

# Architecture des ordinateurs

## Chapitre 1:

### Généralités sur l'architecture des ordinateurs et codage de l'information

# Chapitre 1 : Généralités sur l'ordinateur

## 1. Histoire des ordinateurs

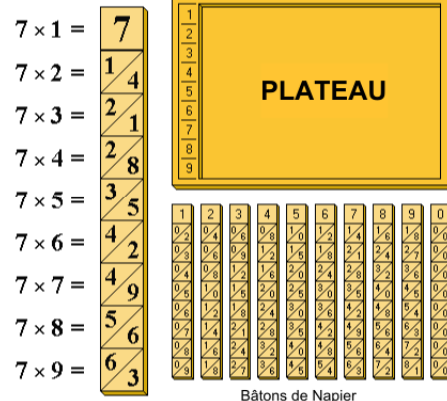
L'ordinateur est né du travail de plusieurs ingénieurs et théoriciens pendant la Seconde Guerre mondiale. Il n'y avait pas un pôle de développement, mais plusieurs centres indépendants, chacun essayant de construire des machines qui n'existaient pas à l'époque. [1]

L'évolution de l'ordinateur actuel est passée par plusieurs étapes fortement liées aux inventions technologiques qui se distinguent en plusieurs générations. [2]

### a. La génération zéro : les calculateurs mécaniques (Avant 1945)

#### ► Les abaques (avant 1600) :

- Instruments mécaniques facilitant le calcul



#### ► La Pascaline (1642) :

- Inventée par Blaise Pascal.
- Machine qui additionne et soustrait les nombres de 6 chiffres en base 10.
- Les multiplication et divisions se faisaient par répétitions.
- Première machine à calculer !



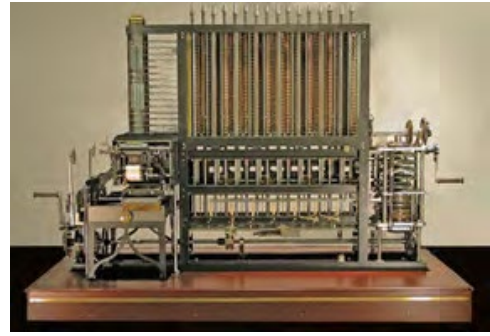
#### ► Le métier Jacquard (1805)

- Métier à tisser de Joseph Jacquard.
- Créé d'après des idées de Falcon en 1728.
- Système mécanique programmable.
- Utilise des cartes perforées pour métiers à tisser.
- C'est le **1er programme** !



► La machine analytique de Charles Babbage (1833)

- Machine programmable.
- Capable de réaliser différentes opérations codées sur des cartes perforées.
- Dotée d'un dispositif d'entrées et sorties.
- Un organe de commande gérant le transfert des nombres et leur mise en ordre pour le traitement.
- Un magasin permettant de stocker les résultats intermédiaires ou finaux (mémoire).
- Un moulin chargé d'exécuter les opérations sur les nombres.
- Un dispositif d'impression.



*b. La première génération : les tubes à vide (1945 - 1955)*

- La Seconde Guerre Mondiale précipite l'avènement des ordinateurs.
- Les sous-marins allemands communiquaient par radio → interception facile.
- Les messages étaient chiffrés avec une machine ENIGMA, volée aux allemands.
- Besoin de beaucoup de calculs, rapidement pour les décrypter → Création du premier ordinateur électronique : le **COLOSSUS**.



- Besoins de l'armée américaine pour le réglage des tirs d'artillerie.
- La course aux calculateurs est lancée à travers le monde !

► Les tubes à vide :

- Également appelés tubes électroniques ou même lampes.
- C'est des amplificateurs de signal.
- Un ensemble d'électrodes placées dans le vide ou dans un gaz.
- C'est une source d'électrons.
- Remplacés plus tard par des semi-conducteurs.



► L'ENIAC de John Mauchly (1946) :

- Electronic Numerical Integrator And Computer.
- 1<sup>er</sup> ordinateur électronique Turing-complet <sup>1</sup>.
- Calculs en système décimal.
- Composé de 18 000 tubes à vide et 1500 relais.
- 6000 commutateurs et une forêt de câbles.
- Pèse 30 tonnes, et occupe 167 m<sup>2</sup>.
- Incapable d'enregistrer un programme.

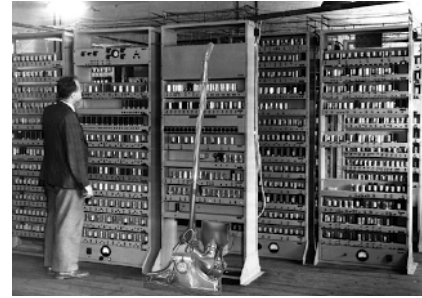


---

<sup>1</sup> Turing-complet désigne en informatique un système formel ayant au moins le pouvoir des machines d'Alan Turing (machine universelle qui peut exécuter n'importe quel programme).

► L'EDSAC de von Neumann, Eckert et Mauchly (1946) :

- Basée sur l'ordinateur EDVAC (Electronic Discrete Variable Automatic Computer).
- Utilise un système binaire.
- Les opérations d'addition +, soustraction - et multiplication  $\times$  étaient automatiques.
- La division  $\div$  était programmable.
- Capacité mémoire initiale : 1000 mots de 44 bits.



*c. La deuxième génération : les transistors (1955 - 1965)*

- Le transistor a été inventé en 1948, aux Bell Labs (prix Nobel de Physique en 1956).
- Les ordinateurs à tubes à vide deviennent obsolètes à la fin des années 50's.
- Le MIT (Massachusetts Institute of Technology) est précurseur avec le TX-0.
- Les ordinateurs deviennent assez stables pour être vendus à des clients : naissance de l'industrie de la mini-informatique (IBM, DEC, HP, ...).
- Premier jeu vidéo avec le PDP-1 : spacewar !
- Apparition des OS (Operating System) et langages évolués (FORTRAN et COBOL).



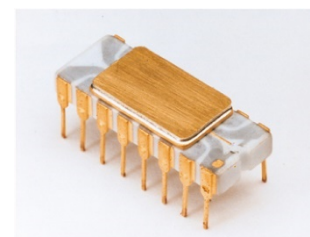
*d. La troisième génération : les circuits intégrés (1965 - 1973)*

- Le circuit intégré a été inventé en 1958.
- Des dizaines de transistors sur une seule puce.
- L'intégration a permis des ordinateurs plus petits, plus rapides et moins chers.
- Apparition de la multiprogrammation (plusieurs programmes exécutés en même temps sur la même machine).
- Possibilité d'émulation d'anciens modèles.



