

Chapitre 2

La représentation des informations en ordinateur

Module : Introduction à l'informatique

**1^{ère} ANNEE LICENCE D'EDUCATION D'ENSEIGNEMENT SECONDAIRE :
MATHEMATIQUES (LEESM)**


Ghizlane MOUKHLISS

Plan du Chapitre

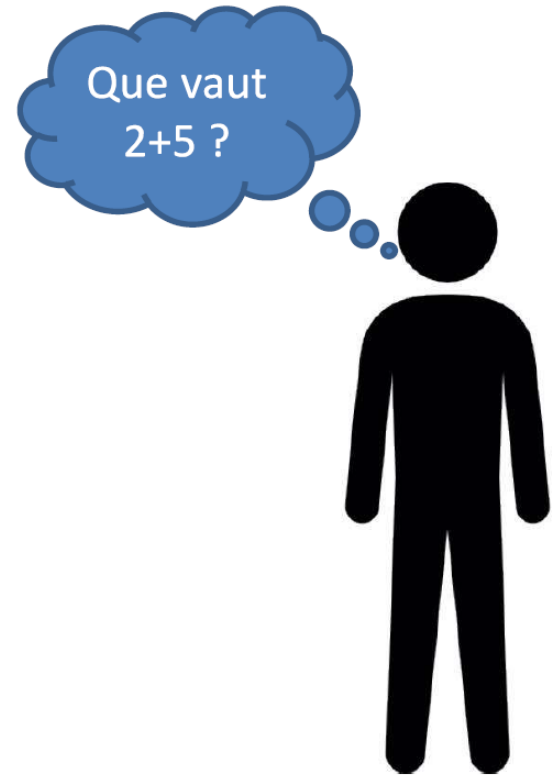
1. Introduction
2. Systèmes de numérotation et codage des nombres
3. La représentation des entiers naturels (\mathbb{N})
4. La représentation des entiers signés (\mathbb{Z})
5. Les opérations arithmétiques en binaire
6. La représentation des nombres fractionnaires (\mathbb{Q}/\mathbb{R})
7. Représentation des caractères
8. Autres types des données

Introduction

- Aujourd'hui les utilisateurs peuvent profiter de plusieurs types de services tels que : internet, jeux, application de gestion, application de stock, etc.
- En général, quelque soit le type de service, ce dernier contient les informations suivantes :
 - texte ;
 - image ;
 - vidéo ;
 - son ;
- En programmation, le mot information désigne les données et les instructions.
- Les données peuvent être classifiées en deux groupes :
 1. Numériques : entiers naturels, entiers relatifs et nombres réels.
 2. Non numériques : caractères('a', 'b' ..., 'z', ' ;', ' :', ' ?', ' ,', '@', '\$', ..., '0', '1', ..., '9') et chaîne de caractères ("ENS", "Maroc 2023").

 L'objectif de ce chapitre est maîtriser la représentation de chaque type de données à l'intérieure de l'ordinateur.

Langage de l'ordinateur



Systèmes de numérotation et codage des nombres

Définition du codage

- Le codage est une opération qui consiste à associer une représentation *externe* de l'information (comprise par l'humain) à une autre dite *interne* (compréhensible par la machine).
- Le codage de l'information s'effectue principalement en trois étapes :
 1. Déterminer le code utilisé ;
 2. Le nombre de bits ; (bit : 0 ou 1)
 3. Etat physique : il signifie l'interprétation réelle de 0 et 1, et il dépend du support physique.

Systèmes de numérotation et codage des nombres

Codage d'un bit par un état physique

- Dans la carte mère, un bit est un courant électrique :
 - 5V ===== > 1
 - 0V ===== > 0
- Disque dur, un bit mémoire est donné par la magnétisation.
 - Polarisation nord ===== > 1
 - Polarisation sud ===== > 0
- Dans la mémoire vive RAM un bit est un circuit électronique contenant condensateur-transistor : chargé (bit 1) ou non chargé (bit 0)

Exemple : Le codage de 12

- code : système binaire
- nombre de bits : 4 bits
- support physique : carte mère

Systèmes de numérotation et codage des nombres

Système de numération

- Un système de numération décrit la façon avec laquelle les nombres sont représentés.
- Un système de numération est défini par :
 1. Un alphabet A : ensemble de symboles ou chiffres ;
 2. Des règles d'écritures des nombres : Juxtaposition de symboles.

Définition du transcodage

- C'est l'opération qui permet de passer d'une représentation d'un nombre exprimé dans une base vers une autre représentation du même nombre exprimé dans une autre base.
- Exemple :
conversion de décimal (10) vers le binaire (2)
conversion de l'octal (8) vers le binaire (2)

Le système décimal

Principe

- Base={10}
- Chiffres={0,1,2,3,4,5,6,7,8,9}
- Règle : pour représenter un nombre $X \in \mathbb{N}$ dans le système décimal, il suffit de l'écrire sous la forme :

$$X = \sum_{i=0}^n a_i 10^i = (a_n a_{n-1} \dots a_i \dots a_1 a_0)_{10}$$

Le système décimal

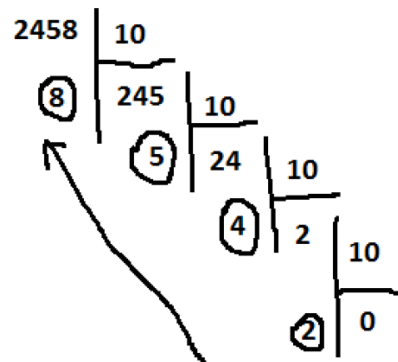
Principe

- Exemple : coder le nombre 2458 dans le système décimal.

