Welcome to ACE Engineering Academy - online live class

Subject: **Computer Organization and Architecture**

Faculty: **Y.V. Ramaiah**

Subject

Computer organization & Architecture

Chapters (Topics)

I. Computer Arithmetic

II. Memory Organization

III. Secondary Memories

IV. Basic processor organization and Design

V. Pipeline organization

VI. Control unit Design

VII. IO Organization

Recommended books for COA subject

Chapter (Topic)	Author
I. Computer Arithmetic	CARL Hamachar & william stallings
II. Memory Organization	Nicholas CARTER & PAUL Chaudhuri
III. Secondary Memories	CARL Hamachar & William stalling
IV. Basic processor organization and Design	Morris mono & Hamachar
V. Pipeline organization	Nicholas carter
VI. Control unit Design	Nicholas carter, Morris mono, Paul Chaudhuri
VII. IO Organization	Morris Mono

VVV Imp

Low weightage

Author Name	Title of the Book
CARL Hamachar	Computer Organization
Nicholas Carter	Computer Architecture
Morris mono	Computer System Architecture
William stallings	Computer Organization and Architecture (6th edition old)
PAUL chaudhuri	Computer Organization and Design

C.O.A. subject weightage in marks (chapter-wise) in GATE

from 2016 to ~~2021~~ 2022

Name of the Chapter	2016		2017		2018	2019	2020	2021		2022
	Set I	Set II	Set I	Set II				Set I	Set II	
I. Computer Arithmetic	1	0	1	1	1	0	1	2	2	2
II. Memory Organization	1	1	2	3	2	3	3	1	2	3
III. Secondary Memories	0	0	1	0	1	0	0	0	0	0
IV. Basic Processor Organization & Design	0	2	2	0	2	0	2	1	0	0
V. Pipeline Organization	1	2	1	0	1	0	1	1	1	1
VI. Control Unit Design	0	0	1	0	0	0	1	0	0	0
VII. IO Organization	1	0	0	0	0	0	1	0	1	1
Total No. of Questions	**4**	**5**	**8**	**4**	**7**	**3**	**9**	**5**	6	7

Chapter - 1
Computer Arithmetic

1. Signed binary data representation (Integer)
2. Overflow concept in signed 2's complement representation
3. Different Arithmetical operations
4. Booth's Algorithm
5. Floating point representation
6. IEEE standards for floating point representation
7. Ripple carry Adder
8. Carry lookahead generator and Adder

Q. Let R1 and R2 be two 4-bit registers that store numbers in 2's complement form. For the operation R1+R2, which one of the following values of R1 and R2 gives an arithmetic overflow?

(A) R1 = 1011 and R2 = 1110

(B) R1 = 1100 and R2 = 1010

(C) R1 = 0011 and R2 = 0100

(D) R1 = 1001 and R2 = 1111

-8 to +7

-8 to +7

-5 + -2 = -7 No OVFL

-4 + -6 = (-10) Yes

+3 + +4 = +7 No

-7 + -1 = (-8) No

on Dec 26th 2021.
Expected gate 2022 Questions live session on Youtube
Started with OVFL Question.

Signed Binary Data Representation (Integer)

unsigned

25

10110_2

Value

Signed

+25

−25

MSB

10110

5 bit

Value

Sign bit

let MSB = '0', +ve

MSB = 1, −ve

26 825

INT M

unsigned Integer Conversions

25 = 16 8 4 2 1

1 1 0 0 1

$25 = (11001)_2$

2 | 25

2 | 12 − 1

2 | 6 − 0

2 | 3 − 0

2 | 1 − 1

64 32 16 8 4 2 1

$78 = 1001110_2$

$139 = 10001011_2$

Binary to Decimal conversion

$11011 = (27)_{10} = 27 = 11011_2$

$2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$(1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

$16 + 8 + 0 + 2 + 1$

n bit unsigned $(a_{n-1} \quad a_2\ a_1\ a_0)_2 = (\quad)_{10}$

$2^{n-1} \qquad 2^2\ 2^1\ 2^0$

$(a_{n-1}) * (2^{n-1}) + \ldots (a_2 * 2^2) + (a_1 \times 2^1) + (a_0 * 2^0)$

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

Unsigned Fraction Conversions

42·8125

Fraction

FIFO

$26.8125 = (11010.1101)_2$

.8125 × 2	
1.6250 × 2	
1.2500 × 2	
0.5000 × 2	
1.0	

Self check

1	1	0	1	1
16	8	4	2	1
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$		$\frac{1}{16}$

