# Number Theory: Representations of Numbers

*Written by Rachel Singh, last updated Oct 25, 2018*

# About

If a computer only understand binary, how does it store data? (text, files, etc.)

In this part we will talk about numeric representation.

# Topics

1. Number Systems

2. Convertsion between different bases

3. Text to Binary

# Number Systems

# 1. Number Systems

**Definition:** Given a positive integer X, the ***decimal representation*** for X is a string consisting of digits from {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} that looks like $d_n d_{(n-1)} \ldots d_2 d_1 d_0$ , where

$$X = \sum_{i=0}^{n} d_i \cdot 10^i$$

$$= d_n \cdot 10^n + d_{(n-1)} + 10^{(n-1)} + \ldots + d_2 \cdot 10^2 + d_1 \cdot 10^1 + d_0 \cdot 10^0$$

From Discrete Mathematics, Ensley & Crawley

Notes

# 1. Number Systems

Or, think of $= d_n \cdot 10^n + d_{(n-1)} + 10^{(n-1)} + \ldots + d_2 \cdot 10^2 + d_1 \cdot 10^1 + d_0 \cdot 10^0$

As being the different "places" of a digit – 100s place, 10s place, 1s place, etc.

| 5 | 3 | 2 | 1 |
|---|---|---|---|
| Thousands place | Hundreds place | Tens place | Ones place |
| $10^3$ | $10^2$ | $10^1$ | $10^0$ |

*Is essentially 5x1000 + 3x100 + 2x10 + 1x1*

Notes

# 1. Number Systems

**Definition:** The base-two (binary) representation of a positive integer X is a string consisting of the digits from {0, 1} that looks like $b_n b_{(n-1)} \ldots b_2 b_1 b_0$ where

$$X = \sum_{i=0}^{n} b_i \cdot 2^i$$

$$= d_b \cdot 2^n + b_{(n-1)} + 2^{(n-1)} + \ldots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

From Discrete Mathematics, Ensley & Crawley

Notes

# 1. Number Systems

So for binary numbers, the break down is:

| 1 | 0 | 1 | 1 |
|---|---|---|---|
| Eights place | Fours place | Twos place | Ones place |

*And this is 1x8 + 0x4 + 1x2 + 1x1 = 8 + 2 + 1 = 11*

$2^3$ $2^2$ $2^1$ $2^0$

Notes

# 1. Number Systems

This applies for any number system.

- If you have base 2, then the digits available are {0, 1} and the places are the $2^0$, $2^1$, $2^2$, etc. places.
- If you have base 3, then the digits available are {0, 1, 2} and the places are the $3^0$, $3^1$, $3^2$, etc. places.
- If you have base 4, then the digits available are {0, 1, 2, 3} and the places are the $4^0$, $4^1$, $4^2$, etc. places.
- And so on…

Notes

# 1. Number Systems

The hexadecimal number system (base 16) goes from 0 to 15, HOWEVER…

Because Hex is used in computers a lot, we wanted a way to write 10 through 15 with only one character, so we use A – F for this.

Notes

# 1. Number Systems

| Base 10 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Base 16 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

| Base 10 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|
| Base 16 | 8 | 9 | A | B | C | D | E | F |

Notes

# Conversion between Different Bases

# 2. Conversion between Different Bases

## Any base → Decimal

To convert any base *b* to base-10, expand out the number into the equation and compute…

$(123)_4$ 123 base 4

| $4^2$ place | $4^1$ place | $4^0$ place |
|---|---|---|
| 1 | 2 | 3 |

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

$= 16 + 8 + 3$ **$= 27$**

Notes

**Convert base *b* to 10:** Expand number by its places and do the math.

# 2. Conversion between Different Bases

## Binary ↔ Hexadecimal

To convert between binary and hexadecimal, take 4 bits at a time and convert the bits directly to a hex number.

| Decimal | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Binary | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
| Hex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | | | | | | | | |
| Decimal | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Binary | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| Hex | 8 | 9 | A | B | C | D | E | F |

Notes

**Convert base *b* to 10:** Expand number by its places and do the math.

# 2. Conversion between Different Bases

## Decimal → Binary

To convert from base-10 to base-2, follow this algorithm:

1) Input: Some natural number $n$ (base 10)
2) While $n > 0$, do:
   1) Divide $n$ by $b$ and get some quotient $q$ and remainder $r$.
   2) Write $r$ as the next (right-to-left) digit in the binary string.
   3) Replcae the value of $n$ with $q$, and repeat the loop.

Notes

**Convert base *b* to 10:** Expand number by its places and do the math.

# 2. Conversion between Different Bases

**Example:** Convert $(15)_{10}$ to binary (b=2) using the algorithm…

1) Input: Some natural number *n* (base 10)
2) While *n* > *0*, do:
   1) Divide *n* by *b* and get some quotient *q* and remainder *r*.
   2) Write *r* as the next (right-to-left) digit in the binary string.
   3) Replcae the value of *n* with *q*, and repeat the loop.

