COMPUTER ARITHMETIC

Danang Wahyu Utomo
danang.wu@dsn.dinus.ac.id
+6285 725 158 327

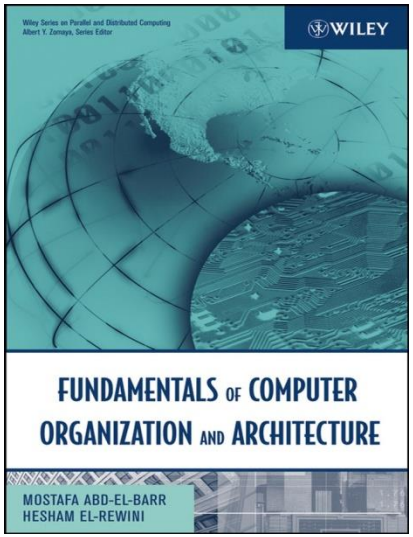
RENCANA KEGIATAN PERKULIAHAN SEMESTER

W	Pokok Bahasan
1	Organisasi dan Arsitektur Komputer
2	Sistem Komputer
3	Instruction Set Architecture and Design
4	
5	**Computer Arithmetic**
6	
7	Review Materi 1-6
8	**Ujian Tengah Semester**

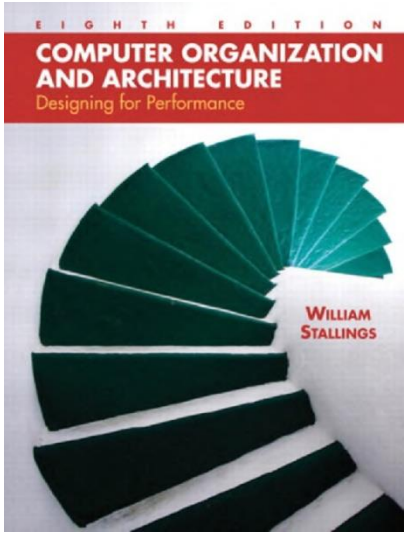
W	Pokok Bahasan
9	Desain Unit Pemrosesan
10	
11	Desain Sistem Memory
12	
13	Desain dan Organisasi Input Output
14	
15	Teknik Desain Pipelining
16	**Ujian Akhir Semester**

Reference

- Mustafa Abd-el-Bhar, Hesham El Rewini – Fundamentals of Computer Organization and Architecture 9th Edition (2004)



- William Stallings – Computer Organization and Architecture Designing For Performance 8th Edition (2009)



Content

- Number System
- Integer Arithmetic
- Floating-Point Arithmetic

Number System

- Decimal
- Binary
- Hexadecimal
- Octal

Example :

$255_{(10)}$ = ……$_{(2)}$

$44_{(10)}$ = ……$_{(16)}$

$1111_{(2)}$ = ……$_{(16)}$

Sign Magnitude

- Alternative conventions used to represent negative as well as positive integers.
- The most significant (leftmost) bit in the word as a sign bit
- If the sign bit is 0, the number is positive
- If the sign bit is 1, the number is negative
- Example :
  - + 18 = **0**0010010
  - - 18 = **1**0010010

Two Complement Representation

- Two complement representation uses the most significant bit as a sign bit
- Two steps of two complement representation:
  - Perform the boolean complement of each bit (including the sign bit), that is set each 1 to 0 an each 0 to 1.
  - Add 1 to the least significant bit. Treating the result as an unsigned binary integer.

+ 18	= 00010010 (two complement)
Bitwise complement	= 11101101
	+ 1
	11101110

Example

- - 18 ?
- + 7 ?
- - 7 ?

