Activité 3 – Architecture matérielle

I. Histoire des ordinateurs

1. Machines à programmes externes

- *Machines électromagnétiques*

— L'allemand Konrad Zuse achève le Z1 en 1938 puis le Z3 en 1941 qui lisait son programme sur une bande perforée. Le Z3 utilisait le calcul en virgule flottante et réalisait 3 ou 4 additions à la seconde.
— En 1944, L'américain Howard Aiken construit l'ordinateur électromécanique **Mark I** qui pesait 5 tonnes.

- *Machines électroniques*

— L'apparition des **tubes à vide** marque le début de l'électronique.
— John Vincent Atanasoff en 1942 construit la première machine électronique sans l'achever complètement.
— Entre 1943 et 1945 les Britanniques Max Newmann et Tomy Flowers utilisent les premiers ordinateurs à tubes à vides nommés **Colossus** pour déchiffrer le code de Lorenz employé par les Allemands. Notons que cette information n'a été révélée qu'en 1970 !
— Le célèbre **ENIAC** de John W. Mauchly et J. presper Eckert est achevé en 1945 et permet avec ses 18 000 tubes à vide d'effectuer des calculs balistiques.

2. Machines à programmes enregistrés

— En 1948 apparaissent les premières machines à programmes enregistrés, ancêtres directs des ordinateurs actuels. Les données et programmes résident en mémoire. Ces machines sont basés sur les travaux de Mauchly, Eckert et Von Neumann.
— Au début des années 1950 apparaisent les premiers ordinateurs commerciaux avec **IBM**, **DEC** et **Bull** (en France depuis 1931).

3. Du mini-ordinateur à la micro-informatique

- *Miniaturisation*

— Le **transistor** apparaît en 1947 à la place des tubes à vide et on commence à le fabriquer à faible coût au milieu des années 1950.
— Le **circuit intégré** apparaît en 1958.
— En 1971 le premier **microprocesseur** voit le jour, c'est l'Intel 4001. L'informatique s'ouvre alors aux particuliers.
— De multiples machines sont commercialisées : l'Altair 8008, l'Apple II (1977), l'IBM PC (1981), le ZX 81 (1981), le commodore 64 (1982), le Macintosh (1984) ...
— Le **ZX 81** est considéré à son époque comme le premier ordinateur familial en kit en France, sa résolution et sa capacité mémoire (1 ko) ne permettait pas énormément de prouesses au niveau des jeux.

- *Langage et système d'exploitation*

— Après le premier compilateur conçu en 1951 par Grace Hopper, le langage **Fortran** est spécifié en 1954 et achevé en 1956 par John Backus.
— Suivent ensuite : le **Lisp**, le **Cobol** et le **Basic** en 1964. Puis de 1970 à 1980 : le C (1972), le ML (1973) dont est issu Caml, Ada (1983) et C++ (1986).
— La première version de **Python** date de 1991 (Guido van Rossum) et **JavaScript** a été publié en 1995.
— Au milieu des années 1960, chaque constructeur développe son propre système d'exploitation : OS/360, puis MSV chez IBM, système Unix (1970) chez AT&T (Bell ...)

— **MS-DOS** écrit par Microsoft pour IBM s'impose. Il est suivi par **Windows** en 1985.
— Le 27 septembre 1983, Richard Stallman dévoile son projet de développer un système d'exploitation compatible UNIX appelé **GNU** - acronyme récursif (la forme développée du sigle contient sa forme réduite) qui signifie en anglais " GNU's Not UNIX " (littéralement, " GNU n'est pas UNIX ") , en invitant la communauté hacker à le rejoindre et participer à son développement.

