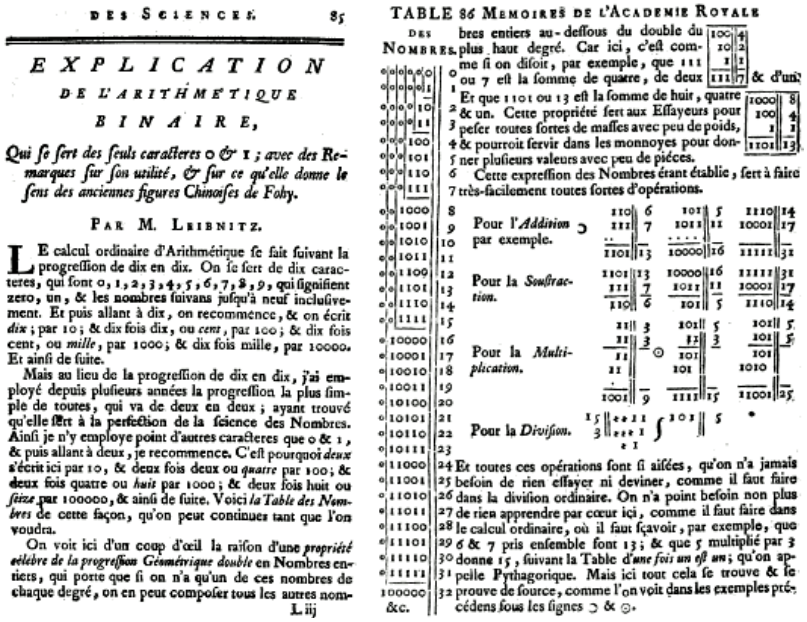
CODAGE BINAIRE DES NOMBRES

Gottfried Leibniz[1] (1646 – 1716)



DES SCIENCES. 85

EXPLICATION
DE L'ARITHMÉTIQUE
BINAIRE,

Qui ſe ſert des ſeuls caracteres 0 *&* 1 *; avec des Remarques ſur ſon utilité, & ſur ce qu'elle donne le ſens des anciennes figures Chinoiſes de Fohy.*

PAR M. LEIBNITZ.

LE calcul ordinaire d'Arithmétique ſe fait ſuivant la progreſſion de dix en dix. On ſe ſert de dix caracteres, qui ſont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, qui ſignifient zero, un, & les nombres ſuivans juſqu'à neuf inclusivement. Et puis allant à dix, on recommence, & on écrit *dix* ; par 10 ; & dix fois dix, ou *cent*, par 100 ; & dix fois cent, ou *mille*, par 1000 ; & dix fois mille, par 10000. Et ainſi de ſuite.

Mais au lieu de la progreſſion de dix en dix, j'ai employé depuis pluſieurs années la progreſſion la plus ſimple de toutes, qui va de deux en deux ; ayant trouvé qu'elle ſert à la perfection de la ſcience des Nombres. Ainſi je n'y employe point d'autres caracteres que 0 & 1, & puis allant à deux, je recommence. C'eſt pourquoi *deux* s'écrit ici par 10, & deux fois deux ou *quatre* par 100 ; & deux fois quatre ou *huit* par 1000 ; & deux fois huit ou *ſeize* par 10000, & ainſi de ſuite. Voici *la Table des Nombres* de cette façon, qu'on peut continuer tant que l'on voudra.

On voit ici d'un coup d'œil la raiſon d'une *propriété célebre de la progreſſion Géométrique double* en Nombres entiers, qui porte que ſi on n'a qu'un de ces nombres de chaque degré, on en peut compoſer tous les autres nom-

L iij

TABLE 86 MEMOIRES DE L'ACADEMIE ROYALE

DES NOMBRES.

bres entiers au-deſſous du double du plus haut degré. Car ici, c'eſt comme ſi on diſoit, par exemple, que 111 ou 7 eſt la ſomme de quatre, de deux & d'un. Et que 1101 ou 13 eſt la ſomme de huit, quatre & un. Cette propriété ſert aux Eſſayeurs pour peſer toutes ſortes de maſſes avec peu de poids, & pourroit ſervir dans les monnoyes pour donner pluſieurs valeurs avec peu de piéces.

Cette expreſſion des Nombres étant établie, ſert à faire très-facilement toutes ſortes d'opérations.

Pour l'*Addition* par exemple.

Pour la *Souſtraction*.

Pour la *Multiplication*.

Pour la *Diviſion*.

Et toutes ces opérations ſont ſi aiſées, qu'on n'a jamais beſoin de rien eſſayer ni deviner, comme il faut faire dans la diviſion ordinaire. On n'a point beſoin non plus de rien apprendre par cœur ici, comme il faut faire dans le calcul ordinaire, où il faut ſçavoir, par exemple, que 6 & 7 pris enſemble font 13 ; & que 5 multiplié par 3 donne 15, ſuivant la Table d'*une fois un eſt un* ; qu'on appelle Pythagorique. Mais ici tout cela ſe trouve & ſe prouve de ſource, comme l'on voit dans les exemples précédens ſous les ſignes ☽ & ☉.

&c.

« Explication de l'arithmétique binaire.
Qui se sert des seuls caractères 0 et 1 avec des remarques sur son utilité et sur ce qu'elle donne le sens des anciennes figures chinoises de Fohy » (Leibniz 1703)

Pré-requis pour prendre un bon départ :	☹	😐	☺	☺☺
Puissances de 2 : 2^0 = ….. et non ….. ! 2^1 = ….. 2^3 = ….. 2^{-1} = ….				
Division euclidienne (entière) : dividende = diviseur × quotient + reste.				
Différence entre les chiffres et les nombres.				
Ecrire un algorithme dans ses grandes lignes puis en pseudo-code.				
Calcul avec les fractions !				
Ecriture scientifique des nombres : $a \times 10^n$ avec $1 \leq a < 10$.				

[1] Gottfried Leibniz (1646 – 1716) : Immense philosophe et mathématicien allemand, contemporain de Isaac Newton avec qui il polémiqua.
1679 : écrit « De progressione dyadica », publié en 1966 sous le titre « Herr von Leibniz, Rechnung mit Null und Eins » (calcul avec des 0 et 1).
1703 : il est le premier à théoriser l'arithmétique binaire dans son mémoire remis à l'Académie des Sciences de Paris. Leibniz y concluait qu'il ne parvenait pas à voir l'utilité de ce mode de calcul, sinon une beauté essentielle : celle des liens qui unissent les nombres.
Et pourtant, près de 250 ans avant l'apparition de l'Informatique, Leibniz posait sans le savoir le fondement du fonctionnement des ordinateurs.

NOM et prénom : …… *Première spécialité NSI*

I. NUMERATION DECIMALE VS NUMERATION BINAIRE.

➢ Nous écrivons naturellement les nombres en base …….. à l'aide des 10 signes 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 qui s'appellent les ……

La numération décimale (le système d'écriture des nombres en base 10) s'appuie sur le tableau de numération suivant qui indique la valeur de chaque chiffre suivant sa position dans l'écriture du nombre :

Valeurs et noms des chiffres	*etc.*	10^4 = 10 000 Dizaine de milliers	10^3 = 1 000 Milliers	10^2 = 100 Centaines	10^1 = 10 Dizaines	10^0 = 1 Unités	10^{-1} = 1/10 Dixièmes	10^{-2} = 1/100 Centièmes	*etc.*
Chiffres	0	0	0	3	0	6	2	0	0

Exemple : Pour calculer la valeur d'un nombre qui a pour chiffre des centaines 3, pour chiffre des unités 6 et pour chiffre des dixièmes 2, on fait : $3 \times 10^2 + 6 \times 10^0 + 2 \times 10^{-1} = 3 \times 100 + 6 \times 1 + 2 \times 0{,}1 = 306{,}2$.

➢ Qu'en est-il en binaire ?

