Codification et Représentation de l'Information (CRI)

<u>Chapitre 0</u> : Introduction

<u>Chapitre 1</u> : Codification et représentation des nombres

a. Les Entiers Positifs

a.1 Systèmes de numération

Définition d'un système de numération

Définition d'une Base

Changement de Base

a.2 Arithmétique

Opérations arithmétiques binaires

b. Les Entiers Négatifs

b.1 Représentation des nombres négatifs en SVA (signe et valeur absolue)

b.2 Représentation des nombres négatifs en CP1 (complément à 1)

b.3 Représentation des nombres négatifs en CP2 (complément à 2)

a.2 Arithmétique

Opérations en C2

c. Les nombres réels

c.1 Représentation des nombres Réels en virgule fixe

c.2 Représentation des nombres Réels en virgule flottante

c.3 Arithmétique

Opérations en virgules flottante

<u>Chapitre 2</u> : Codification et représentation α-Numérique

a. Le code ASCII
b. Le code BCD
c. Le code GRAY
d. L'Unicode

<u>Chapitre 3</u> : Algèbre de Boole

a. Introduction
b. Terminologie
c. Opérations de base
d. Théorèmes et postulats de l'algèbre de Boole
e. Expressions booléennes
f. Les tables de vérité
g. **Les fonctions booléennes**

g.1-Simplification des fonctions booléennes

g.2 Application des fonctions booléennes aux circuits combinatoires

L'Additionneur, le Décodeur, le Multiplexeur

Introduction

Le langage est la faculté d'exprimer et de communiquer sa pensée au moyen de signes.

Notre langage écrit utilise un code basé sur 26 lettres (majuscules et minuscules), 10 chiffres, des symboles de ponctuation et des signes mathématiques.
Grâce à ce code et à ces règles nous pouvons transmettre des informations, donner des instructions, dénombrer...

Bien que les ordinateurs soient qualifiés "d'intelligence artificielle", ils n'ont aucune faculté d'appréhender le monde extérieur.
Leur seule intelligence réside dans leur rapidité d'exécution de combinaisons d'ordre à deux états (0 , 1)équivalent à (éteint , allumé)

Le terme bit signifie « binary digit », c'est-à-dire 0 ou 1 en numérotation binaire. Il s'agit de la plus petite unité d'information manipulable par une machine numérique.
Il est possible de représenter physiquement cette information binaire par un signal électrique ou magnétique

La **codification** consiste à établir une correspondance entre la représentation externe de l'information dont nous sommes utilisateurs et sa représentation interne dans la machine, qui est une suite de bits (suite de 0 et 1)

Systèmes de numération

1- Système de numération

Un système de numération est défini par un ensemble de symboles (chiffres ou lettres) et des règles d'écriture pour le positionnement de ces symboles.

L'exemple le plus répandu de système de numération est la numération décimale.

Ce système est composé de dix chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Un nombre est représenté par une succession de chiffres. Chaque chiffre possède un poids.

Exemple :

$4523 = 4 \times 10^3 + 5 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$

Le poids de 4 est 3, le poids de 5 est 2 … le poids de 3 est 0

(le premier chiffre à gauche est le chiffre de poids fort)

2- Bases

Un système de numération à base B est défini par B symboles (chiffres ou lettres)

Soit N un nombre de n chiffres représenté en base B

$N = a_{n-1}a_{n-2}....a_i...a_0 \quad \forall i \; a_i < B$ (Tous les symboles sont strictement inférieurs à B)

Quelque soit la base, la forme polynomiale de N est :

$N = a_{n-1} * B^{n-1} + a_{n-2} * B^{n-2} .... + a_i * B^i .... + a_0 * B^0$

2.1- Système Binaire (Base 2) :

Ce système utilise 2 chiffres {0, 1} il est utilisé principalement dans le fonctionnement des ordinateurs

2.2- Système Octal (Base 8) :

Ce système utilise 8 chiffres {0, 1, 2, 3, 4, 5, 6, 7}

2.3- Système Hexadécimal (Base 16) :

Ce système utilise 16 symboles (chiffres et lettres) : {0,1,2,3,4,5,6,7,8,9,A,B C,D, E, F}

. A est l'équivalent de 10 en décimal, B est l'équivalent de 11…F est l'équivalent de 15.

Cette base est très utilisée dans le monde de la micro informatique notamment pour la représentation des adresses mémoire.

