I – Le Codage de L'information

I .1– Le Code de Gray :

1- Définition du Code de Gray :

Le **code de Gray**, également appelé **code Gray** ou **code binaire réfléchi**, est un type de codage binaire permettant de ne modifier qu'un seul bit à la fois quand un nombre est augmenté d'une unité. Cette propriété est importante pour plusieurs applications.

Le nom du code vient de l'ingénieur américain **Frank Gray** qui publia un brevet sur ce code en 1953, mais le code lui-même est plus ancien.

2- Principe de Code de Gray :

Le code de Gray est un codage binaire, c'est-à-dire une fonction qui associe à chaque nombre une représentation binaire. Cette méthode est différente du codage binaire naturel. Le tableau suivant montre partiellement le codage sur 4 bits (seules les 8 premières valeurs sont présentées, les huit suivantes avec le premier bit à 1 n'y sont pas).

Codage décimal	Codage binaire naturel	Codage Gray ou binaire réfléchi
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

La différence principale entre les deux est le fait que le codage de Gray de deux nombres consécutifs ne diffère que d'une position. Par exemple 5 est codé par 0111, et 6 est codé par 0101 : ici seul le deuxième bit change.

3- Conversion du code binaire vers le code de Gray

Soit B un nombre écrit en binaire naturel pur sur m bits

$B_{(2)}$= Bm ……$B_4B_3B_2B_1$; Bm est le bit du poids fort

G est l'équivalent en code de Gray du nombre B écrit lui aussi sur m bits

$G_{(Gray)}$= Gm ……$G_4G_3G_2G_1$

Le passage du binaire pur au code de Gray se fait selon les deux étapes suivantes :

1- Gm = Bm.
2- Maintenant on compare les paires :
   a. Si $B_n = B_{n-1}$ alors $G_{n-1}= 0$:
   b. Si $B_n \neq B_{n-1}$ alors $G_{n-1}= 1$:

Exemples :

Soit le nombre Binaire = 0101 (5 en décimal, on travaille ou on code sur 4 bits)

Quel est son équivalent en code de Gray ?

B = 0101 = $B_3\ B_2\ B_1\ B_0$

- **$G_3 = B_3 = 0$** ; (C'est la première étape).
- La deuxième étape est la comparaison des bits du nombre B.
  - $B_3 \neq B_2$ alors **$G_2 = 1$**.
  - $B_2 \neq B_1$ alors **$G_1 = 1$**.
  - $B_1 \neq B_0$ alors **$G_0 = 1$**.

On conclut que G = 0111.

*Prenons un autre exemple :

Soit le nombre Binaire = 1101101 (on travaille ou on code sur 7 bits)

Quel est son équivalent en code de Gray ?

B = 1101101 = $B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$

- **$G_6 = B_6 = 1$** ; (C'est la première étape).
- La deuxième étape est la comparaison des bits du nombre B.
  - $B_6 = B_5$ alors **$G_5 = 0$**.
  - $B_5 \neq B_4$ alors **$G_4 = 1$**.
  - $B_4 \neq B_3$ alors **$G_3 = 1$**.
  - $B_3 = B_2$ alors **$G_2 = 0$.**
  - $B_2 \neq B_1$ alors **$G_1 = 1$**.
  - $B_1 \neq B_0$ alors **$G_0 = 1$**.

On conclut que G = 1011011

4- Conversion du code de Gary vers le code Binaire pur :

Soit G un nombre écrit en code de Gray sur m bits

$G_{(Gray)}$= Gm ……$G_4G_3G_2G_1$ Gm est le bit du poids fort

$B_{(2)}$ est l'équivalent en code Binaire pur ou Naturel du nombre G écrit lui aussi sur m bits

$B_{(2)}$= Bm ……$B_4B_3B_2B_1$

Le passage du Code de Gray au code binaire pur se fait selon les deux étapes suivantes :

3- Bm = Gm.
4- Maintenant on compare les paires :
   a. Si $G_{n-1} = B_n$ alors $B_{n-1}= 0$:
   b. Si $G_{n-1} \neq B_n$ alors $B_{n-1}= 1$:

Exemples :

Soit le nombre G écrit en code de Gray = 110010, on veut trouver son équivalent en binaire pur.

