الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERHCE SCIENTIFIQUE

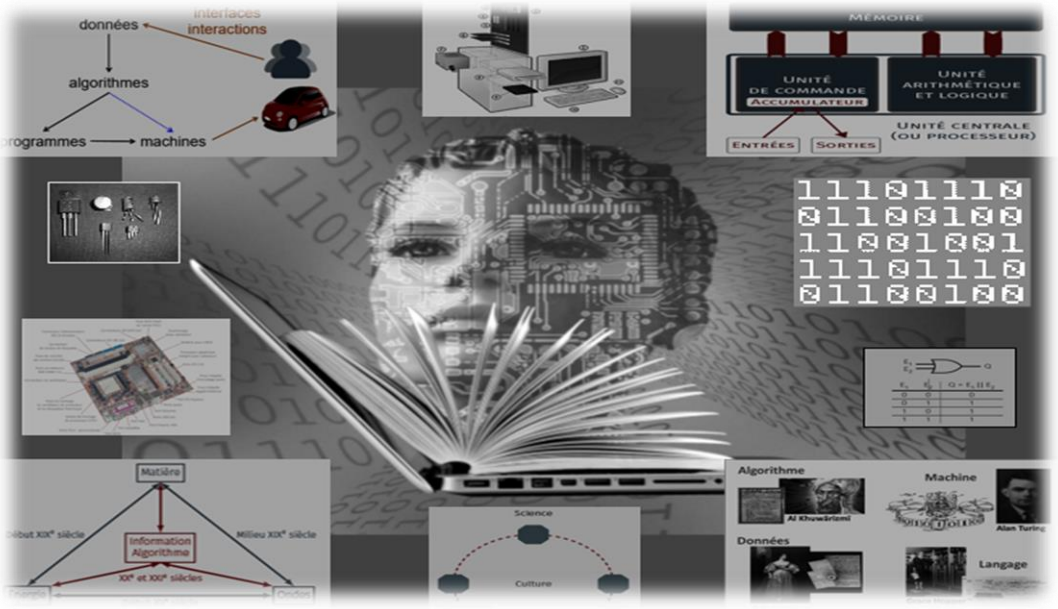
Ecole Normale Supérieure de
l'Enseignement Technologique de Skikda
Département Mathématiques et Informatique

المدرسة العليا لأساتذة التعليم
التكنولوجي –سكيكدة–
قسم الرياضيات والاعلام الآلي

ARCHITECTURE DES ORDINATEURS

Polycopié

Dr. SELLAMI Samir



1ère Année Prof. d'Enseignement Moyen

ENSET-SKIKDA

Année Universitaire 2022/2023

Architecture des Ordinateurs

Polycopié

Domain Mathématiques et Informatique

Filière Informatique

Information sur la matière:

Intitulé : Architecture des Ordinateurs-1
Niveau: 1ère Année profile PEM
V.H.H: 03H00 (Cours + TD)
Coefficient: 04
A/S : Annuel

Dr. SELLAMI Samir

Avant-propos

Ce document est basé sur un enseignement donné depuis deux ans aux étudiants de 1ère première année en enseignement de l'informatique à l'Ecole Normale Supérieure de l'Enseignement Technologique de Skikda (ENSET-Skikda). Il rassemble les concepts clés de *l'architecture des ordinateurs* sous forme de notes et de mots-clés faciles à retenir pour les étudiants. Ces notes de cours résument brièvement les points importants abordés en classe.

L'objectif de cette matière est de faire découvrir aux étudiants de 1ère première année les principes de bases qui régissent l'ensemble des composantes des ordinateurs existantes (architecture de Von Neumann, décomposition en couches du niveau matériel jusqu'à la programmation haut-niveau, circuits logiques et langage d'assemblage) et leur impact sur l'environnement informatique, et plus généralement de faire découvrir l'architecture des ordinateurs à tous ceux qui s'intéressent à ce domaine.

Ce recueil vise à aider les étudiants à mieux comprendre les concepts vus en cours et en travaux dirigés (TD) en utilisant des photographies et des schémas illustratifs. Il est organisé en cinq chapitres qui couvrent l'intégralité du programme de la matière.

A la fin de chaque chapitre du document, des séries d'exercices sont proposées pour les travaux dirigés. Ces exercices peuvent également être utilisés comme support de soutient pour les étudiants lors de la révision lors des contrôles d'évaluation.

L'auteur remercie chaleureusement ses collègues, le Dr. BOUDJAADAR Djamel, maître de conférences et directeur de l'ENSET-Skikda, Pr. ZAROUR Nacer Eddine, professeur à l'université de Constantine 2 - Abdelhamid Mehri et Dr. KISSOUM Yacine maître de conférences à l'université 20 août 1955 - Skikda pour leur expertise de ce travail et exprime sa profonde reconnaissance pour tous leurs commentaires et suggestions pertinents visant à améliorer la qualité de ce document.

Table des matières

ARCHITECTURE DES ORDINATEURS

Introduction

L'informatique, qui est l'une des principales disciplines intégrées à l'enseignement, est une science centrée sur l'ordinateur. Les futurs enseignants spécialistes en informatique doivent maîtriser les fonctionnements de base d'un ordinateur pour pouvoir enseigner efficacement et améliorer les compétences des générations futures.

L'architecture des ordinateurs est une discipline qui permet d'étudier les différents composants internes des machines, expliquant leur construction et leurs interactions. Un ordinateur est un instrument complexe capable d'effectuer diverses tâches et dont les performances globales dépendent des spécifications de tous ses composants.

Comprendre la manière dont les différents composants d'un ordinateur interagissent entre eux permet de savoir comment les caractéristiques spécifiques de chaque composant influencent la performance de la machine en fonction de son utilisation: par ex. comment l'ajout de mémoire peut-il accélérer l'ordinateur, pourquoi le temps d'accès d'un disque dur n'est-il qu'un des paramètres pour mesurer son efficacité, comment les processeurs deviennent-ils de plus en plus rapides ?

L'objectif principal de ce cours est de comprendre les composants de base d'un ordinateur et le schéma fonctionnel qui les relie. A la fin de ce cours, les étudiants seront capables de lister tous les composants de base d'un ordinateur, de distinguer les fonctions de chaque composant, et de facilement identifier les causes de problèmes de performance et proposer des solutions matérielles appropriées.

Ce document est organisé comme suit :

Le chapitre I regroupe des notions générales sur la structure et les composantes d'un ordinateur en présentant un petit historique sur l'évolution des ordinateurs depuis leurs apparitions jusqu'aux modèles actuels. (Le volume horaire allouée V.H: 4,5 h / [02] Cours et [01] TD / 6 % du programme).

Dans le chapitre II, une brève description des mémoires est donnée pour permettre de bien comprendre les caractéristiques, la structure et le fonctionnement de la mémoire centrale d'un ordinateur.
(Le V.H: 6 h / [02] Cours et [02] TD / 10 % du programme).

Le chapitre III, aborde le composant maître de l'ordinateur à savoir le processeur. Ce chapitre décrit la structure interne du microprocesseur et les mécanismes de fonctionnement et de communication des différents composants. (Le V.H: 9 h / [03] Cours et [03] TD / 14 % du programme).

Le chapitre IV traite la manière dont l'information est représentée et codée dans un ordinateur, en se concentrant notamment sur le codage binaire des nombres et leurs arithmétiques, ainsi que sur la façon dont d'autres types d'informations tels que les caractères, les images et les sons sont encodés. (Le V.H: 14 h / [07] Cours et [07] TD / 32% du programme).

Le dernier chapitre, chapitre V, est un rappel des principes fondamentaux de la logique combinatoire et séquentielle, sur lesquels reposent la conception et le fonctionnement de la plupart des composants de l'ordinateur. Ce rappel permet à l'étudiant de mieux comprendre l'architecture interne de l'ordinateur et son mode de fonctionnement, en se concentrant sur les aspects logiques et électroniques des différents composants. (Le V.H: 16 h / [08] Cours et [08] TD / 38% du programme).

Chapitre I : Généralités sur l'ordinateur

1. *Histoire des ordinateurs (machines ancêtres)*

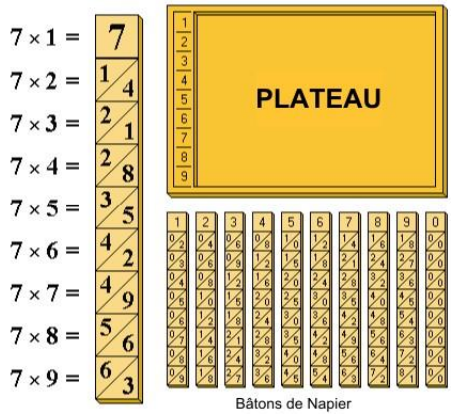
L'ordinateur est un produit de l'effort de plusieurs ingénieurs et théoriciens au cours de la Seconde Guerre mondiale. Il n'y avait pas un seul lieu de développement, mais plutôt plusieurs centres indépendants, chacun a travaillé à la construction de machines qui n'existaient pas à l'époque [1].

L'évolution de l'ordinateur actuel a traversé plusieurs étapes étroitement liées aux avancées technologiques, ces étapes sont généralement regroupées en différentes générations d'ordinateurs [2].

a. *La génération zéro : les calculateurs mécaniques (Avant 1945)*

► Les abaques (avant 1600):

- Instruments mécaniques facilitant le calcul.



Bâtons de Napier

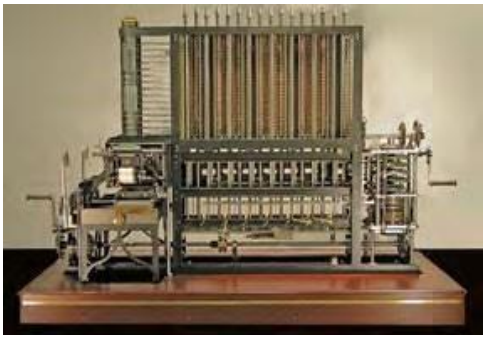
► La Pascaline (1642):

- Inventée par Blaise Pascal.
- Machine qui additionne et soustrait les nombres de 6 chiffres en base 10.
- Les multiplication et divisions se faisaient par répétitions.
- Première machine à calculer!

► Le métier Jacquard (1805):

- Métier à tisser de Joseph Jacquard.
- Créé d'après des idées de Falcon en 1728.
- Système mécanique programmable.
- Utilise des cartes perforées pour métiers à tisser.
- C'est le **1er programme** !

► La machine analytique de Charles Babbage (1833):



- Machine programmable.
- Capable de réaliser différentes opérations codées sur des cartes perforées.
- Dotée d'un dispositif d'entrées et sorties.
- Un organe de commande gérant le transfert des nombres et leur mise en ordre pour le traitement.
- Un magasin permettant de stocker les résultats intermédiaires ou finaux (mémoire).
- Un moulin chargée d'exécuter les opérations sur les nombres.
- Un dispositif d'impression.

b. La première génération des ordinateurs: les tubes à vide (1945 - 1955)

- La Seconde Guerre Mondiale précipite l'avènement des ordinateurs.
- Les sous-marins allemands communiquaient par radio → interception facile.
- Les messages étaient chiffrés avec une machine **ENIGMA** [3], volée aux allemands.
- Besoin de beaucoup de calculs, rapidement pour les décrypter → création du premier ordinateur électronique : le **COLOSSUS** [4].

↓

➢ Besoins de l'armée américaine pour le réglage des tirs d'artillerie.

➢ La course aux calculateurs est lancée à travers le monde !

► Les tubes à vide:

- Également appelés tubes électroniques ou même lampes.
- C'est des amplificateurs de signal.
- Un ensemble d'électrodes placées dans le vide ou dans un gaz.
- C'est une source d'électrons.
- Remplacés plus tard par des semi-conducteurs.

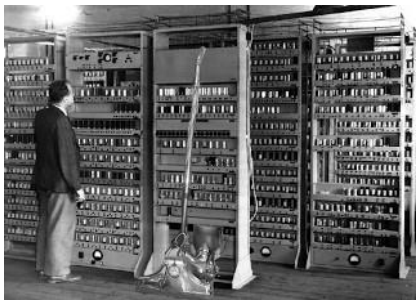
► L'ENIAC de John Mauchly (1946):

- **E**lectronic **N**umerical **I**ntegrator **A**nd **C**omputer.
- 1er ordinateur électronique Turing-complet [1].
- Calculs en système décimal.
- Composé de 18 000 tubes à vide et 1500 relais.
- 6000 commutateurs et une forêt de câbles.
- Pèse 30 tonnes, et occupe 167 m^2.
- Incapable d'enregistrer un programme.

[1] Turing-complet désigne en informatique un système formel ayant au moins le pouvoir des machines d'Alan Turing (machine universelle qui peut exécuter n'importe quel programme).

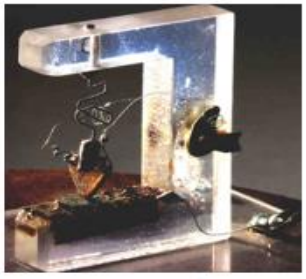
► L'EDSAC de Von Neumann, Eckert et Mauchly (1946):

- Basée sur l'ordinateur EDVAC (**E**lectronic **D**iscrete **V**ariable **A**utomatic **C**omputer).
- Utilise un système **binaire**.
- Les opérations d'addition +, soustraction - et multiplication × étaient automatiques.
- La division ÷ était programmable.
- Capacité mémoire initiale : 1000 mots de 44 bits.

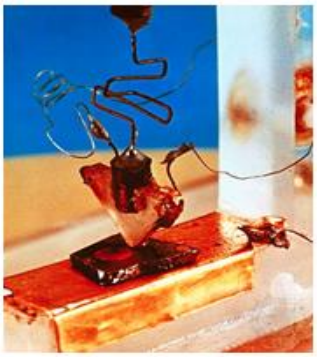


c. La deuxième génération des ordinateurs: les transistors (1955 - 1965)

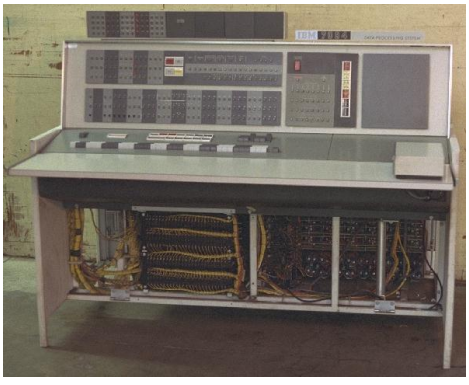
- Le transistor a été inventé en 1948, aux Bell Labs par John Bardeen, William Bradford Shockley et Walter Houser Brattain (prix Nobel de Physique en 1956).
- Les ordinateurs à tubes à vide deviennent obsolètes à la fin des années 50's.
- Le MIT (Massachusetts Institute of Technology) est précurseur avec le TX-0 [5].
- Les ordinateurs deviennent assez stables pour être vendus à des clients : naissance de l'industrie de la mini-informatique (IBM, DEC, HP, etc.).
- Premier jeu vidéo avec le PDP-1: Spacewar !
- Apparition des OS (Operating System) et langages évolués (FORTRAN et COBOL).



semi-conducteurs
Silicium-germanium



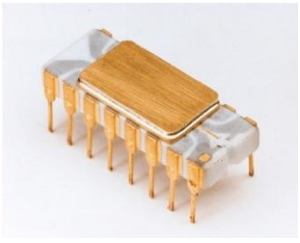
- L'IBM 7094 (1962) est l'équivalent « transistorisée » de l'IBM 704 qui a été basé sur les tubes à vide.
- Nous passons de 5 kFLOPS (Floating-point Operations Per Second) à 200 kFLOPS.



d. La troisième génération des ordinateurs: les circuits intégrés (1965 - 1973)

- Le circuit intégré a été inventé par Jack KILBY en 1958.
- Des dizaines de transistors sur une seule puce.
- L'intégration a permis des ordinateurs plus petits, plus rapides et moins chers.
- Apparition de la multiprogrammation (plusieurs programmes exécutés en même temps sur la même machine).
- IBM lance le 360 en 1964: Cette machine est une des première à utiliser les circuits intégrés et elle introduit également un nouveau concept : la compatibilité entre les familles d'ordinateurs.

e. La quatrième génération des ordinateurs: les microprocesseurs (1971 - 1980)



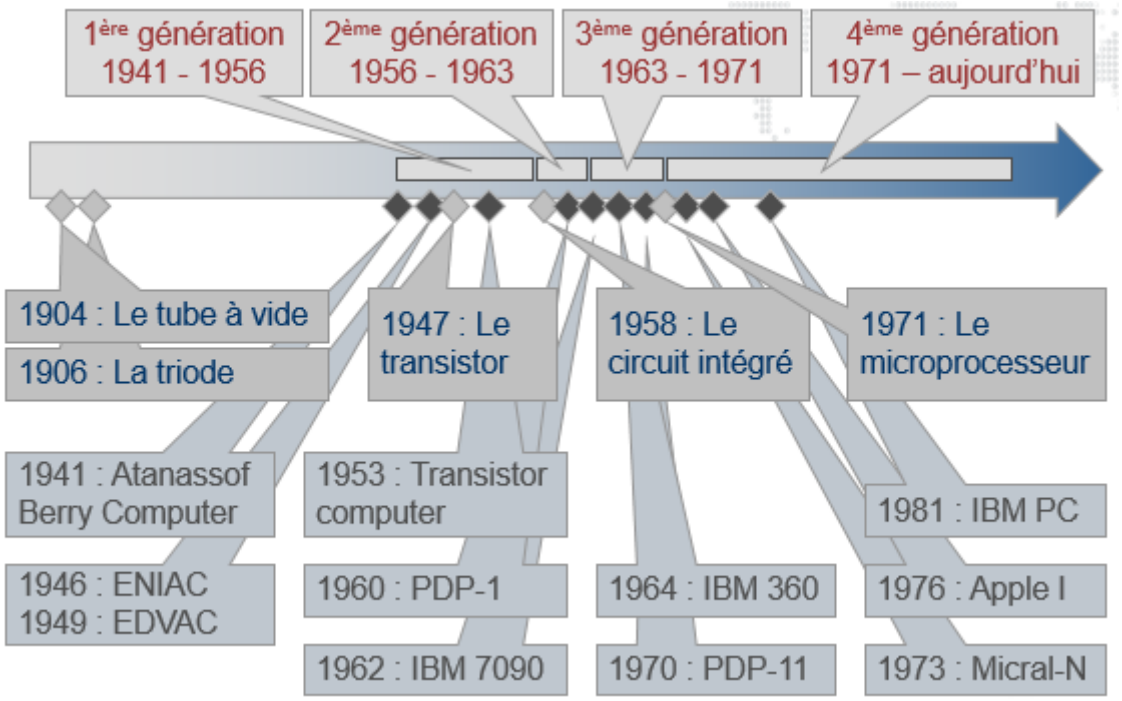
- Miniaturisation des circuits : l'ère de la micro-informatique.
- Evolution de l'intégration avec la technologie VLSI (**V**ery **L**arge **S**cale **I**ntegration).
- Ted HOFF met au point le premier microprocesseur, baptisé 4004, en 1971, pour le compte de la start-up Intel.
- André TRUONG et François GERNELLE construisent le premier micro-ordinateur (le Micral N) en 1973.

- Steve WOZNIAK construit l'Apple I et Steve JOBS propose de le commercialiser en 1976 pour 666,66 $.
- Création en 1977 d'Apple Computer, Inc. qui connaît des succès commerciaux avec l'Apple II et le Macintosh.

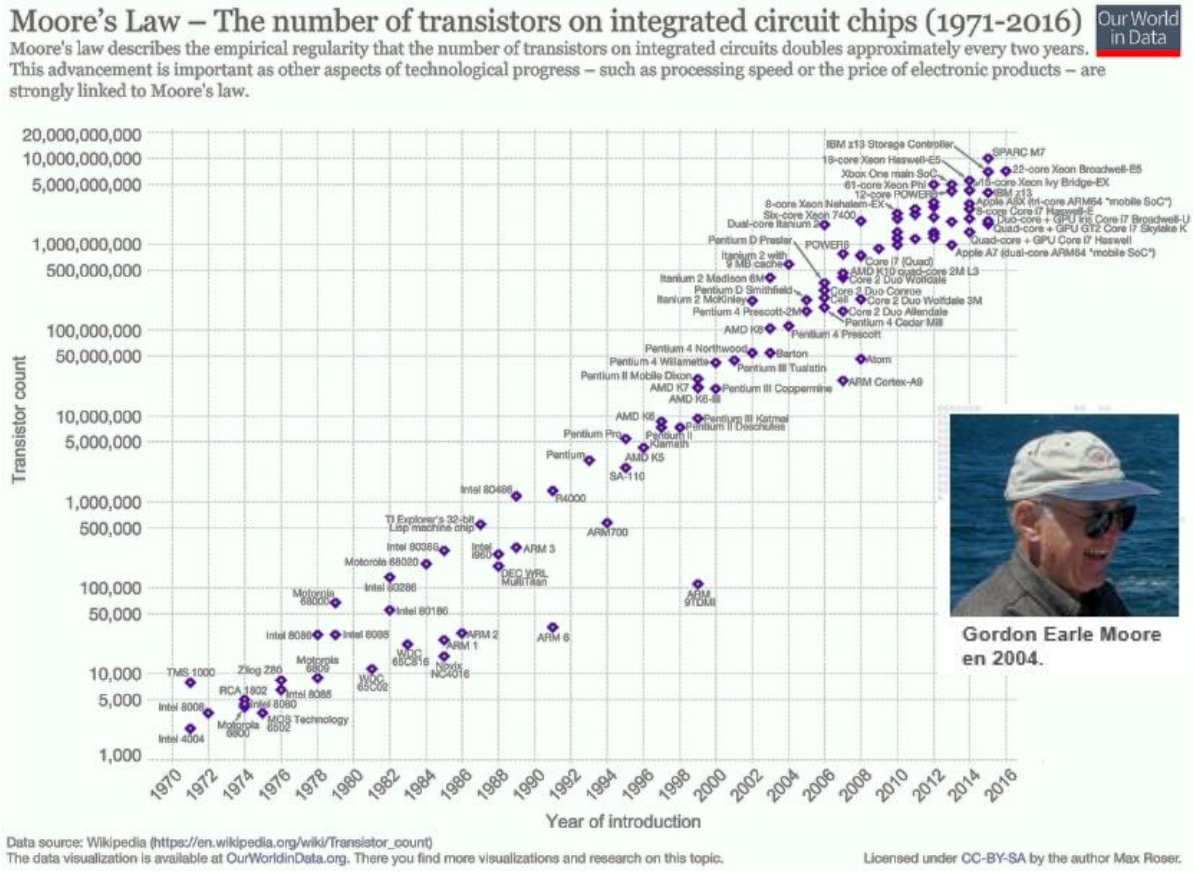
- IBM ne croit pas trop au marché des micro-ordinateurs, mais elle décide, cependant, de construire un ordinateur "bas de gamme" pour être présent sur le marché.
- IBM commercialise le PC en 1981 tout en rendant publique les spécifications de cette machine, ce qui permet le développement des clones et du marché des PC.

- L'ordinateur s'invite dans les foyers grâce aux jeux vidéo avec les consoles d'Atari et de Nintendo, et plus tard avec des ordinateurs comme l'Amiga.

- **Synthèse:** quelques points de repère historiques :



f. Le numérique d'aujourd'hui: la loi de Moore [2] est encore d'actualité

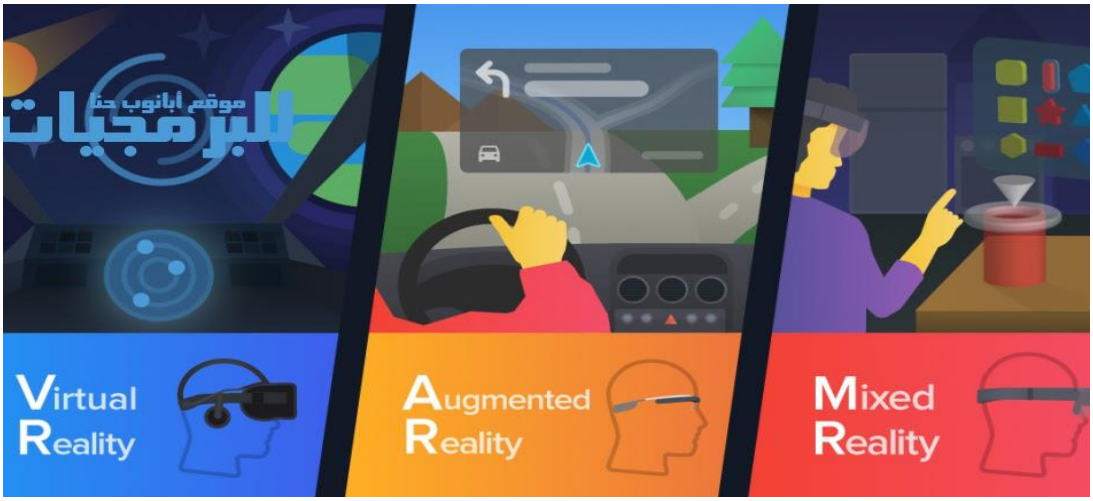


[2] Émise par l'ingénieur Gordon E. Moore en 1965, montrant l'évolution de la puissance de calcul des ordinateurs en fonction de la complexité du matériel informatique.

Actuellement, les ordinateurs font partie intégrante de nos vies quotidiennes :

- Plus généralement, le numérique envahit les appareils du quotidien (les téléphones, de la machine à laver au magnétoscope en passant par la cafetière programmable).
- Nous commençons à parler de domotique (du latin *domus* signifiant maison) et de maison intelligente.

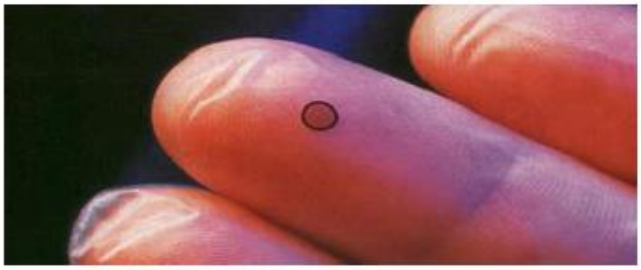
- Réalité virtuelle, augmentée et mixte.



- Mainframe : Traitement par lot dans une banque.
- Les super ordinateurs dans le monde (Prévisions météo à long terme).

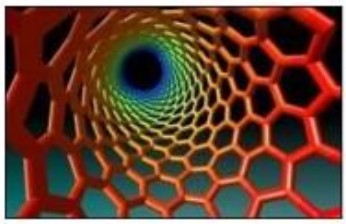
g. Le numérique de demain: L'avenir dans les nanotechnologies

- Richard FEYNMAN prédisait en 1959 qu'il serait possible de stocker les 24 volumes de l'encyclopédie Britannica dans une tête d'épingle [6].
- Ce fut chose faite dans les années 80.

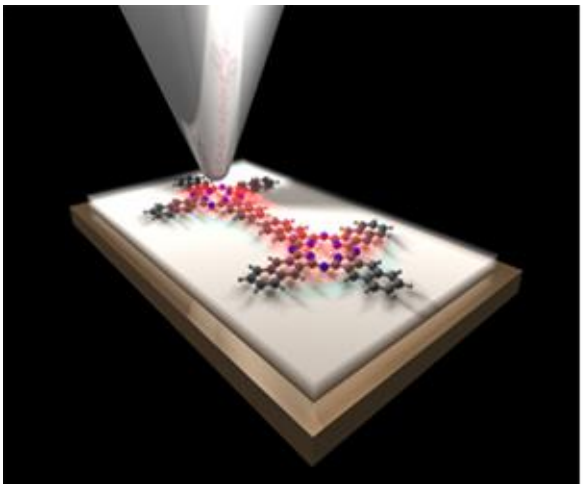


- Stocker 20 fois plus de données par millimètre carré que sur les supports magnétiques actuels grâce à des minuscules aiguilles qui déforme temporairement une plaque de molécules préalablement chauffée.
- 500 Mo de données sur une puce carrée de 3 mm de côté.
- Des densités de stockage plusieurs milliers de fois supérieures à celles d'aujourd'hui seraient envisageables.

- 8 fois plus denses que les circuits actuels car la matrice de routage (crossbar switch), placée au-dessus des blocs logiques (contrairement aux FPGA actuels) [7].
- Moins gourmands en énergie car utilisation de nano-câble pour conduire l'électricité.
- La technologie FPNI (Field Programmable Nanowire InterConnect) : une interconnexion de nano fils programmable qui permet une famille d'architectures de circuits hybrides nano/CMOS qui généralise l'approche hybride CMOS/moléculaire [8].



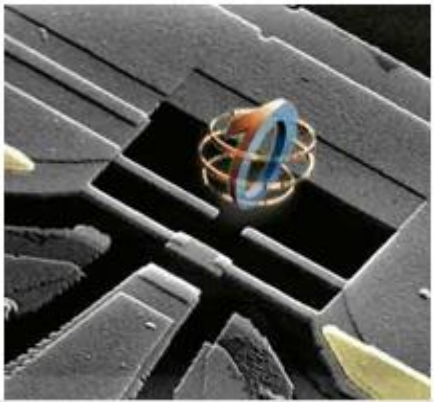
- IBM implante des portes logiques OU et ET 260000 fois plus petits que ceux des puces à semi-conducteur actuelles les plus perfectionnées en utilisant le déplacement en cascade des molécules comme dans un jeu de domino (http://www.research.ibm.com/)



IBM's SINGLE-MOLECULE LOGIC SWITCH
Schematic three-dimensional image of a molecular "logic gate" of two naphthalocyanine molecules, which are probed by the tip of the low-temperature scanning tunneling microscope. By inducing a voltage pulse through the tip to the molecule underneath the tip (shown in the back), the two hydrogen atoms in the adjacent molecule (in white at the center of the molecule in front) change position and electrically switch the entire molecule from "on" to "off".

