1 Constituants d'un ordinateur

Un ordinateur contient une **unité centrale** et des **périphériques** (clavier, souris, écran, disque dur, carte graphique, carte réseau, imprimante, etc.)

L'unité centrale comprend la **carte mère** sur laquelle sont branchés les différents composants et connectée à l'alimentation électrique.

Les éléments essentiels sont le **processeur** et la **mémoire**.

Voir en ligne `https://www.youtube.com/watch?v=Fk2kYo2E61A`

2 Les mémoires

On appelle mémoire tout composant électronique capable de stocker des données.

Les mémoires sont caractérisées entre autres par :
- la capacité, représentant le volume global d'informations (en bits) que la mémoire peut stocker.
- le temps d'accès, correspondant à l'intervalle de temps entre la demande de lecture/écriture et la disponibilité de la donnée.
- la non-volatilité caractérisant l'aptitude d'une mémoire à conserver les données lorsqu'elle n'est plus alimentée électriquement.

Les mémoires rapides sont également les plus onéreuses. C'est la raison pour laquelle des mémoires utilisant différentes technologiques sont utilisées dans un ordinateur. Les mémoires les plus rapides sont situées en faible quantité à proximité du processeur et les mémoires de masse, moins rapides, servent à stocker les informations de manière permanente.

Mémoire vive : La mémoire vive, généralement appelée **RAM** (Random Access Memory, mémoire à accès direct), est la **mémoire principale** du système : il s'agit d'un espace permettant de stocker de manière temporaire des données lors de l'exécution d'un programme. Elle est volatile, c'est-à-dire qu'elle permet uniquement de stocker des données tant qu'elle est alimentée électriquement.

Mémoire morte : La mémoire morte, appelée ROM (Read Only Memory) est un type de mémoire permettant de conserver les informations qui y sont contenues même lorsque la mémoire n'est plus alimentée électriquement.

Mémoire flash : La mémoire flash est un compromis entre les mémoires de type RAM et les mémoires mortes. En effet, la mémoire flash possède la non-volatilité des mémoires mortes tout en pouvant facilement être accessible en lecture ou en écriture (ex : clef USB).

Mémoire cache : La mémoire cache accélère le fonctionnement de l'ordinateur en stockant les données utilisées le plus récemment. C'est la mémoire la plus rapide de tout l'ordinateur. Elle est située proche du processeur.

3 Le processeur

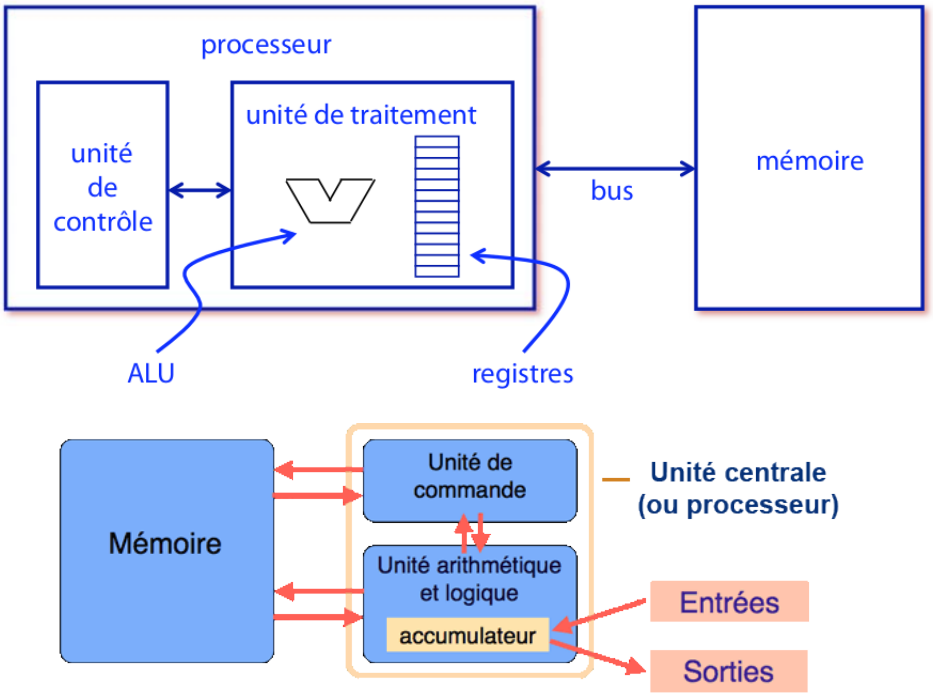
3.1 Description

Le **processeur** est le cœur de l'ordinateur : **CPU (Central Processing Unit)**.

