

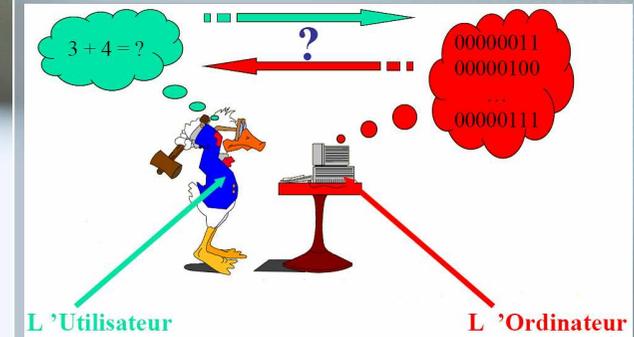
## M5E2 – Informatique 2

E2: Informatique 2

himmi@fsr.ac.ma

Département de physique

## Informatique ?

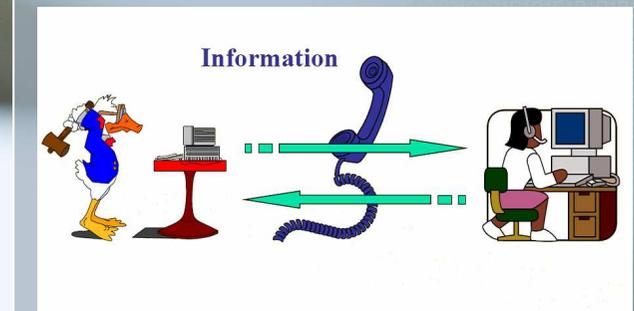


## Informatique 2

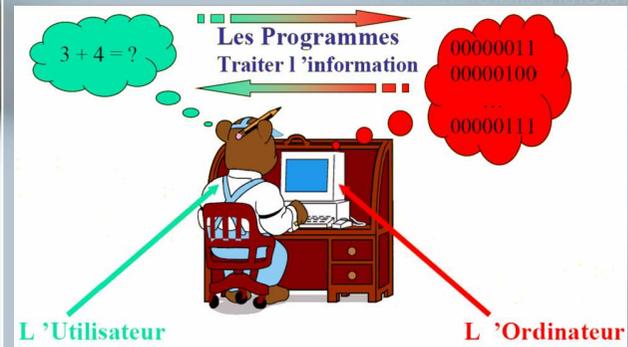
### ■ Objectifs

- Expliquer les principes de fonctionnement d'un ordinateur;
- Expliquer comment l'information est représentée, stockée;
- Expliquer le rôle d'un système d'exploitation;
- Apprendre à structurer et à organiser l'information.
- Revoir la logique booléenne
- Initiation à l'algorithmique
- Quelques logiciels utiles
- ...

## Informatique ?



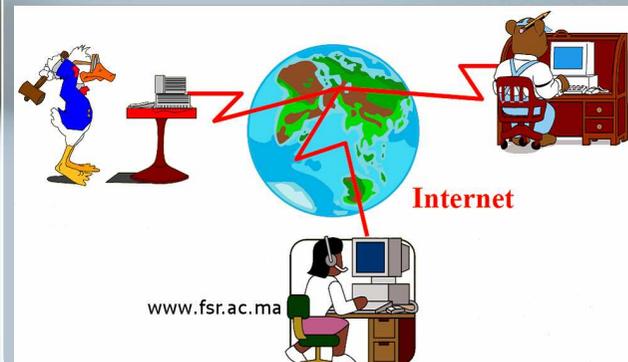
## Informatique ?



## Science Informatique

- **Science** qui regroupe l'ensemble des théories et des techniques permettant de **traiter de l'information** à l'aide d'un ordinateur
- **Sciences et Technologies de l'Information et de la Communication**

## Informatique ?



## Science Informatique

- **Information**
  - Élément de connaissance représenté à l'aide de conventions en vue d'être conservé, traité et communiqué
  - Différentes formes : son, image, texte, vidéo ...
- **Traitement**
  - Passer d'informations appelées données à d'autres informations dites résultats
  - Exemples : addition, traduction...

