

الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي والبحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE TLEMCCEN
FACULTE DE TECHNOLOGIE
DEPARTEMENT DE GENIE ELECTRIQUE
ET ELECTRONIQUE



جامعة أبي بكر بلقايد - تلمسان -
كلية التكنولوجيا
قسم الهندسة الكهربائية والإلكترونية

ARCHITECTURE DES ORDINATEURS

Note de Cours et Exercices

Dr. BETAOUAF Talib Hicham



2EME ANNEE LICENCE EN PRODUCTIQUE
FILIERE GÉNIE INDUSTRIEL
Université de Tlemcen
Année Universitaire 2020/2021

Architecture des Ordinateurs

Note de Cours et Exercices

2^{ème} Année Licence en Productique

Filière Génie Industriel

Dr. BETAOUAF Talib Hicham

Avant-propos

Ce document est inspiré d'un enseignement délivré depuis plusieurs années aux étudiants en 2^{ème} année de Licence en Productique. Il regroupe toutes les notions essentielles à l'architecture des ordinateurs sous forme de collections de notes et de mots clés faciles à retenir pour l'étudiant. Ces notes de cours résument très sommairement les points importants abordés en cours.

L'objectif est de faire comprendre les mécanismes internes de l'ordinateur et leurs implications sur l'environnement informatique aux étudiants de licence, et de façon plus générale de faire découvrir l'architecture des machines à tous ceux qui s'intéressent à ce sujet.

Ce recueil a pour ambition d'aider l'étudiant à mieux assimiler les notions vues en cours et en TD grâce aux photographies et schémas illustratifs. Il est organisé en quatre chapitres qui couvrent la totalité du programme de la matière.

A la fin de chaque chapitre du polycopié, des séries d'exercices sont proposées pour des travaux dirigés. Ces exercices peuvent aussi servir comme soutien aux étudiants pour la révision lors des contrôles d'évaluation.

L'auteur remercie chaleureusement ses collègues, Dr MENADJLIA Nardjes, maître de conférences à l'université de Tlemcen et Mr BELKHERROUBI Mustafa Kamel maître assistant à l'université de Tlemcen, d'avoir expertisé ce travail et leur exprime son immense gratitude pour toutes leurs remarques et suggestions pertinentes dans le but d'améliorer la qualité de ce document.

Table des matières

AVANT-PROPOS	2
INTRODUCTION	5
CHAPITRE 1 : GENERALITES SUR L'ORDINATEUR	6
1. HISTOIRE DES ORDINATEURS	6
A. LA GENERATION ZERO : LES CALCULATEURS MECANIQUES (AVANT 1945)	6
B. LA PREMIERE GENERATION : LES TUBES A VIDE (1945 - 1955)	7
C. LA DEUXIEME GENERATION : LES TRANSISTORS (1955 - 1965).....	8
D. LA TROISIEME GENERATION : LES CIRCUITS INTEGRES (1965 - 1973).....	8
E. LA QUATRIEME GENERATION : LES MICROPROCESSEURS (1971 - 1980)	8
F. AUJOURD'HUI : LA LOI DE MOORE EST ENCORE D'ACTUALITE	9
2. STRUCTURE DE BASE D'UN ORDINATEUR	9
A. DEFINITIONS.....	9
B. ARCHITECTURE DE JOHN VON NEUMANN	10
C. ARCHITECTURE EN COUCHES	10
D. L'ORDINATEUR DU POINT DE VUE INTERNE	11
3. REPRESENTATION ET CODAGE DE L'INFORMATION	18
A. CODAGE DE L'INFORMATION	18
B. SYSTEME DE NUMERATION.....	18
C. CODAGE DES NOMBRES	21
D. CODAGE DES CARACTERES	26
E. CODAGE D'UNE IMAGE	27
F. CODAGE DU SON	28
EXERCICES	29
CHAPITRE 2 : LOGIQUE COMBINATOIRE ET SEQUENTIELLE	30
1. CIRCUITS LOGIQUES	30
A. FONCTIONS LOGIQUES ELEMENTAIRES	30
B. REGLES DE CALCUL	32
C. CONSTRUCTION ET OPTIMISATION	32
2. CIRCUITS COMBINATOIRES	36
A. DEFINITIONS.....	36
B. CARACTERISTIQUES	36
C. CIRCUITS COMBINATOIRES DE BASE	37
3. CIRCUITS ARITHMETIQUES	40
A. DEMI-ADDITIONNEUR.....	40
B. ADDITIONNEUR COMPLET	41
C. UNITE ARITHMETIQUE ET LOGIQUE.....	41
4. CIRCUITS SEQUENTIELS	43
A. DEFINITIONS.....	43

B. LES BASCULES	44
C. LES REGISTRES	49
EXERCICES	50
CHAPITRE 3 : MEMOIRES.....	52
1. DEFINITIONS	52
2. CARACTERISTIQUES	52
A. CAPACITE D'UNE MEMOIRE.....	52
B. VOLATILITE	53
C. MODE D'ACCES A L'INFORMATION	53
D. TEMPS D'ACCES.....	53
3. TYPES DE MEMOIRES	54
A. LES REGISTRES	54
B. LA MEMOIRE CACHE.....	54
C. LA MEMOIRE PRINCIPALE	54
D. LA MEMOIRE D'APPUI.....	54
E. LA MEMOIRE DE MASSE	54
4. CLASSIFICATION DES MEMOIRES	55
5. MEMOIRE CENTRALE	55
A. VUE LOGIQUE DE LA MEMOIRE CENTRALE.....	56
B. STRUCTURE PHYSIQUE D'UNE MEMOIRE CENTRALE	56
EXERCICES	58
CHAPITRE 4 : PROCESSEURS	59
1. MICROPROCESSEUR	59
2. ARCHITECTURE INTERNE.....	60
A. L'UNITE DE COMMANDE.....	60
B. L'UNITE DE TRAITEMENT	61
C. LES BUS DE COMMUNICATION.....	61
3. CYCLE D'UNE INSTRUCTION	63
4. JEU D'INSTRUCTIONS.....	64
A. DEFINITION.....	64
B. TYPE D'INSTRUCTIONS	64
C. CODAGE.....	65
D. MODE D'ADRESSAGE.....	65
E. NOTION D'ARCHITECTURE RISC ET CISC	66
EXERCICES	68
REFERENCES.....	69

Introduction

L'informatique, qui est l'une des principales disciplines intégrées à la productique, est une science centrée sur l'ordinateur. Tout spécialiste en productique doit maîtriser le fonctionnement de bas niveau d'un ordinateur pour la mise en œuvre et le perfectionnement des systèmes de production automatisés.

L'architecture des ordinateurs est une science 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 son architecture permet de savoir dans quelle mesure les caractéristiques spécifiques de chaque composant influencent la réactivité de la machine en fonction de son utilisation : pourquoi l'ajout de mémoire accélère-t-il 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 vont-ils toujours plus vite ?

L'objectif principal de ce cours est de connaître les composants de base d'un ordinateur et de comprendre le schéma fonctionnel qui les relie. A la fin de ce cours, l'étudiant devra être capable de lister tous les composants de base d'un ordinateur. Il sera également apte à distinguer entre les fonctions de chaque composant. Aussi, il pourra facilement identifier la cause d'un problème de performance et proposer une solution matérielle appropriée.

Ce document est organisé comme suit :

Le chapitre 1 regroupe des notions générales sur la structure d'un ordinateur en présentant un petit historique sur l'évolution des ordinateurs depuis leurs apparition jusqu'aux modèles actuels. Ce chapitre aborde également la représentation de l'information dans un ordinateur et plus particulièrement le codage des nombres et l'arithmétique binaire.

Le chapitre 2 est un rappel des principes fondamentaux de la logique combinatoire et séquentielle, sur laquelle sont basés la plupart des composants de l'ordinateur.

Dans le chapitre 3, une brève description des mémoires est donnée pour permettre de bien comprendre la structure et le fonctionnement de la mémoire centrale d'un ordinateur.

Le quatrième et dernier chapitre 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.

Chapitre 1 : Généralités sur l'ordinateur

1. Histoire des ordinateurs

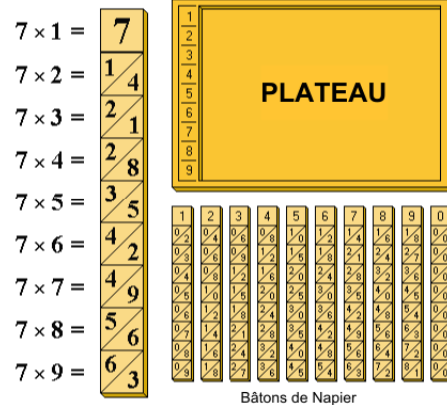
L'ordinateur est né du travail de plusieurs ingénieurs et théoriciens pendant la Seconde Guerre mondiale. Il n'y avait pas un pôle de développement, mais plusieurs centres indépendants, chacun essayant de construire des machines qui n'existaient pas à l'époque. [1]

L'évolution de l'ordinateur actuel est passée par plusieurs étapes fortement liées aux inventions technologiques qui se distinguent en plusieurs générations. [2]

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

► Les abaques (avant 1600) :

- Instruments mécaniques facilitant le calcul



► 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 !



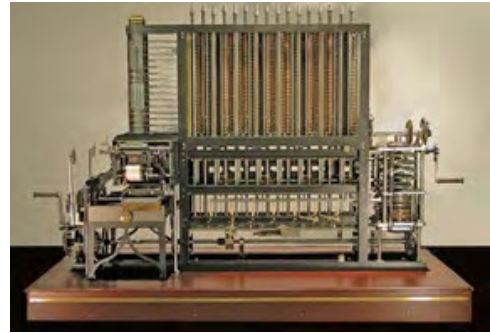
► Le métier Jacquard (1805)

- 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 : 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, volée aux allemands.
- Besoin de beaucoup de calculs, rapidement pour les décrypter → Création du premier ordinateur électronique : le **COLOSSUS**.



- 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) :

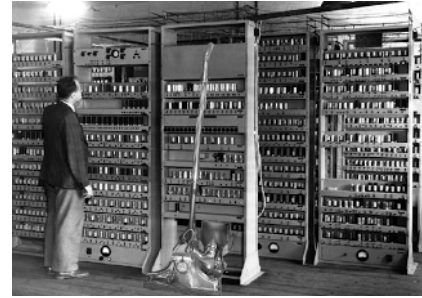
- Electronic Numerical Integrator And Computer.
- 1^{er} ordinateur électronique Turing-complet ¹.
- Calculs en système décimal.
- Composé de 18 000 tubes à vide et 1500 relais.
- 6000 commutateurs et une forêt de câbles.
- Pèse 30 tonnes, et occupe 167 m².
- Incapable d'enregistrer un programme.



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 (Electronic Discrete Variable Automatic Computer).
- Utilise un système binaire.
- Les opérations d'addition +, soustraction - et multiplication \times étaient automatiques.
- La division \div était programmable.
- Capacité mémoire initiale : 1000 mots de 44 bits.



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

- Le transistor a été inventé en 1948, aux Bell Labs (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.
- Les ordinateurs deviennent assez stables pour être vendus à des clients : naissance de l'industrie de la mini-informatique (IBM, DEC, HP, ...).
- Premier jeu vidéo avec le PDP-1 : spacewar !
- Apparition des OS (Operating System) et langages évolués (FORTRAN et COBOL).



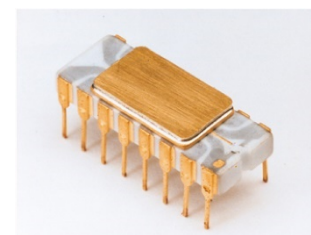
d. La troisième génération : les circuits intégrés (1965 - 1973)

- Le circuit intégré a été inventé 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).
- Possibilité d'émulation d'anciens modèles.



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

- Miniaturisation des circuits : l'ère de la micro-informatique
- Evolution de l'intégration avec la technologie VLSI (Very Large Scale Integration).
- Premier micro-processeur inventé par INTEL en 1971.



- Apparition des ordinateurs personnels (PC) utilisés pour :
 - Traitement de texte
 - Tableur
- Apparition d'Apple (1976)



f. Aujourd'hui : la loi de Moore² est encore d'actualité

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

- Ordinateur jetable : Cartes de vœux
- Ordinateur enfoui : Montres, voitures
- Ordinateur de jeux : Jeux vidéo
- Ordinateur personnel (micro-ordinateur) : Ordinateurs portables ou de bureau
- Serveur : Serveurs de réseau
- Ensemble de stations de travail : Mini-superordinateur
- Mainframe : Traitement par lot dans une banque
- Superordinateur : Prévisions météo à long terme

2. Structure de base d'un ordinateur

a. Définitions

Ordinateur : Machine capable de résoudre des problèmes en appliquant des instructions.

Instruction : Action à effectuer par l'ordinateur, correspondant à une étape dans un programme.

Programme : Suite d'instructions décrivant la façon dont l'ordinateur doit effectuer un travail.

Langage machine : Ensemble des instructions exécutables directement par un ordinateur.

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.

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 la première description d'un ordinateur dont le programme est stocké dans sa mémoire. [3]



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

Cette architecture est basée sur 4 parties principales :

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

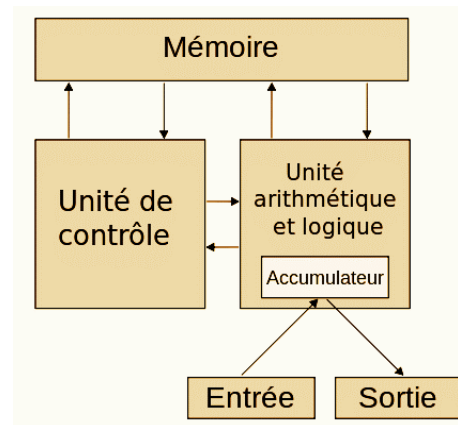


Figure 1 : Schématisation de l'architecture de von Neumann. [4]

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.

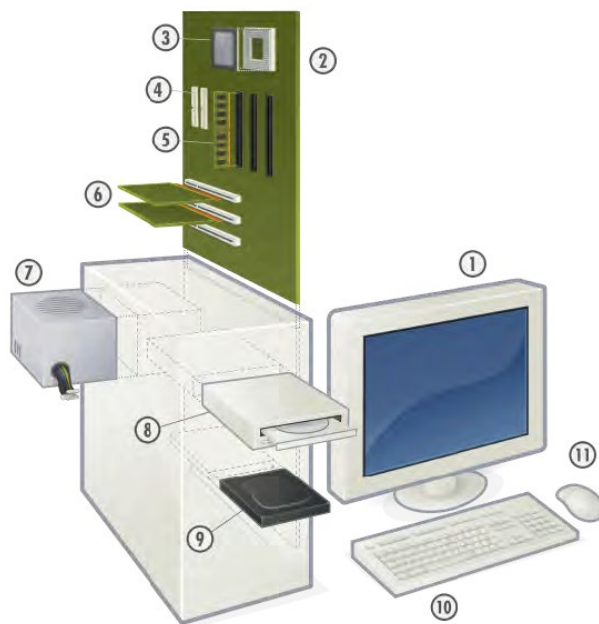
Niveau 2 : Couche architecture du jeu d'instructions (propre à chaque machine).