- On applique la règle (on travaille sur 6 bits, car G est codé sur 6 bits)
  - G = 110010 = $G_5G_4G_3G_2G_1G_0$
  - 1- $B_5 = G_5 = 1$
  - 2-
    - $G_4 = B_5$ alors $B_4= 0$
    - $G_3= B_4$ alors $B_3=0$
    - $G_2= B_3$ alors $B_2=0$
    - $G_1\neq B_2$ alors $B_1 =1$.
    - $G_0\neq B_1$ alors $B_0 =1$.

  On peut alors déduire que B = 100011.

Prenons un autre exemple :

Soit G un nombre codé en code de Gray qui vaut 10111011, trouver sa valeur en code binaire pur.

On applique la règle, mais on va la schématiser pour plus de clarté.

	G_7	G_6	G_5	G_4	G_3	G_2	G_1	G_0
Gray	1	0	1	1	1	0	1	1
	↓	≠ ↙	= ↙	≠ ↙	= ↙	= ↙	≠ ↙	= ↙
Binaire pur	1	1	0	1	0	0	1	0
	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0

I.2 – Le Code DCB : Décimal Codé Binaire :

1- Définition du Code DCB :

Le code DCB ou BCD, est l'acronyme de Binary Coded Decimal en anglais est un système de numération utilisé en électronique numérique et en informatique pour coder des nombres en se rapprochant de la représentation humaine usuelle, en base 10. Dans ce format, les nombres sont

représentés par un ou plusieurs chiffres compris entre **0 et 9**, et chacun de ces chiffres est codé sur quatre bits.

2- Principe du Code DCB :

Pour coder un nombre décimal en BCD, on va coder séparément chaque chiffre du nombre de base dix en Binaire selon le tableau ci-dessous, on rappelle qu'on travaille sur **4 bits (4 positions),** ainsi les nombres de 10 à 15 (1010 à 1111) ne sont pas supportés par le code DCB.

décimal	DCB
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Codage des Chiffres Décimaux en DCB.

Exemple :
On va coder le nombre $(785)_{10}$ en DCB, pour cela, chaque chiffre du nombre sera codé séparément en binaire sur 4 bits.

Décimal	7	8	5
DCB	0111	1000	0101

Donc $(785)_{10}$ = (0111 1000 0101) $_{DCB}$

3- L'addition en code DCB :

Comme on l'a déjà mentionné, le code DCB permet le codage de 10 nombres uniquement, à savoir, les chiffres de 0 à 9 , de ce fait, les nombres de 10 à 15, ne sont pas autorisés, c'est-à-dire que le code DCB ne les connait pas.

Lorsqu'on additionne 2 nombres codés en DCB, on additionne chaque bloc de 4 bits du premier nombre, avec son équivalent du second nombre.

Maintenant, si le résultat d'un des blocs dépasse 9, c'est-à-dire sa valeur est comprise entre 10 et 15 (1010 à 1111), on rajoute à ce bloc le chiffre 6 (0110) .

Exemples :

- Soit à additionner $(352)_{10} + (34)_{10}$ en DCB

Decimal		DCB		
352		0011	0101	0010
+		+		
34			0011	0100
=		=		
386		0011	1000	0110

Là, le résultat est équivalent, et correcte.

- Soit à additionner $(153)_{10} + (151)_{10}$ en DCB

Décimal		DCB		
153		0001	0101	0011
+		+		
151		0001	0101	0001
=		=		
304		0010	1010	0010

Là, le résultat obtenue en DCB n'est pas équivalent à celui du Décimal, de plus, on a obtenu la valeur 1010 (10 en Décimal qui est >9) qui n'est pas reconnu par le code DCB, là, on doit rajouter à ce bloc qui dépasse 9, la valeur 6 (0110).

