Système de numération

Un processeur est un système automatique de traitement d'information. Le mot information, dont dérive le terme informatique est pris dans le sens "éléments significatifs" tels que texte, parole, image, mesure d'une grandeur physique, nombre, etc...
Cette information devant être représentée sous une forme physique appropriée au traitement quelle doit subir, la première étape consiste en une transformation appelée codage.

Nous coderons donc les signaux (images, paroles, textes) sous forme de 0 et de 1, compréhensibles par une machine.
Notre système conventionnel de comptage en base 10 incompatible avec la machine, nous à donc conduit à étudier d'autres systèmes de numération.
Les systèmes de numération consistent à utiliser un ensemble de symboles appelés digits (comptage avec les doigts) ainsi qu'une convention d'écriture.
Le nombre de digits utilisés correspond à la base du système.

Organisation du cours

Système de numération décimal, octal, binaire et hexadécimal

Les systèmes de numérations binaire et hexadécimal sont très utilisés dans les domaines de l'électronique et de l'informatique. Tout programmeur se doit de les connaître en plus des systèmes décimal et octal.

Principe d'une base

- La base est le nombre qui sert à définir un système de numération.
  La base du système décimal est dix alors que celle du système octal est huit.
  Quelque soit la base numérique employée, elle suit la relation suivante :

  ou : **b**i : chiffre de la base de rang i
  et : **a**i : puissance de la base a d'exposant de rang i

  Exemple : base 10
  1986 = (1 x 103) + (9 x 102) + (8 x 101) + (6 x 100)

Le système décimal

- Le système décimal est celui dans lequel nous avons le plus l'habitude d'écrire.
  Chaque chiffre peut avoir 10 valeurs différentes :
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, de ce fait, le système décimal a pour **base** 10.
  Tout nombre écrit dans le système décimal vérifie la relation suivante :
  $745 = 7 \times 100 + 4 \times 10 + 5 \times 1$
  $745 = 7 \times 10 \times 10 + 4 \times 10 + 5 \times 1$
  $745 = 7 \times 10^2 + 4 \times 10^1 + 5 \times 100$

  Chaque chiffre du nombre est à multiplier par une puissance de 10 : c'est ce que l'on nomme le **poids du chiffre**.
- L'exposant de cette puissance est nul pour le chiffre situé le plus à droite et s'accroît d'une unité pour chaque passage à un chiffre vers la gauche.
  $12\,435 = 1 \times 10^4 + 2 \times 10^3 + 4 \times 10^2 + 3 \times 10^1 + 5 \times 100$.

  Cette façon d'écrire les nombres est appelée **système de numération de position**.

  Dans notre système conventionnel, nous utilisons les puissances de 10 pour **pondérer** la valeur des chiffres selon leur position, cependant il est possible d'imaginer d'autres systèmes de nombres ayant comme base un nombre entier différent.

Le système octal

- Le système octal utilise un système de numération ayant comme base 8 (octal => latin octo = huit).
  Il faut noter que dans ce système nous n'aurons plus 10 symboles mais 8 seulement :

  0, 1, 2, 3, 4, 5, 6, 7

  Ainsi, un nombre exprimé en base 8 pourra se présenter de la manière suivante :

  (745)8

  Lorsque l'on écrit un nombre, il faudra bien préciser la base dans laquelle on l'exprime pour lever les éventuelles indéterminations (745 existe aussi en base 10). Ainsi le nombre sera mis entre parenthèses (745 dans notre exemple) et indicé d'un nombre représentant sa base (8 est mis en indice).
- Cette base obéira aux même règles que la base 10, vue précédemment, ainsi on peut décomposer (745)8 de la façon suivante :

$$(745)_8 = 7 \times 8^2 + 4 \times 8^1 + 5 \times 8^0$$
$$(745)_8 = 7 \times 64 + 4 \times 8 + 5 \times 1$$
$$(745)_8 = 448 + 32 + 5$$

Nous venons de voir que :

$(745)_8 = (485)_{10}$

Le système binaire

- Dans le système binaire , chaque chiffre peut avoir 2 valeurs différentes : 0, 1.

