MODUL ONLINE 2
MATA KULIAH ORGANISASI DAN ARSITEKTUR KOMPUTER
KODE MATA KULIAH CC0120

DISUSUN OLEH

NIZIRWAN ANWAR & TEAM

FAKULTAS ILMU KOMPUTER
UNIVERSITAS ESA UNGGUL
JAKARTA
20181

ALU dan Representasi Data

1. Pendahuluan

Kita mulai pemeriksaan kami prosesor dengan gambaran aritmatika dan unit logika (ALU). Materi akan menjelaskanfokus pada aspek paling rumit dari ALU, aritmatika komputer. Fungsi logika yang merupakan bagian dari ALU dijelaskan dalam lebih lanjut pada topik kumpulan instruksi, dan implementasi logika sederhana dan fungsi aritmatika dalam digital logika dijelaskan dalam konsep logika digital. Aritmatika komputer umumnya dilakukan pada 2 (dua) jenis yang sangat berbeda angka: integer dan floating point, dan dilanjutkan pada pembahasan secara mendalam tentang operasi aritmatika.

Point penting yang harus diperhatikan dan dipelajari !!!

(a) Terdapat 2 (dua) fokus utama untuk aritmatika komputer adalah bagaimana caranya merepresentasikan angka diwakili (format biner) dan algoritma yang digunakan untuk operasi aritmatika dasar (menambah, mengurangi, mengalikan, membagi). Hal ini berlaku untuk representasi integer dan floating-point (FP).

(b) floating-point dinyatakan sebagai angka (significant) dikalikan oleh konstanta (basis) yang merupakan bilangan integer (dalam bentuk eksponen). Angka Floating-point digunakan untuk mewakili angka yang sangat besar dan sangat kecil.

(c) Pada umumnya atau sebagian besar prosesor menerapkan standar IEEE 754 untuk representasi floating-point dan aritmatika floating-point. IEEE 754 mendefinisikan keduanya 32-bit dan format 64-bit serta terbaru format 128-bit (pra-konsep)

Review sistem Bilangan

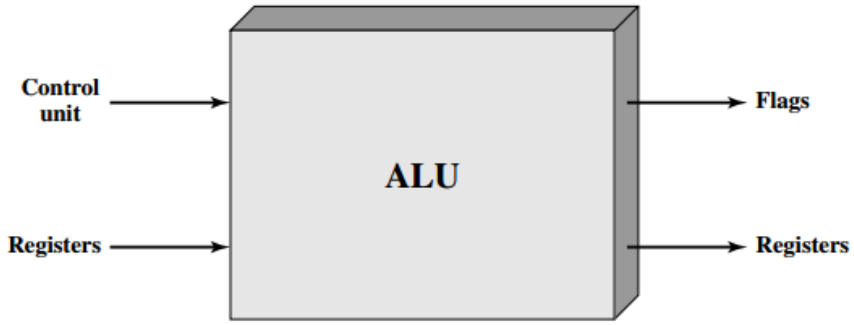
Sistem	Radix (Basis)	Symbol
Biner	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6 dan 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9
Heksadesimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9, A … F

2. Aritmatika Komputer

(1) Aritmatika Logika Unit (ALU)

Arithmatic Logical Unit (ALU), adalah komponen dalam sistem komputer yang berfungsi melakukan operasi perhitungan aritmatika dan logika (Contoh operasi aritmatika adalah operasi penjumlahan dan pengurangan, sedangkan

contoh operasi logika adalah logika AND dan OR. ALU bekerja besama-sama memori, di mana hasil dari perhitungan di dalam ALU di simpan ke dalam memori. Semua elemen lain dari sistem komputer unit kontrol, register, memori, I/O - ada terutama untuk membawa data ke ALU untuk di proses dan kemudian mengambil hasilnya kembali. Unit kontrol menyediakan sinyal yang mengontrol operasi ALU dan pergerakan data ke dalam dan keluar dari ALU. Perhitungan dalam ALU menggunakan kode biner, yang merepresentasikan instruksi yang akan dieksekusi (opcode) dan data yang diolah (operand). ALU biasanya menggunakan sistem bilangan biner two's Komplemen. ALU mendapat data dari register. Kemudian data tersebut diproses dan hasilnya akan disimpan dalam register tersendiri yaitu ALU output register, sebelum disimpan dalam memori. Processor terdiri dari 4 elemen yang melakukan sistem operasi terhadap data, 4 elemen itu adalah instruksi, petunjuk instruksi, beberapa register dan ALU (Arithmetic Logic Unit). Adalah sebuah petunjuk instruksi akan memberi tahu processor dimana instruksi dari sebuah aplikasi diletakkan di memori. Adapun alur proses dari ALU yang ditunjukan oleh gambar dibawah ini.

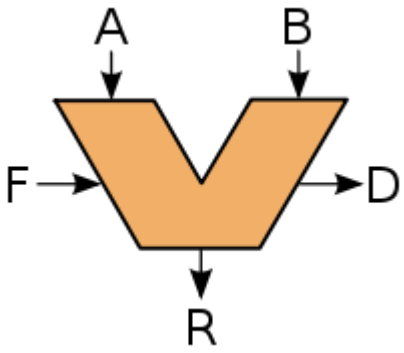


Gambar 1 Masukan dan Luaran ALU (William Stalling, 2010)

(2) Tugas dan Fungsi ALU

Tugas dari ALU adalah melakukan keputusan dari operasi logika sesuai dengan instruksi program. Operasi logika (logical operation) meliputi perbandingan dua buah elemen logika dengan menggunakan operator logika, yaitu :

(a) sama dengan (=)
(b) tidak sama dengan (<>)
(c) kurang dari (<)
(d) kurang atau sama dengan dari (<=)
(e) lebih besar dari (>)
(f) lebih besar atau sama dengan dari (>=)