Les nombres binaires sont écrits en base …… à l'aide des 2 signes : ……. et ……. qui ne s'appellent pas les chiffres mais les ……

La numération binaire (le système d'écriture des nombres en base 2) s'appuie sur le tableau de numération binaire suivant qui indique la valeur de chaque bit suivant sa position dans l'écriture binaire du nombre :

Valeurs et noms des bits	*etc.*	2^4 = …… Seizaines	2^3 = …… Huitaines	2^2 = ….. Quatraines	2^1 = ….. Deuxaines	2^0 = ….. Unités	2^{-1} = 1/2 Demis	2^{-2} = 1/4 Quarts	*etc.*
Bits	0	0	0	1	0	1	1	0	0

Exemple : Pour calculer la valeur d'un nombre qui a pour bit des quatraines 1, pour bit des unités 1 et pour bit des demis 1, on additionne des puissances de 2 : $1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-1} = 1 \times 4 + 1 \times 1 + 1 \times 0{,}5 = 5{,}5$.

➢ Question qui nous brûle les lèvres : comment passe-t-on d'une écriture à l'autre ? Pour y répondre, plaçons-nous dans le cas des nombres les plus simples : les …….

II. ECRITURE BINAIRE EXACTE DES NOMBRES ENTIERS.

Rappel : **Le codage binaire est l'ensemble des algorithmes permettant d'encoder décoder en binaire.**

Théorème mathématique : **Tout nombre entier admet une écriture binaire unique et exacte.**

A. Base 2 → Base 10 : Calcul de la valeur d'un nb binaire (décodage).

Calcul de la valeur d'un entier écrit en binaire par somme des puissances de 2 : exemple.

$1001 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ On décompose selon les puissances décroissantes de 2.

$= 8 + 0 + 0 + 1$ On calcule comme d'habitude en n'oubliant pas que $\mathbf{2^0 = 1}$!

$= 9$

$(1001)_2 = (9)_{10}$ Dit autrement l'écriture 1001 en base 2 vaut 9 en base 10.

➢ Décodage Base 2 → Base 10 : Calcul de la valeur d'un nombre binaire par somme de puissances de 2.

2^n	256	128	64	32	16	8	4	2	1
bits									
bits									

2^n									
bits									
bits									

❶ A l'aide des tableaux de numération binaire au-dessus, calculer la valeur des nombres binaires suivants :

$(1011)_2$ = *R = 11*

$(101101)_2$ = *R = 45*

$(1100\ 1010)_2$ = *R = 202*

$(1111\ 1111)_2$ = *R = 255*

Vérifier vos résultats avec votre calculette ou celle de l'ordinateur, **en mode programmeur.**

❷ Comment reconnaît-on l'écriture binaire d'un entier pair ? D'un entier impair ?

❸ Rajout d'un chiffre à droite de l'écriture d'un nombre binaire :

• En écriture décimale, rajouter un 0 à droite de l'écriture d'un entier revient à multiplier sa valeur par .....

En écriture binaire, rajouter un 0 à droite de l'écriture revient à multiplier la valeur par ……

En écriture binaire, rajouter 00 à droite de l'écriture revient à multiplier la valeur par …….. etc.

• En écriture binaire, rajouter un 1 à droite de l'écriture revient à ……..

❹ Soit une écriture binaire. Ecrire un algorithme qui en entrée reçoit cette écriture binaire et calcule petit à petit sa valeur. En sortie, valeur finale de cette écriture binaire.

Grandes étapes de l'algorithme

- Entrée et Initialisations.
- Répéter : (combien de fois ?)

  Calculer le bit × puissance de 2 correspondante (dans quel ordre des bits ?) et l'ajouter à la valeur partielle.
- Renvoyer la valeur finale.
- Validation : Quels cas faut-il tester ?

Algorithme en pseudo-code

-

❺ Ecrire en Python une fonction décodage_binaire( ) avec en paramètre d'entrée une écriture binaire (type ?) et en sortie sa valeur (type ?).

B. Plage (intervalle) d'entiers représentables avec n bits :

Après le décodage (conversion base 2 → base 10) et avant l'encodage (conversion base 10 → base 2), il faut savoir quel intervalle d'entiers est représentable pour une écriture binaire avec un nb de bits donné.

1. Exemples :

1. Compléter ce tableau :

	avec 2 bits.		*avec 3 bits.*	
	Ecriture binaire	*Valeur*	*Ecriture binaire*	*Valeur*
Plus petit nombre binaire				
Plus grand nombre binaire				

2. Une machine traite les nombres entiers positifs sur 2 bits puis sur 3 bits, puis sur 4 bits.
   Ecrire alors dans chaque cas la plage de nombres entiers manipulables :

Codage	*sur 2 bits*	*sur 3 bits*	*sur 4 bits*
Plage d'entiers représentables	entiers de 0 à 3 noté ⟦ 0 ; 3 ⟧	entiers de 0 à ….. noté ⟦ 0 ; … … ⟧	entiers de …… à …… noté

2. Plage (intervalle) d'entiers représentables avec n bits : formule.

Sur n bits, l'intervalle d'entiers représentables est ⟦… … … ; … … … … … … ⟧.

Preuve (Niveau 1ère Spé Maths) :

- Le plus petit entier qu'on peut écrire avec n bits est évidemment $(\overbrace{00etc00}^{\text{n bits}})_2$ qui vaut bien sûr ……
- Le plus grand entier qu'on peut écrire avec n bits est évidemment (……)$_2$.

Grâce à la somme des n termes d'une suite géométrique de raison 2, on montre que $(\underbrace{11etc11}_{\text{n bits}})_2 = 2^n - 1$.

3. Application :

Les processeurs manipulent les écritures binaires selon une taille en bits standard.

Plus cette taille est longue, plus la plage d'entiers représentables est grande et donc plus le processeur est puissant car les données qu'il peut traiter à chaque cycle d'horloge peuvent être plus grandes.

Ecrire la plage d'entiers manipulables pour les processeurs historiques suivants :

Processeurs historiques	*1972 : Intel 8008* *1974 : Motorola 6800*	*1973 : National Imp-16* *1976 : DEC LSI-11* *1978 : Intel 8088*	*1979 : National 32016* *1981 : HP Focus* *1986 : Intel 80386*	*1991 : MIPS R4000* *1997 : IBM Power3* *2006 : Intel Core2*
Taille des nbs binaires	*8 bits*	*16 bits*	*32 bits*	*64 bits*
Plage d'entiers représentables	⟦ 0 ; $2^8 - 1$ ⟧	⟦ …. ; …… ⟧		

C. Base 10 → Base 2 : encodage binaire des entiers naturels.

On veut trouver par exemple l'écriture binaire de l'entier naturel 52.

Il s'agit donc de trouver les facteurs devant chaque puissance de 2 dans la décomposition de 52 en base 2 :

$$52 = \text{etc.} + c \times 2^2 + b \times 2^1 + a \times 2^0 \text{ (lire de droite à gauche)}$$

Il s'agit donc de trouver son bit « a » des unités, son bit « b » des deuxaines, son bit « c » des quatraines etc.

Pour trouver tous ces chiffres binaires, 2 algorithmes (2 méthodes) principalement existent :

1. Encodage par soustractions successives.

C'est l'algorithme le plus intuitif pour trouver les chiffres binaires. Il nécessite juste de connaître les puissances de 2 et de savoir calculer des restes par soustraction. Il peut donc être mis en œuvre dès la 4ème.

❶ Exemples : Grâce aux tableaux de numération binaire, trouver les écritures binaires de 52 puis 29.

2^n	2^6 =……	2^5 =……	2^4 =……	2^3 =……	2^2 =……	2^1 =……	2^0 =……
bits							
reste							

$(52)_{10}$ = (……..……..)$_2$

2^n							
bits							
reste							

$(29)_{10}$ = (……)$_2$

Dans quel ordre obtient-t-on les bits ? On obtient les bits de la …… vers la …….. donc du bit de poids le plus …… au bit de poids le plus …….

