# Introduction aux ordinateurs

# Plan

Notre objective est décrire les données qui sont manipulés par les programmes et la manière d'organiser ses données

# Plan

L'expression « structure de données » peut avoir plusieurs sens, suivant les auteurs. Elle est parfois utilisée pour désigner la façon dont les éléments d'un type sont implantés (l'implantation) mais elle peut aussi désigner le type abstrait ou même la combinaison des deux. Dans ce support de cours, elle désigne le type abstrait et son implantation

# Plan

- Introduction
- Composantes matérielles
- Représentation des données
  - Systèmes de numérotation
    - Système binaire
    - Système octal et hexadécimal
    - Codage de couleur
  - Implementation des nombres
- Paradigmes de Programmation

# Introduction

- Définition d'un ordinateur

  Ordinateur = Machine qui **saisit**, **stocke**, **traite** et **restitue** des informations

**Données**
**Instructions**

**Résultats**

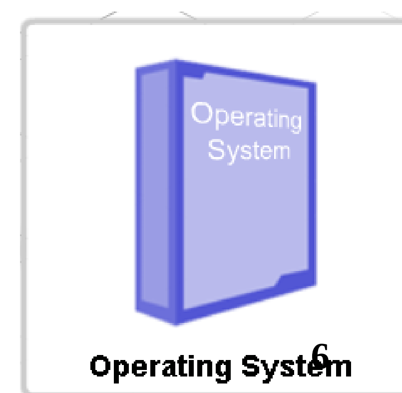
# Introduction

- Comment et ou les ordinateurs sont utilisés?

  Tous les ordinateurs ont des caractéristiques communes

1. Matériel (Hardware) : composantes physiques internes ou externes
2. Système d’exploitation (Operating System) : ensemble de programmes qui pilotent/gèrent le hardware
3. Applications logicielles (Software) : programme qui réalise une fonction précise (Word)





# Système d’exploitation

- Système d’exploitation (OS)

  Les composantes d’un ordinateur et les périphériques ne sont au final qu’une collection de composantes électroniques et mécaniques.

  Un OS est le chef d’orchestre. C’est lui qui détermine :

  - Quel programme utilisateur va être exécuté
  - Comment répartir la mémoire entre les différents programmes
  - Comment lire/enregistrer les données sur les mémoires de masse
  - Les droits de chaque utilisateur du système

  Un OS se comporte comme un traducteur entre les applications de l’utilisateur et le matériel. Il est responsable de la communication entre l’application et le hardware.

# Système d’exploitation

- Système d’exploitation : noyau et shell

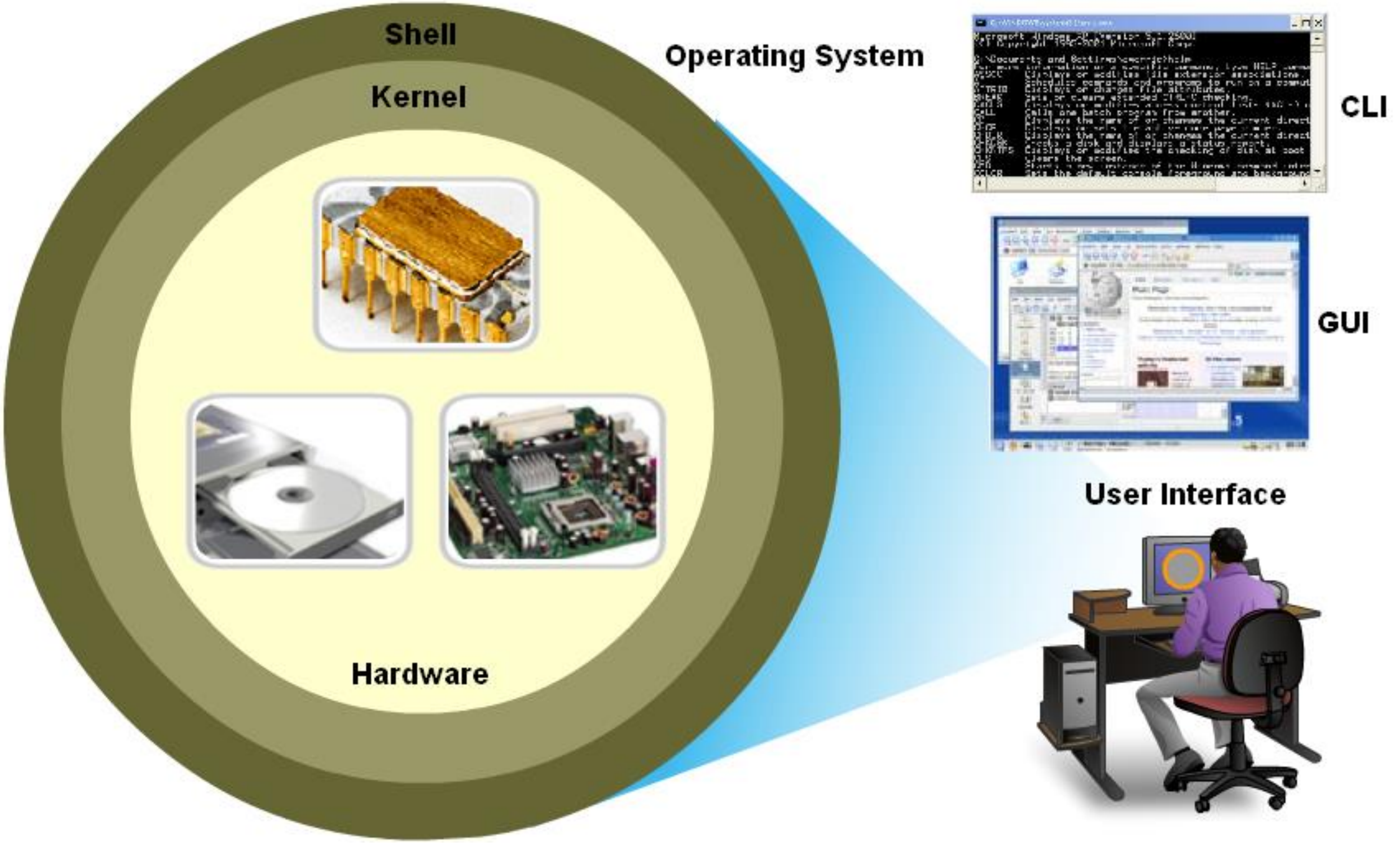
  Lorsqu’un ordinateur est mis sous tension, il charge l’OS, généralement à partir du disque dur.

  - La portion du code de l’OS qui interagit directement avec le hardware est connue sur le nom de noyau (kernel).
  - La portion qui interagit avec l’application et l’utilisateur est connue sous le nom de shell.

  L’utilisateur peut interagir avec le shell soit à travers une interface ligne de commande (CLI pour Command Line Interface) ou un interface graphique (GUI pour Graphical User Interface).

# Système d’exploitation

- Système d’exploitation : noyau et shell



# Système d’exploitation

- Choix d’un OS

Exemples de systèmes d’exploitation

- Microsoft Windows: Seven, Ten, Eleven
- UNIX-Based: IBM AIX, Hewlett Packard HPUX, and Sun Solaris
- BSD - Free BSD
- Linux-Based (Many varieties)
- Macintosh OS X

Les critères du choix d’un OS dépendent de: La disponibilité, portabilité, fiabilité, sécurité

# Composantes matérielles

- Composantes matérielles

  Il existe plusieurs types d'ordinateurs. Ce qui rend un type d'ordinateur plus approprié à une utilisation particulière est les composantes et les périphériques qu'il contient.

# Composantes matérielles

- Composantes matérielles

  Un ordinateur aujourd'hui est composé :

  - D'une carte mère
  - D'un processeur
  - D'une mémoire vive
  - De mémoires de masse
  - De cartes d'extension
  - De périphériques d'entrée/sortie
  - Connecteurs

# Composantes matérielles

- Carte mère

  Carte électronique qui permet aux différents composants de communiquer via différents bus de communication

# Composantes matérielles

- Processeur ou CPU (Central Processing Unit)

  C’est le “cerveau” de l’ordinateur. Il contient les différents composants (dont l’unité de calculs (UC), le décodeur d’instruction, etc.).

  Deux facteurs importants lors de la sélection d’un CPU : la vitesse du processeur et celle du bus.

  - La vitesse du processeur mesure la capacité de traitement de l’information (MHz ou GHz).
  - La vitesse du bus donne la vitesse de transfert des données entre les différentes mémoires.

- https://www.ldlc.com/fiche/PB00421683.html
- https://versus.com/fr/intel-core-i9-11900k-vs-intel-core-i9-9980xe

# Composantes matérielles

- Mémoire

  La mémoire est organisée en cellules (octets ou mots). Chaque cellule est repérée par son adresse qui permet à l'ordinateur de trouver les informations dont il a besoin

  - 2 Modes d'accès à la mémoire : Lecture (aucun effet sur le contenu) + Ecriture (modifie son contenu)
  - Caractéristiques
    - Capacité : nombre d'octets
    - Accès
      - direct : grâce à l'adresse, accès immédiat à l'information (on parle de support adressable)
      - séquentiel : pour accéder à une information, il faut avoir lu toutes les précédentes (ex : cassette audio)
    - Temps d'accès : temps écoulé entre l'instant où l'information est demandée et celui où elle est disponible (en ms)

# Composantes matérielles

- Mémoire

Il existent plusieurs types de mémoire (les plus importantes)

- Mémoire vive (dite aussi RAM)
- Mémoire de masse (non-volatile)
- Mémoire cache
- Registres

Unités de mesure

1octet = 8 bits

1Ko (kilo octet) ≈1 000 octets (exactement $2^{10}$ octets)

1Mo (méga octet) ≈ 1 000 000 octets ($2^{20}$ octets)

1Go (giga octet) ≈ 1 000 000 000 octets ($2^{30}$ octets)

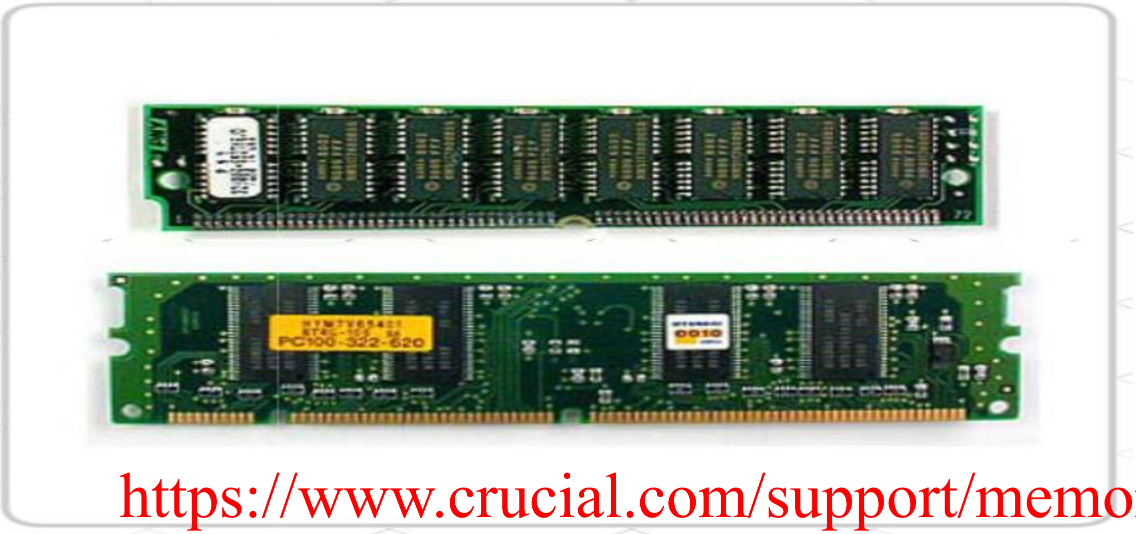
1To (téra octet) ≈ 1 000 000 000 000 octets ($2^{40}$ octets)