Gambar 2 Simbol ALU (Wikipedia, 2017)

(3) Adder ALU

Adder merupakan rangkain ALU yang digunakan untuk menjumlahkan bilangan, dalam hal ini adder digunakan untuk memproses operasi aritmatika, maka adder juga sering disebut rangkaian kombinasional aritmatika. Adder terdapat 3 (tiga) jenis yaitu:

[1] Rangkaian adder yang hanya menjumlahkan dua bit disebut Half Adder (HA).
[2] Rangkaian adder yang hanya menjumlahkan tiga bit disebut Full Adder (FA)
[3] Rangkaian adder yang menjumlahkan banyak bit disebut Paralel Adder (PA)

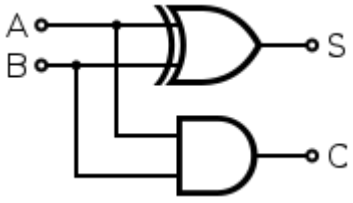
Penjelasan ;

[1] Half Adder (HA)

Rangkain half adder merupakan dasar bilangan biner yang masing-masing hanya terdiri dari satu bit, oleh karena itu dinamakan penjumlah tak lengkap.

- Jika A=0 dan B=0 dijumlahkan, hasilnya S (Sum) = 0.
- Jika A=0 dan B=0 dijumlahkan, hasilnya S (Sum) = 1.
- Jika A=1 dan B=1 dijumlahkan, hasilnya S (Sum) = 0. Dengan nilai pindahan C (Carry Out) = 1.

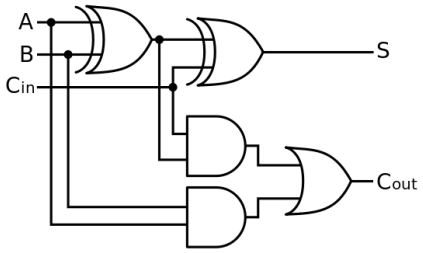
Dengan demikian, HA memiliki 2 (dua) masukan (A dan B), dan dua luaran (S dan C).



Gambar 3 Half Adder (Wikimedia, 2016)

[2] Full Adder (FA)

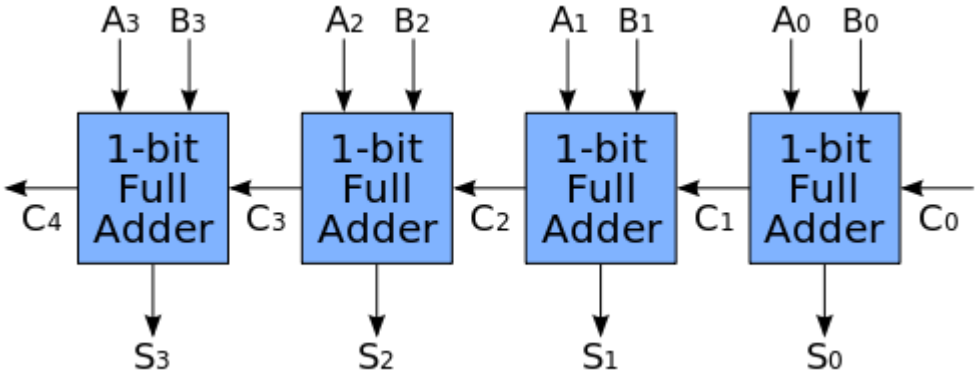
Full adder adalah rangkaian elektronik yang bekerja melakukan perhitungan penjumlahan sepenuhnya dari dua buah bilangan binary, yang masing-masing terdiri dari satu bit. Rangkaian ini memiliki tiga input dan dua buah output, salah satu input merupakan nilai dari pindahan penjumlahan, kemudian sama seperti pada HA salah satu outputnya dipakai sebagai tempat nilai pindahan dan yang lain sebagai hasil dari penjumlahan.



Gambar 4 Full Adder (Wikipedia, 2017)

[3] Parallel Adder (PA).

Paralel Adder adalah rangkaian FA yang disusun secara paralel dan berfungsi untuk menjumlahkan bilangan biner berapa pun bitnya, tergantung jumlah Full Adder yang diparalelkan. Gambar dibawah ini menunjukan PA yang terdiri dari 4 buah FA yang disusun paralel sehingga membentuk sebuah penjumlahan 4 bit.



Gambar 5 Paralel Adder (Wikipedia, 2017)

3. Representasi Integer dan Aritmatika

[1] Representasi Integer

Penyajian Representasi Integer terdapat 2 (dua) pendekatan ;

(a) Unsigned Magnitude
(b) Signed Magnitude (komplemen dua)

Penjelasan ;

(a) "Unsigned magnitude"

Komputer di era modern biasanya mendukung bilangan integer 8, 16, 32, atau 64 bit.

Table Range Integer Biner *Unsigned* [1]

Bit	Range
8	0 to 2^8-1 (255)
16	0 to 2^{16}-1 (65,535)
32	0 to 2^{32}-1 (4,294,967,295)
64	0 to 2^{64}-1 (18,446,744,073,709,551,615 = 1.844674407 x 10^{19})

Tabel 2, di bawah ini contoh untuk data integer positif dan negatif dengan menggunakan data 4-bit.

