# Systèmes de Numération & Codage

*Objectif :*

*L'électronicien est amené à manipuler des valeurs exprimées dans différentes bases (notamment avec les systèmes informatiques). Il est essentiel de posséder quelques notions sur les systèmes de numération en général et sur les systèmes binaire et hexadécimal en particulier.*

## 1/ Quelques définitions :

**DIGIT** : Contraction de "digital unit" unité digitale. Un digit est un élément d'information numérique de base quelconque.

ex : Les nombres 1644 (base 10) et A84F (base 16) sont constitués chacun de 4 digits.

**POIDS D'UN DIGIT :** La valeur de chaque digit dépend de sa position. A chaque rang (position), est affecté un poids. Les positions des digits d'un nombre écrit en base B ont pour poids des puissances de B. (voir § suivant)

**BIT** : Contraction de "binary digit" digit binaire. Un bit ne peut prendre que deux états 0 ou 1.

ex : le nombre binaire 10100101 est constitué de 8 bits.

**MSD** : C'est le digit le plus significatif, de poids le plus fort (Most Significant Digit).

ex : pour le nombre A4F5, le MSB est un

**LSD** : C'est le digit le moins significatif, de poids le plus faible (Least Significant Digit).

ex : pour le nombre A4F5, le LSB est un

**MOT** : Un MOT est l'association (concaténation) de plusieurs digits ou bits (peut être aussi appelé courant un « nombre »)
-> un mot de 4 bits s'appelle un quartet; ex : 1010
-> un mot de 8 bits s'appelle un octet; ex : 1011 0110

**BASE** : Un nombre est écrit en base **B**, chacun de ses digits peut être écrit avec **B** symboles différents :

Valeurs en base 10 [*10 symboles*]: 0 1 2 3 4 5 6 7 8 9

Symboles en base 16 [*16 symboles*]: 0 1 2 3 4 5 6 7 8 9 A B C D E F

**CAPACITE DE COMPTAGE :** Avec N digits écrits en base B, on peut compter de 0 à $\mathbf{B}^{\mathbf{N}}$-1, soit $\mathbf{B}^{\mathbf{N}}$ nombres différents.

ex1 : avec un nombre de **3** digits en base **10,** on peut compter de 0 à (**=** ), soit nombres différents.
ex2: avec un nombre de 4 digits en base 2, on peut compter de 0 à (**=** ). soit nombres différents.
2/ Expression générale d'un nombre entier positif N (décomposition) :

De nombreux systèmes de numération sont utilisés en électronique numérique. Les plus courants sont les systèmes de numération suivants :

- Binaire (Base 2)
- Décimal ( Base 10)
- Hexadécimal (Base 16)

- Octal (Base 8)

### 2.1/ Le système de numération « Décimal »

Le système de numération que nous employons couramment utilise **10** chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

On l'appelle pour cela "système décimal" ou système à base **10**.

Dans ce système, un nombre peut être décomposé en puissance de **10**.

**Par exemple décomposons le nombre 546 :**

546 =

- Le digit « 6 », situé au premier rang à partir de la droite a une valeur de 6

- Le digit « 4 », situé au deuxième rang a une valeur de 40.

- Le digit « 5 », situé au troisième rang a une valeur de 500.

Ainsi, chaque digit a un "poids" différent selon son rang :

- au premier rang **(rang de niveau 0)** : le poids est de 1 (ou $10^0$),

- au deuxième rang **(rang de niveau 1)** : le poids est de 10 (ou $10^1$),

- et au troisième rang **(rang de niveau 2)** le poids est de 100 (ou $10^2$).

Le poids est la puissance nième de 10 ( $10^n$ ) si on numérote les rangs de droite à gauche et en commençant par le **rang n° 0.**

**Exercice :** Décomposer les nombres suivants sous forme de puissance de 10 :

**28** =

**4509** =

**60123** =

### 2.2/ Généralisation : Décomposition d'un nombre

Les nombres tels que nous les utilisons sont, en réalité, une convention d'écriture. Tout nombre entier positif peut s'écrire sous la forme d'un polynôme arithmétique.

$$N = a_n \times B^n + a_{n-1} \times B^{n-1} + \ldots\ldots + a_1 \times B^1 + a_o \times B^0$$

où **B est la base**, **a** est le chiffre de rang **n** et **n** représente le poids.

