (and related) shells, AT&T assembly language and likewise the C programming language (and its syntactic descendants such as C++, C#, Go, D, Java, JavaScript, Python and Windows PowerShell) use the prefix `0x` for numeric constants represented in hex: `0x5A3`. Character and string constants may express character codes in hexadecimal with the prefix `\x` followed by two hex digits: `'\x1B'` represents the Esc control character; `"\x1B[0m\x1B[25;1H"` is a string containing 11 characters with two embedded Esc characters.[3] To output an integer as hexadecimal with the printf function family, the format conversion code `%X` or `%x` is used.
- In URIs (including URLs), character codes are written as hexadecimal pairs prefixed with `%`: `http://www.example.com/name%20with%20spaces` where `%20` is the code for the space (blank) character, ASCII code point 20 in hex, 32 in decimal.

- In XML and XHTML, characters can be expressed as hexadecimal numeric character references using the notation `&#x`*code*`;`, for instance `&#x2019;` represents the character U+2019 (the right single quotation mark). If there is no `x` the number is decimal (thus `&#8217;` is the same character).[4]
- In the Unicode standard, a character value is represented with `U+` followed by the hex value, e.g. `U+20AC` is the Euro sign (€).
- Color references in HTML, CSS and X Window can be expressed with six hexadecimal digits (two each for the red, green and blue components, in that order) prefixed with `#`: white, for example, is represented as `#FFFFFF`.[5] CSS also allows 3-hexdigit abbreviations with one hexdigit per component: #FA3 abbreviates #FFAA33 (a golden orange: ).
- In MIME (e-mail extensions) quoted-printable encoding, character codes are written as hexadecimal pairs prefixed with `=`: `Espa=F1a` is "España" (F1 is the code for ñ in the ISO/IEC 8859-1 character set).[6])
- In Intel-derived assembly languages and Modula-2,[7] hexadecimal is denoted with a suffixed `H` or `h`: `FFh` or `05A3H`. Some implementations require a leading zero when the first hexadecimal digit character is not a decimal digit, so one would write `0FFh` instead of `FFh`. Some other implementations (such as NASM) allow C-style numbers (`0x42`).
- Other assembly languages (6502, Motorola), Pascal, Delphi, some versions of BASIC (Commodore), GameMaker Language, Godot and Forth use `$` as a prefix: `$5A3`.
- Some assembly languages (Microchip) use the notation `H'ABCD'` (for $ABCD_{16}$). Similarly, Fortran 95 uses Z'ABCD'.
- Ada and VHDL enclose hexadecimal numerals in based "numeric quotes": `16#5A3#`. For bit vector constants VHDL uses the notation `x"5A3"`.[8]
- Verilog represents hexadecimal constants in the form `8'hFF`, where 8 is the number of bits in the value and FF is the hexadecimal constant.
- The Smalltalk language uses the prefix `16r`: `16r5A3`
- PostScript and the Bourne shell and its derivatives denote hex with prefix `16#`: `16#5A3`. For PostScript, binary data (such as image pixels) can be expressed as unprefixed consecutive hexadecimal pairs: `AA213FD51B3801043FBC`...
- Common Lisp uses the prefixes `#x` and `#16r`. Setting the variables *read-base*[9] and *print-base*[10] to 16 can also be used to switch the reader and printer of a Common Lisp system to Hexadecimal number representation for reading and printing numbers. Thus Hexadecimal numbers can be represented without the #x or #16r prefix code, when the input or output base has been changed to 16.
- MSX BASIC,[11] QuickBASIC, FreeBASIC and Visual Basic prefix hexadecimal numbers with `&H`: `&H5A3`
- BBC BASIC and Locomotive BASIC use `&` for hex.[12]
- TI-89 and 92 series uses a `0h` prefix: `0h5A3`
- ALGOL 68 uses the prefix `16r` to denote hexadecimal numbers: `16r5a3`. Binary, quaternary (base-4) and octal numbers can be specified similarly.
- The most common format for hexadecimal on IBM mainframes (zSeries) and midrange computers (IBM i) running the traditional OS's (zOS, zVSE, zVM, TPF, IBM i) is `X'5A3'`, and is used in Assembler, PL/I, COBOL, JCL, scripts, commands and other places. This format was common on other (and now obsolete) IBM systems as well. Occasionally quotation marks were used instead of apostrophes.
- Any IPv6 address can be written as eight groups of four hexadecimal digits (sometimes called hextets), where each group is separated by a colon (`:`). This, for example, is a valid IPv6 address: `2001:0db8:85a3:0000:0000:8a2e:0370:7334` or abbreviated by removing zeros as `2001:db8:85a3::8a2e:370:7334` (IPv4 addresses are usually written in decimal).
- Globally unique identifiers are written as thirty-two hexadecimal digits, often in unequal hyphen-separated groupings, for example `3F2504E0-4F89-41D3-9A0C-0305E82C3301`.

## Other symbols for 10–15 and mostly different symbol sets

The use of the letters *A* through *F* to represent the digits above 9 was not universal in the early history of computers.

- During the 1950s, some installations, such as Bendix-14, favored using the digits 0 through 5 with an overline to denote the values 10–15 as $\overline{0}$, $\overline{1}$, $\overline{2}$, $\overline{3}$, $\overline{4}$ and $\overline{5}$.
- The SWAC (1950)[13] and Bendix G-15 (1956)[14][13] computers used the lowercase letters *u*, *v*, *w*, *x*, *y* and *z* for the values 10 to 15.
- The ORDVAC and ILLIAC I (1952) computers (and some derived designs, e.g. BRLESC) used the uppercase letters *K*, *S*, *N*, *J*, *F* and *L* for the values 10 to 15.[15][13]
- The Librascope LGP-30 (1956) used the letters *F*, *G*, *J*, *K*, *Q* and *W* for the values 10 to 15.[16][13]
- On the PERM (1956) computer, hexadecimal numbers were written as letters *O* for zero, *A* to *N* and *P* for 1 to 15. Many machine instructions had mnemonic hex-codes (*A*=add, *M*=multiply, *L*=load, *F*=fixed-point etc.); programs were written without instruction names.[17]
- The Honeywell Datamatic D-1000 (1957) used the lowercase letters *b*, *c*, *d*, *e*, *f*, and *g* whereas the Elbit 100 (1967) used the uppercase letters *B*, *C*, *D*, *E*, *F* and *G* for the values 10 to 15.[13]
- The Monrobot XI (1960) used the letters *S*, *T*, *U*, *V*, *W* and *X* for the values 10 to 15.[13]
- The NEC parametron computer NEAC 1103 (1960) used the letters *D*, *G*, *H*, *J*, *K* (and possibly *V*) for values 10–15.[18]
- The Pacific Data Systems 1020 (1964) used the letters *L*, *C*, *A*, *S*, *M* and *D* for the values 10 to 15.[13]
- New numeric symbols and names were introduced in the Bibi-binary notation by Boby Lapointe in 1968. This notation did not become very popular.
- Bruce Alan Martin of Brookhaven National Laboratory considered the choice of A–F "ridiculous". In a 1968 letter to the editor of the CACM, he proposed an entirely new set of symbols based on the bit locations, which did not gain much acceptance.[19]
- R. O. Whitaker of Rowco Engineering Co., in 1972, proposed a triangular font that allows "direct binary reading" in order to "permit both input and output from computers without respect to encoding matrices." [20]
- Some seven-segment display decoder chips (i.e., 74LS47) show unexpected output due to logic designed only to produce 0–9 correctly.[21]



