CODAGE/NUMERATION

Numération-codage

PLAN

Numération-codage

PLAN

- I) Notions de codage-décodage
- II) Représentation des nombres
- III) Méthodes de conversion
- IV) Code Binaire Naturel
- V) Nombres Binaires Signés
- VI) Codage des nombres flottants
- VII) Autres codes

I) Notions de codage / décodage

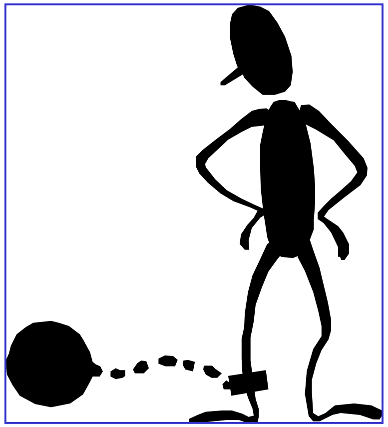
Ordinateurs : nombres en précision finie et fixe

Exemple : ensemble des entiers positifs à trois chiffres décimaux

A = {000,001,002, ..., 999} Card(**A**)=1000

Ne peut représenter :

> 999
< 0
fractionnaires
irrationnels
complexes



I) Notions de codage / décodage

$$\textit{Soient } i, j \in Z \quad \Rightarrow \quad \begin{array}{l} i + j \in Z \\ i - j \in Z \\ i * j \in Z \\ i / j \notin Z \text{ en général} \end{array}$$

$$\text{Sur A} : 600 + 600 = 1200 \notin \text{A (dépassement)}$$

$$003 - 005 = -2 \notin \text{A (dépassement)}$$

$$050 * 050 = 2500 \notin \text{A (dépassement)}$$

$$007 / 002 = 3{,}5 \notin \text{A (dépassement)}$$

Dépassement = overflow en anglais

I) Notions de codage / décodage

Conséquences sur les ordinateurs

Possibilités de résultats faux en conditions normales de fct.
(pas de panne)

Dans **A :** **a = 700, b = 400, c = 300**

a + (b - c) = (a + b) - c commutativité

700 + 100 = 800 dépassement -300 = dépassement

I) Notions de codage / décodage

Conséquences sur ordinateurs (suite)

Dans A : a = 5, b = 210, c = 195

a * (b - c) = a * b - a * c distributivité

5 * 15 = 75

Overflow - Overflow = OverFlow

Il faut connaître les méthodes de représentation pour prévoir les problèmes éventuels

Numération-codage

PLAN

- I) Notions de codage-décodage
- II) Représentation des nombres
- III) Méthodes de conversion
- IV) Code Binaire Naturel
- V) Nombres Binaires Signés
- VI) Codage des nombres flottants
- VII) Autres codes

II) Représentation des nombres

Evolution : romaine (invention du zéro), inde, arabe

Principe de numération : Juxtaposition de symboles
appelés chiffres (caillou en arabe)

Système décimal : dix symboles {0,1,2, ...,9}
Nombre de symbole = Base de numération

Ecriture d'un nombre : position du chiffre détermine son poids

NUMERATION DE POSITION

$$1578 = 1.10^3 + 5.10^2 + 7.10^1 + 8.10^0$$

(en europe : 970 Gesbert d'Aurillac devenu en 999 Sylvestre II, relayé en 1202 par Fibonnacci)

II) Représentation des nombres

Généralisation

Base b associée à b symboles $\{S_0, S_1, S_2, ..., S_{b-1}\}$

N s'écrit ($a_n\, a_{n-1}\, a_{n-2} \ldots a_0 , a_{-1} \ldots a_{-m}$) (dépend de la base)

avec a_i dans $\{S_0, S_1, S_2, ..., S_{b-1}\}$

Valeur de N = $a_n.b^n + a_{n-1}.b^{n-1} + \ldots + a_0.b^0 + a_{-1}.b^{-1} + \ldots + a_{-m}.b^{-m}$

$$= \sum_{-m}^{n} a_i.b^i$$

Forme polynomiale

La valeur est indépendante de la base

II) Représentation des nombres

Définitions

$$N = (a_n \, a_{n-1} \, a_{n-2} \ldots a_0 \, , \, a_{-1} \ldots a_{-m})_b$$

a_i chiffre de rang i (ou digit)

b^i poids associé à a_i

a_n chiffre le plus significatif (ou de poids fort) MSD

a_{-m} chiffre le moins significatif LSD

$(a_n \, a_{n-1} \, a_{n-2} \ldots a_0)$ partie entière

$(a_{-1} \ldots a_{-m})$ partie fractionnaire (<1)

