Computers' arthmetics

wer. 10 z drobnymi modyfikacjami!

Wojciech Myszka

2023-10-16 18:05:52 +0200

Wrocław University
of Science and Technology

Binary numeral system

The binary numeral systems is rather old:

- India — 5–2 BC
- in the 11th century in China, arranging the I Ching (Yijing) sets of hexagrams with, yin as 0, yang as 1 from 0 to 63. *I Ching* also known as *Classic of Changes* or *Book of Changes* is one of the oldest of the Chinese classic texts. The book contains divination system, and is still used for this purpose. The text is now an important part of the Chinese culture.
- Gottfried Leibniz describe it in 1679. See, for example, `http://books.google.pl/books?id=Fuk8AAAAcAAJ&printsec=frontcover#v=onepage&q&f=false` (in French!)

I Ching hexagrams

I Ching hexagrams															
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
䷀	䷁	䷂	䷃	䷄	䷅	䷆	䷇	䷈	䷉	䷊	䷋	䷌	䷍	䷎	䷏
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
䷐	䷑	䷒	䷓	䷔	䷕	䷖	䷗	䷘	䷙	䷚	䷛	䷜	䷝	䷞	䷟
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
䷠	䷡	䷢	䷣	䷤	䷥	䷦	䷧	䷨	䷩	䷪	䷫	䷬	䷭	䷮	䷯
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
䷰	䷱	䷲	䷳	䷴	䷵	䷶	䷷	䷸	䷹	䷺	䷻	䷼	䷽	䷾	䷿

Binary system

The binary system is a positional, base 2 counting system.

1. Two digits: 0 and 1
2. In the decimal system there are ten digits (figures): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
3. In the hexadecimal system (base 16 system) there are 16 digits: 0, 1, 2, 3, 4, 5 6, 7, 8, 9, A, B, C, D, E, F

Examples:

dec	bin	hex
10	1010	A
100	1100100	64
123.75	1111011.11	7B.C

Why hexadecimal system is important?

- Each **BI**nary digi**T** is called a bit (abbreviation small letter "b").
- Eight bits is called a byte (abbreviation big letter "B").
- Computers usually use **even** multiples of bytes to store numbers (two, four or eight, sometimes sixteen).
- Each hex (hexadecimal in short) digit is four bits, so a byte is two hexadecimal digits.
- It is relatively easy to memorize the binary appearance of all hexadecimal digits ...

"Big" numerals I

In the SI system we are using prefixes to indicate decadic multiplay or fraction

- kilo (10^3), mega (10^6), giga (10^9),... "big numbers"
- mili (10^{-3}), micro (10^{-6}), nano (10^{-9}),... "small numbers"

Because 2^{10} is 1024 we (I mean computer scientists) start to use the prefix "kilo" in meaning 1024 bytes or 1 "kilo-byte").
In consequence:

- mega-byte 1024×1024
- giga-byte $1024 \times 1024 \times 1024$

This is not correct!

"Big" numerals II

To standardize prefixes IEC 60027-2:1998 standard was developed:

kibi	Ki	2^{10}
mebi	Mi	2^{20}
gibi	Gi	2^{30}
tebi	Ti	2^{40}
pebi	Pi	2^{50}
eksbi	Ei	2^{60}
zebi	Zi	2^{70}
jobi	Yi	2^{80}

Conversions

Decimal to binary

Integers:
The number is divided by two, and we note the result on the left and reminder on the right of the vertical line:

$10 \,|$

Conversions

Decimal to binary

Integers:
The number is divided by two, and we note the result on the left and reminder on the right of the vertical line:

$$\begin{array}{r|l} 10 & \\ 5 & 0 \end{array}$$

Integers:
The number is divided by two, and we note the result on the left and reminder on the right of the vertical line:

$$\begin{array}{r|l} 10 & \\ 5 & 0 \\ 2 & 1 \end{array}$$

Integers:
The number is divided by two, and we note the result on the left and reminder on the right of the vertical line:

$$
\begin{array}{r|l}
10 & \\
5 & 0 \\
2 & 1 \\
1 & 0
\end{array}
$$

Integers:
The number is divided by two, and we note the result on the left and reminder on the right of the vertical line:

$$
\begin{array}{r|l}
10 & \\
5 & 0 \\
2 & 1 \\
1 & 0 \\
0 & 1
\end{array}
$$

Conversions

Decimal to binary

Integers:
The number is divided by two, and we note the result on the left and reminder on the right of the vertical line:

10	
5	0
2	1
1	0
0	1

Reminders (figures on the right), read from bottom to top gives binary value of the converted number.