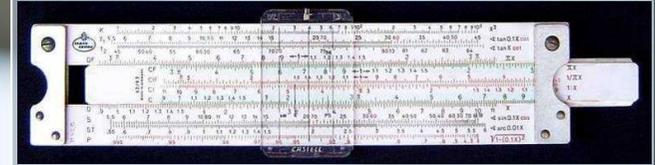
## L'ordinateur

- Machine (**calculateur**) commandée par un **programme** enregistré qui permet de traiter des informations en exécutant une **séquence finie d'instructions** (opérations arithmétique et logique)
  - **Universel** (qui peut s'appliquer à toutes et tous, qui peut être reconnu par le monde entier comme utilisable)
  - **Rapide** (Millions d'Instructions par seconde)
  - **Fiable** (accomplir une fonction requise dans des conditions données pour une période de temps donnée)
  - **Grande Capacité mémoire**



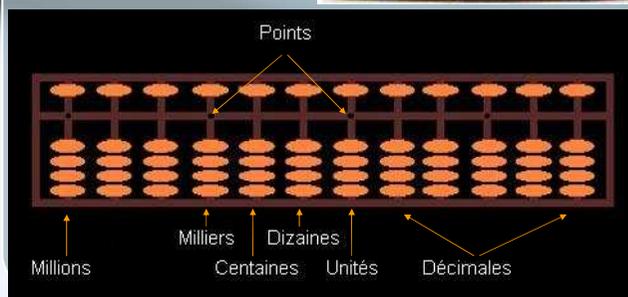
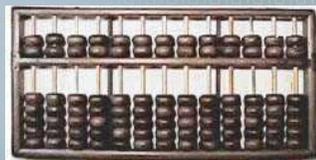
## Historique