II) Représentation des nombres

Systèmes utilisés

Base 2 (ou binaire) {0,1} digit = élément binaire ou eb

binary digit ou bit

la plus utilisée : MSB, LSB 00110101 = octet

1101 = quartet

Base 10 (ou décimal)

celle de l'école primaire ou de tous les jours

Base 8 (ou octal) {0,1,2,3,4,5,6,7}

Base 16 (ou héxadécimal) {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

raccourci d'écriture de la base 2

Numération-codage

PLAN

- I) Notions de codage-décodage
- II) Représentation des nombres
- III) Méthodes de conversion
- IV) Code Binaire Naturel
- V) Nombres Binaires Signés
- VI) Codage des nombres flottants
- VII) Autres codes

III) Méthodes de conversion

<u>Problème :</u> exprimer le même nombre dans des bases différentes

Sous problèmes :

de b^m vers b
de b vers b^m

de b vers 10
de 10 vers b

de i vers j

Numération-codage

PLAN

- III) Méthodes de conversion
  - III-1) Conversion : de 2^m vers 2 / 2 vers 2^m
  - III-2) Conversion de B vers 10
  - III-3) Conversion de 10 vers B
  - III-4) Conversion de i vers j
  - III-5) Représentation binaire

III-1) Conversion : de 2^m vers 2 / 2 vers 2^m

2^m vers 2 : expansion d'un digit en m bits
2 vers 2^m : regroupement de bits par paquets de m

$$\boxed{N = a_7.2^7 + a_6.2^6 + a_5.2^5 + a_4.2^4 + a_3.2^3 + a_2.2^2 + a_1.2^1 + a_0.2^0}$$

$$= (a_7.2^3 + a_6.2^2 + a_5.2^1 + a_4.2^0).2^4 + (a_3.2^3 + a_2.2^2 + a_1.2^1 + a_0.2^0)$$

$$= (a_7.2^3 + a_6.2^2 + a_5.2^1 + a_4.2^0).16^1 + (a_3.2^3 + a_2.2^2 + a_1.2^1 + a_0.2^0).16^0$$

$$\boxed{N = b_1.16^1 + b_0.16^0}$$

$$0 \le a_3.2^3 + a_2.2^2 + a_1.2^1 + a_0.2^0 \le 15$$

III-1) Conversion : de 2^m vers 2 / 2 vers 2^m

Ecriture de $(622)_8$ en base 2 et base 16 ?

6 2 2	base 8
110 010 010	base 2
1 1001 0010	base 2
1 9 2	base 16

Ecriture de $(622,663)_8$ en base 2 et base 16 ?

6 2 2 , 6 6 3	base 8
110 010 010 , 110 110 011	base 2
1 1001 0010 , 1101 1001 1	base 2
1 9 2 , D 9 8	base 16

III-2) Conversion base B vers base 10

Application de la forme polynomiale

Valeur de N = $a_n.b^n + a_{n-1}.b^{n-1} + \ldots + a_0.b^0 + a_{-1}.b^{-1} + \ldots + a_{-m}.b^{-m}$

$$= \sum_{-m}^{n} a_i.b^i$$

si B = 2 $(10001101)_2 = 2^7 + 2^3 + 2^2 + 1 = (141)_{10}$

si B = 16 $(FF)_h = 15*16 + 15 = (255)_{10}$

III-3) Conversion base 10 vers base B

Première méthode : soustraction

$(363)_{10}$ en base 2 ?