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

$$\begin{array}{c|l} & .33 \\ 0 & .66 \end{array}$$

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

$$\begin{array}{c|l} & .33 \\ 0 & .66 \\ 1 & .32 \end{array}$$

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33
0	.66
1	.32
0	.64

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33
0	.66
1	.32
0	.64
1	.28

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33
0	.66
1	.32
0	.64
1	.28
0	.56

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33
0	.66
1	.32
0	.64
1	.28
0	.56
1	.12

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

$$\begin{array}{r|l} & .33 \\ 0 & .66 \\ 1 & .32 \\ 0 & .64 \\ 1 & .28 \\ 0 & .56 \\ 1 & .12 \\ 0 & .24 \end{array}$$

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33
0	.66
1	.32
0	.64
1	.28
0	.56
1	.12
0	.24
0	.48

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33
0	.66
1	.32
0	.64
1	.28
0	.56
1	.12
0	.24
0	.48
0	.96

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33
0	.66
1	.32
0	.64
1	.28
0	.56
1	.12
0	.24
0	.48
0	.96
1	.92

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33
0	.66
1	.32
0	.64
1	.28
0	.56
1	.12
0	.24
0	.48
0	.96
1	.92
1	.84

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

	.33
0	.66
1	.32
0	.64
1	.28
0	.56
1	.12
0	.24
0	.48
0	.96
1	.92
1	.84
1	.68

Conversions

Decimal to binary

Fractions: The fraction part is multiplied by two, and the integer part (0 or 1) is noted on the left side of vertical line (integer part is removed for the next calculation)

$$
\begin{array}{c|l}
 & .33 \\
0 & .66 \\
1 & .32 \\
0 & .64 \\
1 & .28 \\
0 & .56 \\
1 & .12 \\
0 & .24 \\
0 & .48 \\
0 & .96 \\
1 & .92 \\
1 & .84 \\
1 & .68
\end{array}
$$

$0.010101000111_{(2)} = 0.329833984375_{(10)}$

Homework!

Processor

Logical operations (Boole's algebra)

Logical AND $Y = A \bigcap B$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Processor

Logical operations (Boole's algebra)

Logical OR $Y = A \bigcup B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Processor

Logical operations (Boole's algebra)

Exclusive OR $Y = A \bigoplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Processor

Logical operations (Boole's algebra)

Logical AND $Y = A \bigcap B$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Logical OR $Y = A \bigcup B$

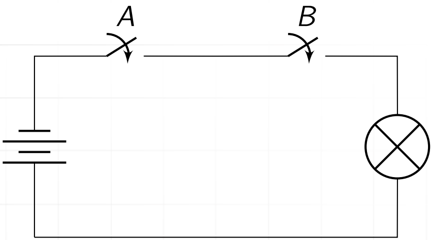
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Exclusive OR $Y = A \bigoplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Logical Operations

