Architectures matérielles et OS	Chap 02	Modèle d'architecture d'un ordinateur

2.1 L'architecture de von Neumann

Un peu de contexte

Les grands principes de fonctionnement des ordinateurs actuels remontent à des travaux menés au milieu des années 1940. Ces travaux ont défini un schéma d'organisation appelé **architecture de von Neumann**, en référence à John von Neumann (1903–1957), un mathématicien et physicien américano-hongrois qui a publié ces travaux en 1945.

Ce qui est remarquable, c'est que ce modèle, défini il y a près de 80 ans, reste celui de **tous** les ordinateurs actuels : téléphones, consoles, serveurs, etc.

Fig. 1 – John von Neumann (via Wikimedia Commons)

Exercice 1 — Les composants d'un ordinateur

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

1. Associer le bon élément à chaque numéro du schéma ci-dessous.

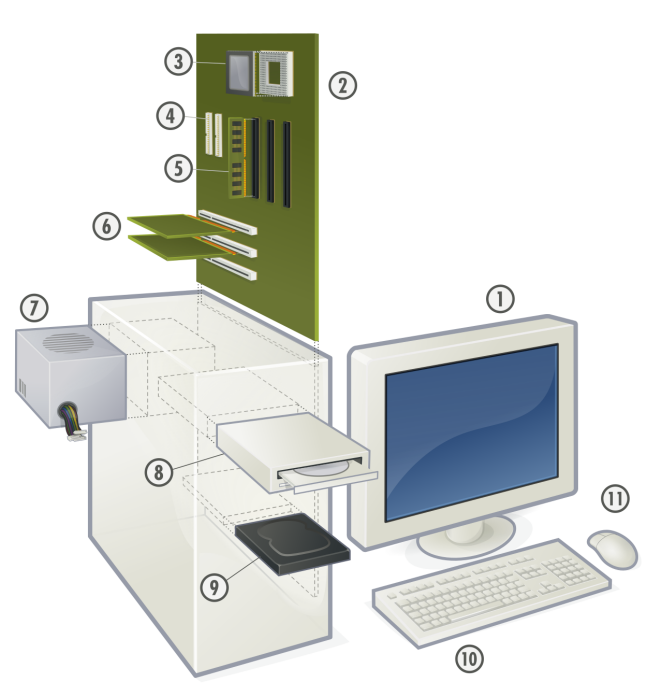


Fig. 2 – Les composants d'un ordinateur (Gustavb, CC BY-SA 3.0, via Wikimedia Commons)

1	.....	**7**	.....
2	.....	**8**	.....
3	.....	**9**	.....
4	.....	**10**	.....
5	.....	**11**	.....
6	.....		

2. Relier chaque composant à sa description.

Description	**Composant**
Cerveau de l'ordinateur : effectue les opérations (calculs) demandées	.....
Stocke les programmes et données en cours de fonctionnement	.....
Relie tous les éléments par des bus (circuits imprimés)	.....
Carte d'extension qui produit une image affichable sur un écran	.....
Périphérique d'entrée qui permet d'écrire	.....
Assure la mise sous tension de l'ensemble des composants	.....
Stocke et lit des données sur un support externe non volatile	.....
Lecteur de support amovible (CD, DVD)	.....
Périphérique d'entrée qui déplace le curseur de pointage à l'écran	.....
Permet de connecter les périphériques à la carte mère	.....
Périphérique de sortie qui affiche les informations de l'ordinateur	.....

2.2 Le principe de l'architecture

L'idée centrale de von Neumann

L'idée majeure de l'architecture de von Neumann est d'utiliser une **zone de stockage unique** – la mémoire de l'ordinateur – pour conserver à la fois :

- les **programmes**, c'est-à-dire les instructions à exécuter ;
- les **données** sur lesquelles ces instructions travaillent.

Avant cette idée, programmes et données étaient traités séparément. Les ranger dans une même mémoire a rendu les ordinateurs bien plus souples : un programme peut être chargé, remplacé ou modifié comme une simple donnée.

Un peu d'histoire : l'EDVAC

Les travaux publiés par von Neumann en 1945 concernaient la conception de l'**EDVAC**, l'un des premiers ordinateurs fondés sur cette architecture. Il savait additionner, soustraire, multiplier et diviser en binaire. Sa capacité mémoire équivalait à environ **5,5 ko** d'aujourd'hui, alors qu'il occupait **45 m²** et pesait près de **8 tonnes**. Pour le faire fonctionner, trois équipes de trente personnes se relayaient en continu.

