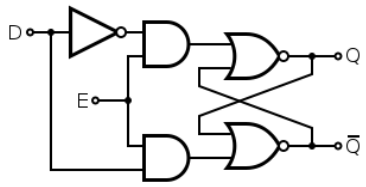
Architecture machine, modèle de Non Neumann

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)

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



La mémoire gère donc des octets rangés dans des « cellules ». Chacune de ces cellules possède une adresse. Les opérations sur la mémoire sont de 2 types : lecture / écriture.

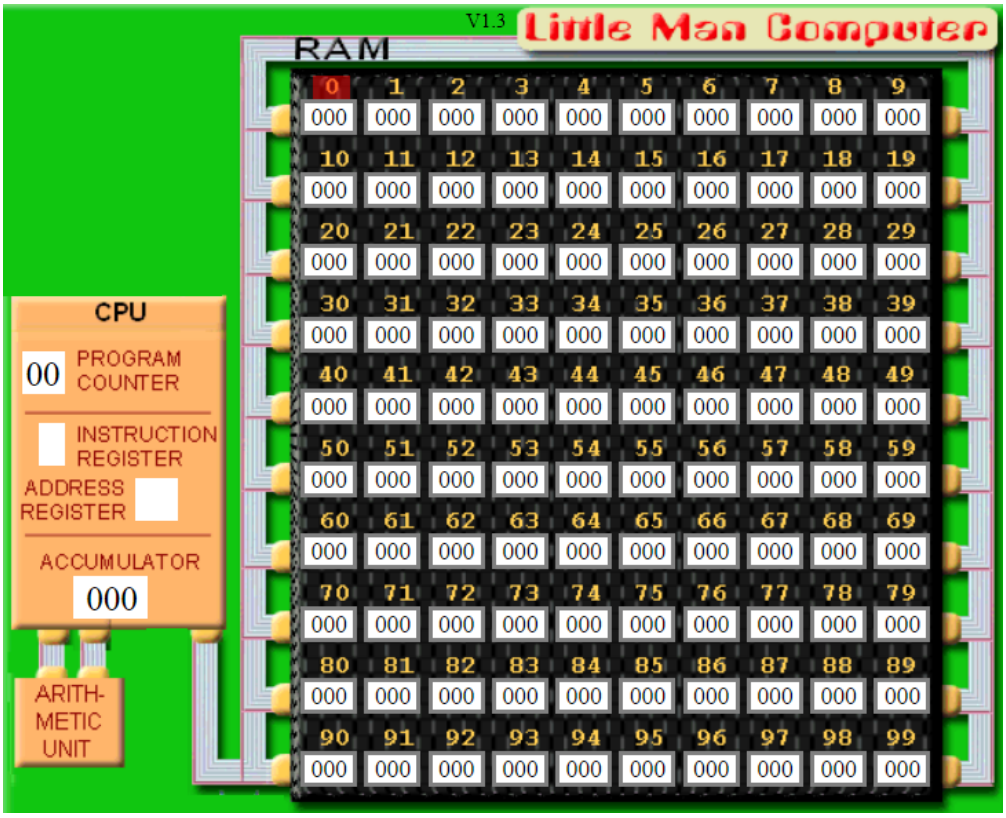
Le microprocesseur CPU (Central Processing Unit)

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)

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
- **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 ? . . . . . .

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 :

. . . . . .

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
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.

champ code opération	champ opérandes

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)

. . . . . .

LMC vous montre la circulation des informations sous forme de petits ronds rouges et bleus car elles sont de deux natures bien distinctes : Expliquer. . . . . .
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 **<u>en même temps</u>**. 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	…
Charger 2 valeurs, calculer et afficher la somme des 2	…
Charger 3 valeurs, calculer et afficher la somme des 3	…
Charger 2 valeurs et afficher la différence ($1^{er} – 2^{ème}$)	…
Charger 2 valeurs et afficher la plus grande des deux	…
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*) ? ? ? ? ? ? ? ?
- Pourquoi la case *Instruction Register* n'a qu'un seul chiffre ? ? ? ? ?
- Quand l'***ALU*** est-elle sollicitée ? ? ? ? ? ? ? ? ?