- la Règle à Calcul, 1622



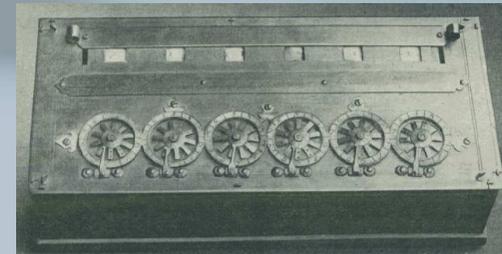
## Historique

- Boulier chinois, 700



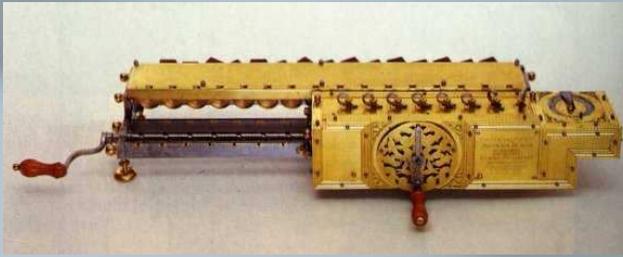
## Historique

- Machine de Pascal, 1642



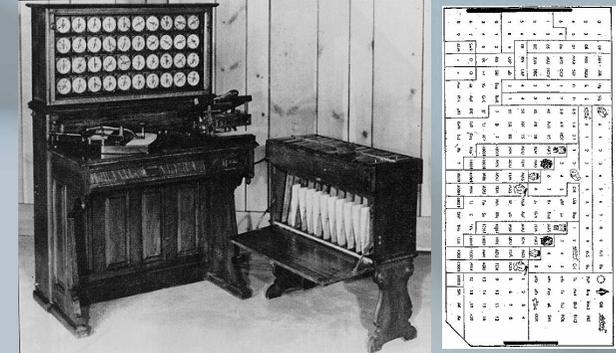
## Historique

- la calculatrice de Leibniz, 1672, 4 opérations et extrait les racines carrées



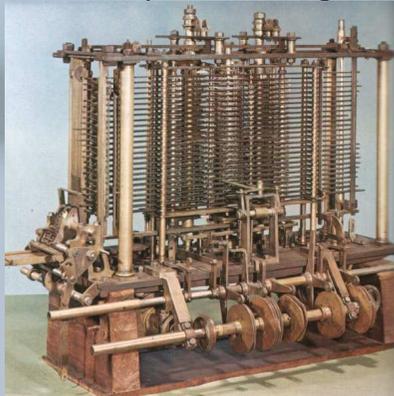
## Historique

- Machine à carte de Hollerith, 1890



## Historique

- Mémoire mécanique de Babbage, 1883



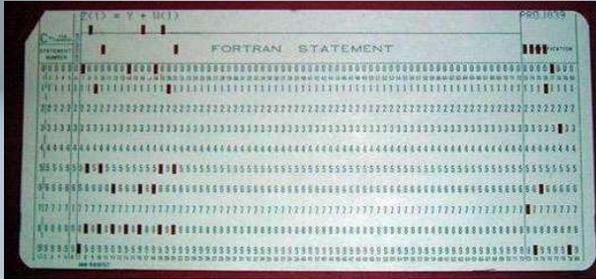
## Historique

- Machine à carte de Hollerith, 1890



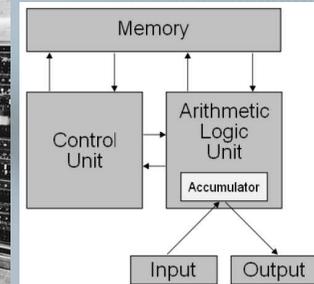
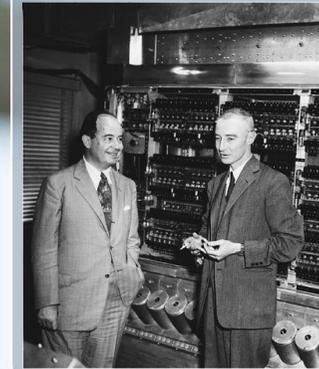
## Historique

- Carte perforée, 1890



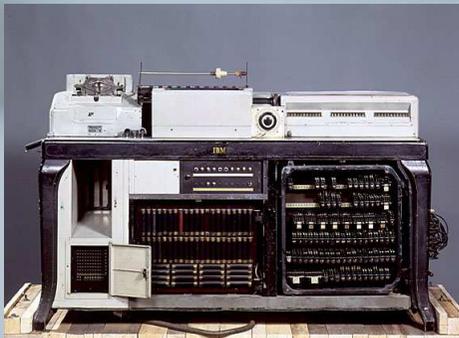
## Historique

- Von Neumann, 1946



## Historique

- Machines électroniques IBM, 1940



A printed output from the IBM machine. It shows a table of data with columns and rows. The table is printed on a piece of paper.

Program	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2																				
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				
11																				
12																				
13																				
14																				
15																				
16																				
17																				
18																				
19																				
20																				

## Historique

- **Génération 1 (1945 - 1960)**
  - langage binaire
  - Entrée : Carte perforée, Sortie : Imprimante
  - 1000 opérations élémentaires/s
- **Génération 2 (1960-1965)**
  - Langage évolué
  - transistors, diodes, mémoires à tores
  - 100000 opérations élémentaires/s

## Historique

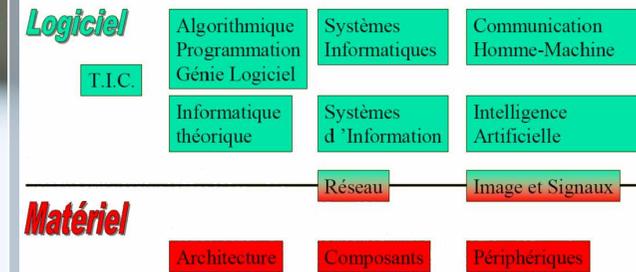
### ■ Génération 3 (1965-1975)

- Circuit intégré, la puce
- $10^9$  à  $10^{12}$  opérations élémentaires/s

### ■ Génération 4 (1975- ? )

- microprocesseurs, dizaines de circuits sur une puce
- Développement des ordinateurs personnels, ...

