République Algérienne Démocratique & Populaire

Ministère de L'Enseignement Supérieure et de la recherche Scientifique
Centre Universitaire de Béchar
Département de Science de Gestion

Initiation à l'informatique

Préparé par : *Bendjima Mostefa*

Avril 2015

Présentation générale

1. Introduction

C'est en 1962 que Philippe Dreyfus employé le mot informatique pour définir le traitement automatique de l'information. En fait ce mot peut correspondre à deux groupes de disciplines distinctes: l'ensemble des techniques mises en oeuvre pour l'emploi de l'ordinateur *(electronic data processing)*, une science nouvelle, qui ne nécessite pas obligatoirement l'utilisation des ordinateurs, ces derniers n'en sont qu'un outil majeur *(computer science)*.
On peut donc affirmer que l'informatique est une discipline carrefour, dont les ordinateurs actuels, les structures intellectuelles (algorithmes de calcul) et institutionnelles (organisation comptable, organisation industrielle) déterminent en majeure partie le contenu.
D'un point de vue plus général, il n'est plus à démontrer que l'informatique envahit progressivement, sous beaucoup de formes, notre vie quotidienne, tant est puissant son développement. L'informatique aujourd'hui, c'est à la fois les calculettes de poche et les ordinateurs portables, les consoles et jeux vidéo, mais aussi l'aventure spatiale, les robots industriels, les applications médicales telles que le scanner, les cartes à puce et bien d'autres applications. L'informatique à travers l'ordinateur modifie, et modifiera plus encore, l'organisation du travail et les rapports entre les individus.

2. L'histoire de l'informatique

C'est durant la période de 10000 avant J.C. que naquit le traitement rationnel de l'information. L'homme changea son mode de vie en passant du stade de chasseur à celui d'agriculteur. Dans ce nouveau mode de vie, l'homme doit disposer de semences, d'outils, d'animaux de ferme. C'est pourquoi il lui est nécessaire de mettre en place un système de troc et de ce fait, l'homme doit apprendre à s'organiser, compter, écrire : c'est l'apparition du traitement de l'information. On découvrit alors en Mésopotamie l'utilisation de boules, de jetons d'argile et de tablettes par les Sumériens qui servait dans ce système d'échange.

En arrivant au XIIIe siècle, époque de grandes effervescences intellectuelles, pour voir apparaître des systèmes de calcul plus rapides et plus automatiques.

- XIIIe : fabrication de l'Ars Magna, par Raymond Lulle : il s'agit d'une «machine logique» faite de cercles concentriques contenant des mots qui, disposés dans un certain ordre, forment des questions tandis que d'autres mots y répondent.
- XVIe : invention du codage binaire par Francis Bacon et du logarithme (à l'origine créé pour simplifier des calculs compliqués) par Napier.
- 1624 : Wilhem Schickard construit une «horloge automatique calculante» à Heidelberg
- 1642 : Blaise Pascal, à 19 ans, crée la «Pascaline», machine à calculer mécanique à base de roues dentées, capable de faire des additions et des soustractions, le langage PASCAL sera plus tard ainsi nommé en son honneur.
- 1673 : Leibniz, grand mathématicien, améliore la Pascaline en y ajoutant la multiplication et la division ; par ailleurs, il s'intéresse beaucoup à la numérotation binaire avec laquelle il essaie de concevoir une «caractéristique universelle» dont l'objectif est de réduire toutes les opérations logiques à un calcul.
- XVIIIe : La Mettrie, philosophe disciple de Descartes, radicalise la philosophie de ce dernier et écrit *L'homme machine*, où il argumente en faveur d'une vision mécaniste du vivant (Descartes lui-même aurait construit un automate à visage humain). Les automates sont très à la mode à cette époque. L'horloger suisse Vaucansson en construit de très célèbres parmi lesquels un joueur de flûte et un canard pourvu de fonctions locomotrices et digestives, exposés à paris en 1738 : leur fonctionnement utilise un «arbre à came» (comme dans les boîtes à musique), codage binaire du mouvement. Un célèbre «joueur d'échec artificiel» parcours aussi les cours européennes à la fin du siècle (il aurait notamment battu Napoléon)

avant qu'on ne démasque la supercherie : un nain caché sous la table actionnait en fait le mécanisme.

- 1805 : Jacquart crée les métiers à tisser automatiques, qui utilisent des «programmes» sous forme de cartes perforées, également utilisées dans les pianos mécaniques.
- 1818 : Mary Shelley publie «Frankenstein», où l'électricité donne l'étincelle de vie.
- 1822 : l'ingénieur anglais Babbage fait les premiers plans de sa «machine à différences», sorte de machine à calculer mécanique utilisant les logarithmes : trop complexe pour la technologie de l'époque, elle ne sera construite d'après ces plans qu'au XXième siècle.
- 1832 : invention du langage Morse.
- 1833 : Babbage conçoit sa «analytical engine», encore plus performante (et compliquée) que la «machine à différence», utilisant des cartes perforées pour enchaîner l'exécution d'instructions élémentaires sur un calculateur universel (mécanique) : il passera sa vie et se ruinera à essayer en vain de construire sa machine. Il sera aidé par Lady Ada Lovelace, fille du poète Lord Byron, qui écrira les premiers «programmes» qu'aurait pu exécuter la machine (le langage de programmation ADA sera ainsi nommé pour lui rendre hommage). Cette machine aurait pourtant répondu aux besoins croissants en calcul dans la société anglaise, notamment pour l'astronomie et la navigation.
- 1854 : Le logicien anglais Georges Boole publie son livre *The Mathematical Analysis of Logic*, où il définit les opérateurs logiques dits «booléens», fondés sur deux valeurs 0/1 pour coder Vrai/Faux.
- 1876 : Bell invente le téléphone.
- 1884 : L'ingénieur américain Hollerith dépose un brevet de machine à calculer automatique
- 1890 : Hollerith commercialise des machines à calculer électriques, utilisées notamment pour traiter automatiquement les données d'un recensement aux Etats-Unis. Les besoins industriels en calcul automatique se multiplient.
- 1896 : Hollerith crée une société appelée «Tabulation Machine Corporation», qui deviendra en 1924, «International Business Machine» (IBM), qui existe toujours.
- 1921 : invention du mot «robot» par Karel Capek, auteur dramatique tchèque.
- 1925 : Vannevar Bush, ingénieur américain, construit un calculateur analogique au MIT (Massachusetts Institute of Technology, prestigieuse école d'ingénieur américaine).
- 1927 : la télévision et la radio deviennent opérationnels.
- 1931 : l'allemant Konrad Zuse construit un calculateur automatique, le Z1.
- 1936 : Alan Turing propose sa définition des «machines de Turing» et Church invente le «lambda-calcul», qui se révèlent avoir des capacités de calcul équivalentes.
- 1938 : fondation de Hewlett Packard, société de matériels électroniques.
- 1939 : John Atanassoff et Clifford Berry, son étudiant, conçoivent un prototype appelé ABC à l'université de l'Iowa, reconnu comme le premier ordinateur digital.
- 1939-1945 : pendant la guerre,
    - Alan Turing travaille dans le service anglais de décryptage des messages secrets allemands (codés suivant le système appelé «Enigma») : il réalise une machine à décrypter qui contribuera à la victoire des alliés, en 1941, il construit le « Colossus » à l'université de Manchester (bientôt suivi du Mark I et du Mark II), premiers ordinateurs européens avec le Z3 de Konrad Zuse qui, pour la première fois, propose un contrôle automatique de ses opérations
    - John Von Neumann, travaille sur les calculs de balistique nécessaires au projet *Manhattan* (conception et réalisation de la première bombe atomique américaine).
- 1945 : John Von Neumann écrit un rapport où il propose l'architecture interne d'un calculateur universel (ordinateur), appelée désormais «architecture de Von Neumann».
- 1946 : construction de l'ENIAC à l'Université de Pennsylvanie, dernier gros calculateur électrique programmable (mais pas universel) : il fait 30 tonnes, occupe $160m^2$ et sa mémoire est constituée de 18 000 tubes à vide, sa puissance est équivalente à celle d'une petite calculette actuelle, pendant ce temps, Wallace Eckler et John Mauchly conçoivent le

Binac (Binary Automatic Computer), qui opère pour la première fois «en temps réel» mais ne sera construit qu'en 1949, avec l'apport de Von Neumann.
- 1947 : invention du transistor (qui peut être vu comme un interrupteur miniature).
- 1948 : Claude Shannon publie sa *Théorie mathématique de l'information*, où est introduite la notion de quantité d'information d'un objet et sa mesure en bits, l'année suivante il construit la première machine à jouer aux échecs.

A partir de cette date, l'ordinateur existe et son histoire matérielle se réduit donc à l'évolution des progrès technologiques, qu'on découpe habituellement en termes de «générations». Les avancées conceptuelles les plus spectaculaires concernent, elles, principalement la conception de nouveaux langages de programmation évolués.

Première génération : les monstres

- 1949 : construction de l'EDVAC, premier ordinateur construit suivant l'architecture de Von Neumann et stockant ses données sur disques magnétiques.
- 1950 : Turing écrit un article dans une revue philosophique pour argumenter que le modèle des ordinateurs peut réaliser tout ce que fait l'esprit humain.
- 1952 : IBM commercialise les premiers ordinateurs à lampes et à tubes à vide, IBM 650 puis IBM 701.
- 1954 : premiers essais de programmation avec le langage FORTRAN (FORmula TRANslator), encore utilisé de nos jours pour le calcul scientifique.
- 1955 : invention du mot «ordinateur» en France, à la demande d'IBM.
- 1956 : le terme d'Intelligence Artificielle est inventé lors d'une conférence à Dartmouth, aux Etats-Unis.

Deuxième génération : intégration du transistor

- 1958 : l'IBM 7044, 64 Koctets de mémoire, est le premier ordinateur intégrant des transistors ; John McCarthy invente le LISP, premier langage de l'Intelligence Artificielle
- 1959 : conception de COBOL (Common Business Oriented Language) : langage de programmation spécialisé pour la gestion et le domaine bancaire, encore utilisé de nos jours et du langage LISP (List Processing), adapté à des applications d'intelligence artificielle.
- 1960 : conception de ALGOL (ALGOrithmic Language), langage évolué de calcul scientifique

Troisième génération : les circuits intégrés

- 1962 : le terme «informatique» est créé en France par contraction de «information automatique»
- 1964 : utilisation des circuits intégrés (circuits électroniques miniatures).
- 1965 : le premier doctorat (thèse) en informatique est attribué à l'université de Pennsylvanie, conception du langage BASIC (Beginners' All-purposes Symbolic Instruction Code) et du langage PL/1 (Programming Language 1)
- 1969 : premier essai de transfert de fichier à distance par le réseau Arpanet, ancêtre d'Internet, invention du langage PASCAL par Nicklaus Wirth
- 1971 : introduction des disquettes pour l'IBM 370, conception du langage LOGO, destiné à l'initiation pédagogique aux concepts de la programmation.

Quatrième génération : les micro-ordinateurs

- 1972 : conception du langage C, particulièrement adapté à la programmation et à l'utilisation de systèmes d'exploitation.
- 1973 : apparition des premiers micro-ordinateurs munis d'un clavier et d'un écran, création de MICRAL, le premier micro-ordinateur français, et invention à Marseille du langage PROLOG (PROgrammation LOGique), par Alain Colmerauer.
- 1975 : Bill Gates commercialise le langage BASIC et crée la société Microsoft avec Paul Allen, le premier magasin spécialisé en informatique ouvre en Californie.
- 1976 : conception du langage Smalltalk, qui introduit la programmation «orientée objet».
- 1977 : création de la société Apple par Steve Jobs et Steve Wozniak et commercialisation de l'Apple II, premier micro-ordinateur largement diffusé.

Cinquième génération : l'interface graphique et les réseaux

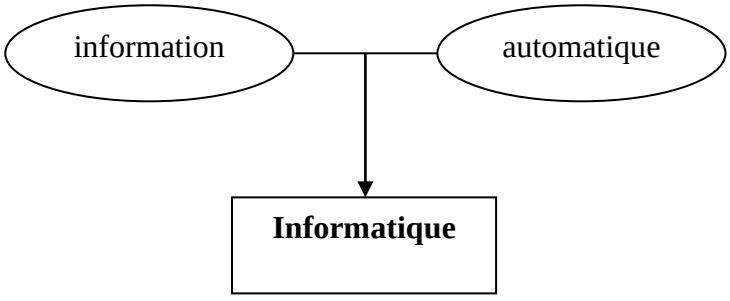
Les japonais avaient annoncé pour les années 90 l'apparition d'un nouveau type d'ordinateurs «cinquième génération» dédiés à des applications d'Intelligence Artificielle, mais ces machines d'un nouveau genre n'ont jamais vu le jour, et les évolutions majeures récentes sont plutôt à chercher du côté d'Internet.

- 1983 : conception du langage ADA (en hommage à Lady Ada Lovelace), extension du langage PASCAL, pour répondre à une demande de la NASA
- 1984 : Le McIntosh d'Apple introduit pour la première fois une interface graphique (menus, icônes...) et la souris, conception du langage C++, version orientée objet du langage C.
- 1992 : création de Mosaïc au CERN de Genève, premier navigateur permettant de visualiser des pages Web (et donc ancêtre de Netscape).
- 1995 : Windows 95 généralise l'interface graphique sur les PCs.
- …….jusqu'a ce jours.

💣 Donc, l'informatique n'est pas née d'hier, les premiers grands concepts datent de plus de 10000 ans. Cependant cette science ne s'est développée que depuis un demi siècle où la guerre a été le facteur clef de développement de l'informatique.

3. Définitions

3.1. Le mot "informatique" : il est composé des deux mots *information* et *automatique*. L'Académie française en a donné la définition suivante en 1965 : " Science du traitement rationnel, notamment à l'aide de machines automatiques, de l'information, considérée comme le support de connaissances dans les domaines scientifique, économique et social. ".



🕮 En général, l'informatique permet la manipulation, la gestion, l'organisation et le stockage de l'information.

On peut citer parmi les avantages de l'informatique :

- Calculer avec une grande précision ;
- Gain en terme de temps ;
- Aider à prendre des décisions.

3.2. Le mot "ordinateur" : l'ordinateur signifiait au départ 'calculateur numérique électronique', mais, aujourd'hui, pourrait être définit comme *'une machine de traitement de l'nformation'*.
Un ordinateur (computer) est capable de :

- Acquérir des informations ;
- Conserver des informations ;
- Effectuer des traitements sur des informations ;
- Restituer des informations.

3.3. Le mot "information" : généralement, on attribue le mot information le sens de données. L'information peut être numérique, texte, image, son, dessin, vidéo, etc…

🕮 Pour pouvoir réaliser ces différentes opérations, un ordinateur doit posséder divers organes, tels qu'un clavier pour la saisie manuelle d'informations, une mémoire, une unité centrale de traitement et une imprimante ou un écran pour sortir les résultats, etc…

4. Les aspects de l'informatique

L'informatique est représentée par deux aspects distincts mais indissociables :

4.1. Le HARDWARE : qui est composé de l'ensemble de matériels (l'ordinateur et ses périphériques).

4.2. Le SOFTWARE : pour animer un ordinateur, il faut avoir des logiciels (programmes) qui trouvent place dans deux catégories :

- **Logiciel système** : qui est un ensemble de programmes chargés d'exploiter les ressources matérielles et de définir la façon dont ces derniers doit se comporter à l'égard des commandes introduites par un utilisateur. Exemples de logiciel système : le MS-DOS et Windows etc...
- **Logiciel d'application :** qui est un ensemble de programmes destinés à réaliser des taches biens définies, tels que le traitement de texte Microsoft Word et le tableur Microsoft Excel etc…

5. La programmation

La programmation consiste à partir d'un problème donné, à réaliser un programme dont l'exécution apporte une solution satisfaisante au problème posé. Elle consiste à écrire une suite d'instructions dans un langage compréhensible par un ordinateur.

L'activité de programmation, ou plus généralement de développement de projets, se décompose en plusieurs phases qui constituent le cycle de vie du logiciel :

- Compréhension du problème ;
- Spécification des fonctionnalités du système, on dit ce qu'on veut faire mais pas comment on veut le faire ;
- Conception des algorithmes pour résoudre le problème ;
- Programmation c'est à dire faire des programmes ;
- Tests et validation des programmes ;
- Maintenance des programmes.

5.1. Un algorithme : est une succession d'actions (opérations) destinées à résoudre un problème en un nombre fini d'instructions.

5.2. Un programme : est une suite d'instructions, écrites dans un langage donné, définissant un traitement exécutable sur un ordinateur.

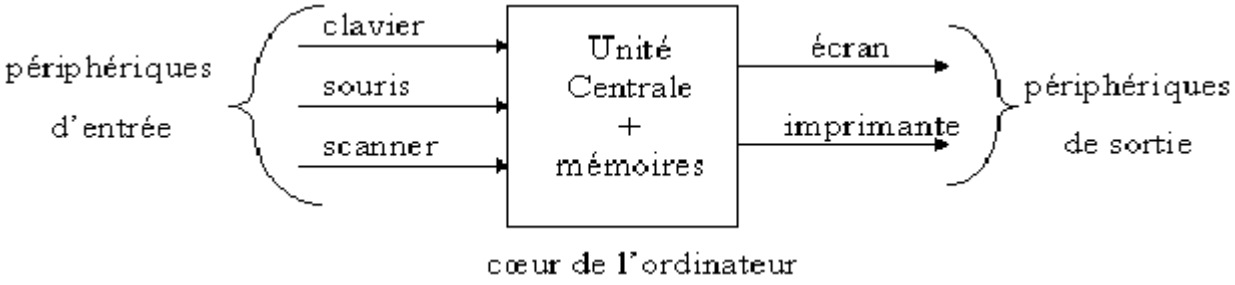
5.3. Langage de programmation : les premiers programmes étaient écrits en langage machine (code binaire pur), puis en langage assembleur qui avait l'avantage d'utiliser des mnémoniques et des symboles. Les mnémoniques permettent de remplacer des séquences de 0 et de 1 par des caractères alphabétiques ou des noms plus faciles à mémoriser. Ensuite sont apparus des langages évolués tels que Pascal, Fortran, Prolog, Delphi etc…

Un programme écrit en langage évolué se trouve sous forme de code source. Pour pouvoir être exécuté par un ordinateur, ce code source doit subir les étapes suivantes : compilation, édition de liens et chargement dans la mémoire centrale.