$26.8125 \quad \frac{8+4+0+1}{16} = \frac{13}{16}$

$(\overset{MSB}{a_1} a_2 a_3 \cdots \overset{LSB}{a_n})_2 = (\quad)_{10}$

$\frac{1}{2} \quad \frac{1}{4} \quad \frac{1}{8} \qquad \frac{1}{2^n}$

$2^{-1} \quad 2^{-2} \quad 2^{-3} \qquad 2^{-n}$

$(a_1 \times 2^{-1}) + (a_2 \times 2^{-2}) + \cdots (a_n \times 2^{-n})$

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

$0.111_2 = (\quad)_{10}$

$\frac{1}{2}\ \frac{1}{4}\ \frac{1}{8}$

$\frac{4+2+1}{8} = \frac{7}{8} = 1 - \frac{1}{8} \cong 1$ (LSB)

$= 1 - \text{LSB Value}$.

Note:- When Fraction field is filled with all '1's, then it's Decimal Value $\cong 1$ but accurate Value $= 1 - \text{LSB}$

$0.11111_2 = (\quad)_{10}$

$\frac{1}{2}\ \frac{1}{4}\ \frac{1}{8}\ \frac{1}{16}\ \frac{1}{32}$

$\frac{16+8+4+2+1}{32} = \frac{31}{32}$

$= 1 - \frac{1}{32} \cong 1$

$= 1 - \text{LSB Value}$

$0.\underset{\text{MSB}}{\frac{1}{2}}\ \frac{1}{4}\ \frac{1}{8}\ \frac{1}{16}\ \underset{\text{LSB}}{\frac{1}{32}}$

Q) The n bit fixed point Representation of an unsigned Real Number (x) uses 'f' bits for the fraction field Let $i = (n-f)$. The Decimal Value Range for (x) in this Notation is

a) $(2^{-f}$ to $2^{-i})$ b) 2^{-f} to $(2^{i} - 2^{-f})$

c) 0 to 2^{i} d) 0 to $(2^{i} - 2^{-f})$

Int . Fraction

$XXXX . XXX \rightarrow$ Real Number = x

$x \cdots x\,x\,x\,x\,x \cdot \underbrace{x\,x\,x\,x \cdots x}_{\bar{2}^1\ \bar{2}^2 \cdots \bar{2}^f}$

$i = (n-f)$

$i = n-f$

fixed point Notation

$(0 \text{ to } 2^i - 1)$ $\quad$ $0 \text{ to } (1 - \bar{2}^f)$

Total Range

$0 \text{ to } 2^i - 1 + 1 - \bar{2}^f$

$0 \text{ to } (2^i - \bar{2}^f)$

$0.111\cdots1 \simeq 1$

$\simeq (1 - \bar{2}^f)$

Integer Range (n bit)

min: 0 0 0 0 ⋯ 0 : 0

to

max 1 1 1 1 ⋯ 1 : $(2^n - 1)$

When Integer is filled with all '1's, it's Decimal value = $(2^n - 1)$

Int fraction.

25 . 525

fixed point Notation.

=

25.525 $\times 10^0$

Floating point Representation.

Signed Data can be Represented in 3 ways

(i) Signed magnitude Notation

(ii) Signed 1's Complement Notation

(iii) Signed 2's Complement Notation

most of systems use this.

Signed 2's Complement Notation:

→ It is free from Negative Zero

→ It provides more range for Negative Values than other Notations

→ It permits 8421 weighted code while finding it's Decimal value; in this process only MSB weight is Negative → It permits Sign bit Extension technique

4bit data

S					
+ve	0	0	0	0	+0
					to
	0	1	1	1	+7
-ve	1	0	0	0	-8
	1	1	1	1	-1

-8 to $+7$

-2^{3} to $+2^{3}-1$

-2^{4-1} to $+2^{4-1}-1$

for nbit, Range; $-(2^{n-1})$ to $+(2^{n-1}-1)$

The following values are in 2's Complement Notation, Their decimal values are

(i) 1101

4 bit

MSB = 0 +ve

MSB = 1 -ve

1 101

2's (-) 101 + 1

(-) 010 + 1

= (-) 011 = -3

(ii) 0 1110

+ve

(+) 1110 = +14

8 4 2 1

MSB

0 1110 = +14

-16 8 4 2 1

1101 = -8+5

-8 4 2 1 = -3

Let 11101 is in Signed 2's Complement Notation; it's Decimal Value is ____.