  De ce fait, le système a pour base 2.
  Tout nombre écrit dans ce système vérifie la relation suivante :

  $(10\ 110)2 = 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$
  donc : $(10110)2 = (22)10$.

  Tous les systèmes de numération de position obéissent aux règles que nous venons de voir.

  Tous les systèmes de numération de position obéissent aux règles que nous venons de voir.

Tableau récapitulatif

Système décimale	Système octal	Système binaire
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111
16	20	10000
17	21	10001
-	-	-
-	-	-
63	77	111111
64	100	1000000
65	101	1000001

Le système hexadécimal

- Le système hexadécimal utilise les 16 symboles suivant :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

De ce fait, le système a pour base 16.

Un nombre exprimé en base 16 pourra se présenter de la manière suivante :

(5AF)16

La correspondance entre base 2, base 10 et base 16 est indiquée dans le tableau ci-après :

Base 10	Base 16	Base 2
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Le système hexadécimal

- Le nombre (5AF)16 peut se décomposer comme suit :

$$(5AF)16 = 5 \times 162 + A \times 161 + F \times 160$$

En remplaçant A et F par leur équivalent en base 10, on obtient :

$$(5AF)16 = 5 \times 162 + 10 \times 161 + 15 \times 160$$
$$(5AF)16 = 5 \times 256 + 10 \times 16 + 15 \times 1$$

donc = (5AF)16 = (1455)10

Conversion décimal binaire, octal, hexadécimal et changement de base

- Conversion d'un nombre de base quelconque en nombre décimal

  En exposant les principes des **systèmes de numération** de position, nous avons déjà vu comment convertir les nombres de base 8, base 2 et base 16 en nombres décimaux.

- Conversion d'un nombre décimal en nombre binaire

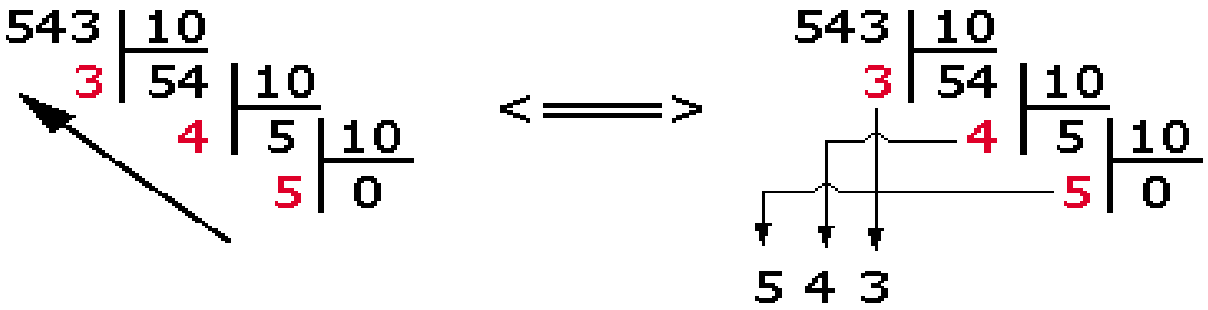
Pour expliquer ce type de conversion, on peut revenir sur le système décimal. Si nous divisons le nombre (543)10 par 10, nous obtenons comme quotient 54 et 3 comme reste. Cela signifie que ce nombre équivaut à :

(54 x 10) + 3

Le reste 3 est le chiffre indiquant le nombre d'unités.

En redivisant ce quotient (54) par 10, nous obtenons 5 comme deuxième quotient et 4 comme reste. Ce reste donne le deuxième chiffre du nombre, donc celui des dizaines.

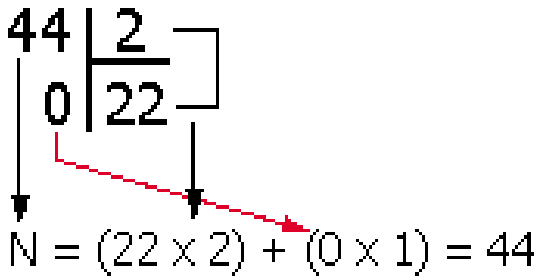
- Enfin, si l'on divise ce deuxième quotient par 10, nous obtenons O et il restera 5 qui représentera le chiffre des centaines.