Dans la **base B**, on a besoin de **B symboles** pour écrire tous les nombres.

### 2.3/ Les autres bases de numération utilisées

A la place du décimal, nous pouvons utiliser la numération binaire, octale ou l'hexadécimale :

· **La base 2** (binaire) est employée pour traduire les états d'un système logique
[0 ou 1, tout ou rien, juste ou faux…]

· **La base 8** (octal) autrefois très utilisée, elle tend aujourd'hui à disparaître au profit de la base 16 suite à l'évolution technologique des composants (16 bits et +)

· **La base 16** (hexadécimal) est apparue avec la logique microprogrammée et les microprocesseurs. Elle

permet de traduire plus facilement un nombre binaire et autorise une représentation plus conviviale des grands nombres.

· La base 10 (décimal) est universellement employée par l'homme depuis qu'il sait compter sur ses doigts (10 doigts…)

**Récapitulatif des bases :**

| Base | Système | Nbre de symboles | Symboles utilisés | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | Binaire | | | | | | | | | | | | | | | | | |
| 8 | Octal | | | | | | | | | | | | | | | | | |
| 10 | Décimal | | | | | | | | | | | | | | | | | |
| 16 | Hexadécimal | | | | | | | | | | | | | | | | | |

Il est important de savoir convertir un nombre exprimé dans une base vers une autre.

**Tableau de correspondance entre les différentes bases pour les premières valeurs :**

| Base 2 | Base 8 | Base 16 | Base 10 |
|---|---|---|---|
| 0000 0000 | | | |
| 0000 0001 | | | |
| 0000 0010 | | | |
| 0000 0011 | | | |
| 0000 0100 | | | |
| 0000 0101 | | | |
| 0000 0110 | | | |
| 0000 0111 | | | |
| 0000 1000 | | | |
| 0000 1001 | | | |
| 0000 1010 | | | |
| 0000 1011 | | | |
| 0000 1100 | | | |
| 0000 1101 | | | |
| 0000 1110 | | | |
| 0000 1111 | | | |
| 0001 0000 | | | |
| 0001 0001 | | | |
| 0001 0010 | | | |
| 0001 0011 | | | |
| 0001 0100 | | | |

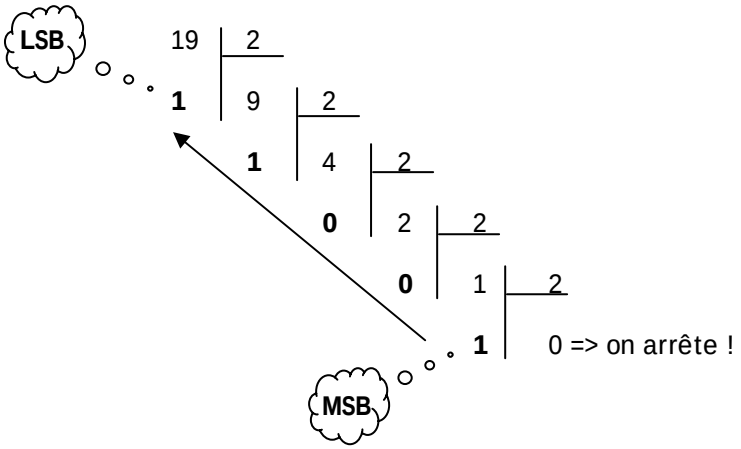
## 3 / Conversion d'une base dans une autre (transcodage)

### 3.1 / Conversion d'un nombre en <u>décimal</u> vers son équivalent en <u>binaire</u> [$(N)_{10}$ -> $(N)_2$]

La méthode consiste à répéter la division par 2 du nombre décimal à convertir et au report des restes jusqu'à ce que le quotient soit 0. Le nombre binaire résultant s'obtient en écrivant le premier reste à la position du bit de poids le plus faible (LSB = Least Significant Bit) et le dernier à la position du bit de poids le plus fort (MSB = Most Significant Bit).

**Exemple :** conversion du nombre décimal 19 en binaire

$(19)_{10}$ = (.....)$_2$ ?




