Chapitre 1. Système de numération et codage

I- Introduction :

Les machines qui assurent le traitement l'information, sont constituées de circuits numériques qui ne possèdent que deux états électriques stables ; représentées par les valeurs binaires « 0 » et « 1 ».

Généralement, les informations traitées ne sont pas numériques, elles sont souvent exprimées sous forme de texte : caractères alphanumériques 0,1,….9, a, b …, z et caractères spéciaux. Afin de traiter et manipuler ces informations dans la machines, une codification sous forme binaire est nécessaire.

I-1 Définition :

La numération permet de représenter un mot (ou nombre) par juste-a-position ordonnée de variable (ou symboles) pris parmi un ensemble. Basée sur trois notions de : Base du système, de digit (chiffres) du système et du poids du digit selon son rang.

La base d'un système est nombre entier quelconque soit B.

I -2 Système de numération

I- 2-1 La représentation des nombres

L'écriture d'un nombre consiste à associer plusieurs chiffres dans un ordre déterminé.

Exemple : N_B = (ACDEF) $_B$

Chaque chiffre (digit) intervient avec un poids différent selon son rang.

Ce poids est de B^0 pour le 1^{er} digit de nombre (dans le cas de nombre *N* : F) ; B^1 pour le 2^{eme} digit (E) ; B^2 pour le 3^{eme} digit (D) ;…...B^{n-1}.

D'où : le nombre N exprimé dans le système de base B vaut ;

$N= A*B^4 + C*B^3 + D*B^2 + E*B^1 + F*B^0$

Remarque : Dans un système de numération de base B ; tout nombre est représenté par une suite de chiffres allant de 0 a B-1. Les chiffres sont appelés des digits, la position de chacun représente une puissance entière (positive ou négative) de la base B ; la place occupée par le chiffre (digit) dans un nombre représente son rang.

Exemple : Soit le nombre 93452 nous obtenons le tableau suivant :

Chiffre (Digits)	9	3	4	5	2
Rangs	4	3	2	1	0
Puissance	10^4	10^3	10^2	10^1	10^0
Pondération	$9*10^4$	$3*10^3$	$4*10^2$	$5*10^1$	$2*10^0$
Forme polynomiale	$9*10^4 + 3*10^3 + 4*10^2 + 5*10^1 + 2*10^0$				
Poids fort (MSB)	9 : Bite le plus à gauche				
Poids faible (LSB)	2 : Bite le plus à droite				

Exemple 2 : dans le cas de nombre $(178,25)_{10}$, cela signifie :

$1*10^2 +7*10^1+8*10^0+2*10^{-1}+5*10^{-2}$ (cette forme dit : forme Polynomiale)

C.-à-d. : $(178,25)_{10} = 1*10^2 +7*10^1+8*10^0+2*10^{-1}+5*10^{-2}$

Remarque : Le signe «-» du rang, indique qu'on est en train de représenté la partie fractionnelle du nombre.

$178,25 = (178,25)_{10}$	$1*10^2$ +	$7*10^1$ +	$8*10^0$ +	$2*10^{-1}$ +	$5*10^{-2}$
Rangs	2	1	0	-1	-2

I-2-1-1 Le système décimal : appelé aussi base 10 ; peut prendre 10 valeurs :

0, 1, 2, 3, 4, 5, 6, 7, 8,9.

Par exemple : la représentation précédente (exemple 2) du nombre réel 178,25.

I-2-1-2 Le système binaire ou base 2 (B=2) :

Avec le chiffre **0** et **1,** qu'on appelle dans ce cas des Bits (binary digits), ce système de numération est le plus utilisé dans les calculateurs numériques.

