LES CODES BINAIRES

Comme son nom l'indique un code est une convention entre utilisateurs qui leur permet de se communiquer des informations. Il peut donc exister une infinité de codes binaires. Nous étudierons ici les codes les plus courants sans prétendre être exhaustif.

Le code binaire naturel

Le code binaire naturel (BN) est la succession naturelle des nombres binaires. Or cette succession est illimitée il faudra donc préciser la taille des nombres afin d'être précis.

Code binaire naturel 4 bits

CODE BINAIRE	EQUIVALENT DECIMAL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Le code binaire DCBN

Ce code binaire a déjà été étudié au premier chapitre nous ne le rappelons ici que pour mémoire

Code DCBN

CODE DCBN	EQUIVALENT DECIMAL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Le code binaire réfléchi ou code de Gray

Le code binaire réfléchi (BR) ou code de Gray est un code utilisé pour ses propriétés de symétrie dans le codage de position ou pour la simplification des équations logiques.
Ce code ne peut être utilisé pour des opérations arithmétiques

Mécanisme de construction du code binaire réfléchi.

Comme le code binaire naturel il peut se faire à N bits faisons le code BR à 4 bits

Les deux premiers termes s'écrivent comme en binaire naturel

0 0 0 0
0 0 0 1

les deux termes suivants seront réalisés en faisant une symétrie par rapport à un axe de la première colonne, colonne de droite, de 01 elle devient 10, puis dans la deuxième colonne nous mettrons des 1

Les 4 premiers termes s'écriront:

0 0 0 0
0 0 0 1
------- Axe de symétrie pour la première colonne
0 0 1 1
0 0 1 0

pour écrire les 4 termes suivants nous ferons une symétrie des 2 premières colonnes dont les termes sont actuellement 00, 01, 11, 10 et deviennent 10, 11, 01, 00 puis nous mettrons des 1 dans la troisième colonne

Les 8 premiers termes s'écrieront donc:

0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
---------- Axe de symétrie des deux premières colonnes
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0

et ainsi de suite
D'où le code binaire réfléchi 4 bits:

0 0 0 0
0 0 0 1 Axe de symétrie de la première colonne
0 0 1 1
0 0 1 0 Axe de symétrie des deux premières colonnes
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0 Axe de symétrie des trois premières colonnes
1 1 0 0
1 1 0 1
1 1 1 1
1 1 1 0
1 0 1 0
1 0 1 1
1 0 0 1
1 0 0 0

III - 3 - 2 - Propriétés du code binaire réfléchi.

Le code binaire réfléchi est employé essentiellement pour ses propriétés de symétrie et d'adjacence. On dit que deux nombres ou combinaisons binaires sont adjacentes lorsqu'elles ne diffèrent que d'un seul bit par exemple, 1100 et 1101 sont adjacents car ils ne diffèrent que par leur LSB.

On voit que dans le code binaire réfléchi deux combinaisons voisines sont adjacentes quel que soit ce nombre. On peut également le dire de deux combinaisons symétriques par rapport à l'un des axes qui ont servi à la construction du code. Ainsi , prenons les deux combinaisons situées trois lignes au

dessus et au dessous de l'axe de symétrie des trois premières colonnes combinaisons 5 et 10 nous trouvons: 0111 et 1111 qui sont deux nombres binaires adjacents

Cependant les propriétés de symétrie ne s'arrêtent pas là car les axes de symétrie sont réfléchis également lors de la construction du code. Nous pouvons donc voir le code binaire réfléchi comme ci dessous:

RANG DE LA COMBINAISON	CODE BR 4 BITS				
0	0	0	0	0	
1	0	0	0	1	Axe de symétrie de la première colonne
2	0	0	1	1	
3	0	0	1	0	Axe de symétrie des deux premières colonnes
4	0	1	1	0	
5	0	1	1	1	Réflexion du premier Axe
6	0	1	0	1	
7	0	1	0	0	Axe de symétrie des trois premières colonnes
8	1	1	0	0	
9	1	1	0	1	Réflexion du premier Axe
10	1	1	1	1	
11	1	1	1	0	Réflexion du deuxième Axe
12	1	0	1	0	
13	1	0	1	1	Réflexion du premier Axe
14	1	0	0	1	
15	1	0	0	0	

On peut constater également que si le code est écrit sur un cylindre, la dernière combinaison voisinant avec la première, ces deux combinaisons sont adjacentes, et toutes les propriétés de symétrie se conservent, il n'y a pas de discontinuité.

