Nombres, codage et arithmétique informatique

A. Code binaire naturel

Lorsque nous écrivons 138 c'est l'écriture condensée de $1.10^2 + 3.10^1 + 8.10^0$ où 10^0 10^1 et 10^2 valent respectivement 1, 10 et 100 soit les puissances successives de la base 10 (unité, dizaine, centaine). Les coefficients qui multiplient les puissances de 10 peuvent prendre toutes les valeurs entières inférieures à la base donc 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 : c'est un *système de numération décimal* puisque la base vaut 10.

Le système binaire est un système de numération de base égale à 2. Les coefficients ne peuvent donc être que 0 ou 1. Les puissances successives de la base 2 sont 2^0, 2^1, 2^2, 2^3, 2^4, etc... soit 1, 2, 4, 8, 16, etc. Comme pour le système décimal nous pouvons utiliser une écriture condensée en ne retenant que les coefficients des puissances de 2. Ainsi le nombre binaire 1 0 0 0 1 0 1 0 vaut

$1 .2^7 + 0.2^6 + 0.2^5 + 0.2^4 + 1 .2^3 + 0.2^2 + 1 .2^1 + 0.2^0$

1.128 +1 .8 + 1.2

soit 138

Nous retrouvons notre nombre 138, mais il nous a fallu beaucoup plus de coefficients. Nous pouvons calculer le nombre binaire pour chacun des chiffres décimaux de 0 à 9. Pour cela quatre puissances de 2 nous sont nécessaires. Nous dirons qu'un chiffre décimal est exprimé par quatre *moments binaires ou* quatre bits. Ainsi le chiffre 4 sera représenté par 0100 soit $0.2^3 + 1.2^1 + 0.2^1 + 0.2^0$ ce qui donne bien 4. Ce système de représentation des nombres est le *code binaire naturel.*

Avec n bits il est possible de représenter 2^n nombres, le plus grand de ces nombres valant $2^n - 1$. Le système binaire est celui utilisé dans les ordinateurs, les calculateurs ou les microprocesseurs.

B. Code BCD ou DCB

Pour exprimer notre nombre 138 il est également possible de coder en binaire sur 4 bits chacun des trois chiffres décimaux 1, 3, 8. Nous obtenons alors

138 = 0001 0011 1000

1 3 8

Ce système de codification est appelé en terme anglo-saxon BCD *(Binary Coded Decimal) ou,* en français, DCB *(Décimal Codé Binaire).*

Ce code est également utilisé dans les microprocesseurs.

C. Code ASCII

Il est nécessaire en informatique de pouvoir coder les nombres mais aussi les lettres de l'alphabet et certains symboles. Deux codes sont fréquemment utilisés : le code ASCII et le code ISO. Dans ces deux codes, les informations sont exprimées avec 8 bits. Ainsi lorsqu'on appuie sur l'une des touches du clavier d'une télétype, la sortie de la télétype donne la codification sur 8 bits en ASCII du caractère (nombre, lettre ou symbole) correspondant à la touche appuyée (7 bits + bit de parité).

Voir tableau ASCII ci-dessous.

CODE ASCII

Bits 4 à 6	–	0	1	2	3	4	5	6	7
Bits 0 à 3	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

La table ci-dessus donne l'expression hexa-décimale du code ASCII sur 7 bits (bit de parité = bit 7 = zéro) pour les chiffres, les lettres et les symbôles.

Exemple : 5 a pour expression héxadécimale de son code ASCII : 35 (colonne 3 ligne 5) ce qui veut dire que le code ASCII de 5 est 00110101.

Inversement le code ASCII 01010100 a pour expression hexadécimale 54 : c'est donc le code ASCII de la lettre T (colonne 5, ligne 4).

