Analisis Numerik
Representasi Bilangan

Ahmad Rio Adriansyah

STT Terpadu - Nurul Fikri

ahmad.rio.adriansyah@gmail.com
arasy@nurulfikri.ac.id

March 2, 2019

Representasi Bilangan Bulat

Dalam komputer, bilangan dituliskan dalam bit (binary digit).

Contoh : 13 direpresentasikan dalam 8-bit

$$
\begin{aligned}
13_{10} &= 00001101_2 \\
&= 0x2^7 + 0x2^6 + 0x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 \\
&= 0 + 0 + 0 + 0 + 8 + 4 + 0 + 1
\end{aligned}
$$

Binary ke Desimal

$$00001101_2 = \underbrace{0}_{2^7}\underbrace{0}_{2^6}\underbrace{0}_{2^5}\underbrace{0}_{2^4}\underbrace{1}_{2^3}\underbrace{1}_{2^2}\underbrace{0}_{2^1}\underbrace{1}_{2^0}$$

$$2^3 + 2^2 + 2^0 = 8 + 4 + 1 = 13_{10}$$

Desimal ke Binary

$$
\begin{aligned}
13 &= 6x2 + 1 \\
6 &= 3x2 + 0 \\
3 &= 1x2 + 1 \\
1 &= 0x2 + 1 \qquad \uparrow
\end{aligned}
$$

Dibaca dari yang paling bawah : $13_{10} = 1101_2$
atau...

Desimal ke Binary

Nilai 2^n	Desimal	Nilai 2^n	Desimal
2^0	1	2^4	16
2^1	2	2^5	32
2^2	4	2^6	64
2^3	8	2^7	128

$$13 - \boxed{8} = 5 \quad ; 2^3$$
$$5 - \boxed{4} = 1 \quad ; 2^2$$
$$1 - \boxed{1} = 0 \quad ; 2^0$$

Dengan kata lain $13_{10} = 2^3 + 2^2 + 2^0 = \underbrace{1}_{2^3}\underbrace{1}_{2^2}\underbrace{0}_{2^1}\underbrace{1}_{2^0} = 1101_2$

Representasi Bilangan Negatif

Signed Magnitude
Dalam metode ini, digit paling kiri digunakan sebagai penanda positif dan negatif. (positif sebagai "0" dan negatif sebagai "1")

Contoh :
$00001101 = 13$
$10001101 = -13$

Representasi Bilangan Negatif

Signed Magnitude
Dalam metode ini, digit paling kiri digunakan sebagai penanda positif dan negatif. (positif sebagai "0" dan negatif sebagai "1")

Contoh :
$00001101 = 13$
$10001101 = -13$

Kekurangan : tidak mendukung operasi aritmatika pada bilangan biner

Representasi Bilangan Negatif

Signed Magnitude
Dalam metode ini, digit paling kiri digunakan sebagai penanda positif dan negatif. (positif sebagai "0" dan negatif sebagai "1")

Contoh :
$00001101 = 13$
$10001101 = -13$

Kekurangan : tidak mendukung operasi aritmatika pada bilangan biner

$$\begin{array}{rr} 00001101 & 13 \\ 10001101 & -13 \quad + \\ \hline 10011010 & -26 \end{array}$$

Representasi Bilangan Negatif

Komplemen 1 (*One's Complement*)
Dalam metode ini, negatif dari suatu bilangan dinyatakan dengan cara membalik seluruh digitnya, "0" menjadi "1" dan sebaliknya, "1" menjadi "0"

Contoh :
$00001101 = 13$
$11110010 = -13$

Representasi Bilangan Negatif

Komplemen 1 (*One's Complement*)
Dalam metode ini, negatif dari suatu bilangan dinyatakan dengan cara membalik seluruh digitnya, "0" menjadi "1" dan sebaliknya, "1" menjadi "0"

