Hexadecimal Numbers

Craig Stuart Sapp
craig@ccrma.stanford.edu

Representation of Numbers

24

Representation of Numbers

24

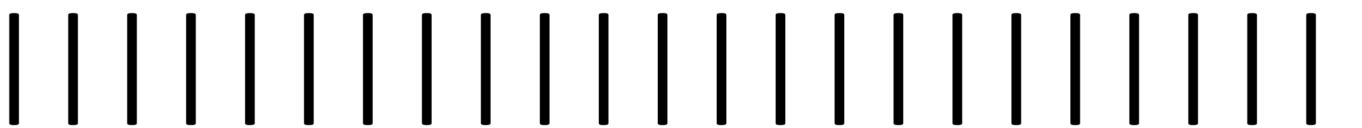
Direct representation:

| | | | | | | | | | | | | | | | | | | | | | | |

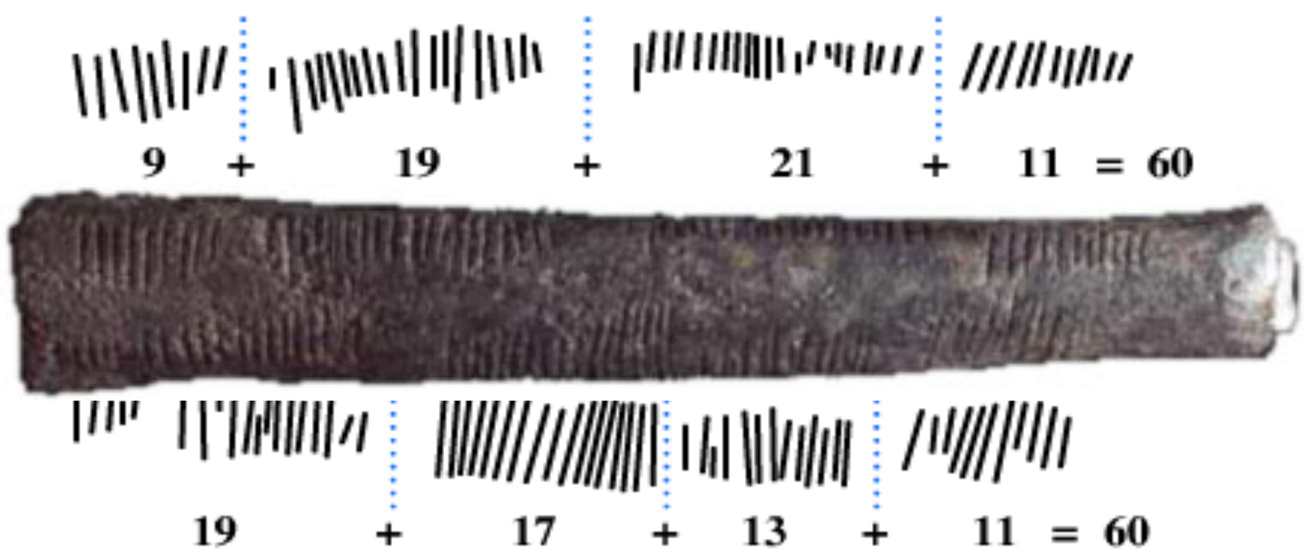
Representation of Numbers

24

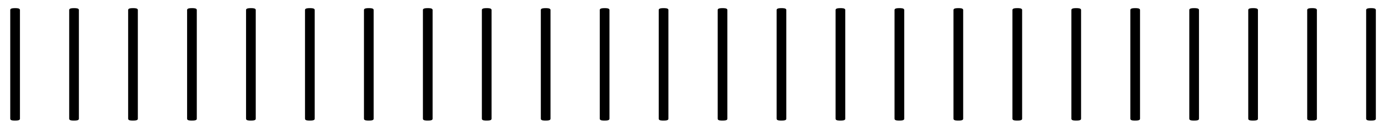
Direct representation:



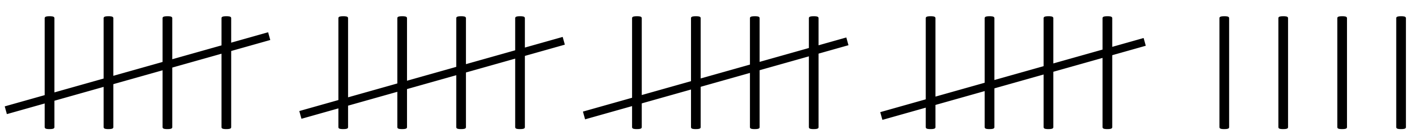
Ishango bone ~20,000 years old www.wikipedia.org/wiki/Ishango_bone