Recherche de la puissance 2 juste supérieure = $2^9 = 512$

$363 - 2^8 = 107$	1	MSB
2^7 trop grand	0	
$107 - 2^6 = 43$	1	
$43 - 2^5 = 11$	1	
2^4 trop grand	0	
$11 - 2^3 = 3$	1	
2^2 trop grand	0	
$3 - 2^1 = 1$	1	
1	1	LSB

$$(363)_{10} = (101101011)_2$$

III-3) Conversion base 10 vers base B

Cette méthode est applicable pour toute les bases

Autre exemple : $(363)_{10}$ en base 16 ?

$363 < 16^3 = 4096$

$363 = 1.16^2 + 107$	1
$107 = 6.16^1 + 11$	6
$11 = B.16^0$	B

$$\boxed{(363)_{10} = (16B)_{16}}$$

inconvénient de cette méthode : il faut connaître les puissances

III-3) Conversion base 10 vers base B

<u>deuxième méthode : division / multiplication</u>

Principe : En base 10 $xyz = xy * 10 + z$

$$
\begin{array}{r|l}
xyz & 10 \\ \cline{2-2}
z & xy \;\begin{array}{|l} 10 \\ \hline x \end{array} \\
 & \quad y
\end{array}
$$

LSD

MSD

III-3) Conversion base 10 vers base B

deuxième méthode : division / multiplication

$(362)_{10}$ en base 2 ?

362 | 2
LSB 0 | 181 | 2
1 | 90 | 2
0 | 45 | 2
1 | 22
2 | 2
0 | 1
MSB

$(101101010)_2$

= 256 + 64 + 32 + 8 + 2

$(362)_{10}$ en base 16 ?

362 | 16
LSB
10 (A) | 22 | 16
6 | 1
MSB

$(16A)_h$

$= 1*16^2 + 6*16 + 10$

III-4) Conversion base i vers base j

si $I = B^m$ et $J = B^n$

B^m vers B puis B vers B^n

sinon

I vers 10 puis 10 vers J

Exemple : $(77)_8 = (11\ 1111)_2 = (3F)_h$

III-5) Représentation binaire

Définitions :	format	nb de bit de utilisés
	convention	protocole de codage
	dynamique	différence entre le max et le min
	résolution	différence entre deux consécutifs

Exemple : format 8 bits
convention entiers positifs
dynamique 2^8
résolution 1 (constante sur la dynamique)

$(255)_{10} = (1111\ 1111)_2$ $(1)_{10} = (0000\ 0001)_2$

→ Code Binaire Naturel

Numération-codage

PLAN

- I) Notions de codage-décodage
- II) Représentation des nombres
- III) Méthodes de conversion
- IV) Code Binaire Naturel
- V) Nombres Binaires Signés
- VI) Codage des nombres flottants
- VII) Autres codes

IV) Code Binaire Naturel (CBN)

- CBN => codage des entiers positifs

- Avec N bits, on code les entiers positifs de 0 à $2^N - 1$

Exemple : avec N=8, on code de 0 à $2^8-1=255$

- On démontre

$$\forall n \geq 1 \quad 2^n - 1 = \sum_{i=0}^{n-1} 2^i$$

LSB=1 => impair

- LSB => Bit de parité (à démontrer)

LSB=0 => pair

- Opérations en base 2 avec un CBN (+, -, *, /)

Numération-codage

PLAN

- IV) Code Binaire Naturel
  - IV-1) Addition
  - IV-2) Soustraction
  - IV-3) Multiplication
  - IV-4) Division

IV-1) Addition

En CBN, le principe est le même qu'en base 10

Retenue (Carry)

Base 10

	1	1	
	1	4	5
+	2	5	8
	4	0	3

Base 2

1	1	1	
	1	1	1
+		1	1
1	0	1	0

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

IV-2) Soustraction

En CBN, le principe est le même qu'en base 10

		1	1
	1	**2**	**0**
-		**7**	**2**
	1	1	
		4	**8**

Base 10

Retenue
(Carry)

		1	1
	1	**0**	**0**	**1**
-		**1**	**1**	**0**
	1	1		
			1	**1**

Base 2

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1
et 1 de report

IV-3) Multiplication

En CBN, le principe est le même qu'en base 10

Base 10

```
     3  5
x    1  2
---------
     7  0
 + 3 5
---------
  4  2  0
```

