COMPUTER SCIENCE

Computer Organization and Architecture

Floating Point Representation

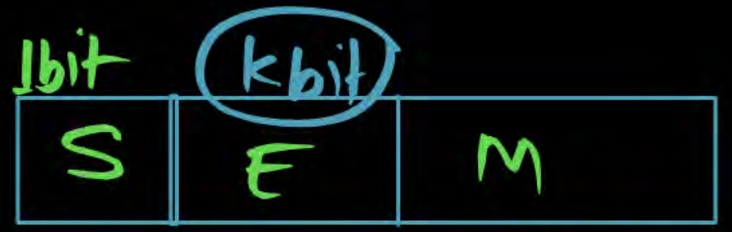
Lecture_03

Vijay Agarwal sir

01 Floating Point Representation

02 IEEE 754 Floating Point Representation

Floating Point Representation.



M: Mantissa.

S: Sign bit
- 0 +ve
- 1 -ve.

$$bias = 2^{k-1}$$

BE / E : Exponent / Biased Exponent

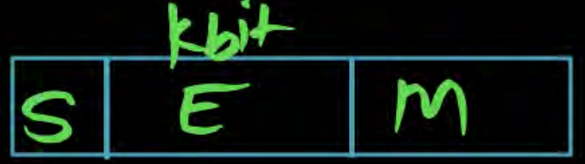
$$E = e + bias$$

$$bias = 2^{\text{Exponent bit} - 1}$$

$E = e + bias$

$e = E - bias$

Mantissa



$E = e + bias$

$bias = 2^{k-1}$

Implicit Normalized

$1 \cdot Something \times 2^{e}$

$(-1)^{S} \; 1 \cdot M \times 2^{e}$

$(-1)^{S} \; 1 \cdot M \times 2^{E - bias}$

Explicit Normalized

$0 \cdot 1xxx \times 2^{e}$

$(-1)^{S} \; 0 \cdot M \times 2^{e}$

$(-1)^{S} \; 0 \cdot M \times 2^{E - bias}$

Normalized Mantissa

Explicit Normalized

Syntax

$0.1\underline{\text{.....}} \times 2^{e}$
M

Formula to get number

[value formula]

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

$(-1)^{S} \times 0.M \times 2^{E\text{-bias}}$

Implicit Normalized

Syntax

$1.\underline{\text{.....}} \times 2^{e}$
M

Formula to get number

[value formula]

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

$(-1)^{S} \times 1.M \times 2^{E\text{-bias}}$

Explicit

0.1 After the point,

Immediate first bit should be 1

Example

(101.11)

0.10111×2^3

M = 10111,

e = 3

E = e + bias

Implicit

Before the point 1 means 1. .....

Example 1.Something.

(101.11)

1.0111×2^2

M = 0111,

e = 2

E = e + bias

Floating-Point Representation

S	E	M

S: sign bit
- 0 +ve
- 1 –ve

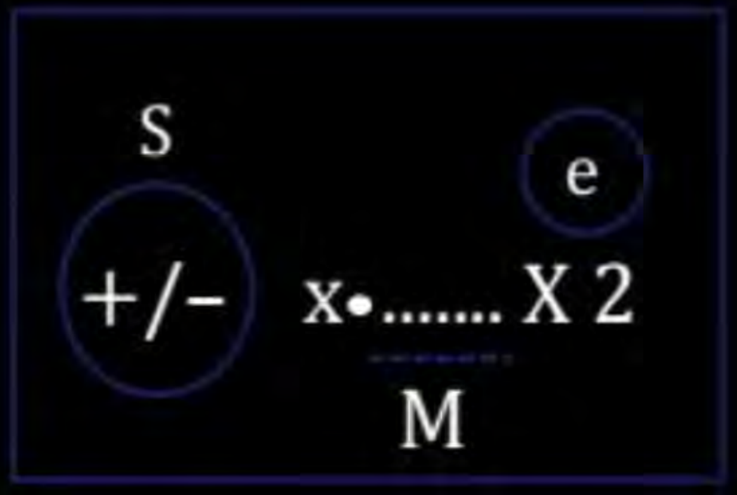
E:Biased exponent

M: Mantissa

$E = e + bias$

or

$BE = AE + bias$



$. ..... X2^{e}$

Q. +21.75

Implicit ?

1 bit	7 bit	8 bit
S	E	M

10101.11

1.010111×2^4

$M = 010111$

$e = 4, \text{bias} = 2^{7-1}$

$E = 4 + 64$

$E = 68 = (1000100)_2$

Value Formula:

$(-1)^S \times 1.M \times 2^e$

$(-1)^0 \times 1.010111 \times 2^{68-64}$

1.010111×2^4

$10101.11 = (21.75)_{10}$

Ans

S(1bit)	E(7bit)	M(8 bit)
0	1000100	01011100

$\text{Hexadecimal} = (445C)_{16}$

Q.

Consider a 16 bit register used to store floating point number. Mantissa is Explicit normalized signed fraction number. Exponent is in Excess-32 form then what is 16-bit for $-(29.75)_{10}$ in the register?

Solution

1 bit	6 bit	9 bit
S	E	M

-29.75

-11101.11

-0.1110111×2^5

Implicit.

