# Cours de structure machine

A l'issue du cours de structure machine, l'étudiant sera capable de concevoir des circuits électroniques numériques élémentaires pouvant constituer les ingrédients de base de divers dispositifs électroniques comme les ordinateurs. Le cours est proposé sous forme d'une séance de cours et d'une séance de travaux dirigés par semaine durant un semestre. Voici son programme détaillé :

Chapitre I – Systèmes de numération

1. Définition (système de numération, base)
2. Conversions (10↔B, B1↔B2)
3. Arithmétique binaire
4. Représentation des nombres numériques ou entiers
5. Soustraction par la méthode du complément
6. Problèmes liés à la longueur des nombres

Chapitre II – Algèbre de Boole

1. Définition
2. Théorèmes fondamentaux
3. Principe de dualité
4. Fonctions booléennes
5. Manipulations algébriques
6. Simplification des fonctions logiques (méthode de Karnaugh)

Chapitre III – Circuits logiques combinatoires

1. Définition
2. Additionneur (1/2 additionneur, additionneur complet)
3. Décodeur :
    a. Circuit interne du décodeur 2x4,
    b. Applications : représentation d'une fonction à l'aide de décodeur $Nx2^N$, sélection d'un circuit dans une UAL composé de 4 circuits, transfert d'information d'un registre vers un R0, R1, R2 ou R3.
4. Autres circuits combinatoires
    a. Afficheur à segments
    b. transcodeur

Chapitre IV – Circuits logiques séquentiels

1. Définition
2. Bascules : D, T, SR et JK (table caractéristique, table d'excitation, chronogramme)
3. Applications : registres à décalage, compteurs synchrones et asynchrones

# Chapitre I - Systèmes de numération

# Chapitre I - Systèmes de numération

## I.1 – Définition

Un système de numération est un ensemble de règles et de symboles permettant de représenter des informations quantitatives. Il est caractérisé par trois entités mathématiques importantes :

- Une base
- Un ensemble de chiffres
- Des règles de représentation des nombres

La base est une valeur permettant de déterminer les poids des chiffres dans la représentation des nombres. L'ensemble des chiffres est composé de symboles représentant chacun une quantité. Le nombre de chiffres doit être impérativement égal à la base et ils sont reliés par une relation d'ordre. Le plus grand chiffre doit être égal à la base – 1. Les règles de représentation des nombres permettent d'écrire les nombres et de les interpréter. En général, les nombres sont représentés sous forme d'une suite ordonnée de chiffres.

Exemples :

| Base | 10 | 2 | 3 | 4 | 8 | 16 |
|---|---|---|---|---|---|---|
| Ensemble des chiffres | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 | 0,1 | 0, 1, 2 | 0, 1, 2, 3 | 0, 1, 2, 3, 4, 5, 6, 7 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F |
| Exemple de nombres | 129 | 10000001 | 11210 | 2001 | 201 | 81 |

☞ Remarque :

- La base 10 est la plus utilisée par nous les êtres humains, ceci est dû au fait qu'à l'origine, nous utilisions nos doigts pour compter (nous avons 10 doigts !)
- La base 2 est très utilisée depuis l'invention de l'ordinateur. Ceci est dû au fait que ce dernier se sert du système binaire pour représenter les informations.
- Les bases 8 (dite aussi octale) et 16 (dite aussi hexadécimale) sont largement utilisées. Ceci est dû au fait que, nous les humains, ne sommes pas à l'aise avec le système binaire et comme c'est très facile de passer du système binaire à la base 8 ou 16, nous nous servons de ces deux bases aussi.
- Théoriquement, il existe une infinité de systèmes de numération

☞ Notation :

- Afin d'identifier dans quelle base est représenté un nombre, nous écrirons les nombres entre parenthèse en indiquant, en indice, la base. Exemple :
  - Pour représenter 129,5 en décimal, il faut écrire $(129,5)_{10}$.
  - Pour représenter 126,5 en octale (rappelez-vous octale veut dire base 8), il faut écrire $(126,5)_8$.
  - Attention $(129,5)_{10}$ est différent de $(129,5)_8$.

☞ Comment interpréter un nombre ?

Réponse : d'abord, il faut savoir qu'il représente une valeur. Cette valeur est calculée selon la formule suivante :

$$(N)_B = (a_{n-1}\ B^{n-1} + a_{n-2}\ B^{n-2} + \dots + a_1\ B^1 + a_0\ B^0 + a_{-1}\ B^{-1} + a_{-2}\ B^{-2} + \dots\ a_{-p+1}\ B^{-p+1} + a_{-p}\ B^{-p})$$

Chiffre le plus significatif

Chiffre le moins significatif

$$(N)_B = \sum_{i=-p}^{n-1} a_i B^i$$

Légende :

- N : notre nombre
- B : notre base
- $a_i$ : chiffres (attention $a_i$<B)
- $B^i$ : poids des chiffres
- i : rang

