

---

# Chapitre 2

## Représentation des informations dans l'ordinateur

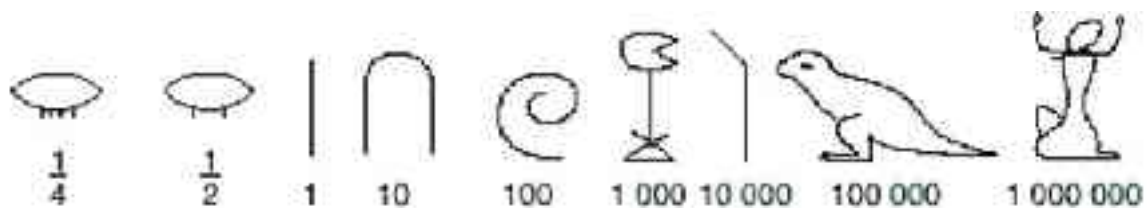
---

<b>Pré requis</b>	Concepts et définitions de l'informatique
<b>Objectifs du chapitre</b>	Connaître comment sont représentés les différents types de données d'un ordinateur. Appliquer l'arithmétique binaire pour résoudre des opérations dans le système usuel.
<b>Mots clés du chapitre</b>	Représentation interne, Représentation externe, Codage, Décodage, Transcodage, Bit, Octet, Caractère, Débordement, Virgule fixe, Virgule flottante.
<b>Éléments de contenu</b>	<ol style="list-style-type: none"><li>1.Introduction<ol style="list-style-type: none"><li>1.1.1.1. Bit</li><li>1.2.1.2. Codage et décodage</li></ol></li><li>2.Représentation des données non numériques<ol style="list-style-type: none"><li>2.1.Le codage des caractères</li><li>2.2.Le transcodage</li></ol></li><li>3.Représentation des données numériques<ol style="list-style-type: none"><li>3.1.Entiers positifs ou nuls</li><li>3.2.Les unités employées :</li><li>3.3.Changement de base</li><li>3.4.L'arithmétiques binaires</li><li>3.5.Entiers négatifs</li><li>3.6.Les nombres fractionnaires</li></ol></li><li>4.Représentation des instructions<ol style="list-style-type: none"><li>4.1.Code opération</li><li>4.2.Les opérandes</li><li>4.3.Les machines selon le nombre d'adresse supportées</li></ol></li><li>5.En résumé</li><li>6.Série d'exercices</li></ol>

## 1. Introduction

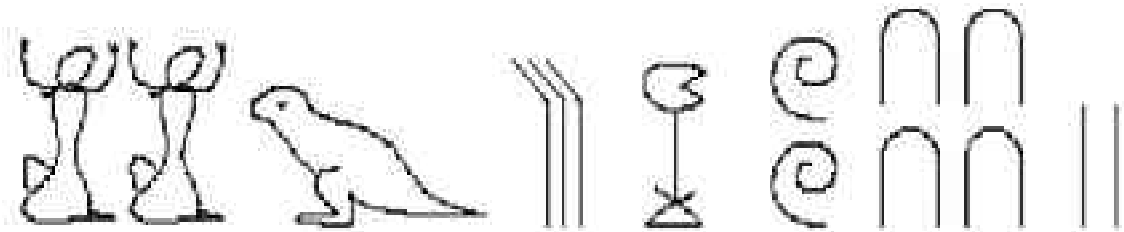
Pour traiter et communiquer l'information, l'être humain à toujours eu recours à des conventions pour la représenter et des règles pour la manipuler.

Par exemple le système utilisé par les anciens égyptiens comportait des symboles pour les puissances de 10 allant de 0 à 6 :



**Figure 2.1. Les symboles utilisés par les anciens égyptiens pour représenter des nombres**

Le coefficient associé à chaque puissance était simplement présenté par une répétition du signe correspondant, par exemple la valeur 2131242 est représenté comme suit :



**Figure 2.2. Représentation du nombre 2131242 par les anciens égyptiens.**

Les informations traitées par l'ordinateur sont de différents types (nombres, instructions, images, séquences d'images animées, sons, etc.), mais pour des raisons technologiques (reliées à la réalisation de l'ordinateur), elles sont toujours représentées au niveau de ce premier sous forme binaire : c'est à dire une suite de chiffres parmi 0 et 1.

### 1.1. 1.1. Bit

**Définition** C'est l'information la plus élémentaire dans un ordinateur, elle correspond à un chiffre binaire qui prend comme valeur 0 ou 1.

**Remarque**

- *Au niveau interne de l'ordinateur, chaque bit correspond à un circuit électronique caractérisé physiquement par deux états d'équilibre, par exemple 0 et 5 Volts.*
- *Une information plus complexe dans un ordinateur tel qu'une lettre, un nombre, ..., se ramène toujours à un ensemble de bits.*

**Remarque** *L'équivalent du terme octet en anglais est Byte. Le terme bit existe en anglais et possède la même signification qu'en français.*

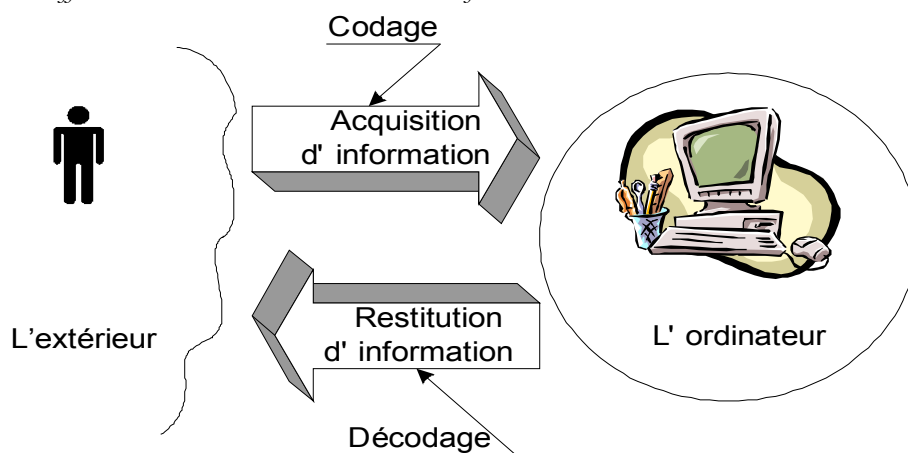
## 1.2. Codage et décodage

Comme la représentation de l'information est différente entre l'Homme et l'ordinateur, on parle alors de représentation externe et interne (on raisonne par rapport à l'ordinateur). L'acquisition de l'information par l'ordinateur se traduit par un passage de la représentation externe vers celle interne et vice versa pour la restitution.

**Définition** Le codage d'une information revient à établir une correspondance entre la représentation externe de l'information (exemple : lettre A) et sa représentation interne (au niveau de l'ordinateur) sous forme de suite de bits.

**Définition** Le décodage d'une information revient à établir une correspondance entre la représentation interne (au niveau de l'ordinateur) de l'information sous forme de suite de bits et sa représentation externe (exemple : lettre A).

**Remarque** *Le codage s'effectue au moment de l'acquisition de l'information par l'ordinateur. Le décodage s'effectue au moment de restitution de l'information vers l'extérieur.*



**Figure 2.3. Codage et décodage d'informations pendant son acquisition et sa restitution**

Pour les informations manipulées par l'ordinateur, on distingue :

- Les instructions : Ecrite en langage machine, les instructions représentent les opérations que l'ordinateur est capable d'effectuer.
- Les données : Ce sont les opérands sur lesquelles portent les opérations ou produites par celles-ci. Une addition par exemple s'applique à deux opérands donnant un résultat qui est leur somme.

## 2. Représentation des données non numériques

Les données non numérique correspondent aux caractères alphanumériques (A, B...Z, a, b,...z, 0, 1...9) et aux caractères spéciaux (? , !, \$, etc.). Parmi les caractères spéciaux il y a des caractères non imprimables c'est à dire des caractères au quel ne correspond pas un symbole. Ces derniers sont des caractères de contrôles (NUL, STX, ETX, EOT, ACK, BEL, LF,...). Ils sont nécessaires pour certains types d'applications spéciales (passer à la ligne (CR, LF), émettre un bip sonore (BEL), etc.).

