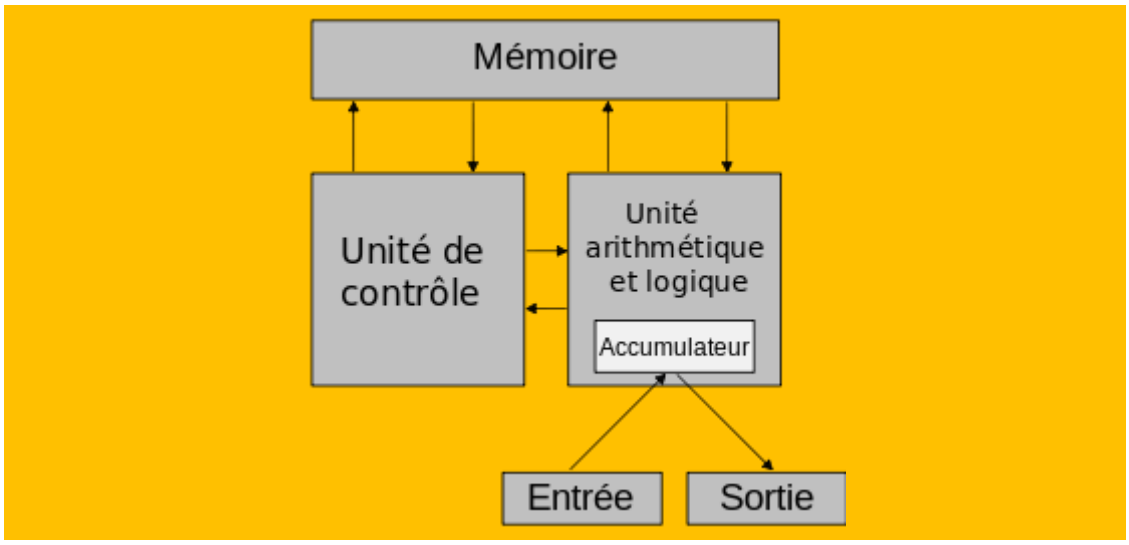
Architecture de von Neumann

- L'architecture dite **architecture de von Neumann** est un modèle pour un ordinateur qui utilise une structure de stockage unique pour conserver à la fois les instructions et les données demandées ou produites par le calcul.
- De telles machines sont aussi connues sous le nom d'ordinateur à programme enregistré.
- La séparation entre le stockage et le processeur est implicite dans ce modèle.
- Cette architecture est appelée ainsi en référence au mathématicien **John von Neumann** qui a élaboré la première description d'un ordinateur dont le programme est stocké dans sa mémoire.



Schématisation de l'architecture de von Neumann.

- Il y a quatre étapes que presque toutes les architectures de von Neumann utilisent :

1. ***Fetch***, recherche de l'instruction.
2. ***Decode***, interprétation de l'instruction (opération et opérandes).
3. ***Execute***, exécution de l'instruction.
4. ***Writeback***, écriture du résultat.

1. La première étape, ***fetch*** (recherche de l'instruction), recherche une instruction dans la mémoire vive de l'ordinateur.

- L'emplacement dans la mémoire est déterminé par le compteur de programme (**PC**), qui stocke l'adresse de la prochaine instruction dans la mémoire de programme.
- Après qu'une instruction ait été recherchée, le **PC** est incrémenté par la longueur du **mot d'instruction**.
- Dans le cas de mot de longueur constante simple, c'est toujours le même nombre.
- Par exemple, un mot de 32 bits de longueur constante qui emploie des mots de 8 bits de mémoire incrémenterait toujours le PC par 4 (excepté dans le cas des branchements).
- Le jeu d'instructions qui emploie des instructions de longueurs variables comme **l'x86**, incrémentent le PC par le nombre de mots de mémoire correspondant à la dernière longueur d'instruction.
- En outre, dans des processeurs plus complexes, l'incrémentation du PC ne se produit pas nécessairement à la fin de l'exécution d'une instruction.
- C'est particulièrement le cas dans des architectures fortement parallélisées et superscalaires.
- Souvent, la recherche de l'instruction doit être opérée dans des mémoires lentes, ralentissant le processeur qui attend l'instruction.
- Cette question est en grande partie résolue dans les processeurs modernes par l'utilisation de caches et de pipelines.

2. La seconde étape, ***decode*** (interprétation de l'instruction), découpe l'instruction en plusieurs parties telles qu'elles puissent être utilisées par d'autres parties du processeur.
   - La façon dont la valeur de l'instruction est interprétée est définie par le jeu d'instructions du processeur.
   - Souvent, une partie d'une instruction, appelée **opcode** (**code d'opération**), indique l'opération à effectuer, par exemple une addition.
   - Les parties restantes de l'instruction comportent habituellement les **opérandes** de l'opération.
   - Ces opérandes peuvent prendre une valeur constante, appelée valeur immédiate, ou bien contenir l'emplacement où retrouver (dans un registre ou une adresse mémoire) la valeur de l'opérande, suivant le **mode d'adressage** utilisé.

- Dans les conceptions anciennes, les parties du processeur responsables de l'interprétation étaient fixes et non modifiables car elles étaient codées dans les circuits.
- Dans les processeurs plus récents, un **microprogramme** est souvent utilisé pour l'interprétation.
- Ce microprogramme est parfois modifiable pour changer la façon dont le processeur interprète les instructions, même après sa fabrication.