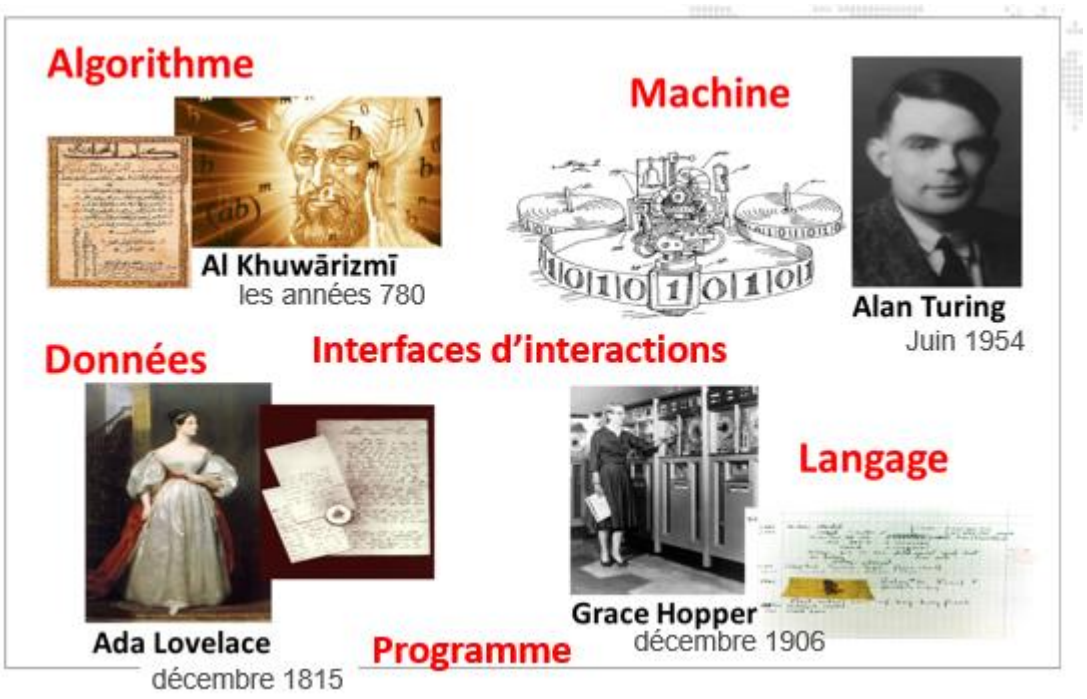
Source: IBM

- Le CEA travaille à la mise au point d'un « **qubit** », l'équivalent quantique d'un bit, permettant de stocker simultanément une "superposition" de 0 et de 1 [9].
- Ce qubit devrait permettre de concevoir des processeurs plusieurs milliers de fois plus rapides dans 15 ou 20 ans (http://oxfordquantum.org/).



2. *Les piliers de l'informatique*

La science informatique a des concepts bien définis :



a. *Définitions*

Les données: représentent sous une forme numérique unifiée des informations très diverses : textes, images, sons, mesures physiques, sommes d'argent, etc.;

Les algorithmes: qui spécifient de façon abstraite et précise des traitements à effectuer sur les données à partir d'instructions élémentaires;

Instruction: action à effectuer par la machine, correspondant à une étape dans un programme.

Programme: suite d'instructions décrivant la façon dont la machine doit effectuer un travail.

Les langages: qui permettent de traduire les algorithmes en programmes textuels ou graphiques de façon à ce qu'ils soient exécutables par les machines;

Les machines: et leurs systèmes d'exploitation, qui permettent d'exécuter des programmes, d'assurer le stockage des données, et de gérer les communications, y compris les objets connectés et les réseaux;

Les interfaces: qui permettent la communication avec les humains, la collecte des données et la commande des systèmes;

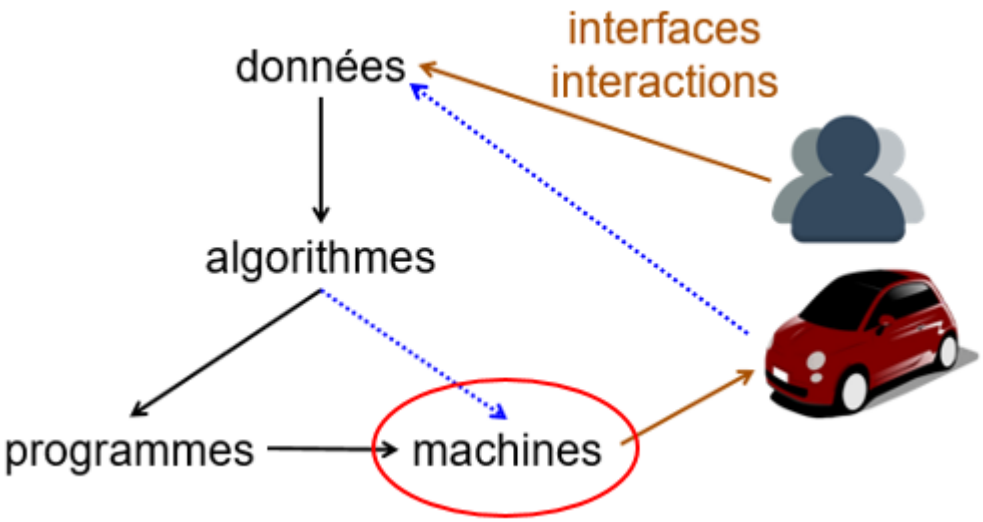
► Les instructions peuvent être classifiées en 4 catégories :

- *Les Instructions d'affectations :* permet de faire le transfert des données ;
- *Les instructions arithmétiques et logiques ;*
- *Les Instructions de branchement* (conditionnelle et inconditionnelle) ;
- *Les Instructions d'entrées-sorties.*

► Etapes d'exécution d'un programme:

1. **Édition,** | print('Hello');
2. **Compilation,** | 01001010100010101
3. **Chargement dans la mémoire,**
4. **Exécuter,** | Hello

► Relations entre les concepts de l'informatique :



► Les clefs de la pensée informatique:

- L'information est la même partout :
  - Une seule notion d'information en médias, télécoms, physique, biologie, neurologie, histoire, etc.
  - Une seule notion d'algorithme pour tous les domaines.
  - Une machine universelle, unique dans l'histoire *(la machine de Von Neumann).*
- Le levier de l'information est hyper-efficace :
  - Textes, musiques, hôtels, voitures → information.
  - Posséder l'information → posséder l'hôtel ou la voiture.
- Mais une difficulté mentale majeure :
  - Le raisonnement et l'action sur l'information sont très différents de ceux sur la matière ou l'énergie.

Comprendre l'essence de l'informatique
est essentiel pour la plupart des activités de demain

b. Architecture de John Von Neumann

Cette architecture est appelée ainsi en référence au mathématicien John Von Neumann qui a élaboré en juin 1945 dans le cadre du projet EDVAC1 [10] a première description d'un ordinateur dont le programme est stocké dans sa mémoire [11].

La plupart des ordinateurs modernes utilisent cette architecture, seules les technologies ont changés.

Cette architecture est basée sur 4 parties principales :

- **Unité arithmétique et logique** (UAL) : effectue les opérations de base.
- **Unité de contrôle** (UC): chargée du séquençage des opérations.
- **Mémoire central** (MC) : contient les données et le programme ○ Mémoire vive ○ Mémoire de masse.
- **Entrées/Sorties** (E/S): permettent de communiquer avec le monde extérieur.

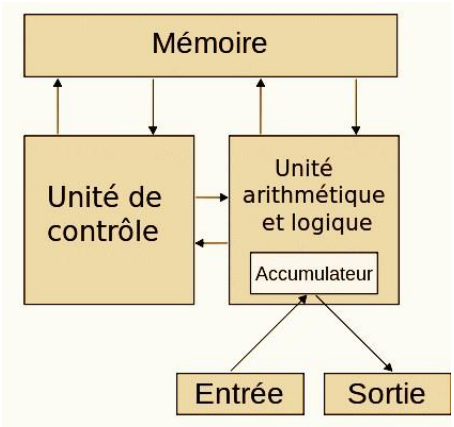


Figure 1. Schématisation de l'architecture de Von Neumann [12].

c. Architecture en couches

L'architecture matérielle et logicielle d'un ordinateur peut être représentée sous forme de couches superposées :

Niveau 5: Couche des langages d'application (langages haut niveau)

Niveau 4: Couche du langage d'assemblage

Niveau 3: Couche du système d'exploitation

Niveau 2: Couche architecture du jeu d'instructions (propre à chaque machine)

Niveau 1: Couche microarchitecture (UAL, opérations, registres, etc.)

Niveau 0: Couche logique numérique (circuits logiques)

d. L'ordinateur du point de vue interne

- Machine électronique binaire.
- Fonctionnement des composants de base : circuits électroniques.
- Organisation et communication entre les composants.
- Utilise un langage machine (système binaire).
- Munie d'un système d'exploitation :
    - Programme principal de l'ordinateur ;
    - Permet l'exécution simultanée d'autres programmes ;
    - Gère des périphériques : entrées/sorties, stockage.

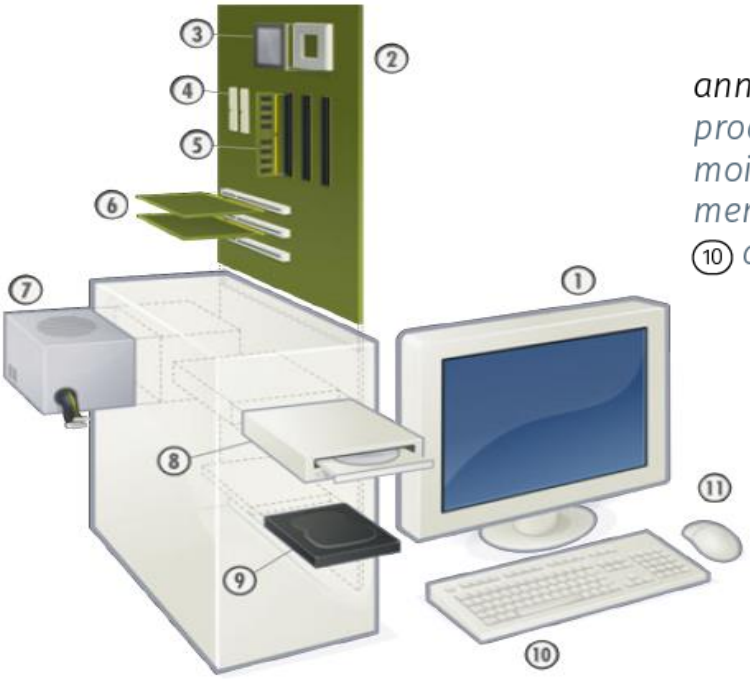
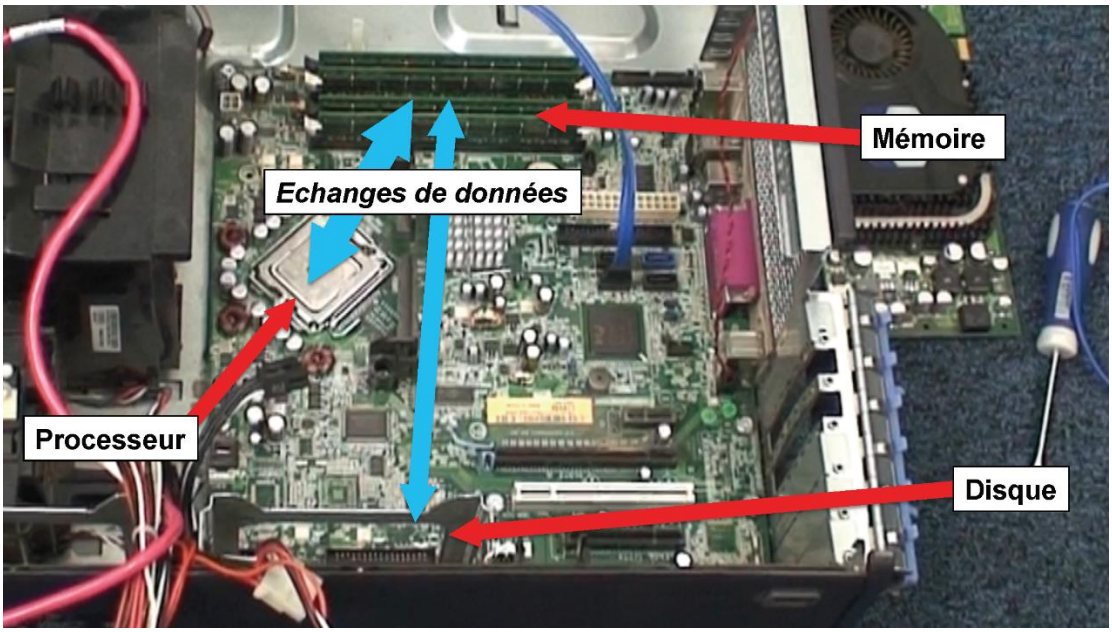


Figure 2. Eclaté d'un PC au milieu des années 2000 [12].

3. Les composants élémentaires d'un ordinateur

L'unité centrale de l'intérieur - échanges de données entre Processeur, Mémoire et Disques:



a. La carte mère

La carte mère est un circuit imprimé qui permet de mettre en contact physique les différents composants et périphériques.

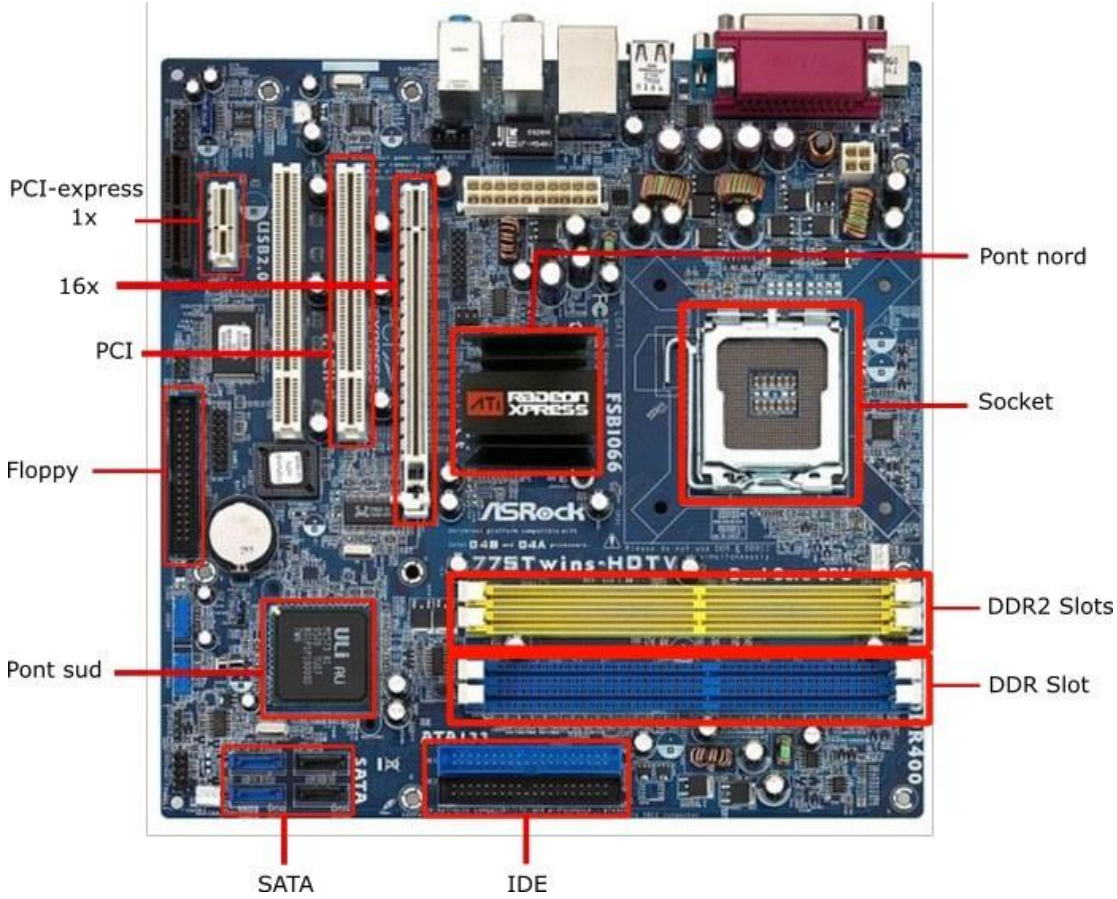


Figure 3 : Composants de la carte mère [13].

b. Le chipset

Composé de 2 parties :

Le pont nord (Northbridge) : chargé de gérer les composants qui ont besoin d'une bande passante importante :

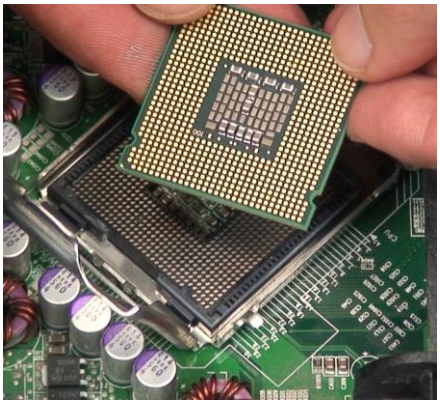
- Microprocesseur ;
- Mémoire vive (RAM) ;
- Carte graphique.

Le pont sud (Southbridge) : chargé de gérer les périphériques qui ont besoin d'une faible bande passante :

- Clavier, souris, audio ;
- Port parallèle, port série ;
- Périphériques USB, FireWire ;
- Réseau ;
- Disques durs, CD/DVD Rom ;

Figure 4 : Vue schématique d'une carte mère de type Intel avec chipset Sis 656fx/966, (2006) [13].

c. *Le processeur*

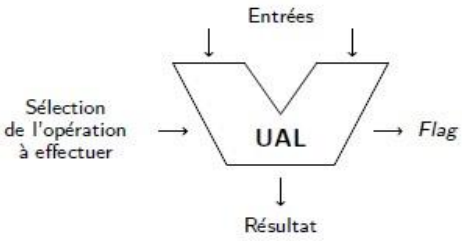


- En anglais CPU (**C**entral **P**rocessing **U**nit) ou UC (**U**nité **C**entrale) en français.
- C'est le « Cerveau » de l'ordinateur.
- Tous les calculs sont exécutés dans le Processeur.
- Exécute les programmes stockés en mémoire principale.
- Plusieurs milliards d'opérations par seconde
- Il est responsable du :
  - Chargement des instructions ;
  - Décodage des instructions ;
  - Exécution des instructions, l'une après l'autre ;

Il est composé de :

- **Unité Arithmétique et Logique** (UAL) :
  - En anglais ALU (**A**rithmetic and **L**ogical **U**nit).
  - Responsable des opérations indiquées par les instructions.
- **Unité de commande :**
  - Récupère les instructions présentes en mémoire principale.
  - Décode les instructions.
- **Les registres :**
  - Petites zones mémoires.
  - Peuvent être lus ou écrits extrêmement rapidement.
- **Les bus :** interconnectent les éléments fonctionnels internes.

1. L'Unité Arithmétique et Logique (UAL)



- Responsable des calculs sur des nombres entiers.
- Opérations communes :
  - **Opérations arithmétiques** : addition, soustraction, changement de signe, etc.
  - **Opérations logiques** : compléments, et, ou, ou-exclusif, non, non-et, non-ou, etc.
  - **Comparaisons** : test d'égalité, supérieur, inférieur, etc.
  - **Décalages** : des bits à gauche, à droite, etc.

2. Les registres communs :

- **Compteur ordinal (CO)** : contient l'adresse mémoire de l'instruction en cours d'exécution ou prochainement exécutée.
- **Accumulateur (ACC)** : pour stocker les données en cours de traitement par l'UAL.
- **Registre d'instructions (RI)** : contient l'instruction en cours de traitement.
- **Pointeur(s) de pile (SP)** : (**S**tack **P**ointer) contient l'adresse du sommet de la pile [3].
- **Registres généraux (R_0, …, R_n)** : registres de stockage intermédiaire pour les calculs.

3. Exécution d'une instruction :

L'exécution d'une instruction par le processeur passe par les étapes suivantes :

1) Charger la prochaine instruction à exécuter dans le registre instruction (RI) ;
2) Modifier le compteur ordinal (CO) pour qu'il pointe sur l'instruction suivante ;
3) Décoder (analyser) l'instruction chargée ;
4) Localiser en mémoire d'éventuelles données nécessaires à l'instruction ;
5) Charger, si nécessaire, les données dans les registres généraux ;
6) Exécuter l'instruction ;
7) Recommencer à l'étape 1.

d. Le disk dure

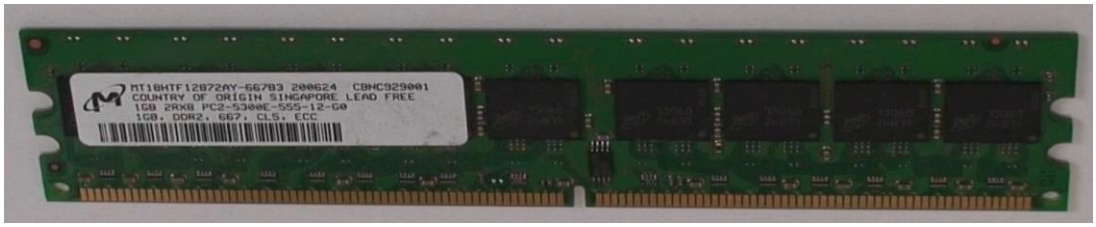
- Le processeur utilise des données en entrée et génère des données en sortie (ex. 2 + 3 = 5).
- Où stocker ces données ?
- Initialement elles sont souvent sur le disque dur
  - Image à afficher, fichier texte à modifier, programme à lancer, etc.
- À la fin, le résultat retourne souvent sur le disque dur.
  - Image modifiée, fichier texte complété.

[3] Une pile est une zone de la mémoire gérée par le processeur selon le principe « dernier arrivé, premier sorti » (en anglais LIFO pour last in, first out).

e. La mémoire principale

À quoi sert la mémoire RAM ?

- Le disque dur est très lent :
  - Le processeur exécute une opération toutes les nanosecondes environ.
  - Le disque réagit **beaucoup** moins vite.
- La mémoire RAM sert d'intermédiaire :
  - Seules les données initiales et finales sont sur le disque dur.
  - Les résultats intermédiaires restent en mémoire.
- La **RAM** est **plus petite**, mais **beaucoup plus rapide.**

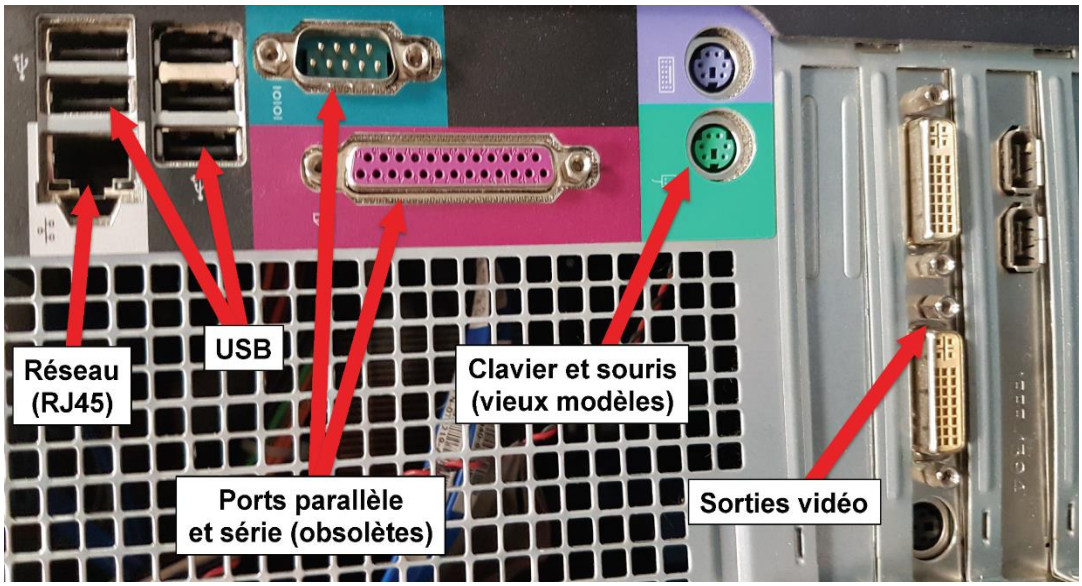


f. Les Entrées/Sorties

Communiquer avec l'extérieur ?

- Interagir avec l'utilisateur ?
- Communiquer avec les autres ordinateurs ?
- Des ***Périphériques*** connectés au processeur et à la mémoire :
  - Écran, imprimante, casque de réalité virtuelle, haut-parleur, … pour la **sortie.**
  - Souris, clavier, joystick, micro, tactile, caméra, capteurs, … pour **l'entrée.**
  - Carte réseau filaire, wifi, antenne, … pour les **communications.**
- Le processeur lit et/ou écrit dans ces périphériques.

► Quelques ports de périphériques :



► Quelques exemples de périphériques :

Périphérique	*Entrée*	*Sortie*	*Stockage*	*Traitement*
Clavier	●			
Souris	●			
Ecran		●		
Ecran tactile	●	●		
CD/DVD			●	
Lecteur-Graveur CD/DVD	●	●		
Webcam	●			
Imprimante		●		
Carte réseau				●
Microphone	●			
Enceinte	●	●		
Scanner	●			
Disque dur			●	
Carte son				●

g. Les bus

- Canaux de communication à l'intérieur de l'ordinateur.
- Relient les différents composants de l'ordinateur.
- Caractérisés par :
  - **Un type** : parallèle ou série.
  - **Une largeur** : nombre de bits que le bus peut transmettre à la fois.
  - **Une fréquence** (vitesse) : nombre de paquets envoyées par seconde (en Hz).
  - **Une bande passante** (débit) = largeur × fréquence.

Il y a 3 types de bus :

- **Bus de données** : définit la taille des données pour les E/S.
- **Bus d'adresse** : permet l'adressage de la mémoire.
- **Bus de contrôle** : permet la gestion du matériel, via les interruptions (commandes).

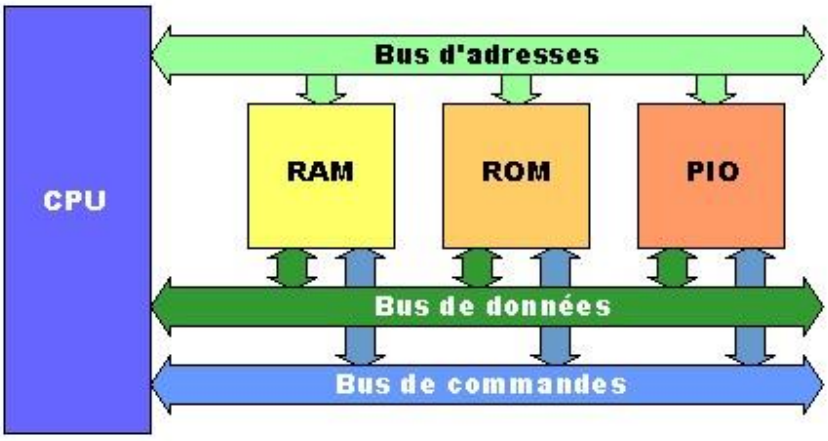


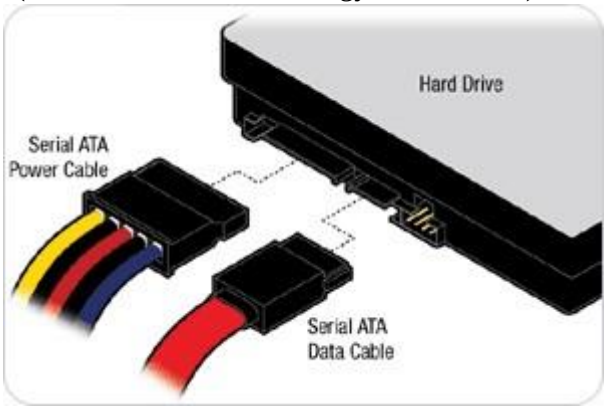
Figure 5 : Schéma des bus de communication [13].

► La connectique :

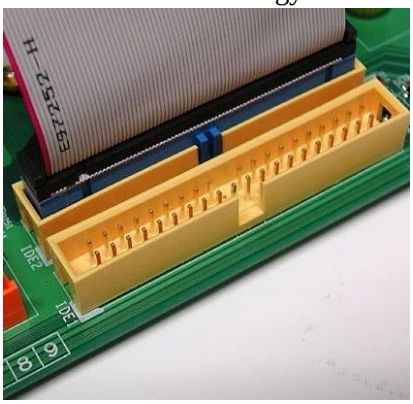
Les composants de l'unité centrales sont connectés à la carte mère via des ports spécifiques.

