

CODAGE DE L'INFORMATION

NOMBRES ET CARACTÈRES

A.MAZZA

mazza8azzouz@gmail.com

Lycee Omar Ibn Abdelaziz

CPGE Oujda

October 1, 2024



Contents

- 1 Problématique
- 2 Représentation des nombres dans une base
- 3 Transcodage (ou conversion de base)
- 4 Codage des nombres entiers
- 5 Codage des nombres réels

Problématique

Introduction

Les informations traitées par les ordinateurs sont de différentes natures :

- nombres, texte,
- images, sons, vidéo,
- programmes, ...

Dans un ordinateur, elles sont toujours représentées sous forme d'un nombre binaire (BIT : **B**inary dig**IT**)

- une suite de 0 et de 1

Donc nous allons nous intéresser à la façon dont un nombre (entier ou réel) peut être représenté à l'intérieur d'un ordinateur.

Représentation des nombres dans une base

Représentation des nombres dans une base

Nous sommes habitués depuis l'enfance à utiliser l'écriture en base 10 des entiers : par exemple, 2985 représente le nombre $2 \times 10^3 + 9 \times 10^2 + 8 \times 10 + 5 \times 10^0$.

Mais plus généralement, pour tout entier $b \geq 2$ on peut définir la représentation en base b d'un entier en convenant que l'écriture $(a_p a_{p-1} \dots a_0)_b$ représente le nombre : $a_p \times b^p + a_{p-1} \times b^{p-1} + \dots + a_1 \times b + a_0$.

Pour s'assurer de l'unicité de l'écriture d'un entier dans une base donnée, il est nécessaire en outre d'imposer : $\forall k \in [0, p]$, $a_k \in [0, b-1]$ et $a_p \neq 0$.

Ainsi, en base 3 par exemple, seuls les chiffres 0, 1 et 2 seront utilisés, et le nombre $(210122)_3$ représente l'entier $2 \times 3^5 + 3^4 + 3^2 + 2 \times 3 + 2$, c'est à dire 584

Bases usuelles :

- base décimale ($b=10$) : c'est la base à laquelle on est habitué depuis l'enfance
- base binaire ($b=2$)
 - utilise deux chiffres : $\{0, 1\}$
 - C'est le système de numération avec lequel fonctionnent les ordinateurs
- base octale ($b=8$) :
 - utilise huit chiffres : $\{0, 1, 2, 3, 4, 5, 6, 7\}$
 - utilisée il y a un certain temps en informatique
 - permet de coder 3 bits par un seul symbole
- base hexadécimale ($b=16$) :
 - utilise seize chiffres : $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$
 - très utilisée en micro informatique
 - permet de coder 4 bits par un seul symbole

Transcodage (ou conversion de base)

Transcodage (ou conversion de base)

Le transcodage (ou conversion de base) est l'opération qui permet de passer de la représentation d'un nombre exprimé dans une base à la représentation du même nombre mais exprimé dans une autre base.

Par la suite, on verra les conversions suivantes:

- Décimale vers Binaire, Octale et Hexadécimale
- Binaire vers Décimale, Octale et Hexadécimale

Passage de la base décimale B_{10} à une base B_N

La passage de la base décimale B_{10} à une base B_N se fait en recherchant la puissance de N immédiatement inférieure au nombre décimal à convertir, puis la puissance de N immédiatement inférieure au reste, et ainsi de suite jusqu'à un reste nul.

$$a = \sum_{j=0}^m a_j \cdot N^j \quad \text{En factorisant par } N, \text{ on obtient :}$$

$$a = N \cdot \left(\sum_{j=0}^{m-1} a_{j+1} \cdot N^j \right) + a_0(1)$$

Frame Title

Le reste de la division de a par N est égale au coefficient a_0 . Par division successive par N , on obtient les autres coefficients associés aux puissances de B_N :

$$a = N. \left(N. \left(\sum_{j=0}^{m-2} a_{j+2}.N^j \right) + a_1 \right) + a_0 \quad (2)$$

exemple: $(105)_{10} \Leftrightarrow (?)_2$

Algorithme:

Algorithm 1 Passage de la base décimale à la base 2

donnée: n : un nombre entier positif exprimé en base 10

résultat: r : une chaîne de caractères représentant le code binaire naturel du nombre n

1: $r \leftarrow ""$ initialisation

2: **tant que** $n \neq 0$ **faire**

3: **si** $n \bmod 2 == 0$ **alors**

4: $r = "0" + r$

5: **sinon**

6: $r = "1" + r$

7: **fin si**

8: $n = n / 2$

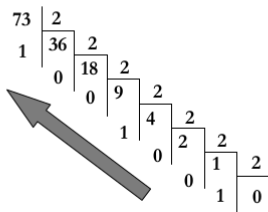
9: **fin tant que**

renvoi: r

Figure: Passage de la base décimale à la base 2

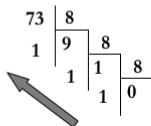
Exemple : décimale vers binaire

$$\blacksquare 73_{(10)} = 1001001_{(2)}$$



Exemple : décimale vers octale


$$\bullet 73_{(10)} = 111_{(8)}$$



Exemple : décimale vers Hexadécimale

$$\bullet 73_{(10)} = 49_{(16)}$$

73		16	
9		4	16
		4	0



Changement de base : de la base binaire vers une base b

Première solution :

Convertir le nombre en base binaire vers la base décimale puis convertir ce nombre en base 10 vers la base b .

Exemple :

$$10010_{(2)} = ?_{(8)}$$

$$10010_{(2)} = (2^4 + 2)_{(10)} = 18_{(10)} = (2 * 8^1 + 2 * 8^0)_{(10)} = 22_{(8)}$$

Changement de base : de la base binaire vers une base b

Deuxième solution :

- Binaire vers décimale : par définition $(a_n \dots a_0)_{(2)} = \sum_{i=0}^n a_i * 2^i$
- Binaire vers octale : regroupement des bit en des sous ensemble de trois bits puis remplacé chaque groupe par le symbole correspondant dans la base 8.(voir table correspondante)
- Binaire vers Hexadécimale : regroupement des bit en des sous ensemble de quatre bits puis remplacé chaque groupe par le symbole correspondant dans la base 16.(voir table correspondante)

Correspondance Octale \Binaire

Symbole Octale	suite binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Correspondance Hexadécimale \ Binaire

Hexadécimale \ Binaire

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

Exemple : binaire vers décimale

- Soit N un nombre représenté en binaire par :

$$N = 1010011101_{(2)}$$

- Représentation Décimale?

$$\begin{aligned} N &= 1.2^9 + 0.2^8 + 1.2^7 + 0.2^6 + 0.2^5 + 1.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 \\ &= 512 + 0 + 128 + 0 + 0 + 16 + 8 + 4 + 0 + 1 \\ &= 669_{(10)} \end{aligned}$$

$$1010011101_{(2)} = 669_{(10)}$$

Exemple : binaire vers octale

- Soit N un nombre représenté en base binaire par :

$$N = 1010011101_{(2)}$$

- Représentation Octale?

$$N = 001 \quad 010 \quad 011 \quad 101_{(2)}$$

$$= 1 \quad 2 \quad 3 \quad 5_{(8)}$$

$$1010011101_{(2)} = 1235_{(8)}$$

Exemple : binaire vers Hexadécimale

- Soit N un nombre représenté en base binaire par :

$$N = 1010011101_{(2)}$$

- Représentation Hexadécimale?

$$N = 0010 \quad 1001 \quad 1101_{(2)}$$

$$= \quad 2 \quad \quad 9 \quad \quad D_{(16)}$$

$$1010011101_{(2)} = 29D_{(16)}$$

94

Codage des nombres entiers

Codage des entiers positifs :

Utilisation du code binaire pour :

- L'entier naturel (positif ou nul) est représenté en base 2
- Les bits sont rangés selon leur poids, on complète à gauche par des 0.

