Arsitektur dan Organisasi Komputer

COM 60011

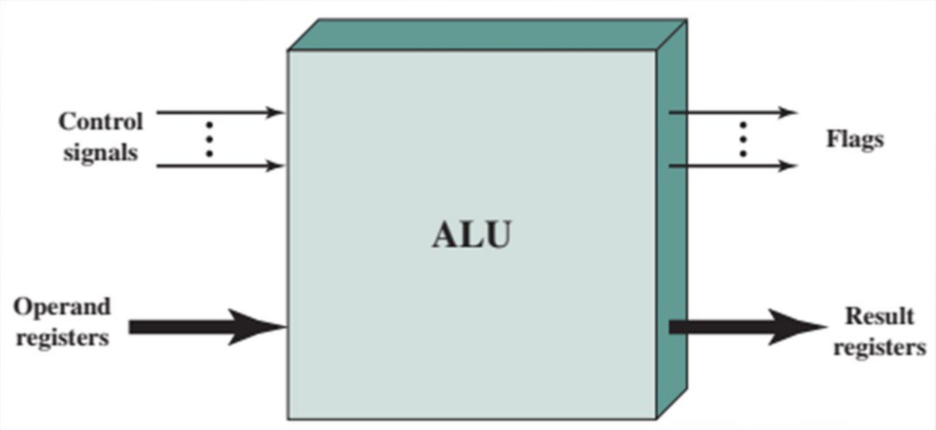
Topik #9 – Aritmatika Komputer

Pendahuluan

- **Tujuan**
  - Memahami perbedaan representasi bilangan biner dan algoritma yang digunakan untuk operasi aritmatika dasar
  - Menjelaskan representasi two's complement
  - Menjelaskan gambaran umum operasi aritmatika dasar pada notasi two's complement
  - Memahami penggunaan significand, base, dan exponent pada representasi bilangan floating-point
  - Menjelaskan gambaran umum standar IEEE 754 pada bilangan floating-point
  - Memahami beberapa konsep utama yang terkait aritmatika floating-point
  - Memahami konsep dasar logika digital
- **Materi**
  - Arithmetic & Logic Unit (ALU)
  - Representasi integer dan aritmatika integer
  - Representasi floating-point dan aritmatika floating-point
  - Konsep dasar logika digital

Arithmetic & Logic Unit (ALU)

- ALU → bagian dari komputer yang melakukan operasi aritmatika dan logika
- Semua elemen dari sistem komputer membawa data ke ALU untuk diproses dan mengambil hasilnya lagi
- Penggunaan perangkat logika digital sederhana yang menyimpan digit biner dan melakukan operasi Boolean



Representasi Integer

- **Dalam sistem biner, bilangan direpresentasikan dengan:**
  - Digit 1 dan 0
  - Tanda minus (bilangan negatif)
  - Tanda titik (radix point) → bilangan pecahan
- **Data pada komputer menggunakan representasi bilangan biner (0 dan 1) → bit**

00000000	=	0
00000001	=	1
00101001	=	41
10000000	=	128
11111111	=	255

Sign-Magnitude

- **Sign bit → bit paling kiri (Most Significant Bit - MSB)**
  - 0 → positif
  - 1 → negatif

+18	= 00010010	
−18	= 10010010	(sign magnitude)

- **Kekurangan:**
  - penambahan dan pengurangan memerlukan pertimbangan baik tanda bilangan ataupun nilai relatifnya agar dapat berjalan pada operasi yang diperlukan
  - terdapat dua representasi bilangan 0
  - jarang digunakan pada implementasi integer di ALU

$+0_{10}$	= 00000000	
-0_{10}	= 10000000	(sign magnitude)

Twos Complement

- **Mengatasi kekurangan yang terdapat pada sign-magnitude:**
  - Operasi penjumlahan dan pengurangan
  - Representasi bilangan nol

Range	-2_{n-1} through $2_{n-1} - 1$
Number of Representations of Zero	One
Negation	Take the Boolean complement of each bit of the corresponding positive number, then add 1 to the resulting bit pattern viewed as an unsigned integer.
Expansion of Bit Length	Add additional bit positions to the left and fill in with the value of the original sign bit.
Overflow Rule	If two numbers with the same sign (both positive or both negative) are added, then overflow occurs if and only if the result has the opposite sign.
Subtraction Rule	To subtract B from A, take the twos complement of B and add it to A.

Konversi Value Box

−128	64	32	16	8	4	2	1

(a) An eight-position twos complement value box

−128	64	32	16	8	4	2	1
1	0	0	0	0	0	1	1

−128 +2 +1 = −125

(b) Convert binary 10000011 to decimal

−128	64	32	16	8	4	2	1
1	0	0	0	1	0	0	0

−120 = −128 +8

(c) Convert decimal −120 to binary

Range Extension

- **Rentang bilangan dapat diperpanjang dengan menambah panjang bit**
- **Dalam notasi sign-magnitude dilakukan dengan memindahkan sign-bit ke posisi paling kiri baru dan mengisinya dengan nol**

+18	=	00010010	(sign magnitude, 8 bits)
+18	=	0000000000010010	(sign magnitude, 16 bits)
−18	=	10010010	(sign magnitude, 8 bits)
−18	=	1000000000010010	(sign magnitude, 16 bits)

+18	=	00010010	(twos complement, 8 bits)
+18	=	0000000000010010	(twos complement, 16 bits)
−18	=	11101110	(twos complement, 8 bits)
−32,658	=	1000000001101110	(twos complement, 16 bits)

- **Prosedur ini tidak akan bekerja untuk twos complement bilangan bulat negatif, maka berikut aturannya:**
  - Memindahkan sign-bit ke posisi paling kiri baru dan mengisi dengan salinan sign-bit.
  - Untuk bilangan positif isikan dengan angka 0, dan untuk bilangan negatif isikan dengan angka 1.
  - Ini disebut sign extension

−18	=	11101110	(twos complement, 8 bits)
−18	=	1111111111101110	(twos complement, 16 bits)

Negation

- **MSB sebagai sign-bit seperti sign-magnitude, namun berbeda pada bilangan negatif. Contoh → +21 = 00010101**
  - Komplemen 1 dari bilangan biner (dibalik 1 ke 0 dan 0 ke 1)
    11101010
  - Tambahkan 1 pada LSB
    11101010
    1
    -------- +
    11101011 → Diperoleh bilangan negatif yaitu -21
- **Jika negasi dari bilangan negatif itu sendiri, misal -21**
  11101011 → -21
  00010100 → komplemen 1
  1
  -------- +
  00010101 → Diperoleh nilai yaitu +21

Special Case – Negation

0	=	00000000	(twos complement)
bitwise complement	=	11111111	
		+ 1	
		100000000 = 0	

- Ada carry out (penyimpanan angka) pada posisi MSB, diabaikan.
- Hasilnya negasi dari 0 adalah 0 → -0 = 0

+128	=	10000000	(twos complement)
bitwise complement	=	01111111	
		+ 1	
		10000000 = −128	

- Jika kita mengambil negasi dari pola bit 1 diikuti dengan n – 1, maka mendapatkan angka yang sama.

Penjumlahan Twos Complement

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)

ATURAN OVERFLOW → Jika dua angka ditambahkan, dan keduanya positif atau keduanya negatif, overflow terjadi jika dan hanya jika hasilnya bertanda sebaliknya

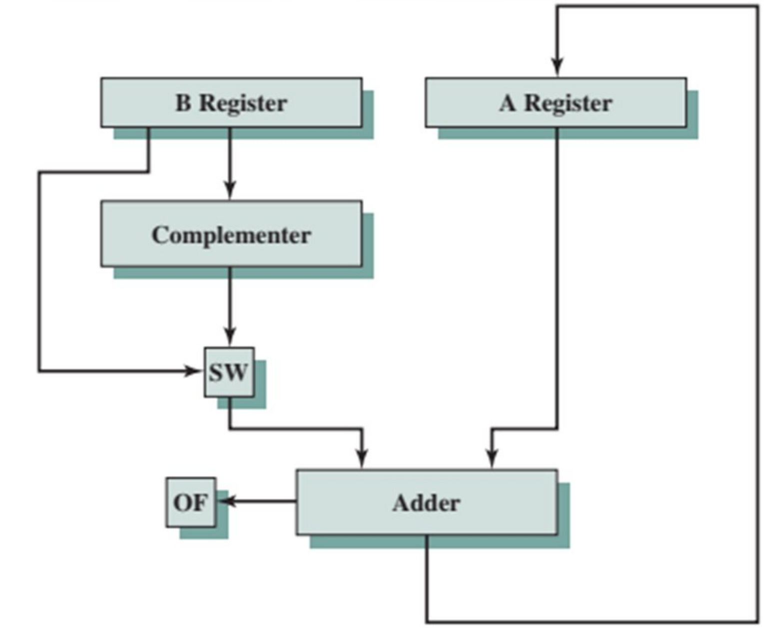
Pengurangan Twos Complement

a - b = c
a ⟶ Minuend
b ⟶ Pengurang (Subtrahend)
c ⟶ Selisih (Difference)