*e. La quatrième génération : les microprocesseurs (1971 - 1980)*

- Miniaturisation des circuits : l'ère de la micro-informatique
- Evolution de l'intégration avec la technologie VLSI (Very Large Scale Integration).
- Premier micro-processeur inventé par INTEL en 1971.



- Apparition des ordinateurs personnels (PC) utilisés pour :
  - Traitement de texte
  - Tableur
- Apparition d'Apple (1976)



#### *f. Aujourd'hui : la loi de Moore<sup>2</sup> est encore d'actualité*

Actuellement, les ordinateurs font partie intégrante de nos vies quotidiennes :

- Ordinateur jetable : Cartes de vœux
- Ordinateur enfoui : Montres, voitures
- Ordinateur de jeux : Jeux vidéo
- Ordinateur personnel (micro-ordinateur) : Ordinateurs portables ou de bureau
- Serveur : Serveurs de réseau
- Ensemble de stations de travail : Mini-superordinateur
- Mainframe : Traitement par lot dans une banque
- Superordinateur : Prévisions météo à long terme

## **2. Structure de base d'un ordinateur**

### *a. Définitions*

**Ordinateur** : Machine capable de résoudre des problèmes en appliquant des instructions.

**Instruction** : Action à effectuer par l'ordinateur, correspondant à une étape dans un programme.

**Programme** : Suite d'instructions décrivant la façon dont l'ordinateur doit effectuer un travail.

**Langage machine** : Ensemble des instructions exécutables directement par un ordinateur.

---

<sup>2</sup> Émise par l'ingénieur Gordon E. Moore en 1965, montrant l'évolution de la puissance de calcul des ordinateurs en fonction de la complexité du matériel informatique.



### *b. Architecture de John Von Neumann*

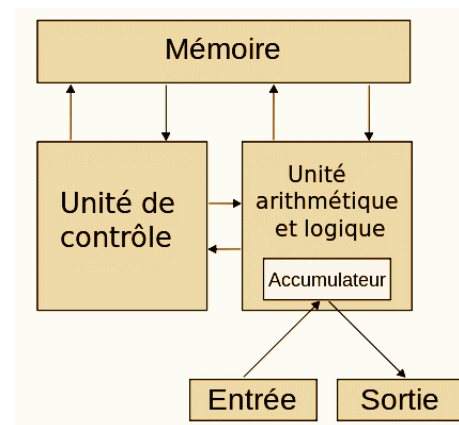
Cette architecture est appelée ainsi en référence au mathématicien John von Neumann qui a élaboré en juin 1945 dans le cadre du projet EDVAC1 la première description d'un ordinateur dont le programme est stocké dans sa mémoire. [3]



La plupart des ordinateurs modernes utilisent cette architecture, seules les technologies ont changé.

Cette architecture est basée sur 4 parties principales :

- **Unité arithmétique et logique (ALU) :** effectue les opérations de base.
- **Unité de contrôle :** chargée du séquençage des opérations.
- **Mémoire :** contient les données et le programme
  - Mémoire vive
  - Mémoire de masse
- **Entrées/Sorties :** permettent de communiquer avec le monde extérieur.



**Figure 1 :** Schématisation de l'architecture de von Neumann. [4]

### *c. Architecture en couches*

L'architecture matérielle et logicielle d'un ordinateur peut être représentée sous forme de couches superposées :

**Niveau 5 :** Couche des langages d'application (langages haut niveau).

**Niveau 4 :** Couche du langage d'assemblage.

**Niveau 3 :** Couche du système d'exploitation.

**Niveau 2 :** Couche architecture du jeu d'instructions (propre à chaque machine).

**Niveau 1 :** Couche microarchitecture (UAL, opérations, registres, ...).

**Niveau 0 :** Couche logique numérique (circuits logiques).

#### d. L'ordinateur du point de vue interne

- Machine électronique binaire.
- Fonctionnement des composants de base : circuits électroniques
- Organisation et communication entre les composants
- Utilise un langage machine (système binaire).
- Munie d'un système d'exploitation :
  - Programme principal de l'ordinateur.
  - Permet l'exécution simultanée d'autres programmes.
  - Gère des périphériques : entrées/sorties, stockage.

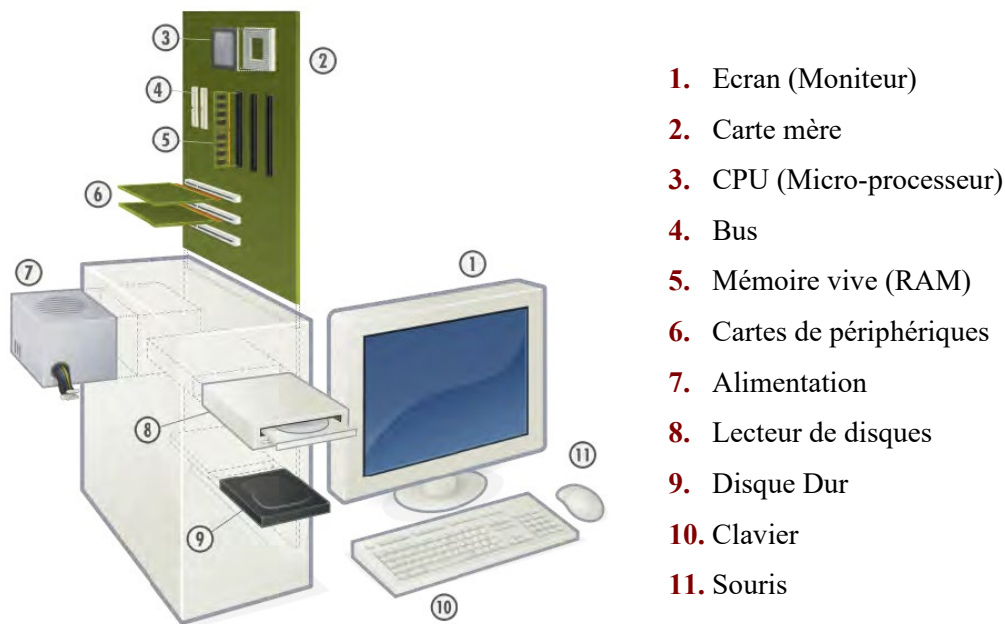


Figure 2 : L'ordinateur du point de vue interne [2]