- Le compilateur : est un programme qui transforme un module en code source en un module en code objet (code machine) ;
- L'éditeur de liens : est un programme s'occupe de mettre ensemble les différents modules d'un programme tels que les sous programmes, procédures, fonctions, etc…
- Le chargeur : est un programme s'occupe d'amener en mémoire centrale, depuis une mémoire auxiliaire, un programme complet et prêt à être exécuté.

6. Architecture d'un ordinateur (Von Neumann)

L'architecture d'un ordinateur est la description de ses organes fonctionnels et de leurs interconnections.



Organisation générale d'un ordinateur d'après de Von Neumann

Il convient d'abord de distinguer l'ordinateur lui-même de ses «périphériques», qui ne sont que des constituants annexes. Le coeur d'un ordinateur est constitué de trois unités :

- De l'Unité Centrale (UC), ou «microprocesseur», appelé familièrement «puce» ;
- De mémoires, parmi lesquelles on distingue plusieurs types :
  - la mémoire ROM (Read Only Memory : mémoire à accès en lecture seule) : ensemble de bits dont l'état est fixé une fois pour toute, lors de la construction de l'ordinateur. Elle sert à stocker des informations permanentes (procédures de démarrage...) ;
  - la mémoire RAM ou «mémoire vive» (Random Access Memory : mémoire à accès aléatoire) : ensemble de bits modifiables à volonté, où se trouvent stockées les données sur lesquelles travaille l'ordinateur. Il ne faut pas comprendre aléatoire dans le sens de «au hasard», mais par opposition à séquentiel ; cela signifie que l'on peut

avoir accès directement à tout endroit de cette mémoire, sans avoir à la parcourir bit à bit. Cette mémoire est volatile, c'est-à-dire qu'elle ne conserve les données que tant que la machine est sous tension.

- les mémoires secondaires ou auxiliaires : ce sont des dispositifs permettant de stocker des bits de façon stable (qui reste fixée même si on éteint la machine) tout en étant généralement modifiable. On peut inclure parmi elles les disques durs, les disquettes, les bandes magnétiques.

- De l'unité d'E/S, ou unités d'échange sont des éléments qui permettent de transférer des informations entre l'unité central et les périphériques d'E/S.

Les autres composants sont donc :

- soit des *périphériques d'entrée*, c'est-à-dire permettant à un utilisateur extérieur de *fournir des informations* (données/programmes) à la machine sous forme numérique : souris, clavier, scanner... Ces dispositifs peuvent tous être conçus comme des *numériseurs* puisqu'ils transforment un comportement (l'appui sur la touche d'un clavier, le mouvement de la souris) ou un objet (une photo analogique) en une suite de bits.
- soit des *périphériques de sortie*, c'est-à-dire permettant de visualiser ou de transmettre des données internes à l'extérieur : écran, imprimante... A l'inverse des numériseurs, ces dispositifs traduisent des suites de bits en information interprétable par les humains.

Fonctionnement d'un ordinateur

1. Configuration d'un ordinateur

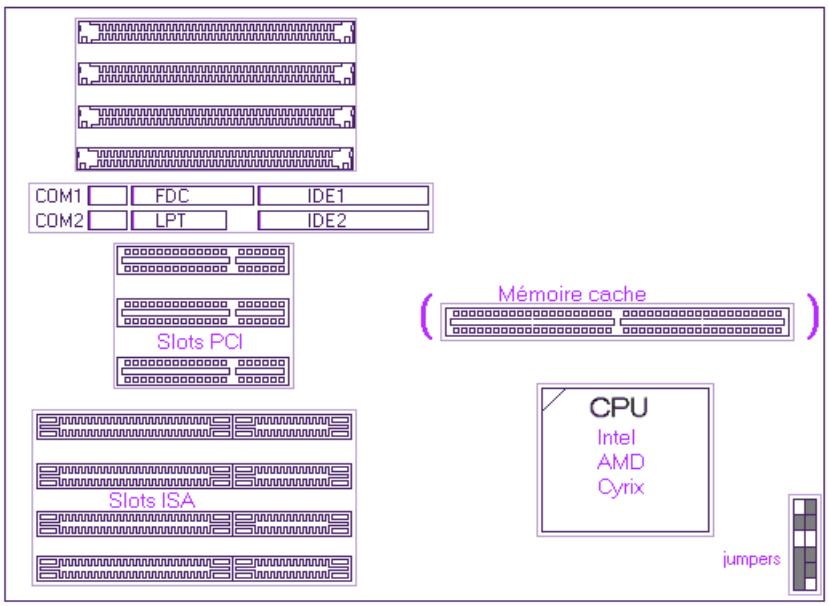
La configuration d'un ordinateur correspond à l'organisation adoptée pour mettre ensemble et faire fonctionner les divers éléments matériels de l'ordinateur. Les configurations possibles sont fonction de l'importance et de la finalité du système mis en œuvre.

Un ordinateur se décompose d'un ensemble d'éléments pouvant s'associer dans un boîtier, il est généralement composé:

- D'une unité centrale (processeur, carte mère, mémoire…);
- De périphériques internes (cartes de son, carte vidéo, ...);
- D'un lecteur de disquettes, d'un lecteur de CD-ROM ou de DVD-ROM;
- Eventuellement, de cartes d'extension diverses ;
- Périphérique d'entrée ;
- Périphérique de sortie.

1.1. La carte mère : la carte mère est le principal constituant de l'ordinateur. C'est sur cette carte que sont connectés les autres éléments :

- Le microprocesseur (cerveau de l'ordinateur);
- La mémoire (RAM : *Random Access Memory*, la mémoire cache);
- Le disque dur, le lecteur de CD-ROM, le lecteur de disquettes;
- Les périphériques internes : carte de son, carte vidéo.



- **Le microprocesseur** : le processeur (CPU) est le cerveau de l'ordinateur, c'est lui qui coordonne le reste des éléments, il se charge des calculs, bref, il exécute les instructions qui ont été programmées. Toutes ces opérations sont des informations numériques. Les microprocesseurs utilisent des petits transistors pour faire des opérations de base, il y en a plusieurs millions sur un seul processeur.

  Les principaux éléments d'un microprocesseur sont :

  - Une horloge qui rythme le processeur. À chaque TOP d'horloge, le processeur effectue une instruction. Ainsi plus l'horloge a une fréquence élevée, plus le processeur effectue d'instructions par seconde (MIPS : Millions d'instruction par seconde). Par exemple un

ordinateur ayant une fréquence de 100 mégahertz (MHz) effectue 100 000 000 d'instructions par seconde;
- Une unité de gestion des bus qui gère les flux d'informations entrant et sortant;
- Une unité d'instruction qui lit les données, les décode puis les envoie à l'unité d'exécution;
- Une unité d'exécution accomplit les tâches données par l'unité d'instruction.

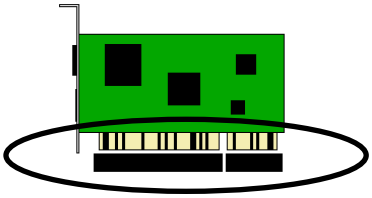
Le processeur travaille, en fait, grâce à un nombre très limité de fonctions comme des expressions logiques (ET, OU, NON, etc.), des expressions mathématiques (addition, soustraction, multiplication, etc.). Celles-ci sont directement câblées sur les circuits électroniques. Il est impossible de mettre toutes les instructions sur un processeur car celui-ci est limité par la taille de la gravure. Ainsi pour mettre plus d'instructions il faudrait un processeur ayant une très grande surface. Or le processeur est constitué de silicium et celui-ci coûte cher, et d'autre part il chauffe beaucoup. Le processeur traite donc les informations compliquées à l'aide d'instructions simples.

- **La mémoire cache** : la mémoire cache permet au processeur de se «rappeler» les opérations déjà effectuées auparavant. Elle est utilisée par le microprocesseur pour conserver temporairement des instructions élémentaires. En effet, elle stocke les opérations effectuées par le processeur, afin que celui-ci ne perde pas de temps à recalculer des calculs déjà faits précédemment. La taille de la mémoire cache est généralement de l'ordre de 512Ko ou 1Go.

- **La mémoire vive** : la mémoire vive, généralement appelée RAM (*Random Access Memory*, traduisez *mémoire à accès aléatoire*) ce qui signifie que l'on peut accéder instantanément à n'importe quelle partie de la mémoire, permet de stocker des informations pendant tout le temps de fonctionnement de l'ordinateur. Par contre, cette mémoire est détruite lors de la mise hors-tension de l'ordinateur, contrairement à une mémoire de masse comme le disque dur qui garde les informations même lorsqu'il est hors tension. La mémoire vive contient les données et les instructions des applications en cours.

- **La mémoire morte (ROM)** : mémoire permanente contenant des microprogrammes enregistrés sur des puces électroniques de la carte mère (ou *mother board*) contenant les routines de démarrage du micro-ordinateur. ROM (*Read Only Memory*, dont la traduction est *mémoire en lecture seule*) est appelée aussi parfois *mémoire non volatile*, car elle ne s'efface pas lors de la mise hors tension du système. En effet, ces informations ne peuvent être stockées sur le disque dur étant donné que les paramètres du disque (essentiels à son initialisation) font partie de ces données vitales à l'amorçage.

- **Les slots d'extension** : les slots (ou Les fentes) d'extension sont des réceptacles dans lesquels on peut enficher des cartes. Il en existe de trois types : les cartes ISA (les plus lentes fonctionnant en 16 bits), les cartes PCI (beaucoup plus rapides fonctionnant en 32 bits), et les cartes AGP (les plus rapides). Ils se branchent, grâce à des nappes, sur les broches prévues à cet effet sur la carte mère.



🕮 Les disques durs, CD-ROM et lecteurs de disquettes se branchent, grâce à des nappes, sur les broches prévues à cet effet sur la carte-mère. Il y en a en général au moins six:

- Les ports de communication (souris) se branchent sur les emplacements notés COM1, COM2 (parfois COM3 ...) ;
- Le port imprimante se branche sur l'emplacement noté LPT ;
- Le lecteur de disquette se branche sur l'emplacement noté FDC ("Floppy Disk controller" traduisez "Contrôleur de disquette") ;
- Les disques durs IDE, CD-ROM IDE se branchent par l'intermédiaire d'une nappe sur les emplacements notés IDE1 et IDE2.

- **Le disque dur** : le disque dur se présente sous la forme d'un boîtier hermétique à l'intérieur duquel se trouve une pile de plateaux ou disques magnétiques superposés et regroupés autour d'un même axe. Chaque disque possède deux faces : une face supérieure et une face inférieure. A chaque face est associée une tête de L/E fixée sur un bras mobile (deux têtes pour chaque disque).

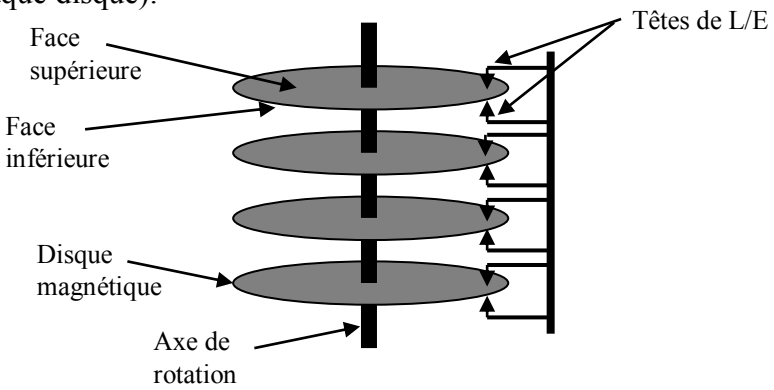
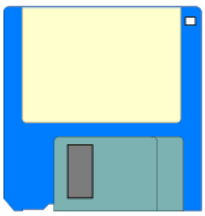


Schéma : disque dur de 4 plateaux et 8 têtes

- **La disquette** : constituée d'une rondelle de plastique souple recouverte d'un carré de plastique dur pour la protéger, la disquette à pour avantage d'être amovible, c'est-à-dire que l'on peut l'insérer et l'enlever du lecteur très facilement. Sa petite taille nous permet de la glisser dans une poche. Elle existe principalement en deux formats : 720 Ko et 1,44 Mo.



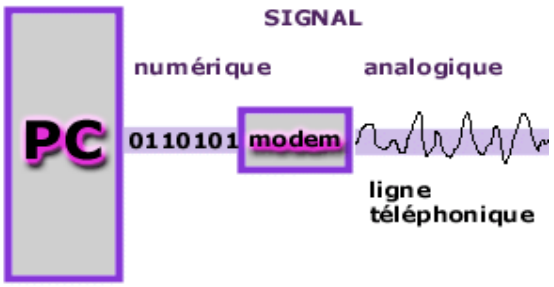
- **Le CD-ROM** : il utilise des disques portatifs de grande capacité au format pratique, de plus en plus utilisé pour la vente de logiciels. Le CD-ROM (Compact Disc - Read Only Memory) est un disque optique de 12 cm de diamètre et de 1mm d'épaisseur, permettant de stocker des informations numériques, c'est-à-dire correspondant à 650 Mo de données informatiques ou bien jusqu'à 78 min de données audio.
  Le CD est constitué de matière plastique, recouvert d'une fine pellicule métallique d'aluminium sur une des faces. Les pistes sont gravées en spirales, ce sont en fait des

alvéoles d'une profondeur de 125nm et espacées de 1,6µ. Ces alvéoles forment un code binaire, une alvéole correspond à un 0, un espace à un 1.

- **Le DVD-ROM** : le DVD-ROM (Digital Versatile Disc - Read Only Memory) est une variante du CD-ROM dont la capacité est largement plus grande. En effet, les alvéoles du DVD sont beaucoup plus petite (0,4µ et un espacement de 0.74µ), impliquant un laser avec une longueur d'onde beaucoup plus faible.
  Il existe 4 types de DVD différents :

Type de support	Capacité	Temps musical équivalent	Nombre de CD équivalent
CD	650Mo	1h18 min	1
DVD simple face simple couche	4.7Go	9 h 30	7
DVD simple face double couche	8.5Go	17 h 30	13
DVD double face simple couche	9.4Go	19 h	14
DVD double face double couche	17Go	35 h	26

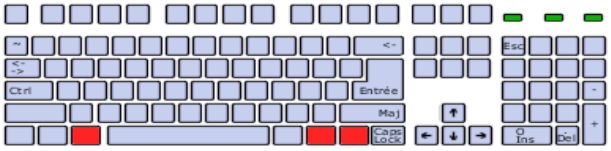
- **Le modem** : le morse a été le premier codage à permettre une communication longue distance. C'est *Samuel F.B.Morse* qui l'a mis au point en 1844. Ce code est composé de points et de tirets (un langage binaire en quelque sorte). L'interpréteur était l'homme à l'époque, il fallait toutefois une bonne connaissance du code.
  De nombreux codes furent inventés dont le code d'Émile Baudot (portant d'ailleurs le nom de code *Baudot*, les anglais l'appelaient *Murray Code*).
  Le 10 mars 1876, le Dr Graham Bell met au point le téléphone, une invention révolutionnaire qui permet de faire circuler de l'information vocale dans des lignes métalliques.
  Ces lignes permirent l'essor des télescripteurs, des machines permettant de coder et décoder des caractères grâce au code Baudot (Les caractères étaient alors codés sur 5 bits, il y avait donc 32 caractères uniquement...).
  Dans les années 60, le code ASCII (American Standard Code for Information Interchange) est adopté comme standard. Il permet le codage de caractères sur 8 bits, soit 256 caractères possibles.
  Grâce aux techniques de numérisation et de modulation autour de 1962 ainsi que de l'essor des ordinateurs et des communications, le transfert de données via modem vit le jour.
  Le modem est le périphérique utilisé pour transférer des informations entre plusieurs ordinateurs (2 à la base) via les lignes téléphoniques. Les ordinateurs fonctionnent de façon digitale, ils utilisent le langage binaire, mais les modems sont analogiques. Les signaux digitaux passent d'une valeur à une autre, il n'y a pas de milieu, de moitié, c'est du Tout Ou Rien (un ou zéro).
  Le modem convertit en analogique l'information binaire provenant de l'ordinateur. Il envoie ensuite ce nouveau code dans la ligne téléphonique. On peut entendre des bruits bizarres si on le volume du son provenant du modem.
  Ainsi, le modem module les informations numériques en ondes analogiques, en sens inverse il démodule les données numériques.
  C'est pourquoi modem est l'acronyme de MOdulateur/DEModulateur.



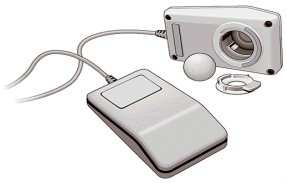
- **La carte réseau** : la carte réseau est utilisée à d'interface physique entre l'ordinateur et le câble. Elle traite les données émises par l'ordinateur, elle les transfère et contrôle le flux de données entre l'ordinateur et le câble. Elle traduit aussi les données venant du câble en octets de façon que l'Unité Centrale de l'ordinateur puisse les comprenne. Enfin, la carte réseau s'insère dans un connecteur d'extensions (slot).