Bruce Alan Martin's hexadecimal notation proposal[19]

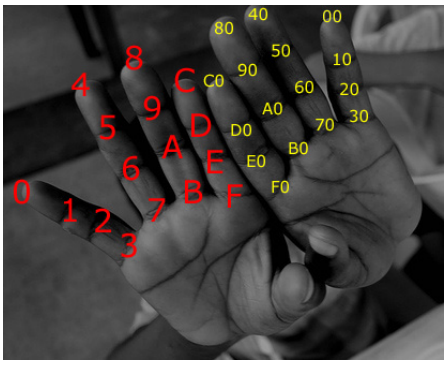
## Verbal and digital representations

Since there were no traditional numerals to represent the quantities from ten to fifteen, alphabetic letters were re-employed as a substitute. Most European languages lack non-decimal-based words for some of the numerals eleven to fifteen. Some people read hexadecimal numbers digit by digit, like a phone number, or using the NATO phonetic alphabet, the Joint Army/Navy Phonetic Alphabet, or a similar *ad-hoc* system. In the wake of the adoption of hexadecimal among IBM System/360 programmers, Magnuson (1968)[22] suggested a pronunciation guide that gave short names to the letters of hexadecimal – for instance, "A" was pronounced "ann", B "bet", C "chris", etc.[22] Another naming system was elaborated by Babb (2015), based on a joke in Silicon

Valley.[23] Yet another naming-system was published online by Rogers (2007)[24] that tries to make the verbal representation distinguishable in any case, even when the actual number does not contain numbers A–F. Examples are listed in the tables below.

Others have proposed using the verbal Morse Code conventions to express four-bit hexadecimal digits, with "dit" and "dah" representing zero and one, respectively, so that "0000" is voiced as "dit-dit-dit-dit" (....), dah-dit-dit-dah (-..-) voices the digit with a value of nine, and "dah-dah-dah-dah" (----) voices the hexadecimal digit for decimal 15.

Systems of counting on digits have been devised for both binary and hexadecimal. Arthur C. Clarke suggested using each finger as an on/off bit, allowing finger counting from zero to $1023_{10}$ on ten fingers.[25] Another system for counting up to $FF_{16}$ ($255_{10}$) is illustrated on the right.



Hexadecimal finger-counting scheme

Magnuson (1968)[22] naming method

| Number | Pronunciation |
|---|---|
| A | ann |
| B | bet |
| C | chris |
| D | dot |
| E | ernest |
| F | frost |
| 1A | annteen |
| A0 | annty |
| 5B | fifty-bet |
| A01C | annty christeen |
| 1AD0 | annteen dotty |
| 3A7D | thirty-ann seventy-dot |

Rogers (2007)[24] naming method

| Number | Pronunciation |
|---|---|
| A | ten |
| B | eleven |
| C | twelve |
| D | draze |
| E | eptwin |
| F | fim |
| 10 | tex |
| 11 | oneteek |
| 1F | fimteek |
| 50 | fiftek |
| C0 | twelftek |
| 100 | hundrek |
| 1000 | thousek |
| 3E | thirtek-eptwin |
| E1 | eptek-one |
| C4A | twelve-hundrek-fourtek-ten |
| 1743 | one-thousek-seven--hundrek-fourtek-three |

## Signs

The hexadecimal system can express negative numbers the same way as in decimal: −2A to represent $-42_{10}$ and so on.

Hexadecimal can also be used to express the exact bit patterns used in the processor, so a sequence of hexadecimal digits may represent a signed or even a floating-point value. This way, the negative number $-42_{10}$ can be written as FFFF FFD6 in a 32-bit CPU register (in two's-complement), as C228 0000 in a 32-bit FPU register or C045 0000 0000 0000 in a 64-bit FPU register (in the IEEE floating-point standard).

## Hexadecimal exponential notation

Just as decimal numbers can be represented in exponential notation, so too can hexadecimal numbers. By convention, the letter *P* (or *p*, for "power") represents *times two raised to the power of*, whereas *E* (or *e*) serves a similar purpose in decimal as part of the E notation. The number after the *P* is *decimal* and represents the *binary* exponent. Increasing the exponent by 1 multiplies by 2, not 16. 10.0p1 = 8.0p2 = 4.0p3 = 2.0p4 = 1.0p5. Usually, the number is normalized so that the leading hexadecimal digit is 1 (unless the value is exactly 0).

Example: 1.3DEp42 represents $1.3\mathrm{DE}_{16} \times 2^{42_{10}}$.