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

- ATURAN PENGURANGAN ⟶ Untuk mengurangi satu bilangan (pengurang) dari yang lain (minuend), ambil twos complement (negasi) dari pengurang dan tambahkan ke minuend

Diagram Blok Hardware (+ dan –)



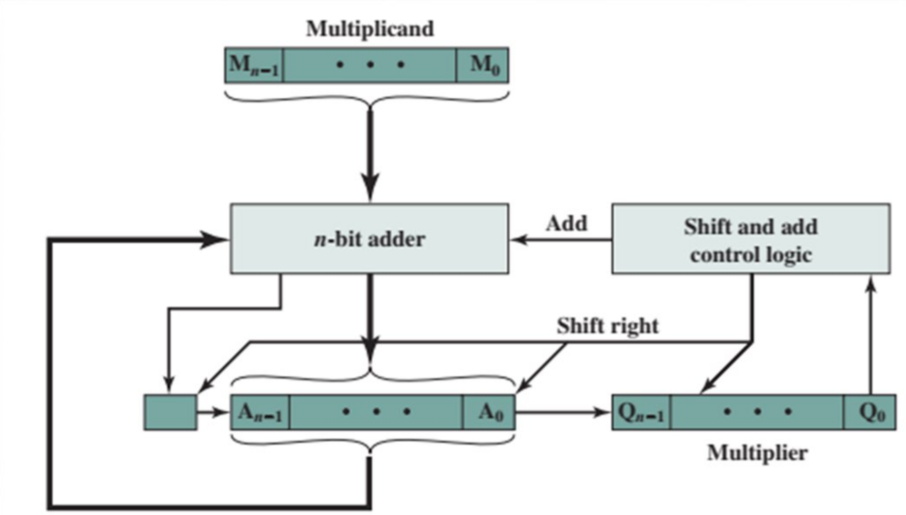
OF = Overflow bit
SW = Switch (select addition or subtraction)

Figure 10.6 Block Diagram of Hardware for Addition and Subtraction

Perkalian – Unsigned Binary Integer

```
    1011     Multiplicand (11)
   ×1101     Multiplier (13)
   -----
    1011  ⎫
   0000   ⎪
  1011    ⎬  Partial products
 1011     ⎭
--------
10001111     Product (143)
```

Implementasi Hardware – Perkalian Unsigned Binary Integer



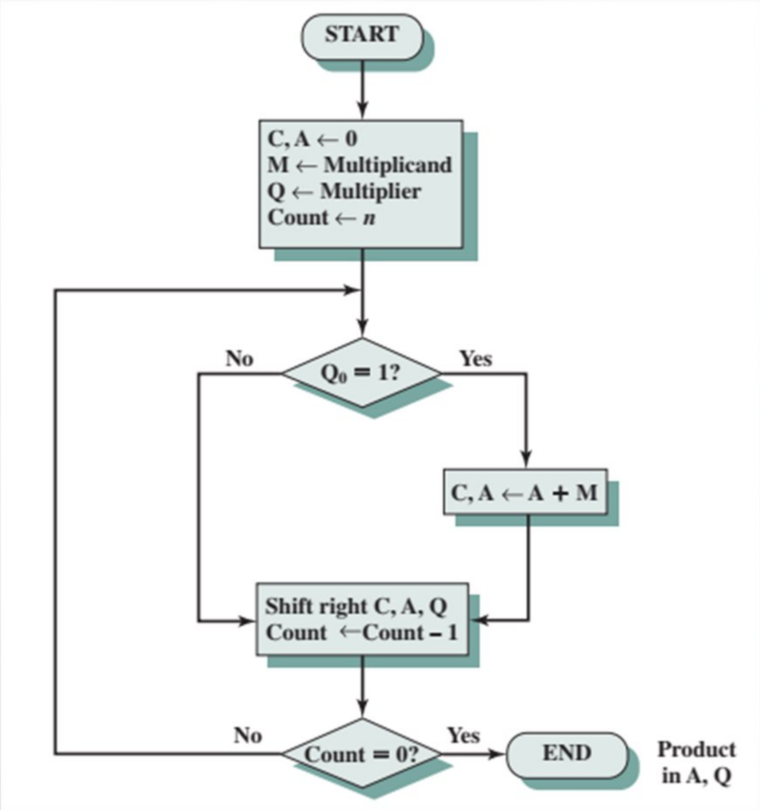
(a) Block diagram

C	A	Q	M		
0	0000	1101	1011	Initial values	
0	1011	1101	1011	Add	First cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second cycle
0	1101	1111	1011	Add	Third cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth cycle
0	1000	1111	1011	Shift	

(b) Example from Figure 10.7 (product in A, Q)

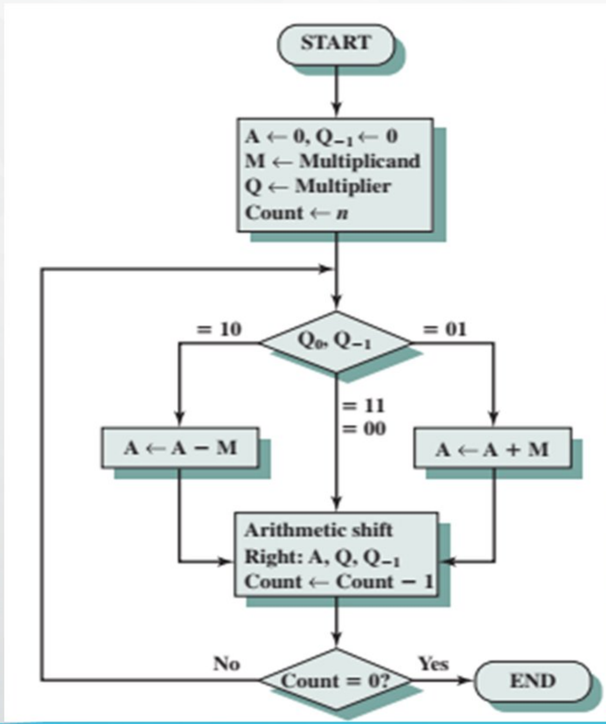
Q → Multiplier, M → Multiplicand, A → Hasil, C → carry bit

Flowchart Perkalian Unsigned Binary Integer



Perkalian Twos Complement Algoritma Booth

- **Algoritma Booth:**
  - Register Q (multiplier) , M (multiplicand), A (accumulator), dan register 1-bit di kanan Q yang ditandai dengan Q_{-1}. Hasil perkalian disimpan di register A dan Q.
  - A dan $Q_{-1} \rightarrow 0$

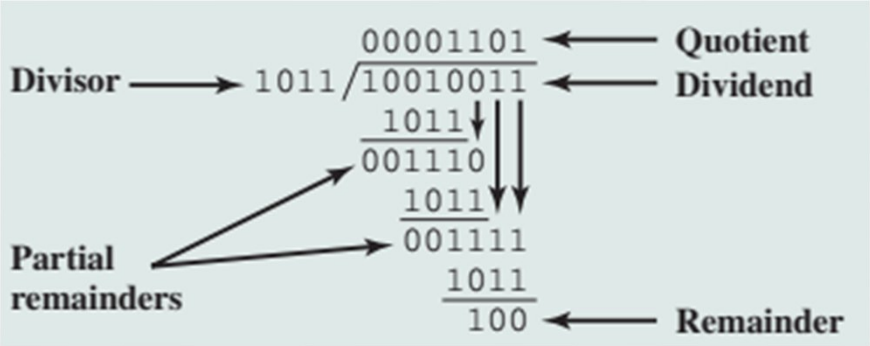


Contoh Algoritma Booth

Operasi Perkalian 7x3

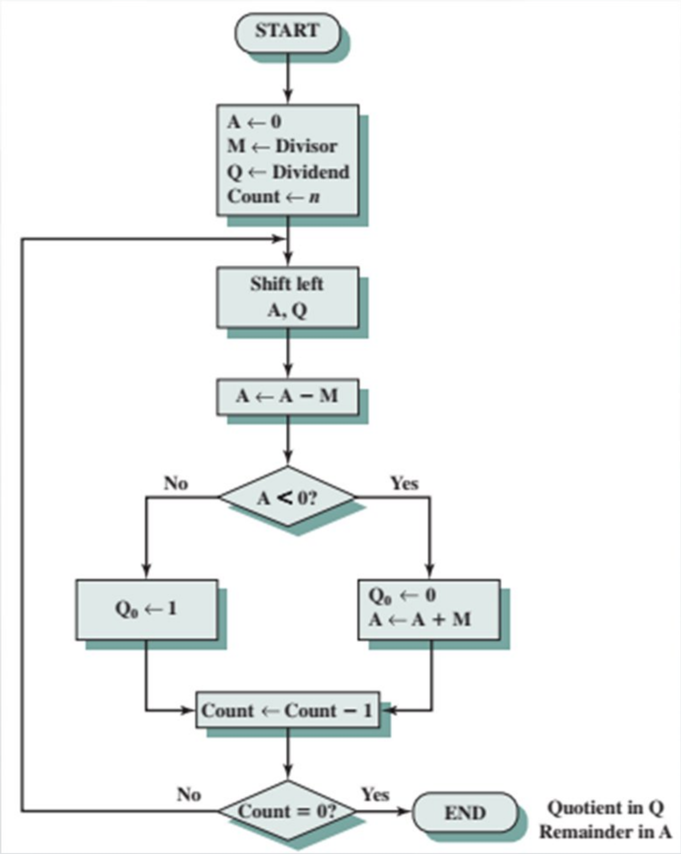
A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial values	
1001	0011	0	0111	A←A − M	First cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second cycle
0101	0100	1	0111	A←A + M	Third cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth cycle

