II- LES DIFFERENTS CODES BINAIRES

Les informations traitées par l'ordinateur sont de différents types mais elles sont toujours représentées à la base sous forme binaire.

Le codage d'une information consiste à établir une correspondance entre la représentation externe de l'information (le caractère A ou le nombre 35 par exemple) et sa représentation interne qui est une combinaison de 0 et de 1 représentés physiquement par les deux états stables des circuits utilisés dans les ordinateurs.

Les codes sont donc des codes binaires. Et on distingue les codes pondérés et les codes non pondérés.

1) Les codes pondérés

Dans un code pondéré, chaque chiffre du nombre codé possède un poids particulier, par exemple 1, 10, 100, 1000,... en numération décimale, ou 1, 2, 4, 8,...en numération binaire, de telle sorte que la somme des poids affectés, selon le rang des éléments binaires, donne le nombre codifié.

Le nombre décimal est obtenu en faisant la somme des bits affectés de leurs poids.

Un code pondéré permet une conversion décimale facile, ce qui constitue un critère de choix parmi les codes possibles.

Les codes pondérés les plus courants sont les suivants :

a) le code binaire naturel et ses dérivés (octal, hexadécimal)

Ce sont ceux que l'on utilise en arithmétique binaire. Les principaux inconvénients des codes simples sont :

- La longueur du nombre (ou le nombre de chiffres) augmente avec le nombre.
- Plusieurs variables changent d'état entre deux combinaisons successives.
- L'inadaptation à la détection ou à la correction automatique des erreurs.

b) le code DCB 8421 (Décimal Codé Binaire)

C'est le code binaire « naturel » des chiffres décimaux (Binary Coded Decimal).

Dans ce code, chaque chiffre décimal de 0 à 9 est représenté par 4 bits du binaire pur, avec dans l'ordre les poids 8, 4, 2,1, c'est à dire $8 = 2^3$ pour le bit n°3, $4 = 2^2$ pour le bit n°2, $2 = 2^1$ pour le bit n°1, $1 = 2^0$ pour le bit n°0.

Exemple :

$1011_{DCB} \Rightarrow 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 = 11_{10}$

$32_{10} \Rightarrow 00110010_{DCB}$

On doit tenir compte, lors des opérations arithmétiques, des 6 configurations inutilisées.

Pour l'addition, il faut ajouter 6 chaque fois que le résultat est supérieur à 9.

Exemple :

décimal		binaire	BCD	
15		01111	0001 0101	
+ 18	+	10010	+ 0001 1000	
= 33		100001	0010 1101	> 9
		(33)	+ 0110	
			00110011	(33)

Pour une soustraction, il faut retrancher 6 chaque fois que le résultat est négatif.

Les opérations arithmétiques sont donc assez compliquées. Par contre, les opérations d'Entrées/Sorties sont faciles : chaque entité DCB est directement associée à un caractère.

L'intérêt de ce code consiste en ce que la conversion peut se faire par calcul (et donc par programme).

c) le code DCB 2421 ou AIKEN

Ce code ne permet que la représentation des chiffres décimaux.

Dans ce code, les poids affectés aux variables binaires sont 2, 4, 2,1 (voir tableau ci dessous). C'est un code auto complémentaire, cela signifie que les bits de deux combinaisons représentatives de deux chiffres décimaux dont la somme fait 9 sont respectivement inverses l'un de l'autre. Par exemple si la combinaison relative à 3 est 0011, celle relative à 6 est 1100.

On remarque que le code AIKEN correspond au code binaire naturel pondéré 8 4 2 1 auquel on a supprimé les 6 combinaisons médianes : il est donc symétrique.

Comptage dans les codes pondérés DCB 8421 et 2421 :

N Décimal	DCB 8 4 2 1	DCB 2 4 2 1
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 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 1 0 0
7	0 1 1 1	1 1 0 1
8	1 0 0 0	1 1 1 0
9	1 0 0 1	1 1 1 1

Les états interdits sont ceux allant de 10 à 15.