Exemples :

| $N = (129,5)_{10}$ (Base 10) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rangs | … | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | … |
| Poids selon la base 10 | … | $10^9$ | $10^8$ | $10^7$ | $10^6$ | $10^5$ | $10^4$ | $10^3$ | $10^2$ | $10^1$ | $10^0$ | $10^{-1}$ | … |
| Position des chiffres | … | | | | | | | | **1** | **2** | **9** | 5 | … |
| Ce qui donne | $1x10^2 + 2x10^1 + 9x10^0 + 5x10^{-1}$ | | | | | | | | | | | | |

| $N = (10000001,1)_2$ (Base 2) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rangs | … | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | … |
| Poids selon la base 2 | … | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ | … |
| Position des chiffres | … | | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | … |
| Ce qui donne | $1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}$ | | | | | | | | | | | | |

| $N = (11210,1)_3$ (Base 3) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rangs | … | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | … |
| Poids selon la base 3 | … | $3^9$ | $3^8$ | $3^7$ | $3^6$ | $3^5$ | $3^4$ | $3^3$ | $3^2$ | $3^1$ | $3^0$ | $3^{-1}$ | … |
| Position des chiffres | … | | | | | | **1** | **1** | **2** | **1** | **0** | 1 | … |
| Ce qui donne | $1 \times 3^4 + 1 \times 3^3 + 2 \times 3^2 + 1 \times 3^1 + 0 \times 3^0 + 1 \times 3^{-1}$ | | | | | | | | | | | | |

| $N = (2001,2)_4$ (Base 4) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rangs | … | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | … |
| Poids selon la base 4 | … | $4^9$ | $4^8$ | $4^7$ | $4^6$ | $4^5$ | $4^4$ | $4^3$ | $4^2$ | $4^1$ | $4^0$ | $4^{-1}$ | … |
| Position des chiffres | … | | | | | | | **2** | **0** | **0** | **1** | **2** | … |
| Ce qui donne | $2 \times 4^3 + 0 \times 4^2 + 0 \times 4^1 + 1 \times 4^0 + 2 \times 4^{-1}$ | | | | | | | | | | | | |

| $N = (201,4)_8$ (Base 8) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rangs | … | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | … |
| Poids selon la base 8 | … | $8^9$ | $8^8$ | $8^7$ | $8^6$ | $8^5$ | $8^4$ | $8^3$ | $8^2$ | $8^1$ | $8^0$ | $8^{-1}$ | … |
| Position des chiffres | … | | | | | | | | **2** | **0** | **1** | **4** | … |
| Ce qui donne | $2 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 + 4 \times 8^{-1}$ | | | | | | | | | | | | |

| $N = (81,8)_{16}$ (Base 16) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rangs | … | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | … |
| Poids selon la base 16 | … | $16^9$ | $16^8$ | $16^7$ | $16^6$ | $16^5$ | $16^4$ | $16^3$ | $16^2$ | $16^1$ | $16^0$ | $16^{-1}$ | … |
| Position des chiffres | … | | | | | | | | | **8** | **1** | **8** | … |
| Ce qui donne | $8 \times 16^1 + 1 \times 16^0 + 8 \times 16^{-1}$ | | | | | | | | | | | | |

☞ Remarque :

- Dans la représentation des nombres, on n'écrit pas les chiffres non significatifs.
- Les chiffres non significatifs correspondent à tous les zéro situés à gauche de la partie entière et à droite de la partie décimale
- Dans les exemples cités ci-dessus, on voit qu'il est possible de trouver (en décimal) la valeur d'un nombre écrit dans une base quelconque : il suffit de développer le calcul selon la formule suivante : $(N)_B = (\sum_{i=-p}^{n-1} C_i B^i)_{10}$.

## I.2 - Conversions (10↔B, B1↔B2)

### I.2.1 – Conversions 10↔B

Soit N un nombre quelconque exprimé en base B. Nous rappelons que l'on peut l'écrire selon la formule suivante :

$(N)_B = (a_{n-1} B^{n-1} + a_{n-2} B^{n-2} + \dots + a_1 B^1 + a_0 B^0 + a_{-1} B^{-1} + a_{-2} B^{-2} + \dots \; a_{-p+1} B^{-p+1} + a_{-p} B^{-p})$

$(N)_B = \sum_{i=-p}^{n-1} a_i B^i$

Légende :

- N : notre nombre
- B : notre base
- $a_i$ : chiffres (attention $a_i$<B)
- $B^i$ : poids des chiffres
- i : rang

Exemples :

- $(129,5)_{10}$ $= 1 \times 10^2 + 2 \times 10^1 + 9 \times 10^0 + 5 \times 10^{-1}$ $= (129,5)_{10}$
- $(10000001,1)_2$ $= 1x2^7 + 0x2^6 + 0x2^5 + 0x2^4 + 0x2^3 + 0x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1}$ $= (129,5)_{10}$
- $(11210,1\dots.1)_3$ $= 1x3^4 + 1x3^3 + 2x3^2 + 1x3^1 + 0x3^0 + 1x3^{-1} + \dots + 1x3^{-infini}$ $= (129,5)_{10}$
- $(2001,2)_4$ $= 2 \times 4^3 + 0 \times 4^2 + 0 \times 4^1 + 1 \times 4^0 \times 2 \times 4^{-1}$ $= (129,5)_{10}$
- $(201,4)_8$ $= 2 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 + 4 \times 8^{-1}$ $= (129,5)_{10}$
- $(81,8)_{16}$ $= 8 \times 16^1 + 1 \times 16^0 + 8 \times 16^{-1}$ $= (129,5)_{10}$
- $(129,8)_{16}$ $= 1 \times 16^2 + 2 \times 16^1 + 9 \times 16^0 + 8 \times 16^{-1}$ $= (297,5)_{10}$