- **Résumer du principe de conversion**
  En divisant successivement un nombre par la base (10) et en ne **conservant que les restes**, on a réussi à exprimer le nombre par des chiffres inférieurs de 10. Mais attention, il faut **lire les restes de bas en haut**.

Conversion binaire

Maintenant si nous divisons un nombre décimal par 2, le quotient indique le nombre de fois que 2 est contenu dans ce nombre et le reste indique le chiffre des unités dans l'expression du nombre binaire.
Soit N le nombre, Q1 le quotient et R1 le reste, nous avons :
N = (Q1 x 2) + (R1 x 1)
N = (Q1 x 21) + (R1 x 20)
Exemple :

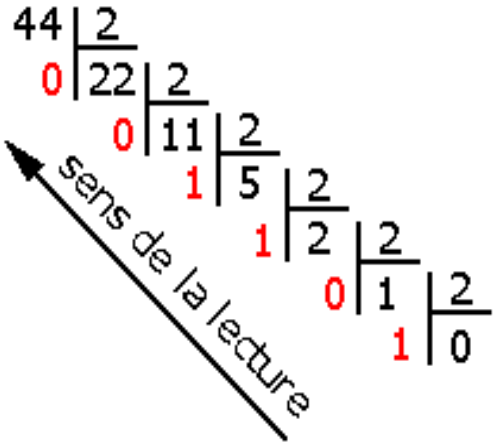


Exemple :

soit :

N = (22 x 2) + (0 x 1) = 44

- Pour obtenir l'expression binaire d'un nombre exprimé en décimal, il suffit de **diviser successivement ce nombre par 2** jusqu'à ce que le quotient obtenu soit égal à O.
  Comme pour la conversion dans le système décimal les restes de ces divisions lus de bas en haut représentent le nombre binaire.