Le **code binaire réfléchi n'est pas un code pondéré.** On dit qu'un code binaire est pondéré lorsqu'on peut donner à un bit une valeur (un poids) en fonction de sa position dans le nombre ainsi on sait que dans le code binaire naturel, le bit le plus à droite dans le nombre vaut 1, le suivant 2, puis 4 etc...la combinaison 12 est la combinaison qui contient le bit de poids 8 et celui de poids 4 elle s'écrit 1100. Dans le code binaire réfléchi un bit n'a pas une valeur en fonction de sa place dans le nombre. Si nous regardons la combinaison de rang 15 du code BR 1000 nous serions tentés de dire que le 4ème bit à le poids 15, mais alors que dire de la combinaison 1111 de rang 10.

Ce code n'est pas pondéré, il ne peut être employé pour faire des opérations arithmétiques, non seulement additions et soustractions qui viennent à l'esprit immédiatement mais également des comparaisons.

Code binaire signé, Code complément à deux

Jusqu'à présent nous n'avons pas parlé du signe des nombres (+ ou -). Il est possible de donner un signe aux nombres binaires, il suffit pour cela d'ajouter un bit qui donnera cette information. Le **code complément à deux** est un code binaire signé qui est constitué pour sa partie positive de nombres binaire dont le bit de poids fort est un 0 et pour sa partie négative du complément à deux du nombre positif. Nous avons déjà vu le mécanisme de construction du complément à 2 d'un nombre et l'utilisation de ce complément à 2 pour transformer les soustractions en additions. L'utilisation de ce code sera essentiellement arithmétique. Il est très employé par les microprocesseurs.

Le code complément à 2 peut être réalisé à N bits nous l'écrierons ici à 4 bits.

CODE BINAIRE COMPLEMENT à 2	EQUIVALENT DECIMAL
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

On peut constater que le code complément à deux 4 bits est limité à +7 en positif et -8 en négatif. Le bit de poids fort donnant l'information sur le signe du nombre, toute tentative d'écrire des nombres dépassant ces valeurs entraînerait une erreur.

Exemples d'application:

a/ Additionner 4 et 2

Nous ferons l'addition des deux nombres positifs +4 (0100) et +2 (0010)
Soit l'addition:

0100 + 0010 = 0110

Nous voyons que le résultat (0110) est positif et égal à 6

b/ Soustraire 2 à 5

Nous ferons l'addition de +5 (0101) et -2 (1110)

Soit l'addition:

0101 + 1110 = 0011 (sans tenir compte du 5ème bits)

Le résultat (0011) est positif et égal à 3

c/ Soustraire 5 à 2

Nous ferons l'addition de +2 (0010) avec -5 (1011)

Soit l'addition:

0010 + 1011 = 1101

Nous voyons que le résultat est négatif en supposant que l'opération ait été effectuée en 16 bits, nous n'aurions pas le code à notre disposition (65536 combinaisons) pour connaître la grandeur du résultat, il suffit d'en faire le complément à 2 pour le rendre positif et plus facilement convertible.
Ici le complément à deux de 1101 nous donne 0011 ce qui est +3 donc le résultat 1101 est égal à -3.

d/ Additionner 5 et 4

Nous ferons l'addition de +5 (0101) avec +4 (0100)

Soit l'addition:

0101 + 0100 = 1001

Nous voyons que l'addition de deux nombres positifs nous donne un résultat négatif ce qui est anormal. **Une erreur a été effectuée par dépassement de la capacité du code, nous avons essayé d'écrire +9 avec un code qui ne peut dépasser +7.** Lorsque nous étudierons les microprocesseurs, ce type d'erreur sera appelée Overflow.

Code binaire avec bit de parité.

Les bits de parité ajoutés aux codes binaires sont destinés à accroître la fiabilité des systèmes. Le principe consiste à ajouter un bit à chaque combinaison du code de telle sorte que le nombre de "1" contenu dans chaque combinaison soit paire ou impaire suivant une convention entre émetteur et récepteur. Le récepteur recevant une combinaison binaire vérifie que le nombre de 1 est conforme à la convention. Lorsque la vérification est faite, le récepteur élimine le bit de parité dont l'usage est limité à la transmission de l'information et sa vérification.