Pembagian Unsigned Binary Integer



- **Melibatkan shifting berulang, penambahan atau pengurangan**
- **Elemen:**
  - Dividend → bilangan yang dibagi
  - Divisor → pembagi
  - Quotient → hasil bagi
  - Remainder → sisa pembagian

Flowchart Pembagian Unsigned Binary Integer



Contoh Pembagian Twos Complement

Operasi Pembagian 7/3

A	Q	
0000	0111	Initial value
0000	1110	Shift
1101		Use twos complement of 0011 for subtraction
1101		Subtract
0000	1110	Restore, set $Q_0 = 0$
0001	1100	Shift
1101		
1110		Subtract
0001	1100	Restore, set $Q_0 = 0$
0011	1000	Shift
1101		
0000	1001	Subtract, set $Q_0 = 1$
0001	0010	Shift
1101		
1110		Subtract
0001	0010	Restore, set $Q_0 = 0$

Representasi Floating-Point - Principle

- **Fixed point (twos complement) → rentang bilangan bulat positif dan negatif yang berpusat atau mendekati 0**
- **Radix point → representasi bilangan pecahan (memisahkan bilangan bulat dan pecahan)**
- **Pendekatan ini memiliki keterbatasan. Angka yang sangat besar tidak dapat diwakili, begitu pula pecahan yang sangat kecil.**
- **Misal bilangan desimal mengatasinya dengan notasi ilmiah:**
  - 955.000.000 → 9.55×10^{8}
  - 0.0000000955 → 9.55×10^{-8}

Floating Point

Format bilangan biner:

$\pm S \times B^{\pm E}$

atau

Significand (mantissa) atau fraction

$(-1)^{sign} \times$ significand $\times 2^{exponent}$

(+ atau -) → Sign

S →

B → Base

E →

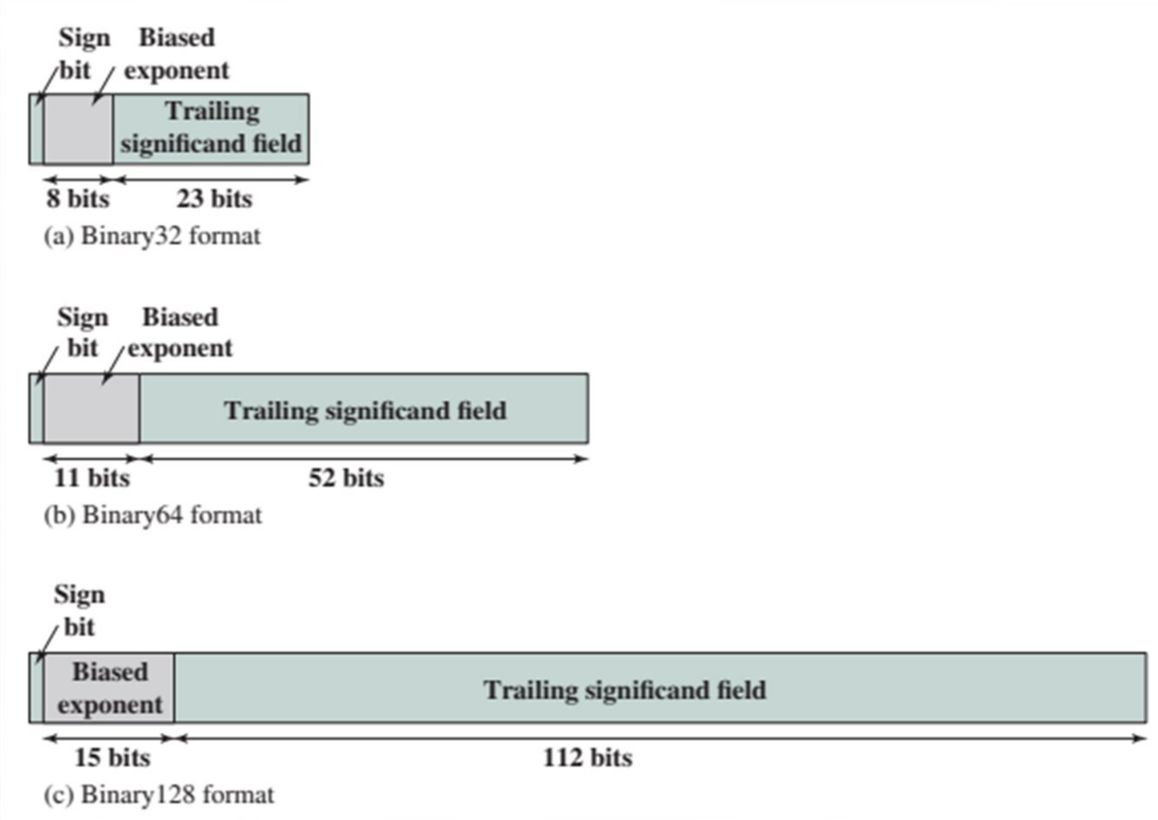
Exponent

Contoh:

Bilangan	Significand	Base	Exponent
3×10^6	3	10	6
110×2^8	110	2	8
6133.566	0.6133566	10	4

Format Standar IEEE 754

- **Representasi nilai:**
  $(-1)^{sign} \times (1.F) \times 2^{exponent-bias}$
- **32 bit → single precision**
- **64 bit → double precision**
- **Exponent → bias**
  - 127 → single precision (32 bit)
  - 1023 → double precision (64 bit)
  - Range 2^{-126} s/d 2^{+127}
  - Range 10^{-38} s/d 10^{+38}
- **Significand**
  - $(1+F) = 1.b_{-1}b_{-1}....b_{-23}$
- **Overflow: Eksponent membutuhkan lebih dari 8 bit. Bisa + atau –**
- **Underflow: Fraction membutuhkan lebih dari 23 bit. Bisa + atau –**



Format Standar IEEE 754 (lanj.)

Parameter	Format		
	Binary32	Binary64	Binary128
Storage width (bits)	32	64	128
Exponent width (bits)	8	11	15
Exponent bias	127	1023	16383
Maximum exponent	127	1023	16383
Minimum exponent	−126	−1022	−16382
Approx normal number range (base 10)	$10^{-38}, 10^{+38}$	$10^{-308}, 10^{+308}$	$10^{-4932}, 10^{+4932}$
Trailing significand width (bits)*	23	52	112
Number of exponents	254	2046	32766
Number of fractions	2^{23}	2^{52}	2^{112}
Number of values	1.98×2^{31}	1.99×2^{63}	1.99×2^{128}
Smallest positive normal number	2^{-126}	2^{-1022}	2^{-16362}
Largest positive normal number	$2^{128} - 2^{104}$	$2^{1024} - 2^{971}$	$2^{16384} - 2^{16271}$
Smallest subnormal magnitude	2^{-149}	2^{-1074}	2^{-16494}

Notasi Ilmiah – Bilangan Biner

- **Notasi Ilmiah**
  - $0.525 \times 10^5 = 5.25 \times 10^4 = 52.5 \times 10^3$
  - 5.25×10^4 merupakan notasi ilmiah yang dinormalisasi
    - menandakan posisi desimal pada fixed point
    - digit depan tidak 0
- **Bilangan Biner**
  - $5.25 = 101.01 = 1.0101 \times 2^2$
    - perkalian dengan pergeseran 2 titik ke kiri
    - pembagian dengan pergeseran 2 titik ke kanan

Tabel Nilai Exponent 32-bit

Eksponen (E)	Signifikan=0	signifikan≠0	Persamaan
0	0, -0	subnormal	$(-1)^S \times 0.\mathit{bit\ signifikan} \times 2^{-126}$
1-254	Nilai ternormalisasi		$(-1)^S \times 1.\mathit{bit\ signifikan} \times 2^{E-127}$
255	∞	bukan bilangan (NAN=not-a-number)	

$E_{min} = 1$, $E_{max} = 254$, menghasilkan eksponen (bias=127):
$E = 1 - 127 = -126$ dan $E = 254 - 127 = 127$

Nilai Spesial IEEE 754

- **Zero (nol)**
  - Nol adalah nilai khusus yang dilambangkan dengan eksponen dan significand 0. -0 dan +0 adalah nilai yang berbeda, meskipun keduanya sama.
- **Denormalised**
  - Jika eksponen semuanya 0, tetapi significant tidak, maka nilainya adalah bilangan yang dinormalisasi. Ini berarti angka ini tidak diasumsikan di depan titik biner.
- **Infinity (tak hingga)**
  - Nilai $+\infty$ dan $-\infty$ dilambangkan dengan eksponen semua 1 dan significand semua 0.
  - Sign bit membedakan antara infinity negatif dan infinity positif. Operasi dengan nilai infinity didefinisikan dengan baik di IEEE
- **Not a Number (NaN)**
  - Nilai NaN digunakan untuk merepresentasikan nilai yang merupakan kesalahan.
  - Eksponen semua 1 dengan sign bit 0 atau significant yang tidak diikuti oleh 0. Ini adalah nilai khusus yang mungkin digunakan untuk menunjukkan variabel yang belum memiliki nilai.