Tally Marks



Easier to read:

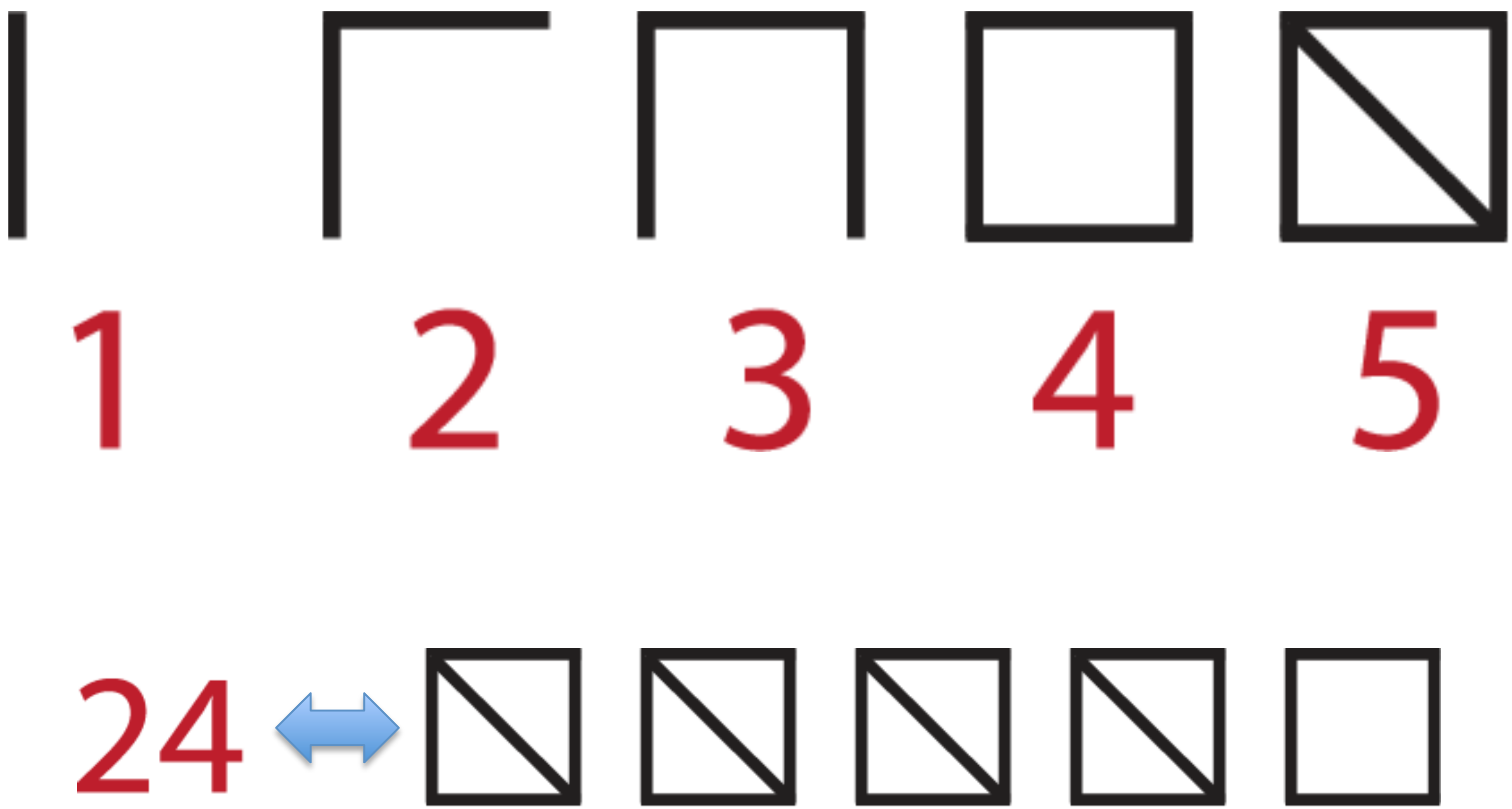


Northern European/American tally marks

Tally Marks (2)