$1.1101111 \times 2^{+4}$

M: 1110111

$e = 5$

$bias = 2^{6-1}$

$bias = 32$

$E = 5 + 32 = 37 = (100101)_2$

S(1 bit)	E(6 bit)	M(9 bit)
1	100101	111011100

Q. +21.75

Implicit ?

1 bit	7 bit	8 bit
S	E	M

10101.11

1.010111×2^4

$M = 010111$

$e = 4, \text{bias} = 2^{7-1}$

$E = 4 + 64$

$E = 68 = (1000100)_2$

Value Formula:

$(-1)^S \times 1.M \times 2^e$

$(-1)^0 \times 1.010111 \times 2^{68-64}$

1.010111×2^4

$10101.11 = (21.75)_{10}$

Ans

S(1bit)	E(7bit)	M(8 bit)
0	1000100	01011100

$\text{Hexadecimal} = (445C)_{16}$

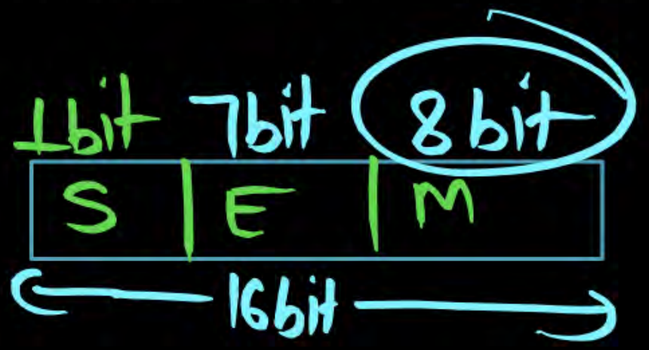
Q.

Consider a 16 bit register used to store floating point number. Mantissa is Implicit normalized signed fraction number. Exponent is in Excess-64 form then

(i)what is the First Smallest Positive number?

(ii)what is the Second Smallest Positive number?

(iii)what is the Difference between First Smallest & Second Smallest Positive number?



Excess − 64 $\Rightarrow$ bias = 64.

$2^{k-1} = 64 \Rightarrow 2^{k-1} = 2^{6}$

$k - 1 = 6$

$k = 7$bit

We Can not Represent '0':

Implicit ⇒ 1.Something $\neq 0$ (Not zero)

OR

Explicit ⇒ 0.1xxx $\neq 0$ (Not zero)

Note In the Implict & Explicit Representation we Can not Represent '0'. In IEEE 754 floating point we Can Represent '0'.

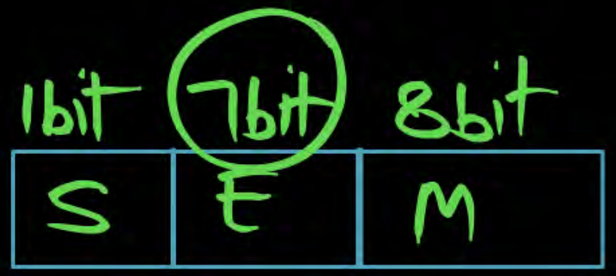
Q.

Consider a 16 bit register used to store floating point number. Mantissa is Implicit normalized signed fraction number. Exponent is in Excess-64 form then

(i)what is the First Smallest Positive number?

(ii)what is the Second Smallest Positive number?

(iii)what is the Difference between First Smallest & Second Smallest Positive number?



$bias = 2^{7-1}$

$= 2^6$

$bias = 64$

I^st Smallest Implicit

E=0

S(1bit)	E(7bit)	M(8bit)
0	0000000	00000000

bias = 64

E = 0

E-bias

$(-1)^S \; 1.M \times 2^{0-64}$

$(-1)^0 \; 1.00000000 \times 2^{-64}$

$I^{st}\text{Smallest} = 1.00000000 \times 2^{-64}$

II^nd Smallest +ve Number

S(1bit)	E(7bit)	
0	0000000	00000001

E-bias

$(-1)^S \; 1.M \times 2^{0-64}$

$(-1)^0 \; 1.00000001 \times 2^{-64}$

$II^{nd}\text{Smallest} = 1.00000001 \times 2^{-64}$

Difference b/w 1st & 2nd Smallest.

$$\text{Difference} = 1.00000001 \times 2^{-64} - 1.000000000 \times 2^{-64}$$

$$[1.00000001 - 1.000000000] \times 2^{-64}$$

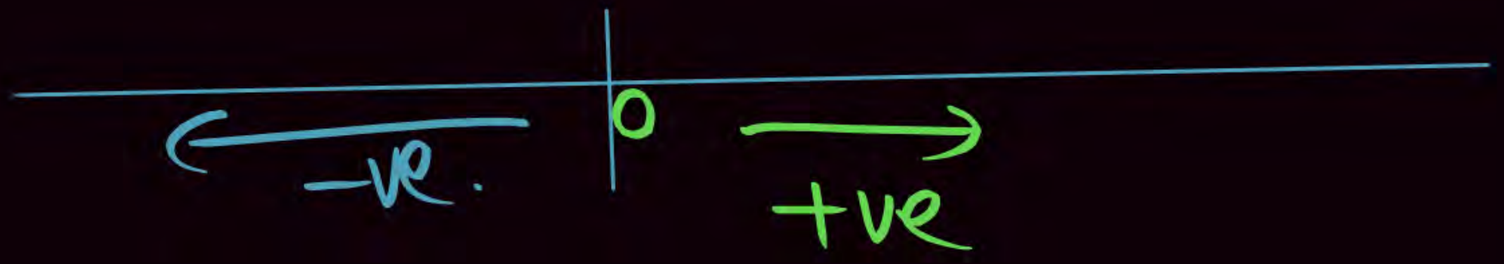
$$\Rightarrow [0.00000001] \times 2^{-64}$$

$$2^{-8} \times 2^{-64}$$

Difference b/w 2 smallest +ve Number.

