Modèle d'architecture séquentielle (Von Neumann)

Objectifs pédagogiques :

- ✓ Distinguer les rôles et les caractéristiques des différents constituants d'une machine.
- ✓ Dérouler l'exécution d'instructions simples du type langage machine

Depuis plus de soixante ans, l'architecture des ordinateurs est conforme à un schéma qui a peu évolué depuis son origine : le modèle dit « de von Neumann ». La naissance de ce modèle, sa diffusion et ses premières mises en œuvre sont un moment-clé de l'histoire de l'informatique

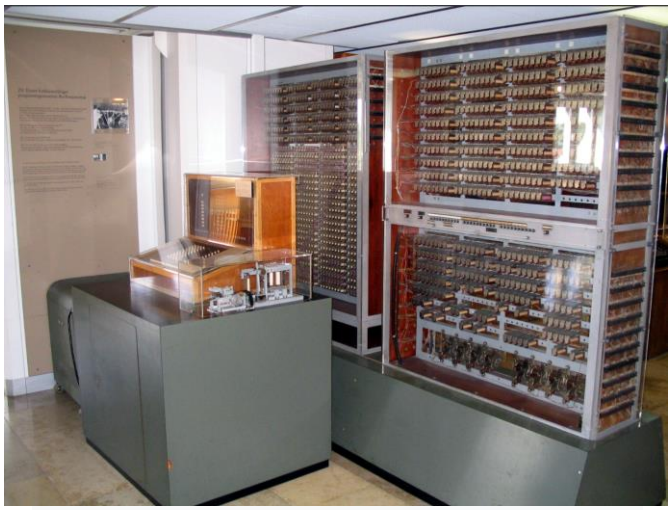
1. Brève histoire des ordinateurs

1.1. Les machines à programmes externes

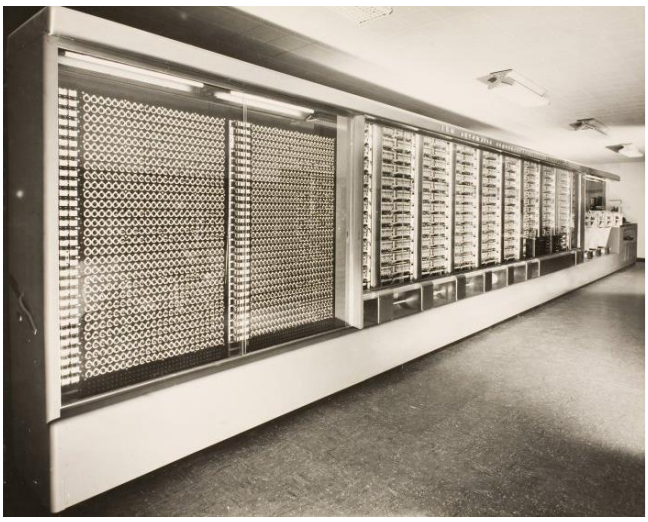
■ Machines électromécaniques

L'allemand **Konrad ZUSE** achève le **Z1** en 1938, un ordinateur mécanique utilisant le système binaire, puis le **Z3** en 1941, premier ordinateur complètement automatique lisant son programme sur une bande perforée. Le Z3 utilisait déjà le calcul en virgule flottante et réalisait 3 à 4 additions par seconde.

Peu après aux Etats-Unis, **Howard H. AIKEN** construit l'ordinateur électromécanique **MARK I** (1944) inspiré par les travaux de Babbage. Le **MARK I** pesait environ 5 tonnes et était composé de plus de 750 000 pièces !



