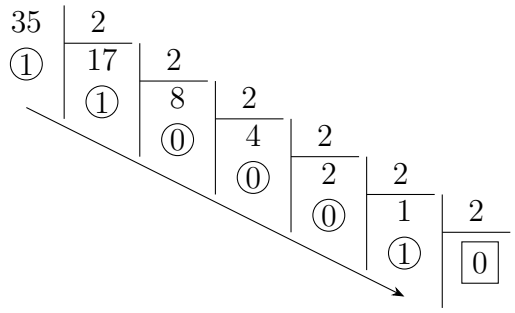
Solution de Série TD2

Exercice 1

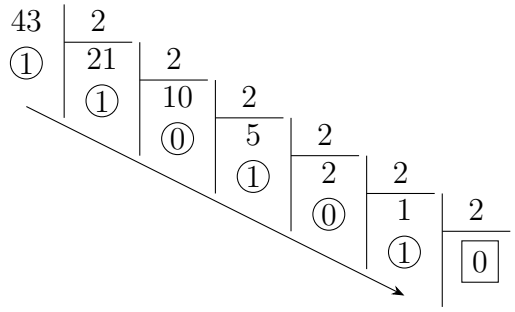
Encodage des valeurs décimaux en SVA, C1, et C2 sur 8 bits :

$(+35)_{10} = (+\underleftarrow{10\,0011})_2$



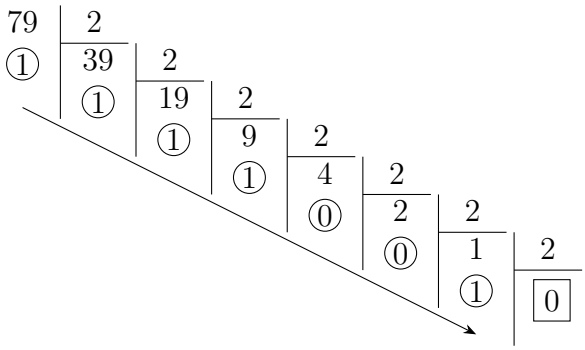
Encodage en Signe et Valeur Absolue :
$(+35)_{10} \rightarrow (0|010\,0011)_{SVA}$
Encodage en Complément-à-1 :
$(+35)_{10} \rightarrow (0|010\,0011)_{C1}$
Encodage en Complément-à-2 :
$(+35)_{10} \rightarrow (0|010\,0011)_{C2}$

$(-43)_{10} = (-\underleftarrow{10\,1011})_2$



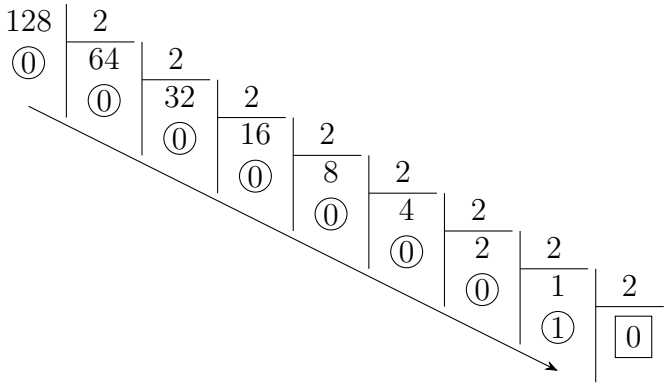
Encodage en Signe et Valeur Absolue :
$(-43)_{10} \rightarrow (1|010\,1011)_{SVA}$
Encodage en Complément-à-1 :
$(-43)_{10} \rightarrow (1|\overline{010\,1011}) \rightarrow (1|101\,0100)_{C1}$
Encodage en Complément-à-2 :
$(-43)_{10} \rightarrow (1|101\,0100)_{C1} + 1$
$\rightarrow (1|101\,0101)_{C2}$