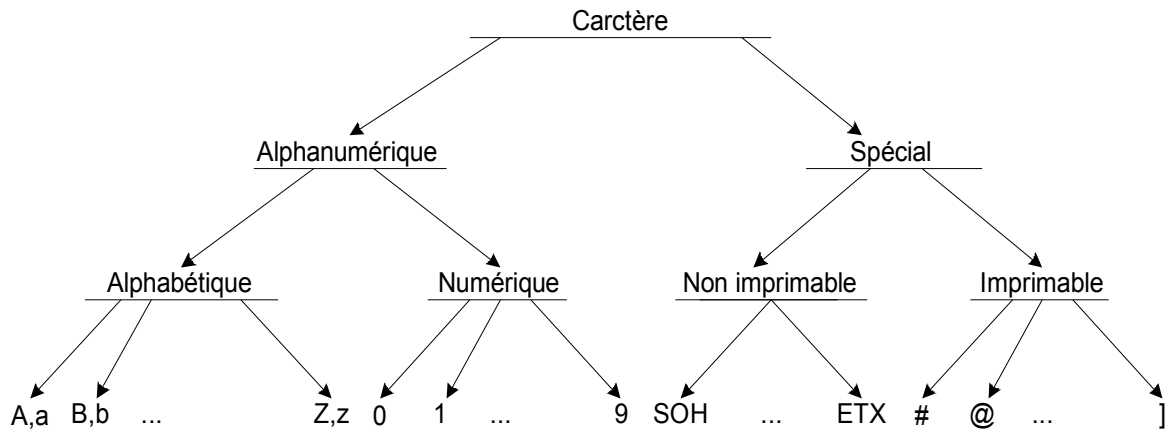


Figure 2.4. Les différents types de caractères.

## 2.1. Le codage des caractères

**Définition** Le codage des caractères est une convention qui associe à chaque caractère un code binaire unique. Il est réalisé par une table de correspondance, propre à chaque code utilisé (Voir tableau 2.2).

Parmi les plus connus, on peut citer les codes :

- BCD (Binary Code Decimal), un caractère est codé sur 6 bits.
- ASCII (American Standard Code for Information Interchange) 7 bits
- EBCDIC (Extended Binary Coded Decimal Internal Code) 8 bits
- UNICODE (16 bits)
- ISO/IEC 10646 (32 bits)

Caractère	Codage		
	BCD (6 bits)	ASCII (7 bits)	EBCDIC (8 bits)
0	000000	0110000	11110000
1	000001	0110001	11110001
2	000010	0110010	11110010
...	...	...	...
9	001001	0111001	11111001
...	...	...	...
A	010001	1000001	11000001
B	010010	1000010	11000010
C	010011	1000011	11000011
...	...	...	...

Tableau 2.1. Extrait des tables de correspondance de certains codes.

- Remarques**
- *Les deux derniers codes ont été créés récemment (début des années 90). En effet, 128 ou 256 valeurs ne suffisent pas pour représenter l'ensemble de tous les caractères de toutes les langues de la planète.*
  - *Un texte est représenté dans un ordinateur en faisant son codage caractère par caractère.*
  - *Plusieurs points importants à propos du code ASCII :*
    - *Les lettres se suivent dans l'ordre alphabétique (codes 65 à 90 pour les majuscules, 97 à 122 pour les minuscules), ce qui simplifie les comparaisons.*
    - *On passe des majuscules aux minuscules en modifiant le 5<sup>ème</sup> bit, ce qui revient à ajouter 32 au code ASCII décimal.*
    - *Les chiffres sont rangés dans l'ordre croissant (codes 48 à 57), et les 4 bits de poids faibles définissent la valeur en binaire du chiffre.*

## 2.2. Le transcodage

**Définition** L'opération de passage de la représentation d'une donnée dans un code à un autre s'appelle transcodage.

**Remarque** *Le passage d'un code à un autre est indispensable au moment de la communication entre deux ordinateurs qui ne possèdent pas le même type de codage de caractères.*

## 3. Représentation des données numériques

Avant qu'elles subissent un traitement, les données doivent subir une opération de codage qui permet de déduire leur représentation interne. Ensuite, les opérations demandées sont effectuées, généralement en arithmétique binaire. Une opération de décodage (passage de la représentation interne vers la représentation externe) est nécessaire pour les restituer vers l'extérieur.

Les données numériques sont de différents types :

- Nombres entiers positifs ou nul : 0, 1, 1254..
- Nombre entiers négatifs :-1, -1245...
- Nombre fractionnaires : 3.1457, -0.514
- Nombre en notation scientifique : 1.5 10<sup>7</sup>

### 3.1. Entiers positifs ou nuls

La représentation des nombres pose quelques problèmes car les ensembles de nombres sont tous infinis. Elle nécessite des champs de codes de longueurs variables. On ne peut tout de même pas attribuer à chacun un symbole différent. Il est indispensable d'adopter un alphabet (les chiffres) et des règles d'agencement qui permettent une correspondance univoque avec les nombres.

Il faut donc réutiliser les mêmes chiffres (l'alphabet numérique). La solution est de leur attribuer des valeurs différentes suivant leur position relative. Exemple : douze = 1 x dix + 2 et est noté en base DIX : 1 suivi de 2.

#### 3.1.1. Système de numération :

**Définition** Un système de numération est un ensemble de conventions qui permettent de former, énoncer et écrire des nombres.

**Remarque** *Un système de numération fait correspondre, à chaque un nombre N, une suite unique de symboles.*

**Définition** Une base dans un système de numération est le nombre de symboles à utiliser dans ce système pour représenter les nombres.

**Remarque** Dans une base  $p > 1$ , les nombres  $0, 1, \dots, p-1$  sont appelés chiffres.

**Théorème** Tout nombre  $N$  entier positif peut être représenté, dans la base  $p$ , par une expression de la forme :

$$N = a_n p^n + a_{n-1} p^{n-1} + \dots + a_1 p + a_0$$

avec  $a_i \in \{0, 1, \dots, p-1\}$  et  $a_n \neq 0$

**Remarque** La notation condensée  $N = (a_n a_{n-1} \dots a_1 a_0)_p$  est équivalente à la forme ci-dessus. On dit que  $(a_n a_{n-1} \dots a_1 a_0)_p$  est la représentation de  $N$  dans la base  $p$ .

$N =$	$a_n \times p^n +$	$a_{n-1} \times p^{n-1} +$	$\dots +$	$a_1 \times p^1 +$	$a_0 \times p^0$
Rang	$n$	$n-1$	$\dots$	$1$	$0$
Poids	$p^n$	$p^{n-1}$	$\dots$	$p^1$	$p^0$
Écriture	$a_n$	$a_{n-1}$	$\dots$	$a_1$	$a_0$

**Tableau 2.2. Représentation d'un entier naturel dans une base p.**

**Exemple** Le nombre 21 ( $p=10$ ,  $2 \times 10^1 + 1 \times 10^0$ ) est représenté en binaire ( $p=2$ ) par : 10101 soit  $1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ .

Ce qui s'écrit :  $21_{10} = 10101_2$ .



$21 =$	$1 \times 2^4 +$	$0 \times 2^3 +$	$1 \times 2^2 +$	$0 \times 2^1 +$	$1 \times 2^0$
Rang	$4$	$3$	$2$	$1$	$0$
Poids	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
Écriture	$1$	$0$	$1$	$0$	$1$

**Tableau 2.3. Représentation d'un entier naturel dans une base 2**

### 3.1.2. Les systèmes de numération les plus utilisés :

#### a. SYSTEME DECIMAL

Dans le système décimal ou en base DIX, l'idée clé est d'effectuer des groupements par DIX ou si nécessaire suivant les puissances successives de DIX. On comprend alors que l'opération qui permet cette écriture est la division euclidienne.

