Introduction to the Use of Computers

Christophe Rhodes
c.rhodes@gold.ac.uk

Autumn 2012, Fridays: 10:00–12:00: WTA & 15:00–17:00: WHB 300

Bits and Bytes
Bits

A bit:

- 0 or 1; ("binary digit", or a digit in base 2)
- unit in storage or communication; (the space or time it takes to transmit something which can be either a 0 or a 1)
- information unit. (*The quantity of information to distinguish two equiprobable mutually exclusive states*)

The unit is sometimes abbreviated as 'b'.

The bit is the fundamental building block of *digital* (as opposed to *analogue*) computing.

Bits and Bytes

Bytes

A byte:

- originally, simply a collection of bits
- now commonly used for specifically 8 bits ("octets")

The unit is sometimes abbreviated as 'B'.

0	1	1	1	0	1	0	1

An 8-bit byte (if interpreted as an integer, representing the number 117)

Derived units:

- *nybble*: 4 bits
- *crumb*: 2 bits

Units and Prefixes

SI Prefixes

Prefix	Value
kilo- (k)	$1000 = 10^3$
mega- (M)	$1000^2 = 10^6$
giga- (G)	$1000^3 = 10^9$
tera- (T)	$1000^4 = 10^{12}$
peta- (P)	$1000^5 = 10^{15}$
exa- (E)	$1000^6 = 10^{18}$
zetta- (Z)	$1000^7 = 10^{21}$
yotta- (Y)	$1000^8 = 10^{24}$

Examples:

- kilogram (one thousand grams)
- megaton (equivalent of one million tons of TNT)
- gigahertz (one thousand million per second)

Units and Prefixes

Binary Prefixes

Prefix	**Value**
kibi- (ki)	$1024 = 2^{10}$
mebi- (Mi)	$1024^2 = 2^{20}$
gibi- (Gi)	$1024^3 = 2^{30}$
tebi- (Ti)	$1024^4 = 2^{40}$
pebi- (Pi)	$1024^5 = 2^{50}$
exbi- (Ei)	$1024^6 = 2^{60}$
zebi- (Zi)	$1024^7 = 2^{70}$
yobi- (Yi)	$1024^8 = 2^{80}$

Effectively only used with bytes: MiB, GiB are the most common.

History

Storage Devices

Punch cards:

Wikimedia Commons (user Ghw)
Public Domain

Each hole position: one bit

History

Storage Devices

Tape:

Wikimedia Commons (user Poil)

About 1000 bits per inch (1951), about 10^8 (2007)
Modern tape drives: low cost per bit, high capacity (1 TB)

History

Storage Devices

Floppy disks:

Wikimedia Commons
CC-BY-SA 3.0

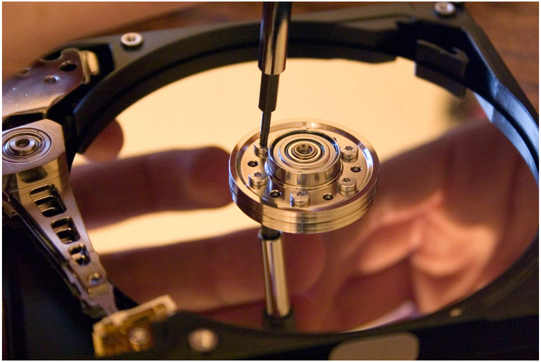
(user Brokensegue)
CC-BY-SA 3.0

175kB (1972), 1.44MB (1987)
Now rarely used: small capacity, bulky, easily damaged.

History

Storage Devices

Hard disks:



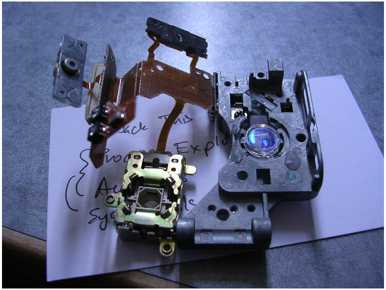
WP:Dave Indesch

5MB (1980), 1GB (1995), 1TB (2007), 4TB (2012)
Concern: capacity has outstripped access/seek times

History

Storage Devices

Optical Media:

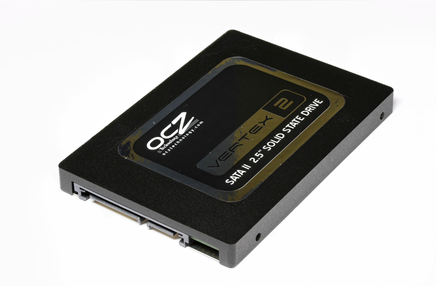


CD: 600MB (1985); DVD: 4.7GB (1995)
HD DVD: 30GB (2006); Blu-Ray: 50GB (2006)
(Blu-Ray won in 2008: now up to 150GB)

History

Storage Devices

Solid-state Drives:



Wikimedia Commons (user D-Kuru)

Fast, quiet
Concern: lifetime of data storage, wearing out

Binary integers

Place-value systems

We write numbers so that their relative placement indicates their value:

1 7 2

Binary integers

Place-value systems

We write numbers so that their relative placement indicates their value:

1 7 2

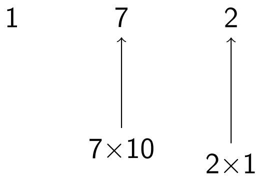
↑

2×1

Binary integers

Place-value systems

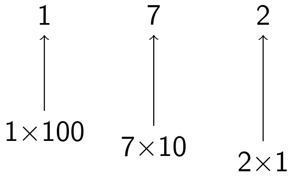
We write numbers so that their relative placement indicates their value:



Binary integers

Place-value systems

We write numbers so that their relative placement indicates their value:



Binary integers

Binary representation

1 0 1 0 1 1 0 0

Binary integers

Binary representation

$$1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0$$

0×1

Binary integers

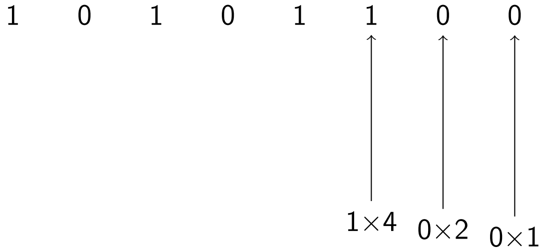
Binary representation

1 0 1 0 1 1 0 0

0×2 0×1

Binary integers

Binary representation



Binary integers

Binary representation

1 0 1 0 1 1 0 0

1×8 1×4 0×2 0×1

Binary integers

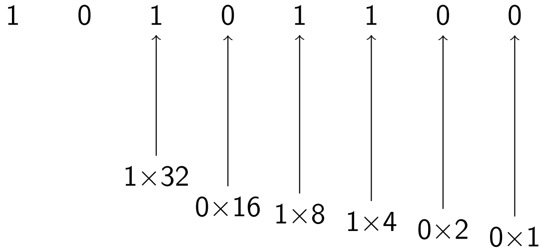
Binary representation

1 0 1 0 1 1 0 0

0×16 1×8 1×4 0×2 0×1

Binary integers

Binary representation



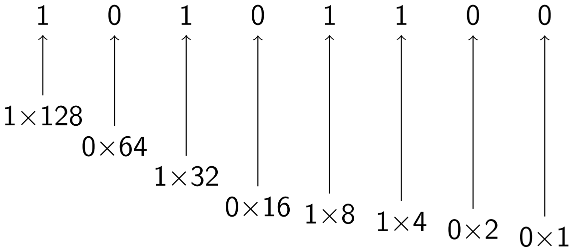
Binary integers

Binary representation

1	0	1	0	1	1	0	0
	0×64	1×32	0×16	1×8	1×4	0×2	0×1

Binary integers

Binary representation



Two methods:

- Multiply out;
- Repeated division.