Voyons, ce qui se passe, lorsqu'on rajoute 6 (0110) à ce bloc.

Décimal		DCB		
153		0001	0101	0011
+		+		
151		0001	0101	0001
=		=		
304		0010	1010	0100
	+		0110	
	=	0011	0000	1000

Là, le résultat, est correcte !.

Une question subsiste, pourquoi rajouter 6 (0110) spécialement ?.

Eh bien, parce que comme en DCB, on code sur 4 bits, donc 2^4=16 nombres sont possibles à coderen binaire, mais seulement 10 nombres sont autorisés (de 0 à 9), les 6 nombres restants (de 10 à 15) sont interdits, lorsque la somme dépasse 9, on rajoute 6 pour faire une boucle et retourner à la valeur 0.

Remarques :

- Le nombre codé en BCD ne correspond pas au nombre décimal converti en binaire naturel.
- Les combinaisons supérieures à 9 (de 10 à 15) sont interdites. Par exemple la combinaison 1010 n'appartient pas au code BCD.
- Le codage décimal BCD est simple, mais il n'est pas possible de faire des opérations mathématiques directement dessus.
- Ce code est surtout utilisé pour l'affichage de données décimales. (Dans les calculatrices par exemple)

I.3 – Le Code DCB plus trois : DCB +3 :

1- Définition du Code DCB+3 :

Le code **DCB plus 3** ou **code excess 3** appelé aussi **code Stibiz** du nom de son inventeur, est un code non pondéré issu du code **DCB** auquel on ajoute systématiquement **3** à chaque chiffre. Ce code est souvent utilisé sur des unités arithmétiques qui calculent en système numérique décimal plutôt qu'en système binaire.

Le code plus 3 permet d'effectuer les opérations arithmétiques d'addition et de soustraction avec un minimum de fonctions logiques.

2- Principe du Code DCB+3 :

Pour coder un nombre décimal en BCD+3, avant de le coder en DCB, on lui rajoute 3, puis on le code en DCB, on rappelle qu'on travaille sur **4 bits (4 positions),** ainsi on a 16 nombres (de 0 à 15) qu'on peut coder en DCB +3, mais les combinaisons de 13 à 15 (1101 à 1111) sont interdites, on peut juste coder de 0 à 12.

Le tableau ci- dessous illustre le codage des nombres décimaux en DCB+3

Décimal	Décimal +3	DCB+3
0	3	0011
1	4	0100
2	5	0101
3	6	0100
4	7	0111
5	8	1000
6	9	1001
7	10	1010
8	11	1011
9	12	1100

Codage des Chiffres Décimaux par DCB+3

Exemple :

On va coder le nombre $(785)_{10}$ en DCB+3, pour cela, on rajoute à chaque chiffre 3, puis il sera codé séparément en binaire sur 4 bits, puis on l

Décimal	7	8	5
Décimal +3	10	11	8
DCB+3	1010	1011	1000

Donc $(785)_{10}$ = (1010 1011 1000) $_{DCB+3}$.

II- REPRESENTATION DES NOMBRES ENTIERS

II.1 Représentation d'un entier naturel

Un entier naturel est un nombre entier positif ou nul. Le choix à faire (c'est-à-dire le nombre de bits à utiliser) dépend de l'intervalle des nombres que l'on désire utiliser. Pour coder des nombres entiers naturels compris entre 0 et 255, il nous suffira de 8 bits (un octet) car 2^8=256. D'une manière générale un codage sur n bits pourra permettre de représenter des nombres entiers naturels compris entre 0 et 2^{n-1}.

Exemples : 9 = $(00000101)_2$, 128 = $(10000000)_2$

II.2 Représentation d'un entier relatif

Un entier relatif est un entier pouvant être négatif. Il faut donc coder le nombre de telle façon que l'on puisse savoir s'il s'agit d'un nombre positif ou d'un nombre négatif.

- **Problème :** Comment indiquer à la machine qu'un nombre est négatif ou positif ????