1 1 1 0 1

-32 16 8 4 2 1

= -32 + 27

= -5

(Ex) Let 1111111 is in signed 2's Complement Notation, it's Decimal Value is ____.

1 1 1 1 1 1 1 = -64 + 63
-64 32 16 8 4 2 1 = (-1).

Ex What is the signed 2's Complement Notation for -28, -14 and -7

For (-7): S = (-), 7 = 111, $\bar{7}$ = 000, $\bar{7}$+1 = 001

(-7) = 1001 = -7.
-8 1

(-7) = -8 + 1 =

-8	4	2	1
1	0	0	1

$-14 = -16 + 2 =$ $\begin{matrix} -16 & 8 & 4 & 2 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{matrix}$

$-28 = -32 + 4 =$ $\begin{matrix} -32 & & & 4 & \\ 1 & 0 & 0 & 1 & 0 \end{matrix}$ 0

$-126 = -128 + 2$

$= \begin{matrix} -128 & & & & & & 2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix}$

(Ex) 2's Complement Notation for -49 is

$-49 = -64 + 15 = -64 + 8 + 4 + 2 + 1$

$\begin{matrix} -64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix}_2$

$-49 = \overset{-64}{1}\,0\,0\,1\,1\,1\,1$

(Ex) 2's complement Notation for -22 is

$\overset{-32}{1}\,0\,\overset{8}{1}\,0\,\overset{2}{1}\,0 = -22$

Sign bit extension :- It is the process of converting the smaller size signed Data to Larger size by padding the sign bit to left. All these additional padded bits are known as Guard bits or Dummy bits.

The below data are represented in Signed 2's Complement Notation

$-8\ 4\ 2\ 1$

$1001 = -7$ (4 bit)

S

$111001 = -7$ (6 bit)

$-32\ 16\ 8\ 4\ 2\ 1$

$11111001 = -7$ (8 bit)

$-128\ 64\ 32\ 16\ 8\ 4\ 2\ 1$

Same No Change in it's Decimal Value

Let $x_3x_2x_1x_0$ data is in Signed 2's Complement Notation, it's 8 bit format

is

(5)

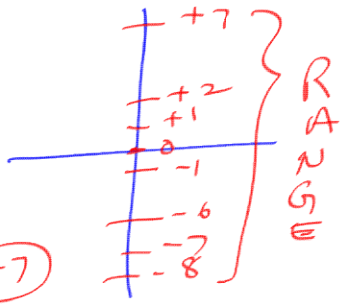
$x_3x_3x_3x_3x_3x_2x_1x_0$

a	$0000\ x_3x_2x_1x_0$
b	$1111\ x_3x_2x_1x_0$
c	$x_3x_3x_3x_3x_3x_2x_1x_0$
d	$\bar{x}_3\bar{x}_3\bar{x}_3\bar{x}_3x_3x_2x_1x_0$

4 bit — 2's complement Notation Range

S						
+ve	0	0	0	0	+0	−8 to −1 and
		to			to	+0 to +7
	0	1	1	1	+7	
−ve	1	0	0	0	−8	
		to			to	
	1	1	1	1	−1	
	−8	4	2	1		

(−8) to (+7)



4 bit Range −8 to +7

(1) + 3 = Value

(S)

$$-2^{3} \text{ to } +2^{3}-1$$

$$-2^{4-1} \text{ to } +2^{4-1}-1$$

For 'n' bit size, the Range is

$$-2^{n-1} \text{ to } +2^{n-1}-1$$

For 8 bit, Range −128 to +127

Overflow (OVFL) in Signed 2's complement Notation.

→ Overflow is also considered as Signed Carry.

→ After adding 2 no of n bit data, the OVFL occurs when 'n' bits are not sufficient to store the Result (in Decimal)

→ OVFL doesnot occur after adding one positive data and one Negative data

→ Occurence of OVFL changes the Target sign bit in the Result.

Overflow Detection

Cin = It is the carry generated by adding data that enters in Sign bit position.

Cout = It is the carry generated by adding both sign bits.

→ presence of both Cin and Cout is needed to detect OVFL.

Cin Cout	XOR o/p
0 0 1 1	0
0 1 1 0	1

→ Cout participates in final Result; when OVFL has occured only.