3- Changement de base

3.1 De la Base 10 vers une base B

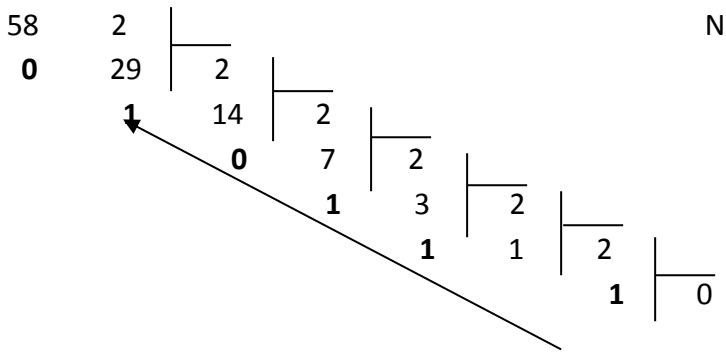
3.1.1 Conversion de nombres entiers de la base 10 vers une base B

La règle à suivre est celle des divisions successives
On divise le nombre par B on sauvegarde le reste puis on divise le quotient par B
Ainsi de suite jusqu'à obtention d'un quotient nul
La suite des restes correspond au nombre de la base visée
Le premier reste correspond au poids faible et le dernier au poids fort

Exemples : Soit N un nombre représenté en base 10

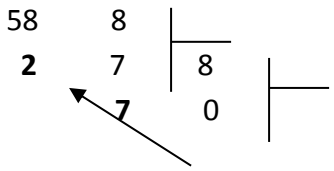
$N = (58)_{10}$

Représentation binaire



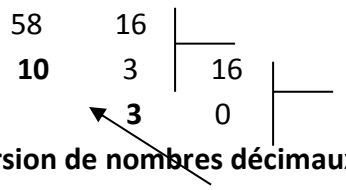
$N = (1\,1\,1\,0\,1\,0)_2$

Représentation octale



$N = (7\,2)_8$

Représentation hexadécimale



$N = (3\,A)_{16}$

3.1.2 Conversion de nombres décimaux de la base 10 vers une base B

La partie entière est traitée comme indiqué précédemment.
Pour la partie décimale la règle à suivre est celle des multiplications successives.
On multiplie la partie décimale par B, on sauvegarde la parie entière du résultat

Puis on multiplie la nouvelle partie décimale par B
Ainsi de suite jusqu'à obtention d'une partie décimale nulle
La suite des parties entières obtenues correspond à la partie décimale de la base visée.

Exemple : Convertir $(58,375)_{10}$ en base 2

0,375	0, 750	0,500
x 2	x 2	x 2
= **0**,750	= **1**,500	= **1**,000

$(5\ 8)_{10} = (111010)_2$ et $(0,375)_{10} = (0,011)_8$ donc $(5\ 8,375)_{10} = (111010,011)_8$

3.2 Passage de la Base B vers une base 10

Pour convertir un nombre d'une base B vers la base 10 il suffit de représenter le nombre sous sa forme polynomiale et calculer la somme de tous les termes.

Exemples : Soit N un nombre représenté en base B

B = 2 $N = (1111011)_2$

$N = 1x2^6 + 1x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0$

$N = 64 + 32 + + 16 + 8 + 0 + 2 + 1$

$N = (123)_{10}$

B = 16 $N = (7B)_{16}$

$N = 7x16^1 + 11x16^0$

$N = 112 + 11$

$N = (123)_{10}$

Pour les nombres décimaux, on utilisera des exposants négatifs.

Exemple : Convertir $(7B,84)_{16}$ en base 10

$N = 7x16^1 + 11x16^0 + 8x16^{-1} + 4x16^{-2}$

$N = 112 + 11 + 0,5 + 0,015625 = (123,515625)_{10}$

3.3 Passage d'une Base B1 vers une base B2

Pour passer d'une base B1 vers une base B2 il faut 2 opérations.

Il faut d'abord passer de la base B1 vers la base 10 puis de la base 10 vers la base B2

Exemple : Convertir $(3141)_5$ vers la base 16

$(3141)_5 = (421)_{10}$

$(421)_{10} = (1A5)_{16}$

$(3141)_5 = (1A5)_{16}$

3.4 Applications aux bases 2, 8 et 16

3.4.1 - Pour convertir un nombre de la base 2 vers la base 8, il faut découper ce nombre en groupes de 3 bits et remplacer chaque groupe par sa valeur octale (en partant de la droite).

