Codage de l'information

Plan

- **Introduction**
- **Systèmes de numération et Représentation des nombres**
  - **Systèmes de numération**
  - **Système de numération décimale**
  - **Représentation dans une base b**
  - **Représentation binaire, Octale et Hexadécimale**
  - **Transcodage ou changement de base**
- **Codage des nombres**
  - **Codage des entiers positifs (binaire pur)**
  - **Codage des entiers relatifs (complément à 2)**
  - **Codage des nombres réels (virgule flottante)**
- **Codage des caractères :**
  - **ASCII et**
  - **ASCII étendu,**
  - **Unicode , …**
- **Codage du son et des images**

Codage d’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**

Codage d’information : -Définition-

- **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 règle précise.

- **Exemple :**

***** Le nombre 35 : 35 est la représentation externe du nombre trente cinq

* La représentation interne de 35 sera une suite de 0 et 1 (100011)

Codage d'information (suite)

- En informatique, Le codage de l'information s'effectue principalement en trois étapes :
  - L'information sera exprimée par une suite de nombres (Numérisation)
  - Chaque nombre est codé sous forme binaire (suite de 0 et 1)
  - <u>Chaque élément binaire est représenté par un état physique</u>

(Elément binaire → Etat physique)

- Codage de l'élément binaire par un état physique
  - Charge électrique (RAM : Condensateur-transistor) : Chargé (bit 1) ou non chargé (bit 0)
  - Magnétisation (Disque dur, disquette) : polarisation Nord (bit 1) ou Sud (bit 0)
  - Alvéoles (CDROM): réflexion (bit 1) ou pas de réflexion (bit 0)
  - Fréquences (Modem) : dans un signal sinusoïdal
    - Fréquence f_1 (bit 1) : $s(t) = a \sin(2\pi f_1 t + \psi)$
    - Fréquence f_2 (bit 0) : $s(t) = a \sin(2\pi f_2 t + \psi)$
  - ….

Système de numération

- Système de numération décrit la façon avec laquelle les nombres sont représentés.

- Un système de numération est défini par :
  - Un alphabet $\mathcal{A}$: ensemble de symboles ou chiffres,
  - Des règles d'écritures des nombres : Juxtaposition de symboles

Exemples de Système de numération (1)

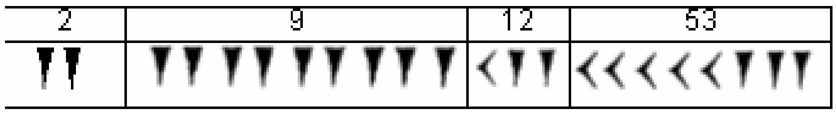
- Numération Romaine

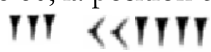
système romain	I	V	X	L	C	D	M
valeur décimal	1	5	10	50	100	500	1000

  - Lorsqu'un symbole est placé à la droite d'un symbole plus fort que lui, sa valeur s'ajoute : CCLXXI → 271
  - Lorsqu'un symbole est placé à la gauche d'un symbole plus fort que lui, on retranche sa valeur : CCXLIII → 243
  - On ne place jamais 4 symboles identique à la suite : 9 s'écrit IX et non VIIII
  - Le plus grand nombre exprimable est : 3999 (MMMCMXCIX)
  - Système inadapté au calcul

Exemples de Système de numération (2)

- **Numération babylonienne**
  - Chez les Babyloniens (environ 2000 ans avant J.C.), les symboles utilisés sont le clou pour l'unité et le chevron pour les dizaines. C'est un système de position.



  - A partir de 60, la position des symboles entre en jeu :
    - 204 : 
    - 7392 : 
  - Le nombre 60 constitue la base de ce système.

