Modèle d'architecture d'un ordinateur

Architecture de von Neumann

Les grands principes de fonctionnement des ordinateurs actuels résultent de travaux menés au milieu des années 1940. Ces travaux ont défini un schéma d'architecture appelée architecture de von Neumann, en référence à John Von Neumann (1903-1957), un mathématicien et physicien (et bien d'autres choses) américano-hongrois qui a participé et publié les travaux en 1945.

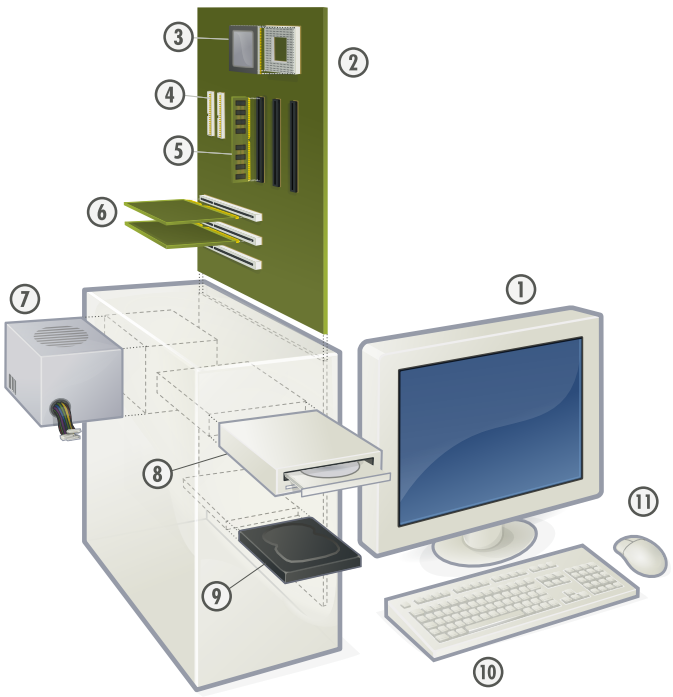
John von Neumann. Source : Wikimedia Commons

Avant de détailler cette architecture, commençons par une petite activité.

Activité 1 : Composants d'un ordinateur

Lorsqu'on ouvre un ordinateur, quelle que soit sa marque, nous retrouvons en général les mêmes éléments : le processeur ou microprocesseur, la mémoire vive (RAM), la carte mère, la carte graphique, interface de connexion des périphériques, le lecteur de disque, le disque dur, le clavier, l'écran, la souris, l'alimentation électrique.

Question 1 : Associez le bon élément à chaque numéro du schéma ci-dessous.



Crédit : Gustavb, CC BY-SA 3.0, via Wikimedia Commons

1 :
2 :
3 :
4 :
5 :
6 :
7 :
8 :
9 :
10 :
11 :

Question 2 : Associez chaque élément à sa description

Cerveau de l'ordinateur, qui permet à l'ordinateur d'effectuer les opérations (calculs) demandés.	
Stocke les informations des programmes et données en cours de fonctionnement	
Relie tous les éléments constituant un ordinateur. Sa principale fonction est la mise en relation de ces composants par des bus sous forme de circuits imprimés.	
Carte d'extension d'ordinateur qui permet de produire une image affichable sur un écran.	
Périphérique d'entrée qui permet d'écrire	
Assure la mise sous tension de l'ensemble des composants	
Périphérique d'entrée-sortie qui stocke les données de base de la machine.	
Périphérique d'entrée-sortie, assure le stockage et la lecture de données sur support externe non volatile	
Périphérique d'entrée qui permet de déplacer le curseur de pointage à l'écran	
Permet de connecter les périphériques (disque dur, lecteur DVD, etc.) à la carte mère.	
Périphérique de sortie qui permet de visualiser les informations venant de l'ordinateur	

Principe de l'architecture

L'idée majeure de l'architecture de von Neumann, était d'utiliser une zone de stockage *unique*, à savoir la mémoire de l'ordinateur, pour conserver à la fois les *programmes* (instructions) et les *données* qu'ils devaient manipuler.

Les travaux de von Neumann publiés en 1945 concernaient la conception de l' EDVAC, un ordinateur basé sur cette architecture.

