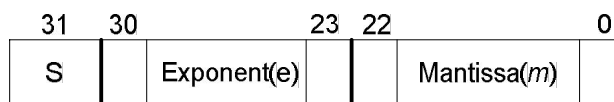


5. Floating Point Arithmetic

5.1. Bentuk Bilangan Floating Point

Bilangan Floating Point memiliki bentuk umum : $\pm m^*b^e$, dimana m (disebut juga dengan *mantissa*), mewakili bilangan pecahan dan umumnya dikonversi ke bilangan binernya, e mewakili bilangan *exponent*-nya, sedangkan b mewakili *radix* (basis) dari *exponent*.



Gambar 5.1.

Contoh :

Pada gambar diatas, menunjukkan tentang panjang bit pada bilangan floating point $m = 23$ bit, $e = 8$ bit, dan S (bit sign) = 1. Jika nilai yang tersimpan di S adalah 0, maka bilangan tersebut adalah positif dan jika nilai yang tersimpan pada S adalah 1, maka bilangan tersebut adalah negatif.

Bilangan *exponent* pada contoh diatas, hanya dapat digunakan pada bilangan positif 0 hingga 255. Untuk dapat menggunakan bilangan *exponent* negatif dan positif, nilai bulat yang disebut dengan *bias*, dikurangkan dengan bilangan pada kolom *exponent* dan menghasilkan bilangan *exponent* akhir. Misalkan pada contoh diatas menggunakan $bias = 128$, maka bilangan *exponent* akhirnya memiliki range antara -128 (disimpan sebagai 0 pada kolom *exponent*) hingga +127 (disimpan sebagai 255 pada kolom *exponent*). Berdasarkan bentuk seperti ini, bilangan *exponent* +4 dapat digunakan dengan menyimpan 132 pada kolom *exponent*, sedangkan bilangan *exponent* -12 dapat digunakan dengan menyimpan 116 pada kolom *exponent*.

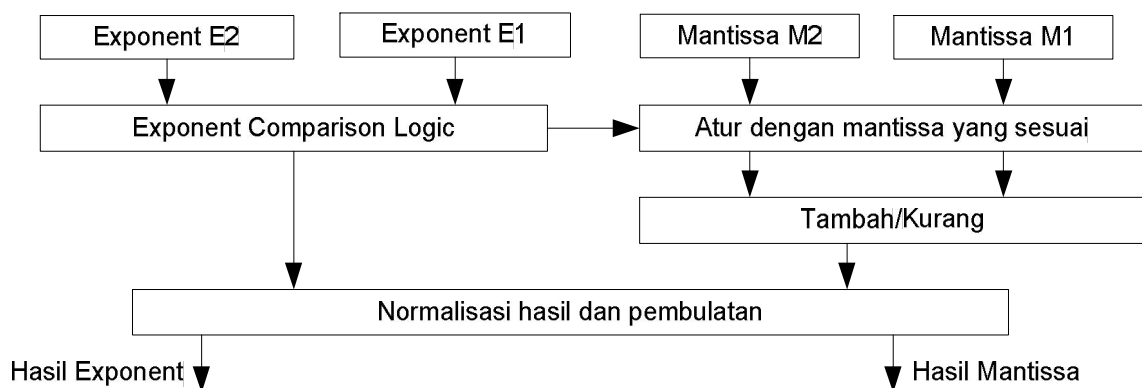
Anggap $b = 2$, maka bilangan floating point seperti 1,75 dapat menggunakan salah satu dari bentuk umum seperti pada gambar berikut:

3. Lakukan normalisasi dengan 'menggeser' nilai *mantissa* dan mengatur nilai eksponensialnya

Contoh : Jumlahkan dua bilangan *floating point* $1,1100 * 2^4$ dan $1,1000 * 2^2$