Infinity

- **Pembatas aritmatika aritmatika real, dengan notasi**

  $-\infty$ < bilangan terbatas < $+\infty$

$$
\begin{array}{ll}
5 + (+\infty) = +\infty & 5 \div (+\infty) = +0 \\
5 - (+\infty) = -\infty & (+\infty) + (+\infty) = +\infty \\
5 + (-\infty) = -\infty & (-\infty) + (-\infty) = -\infty \\
5 - (-\infty) = +\infty & (-\infty) - (+\infty) = -\infty \\
5 \times (+\infty) = +\infty & (+\infty) - (-\infty) = +\infty
\end{array}
$$

Not a Number (NaN)

Operation	Quiet NaN Produced By
Any	Any operation on a signaling NaN
Add or subtract	Magnitude subtraction of infinities: $(+\infty) + (-\infty)$ $(-\infty) + (+\infty)$ $(+\infty) - (+\infty)$ $(-\infty) - (-\infty)$
Multiply	$0 \times \infty$
Division	$\frac{0}{0}$ or $\frac{\infty}{\infty}$
Remainder	x REM 0 or ∞ REM y
Square root	$\sqrt{x}$, where $x < 0$

Format 32-bit Floating Point

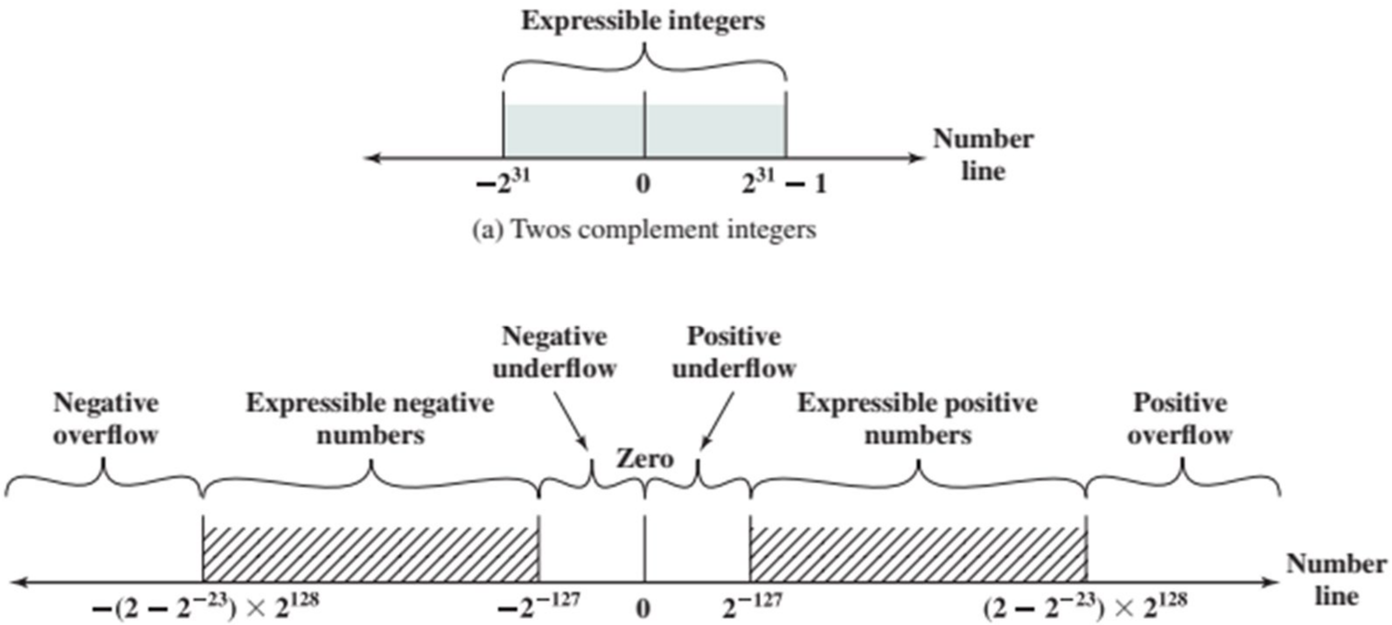
Bias exponent (0-255), bias 127 (0111111) dikurangi

$1.1010001 \times 2^{10100}$	= 0	10010011	10100010000000000000000	=	1.6328125×2^{20}
$-1.1010001 \times 2^{10100}$	= 1	10010011	10100010000000000000000	=	-1.6328125×2^{20}
$1.1010001 \times 2^{-10100}$	= 0	01101011	10100010000000000000000	=	1.6328125×2^{-20}
$-1.1010001 \times 2^{-10100}$	= 1	01101011	10100010000000000000000	=	$-1.6328125 \times 2^{-20}$

sign-bit bias exponent fraction bit (significand)

$1.1010001 = 2^{0} + 2^{-1} + 2^{-3} + 2^{-7} = 1 + 0.5 + 0.125 + 0.0078125$
$= 1.6328125$

Format 32 bit Expressible



(a) Twos complement integers

(b) Floating-point numbers

Contoh Biner ke Desimal

$(0100\ 0010\ 0101\ 0100\ 0000\ 0000\ 0000\ 0000)_2$

sign exponent significand

Petunjuk: Ubah ke format desimal dengan notasi ilmiah IEEE 754

Langkah:

1. Sign bit 0 → positif
2. Exponent $(10000110)_2 = 134$
   Exp. bias = 127
   e = 134 – 127 = 7
1. Significand/fraction
   $= 2^{-1} + 2^{-3} + 2^{-5} = 1/2 + 1/8 + 1/32 = 0{,}65625$
1. Hasil (bentuk umum) $= (-1)^S \times (1.F) \times B^e$
   $= (-1)^0 \times (1{,}65625) \times 2^7$
   $= 1{,}65625 \times 2^7$

Contoh Desimal ke Biner

$263{,}3_{(10)}$ → Desimal base 10 bilangan positif → sign bit 0
Petunjuk: Ubah ke format biner single precision IEEE 754
Langkah:

1. Konversi ke bilangan biner
   - (bag. integer → 263)
   - bagi dengan angka 2, catat setiap sisa
   - berhenti ketika hasil bagi = 0

263/2 = 131 sisa 1
131/2 = 65 sisa 1
65/2 = 32 sisa 1
32/2 = 16 sisa 0

Susun ulang nilai biner dari setiap sisa bag. bawah ke atas

Hasil susunan $263_{(10)}$:
$1\ 0000\ 0111_{(2)}$

loop

2. Konversi ke bilangan biner
   - (bag. pecahan → 0,3)
   - kali dengan angka 2
   - gunakan angka depan

0,3 x 2 = 0,6 → 0
0,6 x 2 = 1,2 → 1
0,2 x 2 = 0,4 → 0
0,4 x 2 = 0,8 → 0

Susun ulang nilai biner dari setiap angka depan, atas ke bawah

Hasil susunan $0{,}3_{(10)}$:
$0.0100\ 1100\ 1100\ 1100\ 1100\ 1100_{(2)}$

Contoh Desimal ke Biner (lanj.)

3. Kombinasikan bag. integer dan pecahan
- kemudian dinormalisasi dengan notasi ilmiah

  1 0000 0111.0100 1100 1100 1100 1100 1100 → hasil kombinasi

  1.0000 0111 0100 1100 1100 1100 1100 1100 x 2^8 → normasilasi
4. Sesuaikan exponent

  127+8 = 135, kemudian ubah ke biner → 1000 0111
5. Normalisasi significand
- hapus bit paling kiri dan sisa bit dari kanan (panjang 23 bit)

  ~~1~~.0000 0111 0100 1100 1100 110~~0 1100 1100~~
6. Bentuk umum = x yyyyyyyy zzzzzzzzzzzzzzzzzzzzzzz