- Ports pour disques dur et lecteurs CD/DVD :

Ports SATA :
(**S**erial **A**dvanced **T**echnology **A**ttachement)

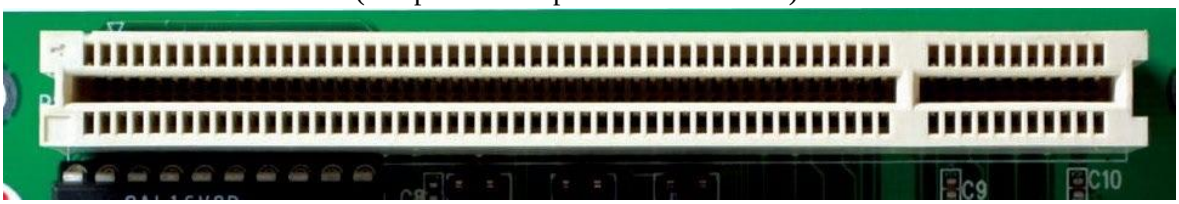


Port IDE (PATA) :
(**P**aralle **A**dvanced **T**echnology **A**ttachement)

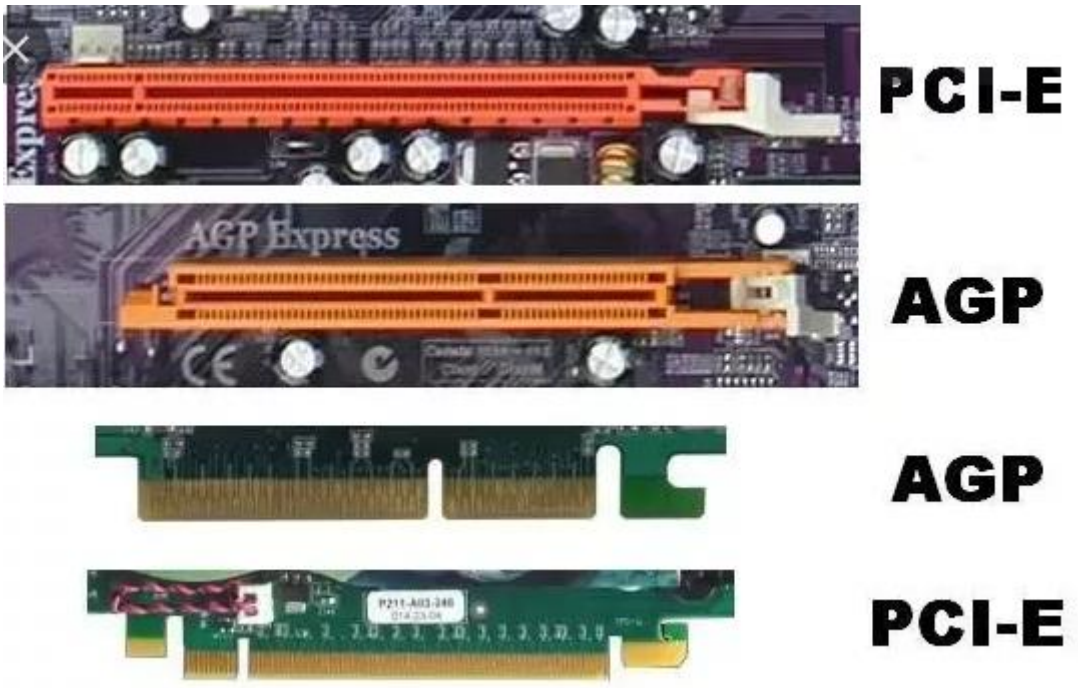


- Ports pour cartes additionnelles : (cartes graphique, carte son, carte TV, etc.)

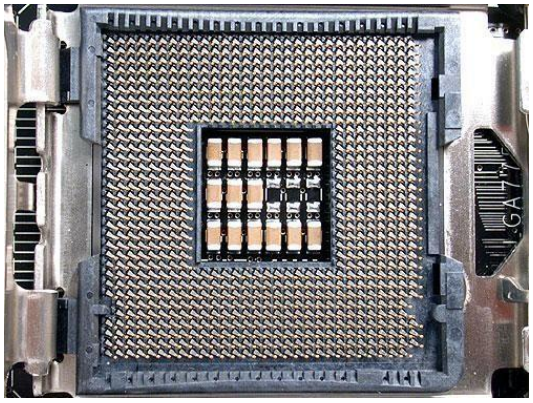
Port PCI
(**P**eripheral **C**omponent **I**nterconnect)

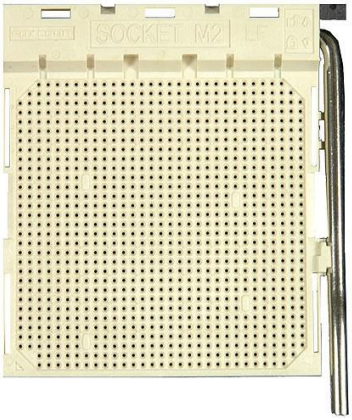


Port AGP
(**A**ccelerated **G**raphics **P**ort)



- Socket : pour brancher le microprocesseur.





- Ports USB: (**U**niversal **S**erial **B**us)

Chapitre II : Les mémoires

1. *Définitions*

- Avec une bascule [4] c'est possible de mémoriser une information sur (1) seul bit.
- Avec un registre (ensemble de bascules) c'est possible de mémoriser une information sur (n) bits.
- Si on veut mémoriser une information de taille importante ⇒ il faut utiliser une mémoire.

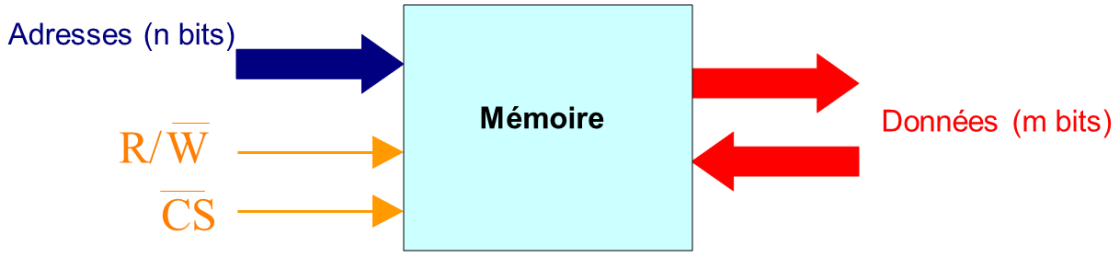
C'est quoi une **mémoire** ?

- Une mémoire est un dispositif capable :
  - D'enregistrer une information ;
  - De la **conserver** (mémoriser) ;
  - Et de la **restituer** (possible de la lire ou la récupérer par la suite) [14].

► Exemple de mémoires :

- La mémoire centrale ○ Un disque dur ○ Une disquette ○ Une mémoire ○ flash

- La mémoire peut être dans le processeur (des registres), interne (Mémoire centrale ou principale) ou externe (Mémoire secondaire) [14].



- Une mémoire communique avec les autres composant de l'ordinateur à travers :
  - Les entrées d'adresses ;
  - Les entrées de données ;
  - Les sorties de données ;
  - Les entrées de commandes :
    - Une entrée de sélection de lecture ou d'écriture (R/W) ;
    - Une entrée de sélection du circuit (CS).

2. *Caractéristiques*

a. *Capacité d'une mémoire*

La capacité (taille) d'une mémoire est le nombre (quantité) d'informations qu'on peut enregistrer (mémoriser) dans cette mémoire.
La capacité peut s'exprimer en :

[4] Une bascule est un circuit logique capable, dans certaines circonstances, de maintenir les valeurs de ses sorties malgré les changements de valeurs d'entrées.

- **Bit** : un (1) bit est l'élément de base pour la représentation de l'information.
- **Octet** : 1 Octet = 8 bits.
- **Kilo-octet (Ko)** : 1 Ko = 1024 octets = 2^{10} octets.
- **Méga-octet (Mo)** : 1 Mo = 1024 Ko = 2^{20} octets.
- **Giga-octet (Go)** : 1 Go = 1024 Mo = 2^{30} octets.
- **Téra-octet (To)** : 1 To = 1024 Go = 2^{40} octets.

b. Volatilité d'une mémoire

Si une mémoire perd son contenu (les informations) lorsque la source d'alimentation est coupée alors la mémoire est dite **volatile**.
Si une mémoire ne perd pas (conserve) son contenu lorsque la sources d'alimentation est coupée alors la mémoire est dite non volatile (mémoire permanente ou stable).

c. Mode d'accès à l'information

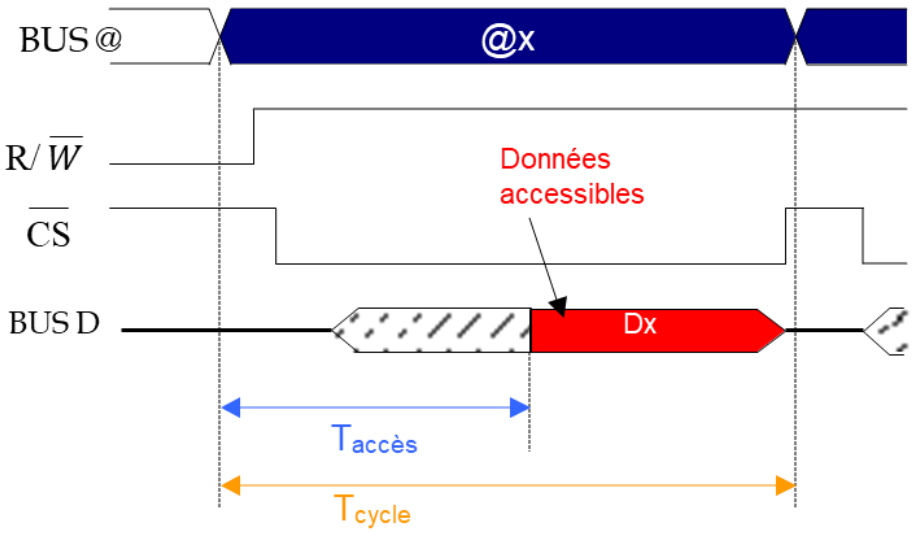
Sur une mémoire on peut effectuer une opération de :

- **Lecture** : récupérer / restituer une information à partir de la mémoire.
- **Écriture** : enregistrer une nouvelle information ou modifier une information déjà existante dans la mémoire.

Il existe des mémoires qui offrent les deux modes lecture/écriture, ces mémoires s'appellent mémoires **vives**.
Il existe des mémoires qui offrent uniquement la possibilité de la lecture (ce n'est pas possible de modifier le contenu). Ces mémoires s'appelles mémoires **mortes**.

d. Temps d'accès

C'est le temps nécessaire pour effectuer une opération de lecture ou d'écriture.
Par exemple pour l'opération de lecture, le temps d'accès est le temps qui sépare la demande de la lecture de la disponibilité de l'information [14].



Le **temps d'accès** est un critère important pour déterminer les **performances** d'une mémoire ainsi que les performances d'une machine.

3. Types de mémoires

a. Les registres

Ce sont les éléments de mémoire les plus rapides. Ils sont situés au niveau du processeur et servent au stockage des opérandes et des résultats intermédiaires.

b. La mémoire cache

C'est une mémoire rapide de faible capacité destinée à accélérer l'accès à la mémoire centrale en stockant les données les plus utilisées.

c. La mémoire principale

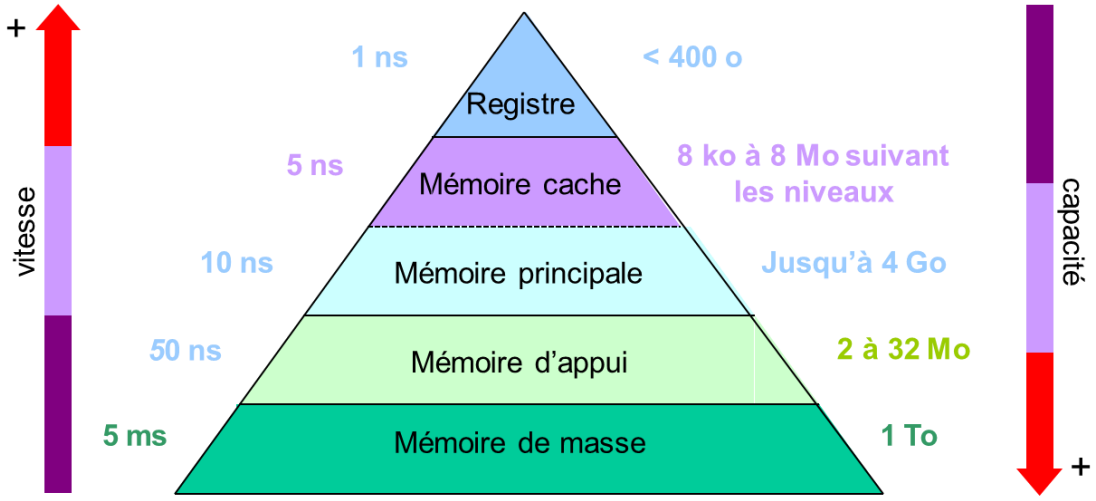
C'est l'organe principal de rangement des informations. Elle contient les programmes (instructions et données) et est plus lente que les deux mémoires précédentes.

d. La mémoire d'appui

Elle sert de mémoire intermédiaire entre la mémoire centrale et les mémoires de masse. Elle joue le même rôle que la mémoire cache.

e. La mémoire de masse

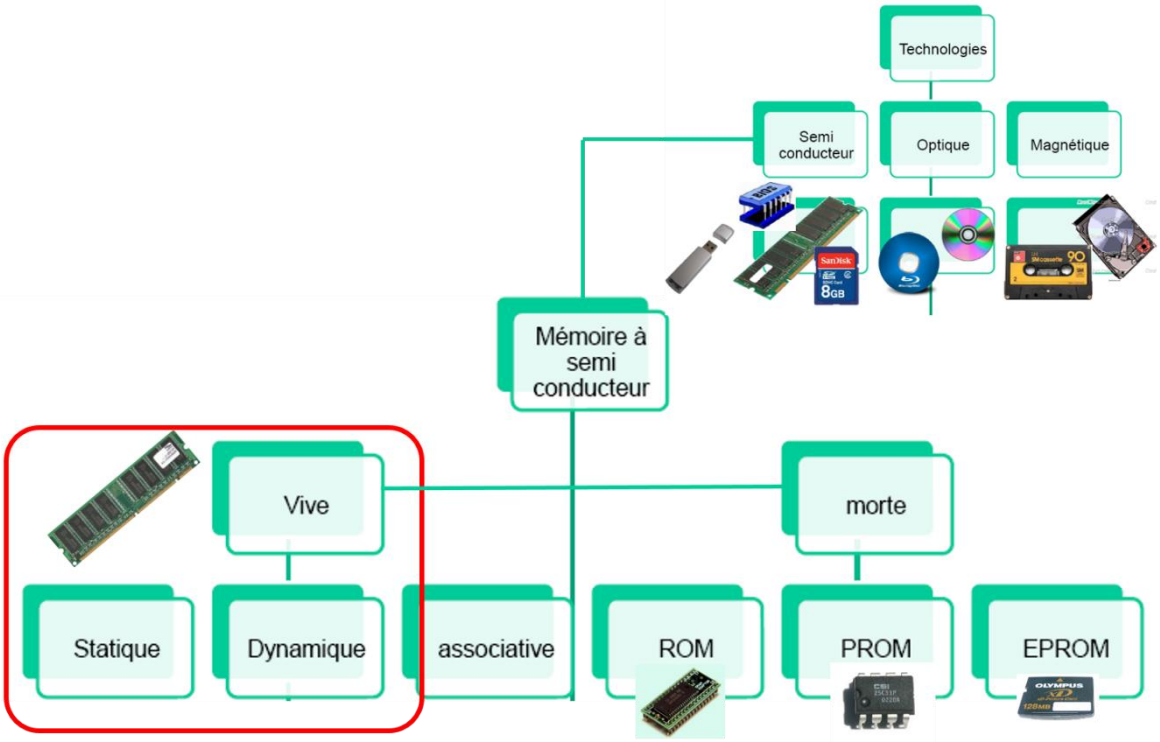
C'est une mémoire périphérique de grande capacité utilisée pour le stockage permanent ou la sauvegarde des informations. Elle utilise pour cela des supports magnétiques (disque dur, ZIP) ou optiques (CDROM, DVDROM) [14].



4. Classification des mémoires

Les mémoires peuvent être classées en trois catégories selon la technologie utilisée :

- **Mémoire à semi-conducteur :** (mémoire centrale, ROM, PROM, …) : très rapide mais de taille réduit.
- **Mémoire magnétique :** (disque dur, disquette, …) : moins rapide mais stock un volume d'informations très grand.
- **Mémoire optique** : (DVD, CDROM, ...)



5. La mémoire centrale

La mémoire centrale (MC) représente l'espace de travail de l'ordinateur (calculateur). C'est l'organe principal de rangement des informations utilisées par le processeur.

Dans une machine (ordinateur/calculateur) pour exécuter un programme il faut le charger (copier) dans la mémoire centrale. Le temps d'accès à la mémoire centrale et sa capacité sont deux éléments qui influent sur le temps d'exécution d'un programme (performance d'une machine).

La mémoire centrale est une mémoire vive (accès en lecture et écriture), réalisée à base de semi-conducteurs. Elle est dite à accès aléatoire (RAM : **R**andom **A**cces **M**emory) c'est-à-dire que le temps d'accès à l'information est indépendant de sa place en mémoire. Un temps d'accès à une mémoire centrale est moyen mais plus rapide que les mémoires magnétiques.

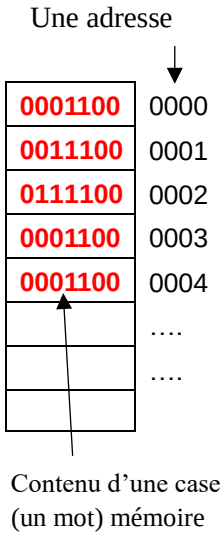
La mémoire centrale est volatile : la conservation de son contenu nécessite la permanence de son alimentation électrique. La capacité d'une mémoire centrale est limitée mais il y a toujours une possibilité d'une extension.

Pour la communication avec les autres organes de l'ordinateur, la mémoire centrale utilise les bus (bus d'adresses et bus de données).

Ils existent deux grandes familles des mémoires centrales : les mémoires **statiques** (SRAM) et les mémoires **dynamiques** (DRAM).

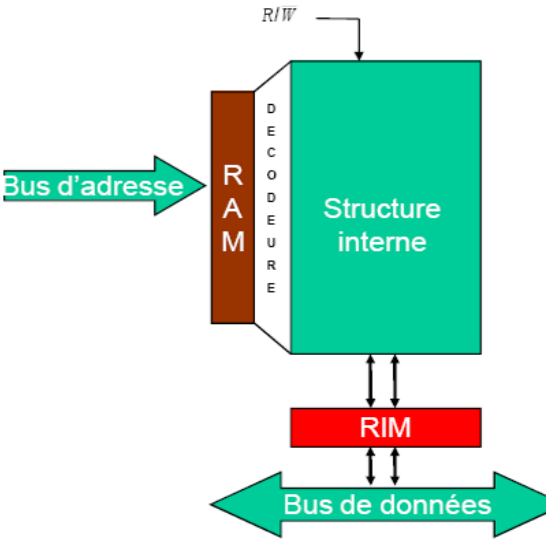
- Les mémoires statiques sont à base de bascules de type D, elles possèdent un faible taux d'intégration mais un temps d'accès rapide (Utilisation pour les mémoires cache).
- Les mémoires dynamiques à base de condensateurs, ces mémoires possèdent un très grand taux d'intégration, elles sont plus simples que les mémoires statiques mais avec un temps d'accès plus long.

a. *Vue logique de la mémoire centrale*



- La mémoire centrale peut être vu comme un large **vecteur** (tableau) de mots ou octets.
- Un **mot mémoire** stocke une information sur n bits.
- Un mot mémoire contient plusieurs cellules mémoire.
- Une **cellule** mémoire stock (1) seul bit.
- Chaque mot possède sa propre adresse.
- Une **adresse** est un numéro unique qui permet d'accéder à un mot mémoire.
- Les adresses sont séquentielles (consécutives).
  - La taille de l'adresse (le nombre de bits) dépend de la capacité de la mémoire.

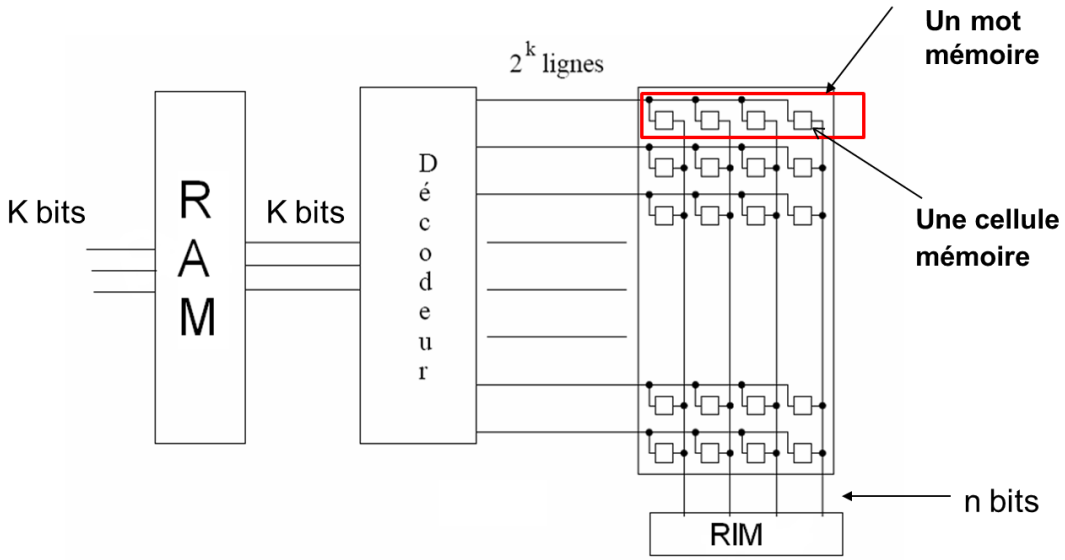
b. *Structure physique d'une mémoire centrale*



- ***RAM*** (Registre d'adresse Mémoire) : ce registre stocke l'adresse du mot à lire ou à écrire.
- ***RIM*** (Registre d'information mémoire) : stocke l'information lu à partir de la mémoire ou l'information à écrire dans la mémoire.
- **Décodeur** : permet de sélectionner un mot mémoire.
- $\boldsymbol{R/W}$: commande de lecture/écriture, cette commande permet de lire ou d'écrire dans la mémoire (si $R/W = 1$ alors lecture sinon écriture).
- **Bus d'adresses** de taille k bits.
- **Bus de données** de taille n bits.

► Comment sélectionner un mot mémoire ?

Lorsqu'une adresse est chargée dans le registre *RAM*, le décodeur va recevoir la même information que celle du *RAM*. A la sortie du décodeur nous allons avoir une seule sortie qui est active. Cette sortie va nous permettre de sélectionner un seul mot mémoire.



► Comment calculer la capacité d'une MC ?

- Soit k la taille du bus d'adresses (taille du registre RAM).
- Soit n la taille du bus de données (taille du registre RIM ou la taille d'un mot mémoire)
- On peut exprimer la capacité de la mémoire centrale soit en nombre de mots mémoire ou en bits (octets, kilo-octets, …)
    - La capacité = 2^k Mots mémoire ;
    - La capacité = $2^k \times n$ Bits.

► Exemple :

Dans une mémoire la taille du bus d'adresses K=14 et la taille du bus de données n=4.
Calculer la capacité de cette mémoire ?

$C = 2^{14} = 16384$ mots de 4 bits.

$C = 2^{14} \times 4 = 65536$ Bits $= 8192$ Octets $= 8$ Ko.

► Comment lire une information ?

Pour lire une information en mémoire centrale il faut effectuer les opérations suivantes :

1) Charger dans le registre *RAM* l'adresse du mot à lire.
2) Lancer la commande de lecture ($R/W = 1$).
3) L'information est disponible dans le registre *RIM* au bout d'un certain temps (temps d'accès).

► Comment écrire une information ?

Pour écrire une information en MC il faut effectuer les opérations suivantes :

1) Charger dans le *RAM* l'adresse du mot où se fera l'écriture.
2) Placer dans le *RIM* l'information à écrire.
3) Lancer la commande d'écriture pour transférer le contenu du *RIM* dans la mémoire.

Série de TD N°1

(Les mémoires)

Exercice 1 : (Capacité Mémoire)

1/ Quelle est la capacité en bits d'une mémoire de 16 Kbits ?

2/ Une mémoire possède 10 lignes d'adresses et 08 lignes de données, quelle est sa capacité en bits ?

3/ Combien de lignes d'adresses doit-on avoir pour accéder à 256 K.Octets sachant que chaque mot est formé d'un octet ?

4/ Soit une mémoire de capacité de 1024 bits, pour chacun des cas suivants, tracer le schéma correspondant à cette mémoire.

a) On utilisera des mots-mémoires de 1 bit.

b) On utilisera des mots-mémoires de 8 bits.

Exercice 2 : (Adressage Mémoire)

On considère une mémoire centrale de 2 M.Octets, où chaque octet est adressable séparément,

1/ Calculer l'adresse, en octal du 6ème élément d'un tableau dont l'adresse du premier élément est 77_8 et dont tous les éléments sont composés de 16 bits.

2/ Calculer, en décimal, le nombre d'octets précédents l'adresse 77_8.

3/ Calculer la taille de cette mémoire en l'exprimant en mots de 16 bits puis en mots de 32 bits.

Exercice 3 : (Temps de chargement)

Soit un fichier de taille 8,3 Méga Octets. Le chargement depuis le disque vers la mémoire centrale (MC) s'effectue via un bus de 64 bits. Sachant que la vitesse du bus ne dépassant pas les 66 Mhz, déterminer le temps nécessaire pour charger le fichier (du disque vers la MC).

Exercice 4 : (Supplémentaire)

On considère une machine avec la configuration suivante :

- Mémoire centrale de taille 1 M.Octets
- Mot Mémoire de taille 2 Octets
- Bus d'adresse (ou registre d'adresse) de taille 20 bits.

1/ Calculer la taille minimale du bus d'adresse qui permet d'accéder à cette mémoire.

2/ Déterminer la plage d'adressage de cette mémoire.

Chapitre III : Les processeurs

Un MICROPROCESSEUR est un composant électronique minuscule, fabriqué le plus souvent en silicium, qui regroupe un certain nombre de transistors élémentaires interconnectés. Le microprocesseur exécute les fonctions d'unité centrale d'ordinateur (CPU), c'est-à-dire d'exécuter des instructions envoyées par un programme [14].

Les principales caractéristiques d'un microprocesseur :

- Le **format** des mots de données : 8 bits,16 bits, etc.
- La **taille** de l'espace adressable : dépend du nombre de bits d'adresses (ex: 65536 emplacements pour 16 bits).
- La **puissance** de traitement : s'exprime en MIPS (Millions d'Instructions Par Seconde)
- Le **jeu d'instructions** :
    - Etendu (CISC)
    - Réduit (RISC)

1. Le microprocesseur

Le microprocesseur a été inventé par Marcian Hoff (surnommé Ted Hoff) en 1971, alors qu'il était ingénieur chez Intel. En 1990, Gilbert Hyatt a revendiqué la paternité du microprocesseur en se basant sur un brevet qu'il avait déposé en 1970. La reconnaissance de l'antériorité du brevet de Hyatt lui aurait permis de réclamer des redevances sur tous les microprocesseurs fabriqués de par le monde [15].
Cependant, le brevet de Hyatt a été invalidé en 1995 par l'office américain des brevets sur la base du fait que le microprocesseur décrit dans la demande de brevet n'avait pas été réalisé et n'aurait d'ailleurs pas pu l'être avec la technologie disponible au moment du dépôt du brevet [16]. Il semble que Gilbert Hyatt n'ait pas abandonné et espère faire revoir cette décision.
Le premier microprocesseur commercialisé, le 15 novembre1971, est l'Intel 4004 4-bits. Il fut suivi par l'Intel 8008. Ce microprocesseur a servi initialement à fabriquer des contrôleurs graphiques en mode texte, mais jugé trop lent par le client qui en avait demandé la conception, il devint un processeur d'usage général. Ces processeurs sont les précurseurs des Intel 8080, Zilog Z80, et de la future famille des Intel x86 [17].

Le microprocesseur exécute le programme, qui est une suite d'instructions [18].

Une instruction est une opération SIMPLE sur un (ou plusieurs) mot(s) de données :

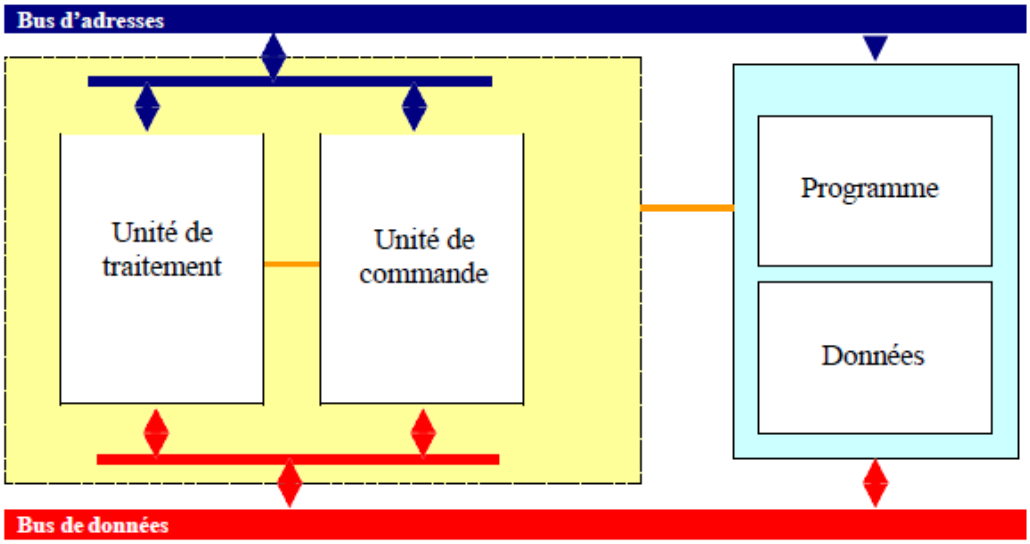
- Lecture (LOAD) ou Ecriture (STORE) en mémoire.
- Opération logique (ET, OU, etc.).
- Opération arithmétique (addition, soustraction, etc.).

2. *Architecture interne*

Un microprocesseur est construit autour de deux éléments principaux :

- Une unité de commande ;
- Une unité de traitement ;

Associés à des registres chargées de stocker les différentes informations à traiter. Ces trois éléments sont reliés entre eux par des bus interne permettant les échanges d'informations [14].