#### Kézako ???



Annnonce trouvée sur un site de vente en ligne :

Asus ET2210IUTS-B002E <sup>1</sup>, Microsoft Windows 7 Pro 64 bits <sup>2</sup>, Intel Core i3-2120 <sup>3</sup>, Multi-points Full HD <sup>4</sup>, 21.5" <sup>5</sup>, 1920 x 1080 <sup>6</sup>, Intel HD Graphics <sup>7</sup>, 1To <sup>8</sup>, 4096Mo <sup>9</sup>, DVD+/-RW Super Multi <sup>10</sup>, 1,3 M Pixel + Micro <sup>11</sup>, 2x 2W + Sonic Master DTS Surround Sensation UltraPC <sup>12</sup>, [1x Entrée HDMI, 1x Sortie HDMI, LAN port (RJ 45)] <sup>13</sup>, 802.11 b/g/n <sup>14</sup>, [2x ports USB 3.0, 3x ports USB 2.0] <sup>15</sup>, 590 x 461 x 60-230 mm <sup>16</sup>, 10.8kg <sup>17</sup>, Titane noir <sup>18</sup>

- |                             |                        |                          |
|-----------------------------|------------------------|--------------------------|
| 1. Référence modèle         | 7. Carte graphique     | 13. Connectiques         |
| 2. Système d'exploitation   | 8. Disque dur          | 14. Wi-Fi                |
| 3. Processeur               | 9. RAM                 | 15. Ports USB            |
| 4. Affichage                | 10. Lecteur de disques | 16. Taille de l'ensemble |
| 5. Taille écran (en pouces) | 11. Webcam             | 17. Poids de l'ensemble  |
| 6. Résolution               | 12. Enceintes          | 18. Couleur              |

### i. Carte mère

La carte mère est un circuit imprimé qui permet de mettre en contact physique les différents composants et périphériques.

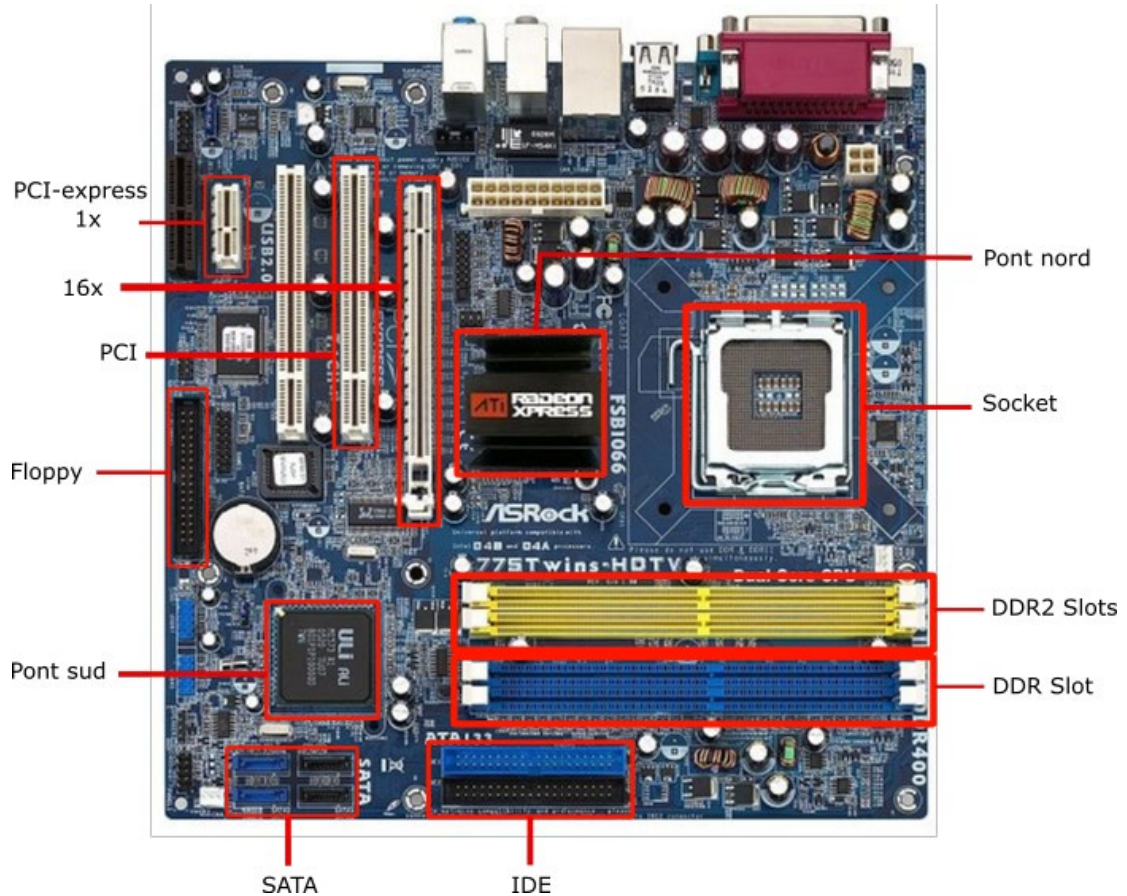


Figure 3 : Composants de la carte mère [5]

### ii. Chipset

Composé de 2 parties :