  1 8 23

  x → sign bit, y → exponent, z → significand

  hasil akhir → 0100 0011 1000 0011 1010 0110 0110 0110

Aritmatika Floating Point

Floating-Point Numbers	Arithmetic Operations
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$\left.\begin{array}{l} X + Y = (X_S \times B^{X_E - Y_E} + Y_S) \times B^{Y_E} \\ X - Y = (X_S \times B^{X_E - Y_E} - Y_S) \times B^{Y_E} \end{array}\right\} X_E \leq Y_E$ $X \times Y = (X_S \times Y_S) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_S}{Y_S}\right) \times B^{X_E - Y_E}$

$$X = 0.3 \times 10^2 = 30$$

$$Y = 0.2 \times 10^3 = 200$$

$$X + Y = (0.3 \times 10^{2-3} + 0.2) \times 10^3 = 0.23 \times 10^3 = 230$$

$$X - Y = (0.3 \times 10^{2-3} - 0.2) \times 10^3 = (-0.17) \times 10^3 = -170$$

$$X \times Y = (0.3 \times 0.2) \times 10^{2+3} = 0.06 \times 10^5 = 6000$$

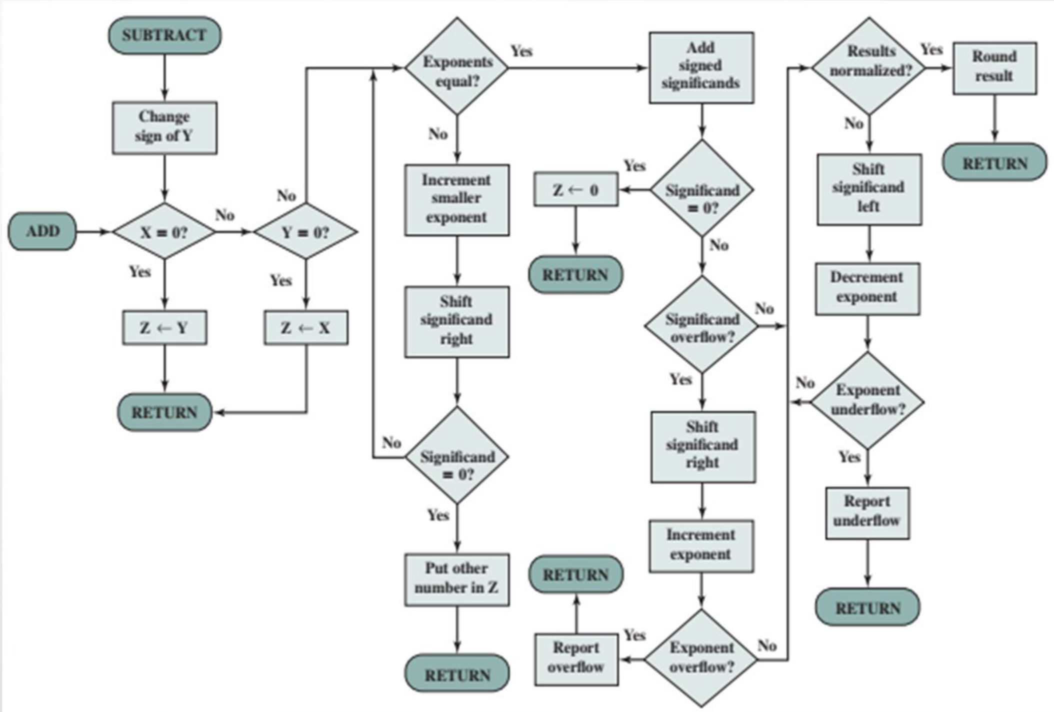
$$X \div Y = (0.3 \div 0.2) \times 10^{2-3} = 1.5 \times 10^{-1} = 0.15$$

Operasi Aritmatika Floating Point

- **Beberapa issues yang muncul pada operasi aritmatika floating point:**
  - Exponent overflow
    - Sebuah exponent positif yang melampaui nilai exponent maksimum. Dalam beberapa sistem ditandai dengan +∞ atau -∞
  - Exponent underflow
    - Sebuah exponent negatif yang melampaui nilai exponent minimum (-200 kurang dari -127). Hal ini berarti bilangan terlalu kecil untuk dapat direpresentasikan, dapat dinyatakan sebagai 0
  - Significand overflow
    - Dalam penambahan dua significand yang memiliki sign sama dapat menghasilkan carry out pada MSB
  - Significand underflow
    - Dalam proses penyejajaran significand,digit dapat mengalir ke ujung kanan significand. Hal ini diperlukan pembulatan nilai (rounding)

Operasi Penambahan dan Pengurangan Floating Point

- **Ada 4 fase:**
  - Cek untuk zero
  - Penyejajaran significand
    - Pilih angka exponent terkecil (geser significand agar sama dengan exponent terbesar)
    - Atur exponent dari hasil agar sama dengan exponent terbesar
  - Penambahan/pengurangan
    - Operasi penambahan/pengurangan significand dan penentuan sign dari hasil
  - Normalisasi



Penambahan Floating Point

$1.1100 \times 2^4 + 1.1000 \times 2^2$

1). Cek zero

$1.1100 \times 2^4 = 1.75 \times 2^4$

$1.1000 \times 2^2 = 1.5 \times 2^2$

Jika dijumlah hasil 34

2). Penyejajaran significand

$1.1000 \times 2^2 \rightarrow 0.0110 \times 2^4$

3). Penambahan significand

$1.1100 \quad \times 2^4$

$0.0110 \quad \times 2^4$

-------------- +

$10.0010 \quad \times 2^4 \rightarrow 2.125 \times 2^4$

4). Normalisasi hasil

$1.00010 \times 2^5 \rightarrow 1.0625 \times 2^5$ Jika dijabarkan nilai 34

Pengurangan Floating Point

$1.1100 \times 2^4 - 1.1000 \times 2^2$

1). Cek zero

$1.1100 \times 2^4 = 1.75 \times 2^4$

$1.1000 \times 2^2 = 1.5 \times 2^2$

Jika dikuranghasil 22

2). Penyejajaran significand

$1.1000 \times 2^2 \rightarrow 0.0110 \times 2^4$

3). Penambahan significand

$1.1100 \quad \times 2^4$

$0.0110 \quad \times 2^4$

-------------- –

$1.0110 \times 2^4 \rightarrow 2.125 \times 2^4$

4). Normalisasi hasil

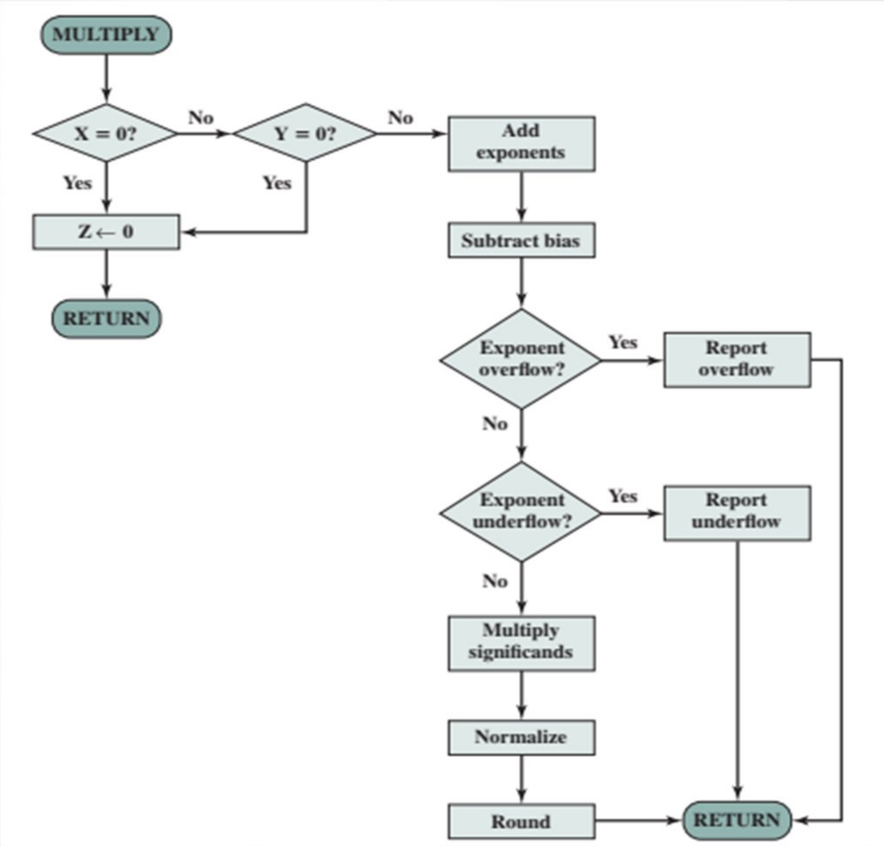
$1.0110 \times 2^4 \rightarrow 1.375 \times 2^4$

Jika dijabarkan nilai 22

Perkalian Floating Point

Langkah-langkah:

1. Cek zero
2. Memisahkan significand dari exponent
3. Kalikan bagian significand bersama
4. Tambahkan exponent bersama
5. Kombinasikan 2 hasil
6. Normalisasi



Perkalian Floating Point

$1.1100 \times 2^4 \times 1.1000 \times 2^2$

1). Cek zero 4).

Tambahkan exponent

$1.1100 \times 2^4 = 1.75 \times 2^4$ Jika dikali hasil 168 4 + 2 =

6

$1.1000 \times 2^2 = 1.5 \times 2^2$ 5).