Exemples de Système de numération (3)

- **Numération décimale :**
  - **C'est le système de numération le plus pratiqué actuellement.**
  - **L'alphabet est composé de dix chiffres :**

A = {0,1,2,3,4,5,6,7,8,9}

  - **Le nombre 10 est la base de cette numération**
  - **C'est un système positionnel. Chaque position possède un poids.**
  - **Par exemple, le nombre 4134 s'écrit comme :**

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

Système de numération positionnel pondéré à base b

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

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

- **Soit N = $a_{n-1}\, a_{n-2} \ldots a_1\, a_{0\,(b)}$: 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**

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

$$N = a_{n-1}.b^{n-1} + a_{n-2}.b^{n-2} + \ldots + a_0.b^0{}_{(10)} = \sum_{i=0}^{n-1} a_i b^i$$

Bases de numération (Binaire, Octale et Hexadécimale)

- **Système binaire (b=2) utilise deux chiffres :** {0,1}
  - C'est avec ce système que fonctionnent les ordinateurs

- **Système Octale (b=8) utilise huit chiffres :**{0,1,2,3,4,5,6,7}
  - Utilisé il y a un certain temps en Informatique.
  - Elle permet de coder <u>3 bits</u> par un seul symbole.

- **Système Hexadécimale (b=16) utilise 16 chiffres :**

$\{0,1,2,3,4,5,6,7,8,9,A=10_{(10)},B=11_{(10)},C=12_{(10)},D=13_{(10)},E=14_{(10)},F=15_{(10)}\}$

  - Cette base est très utilisée dans le monde de la micro informatique.
  - Elle permet de coder <u>4 bits</u> par un seul symbole.

Transcodage (ou conversion de base)

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

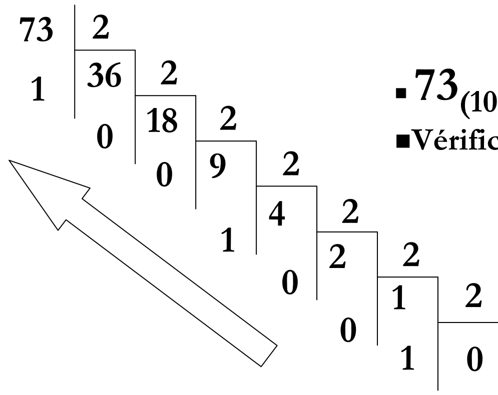
- **Par la suite, on verra les conversions suivantes:**
  - **<u>Décimale vers Binaire, Octale et Hexadécimale</u>**
  - **<u>Binaire vers Décimale, Octale et Hexadécimale</u>**

Changement de base de la base 10 vers une base b

- **La règle à suivre est les divisions successives :**
  - **On divise le nombre par la base b**
  - **Puis le quotient par la base b**
  - **Ainsi de suite jusqu'à l'obtention d'un quotient nul**
  - **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.**

Exemple : décimale vers binaire

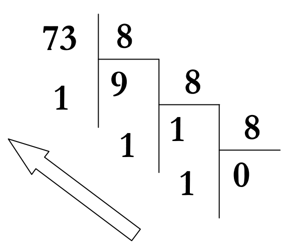
- Soit N le nombre d'étudiants d'une classe représenté en base décimale par : $N = 73_{(10)}$
- Représentation en Binaire?



- **$73_{(10)} = 1001001_{(2)}$**
- **Vérification**

Exemple : décimale vers octale

- Soit N le nombre d'étudiants d'une classe représenté en base décimale par : $N = 73_{(10)}$
- Représentation en Octale?



- **$73_{(10)} = 111_{(8)}$**
- **Vérification**

Exemple : décimale vers Hexadécimale

- Soit N le nombre d'étudiants d'une classe représenté en base décimale par : $N = 73_{(10)}$
- Représentation en Hexadécimale?

$$
\begin{array}{r|l}
73 & 16 \\ \hline
9 & 4 \;\; | \; 16 \\
 & 4 \;\; | \; 0
\end{array}
$$

