

SYSTÈME DE NUMÉRATION(NUMBER SYSTEM)

Mme AMGHAR.D
1ere ING Informatique
Module :ARCHI1

CODAGE DE L'INFORMATION

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 binaire (**BIT : Binary digit**) : une suite de 0 et de 1 .

Par exemple: 01001011.

CODAGE DE L'INFORMATION

- Le terme bit (b minuscule dans les notations) signifie « **binary digit** », c'est-à-dire 0 ou 1 en numérotation binaire.
- Il s'agit de la plus petite unité d'information manipulable par une machine numérique.
- Il est possible de représenter physiquement cette information binaire par un phénomène physique (un signal électrique, magnétique..), qui, au-delà d'un certain seuil, correspond à la valeur 1.

CODAGE DE L'INFORMATION

Codage de l'information :

permet d'établir une correspondance qui permet sans ambiguïté de passer d'une représentation (dite externe) d'une information à une autre représentation (dite interne : sous forme binaire) de la même information, suivant un ensemble de règles précises.

Exemple : Le nombre 12 :

- 12 est la **représentation externe** du nombre douze
- 1100 est la **représentation interne** de 12 dans la machine

SYSTÈME DE NUMÉRATION(NUMBER SYSTEM)

alphabet a (set of symbols or digits)

Un système de numération décrit la façon avec laquelle les nombres sont représentés. Il est défini par :

- Un **alphabet** A (Set of Symbols or Digits- A) :
L'alphabet d'un système de numération est l'ensemble de symboles ou de chiffres qui sont utilisés pour représenter les nombres.

Exemple: le système décimal (decimal system) utilise les chiffres(number) de 0 à 9 comme son alphabet.

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

rules for writing numbers (juxtaposition of symbols)

Des règles d'écritures des nombres (Juxtaposition de symboles): Les règles d'écriture définissent comment les symboles de l'alphabet sont combinés pour représenter des nombres. La position des chiffres dans un nombre est cruciale et détermine la valeur du chiffre. Cela suit généralement une règle de puissances de la base du système.

Exemple de Système de numération decimal:

le nombre **4134** s'écrit comme suit :

$$4134 = 4 \times 10^3 + 1 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

Chiffre de poids 10^3 Chiffre de poids 10^2

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

rules for writing numbers (juxtaposition of symbols)

Système de numération positionnel pondéré à base b :

- Un système de numération positionnel pondéré à **base** b est défini sur un **alphabet** de b chiffres :

$$A = \{c_0, c_1, \dots, c_{b-1}\} \text{ avec } 0 \leq c_i < b$$

Soit $N = a_{n-1} a_{n-2} \dots a_1 a_0 (b)$: représentation en base b sur n chiffres

a_i : est un chiffre de l'alphabet de poids i (position i).

a_0 : chiffre de poids 0 appelé le chiffre de poids faible.

a_{n-1} : chiffre de poids $n-1$ appelé le chiffre de poids fort.

$$4134 = 4 \times 10^3 + 1 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

positional weighted number system with base

Système de numération positionnel pondéré à base b :

- La valeur de N en base 10 est donnée par :

$$N = a_{n-1} a_{n-2} \dots a_1 a_0 (10)$$

$$= \sum_{i=0}^{n-1} a_i b^i \quad (\text{ceci s'appelle forme polynomiale})$$

Exemple: base10={1,2,3,4,5,6,7,8,9}

$$N = 4134_{(10)}$$

$$N = 4 \times 10^3 + 1 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

base of the number system

Bases de numération (Binaire, Octale et Hexadécimale)

- Système **binaire** (binary system) (**b=2**) utilise deux chiffres : $\{0,1\}$ C'est avec ce système que fonctionnent les ordinateurs .
- Système **Octale** (octal system) (**b=8**) utilise huit chiffres : $\{0,1,2,3,4,5,6,7\}$ Utilisé il y a un certain temps en Informatique, il permet de coder **3 bits** par un seul symbole.

SYSTÈME DE NUMÉRATION(NUMBER SYSTEM)

base of the number system

Bases de numération (Binaire, Octale et Hexadécimale)

- Système **Hexadécimale**(hexadecimal system) (**b=16**)

utilise 16 chiffres :

{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

Où : **A=10₍₁₀₎, B=11₍₁₀₎, C=12₍₁₀₎, D=13₍₁₀₎,**

E=14₍₁₀₎ et F=15₍₁₀₎

Cette base est très utilisée dans le monde de la micro informatique. Elle permet de coder **4 bits** par un seul symbole.

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

Codage des données:

On distingue 2 types de données:

- **non numériques** (codage assez simple car aucune opération arithmétique ou logique ne sera appliquée sur ces données, une table de correspondance suffit)
- **numériques** (codage complexe qui doit faciliter la mise en place de circuits réalisant les opérations arithmétiques).

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

Données non numériques:

- ✓ Afin de faciliter les échanges entre machines, des codages binaires normalisés ont été établis (EBCDIC, ASCII, Unicode, etc.)
- ✓ Nombre variables de bits: 6, 7, 8 16, 32
- ✓ Certains bits sont réservés au "contrôle" ou à la "correction" des autres.

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

Données numériques:

- ✓ Nombres entiers positifs ou nuls : 0 ; 1 ; 315
- ✓ Nombres entiers négatifs : -1 ; -1255
- ✓ Nombres fractionnaires : 3,1415 ; - 0,5
- ✓ Les opérations arithmétiques (+, -, *, /) sont effectuées en arithmétique binaire.

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

transcoding or base conversion

Transcodage ou Conversion de bases

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.

Les conversions possibles sont les suivantes:

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

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

transcoding or base conversion

Correspondance entre les systèmes les plus utilisés

<i>Décimal</i>	<i>Octal</i>	<i>Binaire</i>	<i>Hexadécimal</i>
0	0	0	0
1	1	1	1
2	2	10	2
3	3	11	3
4	4	100	4
5	5	101	5
6	6	110	6
7	7	111	7
8	10	1000	8
9	11	1001	9
10	12	1010	A
11	13	1011	B
12	14	1100	C
13	15	1101	D
14	16	1110	E
15	17	1111	F
16	20	10000	10

SYSTÈME DE NUMÉRATION(NUMBER SYSTEM)

converting a base b to base decimal number

Binaire, octal ou hexadécimal vers décimal:

Exemple : trouvez la représentation décimal des Nombre

Base 2: $1001001_{(2)} = 1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 0 \times 2^5 + 1 \times 2^6$
 $= 1 + 0 + 0 + 8 + 0 + 0 + 64$
 $= (73)_{10}$

Base 8: $(123)_8 = 3 * 8^0 + 2 * 8^1 + 1 * 8^2$
 $= 3 + 16 + 64$
 $= (83)_{10}$

SYSTÈME DE NUMÉRATION(NUMBER SYSTEM)

converting a base b to base decimal number

Binaire, octal ou hexadécimal vers décimal:

Base16:

Exemple : trouvez la représentation décimal des nombre $(E5)_{16}$

$$\begin{aligned}(E5)_{16} &= 5*16^0 + E*16^1 \\ &= 5 + 14*16^1 \\ &= 5 + 224 \\ &= (229)_{10}\end{aligned}$$

SYSTÈME DE NUMÉRATION(NUMBER SYSTEM)

converting a decimal number to base b

Du Décimal vers une Base $b(2,8,16)$:

La règle à suivre est les **divisions successives** :

- On divise le nombre **par la base b** puis le quotient par la base et ainsi de suite jusqu'à obtenir 0.
- La suite des restes correspond aux symboles de la base visée.
- On obtient en premier le chiffre de **poids faible** et en dernier le chiffre de **poids fort**.

SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

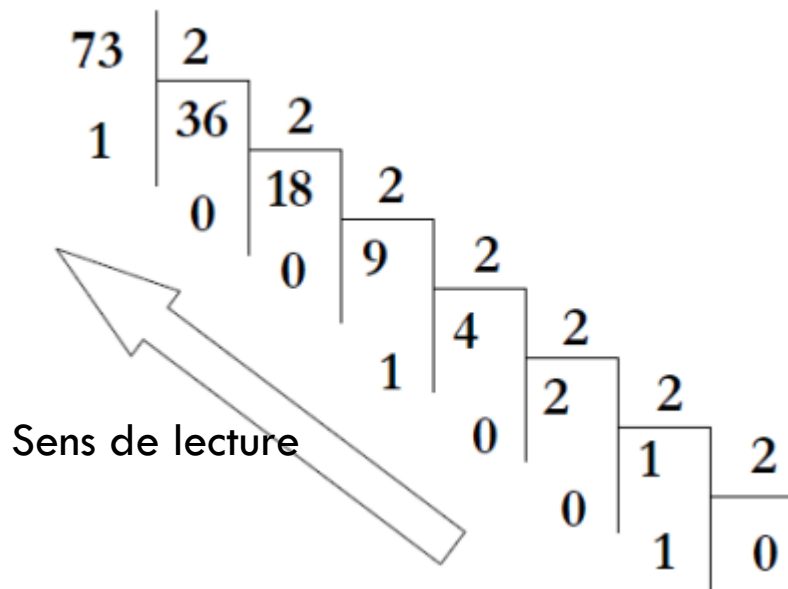
converting a decimal number to base 2

Décimal vers Binaire :