Decimal (unsigned)

Ex1

$$\begin{array}{r} 9 \\ +\ 3 \\ \hline (1)(2) \end{array}$$

carry ↓ Sum

Decimal Signed

$$\begin{array}{r} +9 \\ +\ +3 \\ \hline (+1)(2) \end{array}$$

OVFL Sum

Let data size is 4bit; Range permitted -8 to $+7$

S -8 4 2 1 databits

Ex1

1 1 1 1 (−1)

$$\begin{array}{r} \text{Cin}\ 1\ 1\ 1\ 0 \quad (-2) \\ +\ (1)\ 1\ 1\ 1 \\ \hline \text{Cout}\ (1)\ 1\ 1\ 0\ 1 \quad (-3) \end{array}$$

1 → 1 → 0

No OVFL

Ex2

S data

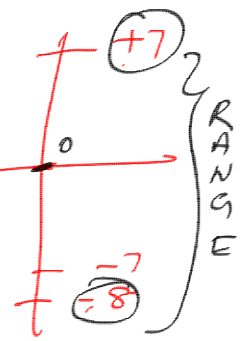
1 0 1 0 −6

-8 4 2 1

$$\begin{array}{r} 0\ 1\ 1\ 1 \quad +7 \\ +\ 0\ 0\ 0\ 1 \quad +1 \\ \hline (1)\ 0\ 0\ 0\ 1 \end{array}$$

Cout 1 → 1 → 0

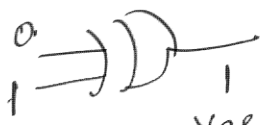
No OVFL



Ex3

1001 (-7)
-8 4 2 1
+ 1010 (-6)
Cin 0
1 0011 -13
Cout

Yes !



OVFL Yes
occured

Ex4

0111 +7
0101 +5
Cin 1 1 1
01100 +12
Cout

Yes

Yes
occured

Ex5

1111 (-1)
-8 4 2 1
Cin 1111 (-1)
1 1 1
Cout 1 1110 (-2)

No OVFL

NO OVFL

Cout participates in final Result when OVFL has occured only.

Ex1

0110 +6
+ 1001 -7
0 0
0 1111 (-1) ✓
Cout

NO OVFL

final Result

~~1111~~ = -1
-8 4 2 1

Ex2

1001 (-7)
+ 1001 (-7)
1 1
Cout 1 0010 -14
Yes.

final Result

10010 = -14
-16 8 4 2 1

Q) After adding 2 no. of signed 2's Complement Notation data, the overflow

MSQ

a) occurs, when $C_{in} = 1$ ✗ False

b) occurs, when $C_{out} = 1$ False

c) occurs, when $C_{in} \neq C_{out}$ Yes

d) Doesnot occur after adding one positive data and one Negative Data Yes

C and D

Subject

Computer organization & Architecture

Chapters (Topics)

I. Computer Arithmetic ✓

II. Memory Organization

III. Secondary Memories

IV. Basic processor organization and Design

V. Pipeline organization

VI. Control unit Design

VII. IO Organization

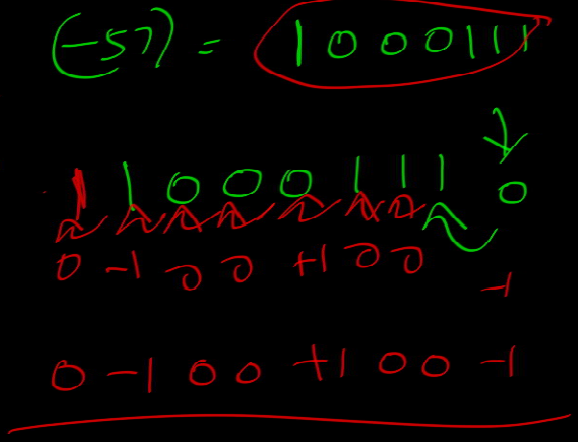
Q. Booth's coding in 8 bits for the decimal number – 57 is:

(a) 0 – 1 0 0 + 1 0 0 0

(b) 0 – 1 0 0 + 1 0 0 – 1

(c) 0 – 1 + 1 0 0 – 1 0 +1

(d) 0 0 – 1 0 + 1 0 0 – 1



Q. Sign extension is the step in

(a) Floating point multiplication

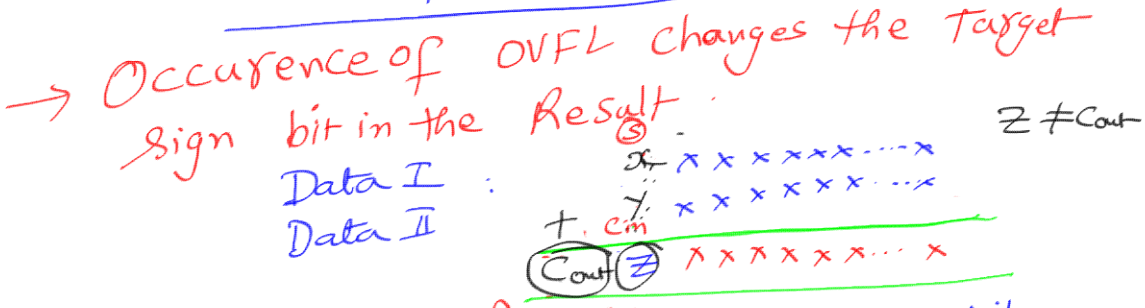
(b) Signed 16 bit integer addition

(c) Arithmetic left shift

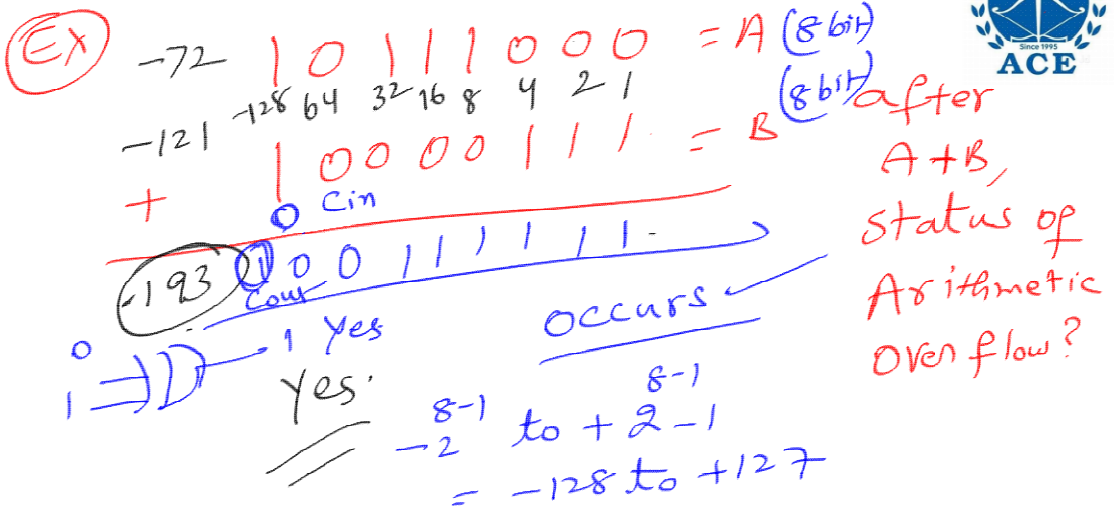
(d) Converting a signed integer from one size to another

Overflow in Signed 2's Complement Notation

→ Occurence of OVFL changes the Target sign bit in the Result.



x = Sign bit in first data.
y = Sign bit in 2nd data,
z = Sign bit in the Result



Q) Let x is the sign bit in first data, Y is the sign bit in 2nd data and Z is the sign bit in Result. The SOP Expression for occurence of overflow.

a) $x+y+z$ b) $xy+yz+xz$

c) $xy\bar{z}+\bar{x}\bar{y}z$ d) $\bar{x}\,\bar{y}\,\bar{z}$

xy	x y z	OVFL
+ +	0 0 0	0
+ +	0 0 1	1
+ −	0 1 0	0
+ −	0 1 1	0
− +	1 0 0	0
− +	1 0 1	0
− −	1 1 0	1
− −	1 1 1	0

SOP Expression

$\bar{x}\bar{y}z+xy\bar{z}$

Different Arithmetical operations performed by the C.P.U.

→ multiplication operation can be performed in 3 methods

(i) Repetitive Addition
(ii) Shift and Add technique.
(iii) Booth's multiplication.