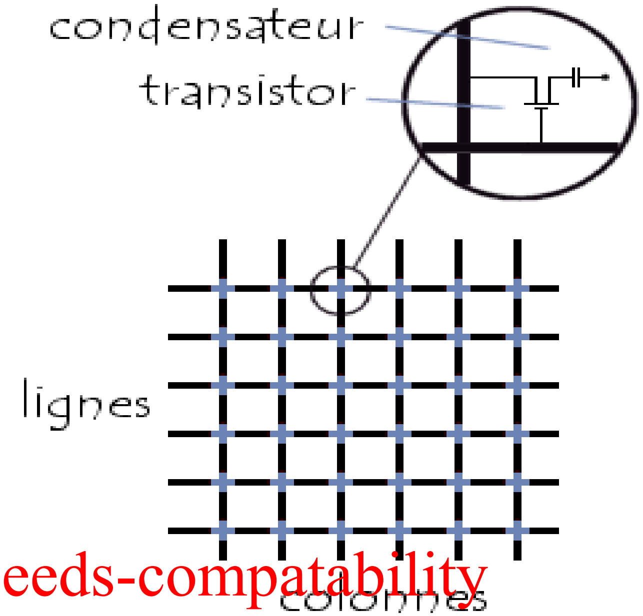
# Composantes matérielles

- Mémoire vive

  Généralement des “barrettes” qui s’enfichent sur la carte mère :

  - C’est une mémoire rapide (RAM) de “petite” capacité
  - Perd son information lorsque l’on coupe le courant
  - Exemple : SDRAM, SIMM, DIMM, DDRAM, RDRAM, DDR4/5/6, etc.




https://www.crucial.com/support/memory-speeds-compatability
https://www.eatyourbytes.com/ddr4-ram-data-transfers-speed-latency-and-voltage/

# Composantes matérielles

- Mémoire de masse

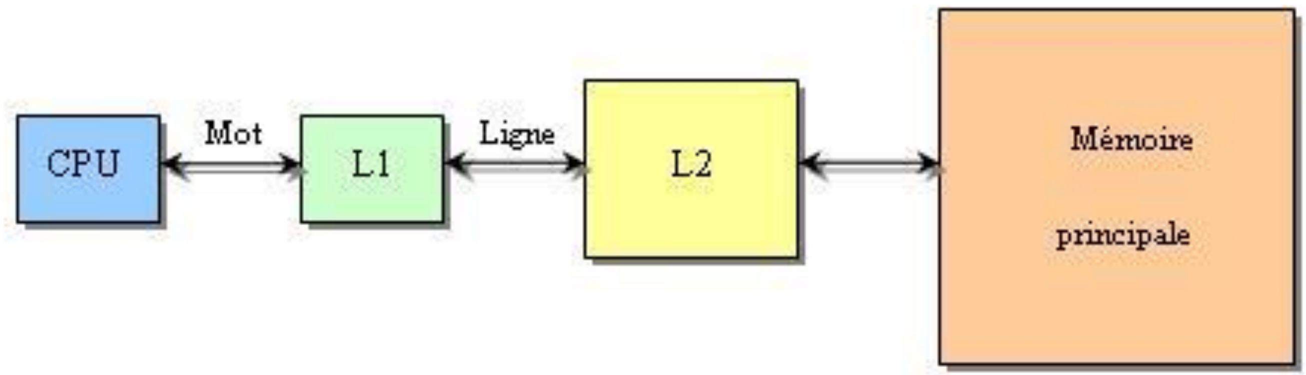
  Mémoire “lente” mais de grande capacité

  - N’a pas besoin de courant pour garder l’information
  - Exemple : Disquette, Disque Dur, SSD, Clé USB, CD-ROM, DVD, etc.

# Composantes matérielles

- Mémoire cache

  La RAM est rapide, mais le microprocesseur l'est encore plus ! Afin de ne pas limiter ses performances en l'obligeant à attendre (on parle de goulot d'étranglement), on utilise de petites unités de mémoires, beaucoup plus rapides, mais nettement plus chères !




  Ses mémoires cache servent à conserver un court instant des informations fréquemment consultées. Ordres de grandeur :

  - capacité : quelques ko (L1) à quelques Mo (L2)
  - vitesse : jusqu'à 5 Go/s

# Composantes matérielles

- Registre

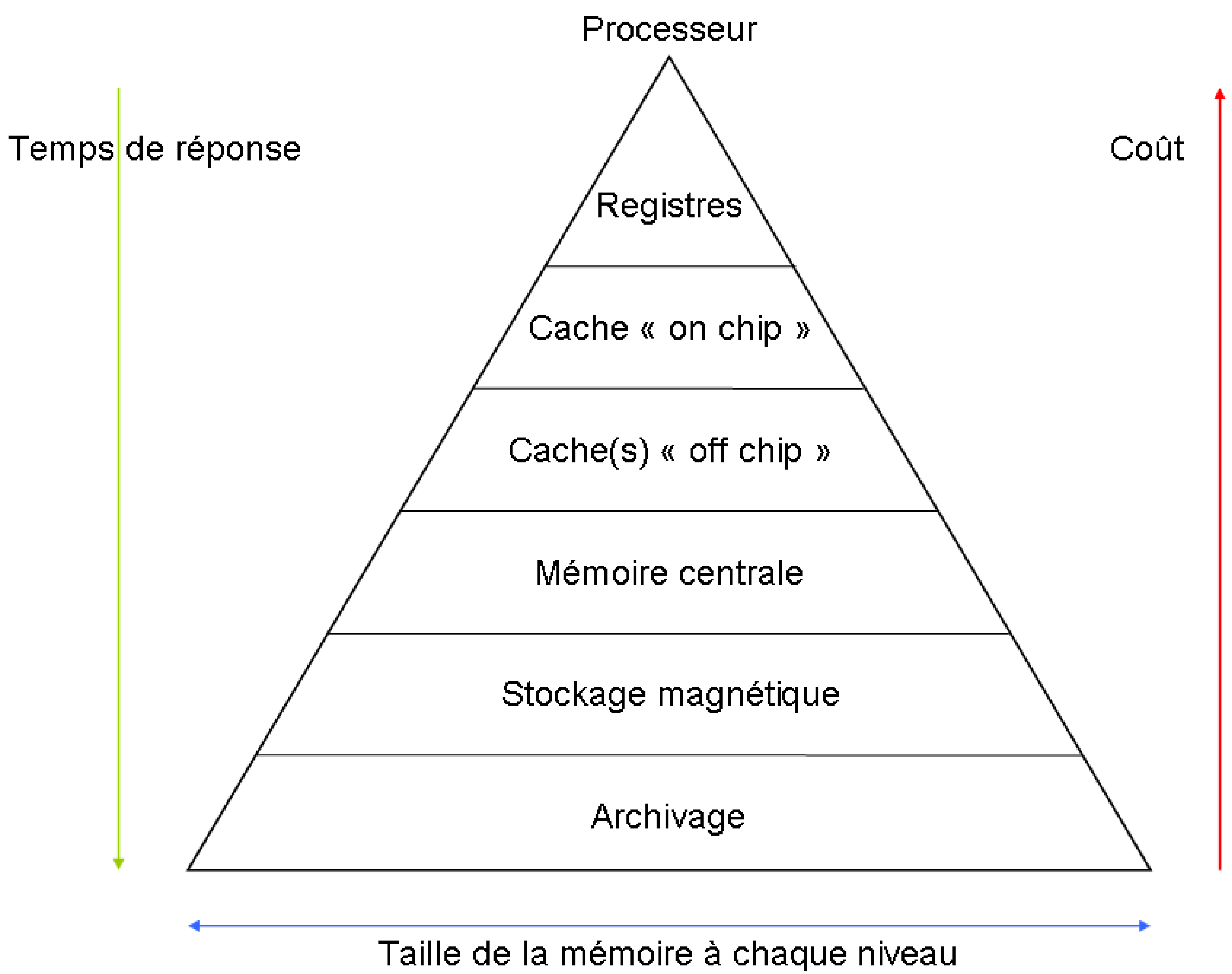
  Il est intégré au processeur. Ce type de mémoire est très rapide mais aussi très cher et est donc réservé à une très faible quantité de données.

  Ordres de grandeur :

  - capacité : quelques dizaines d'octets
  - vitesse : jusqu'à 30 Go/s

# Composantes matérielles

- Mémoire



# Composantes matérielles

- Cartes d’extension

  Permet d’ajouter des fonctionnalités (souvent de communication) comme par exemple les cartes graphiques, réseau, son, modem, usb, etc.

