Codification et représentation a-Numériques

1 Objectifs

- Savoir coder et décoder un mot selon le code ASCII.
- Savoir coder un nombre décimal en BCD et vice versa.
- Savoir coder un nombre décimal ou BCD en code « BCD+3 » et vice versa.
- Savoir faire les additions en code BCD.
- Savoir coder un nombre binaire en code Gray et vice versa.
- Savoir comment la machine détecte et corrige une erreur de transmission.

2 Introduction

Nous avons vu dans le chapitre précédent comment sont codés les nombres par notre ordinateur. Nous savons donc comment est codé un nombre positif, négatif ainsi que les réels. Mais comment est représenté un texte dans un ordinateur ?une image ? un son ?, etc.. Il existe pour cela de multiples codes, nous verrons dans ce chapitre quelques types de codes existants.

3 Le code ASCII (American Standard Code for Information Interchange).

Un ordinateur ne serait pas d'une grande utilité s'il n'était pas capable de traiter l'information non numérique. On veut dire par là qu'un ordinateur doit reconnaître des codes qui correspondent à des nombres, des lettres et des caractères spéciaux. Les codes de ce genre sont dits alphanumériques. Le code ASCII a été créé pour représenter les caractères alphanumériques usuels de l'anglais. Il transforme tout symbole d'un alphabet contenant 87 caractères (les 10 chiffres décimaux, 26 lettres majuscules, 26 lettres minuscules et environ 25 caractères spéciaux par exemple +,-,*,/,$, etc.…) en un mot binaire de 7 bits (car $2^6 < 87 < 2^7$). Ce code est une norme presque universelle dans les transmissions entre machines qui s'effectuent souvent sur un format de 8 bits [4]. Le huitième bit est dit de parité sert à détecter les erreurs de transmission. Le tableau 13 illustre quelques exemples de mots du code ASCII.

Exemples de codes :

Le code binaire est donné par $b_6\, b_5\, b_4\, b_3\, b_2\, b_0$ (b_0 est le bit de poids faible)

- Le code de U (majuscule) est représenté par 1010101 en code Ascii, par 85 en code décimale et par 55 en code hexadécimal.
- u (minuscule) est $1110101_{(2)}$, soit $117_{(10)}$ soit $75_{(16)}$.
- Le code ascii 1000001 $=41_{(16)}=65_{(10)}$.
- $(9)_{10}$ = (011**1001**) en code ASCII.

Mais le problème est que cette table a été inventée par les Américains : les accents sont donc inexistants. Néanmoins, avec le développement des technologies de l'information, des représentations plus évoluées ont vu le jour, dont le code Unicode, comportant 16 bits, pour un total de 65536 mots possibles et permettant de représenter de nombreux caractères de différents alphabets utilisés un peu partout dans le monde.

Binaire				Hexadécimal	Décimal	0	16	32	48	64	80	96	112
0	0	0	0	0	+0	NUL	TC7	SP	0	@	P	`	p
0	0	0	1	1	+1	TC1	DC1	!	1	A	Q	a	q
0	0	1	0	2	+2	TC2	DC2	"	2	B	R	b	r
0	0	1	1	3	+3	TC3	DC3	#	3	C	S	c	s
0	1	0	0	4	+4	TC4	DC4	$	4	D	T	d	t
0	1	0	1	5	+5	TC5	TC8	%	5	E	U	e	u
0	1	1	0	6	+6	TC6	TC9	&	6	F	V	f	v
0	1	1	1	7	+7	BEL	TC10	'	7	G	W	g	w
1	0	0	0	8	+8	FE0	CAN	(	8	H	X	h	x
1	0	0	1	9	+9	FE1	EM	)	9	I	Y	i	y
1	0	1	0	A	+10	FE2	SUB	*	:	J	Z	j	z
1	0	1	1	B	+11	FE3	ESC	+	;	K	[	k	{
1	1	0	0	C	+12	FE4	IS4	,	<	L	\	l	|
1	1	0	1	D	+13	FE5	IS3	-	=	M	]	m	}
1	1	1	0	E	+14	SO	IS2	.	>	N	^	n	~
1	1	1	1	F	+15	SI	IS1	/	?	O	_	o	DEL