→ For (i) and (ii), one operand is compulsory unsigned

→ while adding 2 no.of signed data both operands size is compulsory same

(Ex1)

-16 8 4 2 1

1 0 0 1 0 (5bit) -14

-4 2 1

1 0 1 3bit -3

-17

1 0 1 1 1

-16 8 4 2 1

Doesnot Satify (-9) ✗

1 0 0 1 0 I

+ 1 1 1 0 1 II

cout 1 0 1 1 1 1

-32 16 8 4 2 1

-32 + 15 = -17

Satisfies

VVVIMP.

→ After adding 2 no. of n bit data, the longest Result size is (n+1) bits

→ After multiplying 2 no. of n bit data, the longest Result size = 2n bits.

→ After multiplying one m bit size and another n bit size data; (m+n) bits

Repetive Addition

(-7) × (4)

-7 (1)
-7 (2)
-7 (3)
-7 (4)

= -28

Shift and Add technique.

Shift left is used to perform multiplication

In Decimal

$26 \times 1000_{10}$

$= 26 \times 10^3$

26000

Binary (signed)

$(-7) \times 8 = -56$

$-7 = \overset{-8}{1}001 \times 8 = 1001 \times \frac{3}{2}$

1001000 = -64 + 8

-64 32 16 8 4 2 1 = -56

Ex $(-7) \times 9 = -63$

Signed unsigned $= (-7) \times \left(\frac{3}{2}+1\right)$

$-7 = \overset{-8}{1}001, \quad 1001 \times \left(\frac{3}{2}+1\right)$

$= \left(1001 \times \frac{3}{2}\right) + 1001 =$ 1001000

Shift

Adding + 1111001

11000001

-128 64 32 16 8 4 2 1

$= -128 + 65 = -63$

Booth's multiplication.

→ It permits only signed operands.

→ It provides Excellent performance when the multiplier is having string of zeros or string of 1's and it provides worst performance when the multiplier is having more no. of bit changes.

→ In this technique the Booth's code is generated for multiplier only.

→ For generating Booth's code, the multiplier is scanned from LSB and always one special bit is taken for LSB scanning (0)

→ while scanning moves from 0 to 0 or 1 to 1 no arithmetical operation is required, just the partial Result is shifted.

→ while scanning moves from 0 to 1, the Shifted multiplicand is added for (-1) times and while scanning moves from (1 to 0), the shifted multiplicand is added for +1 times.

M = multiplicand
Q = multiplier

Standard Result
size = 2n bits

EX1 $(-7) \times (+4) = -28$

M Q

$M = -7 = 1001$

$(-1) * M = +7 = 0111$

$M = 1001, \ \overline{M} = 0110$

$\overline{M}+1 = 0111 = (+7)$

$Q = +4$
$= 0100$

0 1 0 0 0

+1 (-1) 0 0

Booth's Code for +4

+1 -1 0 0

8 4 2 1

$= 8-4 = 4$

$(-7) * (+4) = -28$

$-7 = 1001$ (4 bit)

Booth's code for +4
$= +1\ -1\ 0\ 0$

Standard Result
size is 8 bit

$M = -7 = 1001$

$(-1)M = +7 = 0111$

1001 - (M)

* +1 (-1) 0 0

0 0 0 0 0 0 0 0 | 0

0 0 0 0 0 0 0 0 - | 0

0 0 0 1 1 1 - - | (-1)

\+ 1 1 1 0 0 1 - - - | +1

1 1 1 0 0 1 0 0

128 64 32 16 8 4 2 1

S S

$= -128 + 100$
$= -28$

Result = +1 0 0 1 0 0

-32 16 8 4 2 1

$= -34 + 4 = -28$

(Ex) Generate the Booth's code for $(-37)_{10}$, $(+65)$ multipliers.

$-64+27$

$-37 = 1\,0\,1\,1\,0\,1\,1$ (−64 16 8 2 1)

1 0 1 1 0 1 1 0

−1 +1 0 −1 +1 0 −1

−1 +1 0 −1 +1 0 −1

−64 32 8 4 1

−73
36
−37

$+65 = 64+1$

0 1 0 0 0 0 0 1 0

+1 −1 0 0 0 0 +1 −1

+1 −1 0 0 0 0 +1 −1

128 64 2

$+130-65=+65$

Find the Booth's code for

(i) 1 1 1 1 1 1 1 0 (ii) 1 0 1 0 1 0 1 0

Best performance

1 1 1 1 1 1 1 0 0

0 0 0 0 0 0 −1 0

one operation (1)