**n = 15**

```
2 ⟌ 15
   -14
   ----
     1
```

## Notes

**Convert base *b* to 10:** Expand number by its places and do the math.

# 2. Conversion between Different Bases

**Example:** Convert $(15)_{10}$ to binary (b=2) using the algorithm…

1) Input: Some natural number *n* (base 10)
2) While *n* > *0*, do:
   1) Divide *n* by *b* and get some quotient *q* and remainder *r*.
   2) Write *r* as the next (right-to-left) digit in the binary string.
   3) Replcae the value of *n* with *q*, and repeat the loop.

**n = 15**

```
      7 r 1
   ┌──────
2  │ 15
   │-14
   │----
   │  1
```

**q = 7** **r = 1**

**new n = 7**

**Binary:**
**1**

## Notes

**Convert base *b* to 10:** Expand number by its places and do the math.

# 2. Conversion between Different Bases

**Example:** Convert $(15)_{10}$ to binary (b=2) using the algorithm…

1) Input: Some natural number *n* (base 10)
2) While *n* > *0*, do:
   1) Divide *n* by *b* and get some quotient *q* and remainder *r*.
   2) Write *r* as the next (right-to-left) digit in the binary string.
   3) Replcae the value of *n* with *q*, and repeat the loop.

**n = 7**

```
      3 r 1
   ┌──
2  │ 7
   - 6
   ----
     1
```

**q = 3** **r = 1**

**new n = 3**

**Binary: 11**

## Notes

**Convert base *b* to 10:** Expand number by its places and do the math.

# 2. Conversion between Different Bases

**Example:** Convert $(15)_{10}$ to binary (b=2) using the algorithm…

1) Input: Some natural number *n* (base 10)
2) While *n* > *0*, do:
   1) Divide *n* by *b* and get some quotient *q* and remainder *r*.
   2) Write *r* as the next (right-to-left) digit in the binary string.
   3) Replcae the value of *n* with *q*, and repeat the loop.

**n = 3**

```
        1 r 1
    ┌──────
2   │  3
    -  2
    ----
       1
```

**q = 1** **r = 1**

**new n = 1**

**Binary: 111**

## Notes

**Convert base *b* to 10:** Expand number by its places and do the math.

# 2. Conversion between Different Bases

**Example:** Convert $(15)_{10}$ to binary (b=2) using the algorithm…

1) Input: Some natural number *n* (base 10)
2) While *n > 0*, do:
   1) Divide *n* by *b* and get some quotient *q* and remainder *r*.
   2) Write *r* as the next (right-to-left) digit in the binary string.
   3) Replcae the value of *n* with *q*, and repeat the loop.

**n = 1**

```
        0 r 1
    ┌──────
2   │  1
    │- 2
    │----
      1
```

**q = 0** **r = 1**

**new n = 0**
**(End loop)**

**Binary:**
**1111**

## Notes

**Convert base *b* to 10:** Expand number by its places and do the math.

# Text to Binary

# 3. Text to Binary

A computer stores each symbol or character it will use as a number code. For example, early computers used ASCII, and the symbol table looked like this:

## ASCII TABLE

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | [NULL] | 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | [START OF HEADING] | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | [START OF TEXT] | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | [END OF TEXT] | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | [END OF TRANSMISSION] | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | [ENQUIRY] | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | [ACKNOWLEDGE] | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | [BELL] | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | [BACKSPACE] | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | [HORIZONTAL TAB] | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | [LINE FEED] | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | [VERTICAL TAB] | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | [FORM FEED] | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | [CARRIAGE RETURN] | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | [SHIFT OUT] | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | [SHIFT IN] | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | [DATA LINK ESCAPE] | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | [DEVICE CONTROL 1] | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | [DEVICE CONTROL 2] | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | [DEVICE CONTROL 3] | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | [DEVICE CONTROL 4] | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | [NEGATIVE ACKNOWLEDGE] | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | [SYNCHRONOUS IDLE] | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | [ENG OF TRANS. BLOCK] | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | [CANCEL] | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | [END OF MEDIUM] | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | [SUBSTITUTE] | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | [ESCAPE] | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | [FILE SEPARATOR] | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | [GROUP SEPARATOR] | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | [RECORD SEPARATOR] | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | [UNIT SEPARATOR] | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

From https://simple.wikipedia.org/wiki/Main_Page

# 3. Text to Binary

## Capital letter symbols

| A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |

## Lower-case letter symbols

| a | b | c | d | e | f | g | h | i | j | k | l | m |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 |
| n | o | p | q | r | s | t | u | v | w | x | y | z |
| 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 |

Notes

# 3. Text to Binary

So to convert some text to binary, you need to find the code equivalent for each letter, and convert each letter from base-10 to base-2.

| C | A | T | S |
|---|---|---|---|
| 67 | 65 | 84 | 83 |
| 0100 0011 | 0100 0001 | 0101 0100 | 0101 0011 |

Notes

# Conclusion

Yay.