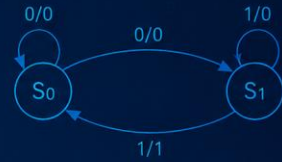


II. Numération et Codage

Représentation des grandeurs au cœur des architectures


ENSGTI
 ÉCOLE D'INGÉNIEURS

Arnaud Martin

1

1


 II. Numération
 et Codage

1. Numération, bases classiques

Les bases utilisées en électronique numérique sont:

- base 2 (binaire)
- base 8 (octal)
- base 10 (décimal)
- base 16 (hexadécimal)



Remarques:

- D'un point de vue physique (électrique), les signaux sont toujours représentés par des niveaux hauts ou bas (2 états). Les autres bases permettent de **compacter l'écriture** (base 8 et 16) pour faciliter la lecture et l'écriture des valeurs et ainsi limiter les erreurs.
 $(1101011100110111)_2 \rightarrow (D737)_{16}$

- La base 10 facilite l'interfaçage homme-machine (affichage-saisie).

2

2



Remarques:

- Toute base s'écrit **10** dans son propre système de numération
- Un décalage à gauche multiplie un nombre par sa base
- Un décalage à droite divise un nombre par sa base



Question : Combien faut-il de chiffres n dans une base B pour représenter un nombre décimal de N chiffres ?

Application numérique: base 2 et $N=2$.

3

3

On attribue le poids de chaque chiffre.

Soit un nombre $N = (a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots)_B$

Alors la valeur en base 10 correspondant est :

$$(N)_{10} = a_n B^n + a_{n-1} B^{n-1} + \dots + a_1 B^1 + a_0 B^0 + a_{-1} B^{-1} + a_{-2} B^{-2} + \dots$$

Exemples :

4

4

2. Conversions, passage d'une base à l'autre

b. De la base décimale vers une base quelconque

On traite différemment les parties entières et fractionnaires.

Soit un nombre N et sa représentation en base B sous la forme :

$$(a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots)_B$$

$E[N]$ $Frac[N]$

Divisions successives par la base

La partie entière, $E[N]$, peut se mettre sous la forme :

$$E[N] = a_n B^n + a_{n-1} B^{n-1} + \dots + a_2 B^2 + a_1 B^1 + a_0 B^0$$

en factorisant par la base B, on obtient :

$$E[N] = B(a_n B^{n-1} + a_{n-1} B^{n-2} + \dots + a_2 B^1 + a_1 B^0) + a_0$$

en répétant l'opération :

$$E[N] = B(B(a_n B^{n-2} + a_{n-1} B^{n-3} + \dots + a_2 B^0) + a_1) + a_0$$

1^{ère} division donne a_0
2^{ème} division donne a_1
3^{ème} division donne a_2
...

On fait ainsi apparaître que les a_k peuvent être obtenus par divisions successives de $E[N]$ par B.

5

5

2. Conversions, passage d'une base à l'autre

b. De la base décimale vers une base quelconque (cont.)

La partie fractionnaire, $Frac[N]$, peut se mettre sous la forme :

$$Frac[N] = a_{-1} B^{-1} + a_{-2} B^{-2} + \dots + a_{-m} B^{-m}$$

En multipliant par la base, on obtient :

$$B.Frac[N] = a_{-1} + a_{-2} B^{-1} + \dots + a_{-m} B^{-m+1}$$

En répétant l'opération :

$$B^2.Frac[N] = B a_{-1} + (a_{-2} + \dots + a_{-m} B^{-m+2})$$

Multiplications successives par la base

1^{ère} multiplication donne a_{-1}
2^{ème} division donne a_{-2}
3^{ème} division donne a_{-3}
...

On fait ainsi apparaître que les a_k peuvent être obtenus par multiplications successives par B.

6

6

2. Conversions, passage d'une base à l'autre

c. Conversion entre base 2^k -aire

- de la base 2 vers base 2^k avec $k > 1$
regroupement des bits par k
base 2 vers base 8 : regroupement par 3 des bits
base 2 vers base 16 : regroupement par 4 des bits
- des bases 2^k avec $k > 1$ vers la base 2
remplacement des chiffres de la base 2^k par
leurs expressions dans la base 2
- d'une base 2^k ($k > 1$) vers une autre base 2^k ($k > 1$)
le plus simple est de passer par l'intermédiaire de la
base 2 (développement puis regroupement)

exemples

Base 2 : 010 011 100 011
Base 8 : 2 3 4 3

Base 2 : 0100 1110 0011
Base 16 : 4 E 3

7

7

3. Codages à partir des deux chiffres 0 et 1

1010



D'un point de vue de l'électronique numérique, les données réellement manipulées sont des **niveaux hauts ou bas**, c'est-à-dire des 1 ou des 0 binaires.

À partir de ces deux chiffres, il existe plusieurs possibilités pour le codage des grandeurs ou signaux. Ces possibilités sont choisies en fonction de leur adéquation à une fonction (affichage, capteur de position), de leur précision ou dynamique de représentation ou bien de leur simplicité d'implémentation (réduction de l'électronique associée).