- **$73_{(10)} = 49_{(16)}$**
- **Vérification**

de la base binaire vers une base b -Solution 1-

- **Première solution :**
  - **convertir le nombre en base binaire vers la base décimale puis convertir ce nombre en base 10 vers la base b.**
- **Exemple :**
  - **$10010_{(2)} = ?_{(8)}$**
  - **$10010_{(2)} = 2^4+2_{(10)}=18_{(10)}=2*8^1+2*8^0{}_{(10)}=22_{(8)}$**

de la base binaire vers une base b
-Solution 2-

- **Deuxième solution :**
  - **Binaire vers décimale : par définition ($\sum_{i=0}^{n-1} a_i b^i$)**
  - **Binaire vers octale : regroupement des bits en des sous ensembles de trois bits puis remplacer chaque groupe par le symbole correspondant dans la base 8.(<u>Table</u>)**
  - **Binaire vers Hexadécimale : regroupement des bits en des sous ensembles de quatre bits puis remplacer chaque groupe par le symbole correspondant dans la base 16.(<u>Table</u>)**

Correspondance Octale \Binaire

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

<u>Retour</u>

Correspondance Hexadécimale \Binaire

Hexadécimale\Binaire

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

Retour

Exemple : binaire vers décimale

- Soit N un nombre représenté en binaire par :
  $N = 1010011101_{(2)}$
- Représentation Décimale?

$N=1.2^9+0.2^8+1.2^7+0.2^6+0.2^5+1.2^4+1.2^3+1.2^2+0.2^1+1.2^0$

$=512 + 0 + 128 + 0 + 0 + 16 + 8 + 4 + 0 + 1$

$=669_{(10)}$

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

Exemple : binaire vers octale

- Soit N un nombre représenté en base binaire par :
  $N = 1010011101_{(2)}$
- Représentation Octale?

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

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

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

Exemple : binaire vers Hexadécimale

- Soit N un nombre représenté en base binaire par :
  $N = 1010011101_{(2)}$
- Représentation Hexadécimale?

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

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

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

Exercice

Décimal	Binaire	Hexadécimal	Octal
1	00000001	001	001
10			
	01100100		
		065	
			764

Plan

- Introduction
- Systèmes de numérotation et Codage des nombres
  - Systèmes de numérotation
  - Système de numération décimale
  - Représentation dans une base b
  - Représentation binaire, Octale et Hexadécimale
  - Transcodage ou changement de base
- *Codage des nombres*
  - *Codage des entiers positifs (binaire pur)*
  - *Codage des entiers relatifs (complément à 2)*
  - *Codage des nombres réels (virgule flottante)*
- Codage des caractères :
  - ASCII et
  - ASCII étendu,
  - Unicode , …
- Codage du son et des images

Codage des entiers naturels (1)

- **Utilisation du code binaire pur :**
  - **L'entier naturel (positif ou nul) est représenté en base 2,**
  - **Les bits sont rangés selon leur poids, on complète à gauche par des 0.**
  - **Exemple : s**ur un octet, **$10_{(10)}$** se code en binaire pur?

$0\,0\,0\,0\,1\,0\,1\,0_{(2)}$

Codage des entiers positifs (2)

- **Etendu du codage binaire pur :**
  - **Codage sur n bits : représentation des nombres de 0 à $2^n - 1$**
  - **sur 1 octet (8 bits): codage des nombres de 0 à 2^8 - 1 = 255**
  - **sur 2 octets (16 bits): codage des nombres de 0 à 2^{16} - 1 = 65535**
  - **sur 4 octets (32 bits) : codage des nombres de 0 à 2^{32} - 1 = 4 294 967 295**

Arithmétique en base 2

- **Les opérations sur les entiers s'appuient sur des tables d'addition et de multiplication :**

Addition

0	0	0
0	1	1
1	0	1
1	1	(1) 0

Retenu

Multiplication

0	0	0
0	1	0
1	0	0
1	1	1

Exemple (Addition)

- Addition binaire (8 bits)

```
    1 0 0 1 0 1 1 0
  + 0 1 0 1 0 1 0 1
  -----------------
    1 1 1 0 1 0 1 1
```

