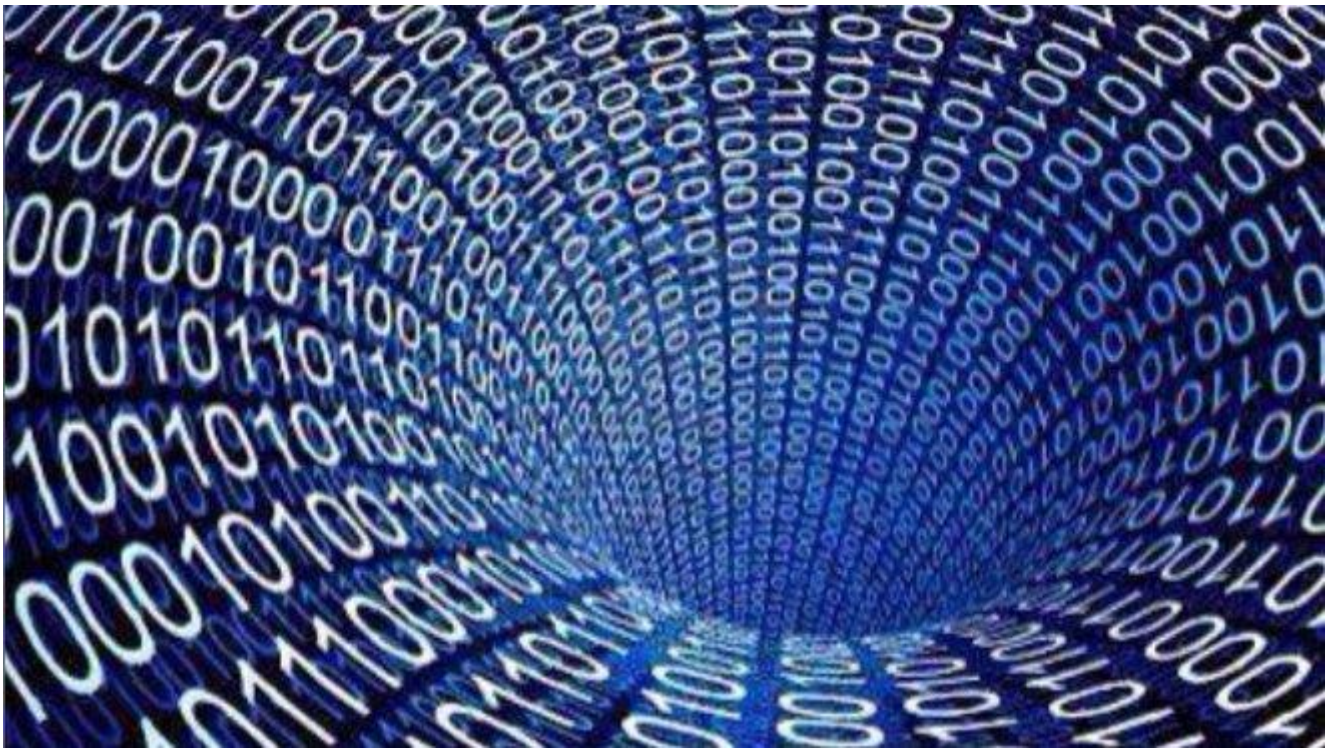
NUMERISATION DES NOMBRES



- *Matériel :*
- *Pré-requis pour prendre un bon départ :*

	☹	😐	☺	☺☺
Puissances de 2				
Division euclidienne (entière)				
Différence entre les chiffres et les nombres.				
Ecrire un algorithme dans ses grandes lignes puis en pseudo-code.				

I. CODAGE BINAIRE DES NOMBRES : DECODAGE - ENCODAGE.

➢ Nos nombres sont usuellement écrits en base …….. Les nombres binaires sont écrits en base ……

Les 2 questions complémentaires qui se posent sont donc :

- Comment convertir de la base 2 vers la base 10 ? Ou, ce qui revient au même,
  Comment calculer la valeur d'un nombre écrit en binaire ?
  C'est la question du décodage (decoding en anglais).
- Inversement, comment convertir de la base 10 vers la base 2 ?
  C'est la question de l'encodage (encoding en anglais).

➢ Cette double conversion, décodage-encodage dans une certaine base, s'appuie mathématiquement sur la décomposition d'un nombre selon les puissances décroissantes de cette base.