Par exemple, la division entière par dix d'un nombre inférieur à cent permet de trouver le nombre de groupes de dix, le reste étant nécessairement inférieur (strictement) à dix. On le note alors :

(nb de groupes de dix, les dizaines suivies du reste, les unités).

On a donc besoin, en système décimal, de dix symboles, les chiffres de 0 à 9 pour écrire tous les naturels.

**On dit que DIX est la BASE de numération alors choisie.**

Il s'agit d'un code PONDERE : chaque chiffre décimal acquiert un poids lié à sa position dans l'écriture du nombre.

LE POIDS VAUT :  $10^r$ , où r est le RANG du CHIFFRE (r=0 à droite, chiffre des unités).

Ainsi :

1997	=	$1 \times 10^3$	+ $9 \times 10^2$	+ $9 \times 10^1$	+ $7 \times 10^0$
N°RANG :		3	2	1	0 correspond à l'exposant.
POIDS :		1000	100	10	1 valeur du chiffre
ÉCRITURE		1	9	9	7

**b. CAS GENERAL : BASE B**

**Définition** Soit B un Naturel dans  $\mathbb{N}^* - \{1\}$ . On appelle système de numération pondérée à base B la donnée de B symboles différents appelés CHIFFRES, affectés à la notation des premiers naturels de 0 à B-1.

**Remarque** L'adoption de la règle d'écriture des naturels à partir de B par pondération suivant les puissances, définie par :

👉 le chiffre de droite, de rang 0, a le poids  $B^0=1$ .

👉 tout chiffre à gauche d'un chiffre I a un poids B fois supérieures à celui de I.

*Notation* : Pour éviter toute ambiguïté, on place le symbole B à droite des chiffres.

En hexadécimal (base seize), on met h (ou en Pascal \$ devant ), en binaire b, en décimal d.

**Exemple** Trouver l'écriture décimale du nombre :

$$123(8) = 1 \times 8^2 + 2 \times 8 + 3 = \dots$$

Conséquence : en particulier le nb B s'écrit 10, ce qui se lit "un, zéro dans la base B"

$$B = 10 (B) \qquad B^2 = \dots (B) \qquad \text{inversement } 1\ 000 (B) = 1 \times B0^3 = B^3$$

**c. Numération binaire**

➤ *Chiffre binaire* ou élément binaire ou bit :

Chaque information traitée par l'ordinateur est définie par un **ensemble fini d'informations** simples à deux états possibles, représentées par un **chiffre** 1 ou 0 : des **bits** (bit = contraction de *binary digit*).

- Avec 1 bit, on a 2 valeurs possibles : 0 et 1
- Avec 2 bits, on a 4 valeurs possibles : 00, 01, 10 et 11
- Avec 3 bits, on a 8 valeurs possibles : 000, 001, 010, 011, 100, 101, 110, 111
- ...

Quelles sont les avantages de la représentation binaire ?

👉 Elle est facile à réaliser techniquement, à l'aide de bistables (systèmes à 2 états d'équilibre).

👉 Les opérations fondamentales sont relativement simples à effectuer, sous forme de circuits logiques. En effet, l'arithmétique binaire peut être réalisée à partir de la logique symbolique à deux états (0,1).

➤ *Quartet* : nombre binaire de 4 bits

➤ *Octet* (BYTE) On définit en général les informations par des séquences de 8 bits appelés **octets** (*Bytes*).

1 octet = 8 bits => **2<sup>8</sup> états** (256 valeurs possibles)

En français, un octet se note parfois **o**, et en anglais **B** (Byte).

- **Ko** ou KB (ou K) = kilooctets (*KiloBytes*) : ( $2^{10}=1024$ ) octets

- **Mo** ou MB = mégaoctets (*MegaBytes*) :  $2^{20}$  octets ( $1024*1024=1.048.576$ )
- **Go** ou GB = gigaoctets (*GigaByte*) :  $2^{30}$  octets ( $1.048.576*1024=1.073.741.824$ ).

### 3.2. Les unités employées :

préfixe	symbole	valeur	valeur	valeur approximative	exemple d'utilisation
<b>Kilo</b>	<b>K</b>	$2^{10}$	1 024	$10^3$	un modem téléphonique débite à 56,6 Kbps
<b>Méga</b>	<b>M</b>	$2^{20}$	1 048 576 = 1 024 K	$10^6$	le bus d'un PC est à des centaines de MHz  Disquettes 3 <sup>1/2</sup> <b>HD</b> (Haute densité - <i>High Density</i> ) : 1.457.664 octets = 1,39 Mo
<b>Giga</b>	<b>G</b>	$2^{30}$	1 073 741 824 = 1 024 M = 1 048 576 K	$10^9$	les disques durs des PC font des dizaines de Go
<b>Téra</b>	<b>T</b>	$2^{40}$		$10^{12}$	Les grandes bases de données archivent des To
<b>Péta</b>	<b>Z</b>	$2^{50}$		$10^{15}$	le processeur Power pc adresse jusqu'à 4 Po
<b>Exa</b>	<b>E</b>	$2^{60}$		$10^{18}$	
<b>Zetta</b>	<b>Z</b>	$2^{70}$		$10^{21}$	
<b>Yotta</b>	<b>Y</b>	$2^{80}$		$10^{24}$	

**Tableau 2.4. Tableau des unités employées**

➤ **Tableau de conversion binaire - décimale**

$$0(2) = 0 \cdot 2^0 = 0(10)$$

$$1(2) = 1 \cdot 2^0 = 1(10)$$

$$10(2) = 0 \cdot 2^0 + 1 \cdot 2^1 = 2(10)$$

$$11(2) = 1 \cdot 2^0 + 1 \cdot 2^1 = 3(10)$$

<i>Bi na ire</i>	<i>Décimale</i>
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9

**Tableau 2.5. Tableau de correspondance binaire- décimale**

**d. Numération octale**

Base 8  $\Rightarrow$  huit symboles (chiffres) qui sont 0, 1, 2, 3, 4, 5, 6 et 7.

*Exemple :*

$$53(8) = 5 \cdot 8^1 + 3 \cdot 8^0 = 43(10).$$

➤ **tableau de conversion octale - décimale:**

<i>Octale</i>	<i>Décimale</i>
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
10	8
11	9
.	.

**Tableau 2.6. Tableau de correspondance octale- décimale**

**e. BASES SUPERIEURES À 10**

D'autres symboles en supplément des chiffres décimaux 0,1,2,3...9 sont nécessaires :

Base douze (*duodécimale*) : Il comporte douze symboles qui sont :

0, 1, 2, 3, 4, ..., 9, 10, 11.

On rencontre ici une confusion entre les symboles (chiffres) 10 (dix) et 11 (onze) qui sont respectivement 10 et 11 en décimal et les nombres 10 (un zéro) et 11 (un un) dans la base 12 qui sont respectivement 12 et 13 en décimal.

Pour éviter cette confusion on représente le chiffres 10 et 11 par les lettres A et B. Ce qui donnera :

$$A(12) = 10(10) ; 10(12) = 12(10).$$

$$B(12) = 11(10) ; 11(12) = 13(10).$$

**Base seize (*hexadécimale*) :**

On a 16 symboles qui sont : 0, 1, 2, 3, 4, ....., 9, A, B, C, D, E, F  
 dix onze douze treize quatorze quinze.

$$B7(16) = 11 \cdot 16^1 + 7 \cdot 16^0 = 183(10)$$

➤ **Table de correspondance :**

<i>Binaire</i>	<i>Hexadécimale</i>	<i>Décimale</i>
----------------	---------------------	-----------------

0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

**Tableau 2.7. Tableau de correspondance binaire – hexadécimale - décimale**

👉 Son intérêt en informatique est de permettre TRES FACILEMENT d'obtenir une écriture condensée de l'écriture binaire ; cela résulte simplement de  $16 = 2^4$ .