Par la suite, on distinguera les codes pondérés des codes non pondérés.

Définitions :

Un code est **pondéré** si chaque chiffre est affecté d'un poids.

Un code est **continu** si les mots binaires consécutifs sont adjacents, c'est-à-dire qu'ils ne diffèrent que d'un bit.

Un code est **cyclique** s'il est continu et si le premier et les derniers mots sont adjacents.

8

8

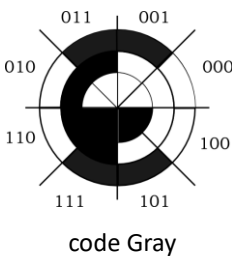
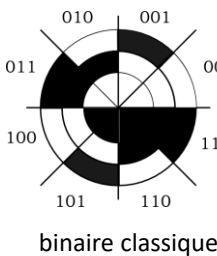
3. Codages à partir des deux chiffres 0 et 1

a. Codes classiques couramment utilisés

Code Gray

Le code Gray ou binaire réfléchi est cyclique par construction.

Valeur	code Gray
0	0 0 0
1	0 0 1
2	0 1 1
3	0 1 0
4	1 1 0
5	1 1 1
6	1 0 1
7	1 0 0

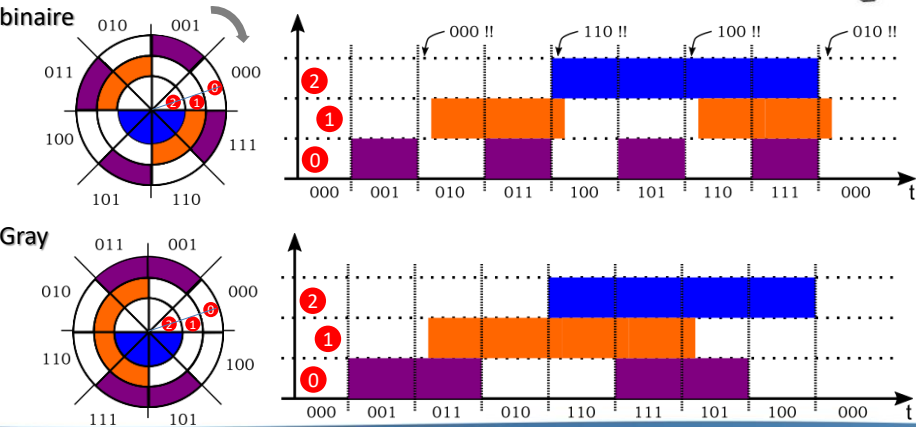


Utilisé dans les codeurs de position optique
(une erreur de lecture -bit flou- ne conduit à une incertitude que sur deux valeurs successives)
Utilisé dans certaines machines d'état pour réduire la taille des blocs logiques

3. Codages à partir des deux chiffres 0 et 1

a. Codes classiques couramment utilisés (cont.)

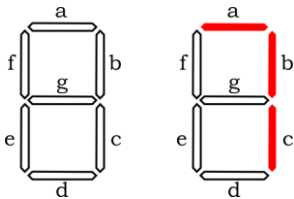
Application code Gray



Code 7 segments

Codage utilisé pour les afficheurs 7 segments

chiffre à afficher	code 7 segments						
	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	.	.	.	.	.	.	.
3	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.
6	.	.	.	.	.	.	.
7	1	1	1	0	0	0	0
8	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.



Code BCD (Binary Coded Decimal)

Aussi appelé code DCB (Décimal codé binaire)
Dans ce codage, chaque chiffre d'un nombre décimal est codé individuellement en binaire.

Exemple : 2 3 1



Remarque : on ne peut pas faire de calcul facilement avec ce code.

3. Codages à partir des deux chiffres 0 et 1

b. Autres codes

N ₁₀	Codes DCB pondérés								Codes DCB non pondérés			
	DCBN 8421	biquinaire 50 86420	quibinaire 10 43210	7421	MBQ 5 421	5211	Aiken 2421	753 -6	XS - 3	décimal Johnson	Gray DCB	2 parmi 5
0	0000	01 00001	01 00001	0000	0 000	0000	0000	000 0	0011	00000	0000	11000
1	0001	10 00001	01 00010	0001	0 001	0001	0001	100 1	0100	00001	0010	00011
2	0010	01 00010	01 00100	0010	0 010	0100	0010	011 1	0101	00011	0110	00101
3	0011	10 00010	01 01000	0011	0 011	0101	0011	001 0	0110	00111	0111	00110
4	0100	01 00100	01 10000	0100	0 100	0111	1010	101 1	0111	01111	0101	01001
5	0101	10 00100	10 00001	0101	1 000	1000	0101	010 0	1000	11111	1101	01010
6	0110	01 01000	10 00010	0110	1 001	1010	1100	110 1	1001	11110	1111	01100
7	0111	10 01000	10 00100	1000	1 010	1011	1101	100 0	1010	11100	1110	10001
8	1000	01 10000	10 01000	1001	1 011	1110	1110	011 0	1011	11000	1010	10010
9	1001	10 10000	10 10000	1010	1 100	1111	1111	111 1	1100	10000	1000	10100
auto complémentaires						oui	oui	oui	oui			
cyclique										oui	oui	
détection d'erreur		oui	oui									oui

