Architecture de von Neumann et ses alternatives. Description et état des lieux

BRUNO DARID

20 octobre 2019

PLAN

1 Repères historiques **1**
1.1 De l'objet conceptuel... 1
1.2 Au modèle d'architecture de von Neumann 2

2 Modèle d'architecture séquentielle (*dite de von Neumann*) **3**
2.1 Les composants essentiels 3
2.1.1 L'unité arithmétique et logique 3
2.1.2 L'unité de commande (*ou de contrôle*) 3
2.1.3 Les mémoires 4
2.1.4 Les entrées-sorties 4
2.2 Les bus 5
2.3 Les instructions 6
2.3.1 Exemple commenté n° 1 6
2.3.2 Exemple commenté n° 2 6
2.3.3 E1C5 Que fait cette séquence d'instructions ? 7
2.3.4 E2C5 Langage d'assemblage 7
2.4 E3C5 Execution d'une séquence d'instructions 7

3 Qu'en est-il aujourd'hui ? **8**

4 Alternative : architecture de Harvard **8**

1 Repères historiques

1.1 De l'objet conceptuel...

En 1936, Alan Turing - un des plus brillants mathématiciens du XX^e siècle - décrit une machine abstraite pour donner un support formel aux notions d'algorithme et de calcul : la fameuse « machine de Turing ». Il s'agit d'un automate imaginaire muni d'un programme (sous la forme d'une table de transition entre états) et pouvant lire et écrire des caractères sur un ruban de longueur illimitée (voir figure 1 page 2).

Le coup de génie de Turing est de démontrer l'existence d'une **Machine Universelle**. Si on fournit à cette dernière la table de transition d'une machine de Turing particulière, autrement

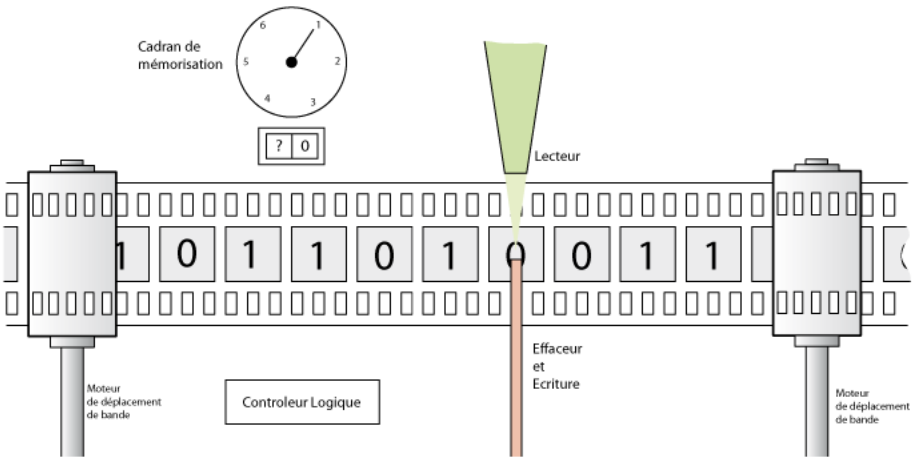

Fig. 1 – La machine de Turing

dit le programme d'un algorithme, **elle est capable de reproduire le fonctionnement de cette machine, donc d'exécuter le programme en question**!

1.2 Au modèle d'architecture de von Neumann

En 1945, John von Neumann (1903-1957), mathématicien américano-hongrois qui avait déjà rencontré Alan Turing et qui connaissait ses travaux, publia un rapport dans lequel il définit le modèle de l'ordinateur à « programme enregistré ». La mémoire de cet ordinateur **contient à la fois les programmes et les données, comme dans la machine de Turing universelle**!