Logical AND

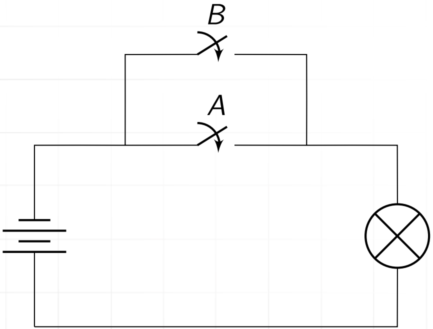


$A \cap B$

AND	0	1
0	0	0
1	0	1

Logical operations

Logical OR



$A \cup B$

OR	0	1
0	0	1
1	1	1

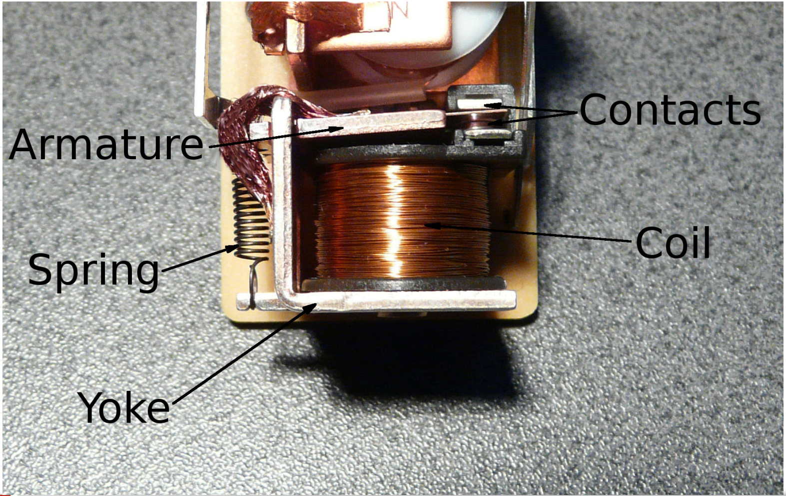
Logical operations

XOR

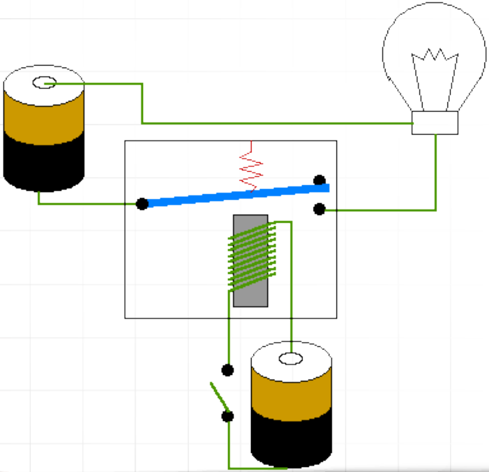
XOR	0	1
0	0	1
1	1	0

$$(\bar{A} \cap B) \cup (A \cap \bar{B})$$

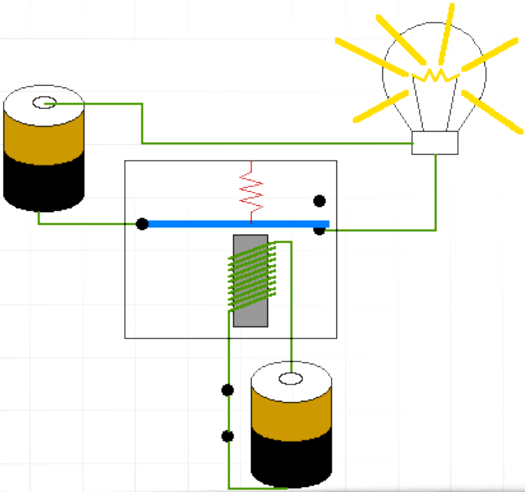
Relay



Relay

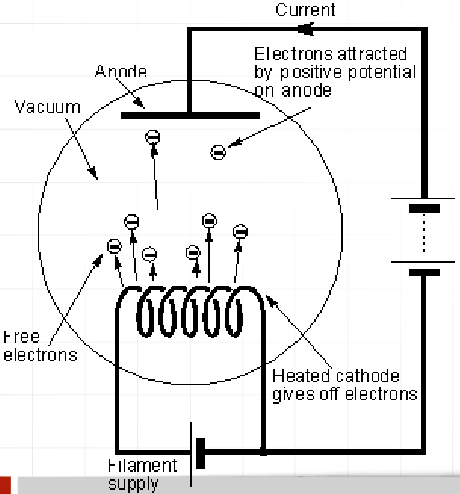


Relay



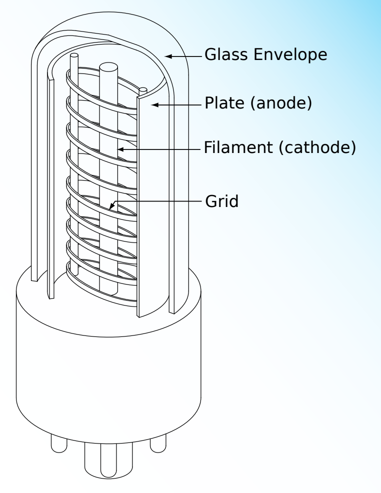
Vacuum tubes (valves)