Contoh :
$00001101 = 13$
$11110010 = -13$

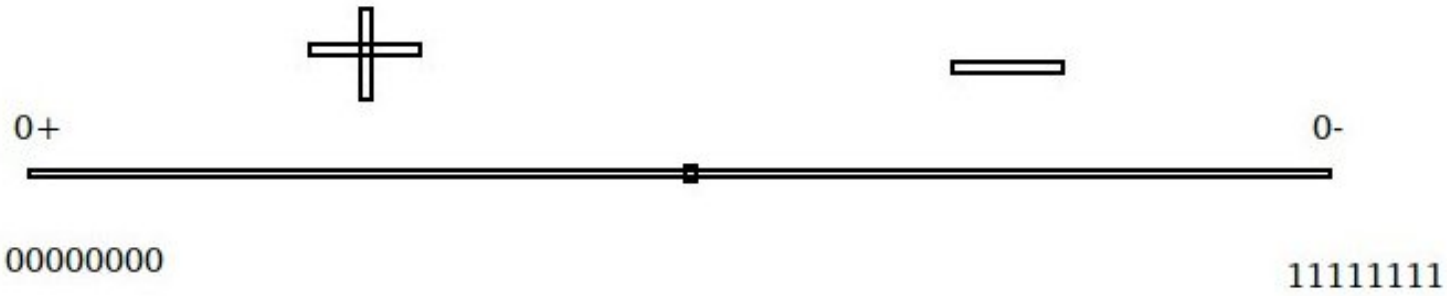
Jika diperhatikan lebih jauh,
$-13 = 11110010 = 242$
$242 + 13 = 255$ (bilangan tertinggi dalam 8 bit)

Jadi komplemen dari 13 adalah 242

Representasi Bilangan Negatif

Komplemen 1 (*One's Complement*)

Pada metode ini terdapat 2 buah nol. Komplemen dari 0^+ (nol positif), 00000000 adalah 0^- (nol negatif), 11111111



Untuk sistem 8 bit, metode ini bisa merepresentasikan -127 s.d. 127

Representasi Bilangan Negatif

Komplemen 1 (*One's Complement*)

Sebagian masalah terpecahkan. Tapi tetap ada masalah.

Coba jumlahkan 5 dan -4

$$\begin{array}{ll} 00000101 & 5 \\ 11111011 & -4 \quad + \\ \hline \end{array}$$

Representasi Bilangan Negatif

Komplemen 1 (*One's Complement*)

Sebagian masalah terpecahkan. Tapi tetap ada masalah.

Coba jumlahkan 5 dan -4

$$\begin{array}{rr} 00000101 & 5 \\ 11111011 & -4 \quad + \\ \hline 1.00000000 & 0 \end{array}$$

Seharusnya $5 - 4 = 1$, tapi dari perhitungan hasilnya 0. SALAH

Aturan:
Jika ada nilai pada bit ke-(n+1) setelah **penjumlahan**, maka harus **ditambahkan** nilai 1 pada hasilnya.
Jika ada nilai pada bit ke-(n+1) setelah **pengurangan**, maka harus **dikurangkan** nilai 1 pada hasilnya.

Representasi Bilangan Negatif

Komplemen 2 (*Two's Complement*)
Negatif dari suatu bilangan dinyatakan dengan komplemennya, ditambahkan 1.

Dengan kata lain, "komplemen 2" adalah ("komplemen 1" + 1)

Contoh :
$00001101 = 13$
$11110010 = -13$ (Komplemen 1)

Representasi Bilangan Negatif

Komplemen 2 (*Two's Complement*)
Negatif dari suatu bilangan dinyatakan dengan komplemennya, ditambahkan 1.