## Domaines de l'informatique



## Familles d'ordinateur

### ■ Ordinateurs centraux (MainFrame)

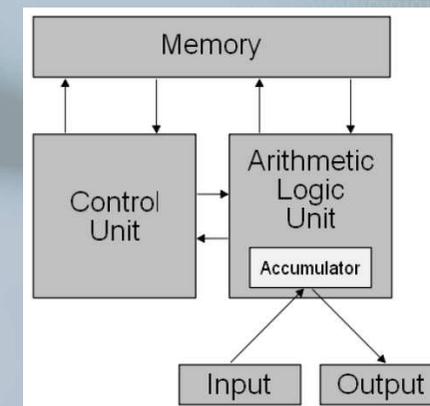
### ■ Ordinateurs Personnels (Personal Computer)

- Ordinateur de Bureau (Desktop computer)
- Ordinateur Portable (Laptop, Notebook)

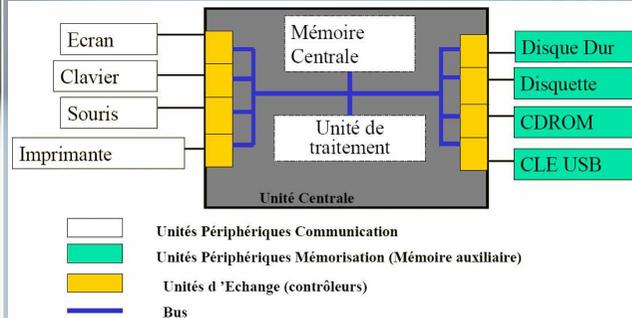
### ■ Ordinateurs de poche/ assistants personnels (Personal Digital Assistant)

### ■ Systèmes temps réel (Industrie)

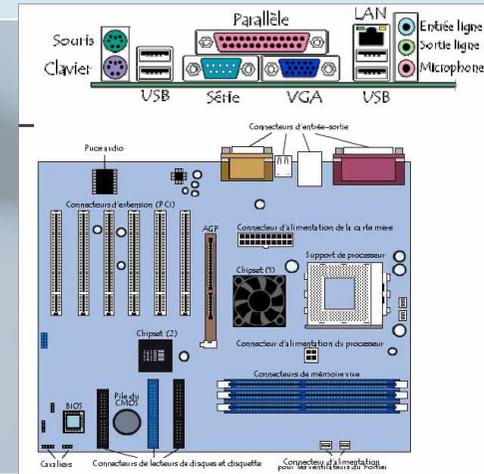
## Ordinateur de Von Neumann



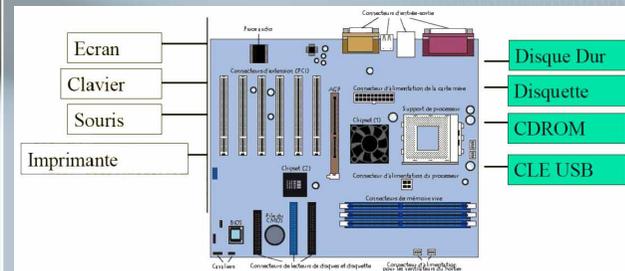
## Ordinateur de Von Neumann



## Ordinateur de type PC



## Ordinateur de type PC



## La mémoire

### ■ Dispositif capable d'enregistrer, de stocker et de restituer des informations

- Ensemble fini de cellules
- Chaque case a un numéro : adresse
- Unités de mesure: **bit**, octet (Byte), mot
- Chaque case correspond à un mot

### ■ Trois types

- mémoire morte ROM: lecture seule, Quelques Ko
- mémoire vive RAM: lecture et écriture, Mo – Go
- mémoire de masse ou auxiliaire

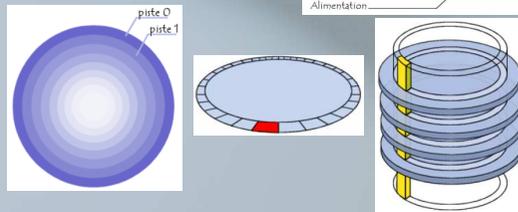
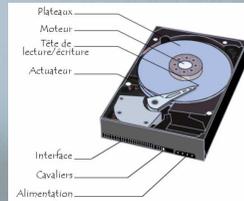


## La mémoire auxiliaire

### ■ Stockage et restitution d'information

#### ■ disque dur

- lecture et écriture
- taille : centaines de Go
- débit de transfert : 100 Mo/s
- logiciels + données