Exemple :

Soit $N = 73_{(10)}$ un nombre représenté en base décimale 10 Trouver sa représentation binaire :

$$N = 1001001_{(2)}$$



SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

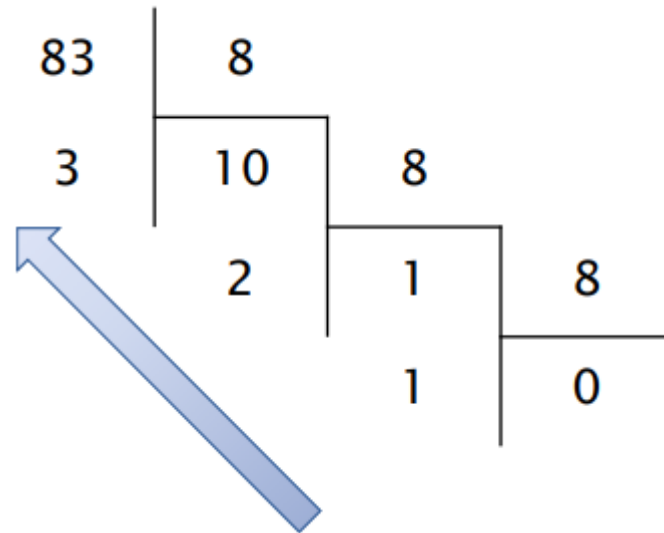
converting a decimal number to base 8

Décimal vers Octal :

Exemple :

Soit $N = 83_{(10)}$ un nombre représenté en base décimale 10

Trouver sa représentation en octal :



$$83_{(10)} = 123_{(8)}$$

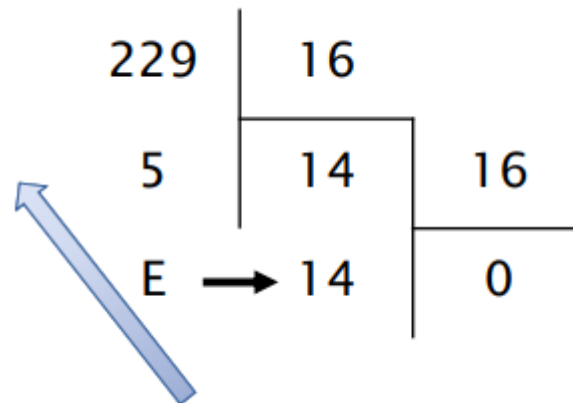
SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

converting a decimal number to base 16

Décimal vers Hexadécimal :

Soit $N = 229_{(10)}$ un nombre représenté en base décimale 10

Trouver sa représentation en hexadécimal :



$$229_{(10)} = E5_{(16)}$$

SYSTÈME DE NUMÉRATION(NUMBER SYSTEM)

Converting a binary number to octal or hexadecimal

Binaire vers octal ou hexadécimal :

Première solution :

convertir le nombre en **base binaire** → vers la **base décimale** puis **convertir ce nombre** (en base 10) vers la base souhaitée (8 ou 16).

Exemple :

$$1011100_{(2)} = 0 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 0 \times 2^5 + 1 \times 2^6$$

$$= 0 + 0 + 4 + 8 + 16 + 0 + 64$$

$$= 92_{10}$$

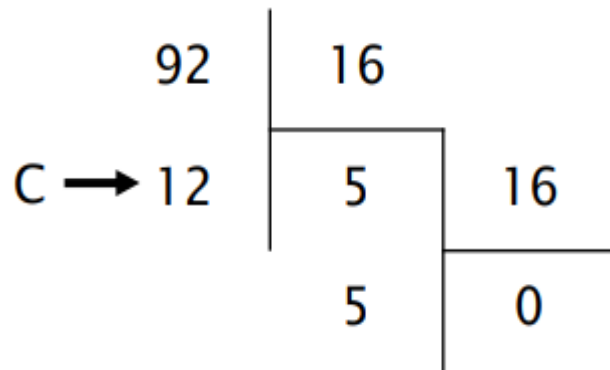
SYSTÈME DE NUMÉRATION (NUMBER SYSTEM)

Converting a binary number to octal or hexadecimal

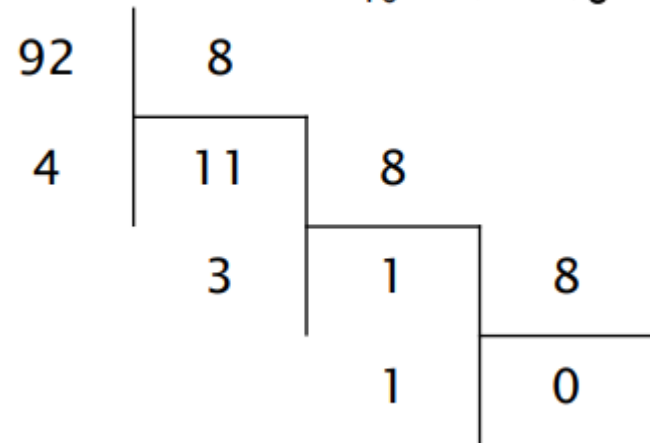
Binaire vers octal ou hexadécimal :

Première solution :

$$(92)_{10} = (5C)_{16}$$



$$(92)_{10} = (134)_8$$



SYSTÈME DE NUMÉRATION(NUMBER SYSTEM)

Converting a binary number to octal or hexadecimal

Binaire vers octal ou hexadécimal :

Deuxième solution : (Utiliser les tables de correspondance)

- **Binaire vers octale** : regroupement des bits en des **groupes** de **trois bits** de droite à gauche puis remplacer chaque groupe par le symbole correspondant dans la base 8.
- **Binaire vers Hexadécimale** : regroupement des bits en des **groupes** de **quatre bits** de droite à gauche puis remplacer chaque groupe par le symbole correspondant dans la base 16.

SYSTÈME DE NUMÉRATION

Binaire vers octal ou hexadécimal :

Deuxième solution :

Exemple :

$$(1011100)_2 = \frac{\textcolor{red}{00}1}{1} \frac{011}{3} \frac{100}{4} = (134)_8$$

$$(1011100)_2 = \frac{\textcolor{red}{0}101}{5} \frac{1100}{C} = (5C)_{16}$$

SYSTÈME DE NUMÉRATION

Binaire vers octal ou hexadécimal : Tables de correspondance

Octale vers Binaire

Octal	Binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Hexadécimal vers binaire

Hexa	Binaire	Hexa	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

CALCUL ARITHMÉTIQUE(ARITHMETIC CALCULATIONS) operations in the binary system

Opérations dans le système binaire :

- Addition +
- Soustraction -
- Multiplication *
- Division /

CALCUL ARITHMÉTIQUE (ARITHMETIC CALCULATIONS)

operations in the binary system

OPÉRATIONS DANS LE SYSTÈME BINAIRE :

Addition :

On écrit les nombres sur des lignes successives en les mettant en colonne, en partant de la droite ; ensuite on additionne les chiffres de chaque colonne en commençant par celle de droite.

On suit les règles suivantes :

$$0 + 0 = 0 ,$$

$$1 + 0 = 1 ,$$

$$0 + 1 = 1 ,$$

$$1 + 1 = 0 \text{ et on retient } 1, \text{ soit } 10$$

CALCUL ARITHMÉTIQUE (ARITHMETIC CALCULATIONS)

operations in the binary system

OPÉRATIONS DANS LE SYSTÈME BINAIRE :

Addition :

Exemple : additionner les nombres $(11011)_2 + (10101)_2$

:

$$\begin{array}{r} \overset{1}{1} \overset{1}{1} \overset{1}{0} \overset{1}{1} \\ + 10101 \\ \hline 110000 \end{array}$$

NB : si on dépasse le nombre de bits autorisé alors on appelle ça un dépassement de capacité ou **overflow**

CALCUL ARITHMÉTIQUE (ARITHMETIC CALCULATIONS)

operations in the binary system

Remarque :

L'entier maximal pouvant être codé dépendra du nombre de bits que l'on réserve pour coder un nombre.

En général les entiers sont codés sur un mot e.g., pour un ordinateur 32 bits : $2^{32} - 1 = 4\,294\,967\,295$.

Dépassement de capacité *overflow*

Lorsque par exemple le résultat d'une opération sur des nombres produit un nombre plus grand que la taille du mot prévu pour représenter ces nombres (e.g., bit de retenue).

