Sistem Bilangan dalam Komputer

Outline

1. Sistem bilangan dalam komputer

2. Representasi data di komputer

Sistem bilangan dalam komputer

Mengapa biner?

- Perintah dalam bahasa mesin dikodekan dalam format biner (*binary*).
- Sistem biner: bilangan berbasis 2
  - Posisi digit menandakan pangkat dari 2
  - Digit yang dipakai hanya 0 dan 1
- Bandingkan dengan sistem desimal!
  - Digit yang dipakai: $0,1,2,3,4,5,6,7,8,9$
  - Posisi digit menandakan pangkat dari 10
- Mengapa pakai sistem biner?
  - Mudah dan murah diimplementasikan (via transistor dalam sirkuit elektronik)
    - 0/1, false/true, low voltage/high voltage, etc
  - Handal (*reliable*) jika hanya butuh dua macam digit

Contoh ekspansi desimal

- Bilangan terdiri dari serangkaian digit, masing-masing dengan sebuah bobot (*weight*).

5	3	7	.	2	1	4	digit
100	10	1		1/10	1/100	1/1000	bobot

- Bilangan desimal memiliki basis (*base*) atau *radix* 10.
- Bobot: pangkat dari basis.

5	3	7	.	2	1	4	digit
10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	bobot

- Nilai bilangannya diperoleh dengan cara: kalikan tiap digit dengan bobotnya, lalu jumlahkan.

$$(5 \times 10^2) + (3 \times 10^1) + (7 \times 10^0) + (2 \times 10^{-1}) + (1 \times 10^{-2}) + (4 \times 10^{-3}) = 537.214$$

Bentuk umum representasi berbasis posisi

- Bilangan D dalam basis/radix R ditulis dalam bentuk serangkaian digit, yakni:

$$D = d_{m-1}d_{m-2}\ldots d_1 d_0 . d_{-1}d_{-2}\ldots d_{-n}$$

  di mana setiap $d_{m-1}, \ldots, d_0, d_{-1}, \ldots, d_{-n}$ adalah digit yang diijinkan untuk radix R.
- Contoh: untuk desimal, radix $R = 10$. Digit yang diizinkan: $0, 1, \ldots, 9$.
- Contoh: untuk representasi biner, radix $R = 2$, dan digit yang diizinkan: $0, 1$.
- Nilai D jika dinyatakan dalam desimal adalah:

$$D = \sum_{i=-n}^{m-1} d_i R^i$$

Representasi biner

- Representasi biner menggunakan radix 2, sehingga bobot yang dipakai adalah pangkat dari 2.
- Contoh: untuk bilangan biner 1101.01

1	1	0	1	.	0	1	binary digits atau bits
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	bobot dalam basis 2

- Secara umum, representasi biner berbentuk:

$$b_{m-1}b_{m-2}\ldots b_1 b_0 . b_{-1}b_{-2}\ldots b_{-n}$$

  - setiap $b_{m-1},\ldots,b_0,b_{-1},\ldots,b_{-n}$ bernilai 0 atau 1
  - bit b_{m-1} (yang bobotnya terbesar) disebut most significant bit (MSB)
  - bit b_{-n} (yang bobotnya terkecil) disebut least significant bit (LSB)

Konversi biner ke desimal

- Nilai desimal dari bilangan biner 1101.01 adalah

$$\begin{aligned}&(1\times 2^3)+(1\times 2^2)+(0\times 2^1)+(1\times 2^0)+(0\times 2^{-1})\\&\quad+(1\times 2^{-2})\\&=8+4+0+1+0+0.25\\&=13.25\end{aligned}$$

- Secara umum, jika bilangan biner B berbentuk

$$b_{m-1}b_{m-2}\ldots b_1 b_0 . b_{-1}b_{-2}\ldots b_{-n}$$

maka, nilai desimalnya:

$$B=\sum_{i=-n}^{m-1} b_i\times 2^i$$

Pangkat dari 2

$2^0 = 1$ $2^1 = 2$ $2^2 = 4$ $2^3 = 8$

$2^4 = 16$ $2^5 = 32$ $2^6 = 64$ $2^7 = 128$

$2^8 = 256$ $2^9 = 512$ $2^{10} = 1024$ $2^{11} = 2048$

$2^{12} = 4096$ $2^{13} = 8192$ $2^{14} = 16384$ $2^{15} = 32768$

$2^{-1} = 0.5$ $2^{-2} = 0.25$ $2^{-3} = 0.125$

$2^{-4} = 0.0625$ $2^{-5} = 0.03125$ $2^{-6} = 0.015625$

Mari dicoba!