$(-79)_{10} = (-\underleftarrow{100\,1111})_2$



Encodage en Signe et Valeur Absolue :
$(-79)_{10} \rightarrow (1|100\,1111)_{SVA}$
Encodage en Complément-à-1 :
$(-79)_{10} \rightarrow (1|\overline{100\,1111}) \rightarrow (1|011\,0000)_{C1}$
Encodage en Complément-à-2 :
$(-79)_{10} \rightarrow (1|011\,0000)_{C1} + 1$
$\rightarrow (1|011\,0001)_{C2}$

$(-128)_{10} = (-\underleftarrow{1000\,0000})_2$



Encodage en Signe et Valeur Absolue :
$(-128)_{10} \notin [-127, +127]$: non représentable.
Encodage en Complément-à-1 :
$(-128)_{10} \notin [-127, +127]$: non représentable.

Encodage en Complément-à-2 :
$(-128)_{10} \in [-128, +127]$: représentable
$(-128)_{10} \rightarrow (1|\overline{000\ 0000}) + 1$
$\rightarrow (1|111\ 1111)_{C1} + 1$
$\rightarrow (1|000\ 0000)_{C2}$

Décodage des représentations EnS, SVA, C1, et C2 sur 8 bits en valeurs décimales :

Entier non Signé :

$(0001\ 1010)_{EnS} \rightarrow (0001\ 1010)_2 = 2^4 + 2^3 + 2^1 = \boxed{(+26)_{10}}$

$(1000\ 1101)_{EnS} \rightarrow (1000\ 1101)_2 = 2^7 + 2^3 + 2^2 + 2^0 = \boxed{(+141)_{10}}$

Signe et Valeur Absolue :

$(0|010\ 1100)_{SVA} \rightarrow (+10\ 1100)_2 = +(2^5 + 2^3 + 2^2) = \boxed{(+44)_{10}}$

$(1|000\ 1110)_{SVA} \rightarrow (-1110)_2 = -(2^3 + 2^2 + 2^1) = \boxed{(-14)_{10}}$

Complement-à-1 :

$(0|001\ 0111)_{C1} \rightarrow (+1\ 0111)_2 = +(2^4 + 2^2 + 2^1 + 2^0) = \boxed{(+23)_{10}}$

$(1|100\ 1011)_{C1} \rightarrow (1|\overline{100\ 1011})$
$\rightarrow (1|011\ 0100)$
$\rightarrow (-11\ 0100)_2 = -(2^5 + 2^4 + 2^2) = \boxed{(-52)_{10}}$

Complement-à-2 :

$(0|001\ 1001)_{C2} \rightarrow (+1\ 1001)_2 = +(2^4 + 2^3 + 2^0) = \boxed{(+25)_{10}}$

$(1|010\ 1110)_{C2} \rightarrow (1|\overline{010\ 1110}) + 1$
$\rightarrow (1|101\ 0001) + 1$
$\rightarrow (1|101\ 0010)$
$\rightarrow (-101\ 0010)_2 = -(2^6 + 2^4 + 2^1) = \boxed{(-82)_{10}}$

Exercice 2

1. Règles de calcul d'intervalles des valeurs représentables sur n bits :

Encodage	Intervalle
Entier non Signé	$[0, +2^n - 1]$
Signe et Valeur Absolue	$[-(2^{n-1} - 1), +(2^{n-1} - 1)]$
Complément-à-1	$[-(2^{n-1} - 1), +(2^{n-1} - 1)]$
Complément-à-2	$[-(2^{n-1}), +(2^{n-1} - 1)]$

Calcul d'intervalles des types entiers d'un langage de programmation comme le Java :

Type	Formule de calcul	Intervalle
byte (8 bits)	$[-(2^{8-1}), +(2^{8-1} - 1)]$	$[-128, +127]$
short (16 bits)	$[-(2^{16-1}), +(2^{16-1} - 1)]$	$[-32\,768, +32\,767]$
int (32 bits)	$[-(2^{32-1}), +(2^{32-1} - 1)]$	$[-2\,147\,483\,648, +2\,147\,483\,647]$
long (64 bits)	$[-(2^{64-1}), +(2^{64-1} - 1)]$	$\approx [-9 \times 10^{18}, +9 \times 10^{18}]$

NB : Tous les langages de programmation et toutes les machines actuelles utilisent le Complément-à-2 pour représenter les valeurs entières signées. Les valeurs non signées (Unsigned) quand à-elles, sont représentées en Entier non Signé.