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

```
    1 0 0 1 0 1 1 0
  + 0 1 1 1 0 1 0 1
  -----------------
  1 0 0 0 0 1 0 1 1
```

overflow

Exemples

- Multiplication binaire

```
          1 0 1 1  (4 bits)
   *      1 0 1 0  (4 bits)
   ------------------
          0 0 0 0
        1 0 1 1  .
      0 0 0 0 .
    1 0 1 1  .
   ------------------
  0 1 1 0 1 1 1 0
```

Sur 4 bits le résultat est faux

Sur 7 bits le résultat est juste

Sur 8 bits on complète à gauche par un 0

Codage des entiers relatifs

- **Il existe au moins trois façons pour coder :**
  - **code binaire signé (*par signe et valeur absolue*)**
  - **code complément à 1**
  - **code complément à 2 (Utilisé sur ordinateur)**

Codage des nombres relatifs -Binaire signé-

- **Le bit le plus significatif est utilisé pour représenter le signe du nombre :**
    - si le bit le plus fort **= 1** alors **nombre négatif**
    - si le bit le plus fort **= 0** alors **nombre positif**
- **Les autres bits codent la valeur absolue du nombre**
- **Exemple : Sur 8 bits, codage des nombres -24 et -128 en (bs)**
    - **-24 est codé en binaire signé par : $1\,0\,0\,1\,1\,0\,0\,0_{(bs)}$**
    - **-128 hors limite → nécessite 9 bits au minimum**

Codage des nombres relatifs -Binaire signé- (suite)

- **Etendu de codage :**
    - **Avec n bits, on code tous les nombres entre**

      $-(2^{n-1}-1)$ **et** $(2^{n-1}-1)$
    - **Avec 4 bits : -7 et +7**
- **Limitations du binaire signé:**
    - **Deux représentations du zéro : + 0 et - 0**
    - **Sur 4 bits : $+0 = 0000_{(bs)}$, $-0 = 1000_{(bs)}$**
    - **Multiplication et l'addition sont moins évidentes.**

Binaire signé (Exercices)

- Coder 100 et -100 en binaire signé sur 8 bits

$100_{(10)} = (01100100)_{(bs)}$

$-100_{(10)} = (11100100)_{(bs)}$

- Décoder en décimal $(11000111)_{(bs)}$ et $(00001111)_{(bs)}$

$(11000111)_{(bs)} = -71_{(10)}$

$(00001111)_{(bs)} = 15_{(10)}$

- Calculer : 1– 2 en binaire signé sur 8 bits

Codage des entiers relatifs (code complément à 1)

- Aussi appelé Complément Logique (CL) ou Complément Restreint (CR) :
  - les nombres positifs sont codés de la même façon qu'en binaire pure.
  - un nombre négatif est codé en inversant chaque bit de la représentation de sa valeur absolue
- Le bit le plus significatif est utilisé pour représenter le signe du nombre :
  - si le bit le plus fort = 1 alors nombre négatif
  - si le bit le plus fort = 0 alors nombre positif

Codage des entiers relatifs -code complément à 1- (suite)

- Exemple : -24 en complément à 1 sur 8 bits
  - |-24| en binaire pur → 0 0 0 1 1 0 0 0 $_{(2)}$ puis
  - on inverse les bits → 1 1 1 0 0 1 1 1 $_{(cà1)}$
- Limitation :
  - deux codages différents pour 0 (+0 et -0)
  - Sur 8 bits : +0=0 0 0 0 0 0 0 0$_{(cà1)}$ et -0=1 1 1 1 1 1 1 1$_{(cà1)}$
  - Multiplication et l'addition sont moins évidentes.