Exemple : En numération binaire :

$(1011,01)_2 =$	$1*2^3$ +	$0*2^2$ +	$1*2^1$ +	$1*2^0$ +	$0*2^{-1}$ +	$1*2^{-2}$
Rangs	3	2	1	0	-1	-2

I-2-1-3 Le système Octale ou base 8 :

Peut prendre que les chiffres : **0, 1, 2, 3, 4, 5, 6, 7.**

***Exemple* :** le nombre 472 exprimé en octal signifie **:**

$(472)_8 = 4*8^2+7*8^1+2*8^0$

$= 4* 64 + 7*8 + 2* = 256 +56 +2 = 314$ D'où $(472)_8 = (314)_{10}$

I-2-1-4 Le système Hexadécimal ou base 16 :

La base B est égale à 16 ; il y a **16** chiffres (digits). Les digits de 0 à 9 sont les chiffres du système décimal et les digits de 10 à 15 sont les six (06) premières lettres de l'alphabet :

0, 1, 2, 3, 4, 5, 6, 7, 8,9, A, B, C, D, E, F

***Exemple* :** le nombre **3BA2** exprimé en Hexadécimal signifie **:**

Chiffres (digits)	**3**	**B**	**A**	**2**
Rang	**3**	**2**	**1**	**0**
Puissance	$\mathbf{16^3}$	$\mathbf{16^2}$	$\mathbf{16^1}$	$\mathbf{16^0}$
Pondérations	$\mathbf{3*16^3}$	$\mathbf{B*16^2 = 11*16^2}$	$\mathbf{A*16^1= 10*16^1}$	$\mathbf{2*16^0}$
Forme polynomiale	$\mathbf{3*16^3 +11*16^2 +10*16^1 + 2*16^0}$			

Alors : $(3BA2)_{16}$ = $3*16^3 + 11*16^2 + 10*16^1 + 2*16^0$
= 12288 +2816+160+2
= 15266
D'où : **$(3BA2)_{16}$** = 15266 (équivalent en décimal du **$(3BA2)_{16}$**)

Conclusion : soit d'une manière générale dans le système de numération de base B, on a :

$N_B = (a_{n-1} \ldots\ldots a_1\ a_0\ a_{-1}\ a_{-2} \ldots)_B$
$N_B = a_{n-1}*B^{n-1} + \ldots. + a_1*B^1 + a_0*B^0 + a_{-1}*B^{-1} + \ldots\ldots$

- Toute nombre de base B= (2, 4, 8 et de 16), sont codés dans un nombre de bits, qui est déterminé de la manière suivante :
  Base B : $B = 2^n$ ⟶ on utilise « n » bit

Exemple :

Base $2 = 2^1$ ⟶ 1 : on utilise un seul bit.
Base $8 = 2^3$ ⟶ 3 : on utilise trois (3) bits

Dans le tableau suivant on illustre les codes, binaire, décimal et le hexadécimal :

Décimal	Binaire	Octale	hexadécimal
0	0000	000	0
1	0001	001	1
2	0010	010	2
3	0011	011	3
4	0100	100	4
5	0101	101	5
6	0110	110	6
7	0111	111	7
8	1000	/	8
9	1001	/	9
10	1010	/	A
11	1011	/	B
12	1100	/	C
13	1101	/	D
14	1110	/	E
15	1111	/	F

I-2-2 Conversion d'un système de numération en un autre (changement de base) :
I-2-2-1 : Base B vers base 10 :

Deux exemples suffiront pour expliquer la méthode :

- B= 2 : $(100101)_2 = 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0$
  = 32 + 0 + 0 + 4 + 0 +1 = 32 + 4 + 1
  = 37
  D'où : $(100101)_2 = (37)_{10}$
- $(3BA2)_{16} = 3*16^3 + 11*16^2 + 10*16^1 + 2*16^0$
  = 12288 +2816+160+2
  = 15266 D'où : $(3BA2)_{16} = (15266)_{10}$

I-2-2-2 : Base 10 vers base B

- Une première méthode consiste à ***soustraire*** successivement la plus grande puissance de B (Base) ;

Exemple : convertir le nombre $(363)_{10}$; en base 2 et puis en base 16 par exemple

