1. Systèmes de numération

11. Système décimal : Base 10

C'est le système utilisé dans la vie courante, il est basé sur le nombre 10.
Pour représenter les nombres décimaux, on utilise les chiffres de 0 à 9.

Exemple : **Ecriture d'un nombre décimal**

$(7348)_{10}=7.10^3+3.10^2+4.10^1+8.10^0$
10 représente la base du système décimal et les puissances de 0 à 3 constituent le rang de chaque chiffre.
10^3 est le poids du chiffre 7, 10^2 est le poids du chiffre 3,…, 10^0 est le poids du chiffre 8.
7 est le chiffre le plus significatif ou le chiffre du poids le plus fort, couramment appelé **MSB**.
8 est le chiffre le moins significatif ou le chiffre du poids le plus faible, couramment appelé **LSB**.

12. Système binaire : Base 2

C'est le système le plus utilisé en électronique numérique ou digitale.
Pour représenter les nombres binaires, on dispose uniquement de deux chiffres 0 et 1.
Un nombre binaire N s'écrit : $(N)_2=A_{n-1}\ \dots\ A_i\ \dots\ A_1A_0$ avec $A_i \in \{0,1\}$.
Chaque chiffre A_i est appelé couramment un **bit** et N est désigné par le mot binaire de n bits.
Ce nombre a pour valeur $N=A_{n-1}.2^{n-1}+\dots+A_i.2^i+\dots+A_1.2^1+A_0.2^0$ (forme polynomiale).
A_{n-1} est le chiffre le plus significatif ou le chiffre du poids le plus fort, couramment appelé **MSB**.
A_0 est le chiffre le moins significatif ou le chiffre du poids le plus faible, couramment appelé **LSB**.

Exemple : Soit le nombre $(N)_2=110101$, il a pour valeur décimale $(N)_{10}=1.2^5+1.2^4+0.2^3+1.2^2+0.2^1+1.2^0$.

13. Système hexadécimal : Base 16

Le système hexadécimal est apparu avec la logique microprogrammée et les microprocesseurs.
Pour représenter les nombres hexadécimaux, on utilise les chiffres de 0 à 9 et les lettres de A à F.
Par convention, les lettres **A**, **B**, **C**, **D**, **E** et **F** valent respectivement **10**, **11**, **12**, **13**, **14** et **15**.
La base 16 est une forme contractée de la base 2.

Exemple : Soit le nombre $(N)_{16}=B4E9$, il a pou valeur décimale $(N)_{10}=B.16^3+4.16^2+E.16^1+9.16^0$ soit alors :
$(N)_{10}=11.16^3+4.16^2+14.16^1+9.16^0$.

14. Correspondance entre les bases 10, 2 et 16

Le tableau récapitulatif ci-dessous donne l'équivalence de quelques nombres pour les bases 10, 2 et 16.

$(N)_{10}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$(N)_2$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
$(N)_{16}$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

2. Conversion entre les bases

21. Conversion des bases 2 et 16 à la base 10

On exploite directement la forme polynomiale des nombres à convertir, soit de la base 2 ou de la base 16.

211. Passage de la base 2 à la base 10

Exemple : $(100110)_2=1.2^5+0.2^4+0.2^3+1.2^2+1.2^1+0.2^0=(38)_{10}$

212. Passage de la base 16 à la base 10

Exemple : $(9A8E)=9.16^3+10.16^2+8.16^1+14.16^0=(39566)_{10}$.

Fonction Traiter

22. Conversion de la base 10 aux bases 2 et 16

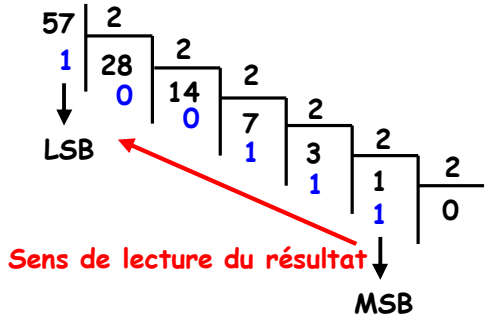
La méthode de division est la plus utilisée, elle consiste en des divisions successives du nombre $(N)_{10}$ par 2 ou par 16, jusqu'à obtenir un quotient nul. Les restes des divisions successives, écrits dans l'ordre inverse, constituent le nombre N dans la base 2 $(N)_2$ soit le nombre N dans la base 16 $(N)_{16}$.