Binary integers

Conversion between bases

Multiply out:

- already seen binary to decimal
- example: convert decimal 42 to binary:

$$
\begin{aligned}
(42)_{10} &= (4)_{10} \times (10)_{10} + (2)_{10} \\
&= (100)_2 \times (1010)_2 + (10)_2 \\
&= (101000)_2 + (10)_2 \\
&= (101010)_2
\end{aligned}
$$

Binary integers

Conversion between bases

Repeated division:

$$\begin{aligned}
(42)_{10} &= 2 \times (21)_{10} \text{ remainder } 0 \\
(21)_{10} &= 2 \times (10)_{10} \text{ remainder } 1 \\
(10)_{10} &= 2 \times (5)_{10} \text{ remainder } 0 \\
(5)_{10} &= 2 \times (2)_{10} \text{ remainder } 1 \\
(2)_{10} &= 2 \times (1)_{10} \text{ remainder } 0 \\
(1)_{10} &= 2 \times (0)_{10} \text{ remainder } 1
\end{aligned}$$

Read off the remainders in sequence, right to left.

Binary integers

Hexadecimal notation

- Binary notation is wasteful of space;
- Long strings of digits hard to understand;
- Often convenient to use hexadecimal instead.

dec	hex	bin	dec	hex	bin
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	a	1010
3	3	0011	11	b	1011
4	4	0100	12	c	1100
5	5	0101	13	d	1101
6	6	0110	14	e	1110
7	7	0111	15	f	1111

Binary arithmetic

Arithmetic Tables

Addition:

+	0	1
0	0	1
1	1	10

Binary arithmetic

Arithmetic Tables

Subtraction:

−	0	1
0	0	-1
1	1	0

Binary arithmetic

Arithmetic Tables

Multiplication:

×	0	1
0	0	0
1	0	1

Binary Arithmetic

Examples

Binary Logic

Binary logic operates on binary numbers:

- 0 is conventionally ‘false’; 1 is ‘true’;
- operations can have 1 or more inputs
- outputs are logical operations on the inputs

Binary Logic

Binary logic operates on binary numbers:

- 0 is conventionally 'false'; 1 is 'true';
- operations can have 1 or more inputs
- outputs are logical operations on the inputs

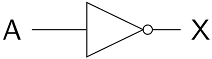
Logic circuits are built from components called logic *gates*, implementing logical or Boolean operations:

- common elements: AND, OR, NOT
- primitives: NAND, NOR

Binary Logic

Logic Gates: NOT

NOT relation implemented by a logic gate:



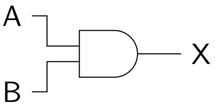
A	X
0	1
1	0

Notation: $X = \neg A$; $\overline{A}$; A'

Binary Logic

Logic Gates: AND

AND relation implemented by a logic gate:



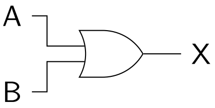
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Notation: $X = A \wedge B$; $A \cdot B$; $A \times B$

Binary Logic

Logic Gates: OR

OR relation implemented by a logic gate:



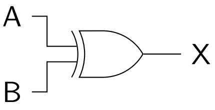
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Notation: $X = A \vee B$; $A + B$

Binary Logic

Logic Gates: XOR

XOR relation implemented by a logic gate:



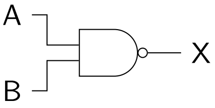
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Notation: $X = A \oplus B$

Binary Logic

Logic Gates: NAND

NAND relation implemented by a logic gate:



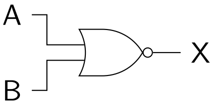
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Notation: $X = \overline{A \wedge B}$

Binary Logic

Logic Gates: NOR

NOR relation implemented by a logic gate:

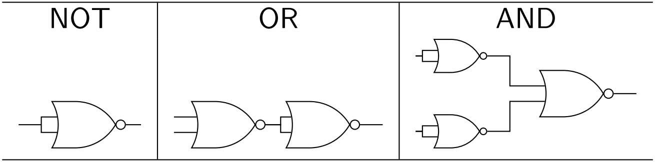


A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Notation: $X = \overline{A \vee B}$

Binary Logic

Logic Gates: NOR as primitive



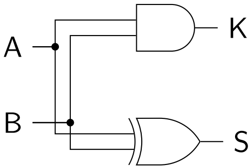
Binary Arithmetic

Sum and Carry: Half Adder

A	B	S	K
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Binary Arithmetic
Sum and Carry: Half Adder