1.2. Périphériques d'entrée

- **Le clavier** : le clavier est le plus important périphérique d'entrée de données. Grâce à lui, il est possible de transférer des textes ou encore de donner ordre à la machine d'effectuer des opérations particulières. De la même façon que sur une machine à écrire, le clavier permet de saisir des caractères (lettres, chiffres, symboles, ...).



- **La souris** : le déplacement de la souris permet de déplacer un curseur sur l'écran avec lequel (en cliquant sur les boutons) on peut sélectionner, déplacer, manipuler des objets à l'écran.



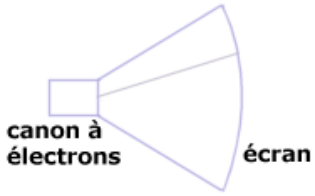
- **Le numériseur (scanner)** : Périphérique d'entrée qui permet, par balayage optique, la restitution d'une image à l'écran de l'ordinateur. Le numériseur est semblable, dans sa forme, au photocopieur, avec toutefois une importante distinction; l'image récupérée par l'appareil est transmise à l'ordinateur au lieu d'être immédiatement imprimée sur du papier.

- **La caméra numérique** : Les caméras numériques sont des appareils photographiques qui ne contiennent pas de film. Les photos sont enregistrées sur une petite disquette au lieu de s'imprégner sur une pellicule. La photographie obtenue pourra être visionnée à partir de l'écran d'un ordinateur, ou encore d'un téléviseur. Le grand avantage de ces nouveaux

appareils est leur capacité à transmettre une photo à un ordinateur, par l'intermédiaire d'un fil, pour ensuite l'intégrée à un document.

1.3. Périphériques de sortie

- **L'écran ou le moniteur** : Nous venons de voir une série de périphériques d'entrée, voyons maintenant les périphériques de sortie. Nous en retrouvons principalement deux, l'écran et l'imprimante
  L'écran, aussi appelé moniteur, affiche une image dont la netteté dépend de la résolution. Si l'image est composée de petits points, elle sera plus claire. Si les points sont plus gros, elle sera par le fait même beaucoup moins claire. Chacun de ses points s'appelle un pixel.
  Les moniteurs (écrans d'ordinateur) sont la plupart du temps des tubes cathodiques, c'est-à-dire un tube en verre dans lequel un canon à électrons émet des électrons dirigés par un champ magnétique vers un écran sur lequel il y a de petits éléments phosphorescents (luminophores) constituant des points (pixels) émettant de la lumière lorsque les électrons viennent les heurter.



- **L'imprimante :** Visionner son travail à l'écran est utile mais le résultat final doit souvent se retrouver sur du papier. Il faut alors l'imprimer.
  L'imprimante permet de faire une sortie imprimée (sur papier) des données de l'ordinateur.
  Il en existe plusieurs types d'imprimantes, dont les plus courantes sont :
    - l'imprimante laser;
    - l'imprimante à jet d'encre;
    - l'imprimante à bulles d'encre;
    - l'imprimante matricielle (à aiguilles);
    - l'imprimante à marguerite.

2. Principe de fonctionnement

Un ordinateur se compose d'une mémoire central (mémoire principal), qui contient programmes et données, d'une unité central de traitement (processeur ou CPU), qui exécute un programme chargé en mémoire centrale, et d'unité d'entrée/sortie permettant l'échange d'informations avec des unités périphériques.

L'exécution d'un programme se déroule selon le modèle suivant :

- le programme et les données sont chargés en mémoire centrale.
- Les instructions du programme sont amenées une par une, séquentiellement, à l'unité de contrôle (unité de commande) qui les analyse et déclenche le traitement approprié en envoyant des signaux à l'unité arithmétique et logique. Le passage à l'instruction suivante est automatique.
- Le traitement peut nécessiter de faire appel aux unités d'entrés/sorties ou à la mémoire centrale.

Processeur = unité centrale de traitement = unité central = CPU(central processing unit)
Mémoire central = mémoire principal
Unité d'entrée/sortie = unité d'E/S = unité d'I/O = unité d'échange
Unité de contrôle = unité de commande
Unité arithmétique et logique = UAL = unité de traitement = unité de calcul

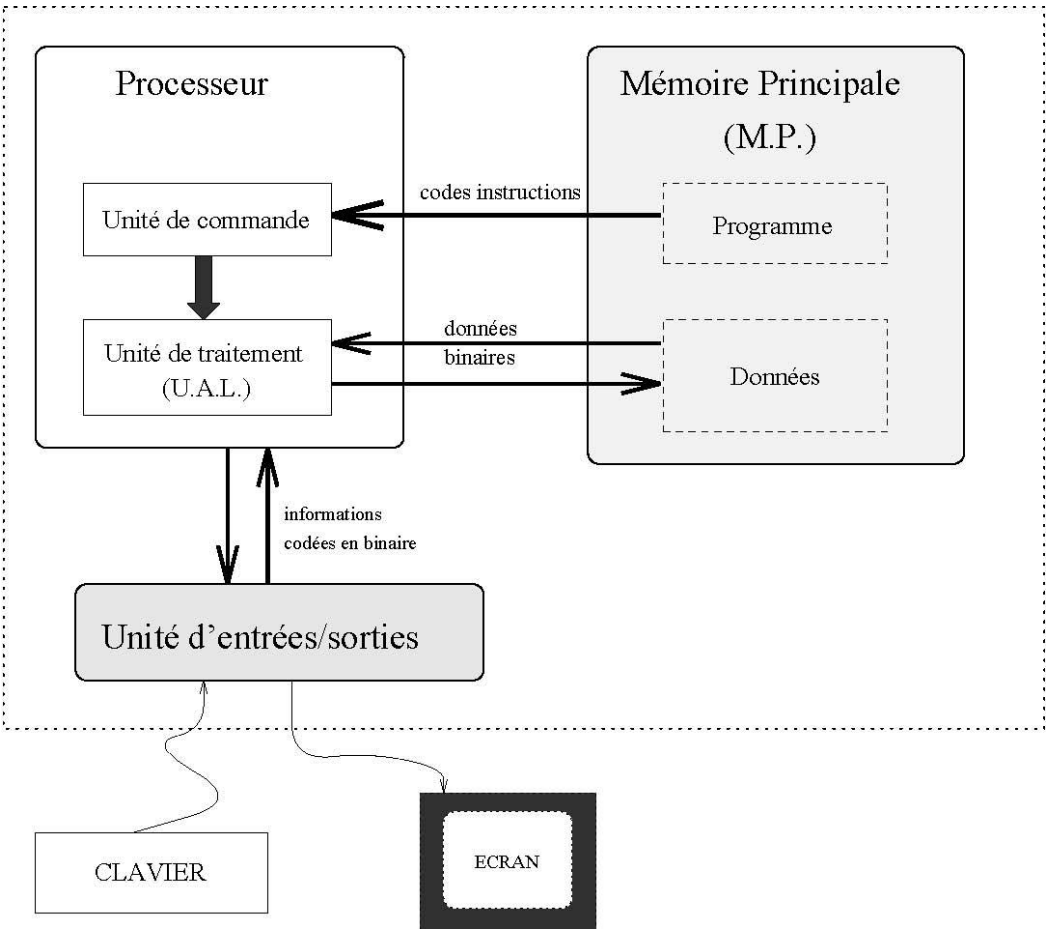


Schéma général d'un ordinateur

2.1. Unité central de traitement

L'unité centrale (CPU) est le "cerveau" de l'ordinateur. Son rôle est d'exécuter les programmes en mémoire centrale en chargeant les instructions, en les décodant et en les exécutant l'une après l'autre. L'UC est composée :

- d'une unité de commande qui charge les instructions et les décode,
- et d'une unité arithmétique et logique (UAL) qui exécute des opérations de la mémoire centrale.

- **Unité de commande :** l'unité de commande est constituée de plusieurs organes qui permettent la recherche en mémoire et le décodage d'une instruction. On trouve :
  - Le compteur ordinal qui est un registre contenant l'adresse de l'instruction à rechercher.
  - Le registre d'instruction qui reçoit l'instruction à exécuter.
  - Le décodeur de code opération qui détermine l'opération à effectuer parmi toutes celles possibles.
  - Le séquenceur qui génère les signaux de commande.
  - L'horloge qui synchronise toutes les actions de l'unité centrale.

L'Unité de commande comprend une mémoire très rapide qui lui permet de stocker des résultats temporaires ou des informations de commande. Cette mémoire est formée de quelques registres, chaque registre ayant une fonction particulière.
Le registre le plus important est le compteur ordinal (CO) qui pointe sur la prochaine instruction à exécuter. On trouve aussi le registre instruction (RI) qui contient l'instruction en cours d'exécution.
La plupart des ordinateurs contiennent également d'autres registres qui permettent aux programmeurs de stocker des résultats intermédiaires.

L'exécution d'une instruction par l'UC passe par les étapes suivantes :

1. Chargement de la prochaine instruction à exécuter depuis la mémoire jusque dans le registre instruction.
2. Modification du compteur ordinal pour qu'il pointe sur l'instruction suivante.
3. Décodage de l'instruction que l'on vient de charger.
4. Localisation dans la mémoire des éventuelles données utilisées par l'instruction.
5. Chargement des données, si nécessaire, dans les registres internes de l'unité centrale.
6. Exécution de l'instruction.
7. Stockage des résultats à leurs destinations respectives.
8. Retour à l'étape 1 pour exécuter l'instruction suivante.

- **Unité arithmétique et logique :** L'UAL réalise les opérations mathématiques et logiques nécessaires au fonctionnement de l'ordinateur et permet tout particulièrement le mode de calcul en virgule flottante. Elle possède les registres suivants :
  - Les registres arithmétiques.
  - Les registres de base et d'index (calcul d'adresse par rapport à une base ou index).
  - Les registres banalisés (ex : stockage de résultats intermédiaires).
  - Le registre d'état PSW (Program Status Word) qui indique l'état du système.

C'est l'UAL qui exécute les additions, les soustractions et toutes les opérations simples sur ses entrées, et qui produit un résultat placé dans le registre de sortie. Le contenu du registre de sortie peut alors être placé dans un autre registre avant de rejoindre, si nécessaire, la mémoire.
On peut regrouper les instructions en trois catégories :

- Registre−mémoire (2 à 3 cycles) les instructions registre−mémoire permettent de charger des mots dans des registres qui pourront, par exemple, être utilisés par d'autres instructions comme entrées de l'UAL.
- Registre−registre (1 cycle) les instructions registre−registre typiques chargent deux opérandes pris dans les registres, les placent dans les registres d'entrée de l'UAL,

exécutent sur eux une certaine opération et remettent le résultat dans un registre.

- Mémoire−mémoire (plusieurs cycles) Une instruction mémoire−mémoire prend ses opérandes dans la mémoire et les place dans les registres entrées de l'UAL, exécute ensuite une opération, et place le résultat en mémoire.

2.2. Mémoire central

La mémoire centrale contient principalement deux types d'informations :

- Les instructions de différents programmes ;
- Les données nécessaires à l'exécution des programmes.

Les instructions sont stockées sous formes de code machine. On remarque qu'au niveau physique, la mémoire centrale ne contient que des bits, qui constituent l'unité élémentaire d'information. Un bit peut prendre soit la valeur 0, soit la valeur 1. Les bits sont regroupés par 8 pour constituer un caractère. Pourquoi ce nombre de 8 bits pour coder un caractère ? Car avec 8 bits il est possible de coder 2^8 = 256 informations différentes, ce qui est suffisant pour coder tous les caractères alphanumériques et tous les caractères spéciaux.

Parallèlement aux caractères, qui constituent une unité logique d'informations, la mémoire centrale est divisée physiquement en cellules. Chaque cellule correspond à un mot-mémoire et possède une adresse qui lui est propre. Ainsi les cellules peuvent être adressées séparément pour une opération de lecture ou d'écriture. La longueur d'un mot-mémoire varie d'une machine à l'autre, par exemple : 8, 16, 32, 64 bits. La valeur 32 tend à se généraliser dans la plupart des ordinateurs.

Le mot-mémoire (word) est l'unité d'information adressable, c'est-à-dire que toute opération de lecture ou d'écriture porte sur un mot-mémoire.
A chaque mot-mémoire est donc associé :

- une adresse (unique), indiquant la position en mémoire
- un contenu (instruction ou donnée)

- la capacité d'une mémoire s'exprime en fonction du nombre de mot-mémoire ainsi que du nombre de bits par mot.

1 bit = 0 ou 1
1 octet = 1 caractère = 1 byte = 8 bits
1 Ko = 1024 octets
1Mo = 1024 Ko
1 Go = 1024 Mo

💣 Un registre est une cellule mémoire ayant une fonction particulière.

Dans la mémoire centrale on trouve deux types de registres,
- **le registre d'adresse :** qui contient l'adresse d'un mot-mémoire
- **le registre mot :** qui contient le contenu d'un mot-mémoire

💣 Un registre mot a la même taille qu'un mot-mémoire, alors qu'un registre d'adresse doit permettre d'adresser tous les mots de la mémoire.

Exemple :
Si la mémoire comporte 256 mots, le registre d'adresse doit avoir $\log_2(256) = \log_2(2^8) = 8$bits.
Un registre d'adresse de 32 bits permet d'adresser 2^8 bits permet d'adresser 2^{32} mots différents.

Les opérations possibles dans la mémoire centrale sont la lecture et l'écriture de mot-mémoire :
- **lecture :** le registre d'adresse contient l'adresse du mot à lire, le dispositif de sélection et d'accès permet de transférer le contenu de ce mot dans le registre mot.
- **Ecriture :** le registre d'adresse contient l'adresse d'un mot dans lequel on va écrire le contenu du registre mot.

Le temps nécessaire à l'écriture ou à la lecture d'un mot-mémoire est appelé le temps d'accès. Il varie entre quelques nano-secondes (1ns = 10^{-9} seconde) et quelques micro-secondes (1 μ =10^{-6}s)

2.3. Unité d'entrées/Sorties

Les unités d'E/S, ou unités d'échange sont des éléments qui permettent de transférer des informations entre l'unité central et les unités périphériques. Les unités d'E/S (I/O) les plus courantes sont :

- **le bus :** le bus est un simple câble de n lignes qui permet l'échanges et le transfert de données entre les éléments internes de l'ordinateur.
- **le DMA (Direct Memory Access):** permet à un périphérique d'accéder directement à la mémoire sans passer par le CPU, il est doté d'un registre d'adresses, d'un compteur, d'un registre de données et d'un dispositif de commande capable d'assurer le transfert.
- **le canal :** il est plus performant que le DMA, et permet à plusieurs périphériques de travailler simultanément.

Plus on s'éloigne du CPU plus les vitesses de transfert d'informations sont lentes, le CPU travaille donc plus vite que toutes les unités périphériques. Si le CPU devait lui-même s'occuper de toutes les opérations d'E/S, il passerait son temps à attendre, c'est pourquoi on utilise des processeurs spécialisés tels que les DMA et les canaux pour gérer ces E/S.

🕮 les unités périphériques se répartissent en deux classes :
- les périphériques d'entrées (clavier, sourie, etc…) ;
- les périphériques de sorties (écran, imprimante, etc…) ;
- les mémoires auxiliaires (disque dur, disquette, cassette, etc…).

💣 A chaque catégorie de périphériques est associé un contrôleur de périphériques qui s'occupe de la gestion de ces périphériques et de l'interface avec les unités d'E/S.

Le système d'exploitation et les fichiers

1. Système d'exploitation

Au cœur de l'ordinateur se trouve un ensemble de programmes que l'on nomme système d'exploitation (operating system). Ce logiciel, qui est à la base de toute exploitation de l'ordinateur, coordonne l'ensemble des taches essentielles à la bonne marche du complexe matériel et assure la gestion des ses ressources. Il facilite aussi le travail de l'utilisateur en se chargent de toutes les taches fastidieuses ou compliquées, comme le contrôle des périphériques ou le stockage et la gestion des fichiers. Il permet l'interaction directe entre l'homme et la machine, en offrant une interface convenable et en organisant le traitement et le stockage des programmes et des données.

Des exemples de systèmes d'exploitation : MS-DOS, Windows, Unix, VAX, VMS, etc…

📖 Le système d'exploitation n'existait pas dans les machines de la première génération, où toute forme de programmation était l'affaire de l'utilisateur. Aujourd'hui, le système d'exploitation est devenu l'intermédiaire obligatoire entre l'utilisateur et la machine.

1.1. Définition : le système d'exploitation est l'ensemble des programmes qui se chargent de tous les problèmes relatifs à l'exploitation de l'ordinateur.

1.2. Rôle de système d'exploitation : le système d'exploitation a deux buts biens distincts :
- faciliter la tache de l'utilisateur en lui présentant une machine (virtuelle) plus simple à exploiter que la machine réelle et en assurant un service fiable
- assurer l'exploitation efficace et économique des ressources critiques de l'ordinateur.

2. Système d'exploitation MS-DOS

Il est conçu au début des années 80. Il signifie MicroSoft Disk Opérating (Système d'exploitation sur disque de MicroSoft) et indique le nom du programme qui nous permet de faire fonctionner notre ordinateur. Il permet de réaliser les taches suivantes :

- Gérer les fichiers et les répertoires ;
- Mettre les disques à jour ;
- Configurer le matériel ;
- Optimiser la mémoire ;
- Accélérer l'exécution des programmes.