Hexadecimal exponential notation is required by the IEEE 754-2008 binary floating-point standard. This notation can be used for floating-point literals in the C99 edition of the C programming language.[26] Using the *%a* or *%A* conversion specifiers, this notation can be produced by implementations of the *printf* family of functions following the C99 specification[27] and Single Unix Specification (IEEE Std 1003.1) POSIX standard.[28]

# Conversion

## Binary conversion

Most computers manipulate binary data, but it is difficult for humans to work with a large number of digits for even a relatively small binary number. Although most humans are familiar with the base 10 system, it is much easier to map binary to hexadecimal than to decimal because each hexadecimal digit maps to a whole number of bits ($4_{10}$). This example converts $1111_2$ to base ten. Since each position in a binary numeral can contain either a 1 or a 0, its value may be easily determined by its position from the right:

- $0001_2 = 1_{10}$
- $0010_2 = 2_{10}$
- $0100_2 = 4_{10}$
- $1000_2 = 8_{10}$

Therefore:

$$\begin{aligned} 1111_2 &= 8_{10} + 4_{10} + 2_{10} + 1_{10} \\ &= 15_{10} \end{aligned}$$

With little practice, mapping $1111_2$ to $F_{16}$ in one step becomes easy: see table in written representation. The advantage of using hexadecimal rather than decimal increases rapidly with the size of the number. When the number becomes large, conversion to decimal is very tedious. However, when mapping to hexadecimal, it is trivial to regard the binary string as 4-digit groups and map each to a single hexadecimal digit.[29]

This example shows the conversion of a binary number to decimal, mapping each digit to the decimal value, and adding the results.

$$\begin{aligned} (01011110101101010010)_2 &= 262144_{10} + 65536_{10} + 32768_{10} + 16384_{10} + 8192_{10} + 2048_{10} + \\ &\quad 512_{10} + 256_{10} + 64_{10} + 16_{10} + 2_{10} \\ &= 387922_{10} \end{aligned}$$

Compare this to the conversion to hexadecimal, where each group of four digits can be considered independently, and converted directly:

$$\begin{aligned} (01011110101101010010)_2 &= 0101\ \ 1110\ \ 1011\ \ 0101\ \ 0010_2 \\ &= \ \ 5 \ \ \ \ E \ \ \ \ B \ \ \ \ 5 \ \ \ \ 2_{16} \\ &= 5EB52_{16} \end{aligned}$$

The conversion from hexadecimal to binary is equally direct.[29]

## Other simple conversions

Although quaternary (base 4) is little used, it can easily be converted to and from hexadecimal or binary. Each hexadecimal digit corresponds to a pair of quaternary digits and each quaternary digit corresponds to a pair of binary digits. In the above example $5\ E\ B\ 5_{16} = 11\ 32\ 23\ 11\ 02_4$.

The octal (base 8) system can also be converted with relative ease, although not quite as trivially as with bases 2 and 4. Each octal digit corresponds to three binary digits, rather than four. Therefore, we can convert between octal and hexadecimal via an intermediate conversion to binary followed by regrouping the binary digits in groups of either three or four.

## Division-remainder in source base

As with all bases there is a simple algorithm for converting a representation of a number to hexadecimal by doing integer division and remainder operations in the source base. In theory, this is possible from any base, but for most humans only decimal and for most computers only binary (which can be converted by far more efficient methods) can be easily handled with this method.

Let d be the number to represent in hexadecimal, and the series $h_i h_{i-1} ... h_2 h_1$ be the hexadecimal digits representing the number.

1. $i \leftarrow 1$
2. $h_i \leftarrow d \bmod 16$
3. $d \leftarrow (d - h_i) / 16$
4. If $d = 0$ (return series $h_i$) else increment i and go to step 2

"16" may be replaced with any other base that may be desired.

The following is a JavaScript implementation of the above algorithm for converting any number to a hexadecimal in String representation. Its purpose is to illustrate the above algorithm. To work with data seriously, however, it is much more advisable to work with bitwise operators.

```
function toHex(d) {
  var r = d % 16;
  if (d - r == 0) {
    return toChar(r);
  }
  return toHex((d - r) / 16) + toChar(r);
}

function toChar(n) {
  const alpha = "0123456789ABCDEF";
  return alpha.charAt(n);
}
```

## Conversion through addition and multiplication

It is also possible to make the conversion by assigning each place in the source base the hexadecimal representation of its place value — before carrying out multiplication and addition to get the final representation. For example, to convert the number B3AD to decimal, one can split the hexadecimal number into its digits: B ($11_{10}$), 3 ($3_{10}$), A ($10_{10}$) and D ($13_{10}$), and then get the final result by multiplying each decimal representation by $16^p$ ($p$ being the corresponding hex digit position, counting from right to left, beginning with 0). In this case, we have that:

$$B3AD = (11 \times 16^3) + (3 \times 16^2) + (10 \times 16^1) + (13 \times 16^0)$$

which is 45997 in base 10.

### Tools for conversion

Many computer systems provide a calculator utility capable of performing conversions between the various radices frequently including hexadecimal.

In Microsoft Windows, the Calculator utility can be set to Scientific mode (called Programmer mode in some versions), which allows conversions between radix 16 (hexadecimal), 10 (decimal), 8 (octal) and 2 (binary), the bases most commonly used by programmers. In Scientific Mode, the on-screen numeric keypad includes the hexadecimal digits A through F, which are active when "Hex" is selected. In hex mode, however, the Windows Calculator supports only integers.

| × | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 |
| 2 | 2 | 4 | 6 | 8 | A | C | E | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E | 20 |
| 3 | 3 | 6 | 9 | C | F | 12 | 15 | 18 | 1B | 1E | 21 | 24 | 27 | 2A | 2D | 30 |
| 4 | 4 | 8 | C | 10 | 14 | 18 | 1C | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C | 40 |
| 5 | 5 | A | F | 14 | 19 | 1E | 23 | 28 | 2D | 32 | 37 | 3C | 41 | 46 | 4B | 50 |
| 6 | 6 | C | 12 | 18 | 1E | 24 | 2A | 30 | 36 | 3C | 42 | 48 | 4E | 54 | 5A | 60 |
| 7 | 7 | E | 15 | 1C | 23 | 2A | 31 | 38 | 3F | 46 | 4D | 54 | 5B | 62 | 69 | 70 |
| 8 | 8 | 10 | 18 | 20 | 28 | 30 | 38 | 40 | 48 | 50 | 58 | 60 | 68 | 70 | 78 | 80 |
| 9 | 9 | 12 | 1B | 24 | 2D | 36 | 3F | 48 | 51 | 5A | 63 | 6C | 75 | 7E | 87 | 90 |
| A | A | 14 | 1E | 28 | 32 | 3C | 46 | 50 | 5A | 64 | 6E | 78 | 82 | 8C | 96 | A0 |
| B | B | 16 | 21 | 2C | 37 | 42 | 4D | 58 | 63 | 6E | 79 | 84 | 8F | 9A | A5 | B0 |
| C | C | 18 | 24 | 30 | 3C | 48 | 54 | 60 | 6C | 78 | 84 | 90 | 9C | A8 | B4 | C0 |
| D | D | 1A | 27 | 34 | 41 | 4E | 5B | 68 | 75 | 82 | 8F | 9C | A9 | B6 | C3 | D0 |
| E | E | 1C | 2A | 38 | 46 | 54 | 62 | 70 | 7E | 8C | 9A | A8 | B6 | C4 | D2 | E0 |
| F | F | 1E | 2D | 3C | 4B | 5A | 69 | 78 | 87 | 96 | A5 | B4 | C3 | D2 | E1 | F0 |
| 10 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | A0 | B0 | C0 | D0 | E0 | F0 | 100 |

A hexadecimal multiplication table

# Elementary arithmetic

Elementary operations such addition, subtraction, multiplication and division can be carried out indirectly through conversion to an alternate numeral system, such as the commonly-used decimal system or the binary system where each hex digit corresponds to four binary digits.