1. Solution1 : on écrit 2458 comme combinaison linéaire de puissance de 10

$$\begin{aligned} 2458 &= 2000 + 400 + 50 + 8 \\ &= 2 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 8 \times 10^0 = (2458)_{10} \end{aligned}$$

2. Solution 2 : On applique la division euclidienne de 2458 par 10



Donc $2458 = (2458)_{10}$

Le système binaire

Principe

- Base={2}
- Chiffres={0,1}
- Règle : pour représenter un nombre $X \in \mathbb{N}$ dans le système binaire, il suffit de l'écrire sous la forme :

$$X = \sum_{i=0}^n a_i 2^i = (a_n a_{n-1} \dots a_i \dots a_1 a_0)_2$$

avec $a_i \in \{0; 1\}$

- a_n est appelé bit du poids fort.
- a_0 est appelé bit du poids faible.

Le système binaire

- Exemple 1 :

On écrit 17 comme combinaison linéaire de puissance de 2

$$17 = 16 + 1$$

$$= 1 \times 2^4 + 1 \times 2^0$$

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

$$= (10001)_2$$

On a utilisé 5 bits pour coder le nombre 17 en binaire.

Remarque

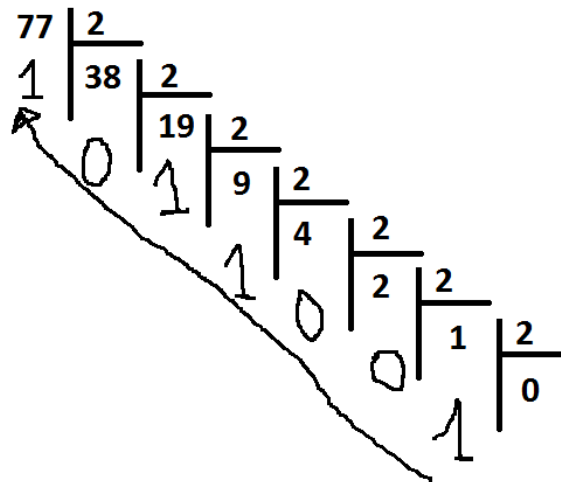
Pour coder le même nombre 17 en binaire sur 8 bits, il suffit d'insérer trois 0 non significatifs au début de la suite binaire.

$$17 = (00010001)_2$$

Le système binaire

- Exemple 2 : On applique l'algorithme de la division euclidienne par 2
- Principe de l'algorithme : on divise par 2 le nombre N puis on divise par 2 les quotients successifs jusqu'à ce que le quotient soit nul. La suite des restes est alors l'écriture binaire de N.

On représente 77 en binaire



Donc $77 = (1001101)_2$

Conversion du binaire vers le décimal

Principe

Pour le passage du binaire vers le décimal, on additionne les puissances de 2 dont la valeur du bit (symbole) est différent de 0.

Exemple 1 :

Que vaut $N = (1101)_2$ en base 10 ?

$$\begin{aligned} 1101_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 8 + 4 + 0 + 1 \\ &= 13 \end{aligned}$$

Exemple 2 :

Que vaut $N = (01001110)_2$ en base 10 ?

$$\begin{aligned} 01001110_2 &= 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 0 + 64 + 0 + 0 + 8 + 4 + 2 + 0 \\ &= 78 \end{aligned}$$

Le système octal

Principe

- Base={8}
- Chiffres={0,1,3,4,5,6,7}
- Règle : pour représenter un nombre $X \in \mathbb{N}$ dans le système octal, il suffit de l'écrire sous la forme :

$$X = \sum_{i=0}^n a_i 8^i = (a_n a_{n-1} \dots a_i \dots a_1 a_0)_8$$

avec $a_i \in \{0,1,3,4,5,6,7\}$

Le système octal

- Exemple : coder les deux nombres 17 et 92 dans le système octal.

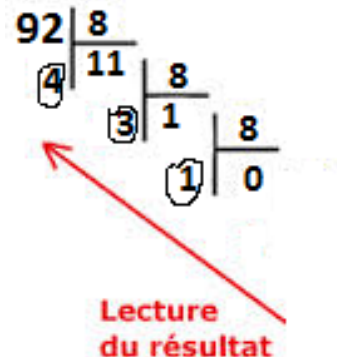
1. Méthode 1 : on écrit 17 comme combinaison linéaire de puissance de 8

$$17 = 16 + 1$$

$$17 = 2 \times 8^1 + 1 \times 8^0$$

$$17 = (21)_8$$

2. Méthode 2 : On applique la division euclidienne de 92 par 8



Donc $92 = 134_8$

Conversion de l'octal vers le décimal

Principe

Pour le passage de l'octal vers le décimal, on additionne les puissances de 8 dont la valeur du symbole est différent de 0.