Tableau 13 : Table des codes ASCII

4 Unicode

Le code UNICODE a été développé par le Consortium Unicode. Contrairement au code ASCII qui permet d'utiliser seulement les codes 0 à 127, UNICODE utilise des codes de valeurs bien plus grandes [4]. Il possède 16 bits de largeur et permet de coder 65536 caractères différents. Il permet de représenter les caractères spécifiques aux différentes langues : latins (accentués ou non), grecs, arabes, cyrillics, arméniens, hébreux, thaï, hiragana, katakana... L'alphabet Chinois Kanji comporte à lui seul 6879 caractères. L'Unicode est

accepté par les systèmes d'exploitation, applications et langages actuels tels que Windows, JAVA, etc. Il permet donc :

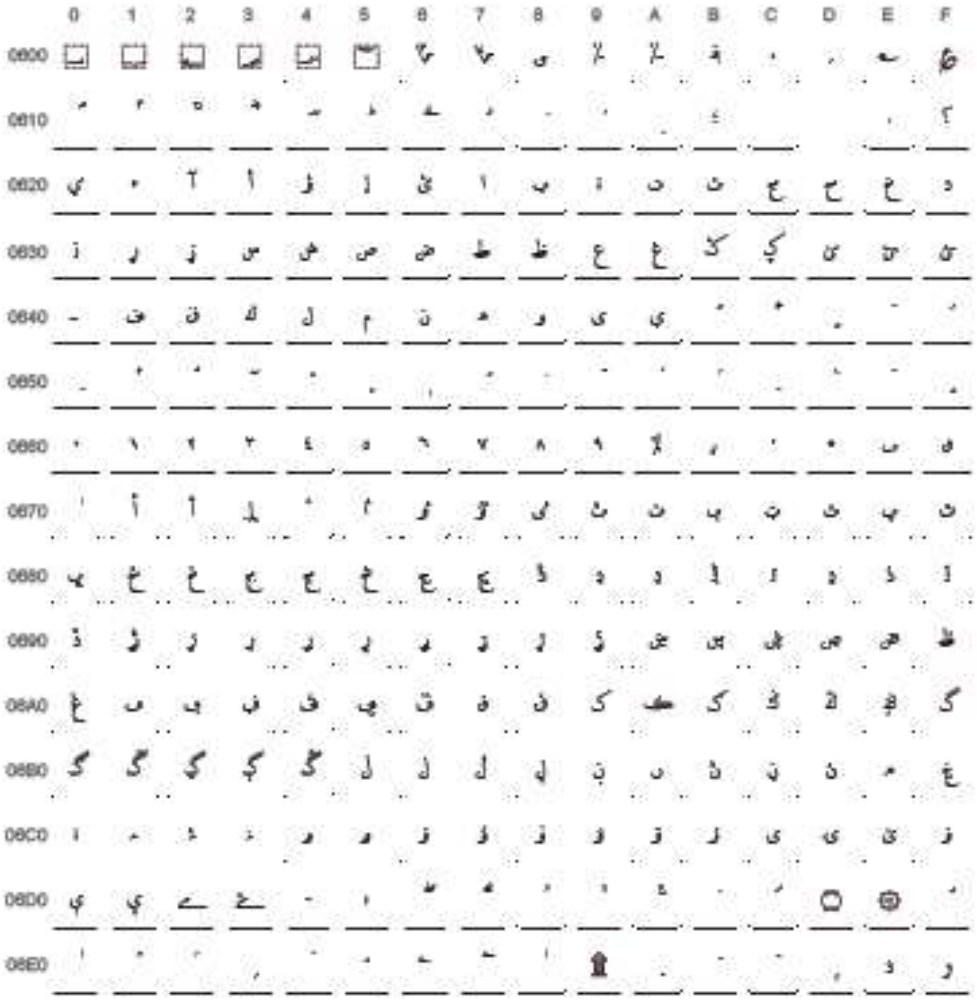
- D'unifier l'encodage de caractères, quelque soit le système d'exploitation,
- D'utiliser simultanément plusieurs alphabets et de multiples symboles dans un même document,