Code Complément à 1 (Exercices)

- Coder 100 et -100 par complément à 1 (cà1) sur 8 bits

  $100_{(10)}$ = (01100100) $_{(cà1)}$

  $-100_{(10)}$ = (10011011) $_{(cà1)}$

- Décoder en décimal (11000111)$_{(cà1)}$ et (00001111)$_{(cà1)}$

  (11000111) $_{(cà1)}$ = $-56_{(10)}$

  (00001111) $_{(cà1)}$ = $15_{(10)}$

- Calculer : 1 – 2 en complément à 1 sur 8 bits

Codage des entiers relatifs -code complément à 2- (1)

- **Aussi appelé Complément Vrai (CV) :**
  - **les nombres positifs sont codés de la même manière qu'en binaire pure.**
  - **un nombre négatif est codé en ajoutant la valeur 1 à son complément à 1**
- **Le bit le plus significatif est utilisé pour représenter le signe du nombre**
- **Exemple : -24 en complément à 2 sur 8 bits**
  - **24 est codé par 0 0 0 1 1 0 0 0$_{(2)}$**
  - **-24 → 1 1 1 0 0 1 1 1$_{(cà1)}$**
  - **donc -24 est codé par 1 1 1 0 1 0 0 0$_{(cà2)}$**

Codage des entiers relatifs -code complément à 2- (2)

- **Un seul codage pour 0. Par exemple sur 8 bits :**
  - **+0 est codé par 00000000$_{(cà2)}$**
  - **-0 est codé par 11111111$_{(cà1)}$**
  - **Donc -0 sera représenté par 00000000$_{(cà2)}$**
- **Etendu de codage :**
  - **Avec n bits, on peut coder de $-(2^{n-1})$ à $(2^{n-1}-1)$**
  - **Sur 1 octet (8 bits), codage des nombres de -128 à 127**
    - **+0 = 00000000** **-0=00000000**
    - **+1 = 00000001** **-1=111111111**
    - **…** **…**
    - **+127= 01111111** **-128=10000000**

Code Complément à 2
-Exercices-

- Coder $100_{(10)}$ et $-100_{(10)}$ par complément à 2 sur 8 bits

  $100_{(10)} = 01100100_{(Cà2)}$

  $-100_{(10)} = 10011010_{(Cà2)}$

- Décoder en décimal $11001001_{(Cà2)}$ et $01101101_{(Cà2)}$

  $11001001_{(Cà2)} = -55_{(10)}$

  $01101101_{(Cà2)} = 109_{(10)}$

- Calculer : 1-2 en complément à 2 sur 8 bits

Exercices

- Quel est l'entendu de codage sur 6 et 9 bits :
  - Binaire pur, Binaire signé, complément à 2
- Quelle est la valeur décimale des suites binaires (1010, 10010110 et 1011010011101001), s'elles sont codées en :
  - binaire pur, Binaire signé, Complément à 1, Complément à 2
- Sur 4, 8 et 16 bits, coder les nombres +20 et -15 en :
  - Binaire pur, Binaire signé, Complément à 1, Complément à 2
- Calculer 20-15 sur 8 et 16 bits en :
  - Complément à 2

Codage des nombres Réels

Codage des nombres réels

- Les formats de représentations des nombres réels sont :
  - Format virgule fixe
    - utilisé par les premières machines
    - possède une partie 'entière' et une partie 'décimale' séparés par une virgule. La position de la virgule est fixe d'où le nom.
    - Exemple : $54{,}25_{(10)}$; $10{,}001_{(2)}$; $A1{,}F0B_{(16)}$
  - Format virgule flottante (utilisé actuellement sur machine)
    - défini par : $\pm\, m\,.\,b^{e}$
    - un signe + ou –
    - une mantisse m (en virgule fixe)
    - un exposant e (un entier relative)
    - une base b (2,8,10,16,…)
    - Exemple : $0{,}5425 \,.\, 10^{2}{}_{(10)}$; $10{,}1 \,.\, 2^{-1}{}_{(2)}$; $A0{,}B4.16^{-2}{}_{(16)}$
  - …

Codage en Virgule Fixe (1)