Base 2

Retenue (Carry)

```
        1  0  1
x       1  0  1
---------------
     1
        1  0  1
     0  0  0  .
  1  0  1
---------------
  1  1  0  0  1
```

0 * 0 = 0
1 * 0 = 0
0 * 1 = 0
1 * 1 = 1

IV-4) Division

En CBN, le principe est le même qu'en base 10

```
  1 5 6 | 1 3
- 1 3   |-----
-----   | 1 2
    2 6 |
  - 2 6 |
  -----
      0
```

Base 10

```
  1 0 0 1 0 | 1 1
-   0       |---------
-------     | 0 1 1 0
  1 0 0     |
-   1 1     |
  -------   |
      1 1   |
    - 1 1   |
    -----   |
        0 0 |
      -   0 |
      ----- |
          0 |
```

Base 2

Reste

Numération-codage

PLAN

- I) Notions de codage-décodage
- II) Représentation des nombres
- III) Méthodes de conversion
- IV) Code Binaire Naturel
- V) Nombres Binaires Signés
- VI) Codage des nombres flottants
- VII) Autres codes

V) Nombres Binaires Signés

- CBN => codage des entiers positifs

- Il est nécessaire de pouvoir coder les entiers relatifs

Exemple : une commande -5V +5V

- Il existe différents solutions (avantages et inconvénients)

  * Bit de Signe
  * Complément à 1
  * Complément à 2

Numération-codage

PLAN

- V) Nombre Binaires Signés
  - V-1) Bit de signe
  - V-2) Complément à 1
  - V-3) Complément à 2

V-1) Bit de signe

Sur n bits on garde 1 bit pour indiquer le signe

S	Msb xxxxxx Lsb
Signe	Module (positif)
1 bit	n-1 bits

Convention :

S=0 pour positif

S=1 pour négatif

Dynamique : $-(2^{n-1}-1)$ à $(2^{n-1}-1)$

Exemple sur 8 bits : $-22 = (1\ 0010110)_{2,S+M}$

V-1) Bit de signe

Avantages :

- Multiplications faciles

N1*N2

Abs(N1)*Abs(N2)

S = S1 xor S2

Inconvénients :

- Deux représentations du zéro

Sur 4 bits +0 = 0000, -0 = 1000

- Additions moins simples

V-2) Complément à 1

- Complément réduit

Notation : $CR(X) = \overline{X}$

BASE

- Définition :Complément chiffre à chiffre
  $(xyz)_b$ donne $(x'y'z')_b$ tel que $x+x'=y+y'=z+z'= b-1$

En binaire 00110 donne 11001
et 00110 + 11001 = 11111

Inconvénients :

- le chiffre 0 est codé 2 fois
- les additions binaires ne sont pas directes

V-2) Complément à 1

- Soit $(X)_b$ codé sur n digits ⇒ On a $X + CR(X) = b^n - 1$

Dans le format considéré $2^n = 0$

Partie interprétée

sur 4 bits : $2^4 = 1\boxed{0000} = 0$

d'où : $CR(X) + 1 = -X$

⇒ Complément à 2

V-3) Complément à 2 (complément vrai)

<u>Définition :</u> N^* complément vrai de N sur n chiffres en base B

$$N^* = B^n - N = -N \qquad \text{(rappel : } B^n = 0 \text{ sur n chiffres)}$$

Calcul de l'opposé (sur n bits) :

$$N^* = (-N) = 2^n - N = [2^n - 1] - N + 1$$

$$= [N + CR(N)] - N + 1 = CR(N) + 1$$

$$\Rightarrow \qquad \boxed{N^* = CR(N) + 1 = CV(N)}$$

V-3) Complément à 2 (complément vrai)

Sur 4 bits :

		-8	1000
7	0111	-7	1001
6	0110	-6	1010
...			
0	0000	-0	0000

Remarques : le bit de poids fort = signe (0:positif, 1:négatif)
poids du bit de signe = $-(2^{n-1})$
0 n'a qu'une représentation

exemple: sur 8 bits signés: $(11110000)_2 = -128 + 64 + 32 + 16$
$= (-16)_{10}$

$C1 + 1 = (00001111)_2 + 1 = (16)_{10}$

V-3) Complément à 2 (complément vrai)