Exemple 1 :

Que vaut $N = (31)_8$ en base 10 ?

$$31_8 = 3 \times 8^1 + 1 \times 8^0$$

$$= 24 + 1$$

$$= 25$$

Exemple 2 :

Que vaut $N = (106)_8$ en base 10 ?

$$106_8 = 1 \times 8^2 + 0 \times 8^1 + 6 \times 8^0$$

$$= 64 + 0 + 6$$

$$= 70$$

Le système hexadécimal

Principe

- Base={ 16 }
- Chiffres={ 0,1,3,4,5,6,7,8,9,A,B,C,D,E,F }
- Règle : un nombre $X \in \mathbb{N}$ dans le système hexadécimal, il suffit de l'écrire sous la forme :

$$X = \sum_{i=0}^n a_i 16^i = (a_n a_{n-1} \dots a_i \dots a_1 a_0)_{16}$$

avec $a_i \in \{0,1,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

- Ce système est utilisé pour l'adressage des cases mémoires ou aussi pour le codage des couleurs.
- Il facilite la représentation d'une séquence trop longue en binaire.

Le système hexadécimal

- Exemple : coder les deux nombres 17 et 92 dans le système hexadécimal.

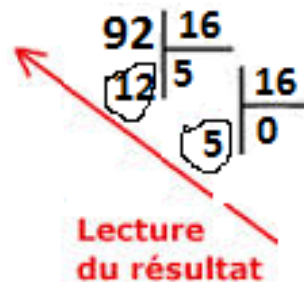
1. Méthode 1 : on écrit 17 comme combinaison linéaire de puissance de 16

$$17 = 16 + 1$$

$$17 = 1 \times 16^1 + 1 \times 16^0$$

$$17 = (11)_{16}$$

2. Méthode 2 : On utilise la division euclidienne de 92 par 16



Donc $92 = 5C_{16}$ ($C \leftrightarrow 12$)

Conversion de l'hexadécimal vers le décimal

Principe

Pour le passage de l'hexadécimal vers le décimal, on additionne les puissances de 16 dont la valeur du symbole est différent de 0.

Exemple 1 :

Que vaut $N = (24)_{16}$ en base 10 ?

$$24_{16} = 2 \times 16^1 + 6 \times 16^0$$

$$= 32 + 6$$

$$= 38$$

Exemple 2 :

Que vaut $N = (F16)_{16}$ en base 10 ?

$$F16_{16} = F \times 16^2 + 1 \times 16^1 + 6 \times 16^0$$

$$= 15 \times 256 + 16 + 6$$

$$= 3862$$

Conversion du binaire vers l'octal (hexadécimal)

Principe

Pour convertir une suite binaire (base 2) dans le système octal (resp hexadécimal), d'abord on décompose cette suite en blocks de 3 bits (4 bits) de droite à gauche, ensuite, on donne la correspondance de chaque bloc en décimal.

Exemple 1 :

Coder en octal puis en hexadécimal la suite binaire $(1101)_2$

$$1101 = 001101 = \underbrace{001}_{1} \underbrace{101}_{5} = (15)_8$$

$$1101 = \underbrace{1101}_{13} = (D)_{16}$$

Exemple 2 :

Coder en octal puis en hexadécimal la suite binaire $(1001010)_2$

$$1001010 = 001001010 = \underbrace{001}_{1} \underbrace{001}_{1} \underbrace{010}_{2} = (112)_8$$

$$1001010 = 01001010 = \underbrace{0100}_{4} \underbrace{1010}_{A} = (4A)_{16}$$

Conversion de l'octal (hexadécimal) vers binaire

Principe

Pour le passage du système octal (resp hexadécimal) vers le système binaire, il suffit de représenter chaque symbole de l'octal (hexadécimal) en binaire sur 3 bits (resp 4 bits).

Exemple 1 :

Convertir en binaire les valeurs suivants 13_8 et 207_8

$$13_8 = 001001_2$$

$$207_8 = 010000111_2$$

Exemple 2 :

Convertir en binaire les valeurs suivants 15_{16} et $A10_{16}$

$$15_{16} = 00010101_2$$

$$A10_{16} = 1010\ 0001\ 0000_2$$

Remarque

Pour faire le transcodage d'une façon rapide, il suffit de connaître le tableau suivant :

Nombre	sur 3 bits	sur 4 bits
0	000	0000
1	001	0001
2	010	0010
3	011	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	impossible	1000
9	impossible	1001
10	impossible	1010
11	impossible	1011
12	impossible	1100
13	impossible	1101
14	impossible	1110
15	impossible	1111

Les opérations arithmétiques : Addition

L'addition binaire

- L'addition binaire est l'opération la plus importante des circuits numériques car les autres opérations comme la soustraction, la division et la multiplication en découlent.
- Les circuits numériques ne permettent pas d'additionner plus de deux nombres binaires à la fois.
- Les additions sont, en fait, une succession d'additions de deux nombres : le résultat de l'addition du premier et du deuxième est ajouté au troisième et ainsi de suite.

Les opérations arithmétiques : Addition

Principe

L'addition binaire classique est analogue à l'addition décimale → Il faut commencer par le bit de poids faible en utilisant les relations suivantes :

+	0	1
0	0	1
1	1	0

⇒ $1 + 1 = 0$ avec un report de 1

Exemple : Calculer en binaire $3 + 5$

On a $3 = 011_2$ et $5 = 101_2$

$$\begin{array}{r} 3 \\ + 5 \\ \hline 8 \end{array} \qquad \begin{array}{r} 011 \\ + 101 \\ \hline 1000 \end{array}$$

Les opérations arithmétiques : Addition

Remarques

- Additionner deux nombre égaux revient à multiplier par $(10)_2$ en base 2, donc à lui ajouter un 0.
- L'addition peut faciliter la conversion d'une base vers la base décimale en décomposant le nombre à convertir.