# Composantes matérielles

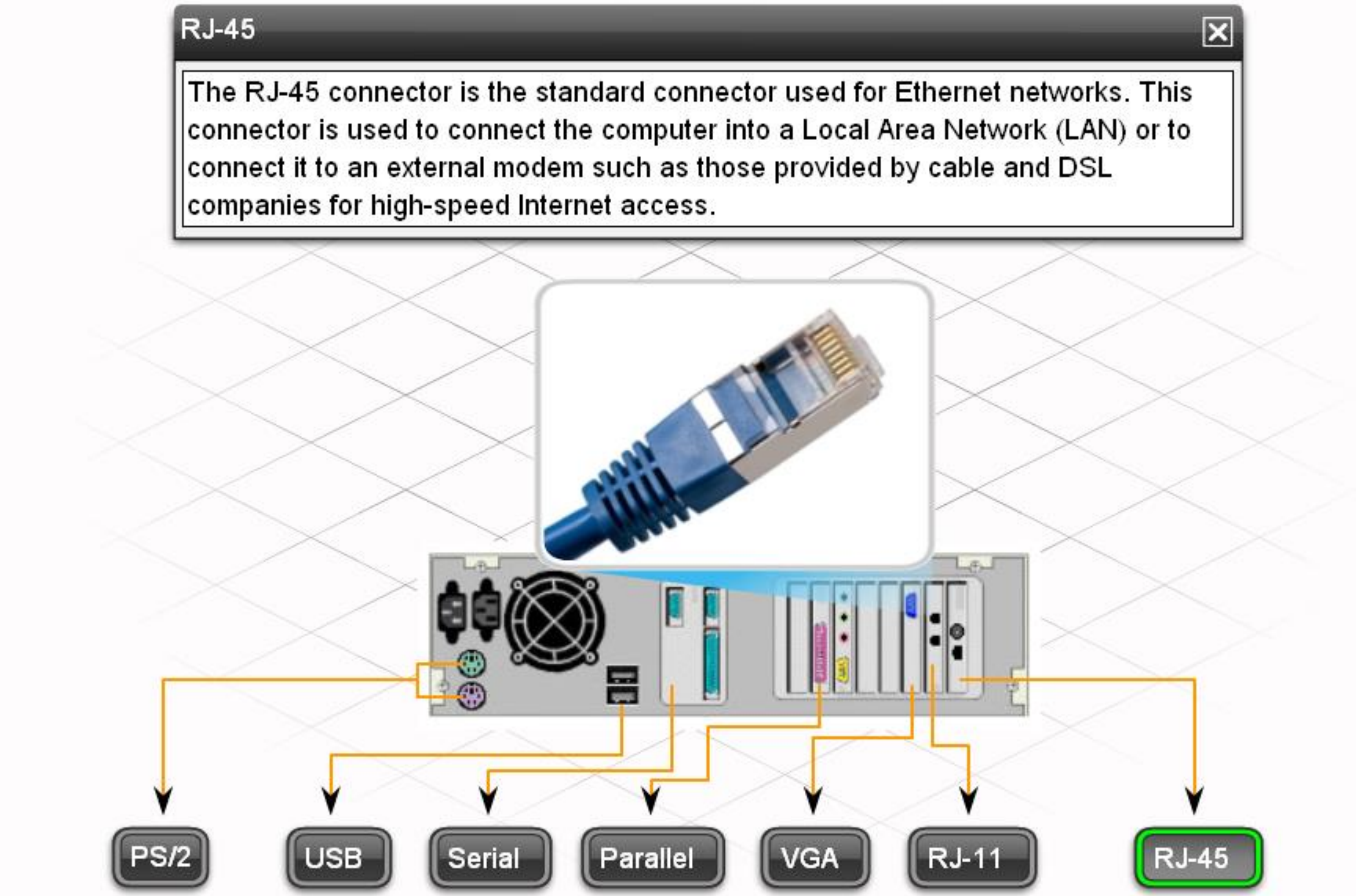
- Périphériques d’entrée/sortie

  Ce sont les composants électroniques qui permettent à l’ordinateur de communiquer avec l’extérieur (utilisateur ou autre ordinateur)

  - Périphériques d’entrée : Clavier, Souris, Scanner, Ecran tactile, carte réseau, mémoires de masse, etc.
  - Périphériques de sortie : Ecran (tactile ou non), Imprimante, carte réseau, mémoires de masse, etc.

# Composantes matérielles

- Les connecteurs

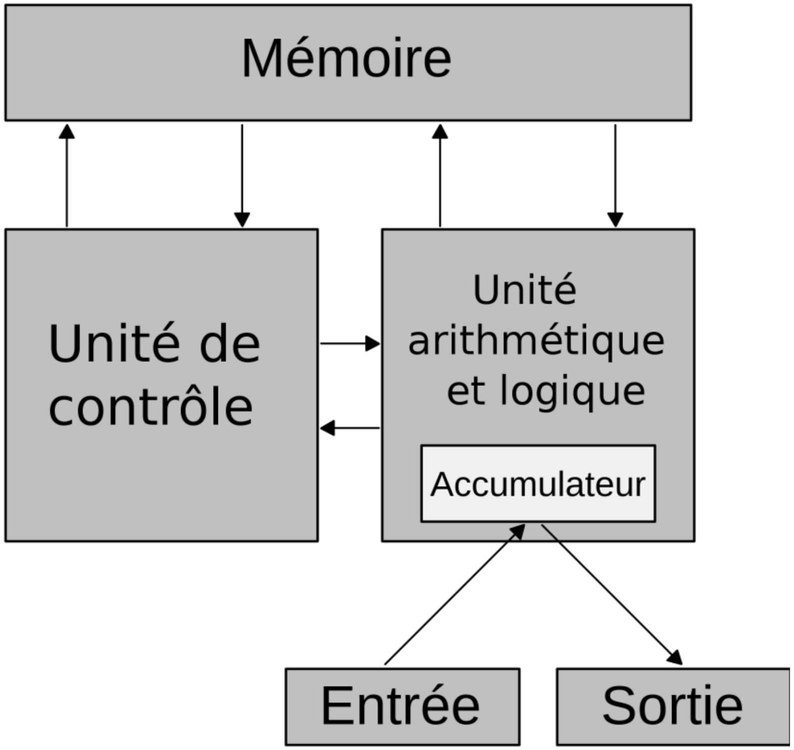


# Composantes matérielles

- Architecture de von Neumann

C'est un modèle structurel d'ordinateur dans lequel une unité de stockage (mémoire) unique sert à conserver à la fois les instructions et les données demandées ou produites par le calcul.

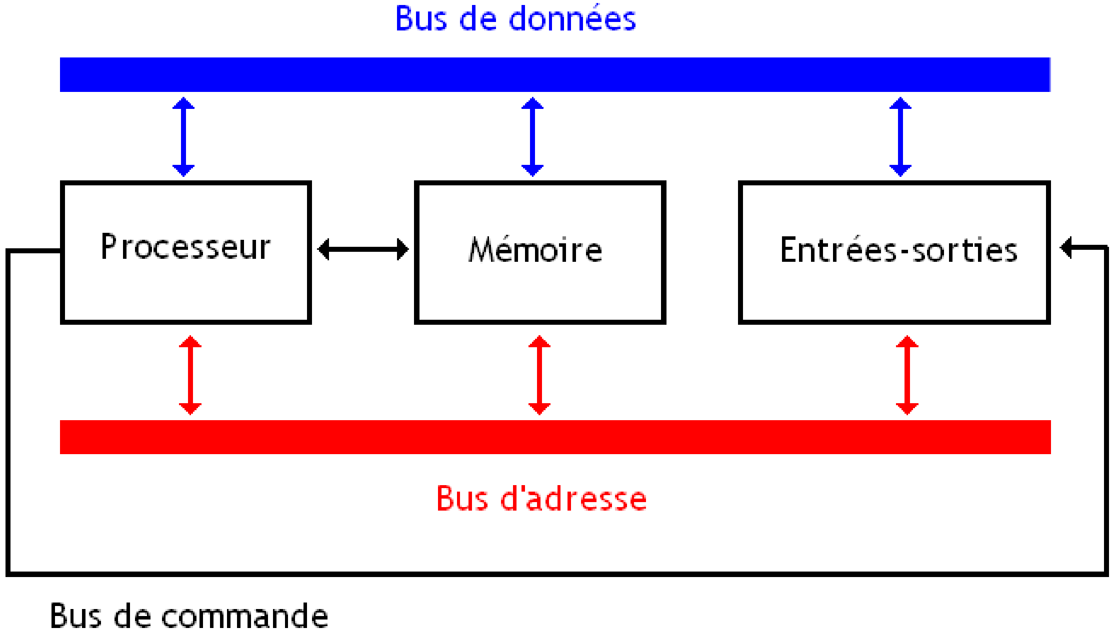
Les ordinateurs actuels sont tous basés sur des versions améliorées de cette architecture.



# Composantes matérielles

- Architecture de von Neumann

Pour que les données circulent entre les différentes parties d'un ordinateur, il existe des systèmes de communication appelés bus




- Le bus d'adresse: permet de faire circuler des adresses par exemple l'adresse d'une donnée à aller chercher en mémoire
- Le bus de données permet de faire circuler des données
- Le bus de contrôle permet de spécifier le type d'action, exemples : écriture d'une donnée en mémoire, lecture d'une donnée en mémoire.

# Applications logicielles

- Application logicielle

  Dans ce cours,

  une application logicielle = Application

  Ce sont des programmes qui sont lancés par l'utilisateur. Par exemple :

  - Editeur de texte : word, edit, …
  - Nafigateur : Interet Explorer, Firefox
  - Messagerie instantanée : msn, skype
  - Compilateur
  - Interface graphique

# Applications logicielles

- Application logicielle

A chaque tâche, que notre PC réalise, correspond un programme.

Mais c’est quoi un programme?

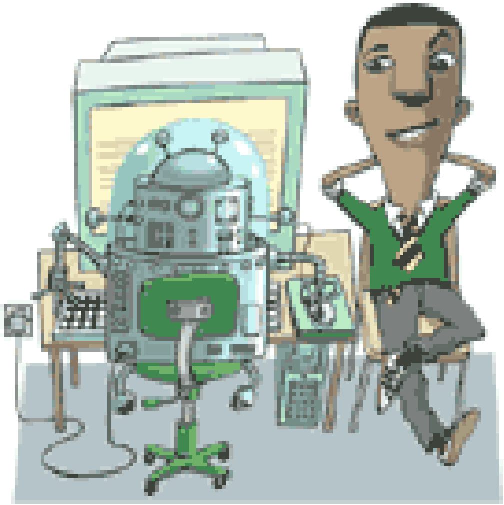
# Applications logicielles

- Application logicielle

  Un programme est constitué d'une suite d'instructions compréhensible par l'ordinateur (en langage machine).

  Une instruction spécifie :

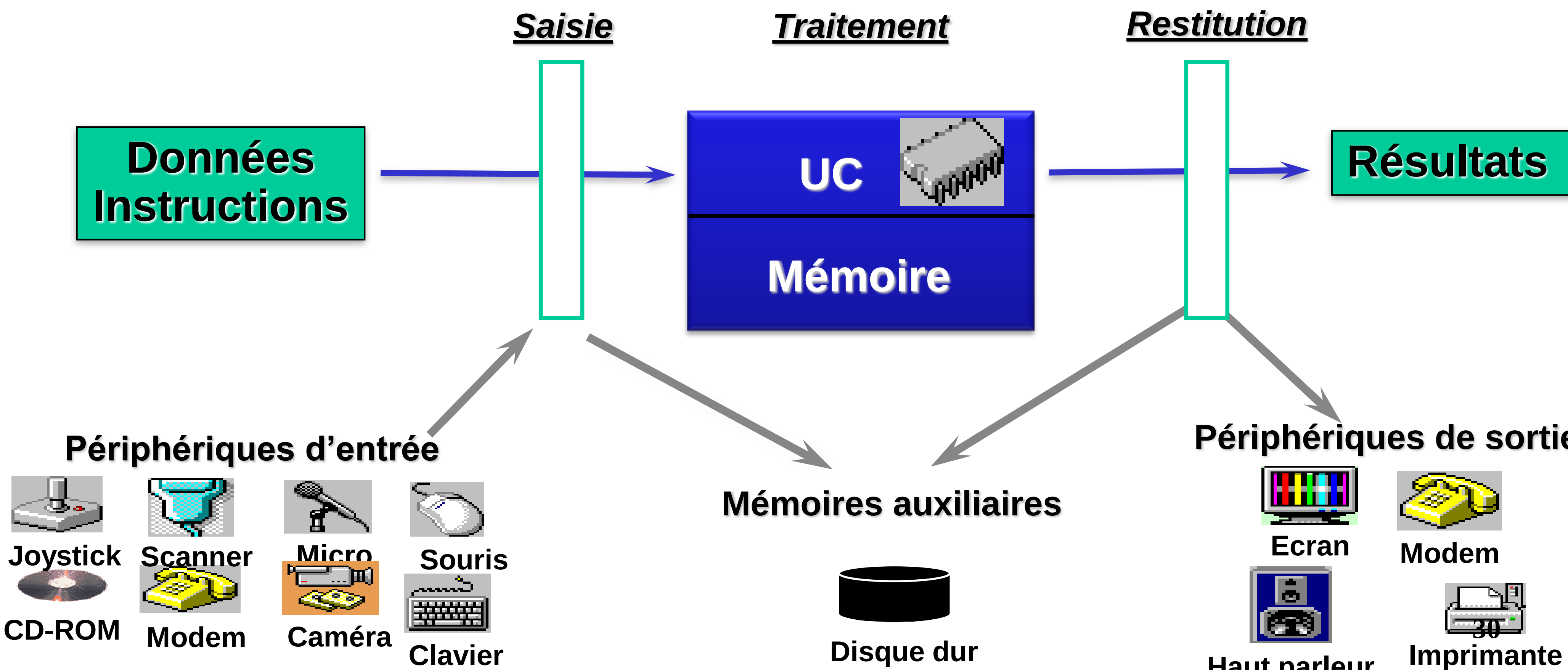
  - Les opérations à exécuter
  - La façon dont elles s'enchaînent