**Le pont nord** (northbridge) : chargé de gérer les composants qui ont besoin d'une bande passante importante :

- microprocesseur
- mémoire vive (RAM)
- carte graphique

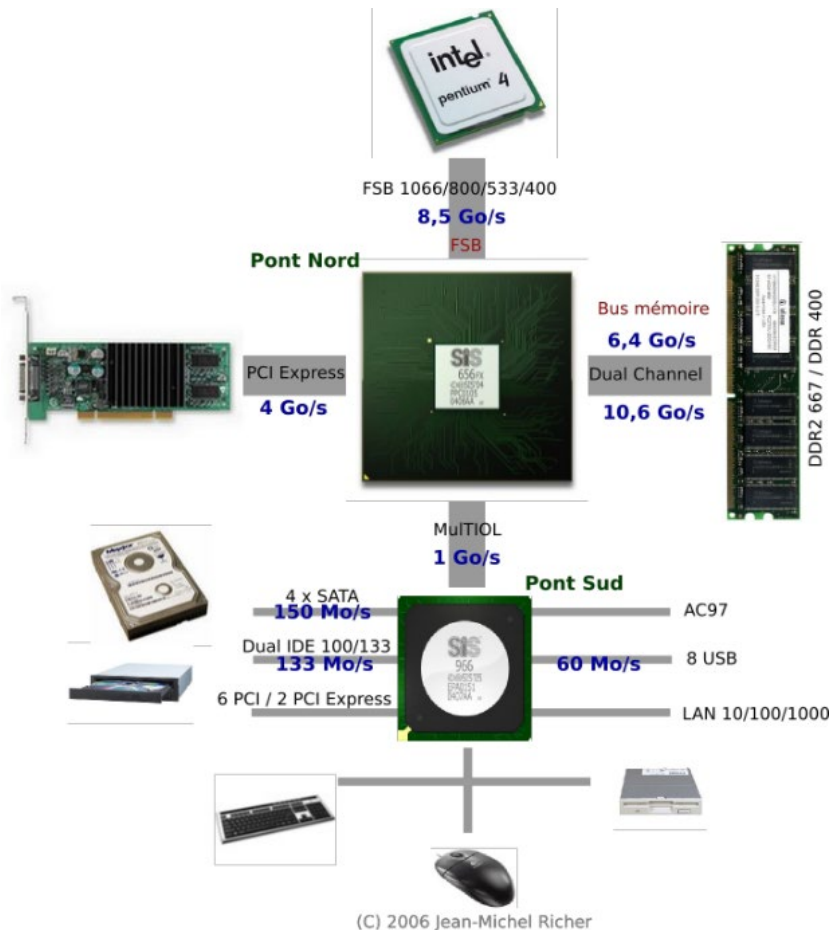


**Le pont sud** (southbridge) : chargé de gérer les périphériques qui ont besoin d'une faible bande passante :

- clavier, souris, audio
- port parallèle, port série
- périphériques USB, FireWire
- réseau
- disques durs, CD/DVD Rom



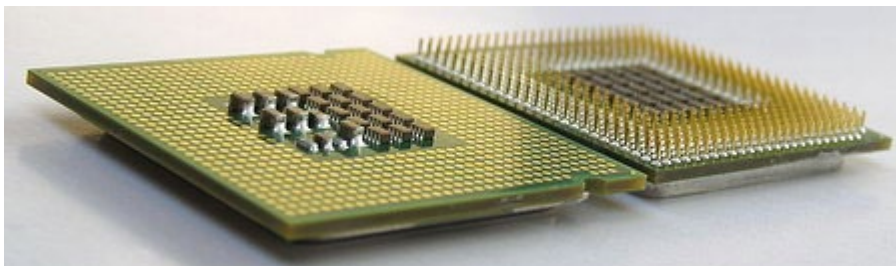




**Figure 4 :** Vue schématique d'une carte mère de type Intel avec chipset Sis 656fx/966, (2006) [5]

### iii. Le Processeur

- En anglais CPU (Central Processing Unit) ou UC (Unité Centrale) en français.
- C'est le « Cerveau » de l'ordinateur.
- Exécute les programmes stockés en mémoire principale.
- Il est responsable du :
  - Chargement des instructions
  - Décodage des instructions
  - Exécution des instructions, l'une après l'autre



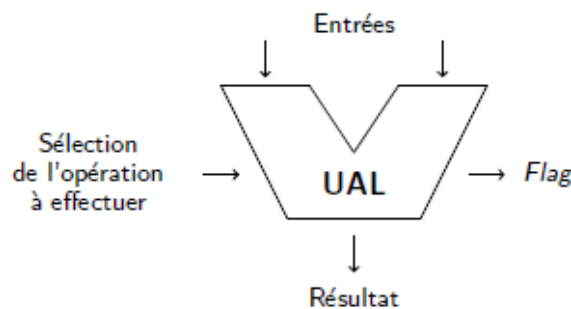
Il est composé de :

- **Unité Arithmétique et Logique (UAL) :**
  - En anglais ALU (Arithmetic and Logical Unit).
  - Responsable des opérations indiquées par les instructions.

- **Unité de commande :**
  - Récupère les instructions présentes en mémoire principale.
  - Décode les instructions.
- **Les registres :**
  - Petites zones mémoires.
  - Peuvent être lus ou écrits extrêmement rapidement.
- **Les bus :** interconnectent les éléments fonctionnels internes.

### 1. L'Unité Arithmétique et Logique (UAL)

- Responsable des calculs sur des nombres entiers.
- Opérations communes :
  - **Opérations arithmétiques** : addition, soustraction, changement de signe, . . .
  - **Opérations logiques** : compléments, et, ou, ou-exclusif, non, non-et, . . .
  - **Comparaisons** : test d'égalité, supérieur, inférieur, . . .
  - **Décalages** : des bits à gauche, à droite...



### 2. Les registres communs

- **Compteur ordinal (CO)** : contient l'adresse mémoire de l'instruction en cours d'exécution ou prochainement exécutée.
- **Accumulateur (ACC)** : pour stocker les données en cours de traitement par l'UAL.
- **Registre d'instructions (RI)** : contient l'instruction en cours de traitement.
- **Pointeur(s) de pile (SP)** : (Stack Pointer) contient l'adresse du sommet de la pile <sup>3</sup>.
- **Registres généraux ( $R_0, \dots, R_n$ )** : registres de stockage intermédiaire pour les calculs.

### 3. Exécution d'une instruction

L'exécution d'une instruction par le processeur passe par les étapes suivantes :

- 1) Charger la prochaine instruction à exécuter dans le registre instruction (RI).
- 2) Modifier le compteur ordinal (CO) pour qu'il pointe sur l'instruction suivante.
- 3) Décoder (analyser) l'instruction chargée.
- 4) Localiser en mémoire d'éventuelles données nécessaires à l'instruction.
- 5) Charger, si nécessaire, les données dans les registres généraux.
- 6) Exécuter l'instruction.
- 7) Recommencer à l'étape 1.

---

<sup>3</sup> Une pile est une zone de la mémoire gérée par le processeur selon le principe « dernier arrivé, premier sorti » (en anglais LIFO pour last in, first out).

#### iv. La Mémoire Principale

- Mémoire « de travail » de l'ordinateur.
- Mémoire vive : RAM (Random Access Memory)
- Caractéristiques :
  - Rapide d'accès.
  - Volatile.
- Le processeur y accède pour lire/écrire des données.

#### v. Les Entrées/Sorties

- En anglais « Input/Output » (I/O)
- Permettent les échanges d'information entre le processeur et les périphériques associés.
- **Entrées** : données envoyées par un périphérique à destination du processeur.
- **Sorties** : données émises par le processeur à destination des périphériques.