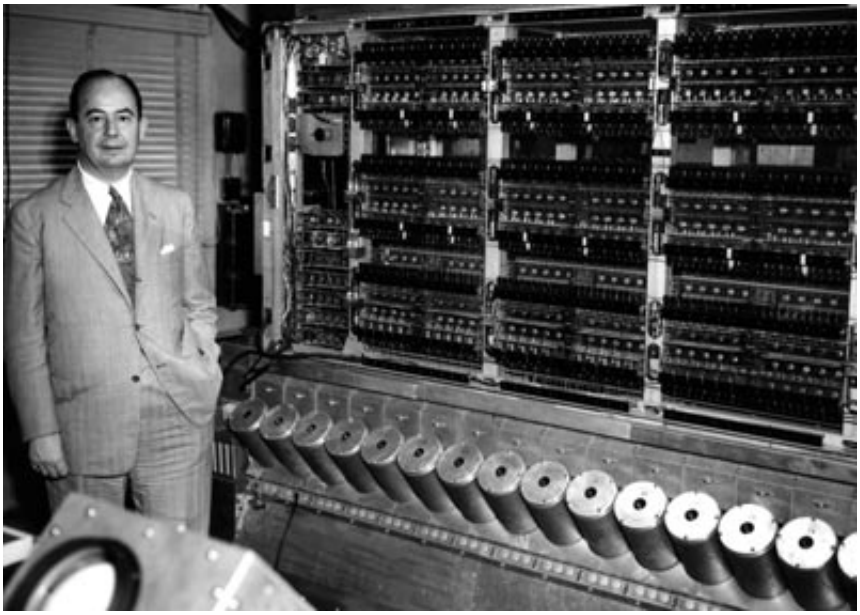

Fig. 2 – John von Neumann

2 Modèle d'architecture séquentielle (*dite de von Neumann*)

Un des grands points faibles du premier calculateur électronique l'ENIAC était le fait que les données étaient lues sur cartes perforées, mais le programme était représenté sur un support externe, sous la forme d'un panneau de connexion analogue à celui d'un standard téléphonique. L'idée fondamentale de John von Neumann a été de stocker les données **et** les instructions des programmes en mémoire centrale.

2.1 Les composants essentiels

En juin 1945 dans la première version du rapport sur la conception de l'EDVAC John von Neumann décrit un schéma d'architecture d'un calculateur organisé autour des éléments suivants :

— une unité centrale ou processeur (on dit aussi **CPU** pour **C**entral **P**rocessing **U**nit), formée d'une unité arithmétique et logique (*UAL*) et d'une unité de commande (*Control Unit*) ;

— la mémoire ;

— des unités d'entrées/sorties.

Ces éléments étant reliés entre eux par des bus.

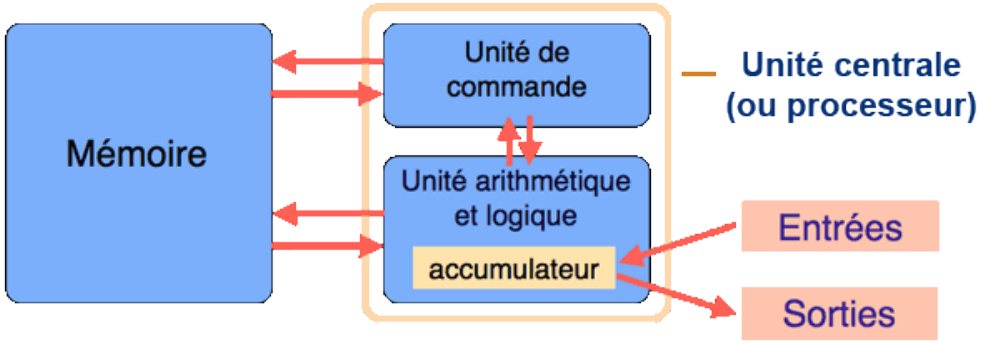

Fig. 3 – Modèle original

2.1.1 L'unité arithmétique et logique

Elle est chargée d'effectuer le traitements des opérations arithmétiques ou booléennes :

— addition, multiplication, soustraction, division
— ET, OU et NON logiques ;
— décalages de bits dans des registres

L'UAL est entourée généralement de **registres de données** (*mémoires rapides*) et d'un **accumulateur** qui accueille les opérandes des opérations ou le résultat.

2.1.2 L'unité de commande (*ou de contrôle*)

Elle est chargée de contrôler les échanges, gérer l'enchainement des instructions et les transferts entre les différents éléments.