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

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.



1. Ecran (Moniteur)
2. Carte mère
3. CPU (Micro-processeur)
4. Bus
5. Mémoire vive (RAM)
6. Cartes de périphériques
7. Alimentation
8. Lecteur de disques
9. Disque Dur
10. Clavier
11. Souris

Figure 2 : L'ordinateur du point de vue interne [2]



Kézako ???

Annonce trouvée sur un site de vente en ligne :



Asus ET2210IUTS-B002E ¹, Microsoft Windows 7 Pro 64 bits ², Intel Core i3-2120 ³, Multi-points Full HD ⁴, 21.5" ⁵, 1920 x 1080 ⁶, Intel HD Graphics ⁷, 1To ⁸, 4096Mo ⁹, DVD+/-RW Super Multi ¹⁰, 1,3 M Pixel + Micro ¹¹, 2x 2W + Sonic Master DTS Surround Sensation UltraPC ¹², [1x Entrée HDMI, 1x Sortie HDMI, LAN port (RJ 45)] ¹³, 802.11 b/g/n ¹⁴, [2x ports USB 3.0, 3x ports USB 2.0] ¹⁵, 590 x 461 x 60-230 mm ¹⁶, 10.8kg ¹⁷, Titane noir ¹⁸

- | | | |
|-----------------------------|------------------------|--------------------------|
| 1. Référence modèle | 7. Carte graphique | 13. Connectiques |
| 2. Système d'exploitation | 8. Disque dur | 14. Wi-Fi |
| 3. Processeur | 9. RAM | 15. Ports USB |
| 4. Affichage | 10. Lecteur de disques | 16. Taille de l'ensemble |
| 5. Taille écran (en pouces) | 11. Webcam | 17. Poids de l'ensemble |
| 6. Résolution | 12. Enceintes | 18. Couleur |

i. Carte mère

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

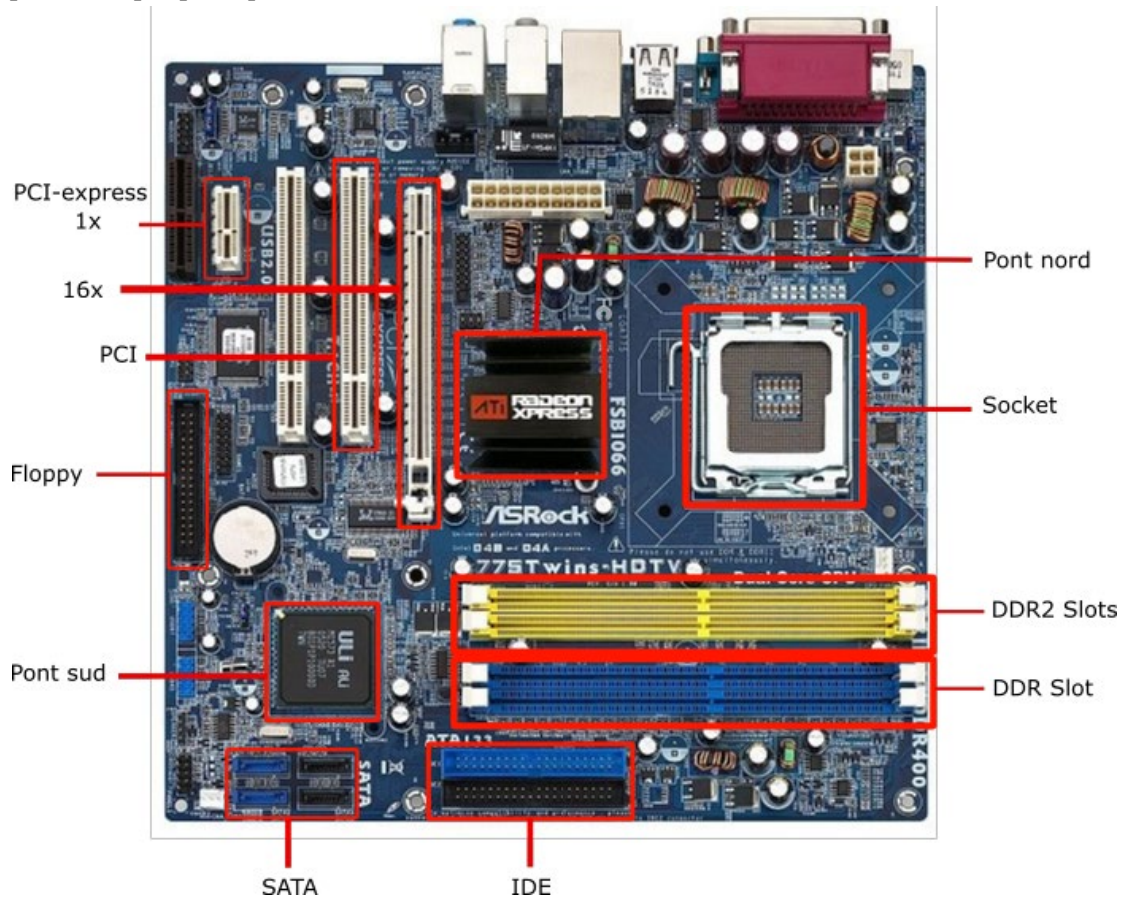


Figure 3 : Composants de la carte mère [5]

ii. 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



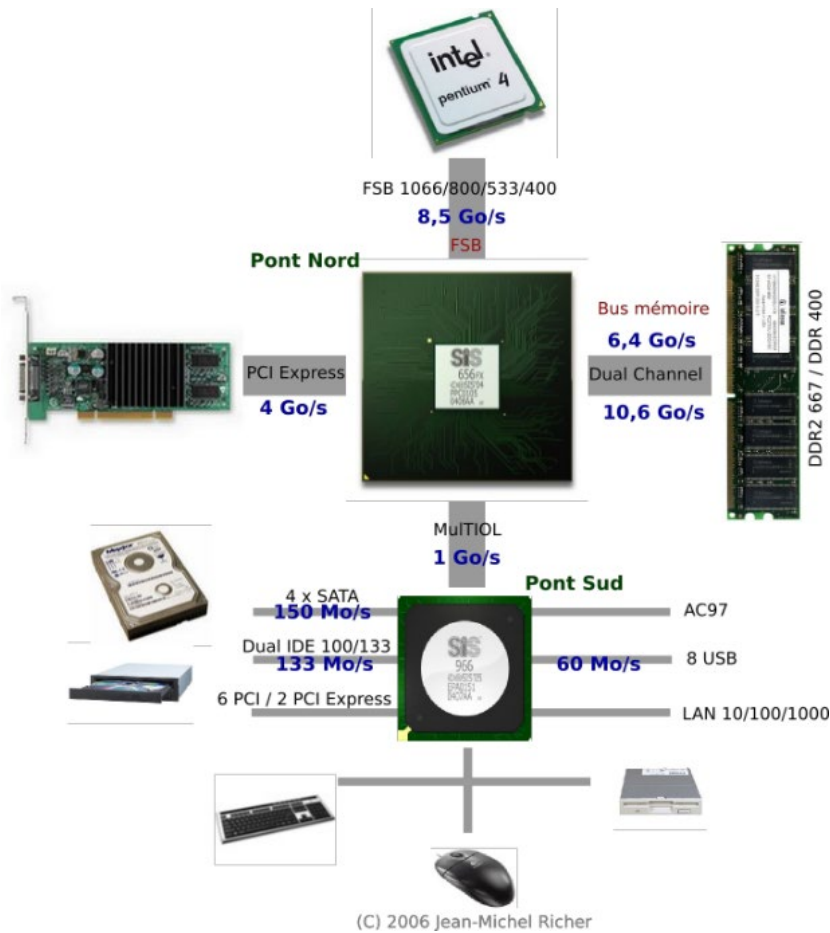
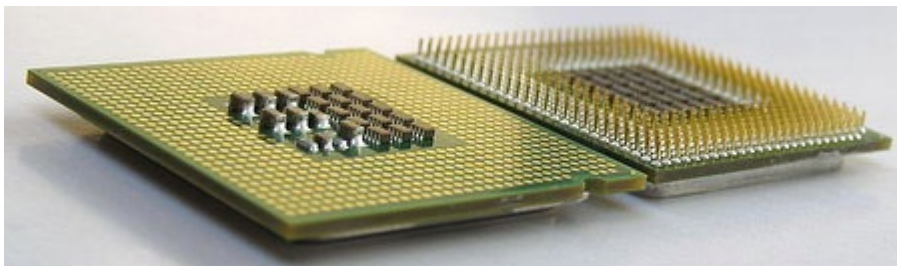


Figure 4 : Vue schématique d'une carte mère de type Intel avec chipset Sis 656fx/966, (2006) [5]

iii. Le Processeur

- En anglais CPU (Central Processing Unit) ou UC (Unité Centrale) en français.
- C'est le « Cerveau » de l'ordinateur.
- Exécute les programmes stockés en mémoire principale.
- Il est responsable du :
 - Chargement des instructions
 - Décodage des instructions
 - Exécution des instructions, l'une après l'autre



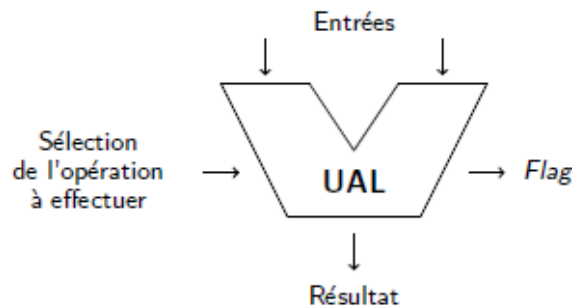
Il est composé de :

- **Unité Arithmétique et Logique (UAL) :**
 - En anglais ALU (Arithmetic and Logical Unit).
 - 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, . . .
 - **Opérations logiques :** compléments, et, ou, ou-exclusif, non, non-et, . . .
 - **Comparaisons :** test d'égalité, supérieur, inférieur, . . .
 - **Décalages :** des bits à gauche, à droite...



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) :** (Stack Pointer) contient l'adresse du sommet de la pile ³.
- **Registres généraux (R₀, ..., 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.

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).

iv. La Mémoire Principale

- Mémoire « de travail » de l'ordinateur.
- Mémoire vive : RAM (Random Access Memory)
- Caractéristiques :
 - Rapide d'accès.
 - Volatile.
- Le processeur y accède pour lire/écrire des données.

v. Les Entrées/Sorties

- En anglais « Input/Output » (I/O)
- Permettent les échanges d'information entre le processeur et les périphériques associés.
- **Entrées** : données envoyées par un périphérique à destination du processeur.
- **Sorties** : données émises par le processeur à destination des périphériques.

1. Les périphériques d'entrée :

Permettent à l'utilisateur de fournir une information à l'ordinateur.

- ▶ Exemples : clavier, scanner, . . .

2. Les périphériques de sortie :

Permettent à l'ordinateur de fournir une information à l'utilisateur

- ▶ Exemples : écran, enceintes, . . .

3. Les périphériques d'entrée-sortie :

Permettent à l'utilisateur/l'ordinateur de fournir/recevoir une information

- ▶ Exemples : clé USB, . . .

- ▶ Quelques exemples de périphériques :

<i>Périphérique</i>	<i>Entrée</i>	<i>Sortie</i>	<i>Entrée/Sortie</i>
<i>Clavier</i>	●		
<i>Souris</i>	●		
<i>Ecran</i>		●	
<i>Ecran tactile</i>			●
<i>Lecteur CD/DVD</i>	●		
<i>Graveur CD/DVD</i>			●
<i>Webcam</i>	●		
<i>Imprimante</i>		●	
<i>Carte réseau</i>			●
<i>Microphone</i>	●		
<i>Enceinte</i>		●	
<i>Scanner</i>	●		
<i>Disque dur</i>			●
<i>Clé USB</i>			●

vi. 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

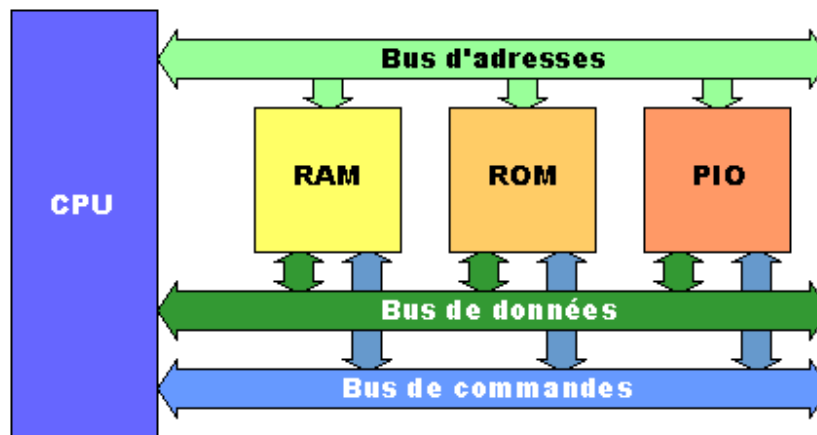
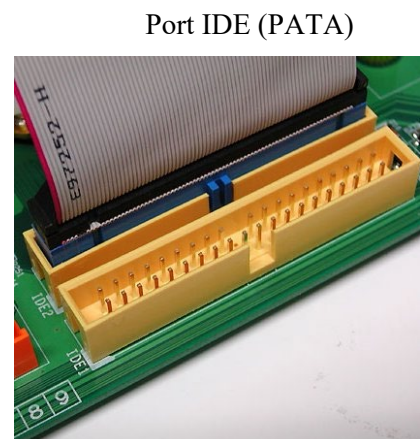
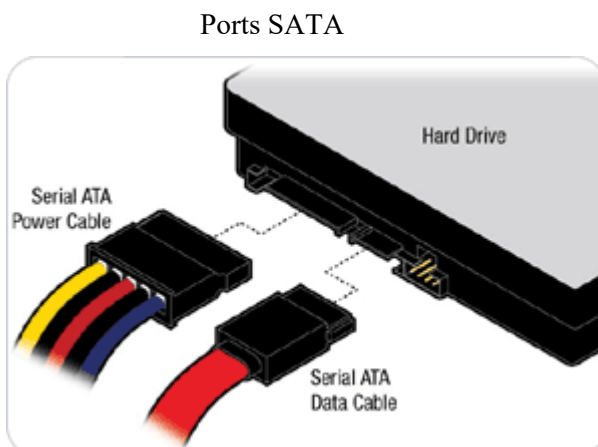


Figure 5 : Schéma des bus de communication [5]

► La connectique :

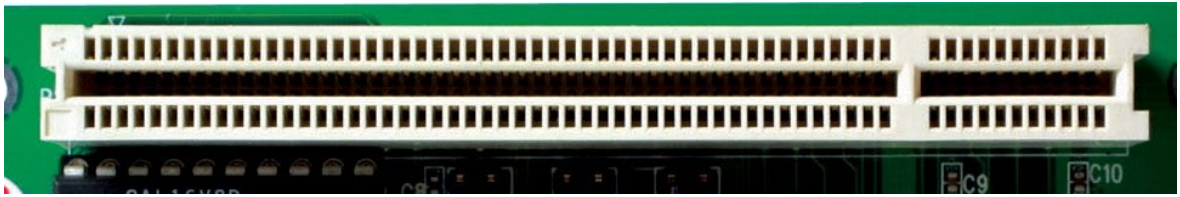
Les composants de l'unité centrales sont connectés à la carte mère via des ports spécifiques.