Réplique du Zuse Z3 au Deutsches Museum de Munich.
Source : Wikipédia



L'ordinateur MARK I inauguré à Harvard le 7 août 1944 mesurait 16m de long, 2,40m de haut, 0,5 m de profondeur et pesait 4,5 tonnes. **Source : espace-turing.fr**

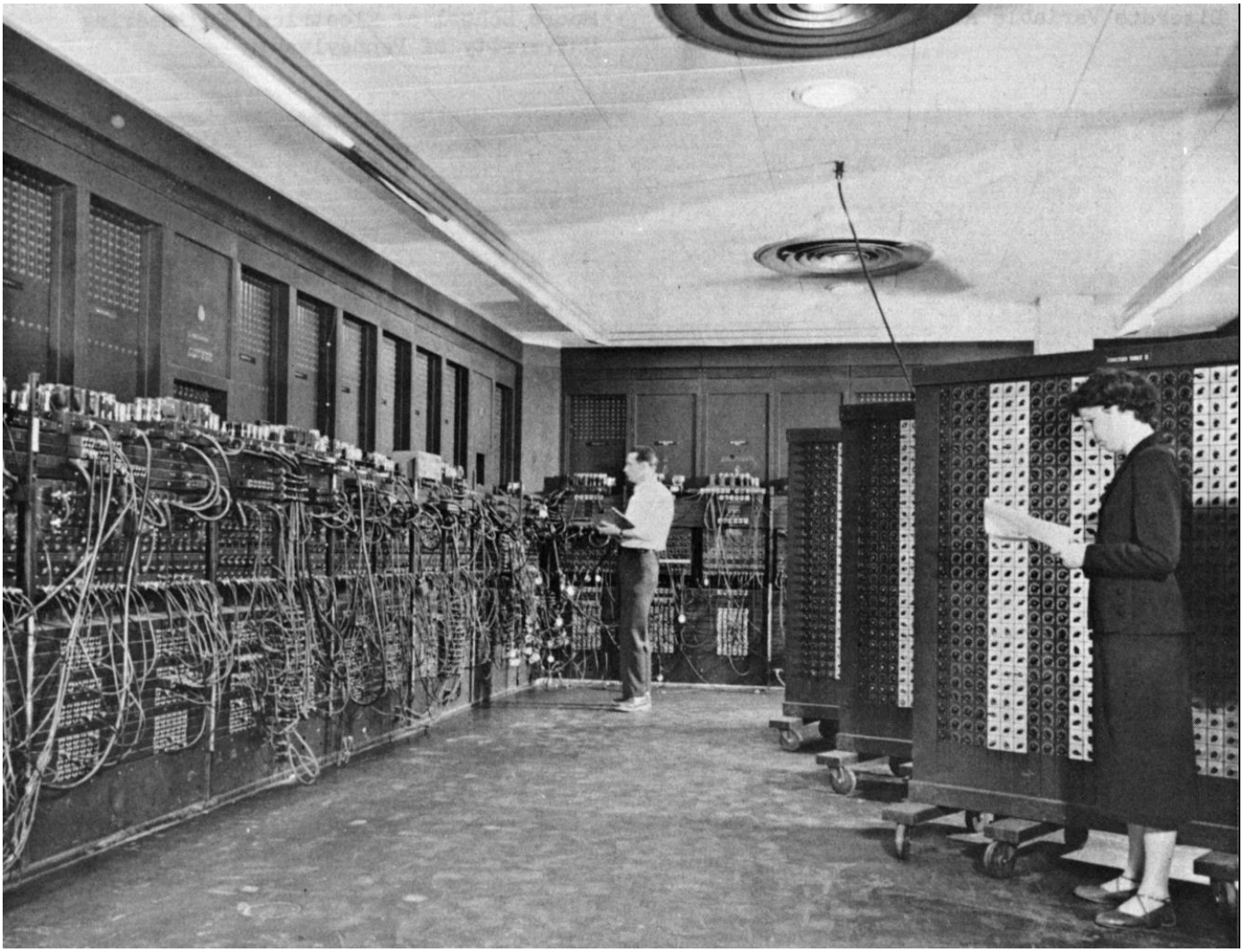
Le **Mark I** lisait ses instructions sur des cartes perforées et exécutait l'instruction courante puis lisait la suivante.