Fig. 3 – L'EDVAC : un ordinateur qui occupait toute une pièce (U.S. Army, domaine public, via Wikimedia Commons)

Définition 1 — Les quatre parties d'un ordinateur

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

1. l'**unité arithmétique et logique** (UAL, ou **ALU** en anglais) : son rôle est d'effectuer les opérations de base (calculs, comparaisons) ;
2. l'**unité de contrôle** (UC, ou **Control Unit** en anglais) : c'est le chef d'orchestre. Elle récupère en mémoire les instructions du programme et les données à traiter, puis pilote leur exécution ;
3. la **mémoire**, qui contient les programmes **et** les données ;
4. les **dispositifs d'entrée-sortie**, qui permettent de communiquer avec l'extérieur (clavier, écran, etc.).

Processeur et microprocesseur

Le **processeur**, aussi appelé **CPU** (**Central Processing Unit**), regroupe l'unité arithmétique et logique **et** l'unité de contrôle.

Lorsque ces deux unités sont réunies dans un seul et même circuit électronique, on parle de **microprocesseur**.

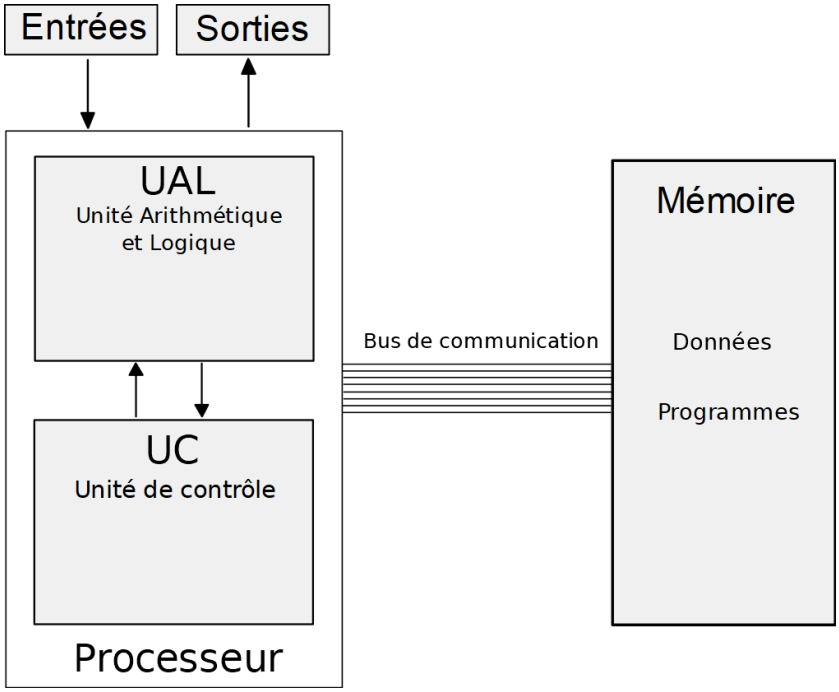


Fig. 3 – Schéma de l'architecture de von Neumann (adaptation d'une image de Schega, CC BY-SA 3.0, via Wikimedia Commons)

Remarque 1. Les échanges entre le processeur et la mémoire se font par l'intermédiaire de **bus de communication** : ce sont des ensembles de fils qui transportent les bits d'un composant à l'autre.

2.3 Fonctionnement détaillé des composants

2.3.1 Les registres : la mémoire interne du processeur

Définition 2 — Registre

Un **registre** est une mémoire interne au processeur, de très petite taille mais d'accès **extrêmement rapide**. Un processeur n'en possède qu'un petit nombre.

Des registres aux rôles différents

Le processeur possède plusieurs registres, qui n'ont pas tous le même rôle.

Dans l'**unité de contrôle** :

- le **registre d'instruction**, noté **IR** (**Instruction Register**), contient l'instruction en cours d'exécution ;
- le **compteur de programme**, noté **PC** (**Program Counter**), indique l'adresse mémoire de la **prochaine** instruction à exécuter.

Dans l'**unité arithmétique et logique** :

- plusieurs **registres de données** contiennent les valeurs sur lesquelles portent les calculs ;
- le résultat d'un calcul est rangé dans un registre, ce qui permet de le réutiliser très vite et d'éviter des allers-retours coûteux vers la mémoire.

Un processeur « 64 bits », qu'est-ce que cela signifie ?

Lorsqu'on parle d'un processeur **32 bits** ou **64 bits**, on fait référence à la **taille de ses registres**, c'est-à-dire au nombre de bits qu'un registre peut contenir.

