Leçon 6 : Codage des caractères et des nombres

I. Introduction

Les informations traitées par l'ordinateur sont de différents types (nombres, textes, instructions, images, séquences d'images animées, sons) mais elles sont toujours représentées à la base, sous forme binaire.

Le codage d'une information consiste à établir une correspondance entre sa représentation externe et sa représentation interne qui est une suite de bits. La transformation inverse est appelée décodage.

Les ordinateurs sont généralement organisés pour travailler sur une succession de groupe de bits appelés mots (qui peuvent être de 8,16, 32 bits) ; ils peuvent représenter des nombres ou des lettres de l'alphabet, en fonction d'une loi de codage, qui fait correspondre une combinaison possible à un nombre ou une lettre.

En rappel :

8 bits = 1 octects

Le kilo Octets (Ko): 1024 Octets

Le Méga Octets (Mo) : 1024 Kilo Octets

Le Giga Octets (Go) : 1024 Méga Octets

Le Tétra Octets (To): 1024 Giga Octets

II. Représentation des caractères

Les données non numériques correspondent aux caractères alphanumériques : A, B, …, Z, a, b, …z, 0, 1, …, 9 et aux caractères spéciaux : ?, !, $, etc. Le codage est réalisé par une table de correspondance propre à chaque code utilisé. Parmi les codes les plus connus, on peut citer :

a. BCD (Binary Coded Décimal)

Consiste à représenter chaque chiffre d'un nombre décimal par son équivalent binaire sur 4 bits. Exemple (3 -> 0011)

b. ASCII (American Standard Code for Information Interchange)

Permet le codage de 128 caractères sur 7 bits. Il englobe des lettres, des chiffes, des signes de ponctuations et un certain nombre de signaux de commande.

	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	_	o	DEL

Exemple : A => $(01000001)_2$ => $(41)_{16}$ => $(65)_{10}$

c. ASCII étendu (8 bits)

Ce code est normalisé ISO (International Standard Organisation). C'est une extension du code ASCII étendu. Il permet le codage de caractères sur 8 bits, soit 256 caractères possibles.

d. EBCDIC (Extended Binary Coded Decimal Internal Code)

Code à 8 éléments binaires utiles, soit 256 combinaisons possibles.

e. UNICODE (16 bits)

Ce code permet de représenter des caractères appartenant à plusieurs langues (arabes, hébreu, japonais, coréen,) : 65536 caractères

III. Représentation des nombres

Les valeurs numériques manipulées dans l'ordinateur, peuvent être soit des entiers naturels, soit des entiers relatifs, soit des nombres fractionnaires. Ces nombres vont être représentés en binaire pour pouvoir être traités correctement.

a. Représentation des entiers non signé

La représentation d'un entier non signé (entier naturel) est tout simplement sa représentation binaire, ce qui revient à faire sa conversion dans la base binaire. En rappel toute représentation de nombre est toujours effectue sur n bits (8, 16, 32)

Exemple :

Pour n = 8bits représenter $(4)_{10} = (0000100)_2$

- Le plus grand entier non signé représentable sur 8 bits est $(11111111)_2 = 2^8 - 1 = 255$
- Le plus petit entier non signe représentable sur 8 bits est $(00000000)_2 = 0$

Généralisation :

- Le plus grand entier non signé représentable sur n bits est $2^n - 1$
- Le plus petit entier non signe représentable sur 8 bits est 0
- Donc sur n bits l'intervalle des entiers non signé représentable est $[0 ; 2^n - 1]$

NB : Les instructions arithmétiques et quelques autres manipulent des entiers de taille fixe, qui ne peuvent prendre leurs valeurs que dans un intervalle. Si le résultat d'un calcul sort de cet intervalle, il ne peut pas être représenté par l'ordinateur : il se produit ce qu'on appelle un **débordement d'entier**.

b. Représentation des entiers signées

Il y a trois techniques permettant de représenter ce type d'entier :

i. Valeur absolue signée

Pour représenter un nombre relatif N en binaire, on procède comme suit :

- Représenter la valeur absolue de N en binaire sur (n-1) bits.
- Consacrer le nième bit pour le signe. Si N est positif alors le bit de signe sera positionné à 0. Si N est négatif alors le bit de signe sera positionné à 1.