Ubah bilangan-bilangan dalam representasi biner berikut menjadi representasi desimal.

- 11011_2
- 101101_2
- 1011.011_2

Konversi desimal ke biner

- Bagi bagian bulat desimal (di depan dot/koma) dengan 2 berulang-ulang hingga hasil bagi atau quotient-nya 0. Lalu kumpulkan sisa bagi (remainder) mulai dari hasil bagi yang terakhir.
- Untuk bagian pecahan (di belakang dot/koma), kalikan bagian pecahan tersebut dengan 2 sampai jadi 0 di belakang koma. Lalu kumpulkan bagian bulat dari hasil perkalian tersebut dari depan.

Contoh: desimal 162.375

$162/2 = 81$ rem $0 \sim b_0 = 0$ ⇑

$81/2 = 40$ rem $1 \sim b_1 = 1$ ⇑

$40/2 = 20$ rem $0 \sim b_2 = 0$ ⇑

$20/2 = 10$ rem $0 \sim b_3 = 0$ ⇑

$10/2 = 5$ rem $0 \sim b_4 = 0$ ⇑

$5/2 = 2$ rem $1 \sim b_5 = 1$ ⇑

$2/2 = 1$ rem $0 \sim b_6 = 0$ ⇑

$1/2 = 0$ rem $1 \sim b_7 = 1$ ⇑

$0.375 \times 2 = 0.75 \sim b_{-1} = 0$ ⇓

$0.75 \times 2 = 1.5 \sim b_{-2} = 1$ ⇓

$0.5 \times 2 = 1.0 \sim b_{-3} = 1$ ⇓

Jadi $162.375_{10} = (b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 . b_{-1} b_{-2} b_{-3})_2 = 10100010.011_2$

Mari coba!

Ubah bilangan-bilangan desimal berikut ke dalam representasi biner.

- 17
- 127.75
- 0.625

Mengapa proses konversinya benar?

- Teknik yang dideskripsikan sebelumnya dapat diterapkan untuk konversi desimal ke representasi basis **berapa saja**.
- Perhatikan konversi 162.375 dari desimal ke desimal:

$$162/10 = 16 \text{ rem } 2$$
$$16/10 = 1 \text{ rem } 6$$
$$1/10 = 0 \text{ rem } 1$$

kumpulkan sisa bagi dari "bawah" diperoleh 162

- Setiap operasi pembagian membuang digit paling kanan (yang menjadi sisa bagi). Quotient-nya merupakan digit-digit yang tersisa.
- Demikian pula, untuk mengkonversi bagian pecahan, setiap perkalian membuang digit paling kiri (bagian bulat). Bagian pecahan (fraction) dari hasilnya mewakili digit-digit yang tersisa.

$$0.375 \times 10 = 3.75$$
$$0.75 \times 10 = 7.50$$
$$0.50 \times 10 = 5.00$$

Hexadecimal

- Notasi biner paaaaaanjang.
- Heksadesimal bisa dipakai untuk menyederhanakan notasi biner.
- Radix = 16
- Digit yang dipakai: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F (boleh huruf kecil).
- Contohnya digunakan untuk mendeskripsikan IPv6 addresses atau 24-bit colors.
  - IPv6: FE80:0000:0000:0000:0202:B3FF:FE1E:8329
  - Merah: #FF0000

Menyingkat biner dengan hexadecimal

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

- Konversi biner ke hex: ambil 4 bit dari dot/koma, konversi sesuai tabel, tambahkan 0 di ujung jika perlu.
- Konversi hex ke biner: konversi tiap digit dengan 4 bit sesuai tabel.

Biner	Hex
10110100	B4
101.101	5.A
1001100001.00110101	261.35
1010011100	?
10.10111	?