### 3.3. Changement de base

#### 3.3.1. Passage d'une base p à la base décimale (10)

La conversion se fait en additionnant les puissances de **p** correspondant aux bits de valeur 1.

$$N_p = (a_n a_{n-1} \dots a_1 a_0)_p = a_n p^n + a_{n-1} p^{n-1} + \dots + a_1 p + a_0$$

#### Exemples

$$(AE)_{16} = 10 * 16^1 + 14 * 16^0 = (174)_{10}$$

$$(0101)_2 = 2^2 + 2^0 = (5)_{10}$$

**Définition** Soit un nombre entier N représenté dans la base 10. On effectue des opérations de division euclidienne successives de ce nombre par la base p jusqu'à l'obtention d'un quotient nul. Le nombre est obtenu en lisant les restes du dernier vers le premier.

#### Exemples

Soit le nombre  $Y = (2596)_{10}$ , on veut l'exprimer en hexadécimal

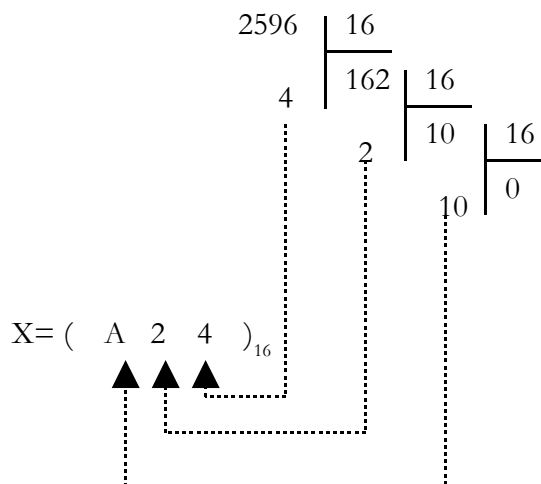


Figure 2.5. Exemple de passage de la base décimale vers la base hexadécimale

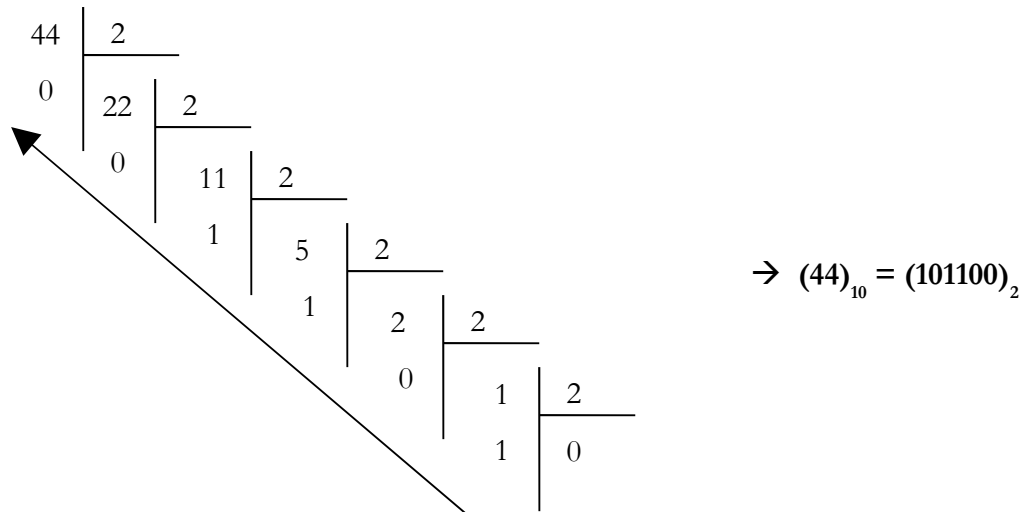


Figure 2.6. Exemple de passage de la base décimale vers la base binaire

**3.3.2. Passage de la base 8 (respectivement 16) vers la base 2 :**

La conversion correspond à l'éclatement de chaque chiffre octal (respectivement hexadécimal) en son équivalent binaire sur 3 (respectivement 4) bits.

**Exemples**

Soit le nombre octal  $X = (175)_8 = ( ? )_2$  :

Pour trouver l'équivalent binaire de nombre octal, il suffit de trouver l'équivalent binaire de chaque octal

1	7	5
0 0 1	1 1 1	1 0 1

Figure 2.7. Exemple de passage de la base octale vers la base binaire.

$(6F5)_{16} = (0110 1111 0101)_2$

$(135)_8 = (001 011 101)_2$

**3.3.3. Passage de la base 2 vers la base 8 (respectivement 16)**

La conversion correspond à effectuer un remplacement, de droite à gauche, de 3 (respectivement 4) bits par le chiffre octal (respectivement hexadécimal) correspondant. Si le nombre de bits n'est pas multiple de 3 (respectivement 4) compléter à gauche par des zéros.

**Exemples**

Soit le nombre  $X = (101000100100)_2 = ( ? )_{16}$

On regroupe les bits 4 à 4, en commençant par la droite. Si le dernier groupe n'a pas 4 bits, ajouter des zéros. On remplace chaque groupe de 4 bits (binaire) par son équivalent hexadécimal. En Conclusion  $X = (101000100100)_2 = (A24)_H$

1	0	1	0	0	0	1	0	0	1	0	0
A				2				4			

Figure 2.8. Exemple de passage de la base 2 vers la base 16.

$$(0110\ 1111\ 0101)_2 = (6F5)_{16}$$

$$(01\ 011\ 101)_2 = (135)_8$$

### 3.4. L'arithmétiques binaires

#### 3.4.1. Opérations de base

L'arithmétique binaire possède les mêmes règles de calculs que l'arithmétique décimale. La seule différence entre eux, c'est qu'elles possèdent des tables d'opérations élémentaires différentes.

Addition	Soustractio n	Multiplication	Division										
$0 + 0 = 0$ $0 + 1 = 1$ $1 + 0 = 1$ $1 + 1 = 1$ avec une retenue	$0 - 0 = 0$ $0 - 1 = 1$ si déduction possible $1 - 0 = 1$ $1 - 1 = 0$	$0 * 0 = 0$ $0 * 1 = 0$ $1 * 0 = 0$ $1 * 1 = 1$	<table border="1"> <thead> <tr> <th></th> <th>Quotient</th> </tr> </thead> <tbody> <tr> <td>0 : 0</td> <td>Impossible</td> </tr> <tr> <td>0 : 1</td> <td>0</td> </tr> <tr> <td>1 : 0</td> <td>Impossible</td> </tr> <tr> <td>1 : 1</td> <td>1</td> </tr> </tbody> </table>		Quotient	0 : 0	Impossible	0 : 1	0	1 : 0	Impossible	1 : 1	1
	Quotient												
0 : 0	Impossible												
0 : 1	0												
1 : 0	Impossible												
1 : 1	1												