13

13

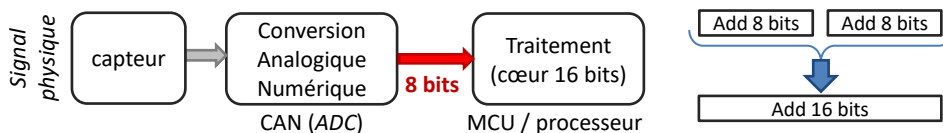
3. Codages à partir des deux chiffres 0 et 1

c. Codage pour le calcul

Problématique du codage pour le calcul

Caractéristiques requises de la représentation des nombres pour le calcul

- **bijectif** (chaque valeur n'a qu'une seule représentation)
- **extensible** (CAN 8 bits -> processeur 16 bits par instruction élémentaire)
- **simplicité des opérateurs de calculs (ALU)**
- **opérateurs cascadables** (2 additionneurs 8 bits pour former 1 additionneur 16 bits)
- **même représentation des nombres positifs en signé ou non**



14

14

binaire naturel (base 2)

sur N bits

$$x = b_{N-1} 2^{N-1} + b_{N-2} 2^{N-2} + \dots + b_2 2^2 + b_1 2^1 + b_0 2^0$$

valeurs

nombre entre [0 ;]

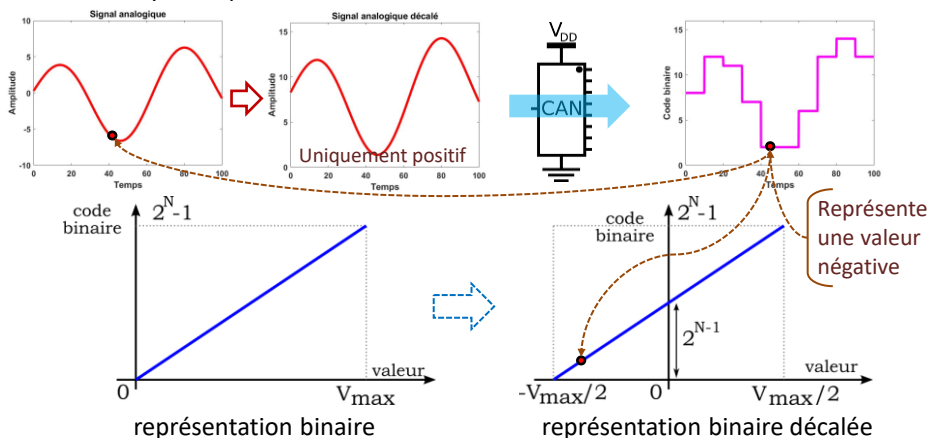
Propriétés : bijectif, extensible, opérateurs cascadables

15

15

binaire décalé

Ce type de codage vient des convertisseurs analogique/numérique (CAN), à cause de l'alimentation dissymétrique des CANs.



16

16



c'est LE codage utilisé
dans les circuits et
architectures logiques
pour le calcul avec des
nombres signés

Complément à 2

sur N bits

$$x = -b_{N-1} 2^{N-1} + b_{N-2} 2^{N-2} + \dots + b_2 2^2 + b_1 2^1 + b_0 2^0$$

valeurs
nombre entre [;]



Exemples :

Propriétés : code bijectif, extensible, même représentation des nombres positifs en signé ou non, conduit à des structures d'opérateurs simples et cascadables.



Règle d'extension (pour passer d'une représentation sur N bits à une représentation sur N+1 bits) :
simplement par **recopie du bit de poids fort**.

nombre positif commence par : nombre négatif commence par :

17

17

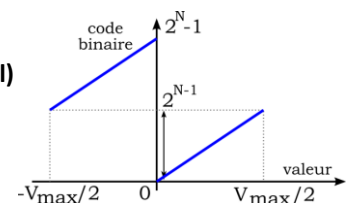


Remarque : Tant que les bits de poids forts successifs sont identiques, on peut les supprimer pour ne garder que le dernier.