Exemple 1 :

Calculer en binaire $1001 + 1001$

On a $1 + 1 = 10$

Donc $1001 + 1001 = 10010$

Exemple 2 :

$(14A6)_{16}$ est la somme de $(1000)_{16} + (400)_{16} + (A0)_{16} + (6)_{16}$.

$$\Rightarrow (14A6)_{16} = 1 \times 16^3 + 4 \times 16^2 + 10 \times 16^1 + 6$$

$$\Rightarrow (14A6)_{16} = 4096 + 1024 + 160 + 6 = (5286)_{10}$$

Les opérations arithmétiques : Multiplication

Principe

La multiplication binaire est analogue à la multiplication décimale en utilisant les relations suivantes :

x	0	1
0	0	0
1	0	1

Exemple : Calculer en binaire 3×5

$$\begin{array}{r} 3 \\ \times 5 \\ \hline = 15 \end{array}$$

$$\begin{array}{r} 011 \\ \times 101 \\ \hline = 011 \\ = 000- \\ = 011-- \\ = 1111 \end{array}$$

Les opérations arithmétiques : Soustraction

Principe

La soustraction en binaire s'effectue de la même manière qu'en décimal.

—	0	1
0	0	1
1	1	0

$\Rightarrow 0 - 1 = 1$ et j'emprunte 1

Exemple : Calculer en binaire $91 - 47$

On a $91 = 1011011_2$ et $47 = 101111_2$

$$\begin{array}{r} 1011011 \\ - 101111 \\ \hline = 0101100 \end{array}$$

Les opérations arithmétiques : Division

Principe

La division binaire s'effectue à l'aide de soustractions et de décalages, comme la division décimale, sauf que les digits du quotient ne peuvent être que 1 ou 0. Le bit du quotient est 1 si on peut soustraire le diviseur, sinon il est 0.

Exemple : Calculer en binaire $201/16$

On a $201 = 11001001_2$ et $16 = 10000_2$

$\begin{array}{r} 11001001 \\ - 10000 \\ \hline 010010 \\ - 10000 \\ \hline 000100 \\ - 000 \\ \hline 1001 \\ - 0000 \\ \hline 1001 \end{array}$	$\begin{array}{r} 10000 \\ \hline 1100 \end{array}$
--	---

reste → 1001

Le résultat est correct car $201/16 = 12$ et $12 = 1100_2$

Dépassement de capacité (overflow)

Exercice Calculer en binaire sur 8 bits $200 + 100$

$$200 = (11001000)_2$$

$$100 = (01100100)_2$$

Remarques :

- On peut pas calculer $200 + 100$ sur 8 bits car on obtient un dépassement de capacité
- Sur 8 bits, le plus grand nombre entier qu'on peut coder est $2^8 - 1 = 255$
- Sur 32 bits, le plus grand nombre entier acceptable par le processeur est $2^{32} - 1 = 4294967295$
- Sur 64 bits, le plus grand nombre entier acceptable par le processeur est $2^{64} - 1 = 184467440737095$

La représentation des entiers signés (\mathbb{Z})

Définition

- On appelle un nombre signé (ou relatif) toute valeur qui peut être soit positive ou négative.
- Exemple : +33 ; +13330 ; -19 ; etc
- Il existe 3 codes pour représenter les nombres signés :
 1. Signe + valeur absolue (s+va)
 2. Complément à 1 (Ca1)
 3. Complément à 2 (Ca2)

Le code signe + valeur absolue (s+va)

Principe

Dans ce code le premier bit est réservé pour le signe. Si le nombre est positif le bit=0, si le nombre est négatif le bit=1. Ensuite, la partie de la valeur absolue est codée de la même façon que les entiers naturels.

Exemple 1 : conversion de décimal vers S+VA

−5=1101_{s+va}, on a le bit de signe est 1 et $|5| = 101_2$

−23 en s+va sur 8 bits : on a $23 = (10111)_2$

−23=110111_{s+va} sur 6 bits

−23=10010111_{s+va} sur 8 bits

+23 en S + VA sur 8 bits : +23=010111_{s+va}=00010111_{s+va}

Exemple 2 : conversion de S+VA vers le système décimal

1100_{s+va}=−4 car on a le bit de signe= 1 et $100_2 = 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 4$

011010_{s+va}=+26 car on a le bit de signe= 0 et $11010_2 = 2^4 + 2^3 + 2^1 = 26$

Remarque :

Le code S+VA possède deux inconvénients :

- le 0 possède deux représentations (+0 et -0).
- Les opérations arithmétiques ne sont pas corrects.

Le code complément à 1 (Ca1)

Principe

Pour convertir un entier signé X dans ce code, d'abord on représente X dans le code $S+VA$, ensuite on garde le bit de signe sans modification et on inverse les autres bits.