☞ Remarque :

- Une même suite de chiffres écrite dans deux bases différentes ne correspond pas à la même valeur. En effet : $(129)_{10}$ est différents de $(129)_{16}$.
- Deux suites de chiffres différentes écrites dans deux bases différentes peuvent correspondre à la même valeur. Exemple : $(129)_{10} = (10000001)_2 = (11210)_3 = (2001)_4 = (201)_8 = (81)_{16}$.
- Il est très facile de calculer la valeur d'un nombre représenté dans une base quelconque : il suffit de le développer selon la formule donnée précédemment.
- Attention, pour la base 16, on emprunte 6 lettres de l'alphabet pour représenter les chiffres dépassant 9. En effet nous avons les chiffres A, B, C, D, E et F qui correspondent respectivement aux valeurs décimales 10, 11, 12, 13, 14 et 15. (*Rappelez-vous le plus grand chiffre d'une bases est égale à la base – 1 !*)

Nous venons de voir comment convertir un nombre d'une base B vers la base 10. Maintenant qu'en est-il de l'opération inverse (Base B vers base 10) ?

*Réponse* : On utilise en général la méthode des divisions et multiplications successives pour convertir un nombre écrit en base 10 vers une base B.

Cas de la conversion des entiers : Pour convertir un nombre entier $(N)_{10}$ vers une base B on procède comme suit :

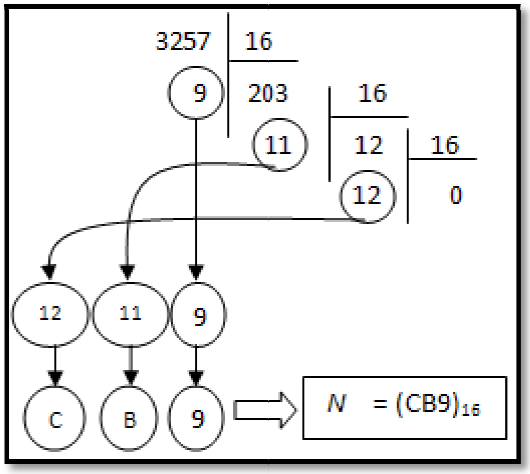
1. Effectuer la division entière de *N* par *B*. Soit *D* le résultat de cette division et *R* le reste
2. Si *D* >= *B*, recommencer en 1
3. Sinon, l'écriture en base *B* de *N* est égal à la concaténation du dernier résultat et de tous les restes en commençant par le dernier.

Exemple : conversion du nombre $(3257)_{10}$ en base 16

D'abord considérons la partie entière N = $(3257)_{10}$

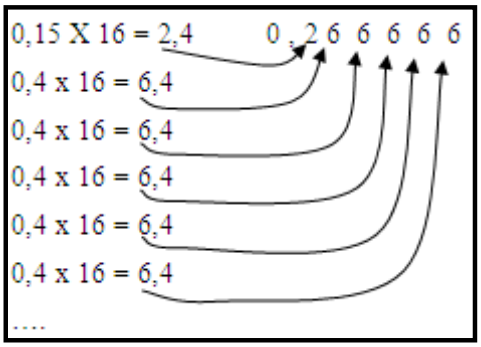
Il faut maintenant procéder avec des divisions successives selon le schéma suivant :

☞ *Remarque* : 11 se note B, 12 se note C en base 16.




Cas des nombres fractionnaires : Avant toute autre chose, il faut savoir qu'un nombre fractionnaire est composé d'une partie entière et d'une partie décimale. Pour convertir la partie entière de la base 10 vers une base B on procède avec des divisions successives. Pour la partie décimale, on procède avec des multiplications successives. On multiplie la partie fractionnaire par la base B et on répète cette opération avec la partie fractionnaire du produit jusqu'à ce qu'elle devient égale à 0. La valeur recherchée est en fait composée de la concaténation des parties entières des différents produits obtenus. Dans le cas où on n'arrive pas à obtenir une partie fractionnaire nulle, on doit s'arrêter au bout d'un certain nombre d'opérations si on juge que la précision est suffisante.

Par exemple pour N = $(3257,15)_{10}$ nous avions déjà trouvé, pour la partie entière $(3257)_{10}$ la valeur $(CB9)_{16}$. Maintenant on doit calculer $(0,15)_{10}=(\ ?)_{16}$.




Ce qui nous donne $(0,15)_{10}= (0,26666\ \ldots.)_{16}$ et si on se contente de 4 chiffres après la virgule nous aurons : $(3257,15)_{10} = (CB9,2666)_{16}$

☞ *Remarque* : Dans l'exemple précédent, nous avons un cas de figure où on ne pourra jamais trouver une partie fractionnaire égale à 0. Dans ce cas, on doit s'arrêter en fixant une précision définie par un nombre de chiffres après la virgule.

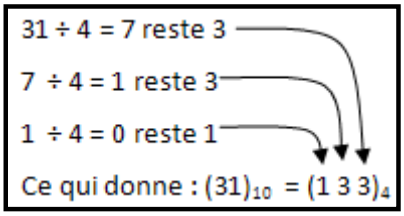
### I.2.2 – Conversions B1↔B2

En général, pour convertir un nombre d'une base B1 vers une base B2, il faut passer par une base intermédiaire qui est la base 10 : Convertir de la base B1 vers la base 10 puis convertir de la base 10 vers B2.