Pour obtenir l'expression de la valeur négative d'un nombre, on procède en trois étapes comme suit :



- 1- expression de la valeur absolue du nombre (binaire naturel)
- 2- complément à 1 de chacun des bits
- 3- ajout d'un bit à '1' tout à droite



18

18

Exemples (1/2)

19

19

Exemples (2/2) -1563

1- expression de la valeur absolue du nombre (binaire naturel)

1563

$$\begin{array}{r|l} 1563 & 16 \\ 11 & 97 \quad 16 \\ & 1 \quad 6 \end{array}$$

1563



61B



0110 0001 1011

2- complément à 1 de chacun des bits

0110 0001 1011

1001 1110 0100

3- ajout d'un bit à '1' tout à droite

1001 1110 0100

+1

1001 1110 0101

20

20

Démonstration de cette construction en trois étapes

21

21

Data Sheet

AD5724/AD5734/AD5754

Ideal Output Voltage to Input Code Relationship—AD5754

Table 7. Bipolar Output, Offset Binary Coding

Digital Input				Analog Output		
MSB	LSB			±5 V Output Range	±10 V Output Range	±10.8 V Output Range
1111	1111	1111	1111	+2 × REFIN × (32,767/32,768)	+4 × REFIN × (32,767/32,768)	+4.32 × REFIN × (32,767/32,768)
1111	1111	1111	1110	+2 × REFIN × (32,766/32,768)	+4 × REFIN × (32,766/32,768)	+4.32 × REFIN × (32,766/32,768)
...	...	...	...	...	...	...
1000	0000	0000	0001	+2 × REFIN × (1/32,768)	+4 × REFIN × (1/32,768)	+4.32 × REFIN × (1/32,768)
1000	0000	0000	0000	0 V	0 V	0 V
0111	1111	1111	1111	−2 × REFIN × (1/32,768)	−4 × REFIN × (1/32,768)	−4.32 × REFIN × (32,766/32,768)
...	...	...	...	...	...	...
0000	0000	0000	0001	−2 × REFIN × (32,767/32,768)	−4 × REFIN × (32,767/32,768)	−4.32 × REFIN × (32,767/32,768)
0000	0000	0000	0000	−2 × REFIN × (32,768/32,768)	−4 × REFIN × (32,768/32,768)	−4.32 × REFIN × (32,768/32,768)

Résumé représentation
entiers non-signés/signés

Exemple d'un composant convertisseur
numérique analogique
(CNA, DAC Digital Analog Converter)

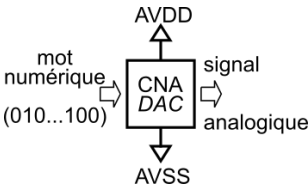


Table 8. Bipolar Output, Twos Complement Coding

Digital Input				Analog Output		
MSB	LSB			±5 V Output Range	±10 V Output Range	±10.8 V Output Range
1111	1111	1111	1111	+2 × REFIN × (32,767/32,768)	+4 × REFIN × (32,767/32,768)	+4.32 × REFIN × (32,767/32,768)
0111	1111	1111	1110	+2 × REFIN × (32,766/32,768)	+4 × REFIN × (32,766/32,768)	+4.32 × REFIN × (32,766/32,768)
...	...	...	...	...	...	...
0000	0000	0000	0001	+2 × REFIN × (1/32,768)	+4 × REFIN × (1/32,768)	+4.32 × REFIN × (1/32,768)
0000	0000	0000	0000	0 V	0 V	0 V
1111	1111	1111	1111	−2 × REFIN × (1/32,768)	−4 × REFIN × (1/32,768)	−4.32 × REFIN × (1/32,768)
...	...	...	...	...	...	...
1000	0000	0000	0001	−2 × REFIN × (32,767/32,768)	−4 × REFIN × (32,767/32,768)	−4.32 × REFIN × (32,767/32,768)
1000	0000	0000	0000	−2 × REFIN × (32,768/32,768)	−4 × REFIN × (32,768/32,768)	−4.32 × REFIN × (32,768/32,768)

Table 9. Unipolar Output, Straight Binary Coding

Digital Input				Analog Output		
MSB	LSB			±5 V Output Range	±10 V Output Range	±10.8 V Output Range
1111	1111	1111	1111	+2 × REFIN × (65,535/65,536)	+4 × REFIN × (65,535/65,536)	+4.32 × REFIN × (65,535/65,536)
1111	1111	1111	1110	+2 × REFIN × (65,534/65,536)	+4 × REFIN × (65,534/65,536)	+4.32 × REFIN × (65,534/65,536)
...	...	...	...	...	...	...
1000	0000	0000	0001	+2 × REFIN × (32,769/65,536)	+4 × REFIN × (32,769/65,536)	+4.32 × REFIN × (32,769/65,536)
1000	0000	0000	0000	+2 × REFIN × (32,768/65,536)	+4 × REFIN × (32,768/65,536)	+4.32 × REFIN × (32,768/65,536)
0111	1111	1111	1111	+2 × REFIN × (32,767/65,536)	+4 × REFIN × (32,767/65,536)	+4.32 × REFIN × (32,767/65,536)
...	...	...	...	...	...	...
0000	0000	0000	0001	+2 × REFIN × (1/65,536)	+4 × REFIN × (1/65,536)	+4.32 × REFIN × (1/65,536)
0000	0000	0000	0000	0 V	0 V	0 V

