RADIX r NUMBER SYSTEM

A number in radix r number system is represented in terms of POSITIONAL WEIGHTING

$$(N)_r = d_{n-1}\, r^{n-1} + d_{n-2}\, r^{n-2} + \ldots + d_1\, r^1 + d_0\, r^0 \;\underset{\text{RADIX POINT}}{\bullet}\; d_{-1}\, r^{-1} + \ldots + d_{-m}\, r^{-m}$$

INTEGRAL PART FRACTIONAL PART

d_k = digit in position k, $-m \le k \le n-1$
r^k = weight of position k, $-m \le k \le n-1$
n = number of integral digits in N
m = number of fractional digits in N

(0, ..., r-1) = DIGITS IN RADIX r NUMBER SYSTEM

e.g.,

- r = 10 (DECIMAL number system)
  digits are (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
  $(34.25)_{10} = 3 \times 10^1 + 4 \times 10^0 \bullet 2 \times 10^{-1} + 5 \times 10^{-2}$

- r = 2 (BINARY number system)
  digits are (0, 1)
  $(100010.01)_2 = 1x2^5 + 0x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 0x2^0 \bullet 0x2^{-1} + 1x2^{-2}$

- r = 8 (OCTAL number system)
  digits are (0, 1, 2, 3, 4, 5, 6, 7)
  $(42.2)_8 = 4 \times 8^1 + 2 \times 8^0 \bullet 2 \times 8^{-1}$

- r = 16 (HEXADECIMAL number system)
  digits are (0, 1,.2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)
  $(22.4)_{16} = 2 \times 16^1 + 2 \times 16^0 \bullet 4 \times 16^{-1}$

d′ = COMPLEMENT OF A DIGIT d IN RADIX r NUMBER SYSTEM
= (r-1) - d

e.g., $0' = r-1-0 = r-1$

$1' = r-1-1 = r-2$

$(r-2)' = r-1-(r-2) = r-1-r+2 = 1$

REPRESENTATION OF HEXADECIMAL, DECIMAL, OCTAL DIGITS

$2^3 2^2 2^1 2^0$	r=16	r=10	r=8
0 0 0 0	0	0	0
0 0 0 1	1	1	1
0 0 1 0	2	2	2
0 0 1 1	3	3	3
0 1 0 0	4	4	4
0 1 0 1	5	5	5
0 1 1 0	6	6	6
0 1 1 1	7	7	7
1 0 0 0	8	8	
1 0 0 1	9	9	
1 0 1 0	A		
1 0 1 1	B		
1 1 0 0	C		
1 1 0 1	D		
1 1 1 0	E		
1 1 1 1	F		

n	2^n	n	2^n
0	1	8	256
1	2	9	512
2	4	10	1,024
3	8	11	2,048
4	16	12	4,096
5	32	13	8,192
6	64	14	16,384
7	128	15	32,768

I- BINARY ⟷ DECIMAL

A) BINARY TO DECIMAL

e.g., $(100010.01)_2$ $= 1x2^5 + 0x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 0x2^0 + 0x2^{-1} + 1x2^{-2}$
$= 1x32 + 0 + 0 + 0 + 1x2 + 0 + 0 + 1x1/4$
$= 32 + 2 + 1/4 = 34 + 0.25$
$= (34.25)_{10}$

B) DECIMAL TO BINARY

e.g., $(34.25)_{10} = (?)_2$

i) CONVERT INTEGRAL PART BY DIVIDING IT SUCCESSIVELY BY 2;

34	2	R
17		0
8		1
4		0
2		0
1		0
0		1

AND BY READING THE REMAINDERS UP, THE RESULT IS $(100010)_2$

ii) CONVERT FRACTIONAL PART BY MULTIPLYING IT SUCCESSIVELY BY 2

0.25 X 2

⟶ POINT

0.50 X 2

1.00

AND BY READING THE INTEGRAL DIGITS DOWN, THE RESULT IS $(0.01)_2$

iii) COMBINE THE RESULTS i) & ii)

$(34.25)_{10} = (100010.01)_2$

II- BINARY ⟷ OCTAL

A) BINARY TO OCTAL

FORM GROUPS OF 3 BITS STARTING AT BINARY POINT
EACH GROUP OF 3 BITS IS AN OCTAL DIGIT

e.g., $(100010.01)_2 \longrightarrow 100 \quad 010 \bullet 010$

$$= \overset{4}{} \quad \overset{2}{} \quad \overset{2}{} \quad (42.2)_8$$

B) OCTAL TO BINARY.