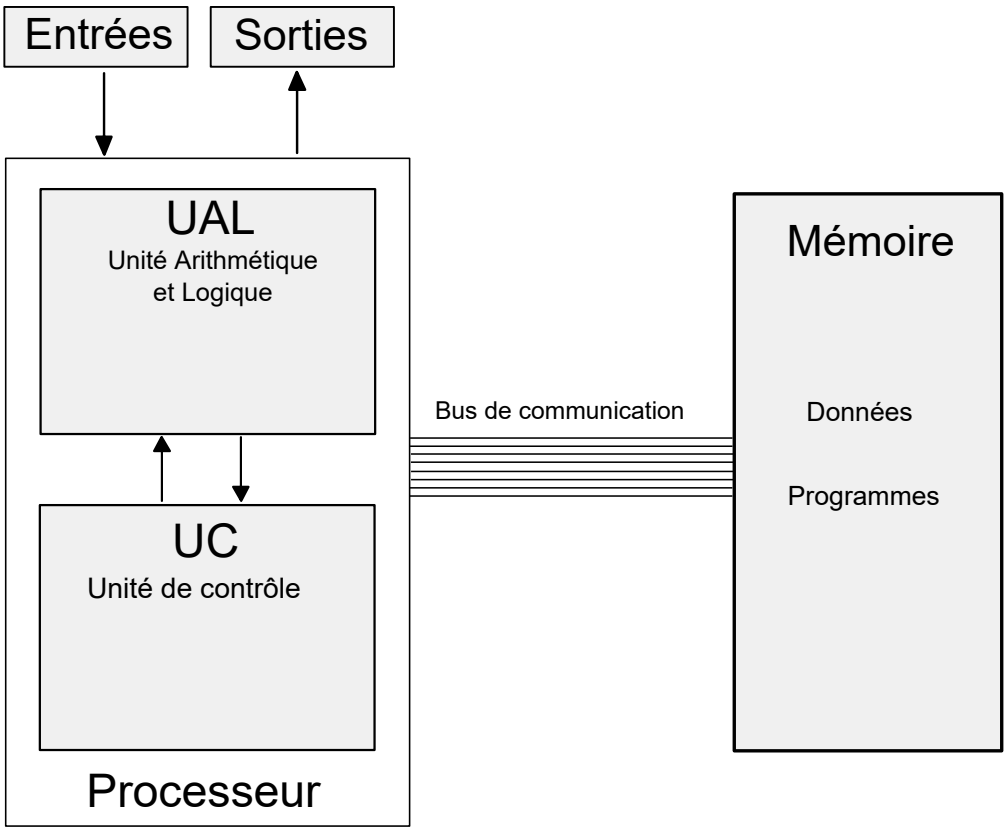
Cet ordinateur était capable d'additionner, soustraire, multiplier et diviser en binaire. Sa capacité mémoire est l'équivalent actuel de 5,5 ko, alors qu'il occupait une surface de 45 m² et pesait presque 8 tonnes. Pour le faire fonctionner, trois équipes de trente personnes se succédaient en continu.

Dans l'architecture de von Neumann, un ordinateur est composé de 4 parties :

- L'unité artithmétique et logique (UAL ou *ALU* en anglais) ou unité de traitement : son rôle est d'effectuer les opérations (calculs) de base ;
- L'unité de contrôle (UC ou *CU* en anglais pour Control Unit) : c'est le chef d'orchestre de l'ordinateur, elle récupère les instructions du programme en mémoire et les données sur lesquelles doivent s'opérer les instructions (via des bus de communication), puis les envoie à l'unité arithmétique et logique ;
- La mémoire qui contient les programmes ET les données, et qui indiquera à l'unité de contrôle quels sont les calculs à faire sur ces données ;
- Les dispositifs d'entrée-sortie pour communiquer avec l'extérieur

Le CPU (*Central Processing Unit* ou Unité Centrale de Traitement), aussi appelé processeur, regroupe à la fois l'unité artithmétique et logique et l'unité de contrôle.

Lorsque le processeur rassemble ce deux unités dans un seul et même circuit électronique, on parle de *microprocesseur*.



Fonctionnement (un peu) plus détaillé des composants

Nous décrivons dans ce paragraphe le fonctionnement des 4 parties de l'architecture de von Neumann. Ces informations prendront encore davantage de sens lorsque nous aborderons le *langage machine* à la fin de ce cours.

