



Arsitektur dan Organisasi Komputer

6-2

Aditya Wikan M, S.Kom & Samuel Gandang G, S.Kom, S.Si

Week 10



Computer Arithmetic (2) Operasi Pecahan [Floating Point Operation]

Representasi Pecahan (1)



- Bilangan real adalah bilangan yang mengandung pecahan
- Secara matematis, dapat dinyatakan dengan bilangan biner murni:
 - $1001.1010 = 2^3 + 2^0 + 2^{-1} + 2^{-3} = 9.625$
- Masalah yang timbul di komputer: bagaimana menyimpan tanda koma (point).
- Fixed Point (tanda koma selalu berada pada batas digit yang sama) → terlalu terbatas
- Moving Point (bebas bergerak) → susah menunjukkan sekarang point ada di mana

Representasi Pecahan (2)



- Digunakanlah sistem **floating point**
- Floating point berarti koma (point) dapat digeser sesuai kebutuhan, dan menggunakan eksponen n untuk menunjukkan posisi point
- Contoh: *decimal point*: dengan eksponen 10
 $976,000,000,000,000 = 9.76 \times 10^{14}$
 $0.00000000000000976 = 9.76 \times 10^{-14}$
- Keuntungan: representasi bit dapat dibagi dalam proporsi yang sama untuk bilangan apapun: $\pm S \times B^{\pm E}$

Representasi Pecahan (3)



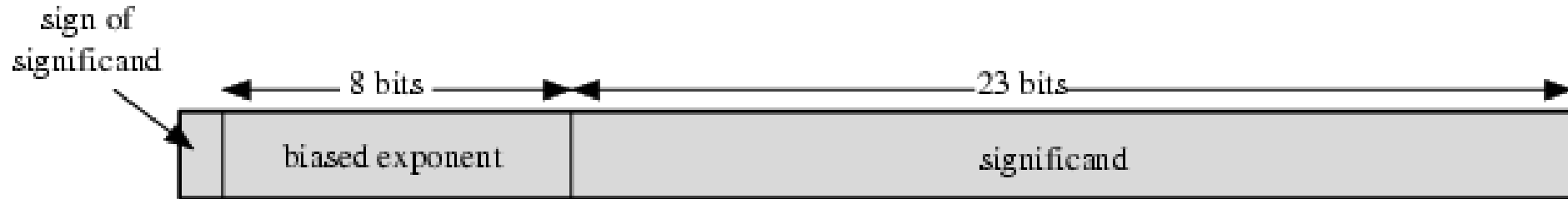
Floating point biner:

Sign bit	Biased Exponent	Significand or Mantissa
----------	-----------------	-------------------------

- Mewakili bentuk $\pm \text{.significand} \times 2^{\text{exponent}}$
- Eksponen disimpan dalam bentuk terbias
Bentuk terbias = eksponen asli + B (bilangan bias)

Representasi Pecahan (4)

Representasi floating point biner 32-bit



(a) Format

$$\begin{aligned}
 1.1010001 \times 2^{10100} &= 0 \ 10010011 \ 101000100000000000000000 = 1.638125 \times 2^{20} \\
 -1.1010001 \times 2^{10100} &= 1 \ 10010011 \ 101000100000000000000000 = -1.638125 \times 2^{20} \\
 1.1010001 \times 2^{-10100} &= 0 \ 01101011 \ 101000100000000000000000 = 1.638125 \times 2^{-20} \\
 -1.1010001 \times 2^{-10100} &= 1 \ 01101011 \ 101000100000000000000000 = -1.638125 \times 2^{-20}
 \end{aligned}$$

(b) Examples

Floating Point Biner

- MSB dipakai untuk tanda seluruh bilangan, 0: positif, 1: negatif
- Significand S disimpan dalam bentuk biner biasa, tetapi diletakkan dari KIRI
- Eksponen E disimpan dalam notasi terbias (excess / kelebihan). Untuk lebar eksponen k-bit, maka bilangan biasanya = $2^{k-1} - 1$.

Misal lebar eksponen 8-bit, maka biasanya = $2^7 - 1 = 128 - 1 = 127$ (biner: 01111111)

2^{10100} eksponennya akan disimpan sebagai
 $00010100 + 01111111 = 10010011$

Normalisasi

- Sebelum disimpan ke dalam format floating point, suatu bilangan harus dinormalisasi
- Normalisasi: dibuat menjadi bentuk biner
$$\pm 1.bbb \dots b \times 2^{\pm E}$$
- (c.f. Scientific notation where numbers are normalized to give a single digit before the decimal point e.g 3.123×10^3)
- Otomatis eksponen harus diubah menyesuaikan normalisasi
- Keuntungan: karena pasti ada angka 1 didepan point, angka itu tidak perlu disimpan