Dengan kata lain, "komplemen 2" adalah ("komplemen 1" + 1)

Contoh :
$00001101 = 13$
$11110010 = -13$ (Komplemen 1)

11110010	-13	(Komplemen 1)	
00000001	1	(1 biner)	+
11110011	-13	(Komplemen 2)	

Representasi Bilangan Negatif

Komplemen 2 (*Two's Complement*)

$$\begin{array}{rr} 00000101 & 5 \\ 11111100 & -4 \quad + \\ \hline 100000001 & 1 \end{array}$$

Komplemen 2 menyelesaikan masalah aritmatika binari yang ditemukan pada *signed magnitude* ataupun komplemen 1. Karena itu metode ini yang banyak digunakan pada banyak sistem komputer dewasa ini.

Sistem dengan komplemen 2 sekarang memiliki 1 buah representasi 0. Sistem yang menggunakan 8 bit dapat merepresentasikan bilangan -128 s.d. 127.

Representasi Bilangan Negatif

Excess-K :
Modifikasi dari Two's Complement
Bilangan K disebut nilai bias
K (dalam binari Two's Complement) dianggap sebagai angka 0
-K diwakili oleh bilangan yang semua bitnya 0

Digunakan terutama untuk eksponen dari titik kambang :
Single precision (32 bit) = 8 bit excess-127
Double precision (64 bit) = 11 bit excess-1023

Representasi Bilangan Negatif

Contoh :
Excess-128 (8 bit)

binari	unsigned	two's complement	excess-128
00000000	0	0	-128
00000001	1	1	-127
...	...	...	...
01111111	127	127	-1
10000000	128	-127	0
10000001	129	-126	1
...	...	...	...
11111110	254	-2	126
11111111	255	-1	127

Representasi Bilangan Negatif

Basis (-2)
Sama seperti perhitungan biner, tapi basisnya bukan 2, melainkan -2
Jadi posisi paling kanan (bit ke-0) mewakili $-2^0 = 1$
Bit ke-1 mewakili $-2^1 = -2$
Bit ke-i mewakili -2^i

Contoh :
00001001 = -7
00000110 = 2

Representasi Bilangan Real

Pada komputer, bilangan real dituliskan dalam bit juga. Untuk mempermudah dalam penalaran, sebagai contoh akan digunakan desimal (basis 10) terlebih dahulu.

Jadi tiap digit dalam komputer diasumsikan dapat diisi angka $0, 1, 2, ...,$ atau 9, seperti format bilangan yang biasa kita gunakan.

Representasi Bilangan Real

$64636,8968$ bisa dituliskan dalam bentuk

$$\begin{aligned} 64,6368968 &\times 10^3, \\ 646,368968 &\times 10^2, \\ 0,646368968 &\times 10^5, \\ &\text{atau lainnya} \end{aligned}$$

Bentuk $0,646368968 \times 10^5$ disebut sebagai bentuk ternormalisasi atau bentuk normal

Representasi Bilangan Real

Bilangan real $254,11377$ dapat dituliskan dalam bentuk normal sebagai

$$0,25411377 \times 10^3$$

Secara umum, bilangan real a dapat dituliskan dalam bentuk

$$a = \pm m \times B^p$$

dimana
m adalah mantisa (nilai "$0, d_1 d_2 d_3 ...$" dari bilangan real),
B adalah basis, dan
p adalah nilai pangkat dari basisnya

Dalam bentuk normal, digit pertama dari mantisa tidak boleh nol

Representasi Bilangan Real

Dalam komputer, bilangan real disimpan dalam bentuk titik-kambang (*floating point*).
Penyimpanannya dalam bentuk data dibagi menjadi 3 bagian, yaitu "tanda", "pangkat", dan "mantisa"

tanda (+ atau -)
pangkat (p)
mantisa (m) = angka-angka di belakang koma

$254,11377 = 0,25411377 \times 10^3$ di dalam *floating point* dituliskan sebagai

+	3	25411377

Representasi Bilangan Real

Contoh Lain :

$354860,206 = 0,345860206 \times 10^6$ di dalam bentuk *floating point* dituliskan sebagai

$+$	6	345860206

$0,004849458 = 0,4849458 \times 10^{-2}$ di dalam bentuk *floating point* dituliskan sebagai

$+$	-2	4849458

$-29,7767025 = -0,297767025 \times 10^2$ di dalam bentuk *floating point* dituliskan sebagai

$-$	2	297767025

Representasi Bilangan Real

Dalam komputer, penyimpanan data tidak dapat menggunakan desimal, tetapi binari. Prinsip yang digunakan sama.

