# UFR SET de Thies Electronique numérique

# Chapitre4 : REPRESENTATION ET ARITHMETIQUE BINAIRE DANS LES ORDINATEURS

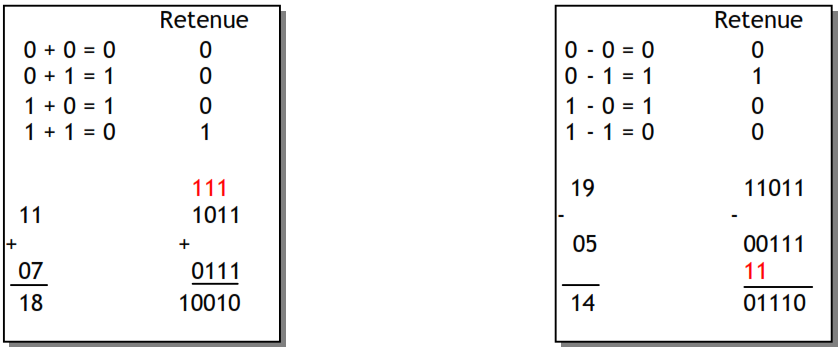
1. **Objectifs**

* **Etre capable de donner les différentes représentations des nombres**
* **Etre capable d’effectuer des calculs arithmétiques en binaire**

1. **Représentation des nombres dans les ordinateurs**

Il y a trois représentations possibles pour réaliser les opérations de base (ADD, SOUST, MULT ET DIV) dans les ordinateurs.

* Représentation en module et signe (MS) ;
* Représentation en complément à 1 (C1) ;
* Représentation en complément en 2 (C2).

Ces nombres peuvent faire l’objet de plusieurs types de calculs arithmétiques se basant principalement sur les règles appliquées en base dix. Nous appliquerons en priorité ces règles de calcul à l’addition et à la soustraction dont le tableau ci-dessous résume les opérations stratégiques.

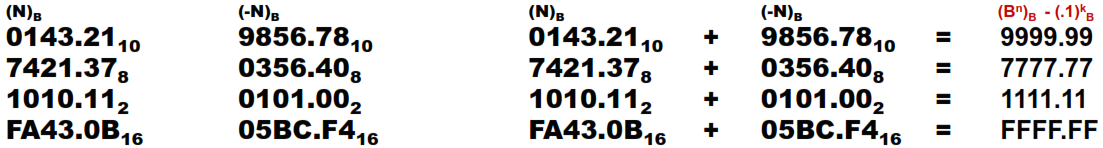
* 1. **Représentation en module et signe (sign and magnitude)**

Pour la représentation en module et signe, les nombres de n bits qu’on peut représenter vont de –(Bn-1 -1) à +(Bn-1 -1). Les nombres décimaux de 3 bits vont de -99 à +99, les nombres octaux de 3 bits vont de -77 +77 et les nombres hexadécimaux de 3 bits vont de –FF à +FF. L’extension d’un nombre codé sur n bits à un format n+k bits consiste à décaler le bit de signe à la position du MSB et à compléter les autres par 0. Cela peut s’appliquer sur les nombres positifs comme sur les nombres négatifs. Exemples : +98 sur 3 bits s’écrit **0**98 et sur 6 bits ou digits **0**00098. -98 sur 3 bits s’écrit **9**98 et sur 6 bits ou digits **9**00098.

***Inconvénients :*** cette méthode impose que le signe soit traité indépendamment de la valeur. Il faut donc des circuits différents pour l’addition et la soustraction. De plus on obtient deux représentations différentes pour 0, soit +0 et -0. Pour pallier à ces inconvénients, il faut un mode de représentation qui permet d’utiliser le même circuit pour effectuer l’addition et la soustraction. Comme x – y = x + (-y), il faut que le signe – soit traité comme partie intégrante de la valeur négative. La forme complémentée utilisée pour représenter les nombres négatifs permet de réaliser cet objectif.

