Modèle de von NEUMANN (par Pascal Combe Deschaumes)

Pascaline – 1652

ENIAC – 1946

Macintosh – 1984

MacBookAir – 2008

Au programme[1]

Exprimer un algorithme dans un langage de programmation a pour but de le rendre exécutable par une machine dans un contexte donné.

La découverte de l'**architecture des machines** et de leur système d'exploitation constitue une étape importante.

Les circuits électroniques sont au cœur de toutes les machines informatiques.

Les réseaux permettent de transmettre l'information entre machines. Les systèmes d'exploitation gèrent et optimisent l'ensemble des fonctions de la machine, de l'exécution des programmes aux entrées-sorties et à la gestion d'énergie.

Contenus	Capacités attendues	Commentaires
Modèle d'**architecture séquentielle** (**von Neumann**)	- Distinguer les **rôles** et **les caractéristiques des différents constituants** d'une machine - **Dérouler l'exécution d'une séquence** d'instructions simples du type langage machine	- La présentation se limite aux concepts généraux - On distingue les architectures **monoprocesseur** et les architectures **multiprocesseur** - Des activités débranchées sont proposées - Les **circuits combinatoires** réalisent des fonctions booléennes

[1] http://cache.media.education.gouv.fr/file/SP1-MEN-22-1-2019/26/8/spe633_annexe_1063268.pdf

1. Introduction

Même si tous les éléments du puzzle sont présents en 1945, il n'est pas encore possible, à cette date, de parler d'ordinateur. En effet, tous les calculateurs de l'époque sont fondamentalement conçus sur le même modèle que les premières machines à calculer. Même l'ENIAC (Electronic Numerical Integrator And Computer) en 1946, longtemps qualifié à tort de premier ordinateur, nécessitait une saisie préalable et des données et des étapes de calcul à effectuer. **À cette époque on ne programmait pas encore : on configurait la machine en vue de tel ou tel calcul**.

ENIAC

EDSAC

Par le plus grand des hasards, en août 1944, un responsable de la supervision militaire de l'ENIAC croise sur un quai de gare John von NEUMANN (probablement l'un des plus grands logiciens du 20e siècle et un très grand physicien). Il lui demande son avis sur l'ENIAC qui n'était pas encore opérationnel, ainsi que des propositions pour construire son successeur. À la suite de nombreux échanges, von NEUMANN publiera un pré-rapport qui définit l'architecture, dite « **architecture de von NEUMANN**», du successeur de l'ENIAC, l'EDVAC, et, de fait, des ordinateurs qui suivront.

La conception en est résolument nouvelle.
Le calculateur devient une machine arithmétique, logique et, plus généralement, de traitement de l'information. Il est doté d'une vaste mémoire permettant de stocker et des données et **des programmes enregistrés, qui ne sont désormais plus fondamentalement différents des autres données** (idée issue des travaux sur les machines de TURING). Il est **piloté par une unité de commande interne** (idée de von **NEUMANN**).

Note Historique

> **Janos NEUMANN** est né le 28 décembre 1903 à Budapest. Il se fera appeler John Von NEUMANN après sa naturalisation américaine. Il vit dans un milieu intellectuel stimulant. Les plus grands scientifiques, écrivains … fréquentent le salon de ses parents. Il dispose de dons exceptionnels pour l'apprentissage.
>
> Il étudie la chimie à Zurich. Peu intéressé par ce domaine, il suit en parallèle le cours d'EINSTEIN à Berlin, et des cours de mathématiques à Budapest. Il obtient son diplôme d'ingénieur chimiste et un doctorat de mathématiques en 1926.
>
> Le début de sa carrière est consacré aux fondements logiques des mathématiques (à la suite des travaux de David HILBERT) et aux fondements mathématiques de la mécanique quantique. Il unifie les théories de

SCHRÖDINGER et de HEISENBERG, et apporte notamment le puissant outil des algèbres d'opérateurs.