Addition and Subtraction

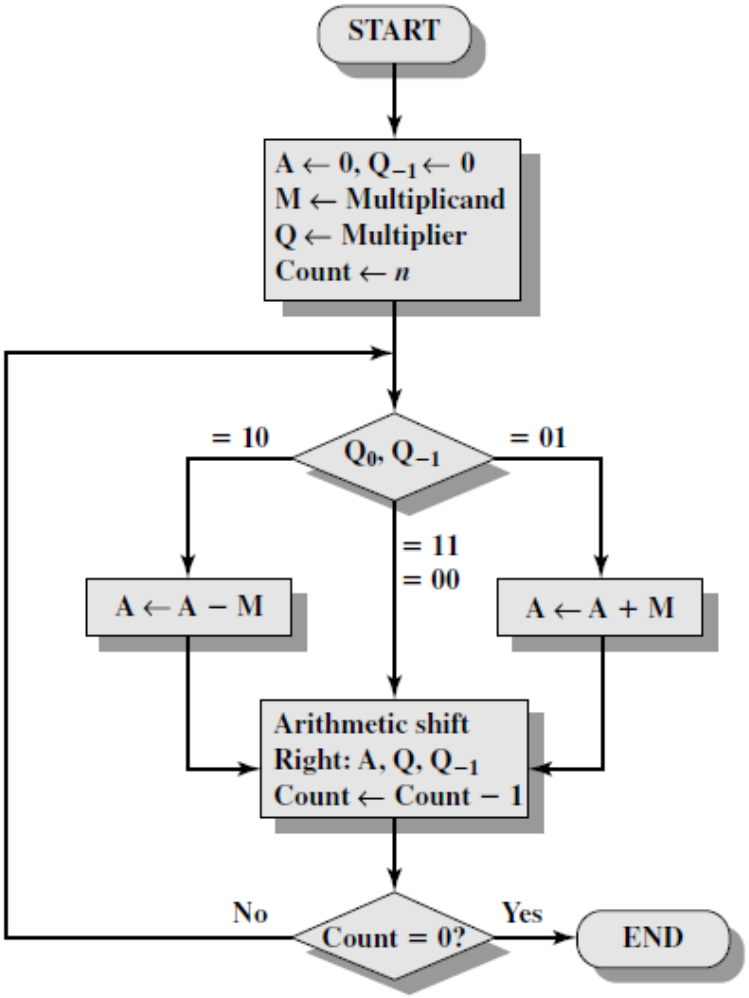
- Addition and subtraction in twos complement
- The result may be larger than can be held in the word size (-2^{n-1} through $2^{n-1} - 1$) being used is called **OVERFLOW**

Overflow Rule :
If two numbers are added, and they are both positive or both Negative, then overflow occurs if and only if the result has the opposite sign

Example 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)