La vérification s'effectue en faisant la somme modulo 2 des bits faisant partie de la combinaison binaire. La somme modulo 2 est désignée par le symbole $\oplus$, elle consiste à additionner sans tenir compte des reports

Exemples:

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 1 = 0$
$1 \oplus 1 \oplus 1 = 1$
$1 \oplus 1 \oplus 1 \oplus 1 = 0$

Ainsi lorsqu'on additionne un nombre pair de "1" le résultat donne 0, si le nombre de "1" est impair le résultat donne 1.

Exemple de code avec bit de parité

Il est possible d'ajouter un bit de parité à tous codes, prenons quelques combinaisons du code binaire naturel et imposons une parité impaire:

CODE BINAIRE NATUREL				↓ Bit de parité (parité impaire)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1

III - 6 - Code auto-correcteur d'erreur de Hamming

Le code de Hamming nécessite pour un code de base de 4 bits, 3 bits de parité. Ils sont placés dans les colonnes 1,2,4 le code 4 bits étant lui même logé dans les colonnes 3,5,6,7.

Les bits de parité sont élaborés à partir des équations ci-dessous:

$1 \oplus 3 \oplus 5 \oplus 7 = 0$
$2 \oplus 3 \oplus 6 \oplus 7 = 0$
$4 \oplus 5 \oplus 6 \oplus 7 = 0$

Exemple: élaboration des bits de parité pour la combinaison 6 du code binaire naturel.

Commençons par loger le code binaire naturel dans ses colonnes:

7	6	5	4	3	2	1
0	1	1	Z	0	Y	X

Détermination du bit de parité colonne 1 nous partons de l'équation déjà définie:

$1 \oplus 3 \oplus 5 \oplus 7 = 0$

En remplaçant le N° des colonnes par le bit correspondant nous obtenons:

$(X) \oplus 0 \oplus 1 \oplus 0 = 0$

pour que l'égalité soit vérifiée il faut que X=1 d'où:

7	6	5	4	3	2	1
0	1	1	Z	0	Y	1

Répétons ces opérations pour les bits 2 et 4

$2 \oplus 3 \oplus 6 \oplus 7 = 0$

$Y \oplus 0 \oplus 1 \oplus 0 = 0$

$Y=1$

et

$4 \oplus 5 \oplus 6 \oplus 7 = 0$

$Z \oplus 1 \oplus 1 \oplus 0 = 0$

$Z=0$

La combinaison 6 devient donc:

7	6	5	4	3	2	1
0	1	1	0	0	1	1

Code de Hamming réalisé à partir du code binaire naturel 4 bits

7	6	5	4	3	2	1
0	0	0	0	0	0	0
0	0	0	0	1	1	1
0	0	1	1	0	0	1
0	0	1	1	1	1	0
0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	1	0	0	1	1
0	1	1	0	1	0	0
1	0	0	1	0	1	1
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	0	1	0	1
1	1	0	0	0	0	1
1	1	0	0	1	1	0
1	1	1	1	0	0	0
1	1	1	1	1	1	1

Emploi du code de Hamming:

Le code de Hamming va permettre à un récepteur de vérifier l'exactitude de l'information reçue, et au cas ou une erreur serait détectée de la corriger automatiquement;
Pour ce faire, le récepteur va effectuer les trois tests qui nous ont permis d'élaborer les bits de parité nous les nommerons □□□

$1 \oplus 3 \oplus 5 \oplus 7 = \boldsymbol{\alpha}$

$2 \oplus 3 \oplus 6 \oplus 7 = \boldsymbol{\beta}$

$4 \oplus 5 \oplus 6 \oplus 7 = \boldsymbol{\gamma}$

Les résultats doivent donner 0, si ce n'est pas le cas une erreur s'est glissée, pour la correction le récepteur rangera le résultat des tests de la façon suivante:

γ β α

Ce qui composera le N° de la colonne erronée. Il suffira pour corriger l'erreur d'inverser le bit de la colonne en question. Tout ceci bien évidemment pourra s'automatiser sans difficulté.