2) Les codes non pondérés

Ce sont des codes dans lesquels les chiffres ne possèdent pas de poids particuliers. Parmi les codes non pondérés les plus utilisés, on peut citer :

a) le code excédent 3

On obtient ce code en décalant vers le haut de 3 lignes le code DCB 8421. C'est un code auto complémentaire donc symétrique. Ce code est très adapté aux opérations de soustraction car il permet d'obtenir facilement le complément à 9 de chaque chiffre en inversant chaque bit. Ainsi, la soustraction se ramène à une addition.

Exemple :

$7 - 4 = (7 + 3) - (4 + 3) = 1010 - 0111$
$= 1010 + (1111 - 0111) - 1111$
$= 1010 + 1000 + 0001 - 10000$
$= 10011 - 10000$
$= 0011 = 3$

On ajoute à 7 le complément à 9 de 4, c'est à dire 5, puis on ajoute encore 1, le tout en excédent 3 et on rejette le $5^{ème}$ bit : on obtient le résultat 3 en code binaire naturel.

b) un code cyclique : le code binaire réfléchi ou code GRAY

Ce code ne permet que la représentation des nombres décimaux. Il est caractérisé par le fait qu'en passant d'une combinaison à la suivante, un seul bit change de valeur. En effet, à chaque augmentation d'une unité du chiffre décimal un seul chiffre du nombre binaire équivalent change de valeur par rapport au nombre binaire précédent. Ainsi deux combinaisons quelconques successives sont **adjacentes**.

Dans le code binaire naturel, la variable de poids 2^0 prend successivement les valeurs 0, 1, 0, 1, 0, 1,... Dans le code binaire réfléchi ou code GRAY, il en est autrement :

- La variable de poids 2^0 conserve la valeur 2 fois (2^1).
- La variable de poids 2^1 conserve la valeur 4 fois (2^2).
- La variable de poids 2^2 conserve la valeur 8 fois (2^3), etc...

Il faut remarquer que dans ce code, chaque bit a une pondération de $\pm (2^n - 1)$ où n est le rang du bit dans le nombre. Le premier chiffre 1, rencontré en partant de la gauche est affecté du signe +, le deuxième chiffre 1 est affecté du signe -, le troisième du signe +, etc,...

Exemple :

Soit le nombre 1011 en binaire réfléchi :

▪ chiffres	1	0	1	1
▪ rangs	4	3	2	1
▪ poids	$(2^4 - 1)$		$-(2^2 - 1)$	$(2^1 - 1)$
Soit	+ 15		- 3	+ 1

D'où l'équivalent décimal de ce nombre binaire :

$$15 - 3 + 1 = 13$$

- **Conversion binaire naturel -binaire réfléchi** :

On obtient la représentation binaire réfléchi d'un nombre binaire naturel en l'additionnant (sans retenue) à lui même décalé d'un rang vers la gauche et en abandonnant le chiffre de plus petit poids.

Exemple :

Soit : $N = 7_{10} = 111_2$

$+ \ 111_2$

$= \overline{100}(1)$ en code réfléchi

- **Passage du code réfléchi au code binaire naturel :**

On part de gauche à droite et on transcrit chaque chiffre tel quel ou on le remplace par son complément, selon que le chiffre précédent obtenu pour l'équivalent binaire naturel est un 0 ou un 1.

Soit un nombre $B = b_n b_{n-1} \ldots b_0$ écrit en code GRAY, déduire le nombre en binaire naturel qui aura la forme $A = a_n a_{n-1} \ldots a_0$

On aura :

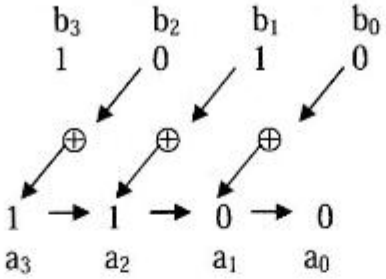
$$a_n = b_n$$

$$a_{n-i} = a_{n-i+1} \oplus b_{n-i} \qquad i \geq 1$$

Exemple :