Exercice 1
Question 1
Les tubes à vide, volumineux et peu fiables, ont été remplacé au milieu des années 1950 par :
a. les cartes perforées
b. les tubes à plasma
c. les transistors
d. les cartes magnétiques

Question 2
Le premier microprocesseur (Intel 4004) a permis :
a. l'ouverture du marché aux particuliers
b. de déchiffrer le code de la machine Enigma
c. de permette les voyages spatiaux
d. de déchiffrer le code de César

Question 3
Le premier langage informatique de haut niveau, autre que le langage machine a été (en 1954) :
a. Fortran
b. Python
c. C
d. Java

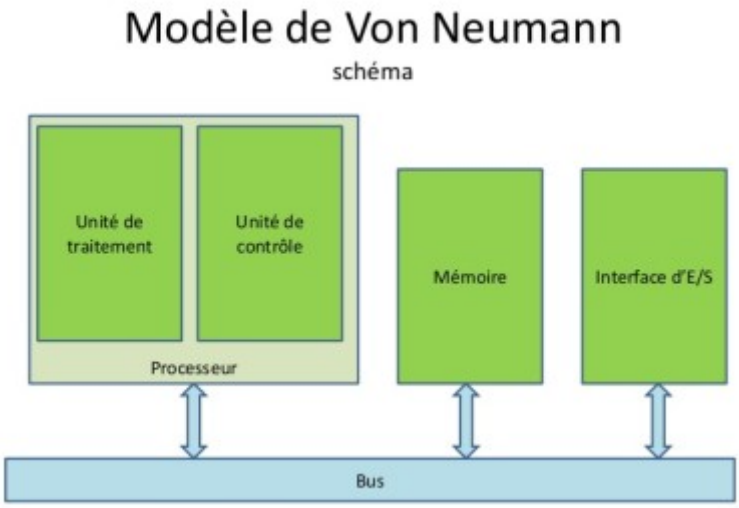
II. Architecture de von Neuman

L'architecture de tous les ordinateurs actuels est conforme à un schéma qui a assez peu évolué depuis les premiers ordinateurs électronique à tubes à vide de 1945 (Colossus et ENIAC). Ce modèle est dit de **von Neumann**.

1. L'architecture de von Neumann

- Les principales structures

Dans l'architecture de von Neumann, un ordinateur est constitué de quatre parties distinctes :
1. le **CPU : Central Processing Unit** (unité centrale de traitement) appelé aussi processeur.
2. la **mémoire** où sont stockés les données et les programmes ;
3. des **bus** qui sont des fils conduisant des impulsions électriques et qui relient les différents composants.
4. des **entrées-sorties** (E/S ou I/O input/Output) pour échanger avec l'extérieur.



- Les sous-structures

— Les échanges entre la mémoire et les registres du CPU se font **via des bus** selon une chronologie **organisée par l'horloge** et suivant la nature des échanges : données ou adresses.
— Un programme est enregistré dans la mémoire.
— L'**adresse** (un entier) **de l'instruction** en cours de traitement est stockée dans une mémoire interne au processeur, le **cp** ou **pc** : « registre compteur de programme »
— La **valeur de cette instruction** (un entier) est stockée dans une autre mémoire interne, le **ri** : « registre d'instruction » .
— Le **CPU** dispose aussi de mémoires internes dans le « banc de registres » où sont placées données du programme avant utilisation.
— L'**UAL** ou **ALU** (Unité Arithmétique et Logique) effectue les opérations arithmétiques et logiques, sur les données et les adresses, en interprétant les impulsions électriques.

2. Le rôle de l'horloge

Les traitements réalisés par l'ordinateur sont faits sur des représentations binaires des données et des traitements à réaliser, et ce en calculant des fonctions logiques.

- Cadence d'un processeur

— Le CPU dispose d'une horloge qui cadence l'accomplissement des instructions et dont l'unité est appelée cycle.
— La fréquence s'exprime en GigaHertz (GHz), elle signifie le nombre d'opérations que fait le processeur en une seconde. (3 GHz (GigaHertz) : 3 milliards d'opération à la seconde) En clair, elle influe sur la vitesse de fonctionnement du processeur.
— En 2020, les processeurs tournent entre 1,5 et 3 GHz. Certains atteignent 3,6 GHz mais la course à la fréquence à pris fin depuis 2005 environ car au-delà d'un certain cap, la chaleur dégagée est trop importante et perturbe la lecture des tensions.