Table Representasi Data Integer 4-bit [2]

Desimal	Unsigned Magnitude	Sign Magnitude	Ones' Komplemen	Two's Komplemen
+16	-	-	-	-
+15	1111	-	-	-
+14	1110	-	-	-
+13	1101	-	-	-
+12	1100	-	-	-
+11	1011	-	-	-
+10	1010	-	-	-
+9	1001	-	-	-
+8	1000	-	-	-
+7	111	111	111	111
+6	110	110	110	110
+5	101	101	101	101
+4	100	100	100	100
+3	11	11	11	11
+2	10	10	10	10
+1	1	1	1	1
+0	0000	0000	0000	0000
−0		1000	1111	
−1	-	1001	1110	1111
−2	-	1010	1101	1110
−3	-	1011	1100	1101
−4	-	1100	1011	1100

[1] http://www.cs.uwm.edu/classes/cs315/Bacon/Lecture/HTML/ch04s10.html
[2] https://en.wikipedia.org/wiki/Signed_number_representations

−5	-	1101	1010	1011
−6	-	1110	1001	1010
−7	-	1111	1000	1001
−8	-	-	-	1000
−9	-	-	-	-
−10	-	-	-	-
−11	-	-	-	-

Dengan menggunakan data yang sama pada table 2, asumsi dengan data biner direpresentasikan sebagai berikut

Table Representasi Data Biner 4-bit [3]

Binary	Unsigned	Sign and magnitude	Ones' Komplemen	Two's Komplemen
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	−0	−7	−8
1001	9	−1	−6	−7
1010	10	−2	−5	−6
1011	11	−3	−4	−5
1100	12	−4	−3	−4
1101	13	−5	−2	−3
1110	14	−6	−1	−2
1111	15	−7	−0	−1

Dalam bentuk persamaan secara umum unsigned integer magnitude untuk n-bit ;

$$A = \sum_{i=0}^{n-1} 2^i a_i$$

Kelemahan unsigned integer magnitude, hanya dapat menyatakan bilangan positif saja.sehingga tidak dipergunakan pada bilangan integer negatif.

(b) “Signed magnitude”

Format paling umum yang digunakan untuk representassi data integer yang diproses pada komputer modern adalah komplemen dua (2s’). Dalam sistem

[3] https://en.wikipedia.org/wiki/Signed_number_representations

biner, representasi bilangan signed berisi tanda (sign) dan besar nilai (magnitude), Tanda diyatakan oleh bit paling kiri (0) = bilangan positif, (1) = bilangan negatif).

Contoh ;

$$+ 14_{10} = 01110_{2s'}$$

Dan

$$- 14_{10} = 10001 + 1 = 10010_{2s'}$$

Table Range Integer Biner *Signed* [4]

Bit	Range
8	**$-2^7$0 (-128) to $+2^7$-1 (+127)**
16	**-2^{15} (-32,768) to $+2^{15}$-1 (32,767)**
32	**-2^{31} (-2,147,483,648) to $+2^{31}$-1 (+2,147,483,647)**
64	**-2^{63} (-9,223,372,036,854,775,808) to $+2^{63}$-1 (9,223,372,036,854,775,807)**

Di bilangan signed, terdapat 3 (tiga) format yang umum digunakan untuk representasi bilangan negative ;

(a) Sign Magnitude

Bilangan sign-magnitude menggunakan 1 bit paling kiri untuk Dalam sistem biner, representasi bilangan signed berisi tanda (sign) dan besar nilai (magnitude), Tanda diyatakan oleh bit paling kiri (0) = bilangan positif, (1) = bilangan negatif).

	0	1	2	3	4	5	6	7
Positif	0000	0001	0010	0011	0100	0101	0110	0111
Negatif	1000	1001	1010	1011	1100	1101	1110	1111

Dalam bentuk persamaan secara umum sign integer magnitude untuk n-bit ;

$$A = \begin{cases} \sum_{i=0}^{n-2} 2^i a_i & \text{if } a_{n-1} = 0 \\ -\sum_{i=0}^{n-2} 2^i a_i & \text{if } a_{n-1} = 1 \end{cases}$$

[4] http://www.cs.uwm.edu/classes/cs315/Bacon/Lecture/HTML/ch04s13.html

(b) 1's Komplemen

Bilangan n-bit negatif K dapat diperoleh dari mengurangkan $2^n - 1$ dengan bilangan positif ekivalennya P. dalam bentuk rumus

$$1S' = (2^n - 1) - P$$

	0	1	2	3	4	5	6	7
Positif	0000	0001	0010	0011	0100	0101	0110	0111
Negatif	1111	1110	1101	1100	1011	1010	1001	1000

(c) 2's Komplemen

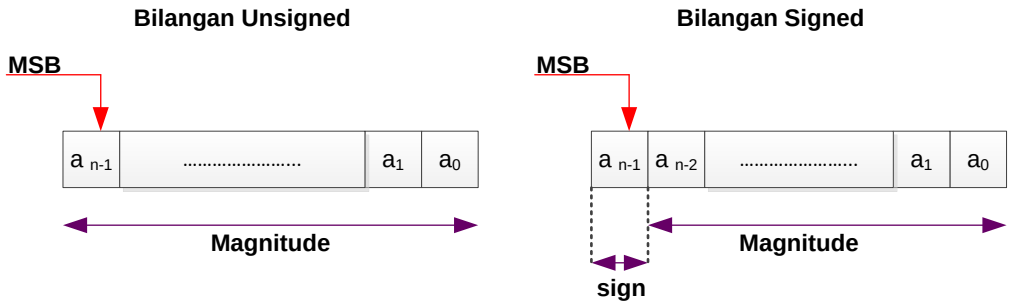
Bilangan n-bit negatif K dapat diperoleh dari mengurangkan 2^n dengan bilangan positif ekivalennya P. dalam bentuk rumus

$$2S' = 2^n - P$$

	0	1	2	3	4	5	6	7	8
Positif	0000	0001	0010	0011	0100	0101	0110	0111	-
Negatif	0000	1111	1110	1101	1100	1011	1010	1001	1000

Hubungan point (b) dan (c), bahwa 2's Komplemen dapat dibentuk dengan mengkomplemenkan tiap bit bilangan dan menambahkan 1

2S' = 1S' + 1 (posisi pada LSB)