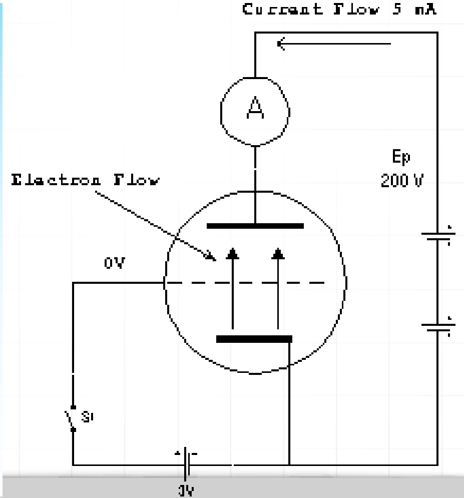
Diode



Vacuum tubes (valves)

Triode





Binary arithmetic operations

1. Addition:
   - $0 + 0 = 0$
   - $0 + 1 = 1$
   - $1 + 0 = 1$
   - $1 + 1 = 10$

Binary arithmetic operations

1. Addition:
   - $0 + 0 = 0$
   - $0 + 1 = 1$
   - $1 + 0 = 1$
   - $1 + 1 = 10$
2. Multiplication:
   - $0 * 0 = 0$
   - $0 * 1 = 0$
   - $1 * 0 = 0$
   - $1 * 1 = 1$

Processor

Addition

1. "Half-Adder"
2. Only two bits ($Y = X_1 + X_2$)
3. Carry (C_{out})
4. "Truth table"

X_1	X_2	Y	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$Y = X_1 \oplus X_2$$

$$C_{out} = X_1 \cap X_2$$

Processor

Addition: Full Adder

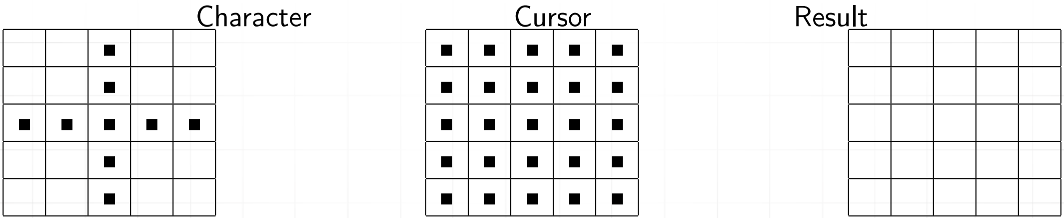
C_{in}	X_1	X_2	Y	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$Y = C_{in} \oplus (X_1 \oplus X_2)$$