Exemple : $(47)_6 = (?)_4$

D'abord convertir $(47)_6$ en base 10 : $(47)_6 = 4x6^1 + 7x6^0 = (31)_{10}$.

Ensuite convertir $(31)_{10}$ en base 4 :




☞ Remarque :

- Bien que l'ordinateur utilise le système binaire pour traiter les nombres, nous les humains, préférons le système décimal, octal (base 8) ou hexadécimal (base 16).
- Comme il nous est difficile de lire les nombres binaires, on les converti souvent en base 8 ou 16. Heureusement qu'il est très facile de passer d'une base à une autre dans ces cas de figure (binaire↔octale et binaire↔hexadécimale).

Voyons comment passer de la base 2 vers la base 8 et inversement.

### I.2.3 – Conversions base 2 ↔ base 8 et base 2 ↔ base 16

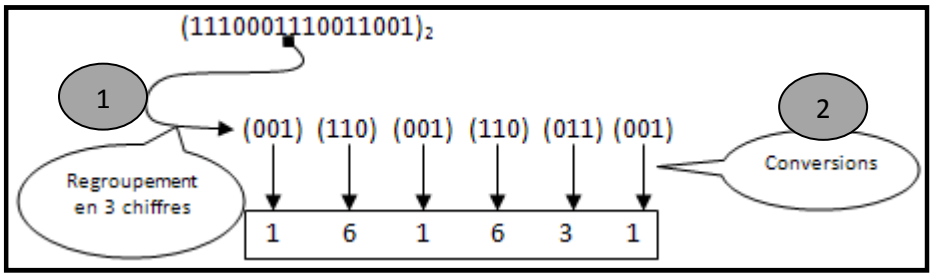
Avant d'aborder les conversions de la base 2 vers les bases 8 et 16, nous vous présentons la table de conversion des chiffres de la base 8 et 16 vers la base 2 :

| Base 10 | Base 8 | Base 16 | Base 2 |
|---|---|---|---|
| 0 | 0 | 0 | 0000 |
| 1 | 1 | 1 | 0001 |
| 2 | 2 | 2 | 0010 |
| 3 | 3 | 3 | 0011 |
| 4 | 4 | 4 | 0100 |
| 5 | 5 | 5 | 0101 |
| 6 | 6 | 6 | 0110 |
| 7 | 7 | 7 | 0111 |
| 8 | | 8 | 1000 |
| 9 | | 9 | 1001 |
| 10 | | A | 1010 |
| 11 | | B | 1011 |
| 12 | | C | 1100 |
| 13 | | D | 1101 |
| 14 | | E | 1110 |
| 15 | | F | 1111 |

### *I.2.3.1 – Conversion : base 2 ↔ base 8*

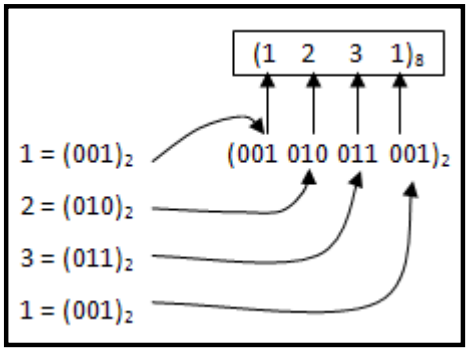
Pour convertir un nombre entier $(N1)_2 \rightarrow (N2)_8$, on regroupe le nombre N1 en sous ensembles de 3 chiffres binaires à partir de la virgule vers la droite et vers la gauche. Si on n'obtient pas, aux extrémités du nombre, des groupes de 3 bits, alors on doit les compléter par des zéros non significatifs. On convertie, ensuite, chaque sous ensemble en base 8.

Exemple 1 : $(1110001110011001)_2 = (\ ?)_8$.




Ce qui nous donne $(1110001110011001)_2 = (161631)_8$.

Exemple 2 : $(1231)_8 = (\ ?)_2$




Ce qui nous donne $(1231)_8 = (001\ 011\ 011\ 001)_2$

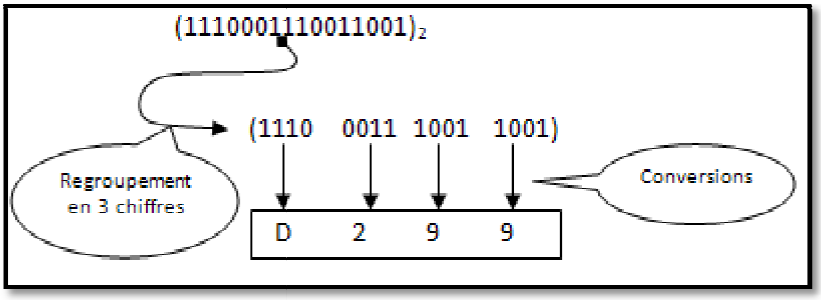
Exemple 2 : $(1231,5)_8 = (\ ?)_2$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ( | 1 | 2 | 3 | 1 | , | 5 | $)_8$ |
| | ↑↓ | ↑↓ | ↑↓ | ↑↓ | | ↑↓ | |
| ( | 001 | 010 | 011 | 001 | , | 101 | $)_2$ |