L'Unicode définie donc une correspondance entre symboles et nombres. Chaque caractère ou symbole est représenté par un code qui est noté U+xxxx où xxxx est en hexadécimal, et comporte 4 à 6 chiffres :

- 4 chiffres pour le plan multilingue de base (entre U+0000 et U+FFFF) ;
- 5 chiffres pour les 15 plans suivants (entre U+10000 et U+FFFFF) ;
- 6 chiffres pour le dernier plan (entre U+100000 et U+10FFFF).

Ainsi, le caractère nommé (*f*) a un index de U+0192. Il appartient au premier plan.
256 caractères arabes sont codés par l'unicode grâce au bloc « Arabe », entre U+0600 à U+06FF. par exemple le caractère 'ا'est codé par U+0623.

Voici une toute petite partie des tables UNICODE destiné pour la langue arabe [4],[5]:



5 UTF-8.

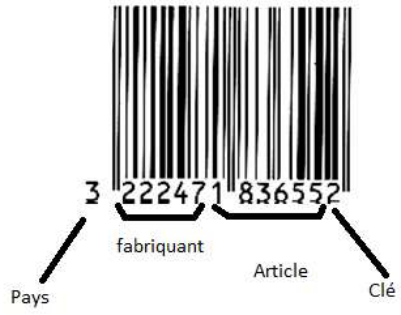
L'UTF-8 rassemble le meilleur de deux mondes: l'efficacité de l'ASCII et l'étendue de l'Unicode. Généralement en UNICODE, un caractère prend 2 octets. Autrement dit, le moindre texte prend deux fois plus de place qu'en ASCII. Ce qui permet de coder 65 535 caractères. De plus, si on prend un texte en français, la grande majorité des caractères utilisent seulement le code ASCII. Seuls quelques rares caractères nécessitent L'UNICODE. D'où l'intérêt du codage UTF-8 [6].

Un texte en UTF-8 est simple: il est partout en ASCII, et dès qu'on a besoin d'un caractère appartenant à l'Unicode, on utilise un caractère spécial signalant "attention, le caractère suivant est en Unicode". D'ailleurs l'UTF-8 a été adopté comme norme pour l'encodage des fichiers XML. La plupart des navigateurs récents supportent également l'UTF-8 et le détectent automatiquement dans les pages HTML.

6 Le Code Barre

Ce principe de codage, apparu dans les années 80, est largement utilisé sur les produits de grande consommation, car il facilite la gestion des produits. Le marquage comporte un certain nombre de barres verticales ainsi que 13 chiffres :

- Le 1er chiffre désigne le pays d'origine : 3 = France, 4 = Allemagne, 0 = U.S.A, etc. …
- Les cinq suivants sont ceux du code « fabricant »,
- Les six autres sont ceux du code de l'article,
- Le dernier étant une clé de contrôle
- Les barres représentent le codage de ces chiffres sur 7 bits, à chaque chiffre est attribué un ensemble de 7 espaces blancs ou noirs (voir figure ci_dessous) [7],[8].



7 Le code BCD

Le code BCD, qui est l'abréviation de Binary Coded Decimal en anglais, est un code binaire permettant de convertir directement un nombre décimal en un nombre binaire [1], [2]. Il est utilisé par les machines à calculer. Pour coder un nombre décimal en BCD, on va coder séparément chaque chiffre du nombre de base dix en Binaire selon le tableau 14.

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

Tableau 14 : Codage des chiffres décimaux par le code BCD

On appelle ce code aussi code 8421 car les 4 bits (du plus fort au plus faible) pondèrent les nombres 8, 4, 2 et 1 respectivement. Ainsi, 1001 correspond à 9 car 9 = **1** x 8 + **0**x 4 + **0** x 2 + **1** x 1. En BCD, chaque chiffre d'un nombre peut prendre une valeur entre 0 et 9 c'est-à-dire entre {0000 et 1001}. Un nombre de plusieurs décimaux est formé en groupant les quartets.