Il existe 3 méthodes pour représenter les nombres négatifs :

II.2.1 Signe et valeur absolue (S/VA)

Si on travaille sur n bits , alors le bit du poids fort est utilisé pour indiquer le signe (1 : signe négatif, 0 : signe positif) et les autres bits (n -1) désignent la valeur absolue du nombre.

- **Exemple** : Si on travaille sur 4bits

  $(-5)_{10}$= $(1\ 101)_2$; $(+5)_{10}$ = $(0101)_2$

Signe	VA	Valeurs
0	**00**	**+ 0**
0	01	+1
0	10	+2
0	11	+3
1	**00**	**-0**
1	01	-1
1	10	-2
1	11	-3

Sur n bits, l'intervalle de valeurs qu'on peut représenter en S/VA :

$$-(2^{(n-1)} - 1) \leq N \leq +(2^{(n-1)}-1)$$

Essayons de calculer 2 + (-2) = 0

	Binaire					Décimal	
	0	0	1	0		2	
+	1	0	1	0	+	−2	
=	1	1	0	0	=	−4	

Ce n'est pas ce que nous voulions :-4 au lieu de 0 !!

Avantages : C'est une représentation assez simple.

Inconvénient : le zéro possède deux représentations +0 et -0

II.2.2 Le complément à un

On appelle complément à un (c.à.1) d'un nombre N un autre nombre N' tel que : $N+N'=2^n-1$

n : est le nombre de bits de la représentation du nombre N .

Exemple :

Soit N=1010 sur 4 bits donc son complément à un de N : $N'= (2^4 - 1)-N$

$N'=(16-1)-(1010)_2= (15)_{10} - (1010)_2 = (1111)_2 – (1010)_2 = 0101$

$$\begin{array}{r} 1\ 0\ 1\ 0 \\ +\ 0\ 1\ 0\ 1 \\ \hline 1\ 1\ 1\ 1 \end{array}$$

Pour trouver le complément à un d'un nombre, il suffit d'inverser tous les bits de ce nombre : Si le bit est un 0 mettre à sa place un 1 et si c'est un 1 mettre à sa place un 0 .

Exemples

$(10010011) = (01101100)_{c.à.1}$

$(11001100) = (00110011)_{c.à.1}$

En Complément à un, le bit du poids fort nous indique le signe (0 : positif , 1 : négatif).

Le complément à un du complément à un d'un nombre est égale au nombre lui même.

CA1(CA1(N))= N

Exemple :

Quelle est la valeur décimale représentée par la valeur 101010 en complément à 1 sur 6 bits ?

- Le bit poids fort indique qu'il s'agit d'un nombre négatif.
- Valeur = - CA1(101010)

$= - (010101)_2 = - (21)_{10}$

Si on travaille sur 3 bits

CA1	Binaire	Décimal
000	000	**+ 0**
001	001	+ 1
010	010	+ 2
011	011	+ 3
100	- 011	- 3
101	- 010	- 2
110	- 001	- 1
111	- 000	**- 0**

Inconvénient : Dans cette représentation le zéro possède une double représentation.

Si on travaille sur n bits, l'intervalle des valeurs qu'on peut représenter en CA1 :

$$-(2^{(n-1)} - 1) \leq N \leq +(2^{(n-1)} - 1)$$

II.2.3 Le complément à deux

Les nombres positifs sont codés de la même manière qu'en binaire pure alors qu'un nombre négatif est codé en ajoutant la valeur 1 à son complément à 1. Le bit le plus significatif est utilisé pour représenter le signe du nombre.

La représentation en complément à deux est la représentation la plus utilisée pour la représentation des nombres négatifs dans la machine.

Le complément à deux du complément à deux d'un nombre est égal au nombre lui-même.