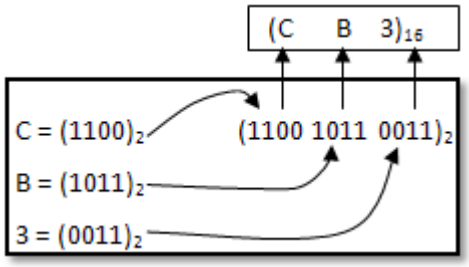
### *I.2.3.1 – Conversion : base 2 ↔ base 16*

Pour convertir un nombre $(N1)_2$ vers un nombre $(N2)_{16}$, on regroupe le nombre N1 en sous ensembles de 4 chiffres binaires à partir de la virgule vers la droite et vers la gauche. Si on n'obtient pas, aux extrémités du nombre, des groupes de 4 bits, alors on doit les compléter par des zéros non significatifs. On convertie, ensuite, chaque sous ensemble en base 16.

Exemple 1 : $(1110001110011001)_2 = (\ ?)_{16}$.




Exemple 4 : $(CB3)_{16} = (\ ?)_2$




☞ *Remarque* : Rappelez-vous que lorsque vous avez des nombres fractionnaires, les regroupements sont font :

- Pour la partie entière de droite (juste avant la virgule) vers la gauche.
- Pour la partie décimale de gauche (juste après la virgule) vers la droite

Exemple 3 : $(1111111100110,011110)_2 = ()_{16}$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $(1111111100110,011110)_2 =$ | (0001 | 1111 | 1110 | 0110 | , 0111 | 1000) |
| $(1111111100110,011110)_2 =$ | (1) | (F) | (E) | (6) | , (7) | (8) |

Ce qui donne $(1FE67,78)_8$

## I.3 – Arithmétique binaire

### I.3.1 – Addition binaire

Pour additionner deux nombres binaires, on procède exactement comme en décimale, mais en prenant en compte la table d'addition élémentaire suivante :

- 0+0 = 0 retenue 0
- 0+1 = 1 + 0 = 1 retenue 0
- 1 + 1 = 0 retenue 1
- 1 + 1 + 1 = 1 retenue 1

Exemple :

| | En binaire | | | | | | | | En décimal | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Retenus → | | *1* | *1* | *1* | *1* | *1* | *1* | *1* | | | *1* | |
| | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | | 1 | 1 | 5 |
| + | | | | 1 | 1 | 1 | 0 | 1 | + | | 2 | 9 |
| Résultat→ | = | **1** | **0** | **0** | **1** | **0** | **0** | **0** | = | **1** | **4** | **4** |

### 1.3.2 – Soustraction binaire

Dans la soustraction binaire, on procède comme en décimal. Quand la quantité à soustraire est supérieure à la quantité dont on soustrait, on emprunte 1 au voisin de gauche. En binaire, ce 1 ajoute 2 à la quantité dont on soustrait, tandis qu'en décimal il ajoute 10.

Table de soustraction binaire :

- 0 - 0 = 0 retenue 0
- 1 - 0 = 0 retenue 0
- 0 - 1 = 1 retenue 1 à soustraite au chiffre voisin de gauche
- 0 - 1 - 1 = 0 retenue 1 à soustraite au chiffre voisin de gauche

Exemple :

| | En binaire | | | | | En décimal | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 1 | 0 | 0 | 0 | 2 | 4 |
| | - | 0 | 0 | 1 | 1 | 1 | - | 7 |
| Retenus → | - | | *1* | *1* | *1* | | | |
| Résultat→ | = | **1** | **0** | **0** | **0** | **1** | = | **1** | **7** |

### 1.3.3 – Multiplication binaire

La multiplication binaire se réalise comme une multiplication décimale. Voici les règles de calcul à utiliser :

- 0 x 0 = 0
- 0 x 1 = 0
- 1 x 0 = 0
- 1 x 1 = 1

Elle consiste à faire une suite d'additions avec le multiplicande décalé vers la gauche. Cette opération est répétée autant de fois qu'il y a d'éléments binaires (à 1) dans le multiplicateur.

Exemple :

| | |
|---|---|
| 1101 | multiplicande |
| x 1011 | multiplicateur |
| 0001101 | |
| + 001101 | décalage 1 pas |
| + 1101 | décalage 3 pas |
| **10001111** | **résultat** |

☞ *Remarque* : Lorsqu'une opération donne plus de deux produits partiels, effectuez la somme de ces derniers 2 à 2 pour diminuer le risque d'erreur.

### 1.3.4 – Division binaire

La division binaire s'effectue à l'aide de soustractions et de décalages, comme la division décimale, sauf que les chiffres du quotient ne peuvent être que 1 ou 0. Le bit du quotient est 1 si on peut soustraire le diviseur, sinon il est 0.

