Informatique pour tous

Architecture des ordinateurs - II

Yannick Le Bras - MPSI

Septembre 2013

Section 1

Mémoire

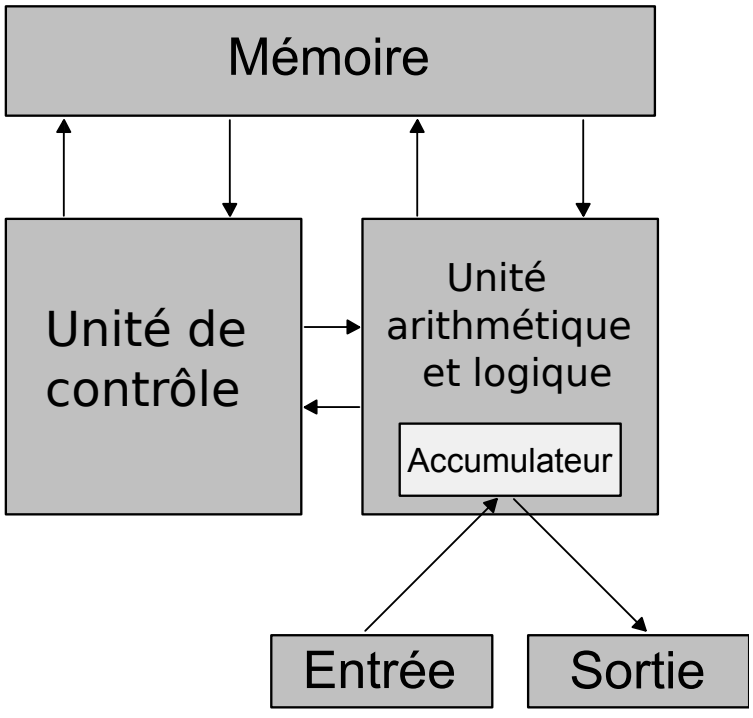


FIGURE : Architecture de von Neumann (source :Wikipedia, GFDL)

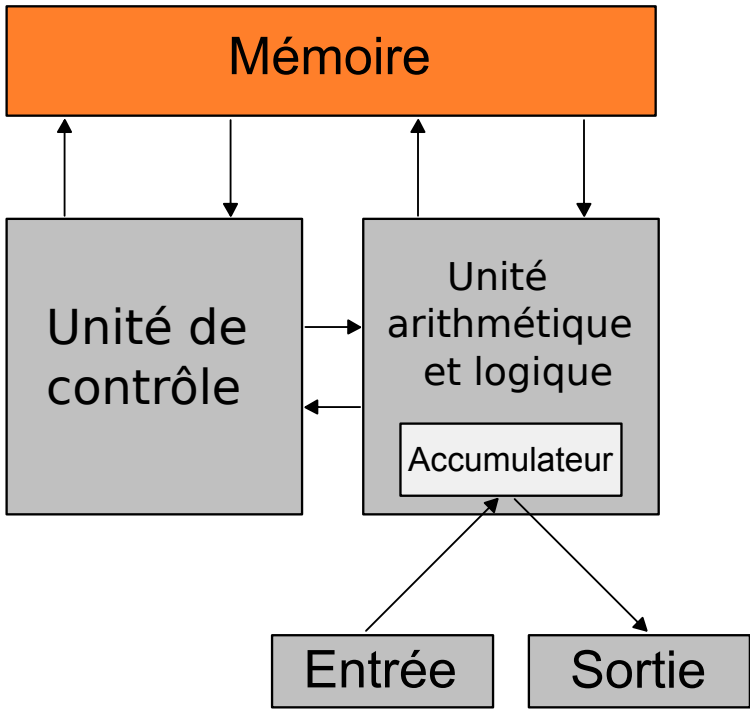
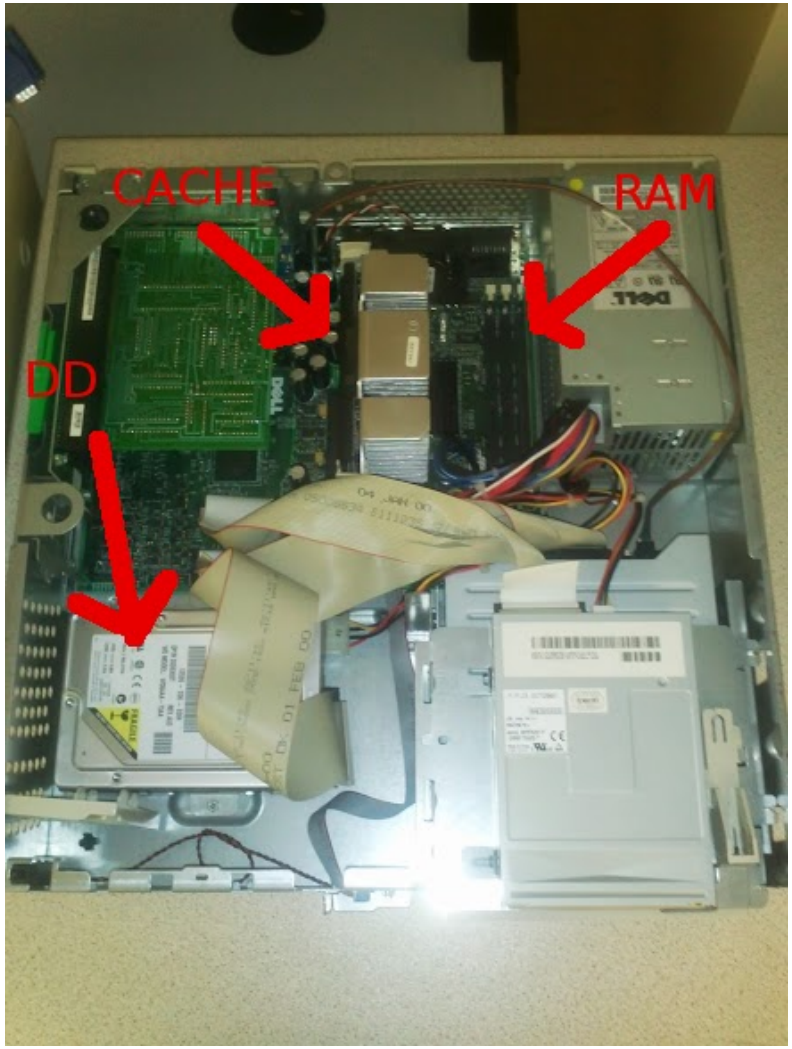