On y trouve (*entre autres*) :

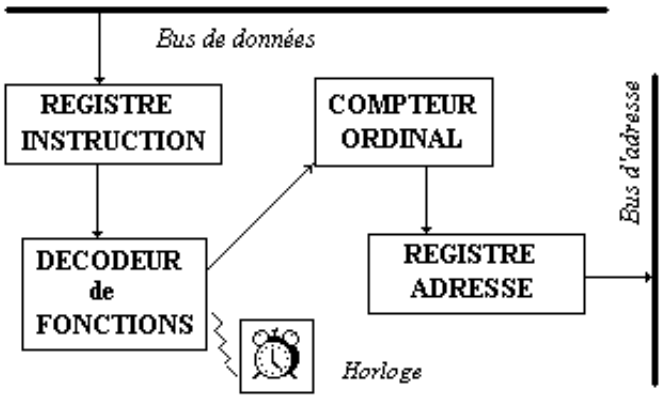

Fig. 4 – Unité de contrôle

— un compteur ordinal ou « Program Counter PC »qui contient l'adresse de la prochaine instruction ;
— un registre d'instruction ou « Instruction Register IR ou Current Instruction Register CIR »qui contient l'instruction lue ;
— un décodeur d'instruction ;
— un registre d'adresse ou « Memory Address Register MAR »qui contient l'adresse mémoire de l'instruction à lire ;
— une horloge.

2.1.3 Les mémoires

Le rôle des mémoires est d'enregistrer les programmes et les données pouvant être executées par le microprocesseur. On peut classer les mémoires en deux grandes catégories :

— les mémoires de travail ;
— les mémoires de stockage.

Les mémoires vives ou RAM (*Random Access Memory*) permettent des opérations de lecture et d'écriture. Elles sont volatiles. Les mémoires accessibles uniquement en lecture sont connues sous le nom de ROM (*Read Only Memory*).

Les mémoires de stockage (ou mémoire de masse) se présentent souvent sous la forme de disque. Même si la fonction est la même (*mémorisation de l'information*), toutes ces mémoires ont des caractéristiques différentes (voir figure 5 page 5).

2.1.4 Les entrées-sorties

Les périphériques d'entrée-sortie permettent à l'ordinateur de communiquer avec l'homme ou d'autres machines. Ils étaient initialement commandés par l'unité centrale. Depuis le début des années 1960 ils sont sous le contrôle de processeurs autonomes.
Exemple de fonctionnement d'un périphérique d'entrée-sortie : le disque dur (voir figure 6 page 5).
Supposons que l'unité centrale veuille lire un bloc sur le disque dur. Les principales étapes sont les suivantes :

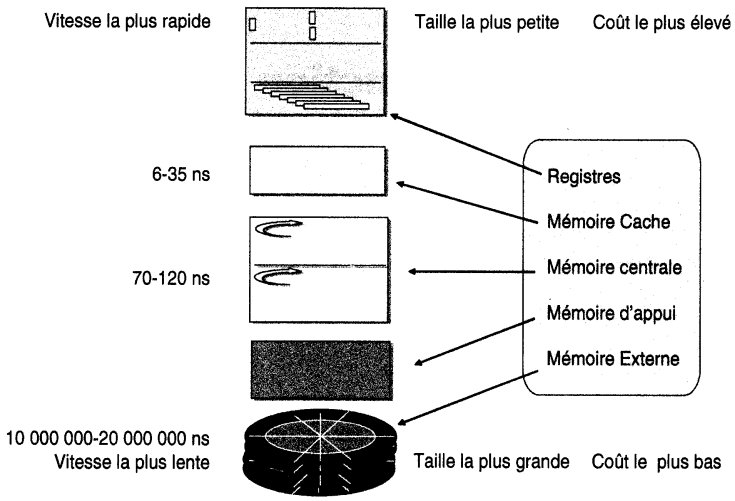

Fig. 5 – Les mémoires

— l'unité centrale formule cette demande en écrivant à l'adresse mémoire du contrôleur de disque ;
— le contrôleur de disque se charge d'écrire directement en mémoire les octets lus depuis le disque ;
— le contrôleur de disque signale la fin de la lecture sur le disque par le biais d'un mécanisme d'interruption (non abordé ici) ;
— l'unité centrale execute la routine d'interruption puis reprend le flot d'instructions où il a été interrompu.

