❄ Chapitre 1 ❄

Architecture von Neumann

I. Un peu d'histoire

En 1946 le premier ordinateur entièrement électronique est enfin opérationnel. Il s'agit de **l'ENIAC** (Electronic Numerical Integrator And Computer). Il pouvait en principe être reprogrammé pour résoudre tous les problèmes calculatoires (il est dit Turing-complet).

Il s'agit d'une machine imposante (17468 tubes à vide, 7200 diodes à cristal, 1500 relais, 70000 résistances, 10000 condensateurs et environ 5 millions de soudures faites à la main) occupant une surface de 167 m^2.

6 femmes ont été les premières personnes à programmer l'ENIAC pour le calcul de tirs balistiques. Pour programmer un calcul il fallait faire un plan des connexions nécessaires, puis procéder au câblage physique.
Le travail était long et les erreurs dures à détecter.

II. John von Neumann

Le mathématicien/physicien/... américano -hongrois John von Neumann (1903 - 1957), qui a participé à l'élaboration du projet ENIAC, propose alors un nouveau modèle pour simplifier le fonctionnement d'un ordinateur. Il décide de séparer physiquement la partie contrôle/calculs de la partie mémoire, tout en affirmant qu'une instruction n'est qu'une donnée de mémoire comme une autre. C'est le modèle d'architecture de von Neumann, qui est encore utilisé de nos jours!

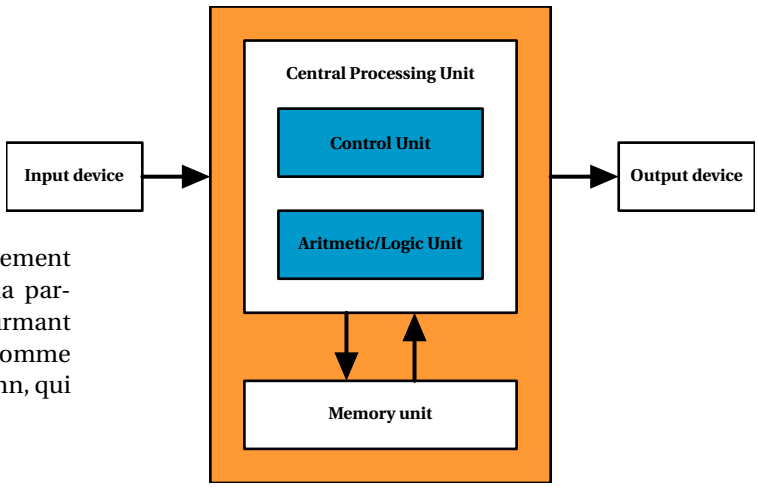

Schéma de l'architecture de von Neumann

III. Les différentes parties

1. La mémoire

Elle permet de stocker des informations sous forme de bits. La mémoire ne peut pas stocker un seul bit à la fois, elle stocke des mots de 8 bits (ou 16/32/64 bits) appelés un octet. Le processeur accède à différents mots de la mémoire grâce à leur adresse.

2. Le processeur (CPU)

C'est le cœur de tout ordinateur. Il est lui même composé de 2 parties indépendantes, d'une part l'Unité de Contrôle (UC) et d'autre part l'Unité Arithmétique et Logique (UAL ou ALU en anglais).

Définition 1: *Unité de contrôle*

Dans un système logique, en particulier dans un processeur, l'unité de contrôle (de commande) ou séquenceur commande et contrôle le fonctionnement du système, notamment du chemin de données. Une unité de contrôle est un circuit logique séquentiel qui réalise un automate fini, qui génère des signaux de contrôle pour piloter les éléments du chemin de données

Définition 2: *Unité Arithmétiques et Logique*

L'unité arithmétique et logique est l'organe de l'ordinateur chargé d'effectuer les calculs. Le plus souvent, l'UAL est incluse dans l'unité centrale de traitement ou le microprocesseur. Les UAL élémentaires calculent sur des nombres entiers, et peuvent effectuer les opérations communes, que l'on peut séparer en quatre groupes : les opérations arithmétiques, les opérations logiques, les comparaisons et éventuellement des décalages et rotations.