Contoh : bilangan $13,625$ diubah dalam bentuk binari

$$\begin{aligned} 13,625_{10} &= 8+4+1+0,5+0,125 \\ &= 2^3+2^2+2^0+2^{-1}+2^{-3} \\ &= 1101,101_2 \\ &= 0,1101101 \times 2^4 \end{aligned}$$

dituliskan dalam bentuk *floating point* sebagai

$+$	100	1101101

Representasi Bilangan Real

Pada prakteknya, modifikasi dilakukan untuk mempermudah perhitungan dan menghemat tempat penyimpanan (*storage*) pada memori. Misalnya :

- Tanda $+$ disimbolkan dengan 0 dan $-$ disimbolkan dengan 1
- Nilai pangkat positif dan negatif disimbolkan dengan binari (misal 3 disimbolkan dengan 00000010 dan -3 disimbolkan dengan 11111110)
- Dalam bentuk normal, digit pertama pasti 1. Digit ini tidak disimpan untuk menghemat 1 bit penyimpanan.

Tetapi pada kuliah ini, kita tetap menggunakan tanda $+$ dan $-$, serta semua mantisa dituliskan. Sama seperti contoh.

Representasi Bilangan Real

Nilai 2^n	Desimal	Nilai 2^n	Desimal
2^0	1	2^{-4}	0,0625
2^{-1}	0,5	2^{-5}	0,03125
2^{-2}	0,25	2^{-6}	0,015625
2^{-3}	0,125	2^{-7}	0,0078125

Representasi Bilangan Real

Contoh : bilangan $5,7890625$ diubah dalam bentuk binari

$$
\begin{aligned}
5,7890625 - \boxed{4} &= 1,7890625 && ;2^2 \\
1,7890625 - \boxed{1} &= 0,7890625 && ;2^0 \\
0,7890625 - \boxed{0,5} &= 0,2890625 && ;2^{-1} \\
0,2890625 - \boxed{0,25} &= 0,0390625 && ;2^{-2} \\
0,0390625 - \boxed{0,03125} &= 0,0078125 && ;2^{-5} \\
0,0078125 - \boxed{0,0078125} &= 0 && ;2^{-7}
\end{aligned}
$$

$$
\begin{aligned}
5,7890625_{10} &= 2^2 + 2^0 + 2^{-1} + 2^{-2} + 2^{-5} + 2^{-7} \\
&= 101,1100101_2 = 0,1011100101 \times 2^3
\end{aligned}
$$

Dituliskan dalam bentuk *floating point* sebagai

$+$	11	1011100101

Representasi Bilangan Real

Contoh : bilangan $-7,6875$ diubah dalam bentuk binari

$$
\begin{aligned}
7,6875 - \boxed{4} &= 3,6875 && ;2^2 \\
3,6875 - \boxed{2} &= 1,6875 && ;2^1 \\
1,6875 - \boxed{1} &= 0,6875 && ;2^0 \\
0,6875 - \boxed{0,5} &= 0,1875 && ;2^{-1} \\
0,1875 - \boxed{0,125} &= 0,0625 && ;2^{-3} \\
0,0625 - \boxed{0,0625} &= 0 && ;2^{-4}
\end{aligned}
$$

$$
\begin{aligned}
-7,6875_{10} &= -(2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4}) \\
&= -111,1011_2 = -0,1111011 \times 2^3
\end{aligned}
$$

Dituliskan dalam bentuk *floating point* sebagai

$-$	11	1111011

Representasi Bilangan Real

Contoh : bilangan $0,28125$ diubah dalam bentuk binari

$$0,28125 - \boxed{0,25} = 0,03125 \quad ;2^{-2}$$
$$0,03125 - \boxed{0,03125} = 0 \quad ;2^{-5}$$

$$0,28125_{10} = 2^{-2} + 2^{-5}$$
$$= 0,01001_2 = 0,1001 \times 2^{-1}$$

Dituliskan dalam bentuk *floating point* sebagai

$+$	-1	1001

Representasi Bilangan Real

Banyak digit yang digunakan masing-masing bagian dalam *floating point* tergantung kepada jenis komputer (*hardware*) dan kompiler yang dipakai.