Note : MSB (Most Significant Bit) dan a_0 sebagai LSB (Least Significant Bit)

[2] Aritmatika Integer

(a) Negasi (*Negation*)
(b) Penjumlahan (*Addition*)
(c) Pengurangan (*Subtraction*)
(d) Perkalian (*Multiplication*)

(e) Pembagian (*Division*)

Penjelasan ;'

(a) Negasi (*Negation*)

Dalam representasi sign-magnitude, aturan untuk membentuk negasi dari integer adalah sederhana adalah membalikkan bit tanda. Notasi komplemen dua, negasi dari integer dapat dibentuk dengan aturan berikut:

(a) Ambil komplemen Boolean dari setiap bit integer (termasuk sign bit). Dengan kata lain bit 1 dikonversi menjadi bit 0 dan berlaku sebaliknya

(b) Sehingga memperoleh biner integer biner unsigned, dengan menambahkan tambahkan bit 1 (LSB)

Contoh ;

+ 18 = 00010010 (komplemen 2S')
11101101 (komplemen satu) + 1 (**LSB**)
= 11101110 = **- 18**

Bila proses dikembalikan seperti semula (awal) dengan soal yang sama (diatas) ;

- 18 = 11101110 (komplemen 2S')
00010010 (komplemen satu) + 1 (**LSB**)
= 00010010 = **+ 18**

(b) Penjumlahan (*Addition*)

1001 = −7 +0101 = 5 1110 = −2 (a) (−7) + (+5)	1100 = −4 +0100 = 4 10000 = 0 (b) (−4) + (+4)
0011 = 3 +0100 = 4 0111 = 7 (c) (+3) + (+4)	1100 = −4 +1111 = −1 11011 = −5 (d) (−4) + (−1)
0101 = 5 +0100 = 4 1001 = Overflow (e) (+5) + (+4)	1001 = −7 +1010 = −6 10011 = Overflow (f) (−7) + (−6)

(c) Pengurangan (*Subtraction*)

0010 = 2 +1001 = –7 1011 = –5 (a) M = 2 = 0010 S = 7 = 0111 –S = 1001	0101 = 5 +1110 = –2 10011 = 3 (b) M = 5 = 0101 S = 2 = 0010 –S = 1110
1011 = –5 +1110 = –2 11001 = –7 (c) M = –5 = 1011 S = 2 = 0010 –S = 1110	0101 = 5 +0010 = 2 0111 = 7 (d) M = 5 = 0101 S = –2 = 1110 –S = 0010
0111 = 7 +0111 = 7 1110 = Overflow (e) M = 7 = 0111 S = –7 = 1001 –S = 0111	1010 = –6 +1100 = –4 10110 = Overflow (f) M = –6 = 1010 S = 4 = 0100 –S = 1100

(d) Perkalian (*Multiplication*)

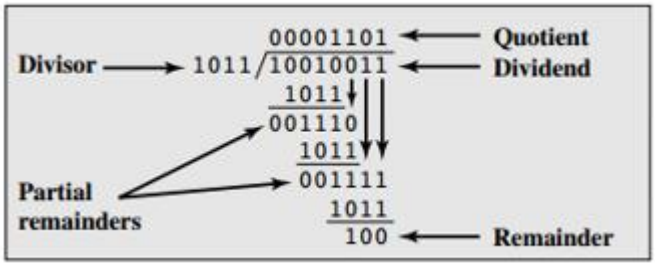
- **Unsigned Integer Biner**

```
   1011      Multiplicand (11)
  ×1101      Multiplier (13)
  ------
   1011   \
   0000    |
  1011     |  Partial products
 1011      |
------     /
10001111    Product (143)
```

```
     1011
   × 1101
  --------
  00001011   1011 × 1 × 2^0
  00000000   1011 × 0 × 2^1
  00101100   1011 × 1 × 2^2
  01011000   1011 × 1 × 2^3
  --------
  10001111
```

1001 (9)	1001 (−7)
× 0011 (3)	× 0011 (3)
00001001 1001 × 2^0	11111001 (−7) × 2^0 = (−7)
00010010 1001 × 2^1	11110010 (−7) × 2^1 = (−14)
00011011 (27)	11101011 (−21)

(e) Pembagian (*Division*)



(4) Representasi dan Aritmatika Floating Point

Dalam komputasi floating point menjelaskan metode mewakili perkiraan dari sejumlah nyata dalam cara yang dapat mendukung berbagai nilai. Jumlahnya secara umum mewakili sekitar untuk tetap jumlah digit yang signifikan (mantissa) dan menggunakan eksponen.
Dengan asumsi bahwa resolusi terbaik adalah di tahun cahaya hanya 9 desimal yang paling signifikan digit sedangkan sisanya 30 digit dan dengan demikian dapat dengan aman diabaikan. Ini merupakan penghematan dari 100 bit penyimpanan data komputer . Alih-alih dari 100 bit , jauh lebih sedikit digunakan untuk mewakili skala (eksponen) misalnya 8 bit atau 2 digit decimal. Bilangan yang mempunyai nilai pecahan (misalnya 3.2575) dapat direpresentasikan dengan dua format bilangan: fixed-point dan floating-point (Eko Didik Widianto, 2012).
Bilangan pecahan fixed-point mempunyai jangkauan yang dibatasi oleh jumlah digit signifikan yang digunakan untuk merepresentasikan bilangan tersebut. Misalnya bilangan pecahan desimal sepuluh digit. Bilangan tersebut dinyatakan dengan fixed-point, yaitu satu digit untuk tanda, 4 (empat) digit untuk angka utuh dan lima digit untuk angka pecahan. Jangkauan bilangan tersebut adalah 0 sampai 9999 untuk angka utuh dan 0.00001 sampai 0.99999 untuk angka pecahan, sehingga nilai bilangan yang mungkin adalah -9999.99999 sampai + 9999.99999 dengan presisi 0.00001. Contoh bilangan tersebut yang valid adalah - 9.00102 dan 100.99998. Bilangan ± 10000 tidak bisa dinyatakan dengan sistem bilangan sepuluh digit ini. Sedangkan bilangan 0.000005 tidak memenuhi derajat presisi yang diinginkan, walaupun berada dalam jangkauan bilangan. Bilangan tersebut akan diintegerkan ke 0.00000 atau 0.00001, yang berarti ada selisih sebesar ± 0.000005 dari nilai yang diinginkan.