1. Sesuaikan : $1,1000 * 2^2$ diubah menjadi $0,0110 * 2^4$
2. Jumlahkan : hasil penjumlahan $10,0010 * 2^4$
3. Normalisasi : hasil setelah dinormalisasi adalah $0,1000 * 2^6$ (dianggap bit yang diijinkan setelah koma adalah 4)

Operasi penjumlahan/pengurangan dua bilangan *floating point* diilustrasikan dengan skema seperti pada gambar berikut :



Gambar 5.3. Skema penjumlahan/pengurangan bilangan *floating point*

5.4. Perkalian

Perkalian dari dua bilangan *floating point* dengan bentuk $X = m_x * 2^a$ dan $Y = m_y * 2^b$ setara dengan $X * Y = (m_x * m_y) * 2^{a+b}$

Algoritma umum untuk perkalian dari bilangan *floating point* terdiri dari tiga langkah :

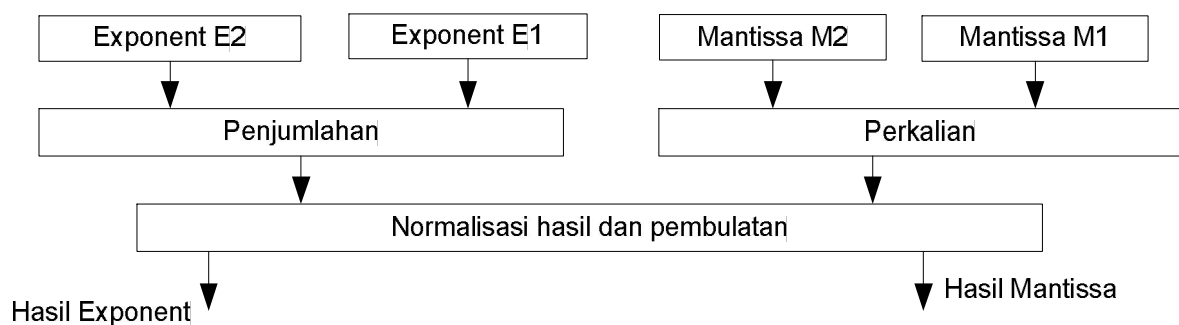
1. Hitung hasil eksponensial dengan menjumlahkan nilai eksponen dari kedua bilangan
2. Kalikan kedua bilangan *mantissa*
3. Normalisasi hasil akhir

Contoh : Perkalian antara dua bilangan *floating point* $X = 1,000 * 2^{-2}$ dan $Y = -1,010 * 2^{-1}$

1. Tambahkan bilangan exponennya : $-2 + (-1) = -3$
2. Kalikan *mantissa*: $1,0000 * -1,010 = -1,010000$

Hasil perkaliannya adalah $-1,0100 * 2^{-3}$

Perkalian dari dua bilangan *floating point* diilustrasikan menggunakan skema seperti tampak pada gambar berikut :



Gambar 5.4. Skema perkalian bilangan *floating point*

5.5. Pembagian

Pembagian dari dua bilangan *floating point* dengan bentuk $X = m_x * 2^a$ dan $Y = m_y * 2^b$ setara dengan $X / Y = (m_x / m_y) * 2^{a-b}$

Algoritma umum untuk pembagian dari bilangan *floating point* terdiri dari tiga langkah :

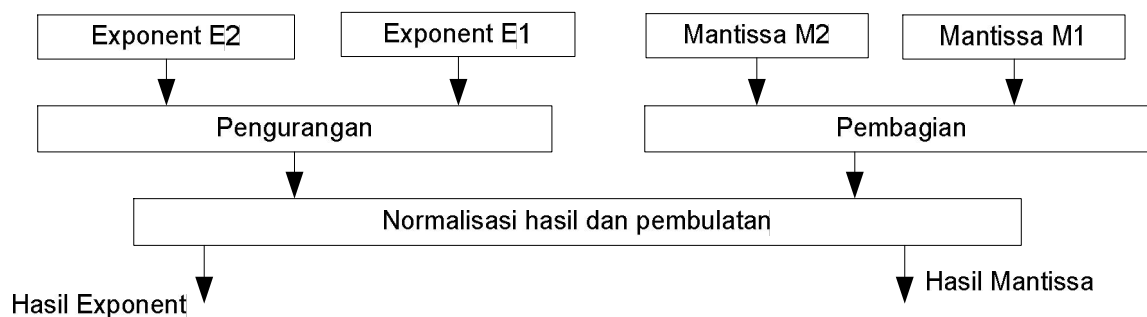
1. Hitung hasil eksponensial dengan mengurangkan nilai exponent dari kedua bilangan
2. Bagi kedua bilangan *mantissa*
3. Normalisasi hasil akhir