Contoh penggunaan hex di Python/Java

- Unicode character values
  - Contoh: `"\u03B1"` adalah karakter α
- Hexadecimal integer biasanya ditulis dengan awalan `0x`
  - Contoh: `0xFACE21`
- Basis kadang ditulis sebagai subscript: $7A_{16}$, 99_{16}

Base 8 - Octal

- Representasi octal menggunakan 8 digits:
  - Radix = 8
  - Digits: 0,1,2,3,4,5,6,7
- Banyak digunakan di awal sejarah komputer, sekarang masih muncul misalnya di file access permissions untuk sistem Unix.

Konversi antara biner dan octal

- Konversi biner ke hex: ambil 3 bit dari dot/koma, konversi sesuai tabel, tambahkan 0 diujung jika perlu.
- Konversi hex ke biner: konversi tiap digit dengan 3 bit sesuai tabel.

Desimal	Biner	Octal
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
55	110111	67
?	?	765
?	?	7.432

Konversi hexadecimal, biner, octal

- $9FB_{16}$ = (. . . . .)$_2$
- $4.C8A_{16}$ = (. . . . .)$_8$
- 330_{10} = (. . . . .)$_8$

Unsigned binary numbers

Jumlah bit menentukan rentang bilangan yang dapat direpresentasikan.

Jika hanya mempertimbangkan bilangan positif:

- 2 bit ~ min: 0_{10}, max: 3_{10}
- 4 bit ~ min: 0_{10}, max: 15_{10}
- 8 bit ~ min: 0_{10}, max:

Unsigned binary numbers

Jumlah bit menentukan rentang bilangan yang dapat direpresentasikan.

Jika hanya mempertimbangkan bilangan positif:

- 2 bit ~ min: 0_{10}, max: 3_{10}
- 4 bit ~ min: 0_{10}, max: 15_{10}
- 8 bit ~ min: 0_{10}, max: 255_{10}
- 16 bit ~ min: 0_{10}, max:

Unsigned binary numbers

Jumlah bit menentukan rentang bilangan yang dapat direpresentasikan.

Jika hanya mempertimbangkan bilangan positif:

- 2 bit ~ min: 0_{10}, max: 3_{10}
- 4 bit ~ min: 0_{10}, max: 15_{10}
- 8 bit ~ min: 0_{10}, max: 255_{10}
- 16 bit ~ min: 0_{10}, max: $2^{16} - 1 = 65535_{10}$
- n bit ~ min: 0_{10}, max:

Unsigned binary numbers

Jumlah bit menentukan rentang bilangan yang dapat direpresentasikan.

Jika hanya mempertimbangkan bilangan positif:

- 2 bit ~ min: 0_{10}, max: 3_{10}
- 4 bit ~ min: 0_{10}, max: 15_{10}
- 8 bit ~ min: 0_{10}, max: 255_{10}
- 16 bit ~ min: 0_{10}, max: $2^{16} - 1 = 65535_{10}$
- n bit ~ min: 0_{10}, max: $2^n - 1_{(10)}$

Bilangan negatif dalam representasi biner

- Untuk mengakomodasi **bilangan negatif**,
  n bit tidak merepresentasikan 0 s.d. $2^n - 1$
- Tapi, n bit merepresentasikan $-(2^{n-1})$ s.d. $+(2^{n-1} - 1)$
- Alternatif representasi
  - Signed-and-magnitude
    - Gunakan 1 bit (biasanya di posisi MSB) sebagai tanda
      0 ~ positif 1 ~ negatif
  - One's complement (silakan baca sendiri di e.g., Wikipedia)
  - Two's complement
- Dengan representasi bilangan negatif, kita bisa melakukan pengurangan (*subtraction*) dengan operasi penjumlahan (*addition*).
  - $A - B = A + (-B)$

Two's complement

- Digunakan di banyak komputer modern, karena lebih mudah diimplementasikan di hardware.
- Mengubah bilangan biner ke two's complement-nya:
  - Ubah semua bit 1 jadi 0 dan semua 0 jadi 1.
  - Jumlahkan dengan 1 ke hasilnya.
- Jumlah digit yang disediakan harus diperhatikan.
- Buang *carry* (lebihan dalam operasi) jika ada.