CALCUL ARITHMÉTIQUE (ARITHMETIC CALCULATIONS)

operations in the binary system

OPÉRATIONS DANS LE SYSTÈME BINAIRE :

Soustraction :

On écrit les nombres sur des lignes successives en les mettant en colonne, en partant de la droite ; ensuite on soustrait les chiffres de chaque colonne en commençant par celle de droite. On suit les règles suivants :

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0,$$

$0 - 1 = 1$ et on retient 1 (dans ce cas ou la soustraction est impossible donc on empreinte 1 puis on le retourne dans la deuxième colonne)

CALCUL ARITHMÉTIQUE (ARITHMETIC CALCULATIONS)

operations in the binary system

OPÉRATIONS DANS LE SYSTÈME BINAIRE :

Soustraction :

Exemple : effectuer l'opération suivante :

$$(10011)_2 - (1110)_2$$

$$\begin{array}{r} 10011 \\ - 1110 \\ \hline 00101 \end{array}$$

CALCUL ARITHMÉTIQUE (ARITHMETIC CALCULATIONS)

operations in the binary system

OPÉRATIONS DANS LE SYSTÈME BINAIRE :

Multiplication : On écrit les nombres sur des lignes successives en les mettant en colonne, en partant de la droite ; ensuite on effectue la multiplication comme le décimal. On suit les règles suivants :

$$0 * 0 = 0 ,$$

$$1 * 0 = 0 ,$$

$$1 * 1 = 1 ,$$

CALCUL ARITHMÉTIQUE (ARITHMETIC CALCULATIONS)

operations in the binary system

OPÉRATIONS DANS LE SYSTÈME BINAIRE :

Multiplication :

Exemple :

effectuer l'opération suivante : $(1101)_2 \times (101)_2$

$$\begin{array}{r} 1101 \\ * 101 \\ \hline 1101 \\ 0000 \\ 1101 \\ \hline 1000001 \end{array}$$

CALCUL ARITHMÉTIQUE (ARITHMETIC CALCULATIONS)

operations in the binary system

OPÉRATIONS DANS LE SYSTÈME BINAIRE :

Division :

la division s'effectue selon les règles arithmétiques traditionnelles et elle basée sur la soustraction en suivant les règles suivantes :

- soustraction possible le résultat est 1, et 0 dans le cas contraire

0 / 0 indéterminé

1 / 0 indéterminé

0 / 1 = 0

1 / 1 = 1

CALCUL ARITHMÉTIQUE (ARITHMETIC CALCULATIONS)

operations in the binary system

OPÉRATIONS DANS LE SYSTÈME BINAIRE :

Division :

Exemple : effectuer les opérations suivantes :

$$(100100)_2 / (1100)_2$$

$$\begin{array}{r} \overline{100100} \\ - 1100 \\ \hline 001100 \\ - 1100 \\ \hline 0000 \end{array}$$

$$(111100)_2 / (1101)_2$$

$$\begin{array}{r} \overline{111100} \\ - 1101 \\ \hline 001000 \\ - 1101 \\ \hline 001000 \end{array}$$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

- Les nombres **fractionnaires** (fractional numbers) sont ceux qui comportent des **chiffres après la virgule** (digits after the decimal point).
- On représente un nombre fractionnaire N dans une base b comme suit :

$$N = a_{n-1} a_{n-2} \dots a_1 a_0 a_{-1} a_{-2} \dots a_{-p}_{(b)}$$

- a_{n-1} est le chiffre de poids fort (most significant digit.)
- a_{-p} est le chiffre de poids faible (least significant digit).
- n Le nombre de chiffre avant la virgule.
- p Le nombre de chiffre après la virgule.

N=32,53

Avant virgule Après virgule

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

- **Changement d'une base b (2, 8 ou 16) vers la base décimale**

Le passage d'un nombre fractionnaire de la base b (2, 8 ou 16) vers la base 10 est défini par la **formule polynomiale** suivante :

$$(N)_{10} = (a_{n-1}b^{n-1} + \dots + a_1b^1 + a_0b^0 + a_{-1}b^{-1} + a_{-2}b^{-2} \dots + a_{-p}b^{-p})_b$$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

- Changement d'une base b (2, 8 ou 16) vers la base décimale

Exemple 1 :

$$(101.011)_2 = (?)_{10}$$

2 1 0 -1 -2 -3

(101.011)₂

$$(101.011)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$$

$$= 4 + 0 + 1 + 0 + 0.25 + 0.125$$

$$= (5.375)_{10}$$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

- Changement d'une base b (2, 8 ou 16) vers la base décimale :

Exemple 2 : $(123.26)_8 = (?)_{10}$

2 1 0 -1 -2

$\underbrace{\hspace{1.5cm}}_{(123.26)_8}$

$$(123.26)_8 = 1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0 + 2 \times 8^{-1} + 6 \times 8^{-2}$$

$$= 64 + 16 + 3 + 0.25 + 0.09375$$

$$= (83.34375)_{10}$$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

• Changement d'une base **b** (2, 8 ou 16) vers la base décimale

Exemple 3 : $(AB3.E5)_{16} = (?)_{10}$

$$\begin{array}{ccccccc} & & & 2 & 1 & 0 & -1 & -2 \\ & & & \underbrace{\hspace{1.5cm}} & & & & \\ & & & (AB3.E5)_{16} & & & & \end{array}$$

$$\begin{aligned} (AB3.E5)_{16} &= A \times 16^2 + B \times 16^1 + 3 \times 16^0 + E \times 16^{-1} + 5 \times 16^{-2} \\ &= 10 \times 16^2 + 11 \times 16^1 + 3 \times 16^0 + 14 \times 16^{-1} + 5 \times 16^{-2} \\ &= 2560 + 176 + 3 + 0.875 + 0.01953125 \\ &\approx (2739.895)_{10} \end{aligned}$$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

• **Changement de base : 10 \longrightarrow base(2 , 8 ou 16)**

Le passage de la base 10 à la base b est défini par :

- Partie **entière**(integer part) est codée sur **p** bits (**division successive** par b)
- Partie **fractionnaire**(fractional part) est codée par **q** bits en **multipliant** par b successivement jusqu'à ce que la partie fractionnaire soit nulle ou bien le nombre de bits q atteint.

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

• Changement de base : 10 \longrightarrow base (2 , 8 ou 16)

Exemple 1 :

$$(4.25)_{10} = (?)_2$$

- $(4)_{10} = (100)_2$ (**division successive** par 2)
- $0.25 * 2 = 0.5 \ 0$
- $0.5 * 2 = 1.0 \ 1$, partie fractionnaire nulle donc **arrêt** .
- Donc $(4.25)_{10} = (100.01)_2$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

• Changement de base : 10  base(2 , 8 ou 16)

Exemple 2 :

coder $(27.87)_{10}$ en binaire avec 6 bits pour la partie entière et 4 bits pour la partie fractionnaire

• $(27)_{10} = (011011)_2$ (**division successive** par 2)

• $0.87 * 2 = 1.74$ **1**

• $0.74 * 2 = 1.48$ **1**

• $0.48 * 2 = 0.96$ **0**

• $0.96 * 2 = 1.92$ **1** (le nombre de bits pour la partie fractionnaire est atteint)

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

Changement de base : 10 \longrightarrow base (2 , 8 ou 16)

Exemple 2 :

$$\text{Donc } (27,87)_{10} = (011011.1101)_2$$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

• Changement de base : $10 \longrightarrow b(2, 8 \text{ ou } 16)$

Exemple 3 :

Coder $(46.73)_{10}$ en hexadécimal avec 4 chiffres dans la partie fractionnaire (fractional part)

$$(46)_{10} = (2E)_{16}$$

$$0.73 \times 16 = 11.68 \longrightarrow B \quad 0.68 \times 16 = 10.88 \longrightarrow A$$

$$0.88 \times 16 = 14.08 \longrightarrow E \quad 0.08 \times 16 = 1.28 \longrightarrow 1$$

le nombre de chiffres dans la partie fractionnaire est atteint ,

$$\text{donc } (46.73)_{10} = (2E.BAE1)_{16}$$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

- **Changement de base : 2 \rightarrow 8, 16 ou l'inverse**

Le principe de regroupement des bits est appliqué :

- Dans la partie **entière** (**integer part**) le regroupement se fait

de droite à gauche

- Dans la partie **fractionnaire** (**fractional part**) le regroupement se fait de gauche à droite et s'il manque des bits on ajoute des 0 à droite.

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

- **Changement de base : 2 \rightarrow 8, 16 ou l'inverse**
- **Exemple 1 : Coder $(1101101.110011)_2$ en octal et en hexadécimal**
 $(1101101.110011)_2 = (\textcolor{teal}{001} \text{ } 101 \text{ } 101.110 \text{ } 011)_2$

1 5 5 . 6 3
- $(1101101.110011)_2 = (155.63)_8$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

Changement de base : $2 \rightarrow 8, 16$ ou l'inverse

• **Exemple 1** : Coder $(1101101.110011)_2$ en octal et en hexadécimal

$$\bullet (1101101.110011)_2 = (\overset{6}{0}110 \overset{D}{1101}.\overset{C}{1100} \overset{C}{1100})_2$$

$$(1101101.110011)_2 = (6D.CC)_{16}$$

REPRÉSENTATION DES NOMBRES FRACTIONNAIRES (REPRESENTATION OF FRACTIONAL NUMBERS IN A NUMERAL SYSTEM)

- Changement de base : 2 \rightarrow 8, 16 ou l'inverse

- Exemple 1 :

Coder $(1101101.110011)_2$ en octal et en hexadécimal

$$\bullet (1101101.110011)_2 = (\underset{\text{6}}{0110} \underset{\text{D}}{1101}. \underset{\text{C}}{1100} \underset{\text{C}}{1100})_2$$

$$\bullet (1101101.110011)_2 = (6D.CC)_{16}$$

REPRÉSENTATION DES ENTIERS SIGNÉS

(REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

Le codage des entiers signés *signed integers* (positifs et négatifs)

peut se faire de trois manières :

- Binaire **signé**(*signed binary*) (ou signe + valeur absolue).
- Binaire **complément à 1**(*one's complement binary*) (ou complément restreint ou logique).
- Binaire **complément à deux**(*two's complement binary*) (ou complément vrai ou arithmétique)

REPRÉSENTATION DES ENTIERS SIGNÉS

(REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

1 - Binaire signé SVA (signed binary) :

sur un ensemble de bits on réserve le bit le plus à gauche (plus fort) (Reserve the leftmost bit(most significant))

La valeur **1** au signe « - »

La valeur **0** au signe « + »

le reste(remainder) pour représenter la valeur absolue du nombre en binaire(The absolute value of the binary number).

