Représentation de l'information

1. L'information

1-1 Dualité état et temps

La notion d'information correspond à la connaissance d'un état donné parmi plusieurs possibles à un instant donné. La Figure 1 illustre cette notion avec un signal électrique. Elle montre qu'il y a deux états significatifs, l'état bas lorsque la tension est inférieure à une référence basse, et un état haut lorsque la tension est supérieure à une référence haute. Le troisième état, situé entre les références basse et haute, ne peut être utilisé comme support d'information. Pour qu'il y ait information, il faut préciser l'instant auquel on regarde l'état du signal : par exemple, en t_1 le signal est haut et en t_2, le signal est bas.

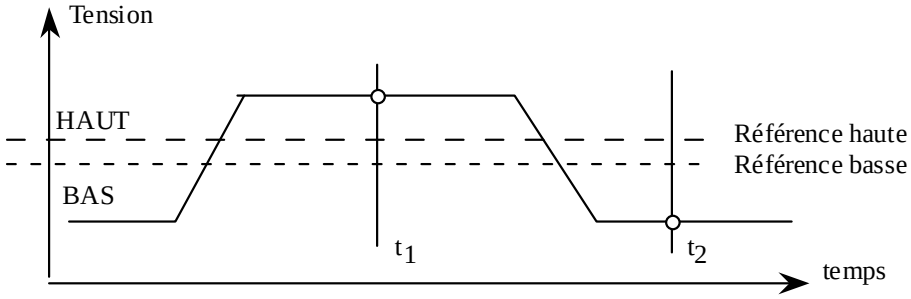

Figure 1 .- Information : état et temps.

1-2 Quantité d'information et codage de l'information

L'information est mesurable, avec une unité qui s'appelle le bit. La formule I (bits)= $\log_2(N)$ où N est le nombre d'états possibles donne la quantité d'information. Un bit correspond donc à la connaissance d'un état parmi deux. Par exemple, la connaissance d'un état parmi 8 correspond à une quantité d'information de I = $\log_2(8)$ = 3 bits. Les 8 états possibles sont repérés à l'aide de 3 chiffres binaires (ayant l'un des deux états possibles 0 ou 1), comme le montre la Table 1. La quantité d'information exprimée en bits étant égale au nombre de chiffres binaires correspondants, les chiffres binaires sont appelés bits.

Un mot de n bits correspond à n bits d'information, puisqu'il correspond à une configuration parmi 2^n. Mais, il faut préciser à quoi correspondent les 2^n configurations. En effet, le mot de n bits peut représenter une instruction machine ou un opérande, qui peut être un nombre ou un caractère. Nous présentons brièvement le codage ou représentation des nombres et des caractères.

Avant d'introduire les codages des nombres et des caractères, il est utile d'introduire la notation hexadécimale, qui n'est pas un codage de l'information, mais une manière simplifiée d'écrire des nombres binaires. Un nombre binaire de n bits peut être écrit à l'aide de n/4 chiffres hexadécimaux si n est multiple de 4 (ou

n/4 +1 sinon) en remplaçant chaque groupe de 4 chiffres binaires, en partant des poids faibles, par le chiffre hexadécimal correspondant (Table 2).

État	X_2	X_1	X_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Table 1 .- Représentation de huit états différents

Chiffre hexadécimal	d_3	d_2	d_1	d_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
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

Table 2 .- Notation hexadécimale

2. Représentation des nombres

La Figure 2 représente un mot de n bits.

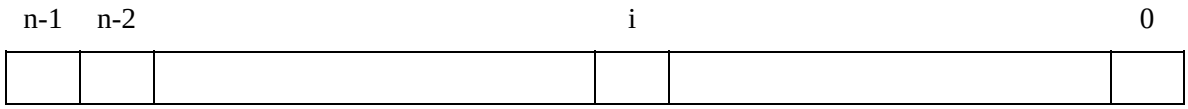

Figure 2 .- Mot de n bits

2-1 Les nombres entiers

2-1-1 Entiers positifs

Un mot de n bits peut représenter tous les nombres positifs compris entre 0 et 2^n-1. d_i étant le chiffre binaire de rang i, un mot de n bits correspond au nombre entier

décimal N = $\sum_{i=0}^{n-1} d_i 2^i$. Avec un octet, on peut donc représenter tous les entiers positifs entre 0 et 255. Un mot de 32 bits permet de représenter tous les entiers positifs entre 0 et 4 294 967 295.