- Cycle d’instructions

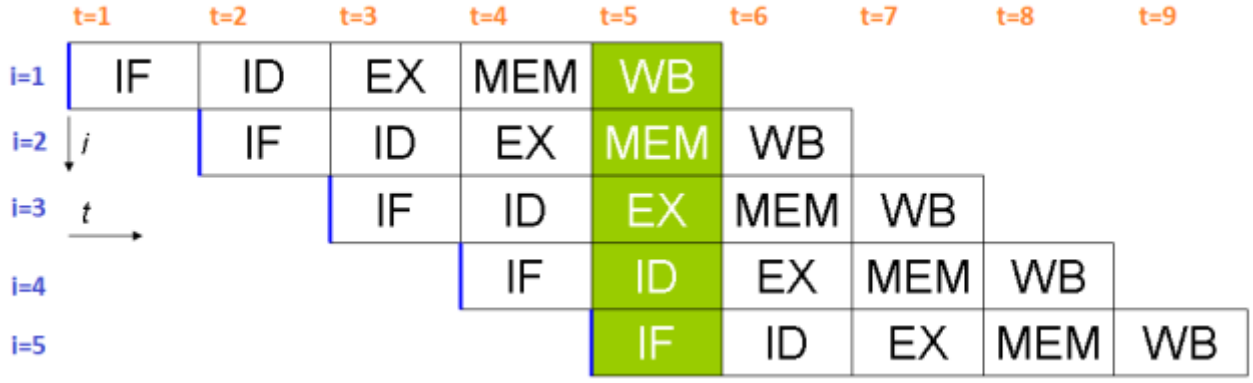
Le pipeline d’instruction

Dans un processeur, 5 étapes sont nécessaires pour traiter une instruction. Chacune de ces 5 instructions est exécuté lors d’un cycle.
— **LI** (ou **IF** - Instruction Fetch) : charge l’instruction à exécuter dans le pipeline (fetch = aller chercher).
— **ID** : décoder l’instruction ;
— **EX** : exécuter l’opération dans l’UAL ;
— **MEM** : accéder à la mémoire en lecture ou en écriture ;
— **ER** (ou **WB**, Write Back) : écrire le résultat dans un registre.

Pour gagner du temps, le processeur n’exécute pas les instruction de façon séquentielle mais il exécute simultanément plusieurs instructions qui sont à des étapes différentes de leur traitement. C’est le pipeline d’instruction.
Grâce au pipeline, le traitement des instructions nécessite au maximum les cinq étapes précédentes. Dans la mesure où l’ordre de ces étapes est invariable (LI, DI, EX, MEM et WB), il est possible de créer dans le processeur un certain nombre de circuits spécialisés pour chacune de ces phases.

Exemple d’un pileline
En supposant que chaque étape met 1 cycle d’horloge pour s’exécuter, il faut normalement 5 cycles pour exécuter une instruction, 15 pour 3 instructions :

En utilisant la technique du pipeline, notre processeur peut alors contenir plusieurs instructions, chacune à une étape différente. Il faut 9 cycles pour exécuter 5 instructions. À t = 5, tous les étages du pipeline sont sollicités, et les 5 opérations ont lieu en même temps.



Le premier ordinateur à utiliser cette technique est l’IBM Stretch, conçu en 1961.

Exercice 2

Question 4

Un pipeline d'instruction permet :
a. de faire passer les instructions dans un tunnel
b. d'exécuter des instructions séquentiellement
c. d'exécuter des instructions simultanément
d. d'exécuter des instructions randomisées

Question 5

L'UAL :
a. permet de gérer la mémoire
b. permet de gérer les entrées/sorties
c. est une adresse internet
d. est l'endroit où les calculs sont effectués

III. Mémoire et langage machine

En son cœur, la machine effectue des calculs à partir d'instructions simples en binaire qui peuvent être directement traduites dans un langage, l'assembleur. L'assembleur prend, traite et remplit des cases mémoires.