Dynamique sur n bits : $-(2^{n-1})$ à $(2^{n-1}-1)$

Avantages : additions directes en binaire sans tenir compte de la retenue

Exemple 1: 4 + (-3)

	0	**1**	**0**	**0**
+	**1**	**1**	**0**	**1**
1	**0**	**0**	**0**	**1**

Exemple 2: 4 + (-5)

	0	**1**	**0**	**0**
+	**1**	**0**	**1**	**1**
0	**1**	**1**	**1**	**1**

V-3) Complément à 2

Nombres signés : comparaison (sur 4 bits)

N_{10}	N_2	$-N_{S+M}$	$-N_{2,CR}$	$-N_{2*}$
0	0000	1000	1111	0000
1	0001	1001	1110	1111
2	0010	1010	1101	1110
3	0011	1011	1100	1101
4	0100	1100	1011	1100
5	0101	1101	1010	1011
6	0110	1110	1001	1010
7	0111	1111	1000	1001
8	1000	....	....	(1000)

V-3) Complément à 2

Propriétés

- pas de gestion de retenue intermédiaire
- détection simple d'overflow (dépassement de capacité)

sur 4 bits : (-8 à +7)

3		0011
+ 6		0110
= 9	=	1001
(-7) !		↑
		hors format

$X + (-Y) = N$ avec $X > N > -Y$

4 + 5 = 9 (of) 0100 + 0101 = 1001

-4 - 5 = -9 (of) 1100 + 1011 = 0111

Indicateur d'overflow :
(dans les microprocesseurs)
$OF = S_a \text{ xor } S_b \text{ xor } S_r \text{ xor } Carry_r$

V-3) Complément à 2

exemples

- Sur 8 bits signés : $(127)_{10}+1$ et $(127)_{10}-1$ et $(-128)_{10}$ -1

```
   1 1 11  1 1 1
   0111 1111
+  0000 0001
  ------------
 0 1000 0000
```

Overflow=1

$(-128)_{10}$

```
   1 1 11  1 1 1
   0111 1111
+  1111 1111
  ------------
 1 0111 1110
```

Overflow=0

$(64)_{10} + (32)_{10} + (16)_{10} + (8)_{10} + (4)_{10} + (2)_{10} = (126)_{10}$

```
   1000 0000
+  1111 1111
  ------------
 1 0111 1111
```

Overflow=1

$(127)_{10}$

V-3) Complément à 2

Exemple (suite et fin)

- Sur 8 bits signés : $(-127)_{10}-1$

```
    1 1 1 1 1 1 1
     1000 0001
 +
     1111 1111
   -----------
   1 1000 0000
```

Overflow=0

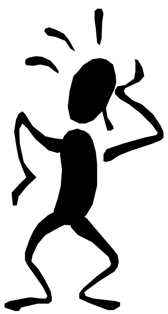
$(-128)_{10}$

Numération-codage

PLAN

VI) Codage des nombres flottants

Dans un calculateur : nombre sous format déterminé (entier, virgule fixe, virgule flottante ...)



TOUT EST QUESTION DE CONVENTION

A quoi est associé 1101100011100110 ?

nombre entier 2*, Caractère ASCII, pixel d'une image, nombre fractionnaire ... ?

Virgule flottante

Virgule fixe

Numération-codage

PLAN

- VI) Codage des Nombres Flottants
  - VI-1) Virgule fixe
  - VI-2) Virgule flottante

VI-1) Virgule fixe

Par <u>convention</u> on place la virgule quelque part et <u>on interprète</u>

2^{n-1} 2^{0} avant de placer la virgule

MSB xxxxxx , xxxx LSB

2^{n-1-k} $2^{0},2^{-1}$ 2^{-k} avec la virgule au rang k

Dynamique : 2^{n-1-k}

Résolution : 2^{-k} # 0

VI-1) Virgule fixe

avantage Bon format pour l'addition :

```
   12,32
+  23,45
=  35,77  La virgule reste fixe
```

inconvénient Mauvais format pour la multiplication :

```
   13,4
*  10,1
= 135,34  dépassement à gauche (overflow)
          dépassement à droite (arrondi)
```