Il s'installe aux Etats-Unis suite à la montée du nazisme et de l'antisémitisme en Europe. Avec l'imminence de la guerre, et après sa naturalisation, il devient un des principaux consultants de l'armée américaine et participe activement à la mise au point de la première bombe atomique. A cette occasion, il développe avec Steeve ULAM les méthodes dites de MONTE-CARLO, qui permettent en simulant un grand nombre de tirages aléatoires, de donner des solutions numériques à des équations aux dérivées partielles.

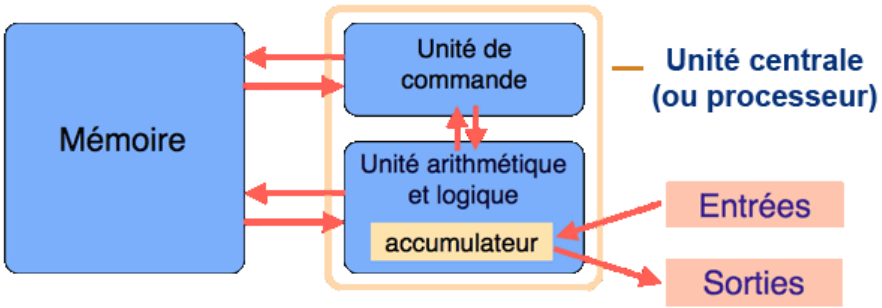
Il perçoit, lors de la réalisation de la bombe, l'importance à venir des machines électroniques pour réaliser des calculs insurmontables à la main et **contribue de façon décisive à la mise en œuvre des premiers ordinateurs**. Il est ainsi le premier à avoir l'idée que **le programme doit être codé et rangé dans la mémoire de la machine à côté des données des calculs**. En particulier, **une seule machine peut réaliser toute sorte de calculs différents**. Ce **modèle de Von NEUMANN** préside toujours à la conception des ordinateurs modernes.

2. Architecture séquentielle

C'est en 1834 que **Charles BABBAGE** eut l'idée d'incorporer des cartes perforées dans la machine à calculer, donnant ainsi une suite d'instructions à réaliser (il ne terminera jamais sa machine la plus performante faute de financemenent). **Ada LAVALACE**, qui travailla avec lui, écrivit le premier langage informatique de l'histoire.

2.1. Le modèle architectural de Von NEUMANN

Depuis cette époque, on utilise un schéma de référence tel que celui ci :



Cette architecture décompose l'ordinateur en **4** parties distinctes :

1. l'**unité arithmétique et logique** (UAL) ou **unité de traitement** qui effectue les opétations élémentaires (les opérations arithmétiques (+, −, ×, /, …), les opérations logiques (∨, ∧, ⊕, ¬, …), les opérations de comparaison (=, ≠, <, >, …))

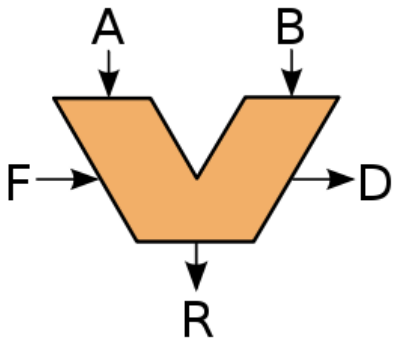
2. l'**unité de commande** (UC) qui assure la partie séquentielle des opérations
3. la **mémoire** qui contient à la fois les données et le programme qui indique à l'UC les calculs à effectuer sur les données
4. les **dispositifs d'entrée** (permet de recevoir des informations) **et de sortie** (permet d'envoyer des informations) qui permettent de communiquer avec le monde extérieur.

Remarque

- L'**UAL** et l'**UC** sont **regroupées dans le procésseur**. Il communique avec la mémoire par l'intermédiaire du bus.
- **L'ensemble est rythmé par une horloge** interne qui détermine la fréquence du processeur.