*Exemple* :

Division décimale

```
  1 6 5 | 1 1
- 1 1   | 1 5
    5 5
  - 5 5
      0
```

Division binaire

```
  1 0 1 0 0 1 0 1 | 1 0 1 1
- 0 0 0 0         | 0 1 1 1 1
  1 0 1 0 0
-   1 0 1 1
    1 0 0 1 1
  -   1 0 1 1
      1 0 0 0 0
    -   1 0 1 1
        0 1 0 1 1
      -   1 0 1 1
          0 0 0 0
```

## I.4 – Représentation des nombres

- Coder une information consiste à établir une correspondance entre sa représentation externe (habituelle) et sa représentation interne dans la machine, qui est une suite de bits.
- La représentation (codification) des nombres est nécessaire afin de les stocker et de les manipuler par un ordinateur. Le principal problème est la limitation de la taille du codage : un nombre mathématique peut prendre des valeurs arbitrairement grandes, tandis que le codage dans la machine doit s'effectuer sur un nombre fixe de bits.

### I.4.1 – Codage des entiers naturels

- Les entiers naturels (positifs ou nuls) sont codés sur un nombre fixe d'octets (1 octet = 8 bits). On rencontre habituellement des codages sur 1 , 2, 4 octets, et plus rarement sur 8 octets (64 bits)
- Un codage sur ***n*** bits permet de représenter tous les entiers naturels compris entre ***0*** et $\boldsymbol{2^n-1}$.

Exemple : avec un octet (8 bits), on peut représenter (coder) les nombres appartenant à l'intervalle : $[0, 2^8-1] = [0,255]$.

### I.4.2 – Codage des entiers relatifs (nombres signés)

La représentation des entiers signés pose des problèmes surtout au niveau de la représentation du signe. Il existe plusieurs manières pour coder les nombres signés :

- Codage en signe + valeur absolue
- Codage en complément restreint (C à 1)
- Complément vrai (Cà 2)

Convention : Quelques soit le codage utilisé, le bit de poids fort est réservé pour la représentation du signe : Un nombre négatif a un bit de signe à 1 et un nombre positif a un bit de signe à 0.

#### *I.4.2.1 – Codage en signe + valeur absolue*

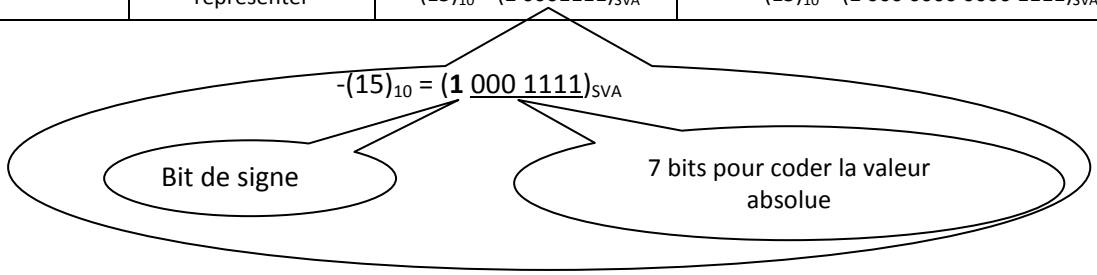
- Avec *n* bits, le $n^{ième}$ bit est réservé pour le signe et les *n-1* bits restants sont utilisés pour la représentation de la valeur absolue du nombre à coder.
- Un codage sur *n* bits permet de coder tous les entiers appartenant à l'intervalle $[-(2^{n-1}-1), +(2^{n-1}-1)]$

Exemple 1 : Avec 4 bits on peut coder les entiers relatifs suivants :

| 1111 | 1110 | 1101 | 1100 | 1011 | 1010 | 1001 | 0000 | 0110 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -7 | -6 | -5 | -4 | -3 | -2 | -1 | -0 | +0 | +1 | +2 | +3 | +4 | +5 | +6 | +7 |

Exemples 2 :

| | Sur n = 4 bits | Sur n = 8 bits | Sur n = 16 bits |
|---|---|---|---|
| Intervalle des valeurs couvertes | [-7, +7] | [-127,+127] | [-32767,+32767] |
| Statut de la valeur zéro | La valeur zéo est 0000 = 1000 | La valeur zéo est 00000000 = 1000000 | La valeur zéo est 0000 0000 0000 0000 = 1000 0000 0000 0000 |
| Exemples | $+(3)_{10} = (\mathbf{0}\ 011)_{SVA}$ | $+(3)_{10} = (\mathbf{0}\ 0000011)_{SVA}$ | $+(3)_{10} = (\mathbf{0}\ 000\ 0000\ 0000\ 0011)_{SVA}$ |
| | $-(3)_{10} = (\mathbf{1}\ 011)_{SVA}$ | $-(3)_{10} = (\mathbf{1}\ 0000011)_{SVA}$ | $-(3)_{10} = (\mathbf{0}\ 000\ 0000\ 0000\ 0011)_{SVA}$ |
| | $+(15)_{10}$ et $-(15)_{10}$ impossible à représenter | $+(15)_{10} = (\mathbf{0}\ 0001111)_{SVA}$ | $+(15)_{10} = (\mathbf{0}\ 000\ 0000\ 0000\ 1111)_{SVA}$ |
| | | $-(15)_{10} = (\mathbf{1}\ 0001111)_{SVA}$ | $-(15)_{10} = (\mathbf{1}\ 000\ 0000\ 0000\ 1111)_{SVA}$ |




Avantages : Facile à interpréter

Inconvénients :

- 2 représentations pour le 0 (+0 et -0)
- Problème d'addition de deux nombres de signes opposés.

