🖥️ L'Architecture de Von Neumann

Synthèse générée par Chat GPT à partir de la version longue de mon cours

📌 Qu'est-ce qu'un ordinateur ?

Un **ordinateur** est une machine *programmable*, *automatique* et *universelle* :

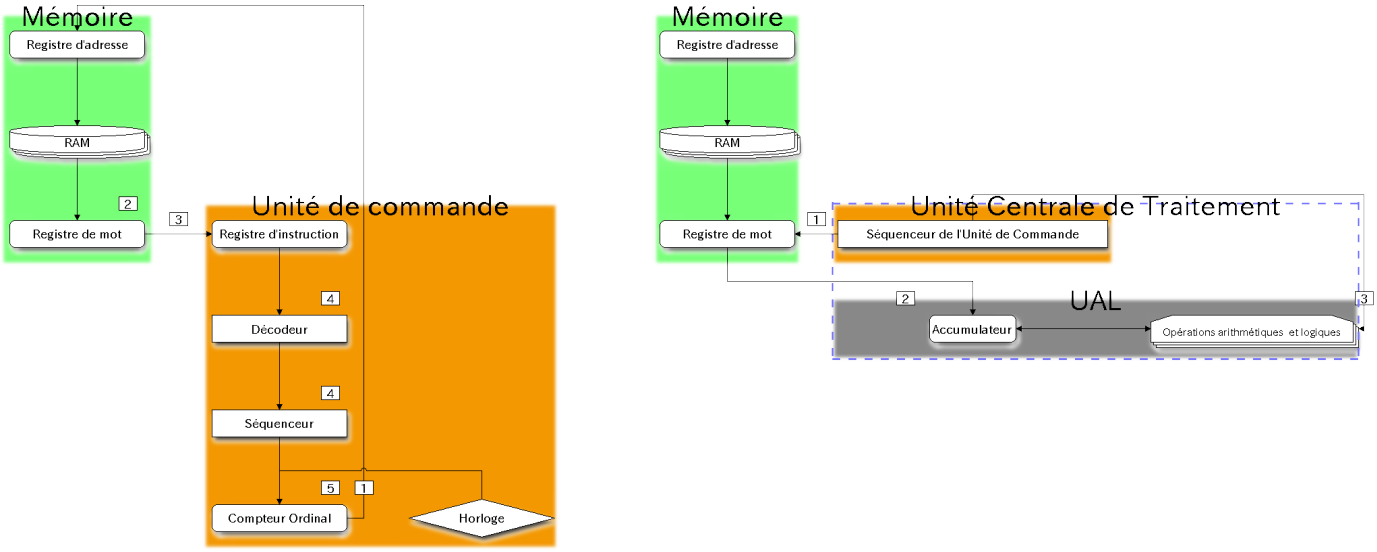
- *programmable* : la séquence d'opérations exécutée par un ordinateur peut être entièrement spécifiée dans le texte d'un programme ;
- *automatique* : un ordinateur peut exécuter un programme sans intervention extérieure (câblage) ;
- *universelle* : un ordinateur peut exécuter tout programme calculable (selon le modèle théorique de la machine d'Alan Turing) avec le jeu d'instructions de son processeur.

Dans le modèle de **Von Neumann**, proposé par John Von Neumann en 1945, un ordinateur est composé des éléments suivants :

1. **Unité Centrale de Traitement (CPU)** : Composée de :
   - **L'Unité de Commande (UC)** : orchestre l'exécution des instructions.
   - **L'Unité Arithmétique et Logique (UAL)** : réalise les calculs et opérations logiques.
2. **Mémoire Centrale** : Stocke **les programmes et les données**.
3. **Bus** : Assure la communication entre le processeur, la mémoire et les périphériques.
4. **Entrées/Sorties** : Permettent d'interagir avec l'extérieur (clavier, écran, disque dur, etc.).