- **Etant donné une base b, un nombre x est représenté, en format virgule fixe, par :**
  - $x = a_{n-1}a_{n-2}\ldots a_1a_0,a_{-1}a_{-2}\ldots a_{-p\ (b)}$
  - a_{n-1} **est le chiffre de poids fort (MSB)**
  - a_{-p} **est le chiffre de poids faible (LSB)**
  - **n est le nombre de chiffre avant la virgule**
  - **p est le nombre de chiffre après la virgule**
  - **La valeur de x en base 10 est :** $x = \sum_{-p}^{n-1} a_i b^i{}_{(10)}$
  - **Exemple :**

    $101{,}01_{(2)}=1.2^2+0.2^1+1.2^0+0.2^{-1}+1.2^{-2} = 5{,}25_{(10)}$

Codage en Virgule Fixe (2)

Changement de base 10→2

- **Le passage de la base 10 à la base 2 est défini par :**
  - **Partie entière est codée sur p bits (division successive par 2)**
  - **Partie décimale est codée sur q bits en multipliant par 2 successivement jusqu'à ce que la partie décimale soit nulle ou le nombre de bits q est atteint.**
  - **Exemple :** $4{,}25_{(10)} = ?_{(2)}$ **format virgule fixe**
    - $4_{(10)} = 100_{(2)}$
    - $0{,}25 \times 2 = 0{,}5 \rightarrow 0$
    - $0{,}5 \times 2 = 1{,}0 \rightarrow 1$
    - **donc** $4{,}25_{(10)} = 100{,}01_{(2)}$
- **Exercice : Coder** $7{,}875_{(10)}$ **et** $5{,}3_{(10)}$ **avec** $p = 8$ **et** $q = 8$

Codage en Virgule Flottante

$$x = \pm M . 2^{E}$$

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)

$3{,}25_{(10)} = 11{,}01_{(2)}$ (en virgule fixe)

$= 1{,}101 . 2^{1}_{(2)}$

$= 110{,}1 . 2^{-1}_{(2)}$

Pb : différentes manières de représenter E et M

→ Normalisation

Codage en Virgule Flottante -Normalisation-

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

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

SM	Eb	M
1bit	p bits	q bits

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

SM	E	M
1bit	8 bits	23 bits

Double précision sur 64 bits :
- 1 bit de signe de la mantisse
- 11 bits pour l'exposant
- 52 bits pour la mantisse

SM	E	M
1bit	11 bits	52 bits

Conversion décimale - IEEE754 (Codage d'un réel)

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

Nombre positif, donc SM = 0

$35{,}5_{(10)} \quad = \quad 100011{,}1_{(2)}$ (virgule fixe)

$\quad = \quad 1{,}000111 \,.\, 2^5{}_{(2)}$ (virgule flottante)

Exposant = Eb-127 = 5, donc Eb = 132

1,M = 1,000111 donc M = 00011100...

$$\overbrace{0}^{\text{SM}}\underbrace{10000100}_{\text{Eb}}\overbrace{00011100000000000000000}^{\text{M}}{}_{\text{(IEEE 754 SP)}}$$

Conversion IEEE754 - Décimale (Evaluation d'un réel)

$$\overbrace{0}^{SM}\underbrace{10000001}_{Eb}\overbrace{11100000000000000000000}^{M}{}_{\text{(IEEE 754 SP)}}$$

S = 0, donc nombre positif

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

1,M = 1,111

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

Caractéristiques des nombres flottants au standard IEEE

	Simple précision	Double précision
Bit de signe	1	1
Bit d'exposant	8	11
Bit de mantisse	23	52
Nombre total de bits	32	64
Codage de 1'exposant	Excédant 127	Excédant 1023
Variation de 1'exposant	-126 à +127	-1022 à +1023
Plus petit nombre normalisé	2^{-126}	2^{-1022}
Plus grand nombre normalisé	**environ 2^{+128}**	**environ 2^{+1024}**