Booth's Algorithm



Exampe 3 X 7

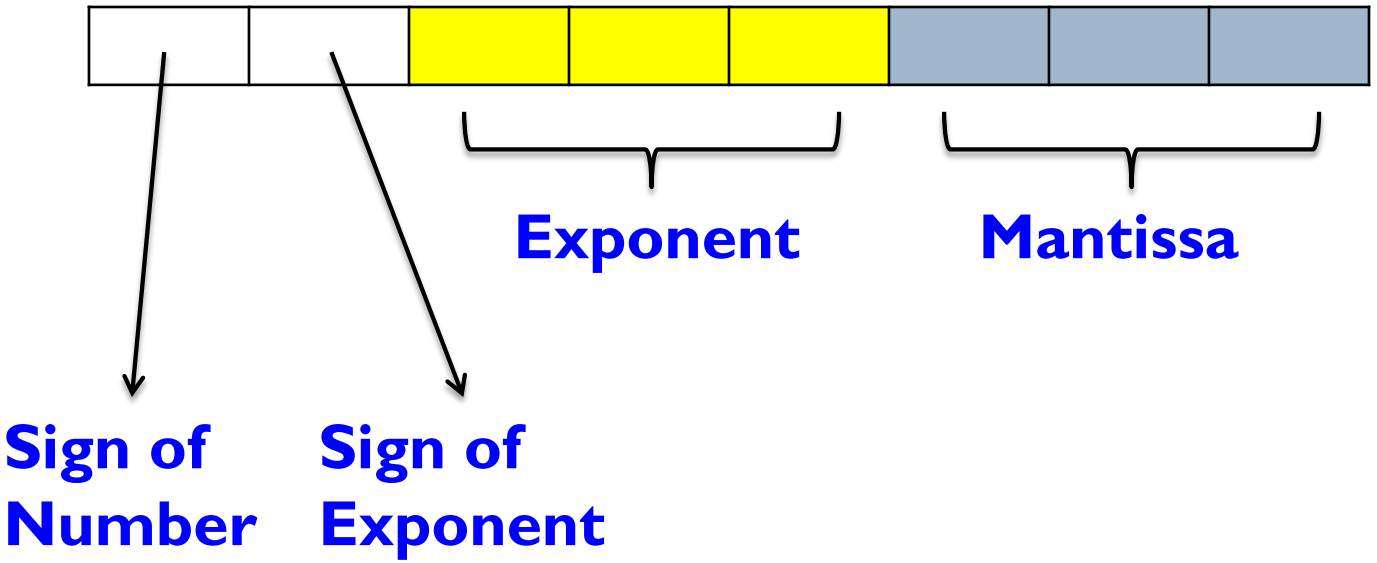
A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial values	
1001	0011	0	0111	A ← A − M	First cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second cycle
0101	0100	1	0111	A ← A + M	Third cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth cycle

Latihan

- -3 X 7
- 3 x -7
- -3 x -7
- 4 x 4
- -4 x -4

Floating Point Representation

- 8 bit word



Example

- $-13.9_{(10)}$?

-13.9 ???

- $-(13.9)_{10} = 13_{10} + 0.9_{10}$ ⟶ **Representation in binary number**

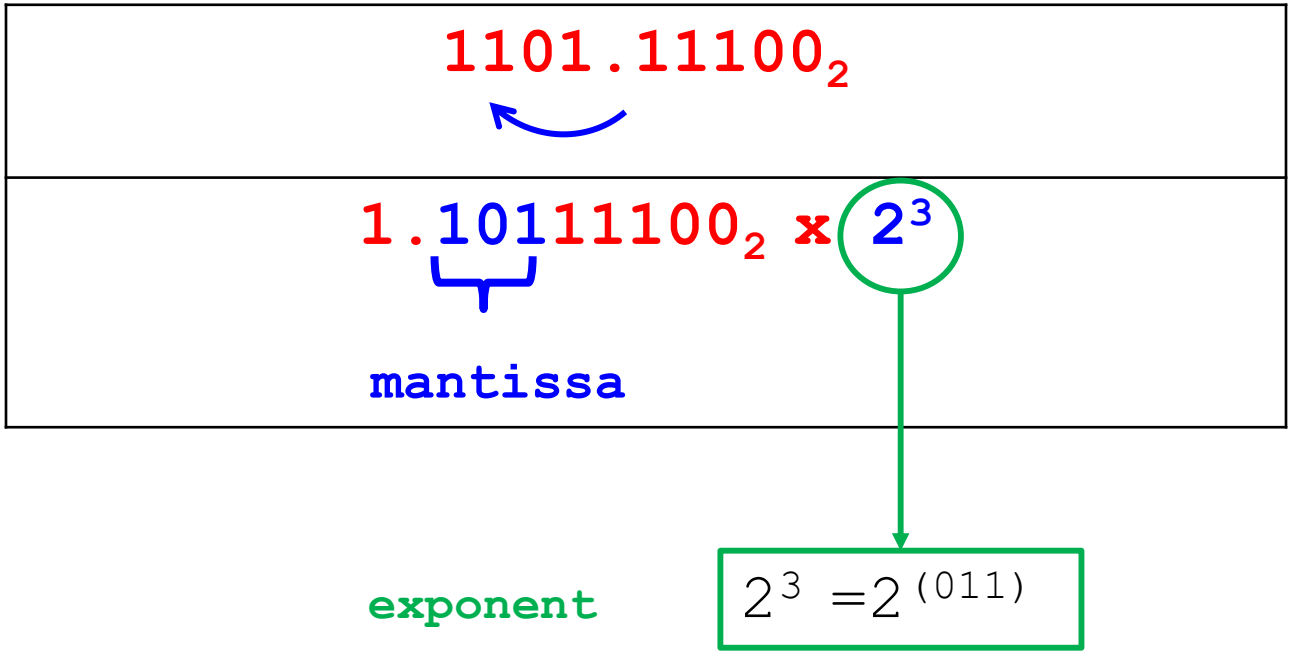
$\mathbf{13_{10}}$ → (2)	$\mathbf{0.9_{10}}$ → (2)
› 1101_2	› 0.11100_2 **How ???**