Spanish/French/Latin American tally marks

Tally Marks (2)



Spanish/French/Latin American tally marks

Tally Marks (3)

Chinese/Japanese/Korean tally marks

Tally Marks (3)

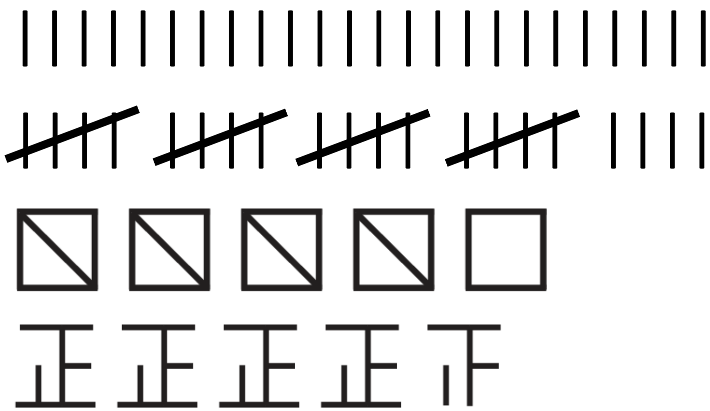
1 2 3 4 5

24

Chinese/Japanese/Korean tally marks

Equivalent Representations

24



Positional Notation

- Tally mark groupings make direct representation of numbers easier to read (>5).

Positional Notation

- Tally mark groupings make direct representation of numbers easier to read (>5).
- But fails to convey or be convenient for very large numbers:

Positional Notation

- Tally mark groupings make direct representation of numbers easier to read (>5).
- But fails to convey or be convenient for very large numbers:

- Therefore, positional notation is a more useful method of representing larger numbers:

365

Positional Notation (2)

- What does “365” mean?

Positional Notation (2)

- What does "365" mean?

365 = 300   + 60   + 5

Positional Notation (2)

- What does "365" mean?

$365 = 300 + 60 + 5$

$= 3 \times 100 + 6 \times 10 + 5 \times 1$

Positional Notation (2)

- What does "365" mean?

$$365 = 300 + 60 + 5$$

$$= 3 \times 100 + 6 \times 10 + 5 \times 1$$

$$= 3 \times 10^{2} + 6 \times 10^{1} + 5 \times 10^{0}$$

Positional Notation (2)

- What does “365” mean?

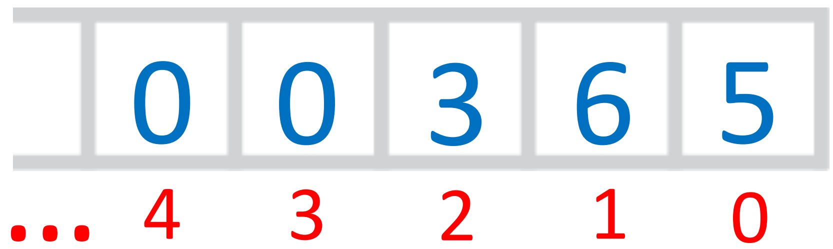
$$365 = 300 + 60 + 5$$

$$= 3 \times 100 + 6 \times 10 + 5 \times 1$$

$$= 3 \times 10^{2} + 6 \times 10^{1} + 5 \times 10^{0}$$

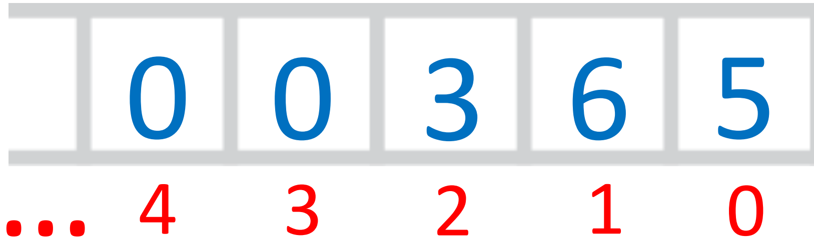
0	0	3	6	5
4	3	2	1	0

Positional Number Bases



- What do these numbers mean?