CA2(CA2(N))= N

CA2	binaire	valeur
000 001 010 011	000 001 010 011	**+ 0** + 1 + 2 + 3
100 101 110 111	- 100 - 011 - 010 - 001	- 4 - 3 - 2 - 1

Avantage : On remarque que le zéro n'a pas une double représentation

Si on travaille sur n bits, l'intervalle des valeurs qu'on peut représenter en CA2 :

$$-(2^{(n-1)}) \leq N \leq +(2^{(n-1)} - 1)$$

Exemple

On désire coder la valeur −19 sur 8 bits. Il suffit :

1. d'écrire 19 en binaire : 00010011

2. d'écrire son complément à 1 : 11101100

3. et d'ajouter 1 : 11101101

La représentation binaire de −19 sur 8 bits est donc 11101101.

- **Opérations arithmétique en complément à deux**

$$\begin{array}{ll} +9 & 01001 \\ \underline{+4} & \underline{00100} \\ +13 & \mathbf{0}1101 \end{array}$$

(le bit de signe = 0 ➔ le nombre est positif)

$(01101)_2 = (13)_{10}$

$$\begin{array}{ll} +9 & 01001 \\ \underline{-4} & \underline{11100} \\ +5 & 100101 \end{array}$$

On remarque que le résultat est sur 6 alors qu'on travaille sur 5 bits dans ce cas on ignore le sixième bit (appelé report) de ce fait le résultat de l'addition sera 00101(Résultat positif)

$(00101)_2 = (5)_{10}$

$$\begin{array}{ll} -9 & 10111 \\ \underline{-4} & \underline{11100} \\ -13 & 110011 \end{array}$$

On ignore le report le résultat est 110011 (nombre négatif)

Résultat = - CA2 (10011)= -(01101)

= - 13

$$\begin{array}{ll} -9 & 10111 \\ \underline{+9} & \underline{01001} \\ 0 & 100000 \end{array}$$

Le résultat est positif

$(00000)_2 = (0)_{10}$

➢ La retenue et le débordement

- ✓ On dit qu'il y a une retenue si une opération arithmétique génère un report.
- ✓ On dit qu'il y a un débordement (Over Flow) si le résultatde l'opération sur n bits **et** faux.
- ✓ Le nombre de bits utilisés est insuffisant pour contenir le résultat, autrement dit le résultat dépasse l'intervalle des valeurs sur les n bits utilisés.

➢ Cas de débordement

$$\begin{array}{rr} +9 & 01001 \\ \underline{+8} & \underline{01000} \\ +17 & 10001 \end{array}$$

Le résultat est négatif alors qu'il doit être positif ➔ Débordement !!!

$$\begin{array}{rr} -9 & 10111 \\ \underline{-8} & \underline{11000} \\ -17 & 01011 \end{array}$$

Le résultat est positif alors qu'il doit être négatif ➔ Débordement !!!

Nous avons un débordement si la somme de deux nombres positifs donne un nombre négatif, ou la somme de deux nombres négatifs donne un Nombre positif

Il n'y a jamais un débordement si les deux nombres sont de signes différents.

III. LA REPRESENTATION DES NOMBRES REELS

Un nombre réel est constitué de deux parties : la partie entière et la partie fractionnelle (les deux parties sont séparées par une virgule)

Problème : comment indiquer à la machine la position de la virgule ?

Il existe deux méthodes pour représenter les nombre réel :

III.1 Nombres à virgule fixe

Possède une partie 'entière' et une partie 'décimale' séparés par une virgule, utilisé parles premières machines. La position de la virgule est fixe d'où le nom.

Exemple : $(11.01)_2$, $(75.23)_8$, $(E7,A4)_{16}$

III.2 Nombres à virgule flottante

Les nombres à virgule flottante (nombres flottants) (floating-point numbers), sont en général des nombres non entiers dont on écrit les chiffres uniquement après la virgule et auquel on ajoute un exposant pour préciser de combien de positions la virgule doit être déplacées.

Par exemple en notation décimale, on écrira :

le nombre 31, 41592 sous la forme : $0.3141592 * 10^2$

le nombre -0,01732 sous la forme : $0.1732 *10^{-1}$

En informatique, une norme s'est imposée pour la représentation des nombres flottants. C'est la norme IEEE 754.