Jangkauan (Range) Floating Point

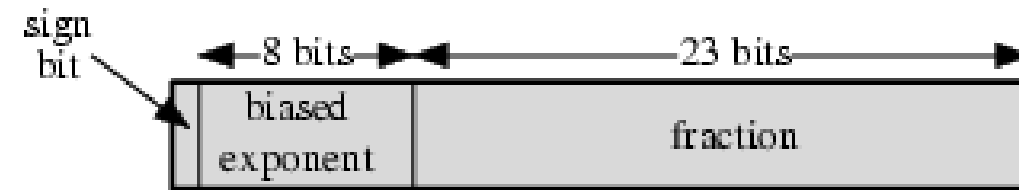
- Untuk representasi floating point 32 bit
 - 8 bit eksponen
 - +/- $2^{256} \approx 1.5 \times 10^{77}$
- Akurasi
 - Berubah mengikuti perubahan LSB's mantissa
 - 23 bit mantissa $2^{-23} \approx 1.2 \times 10^{-7}$
 - Sekitar 6 digit di belakang koma

Standarisasi IEEE 754

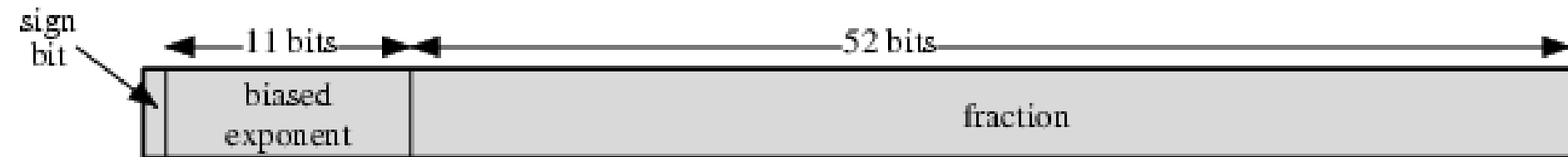
- Standard for floating point storage
- 32 and 64 bit standards
- 8 and 11 bit exponent respectively
 - Bias: 127 dan 1023
- 32 bit dinamakan SINGLE precision
- 64 bit dinamakan DOUBLE precision



Format IEEE 754



(a) Single format



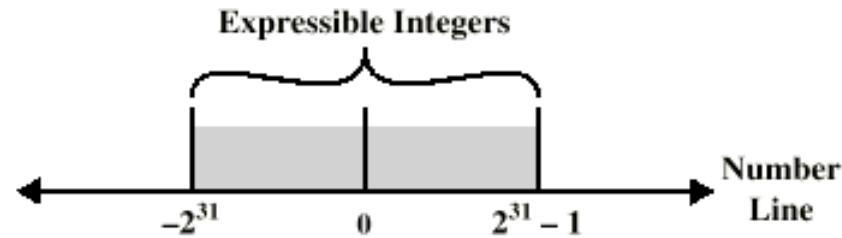
(b) Double format

Aritmetik Pecahan – Overflow

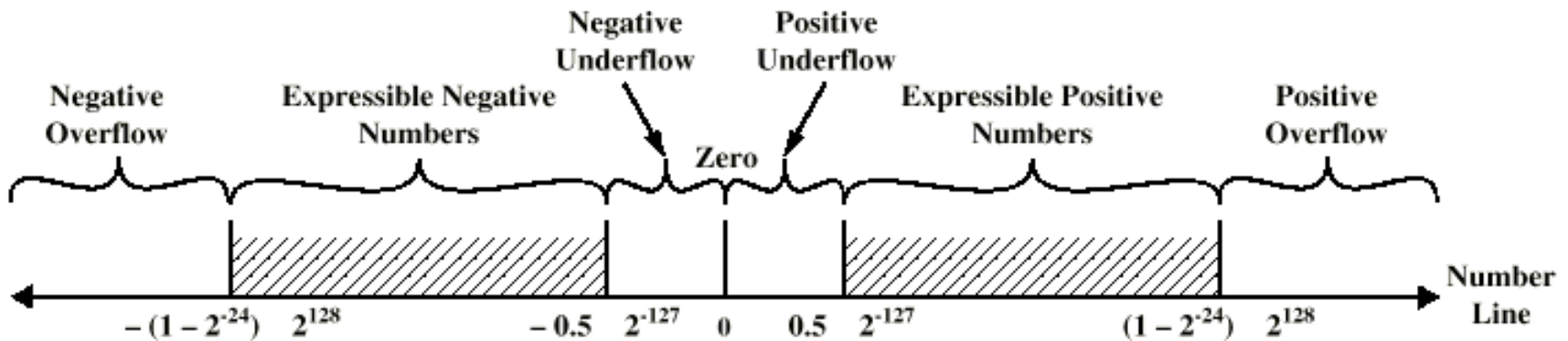
- Dalam floating point, tidak hanya dikenal overflow, tapi juga UNDERFLOW
- Overflow → sama seperti pada integer
- Underflow → jika sebuah bilangan dalam bentuk pecahan yang terlalu kecil sehingga pangkat (eksponen) –nya tidak tertampung dalam format floating point tertentu (misal untuk format 32-bit: 1.1101×2^{-140})
- Biasanya dibulatkan menjadi NOL → bilangan yang sangat kecil kan tidak penting