##### 1. Les périphériques d'entrée :

Permettent à l'utilisateur de fournir une information à l'ordinateur.

- Exemples : clavier, scanner, . . .

##### 2. Les périphériques de sortie :

Permettent à l'ordinateur de fournir une information à l'utilisateur

- Exemples : écran, enceintes, . . .

##### 3. Les périphériques d'entrée-sortie :

Permettent à l'utilisateur/l'ordinateur de fournir/recevoir une information

- Exemples : clé USB, . . .

- Quelques exemples de périphériques :

<i>Périphérique</i>	<i>Entrée</i>	<i>Sortie</i>	<i>Entrée/Sortie</i>
<i>Clavier</i>	•		
<i>Souris</i>	•		
<i>Ecran</i>		•	
<i>Ecran tactile</i>			•
<i>Lecteur CD/DVD</i>	•		
<i>Graveur CD/DVD</i>			•
<i>Webcam</i>	•		
<i>Imprimante</i>		•	
<i>Carte réseau</i>			•
<i>Microphone</i>	•		
<i>Enceinte</i>		•	
<i>Scanner</i>	•		
<i>Disque dur</i>			•
<i>Clé USB</i>			•

#### vi. Les bus

- Canaux de communication à l'intérieur de l'ordinateur.
- Relient les différents composants de l'ordinateur.
- Caractérisés par :
  - **Un type** : parallèle ou série.
  - **Une largeur** : nombre de bits que le bus peut transmettre à la fois.
  - **Une fréquence** (vitesse) : nombre de paquets envoyées par seconde (en Hz).
  - **Une bande passante** (débit) = largeur × fréquence.

Il y a 3 types de bus :

- **Bus de données** : définit la taille des données pour les E/S
- **Bus d'adresse** : permet l'adressage de la mémoire
- **Bus de contrôle** : permet la gestion du matériel, via les interruptions

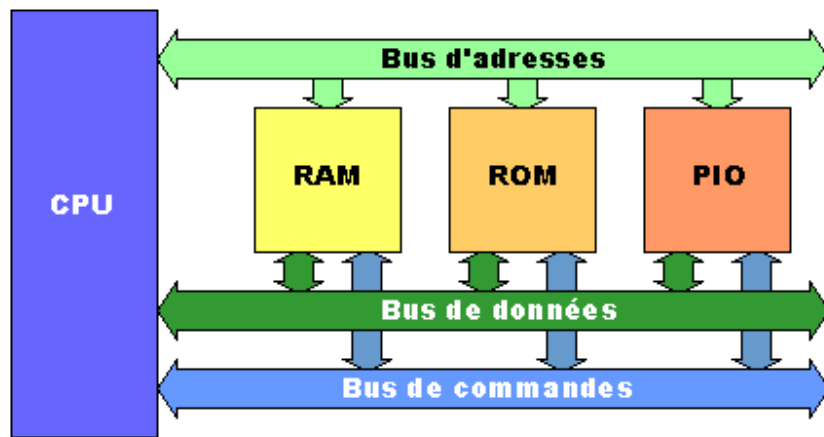


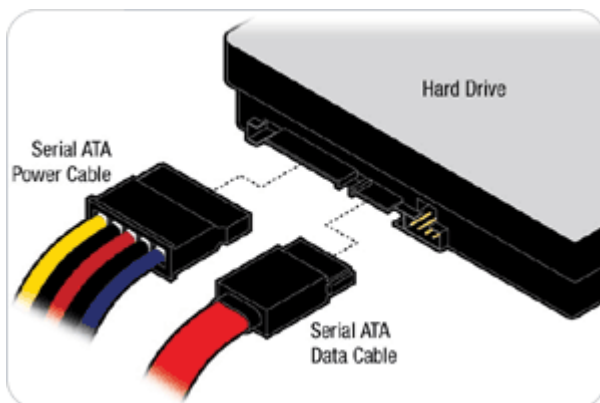
Figure 5 : Schéma des bus de communication [5]

#### ► La connectique :

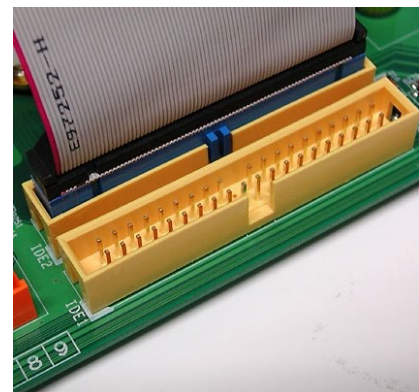
Les composants de l'unité centrales sont connectés à la carte mère via des ports spécifiques.

- Ports pour disques dur et lecteurs CD/DVD :

Ports SATA



Port IDE (PATA)



- Ports pour cartes additionnelles : (cartes graphique, carte son, carte TV...)

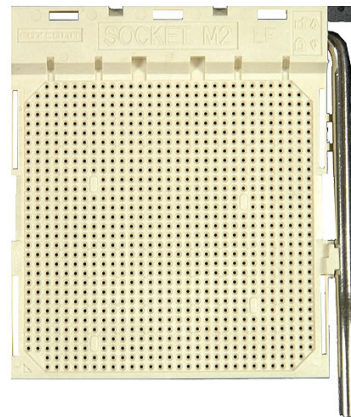
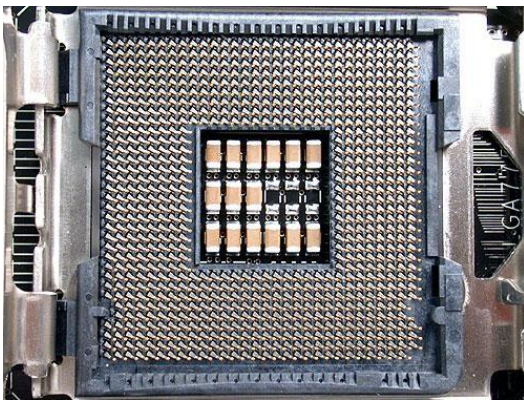
Port PCI



Port AGP



- Socket : pour brancher le microprocesseur.



- Ports USB: **U**niversal **S**erial **B**us





### 3. Représentation et 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.

#### a. Codage de l'information

Le 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 35 : 35 est la représentation externe du nombre trente-cinq.
- La représentation interne de 35 sera une suite de 0 et 1 (100011).