Soit N = 1010 $_{GRAY}$, trouver son équivalent binaire :

On a:



Comptage dans les codes non pondérés Excédent 3 et GRAY :

N Décimal	Excèdent 3	GRAY
0	0 0 1 1	0 0 0 0
1	0 1 0 0	0 0 0 1
2	0 1 0 1	0 0 1 1
3	0 1 1 0	0 0 1 0
4	0 1 1 1	0 1 1 0
5	1 0 0 0	0 1 1 1
6	1 0 0 1	0 1 0 1
7	1 0 1 0	0 1 0 0
8	1 0 1 1	1 1 0 0
9	1 1 0 0	1 1 0 1

III- LES CODES REDONDANTS

Lorsque l'on redoute une modification intempestive de l'information lors de sa transmission ou de son stockage, on est conduit à utiliser des codes permettant soit de détecter les erreurs (codes auto vérificateurs), soit de les corriger (codes auto correcteurs).

Ceci implique que les codes portent sur un nombre de digits binaires supérieur à celui strictement nécessaire pour coder l'information. Par exemple, l'addition d'un digit supplémentaire

dit "digit de parité" constitue une méthode simple : on rajoute, éventuellement, un 5ème digit de telle façon que le nombre de 1 soit pair (ou impair). Il sera alors facile de détecter toute perte d'information dans la transmission.

Par ailleurs, en augmentant la redondance, il est possible d'arriver à savoir non seulement s'il y a une erreur mais également à quel endroit elle s'est produite. Dans ce cas, on peut alors concevoir une autocorrection du système d'acquisition qui, sachant où est l'erreur, la corrige immédiatement.

Parmi les codes détecteurs d'erreurs les plus connus, on cite le code 2 sur 5 et le code biquinaire.

Le code de HAMMING constitue un code correcteur d'erreur simple et détecteur d'erreur double.

1) Le code 2 sur 5 (2 parmi 5) :

Chaque chiffre décimal est codé sur 5 bits dans lesquels le nombre de 1 est constant et égal à deux. Il est pondéré 74210 (le zéro a été modifié pour que son écriture comporte deux " 1").

N décimal	2 parmi 5 7 4 2 1 0
0	1 1 0 0 0
1	0 0 0 1 1
2	0 0 1 0 1
3	0 0 1 1 0
4	0 1 0 0 1
5	0 1 0 1 0
6	0 1 1 0 0
7	1 0 0 0 1
8	1 0 0 1 0
9	1 0 1 0 0

2) Le code biquinaire

Ce code est pondéré 5043210. Un chiffre décimal est codé en un nombre binaire, sur 7 bits, séparé en deux groupes, ce qui facilite la localisation des erreurs. Un bit (et un seul) est toujours à 1 dans les deux positions de gauche et un bit (et un seul) est à 1 dans les cinq positions de droite.

N	Code biquinaire (2 parmi 7)						
Décimal	**Premier groupe**		**Deuxième groupe**				
	5	**0**	**4**	**3**	**2**	**1**	**0**
0	0	1	0	0	0	0	1
1	0	1	0	0	0	1	0
2	0	1	0	0	1	0	0
3	0	1	0	1	0	0	0
4	0	1	1	0	0	0	0
5	1	0	0	0	0	0	1
6	1	0	0	0	0	1	0
7	1	0	0	0	1	0	0
8	1	0	0	1	0	0	0
9	1	0	1	0	0	0	0

3) Le code de HAMMING

La notion de bit de parité peut se généraliser. On pourrait former plusieurs bits de contrôle à l'aide de relations linéaires, et augmenter ainsi les performances du code.