Aritmetik Pecahan – Overflow dan Underflow



(a) Twos Complement Integers



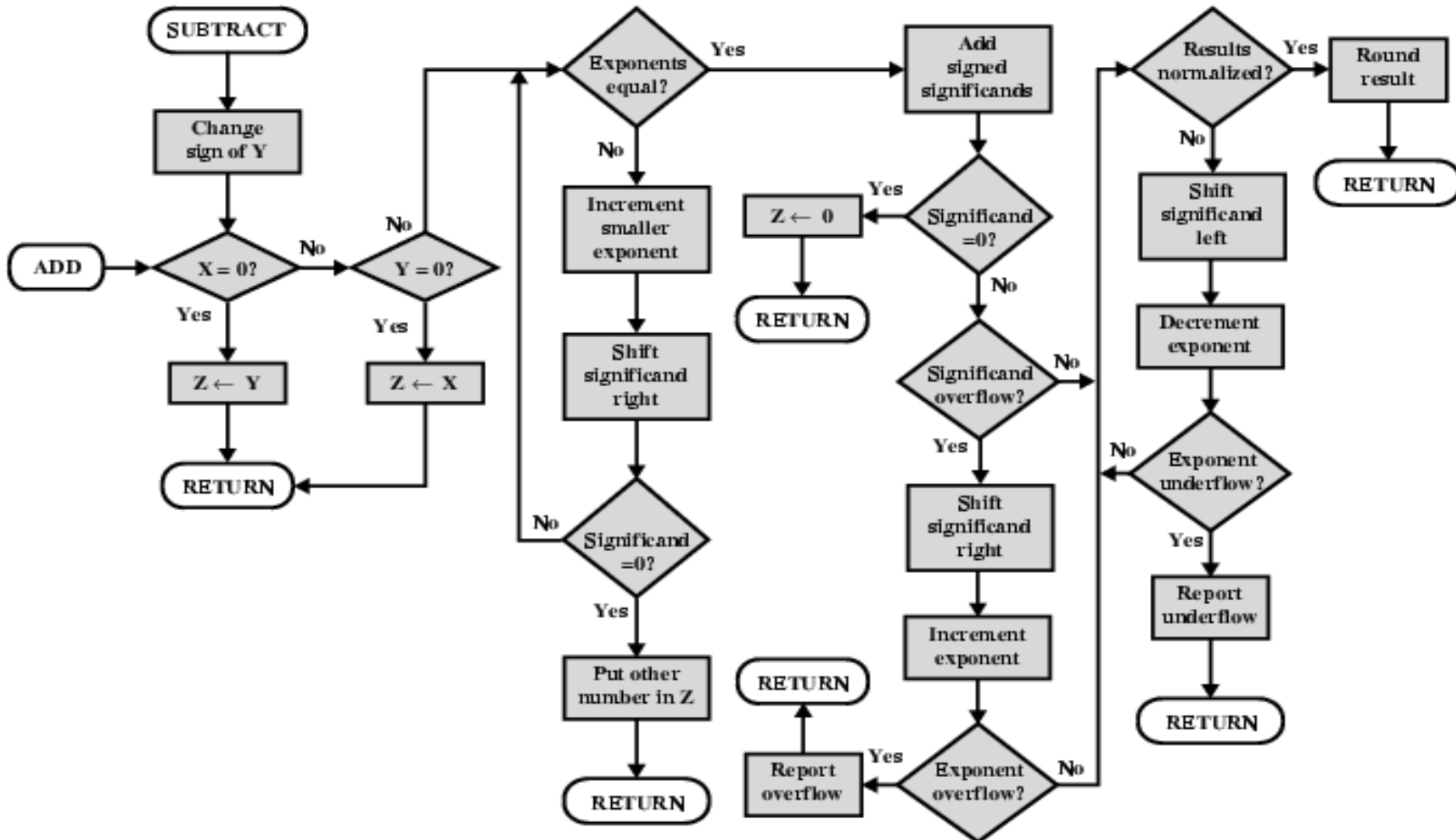
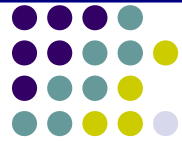
(b) Floating-Point Numbers