Exemple 1 : conversion de décimal vers Ca1

$$-5 = \mathbf{1101}_{s+va} = \mathbf{1010}_{Ca1}$$

-23 en Ca1 sur 8 bits

on a $23 = (10111)_2$

$$-23 = \mathbf{110111}_{s+va} = \mathbf{10010111}_{s+va} = \mathbf{11101000}_{Ca1}$$

$$+23 \text{ en Ca1 sur 8 bits : } +23 = \mathbf{00010111}_{s+va} = \mathbf{00010111}_{Ca1}$$

Exemple 2 : conversion de Ca1 vers le système décimal

$$\mathbf{1001}_{Ca1} = \mathbf{1110}_{s+va} = -6$$

$$\mathbf{1110010}_{Ca1} = \mathbf{1001101}_{s+va} = -13$$

$$\mathbf{01010}_{Ca1} = \mathbf{01010}_{s+va} = +10$$

Remarque :

Le code Ca1 possède les mêmes inconvénients que le code $S + VA$

Le code complément à 2 (Ca2)

Principe

Il existe deux méthodes pour coder un entier signé X dans ce code

1. Méthode 1 : on applique la règle suivante : $Ca2 = Ca1 + 1$
2. Méthode 2 : pour coder X en $Ca2$ sur N bits, on applique la règle suivante :
 - Si $X < 0$ alors les N bits représentent le code de $2^N + X$ en base 2
 - Si $X \geq 0$ alors les N bits représentent le code de X en base 2

Exemple 1 : conversion de décimal vers $Ca2$

Que vaut -5 en code $Ca2$ sur 4 bits ?

méthode 1 : $-5 = 1101_{s+va} = 1010_{ca1} = 1011_{ca2}$

méthode 2 : on code $2^4 - 5$ en base 2 $\rightarrow 2^4 - 5 = 16 - 5 = 11 = 1011_{ca2}$

Exemple 2 : conversion de $Ca2$ vers le système décimal

Que vaut la suite binaire 10111101_{ca2} en base 10 ?

$10111101_{ca2} = 10111100_{ca1} = 11000011_{s+va} = -67$

Remarque :

Dans le code $Ca2$, le 0 possède une représentation unique et les opérations arithmétiques sont correctes.

Les opérations arithmétiques en binaire

Exercice 1: Calculer en binaire sur 8 bits $14 - 9$

Nombre	S+VA	Ca1	Ca2
14			
-9			
= 5			
décimal			

Exercice 2: Calculer en binaire sur un octet $-6 * 45$

Nombre	S+VA	Ca1	Ca2
-6			
45			
=			
décimal			

La représentations des nombres fractionnaires (\mathbb{R})

Définition

- Un nombre fractionnaire est un nombre qui contient une partie décimale inférieure à 1.
- Un nombre fractionnaire est représenté en décimal sous la forme :

$$d = \sum_{i=-n}^m d_i 10^i = (d_m d_{m-1} \dots d_1 d_0 . d_{-1} d_{-2} \dots d_{-n})$$

- On distingue deux types :
 1. Nombre en virgule fixe : est une valeur qui contient toujours le même nombre de chiffres avant et après la virgule.
Exemple : 1/4, 10.5, 123.25, ...
 2. Nombre en virgule flottante : est un nombre dans lequel, à chaque fois on fixe un signifiant m entier, on peut obtenir plusieurs nombres avec le même signifiant, mais pour lesquels la virgule sera placée à différents endroits.
Exemple : 1/3(0.33333), 109.2, 123.52, ...

Représentation en binaire d'un nombre en virgule fixe

Principe

Pour coder en binaire un nombre en virgule fixe on applique la règle suivante :

- La partie entière est codée de la même façon que les entiers (naturels ou signés).
- Tandis que le codage partie décimale se fait par multiplication successives par 2 jusqu'à obtenir une partie décimale nulle.

Exemple 1 : conversion de décimal vers virgule fixe

Que vaut 5,25 et 37,625 en binaire ?

$$5 = 101_2$$

$$0,25 \times 2 = 0,5 = 0 + 0,5$$

$$0,5 \times 2 = 1,0 = 1 + 0,0$$

$$0,25 \times 2 = 0,5 = 0 + 0,25$$

$$\text{Donc } 5,25 = 101,01_2$$

$$37 = 32 + 4 + 1 = 2^5 + 2^2 + 2^0 = 100101_2$$

$$0,625 \times 2 = 1,25 = 1 + 0,25$$

$$0,25 \times 2 = 0,5 = 0 + 0,5$$

$$0,5 \times 2 = 1 + 0,0$$

$$\text{Donc } 37,625 = 100101,101_2$$

Exemple 2 : conversion de virgule fixe vers le système décimal

Que vaut la suite binaire $1010,11_2$ en base 10 ?

$$1010,11_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 8 + 2 + 0,5 + 0,25 = 10,75$$

Codage en binaire d'un nombre en virgule flottante (\mathbb{R})

Principe

- La représentation en binaire, d'un nombre en virgule flottante est inspirée de l'écriture scientifique :

$$\forall x \in \mathbb{R} \quad x = (-1)^s m B^E$$

- s : le signe du nombre (si x est positif $s=1$ sinon $s=0$) ;
- m : c'est la mantisse (càd partie significative du nombre) ;
- B : représente la base utilisée ;
- E : c'est l'exposant effectif.
- Le codage d'un nombre en virgule flottante se fait d'une manière approximative en machine.
- Selon la norme IEEE 754, il existe trois formats pour coder un nombre réel :
 1. Simple précision : on utilise 32 bits.
 2. Double précision : on utilise 64 bits.
 3. Etendu : on utilise 80 bits.