❷ **Algorithme par soustractions successives**

Grandes étapes de l'algorithme

- Entrée(s) et Initialisation(s).
- Plus grande puissance de 2 contenue ds le nb ?
  En déduire le nombre de bits.
- Ecriture du nombre binaire :

• Si la puissance de 2 est dans le nombre, écrire '1' (à gauche, à droite ?) puis calculer le reste.
  Sinon écrire '0' (à gauche, à droite ?).
• Et ainsi de suite → Boucle For ou While ? Combien de tours de boucle ? ……

- Renvoyer le résultat.

Tests : Quels entiers faudra-t-il tester et combien ?

Algorithme en pseudo-code

- Nb_base10 ← entrée (type ……...) et exp ← ...
- Tant que 2^{exp} …..…. Nb_base10 faire :
  exp ← …… + ……
  plus_grand_exp ← ……..
  nb_bits ← ……
- Reste ← Nb_base10 ; Nb_binaire ← ' '

Pour k allant de 0 à ……..., faire :
  Si ……...…… Reste :
    Nb_binaire ← …….. + ……
    Reste ← ……..
  Sinon Nb_binaire ← ……..…… + ……

- Renvoyer …….

Cet algorithme « naïf » (intuitif) peut être très largement amélioré. On s'en contentera pour l'instant.

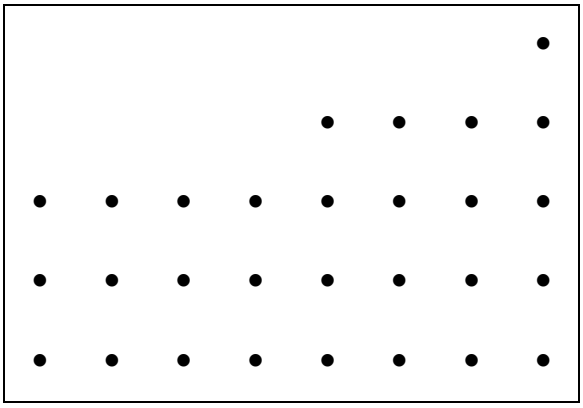
❸ Ecrire en Python une fonction encodage_par_soustractions( ) (paramètre d'entrée : un entier ; corps : algorithme par soustractions successives ; sortie : écriture binaire de cet entier).

Ne pas oublier de traiter le cas de 0 !

2. Encodage par divisions entières par 2 successives.

• Voici un autre algorithme qui s'appuie sur le fondement de la numération : le regroupement par paquets.

On veut (re)trouver l'écriture binaire de 29 par exemple en utilisant des regroupements par paquets de 2 :



Regrouper par paquets de 2. Reste …… unité.

Regrouper par paquets de 4. Reste …… paquet de 2.

Regrouper par paquets de 8. Reste ……. paquet de 4.

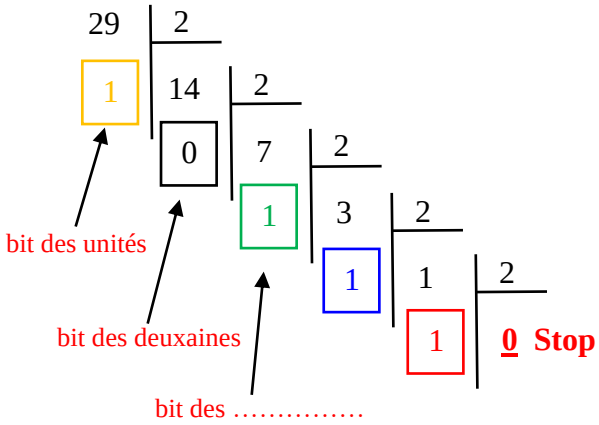
Regrouper par paquets de 16. Reste ……. paquet de 8.

Paquets de 32 possible ? …….. Reste ……. paquet de 16.

Arrêt de l'algorithme. → Sortie : $(29)_{10}$ = (……)$_2$

• En pratique, impossible de dessiner ces regroupements pour de grands nombres ! Regrouper successivement par paquets revient en fait mathématiquement à effectuer des divisions euclidiennes s'enchaînant à la suite.

Concrètement voici ce que cet algorithme par divisions entières par 2 successives donne pour le nombre 29 :



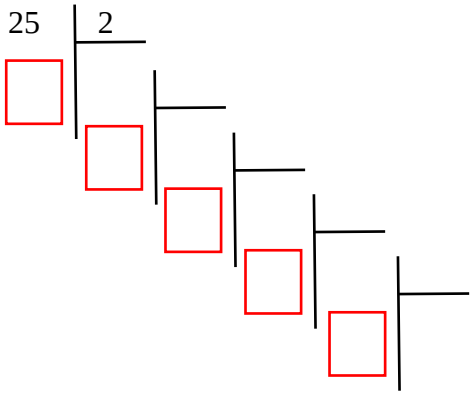
⇨ $(29)_{10}$ = $(0001\ 1101)_2$ sur 8 bits

Attention ! Ecriture des restes de droite à gauche !

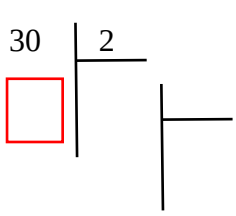
• La méthode apparaît donc comme une suite de …….... par …… successives :

- qui va-t-on diviser successivement ? Les ……..
- par combien divisera-t-on successivement ces quotients ? Toujours par …….
- quand arrête-t-on le processus ? ……
- les bits cherchés sont les ……
- ordre de sortie des bits ? ……

❶ Par cette méthode par divisions successives par 2, trouver les écritures binaires sur 8 bits de 25 puis 30 :



⇨ $(25)_{10}$ =



⇨

❷ **Algorithme par divisions entières par 2 successives**

Grandes Lignes de l'algorithme

- Entrée(s) et Initialisation(s).
- Corps de l'algorithme :

> • Calculer le nouveau quotient et le reste (*dans quel ordre ?*)
> • Accoler le reste (*à gauche, à droite ?*) à la chaîne de caractères donnant le nombre binaire.

et ainsi de suite jusqu'à ce que le quotient valle 0.

- Sortie de l'algorithme :

Validation : Quels entiers tester et combien ?

Algorithme en pseudo-code

- Nb_10 ← entrée (type ……..)

…….

…….

- Tant que …… > ……, faire :

  …….. ← ……..

  …….. ← ……..

  Nb_binaire ← ……..….. + ……

- Renvoyer ……..……..

Que se passe-t-il si on met ≥ au lieu de > dans le Tant que ?

❸ Ecrire en Python une fonction encodage_par_divisions( ) (paramètre d'entrée : un entier ; corps : algorithme par divisions entières par 2 successives ; sortie : écriture binaire de cet entier).

Ne pas oublier de traiter le cas de 0 !

3. Comparaison des 2 algorithmes d'encodage binaire :

	Par soustractions successives (ou questions successives)	Par divisions successives
Avantages		
Inconvénients		
Ordre de sortie des chiffres		
Complexité (rapidité) de l'algorithme		

Après l'encodage des entiers naturels, passons à l'encodage des entiers .....

III. ENCODAGE DES NOMBRES ENTIERS RELATIFS SIGNES.

Un entier relatif (exemple : +5 ou -3) a un signe + ou – devant sa valeur absolue (sa partie chiffrée).

En binaire, on ne peut pas ajouter bêtement un signe + ou un signe – devant les bits. En effet, une écriture binaire n'est composée que de 0 et de 1 et rien d'autre !

Comment alors écrire en binaire un entier signé (un entier avec un signe) ? Voyons d'abord cela sur 3 bits.