2.1. Lancement du système d'exploitation MS-DOS : lorsqu'on met notre ordinateur sous tension, plusieurs mécanismes entrent en jeu. Différents messages très obscurs défilent sous nos yeux.
A l'allumage de l'ordinateur, celui-ci effectue automatiquement certaines taches. Tout d'abord, il vérifie le matériel mis à sa disposition, en commençant par la carte vidéo et la mémoire. Sur de nombreuses machines, l'intitulé de la carte et la mémoire disponible s'affiche. Puis le décompte de la mémoire s'enclenche. Les différents périphériques sont détectés et prêt à être utilisés, grâce à un driver (pilote) spécifique.
Enfin, l'ordinateur va lire (grâce aux informations contenue dans sa RAM), les fichiers de démarrage de notre ordinateur (IO.sys et MSDOS.sys) et les fichiers de configuration de notre système (AUTOEXEC.bat et CONFIG.sys).

💣 Tout cet enchaînement s'appelle la séquence de boot.

2.2. Description de quelques fichiers spéciaux

- **Autoexec.bat :** c'est un fichier qui s'exécute automatiquement chaque fois qu'on démarre notre ordinateur. Il contient des informations qui vont régler le fonctionnement de celui-ci.

- **Config.sys :** comme Autoexec.bat, c'est un fichier qui se lance automatiquement à chaque démarrage de l'ordinateur. Il contient des informations pour le réglage des périphériques.

- **Command.com :** contient les commandes internes que MS-DOS charge en mémoire RAM lors du démarrage du système.

- **Driver :** il se nomme pilote ou gestionnaire de périphérique. Un programme un peu particulier qui va indiquer à l'ordinateur comment utiliser le matériel mis à sa disposition.

2.3. Notions sur les fichiers

- **Les fichiers :** Le fichier est un moyen d'organiser les données pour fonctionner et pour démarrer certaines applications. Ainsi, les caractères, images, et sons sont stockés sous forme de fichiers.

- **Caractéristiques des fichiers :** les fichiers sont composés de deux parties : le nom et l'extension séparées par un point « . ». les fichiers possèdent aussi une heure et date de création ainsi que des attributs.
  Par exemple : command.com

- **Opérations sur les fichiers :**
  - La création d'un fichier
  - La consultation d'un fichier
  - Mise à jour d'un fichier (modification)
  - Suppression d'un fichier

- **Les répertoires :** un répertoire sert à regrouper un ensemble de fichiers associés lorsque leur nombre devient volumineux. Lorsqu'un répertoire est volumineux, la recherche d'un fichier devient difficile. Il est alors préférable de subdiviser ce répertoire (répertoire père) en sous répertoires c'est-à-dire en répertoire à l'intérieur du répertoire pour faciliter la recherche des fichiers.

2.4. Désignation des unités de disque : Chaque unité disque est désignée par une lettre. Les différentes désignations sont :

A : pour désigner le premier lecteur de disquette
B : pour désigner le deuxième lecteur de disquette
C : pour désigner le disque dur
D, E, F, etc… : pour désigner soit le lecteur CD-ROM et les disques amovibles, soit les unités logiques (partitions) du disque dur.

2.5. Quelques commandes de MS-DOS : MS-DOS possède deux genres de commandes : les commandes internes contenues dans le fichiers COMMAND.COM (elle sont fréquemment utilisées) et les commandes externes stockées sur disque et précisément dans le répertoire DOS.

- **La commande DIR** : La commande la plus utilisée du DOS est la commande DIR. Elle permet d'afficher à l'écran la liste des fichiers d'un disque.

Pour voir une liste de fichier, tapez DIR à la suite de l'indicatif du DOS puis appuyez sur Entrée.
C:\>DIR

Pour que l'ordinateur fasse une pause pendant le défilement de la liste tapez :
C:\>DIR / P

Pour avoir seulement une liste des noms des fichiers triés en colonne, tapez :
C:\>DIR / W

Pour voir les fichiers d'une disquette, faites suivre DIR par le nom du lecteur :
C:\>DIR A:

Utilisez le joker pour chercher des fichiers précis :
Le joker * sert à masquer un groupe de caractère dans un fichier.
Par exemple pour trouver tous les fichiers commençant par n, tapez :
DIR N*
Pour avoir les fichiers exécutables, taper
DIR *.exe

- **Lire ce que contient un fichier** : La commande à utiliser pour voir ce que contient un fichier s'appelle TYPE.

C:\>TYPE NOMFICH.EXT

Par exemple pour voire le contenu du fichier lisez.moi tapez
C:\>TYPE LISEZ.MOI
Suivi de la touche Entrée.

Pour faire une pause entre chaque écran, tapez :
C:\>TYPE LISEZ.MOI | MORE
(la barre verticale s'obtient par la combinaison [ALT]+[6]).

- **L'aide :** pour obtenir de l'aide sur une commande, il faut taper le nom de celle-ci suivi de / et d'un point d'interrogation (?).

C:\>nom de commande/ ?

Pour avoir une aide plus détaillée sur toutes les commandes MS-DOS, on tape uniquement :
C:\>HELP

Ou seulement sur une commande bien précise :
C:\>HELP nom de commande
Par exemple : C:\>HELP DIR

- **Comment changer de répertoire :** Pour passer à un autre répertoire du disque, il faut utiliser la commande CD suivie du nom du répertoire.
  C:\>CD WINDOWS

Pour passer dans la racine du disque, tapez
C:\>CD \

Pour revenir au répertoire précédent, tapez CD..
C:\>CD..

- **La touche F3 :** Pour rappeler la dernière commande tapée, appuyez sur la touche F3

- **Copier un fichier :** La commande COPY sert à copier un fichier
  Elle s'utilise de cette façon :
  C:\>COPY source destination

Par exemple pour copier le fichier autoexec.bat dans le répertoire WINDOWS, tapez :
C:\>COPY AUTOEXEC.BAT C:\WINDOWS

On peut se servir du joker pour copier plusieurs fichiers en même temps.
Voir La commande DIR.

- **Supprimer un fichier :** Pour supprimer un fichier, il faut utiliser la commande DEL
  C:\>DEL source

Par exemple pour supprimer tous les fichiers commençant par un A, tapez :
C:\>DEL A*
Faites attention à ne pas supprimer n'importe quels fichiers !!

- **Renommer un fichier :** La commande REN du Dos vous permet de renommer un fichier. Le contenu et la position du fichier sur le disque ne change pas : seul son nom change.

Pour renommer le fichier AAAAA.doc en BBBBB.doc, il faut taper :
C:\>REN AAAA.doc BBBB.doc
Il faut mettre le nom actuel, suivi d'un espace et du nom futur.

Si le fichier ne se trouve pas dans le répertoire courant, il suffit de taper son chemin d'accès.
Par exemple si le fichier AAAA.doc se situe dans le répertoire C:\WORD, il faut tapez :
C:\>REN C:\WORD\AAAA.doc BBBB.doc

- **Formater une disquette :** Pour formater une disquette, il faut utiliser la commande FORMAT

Pour formater le disque A: tapez :
C:\>FORMAT A:

Pour effectuer un formatage rapide tapez :
C:\>FORMAT A:/Q

Pour faire une disquette système, tapez :
C:\>FORMAT A:/S

Vous pouvez combiner les deux: si vous voulez créer une disquette système rapidement, tapez :
C:\>FORMAT A: /Q /S

- **Dupliquer des disquettes :** La commande DISKCOPY permet de dupliquer des disquettes. Vous ne pouvez pas utiliser DISKCOPY pour dupliquer deux disques de capacités différentes.
  Vous ne pouvez pas utiliser DISKCOPY avec un disque dur.
  Il suffit de taper (si vous lecteur de disquette est A) :
  C:\>DISKCOPY A: A:

Après avoir chargé la première disquette, l'ordinateur vous demande d'insérer la disquette destination.

- **Editer un fichier :** Pour éditer un fichier quelconque, utilisez la commande EDIT. Pour cela tapez :

C:\>EDIT chemin\nom du fichier
Vous obtiendrez alors l'éditeur du DOS :
Pour éditer le fichier AAAA.doc du répertoire C:\WORD, tapez :
C:\>EDIT C:\WORD\AAAA.doc

2.6. Guide des commandes DOS

- CD : Cette commande permet de changer de répertoires.
- CLS : Cette commande efface l'écran, en retirant tous les messages encombrants. Cette commande est très utile.
- COPY : Cette commande permet de copier un fichier.
- DEL : Cette commande permet de supprimer des fichiers.
- DIR : Cette commande affiche la liste des fichiers et des répertoires présents sur le disque.
- DISKCOPY : Cette commande crée une copie exacte d'une disquette.
- FORMAT : Cette commande permet de formater des disques.
- MD : Cette commande permet de créer des répertoires.
- DELTREE : Cette commande permet de supprimer les répertoires, les sous répertoires et tous les fichiers.
- RD : Cette commande ne peut supprimer un répertoire que si ses répertoires et tous ses fichiers sont eux-mêmes supprimés au préalable.
- REN : Cette commande permet de renommer un fichier.
- TIME : Cette commande affiche la date et l'heure et vous permettez de les modifier.
- TYPE : Cette commande affiche le contenu d'un fichier à l'écran.
- TREE : Pour afficher l'arborescence d'un répertoire.

2.7. Les commandes à ne jamais utiliser : N'utilisez jamais les commandes du DOS suivantes :

- CTTY : Cette commande déconnecte le DOS du clavier et de l'écran.
- DEBUG: Il s'agit d'un utilitaire permettant de créer des programmes et de modifier la mémoire. Une mauvaise utilisation peut entraîner le bouleversement de votre disque dur.
- FDISK : Cette commande permet de créer des partitions. Mal utilisée, cette commande peut détruire toutes les informations de votre disque dur.
- FORMAT C : Cette commande formate votre disque dur, ne l'utilisez pas à moins d'être sur de ce que vous faites.
- RECOVER : Contrairement à ce qu'on pourrait croire, cette commande n'est pas un sauveteur. Cette commande détruit tous les fichiers et tous les répertoires de votre disque, et cela, sans de demande de confirmation.

3. Les fichiers

3.1. Introduction

Le fonctionnement de l'ordinateur pour sauvegarder un système d'information repose sur la notion de fichier. En effet, tout ce que traite un ordinateur ne peut l'être que sous forme de fichiers : programmes ou données.

- *Les programmes* : spécialement conçus et réalisés pour répondre au type du problème posé.
- *Les données* : relatives au problème et sur lesquelles vont agir les programmes pour aboutir aux résultats.

Ces données et programmes sont manipulés par la machine sous forme de fichiers. Chaque fichier est identifié par un nom et une extension.
Ainsi, on distingue deux types de fichiers :

- *Fichier programmes :* ce sont des fichiers qui contiennent les instructions du programme à exécuter. Ces instructions sont d'abord écrites dans un langage de programmation quelconque.
- *Fichier de données :* ce sont les fichiers qui regroupent les données qu'un programme peut éventuellement utiliser. Les fichiers de données sont évolutifs, c'est à dire qu'une donnée peut être : modifiée, supprimée, ajoutée, ou consultée.

Les fichiers qui nous intéressent ici, sont les fichiers de données structurées (des personnes, des ouvrages, des produits,…), crées par l'utilisateur pour la gestion d'une application donnée.

Exemple : pour la gestion d'une entreprise commerciale, les fichiers : clients, fournisseurs, produits, factures, commandes,…etc, forment une base de données. Tout ces fichiers sont reliés entre eux : un fournisseur fournit un produit, un client passe une commande pour l'achat d'un produit, il règle une facture,…etc. les fichiers précédents forment une base de données.

3.2. Concepts d'un fichier

- Indépendamment du support physique utilisé pour le stocker, un fichier est un ensemble d'informations de même nature qui décrivent des individus ou des objets possédant des caractéristiques communes.
- Un fichier peut être assimilé à un classeur regroupant des fiches.
- Chaque fiche contient un ensemble d'informations qui décrivent un individu (objet ou entité) bien précis. Une fiche forme un *enregistrement*.
- Dans un enregistrement, les informations sont écrites dans un ordre fixe et immuable. Chaque information est appelée *champ.*
- Chaque enregistrement est identifié de façon unique, par une information appelée *clé* ou identificateur.
- Un *fichier* est un ensemble de *champs* regroupés sous forme *d'enregistrements*, identifié par une *clé*.
- Un *fichier logique* est décrit par sa structure, c'est à dire, les différents champs qu'il regroupe. les enregistrements d'un fichier logique sont dits *articles* (ou *enregistrements logiques*).
- Un *fichier physique* est le résultat du stockage du fichier logique sur un support physique. Les enregistrements du fichier physique sont dits enregistrements physiques.
- Un enregistrement physique représente la quantité d'informations échangée entre la MC et l'unité de stockage.
- Le *facteur de blocage* représente le nombre des enregistrements logiques dans un enregistrement physique. $facteurdeblocage = \frac{nombred'enregistrementslogiques}{nombred'enregistrementsphysique}$

3.3. Exploitation d'un fichier

Par exploitation d'un fichier, on désigne la manière de retrouver l'emplacement d'un enregistrement sur un support physique. Cette exploitation se base sur la connaissance des paramètres suivants :

- ✓ Taux de remplissage
- ✓ Zone de débordement
- ✓ Lien de chaînage
- ✓ Code de validité

- **Le taux de remplissage :** désigne le rapport entre le nombre d'enregistrements effectivement stockés et le nombre d'emplacements disponibles pour le fichier sur le support.

$$\zeta_r = \frac{nombre\ d'enregistrements\ effectivements\ stockés}{nombre\ d'emplacements\ disponibles\ pour\ le\ fichier}$$

- **La zone de débordement :** est une zone spécifique sur le support, utilisée pour stocker les enregistrements qui n'ont pu être stockés dans la zone.
- **Un lien de chaînage :** est le contenu d'une zone de l'enregistrement qui indique l'adresse de l'enregistrement suivant (logiquement).
- **Le code de validité :** est le contenu d'une zone de l'enregistrement qui signale la présence ou l'absence logique d'un enregistrement (c'est la suppression logique d'enregistrements).

3.4. Caractéristiques d'utilisation des fichiers

Un fichier est crée pour être utilisé pour la gestion d'une application, il subit plus ou moins de manipulations (consultation, mise à jour, …). Ces manipulations déterminent les caractéristiques d'utilisation du fichier, à savoir :

- ✓ L'activité d'un fichier
- ✓ Le volume (la taille) d'un fichier
- ✓ L'accroissement d'un fichier

3.4.1 L'activité d'un fichier : l'activité d'un fichier caractérise l'ensemble des manipulations effectuées sur le fichier. Elle est définie par les quatre caractéristiques suivantes :

- ✓ taux de consultation
- ✓ fréquence de consultation
- ✓ taux de renouvellement
- ✓ stabilité du fichier

- **Le taux de consultation :** désigne le rapport entre le nombre d'enregistrements consultés (ou modifiés) et le nombre total d'enregistrements du fichier :

$$\zeta_c = \frac{nombre\ d'enregistrements\ consultés}{nombre\ total\ d'enregistrements}$$

- **La fréquence de consultation :** désigne une fréquence annuelle, c'est à dire le nombre d'accès à un enregistrement du fichier pour simple consultation ou mise à jour.
- **Le taux de renouvellement :** est relative à une période donnée. Il exprime le nombre relatif de nouveaux enregistrements qui sont insérés dans le fichier.
- **La stabilité du fichier :** *est relative à une période donnée.* Un fichier est dit stable pendant une période si le nombre d'enregistrements crées est approximativement égal au nombre d'enregistrements supprimés.

3.4.2. Le volume ou la taille d'un fichier : il désigne le nombre de caractères contenus dans le fichier. C'est une caractéristique très importante pour l'utilisation future du fichier (implantation physique, estimation du temps de manipulation du fichier,… .

3.4.3 L'accroissement d'un fichier : il désigne le nombre d'enregistrements crées par rapport à celui des enregistrements supprimés. Il est dit négatif lorsque le nombre d'enregistrements supprimés est supérieur au nombre d'enregistrements crées.

3.5. Typologie des fichiers5

On peut distinguer plusieurs types de fichiers selon :

- ✓ La nature des informations qu'il contient.
- ✓ La durée de vie
- ✓ Le type de support utilisé pour son stockage
- ✓ L'organisation des informations

3.5.1. Types de fichiers selon la nature des informations : un fichier peut contenir deux types d'informations : des données ou des programmes et selon le cas, on parle de fichier de données ou de fichier programme. Les données contenues dans un fichier de données peuvent être de différents types et on parle de fichier d'entiers, de réels, de caractères, d'étudiants, de client, …etc.

Exemple : les fichiers module et étudiant sont deux fichiers de données.

Le fichier calcule est un fichier programme qui permet de calculer la moyenne des étudiants.

3.5.2. Types de fichiers selon leur durée de vie : selon le rôle des informations contenues dans un fichier, leur utilité et importance, un fichier peut exister de façon permanente ou temporaire. Ainsi, on peut classifier les fichiers en quatre types :

- ✓ fichiers permanents
- ✓ fichiers mouvements
- ✓ fichiers de manœuvre
- ✓ fichiers intermédiaires

- **Un fichier permanent :** est un fichier dont les informations sont d'une importance vitale au sein de l'application pour laquelle il a été conçu. Son contenu ne subit pas de fréquentes modifications.
  Exemple : le fichier étudiants.

- **Un fichier mouvement :** sert à mettre à jour un fichier permanent. Il est caractérisé par une durée de vie courte.
  Exemple : on considère le cas d'un établissement scolaire qui gère ses étudiants à travers un fichier étudiants.
  Au début de chaque session, il y a une nouvelle section qui commence. Les nouveaux inscrits sont d'abord stockés dans un fichier inscrit, puis une fois leurs scolarités justifiées, ils sont ajoutés au fichier étudiant qui regroupe les informations concernant tous les étudiants de l'établissement, les nouveaux et les anciens.
  Dans cet exemple, le fichier inscrit est un fichier mouvement. Il sert à mettre à jour le fichier étudiant chaque fois qu'il y a de nouvelles inscriptions.