* 1. **Conversion binaire signé vers décimal**
* Si le bit de signe de x2 = 0, on le converti directement en décimal ;
* Si le bit de signe de x2 = 1, cela veut dire que x10 est négatif : on cherche le complément à 2 de x2, on le convertit en décimal et un rajoute un moins (-) devant. ***Exemple*** : 01110100 = 116 ; 10100010 : le bit de signe = 1, le nombre est négatif. On prend le complément à 2 soit 01011110. On le convertit en décimal soit 94. Donc 10100010 = -94
  1. **Représentation en complément B-1**

Si un nombre N de base B est composé de m chiffres dont la partie entière fait n chiffres, le signe inclus et une partie fractionnaire de k chiffres, nous pouvons représenter ce nombre en notation complément de base moins un (B-1) comme suit : ***Exemple***: N = 0123.45 avec m = 6, n = 4 et k = 2

Les nombres qu’on peut représenter vont de –(2n-1 – 1) à +(2n-1 – 1)

* 1. **Représentation en complément à 1 (ou complément restreint : C1)**

La représentation d’un nombre en complément en 1 pose le problème du zéro qui a une double représentation. +0 donne 0000. Le complément de 0 doit rester 0 s’écrit 1111. Nous nous trouvons avec 0 qui s’écrit 0000 et 0 qui s’écrit 1111. Nous appelons le premier zéro +0 et le second zéro -0.

S’il s’agit de la base 8, 10 et 16, nous rencontrons les mêmes problèmes. En base 8, +0 s’écrit 0000 et son complément de 7 est 7777 qui est -0. En base 10, +0 s’écrit 0000 et son complément de 9 est 9999 qui est -0. En base 16, +0 s’écrit 0000 et son complément de 15 est FFFF qui est -0.

Pour éviter cette situation, nous allons représenter un nombre négatif par le complément de B qui est le complément de (B-1) plus 1.

En décimal, on forme le complément à 9 d’un nombre en remplaçant à chaque chiffre de ce nombre par sa différence avec 9. ***Exemple*** : le complément à 9 de 6473, noté C9(6473) est 3526. En binaire, on forme le complément à 1 d’un nombre en remplaçant chaque chiffre de ce nombre par sa différence avec 1 (on remplace les 1 par des 0 et réciproquement). ***Exemple*** : le complément à 1 de 11010, noté C1(11010) est 00101.

* 1. **Représentation en complément à 2 (ou complément vrai : C2)**

En décimal, on forme le complément à 10 d’un nombre en remplaçant le chiffre des unités par sa différence avec 10 et les autres chiffres de ce nombre par leur différence avec 9. ***Exemple*** : le complément à 10 de 6473, noté C2(6473) est 3527. On peut aussi dire que le complément à 10 d’un nombre s’obtient en le soustrayant de la puissance de 10 immédiatement supérieure à ce nombre : en effet 3527 = 10000 – 6473. On voit que à 10 s’obtient en ajoutant 1 au complément à 9. En binaire, on va utiliser la règle ci-dessus et former le complément à 2 d’un nombre en ajoutant 1 au complément à 1. ***Exemple*** : le complément à 2 de 11010, noté C2(11010) est 00101 + 1 = 00110.

* + 1. **Conversion décimal vers binaire signé**
* Si x10 est positif ou nul, on le convertit directement en binaire ;
* Si x10 est négatif, on convertit sa valeur absolue en binaire et on prend le complément à 2.
  + 1. **Conversion binaire signé vers décimal**
* Si le bit de signe de x2 = 0, on le convertit directement en décimal ;
* Si le bit de signe de x2 = 1, cela veut dire que x10 est négatif : on cherche le complément à 2 de x2, on le convertit en décimal et on rajoute un moins (-) devant.

***Exemples :*** 0111 0100 = 116 ; 1010 0010 : le bit de signe est 1, le nombre est donc négatif on en prend le complément à 2 soit 0101 1110. On le convertit en décimal soit 94. Donc 1010 0010 = -94

***Exemples :*** 68 = 0100 0100 ; -67 : /-67/ = 0100 0011, donc -67 = C2(0100 0011) = C1(0100 0011)+1=1011 1100+1=1011 1101

* + 1. **Limites**

Comme on s’est limité à 8 bits, on ne pourra coder en binaire que 256 valeurs différentes. En nombres non signés (donc exclusivement positifs) on pouvait aller de 010 = (0000 00002) à 25510 = (1111 11112) soit 256 valeurs en tout car 256 = 28. En nombres signés, le plus grand nombre commençant par un 0 (donc positif) est 0111 1111 = 127. Le nombre binaire suivant est 1000 0000 mais, il commence par un 1, il doit être considéré comme négatif : pour trouver sa valeur décimale, on en prend le complément à 2 soit 1000 0000 (c’est le même) qui, convertit en décimal, donne 128. 128 est donc la valeur absolue de 1000 0000 et on obtient en définitive 1000 0000 = -128. Sur un octet les 256 valeurs codables en nombres signés vont donc de -128 à +127. Le nombre 128 n’est pas codable en nombres signés sur 8 bits. En nombres signés sur n bits, on peut coder les nombres de -2n-1 à 2n-1-1