Exemple

Pour n=8bits encoder $(25)_{10} = (..)_{vas}$ et $(-25)_{10} = (..)_{vas}$

$(25)_{10} = 16 + 9 = 16 + 8 + 1 = (00011001)_{vas}$

$|-25| = 25 = 16 + 9 = 16 + 8 + 1 = (0011001)$ sur 7 bits donc $(-25)_{10} = (10011001)_{vas}$

Pour n=8bits décoder $(10101000)_{vas} = (..)_{10}$ et $(01001010)_{vas} = (..)_{10}$

$(10101000)_{vas} = -(0101000) = -(32+8) = (-40)_{10}$

$(01001010)_{vas} = +(1001010) = +(64+8+2) = (+74)_{10}$

Pour n=8bits

- Le plus grand positif représentable : $(01111111)_{vas} = 2^7 - 1 = +127$
- Le plus petit positif représentable : $(00000000)_{vas} = +0$
- Négatif a plus petite valeur absolu : $(10000000)_{vas} = -0$
- Négatif a plus grande valeur absolu : $(11111111)_{vas} = -(2^7 - 1) = -127$

Généralisation :

- Le plus grand entier signé représentable en vas sur n bits est $2^{n-1} - 1$
- Le plus petit entier signe représentable en vas sur n bits est $-(2^{n-1} - 1)$

- Donc sur n bits l'intervalle des entiers non signé représentable est $[-(2^{n-1}-1) ; 2^{n-1}-1]$

ii. Complément à 1

On appelle complément à 1 de N ($\alpha 0\ \alpha 1 \ldots \alpha p$) le nombre N'= C1(N) tel que pour $0 \leq i \leq p$, vérifie :

- Si $\alpha i = 1$ alors $\alpha' i = 0$
- Si $\alpha i = 0$ alors $\alpha' i = 1$

Exemple pour N=01101110 C1 (N) = 10010001

Le codage en complément à 1 revient à inverser les bits à 1 en bits à 0 et vis vers ça. Lorsqu'il s'agit d'un nombre binaire pure (positif), son interprétation en décimal est directe et son bit du poids fort est à 0 alors que pour un nombre binaire négatif, il garde aussi son bit du poids fort à 1, mais ne peut être interpréter en décimal qu'en cherchant sa valeur absolue par le complément à 1.

Exemple

Pour n=8bits encoder $(25)_{10} = (..)_{C1}$ et $(-25)_{10} = (..)_{C1}$

$(25)_{10} = 16 + 9 = 16 + 8 + 1 = (00011001)_{C1}$

$|-25| = 25 = 16 + 9 = 16 + 8 + 1 = (0011001)$ sur 7 bits puis C1 = (1100110)

Donc $(-25)_{10} = (11100110)_{C1}$

Pour n=8bits décoder $(10101000)_{C1} = (..)_{10}$ et $(01001010)_{C1} = (..)_{10}$

$(10101000)_{C1} = -(0101000)$ puis C1 = - (1010111) = - (64+16+4+2+1) = $-(87)_{10}$

$(01001010)_{C1} = +(1001010) = +(64+8+2) = (+74)_{10}$

Pour n=8bits

- Le plus grand positif représentable : $(01111111)_{C1} = 2^7 - 1 = +127$
- Le plus petit positif représentable : $(00000000)_{C1} = +0$
- Négatif a plus petite valeur absolu : $(11111111)_{C1} = -0$
- Négatif a plus grande valeur absolu : $(10000000)_{C1} = -(2^7 - 1) = -127$

Généralisation :

- Le plus grand entier signé représentable en vas sur n bits est $2^{n-1} - 1$
- Le plus petit entier signe représentable en vas sur n bits est $-(2^{n-1} - 1)$
- Donc sur n bits l'intervalle des entiers non signé représentable est $[-(2^{n-1} - 1) ; 2^{n-1} - 1]$

iii. Complément à 2

On appelle complément à 2 d'un nombre binaire N, le nombre binaire N'= C1(N) +1.

