Crédit du cours : JP Giard (Lycée Assomption) ,
David Roche (https://pixees.fr/informatiquelycee/n_site/nsi_prem_von_neu.html)

# Architecture machine, modèle de Non Neumann

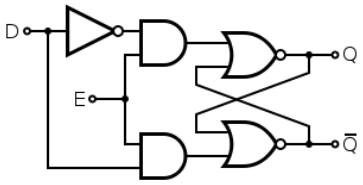
Une chose est très importante à bien comprendre : à la base nous avons le transistor, une combinaison de transistors (sous forme de circuit intégré) permet d'obtenir des circuits logiques, la combinaison de circuits logiques permet d'obtenir des circuits plus complexes (exemple : l'additionneur), et ainsi de suite...

Au sommet de cet édifice (on pourrait parler de poupée russe), nous allons trouver la mémoire vive (RAM) et le microprocesseur (CPU).

## La mémoire vive RAM (Random Acess Memory)

La mémoire vive permet de stocker des données et des programmes. Comme nous l'avons vu, l'ordinateur utilise uniquement 2 états, la mémoire va donc stocker les données sous forme de bits (0 ou 1), mais encore une fois, il ne faut pas s'imaginer que la mémoire est pleine de "petit 0" et de "petit 1", ce sont des "états électriques" qui sont stockés dans cette mémoire.

Ce sont des "états électriques" qui sont stockés 8 par 8. Ci-dessous le schéma d'un dispositif électronique de type « bascule » utile au stockage d'un seul bit




On peut se représenter la mémoire comme une série de cellules, chaque cellule étant capable de stocker 1 octet. Chacune de ces cellules possède une adresse. Les opérations sur la mémoire sont de 2 types : lecture / écriture. Une opération de lecture consiste à aller lire l'octet situé à l'adresse mémoire XXXXX (ces adresses mémoire étant bien évidemment codées en binaire) et une opération d'écriture consiste à écrire un octet donné à l'adresse mémoire YYYYY.

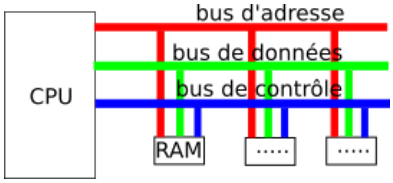
## Le microprocesseur CPU (Central Processing Unit)

Le microprocesseur est le "coeur" d'un ordinateur : les instructions sont exécutées au niveau du CPU. Il est schématiquement constitué de 3 parties :

- les **registres** permettent de mémoriser de l'information (donnée ou instruction) au sein même du CPU. Leur nombre et leur taille sont variables en fonction du type de microprocesseur. Dans la suite on nommera ces registres R1, R2, R3...
- **L'unité arithmétique et logique** (UAL ou ALU en anglais) est chargée de l'exécution de tous les calculs que peut réaliser le microprocesseur. Nous allons retrouver dans cette UAL des circuits comme l'additionneur (voir plus haut)
- **L'unité de commande** permet d'exécuter les instructions (les programmes)

## Le bus

les données doivent circuler entre les différentes parties d'un ordinateur, notamment entre la mémoire vive et le CPU. Le système permettant cette circulation est appelé bus. Il existe, sans entrer dans les détails, 3 grands types de bus :




- Le bus d'adresse permet de faire circuler des adresses (par exemple l'adresse d'une donnée à aller chercher en mémoire)
- Le bus de données permet de faire circuler des données

Crédit du cours : JP Giard (Lycée Assomption) ,
David Roche (https://pixees.fr/informatiquelycee/n_site/nsi_prem_von_neu.html)

- Le bus de contrôle permet de spécifier le type d'action (exemples : écriture d'une donnée en mémoire, lecture d'une donnée en mémoire).