Menurut IEEE-754 (sebuah standard teknis untuk komputasi *floating point*), format yang digunakan dalam basis 2 adalah

Nama	Nama Umum	Bit Mantisa	Bit Pangkat	Digit Desimal	Pangkat Desimal
binary16	Half Precision	10	5 $-14 \sim 15$	3,31	4,51
binary32	Single Precision	23	8 $-126 \sim 127$	7,22	38,23
binary64	Double Precision	52	11 $-1022 \sim 1023$	15,95	307,95
binary128	Quadruple Precision	112	15	34,02	4931,77

Representasi Bilangan Real

Bit-bit yang digunakan untuk mantisa dan pangkat dapat diperluas sesuai kebutuhan

- Semakin tinggi bit pangkatnya, semakin tinggi bilangan yang bisa diwakilkan
- Semakin tinggi bit mantisanya, semakin teliti bilangan tersebut diwakilkan (presisinya semakin tinggi)

Float Converter

Konverter berikut dapat digunakan untuk meningkatkan pemahaman bagaimana sebuah bilangan dikonversi dalam bit dan seberapa besar error yang dihasilkannya.

https://www.h-schmidt.net/FloatConverter/IEEE754.html

Resume

1. Bilangan dituliskan pada komputer dalam bentuk biner (*binary*)
2. Bilangan bulat (integer) dituliskan sesuai nilai binernya
3. Bilangan rill (real) dituliskan dalam bentuk ternormalisasi

$$a = \pm m \times B^p$$

dan disimpan dalam bentuk titik kambang (*floating point*)

$\pm$	p	m

Aritmatika Pada Floating Point

Bagaimana menjumlahkan $15,912$ dan $1131,8$?

Aritmatika Pada Floating Point

Bagaimana menjumlahkan $15,912$ dan $1131,8$?

Keduanya direpresentasikan dalam bentuk ternormalisasi

$$\begin{array}{ll} 0,15912 & \times 10^{2} \\ 0,11318 & \times 10^{4} \; + \\ \hline \end{array}$$

Aritmatika Pada Floating Point

Bagaimana menjumlahkan $15,912$ dan $1131,8$?

Keduanya direpresentasikan dalam bentuk ternormalisasi

$$\begin{array}{ll} 0,15912 & \times 10^2 \\ 0,11318 & \times 10^4 \; + \\ \hline \end{array}$$

Sebelum dijumlahkan, perlu dilakukan pergeseran digit untuk menyamakan pangkat

$$\begin{array}{ll} 0,0015912 & \times 10^4 \\ 0,11318 & \times 10^4 \; + \\ \hline \end{array}$$

Aritmatika Pada Floating Point

Bagaimana menjumlahkan $15,912$ dan $1131,8$?

Keduanya direpresentasikan dalam bentuk ternormalisasi

$$\begin{array}{ll} 0,15912 & \times 10^2 \\ 0,11318 & \times 10^4 \; + \\ \hline \end{array}$$

Sebelum dijumlahkan, perlu dilakukan pergeseran digit untuk menyamakan pangkat

$$\begin{array}{ll} 0,0015912 & \times 10^4 \\ 0,11318 & \times 10^4 \; + \\ \hline 0,1147712 & \times 10^4 \end{array}$$

Aritmatika Pada Floating Point

Jika digunakan komputer dengan mantis 5 digit (basis 10), maka hasil penjumlahannya akan terkena galat pembulatan di digit ke-6 dan seterusnya.

$$0,1147712 \times 10^4 \longrightarrow 0,11477 \times 10^4 + 0,0000012 \times 10^4$$

Yang akan disimpan adalah $0,11477 \times 10^4$
dengan galat pembulatan sebesar $\epsilon = 0,0000012 \times 10^4$

Aritmatika Pada Floating Point

Bagaimana menjumlahkan $3542,99911123$ dan $0,00088877$?