# Introduction

- Fonctionnement d'un ordinateur

  Ordinateur = Machine qui saisit (périphériques d'entrée), stocke (mémoire), traite (programmes) et restitue (périphériques de sortie) des informations



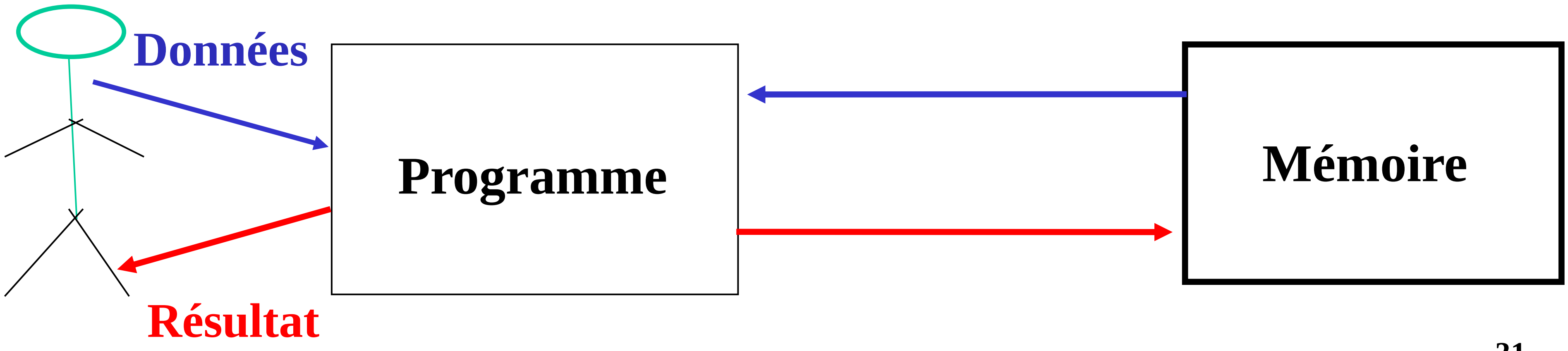
# Applications logicielles

- Fonctionnement d’un ordinateur

  Un programme peut nécessité des données. Il peut aussi retourner un résultat.

  Exemple: on dispose d’un programme qui calcule la moyenne des notes.

  - Celui-ci a besoin qu’on lui fournisse les notes (données)
  - Pour qu’il nous retourne la moyenne (résultat)

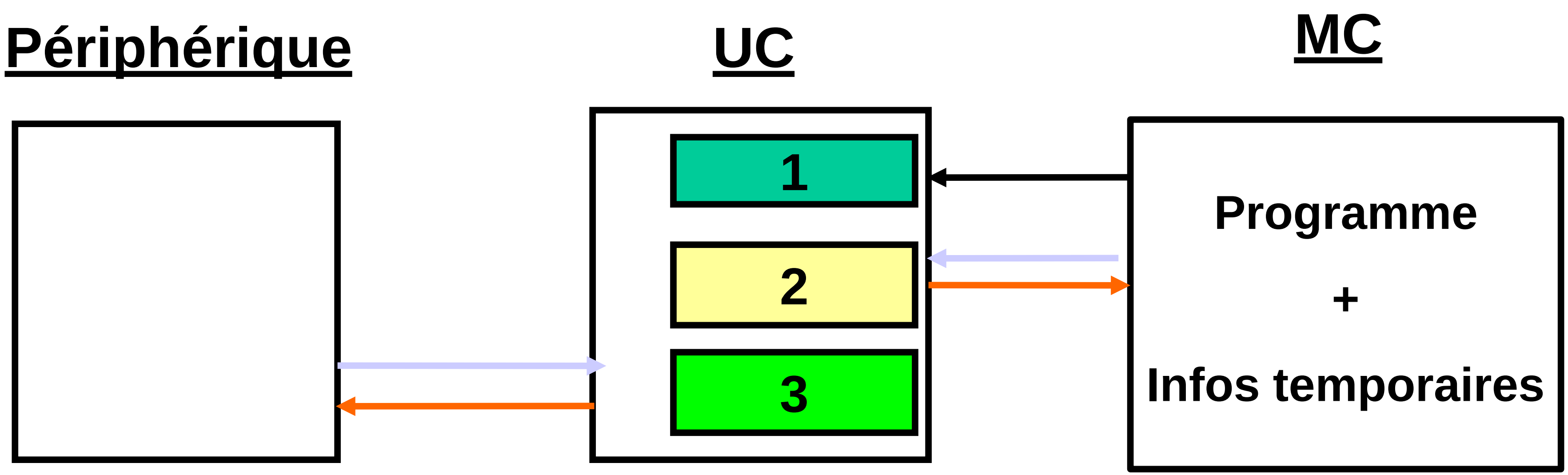


# Applications logicielles

- Exécution simplifiée d'un programme

  UC = Unité centrale ou CPU

  MC = Mémoire centrale




1) Prélèvement d'une instruction

2) Exécution de l'instruction avec possibilité d'échange avec la MC

3) Exécution d'une instruction d'échange avec un périphérique

# Applications logicielles

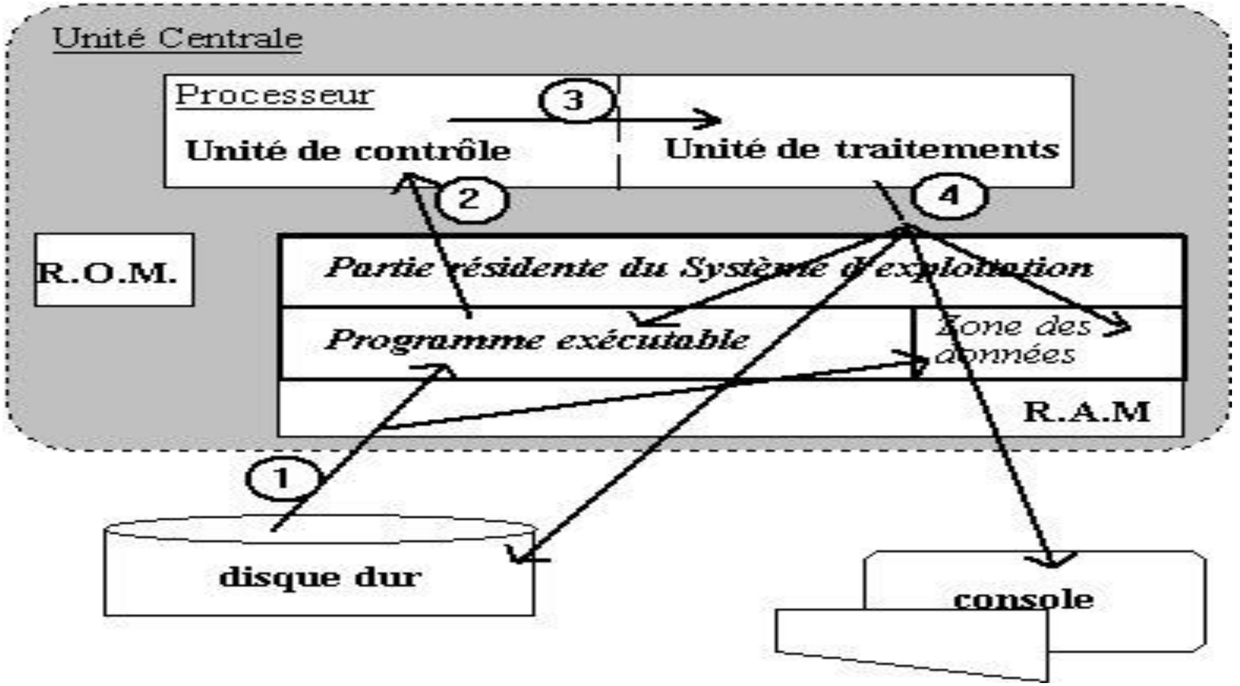
- Exécution simplifiée d'un programme

  L'ordinateur traite l'information grâce à un programme qu'il mémorise. Il communique et stocke des informations

- Mémoire centrale: contient le programme et informations temporaires
- Unité centrale: chargée de prélever une à une les instructions du programme
  - Contient
    - des circuits (circuit de calcul UAL, unité de contrôle ou de commande)
    - des mémoires liées aux circuits (nommées registres)
  - Deux types d'instructions
    - Opérations internes (addition, soustraction, …)
    - Opérations de communication (affichage, stockage, …)
- Périphériques: d'entrée, de sortie, d'entrée/sortie

# Applications logicielles

- Exécution simplifiée d’un programme
  - Mémoire centrale: contient le programme et informations temporaires
  - Unité centrale: chargée de prélever les instructions du programme
    - des circuits (circuit de calcul UAL, unité de contrôle)
    - des mémoires liées aux circuits (nommées registres)



# Plan

- Introduction
- Composantes matérielles
- Représentation des données
  - Systèmes de numérotation
    - Système binaire
    - Système octal et hexadécimal
    - Codage de couleur
  - Implementation des nombres
- Paradigmes de Programmation

# Représentation des données

- Exécution simplifiée d’un programme

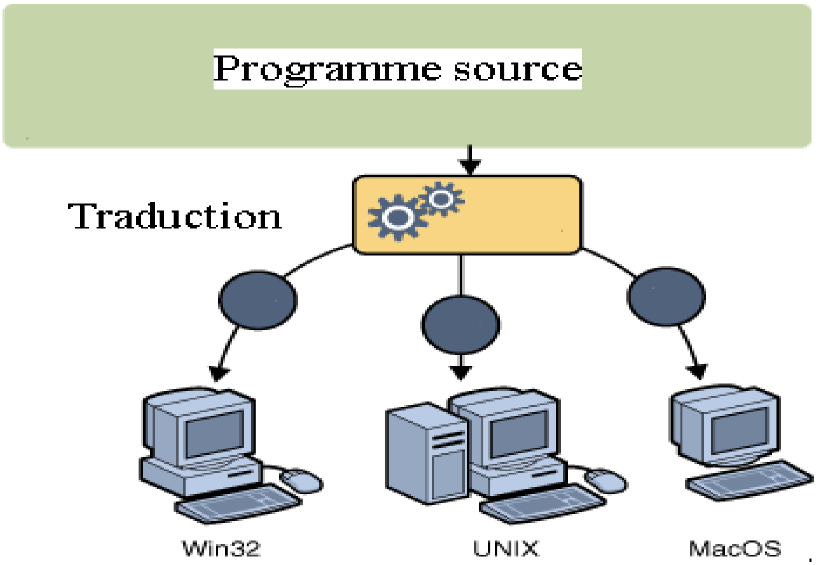
Pour calculer 12+5, il faut une suite d'instructions