■ **Machines électroniques**

L'apparition des **tubes à vide**, bien plus rapides que les relais des machines électromécaniques, marque le début de l'électronique moderne. La construction de la première machine à base de composants électroniques est initiée par John Vincent ATANASOFF en 1942. Il ne pourra pas l'achever complètement.

Entre 1943 et 1945 les Britanniques **Max NEWMAN** et **Tomy FLOWERS** mettent en service les **COLOSSUS** utilisés pour déchiffrer le code de LORENZ employé par les allemands durant la seconde guerre mondiale.

Le célèbre **ENIAC** des Américains **John MAUCHLY** et **John ECKERT** est achevé en novembre 1945 et effectue des calculs balistiques à l'aide de 18 00 tubes à vide.



L'**ENIAC** (Electronic Numerical Integrator And Computer) : premier ordinateur entièrement électronique construit pour être Turing-complet (1945). Il peut être reprogrammé pour résoudre, en principe, tous les problèmes calculatoires. **Source : Wikipédia**

1.2. Les machines à programmes enregistrés

■ Basés sur les travaux de **MAUCHLY**, **ECKERT** et **VON NEUMANN**, les machines à programmes enregistrés sont les ancêtres directs des ordinateurs actuels. Dans ce type de machines, les données et les programmes résident en mémoire. Les premières machines de ce type appariassent dès 1948 avec les ordinateurs britanniques **BABY** et **EDSAC**, suivis par leurs homologues américains **EDVAC** et **UNIVAC**.

■ Le début des années 1950 voient apparaître les premiers ordinateurs commerciaux et les modèles se succèdent avec comme principaux acteurs les constructeurs **IBM** (International Business Machines), **DEC** (Digital Equipment Corporation) et **BULL**.

1.3. Du micro-ordinateur à la micro-informatique

■ Miniaturisation et explosion du marché

Le **transistor** (1947) devient un produit industriel très fiable qu'on peut fabriquer à faible coût au milieu des années 1950. Son émergence technologique, marque la fin des **tubes à vide**. Le circuit intégré qui rassemble de nombreux composants miniaturisés sur une petite surface apparaît en 1958 et ne cessera de se perfectionner au fil du temps.
Durant plus de 10 ans, **IBM** et **DEC** dominent le marche alors que la science informatique encore naissante, permet de former dans les université des personnes qui se spécialisent dans l'utilisation et la programmation des ordinateurs.
L'apparition du **microprocesseur** (INTEL 4004 en 1971) permet à de nouveaux acteurs d'apparaître et d'innover : l'informatique s'ouvre au grand public.
Une multitude de machines sont commercialisées : l'**ALTAIR 8800** premier ordinateur grand public, sans clavier ni écran, l'**APLLE II** (1977), l'**IBM PC** (1981), le **COMMODORE 64** (1982), le **APPLE MACINTOSH** (1984). Ce dernier est d'ailleurs le premier ordinateur personnel équipé d'une souris et d'une interface graphique.

ALTAIR 8800

APLLE II

IBM PC

COMMODORE 64

APPLE MACINTOSH

■ Langages et système d'exploitation

Après la naissance des premiers compilateurs conçus Grace HOPPER à partir de 1951, **John BACKUS** achève l'élaboration du langage **FORTRAN** en 1956. Il est suivi par **LISP**, **COBOL** et enfin **BASIC** en 1964. Les grands principes des langages de programmation étant formulés, de nombreux langages voient le jour entre les années 1970 et 2000 : le **C** (1972), **ML** (1973) dont est issu CAML, **ADA** (1983) et **C++** en 1986. Le langage **PYTHON** verra le jour quant à lui en 1991 et JAVASCRIPT en 1995.
Au milieu des années 1960, chaque constructeur développe son propre système d'exploitation : **OS 360** puis **MVS** chez **IBM**, le système **UNIX** (1970) chez AT&T…
En 1984 **Richard STALLMAN**, chercheur au MIT entame la création du système **GNU** (**G**NU's **N**ot **U**nix) et promeut le mouvement du logiciel libre.
Par la suite, c'est finalement **MS-DOS**, écrit par Microsoft pour IBM qui s'imposera sur les micro-ordinateurs grand public, suivi par Windows en 1985.
A l'heure actuelle on distingue trois grands types d'OS (Operating System) équipant les ordinateurs modernes : **WINDOWS**, **MAC OS** et **GNU LINUX** créé par **Linus TORVALDS** en 1991.