- Ports pour disques dur et lecteurs CD/DVD :



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

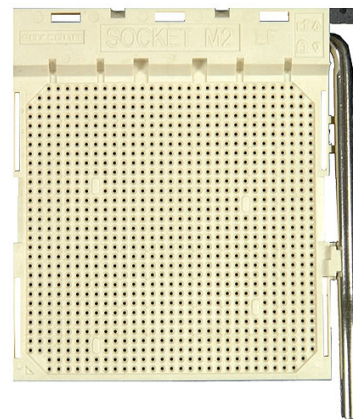
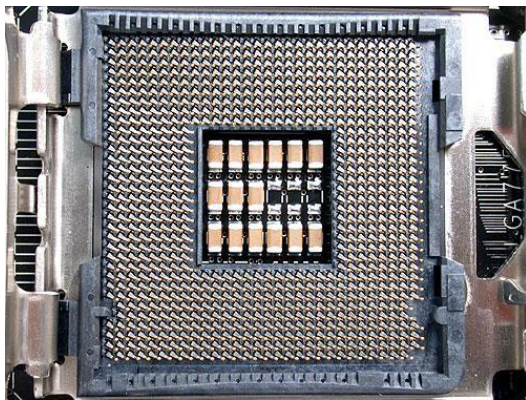
Port PCI



Port AGP



- Socket : pour brancher le microprocesseur.



- Ports USB: **Universal Serial Bus**



3. Représentation et 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.

a. Codage de l'information

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.

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 (CDROM) : réflexion (bit 1) ou pas de réflexion (bit 0)
- Fréquences (Modem) : dans un signal sinusoïdal.


b. Système de numération

Système de numération décrit la façon avec laquelle les nombres sont représentés.

Un système de numération est défini par :

- Un alphabet \mathcal{A} : ensemble de symboles ou chiffres,
- Des règles d'écritures des nombres : juxtaposition de symboles.

i. Exemples de Système de numération

- Numération Romaine : I, II, III, IV, ..., X, ..., CCLXXI -> 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.

ii. Système de numération positionnel pondéré à base b

Un système de numération positionnel pondéré à base b est défini sur un alphabet de b chiffres:

$$\mathcal{A} = \{c_0, c_1, \dots, c_{b-1}\} \text{ avec } 0 \leq c_i < b$$

- Soit $N = a_{n-1} a_{n-2} \dots a_1 a_0$ (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 :
$$N = a_{n-1} \cdot b^{n-1} + a_{n-2} \cdot b^{n-2} + \dots + a_0 \cdot b^0 \quad (10) = \sum_{i=0}^{n-1} a_i \cdot b^i$$

iii. 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.

iv. 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.

Par la suite, on verra les conversions suivantes :

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

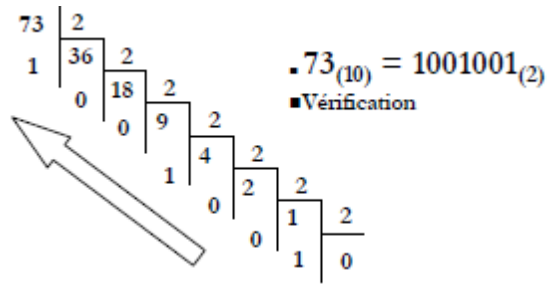
1. De la base 10 vers une base b

La règle à suivre est les divisions successives :

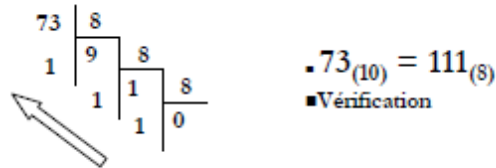
- On divise le nombre par la base b.
- Puis le quotient par la base b.
- Ainsi de suite jusqu'à l'obtention d'un quotient nul.
- La suite des restes correspond aux symboles de la base visée.
- On obtient en premier le chiffre de poids faible et en dernier le chiffre de poids fort.

- **Exemple** : Soit N le nombre d'étudiants d'une classe représenté en base décimale par : $N = 73_{(10)}$

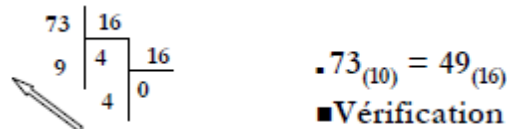
Représentation en Binaire ?



Représentation en Octal ?



Représentation en Hexadécimal ?



2. De la base 2 vers une base b

- **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 décimal : par définition ($\sum_{i=0}^{n-1} a_i \cdot b^i$)

- **Exemple** : Soit $N = 1010011101_{(2)} = ?_{(10)}$

$$N = 1 \times 2^9 + 0 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 512 + 0 + 128 + 0 + 0 + 16 + 8 + 4 + 0 + 1$$

$$= 669_{(10)}$$

$$\Rightarrow 1010011101_{(2)} = 669_{(10)}$$

- Binaire vers octal : regroupement des bits en des sous-ensembles de trois bits puis remplacer chaque groupe par le symbole correspondant dans la base 8. (Voir tableau ci-contre)

- **Exemple** : Soit $N = 1010011101_{(2)} = ?_{(8)}$

$$N = 001\ 010\ 011\ 101_{(2)}$$

$$= 1\ 2\ 3\ 5_{(8)}$$

$$\Rightarrow 1010011101_{(2)} = 1235_{(8)}$$

Symbole Octale	Suite binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

- Binaire vers Hexadécimal : regroupement des bits en des sous-ensembles de quatre bits puis remplacer chaque groupe par le symbole correspondant dans la base 16. (Voir tableau ci-contre).

Symbole Hexadécimal	Suite 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

► **Exemple** : Soit $N = 1010011101_{(2)} = ?_{(16)}$

$$N = 0010\ 1001\ 1101_{(2)}$$

$$= 2\ 9\ D_{(16)}$$

$$\Rightarrow 1010011101_{(2)} = 29D_{(16)}$$

c. Codage des nombres

i. 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 : $00001010_{(2)}$

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 et de multiplication :

Addition

0	0	0
0	1	1
1	0	1
1	1	⁽¹⁾ 0

Retenue

► Exemple (Addition)

Multiplication

0	0	0
0	1	0
1	0	0
1	1	1

Addition binaire (8 bits)

$$\begin{array}{r} 10010110 \\ + 01010101 \\ \hline 11101011 \end{array}$$

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

$$\begin{array}{r} 10010110 \\ + 01110101 \\ \hline 100001011 \\ \Rightarrow \text{Overflow} \end{array}$$

► Exemple (Multiplication)

Multiplication binaire

$$\begin{array}{r}
 1011 \text{ (4 bits)} \\
 \times 1010 \text{ (4 bits)} \\
 \hline
 0000 \\
 1011 \\
 0000 \\
 1011 \\
 \hline
 01101110
 \end{array}$$

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

ii. Codage des entiers relatifs

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 signé

➤ 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 (bs)
 - 24 est codé en binaire signé par : $10011000_{(bs)}$
 - 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 et +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 $\Rightarrow 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0_{(2)}$, puis on inverse les bits $\Rightarrow 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1_{(cà1)}$

- 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 a 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 $0\ 0\ 0\ 1\ 1\ 0\ 0\ 0_{(2)}$
 $-24 \Rightarrow 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1_{(cà1)}$, 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 $11111111_{(cà1)} \Rightarrow$ 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 = 11111111
 - +127 = 01111111 -128 = 10000000

iii. 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)
 - Défini par : $\pm m \cdot b^e$
 - un signe + ou -
 - une mantisse **m** (en virgule fixe)
 - un exposant **e** (un entier relative)
 - une base **b** (2,8,10,16,...)
 - ▶ Exemple : $0,5425 \cdot 10^2_{(10)}$; $10,1 \cdot 2^{-1}_{(2)}$; $A0,B4 \cdot 16^{-2}_{(16)}$

1. Codage en Virgule Fixe

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

- $X = a_{n-1} a_{n-2} \dots a_1 a_0 , a_{-1} a_{-2} \dots a_{-p} (b)$
- a_{n-1} est le chiffre de poids fort (MSB⁴).
- a_{-p} est le chiffre de poids faible (LSB⁵).
- 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 : $X = \sum_{i=0}^{n-1} a_i \cdot b^i$

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

✓ Changement de base $10 \rightarrow 2$

- 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

$$4_{(10)} = 100_{(2)}$$

$$0,25 \times 2 = 0,5 \rightarrow 0$$

$$0,5 \times 2 = 1,0 \rightarrow 1$$

$$\Rightarrow \text{donc } 4,25_{(10)} = 100,01_{(2)}$$

2. Codage en Virgule Flottante

Un nombre réel est représenté en virgule flottante sous la forme : $\pm M \times 2^E$, 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

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

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

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

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

\Rightarrow **Normalisation** : afin d'avoir la même représentation, tout nombre doit d'abord être normalisé sous la forme : $\pm 1, M \times 2^{Eb}$

- Le signe est codé sur 1 bit ayant le poids fort (**S**):
 - Le signe $-$: bit 1
 - Le signe $+$: bit 0

⁴ Most significant bit

⁵ Less significant bit

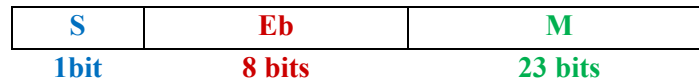
- Exposant biaisé (**Eb**) :
 - Placé avant la mantisse pour simplifier la comparaison.
 - Codé sur p bits et biaisé pour être positif (ajout de $2^{p-1}-1$).
- Mantisse normalisée (**M**) :
 - Normalisée : la virgule est placée après le bit à 1 ayant le poids fort.
 - M est codé sur q bits
 - Exemple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



➤ 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**).



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



▶ Conversion décimale - IEEE754 (Codage d'un réel)

$35,5_{(10)} = ?_{(2)}$ (IEEE 754 simple précision)
 Nombre positif, \Rightarrow donc $S = 0$
 $35,5_{(10)} = 100011,1_{(2)}$ (virgule fixe)
 $= 1,000111 \times 2^5$ (virgule flottante)
 Exposant = $Eb - 127 = 5$, donc $Eb = 132$
 $1,M = 1,000111$ donc $M = 00011100\dots$
 0 1000100 000111000000000000000000 (IEEE 754 SP)
 S Eb M

▶ Conversion IEEE754 – Décimale (Evaluation d'un réel)

0 1000001 111000000000000000000000 (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
Bit de signe	1	1
Bits d'exposant	8	11
Bits de mantisse	23	52
Nombre total de bits	32	64
Codage de l'exposant	Excédant 127	Excédant 1023
Variation de l'exposant	-126 à +127	-1022 à +1023
Plus petit nombre normalisé	2^{-126}	2^{-1022}
Plus grand nombre normalisé	Environ 2^{+128}	Environ 2^{+1024}

d. 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.

► 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)

LSS	MSB	8	9	A	B	C	D	E	F
		1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	1	2	3	4	5	6	7
1	0001	8	9	10	11	12	13	14	15
2	0010	16	17	18	19	20	21	22	23
3	0011	24	25	26	27	28	29	30	31
4	0100	32	33	34	35	36	37	38	39
5	0101	40	41	42	43	44	45	46	47
6	0110	48	49	50	51	52	53	54	55
7	0111	56	57	58	59	60	61	62	63
8	1000	64	65	66	67	68	69	70	71
9	1001	72	73	74	75	76	77	78	79
A	1010	80	81	82	83	84	85	86	87
B	1011	88	89	90	91	92	93	94	95
C	1100	96	97	98	99	100	101	102	103
D	1101	104	105	106	107	108	109	110	111
E	1110	112	113	114	115	116	117	118	119
F	1111	120	121	122	123	124	125	126	127

- 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.

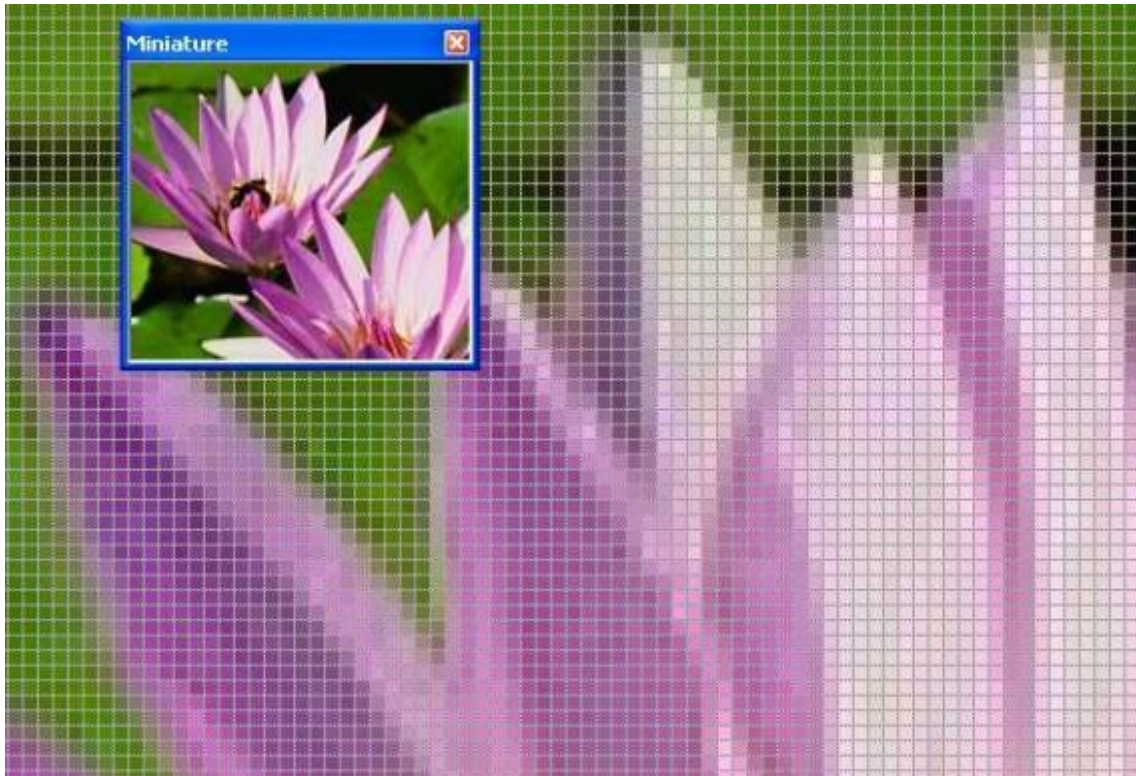
	FE7	FE8	FE9	FEA	FEB	FD0
0	ﻻ	ﺀ	ﺏ	ﺝ	ﺯ	ﻻ
1	ﻻ	ﺀ	ﺏ	ﺝ	ﺯ	ﻻ
2	ﻻ	ﺀ	ﺏ	ﺝ	ﺯ	ﻻ
3	ﻻ	ﺀ	ﺏ	ﺝ	ﺯ	ﻻ
4	ﻻ	ﺀ	ﺏ	ﺝ	ﺯ	ﻻ

e. 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 (**P**ICTURE **E**LEMENT) :
 - 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.