2.1.1. L'unité arithmétique et logique (UAL)

L'unité arithmétique et logique constitue le cœur de l'ordinateur, dont la représentation habituelle est la suivante :



A et **B** sont les opérandes (**entrées**)
R est le **résultat**
F est une **fonction binaire** (opération à réaliser)
D est un **drapeau,** indiquant un résultat secondaire (signe, erreur, division par 0, …)

Remarque

- Il existe des UAL spécialisées.
- Un même procécesseur peut comporter plusieurs UAL.
- Les UAL fonctionnent à l'aide de circuits électroniques qui executent des fonctions binaires.

2.1.2. La mémoire

C'est un tableau dans lequel sont stockés des variables et des programmes. Chaque cellule du tableau possède une adresse X et la valeur de chaque cellule est notée sous la forme M[X].

Il existe des mémoires de différents types :

- mémoire **vive** (RAM) : **mémoire volatile** qui stocke à court terme les données qui vont être traitées (en mode lecture ou en mode écriture) par le processeur
- mémoire **morte** (ROM) : utile au démarrage de l'ordinateur (mode **lecture**)
- mémoire externe (**de masse**) : disques durs, clés USB, …

2.1.3. L'unité de commande

C'est elle qui donne les ordres aux autres parties de l'ordinateur. Elle **se compose** essentiellement **de 4 registres** :

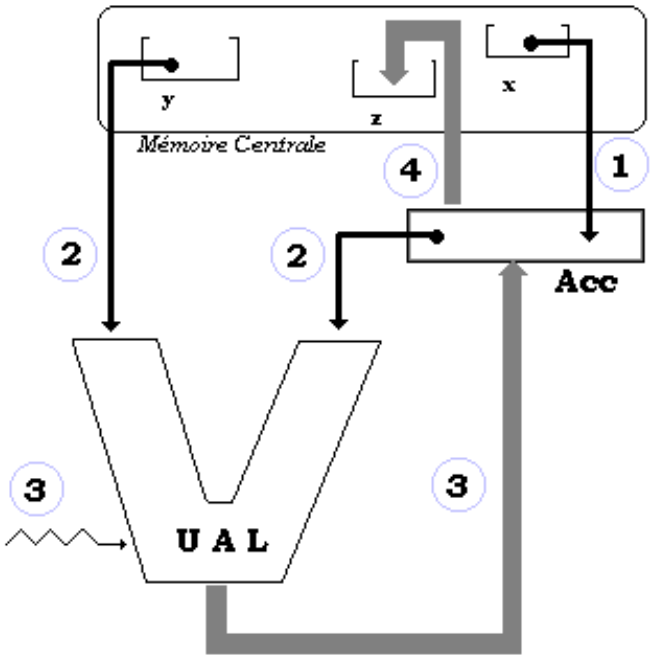
- le **compteur ordinal** (CO) designe, au registre d'instructions, l'instruction à exécuter, située à l'adresse M[X]
- le **registre d'intructions** (RI) charge l'instruction à exécuter (opération **OP** et opérande **AD**)
- **l'accumulateur** (AC) stocke les opérandes intermédiaires
- le **code condition** (CC) utilisé pour les instructions de rupture conditionnelle (saut). Le contenu du compteur ordinal est remplacé par l'adresse du saut.

2.2. Aspect séquentiel des opérations assurées par l'unité de contrôle

Le langage de base d'une machine est **l'assembleur**. **Il existe autant de langages assembleur qu'il existe de processeurs**. Un progamme écrit en assembleur ne sera « compris » que par le processeur destinataire contrairement à un programme écrit dans un langage comme Python qui sera « compris » par plusieurs machines.
Dans la forme la plus simple du langage assembleur, une instruction est stockée dans une cellule mémoire désignée par une adresse.
Une instruction comprend deux parties :

- un **code opération** (CO) : indique quelle opération effectuer
- une **adresse** (ADR) : opérande (i.e. arguments) de l'opération

Exemple : **Somme de** deux nombres **x et y**



Z = X + Y est **décomposée** (fictivement) en **3 opérations** distinctes :