En informatique, Le codage de l'information s'effectue principalement en trois étapes :

1. L'information sera exprimée par une suite de nombres (**Numérisation**).
2. Chaque nombre est codé sous forme **binaire** (suite de 0 et 1).
3. Chaque élément binaire est représenté par un état **physique**.

Le codage de l'élément binaire par un état physique peut se faire par :

- 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.


#### b. 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.

##### i. Exemples de Système de numération

- Numération Romaine : I, II, III, IV, ..., X, ..., CCLXXI -> 271
- Numération babylonienne : 
- Numération décimale :  $\mathcal{A} = \{0,1,2,3,4,5,6,7,8,9\}$ 
  - Le nombre 10 est la base de cette numération
  - C'est un système positionnel. Chaque position possède un poids.

## ii. 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:

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

- 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.
- La valeur de N en base 10 est donnée par :
$$N = a_{n-1} \cdot b^{n-1} + a_{n-2} \cdot b^{n-2} + \dots + a_0 \cdot b^0_{(10)} = \sum_{i=0}^{n-1} a_i \cdot b^i$$

## iii. Bases de numération

- **Système binaire** : système à base 2 (b=2) utilise deux chiffres : {0,1}
  - C'est avec ce système que fonctionnent les ordinateurs
- **Système Octal** : système à base 8 (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 **3 bits** par un seul symbole.
- **Système Hexadécimal** : système à base 16 (b=16) utilise 16 chiffres :  
{0,1,2,3,4,5,6,7,8,9, A=10<sub>(10)</sub>, B=11<sub>(10)</sub>, C=12<sub>(10)</sub>, D=13<sub>(10)</sub>, E=14<sub>(10)</sub>, F=15<sub>(10)</sub>}
  - Cette base est très utilisée dans le monde de la micro-informatique.
  - Elle permet de coder **4 bits** par un seul symbole.

## iv. Transcodage (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 :

- Décimal vers Binaire, Octal et Hexadécimal.
- Binaire vers Décimal, Octale et Hexadécimal.

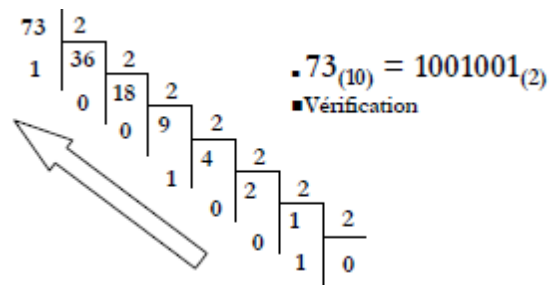
### 1. De la base 10 vers une base b

La règle à suivre est les divisions successives :

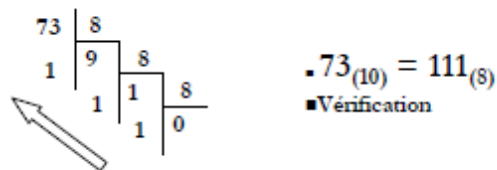
- On divise le nombre par la base b.
- Puis le quotient par la base b.
- Ainsi de suite jusqu'à l'obtention d'un quotient nul.
- La suite des restes correspond aux symboles de la base visée.
- On obtient en premier le chiffre de poids faible et en dernier le chiffre de poids fort.

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

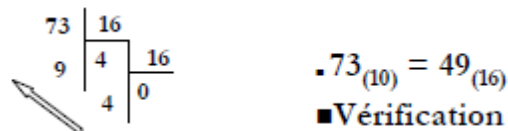
Représentation en Binaire ?



Représentation en Octal ?



Représentation en Hexadécimal ?



## 2. De la base 2 vers une base b

- **Solution 1** : convertir le nombre en base binaire vers la base décimale puis convertir ce nombre en base 10 vers la base b.

○ Exemple :

$$10010_{(2)} = ?_{(8)}$$

$$10010_{(2)} = 2^4 + 2^1_{(10)} = 18_{(10)} = 2*8^1 + 2*8^0_{(10)} = 22_{(8)}$$

- **Solution 2** :

○ Binaire vers décimal : par définition ( $\sum_{i=0}^{n-1} a_i \cdot b^i$ )

► **Exemple** : Soit  $N = 1010011101_{(2)} = ?_{(10)}$

$$N = 1 \times 2^9 + 0 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

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

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

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

○ Binaire vers octal : regroupement des bits en des sous-ensembles de trois bits puis remplacer chaque groupe par le symbole correspondant dans la base 8. (Voir tableau ci-contre)

► **Exemple** : Soit  $N = 1010011101_{(2)} = ?_{(8)}$

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

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

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

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

- Binaire vers Hexadécimal : regroupement des bits en des sous-ensembles de quatre bits puis remplacer chaque groupe par le symbole correspondant dans la base 16. (Voir tableau ci-contre).

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

► **Exemple** : Soit  $N = 1010011101_{(2)} = ?_{(16)}$

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

$$= \text{2}\ \text{9}\ \text{D}_{(16)}$$

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

### c. Codage des nombres

#### i. 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 : sur un octet,  $10_{(10)}$  se code en binaire pur :  $00001010_{(2)}$

##### 2. Etendu du codage binaire pur :

- Sur n bits on peut coder 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  $2^{32} - 1 = 4\ 294\ 967\ 295$

##### 3. 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	<sup>(1)</sup> 0

Retenue

► Exemple (Addition)

Multiplication

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

Addition binaire (8 bits)

$$\begin{array}{r} 10010110 \\ + 01010101 \\ \hline 11101011 \end{array}$$

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

$$\begin{array}{r} 10010110 \\ + 01110101 \\ \hline 100001011 \\ \Rightarrow \text{Overflow} \end{array}$$

► Exemple (Multiplication)

Multiplication binaire

$$\begin{array}{r}
 1\ 0\ 1\ 1\ (4\ \text{bits}) \\
 \times 1\ 0\ 1\ 0\ (4\ \text{bits}) \\
 \hline
 0\ 0\ 0\ 0 \\
 1\ 0\ 1\ 1\ . \\
 0\ 0\ 0\ 0\ . \\
 1\ 0\ 1\ 1\ . \\
 \hline
 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0
 \end{array}$$

Sur 4 bits le résultat est faux  
 Sur 7 bits le résultat est juste  
 Sur 8 bits on complète à gauche par un 0

ii. Codage des entiers relatifs

Il existe au moins trois façons pour coder un entier relatif (signé) :

- Code binaire signé (par signe et valeur absolue).
- Code complément à 1.
- Code complément à 2 (utilisé sur ordinateur).