### *I.4.2.2 – Codage en complément restreint (CR) ou complément à 1 (C à 1)*

- On obtient le complément à 1 d'un nombre binaire en inversant (le 1 devient 0, et le 0 devient 1) chacun de ses bits.
- Les nombres positifs sont codés comme dans le codage en signe plus valeurs absolue (SVA).
- Les nombres négatifs sont déduits des nombres positifs par complémentation bit à bit, c'est-à-dire : ***(-N) = CR(N)*** (en supposant que N est un nombre positif).
- Un codage sur ***n*** bits permet de coder tout entier relatif appartenant à l'intervalle $[-(2^{n-1}-1), +(2^{n-1}-1)]$

Exemple : coder +15 et -15 sur 8 bits en utilisant le codage en CR :

- $(+15)_{10} = (00001111)_2$
- $(-15)_{10} = CR(00001111) = (11110000)_{CR}$.

Inconvénient : Deux représentations pour le zéro

### *I.4.2.3 – Codage en complément vrai (CV) ou complément à 2 (C à 2)*

- Pour obtenir le complément vrai d'un nombre entier, il suffit d'ajouter un 1 à son complément restreint : **CV(*N*) = CR(*N*)+1** où *N* est un entier quelconque.
- En codage en complément à 2 :
  - Un nombre positif est représenté de la même manière qu'en codage en signe plus valeurs absolue.
  - Un nombre négatif est représenté par le complément vrai de son opposé (qui est bien évidemment positif)
- Un codage sur *n* bits permet de coder tout entier relatif appartenant à l'intervalle $[-2^{n-1}, +(2^{n-1}-1)]$

Exemple : coder +15 et -15 sur 8 bits en utilisant le codage en CV :

- $(+15)_{10} = (00001111)$
- $(-15)_{10} = CR(00001111) + 1 = (11110001)_{CV}$.

Avantages :

- Un seul codage pour 0
- Pas de problème pour effectuer l'opération d'addition

Inconvénient : Difficile à interpréter.

## I.5 – Soustraction par la méthode du complément

### I.5.1 – Complément restreint

Pour une machine travaillant en complément restreint, la soustraction sera obtenue par l'addition du complément restreint du nombre à soustraire avec le nombre dont il doit être soustrait et report de la retenue (ajout de la retenue).

S'il n'ya pas de retenue, cela signifie que le nombre est négatif. Il se présente sous la forme complémentée (complément restreint). Il suffira de retrouver le CR de ce résultat pour obtenir la valeur recherchée.

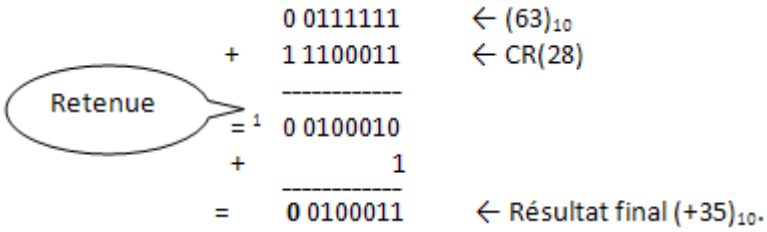
Exemple 1 : Effectuer l'opération suivante en utilisant la technique CR sur 8 bits :

$(63)_{10}$-$(28)_{10}$.

$(63)_{10} = (00111111)_2$

$(28)_{10} = (00011100)_2$.

$CR(28) = CR(00011100) = (11100011)_{CR}$.




Dans cet exemple, il y a une retenue, alors on l'a rajouté au résultat ce qui a donné un nombre positif : $(63)_{10}$-$(28)_{10} = (+35)_{10}$.

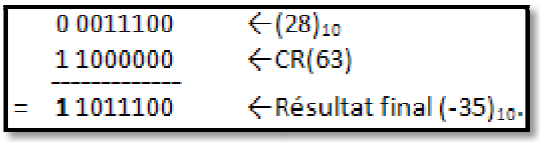
Exemple 2 : Effectuer l'opération suivante en utilisant la technique CR sur 8 bits :

$(28)_{10}$ - $(63)_{10}$

$(63)_{10} = (00111111)_2$

$(28)_{10} = (00011100)_2$.

$CR(63) = CR(00111111) = (11000000)_{CR}$.




Dans cet exemple, il n'ya pas de retenue, le résultat est négatif, alors on calcul son CR : $CR(11011100) = (00100011)_2 = (35)_{10}$ confirmant l'égalité : $(28)_{10}$ - $(63)_{10}$=$(-35)_{10}$.

### I.5.2 – Complément vrai

Le principe est le même que pour le CR sauf que cette fois-ci on ignore la retenue. Au lieu de travailler avec des CR, on détermine des compléments vrais.

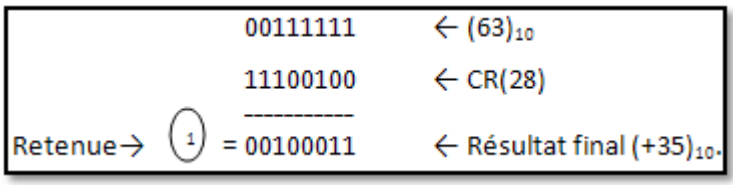
Exemple 1 : Effectuer l'opération suivante en utilisant la technique CR sur 8 bits :

$(63)_{10}$-$(28)_{10} = (63)_{10} + CV(28)$

$(63)_{10} = (00111111)_2$

$(28)_{10} = (00011100)_2$.