Alternatively, one can also perform elementary operations directly within the hex system itself — by relying on its addition/multiplication tables and its corresponding standard algorithms such as long division and the traditional subtraction algorithm.

# Real numbers

### Rational numbers

As with other numeral systems, the hexadecimal system can be used to represent rational numbers, although repeating expansions are common since sixteen ($10_{16}$) has only a single prime factor; two.

For any base, 0.1 (or "1/10") is always equivalent to one divided by the representation of that base value in its own number system. Thus, whether dividing one by two for binary or dividing one by sixteen for hexadecimal, both of these fractions are written as `0.1`. Because the radix 16 is a perfect square ($4^2$), fractions expressed in hexadecimal have an odd period much more often than decimal ones, and there are no cyclic numbers (other than trivial single digits). Recurring digits are exhibited when the denominator in lowest terms has a prime factor not found in the radix; thus, when using hexadecimal notation, all fractions with denominators that are not a power of two result in an infinite string of recurring digits (such as thirds and fifths). This makes hexadecimal (and binary) less convenient than decimal for representing rational numbers since a larger proportion lie outside its range of finite representation.

All rational numbers finitely representable in hexadecimal are also finitely representable in decimal, duodecimal and sexagesimal: that is, any hexadecimal number with a finite number of digits also has a finite number of digits when expressed in those other bases. Conversely, only a fraction of those finitely representable in the latter bases are finitely representable in hexadecimal.

For example, decimal 0.1 corresponds to the infinite recurring representation 0.19 in hexadecimal. However, hexadecimal is more efficient than duodecimal and sexagesimal for representing fractions with powers of two in the denominator. For example, $0.0625_{10}$ (one-sixteenth) is equivalent to $0.1_{16}$, $0.09_{12}$, and $0;3,45_{60}$.

| n | Decimal<br>Prime factors of: base, b = 10: 2, 5; b − 1 = 9: 3 | | Hexadecimal<br>Prime factors of: base, b = $16_{10}$ = 10: 2; b − 1 = $15_{10}$ = F: 3, 5 | | | |
|---|---|---|---|---|---|---|
| | Reciprocal | Prime factors | Positional representation (hexadecimal) | Positional representation (decimal for comparison) | Prime factors | Reciprocal |
| 2 | 1/2 | 2 | **0.8** | **0.5** | 2 | 1/2 |
| 3 | 1/3 | 3 | **0.**5555... = **0.**$\overline{5}$ | **0.**3333... = **0.**$\overline{3}$ | 3 | 1/3 |
| 4 | 1/4 | 2 | **0.4** | **0.25** | 2 | 1/4 |
| 5 | 1/5 | 5 | **0.**$\overline{3}$ | **0.2** | 5 | 1/5 |
| 6 | 1/6 | 2, 3 | **0.2**$\overline{A}$ | **0.1**$\overline{6}$ | 2, 3 | 1/6 |
| 7 | 1/7 | 7 | **0.**$\overline{249}$ | **0.**$\overline{142857}$ | 7 | 1/7 |
| 8 | 1/8 | 2 | **0.2** | **0.125** | 2 | 1/8 |
| 9 | 1/9 | 3 | **0.**$\overline{1C7}$ | **0.**$\overline{1}$ | 3 | 1/9 |
| 10 | 1/10 | 2, 5 | **0.1**$\overline{9}$ | **0.1** | 2, 5 | 1/A |
| 11 | 1/11 | 11 | **0.**$\overline{1745D}$ | **0.**$\overline{09}$ | B | 1/B |
| 12 | 1/12 | 2, 3 | **0.1**$\overline{5}$ | **0.08**$\overline{3}$ | 2, 3 | 1/C |
| 13 | 1/13 | 13 | **0.**$\overline{13B}$ | **0.**$\overline{076923}$ | D | 1/D |
| 14 | 1/14 | 2, 7 | **0.1**$\overline{249}$ | **0.0**$\overline{714285}$ | 2, 7 | 1/E |
| 15 | 1/15 | 3, 5 | **0.**$\overline{1}$ | **0.0**$\overline{6}$ | 3, 5 | 1/F |
| 16 | 1/16 | 2 | **0.1** | **0.0625** | 2 | 1/10 |
| 17 | 1/17 | 17 | **0.**$\overline{0F}$ | **0.**$\overline{0588235294117647}$ | 11 | 1/11 |
| 18 | 1/18 | 2, 3 | **0.0**$\overline{E38}$ | **0.0**$\overline{5}$ | 2, 3 | 1/12 |
| 19 | 1/19 | 19 | **0.**$\overline{0D79435E5}$ | **0.**$\overline{052631578947368421}$ | 13 | 1/13 |
| 20 | 1/20 | 2, 5 | **0.0**$\overline{C}$ | **0.05** | 2, 5 | 1/14 |
| 21 | 1/21 | 3, 7 | **0.**$\overline{0C3}$ | **0.**$\overline{047619}$ | 3, 7 | 1/15 |
| 22 | 1/22 | 2, 11 | **0.0**$\overline{BA2E8}$ | **0.0**$\overline{45}$ | 2, B | 1/16 |
| 23 | 1/23 | 23 | **0.**$\overline{0B21642C859}$ | **0.**$\overline{043478260869565217391}$ | 17 | 1/17 |
| 24 | 1/24 | 2, 3 | **0.0**$\overline{A}$ | **0.041**$\overline{6}$ | 2, 3 | 1/18 |
| 25 | 1/25 | 5 | **0.**$\overline{0A3D7}$ | **0.04** | 5 | 1/19 |
| 26 | 1/26 | 2, 13 | **0.0**$\overline{9D8}$ | **0.0**$\overline{384615}$ | 2, D | 1/1A |
| 27 | 1/27 | 3 | **0.**$\overline{097B425ED}$ | **0.**$\overline{037}$ | 3 | 1/1B |
| 28 | 1/28 | 2, 7 | **0.0**$\overline{924}$ | **0.03**$\overline{571428}$ | 2, 7 | 1/1C |
| 29 | 1/29 | 29 | **0.**$\overline{08D3DCB}$ | **0.**$\overline{0344827586206896551724137931}$ | 1D | 1/1D |
| 30 | 1/30 | 2, 3, 5 | **0.0**$\overline{8}$ | **0.0**$\overline{3}$ | 2, 3, 5 | 1/1E |
| 31 | 1/31 | 31 | **0.**$\overline{08421}$ | **0.**$\overline{032258064516129}$ | 1F | 1/1F |
| 32 | 1/32 | 2 | **0.08** | **0.03125** | 2 | 1/20 |
| 33 | 1/33 | 3, 11 | **0.**$\overline{07C1F}$ | **0.**$\overline{03}$ | 3, B | 1/21 |
| 34 | 1/34 | 2, 17 | **0.0**$\overline{78}$ | **0.0**$\overline{2941176470588235}$ | 2, 11 | 1/22 |
| 35 | 1/35 | 5, 7 | **0.**$\overline{075}$ | **0.0**$\overline{285714}$ | 5, 7 | 1/23 |
| 36 | 1/36 | 2, 3 | **0.0**$\overline{71C}$ | **0.02**$\overline{7}$ | 2, 3 | 1/24 |