2. Faire les opérations d'addition en C2. a, b, et c étant des variables de type byte :

On a le code en Java :

```
a = 18 ;
b = 25 ;
c = a+b ;
```

$a = (18)_{10} = (+1\,0010)_2 \rightarrow (0|001\,0010)_{C2}$ $\quad b = (25)_{10} = (+1\,1001)_2 \rightarrow (0|001\,1001)_{C2}$

$(18)_{10} = (1\underleftarrow{0010})_2$

18	2				
⓪	9	2			
	①	4	2		
		⓪	2	2	
			①	1	2
				①	$\boxed{0}$

$(25)_{10} = (1\underleftarrow{1001})_2$

25	2				
①	12	2			
	⓪	6	2		
		⓪	3	2	
			①	1	2
				①	$\boxed{0}$

Le calcul de l'addition en C2 de c=a+b; normalement on devrait trouver $c = 18 + 25 = 43$ en C2 :

$$\begin{array}{rl} & 0|\overset{1}{0}01\,1001 \\ + & 0|001\,0010 \\ \hline = & 0|010\,1011 \end{array}$$

$(0|010\,1011)_{C2} \rightarrow (+10\,1011)_2 = +(2^5 + 2^3 + 2^1 + 2^0) = \boxed{(+43)_{10}} = 43$

Le résultat en C2 est conforme à la valeur prévue 43, donc on peut dire que le calcul est correct et qu'il n'y a pas d'overflow.

On a aussi le code en Java :

```
a = 96 ;
b = 50 ;
c = a+b ;
```

$a = (96)_{10} = (+1100000)_2 \rightarrow (0|1100000)_{C2} \quad b = (50)_{10} = (+11\,0010)_2 \rightarrow (0|011\,0010)_{C2}$

$(96)_{10} = (\underleftarrow{110\,0000})_2$

96	2					
⓪	48	2				
	⓪	24	2			
		⓪	12	2		
			⓪	6	2	
				⓪	3	2
					①	1	2
						①	$\boxed{0}$

$(50)_{10} = (\underleftarrow{11\,0010})_2$

50	2				
⓪	25	2			
	①	12	2		
		⓪	6	2	
			⓪	3	2
				①	1	2
					①	$\boxed{0}$

Le calcul de l'addition en C2 de `c=a+b` ; normalement on devrait trouver $c = 96 + 50 = 146$ en C2 :

$$\begin{array}{rr} & \overset{1}{0}|\overset{1}{1}10\,0000 \\ + & 0|011\,0010 \\ \hline = & 1|001\,0010 \end{array}$$

$$\begin{aligned}(1|001\,0010)_{C2} &\rightarrow (1|\overline{001\,0010}) + 1 \\ &\rightarrow (1|110\,1101) + 1 \\ &\rightarrow (1|110\,1110) \\ &\rightarrow (-110\,1110)_2 = -(2^6 + 2^5 + 2^3 + 2^2 + 2^1) = \boxed{(-110)_{10}} \neq 146\end{aligned}$$