$$C_{out} = (X_1 \cap X_2) \cup (C_{in} \cap (X_1 \oplus X_2))$$

Logical operations

XOR application



Logical operations

XOR application

Character

0	0	1	0	0
0	0	1	0	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

Cursor

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Result

Logical operations

XOR application

Character

0	0	1	0	0
0	0	1	0	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

Cursor

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Result

1				

Logical operations

XOR application

Character

0	0	1	0	0
0	0	1	0	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

Cursor

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Result

1	1			

Logical operations

XOR application

Character

0	0	1	0	0
0	0	1	0	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

Cursor

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Result

1	1	0		

Logical operations

XOR application

Character

0	0	1	0	0
0	0	1	0	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

Cursor

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Result

1	1	0	1	

Logical operations

XOR application

Character

0	0	1	0	0
0	0	1	0	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

Cursor

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Result

1	1	0	1	1

Logical operations

XOR application

Character

0	0	1	0	0
0	0	1	0	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

Cursor

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Result

1	1	0	1	1

Logical operations

XOR application

Character

0	0	1	0	0
0	0	1	0	0
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0

Cursor

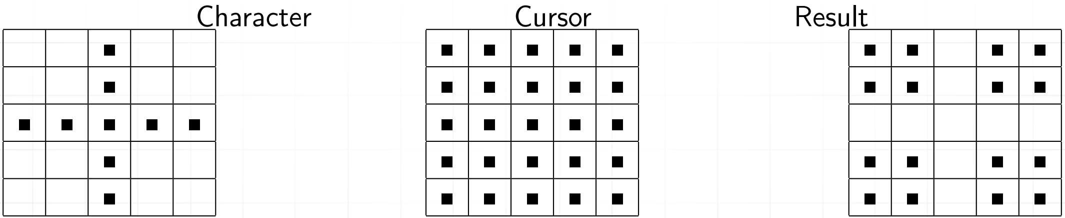
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Result

1	1	0	1	1
1	1	0	1	1
0	0	0	0	0
1	1	0	1	1
1	1	0	1	1

Logical operations

XOR application



XOR — Patent nonsense

Method for dynamically viewing image elements stored in a random access

Patent number: 4197590
Filing date: Jan 19, 1978
Issue date: Apr 8, 1980
Inventors: Josef S. Sukonick, Greg J. Tilden
Assignees: NuGraphics, Inc.
Primary Examiner: Thomas M. Heckler

Binary numbers

- Each binary digit is **bit**: **BI**nary digi**T**

Binary numbers

- Each binary digit is **bit**: **BI**nary digi**T**
- Eight bits is **byte**. In byte:

Binary numbers

- Each binary digit is **bit**: **BI**nary digi**T**
- Eight bits is **byte**. In byte:
  - 00000000 to 0 (zero)

Binary numbers

- Each binary digit is **bit**: **BI**nary digi**T**
- Eight bits is **byte**. In byte:
  - 00000000 to 0 (zero)
  - 11111111 to 255 $1 * 2^7 + 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 2^8 - 1$

Binary numbers

- Each binary digit is **bit**: **BI**nary digi**T**
- Eight bits is **byte**. In byte:
  - 00000000 to 0 (zero)
  - 11111111 to 255 $1*2^7+1*2^6+1*2^5+1*2^4+1*2^3+1*2^2+1*2^1+1*2^0=2^8-1$
- Word (Computer jargon) is group of bytes

Binary numbers

- Each binary digit is **bit**: **BI**nary digi**T**
- Eight bits is **byte**. In byte:
  - 00000000 to 0 (zero)
  - 11111111 to 255 $1*2^7+1*2^6+1*2^5+1*2^4+1*2^3+1*2^2+1*2^1+1*2^0=2^8-1$
- Word (Computer jargon) is group of bytes
  - 16 bits architecture: 2

Binary numbers

- Each binary digit is **bit**: **BI**nary digi**T**
- Eight bits is **byte**. In byte:
  - 00000000 to 0 (zero)
  - 11111111 to 255 $1 * 2^7 + 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 2^8 - 1$
- Word (Computer jargon) is group of bytes
  - 16 bits architecture: 2
  - 32 bits architecture: 4

Binary numbers

- Each binary digit is **bit**: **BI**nary digi**T**
- Eight bits is **byte**. In byte:
  - 00000000 to 0 (zero)
  - 11111111 to 255 $1*2^7+1*2^6+1*2^5+1*2^4+1*2^3+1*2^2+1*2^1+1*2^0=2^8-1$
- Word (Computer jargon) is group of bytes
  - 16 bits architecture: 2
  - 32 bits architecture: 4
  - 64 bits architecture: 8

Negative numbers?

1. In the decimal system: $+3$ or 3, and negative: -3

Negative numbers?

1. In the decimal system: $+3$ or 3, and negative: -3
2. In binary system (theoretically): $+00000011$ or $-00000011\ldots$

Negative numbers?