Exemple : Convertir $(1011010)_2$ vers la base 8

$(001\ 011\ 010)_2 = (1\ 3\ 2)_8$

Si le nombre de bits n'est pas un multiple de 3, il faut compléter par des 0 à gauche.

3.4.2 - Pour convertir un nombre de la base 8 vers la base 2, il suffit de transcrire chaque chiffre de ce nombre en binaire sur 3 bits (en partant du poids faible).

Exemple : Convertir $(645)_8$ vers la base 2

$(6\ 4\ 5)_8 = (110\ 100\ 101)_2$

3.4.3 - Pour convertir un nombre de la base 2 vers la base 16, il faut découper le nombre en groupes de 4 bits et remplacer chaque groupe par sa valeur hexadécimale (en partant de la droite).

Exemple : Convertir $(1011010)_2$ vers la base 16

$(0101\ 1010)_2 = (5\ A)_{16}$

Si le nombre de bits n'est pas un multiple de 4, il faut compléter par des 0 à gauche.

3.4.4 - Pour convertir un nombre de la base 16 vers la base 2, il suffit de transcrire chaque chiffre de ce nombre en binaire sur 4 bits en partant du poids faible.

Exemple : Convertir $(1A5)_{16}$ vers la base 2

$(1\ A\ 5)_{16} = (0001\ 1010\ 0101)_2$

3.4.5 - Pour convertir un nombre de la base 8 vers la base 16 ou inversement, on peut transiter par la base 10 mais on peut également passer par la base 2

Base 8 ⟶ base 2 ⟶ base 16

Base 16 ⟵ base 2 ⟵ base 8

Exemple : Convertir $(232)_8$ vers la base 16

$(2\ 3\ 2)_8 = (010\ 011\ 010)_2 = (0\ 1001\ 1010)_2 = (9\ A)_{16}$

Remarque

Pour la conversion des nombres décimaux, on sépare la partie entière de la partie décimale.
La partie entière est traitée comme indiqué précédemment.
La conversion de la partie décimale se fait de droite à gauche.

$(11101,01011)_2 = (011\ 101\ ,\ 010\ 110)_2 = (\ 35,26)_8$

$(11101,01011)_2 = (0001\ 1101\ ,\ 0101\ 1000) = (1D,58)_{16}$

On peut compléter la partie décimale si nécessaire

Tableau de conversion Décimal - Binaire

Décimal	Binaire
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Arithmétique binaire (Exemples d'opérations réalisées en binaire)

Addition

$^{1}1\ ^{1}1\ ^{1}1\ ^{1}0\ 1$	2 9
+ 1 1 1	+ 7
= 1 0 0 1 0 0	= 3 6

Soustraction

```
  1 1 ¹1 ¹0 1          29
- ₁0 ₁1 1 1         - 7
-----------          ----
= 1 0 1 1 0          = 22
```

Multiplication

```
      1 1 1 0 1          ¹0 ¹1 1 1 0 1          29
x         1 1 1     +    1 1 1 0 1 0      x    7
-----------------    ------------------     -----
      1 1 1 0 1      =  1 0 1 0 1 1 1 1     = 203
.   1 1 1 0 1        +  1 1 1 0 1 0 0
. 1 1 1 0 1          =  1 1 0 0 1 0 1 1
-------------
1 1 0 0 1 0 1 1
```

Si le nombre de lignes des résultats est supérieur à 2, il faut faire les additions 2 par 2

Divisions

```
11101 | 111          29 | 7
------|----          ---|---
00001 | 100           1 | 4
      |
```

```
11000001 | 101          193 | 5
---------|-----          ----|---
001000   | 100110         43 | 38
  00110  |                 3  |
    0011 |
```

4- Représentation des nombres entiers relatifs

Les entiers relatifs sont représentés en binaire dans un format fixe (on ne peut pas comparer par exemple un nombre de 5 bits avec un nombre de 8 bits).

Le bit de poids fort représente le signe, il est égal à 0 si le nombre est positif et il est égal à 1 si le nombre est négatif

4.1- représentation en signe et valeur absolue (SVA)

Le bit de poids fort représente le signe du nombre (0 pour + et 1 pour –)

Les autres bits représentent la valeur absolue du nombre.