VI-1) Virgule fixe

solution

Problèmes réglés si les nombres sont inférieurs à 1 :

$$\begin{array}{r} 0,87 \\ *\ \underline{0,74} \\ =0,64_{38} \end{array}$$

- On place la virgule toujours à gauche
- On utilise un autre groupe de bit pour connaître la position de la virgule

$\Longrightarrow$ **Format virgule flottante**

VI-2) Virgule flottante

$N = M.b^E$

M = mantisse en 2* de forme 0,xxx
b = base de l'exponentiation (2)
E = exposant en binaire signé (C2)

On stocke la chaîne de bit **ME** dans le calculateur

Exemple : codage de PI sur 5 chiffres de mantisse
et 2 chiffres d'exposant (en décimal)

$0{,}3141.10^1 = 0{,}0003.10^4$!!! PI*10000=3

On dit qu'un flottant est normalisé quand le premier chiffre significatif est juste derrière la virgule (précision maximum)

VI-2) Virgule flottante

Calcul et stockage

Multiplication : $M_1.b^{E_1} * M_2.b^{E_2} = M_1.M_2.b^{(E_1+E_2)}$

dénormalisation du plus petit nombre (vers la droite)

Addition : $M_1.b^{E_1} + M_2.b^{E_2} = M_1.b^{(E_1-E_2)}.b^{E_2} + M_2.b^{E_2}$

$= (M_1.b^{(E_1-E_2)} + M_2).b^{E_2}$

Si $E_2 > E_1$

puis renormalisation

il faut comparer facilement

$\Longrightarrow$ Exposant codé en binaire décalé

S_M	EXPOSANT	MANTISSE

VI-2) Virgule flottante

<u>Une possibilité parmi d'autres</u>

1) Codage du nombre positif (partie entière puis partie décimale)
2) Fusion des 2 parties et décalage de la virgule devant le 1er chiffre significatif
3) Codage de la Mantisse et de l'Exposant (complément à 2)
4) Codage en C2 de la Mantisse si le nombre initial est négatif

exemple : $(-6.625)_{10}$ sur 12 bits dont 4 pour l'exposant

Nombre positif :6.625
Partie entière 6 -> 110
Partie décimale 0.625 ->0.101 (multiplications successives)
Fusion et décalage -> 110.101 = 0.110101 * 2^3
Codage -> 0/1101010 0/011
Traitement de la négation -> 1/0010110 0/011

VI-2) Virgule flottante

<u>**Norme internationale :**</u> IEEE 754 flottant sur 32 bits

b_{31} .... b_0
signe mantisse, exposant, mantisse
1 bit 8 bit 23 bits

Le bit de signe est 1 pour négatif et 0 pour positif
La mantisse vaut toujours 1,xxxx et on ne stocke que xxxx
L'exposant est en excédent 127
La valeur 0 correspond à des 0 partout (en fait $1{,}0.2^{-127}$)