## Irrational numbers

The table below gives the expansions of some common irrational numbers in decimal and hexadecimal.

| Number | Positional representation | |
|---|---|---|
| | Decimal | Hexadecimal |
| $\sqrt{2}$ (the length of the diagonal of a unit square) | 1.414 213 562 373 095 048... | 1.6A09E667F3BCD... |
| $\sqrt{3}$ (the length of the diagonal of a unit cube) | 1.732 050 807 568 877 293... | 1.BB67AE8584CAA... |
| $\sqrt{5}$ (the length of the diagonal of a 1×2 rectangle) | 2.236 067 977 499 789 696... | 2.3C6EF372FE95... |
| $\varphi$ (phi, the golden ratio = $(1+\sqrt{5})/2$) | 1.618 033 988 749 894 848... | 1.9E3779B97F4A... |
| $\pi$ (pi, the ratio of circumference to diameter of a circle) | 3.141 592 653 589 793 238 462 643 383 279 502 884 197 169 399 375 105... | 3.243F6A8885A308D313198A2E0 3707344A4093822299F31D008... |
| $e$ (the base of the natural logarithm) | 2.718 281 828 459 045 235... | 2.B7E151628AED2A6B... |
| $\tau$ (the Thue–Morse constant) | 0.412 454 033 640 107 597... | 0.6996 9669 9669 6996... |
| $\gamma$ (the limiting difference between the harmonic series and the natural logarithm) | 0.577 215 664 901 532 860... | 0.93C467E37DB0C7A4D1B... |

## Powers

Powers of two have very simple expansions in hexadecimal. The first sixteen powers of two are shown below.

| $2^x$ | Value | Value (Decimal) |
|---|---|---|
| $2^0$ | 1 | 1 |
| $2^1$ | 2 | 2 |
| $2^2$ | 4 | 4 |
| $2^3$ | 8 | 8 |
| $2^4$ | $10_{hex}$ | $16_{dec}$ |
| $2^5$ | $20_{hex}$ | $32_{dec}$ |
| $2^6$ | $40_{hex}$ | $64_{dec}$ |
| $2^7$ | $80_{hex}$ | $128_{dec}$ |
| $2^8$ | $100_{hex}$ | $256_{dec}$ |
| $2^9$ | $200_{hex}$ | $512_{dec}$ |
| $2^A$ ($2^{10_{dec}}$) | $400_{hex}$ | $1024_{dec}$ |
| $2^B$ ($2^{11_{dec}}$) | $800_{hex}$ | $2048_{dec}$ |
| $2^C$ ($2^{12_{dec}}$) | $1000_{hex}$ | $4096_{dec}$ |
| $2^D$ ($2^{13_{dec}}$) | $2000_{hex}$ | $8192_{dec}$ |
| $2^E$ ($2^{14_{dec}}$) | $4000_{hex}$ | $16{,}384_{dec}$ |
| $2^F$ ($2^{15_{dec}}$) | $8000_{hex}$ | $32{,}768_{dec}$ |
| $2^{10}$ ($2^{16_{dec}}$) | $10000_{hex}$ | $65{,}536_{dec}$ |

# Cultural history

The traditional Chinese units of measurement were base-16. For example, one jīn (斤) in the old system equals sixteen taels. The suanpan (Chinese abacus) can be used to perform hexadecimal calculations such as additions and subtractions.[30]

As with the duodecimal system, there have been occasional attempts to promote hexadecimal as the preferred numeral system. These attempts often propose specific pronunciation and symbols for the individual numerals.[31] Some proposals unify standard measures so that they are multiples of 16.[32][33] An early such proposal was put forward by John W. Nystrom in *Project of a New System of Arithmetic, Weight, Measure and Coins: Proposed to be called the Tonal System, with Sixteen to the Base*, published in 1862.[34] Nystrom among other things suggested hexadecimal time, which subdivides a day by 16, so that there are 16 "hours" (or "10 *tims*", pronounced *tontim*) in a day.[35]

The word *hexadecimal* is first recorded in 1952.[36] It is macaronic in the sense that it combines Greek ἕξ (hex) "six" with Latinate *-decimal*. The all-Latin alternative *sexadecimal* (compare the word *sexagesimal* for base 60) is older, and sees at least occasional use from the late 19th century.[37] It is still in use in the 1950s in Bendix documentation. Schwartzman (1994) argues that use of *sexadecimal* may have been avoided because of its suggestive abbreviation to *sex*.[38] Many western languages since the 1960s have adopted terms equivalent in formation to *hexadecimal* (e.g. French *hexadécimal*, Italian *esadecimale*, Romanian *hexazecimal*, Serbian *хексадецимални*, etc.) but others have introduced terms which substitute native words for "sixteen" (e.g. Greek δεκαεξαδικός, Icelandic *sextándakerfi*, Russian *шестнадцатеричной* etc.)