Unité de contrôle

Elle possède notamment deux *registres* (= mémoires internes très rapides) :

- le *registre d'instruction*, nommé **IR** (*Instruction Register* en anglais), qui contient l'instruction courante à exécuter ;
- le *compteur de programme*, nommé **PC** (*Program Counter* en anglais), qui indique l'emplacement mémoire de la prochaine instruction à exécuter.

Unité arithmétique et logique

Elle est composée notamment de :

- plusieurs registres, appelés *registres de données*, dans lesquels sont chargés les données sur lesquelles les instructions portent
- d'un registre particulier appelé *accumulateur* dans lequel vont s'effectuer tous les calculs
- plein de circuits électroniques pour réaliser des opérations arithmétiques (addition, soustraction, etc.), des opérations logiques (et, ou, complément à un, etc.), des comparaisons (égalité, inférieur, supérieur, etc.), des opérations sur les bits (décalages, rotations) ou des opérations de déplacement mémoire (copie de ou vers la mémoire).

Nous parlerons davantage de ces circuits électroniques dans un prochain chapitre, à suivre donc...

Les registres contiennent les valeurs de la mémoire de l'ordinateur nécessaires à l'exécution des instructions. Le résultat d'un calcul (arithmétique ou logique) est stocké dans l'accumulateur et peut être utilisé pour les instructions ultérieures de manière très rapide car cela évite les allers-retours en mémoire (l'accès aux registres

est très rapide comparé à l'accès aux données en mémoire).

La mémoire

Les échanges de données entre le processeur et la mémoire se font par l'intermédiaire de *bus de communication* .

La mémoire de l'ordinateur (à ne pas confondre avec le disque dur) est composée de plusieurs milliards de circuits mémoires qui sont organisés en agrégats de 8, 16, 32, 64 bits (voire plus) appelés cases mémoires. Leur nombre définit la taille mémoire de l'ordinateur et chaque case possède une *adresse* pour les distinguer. Comme dit précédemment, ces cases mémoires contiennent à la fois les programmes *et* les données sur lesquelles portent les programmes.

> Les registres du processeur sont également des cases mémoires, dont l'accès (lecture/écriture) est très rapide. Lorsque l'on parle de processeurs 32 bits, 64 bits, on fait en faire référence à la taille de ces registres.

Les périphériques d'entrée-sortie

Il en existe un très grand nombre que l'on peut classer en deux catégories : les *périphériques d'entrée* et les *périphériques de sortie* . Certains appartiennent à ces deux familles.

> Pouvez-vous citer quelques périphériques de chaque famille ? Et appartenant aux deux ?

Le langage machine

Dans cette partie, nous allons voir comment un ordinateur exécute un *programme*.

Un programme est une suite de nombres binaires placés en mémoire représentant des instructions exprimées en langage machine. C'est le seul langage que peut comprendre le processeur chargé d'exécuter ces différentes instructions.

Ainsi, un programme écrit dans un langage de programmation évolué (on dit de *haut niveau*) comme Python, Java, C, etc. ne peut pas être compris par le processeur. On doit donc utiliser un *compilateur* (pour le C, par exemple) ou un *interpréteur* (pour Python, par exemple) pour transformer le programme en langage machine.

Instruction machine

Une instruction machine est un mot binaire composé de deux parties :

- le champ *code opération* qui indique au processeur quelle opération il doit effectuer (charger une donnée en mémoire, faire une addition, une comparaison, etc.)
- le champ *opérandes* qui indique au processeurs la (les) donnée(s) sur laquelle (lesquelles) doit s'appliquer l'opération (l'adresse mémoire de la donnée à charger, les deux valeurs à additionner, les deux valeurs à comparer, etc.)

Une instruction machine possède le schéma suivant :

Nous allons voir des exemples tout de suite !

Le langage *assembleur*

Il n'est pas facile (voire impossible) pour un humain d'écrire directement les mots binaires d'un programme en langage machine. On peut utiliser à la place un langage d'assemblage, appelé assembleur, qui est le langage de plus *bas niveau* d'un ordinateur lisible par un humain. Nous allons détailler un langage d'assemblage pour que vous puissiez comprendre l'exécution d'une séquence d'instructions de type langage machine.