Exemple :
Coder le nombre décimal $804_{(10)}$ en code BCD

Solution
$8\ 0\ 4_{(10)} = 1000\ 0000\ 0100_{(BCD)}$.

Les opérations arithmétiques effectuées dans ce code sont plus compliquées qu'en binaire naturel.
Exemple : effectuer l'addition suivante en BCD :

$$153_{(10)} + 351_{(10)}$$

Solution
$1\ 5\ 3_{(10)} = 0001\ 0101\ 0011_{(BCD)}$
$3\ 5\ 1_{(10)} = 0011\ 0101\ 0001_{(BCD)}$

```
  0001  0101  0011
+ 0011  0101  0001
-------------------
  0100  1010  0100
```

Nous rencontrons ici un mot codé qui ne correspond pas à une valeur BCD connue. Pour résoudre ce problème, on peut ajouter la valeur $(6)_{10} = (0110)_2$ à ce mot codé, ce qui donnera

```
  1010
  0110
 -----
 10000
```

Ajouter 1 (report) au nombre suivant, le résultat sera donc :

0101 0000 0100 $_{(BCD)}$.=5 0 4 $_{(10)}$

Cette difficulté provient du fait quatre bits donnent 2^4=16 bits dont 10 seulement servent à coder les chiffres décimaux.

Remarque *:*

- Le nombre codé en BCD ne correspond pas au nombre décimal converti en binaire naturel.
- Les combinaisons supérieures à 9 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).

8 Le code « Plus Trois » ou « BCD+3 »

L'inconvénient cité ci-dessus à propos des opérations arithmétiques en BCD a été contourné dans les premiers ordinateurs à l'aide du code « plus Trois »[1]. Comme son nom l'indique, ce code consiste à calculer pour chaque chiffre décimal (de 0 à 9) son code BCD, puis lui ajouter la valeur 3. Voir le tableau suivant :

Remarque :

L'addition de deux nombres en BCD+3, nous mène vers deux cas possibles :

- L'addition génère un report : dans ce cas ajouter la valeur 3 au résultat.
- L'addition ne génère pas un report : dans ce cas, retrancher 3 du résultat.

Retrancher la valeur 3 (-3) revient à ajouter la valeur 1101. Car (-3) = -0011= 1101 en complément à 2.

Exemple 1 : avec report

```
  8                      1011
+ 6                      1001
 --    En BCD +3 →  ---------
 14                 0001 0100
                    0011 0011
                    ---------
                    0100 0111 (BCD+3)
```

Exemple 2 : sans report

```
  3                        0110
+ 4                        0111
 --     En BCD +3 →  ---------
  7                        1101
                         + 1101
                        ---------
                       (1) 1010 (BCD+3)
Ignorer le débordement
```

Décimal	BCD
0	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
4	0 1 1 1
5	1 0 0 0
6	1 0 0 1
7	1 0 1 0
8	1 0 1 1
9	1 1 0 0