REPRÉSENTATION DES ENTIERS SIGNÉS

(REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

1 - Binaire signé BS/SVA (signed binary) :

Exemple :

représenter sur **8 bits** les valeurs +83, -51, -128

Sur 8 bits donc 1er bit pour le signe, les 7 restants pour la Valeur Absolue(**Absolute Value**)

$$(83)_{10} = (1010011)_2 = (01010011)_{BS}$$

$$|-51| = (51)_{10} = (0110011)_2 = (10110011)_{BS}$$

$$|-128| = (128)_{10} = (10000000)_2$$

-> **Dépassement**

over flow **nécessite 9 bits**

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

1 - Binaire signé SVA (signed binary) :

- Intervalle des nombres (Number range) : sur n bits on peut coder les nombres entre $-(2^{n-1} - 1)$ et $+(2^{n-1} - 1)$.

Exemple : sur 4 bits (entre -7 et +7)

- **Inconvénients** :

1- Deux représentations différentes pour le 0 :
par exemple sur 4 bits :

$$(+0)_{10} = (0000)_{BS}$$

$$(-0)_{10} = (1000)_{BS}$$

REPRÉSENTATION DES ENTIERS SIGNÉS

(REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

1 - Binaire signé SVA (signed binary) :

- Inconvénients :

2 - Conflits lors des opérations arithmétiques :

Exemple :

Faite l'opération suivante en binaire sur 8 bits :

$$3 - 4 = 3 + (-4)$$

$$3 \rightarrow 0000\ 0011$$

$$-4 \rightarrow 1000\ 0100$$

$$\begin{array}{r} 3 \\ -4 \\ \hline -1 \end{array} \quad \begin{array}{r} 0000\ 0011 \\ 1000\ 0100 \\ \hline 1000\ 0111 \end{array} \rightarrow (-7)_{10} \text{ faux}$$

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

2- Binaire complément à 1 CA1 (one's complement binary) : Aussi appelé **Complément Logique (CL)** ou **Complément Restreint (CR)** :

- les nombres positifs (positif number) sont codés de la même façon qu'en binaire pure.
- Un nombre négatif (negatif number) est codé en inversant chaque bit de la représentation de sa valeur absolue
- le bit le plus à gauche (the leftmost bit) est utilisé pour représenter le signe du nombre :
 - s'il est = 1 alors le nombre est négatif (negatif number)
 - sinon le nombre est positif (positif number)

REPRÉSENTATION DES ENTIERS SIGNÉS

(REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

2 - Binaire complément à 1 **CA1** :

Exemple :

$$(+33)_{10} = (00100001)_{BS} = (00100001)_{CA1}$$

$$(-33)_{10} = (10100001)_{BS} = (\mathbf{1}1011110)_{CA1}$$

- **Inconvénients :**

1- Deux représentations différentes pour le 0 : par exemple sur 4 bits :

$$(+0)_{10} = (0000)_{CA1}$$

$$(-0)_{10} = (1111)_{CA1}$$

2- Opérations arithmétiques difficiles (Difficult arithmetic operations).

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

2 - Binaire complément à 1 **CA₁** :

- **Addition**

L'addition en complément à 1 est effectuée comme le binaire pur en suivant les règles suivante :

- Deux nombres de **signes différents** (different signs) : pas de dépassement de capacité(overflow) et si on trouve une retenue on l'ajoute au résultat.
- Deux nombres de **même signes négatifs** (same negative signs) : risque de dépassement (overflow) si on trouve une retenue on l'ajoute au résultat, si le bit du signe est 0 alors **Overflow**.

REPRÉSENTATION DES ENTIERS SIGNÉS

(REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

2 - Binaire complément à 1 CA₁ :

- **Addition**
- Deux nombres de même signes positifs (same positive signs) : risque de dépassement overflow et si on trouve une retenue on l'ajoute au résultat, si le bit du signe est 1 alors **Overflow**.

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

2 - Binaire complément à 1 **CA1** :

- **Addition**
- Exemple : Sur 4 bits effectuer les opérations suivantes en CA1 : $2+5$, $-2+5$, $-2-5$, $5+3$, $-5-3$

$$2+5 = (+2)_{10} + (+5)_{10} = (0010)_{CA1} + (0101)_{CA1} \\ = (0111)_{CA1} = (+7)_{10}$$

$$-2+5 = (-2)_{10} + (+5)_{10} = (1101)_{CA1} + (0101)_{CA1} \\ = (0010 + 1 \text{ du retenue})_{CA1} \\ = (0011)_{CA1} = (+3)_{10}$$

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

2 - Binaire complément à 1: CA1 Addition

- Exemple : Sur 4 bits effectuer les opérations suivantes en

CA1 : $2+5$, $-2+5$, $-2-5$, $5+3$, $-5-3$

$$\begin{aligned}-2 - 5 &= (-2)_{10} + (-5)_{10} = (1101)_{CA1} + (1010)_{CA1} \\ &= (0111 + 1 \text{ du retenue})_{CA1} \\ &= (1000)_{CA1} = (-7)_{10}\end{aligned}$$

$$\begin{aligned}5 + 3 &= (+5)_{10} + (+3)_{10} = (0101)_{CA1} + (0011)_{CA1} \\ &= (1000)_{CA1} \rightarrow \text{Overflow}\end{aligned}$$

$$\begin{aligned}-5 - 3 &= (-5)_{10} + (-3)_{10} = (1010)_{CA1} + (1100)_{CA1} \\ &= (0110 + 1 \text{ du retenue})_{CA1} = (0111)_{CA1} \rightarrow \text{Overflow}\end{aligned}$$

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

3 - Binaire complément à 2 **CA2**:

Aussi appelé **Complément vrais** (CV) ou **complément arithmétique** :

- Les nombres positifs (**positifs numbers**) sont codés de la même façon qu'en binaire pure.
- Un nombre négatif (**negatifs numbers**) est codé en inversant chaque bit de la représentation de sa valeur absolue puis on ajoute 1 (c.à.d : **CA2=CA1+1**)
- le bit le plus à gauche (**the leftmost bit**) est utilisé pour représenter le signe du nombre :
 - s'il est = 1 alors le nombre est négatif sinon le nombre est positif

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

3 - Binaire complément à 2:

Exemple :

on travaille sur 8 bits

$$\begin{aligned} (+33)_{10} &= (00100001)_{BS} = (00100001)_{CA1} \\ &= (00100001)_{CA2} \end{aligned}$$

$$\begin{aligned} (-33)_{10} &= (10100001)_{BS} = (11011110)_{CA1} + 1 \\ &= (11011111)_{CA2} \end{aligned}$$

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

3 - Binaire complément à 2 CA2 :

Une autre méthode pour calculer le complément à 2 consiste à démarrer du bit le plus faible :

- Si le bit est un zéro, on recopie le « 0 » jusqu'au premier 1 rencontré.
- Si c'est un « 1 » on garde ce premier 1 ensuite on inverse tous les bits restant .

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

3 - Binaire complément à 2 **CA2** :

Exemple :

$$\begin{aligned}(-33)_{10} &= (10100001)_{BS} \\ &= (11011111)_{CA2}\end{aligned}$$

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

3 - Binaire complément à 2 CA2 :

Un seul codage pour 0. Par exemple sur 8 bits :

$$+0 \text{ ou } -0 = (00000000)_{CA2}$$

Etendu de codage : Avec n bits, on peut coder de $-(2^{n-1})$ à $(2^{n-1} - 1)$

Sur 8 bits par exemple : codage des nombres de -128 à +127

$$+0 = 00000000 \quad -0 = 00000000$$

$$+1 = 00000001 \quad -1 = 11111111$$

... ..

$$+127 = 01111111 \quad -128 = 10000000$$

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

3 - Binaire complément à 2 **CA2** :

Addition

L'Addition est effectuée comme le binaire pur en suivant les règles :

- Deux nombres de signes différents (**opposite signs**) : pas de dépassement et si on trouve une retenue **on l'ignore** et le résultat est en CA2.
- Deux nombres négatifs (**same negative signs**) : si on trouve une retenue **on l'ignore**, si le bit du signe du résultat est 0 alors **Overflow**.

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

3- Binaire complément à 2: CA2

Addition

- Deux nombres positifs (same positive signs) : si on trouve une retenue on l'ignore et si le bit du signe du résultat est 1 alors **Overflow**.