> Il existe plusieurs sortes de langage assembleur car ce dernier dépend du processeur utilisé.

Chaque instruction écrite en assembleur possède les deux champs "code opération" et "opérandes".

Dans la suite on utilisera le jeu d'instructions du simulateur RISC développé par Peter Higginson et disponible à cette adresse : https://peterhigginson.co.uk/RISC/ .

> On ne donne qu'une partie des instructions possibles, la liste complète est disponible ici. Vous utiliserez ce simulateur dans les activités pour bien fixer les idées.

Exemples d'instructions

Voici quelques instructions écrites en assembleur et leurs significations.

Instruction en assembleur	Signification
`LDR R1,12`	Charge dans le registre `R1` la valeur stockée à l'adresse mémoire `12`.
`STR R3,125`	Stocke la valeur du registre `R3` en mémoire vive à l'adresse `125`.
`ADD R1,#128`	Additionne le *nombre* `128` (une valeur immédiate est identifiée grâce au symbole `#`) et la valeur stockée dans le registre `R1`, stocke le résultat dans le registre `R1`.
`ADD R0,R1,R2`	Additionne la valeur stockée dans le registre `R1` et la valeur stockée dans le registre `R2`, stocke le résultat dans le registre `R0`.
`SUB R1,#128`	Soustrait le *nombre* `128` de la valeur stockée dans le registre `R1`, stocke le résultat dans le registre `R1`.
`SUB R0,R1,R2`	Soustrait la valeur stockée dans le registre `R2` de la valeur stockée dans le registre `R1`, stocke le résultat dans le registre `R0`.
`MOV R1,#23`	Place le nombre `23` dans le registre `R1`.
`MOV R0,R3`	Place la valeur stockée dans le registre `R3` dans le registre `R0`.
`HLT`	Arrête l'exécution du programme.

On trouve pour chacune un premier mot de 3 caractères qui correspond au "code opération" suivi d'un espace et des "opérandes".

Ainsi, le programme Python

```
a = 2
b = 7
c = a + b
```

pourrait s'écrire en assembleur de la façon suivante

```
MOV R0,#2
MOV R1,#7
ADD R1,R0,R1
STR R1,21
HLT
```

Dans ce cas, quels sont les états des registres à la fin du programme ? Où est stockée la valeur de la variable `c` à la fin du programme ?

Ce programme en assembleur peut être converti en une suite d'instructions machine pouvant être exécutées par le processeur. En l'occurrence, dans le cas du simulateur RISC cité plus haut, on obtiendrait, en utilisant la documentation, le code machine suivant (les cases mémoires sont de taille 16 bits ici) :

```
0010100000000010 0010100100000111 0110000001000001 1110001000010101 0
```

On peut voir les correspondances, dans le tableau ci-dessous, où on a écrit en rouge le champ *code opération* :

Instructions machine	Correspondance
0010100000000010	MOV R0,#2
0010100100000111	MOV R1,#7
0110000001000001	ADD R1,R0,R1
1110001000010101	STR R1,21
0000000000000000	HLT

En copiant le code assembleur dans la fenêtre de gauche du simulateur, on peut observer tout le déroulé de l'exécution du programme (on veillera à choisir l'option "binary" pour voir les mots en mémoire en binaire) : https://peterhigginson.co.uk/RISC/

Ce simulateur utilise des cases mémoires de 16 bits, les ordinateurs plus récents en ont de taille 64 bits généralement. De plus, le simulateur possède beaucoup d'autres instructions machine, certaines seront développées dans les activités

Cycle et horloge

Un CPU possède une horloge qui définit le rythme auquel les instructions sont exécutées. Pour exécuter une instruction, le processeur va effectuer ce qu'on appelle un cycle d'exécution qui s'effectue en trois étapes :

1. Chargement (*load* en anglais) : l'unité de contrôle récupère le mot binaire (qui contient la prochaine instruction à exécuter) situé en mémoire à l'adresse indiquée par son registre **PC** et la stocke dans son registre **IR** ;
2. Décodage : la suite de bits de l'instruction contenue dans le registre **IR** est décodée pour déduire quelle instruction est à exécuter et sur quelles données. Cette étape peut alors nécessiter de lire d'autres mots binaires depuis la mémoire ou les registres, pour charger les données (les "opérandes") sur lesquelles portent l'opération à effectuer.
3. Exécution : l'instruction est exécutée, soit par l'ALU s'il s'agit d'une opération arithmétique ou logique, soit par l'UC s'il s'agit d'une opération de branchement qui va modifier la valeur du registre **PC**.