Notes:

- Nilai $2^n - 1$ dalam sistem biner menjadi n buah bit 1. Contoh: $2^5 - 1 = 11111_2$
- Untuk bilangan biner B yang terdiri n bit, $(2^n - 1) - B$ adalah komplemen dari B. Contoh: komplemen (5-bit) dari 10001_2 adalah $(2^5 - 1) - 10001_2 = 01110_2$.
- Two's complement dari B adalah 1 + komplemen dari B, atau $(2^n - 1) - B + 1$. Jadi, two's complement dari 10001 adalah 1 + 01110 = 01111.

Contoh

Jumlah digit = 8

X	Two's complement dari X
01001101	?
00000000	?
10000000	?

Two's complement sebagai representasi negatif biner

- Negatif dari bilangan biner B adalah two's complement dari B.
- Sign bit = 1 berarti negatif, sign bit = 0 berarti positif.
  - Sign bit terletak di sebelah kiri MSB.

Contoh untuk 4 bit

Decimal	Two's compl. bin.	Signed-magnitude bin.
-8	1000	-
-7	1001	1111
-6	1010	1110
-5	1011	1101
-4	1100	1100
-3	1101	1011
-2	1110	1010
-1	1111	1001
0	0000	1000 atau 0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111

Representa komputer

Penjumlahan dan pengurangan dengan two's complement

- Penjumlahan dilakukan seperti biasa asalkan tidak melebihi kapasitas (tidak *overflow*)
- Pengurangan dilakukan dengan menjumlahkan bilangan yang dikurangi (*minuend*) dengan two's complement dari bilangan pengurang (*subtrahend*)
  - Buang carry jika ada.

Contoh penjumlahan untuk 4 bit

```
                0011   (carry)     0110   (carry)
0010 (+2)        0011 (+3)          0111 (+7)
0100 (+4)        0011 (+3)          0110 (+6)
========= +    ========== +       ========== +
0110 (+6)        0110 (+6)          1101 (-5) invalid!

1100   (carry)
 1110 (-2)
 1100 (-4)
========= +
 1010 (-6)
```

Contoh pengurangan untuk 4 bit

```
                    0000  (carry)
0010 (+2)            0010 (+2)
0100 (+4)     ==>    1100 (-4)
========= -          ========= +
                     1110 (-2)

                    1100  (carry)
1110 (-2)            1110 (-2)
1100 (-4)     ==>    0100 (+4)
========= -          ========= +
                     0010  (+2)
```

Konversi biner two's complement ke desimal

- Sama caranya dengan konversi biner unsigned untuk semua bit kecuali sign bit.
- Untuk sign bit, bobotnya dinegatifkan sebelum dikalikan dengan bit-nya.

Contoh untuk 3 bit:

- Unsigned:
  $1100_2 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 12_{10}$
  $0100_2 = (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4_{10}$
- Signed (two's complement):
  $1100_2 = (1 \times -(2^3)) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = -4_{10}$
  $0100_2 = (0 \times -(2^3)) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4_{10}$

Sign extension

- Kadang-kadang kita perlu “memperpanjang” representasi biner, misalnya dari 4 bit jadi 8 bit.
- Agar tidak kehilangan informasi sign bit, maka tambahan bit (*padding*) dilakukan di sebelah kiri sign bit dengan menduplikasi sign bit.
- Nilai bilangan tidak berubah.

Sign extension

- Kadang-kadang kita perlu “memperpanjang” representasi biner, misalnya dari 4 bit jadi 8 bit.
- Agar tidak kehilangan informasi sign bit, maka tambahan bit (*padding*) dilakukan di sebelah kiri sign bit dengan menduplikasi sign bit.
- Nilai bilangan tidak berubah.
- Contoh:
  - 4 bit: 0101 6 bit: 000101

Sign extension

- Kadang-kadang kita perlu “memperpanjang” representasi biner, misalnya dari 4 bit jadi 8 bit.
- Agar tidak kehilangan informasi sign bit, maka tambahan bit (*padding*) dilakukan di sebelah kiri sign bit dengan menduplikasi sign bit.
- Nilai bilangan tidak berubah.
- Contoh:
  - 4 bit: 0101 6 bit: 000101
    Nilainya sama-sama 5_{10}.