D'où $(19)_{10}$ = (**10011**)$_2$

**Exercice :** Quel est le code binaire correspondant à $(65)_{10}$ et $(41)_{10}$ ?

$(65)_{10}$ = (  )$_2$

$(41)_{10}$ = (  )$_2$

### 3.2/ Conversion d'un nombre en binaire vers son équivalent en décimal [$(N)_2$ -> $(N)_{10}$]

Il s'agit ici d'appliquer la formule donné au paragraphe 2.2 en prenant B= 2.

$(\mathbf{1101})_2 = \mathbf{1}x2^3 + \mathbf{1}x2^2 + \mathbf{0}x2^1 + \mathbf{1}x2^0$

$= \mathbf{1}x8 + \mathbf{1}x4 + \mathbf{0}x2 + \mathbf{1}x1 = (13)_{10}$

Table des puissances

| Puissance de 2 | Puissance de 8 | Puissance de 10 | Puissance de 16 |
|---|---|---|---|
| $2^0 = 1$ | $8^0 = 1$ | $10^0 = 1$ | $16^0 = 1$ |
| $2^1 = 2$ | $8^1 = 8$ | $10^1 = 10$ | $16^1 = 16$ |
| $2^2 = 4$ | $8^2 = 64$ | $10^2 = 100$ | $16^2 = 256$ |
| $2^3 = 8$ | $8^3 = 512$ | $10^3 = 1000$ | $16^3 = 4096$ |
| $2^4 = 16$ | $8^4 = 4096$ | $10^4 = 10000$ | $16^4 = 65536$ |
| $2^5 = 32$ | $8^5 = 32768$ | | |
| $2^6 = 64$ | | | |
| $2^7 = 128$ | | | |
| $2^8 = 256$ | | | |
| $2^9 = 512$ | | | |
| $2^{10} = 1024$ | | | |
| $2^{11} = 2048$ | | | |
| $2^{12} = 4096$ | | | |

**Exercice :** Quel est le code décimal correspondant à $(1\ 1001\ 1000)_2$ et $(1010\ 1010)_2$ ?

$(1\ 1001\ 1000)_2 = (\quad)_{10}$

$(1010\ 1010)_2 = (\quad)_{10}$

### 3.3/ Conversion d'un nombre en décimal vers son équivalent en octal ou hexadécimal [$(N)_{10}$ -> $(N)_8$ ou $(N)_{16}$]

Il s'agit ici d'appliquer la même méthode que pour le passage du décimal vers le binaire ( § 3.1) en divisant successivement le nombre décimal par 8 (conversion en octal) ou par 16 (conversion en hexadécimal).

**Exercice :** Convertir les nombres suivants :

$(1028)_{10} = (\quad)_8$

$(61)_{10} = (\quad)_{16}$

$(4095)_{10} = (\quad)_8$

$(2748)_{10} = (\quad)_{16}$

### 3.4/ Conversion d'un nombre en octal ou hexadécimal vers son équivalent en décimal [$(N)_8$ -> $(N)_{10}$ ou $(N)_{16}$ -> $(N)_{10}$]

Il s'agit ici d'appliquer la même méthode que celle pour le passage d'un nombre binaire en décimal ( § 3.2), avec dans la formule du § 2.2 respectivement B=8 ou B=16.

**Exercice :** Convertir les nombres suivants :

$(1027)_8$ = ( $)_{10}$

$(61F)_{16}$ = ( $)_{10}$

### 3.5/ Récapitulatif méthode à employer pour le transcodage

| Base de départ | Base d'arrivée | Méthode de transcodage |
|---|---|---|
| Décimal | Binaire | Méthode de la division par 2 du nombre |
| | Octal | Méthode de la division par 8 du nombre |
| | Hexadécimal | Méthode de la division par 16 du nombre |
| Binaire | Décimal | $(N)_{10}= a_n \times B^n + a_{n-1} \times B^{n-1} + \ldots\ldots + a_1 \times B^1 + a_o \times B^0$ avec B=2 |
| Octal | | $(N)_{10}= a_n \times B^n + a_{n-1} \times B^{n-1} + \ldots\ldots + a_1 \times B^1 + a_o \times B^0$ avec B=8 |
| Hexadécimal | | $(N)_{10}= a_n \times B^n + a_{n-1} \times B^{n-1} + \ldots\ldots + a_1 \times B^1 + a_o \times B^0$ avec B=16 |

### 3.6/ Conversions directes binaire <=> hexadécimal

La conversion de la base 2 à la base 16 (et inversement) se fait aisément, la base 16 étant un multiple entier de la base 2. Elle permet de représenter sous une forme réduite un nombre binaire.