Kombinasikan hasil poin 3 dan 4 Jika dijabarkan nilai 168

2). Memisahkan significand dengan exponent 10.10100000×2^6

Significand 1.1100 dan 1.1000 6). Normalisasi hasil

Exponent 4 dan 2 1.010100000 x

2^7

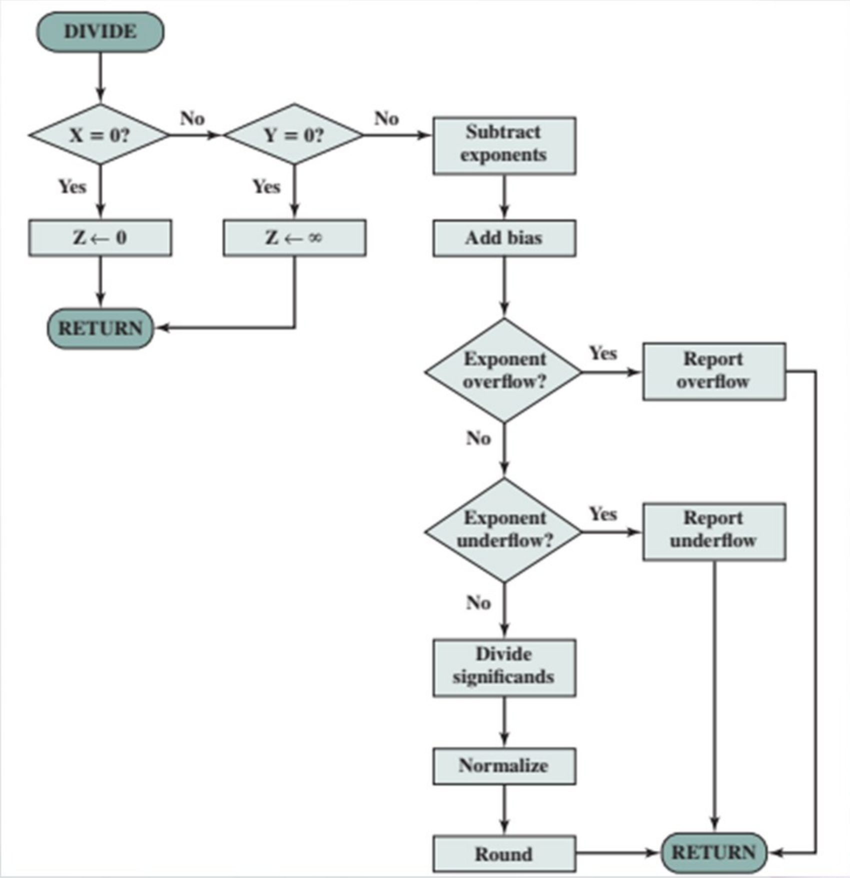
3). Kalikan bagian significand

```
                 1.1 100
                 1.1000
        ------ x
        0 0 0 0 0
     0 0 0 0 0
    0 0 0 0 0
   1 1  1  0 0
  1 1 1 0 0
----------------- +
   10 10 1 0 0 0 0 0
```

Pembagian Floating Point

Langkah-langkah:

1. Cek zero
2. Memisahkan significand dari exponent
3. Bagi bagian significand bersama
4. Kurangkan exponent bersama
5. Kombinasikan 2 hasil
6. Normalisasi



Pembagian Floating Point

$1.1110 \times 2^4 / 1.1000 \times 2^2$

1). Cek zero

Kurangkan exponent

$1.1110 \times 2^4 = 1.875 \times 2^4$ Jika dibagi hasil 5

$1.1000 \times 2^2 = 1.5 \times 2^2$

Kombinasikan hasil poin 3 dan 4

2). Memisahkan significand dengan exponent

Significand 1.1110 dan 1.1000

Exponent 4 dan 2

3). Bagi bagian significand

1.1 1 1 0

1.1000

------ :

1.0100

4). 4 – 2 = 2

5). Jika dijabarkan nilai 5

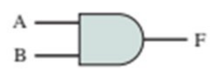
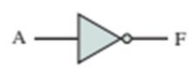
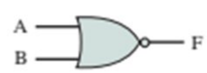
1.0100×2^2

6). Normalisasi hasil

1.0100×2^2

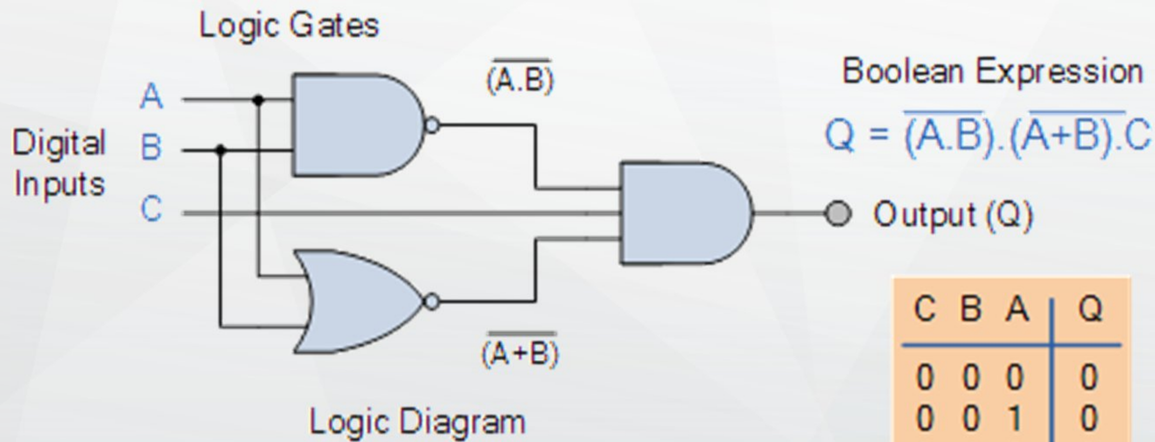
Gerbang Logika

- **Rangkaian elektronik yang menghasilkan sinyal keluaran yang merupakan operasi Boolean sederhana pada sinyal masukannya**

Name	Graphical Symbol	Algebraic Function	Truth Table
AND		$F = A \cdot B$ or $F = AB$	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = A + B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F = \overline{A}$ or $F = A'$	A | F 0 | 1 1 | 0
NAND		$F = \overline{AB}$	A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{A + B}$	A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR		$F = A \oplus B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Rangkaian Kombinasional

- **Sekumpulan gerbang yang saling berhubungan yang outputnya setiap saat hanya merupakan fungsi dari input pada saat itu.**
- **Rangkaian kombinasional terdiri dari n masukan biner dan m keluaran biner.**

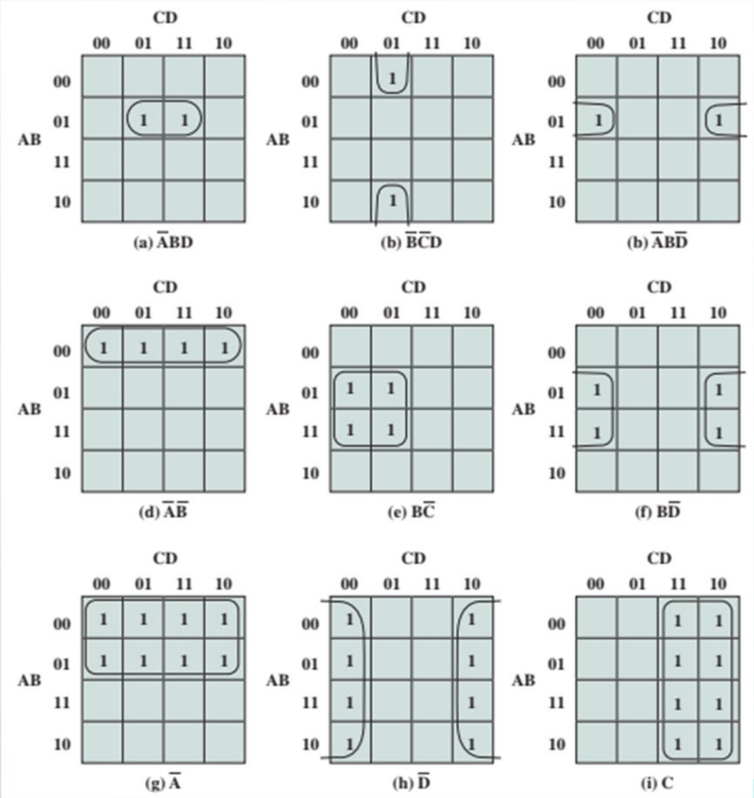


Typical Truth Table

C	B	A	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

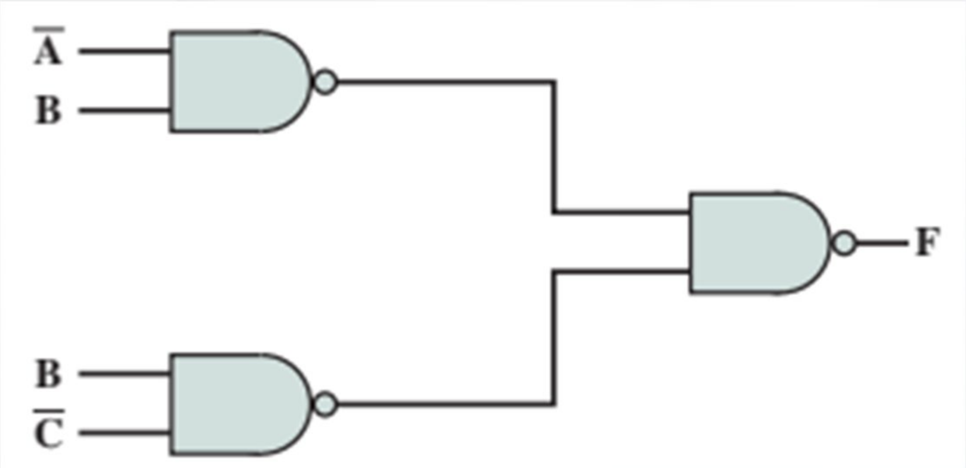
Karnaugh Maps

- **Cara mudah untuk merepresentasikan fungsi Boolean dari sejumlah kecil (hingga empat) variabel.**