2.3.2 La mémoire : une hiérarchie

Toutes les mémoires ne se valent pas

Un ordinateur ne possède pas un seul type de mémoire, mais **plusieurs**, qui se distinguent par deux critères opposés :

- la **rapidité** d'accès (lecture / écriture) ;
- la **capacité** de stockage.

Une mémoire très rapide coûte cher : on n'en met donc qu'une toute petite quantité, au plus près du processeur. Une mémoire de grande capacité est plus lente, mais bon marché. On organise alors ces mémoires en **hiérarchie**.

Définition 3 — Mémoire volatile / mémoire persistante

Une mémoire est dite **volatile** si son contenu est **effacé** lorsqu'on éteint l'ordinateur.

Elle est dite **persistante** (ou **non volatile**) si son contenu est **conservé** après l'extinction.

Définition 4 — Mémoire vive (RAM)

La **mémoire vive**, ou **RAM** (**Random Access Memory**), contient les **programmes et les données en cours d'utilisation**. Elle est rapide, de capacité moyenne (quelques Go), et **volatile**.

Définition 5 — Mémoire de masse

La **mémoire de masse** (**disque dur** ou **SSD**) assure le **stockage permanent** des fichiers, des logiciels et du système d'exploitation. Elle est de très grande capacité (plusieurs centaines de Go à plusieurs To), **persistante**, mais beaucoup plus lente que la mémoire vive.

Deux mémoires en bonus

La mémoire cache : une petite mémoire très rapide, intermédiaire entre le processeur et la mémoire vive, qui garde sous la main les données les plus utilisées.

La mémoire morte (ROM) : une mémoire **persistante** en lecture seule, qui contient les instructions de démarrage de la machine.

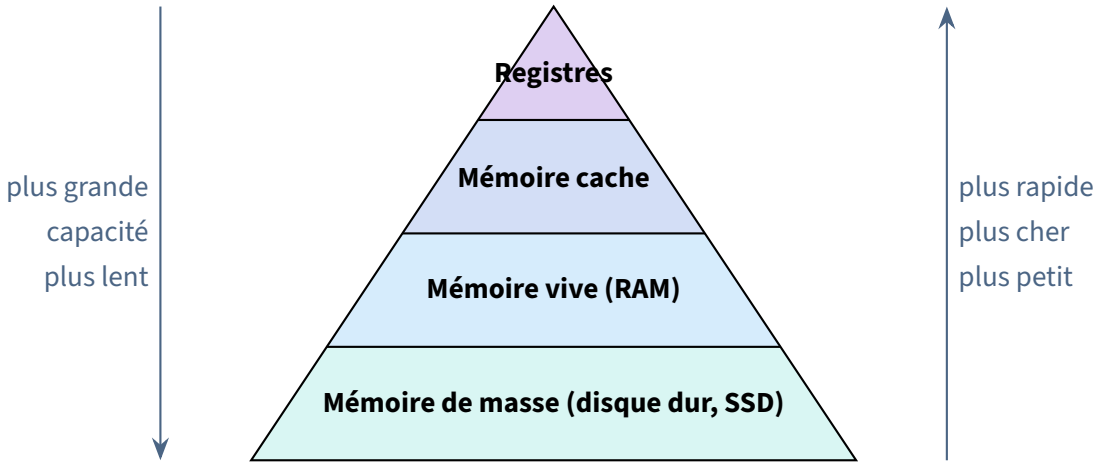


Fig. 6 – La hiérarchie des mémoires

Propriété 1. Plus une mémoire est proche du processeur, plus elle est **rapide** mais de **petite** capacité. Plus elle en est éloignée, plus elle est **lente** mais de **grande** capacité.

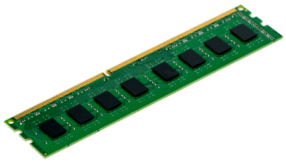
Ne pas confondre mémoire vive et mémoire de masse

La **RAM** et le **disque dur** sont deux mémoires différentes. La RAM est rapide mais **volatile** : tout ce qu'elle contient disparaît à l'extinction. Le disque dur est lent mais **persistant** : c'est lui qui conserve les fichiers.

Tout comme en cuisine !

On peut comparer ces mémoires au plan de travail d'un cuisinier :

- les **registres** sont ses deux mains : il n'y tient que quelques ingrédients, mais il agit dessus immédiatement ;
- la **mémoire vive** est le plan de travail : tout ce qu'il faut pour la recette en cours y est posé ;
- la **mémoire de masse** est le garde-manger : tout y est rangé durablement, mais il faut aller le chercher.