- **Un fichier de manœuvre :** trouve sa raison d'être lorsqu'il n'y a pas assez d'espace en MC pour contenir toutes les données nécessaires à un certain traitement. Sa durée de vie est limitée par celle du traitement qui l'a crée.

- **Un fichier intermédiaire :** contient des résultats d'un traitement donné pour être utilisés soit durant le même traitement, soit par d'autres traitements ultérieurs. Il permet l'échange de données entre programmes, contrairement au fichier de manœuvre qui ne communique ses données qu'au même traitement qui l'a généré. En outre la durée de vie d'un fichier intermédiaire n'est pas limitée par celle du traitement l'ayant crée, afin de permettre aux autres traitements de l'utiliser.

3.5.3. Type de fichiers selon le support utilisé : bien que le contenu d'un fichier reste le même quel que soit le support utilisé pour le stocker, certaines caractéristiques du fichier sont liées étroitement avec la nature de ce support et notamment le mode d'accès aux données qu'il contient. Par exemple, pour un fichier stocké sur une bande magnétique, seul un accès séquentiel peut être pratiqué, alors que sur un disque magnétique, l'accès technologies de fabrication utilisées pour les unités de stockage, en général.

3.5.4 Type de fichiers selon l'organisation des informations : l'organisation adoptée pour un fichier est une de ses caractéristiques les plus importantes, puisqu'elle permet de définir la manière d'accéder aux informations qu'il contient.

- ✓ Organisation séquentielle
- ✓ Organisation séquentielle indexée
- ✓ Organisation aléatoire (ou relative)

3.6. Opération sur les fichiers

Les opérations qu'on peut effectuer sur les fichiers sont les suivantes :

- Création
- Suppression
- Tri
- Fusion
- Eclatement

3.7. Opération sur les enregistrements

Apres la création du fichier, plusieurs opérations peuvent être effectuée sur ses enregistrements:

- *La création :* qui consiste la saisie d'un enregistrement
- *La suppression :* qui revient à effacer un ou plusieurs enregistrements. Si tous les enregistrements sont supprimés, on obtient un fichier vide.
- *La modification :* consiste à changer les valeurs d'un ou de plusieurs champs.
- *La consultation :* qui revient à lire la valeur d'un ou de plusieurs champs d'un enregistrement.

3.8. Les fichiers et le système d'exploitation

Pour exploiter facilement les fichiers (création, suppression, …), le système d'exploitation offre à l'utilisateur un certain nombre d'outils regroupés sous le nom de *système de gestion de fichiers (SGF)*. Le SGF joue le rôle d'intermédiaire entre l'utilisateur et les fichiers stockés sur le support correspondant.

En outre, le SGF permet de protéger les fichiers (contre l'écriture, la lecture, …), de les partager avec d'autres, de bien gérer l'espace de stockage,…etc.

3.9. Organisation des fichiers

Le but visé par l'organisation est de ranger les enregistrements dans un ordre bien déterminé de manière à :

- Accéder le plus rapidement possible aux enregistrements du fichier.
- Occuper le moins de place possible sur le support et ce afin d'éviter les pertes d'espace sur le support et faciliter l'insertion de nouveaux éléments.

L'organisation d'un fichier est choisie lors de sa création et elle peut être :

- Séquentielle
- Séquentielle indexée
- Aléatoire

3.9.1 Organisation séquentielle : dans une organisation séquentielle, les articles sont enregistrés dans l'ordre où ils se présentent à la saisie. Ils sont placés les uns à la suite des autres. Ainsi, un nouvel enregistrement saisi est toujours écrit à la fin du fichier.

Pour rechercher un $i^{ème}$ enregistrement, il faut d'abord parcourir les (i-1) enregistrements qui le précédent.

Caractéristiques

- les enregistrements sont écrits selon l'ordre de leur arrivée.
- l'insertion de nouveaux enregistrements se fait uniquement en fin de fichier.
- Chaque enregistrement possède un prédécesseur (sauf le premier) et un successeur (sauf le dernier).

Avantages

- Simple à appliquer
- Facile à implémenter
- Implémenter sur n'importe quel type de support (bande magnétique, disque magnétique,…..).
- Economique en espace mémoire.

Inconvénients

- elle n'est pas pratique pour les fichiers de grande taille, car elle nécessite des temps d'accès très longs.
- Pour insérer de nouveaux enregistrements au milieu du fichier, il faut copier le fichier intégralement. En effet, il faudra copier une partie du fichier jusqu'à la position où devra se faire l'insertion, enregistrer le nouvel enregistrement, puis copier la deuxième partie du fichier.

3.9.2. Organisation séquentielle indexée : comme pour l'organisation séquentielle, les enregistrements sont écrits dans l'ordre de leur arrivée. Seulement, ici à chaque fois qu'un enregistrement est écrit, une clé lui est associée. Cette clé, ainsi que l'adresse relative de l'enregistrement dans le fichier sont inscrites dans une table appelée *table d'index*.

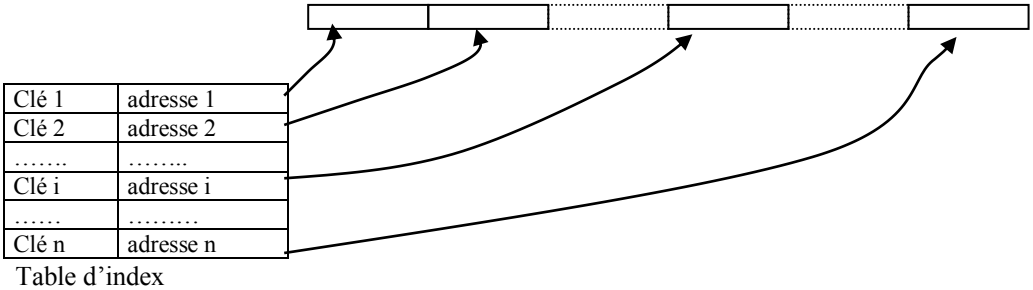


Fig : Fichier organisé en séquentiel indexé

🕮 L'insertion d'un nouvel article se fait à la fin du fichier. A chaque insertion, la clé de l'article inséré, ainsi que son adresse dans le fichier sont ajoutés à la table d'index.

🕮 A l'ouverture du fichier, la table d'index est chargée en mémoire. Pour accéder à un enregistrement, il suffit d'utiliser sa clé pour retrouver son adresse dans la table d'index.

Caractéristiques

- Les articles sont écrits dans l'ordre de leur arrivée.
- Une table d'index est associée au fichier. Elle contient les clés des articles et leurs adresses relatives dans le fichier.
- La recherche d'un article est facilitée par la table d'index. Il suffit de connaître la clé pour connaître l'adresse à laquelle l'article est stocké.

Avantages

- 👍 La table d'index permet de retrouver plus facilement les articles dans le fichier.
- 👍 La table d'index permet d'accéder rapidement et directement (si le support de stockage le permet) aux articles.

Inconvénients

Quand le fichier est trop grand, la table d'index devient de taille importante, ce qui génère deux inconvénients :

- 👎 La table occupe un espace mémoire non négligeable.
- 👎 La manipulation de la table devient lourde.

3.9.3. Organisation aléatoire : ce mode d'organisation repose sur un principe simple :

Le fichier est organisé en pages (des pages principales et des pages de débordement). Dans une même page, les enregistrements du fichier sont écrits selon leur ordre d'arrivée. Pour déterminer le numéro de la page où devrait être écrit un enregistrement donné, un calcul est effectué sur sa clé. Ce calcul est réalisé à partir d'une fonction appelée fonction de répartition (ou fonction de randomisation).

Si la page devant contenir un nouvel enregistrement est saturée, elle sera chaînée à la page de débordement où sera écrit ce nouvel enregistrement. En général, les pages de débordement sont gérées de la même manière que les pages principales, c'est à dire qu'une fonction de répartition est appliquée pour déterminer le numéro de la page de débordement où sera écrit l'enregistrement qui a causé le débordement.

Exemple : à créer le fichier étudiant avec les enregistrements dont les clés sont : 6 , 8 , 10 , 13 , 18 , 9 , 20 , 2 et 29.

Le fichier étudiant est un fichier aléatoire organisé en 9 pages principales et 4 pages de débordement. La fonction de répartition (pour le calcul des numéros de pages) est définie comme suit :

N=clé modulo [nombre de pages principales], c'est-à-dire que le numéro de la page N est le reste de la division de la clé sur le nombre de pages principales.

Ainsi, pour écrire les enregistrements précédents, on doit appliquer cette fonction :

N_1 = 6 modulo [9] = 6 donc, 6 sera écrit dans la page numéro 6 (les pages sont numérotées à partir de 0).
N_2 = 8 modulo [9] = 8 ;
N_3 = 10 modulo [9] = 1 ;
N_4 = 13 modulo [9] = 4 ;
N_5 = 18 modulo [9] = 0 ;
N_6 = 9 modulo [9] = 0 ;
N_7 = 20 modulo [9] = 2 ;
N_8 = 2 modulo [9] = 2 ;
N_9 = 29 modulo [9] = 2 ;

On remarque, sur le schéma, que la page numéro 2 est saturée. Essayons d'insérer l'enregistrement 47.
Nous avons : 47 modulo [9] = 2, puisqu'il n'y a plus de place dans la page 2, ce nouvel enregistrement sera écrit dans une page de débordement. Pour déterminer le numéro de ctte page de débordement, on utilise toujours la fonction de répartition précédente :
Nombre de pages de débordement = 4.
D'où N = 47 modulo [4] = 3.
L'enregistrement sera écrit dans la page de débordement numéro 3. Un lien de chaînage va lier la page principale numéro 2 à l'enregistrement en page de débordement.

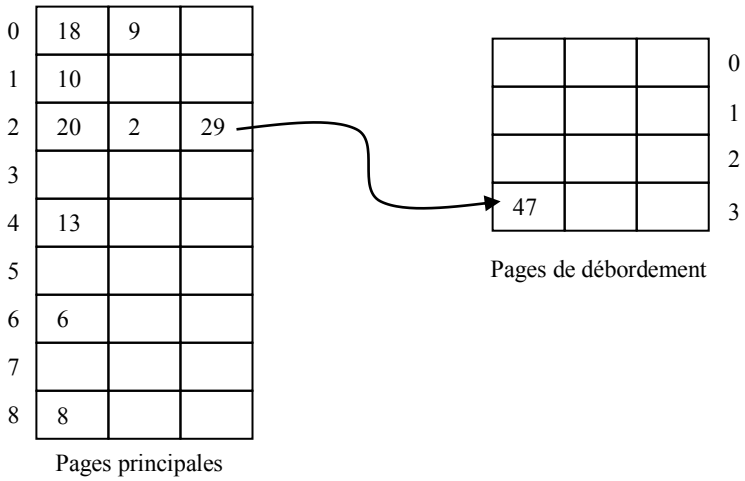


Schéma : le fichier étudiant organisé en aléatoire

- Pour obtenir une répartition équilibrée des enregistrements sur les pages et éviter des débordements fréquents, il faut bien choisir les paramètres suivants et ce dés la création du fichier :
  - Taille d'une page
  - Nombre de pages
  - Fonction de répartition

Avantages

- Le principe du calcul de l'adresse d'un enregistrement (le numéro de page) facilite son insertion dans le fichier.
- L'organisation aléatoire permet d'avoir bons temps de réponse du fait que l'accès aux enregistrements se fait par calcul d'adresse.

Inconvénients

- Elle ne permet pas le traitement efficace du fichier en séquentiel, chose due au débordement des pages principales.
- En cas d'un nombre important de débordement, une réorganisation du fichier s'impose.

3.10. Choix d'une organisation

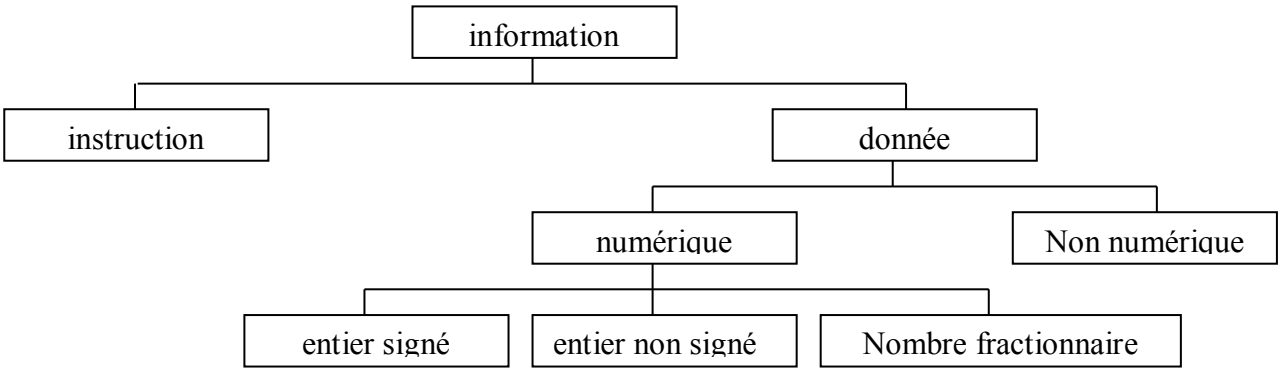
L'organisation d'un fichier se choisit de manière à satisfaire les objectifs suivants :

- Permettre un accès rapide aux enregistrements du fichier
- Assurer un gain en place sur le support de stockage
- Faciliter l'insertion de nouveaux enregistrements.

Représentation des informations en mémoire

Les informations que doit traiter l'ordinateur sont constituées de caractères ou de symboles appartenant à un alphabet. Cet alphabet comprend des chiffres (0,1,…,9), des lettres (a,b,…,z) et des caractères spéciaux (+,-,*, ?,…).
Du fait de sa structure électronique, l'ordinateur ne peut assimiler que deux états 0 et 1, c'est pourquoi, toutes les informations qui y circulent doivent être exprimées dans le système binaire.

1. Typologie de l'information



Les informations traitées par la machine sont deux types :
- des instructions
- des données

Une instruction est codée en binaire et contient, selon le type de machines, un ou plusieurs champs définissant le code opération et les opérandes sur lesquels va porter l'opération.
Les données sur lesquelles porte une instruction peuvent être de différents types :
- de données numériques (chiffres)
- données non numériques (non chiffres)

Pour les représenter dans un format assimilable par la machine, ces données doivent être traitées séparément, car chaque type de données présente des caractéristiques propres à elles.

2. Représentation des données non numériques

Les données non numériques sont exprimées sous la forme de texte, caractères alphanumériques, caractères spéciaux, etc…
Pour les introduire dans la machine, il faut d'abord les coder sous forme binaire. Plusieurs codes ont été conçus pour réaliser cette opération de codage se basant tous sur une table de correspondance qui permet de trouver le code associé à un caractère donnée et inversement, le caractère correspondant à un code donnée.
Selon le nombre de bits associé au code, on distingue :
- le code BCD (code à 6 bits)
- le code ASCII (code à 7 bits)
- le code EBCDIC (code à 8 bits)

Le tableau suivant illustre quelques exemples de codes :

Caractère	Code BCD	Code ASCII	Code EBCDIC
0	000 000	011 0000	1111 0000
1	000 001	011 0001	1111 0001
2	000 010	011 0010	1111 0010
…	…….	……...	…….
A	010 001	100 0001	1100 0001
B	010 010	100 0010	1100 0010
…	……	…….	…….

3. Représentation des données numériques

Les données numériques regroupent les nombres suivants :

- entiers non signés (0 , 1 , 2 , ….)
- entiers signés (-2 , -1 , 0 , +1 , +2 , ….)
- nombres fractionnaires (-0.5 , +5.13 , ….)

Représentation des entiers non signés : Les entiers positifs ou nuls peuvent être codés en binaire pur. A l'aide de n bits nous pouvons représenter 2^n nombres compris entre 0 et 2^n-1

$$A = \sum_{k=0}^{n-1} a_k \, 2^k \ .$$

3.1.1. Les systèmes de numération : un système de numération se définit par deux éléments :

- la base du système
- les symboles du système

Les systèmes les plus utilisés sont les suivantes :

Système	Base	Symbole	Nombre de symbole
Décimal	10	0,1,2,3,4,5,6,7,8,9	10
Binaire	2	0,1	2
Octal	8	0,1,2,3,4,5,6,7	8
Hexadécimal	16	0,1,2,…,9,A,B,C,D,E,F	16

Notation : soit N un nombre quelconque exprimé dans une base b. le nombre N sera noté comme suit :

$N = (a_n \, a_{n-1} \, a_{n-2} \, \ldots \, a_0)$ telque : b : base du système de numération

a_i : symbole du système i = 0, … ,n avec $a_i < b$

Exemple :

$N1 = (19017)_{10}$ (en décimal, avec : $a_4=1$, $a_3=9$, $a_2=0$, $a_1=1$, $a_0=7$)

$N2 = (1011101)_2$ (en binaire, avec : $a_6=1$, $a_5=0$, $a_4=1$, $a_3=1$, $a_2=1$, $a_1=0$, $a_0=1$)

$N3 = (1370)_8$ (en octal, avec : $a_3=1$, $a_2=3$, $a_1=7$, $a_0=0$)

$N4 = (A9120)_{16}$ (en hexadécimal avec : $a_4=A$, $a_3=9$, $a_2=1$, $a_1=2$, $a_0=0$)

Rang et poids d'un chiffre : soit le nombre N exprimé dans une base b, comme suit : $N=(a_n a_{n-1} a_{n-2} \ldots a_0)_b$

On appelle rang d'un chiffre sa position i dans le nombre à partir de la droite. Ainsi :

- le rang du chiffre a_n est n
- le rang du chiffre a_1 est 1
- le rang du chiffre a_0 est 0

On appelle poids d'un chiffre le nombre b^i telque b est la base et i le rang du chiffre ainsi :

- le poids du chiffre a_n est b^n
- le poids du chiffre a_1 est b^1
- le poids du chiffre a_0 est b^0

Le poids faible (ou b^0) est celui du chiffre à l'extrême droite. a_0 est le chiffre de poids faible. Le poids fort est celui du premier chiffre du nombre à partir de la gauche. a_n est chiffre de poids fort.