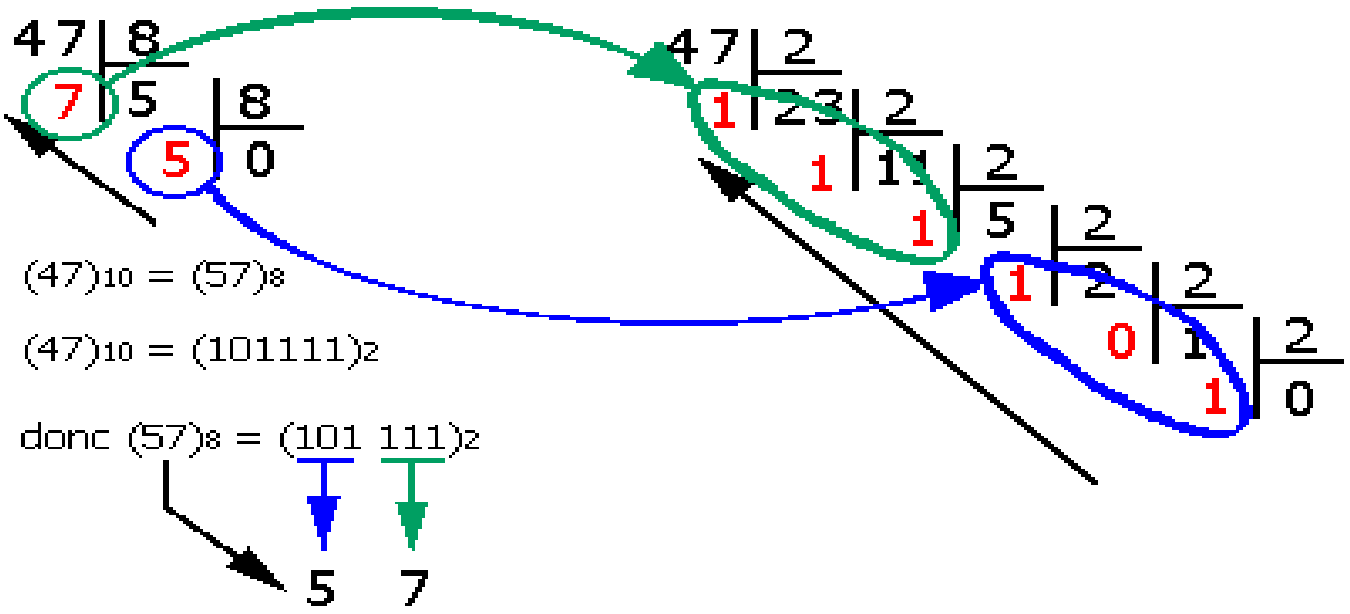


(44)10 = (101100)2

Relation entre les nombres binaires et les nombres octaux

Exprimons (47)10 dans le système octal et le système binaire.

Nous obtenons :

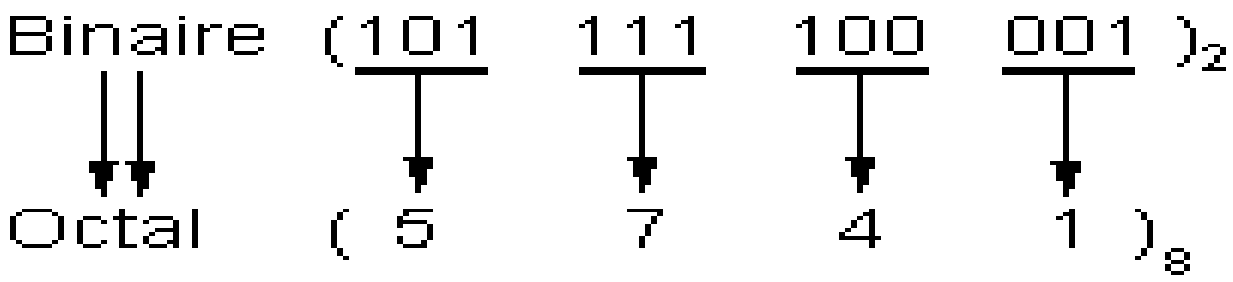


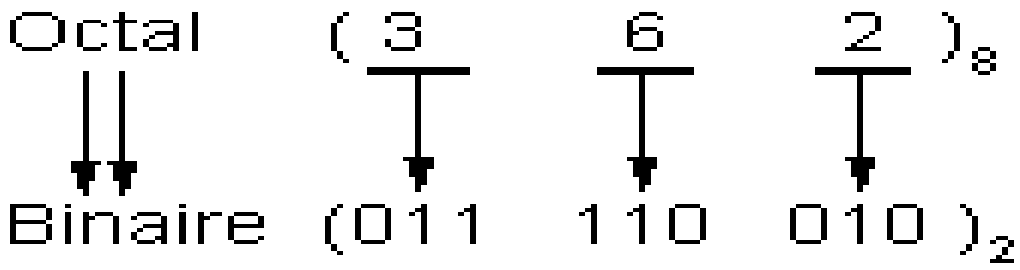
- Nous pouvons remarquer qu'après 3 divisions en binaire nous avons le même quotient qu'après une seule en octal. De plus le premier reste en octal obtenu peut être mis en relation directe avec les trois premiers restes en binaire :

  (111)2 = 1 x 22 + 1 x 21 + 1 x 20
  (111)2 = 1 x 4 + 1 x 2 + 1 x 1
  (111)2 = (7)8
  et il en est de même pour le caractère octal suivant :

  (101)2 = 1 x 22 + 0 x 21 + 1 x 20
  (101)2 = 1 x 4 + 0 x 2 + 1 x 1
  (101)2 = (5)8
- Cette propriété d'équivalence entre chaque chiffre octal et chaque groupe de 3 chiffres binaires permet de passer facilement d'un système à base 8 à un système à base 2 et vice versa.