$2^4$ = 16 => un groupe binaire de 4 bits est **transcodable** directement en un digit hexadécimal.

**Méthode :** On divise le nombre binaire en tranches de 4 bits (à partir du LSB). Chacun des quartets est ensuite converti en un digit hexadécimal par simple sommation pondérée.

Exemple 1 :

$2^3\ 2^2\ 2^1\ 2^0 \quad 2^3\ 2^2\ 2^1\ 2^0 \quad 2^3\ 2^2\ 2^1\ 2^0$

$(111000110101)_2$ = 1 1 1 0 | 0 0 1 1 | 0 1 0 1

$(1110\ 0011\ 0101)_2$ = **( E 3 5 )$_{16}$**

Exemple 2 :

$(D4C7)_{16}$ = D 4 C 7

1101 0100 1100 0111 = $(1101\ 0100\ 1100\ 0111)_2$

### 3.7 / Conversion Décimal <=> code BCD (Décimal Codé Binaire)

Le code BCD est utilisé pour les afficheurs lumineux, son principe repose sur le codage de chaque digit décimal (chiffre) en son équivalent en binaire sur 4 bits (et inversement).

Exemple :

$(1\ 2\ 7)_{10}$ = $(0001\ 0010\ 0111)_{BCD}$

Exercice : Effectuer les transcodages suivants :

$( 5\ 7\ 6)_{10} = (\qquad\qquad\qquad\qquad)_{BCD}$

$( 9\ 9\ )_{10} = (\qquad\qquad\qquad)_{BCD}$

$( 1000\ 0011\ 0110)_{BCD} = (\qquad\qquad)_{10}$

Exercice : Combien faut-il de bits pour représenter un nombre décimal de 5 chiffres dans le code BCD ?

## 4. / Le code de GRAY

Le « code à distance unité » ou **code de Gray**, également appelé code reflex ou code binaire réfléchi, est **un code non pondéré**, c'est à dire que les **positions binaires ne sont affectées d'aucun poids**.

Le code GRAY est aussi utilisé dans l'écriture des **tableaux de Karnaugh** (c'est pour plus tard…)

Principe d'obtention du code GRAY :
Si l'on dispose d'un ensemble de $2^N$ mots réalisant un code de Gray , il est facile d'obtenir un ensemble de $2^{(N+1)}$ mots respectant le même code.

Ce nouvel ensemble est constitué :
- d'une part des mots-code du code initial , précédés de « 0 »'
- d'autre part des mots-code du même code initial , présentés dans l'ordre inverse et précédés de « 1 »

| Décimal | Binaire | Code Gray |
|---|---|---|
| 0 | 0000 | 000**0** |
| 1 | 0001 | 000**1** |
| 2 | 0010 | 00**11** |
| 3 | 0011 | 00**10** |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |
| 16 | | |
| 17 | | |
| 18 | | |
| 19 | | |

## 5. / Les autres codes.

### 5.1/ Le code ASCII

Il tire son appellation de l'abréviation américaine : **American Standard Code Interchange Information.**

C'est un code qui permet la représentation des caractères alphanumériques d'un micro-ordinateur, chaque caractère étant codé par un mot de 8 bits appelé **octet** .

Ce code est très répandu dans le milieu de la micro-informatique.

**Table des caractères ASCII**

Table ASCII standard (codes de caractères de 0 à 127)