On a : $363 = 1*2^8+107$

$107 = 1*2^6+43$ (le rang : 8, 6, 5, 3, 1,0)

$43 = 1*2^5+11$ → $(363)_{10} = (101101011)_2$

$11 = 1*2^3+3$

$3 = 1*2^1+1$

$1 = 1*2^0+0$

(quatre bits)

En base 16 :

$363 = 1*16^2+107$

$107 = 6*16^1+11$

$11 = B*16^0$

D'où : $(363)_{10} = (16B)_{16}$

Comme aussi on peut : $(363)_{10} = (101101011)_2$

(via nombre de bit : $16 = 2^4$)

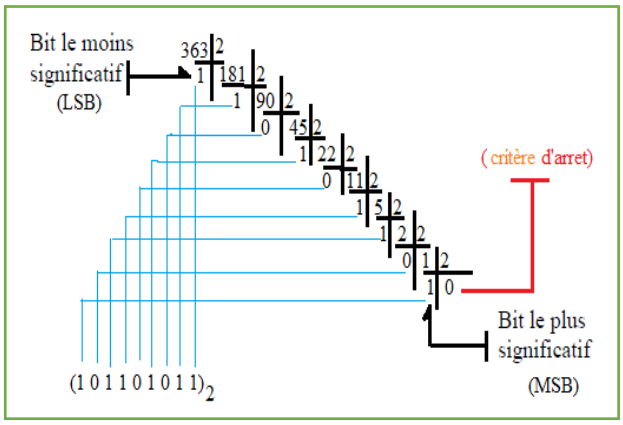
$(363)_{10} = (16B)_{16}$

- Une deuxième méthode consiste a ***divisé*** par « B » (la base) autant de fois que cela est nécessaire pour obtenir un quotient nul, puis on les restes dans le sens inverse ou ils ont été obtenus.

Exemple :

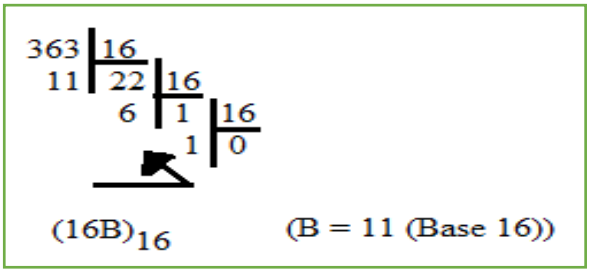
* L'équivalent de $(363)_{10}$ en binaire est :

$(363)_{10} = (101101011)_2$



- L'équivalent de $(363)_{10}$ en hexadécimal est :

$(363)_{10} = (16B)_{16}$

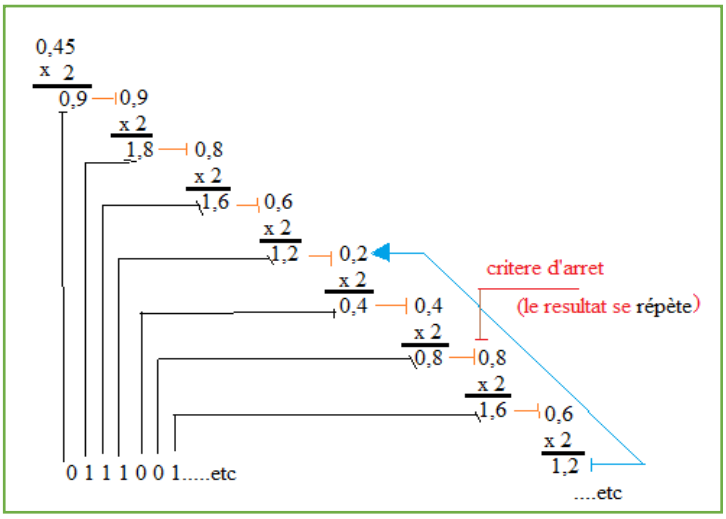


Remarque : Quand il y a une partie fractionnaire on le convertit par multiplications successives par la base B autant de fois que cela est nécessaire pour obtenir la précision voulu.