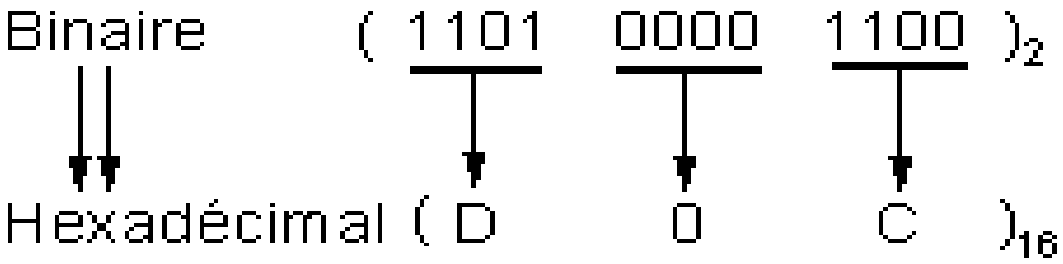
Exemple de conversion binaire octal et octal binaire :

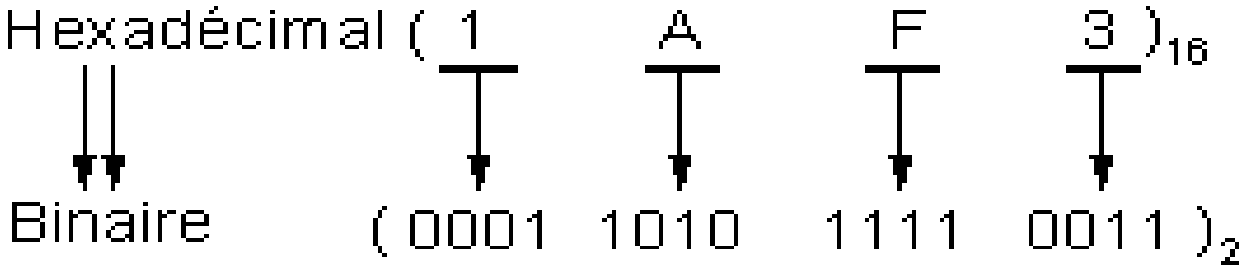




Relation entre les nombres binaires et les nombres hexadécimaux

La propriété d'équivalence que nous venons de voir entre le binaire et l'octal existe entre l'hexadécimal et le binaire.
La seule différence est qu'il faut exprimer chaque caractère hexadécimal à l'aide de 4 informations binaires.





Nombre à virgule flottante

Représentation des nombres à virgule flottante

- Nous savons qu'il est nécessaire de stocker des données dans les machines. Ainsi le nombre 9,750 se trouvera mémorisé sous la forme suivante :

  1001,11

  Toutefois cette expression binaire ne suffit pas à définir totalement notre donnée car il n'y a aucune indication sur la valeur du poids binaire affecté aux différents bits, d'où la notion de virgule.
- En utilisant cette notion de virgule, notre nombre peut s'écrire de la manière suivante :

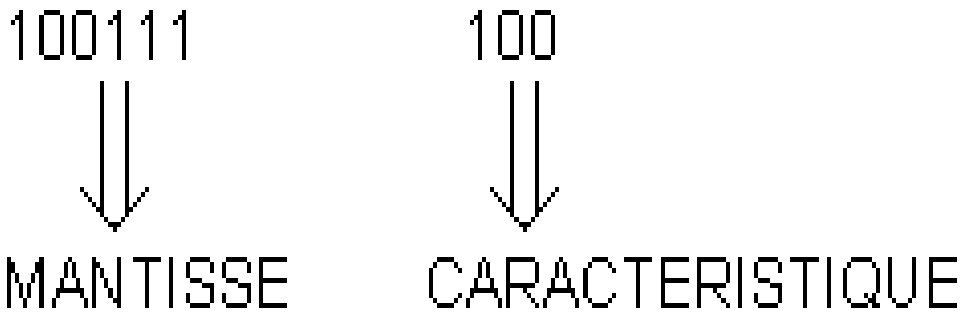
  N = 1001,11 x 20
  N = 100,111 x 21
  N = 10,0111 x 22
  N = 1,00111 x 23
  N = 0,100111 x 24

  Cette dernière expression présente l'avantage de représenter la grandeur par un nombre inférieur à 1 multiplié par une puissance de 2. L'exposant 4 est bien entendu représentatif de la position de la virgule.