Exemple :
On considère le nombre N exprimé en octal : N=$(13042)_8$
- le rang du chiffre 2 est 0
- le rang du chiffre 4 est 1
- le rang du chiffre 0 est 2
- le rang du chiffre 3 est 3
- le rang du chiffre 1 est 4

Les poids de ces chiffres sont :
- le poids du chiffre 2 est 8^0=1
- le poids du chiffre 4 est 8^1=8
- le poids du chiffre 0 est 8^2=64
- le poids du chiffre 3 est 8^3=512
- le poids du chiffre 1 est 8^4=4096

Le chiffre du poids faible est 2, son poids est 1
Le chiffre du poids fort est 1, son poids est 4096

- il est possible d'exprimer n'importe quel nombre dans n'importe quelle base. Pour cela, il suffit de connaître les méthodes de conversion.

3.1.2. Les méthodes de conversion

Passage de la base b à la base 10
Pour avoir la représentation en décimal du nombre N exprimé dans une base b quelconque, il suffit d'effectuer le calcul suivant :
$(N)_{10} = a_n*b^n + a_{n-1}*b^{n-1} + \ldots + a_1*b^1 + a_0*b^0$

La formule général s'écrit comme suit: $(N)_{10} = \sum_{i=0}^{n} a_i b^i$, i étant le rang du chiffre a_i

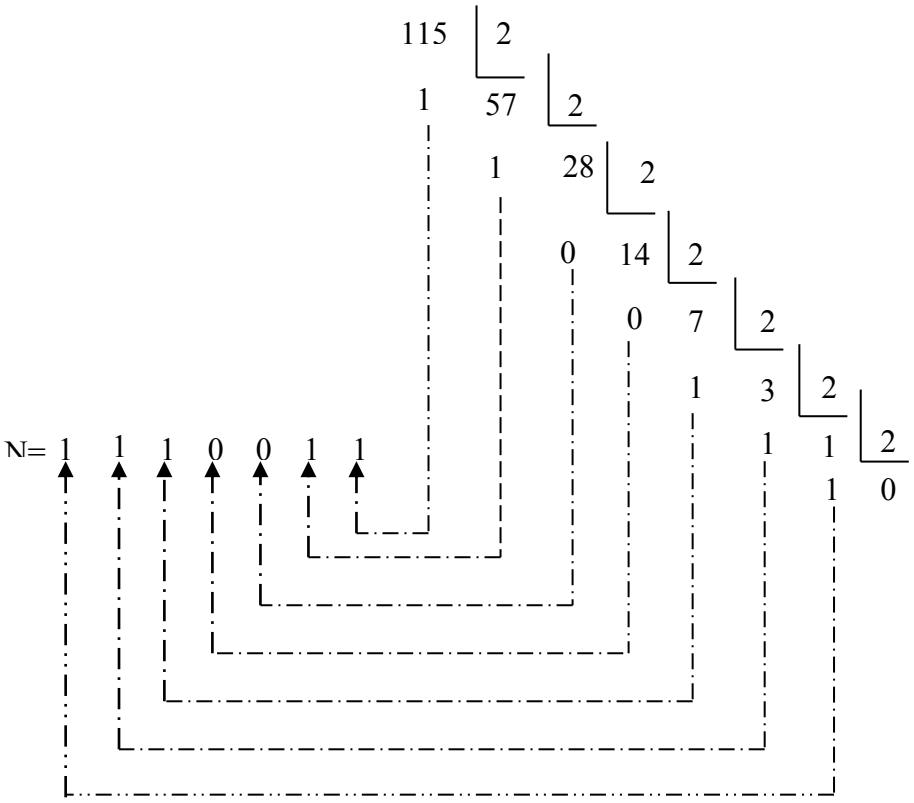
Exemple :
$N1 = (1023)_4 = 1*4^3 + 0*4^2 + 2*4^1 + 3*4^0 = 75$ donc : $N = (1023)_4 = (75)_{10} = 75$
$N2 = (10111001)_2 = 1*2^7 + 0*2^6 + 1*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = (185)_{10} = 185$
$N3 = (175)_8 = 1*8^2 + 7*8^1 + 5*8^0 = (125)_{10} = 125$
$N4 = (A24)_{16} = (A24)_H = A*16^2 + 2*16^1 + 4*16^0 = 10*16^2 + 2*16^1 + 4*16^0 = (2596)_{10} = 2596$

Passage de la base 10 à la base b

Soit N un nombre exprimer dans la base 10. Pour l'exprimer dans une autre base b, il suffit d'effectuer des divisions successives sur b jusqu'à l'obtention d'un résultat nul.

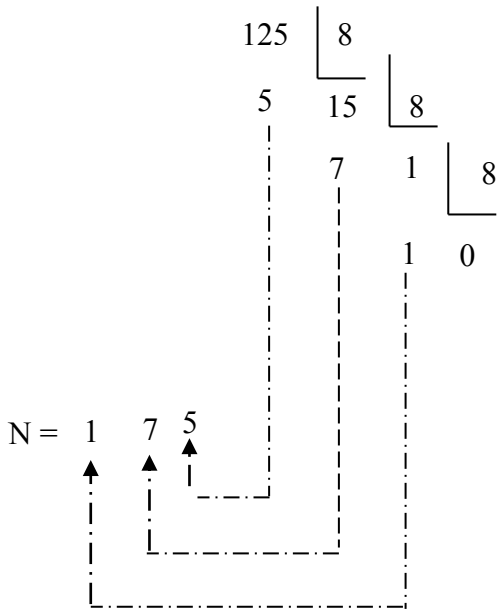
Exemple :

- Soit le nombre N1=$(115)_{10}$, convertir ce nombre en binaire :



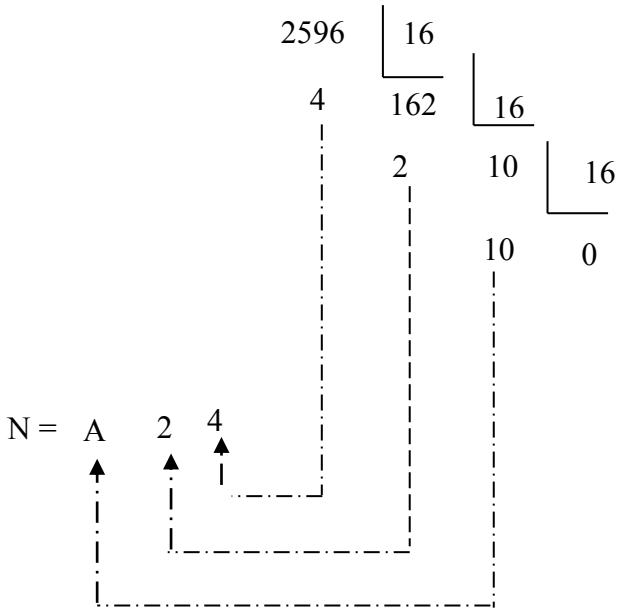
Donc N1 = $(115)_{10}$ = $(1110011)_2$

- Soit le nombre N2 = $(125)_{10}$, convertir ce nombre en octal :



Donc N2 = $(125)_{10}$ = $(175)_8$

- Soit le nombre N3 = $(2596)_{10}$, convertir ce nombre en hexadécimal :



Donc N3 = $(2596)_{10} = (A24)_H$

Passage d'une base p à une base q

Pour convertir un nombre de la base p en un nombre de la base q, il faut passer par une base intermédiaire qui est la base 10.

Exemple :

- Soit le nombre N1 = $(175)_8$ = (?)$_2$

Passage de l'octal à la base 10 : N1 = $(175)_8 = (1*8^2 + 7*8^1 + 5*8^0) = (125)_{10}$

Passage du décimal au binaire : $(125)_{10} = (1111101)_2$

Donc, $(175)_8 = (1111101)_2$

- soit le nombre N2 = $(A24)_H$ = (?)$_2$

Passage de l'hexadécimal à la base 10 : N2 = $(A24)_H = A*16^2 + 2*16^1 + 4*16^0 = 2596$

Passage du décimal au binaire : $(2596)_{10} = (101000100100)_2$

Donc, $(A24)_H = (101000100100)_2$

💣 Lorsqu'une base est une puissance d'une autre base, la passage de l'une à l'autre devient très facile et ne nécessite pas une base intermédiaire.

🕮 Le passage de la base 8 (2^3) ou 16 (2^4) à la base 2 peut s'effectuer sans passer par la base 10.

Exemple :

i. Passage de l'octal au binaire :

La base 8 est une puissance de la base 2. pour convertir un nombre octal en binaire, on procède comme suit :

On a : $8=2^3$ cela veut dire que pour représenter un seul chiffre octal en binaire, il faut utiliser 3 bits.

Ainsi, la représentation des chiffres de la base 8 en binaire est la suivante :

Chiffre octal	Chiffre binaire équivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

- Soit le nombre octal N1= $(175)_8$ = (?)$_2$

Pour trouver l'équivalent binaire de ce nombre octal, il suffit de trouver l'équivalent binaire de chaque chiffre octal.

1			7			5		
0	0	1	1	1	1	1	0	1

Donc, N1 = $(175)_8$ = $(001111101)_2$ = $(1111101)_2$ (les deux 0 sur la gauche sont superflus).

- Soit le nombre binaire N2 = $(1111101)_2$ = (?)$_8$

Pour trouver l'équivalent octal de ce nombre binaire, il faut :

Regrouper les bits du nombre binaire en groupes de 3 bits en partant de la droite. si le dernier groupe ne contient pas trois bits, on ajoute des 0.

0	0	1	1	1	1	1	0	1
1			7			5		

Donc, N2 = $(1111101)_2$ = $(175)_8$

ii. Passage de l'hexadécimal au binaire :
La base 16 est une puissance de la base 2 ($16 = 2^4$). Pour convertir un nombre hexadécimal en binaire, on procède comme suit :

On a : $16=2^4$ cela veut dire que pour représenter un seul chiffre hexadécimal en binaire, il faut utiliser 4 bits.

Chiffre hexadécimal	Chiffre binaire équivalent
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

- On considère le nombre $N1 = (A24)_H = (\ ?)_2$
On va représenter chaque chiffre hexadécimal par son équivalent binaire :

A				2				4			
1	0	1	0	0	0	1	0	0	1	0	0

Donc, $N1 = (A24)_{16} = (101000100100)_2$

- On considère le nombre $N2 = (101000100100)_2 = (\ ?)_H$
On regroupe les bits 4 à 4, en commencent par la droite. Si le dernier groupe n'a pas 4 bits, on ajoute des 0.

1	0	1	0	0	0	1	0	0	1	0	0
A				2				4			

Donc, $N2 = (101000100100)_2 = (A24)_{16}$

3.1.3. Les opérations arithmétiques binaires : Le système binaire permet d'effectuer les 4 opérations arithmétiques élémentaires :
- l'addition
- la soustraction
- la multiplication
- la division.

L'addition :
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 avec une retenue r = 1.

Exemple :
Soit à effectuer l'addition binaire suivante : 11011 + 10011

```
débordement ⟶1      1 1
                1 1 0 1 1
          +
                1 0 0 1 1
            ___________
              0 1 1 1 0
```

📖 dans un ordinateur, la taille des registres est fixe, ce qui impose une limite aux nombres représentés. Il arrive que cela génère des anomalies lors d'une opération arithmétique sur ces nombres. Ainsi, lorsqu'il y a une retenue qui est générée par le bit le plus à gauche, on dit qu'il y a débordement ou un dépassement de capacité.

La soustraction
0 – 0 = 0
1 – 0 = 1
1 – 1 = 0
0 – 1 = 1 avec une retenue r = 1

Exemple :
Soit à effectuer l'opération 1110 – 10011

```
                0 1 1 1 0
          -
                1 0 0 1 1
            ___________
              1 1 0 1 1
```

La multiplication
0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1

Exemple :
Soit à effectuer l'opération 1101 * 101

```
                  1 1 0 1
            *
                    1 0 1
              _____________
                1 1 0 1
              0 0 0 0 .
            1 1 0 1 .
            _____________
            1 0 0 0 0 0 1
```

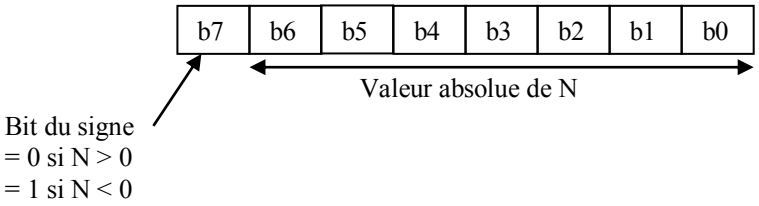
3.2. Représentation des entiers signés : Il y a un sérieux problème pour la représentation des entiers signés, comment différencier entre les nombres positifs et les nombres négatifs ?
Des solutions on été proposées, parmi lesquelles on cite :

- représentation en signe + valeur absolue
- représentation en complément à 1
- représentation en complément à 2

3.2.1. Représentation signe+valeur absolue : en représentation signe + valeur absolue, un nombre est représenté sur 4, 8 ou 16 bits telque :

- le bit le plus à gauche désigne son signe :
  - 0 pour un nombre positif
  - 1 pour un nombre négatif
- les bits restent désignent la valeur absolue du nombre.

Soit N un nombre quelconque à représenter sur 8 bits.
En représentation signe+valeur absolue, N sera représenté comme suit :



Exemple :
Le tableau suivant donne la représentation signe + valeur absolue pour les nombres compris entre +7 et -7 représentés sur 4 bits (1bits pour le signe et 3bits pour la valeur absolue) :

N	Signe+valeur absolue
+7	0 111
+6	0 110
+5	0 101
+4	0 100
+3	0 011
+2	0 010
+1	0 001
+0	0 000
-0	1 000
-1	1 001
-2	1 010
-3	1 011
-4	1 100
-5	1 101
-6	1 110
-7	1 111

Cependant, la représentation signe + valeur absolue pose trois types de problèmes :

- dans le tableau, on remarque que le chiffre 0 possède deux représentations : 1000 et 0000
- le fait de réserver un bit pour le signe réduit la valeur maximale du nombre à représenter.
- Pour effectuer une opération (+ , - , * , /) avec les entiers signés, le problème pose au niveau du bit de signe. Le bit du signe est traité séparément, mais le résultat n'est pas toujours correct.

Exemple :
Soit à effectuer les opérations : (-8)+(+6) et (+8)+(-6) :
$(-8) = (1\ 1000)_2$
$(-6) = (1\ 0110)_2$
$(+8) = (0\ 1000)_2$
$(+6) = (0\ 0110)_2$

```
   +8              0 1000
 +                +
   -6              1 0110
 ------          --------
   +2            ? 1110   = ?14 ≠ +2
                 ↗
        Le bit du signe
        est traité à part
```

```
   -8              1 1000
 +                +
   +6              0 0110
 ------          --------
   -2            ? 1110   = ?14 ≠ -2
                 ↗
        Le bit du signe
        est traité à part
```

Conclusion : l'addition ne fonctionne pas correctement avec la représentation signe + valeur absolue, en plus il faut traiter le bit du signe séparément.

3.2.2. Représentation en complément à 1 (complément logique) : dans cette représentation, les nombres positifs gardent le format binaire. Les nombres négatifs sont complémentés :

- les 0 sont transformés en 1
- les 1 sont transformés en 0

Le tableau suivant donne le complément à 1 des nombres compris entre -7 et +7 :

N	Signe+valeur absolue	Complément à 1
+7	0 111	0 111
+6	0 110	0 110
+5	0 101	0 101
+4	0 100	0 100
+3	0 011	0 011
+2	0 010	0 010
+1	0 001	0 001
+0	0 000	0 000
-0	1 000	1 111
-1	1 001	1 110
-2	1 010	1 101
-3	1 011	1 100
-4	1 100	1 011
-5	1 101	1 010
-6	1 110	1 001
-7	1 111	1 000

Toujours le même problème par exemple :
On va effectuer les mêmes opérations que précédemment, mais en complément à 1 :
(+8) : complément à 1 = 0 1000
(-8) : complément à 1 = 1 0111
(+6) : complément à 1 = 0 0110
(-6) : complément à 1 = 1 1001

On va effectuer l'opération (-8) + (-6)

```
   -8              1 0111
 +               +
   -6              1 1001
 ------          ---------
  -14            1 1 0000
                +       1
                ---------
                  1 0001  ⟶ Le complément à 1 de (1 0001) est (1 1110)2 = (-14)10
```

📖 la retenue générée par le bit de signe (le bit le plus à gauche) va etre ajoutée au résultat de l'opération et on obtient le nombre (1 0001) en complément à 1 qui est un nombre négatif correspondant au nombre (1 1110).

On va effectuer l'opération (-8) + (+6)

```
   -8              1 0111
 +               +
   +6              0 0110
 ------          ---------
  -2             1 1101  ⟶ Le complément à 1 de (1 1101) est (1 0010)2 = (-2)10
```

📖 dans se cas, il n'y a pas de retenue générée par le bit de signe, le résultat étant négatif (1 1101), son complément à 1 (1 0010) qui correspond au nombre (-2).

On va effectuer l'opération (+8)+(-6) :

```
   -8              0 1000
 +               +
   -6              1 1001
 ------          ---------
  -14            1 0 0001
                +       1
                ---------
                  0 0010  ⟶ Le complément à 1 de (0 0010) est (0 0010)2 = (+2)10
```

📖 la retenue générée par le bit le plue à gauche (bit du signe), elle sera additionnée au résultat de l'opération et on obtient un nombre positif $(0\ 0010)_2 = (+2)_{10}$.

