Hexadecimal

From Wikipedia, the free encyclopedia

In mathematics and computing, **hexadecimal** (also **base 16**, or **hex**) is a positional numeral system with a radix, or base, of 16. It uses sixteen distinct symbols, most often the symbols **0–9** to represent values zero to nine, and **A**, **B**, **C**, **D**, **E**, **F** (or alternatively **a**, **b**, **c**, **d**, **e**, **f**) to represent values ten to fifteen.

Hexadecimal numerals are widely used by computer system designers and programmers. As each hexadecimal digit represents four binary digits (bits), it allows a more human-friendly representation of binary-coded values. One hexadecimal digit represents a nibble (4 bits), which is half of an octet or byte (8 bits). For example, a single byte can have values ranging from 00000000 to 11111111 in binary form, but this may be more conveniently represented as 00 to FF in hexadecimal.

In a non-programming context, a subscript is typically used to give the radix, for example the decimal value 10,995 would be expressed in hexadecimal as $2AF3_{16}$. Several notations are used to support hexadecimal representation of constants in programming languages, usually involving a prefix or suffix. The prefix "0x" is used in C and related languages, where this value might be denoted as 0x2AF3.

Contents

Representation

Written representation

Using 0–9 and A–F

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 is equal to 345_{10}. Some authors prefer a text subscript, such as 159_{decimal} and 159_{hex}, or 159_{d} and 159_{h}.

$\mathbf{0}_{\text{hex}}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{\text{hex}}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{\text{hex}}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{\text{hex}}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{\text{hex}}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{\text{hex}}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{\text{hex}}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{\text{hex}}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{\text{hex}}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{\text{hex}}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{\text{hex}}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{\text{hex}}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{\text{hex}}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{\text{hex}}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{\text{hex}}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{\text{hex}}$	=	15_{dec}	=	17_{oct}	1	1	1	1

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

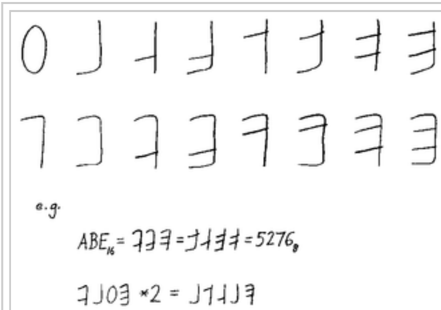
- In URIs (including URLs), character codes are written as hexadecimal pairs prefixed with `%`: `http://www.example.com/name%20with%20spaces` where `%20` is the space (blank) character, ASCII code point 20 in hex, 32 in decimal.
- In XML and XHTML, characters can be expressed as hexadecimal numeric character references using the notation `&#x`*`code`*`;`, where the *x* denotes that *code* is a hex code point (of 1- to 6-digits) assigned to the character in the Unicode standard. Thus `’` represents the right single quotation mark ('), Unicode code point number 2019 in hex, 8217 (thus `’` in decimal).[1]
- 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 `#FFFFFF` .[2] CSS allows 3-hexdigit abbreviations with one hexdigit per component: #FA3 abbreviates #FFAA33 (a golden orange:).
- *nix (Unix and related) shells, AT&T assembly language and likewise the C programming language, which was designed for Unix (and the syntactic descendants of C – including C++, C#, 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 (plus a trailing NUL to mark the end of the string) 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 MIME (e-mail extensions) quoted-printable encoding, characters that cannot be represented as literal ASCII characters are represented by their codes as two hexadecimal digits (in ASCII) prefixed by an *equal to* sign `=`, as in `Espa=F1a` to send "España" (Spain). (Hexadecimal F1, equal to decimal 241, is the code number for the lower case n with tilde in the ISO/IEC 8859-1 character set.)
- In Intel-derived assembly languages and Modula-2,[4] 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`
- Other assembly languages (6502, Motorola), Pascal, Delphi, some versions of BASIC (Commodore), Game Maker Language, Godot and Forth use `$` as a prefix: `$5A3`.
- Some assembly languages (Microchip) use the notation `H'ABCD'` (for $ABCD_{16}$).
- Ada and VHDL enclose hexadecimal numerals in based "numeric quotes": `16#5A3#`. For bit vector constants VHDL uses the notation `x"5A3"`.[5]
- 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`...
- In early systems when a Macintosh crashed, one or two lines of hexadecimal code would be displayed under the Sad Mac to tell the user what went wrong.
- Common Lisp uses the prefixes `#x` and `#16r`. Setting the variables *read-base*[6] and *print-base*[7] to 16 can also 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,[8] QuickBASIC, FreeBASIC and Visual Basic prefix hexadecimal numbers with `&H`: `&H5A3`
- BBC BASIC and Locomotive BASIC use `&` for hex.[9]
- 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 System 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.
- Donald Knuth introduced the use of a particular typeface to represent a particular radix in his book *The TeXbook*.[10] Hexadecimal representations are written there in a typewriter typeface: `5A3`
- Any IPv6 address can be written as eight groups of four hexadecimal digits, where each group is separated by a colon (`:`). This, for example, is a valid IPv6 address: 2001:0db8:85a3:0000:0000:8a2e:0370:7334; this can be abbreviated as 2001:db8:85a3::8a2e:370:7334. By contrast, 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}`.