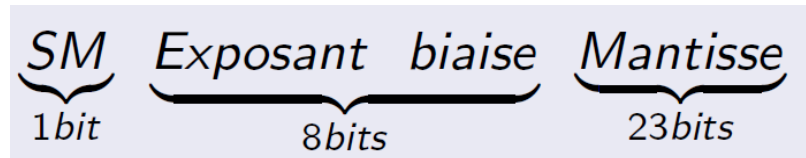
Codage d'un nombre réel en simple précision

Principe

- Le codage d'un nombre en simple précision sur 32 bits se fait de la façon suivante :
 - 1 bit de signe de la mantisse
 - 8 bits pour l'exposant biaisé (décalé)
 - 23 bits pour la mantisse
- Le calcul de l'exposant biaisé (Eb) est donné par :

$$E_b = \text{biais} + E = 2^7 + E$$

- La mantisse doit être normalisée c'est-à-dire que le premier bit après la virgule est 1.



Remarque :

Pour le codage en double précision on a :

The diagram shows the 64-bit layout of a double-precision floating-point number. It is divided into three sections: a 1-bit sign (SM), an 11-bit biased exponent (Exposant biaisé), and a 52-bit mantissa (Mantisse). Each section is represented by a bracket with its label above and its bit count below.

Codage d'un nombre réel en simple précision

Principe

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

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

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

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

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

Pb : différentes manières de représenter E et M

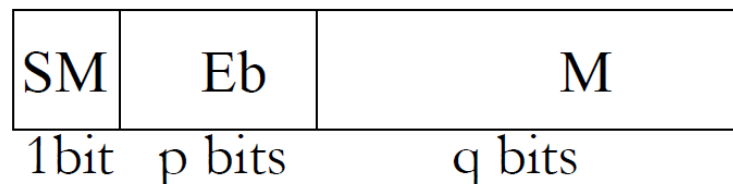
→ Normalisation

Codage d'un nombre réel en simple précision

Normalisation

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

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



Codage d'un nombre réel en simple précision

Normalisation

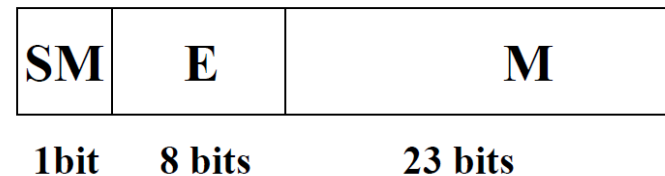
Standard IEEE 754 (1985)

Simple précision sur 32 bits :

1 bit de signe de la mantisse

8 bits pour l'exposant

23 bits pour la mantisse

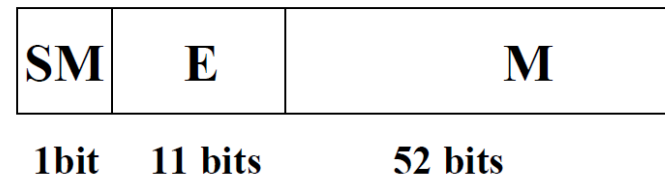


Double précision sur 64 bits :

1 bit de signe de la mantisse

11 bits pour l'exposant

52 bits pour la mantisse



Codage d'un nombre réel en simple précision

Exemple 1 : Coder en simple précision 14,25

$$14 = 1110_2$$

$$0,25 = 0,01_2$$

$$14,25 = 1110,01_2$$

$$14,25 = (-1)^0 \times 1,111001 \times 2^3$$

$$\text{Exposant biaisé} = 2^7 + 3 = 10000011_2$$

$$\underbrace{0}_{SM} \quad \underbrace{1000011}_{\text{Exposant}} \quad \underbrace{111001...00}_{\text{biaise mantisse}}$$

Exemple 2 : Coder en simple précision -48,2

$$48 = 110000_2$$

$$0,2 \times 2 = 0,4 = 0 + 0,4$$

$$0,4 \times 2 = 0,8 = 0 + 0,8$$

$$0,8 \times 2 = 1,6 = 1 + 0,6$$

$$0,6 \times 2 = 1,2 = 1 + 0,2$$

on revient à l'état initial

$$-48,2 = (-1)^1 \times 110000,00110011....$$

$$-48,2 = (-1)^1 \times 1,1000000110011... \times 2^5$$

$$Eb = 2^7 + 5 = 10000101_2$$

$$\underbrace{1}_{SM} \quad \underbrace{10000101}_{\text{Exposant}} \quad \underbrace{1000000110011...0011}_{\text{biaise mantisse}}$$

Exemple 3 :

Quelle est la valeur réelle de 1 10000101 111101110000000000000000

La réponse est -125,75

Représentation des caractères/chaîne de caractères

Définition d'un caractère/chaîne de caractère

- On appelle caractère tout élément de l'ensemble :
 - **Alphabet** : 'a', 'b', 'c', ..., 'z' et 'A', 'B', ..., 'Z'
 - **Signe de ponctuation** : ',', ';', ':', '!', '?'
 - **Caractère spécial** : '/', '@', '\$', ...
 - **Chiffres** : '0', '1', ..., '9'
- On appelle chaîne de caractères toute suite de caractères.

Codage d'un caractère/chaîne de caractère

- Le codage d'un caractère (jeux de caractères) consiste à attribuer une valeur numérique unique pour ce caractère, ensuite coder ce caractère en binaire.
- Il existe plusieurs codes : ACSII, ISO 8859-1, Unicode, UTF-8, UTF-16, ...
- La représentation d'une chaîne de caractères selon un jeu de caractères donné consiste à coder chaque caractère de la chaîne en utilisant ce jeu de caractères.

Les différents codes pour représenter un caractère

Le code ASCII (American Standard Code for Information Interchange)

- Dans ce code, chaque caractère est codé sur 7 bits son inconvénient est qu'on ne peut pas coder les caractères accentués : é, à, â, ...