Son fonctionnement est **cadencé par l'horloge interne du système** qui rythme chaque opération élémentaire (à quelques GHz sur les processeurs modernes).

Les éléments constitutifs du processeur sont :

- L'**Unité Arithmétique et Logique (UAL)** : c'est l'UAL qui effectue les calculs de l'ordinateur : opérations élémentaires, opérations logiques et opérations de comparaison.
- Des **registres** : zones mémoires à accès rapide dans laquelle l'UAL peut lire ou écrire des données.
- L'**Unité de Contrôle (ou de Commande)** (UC) : prend ses instructions dans la mémoire. Celles-ci lui indiquent ce qu'elle doit ordonner à l'UAL et comment elle devra éventuellement agir selon les résultats que celle-ci lui fournira. Une fois l'opération terminée, l'unité de contrôle passe soit à l'instruction suivante, soit à une autre instruction à laquelle le programme lui ordonne de se brancher.



La **mémoire** peut être décrite comme une suite de **cellules numérotées contenant chacune une petite quantité d'information**s. Cette information peut servir à indiquer à l'ordinateur ce qu'il doit faire, les **instructions**, et contenir des **données** à traiter.

Les **bus** de liaison :
Les différentes parties sont reliées par trois bus. Un bus est un groupement d'un certain nombre de fils électriques réalisant une liaison pour transporter des informations binaires.

- Le **bus d'adresse** transporte les adresses pour sélectionner une case mémoire ou un registre interne.
- Le **bus de données** transporte les données échangées entre les différents éléments du système.
- Le **bus de contrôle** transporte les différents signaux de synchronisation nécessaires au fonctionnement du système : signal de lecture, signal d'écriture, signal de sélection.

Les **entrées/sorties** : les dispositifs d'entrée/sortie permettent à l'ordinateur de communiquer avec l'extérieur (clavier, écran, imprimantes, etc.).

3.2 Architecture séquentielle de Von Neumann

Dans les premiers ordinateurs, les différents pas nécessaires à l'exécution d'une tâche, le programme, étaient directement câblés dans l'unité de contrôle (ruban, cartes perforées, tableau de connexions).

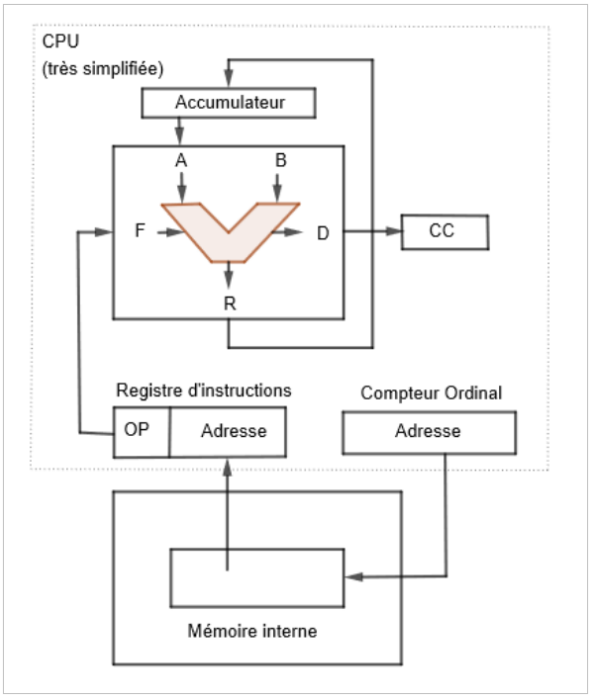
> Un grand progrès a été effectué lorsque le programme, comme les données, a été codé et stocké dans la mémoire principale : c'est l'**architecture de Von Neumann** (1945).

Un changement de programme se fait alors par une simple réécriture de la mémoire. Un emplacement de mémoire peut contenir indifféremment des instructions et des données, et une conséquence majeure est qu'**un programme peut être traité comme une donnée par d'autres programmes**.

> Selon cette architecture, la fonction de l'unité de contrôle est de :
>
> 1. **lire une instruction (programme) de la mémoire (fetch).**
> 2. **décoder l'instruction (decode).**
> 3. **commander son exécution (execute).**

3.3 Précision sur l'UC et l'UAL

Le schéma de fonctionnement suivant donne quelques détails complémentaires :



L'unité de contrôle organise le flot de séquencement des instructions.
Elle est principalement constituée de quatre registres :

- CO : le **compteur ordinal** (ou PC Program Counter), qui contient l'adresse de la prochaine instruction à exécuter. Il est automatiquement incrémenté après exécution de l'instruction ou explicitement modifié par les instructions de branchement.
- RI : le **registre d'instruction** qui contient l'instruction en cours d'exécution.
- ACC : l'**accumulateur** chargé de stocker des opérandes intermédiaires.
- CC : le **code condition**, utilisé pour les instructions de rupture conditionnelle (branchement).