Tableau 2.8. Opérations élémentaires d'arithmétique binaire

#### Exemples

- a. Soit à effectuer l'addition binaire suivante :  $11011 + 10011$

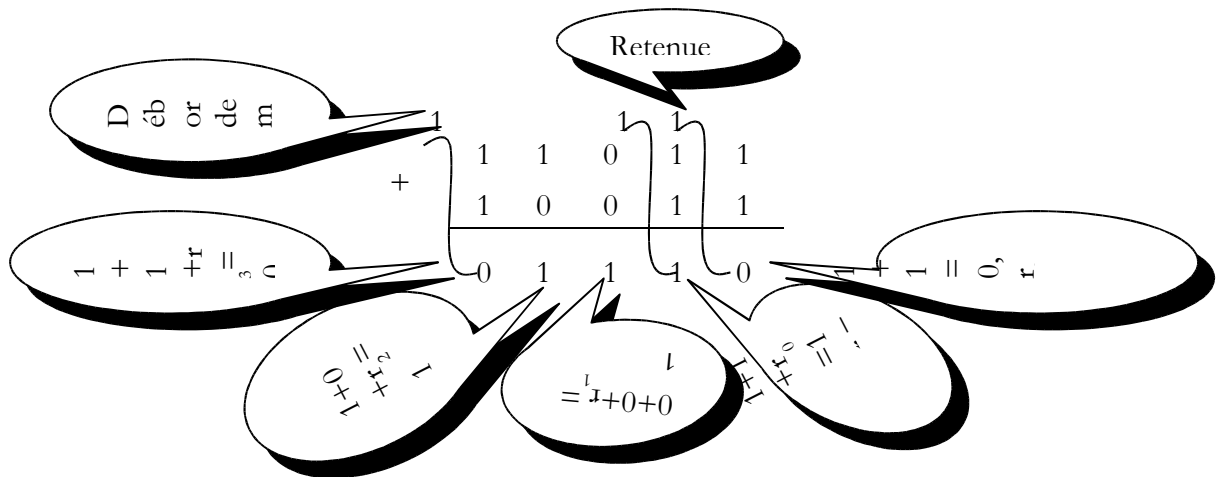


Figure 2.9. Premier exemple d'addition binaire

- b. Addition binaire sur 4 bits

	9
+	
	8
=	17

	1	0	0	1	
+					
	1	0	0	0	
=	1	0	0	0	1

Figure 2.10. Deuxième exemple d'addition binaire

c. Soit à effectuer l'addition binaire suivante : 01110 – 10011

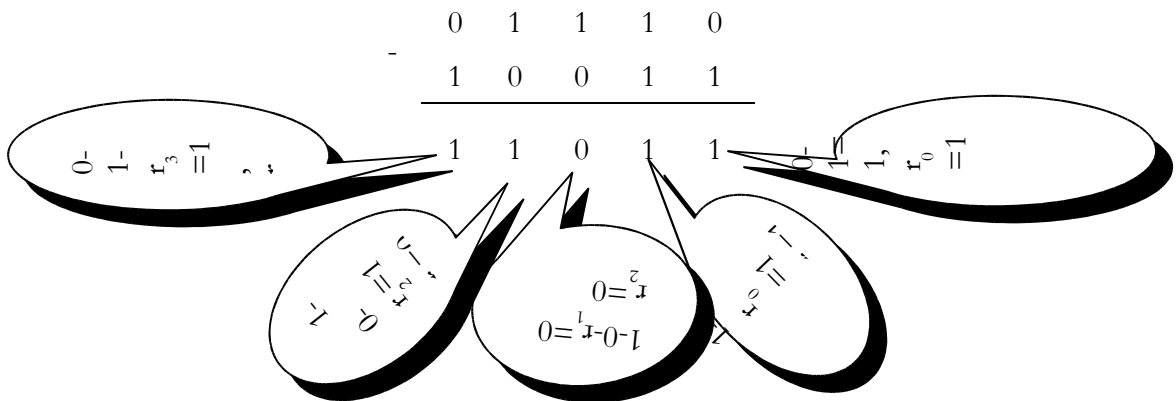


Figure 2.11. Premier exemple de soustraction binaire

d. Soustraction sur 4 bits

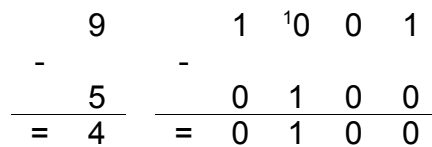


Figure 2.12. Deuxième exemple de soustraction binaire

e. Soit à trouver le produit des deux nombres binaires 1101 et 101

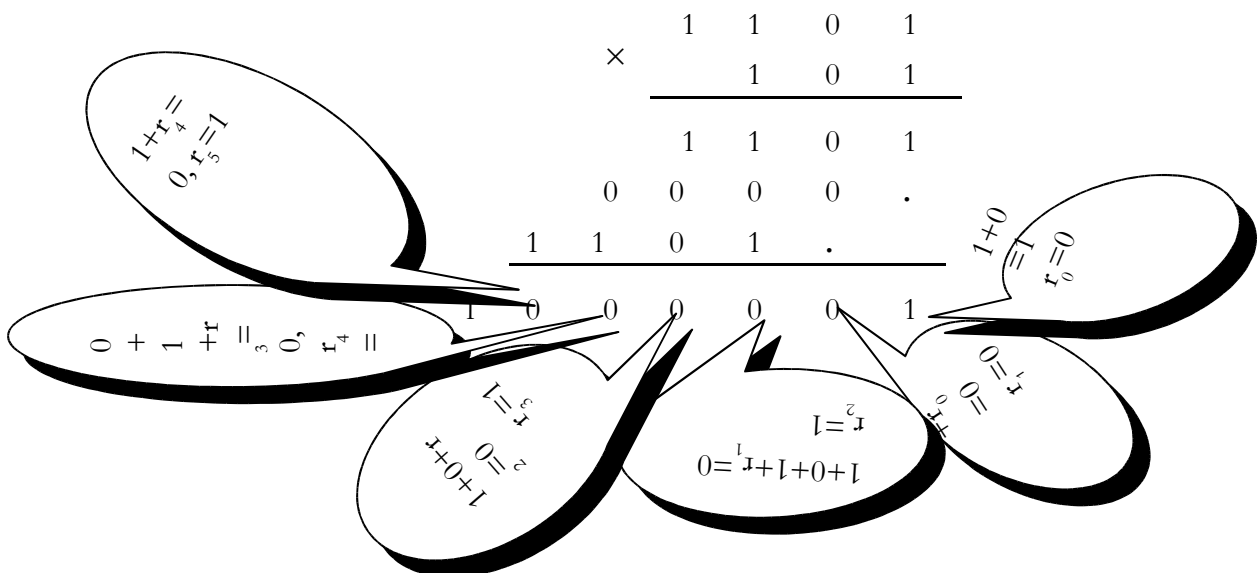


Figure 2.13. Premier exemple de multiplication binaire

f. Multiplication sur 6 bits



### 3.5. Entiers négatifs

Les entiers négatifs peuvent être codés avec l'une des trois méthodes suivantes :

- Signe et valeur absolue;
- Complément logique (complément à 1);
- Complément arithmétique (complément à 2).

#### 3.5.1. Signe et valeur absolue

Les nombres sont codés de la façon suivante +/- valeur absolue ; on sacrifie un bit pour représenter le signe. Normalement 0 est le signe positif, 1 est le signe négatif. Ainsi, avec k bits, on peut coder N entiers positifs et négatifs tel que :  $-(2^{k-1} - 1) \leq N \leq 2^{k-1} - 1$ .

**Remarque** Cette méthode a les inconvénients suivants :

- 0 a deux représentations 0000000 et 1000000 (k=7)
- Les tables de multiplication et d'addition sont compliquées à cause du bit de signe qui doit être traité à part.

#### Exemples

$$\begin{array}{r}
 7 \qquad \qquad 0 \ 1 \ 1 \ 1 \\
 + \qquad \qquad + \\
 -5 \qquad \qquad 1 \ 1 \ 0 \ 1 \\
 \hline
 2 \qquad \qquad 1 \ 0 \ 1 \ 0 \ 0
 \end{array}$$

**Figure 2.17. Réaliser d'une manière classique L'addition en signe et valeur absolue est erronée**

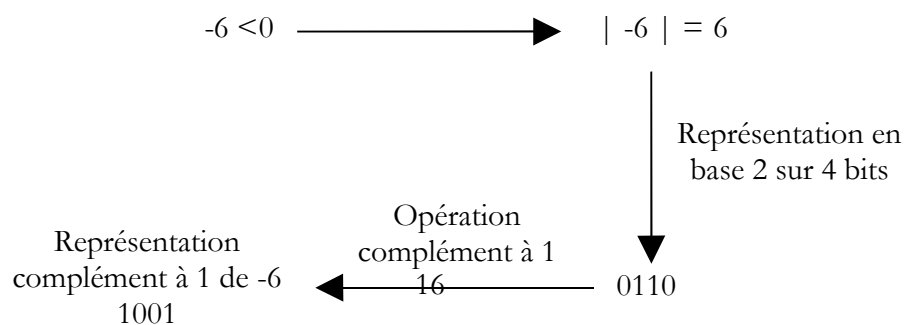
#### 3.5.2. Complément logique (complément à 1)

On calcule le complément logique en remplaçant pour les valeurs négatives, chaque bit à 0 par 1 et vice versa. Pour représenter un nombre négatif, on représente sa valeur absolue en binaire puis on détermine son complément à 1. Il y a toujours un bit consacré pour le signe et le nombre négatif se voit toujours accordé un 1 dans son dernier bit à gauche. Avec k bits, on peut coder N entiers positifs et négatifs tel que  $-(2^{k-1} - 1) \leq N \leq 2^{k-1} - 1$ .