Codage des caractères

- **Caractères : Alphabétique (A-Z , a-z), numérique (0 ,…, 9), ponctuation, spéciaux (&, $, %,…) …etc.**
- **Données non numérique (addition n'a pas de sens)**
- **Comparaison ou tri → très utile**
- **Codage revient à créer une Table de correspondance entre les caractères et des nombres.**

Codage des caractères Les Standards (1)

- **Code (ou Table) ASCII (American Standard Code for Information Interchange)**
  - **7 bits pour représenter 128 caractères (0 à 127)**
  - **48 à 57 : chiffres dans l'ordre (0,1,…,9)**
  - **65 à 90 : les alphabets majuscules (A,…,Z)**
  - **97 à 122 : les alphabets minuscule (a,…z)**

Codage des caractères
Les Standards (2)

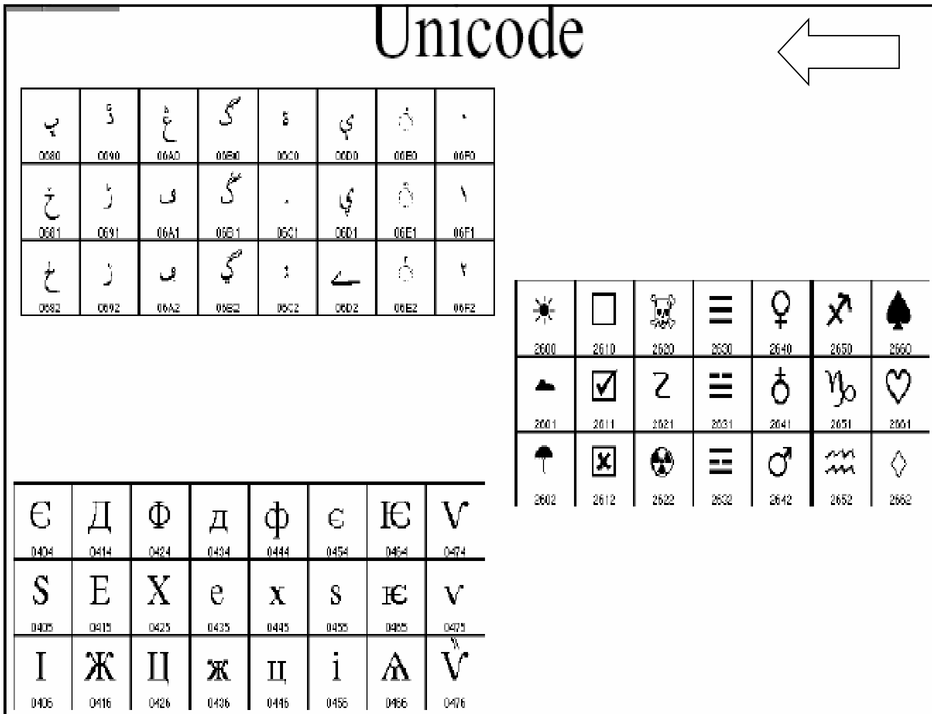
- Table <u>ASCII Etendu</u>
  - 8 bits pour représenter 256 caractères (0 à 255)
  - Code les caractères accentués : à, è,…etc.
  - Compatible avec ASCII

- Code <u>Unicode</u> (mis au point en 1991)
  - 16 bits pour représenter 65 536 caractères (0 à 65 535)
  - Compatible avec ASCII
  - Code la plupart des alphabets : Arabe, Chinois, ….
  - On en a défini environ 50 000 caractères pour l'instant