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | | (nul) | 016 | ► | (dle) | 032 | sp | 048 | 0 | 064 | @ | 080 | P | 096 | ` | 112 | p |
| 001 | ☺ | (soh) | 017 | ◄ | (dc1) | 033 | ! | 049 | 1 | 065 | A | 081 | Q | 097 | a | 113 | q |
| 002 | ☻ | (stx) | 018 | ↕ | (dc2) | 034 | " | 050 | 2 | 066 | B | 082 | R | 098 | b | 114 | r |
| 003 | ♥ | (etx) | 019 | ‼ | (dc3) | 035 | # | 051 | 3 | 067 | C | 083 | S | 099 | c | 115 | s |
| 004 | ♦ | (eot) | 020 | ¶ | (dc4) | 036 | $ | 052 | 4 | 068 | D | 084 | T | 100 | d | 116 | t |
| 005 | ♣ | (enq) | 021 | § | (nak) | 037 | % | 053 | 5 | 069 | E | 085 | U | 101 | e | 117 | u |
| 006 | ♠ | (ack) | 022 | ▬ | (syn) | 038 | & | 054 | 6 | 070 | F | 086 | V | 102 | f | 118 | v |
| 007 | • | (bel) | 023 | ↨ | (etb) | 039 | ' | 055 | 7 | 071 | G | 087 | W | 103 | g | 119 | w |
| 008 | ◘ | (bs) | 024 | ↑ | (can) | 040 | ( | 056 | 8 | 072 | H | 088 | X | 104 | h | 120 | x |
| 009 | | (tab) | 025 | ↓ | (em) | 041 | ) | 057 | 9 | 073 | I | 089 | Y | 105 | i | 121 | y |
| 010 | | (lf) | 026 | | (eof) | 042 | * | 058 | : | 074 | J | 090 | Z | 106 | j | 122 | z |
| 011 | ♂ | (vt) | 027 | ← | (esc) | 043 | + | 059 | ; | 075 | K | 091 | [ | 107 | k | 123 | { |
| 012 | ♀ | (np) | 028 | ∟ | (fs) | 044 | , | 060 | < | 076 | L | 092 | \ | 108 | l | 124 | \| |
| 013 | | (cr) | 029 | ↔ | (gs) | 045 | - | 061 | = | 077 | M | 093 | ] | 109 | m | 125 | } |
| 014 | ♫ | (so) | 030 | ▲ | (rs) | 046 | . | 062 | > | 078 | N | 094 | ^ | 110 | n | 126 | ~ |
| 015 | ☼ | (si) | 031 | ▼ | (us) | 047 | / | 063 | ? | 079 | O | 095 | _ | 111 | o | 127 | ⌂ |

### 5. 2/ Le Code Barre

Ce principe de codage, apparu dans les années 80, est largement utilisé sur les produits de grande consommation, car il facilite la gestion des produits.

Le marquage comporte un certain nombre de barres verticales ainsi que 13 chiffres :

- Le 1er chiffre désigne le pays d'origine : 3 = France, 4 = Allemagne, 0 = U.S.A, Canada etc …
- Les cinq suivants sont ceux du code « fabricant »,
- Les six autres sont ceux du code de l'article,
- Le dernier étant une clé de contrôle

Les barres représentent le codage de ces chiffres sur 7 bits , à chaque chiffre est attribué un ensemble de 7 espace blancs ou noirs.

*REMARQUE : D'autres codes moins utilisés existent comme le code Aiken, le code EXCESS 3 etc…*

## 6. / Représentation des nombres signés en binaire

Pour le moment, nous n'avons parlé que de nombres positifs. Il peut s'avérer indispensable de traiter également des nombres négatifs. **Le langage binaire ne connaît pas le signe -** (!)

Il existe 3 conventions pour exprimer les nombres signés dans le système binaire :
- Représentation de la valeur et du signe indépendamment (binaire signé),
- Représentation en complément à 1,
- Représentation en complément à 2.

### 6.1/ Le binaire signé

L'une des méthodes est de réserver **un bit pour indiquer le signe** du nombre, d'où l'appellation de binaire signé.
Le bit réservé au signe est toujours le bit le plus à gauche. (attention : Il ne correspond plus au MSB !)
Pour le bit de signe et par convention, le **0 représente le +** et le **1 le -**.

**Exemple :** $(-23)_{10}$ = (**1** 0010111) ; $(+23)_{10}$ = (**0** 0010111)

### 6.2/ Complément à 1 d'un nombre binaire

Pour calculer le complément à 1(CA1) d'un nombre binaire, il suffit de complémenter chaque bit de ce nombre c'est-à-dire remplacer les 1 par des 0 et les 0 par des 1.
**Attention : En notation signé, le bit de signe reste inchangé (ne pas le complémenter)**

**Exemple :**

$(-23)_{10}$ = s'écrit (**1** 0010111) en binaire signé (ci-dessus) et son complément à 1 est (**1** 1101000**), le bit de signe étant inchangé.**

### 6.2/ Complément à 2 d'un nombre binaire

Pour calculer le CA2 d'un nombre binaire N, on ajoute la valeur 1 au CA1 de N. (CA2 = CA1 + 1)

**Exemple 1 :**

Nombre binaire N (non signé) : N = 0101
Complément à 1 de N : $(N)_{CA1}$= 1010
Complément à 2 de N : $(N)_{CA2}$ = 1011

**Exemple 2 :**

$(-23)_{10}$ = s'écrit (**1** 0010111) en binaire signé,
son complément à 1 est (**1** 1101000),
son complément à 2 est (**1** 1101001).