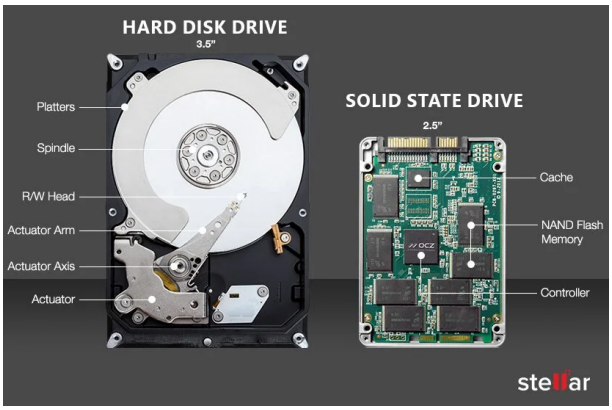


Fig. 7 – Une barrette de mémoire vive, un disque dur et un SSD

2.3.3 L'organisation de la mémoire

Définition 6 — Case mémoire et adresse

La mémoire est composée d'un très grand nombre de **cases mémoires**, c'est-à-dire de petits groupes de bits (8, 16, 32 ou 64 bits).
Chaque case possède une **adresse** qui permet de la repérer de manière unique. Comme l'avait prévu von Neumann, ces cases contiennent à la fois les **programmes** et les **données**.

Exercice 2 — Quelle mémoire pour quel usage ?

Pour chaque situation, indiquer quelle mémoire est concernée :

1. On enregistre une photo pour la retrouver dans un mois.

   ---

2. Le processeur additionne deux nombres au cours d'un calcul.

   ---

3. On ouvre un logiciel : son code est chargé pour pouvoir être exécuté.

   ---

4. Après une coupure de courant, un document non enregistré a disparu. Dans quelle mémoire se trouvait-il ?

2.4 Le langage machine

Le seul langage compris par le processeur

Un **programme** est une suite de nombres binaires placés en mémoire, qui représentent des **instructions** exprimées en **langage machine**. C'est le seul langage que le processeur sait exécuter.

Un programme écrit dans un langage de haut niveau (Python, Java, C, etc.) ne peut donc pas être compris directement par le processeur. Il faut le **traduire** en langage machine à l'aide :

— d'un **compilateur** (pour le C, par exemple) ;

— ou d'un **interpréteur** (pour Python, par exemple).

Définition 7 — Instruction machine

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

— le **champ code opération**, qui indique **quelle** opération effectuer (charger une donnée, additionner, comparer, etc.) ;

— le **champ opérandes**, qui indique **sur quoi** porte l'opération (une adresse mémoire, les valeurs concernées, etc.).

champ code opération	champ opérandes

2.5 Le langage assembleur

Un langage lisible par un humain

Écrire directement les mots binaires d'un programme est pratiquement impossible pour un humain. On utilise donc un **langage d'assemblage**, appelé **assembleur** : c'est le langage de plus bas niveau qui reste **lisible par un humain**.

Chaque instruction en assembleur possède elle aussi un **code opération** et des **opérandes**. Il existe plusieurs assembleurs différents, car le langage dépend du processeur utilisé. Dans ce cours, on utilise le jeu d'instructions du simulateur RISC de Peter Higginson.

Exemple 1 — Quelques instructions

Instruction	Signification
`LDR R1,12`	Charge dans `R1` la valeur stockée à l'adresse mémoire 12.
`STR R3,125`	Stocke la valeur de `R3` en mémoire à l'adresse 125.
`ADD R1,#128`	Ajoute la valeur immédiate 128 (repérée par #) à `R1`, range le résultat dans `R1`.
`ADD R0,R1,R2`	Additionne `R1` et `R2`, range le résultat dans `R0`.
`SUB R1,#128`	Soustrait 128 à `R1`, range le résultat dans `R1`.
`SUB R0,R1,R2`	Soustrait `R2` à `R1`, range le résultat dans `R0`.
`MOV R1,#23`	Place la valeur 23 dans `R1`.
`MOV R0,R3`	Recopie la valeur de `R3` dans `R0`.
`HLT`	Arrête l'exécution du programme.

Chaque instruction commence par un mot de trois lettres (le code opération), suivi des opérandes.

Exemple 2 — Du Python à l'assembleur

Le programme Python suivant :

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

peut s'écrire en assembleur de la façon suivante :

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

Question

À la fin de ce programme, quelles sont les valeurs contenues dans les registres `R0` et `R1` ? Où est stockée la valeur de la variable `c` ?