Terminology and notation did not become settled until the end of the 1960s. Donald Knuth in 1969 argued that the etymologically correct term would be *senidenary*, or possibly *sedenary*, a Latinate term intended to convey "grouped by 16" modelled on *binary*, *ternary* and *quaternary* etc. According to Knuth's argument, the correct terms for *decimal* and *octal* arithmetic would be *denary* and *octonary*, respectively.[39] Alfred B. Taylor used *senidenary* in his mid-1800s work on alternative number bases, although he rejected base 16 because of its "incommodious number of digits".[40][41]

The now-current notation using the letters A to F establishes itself as the de facto standard beginning in 1966, in the wake of the publication of the Fortran IV manual for IBM System/360, which (unlike earlier variants of Fortran) recognizes a standard for entering hexadecimal constants.[42] As noted above, alternative notations were used by NEC (1960) and The Pacific Data Systems 1020 (1964). The standard adopted by IBM seems to have become widely adopted by 1968, when Bruce Alan Martin in his letter to the editor of the CACM complains that

> With the ridiculous choice of letters A, B, C, D, E, F as hexadecimal number symbols adding to already troublesome problems of distinguishing octal (or hex) numbers from decimal numbers (or variable names), the time is overripe for reconsideration of our number symbols. This should have been done before poor choices gelled into a de facto standard!

Martin's argument was that use of numerals 0 to 9 in nondecimal numbers "imply to us a base-ten place-value scheme": "Why not use entirely new symbols (and names) for the seven or fifteen nonzero digits needed in octal or hex. Even use of the letters A through P would be an improvement, but entirely new symbols could reflect the binary nature of the system".[19] He also argued that "re-using alphabetic letters for numerical digits represents a gigantic backward step from the invention of distinct, non-alphabetic glyphs for numerals sixteen centuries ago" (as Brahmi numerals, and later in a Hindu–Arabic numeral system), and that the recent ASCII standards (ASA X3.4-1963 and USAS X3.4-1968) "should have preserved six code table positions following the ten decimal digits -- rather than needlessly filling these with punctuation characters" (":;<=>?") that might have been placed elsewhere among the 128 available positions.

# Base16 (transfer encoding)

**Base16** (as a proper name without a space) can also refer to a binary to text encoding belonging to the same family as Base32, Base58, and Base64.

In this case, data is broken into 4-bit sequences, and each value (between 0 and 15 inclusively) is encoded using one of 16 symbols from the ASCII character set. Although any 16 symbols from the ASCII character set can be used, in practice the ASCII digits '0'–'9' and the letters 'A'–'F' (or the lowercase 'a'–'f') are always chosen in order to align with standard written notation for hexadecimal numbers.

There are several advantages of Base16 encoding:

- Most programming languages already have facilities to parse ASCII-encoded hexadecimal
- Being exactly half a byte, 4-bits is easier to process than the 5 or 6 bits of Base32 and Base64 respectively
- The symbols 0–9 and A-F are universal in hexadecimal notation, so it is easily understood at a glance without needing to rely on a symbol lookup table

- Many CPU architectures have dedicated instructions that allow access to a half-byte (otherwise known as a "nibble"), making it more efficient in hardware than Base32 and Base64

The main disadvantages of Base16 encoding are:

- Space efficiency is only 50%, since each 4-bit value from the original data will be encoded as an 8-bit byte. In contrast, Base32 and Base64 encodings have a space efficiency of 63% and 75% respectively.
- Possible added complexity of having to accept both uppercase and lowercase letters

Support for Base16 encoding is ubiquitous in modern computing. It is the basis for the W3C standard for URL percent encoding, where a character is replaced with a percent sign "%" and its Base16-encoded form. Most modern programming languages directly include support for formatting and parsing Base16-encoded numbers.

# See also

- Base32, Base64 (content encoding schemes)
- Hexadecimal time
- IBM hexadecimal floating-point
- Hex editor
- Hex dump
- Bailey–Borwein–Plouffe formula (BBP)
- Hexspeak

# References


1. "The hexadecimal system" (https://www.ionos.co.uk/digitalguide/server/know-how/hexadecimal-system/). *IONOS Digitalguide*. Retrieved 2022-08-26.
2. Knuth, Donald Ervin (1986). *The TeXbook* (https://www.worldcat.org/oclc/12973034). Duane Bibby. Reading, Mass. ISBN 0-201-13447-0. OCLC 12973034 (https://www.worldcat.org/oclc/12973034).
3. The string `"\x1B[0m\x1B[25;1H"` specifies the character sequence `Esc [ 0 m Esc [ 2 5 ; 1 H Nul`. These are the escape sequences used on an ANSI terminal that reset the character set and color, and then move the cursor to line 25.
4. "The Unicode Standard, Version 7" (https://www.unicode.org/charts/PDF/U2000.pdf) (PDF). *Unicode*. Archived (https://web.archive.org/web/20160303175510/http://www.unicode.org/charts/PDF/U2000.pdf) (PDF) from the original on 2016-03-03. Retrieved 2018-10-28.
5. "Hexadecimal web colors explained" (https://web.archive.org/web/20060422004336/http://www.web-colors-explained.com/hex.php). Archived from the original (http://www.web-colors-explained.com/hex.php) on 2006-04-22. Retrieved 2006-01-11.
6. "ISO-8859-1 (ISO Latin 1) Character Encoding" (https://www.ic.unicamp.br/~stolfi/EXPORT/www/ISO-8859-1-Encoding.html). *www.ic.unicamp.br*. Archived (https://web.archive.org/web/20190629203430/http://www.ic.unicamp.br/~stolfi/EXPORT/www/ISO-8859-1-Encoding.html) from the original on 2019-06-29. Retrieved 2019-06-26.
7. "Modula-2 – Vocabulary and representation" (http://modula2.org/reference/vocabulary.php). *Modula −2*. Archived (https://web.archive.org/web/20151213053318/http://www.modula2.org/reference/vocabulary.php) from the original on 2015-12-13. Retrieved 2015-11-01.
8. "An Introduction to VHDL Data Types" (https://www.fpgatutorial.com/vhdl-types-and-conversions#vhdl-assign-data). *FPGA Tutorial*. 2020-05-10. Archived (https://web.archive.org/web/20200823094252/https://www.fpgatutorial.com/vhdl-types-and-conversions/#vhdl-assign-data) from the original on 2020-08-23. Retrieved 2020-08-21.