22



Remarques : décalages arithmétiques et logiques

- Un décalage à droite correspond à une division par 2
exemple :
- Un décalage à gauche correspond à une multiplication par 2



exemple:

Attention au débordement



- Différence entre LSR (Logic Shift Right) décalage à droite logique (introduction d'un 0) et ASR (Arithmetic Shift Right) (introduction du bit de signe)
- Décalages spéciaux sur certains processeurs/microcontrôleurs
par exemple PIC18 : décalage circulaire (RRNCF (rotate right f))

23

23

Représentation à virgule fixe en $Q_{m,k}$

$$b_{N-1} b_{N-2} \dots b_k, b_{k-1} \dots b_2 b_1 b_0$$

$$(2^{N-1} 2^{N-2} \dots 2^k 2^{k-1} \dots 2^2 2^1 2^0) \cdot 2^{-k} \text{ k décalages}$$

$$2^{N-1-k} 2^{N-2-k} \dots 2^0 2^{-1} \dots 2^{2-k} 2^{1-k} 2^{-k}$$

Alors la valeur représentée appartient à l'intervalle :

Le quantum de quantification est :



Remarque : La virgule n'a pas d'existence matérielle. Les opérateurs arithmétiques traitent les nombres à virgule fixe comme des entiers (suite de 0 et de 1).

24

24

3. Codages à partir des deux chiffres 0 et 1

d. Représentations des nombres non entiers (cont.)

Représentation à virgule fixe en $Q_{m,k}$ et complément à 2

On n'ajoute pas 1

Exemple : -12,25

12,25 : 0000 1100 0100

Complément à 1

1111 0011 1011

+1

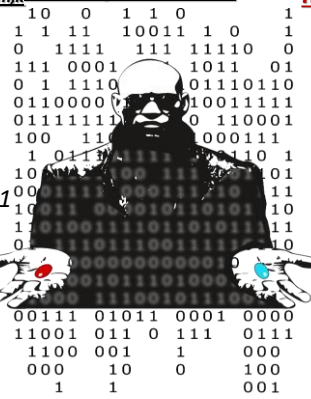
1111 0100 1011

1 0100 1011

-16 +4 +0,5 +0,125 +0,0625

-11,3125

FAUX



La virgule n'a pas d'existence matérielle. **Dans la machine, il n'y a que des 0 et des 1.**

nombres non entiers signés

On ajoute un bit à 1 à droite de la représentation

12,25 : 0000 1100 0100

Complément à 1

1111 0011 1011

+1 à droite

1111 0011 1100

1 0011 1100

-16 +2 +1 +0,5 +0,25

-12,25 ✓

25

25

3. Codages à partir des deux chiffres 0 et 1

d. Représentations des nombres non entiers (cont.)

Obtenir la représentation à virgule fixe en $Q_{m,k}$

Méthode 1 :

Si négatif : représentation de la valeur absolue puis complément à 2.

Pour les nbs positifs et la valeur absolue, obtention de la partie entière par divisions successives, obtention de la partie fractionnaire par multiplications successives.

Méthode 2 :

Multiplication du nombre à représenter par 2^k .

Représentation de l'entier obtenu sur $m+k$ bits.

k décalages à droite de la représentation.

26

26

Exemples (1/2)

27

27

Exemples (2/2) conversion en base 2 (16 bits virgule fixe Q12,4) de -621,17

nombre négatif, conversion en 3 étapes

Méthode 1

1- On représente la valeur absolue du nombre soit 621.17

partie entière div. succ.

partie fractionnaire mult. succ.

621	16
13	38
	16
6	2

621 → 26D_h

621 → 001001101101

0,17	0,72
x 16	x 16
2,72	11,52

0,17 → 0,2B_h
arrondi 0,17 → 0,3_h
0,17 → 0,0011

2- Complément à « 1 » de chacun des bits

0010 0110 1101 0011

1101 1001 0010 1100

3- On ajoute un bit à « 1 » tout à droite

+1

1101 1001 0010 1101

L'erreur est introduite lors de l'arrondi : 0,0011 → 0,1875

$$\text{erreur} = |0,1875 - 0,17| = 0,0175 < \text{erreur}_{\max} = q/2 = 2^{-5} = 0,03125$$

28

28

Exemples

conversion en base 2 (16 bits virgule fixe Q12,4) de -621,17

Multiplication du nombre à représenter par 2^k ici 2^4

$$-621,17 * 2^4 = -9938,72$$

Méthode 2

Représentation de l'entier obtenu sur $m+k$ bits, ici 16 bits. L'entier est négatif, on commence par représenter la valeur absolue du nombre.

9938,72 → arrondi 9939

Erreur de conversion
introduite ici (arrondi)

9939	16
3	621
13	38
6	2

9939 → 26D3

0010 0110 1101 0011
1101 1001 0010 1100 Compl à 1
1101 1001 0010 1101 +1

k décalages à droite de la représentation.