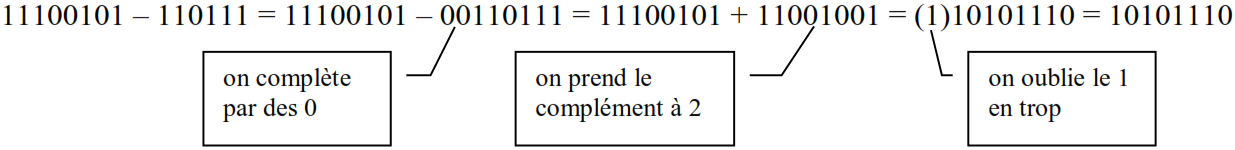
* 1. **Soustraction par complémentation à 2**

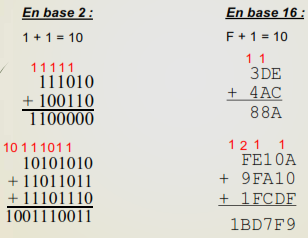
Le but est de remplacer la soustraction par une addition.

* + 1. **En décimal**

Soit à faire la soustraction suivante 742 – 568 = 174. Comme 568 = 1000 – 432, on peut écrire 742 – 568 = 742 – (1000 – 432) = 742 + 432 – 1000 = 174. Comme 432 est le complément à 10 de 568, on a en fait effectué l’opération suivante : x – y = x + C10(y) – (puissance 10 immédiatement supérieure à x). Au lieu de soustraire 1000, il aurait suffi de négliger la dernière retenue dans le résultat. En effet 742 + 432 = 1174 = 174 si on néglige le 1 de gauche. *En résumé, au lieu de soustraire un nombre y d’un nombre x, on ajoute à x le complément à 10 de y et on oublie le (n+1)ième chiffre si on travaille sur n chiffres.* ***Remarque*** : si les deux nombres x et y ne comportent pas le même nombre de chiffres, il faut rajouter des 0 devant y avant de chercher son complément à 10. ***Exemple*** : 742 – 76 = 742 – 076 = 742 + 933 = (1)675 = 675.

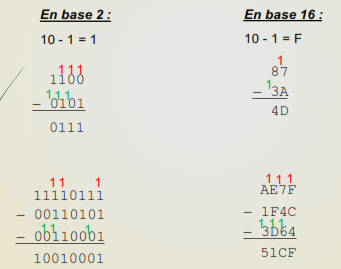
* + 1. **Binaire**

C’est pareil (avec la même remarque que ci-dessus). x - y = x + C2(y) – (puissance de 2 immédiatement supérieure à x).

1. **Addition binaire**

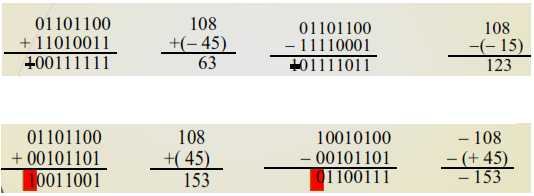
Elle s’effectue comme en décimal. Quand on additionne deux nombres, on obtient une retenue quand la somme d’une colonne est supérieure ou égale à B. ***Exemple*** :

1. **Soustraction binaire**

Elle s’effectue comme en décimal. Quand on soustrait deux nombres, on ajoute une retenue en haut quand la soustraction sur une colonne est impossible. Et on soustrait 1 en bas sur la colonne suivante.

1. **Opération avec le complément à 2**

On peut donc effectuer les différentes opérations avec les nombres signés en complément à 2. ***Exemple*** : sur 8 bits :