2-1-2 Les entiers signés

La représentation des entiers signés pose un problème lié au zéro. En effet, il y a un nombre pair de configurations associées à n bits, à répartir entre nombres positifs, nombres négatifs et la valeur 0. Il y a plusieurs représentations possibles des entiers signés. Dans toutes les configurations, les nombres positifs sont représentés de la même manière, correspondant à la représentation des entiers positifs sur n-1 bits, avec $d_{n-1} = 0$. Le bit de poids fort d_{n-1} est appelé le bit de signe : il est à 0 pour les nombres positifs.

a) Valeur absolue et signe

Dans cette représentation, le bit n-1 est le bit de signe, et les bits 0 à n-2 donnent la valeur absolue. Un mot de n bits correspond au nombre entier décimal signé N = $(-1)^{d_{n-1}} . \sum_{i=0}^{n-2} d_i 2^i$, soit N = $-1 \sum_{i=0}^{n-2} d_i 2^i$ quand le bit de signe est à 1 et $\sum_{i=0}^{n-2} d_i 2^i$ lorsque le bit de signe est à zéro.

Un octet permet de représenter les entiers signés compris entre -127 et + 127. Il y a deux représentations possibles du zéro, qui sont 00000000_2 (+0) et 10000000_2 (-0). Un mot de 32 bits permet de représenter tous les entiers signés compris entre $-(2^{31}-1)$ et $2^{31}-1$, avec toujours deux zéros.

b) Entiers signés en complément à 1

Dans la représentation en complément à 1, le nombre négatif -N est obtenu en remplaçant chaque chiffre binaire d_i du nombre positif N par le complément $\overline{d_i}$. (cf. l'opération complément de l'algèbre de Boole) : les bits 1 sont remplacés par des 0 et réciproquement. Un octet permet de représenter tous les entiers signés compris entre -127 (1000000_2) et +127 (01111111_2), avec deux zéros qui sont 00000000_2 (+0) et 11111111_2 (-0).

c) Entiers signés en complément à 2

En complément à 2, un nombre est représenté par N = $- a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$. L'écriture en complément à 2 correspond donc à la situation où le bit de poids fort est de poids négatif.

La Table 3 donne l'ensemble des nombres en complément à 2 avec des mots de 3 bits

d_2	d_1	d_0	N
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	-4
1	0	1	-3
1	1	0	-2
1	1	1	-1

Table 3 .- Nombres en complément à 2 sur 3 bits

Un mot de n bits permet de représenter tous les entiers signés compris entre -2^{n-1} et $2^{n-1} - 1$. Il y a maintenant une seule représentation de 0 (qui est le 0 positif). Avec un octet, on représente les entiers signés entre -128 (10000000_2) et +127 (01111111_2). La représentation en complément à 2 d'un nombre négatif peut être obtenue à partir de la représentation en complément à 1, à laquelle on ajoute +1.

d) la représentation excès N.

L'excès N est choisi de manière à ce que la somme de l'excès et du nombre ne soit jamais négative. Cette somme est représentée comme un nombre positif normal.

La représentation en complément à 2 est la plus couramment utilisée pour l'arithmétique sur les nombres entiers. Elle a l'avantage de ne pas introduire de traitement particulier pour le signe dans le cas des additions et des soustractions, et de permettre une détection facile des cas de débordement.

Les représentations en signe-valeur absolue et en complément à un ont des opérations arithmétiques plus complexes. Elles ont aussi l'inconvénient d'avoir deux représentations de zéro. La représentation signe-valeur absolue est utilisée pour la multiplication des mantisses des nombres flottants. La représentation excès N est utilisée pour les exposants des nombres flottants. Elle permet de ramener la comparaison d'exposants de signe contraire à des comparaisons d'entiers positifs.