$CV(28) = CR(00011100) + 1 = (11100100)_{CR}$.




Dans cet exemple, il y a une retenue, alors il faut l'ignorer. On obtient un résultat positif : $(63)_{10}$-$(28)_{10} = (+35)_{10}$.

Exemple 2 : Effectuer l'opération suivante en utilisant la technique CV sur 8 bits :

$(28)_{10}-(63)_{10} = (28)_{10} + CV(63)$

$(63)_{10} = (00111111)_2$

$(28)_{10} = (00011100)_2$.

$CV(63) = CR(0\ 0111111) + 1 = (1\ 1000001)_{CV}$.

```
  0 0011100      ← (28)10
+ 1 1000001      ← CV(63)
  ---------
= 1 1011101      ← Résultat final (-35)10.
```

Dans cet exemple, il n'y a pas de retenue, le résultat est négatif alors on calcul son CV : CV(11011101) = CR(11011101) +1 = 00100010 + 1 = 00100011

On obtient au final : $(28)_{10}-(63)_{10} = (-35)10$.

## I.6 – Problèmes liés à la longueur des nombres

Rappel : En complément vrai sur *n* bits, les nombres sont compris entre $-2^{n-1}$ et $+(2^{n-1}-1)$.

### I.6.1 – Addition de deux nombres positifs

En additionnant deux nombres positifs, on peut obtenir un résultat négatif (le bit de signe du résultat à 1). Ceci est du au fait que le résultat n'appartient pas à l'intervalle autorisé avec le nombre de bits prévu.

Exemple : Effectuer l'opération suivante en utilisant la technique CV sur 8 bits :

$(+49)_{10} + (88)_{10}$

```
    0 0110001      ← (+49)10
+   0 1011000      ← (88)10
    ---------
=   1 0001001
    |
    Dépassement de capacité
```

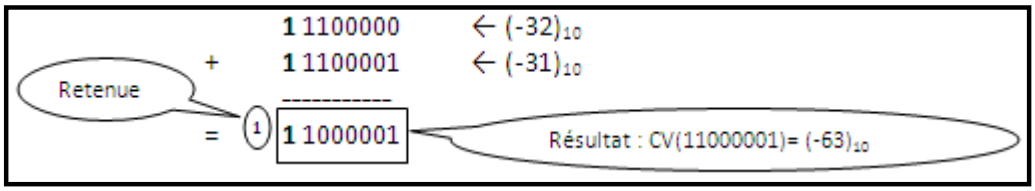
Dans cet exemple nous avons additionné deux nombres positifs, tous les deux tenant sur 8 bits. Malheureusement, nous avons obtenu un résultat qui est situé en dehors de l'intervalle des valeurs autorisées pour le codage sur 8 bits ($[-2^{n-1}$ et $+(2^{n-1}-1)] = [-128$ et $+127]$ avec n = 8). En effet, le résultat de 88 est en dehors de cet intervalle.

### I.6.1 – Addition de deux nombres négatifs

En additionnant deux nombres négatifs représentés par leurs CV (bit de signe à 1), on peut obtenir un résultat positif (le bit de signe du résultat à 0). En effet, il y a toujours une retenue car les bits de poids fort des nombres à additionner sont à 1.

Exemple 1 : Effectuer l'opération suivante en utilisant la technique CV sur 8 bits :

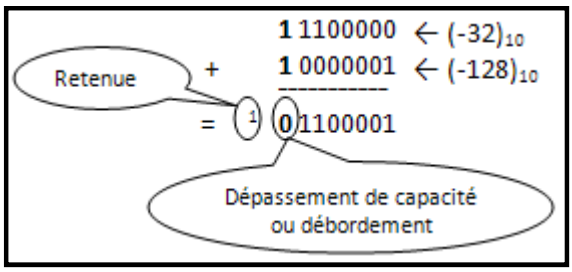
$(-32)_{10} + (-31)_{10}$




Dans l'exemple ci-dessus, nous avons additionné deux nombres négatifs (voir le bit de signe qui est à 1) et nous avons obtenu un nombre négatif (voir le bit de signe à 1). Malgré que nous avons obtenue une retenue, notre résultat est correcte, il faut juste ignorer cette retenue (car nous utilisons ici un codage en complément à 2 ou complément vrai).

Exemple 2 : Effectuer l'opération suivante en utilisant la technique CV sur 8 bits :

$(-32)_{10} + (-128)_{10}$




En ignorant la retenue, on obtient un résultat positif (bit de signe à 0) donc, on déduit qu'il y a débordement ou dépassement de capacité. En décimal : $(-32)_{10} + (-127)_{10} = (-159)_{10}$ et -159 n'est pas situé dans l'intervalle [-128 et + 127].

**Indicateur de dépassement de capacité** : Les calculateurs utilisent un indicateur de dépassement de capacité (Overflow) qui est mis à 1 si le bit de signe du résultat est 0 alors que les deux nombres à additionner sont négatifs ou lorsque le bit de signe du résultat est à 1 alors que les deux nombres à additionner sont positifs.