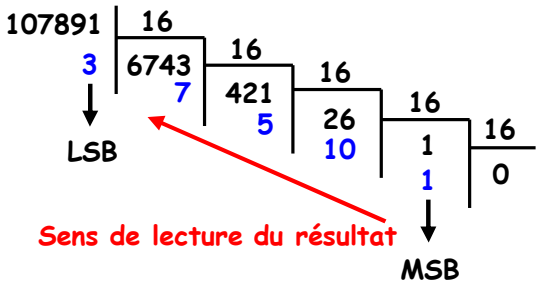
221. Passage de la base 10 à la base 2

Exemple : $(57)_{10}=(111001)_2$

222. Passage de la base 10 à la base 16

Exemple : $(107891)_{10}=(1A573)_{16}$





23. Conversion de la base 16 à la base 2 et inversement

231. Passage de la base 16 à la base 2

La conversion s'obtient en remplaçant directement chaque chiffre hexadécimal du nombre en sa représentation binaire. On peut utiliser le tableau de correspondance de la page 1.

Exemple : $(B9A)_{16}=(101110011010)_2$

232. Passage de la base 2 à la base 16

La conversion consiste tout simplement à regrouper les termes du nombre en base 2 par groupes de 4 en commençant par la droite puis convertir chaque groupe en écriture hexadécimale. On peut utilise le tableau de correspondance de la page 1.

Exemple : $(110101011100)_2=(D5C)_{16}$

2. Codage de l'information binaire

Un système électronique traite les informations en binaire. Or, ces informations à traiter sont de natures différentes (nombres, textes, images, sons, vidéos, programmes, …), alors on procède à une retranscription de ces informations dans un code compréhensible par les machines de traitement.
Cette procédure s'appelle "**Codage**" de l'information binaire. Elle permet d'établir une correspondance pour passer d'une représentation d'une information à une autre représentation sous forme binaire.
La technique du codage utilise plusieurs codes suivant le domaine d'application. L'opération inverse s'appelle "**Décodage**" ou "**Transcodage**". On étudie en particulier : Le code binaire pur, le code **GRAY**, le code **BCD** et le code **ASCII**.

21. Code binaire pur

Il est aussi appelé code binaire naturel. C'est le code binaire sans aucune codification, c'est-à-dire, qui découle directement du principe général de la numération. C'est le code naturel utilisé dans les systèmes numériques (ordinateur, API, …). Le tableau de la page 3 donne le code binaire pur pour un exemple d'un mot de 4 bits (A_3 A_2 A_1 A_0).

Fonction Traiter

Remarque :

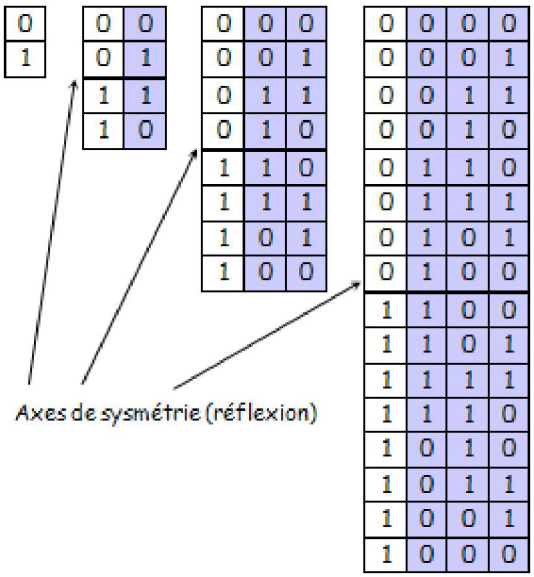
Lors du passage d'une combinaison à la suivante, deux bits ou plus peuvent changer simultanément d'état.
Exemple : Pour passer de la valeur 7 à la valeur 8, tous les bits changent simultanément d'état.

22. Code GRAY

Le code **GRAY** est un code binaire qui présente la particularité que seul un bit change d'état entre deux combinaisons successives. On l'appelle aussi "**code binaire réfléchi**" parce que pour le construire, on utilise le principe de réflexion par miroir plan comme l'indique le tableau suivant avec 4 bits ($G_3\,G_2\,G_1\,G_0$).
Exemple : Pour passer de la valeur 7 à la valeur 8, un seul bit a changé d'état.

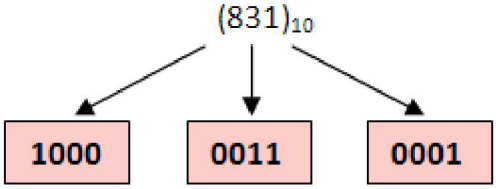
Ce code est principalement utilisé dans :
☑ Les codeurs numériques de position pour la mesure des déplacements linéaires ou angulaires.
☑ Simplification des équations logiques à l'aide des tableaux de **Karnaugh** (voir plus tard).