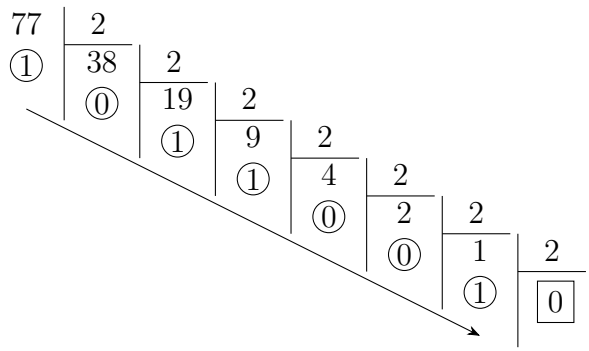
Le résultat en C2 est différent de la valeur prévue 146, dans ce cas on peut dire qu'il y a erreur dans le calcul de la machine et qu'il y a overflow. La raison de l'overflow est que $146 \notin [-128, +127]$ du type `byte`.

On a aussi le code en Java :

```
a = 77 ;
b = -50 ;
c = a+b ;
```

$a = (77)_{10} = (+1001101)_2 \rightarrow (0|1001101)_{C2}$

$(77)_{10} = (\underleftarrow{100\ 1101})_2$



$b = (-50)_{10} = (-11\,0010)_2$
$\rightarrow (1|\overline{011\ 0010}) + 1$
$\rightarrow (1|100\ 1101) + 1$
$\rightarrow (1|100\ 1110)_{C2}$

50 en binaire étant calculée précédemment.

Le calcul de l'addition en C2 de c=a+b ; normalement on devrait trouver $c = 77 - 50 = 27$ en C2 :

$$\begin{array}{r} {\scriptstyle 1\,1\quad\quad 1\ 1} \\ 0|100\ 1101 \\ +\ \ 1|100\ 1110 \\ \hline = \cancel{1}0|001\ 1011 \end{array}$$

NB : Le 9-ième bit dans ce cas est éliminé et n'est pris en compte par la machine, l'addition en C2 a l'avantage d'être plus simple pour le traitement du 9-ième bit. Tant dit que le 8-ième bit reste toujours le bit de signe.

$(0|001\ 1011)_{C2} \rightarrow (+1\ 1011)_2 = +(2^4 + 2^3 + 2^1 + 2^0) = \boxed{(+27)_{10}} = 27$

Le résultat en C2 est conforme à la valeur prévue 27, donc on peut dire que le calcul est correct et qu'il n'y a pas d'overflow.

On a aussi le code en Java :

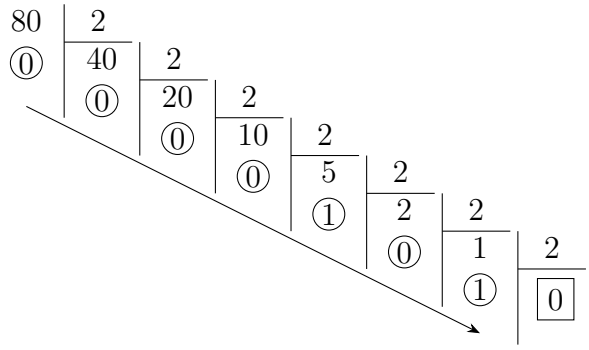
```
a = -50 ;
b = -80 ;
c = a+b ;
```

$a = (-50)_{10} = (1|100\ 1110)_{C2}$

-50 en C2 était calculée précédemment.

$$\begin{aligned} b = (-80)_{10} &= (-\underline{101\ 0000})_2 \\ &\rightarrow (1|\overline{101\ 0000}) + 1 \\ &\rightarrow (1|010\ 1111) + 1 \\ &\rightarrow (1|011\ 0000)_{C2} \end{aligned}$$

$(80)_{10} = (\underleftarrow{101\ 0000})_2$



Le calcul de l'addition en C2 de `c=a+b;` normalement on devrait trouver $c = -50 + (-80) = -130$ en C2 :

$$\begin{array}{r} {\scriptstyle 1}\ \ 1|100\ 1110 \\ +\ \ 1|011\ 0000 \\ \hline = \cancel{1}0|111\ 1110 \end{array}$$

$(0|111\ 1110)_{C2} \rightarrow (+111\ 1110)_2 = +(2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1) = \boxed{(+126)_{10}} \neq -130$

Le résultat en C2 est différent de la valeur prévue -130, dans ce cas on peut dire qu'il y a erreur dans le calcul de la machine et qu'il y a overflow. La raison de l'overflow est que $-130 \notin [-128, +127]$ du type `byte`.