Source : Wikipédia

Q1. Sur quel type de support externe étaient stockés les instructions que devaient exécuter les tous premiers ordinateurs ?
Q2. Quel composant électronique a remplacé les tubes à vide volumineux et peu fiables ?
Q3. Quel est Le premier langage informatique de haut niveau autre que le langage machine ?
Q4. En quelle année est apparu sur le marché le premier microprocesseur ?
Q5. Quel est le premier ordinateur grand public ayant proposé une souris et une interface graphique ?

2. Architecture de VON NEUMANN

L'architecture des ordinateurs est conforme à un schéma qui a peu évolué depuis son origine en 1945 : le modèle dit de VON NEUMANN.

John VON NEUMANN (1903-1957)**,** était un mathématicien et physicien américano-hongrois. Il a apporté d'importantes contributions tant en mécanique quantique qu'en analyse fonctionnelle, en théorie des ensembles, en informatique, en sciences économiques ainsi que dans beaucoup d'autres domaines des mathématiques et de la physique.

2.1. Organisation générale

Les instructions qui composent les programmes sont exécutées par le **CPU** (**C**entral **P**rocessing **U**nit). Il est schématiquement constitué de 3 parties.

▷ **L'unité arithmétique et logique** (**UAL** ou **ALU** en anglais) 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 contrôle** est chargée du séquençage des opérations : elle 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. Certains registres jouent des rôles particuliers dans le fonctionnement du processeur et portent des noms en conséquence. D'autres registres ont un usage plus général ; dans la suite on nommera ces registres R1, R2, R3…

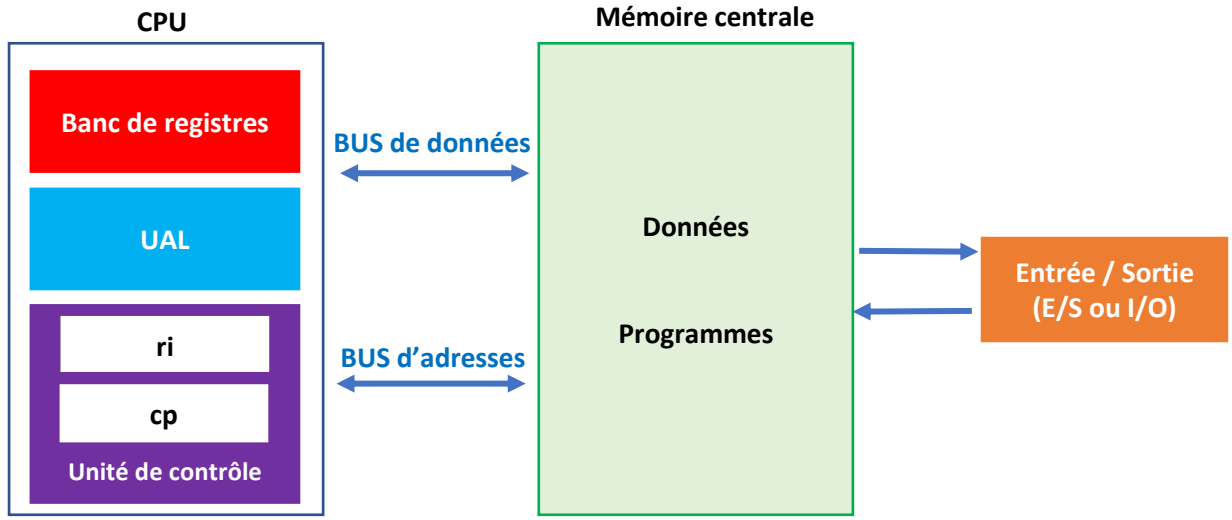
Les données doivent circuler entre les différentes parties d'un ordinateur, notamment entre la mémoire vive et le CPU. Le système permettant cette circulation est appelé **bus**. Il existe, sans entrer dans les détails, 3 grands types de bus :

▷ Le **bus d'adresses** permet de faire circuler des adresses (par exemple l'adresse d'une donnée à aller chercher en mémoire).