Il existe deux types de registres :

- Les registres d'usage général permettent à l'unité de traitement de manipuler des données à vitesse élevée. Ils sont connectés au bus des données interne au microprocesseur.
- Les registres d'adresses (pointeurs) connectés sur le bus adresses.

a. *L'unité de commande*

Elle permet de séquencer le déroulement des instructions. Elle effectue la recherche en mémoire de l'instruction. Comme chaque instruction est codée sous forme binaire, elle en assure le décodage pour enfin réaliser son exécution puis effectue la préparation de l'instruction suivante. Pour cela, elle est composée de :

- **Compteur de programme** : constitué par un registre dont le contenu est initialisé avec L'adresse de la première instruction du programme. Dès le lancement du programme ce compteur contient l'adresse de la première instruction à exécuter :
  - Soit par incrémentation automatique dans le cas où les adresses des instructions se suivent.
  - Soit par chargement de l'adresse de branchement dans le cas de sauts programmés.
- **Registre d'instruction et le décodeur d'instruction** : chacune des instructions à exécuter dont le format dépend de l'architecture du processeur est rangée dans le registre instruction puis est décodée par le décodeur d'instruction. Le premier mot est toujours le code de l'opération que le décodeur d'instruction doit identifier.
- **Bloc logique de commande (ou séquenceur)** : Il organise l'exécution des instructions au rythme d'une horloge. Il élabore tous les signaux de synchronisation

internes ou externes (bus de commande) du microprocesseur en fonction des divers signaux de commande provenant du décodeur d'instruction ou du registre d'état par exemple. Il s'agit d'un automate réalisé soit de façon câblée (obsolète), soit de façon micro-programmée, on parle alors de microprocesseur.

- **Registre d'adresses** : est un registre tampon qui assure l'interfaçage entre le microprocesseur et son environnement. Il conditionne le bus externe des adresses.

- **Registre de données** : est un registre tampon qui assure l'interfaçage entre le microprocesseur et son environnement ou inversement. Il conditionne le bus externe ou le bus interne des données

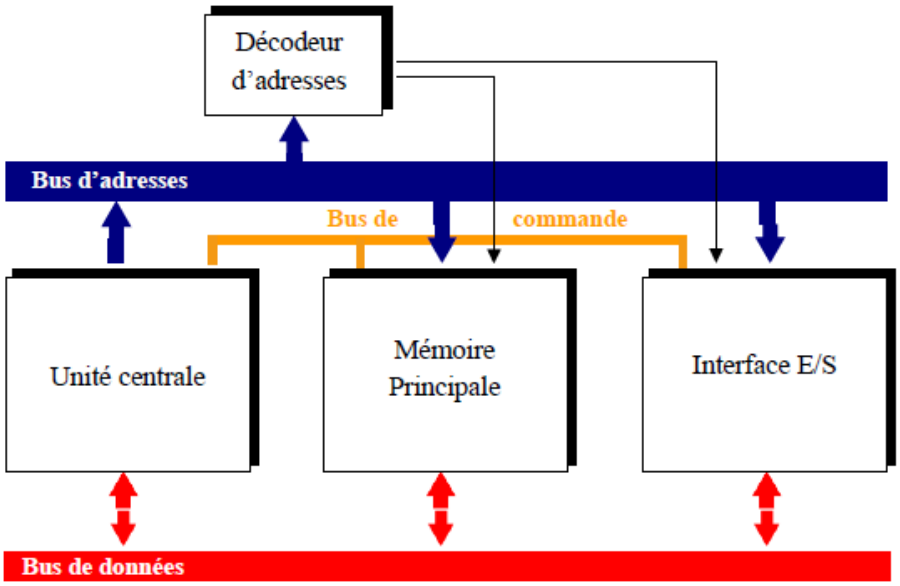
b. L'unité de traitement

C'est le cœur du microprocesseur. Elle regroupe les circuits qui assurent les traitements nécessaires à l'exécution des instructions :

- **L'Unité Arithmétique et Logique (UAL)** est un circuit complexe constituée par un certain nombre de circuits tels que : complémenteur, additionneur, décaleur, portes logiques, et qui assure les fonctions :
  - Arithmétique (Addition, soustraction).
  - Logiques (ET, OU, Comparaison, Décalage, etc.).
  - Comparaison, décalage à droite ou à gauche, incrémentation, décrémentation, mise à 1 ou à 0 d'un bit, test de bit.
- **Le registre d'état** est généralement composé de 8 bits à considérer individuellement. Chacun de ces bits est un indicateur dont l'état dépend du résultat de la dernière opération effectuée par l'UAL. On les appelle indicateur d'état ou flag ou drapeaux. Dans un programme le résultat du test de leur état conditionne souvent le déroulement de la suite du programme. On peut citer par exemple les indicateurs de :
  - Retenue (Carry : C) ○ Retenue intermédiaire (Auxiliary-Carry : AC)
  - Signe (Sign : S)
  - Débordement (Overflow : OV ou V) ○ Zéro (Z) ○ parité (Parity : P)
- **Les accumulateurs** sont des registres de travail qui servent à stocker un opérande au début d'une opération arithmétique et le résultat à la fin de l'opération. Ils permettent aussi de stocker temporairement des données en provenance de l'extérieur du microprocesseur avant leur reprise pour être rangées en mémoire ou des données provenant de la mémoire ou de l'UAL pour les présenter vers l'extérieur du microprocesseur.
- **Les registres auxiliaires** permettent de stocker le résultat des instructions exécutées par l'ALU.

c. Les bus de communication

Ils relient électriquement les composants internes du microprocesseur (bus internes) et relient les périphériques (mémoires et interfaces E/S) au processeur (bus externes).



On distingue 3 types de bus :

i Le bus de données

Il est bidirectionnel et assure le transfert des informations entre le microprocesseur et son environnement, et inversement. Son nombre de lignes est égal au format des mots de données du microprocesseur.

ii Le bus d'adresses

Il est unidirectionnel et permet la sélection des informations à traiter dans un espace mémoire (ou espace adressable) qui peut avoir 2kemplacements, avec k = nombre de conducteurs du bus d'adresses.

iii Le bus de commande (ou bus de contrôle)

Il est constitué par quelques conducteurs qui assurent la synchronisation des flux d'informations sur les bus de données et d'adresses.

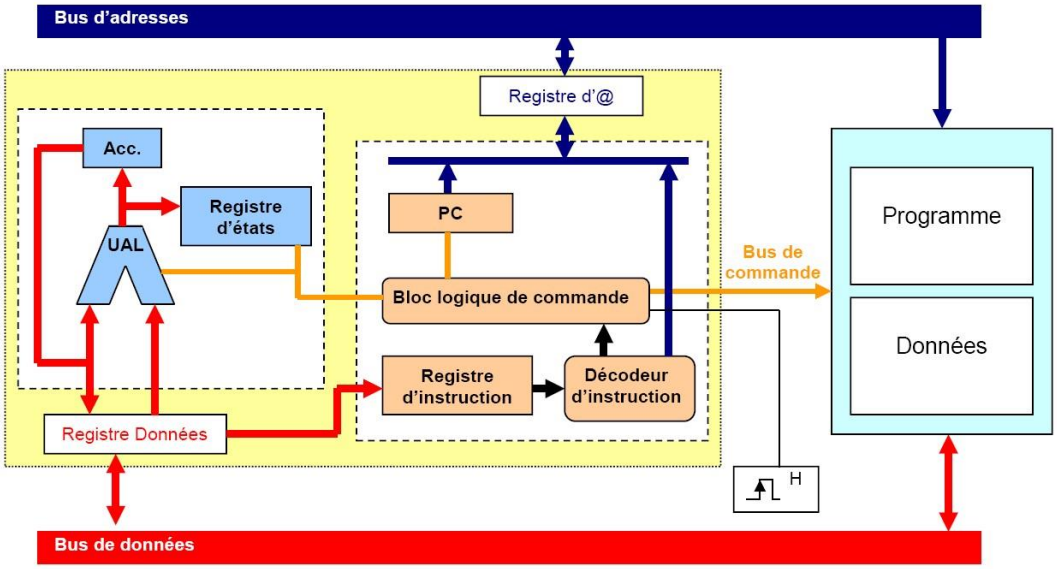


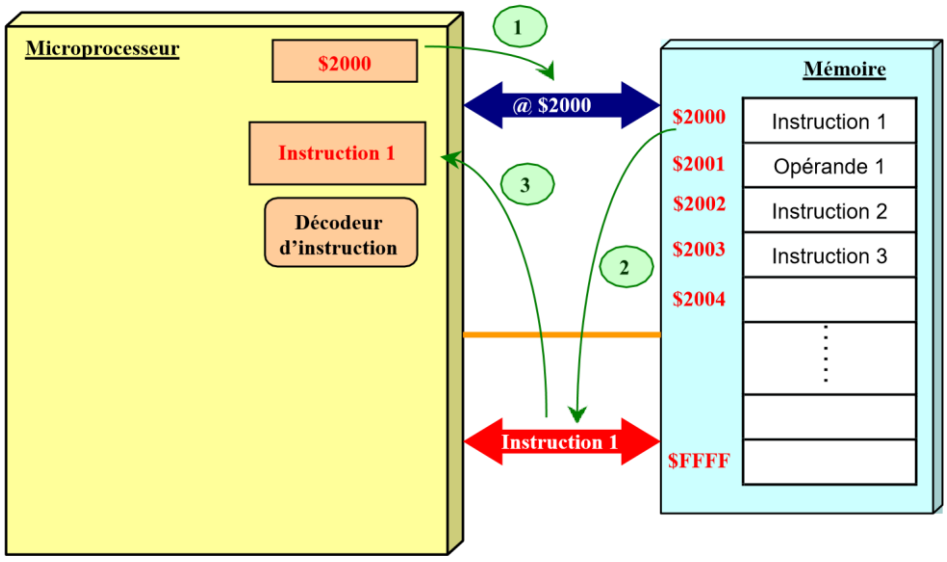
Figure 6 : Le schéma fonctionnel du microprocesseur.

3. Cycle d'une instruction

Le microprocesseur ne comprend qu'un certain nombre d'instructions qui sont codées en binaire. Le traitement d'une instruction peut être décomposé en trois phases :

• **Phase 1** : Recherche de l'instruction à traiter

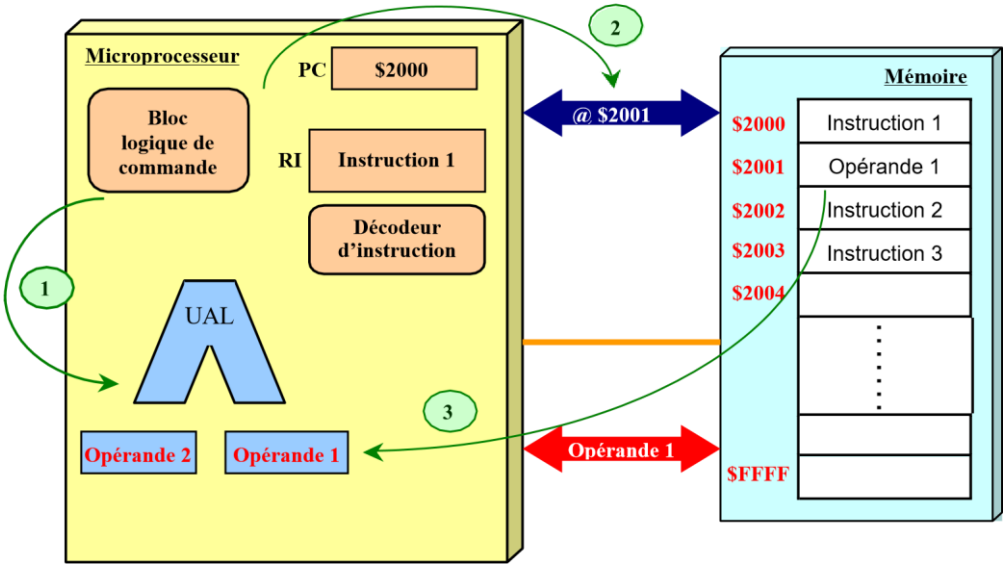
1. Le PC contient l'adresse de l'instruction suivante du programme. Cette valeur est placée sur le bus d'adresses par l'unité de commande qui émet un ordre de lecture.
2. Au bout d'un certain temps (temps d'accès à la mémoire), le contenu de la case mémoire sélectionnée est disponible sur le bus des données.
3. L'instruction est stockée dans le registre instruction du processeur.



• **Phase 2 :** Décodage de l'instruction et recherche de l'opérande

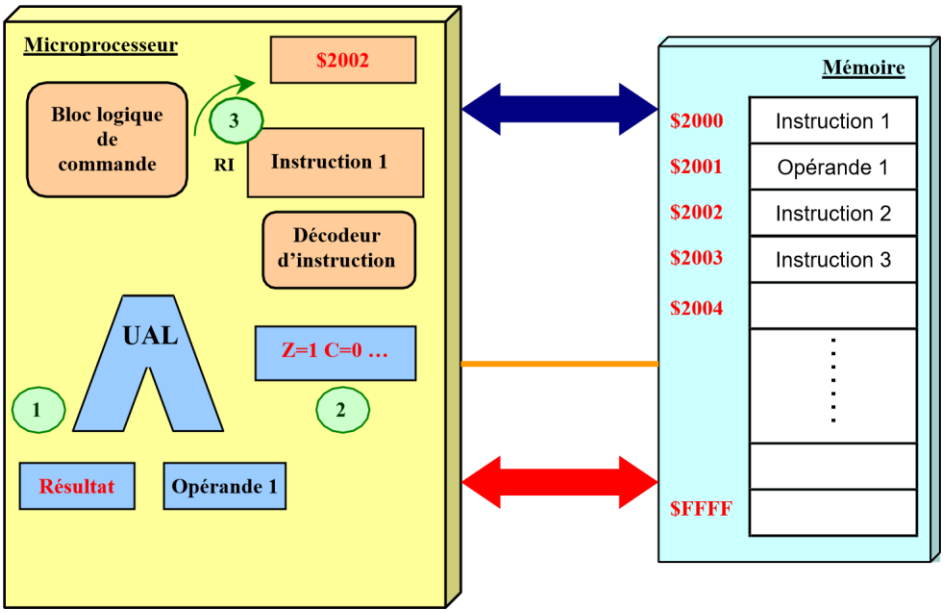
Le registre d'instruction contient maintenant le premier mot de l'instruction qui peut être codée sur plusieurs mots. Ce premier mot contient le code opératoire qui définit la nature de l'opération à effectuer (addition, rotation, ...) et le nombre de mots de l'instruction.

1. L'unité de commande transforme l'instruction en une suite de commandes élémentaires nécessaires au traitement de l'instruction.
2. Si l'instruction nécessite une donnée en provenance de la mémoire, l'unité de commande récupère sa valeur sur le bus de données.
3. L'opérande est stocké dans un registre.



• **Phase 3 :** Exécution de l'instruction

1. Le séquenceur réalise l'instruction est exécuté.
2. Les drapeaux sont positionnés (registre d'état).
3. L'unité de commande positionne le PC pour l'instruction suivante.



5. Jeu d'instructions

a. Définition

La première étape de la conception d'un microprocesseur est la définition de son jeu d'instructions. Le jeu d'instructions décrit l'ensemble des opérations élémentaires que le microprocesseur pourra exécuter. Il va donc en partie déterminer l'architecture du microprocesseur à réaliser et notamment celle du séquenceur. A un même jeu d'instructions peut correspondre un grand nombre d'implémentations différentes du microprocesseur [14].

b. Type d'instructions

Les instructions que l'on retrouve dans chaque microprocesseur peuvent être classées en 4 groupes :

- **Transfert de données** pour charger ou sauver en mémoire, effectuer des transferts de registre à registre, etc…
- **Opérations arithmétiques** : addition, soustraction, division, multiplication
- **Opérations logiques** : ET, OU, NON, NAND, comparaison, test, etc…
- **Contrôle de séquence** : branchement, test, etc…

c. Codage

Les instructions et leurs opérandes (paramètres) sont stockés en mémoire principale. La taille totale d'une instruction (nombre de bits nécessaires pour la représenter en mémoire) dépend du type d'instruction et aussi du type d'opérande. Chaque instruction est toujours codée sur un nombre entier d'octets afin de faciliter son décodage par le processeur. Une instruction est composée de deux champs :

- Le code instruction, qui indique au processeur quelle instruction réaliser
- Le champ opérande qui contient la donnée, ou la référence à une donnée en mémoire (son adresse).

► Exemple :

Code instruction	Code Opérande
1001 0011	**0011 1110**

Le nombre d'instructions du jeu d'instructions est directement lié au format du code instruction. Ainsi un octet permet de distinguer au maximum 256 instructions différentes.

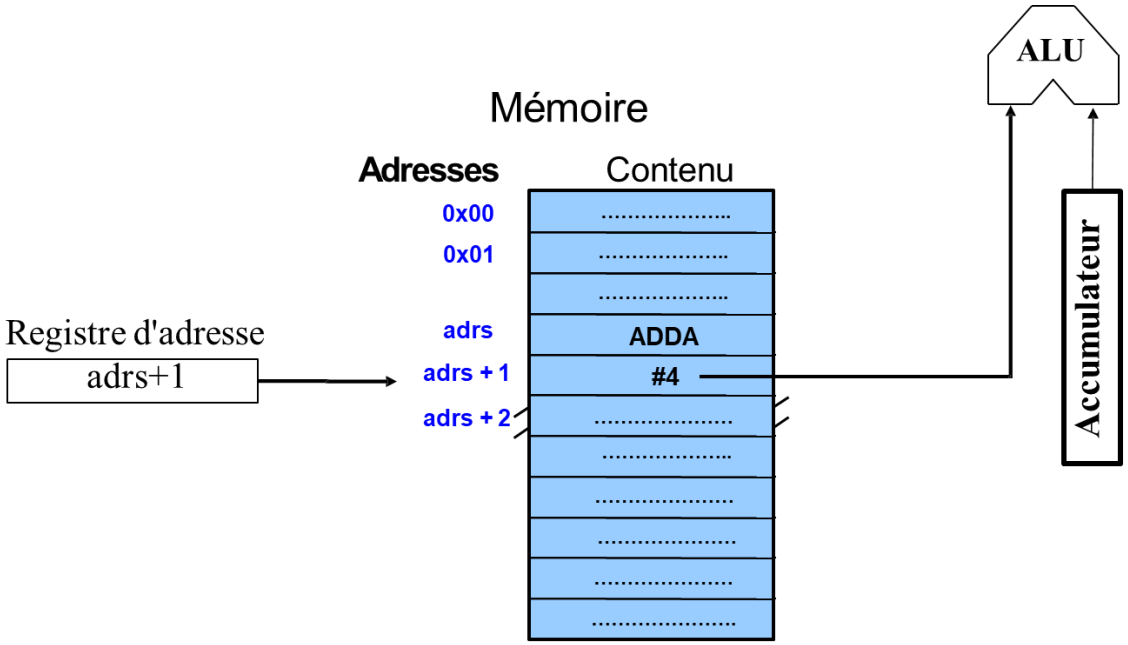
d. Mode d'adressage

Un mode d'adressage définit la manière dont le microprocesseur va accéder à l'opérande.

Les différents modes d'adressage dépendent des microprocesseurs mais on retrouve en général:

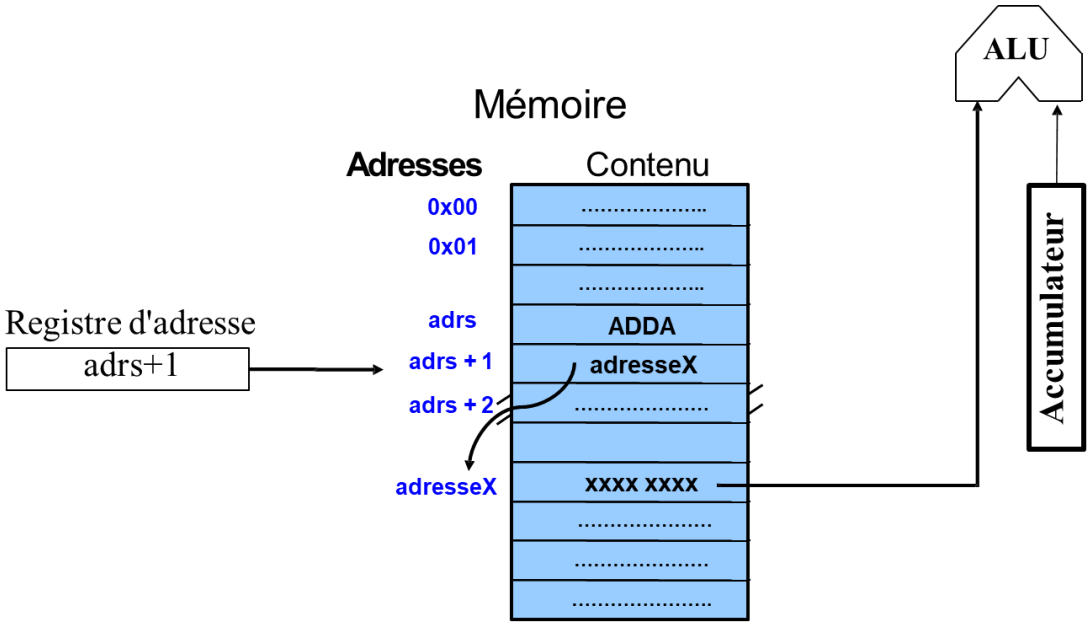
i ***L'adressage immédiat***

► Exemple : ADDA #4



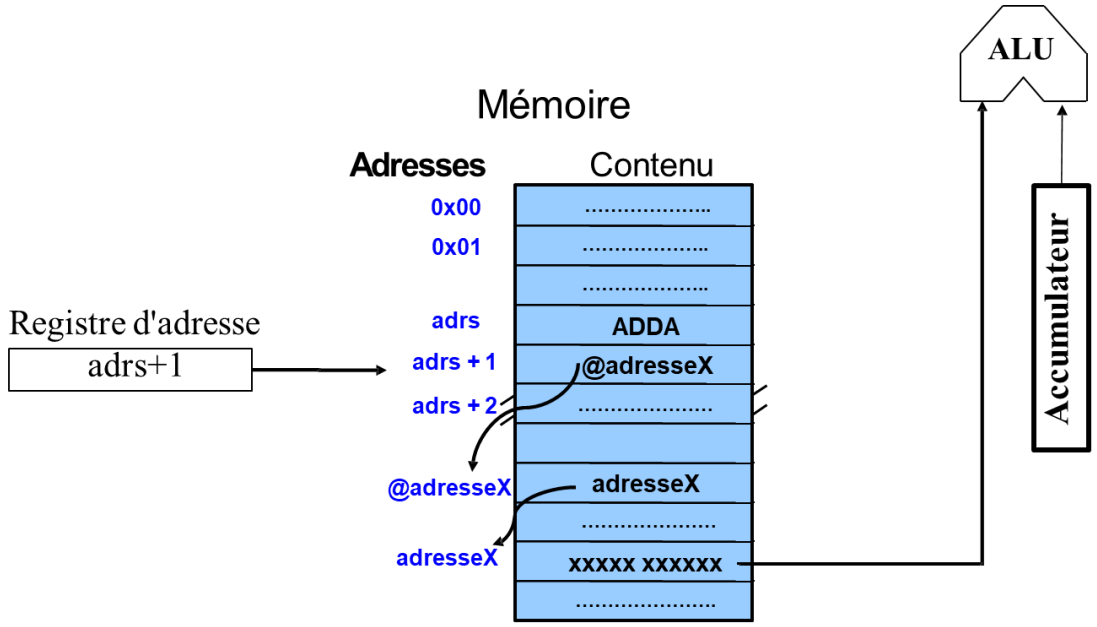
ii L'adressage direct

► Exemple : ADDA adresseX



iii L'adressage indirect

► Exemple : ADDA @adresseX



e. Notion d'architecture RISC et CISC

Actuellement l'architecture des microprocesseurs se composent de deux grandes familles :

- L'architecture **CISC** (***C**omplex **I**nstruction **S**et **C**omputer*)
- L'architecture **RISC** (***R**educed **I**nstruction **S**et **C**omputer*)

Le choix dépendra des applications visées. En effet, si on diminue le nombre d'instructions, on crée des instructions complexes (CISC) qui nécessitent plus de cycles pour être décodées et si on diminue le nombre de cycles par instruction, on crée des instructions simples (RISC) mais on augmente alors le nombre d'instructions nécessaires pour réaliser le même traitement.

Architecture RISC	Architecture CISC
• Instructions simples ne prenant qu'un seul cycle	• Instructions complexes prenant plusieurs cycles
• Instructions au format fixe	• Instructions au format variable
• Décodeur simple (câblé)	• Décodeur complexe (microcode)
• Beaucoup de registres	• Peu de registres
• Seules les instructions LOAD et STORE ont accès à la mémoire	• Toutes les instructions sont susceptibles d'accéder à la mémoire
• Peu de modes d'adressage	• Beaucoup de modes d'adressage
• Compilateur complexe	• Compilateur simple

Durant longtemps, les CISC et les RISC eurent chacun leurs admirateurs et leurs détracteurs. Mais au final, on ne peut pas dire qu'un processeur CISC sera meilleur qu'un RISC ou l'inverse : chacun a des avantages et des inconvénients, qui rendent le RISC/CISC adapté ou pas selon la situation.

Série de TD N°2

(Les microprocesseurs)

Exercice 1 : (Macro-instructions)

Soit l'instruction suivante : ADD 20, 18, 30 # équivalente à [30] = [20] + [18]

Sachant que [20] =15 ; [18] =30 ;

1/ réécrire cette instruction en une suite d'instructions de format à deux (02) adresses.

2/ réécrire cette instruction en une suite d'instructions de format à une (01) adresse.

Exercice 2 : (Modes d'adressage)

Trouvez les résultats du fragment de programme suivant pour les 03 modes d'adressage suivants : Immédiat – Direct – Indirect, sachant que [ACC]=100 ; [20] =60 ; [60] =5 ; [5] =20. 10 : ADD 20

11 : SUB 60

12 : MPY 5

13 : DIV 20

Exercice 3 : (Etapes d'exécution)

a) Décrivez les différentes étapes d'exécution des instructions

: 10 : ADD 25 ;

17 : SUB 40 ;

18 : STR 25 ;

b) Dérouler le petit programme suivant, sachant que : [ACC]=50 ; [30] =10 ; [31] =20 ; 10 : ADD 30

11 : DIV 31

12 : STR 32

13 : Branch si p=1/-4

Exercice 4 : (Microprogrammes)

Donnez l'expression de X effectuée par le programme suivant en mode direct

Sachant que [70] =A ; [50] =B ; [3] =C ; [160] =X.

10: LOAD 70 15: DIV 70

11: ADD 50 16 : SUB 3

12: MPY 3 17 : ADD 100

13: STR 100 18 : STR 160

14: LOAD 50

Exercice 5 : (supplémentaire)

Réalisez un programme qui calcule l'expression suivante dans une machine à une adresse et dans une machine à zéro adresse.

$$X = (A + B * C) / (D – E * F).$$

Chapitre IV : Représentation et codage de l'information

1. Histoire de la numérotation

a. Les trigrammes de Fu Hsi

- Fu Hsi, considéré comme le premier roi chinois (le premier des trois Augustes), aurait vécu, il y a plus de 5000 ans [19].
- A l'origine du Yi King (livre des mutations) contenant les trigrammes disposés dans l'ordre du ciel antérieur [20].

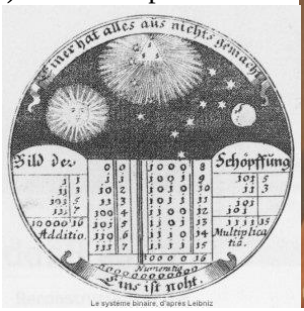
b. Les trigrammes et les hexagrammes du roi Wen

- Le roi Wen (1150-750 av. J.C), aurait proposé une disposition inversée des trigrammes dit du ciel postérieur.
- Il aurait aussi complété le Yi King en introduisant les 64 hexagrammes qui sont la combinaison de 2 trigrammes [20].

Trigrammes supérieur → inférieur ↓	☰ qián le Ciel	☳ zhèn le Tonnerre	☵ kǎn l'Eau	☶ gèn la Montagne	☷ kūn la Terre	☴ xùn le Vent	☲ lí le Feu	☱ duì la Brume
☰ qián le Ciel	1	34	5	26	11	09	14	43
☳ zhèn le Tonnerre	25	51	3	27	24	42	21	17
☵ kǎn l'Eau	6	40	29	4	7	59	64	47
☶ gèn la Montagne	33	62	39	52	15	53	56	31
☷ kūn la Terre	12	16	8	23	2	20	35	45
☴ xùn le Vent	44	32	48	18	46	57	50	28
☲ lí le Feu	13	55	63	22	36	37	30	49
☱ duì la Brume	10	54	60	41	19	61	38	58

c. L'algèbre binaire de Leibniz

- Gottfried Wilhelm Leibniz (1646-1716) a étudié le procédé de chiffrage de Bacon et les trigrammes du Yi King [21].
- Met au point l'arithmétique binaire et expose ses travaux devant l'Académie des Sciences de Paris.
- Publie "Explication de l'arithmétique binaire" en 1703 [22].