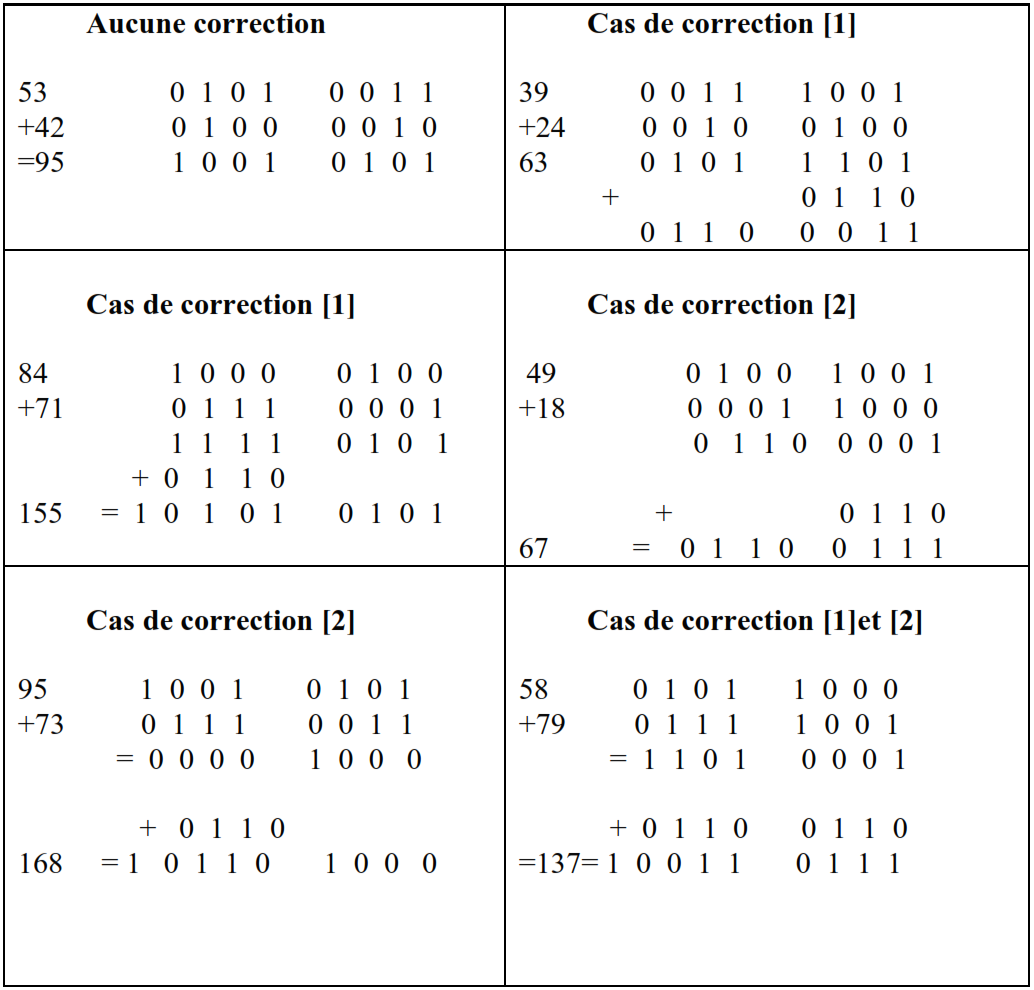


1. **Les opérations arithmétiques en codification BCD**
   1. **Addition en BCD**

Lors d’une addition de deux nombres binaires codés en BCD, il faut tenir compte :

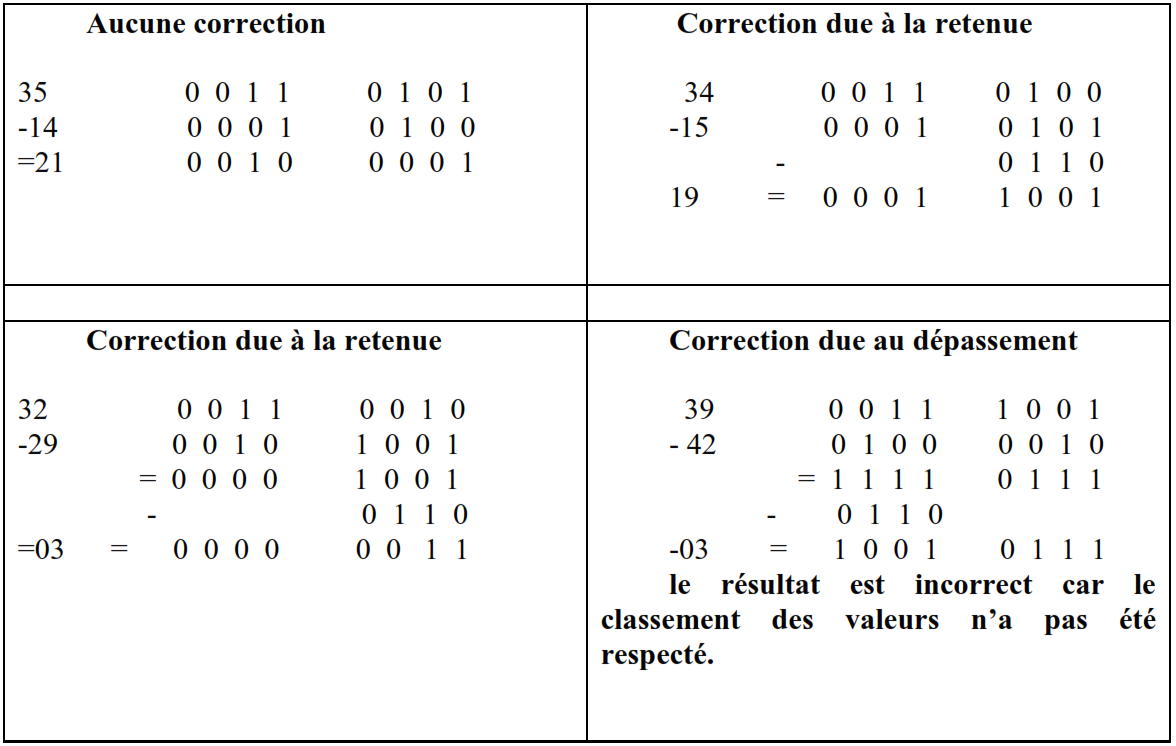
* Du dépassement de capacité de chaque quartet ;
* De la retenue terminale de chaque quartet.