Exemples

Caractère ----- > ASCII

'A' ----- > 65

'B' ----- > 66

'C' ----- > 67

Caractère ----- > ASCII

'a' ----- > 97

'b' ----- > 98

'0' ----- > 48

Le code ISO 8859-1 (ASCII Étendu)

Dans ce code, chaque caractère est codé sur 8 bits. De plus, il permet de coder les caractères accentués : é, à, â,...

Autre jeux de caractères

- Unicode** : Le code UNICODE permet de représenter tous les caractères spécifiques aux différentes langues.
- UTF-8** : est un codage de caractères informatiques conçu pour coder l'ensemble des caractères sur 8 bits.
- UTF-6** : est un codage étendu de UTF-8 sur 16 bits.

Représentation des données logiques

Définition d'une donnée logique

- Une donnée logique représente une expression qui peut avoir valeur soit vraie (True) soit fausse (False).

Exemple

- $5 > 2$ ----- $> \text{True}$
- $10 == 20$ ----- $> \text{False}$
- $10! = 20$ and $2 \leq 100$ ----- $> \text{True}$
- $'A' < 'B'$ or « sami » == « amine » ----- $> \text{True}$

Codage d'une donnée logique

Le codage d'une donnée logique (booléenne) nécessite un seul bit

1 ----- $> \text{True}$
0 ----- $> \text{False}$

Le code BCD (Binary Coded Decimal)

Définition

- Le binary coded decimal (BCD), qui peut se traduire en français par décimal codé binaire, est un système de numération utilisé en électronique et en informatique pour coder des nombres d'une façon relativement proche de la représentation humaine usuelle (en base 10).
- Le BCD est un code dans lequel chaque chiffre d'un nombre décimal est codé en binaire sur 4 bits.
- Ces chiffres peuvent être représentés sur un octet individuel, c'est le BCD non compacte.

Exemple

$$(327)_{10} = (000000110000001000000111)_{\text{BCD}}$$

La représentation interne des images en noir et blanc

Définition

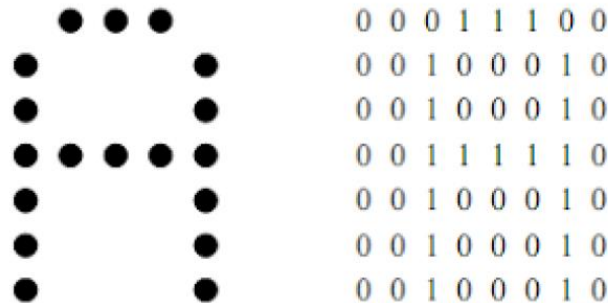
- Les images sont formées de pixels (abréviation de picture element).
- Pixel : le plus petit point que l'on peut distinguer dans une image.
- Pour les images en noir et blanc, un pixel est soit noir soit blanc.



La représentation interne des images en noir et blanc

Principe de codage

- Pixel noir → état 1
- Pixel blanc → état 0
- En noir et blanc chaque pixel est codé sur un bit.
- L'image présentée possède 50 pixels sur chaque ligne et 50 pixels sur chaque colonne.
- On peut la représenter par une matrice 50x50 dont chaque élément a soit la valeur 0 soit la valeur 1.



La représentation interne des images en noir et blanc

Traitement d'images

Une fois l'image codée en binaire, l'ordinateur peut faire tous les calculs et les transformations demandés sur la matrice qui la représente.

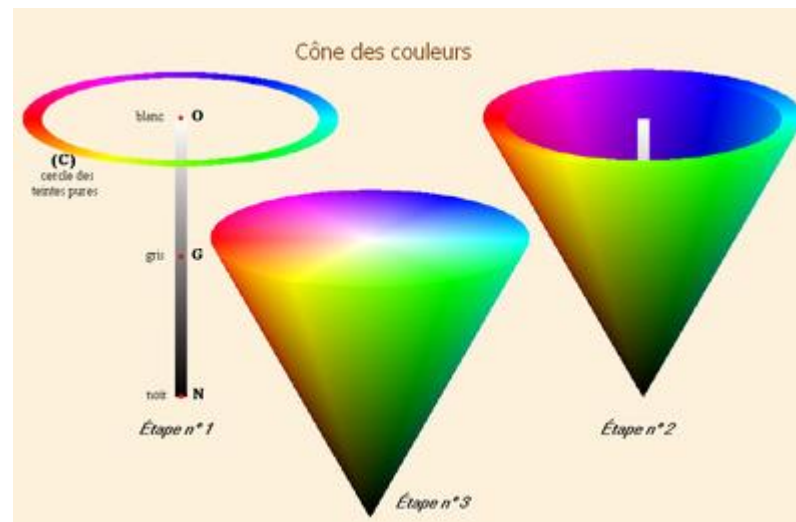
Exemple

Pour obtenir le négatif de cette image l'ordinateur inverse chaque élément de matrice. (l'inverse de 0 est 1, celui de 1 est 0).