1. In the decimal system: $+3$ or 3, and negative: -3
2. In binary system (theoretically): $+00000011$ or $-00000011\ldots$
3. $\ldots$ but how to note signs $+$ and $-$?

Negative numbers?

1. In the decimal system: $+3$ or 3, and negative: -3
2. In binary system (theoretically): $+00000011$ or $-00000011\ldots$
3. $\ldots$ but how to note signs $+$ and $-$?
4. The easiest way is to use zero as plus   3 — 00000011

Negative numbers?

1. In the decimal system: $+3$ or 3, and negative: -3
2. In binary system (theoretically): $+00000011$ or $-00000011\ldots$
3. ... but how to note signs $+$ and $-$?
4. The easiest way is to use zero as plus 3 — 00000011
5. The easiest way is to use one as minus -3 — 10000011

Negative numbers?

1. In the decimal system: $+3$ or 3, and negative: -3
2. In binary system (theoretically): $+00000011$ or $-00000011\ldots$
3. $\ldots$ but how to note signs $+$ and $-$?
4. The easiest way is to use zero as plus 3 — 00000011
5. The easiest way is to use one as minus -3 — 10000011
6. How comfortable do integer calculations (positive and negative)?

Negative integers

"Subtraction Table:"

$-$	0	1
0	0	1
1	1	0

Negative integers

"Subtraction Table:"

$-$	0	1
0	0	1
1	1	0

We will try it.

(Let assume that we use 4-bit numbers)

- $0011 - 1 = 0010$

Negative integers

"Subtraction Table:"

$-$	0	1
0	0	1
1	1	0

We will try it.

(Let assume that we use 4-bit numbers)

- $0011 - 1 = 0010$
- $0010 - 1 = 0001$

Negative integers

"Subtraction Table:"

$-$	0	1
0	0	1
1	1	0

We will try it.

(Let assume that we use 4-bit numbers)

- $0011 - 1 = 0010$
- $0010 - 1 = 0001$
- $0001 - 1 = 0000$

Negative integers

"Subtraction Table:"

$-$	0	1
0	0	1
1	1	0

We will try it.

(Let assume that we use 4-bit numbers)

- $0011 - 1 = 0010$
- $0010 - 1 = 0001$
- $0001 - 1 = 0000$
- $0000 - 1 = 1111$

Negative integers

"Subtraction Table:"

$-$	0	1
0	0	1
1	1	0

We will try it.

(Let assume that we use 4-bit numbers)

- $0011 - 1 = 0010$
- $0010 - 1 = 0001$
- $0001 - 1 = 0000$
- $0000 - 1 = 1111$

Negative integers

"Subtraction Table:"

$-$	0	1
0	0	1
1	1	0

We will try it.

(Let assume that we use 4-bit numbers)

- $0011 - 1 = 0010$
- $0010 - 1 = 0001$
- $0001 - 1 = 0000$
- $0000 - 1 = 1111$

So -1 is 1111 (in 4-bit word). Isn't it?

Negative integers

Let's check (ance again only four bits):

$$5 + (-1)$$

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ \hline \end{array}$$

Negative integers

Let's check (ance again only four bits):

$$5 + (-1)$$

$$
\begin{array}{ccccc}
 & 0 & 1 & 0 & 1 \\
 & 1 & 1 & 1 & 1 \\
\hline
1 & 0 & 1 & 0 & 0
\end{array}
$$

Negative integers

Let's check (ance again only four bits):

$$5 + (-1)$$

$$\begin{array}{ccccc} & 0 & 1 & 0 & 1 \\ & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \end{array}$$

How do you like this?