## La mémoire auxiliaire

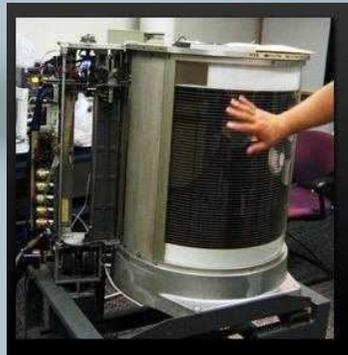
### ■ disquettes

- lecture et écriture
- taille : 1,44 Mo
- débit de transfert : 150 ko/s
- sauvegarde données (transport)



## La mémoire auxiliaire

Disque Dur de 5 MB, 1956

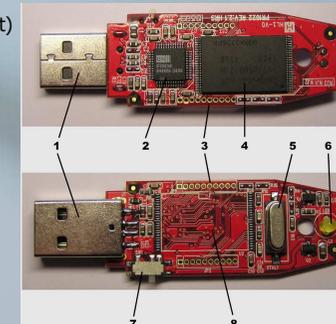


## La mémoire auxiliaire

### ■ clés USB

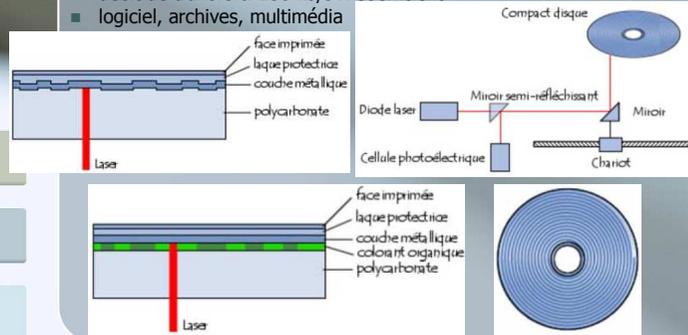
- lecture et écriture
- taille : centaines de Mo
- débit de transfert : 1 Mo/s
- sauvegarde données (transport)

1. Connecteur USB mâle (type A).
2. Contrôleur pour l'USB 2.0
3. JP1 et JP2 pour tests et débogage.
4. Mémoire flash
5. Oscillateur à quartz 12 MHz.
6. DEL pour indiquer l'activité de la clé.
7. Interrupteur (protéger la clé en écriture).
8. Zone prévue pour étendre la capacité sans avoir à créer un autre schéma.



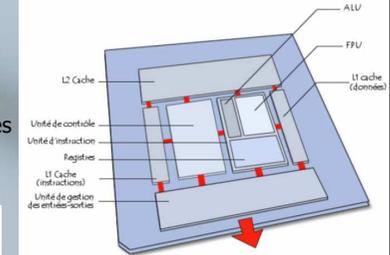
## La mémoire auxiliaire

- CDROM, DVDROM
- lecture
- taille : 700 Mo (CD), 5 Go (DVD)
- débit de transfert: 150 ko/s x Coefficient
- logiciel, archives, multimédia