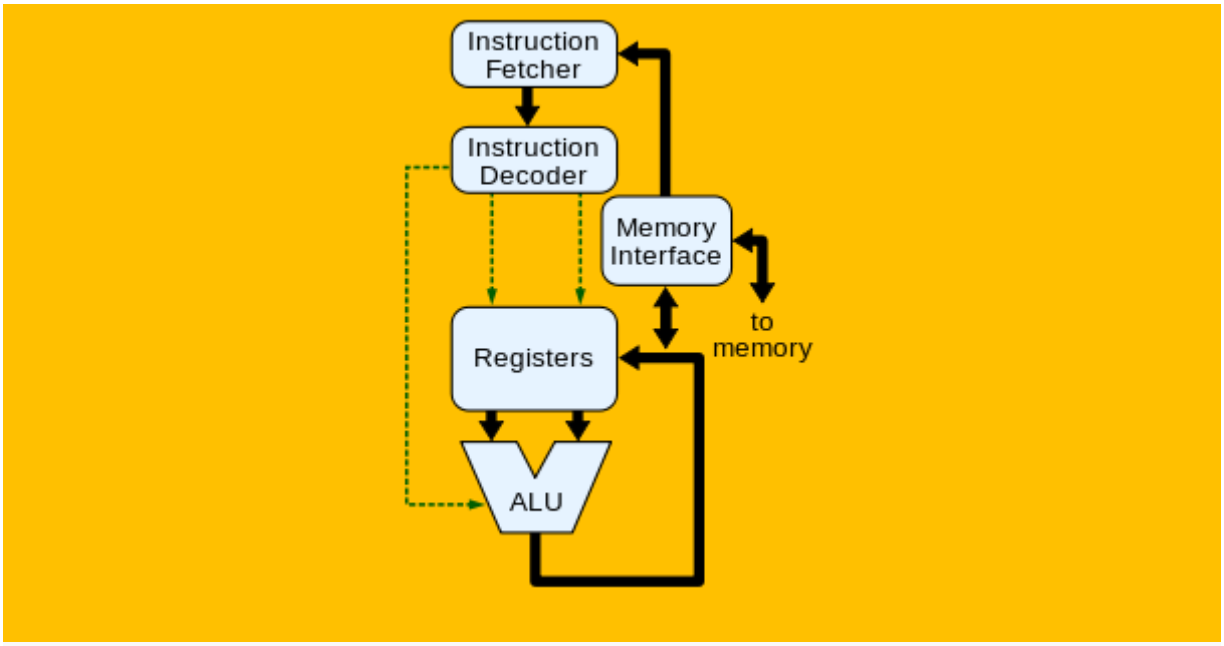


Diagramme fonctionnel d'un processeur simple.

3. La troisième étape, ***execute*** (exécution de l'instruction), met en relation différentes parties du processeur pour réaliser l'opération souhaitée.
   - Par exemple, pour une addition, l'unité arithmétique et logique (**ALU**) sera connectée à des entrées et une sortie.
   - Les entrées contiennent les nombres à additionner et la sortie contient le résultat.
   - L'ALU est dotée de circuits pour réaliser des opérations d'arithmétique et de logique simples sur les entrées (addition, opération sur les bits).
   - Si le résultat d'une addition est trop grand pour être codé par le processeur, un signal de débordement est positionné dans un registre d'état.
4. La dernière étape, ***writeback*** (écriture du résultat), écrit les résultats de l'étape d'exécution en mémoire.

- Très souvent, les résultats sont écrits dans un registre interne au processeur pour bénéficier de temps d'accès très courts pour les instructions suivantes.
- Parfois, les résultats sont écrits plus lentement dans la mémoire vive pour bénéficier de codages de nombres plus grands.
- Certains types d'instructions manipulent le compteur de programme (PO) plutôt que de produire directement des données de résultat.
- Ces instructions sont appelées des **branchements** (***branch***) et permettent de réaliser des boucles (***loops***), des programmes à exécution conditionnelle ou des fonctions (sous-programmes) dans des programmes.
- Beaucoup d'instructions servent aussi à changer l'état de **drapeaux** (*flags*) dans un registre d'état.
- Ces états peuvent être utilisés pour conditionner le comportement d'un programme, puisqu'ils indiquent souvent la fin d'exécution de différentes opérations.
- Par exemple, une instruction de comparaison entre deux nombres va positionner un drapeau dans un registre d'état suivant le résultat de la comparaison.
- Ce drapeau peut alors être réutilisé par une instruction de saut pour poursuivre le déroulement du programme.
- Après l'exécution de l'instruction et l'écriture des résultats, tout le processus se répète,
- le prochain cycle d'instructions recherche l'instruction suivante puisque le compteur de programme avait été incrémenté.
- Si l'instruction précédente était un saut, c'est l'adresse de destination du saut qui est enregistrée dans le compteur de programme.
- Dans des processeurs plus complexes, plusieurs instructions peuvent être recherchées, décodées et exécutées simultanément, on parle alors d'architecture pipeline, aujourd'hui communément utilisée dans les équipements électroniques.