Rappel : Le C2 possède plusieurs avantages par rapport aux autres encodages et l'addition en fait partie, l'addition en C2 est très simple, quelque soit le signe des valeurs, positif ou négatif, elle est effectuée bit par bit comme pour l'addition binaire, même le bit de signe est additionné, et le résultat reste correct en C2 avec le 8-ième bit comme bit de signe, même s'il y a un 9-ième bit ce dernier est ignoré et le résultat demeure correct. La machine n'a même pas besoin d'avoir un soustracteur en C2, puisque l'additionneur peut aussi faire la soustraction en changeant le signe du 2-ième opérande, c-à-d $A - B$ devient $A + (-B)$. Tous ces avantages ont conduit le C2 à devenir l'encodage prédominant dans les machines actuelles.

Définition de l'overflow : L'overflow ou le dépassement de capacité ou le débordement est un phénomène qui surgit dans les machines comme un défaut, dans lequel la machine produit un résultat erroné, la raison est que la machine possède un nombre limité de bits pour représenter les valeurs. Pour détecter l'overflow il existe plusieurs méthodes, par exemple c'est détecté lorsque le résultat de la machine est différent du résultat mathématique, ou lorsque le résultat du calcul ne peut être contenu dans le nombre de bits réservés, mais la manière la plus facile pour détecter overflow est observant les bits de signe, si l'addition d'un nombre positif avec un nombre positif donne un nombre négatif, ou inversement, un nombre négatif avec négatif donne un nombre positif, alors sûrement il y a overflow.

Solution pour l'overflow : La plus simple méthode pour contourner l'overflow est d'augmenter le nombre de bits pour la représentation des valeurs, ça se traduit par le changement de type vers `short`, `int` ou plus, dans le langage de programmation.

Si on prend l'exemple de l'instruction en Java `short a = 45000 ;` on peut constater qu'il y a overflow, puisque $+45000 \notin [-32\,768, +32\,767]$

Exercice 3

1. La représentation des nombres réels en virgule flottante IEEE 754 simple précision :

La représentation IEEE 754 en binaire est comme suite :

$$\underbrace{\overbrace{Signe(S)}^{1\text{ bit}} \mid \overbrace{ExposantDécalé(E_\Delta)}^{8\text{ bits}} \mid \overbrace{Mantisse(M)}^{23\text{ bits}}}_{32\text{ bits}}$$

Les nombres réels en binaire sont décrit par les champs en haut comme suite :

$\boxed{N = S \cdot 1, M \times 2^{E_r}}$ en sachant que : $\boxed{E_\Delta = E_r + \Delta}$

E_r est l'exposant réel, il est décalé de $\Delta = 127$, Δ c'est sont biais de décalage.

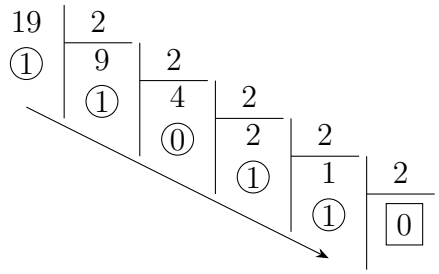
$Remarque1$: Il est impératif de transformer le nombre réel avant de formuler la représentation IEEE 754 en écriture binaire normalisé (écriture scientifique), *ex:* $-101,01 \rightarrow -1,0101 \times 2^2$.

$Remarque2$: Le décalage dans IEEE 754 est souvent appelé *Biais* et l'Exposant Décalé (E_Δ) est appelé *Exposant Biaisé* (E_b).