Le système binaire, d'après Leibniz

Einer hat alles aus nichts gemacht

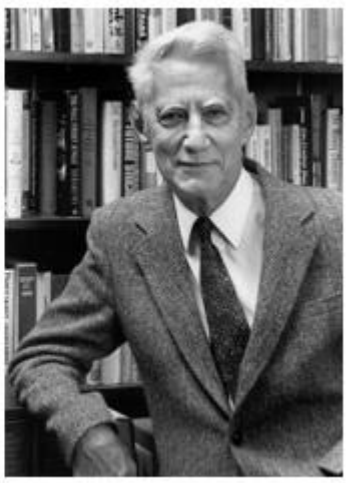
Tout est créé à partir de l'unité (1, Dieu) et du néant (0)

كل شيء مخلوق من الوحدة (1 ، الله) والعدم (0)

— Gottfried Wilhelm Leibniz

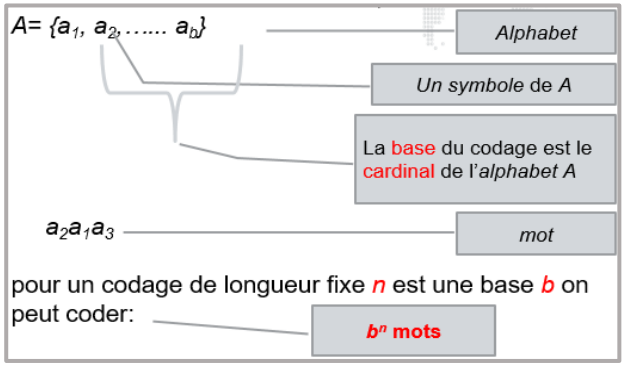
d. Pourquoi le binaire ?

- En 1938, Claude Shannon (1916-2001) expose lors de ses travaux de thèse, le principe d'ouverture (le 0), fermeture (le 1) de circuit électrique. Il explique le principe de la machine à relais [23].



- Un ordinateur est un circuit (électronique) ou circule de la tension. Le binaire représente les états de logique de l'ordinateur traduit par 1, s'il y a une tension, ou 0, en absence de tension.
- Nos programmes manipulent, calculent et utilisent donc uniquement des mots binaires.
- Les opérations binaires seront utiles pour construire et comprendre les réseaux informatiques.

3. Système de numération



Un système de numération décrit la façon avec laquelle les nombres sont représentés [24].
Un système de numération est défini par:

a. Exemples de système de numération

- Numération Romaine : I, II, III, IV, …, X, …, XXI -> 271
- Numération babylonienne : 
- Numération décimale : $\mathcal{A}$ = {0,1,2,3,4,5,6,7,8,9}
  - Le nombre 10 est la base de cette numération.
  - C'est un système positionnel. Chaque position possède un poids.

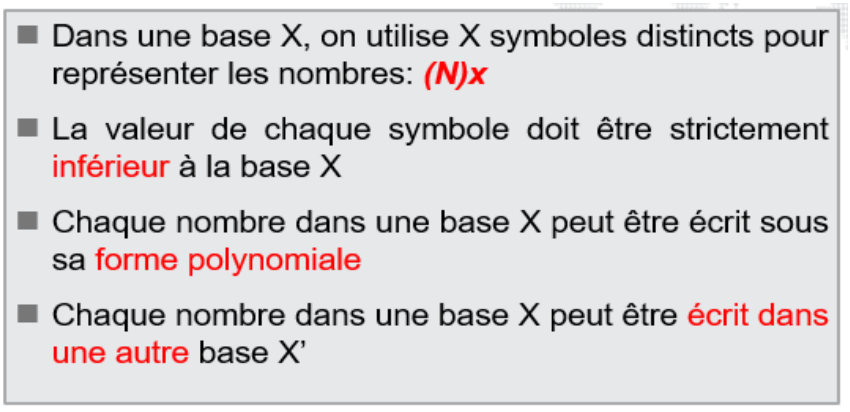
b. Système de numération positionnel pondéré à base b

Un système de numérotation positionnel pondéré à base b est défini sur un alphabet de b chiffres: $\mathcal{A}$ = {c_0, c_1, …, c_{b-1}} avec $0 \leq c_i < b$.

- Soit N = $a_{n-1}\ a_{n-2}$ ... $a_1\ a_0$ $_{(b)}$: représentation en base b sur n chiffres.
  - a_i : est un chiffre de l'alphabet de poids i (position i).
  - a_0 : chiffre de poids 0 appelé le chiffre de poids faible.
  - a_{n-1} : chiffre de poids n-1 appelé le chiffre de poids fort.
- La valeur de N en base 10 est donnée par :
  $N = a_{n-1}.b_{n-1} + a_{n-2}.b_{n-2} + ... + a_0.b_0$ $_{(10)} = \sum_{ii=0}^{n-1} . b_{ii}$ (forme polynomiale)

c. Bases de numération

- **Système binaire :** système à base 2 (b=2) utilise deux chiffres : {0,1}
  - C'est avec ce système que fonctionnent les ordinateurs.
- **Système Octal :** système à base 8 (b=8) utilise huit chiffres : {0,1,2,3,4,5,6,7}
  - Utilisé il y a un certain temps en Informatique.
  - Elle permet de coder **3 bits** par un seul symbole.
- **Système Hexadécimal :** système à base 16 (b=16) utilise 16 chiffres :
  {0,1,2,3,4,5,6,7,8,9, A=$10_{(10)}$, B=$11_{(10)}$, C=$12_{(10)}$, D=$13_{(10)}$, E=$14_{(10)}$, F=$15_{(10)}$}
  - Cette base est très utilisée dans le monde de la micro-informatique.
  - Elle permet de coder **4 bits** par un seul symbole.
- ***Synthèse :***



d. Transcodage (conversion de base)

Le transcodage ou conversion de base est l'opération qui permet de passer de la représentation d'un nombre exprimé dans une base à la représentation du même nombre mais exprimé dans une autre base [24].

Par la suite, on verra les conversions suivantes :

- Décimal vers Binaire, Octal et Hexadécimal.
- Binaire vers Décimal, Octale et Hexadécimal.

i ***Base 10 vers une Base b***

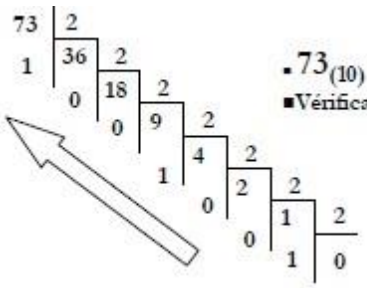
Convertir un nombre décimal $(N)_{10}$ → base b se fait par:

1. la conversion de la partie entière à part (divisions successives du nombre par la base b *jusqu'à avoir un résultat nul*) et;
2. la conversion de la partie fractionnelle à part (*multiplications successives* par la base b *jusqu'à avoir un résultat nul ou obtenir une précision donnée*)

► **Exemple 1**: Soit N le nombre d'étudiants d'une classe représenté en base décimale par:

$N = 73_{(10)}$

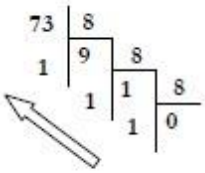
Représentation en Binaire ?



$.73_{(10)} = 1001001_{(2)}$

▪Vérification

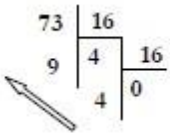
Représentation en Octal ?



$.73_{(10)} = 111_{(8)}$

▪Vérification

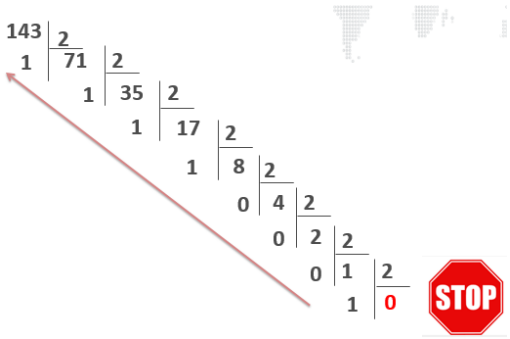
Représentation en Hexadécimal ?



$.73_{(10)} = 49_{(16)}$

▪Vérification

► **Exemple 2**: $(143,526)_{10}=(?)_2$ *avec 7 chiffres après la virgule.*

Partie entière: divisions successives	**Partie décimale:** multiplications successives
	0,526*2=1,052 0,052*2=0,104 0,104*2=0,208 0,208*2=0,416 0,416*2=0,832 0,832*2=1,664 0,664*2=1,328 $(0,526)_{10}=(0,1000011)_2$ Stop (7 chiffres après la virgule)

Donc: $(143,526)_{10}=(10001111,1000011)_2$

ii ***Base b vers la Base 10***

Pour convertir un nombre en base b → Décimale:

1. $(a_{n-1}a_{n-2}.....a_1a_0, a_{-1}a_{-2}.....a_{-p})_b$, il suffit de l'écrire sous sa forme polynomiale dans la base b,
2. ensuite faire l'addition décimale

$$(a_{n-1}a_{n-2}.....a_1a_0, a_{-1}a_{-2}.....a_{-p})_b = a_{n-1}xb^{n-1}+a_{n-2}xb^{n-2}+.....+a_1xb^1+a_0xb^0+ a_{-1}xb^{-1}+a_{-2}xb^{-2}+.....a_{-p}xb^{-p}$$

► **Exemple 2**: $(1572,36)_8 =(?)_{10}$

3 2 1 0 -1-2 ← les positions

$(1572 , 36)_8 =$

$1x8^3+5x8^2+7x8^1+2x8^0+3x8^{-1}+6x8^{-2} =$

$512+320+56+2+0,375+0,09375=$

$(890,46875)_{10}$

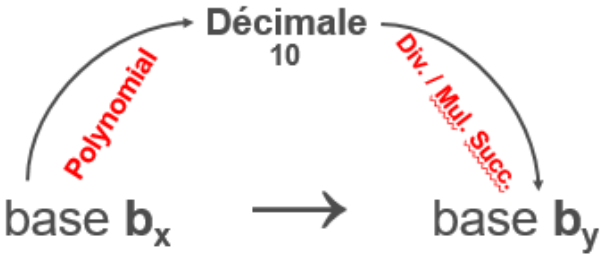
iii ***Base b_x vers une Base b_y***

Convertir un nombre dans un base b_x → base b_y se fait par:

1. la conversion de du nombre base b_x → **Décimale**;
2. la conversion de du nombre base **Décimale** → b_y

iv ***<u>Cas particuliers: de la Base 2 vers l'Octal ou l'Hexadécimale et inversement</u>***

• ***Solution 1*** *:* convertir le nombre en base binaire vers la base décimale puis convertir ce nombre en base 10 vers la base b.



► **Exemple :**

$10010_{(2)} = ?_{(8)}$

$10010_{(2)} = 2^4+2^{1(}{}_{10)} = 18_{(10)} = 2*8^1 + 2*8^0{}_{(10)} = 22_{(8)}$

• ***Solution 2*** *:*

○ Binaire vers Octal :

Elle consiste à regrouper les bits en trois positions de droit à gauche pour la partie entière et de gauche a droit pour la partie fractionnelle, puis remplacer chaque groupe par son équivalent octal,

Binaire $_2$ → Octale $_8$ $\boxed{8 = 2^3}$

► **Exemple:** $(1\,1\,0\,1\,0\,,\,0\,1\,1\,1)_2$

$(0\,1\,1\,0\,1\,0\,,\,0\,1\,1\,1\,0\,0)$

3 2 , 3 4

Octal	Binaire
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

$(\underline{011}\ \underline{010},\underline{011}\ \underline{100})_2=(32,34)_8$

○ Octal vers Binaire:

Il suffit de remplacer chaque digit octal par son équivalent binaire en trois positions

► **Exemple:** $(726,31)_8=(?)_2$

$(7 \quad 2 \quad 6 \quad , \quad 3 \quad 1)_8$

$111 \quad 010 \quad 110 \quad , \quad 011 \quad 001$

Octal	Binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

$(726,31)_8=(111\ 010\ 110,011\ 001)_2$

- Binaire vers Hexadécimal :

Elle consiste à regrouper les bits en quatre positions de droit à gauche pour la partie entière et de gauche a droit pour la partie décimale, puis remplacer chaque groupe par son équivalent hexadécimal

Binaire $_2$ → Hexadécimale $_{16}$ $\quad 16 = 2^4$

► **Exemple:**

$(1\ 1\ 0\ 1\ 0\ ,\ 1\ 0\ 1\ 1\ 1)_2$

$(0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ ,\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0)$

$1 \quad A \quad , \quad B \quad 8$

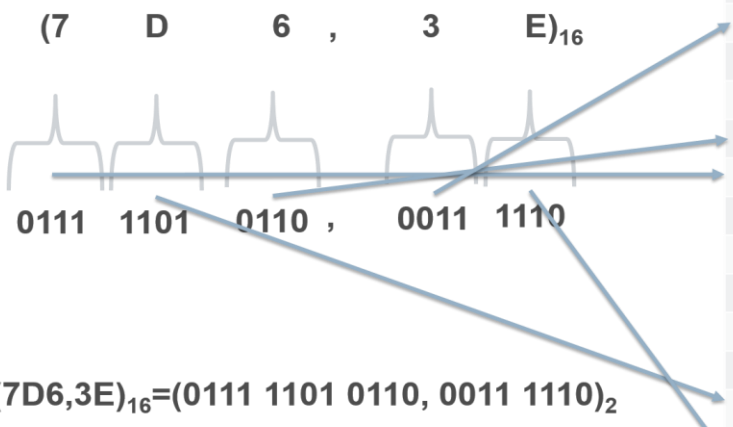
$=(\underline{0001}\ \underline{1010},\underline{1011}\ \underline{1000})_2=(1A,B8)_{16}$

Héxad.	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

- Hexadécimal vers Binaire:

Il suffit de remplacer chaque digit hexadécimal par son équivalent binaire en quatre positions,

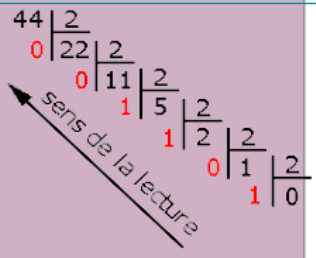
► **Exemple:** $(7D6,3E)_{16}=(?)_2$



Héxad.	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

• ***Synthèse :***

Méthode

conversion	Méthode	Exemple
10 =>X	Division successive sur X القسمة الإقليدية المتتالية على العدد X، حتى يصبح الحاصل 0، ثم أخذ البواقي من اليمين إلى اليسار	 $(44)_{10}=(101100)_2$
X => 10	Développement polynomial نشر كثير حدود بالضرب في قوى الأساس x	5^2 | 5^1 | 5^0 2 | 1 | 0 $(210)_5=0\times5^0+1\times5^1+2\times5^2$ $=0+5+25\times2=(55)_{10}$
X=> Y	Passer par la base 10 المرور بالأساس 10	$(210)_5=(55)_{10}=(67)_8$
2 =>8	3 chiffres binaires => un chiffre octal كل ثلاثة أرقام ثنائية يقابلها رقم ثماني	Binaire (101 110 011)$_2$ ↓ ↓ ↓ Octal (5 6 3)$_8$
8 => 2	un chiffre octal => 3 chiffres binaires كل رقم ثماني يقابل ثلاثة أرقام ثنائية	Octal (5 6 3)$_8$ ↓ ↓ ↓ Binaire (101 110 011)$_2$
2 => 16	4 chiffres binaires => un chiffre octal كل أربعة أرقام ثنائية تقابل رقما ستعشريا	Binaire (1010 0110 0011)$_2$ ↓ ↓ ↓ Hexa (A 6 3)$_8$
16 => 2	un chiffre hexadécimal => 4 chiffres binaires كل رقم ستعشري يقابل أربعة أرقام ثنائية	Hexa (A 6 3)$_{16}$ ↓ ↓ ↓ Binaire (1010 0110 0011)$_2$

4. Codage de l'information

Les informations traitées par les ordinateurs sont de différentes natures :

- Nombres, texte,
- Images, sons, vidéo,
- Programmes, …
- Dans un ordinateur, elles sont toujours représentées sous forme binaire (BIT: Binary digIT), une suite de 0 et de 1.

Le codage de l'information permet d'établir une correspondance qui permet sans ambiguïté de passer d'une représentation (dite externe) d'une information à une autre représentation (dite interne: sous forme binaire) de la même information, suivant un ensemble de règles précises [24].

► **Exemple :**

- Le nombre 35 : 35 est la représentation externe du nombre trente-cinq.
  - La représentation interne de 35 sera une suite de 0 et 1 (100011).

En informatique, Le codage de l'information s'effectue principalement en trois étapes :

1. L'information sera exprimée par une suite de nombres (**Numérisation**).
2. Chaque nombre est codé sous forme **binaire** (suite de 0 et 1).
3. Chaque élément binaire est représenté par un état **physique**.

Le codage de l'élément binaire par un état physique peut se faire par :

- Charge électrique (RAM : Condensateur-transistor) : Chargé (bit 1) ou non chargé (bit 0)
- Magnétisation (Disque dur, disquette) : polarisation Nord (bit 1) ou Sud (bit 0)
- Alvéoles (CD-ROM) : réflexion (bit 1) ou pas de réflexion (bit 0)
- Fréquences (Modem) : dans un signal analogique sinusoïdal.

a. Codage des nombres

iv Codage des entiers naturels

1. Utilisation du code binaire pur:

- L'entier naturel (positif ou nul) est représenté en base 2,
- Les bits sont rangés selon leur poids, on complète à gauche par des 0.

► **Exemple :** sur un octet, $10_{(10)}$ se code en binaire pur : $0\ 0\ 0\ 0\ 1\ 0\ 1\ 0_{(2)}$

2. Etendu du codage binaire pur :

- Sur n bits on peut coder des nombres de **[0 à $2^n - 1$]**
- Sur 1 octet (8 bits) : codage des nombres de 0 à 2^8 - 1 = 255
- Sur 2 octets (16 bits) : codage des nombres de 0 à 2^{16} - 1 = 65535
- Sur 4 octets (32 bits) : codage des nombres de 0 à 2^{32} - 1 = 4 294 967 295

3. Arithmétique en base 2

- Les opérations sur les entiers s'appuient sur des tables d'addition soustraction et de multiplication :

Addition

0	0	**0**
0	1	**1**
1	0	**1**
1	1	(1) **0**

Retenue

Soustraction

0	0	**0**
0	1	(-1) **0**
1	0	**1**
1	1	**0**

Retenue Négatif

Multiplication

0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

►**Exemple (Addition) :**

Addition binaire (8 bits)

1 0 0 1 0 1 1 0
+ 0 1 0 1 0 1 0 1

1 1 1 0 1 0 1 1

Addition binaire (8 bits) avec (débordement ou *overflow*) :

1 0 0 1 0 1 1 0
+ 0 1 1 1 0 1 0 1

1 0 0 0 0 1 0 1 1 ⇒ *Overflow*

►**Exemple (Soustraction) :**

		-1	-1	-1	-1	-1	-1	
	1	1	10	10	11	10	10	10
–					1	1	0	1
	1	0	1	1	1	0	1	1

► **Exemple (Multiplication)**

Multiplication binaire

1 0 1 1 (4 bits)
× 1 0 1 0 (4 bits)

0 0 0 0
1 0 1 1 .
0 0 0 0 .
1 0 1 1 .
0 1 1 0 1 1 1 0

Sur 4 bits le résultat est faux
Sur 7 bits le résultat est juste
Sur 8 bits on complète à gauche par un 0

v ***Codage des entiers signé***

Il existe au moins trois façons pour coder un entier relatif (signé) :

- **Code binaire signé** (par signe et valeur absolue).
- **Code complément à 1**.
- **Code complément à 2** (utilisé sur ordinateur).

1. Binaire en signe et valeur absolue

➢ Le bit le plus significatif est utilisé pour représenter le signe du nombre :

- Si le bit le plus fort = 1 alors nombre négatif
- Si le bit le plus fort = 0 alors nombre positif

➢ Les autres bits codent la valeur absolue du nombre.

► **Exemple :** Sur 8 bits, codage des nombres -24 et -128 en (sva)

-24 est codé en binaire signé par : 1 0 0 1 1 0 0 0 $_{(sva)}$

-128 **hors limite** nécessite 9 bits au minimum.

➢ Etendu de codage :

- Avec n bits, on code tous les nombres entre **$[-(2^{n-1}-1)$ et $(2^{n-1}-1)]$.**
- Avec 4 bits : -7 jusqu'à +7.

➢ Limitations du binaire signé :

- Deux représentations du zéro : +0 et -0
- Sur 4 bits : +0 = $0000_{(bs)}$, -0 = $1000_{(bs)}$
- Multiplication et l'addition sont moins évidentes.

2. Code complément à 1

Aussi appelé Complément Logique (CL) ou Complément Restreint (CR) :

- Les nombres positifs sont codés de la même façon qu'en binaire pur.
- Un nombre négatif est codé en **inversant** chaque bit de la représentation de sa valeur absolue.

➢ Le bit le plus significatif est utilisé pour représenter le signe du nombre :

- Si le bit le plus fort = 1 alors nombre négatif.
- Si le bit le plus fort = 0 alors nombre positif.

► **Exemple :** -24 en complément à 1 sur 8 bits

|-24| en binaire pur ⇒ 0 0 0 1 1 0 0 0 $_{(2)}$, puis on inverse les bits ⇒ 1 1 1 0 0 1 1 1 $_{(cà1)}$

➢ Etendu de codage :

- Avec n bits, on code tous les nombres entre **$[-(2^{n-1}-1)$ et $(2^{n-1}-1)]$.**
  - Avec 8 bits : -255 jusqu'à +255.

➢ Limitations du complément à 1 :

- Deux codages différents pour 0 (+0 et -0)
- Sur 8 bits : +0 = $0\ 0\ 0\ 0\ 0\ 0\ 0\ 0_{(cà1)}$ et -0 = $1\ 1\ 1\ 1\ 1\ 1\ 1\ 1_{(cà1)}$
- Multiplication et l'addition sont moins évidentes.

3. Code complément à 2

Aussi appelé Complément Vrai (CV) :

- Les nombres positifs sont codés de la même manière qu'en binaire pur.
- Un nombre négatif est codé en ajoutant la valeur 1 à son complément à 1.

➢ Le bit le plus significatif est utilisé pour représenter le signe du nombre.

► **Exemple :** -24 en complément à 2 sur 8 bits

24 est codé par $0\ 0\ 0\ 1\ 1\ 0\ 0\ 0_{(2)}$

-24 ⇒ $1\ 1\ 1\ 0\ 0\ 1\ 1\ 1_{(cà1),}$ puis on lui additionne 1 donc -24 est codé par $1\ 1\ 1\ 0\ 1\ 0\ 0\ 0_{(cà2)}$

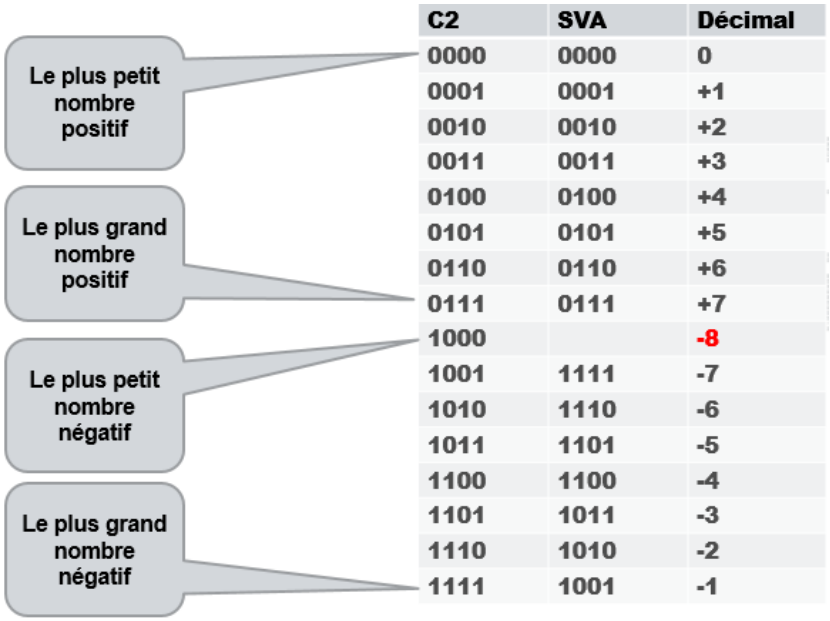
➢ Un seul codage pour 0. Par exemple sur 8 bits :

- +0 est codé par $00000000_{(cà2)}$
- -0 est codé par $11111111_{(cà1)}$ ⇒ donc -0 sera représenté par $00000000_{(cà2)}$

➢ Etendu de codage :

- Avec n bits, on peut coder de **$[-(2^{n-1})$ à $(2^{n-1}-1)]$**
- Sur 1 octet (8 bits), codage des nombres de -128 à 127
  - +0 = 00000000 -0 = 00000000
  - +1 = 00000001 -1 = 111111111
  - +127 = 01111111 -128 = 10000000

► **Exemple :** sur 4 bits Domaine couvert $=[-2^{n-1}, +2^{n-1}-1] =[-2^{4-1}, +2^{4-1}-1] =[-8, +7]$.



C2	SVA	Décimal
0000	0000	0
0001	0001	+1
0010	0010	+2
0011	0011	+3
0100	0100	+4
0101	0101	+5
0110	0110	+6
0111	0111	+7
1000		-8
1001	1111	-7
1010	1110	-6
1011	1101	-5
1100	1100	-4
1101	1011	-3
1110	1010	-2
1111	1001	-1

vi ***Codage des nombres réels***

Les formats de représentations des nombres réels sont :

- **Format virgule fixe** (utilisé par les premières machines)
  - Possède une partie « entière » et une partie « décimale » séparés par une virgule.
  - La position de la virgule est fixe d'où le nom.
    - ► **Exemple :** $54{,}25_{(10)}$; $10{,}001_{(2)}$; $A1{,}F0B_{(16)}$

- **Format virgule flottante** : (utilisé actuellement sur machine)
  - o Défini par : **± m . b ^{e}**
    - un signe + ou –
    - une mantisse **m** (en virgule fixe)
    - un exposant **e** (un entier relative)
    - une base **b** (2,8,10,16,…)

► **Exemple :** $0{,}5425.10^{2}_{(10)}$; $10{,}1.2^{-1}_{(2)}$; $A0{,}B4.16^{-2}_{(16)}$

1. Codage en Virgule Fixe

Etant donné une base b, un nombre X est représenté, en format virgule fixe, par :

- $X = a_{n-1}\, a_{n-2} \ldots a_1\, a_0\, ,\, a_{-1}\, a_{-2} \ldots a_{-p}$ (b)
- a_{n-1} est le chiffre de poids fort (MSB[5]).
- a_{-p} est le chiffre de poids faible (LSB[6]).
- n est le nombre de chiffre avant la virgule.
- p est le nombre de chiffre après la virgule.
- La valeur de X en base 10 est : $X = \left(\sum_{i=0}^{n-1} a_i . b^i\right)$

► **Exemple :** $101{,}01_{(2)} = 1\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} = 5{,}25_{(10)}$

✓ Changement de base 10 → 2

- Le passage de la base 10 à la base 2 est défini par :
  - Une partie entière codée sur p bits (division successive par 2)
  - Une partie décimale codée sur q bits en multipliant par 2 successivement jusqu'à ce que la partie décimale soit nulle ou le nombre de bits q est atteint.

► **Exemple :** $4{,}25_{(10)}$ = ? (2) format virgule fixe

Partie entière : $4_{(10)} = 100_{(2)}$

Partie décimale : 0,25 x 2 = 0,5 → 0 0,5

x 2 = 1,0 → 1

⇒ donc $4{,}25_{(10)} = 100{,}01_{(2)}$

2. Codage en Virgule Flottante

Un nombre réel est représenté en virgule flottante sous la forme : **± M × 2 ^{E}**, où M est la mantisse (virgule fixe) et E l'exposant (signé). Coder en base 2 au format virgule flottante, revient à coder le signe, la mantisse et l'exposant.

► **Exemple :** Codage en base 2, format virgule flottante, du nombre 3,25

$3{,}25_{(10)} = 11{,}01_{(2)}$ (en virgule fixe)

$= 1{,}101 \times 2^{1}_{(2)}$ → on décale la virgule par 1 bit à gauche.

$= 110{,}1 \times 2^{-1}_{(2)}$ → on décale la virgule par 1 bit à droite.

Problème : il y a différentes manières de représenter E et M.

[5] Most significant bit.

[6] Less significant bit.

⇒ **Normalisation** : afin d'avoir la même représentation, tout nombre doit d'abord être normalisé sous la forme : **± 1,M × 2^{Eb}**

- Le signe est codé sur 1 bit ayant le poids fort (**S**):
  - Le signe – : bit 1
  - Le signe + : bit 0
- Exposant biaisé (**Eb**) :
  - Placé avant la mantisse pour simplifier la comparaison.
  - Codé sur p bits et biaisé pour être positif (ajout de **2^{p-1}-1**).
- Mantisse normalisée (**M**) :
  - Normalisée : la virgule est placée après le bit à 1 ayant le poids fort.
  - M est codé sur q bits ○ Exemple : 11,01 ⇒ 1,101×2^1 donc M =101

S	Eb	M
1 bit	p bits	q bits

➢ **Standard IEEE 754 (1985)**

C'est la norme la plus employée actuellement pour le calcul des nombres à virgule flottante avec les CPU. Elle définit les nombres réels sur les formats :

- **Simple précision,** codé sur 32 bits :
  - 1 bit de signe (**S**).
  - 8 bits pour l'exposant biaisé (**Eb**).
  - 23 bits pour la mantisse (**M**).

S	Eb	M
1bit	8 bits	23 bits

- **Double précision,** codé sur 64 bits :
  - 1 bit de signe (**S**).
  - 11 bits pour l'exposant biaisé (**Eb**).
  - 52 bits pour la mantisse (**M**).

S	Eb	M
1bit	11 bits	52 bits

- **Précision étendue,** codé sur 80 bits :
  - 1 bit de signe (**S**).
  - 15 bits pour l'exposant biaisé (**Eb**).
  - 64 bits pour la mantisse (**M**).

S	Eb	M
1bit	15 bits	64 bits

► **Exemple :** Conversion décimale - IEEE754 (Codage d'un réel)

$35,5_{(10)}$ = ? (2) (IEEE 754 simple précision)

Nombre positif, ⇒ donc S = 0

$35,5_{(10)} = 100011,1_{(2)}$ (virgule fixe)

$= 1,000111 \times 2^{5}{}_{(2)}$ (virgule flottante)

Exposant = Eb-127 = 5, donc Eb = 132

1,M = 1,000111 donc M = 00011100...