5th

Worst performance

1 0 1 0 1 0 1 0

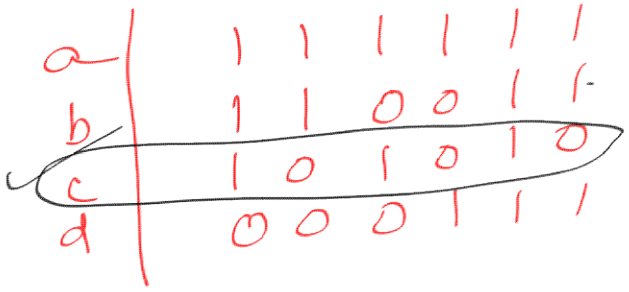
−1 +1 −1 +1 −1 +1 −1 0

−1 +1 −1 +1 −1 +1 −1 0

सात (7)

Q) In the given below Booth's multipliers, which one gives worst performance?

1999 GATE



more no of operations

ISRO.

Floating point Representation:

It provides more range than Integer Representation for the given size of the data.

10 bit Register

x x x x x x x x x x

signed Integer

Range provided

-2^{10-1} to $+2^{10-1}-1$

-512 to $+511$

Fixed point Notation	Floating point Notation
625.73_{10} (REAL Number)	$= 625.73 \times 10^{0}$
INT, Mantissa Fraction Significand	10 = base = r 0 = Exponent
11001.0101_2	$= 11001.0101 \times 2^{0}$

Floating point Representation requires Mantissa (M), Base (Binary Base 2), Exponent, Sign of the Mantissa and Sign of the Exponent

$S \pm e$

$S \pm 0.M \times 2$

M = Mantissa

e = exponent

VVVImp → only Sign of the Mantissa gives the sign of the data but not Exponent sign

Ex1 $625_{10} = 625.0 = 625.0 \times 10^{0}$

$= +0.\underbrace{625}_{m} \times 10^{+3}_{r}$

Ex2 $0.00528 = 0.00528 \times 10^{0}$

$= 0.528 \times 10^{-2}$

Ex $+0.\underbrace{110101}_{M} \times 2^{-4}$ (-) $=$

$= +0.000001101 \times 2^{1}$ +

Sign of the Exponent gives the Relative position of the decimal point in the fractional Mantissa

Mantissa range: M_{range} — n bit

unsigned.

$M_{min} = 0.0000\cdots0 = 0$

$M_{max} = 0.\underset{2^{-1}}{1}\underset{2^{-2}}{1}\underset{2^{-3}}{1}\underset{2^{-4}}{1}\cdots\underset{2^{-n}}{1} = \frac{15}{16} = 1-\frac{1}{16} = 1-2^{-n}$

(MSB: 2^{-1}, LSB: 2^{-n})

In mantissa, when all bits are '1' then its Value $\simeq 1$, accurate Value $= 1 - LSB = 1 - 2^{-n}$

Expression Value for Floating Data Representation

$(\pm)\, 0.M \times 2^{\pm e}$

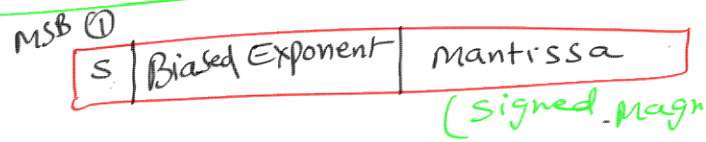
$(-1)^{S}\, 0.M \times 2^{\pm e}$

S = Sign of the mantissa

S = 0 +ve

S = 1 -ve

Representation of Floating point data in a Register



S = Sign of the 'M' 0 for +ve, '1' for -ve.

Biasing: It is the process of converting signed Exponent to unsigned Exponent by adding some fixed value to the TRUE Exponent.

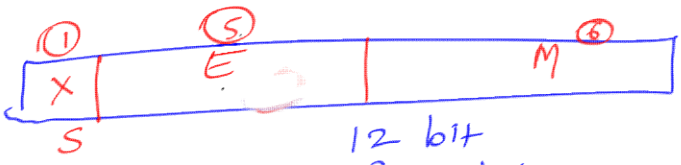
E = Biased Exponent (unsigned)

b = biasing value

e = TRUE Exponent (original Exponent)

$$E = e + b$$

→ Always Mantissa is expressed in signed Magnitude Notation only



e = Signed

E field size = 5 bit

unsigned minimum

$E_{min} = 00000_2 = 0$

$E_{max} = 11111_2 = 31$

with 5 bit, Range of TRUE exponent (e)

e_{range} -2^{5-1} to $+2^{5-1}-1 = -16$ to $+15$

E field size = 5 bit, with 5 bits,

(00000) $0 \le E \le 31$ (11111)