Aritmatika Pada Floating Point

Bagaimana menjumlahkan $3542,99911123$ dan $0,00088877$?

Keduanya direpresentasikan dalam bentuk ternormalisasi

$$\begin{array}{rll} 0,354299911123 & \times 10^{4} & \\ 0,88877 & \times 10^{-3} & + \\ \hline ??? & ?? & \end{array}$$

Aritmatika Pada Floating Point

Kalau digunakan komputer dengan mantis 5 digit (basis 10), terdapat galat pembulatan di digit ke-6 dan seterusnya.

Pembulatannya bisa berupa :

1. Pemenggalan (*chopping*)
   Digit yang tidak dapat disimpan dalam memori dihilangkan sama sekali
2. Pembulatan (*in-rounding*)
   Digit yang tidak dapat disimpan dalam memori dibulatkan ke digit terdekat
   $0, 1, 2, 3, 4$ dibulatkan ke bawah pada digit sebelah kirinya
   $5, 6, 7, 8, 9$ dibulatkan ke atas pada digit sebelah kirinya

Aritmatika Pada Floating Point

Pembulatan dengan *chopping*

$$\begin{array}{rll} 0,35429 & \times 10^4 & \\ 0,88877 & \times 10^{-3} & + \\ \hline ??? & ?? & \end{array}$$

Dan pada saat pergeseran digit untuk menyamakan pangkat, bilangan yang kecil tidak masuk ke dalam perhitungan

$$\begin{array}{rll} 0,35429 & \times 10^4 & \\ 0,000000088877 & \times 10^4 & + \\ \hline 0,354290088877 & \times 10^4 & \end{array}$$

Galat pembulatannya $0,000009911123 \times 10^4$ dari penyimpanan bilangan pertama dan $0,000000088877 \times 10^4$ dari hasil penjumlahannya.

Aritmatika Pada Floating Point

Galat pembulatannya $0,000009911123 \times 10^4$ dari penyimpanan bilangan pertama dan $0,000000088877 \times 10^4$ dari hasil penjumlahannya.

Total galatnya $0,00001 \times 10^4$

Aritmatika Pada Floating Point

Pembulatan dengan *in-rounding*

$0,354299911123 \times 10^4$ dibulatkan menjadi $0,35430 \times 10^4$

$$\begin{array}{ll} 0,35430 & \times 10^4 \\ 0,88877 & \times 10^{-3} \\ \hline ??? & ?? \end{array} +$$

Pada saat pergeseran digit untuk menyamakan pangkat, bilangan yang kecil tidak masuk ke dalam perhitungan

$$\begin{array}{ll} 0,35430 & \times 10^4 \\ 0,000000088877 & \times 10^4 \\ \hline 0,354300088877 & \times 10^4 \end{array} +$$

Galat pembulatannya $-0,000000088877 \times 10^4$ dari penyimpanan bilangan pertama dan $0,000000088877 \times 10^4$ dari penjumlahannya.

Aritmatika Pada Floating Point

Galat pembulatannya $-0,000000088877 \times 10^4$ dari penyimpanan bilangan pertama dan $0,000000088877 \times 10^4$ dari penjumlahannya.

Total galatnya $0,0 \times 10^4$.
Dengan kata lain, nilai yang diperoleh sama persis dengan nilai yang sebenarnya.