# Initiation à l'assembleur

Revenons sur ces instructions aussi appelées "instructions machines" exécutées par l'unité de commande. Comme vous le savez déjà, un ordinateur exécute des programmes qui sont des suites d'instructions. Le CPU est incapable d'exécuter directement des programmes écrits, par exemple, en Python.
En effet, comme tous les autres constituants d'un ordinateur, le CPU gère uniquement 2 états (toujours symbolisés par un "1" et un "0"), les instructions exécutées au niveau du CPU sont donc codées en binaire. L'ensemble des instructions exécutables directement par le microprocesseur constitue ce que l'on appelle le "langage machine".

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 "00100110" donne l'ordre au CPU d'effectuer une multiplication.
- le champ "opérandes" indique la nature des données sur lesquelles l'opération désignée par le "code opération" doit être effectuée.

| champ code opération | champ opérandes |
|---|---|

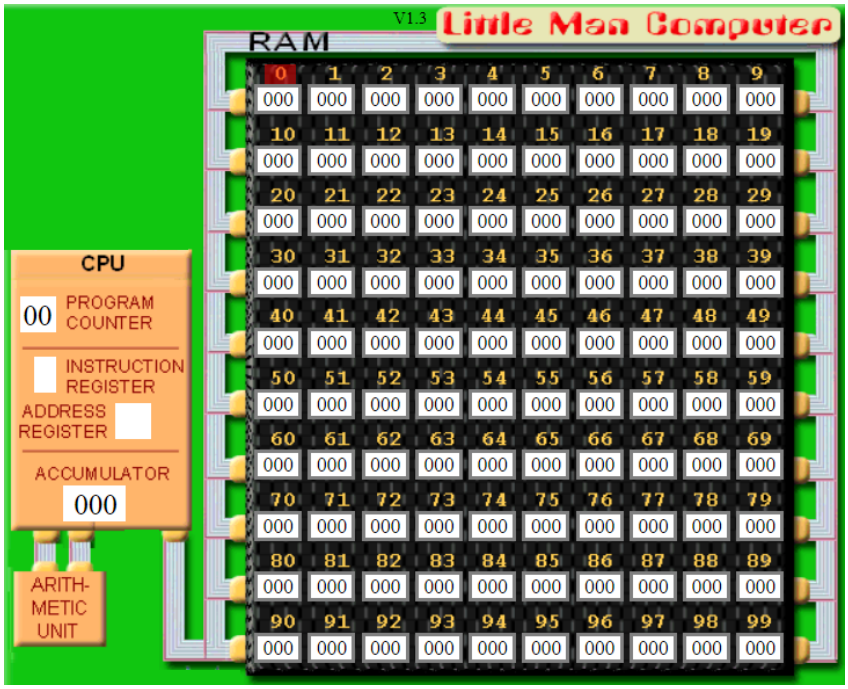
## Little Man Computer

https://fr.wikipedia.org/wiki/Little_man_computer

Sur le modèle **très simplifié** ci-dessous on retrouve

En noir et blanc à droite, la **RAM** constituée ici de 100 cellules dont les adresses vont de 0 à 99 (en binaire dans une vraie machine) et dont les contenus sont tous à 0 (pour l'instant)

Couleur saumon à gauche, le **microprocesseur**. Dans cet exemple il contient :
* 3 registres : registre d'instruction, registre d'adresse, accumulateur
* son unité arithmétique et logique (tout en bas)
* son unité de commande (program counter en haut)




Retrouvez le simulateur **Little Man Computer** (LMC) à la page http://www.peterhigginson.co.uk/LMC/. En plus de l'image ci-dessus vous observez des périphériques :

- **Input** relié au microprocesseur, considérez-le comme un clavier
- **Output** relié au microprocesseur, considérez-le comme un écran

Crédit du cours : JP Giard (Lycée Assomption) ,
David Roche (https://pixees.fr/informatiquelycee/n_site/nsi_prem_von_neu.html)

- **Assembly Language Code**, large zone en blanc à gauche qui va vous permettre de programmer LMC

A faire vous –même :

A gauche de l'écran (sous le mot Assembly) saisissez sur deux lignes les instructions suivantes:

INP
STA 20

Observez bien les premières adresses de RAM (0,1,2,3 …) et cliquez sur **submit** en dessous de vos instructions.