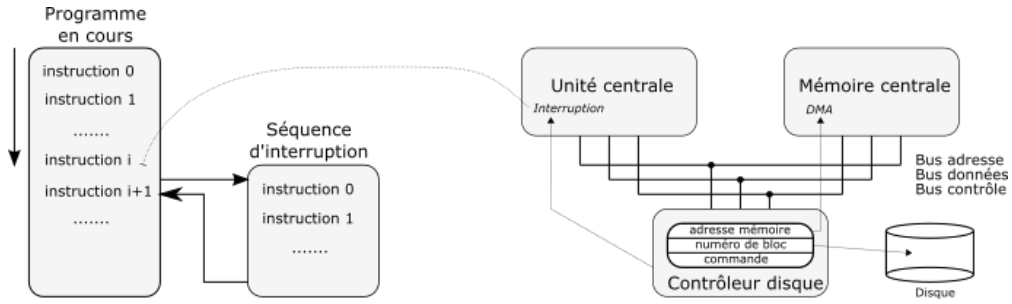

Fig. 6 – Exemple de dispositif d'entrées-sorties : disque dur et contrôleur disque

2.2 Les bus

Les différentes unités sont interconnectées par des systèmes de câblage appelé bus. Autour du processeur on trouve :

— le bus d'adresse unidirectionnel ;
— le bus de données bidirectionnel ;

— le bus de contrôle bidirectionnel.

Remarque : les ordinateurs récents possèdent d'autres types de bus.

2.3 Les instructions

Une instruction désigne un ordre donné au processeur. Il s'agit d'une chaine binaire de p bits composée de deux parties.

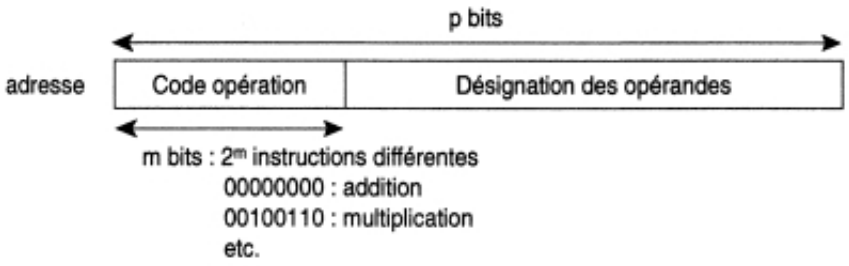

Fig. 7 – Format d'une instruction

Pour s'affranchir des codes binaires et des calculs d'adresse le programmeur utilise plutôt un **langage d'assemblage** où les instructions binaires sont remplacés par une chaine de caractères mnémoniques. Un programme appelé **assembleur** réalisera ensuite le passage vers le code binaire.

Les intructions du langage machine peuvent être rangées dans six catégories : calcul, transfert, entrées/sorties, saut, appel de sous programme, instructions particulières (arrêt par exemple). L'ensemble des codes opération reconnu par un processeur s'appelle son *jeu d'instructions*.

Il n'est pas question dans ce cours de détailler un quelconque jeu d'instructions, ni les subtilités de son utilisation. Il s'agit plutôt de présenter quelques séquences simples.

Dans les exemples qui suivront le jeu d'instructions minimaliste utilisé (*voisin de celui des processeurs ARM*) peut être trouvé à cette adresse. Les commentaires sont précédés du caractère ;.

2.3.1 Exemple commenté n° 1

```
MOV R0,#30;chargement du registre R0 avec la valeur 30
MOV R1,#20
ADD R2,R1,R0;additionne les contenus de R0 et R1 et place le contenu dans le registre R2
STR R2,100;transfert le contenu de R2 à l'adresse 100
HALT;arret des traitements
```

2.3.2 Exemple commenté n° 2

```
0       MOV R0,#30
1       MOV R1,#20
2       MOV R3,#0
3       ADD R2,R1,R0
4       CMP R2,#0
5 si:   BNE fin
```

```
6       MOV R2,R3
7 fin:  HALT
```

Dans cet exemple, on introduit la possibilité d'introduire des *étiquettes* utiles pour réaliser des sauts. Elles évitent d'avoir à calculer l'adresse de l'instruction en question (*laissant ce travail à l'assembleur*). La structure présentée ici est équivalente à un test sans alternatives.