  - Transférer:
    - le nombre 12 saisi au clavier dans la mémoire
    - le nombre 5 saisi au clavier dans la mémoire
    - le nombre 12 de la mémoire vers un registre A
    - le nombre 5 de la mémoire vers un registre B
  - Demander à l'unité de calcul (UAL  unité arithmétique et logique) de faire l'addition
  - Transférer:
    - le contenu du résultat dans la mémoire
    - le résultat (17) se trouvant en mémoire vers l'écran de la console (pour l'affichage)

# Représentation des données

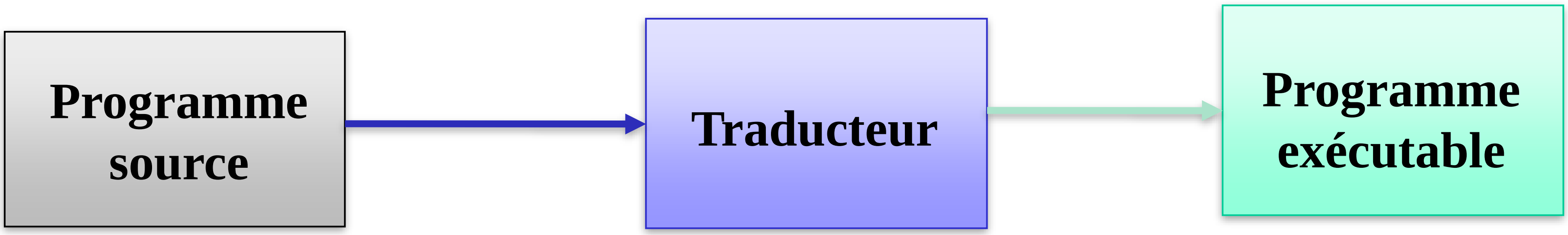
- Programmation?

- L'ordinateur ne comprend que le binaire (0 et1), est-ce pour autant qu'on doive écrire des programmes en binaire ?
- Il existe des langages de programmation dits « évolués » (proches du langage courant)
- Pour chaque langage, il existe un programme « qui le traduit » en binaire (ou langage machine)



# Représentation des données

- Traduction des programmes




Il existe essentiellement deux modes de traduction

- Compilation: la traduction se fait une fois pour toute
- Interprétation: a chaque fois qu’on veut exécuter le programme, l’interprète traduit une instruction à la fois. Une fois que celle-ci est exécutée, il passe à l’instruction suivante.

# Représentation des données

Mais avant de parler programmation, il faut voir comment l’information est représentée dans l’ordinateur

# Représentation des données

- Notions de codage

  Exemple 1 :

  Nous utilisons le codage chaque jour

  - 11, onze, XI

  Nous avons interprété XI par le nombre 11. Mais, comment a-t-on pu dire que ce ne sont pas les lettres X et I ?

# Représentation des données

- Notions de codage

  Exemple 2:

  Si on demande le nom correspondant à l'image :

  - À un arabophone, il répond : بيت
  - À un francophone, il répond : maison
  - À un anglo-saxon, il répond : house

  Mais c'est toujours la même image, seule le nom la désignant change

# Représentation des données

- Notions de codage

  Exemple 3 : les Shadok

  Les Shadoks est une série télévisée d'animation française en 208 épisodes de deux à trois minutes, diffusée entre le 29 avril 1968 et 1973 (trois premières saisons) et à partir de janvier 2000 (quatrième saison) sur Canal+ et rediffusée sur Cartoon Network.

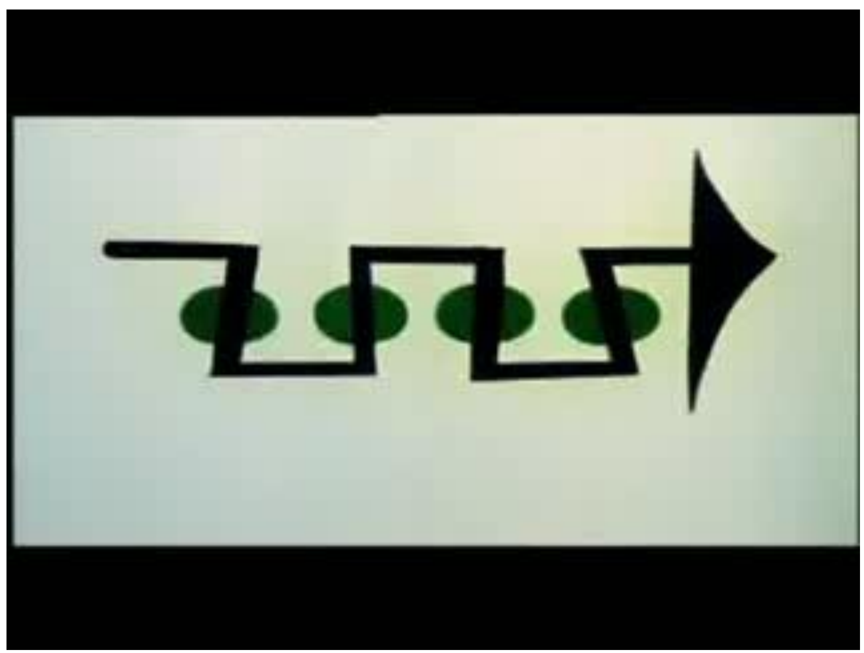
  Cette série raconte les histoires des Shadoks, une sorte d'oiseaux rondouillards avec de longues pattes et de petites ailes ridicules,

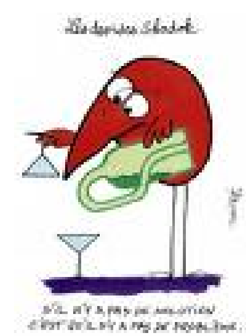
  Les Shadoks possèdent pour tout vocabulaire quatre mots monosyllabiques : « ga, bu, zo, meu ». Les Shadoks sont excessivement méchants et idiots..

# Représentation des données

- Notions de codage chez les Shadok

Le professeur Shadoko avait inventé exactement le même système de comptage que les humains, la seule différence étant qu'il avait choisi la base 4 (normal, les shadoks n'avaient que 4 mots).





**https://www.youtube.com/watch?v=lP9PaDs2xgQ**

# Représentation des données

- Codage binaire

Le langage des ordinateurs : Tout est 0 ou 1

Toutes les informations traitées par l'ordinateur sont en binaire

- Quand on tape sur une touche du clavier, l'ordinateur la transforme en binaire (suite de zéros et de uns)
- Quand l'ordinateur affiche sur l'écran un résultat, il fait l'opération inverse

Les humains interprètent les mots et les figures, les ordinateurs interprètent les zéro et les uns. Lorsqu'on lit 'A' dans un texte, l'ordinateur lit '01000001' (codage ASCII)

# Représentation des données

- Codage binaire

  Il est important de distinguer le concept de nombre de sa représentation graphique

- La représentation graphique d'un nombre dépend :
  - des symboles utilisés (les chiffres)
  - de la base utilisée (le nombre de chiffres disponibles)
- Un même nombre peut être représenté dans plusieurs bases. Par exemple, le nombre 123 est représenté, en utilisant les chiffres arabes:
  - 123 en base 10 (decimal)
  - 1111011 en base 2 (binaire)

# Représentation des données

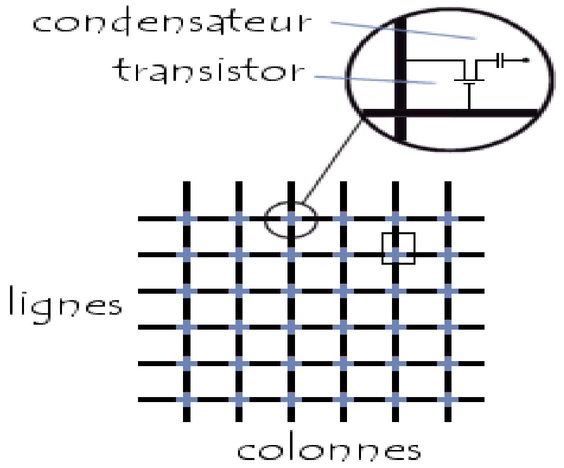
- Codage binaire et composants électroniques : exemple

Interrupteurs

Composantes électroniques laissant passer le courant principal lorsque la tension sur le fil de commande est de 5V


Mémoires

Composants électroniques capables de mémoriser des tensions (0 ou 5V)




Toutes communications à l'intérieur de l'ordinateur sont faites avec des signaux électriques

- 0: éteint (absence de signal électrique)
- 1: allumé (présence de signal électrique)

# Représentation des données

- Codage binaire et composants électroniques

  On a définit la mémoire comme étant un composant électronique capable de mémoriser des tensions

  On peut assigner deux valeurs à une mémoire :

  - 0 lorsque la tension est de 0V
  - 1 lorsque la tension est de 5V

  On appelle ce type de mémoire un bit (Binary digIT) : unité de stockage élémentaire.

# Représentation des données

- Codage binaire

Si on dispose de deux bits, le nombre total d'états possibles que peuvent prendre ces deux bits est quatre : 00, 01, 10 ou 11 ($2^2$)

Avec trois bits le nombre total d'états possibles est huit :

000, 001, 010, 011, 100, 101, 110, 111 ($2^3$)

Avec **n** bits on peut avoir $2^n$ états possibles

# Représentation des données

- Codage des caractères
- Les caractères alphanumériques en anglais :
  - a,b,c,…,y,z = 26 ,     A,B,C,…,Y,Z = 26,   0,1,…9 = 10
- Les opérations +,-,*,/,<,>…= (=15)
- Ponctuation . , ; : ! ? » ()[]{}_... = 20
- Caractère spéciaux: $ £ & @.... = 10
- Signes invisibles: retour à la ligne, espace... = 5

Au total environ 112

Pour associer à un caractère une suite distincte de 0 et 1 qui le code, nous avons besoin au minimum de 7 bits, puisque $2^7 = 128$

# Représentation des données

- Codage des caractères
- Certaines langues utilisent en outre d’autres caractères : é, è, à, ç, æ, ë, Ñ, Ã, õ, β,……
- Il est donc plus raisonnable de conserver une marge de manœuvre, et de prendre plutôt 8 bits, qui permettent 256 combinaisons différentes.
- Un groupe de huit bits s’appelle un octet (en anglais, byte)
  - méfiance avec le byte qui vaut un octet, c'est-à-dire huit bits
- Si on veut coder des nombres de grande taille, des nombres négatifs, des nombres décimaux, ou d’autre caractères, alors il faut mobiliser plus d’un octet.
  - Avec deux octets, on a 256 x 256 = 65 536 possibilités
  - Avec trois octets, on passe à 256 x 256 x 256 = 16 777 216

# Représentation des données

- Codage des caractères
- Quel caractère doit être représenté et par quel état de l'octet?
- Si ce choix était librement laissé à chaque fabricant d'ordinateur, alors :
  - Il faudrait des programmes spécifiques pour chaque type d'attribution
  - La communication entre deux ordinateurs serait difficilement réalisable

# Représentation des données

- Codage des caractères

  Il existe des standards internationaux de codage à l'exemple de

  - ASCII (pour American Standard For Communication and International Interchange)
  - Son rôle est la stipulation de quel état de l'octet correspond à quel signe du clavier

| Caractère | Code ASCII |
|---|---|
| 1 | 00110001 |
| z | 01101010 |
| $ | 00100100 |
| { | 01111011 |

# Représentation des données

- Codage ASCII

| Dec | Hx | Oct | Char | | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL | (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH | (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX | (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX | (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT | (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ | (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK | (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL | (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS | (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB | (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF | (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT | (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF | (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR | (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO | (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI | (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE | (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 | (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 | (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 | (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 | (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK | (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN | (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB | (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN | (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM | (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB | (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC | (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS | (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS | (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS | (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US | (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Source: www.LookupTables.com

# Représentation des données

- Systèmes de numérotation:

  Il existe plusieurs systèmes de numérotation

  - Système décimal
  - Système binaire
  - Système octal
  - Système hexadécimal
  - Système des couleurs
  - ….