**1. 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.

► 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.

**2. 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 pur.
- 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.
  - ▶ Exemple : -24 en complément à 1 sur 8 bits  
 $|-24|$  en binaire pur  $\Rightarrow 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0_{(2)}$ , puis on inverse les bits  $\Rightarrow 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1_{(cà1)}$

- Limitations du complément à 1 :
  - 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.

### 3. Code complément a 2

Aussi appelé **Complément Vrai (CV)** :

- Les nombres positifs sont codés de la même manière qu'en binaire pur.
- 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  $0\ 0\ 0\ 1\ 1\ 0\ 0\ 0_{(2)}$   
 $-24 \Rightarrow 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1_{(cà1)}$ , puis on lui additionne 1 donc -24 est codé par  $1\ 1\ 1\ 0\ 1\ 0\ 0\ 0_{(cà2)}$
- Un seul codage pour 0. Par exemple sur 8 bits :
  - +0 est codé par  $00000000_{(cà2)}$
  - -0 est codé par  $11111111_{(cà1)} \Rightarrow$  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 = 11111111
    - +127 = 01111111      -128 = 10000000

### iii. 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 :  $0,5425 . 10^2_{(10)}$  ;  $10,1 . 2^{-1}_{(2)}$  ;  $A0,B4.16^{-2}_{(16)}$

## 1. Codage en Virgule Fixe

Etant donné une base  $b$ , un nombre  $X$  est représenté, en format virgule fixe, par :

- $X = a_{n-1} a_{n-2} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-p} (b)$
- $a_{n-1}$  est le chiffre de poids fort (MSB<sup>4</sup>).
- $a_{-p}$  est le chiffre de poids faible (LSB<sup>5</sup>).
- $n$  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 :  $X = \sum_{i=0}^{n-1} a_i \cdot b^i$

► Exemple :  $101,01_{(2)} = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 5,25_{(10)}$

✓ Changement de base  $10 \rightarrow 2$

- Le passage de la base 10 à la base 2 est défini par :
  - Une partie entière codée sur  $p$  bits (division successive par 2)
  - Une partie décimale 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

$$4_{(10)} = 100_{(2)}$$

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

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

$$\Rightarrow \text{donc } 4,25_{(10)} = 100,01_{(2)}$$

## 2. Codage en Virgule Flottante

Un nombre réel est représenté en virgule flottante sous la forme :  $\pm M \times 2^E$ , où  $M$  est la mantisse (virgule fixe) et  $E$  l'exposant (signé). Coder en base 2 au format virgule flottante, revient à coder le signe, la mantisse et l'exposant.

► Exemple : Codage en base 2, format virgule flottante, du nombre 3,25

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

$$= 1,101 \times 2^1_{(2)} \rightarrow \text{on décale la virgule par 1 bit à gauche.}$$

$$= 110,1 \times 2^{-1}_{(2)} \rightarrow \text{on décale la virgule par 1 bit à droite.}$$

**Problème** : il y a différentes manières de représenter  $E$  et  $M$ .

$\Rightarrow$  **Normalisation** : afin d'avoir la même représentation, tout nombre doit d'abord être normalisé sous la forme :  $\pm 1, M \times 2^{Eb}$

- Le signe est codé sur 1 bit ayant le poids fort (**S**):
  - Le signe  $-$  : bit 1
  - Le signe  $+$  : bit 0

---

<sup>4</sup> Most significant bit

<sup>5</sup> Less significant bit

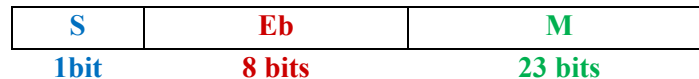
- Exposant biaisé (**Eb**) :
  - Placé avant la mantisse pour simplifier la comparaison.
  - Codé sur p bits et biaisé pour être positif (ajout de  $2^{p-1}-1$ ).
- Mantisse normalisée (**M**) :
  - Normalisée : la virgule est placée après le bit à 1 ayant le poids fort.
  - M est codé sur q bits
  - Exemple :  $11,01 \Rightarrow 1,101 \times 2^1$  donc M = 101



➤ Standard IEEE 754 (1985)

C'est la norme la plus employée actuellement pour le calcul des nombres à virgule flottante avec les CPU. Elle définit les nombres réels sur les formats :

- **Simple précision**, codé sur 32 bits :
  - 1 bit de signe (**S**).
  - 8 bits pour l'exposant biaisé (**Eb**).
  - 23 bits pour la mantisse (**M**).



- **Double précision**, codé sur 64 bits :
  - 1 bit de signe (**S**).
  - 11 bits pour l'exposant biaisé (**Eb**).
  - 52 bits pour la mantisse (**M**).



► Conversion décimale - IEEE754 (Codage d'un réel)

$35,5_{(10)} = ?_{(2)}$  (IEEE 754 simple précision)

Nombre positif,  $\Rightarrow$  donc S = 0

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

$= 1,000111 \times 2^5$  (virgule flottante)

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

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

0 10000100 000111000000000000000000 (IEEE 754 SP)

S      Eb                                  M

► Conversion IEEE754 – Décimale (Evaluation d'un réel)

0 10000001 111000000000000000000000 (IEEE 754 SP)

S      Eb                                  M

S = 0, donc nombre positif

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

1,M = 1,111

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

► Caractéristiques des nombres flottants au standard IEEE

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

d. Codage des caractères

Les caractères peuvent être soit Alphabétique (A-Z, a-z), soit numérique (0, ..., 9), des signes de ponctuation, ou des caractères spéciaux (&, \$, %, ...). Le codage des caractères revient à créer une table de correspondance entre les caractères et des nombres.

► Les Standards :

• 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)

	MSB	8	9	A	B	C	D	E	F
LSS	1000	1001	1010	1011	1100	1101	1110	1111	
0	0000	0	1	2	3	4	5	6	7
1	0001	8	9	10	11	12	13	14	15
2	0010	16	17	18	19	20	21	22	23
3	0011	24	25	26	27	28	29	30	31
4	0100	32	33	34	35	36	37	38	39
5	0101	40	41	42	43	44	45	46	47
6	0110	48	49	50	51	52	53	54	55
7	0111	56	57	58	59	60	61	62	63
8	1000	64	65	66	67	68	69	70	71
9	1001	72	73	74	75	76	77	78	79
A	1010	80	81	82	83	84	85	86	87
B	1011	88	89	90	91	92	93	94	95
C	1100	96	97	98	99	100	101	102	103
D	1101	104	105	106	107	108	109	110	111
E	1110	112	113	114	115	116	117	118	119
F	1111	120	121	122	123	124	125	126	127

• Code **ASCII Etendu** :

- 8 bits pour représenter 256 caractères (0 à 255)
- Code les caractères accentués : à, é, è, ...etc.
- Compatible avec ASCII.