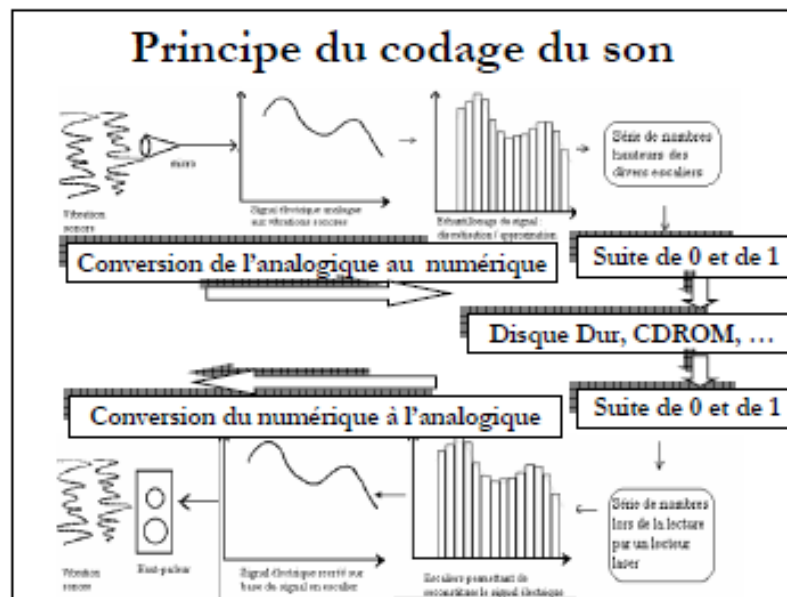
Le **poids** d'une image (exprimé en Ko ou en Mo) : est égal à son nombre de pixels (définition) que multiplie le poids de chacun des pixels (profondeur).

f. 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.

Exercices

Série de TD N°1

(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 $(\pm 1, M \cdot 2^{Eb})$ 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 30).
- 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 IEEE754, les deux nombres $AE8_{(16)}$ et $9D0_{(16)}$ puis calculer leur valeur décimale.

Chapitre 2 : Logique combinatoire et séquentielle

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. [1]

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. 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) :

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

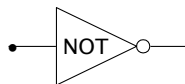
Les composants de base d'un circuit logique sont : les **portes logiques**.

► 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.

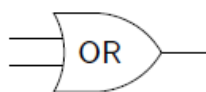
a. Fonctions logiques élémentaires

i. Fonction NON (NOT)



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

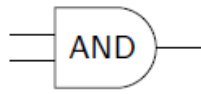
ii. Fonction OU (OR)



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

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

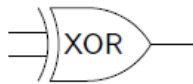
iii. Fonction ET (AND)



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

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

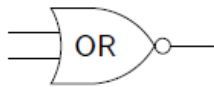
iv. Fonction OU-exclusif (XOR)



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

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

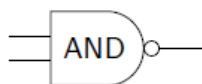
v. Fonction NON-OU (NOR)



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

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

vi. Fonction NON-ET (NAND)



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

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

► Exemple des fonctions booléennes de 2 variables :

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

b. Règles de calcul

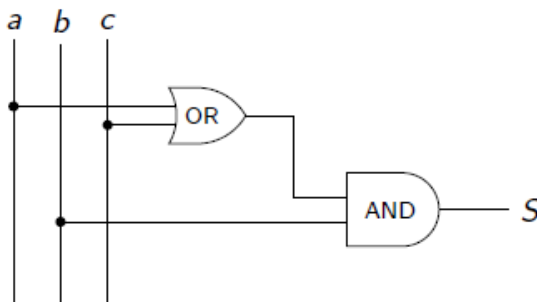
Commutativité	Associativité	Distributivité
$a + b = b + a$ $ab = ba$ $a \oplus b = b \oplus a$	$a + (b + c) = (a + b) + c = a + b + c$ $a(bc) = (ab)c = abc$ $a \oplus (b \oplus c) = (a \oplus b) \oplus c = a \oplus b \oplus c$	$a + (bc) = (a + b)(a + c)$ $a(b + c) = (ab) + (ac) = ab + ac$ $a(b \oplus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

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 :

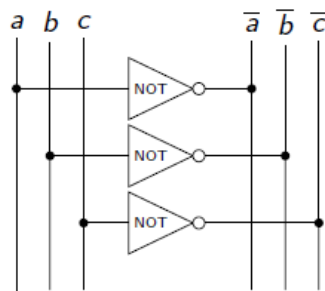
1) Ecrire l'équation de la fonction à partir de sa table de vérité

<i>a</i>	<i>b</i>	<i>c</i>	<i>S</i>
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*.

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

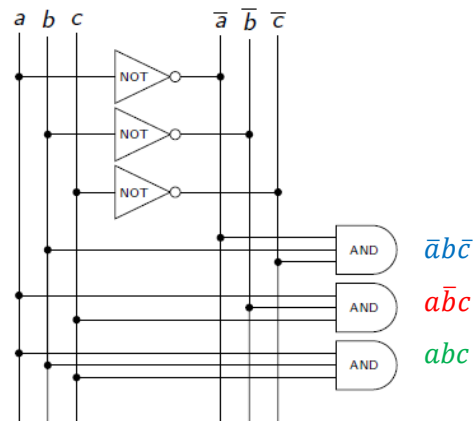
2) Réaliser la négation de toutes les variables d'entrée



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

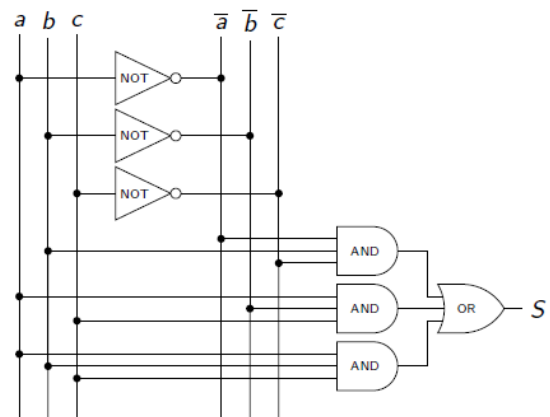
4) Etablir le câblage des portes OR avec les entrées appropriées.

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



5) Réunir l'ensemble des sorties des portes AND vers une porte OR, dont la sortie est 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).

- Exemple : la fonction majoritaire \Rightarrow donne 1 en sortie si la majorité des bits en entrée sont des 1 sinon 0.

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

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 \Rightarrow 1 case
n variables \Rightarrow 2ⁿ 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.

<i>a</i>	<i>b</i>	<i>c</i>	<i>S</i>
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

⇒

<i>a</i>	<i>b</i>	<i>c</i>	<i>S</i>
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.

<i>a</i>	<i>b</i>	<i>c</i>	<i>S</i>
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

⇒

<i>bc</i> \ <i>a</i>	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.

<i>bc</i> \ <i>a</i>	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.

	<i>bc</i>	00	01	11	10
<i>a</i>					
0		0	0	1	0
1		0	1	1	1

$$\Rightarrow S = f(a, b, c) = bc + ac + ab$$

2. Circuits combinatoires

a. Définitions

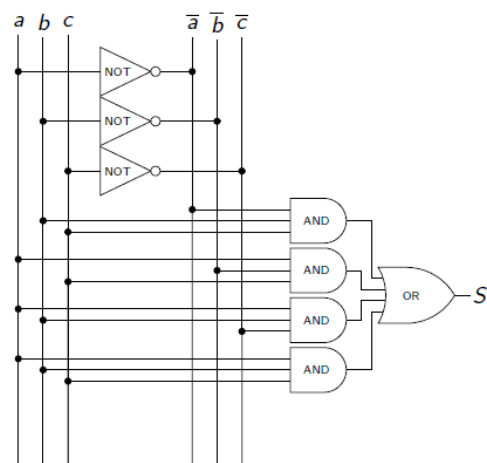
Les circuits combinatoires sont des circuits logiques combinés pour obtenir un nouveau circuit avec une fonction logique plus complexe. Ils sont nés du besoin de circuits logiques à plusieurs entrées et plusieurs sorties, et parfois des sorties qui dépendent d'entrées supplémentaires dites de sélection.

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

<i>a</i>	<i>b</i>	<i>c</i>	<i>S</i>
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. Circuits combinatoires de base

Les circuits combinatoires les plus communs :

- Le multiplexeur
- Le démultiplexeur
- Le décodeur
- Le comparateur
-

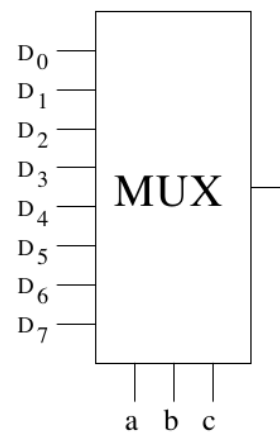
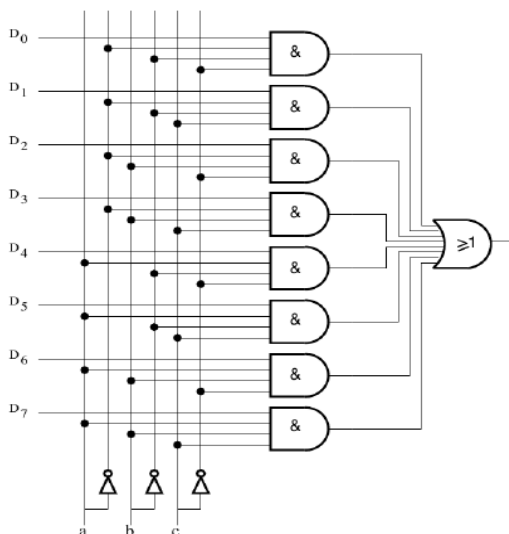
i. Le multiplexeur $2^n \times n$

- **Entrées** :
 - 2^n lignes d'entrée (données) : D_0, \dots, D_{2^n-1}
 - n lignes de sélection : a, b, c, \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.

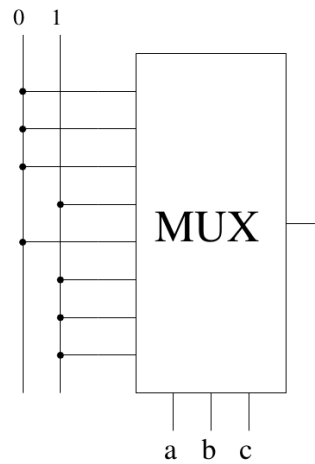
a	b	c	S
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

► Câblage du multiplexeur 8×3



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

<i>a</i>	<i>b</i>	<i>c</i>	<i>S</i>
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

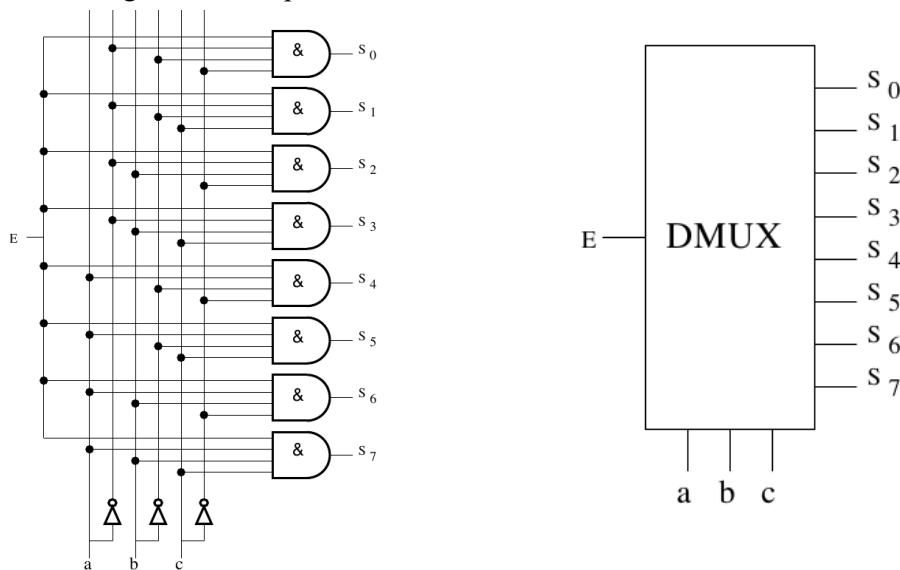


ii. Le démultiplexeur $2^n \times n$

- **Entrées :**
 - une ligne d'entrée (donnée) : *E*
 - *n* lignes de sélection : *a, b, c, ...*
- **Sortie :** 2^n lignes de sortie S_0, \dots, S_{2^n-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.

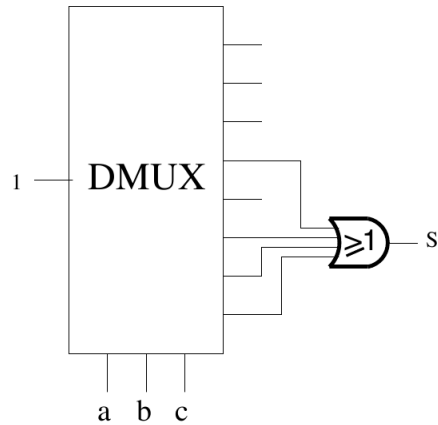
<i>a</i>	<i>b</i>	<i>c</i>	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	<i>E</i>	0	0	0	0	0	0	0
0	0	1	0	<i>E</i>	0	0	0	0	0	0
0	1	0	0	0	<i>E</i>	0	0	0	0	0
0	1	1	0	0	0	<i>E</i>	0	0	0	0
1	0	0	0	0	0	0	<i>E</i>	0	0	0
1	0	1	0	0	0	0	0	<i>E</i>	0	0
1	1	0	0	0	0	0	0	0	<i>E</i>	0
1	1	1	0	0	0	0	0	0	0	<i>E</i>

► Câblage du démultiplexeur 8×3



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

<i>a</i>	<i>b</i>	<i>c</i>	<i>S</i>
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



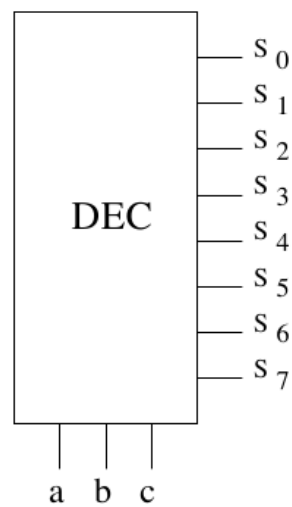
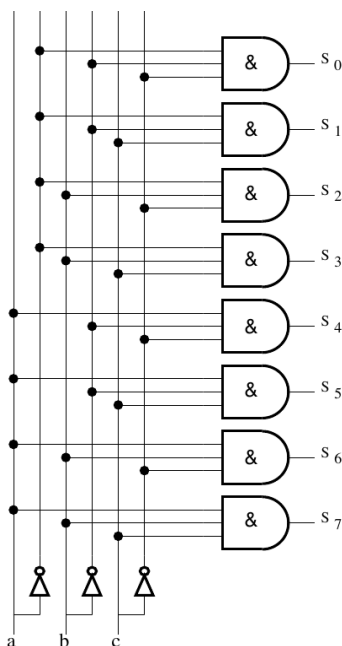
iii. Le décodeur $2^n \times n$

- **Entrées** : n lignes de sélection : a, b, c, \dots
- **Sortie** : 2^n lignes de sortie S_0, \dots, 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.