$$\Rightarrow \boxed{2^{-72}} \quad \underline{\text{Ans}}$$

If We want the Same thing for Negitive.



Q.2

Consider a 16 bit register used to store floating point number. Mantissa is Implicit normalized signed fraction number. Exponent is in Excess-64 form then

(i)what is the First Highest Positive number?

(ii)what is the Second Highest Positive number?

(iii)what is the Difference between First Highest & Second Highest Positive number?

Excess -64 bias = 64

1bit	7bit	8bit
S	E	M

$2^{k-1} = 2^6$

$k-1 = 6$

$k = 7$

(i) 1st Highest +ve Number.

(ii) IInd Highest +ve Number :

Difference : Home work

(1) 0.111 $[1-2^{-3}]$ or $1-1/2^3$

(2) 0.1111 $[1-2^{-4}]$ or $1-1/2^4$

(3) 0.11111 $[1-2^{-5}]$ or $1-1/2^5$

(4) 0.111111 $[1-2^{-6}]$ or $1-1/2^6$.

Proof in Next Pages.

V.V.Imp.

(i) $0.111 \Rightarrow 1-\frac{1}{2^3}$ OR $1-2^{-3}$

(ii) $0.1111 \Rightarrow 1-\frac{1}{2^4}$

(iii) $0.11111 = 1-\frac{1}{2^5}$

(iv) $0.111111 = 1-\frac{1}{2^6}$

Proof

Left Alignment

0.111

$\Rightarrow 111.0 \times 2^{-3}$

$7\,[2^3-1]$

$\Rightarrow [2^3-1] \times 2^{-3}$

$\Rightarrow 2^3 \times 2^{-3} - 1 \times 2^{-3}$

$1-2^{-3}$

OR GP series

$= 1-\frac{1}{2^3}$

Proof

$0.1\,1\,1$

$0.\frac{1}{2}\,\frac{1}{2^2}\,\frac{1}{2^3}$

GP Series.

$a = \frac{1}{2}$

$r = \frac{1}{2}$

$$\frac{a(1-r^n)}{1-r}$$

$$\Rightarrow \frac{\frac{1}{2}\left(1-\frac{1}{2^3}\right)}{\left(1-\frac{1}{2}\right)} = \boxed{1-\frac{1}{2^3}}$$ Ans