- Remarque**
- Cette méthode a l'inconvénient suivant :
  - 0 a deux représentations 0000000 et 1111111 (k=7)
  - Mais elle a l'avantage suivant :
  - L'addition devient simple. En effet, si une retenue est généré par le bit de signe alors elle doit être ajouté au résultat obtenu.

#### Exemples

Représentation de -6 en complément à 1 sur 4 bits



Représentation de 7 en complément à 1 sur 4 bits

$$7 \geq 0 \longrightarrow \begin{array}{l} \text{Représentation en} \\ \text{complément à 1 de} \\ 7 \end{array} = \begin{array}{l} \text{Représentation en} \\ \text{base 2 sur 4 bits} \end{array} = 0111$$

Figure 2.18. Exemple de représentation de nombres en complément à un

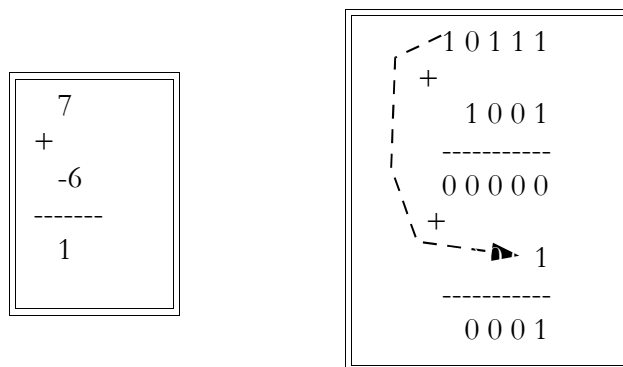


Figure 2.19. Exemple d'addition en complément à un

### 3.5.3. Complément arithmétique (complément à 2)

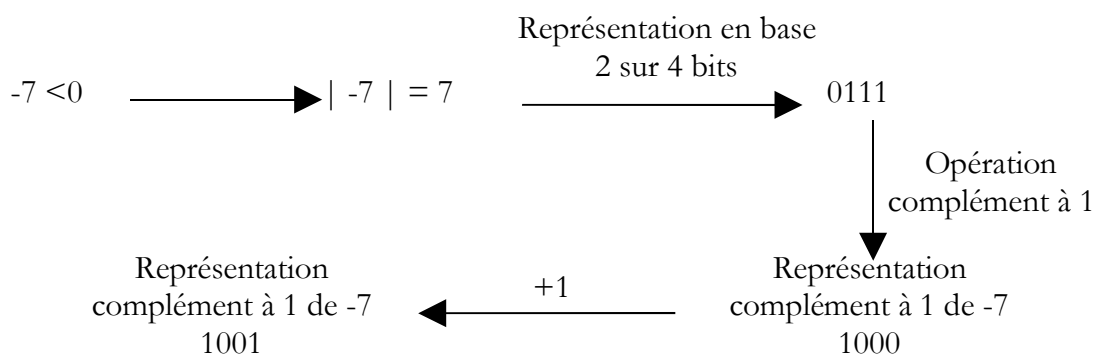
On obtient le complément à 2 d'un nombre en remplaçant chaque bit par son complément à 1 puis on ajoute 1 au résultat obtenu. On représente un nombre négatif en considérant sa valeur absolue et en la représentant en complément à 2.

**Remarque** Cette méthode a les avantages suivants :

- une seule représentation pour 0.
- les additions deviennent de plus en plus simples.

#### Exemples

Représentation de -7 et de 7 en complément à 2 sur 4 bits



$$7 \geq 0 \longrightarrow \begin{array}{l} \text{Représentation} \\ \text{complément à} \\ \text{2 de 7} \end{array} = \begin{array}{l} \text{Représentation} \\ \text{en base 2 sur 4} \\ \text{bits} \end{array} = 0111$$

Figure 2.20. Représentation de -7 et de 7 en complément à 2 sur 4 bits

+7
+
-7
-----
1

<sup>1</sup> 0111
+
1001
-----
0000

Figure 2.21. Exemple d'addition en complément à 2

### 3.6. Les nombres fractionnaires

Les nombres fractionnaires sont les nombres qui comportent une partie strictement inférieure à 1. Dans la machine, tout nombre est codé avec un nombre fini de chiffres ; il n'est pas possible de représenter tout les rationnels, et à fortiori tout les réel. On utilise une représentation prenant en compte la virgule dans les nombres.

#### 3.6.1. Changement de base

##### 1. Passage d'une base P à la base Décimale

Pour la partie purement fractionnaire, le passage se fait par l'addition de puissances négatives de p.

$$N_p = (a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-m})_p = a_n p^n + \dots + a_1 p + a_0 + a_{-1} p^{-1} + \dots + a_{-m} p^{-m}$$

##### Exemples

Représentation de 1.01

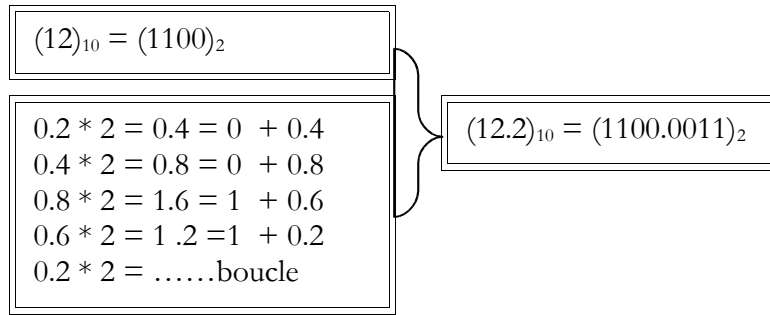
$$(1.01)_2 = 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = (1.25)_{10}$$

##### 2. Passage de la base Décimale à une base P

La conversion se fait, ici par des multiplications successives par la base des nombres purement fractionnaires. Cet algorithme doit s'arrêter dès qu'on obtient une partie fractionnaire nulle ou bien quand le nombre de bits obtenues correspond à la taille de la mémoire dans lequel on va stocker la valeur. Le nombre, correspondant à la partie fractionnaire, cherché s'obtient en lisant les parties entières de la première vers la dernière obtenue.

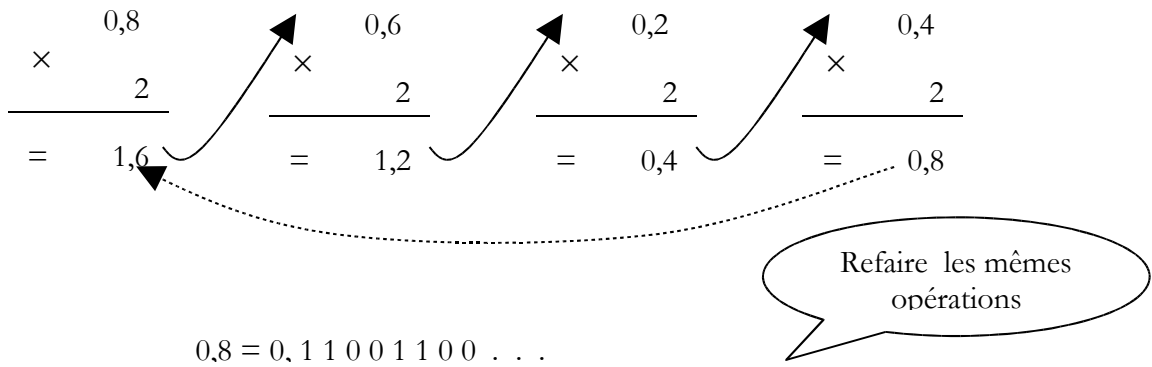
##### Exemples

Représentation de  $(12.2)_{10} = (?)_2$  : On prend la partie fractionnaire



**Figure 2.22. Exemple de conversion d'un nombre fractionnaire en base 10 vers la base 2**

Représentation de  $(0.8)_{10} = (?)_2$



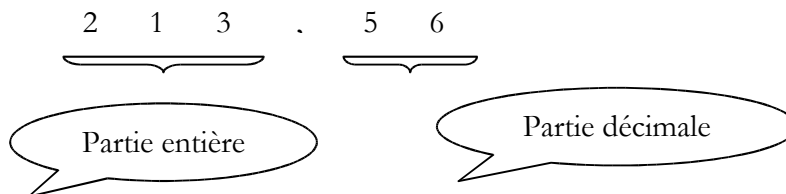
**Figure 2.23. Exemple de conversion d'un nombre fractionne de la base décimale vers la base binaire**