<i>a</i>	<i>b</i>	<i>c</i>	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

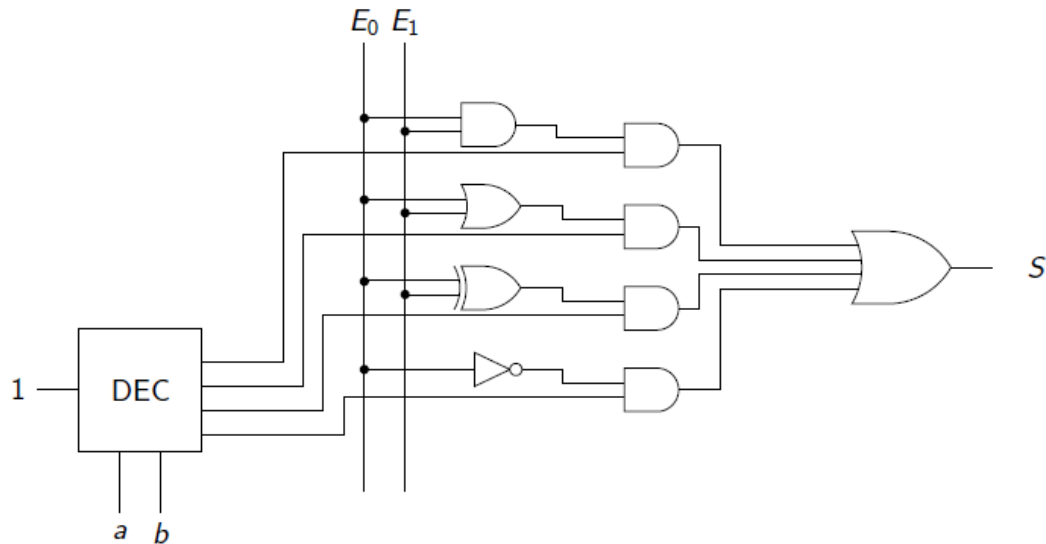


► Exemple d'utilisation d'un décodeur : activation de fonction

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	$\overline{E_0}$

Réaliser le circuit logique correspondant en utilisant un décodeur



3. Circuits arithmétiques

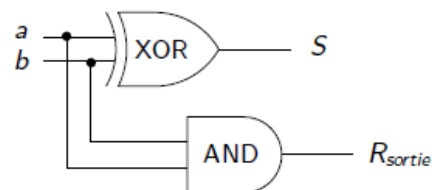
Les circuits arithmétiques sont des circuits combinatoires permettant d'effectuer des opérations arithmétiques sur les nombres en entrée. Les circuits arithmétiques de base sont :

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

a. 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

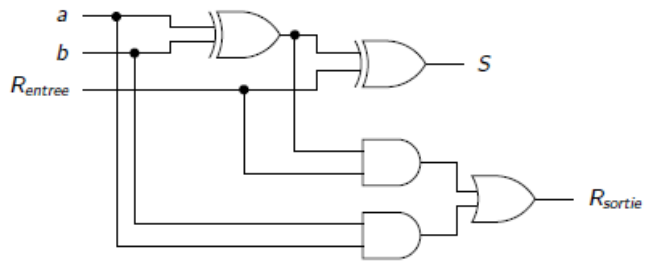


► **Problème** : si plusieurs additions successives, comment reporter la retenue ?

b. Additionneur complet

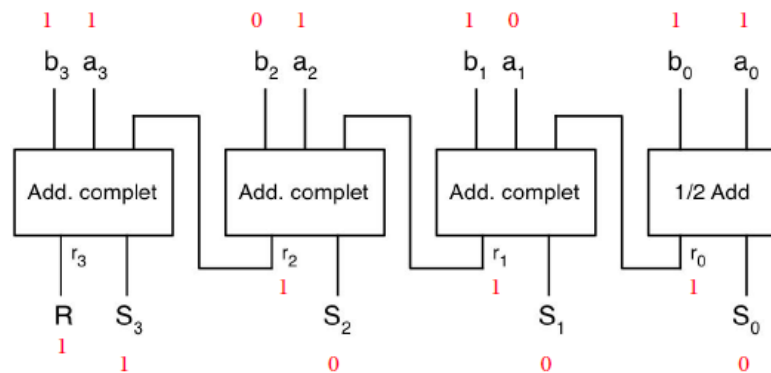
- **Entrées** :
 - Les 2 bits à additionner a et b .
 - La retenue d'entrée $R_{entrée}$
- **Sorties** :
 - La somme $S = a + b + R_{entrée}$
 - La retenue de sortie R_{sortie}
- **Rôle** : Additionner a et b en prenant en compte la retenue d'entrée $R_{entrée}$ et en conservant la retenue de sortie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

a	b	$R_{entrée}$	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 : $A = a_3 a_2 a_1 a_0 = 1 1 0 1_{(2)}$ $B = b_3 b_2 b_1 b_0 = 1 0 1 1_{(2)}$



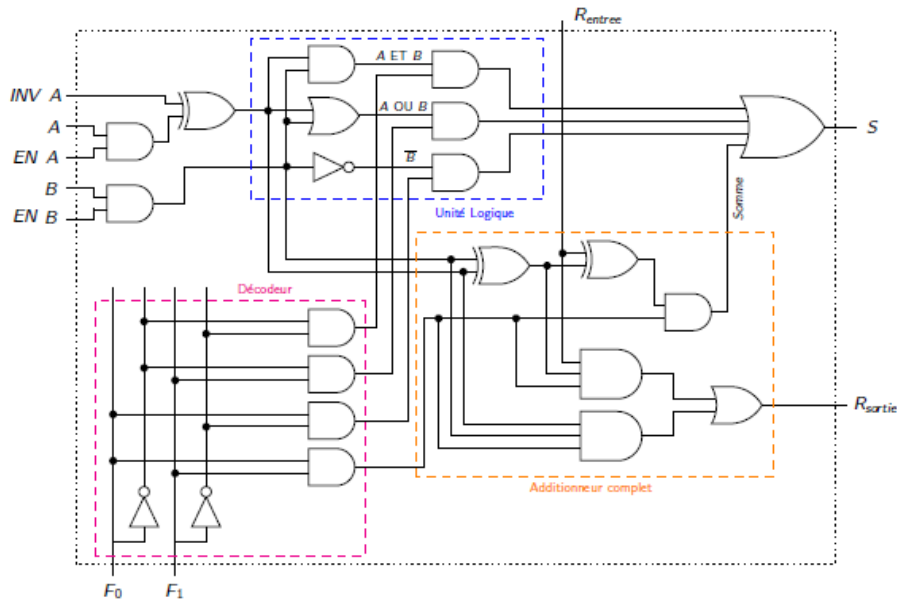
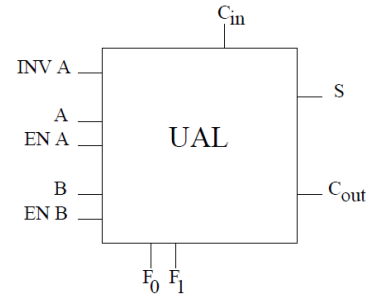
► $S = A + B = S_3 S_2 S_1 S_0 = 1 0 0 0_{(2)}$ avec retenue $R = 1$.

c. Unité Arithmétique et Logique

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

Elle est composée d'un ensemble de circuits combinatoires.

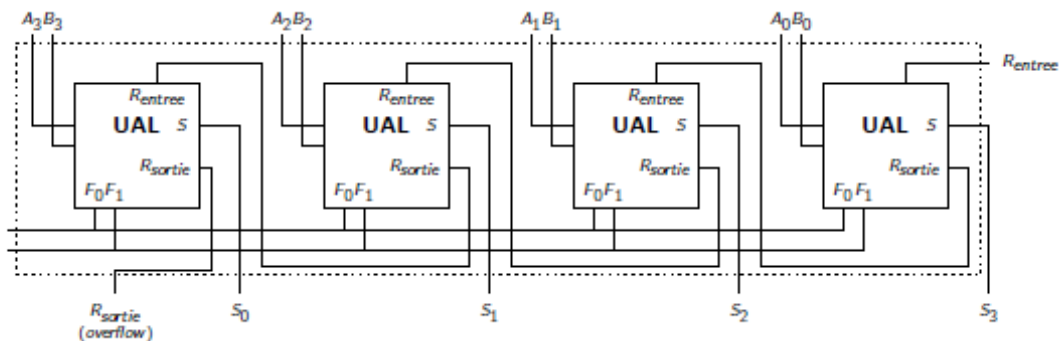
- **Entrées :**
 - A et B : les variables (données)
 - F_0 et F_1 : bits de choix du signal d'activation
 - R_{entree} (C_{in}) : la retenue d'entrée
 - EN A et EN B : les bits inhibiteurs de A et B (optionnel)
 - INV A : pour obtenir A (optionnel)
- **Sorties :**
 - S : le résultat de l'opération
 - 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
 - $A + B + R_{entree}$



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.

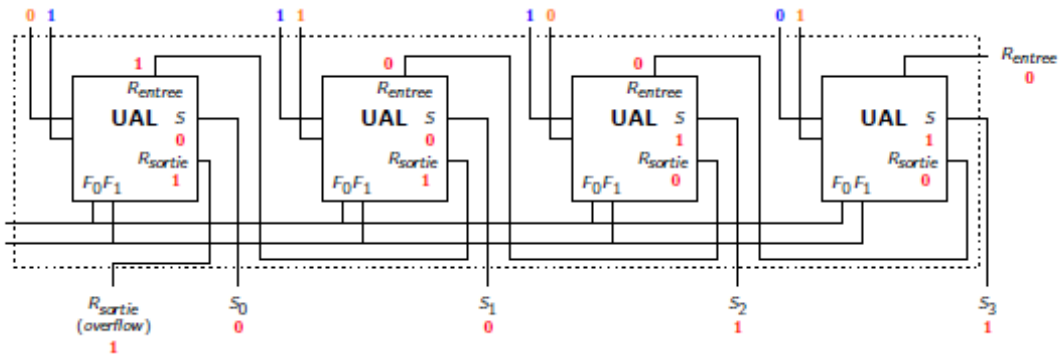


► Exemple UAL à 4 bits :

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

$$A = 14_{(10)} = a_3 a_2 a_1 a_0 = 1 1 1 0_{(2)}$$

$$B = 5_{(10)} = b_3 b_2 b_1 b_0 = 0 1 0 1_{(2)}$$



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

► UAL - Résumé des fonctions :

F_0	F_1	$EN A$	$EN B$	$INVA$	$R_{entrée}$	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	\bar{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. 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.

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 :

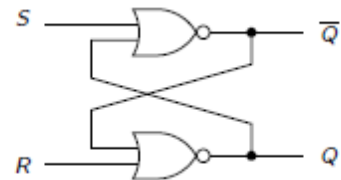
- permet de mémoriser un bit.
- « se souvient » de la valeur que le circuit a enregistrée.
- est construite avec une ou deux portes logiques NON-OU (ou NON-ET).
- comporte une ou plusieurs entrées.
- comporte une ou deux sorties.

La sortie maintient son état même après disparition du signal de commande

⇒ **Logique séquentielle**

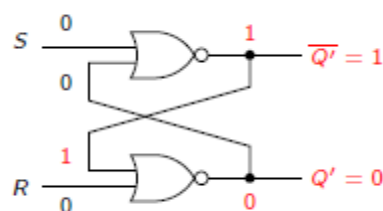
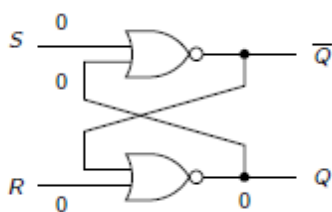
i. Bascule RS

- **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})



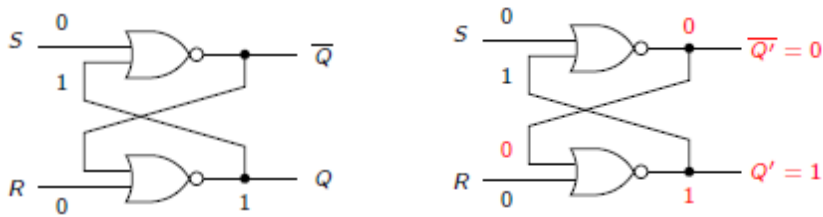
► Bascule RS : états stables

- 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 $Q' = 1$

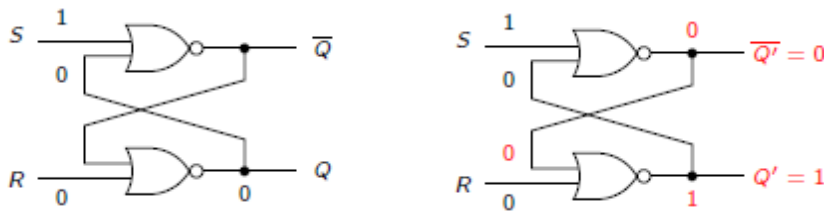


Bascule RS à l'état 1

- 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 .

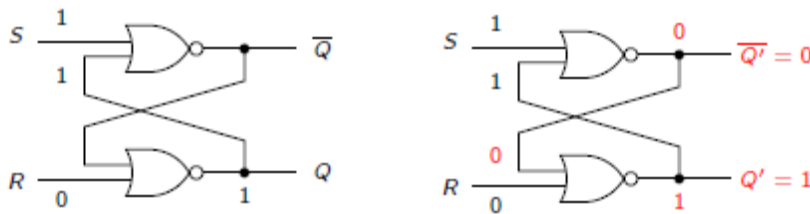
► Bascule RS : activation

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



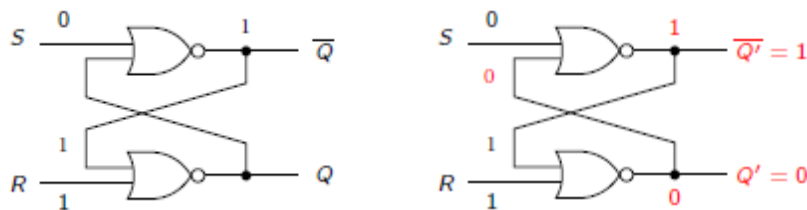
Bascule RS à l'état 1

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



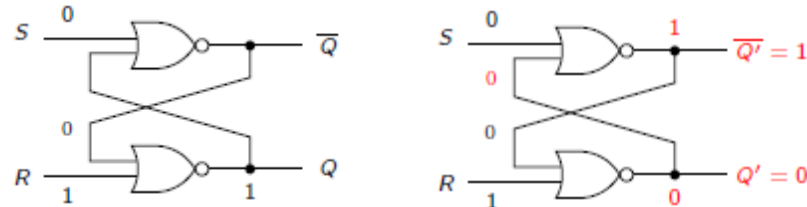
Bascule RS à l'état 1

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



Bascule RS à l'état 0

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

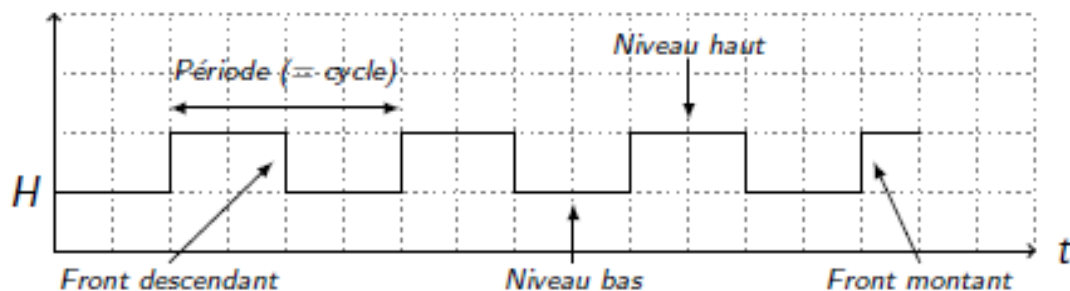
ii. Bascule RSH

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).