3. Les dispositifs d'entrées et de sorties

Ils permettent d'échanger des informations avec le processeur.
Quelques exemples de dispositifs d'entrées :

.....

.....

.....

Quelques exemples de dispositifs de sorties :

.....

.....

.....

4. Le bus

C'est la partie physique qui permet aux autres composants de communiquer entre-eux.

IV. Le processeur et ses instructions

Les UC et UAL d'un processeur ne connaissent qu'un nombre réduit d'instructions. Celles-ci dépendent du modèle et de la marque du processeur mais la plupart des processeurs récents ont une partie de leur jeu d'instructions commune.

Afin de pouvoir travailler avec des informations, un processeur possède un petit espace mémoire propre pour stocker des données. Il y a généralement au moins 8 registres permettant de stocker un mot de 8 bits chacun (voir 32 ou même 64 bits pour les plus récents).

Il n'est pas demandé en $1^{ère}$ NSI de connaître par cœur le langage assembleur (qui permet de donner presque directement des instructions au processeur) mais vous devez en comprendre le fonctionnement.

Voici quelques instructions classiques mais simplifiées pour un microprocesseur ARM moderne :

Instruction	Opérandes (exemples)	Commentaires
LDR	R0, 102	Charge le mot à l'adresse 102 en mémoire dans le registre R0.
STR	R1, 63	Écrit le contenu du registre R1 dans la mémoire à l'adresse 63.
MOV	R0, R2	Copie la valeur de R2 dans R0. MOV ne peut pas accéder à la mémoire.
ADD	R0, R3, # 13	Ajoute le contenu de R3 et le nombre 13 (opérande immédiat grâce au #, sinon c'est une adresse mémoire) et écrit le résultat dans le registre R0.
SUB	R1, R2, R0	Soustrait le contenu de R0 à R2 et écrit le résultat dans R1.
B	18	La prochaine instruction se trouve à l'adresse mémoire 18 (saut).
CMP	R0,R1	**C**o**MP**are les valeurs des registres R0 et R1. Commande suivie d'une instruction de saut conditionnel.
BEQ	13	En cas d'égalité (**EQ**ual) du CMP, l'instruction suivante est à l'adresse 13.
BNE	13	En cas d'inégalité (**N**ot **E**qual) du CMP, ...
BGT	13	Si le premier opérande du CMP est supérieur au second (**G**reater **T**han), ...
BLT	13	Si le premier opérande du CMP est inférieur au second (**L**ess **T**han), ...
HALT		Fin de l'exécution. Important sinon le programme continue !

Vous trouverez ici un simulateur pour processeur ARM.

Exemple 1:

1. Que font les instructions suivantes?

MOV R1, # 33	.....
ADD R0, R1, # 12	.....
STR R0, 100	.....

2. Que fait le programme suivant si l'adresse mémoire 102 contient le nombre 42?

```
LDR R0, 102
MOV R1, R0
ADD R0, R0, R1
STR R0, 102
```

.....
.....
.....
.....
.....

Remarque :

L'assembleur permet aussi d'utiliser des étiquettes pour les adresses mémoires (ce qui est pratique pour les sauts) et se charge de convertir l'étiquette par l'adresse mémoire correspondante.

Exemple 2:

```
        LDR R0, 30
        LDR R1, 31
        CMP R0, R1
        BGT saut
        SUB R0, R1, R0
        STR R0, 32
        HALT
saut :
        SUB R0, R0, R1
        STR R0, 32
        HALT
```

1. Que contient l'adresse mémoire 32 à la fin du programme si :

Adresse mémoire 30 au début	Adresse mémoire 31	Adresse mémoire 32 à la fin
24	18	
30	45	
12	12	

2. Que fait donc le programme précédent?
3. Écrire un programme en Python faisant la même chose (on considérera que l'adresse mémoire 30 sera une variable x et l'adresse 31 une variable y).

Le défi du chapitre :

Êtes-vous capable d'écrire un programme en assembleur utilisant au maximum 4 registres pour multiplier un nombre entier positif écrit à l'adresse mémoire 100 par un nombre entier positif écrit à l'adresse mémoire 101 et enregistrer le résultat à l'adresse mémoire 102?

Sources : Wikipédia : Architecture de von Neumann
Pixees : Ressources pour les sciences numériques