2.3.3 E1C5 Que fait cette séquence d'instructions ?

1. Expliquer ce réalise les instructions ci-dessous :

```
MOV R0,#25
     STR R0,20
     MOV R1,#6
     STR R1,21
     ADD R0,R1,R0
     LSL R0,R0,#1
     STR R0,22
     HALT
```

2. Vérifier vos résultats avec le simulateur de Peter Higginson.

2.3.4 E2C5 Langage d'assemblage

Écrire en langage d'assemblage les instructions correspondant aux actions suivantes :

- ▷ comparer la valeur du registre R4 avec la valeur 18 (décimale)
- ▷ si celle-ci est plus grande alors sauter à l'étiquette 'etiqu1'
- ▷ charger la valeur 14 dans le registre R0 ;
- ▷ arrêter les traitements
- ▷ déclarer l'étiquette 'etiqu1'
- ▷ charger la valeur 18 dans le registre R0 ;
- ▷ arrêter les traitements

2.4 E3C5 Execution d'une séquence d'instructions

On utilise le simulateur de Peter Higginson.

1. Entrer les instructions suivantes (**sans les numéros de ligne !**) :

```
0       MOV R0,#30
1       MOV R1,#20
2       MOV R3,#0
3       ADD R2,R1,R0
4       CMP R2,#0
5 si:   BNE fin
6       MOV R2,R3
7 fin:  HALT
```

2. Executer cette séquence pas à pas (bouton **STEP**). Apeler le professeur pour validation.
3. Comment évolue le registre PC du processeur ?
4. Que se passe-t-il à la ligne 5 ? Expliquer l'évolution du registre PC.

3 Qu'en est-il aujourd'hui ?

Presque 75 ans après sa présentation le modèle d'architecture de von Neumann est toujours valable. Cependant, les différences de vitesse entre le processeur et la mémoire ont conduit les fabricants d'ordinateur à intercaler des mémoires caches très rapides entre la mémoire centrale et le processeur.

Par ailleurs, les ordinateurs actuels comportent plusieurs processeurs (on dit aussi plusieurs « cœurs ») intégrés sur une même puce. Cette tendance au « parallélisme »dans le traitement et la circulation des informations conduit à une augmentation de la puissance de calcul sans augmenter la fréquence des processeurs individuels.

Enfin, une différence peu significative par rapport au modèle original, est que les dispositifs d'entrées/sorties peuvent communiquer avec la mémoire par le biais de contrôleur dédié (DMA). Le schéma actuel serait plutôt le suivant :

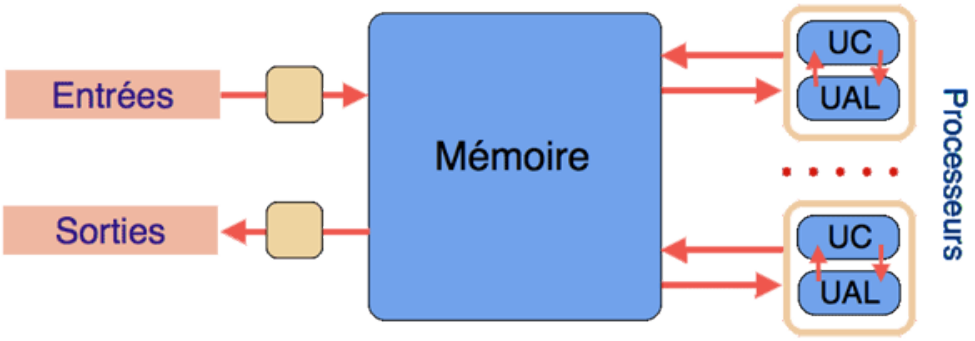

Fig. 8 – Modèle actuel

4 Alternative : architecture de Harvard

Dans le modèle d'architecture de Harvard les instructions et les données sont situées dans des mémoires différentes et sont véhiculés sur des bus indépendants.

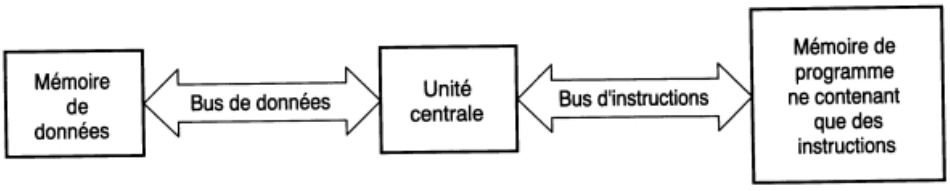

Fig. 9 – Modèle d'architecture de Harvard

La vitesse d'execution est de fait améliorée car en un seul cycle d'horloge on peut récupérer les données et le code instruction.

L'architecture de Harvard se retrouve beaucoup dans les systèmes embarqués.