1	**LD X**	Chargement de l'**accumulateur** avec x
2	**ADD Y**	Préparation des opérandes (entrées) X et Y vers l'**UAL**
3		Lancement **commande de l'opération** dans l'**UAL**
		Résultat (sortie) transféré dans l'accumulateur
4	**STO Z**	Copie de l'accumulateur **dans Z (l**'accumulateur garde son contenu)

Commentaires

- **LD X** : chargement (*load*) du contenu de la cellule mémoire X dans l'accumulateur (i.e. un des registres noté A)
- **ADD Y** : additionner le contenu de la cellule mémoire Y avec le contenu de l'accumulateur
- **STO Z** : stocke (*store*) le contenu de l'accumulateur dans la cellule mémoire Z

- Toute expression arithmétique ou logique peut être décomposée en une suite d'instructions exprimées en assembleur.
- L'unité de commande enchaine alors séquentiellement la suite d'instructions pour calculer cette expression.

Jeu d'instructions

Une **instruction-machine** est une instruction qui est directement exécutable par le processeur.

L'ensemble de toutes les instructions-machine exécutables par le processeur s'appelle le " **jeu d'instructions** " de l'ordinateur.

Il est composé au minimum de quatre grandes classes d'instructions dans les micro-processeurs :

- instructions de **traitement**
- instructions de **branchement** ou de **déroutement**
- instructions **d'échanges**
- instructions de **comparaisons**

Remarque

D'autres classes peuvent être ajoutées pour améliorer les performances de la machine (instructions de gestion mémoire, multimédias etc..)

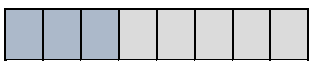
3. **Application**

On imagine un ordinateur simplifié doté d'une **mémoire centrale** constituée d'un tableau de $2^5 = 32$ cellules.

Mémoire

00				04		08		12		16		20		24		28	
01				05		09		13		17		21		25		29	
02				06		10		14		18		22		26		30	
03				07		11		15		19		23		27		31	

Chaque cellule de la mémoire est constituée d'un octet (8 bits).



Cellule mémoire
00 (8 bits)

Chaque **instruction** est composée d'un **code opération** codé sur trois bits (2^3 = 8 instructions possibles) et d'une **adresse** codée sur cinq bits (2^5 = 32 adresses possibles).

Code op Adresse

Instruction

Jeu des 8 instructions

Instruction	Effet de l'instruction	
LD X	Charge le contenu de la cellule X (dans l'accumulateur)	ACC ← M[X]
STO X	Enregistre le contenu de l'accumulateur dans la cellule X	M[X] ← ACC
ADD X	Additionne le contenu de la cellule X au contenu de l'accumulateur	ACC ← ACC + M[X]
SUB X	Soustrait le contenu de l'accumulateur avec le contenu de la cellule X	ACC ← ACC – M[X]
JMP ADR	Saute à l'adresse ADR	CO ← ADR (Le contenu du Compteur Ordinal est remplacé par l'adresse ADR)
JMP**Z** ADR	Saute à l'adresse ADR si le contenu de l'accumulateur est **nul**	
JMP**P** ADR	Saute à l'adresse ADR si le contenu de l'accumulateur est **positif**	
JMP**N** ADR	Saute à l'adresse ADR si le contenu de l'accumulateur est **négatif**	

Application 1 – **Programme mystère**

On suppose qu'un programme commence à l'adresse 8 et que les adresses 0 à 7 sont réservées au stockage des données.

Que fait le programme suivant ?

00	48
01	37
02	
…	…
08	LD 0
09	SUB 1
10	STO 2
11	END

Application 2 – **Programme de comparaison**

On suppose que les cases mémoire M[0] et M[1] contiennent respectivement les valeurs a et b.

Ecrire un programme qui lit le contenu de la case mémoire 1, qui la compare à celle de la case mémoire 0, et, si elle est plus grande, l'écrit dans la case mémoire 2.