Negative integers

Let's check (ance again only four bits):

$$5 + (-1)$$

$$\begin{array}{ccccc} & 0 & 1 & 0 & 1 \\ & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \end{array}$$

How do you like this?
It is called "Two's complement (code)"

An excursus

Decimal numbers, only two digits:

$$\begin{array}{cc} 3 & 3 \\ 9 & 9 \\ \hline \end{array}$$

An excursus

Decimal numbers, only two digits:

$$\begin{array}{ccc} & 3 & 3 \\ & 9 & 9 \\ \hline 1 & 3 & 2 \end{array}$$

Negation

To get the two's complement of a binary number, the bits are inverted, or "flipped", by using the bitwise NOT operation; the value of 1 is then added to the resulting value, ignoring the overflow which occurs when taking the two's complement of 0.:

1 is 0001
inversion: 1110
adding 1: 1111
2 to 0010
inversion: 1101
adding 1: 1110
checking $5 + (-2)$

$$
\begin{array}{ccccc}
 & 0 & 1 & 0 & 1 \\
 & 1 & 1 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1
\end{array}
$$

Multiplication?

Exercise

Let's assume that our computer is 5-bit (4 bits + sign?)

$$
\begin{array}{cccccc}
 & 0 & 0 & 1 & 0 & 1 \\
* & 1 & 1 & 1 & 1 & 0 \\
\hline
\end{array}
$$

Multiplication?

Exercise

Let's assume that our computer is 5-bit (4 bits + sign?)

$$
\begin{array}{cccccc}
 & 0 & 0 & 1 & 0 & 1 \\
* & 1 & 1 & 1 & 1 & 0 \\
\hline
\end{array}
$$

Homework!

GOTHOFREDI GUILLELMI

LEIBNITII,

S. Cæſar. Majeſtatis Conſiliarii, & S. Reg. Majeſt. Britanniarum a Conſiliis Juſtitiæ intimis, nec non a ſcribendâ Hiſtoriâ,

OPERA OMNIA,

Nunc primum collecta, in Claſſes diſtributa, præfationibus & indicibus exornata, ſtudio

LUDOVICI DUTENS.

TOMUS QUARTUS,

In tres partes diſtributus, quarum

I. Continet Philoſophiam in genere, & opuſcula Sinenſes attingentia.

II. Hiſtoriam & Antiquitates.

III. Juriſprudentiam.

GENEVÆ,
Apud FRATRES DE TOURNES.

MDCCLXVIII.

DES CARACTERES DONT FOHI FONDATEUR DE L'EMPIRE CHINOIS S'EST SERVI DANS SES ECRITS, ET DE L'ARITHMETIQUE BINAIRE.

LXVIII. *Des caractères de* FOHI, *fondateur de l'Empire.* LXIX. *De l'arithmétique binaire.* LXX. *De l'arithmétique quinaire, denaire, &c.* LXXI. *De l'arithmétique binaire.* LXXII. *De l'addition.* LXXIII. *De la soustraction & de la multiplication.* LXXIV. *De la division.* LXXV. *De l'utilité de l'arithmétique binaire.*

LXVIII. Il y a bien de l'apparence, que si nos Européens étoient assez informés de la Literature Chinoise, le secours de la Logique, de la Critique, des Mathématiques & de nôtre maniére de nous exprimer plus déterminée que la leur, nous feroit découvrir dans les Monumens Chinois d'une antiquité si reculée, bien des choses inconnues aux Chinois modernes, & même à leurs interprètes postérieurs, tout classiques qu'on les croie. C'est ainsi que le R. P. *Bouvet* & moi nous avons découvert le sens apparemment le plus véritable selon la lettre des caractères de FOHI fondateur de l'Empire, qui ne consistent que dans la combinaison des lignes entiéres & interrompues, & qui passent pour les plus anciens de la Chine, comme ils en sont aussi sans difficulté les plus simples. Il y en a 64 figures comprises dans le livre appellé YE KIM, c'est-à-dire, le livre des Variations; plusieurs siécles après FOHI, l'Empereur VEN' VAM & son fils CHEU CUM, & encore plus de cinq siécles après le célébre CONFUCIUS, y ont cherché des mystères philosophiques. D'autres en ont même voulu tirer une maniére de Géomance, & d'autres vanités semblables.

Au contraire feu M. *Erhard Weigelius* alla à un moindre nombre, attaché au quaternaire ou Tetractys à la façon de *Pythagore* ; ainsi comme dans la progression par 10, nous écrivons tous les nombres dans sa progression quaternaire par 0, 1, 2, 3, par exemple 321 lui signifioit 48 + 8 + 1, c'est-à-dire 57 selon l'expression commune.