Sign extension

- Kadang-kadang kita perlu “memperpanjang” representasi biner, misalnya dari 4 bit jadi 8 bit.
- Agar tidak kehilangan informasi sign bit, maka tambahan bit (*padding*) dilakukan di sebelah kiri sign bit dengan menduplikasi sign bit.
- Nilai bilangan tidak berubah.
- Contoh:
  - 4 bit: 0101 6 bit: 000101
    Nilainya sama-sama 5_{10}.
  - 4 bit: 1100 6 bit: 111100

Sign extension

- Kadang-kadang kita perlu “memperpanjang” representasi biner, misalnya dari 4 bit jadi 8 bit.
- Agar tidak kehilangan informasi sign bit, maka tambahan bit (*padding*) dilakukan di sebelah kiri sign bit dengan menduplikasi sign bit.
- Nilai bilangan tidak berubah.
- Contoh:
  - 4 bit: 0101 6 bit: 000101
    Nilainya sama-sama 5_{10}.
  - 4 bit: 1100 6 bit: 111100
    Nilainya sama-sama -4_{10}.
    (Coba cek dengan menghitung two’s complement dari 4 untuk panjang 6 bit.)

Representasi data di komputer

1. Sistem bilangan dalam komputer

2. Representasi data di komputer

Bits, Bytes, Words

- Bit: satu digit 0 atau 1
- Byte: 8 bit
- Word: standar satuan penyimpanan data di komputer.
  - Komputer 32-bit, 1 word = 32 bit = 4 byte
  - Komputer 64-bit, 1 word = 64 bit = 8 byte

Aproksimasi desimal dari biner

$$2^{10} = 1024 \approx 10^3 = 1000 \text{ (kilo)}$$

$$2^{20} = 1{,}048{,}576 \approx 10^6 = 1{,}000{,}000 \text{ (mega)}$$

$$2^{30} = 1{,}073{,}741{,}824 \approx 10^9 = 1{,}000{,}000{,}000 \text{ (giga)}$$

$$2^{40} = 1{,}099{,}511{,}627{,}776 \approx 10^{12} = 1{,}000{,}000{,}000{,}000 \text{ (tera)}$$

$$2^{50} = 1{,}125{,}899{,}906{,}842{,}624 \approx 10^{15} = 1{,}000{,}000{,}000{,}000{,}000 \text{ (peta)}$$

$$2^{60} \approx 10^{18} \text{ (exa)}$$

$$2^{70} \approx 10^{21} \text{ (zetta)}$$

$$\vdots$$

Floating point

- Data semua berbentuk biner: tidak ada pecahan
- Pecahan, bilangan irrasional, dan bilangan real (= floats), diaproksimasi sebagai bilangan desimal dengan akurasi sampai sekian digit “di belakang koma”

Karakter/Huruf

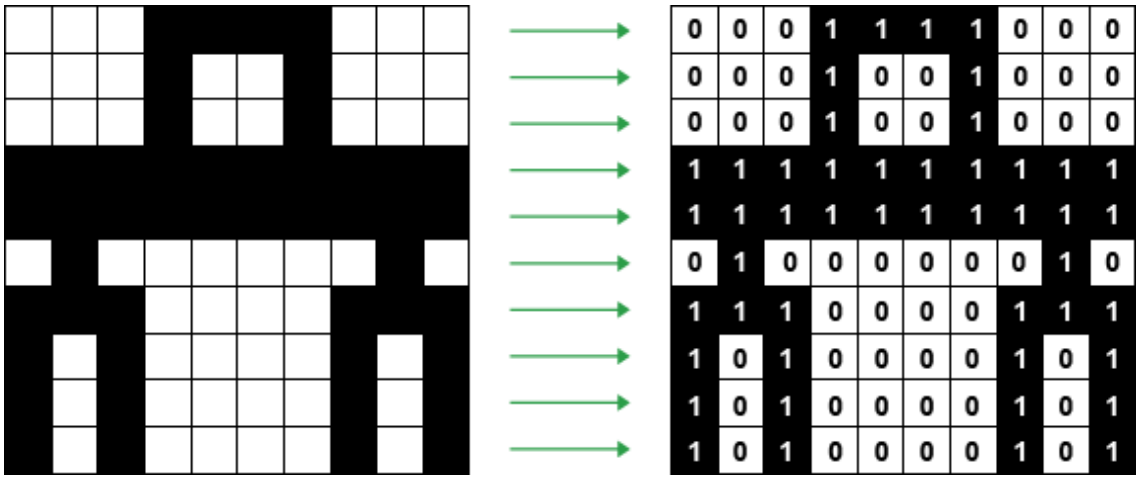
- Huruf/karakter direpresentasikan (*encoding*) sebagai bilangan (biner).
- Contoh: ASCII dan UTF-8 encodings.