# Représentation des données

- Systèmes de numérotation:

  **Base** d'un système de numération : C'est le **regroupement** de signes différents utilisés par ce système pour coder l'information

  **La base** correspond souvent au nombre de signes différents de ce système

  - La base dix utilise 10 signes (chiffres de 0 à 9) différents pour représenter un nombre
  - Le système binaire utilise " 2 " signes (0, 1)
  - …

  **Rang** d'un chiffre dans un nombre : c'est la place de ce chiffre dans le nombre en comptant à partir de la **droite**

  - Il définit le "poids" du chiffre dans le nombre

# Représentation des données

- Systèmes de numérotation: décimal

Le système décimal est un système de numération de base 10 (utilise les chiffres de **0 à 9)**

Rappel : tout nombre décimal peut s'écrire sous forme d'une somme de facteurs de $\mathbf{10^n}$

Exemple :

$$\mathbf{6523 = 6*10^3 + 5*10^2 + 2*10^1 + 3*10^0}$$

# Représentation des données

- Systèmes de numérotation: binaire

Dans le système binaire, on utilise 2 chiffres :

  - le système binaire est donc un système de numération de base 2
  - Ainsi les nombres binaires se décomposent (en décimal) en une somme de facteurs de **$2^n$**
  - Exemple :

    Le nombre binaire 100111 correspond (en décimal) à ?

$$\mathbf{(100111)_2 = 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0}$$

# Représentation des données

- Systèmes de numérotation: binaire et opérations

  L'addition en binaire de deux nombres consiste à effectuer l'addition binaire sur les bits de même poids de chaque nombre reportant de droite à gauche les retenues successives

- Exemples d'addition :
  - 1011+ 0100 = 1111
  - 010 + 111 = 1001
  - 0011 + 0111 = 1010

# Représentation des données

- Systèmes de numérotation: binaire et opérations

On peut résumer ces opérations à l'aide d'une table de vérité

- L'addition correspond à l'opérateur booléen **XOR** (OU-exclusif) et que la retenue peut elle être calculée à l'aide de l'opérateur **ET**

| A | B | XOR (A+B) | ET |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 |

# Représentation des données

- Systèmes de numérotation: octal

Le système octale utilise 8 chiffres de 0 à 7: système de base **8**

$17_8 = 1 * 8^1 + 7 * 8^0 = 15_{10}$

**$117_8 = ?$**

# Représentation des données

- Systèmes de numérotation: hexadécimal

  Le système hexadécimal utilise les chiffres 0, 1, ..., 9, A, B, C, D, E, F : système de base 16

  - A, B, C, D, E, F représentent dans le système décimal 10, ..., 15

  La base hexadécimale est introduite pour abréger l'écrire des nombres des bases binaire et octale

  1 000 000 000 000 binaire = 10000 octal = 1000 hexadécimal

  Le codage hexadécimal est très souvent utilisé quand on a besoin de représenter les octets individuellement, car dans ce codage, tout octet correspond à seulement deux signes

  - Voir exemple de codage de couleurs (plus loin)

# Représentation des données

- Systèmes de numérotation: RBG ( Red, Blue and Green)

  RGB (*Red Green Blue*) est un format de codage des couleurs

  - se présente comme un nombre hexadécimal à six chiffres : FF06C3 par exemple
  - Chaque paire de chiffres est dédiée à une couleur primaire. Sur le même exemple, cela donne :
    - FF pour le rouge ;
    - 06 pour le vert ;
    - C3 pour le bleu ;
  - 000000 : noir
  - FFFFFF : blanc
  - Combien de couleurs sont possibles? combien d'octets en mémoire?

# Représentation des données

- Systèmes de numérotation: d'une façon générale

En base **b**, on utilise les nombres **0, 1, ..., b-1**

# Représentation des données

- Changement de base : d'une façon générale

supposer que vous avez deux bases a et b et deux nombre

  - X dans la base b
  - Y dans la base a
  - Z dans la base décimale

Donner le codage de :

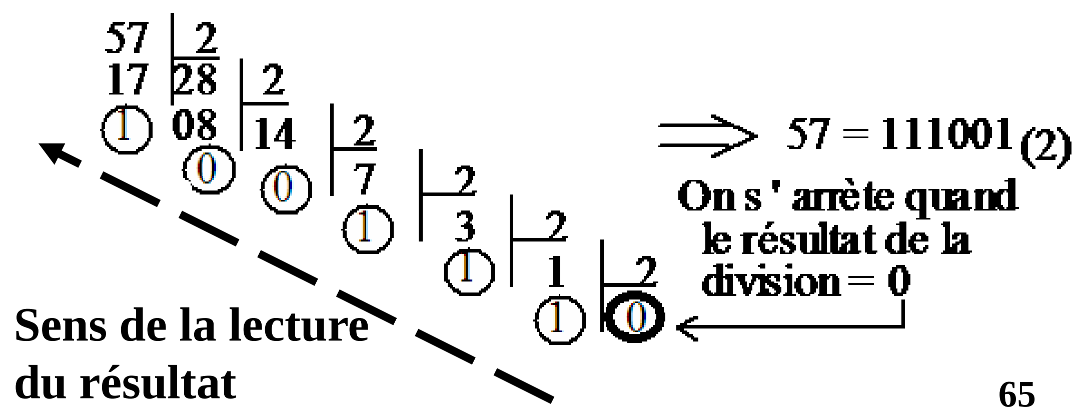
  - Z dans la base a et son codage dans la base b
  - X et Y dans la base 10
  - X dans la base a
  - Y dans la base b

# Représentation des données

- Changement de base : du décimal à la base b

  Règles de conversion de la base décimale vers la base **b** :

  - On divise le nombre par **b**, puis le quotient obtenu par **b**, et ainsi de suite jusqu’a obtention d’un **quotient nul**
  - La suite des restes obtenus correspond aux chiffres dans la base **b** visée
  - Exemple avec b=2
    - Exprimer en binaire 57



# Représentation des données

- Changement de base : de la base b au décimal

Règles de conversion de la base **b** au décimal :

$$\left(a_n a_{n-1} \ldots a_1 a_0\right)_b = a_n b^n + a_{n-1} b^{n-1} + \ldots . + a_0 b^0$$

en base b                en base 10

On calcule la somme des puissances de b multipliées par le chiffre correspondant à la puissance

- Exemple avec b=2

**$(100111)_2 = 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0$**

# Représentation des données

- Changement de base : de la base « b » à la base « a »

Règles de conversion de la base **b à la base a** :

- Passage par le décimal : de la base b au décimal puis du décimal à a

# Représentation des données

- Changement de base : de la base « b » à la base « a »

  Exemples avec **binaire** => **Hexadécimal et** avec **octal** => **binaire**

  - Première méthode de conversion :
    - Passage par la base décimale :
      - Utiliser la divisions pour passer de binaire vers le décimal et de décimal vers le hexadécimal/octal

NB: Au besoin, nous pouvons rajouter des 0 à gauche

# Représentation des données

- Changement de base : de la base « b » à la base « a »

  Exemple avec **binaire => hexa/octal**

  - Deuxième méthode de conversion :
    - Directement du binaire vers l'hexadécimal /octal :
      - Par regroupement de bits
      - Dans le cas d'un passage à la base 16, on regroupe les bits à partir de la droite par tranche de quatre bits puis on code chaque regroupement dans la base hexadécimale
      - Dans le cas d'un passage à la base 8 les bits sont regroupés par 3 à partir de la droite puis on code chaque regroupement dans la base octale

NB: Au besoin, nous pouvons rajouter des 0 à gauche

# Représentation des données

- Changement de base : de la base « b » à la base « a »

  Exemple avec **octal** <=> **binaire**

  - Deuxième méthode de conversion :
    - Directement du binaire vers l'hexadécimal /octal :

$$1010101010_2 \text{ (à convertir en octal)}$$

(on regroupe par 3 à partir de la droite)

$$\underset{1}{\underline{001}}\ \underset{2}{\underline{010}}\ \underset{5}{\underline{101}}\ \underset{2}{\underline{010}} \quad \text{Donc } 1010101010_2 = 1252_8$$

# Représentation des données

- Changement de base : de la base « b » à la base « a »

Exemple avec **binaire** <=> **Hexadécimal**

  - Deuxième méthode de conversion :
    - Directement du binaire vers l'hexadécimal /octal :

$1010101010_2$ (à convertir en hexadécimal)

(on regroupe par 4 à partir de la droite)

$\underset{2}{\underline{0010}}\ \underset{A}{\underline{1010}}\ \underset{A}{\underline{1010}}$ Donc $1010101010_2 = 2AA_{16}$

# Exercice

- Exprimer $(194)_2$ en hexadécimal par les deux méthodes
  - Passage par le binaire obligatoire pour la deuxième méthode
- Sans passer par la base décimale, écrire en base binaire :
  - $(FC)_{16}$
  - $(756)_8$

# Solution

- Par divisions :
  - $(194)_{10} = (\text{ C2})_{16}$
- Par regroupement
  - De décimal vers le binaire
    - $(194)_{10} = (11000010)_2$
  - De binaire vers le hexadécimal :
    - $(11000010\ )_2$ soit 1100 - 0010 c'est à dire 12-2 ce qui donne $(\text{ C2})_{16}$

# Représentation des données

- Représentation des nombres :
  - On a vu jusqu'à maintenant le codage des nombres entiers positifs dans différentes bases
  - Mais on doit aussi pouvoir manipuler des
    - Nombres réels
    - Nombres négatifs

# Représentation des données

- Codage des réels :
  - Codage d'un nombre entier positif N en base B :

    $$N = (a_n \; a_{n-1} \; a_{n-2} \; \ldots \; a_1 \; a_0)_B$$

  - Pour coder un nombre réel positif N : on rajoute une partie fractionnaire après une virgule

    $$N = (a_n \; a_{n-1} \; \ldots \; a_1 \; a_0 \; , \; \mathbf{b_1 \; b_2 \; \ldots \; b_{m-1} \; b_m})_B$$

  - La valeur en décimal d'un tel nombre est alors donnée par le calcul de

    $$a_nB^n + a_{n-1}B^{n-1} + \ldots a_1B + a_0 + b_1B^{-1} + b_2B^{-2} + \ldots b_{m-1}B^{-m+1} + b_mB^{-m}$$

# Représentation des données

- Codage des réels : exemples
  - $(123,45)_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$
  - $(101,101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

    $= 4 + 1 + 0,5 + 0,125$

    $= 5,625$
  - $(AB,4E)_{16} = 10 \times 16^1 + 11 \times 16^0 + 4 \times 16^{-1} + 14 \times 16^{-2}$