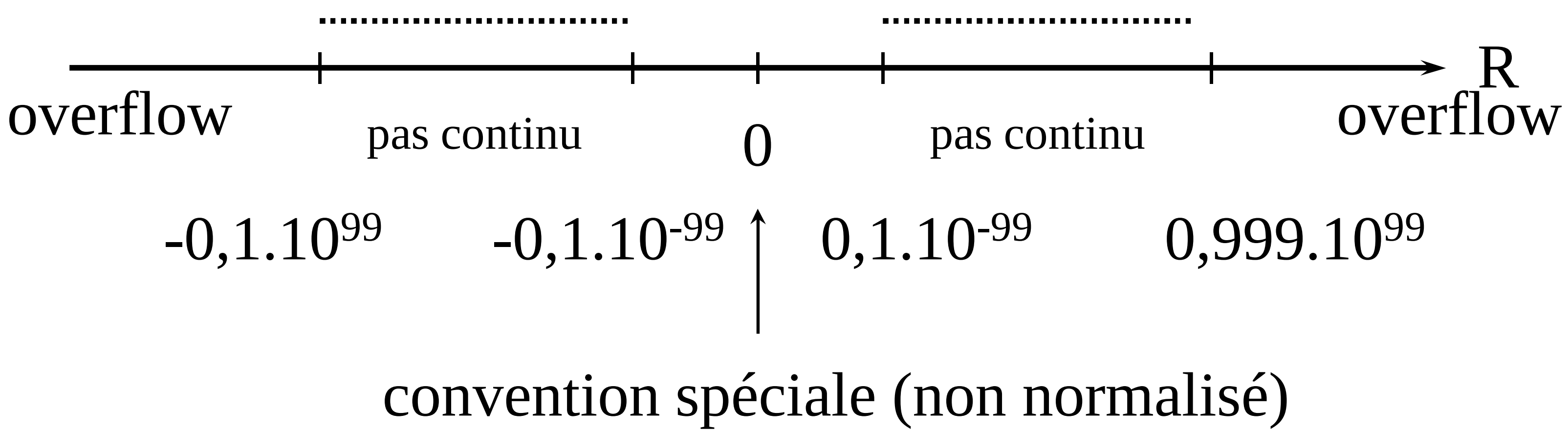
exemple : 1 10000011 11000000000000000000000 = $-1{,}75.2^{4}$ = -28
0 01111111 00000000000000000000000 = $1{,}0.2^{0}$ = 1

VI-2) Virgule flottante

Attention aux résultats !

Exemple : (en base 10) Mantisse 3 chiffres $0,999 \geqslant |M| \geqslant 0,1$

Exposant 2 chiffres -99 à 99



ON NE MANIPULE JAMAIS L'ENSEMBLE DES REELS

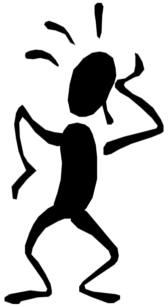
Numération-codage

PLAN

- I) Notions de codage-décodage
- II) Représentation des nombres
- III) Méthodes de conversion
- IV) Code Binaire Naturel
- V) Nombres Binaires Signés
- VI) Codage des nombres flottants
- VII) Autres codes

VII) Autres codes

Dans un calculateur : nombre sous format déterminé (entier, virgule fixe, virgule flottante ...)



TOUT EST QUESTION DE CONVENTION

A quoi est associé 1101100011100110 ?

nombre entier, négatif, Caractère ASCII, pixel d’une image, nombre fractionnaire … ?

- Code Binaire Réfléchi
- Code BCD
- Code ASCII

Numération-codage

PLAN

- VII) Autres codes
  - VII-1) Code Binaire Réfléchi
  - VII-2) Code BCD
  - VII-3) Code ASCII

VII-1) Code Binaire Réfléchi (code GRAY)

Un seul bit de modifié à la fois

(01,10)

(0011,1100)

(00001111, ...)

...

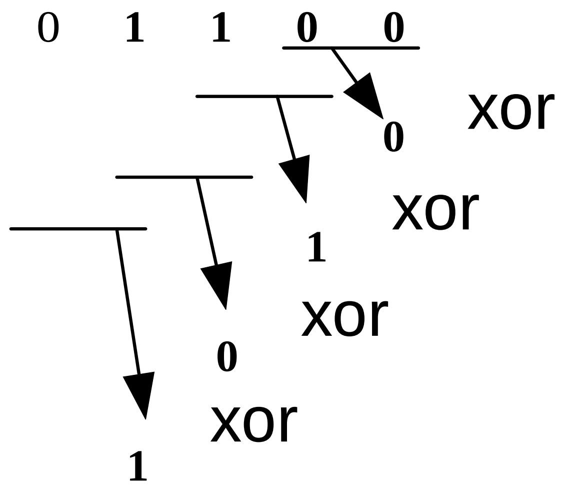
<u>**Intérêt :**</u> simplification d'équations logiques

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

VII-1) Code Binaire Réfléchi (code GRAY)