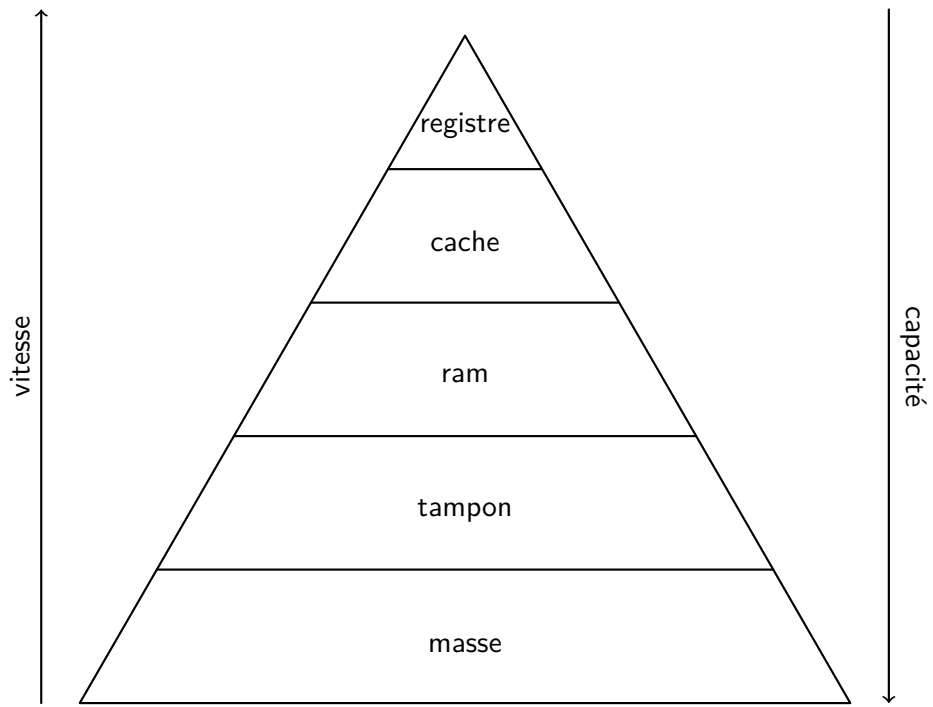
FIGURE : Architecture de von Neumann (source :Wikipedia, GFDL)


DELL


CACHE
RAM
DD

Les mémoires diffèrent par :

- leur capacité ;
- le temps d'accès.



Mémoire Cache

La mémoire cache est une mémoire :

- physiquement accolée au processeur ;
- très rapide d'accès ;
- divisée en trois niveau :
  - L1 : 32 ou 64ko, temps d'accès de 4ns ;
  - L2 : 256ko, 512ko ou 1mo, temps d'accès >5ns ;
  - L3 : 1 à 4Mo, niveau peu courant, temps d'accès 30ns ;

Le plus souvent, la technologie utilisée est la mémoire RAM statique.

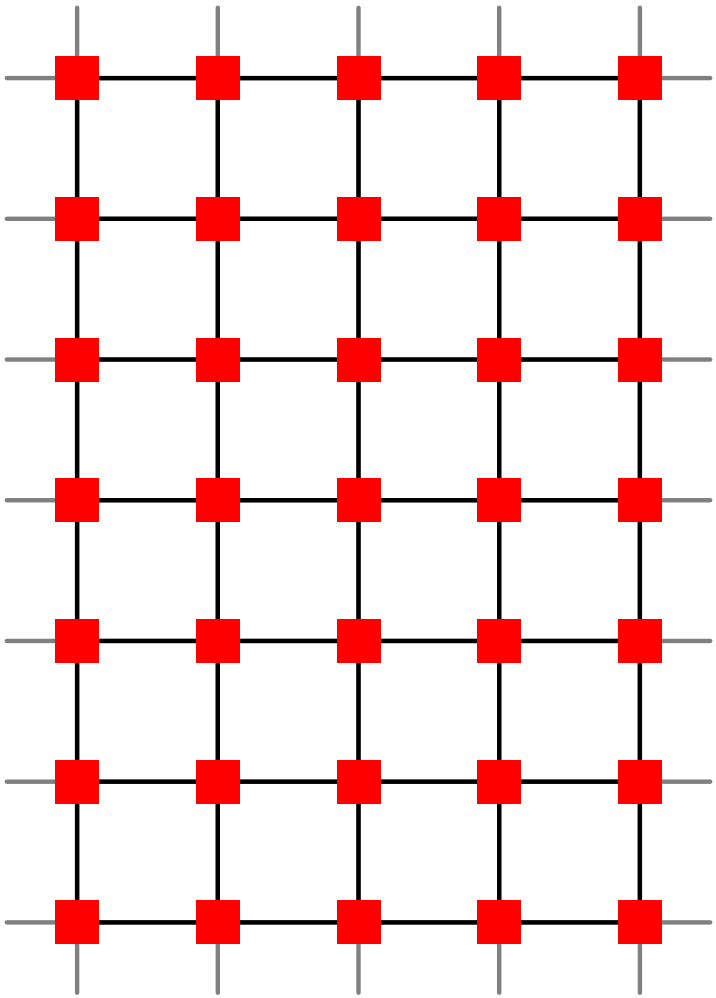
Mémoire RAM

La mémoire RAM est une mémoire :

