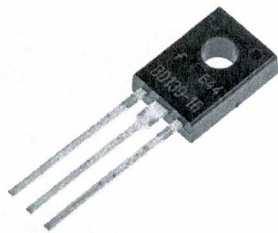


Chapitre 7 : Modèle d'architecture séquentiel & Langage Machine

1. Des bribes d'électronique : du transistor au CPU

1.1. Tube à vide, transistor et circuit intégré

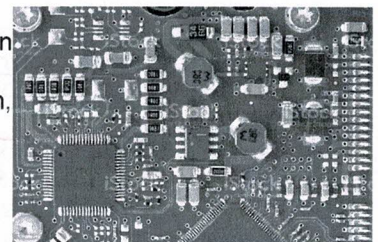


À la base de la plupart des composants d'un ordinateur, on retrouve le *transistor*. Ce composant électronique a été inventé fin 1947 par les Américains **John Bardeen**, **William Shockley** et **Walter Brattain**.

L'invention du *transistor* a été un immense progrès, mais les premiers ordinateurs sont antérieurs à cette invention. En effet, ces premiers ordinateurs, par exemple le Colossus qui date de 1943, étaient conçus à base de tubes électroniques (on parle aussi de tubes à vide) qui, bien que beaucoup plus gros et beaucoup moins fiables que les transistors, fonctionnent sur le même principe que ce dernier.

Dans un ordinateur, les transistors sont regroupés au sein de ce que l'on appelle des *circuits intégrés*.

Dans un circuit intégré, les transistors sont gravés sur des plaques de silicium, les connexions entre les millions de transistors qui composent un *circuit intégré* sont, elles aussi, gravées directement dans le silicium.



1.2. Circuits logiques

Le transistor est l'élément de base des circuits logiques. Un circuit logique permet de réaliser une opération booléenne. Il existe deux catégories de circuit logique :

- les circuits combinatoires (les états en sortie dépendent uniquement des états en entrée)
- les circuits séquentiels (les états en sortie dépendent des états en entrée ainsi que du temps et des états antérieurs)

Dans la suite nous nous intéresserons principalement aux circuits combinatoires.

Un exemple de circuits séquentiels est la machine de Turing : <https://interstices.info/comment-fonctionne-une-machine-de-turing/#:~:text=La%20machine%20imagin%C3%A9e%20par%20Turing,gauche%20ou%20vers%20la%20droite>.

1.3. La mémoire vive (RAM)

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 et écriture*.

Une opération de *Lecture* consiste à aller lire l'octet situé à l'adresse mémoire XXXXX et une opération d'*écriture* consiste à écrire un octet donné à l'adresse mémoire YYYYY.

On n'est ainsi pas obligé de passer par une cellule pour accéder à une autre (ce qui formerait une mémoire à accès séquentiel).

Cette faculté d'accéder directement à chaque cellule explique le nom **R A M**

(Random Access Memory)

2. Le microprocesseur (CPU)

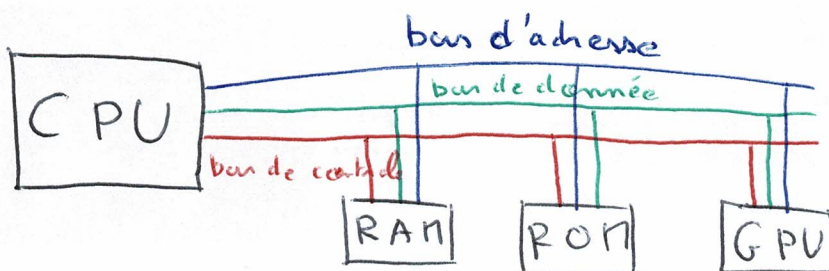
2.1. Organisation générale

Les instructions qui composent les programmes sont exécutées par le CPU (Central processing unit). Il est schématiquement constitué de 3 parties.

- L'unité arithmétique (UAL ou ALU) 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.
- L'unité de commande permet d'exécuter les instructions (les programmes).
- Une toute petite quantité de mémoire appelée les registres, qui permettent de mémoriser de l'information transitoirement pour opérer dessus ou avec. Leur nombre et leur taille et leur rôle sont variables en fonction du type de microprocesseur.

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 :

- bus d'adresse permet de faire circuler des adresses (par exemple l'adresse d'une donnée à aller chercher en mémoire).
- bus de données permet de faire circuler des données.
- bus de contrôle permet de spécifier le type d'action (exemples : écriture d'une donnée en mémoire, lecture d'une donnée en mémoire).



2.2. Fonctionnement

Un processeur donné est capable d'exécuter un certain nombre d'opérations de base, celles pour lesquelles il dispose d'un circuit électronique qui les réalise.

L'ensemble des instructions exécutables directement par le microprocesseur (instructions machines) constitue ce que l'on appelle le "langage machine" du processeur.

Chaque instruction machine correspond à une configuration électronique binaire composée principalement de 2 parties.

- Le champ "code opération" (opcode) qui indique au processeur le type de traitement à réaliser. Par exemple, sur un certain modèle de processeur, le code "00100110" donne l'ordre d'effectuer une multiplication.
- Le champ "opérandes" indique la nature des données sur lesquelles l'opération désignée par le "code opération" doit être effectuée.
Un opérande peut être de 3 natures différentes :
 - l'opérande est une valeur immédiate : l'opération est effectuée directement sur la valeur donnée dans l'opérande ;
 - l'opérande est un registre du CPU : l'opération est effectuée sur la valeur située dans un des registres (R0, R1, R2, ...) ; l'opérande indique de quel registre il s'agit ;
 - l'opérande est une donnée située en mémoire vive : l'opération est effectuée sur la valeur située en mémoire vive à l'adresse XXXXX. Cette adresse est indiquée dans l'opérande.