1101 1001 0010 1101

Partie entière ↑ Partie fractionnaire

29

29

Représentation à virgule flottante

Une valeur est représentée par une mantisse et un exposant.

$$x = \pm m.b^e$$

b : base; m : mantisse, e : exposant.



Question : Pourquoi virgule flottante ?

Par la suite, on ne considère que l'étude en base 2 (b=2).

$$x = \pm m.2^e$$

30

30

On adapte le zoom



Représentation à virgule flottante (cont.)

Avec cette écriture, on peut avoir plusieurs solutions pour représenter une même valeur. Alors, on fixe une **contrainte sur la mantisse** de façon à avoir une écriture unique pour la valeur.

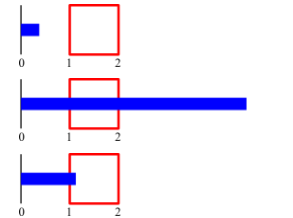
$$1 \leq |m| < 2$$

La mantisse est représentée en virgule fixe.
On utilise un bit pour le signe.

Finalement, le nombre en virgule flottante s'écrit :

$$x = (-1)^S \cdot 1, F \cdot 2^E$$

Le 1 de la partie entière n'a pas besoin d'être stocké. Il s'agit d'un 1 implicite.



31

31

Représentation à virgule flottante (cont.)

Norme IEEE 754 (1985)

C'est la représentation standard pour l'arithmétique en virgule flottante, utilisée par la plupart des systèmes pour améliorer la compatibilité entre les architectures et la portabilité des applications.

$$x = (-1)^S \cdot 1, F \cdot 2^{E-B}$$

(B biais ou décalage
1+mantisse
(E≠0))

Exposant biaisé : l'exposant est représenté par un entier auquel on soustrait un biais B.

Exemple de la représentation IEEE 754 sur 32 bits (*simple précision*) :

Exposant E sur 8 bits, F sur 23 bits : $x = (-1)^S \cdot 1, F \cdot 2^{E-127}$

$$S \ E_7 E_6 E_5 E_4 E_3 E_2 E_1 E_0 \ F_{22} F_{21} F_{20} \dots F_2 F_1 F_0$$

Représentation IEEE 754 sur 64 bits (*double précision*), exposant E sur 11 bits et F sur 52.

32

32

Représentation à virgule flottante (cont.)

Norme IEEE 754 (1985) : exemple

-44 828 721,152

1- mise en forme : signe puissance de 2 et mantisse

33

33

Représentation à virgule flottante (cont.)

Norme IEEE 754 (1985) exemple

$A = -40468,48$

1- mise en forme : signe puissance de 2 et mantisse $A = (-1)^1 \times 1,235 \times 2^{15}$

$E-127 = 15 \rightarrow E = 142 \rightarrow 1000\ 1110$

0,235	0,76	0,16	0,56	0,96	0,36
<u>x16</u>	<u>x16</u>	<u>x16</u>	<u>x16</u>	<u>x16</u>	<u>x16</u>
3,76	12,16	2,56	8,96	15,36	5,76
0011	1100	0010	1000	1111	0101
					011

$A = 1\ 1000\ 1110\ 0011\ 1100\ 0010\ 1000\ 1111\ 011$

34

34

Représentation à virgule flottante (cont.)

Norme IEEE 754 (cont.)

Nombres dénormalisés : en plus de la représentation normalisée avec le 1 implicite, la norme définit une représentation alternative lorsque l'exposant vaut 0. Pour représenter les valeurs très proches de 0.

$$x = (-1)^S \cdot 0, F \cdot 2^{1-B}$$

Exposant = 0

Mantisse $\neq 0$

Cette dénormalisation permet de représenter le 0 lorsque $F=0$.

Valeur la plus petite dénormalisée : $0,0000\dots 01 \cdot 2^{1-B}$

Continuité avec normalisé :
 dénormalisé : $0,1111\dots 111 \cdot 2^{1-B}$
 normalisé : $1,0000\dots 0000 \cdot 2^{1-B}$

Représentations spéciales :

infini : représenté par un exposant maximum et $F=0$. Le bit de signe distingue $+\infty$ et $-\infty$.

NaN : Not a Number, correspondant au résultat de calcul incorrect.

Représenté par un exposant maximal et F non nul.

35

35

Représentation à virgule flottante (cont.)

Norme IEEE 754 (cont.)

	valeur	S	$E_7 E_6 E_5 E_4 E_3 E_2 E_1 E_0$	$F_{22} F_{21} F_{20} \dots F_2 F_1 F_0$
Exemples : sur 32 bits	0	0	0000 0000	000.....000
	-0	1	0000 0000	000.....000
	∞	0	1111 1111	000.....000
	$-\infty$	1	1111 1111	000.....000
	NaN	0	1111 1111	00010....0
	6,5	0	1000 0001	1010.....00
	$7,347 \cdot 10^{-39}$	0	0000 0000	1010.....10

36

36



Nombre de décimales significatives avec 32 bits pour π ?