REPRÉSENTATION DES ENTIERS SIGNÉS

(REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

3 - Binaire complément à 2:

Addition

- **Exemple** : Sur 4 bits effectuer les opérations suivantes en CA2 : $2+5$, $-2+5$, $5+3$

$$\begin{aligned} 2 + 5 &= (+2)_{10} + (+5)_{10} = (0010)_{CA2} + (0101)_{CA2} \\ &= (0111)_{CA} = (+7)_{10} \end{aligned}$$

$$\begin{aligned} -2 + 5 &= (-2)_{10} + (+5)_{10} = (1110)_{CA2} + (0101)_{CA2} \\ &= (1 \text{ ignoré } 0011)_{CA2} = (0011)_{CA2} = (+3)_{10} \end{aligned}$$

$$\begin{aligned} 5 + 3 &= (+5)_{10} + (+3)_{10} = (0101)_{CA2} + (0011)_{CA2} \\ &= (1000)_{CA2} \rightarrow \text{Overflow} \end{aligned}$$

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

Intervalle de représentation des entiers N :

SVA/Bs: $- (2^{n-1} - 1) \leq N \leq + (2^{n-1} - 1)$

C à 1 (CR): $- (2^{n-1} - 1) \leq N \leq + (2^{n-1} - 1)$

C à 2 (CV): $- 2^{n-1} \leq N \leq + (2^{n-1} - 1)$

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

➤ Opérations arithmétiques avec les nombres signés en complément à 2

Soustraction

La soustraction est considérée comme un cas particulier de l'addition :

$$A - B = A + (-B)$$

$$-A - B = (-A) + (-B)$$

- On prend donc le système complément à deux pour représenter les négatifs, et on effectue une addition.

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

➤ Opérations arithmétiques avec les nombres signés en complément à 2

Multiplication

Les deux nombres doivent être représentés dans une forme sans complément (i.e., valeur absolue BS).

On effectue la multiplication et on décide du signe du résultat :

- ✓ opérandes sont de mêmes signes : le résultat est positif
- ✓ opérandes signés différents : le résultat est négatifs, on le représente avec son complément à 2

REPRÉSENTATION DES ENTIERS SIGNÉS (REPRESENTATION OF SIGNED INTEGERS IN NUMERAL SYSTEMS)

1 - Binaire complément à 2: Addition

• Exercice :

Réaliser les opérations suivantes en complément à 2. Utiliser un format d'un octet pour chaque nombre (y compris le bit de signe) :

$1-2$, $51+127$, $-3-127$, $-127+127$, $-63-63$.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

Les formats de représentation des nombres réels sont :

1. Format virgule fixe (Fixed-point format)
2. Format virgule flottante (floating-point format)

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

1. Format virgule fixe :

- Utilisé par les premières machines
- Possède une partie entière (*integer part*) et une partie décimale fractionnaire (*fractional part*) séparés par une virgule.
- La position de la virgule est fixe.

Exemple : $(54.25)_{10}$, $(10.001)_2$, $(A1.F0B)_{16}$

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

1. Format virgule fixe :

- Le codage du nombre en virgule fixe consiste à définir la position de la virgule selon un format donné, c'est-à-dire la taille de la partie entière (integer part) ainsi que la taille de la partie fractionnaire (fractional part).
- Les bits à gauche (the leftmost bits) de la virgule représentent la partie entière signée du nombre tandis que la partie droite (the rightmost bits) représente la partie fractionnaire
- Le bit le plus à gauche pour le signe (the leftmost bit for the sign).

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

2. Format virgule flottante :

- Utilisé actuellement par les machines.
- Défini par $\pm m \times b^e$:

un signe \pm ou -1 (the sign)

une mantisse m (the fraction or mantissa)

un exposant e (un entier relatif) (the exponent)

b une base (2, 8, 10 ou 16)

Exemple :

$$(0.45 \times 10^2)_{10}, (10.1 \times 10^{-2})_2, (A0.C4)$$

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule fixe :**

Supposant un nombre N et une base b il est représenté en format virgule fixe , par la formule suivante :

$$(N)_b = (a_{n-1} \dots \dots a_1 a_0 . a_{-1} a_{-2} \dots a_{-p})_b$$

a_{n-1} est le chiffre de poids fort.

a_{-p} est le chiffre de poids faible.

n Le nombre de chiffre avant la virgule.

p Le nombre de chiffre après la virgule

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule fixe :**

Exemple :

$$(101.011)_2 = (5.25)_{10}$$

$$(AB3.E5)_{16} = (2739.895)_{10}$$

$$(128.26)_8 = (83.34375)_{10}$$

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- Virgule fixe :
- Changement d'une base b (2,8 ou 16) vers la base décimale (decimal base)

Le passage d'un nombre fractionnaire (fractional number) de la base b (2,8 ou 16) vers la base 10 est défini par la formule polynomiale suivante :

$$(N)_{10} = (a_{n-1}b_{n-1} + \dots + a_1b_1 + a_0b_0 + a_{-1}b_{-1} + a_{-2}b_{-2} \dots + a_{-p}b_{-p})_b$$

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

Virgule fixe:

- **Changement de base : 10 vers une base b (2, 8 ou 16)**

Le passage de la base 10 à la base b est défini par :

- ✓ Partie **entière** (**integer part**) est codée sur p bits (**division successive** par b)
- ✓ Partie **fractionnaire** (**fractional part**) est codée par q bits en **multipliant** par b successivement jusqu'à ce que la partie fractionnaire soit nulle ou bien le nombre de bits q atteint.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule fixe**

Changement de base : 10 vers une base b (2 , 8 ou 16)

- **Exemple** : Représenter le nombre réel (-6.125) en format virgule fixe (**1 bit** de signe, **8 bit** pour la partie entière et **7 bits** pour la partie fractionnaire)

(**1 bit** for the sign, **8 bits** for the integer part, and **7 bits** for the fractional part)

- Il faut d'abord convertir le nombre en binaire pour pouvoir le représenter en machine :

$$(-6.125)_{10} = (-110,001)_2$$

On représente donc le nombre selon le format indiqué. On obtient donc :

1 | 00000110 | 0010000

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule flottante** floating-point :

Par exemple si on veut représenter une valeur scientifique telle que la masse d'un électron (**The mass of an electron**) qui est infiniment petite : **me=0.000.....000091094**

- On préfère utiliser l'écriture scientifique(**The scientific notation**) suivante $me=9.1094 \times 10^{-31}$ au lieu d'écrire zéro virgule une trentaine de zéros puis 91094.

- Généralisation :

soit x un réel **x= signe mantisse x 10^n**

X real= sign mantissa x 10^n

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule flottante:**

Avantage: permet de représenter des nombres très grands et très petits sans s'encombrer de zéros.

- Dans la **base binaire** c'est pratiquement la même chose.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule flottante:**

Au lieu d'écrire les nombres réels avec une virgule fixe qui sépare deux blocs de 0 et 1, on préfère représenter celui-ci avec une écriture semblable à l'écriture scientifique(*scientific notation*) utilisée en base 10.

Exemple 1 : On veut représenter le nombre réel 12.625 en binaire ce qui nous donne :

$$(12.625)_{10} = (1100.101)_2$$

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

Virgule flottante:

Exemple 1 :

- On va représenter ce code binaire sous une forme semblable à l'écriture scientifique (**scientific notation**) du décimal comme suit :

$$1100.101 = 1.100101 \times 2^3$$

- Nous avons déplacé la virgule de trois positions vers la gauche c'est pourquoi nous avons multiplié le nombre binaire résultant par 2 qu'on a élevé à la puissance positive 3

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

Virgule flottante:

Exemple 2 :

- De la même manière si on veut représenté le nombre 0.375 en binaire on aura :
 $(0.375)_{10} = (0.011)_2 = 1.1 \times 2^{-2}$
- Nous avons fait en sorte de laisser le chiffre 1 à gauche de la virgule et tout le reste a sa droite, nous avons donc déplacé la virgule de 2 position vers la droite ce qui est équivalent à la puissance négative -2 .

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule flottante:**

Comme c'est le cas des nombre négatifs qui contiennent le signe « - » qui est inconnu par les ordinateurs et transformé en nombre signés qui ne contient que des 0 et 1. Les nombres réels contiennent d'autres symboles autres que le 0 et 1 comme par exemple : 1.1×2^{-2}

- Ce qui veut dire que les ordinateurs auront du mal à les interpréter.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule flottante:**

En général le codage d'un nombre réel en virgule flottante se présente comme suit : $\pm m \times b^e$

Ou **m** : mantisse

e : exposant

b : base

Exemple 1 :

$$153.36 = 1.5336 \times 10^2 \quad 0.000084 = 8.4 \times 10^{-5}$$

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

Virgule flottante:

Exemple 2 : Codage en base 2, format virgule flottante, de (3.25)

$$\begin{aligned}(3.25)_{10} &= (11.01)_2 \text{ (en virgule fixe)} \\ &= 1.101 \times 2^1 = 110.1 \times 2^{-1}\end{aligned}$$

- L'écriture devient alors:

$$\text{signe mantisse} \times 2^e$$

- **Signe** : c'est le signe du nombre (codé sur 1 bit de poids le plus fort : signe – : bit 1 et le signe + : bit 0)
- **e** : c'est l'exposant qui un entier relatif positif ou négatif.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule flottante:**
- **m** : c'est la mantisse qui est un nombre binaire à virgule. Le seul chiffre avant la virgule étant toujours 1, il n'est pas représenté (on le dit *implicite*), et le codage binaire de la mantisse représente donc uniquement les chiffres après la virgule qui sont en base 2.
- **b** : c'est la base du système.
- Le codage en binaire revient à coder l'exposant et la mantisse..