1. Organisation de la mémoire

- Les différents types de mémoire

Il existe de nombreux mécaniques de mémoires qui se distinguent par leur durabilité (volatile ou permanente), leur mode d'accès (par adresse ou dans l'ordre de rangement).

1. Mémoire vive RAM (Random Acces Memory).

La mémoire vive (RAM) est une mémoire volatile (qui perd ses données lorsqu'on coupe son alimentation électrique). Il s'agit des registres, des mémoires cache, de la mémoire centrale.
Il y a deux types principaux de mémoire vive :
— la mémoire vive dynamique (DRAM) qui, même sous alimentation électrique, doit être réactualisé périodiquement pour éviter la perte d'information ;
— la mémoire vive statique (SRAM) qui n'a pas besoin d'un tel processus lorsque sous alimentation électrique.

2. Mémoire morte ROM (Read-Only Memory).

La mémoire morte est une mémoire non volatile (mémoire rémanente qui conserve ses données même lorsqu'on coupe son alimentation électrique). Par exemple les disque SSD (Solid State Drive), les disques magnétiques.
Les mémoires mortes sont utilisées, entre autres, pour stocker :
— les informations nécessaires au démarrage d'un ordinateur (BIOS, instructions de démarrage, microcode) ;
— des tables de constantes ou des tables de facteurs de conversion ;

Elle fait aussi partie des microprogrammes présents dans les ordinateurs et la plupart des appareils électroniques (smartphone, baladeur et autres lecteurs de CD/DVD) mais aussi la plupart des appareils programmables (TV, réveil, machine à laver, lave vaisselle, etc.).
Le temps d'accès à la mémoire morte est de l'ordre de grandeur de 150 nanosecondes comparativement à un temps d'accès d'environ 10 nanosecondes pour la mémoire vive.

- Les registres

— Un registre est un emplacement de mémoire interne à un processeur. Les registres se situent au sommet de la hiérarchie mémoire : il s'agit de la mémoire la plus rapide d'un ordinateur, mais dont le coût de fabrication est le plus élevé car la place dans un microprocesseur est limitée.
— L'accès par le processeur à une information située dans la DDR SDRAM de la mémoire centrale est 100 fois plus lente qu'un accès à une information contenue dans un registre.
— Les registres sont utilisés pour stocker des opérandes et des résultats intermédiaires lors des opérations effectuées dans l'UAL (Unité Arithmétique et Logique) du processeur.
— La plupart des PC actuels ont des registres de tailles 64 bits.