A	B	S	K
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Binary Arithmetic

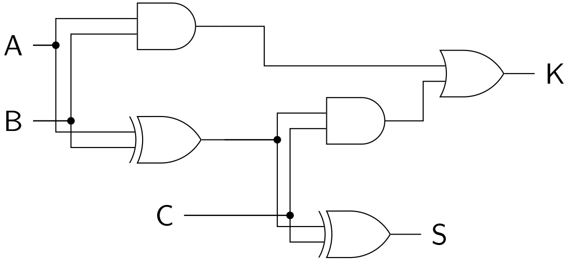
Sum and Carry: Full Adder

A	B	C	S	K
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

- S & K...
  - S is 1 if there are an odd number of 1s in A,B,C
  - K is 1 if there are two or more 1s in A,B,C
- or...
  - S is $A \oplus B \oplus C$
  - K is $(A \wedge B) \vee ((A \oplus B) \wedge C)$

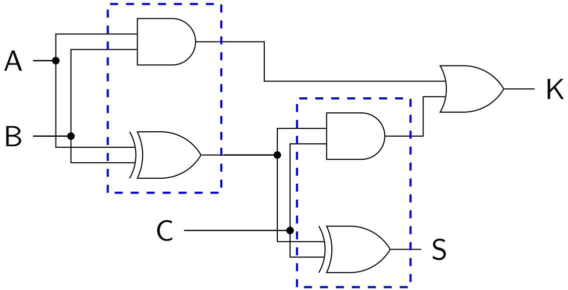
Binary Arithmetic

Sum and Carry: Full Adder



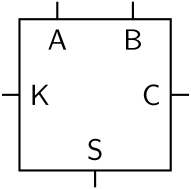
Binary Arithmetic

Sum and Carry: Full Adder



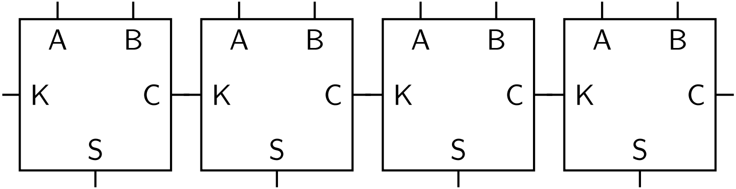
Binary Arithmetic

Sum and Carry: Ripple Adder



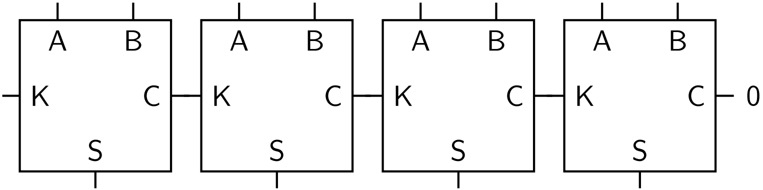
Binary Arithmetic

Sum and Carry: Ripple Adder



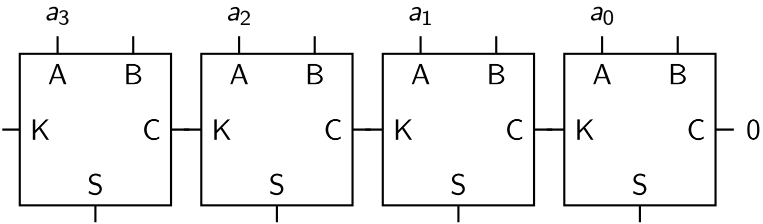
Binary Arithmetic

Sum and Carry: Ripple Adder



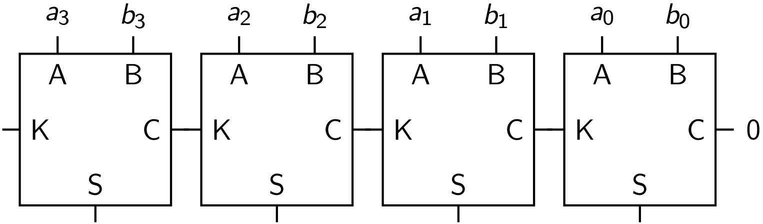
Binary Arithmetic

Sum and Carry: Ripple Adder



Binary Arithmetic

Sum and Carry: Ripple Adder



Binary Arithmetic

Sum and Carry: Ripple Adder