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- A la fin des années 70, chaque ordinateur avait sa propre représentation pour les nombres à virgule flottante.
- **Problème** : différentes manières de représenter e (exposant) et m (mantisse)
- **Solution** : Il y a donc eu la nécessité de normaliser le codage des nombres flottants (Normalize floating-point encoding). → **Normalisation**

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

- **Virgule flottante:**

Définition de la Normalisation :

La **normalisation** est un ensemble de techniques qui ont pour but de définir les produits et/ou les méthodes de fabrication aptes à satisfaire des besoins spécifiés ; **standardisation**.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante ieee 754

- Le standard le plus utiliser pour la représentation en virgule flottante est le standard **IEEE 754**.
- **IEEE** signifie «*Institute of Electrical and Electronics Engineers* » qui est une association professionnelle qui a pour mission promouvoir la connaissance de l'ingénierie liée à l'électricité et l'électronique y compris les télécommunications et l'informatique.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- **IEEE** établit des règles qu'ils faut suivre pour représenter les nombres réels (**real number**) afin que les ordinateurs les comprennent de manière uniforme
- Cette norme à été identifié par le numéro 754, d'où le nom **IEEE 754**.
- Contrairement à la virgule fixe(**fixed_point**) , la virgule flottante (**floating-point**) peut se déplacer à droite ou à gauche (**Shift right and left**)et sa position est compensée par la valeur de l'exposant qui change par conséquent.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS) virgule flottante IEEE 754

- **IEEE 754** décrit de nombreux formats pour représenter un nombre à virgule flottante, les plus connus sont :
- **Simple Précision** Single Precision (32 bits)
- **Double Précision** Double Precision (64 bits)

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- **Simple Précision** Single Precision (32 bits):
Le nombre binaire résultat est codé sur 32 bits



Utilisé pour le type "float" (simple précision)

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- **Simple Précision :**
- **Bit de signe** (The sign bit) : peut avoir soit la valeur 1 si le nombre est négatif , ou bien 0 si le nombre est positif.
- **L'exposant** (The exponent) codé sur 8 bits et contient la valeur de la puissance qui élève le nombre 2.
- **La mantisse** (mantissa) codé sur 23 bits et contient la partie fractionnaire (fraction part) du nombre écrit avec la notation scientifique.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS) virgule flottante IEEE 754

- Simple Précision :

Remarque :

Le chiffre 1 de la partie entière est implicite donc non inclus la représentation du nombre avec la norme IEEE 754.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- Simple Précision :
- Exemple 1 :
 - $(12.625)_{10} = (1100.101)_2$ (représentation réelle en virgule fixe) = 1.100101×2^3 (représentation scientifique) Il faut toujours laisser un 1 dans la partie entière (integer part).

0 00000011 100101000000000000000000

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- **Simple Précision :**
- Si on veut par exemple représenter : 1.1×2^{-2}
- On remarque que l'exposant est négatif.
- Alors comment faire pour représenter un exposant négatif sur 8 bits.
- On aurait pu le coder en complément à deux mais la norme **IEEE 754** propose une méthode différente et toute simple.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

Exemple 1:

- L'exposant est codé sur 8 bits donc la valeur minimale est 00000000 (ce qui vaut 0 en décimal) et la valeur maximale est 11111111 (ce qui vaut 255 en base 10) donc pas de nombres négatifs.
- L'idée consiste à diviser l'intervalle en 2, tous les nombres qui se trouvent droites sont considérés positifs et ceux qui se trouvent à gauche sont considérés négatifs.
- La valeur du milieu est calculé par la formule suivante :
 $D = 2^{n-1} - 1$ (avec n le nombre de bits de l'exposant)

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- Exemple 1:

L'exposant est codé sur 8 bits

$$D = 2^{n-1} - 1$$

Décalage = 127

0000 0000	-	+	1111 1111
(0) ₁₀			(255) ₁₀

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

•Exemple 1:

0 ~~00000011~~ 100101000000000000000000

0 10000010 100101000000000000000000

$$3 + 127 = 130$$

Donc, l'exposant décalé vaut 130

0100 0001 0100 1010 0000 0000 0000 0000

$(414A0000)_{16}$

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

Exemple 2:

- 0,75 en simple précision

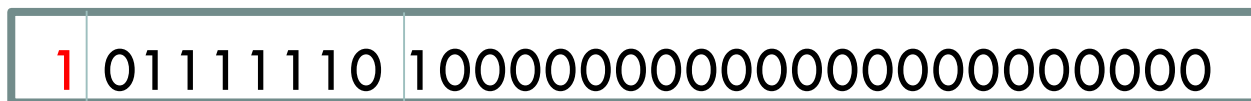
$$0,75 \times 2 = 1,5 ; 0,5 \times 2 = 1,0$$

- $0,75 \rightarrow 0,11$

- soit $1,1 \times 2^{-1}$ en forme normalisée

$$\Rightarrow (-1)^S \times (1 + .1000\ 0000\ 0000\ 0000\ 0000\ 0000) \times 2^{-1 + 127}$$

En simple précision :



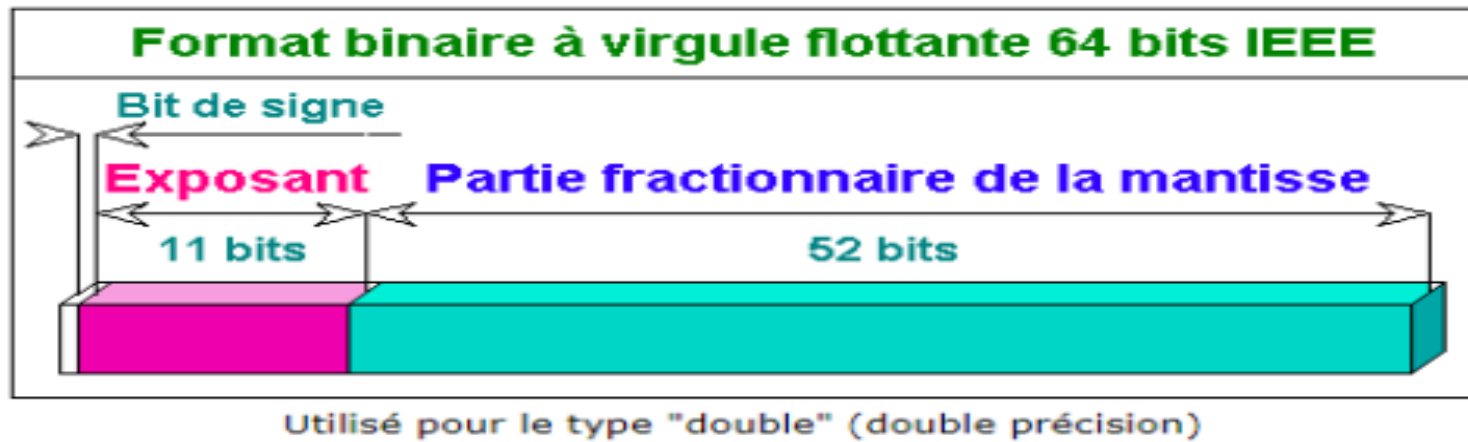
$-1 + 127 = 126$

mantisse

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- **Double précision** : Le nombre binaire résultat est codé sur 64 bits



REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- **Double précision :**
- **Bit de signe :** peut avoir soit la valeur 1 si le nombre est négatif , ou bien 0 si le nombre est positif.
- **L'exposant** codé sur 11 bits et contient la valeur de la puissance qui élève le nombre 2.
- **La mantisse** codé sur 52 bits et contient la partie fractionnaire du nombre écrit avec la notation scientifique.

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- Double précision :
- Exemple 1 :

$$(12.625)_{10} = (1100.101)_2 = 1.100101 \times 2^3$$

$D = 2^{n-1} - 1$ (donc le décalage vaut 1023 car $n=11$)



REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- Double précision :

- Exemple 1 :

$$(12.625)_{10} = 1.100101 \times 2^3$$

0 10000000010 1001010000...0000000

64 bits

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- Double précision :
- Exemple 1 :
- $(12.625)_{10} = 1.100101 \times 2^3$

0100 0000 0010 1001 0100 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000

$(4029400000000000)_{16}$

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- Double précision :
- Exemple 2 :
 - $(-12.625)_{10} = (1100.101)_2 = -1.100101 \times 2^3$

1100 0000 0010 1001 0100 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000

(C02940000000000000)16

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

	Simple précision sur 32 Bits	Double précision Sur 64 bits
Bit de signe	1	1
Exposant	8	11
Mantisse	23	52
Codage de l'exposant	Décalage 127	Décalage 1023
Variation de l'exposant	-126 à + 127	-1022 à + 1023
Plus petit nombre	2^{-126}	2^{-1022}
Plus grand nombre	2^{+127}	2^{+1023}

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

Addition

Elle se fait en trois étapes :

- Dénormaliser les deux nombres afin d'avoir le même exposant (le plus élevé)
- Additionner les mantisses
- Normaliser le résultat

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

Exemple : $7 - 2,25 = 4,75$

$(+7)_{10} = (0 \text{ 10000001 } 110000000000000000000000)_{\text{IEEE754}}$

129

$(-2,25)_{10} = (1 \text{ 10000000 } 001000000000000000000000)_{\text{IEEE754}}$

128

- **Dénormalisation :** $(+7)_{10} = + 1,11 * 2^2$

$$(-2,25)_{10} = -1,001 * 2^1 = 0,1001 * 2^2$$

- **Addition des mantisse**

$$(+1,11) + (-0,1001) = (+0001,1100) + (-0000,1001)$$

$$\begin{aligned} (0 \text{ 0001,1100})_{c2} + (1 \text{ 1111,0111})_{c2} &= (0 \text{ 0001,0011})_{c2} \\ &= (0 \text{ 0001,0011})_2 \end{aligned}$$

REPRÉSENTATION DES NOMBRES RÉELS (REPRESENTATION OF REAL NUMBERS)

virgule flottante IEEE 754

- Normalisation

Signe positif $\longrightarrow SM=0$

Mantisse = 1,0011 $\longrightarrow M=0011$

$E_p=2 \longrightarrow E_D=2+127=129=(10000001)_2$

Donc le résultat est:

(0 10000001 001100000000000000000000)IEEE754

REPRÉSENTATION DE L'INFORMATION

Le codage binaire : représente sous forme de 0 et 1 les informations.