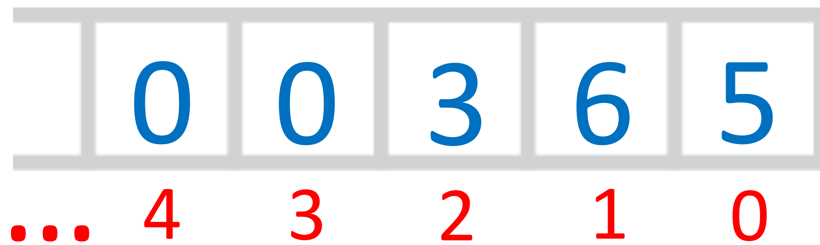
Positional Number Bases



Base-10: $3 \times 10^{2} + 6 \times 10^{1} + 5 \times 10^{0} = 365_{10}$

- What do these numbers mean?
- Powers of 10 are implicit in decimal numbers

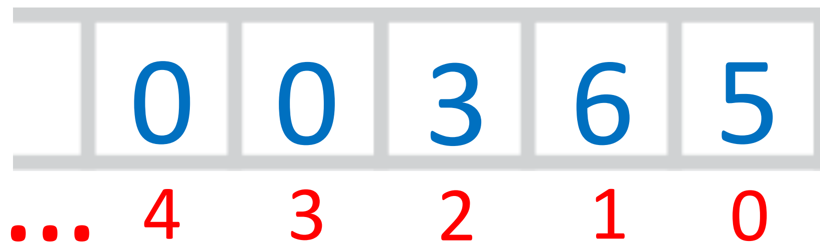
Positional Number Bases



Base-10: $3 x 10^2 + 6 x 10^1 + 5 x 10^0 = 365_{10}$

Base-9: $3 x 9^2 + 6 x 9^1 + 5 x 9^0 = 302_{10}$

Positional Number Bases



Base-10: $3 \times 10^2 + 6 \times 10^1 + 5 \times 10^0 = 365_{10}$

Base-9: $3 \times 9^2 + 6 \times 9^1 + 5 \times 9^0 = 302_{10}$

$$365_9 = 302_{10}$$

Positional Number Bases

$$365_{10} = ???_{9}$$

Positional Number Bases

$$365_{10} = ???_9$$

- Position “2”: $365 / 9^2 =$ **4**

  remainder: $365 - 4 \times 9^2 = 41$

Positional Number Bases

$$365_{10} = ???_9$$

- Position "2": $365 / 9^2 = \mathbf{4}$

  remainder: $365 - 4 \times 9^2 = 41$

- Position "1": $41 / 9^1 = \mathbf{4}$

  remainder: $41 - 4 \times 9^1 = 5$

Positional Number Bases

$$365_{10} = ???_9$$

- Position "2": $365 / 9^2 = \mathbf{4}$

  remainder: $365 - 4 \times 9^2 = 41$

- Position "1": $41 / 9^1 = \mathbf{4}$

  remainder: $41 - 4 \times 9^1 = 5$

- Position "0": $5 / 9^0 = \mathbf{5}$

Positional Number Bases

$$365_{10} = ???_9$$

- Position "2": $365 / 9^2 = \mathbf{4}$

  remainder: $365 - 4 \times 9^2 = 41$

- Position "1": $41 / 9^1 = \mathbf{4}$

  remainder: $41 - 4 \times 9^1 = 5$

- Position "0": $5 / 9^0 = \mathbf{5}$

$$365_{10} = 445_9$$

Common Bases

- **Base-10**: what you learn in school for representing most numbers.

Common Bases

- **Base-10**: what you learn in school for representing most numbers.

- **Base-60**: Minutes/Seconds

  12’30” + 16’42” = 29’12”

“Sexagesimal”

- Developed ~5000 years ago by Sumerians.

Common Bases

- **Base-10**: what you learn in school for representing most numbers.

- **Base-60**: Minutes/Seconds

  12’30” + 16’42” = 29’12”

- **Base-1**: Tally marks

$$|\,|\,|\,| = 1 \times 1^3 + 1 \times 1^2 + 1 \times 1^1 + 1 \times 1^0 = 4$$

Common Bases

- **Base-10**: what you learn in school for representing most numbers.
- **Base-60**: Minutes/Seconds