$E_{min} = 0$

$E_{max} = 31 = (2^5 - 1)$

$E = e + b$

$E_{min} = e_{min} + b$

$0 = -16 + b$

$b = 16$

with 5 bit e_{range}

$+15 \le e \le -16$ (-16 to +15)

$e_{min} = -16$

$e_{max} = +15$

$E_{max} = e_{max} + b$

$31 = 15 + b$; $b = 16$

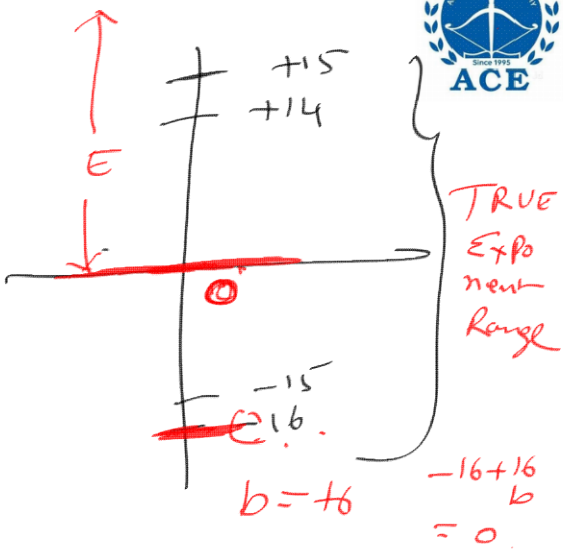
With 5 bit; e range

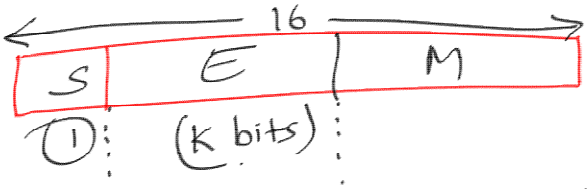
-2^{5-1} to $+2^{5-1}-1$

-16 to $+15$

$e_{min} = -16$

$E_{min} = 0$





With k bit; E range, $E_{min} = 0$; $E_{max} = 2^{k}-1$

e range; -2^{k-1} to $+2^{k-1}-1$ $\quad e_{min} = -2^{k-1}$

$E_{min} = e_{min} + b$ $\quad E_{min} = 0$

$0 = -2^{k-1} + b$, $b = 2^{k-1}$

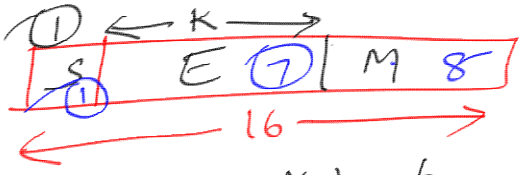
$E_{max} = e_{max} + b$

Some times Biasing Value is also Known as Excess Value.

(Ex) A 16 bit Register is aimed to store Floating data with Excess 64 technique.

1) No. of bits needed for 'M' field is 8

2) No. of bits needed for 'E' field is 7.



$b = 64 = 2^{K-1} = 2^{6}$

$K-1 = 6$

$K = 7$

(Ex) A 12 bit Register is aimed to store a floating data with Excess 16 technique.

1) No. of bits needed for 'M' field is 6

2) " " for 'E' field is 5

3) Store -12.75_{10} in the above Register and Express the Value in Hexadecimal and Octal Notations.

K=5

$b = 16 = 2^{K-1} = 2^4$

-12.75

$12.75 = 1100.11 \times 2^0$

$= 0.110011 \times 2^4$

S = 1,

e = 4

b = 16

E = 20

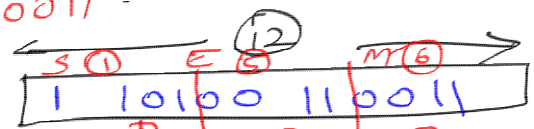
= 10100

M = 110011

.75 × 2

1.5 × 2

1.0



$(D33)_{16}$

$1\ 10|100|110|011 = (\quad)_8$

$6 \quad 4 \quad 6 \quad 3$

$(6463)_8$