Concernant l'unité arithmétique, on trouve :
- A et B (opérandes) : ce sont les entrées (A est le registre accumulateur).
- R : le résultat.
- F : une fonction binaire, c'est l'opération à effectuer (décodée par l'UC).
- D : un « drapeau » (flag) indiquant un résultat secondaire de la fonction (signe, erreur, division par zéro, dépassement, etc).

Deux articles complémentaires à lire en ligne sur Interstice :
- Interstices : mémoire et unité centrale (2019)
- Interstices : architecture de Von Neumann (2011)

3.4 Évolution

Pendant des années, pour augmenter les performances des ordinateurs, les constructeurs augmentaient la fréquence d'horloge des microprocesseurs : la fréquence d'horloge d'un microprocesseur est liée à sa capacité d'exécuter un nombre plus ou moins important d'instructions machines par seconde. **Plus la fréquence d'horloge du CPU est élevée, plus ce CPU est capable d'exécuter un grand nombre d'instructions machines par seconde**.

La fréquence a augmenté de façon exponentielle jusque vers les années 2005-2010 où cette évolution a commencé à être nettement ralentie. Ceci est dû à une contrainte physique : en effet plus on augmente la fréquence d'horloge d'un CPU, plus ce dernier chauffe. Il devenait difficile de refroidir le CPU, les constructeurs de microprocesseurs (principalement Intel et AMD) ont décidé alors d'arrêter la course à l'augmentation de la fréquence d'horloge et ont décidé d'adopter une nouvelle tactique.

L'idée est d'augmenter les performances en augmentant le nombre de cœurs présents sur un CPU : il y a donc plusieurs ensembles {UAL, registres et unité de contrôle}. Un cœur est donc capable d'exécuter des programmes de façon autonome. La technologie permettant de graver toujours plus de transistors sur une surface donnée, il est donc possible, sur une même puce, d'avoir plusieurs cœurs, alors qu'auparavant on trouvait un seul cœur dans un CPU. Cette technologie a été implémentée dans les ordinateurs grand public à partir de 2006, et même les smartphones possèdent des microprocesseurs multicœurs.

Cependant l'augmentation du nombre de cœurs n'entraîne pas obligatoirement une augmentation des performances du CPU. En fait, pour une application qui n'aura pas été conçue pour fonctionner avec un microprocesseur multicœur, le gain de performance sera très faible. La conception d'applications capables de tirer profit d'un CPU multicœur demande la mise en place de certaines techniques de programmation complexes. Il faut aussi avoir conscience que les différents cœurs d'un CPU doivent se "partager" l'accès à la mémoire vive : quand un cœur travaille sur une certaine zone de la RAM, cette même zone n'est pas accessible aux autres cœurs, ce qui, bien évidemment va brider les performances.

4 Langage machine

4.1 Principes

Un ordinateur exécute des programmes qui sont des suites d'instructions. Les instructions exécutées au niveau du CPU sont **codées en binaire**. L'ensemble des instructions reconnues par un processeur et son système de codage forment ce qu'on appelle le **langage machine** du processeur. Il est **spécifique pour chaque processeur** !

Une instruction machine est une chaîne binaire composée principalement de 2 parties :
- le champ « code opération » qui indique au processeur le type de traitement à réaliser. Par exemple le code `1110010110011111` donne l'ordre au CPU d'ajouter le contenu d'une mémoire dans un registre.
- le champ « opérandes » indique la nature des données sur lesquelles l'opération désignée par le "code opération" doit être effectuée.

Les instructions machines sont relativement basiques.
Voici quelques exemples :
- les **instructions arithmétiques ou logiques** (addition, multiplication, etc.).
  Exemple : "additionne la valeur contenue dans le registre R1 et le nombre 789 et range le résultat dans le registre R0".
- les **instructions de transfert de données** qui permettent de transférer une donnée d'un registre vers la mémoire vive et vice-versa.
  Exemple : "prendre la valeur située à l'adresse mémoire 487 et la placer dans le registre R2".
- les **instructions de rupture de séquence (branchement)**.
  Normalement au cours de l'exécution d'un programme, le CPU passe d'une instruction à une autre en passant d'une adresse mémoire à l'adresse mémoire immédiatement supérieure en incrémentant le compteur ordinal (PC). Les instructions de rupture de séquence d'exécution permettent d'interrompre l'ordre initial sous certaines conditions en passant à une instruction située une adresse mémoire donnée.
  Exemple : "si la valeur contenue dans le registre R1 est strictement supérieure à 0 alors exécuter l'instruction située à l'adresse mémoire 4521".
  Dans le cas contraire, la prochaine instruction à exécuter est à l'adresse mémoire immédiatement supérieure à celle de l'instruction en cours.