Considérons le code suivant dit code de HAMMING (7,4) :

N	a_1	a_2	A_3	a_4	a_5	A_6	a_7
0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1
2	0	1	0	1	0	1	0
3	1	0	0	0	0	1	1
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1
6	1	1	0	0	1	1	0
7	0	0	0	1	1	1	1
8	1	1	1	0	0	0	0
9	0	0	1	1	0	0	1
10	1	0	1	1	0	1	0
11	0	1	1	0	0	1	1
12	0	1	1	1	1	0	0
13	1	0	1	0	1	0	1
14	0	1	0	1	1	0	0
15	1	1	1	1	1	1	1

Nous pouvons constater que :

1) Les bits a_3, a_5, a_6 et a_7 reproduisent tout simplement l'équivalent binaire du chiffre.

2) Les bits a_1, a_2 et a_4 sont des combinaisons linéaires des 4 précédents, donnés par :

$$a_1 = a_3 \oplus a_5 \oplus a_7$$

$$a_2 = a_3 \oplus a_6 \oplus a_7$$

$$a_4 = a_5 \oplus a_6 \oplus a_7$$

Si le message est transmis correctement, ces mêmes relations devront être vérifiées. On doit avoir notamment :

$$a_1 \oplus a_3 \oplus a_5 \oplus a_7 = 0$$

$$a_2 \oplus a_3 \oplus a_6 \oplus a_7 = 0$$

$$a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 0$$

Ce qui pourra s'exprimer sous forme matricielle :

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} . \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}$$

$$H \cdot a = e$$

Avec $e_1 = e_2 = e_3 = 0$ dans le cas d'une transmission correcte.

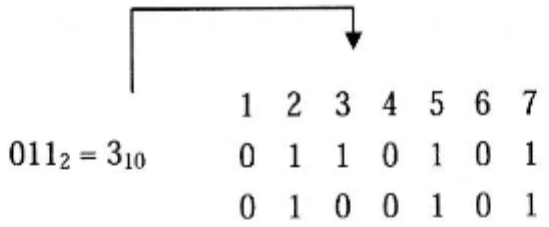
Nous appellerons la matrice H le **syndrome**, la matrice colonne au **mot reçu** et la matrice colonne e **l'erreur**.

Remarquons que les colonnes du syndrome sont constituées des chiffres 1 à 7 en binaire avec le bit de poids faible en haut. Ainsi la matrice colonne erreur permet de repérer le bit erroné, que l'on pourra rétablir par complémentation.

Exemple :

Soit le mot à transmettre N = 0100101. A cause d'une mauvaise transmission, le mot reçu est N' = 0110101. Le contrôle donne

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} . \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$



IV- LES CODES ALPHANUMERIQUES

Un code alphanumérique traduit sous forme binaire les chiffres de 0 à 9, mais aussi les caractères de l'alphabet (parfois majuscules et minuscules), les signes de ponctuation, quelques caractères spéciaux et quelques instructions de service (concernant le fonctionnement du téléimprimeur et la transmission des messages).

A titre d'exemple, on cite comme code alphanumérique le code USASCII (USA Standard Code for Information Interchange).

Chaque symbole y est codé sur 8 bits (dont un bit de parité) précédés d'un signal de début (START) et d'un signal de fin (STOP).

Ce code est utilisé par les réseaux informatiques assurant les connexions entre des ordinateurs et des organes périphériques (claviers, visualisations, imprimantes, etc...).

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

Voici la légende des instructions des services :

ACK	Acknowledge	**BEL**	Bell or alarm
BS	Backspace	**CAN**	Cancel
CR	Carriage return	**DC1**	Device control 1
DC2	Device control 2	**DC3**	Device control 3
DC4	Device control 4	**DEL**	Delete
DLE	Data link espace	**EM**	End of medium
ENQ	Enquiry	**EOT**	End of transmission
ESC	Escape	**ETB**	End of transmission block
ETX	End of text	**FF**	Form feed
FS	File separator	**GS**	Group separator
HT	Horizontal tabulation	**LF**	Line feed
NAK	Negative acknowledge	**NUL**	Null, or all zero
RS	Record separator	**SI**	Shift in
SO	Shift on	**SOH**	Start of heading
SP	Space	**STX**	Start of text
SUB	Substitute	**SYN**	Synchronous idle
US	Unit separator	**VT**	Vertical tabulation