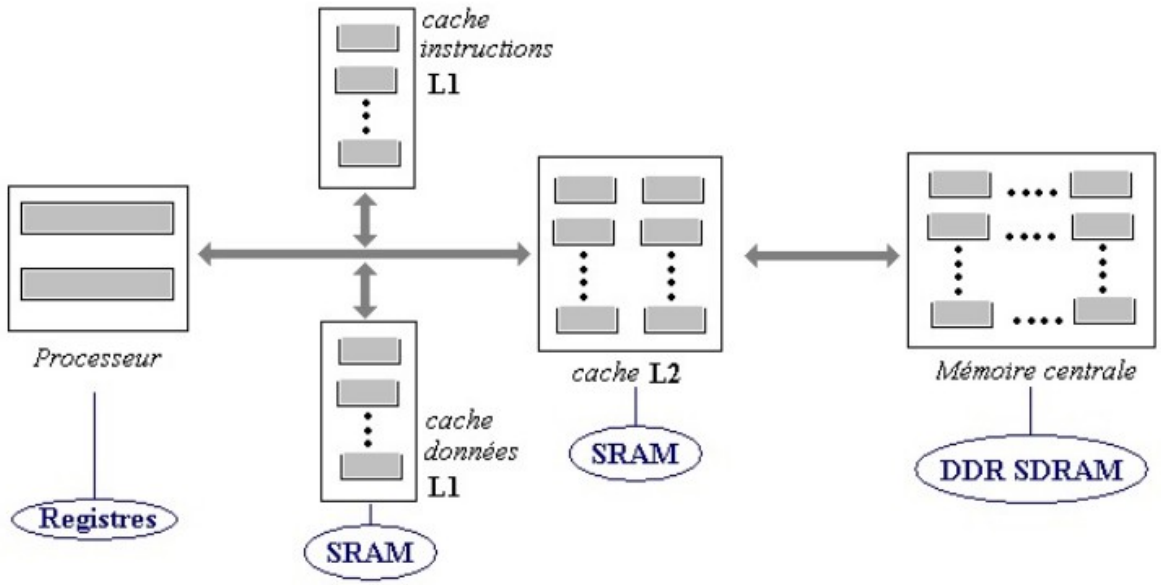
- Mémoires centrales et mémoire cache

1. La mémoire centrale : RAM (Random Acces Memory)

— La mémoire centrale est une mémoire vive qui contient les programmes en cours et les données qu'il utilise. Elle est de taille importante (plusieurs Go).
— Elle est organisée en cellules qui contiennent chacune une donnée ou une instruction repérée par un entier : un adresse mémoire.
— Le temps d'accès à chaque cellule est identique, on parle improprement de mémoire à accès aléatoire ou RAM (Random Acces Memory) mais on devrait plutôt parler de mémoire à accès direct.

2. La mémoire cache

— Pour pouvoir adapter la très grande vitesse du processeur à celle bien plus faible de la RAM, on place entre les deux une mémoire très rapide, la mémoire cache.
— Il existe souvent plusieurs niveaux de mémoire cache : L1, L2 ...
— Généralement la mémoire cache de niveau L1 et celle de niveau L2 sont regroupées dans la même puce que le processeur (cache interne).



VI. Langage machine

Un ordinateur ne peut fonctionner sans une suite d'instructions, le programme, qui lui dise ce qu'il doit faire. La programmation est une des tâches les plus importantes des informaticiens. Pour programmer, il faut un langage manipulable par l'homme et interprétable par la machine.
On connaît trois niveaux de langages de programmation :

— le **langage machine** directement interprétable par la machine ;
— le **langage assembleur**, qui est du langage machine sous forme symbolique ;
— les **langages extérieurs** de programmation, synthétiques et faciles à programmer car ils sont proches du pseudo code utilisé pour les algorithmes mais ils nécessitent d'être traduits en langage machine par un programme complexe appelé compilateur.

1. Langage machine

Les instructions et les données traitées par le processeur sont des suites de 0 et de 1. C'est le langage machine. Le langage machine est le seul langage qu'un processeur puisse exécuter. Le jeu d'instructions est spécifique au processeur. Les instructions du langage machine doivent être codées en binaire et implantées dans la mémoire. Pour leur manipulation par un humain, on les représente le plus souvent en hexadécimal.

Parmi les instructions, on distingue les genres suivants :

— manipulation de registres, notamment les registres de l'accumulateur ;
— opération entre registre et mémoire, par exemple charger un registre avec une donnée mémoire, stocker le contenu d'un registre à une certaine adresse mémoire ou ajouter à un registre la donnée qui est à telle ou telle adresse ;
— branchement à une certaine adresse selon certaines conditions.

Les deux principaux inconvénients de ce langage sont :

— il est très peu parlant, si on ne connaît pas par cœur le code binaire de chaque opération, il faut sans cesse se reporter au répertoire du jeu d'instructions ;
— le programmeur doit gérer lui-même la mémoire, il doit attribuer une adresse-mémoire à chaque donnée qu'il envisage de stocker et se souvenir de toutes les adresses.