Exemple pour N=01101110 C2 (N) = 10010001 + 1 = 10010010

Pour trouver la représentation d'un nombre négatif ; il suffit de chercher la représentation en complément à deux de sa valeur absolue.

Exemple

Pour n=8bits encoder $(25)_{10} = (..)_{C2}$ et $(-25)_{10} = (..)_{C2}$

$(25)_{10} = 16 + 9 = 16 + 8 + 1 = (00011001)_{C2}$

$|-25| = 25 = 16 + 9 = 16 + 8 + 1 = (0011001)$ sur 7 bits puis C1 = (1100110)

Puis C2 = C1 + 1 = 1100110 +1 = 1100111

Donc $(-25)_{10} = (11100111)_{C2}$

Pour n=8bits décoder $(10101000)_{C2} = (..)_{10}$ et $(01001010)_{C2} = (..)_{10}$

$(10101000)_{C2} = -(0101000)$ puis C1 = - (1010111)

Puis C2 = C1 +1 = - (1010111 + 1) = - (1011000) = - (64+16+8) = $-(88)_{10}$

$(01001010)_{C2} = +(1001010) = +(64+8+2) = (+74)_{10}$

Pour n=8bits

- Le plus grand positif représentable : $(01111111)_{C2} = 2^7 - 1 = +127$
- Le plus petit positif représentable : $(00000000)_{C2} = +0$
- Négatif a plus petite valeur absolu : $(11111111)_{C2} = -1$
- Négatif a plus grande valeur absolu : $(10000000)_{C2} = -(2^7) = -128$

Généralisation :

- Le plus grand entier signé représentable en vas sur n bits est $2^{n-1} - 1$
- Le plus petit entier signe représentable en vas sur n bits est $-(2^{n-1} - 1)$
- Donc sur n bits l'intervalle des entiers non signé représentable est $[-(2^{n-1}) ; 2^{n-1} - 1]$

c. Représentation des nombres fractionnaires (réels)

Les formes normalisées connue pour représenter un nombre fractionnaire sont : la forme normalisées IEEE 754 simple précision (sur 32 bits) et la forme normalisées IEEE 754 double précision (sur 64 bits).

i. Forme normalisées IEEE 754 simple précision

La représentation normalisée est sous la forme : $\mathbf{(-1)^s\ 1, M\ 2^{E-127}}$

- **S** : est le signe du nombre (0 : positif, 1 négatif) sur 1 bit

- **1** : est le premier 1 de la partie entière
- **M** : est la mantisse (constitue la partie fractionnaire après la transformation) sur 23 bits
- **E** : est l'exposant dû à cette transformation sur 8 bits
- **2** : est la base

Exemple :

Pour $(1,001101)_2 \times 2^3$ l'on a

E-127=3 alors E= 3 +127 = 130 = 128 +2 = $(1000\ 0010)_2$

S= 0

M = 00110100000000000000000

D'où la représentation est 0 1000 0010 00110100000000000000000

Pour $(1,00\underline{010})_2 \times 2^{-4}$ l'on a

E-127=-4 alors E= -4 +127 = 123 = 64 +32 + 16+8+2+1= $(0111\ 1011)_2$

S= 0

M = 00 010 010 010 010 010 010 010

D'où la représentation est 0 0111 1011 00 010 010 010 010 010 010 010

En rappel :

$(110,01)_2 = 1,1001 \times 2^2$

$(0,0010011)_2 = 1,0011 \times 2^{-3}$

$-(101)_2 = -1,01 \times 2^2$

$(0,00\ \underline{010}\ )_2 = (0,00\ 010\ \underline{010}\ )_2 = 1,0\ \underline{010} \times 2^{-4}$

ii. Forme normalisées IEEE 754 double précision

La représentation normalisée est sous la forme : $\mathbf{(-1)^s\ 1, M\ 2^{E-1023}}$

- **S** : est le signe du nombre (0 : positif, 1 négatif) sur 1 bit
- **1** : est le premier 1 de la partie entière
- **M** : est la mantisse (constitue la partie fractionnaire après la transformation) sur 52 bits
- **E** : est l'exposant dû à cette transformation sur 11 bits
- **2** : est la base