0 10000100 00011100000000000000000 (IEEE 754 SP)

S Eb M

► **Exemple :** Conversion IEEE754 – Décimale (Evaluation d'un réel)

0 10000001 11100000000000000000000 (IEEE 754 SP)

S Eb M

S = 0, donc nombre positif

Eb = 129, donc exposant = Eb-127 = 2

1,M = 1,111

+ $1,111 \times 2^{2}{}_{(2)} = 111,1_{(2)} = 7,5_{(10)}$

► Caractéristiques des nombres flottants au standard IEEE :

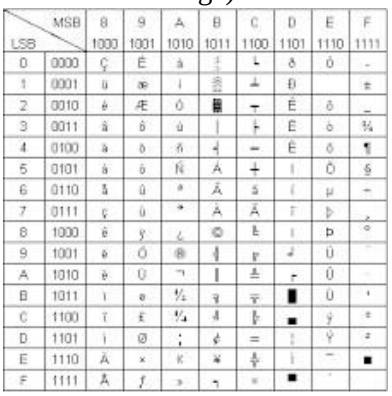
	Simple précision	***Double précision***	***Précision étendue***
Bit de signe	1	1	1
Bits d'exposant	8	11	15
Bits de mantisse	23	52	64
Nombre total de bits	32	64	80
Codage de l'exposant	Excédant 127	Excédant 1023	Excédant 16383
Variation de l'exposant	-126 à +127	-1022 à +1023	- 16383 à + 16383
Plus petit nombre normalisé	2^{-126}	2^{-1022}	2^{-16382}
Plus grand nombre normalisé	Environ 2^{+128}	Environ 2^{+1024}	Environ 2^{+16384}

b. Codage des caractères

Les caractères peuvent être soit Alphabétique (A-Z, a-z), soit numérique (0, …, 9), des signes de ponctuation, ou des caractères spéciaux (&, $, %, …). Le codage des caractères revient à créer une table de correspondance entre les caractères et des nombres [24].

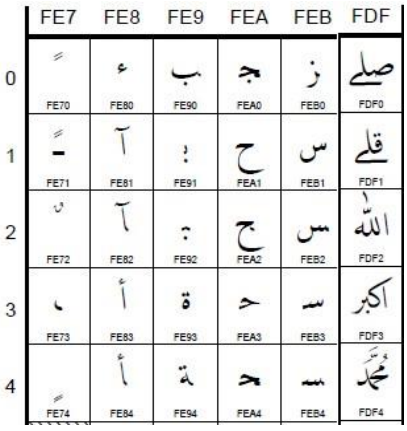
➢ Les Standards:

- Code (ou Table) **ASCII** (American Standard Code for Information Interchange) :
  - 7 bits pour représenter 128 caractères (0 à 127)
  - 48 à 57 : chiffres dans l'ordre (0, 1, …, 9)
  - 65 à 90 : les alphabets majuscules (A, …, Z)
  - 97 à 122 : les alphabets minuscule (a, …, z)



- Code **ASCII Etendu** :
  - 8 bits pour représenter 256 caractères (0 à 255)
  - Code les caractères accentués : à, é, è, …etc.
  - Compatible avec ASCII.

- Code **Unicode** (mis au point en 1991) ○ 16 bits pour représenter 65 536 caractères (0 à 65 535).
  - Compatible avec ASCII.
  - Code la plupart des alphabets : Arabe, Chinois, …etc.
  - On en a défini environ 50 000 caractères pour l'instant.

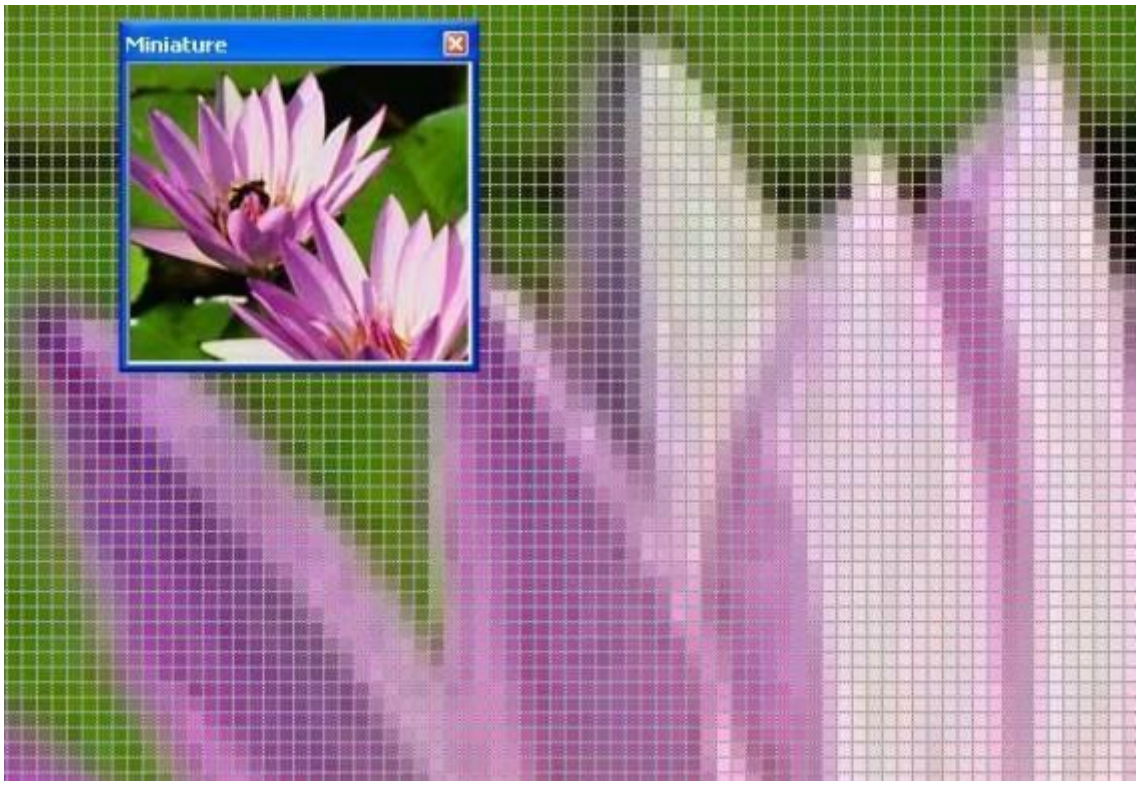


c. Codage d'une image

Le principe du codage d'une image :

- Tout commence par découper l'image en des petits carrés c'est en quelque sorte poser une grille (aussi serrée que possible) sur l'image.
- Deux nombres seront importants pour décrire cette grille : le nombre de petits carrés en largeur et ce même nombre en hauteur.
- Plus ces nombres sont élevés, plus la surface de chaque petit carré est petite et plus le dessin tramé sera proche de l'originale.

On obtient donc pour toute l'image un quadrillage comme celui montré ci-dessous pour une partie.



Il ne reste plus qu'à en déduire une longue liste d'entiers :

- Le nombre de carré sur la largeur.
- Le nombre de carré sur la hauteur.
- Suite de nombres pour coder l'information (Couleur) contenue dans chaque petit carré qu'on appelle PIXEL (**PIC**ture **EL**ement) :
    - Image en noir et blanc → 1 bit pour chaque pixel.
    - Image avec 256 couleurs → 1 octet (8 bits) pour chaque pixel.
    - Image en couleur vrai (True Color : 16 millions de couleurs) → 3 octets (24 bits) pour chaque pixel.

La manière de coder un dessin en série de nombres s'appelle une représentation **BITMAP**.

L'**infographie** est le domaine de l'informatique concernant la création et la manipulation des images numériques.

La **définition** : détermine le nombre de pixel constituant l'image. Une image possédant 800 pixels en largeur et 600 pixels en hauteur aura une définition notée 800x600 pixels.

La **profondeur** ou la dynamique d'une image est le nombre de bits utilisé pour coder la couleur de chaque pixel.

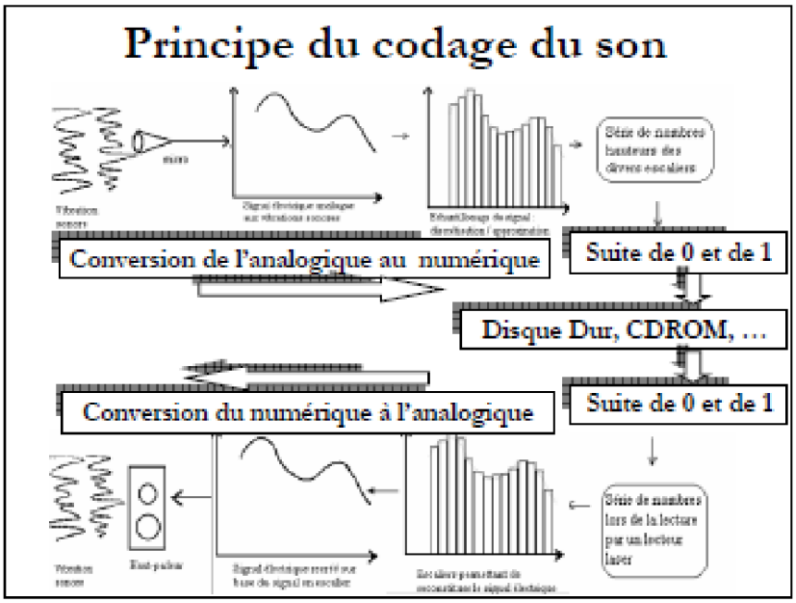
Le **poids** d'une image (exprimé en Ko ou en Mo) : est égal à son nombre de pixels (définition) que multiplie le poids de chacun des pixels (profondeur).

e. Codage du son

Un son est une vibration mécanique se propageant dans l'air ou dans un autre milieu (fluide, solide…). Lors de la numération, il faut transformer le signal analogique (continu) en une suite de nombres qui seront traités par l'ordinateur. C'est le rôle du convertisseur analogique/numérique.

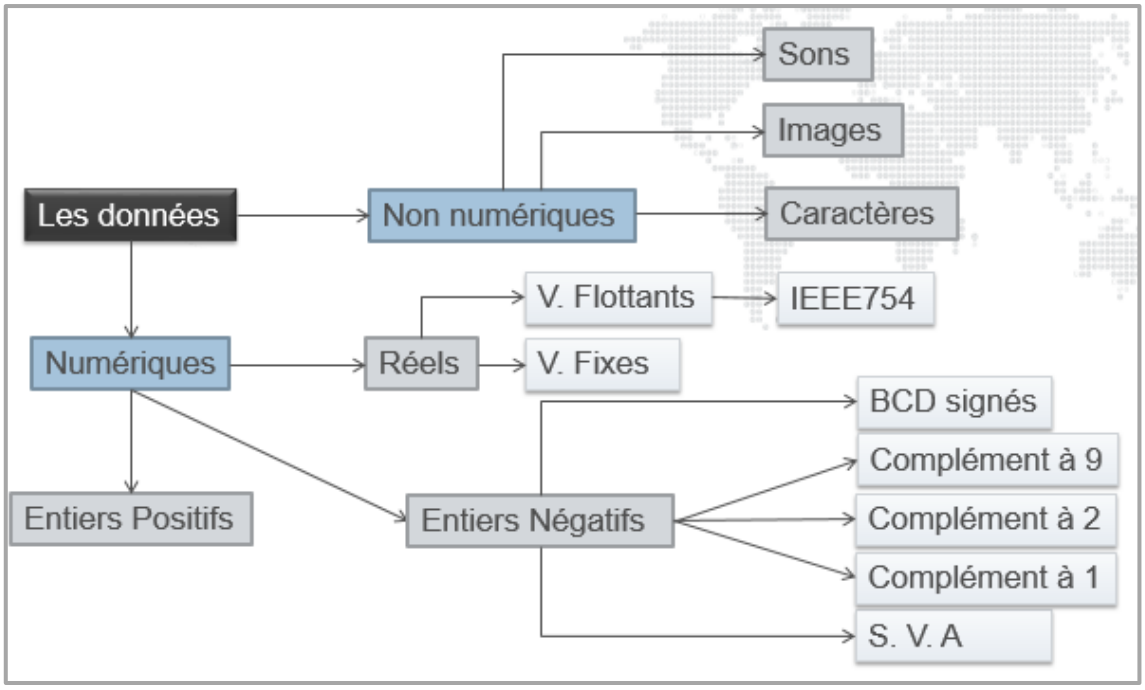
La numérisation d'un son se réalise en deux étapes :

- **L'échantillonnage** : consiste à prélever périodiquement des échantillons d'un signal analogique selon une période que l'on appellera période d'échantillonnage.
- **La quantification** : consiste à affecter une valeur numérique à chaque échantillon prélevé.



Les Codecs, abréviation de codeur/décodeur, sont des logiciels qui permettent de réaliser ce codage/décodage du son dans un ordinateur. Ces logiciels s'appuient sur les éléments matériels disponibles dans la carte son.

- ***Synthèse :***



Série de TD N°3

(Comptage binaire)

Exercice 1 : (Conversion de bases)

Exprimez le nombre décimal 100 dans toutes les bases de 2 à 9 et en base 16.

Exercice 2 : (Représentation des entiers)

Exprimez les nombres décimaux 94, 141, 163 et 197 en base 2, 8 et 16.

Donnez sur 8 bits les représentations « binaire signé », complément à 1 et complément à 2 des nombres décimaux 45, 73, 84, –99, –102 et –118.

Exercice 3 : (Calcul binaire)

1/ Effectuez la soustraction 122 – 43 dans la représentation en complément à 2 en n'utilisant que l'addition.

2/ Multipliez les nombres binaires 10111 et 1011 ; vérifiez le résultat en décimal.

Exercice 4 : (Représentation des réels - virgule fixe)

1/ Coder en virgule fixe au format Q8.8 les nombres suivants : 23.375, 100, 365.2, –54.65.

2/ Décoder le nombre $6C60_{(16)}$ en virgule fixe dans le format Q10.6.

3/ Décoder le nombre $D7EA_{(16)}$ en virgule fixe dans le format Q8.8.

4/ Déterminer le nombre maximum représentable en virgule fixe sur le format Q4.4.

5/ Déterminer le nombre minimal strictement positif représentable sur ce même format.

6/ Déterminer le pas de ce format.

Exercice 5 : (Virgule flottante - Norme IEEE754)

Soit une machine où les nombres réels sont représentés sous la forme ($\pm$ 1,M . 2^{Eb}) sur 32 bits, numérotés de droite à gauche de 0 à 31 avec :

- Une pseudo-mantisse M normalisée sur 23 bits (les bits 0 à 22).
- Un exposant biaisé, représentant une puissance de 2, codé sur 8 bits (les bits 23 à 3).
- Un bit pour le signe de la mantisse (le bit 31).

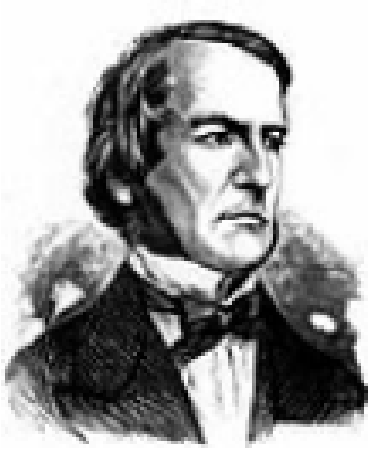
1/ Déterminer le nombre maximum représentable dans ce format.

2/ Déterminer le nombre minimal strictement positif représentable dans ce même format.

3/ Donner en octal, la représentation IEEE754 correspondant au nombre décimal –32,625.

4/ Mettre sous le format *IIIIII*754, les deux nombres $AE8_{(16)}$ et $9D0_{(16)}$ puis calculer leur valeur décimale.

Chapitre V : Logique combinatoire et séquentielle


George Boole
(1815-1864)

L'analyse des circuits logiques nécessite l'utilisation d'une algèbre spécifique, dite « booléenne », fruit des travaux du mathématicien anglais George Boole du 19ème siècle. Ces travaux ont défini un ensemble d'opérateurs de base, ainsi que leurs propriétés, qui constituent une algèbre permettant de concevoir tout type de circuit. La construction de fonctions logiques répondant à des contraintes précises et leur représentation sous forme de circuits électroniques est réalisée par des opérateurs élémentaires [25].

Alors qu'en algèbre classique, les variables et les fonctions peuvent prendre n'importe quelle valeur, ici elles sont limitées aux valeurs 0 et 1. Une fonction de n variables booléennes sera définie à partir de $\{0.\ 1\}^n$ à {0, 1}. Il est également complètement défini par les valeurs qu'il suppose dans les 2^n combinaisons possibles de ses variables d'entrée, comme chaque valeur de fonction ne peut être que 0 ou 1, il existe 2^{2^n} fonctions différentes de n variables. On peut donc décrire une fonction donnée (avec n variables) en expliquant ses 2^n valeurs, par exemple, à partir de sa table de vérité. Il s'agit d'un tableau de 2^n lignes, qui répertorie pour chaque combinaison possible de variables d'entrée (une ligne) la valeur assumée par la fonction.

1. Les circuits logiques

Un circuit logique est un Circuit dans lequel seules 2 valeurs logiques sont possibles : 0 ou 1. Il est réalisé par un circuit électrique (transistors) [25]:

- Une faible tension représente la valeur binaire 0.
- Une tension élevée représente 1.

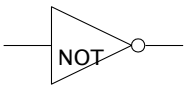
Les composants de base d'un circuit logique sont : les **portes logiques**.

► Porte logique :

- Permet de combiner les signaux binaires.
- Reçoit en entrée une ou plusieurs valeurs binaires (souvent 2).
- Renvoie une unique valeur binaire en sortie.

a. Fonctions logiques élémentaires

i ***Fonction NON (NOT)***



- Si la valeur d'entré est 1, alors la sortie vaut 0.
- Si la valeur d'entrée est 0, alors la sortie vaut 1.

ii ***Fonction OU (OR)***

$$S = f(a,b) = a + b$$

a	b	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

iii ***Fonction ET (AND)***

$$S = f(a,b) = a \times b = ab$$

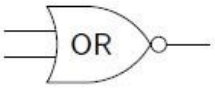
a	b	S
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

iv ***Fonction OU-exclusif (XOR)***

$$\begin{aligned} S = f(a,b) &= a \oplus b \\ &= (a+b)(\overline{ab}) \\ &= (a+b)(\bar{a}+\bar{b}) \\ &= a\bar{a} + a\bar{b} + b\bar{a} + b\bar{b} \\ &= a\bar{b} + b\bar{a} \\ &= \overline{\overline{a\bar{b}}} + \overline{\overline{b\bar{a}}} \\ &= \overline{\overline{a\bar{b}}\,\overline{b\bar{a}}} \end{aligned}$$

a	b	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

v ***Fonction NON-OU (NOR)***



$$S = f(a,b) = \overline{a+b}$$

a	b	S
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

vi ***Fonction NON-ET (NAND)***

$$S = f(a,b) = \overline{a.b} = \overline{ab}$$

a	b	S
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

► Exemple des fonctions booléennes de 2 variables:

$f(a,b)$	00	01	10	11
0	0	0	0	0
ab	0	0	0	1
$a\bar{b}$	0	0	1	0
a	0	0	1	1
$\bar{a}b$	0	1	0	0
b	0	1	0	1
$a \oplus b$	0	1	1	0
$a+b$	0	1	1	1

$f(a,b)$	00	01	10	11
$\overline{a+b}$	1	0	0	0
$\overline{a \oplus b}$	1	0	0	1
$\bar{b}$	1	0	1	0
$a+\bar{b}$	1	0	1	1
$\bar{a}$	1	1	0	0
$\bar{a}+b$	1	1	0	1
$\overline{ab}$	1	1	1	0
1	1	1	1	1

b. *Règles de calcul*

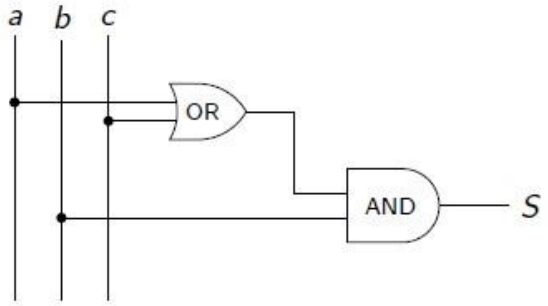
Commutativité	Associativité	Distributivité
$a+b=b+a$ $ab=ba$ $a \oplus b=b \oplus a$	$a+(b+c)=(a+b)+c=a+b+c$ $a(bc)=(ab)c=abc$ $a \oplus (b \oplus c)=(a \oplus b) \oplus c=a \oplus b \oplus c$	$a+(bc)=(a+b)(a+c)$ $a(b+c)=(ab)+(ac)=ab+ac$ $a(b \oplus c)=(ab) \oplus (ac)=ab \oplus ac$
Elément neutre	Elément absorbant	Idempotence
$a+0=a$ $1.a=a$ $a \oplus 0=a$	$a+1=1$ $0.a=0$	$a+a=a$ $aa=a$
Complémentaire	Lois de Morgan	Divers
$a+\bar{a}=1$ $a\bar{a}=0$ $\overline{aa}=\bar{a}$ $\overline{a+a}=\bar{a}$ $\bar{\bar{a}}=a$ $a \oplus \bar{a}=1$ $a \oplus 1=\bar{a}$	$\overline{ab}=\bar{a}+\bar{b}$ $\overline{a+b}=\bar{a}\bar{b}$	$a+ab=a(a+b)=a$ $a+(\bar{a}b)=a+b$ $a(\bar{a}+b)=ab$ $a \oplus a=0$ $a \oplus \bar{b}=\bar{a} \oplus b=\overline{a \oplus b}$ $\bar{a} \oplus \bar{b}=a \oplus b$ $a \oplus b=a\bar{b}+\bar{a}b$ $\overline{a \oplus b}=ab+\bar{a}\bar{b}$

Table 1 : Tableau récapitulatif des règles de calcul logique [1].

c. Construction et optimisation

i ***Du circuit logique à la table de vérité***

Soit le schéma d'un circuit logique :



a	b	c	$a+c$	$S=b(a+c)$
0	0	0	0	**0**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	1	1	**1**
1	0	0	1	**0**
1	0	1	1	**0**
1	1	0	1	**1**
1	1	1	1	**1**

⇒ La table de vérité correspondante.

ii ***De la table de vérité au circuit logique***

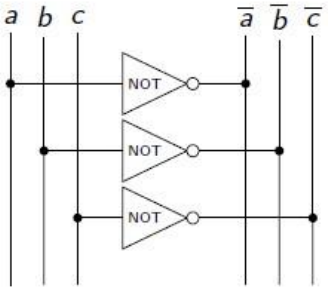
1) Ecrire l'équation de la fonction à partir de sa table de vérité

a	b	c	S
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

- Pour chaque cas où la sortie vaut 1 ajouter un « *minterme* » à la fonction.
- Un *minterme* est un AND de toutes les variables d'entrée de la fonction (éventuellement complémentées) :
  - Si la variable d'entrée vaut 1, elle est écrite directement dans le *minterme*.
  - Si la variable d'entrée vaut 0, elle est complémentée dans le *minterme*.

⇒ $S = f(a,b,c) = \bar{a}b\bar{c} + a\bar{b}c + abc$

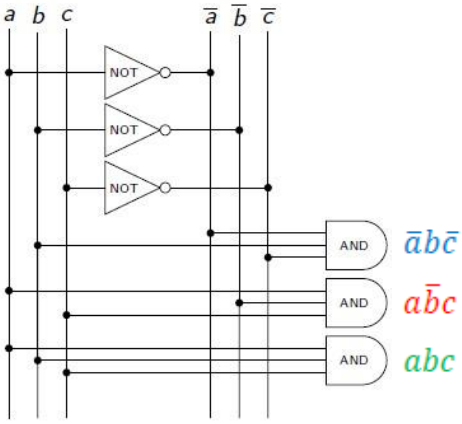
2) Réaliser la négation de toutes les variables d'entrée



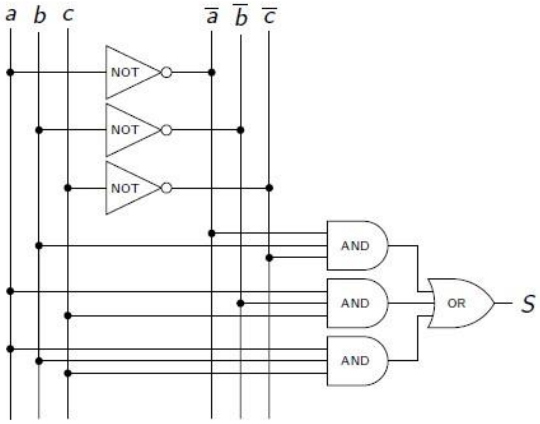
3) Construire une porte AND pour chacun des *mintermes*.

4) Etablir le câblage des portes OR avec les entrées appropriées.

$$S = f(a,b,c) = \bar{a}b\bar{c} + a\bar{b}c + abc \quad \Rightarrow$$



5) Réunir l'ensemble des sorties des portes AND vers une porte OR, dont la sortie et le résultat de la fonction.



► Remarque : ce circuit n'est pas optimal pour la table de vérité initiale. Il faut **simplifier** la fonction pour minimiser le nombre portes logiques.

iii ***Simplification***

- Diminuer le nombre d'opérateurs.
- Diminuer le nombre de portes logiques (et donc le coût).

Il y a deux approches de simplification :

1) Méthode algébrique (algèbre de Boole).

$$\begin{aligned} f(a,b,c) &= \bar{a}bc + a\bar{b}c + abc + abc \\ &= (\bar{a}b + a\bar{b})c + ab(c + \bar{c}) \\ &= (a+b)(\bar{a}+\bar{b})c + ab \\ &= (ac + bc)\overline{ab} + ab \\ &= (ab + ac + bc)(\overline{ab} + ab) \\ &= ab + ac + bc \end{aligned}$$

► Exemple : la fonction majoritaire ⇒ donne 1 en sortie si la majorité des bits en entrée sont des 1 sinon 0.

a	*b*	*c*	*S*
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

2) Méthode des tableaux de Karnaugh

- Permet de visualiser une fonction et d'en tirer naturellement une écriture simplifiée.
- Représentation de toutes les combinaisons d'états possibles pour un nombre de variables donné.
- Outil graphique qui permet de simplifier de manière méthodique des expressions booléennes.
- Exploite le codage de l'information et la notion d'adjacence.

✓ Principe :

- Mettre en évidence sur un graphique les *mintermes* ou *maxtermes* adjacents.
- Transformer les adjacences logiques en adjacences géométriques.

Table de vérité	vs.	**Tableau de Karnaugh**
1 ligne	⇒	1 case
n variables	⇒	2^n cases

✓ La méthode passe par trois phases :

1) Transcription de la fonction dans un tableau codé.
2) Recherche des adjacences pour simplification.
3) Mise en équations des groupements effectués.

► Exemple : fonction majoritaire

1) Ecrire la table de vérité sous la forme d'un code de Gray (ou binaire réfléchi) : les valeurs des entrées ne diffèrent que d'un seul bit entre chaque ligne.

a	***b***	***c***	***S***
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Table Initiale

⇒

a	***b***	***c***	***S***
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**

Code de Gray

2) Compacter la table : représenter la sortie en fonctions des entrées sous forme d'un tableau à 2 dimensions.

a	b	c	S
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**

⇒

a \ bb	**00**	**01**	**11**	**10**
0	0	0	1	0
1	0	1	1	1

3) Regrouper tous les bits 1 de telle sorte que :

- Les bits 1 d'un groupe sont adjacents ou des bords du tableau.
- La taille des groupes est une puissance de 2.
- Un groupe contient le plus de 1 possible.

a \ bb	**00**	**01**	**11**	**10**
0	0	0	1	0
1	0	1	1	1

4) En déduire la formule et le circuit :

- La formule est la somme (OR) des *mintermes*.
- Un *minterme* est le produit (AND) des variables du même groupe de bits :
  - Des variables qui valent toujours 1 dans ce groupe.
  - Des négations de celles qui valent toujours 0.
  - Les autres variables n'apparaissent pas dans le produit.

a \ bb	**00**	**01**	**11**	**10**
0	0	0	1	0
1	0	1	1	1

⇒ $S = f(a, b, c) = bc + ac + ab$

2. *Les circuits combinatoires*

a. *Définitions*

Les circuits combinatoires sont des circuits logiques combinés afin d'obtenir un nouveau circuit avec une fonction logique plus complexe. Ils sont nés du besoin de circuits logiques à plusieurs entrées et plusieurs sorties, et parfois des sorties qui dépendent d'entrées supplémentaires dites de sélection.

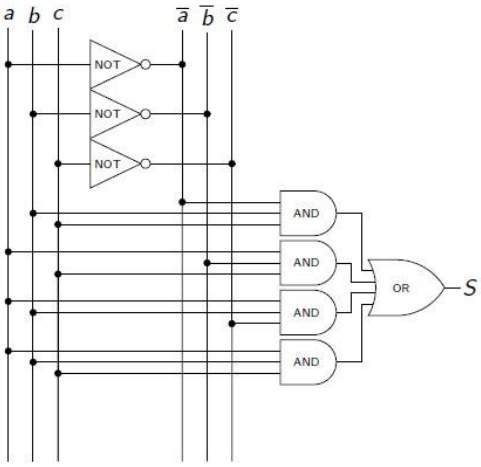
b. *Caractéristiques*

Un circuit combinatoire est caractérisé par :

- **Les entrées** :
  - Les données : ne sont pas des entrées de la table de vérité.
  - Les paramètres : bits de réglage.
  - Les variables d'entrée.
- **La sortie** : pas forcément unique !
  - Fonction logique : une seule valeur en sortie.
  - Circuit : plusieurs fonctions possibles pour obtenir le comportement voulu.
- **Le rôle** de différents éléments :
  - A quoi sert le circuit ?
  - Qu'obtient-on en sortie ?
  - Quel rôle jouent les entrées ?
- **La table de vérité** (une table par fonction).