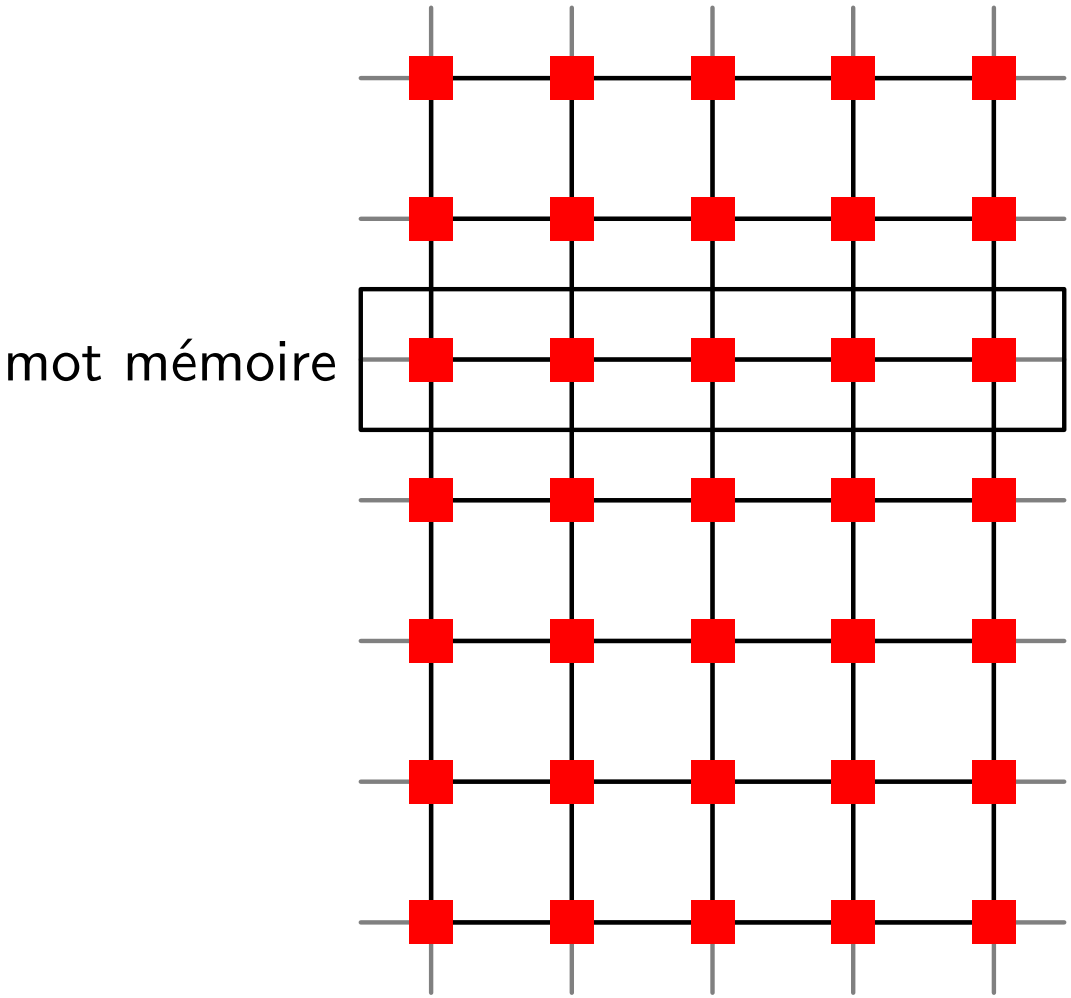
- physiquement séparée du processeur, sur la carte mère ;
- rapide d'accès (env. 60ns) ;
- de capacité importante.

Elle assure un taux de transfert autour de $3 Go/sec$.
Le plus souvent, la technologie utilisée est la mémoire RAM *dynamique*.