La correction des erreurs d’addition en BCD est réalisée en ajoutant le nombre 6 (0110 en binaire) au quartet pour lequel l’un de ces deux cas est détecté [1] et [2]

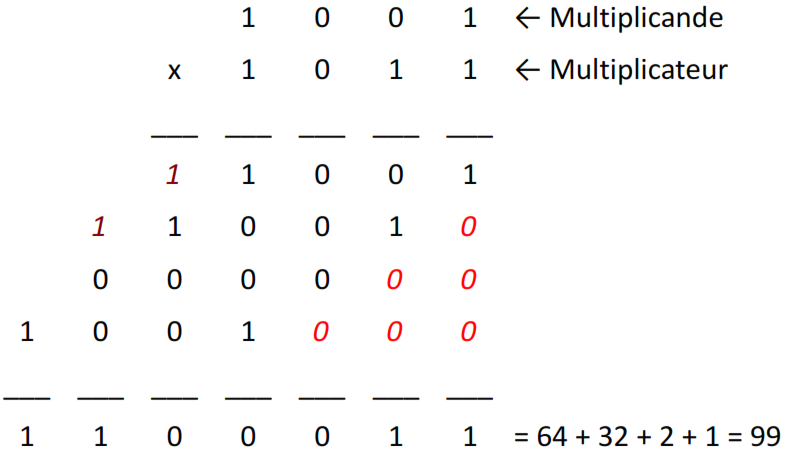


* 1. **Soustraction en BCD**

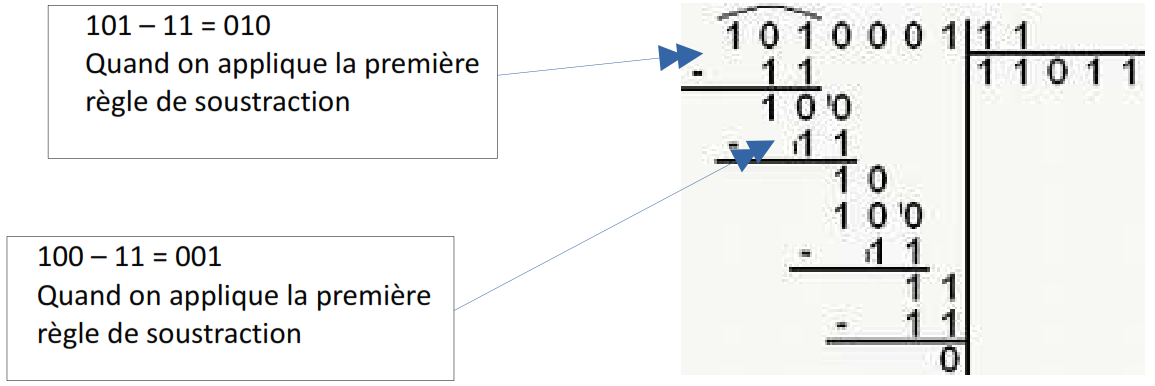
Lors d’une soustraction en BCD, il faut tenir compte du classement relatif de leurs valeurs absolues. Le signal final est le signe du plus grand nombre. Une correction est nécessaire lorsqu’on détecte une retenue terminale. La correction consiste à soustraire par le nombre 6 (0110 en binaire). Si une retenue terminale du dernier quartet apparait, le résultat est faux parce que le classement des valeurs absolues n’a pas été respecté.



1. **Multiplication en binaires**

On multiplie des nombres en binaire de la même manière qu’on multiplie des nombres décimaux. ***Exemple*** : multiplions 9 = 10012 par 11 = 10112

1. **Division en binaires**

Le principe est le même que pour une division en décimal. ***Exemple*** : divisons 81 par 3