$Remarque3$: Le bit de signe dans l'IEEE 754 a la même signification comme pour les entiers signés, ça veut dire que 0 est le signe (+) et 1 est le signe (-).

On veut écrire la valeur +19 au format IEEE 754 simple précision :

$(+19)_{10} = (+1\,0011)_2 = +1,0011 \times 2^{+4}$
$(19)_{10} = (1,\underleftarrow{0011})_2$



D'après la formule $N = S \cdot 1, M \times 2^{E_r}$:
$S = +$
$M = 0011$
$E_r = +4$
et on a :
$E_\Delta = E_r + \Delta = 4 + 127 = 131 = (1000\,0011)_2$

Donc la représentation est : $\boxed{(0|1000\,0011|0011\cdots0)_2 = (4198\,0000)_{16}}$

On veut écrire la valeur -0,625 au format IEEE 754 simple précision :

$(-0,625)_{10} = (-0,101)_2 = -1,01 \times 2^{-1}$
$(0,625)_{10} = (0,\underrightarrow{101})_2$

$0,625 \times 2 = 1,250$
$0,250 \times 2 = 0,500$
$0,500 \times 2 = 1,\boxed{000}$ ↓

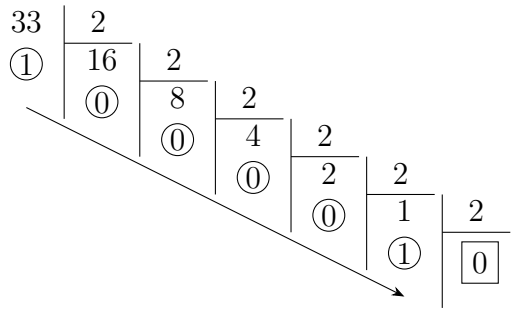
D'après la formule $N = S \cdot 1, M \times 2^{E_r}$:
$S = -$
$M = 01$
$E_r = -1$
et on a :
$E_\Delta = E_r + \Delta = -1 + 127 = 126 = (0111\,1110)_2$

Donc la représentation est : $\boxed{(1|0111\,1110|01\cdots0)_2 = (BF20\,0000)_{16}}$

On veut écrire la valeur -33,0625 au format IEEE 754 simple précision :

$(-33,0625)_{10} = (-10\,0001,0001)_2 = -1,0000\,1000\,1 \times 2^{+5}$

$(33,0625)_{10} = (\underleftarrow{10\,0001},\underrightarrow{0001})_2$



$0,0625 \times 2 = 0,1250$
$0,1250 \times 2 = 0,2500$
$0,2500 \times 2 = 0,5000$
$0,5000 \times 2 = 1,\boxed{0000}$

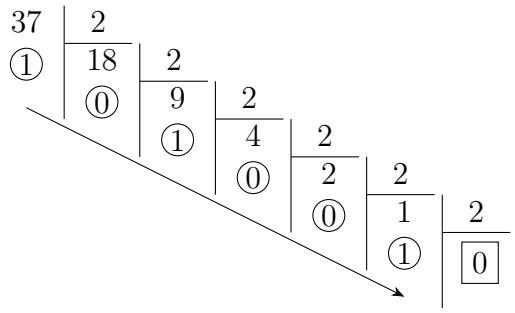
D'après la formule $N = S \cdot 1,M \times 2^{E_r}$:
$S = -$
$M = 0000\,1000\,1$
$E_r = +5$
et on a :
$E_\Delta = E_r + \Delta = +5 + 127 = 132 = (1000\,0100)_2$

Donc la représentation est : $\boxed{(1|1000\,0100|0000\,1000\,1\cdots0)_2 = (C204\,4000)_{16}}$

On veut écrire la valeur +37,75 au format IEEE 754 simple précision :