37

37



Représentation à virgule flottante (cont.)

La représentation IEEE 754 sur 64 bits (double précision), exposant E sur 11 bits et F sur 52 bits, correspond à combien de chiffres significatifs derrière la virgule en décimal ?

38

38

Comparaison virgule fixe/ virgule flottante

Des critères de comparaison sont la **dynamique de représentation**, le **bruit de quantification** et la **complexité des opérateurs arithmétiques**.

La dynamique se définit comme le rapport des valeurs maximale et minimale de la représentation.

$$\text{dynamique} = \frac{\text{valeur max}}{\text{valeur min}}$$

capacité à représenter des valeurs très petites (résolution) et très grandes (étendue)

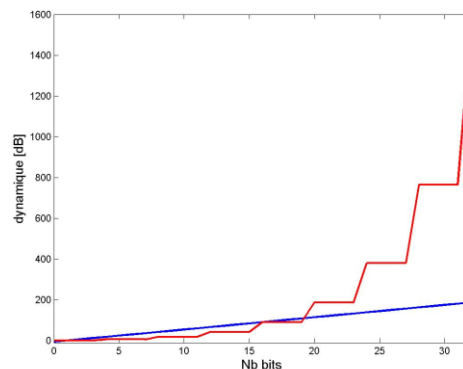
On l'utilise souvent en dB par :

$$D_N(\text{dB}) = 20 \log \left(\frac{\text{valeur max}}{\text{valeur min}} \right)$$

39

39

Comparaison virgule fixe/ virgule flottante (cont.)



virgule fixe 16 bits : $D_N(\text{dB}) = 6.02 \cdot (16-1) = 90.3 \text{ dB}$

virgule flottante 16 bits, E sur 4 bits : $D_N(\text{dB}) = 6.02 \cdot (2^4-1) = 90.3 \text{ dB}$

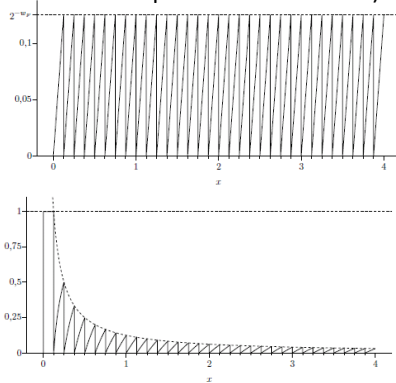
40

40

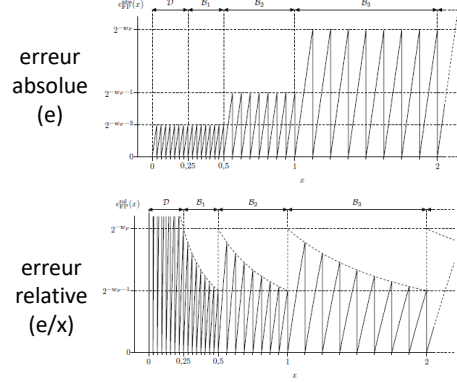
Comparaison virgule fixe/ virgule flottante (cont.)

Critère : bruit de quantification (erreur de représentation relative).

virgule fixe (2 bits partie entière
et 3 bits parties fractionnaire)



virgule flottante (3 bits exposants
et 3 bits mantisse)



Remarque : Déclaration de format de nombre en C

Une déclaration de type de nombre sert à définir :

- la taille (multiple de 8 bits).
- le domaine qui peut varier suivant la nature du type (signé, non signé).

type	taille	domaine
unsigned char	8 bits	0 à 255
char	8 bits	-128 à 127
unsigned int (OS 32 bits)	32 bits	0 à 4 294 967 295
short int	16 bits	-32768 à 32767
int (OS 32 bits)	32 bits	-2 147 483 648 à 2 147 483 647
long	32 bits	-2 147 483 648 à 2 147 483 647
float	32 bits	$1,4 \cdot 10^{-45}$ à $3,4 \cdot 10^{38}$
double	64 bits	$4,9 \cdot 10^{-324}$ à $1,8 \cdot 10^{308}$

3. Codages à partir des deux chiffres 0 et 1

e. Remarque déclaration de format de nombre en C

Remarque : Déclaration de format de nombre en C

```
uint_8 nb = 0b10001111;  
  
itoa(nb, nbText, 10);  
PutString(nbText);
```



143

```
char nb = 0b10001111;  
  
itoa(nb, nbText, 10);  
PutString(nbText);
```



-113

Même contenu mémoire, interprétation différente

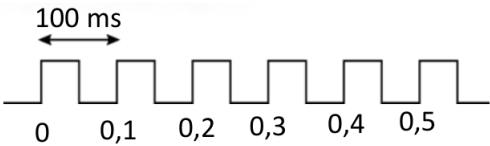
4. De l'importance du choix de codage

En fonction de la vulnérabilité de l'application, le code doit permettre la représentation des grandeurs manipulées sans débordement (ou avec signalisation du débordement), et avoir une résolution suffisante pour que le bruit de quantification (dit autrement l'arrondi de représentation) n'ait pas d'impact sur l'application.