A. Exemple simple : écriture binaire sur 3 bits des entiers signés.

1. Entiers non signés écrits sur 3 bits :

➢ On a déjà vu précédemment p.4 que sur 3 bits, on ne peut écrire en binaire que $2^{\cdots}$ (= …..) entiers non signés (sans signe).

Cette plage d'entiers représentables sur 3 bits va de …… à ……

Le tableau ci-contre répertorie ces 8 entiers non signés et l'écriture binaire de chacun. Compléter les écritures binaires manquantes.

Par quel bit commence l'écriture binaire des 4 premiers entiers ? …..

Par quel bit commence l'écriture binaire des 4 entiers suivants ? …..

	Nombre	Ecriture binaire sur 3 bits
	8	$_1$000
8 entiers représentables sur 3 bits.	*7*	
	6	
	5	
	4	
	3	011
	2	010
	1	001
	0	000

2. Entiers signés écrits sur 3 bits :

➢ Maintenant, on veut coder sur 3 bits des entiers relatifs *signés*. On ne pourra toujours en coder que 8 évidemment ! Mais lesquels choisir ? Lister ces 8 entiers signés qu'on va encoder en binaire (évidemment, par symétrie, il faut « autant » d'entiers positifs que négatifs !) : ……..……...

Donc les entiers signés codés sur 3 bits iront de –…….. à +…….

➢ Reporter dans l'encadré ci-contre ces 8 entiers signés dans la colonne nombre. Il s'agit maintenant pour chacun de ces 8 entiers signés de choisir une écriture binaire parmi les 8 écritures binaires sur 3 bits possibles du premier tableau au-dessus :

- pour les positifs : le choix est tout naturel ! Compléter.
- pour les négatifs : quels choix proposez-vous ?

	Nombre	Ecriture binaire sur 3 bits
8 entiers **signés** représentables sur 3 bits		

Quel est le point commun de toutes les écritures binaires des positifs ? …….....

Quel est le point commun de toutes les écritures binaires des négatifs ? …….....

➢ Maintenant, nous allons généraliser ces résultats non plus sur 3 bits mais sur n bits.

B. Ecriture binaire sur n bits des entiers signés :

1. Tableau d'écriture binaire sur n bits des entiers signés :

➢ On a déjà vu p.4 qu'avec n bits, on ne peut écrire en binaire qu'un nombre total de ….⋯ nombres.

Par principe de symétrie, on veut « autant » d'entiers signés positifs que d'entiers signés négatifs.

➢ Donc la moitié de ces 2^n nombres (soit $\frac{2^n}{2} = 2^{\cdots\cdots\cdots\cdots}$)

seront les entiers signés positifs : de …..... à 2^{n-1} **– 1**.

L'écriture binaire de ces 2^{n-1} entiers positifs commence toujours par le bit …….

Sur n bits, la plus petite écriture binaire commençant par un 0 est …….. qui vaut évidemment ……

n bits

Sur n bits, la plus grande écriture binaire commençant par un 0 est …….. qui vaut le nombre +……

n bits

➢ L'autre moitié de ces 2^n nombres (soit ……..) seront donc les entiers signés négatifs : de …..…. à -2^{n-1}.

L'écriture binaire de ces 2^{n-1} entiers négatifs commence toujours par le bit …….

Nombre	Ecriture binaire sur n bits	
$2^n - 1$	1111..1111	*Les 2^{n-1} entiers suivants*
etc.	*etc.*	
2^{n-1}	……	
…….	…….	2^{n-1} entiers positifs soit la moitié de 2^n.
etc.	etc.	
0	…….	
-1	……	……. *entiers négatifs soit la moitié de 2^n.*
etc.	*etc.*	
……	……	

2^n entiers **signés** représentables sur n bits

Sur n bits, la plus petite écriture binaire commençant par un 1 est …….. qui correspondra au plus petit entier négatif représentable : –……

n bits

Sur n bits, la plus grande écriture binaire commençant par un 1 est …….. qui correspondra au plus grand entier négatif représentable : ……

n bits

➢ Application : Compléter les 2 tableaux d'écriture binaire des entiers signés sur 4 puis 8 bits :

Nombre	Ecriture binaire sur 4 bits	
….	……	….. entiers positifs soit la moitié de 2^4.
etc.	etc.	
0	0000	
-1	……	……. *entiers négatifs soit la moitié de 2^4.*
etc.	*etc.*	
….	……	

….. entiers **signés** représentables sur 4 bits

Nombre	Ecriture binaire sur 8 bits	
……..	……	….. entiers positifs soit la moitié de 2^8.
etc.	etc.	
……	……	
……	……	……. *entiers négatifs soit la moitié de 2^8.*
etc.	*etc.*	
……	……	

….. entiers **signés** représentables sur 8 bits

2. Intervalles d'entiers (non signés, signés) représentables sur n bits :

Grâce à ce qui précède, on peut résumer les plages d'entiers (non signés, signés) représentables, de la plus petite valeur à la plus grande valeur :

Processeur	*4 bits*	*8 bits*	*n bits*
Plage d'entiers ***sans signe*** *représentables*	$[\![0 ; 2^4 - 1]\!]$ c-à-d de 0 à 15.	$[\![0 ; 2^8 - 1]\!]$ c-à-d de 0 à ……	$[\![$ ……. ; …….. $]\!]$
Plage d'entiers relatifs ***signés*** *représentables*	$[\![-2^{4-1} ; +2^{4-1} - 1]\!]$ de ……... à …..…	$[\![-2^{8-1} ; +2^{8-1} - 1]\!]$ de ……. à ……..…..	

C. Ecriture binaire des entiers signés positifs :

Leurs écritures binaires commencent par 0 et sont les mêmes que celles des entiers naturels sans signe. Pour trouver l'écriture binaire d'un entier positif, il suffit donc d'appliquer soit l'algorithme par soustractions successives p.5, soit l'algorithme par divisions successives p.6.

D. Ecriture binaire des entiers signés négatifs :

1. Une fausse bonne idée :

On sait que l'écriture binaire des entiers négatifs commence toujours par ….. pour indiquer le signe ….

- En décimal, l'écriture de l'opposé est simple : il faut juste changer le signe. N'aurait-il pas alors été plus simple d'écrire l'opposé en binaire de la même façon : juste en changeant le premier bit ?

Voyons cela sur 3 bits :

+2 s'écrit **0**10 donc –2 s'écrirait alors **1**10. +1 s'écrit **0**01 donc –1 s'écrirait alors **1**01

- La moindre des choses est que la somme de 2 nombres opposés fasse 0 ! Vérifions donc si la somme en binaire de ces deux paires de nombres opposés donne bien 000 (c-à-d 0).

Les additions posées en binaire se calculent de la même façon qu'en décimal. Sauf pour les retenues qui apparaissent dès qu'on obtient 2. On pose alors 0 et on retient 1 en haut de la colonne gauche suivante.

Si une retenue déborde tout à gauche de l'écriture, elle est tout simplement ignorée !

0 1 0	(+2)
+ 1 1 0	(–2)
1 0 0 0	

0 0 1	(+1)
+ 1 0 1	(–1)
1 1 0	

On a bien +2 + (–2) qui vaut 000 (la petite retenue 1 étant tout simple ignorée car elle sort tout à gauche de l'écriture sur 3 bits) mais hélas, +1 + (–1) ne vaut pas 000 !

- **Donc pour l'encodage des entiers négatifs, inverser seulement le bit de signe fait perdre la structure additive des nombres ce qui est très très très embêtant ! On oublie donc vite cette idée !**

Mais alors comment fait-on pour trouver l'écriture binaire d'un entier négatif ? 😕

2. Méthode algébrique (mathématique) : Complémentaire à 2^n.

Pour simplifier la suite, replaçons-nous sur 3 bits avec notre tableau d'écriture p.8.

➢ On sait que les écritures binaires de -4 , -3 , -2 et -1 sont celles copiées collées de respectivement 4 , 5 , 6 , 7.