$(+37,75)_{10} = (+10\,0101,11)_2 = +1,0010\,111 \times 2^{+5}$

$(37,75)_{10} = (\underleftarrow{10\,0101},\underrightarrow{11})_2$



$0,75 \times 2 = 1,5$
$0,50 \times 2 = 1,\boxed{0}$

D'après la formule $N = S \cdot 1,M \times 2^{E_r}$:
$S = +$
$M = 0010\,111$
$E_r = +5$
et on a :
$E_\Delta = E_r + \Delta = +5 + 127 = 132 = (1000\,0100)_2$

Donc la représentation est : $\boxed{(0|1000\,0100|0010\,111\cdots0)_2 = (4217\,0000)_{16}}$

2. Faire l'opération inverse, à partir de la représentation binaire (ou hexadécimal) retrouver la valeur réel représentée :

On a :
$(4296\,0000)_{16} = (0100\,0010\,1001\,0110\,0000\,0000\,0000\,0000)_2$
Après subdivision des champs IEEE 756, on obtient :
$(4296\,0000)_{16} = (0|1000\,0101|0010\,11\cdots0)_2$

Donc :
$S = +$
$M = 0010\,11$
$E_\Delta = (1000\,0101)_2 = (133)_{10}$
Puisque on a : $E_\Delta = E_r + \Delta \Rightarrow E_r = E_\Delta - \Delta = 133 - 127 = \boxed{6}$

La valeur réel est calculée selon la formule :
$N = S \cdot 1, M \times 2^{E_r} = +1,0010\,11 \times 2^6 = (+100\,1011)_2$
$N = 2^6 + 2^3 + 2^1 + 2^0$
$N = \boxed{+75}$

On a :
$(C164\,0000)_{16} = (1100\,0001\,0110\,0100\,0000\,0000\,0000\,0000)_2$
Après subdivision des champs IEEE 756, on obtient :
$(C164\,0000)_{16} = (1|1000\,0010|1100\,1\cdots0)_2$

Donc :
$S = -$
$M = 1100\,1$
$E_\Delta = (1000\,0010)_2 = (130)_{10}$
Puisque on a : $E_\Delta = E_r + \Delta \Rightarrow E_r = E_\Delta - \Delta = 130 - 127 = \boxed{3}$

La valeur réel est calculée selon la formule :
$N = S \cdot 1, M \times 2^{E_r} = -1,1100\,1 \times 2^3 = (-1110,01)_2$
$N = -(2^3 + 2^2 + 2^1 + 2^{-2})$
$N = \boxed{-14,25}$

On a :
$(BFA0\,0000)_{16} = (1011\,1111\,1010\,0000\,0000\,0000\,0000\,0000)_2$
Après subdivision des champs IEEE 756, on obtient :
$(BFA0\,0000)_{16} = (1|0111\,1111|01\cdots0)_2$

Donc :
$S = -$
$M = 01$

$E_\Delta = (0111\,1111)_2 = (127)_{10}$
Puisque on a : $E_\Delta = E_r + \Delta \Rightarrow E_r = E_\Delta - \Delta = 127 - 127 = \boxed{0}$

La valeur réel est calculée selon la formule :
$N = S \cdot 1, M \times 2^{E_r} = -1,01 \times 2^0 = (-1,01)_2$
$N = -(2^0 + 2^{-2})$
$N = \boxed{-1,25}$

On a :
$(C230\,0000)_{16} = (1100\,0010\,0011\,0000\,0000\,0000\,0000\,0000)_2$
Après subdivision des champs IEEE 756, on obtient :
$(C230\,0000)_{16} = (1|1000\,0100|011\cdots0)_2$

Donc :
$S = -$
$M = 011$
$E_\Delta = (1000\,0100)_2 = (132)_{10}$
Puisque on a : $E_\Delta = E_r + \Delta \Rightarrow E_r = E_\Delta - \Delta = 132 - 127 = \boxed{+5}$