- Nombre , texte,.....
- Images, son, vidéo,....
- Programmes.....

On a plusieurs modes de représentation :

- Code binaire pur.
- Code binaire BCD (Binary Coded Decimal).
- Code binaire Réfléchi (ou de GRAY)

CODAGE BINAIRE BCD

BINARY CODED DECIMAL (BCD)

Le codage **binaire pur** ou **binaire naturel** consiste à une conversion traditionnelle d'un nombre vers le système binaire sur un nombre de bits fixé.

Exemple : $(37)_{10}$ sur 8 bits = $(00100101)_2$

sur 16 bits = $(0000000000100101)_2$

sur 4 bits = ERREUR (Overflow)

Pour un nombre composé de plusieurs chiffres : sa représentation binaire dépend de ce nombre (pas des chiffres qui le composent)

CODAGE BINAIRE BCD

BINARY CODED DECIMAL (BCD)

Le code binaire BCD (**Binary Coded Decimal** ou **décimal codé binaire**) est un code qui conserve les avantages du système Décimal et du code binaire.

- Il est utilisé par les machines à calculer.
- Le code BCD consiste à une conversion des chiffres qui composent un nombre un par un sur 4 bits, le résultat est le regroupement des codes binaires.
- Le BCD est très utilisé en électronique pour l'affichage des chiffres.

CODAGE BINAIRE BCD

BINARY CODED DECIMAL (BCD)

Exemple:

Le nombre **512** en décimal équivaut à $(10000000000)_2$. En code BCD ce nombre sera codé comme suit **(0101 0001 0010)_{BCD}** La conversion de code est établie comme suit :

- Au premier chiffre (qui est 2) correspond sur quatre bits la valeur $(0010)_2$
- Au second chiffre (qui est 1) correspond sur quatre bits la valeur $(0001)_2$.
- Au dernier chiffre (qui est 5) correspond sur quatre bits la valeur $(0101)_2$
- Le code BCD du nombre 512 sera la concaténation dans le même ordre des chiffres décimaux de leur correspondants en binaire, c'est à dire **(0101 0001 0010)_{BCD}**

CODAGE BINAIRE BCD

BINARY CODED DECIMAL (BCD)

Chiffre décimal	Code BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Chiffre décimal	Code BCD
/	1010
/	1011
/	1100
/	1101
/	1110
/	1111

Non
utilisés

CODAGE BINAIRE BCD

BINARY CODED DECIMAL (BCD)

- **Addition en BCD :**

Comme tout autre système de numération dans l'arithmétique l'addition BCD est une opération qui peut être nécessaire.

BCD est un code numérique comportant plusieurs règles pour l'addition.

Ces règles sont données ci-dessous :

CODAGE BINAIRE BCD

BINARY CODED DECIMAL (BCD)

Addition en BCD :

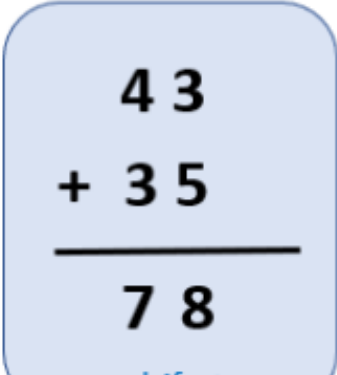
Règles

- Si une somme de 4 bits est ≤ 9 , le résultat est un nombre BCD.
- Si une somme de 4 bits est > 9 , ou une retenue est créée à partir d'un groupe de 4 bits, le résultat est non valide. Il faut additionner 6 (0110) au groupe de 4 bits qui causé l'erreur.

CODAGE BINAIRE BCD BINARY CODED DECIMAL (BCD)

- **Addition en BCD : Exemple 1** : faite l'opération suivante en BCD : 43 + 35

$$\begin{array}{r} 0100 \ 0011 \\ + 0011 \ 0101 \\ \hline 0111 \ 1000 \end{array}$$


$$\begin{array}{r} 43 \\ + 35 \\ \hline 78 \end{array}$$

CODAGE BINAIRE BCD

BINARY CODED DECIMAL (BCD)

- **Addition en BCD :**

L'opération de l'addition se fait en deux étapes :

- Calculer individuellement la somme de chaque paire de groupe de quatre bits et sans porter la retenue au prochains groupe de bits.
- Corriger (de droit à gauche) chaque groupe de quatre bits dépassant le 1001 par ajout de 110, si le groupe dégage une retenue elle sera envoyée au prochain groupe.

Exemple 2 : 39+58

	0011	1001
+	0101	1000
=	1000	10001
+	1	0110
=	1001	0111

CODAGE BINAIRE BCD

BINARY CODED DECIMAL (BCD)

- **soustraction en BCD :**

L'opération de la soustraction se fait en deux étapes :

- Calculer individuellement (de droit à gauche) la soustraction de chaque paire de groupe de quatre bits, emprunter un '1' du groupe suivant si l'opération nécessite un emprunt.
- Corriger (de droit à gauche) chaque groupe de quatre bits dépassant le 1001 par soustraction de 110.

Exemple 3 : 145-118

$$\begin{array}{r}
 \begin{array}{ccc}
 & & 1 \\
 0001 & 0100 & 0101 \\
 - & 0001 & 0001 & 1000 \\
 \hline
 0000 & 0010 & 1101 \\
 - & & 0110 \\
 \hline
 0000 & 0010 & 0111
 \end{array}
 \end{array}$$

CODAGE BINAIRE BCD

BINARY CODED DECIMAL (BCD)

Addition en BCD :

Exercice : faites les opérations suivante en BCD :

$$51 + 58$$

- $75 + 35$

- $98 + 97$

- $20 - 8$

- $14 + 89$

- $18 + 8.$

CODAGE BINAIRE RÉFLÉCHI (CODE DE GRAY)

- Le codage binaire réfléchi, également connu sous le nom de **code de Gray**.

Un système de codage binaire dans lequel deux chiffres consécutifs ne diffèrent que d'un seul bit.

Le code de Gray est utilisé dans divers domaines, y compris les communications, l'électronique et la logique.

- **Avantage** : le cas des capteurs, codeurs

CODAGE BINAIRE RÉFLÉCHI (CODE DE GRAY)

- **Construction du code Gray**

Ce processus de création d'un code de Gray en utilisant une méthode de réflexion par miroir(**using the mirror reflection method**) est une autre façon de générer le code de Gray.

- commençant par les valeurs 0 et 1 (**starting with the values 0 and 1**)

puis en suivant ces étapes:

CODAGE BINAIRE RÉFLÉCHI (CODE DE GRAY)

Étape 1 :

0 est codé 0

- 1 est codé 1

Étape 2 (ajout d'un bit) :

- Symétriser les nombres existants (Mirror the existing numbers) et ajouter 1 au début des nouveaux nombres :

- 0 devient 00

- 1 devient 01

Étape 3 (ajout d'un autre bit) :

- Symétriser les nombres existants (Mirror the existing numbers) et ajouter 1 au début des nouveaux nombres :

- 00 devient 000

- 01 devient 001

- 10 devient 011

- 11 devient 010

CODAGE BINAIRE RÉFLÉCHI (CODE DE GRAY)

- **Construction du code Gray** : Commençons par un exemple simple pour les 4 premiers chiffres décimaux 0 à 3.
- En code Gray, pour passer d'une ligne à la suivante, **on inverse un seul bit à la fois**.
- Le choix se fait sur le bit le plus à droite possible conduisant à un nouveau nombre. Ce qui donne les combinaisons suivantes :

CODAGE BINAIRE RÉFLÉCHI (CODE DE GRAY)

• Construction du code Gray :

0		0	0	
1		0	1	on inverse
2		1	1	1 bits à la fois
3		1	0	le plus à droite possible

Construction du code Gray : nous avons inversé 1 bits à la fois.

CODAGE BINAIRE RÉFLÉCHI (CODE DE GRAY)

- **Construction du code Gray :**

Recommençons avec les valeur décimales de 0 à 7.