▷ Le **bus de données** permet de faire circuler des données.

▷ Le **bus de contrôle** permet de spécifier le type d'action (exemples : écriture d'une donnée en mémoire, lecture d'une donnée en mémoire).

Enfin, les dispositifs d'entrée-sortie permettent de communiquer avec le monde extérieur.



Architecture simplifiée de Von Neumann

2.2. Organisation interne du CPU

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

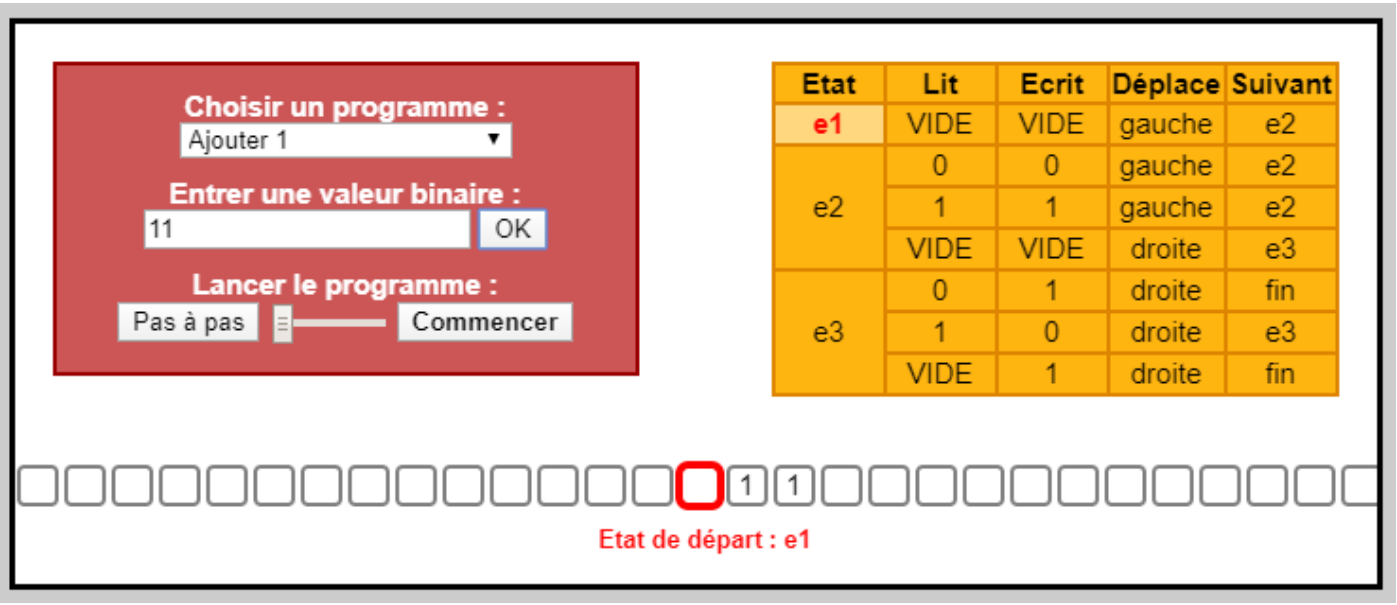
2.3. Le rôle de l'horloge CPU

■ Cadence d'un processeur

Le **CPU** dispose d'une **horloge** qui cadence l'accomplissement des instructions dont l'**unité de temps** est appelée **cycle**. Lorsqu'on l'on parle par exemple d'un processus cadencé à 3 GHz, cela signifie qu'il y a 3 milliards (3×10^9) cycles d'horloge par seconde. Jusqu'en 2004 environ, la fréquence des processeurs a augmenté linéairement selon la loi de MOORE. Depuis elle stagne… En effet, au-delà, la chaleur produite par effet Joule devient trop importante et pourrait perturber la lecture des tensions aux bornes des circuits de l'UAL, voir détériorer physiquement ses circuits électroniques. L'augmentation de la finesse de gravure permet de diminuer l'effet Joule mais on estime qu'en dessous de 4 nm les effets quantiques (effet tunnel notamment) se feront sentir rendant le microprocesseur instable.

■ Cycles d'instructions

Dans une machine de Turing, un cycle représente l'ensemble des quatre actions : lire, écrire, se déplacer, changer d'état. La machine imaginée par Alan TURING comporte un ruban divisé en cases, dans lesquelles elle peut écrire des symboles. La machine ne peut lire qu'une seule case à la fois, de même elle écrit dans une seule case et décale le ruban d'une seule case vers la gauche ou vers la droite. Les symboles sont en nombre fini. Pour que sa machine fonctionne comme une machine à calculer en binaire, Turing envisage le cas particulier où les symboles utilisés sont 0 et 1.



Etat	Lit	Ecrit	Déplace	Suivant
e1	VIDE	VIDE	gauche	e2
e2	0	0	gauche	e2
	1	1	gauche	e2
	VIDE	VIDE	droite	e3
e3	0	1	droite	fin
	1	0	droite	e3
	VIDE	1	droite	fin

Nouvel état : fin

https://interstices.info/comment-fonctionne-une-machine-de-turing/