La valeur réel est calculée selon la formule :
$N = S \cdot 1, M \times 2^{E_r} = -1,011 \times 2^{+5} = (-10\,1100)_2$
$N = -(2^5 + 2^3 + 2^2)$
$N = \boxed{-44}$

Retrouver la valeur réel, à partir de représentation binaire en IEEE 754 double précision :

La représentation IEEE 754 en double précision est comme suite :

$$\underbrace{\overbrace{Signe(S)}^{1\text{ bit}} \mid \overbrace{ExposantDécalé(E_\Delta)}^{11\text{ bits}} \mid \overbrace{Mantisse(M)}^{52\text{ bits}}}_{64\text{ bits}}$$

Toutes les formules restes les mêmes sauf la valeur de Δ, qui devient $\Delta = 1023$.

$Remarque1$: À vrai dire, il existe une formule pour calculer Δ en fonction de n, n étant le nombre de bits dans le champ E_Δ. La formule est $\Delta_n = 2^{n-1} - 1$.

$Remarque2$: La formule ci-dessus est facilement vérifiable en s'appuyant sur le fait que Δ doit toujours prendre en binaire la forme $\Delta_n = (\underbrace{0111\cdots1}_{\text{n bits}})_2$.

On a :
$(C044\,2000\,0000\,0000)_{16} = (1100\,0000\,0100\,0100\,0010\,00\cdots0)_2$
Après subdivision des champs IEEE 756, on obtient :
$(C044\,2000\,0000\,0000)_{16} = (1|100\,0000\,0100|0100\,001\cdots0)_2$

Donc :
$S = -$
$M = 0100\,001$
$E_\Delta = (100\,0000\,0100)_2 = (1028)_{10}$
Puisque on a : $E_\Delta = E_r + \Delta \Rightarrow E_r = E_\Delta - \Delta = 1028 - 1023 = \boxed{+5}$

La valeur réel est calculée selon la formule :
$N = S \cdot 1, M \times 2^{E_r} = -1,0100\,001 \times 2^{+5} = (-10\,1000,01)_2$
$N = -(2^5 + 2^3 + 2^{-2})$
$N = \boxed{-40,25}$

On a :
$(4049\,8000\,0000\,0000)_{16} = (0100\,0000\,0100\,1001\,1000\cdots0)_2$
Après subdivision des champs IEEE 756, on obtient :
$(4049\,9800\,0000\,0000)_{16} = (0|100\,0000\,0100|1001\,1\cdots0)_2$

Donc :
$S = +$
$M = 1001\,1$
$E_\Delta = (100\,0000\,0100)_2 = (1028)_{10}$
Puisque on a : $E_\Delta = E_r + \Delta \Rightarrow E_r = E_\Delta - \Delta = 1028 - 1023 = \boxed{+5}$

La valeur réel est calculée selon la formule :
$N = S \cdot 1, M \times 2^{E_r} = +1,1001\,1 \times 2^{+5} = (+11\,0011)_2$
$N = +(2^5 + 2^4 + 2^1 + 2^0)$
$N = \boxed{+51}$

---

3. Il existe quelques constantes dans le standard IEEE 754, $+0$, -0, $+\infty$, $-\infty$, NaN, elles sont représentées en simple précision comme suite :

$+0 = (0|0000\,0000|\overbrace{000\cdots0}^{=0})_2$ $\qquad$ $-0 = (1|0000\,0000|\overbrace{000\cdots0}^{=0})_2$

$+\infty = (0|1111\,1111|\overbrace{000\cdots0}^{=0})_2$ $\qquad$ $-\infty = (1|1111\,1111|\overbrace{000\cdots0}^{=0})_2$

$NaN = (0/1|1111\,1111|\overbrace{010\cdots0}^{\neq 0})_2$

$Remarque$: NaN c'est *Not-a-Number*, c'est produit lorsque il y a ∞/∞ ou $0/0$...etc.