Tableau 15 : Codage des chiffres décimaux par le code BCD+3

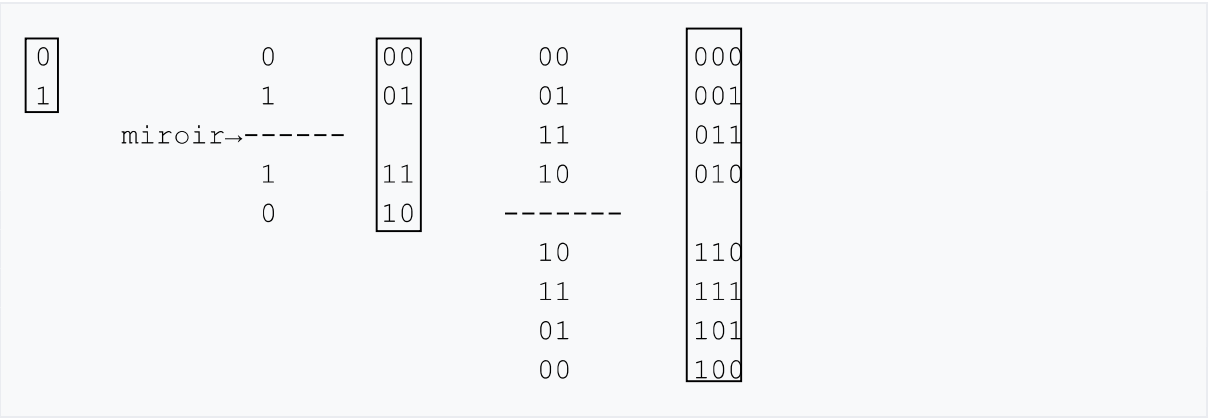
9 Le code Gray

Dans les conversions d'une grandeur analogique en une grandeur numérique, nous avons besoin d'un code dans lequel les grandeurs successives ne différent que d'un caractère. Cela évite les erreurs de transmission. Dans le système binaire naturelle, on peut remarquer que lors du passage de la valeur sept (7) à la valeur huit (8), c'est-à-dire le passage de 0111 à 1000, les quartes bits changent en même temps. S'il y a un problème de synchronisation, on peut détecter des valeurs erronées [1-3]. Le code de Gray est inventé afin d'éviter ce genre de problèmes. Il encode les entiers de telle façon que le passage d'un nombre au suivant ne change qu'un seul bit à la fois. On le retrouve également dans certains systèmes de télécommunication pour la correction d'erreur. Le tableau 15 donne l'équivalent en code Gray des entiers de 0 à15 ainsi que leur équivalent en binaire.

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

Tableau 16 : Code Gray

Le nom du code gray ou binaire réfléchi veut dire que les bits du code Gray peuvent être générés par réflexion (comme une réflexion dans un miroir) comme l'illustre le tableau suivant, puis on rajoute un 0 puis un 1 au début (à gauche) de chacun des codes. On a ainsi doublé le nombre de codes formés.



9.1 Conversion du code binaire en code gray

Soit B un nombre écrit en binaire pur sur m bits

$$B_{(2)} = Bm \ldots\ldots B4B3B2B1$$

Bm est le bit du poids fort

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

$$G_{(Gray)} = Gm \ldots\ldots G4G3G2G1$$

Le passage du binaire pur au code Gray se fait en effectuant une opération OU Exclusif. En désignant par Bn un bit quelconque en code binaire pur et par Gn le bit recherché en code Gray, nous avons alors :

Bm=Gm

Pour tout n <m, nous avons :

$$G_n = B_n \oplus B_{n+1}$$

L'opération consiste donc à calculer le OU exclusif entre le binaire de départ et ce même binaire décalé d'un rang à droite.

Exemple :

Trouver l'équivalent en code Gray du nombre binaire B= $0111_{(2)}$

Solution :

```
  0111
⊕
  0011
------
  0100
```

B= $0111_{(2)}$ =$0100_{(gray)}$

9.2 Conversion du code gray en code binaire

Pour la conversion du code Gray en code binaire appliquons la relation suivante :

Bm=Gm

Pour tout n <m, nous avons :

$B_n = G_n \oplus B_{n+1}$

Donc pour obtenir le code binaire d'un code de Gray donné, on passe de gauche (poid fort) à droite (poids faible). Le bit de poids fort est le même en binaire que dans le code de Gray. Le bit binaire suivant reste le même si le bit de Gray suivant vaut zéro et change si le bit de Gray suivant vaut 1. On répète cela pour tous les bits, jusqu'au bit de poids le plus faible.

Exemple : Trouver l'équivalent en code binaire du nombre binaire en code gray $G=(11011)_{Gray}$

Solution :

$G=(11011)_{Gray} = (10010)_{(2)}$.

10 Code détecteurs d'erreurs

Une erreur de transmission est traduite par le changement d'un bit dans le mot à transmettre d'où l'apparition d'un autre mot. Par exemple le code $(0011)_{(2)} = 3_{(10)}$ transmis par erreur $(0111)_{(7)}$ (inversion du troisième bit). Plusieurs codes ont été inventés afin de détecter les erreurs de transmissions. La plus utilisée est celle de bit de parité paire ou impaire [9],[10].