Dans les nombres fractionnaires, on distingue la représentation en virgule fixe et la représentation en virgule flottante.

**3.6.2. Représentation en virgule fixe**

Les premières machines utilisaient une représentation en virgule fixe ou chaque nombre était séparé en deux parties contenant les chiffres avant et après la virgule. Les ordinateurs n'ont pas de virgule, on traite les nombres fractionnaires comme des entiers avec une virgule virtuelle gérée par le programmeur. Ce dernier doit donc connaître et faire évoluer, au cours des opérations la place de la virgule. (De façon à conserver le maximum de chiffres significatifs).

**Exemple**



**Figure 2.24. Exemple de nombre fractionne représenté en virgule fixe**

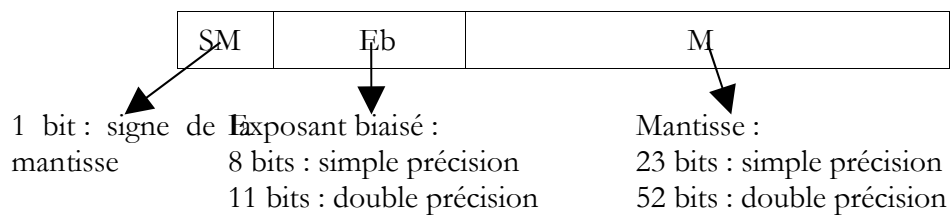
### 3.6.3. Représentation en virgule flottante

#### a. Norme IEEE 754

Elle consiste à représenter les nombres  $N$  sous la forme :  $N = M * B^E$

- $M$  : mantisse
- $B$  : base (2, 8, 10, 16...)
- $E$  : exposant

Un flottant est stocké selon la norme IEEE 754, analogue à la représentation scientifiques des nombres fractionnaires (tel que  $2.5 * 10^{-5}$ ).



**Figure 2.25. Représentation en virgule flottante.**

Dans ce cas figure :

- L'exposant est un entier sans signe mais biaisé de 127. ( $E = E_b - 127$ ). Avec  $E$  est compris entre -126 et 127. ( $E=0$  : cas non normalisé).
  - L'exposant maximum est 127 et non pas 128 qui est réservé pour des configurations spéciales
- 0 1111 1111 000...000 = +INF  
 1 1111 1111 000...000 = -INF

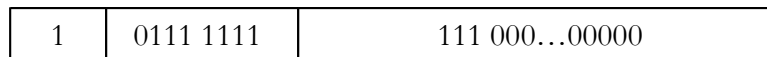
On l'utilise pour signaler des erreurs (exemple : division par 0).

- La mantisse est un nombre purement fractionnaire (n'ayant pas de chiffres significatifs à gauche de la virgule). Celle-ci est normalisé au sens où elle est toujours de la forme 0.1....C'est à dire que le premier bit à droite de la virgule est toujours à 1.

#### Exemples

Représentation de -1.75 en IEEE754

Traduire 1.75 en base 2 :  $1.75 = (1.11)_2$   
 Ecriture normalisée :  $(1.11)_2 = (0.111 * 2^{-1})_2$   
 $E_b = 0 + 127$



**Figure 2.26. Exemple de représentation d'un nombre en virgule flottante**

### 3. Opérations sur les nombres en virgule flottante (Norme IEEE754)

Pour effectuer en virgule flottante les opérations de calcul, il faut suffit de suivre les règles suivantes :

- Addition : pour l'addition et la soustraction, il faut que les exposants aient la même valeur
- multiplication : on additionne les exposants et on multiplie les mantisses.

- division : on soustrait les exposants et on divise les mantisses puis on normalise si c'est nécessaire.
- soustraction : elle s'effectue comme l'addition sauf que l'on doit faire la soustraction et non plus l'addition des mantisses. Pour chacune des opérations arithmétiques, il peut être nécessaire de tronquer la mantisse (ne pas prendre tous les chiffres après la virgule) soustrait les exposants et on divise les mantisses puis on normalise si c'est nécessaire.

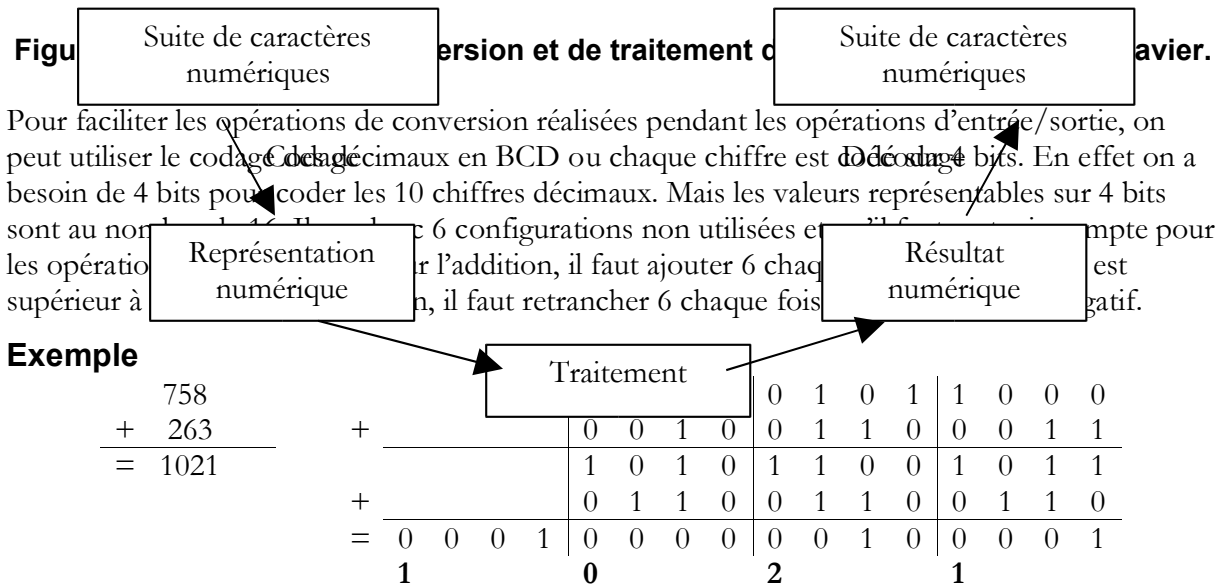
**Exemples**

Exprimer en virgule flottante

- $0.3 \times 10^4 + 0.998 \times 10^6$
- Dénormaliser :  $0.3 \times 10^4 = 0.003 \times 10^6$
- Additionner les mantisses :  $0.003 + 0.998 = 1.001$
- Normaliser :  $1.001 \times 10^6 = 0.1001 \times 10^7$
- $0.2 \times 10^{-3} + 0.3 \times 10^7$
- $-3 + 7 = 4$
- $0.2 \times 0.3 = 0.06$
- Résultat normalisé :  $0.06 \times 10^4 = 0.6 \times 10^3$

**3.6.4. Les nombres codés en BCD :**

Les données numériques sont entrées à l'unité centrale à partir du clavier sous forme d'une suite de caractères. Elles sont ensuite converties en leur représentation numérique. Par exemple, pour saisir le nombre 91, l'utilisateur doit entrer les caractères : '9' et '1'. Ces caractères sont convertis ensuite sous forme d'une représentation numérique. Après avoir effectué un traitement sur les nombres dans leur représentation numérique, il faut convertir le résultat en une suite de caractères qui seront affichés sur le moniteur de l'ordinateur.