Que s'est-il passé au niveau de la RAM ? . . .**Le registre 0 a pris la valeur 901 et le registre 1 a pris la valeur 320**

Cliquez maintenant sur RUN (bouton gris en bas à gauche de l'écran)

Des ronds rouges et bleus commencent à se déplacer et le petit androïd bleu sous le CPU vous explique ce qui se passe (en anglais). Il vous invite bientôt à saisir un nombre dans la case INPUT, faites le et observez la suite jusqu'à ce que plus rien ne se passe, « Program HALTED » vous dit-il !

Pour mieux comprendre, proposez un autre programme à LMC en remplaçant simplement STA 20 par STA 27 puis rentrez une nouvelle valeur au moment du INPUT.

Résumez en 2 lignes ce que ces programmes réalisent :

**La RAM charge en mémoire la valeur donnée en INPUT dans l'emplacement mémoire 20 ou 27.**

Ci-dessous le **jeu d'instructions** de Little Man Computer (11 instructions). Créé dans un seul but d'apprentissage, il est beaucoup plus simple et moins complet que le jeu d'instructions actuel des processeurs X86 (PC et MAC) qui comprend plus de 1000 instructions.

D'autres architectures existent comme RISC ou CISC. L'architecture ARM qui domine dans le domaine de la téléphonie et des tablettes et de type RISC

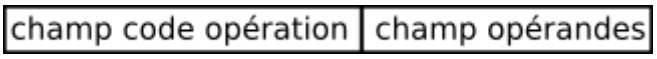
| Mnemonic | Name | Description | Op Code |
|---|---|---|---|
| INP | INPUT | Retrieve user input and stores it in the accumulator. | 901 |
| OUT | OUTPUT | Output the value stored in the accumulator. | 902 |
| LDA | LOAD | Load the Accumulator with the contents of the memory address given. | 5xx |
| STA | STORE | Store the value in the Accumulator in the memory address given. | 3xx |
| ADD | ADD | Add the contents of the memory address to the Accumulator | 1xx |
| SUB | SUBTRACT | Subtract the contents of the memory address from the Accumulator | 2xx |
| BRP | BRANCH IF POSITIVE | Branch/Jump to the address given if the Accumulator is zero or positive. | 8xx |

Crédit du cours : JP Giard (Lycée Assomption) ,
David Roche (https://pixees.fr/informatiquelycee/n_site/nsi_prem_von_neu.html)

| | | | |
|---|---|---|---|
| BRZ | BRANCH IF ZERO | Branch/Jump to the address given if the Accumulator is zero. | 7xx |
| BRA | BRANCH ALWAYS | Branch/Jump to the address given. | 6xx |
| HLT | HALT | Stop the code | 000 |
| DAT | DATA LOCATION | Used to associate a label to a free memory address. An optional value can also be used to be stored at the memory address. | |

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 "00100110" donne l'ordre au CPU d'effectuer une multiplication.
- le champ "opérandes" indique la nature des données sur lesquelles l'opération désignée par le "code opération" doit être effectuée.




Chaque instruction machine est visible dans LMC en base 10 mais évidemment en base 2 « pour de vrai »

Pour la personne qui doit coder dans ce langage machine il serait fastidieux (voire impossible) de saisir sans erreur des suites de 0 et de 1, il utilise donc des **mnémoniques** pour préparer son programme (**ST** signifie **store** pour écrire dans une mémoire, **LD** signifie **Load** pour lire une mémoire, **ADD** signifie **ajouter**, …). La traduction d'un programme rédigé en mnémoniques vers une suite d'instructions en binaire est assurée par un petit logiciel appelé **assembleur**.