Exemple de code de parité paire (P) en code Gray

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

Tableau 17 : Code de parité paire en code Gray

P est le bit de parité paire

Si le nombre de 1 du code est pair alors P=0 sinon P=1.

Une simple erreur dans le code à transmettre ou dans le bit de parité sera immédiatement détectée. Le récepteur pourra demander une retransmission jusqu'à la réception du bon mot. Par contre une double erreur dans le code ne sera pas détectée.

11 Code détecteurs et correcteurs d'erreurs

Ce code est utilisé dans les bandes magnétiques. Généralement les codes sont envoyés ainsi que leurs parités paires (parité longitudinale (P_L) et parité verticale (P_V) comme l'illustre le tableau suivant. Si un code parmi les codes envoyés comporte une erreur, la machine localisera une erreur et la corrigera immédiatement.[9-11]

	0	**1**	**2**	**3**	**P_L**
	0	0	0	0	**0**
	0	0	0	0	**0**
	0	0	1	1	**0**
	0	1	1	0	**0**
P_V	**0**	**1**	**0**	**1**	

Tableau 18 : Code détecteurs et correcteurs d'erreurs

Le tableau affiche les mots à transmettre en code Gray (0, 1, 2 et 3). P_L est la parité calculée sur les lignes et P_V est la parité calculée sur les colonnes. Si par exemple il y a eu une erreur de transmission du nombre 3 (0010 en Gray), par exemple le $3^{ème}$ bit est erronée, la machine localisera une erreur dans la $3^{ème}$ ligne et dans la $4^{ème}$ colonne et la corrigera immédiatement.

12 Exercices

1. Donner le nom du code qui permet le codage les caractères et expliquer son principe.
2. Convertissez les nombres binaires suivants en code gray :
   a) (11010001)2
   b) (1000010)2
   c) (11011101)2
   d) (11000110)2

3. Convertissez chaque code gray en binaire:
   a) $10100000_{(gray)}$
   b) $10010101_{(gray)}$
   c) $11001001_{(gray)}$
   d) $10100111_{(gray)}$
4. Convertissez les nombres décimaux suivants en BCD:
   a) $(125)_{10}$
   b) $(1015)_{10}$
   c) $(187)_{10}$
   d) $(175)_{10}$
5. Convertissez chaque nombre BCD en nombre décimal
   a) 10100000
   b) 10010101
   c) 11001001
   d) 10100111
6. Additionner les nombres BCD suivants :
   a) $1100000_{(bcd)} + 1010111_{(bcd)}$
   b) $10010101_{(bcd)} + 1000100_{(bcd)}$
   c) $1011001_{(bcd)} + 1110101_{(bcd)}$
   d) $10000110_{(bcd)} + 1010101_{(bcd)}$

7. Convertissez les caractères suivants en code ASCII (voir le tableau ASCII):
   a) U
   b) *
   c) 65
   d) 107
   e) a
8. Donner l'équivalent de chaque caractère ASCII (voir le tableau ASCII):
   a) 0100000
   b) 0010101
   c) 1001001
   d) 0100111
9. Déterminer s'il y a une erreur dans les codes de parité suivants (parité paire) :
   a) 11101010
   b) 11101010
   c) 11101010
   d) 11101010
10. Associer les bits de parité impaire correspondant à chaque caractère suivant :
   a) 1101010
   b) 1110110
   c) 1101010
   d) 1111010

13 Conclusion

Il existe un ensemble de codes conçus pour pouvoir satisfaire aux besoins de l'informatique. Dans ce chapitre, nous avons dressé une liste non exhaustive des différents codes binaires utilisés pour représenter les caractères alphanumériques. Dans le chapitre suivant nous nous intéressons aux bases et aux propriétés fondamentales de l'algèbre de Boole indispensables à la compréhension du fonctionnement des systèmes numériques.