Les codes binaire et BCD sont utilisés dans les ordinateurs pour faire des calculs. Les codes ASCII ne servent, eux, qu'à représenter les nombres, les lettres et les symboles en vue de les imprimer ou de les visualiser. Tous ces caractères codés en binaire, en BCD ou en ASCII sont rangés dans une mémoire qui n'est autre qu'une juxtaposition de cases mémoires, chacune d'elles contenant un bit. L'ensemble de huit cases mémoires est une position mémoire car elle permet de loger un mot binaire, c'est à dire un caractère de 8 bits, tout au moins pour les microprocesseurs classiques à 8 bits. En cours d'utilisation il y aura dans la mémoire une quantité de caractères binaires. Lorsque l'on veut exprimer ces suites de bits, il devient fastidieux de les représenter comme ils sont réellement dans la mémoire c'est là dire en binaire. Aussi on utilise une représentation particulière plus facile, c'est le *code hexadécimal* qui fait correspondre à chaque groupe de 4 bits un nombre ou une lettre A, B, C, D, E ou F.

N	2^3	2^2	2^1	2^0
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

. – *Code binaire naturel*

	Code hexa
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

2. – *Code hexadécimal*

Ainsi l'expression hexadécimale 2FA représente la suite binaire

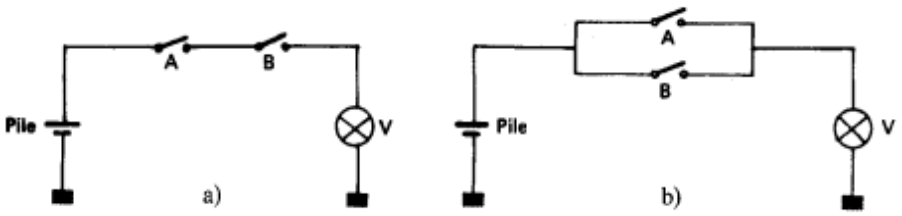
0010 1111 1010 soit 001011111010

2 F A

Le code hexadécimal n'est qu'une commodité d'écriture, mais il est fondamental pour l'utilisation des microprocesseurs.

2. Fonctions logiques et opérateurs

Un interrupteur électrique ne peut être que dans l'un ou l'autre des deux états suivants : ouvert ou fermé. De même pour un relais électromagnétique. L'interrupteur, le relais peuvent constituer des *variables binaires,* c'est à dire ne possédant que deux états possibles. En informatique les variables binaires doivent pouvoir passer très rapidement d'un état à l'autre. Aussi on utilisera des transistors à jonction (bloqués ou saturés), des transistors MOS (conducteurs ou bloqués), des tores magnétiques (aimantés ou non), etc.



Constituons un petit circuit d'éclairage avec une pile, deux interrupteurs A et B et un voyant V. Nous pouvons considérer A, B et V comme des variables binaires, le voyant étant allumé ou éteint, donner arbitrairement la valeur logique 1 aux interrupteurs quand ils sont fermés et 0 au voyant quand il est allumé. Pour la figure 2.1.a) nous pouvons dire :

le voyant V est allumé si A ET B sont fermés

ou

V a la valeur 1si A ET B ont la valeur 1.

ce que nous écrivons symboliquement V = A . B, le point symbolise la Fonction **ET**.

Nous donnons ci-dessous les dessins symboliques des circuits réalisant les différentes fonctions logiques, circuits appelés *opérateurs logiques* et qui utilisent des transistors en général.

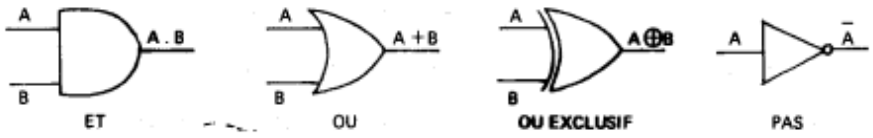


Fig. 2.2 : - Symboles des opérateurs logiques

Pour la figure 2.1.b) nous pouvons dire :

le voyant V est allumé si A OU B est fermé ou si les deux sont fermés

soit

V a la valeur 1 si A OU B a la valeur 1 ou si les deux ont la valeur 1

ce que nous écrivons symboliquement V = A + B ou encore A V B; le signe + ou le symbole V caractérise la FONCTION OU. Les fonctions logiques ET, OU sont les deux fonctions de base.