There is no universal convention to use lowercase or uppercase for the letter digits, and each is prevalent or preferred in particular environments by community standards or convention.

Early written representations

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



Bruce Alan Martin's hexadecimal notation proposal[11]

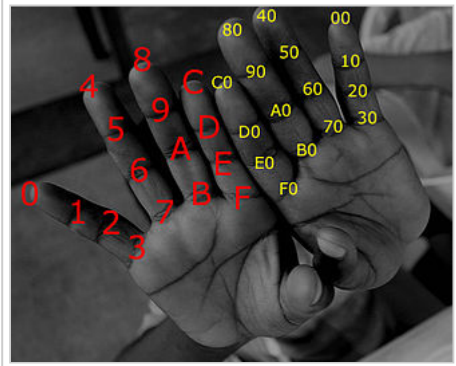
- During the 1950s, some installations favored using the digits 0 through 5 with a macron to denote the values 10–15 as $\bar{0}$, $\bar{1}$, $\bar{2}$, $\bar{3}$, $\bar{4}$ and $\bar{5}$.
- Bendix G-15 computers used the letters *U* through *Z*.
- The Librascope LGP-30 used the letters *F*, *G*, *J*, *K*, *Q* and *W*.[12]
- The ILLIAC I computer used the letters *K*, *S*, *N*, *J*, *F* and *L*.[13]
- Bruce Alan Martin of Brookhaven National Laboratory considered the choice of A–F "ridiculous" and in a 1968 letter to the editor of the CACM proposed an entirely new set of symbols based on the bit locations, which did not gain much, if any, acceptance.[11]
- Soviet programmable calculators Б3-34 and similar used the symbols "−", "L", "C", "Г", "E", " " (space) on their displays.
- Seven-segment display decoder chips used various schemes for outputting values above nine:
    - The Texas Instruments 7446/7447/7448/7449 and 74246/74247/74248/74249 use truncated versions of "2", "3", "4", "5" and "6" for digits A–E. Digit F (1111 binary) was blank.[14]

- The National Semiconductor MM74C912 displayed "o" for A and B, "−" for C, D and E, and blank for F.
- The CD4511 just displays blanks.

Verbal and digital representations

There are no traditional numerals to represent the quantities from ten to fifteen — letters are used as a substitute — and most European languages lack non-decimal names for the numerals above ten. Even though English has names for several non-decimal powers (*pair* for the first binary power, *score* for the first vigesimal power, *dozen*, *gross* and *great gross* for the first three duodecimal powers), no English name describes the hexadecimal powers (decimal 16, 256, 4096, 65536, ...). 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.

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. Another system for counting up to FF_{16} (255_{10}) is illustrated on the right.



Hexadecimal finger-counting scheme.

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. 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.

Usually the number is normalised so that the leading hexadecimal digit is 1 (unless the value is exactly 0).

Example: 1.3DEp42 represents $1.3DE_{16} \times 2^{42}$.

This notation can be used for floating-point literals in the C99 edition of the C programming language.[15] Using the *%a* or *%A* conversion specifiers, this notation can be produced by implementations of the *printf* family of functions following the C99 specification[16] and Single Unix Specification (IEEE Std 1003.1) POSIX standard.[17] Hexadecimal exponential notation is required by the IEEE 754-2008 binary floating-point standard.

Conversion

Binary conversion

Most computers manipulate binary data, but it is difficult for humans to work with the 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.

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

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

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

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

The conversion from hexadecimal to binary is equally direct.