Q2 $0.11111 \Rightarrow \boxed{1-\frac{1}{2^5}}$ Ans

Left align

0.11111

$\Rightarrow \underbrace{11111}_{31} . 0 \times 2^{-5}$

$\Rightarrow (2^5 - 1) \times 2^{-5}$

$\Rightarrow 2^5 \times 2^{-5} - 2^{-5}$

$\boxed{1 - 2^{-5}}$ or $\boxed{1 - \frac{1}{2^5}}$

OR

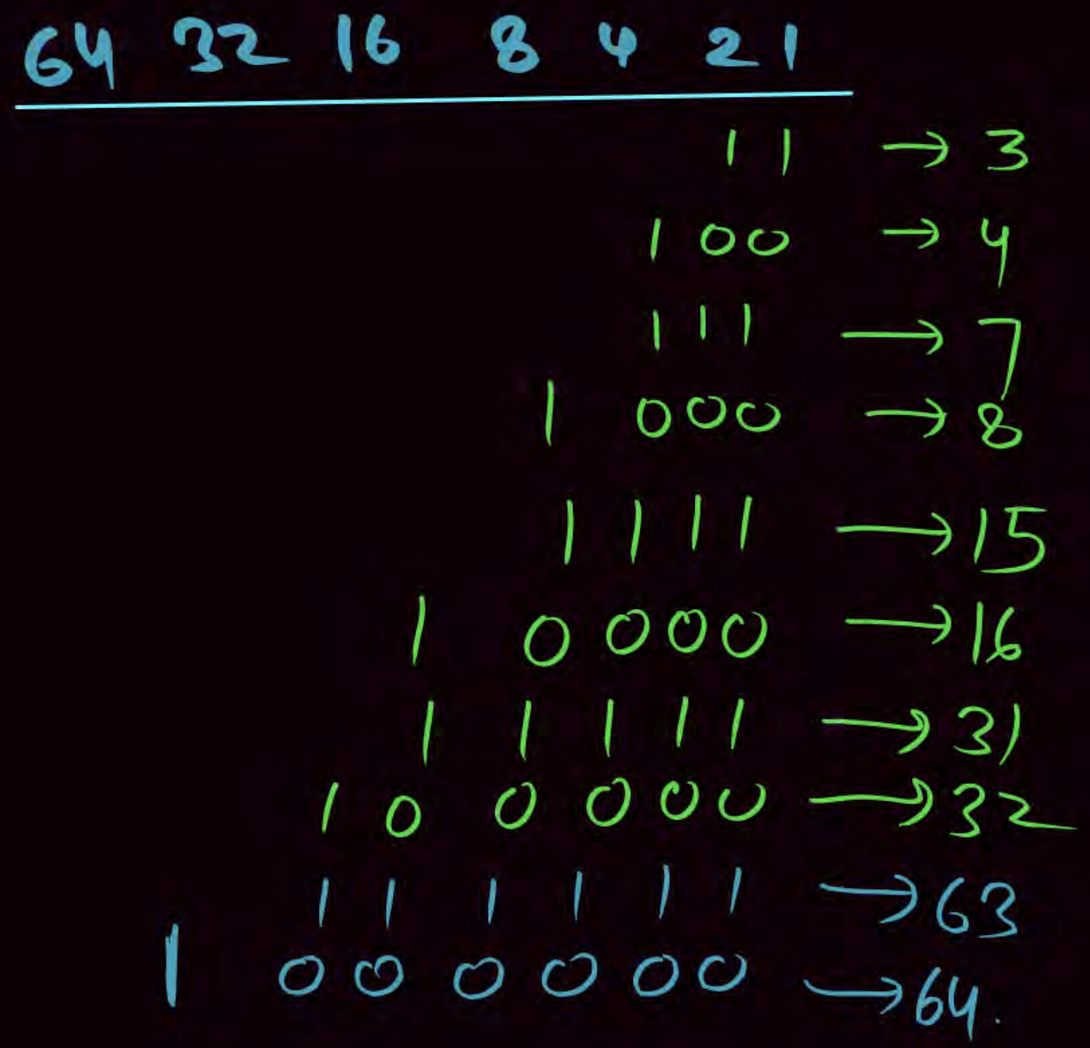
0.11111

GP series $\frac{1}{2}\ \frac{1}{2^2}\ \frac{1}{2^3}\ \frac{1}{2^4}\ \frac{1}{2^5}$

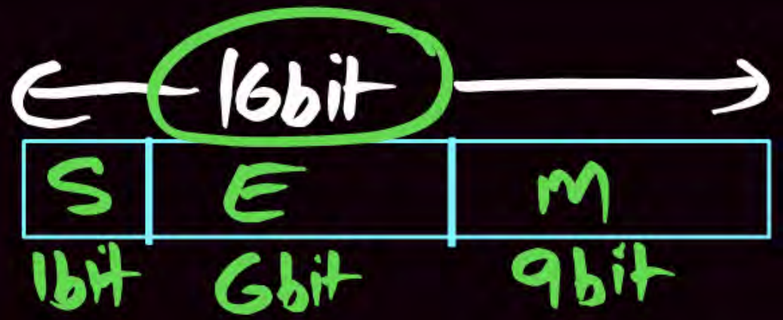
$a = \frac{1}{2} \cdot r = \frac{1}{2}$

$\frac{a(1-r^n)}{1-r} \Rightarrow \frac{\frac{1}{2}(1-\frac{1}{2^5})}{(1-\frac{1}{2})}$

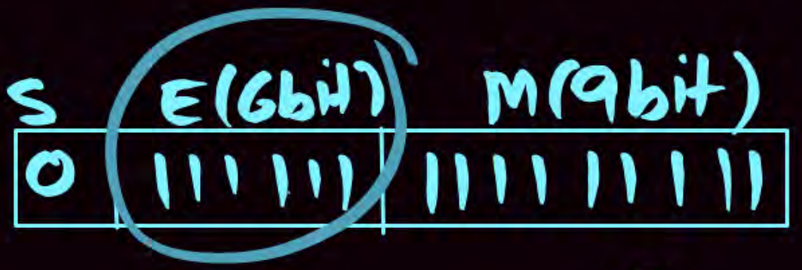
$\Rightarrow \boxed{1-\frac{1}{2^5}}$ Ans


64 32 16 8 4 2 1
1 1 → 3
1 0 0 → 4
1 1 1 → 7
1 0 0 0 → 8
1 1 1 1 → 15
1 0 0 0 0 → 16
1 1 1 1 1 → 31
1 0 0 0 0 0 → 32
1 1 1 1 1 1 → 63
1 0 0 0 0 0 0 → 64.

Q.



maximum +ve using Explicit ?



$$bias = 2^{6-1} = 32$$

bias = 32

E = 111111

E = 63

$$(-1)^S \; 0.m \times 2^{e}$$

$$(-1)^S \; 0.m \times 2^{E-bias}$$

$$(-1)^0 \; 0.111111111 \times 2^{63-32}$$

$$+0.\underbrace{1111111111}_{}\times 2^{31}$$

$$+\left(1-2^{-9}\right)\times 2^{31}$$

$$+2^{31}-2^{31-9}$$

$$+2^{31}-2^{22}$$

$$+2^{31}$$ Ans

1bit	6bit	9bit
S	E	M

← 16bit →

$$0.\underbrace{111111111}_{}1\times 2^{+31}$$

$$\left[\underbrace{111111111}_{}.0\times 2^{-9}\right]\times 2^{31}$$

$$\left[\left(2^{9}-1\right)\times 2^{-9}\right]\times 2^{31}$$

$$\left(1-2^{-9}\right)\times 2^{31}$$

$$\Rightarrow 2^{31}-2^{31-9}$$

$$\Rightarrow +2^{31}-2^{22}$$

$$\Rightarrow +\left(2^{31}\right)$$ Ans

Convential Representation.