Exemple pour le nombre 123 écrit en base 10 qu'on décompose selon les puissances décroissantes de 10 :

$$\mathbf{123} = \mathbf{1} \times 100 + \mathbf{2} \times 10 + \mathbf{3} \times 1$$
$$= \mathbf{1} \times 10^2 + \mathbf{2} \times 10^1 + \mathbf{3} \times 10^0$$

Les chiffres de l'écriture en base 10 sont donnés par les facteurs entre 0 et 9 devant chaque puissance de 10.

La valeur du nombre se trouve en terminant le calcul de cette somme de puissances de 10.

➢ De la même manière :

Les chiffres de l'écriture en base 2 seront donnés par les facteurs entre 0 et 1 devant chaque puissance de 2.

La valeur du nombre écrit en base 2 se trouvera en terminant le calcul de cette somme de puissances de 2.

Voilà donc pour le principe général (valable pour n'importe quelle base d'ailleurs !).

➢ Commençons par le type de nombres le plus simple : les …….

II. REPRESENTATION BINAIRE EXACTE DES NOMBRES ENTIERS NATURELS.

• Tout nombre entier naturel peut être encodé exactement par une suite unique de 0 et de 1.

• Les chiffres des nombres binaires s'appellent le bit des unités, le bit des deuxaines, le bit des quatraines, le bit des huitaines, le bit des ……. etc.

Il est toujours plus simple d'étudier d'abord le décodage, c-à-d la conversion Base ..... → Base .....

A. Base 2 → Base 10 : Calcul de la valeur d'un entier naturel binaire.

➢ On part donc d'un entier naturel écrit en binaire, on veut trouver sa valeur.

Calcul de la valeur d'un nombre 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 $2^0 = 1$.

$= 9$

$(1001)_2 = (9)_{10}$ Dit autrement 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.

❶ Calculer la valeur de $(1011)_2$:

$(1011)_2$ =

R= 11

$(101101)_2$ =

Convertir en base 10 les octets suivants :

$(1100\ 1010)_2$ =

R = 202

$(1111\ 1111)_2$ =

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

❷ Ecrire une méthode permettant de calculer la valeur d'un nb binaire, puis l'algorithme en pseudo-code

« Les Grandes Lignes »

•

Algorithme en pseudo-code

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

❹ Rajouter un chiffre à droite de l'écriture d'un nombre binaire, que se passe-t-il ?

En écriture décimale, lorsqu'on rajoute un 0 à droite de l'écriture d'un entier, on multiplie 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 ……..

❺ a) Une machine traite les nombres entiers positifs sur 2 bits puis sur 3 bits, puis sur 4 bits, puis sur 5 bits.

Ecrire alors dans chaque cas la plage de nombres entiers potentiellement manipulables :

Codage sur	2 bits	3 bits	4 bits	n bits
Plage des entiers représentables	entiers de 0 à 3 noté ⟦0 ; 3⟧	entiers de 0 à ….. noté ⟦0 ; ... ... ⟧		

Que constate-t-on à chaque fois sur la plus grande valeur représentable ?

b) Plus généralement, une machine code les entiers positifs sur n bits.

Montrer que **la plus grande valeur représentable est $2^n - 1$.**

(aide : somme des n premiers termes d'une suite ……..)

c) Les processeurs manipulent les données binaires selon une taille en bits standard.

Plus cette taille est longue, plus la plage d'entiers représentables est grande.

Plus cette taille est longue, plus le processeur est rapide car les données qu'il peut traiter à chaque cycle d'horloge sont plus nombreuses.

Calculer la plage d'entiers manipulables pour les processeurs « historiques » suivants :

Processeur	8 bits	16 bits	32 bits	64 bits
Plage d'entiers représentables	⟦ 0 ; ……. ⟧			

B. Base 10 → Base 2 : Encodage binaire des entiers naturels.

Plutôt qu'un long discours théorique, le mieux est de voir comment faire sur un exemple :

On veut convertir l'entier naturel 53 en base 2.

Il s'agit donc de trouver les facteurs devant chaque puissance de 2 dans la décomposition de 53 en base 2.

$$53 = \text{etc.} + c \times 2^2 + b \times 2^1 + a \times 2^0$$

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 méthodes existent :

1. Méthode ① : Par une suite de questions-soustractions successives.

- Quelle est la plus haute puissance de 2 que je peux mettre dans 53 ? 2^5 car $2^5 = 32 \leq 53$ mais $2^6 = 64 > 53$.

- Est-ce que je peux mettre $2^4 = 16$ dans le reste 21 (= 53 – 32) ? Oui.