Char	Dec	Char	Dec	Char	Dec	Char	Dec
NUL	0	SP	32	@	64	`	96
SOH	1	!	33	A	65	a	97
STX	2	"	34	B	66	b	98
ETX	3	#	35	C	67	c	99
EOT	4	$	36	D	68	d	100
ENQ	5	%	37	E	69	e	101
ACK	6	&	38	F	70	f	102
BEL	7	'	39	G	71	g	103
BS	8	(	40	H	72	h	104

Figure: Contoh UTF-8 Encoding [Table 0.3, Punch & Enbody, 2017].

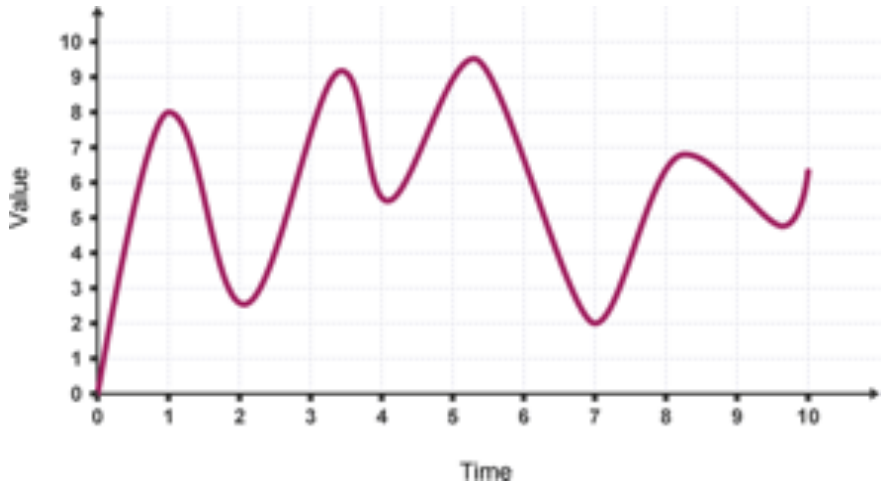
Citra (Gambar) dan Audio (Suara)

- Data citra digital: matriks berisi kumpulan nilai warna (*pixel*)
  - Tiap pixel direpresentasikan sebagai suatu bilangan biner.



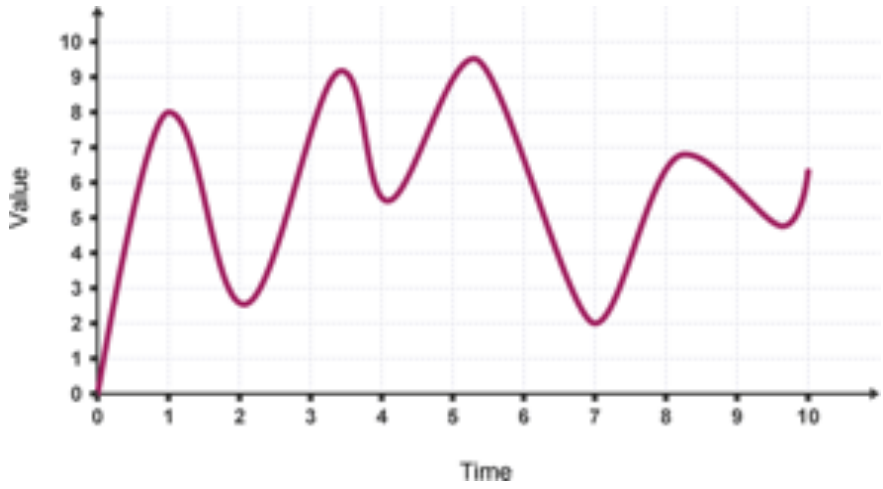
Citra (Gambar) dan Audio (Suara)