Dit autrement, l'écriture binaire de l'opposé de 1 est celle de …..

l'écriture binaire de l'opposé de 2 est celle de …..

l'écriture binaire de l'opposé de 3 est celle de …..

l'écriture binaire de l'opposé de 4 est celle de …..

Que remarquez-vous sur les couples (1 , 7), (2 , 6), (3 , 5) et (4 , 4) ?

	Nombre	Ecriture binaire sur 3 bits
	8	$_1$000
	7	*111*
	6	*110*
	5	*101*
	4	*100*
8 entiers **signés** représentables sur 3 bits	3	011
	2	010
	1	001
	0	000
	-1	*111*
	-2	*110*
	-3	*101*
	-4	*100*

➢ On peut donc affirmer : (avec k un entier positif)

« Sur 3 bits, l'opposé d'un nombre k a la même écriture binaire que le nombre qu'il faut ajouter à k pour retrouver 8 (= 2^3). »

Dit autrement : « Sur 3 bits, l'opposé d'un nombre k a la même écriture binaire que le complémentaire à 8 (= 2^3) de k. »

Plus généralement : **Soient n le nombre de bits et k un entier dans $2^{n-1} - 1$:**

Sur n bits, l'opposé de ce nombre k a la même écriture binaire que le complémentaire à 2^n de k.

Exemple : Quelle est l'écriture binaire sur 4 bits de -7 ?

-7 est l'opposé de 7. Le complémentaire à 2^4 (= 16) de 7 est 9. L'écriture binaire de 9 sur 4 bits est 1001.

Donc l'écriture binaire sur 4 bits de -7 est la même que celle de 9 le complémentaire à 2^4 de 7 : 1001.

➢ Application :

Complémenter le tableau suivant pour trouver les écritures binaires des nombres négatifs indiqués :

*Trouver l'écriture binaire d'un entier signé négatif : **méthode mathématique par complémentation à 2^n.***				
Entier négatif	-7 (sur 4 bits)	-1 (sur 4 bits)	-10 (sur 5 bits)	-16 (sur 5 bits)
Entier positif associé	7			
Complémentaire à 2^n	9 *En effet, 7 + 9 = 16 (2^4)*			
Ecriture binaire du complémentaire à 2^n (donc de l'entier négatif)	1001			

➢ C'est bien joli tout ça, mais est-on bien certain que lorsqu'on ajoute les écritures binaires d'un nombre et de son opposé (donc du complémentaire à 2^n), on trouve bien l'écriture binaire de 0 ?

La suite va nous le prouver.

3. Méthode informatique (binaire) : Inversé bit à bit + 1.

➢ La question qui se pose est donc la suivante : « Quand on fait la somme d'un nombre binaire et de son opposé binaire, comment être sûr d'obtenir à chaque fois 000, ou ce qui revient au même $_1$000 sur 3 bits (le petit $_1$ en retenue qui déborde tout à gauche sur le 4ème bit ne sera tout simplement pas pris en compte car l'écriture est sur 3 bits seulement ! Donc sur 3 bits, $_1$000 = 000 !) ?

• On sait en tous cas à coup sûr obtenir 111 ! En effet, à partir de l'écriture sur 3 bits d'un entier positif, si on lui ajoute son écriture inversée bit à bit (chaque 0 est changé en 1 et vice versa), on obtient forcément 111 !

0 1 0 (+2)	0 0 1 (+1)
+ 1 0 1 inversé bit à bit	+ 1 1 0 inversé bit à bit
1 1 1	1 1 1

Exemples : En repartant des écritures binaires de (+2) et de (+1), quand on rajoute les « inversés bit à bit », on obtient bien 111.

• Maintenant, pour obtenir 000, il suffit de rajouter 1 ! En effet 111 + 001 donne $_1$000 soit 000 !

Au final, l'opposé binaire s'obtient donc comme « l'inversé bit à bit » + 1. Vérifions sur nos 2 exemples :

0 1 0 (+2)	0 0 1 (+1)
+ 1 1 0 inversé bit à bit + 1	+ 1 1 1 inversé bit à bit + 1
$_1$0 0 0	$_1$0 0 0

L'opposé de 010 s'écrit donc 110.

L'opposé de 001 s'écrit donc 111.

➢ Il ne reste plus qu'à vérifier que cet opposé binaire sur 3 bits correspond bien au complémentaire à 2^3.

Sur 3 bits, la méthode de « l'inversé bit à bit + 1 » permet donc de trouver l'écriture binaire sur 3 bits de l'opposé c d'un nombre k. On a donc :

$$(k)_2 + (c)_2 = (000)_2.$$

Plus exactement on a $(k)_2 + (c)_2 = (_1000)_2$ Or $(_1000)_2$ est l'écriture binaire de 8 c-à-d 2^3.

D'où $(k)_{10} + (c)_{10} = 2^3$

L'opposé « c » est le nombre qu'il faut rajouter à k pour retrouver 2^3 sur 3 bits (2^n plus généralement sur n bits). L'opposé « c » est donc bien le complémentaire de k à 2^3 sur 3 bits (à 2^n plus généralement sur n bits).

On vient donc de prouver la chose suivante :

L'écriture binaire du complémentaire à 2^n est aussi donnée par « l'inversé bit à bit + 1 ».

➢ On a donc une 2ème méthode (informatique) pour trouver l'écriture binaire d'un entier signé négatif :

*Trouver l'écriture binaire d'un entier signé négatif : **méthode binaire (inversé bit à bit + 1).***				
Nombre négatif	-7 (sur 4 bits)	-1 (sur 4 bits)	-10 (sur 5 bits)	-16 (sur 5 bits)
Nombre positif Ecriture binaire	7 0111			
Inversé bit à bit	1000			
Inversé bit à bit + 1	1001			

Dans d'autres cours, cette méthode est appelée à tort « méthode du complémentaire à 2 » au lieu de « Méthode binaire pour trouver l'écriture binaire du complémentaire à 2^n » !

E. Exercices sur les entiers signés :

❶ Additions binaires posées :

Les additions posées en binaire se calculent comme d'habitude. Sauf pour les retenues qui apparaissent dès qu'on obtient 2 (on pose alors 0 et on retient 1 à la colonne de gauche) ou 3 (on pose 1 et on retient 1).

Poser puis calculer les 3 sommes binaires suivantes : 101 + 011 110110 + 100111 11011 + 110

```
  1 0 1
+ 0 1 1
-------
```

❷ Addition bit à bit : table des résultats possibles (table de vérité).

L'exercice précédent montre qu'additionner 2 nombres binaires revient à enchaîner plusieurs additions bit à bit (chiffre à chiffre) en tenant compte d'une éventuelle retenue.

1. Le tableau ci-dessous indique **en colonne** tous les résultats possibles pour l'addition bit à bit en fonction des valeurs possibles pour la retenue, le bit du premier nombre et le bit du second nombre. Le compléter.

(retenue : …… choix ; bit1 : ….. choix ; bit2 : ….. choix) ⇒ ….. choix pour le triplet (retenue, bit1, bit2)

retenue	0							
bit1	0							
bit2	0							
bit somme	0							
retenue	0							

2. Ecrire le dictionnaire Additionneur_bit_a_bit correspondant à ce tableau, rempli de STR ou d'INT ?

❸ Complémentaire du complémentaire à 2^n.

L'opposé de l'opposé d'un nombre redonne ce nb. Qu'en est-il pour le complémentaire du complémentaire ?