↳ Can not Represent '0' & '∞'

Implicit : 1. Something (Not zero).

Explicit : 0.1---- (Not zero).

Disadvantage of Convential Representation

It can not store '0'
Explicit = 0.1
Implicit 1.0
It cannot Represent '∞'
[Infinity]
It can not represent number which is not normalized

IEEE 754 Floating Point Representation

Single Precision
(32 bit)
Excess 127

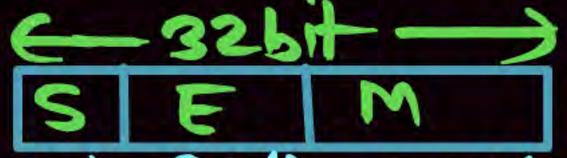
$bias = 2^{K-1} - 1$
$= 2^{8} - 1$
Bias = 127

Double Precision
(64 bit)
Excess 1023

$bias = 2^{11-1} - 1$
Bias = 1023

IEEE 754 Floating Point Representation.

Single Precision Format (32 bit)

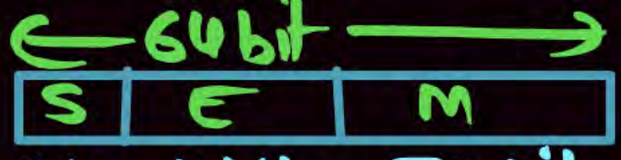


$$bias = 2^{k-1} - 1$$

Excess-127

$$bias = 2^{8-1} - 1 \Rightarrow bias = 127$$

Double Precision Format (64 bit)



$$bias = 2^{k-1} - 1$$

$$bias = 2^{11-1} - 1$$

$$bias = 1023$$

Excess-1023.

IEEE 754 Floating Point Representation.

Single Precision

S	E	M
1bit	8bit	23bit

bias = 127

OR

Excess −127

S	E	M
1bit	11bit	52bit

bias = 1023.

OR

Excess = 1023.

IEEE 754 Floating Point Representation.

Single Precision

S	E	M
1bit	8bit	23bit

bias = 127

OR

Excess - 127

Double Precision

S	E	M
1bit	11bit	52bit

bias = 1023.

OR

Excess = 1023.

Note In IEEE 754 Default is Implicit Normalization

Q. 1 How to represent +(119) into IEEE 754 single precision & Double precision floating Point Representation?

Single Precision.

S	E	M
1bit	8bit	23bit

$119: 1110111.0 \times 2^{0}$

$+1.1101111 \times 2^{+6}$

$S = 0$

$M = 11011110000000000000000$

$e = 6$ $\quad bias = 127$ $\quad E = e + bias = 6 + 127 = 133 = \bar{E}$

$E = 133$

$E = 10000101$

$bias = 2^{8-1} - 1$

$bias = 127$

0	10000101	11011110000000000000000
	4 2 $\bar{E}$ E 0 0 0 0	

$119: (42EE0000)_{16}$

$+(119)$ Single Precision.

$(42EE0000)_{16}$ Ans

Now get the value from $(42EE0000)_{16}$ (P.T.O)

⇓

119 Ans

Single Precision.

$$(-1)^S \; 1.M \times 2^{E-bias}$$

S	E	M
1 bit	8 bit	23 bit

133 − 127

$$bias = 2^{8-1} - 1$$

$$bias = 127$$

$(-1)^0 \; 1.1101110000000000000000 \times 2^{+6}$

$+ \; 1.1101110000000000000000 \times 2^{+6}$

$+1110111.00000000$

(+119) Ans

E = 133

119:

0	10000101	11011100000000000000000
	E (8 bit)	Mantissa (23 bit)

↳ $[42EE0000]_{16}$.

+119)

Now Double Precision.

Q.2 How to represent +(119) into IEEE 754 single precision & Double precision floating Point Representation?

Double Precision.

119: 1110111×2^0

S	E	M
1bit	11bit	52bit

$+1.1101111 \times 2^{+6}$

$bias = 2^{11-1} - 1 \Rightarrow 2^{10} - 1$

bias = 1023.

S=0 | M = 11011100000000000000000000000000000000000000000000

e = +6 | bias = 1023 | E = 1023 + 6 = 1029

E = 1029 | E = 10000000101

S(1bit) | E(11bit) | M(52bit)

0 | 10000000101 | 1101110000000000000000000000000000000000000000

4 0 5 D C 0 0 0 0 0 0

Q. How to represent +(119) into IEEE 754 single precision & Double precision floating Point Representation?

+119 Double Precision

[405DC00000000000]$_{16}$. Ans

Now get the Value.

(2) Double Precision.