Base 10	Base 2 $(b_3b_2b_1b_0)$	Gray $(r_3r_2r_1r_0)$
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Autre technique



Nr = Nb xor (Nb/2)

(on le démontrera)

VII-2) Code BCD

→ Binary Coded Decimal

Les chiffres de 0 à 9 sont codés sur 4 bits (quartet)

Exemple : $(127)_{10} = (\ 0001\ 0010\ 0111)_{BCD}$

avantages

Afficheurs, facilité de conversion en ASCII

inconvénient

Opérations plus complexes (rajouter 6 pour le +)

Exemple : $(9 + 6 = 15)_{10}$ mais l'utilisation de l'addition binaire donne $(1001 + 0110 = 1111)_2$

$(1111 + 0110 = 0001\ 0101)_2$ ce qui est bien égal au nombre BCD recherché.

VII-3) Code ASCII

Code ASCII 7 bits : → Codage des caractères alpha-numériques

Car	Dec	Hex	Car	Dec	Hex
Nul	0	0	SP	32	20
SOH	1	1	!	33	21
STX	2	2	"	34	22
ETX	3	3	#	35	23
EOT	4	4	$	36	24
ENQ	5	5	%	37	25
ACK	6	6	&	38	26
BEL	7	7	'	39	27
BS	8	8	(	40	28
HT	9	9	)	41	29
LF	10	A	*	42	2A
VT	11	B	+	43	2B
FF	12	C	,	44	2C
CR	13	D	-	45	2D
SO	14	E	.	46	2E
SI	15	F	/	47	2F
DLE	16	10	0	48	30
DC1	17	11	1	49	31
DC2	18	12	2	50	32
DC3	19	13	3	51	33
DC4	20	14	4	52	34
NAK	21	15	5	53	35
SYN	22	16	6	54	36
ETB	23	17	7	55	37
CAN	24	18	8	56	38
EM	25	19	9	57	39
SUB	26	1A	:	58	3A
ESC	27	1B	;	59	3B
FS	28	1C	<	60	3C
GS	29	1D	=	61	3D
RS	30	1E	>	62	3E
US	31	1F	?	63	3F

Car	Dec	Hex	Car	Dec	Hex
@	64	40	`	96	60
A	65	41	a	97	61
B	66	42	b	98	62
C	67	43	c	99	63
D	68	44	d	100	64
E	69	45	e	101	65
F	70	46	f	102	66
G	71	47	g	103	67
H	72	48	h	104	68
I	73	49	i	105	69
J	74	4A	j	106	6A
K	75	4B	k	107	6B
L	76	4C	l	108	6C
M	77	4D	m	109	6D
N	78	4E	n	110	6E
O	79	4F	o	111	6F
P	80	50	p	112	70
Q	81	51	q	113	71
R	82	52	r	114	72
S	83	53	s	115	73
T	84	54	t	116	74
U	85	55	u	117	75
V	86	56	v	118	76
W	87	57	w	119	77
X	88	58	x	120	78
Y	89	59	y	121	79
Z	90	5A	z	122	7A
[	91	5B	{	123	7B
\	92	5C	|	124	7C
]	93	5D	}	125	7D
^	94	5E	~	126	7E
_	95	5F	Del	127	7F

VII-3) Code ASCII

<u>Caractéristiques particulières :</u>

De 00H à 1FH : codes de contrôle et de formatage.
Exemples :

ACK :	acknowledge
BS :	Backspace
HT :	Horizontal tabulation
LF :	Line Feed
VT :	Vertical Tabulation
FF :	Form Feed
CR :	Carriage Return
ESC :	Escape

De 30H à 39H : codage des chiffres : La conversion d'un quartet BCD (valeur de 0 à 9) en code ASCII est directe et se réalise par addition de la valeur $(30)_{16}$.

L'ensemble des lettres majuscules commence au code $(41)_{16}$ pour la lettre A et suit ensuite l'ordre alphabétique.

Pour les minuscules, la lettre 'a' commence au code $(61)_{16}$.
La transformation des lettres majuscules en minuscules se fait directement en mettant à 1 le bit 5, respectivement à 0 pour la conversion inverse.

FIN