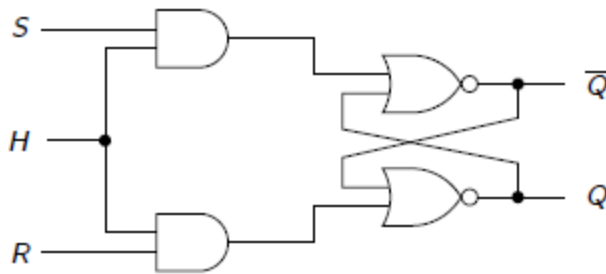
► Horloge :

- 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 μ s.



► 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 $t = n + 1$)

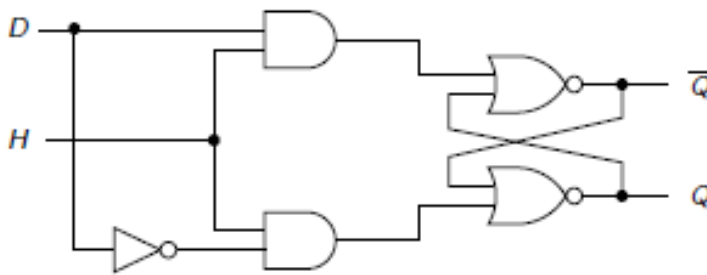


S_n	R_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	?

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

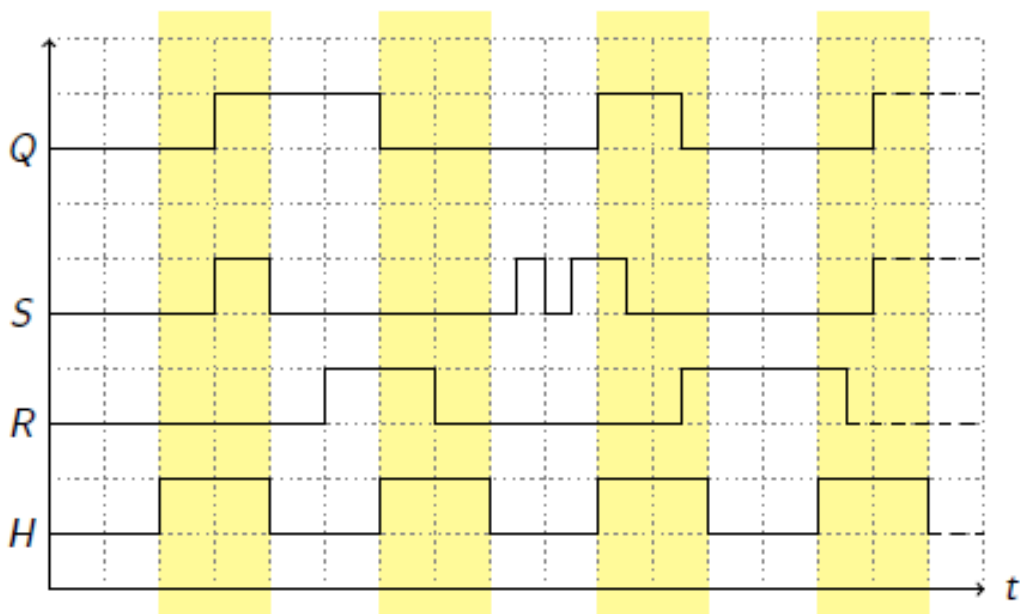
iii. Bascule D

- 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.



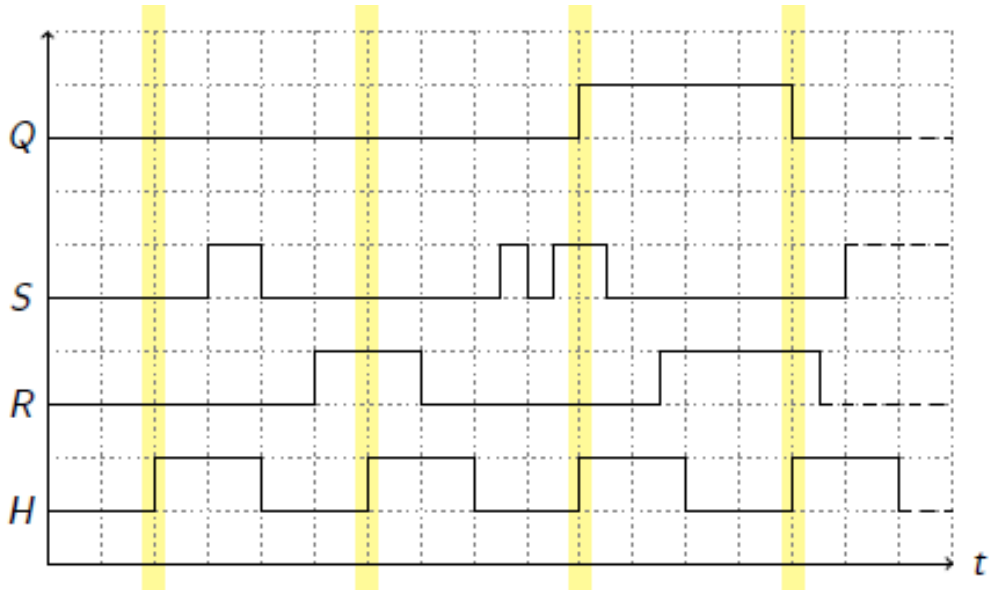
► Bascule latch :

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

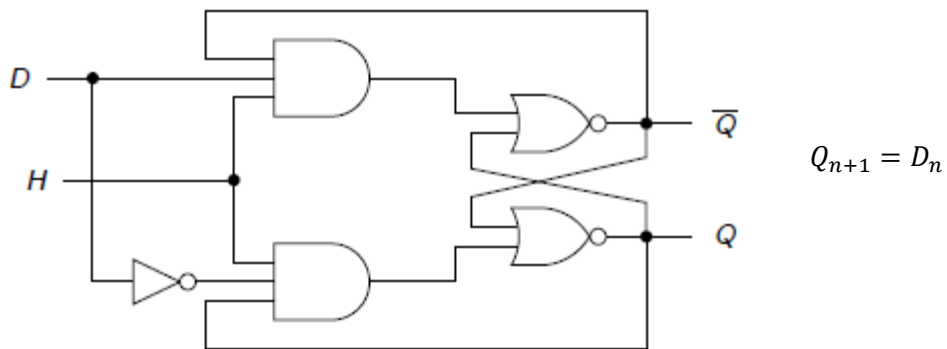


► Bascule flip-flop :

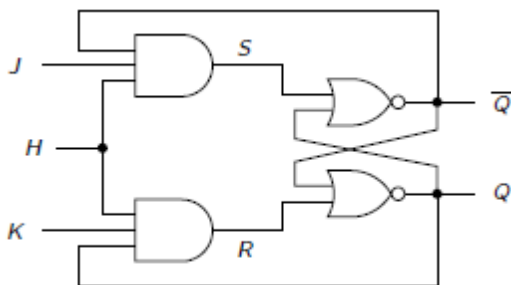
- Bascule synchrone
- Change d'état lorsque l'horloge est en front montant



► Bascule D (flip-flop)



iv. Bascule JK (flip-flop)



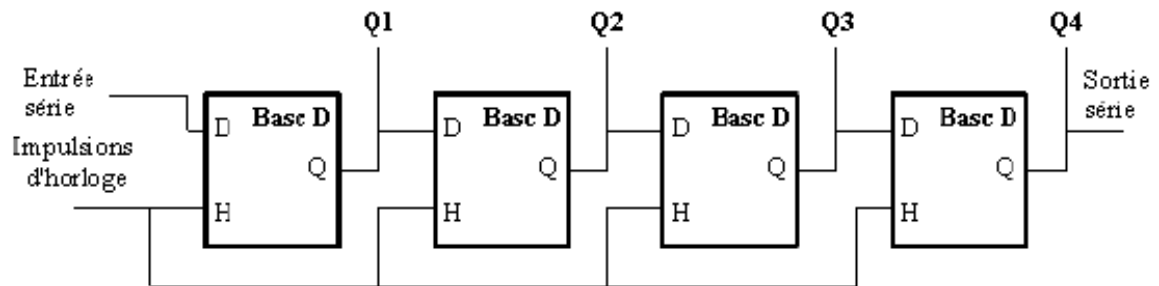
J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

J_n	K_n	Q_n	$\overline{Q_n}$	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

$$Q_{n+1} = J_n \overline{Q_n} + \overline{K_n} Q_n$$

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...

Exercices

Série de TD N°2

(Logique combinatoire et séquentielle)

Exercice 1 : (Fonctions logiques)

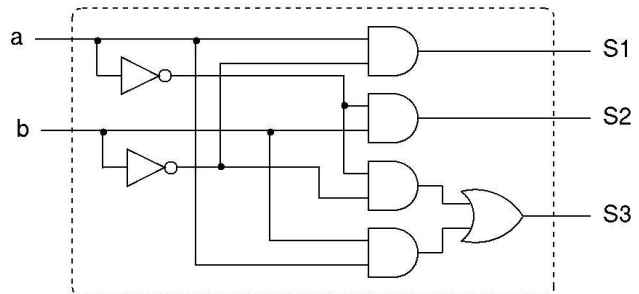
Soit un entier de 0 à 7 représenté par 3 bits $b_2b_1b_0$. Soit F la fonction logique dont les entrées sont ces 3 bits et qui prend la valeur 1 si la valeur de l'entrée vaut 0, 3, 4 ou 7 (codée en binaire), et 0 sinon.

- 1/ Donner la table de vérité de F.
- 2/ Dédurre 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)

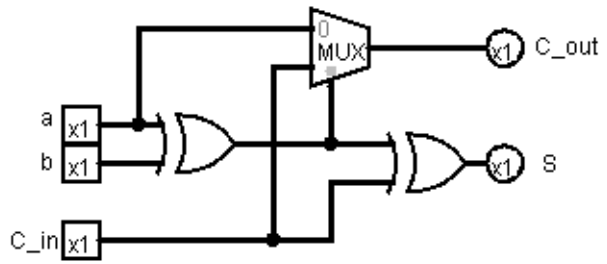
On construit un circuit qui a deux entrées (a et b) et deux sorties (s et r) et une ligne de commande F tel que :

- 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.

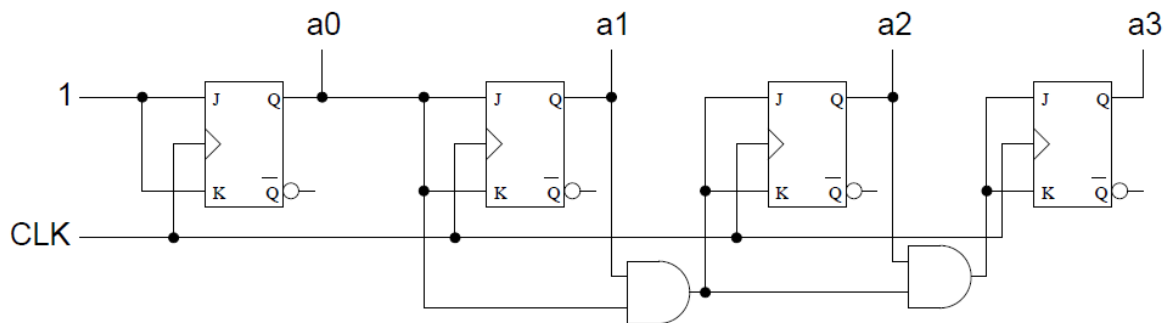
Exercice 4 : (Multiplexeur/Décodeur)

- 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*).

Chapitre 3 : Mémoires

1. Définitions

- Avec une bascule c'est possible de mémoriser une information sur 1 seul bit.
- Avec un registre c'est possible de mémoriser une information sur n bits.
- Si on veut mémoriser une information de taille importante \Rightarrow il faut utiliser une mémoire.

C'est quoi une **mémoire** ?

- Une mémoire est un dispositif capable :
 - d'**enregistrer** une information,
 - de la **conserv**er (mémoriser)
 - et de la **restituer** (possible de la lire ou la récupérer par la suite). [6]
- ▶ Exemple de mémoire :
 - 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) [6].



- Une mémoire communique avec les autres composants 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 :

- Bit : un 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

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 source 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.

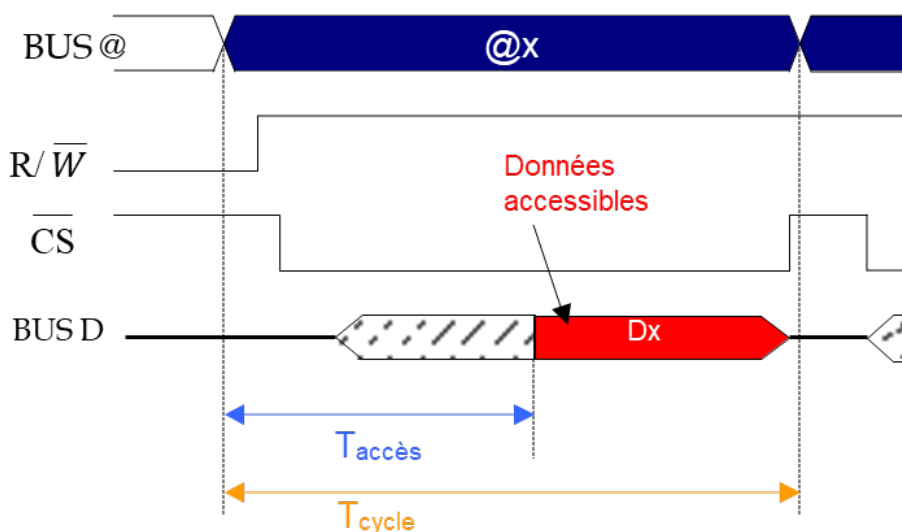
Il existe des mémoires qui offrent les deux modes lecture/écriture, ces mémoires s'appellent mémoires vives.