Exemple :

L'équivalent de $(0,45)_{10}$ en base binaire est :

$$(0,45)_{10} = (0111001)_2$$



I-2-2-3 : Base 2^n vers base 2 et base 2 vers Base 2^n

- **Pour B = 16 = 2^4** ; on convertit chaque chiffre en base 2 (voir tableau de page 4), puis on juste-a-posé les résultats (n = 4 , on utilise quatre bits) ;

Exemple : convertir (3A9) en base 2 ?

3	A	9	Résultat en base 2
0011	1010	1001	(001110101001)

$(3A9)_{16} = (001110101001)_2 = (1110101001)_2$

- **Pour B = 8 = 2^3 ;** on convertit chaque chiffre en base 2 (voir tableau de page 4), puis on juste-a-posé les résultats (n = 3, on utilise quatre bits) ;

Exemple : convertir $(264)_8 = (\ ?)_2$

2	6	4	Résultat en base 2
010	011	100	(010011100)

$(264)_8 = (010011100)_2$

$(264)_8 = (10011100)_2$

Remarque : Dans la conversion inverse (base 2 vers Base 2^n**),** on découpe le nombre binaire en tranches de « n » chiffres qu'en convertit.

Exemple :

- n = 4 : concéderons le nombre suivant : $(111001110011)_2$.

$$\underbrace{(1110}_{14}\ \underbrace{0111}_{7}\ \underbrace{0011)_2}_{3} = (E72)_{16}$$

- **n = 3 :** concéderons le nombre suivant : $(010110100)_2$

$$\underbrace{(010}_{2}\ \underbrace{110}_{6}\ \underbrace{100)_2}_{4} = (264)_8$$

I-2-2-4 : De base « i » vers « j »

- Si « i » et « j » sont tous les deux des puissances de « 2 », on utilise la base 2 comme relais, c.-à-d. i ⟶ 2 ⟶ j

- Si « i » et « j » ne sont pas tous les deux des puissances de « 2 », on utilise la base 10 comme relais, c.-à-d. i → 10 → j

I-3 : Code numériques

Chaque position de chiffres a une valeur intrinsèque (poids) par exemple 1, 10,100, 1000,…, en numération décimal, ou, 1,2, 4,8 … en numération binaire. Ces codes ont des propriétés arithmétiques intéressantes (avoir dans la prochaine partie du cours). Les plus courants sont :

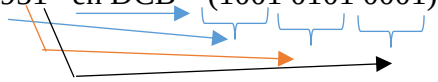
I-3-1 : Code e BCD (code : 8421)

Son principe est simple : chaque chiffre du nombre décimal est codé en binaire, de manière indépendante.

Exemple : le nombre $(951)_{10}$ sera codé

en DCB comme suit :

951 en DCB (1001 0101 0001) DCB



(Codé de manière indépendante)

N	8	4	2	1
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

Tab.a

- Le code BCD(8421) comprend :
  Les combinaisons 0 à 9 uniquement (voir : Tab.a)

I-3-1-1 : Code excédent 3

Dans ce code chaque chiffre décimal est codé en son équivalent binaire additionné de 3.

Exemple : donner le code excèdent de $(951)_{10}$:

1001 0101 0001 en DCB

\+ 11 11 11

1100 1000 0100 en Excédent 3

I-3-2 : Code GRAY (binaire réfléchi)

Code GRAY permet la représentation uniquement que des nombres décimaux, à chaque augmentation d'une unité du nombre. Un seul dit binaire équivalent change de valeur par rapport au nombre binaire précédent.