Epsilon Mesin

Jumlah bit yang digunakan untuk representasi bilangan rill terbatas, berarti banyaknya bilangan rill yang bisa direpresentasikan juga terbatas. (seperti kasus bilangan bulat 8 bit hanya dapat merepresentasikan bilangan -128 s.d. 127)

Jika kita punya sistem bilangan *floating point* dengan 6 bit (1 bit untuk tanda, 3 bit untuk pangkat, dan 2 bit mantisa) dan pangkat yang mungkin dari -2 s.d. 3, bilangan yang mungkin dibentuk adalah

$$\pm 0.10 \times 2^p \text{ atau } \pm 0.11 \times 2^p, \text{ dimana } -2 \leq p \leq 3$$

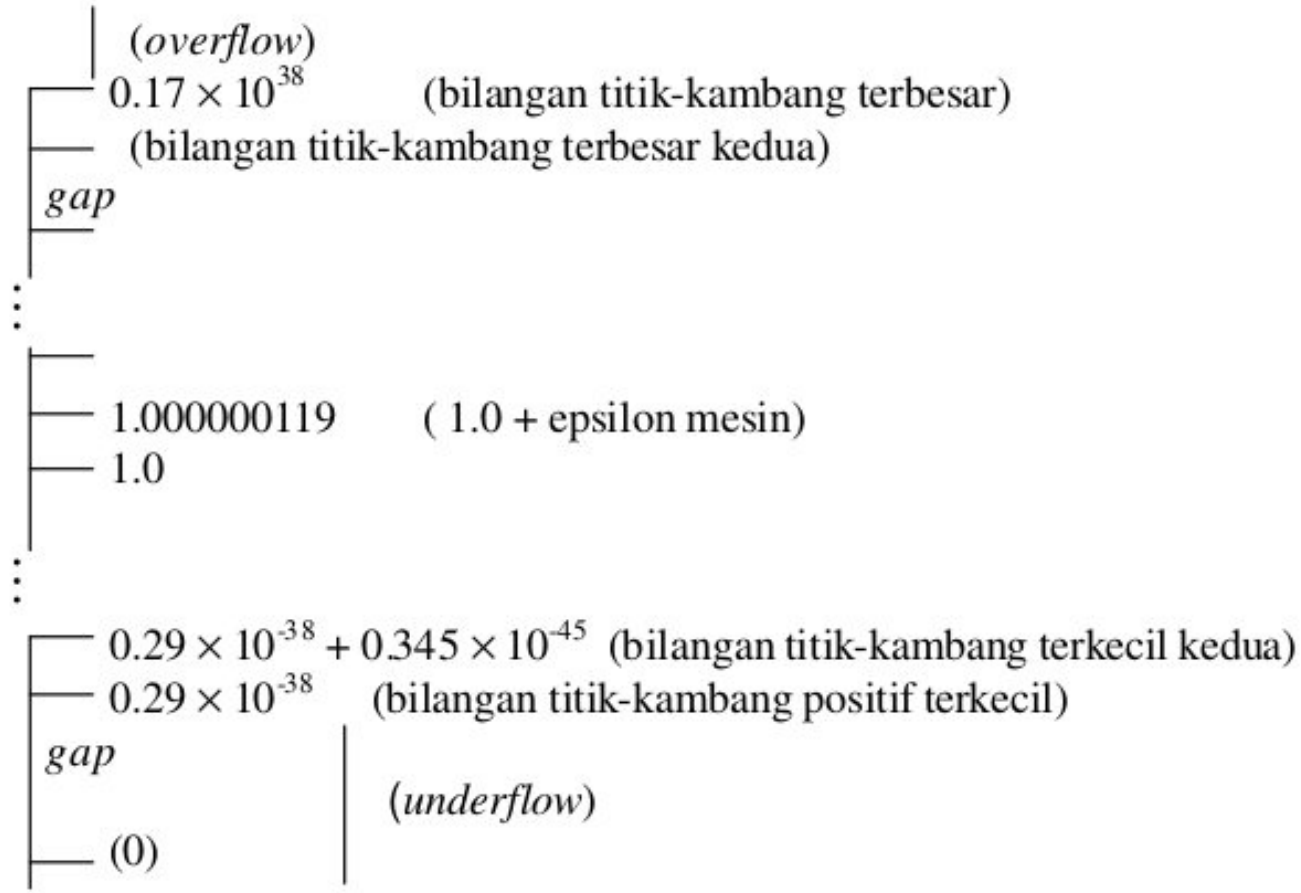
Epsilon Mesin

$$\pm 0.10 \times 2^p \text{ atau } \pm 0.11 \times 2^p, \text{ dimana } -2 \leq p \leq 3$$

$$0.10_2 \times 2^{-2} = (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^{-2} = 1/8 = 0.125_{10}$$
$$0.10_2 \times 2^{-1} = (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^{-1} = 1/4 = 0.25_{10}$$
$$0.10_2 \times 2^{0} = (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^{0} = 1/2 = 0.5_{10}$$
$$0.10_2 \times 2^{1} = (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^{1} = 1 = 1.0_{10}$$
$$0.10_2 \times 2^{2} = (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^{2} = 2 = 2.0_{10}$$
$$0.10_2 \times 2^{3} = (1 \times 2^{-1} + 0 \times 2^{-2}) \times 2^{3} = 4 = 4.0_{10}$$
$$0.11_2 \times 2^{-2} = (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^{-2} = 3/16 = 0.1875_{10}$$
$$0.11_2 \times 2^{-1} = (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^{-1} = 3/8 = 0.375_{10}$$
$$0.11_2 \times 2^{0} = (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^{0} = 3/4 = 0.75_{10}$$
$$0.11_2 \times 2^{1} = (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^{1} = 3/2 = 1.5_{10}$$