$(-1)^S \; 1.M \times 2^{E-bias}$

$(-1)^0 \; 1.11011100000000000\ldots \times 2$

$\Rightarrow + 1.1101110000000000 \times 2^{+6}$

$\Rightarrow 1110111.0000000$

$\Rightarrow (+119)$ Ans

1029 − 1023

S	E	M
1bit	11bit	52bit

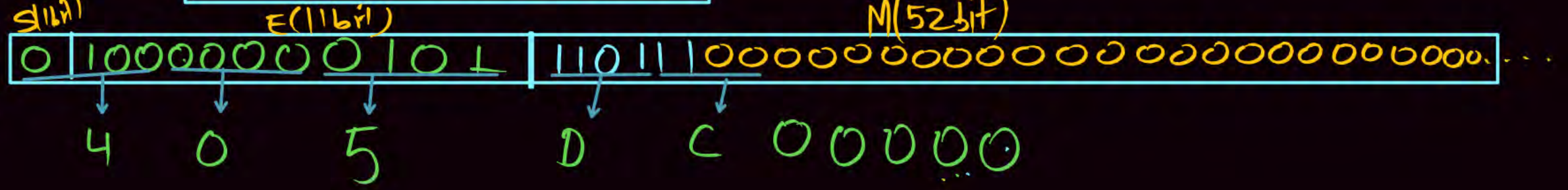
$bias = 2^{11-1} - 1 \Rightarrow 2^{10} - 1$

bias = 1023.

E = 1029

E = 10000000101

(d)



Q. 2 marks

Consider a 32 bit register which stores floating numbers in IEEE single precision format(Implicit). The value of the number, f 32 bit are given below is _____

Sign(1bit)	Exponent(8bit)	Mantissa(23bit)
0	10000100	11000000000000000000000

E = 132 | bias = 127

$(-1)^S \; 1.M \times 2^{E-bias}$

$(-1)^0 \; 1.11000000000 \times 2^{132-127}$

$+ 1.11000000000 \times 2^{+5}$

$\Rightarrow 111000.00000 \times 2^0$

$= +56$

A 48

B 56

C 64

D 60

Solution

Sign(1bit)	Exponent(8bit)	Mantissa(23bit)
0	10000100	11000000000000000000000

$(-1)^S (1.M) \times 2^{E-127}$

$(-1)^0 (1.110.....) \times 2^{132-127}$

$(1.11) \times 2^5$

$= 111000$

$= 56$ Ans

$E = e + bias$

$E - bias$

$E = 10000100$

$E = 132$

Q. GATE 2018

The value of float type variable is represented using the single precision 32 bit floating point IEEE 754 standard use 1 bit sign, 8bit for E and 23 bit mantissa. A float type variable X is assigned the decimal value (–14.25). Then representation of X in Hexa decimal notation is?

(A) 416C0000H

(B) 41640000H

(C) C16C0000H

(D) C1640000H ✓

( )$_{16}$ ( )H

Solution

-14.25

1110.01

1.11001×2^3

$S = 1$

M: 11001

$e = 3$

1 bit	8 bit	23 bit
S	E	M

$\text{Bias} = 2^{8-1} - 1 = 127$

$E = e + \text{bias}$

$= 3 + 127$

$E = 130$

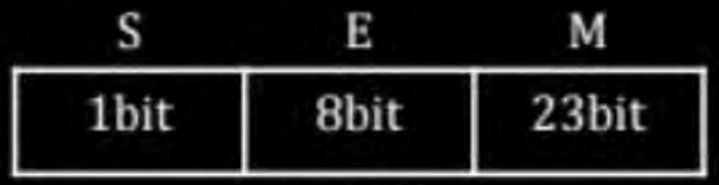
S	E	M
1	10000010	11001 00000000000000

$\text{Hexadecimal} = (C1640000)_H$ Ans

IEEE 754 Floating Point Representation

IEEE 754 Floating Point Representation

Single Precision
(32 bit)
Excess 127



$bias = 2^{K-1} - 1$

$= 2^{8} - 1$

Bias = 127

Double Precision
(64 bit)
Excess 1023

$bias = 2^{11-1} - 1$

Bias = 1023

IEEE 754 Floating Point Representation

Convential Representation

Drawback

Can not Represent

(i) 0

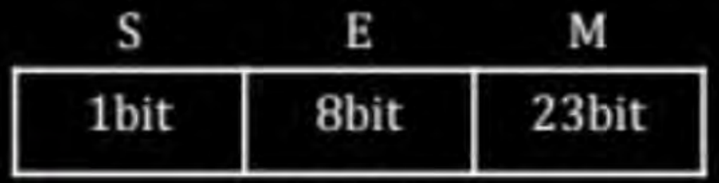
(ii) ∞

(iii) NaN (Not a Number)