- Donc pour définir totalement notre information (9,750) il faudra dans ce système de représentation **deux termes** :

le terme 100111 appelé MANTISSE
le terme 100 appelé CARACTERISTIQUE

Si dans une machine les informations sont représentées en virgule flottante, elles se présenteront de la manière suivante :



et elle sera égale à :

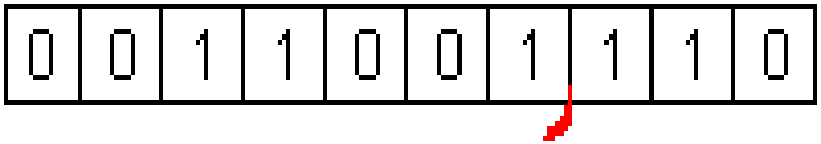
N = 0,100111 x 24
N = 1001,11

Nombre à virgule fixe

Représentation des nombres à virgule fixe

- La représentation de nombre en virgule flottante n'est pas la seule imaginable. Expliquons la représentation de nombre en virgule fixe par un exemple.

  Soit (25,75)10 = (11001,110)2



La position de la virgule est **fixe** arbitrairement à la 4ème case vers la gauche. La position de la virgule n'est pas visualisée.
La case la plus à droite représente le poids 20 : ce qui est évidemment faux.
Cette représentation suppose la multiplication implicite de ce nombre par 2-3
Le terme -3 est représentatif du positionnement fixe de la virgule. Il devra impérativement être mémorisé.

Comparaison des deux représentations (virgule fixe et virgule flottante)

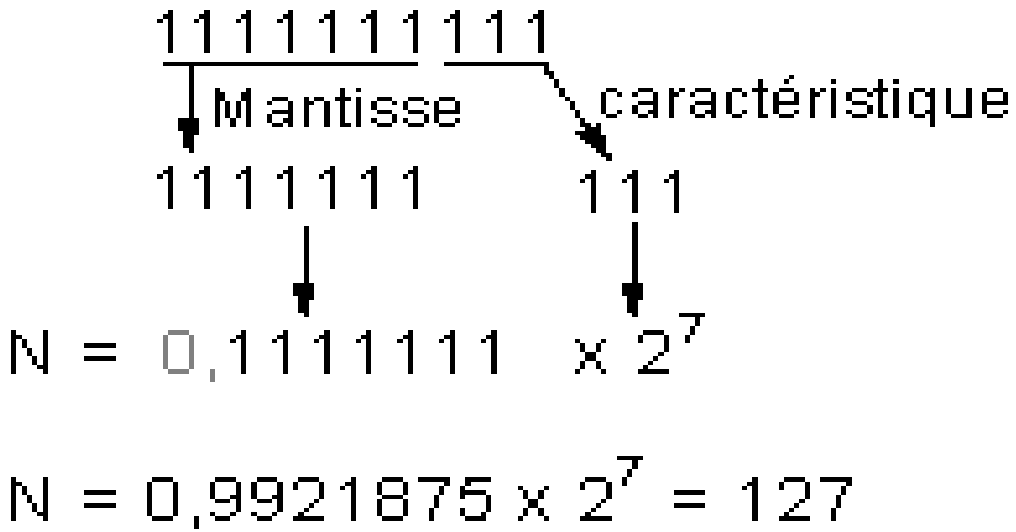
- Nous considérons avoir mémorisé le terme d'élévation à la puissance -3.
  Si on travaille sur 10 positions, le nombre le plus élevé que l'on pourra écrire sera égal à :

  - en virgule fixe

$$1111111111 = 1023 \times 2^{-3}$$
$$= 1023 \times \frac{1}{2^3}$$
$$= 1023 \times \frac{1}{8}$$
$$= \frac{1023}{8} = 127{,}875$$

Comparaison des deux représentations (virgule fixe et virgule flottante) suite

- **en virgule flottante**



- Par ailleurs, si nous voulons écrire un nombre inférieur à 1, par exemple 47/64 (0,734375) nous aurons :

  7 chiffres significatifs en virgule flottante,
  3 chiffres significatifs en virgule fixe.