Alan TURING (1912-1954), était un mathématicien et cryptologue britannique, auteur de nombreux travaux qui fondent scientifiquement l'informatique. Pour résoudre le problème fondamental de la décidabilité en arithmétique, il présente en 1936 une expérience de pensée que l'on nommera ensuite machine de Turing et des concepts de programme et de programmation, qui prendront tout leur sens avec la diffusion des ordinateurs, dans la seconde moitié du XX^{e} siècle.

Dans un processeur, ce que l'on appelle un cycle est un peu différent. En effet, chacune des cinq actions suivantes est exécutée lors d'un cycle : lire l'instruction (LI), décoder l'instruction (DI), exécuter l'opération dans l'UAL (EX), accéder à la mémoire en lecture ou en écriture (M), écrire le résultat dans le registre (ER).
Afin d'optimiser le temps global d'exécution, le processeur n'exécute pas les opérations de manière séquentielle mais exécute simultanément plusieurs instructions qui sont à des étapes différentes de leur traitement : c'est le principe du « pipeline d'instructions ». Le pipeline est un concept s'inspirant du fonctionnement d'une ligne de montage.

Exemple : dans le tableau ci-dessous, on a 5 instructions qui devraient demander 5 x 5 = 25 cycles pour être exécutées séparément. Ici, il en faut seulement 9 (charge maximale du processeur au cycle 5). Cette optimisation peut être ralentie par des branchements ou accélérée par une meilleure organisation de la mémoire.

	Cycles CPU								
Instructions	1	2	3	4	5	6	7	8	9
A	LI	DI	EX	M	ER				
B		LI	DI	EX	M	ER			
C			LI	DI	EX	M	ER		
D				LI	DI	EX	M	ER	
E					LI	DI	EX	M	ER

Q6. Quels sont les principaux éléments d'une architecture de Von Neumann ?
Q7. Quels sont les 3 principaux éléments constitutifs d'un CPU ?
Q8. Quel système permet d'acheminer les données entre le CPU et la mémoire vive ?
Q9. Le nouveau processeur 10 cœurs Intel Core i9 de 9ème génération possède une fréquence de base de 3,30 GHz. Combien ce processeur exécute-t-il de cycles CPU chaque seconde ? Quelle est la durée d'un cycle ?
Q10. Comment sont codées les informations sur le ruban d'une machine de Turing ?
Q11. Quelles sont les actions exécutées par une machine de Turing lors d'un cycle ?
Q12. Dans un processeur moderne, les instructions sont-elles exécutées d'une manière séquentielle, c'est-à-dire les unes après les autres d'une manière séparée comme dans une machine de Turing ?