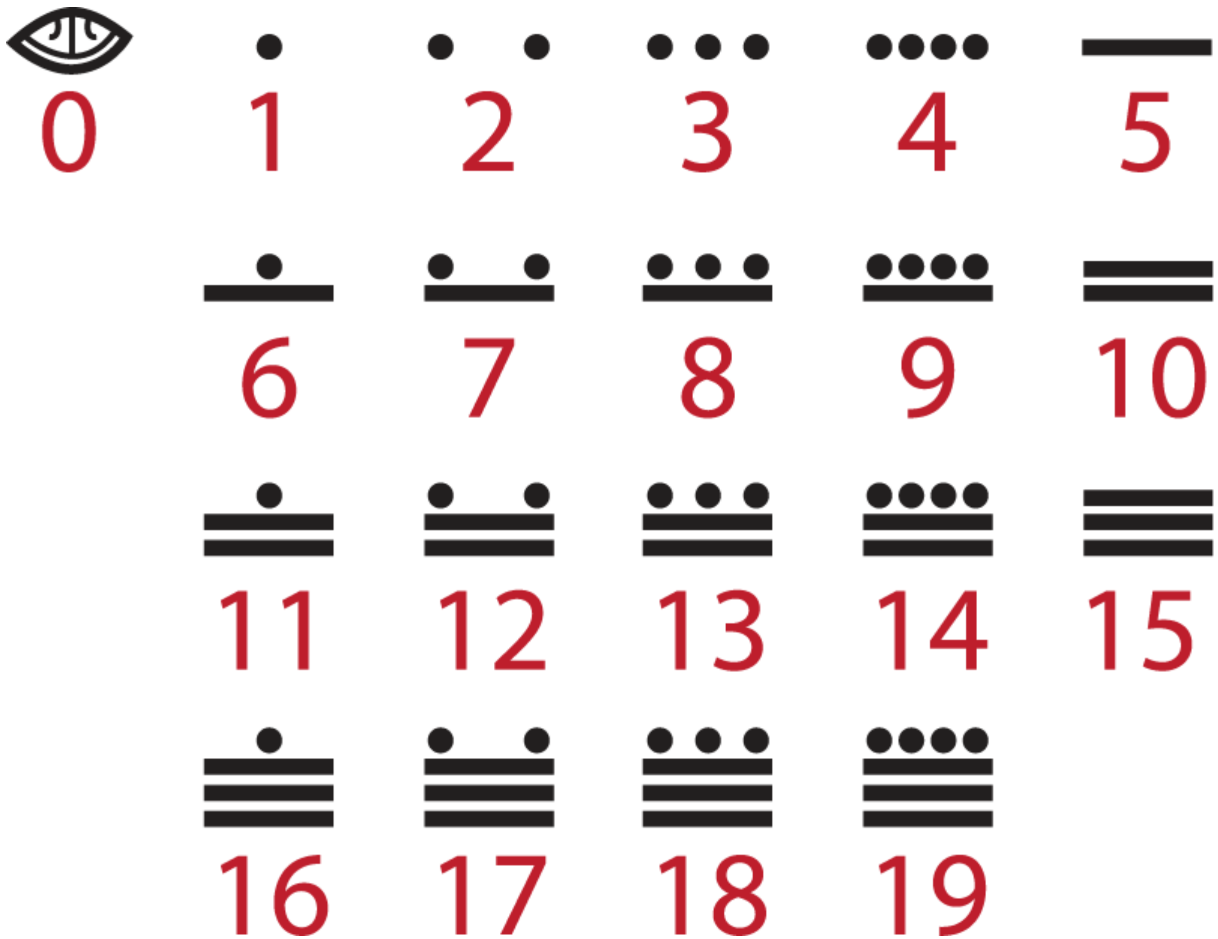
12’30” + 16’42” = 29’12”

- **Base-1**: Tally marks

$$|\,|\,|\,| = 1 \times 1^3 + 1 \times 1^2 + 1 \times 1^1 + 1 \times 1^0 = 4$$

http://en.wikipedia.org/wiki/Positional_notation

Mayan Numbers (Base-20)



Mayan Numbers (2)

http://www.mayacalendar.com/mayacalendar/f-mayamath.html

Mayan Numbers (2)

$3 \text{ x } 20^{2} = 1200$

http://www.mayacalendar.com/mayacalendar/f-mayamath.html

Mayan Numbers (2)

$3 \times 20^2 = 1200$

$0 \times 20^1 = 0$

http://www.mayacalendar.com/mayacalendar/f-mayamath.html

Mayan Numbers (2)

$$3 \times 20^{2} = 1200$$

$$0 \times 20^{1} = 0$$

$$18 \times 20^{0} = 18$$

http://www.mayacalendar.com/mayacalendar/f-mayamath.html

Mayan Numbers (2)