La représentation interne des images en couleurs

Définition

- Le codage informatique des couleurs est l'ensemble des conventions permettant l'affichage ou l'impression par un périphérique informatique d'une image en couleurs, plutôt qu'en noir et blanc. Le codage se base sur la synthèse additive trichrome des couleurs.
- Au niveau du périphérique, l'image est toujours matricielle, et on définit, pour chaque pixel, un triplet de valeurs : rouge, vert et bleu

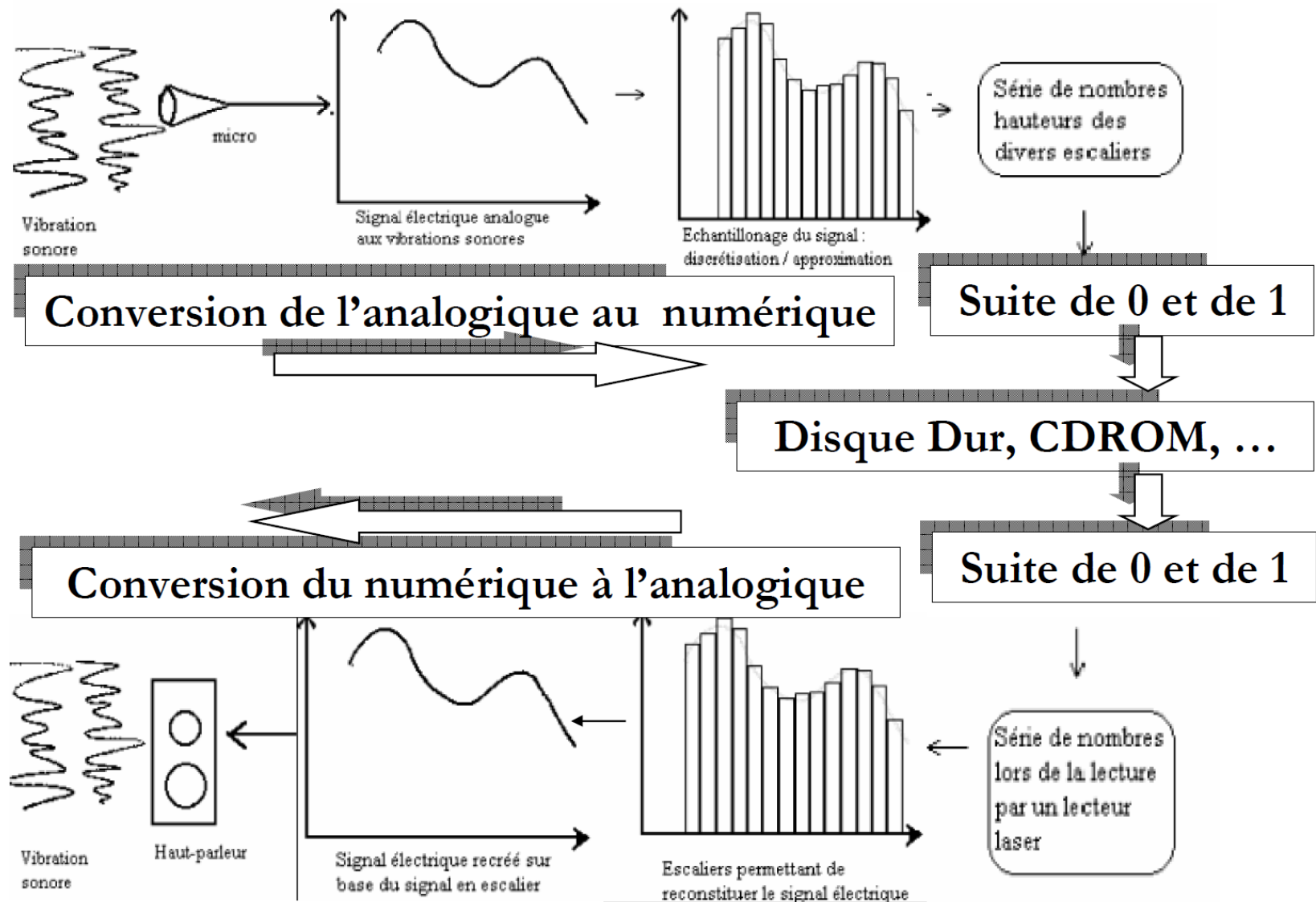


La représentation interne de son

Codage de son

- **échantillonnage** = relever des échantillons de son à intervalles de temps régulier.
- Plus la fréquence d'échantillonnage est élevée, plus l'onde sonore originale est reproduite fidèlement (meilleure est la qualité du son).
- Ex : CD : fréquence de 44100 Hz (sons audibles), 8000 Hz = téléphone
- La qualité d'un son numérisé dépend également du nombre de bits utilisés pour coder chaque échantillon (quantification)
 - WAV et AIFF : formats courants non compressés.
 - MP3 : Compression avec perte : élimine les fréquences inaudibles pour la plupart des auditeurs dans des conditions normales d'écoute RAM, WMA, OGG, . . .
 - MIDI : pas le signal sonore qui est codé mais les notes qui lui correspondent.

La représentation interne de son



La représentation interne de vidéo

Codage de la vidéo

- Afficher une succession d'images par secondes appelées trames afin de produire l'illusion du mouvement.
- Vidéo numérique = images numérisées.
- Vidéo non compressée = 30 Mo/s (plus de 100 Go/H).
- Algorithmes de compression (codecs) : applications de compression (encodeurs) et de décompression (lecteurs multimédia)
- Ex : Lecteur VLC : lit la majorité des codecs
- MPEG : code les changements d'une image à l'autre
 - DivX (basé sur la norme MPEG) : encoder la bande son au format MP3.
 - Les vidéos MPEG : extension .avi, .mpg, .mpeg ou .divx.
 - MOV, RM, WMV : utilisent des codecs propriétaires.
 - Flash permet de coder des animations sous forme vectorielle.