CONVERT EACH OCTAL DIGIT TO ITS EQUIVALENT IN BINARY

e.g., $(27.5)_8 \longrightarrow 010 \quad 111 \bullet 101$

$$= (10111.101)_2$$

III- BINARY ⟷ HEXADECIMAL

A) BINARY TO HEXADECIMAL

FORM GROUPS OF 4 BITS STARTING AT BINARY POINT
EACH GROUP OF 4 BITS IS AN HEXADECIMAL DIGIT

e.g., $(100010.01)_2 \longrightarrow 0010 \quad 0010 \bullet 0100$

$$= \overset{2}{} \quad \overset{2}{} \quad \overset{4}{} \quad (22.4)_{16}$$

B) HEXADECIMAL TO BINARY

CONVERT EACH HEXADECIMAL DIGIT TO ITS EQUIVALENT IN BINARY

e.g., $(5D.A)_{16} \longrightarrow 0101 \quad 1101 \bullet 1010$

$$= (1011101.101)_2$$

REPRESENTATION OF NUMERIC INFO IN BINARY FORM

⇒ BINARY NUMBERS

A) FIXED POINT REPRESENTATION

±N has an IMPLICIT binary point in a FIXED POSITION

a) if ±N is an integer, binary point is at the right of rightmost digit
$d_{n-1}d_{n-2}d_{n-3} \cdots d_2d_1d_0$

b) if ±N is a fraction, binary point is at the left of leftmost digit
$d_{-1}d_{-2}d_{-3} \cdots d_{-m+2}d_{-m+1}d_{-m}$

c) if ±N is mixed, binary point is between two predetermined digits
$d_{n-1}d_{n-2}d_{n-3} \cdots d_2d_1d_0d_{-1}d_{-2}d_{-3} \cdots d_{-m+2}d_{-m+1}d_{-m}$

Let us consider the case where

±N is an **INTEGER** represented in an **n**-bit word $d_{n-1}d_{n-2}d_{n-3} \cdots d_2d_1d_0$

d_{n-1}	d_{n-2}	d_{n-3}	...	d_2	d_1	d_0

A1) UNSIGNED BINARY INTEGERS (UBI)

N is represented in terms of positional weighting : $0 \le N \le 2^n - 1$

A2) SIGNED BINARY INTEGERS

±N is represented in one of three forms given below where

d_{n-1} represents SIGN (0 is positive and 1 is negative)

$d_{n-2}d_{n-3} \cdots d_2d_1d_0$ represents |N|

$? \le |N| \le ?$

A2.1) SIGN - MAGNITUDE FORM (SMF)

$0 \le |N| \le 2^{n-1} - 1$

A2.2) 1's COMPLEMENT FORM (1's CF)

Note : To find 1's COMPLEMENT of a sequence of bits replace each bit d in the sequence by d′ = 1-d

A2.3) 2's COMPLEMENT FORM (2's CF)

Note : To find 2's COMPLEMENT of a sequence of bits first find 1's complement of the sequence then add $(1)_2$ (exceptions: 0 and -2^{n-1})

COMPARISON of SMF, 1's CF, and 2's CF
ASSUME 4 - BIT WORD LENGTH (i.e., n=4)

4-bit word	UBI	SMF	1's CF	2's CF
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

Observe the following from the above table:

A. The leftmost bit is the sign bit such that
if it is 0, the number is positive, and if it is 1, the number is negative.

B. In both SMF and 1's CF, there are
seven positive integers, seven negative integers, and
two 0's, (-0 and +0), making a total of 16 unique numbers.
In 2's CF, there are
seven positive integers, eight negative integers, and
one 0, making a total of 16 unique numbers.

C. Assume n-bit word length.
In both SMF and 1's CF, there are
$2^{n-1}-1$ positive integers,
$2^{n-1}-1$ negative integers, and
2 0's, for a total of 2^n unique numbers.
In 2's CF, there are
$2^{n-1}-1$ positive integers,
2^{n-1} negative integers, and
1 0, for a total of 2^n unique numbers.

FIXED POINT ARITHMETIC IN 2's CF

- ## ADDITION

sum = augend + addend

where augend and addend are in 2's CF

⇒

- NO SPECIAL TREATMENT OF THE SIGN BIT
- DISCARD THE CARRY-OUT

e.g., ASSUME 4 - BIT WORD LENGTH

```
0101        (+5)
0010      + (+2)
0111        (+7)

0101        (+5)
1110      + (- 2)
0011        (+3)        Discard carry out

1011        (- 5)
0010      + (+2)
1101        (- 3)

1011        (- 5)
1110      + (- 2)
1001        (- 7)       Discard carry out
```

BOTH OPERANDS AND THE RESULT ARE IN 2's CF

OVERFLOW

Occurs (only when the sign bits of both operands are the same)
If the sign bit of the result is not the same as the sign bits of the operands

