215. fixed point
1010. number

Representation of floating point numbers

Floating point Number : Contains two parts.

(i) **Mantissa:** Contains signed fixed point number

(ii) **Exponent:** specify the position of decimal (binary) point.

$\pm m \times r^{\pm e}$ → Exponent

↓ Sign ↓ mantissa ↓ radix (Base)

Examples :

$+215.37 = +0.21537 \times 10^{+3}$ → exponent

sign, mantissa, Base (radix)

$+1000.110 = +0.1000110 \times 2^{+4}$ → exponent

sign, mantissa, Base (radix)

$-1010.11 = -0.101011 \times 2^{+6}$ → exponent

sign, mantissa, base

These numbers are called floating point numbers because the position of the decimal (binary) point is fluctuating (floating).

Normalization :

A floating point number is said to be normalized, when we force the integer part of its mantissa to be 1, and allow its fractional part to anything

ex: $13.25 = 1101.01$

$1.10101 \times 2^{+3}$ (Normalized)

integer part, fractional part

exponent

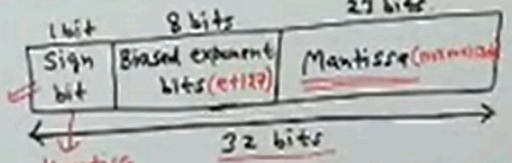
Normalized floating point number $= 1.ffffff \times 2$

↓ not stored in representation

IEEE 754 floating point Representation

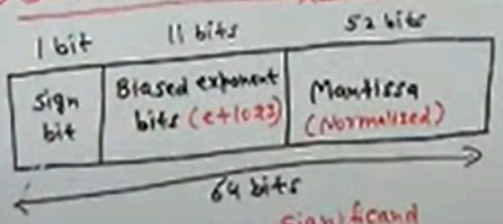
mantissa positive → 0
" negative → 1

① Single Precision floating point standard



mantissa

② Double Precision floating point standard



Mantissa = Significand
↓ ↓
obsolete word — modern word

Example: Represent 85.125 in IEEE 754 single precision floating point representation.

Ans: 85.125

1010101.001

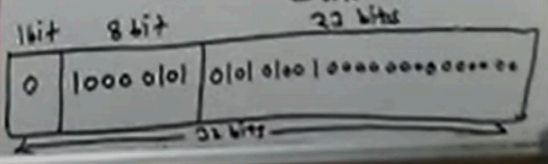
So $85.125 = 1010101.001 = 1.010101001 \times 2^6$

Number is +ve, so sign bit = 0

Exponent (e) = 6

Biased exponent (single precision) $= 6 + (2^{8-1} - 1)$
$= 6 + 127 = 133 = 10000101$

Mantissa (Normalized) = 01010010 (make of 23 bits)
$= 01010010\,0000000000000000$
23 bits



Overflow / Underflow in floating point numbers

$\pm m \times r^{\pm e}$ → base
↓ significand

① Significand overflow:

When two similar sign significand (mantissa) are added, the sum may result in a carry out of the MSB. This is called significand overflow.

ex: $1.101 \times 2^{+3}$
$+\ 1.110 \times 2^{+3}$
$11.011 \times 2^{+3}$ → Corrected by shifting significand one position right
$\Rightarrow 1.1011 \times 2^{+4}$

min. range max
underflow overflow

② Significand underflow:

If the resulting value has 0 in the most significand position of the significand. This is called significand underflow.

ex: $1.101 \times 2^{+5}$
$-\ 1.100 \times 2^{+5}$
$0.001 \times 2^{+5}$
$\Rightarrow 1.0 \times 2^{+2}$ → Corrected by shifting significand to the left.

③ **Exponent Overflow:**

An exponent overflow occurs, when a resulting positive exponent exceeds the maximum possible exponent value

④ **Exponent Underflow:**

A exponent Underflow occurs, when a resulting negative exponent is less than the minimum possible value.

Floating point Addition/Subtraction

Algorithm for floating point addition/subtraction

① **Check for zeros.** If either no. is zero, result will be other numbers with appropriate sign.

② **Align the Mantissas.** If exponent are equal, perform the arithmatic operation; else shift the mantissa with smaller exponent to the right until its exponent equal to the larger exponent.

③ Perform the addition/subtraction depending on the operation and sign of the two mantissas.