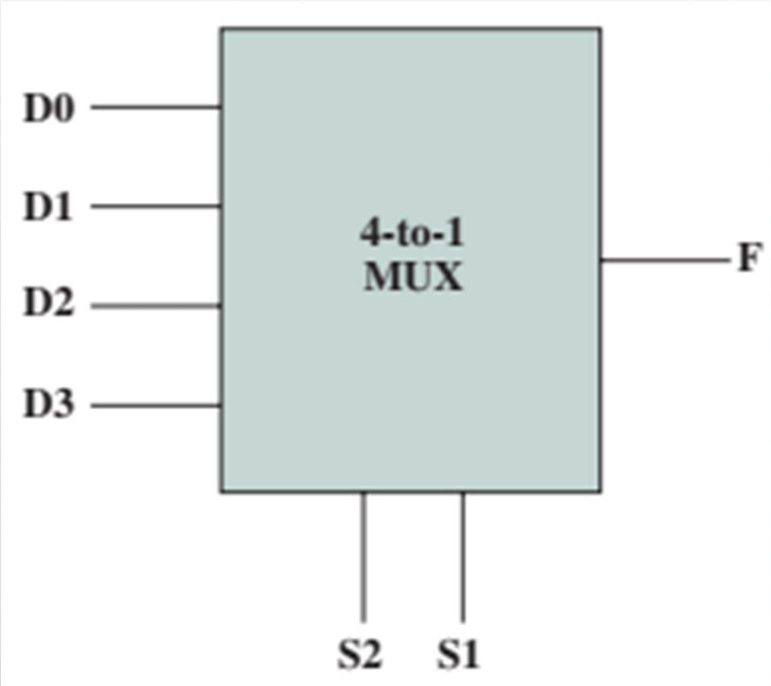
Implementasi NAND dan NOR

- Pertimbangan lain dalam implementasi fungsi Boolean menyangkut jenis gerbang yang digunakan.
- Terkadang untuk mengimplementasikan fungsi Boolean hanya dengan gerbang NAND atau hanya dengan gerbang NOR.
- Meskipun ini mungkin bukan implementasi gerbang minimum, hal ini memiliki keunggulan keteraturan, yang dapat menyederhanakan proses pembuatan.



Multiplexers

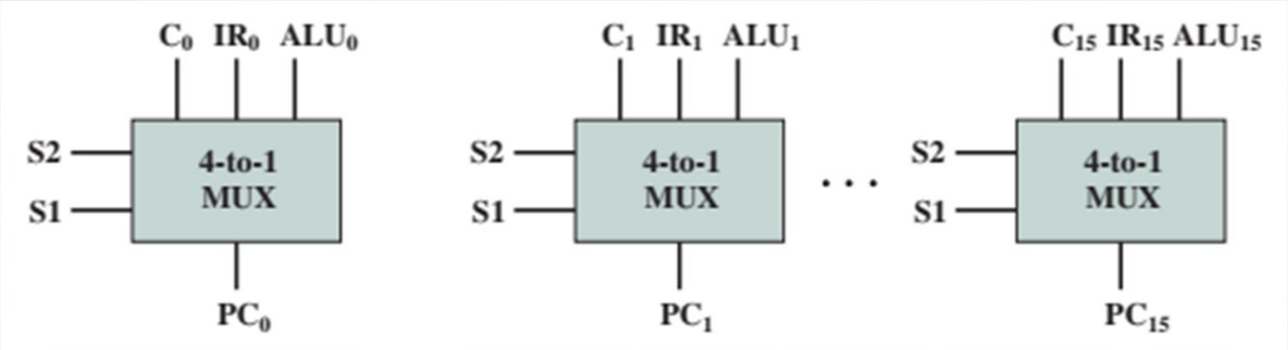
- Menghubungkan banyak input ke satu output
- Untuk memilih satu dari beberapa input, dibutuhkan selector



S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Input Multiplexer ke Program Counter

- **Multiplexer → pengendali sinyal dan data routing, contohnya pemuatan Program Counter (PC)**
- **Nilai yang dimuat ke PC bisa dari beberapa sumber:**
  - Binary counter → PC ditambah untuk next instruction
  - Instruction Register (IR) → instruksi cabang yang menggunakan direct address yang telah dieksekusi
  - Output dari ALU → instruksi cabang menentukan alamat menggunakan displacement mode



Decoders

- Rangkaian kombinasional dengan sejumlah jalur output, hanya satu yang ditetapkan setiap saat. Jalur output mana yang ditegaskan bergantung pada pola jalur input.
- Secara umum, decoder memiliki n input dan 2^n output.

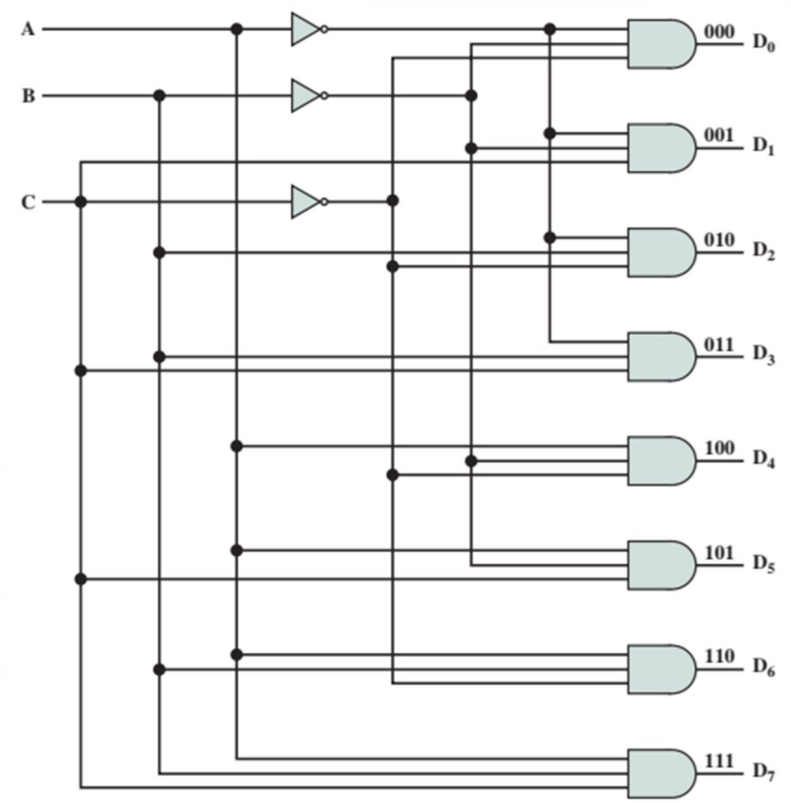
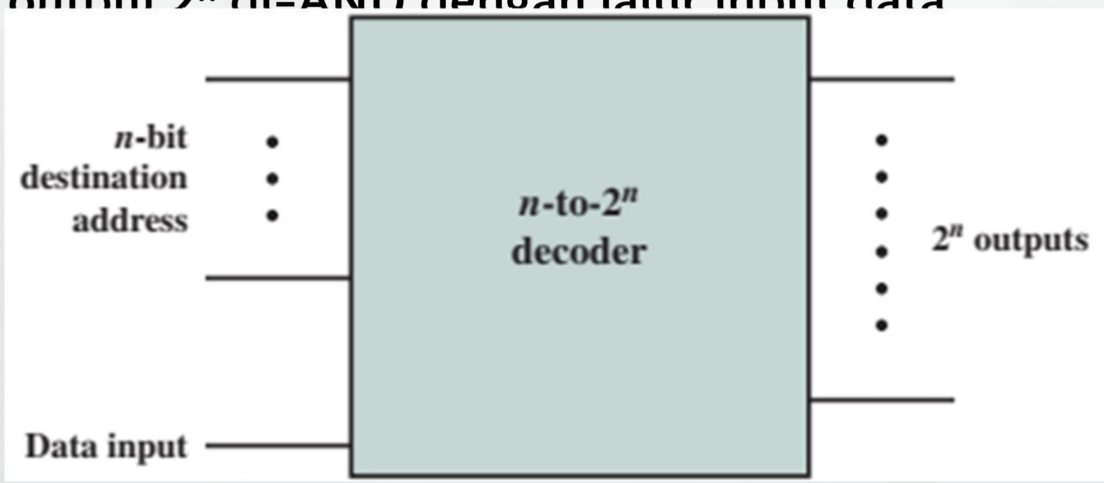