(iv) Denormalized Number

IEEE 754 Floating Point Representation

Single Precision (32 bit) Excess 127

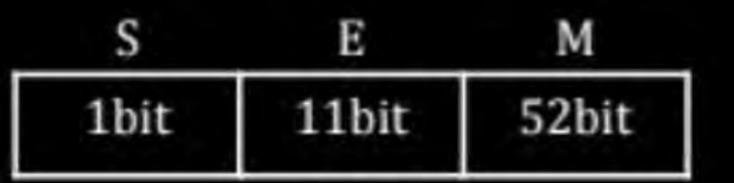


$bias = 2^{K-1} - 1$

$= 2^{8} - 1$

Bias = 127

Double Precision (64 bit) Excess 1023



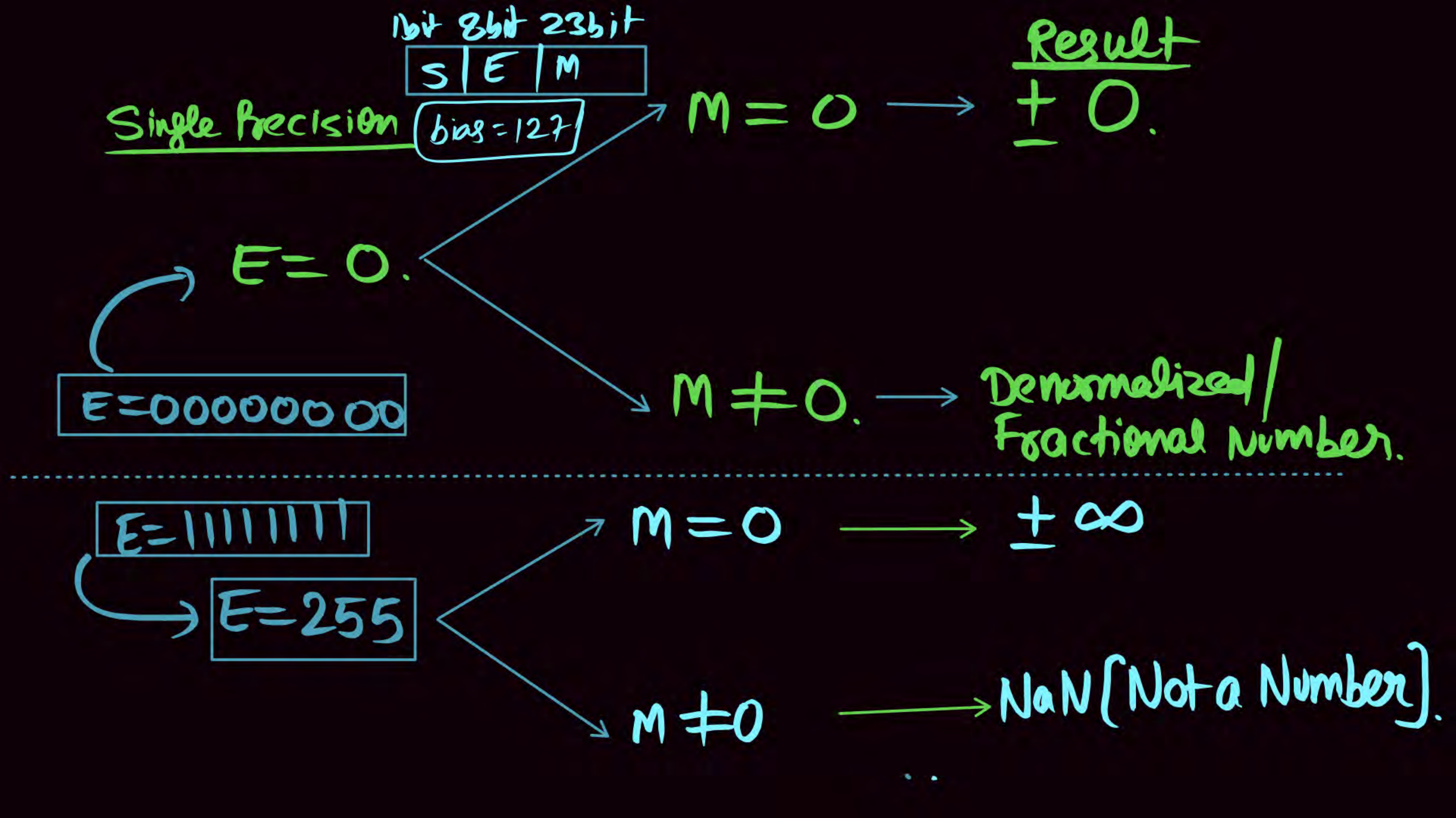
$bias = 2^{11-1} - 1$

Bias = 1023

Single Precision (32 bit)

S	E	M
1 bit	8 bit	23 bit

Sign(1 bit)	E(1 bit)	M(23 bit)	Value
0 or 1	00000000 E = 0	00000000000000000000000 M = 0	±0
0 or 1	11111111 E = 255	00000000000000000000000 M = 0	±∞
0 or 1	1 ≤ E ≤ 254	M = ...	Implicit Normalized form $(-1)^S \times 1.M \times 2^e$ $(-1)^S \times 1.M \times 2^{E-127 \text{ bias}}$
0 or 1	E = 0	M ≠ 0	Denormalized number/Fractional form $(-1)^S \times 0.M \times 2^{E-127 \text{ bias}}$
0 or 1	E = 255	M ≠ 0	Not a Number (NAN)


1bit 8bit 23bit
S | E | M
Single Precision
bias = 127
Result
M = 0 ⟶ ± 0.
E = 0.
E = 00000000
M ≠ 0. ⟶ Denormalized / Fractional Number.
E = 11111111
E = 255
M = 0 ⟶ ± ∞
M ≠ 0 ⟶ NaN [Not a Number].