    $= 160 + 11 + 4 \times 0,0625 + 14 \times 0,00390625$

    $= 171,3046875$

# Représentation des données

- Codage des réels : conversion d'un réel décimal en base B
  - Pour la partie entière
    - Utiliser la méthode de la division entière comme pour les entiers
  - Pour la partie fractionnaire
    - Multiplier la partie fractionnaire par B
    - Noter la partie entière obtenue
    - Recommencer cette opération avec la partie fractionnaire du résultat et ainsi de suite
    - Arrêter quand
      - la partie fractionnaire est nulle
      - Ou quand la précision souhaitée est atteinte car on ne peut pas toujours obtenir une conversion en un nombre fini de chiffres pour la partie fractionnaire
    - La partie fractionnaire dans la base B est la concaténation des parties entières obtenues dans l'ordre de leur calcul

# Représentation des données

- Codage des réels : exemples de conversion
  - Conversion de 12,6875 en binaire
    - Conversion de 12 : donne $(1100)_2$
    - Conversion de 0,6875
      - 0,6875 x 2 = 1,375 = 1 + 0,375
        0,375 x 2 = 0,75 = 0 + 0,75
        0,75 x 2 = 1,5 = 1 + 0,5
        0,5 x 2 = 1 = 1 + 0
    - $(12,6875)_{10} = (1100,1011)_2$

# Représentation des données

- Codage des réels : exemples de conversion
  - Conversion de 171,3046875 en hexadécimal
    - Conversion de 171 : donne $(AB)_{16}$
    - Conversion de 0,3046875
    - 0,3046875 x 16 = 4,875 = 4 + 0,875

      0,875 x 16 = 14,0 = 14 + 0
    - $(171{,}3046875)_{10} = (AB{,}4E)_{16}$

# Représentation des données

- Codage des réels : Exercice
  - Convertir de 11,6046875 en hexadécimal

# Représentation des données

- Codage des réels : Exercice
  - Convertir de 11,6046875 en hexadécimal

Solution :

- 11 donne en hexa B
- 0, 6046875 * 16 = 9,675 = 9 + 0,675
- 0,675 * 16 = 10,8 = 10 + 0,8
- 0,8 * 16 = 12,8 = 12 + 0,8

- La représentation de11,6046875 en hex (B, 9ACC) avec une précision de $16^{-4}$

# Représentation des données

- Implémentation des nombres : les entiers
  - Les entiers vont être représentés par des emplacement mémoires formés d’octets (byte) : mot machine
    - Le mot machine est l’entité de stockage de base et est formé de 1, 2, 3, 4, 8, . . . octets selon les machines, c.a.d. 8, 16, 32, 64, . . . bits. La plupart des ordinateurs actuels utilisent 32 bits (ou plutôt 64 maintenant).
  - Avec N bits on a $2^N$ suites possibles de 0 et 1. Conséquence : les nombres représentables sont en nombre fini. On ne peut pas tout représenter ni calculer. Avec 32 bits on est limité à $2^{32}$ écritures possibles et donc de l’ordre de quelques milliards de nombres (et le plus grand nombre est de cet ordre de grandeur).
    - Donc on ne pourra jamais coder tous les entiers dans une machine.

# Représentation des données

- Implémentation des nombres : les entiers
  - Questions : sachant qu’on dispose de N bits pour représenter un entier.
    - comment représenter les entiers signés ?
    - comment calculer +, − avec la représentation choisie ?

# Représentation des données

- Implémentation des nombres : les entiers
  - Terminologie : bit de poids fort : celui de la plus grande puissance (le plus à gauche ), bit de poids faible : celui des unités (le plus à droit).
  - Il existe trois façon pour représenter les entiers en machine
    - Représentation 1 : bit de signe et valeur absolue
    - Représentation 2 : complément à 1
    - Représentation 3 : complément à 2
  - Pour toutes ces représentations
    - On aura toujours un bit utilisé pour préciser le signe du nombre : le bit de poids fort

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 1 : bit de signe et valeur absolue
    - Principe : considérer que le bit de poids fort code le signe
      - 0 = entier positif,
      - 1 = entier négatif
      - Les autres bits codent le nombre en valeur absolue
        - Nécessité de savoir sur combien de bits on code le nombre pour déterminer quel bit code quoi

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 1 : bit de signe et valeur absolue
    - Exemples si codage sur 3 bits
      - $(111)_2$ = -3 car bit de poids fort à 1
      - $(000)_2$ = +0 car bit de poids fort à 0
      - $(110)_2$ = -2 car bit de poids fort à 1
      - $(001)_2$ = +1 car bit de poids fort à 0
      - $(101)_2$ = -1 car bit de poids fort à 1
      - $(010)_2$ = +2 car bit de poids fort à 0
      - $(100)_2$ = -0 car bit de poids fort à 1
    - Exemples si codage sur 4 bits
      - $(0111)_2$ = 7 car bit de poids fort à 0
      - $(1111)_2$ = -7 car bit de poids fort à 1

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 1 : bit de signe et valeur absolue
    - Exemples si codage sur 16 bits :
      - Nombres positifs : 0XXXXXXXXXXXXXXX
      - Nombres négatifs : 1XXXXXXXXXXXXXXX
      - On a 15 bits pour coder la valeur absolue du nombre soit $2^{15} = 32768$ valeurs possibles
        - Pour le positif : de 0 à 32767
        - Pour le négatif : de -0 à -32767

    - Pour *p bits* : $-(2^{p-1} - 1) <= N <= 2^{p-1} - 1$

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 1 : bit de signe et valeur absolue
    - Inconvénients de la représentation :
      - On code 2 fois le 0 : 0000 (+0) et 1000 (-0)
      - Problème d'addition :
        - Exemple si le codage est sur 3 bits :
          - 001 + 110 = 111 ?
          - 001 + 001 = 010
          - 101 + 100 = 101
          - 011 + 010 = 101 ?

On souhaiterait avoir une représentation donnant une unique représentation à chaque valeur, et pour laquelle on puisse calculer la somme bit à bit.

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 2 : complément à 1
    - Complément logique d'un nombre binaire
      - Les 1 deviennent 0 et les 0 deviennent 1
      - Complément logique est dit « complément à 1 »
      - Codage des nombres signés avec complément logique
        - Nb positif : comme pour un entier non signé
        - Nb négatif : complément logique de son opposé positif
        - Bit de poids fort code le signe : 0 = positif, 1 = négatif
      - Exemple, codage sur un octet :
        - $(00000111)_2 = 7$
        - Complément à 1 : $(11111000)_2 = -7$ (et pas 248)

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 2 : complément à 1
    - Inconvénient :
      - toujours 2 façons de coder le 0 :
        - 00000000 et le 11111111

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 3 : complément à 2
    - Complément arithmétique
      - Complément logique du nombre auquel on rajoute la valeur de 1  : Dit « complément à 2 »
    - Codage nombres signés avec complément arithmétique
      - Nb positif : comme pour un entier non signé
      - Nb négatif : complément arithmétique de son opposé positif
      - Bit de poids fort code le signe : 0 = positif, 1 = négatif

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 3 : complément à 2
    - Exemples sur 3 bits
      - Pour +1: $(001)_2$
      - Pour -1 : $(111)_2$
        - Complément à 1 de -1 est : $(110)_2$
        - Complément à 2 de -1 : $(110)_2 + (001)_2 = (111)_2$
      - $-2 = (110)_2$ et $+2 = (010)_2$
      - $-3 = (101)_2$ et $+3 = (011)_2$
    - Exemples sur 4 bits :
      - $6 = (0110)_2$
      - Complément à 1 : 1001
      - Complément à 2 : 1001 + 1 = 1010 d'ou le codage de -6

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 3 : complément à 2
    - Pour p bits, on code $- 2^{p-1} <= N <= 2^{p-1} - 1$ valeurs
      - Sur 16 bits : $- 32768 <= N <= 32767$
      - Ce codage est le plus utilisé, c'est le standard de fait pour coder les entiers signés
      - Intérêts
        - Plus qu'une seule façon de coder le 0
          - Grace au « +1 » qui décale l'intervalle de codage des négatifs
        - Facilite les additions/soustractions en entier signé
      - Propriétés du complément à 2
        - comp2 ( N ) + N = 0
        - comp2 ( comp2 ( N ) ) = N

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 3 : complément à 2
    - Décodage de la représentation en complément à 2.

      Etant donné une écriture $u = a_{N-1} \ldots a_1\, a_0$ donnée en complément à 2, pour calculer la valeur du nombre x représenté par u, on procède ainsi :
      - Si le premier bit $a_{N-1}$ vaut 0, alors le nombre x est positif ou nul et on a simplement le décodage habituel
      - Si le premier bit $a_{N-1}$ vaut 1, alors le nombre x est strictement négatif et est obtenu de la façon suivante :
        - appliquer l'opération de complément à 1 à u. Notons v la nouvelle écriture obtenue.
        - calculer la valeur en base 2 de la nouvelle écriture v. Notons n cette valeur
        - la valeur de x est donnée par $x = -(n + 1)$.

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 3 : complément à 2
    - Addition à complément 2

      Nous avons vu que le calcul de la représentation d'un nombre x en complément à 2 est plus compliqué mais chaque nombre possède une écriture unique et de plus l'addition peut être exécutée bit à bit :

$$\begin{array}{rr} & 1 \\ + & -2 \\ \hline & -1 \end{array} \quad \begin{array}{rr} & 001 \\ + & 110 \\ \hline & 111 \end{array} \qquad \begin{array}{rr} & 1 \\ + & 1 \\ \hline & 2 \end{array} \quad \begin{array}{rr} & 001 \\ + & 001 \\ \hline & 010 \end{array} \qquad \begin{array}{rr} & -1 \\ + & -2 \\ \hline & -3 \end{array} \quad \begin{array}{rr} & 111 \\ + & 110 \\ \hline & 101 \end{array}$$

# Représentation des données

- Implémentation des nombres : les entiers
  - Représentation 3 : complément à 2
    - Débordement :

      Exemple avec une représentation sur 3 bits on a joute deux entiers.

      Que se passe-t-il si on a un résultat plus grand que 3 ou plus petit que −4 ? Par exemple 3 + 2 = 5 cad 011 + 010 = 101.

      Le résultat est aberrant (il est négatif ), il y a eu débordement. Il faut donc s’assurer que les opérations arithmétiques qu’on effectue restent dans les limites des représentations.

# Représentation des données

- Implémentation des nombres : les entiers
  - Résumé
    - Compléments 2 et 1
      - Utilisables dans n'importe quelle base, pas que en binaire
      - Avec les mêmes propriétés dans toute base
    - Complément à 1 d'un nombre N en base B
      - Nombre pour lequel chaque chiffre $a_x$ de N est remplacé par le chiffre de valeur $B - 1 - a_x$
        - Exemple en base 8 : $comp_1(235) = 542$
    - Complément à 2 = complément à 1 + 1
      - Exemple en base 8 : $comp_2\ (235) = 542 + 1 = 543$
      - Rajoute la valeur 1 quelle que soit la base considérée

# Représentation des données

- Implémentation des nombres : les entiers
  - Exercice 1

    Dans tout l'exercice, on considère le codage des entiers sur 4 bits (4 chiffres binaires).

    Dans un premier temps, on ne code que des entiers positifs ou nuls.

    - Quelle est le codage binaire (sur 4 bits) de l'entier $(12)_{10}$ ?
    - Quel nombre en base 10 correspond au nombre en base 2 $(1010)_2$ ?
    - Quel est le plus grand nombre représentable par ce codage (donnez sa valeur en base 2 et en base 10) ?