Let $X = X_A * 2^{+E_A}$, $Y = X_B * 2^{+E_B}$ } $Z = ?$

① **During addition:** If $Y=0$, $Z = X = X_A * 2^{+E_A}$
If $X=0$, $Z = Y = X_B * 2^{+E_B}$

During Subtraction If $Y=0$, $Z = X = X_A * 2^{+E_A}$
If $X=0$, $Z = -Y = -X_B * 2^{+E_B}$

② Let $X = 1.10 * 2^{+3}$ (Larger exponent)
$Y = 1.01 * 2^{+1}$ (Smaller exponent)
↓ mantissa

Shift the mantissa of y to the right, until its exponent is equal to the exponent of X.

$Y = 0.101 * 2^{+2}$ (one time shift)
$Y = 0.0101 * 2^{+3}$ (two time shift)

Now $X = 1.10 * 2^{+3}$
$Y = 0.0101 * 2^{+3}$

go to step-3

* During addition, if two significand are equal with opposite sign, the result will be zero.

* There is also a possibility of significand (mantissa) overflow by 1 digit, then the mantissa of the result is shifted right & exponent is incremented.

Significand overflow occurs, when the addition of two significand (mantissa) of same sign may result in a carry at MSB of mantissa.

as we increment exponent, there is also a possibility of exponent overflow. Then result will show "error."

(4) Normalize the result: If result is not normalized, then we shift mantissa left and decrement the exponent until value "Left of binary point is 1." As we are decreasing the exponent, so exponent can also be underflow. then again error is reported.