Dalam aplikasi saintifik, mungkin akan terdapat bilangan yang sangat besar atau sangat kecil. Bilangan tersebut harus dapat direpresentasikan dengan tepat (presisi), yaitu menggunakan floating-point. Bilangan floating-point direpresentasikan dengan mantissa yang berisi digit signifikan dan eksponen dari radix R, secara empiris format adalah

$$\mathbf{FF = mantisa \times R^{eksponen}}$$

Contoh ;

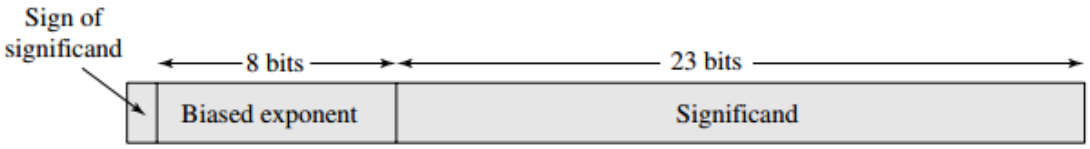
$$\mathbf{1,5 \times 10^{44} \text{ atau } 1,253 \times 10^{-36}}$$

Format bilangan floating-point biner telah distandarkan oleh IEEE 754-2008 (atau ISO/IEC/IEEE 60559:2011), yaitu meliputi format 16-bit (half), 32-bit (single-precision), 64-bit (double-precision), 80-bit (double-extended) dan 128-bit (quad-precision). Materi ini hanya dibahas tentang format dasar, yaitu 32-bit dan 64-bit.

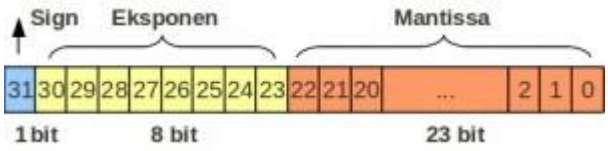
Bilangan Floating-Point 32-bit (single-precision)

Bilangan floating-point 32-bit tersusun atas (Gambar 0.1↓):

- 1 bit tanda (S),
- 8 bit eksponen (E), dan
- 23 bit untuk mantisa (M)



Gambar 6a Single Format bilangan floating-point 32-bit



Gambar 6b Single Format bilangan floating-point 32-bit

Keterangan ; bit tanda (S) menyatakan bilangan positif jika S=0 dan negatif jika S=1.

Contoh floating-point 32-bit

$1.1010001 \times 2^{10100} = 0\ 10010011\ 10100010000000000000000 = 1.6328125 \times 2^{20}$

$-1.1010001 \times 2^{10100} = 1\ 10010011\ 10100010000000000000000 = -1.6328125 \times 2^{20}$

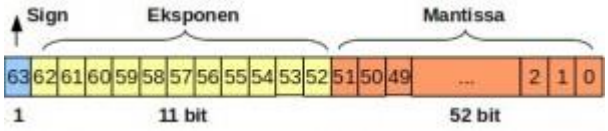
$1.1010001 \times 2^{-10100} = 0\ 01101011\ 10100010000000000000000 = 1.6328125 \times 2^{-20}$

$-1.1010001 \times 2^{-10100} = 1\ 01101011\ 10100010000000000000000 = -1.6328125 \times 2^{-20}$

Bilangan Floating-Point 64-bit (double-precision)

Bilangan floating-point 64-bit tersusun atas (gambar 7)

- 1 bit tanda (S),
- 11 bit eksponen (E), dan
- 52 bit untuk mantisa (M)



Gambar 7 Format bilangan floating-point 64-bit

Keterangan ; bit tanda (S) menyatakan bilangan positif jika S=0 dan negatif jika S=1.

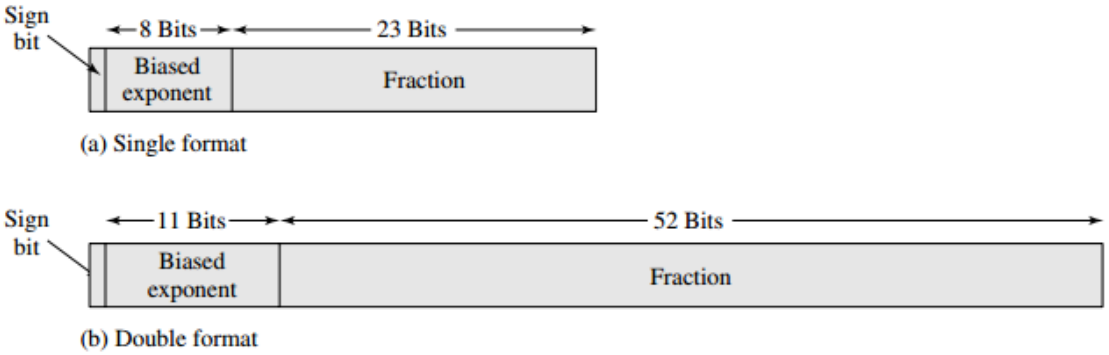


Figure 9.21 IEEE 754 Formats

Table 3 Struktur Memori *Singles* dan *Doubles Precision*

	Sign	Exponent	Mantissa
Single Precision	**1**	**8**	**23**
Double Precision	**1**	**11**	**52**