The octal system can also be useful as a tool for people who need to deal directly with binary computer data. Octal represents data as three bits per character, rather than 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 $h_i h_{i-1} \ldots h_2 h_1$ 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;
  var result;
  if (d - r == 0)
    result = toChar(r);
  else
    result = toHex( (d - r)/16 ) + toChar(r);
  return result;
}

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

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 and then performing multiplication and addition to get the final representation. That is, 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, where p is the corresponding hex digit position, counting from right to left, beginning with 0. In this case we have $B3AD = (11 \times 16^3) + (3 \times 16^2) + (10 \times 16^1) + (13 \times 16^0)$, which is 45997 base 10.

x	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

Tools for conversion

Most modern computer systems with graphical user interfaces provide a built-in calculator utility, capable of performing conversions between various radices, in general 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.

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_{hex}) has only a single prime factor (two):

1/2 =	**0.8**
1/3 =	$0.\overline{5}$
1/4 =	**0.4**
1/5 =	$0.\overline{3}$
1/6 =	$0.2\overline{A}$
1/7 =	$0.\overline{249}$
1/8 =	**0.2**
1/9 =	$0.\overline{1C7}$
1/A =	$0.1\overline{9}$
1/B =	$0.\overline{1745D}$
1/C =	$0.1\overline{5}$
1/D =	$0.\overline{13B}$
1/E =	$0.1\overline{249}$
1/F =	$0.\overline{1}$
1/10 =	**0.1**
1/11 =	$0.\overline{0F}$

where an overline denotes a recurring pattern.

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 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 0.199999999999... in hexadecimal. However, hexadecimal is more efficient than bases 12 and 60 for representing fractions with powers of two in the denominator (e.g., decimal one sixteenth is 0.1 in hexadecimal, 0.09 in duodecimal, 0;3,45 in sexagesimal and 0.0625 in decimal).

n	Decimal Prime factors of base, b = 10: 2, 5; b − 1 = 9: 3; b + 1 = 11: 11			Hexadecimal Prime factors of base, b = 16_{10} = 10: 2; b − 1 = 15_{10} = F: 3, 5; b + 1 = 17_{10} = 11: 11		
	Fraction	**Prime factors**	**Positional representation**	**Positional representation**	**Prime factors**	**Fraction**
2	1/2	2	$\mathbf{0.5}$	$\mathbf{0.8}$	2	1/2
3	1/3	3	$\mathbf{0.}3333... = \mathbf{0.}\overline{3}$	$\mathbf{0.}5555... = \mathbf{0.}\overline{5}$	3	1/3
4	1/4	2	$\mathbf{0.25}$	$\mathbf{0.4}$	2	1/4
5	1/5	5	$\mathbf{0.2}$	$\mathbf{0.}\overline{3}$	5	1/5
6	1/6	2, 3	$\mathbf{0.1}\overline{6}$	$\mathbf{0.2}\overline{A}$	2, 3	1/6
7	1/7	7	$\mathbf{0.}\overline{142857}$	$\mathbf{0.}\overline{249}$	7	1/7
8	1/8	2	$\mathbf{0.125}$	$\mathbf{0.2}$	2	1/8
9	1/9	3	$\mathbf{0.}\overline{1}$	$\mathbf{0.}\overline{1C7}$	3	1/9
10	1/10	2, 5	$\mathbf{0.1}$	$\mathbf{0.1}\overline{9}$	2, 5	1/A
11	1/11	11	$\mathbf{0.}\overline{09}$	$\mathbf{0.}\overline{1745D}$	B	1/B
12	1/12	2, 3	$\mathbf{0.08}\overline{3}$	$\mathbf{0.1}\overline{5}$	2, 3	1/C
13	1/13	13	$\mathbf{0.}\overline{076923}$	$\mathbf{0.}\overline{13B}$	D	1/D
14	1/14	2, 7	$\mathbf{0.0}\overline{714285}$	$\mathbf{0.1}\overline{249}$	2, 7	1/E
15	1/15	3, 5	$\mathbf{0.0}\overline{6}$	$\mathbf{0.}\overline{1}$	3, 5	1/F
16	1/16	2	$\mathbf{0.0625}$	$\mathbf{0.1}$	2	1/10
17	1/17	17	$\mathbf{0.}\overline{0588235294117647}$	$\mathbf{0.}\overline{0F}$	11	1/11
18	1/18	2, 3	$\mathbf{0.0}\overline{5}$	$\mathbf{0.0}\overline{E38}$	2, 3	1/12
19	1/19	19	$\mathbf{0.}\overline{052631578947368421}$	$\mathbf{0.}\overline{0D79435E5}$	13	1/13
20	1/20	2, 5	$\mathbf{0.05}$	$\mathbf{0.0}\overline{C}$	2, 5	1/14
21	1/21	3, 7	$\mathbf{0.}\overline{047619}$	$\mathbf{0.}\overline{0C3}$	3, 7	1/15
22	1/22	2, 11	$\mathbf{0.0}\overline{45}$	$\mathbf{0.0}\overline{BA2E8}$	2, B	1/16
23	1/23	23	$\mathbf{0.}\overline{0434782608695652173913}$	$\mathbf{0.}\overline{0B21642C859}$	17	1/17
24	1/24	2, 3	$\mathbf{0.041}\overline{6}$	$\mathbf{0.0}\overline{A}$	2, 3	1/18
25	1/25	5	$\mathbf{0.04}$	$\mathbf{0.}\overline{0A3D7}$	5	1/19
26	1/26	2, 13	$\mathbf{0.0}\overline{384615}$	$\mathbf{0.0}\overline{9D8}$	2, D	1/1A
27	1/27	3	$\mathbf{0.}\overline{037}$	$\mathbf{0.}\overline{097B425ED}$	3	1/1B
28	1/28	2, 7	$\mathbf{0.03}\overline{571428}$	$\mathbf{0.0}\overline{924}$	2, 7	1/1C
29	1/29	29	$\mathbf{0.}\overline{0344827586206896551724137931}$	$\mathbf{0.}\overline{08D3DCB}$	1D	1/1D
30	1/30	2, 3, 5	$\mathbf{0.0}\overline{3}$	$\mathbf{0.0}\overline{8}$	2, 3, 5	1/1E
31	1/31	31	$\mathbf{0.}\overline{032258064516129}$	$\mathbf{0.}\overline{08421}$	1F	1/1F
32	1/32	2	$\mathbf{0.03125}$	$\mathbf{0.08}$	2	1/20
33	1/33	3, 11	$\mathbf{0.}\overline{03}$	$\mathbf{0.}\overline{07C1F}$	3, B	1/21
34	1/34	2, 17	$\mathbf{0.0}\overline{2941176470588235}$	$\mathbf{0.0}\overline{78}$	2, 11	1/22
35	1/35	5, 7	$\mathbf{0.0}\overline{285714}$	$\mathbf{0.}\overline{075}$	5, 7	1/23