3. Bascules, registres, compteurs, décodeurs

• *Une bascule logique* est un élément permettant de mémoriser la valeur d'une variable binaire (logique). Elle est constituée à l'aide de plusieurs transistors connectés entre eux de façon à conserver l'état qui a été fourni à l'entrée à un moment donné. La valeur mémorisée est disponible en permanence à la sortie de la bascule.

• *Un registre* est une juxtaposition de bascules permettant de décaler les valeurs mémorisées, c'est à dire de les faire passer sur commande d'une case à la suivante. Là encore les bits mémorisés sont disponibles en permanence aux sorties du registre. Plusieurs décalages successifs permettent de sortir tous les bits du registre pour les envoyer dans un autre circuit. Plusieurs types de registres existent selon que l'information à mémoriser est rentrée bit par bit

(entrée série) ou tous les bits en même temps (entrée parallèle); de même pour la sortie (figure 2.3).

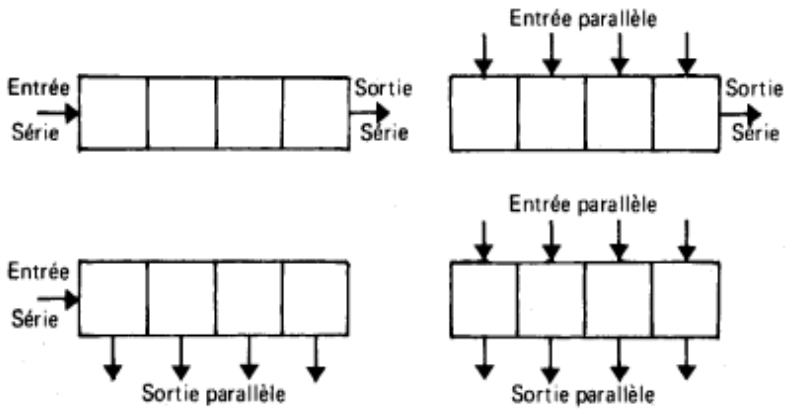


Fig. 2.3 : Registres de 4 bits

• *Un compteur* est un circuit comprenant *n* cases binaires, et comptant en binaire le nombre d'impulsions logiques qu'il reçoit sur son entrée. Ce nombre binaire mémorisé est disponible en permanence à la sortie parallèle du compteur. C'est l'équivalent d'un compteur kilométrique de voiture mais le nombre enregistré dans le compteur est codé en binaire et n'est pas visualisé.

• *Un décodeur* est un circuit qui établit une relation unique (ou plus exactement univoque) entre l'un des fils de sortie du décodeur et l'un des nombres binaires codés entre *n* fils d'entrée. Ainsi avec 4 fils d'entrée, un décodeur devra posséder 2^4 fils de sortie soit 16. Ainsi au nombre binaire 0101 correspondra la sortie S_5, au nombre binaire 1100 correspondra la sortie S_{12}, etc.

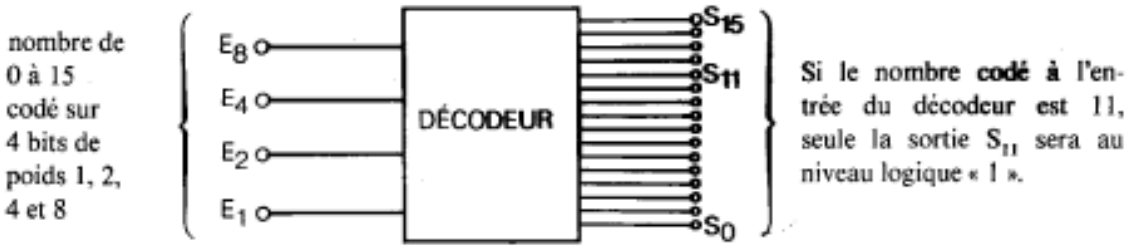


Fig. 2.4 : Décodeur

5. Arithmétique binaire

Tout microprocesseur permet d'effectuer une addition ou une soustraction de nombres binaires.