3. Mémoire et langage machine

Dans son cœur, la machine effectue des calculs à partir d'instructions simples en binaire qui peuvent être directement traduites en langage machine (binaire) qui prend, traite et remplit des cases mémoires.

3.1. Organisation de la mémoire

Il existe de nombreuses technologies de mémoire qui se distinguent par leur durabilité (volatile ou permanente), leur mode d'accès (par adresse ou dans l'ordre de leur rangement) ou leur temps d'accès. En règle générale, plus une mémoire est performante, plus elle est chère.
On parle de **mémoire vive** (ou volatile) quand le contenu est perdu lorsque le courant s'arrête : il s'agit des **registres**, des **mémoires cache**, de la **mémoire centrale**. Les autres mémoires (disque dur, SSD…) sont quant à elles **persistantes**.
Il existe une troisième catégorie de mémoire, la **ROM** (**R**ead **O**nly **M**emory) qui ne fonctionne, en principe, qu'en lecture seule.
Remarque : la mémoire ROM contient notamment le BIOS (Basic Input Output System) qu'il est possible, sur les machines dotés de carte mère récente, de mettre à jour (flashage du BIOS).

3.2. Les registres

Un **registre** est un emplacement mémoire interne au processeur. Il sert à stocker des opérandes et des résultats intermédiaires lors des opérations effectuées dans l'UAL. Leur capacité, leur nombre et leurs rôles varient selon les processeurs. La grande majorité des OC actuels ont des registres de taille 64 bits. Ils sont accessibles via un jeu d'instructions.

3.3. Mémoires centrales et mémoires caches

La **mémoire centrale** est une mémoire vive qui contient les programmes en cours et les données qu'ils manipulent. Elle est de taille importante (plusieurs Go). Elle est organisée en **cellules** appelées « **cases mémoires** » qui contiennent chacune une donnée ou une instruction repérées par une **adresse** qui est un **nombre entier**. Le temps d'accès à chaque cellule est le même : on parle de mémoire à accès aléatoire (**RAM** : **R**andom **A**ccess **M**emory) bien qu'il soit plus judicieux de parler de mémoire à accès direct.
Afin de pouvoir adapter la très grande vitesse du processeur à celle plus faible de la mémoire centrale, on place entre les deux une mémoire plus rapide, la **mémoire cache** qui contient les instructions et les données en cours d'utilisation car, la plupart du temps, les données qui viennent d'être utilisées ont une probabilité plus grande d'être réutilisées que d'autres. La **mémoire cache** (de l'ordre de quelques Mo) est souvent constitué de mémoire de type **statique SRAM** plus rapide mais plus chère que celle de type **RAM dynamique** (**SDRAM**, **DDR** …) utilisée dans la mémoire centrale. Généralement la mémoire cache est intégrée au « socket » du processeur.

Q13. Qu'est-ce qu'un registre ?
Q14. Qu'est-ce que la mémoire vive ? Donner des exemples.
Q15. Quel type de mémoire fait intervenir un disque dur ou un SSD ?
Q16. Quelle(s) différence(s) y-a-t-il entre mémoire centrale et mémoire cache ?
Q17. Comment est repérée une cellule ou case mémoire ?