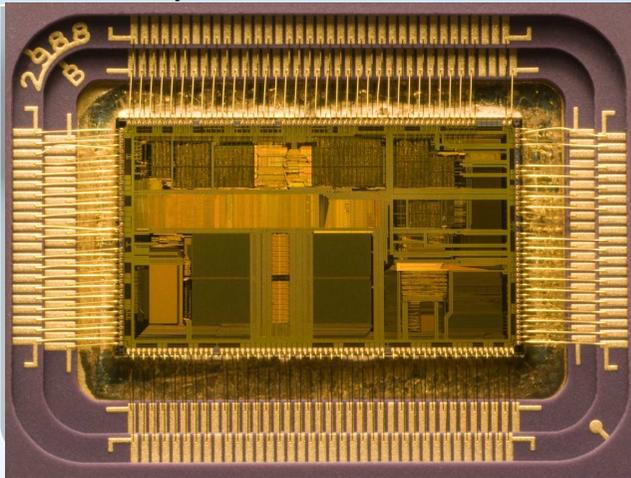


## L'unité de traitement

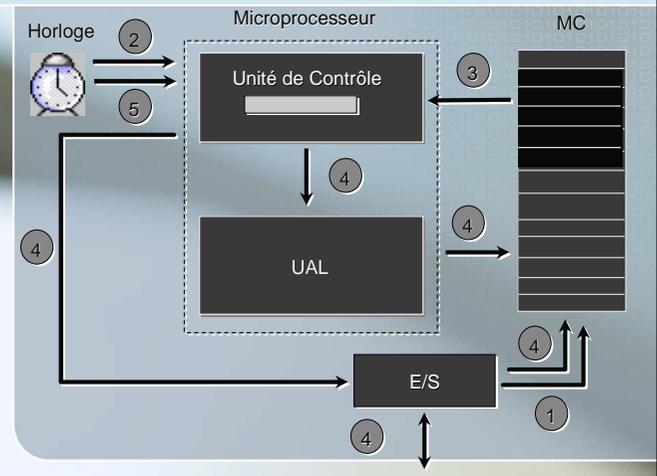
- Le microprocesseur
- Unité Arithmétique et logique (calcul)
- Unité de contrôle
  - Coordinateur
    - UAL
    - Mémoire
    - périphériques
- Les registres: mémoires très rapides
- La mémoire cache



## L'unité de traitement Le microprocesseur



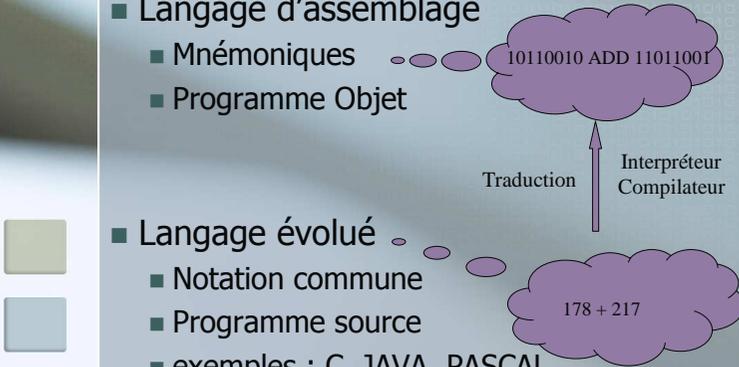
## L'unité de traitement



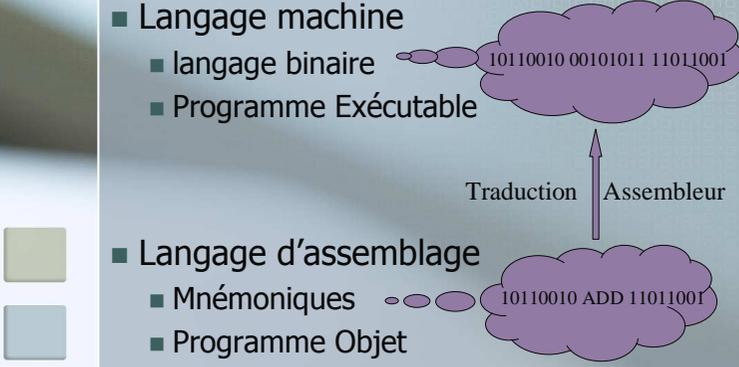
## L'unité de traitement

- fonctionnement de l'unité de contrôle
  - Chercher en mémoire centrale l'instruction à exécuter et la stocker dans un registre
  - décoder l'instruction
  - Charger dans des registres les données nécessaires
  - commander l'exécution par l'UAL
  - récupérer le résultat dans un registre
  - stocker le résultat en mémoire centrale
  - recommencer pour l'instruction suivante au top horloge suivant

## Les programmes et leurs langages

- Langage d'assemblage
    - Mnémoniques
    - Programme Objet
  - Langage évolué
    - Notation commune
    - Programme source
    - exemples : C, JAVA, PASCAL...
- 

## Les programmes et leurs langages

- Langage machine
    - langage binaire
    - Programme Exécutable
  - Langage d'assemblage
    - Mnémoniques
    - Programme Objet
- 

## Les programmes et leurs langages

- Le langage machine et le langage d'assemblage dépendent du processeur (la puce)
- Les langages évolués sont indépendants des machines

## Notation binaire

L'ordinateur manipule exclusivement des informations binaires:

- Ce sont des informations qui n'ont que deux états: ouvert – fermé vrai – faux, etc.
- On symbolise une information binaire par 1 et 0
  - Le 1 et le 0 sont des signes pour désigner une information, indépendamment de son support physique.