# Représentation des données

- Implémentation des nombres : les entiers
  - Exercice 2

    Dans tout l'exercice, on considère le codage des entiers sur 4 bits (4 chiffres binaires).

    Un nombre négatif n est codé par le complément à un de son opposé −n. Rappel : le complément à un d'un nombre binaire consiste à inverser tous les chiffres de ce nombre.

    - Quel est le codage de l'entier $(-3)_{10}$ en utilisant le complément à un ?
    - A quel nombre en base 10 correspond $(1100)_2$ .
    - A quel nombre correspond $(1111)_2$ ?
    - Quel est l'inconvénient de la représentation des entiers négatifs par complément à 1 ?.

# Représentation des données

- Implémentation des nombres : les entiers
  - Exercice 3

    Dans tout l’exercice, on considère le codage des entiers sur 4 bits (4 chiffres binaires).

    Le complément à deux d’un nombre binaire consiste à ajouter 1 à son complément à un. Le décalage à gauche d’un nombre binaire est une opération consistant à décaler tous les chiffres (bits) de ce nombre d’une position vers la gauche.

    - Donner la représentation en complément à 2 de 14 puis -15. Attention codage sur 6 bits
    - $14 = (001110)_2$ et pour $-15 = (110001)_2$
    - Donner le décimal correspondant au complément à 2 de $(110011)_2$

# Représentation des données

- Implémentation des nombres : les entiers
  - Exercice 3

    Le complément à deux d'un nombre binaire consiste à ajouter 1 à son complément à un. Le décalage à gauche d'un nombre binaire est une opération consistant à décaler tous les chiffres (bits) de ce nombre d'une position vers la gauche.

    - Donner la représentation en complément à 2 de 13 puis -14. Attention codage sur 6 bits
    - Donner le décimal correspondant au complément à 2 de $(110011)_2$

# Représentation des données

- Implémentation des nombres : les entiers
  - Exercice 3

    Dans tout l'exercice, on considère le codage des entiers sur 4 bits (4 chiffres binaires).

    Le complément à deux d'un nombre binaire consiste à ajouter 1 à son complément à un. Le décalage à gauche d'un nombre binaire est une opération consistant à décaler tous les chiffres (bits) de ce nombre d'une position vers la gauche.

    - Que réalise l'opération de décalage des nombres binaires sur les entiers en base 10 correspondants ?
    - Sur une représentation par complément à 2, quel est est le plus grand nombre binaire représentant un entier positif auquel on peut appliquer cette opération sans risque de débordement ? Que vaut cet entier en base 10.

# Représentation des données

- Implémentation des nombres : les réels
  - Même problème que pour les entiers mais en plus compliqué :
    - Place finie en mémoire pour une infinité de réels, mais en plus on ne sait pas représenter complètement un réel.
      - 165686979878979678568008998 grand mais complètement déterminé.
      - 1/3 = 0.3333333... pas de représentation décimale finie,
      - $\sqrt{2}$ = 1.414... pas de représentation rationnelle,
      - $\pi$ = 3.14159.... pas de représentation algébrique
    - Seuls les nombres décimaux pas trop grands peuvent se représenter en machine. Par conséquent toute représentation de nombre réel sera imparfaite. De plus comme pour les entiers, les nombres sont représentés par des suites de bits donc en base 2. Cela a des conséquences inattendues : le nombre 0.1 est un décimal en base 10 mais pas en base 2 !

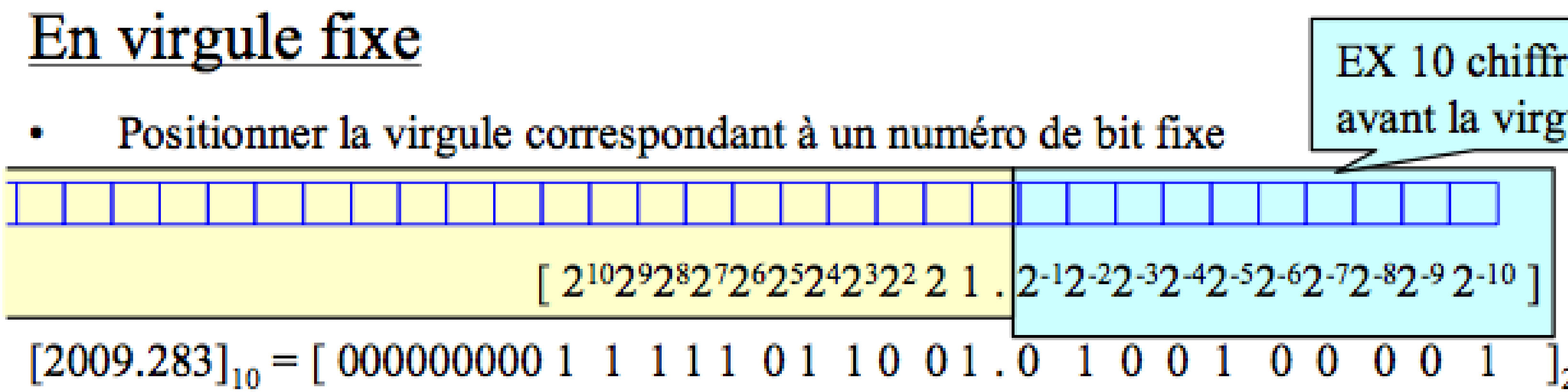
# Représentation des données

- Implémentation des nombres : les réels
  - Représentation 1 : virgule fixe (le nombre de chiffres des parties entières et fractionnaires est fixé)
    - Considérons le réel : 2009,283. Sa représentation binaire est

      (101 1 1110110 01, 0 1 001 0 0 0 0 1) car :
      - Le codage binaire de 2009 est :101 1 1110110 01
      - Le codage binaire de 0,283 est : 0 1 001 0 0 0 0 1 :
        - *0.283 ×2=0.566 0*
        - *0.566 ×2=1.132 1*
        - *0.132 ×2=0.264 0*
        - *0.264 ×2=0.528 0*
        - *0.528 ×2=1.056 1*
        - *0.056 ×2=0.112 0*
        - *0.112 ×2=0.224 0*
        - *0.224 ×2=0.448 0*
        - *0.448 ×2=0.896 0*
        - *0.896 ×2=1.792 1*

# Représentation des données

- Implémentation des nombres : les réels
  - Représentation 1 : virgule fixe (le nombre de chiffres des parties entières et fractionnaires est fixé)




Problème : la partie fractionnelle est fixe et on ne peut pas la modifier. Impossible de combiner des nombres très différents : Abandon au profit de la représentation suivante.

# Représentation des données

- Implémentation des nombres : les réels
  - Représentation 2 : virgule flottante (représentation utilisée par les machines)

$$x = \underbrace{m}_{mantisse} * \underbrace{b}_{base}{}^{\overbrace{e}^{exposant}}$$

    - Avoir une virgule flottante et une précision limitée
    - Ne coder que des chiffres significatifs
    - Le nombre est présenté sous forme normalisée pour déterminer la mantisse et exp. On appelle notation normalisée d'un réel celle où le premier chiffre significatif est placé immédiatement après la virgule (ou le point décimal). Exemple : $1989 = 0.1989 *10^4$ (ou $0.1989\ 10^4$ ).

    Pour stocker ce nombre en mémoire, il suffit de stocker : l'exposant et la mantisse

# Représentation des données

- Implémentation des nombres : les réels
  - Représentation 2 : virgule flottante (représentation utilisée par les machines)

$$x = \underbrace{m}_{mantisse} * \underbrace{b}_{base}{}^{\overbrace{e}^{exposant}}$$

Des représentations approchées de π sont : (0.031, 2), (3.142, 0), (0.003, 3) et on voit qu'elles ne donnent pas la même précision. Pour éviter ce problème on utilisera une mantisse normalisée c'est à dire qu'elle ne contiendra pas de 0 en tête (donc le premier bit de la mantisse sera 1).

# Représentation des données

- Implémentation des nombres : les réels
  - Représentation 2 : virgule flottante (représentation utilisée par les machines)
  - Standard IEEE 754 : codage binaire de réels en virgule flottante
    - Précision simple : 32 bits 1 bit de signe, 8 bits exposant, 23 bits mantisse

$$\underbrace{1}_{signe}\ \underbrace{0\,0\,0\,1\,1\,1\,1\,0}_{exposant}\underbrace{0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0}_{mantisse}$$

    - Précision double : 64 bits 1 bit de signe, 11 bits exposant, 52 bits mantisse
    - Précision étendue : sur 80 bits

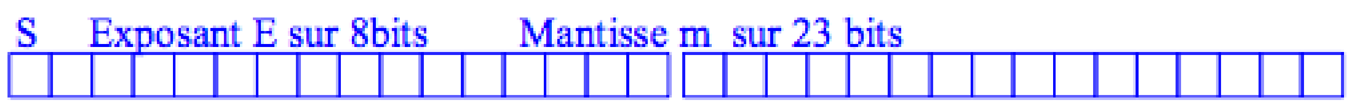
# Représentation des données

- Implémentation des nombres : les réels
  - Représentation 2 : virgule flottante (représentation utilisée par les machines)

  - Standard IEEE 754 : codage binaire de réels en virgule flottante
    - Précision simple : 32 bits 1 bit de signe, 8 bits exposant, 23 bits mantisse

***Format Flottant simple précision sur 32 bits : norme IEEE 754.***

Notation




| | E = 0 | 0 < E < 255 | E = 255 |
|---|---|---|---|
| M = 0 | X = 0 | $X = (-1)^S \times 2^{E-127} \times (1,m)$ | $X = (-1)^S \times \infty$ |
| M ≠ 0 | Forme dénormalisée $X = (-1)^S \times 2^{-126} \times 0,m$ | | X = Not A Number |

# Représentation des données

- Implémentation des nombres : les réels
  - Représentation 2 : virgule flottante (représentation utilisée par les machines)
  - Standard IEEE 754 : codage binaire de réels en virgule flottante

***Format Flottant simple précision sur 32 bits : norme IEEE 754.***

**Notation** S Exposant E sur 8bits Mantisse m sur 23 bits

**Exemple**

$13.625 = 0b\ 1101{,}101 = (-1)^0 \times 2^3 \times (1{,}101101)$

$= (-1)^S \times 2^{E-127} \times (1{,}m)$ $S = 0$; $E = 130$; $m = 0{,}101101$

| 0 | 10000010 | 10110100000000000000000 |
|---|---|---|

# Représentation des données

- Implémentation des nombres : les réels
  - Exercice 1

La représentation d'un nombre flottant sur 32 bits est telle que : le bit de signe est 0, l'exposant est 10001001, la mantisse est 00010110000000000000000.

- Expliquer ce que cela signifie.
  - 0 = positif
  - 10001001 = $(137)_{10}$ = exposant
  - 00010110000000000000000 = la mantisse
- Donner la valeur du nombre flottant en base 10.
  - 1 10101101 10010110000000000000000
  - S = 1
  - E =10101101 = 173
  - M = - 1001011

# Représentation des données

- Implémentation des nombres : les réels
  - Exercice 1

    Les flottants sont représentés de manière normalisée sur 32 bits.

    - Calculer la représentation sur 32 bits du nombre réel 0, 1.
    - Même question pour 0, 2 puis 3, 125.