---

Algorithm for S = X + Y

Given X and Y which are both signed integers represented in n-bit words in 2's CF
S is formed by performing X + Y which requires ignoring carry-out (if any)
S is correct if overflow does not occur, i.e.,

$$-2^{n-1} \le X + Y \le 2^{n-1} - 1$$

```
 0001        0001        1111        1111
+0110       +0111       +1001       +1000
------      ------      ------      ------
 0111        1000       r1000       r0111
                ↑                       ↑
            overflow                overflow
```

- **SUBTRACTION**

difference = minuend - subtrahend

where minuend and subtrahend are in 2's CF

SUBTRACTION IS PERFORMED BY ADDITION

⇒

difference = minuend + (- subtrahend)

where (- subtrahend) is 2's complement of subtrahend

e.g., ASSUME 4 - BIT WORD LENGTH

0101	(+5)		(+5)	0101
0010	- (+2)	⇒	+ (- 2)	1110
			(+3)	0011
0101	(+5)		(+5)	0101
1110	- (- 2)	⇒	+ (+2)	0010
			(+7)	0111
1011	(- 5)		(- 5)	1011
0010	- (+2)	⇒	+ (- 2)	1110
			(- 7)	1001
1011	(- 5)		(- 5)	1011
1110	- (- 2)	⇒	+ (+2)	0010
			(- 3)	1101

BOTH OPERANDS AND THE RESULT ARE IN 2's CF

Algorithm for D = X - Y (subtraction in 2's CF)

Given X and Y which are both signed integers represented in n-bit words in 2's CF

D is formed by

- taking 2's complement of Y
- performing X + (2's complement of Y) which requires ignoring carry-out (if any)

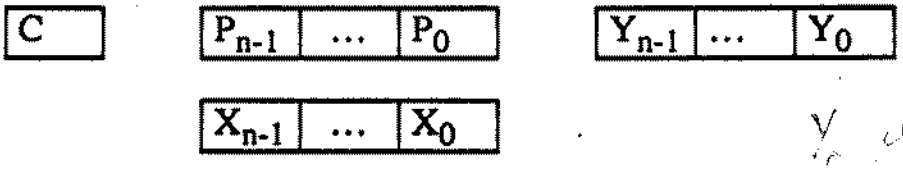
D is correct if overflow does not occur, i.e.,

$$-2^{n-1} \le X + (\text{2's CF of } Y) \le 2^{n-1} - 1$$

addition in 2's CF

- **MULTIPLICATION**
  product = multiplicand * multiplier

Recall the multiplication algorithm for <u>unsigned integers</u>:



Y determines action.

e.g., ? = X * Y, where X = 7 (i.e., 0111) and Y = 13 (i.e., 1101)

n = 4

C	P	Y	
0	0000	1101	initial
	t x		
0	0111	1101	after P = P + X
0	0011	1110	after 1-bit shift right
0	0001	1111	after 1-bit shift right
	t x		
0	1000	1111	after P = P + X
0	0100	0111	after 1-bit shift right
	t x		
0	1011	0111	after P = P + X
0	0101	1011	after 1-bit shift right

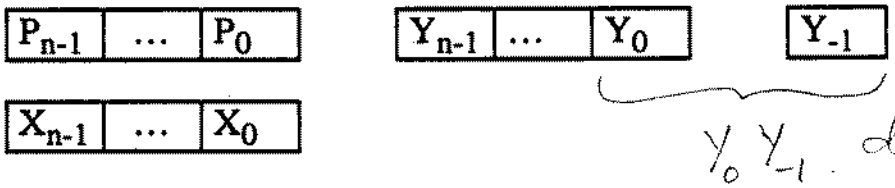
fY = X * Y = 7 * 13 = 91

Algorithm for X * Y where X and Y are unsigned integers
Given X and Y which are both represented in n-bit words
C← 0, P← 0, i ← n
do while i≠ 0
    if Y_0 = 1 then P ← P + X fi
    shift C, P, and Y together to right
    i ← i -1
od
The product is the combined contents of P and Y together

The above algorithm cannot be used if X and Y are signed integers in 2's CF.
If X and Y above were considered as in 2's CF,
X = +7 and Y = -3 and the product must have been -21 in 2's CF i.e.,

11101011 but it won't.