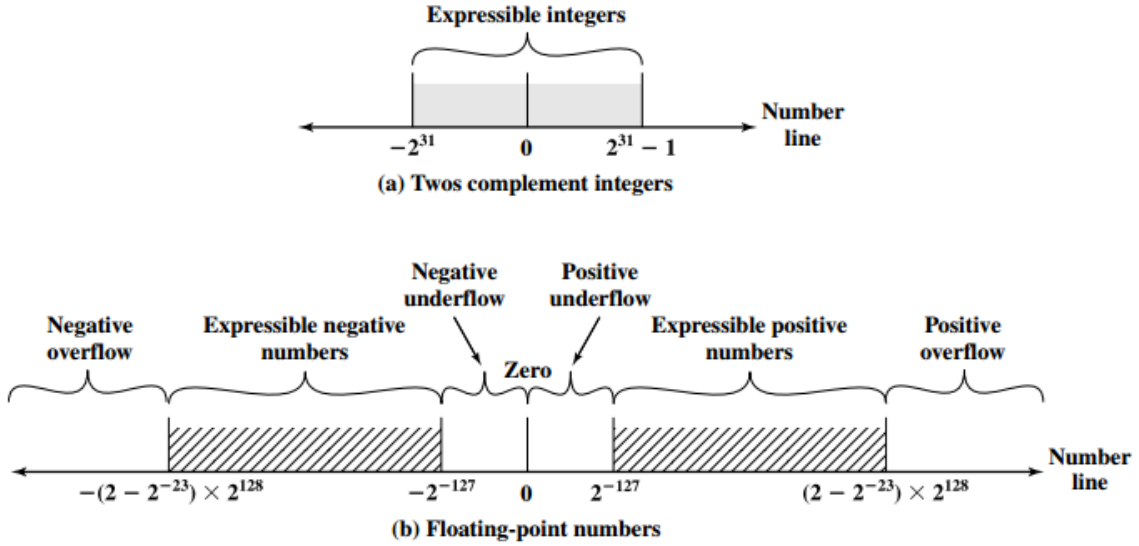


Figure 9.19 Expressible Numbers in Typical 32-Bit Formats

Table 9.3 IEEE 754 Format Parameters

Parameter	Format			
	Single	Single Extended	Double	Double Extended
Word width (bits)	32	≥43	64	≥79
Exponent width (bits)	8	≥11	11	≥15
Exponent bias	127	unspecified	1023	unspecified
Maximum exponent	127	≥1023	1023	≥16383
Minimum exponent	−126	≤−1022	−1022	≤−16382
Number range (base 10)	$10^{-38}, 10^{+38}$	unspecified	$10^{-308}, 10^{+308}$	unspecified
Significand width (bits)*	23	≥31	52	≥63
Number of exponents	254	unspecified	2046	unspecified
Number of fractions	2^{23}	unspecified	2^{52}	unspecified
Number of values	1.98×2^{31}	unspecified	1.99×2^{63}	unspecified

*not including implied bit

Table 9.4 Interpretation of IEEE 754 Floating-Point Numbers

	Single Precision (32 bits)				Double Precision (64 bits)			
	Sign	Biased exponent	Fraction	Value	Sign	Biased exponent	Fraction	Value
positive zero	0	0	0	0	0	0	0	0
negative zero	1	0	0	−0	1	0	0	−0
plus infinity	0	255 (all 1s)	0	∞	0	2047 (all 1s)	0	∞
minus infinity	1	255 (all 1s)	0	−∞	1	2047 (all 1s)	0	−∞
quiet NaN	0 or 1	255 (all 1s)	≠0	NaN	0 or 1	2047 (all 1s)	≠0	NaN
signaling NaN	0 or 1	255 (all 1s)	≠0	NaN	0 or 1	2047 (all 1s)	≠0	NaN
positive normalized nonzero	0	$0 < e < 255$	f	$2^{e-127}(1.f)$	0	$0 < e < 2047$	f	$2^{e-1023}(1.f)$
negative normalized nonzero	1	$0 < e < 255$	f	$-2^{e-127}(1.f)$	1	$0 < e < 2047$	f	$-2^{e-1023}(1.f)$
positive denormalized	0	0	$f \neq 0$	$2^{e-126}(0.f)$	0	0	$f \neq 0$	$2^{e-1022}(0.f)$
negative denormalized	1	0	$f \neq 0$	$-2^{e-126}(0.f)$	1	0	$f \neq 0$	$-2^{e-1022}(0.f)$