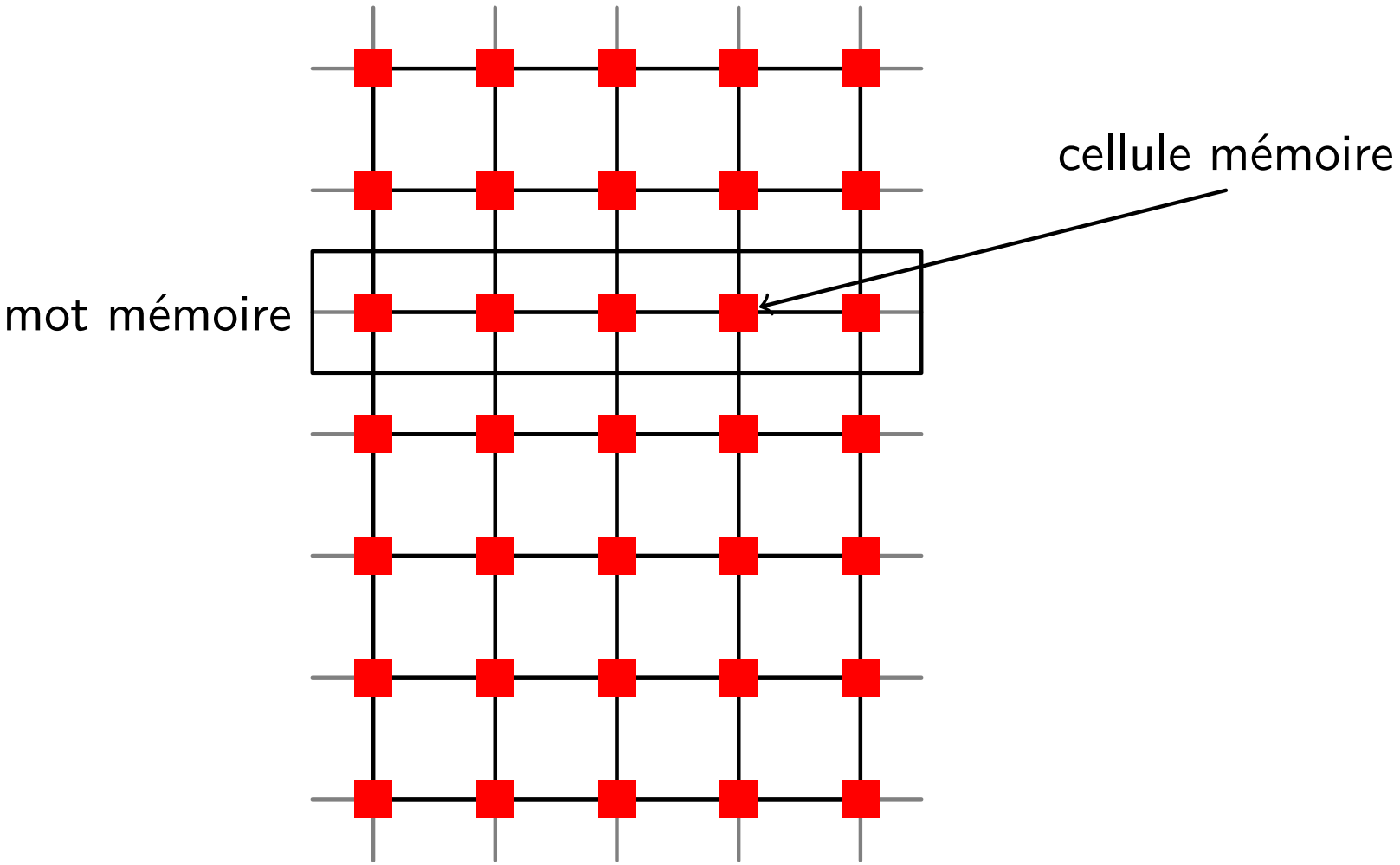
structure de la RAM



structure de la RAM



structure de la RAM



Cellule mémoire

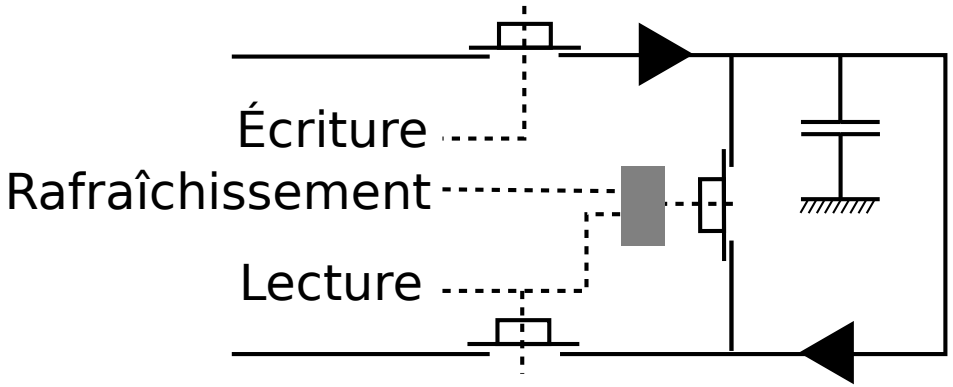
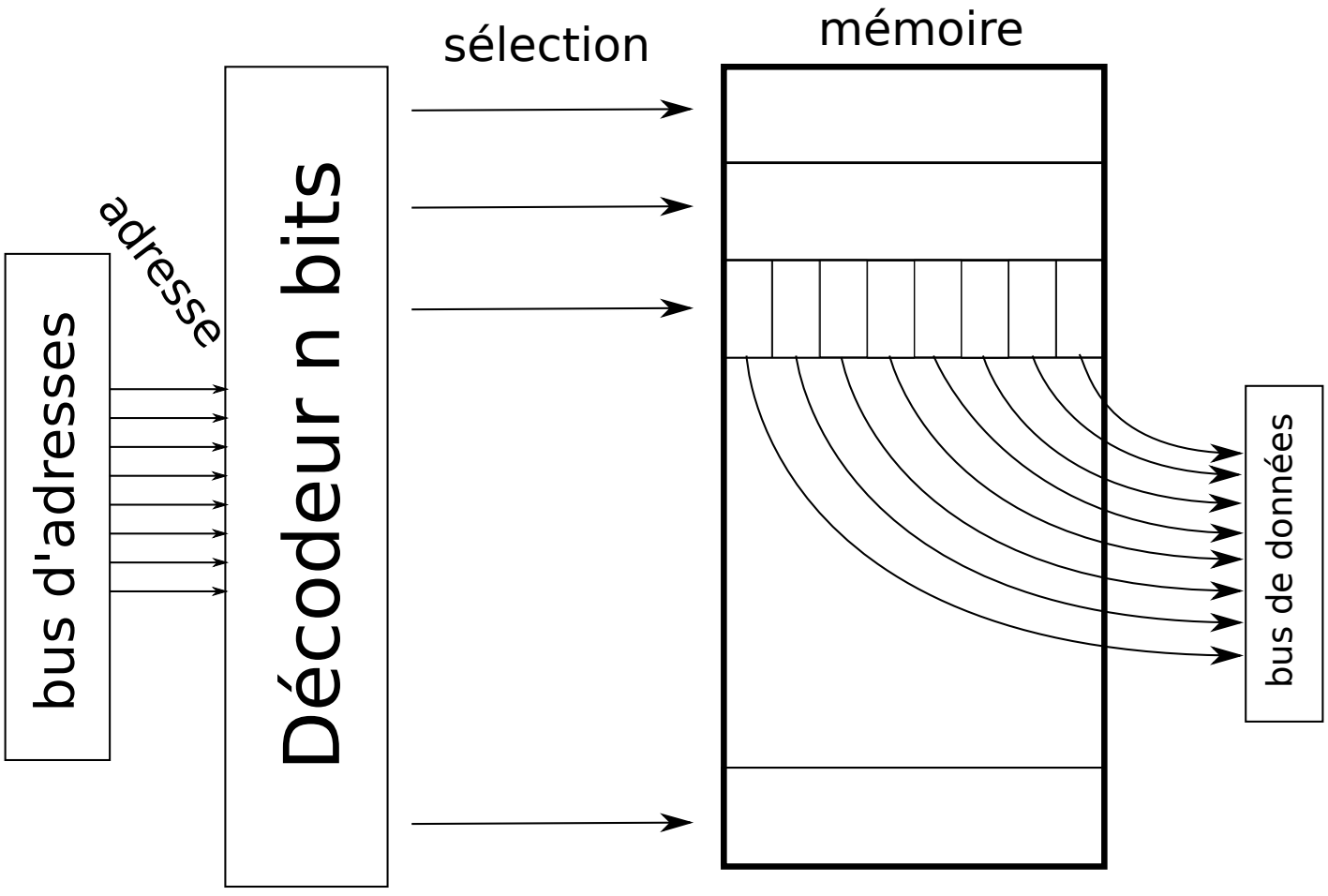


FIGURE : Cellule mémoire dynamique (d'après Blanchet)

- la charge du condensateur indique la présence d'un 1 ;
- il est nécessaire de rafraîchir le contenu ;
- une lecture est destructive : elle doit être suivie d'une écriture.