► Exemple : la fonction majoritaire

a	***b***	***c***	***S***
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**



► Problème : sur un nombre pair d'entrées, une seule sortie ne suffit pas :

- Soit les 0 sont majoritaires (sortie 00)
- Soit les 1 sont majoritaires (sortie 01)
- Soit il n'y a pas de majoritaire (sortie 10)

► Solution : circuit combinatoire à 4 entrées et 2 sorties S_0 et S_1.

a	b	c	d	S_0	S_1
0	0	0	0	**0**	**0**
0	0	0	1	**0**	**0**
0	0	1	0	**0**	**0**
0	0	1	1	**1**	**0**
0	1	0	0	**0**	**0**
0	1	0	1	**1**	**0**
0	1	1	0	**1**	**0**
0	1	1	1	**0**	**1**
1	0	0	0	**0**	**0**
1	0	0	1	**1**	**0**
1	0	1	0	**1**	**0**
1	0	1	1	**0**	**1**
1	1	0	0	**1**	**0**
1	1	0	1	**0**	**1**
1	1	1	0	**0**	**1**
1	1	1	1	**0**	**1**

c. *Les circuits combinatoires de base*

Les circuits combinatoires les plus communs :

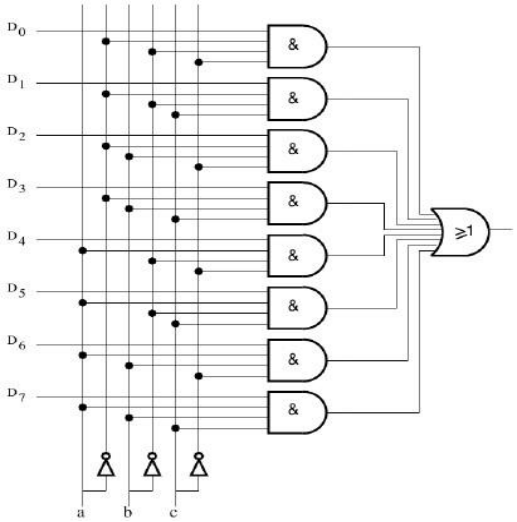
- Le multiplexeur ;
- Le démultiplexeur ;
- Le décodeur ;
- Le comparateur.

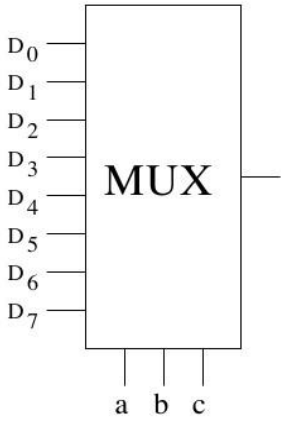
i ***Le multiplexeur $2^n \times n$***

- **Entrées** :
  - 2^n lignes d'entrée (données) : $D_0, \dots, D_{2^n-1}$
  - n lignes de sélection : $a, b, cc, \dots$
- **Sortie** : une seule sortie S.
- **Rôle** : aiguiller la valeur de l'une des 2^n lignes d'entrée vers la sortie S.
  La ligne d'entrée choisie est désignée grâce aux bits de sélection.

a	b	c	S
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

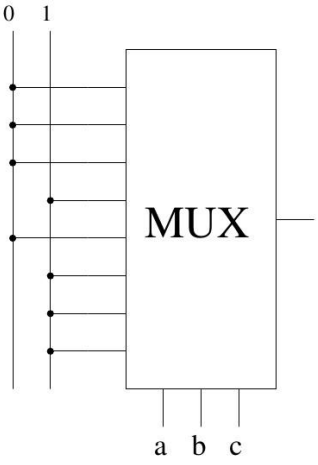
► Câblage du multiplexeur 8 × 3





► Exemple d'utilisation du multiplexeur : La fonction majoritaire avec un multiplexeur.

a	***b***	***c***	***S***
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**

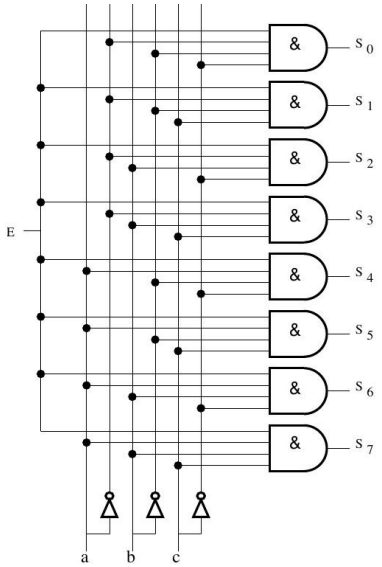


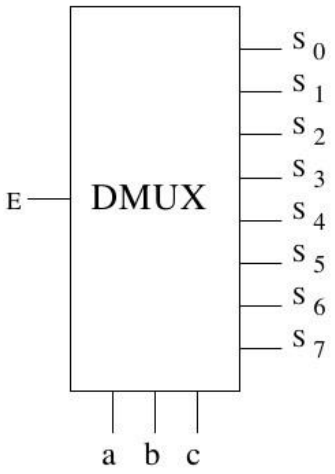
ii ***<u>Le démultiplexeur $2^n \times n$</u>***

- **Entrées** :
  - une lignes d'entrée (donnée) : II
  - n lignes de sélection : $a, b, cc, ...$
- **Sortie** : 2^n lignes de sortie $S_0, \dots, S_{2n-1}$
- **Rôle** : aiguiller l'entrée E vers l'une des 2^n lignes de sortie.
  La ligne de sortie est désignée grâce aux bits de sélection.

a	***b***	***c***	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	E	0	0	0	0	0	0	0
0	0	1	0	E	0	0	0	0	0	0
0	1	0	0	0	E	0	0	0	0	0
0	1	1	0	0	0	E	0	0	0	0
1	0	0	0	0	0	0	E	0	0	0
1	0	1	0	0	0	0	0	E	0	0
1	1	0	0	0	0	0	0	0	E	0
1	1	1	0	0	0	0	0	0	0	E

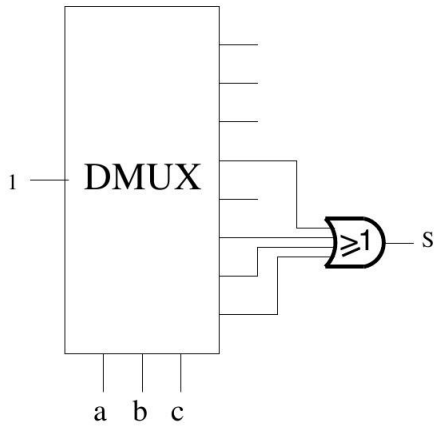
► Câblage du démultiplexeur 8 × 3





► Exemple d'utilisation du démultiplexeur: La fonction majoritaire avec un démultiplexeur.

a	***b***	***c***	***S***
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**

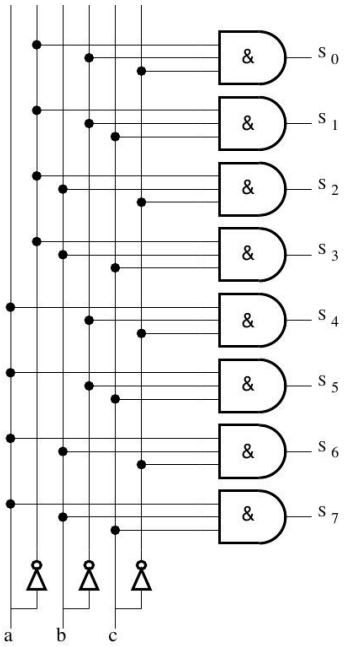


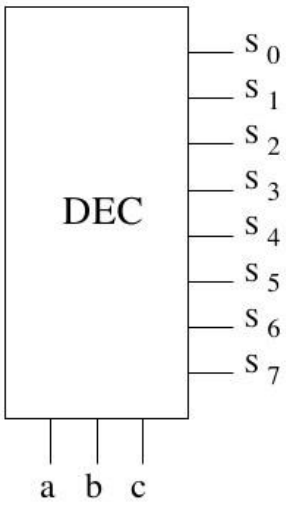
iii ***<u>Le décodeur $2^n \times n$</u>***

- **Entrées** : n lignes de sélection : a, b, c, ...
- **Sortie** : 2^n lignes de sortie S_0, ... , S_{2^n-1}
- **Rôle** : sélectionner (mettre à 1) l'une des 2^n lignes de sortie.
  La ligne de sortie est désignée grâce aux bits de sélection.

a	***b***	***c***	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

► Câblage du décodeur 8 × 3



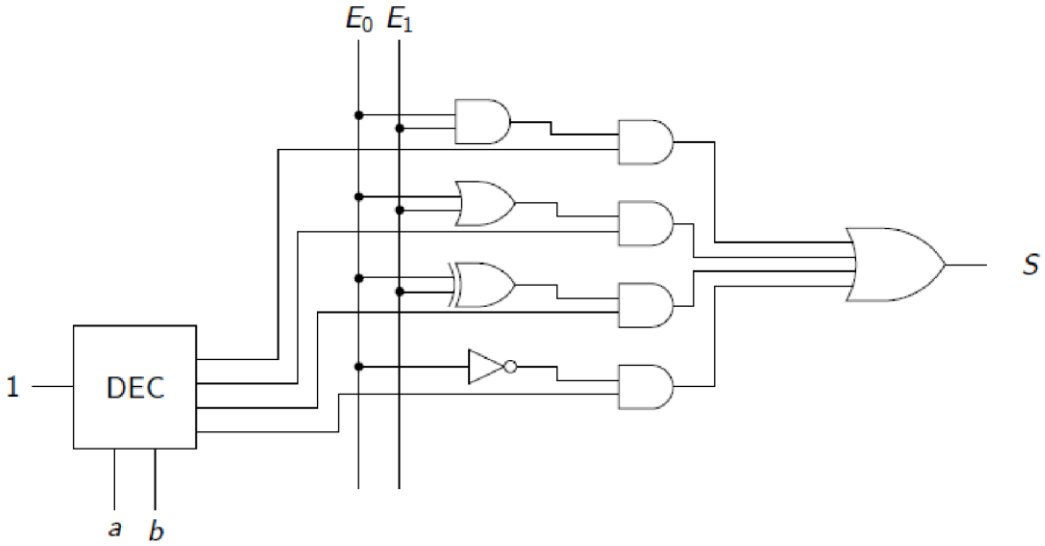


► Exemple d'utilisation d'un décodeur : activation de fonction.

Ce circuit fait, au choix, l'une des 4 fonctions logiques (AND, OR, XOR, NOT) sur les données E_0 et E_1. Le choix de la fonction est déterminé par les valeurs de *a* et *b* selon la table de vérité suivante :

a	***b***	***S***
0	0	$E_0 \times E_1$
0	1	$E_0 + E_1$
1	0	$E_0 \oplus E_1$
1	1	E_0

Réaliser le circuit logique correspondant en utilisant un décodeur.



3. Les circuits arithmétiques

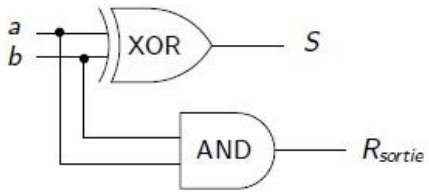
Les circuits arithmétiques sont des circuits combinatoires permettant d'effectuer des opérations arithmétiques sur les nombres en entrée. Les circuits arithmétiques de base sont [25]:

- L'additionneur / Le soustracteur ;
- L'incrémenteur / Le décrémenteur ;
- Le décaleur ;
- L'Unité Arithmétique et Logique (UAL).

a. Le demi-additionneur

- **Entrées** : les 2 bits à additionner a et b.
- **Sorties** :
  - La somme $S = a + b$.
  - La retenue de sortie R_{sortie}.
- **Rôle** : Additionner a et b en conservant la retenue.

a	***b***	***S***	R_{sortie}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

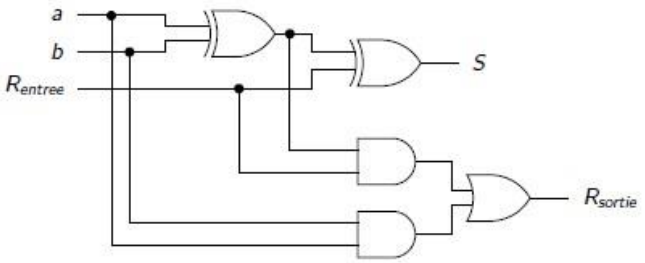


► Problème : si plusieurs additions successives, comment reporter la retenue ?

b. L'additionneur complet

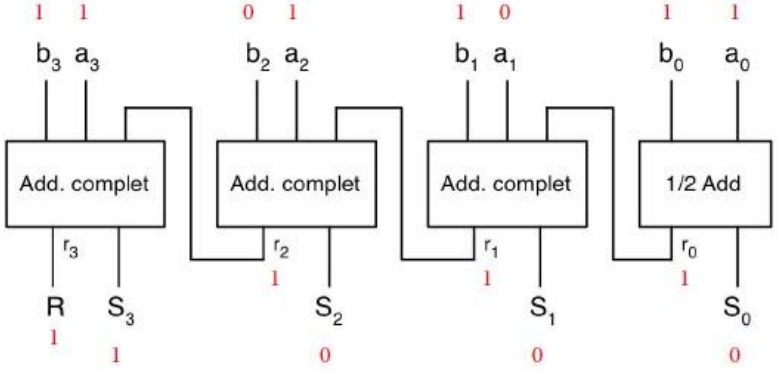
- **Entrées** :
  - Les 2 bits à additionner a et b.
  - La retenue d'entrée R_{sortie}
- **Sorties** :
  - La somme $S = a + b + R_{\text{sortie}}$
  - La retenue de sortie R_{sortie}
- **Rôle** : Additionner a et b en prenant en compte la retenue d'entrée R_{sortie} et en conservant la retenue de sortie R_{sortie}

a	***b***	*R*sortie	***S***	*R*sortie
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



► Exemple : Additionneur 4 bits

Soit 2 nombres à additionner : $A = a_3\ a_2\ a_1\ a_0 = 1\ 1\ 0\ 1_{(2)}$ $B = b_3\ b_2\ b_1\ b_0 = 1\ 0\ 1\ 1_{(2)}$

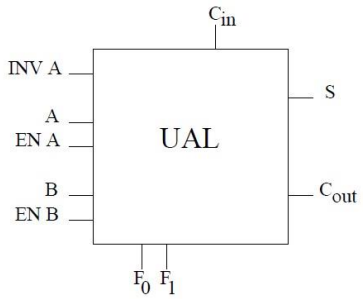


► $S = A + B = S_3\ S_2\ S_1\ S_0 = 1\ 0\ 0\ 0_{(2)}$ avec retenue R = 1.

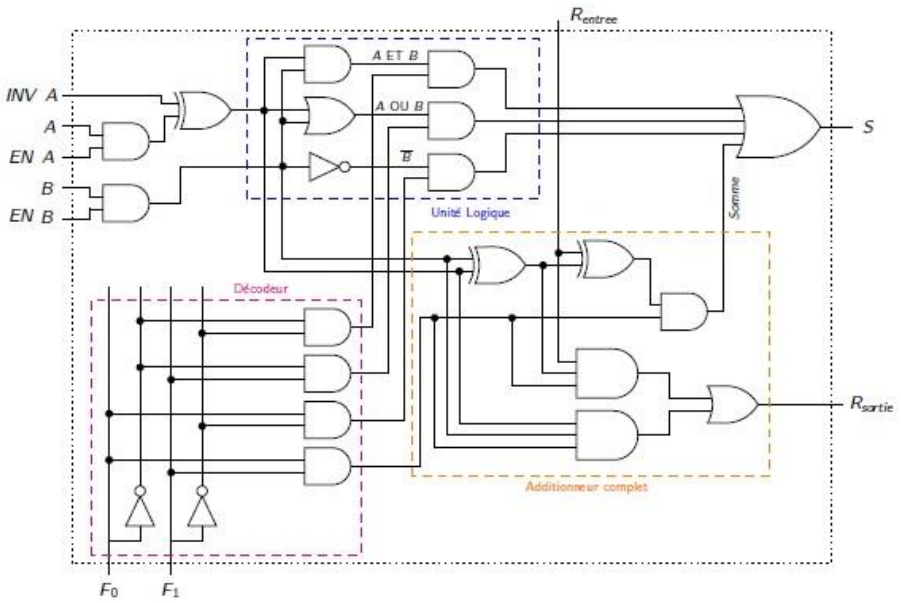
c. L'unité Arithmétique et Logique

L'unité arithmétique et logique est l'organe responsable des calculs arithmétiques effectué par le processeur.

Elle est composée d'un ensemble de circuits combinatoires.



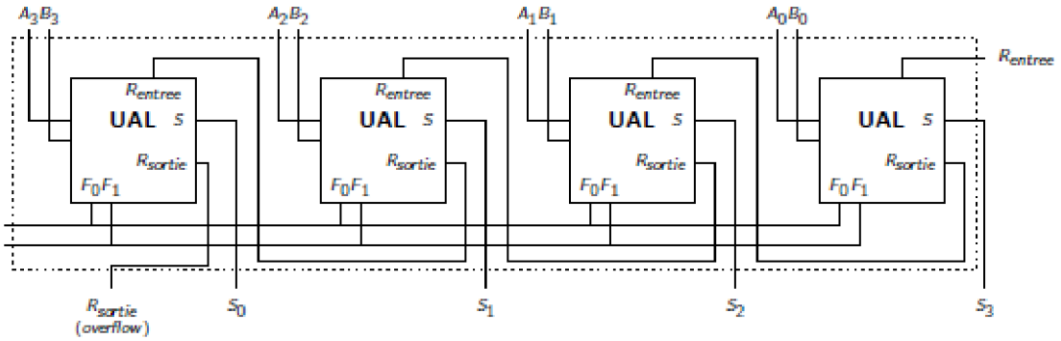
- **Entrées** :
  - A et B : les variables (données).
  - F_0 et F_1 : bits de choix du signal d'activation.
  - $R_{entrée}$ (C_{in}) : la retenue d'entrée.
  - EN A et EN B : les bits inhibiteurs de A et B (optionnel).
  - INV A : pour obtenir A (optionnel).
- **Sorties** :
  - S : le résultat de l'opération ○ R_{sortie} (C_{out}) : la retenue de sortie.
- **Rôle** : Faire l'une des 4 opérations (en fonction des bits d'activation choisis) :
  - A ET B ○ A OU B ○ NON B.
  - A + B + $R_{entrée}$.



Pour 2 bits d'entrée, l'UAL est un circuit qui a peu d'intérêt.

► UAL à n bits :

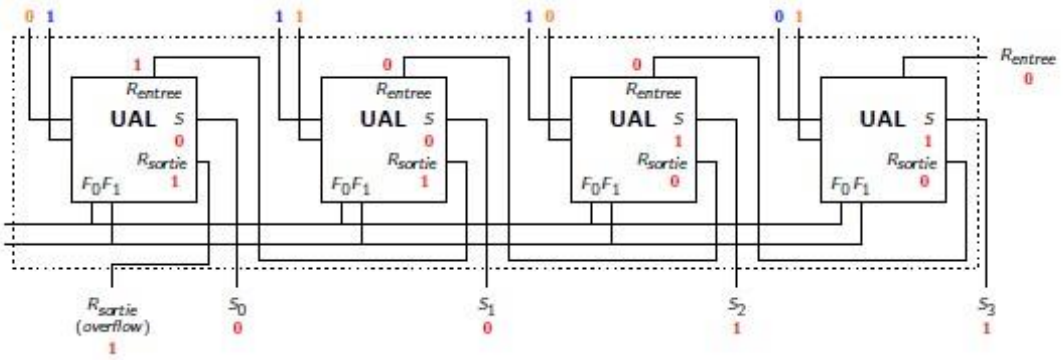
- En connectant les retenues de n UALs, on obtient une UAL n bits telle que :
  - Les opérations logiques sont des opérations bit à bit.
  - Les opérations arithmétiques sont effectuées sur des entiers en complément à 2 sur n bits.



► Exemple UAL à 4 bits :

On souhaite faire l'addition entre A et B (données) telle que :

A = 14 $_{(10)}$ = $a_3\ a_2\ a_1\ a_0$ = 1 1 1 0$_{(2)}$ B = 5 $_{(10)}$ = $b_3\ b_2\ b_1\ b_0$ = 0 1 0 1 $_{(2)}$



S = A + B = $14_{(10)}$ + $5_{(10)}$ = $19_{(10)}$ = 1 1 1 0$_{(2)}$ + 0 1 0 1 $_{(2)}$ = 1 0011 $_{(2)}$

► UAL - Résumé des fonctions :

F_0	F_1	ENA	ENB	$INVA$	$R_{entrée}$	*Fonction*
0	0	1	1	0	0	A Et B
0	1	1	1	0	0	A Ou B
0	1	0	0	0	0	0
0	1	0	1	0	0	B
0	1	1	0	0	0	A
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	B
1	1	1	1	0	0	$A+B$
1	1	0	0	0	1	1
1	1	0	0	1	0	-1
1	1	0	1	0	1	$B+1$
1	1	0	1	1	0	$B-1$
1	1	1	0	0	1	$A+1$
1	1	1	0	1	1	$-A$
1	1	1	1	0	1	$A+B+1$
1	1	1	1	1	1	$B-A$

4. *Les circuits séquentiels*

a. *Définitions*

Dans un circuit combinatoire :
- Les valeurs de sorties, à un instant donné, sont imposées par celles des entrées.
- Le traitement des données est uniquement accessible immédiatement.
- La valeur de la sortie ne dépend que de l'entrée et pas de ce qui s'est passé auparavant.
- Applicable uniquement aux problèmes sans besoin de mémorisation.
- On sait traiter et manipuler l'information, comment la mémoriser ?

⇒ **Circuits séquentiels** (= circuits logiques à mémoire)

Un circuit séquentiel est circuit logique capable de mémoriser des informations. Sa sortie dépend de variables internes en plus des variables d'entrée. L'ensemble des informations mémorisées représente l'état du circuit. La modification des informations mémorisées implique la modification de l'état du circuit [25].

Les circuits séquentiels sont utilisés dans certains types de mémoires, tels que :

- Les bascules ;
- Les bascules latch ;
- Les bascules flip-flop ;
- Les registres.

b. Les bascules

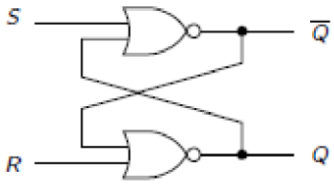
Une bascule :

- Permet de mémoriser un bit.
- « Se souvient » de la valeur que le circuit a enregistrée.
- Est construite avec une ou deux portes logiques NON-OU (ou NON-ET).
- Comporte une ou plusieurs entrées.
- Comporte une ou deux sorties.

La sortie maintient son état même après disparition du signal de commande ⇒ **Logique séquentielle**

i ***La Bascule RS***

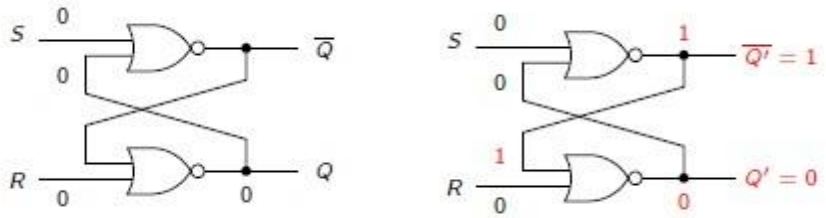
- **Entrée** : deux variables d'entrée :
  - S (Set) pour la mise à l'état 1 de la bascule.
  - R (Reset) pour la mise à l'état 0 de la bascule.
- **Sortie** : deux variables de sortie : Q et $\bar{Q}$



- La valeur de sortie Q_n à l'instant $t = n$ dépend :
  - des variables d'entrées.
  - de la valeur antérieure de la sortie (Q_{n-1}).

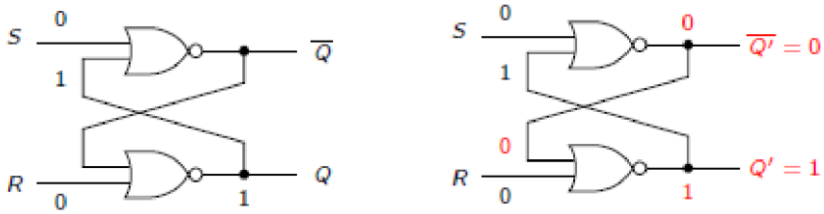
► Bascule RS : états stables

- *Cas 1* : On suppose que $S = R = Q = 0 \Rightarrow \bar{Q}' = 1$ et $Q' = 0$.



Bascule RS à l'état 0

- *Cas 2* : On suppose que $S = R = 0$ et $Q = 1 \Rightarrow \bar{Q}' = 0$ et $Q' = 1$



Bascule RS à l'état 1

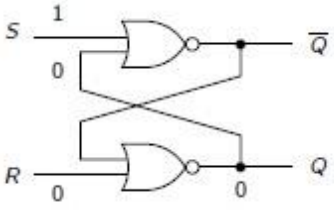
➢ Les deux sorties Q' et $\bar{Q}'$ ne peuvent pas être simultanément à 0.

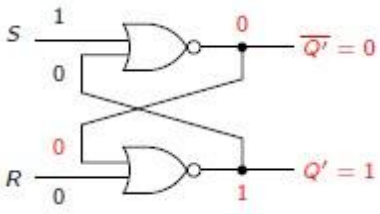
➢ Les deux sorties Q' et $\bar{Q}'$ ne peuvent pas être simultanément à 1.

➢ Pour $S = R = 0$, la bascule offre deux états stables qui dépendent de Q.

► Bascule RS : activation

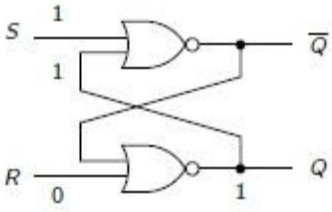
- *Cas 3.1* : On suppose que $S = 1$ et $R = Q = 0 \Rightarrow {}^{-}Q' = 0$ et $Q' = 1$.

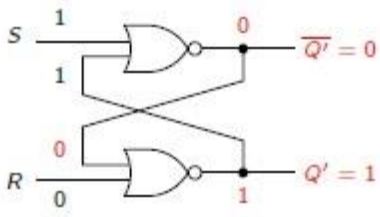




Bascule RS à l'état 1

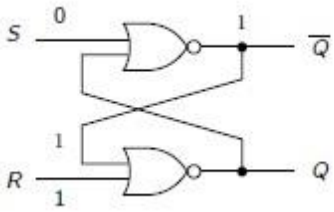
- *Cas 3.2* : On suppose que $S = Q = 1$ et $R = 0 \Rightarrow {}^{-}Q' = 0$ et $Q' = 1$.

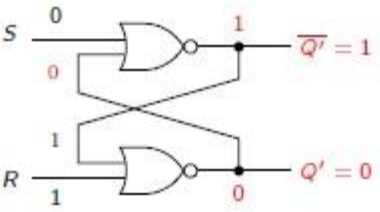




Bascule RS à l'état 1

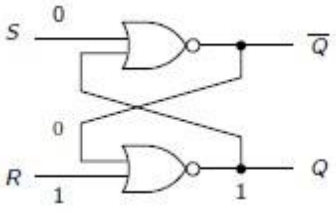
- *Cas 4.1* : On suppose que $S = Q = 0$ et $R = 1 \Rightarrow {}^{-}Q' = 1$ et $Q' = 0$.

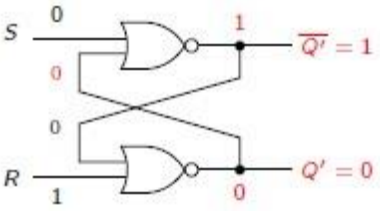




Bascule RS à l'état 0

- *Cas 4.2* : On suppose que $S = 0$ et $R = Q = 1 \Rightarrow {}^{-}Q' = 1$ et $Q' = 0$.





Bascule RS à l'état 0

➢ Si $S = 1$, la bascule RS passe (ou se maintient) à la valeur $Q' = 1$.

➢ Si $R = 1$, la bascule RS passe (ou se maintient) à la valeur $Q' = 0$.

➢ Une bascule RS « se souvient » de l'action antérieure de R ou S.

► Bascule RS : Table de vérité

S	R	Q	$\bar{Q}$	Q'	$\bar{Q}'$
0	0	0	0	x	x
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	x	x
0	1	0	0	x	x
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	x	x
1	0	0	0	x	x
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	x	x
1	1	0	0	x	x
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	x	x

S	R	Q'	$\bar{Q}'$	Etat de la bascule
0	0	Q	$\bar{Q}$	Sorties inchangées
0	1	0	1	RESET : remise à 0
1	0	1	0	SET : mise à1
1	1	0	0	Non utilisé (état instable)

La bascule RS mémorise la valeur des entrées : sa sortie dépend de la dernière entrée mise à 1 (R ou S).

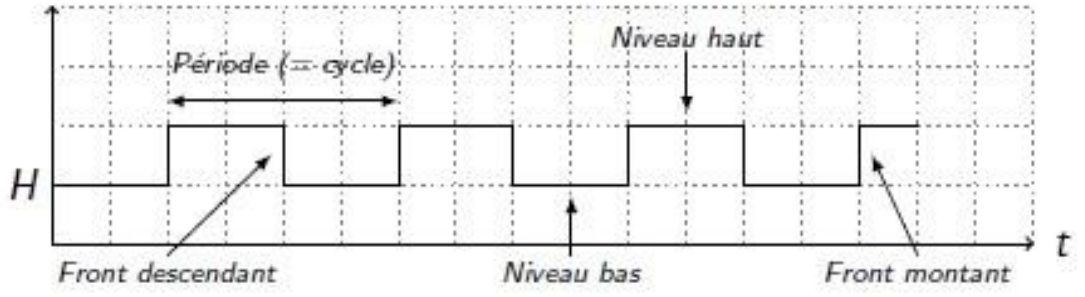
ii ***La Bascule RSH***

L'ordre d'apparition des variables revêt une importance souvent cruciale. La conception des systèmes logiques dépend si une variable arrive avant l'autre ou bien si elles arrivent en même temps.