Exemple:

Soit l'émission du nombre binaire: 1 1 0 0 0 0 1

et le nombre reçu 1 1 1 0 0 0 1

Tests à la réception:

$1 \oplus 0 \oplus 1 \oplus 1 = 1$ **α**
$0 \oplus 0 \oplus 1 \oplus 1 = 0$ **β**
$0 \oplus 1 \oplus 1 \oplus 1 = 1$ **γ**

Le résultat n'est pas nul , rangé comme indiqué plus haut, **γ β α** il nous donne le nombre 101=5 il faut donc inverser le bit de la colonne 5

Nombre binaire reçu: 1 1 1 0 0 0 1

Nombre corrigé par l'inversion du bit 5 1 1 0 0 0 0 1

Le code de Hamming est, si l'on réfléchi bien simple dans sa conception, encore fallait-il y penser. Nous laisserons au lecteur le soin à travers les exercices de fin de chapître de faire ce raisonnement. Cependant, tout est basé sur la notion de probabilité, il faut pour que ceci soit viable que la probabilité pour que deux erreurs se glissent à l'intérieur d'une seule combinaison soit infiniment petite.

Code ASCII

Le code ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) est un code utilisé pour les échanges entre notamment les ordinateurs et leurs périphériques tels que consoles, imprimantes etc..
Il permet le codage des caractères d'un clavier ainsi que d'un certain nombre de commandes.
La totalité du code de la page suivante nécessite 7 bits mais on peut voir que si l'on est prêt à se passer des minuscules le code pourra se ramener à 6 bits.

CODE ASCII

CODE Partie Haute→ Partie Basse ↓	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	NUL	DLE	SP	0		P		p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Signification des abréviations:

NUL	Nul	
SOH	Start Of Heating	Début d'entête
STX	Start of Text	Début de texte
ETX	End of Text	Fin de texte
EOT	End Of Transmission	Fin de transmission
ENQ	Enquiry	Demande d'informations
ACK	Acknowledge	Accusé de réception
BEL	Bell	Signal sonore
BS	Backspace	Retour d'un caractère en arrière
HT	Horizontal Tabulation	Tabulation horizontale
LF	Line Feed	Aller à la ligne suivante
VT	Vertical Tabulation	Tabulation verticale
FF	Form Feet	Aller à la page suivante
CR	Carriage Return	Retour du charriot (retour en début de ligne)
SO	Shift Out	Hors code
SI	Shift In	En code
DLE	Data Link Escape	Echappement de transmission
DCx	Device Control x	Contrôle de circuit x
NAK	Negative Acknowledge	Mauvaise réception
SYN	Synchronous Idle	Synchronisation
ETB	End Transmission Block	Fin de transmission de bloc
CAN	Cancel	Annulation
EM	End of Medium	Fin de support
SUB	Substitute	Substitution
ESC	Escape	Echappement
FS	File Separator	Séparateur de fichiers
GS	Group Separator	Séparateur de groupe
RS	Record Separator	Séparateur d'enregistrements
US	Unit Separator	Séparateur d'unités
SP	Space	Espace
DEL	Delete	Effacement

Exercices

a/ Quelle est la propriété nécessaire que doit possèder un code binaire pour qu'on puisse effectuer des opérations arithmétiques à l'aide de ses combinaisons?

b/ Si l'on travaille avec des nombres de 8 bits incluant un bit de parité avec parité impaire, ajouter le bit de parité aux nombres ci-dessous:
0011000, 1011101, 1111111, 0000000, 1000000

c/ Écrire en code complément à 2 8 bits les nombres suivants:
+1, -1, +122, -55, - 128

d/ Qu'est-ce que des combinaisons binaires adjacentes

e/ Construire le code binaire réfléchi 4 bits

f/ Les informations suivantes nous parviennent en code de Hamming 7 bits réalisé à partir du code binaire naturel les vérifier et éventuellement les corriger et en extraire l'information binaire naturelle.
0011110, 1101011, 1101001
g/ Réflexion sur les tests de Hamming. En supposant qu'il n'y a qu'une erreur dans un nombre, supposez qu'elle se trouve d'abord dans la colonne 1, puis dans la 2, puis dans la 3 etc et tirez en des conclusions quant au choix des bits testés dans chacun des tests.
Dans un code réalisé avec 4 bits de parité, combien de bits utiles pourrait-on avoir, où seraient-ils placés, quels seraient les tests à effectuer.

h/ Quels codes doit recevoir une imprimante qui laisse 3 espaces en début de ligne puis écrit IUT en lettres majuscules revient en début de ligne et passe à la ligne suivante.