9. "*read-base* variable in Common Lisp" (http://www.lispworks.com/documentation/HyperSpec/Body/v_rd_bas.htm). *CLHS*. Archived (https://web.archive.org/web/20160203221612/http://www.lispworks.com/documentation/HyperSpec/Body/v_rd_bas.htm) from the original on 2016-02-03. Retrieved 2015-01-10.
10. "*print-base* variable in Common Lisp" (http://www.lispworks.com/documentation/HyperSpec/Body/v_pr_bas.htm#STprint-baseST). *CLHS*. Archived (https://web.archive.org/web/20141226172420/http://www.lispworks.com/documentation/HyperSpec/Body/v_pr_bas.htm#STprint-baseST) from the original on 2014-12-26. Retrieved 2015-01-10.
11. MSX is Coming — Part 2: Inside MSX (http://www.atarimagazines.com/compute/issue56/107_1_MSX_IS_COMING.php) Archived (https://web.archive.org/web/20101124111223/http://www.atarimagazines.com/compute/issue56/107_1_MSX_IS_COMING.php) 2010-11-24 at the Wayback Machine Compute!, issue 56, January 1985, p. 52
12. BBC BASIC programs are not fully portable to Microsoft BASIC (without modification) since the latter takes & to prefix octal values. (Microsoft BASIC primarily uses &0 to prefix octal, and it uses &H to prefix hexadecimal, but the ampersand alone yields a default interpretation as an octal prefix.
13. Savard, John J. G. (2018) [2005]. "Computer Arithmetic" (http://www.quadibloc.com/comp/cp02.htm). *quadibloc*. The Early Days of Hexadecimal. Archived (https://web.archive.org/web/20180716102439/http://www.quadibloc.com/comp/cp02.htm) from the original on 2018-07-16. Retrieved 2018-07-16.
14. "2.1.3 Sexadecimal notation". *G15D Programmer's Reference Manual* (http://bitsavers.trailing-edge.com/pdf/bendix/g-15/G15D_Programmers_Ref_Man.pdf) (PDF). Los Angeles, CA, USA: Bendix Computer, Division of Bendix Aviation Corporation. p. 4. Archived (https://web.archive.org/web/20170601222212/http://bitsavers.trailing-edge.com/pdf/bendix/g-15/G15D_Programmers_Ref_Man.pdf) (PDF) from the original on 2017-06-01. Retrieved 2017-06-01. "This base is used because a group of four bits can represent any one of sixteen different numbers (zero to fifteen). By assigning a symbol to each of these combinations we arrive at a notation called sexadecimal (usually hex in conversation because nobody wants to abbreviate sex). The symbols in the sexadecimal language are the ten decimal digits and, on the G-15 typewriter, the letters u, v, w, x, y and z. These are arbitrary markings; other computers may use different alphabet characters for these last six digits."
15. Gill, S.; Neagher, R. E.; Muller, D. E.; Nash, J. P.; Robertson, J. E.; Shapin, T.; Whesler, D. J. (1956-09-01). Nash, J. P. (ed.). "ILLIAC Programming – A Guide to the Preparation of Problems For Solution by the University of Illinois Digital Computer" (http://www.textfiles.com/bitsavers/pdf/illiac/ILLIAC/ILLIAC_programming_Sep56.pdf) (PDF). *bitsavers.org* (Fourth printing. Revised and corrected ed.). Urbana, Illinois, USA: Digital Computer Laboratory, Graduate College, University of Illinois. pp. 3–2. Archived (https://web.archive.org/web/20170531153804/http://www.textfiles.com/bitsavers/pdf/illiac/ILLIAC/ILLIAC_programming_Sep56.pdf) (PDF) from the original on 2017-05-31. Retrieved 2014-12-18.
16. *ROYAL PRECISION Electronic Computer LGP – 30 PROGRAMMING MANUAL* (http://ed-thelen.org/comp-hist/lgp-30-man.html#R4.13). Port Chester, New York: Royal McBee Corporation. April 1957. Archived (https://web.archive.org/web/20170531153004/http://ed-thelen.org/comp-hist/lgp-30-man.html) from the original on 2017-05-31. Retrieved 2017-05-31. (NB. This somewhat odd sequence was from the next six sequential numeric keyboard codes in the LGP-30's 6-bit character code.)
17. Manthey, Steffen; Leibrandt, Klaus (2002-07-02). "Die PERM und ALGOL" (http://www.manthey.cc/sites/seminars/src/History.pdf) (PDF) (in German). Retrieved 2018-05-19.
18. *NEC Parametron Digital Computer Type NEAC-1103* (http://archive.computerhistory.org/resources/text/NEC/NEC.1103.1958102646285.pdf) (PDF). Tokyo, Japan: Nippon Electric Company Ltd. 1960. Cat. No. 3405-C. Archived (https://web.archive.org/web/20170531112850/http://archive.computerhistory.org/resources/text/NEC/NEC.1103.1958102646285.pdf) (PDF) from the original on 2017-05-31. Retrieved 2017-05-31.