LXXI. Cela me donna occasion de penser, que dans la progression binaire ou double, tous les nombres pourroient être écrits par 0 & 1. Ainsi

1	1
10	2
100	4
1000	8
10000	16
100000	32
1000000	64
&c.	&c.

10 vaudra 2
100 vaudra 4
1000 vaudra 8
&c.

Et les nombres tout de suite s'exprimeront ainsi :

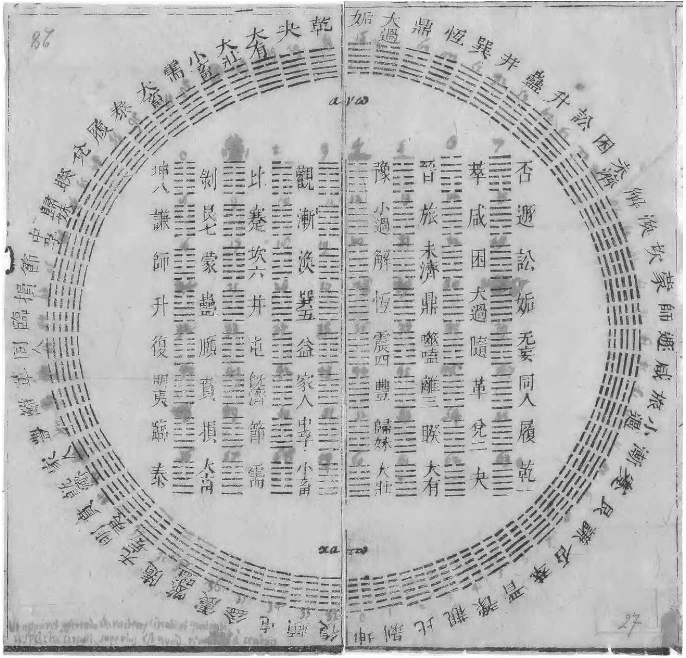
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
10000	16
10001	17
10010	18
10011	19
10100	20
10101	21
10110	22
10111	23
11000	24
11001	25
11010	26
11011	27
11100	28
11101	29
11110	30
11111	31
100000	32
&c.	&c.

Ces Expressions s'accordent avec l'Hypothese, par exemple
$111 = 100 + 10 + 1 = 4 + 2 + 1 = 7$ $11001 = 10000 + 1000 + 1 = 16 + 8 + 1 = 25$
Elles peuvent aussi être trouvées par l'addition continuelle de l'unité, par exemple

Les points marquent l'unité que dans le calcul commun on retient dans la mémoire.

$$
\begin{array}{r}
1 \\
.\,1 \\
\hline
10 \\
1 \\
\hline
11 \\
.\,.\,1 \\
\hline
100 \\
1 \\
\hline
101 \\
.\,1 \\
\hline
110 \\
1 \\
\hline
111 \\
.\,.\,1 \\
\hline
1000
\end{array}
$$

88


乾 夬 大有 大壯 小畜 需 大畜 泰 履 兌 睽 歸妹 中孚 節 損 臨 同人 革 離 豐 家人 既濟 賁 明夷 无妄 隨 噬嗑 震 益 屯 頤 復
姤 大過 鼎 恆 巽 井 蠱 升 訟 困 未濟 解 渙 坎 蒙 師 遯 咸 旅 小過 漸 蹇 艮 謙 否 萃 晉 豫 觀 比 剝 坤
否 遯 訟 姤 无妄 同人 履 乾一
萃 咸 困 大過 隨 革 兌二 夬
晉 旅 未濟 鼎 噬嗑 離三 睽 大有
豫 小過 解 恆 震四 豐 歸妹 大壯
觀 漸 渙 巽五 益 家人 中孚 小畜
比 蹇 坎六 井 屯 既濟 節 需
剝 艮七 蒙 蠱 頤 賁 損 大畜
坤八 謙 師 升 復 明夷 臨 泰

27

Back