Par exemple, un processesseur ayant une fréquence de 3,2 GHz va effectuer 3,2 milliards de cycles d'horloge par seconde.

La fréquence des processeurs a augmenté de manière linéaire depuis les premiers ordinateurs mais stagne depuis une vingtaine d'année car la chaleur produite devient trop importante et pourrait perturber la lecture des informations voire détériorer physiquement les circuits 🌡️🥵

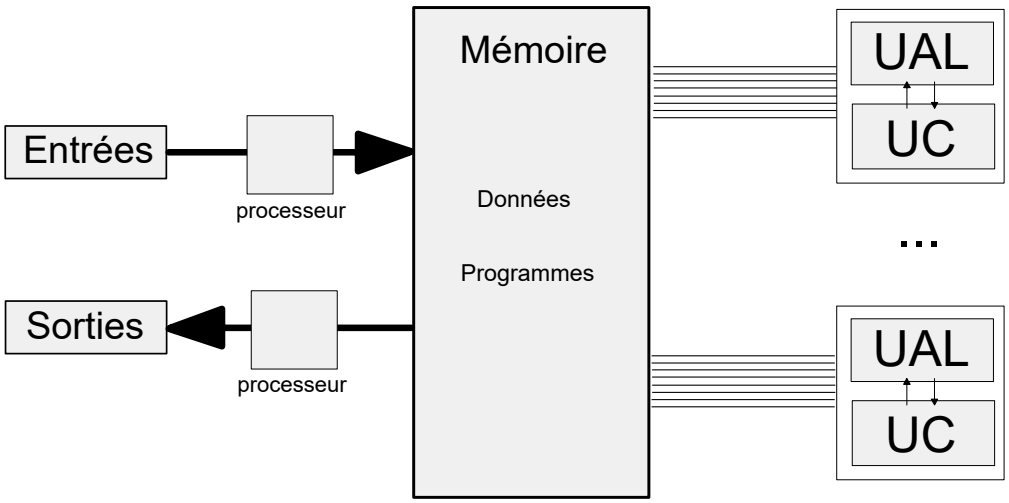
Qu'en est-il aujourd'hui ?

Le modèle de von Neumann défini en 1945 est toujours celui utilisé dans les ordinateurs actuels. Évidemment, les composants se sont de plus en plus miniaturisés et sont de plus en plus performant.

Par rapport au schéma initial, deux évolutions sont à noter :

- les entrées-sorties, intialement commandées par le CPU, sont depuis les années 1960 sous le contrôle de processeurs autonomes
- les ordinateurs sont désormais multiprocesseurs, c'està-dire qu'ils possèdent plusieurs processeurs, qu'il s'agisse d'unités séparées ou de *coeurs* multiples à l'intérieur d'une même puce. Cela permet d'atteindre une puissance de calculs élevée sans augmenter la vitesse des processeurs individuels, limitée par les capacités d'évacuation de la chaleur dans des circuits de plus en plus denses.

Ces deux évolutions ont pour conséquence de mettre la mémoire, plutôt que l'unité centrale de contrôle, au centre de l'ordinateur, et d'augmenter le degré de parallélisme dans le traitement et la circulation de l'information. Mais elles ne remettent pas en cause le modèle de von Neumann.



Bilan

- Les ordinateurs actuels sont basés sur le modèle de von Neumann, dans lequel la mémoire stocke à la fois les données et les programmes.
- Dans l'architecture de von Neumann, un ordinateur est composé de 4 parties : d'une unité arithmétique et logique (ALU), d'une unité de contrôle (UC), d'une mémoire et de dispositifs d'entrée-sortie.
- Le processeur (CPU), composé de l'UAL et de l'UC, est chargé d'exécuter des instructions écrites en langage machine, c'est-à-dire des mots écrits en binaire et stockés dans la mémoire.
- Tout programme doit être "converti" en langage machine afin d'être exécuté.
- Le langage assembleur est le langage de plus bas niveau lisible par un humain. Il permet donc de programmer "au niveau" du langage machine.