$3 \times 20^{2} = 1200$

$0 \times 20^{1} = 0$

$18 \times 20^{0} = 18$

1218

http://www.mayacalendar.com/mayacalendar/f-mayamath.html

Mayan Numbers (3)

Long Count: Calendar by number of days

		digit base	
“quad-century”	Baktun	20	1 baktun = 400 tun = 20 * 20 * 18 * 20 kin = 144000 kin
“bi-decade”	Katun	20	1 katun = 20 tun = 20 * 18 * 20 kin = 7200 kin
“year”	Tun	20	1 tun = 18 uinal 1 tun = 18 * 20 kin = 360 kin
“month”	Uinal	18	1 unial = 20 kin
day	Kin	20	

Mayan Numbers (3)

24 Feburary 2013 → **13.0.0.1.14** Long Count date.

- What is the Long Count date for the day before Baktun 13 starts?

- Baktun 13 started on 21 December 2013. What (proleptic) Gregorian calendar day did the first *kin* occur?

Mayan Numbers (3)

24 Feburary 2013 → 13.0.0.1.14 Long Count date.

- What is the Long Count date for the day before Baktun 13 starts?

20 December 2012 → 12.19.17.19.19 LC

- Baktun 13 started on 21 December 2013. What (proleptic) Gregorian calendar day did the first *kin* occur?

11 August 3114 BC

About the same time as Stonehenge I

British Money before 1971

(positional notation with non-constant base)

After 15 February 1971:
1 pound = 100 (new) pence

Before 15 February 1971:
1 pound = 20 shillings
1 shilling = 12 pence
(1 pound = 240 pence)

£4-12-7 → 4 pounds, 12 shillings and 7 pence

12/7 → 12 shillings and 7 pence

12/- → 12 shillings

12s 7d → 12 shillings (solidi) and 7 pence (denari)

Binary Numbers

Base-2: Internal representation of numbers
on a computer (and most data storage).

Binary Numbers

Base-2: Internal representation of numbers on a computer (and most data storage).

Two numerals in base-2:

- 0 "off"
- 1 "on"

Binary Numbers

Base-2: Internal representation of numbers on a computer (and most data storage).

Two numerals in base-2:

- 0 "off"
- 1 "on"

$$11000_2 = 1x2^4 + 1x2^3 = 16+8 = 24_{10}$$

Motivation for Hexadecimal

Base-2: similar to Base-1 (tally marks)

- Difficult to read large numbers:

$$101101101_2 = 365_{10}$$

Motivation for Hexadecimal

Base-2: similar to Base-1 (tally marks)

- Difficult to read large numbers:

$$101101101_2 = 365_{10}$$

Base-16: more convenient way of working with binary numbers.

- Every four base-2 digits represent one base-16 digit
- Trivial to convert between base-2 and base-16:

$$1{,}0110{,}1101_2 = 16D_{16}$$

Hexadecimal Digits > 9

- Convenient to have one character representing each position.
  - $A = 10_{10}$
  - $B = 11_{10}$
  - $C = 12_{10}$
  - $D = 13_{10}$
  - $E = 14_{10}$
  - $F = 15_{10}$

Hexadecimal Digits > 9

- Convenient to have one character representing each position.
  - A = 10_{10}
  - B = 11_{10}
  - C = 12_{10}
  - D = 13_{10}
  - E = 14_{10}
  - F = 15_{10}

0-9 + A-Z can be used to represent base-36 positional numbers

Base-16 (Hexadecimal)

$$365_{10} = 1x16^{2} + 6x16^{1} + 13x16^{0}$$

Base-16 (Hexadecimal)

$$365_{10} = 1x16^{2} + 6x16^{1} + 13x16^{0}$$

D

Base-16 (Hexadecimal)

$$365_{10} = 1\text{x}16^{2} + 6\text{x}16^{1} + 13\text{x}16^{0}$$

D

$$365_{10} = 16D_{16}$$

Base-16 (Hexadecimal)

$$365_{10} = 1 \text{x} 16^2 + 6 \text{x} 16^1 + 13 \text{x} 16^0$$

D

$$365_{10} = 16D_{16}$$

$$= 16Dh$$

Base-16 (Hexadecimal)