36	1/36	2, 3	$\mathbf{0.02}\overline{7}$	$\mathbf{0.0}\overline{71C}$	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...
φ (phi, the golden ratio = $(1+\sqrt{5})/2$)	1.618 033 988 749 894 848...	1.9E3779B97F4A...
π (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...
τ (the Thue–Morse constant)	0.412 454 033 640 107 597...	0.6996 9669 9669 6996...
γ (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

Etymology

The word *hexadecimal* is composed of *hexa-*, derived from the Greek ἑξ (hex) for *six*, and *-decimal*, derived from the Latin for *tenth*. Webster's Third New International online derives *hexadecimal* as an alteration of the all-Latin *sexadecimal* (which appears in the earlier Bendix documentation). The earliest date attested for *hexadecimal* in Merriam-Webster Collegiate online is 1954, placing it safely in the category of international scientific vocabulary (ISV). It is common in ISV to mix Greek and Latin combining forms freely. The word *sexagesimal* (for base 60) retains the Latin prefix. Donald Knuth has pointed out that the etymologically correct term is *senidenary* (or possibly, *sedenary*), from the Latin term for *grouped by 16*. (The terms *binary*, *ternary* and *quaternary* are from the same Latin construction, and the etymologically correct terms for *decimal* and *octal* arithmetic are *denary* and *octonary*, respectively.)[18] 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".[19][20] Schwartzman notes that the expected form from usual Latin phrasing would be *sexadecimal*, but computer hackers would be tempted to shorten that word to *sex*.[21] The etymologically proper Greek term would be *hexadecadic* / *εξαδεκαδικός* / *exadekadikos* (although in Modern Greek, *decahexadic* / *δεκαεξαδικός* / *dekaexadikos* is more commonly used).

Use in Chinese culture

The traditional Chinese units of weight were base-16. For example, one jīn (斤) in the old system equals sixteen taels. The suanpan (Chinese abacus) could be used to perform hexadecimal calculations.

Primary numeral system

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.[22] Some proposals unify standard measures so that they are multiples of 16.[23][24][25]

An example of unified standard measures is hexadecimal time, which subdivides a day by 16 so that there are 16 "hexhours" in a day.[25]

Key to number base notation

Simple key for notations used in article:

Name	Abbreviation	Number base
Binary	bin	2
Octal	oct	8
Decimal	dec	10
Hexadecimal	hex	16

Transfer encoding

Base16 or hex (not to be confused with Intel HEX and the like) is one of the simplest binary-to-text encodings, which stores each byte as a pair of hexadecimal digits. Many variations of such format are possible, for example either uppercase (A-F) or lowercase (a-f) letters may be used for digits greater than 9; spaces, line breaks or other separators may be added between digit groups of different lengths; header and/or footer with metainformation may be added.