19. Martin, Bruce Alan (October 1968). "Letters to the editor: On binary notation". *Communications of the ACM*. Associated Universities Inc. **11** (10): 658. doi:10.1145/364096.364107 (https://doi.org/10.1145%2F364096.364107). S2CID 28248410 (https://api.semanticscholar.org/CorpusID:28248410).
20. Whitaker, R. O. (January 1972). "Letters to the editor: More on man/machine". *Datamation*. p. 103.
21. "Archived copy" (https://www.ti.com/lit/gpn/sn74ls47). Archived (https://web.archive.org/web/20211020192609/https://www.ti.com/lit/ds/symlink/sn74ls47.pdf?ts=1634757966777) (PDF) from the original on 2021-10-20. Retrieved 2021-09-15.
22. Magnuson, Robert A. (January 1968). "A hexadecimal pronunciation guide". *Datamation*. Vol. 14, no. 1. p. 45.
23. Babb, Tim (2015). "How to pronounce hexadecimal" (https://www.bzarg.com/p/how-to-pronounce-hexadecimal/). *Bzarg*. Archived (https://web.archive.org/web/20201111174319/https://www.bzarg.com/p/how-to-pronounce-hexadecimal/) from the original on 2020-11-11. Retrieved 2021-01-01.
24. Rogers, S.R. (2007). "Hexadecimal number words" (http://www.intuitor.com/hex/words.html). *Intuitor*. Archived (https://web.archive.org/web/20190917015855/http://www.intuitor.com/hex/words.html) from the original on 2019-09-17. Retrieved 2019-08-26.
25. Clarke, Arthur; Pohl, Frederik (2008). *The Last Theorem* (https://archive.org/details/lasttheorem00clar). Ballantine. p. 91 (https://archive.org/details/lasttheorem00clar/page/91). ISBN 978-0007289981.
26. "ISO/IEC 9899:1999 – Programming languages – C" (http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=29237). *ISO*. Iso.org. 2011-12-08. Archived (https://web.archive.org/web/20161010112929/http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=29237) from the original on 2016-10-10. Retrieved 2014-04-08.
27. "Rationale for International Standard – Programming Languages – C" (http://www.open-std.org/jtc1/sc22/wg14/www/C99RationaleV5.10.pdf) (PDF). *Open Standards*. 5.10. April 2003. pp. 52, 153–154, 159. Archived (https://web.archive.org/web/20160606072228/http://www.open-std.org/jtc1/sc22/wg14/www/C99RationaleV5.10.pdf) (PDF) from the original on 2016-06-06. Retrieved 2010-10-17.
28. The IEEE and The Open Group (2013) [2001]. "dprintf, fprintf, printf, snprintf, sprintf – print formatted output" (http://pubs.opengroup.org/onlinepubs/9699919799/functions/printf.html). *The Open Group Base Specifications* (Issue 7, IEEE Std 1003.1, 2013 ed.). Archived (https://web.archive.org/web/20160621211105/http://pubs.opengroup.org/onlinepubs/9699919799/functions/printf.html) from the original on 2016-06-21. Retrieved 2016-06-21.
29. Mano, M. Morris; Ciletti, Michael D. (2013). *Digital Design – With an Introduction to the Verilog HDL* (Fifth ed.). Pearson Education. pp. 6, 8–10. ISBN 978-0-13-277420-8.
30. "算盤 Hexadecimal Addition & Subtraction on a Chinese Abacus" (http://totton.idirect.com/soroban/Hex_as/). *totton.idirect.com*. Archived (https://web.archive.org/web/20190706221609/http://totton.idirect.com/soroban/Hex_as/) from the original on 2019-07-06. Retrieved 2019-06-26.
31. "Base 4^2 Hexadecimal Symbol Proposal" (http://www.hauptmech.com/base42). *Hauptmech*. Archived (https://web.archive.org/web/20211020192525/http://www.hauptmech.com/base42/wiki/index.php?title=Main_Page) from the original on 2021-10-20. Retrieved 2008-09-04.
32. "Intuitor Hex Headquarters" (http://www.intuitor.com/hex/). *Intuitor*. Archived (https://web.archive.org/web/20100904144850/http://www.intuitor.com/hex/) from the original on 2010-09-04. Retrieved 2018-10-28.
33. Niemietz, Ricardo Cancho (2003-10-21). "A proposal for addition of the six Hexadecimal digits (A-F) to Unicode" (http://std.dkuug.dk/jtc1/sc2/wg2/docs/n2677). *DKUUG Standardizing*. Archived (https://web.archive.org/web/20110604035450/http://std.dkuug.dk/jtc1/sc2/wg2/docs/n2677) from the original on 2011-06-04. Retrieved 2018-10-28.

34. Nystrom, John William (1862). *Project of a New System of Arithmetic, Weight, Measure and Coins: Proposed to be called the Tonal System, with Sixteen to the Base* (https://archive.org/details/bub_gb_aNYGAAAAYAAJ). Philadelphia: Lippincott.
35. Nystrom (1862), p. 33: "In expressing time, angle of a circle, or points on the compass, the unit *tim* should be noted as integer, and parts thereof as *tonal fractions*, as 5·86 *tims* is five times and *metonby* [*"sutim and metonby" John Nystrom accidentally gives part of the number in decimal names; in Nystrom's pronunciation scheme, 5=su, 8=me, 6=by, c.f. unifoundry.com (http://www.unifoundry.com/tonal/index.html) Archived (https://web.archive.org/web/20210519080658/http://www.unifoundry.com/tonal/index.html) 2021-05-19 at the Wayback Machine ]."
36. C. E. Fröberg, *Hexadecimal Conversion Tables*, Lund (1952).
37. *The Century Dictionary* of 1895 has *sexadecimal* in the more general sense of "relating to sixteen". An early explicit use of *sexadecimal* in the sense of "using base 16" is found also in 1895, in the *Journal of the American Geographical Society of New York, vols. 27–28, p. 197.*
38. Schwartzman, Steven (1994). *The Words of Mathematics: An etymological dictionary of mathematical terms used in English*. The Mathematical Association of America. p. 105. ISBN 0-88385-511-9. s.v. hexadecimal
39. Knuth, Donald. (1969). *The Art of Computer Programming, Volume 2*. ISBN 0-201-03802-1. (Chapter 17.)
40. Alfred B. Taylor, Report on Weights and Measures (https://archive.org/details/reportonweights00taylgoog), Pharmaceutical Association, 8th Annual Session, Boston, 15 September 1859. See pages and 33 and 41.
41. Alfred B. Taylor, "Octonary numeration and its application to a system of weights and measures", *Proc Amer. Phil. Soc.* Vol XXIV (https://books.google.com/books?id=KsAUAAAAYAAJ&pg=PA296) Archived (https://web.archive.org/web/20160624070056/https://books.google.com/books?id=KsAUAAAAYAAJ&pg=PA296) 2016-06-24 at the Wayback Machine, Philadelphia, 1887; pages 296–366. See pages 317 and 322.
42. IBM System/360 FORTRAN IV Language (http://www.bitsavers.org/pdf/ibm/360/fortran/C28-6515-6_FORTRAN_IV_Language_1966.pdf) Archived (https://web.archive.org/web/20210519073220/http://www.bitsavers.org/pdf/ibm/360/fortran/C28-6515-6_FORTRAN_IV_Language_1966.pdf) 2021-05-19 at the Wayback Machine (1966), p. 13.

---