Soit donc par exemple l'entier binaire *signé* $(0101\ 1110)_2$ écrit sur 8 bits :

1. Plage d'entiers *signés* manipulables sur 8 bits ? De – …….. à + …….

2. Quel est le signe du nb binaire précédent ? Justifier :

2^n	…	…	…	…	…	*4*	*2*	*1*
bits								

3. Calculer sa valeur :

4. Donner l'écriture binaire de l'opposé de ce nombre (méthode binaire) :

• Inversé bit à bit =

• Inversé bit à bit + 1 =

• Phrase réponse : L'écriture binaire de …… est :

5. Trouver l'inversé bit à bit + 1 du nombre trouvé à la question 4. Commenter.

•

•

• On retrouve

6. (Prouver que le complémentaire à 2^n du complémentaire à 2^n d'un nombre k redonne bien k !)

Aide : Utiliser la définition mathématique du complémentaire à 2^n.

❹ Ecriture binaire → entier signé.

Un processeur travaillant sur 5 bits a calculé l'écriture binaire d'un entier ***signé*** et a trouvé 1 0110.

2^n	...	...	...	...	...	...	*2*	*1*
bits								
bits								

1. Quelle est la plage d'entiers ***signés*** manipulables par le processeur ? De ……..…….... à ……..
2. Signe de l'entier signé dont l'écriture binaire est 10110 ? Justifier :

3. Lou Phoque dit que 1 0110 est l'écriture binaire de-6 ! Comment a-t-elle trouvé ce -6 ? A-t-elle raison ?

4. 10110 est l'écriture binaire de quel entier signé ? (2 méthodes : binaire puis algébrique)

5. Un entier *signé* a pour écriture sur 5 bits 1 0011. Cet entier est-il 19 ou -19 ou 3 ou -3 ou 13 ou -13 ?

F. TP Calculatrice binaire.

Voir TP Calculatrice binaire sur mon site

IV. TABLEAU RECAPITULATIF SUR LES TYPES ENTIERS.

Sur n bits	*Entiers **non signés***	*Entiers **signés***	
Ecriture	• Pas de bit de signe. • n bits significatifs.	….. bit de signe : bit à …….....	
		Négatifs **Les n – 1 autres bits sont non significatifs !**	Positifs Les n – 1 autres bits sont significatifs.
Nombre total d'entiers représentables		Négatifs	Positifs
Intervalle d'entiers représentables			
Plus grand entier (écriture binaire)		Négatif	Positif
Plus petit entier (écriture binaire)		Négatif	Positif
Calculer la valeur d'un nb binaire : *Algorithme de décodage (base 2 → base 10)*		Négatifs (exo 4 p.14) • •	Positifs (exos p.3) •
Convertir un nombre en base 2 : *Algorithmes d'encodage (base 10 → base 2)*	• •	Négatifs (p.11 et 12) • •	Positifs (p. 5 à 7) • •

V. ECRITURE BINAIRE INEXACTE DES AUTRES TYPES DE NB.

A. Ecriture dyadique (binaire avec une virgule) :

Après les nombres entiers arrivent les nombres décimaux. Qu'en est-il en binaire ?

Pour cela, nous allons travailler par analogie entre la base 10 et la base 2 comme en début de livret p.2,

	En décimal (base …..)	En binaire (base ….)
Exemples	Ecriture « virgulaire » classique : $(6,25)_{10} = 6 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$ $= 6 \times 1 + 2 \times \frac{1}{10} + 5 \times \frac{1}{100}$ $= 6 + 0,2 + 0,05$ $= 6,25$	Ecriture « virgulaire » binaire : $(110,01)_2 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$ $= 1 \times 4 + 1 \times 2 + 0 \times 1 + 0 \times \frac{1}{2} + 1 \times \frac{1}{4}$ = …… + …… + …… + …….. + ……. = ……..
Définitions	• La décomposition en somme de puissances de 10 s'appelle le développement décimal. L'écriture virgulaire classique s'appelle l'écriture décimale. $7,8 = 7 \times 10^0 + 8 \times 10^{-1}$ Ecriture décimale / Développement décimal • Un nombre dont l'écriture décimale est finie s'appelle un nombre décimal. Ex : $\frac{1}{2}$ est un nombre décimal mais pas $\frac{1}{3}$!	• La décomposition en somme de puissances de 2 s'appelle le développement dyadique. L'écriture virgulaire binaire s'appelle l'écriture …… $(10,01)_2 = 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$ Ecriture dyadique / Développement dyadique • Un nombre dont l'écriture dyadique est finie s'appelle un nombre ……. Ex : ……..…… sont des nombres dyadiques.
Théorème	Tout nb possède une écriture décimale unique. Cette écriture décimale est soit finie (nbs décimaux) soit infinie (nbs non décimaux).	Tout nb possède une écriture dyadique unique. Cette écriture dyadique est soit finie (nbs …….) soit infinie (nbs non dyadiques).
Tout nombre dyadique est de la forme $\frac{a}{2^n}$, avec a > 0 et n entier Ex : $\frac{1}{4}$; $\frac{27}{256}$; 5		

➢ A l'aide du tableau de numération binaire, calculer :

$(1011,011)_2$ =

$(1001,11)_2$ =

2^n	…	…	2	**1**	1/2	1/4	…	…
bits								
bits								

B. Comment trouver l'écriture dyadique d'un nombre positif ?

Trêve de bla bla théorique, on va raisonner sur un exemple :

Soit le nombre positif 6,72. On veut trouver son écriture dyadique (écriture virgulaire binaire) :

- Partie entière 6 : là, no problem, on convertit en binaire comme à son habitude (méthodes p.5 ou 6).
- Partie non entière 0,72 : là, il s'agit de trouver le développement dyadique.

1. Méthode ① : Partie dyadique par questions-soustractions.

➢ L'algorithme qui suit est la continuité de l'algorithme par soustractions successives p.5.

- Peut-on mettre $2^{-1} = \frac{1}{2} = 0,5$ dans 0,72 ? Oui. Et on calcule le reste 0,72 – 0,5= 0,22.

  Peut-on mettre $2^{-2} = \frac{1}{4} = 0,25$ dans 0,22 ? Non. Rien.

  Peut-on mettre $2^{-3} = \frac{1}{8} = 0,125$ dans 0,22 ? Oui. Et on calcule le reste 0,22 – 0,125 = 0,095.

  Peut-on mettre $2^{-4} = \frac{1}{16} = 0,0625$ dans 0,095 ? Oui. Et on calcule le reste 0,095 – 0,0625 = 0,0325.

Et ainsi de suite…

D'où $(0,72)_{10} = \mathbf{0} \times 2^0 + \mathbf{1} \times 2^{-1} + \mathbf{0} \times 2^{-2} + \mathbf{1} \times 2^{-3} + \mathbf{1} \times 2^{-4} + \mathbf{etc.} = (\mathbf{0,1011\ etc.})_2$

- Cet algorithme peut-il se poursuivre indéfiniment ? …….. ⇒ nombre non dyadique !

  A quelle condition cet algorithme peut-il s'arrêter ? Lorsque ……

  L'écriture dyadique est alors finie ⇒ on a donc un nombre ……

- Finalement, sachant que $(6)_{10} = (110)_2$ et $(0,72)_{10} = (0,1011\ etc.)_2$, l'écriture binaire de 6,72 est :

  $(6,72)_{10} = (110,1011\ etc.)_2$, soit environ 0110,1011 avec partie entière et partie dyadique sur 4 bits.

➢ <u>Application :</u> La méthode ci-dessus se résume très facilement dans un tableau de numération dyadique.

Trouver sur 3 + 3 bits l'écriture dyadique approchée ou exacte des nombres 5,3 et $\frac{27}{8}$:

2^n	….	….	…	**1**	1/2	1/4	…..	…..
bits								
restes								

5,3 *R ≈ 101,010*

2^n		…	**1**					
bits								
restes								

$\frac{27}{8}$ *R = 011,011*

2. Méthode ② : Partie dyadique par multiplications par 2 successives.

Reprenons l'exemple plus haut de 6,72. Occupons-nous de la partie décimale 0,72 :

- *Savoir si $\frac{1}{2}$ est dans 0,72 revient à* ***comparer 1 et 0,72 × 2*** *n'est-ce pas ?*

Puisque 0,72 × 2 = 1,44 ≥ 1 donc $\frac{1}{2}$ est bien dans 0,72.