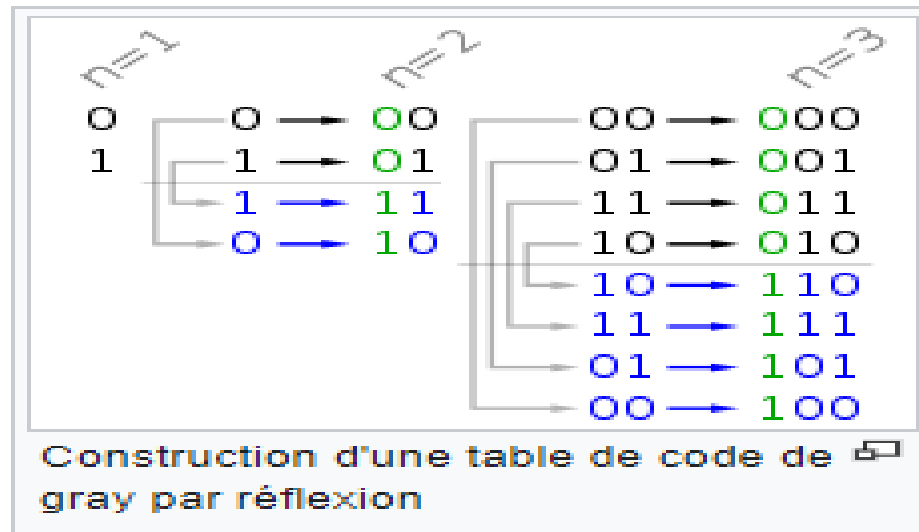
Nous avons besoin de 3 bits pour représenter 8 valeur binaires.

-

0 .00	0 .00	0 000
1 .01	1 .01	1 001
2 .11	2 .11	2 011
3 .10	3 .10	3 010
4 ...	4 .10	4 110
5 ...	5 .11	5 111
6 ...	6 .01	6 101
7 ...	7 .00	7 100

CODAGE BINAIRE RÉFLÉCHI (CODE DE GRAY)

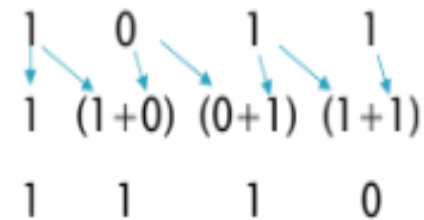
- Construction du code Gray :



CODAGE BINAIRE RÉFLÉCHI (CODE DE GRAY)

- **Conversion binaire pur vers binaire réfléchi :**
- Le principe consiste à prendre le bit le plus à gauche et le mettre dans sa position correspondante, puis les bits suivants sont une addition entre le bit précédent avec le suivant du binaire pur, si le résultat est 10 (1+1) on met 0
- **Exemple :**

$$(1011)_2 = (?)_{GR} = (1110)_{GR}$$

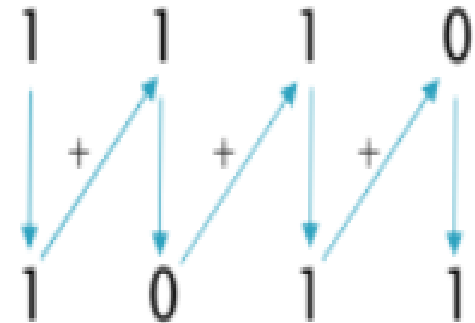


CODAGE BINAIRE RÉFLÉCHI (CODE DE GRAY)

- **Conversion binaire réfléchi vers binaire pur :**

Le principe consiste à prendre le bit le plus à gauche et le mettre dans sa position correspondante, puis les bits suivants sont une addition entre le bit précédent du binaire pur (résultat) avec le suivant du binaire réfléchi, si le résultat est 10 (1+1) on met 0

- **Exemple :** $(1110)_{\text{GR}} = (?)_2 = (1011)_2$



CODAGE DES CARACTÈRES

- **Exemple :**

l'information (1 23) peut être traduite :

$$(1\ 23)_{10} = (1\ 1\ 1\ 1\ 0\ 1\ 1)_2$$

- «1 23» comme texte $\neq (1\ 1\ 1\ 1\ 0\ 1\ 1)_2$
- Pour représenter les textes (caractères) on se base sur les normes internationales.

Il existe plusieurs variantes de normes pour représenter les caractères en binaire, les plus connus :

- Le codage ASCII
- Le codage EBCDIC
- Le codage Unicode

CODAGE DES CARACTÈRES : TABLE ASCII

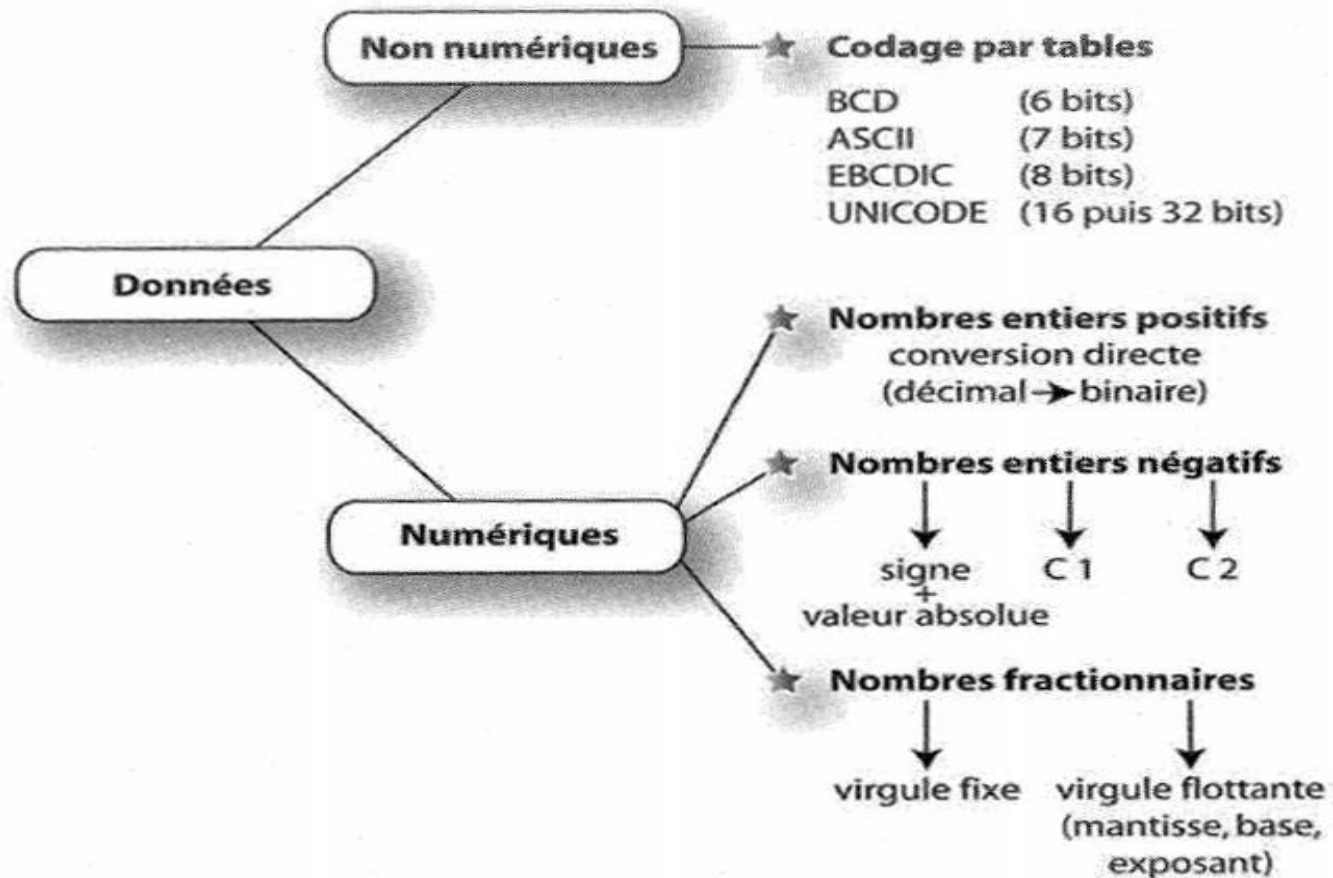
- Le code ASCII Autrement dit : **American Standard Code for Information Interchange**.

C'est un codage sur 7 bits ce qui donne 128 codes différents :

- 0 à 31 : caractères de contrôle (retour à la ligne, Bip sonore, etc....)
- 65 à 90 : majuscules
- 97 à 122 : minuscules
- Le reste des codes représente les caractères spéciaux.
- Dans ce code, les caractères accentués ne sont pas représentés et c'est la raison pour laquelle il est limité uniquement à quelques langues (anglais notamment).

CODAGE DES CARACTÈRES : TABLE ASCII

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



CONCLUSION

Dans ce chapitre nous avons vu les notions suivantes :

- Codage des informations
- Systèmes de numération (2, 10, 8 et 16).
- Conversions entre les différents systèmes de numération.
- Opérations binaires (+, -, *, /).
- Représentation des nombres fractionnaires et conversions.
- Représentations des nombres signés (BS, CA1, CA2) et addition en CA1 et addition en CA2.
- Représentation des nombres réels (virgule fixe, virgule flottante : norme IEEE 754)
- Codage binaire pure, codage BCD et codage de Gray.