Contoh : Pembagian antara dua bilangan *floating point* $X = 1,0000 * 2^{-2}$ dan $Y = -1,0100 * 2^{-1}$

1. Kurangkan bilangan exponennya : $-2 - (-1) = -1$
2. Bagi *mantissa*: $1,0000 / -1,0100 = -0,1101$

Hasil pembagiannya adalah $-0,1101 * 2^{-1}$

Pembagian dari dua bilangan *floating point* diilustrasikan menggunakan skema seperti tampak pada gambar berikut :

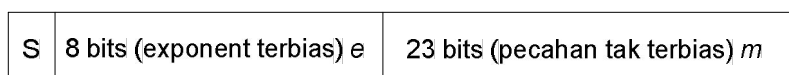


Gambar 5.5. Skema pembagian bilangan *floating point*

5.6. Floating Point standard IEEE

IEEE membuat dua bentuk bilangan *floating point* standard. Bentuk *basic* dan bentuk *extended*. Pada tiap bentuk tersebut, IEEE menentukan dua format, yaitu *single-precision* dan *double precision format*. *Single precision format* adalah model 32bit sedangkan *double precision format* adalah 64bit. Pada *single extended format* setidaknya menggunakan 44 bit, sedangkan pada *double extended format* setidaknya menggunakan 80 bit.

Pada *single precision format*, mengizinkan penggunaan bit tersembunyi, kolom exponentnya adalah 8bit. Bentuk *single precision* ditunjukkan pada gambar berikut.



Gambar 5.6.

IEEE *single precision Format*

Jika jumlah bit bilangan exponent adalah 8, maka nilainya memiliki 256 kombinasi, diantara angka-angka tersebut, dua kombinasi digunakan sebagai nilai khusus:

1. $e = 0$ bernilai *nol* (jika $m = 0$) dan nilai terdenormalisasi (jika $m \neq 0$)

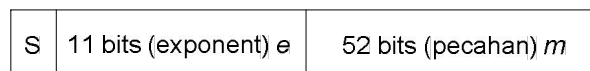
2. $e = 255$ bernilai $\pm \infty$ (jika $m = 0$) dan nilai tak terdefinisi (jika $m \neq 0$)

	$m = 0$	$m \neq 0$
$e = 0$	0	Terdenormalisasi
$e = 255$	$\pm \infty$	Tidak Terdefinisi

Tabel 5.1.

IEEE *Double Precision Format*

Bentuk ini memiliki kolom exponent 11 bit dan kolom nilai *mantissa* sebesar 52 bit. Bentuknya seperti tampak pada gambar.

Gambar 5.7. IEEE *double precision format*

Karakteristik	<i>Single-Precision</i>	<i>Double Precision</i>
Panjang dalam bits	32	64
Bagian pecahan dalam bits	23	52
Bit tersembunyi	1	1
Panjang Exponent dlm bits	8	11
Bias	127	1023
Range	$2^{128} \approx 3,8 \times 10^{38}$	$2^{1024} \approx 9,0 \times 10^{307}$
Nilai ternormalisasi terkecil	$2^{-126} \approx 10^{-38}$	$2^{-1022} \approx 10^{-308}$

Tabel 5.2. Karakteristik dari IEEE *single* dan *double floating point format*