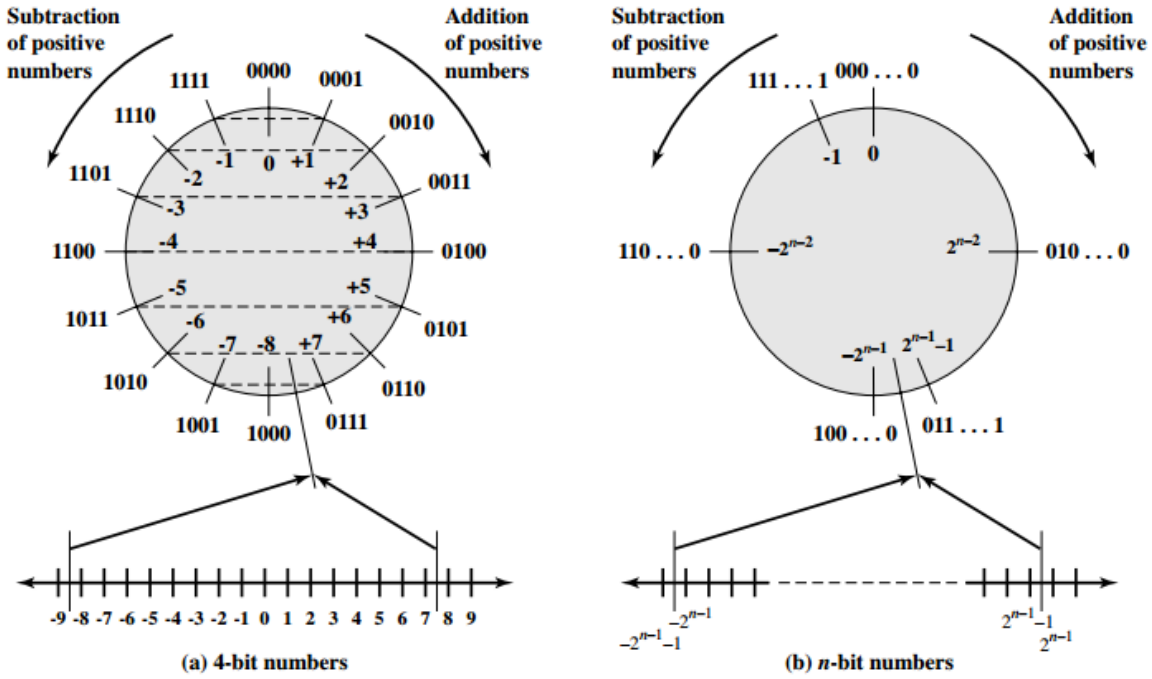


Figure 9.5 Geometric Depiction of Twos Complement Integers

4. Penyimpanan Karakter

[1] ASCII (American Standard Code for Information Interchange)

[2] ISO (International Standards Organization)

[3] UTF (Unicode Transformation Formats)

(1) Instruksi Komputer

KEY POINTS

- *The essential elements of a computer instruction are the opcode, which specifies the operation to be performed; the source and destination operand references, which specify the input and output locations for the operation; and a next instruction reference, which is usually implicit.*
- *Opcodes specify operations in one of the following general categories : arithmetic and logic operations; movement of data between two registers, register and memory, or two memory locations; I/O; and control.*
- *Operand references specify a register or memory location of operand data. The type of data may be addresses, numbers, characters, or logical data.*
- *A common architectural feature in processors is the use of a stack, which may or may not be visible to the programmer. Stacks are used to manage procedure calls and returns and may be provided as an alternative form of addressing memory.The basic stack operations are PUSH, POP, and operations on the top one or two stack locations. Stacks typically are implemented to grow from higher addresses to lower addresses.*
- *Byte-addressable processors may be categorized as big endian, little endian, or bi-endian. A multibyte numerical value stored with the most significant byte in the lowest numerical address is stored in big-endian fashion.The little-endian style stores the most significant byte in the highest numerical address.A bi-endian processor can handle both styles.*

Point penting yang harus diperhatikan dan dipelajari !!!

- Elemen penting dari instruksi komputer adalah opcode, yang menentukan operasi yang akan dilakukan; sumber dan tujuan referensi operand, yang menentukan lokasi input dan output untuk operasi; dan referensi instruksi berikutnya, yang biasanya implisit.
- Opcode menentukan operasi dalam salah satu kategori umum berikut: operasi aritmatika dan logika; pergerakan data antara dua register, daftar dan memori, atau dua lokasi memori; I / O; dan kontrol.
- Referensi Operand menentukan lokasi register atau memori data operan. Jenis data dapat berupa alamat, angka, karakter, atau data logis.
- Fitur arsitektur umum dalam prosesor adalah penggunaan tumpukan, yang mungkin atau mungkin tidak terlihat oleh programmer. Tumpukan digunakan untuk mengelola prosedur panggilan dan pengembalian dan dapat diberikan sebagai bentuk alternative menangani memori. Operasi tumpukan dasar adalah PUSH, POP, dan operasi di satu atau dua lokasi tumpukan teratas. Tumpukan biasanya diterapkan tumbuh dari alamat yang lebih tinggi ke alamat yang lebih rendah.prosesor
- Byte-addressable dapat dikategorikan sebagai big endian, little endian, atau bi-endian. Nilai numerik multibyte yang disimpan dengan byte paling signifikan dalam alamat numerik terendah disimpan dalam mode big-endian. Gaya little-endian menyimpan byte paling signifikan dalam alamat numerik tertinggi. Prosesor bi-endian dapat menangani kedua gaya.