Avec une telle information binaire, on ne va pas loin:

- Utiliser les informations binaires par paquet de 8 ou octets.

## Notation binaire

### Définitions

- 8 bits = octet  $2^8 = 256$
- 16 bits = demi-mot  $2^{16} = 65\,536$
- 32 bits = mot  $2^{32} = 4\,294\,967\,296$

Notons que:  $2^{10} = 1024 \approx 10^3$

- 1 Ko = 1 024 octets
- 1 Mo = 1 024 \* 1 024 = 1 048 576 octets
- 1 Go = 1 024 \* 1 024 \* 1 024 = 1 073 741 824 octets

## Notation binaire

### ■ Un octet peut servir à coder 256 entités différentes:

- nombres entiers de 1 à 256
- nombres entiers de 0 à 255
- nombres entiers de -127 à +128
- autre chose qu'un nombre: souvent employé pour du texte
- ...

C'est une affaire de codification ...

- Pour des nombres plus grands que 256, des nombres négatifs ou décimaux on utilise plus d'un octet:
  - 2 octets →  $256 \times 256 = 65\,536$  possibilités
  - 3 octets →  $256 \times 256 \times 256 = 16\,777\,216$  possibilités
  - ...

## Multiples of bytes

SI decimal prefixes			IEC binary prefixes	
Name (Symbol)	Standard SI	Binary usage	Name (Symbol)	Value
kilobyte (kB)	$10^3$	$2^{10}$	kibibyte (KiB)	$2^{10}$
megabyte (MB)	$10^6$	$2^{20}$	mebibyte (MiB)	$2^{20}$
gigabyte (GB)	$10^9$	$2^{30}$	gibibyte (GiB)	$2^{30}$
terabyte (TB)	$10^{12}$	$2^{40}$	tebibyte (TiB)	$2^{40}$
petabyte (PB)	$10^{15}$	$2^{50}$	pebibyte (PiB)	$2^{50}$
exabyte (EB)	$10^{18}$	$2^{60}$	exbibyte (EiB)	$2^{60}$
zettabyte (ZB)	$10^{21}$	$2^{70}$	zebibyte (ZiB)	$2^{70}$
Yottabyte (YB)	$10^{24}$	$2^{80}$	yobibyte (YiB)	$2^{80}$

## Système de numérotation

Qui se rappelle des règles du système de numérotation par position en base 10?

- on retrouve facilement que: **9562** c'est:  
 $9 \times 1000 + 5 \times 100 + 6 \times 10 + 2 \times 1$   
 $(9 \times 10 \times 10 \times 10) + (5 \times 10 \times 10) + (6 \times 10) + (2 \times 1)$   
ou  $9 \times 10^3 + 5 \times 10^2 + 6 \times 10^1 + 2 \times 10^0$
- Deux conséquences:
  - Nous nous servons de dix chiffres, pas un de plus, pas un de moins.
  - La position du chiffre dans un nombre désigne la puissance de dix par laquelle ce chiffre doit être multiplié pour reconstituer le nombre.

## Notation hexadécimal

Représenter un octet par une suite de huit bits n'est pas très pratique!

- On considère un octet comme deux paquets de 4 bits (les quatre de gauche, et les quatre de droite):
- Avec 4 bits nous pouvons coder  $2^4=16$  nombres différents
- Choisir de calculer à base seize  
16 nombres différents se représentent avec un seul chiffre:  
0, 1, 2, ..., 9, A, B, C, D, E, F

## Système de numérotation

Appliquons ce principe au binaire:

- Pour reconstituer le nombre dans la base décimale:  
Prenons un octet : **1 1 0 1 0 0 1 1**
- Ce nombre représente en base dix:  
 $1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$   
 $1 \times 128 + 1 \times 64 + 1 \times 16 + 1 \times 2 + 1 \times 1$   
 $128 + 64 + 16 + 2 + 1$   
**211**
- Inversement pour un nombre en décimal **186**:  
 $1 \times 128 + 0 \times 64 + 1 \times 32 + 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$   
 $1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$   
**1 0 1 1 1 0 1 0**