Par extension, on dit que l'on code en **assembleur** quand on code en mnémoniques de microprocesseur

A l'aide du jeu d'instructions, expliquez avec précision les informations que LMC a introduites dans la RAM quand vous avez enregistré votre programme (bouton submit)

**INP → Insère dans le registre 0, le codeop 901**
**STA 20 -> Store the value in the Accumulator in the memory address given, ici 20**

| Address | Instruction | | What it does: |
|---|---|---|---|
| | Mnemonic | Machine Code | |
| 00 | INP | 901 | Input a number and put it in the calculator |
| 01 | STA 99 | 399 | Store the number in the calculator in memory slot 99 |
| 02 | INP | 901 | Input a number and put it in the calculator |
| 03 | ADD 99 | 199 | Add the number in memory slot 99 to the number in the calculator |
| 04 | OUT | 902 | Output the number which is now in the calculator |
| 05 | HLT | 000 | Halt |

Crédit du cours : JP Giard (Lycée Assomption) ,
David Roche (https://pixees.fr/informatiquelycee/n_site/nsi_prem_von_neu.html)

LMC vous montre la circulation des informations sous forme de petits ronds rouges et bleus car elles sont de deux natures bien distinctes : Expliquer
**En bleu : les données entrantes dans la CPU**
**En rouge : les données sortantes de la CPU**

Dans le microprocesseur, deux circuits de connexion bien distincts véhiculent ces deux type d'information, on les appelle des bus. C'est grâce à ces deux bus que vous pouvez voir des informations « rouges » et « bleues » circuler **en même temps**. Ici cela vous semble lent mais rassurez-vous, un signal électrique se propage dans une puce électronique à une vitesse proche de celle de la lumière et les distances à parcourir sont très faibles.

A vous de jouer (challenges) : recopier votre code dans la colonne de droite

| | |
|---|---|
| Charger (INPUT) et réafficher une valeur | INP<br>STA 20<br>OUT |
| Charger 2 valeurs, calculer et afficher la somme des 2 | INP<br>STA 20<br>INP<br>STA 30<br>LDA 20<br>ADD 30<br>OUT<br>HLT<br>20 DAT<br>30 DAT |
| Charger 3 valeurs, calculer et afficher la somme des 3 | … |
| Charger 2 valeurs et afficher la différence ($1^{er}$ – $2^{ème}$) | INP<br>STA A<br>INP<br>STA B<br>LDA A<br>SUB B<br>OUT<br>HLT<br>A DAT<br>B DAT |
| Charger 2 valeurs et afficher la plus grande des deux | INP<br>STA A<br>INP<br>STA B<br>LDA A<br>SUB B<br>BRP<br>isPositive<br>LDA B<br>OUT<br>HLT<br>isPositive<br>LDA A<br>HLT |

Crédit du cours : JP Giard (Lycée Assomption) ,
David Roche (https://pixees.fr/informatiquelycee/n_site/nsi_prem_von_neu.html)

| | |
|---|---|
| | A DAT<br>B DAT |
| Charger 3 valeurs puis les afficher dans l'ordre croissant | … |

Observez vos programmes tourner et indiquez.

- Ce que l'on voit en permanence dans la case *Program Counter* (*PC*)
  **Le Compteur de Programme contient l'adresse de la prochaine instruction que le Petit Homme va effectuer**
- Pourquoi la case *Instruction Register* n'a qu'un seul chiffre
  **Le premier chiffre d'une instruction numérique représente la commande à effectuer et les deux derniers chiffres représentent l'adresse mémoire de la boîte aux lettres concernée par cette commande.**
- Quand l'***ALU*** est-elle sollicitée
  **Exécution de tous les calculs de microprocesseur**

**Attention :**
Comme indiqué par l'architecture de von Neumann, la mémoire contient à la fois les instructions et les données. Il est par conséquent nécessaire de faire attention à bien empêcher le Compteur de Programme d'atteindre une adresse mémoire contenant des données autrement le Petit Homme tentera de la traiter comme une instruction.