N	Binaire	Gray		
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0

5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1

On remarque que dans ce code ; chaque bit a une pondération de : $\pm (2^{n+1}-1)$

n : le rang du bit dans le nombre

Le premier chiffre 1, rencontré en partant de la gauche est affecté du signe + ; le 2^{ieme} 1 est affecté du signe - ; le 3^{eme} du signe + etc…

Exemple : Soit le nombre 1011 en GRAY

Chiffres	1	0	1	1	Résultat
Rang (n)	3	2	1	0	
Puissances	2^3	2^2	2^1	2^0	
Pondérations	$+ (2^3-1)$	$- (2^2-1)$	$+ (2^1-1)$	$-(2^0-1)$	
Soit	+15	-3	+1	0	= 13

I-3-3 : codes alphanumériques

C**e** code est basé sur la comparaison et pas sur le calcule. Sont des codes destinés à la transmission d'information, il comprend au moins 36 caractères (10 chiffres et 26 lettres).on peut citer deux type : code ISO et code ASCII.

b7b6b5 / b4b3b2b1	000	001	010	011	100	101	110	111
0000	NUL	DLE	Espace	0	@	P	'	P
0001	SOH	DC_1	!	1	A	Q	a	
0010	STX	DC_2	"	2	B	R	b	
0011	ETX	DC_3	≠	3	C	S	c	
0100	EOT	DC_4	$	4	D	T	d	
0101	ENQ	NAK	%	5	E	U	e	
0110	ACK	SYN	&	6	F	V	f	
0111	BEL	ETB	'	7	G	W	g	
1000	BS	CAN	(	8	H	X	h	
1001	HT	EM	)	9	I	Y	i	
1010	LF	SUB	*	:	J	Z	j	
1011	VT	ESC	+	;	K	[	k	
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	≃
1111	SI	US	/	?	O	_	o	DEL

Tab.b : code ASCII

I-4 : L'arithmétique Binaire

Quand un calculateur effectue une opération arithmétique, deux contraintes interviennent :

- Les circuits travaillent sur des nombres qui ont une même longueur (format).
- Les circuits ne manipulent que deux nombres à la fois c.-à-d. : soit S, X, Y et Z des nombres binaires tel que : S = X+ Y+ Z, cette opération est réellement réalisé en deux opération comme suit : soit p une variable binaire :

$$P = X + Y \text{ puis calculer : } S = P + Y$$

De façon générale, les règles habituelles utilisées en décimal ne changent pas mais bien sûr il faut tenir compte de la base de travail, et de noter que l'opération fondamentale est l'addition, la soustraction n'étant rien que l'addition d'un nombre négatif.

I-4-1 Addition

L'addition binaire repose sur quatre règles de Tab-1 :

0 + 0 = 0	1 + 0 = 1
0 + 1 = 1	1 + 1 = 0 Retenue r =1

Tab-1 : Règles de l'Addition

Exemple : Soit à effectuer l'opération suivante : 11011 + 10011= ?

Résolution :

Débordement → 1

	$1\ 1^{1}0^{1}1\ 1$
+	1 0 0 1 1
	0 1 1 1 0

opération bits par bits (explication)

1+1 = 0 → r_0=1

1+1+ r_0= 1 → r_1 = 1

0 + 0 + r_1= 1 → r_2 =0

1+ 0 + r_2 = 1 → r_3 =0

1+1+r_3=0 → r_4=1

➢ 11011 + 10011= 01110

I-4-2 Soustraction

La soustraction binaire repose sur quatre règles de Tab-2 :

0 – 0 = 0	1 – 1= 0
1– 0 = 1	0 – 1 = 1 , Retenue r =1

Tab-2 : Règles de la soustraction

Exemple : Soit à effectuer l'opération suivante : 01110 – 10011 = ?