∀ X représenté sur un format de n bits $-(2^{n-1}-1) < X < +(2^{n-1}-1)$

Exemples : Représentation sur 8 bits

1- A = -25
A est négatif donc le bit de poids fort est égal à 1

l-25l = 0 0 1 1 0 0 1 sur 7 bits

A = 1 0 0 1 1 0 0 1

2- B = + 525

Ce nombre nécessite 9 bits au minimum donc il ne peut pas être représenté sur un format de 8 bits

n = 8 donc $-(2^7-1) < X < +(2^7-1)$ c a d $-127 < X < +127$

3- C = 0 (on a 2 représentations du zéro +0 et -0)
+ 0 = 0 0 0 0 0 0 0 0 ou 0 = 1 0 0 0 0 0 0 0

4.2- Représentation en complément à 1 (C1)

Le complément à 1 d'un nombre est obtenu en inversant tous les bits.

Le bit de poids fort représente le signe du nombre (0 pour + et 1 pour –).

∀ X représenté sur un format de n bits $-(2^{n-1}-1) < X < +(2^{n-1}-1)$

Exemple : Représentation sur 8 bits

A = 1 0 0 1 1 0 0 1 C1(A) = 0 1 1 0 0 1 1 0

4.3 Représentation complément à 2 (C2)

Le complément à 2 d'un nombre est égal au complément à 1 du nombre auquel on ajoute 1

∀ X représenté sur un format de n bits $-(2^{n-1}) < X < +(2^{n-1}-1)$

$\forall (X) \in [-2^{n-1}, +2^{n-1}-1]$ on a **(-X)= C2 (X)**

Exemples : représentation sur 8 bits

A = + 25
A = 0 0 0 1 1 0 0 1

B = - 30
-B = + 30 - B = 0 0 0 1 1 1 1 0 B = C2 (- B)=> B = 1 1 1 0 0 0 1 0
Pour représenter un nombre négatif en C2 on passe par son opposé et on applique la règle
(-X)= C2 (X)

Exemple avec n=4:

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

$E = [-2^{4-1}, +2^{4-1}-1]$
$E = [-8, +7]$

0 = 0000 -0 = 0000 (en C2 + 0 = - 0)
5 = 0101 -5 = 1011
3 = 0011 -3 = 1101
$8 \notin E$ -8 = 1000

Pour convertir en décimal un nombre négatif X représenté en C2
il faut passer par son opposé –X et appliquer la règle -X = C2 (X)
– X est positif donc on le convertit directement en décimal
puis on lui affecte le signe –

Exercice :
Soit A et B 2 nombres représentés en C2 sur 8 bits, trouver leurs valeurs décimales.

A = 00110101 B = 11101011
A est positif donc $A = 2^5 + 2^4 + 2^2 + 2^0$ A = 53
B est négatif alors - B = C2 (B) = 00010101 = 21 donc B = -21

Opérations arithmétiques en complément à 2

Dans cette représentation la soustraction doit est traitée comme une addition.
Pour les opérations arithmétiques on doit appliquer la règle suivante :
$\forall (A,B) \in E$ si $X = A + B$ alors $X \in E$

Exemples d'opérations arithmétiques avec n=4 (E = [- 8,+ 7]) :

A = 5 0 1 0 1
B = 1 + 0 0 0 1

X = A + B = 0 1 1 0
X = 6 résultat correct (pas de dépassement)

A = 5 0 1 0 1
B = 4 + 0 1 0 0
X = A + B = 1 0 0 1
X = -7 résultat faux (dépassement) en effet (5 + 4) ∉ E

A = 5 0 1 0 1
B = -3 + 1 1 0 1
X = A - B = A +(-B) = 0 0 1 0
X = 2 résultat correct (pas de dépassement)

Traitement du dépassement de capacité pour une addition:

Si les deux opérandes sont de même signe et que le résultat est du même signe que les opérandes, il n'y a pas de dépassement

Si les deux opérandes sont de même signe et que le résultat est du signe opposé alors il y a dépassement.