$$365_{10} = 1 \text{x} 16^{2} + 6 \text{x} 16^{1} + 13 \text{x} 16^{0}$$

D

$$365_{10} = 16D_{16}$$

$$= 16Dh$$

$$= 0x16D$$

Hexadecimal to Binary Conversion

$16D_{16}$

Hexadecimal to Binary Conversion

$16D_{16}$

- convert each hex digit to 4 binary digits

0001_2 0110_2 1101_2

Hexadecimal to Binary Conversion

$16D_{16}$

0001_2 0110_2 1101_2

- convert each hex digit to 4 binary digits

000101101101_2

- concatenate binary digits together

Hexadecimal to Binary Conversion

$$16D_{16}$$

- convert each hex digit to 4 binary digits

$$0001_2 \quad 0110_2 \quad 1101_2$$

- concatenate binary digits together

$$000101101101_2$$

$$1x2^8+1x2^6+1x2^5+1x2^3+1x2^2+1x2^0 = 365_{10}$$

Binary to Hexadecimal Conversion

- $0_{16} = 0000_2 = 0_{10}$
- $1_{16} = 0001_2 = 1_{10}$
- $2_{16} = 0010_2 = 2_{10}$
- $3_{16} = 0011_2 = 3_{10}$
- $4_{16} = 0100_2 = 4_{10}$
- $5_{16} = 0101_2 = 5_{10}$
- $6_{16} = 0110_2 = 6_{10}$
- $7_{16} = 0111_2 = 7_{10}$
- $8_{16} = 1000_2 = 8_{10}$
- $9_{16} = 1001_2 = 9_{10}$
- $A_{16} = 1010_2 = 10_{10}$
- $B_{16} = 1011_2 = 11_{10}$
- $C_{16} = 1100_2 = 12_{10}$
- $D_{16} = 1101_2 = 13_{10}$
- $E_{16} = 1110_2 = 14_{10}$
- $F_{16} = 1111_2 = 15_{10}$

Binary to Hexadecimal Conversion

- $0_{16} = 0000_2 = 0_{10}$
- $1_{16} = 0001_2 = 1_{10}$
- $2_{16} = 0010_2 = 2_{10}$
- $3_{16} = 0011_2 = 3_{10}$
- $4_{16} = 0100_2 = 4_{10}$
- $5_{16} = 0101_2 = 5_{10}$
- $6_{16} = 0110_2 = 6_{10}$
- $7_{16} = 0111_2 = 7_{10}$
- $8_{16} = 1000_2 = 8_{10}$
- $9_{16} = 1001_2 = 9_{10}$
- $A_{16} = 1010_2 = 10_{10}$
- $B_{16} = 1011_2 = 11_{10}$
- $C_{16} = 1100_2 = 12_{10}$
- $D_{16} = 1101_2 = 13_{10}$
- $E_{16} = 1110_2 = 14_{10}$
- $F_{16} = 1111_2 = 15_{10}$

$$0001{,}0110{,}1101_2 = 16D_{16}$$

Endian

BE = Big Endian (bigger digits first)

LE = Little Endian (smaller digits first)

$$24_{10,be} = 42_{10,le}$$

$$2+10^1 + 4x10^0 = 4x10^0 + 2x10^1$$

Endian

BE = Big Endian (bigger digits first)

LE = Little Endian (smaller digits first)

$$24_{10,be} = 42_{10,le}$$

$$2+10^1 + 4x10^0 = 4x10^0 + 2x10^1$$

Sing a song of sixpence,
A pocket full of rye.
Four and twenty blackbirds,
Baked in a pie.

http://en.wikipedia.org/wiki/Sing_a_Song_of_Sixpence

Endian (2)

English: Three hundred twenty-four

→

324

Endian (2)

English: Three hundred twenty-four

German: Drei hundert vier und zwanzig

“three hundred four and twenty”

Arabic: Thalaath mi’a arba’ wa ’ishriin

“three hundred four and twenty”

Endian (3)

(Byte Order)

$54{,}321_{10} \quad = \quad D4{,}31_{16,BE} \quad = \quad 31{,}D4_{16,LE}$

- Computers handle bytes in big-endian or little-endian orderings.
- Intel x86 CPU architectures are little-endian.
- Motorola 6800 CPU is big-endian.
- ARM is bi-endian (can do both ways).

http://en.wikipedia.org/wiki/Endianness