Conclusion : l'addition en complément à 1 se base sur le principe suivant :
Si aucune retenue n'est générée par le bit de signe, le résultat est correct, il est exprimé en complément à 1.
Si au contraire, il y a une retenue qui est générée par le bit de signe, elle sera additionnée au résultat de l'opération, celui-ci est exprimé en complément à 1.

3.2.3. Représentation en complément à 2 (complément arithmétique) : en complément à 2, les nombres positifs ont exactement la même représentation en binaire non signée et en complément à 1. Les nombres négatifs par contre, sont obtenus en calculant d'abord le complément à 1, puis lui ajouter un 1.

Exemple :
N = -17
On a : $+17 = (00010001)_2$
Le complément à 1 de (-17) est le suivant : 11101110
Le complément à 2 de (-17) est donc : complément à 1 + 1 = 11101110 + 1 = 11101111
Le tableau suivant donne le complément à 2 des nombres compris entre -8 et +7 :

N	Signe+valeur absolue	En complément à 1	En complément à 2
+7	0 111	0 111	0 111
+6	0 110	0 110	0 110
+5	0 101	0 101	0 101
+4	0 100	0 100	0 100
+3	0 011	0 011	0 011
+2	0 010	0 010	0 010
+1	0 001	0 001	0 001
+0	0 000	0 000	0 000
-0	1 000	1 111	0 000
-1	1 001	1 110	1 111
-2	1 010	1 101	1 110
-3	1 011	1 100	1 101
-4	1 100	1 011	1 100
-5	1 101	1 010	1 011
-6	1 110	1 001	1 010
-7	1 111	1 000	1 001
-8	/	/	1 000

📖 dans la représentation en complément à 2, le 0 possède une seule représentation (0000), en effet :
En complément à 1 :
+0 = 0000
-0 = 1111
En complément à 2 :
-0 = 1111 + 1 = 0000, la retenue (1) est ignorée.

💣 En complément à 2, tout nombre possède son opposé.

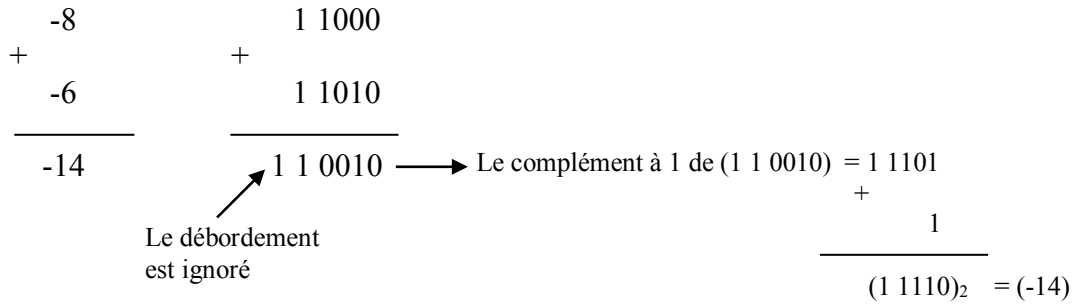
Exemple :
En utilisant le tableau suivant, on peut trouver les opposés des nombres (+2), (-6) et (-8).

Nombre	Complément à 2	Opposé en complément à 2	Nombre opposé
+2	0010	1101 + 1 = 1110	-2
-6	1010	0101 + 1 = 0110	+6
-8	1000	0111 + 1 = 1000	-8

Ainsi, en complément à 2, tout nombre possède un opposé. L'opposé du plus petit nombre est lui-même (-8).

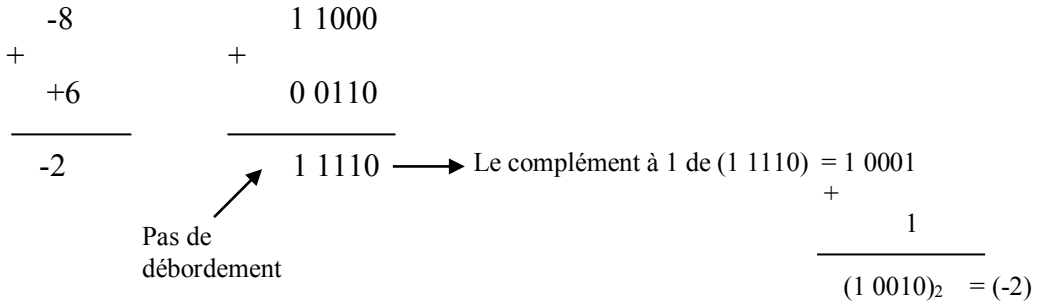
On va effectuer les mêmes opérations que précédemment, mais en complément à 2 :
(+8) : complément à 1 = 0 1000 : complément à 2 = 0 1000
(-8) : complément à 1 = 1 0111 : complément à 2 = 1 1000
(+6) : complément à 1 = 0 0110 : complément à 2 = 0 0110
(-6) : complément à 1 = 1 1001 : complément à 2 = 1 1010

On effectue l'opération (-8) + (-6) :



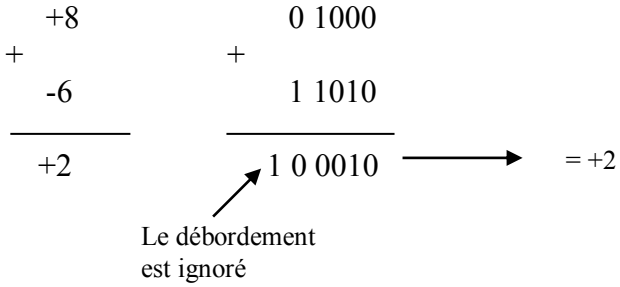
🕮 la retenue générée par le bit de signe (le débordement) est ignorée. Le résultat de l'opération est le nombre (1 0010) en complément à 2 qui est un nombre négatif correspond au nombre :
$(1\ 1110)_2 = (-14)_{10}$

On effectue l'opération (-8) + (+6) :



🕮 il n' y a pas de retenue générée par le bit de signe, le résultat étant négatif (1 1110), exprimé en complément à 2, son équivalent est $(1\ 0010)_2$ qui correspond au nombre $(-2)_{10}$.

On effectue l'opération (+8)+(-6) :



📖 il y a un débordement, il sera ignoré, le résultat obtenu est un nombre positif $(0\ 0010)_2 = (+2)_{10}$.

Conclusion : dans la représentation à 2, l'opération d'addition se déroule comme suit :
- si un débordement est généré, il est ignoré
- si le résultat est négatif, l'exprimer en complément à 2.

3.3. Représentation des nombres fractionnaires : Un nombre fractionnaire est composé de deux parties :
- une partie entière
- une partie fractionnaire ou décimale (à ne pas confondre avec le système décimal)

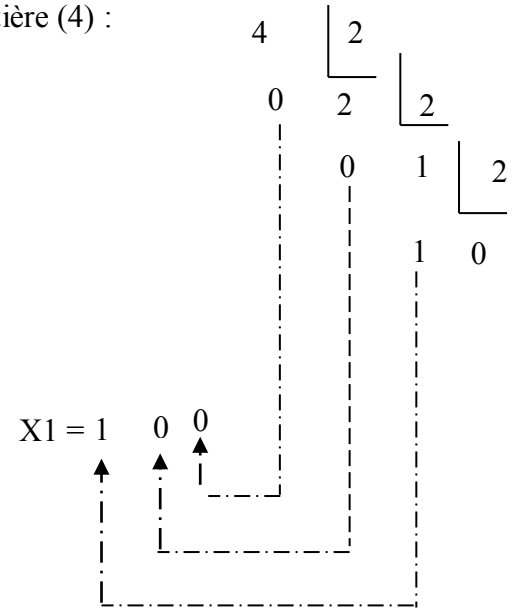
Exemple : 213 , 56
Soit le nombre X exprimer dans une base b sous la forme :
$X = (a_n\ a_{n-1}\ a_{n-2} \ldots a_1\ a_0\ ,\ a_{-1}\ a_{-2} \ldots a_m)_b$ telque n,m deux nombres entiers positives.

- **Passage de la base b à la base 10 :** Tout nombre exprimé dans une base b possède son équivalent en base 10. pour exprimer le nombre X en décimal, il faut traiter séparément la partie entière et la partie décimale.
  - la partie entière : $a_n*b^n + a_{n-1}*b^{n-1} + \ldots + a_1*b^1 + a_0*b^0$
  - la partie décimale : $a_{-1}*b^{-1} + a_{-2}*b^{-2} + \ldots + a_{-m+1}*b^{-m+1} + a_{-m}*b^{-m}$

Exemple :
Soit le nombre $X = (1101\ ,\ 01)_2$. On va exprimer ce nombre en décimal :
La partie entière $(1101)_2 = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 8+4+0+1=13$
La partie décimale $(01)_2 = 0*2^{-1} + 1*2^{-2} = 0+1/4 = 0,25$
Donc, $X = (1101\ ,\ 01)_2 = (13,25)_{10}$

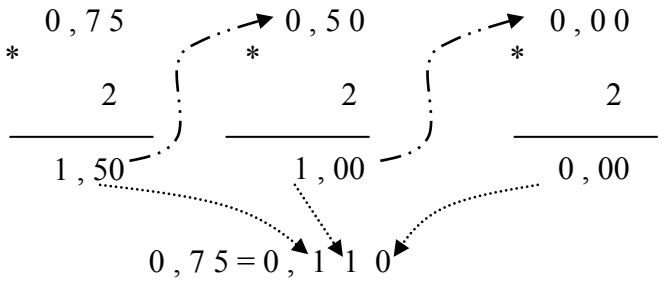
- **Passage de la base 10 à une base b :** La conversion d'un nombre fractionnaire décimal (en base 10) à une autre base b se déroule en deux étapes :
  - convertir la partie entière par divisions successives
  - Au lieu d'effectuer des divisions successives, pour convertir la partie décimale, on effectue des multiplications successives par la base.

Exemple1 :
Soit à convertir en binaire le nombre décimal (en base 10) X1 = 4 , 75
- conversion de la partie entière (4) :



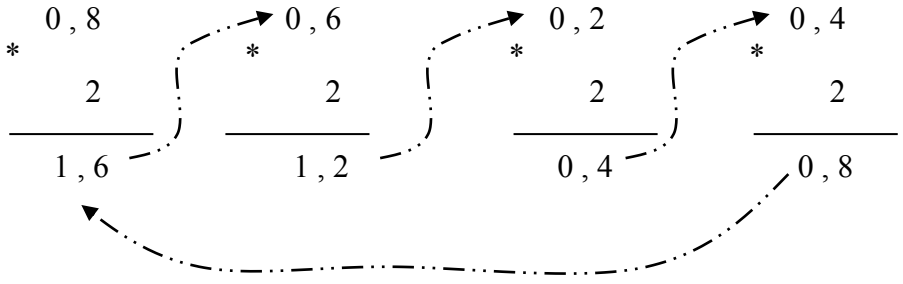
Donc, 4 = $(100)_2$

- conversion de la partie décimale (0 , 75) :
On effectue des multiplications successives, comme suit :



Ainsi, 0 , 75 = $(0 , 11)_2$
Donc, 4 , 75 = $(100 , 11)_2$

Exemple2 :
Soit à convertir en binaire le nombre décimal X2 = 4 , 8
- conversion la partie entière (4) :
4 = $(100)_2$
- conversion la partie décimale (0 , 8) :
On effectue des multiplications successives :



0 , 8 = $(0 , 11001100\ldots)_2$
Donc, l'équivalent binaire du nombre décimal 4,8 est $(100 , 110011\ldots)_2$

Algèbre de Boole et fonctions logiques

1. Systèmes binaires et algèbre de Boole

Actuellement, alors que les ordinateurs analogiques sont encore du domaine de la recherche, les informations traitées par les systèmes informatiques sont codées sous forme binaire. Un système binaire (signal, circuit, etc…) est un système qui ne peut exister que dans deux états autorisés. Le circuit de la figure 1 est un exemple plus que simple de circuit binaire : selon que l'interrupteur S est ouvert ou fermé la tension V_0 ne peut être égale qu'à +5 V ou 0 V.

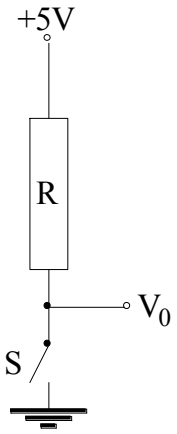


Figure 1

Diverses notations peuvent être utilisées pour représenter ces deux états :

numérique :	1 et 0 (bit : binary digit)
logique :	vrai et faux (true et false)
	oui et non (yes et no)
électronique :	ON et OFF
	haut et bas

🕮 Pour étudier les fonctions de variables binaires on utilise une algèbre développée au XIX^{ème} siècle par un mathématicien anglais : Georges Boole.

L'algèbre de Boole concerne la logique des systèmes binaires. Une variable booléenne ne peut prendre que deux valeurs possibles 0 ou 1. En électronique les deux états d'une telle variable peuvent être associés à deux niveaux de tension : V(0) et V(1) pour les états 0 et 1 respectivement. On distingue les logiques positive et négative selon que V(1) > V(0) ou V(1) < V(0). Ce que nous pouvons résumer dans la table suivante donnant la signification logique des niveaux physiques :

Niveau	Logique positive	Logique négative
H	1	0
B	0	1

Table 1

🕮 En pratique un niveau est défini par un domaine en tension ou en courant. Par exemple, un niveau sera dit haut s'il est compris entre +2 V et +5 V et un niveau sera bas s'il est inférieur à +0.8 V.

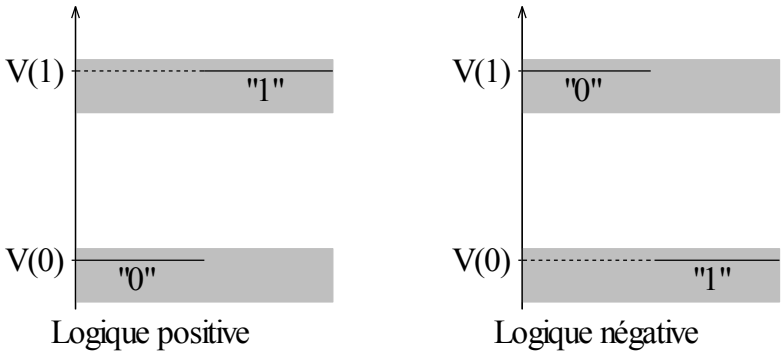


Figure 2

1.1. Porte OU (inclusif)

L'opération OU (OR), encore appelée addition logique, a au moins deux entrées. La sortie d'une fonction OU est dans l'état 1 si au moins une de ses entrées est dans l'état 1. La fonction OU, notée +, est représentée par le symbole indiqué sur la figure 3 et est définie par la table de vérité suivante :

A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Table 2

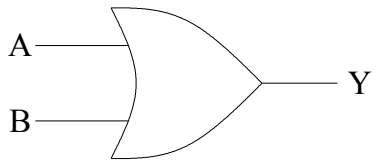


Figure 3

Il est facile de vérifier les propriétés suivantes de la fonction OU :

$(A + B) + C = A + (B + C) = A + B + C$	Associativité
$A + B = B + A$	Commutativité
$A + A = A$	Idempotence
$A + 0 = A$	0 est l'élément neutre
$A + 1 = 1$	1 est l'élément absorbant
$A + \overline{A} = 1$	$\overline{A}$ est l'inverse de A

1.2. Porte ET

L'opération ET (AND), encore dénommée produit logique ou intersection, a au moins deux entrées. La sortie d'une fonction AND est dans l'état 1 si et seulement si toutes ses entrées sont dans l'état 1. La fonction ET, notée •, est représentée par le symbole indiqué sur la figure 4 et est définie par la table de vérité suivante :

A	B	Y = A • B
0	0	0
0	1	0
1	0	0
1	1	1

Table 3

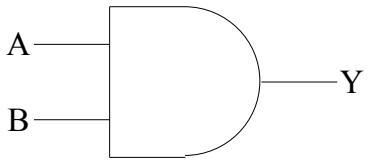


Figure 4

Il est facile de vérifier les propriétés suivantes de la fonction ET :

$(A \bullet B) \bullet C = A \bullet (B \bullet C) = A \bullet B \bullet C$	Associativité
$A \bullet B = B \bullet A$	Commutativité
$A \bullet A = A$	Idempotence
$A \bullet 1 = A$	1 est l'élément neutre
$A \bullet 0 = 0$	0 est l'élément absorbant
$A \bullet \overline{A} = 0$	$\overline{A}$ est l'inverse de A

D'autre part, les opérations ET et OU sont distributives l'une par rapport à l'autre :

$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$

$A + (B \bullet C) = (A + B) \bullet (A + C)$

1.3. Inverseur : porte NON

L'opération NON (NOT) a une seule entrée et une seule sortie. La sortie d'une fonction NON prend l'état 1 si et seulement si son entrée est dans l'état 0. La négation logique est symbolisée par un petit cercle dessiné à l'endroit où une ligne en entrée ou en sortie rejoint un symbole logique, comme par exemple sur la figure 5. La table 4 donne la table de vérité correspondante.