Si les deux opérandes sont de signes opposés, il n'y a jamais de dépassement

Remarque :

Il ne faut pas confondre les 2 expressions
« Représenter X en format C2 » et « Donner le C2 de X »

Exemple :

X = 35

Représenter X en format C2 revient à écrire 35 en C2 c a d X = 00100011

Donner le C2 de X reviens à écrire l'opposé de X C2(X) = -X = 11011101

5. Représentation d'un nombre en virgule flottante dans le format IEEE 754

Format (32 bits)
Bit du signe (1 bit)
Exposant (8 bits)
Mantisse (23 bits)

Soit X un nombre réel écrit en base 2, pour représenter X en virgule flottante il faut décaler la virgule jusqu'à ce qu'il reste un seul 1 dans la partie entière et on rectifie l'exposant à chaque décalage. On obtient alors un nombre au format suivant :

$X_2 = \pm 1, M*2^e$

M est la mantisse , e est l'exposant , S est le signe (0 si X>0 1 si X<0)

La mantisse est la partie décimale de X (23 bits)
Le 1 précédant la virgule n'est pas codé en machine et est appelé bit caché

L'exposant est représenté par sa valeur réelle (e) plus 127_{10},
la valeur obtenue est la caractéristique (c) (8 bits)

$c = e + 127_{10}$

Exemple: Représentation de $X_{10} = 7,625$

Convertir en base 2

$X_2 = 111,101$

Décaler la virgule de telle sorte qu'il ne reste qu' un seul 1 dans la partie entière

$X_2 = 1,11101\ 2^2$

M = 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

e = 2

$c = 2 + 127 = 129_{10} = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1$

Signe = 0

La représentation de X en virgule flottante est

0	1 0 0 0 0 0 0 1	1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

X = 4 0 F 4 0 0 0 0 (Forme condensée en hexadécimal)

5.1 - Conversion d'un nombre X représenté en virgule flottante vers le décimal :

Soit X un nombre représenté en virgule flottante sur 32 bit, pour retrouver la valeur correspondante, il suffit de le décomposer comme suit :

Un bit pour le signe, 8 bits pour la caractéristique et 23 bits pour la mantisse.

On en déduit directement le signe et la partie décimale puis on calcule l'exposant (e).

Si le bit de signe = 0 alors X>0 sinon X <0

Partie décimale = Mantisse

Partie entière = 1

e = c – 127

Exemple : Calculer la valeur décimale de X

X = C 1 E 9 0 0 0 0

X = 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1	1 0 0 0 0 0 1 1	1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

S = 1 => X < 0

Mantisse = 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

c = 1 0 0 0 0 0 1 1 = 131_{10}

e = 131 – 127

e = 4

X_2 = - 1, 1101001* 2^4 = - 11101,001

X_{10} = -29,125

Remarque :

Il existe également un format double précision qui permet de représenter un ensemble de nombres beaucoup plus vaste avec une plus grande précision.
Dans ce cas on a la représentation suivante :

Format (64 bits)
Bit du signe (1 bit)
Exposant (11 bits)
Mantisse (52 bits)

5.2 - Addition de deux nombres en virgule flottante IEEE de simple précision

Pour additionner 2 deux nombres en virgule flottante il faut qu'ils aient le même exposant.

Soit A = 4 0 D 9 0 0 0 0 et B = 3 E 9 A 0 0 0 0 en virgule flottante IEEE Effectuer A + B

A =

0	1 0 0 0 0 0 0 1	1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

c = 1 0 0 0 0 0 0 1 =129_{10} => e = 129 – 127 => e = 2

M = 1 0 1 10 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Donc A = 1 , 1 0 1 1 0 0 1 * 2^2

B =

0	0 1 1 1 1 1 0 1	0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

C = 01111101 = 125_{10} => e = 125 – 127 => e = -2

M = 0 0 1 1 01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Donc B = 1 , 0 0 1 1 0 1 * 2^{-2}

On met B au même exposant que A en déplaçant la virgule de 4 positions vers la gauche.

B = 0 , 0 0 0 1 0 0 1 1 0 1 * 2^2

On effectue A + B

1 , 1 0 1 1 0 0 1 0 0 0 * 2^2

\+ 0 , 0 0 0 1 0 0 1 1 0 1 * 2^2

= 1 , 1 1 0 0 0 1 0 1 0 1 * 2^2

On replace le résultat sous la forme IEEE

0	1 0 0 0 0 0 0 1	1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0

A + B = 4 0 E 2 A 0 0 0

<u>Remarque</u>

S la partie entière du résultat est supérieure à 1, on décale d'un rang à gauche la virgule et on augmente de 1 l'exposant :

Par exemple si A + B = 10,11001101 * 2^2 => A + B = 1,011001101 * 2^3