Le compromis est à trouver entre le nombre de bits utilisés pour la représentation, la dynamique associée et la taille/consommation/coût de l'unité de calcul associée.

Exemple d'effet d'arrondi :

Système réel : incrément de 1/10 toutes 100 ms.



en théorie :

temps [s]	valeur sur 8 bit	valeur sur 24 bits
0	0	0
0,1	0,09375	0,1
0,2	0,1875	0,2
...	...	...
100h	360000	337500
...	...	...
1 semaine	604800	567000

6h15 de décalage

0,34s de décalage

100h

1 semaine

en réalité :

1/10 $\Rightarrow$ base 2 : 0.000110011001100110011...

sur 8 bits : 0.0001100 soit 0,09375

Erreur de $6,25 \times 10^{-3}$ à chaque ajout

Au bout de 100h : 6h15 de décalage

sur 24 bits :

Erreur de $9,5367 \times 10^{-8}$ à chaque ajout

Au bout de 100h : 0,34s de décalage

Au bout d'une semaine : 0,576782 s

Appendix II

Effect of Extended Run Time on Patriot Operation

Hours	Seconds	Calculated Time (Seconds)	Inaccuracy (Seconds)	Approximate Shift In Range Gate (Meters)
0	0	0	0	0
1	3600	3599.9966	.0034	7
8	28800	28799.9725	.0025	55
20 (a)	72000	71999.9313	.0687	137
48	172800	172799.8352	.1648	330
72	259200	259199.7528	.2472	494
100 (b)	360000	359999.6667	.3333	687

a. Continuous operation exceeding about 20 hours--target outside range gate
b. Alpha Battery ran continuously for about 100 hours



Page 15 -- GAO/IMTEC-92-26 Patriot Missile Software Problem

erreur de distance = erreur de temps $\times$ vitesse

~ 2000 m/s

4. De l'importance du choix de codage

Exemple de sous capacité de représentation :

- The internal SRI software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer. This resulted in an Operand Error. The data conversion instructions (in Ada code) were not protected from causing an Operand Error, although other conversions of comparable variables in the same place in the code were protected.
- The error occurred in a part of the software that only performs alignment of the strap-down inertial platform. This software module computes meaningful results only before lift-off. As soon as the launcher lifts off, this function serves no purpose.
- The alignment function is operative for 50 seconds after starting of the Flight Mode of the SRIs which occurs at $H_2 - 3$ seconds for Ariane 5. Consequently, when lift-off occurs, the function continues for approx. 40 seconds of flight. This time sequence is based on a requirement of Ariane 4 and is not required for Ariane 5.
- The Operand Error occurred due to an unexpected high value of an internal alignment function result called BH, Horizontal Bias, related to the horizontal velocity sensed by the platform. This value is calculated as an indicator for alignment precision over time.
- The value of BH was much higher than expected because the early part of the trajectory of Ariane 5 differs from that of Ariane 4 and results in considerably higher horizontal velocity values.

Ariane 501 Inquiry Board report
page 4

coût 500 millions !

ARIANE 5

Flight 501 Failure

Valeur plus grande que la
représentation possible sur 16 bits

Accélération Ariane V 5 fois plus
grande que Ariane IV



4. De l'importance du choix de codage

Quand va-t-on avoir un problème avec le format du temps sur ordinateur ?

Représentation du temps sur les ordinateurs (heure POSIX ou heure UNIX) :

entier signé sur 32bits pour représenter le nombre de secondes depuis le 1^{er} janvier 1970 à minuit.

Maximum :

Binary : 01111111 11111111 11111111 11110000
Decimal : 2147483632
Date : 2038-01-19 03:13:52 (UTC)
Date : 2038-01-19 03:13:52 (UTC)



- Comment sont représentées d'un point de vue électrique les valeurs numériques dans les composants numériques (circuits logiques, microprocesseurs, microcontrôleurs) ?
- Dans quels cas utilise-t-on la base 10 ?
- Pourquoi utilise-t-on la base 16 ?
- Qu'est-ce qu'un code pondéré ?

49

49



- Quelles caractéristiques doit présenter un codage utilisé pour le calcul ?
- Quel codage est utilisé en pratique dans les circuits numériques pour représenter les nombres positifs et négatifs ?
- Comment représente-t-on un nombre positif en complément à 2 ?
- Comment représente-t-on un nombre négatif en complément à 2 ?

50

50



- Comment est représentée physiquement la virgule du codage à virgule fixe, au cœur des circuits numériques ?
- Pourquoi impose-t-on une contrainte à la valeur de la mantisse, dans le cas du codage en virgule flottante ?
- Quels sont les avantages/inconvénients relatifs des codages, en virgule fixe et virgule flottante ?