Valeur décimale	Code binaire				Code GRAY			
	A_3	A_2	A_1	A_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0



23. Code BCD

Le code **BCD** (**B**inary **C**oded **D**ecimal) ou encore appelé **DCB** (**D**écimal **C**odé en **B**inaire) est un code principalement utilisé dans la fonction affichage sur afficheurs 7 segments. Chaque chiffre décimal est codé en binaire sur quatre bits (Quartet) comme l'indique le tableau ci-contre.
Exemple : Soit à coder en BCD le nombre $(831)_{10}$.



On trouve bien alors : $(831)_{10}=(1000\ \ 0011\ \ 0001)_{BCD}$

Valeur décimale	Code BCD			
	A_3	A_2	A_1	A_0
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

Fonction Traiter

24. Code ASCII

Le code **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) est utilisé en informatique pour communiquer entre le clavier du micro-ordinateur et l'unité centrale. On distingue deux codes ASCII : le code ASCII standard et le code ASCII étendu. Le clavier est équipé d'un circuit spécial qui contrôle ses circuits en permanence. A chaque touche correspond un mot binaire. Le code ASCII standard possède **128** caractères, le code ASCII étendu en possède **256**. Pour coder l'ensemble des caractères il faut 7 bits pour le code ASCII standard et 8 bits pour le code ASCII étendu.
Le tableau ci-dessous donne le code ASCII standard.
Exemple : $A=(1000001)_2=(41)_{16}=(65)_{10}$

		$b_6b_5b_4$							
		000	001	010	011	100	101	110	111
$b_3b_2b_1b_0$	H	0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACQ	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[	k	{
1100	C	FF	FS	,	<	L	\	l	|
1101	D	CR	GS	-	=	M	]	m	}
1110	E	SO	RS	.	>	N	^	n	"
1111	F	SI	US	/	?	O	_	o	DEL

3. Opérations arithmétiques binaires

L'addition est l'opération arithmétique la plus importante dans les systèmes numériques. Les opérations de soustraction, de multiplication et de division effectuées par les systèmes numériques tels que l'ordinateur ne sont essentiellement que des variantes de l'opération d'addition.
L'addition binaire est une opération arithmétique qui donne la somme arithmétique de 2 nombres binaires. On se limite dans ce chapitre aux cas de l'addition et la soustraction binaires.

31. Addition binaire

311. Cas de deux nombres binaires de 1 bit

Pour une addition de deux nombres binaires $A=a_0$ et $B=b_0$ de 1 bit, 4 combinaisons sont possibles et le résultat occupera 2 bits : un bit pour la somme (**S**) et un autre pour la retenue ou report (**r**).
Les nombres A et B sont appelés les opérandes de l'opération d'addition.
Le tableau ci-contre résume le principe de l'addition de 2 nombres de 1 bit.

a_0	b_0	S	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Remarque : L'addition de trois nombres $A=a_0$, $B=b_0$ et $C=c_0$ de 1 bit donnera **S=1** et **r=1**.

312. Cas de général

La somme de deux nombres binaires, $A=a_{n-1}\ \dots\ a_1a_0$ et $B=b_{m-1}\dots b_1b_0$, s'effectue en faisant l'addition de chaque bit du 1er terme **A** avec chaque bit du 2ème terme **B** de poids identiques.

Exemple :

Soit à additionner les deux nombres : $A=(67)_{10}$ et $B=(43)_{10}$
On obtient ainsi le résultat suivant : $(1101110)_2=(110)_{10}$

Poids	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
Reports					1	1		
+	1	0	0	0	0	1	1	1er terme A
		1	0	1	0	1	1	2ème terme B
=	1	1	0	1	1	1	0	Résultat

⚠ **Remarque** : Pour décoder le résultat en décimal, on prend en considération la retenue si elle est générée.

32. Soustraction binaire

321. Complément à 1 d'un nombre binaire

Le complément à **1** d'un nombre binaire N quelconque s'obtient en inversant l'état logique de chaque bit.

Exemple : $(N)_2=11010100 \Rightarrow (N)_{C1}=00101011$

322. Complément à 2 d'un nombre binaire

Le complément à **2** d'un nombre binaire N quelconque est égal à son complément à 1 plus 1 :
$(N)_{C2}=(N)_{C1}+1$ et on a aussi $[(N)_{C2}]_{C2}=N$

⚠ **Remarque** : Une solution pratique consiste à retranscrire le nombre dont on cherche le complément à **2** en partant de la droite (poids le plus faible) sans aucun changement jusqu'au premier **1** rencontré, puis à inverser systématiquement les uns et les zéros rencontrés.