  Est-ce que je peux mettre $2^3 = 8$ dans le reste 5 (= 21 – 16) ? Non.

  Est-ce que je peux mettre $2^2 = 4$ dans le reste 5 (= 21 – 16) ? Oui.

  Est-ce que je peux mettre $2^1 = 2$ dans le reste 1 (= 5 – 4) ? Non.

  Est-ce que je peux mettre $2^0 = 1$ dans le reste 1 (= 5 – 4) ? Oui.

  Et il reste 0. Donc la série de questions s'arrête.

- On a $53 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

  L'écriture binaire de 53 est : $(53)_2 = 11\,0101 = 0011\,0101$ (sur 8 bits)

Dans quel ordre obtient-t-on les bits ?

❶ Par cette méthode par questionnement, trouver l'écriture binaire sur un octet de 47 : $(47)_2$ =

❷ Retranscrire les grandes étapes de cette méthode puis écrire l'algorithme correspondant en pseudo-code :

«Les Grandes Lignes »	Algorithme en pseudo-code
•	

Soit N un entier qu'on veut encoder en binaire grâce à cet algorithme. En considérant le nombre de comparaisons effectuées lors du traitement, évaluer en fonction de N la complexité de cet algorithme :

2. Méthode ② : Par une suite de divisions entières successives.

On garde notre exemple 53 comme nombre de départ.

• La méthode précédente renvoyait les bits cherchés dans l'ordre naturel d'écriture (« de la gauche vers la droite ») : du bit de poids le plus fort jusqu'au bit de poids le plus faible.

On veut une méthode qui trouve les bits dans l'ordre inverse (« de la droite vers la gauche ») : du bit de poids le plus faible jusqu'au bit de poids le plus fort.

• On commence donc d'abord à chercher le bit de poids le plus faible c-à-d le chiffre des unités.

Le chiffre des unités est ce qu'il reste après avoir formé le maximum de paquets de 2 dans 53. Le chiffre des unités est donc le reste dans la division euclidienne par 2 de 53.

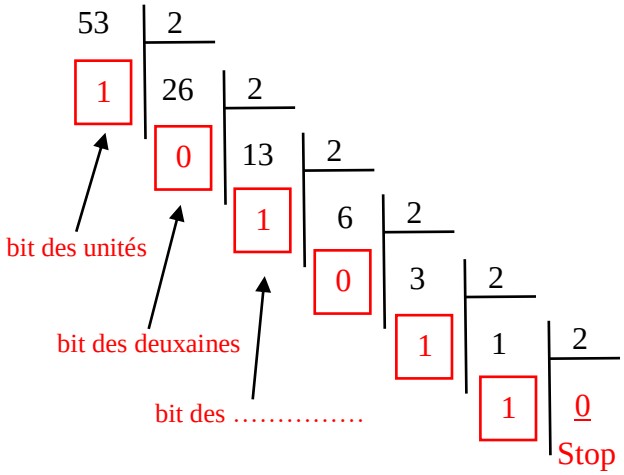
Le chiffre des deuxaines est ce qu'il reste après avoir reformé le maximum de gros paquets de 2 avec les précédents petits paquets de 2. Le chiffre des deuxaines est donc le reste dans la division euclidienne par 2 du quotient trouvé à l'étape précédente.

Et ainsi de suite jusqu'à ce qu'on ne puisse plus reformer de gros gros ... paquets de paquets de paquets etc.

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

- Que 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 ……

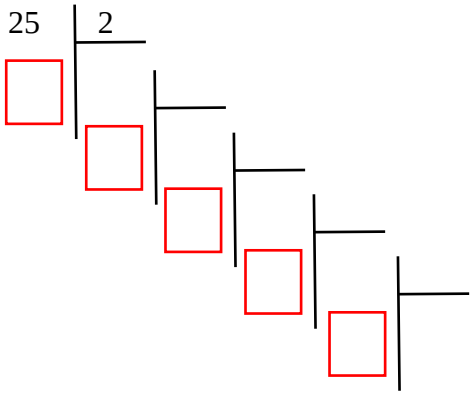
• Concrètement voici ce que cette méthode donne pour 53 :



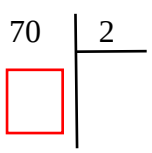
$(53)_{10} = (0011\ 0101)_2$ sur 8 bits

Attention à l'ordre d'écriture inversé !