## 7. / Calcul arithmétiques

### 7.1 Addition binaire

L'addition de 2 nombres binaires est parfaitement analogue à l'addition de 2 nombres décimaux. Il faut commencer par le bit de poids le plus faible en utilisant l'algorithme suivant :

| | |
|---|---|
| 0 + 0 = 0 | 1 + 0 = 1 |
| 1 + 1 = 10 (= 0 + report de 1 sur la gauche) | 1 + 1 + 1 = 11 (= 1 + report de 1 sur la gauche) |

**Exercice :**

Faire les additions suivantes en binaire

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 5 | | | | 15 | | |
| + | 3 | + | _____ | + | 11 | + | _____ |
| = | 8 | = | | = | 26 | = | |

### 7.2 Soustraction binaire

Dans le cas de la soustraction de deux nombres binaires **non signés** on peut utiliser l'algorithme suivant :

| | |
|---|---|
| 0 - 0 = 0 | 0 - 1 = 1 (avec un report de 1 à retrancher au chiffre supérieur) |
| 1 - 0 = 1 | 1- 1 = 0 |

**Exemple :** opération 7 - 2

```
   0111
-  0010
=
```

**Une autre méthode** consiste faire une **addition de deux nombres de signes contraires**

DEUX CAS PEUVENT SE PRESENTER :

**1er cas** : La grandeur du nombre positif est supérieure à celle du nombre négatif

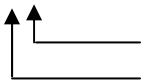
Par exemple $(+17)_{10}$ = (**0** 10001)$_{2\ signé}$ et $(-12)_{10}$ = (**1** 01100)$_{2\ signé}$
$(-12)_{10}$ étant négatif il faut le remplacer par son complément à 2

*Calcul du complément à 2 de $(-12)_{10}$ :*

Complément à 1 de (-12) = ( )$_{CA1}$
Complément à 2 de (-12) = ( )$_{CA2}$

Il reste à faire l'addition avec le complémentant à 2 du nombre à soustraire, **sans oublier son bit de signe**.

1er nombre : **0** 1 0 0 0 1 $(+17)_{2\ signé}$
2eme nombre : + $(-12)_{CA2}$



**Remarque :** nous avons additionné les bits de signe et la retenue ; cela peut entraîner un débordement comme dans le cas ci-dessus, un débordement qui est toujours rejeté.

**2ème cas** : La grandeur du nombre positif est inférieure à celle du nombre négatif

Par exemple $(-17)_{10}$ = (**1** 10001)$_{2\ signé}$ et $(+12)_{10}$ = (**0** 01100)$_{2\ signé}$
comme dans le cas précédent calculons le complément à 2 de $(-17)_{10}$

Complément à 1 de (-17) = ( )$_{CA1}$
Complément à 2 de (-17) = ( )$_{CA2}$

Il reste à faire l'addition avec le complémentant à 2 du nombre à soustraire, **sans oublier son bit de signe**.

1er nombre : $(-17)_{CA2}$
2eme nombre : + **0** 0 1 1 0 0 $(+12)_{2\ signé}$

bit de signe

Le résultat de l'addition est négatif (bit de signe = 1) [c'est d'ailleurs ce qui le différencie du 1er cas]. **Ce résultat est écrit sous la forme du complément à 2**.

Pour passer du complément à 2 au binaire signé il faut utiliser la méthode suivante :

- Soustraire 1 au nombre binaire (passage du complément à 2 au complément à 1),
- Complémenter chaque bit un à un sauf le bit de signe (passage au binaire signé).

Dans l'exemple précédent le résultat donne :

Résultat : ( $)_{CA2}$ -> ( $)_{CA1}$ -> ( $)_{2\ signé}$ = ( $)_{10}$ (CQFD !)