Exemple : $(N)_2=11010100 \Rightarrow (N)_{C1}=00101011 \Rightarrow (N)_{C2}=00101100$

323. Représentation des nombres signés

Pour écrire un nombre signé N dans la notation en complément à 2, on procède de la manière suivante :

☑ On ajoute un bit de signe devant le **MSB** du nombre N.
☑ Si **N>0** alors on affecte un **0** au bit du signe et on a : **N=0** $(N)_2$
☑ Si **N<0** alors on affecte un **1** au bit du signe et on a : **N=1** $(N)_{C2}$
☑ La notation en complément à 2 d'un nombre signé, transforme un nombre positif en un nombre négatif et vice versa.
☑ Dans la notation en complément à 2, les nombres à traiter doivent avoir le même nombre de bits.
☑ Le bit de signe, doit être traité sur le même pied d'égalité que les bits du nombre N.

Exemple 1 :

Le tableau ci-contre donne les nombres signés, positifs et négatifs, pour n=4 (3 bits pour le codage en binaire et 1 bit de signe).

$(N)_{10}$	$(N)_2$	$(N)_{C2}$	$(-N)_{10}$
0	0000	0000	0
1	0001	1111	-1
2	0010	1110	-2
3	0011	1101	-3
4	0100	1100	-4
5	0101	1011	-5
6	0110	1010	-6
7	0111	1001	-7

Exemple 2 :

$(N)_{10}$	Bit de signe	Poids					
		$2^5=32$	$2^4=16$	$2^3=8$	$2^2=4$	$2^1=2$	$2^0=1$
+45	0	1	0	1	1	0	1
-45	1	0	1	0	0	1	1

Fonction Traiter

324. Soustraction binaire

La soustraction binaire se ramène à une addition, en effet : $A-B=A+(B)_{C2}$.
Le résultat se lit directement en complément à 2 :

☑ Si le résultat est positif (bit de signe est égal à 0), alors on lit le résultat directement.
☑ Si le résultat est négatif (bit de signe est égal à 1), alors on convertit le résultat en recherchant son complément à 2.

🔔 **Remarques** :

☑ Pour prendre le complément à 2 du nombre B, on tient compte du bit de signe.
☑ Les deux nombres A et B doivent avoir le même nombre de bits.
☑ Toute retenue générée sera ignorée.

Exemples

☑ $(9)_{10}-(5)_{10}=(4)_{10}$
$(+9)_{10}=(01001)_2$
$(+5)_{10}=(00101)_2 \Rightarrow (-5)_{10}=(11011)_{C2}$

	1		1	1		
+	0	1	0	0	1	$(+9)_{10}$
	1	1	0	1	1	$(-5)_{10}$
1	0	0	1	0	0	$(+4)_{10}$

Retenue ignorée

☑ $(5)_{10}-(9)_{10}=(-4)_{10}$
$(+5)_{10}=(00101)_2$
$(+9)_{10}=(01001)_2 \Rightarrow (-9)_{10}=(10111)_{C2}$
Le bit de signe est égal à 1 donc il faut complémenter à 2 le résultat :
$(1100)_{C2}=0100=(+4)_{10}$ et puisque le bit de signe est égal à 1, le résultat réel est bien évidement $(-4)_{10}$.

		1	1	1	
+	0	0	1	0	1
	1	0	1	1	1
	1	1	1	0	0

☑ $(-9)_{10}-(5)_{10}=(-14)_{10}$
$(+5)_{10}=(00101)_2 \Rightarrow (-5)_{10}=(11011)_{C2}$
$(+9)_{10}=(01001)_2 \Rightarrow (-9)_{10}=(10111)_{C2}$
Le bit de signe est égal à 1 donc il faut complémenter à 2 le résultat :
$(0010)_{C2}=1110=(+14)_{10}$ et puisque le bit de signe est égal à 1, le résultat réel est bien évidement $(-14)_{10}$.

	1	1	1	1	1	
+		1	0	1	1	1
		1	1	0	1	1
	1	1	0	0	1	0

Retenue ignorée

☑ $(67)_{10}-(43)_{10}=(24)_{10}$
$(+67)_{10}=(01000011)_2$
$(+43)_{10}=(00101011)_2 \Rightarrow (-43)_{10}=(11010101)_{C2}$
Le résultat est bien évidement $(24)_{10}=(11000)_2$

	1	1				1	1	1	
+		0	1	0	0	0	0	1	1
		1	1	0	1	0	1	0	1
	1	0	0	0	1	1	0	0	0

Retenue ignorée

☑ $(43)_{10}-(67)_{10}=(-24)_{10}$
$(+43)_{10}=(00101011)_2$
$(+67)_{10}=(01000011)_2 \Rightarrow (-67)_{10}=(10111101)_{C2}$
Le bit de signe est égal à 1 donc il faut complémenter à 2 le résultat :
$(1101000)_{C2}=0011000=(+24)_{10}$ et puisque le bit de signe est égal à 1, le résultat réel est $(-24)_{10}$.

		1	1	1	1	1	1	
+	0	0	1	0	1	0	1	1
	1	0	1	1	1	1	0	1
	1	1	1	0	1	0	0	0

Fonction Traiter