Exemple : sur un octet, $(10)_{(10)}$ se code en binaire par $(00001010)_{(2)}$

L'étendu des nombres qu'on peut coder sur n bits est $[[0; 2^n - 1]]$


Arithmétique en base 2

- Addition binaire (8 bits)

$$\begin{array}{r}
 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0 \\
 +\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1 \\
 \hline
 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1
 \end{array}$$

- Addition binaire (8 bits) avec (débordement ou overflow) :

$$\begin{array}{r}
 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0 \\
 +\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1 \\
 \hline
 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1
 \end{array}$$


 overflow

Arithmétique en base 2

■ Multiplication binaire

$$\begin{array}{r}
 1011 \text{ (4 bits)} \\
 * 1010 \text{ (4 bits)} \\
 \hline
 0000 \\
 1011 \\
 0000 \\
 1011 \\
 \hline
 01101110
 \end{array}$$

Sur 4 bits le résultat est faux
 Sur 7 bits le résultat est juste
 Sur 8 bits on complète à gauche par un 0

Codage des entiers relatifs :

Il existe au moins trois façons pour coder :

- code binaire signé (par signe et valeur absolue)
- code complément à 1
- code complément à 2 (Utilisé sur ordinateur)

Binaire signé :

Le bit le plus significatif est utilisé pour représenter le signe du nombre :

- si le bit le plus fort = 1 alors le nombre négatif
- si le bit le plus fort = 0 alors le nombre positif

Sur les autres bits, on code la valeur absolue du nombre

Exemple : on veut coder -24 et -128 sur 8 bits

- $(-24)_{(10)} = (10011000)_{(bs)}$
- -128 hors limite \rightarrow nécessite 9 bits au minimum

Étendu de codage : $[|-(2^{n-1} - 1)|; 2^{n-1} - 1]$

complément à 1 :

Aussi appelé Complément Logique (CL) ou Complément Restreint (CR) :

- les nombres positifs sont codés de la même façon qu' en binaire pure.
- un nombre négatif est codé en inversant chaque bit de la représentation de sa valeur absolue

Le bit le plus significatif est utilisé pour représenter le signe du nombre :

- si le bit le plus fort = 1 alors le nombre négatif
- si le bit le plus fort = 0 alors le nombre positif

Exemple : on veut coder -24 sur 8 bits

- $(|-24|)_{(10)} = (00011000)_{(2)}$
- on inverse les bits $\rightarrow (-24)_{(10)} = (11100111)_{(ca1)}$

Étendu de codage : $[|-(2^{n-1} - 1)|; 2^{n-1} - 1]$

complément à 2 :

Aussi appelé Complément Vrai (CV) :

- les nombres positifs sont codés de la même façon qu' en binaire pure.
- un nombre négatif est codé en ajoutant la valeur 1 à son complément à 1

Le bit le plus significatif est utilisé pour représenter le signe du nombre :

Exemple : on veut coder -24 sur 8 bits

- $(|-24|)_{(10)} = (00011000)_{(2)}$
- on inverse les bits $\rightarrow (-24)_{(10)} = (11100111)_{(ca1)}$
- puis on ajoute 1 au complément à 1 \rightarrow
 $(-24)_{(10)} = (11101000)_{(ca2)}$

Étendu de codage : $[|-(2^{n-1})|; 2^{n-1} - 1]$

Codage des nombres réels

Les formats de représentations des nombres réels sont :

- Format virgule fixe
 - utilisé par les premières machines
 - possède une partie 'entière' et une partie 'décimale' séparés par une virgule. La position de la virgule est fixe d'où le nom.
 - Exemple : $54,25_{(10)}$; $10,001_{(2)}$; $A1,F0B_{(16)}$
- virgule flottante (utilisé actuellement sur machine)
 - défini par : $\pm m.b^e$
 - un signe + ou -
 - une mantisse m (en virgule fixe)
 - un exposant e (un entier relative)
 - une base b (2,8,10,16,...)
 - Exemple : $0,5425.10^2_{(10)}$; $10,1.2^{-1}_{(2)}$; $(A0,B4.16^{-2})_{(16)}$

virgule fixe :

- Etant donné une base b

➤ un nombre x est représenté par :

- $x = a_{n-1}a_{n-2}\dots a_1a_0,a_{-1}a_{-2}\dots a_{-p} (b)$
- a_{n-1} est le chiffre de poids fort
- a_{-p} est le chiffre de poids faible
- n est le nombre de chiffre avant la virgule
- p est le nombre de chiffre après la virgule