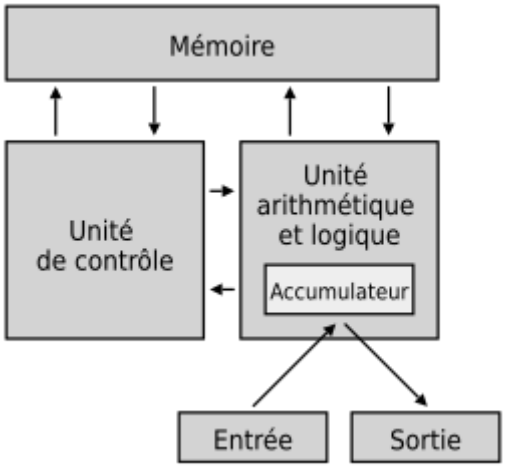
Le fonctionnement repose sur le **cycle d'instruction** qui est composé :

- d'un *cycle de recherche* :
  - **Fetch** : Récupération de l'instruction en mémoire.
  - **Decode** : Décodage de l'instruction par l'Unité de Commande.
- d'un *cycle d'exécution*
  - **Execute** : Exécution de l'instruction par l'UAL sous la commande du séquenceur
  - **Write back** : Écriture du résultat en mémoire.



Cycle de recherche à gauche et cycle d'exécution à droite

🔍 Schéma de l'architecture de Von Neumann



Source : Wikipedia

📌 Hiérarchie des mémoires

Les mémoires sont classées selon leur vitesse et leur persistance :

- **Registres** : Stockage ultra-rapide intégré au processeur.
- **Cache** : Mémoire rapide située entre le processeur et la RAM.
- **RAM** : Stockage temporaire des instructions et données en cours de traitement.
- **Stockage secondaire (disque dur, SSD)** : Stockage persistant des données et programmes.

📌 Miniaturisation des Ordinateurs et Loi de Moore

Depuis l'apparition du premier microprocesseur, l'**Intel 4004** en 1971, la taille des composants électroniques n'a cessé de diminuer, tandis que leur puissance a considérablement augmenté. Ce processeur contenait seulement **2 300 transistors** et fonctionnait à **740 kHz**. Au fil des décennies, la **miniaturisation des transistors** a permis d'accroître les performances des ordinateurs tout en réduisant leur consommation d'énergie.

La **loi de Moore**, énoncée par Gordon Moore en 1965, prédit que le nombre de transistors sur une puce double environ tous les **18 à 24 mois**, entraînant une amélioration exponentielle de la puissance de calcul. Cette tendance a été observée pendant plusieurs décennies. Depuis les années 2000, émergent des architectures avec **processeurs multi-cœurs** (puce unique avec plusieurs unités de calculs) ou **multiprocesseurs** (plusieurs processeurs physique partageant une même mémoire).

Avec la miniaturisation toujours plus poussée, les composants des ordinateurs ont été progressivement **intégrés sur une seule puce**, donnant naissance aux **System on Chip (SoC)**. Ces puces, présentes dans les **smartphones, tablettes et objets connectés**, combinent processeur, mémoire, unités graphiques et gestion des entrées/sorties en un seul circuit intégré. Cette intégration permet une **réduction de la taille et de la consommation d'énergie** tout en améliorant les performances globales.

📌 Du langage de programmation au langage machine

Un programme écrit dans un langage de haut niveau (ex : Python, C, Java) doit être **traduit en langage machine** pour être compris par l'ordinateur. Cette traduction passe par plusieurs étapes :

🔍 Langage de haut niveau

- Écriture du programme en un langage compréhensible par l'humain.
- Exemple en C :

```c
#include <stdio.h>
int main() {
    int a = 10, b = 20;
    int c = a + b;
    printf("Résultat : %d\n", c);
    return 0;
}
```

🔍 Langage d'assemblage (assembleur)

- Traduction en instructions plus proches du processeur, mais encore lisibles par un humain.
- Utilisation de **mnémoniques** représentant les instructions machine.
- Exemple en assembleur :

```
MOV R0, #10    // Charge 10 dans le registre R0
MOV R1, #20    // Charge 20 dans le registre R1
ADD R2, R0, R1 // Additionne R0 et R1 et stocke le résultat dans R2
STR R2, 100    // Stocke le résultat à l'adresse mémoire 100
HALT           // Arrête l'exécution du programme
```

🔍 Langage machine (code binaire)

- Traduction finale en une suite de **0 et 1** compréhensible par le processeur.
- Exemple :

```
101010110101 110101011010 110100101101
```

- Ce code est directement exécutable par l'ordinateur.

🔍 Outils de traduction :

- **Compilateur** : Transforme le code source (C, Java…) en assembleur puis en langage machine (ex : GCC pour C, Java Compiler pour Java).
- **Assembleur** : Convertit un programme en assembleur en langage machine.
- **Interpréteur** : Exécute directement le code source sans compilation préalable (ex : Python). Cette structure de traduction permet une **portabilité** des programmes tout en assurant une exécution rapide et optimisée sur les machines. 🚀