⇒ Besoin de respecter des relations de séquentialité contraignantes.
⇒ Utilisation d'**horloge** (base de temps ou système de cadencement).

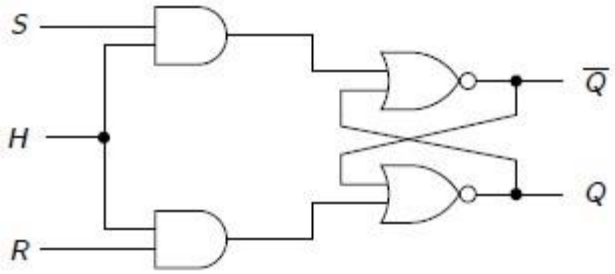
► Horloge :

- Système logique qui émet régulièrement une suite d'impulsions calibrées.
- Intervalle de temps entre deux impulsions = temps de cycle ou période de l'horloge.
- Fréquence des impulsions comprise entre 1 et 100 MHz.
- Temps de cycle compris entre 10 ns à 10 µs.



► Bascule RS + Horloge :

- Permet de faire changer d'état à la bascule à un instant t précis.
- S_n et R_n : états des entrées à l'instant $t = n$.
- Q_{n+1} : sortie au prochain cycle d'horloge (instant $t = n + 1$)

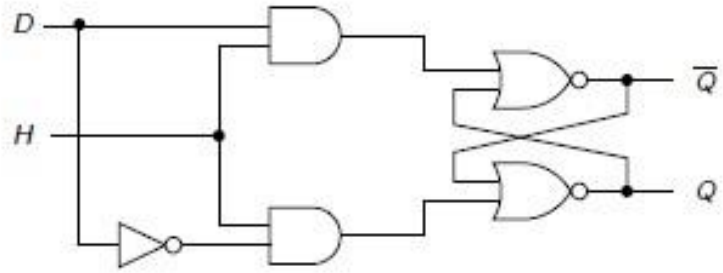


S_n	R_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	?

$Q_{n+1} = S + \bar{R}Q_n$

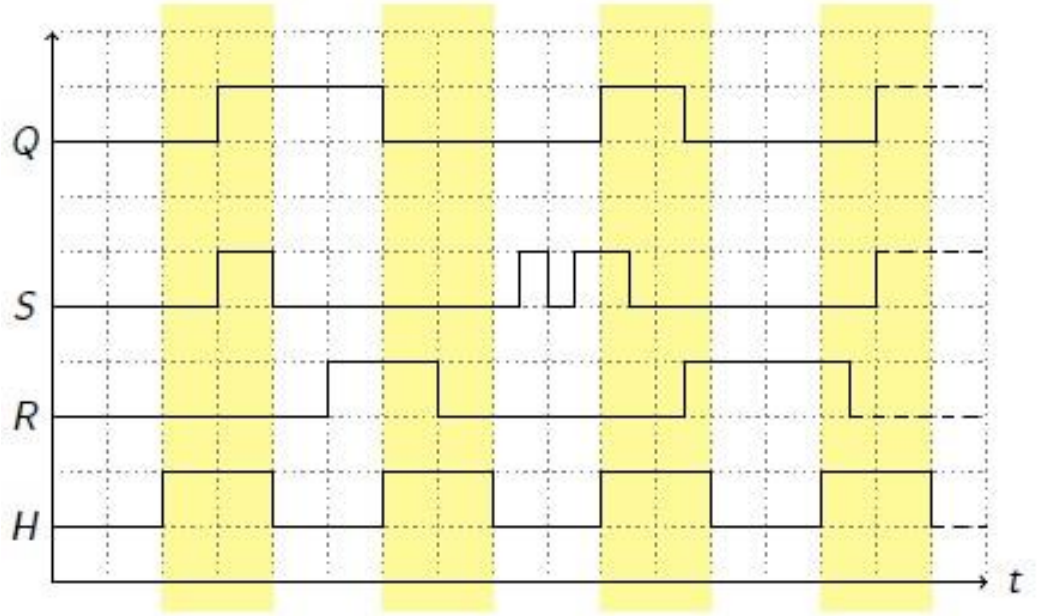
iii ***<u>La Bascule D</u>***

- Pour résoudre l'ambiguïté propre à la bascule RS (quand $S = R = 1$).
- Fait en sorte que l'état correspondant à $S = R = 1$ ne soit jamais en entrée.
- Une seule entrée externe D.



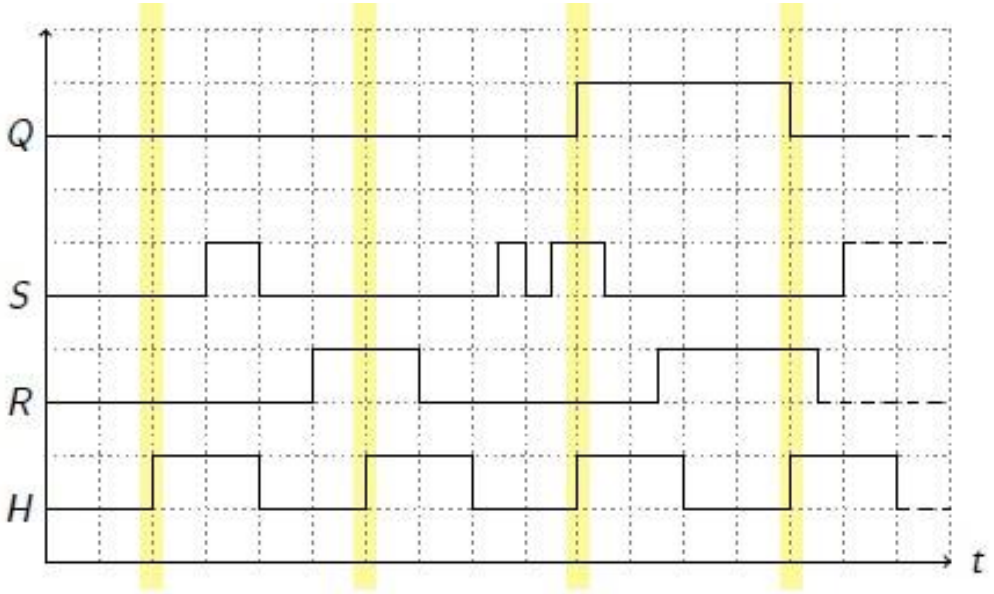
► Bascule latch :

- Bascule asynchrone.
- Change d'état lorsque l'horloge est au niveau 1 (= niveau haut).

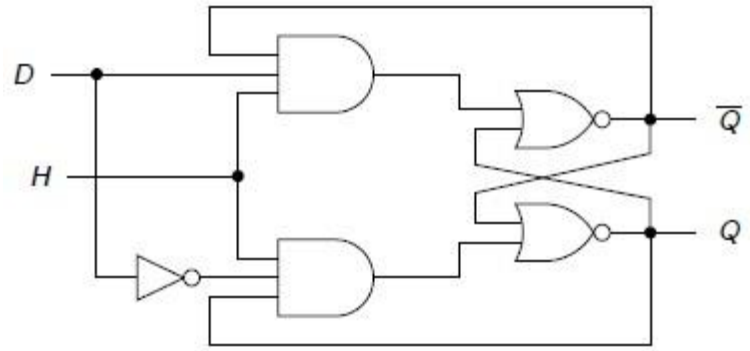


► Bascule flip-flop :

- Bascule synchrone.
- Change d'état lorsque l'horloge est en front montant.

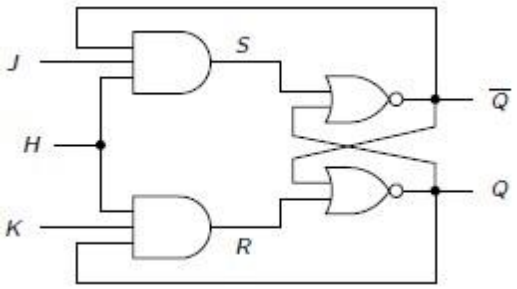


► Bascule D (flip-flop) :



$Q_{n+1} = Dn$

iv ***Bascule JK (flip-flop)***



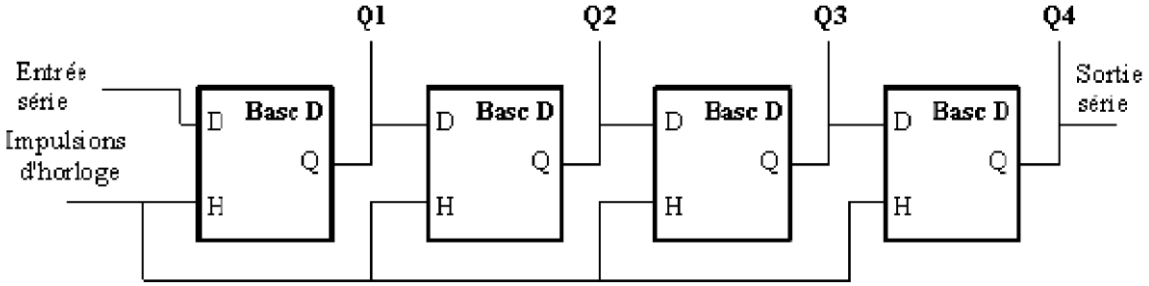
J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q_n

J_n	K_n	Q_n	Q_n	S	R	Q_{n+1}
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	1	0	0	0
0	1	1	0	0	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	1	0	1	1	0	1
1	1	1	0	0	1	0

$$Q_{n+1} = J_n \bar{Q}_n + \bar{K}_n Q_n$$

c. Les registres

- Une bascule est l'élément de base de la logique séquentielle.
- Une bascule permet de mémoriser un seul bit.
- Un registre est un ensemble ordonné de n bascules.
- Un registre permet de mémoriser une information sur n bits.



i ***Types de registres***

- Registres à chargement parallèle ;
- Registres à entrée/sortie série ;
- Registres à entrée série et sortie parallèle ;
- Registres à entrée parallèle et sortie série ;
- Registres à décalage circulaire ii. Entrées asynchrones.

Toutes les bascules précédentes peuvent se voir dotées d'entrées asynchrones permettant de forcer la valeur de la sortie :

- ***Preset*** : pour mettre la sortie Q à 1.
- ***Clear*** : pour mettre la sortie Q à 0.
- ***Enable*** : pour activer/désactiver le signal d'horloge, etc...

Série de TD N°4

(Logique combinatoire et séquentielle)

Exercice 1 : (Fonctions logiques)

Soit un entier de 0 à 7 représenté par 3 bits $b_2b_1b_0$. Soit F la fonction logique dont les entrées sont ces 3 bits et qui prend la valeur 1 si la valeur de l'entrée vaut 0, 3, 4 ou 7 (codée en binaire), et 0 sinon.

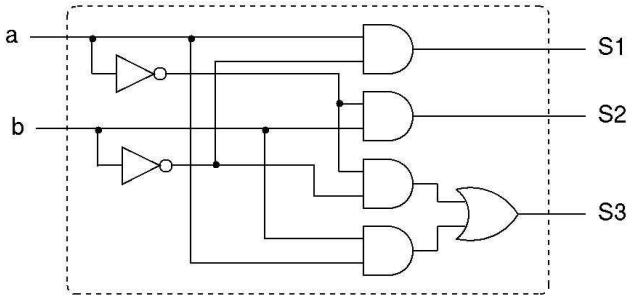
1/ Donner la table de vérité de F.

2/ Déduire la forme canonique de F simplifiée avec la table de Karnaugh.

3/ Dessinez le circuit logique calculant F, uniquement à l'aide de portes NON-ET.

Exercice 2 : (Circuits logiques)

Soit un circuit logique avec le logigramme ci-contre :



1/ Déterminer les expressions logiques des trois sorties du circuit.

2/ Définir la table de vérité du circuit.

3/ Quel est le rôle de ce circuit ?

4/ Réutiliser ce circuit pour construire un circuit au rôle équivalent mais traitant 4 paires (08) d'entrées au lieu de 2.

Exercice 3 : (Additionneur/soustracteur)

On construit un circuit qui a deux entrées (a et b) et deux sorties (s et r) et une ligne de commande F tel que :

- Si F = 0, le circuit effectue une addition (a + b avec une sortie s et une retenue r).
- Si F = 1, le circuit effectue une soustraction (a – b avec une sortie s et une retenue r).

1/ Donnez la table de vérité de ce circuit demi-additionneur/soustracteur. Montrez que s se calcule facilement avec une porte et r avec deux.

2/ On construit maintenant un additionneur/soustracteur complet, c'est-à-dire un circuit à trois entrées (a, b et r), deux sorties (s et r') et une ligne de commande F tel que :

- Si F = 0, le circuit effectue une addition (a + b + r avec une sortie s et une retenue r').
- Si F = 1, le circuit effectue une soustraction (a – (b + r) avec une sortie s et une retenue r').

a) Écrivez la table de vérité de s pour la soustraction et comparez-la à celle de l'additionneur complet.

b) Donnez l'expression logique de s. Écrivez la table de Karnaugh ainsi qu'une expression logique de r'.

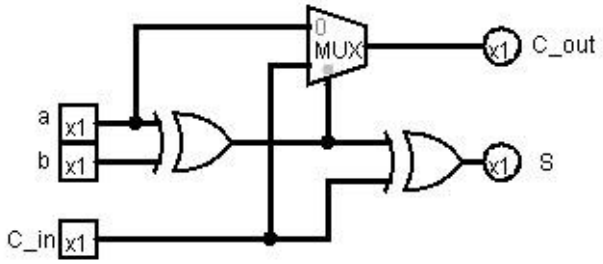
3/ À partir des deux circuits précédents, proposez une construction pour un additionneur/soustracteur sur n bits, c'est-à-dire un circuit avec deux fois n bits en entrée et qui effectue l'addition ou la soustraction de ces deux nombres suivant la valeur d'une ligne de commande.

Exercice 4 : (Multiplexeur/Décodeur)

1/ Prouver algébriquement (par un calcul) que le schéma ci-contre est un additionneur complet.

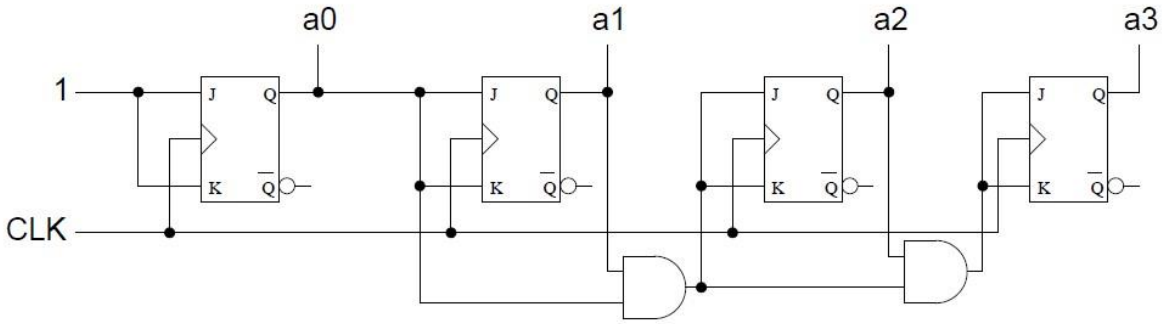
2/ Construire un multiplexeur 8x1 avec 2 multiplexeurs 4x1 et un multiplexeur 2x1.

3/ Construire un décodeur 3x8 avec 2 décodeurs 2x4.



Exercice 5 : (Circuits séquentiels)

Le circuit ci-dessous utilise 4 bascules JK. Le signal CLK correspond à un signal d'horloge et le signal 1 correspond à un signal ayant la valeur de 1 en continu.



1/ Dessiner le chronogramme des 4 sorties a_x du circuit.

2/ En déduire le but de ce circuit.

3/ Modifier ce circuit pour que la valeur maximum calculée ne soit plus 15 mais 9.

Exercice 6 : (Registre mémoire)

Un registre est un élément de stockage qui permet la mémorisation de n bits en parallèle. Cet élément est constitué sur la base d'une mise en parallèle de n bascules mémorisant chacune 1 bit. A l'aide des bascules D, construire un registre qui permet de :

1/ Mémoriser 4 bits en entrée en activant un signal commun d'écriture (*write*).

2/ Vider les bascules via leur entrées asynchrones grâce à un signal commun de remise à zéro (*clear*).

Conclusion

Il est important pour les futurs enseignants spécialistes en informatique de maîtriser les fondamentaux de l'architecture des ordinateurs pour pouvoir enseigner efficacement et améliorer les compétences des générations futures. La compréhension des différents composants d'un ordinateur et leur fonctionnement est primordiale pour optimiser les performances de la machine en fonction de son utilisation.

Au cours de ce document, nous avons examiné les différents chapitres qui traitent l'étude de l'architecture des ordinateurs. Nous avons ainsi parcouru un petit historique sur l'évolution des ordinateurs depuis leurs débuts jusqu'aux modèles actuels. Nous avons également étudié les mémoires, le processeur, la représentation et le codage de l'information ainsi que la logique combinatoire et séquentielle.

Nous espérons que ce document a permis aux étudiants de comprendre les composants de base d'un ordinateur et le schéma fonctionnel qui les relie. Ils seront ainsi en mesure de distinguer les fonctions de chaque composant, de lister tous les composants de base d'un ordinateur et de facilement identifier les causes de problèmes de performance, proposer des solutions matérielles appropriées et ainsi contribuer à l'amélioration de la performance des ordinateurs.

Enfin, il est important de souligner que l'architecture des ordinateurs est une discipline en constante évolution. Les avancées technologiques dans les domaines de la nanotechnologie et de l'informatique quantique ouvrent la voie à de nouvelles perspectives et de nouvelles approches pour l'architecture des ordinateurs. Il est donc essentiel que les enseignants spécialistes en informatique restent à jour et continuent de se former pour transmettre les connaissances les plus récentes aux générations futures.

Liste des Abréviations

AC	Auxiliary Carry (Indécateur de la Retenue auxiliaire)
ACC	Accumulator (Accumulateur)
ADD	Addition (Addition)
AGP	Accelerated Graphics Port (Port Graphique Accéléré)
ALU	Arithmetic Logic Unit (Unité Arithmétique et Logique)
AND	Logical AND (ET Logique)
ASCII	American Standard Code for Information Interchange (Code Standard Américain pour l'Echange d'Informations)
BIT	BInary digiT (Chiffre Binaire)
BITMAP	Binary Image File Format (Format de Fichier d'Image Binaire)
C	Carry (Indécateur de la Retenue)
CD	Compact Disc (Disque Compact)
CDROM	Compact Disc Read-Only Memory (Mémoire en Lecture Seule de Disque Compact)
CEA	Consumer Electronics Association (Association de l'Electronique Grand Public)
CISC	Complex Instruction Set Computing (Calcul avec Ensemble d'Instructions Complexes)
CL	Complément Logique
CLK	Clock (Horloge)
CMOS	Complementary Metal Oxide Semi-conductor
CO	Compteur Ordinal (Ordinal Counter Register)
COBOL	Common Business Oriented Language (Langage de Programmation Orienté Entreprises)
COLOSSUS	Cryptanalytic Computer (Ordinateur Cryptanalytique)
CPU	Central Processing Unit (Unité Centrale de Traitement)
CR	Complément Restreint (Restricted Complement)
CRC	Cyclic Redundancy Check (Vérification de Redondance Cyclique)
CS	Chip Select (Sélection de Puce)
CV	Complément Vrai (True Complement)
D	Bascule bistable D
DEC	Digital Equipment Corporation (Corporation d'Equipements Numériques)
DIV	Division (Division)
DRAM	Dynamic Random Access Memory (Mémoire Vive Dynamique)
DVD	Digital Versatile Disc (Disque Vidéo Numérique)
EDSAC	Electronic Delay Storage Automatic Calculator (Calculateur Automatique à Mémoire de Délai Electronique)
EDVAC	Electronic Discrete Variable Automatic Computer (Ordinateur Automatique à Variables Discretes Electroniques)

EDVAC1	Electronic Discrete Variable Automatic Computer 1 (Ordinateur Automatique à Variables Discretes Electroniques 1)
EN	Enable (Activation)
ENIAC	Electronic Numerical Integrator and Computer (Intégrateur et Ordinateur Numérique Electronique)
ENIGMA	Encryption Machine (Machine de Cryptage)
ENSET	Ecole Normale Supérieure d'Enseignement Technique
ET	Logical AND (ET Logique)
FLOPS	Floating Point Operations Per Second (Opérations à Virgule Flottante par Seconde)
FORTRAN	FORmula TRANslation (Traduction de Formules)
FPGA	Field Programmable Gate Array (Matrice de Portes Programmable sur le Terrain)
FPNI	Field Programmable Nanowire InterConnect (Champ d'Interconnexion Nanowire Programmable)
HP	Hewlett Packard
IBM	International Business Machines
IDE	Integrated Drive Electronics (Electronique de Disque Intégré)
IEEE	Institute of Electrical and Electronics Engineers (Institut des Ingénieurs en Electrique et Electronique)
IEEE754	La norme IEEE-754 décrit les formats à virgule flottante
INV	Inversion
IT	Technologie de l'information
JK	Bascule Flip-flop de Johnson-King
LOAD	Chargement
LSB	Lowest Significant Bit (Bit de poids faible)
MC	Mémoire Central
MHz	Mega Hertz (Fréquence des impulsions)
MIPS	Million Instructions Per Second (Million d'instructions par seconde)
MIT	Massachusetts Institute of Technology (Institut de technologie du Massachusetts)
MPY	Multiplication
MSB	Most Significant Bit (Bit de poids fort)
NAND	Porte logique NON-ET (NOT-AND)
NON	Porte logique NON (NOT)
NOR	Porte logique NON-OU (NOT-OR)
NOT	Porte logique NON (NOT)
OR	Porte logique OU (OR)
OS	Operating System (Système d'exploitation)
OV or V	Overflow (Indécateur de Débordement)
P	Parity (Indécateur de Parité)
PATA	Parallel Advanced Technology Attachment (Advanced Technology Attachment Parallèle)

PC	Personal Computer (Ordinateur personnel)
PCI	Peripheral Component Interconnect (Interconnexion de composants périphériques)
PDP	Programmable Data Processor
PEM	Prof. d'Enseignement Moyen
PIXEL	PICture ELement
PROM	Programmable Read-Only Memory (Mémoire programmable en lecture seule)
R/W	Read or Write Signal (Signale de selection Lecture ou d'Ecriture)
RAM	Random Access Memory (Mémoire vive)
RESET	Réinitialisation
RI	Register Instruction (Registre d'Instruction)
RIM	Registre Information Mémoire
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory (Mémoire morte ()
RS	Bascule bistable de RS
RSH	Bascule bistable de RS avec Horloge H
S	Signe
SATA	Serial Advanced Technology Attachment (Advanced Technology Attachment Série)
SET	Ensembles
SIMPLE	Instruction microprogrammée simple
SP	Stack Pointer (Pile)
SRAM	Static Random Access Memory (Mémoire vive statique)
STORE	Stockage
SUB	Soustraction (Soustration)
TD	Travaux Pratiques
TV	Télévision
TX-0	Ordinateur TX-0 du MIT
UAL	Unité Arithmétique et Logique (Arithmetic and Logic Unit)
UC	Unité centrale (Central Processing Unit)
USB	Universal Serial Bus (Bus série universel)
V.H	Volume Horaire
V.H.H	Volume Horaire Hebdomadaire
VLSI	Very Large Scale Integration (Très grande échelle d'intégration)
XOR	Exclusive OR (OU exclusif)
Z	Zero (Indécateur du Zéro)
ZIP	Zone of Index and Paging (Algorithm de compression Zone d'index et de Pagination)

Références

[1] E. Lazard, *Architecture de l'ordinateur*, Pearson Education. France, 2006.

[2] M. Campbell-Kelly and D. D. Garcia-Swartz, *From Mainframes to Smartphones: A History of the International Computer Industry*, Illustrated edition. Cambridge, Massachusetts: Harvard University Press, 2015.

[3] J. Greenberg, "The Enigma machine," in *The Turing Guide*, J. Copeland, J. Bowen, M. Sprevak, and R. Wilson, Eds., Oxford University Press, 2017, p. 0. doi: 10.1093/oso/9780198747826.003.0018.

[4] B. J. Copeland, *Colossus: The secrets of Bletchley Park's code-breaking computers*, Reprint edition. Oxford: Oxford University Press, 2010.

[5] "Archives:The Computer Pioneers: The TX-0," *ETHW*, May 11, 2015. https://ethw.org/Archives:The_Computer_Pioneers:_The_TX-0 (accessed Mar. 31, 2023).

[6] R. P. Feynman, "There's Plenty of Room at the Bottom," *Engineering and Science*, vol. 23, no. 5, Art. no. 5, Feb. 1960.

[7] S. H. Jo, K.-H. Kim, and W. Lu, "High-Density Crossbar Arrays Based on a Si Memristive System," *Nano Lett.*, vol. 9, no. 2, pp. 870–874, Feb. 2009, doi: 10.1021/nl8037689.

[8] G. S. Snider and R. S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," *Nanotechnology*, vol. 18, no. 3, p. 035204, Jan. 2007, doi: 10.1088/0957-4484/18/3/035204.

[9] CEA, "CEA-Leti to Build Quantum-Photonics Platform to Ensure Ultra-Secure Data for Finance, Energy, Defense and Other Industries," *CEA/Leti (english)*, Oct. 29, 2020. https://www.cea.fr/cea-tech/leti/english/Pages/What's-On/Press release/CEA-Leti-to-Build-Quantum-Photonics-Platform-to-Ensure-Ultra-Secure-Data-for-Finance,-Energy,-Defense-and-Other-Industries.aspx (accessed Feb. 25, 2023).

[10] J. von Neumann, "First draft of a report on the EDVAC," *IEEE Annals of the History of Computing*, vol. 15, no. 4, pp. 27–75, 1993, doi: 10.1109/85.238389.

[11] J. Delacroix and A. Cazes, *Architecture des machines et des systèmes informatiques*, Dunod. in InfoSup. 2018. Accessed: Jan. 14, 2023. [Online]. Available: https://www.dunod.com/sciences-techniques/architecture-machines-et-systemes-informatiques-0

[12] V. Doutaut, *Informatique et culture scientifique du numérique — Mooc Inria ICN–SNT–Python*. Learning Lab Inria, 2019. [Online]. Available: https://hal.inria.fr/hal-03346079

[13] J.-M. Richer, "Page de Jean-Michel Richer," *Page de Jean-Michel Richer*. https://leria-info.univ-angers.fr/~jeanmichel.richer/ensl1i.php (accessed Jan. 14, 2023).

[14] T. Dumartin, "Architecture des ordinateurs Note de cours." Accessed: Jan. 14, 2023. [Online]. Available: https://www.academia.edu/34101357/Architecture_des_ordinateurs_Note_de_cours

[15] G. Hyatt, "Gilbert Hyatt Files the First General Patent on the Microprocessor; It is Later Invalidated : History of Information." https://www.historyofinformation.com/detail.php?id=95 (accessed Jan. 14, 2023).

[16] S. Cass, "Chip Hall of Fame: Intel 4004 Microprocessor - IEEE Spectrum," Jul. 02, 2018. https://spectrum.ieee.org/chip-hall-of-fame-intel-4004-microprocessor (accessed Feb. 24, 2023).

[17] J. Jorda and A. M'zoughi, *Mini manuel d'architecture de l'ordinateur*. 2023. Accessed: Jan. 14, 2023. [Online]. Available: https://www.dunod.com/sciences-techniques/mini-manuel-d-architecture-ordinateur

[18] *Les microprocesseurs...comment ça marche ? - Greg Wyant , Tucker... - Librairie Eyrolles*. Accessed: Jan. 14, 2023. [Online]. Available: https://www.eyrolles.com/Informatique/Livre/les-microprocesseurs-comment-ca-marche--9782100023509/

[19] J. Needham, *Science and Civilisation in China, Volume 3: Mathematics and the Sciences of the Heavens and the Earth*. Cambridge: Cambridge University Press, 1959.

[20] J. Legge, *The I-Ching Or The Book Of Changes: The Yi King*. CreateSpace Independent Publishing Platform, 2008.

[21] R. Tieszen, "Leibniz, Husserl and Gödelian Monadology," in *Kurt Gödel Philosopher-Scientist*, G. Crocco and E.-M. Engelen, Eds., in Épistémè. Aix-en-Provence: Presses universitaires de Provence, 2021, pp. 447–463. Accessed: Feb. 25, 2023. [Online]. Available: http://books.openedition.org/pup/53705

[22] M. B. W. Tent, Ed., *Gottfried Wilhelm Leibniz: The Polymath Who Brought Us Calculus*, 1st edition. Boca Raton: A K Peters/CRC Press, 2011.

[23] C. A. Baylis, “Claude E. Shannon. A symbolic analysis of relay and switching circuits. Transactions of the American Institute of Electrical Engineers, vol. 57 (1938), pp. 713–723.,” *The Journal of Symbolic Logic*, vol. 4, no. 2, pp. 103–103, Jun. 1939, doi: 10.2307/2269093.
[24] Z. Taha, *Structure de la Machine cours et exercices*. Université de Bouira, 2021. [Online]. Available: https://archive.org/details/strm-1-book-taha-zerrouki-09-10-2021
[25] Z. Taha, *Structure de la Machine II*. Université de Bouira, 2022. [Online]. Available: https://infobouirauniv.wordpress.com/2022/05/09/strm2-book/