Un opérande peut être de 3 natures différentes :
- l'opérande est une *valeur immédiate* : l'opération est effectuée directement sur cette valeur.
- l'opérande est un *registre* du CPU : l'opération est effectuée sur la valeur située dans le registre.
- l'opérande est une donnée située en *mémoire vive* : l'opération est effectuée sur la valeur située dans la RAM à la bonne adresse.

Un programme en langage machine est donc une suite trèèèèès longue de 1 et de 0 !
Programmer en langage machine est donc extrêmement difficile et pour pallier cette difficulté, les codes binaires sont remplacés par des symboles **mnémoniques** qui forment le langage **assembleur**. L'assembleur est aussi le nom du logiciel qui convertit un programme assembleur en langage machine.

Exemple :
L'instruction machine "écrit la valeur immédiate 7 dans le registre 5 ", pourra s'écrire `MOV R5, #7` au lieu d'écrire `11100011101000000010000 00000111`.

4.2 Exemple de jeu d'instructions

Voici quelques exemples d'instructions d'un langage assembleur que nous utiliserons en TP (simulateur online de Peter Higginson `https://www.peterhigginson.co.uk/AQA/`) :

Dans le tableau suivant, <operande> peut soit être une *valeur immédiate* s'il commence par un # ou la valeur stockée dans d'un *registre* s'il commence par un R.
(ex : #25 est le nombre 25 ; R6 désigne le registre n° 6).

Un <label> identifie une adresse en mémoire vive dans le programme assembleur. (ex : `fin:` HALT).

LDR Rd, <adresse>	Charge la valeur stockée à l'adresse mémoire <adresse> dans un registre (LoaD Register)
STR Rd, <adresse>	Place la valeur stockée dans le registre d en mémoire vive à l'adresse <adresse> (STore Register)
ADD Rd, Rn, <operande>	Additionne <opérande> et le registre n, place le résultat dans le registre d (ADD)
SUB Rd, Rn, <operande>	Soustrait <opérande> de la valeur stockée dans le registre n, place le résultat dans le registre d (SUBstract)
MOV Rd, <operande>	Copie <opérande> dans le registre d (MOVe)
CMP Rn, <operande>	Compare le registre d et <opérande> dans le registre d (CoMPare). Précède une instruction de branchement conditionnel.
B <label>	Branchement inconditionnel vers l'instruction située en mémoire vive à l'adresse identifiée par <label>
B<condition> <label>	Branchement conditionnel à l'adresse identifiée par <label> si la dernière comparaison satisfait le critère spécifié par <condition>. Les valeurs possibles de <condition> sont : EQ : égal à (EQual), NE : différente (Not Equal) GT : supérieur (Greater Than), LT : inférieur (Less Than)
AND Rd, Rn, <operande>	ET logique bit à bit entre le registre n et <operande>, place le résultat dans le registre d.
ORR Rd, Rn, <operande>	OU logique bit à bit entre le registre n et <operande>, place le résultat dans le registre d.
EOR Rd, Rn, <operande>	OU Exclusif bit à bit entre le registre n et <operande>, place le résultat dans le registre d.
MVN Rd, <operande>	NON logique de la valeur <operande>, place le résultat dans le registre d.
LSL Rd, Rn, <operande>	(Logical Shift Left) : décale les bits du registre n vers la gauche en ajoutant un nombre de 0 à droite égal à <operande>, place le résultat dans le registre d.
LSR Rd, Rn, <operande>	(Logical Shift Right) : décale les bits du registre n vers la droite en ajoutant un nombre de 0 à gauche égal à <operande>, place le résultat dans le registre d.
INP Rd, 2	Lit un nombre depuis une Entrée et enregisttre sa valeur dans le registre d. (INPut)
OUT Rd, <device>	Écrit la valeur du registre d dans une sortie. (OUTput). La valeur de <device> indique le type de sortie : 4 : nombre signé 5 : nombre non signé 6 : hexadécimal 7 : caractère ASCII
HALT	Arrête l'exécution du programme

Exemple de LSL : `LSL R2, R1, 3`. Si R1 vaut `01110111`, R2 devient `10111000`.
Exemple de LSR : `LSR R2, R1, 3`. Si R1 vaut `01110111`, R2 devient `00001110`.