Il existe des mémoires qui offrent uniquement la possibilité de la lecture (ce n'est pas possible de modifier le contenu). Ces mémoires s'appellent 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. [6]



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

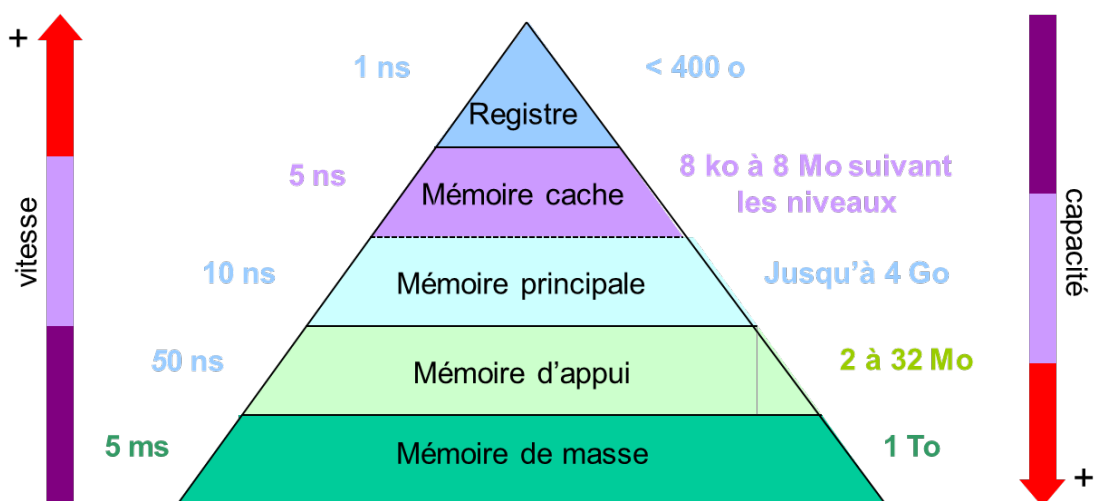
C'est l'organe principal de rangement des informations. Elle contient les programmes (instructions et données) et est plus lente que les deux mémoires précédentes.

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

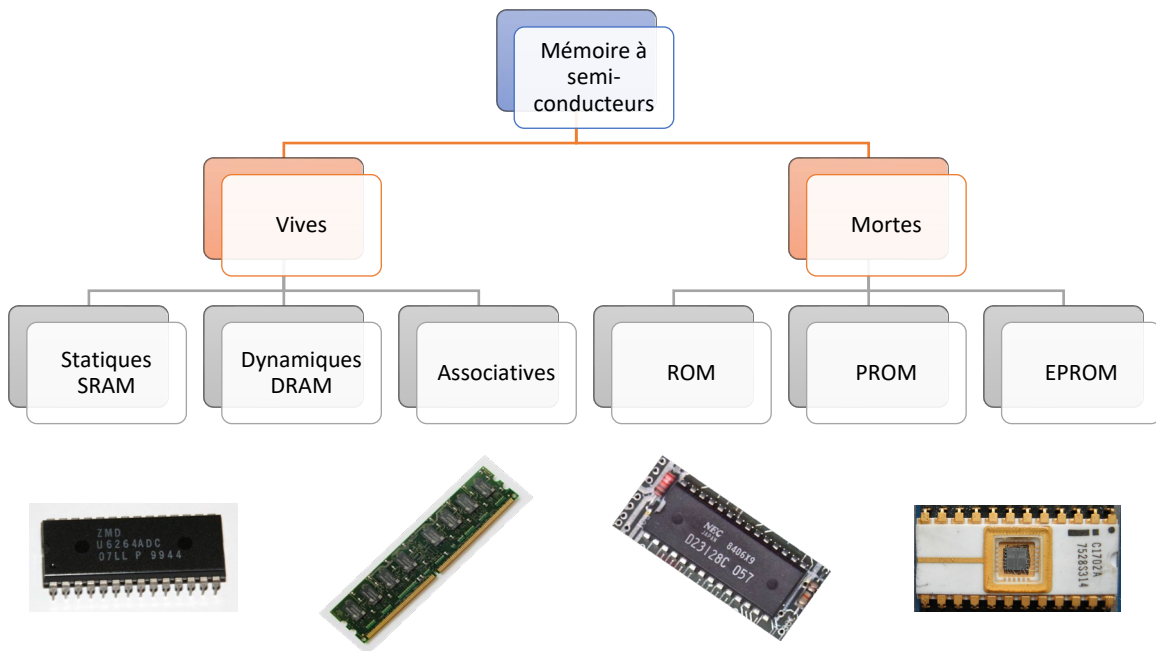
C'est une mémoire périphérique de grande capacité utilisée pour le stockage permanent ou la sauvegarde des informations. Elle utilise pour cela des supports magnétiques (disque dur, ZIP) ou optiques (CDROM, DVDROM) .[6]



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 réduite.
- Mémoire magnétique (disque dur, disquette, ...) : moins rapide mais stocke un volume d'informations très grand.
- Mémoire optique (DVD, CDROM, ...)



5. 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 : Random Access Memory) 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.

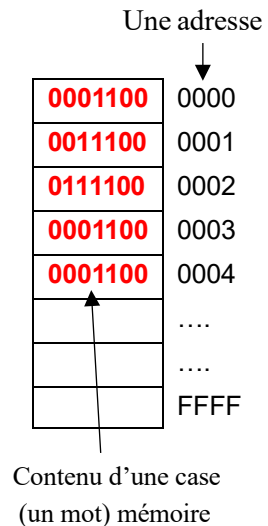
Pour la communication avec les autres organes de l'ordinateur, la mémoire centrale utilise les bus (bus d'adresses et bus de données)

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.

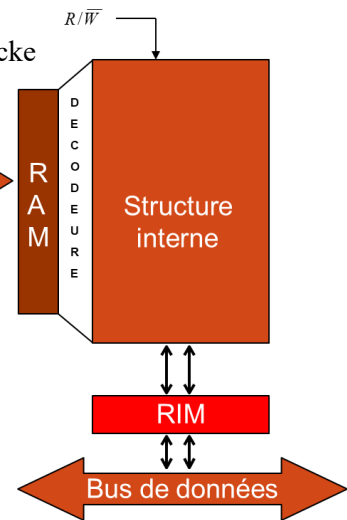
a. Vue logique de la mémoire centrale

- 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.



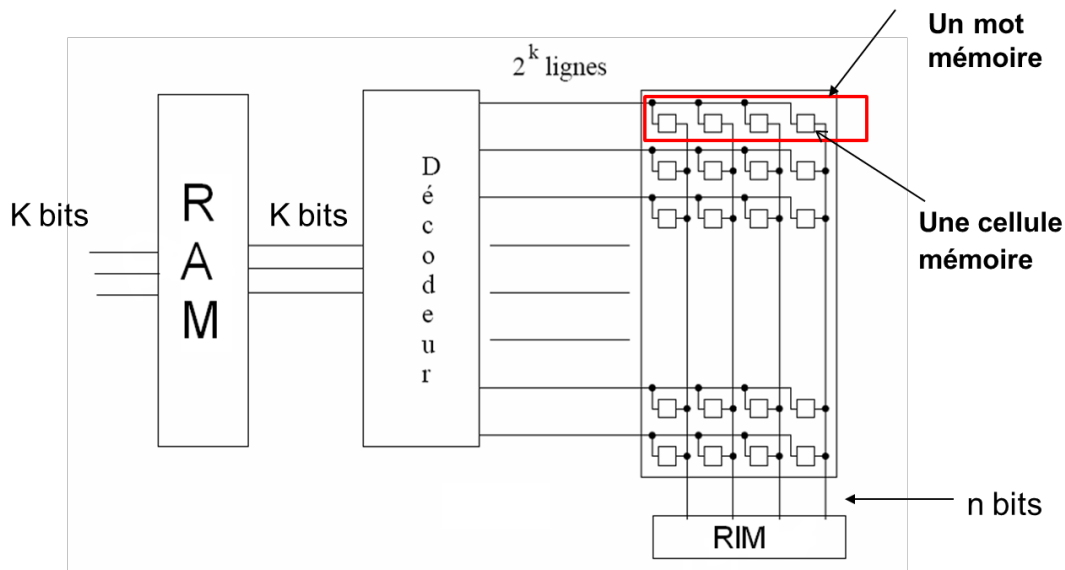
b. Structure physique d'une mémoire centrale

- **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.
- **R/W** : commande de lecture/écriture, cette commande permet de lire ou d'écrire dans la mémoire (si $R/\bar{W} = 1$ 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 ?

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

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

► 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 ($R/\bar{W} = 1$).
- 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.

Exercices

Série de TD N°3

(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 KOctets 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 MOctets, 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 MOctets
- 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 4 : 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. [6]

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. 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. [7]

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 [8]. Il semble que Gilbert Hyatt n'ait pas abandonné et espère faire revoir cette décision.

Le premier microprocesseur commercialisé, le 15 novembre 1971, 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. [9]



Le microprocesseur exécute le programme, qui est une suite d'instructions. [10]

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

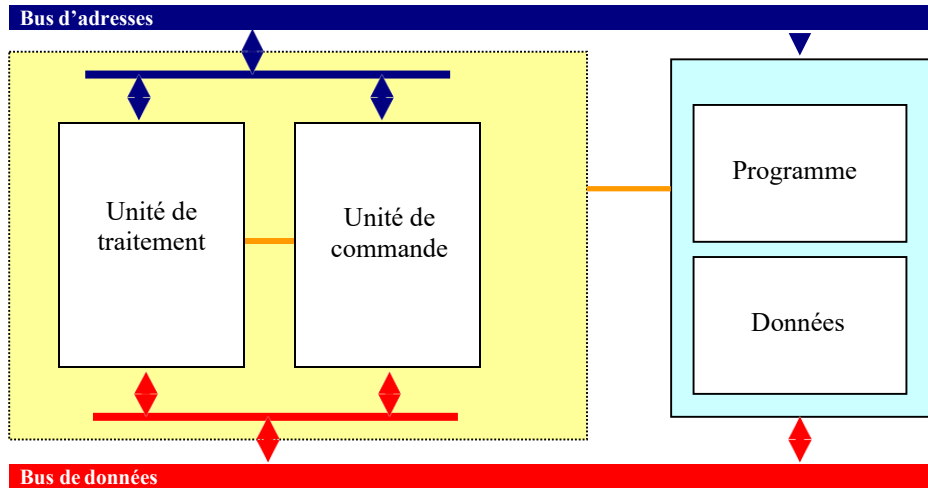
- Lecture (LOAD) ou Ecriture (STORE) en mémoire
- Opération logique (ET, OU, etc.)
- Opération arithmétique (addition, soustraction, etc.)

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 [6].



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

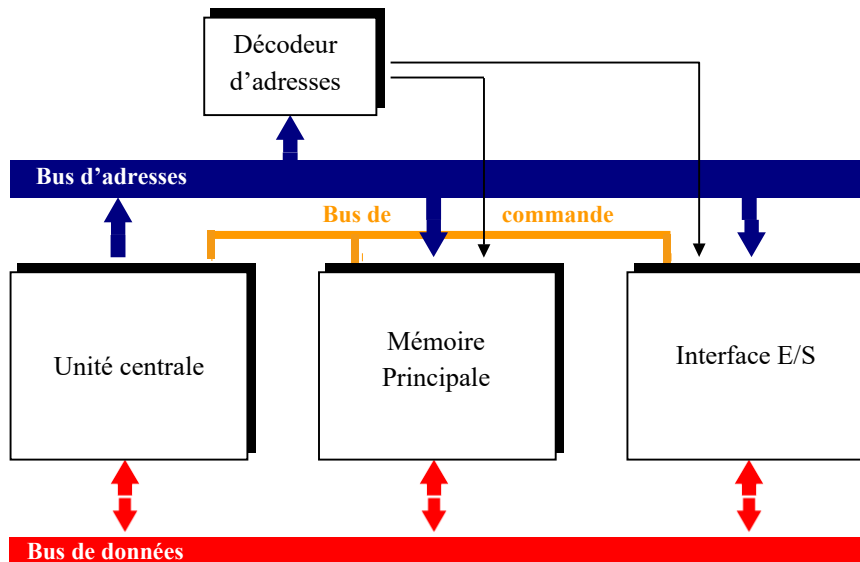
b. L'unité de traitement

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 2^k emplacements, 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.

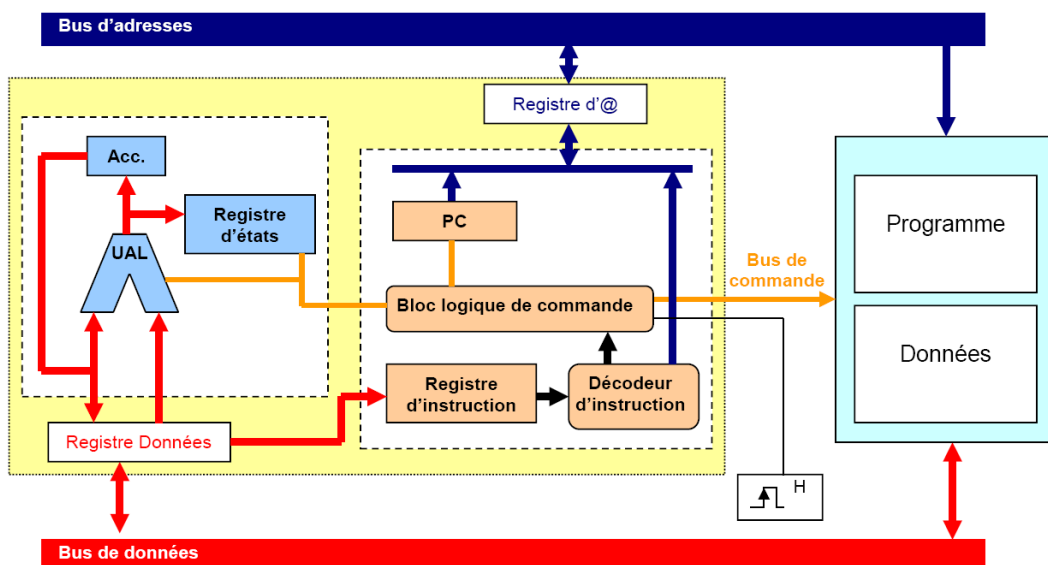
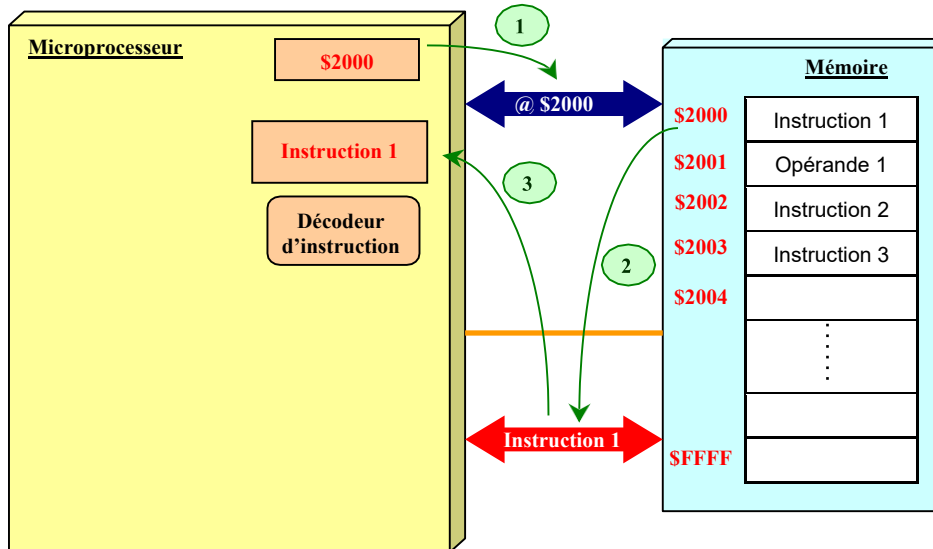