* *Addition binaire :* Le principe est le même que pour une addition dans le système décimal c'est à dire que l'addition de deux bits (ou de trois bits lorsqu'il y a un report précédent à prendre en compte) entraînera un report dès que le résultat dépassera 1. D'où :

0 + 0 = 0; 1 + 0 = 1; 0 + 1= 1. Aucun report pour ces trois cas.

1 + 1 = 0 et report = 1; 1 + 1 + 0 = 0 et report = 1. 1+1+1=1 et report =1. En effet 1+1+1=3 soit11.

* *Soustraction binaire. soit* à effectuer la soustraction 93 - 76. Pour cela les nombres doivent être signés : le bit 2^7 prend la valeur 0 pour un nombre positif et la valeur 1 pour un nombre négatif. La soustraction s'effectue en fait en additionnant à 93 le « complément à 2 » de + 76. Ce complément à 2 est obtenu en inversant bit à bit l'expression binaire de + 76 et en y ajoutant 1 :

```
report  | 111                          report    111
        01011101      93                       10001110      142
      + 01001100  soit + 76                  + 10111001  soit + 185
      ----------      ----                   ----------      -----
        10101001      169                    101000111       327
        ↑     ↑                              ↑
      bit 2^7  bit 2^0                       report au-delà des
                                             8 bits appelé CARRY.
```

+ 76	→	01001100
inversons bit à bit :		10110011
ajoutons 1		+ 1
complément à 2 de + 76 :		10110100

soustraction binaire	+ 93	→ 01011101
	complément à 2 de + 76	→ 10110100
93 – 76		1 00010001

CARRY

On ne tient pas compte du CARRY de sorte que le résultat est 00010001 soit 17 ce qui est bien la différence 93 - 76.

6. Mémoire

Une mémoire est une juxtaposition de cases-mémoires ou plus exactement de positions mémoires, c'est à dire d'ensembles de cases mémoires. Une mémoire contenant des mots de 8 bits (on les appelle des octets) sera une juxtaposition de positions mémoires de 8 bits. Pour sélectionner l'une de ces positions mémoires, il faudra connaître son implantation c'est à dire son *adresse*. Cette sélection se fera à partir d'un compteur et d'un décodeur; le compteur étant généralement externe à la mémoire. Le mot binaire à lire ou à écrire transitera par un registre d'entrée/sortie. Une mémoire ayant 16 bits d'adresse pourra contenir 2^{16} mots binaires soit 65536. En général on exprimera la capacité de la mémoire en kilomots, ou kilooctets pour des mots de 8 bits, ou plus simplement en K octets, le nombre K valant 1 024 (2^{10}). Donc avec 16 bits d'adresse la mémoire aura une capacité de 64 K octets. Une mémoire qui peut être lue ou

écrite est une mémoire lecture/écriture ou *mémoire vive ou RAM* (Random Access Memory). Une mémoire qui ne peut être que lue est une mémoire à lecture seulement, ou *mémoire morte ou ROM* (Read Only Memory) figure 2.5.