Code ASCII Etendu

DECIMAL VALUE	➡	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
⬇	HEXA DECIMAL VALUE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	BLANK (NULL)	►	SP	0	@	P	‘	p	Ç	É	á	░	└	╨	∝	≡
1	1	☺	◄	!	1	A	Q	a	q	ü	æ	í	▒	┴	╤	β	±
2	2	☻	↕	"	2	B	R	b	r	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	♥	‼	#	3	C	S	c	s	â	ô	ú	│	├	╙	π	≤
4	4	♦	¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	♣	§	%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	♠	▬	&	6	F	V	f	v	å	û	ª	╢	╞	╓	µ	÷
7	7	BEL	↨	’	7	G	W	g	w	ç	ù	º	╖	╟	╫	τ	≈
8	8	BS	↑	(	8	H	X	h	x	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	HT	↓	)	9	I	Y	i	y	ë	Ö	⌐	╣	╔	┘	Θ	•
10	A	LF	→	*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Ω	·
11	B	VT	←	+	;	K	[	k	{	ï	¢	½	╗	╦	█	δ	√
12	C	FF	FS	,	<	L	\	l	¦	î	£	¼	╝	╠	▄	∞	ⁿ
13	D	CR	GS	-	=	M	]	m	}	ì	¥	¡	╜	═	▌	φ	²
14	E	♪	RS	.	>	N	^	n	~	Ä	₧	«	╛	╬	▐	∈	■
15	F	☼	US	/	?	O	_	o	△	Å	ƒ	»	┐	╧	▀	∩	BLANK

Unicode



Ce ne sont que des bits !!!

01001001 01001110 01000110 01001111 01010010 01001101 01000001 01010100 01001001 01010001 01010101 01000101

caractères codés en ASCII Etendu (8 bits)	INFORMATIQUE
entiers codés en binaire pur sur 1 octets	73 ; 78 ; 70 ; 79 ; 82 ; 77 ; 65 ; 84 ; 73 ; 81 ; 85 ; 69 (base 10)
entiers codés en binaire pur sur 2 octets	18766 ; 17999 ; 21069 ; 16724 ; 18769 ; 21829 (base 10)
entiers codés en binaire pur sur 4 octets	1 229 866 575 ; 1 380 794 708 ; 1 230 067 013 (base 10)
nombres en flottant simple précision (32 bits)	+ (1,10011100100011001001111) . 2^{19} ; + (1,10011010100000101010100) . 2^{37} ; + (1,10100010101010101000101) . 2^{19} ; 844 900,9375; 220 391 079 936 ; 857 428,3125 (base 10)

Comment coder ce dessin sous forme de suite de nombres?

Principe du codage d'une image(1)

- **Tout commence par découper l'image en des petits carrés c'est en quelque sorte poser une grille (aussi serrée que possible) sur l'image.**
- **Deux nombres seront important pour décrire cette grille : le nombre de petits carrés en largeur et ce même nombre en hauteur**
- **Plus ces nombres sont élevés, plus la surface de chaque petit carré est petite et plus le dessin tramé sera proche de l'originale.**

On obtient donc pour toute l'image un quadrillage comme celui montré ci-dessous pour une partie

Principe du codage d'une image(2)

- Il ne reste plus qu'à en déduire une longue liste d'entiers :
  - Le nombre de carré sur la largeur
  - Le nombre de carré sur la hauteur
  - Suite de nombres pour coder l'information (Couleur) contenue dans chaque petit carré qu'on appelle pixel (PICture ELement) :
    - Image en noir et blanc → 1 bit pour chaque pixel
    - Image avec 256 couleur → 1 octet (8 bits) pour chaque pixel
    - Image en couleur vrai (True Color : 16 millions de couleurs) → 3 octets (24 bits) pour chaque pixel
- La manière de coder un dessin en série de nombres s'appelle une représentation BITMAP

Principe du codage d'une image(3) (Terminologie)

- **Infographie est le domaine de l'informatique concernant la création et la manipulation des images numériques.**
- **La définition : détermine le nombre de pixel constituant l'image. Une image possédant 800 pixels en largeur et 600 pixels en hauteur aura une définition notée 800x600 pixels.**
- **La profondeur ou la dynamique d'une image est le nombre de bits utilisé pour coder la couleur de chaque pixel.**
- **Le poids d'une image (exprimé en Ko ou en Mo) : est égal à son nombre de pixels (définition) que multiplie le poids de chacun des pixels (profondeur).**

Principe du codage du son