See also

- Base32, Base64 (content encoding schemes)
- Hex editor
- Hex dump

References

1. "PDF The Unicode Standard, Version 7" (PDF).
2. "Hexadecimal web colors explained".
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. "Modula-2 - Vocabulary and representation". *Modula -2*. Retrieved 1 November 2015.
5. The VHDL MINI-REFERENCE: VHDL IDENTIFIERS, NUMBERS, STRINGS, AND EXPRESSIONS (http://www.eng.auburn.edu/department/ee/mgc/vhdl.html#numbers)
6. "*read-base* variable in Common Lisp".
7. "*print-base* variable in Common Lisp".
8. MSX is Coming — Part 2: Inside MSX (http://www.atarimagazines.com/compute/issue56/107_1_MSX_IS_COMING.php) Compute!, issue 56, January 1985, p. 52

9. BBC BASIC programs are not fully portable to Microsoft BASIC (without modification) since the latter takes `&` to prefix octal values. (Microsoft BASIC primarily uses `&O` to prefix octal, and it uses `&H` to prefix hexadecimal, but the ampersand alone yields a default interpretation as an octal prefix.
10. Donald E. Knuth. *The TeXbook* (Computers and Typesetting, Volume A). Reading, Massachusetts: Addison–Wesley, 1984. ISBN 0-201-13448-9. The source code of the book in TeX (http://www.ctan.org/tex-archive/systems/knuth/tex/texbook.tex) (and a required set of macros CTAN.org (ftp://tug.ctan.org/pub/tex-archive/systems/knuth/lib/manmac.tex)) is available online on CTAN.
11. 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.
12. This somewhat odd sequence was from the next six sequential numeric keyboard codes in the LGP-30's 6-bit character code. LGP-30 PROGRAMMING MANUAL (http://ed-thelen.org/comp-hist/lgp-30-man.html#R4.13)
13. "ILLIAC Programming" (PDF). University of Illinois via Bitsavers. pp. 3–2. Retrieved 18 December 2014.
14. *BCD-to-Seven-Segment Decoders/Drivers: SN54246/SN54247/SN54LS247, SN54LS248 SN74246/SN74247/SN74LS247/SN74LS248* (PDF), Texas Instruments, March 1988 [March 1974], SDLS083, archived (PDF) from the original on 2017-03-29, retrieved 2017-03-30, "[…] They can be used interchangeable in present or future designs to offer designers a choice between two indicator fonts. The '46A, '47A, 'LS47, and 'LS48 compose the 6 and the 9 without tails and the '246, '247, 'LS247, and 'LS248 compose the 6 and the 0 with tails. Composition of all other characters, including display patterns for BCD inputs above nine, is identical. […] Display patterns for BCD input counts above 9 are unique symbols to authenticate input conditions. […]"
15. "ISO/IEC 9899:1999 - Programming languages - C". Iso.org. 2011-12-08. Retrieved 2014-04-08.
16. "Rationale for International Standard - Programming Languages - C" (PDF). 5.10. April 2003. pp. 52, 153–154, 159. Archived (PDF) from the original on 2016-06-06. Retrieved 2010-10-17.
17. The IEEE and The Open Group (2013) [2001]. "dprintf, fprintf, printf, snprintf, sprintf - print formatted output". *The Open Group Base Specifications* (Issue 7, IEEE Std 1003.1, 2013 ed.). Archived from the original on 2016-06-21. Retrieved 2016-06-21.
18. Knuth, Donald. (1969). *The Art of Computer Programming, Volume 2*. ISBN 0-201-03802-1. (Chapter 17.)
19. A.B. Taylor, Report on Weights and Measures (https://books.google.com/books?id=X7wLAAAAYAAJ&pg=PP5), Pharmaceutical Association, 8th Annual Session, Boston, Sept. 15, 1859. See pages and 33 and 41.
20. 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), Philadelphia, 1887; pages 296-366. See pages 317 and 322.
21. Schwartzman, S. (1994). *The Words of Mathematics: an etymological dictionary of mathematical terms used in English*. ISBN 0-88385-511-9.
22. "Base 4^2 Hexadecimal Symbol Proposal".
23. "Intuitor Hex Headquarters".
24. "A proposal for addition of the six Hexadecimal digits (A-F) to Unicode".
25. 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*. Philadelphia.