2. Langage assembleur

Le code opération de chaque instruction du langage machine est remplacé par un mot appelé mnémonique qui rappelle le mieux possible l'opération accomplie. Les mnémoniques sont à base anglaise.
L'adresse est fournie sous la forme d'un nom symbolique attribué à la donnée. C'est en somme un nom de variable. Le programmeur doit encore attribuer les adresses, mais pour désigner les données il utilise des noms qu'il peut (et doit) choisir aussi parlants que possible.

La transformation de ce code en langage machine est accomplie par un programme nommé assembleur, d'où par raccourci son nom de langage assembleur. Le premier langage assembleur n'a été écrit qu'en 1954 par Nathaniel Rochester.

Exemple : additionner deux nombres N1 et N2 et mettre le résultat dans SOMME.

Load A N1	*mettre le nombre N1 dans le registre A de l'accumulateur*
Add N2	*ajouter le nombre N2 ; l'accumulateur contient N1+N2 dans le registre A*
Store A SOMME	*ranger le contenu du registre A dans un espace mémoire nommé SOMME*

Il y a trois types d'instructions

1. instructions de transfert entre registres et mémoire
   - chargement
   - sauvegarde
2. instructions de calcul
   - additions, multiplications
   - opérations logiques
   - comparaisons, sauvegarde
3. instructions de saut
   - sauts inconditionnels
   - sauts conditionnels
   - sauvegarde
4. appels système

Exercices

Pour les exercices suivants, on imagine un processeur qui contient 4 registres, appelés A, B, C et D, et qui dispose du jeu d'instructions suivant :

— LD X adr. charge le contenu de la case mémoire d'adresse adr. dans le registre X
— ST X adr. stocke le contenu du registre X dans la case mémoire d'adresse adr.
— ADD XYZ additionne le contenu des registres X et Y puis le place dans le registre Z
— DEC X décrémente (-1) le contenu du registre X
— JUMP n saute à l'instruction numéro n du programme
— JUMPZ X n saute à l'instruction numéro n du programme si le contenu du registre X vaut 0
— END fin du programme.

Exercice 1.

Décrire ligne par ligne ce que fait ce programme. Quelle valeur se trouve dans la case mémoire 11 à la fin de ce programme ?

```
1 : LD A 7
2 : LD B 8
3 : ADD ABA
4 : LD B 9
5 : ADD ABA
6 : LD B 10
7 : ADD ABA
8 : ST A 11
9 : END
```

mémoire

adr.	donnée
7	42
8	68
9	47
10	33

Exercice 2.

Expliquer ce que fait le programme ci-dessous en supposant que le nombre x contenu dans la case mémoire d'adresse 10 est strictement positif.

1 : LD A 10		
2 : LD B 10	mémoire	
3 : JUMPZ A 7	adr.	donnée
4 : DEC A	10	x
5 : ADD ABB	11	y
6 : JMP 3		
7 : ST B 11		
8 : ST A 10		

Exercice 3.
Écrire un programme en assembleur équivalent aux programmes Python ci-dessous.

```
x=10
while x>0 :
    x-=1
```

```
x=5
y=0
while x>0 :
    x-=1
    y-=1
```

Exercice 4.
Écrire une séquence d'instructions qui multiplie par 5 le nombre contenu dans la case mémoire d'adresse 10 et stocke le résultat dans la case mémoire d'adresse 11.

Exercice 5.
Écrire un programme qui lit une valeur x dans la case mémoire 10 et calcule son opposé puis stocke le résultat à l'adresse 13. On suppose que cette valeur est un entier positif.

Exercice 6.
Écrire un programme qui lit deux valeurs x et y contenues respectivement dans les cases mémoires 11 et 12, calcule la différence y-x et stocke le résultat à l'adresse 13. On suppose que ces valeurs sont des nombres entiers positifs.

Exercice 7.
Modifier le programme de l'exercice 6 pour qu'il stocke 0 à l'adresse 15 si x est égal à y ou la valeur x sinon.

Exercice 8.
Écrire un programme en assembleur équivalent au programme Python ci-dessous.

```
x=0
while x<5 :
    x+=1
```