## Notation hexadécimal

Prenons un octet au hasard : **1 0 0 1 1 1 1 0**

- Première méthode:  
on passe en décimal : 158  
puis en hexadécimal : **9E**
  - Deuxième méthode:  
Divisons **1 0 0 1 1 1 1 0** en :  
**1 0 0 1** (gauche) → c'est 8 + 1, donc 9  
**1 1 1 0** (droite) → c'est 8 + 4 + 2 donc 14=E  
Le nombre s'écrit donc en hexadécimal : **9E**
- Représentation très simple des octets binaire**

## Exercice Numérotation

BINAIRE	OCTAL	DECIMAL	HEXADECIMAL
			11
		11	
	11		
11			
	1		
		74	
			74
	74		
74			
			ABBA
1010101010			
		255	

## Codage des valeurs numériques

### Codage des entiers négatifs (entiers signés)

- Ce codage doit répondre à trois critères:
  - Les nombres négatifs doivent pouvoir être distingués des positifs.
  - La somme d'un nombre et de son opposé est nulle.
  - L'opposé de l'opposé d'un nombre est égal à ce nombre

## Codage des valeurs numériques

- Les entiers naturels peuvent être directement stockés en binaire.
- Les autres types de valeurs nécessitent **un codage**

## Codage en complément à deux.

- Les nombres positifs sont représentés en binaire simple
- Les nombres négatifs sont obtenus de la manière suivante:
  - On inverse les bits de l'écriture binaire de sa valeur absolue (complément à un) puis,
  - On ajoute 1 au résultat (les dépassements sont ignorés).
- Exemple:  
l'opposé de 00011110  
est égal à 11100001+1 = 11100010.
- **Avec ce système, tous les nombres négatifs commencent par un 1 (digit de gauche)**

## Codage des valeurs numériques

- **Pour vérifier que la somme d'un nombre et de son opposé est nulle, il faut d'abord savoir faire une addition!**
- L'addition se pose exactement comme en base 10, avec des retenues (1+1=10, je pose 0 et je retiens 1...).
- Il y a évidemment une astuce: le résultat de l'addition s'écrit lui aussi sur un octet, la dernière retenue n'apparaîtra donc pas dans ce résultat !

## Calcul binaire (exercices)

1. Vérifiez que la somme de deux opposés est nulle en additionnant 00011110 et 11100010, ou 10001100 et 01110100
2. L'opposé de l'opposé d'un nombre est égal à ce nombre: en utilisant la méthode décrite, prenez l'opposé de 11100010, puis celui de 01110100, et vérifiez que le premier est 00011110 et le second 10001100.
3. Vérifiez que la somme de 11100010 (-30) et 01110100 (116) se lit bien 86, et que celle de 10001100 et 00011110 se lit bien -86.

## Addition binaire

### L'addition en binaire se fait avec les mêmes règles qu'en décimale :

- On commence à additionner les bits de poids faible puis on a des retenues lorsque la somme de deux bits de même poids dépasse la valeur de l'unité la plus grande, cette retenue est reportée sur le bit de poids plus fort suivant...

exemple :

0 1 1 0 1	13
+ 0 1 1 1 0	+ 14
-----	-----
1 1 0 1 1	27

## Calcul binaire (exercices)

4. Quel est le plus grand nombre positif que l'on peut écrire dans un octet, avec ce codage?
5. Comment s'écrit son opposé?
6. Prendre l'opposé de 10000000. Que constate-t-on?
7. Ajouter 10000000 et 01111111. Que vaut le résultat ? Quelle valeur faut-il attribuer à 10000000, avec ce codage?
8. Combien de valeurs numériques peut-on écrire dans un octet, avec ce codage? Comparer au binaire ordinaire.

## Multiplication binaire

- La multiplication se fait en formant un produit partiel pour chaque digit du multiplicateur.
- Lorsque le bit du multiplicateur est nul, le produit partiel est nul, lorsqu'il vaut un, le produit partiel est constitué du multiplicande décalé du nombre de positions égal au poids du bit du multiplicateur.

Exemple :

	0 1 0 1	5	multiplicande
x	0 0 1 0	x 2	multiplicateur
	-----		
	0 0 0 0	1 0	
	0 1 0 1		
	0 0 0 0		
	-----		
	0 0 1 0 1 0		