  - en virgule fixe :
  0 0 0 0 0 0 0 1 0 1

  - en virgule flottante :
  1 0 1 1 1 1 0 0 0 0

  Si l'on cherche l'équivalent décimal :

  - en virgule fixe :
  101 x 2-3 = 1/2 + 1/8 = 40/64

  - en virgule flottante :
  0,1011110 x 20 = (1/2 + 1/8 + 1/16 + 1/32 + 1/64) x 20 = 47/64
  On s'aperçoit que la représentation en virgule fixe apporte une erreur qui peut, dans certains cas, ne pas être négligeable.

Code complément à 2 et nombres signés

- Représentation des nombres signés par leur valeur absolue et leur signe

C'est naturellement la première représentation qui vient à l'esprit. Il suffit d'affecter un bit pour le signe et d'attribuer par convention la **valeur 0 au signe +** et la **valeur 1 au signe -**.
Ainsi le nombre +32 s'écrira dans le système binaire :

0	100000
signe	nombre

et le nombre -32 :

1	100000
signe	nombre

Autres exemples :

- **Le nombre + 9,750 s'écrit :**

0	100111	100
signe	Mantisse	caractéristique

et - 9,750 :

1	100111	100
signe	Mantisse	caractéristique

Représentation des nombres signés dans le code du complément restreint

- **Nous allons d'abord définir ce qu'est le complément restreint. Pour cela il faut tenir compte du format de la donnée et de la base dans laquelle elle est exprimée.**

  Exemples :
  Soit l'information (453)10 ; son format est de 3 caractères et la base utilisé est 10.
  La valeur maximale que l'on peut exprimer dans ce format est :
  9 9 9

  La différence qui existe entre cette valeur maximale et 453 s'appelle le **complément restreint: 999-453=546**

- On le nomme aussi **complément à 9** car la base utilisée est 10.

Complément restreint

- Cette notion de complément restreint se retrouve avec n'importe quelle base utilisée et plus particulièrement en binaire :

  Complément restreint de (1001)2 est: 1111-1001= 0110
  on dit que c'est le complément à 1.

  Complément restreint de (F0A8)16 est: FFFF-F0A8=0F57
  C'est le complément à 15

  Si nous reprenons l'exemple du binaire, il n'est même pas nécessaire d'exécuter une opération de soustraction pour obtenir ce complément restreint on s'aperçoit qu'il suffit de transformer tous les 1 en 0 et vice versa pour l'obtenir **(complément bit à bit).**

  (100110)2 à pour complément restreint : 011001

Représentation des nombres signés dans le code du complément vrai

- **Le complément vrai d'un nombre est la valeur qu'il faut ajouter à ce nombre pour obtenir la valeur maximale + 1 que l'on peut exprimer (en tenant compte du format et de la base utilisés).**

<u>Exemples</u> :

Calcul du complément vrai de (453)10
Valeur maximale ===> 999
Valeur maximale + 1 ===> 1 000
Complément vrai : 1000-453=547

Code complément à 2

- **Calcul du complément vrai de (8AF)16**
  Valeur maximale ===> FFF
  Valeur maximale + 1 ===> 1 000
  Complément vrai : 1000-8AF=$(751)_{16}$

  On peut aussi obtenir le complément vrai d'un nombre en calculant d'abord son complément restreint et en ajoutant ensuite 1.

Exemples : **a) 999-453=546**

546+1=547 (complément vrai)

b) 1111-1001=0110

0110+1=0111 (complément vrai)

Code complément à 2

- **Restons en binaire (base 2) et appliquons une autre méthode pour traduire un nombre en complément à 2. (le complément vrai est également appelé complément à 2)**

  On part du bit de poids le plus faible (bit de droite) :
  ===> si c'est un zéro, on recopie 0 jusqu'au premier 1 rencontré,
  ===> si c' est un "1", on garde ce premier 1.

  Ensuite on inverse tous les bits après le premier 1 rencontré à partir de la droite.

Exemple: Le nombre (42)10 = (101010)2 s'écrit en complément vrai : $(010110)_2$

la méthode du complément restreint + 1 :

- **Complément vrai = complément restreint + 1 = code complément à 2**

Valeur de départ		111011	$(59)_{10}$
Calcul complément restreint	=>	000100	on inverse tous les bits
	+	1	on ajoute 1
Soit un complément vrai	=>	000101	