• Code **Unicode** (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, ...etc.
- On en a défini environ 50 000 caractères pour l'instant.

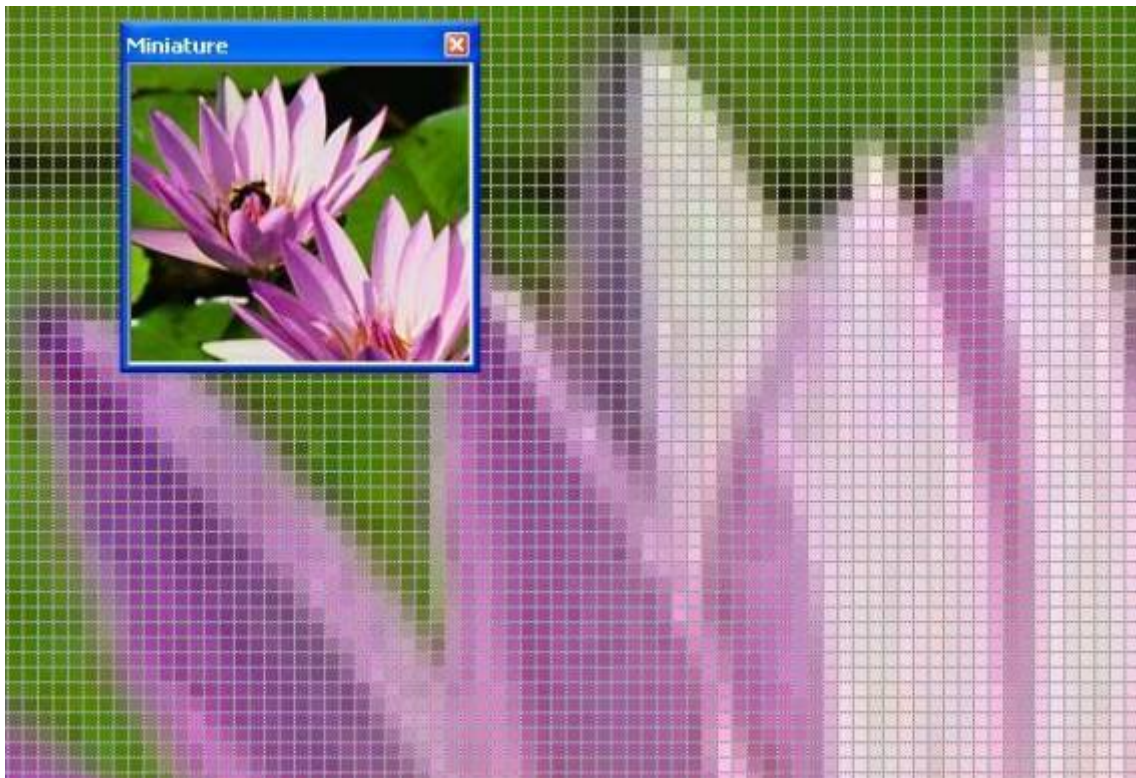
	FE7	FE8	FE9	FEA	FEB	FDF
0	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ
1	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ
2	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ
3	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ
4	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ	ﷲ

### *e. Codage d'une image*

Le principe du codage d'une image :

- 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 importants 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.



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 (**P**ICture **E**LEMENT) :
  - Image en noir et blanc → 1 bit pour chaque pixel.
  - Image avec 256 couleurs → 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**.

L'**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.

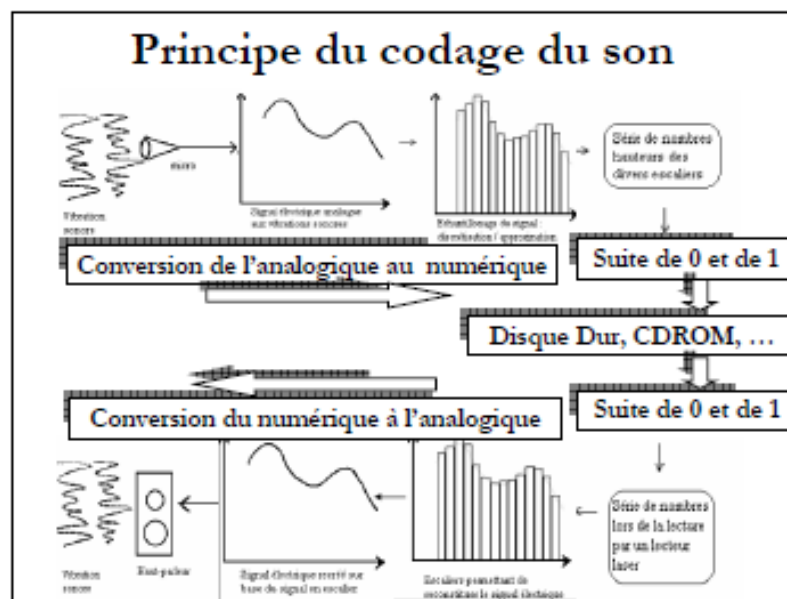
Le **poids** d'une image (exprimé en Ko ou en Mo) : est égal à son nombre de pixels (définition) que multiplie le poids de chacun des pixels (profondeur).

#### *f. Codage du son*

Un son est une vibration mécanique se propageant dans l'air ou dans un autre milieu (fluide, solide...). Lors de la numération, il faut transformer le signal analogique (continu) en une suite de nombres qui seront traités par l'ordinateur. C'est le rôle du convertisseur analogique/numérique.

La numérisation d'un son se réalise en deux étapes :

- **L'échantillonnage** : consiste à prélever périodiquement des échantillons d'un signal analogique selon une période que l'on appellera période d'échantillonnage.
- **La quantification** : consiste à affecter une valeur numérique à chaque échantillon prélevé.



Les Codecs, abréviation de codeur/décodeur, sont des logiciels qui permettent de réaliser ce codage/décodage du son dans un ordinateur. Ces logiciels s'appuient sur les éléments matériels disponibles dans la carte son.