Aritmetik Integer – Addition dan Subtraction

- Cek nol:
 - Jika sebuah operand nol, hasil \leftarrow yang lain
 - Cek apakah perlu perubahan tanda (jika pengurangan)
- Sejajarkan significand (mirip dengan penyamaan pangkat pada bilangan desimal)
$$(123 \times 10^0) + (456 \times 10^{-2})$$
$$= (123 \times 10^0) + (4.56 \times 10^0)$$
$$= 127.56 \times 10^0$$
- Jumlahkan kedua significand
- Normalisasi hasilnya

Addition dan Subtraction

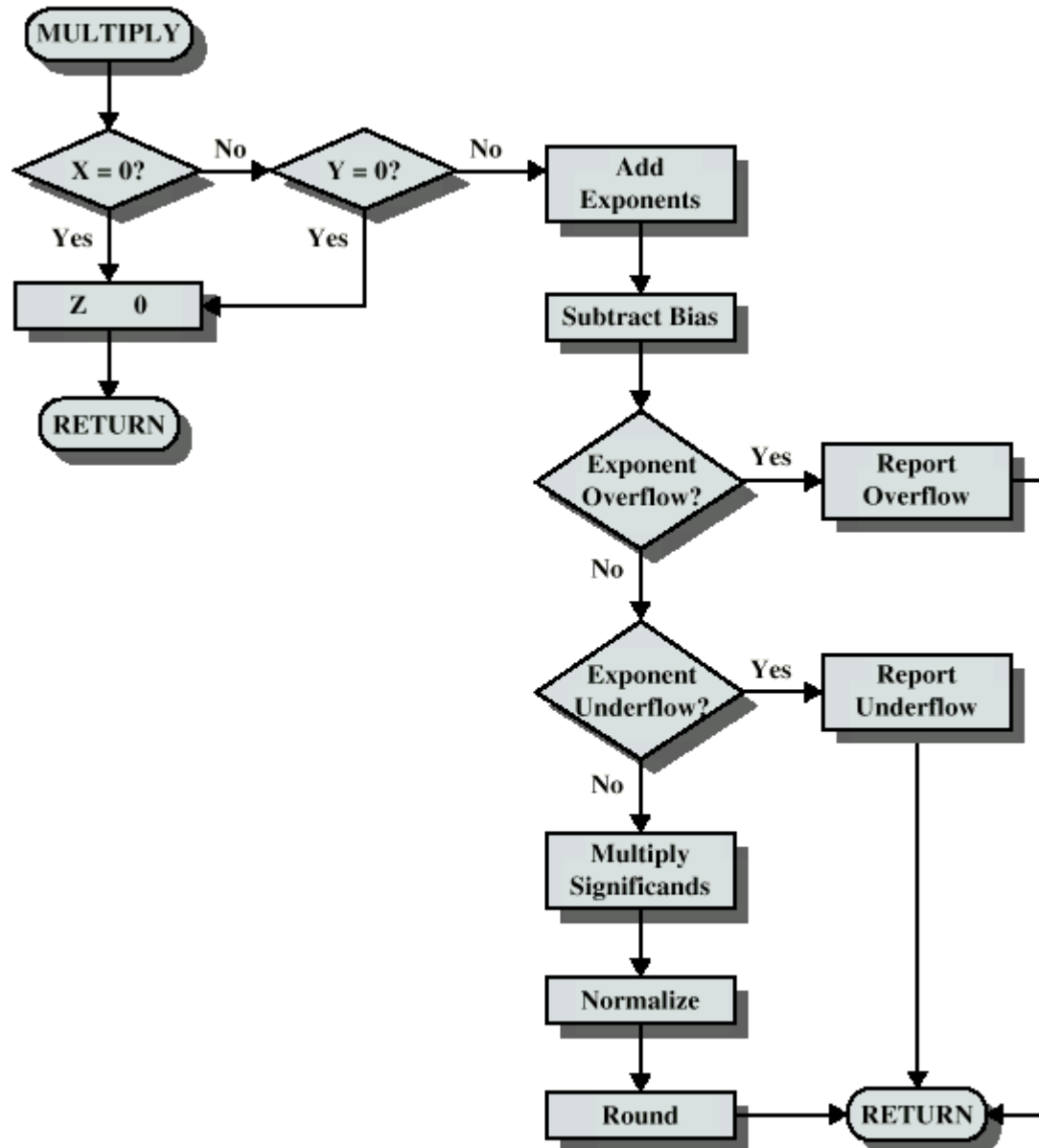


Aritmetik Integer – Multiplication dan Division

- Silakan lihat flowchart supaya lebih jelas
- Alurnya kurang lebih sama
- Menggunakan perkalian dan pembagian biner biasa untuk significand



Multiplication



Division