-13.9 ???

- $-(13.9)_{10} = 13_{10} + 0.9_{10}$ ⟶ **Representation in binary number**

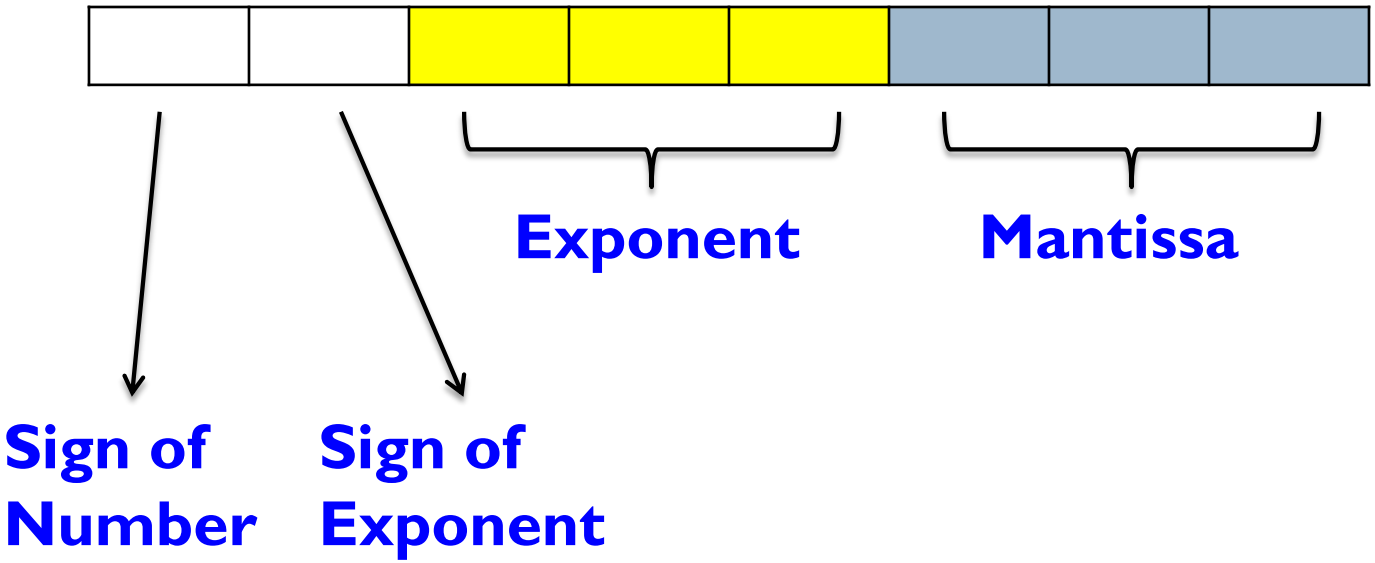
13_{10} → (2)	0.9_{10} → (2)
› 1101_2	› 0.11100_2 **How ???**
1101.11100_2	

-13.9 ???



Floating Point Representation

- $-(13.9)_{10} = -\ (1.\mathbf{101}111 \times 2^{(\mathbf{011})})$



TERIMA KASIH