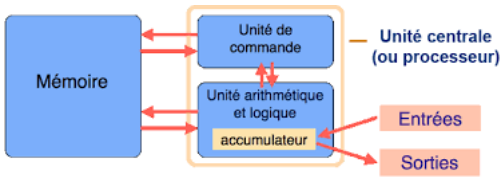
3) Architecture de Von Neumann

Dans le cours précédent, on a compris qu'un ordinateur est composé de 3 grands constituants :
- le processeur
- les mémoires
- les entrées-sorties (I/O= input/output)

Ces 3 constituants peuvent s'organiser de différentes façon dans un ordinateur.

> Dans la configuration de Von Neumann, la mémoire de l'ordinateur, dans laquelle sont stockées les données, doit également servir à stocker les programmes.

Voici un schéma de l'organisation de type Von Neumann d'un ordinateur :



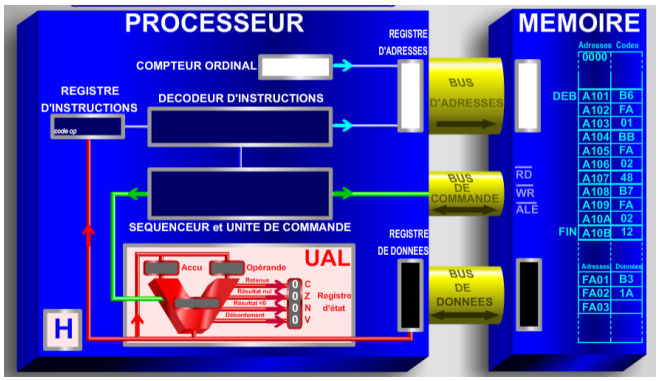
On retrouve bien nos 3 parties (mémoire, processeur et entrées-sorties). Ces composants communiquent grâce à des bus (flèches rouges).

> Le processeur (CPU in English!) est constitué :
> - d'une unité de commande (aussi appelé unité de contrôle)
> - d'une unité ariméthique et logique (UAL)

4) Fonctionnement du processeur (voir aussi : " TP Von Neumann ")

Revoir la vidéo de l'animation : https://www.youtube.com/watch?v=RlSP6WACI9k&t=68s
Site pour l'animation (activer flash) :http://public.iutenligne.net/informatique/informatique-industrielle/jacquemin/NAS_MODULE5/applications.html

Pour expliquer le fonctionnement d'un processeur, on va utiliser l'image de l'animation flash du TP. Sur celle-ci ne sont pas représentées les entrées-sorties.



Cette représentation correspond bien à une organisation " Von Neumann " puisque le programme (colonne " Codes ") est bien stocké dans la mémoire avec les données (2 données ici : B3 et 1A).

a) rôle de l'unité de commande (ou control unit)

> Dans le processeur, **l'unité de commande** se charge de :
> - mémoriser l'adresse de la prochaine instruction à exécuter (compteur ordinal)
> - décoder les instructions (registre d'instruction)
> - gérer les mouvements de données entre la mémoire et l'UAL (Lire (RD) ou écrire (WR) ou autoriser (ALE))

Sur cet exemple, les instructions à décoder sont, dans l'ordre : B6, BB, 48 et B7 qui signifient respectivement : " lis la donnée à l'adresse indiquée ", " additionne dans l'accumulateur ", " donne le complément à 1 " (vous vous souvenez?), " écris la valeur de l'accumulateur à l'adresse indiquée ".
Remarque : l'accumulateur est un registre de l'UAL (voir ci-dessous)

b) Rôle de l'unité arithmétique et logique (UAL)

L'unité arithmétique et logique (UAL) qui se trouve dans le processeur contient des circuits logiques et des mémoires très rapides appelés registres. Ce sont dans les registres que sont placées les données (appelées opérandes). C'est sur ces données que l'UAL va effectuer des opérations :

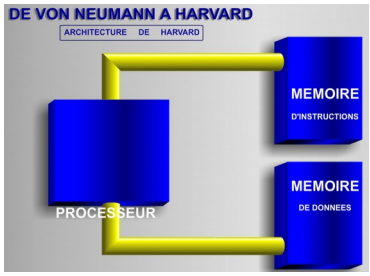
- mathématiques : additions, soustractions,…
- de comparaisons (égalité, inférieur, supérieur)
- sur les bits (compléments, décalages, rotations)
- de déplacements mémoire (copie de ou vers la mémoire)

5) Les limites du modèle de Von Neumann

Ce modèle impose un va-et-vient constant des données (data ou codes) entre le processeur et la mémoire. Le problème est que le temps d'accés aux données en mémoire est grand par rapport au temps d'exécution des opérations par le processeur. Autrement dit, avec l'organisation " Von Neumann ", le processeur perd du temps à attendre les données.

Quelles solutions pour éviter cette perte de temps ?

- faire appel à des **mémoires caches** (voir cours précédent) placées entre le CPU et la mémoire principale. Elles mémorisent des données souvent utilisées. Elles sont moins rapides que les registres mais plus rapides que la mémoire principale.
- utiliser des **architectures parallèles** : par exemple une architecture dotée de plusieurs CPU qui exécutent chacun un programme, de manière indépendante, sur des données différentes.



Dans l'architecture de Harvard, les mémoires d'instructions sont séparées des mémoires de données

III Langage machine et assembleur

Dans l'animation précédente, les instructions (comme B3, BB), les données et les adresses étaient représentées dans le système de numération hexadécimale. Cette représentation est utilisée pour cette animation parce qu'elle est compacte et lisible mais ce n'est pas ce que " lit " l'ordinateur.

Un ordinateur ne " lit " que des informations codées en binaire : c'est le **langage machine**. Toutes les instructions et données d'un programme chargé dans le processeur sont des suites de 1 et de 0. Par exemple, pour le processeur 8088 (processeur des années 80 qui a équipé l'IBM PC), l'instruction :

IBM PC

" écrit la valeur 18 dans le 1er registre " s'écrit en langage machine :" 1011 0 000 00010010 " : 1011 correspond à " déplacer " (MOV) , 0 à " écrire ", 000 à " premier registre " et 00010010 à " 18 "

Cette suite de 0 et de 1 est difficilement lisible pour un être humain. Pour le rendre lisible, on a fait correspondre à chaque code binaire une petite instruction comme MOV pour l'exemple précédent. Un registre sera représenté par AX par exemple. L'ensemble de ces instructions constitue **le langage assembleur.**

Le langage assembleur est une représentation lisible du langage machine. Il dépend du processeur.

En TP, vous allez vous initier à l'assembleur avec une simulation (AMIL)