$$0.11_2 \times 2^{2} = (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^{2} = 3 = 3.0_{10}$$
$$0.11_2 \times 2^{3} = (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^{3} = 6 = 6.0_{10}$$

Epsilon Mesin

Floating point berketelitian tunggal (*float*) dinyatakan dalam 32-bit (1 bit tanda, 8 bit pangkat, dan 23 bit mantisa)



Epsilon Mesin

Epsilon mesin adalah selisih dua buah nilai sangat kecil yang masih dapat dikenali oleh komputer sehingga nilai yang lebih kecil dari itu dianggap nol.

Distandardisasi sebagai bilangan terkecil yang jika ditambahkan dengan 1 memberikan hasil yang lebih besar dari 1.

$$1 + \epsilon > 1$$

Dalam sistem 32-bit di slide sebelumnya, nilai epsilon mesin adalah $0,000000119$, yaitu $0,119 \times 10^{-6}$

Epsilon Mesin

Nilai epsilon mesin dapat berbeda-beda tergantung bahasa pemprograman dan komputer yang digunakan. Juga tergantung tipe datanya.

Terkadang ada beberapa bahasa yang menggunakan bilangan berketelitian ganda (*double precision* atau *double*)

Epsilon mesin dimanfaatkan sebagai kriteria henti kekonvergenan pada iterasi. Jika galat relatif sudah lebih kecil dari epsilon, iterasi diberhentikan (perhitungan sudah konvergen), jika tidak, iterasi diteruskan.

Epsilon Mesin

Cara mencari epsilon mesin

```
procedure HitungEpsilonMesin1(var eps : real);
{ Prosedur untuk menemukan epsilon mesin
   Keadaan Awal : sembarang
  Keadaan Akhir: eps berisi harga epsilon mesin
}
begin
   eps:=1;
   while eps + 1 > 1 do
      eps:=eps/2;
   {eps + 1 < 1}

   eps:=2*eps;          {nilai epsilon mesin}
end;
```

Tugas : Epsilon Mesin

Buat sebuah program (boleh berbentuk skrip python, C++, java, atau yang lainnya) untuk mencari epsilon mesin.