$X = 1.01 * 2^{+3}$

$Y = -1.01 * 2^{+3}$

then $X+Y = 1.01 * 2^{+3} - 1.01 * 2^{+3}$

$Z = X+Y = 0 * 2^{+3} = 0$

If $X = 1.01 * 2^{+3}$

$Y = 1.10 * 2^{+3}$

$Z = X+Y = 10.11 * 2^{+3}$

Carry at MSB

Correction = $1.011 * 2^{+4}$

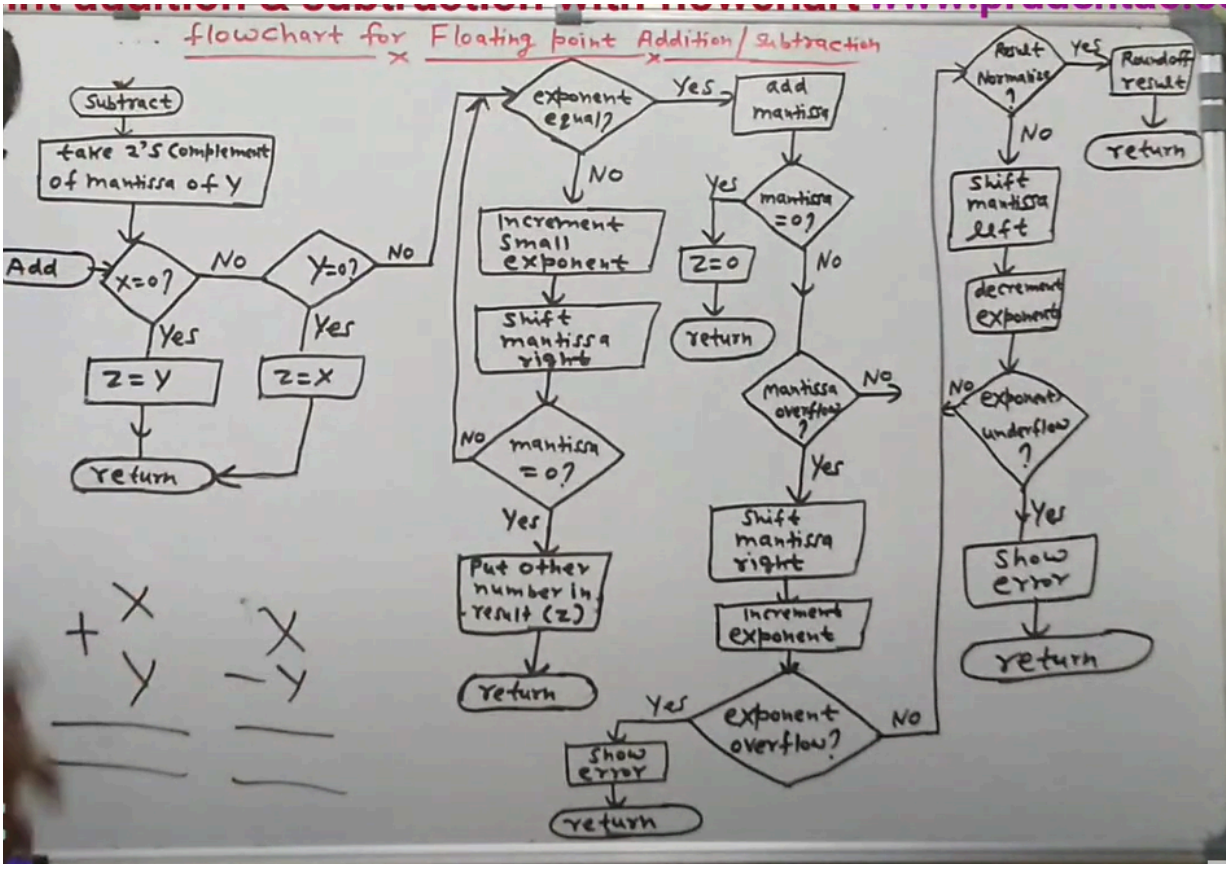
If $X = 1.01 * 2^{+3}$

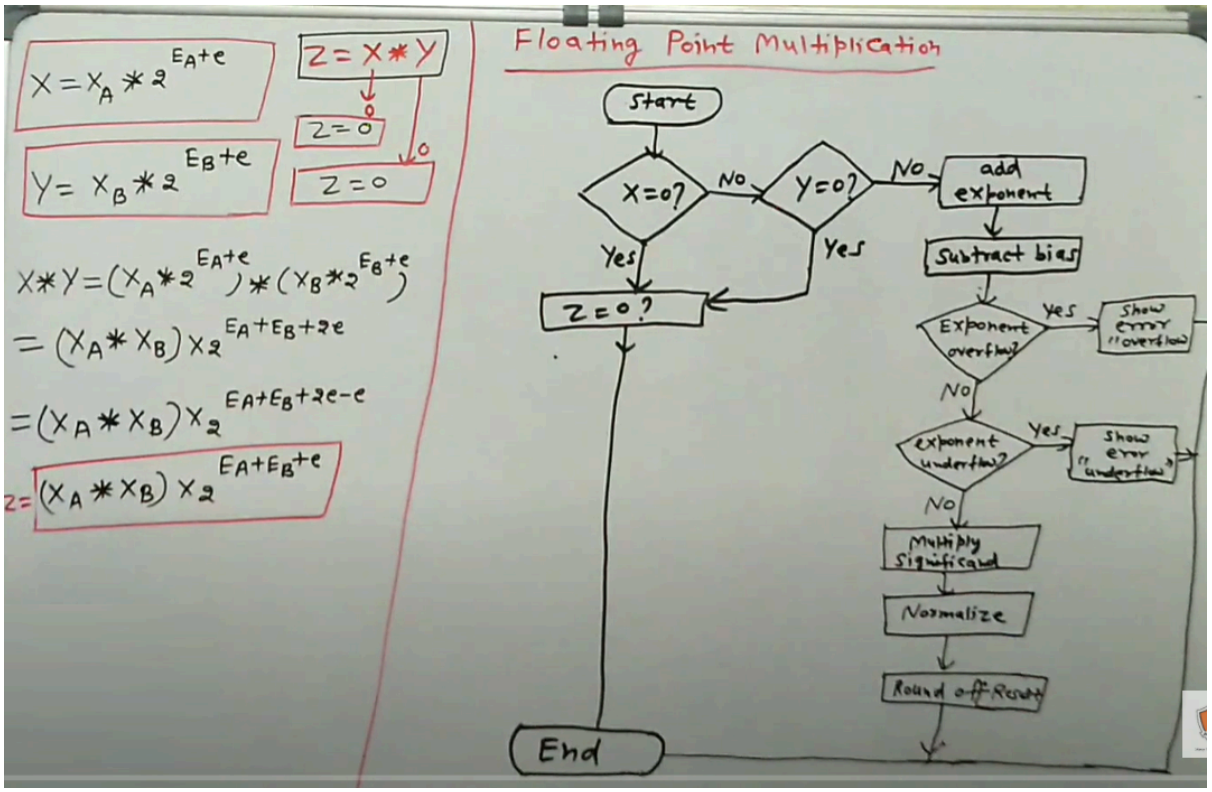
$Y = -1.00 * 2^{+3}$

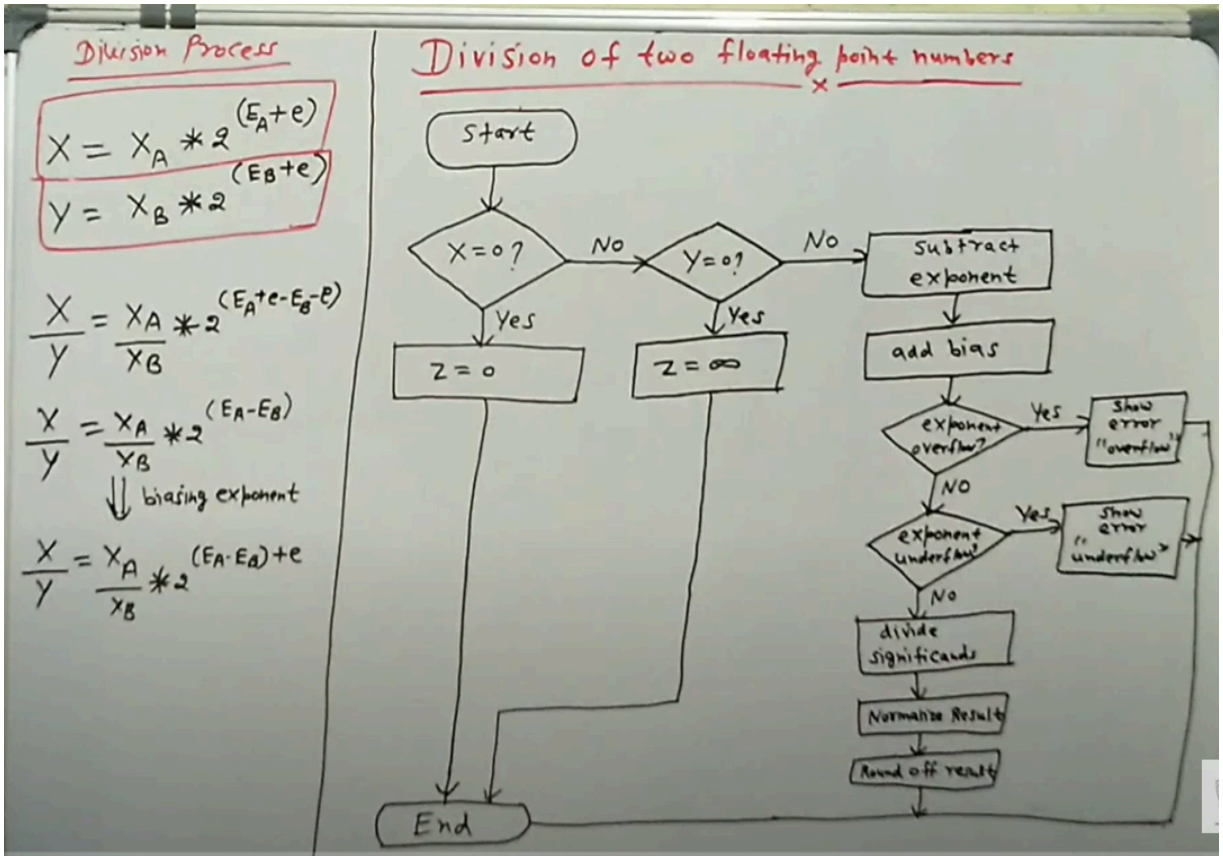
$Z = X+Y = 0.01 * 2^{+3}$ (Not normalized)

$= 1.0 * 2^{+1}$ (Normalized)

Next "flow chart for floating point add/sub"




$X = X_A * 2^{E_A+e}$
$Y = X_B * 2^{E_B+e}$
$X*Y = (X_A*2^{E_A+e}) * (X_B*2^{E_B+e})$
$= (X_A * X_B) X 2^{E_A+E_B+2e}$
$= (X_A * X_B) X 2^{E_A+E_B+2e-e}$
$Z = (X_A * X_B) X 2^{E_A+E_B+e}$
Z = X * Y
Z = 0
Z = 0
Floating Point Multiplication
Start
X=0?
No
Y=0?
No
add exponent
Yes
Yes
Subtract bias
Z=0?
Exponent overflow?
Yes
Show error "overflow"
No
exponent underflow?
Yes
Show error "underflow"
No
Multiply Significand
Normalize
Round off Result
End


Division Process
$X = X_A * 2^{(E_A+e)}$
$Y = X_B * 2^{(E_B+e)}$
$\frac{X}{Y} = \frac{X_A}{X_B} * 2^{(E_A+e-E_B-e)}$
$\frac{X}{Y} = \frac{X_A}{X_B} * 2^{(E_A-E_B)}$
biasing exponent
$\frac{X}{Y} = \frac{X_A}{X_B} * 2^{(E_A-E_B)+e}$
Division of two floating point numbers
Start
X=0?
No
Y=0?
No
Subtract exponent
Yes
Yes
add bias
Z = 0
Z = ∞
exponent overflow?
Yes
Show error "overflow"
No
exponent underflow?
Yes
Show error "underflow"
No
divide significands
Normalize Result
Round off result
End