Débordemenet → 1

	$0\ 1^{1}1^{1}1\ 0$
-	1 0 0 1 1
	1 1 0 1 1

opération bits par bits (explication)

0 -1 = 1 → r_0=1

1 -1- r_0 = 1 → r_1 = 1

1- 0 - r_1= 0 → r_2 =0

1- 0 - r_2 = 1 → r_3 =0

0 -1- r_3 = 1 → r_4=1

➢ 01110 – 10011= 11011

I-4-3 la multiplication : la multiplication binaire se réalise de la même manière de la multiplication de deux nombre décimaux. Elle se pose sur les règles suivantes (Tab-3) :

0*0 = 0	1*0 = 0
0*1 = 0	1*1 = 1

Tab-3

Exemple : Soit à effectuer l'opération suivante : 1101 * 101= ?

```
   1101
 * 101
 ------
   1101
  0000 .
 1101 . .
 -------
 1000001
```

L'opération a réalisé est l'addition

- 1101*101 = 1000001

Remarque : la multiplication binaire se réduit à une série d'addition.

I-4-4 la division

La division binaire se réalise de la même manière de la division de deux nombre décimaux. Elle se pose sur les règles suivantes (Tab-4) :

0/1 = 0	**0/0 cas indéterminé**
1/1 = 1	**1/0 cas indéterminé**

Tab-4

Exemple : Effectuer 1001/11 = ?

```
 1001 | 11 *
-11   | 11
-0011
-  11
 0000
```

- 1001/11 = 11 Reste R= 0

Remarque : la division est une combinaison de multiplications et de soustraction.

NB : comme l'opération fondamentale est l'addition, la soustraction n'étant rien d'autre que l'addition d'un nombre négatif, il est donc important d'avoir une représentation des nombres négatifs qui permette d'effectuer simplement les soustractions.

I-5 Représentation des nombres signés

Des solutions ont été proposées, parmi lesquelles on cite :

- Représentation en signe + valeurs absolue

- Représentation en complément a 1 (complément restreint : CR)
- Représentation en complément a 2 (complément vrai : CV)

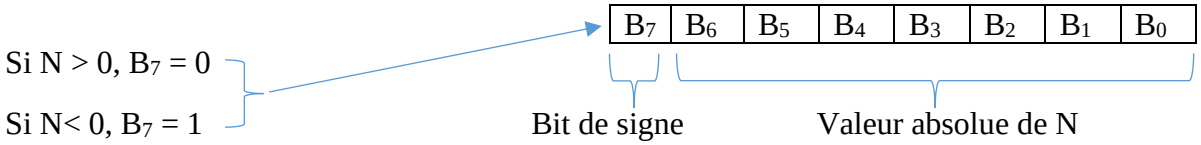
I-5-1 Représentation en signe + valeurs absolue

En représentation signe + valeur, un nombre est représenté sur 4, 8 ou 16 bits tel que :

- Le bit le plus à gauche désigne son signe : 0 → positif, 1 → négatif.

Les bits restants désignant la valeur absolue du nombre.

soit N un nombre quelconque à représenter sur 8 bits ;



Remarque : L'inconvénient de cette convention est double. Il y a deux représentations possibles pour le nombre 0. Par ailleurs la soustraction vue comme une addition bit a bit ne fonctionne pas.

Exemple illustratif : soit le tableau suivant, représentant des entiers de -7 a +7 ;

N	Signe+ val.absolue	N	Signe+ val.absolue	N	Signe+ val.absolue	N	Signe+ val.absolue
+7	0 111	+3	0 011	-0	1 000	-4	1 100
+6	0 110	+2	0 010	-1	1 001	-5	1 101
+5	0 101	+1	0 001	-2	1 010	-6	1 110
+4	0 100	+0	0 000	-3	1 011	-7	1 111

Réaliser les opérations suivante en binaire : - 3 + (+2) = ? Et (+ 3) + (-2)= ?

- 3 + (+2) -1 ; $1^{1}011$ + 0 010 ; ? 101 correspond à 5	3 + (-2) +1 ; $0^{1}011$ + 1 010 ; ? 101 correspond à 5
Résultat : - 1 ≠ 5	résultat : +1 ≠ 5

Conclusion : Non seulement l'addition avec la cette représentation ne fonctionne pas, il faut traiter le bit de signe séparément.