4. Jeu d'instructions

4.1. Nature des instructions

Un programme écrit dans un langage de haut niveau, c'est-à-dire plus proche du langage humain (exemple : PYTHON) que du langage machine dépend peu du processeur et du type de système d'exploitation utilisé.
En revanche, chaque processeur à son langage qui lui est propre. Une chaîne de production permet de passer de langage de haut niveau au langage machine de bas niveau écrit en binaire. Ce processus fait intervenir un jeu d'instructions spécifiques compris par la machine. Ces jeux ont toutefois des structures communes.
Chaque instruction contient un code correspondant à l'opération à effectuer et aux opérandes. Une opération peut être :

- une opération de transfert entre les registres et la mémoire ;
- une opération de calcul arithmétique ou logique (addition, comparaison…) ;
- un branchement via des sauts vers une certaine adresse selon le résultat de l'op »ration précédente (branchement conditionnel) ou non (branchement inconditionnel).

4.2. Notion d'assembleur

Il est parfois utile de programmer au plus près de la machine. On utilise l'assembleur, un programme qui permet de passer directement du langage machine à un langage dit d'assemblage plus compréhensible pour l'être humain :

- le jeu d'instruction est exactement identique à celui du langage machine ;
- les opérations sont désignées par des instructions mnémoniques comme par exemple add (addition), lw (load word), sw (store word) … ;
- les registres ont des noms : $t0, $t1, $a0 … ;
- les constantes peuvent être exprimées en hexadécimal, en binaire … ;
- les branchements se font via des étiquettes

4.3. 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".

Exemples d'instructions (assembleur 6800) : https://www.gladir.com/CODER/ASM6800/referenceopcode.htm

```
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)

```
STR R3,125
```

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

```
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

```
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

```
SUB R1,R0,#128
```

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

```
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

```
MOV R1, #23
```

Place le nombre 23 dans le registre R1

```
MOV R0, R3
```

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

```
B 45
```

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

```
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)

```
CMP R0, R1
```

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

```
CMP R0, #23
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

```
CMP R0, #23
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

```
CMP R0, #23
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

```
CMP R0, #23
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

```
HALT
```

Arrête l'exécution du programme

Simulateur assembleur (ASM 1802) : http://www.peterhigginson.co.uk/AQA/
Jeu d'instructions ASM 1802 : https://www.gladir.com/CODER/ASM1802/reference.htm

Q18. Donner la signification de l'instruction ci-dessous

```
ADD R0, R1, #42
```

Q19. Donner la signification des instructions ci-dessous.

```
CMP R4, #18
BGT 77
```

Q20. Ecrire le code assembleur permettant d'additionner la valeur stockée dans le registre R0 et la valeur stockée dans le registre R1, le résultat est stocké dans le registre R5.

Q21. Ecrire le code assembleur tel que la prochaine instruction à exécuter se situe en mémoire vive à l'adresse 478 si la valeur stockée dans le registre R0 est égale 42.

Les instructions assembleur de branchements **B** (Branch), **BEQ** (Branch if equal), **BNE** (Branch if non equal), **BGT** (Branch if greater) et **BLT** (Branch if less than) n'utilisent pas directement l'adresse mémoire de la prochaine instruction à exécuter, mais des "labels". Un label correspond à une adresse en mémoire vive. L'utilisation d'un label évite donc d'avoir à manipuler des adresses mémoires en binaire ou en hexadécimale :

```
CMP R4, #18
   BGT monLabel
   MOV R0,#14
   HALT
monLabel:
   MOV R0,#18
   HALT
```

Q22. Donner la signification du code assembleur ci-dessus.
Q23. On donne le code PYTHON et sa traduction en assembleur ci-dessous. A quoi correspond les adresses mémoires 23, 75 et 30 en assembleur ?

```
x = 4
y = 8
if x == 10:
  y = 9
else :
  x=x+1
z=6
```

```
   MOV R0, #4
   STR R0,30
   MOV R0, #8
   STR R0,75
   LDR R0,30
   CMP R0, #10
   BNE else
   MOV R0, #9
   STR R0,75
   B endif
else:
   LDR R0,30
   ADD R0, R0, #1
   STR R0,30
endif:
   MOV R0, #6
   STR R0,23
   HALT
```