Mémoire RAM


sélection
mémoire
adresse
bus d'adresses
Décodeur n bits
bus de données

Mémoire de masse

Principalement représentée par le disque dur.

- système électromécanique ;
- temps d'accès très long : de l'ordre de 12 à 16 ms.
- très grande capacité.

Suivant les technologies, le débit est diffférent :

- IDE : les nappes assurait $133 Mo/sec$ au maximum ;
- Serial ATA I : $150 Mo/sec$;
- Serial ATA II : $300 Mo/sec$;
- Serial ATA III : $600 Mo/sec$.

Mémoires Flash

- même principe que la mémoire RAM mais non volatile ;
- 100 fois plus rapide que les disques électromécaniques ;
- 15 fois plus chers (coût au Go début 2012 : 0.7 euros) ;
- capacité plus faible ;
- pas de problèmes mécaniques, de bruit...

Section 2

Processeur

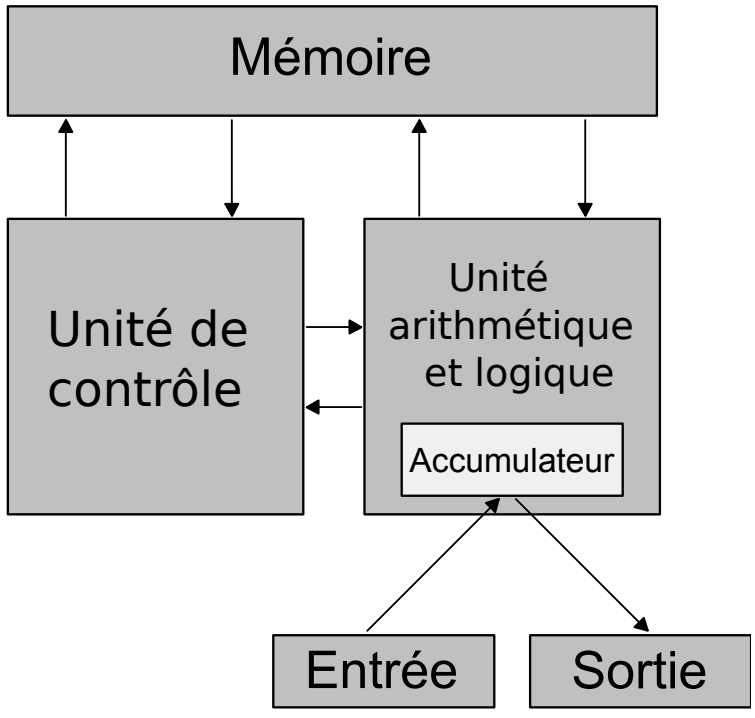


FIGURE : Architecture de von Neumann (source :Wikipedia, GFDL)

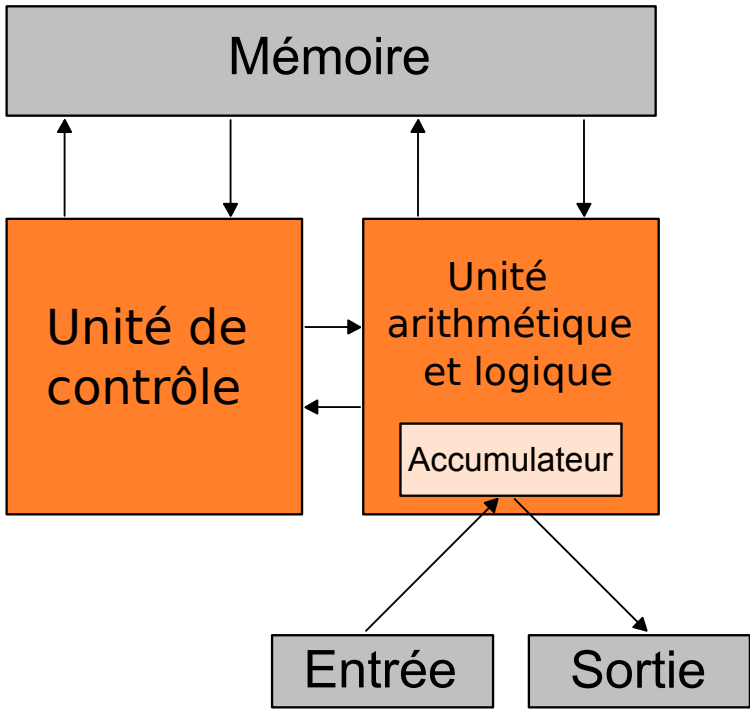
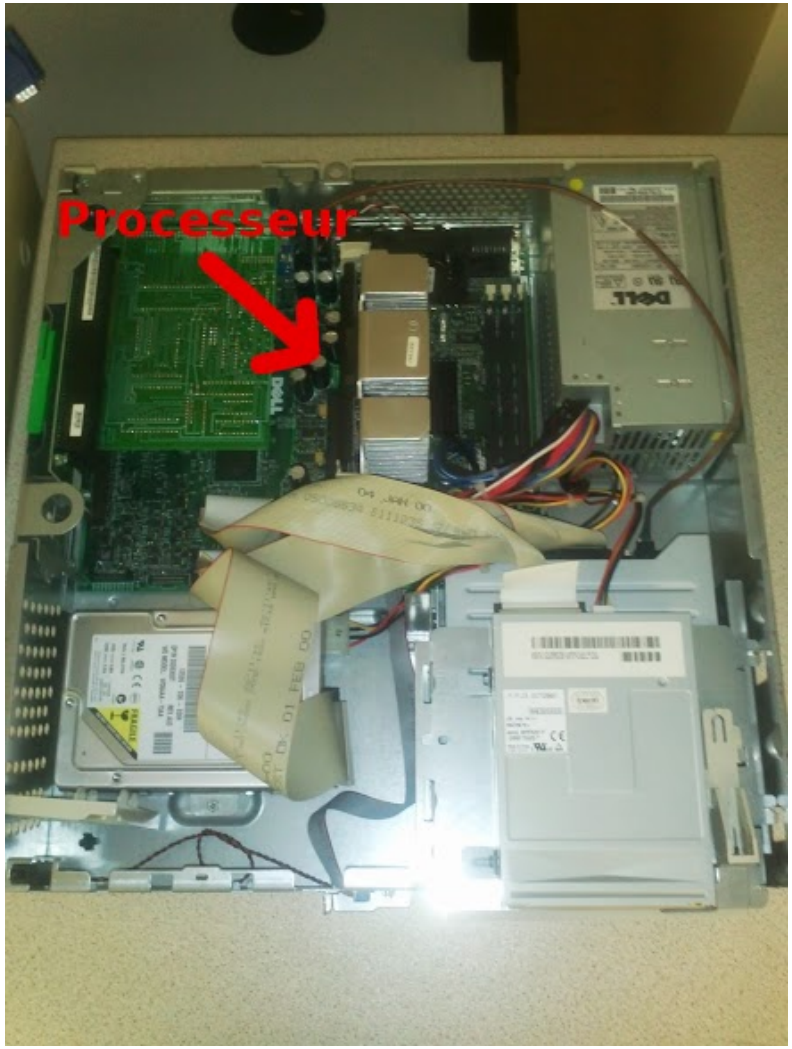