## 4. Représentation des instructions

Une instruction machine renseigne le processeur sur la nature de l'opération à exécuter et sur les données qui vont participer à l'exécution de l'opération.

Représentée sur une taille fixe, Elle est composée de plusieurs parties de taille fixe (champs) qui sont les suivantes :

- Le code de l'opération à effectuer ;
- Les opérandes (les données) impliqués dans l'opération.

### 4.1. Code opération

**Définition** Il représente un nombre fixe de bits. Chacune parmi les opérations de base que l'ordinateur est capable d'exécuter se voit attribuée une suite binaire différente des autres (d'où l'appellation code opération).

### 4.2. Les opérandes

**Définition** ▪ Elles se partagent généralement d'une façon équitable le reste de l'instruction (taille de l'instruction – taille code opération). Elles représentent en binaire, les valeurs des données concernées par cette opération, ou bien leur emplacement dans la mémoire centrale (appelé aussi adresse).

### 4.3. Les machines selon le nombre d'adresse supportées

Le nombre maximum d'opérandes peut changer, d'un ordinateur à un autre. Selon le nombre d'opérandes, on distingue les ordinateurs avec des :

- Instructions à une seule opérande appelé aussi à une seule adresse (car généralement l'opérande est une adresse).
- Instructions à deux adresses.
- Instructions à trois adresses.

Code opération	Opérande 1	Opérande 2
M bits	N bits	N bits

**Figure 2.29. Cas d'un ordinateur avec des instructions à 2 adresses.**

**Remarque** Avec des processeurs à trois adresses, l'exécution est plus rapide, car il permet d'effectuer l'opération demandée et de ranger en même temps, le résultat dans l'adresse fournie dans la troisième opérande.

#### Exemple

Soit une instruction du genre :  $A \leftarrow B + C$  (où A, B et C sont des variables qui correspondent à des emplacements (mots) dans la mémoire centrale :

Adresse (binaire)	MC	Variable
0000	Mot 1	A
0001	Mot 2	B
0010	Mot 3	C
...	...	...

En supposant que l'ordinateur peut exécuter des instructions avec le format ci dessous et que l'addition est codée par 0101

Code opération	Opérande 1	Opérande 2	Opérande 3
4 bits	4 bits	4 bits	4 bits

Alors cette instruction sera représentée comme suit

Code opération	Opérande 1	Opérande 2	Opérande 3
0101	0001	0010	0000
addition	Variable B	Variable C	Variable A

**Figure 2.30. Figure 3.28. Exemple de représentation d'une instruction à 2 adresses Application**

Dans le cas de la figure précédente déterminer le nombre de valeurs possible pour le code opération, ainsi que les opérandes

## 5. En résumé

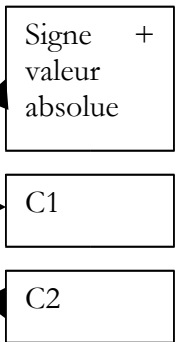
**Figure 2.31. Récapitulatif des différentes représentations relatives aux données**

Instructions :  
représentation propre à chaque type  
machine

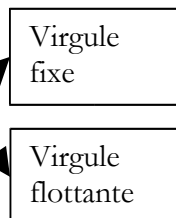
es

entiers positifs  
(on directe  
→ Binaire)

entiers négatifs



fractionnaires



## 6. Série d'exercices

### Exercice 1

Reliez par des flèches les mots des colonnes gauches et droites qui peuvent être en relation.

Bit
1 kilo octet
Une Instruction définie
Les Données sont identifiées par
Code opération
01110
Un entier est codé sur
ASCII code les alphanumériques sur
Le MAX-ENTIER est

8192 bits
Identifie une opération et une seule
8 bits
Leurs adresses dans la mémoire
La plus élémentaire des informations
Est un nombre binaire
Code opération et emplacement des données dans la mémoire centrale
2 octets
32767

### Exercice 2

11001 21011 1o01 50841 0xe 0377 0X5FF IF 01AK1 ISET EX1 eeeeE abcdef

- Parmi ces suites de chiffres, quelles sont celles qui peuvent représenter un nombre en base 2, 8, 10 ou 16 ?
- Donner la plus petite base dans laquelle chaque nombre peut être représenté.

### Exercice 3

Convertir en base 2, 8 et 16 les nombres suivants : 100, 125, 400, 666, 1999, 2002

### Exercice 4

Compléter le tableau suivant :

Base 2	Base 8	Base 10	Base 16
0			
	1		
			1F0
	101		
1101101			
	7777		
			3AD

*Exercice 5*

Compléter le tableau suivant :

Base 2	Base 8	Base 10	Base 16
0,1			
	0,05		
			0,E01
	0,401		
0,1011			
	0,75		
			0,BAC

*Exercice 6*

Compléter le tableau suivant :

Base 2	Base 8	Base 16
10111		
		1DF
	726	
0,110		
		BAC
101,101111		

*Exercice 7*

Utiliser la méthode des divisions successives pour convertir en hexadécimal, en octal, puis en binaire les nombres suivants écrit dans la base 10 : 23 255 35.55

*Exercice 8*

Effectuez les opérations suivantes dans la base 2 en supposant que les nombres sont représentés sur 1 octet et que le bit le plus à gauche est réservé pour indiquer le signe du nombre.

20 – 22	1A – 1B	-5 – 10	5 + 7	(17) <sub>8</sub> - (7) <sub>8</sub> .
---------	---------	---------	-------	--

*Exercice 9*

Utiliser le codage en valeur absolue puis en complément à 1 puis en complément à 2 pour coder sur 8 bits (si possible) puis sur 16 bits les entiers relatifs suivants (donnés en base 10) :

+1 -1 +127 -128 -99 -136 +1024 +32769 -32768 +32767

*Exercice 10*

Donner la valeur décimale de 10110111 dans les codages « entiers naturels » et « entiers relatifs en complément à 2 » et « entiers relatifs en complément à 1 »

*Exercice 11*

Calculer (en complément à 2 sur 8 bits) les additions suivantes :

00101101 + 01101111 ; 11111111 + 11111111 ; 00000001 + 11111111 ; 11110111 + 11101111

En déduire les règles de dépassement de capacité en complément à 2.

*Exercice 12*

En supposant les nombres représentés sur 8 bits, effectuer les opérations suivantes :  $377_8 + 001_8$  et  $177_8 + 200_8$ , en complément à 1 et en complément à 2. Convertir le résultat en décimal.

*Exercice 13*

Coder les réels suivants (sur 32 bits) selon la norme IEEE 754 :

1 2 4 9 1,5 -1 -2 6,125 5/32 -5/32

*Exercice 14*

Trouver le plus grand et le plus petit réel représentable avec la norme IEEE 754 simples précisions

*Exercice 15*

Convertir en décimal les nombres hexadécimaux réels donnés ici au format IEEE 754-32 bits

42<sup>E</sup>48000 3F880000 00800000 C7F00000

*Exercice 16*

En utilisant la table ASCII, coder les chaînes suivantes :

'fin du td' '+128' '-255'

*Exercice 17*

On se place dans le cas d'un ordinateur qui exécute des instructions qui ont le format suivant :

0	7	8	15
Code opération		@ opérande	

Répondre aux questions suivantes :

- c. combien d'opérations peut exécuter cet ordinateur ?
- d. en supposant que l'opérande contient une valeur d'une donnée, quel est le nombre possible de valeurs à traiter ?