❶ Par cette méthode par divisions successives, trouver les écritures binaires sur 8 bits de 25 puis 70 :



$(25)_{10}$ =



❷ Ecrire les grandes étapes de cette méthode puis écrire l'algorithme correspondant en pseudo-code :

« Grandes Lignes »	Algorithme en pseudo-code
•	

Soit N un entier qu'on veut encoder en binaire grâce à cet algorithme. En considérant le nombre de comparaisons effectuées lors du traitement, évaluer en fonction de N la complexité de cet algorithme :

➢ Comparaison des 2 méthodes :

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

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

III. ENCODAGE DES NBS ENTIERS RELATIFS (DITS SIGNES).

Un entier relatif a un signe + ou un signe – devant sa ……..... Exemple : .....

En binaire, on ne peut pas ajouter bêtement un signe + ou – devant l'écriture binaire !

Donc l'un des bits doit jouer le rôle du signe : le bit de poids le plus …….. évidemment.

A. Plage d'entiers relatifs représentables (exemple sur 8 bits) :

Ce bit de signe a pour conséquence d'enlever un bit disponible pour le codage de la valeur absolue.

Donc pour le codage des nombres relatifs sur 8 bits, il y aura le bit le plus à gauche pour le signe (soit 0 pour le signe +, soit 1 pour le signe –) et il ne restera que ..... bits pour la valeur absolue.

Le plus petit nombre avec un signe sur 8 bits s'écrit en binaire …….... et vaut évidemment + 0 !

Le plus grand nombre avec un signe + sur 8 bits s'écrit en binaire ……...…..… et vaut +127.

Donc les nombres entiers avec un signe + codables sur 8 bits vont de +0 à +127 soit 128 entiers positifs.

Par symétrie, il y aura aussi 128 entiers négatifs codables en 8 bits : –1 ; – 2 ; etc. jusqu'à ……. et non ….. !

Processeur	8 bits	16 bits	n bits
Plage d'entiers naturels représentables	de 0 à 255 c-à-d ⟦0 ; 255⟧	⟦0 ; 65 535⟧	⟦ …… ; ……... ⟧
Plage d'entiers relatifs signés représentables	de -128 à +127 c-à-d ⟦-128 ; +127⟧	de -……. à +……..	

B. Encodage des entiers signés positifs :

Leur écriture commence par 0. Puis leur valeur absolue s'écrit comme pour les entiers naturels sans signe.

C. Encodage des entiers signés négatifs :

Pour l'instant, on sait juste que l'écriture des entiers négatifs commence par ….. pour indiquer le signe ….

Mettons-nous sur 3 bits pour simplifier les exemples qui vont suivre :

- 1ère idée : Ecrire l'opposé en binaire de la même façon qu'en décimal c-à-d en changeant juste le signe.

+2 s'écrit **0**10 donc –2 s'écrirait **1**10. +1 s'écrit **0**01 donc –1 s'écrirait **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 dans la colonne gauche suivante.

Si une retenue déborde tout à gauche de la représentation, 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 la représentation sur 3 bits) mais hélas, +1 + (–1) ne vaut pas 000 !

Donc dans l'encodage des entiers négatifs, un changement du bit de signe seulement ne marche pas !

➢ 2ème idée : Revenir à la définition algébrique de l'opposé : « nombre binaire + opposé binaire = 0 ».

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 avec un petit 1 en retenue qui déborde tout à gauche) ?

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

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

En repartant des exemples +2 et +1 puis en rajoutant les « inversés », on obtient bien 111.

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

Au final, l'opposé doit être : « l'inversé » + 1. Vérifions sur nos 2 exemples :

0 1 0 (+2) + 1 1 0 « inversé » + 1 1 0 0 0	0 0 1 (+1) + 1 1 1 « inversé » + 1 1 0 0 0

L'opposé de 010 s'écrit donc 110.
L'opposé de 001 s'écrit donc 111.

➢ En résumé : L'écriture binaire de l'opposé d'un entier s'obtient de la façon suivante :

Technique du Complémentaire à 2^n	Exemple : $(-2)_2$ sur 4 bits ?
1) Donner l'écriture binaire de l'entier.	1) $(2)_2 = 0010$
2) Fabriquer son « inversé » : inverser chaque bit ($0 \rightarrow 1$ et $1 \rightarrow 0$).	2) « inversé » = 1101
3) Rajouter 1 à l'inversé obtenu en 2).	3) Donc $(-2)_2 = 1101 + 0001 = 1110$.