2.5.1 Conversion en langage machine

Du texte au binaire

Le jeu d'instructions du simulateur donne la correspondance entre chaque instruction assembleur et son instruction machine. Le programme précédent se traduit ainsi en une suite de mots binaires (ici, des cases mémoires de 16 bits) :

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

À tester dans le simulateur

En copiant le code assembleur dans la fenêtre de gauche du simulateur (`peterhigginson.co.uk/RISC`), on peut observer tout le déroulé de l'exécution, instruction par instruction. Choisir l'option « binary » pour afficher les mots de la mémoire en binaire.

2.6 Le cycle d'exécution

Le rythme du processeur

Un processeur possède une **horloge** qui fixe le rythme auquel les instructions sont exécutées. Pour chaque instruction, le processeur effectue un **cycle d'exécution** en trois étapes.

Définition 8 — Les trois étapes d'un cycle

1. **Chargement** (**fetch**) : l'unité de contrôle récupère en mémoire, à l'adresse indiquée par `PC`, le mot binaire de la prochaine instruction et le place dans `IR`.
2. **Décodage** (**decode**) : l'instruction contenue dans `IR` est analysée pour déterminer l'opération à effectuer et les données concernées.
3. **Exécution** (**execute**) : l'instruction est réalisée, soit par l'UAL (calcul), soit par l'UC (modification de `PC` en cas de branchement).

Exemple 3 — Fréquence d'un processeur

Un processeur cadencé à **3,2 GHz** effectue environ **3,2 milliards** de cycles d'horloge par seconde.

2.7 Qu'en est-il aujourd'hui ?

Un modèle toujours d'actualité

Le modèle défini par von Neumann en 1945 reste celui de tous les ordinateurs actuels. Les composants ont simplement été miniaturisés et rendus bien plus performants. Deux évolutions sont toutefois à noter :

— les **entrées-sorties**, autrefois pilotées par le CPU, sont depuis les années 1960 gérées par des processeurs autonomes ;

— les ordinateurs sont devenus **multiprocesseurs** : ils possèdent plusieurs processeurs, ou plusieurs **cœurs** sur une même puce, ce qui augmente la puissance de calcul sans augmenter la fréquence (limitée par la chaleur).

Ces évolutions placent désormais la **mémoire** au centre de l'ordinateur, mais ne remettent pas en cause le modèle de von Neumann.

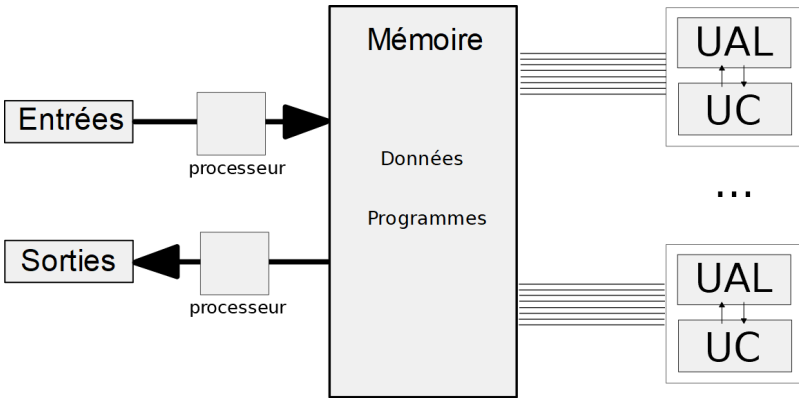


Fig. 8 – Architecture d'un ordinateur multiprocesseur (via Wikimedia Commons)

2.8 Bilan

Ce qu'il faut retenir

— Les ordinateurs actuels reposent sur le **modèle de von Neumann**, où une même mémoire contient les **données** et les **programmes**.

— Un ordinateur comporte quatre parties : l'**unité arithmétique et logique**, l'**unité de contrôle**, la **mémoire** et les **dispositifs d'entrée-sortie**.

— Le **processeur** (CPU), formé de l'UAL et de l'UC, exécute des instructions en **langage machine** (des mots binaires).

— La mémoire est organisée en **hiérarchie**, des registres (rapides et petits) à la mémoire de masse (lente et de grande capacité).

— Tout programme doit être **traduit** en langage machine pour être exécuté ; l'**assembleur** est le langage de plus bas niveau lisible par un humain.

Cours fortement inspiré de https://info-mounier.fr/premiere_nsi/archi_os/modele-archi