FIGURE : Architecture de von Neumann (source :Wikipedia, GFDL)


DELL


Processeur

Deux unités

- unité de contrôle :
  - va chercher les informations en mémoire : instructions, données ;
  - interprète les instructions ;
  - donne des ordres à l'unité artihmétique et logique ;
  - assure le bon séquencement.
- l'unité arithmétique et logique :
  - exécute les opérations désignées par par l'unité de contrôle.

Les échanges se font via des BUS internes.

32 ou 64 bits ?

32/64 bits est la taille des mots que le processeur est capable de formuler, notamment pour les adresses mémoire.

- avec un processeur 32 bits, on peut formuler 2^{32} adresses : la RAM ne peut dépasser 4Go ;
- avec un processeur 64 bits, on peut formuler 2^{64} adresses.
- en réalité, une partie est réservée à d'autres adressages (carte graphique...)

$\rightarrow$ on ne peut pas utiliser $4Go$ de RAM sur un système 32 bits.

Opérations logiques

Les opérations logiques sont assurées par des circuits logiques.

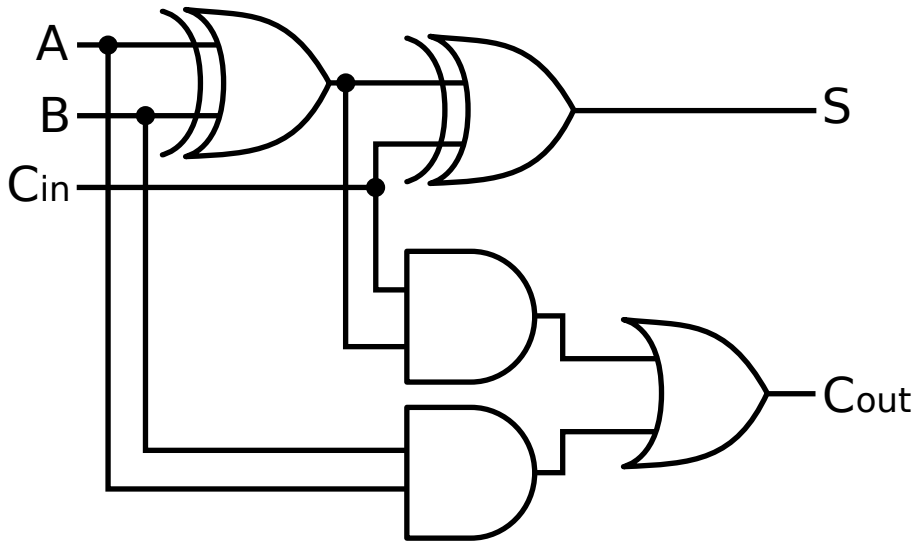


FIGURE : Additionneur complet 1bit

Jeux d'instructions

Chaque processeur reconnaît un certain nombre d'instructions :

- le jeu d'instructions est propre au constructeur : base commune ;
- chaque instruction possède un code ;
- les instructions peuvent être : arithmétiques, de mouvement, de décalage, de branchement...

Exemple : `http://aconit.inria.fr/omeka/exhibits/show/histoire-machines/naissance-ordinateur/modele-von-neumann`

Section 3

Entrées/Sorties

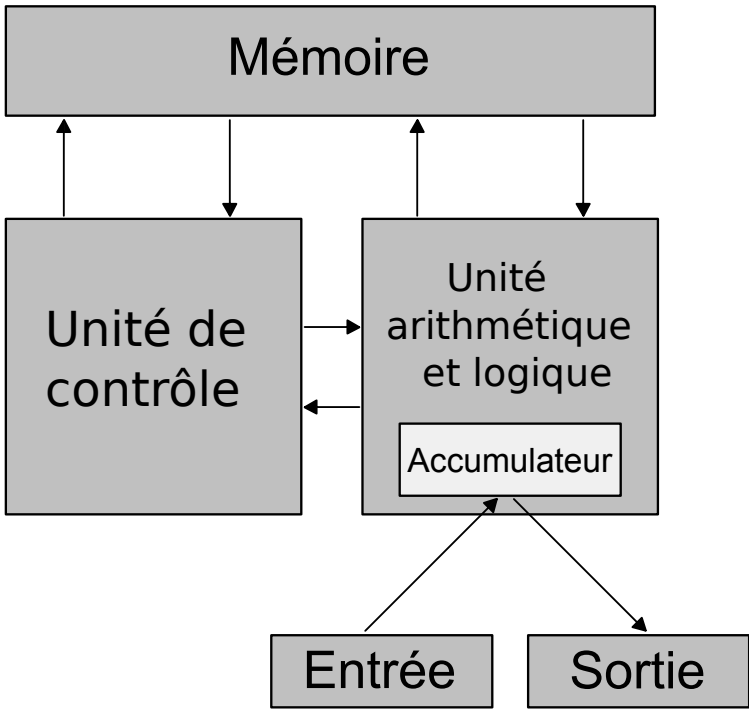


FIGURE : Architecture de von Neumann (source :Wikipedia, GFDL)