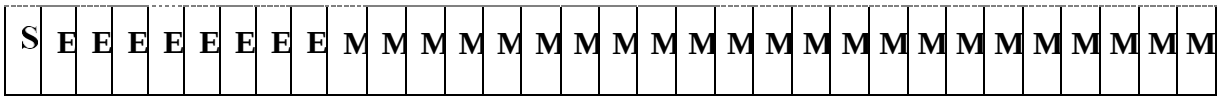
➢ La norme IEEE 754

L'IEEE 754 est un standard pour la représentation des nombres à virgule flottante en binaire. Il est le plus employé actuellement pour le calcul des nombres à virgule flottante dans le domaine informatique,

Dans la norme IEEE 754, un nombre flottant est toujours représenté par un triplet (S,E,M)

1. La première composante S détermine le signe du nombre représenté, ce signe valant 0 pour un nombre positif, et 1 pour un nombre négatif , le signe est représenté par un seul bit, le bit de poids fort (Celui le plus à gauche)
2. la deuxième E désigne l'exposant est codé sur 8 bits consécutifs au signe
3. la troisième M désigne la mantisse (les bits situés après la virgule) sur les 23 bits restants

Ainsi le codage se fait sous la forme suivante :



IEEE 754 Simple précision :

Signe (1 bit)	Exposant (8 bits)	Mantisse (23 bits)

IEEE 754 Double précision :

Signe (1 bit)	Exposant (11 bits)	Mantisse (52 bits)

Certaines conditions sont toutefois à respecter pour les exposants :

- ✓ L'exposant 00000000 est interdit
- ✓ L'exposant 11111111 est interdit. On s'en sert toutefois pour signaler les erreurs, on appelle alors cette configuration NaN (Not a Number)
- ✓ Il faut ajouter 127 à l'exposant (cas de la simple précision) pour une conversion de décimal vers un nombre réel binaire. Les exposants peuvent ainsi aller de -256 à 255.

La formule d'expression des nombres réels est ainsi la suivante :

$$(-1)^s \cdot 2^{(E-127)} \cdot 1,M$$

Remarque Le 127 du (E-127) vient de $2^{\text{nbre bit de l'exposant -1}} - 1$

Exemple 1 : Trouver la représentation IEEE 754 simple précision du nombre

$(35.5)_{10}$ Le nombre est positif ➔ S=0

$(35.5)_{10} = (100011.1)_2$ ……. Virgule fixe

$= 1.000111 * 2^{5}$ ·····Virgule flottante (M= 000111)

Exposant : E-127 = 5 ➔ E= 132 = $(10000100)_2$

Donc

0	10000100	00011100000000000000000
S	E	M

Exemple 2: Trouver la représentation IEEE 754 simple précision du nombre (-525.5)$_{10}$ Le nombre est négatif ➔ S=1

$(525.5)_{10} = (1000001101.1)_2$ ..... Virgule fixe

$= 1.0000011011 * 2^{9}$ ····· Virgule flottante (M= 0000011011)

Exposant : E-127 = 9 ➔ E= 136 = $(10001000)_2$

Donc

1	10001000	00000110110000000000000
S	E	M

Exemple 3 : Trouver la représentation IEEE 754 simple précision du nombre

$(-0.625)_{10}$ Le nombre est négatif ➔ S=1

$(0.625)_{10} = (0.101)_2$ ……. Virgule fixe

$= 1.01 * 2^{-1}$ ·····Virgule flottante (M= 01)

Exposant : E-127 = -1 ➔ E= 126 = $(1111110)_2$

Donc

1	01111110	01000000000000000000000
S	E	M

Exemple 4 : trouver le nombre flottant ayant la représentation IEEE754 suivante :

0	10000001	11100000000000000000000

S =0 ➔ Le nombre est positif

E = $(10000001)_2 = 129$ ➔ E-127 = 129 -127 =2

1.M = 1.111

➔ $1.111 * 2^2 = (111,1)_2 = (7.5)_{10}$