I-5-2 Représentation en complément a 1 (CR(N))

Il s'agit du complément par rapport au plus grand des nombres représentables. Ce nombre est composé de bits tous égaux à 1. En pratique, le complément à 1 d'un nombre N écrit sur n bits est obtenu en complémentant chacun des bits. On a alors ;

$$CR(N) + N = \underbrace{111\ldots111}_{n \text{ bits}} = 2^n - 1$$

En d'autres termes : On peut obtenir le CR(N) d'un nombre N en remplaçant les 0 par des 1 et les 1 par des 0.

Exemple : les nombres représentable avec n= 4 bits sont donnés dans le tableau ci-dessous.

N	CR(N)		N	CR(N)
+3	0 011		-3	1 100
+2	0 010		-2	1 101
+1	0 001		-1	1 110
+0	0 000		-0	1 111

Remarque :

1* La représentation des nombres positifs ne change pas, il s'agit toujours d'un bit de signe égale à 0 suivi de la valeur absolue, par contre, pour les nombres négatifs le bit de signe est toujours 1 mais la valeur absolue change (n'apparait plus)

2* Il y a toujours l'inconvénient de deux représentations pour le nombre 0

Exemple d'application de cette convention :

Réaliser les opérations suivante en binaire : - 3 + (+2) = ? Et (+ 3) + (-2)= ?

- 3 + (+2) = -1	CR (-3) + CR (+2) ?	En CR(N) 1100 + 0010 1110 → CR(1110)= -1		(+ 3) + (-2) = + 1	CR (+3) + CR (-2) ?	En CR(N) ^{1} ^{1}0 ^{1}0^{1}11 + 1 1 0 1 1 0 0 0 0 + → 1 = 0001 → CR(0001)= +1

Remarque :

1* pour les deux cas, en utilisant la représentation en complément à 1, les résultats trouvés sont corrects.

2* Le bit générer par le bit de signe (le plus à gauche) va être ajouté aux résultats finale (comme est montré dans le cas de : (+ 3) + (-2))

Conclusion : la représentation en complément à 1, permet d'effectuer des soustractions.

I-5-3 Représentation en complément a 2 (CV(N))

Les nombres positifs ont exactement la même représentation binaire que les deux représentations vues précédemment (signe +valeur / CR). Par contre les nombres négatifs sont obtenus comme suit : **CV(N) = CR(N) + 1**

Exemple : réalisons l'opération (-3) + (+2), en utilisant le complément à 2

CR (-3) = 1 100

CR (-3) +1= 1100 +1 → CV (-3) = 1101 1101

CV(2)= CR(2)= 0010 + 0010

Avec : (-3) + (+2) = -1 1 1 1 1 → CV(1111) = -1

Le résultat trouvé est correct : -1

La table suivante donne les trois représentations des entiers de -7 à +7 ;

N	Signe+val.absolue(N)	CR(N)	CV(N)
+7	0 111	0 111	0 111
+6	0 110	0 110	0 110
+5	0 101	0 101	0 101
+4	0 100	0 100	0 100
+3	0 011	0 011	0 011
+2	0 010	0 010	0 010
+1	0 001	0 001	0 001
+0	0 000	0 000	0 000
-0	1 000	1111	
-1	1 001	1110	1111
-2	1 010	1101	1110
-3	1 011	1100	1101
-4	1 100	1011	1100
-5	1 101	1010	1011
-6	1 110	1001	1010
-7	1 111	1000	1001

Exemple d'application : Trouver les opposés des nombres +2, -6, et – 7, en utilisant le complément à 2 (CV).

Nombre opposé = $\overline{CV}$ + 1, (sachant que : Si A =1, $\overline{A}$ =0)

Solution :

Nombre	***CV(N)***	$\overline{CV}$ (opposée en cv)	Nombre opposé
+2	0010	1101+1=1110	-2
-6	1010	0101+1= 0110	+6
-7	1001	0110+1=0111	+7