Figure 6 : Le schéma fonctionnel du microprocesseur

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 :

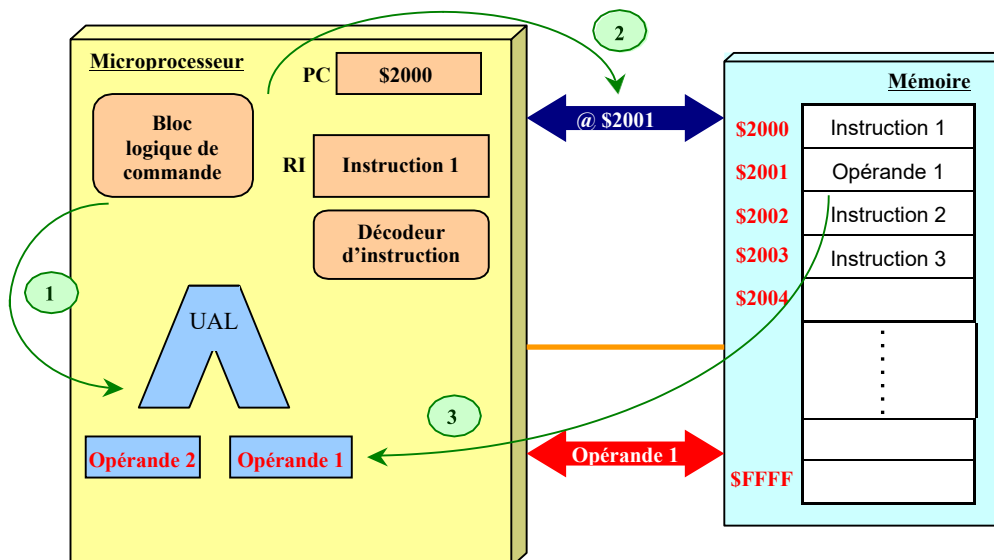
- **Phase 1** : Recherche de l'instruction à traiter
 1. Le PC contient l'adresse de l'instruction suivante du programme. Cette valeur est placée sur le bus d'adresses par l'unité de commande qui émet un ordre de lecture.
 2. Au bout d'un certain temps (temps d'accès à la mémoire), le contenu de la case 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

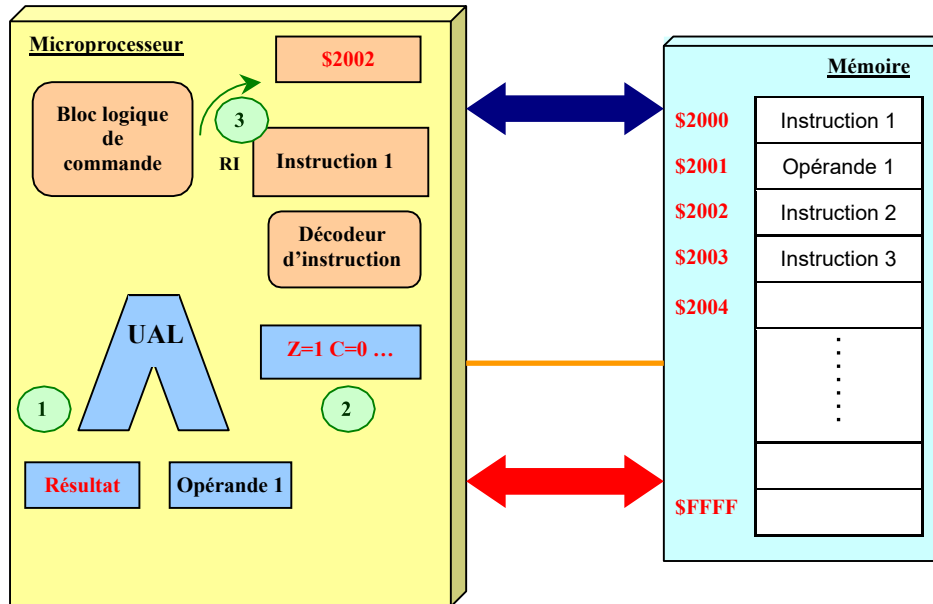
Le registre d'instruction contient maintenant le premier mot de l'instruction qui peut être codée sur plusieurs mots. Ce premier mot contient le code opératoire qui définit la nature de l'opération à effectuer (addition, rotation, ...) et le nombre de mots de l'instruction.

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.



4. 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. [6]

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

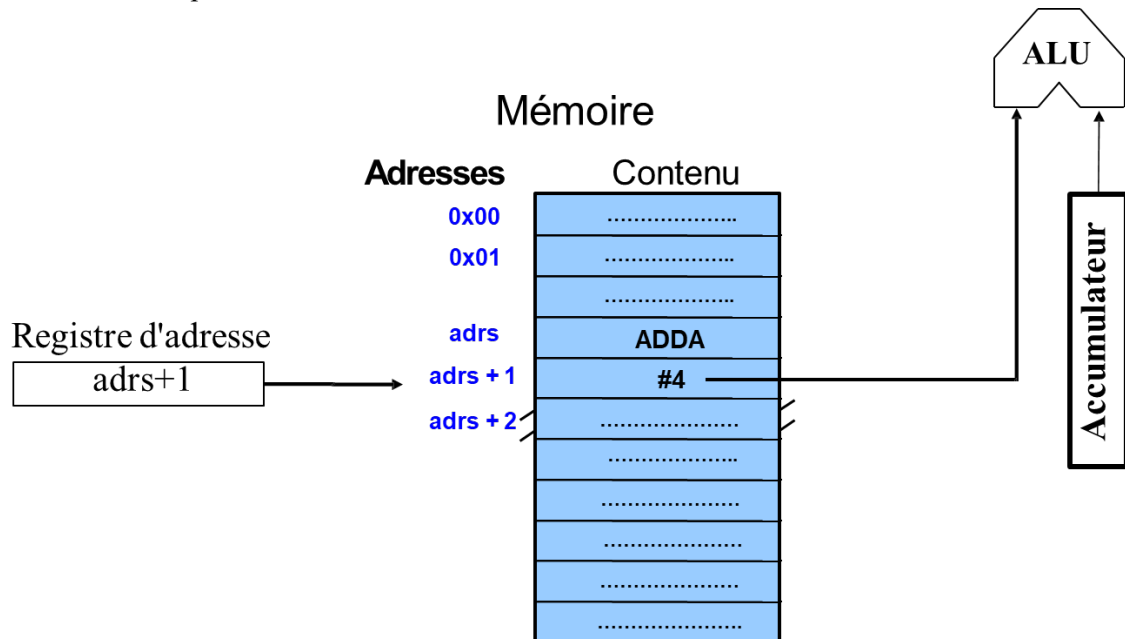
Le nombre d'instructions du jeu d'instructions est directement lié au format du code instruction. Ainsi un octet permet de distinguer au maximum 256 instructions différentes.

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 :

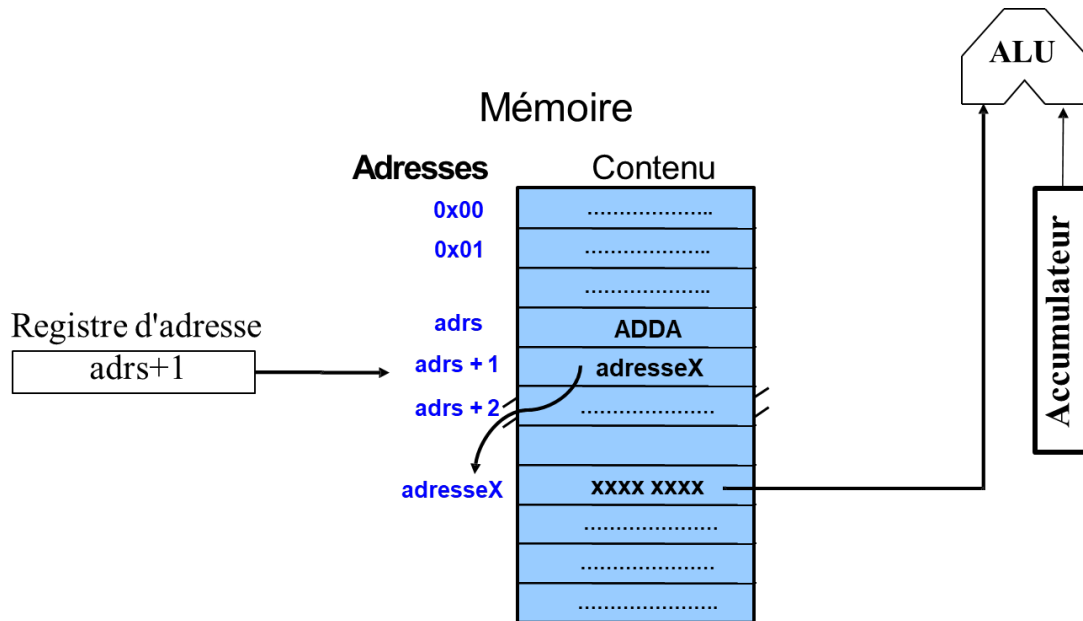
i. Adressage immédiat

- Exemple : ADDA #4



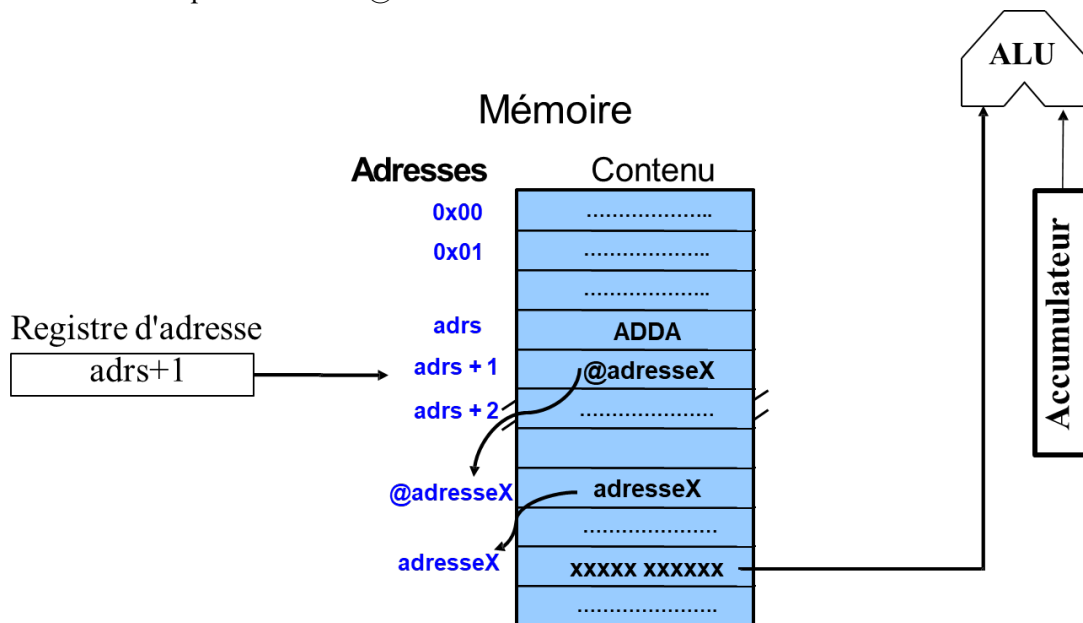
ii. Adressage direct

► Exemple : ADDA adresseX



iii. Adressage indirect

► Exemple : ADDA @adresseX



e. Notion d'architecture RISC et CISC

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

- L'architecture **CISC** (*Complex Instruction Set Computer*)
- L'architecture **RISC** (*Reduced Instruction Set Computer*)

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
<ul style="list-style-type: none"> • Instructions simples ne prenant qu'un seul cycle 	<ul style="list-style-type: none"> • Instructions complexes prenant plusieurs cycles
<ul style="list-style-type: none"> • Instructions au format fixe 	<ul style="list-style-type: none"> • Instructions au format variable
<ul style="list-style-type: none"> • Décodeur simple (câblé) 	<ul style="list-style-type: none"> • Décodeur complexe (microcode)
<ul style="list-style-type: none"> • Beaucoup de registres 	<ul style="list-style-type: none"> • Peu de registres
<ul style="list-style-type: none"> • Seules les instructions LOAD et STORE ont accès à la mémoire 	<ul style="list-style-type: none"> • Toutes les instructions sont susceptibles d'accéder à la mémoire
<ul style="list-style-type: none"> • Peu de modes d'adressage 	<ul style="list-style-type: none"> • Beaucoup de modes d'adressage
<ul style="list-style-type: none"> • Compilateur complexe 	<ul style="list-style-type: none"> • 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.

Exercices

Série de TD N°4

(Les microprocesseurs)

Exercice 1 : (Micro-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 : (Étapes 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.

$$X = (A + B * C) / (D - E * F).$$

Références

- [1] E. Lazard, *Architecture de l'ordinateur*. Pearson Education France, 2006.
- [2] A. Krähenbühl, "Architecture des ordinateurs," 2014, Available: <http://adrien.krahenbuhl.fr/fr/enseignement/>, Accessed on: 06/02/2021.
- [3] J. Delacroix and A. Cazes, *Architecture des machines et des systèmes informatiques*. Sciences Sup, Dunod, 2011.
- [4] P. Darche and I. P. Descartes, *Architecture des ordinateurs*. Vuibert, 2012.
- [5] J.-M. Richer. (2000, 03/02/2021). *Page de Jean-Michel Richer*. Available: http://www.info.univ-angers.fr/~richer/ensl3i_crs1.php
- [6] T. Dumartin, "Architecture des ordinateurs - Note de cours," 2005, Available: https://www.academia.edu/34101357/Architecture_des_ordinateurs_Note_de_cours, Accessed on: 06/02/2021.
- [7] G. Hyatt. (1970, 06/02/2021). *Gilbert Hyatt Files the First General Patent on the Microprocessor; It is Later Invalidated*. Available: <https://www.historyofinformation.com/detail.php?entryid=121>
- [8] F. Faggin. (25/12/2016). *The Intel 4004 Microprocessor and the Silicon Gate Technology*. Available: <http://www.intel4004.com/hyatt.htm>
- [9] J. Jorda and A. M'zoughi, *Mini manuel d'architecture de l'ordinateur*. Dunod, 2012.
- [10] T. H. Greg Wyant, *Les microprocesseurs...comment ça marche ?* Dunod, 1994, p. 216.