Booth's Algorithm for X * Y where X and Y are signed integers
i.e., multiplicand and multiplier are in 2's CF



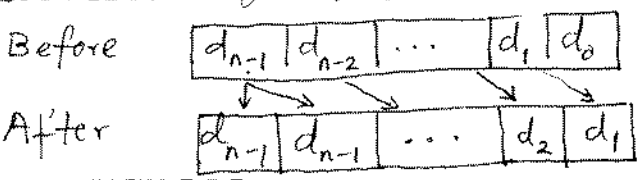
e.g., ? = X * Y, where X = 7 (i.e., 0111) and Y = - 3 (i.e., 1101)

P	Y	Y_{-1}	
0000	1101	0	initial
+ (-X)			
1001	1101	0	after P = P - X
1100	1110	1	after 1-bit arithmetic shift right
+ X			after P = P + X
0011	1110	1	after 1-bit arithmetic shift right
0001	1111	0	
+ (-X)			after P = P - X
1010	1111	0	after 1-bit arithmetic shift right
1101	0111	1	
1110	1011	1	after 1-bit arithmetic shift right

≡ -21 in 2's CF

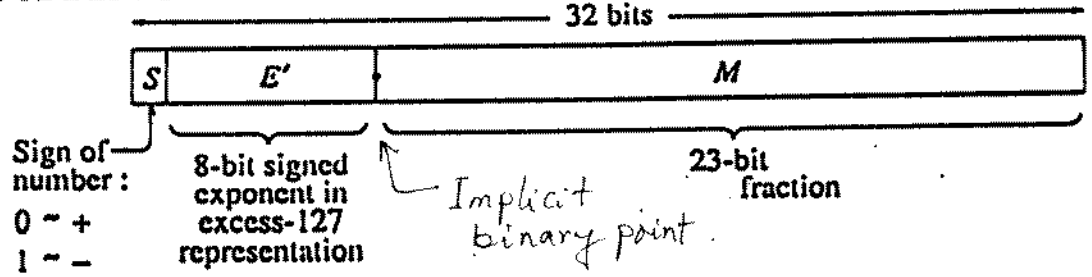
Booth's Algorithm for X * Y where X and Y are signed integers in 2's CF
Given X and Y which are both represented in n-bit words
P← 0, Y_{-1}← 0, i ← n
do while i≠ 0
    if Y_0 = 1 and Y_{-1}= 0 then P ← P - X fi
    if Y_0 = 0 and Y_{-1}= 1 then P ← P + X fi
    arithmetic shift P, Y and Y_{-1} together to right
    i ← i -1
od
The product is the combined contents of P and Y together

Arithmetic shift right



- **DIVISION**
  remainder and quotient ← dividend / divisor
  Recall the division algorithm for unsigned integers: SEEN IN CSI 1101

B) FLOATING POINT REPRESENTATION (IEEE 754)



where S is Sign Bit (0 for + or 1 for -)
$0 \le E' \le 2^8 - 1$
$0 \le M \le 2^{23} - 1$

E' = 0 and E' = 255 are used for four special values:

```
S  00000000  00000000000000000000000   ==> S zero
S  00000000  ddddddddddddddddddddddd   ==> S 2^-126
S  11111111  00000000000000000000000   ==> S Infinity
S  11111111  ddddddddddddddddddddddd   ==> NAN (not a number)
```

where d is 0 or 1

With IEEE 754, a number is represented as SIGN SIGNIFICAND x 2 SIGNED EXPONENT

SIGNIFICAND IS NORMALIZED!
WHY?
to obtain a unique representation for each number
HOW?
by placing the binary bit to the right of the first non-zero bit
Therefore, significand is represented by 1.M

SIGNED EXPONENT IS BIASED!
WHY?
to simplify comparison of exponents
HOW?
by adding $2^{8-1} - 1$ (= 127) to the actual exponent
i.e.,

8 bits	E'	Actual Exponent
00000000	0	-127
00000001	1	-126
01111110	126	-1
01111111	127	0
10000000	128	+1
11111110	254	+127
11111111	255	+128

The number represented in IEEE 754 32-bit format is
S 1.M x $2^{(E' - 127)}$
(provided that 0 < E' < 255 i.e., -126 ≤ E' - 127 ≤ +127)

S E' M
e.g., 1 10000001 11100000000000000000000

S = −

E' = 129 ⇒ $E' - 127 = 2$

M = 111

⇒ $-(1.111)_2 \times 2^2 = -(111.1)_2 \times 2^0 = -(7.5)_{10}$

e.g., + $(0.75)_{10}$

⇒ $+(0.11)_2 \times 2^0 = +(1.1)_2 \times 2^{-1}$

S = +

E' - 127 = −1 ⇒ $E' = 126$

M = 1

0 01111110 10000000000000000000000

e.g., + $(75)_{10}$

⇒ $+(1001011)_2 \times 2^0 = +(1.001011)_2 \times 2^6$

S = +

E' - 127 = 6 ⇒ $E' = 133$

M = 001011

0 10000101 00101100000000000000000