Crédit du cours : JP Giard (Lycée Assomption) ,
David Roche (https://pixees.fr/informatiquelycee/n_site/nsi_prem_von_neu.html)

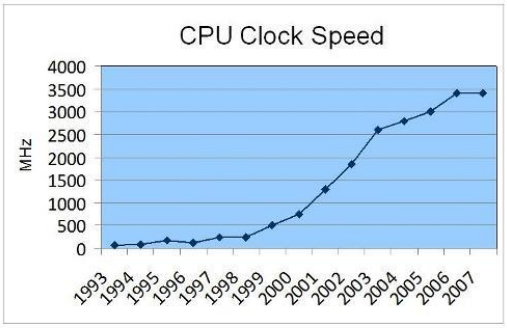
A lire :

https://interstices.info/le-modele-darchitecture-de-von-neumann/

# Architecture de von Neumann

Comme vous avez pu le constater dans les exemples ci-dessus, les données et les instructions sont stockées en mémoire vive, les données et les instructions se partagent la mémoire vive (il n'y a pas une mémoire pour les instructions et une mémoire différente pour les données). C'est John von Neumann (mathématicien et physicien américano-hongrois 1903-1957) qui a eu l'idée en 1945 d'utiliser une structure de stockage unique pour les données et les instructions, voilà pourquoi on parle d'architecture de von Neumann. Encore aujourd'hui, tous les ordinateurs fonctionnent sur ce principe défini par von Neumann. À noter que John von Neumann était un véritable génie "touche à tout" puisqu'il a laissé son nom dans l'histoire de la mécanique quantique, dans l'histoire de la théorie des ensembles...et comme nous venons de le voir, dans l'histoire de l'informatique. Il a aussi participé à l'élaboration de la bombe atomique américaine lors de la 2e guerre mondiale (projet Manhattan).

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 (en fait, c'est un peu plus compliqué que cela, mais nous nous contenterons de cette explication).




Comme vous pouvez le remarquer sur le graphique ci-dessus, à partir de 2006 environ, la fréquence d'horloge a cessé d'augmenter, pourquoi ? À cause d'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 d'arrêter la course à l'augmentation de la fréquence d'horloge, ils ont décidé d'adopter une nouvelle tactique.

Il n'est plus vraiment possible d'augmenter les performances en augmentant la fréquence d'horloge des CPU, et bien augmentons le nombre de coeurs présent sur un CPU ! Mais qu'est qu'un coeur dans un microprocesseur ? Dans un microprocesseur, un coeur est principalement composé : d'une UAL, de registres (R0, R1...) et d'une unité de commande, un coeur 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 coeurs, alors qu'auparavant on trouvait un seul coeur dans un CPU. Cette technologie a été implémentée dans les ordinateurs grand public à partir de 2006. Aujourd'hui (en 2019) on trouve sur le marché des CPU possédant jusqu'à 18 coeurs ! Même les smartphones possèdent des microprocesseurs multicoeurs : le Snapdragon 845 possède 8 coeurs.

On pourrait se dire que l'augmentation du nombre de coeurs entraîne obligatoirement une augmentation des performances du CPU, en faite, c'est plus que complexe que cela : pour une application qui n'aura pas été conçue pour fonctionner avec un microprocesseur multicoeur, le gain de performance sera très faible, voir même nul. En effet, la conception d'applications capables de tirer profit d'un CPU multicoeur demande la mise en place de certaines techniques de programmation (techniques de programmation qui ne seront pas abordées ici). Il faut aussi avoir conscience que les différents coeurs d'un CPU doivent se "partager" l'accès à la mémoire vive : quand un coeur travaille sur une certaine zone de la RAM, cette même zone n'est pas accessible aux autres coeurs, ce qui, bien

évidemment va brider les performances. De plus, on trouve à l'intérieur des microprocesseurs de la mémoire "ultrarapide" appelée mémoire cache (il ne faut pas confondre mémoire cache et registres). Le CPU peut stocker certaines données dans cette mémoire cache afin de pouvoir y accéder très rapidement dans le futur, en effet, l'accès à la mémoire cache est beaucoup plus rapide que l'accès à la RAM. La mémoire cache ayant un coup assez important, la quantité présente au sein d'un CPU est assez limitée, les différents coeurs vont donc devoir se partager cette mémoire cache, ce qui peut aussi provoquer des ralentissements (en faite il existe plusieurs types de mémoire cache appelés L1, L2 et L3, chaque coeur possède son propre cache L1, alors que les caches L2 et L3 sont partagés par les différents coeurs).