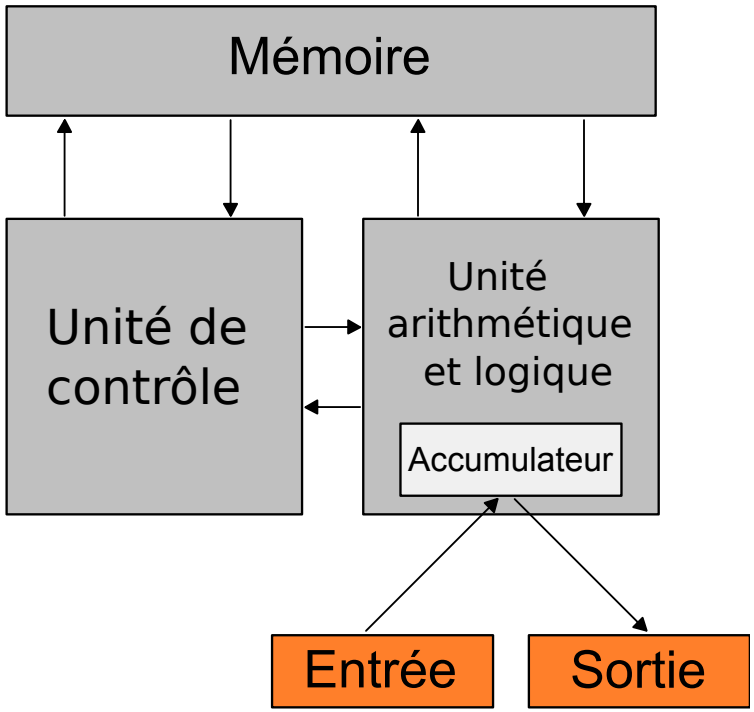


FIGURE : Architecture de von Neumann (source :Wikipedia, GFDL)

Entrées/sorties de base

Les entrées sorties se font via des *périphériques* :

- clavier ;
- souris ;
- écran ;
- haut-parleurs...

La communication se fait sous forme normalisée en général pour ces périphériques.

périphériques avancés

Certains périphériques sont plus évolués :

- clavier/souris sans fil ;
- imprimante ;
- lecteurs/graveur de CD/DVD-Rom ;
- périphériques biométriques.

La communication n'est pas toujours normalisée.

Rôle du système d'exploitation

Il existe plusieurs systèmes d'exploitation grand public :

- Windows, MacOS, Linux

Rôle principal

Permettre la communication entre l'humain et la machine ; permettre la communication entre la machine et les périphériques ; gérer les ressources intelligemment.

La communication avec les périphériques se fait via des pilotes. Notion de *plug&play*.

Gestion du disque dur

Un des rôles principaux de l'OS est la gestion des données.

- mémoire virtuelle en support de la RAM ;
- écriture et lecture du disque dur ;
- organisation des données sur le disque.

$\rightarrow$ fragmentation des disques.

Section 4

Langages de programmation

Notions de langage

Langage machine

L'ensemble des instructions et leur syntaxe forme le *langage machine*.

On peut développer un programme en langage machine :

- langage bas niveau (directement interprétable par l'ordinateur) ;
- temps de développement est très élevé ;
- efficacité optimale.

Exemples de langages

Langage

Un langage de programmation est défini par une syntaxe et une sémantique :

- la syntaxe est un ensemble de mot réservés associés à une grammaire ;
- la sémantique représente le sens des phrases formulées dans la syntaxe.

On parle de *langage de programmation* si le langage est *Turing-Complet*, s'il permet de faire tout ce que fait une machine de Turing.

Exemples de langages

- Langages de programmation :
  - Pascal, Cobol, Fortran ;
  - Java, C, Python, visual Basic ;
  - php, sql ...
- Langages de présentation/balisage :
  - HTML, CSS, XML ;
  - LaTeX ;
  - SVG.
- Langages exotiques :
  - Brainfuck, Ook . . .`http://www.info.univ-angers.fr/~gh/hilapr/detour.htm`

Types de langages

On distingue plusieurs types de langages :

- langages compilés/interprétés ;
- langages impératifs/fonctionnels ;
- langages orientés objet.

Chaque type est adapté à une situation particulière.

Langage Python

Cette année nous utiliserons le langage Python :

- première version publique (0.9.0) en 1991 ;
- langage gratuit et libre ;
- dernière génération depuis 2008 (3.0) ;
- version actuelle 3.3.

Son créateur Guido van Rossum est désigné *dictateur bienveillant à vie*.