- La valeur de x en base 10 est : $x = \sum_{i=-p}^{n-1} a_i b^i$,

- Exemple :

$$101,01_{(2)} = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 5,25_{(10)}$$

virgule fixe :

- Le passage de la base 10 à la base 2 est défini par :
 - Partie entière est codée sur p bits (division successive par 2)
 - Partie décimale est codée sur q bits en multipliant par 2 successivement jusqu'à ce que la partie décimale soit nulle ou le nombre de bits q est atteint.
 - Exemple : $4,25_{(10)} = ?_{(2)}$ format virgule fixe
 - ✓ $4_{(10)} = 100_{(2)}$
 - ✓ $0,25 \times 2 = 0,5 \rightarrow 0$
 - ✓ $0,5 \times 2 = 1,0 \rightarrow 1$
 - ✓ donc $4,25_{(10)} = 100,01_{(2)}$

virgule flottante :

$$x = \pm M \cdot 2^E$$

où M est la mantisse (virgule fixe) et E l'exposant (signé).

Le codage en base 2, format virgule flottante, revient à coder le signe, la mantisse et l'exposant.

Exemple : Codage en base 2, format virgule flottante, de (3,25)

$$\begin{aligned} 3,25_{(10)} &= 11,01_{(2)} \quad (\text{en virgule fixe}) \\ &= 1,101 \cdot 2^1_{(2)} \\ &= 110,1 \cdot 2^{-1}_{(2)} \end{aligned}$$

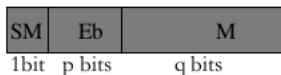
Pb : différentes manières de représenter E et M

→ Normalisation

virgule flottante (Normalisation) :

$$x = \pm 1, M \cdot 2^{Eb}$$

- Le signe est codé sur 1 bit ayant le poids fort :
 - le signe - : bit 1
 - Le signe + : bit 0
- Exposant biaisé (Eb)
 - placé avant la mantisse pour simplifier la comparaison
 - Codé sur p bits et biaisé pour être positif (ajout de $2^{p-1}-1$)
- Mantisse normalisé(M)
 - Normalisé : virgule est placé après le bit à 1 ayant le poids fort
 - M est codé sur q bits
 - Exemple : 11,01 \rightarrow 1,101 donc M =101



Standard IEEE 754 (1985) :

Simple précision sur 32 bits :

1 bit de signe de la mantisse

8 bits pour l'exposant

23 bits pour la mantisse



Double précision sur 64 bits :

1 bit de signe de la mantisse

11 bits pour l'exposant

52 bits pour la mantisse



Conversion décimale - IEEE754 (Codage d' un réel)

$$35,5_{(10)} = ?_{(\text{IEEE 754 simple précision})}$$

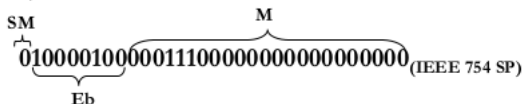
Nombre positif, donc SM = 0

$$35,5_{(10)} = 10011,1_{(2)} \quad (\text{virgule fixe})$$

$$= 1,00111 \cdot 2^5_{(2)} \quad (\text{virgule flottante})$$

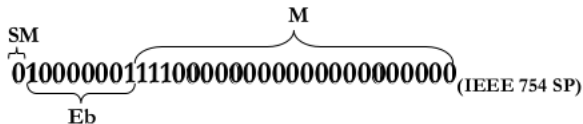
$$\text{Exposant} = E_b - 127 = 5, \text{ donc } E_b = 132$$

$$1, M = 1,00111 \text{ donc } M = 00011100...$$



Exercice: Convertir le nombre décimal 8,625 en virgule flottante suivant la norme IEEE 754

Conversion décimale - IEEE754 (Évaluation d' un réel)



$S = 0$, donc nombre positif

$Eb = 129$, donc exposant = $Eb - 127 = 2$

$1, M = 1,111$

$+ 1,111 \cdot 2^2_{(2)} = 111,1_{(2)} = 7,5_{(10)}$