Figure 11.15 Decoder with 3 Inputs and $2^3 = 8$ Outputs

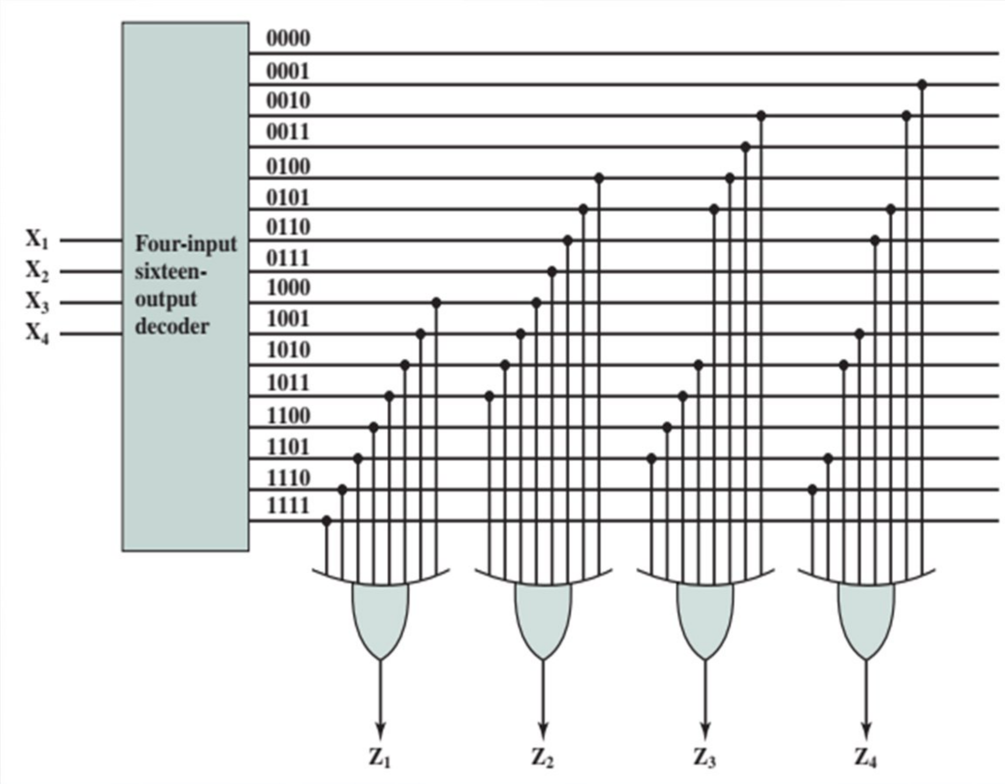
Implementasi Demultiplexer

- Dengan jalur input tambahan, dekoder dapat digunakan sebagai demultiplexer.
- Demultiplexer melakukan fungsi kebalikan dari multiplexer, menghubungkan satu input ke salah satu dari beberapa output.
- Semua jalur output 2^n di-AND dengan jalur input data



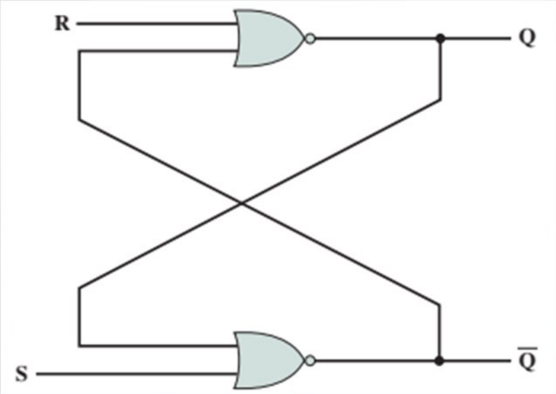
Read Only Memory (ROM)

- Rangkaian kombinasional → rangkaian "memoryless" karena keluarannya hanya bergantung pada input saat ini dan tidak ada riwayat input sebelumnya yang dipertahankan.
- ROM → memori yang menggunakan rangkaian kombinasional, hanya melakukan operasi read (bisa terdiri dari dekoder dan gerbang OR).
- Informasi biner yang disimpan dalam ROM bersifat permanen dan dibuat selama proses fabrikasi.

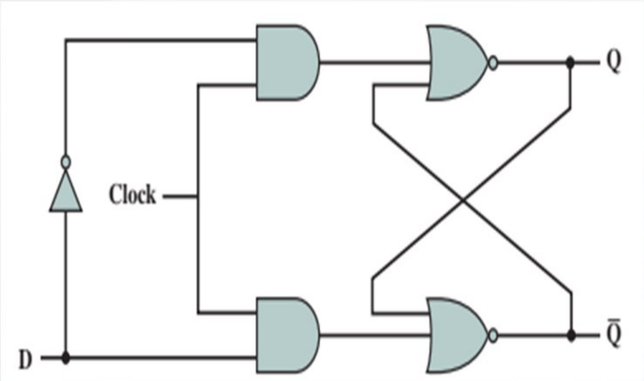


Flip-Flop

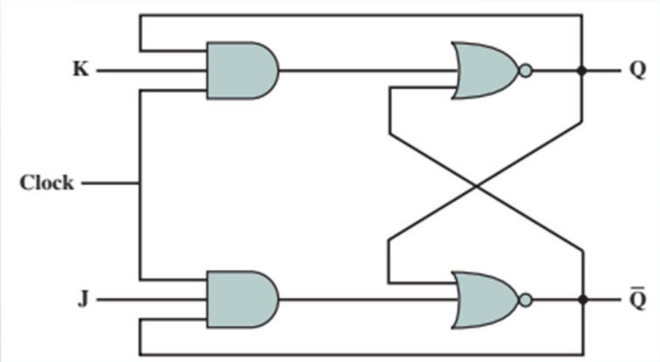
- **Bentuk rangkaian sekuensial yang sederhana**
- **Memiliki sifat yang sama untuk semua jenis flip-flop**
    - Perangkat bistable. Ada di salah satu dari dua state dan jika tidak ada input, tetap dalam state itu. Dengan demikian, flip-flop dapat berfungsi sebagai memori 1-bit.
    - Memiliki dua output yang selalu melengkapi satu sama lain.



S-R flip-flop

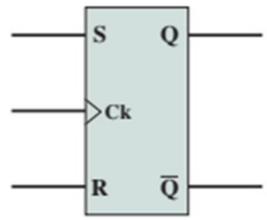
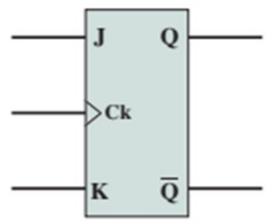
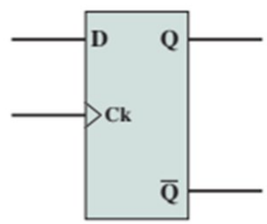


D flip-flop



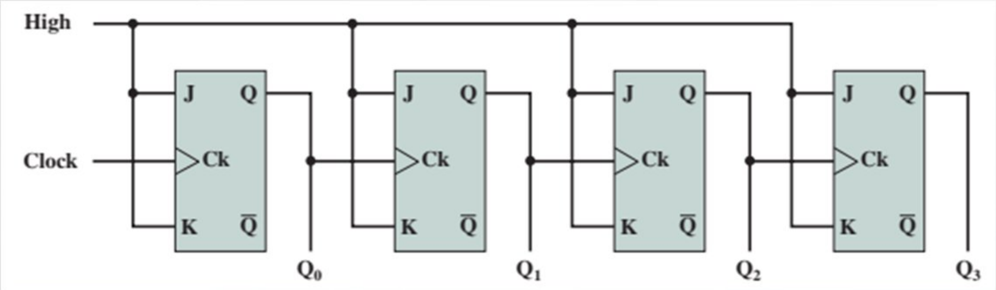
J-K flip-flop

Tabel Kebenaran Flip-Flop

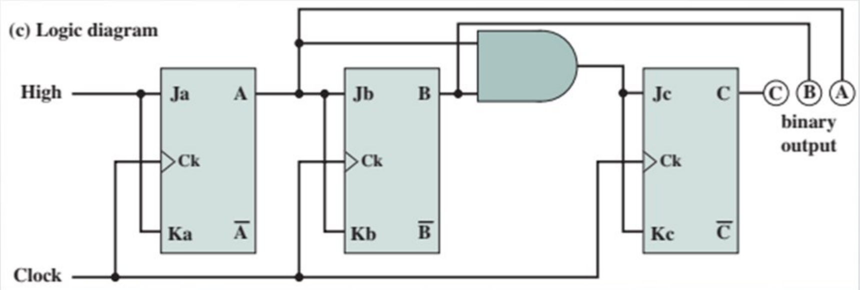
Name	Graphical Symbol	Truth Table
S–R		S R Q_{n+1} 0 0 Q_n 0 1 0 1 0 1 1 1 –
J–K		J K Q_{n+1} 0 0 Q_n 0 1 0 1 0 1 1 1 $\overline{Q_n}$
D		D Q_{n+1} 0 0 1 1

Counter

- **Register yang nilainya dapat dengan mudah bertambah 1 modulo kapasitas register; yaitu, setelah nilai maksimum tercapai, kenaikan berikutnya menetapkan nilai pencacah menjadi 0**
- **Register terdiri dari n flip-flop yang dapat menghitung 2^n - 1 → program counter**
- **Ada 2 jenis:**
  - Asynchronous → relatif lambat
  - Synchronous → lebih cepat, sering dipakai di CPU



Asynchronous



Synchronous

TERIMA KASIH