Calculons le reste = $0,72 - \frac{1}{2} = \frac{2 \times 0,72 - 1}{2} = \frac{0,44}{2}$

- *Savoir si $\frac{1}{4}$ est dans le reste $\frac{0,44}{2}$ revient à comparer 1 et $\frac{0,44}{2} \times 4$ c-à-d* ***comparer 1 et 0,44 × 2*** *:*

Puisque 0,44 × 2 = 0,88 < 1 donc $\frac{1}{4}$ n'est pas dans le reste $\frac{0,44}{2}$.

- *Savoir si $\frac{1}{8}$ est dans le reste $\frac{0,44}{2}$ revient à comparer 1 et $\frac{0,44}{2} \times 8$ c-à-d* ***comparer 1 et (0,44 × 2) × 2*** *:*

Puisque 0,44 × 4 = 1,76 ≥ 1 donc $\frac{1}{8}$ est bien dans le reste $\frac{0,44}{2}$.

Calculons le reste = $\frac{0,44}{2} - \frac{1}{8} = \frac{4 \times 0,44 - 1}{8} = \frac{0,76}{8}$

- *Et ainsi de suite : Savoir si $\frac{1}{16}$ est dans le reste $\frac{0,76}{8}$ revient à comparer 1 et $\frac{0,76}{8} \times 16$ c-à-d* ***comparer 1 et 0,76 × 2****.*

Puisque 0,76 × 2 = 1,52 ≥ 1 donc $\frac{1}{16}$ est bien dans le reste $\frac{0,76}{8}$. Etc.

➢ <u>En résumé :</u> On prend la partie décimale, on la multiplie par 2 et on compare avec 1. On obtient ainsi petit à petit les chiffres de la partie dyadique. Ainsi en partant de 0,72 :

- $0{,}72 \times 2 = 1{,}44$ — 1 sera donc le 1er chiffre après la virgule de l'écriture dyadique.

  $0{,}44 \times 2 = 0{,}88$ — 0 sera donc le 2ème chiffre après la virgule de l'écriture dyadique.

  $0{,}88 \times 2 = 1{,}76$ — 1 sera donc le 3ème chiffre après la virgule de l'écriture dyadique.

  $0{,}76 \times 2 = 1{,}52$ — 1 sera donc le 4ème chiffre après la virgule de l'écriture dyadique.

Et ainsi de suite…

D'où $(0{,}72)_{10} = \mathbf{0} \times 2^0 + \mathbf{1} \times 2^{-1} + \mathbf{0} \times 2^{-2} + \mathbf{1} \times 2^{-3} + \mathbf{1} \times 2^{-4} + \mathbf{etc.} = (\mathbf{0{,}1011\ etc.})_2$

- L'algorithme peut-il se poursuivre indéfiniment ? …….. ⇒ nombre non dyadique !

  A quelle condition cet algorithme peut-il s'arrêter ? Lorsque ……

  L'écriture dyadique est alors finie ⇒ nombre ……

- Finalement, sachant que $(6)_{10} = (110)_2$ et $(0{,}72)_{10} = (0{,}1011\ \text{etc.})_2$, l'écriture binaire de 6,72 est :

  $(6{,}72)_{10} = (110{,}1011\ \text{etc.})_2$, soit environ 0110,1011 avec partie entière et partie dyadique sur 4 bits.

➢ <u>Application :</u> Donner sur 3 + 4 bits l'écriture dyadique approchée ou exacte de $\frac{37}{16}$ et $\frac{10}{3}$:

$\frac{37}{16} = \frac{32 + 5}{16}$

- partie entière = $\frac{32}{16} = 2 = (010)_2$
- partie non entière = $\frac{5}{16}$

A = reste × 2 = …	< 1 ou ≥ 1 ?	→ bit	reste = A – bit
5/16 × 2 = 5/8	< 1	0	5/8 – 0 = 5/8

R = 010,0101

Au final $\frac{37}{16}$

$\frac{10}{3} =$

- partie entière =
- partie non entière =

R ≈ 011,0101etc.

Au final $\frac{10}{3}$

➢ <u>Comparaison des 2 méthodes permettant d'obtenir l'écriture dyadique d'un nombre :</u>

	Intuitif ?	Rapidité à la main ?	Marche aussi pour la partie entière ?
Par soustractions			
Par multiplications			

C. Problèmes liés à la place fixe de la virgule :

Avec les 2 méthodes précédentes, le nombre de chiffres (de bits) de la partie entière et le nombre de chiffres (de bits) de la partie virgulaire étaient fixés à l'avance, avant tout calcul.

L'écriture dyadique est une écriture approximative à virgule fixe des nombres.

	Exemple d'une écriture binaire à virgule fixe sur 8 bits, 4 bits pour la partie entière et 4 bits pour la partie dyadique.	
Exemple	Un grand nombre 1110,0101	Un petit nombre 0001,1010
Constat	Lorsque le nombre est grand, la partie virgulaire est non significative par rapport à la partie entière.	Lorsque le nombre est petit, il y a des 0 inutiles à gauche de la partie entière.
Souhait	Pouvoir augmenter la taille de la partie entière. Avantage : pouvoir ainsi représenter des nombres encore plus grands.	Pouvoir augmenter la taille de la partie virgulaire. Avantage : augmenter ainsi la précision pour les petits nombres.
Idée	Créer une écriture où la taille de la partie virgulaire puisse s'adapter en fonction de la grandeur ou non du nombre ⇒ la place de la virgule ne doit pas être ……	
Bingo	Ce type d'écriture existe : ce sont les écritures à virgule flottante.	

D. Ecritures à virgule flottante :

Dans ce genre d'écriture, il faut :

- une partie constituée des chiffres significatifs : cette partie s'appellera significande ou mantisse.
- une autre partie qui indique la place de la virgule dans le nombre : cette partie s'appellera l'exposant.

Tout ceci rappelle une écriture connue depuis bien longtemps (4ème) : l'écriture …….....

1. Ecriture scientifique : rappels.

➢ L'écriture scientifique d'un nombre non nul est l'écriture de type : $\pm m \times 10^{e}$

avec la mantisse « m » telle que $1 \leq m < 10$ et l'exposant « e » un entier relatif.

➢ Mettre en écriture scientifique les nombres suivants : 25,6 = -0,000 43 =

2. Ecriture binaire pseudo-scientifique :

Soit un nombre en écriture dyadique. Par exemple 0,0110101.

➢ L'idée est de l'écrire sous la forme d'une écriture ressemblant à l'écriture scientifique :

« mantisse × une certaine puissance ».

Quelles conditions sur la mantisse et la puissance ? On déplacera la virgule jusqu'à ce que la mantisse soit entre 1 et 2. Reste à savoir ce qu'implique le déplacement de cette virgule sur la puissance.

Rappel : Déplacer une virgule dans une écriture décimale revient à multiplier par une puissance de ……

Ainsi, déplacer une virgule dans une écriture dyadique revient à multiplier par une puissance de ……

❶ **L'écriture binaire pseudo-scientifique** d'un nombre **non nul** est l'écriture de type : $\pm m \times 2^e$

avec $1 \leq m < 2$ et e un entier relatif.

❷ « **m** » s'appelle la **mantisse (ou significande).** ❸ « **e** » s'appelle l'**exposant**.

➢ Exemples : $0{,}0110101 = 1{,}101\ 01 \times 2^{-2}$ mantisse « m↑ » donc exposant « e↓ ».

$= 1{,}101\ 0100 \times 2^{1110}$ mantisse sur 8 bits et exposant sur 4 bits.

$0{,}001011 =$

Méthode : Trouver l'écriture binaire pseudo scientifique.