Section 5

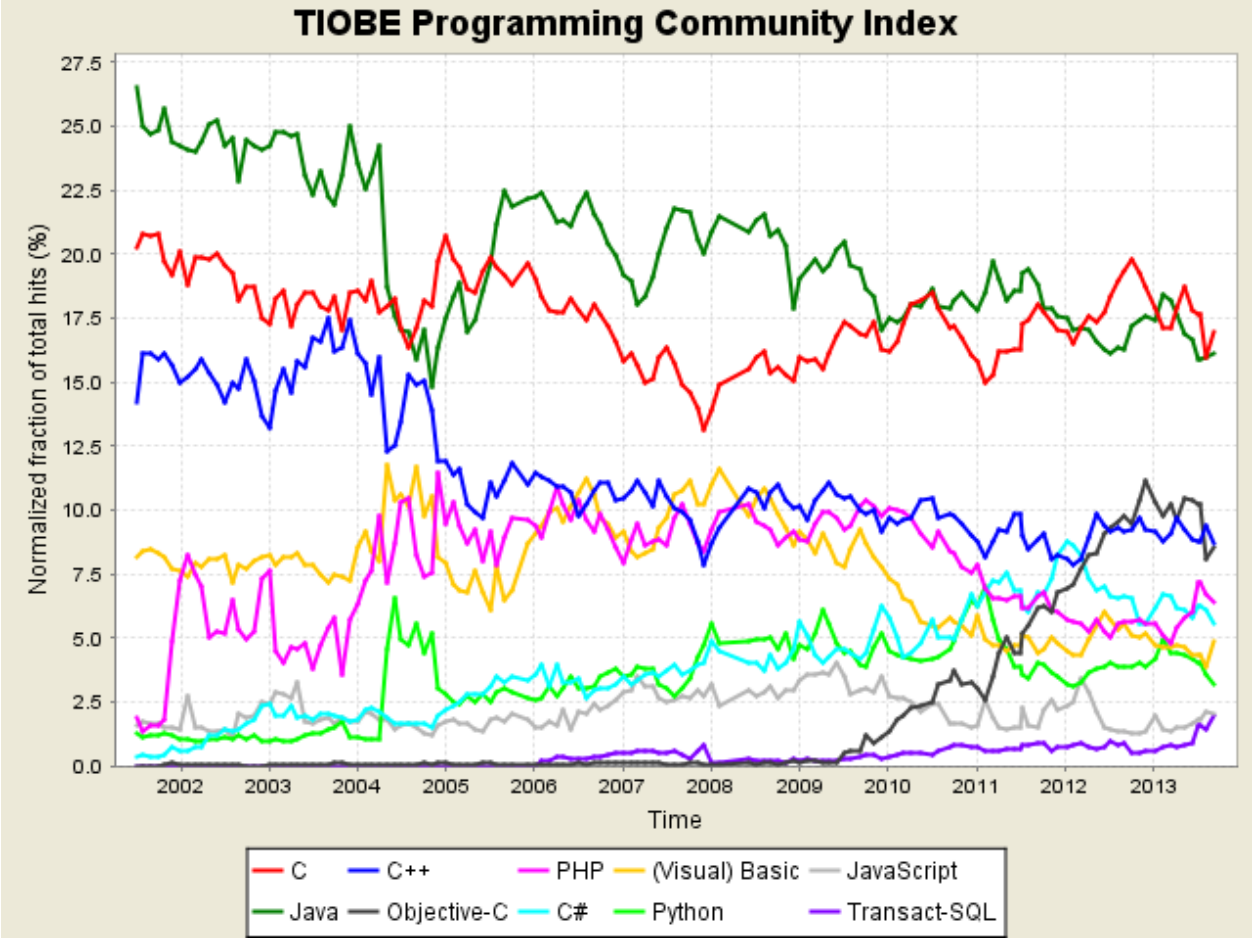
Le cas Python

Avantages de Python

Python possède de grandes qualités :

- il est portable : utilisable sous Windows, MacOS, Linux, Android...
- il est libre et gratuit ;
- il a engendré une grande communauté ;
- il est simple d'utilisation et lisible : c'est un langage élégant.

Utilisation de Python



Éléments de syntaxe

La syntaxe de Python est à la fois intuitive et simple.

- les blocs d'instructions sont délimités par l'*indentation* ;
- les instructions sont délimitées par les retours à la ligne ;
- il n'y a pas de parenthèses parasites.

Python est un langage interprété, facilitant la création de script et le prototypage.

opérateurs et affectations

On retrouve les principaux symboles d'opération et d'affectation.

affectation simple	`X=a`
affectation multiple	`x=y=z=5`
affectation parallèle	`x,y=3,5`
opérateurs arithmétiques	`+, -, *, /, //, **, %`
opérateurs logiques	`or, and, not`
opérateurs de comparaison	`<, <=, >, >=, ==, !=`

`//` est la division entière, `**` l'élévation à la puissance et `%` le reste

`!=` est le symbole *différent*

Les boucles

Les boucles permettent d'automatiser une tâche répétitive.
Boucle While (Tant que) :

```
while condition :
    instruction 1
    instruction 2
    ...
    instruction n
```

ATTENTION : l'indentation est primordiale!

Exemple :

```
i=0
while i<10 :
    print("tour de boucle numero",i)
    i=i+1
```

Les boucles

Les boucles permettent d'automatiser une tâche répétitive.
Boucle For (Pour) :

```
for variable in liste :
    instruction 1
    instruction 2
    ...
    instruction n
```

ATTENTION : l'indentation est primordiale!

Exemple 1 :

```
for i in range(5) :
    print("tour de boucle numero",i)
```

Les boucles

Les boucles permettent d'automatiser une tâche répétitive.
Boucle For (Pour) :
Exemple 2 :

```
for i in [1,2,4,1,5,6] :
    print("la␣variable␣i␣vaut␣:",i)
```

Exemple 3 :

```
for i in "Bonjour" :
    print("la␣lettre␣courante␣est␣:",i)
```

Instructions conditionnelles

Les instructions conditionnelles permettent à l'ordinateur de faire des choix.

```
if condition1 :
    block d'instructions 1
elif condition2 :
    block d'instructions 2
    ...
else
    block d'instructions n+1
```

ATTENTION : pensez au `else` final pour un filtrage exhaustif

Si la condition i est vérifiée et qu'aucune des conditions 1 à $i-1$ ne sont vérifiées, alors le block i est exécuté. Si aucune condition n'est vérifiée, on exécute le block $n+1$.

Instructions conditionnelles

Les instructions conditionnelles permettent à l'ordinateur de faire des choix.
Exemple :

```
temp=int(input("Temperature de la piece ? "))
if temp<15 :
    print("Allumez le chauffage")
elif temp>35 :
    print("Pensez a boire")
else :
    print("Tout va bien")
```

Quelques fonctions de base

Quelques fonctions doivent être connues :

- input(s) : affiche la chaine s et attend une saisie clavier ;
- print(s) : permet l'affichage de la chaine s à l'écran ;
- range(n) : génère un itérable contenant les entiers de 0 à $n-1$;
- len(a) : renvoie la longueur de l'objet définit par a.

Exemple :

```
nom=input("Entrez votre nom : ")
n=len(nom)
print("Vous vous appelez",nom)
print("Votre nom contient",n,"lettres")
```

Apprendre à programmer

Programmer s'apprend en ESSAYANT. Alors essayez de créer un programme qui :

- demande deux entiers et indique lequel est le plus grand ;
- demande un numéro de sécurité sociale et renvoie le sexe, le mois et l'année de naissance associés ;
- demande un entier n et affiche les entiers de 1 à n ;
- demande une phrase et renvoie le nombre de e dans la phrase...

merci.