➢ Remarques importantes :

1. Avec la complémentation à 2^n, on obtient toutes les écritures des entiers négatifs qui ont un symétrique positif (de −3 à −1 dans notre cas de départ sur 3 bits). Mais comment obtient-t-on la représentation du plus petit entier négatif -2^{n-1} (–4 sur 3 bits) qui n'a pas de symétrique positif dans la représentation ?
   On prend la représentation de l'avant dernier négatif (–3 sur 3 bits) et on enlève 1.
   On trouve toujours une écriture de type 10..00 (100 dans notre cas sur 3 bits pour –4).
   Dit autrement, l'écriture binaire du plus petit entier négatif -2^{n-1} est la même que celle de 2^{n-1}.
2. Pourquoi parle-t-on de complémentaire à 2^n ?
   Toujours avec notre exemple de départ, cette méthode permet donc de trouver l'écriture binaire sur 3 bits de l'opposé k d'un nombre binaire p de telle sorte que $(p)_2 + (k)_2 = (000)_2$.
   En fait, ce n'est pas $(000)_2$ exactement mais $(1000)_2$ qui vaut 2^3. Donc l'opposé k est le nombre qu'il faut rajouter à p pour retrouver 2^3 c-à-d le complémentaire de p à 2^3 (à 2^n plus généralement quand on travaille sur n bits).
3. On parle souvent de complémentaire à 2 au lieu de complémentaire à 2^n, mais c'est moins explicite !
4. Puisque l'opposé d'un nombre p est en fait le complémentaire à 2^n, on peut aussi trouver l'écriture binaire de l'opposé en calculant tout simplement $2^n - p$, puis en en donnant l'écriture binaire sur n bits.

➢ Applications :

❶ Un processeur représente les entiers sur 4 bits.

1. Quelle est la plage d'entiers *signés* manipulables par le processeur ? De ……. à …….
2. Donner les écritures sur 4 bits de 5 et –5 ; de 7 et –7 ; de –8.

❷ Par la technique du complément à 2^n, un processeur travaillant sur 5 bits a calculé l'écriture binaire d'un entier signé et a trouvé 10110.

1. Quelle est la plage d'entiers *signés* manipulables par le processeur ? De ……. à …….
2. Quel est le signe de cet entier signé ? Justifier :
3. Lou Phoque affirme que ce nombre est -6 ! Comment elle a trouvé -6 ?
4. A-t-elle raison ? Si non, vraie valeur de cet entier signé représenté par 10110 ? (2 méthodes)
5. Un entier signé a pour écriture sur 5 bits 10011. Cet entier est-il : 19 ou -3 ou 13 ou -13 ?

Bit octet et Python.

IV. REPRESENTATION APPROXIMATIVE DES AUTRES TYPES DE NOMBRES.

A. Approximative ?

- Les entiers ont une écriture décimale finie et on montre en Mathématiques que n'importe quel entier relatif possède une écriture binaire unique finie et exacte.

Qu'en est-il des autres types de nombres (décimaux, rationnels et réels) ?

- La plupart des nombres rationnels (fractions) et réels ont une écriture décimale infinie (exemple : 4/3 = 1,33333… ou $\sqrt{2}$ = 1,4142…). Donc leur représentation binaire en machine, forcément finie (la mémoire disponible dans la machine ne pouvant être ……..…..…… !), ne peut être ……..

- Qu'en est-il des nombres décimaux relatifs ?

Comme pour les entiers, l'écriture décimale des nombres décimaux est ……. Donc il est possiblement possible que leur représentation binaire soit exacte. Faux espoir !

Taper dans la console Python : 0,1 + 0,2 = …… Commentez.

B.

V. BILAN DES ACQUISITIONS.

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

Représentation et codage en binaire (7 compétences)	☹	😐	☺	☺☺
Les conversions binaires Base 10 ↔ Base 2.				
Ecrire un algorithme de conversion binaire d'un nombre entier.				
Additionner 2 nombres en binaire.				
Ecrire la représentation binaire d'un nombre entier signé.				

- Quelle est la suite logique de ce livret ? « Numérisation des données non ……. ».

- Perles des hotlines informatiques :

« Secrétaire - J'ai un problème avec Windows.

Hotline - Qu'avez-vous sur l'écran ?

Secrétaire - Euh... un pot de fleur.

Hotline - Non, je veux dire "Qu'est-ce qui est écrit sur l'écran ?".

Secrétaire - Ha d'accord... euh... Sony. »

« Client - Vous me dites "pas de majuscules pour le mot de passe", c'est bien ça ?

Hotline - Exact.

Client - Et, les chiffres, je les mets en minuscule aussi ? »