- Trouver d'abord l'écriture dyadique (taille de la partie dyadique donnée par la taille de la mantisse).
- Puis mettre en écriture pseudo-scientifique (complémentaire à 2^n si exposant négatif).

➢ Application : Ecriture binaire pseudo-scientifique (mantisse 8 bits ; exposant 4 bits) des nbs suivants :

$\frac{7}{16} =$

$\frac{37}{8} =$

0,1

3. Ecriture pseudo-scientifique normalisée : norme IEEE 754.

La norme IEEE 754 a fixé la taille de la mantisse et de l'exposant dans l'écriture pseudo-scientifique :

Sur 64 bits, un nombre dyadique sera codé de la façon suivante :

- Le bit de poids le plus fort (le bit le plus à gauche) représente le signe.
- Les 11 bits suivants représentent l'exposant (l'ordre de grandeur d'abord).
- Les 52 autres bits représentent la mantisse (la précision ensuite).

Cette norme sera vue plus en détail en études supérieures.

E. Écriture exacte ou approximative des types de nombres ?

	Entiers relatifs *ex : 2 -5*	*Décimaux* *ex : 0,1 1/4*	*Rationnels - Réels* *ex : 1/3 $\sqrt{2}$*
Ecriture décimale (finie ou non ?)	Finie.	…….	……..…….. en général.
Ecriture binaire théorique (finie ?)	Finie.	……..... en général.	……..… en général.
Dans les machines, la taille de la mémoire est-elle infinie ? …….. Donc l'écriture binaire d'un nombre dans la machine est forcément finie ! Conséquence pour la valeur de l'écriture binaire dans la machine du nombre :			
Valeur en machine (exacte ou pas ?)	Exacte.	……..… en général. Taper 0,1 + 0,2 à la console.	……..…..… en général.

VI. PYTHON ET LES DIFFERENTS TYPES DE NOMBRES.

A. Types de nombre gérés par Python :

Type	*En anglais*	*En français*	*Exemples*	*Commentaires*
int	Integer	Entier	• 5 • -5	**Entier de taille arbitraire,** limitée juste par la quantité de mémoire disponible ! Lorsqu'un entier est en dehors de la plage d'entiers fixée par le processeur (par ex de –……. à +……..… sur 64 bits), l'entier est géré automatiquement par un tableau à taille variable de bits.
float	Floating point number	Nombre à virgule flottante	• 5.0 • -5.1	Le séparateur est le « . » et non la « , » ! Utilisé pour représenter **presque toujours approximativement** les types non entiers. Codés en machine avec la norme IEEE 754.

B. Fonctions importantes :

Fonction	*Exemples*	*Commentaires*
int ( )	• int (5.0) → 5 • int ('5') • int (input( ))	• Convertit le float 5.0 en entier 5. • Convertit le caractère '5' en vrai nombre entier. • Convertit en type entier ce qui est entré au clavier.
float ( )	• float (5) → 5.0 • float ('5.45') • float (input( ))	• Convertit l'…… 5 en …… 5.0. • Convertit la chaîne de caractères '5.45' en …… • ……...
type ( )	• type (5) → <type 'int'> • type (input( ))	• Renvoie en sortie le type de la donnée. • Donne le type de ……..

C. Illustration de la dualité du <type 'int'> en Python :

Soit la célèbre suite de Fibonacci notée (F_n) qui calcule un **terme comme la somme des 2 précédents**. Les 2 premiers termes valent $F_0 = 0$ et $F_1 = 1$.

1. Calculer les 3ème, 4ème et 5ème termes : F_2 = F_3 = F_4 =
2. Ecrire un programme en Python qui affiche les 100 premiers termes de la suite de Fibonacci ligne après ligne sous la forme suivante : le numéro du terme ; sa valeur ; son type. Commenter.

D. Problèmes liés aux nombres de <type 'float'> :

➢ Les nombres autres que les entiers relatifs sont représentés : exactement ? approximativement ?

Ce qui explique que 0.1 + 0.2 ne donne pas 0.3 en machine !

A ces erreurs inhérentes à l'écriture se rajoute des erreurs d'arrondis issues des calculs ! Ce qui parfois a réellement causé des bugs très gênants :

- 1994 : Bug du processeur Pentium : 8391667 / 12582905 donnait 0.666869·au lieu de 0.666910.
- Sur certains super-ordinateurs Cray on pouvait déclencher un overflow en multipliant par 1.
- Excel 2007 (premières versions) : en calculant 65535−2−37, on obtenait 100 000.

C'est pourquoi il est très important de connaître et contrôler ces erreurs d'approximation et d'arrondis. Ce qui nécessite une étude approfondie de la norme IEEE 754.

Et surtout surtout surtout :

Eviter de tester une égalité de floats dans un programme !

➢ Ecrire un programme qui en partant de k = 10 lui enlève 0,1 jusqu'à ce que k = 0 (faire afficher k à chaque tour de boucle). Commenter.

E. Entrée au clavier des nombres en binaire :

Taper dans la console 0b11 + 0b10, commenter.

F. Trois fonctions standard de conversion : bin( ), hex( ) et format(,).

1) Entier → Chaîne de caractères binaires : fonction bin( ).

Fonction bin(n) où n est **un entier** (pas un float !), sortie en chaîne de caractères type '0b..'

Exemples : Taper dans la console bin(5) → ……. bin(-5) → ……

Commenter ce dernier exemple :

2) Entier → Chaîne de caractères hexadécimaux (base 16) : fonction hex( ).

Fonction hex(n) où n est **un entier** (pas un float !), sortie en chaîne de caractères type '0x..'

Exemple : Taper dans la console hex(16) = …….

3) Entier écrit dans une certaine base → Ecriture de cet entier dans une autre base : fonction format(,).

Fonction format(nombre_base_quelconque , 'b' pour binaire ou 'X' pour hexadécimal ou '0' pour décimal)

Chercher sur Internet les spécifications exactes de la fonction format(,).

Exemple : Taper dans la console format(0b1011 , 'x') → ……

VII. BILAN DES ACQUISITIONS.

- A la fin de ce livret, je dois savoir :

Codage binaire des nombres (12 compétences)	☹	😐	☺	☺☺
Convertir un nombre dans les 2 sens Base 10 ↔ Base 2.				
Algorithme de conversion Base 10 → Base 2 par soustractions.				
Algorithme de conversion Base 10 → Base 2 par divisions par 2.				
Additionner 2 nombres écrits en binaire.				
Trouver l'écriture binaire d'un nombre entier signé négatif.				
Trouver l'écriture dyadique d'un nombre dyadique ou pas.				
Problèmes liés à l'écriture en virgule fixe.				
Avantages liés à l'écriture en virgule flottante.				
Trouver l'écriture pseudo-scientifique d'un nombre dyadique ou pas.				
Écriture binaire exacte ou approximative des types de nombre.				
Python et les différents types de nombres.				
Entrer directement des nbs binaires en Python. Fonctions de conversion.				

- Il y a 10 sortes de gens, ceux qui comprennent le binaire et les autres.
- Quelle est la suite logique de ce livret ? Numérisation des ……

➢ Perles des hotlines informatiques :

« - Monsieur, faites control+alt+suppr.

- Ah oui, je connais ça. (on entend en fond : click click click....)

- Que se passe-t-il à l'écran Monsieur ?

- Ben rien.

- Recommencez ctrl+alt+supp.

- (click click click click...)

- Et maintenant ?

- Toujours rien.

- Comment faîtes-vous la manipulation ?

- Ben j'appuie sur les touches c,o,n,t,r,o,l et après sur le + et ainsi de suite. Y faut peut-être l'accent sur contrôle ? »

« - Ca me met nom d'utilisateur ou mot de passe incorrect.

- Fermez la fenêtre Monsieur svp.

- Euh oui, c'est vrai qu'il y a un peu de bruit dehors... »