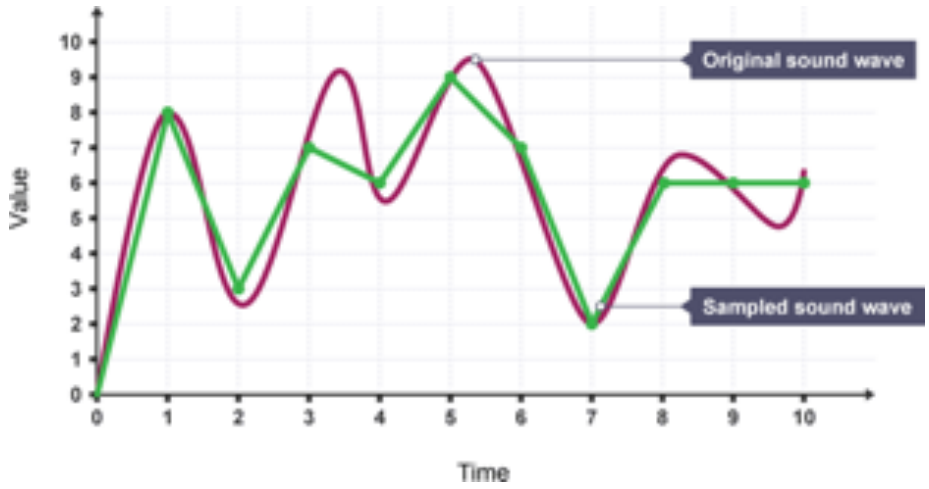
- Data audio digital: hasil sampling dari gelombang suara (yang bersifat analog)



Citra (Gambar) dan Audio (Suara)

- Data audio digital: hasil sampling dari gelombang suara (yang bersifat analog)





Berapa besar?

- 2 KB(ytes): satu halaman teks yang diketik
- 10 KB: satu halaman ensiklopedia
- 50 KB: satu dokumen (terkompresi)
- 100 KB: satu foto dalam resolusi rendah
- 1 MB: satu novel kecil
- 2 MB: satu foto resolusi tinggi
- 5 MB: seluruh karya Shakespeare atau satu video berdurasi 30 detik
- 100 MB: satu rak buku sepanjang 1 meter
- 500 MB: satu keping CD-ROM
- 1 GB: Satu truk pick-up berisi kertas dokumen, atau satu simfoni, atau satu film
- 20 GB: Seluruh karya Beethoven sebagai file audio
- 50 GB: Buku satu lantai perpustakaan
- 100 GB: kapasitas standar Blu-ray
- 1 TB: Seluruh data rontgen sebuah rumah sakit, atau 50 ribu pohon dijadikan kertas dan dicetak, atau data Earth Observation Satellite per hari di tahun 1998
- 2 TB: satu perpustakaan riset akademik
- 20 TB: total foto per bulan di Facebook
- 100 TB: seluruh data di US Library of Congress tahun 2009
- 530 TB: seluruh video di Youtube

1 Petabyte ≈ 1000 TB

- Google cluster: 4 PB RAM, memroses 20 PB data per hari
- World of Warcraft: butuh 1.3 PB untuk mengelola game-nya
- Jika punya storage 1 PB:
  - Jika baca 1 buku (1 MB) per hari selama 80 tahun, kita hanya butuh 30 GB
  - Jika ambil 100 foto resolusi tinggi (4 MB per foto) tiap hari selama 80 tahun, kita hanya butuh 30 TB
  - Jika mendengarkan audio MP3 (1 MB per menit) 24 jam per hari dan 7 hari seminggu selama 80 tahun, kita hanya butuh 42 TB
  - Jika menonton video dalam format DVD (2 GB per jam), maka 1 PB cukup untuk video berdurasi 500 ribu jam atau 57 tahun.

Exabyte & Zettabyte

- 5 Exabyte: seluruh kata yang pernah diucapkan manusia jika ditulis dalam format teks
- 5-8 Exabyte: besar lalu lintas data di Internet setiap bulan
- 500 EB: total konten digital sedunia tahun 2009
- 42 ZB: seluruh kata yang pernah diucapkan manusia sepanjang sejarah, dalam format 16 bit audio 16 KHz.