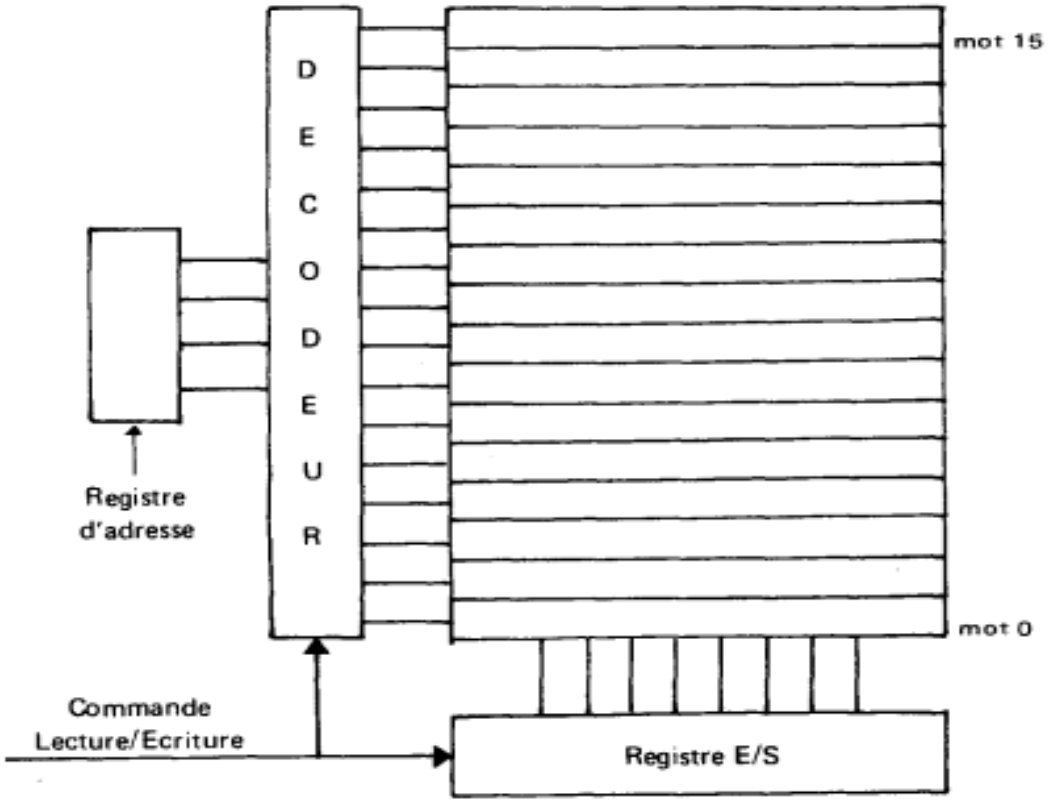


Fig. 2.5 : Mémoire de 16 mot

Exercices

Ex.1 : Dans le cas de binaires, bit signifie ……..

Ex. 2 : Convertir en décimaux les binaires suivants

(a) 10000000 (c) 00110011 (e) 00011111

(b) 00010000 (d) 01100100 (f) 11111111

Ex.3 : Convertir en binaires les décimaux suivants

(a) 23 (b) 39 (c) 55 (d) 48

Ex.4 : 204_{10} =……..……$_2$

Ex.5 : $11101110_2 = \ldots\ldots\ldots\ldots_{10}$

Ex.6 : Ecrire les hexa suivants sous formé binaire

(a) C	(c) F	(e) 1A	(g) A0	(i) 45
(b) 6	(d) E2	(f) 3D	(h) 8B	(j) D7

Ex.7 : Convertir les binaires suivants en hexa

(a) 1001	(c) 1101	(e) 10000000	(g) 00010101
(b) 1100	(d) 1111	(f) 01111110	(h) 11011011

Ex.8 : Convertir les hexas suivants en décimaux

(a) 7E (b) DB (c) 12A3 (d) 34CF

Ex.9 : $217_{10} = \ldots\ldots\ldots\ldots_{16}$

Ex.10 : $48\,373_{10} = \ldots\ldots\ldots\ldots_{16}$

Ex.11 : Effectuer les additions binaires suivantes

(a)	(b)	(c)	(d)
1010	1101	01011011	00111111
+0101	+0101	+ 00001111	+00011111

Ex.12 : Effectuer les soustractions binaires suivantes

(a)	(b)	(c)	(d)
1110	1010	01100110	01111000
- 1000	- 0101	- 00011010	- 00111111

Ex.13 : Indiquer si les nombres sous forme de complément à 2 qui suivent sont positifs ou négatifs

Ex.14 : Exprimer 1978 en binaire pur, en BCD et en ASCII.

Ex.15 : Sachant que N_1= 10010011 et N_2 =01111010 calculer les fonctions logiques N_1+ N_2, N_1.N_2, N_2 et exprimer les résultats en hexadécimal.

Ex.16 : Effectuer l'addition 54 + 39 en binaire et exprimer le résultat en hexadécimal.

Ex.17 : Effectuer l'addition 54 + 39 en BCD.

Ex.18 :Effectuer la soustraction 54 - 39 en binaire.