Le programme exécuté se trouve dans la RAM.

Un registre particulier du processeur, nommé IP (instruction pointer), contient l'adresse de la cellule RAM de la prochaine instruction à exécuter.

Un deuxième registre, IR (instruction register), joue un rôle important en raison de sa connexion physique au reste du processeur : placer dans ce registre la configuration électronique qui dénote une instruction provoque l'activation du circuit dédié à la réalisation de l'opération sous-jacente.

Le CPU a un fonctionnement cyclique :

- il copie dans le registre IR le contenu de la RAM à l'adresse pointée par IP ;
- il décode l'instruction contenue dans IR : ceci provoque l'activation du circuit électronique qui réalise l'opération visée ;
- il exécute l'instruction décodée ; ceci met aussi à jour la valeur de IP pour continuer dans le programme.

La capacité du processeur à exécuter tous les programmes s'explique par ce fonctionnement très souple et par le fait que le jeu d'instructions de base est suffisamment riche pour être universel (on dit que le langage machine est Turing-complet).

Dans le modèle de von Neumann, il y a une seule mémoire vive pour le programme et les données : c'est par sa copie dans le registre IR qu'une configuration électronique initialement présente en RAM joue le rôle d'une instruction. La même configuration pourrait être interprétée comme une donnée (entier, etc.) dans un autre contexte.

2.3. Exemples d'instructions CPU

Les opérations fondamentales sont de trois sortes.

- ➔ *Les instructions arithmétiques* (addition, soustraction, multiplication...) effectuent des calculs mathématiques, soit sur des entiers, soit sur des flottants. Par exemple, on peut avoir une instruction consistant à additionner la valeur contenue dans le registre R1 et le nombre 789 et ranger le résultat dans le registre R0.
- ➔ *Les instructions de transfert de données* permettent de transférer une donnée d'un registre du CPU vers la mémoire vive et vice versa. Par exemple, on peut avoir une instruction consistant à prendre la valeur située à l'adresse mémoire 487 et la placer dans la registre R2, ou encore prendre la valeur située dans le registre R1 et la placer à l'adresse mémoire 512.
- ➔ *Les instructions de rupture de séquence ou instructions de saut* permettent d'agir sur le registre IP qui contient l'adresse de la prochaine instruction à exécuter. Les instructions arithmétiques ou de transfert de données, outre leur effet spécifique, incrémentent la valeur de IP : de cette façon, l'exécution du programme avance séquentiellement. Les instructions de saut permettent de réaliser les autres structures de contrôle d'exécution, notamment l'alternative et la boucle.

3. Initiation à l'assembleur

Programmer en langage machine est extrêmement difficile (très longue suite de 0 et de 1), pour pallier cette difficulté, les informaticiens ont remplacé les codes binaires abscons par des symboles mnémoniques (plus facile à retenir qu'une suite de "1" et de "0"), cela donne l'assembleur. Par exemple un "ADD R1,R2,#125" sera équivalent à "11100010100000100001000001111101".

Le processeur est uniquement capable d'interpréter le langage machine, un programme appelé "assembleur" assure donc le passage de "ADD R1,R2,#125" à "11100010100000100001000001111101". Par extension, on dit que l'on programme en assembleur quand on écrit des programmes avec ces symboles mnémoniques à la place de suite de "0" et de "1".

On va tester le langage machine sur un assembleur :

<http://www.peterhigginson.co.uk/AQA/>

1 LDR R1,78

Place la valeur stockée à l'adresse mémoire 78 dans le registre R1 (par souci de simplification, nous continuons à utiliser des adresses mémoire codées en base 10)

1 STR R3,125

Place la valeur stockée dans le registre R3 en mémoire vive à l'adresse 125

1 ADD R1,R0,#128

Additionne le nombre 128 (une valeur immédiate est identifiée grâce au symbole #) et la valeur stockée dans le registre R0, place

le résultat dans le registre R1

1 ADD R0,R1,R2

Additionne la valeur stockée dans le registre R1 et la valeur stockée dans le registre R2, place le résultat dans le registre R0

1 SUB R1,R0,#128

Soustrait le nombre 128 de la valeur stockée dans le registre R0, place le résultat dans le registre R1

1 SUB R0,R1,R2

Soustrait la valeur stockée dans le registre R2 de la valeur stockée dans le registre R1, place le résultat dans le registre R0

1 MOV R1, #23

Place le nombre 23 dans le registre R1

1 MOV R0, R3

Place la valeur stockée dans le registre R3 dans le registre R0

1 B 45

Nous avons une structure de rupture de séquence, la prochaine instruction à exécuter se situe en mémoire vive à l'adresse 45

1 CMP R0, #23

Compare la valeur stockée dans le registre R0 et le nombre 23. Cette instruction CMP doit précéder une instruction de branchement

conditionnel BEQ, BNE, BGT, BLT (voir ci-dessous)

1 CMP R0, R1

Compare la valeur stockée dans le registre R0 et la valeur stockée dans le registre R1.

1 CMP R0, #23

2 BEQ 78

La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 est égale à 23

1 CMP R0, #23

2 BNE 78

La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 n'est pas égale à 23

1 CMP R0, #23

2 BGT 78

La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 est plus grand que 23

1 CMP R0, #23

2 BLT 78

La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 est plus petit que 23

1 HALT

Arrête l'exécution du programme