2-2 Représentation dite "flottante"

La représentation dite "flottante" a pour but de permettre de représenter une approximation des nombres réels, en permettant une différence non constante entre deux nombres représentés successifs. Dans cette représentation, un nombre est caractérisé par son signe, sa mantisse et son exposant qui est associé à une base : +/- $m.B^e$. En numération décimale, $1,5 \times 10^{15}$ est un exemple de nombre "flottant". En machine, la base B utilisée est 2. Dans la norme IEEE 754 qui est la plus utilisée maintenant, la mantisse, exprimée en signe et valeur absolue, est 1,f où 1 est implicite et f est la partie fractionnaire, correspondant aux puissances successives de 2^{-i}. L'exposant est exprimé en code excès N. La Figure 3 présente la représentation flottante double précision, sur 64 bits, qui est la plus couramment utilisée. La partie exposant donne sur 11 bits la valeur de l'exposant en code excès 1023. La partie

fractionnaire a 52 bits. Le bit 63 donne le signe de la partie fractionnaire. Le nombre flottant double précision correspond à

$$(-1)^{signe} \times (1,f_1f_2...f_{52}) \times 2^{(E-1023)}$$

La norme flottante IEEE 754 permet de représenter les nombres normalisés (bits de E différents de tous à 0 ou tous à 1), les nombres dénormalisés (E tous à 0 et partie fractionnaire non nulle), la valeur zéro (E tous à 0 et partie fractionnaire nulle), l'infini (E tous à 1 et partie fractionnaire nulle) et des caractères spéciaux (*Not a number* lorsque les bits de E sont tous à 1 et la partie fractionnaire est différente de 0).

En représentation double précision, la mantisse m est comprise entre 1 lorsque tous les bits f_1 à f_{52} sont à 0 et $2 - 2^{-52}$ lorsque tous les bits f_1 à f_{52} sont à 1. Le champ exposant E est compris entre 1 et 2046 pour les nombres normalisés. Compte tenu de l'excès 1023, l'exposant réel est donc compris entre -1022 et + 1023. Le plus petit nombre positif normalisé représentable est donc 2^{-1022} et le plus grand est voisin de 2×2^{1023} soit 2^{1024}.

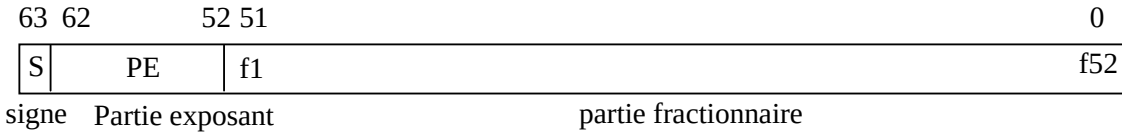

Figure 3 - Format flottant double précision.

La représentation simple précision sur 32 bits a 1 bit de signe, 8 bits de partie exposant (avec excès +127) et 23 bits pour la partie fractionnaire.

2-3 Représentation des nombres décimaux

Un certain nombre d'applications, notamment en gestion, exigent des calculs décimaux exacts, sans arrondis, ce qui implique de travailler avec des nombres décimaux. En effet, avec un nombre fixé de bits, il est impossible de convertir de manière exacte des nombres binaires en nombres décimaux et réciproquement. On utilise alors la représentation décimale codée binaire, dans laquelle chaque chiffre décimal est codé avec 4 chiffres binaires, selon la Table 4. Cette représentation utilise uniquement les chiffres décimaux de la notation hexadécimale.

Chiffre décimal	d_3	d_2	d_1	d_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

Table 4 - Décimal codé binaire

3. Représentation des caractères

La représentation des caractères est fondamentale. Les lettres de l'alphabet latin et un certain nombre de caractères usuels sont représentés par un octet, selon un code qui est le plus souvent le code ASCII (Table 5). Par exemple, dans ce code, la lettre "a" est représentée par 01000000_2 et le chiffre "9" par 00111001_2.

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

Table 5 : Codage ASCII