Lampiran Operand Aritmatika Floating-Point

Addition dan Subtraction

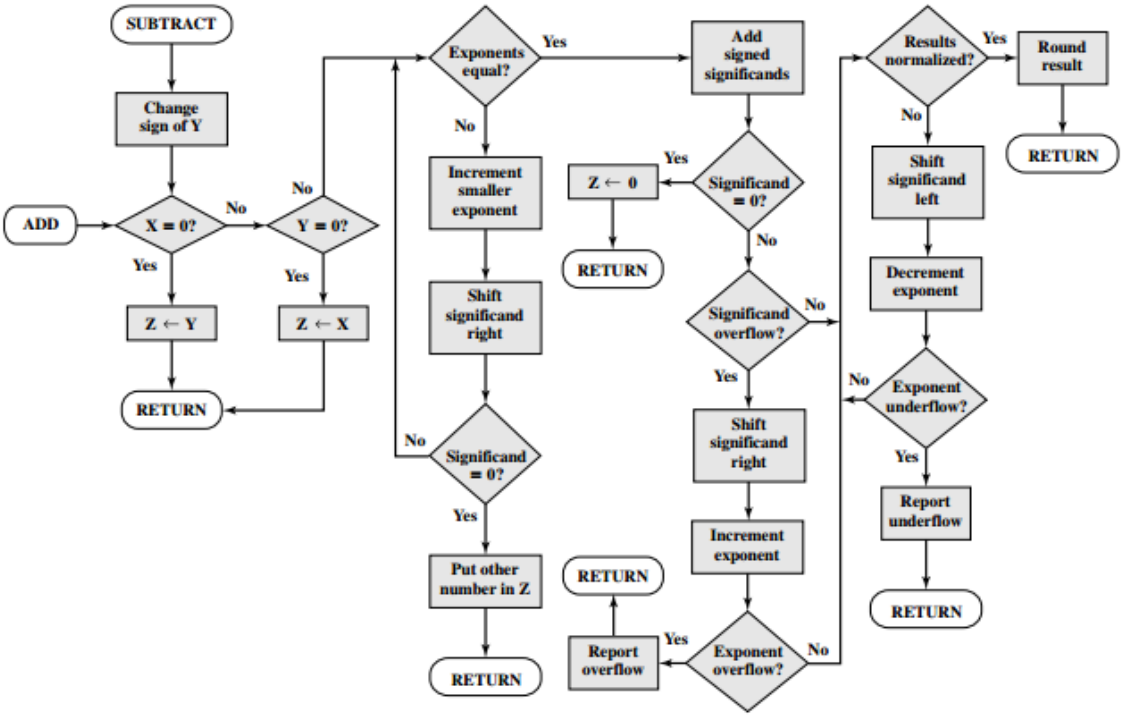


Figure 9.22 Floating-Point Addition and Subtraction ($Z \leftarrow Z \pm Y$)

Multiplication

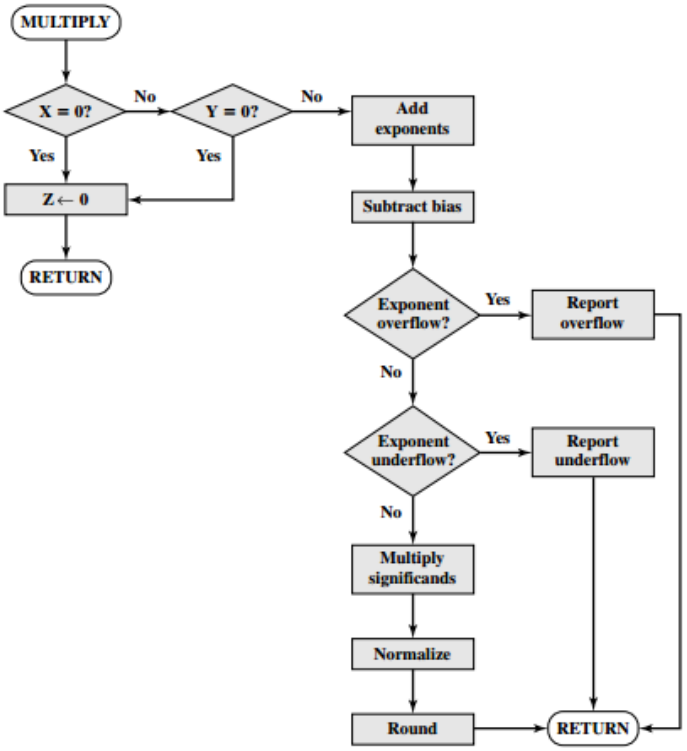


Figure 9.23 Floating-Point Multiplication ($Z \leftarrow X \times Y$)

Division

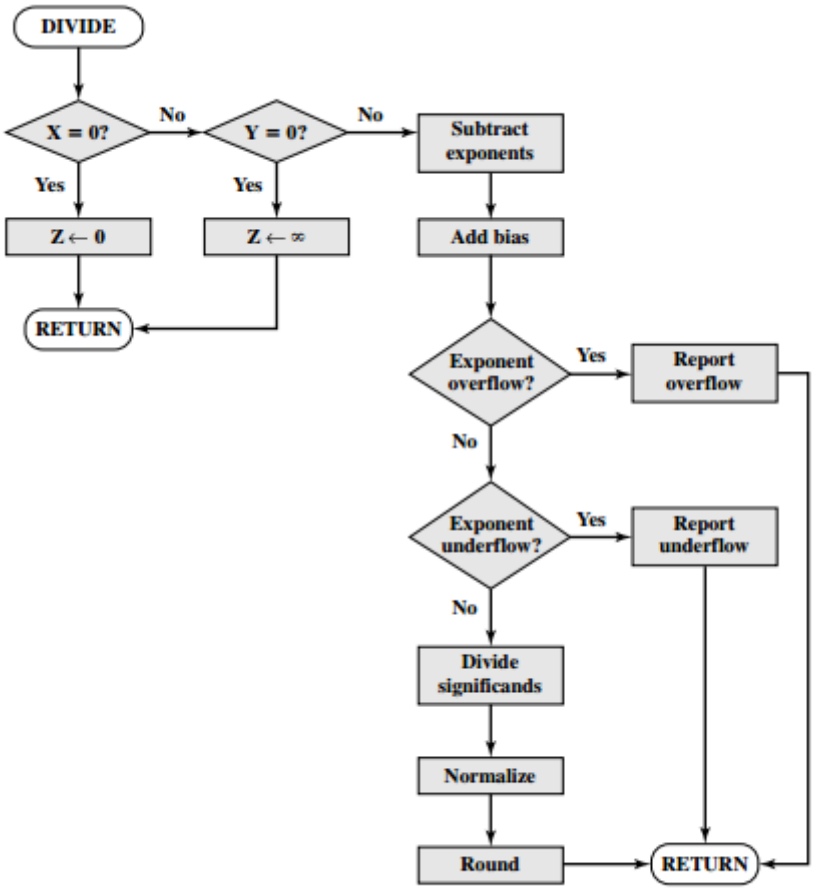


Figure 9.24 Floating-Point Division (Z ← X/Y)