Q. WHy $bias = 2^{k-1} - 1$?

Single Precision

1bit	8bit	23bit
S	E	M

If in Single Precision, if $bias = 2^{8-1} \Rightarrow 2^7$ (bias = 128) then there is a possibility to getting $E = 255$

$E = e + bias$

$e = 127$ then $E = 127 + 128$

$E = 255$

8 bit 2's complement Range $= -2^{8-1}$ to $+2^{8-1} - 1$

$\Rightarrow$ −128 to +127

So $bias = 2^{k-1} - 1$ Ans

If E=255

- $\pm\infty$
- Not a Number

Double Precision

Double Precision

Excess - 1023

Sign (1 bit)	E(11 bit)	M(52 bit)	Value
0 or 1	0000 0000 000 E = 0	00000000000.. M = 0	± 0
0 or 1	1111 1111 111 E = 2047	0000000000.. M = 0	± ∞
0 or 1	1 ≤ E ≤ 2046	M = _ _ _ _ _ _ _ _ _ _	Implicit Normalization $(-1)^S \cdot M \times 2^e$ $(-1)^S \times 1 \cdot M \times 2^{E-1023}$
0 or 1	E = o	M ≠ 0	Denormalized number/ Fractional Form $(-1)^S 0 \cdot M \times 2^{E-1023}$
	E = 2047	M ≠ 0	Not a number

Double Precision

bias = 1027

E = 0

- M = 0 → Result ± 0.
- M ≠ 0 → Denormalized / Fractional Number.

E = 00000000000

E = 11111111111

E = 2047

- M = 0 → ± ∞
- M ≠ 0 → NaN [Not a Number].

NOTE: When $E = 0$ then Value 0 or fractional form

when $M = 0$

when $M \neq 0$

NOTE: When $E = 2047$ then Value ∞ or Not a Number (NAN)

when $M = 0$

when $M \neq 0$

IEEE - 754

Features.

Note Features IEEE 754 are special symbols to represent unusual events....

1. For example instead of interrupting on a divide by 0,softwarecan set the result to a bit pattern representing $+\infty$,- ∞ ; The largest exponent is reserved for these special symbols.

2. IEEE 754 has a symbols for the result of invalid operations, such 0/0, or subtract infinity from infinity . This symbols is NaN, for Not a Number.

   The purpose of NaN is to allow programmer to postpone some test and decisions to a later time in this program. (when it is convenient)

Q. How to represent +(1.0) into IEEE 754 single precision floating Point Repartition?

Sol. To represent +(1.0) into IEEE 754 single precision floating Point Repartition..

$+ 1.0 \times 2^0$

$M = 0$

$e = 0$

$E = 127$

S	E	M
1	8	23

$Bias = 2^{8-1}-1 = 127$

$E = e + bias$

$E = 0 + 127$

0	0 1111111	0000 0000 0000 ....

3 F 8 0 0

(3F80 0000)

Q. What will ne maximum positive vale(+ve) in IEEE 754 single precision floating Point Representation?

Q.

Consider a 64 bit Register which store floating point Number in IEEE 754 Double Precision Format.
What is the value of the Number if 64 bit are given as
0111 1111 1111 0000 0000 0000000000....

Sol.

Consider a 64 bit Register which store floating point Number in IEEE 754 Double Precision Format.

What is the value of the Number if 64 bit are given as
0111 1111 1111 0000 0000 0000000000....

S(1 bit)	E(11 bit)	M(52 bit)
0	1111 1111 111	0000000000000

Here all exponent bits is 1

E = 2047 → Infinity (when M = 0)

E = 2047 → Not a number (when M ≠ 0)

Here M = 0 so its $+\infty$

MCQ

The value of a *float* type variable is represented using the single-precision 32-bit floating point format of IEEE-754 standard that uses 1 bit for sign, 8 bits for biased exponent and 23 bits for mantissa. A *float* type variable X is assigned the decimal value of −14.25. The representation of X in hexadecimal notation is

[GATE-2014-Set2-CS: 2M]

A C1640000H

B 416C0000H

C 41640000H

D C16C0000H

MCQ

Consider the IEEE-754 single precision floating point numbers
P = 0xC1800000 and Q = 0x3F5C2EF4.
Which one of the following corresponds to the product of these numbers (i.e., P × Q), represented in the IEEE-754 single precision format?

[GATE-2023-CS: 2M]

Round off.

A 0x404C2EF4

B 0x405C2EF4

C 0xC15C2EF4

D 0xC14C2EF4

Add/Subtract Rule

1. Choose the number with the smaller exponent and shift its mantissa right a number of steps equal to the difference in exponents.
2. Set the exponent of the result equal to the larger exponent.
3. Perform addition/subtraction on the mantissas and determine the sign of the result.

   Normalize the resulting value, if necessary.

Multiplication and division are somewhat easier than addition and subtraction, in that no alignment of mantissas is needed.

Multiply Rule

1. Add the exponents and subtract 127.
2. Multiply the mantissas and determine the sign of the result.
3. Normalize the resulting value, if necessary.

Divide Rule

1. Subtract the exponents and add 127.
2. Divide the mantissas and determine the sign of the result.
3. Normalize the resulting value, if necessary.

The addition or subtraction of 127 in the multiply and divide rules results from using the excess –127 notation for exponents.

THANK YOU!