A	$Y = \overline{A}$
0	1
1	0

Table 4

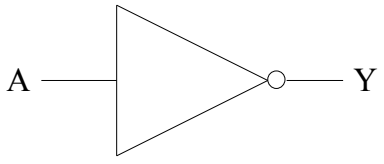


Figure 5

A partir des définitions des fonctions NON, OU et ET nous pouvons déduire :

$$\overline{\overline{A}} = A$$

$$\overline{A} + A = 1$$

$$\overline{A} \bullet A = 0$$

$$A + \left(\overline{A} \bullet B\right) = A + B$$

1.4. Théorèmes de De Morgan

De Morgan a exprimé deux théorèmes qui peuvent se résumer sous la forme suivante :

$$\overline{A \bullet B \bullet C \bullet \ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$$

$$\overline{A + B + C + \ldots} = \overline{A} \bullet \overline{B} \bullet \overline{C} \bullet \ldots$$

Pour vérifier le premier théorème nous remarquons que si toutes les entrées sont à 1 les deux membres de l'équation sont nuls. Par contre si une au moins des entrées est à 0 les deux membres de l'équation sont égaux à 1. Il y a donc égalité quels que soient les états des diverses entrées. Le second théorème se vérifie de la même manière : si toutes les entrées sont à 0 les deux membres de l'équation sont à 1, par contre si au moins une des entrées est à 1 les deux expressions sont à 0.

Les théorèmes de De Morgan montrent qu'une fonction ET peut être fabriquée à partir des fonctions OU et NON. De même une fonction OU peut être obtenue à partir des fonctions ET et NON. La figure 6 montre la conversion d'une porte OU en porte ET et réciproquement, utilisant le fait que :

$$\overline{\overline{A} \bullet \overline{B}} = \overline{\overline{A}} + \overline{\overline{B}} = A + B$$

$$\overline{\overline{A} + \overline{B}} = \overline{\overline{A}} \bullet \overline{\overline{B}} = A \bullet B$$

De même, à partir des théorèmes de De Morgan nous pouvons montrer qu'une porte ET en logique positive fonctionne comme une porte OU en logique négative et vice versa.

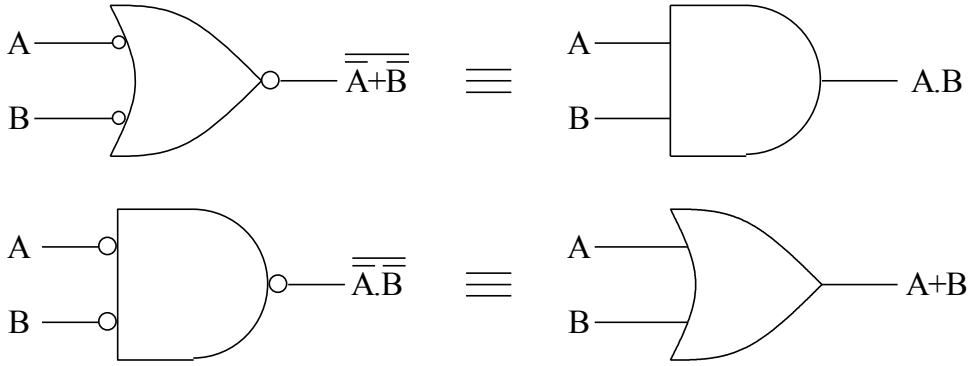


Figure 6

1.5. Portes NON ET et NON OU

Une porte NON ET (NAND : NOT AND) est constituée par un inverseur à la sortie d'une porte ET (fig 7). Une négation à la sortie d'une porte OU constitue une fonction NON OU (NOR : NOT OR) symbolisée sur la figure 8. Leurs tables de vérité respectives sont données par les tables 5 et 6 :

A	B	$Y = \overline{A \bullet B}$
0	0	1
0	1	1
1	0	1
1	1	0

Table 5

A	B	$Y = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

Table 6

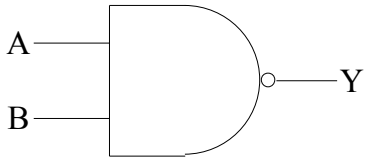


Figure 7

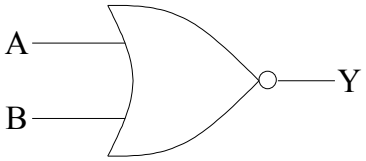


Figure 8

📖 Comme les transistors qui interviennent comme éléments de base des portes sont par essence des inverseurs, les portes NAND et NOR sont très usitées dans la réalisation des circuits logiques. Grâce aux lois de De Morgan il est possible de réaliser des systèmes logiques avec uniquement des portes NAND ou NOR. La figure 9 montre, par exemple, comment les portes NOT, OR et AND peuvent être obtenues à partir de portes NOR.

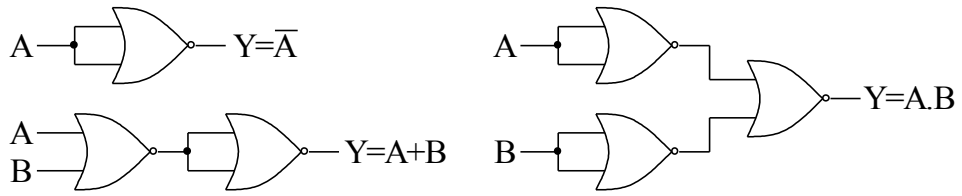


Figure 9

1.6. Porte OU exclusif

La sortie d'une fonction OU exclusif (XOR) à deux entrées est dans l'état 1 si une entrée et seulement une est dans l'état 1. La représentation symbolique d'une fonction XOR (notée $\oplus$) est donnée sur la figure 10 et sa table de vérité est la suivante :

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Table 7

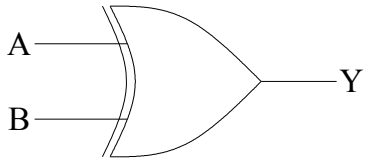


Figure 10

Nous pouvons formuler de diverses manières la définition précédente : $Y = A \oplus B$ est égal à 1 si et seulement si A = 1 ou B = 1 mais pas simultanément. Ce que nous pouvons écrire :

$$A \oplus B = (A + B) \bullet \overline{(A \bullet B)}$$

Nous pouvons encore dire $Y = A \oplus B$ est égal à 1 si A = 1 et B = 0 ou si B = 1 et A = 0. Soit :

$$A \oplus B = (A \bullet \overline{B}) + (B \bullet \overline{A})$$

Une fonction XOR fournit un comparateur d'inégalité : $Y = A \oplus B$ ne vaut 1 que si A et B sont différents. Si A et B sont égaux à 1 ou si A et B sont égaux à 0 alors Y = 0. Ce qui s'écrit :

$$A \oplus B = \overline{(A \bullet B) + (\overline{A} \bullet \overline{B})}$$

La fonction $Z = \overline{Y} = \overline{A \oplus B}$ correspond à un détecteur d'égalité. Nous avons encore la relation suivante qui peut être démontrée en utilisant les théorèmes de De Morgan :

$$A \oplus B = (A + B) \bullet (\overline{A} + \overline{B})$$

A ces quatre relations logiques correspondent quatre circuits réalisant la fonction XOR à partir de portes OR et AND.

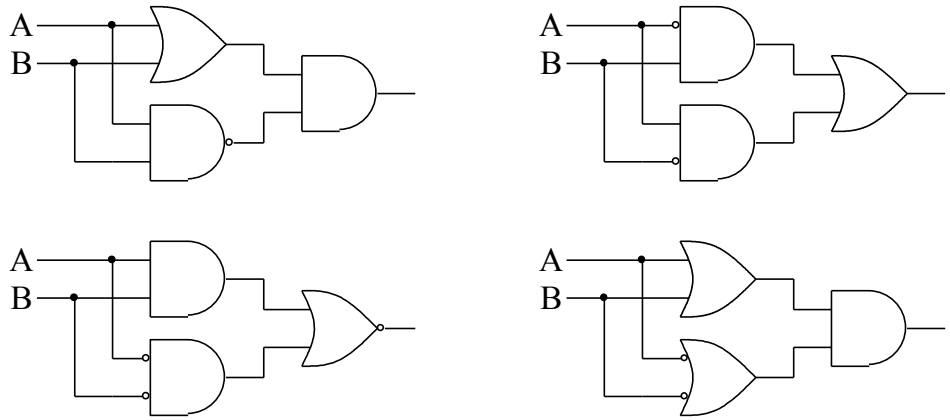


Figure 11

1.7. Porte à Trois Etats

La porte "3 états", ou tri-state", n'est pas une porte logique au sens strict. Elle est principalement utilisée pour connecter une sortie sur une ligne commune à plusieurs circuits (un bus par exemple). Elle remplace généralement une porte ET. En effet, la mise en parallèle sur une même ligne de plusieurs portes ET introduit des capacités parasites. Ceci augmente les constantes de temps et a pour effet de détériorer les fronts de montée et de descente des signaux. Cela peut perturber le fonctionnement d'un système. Une porte 3 états est schématisés sur la figure suivante :

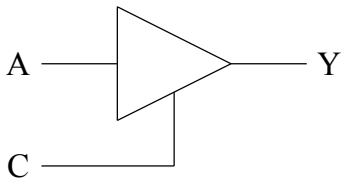


Figure 12

C	A	Y	sortie
1	0	0	faible impédance
1	1	1	faible impédance
0	X	0	haute impédance

Table 8

Lorsque la commande C est à 0 l'impédance de sortie est très grande : pratiquement déconnectée. D'autre part, ces portes "3 états" fournissent une amplification de puissance.

1.8. Résumé des identités booléennes de base

Il est possible de montrer que toute fonction booléenne d'un nombre quelconque de variables peut s'écrire avec les trois fonctions de base ET, OU et NON. Nous avons rassemblé dans la table 9 les relations de base de l'algèbre de Boole qui nous seront utiles par la suite.

OU	$(A + B) + C = A + (B + C) = A + B + C$ $A + B = B + A$ $A + A = A$ $A + 0 = A$ $A + 1 = 1$	Associativité Commutativité Idempotence Elément neutre
ET	$(A \bullet B) \bullet C = A \bullet (B \bullet C) = A \bullet B \bullet C$ $A \bullet B = B \bullet A$ $A \bullet A = A$ $A \bullet 1 = A$ $A \bullet 0 = 0$	Associativité Commutativité Idempotence Elément neutre
Distributivité	$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$ $A + (B \bullet C) = (A + B) \bullet (A + C)$	
NON	$\overline{\overline{A}} = A$ $\overline{A} + A = 1$ $\overline{A} \bullet A = 0$	
	$A + (A \bullet B) = A$ $A \bullet (A + B) = A$ $(A + B) \bullet (A + \overline{B}) = A$ $A + (\overline{A} \bullet B) = A + B$	
De Morgan	$\overline{A \bullet B \bullet C \bullet ...} = \overline{A} + \overline{B} + \overline{C} + ...$ $\overline{A + B + C + ...} = \overline{A} \bullet \overline{B} \bullet \overline{C} \bullet ...$	
OU exclusif	$A \oplus B = (A + B) \bullet \overline{(A \bullet B)}$ $A \oplus B = (A \bullet \overline{B}) + (B \bullet \overline{A})$ $A \oplus B = \overline{(A \bullet B) + (\overline{A} \bullet \overline{B})}$ $A \oplus B = (A + B) \bullet (\overline{A} + \overline{B})$	

Table 9

2. Ecritures canoniques d'une fonction logique

2.1. Somme canonique de produits

Considérons trois variables booléennes x, y et z. A partir de ces trois variables nous pouvons construire huit produits logiques (ou minterms) $P_{i=0,7}$ faisant intervenir x ou $\bar{x}$, y ou $\bar{y}$ et z ou $\bar{z}$. Pour chacune des huit combinaisons $C_{i=0,7}$ (000, 001, 010, etc…) des variables x, y et z, nous pouvons calculer les valeurs de ces produits. Celles-ci sont rassemblées dans la table 10. Chacun de ces produits ne prend la valeur 1 que pour une et une seule combinaison : P_i vaut 1 uniquement pour la combinaison C_i.

				P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
C_i	x	y	z	$\bar{x}\,\bar{y}\,\bar{z}$	$\bar{x}\,\bar{y}\,z$	$\bar{x}\,y\,\bar{z}$	$\bar{x}\,y\,z$	$x\,\bar{y}\,\bar{z}$	$x\,\bar{y}\,z$	$x\,y\,\bar{z}$	$x\,y\,z$
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Table 10

Pour toute fonction logique de trois variables x, y et z, nous pouvons écrire sa table de vérité, c'est-à-dire expliciter sa valeur pour chacune des huit combinaisons C_i. Considérons, par exemple, la fonction F dont la table de vérité est donnée dans la table 11 :

C_i	x	y	z	F	$P_1 + P_3 + P_4$
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	0	0

Table 11

Cette fonction F prend la valeur 1 pour la combinaison C_1 comme le produit P_1, la combinaison C_3 comme P_3 et la combinaison C_4 comme P_4. La fonction F prenant la valeur 0 pour toutes les autres combinaisons comme les produits P_1, P_3, P_4, nous pouvons donc écrire que F est égale à la fonction :

$F = P_1 + P_3 + P_4$

Nous pouvons vérifier cette identité dans la table 11. Nous pouvons donc exprimer F en fonction des variables x, y et z sous la forme :

$F = \bar{x}\,\bar{y}\,z + \bar{x}\,y\,z + x\,\bar{y}\,\bar{z}$

- Cette façon, très générale, d'écrire une fonction booléenne est appelée somme canonique de produits.

2.2. Produit canonique de sommes

Soient encore trois variables binaires x, y et z. Nous pouvons définir huit sommes logiques des trois variables faisant intervenir x ou $\bar{x}$, y ou $\bar{y}$ et z ou $\bar{z}$. La table 12 donne les tables de vérité de ces sommes. Nous constatons que chacune de ces fonctions ne prend la valeur 0 que pour une et une seule combinaison.

				S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
C_i	x	y	z	$x+y+z$	$x+y+\bar{z}$	$x+\bar{y}+z$	$x+\bar{y}+\bar{z}$	$\bar{x}+y+z$	$\bar{x}+y+\bar{z}$	$\bar{x}+\bar{y}+z$	$\bar{x}+\bar{y}+\bar{z}$
0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
2	0	1	0	1	1	0	1	1	1	1	1
3	0	1	1	1	1	1	0	1	1	1	1
4	1	0	0	1	1	1	1	0	1	1	1
5	1	0	1	1	1	1	1	1	0	1	1
6	1	1	0	1	1	1	1	1	1	0	1
7	1	1	1	1	1	1	1	1	1	1	0

Table 12

Reprenons l'exemple précédent de la fonction F. Celle-ci vaut 0 pour les combinaisons C_0, C_2, C_5, C_6 et C_7 en même temps que S_0, S_2, S_5, S_6 et S_7. La fonction F peut donc être vue comme le produit logique de ces cinq sommes, ce qui est vérifié dans la table 13. Nous pouvons donc exprimer la fonction F sous la forme suivante :

$F = (x + y + z) \bullet (x + \bar{y} + z) \bullet (\bar{x} + y + \bar{z}) \bullet (\bar{x} + \bar{y} + z) \bullet (\bar{x} + \bar{y} + \bar{z})$

- Cette écriture est appelée produit canonique de sommes. Celle-ci est moins utilisée que la somme canonique de produits.

C_i	x	y	z	F	$S_0 \bullet S_2 \bullet S_5 \bullet S_6 \bullet S_7$
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	0	0

Table 13

3. Simplification de l'écriture des fonctions logiques

3.1. Simplification algébrique

Simplifier une expression booléenne c'est lui trouver une forme plus condensée, faisant intervenir moins d'opérateurs et conduisant à une réalisation matérielle plus compacte. On peut simplifier une fonction par manipulation algébrique en utilisant par exemple les relations rassemblées dans la table 9. Considérons la fonction F définie par la table de vérité suivante :

x	y	z	F
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 14

Nous en déduisons sa forme canonique somme de produits :

$$F = \bar{x}\, y\, z + x\, \bar{y}\, z + x\, y\, \bar{z} + x\, y\, z$$

Nous pouvons écrire :

$$\begin{aligned} F &= \bar{x}\, y\, z + x\, \bar{y}\, z + x\, y\, \bar{z} + x\, y\, z \\ &= (\bar{x}\, y\, z + x\, y\, z) + (x\, \bar{y}\, z + x\, y\, z) + (x\, y\, \bar{z} + x\, y\, z) \\ &= y\, z\, (\bar{x} + x) + x\, z\, (\bar{y} + y) + x\, y\, (\bar{z} + z) \\ &= x\, y + y\, z + z\, x \end{aligned}$$

Cependant cette méthode, qui demande astuce et chance, n'est pas toujours très aisée à mettre en œuvre. Nous allons maintenant décrire une méthode graphique très utile pour un nombre de variables inférieur à 6.

3.2. Tableaux de Karnaugh

La méthode de simplification de Karnaugh repose sur l'identité :

$$(A \bullet B) + (A \bullet \bar{B}) = A \bullet (B + \bar{B}) = A$$

Elle est basée sur l'inspection visuelle de tableaux disposés de façon telle que les cases adjacentes en ligne et en colonne ne diffèrent que par l'état d'une variable et une seule.

Si une fonction dépend de n variables il y a 2^n produits possibles. Chacun de ces produits est représenté par une case dans un tableau. Les figures suivantes donnent la structure des tableaux de Karnaugh pour 2, 3, 4 et 5 variables. Pour 5 variables, deux représentations sont possibles. Le tableau de Karnaugh peut être traités comme deux tableaux 4x4 superposés (fig. 15) ou un seul tableau de 4x8 (fig. 16). Observez comment sont numérotées les lignes et les colonnes : d'une case à sa voisine une seule variable change d'état.

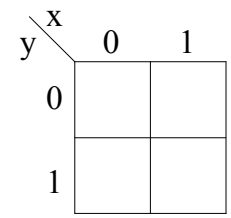


Tableau à 2 variables

Figure 13

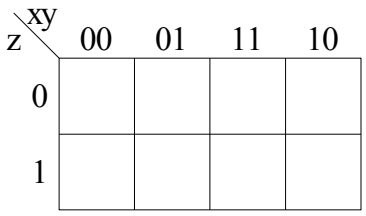


Tableau à 3 variables

Figure 14

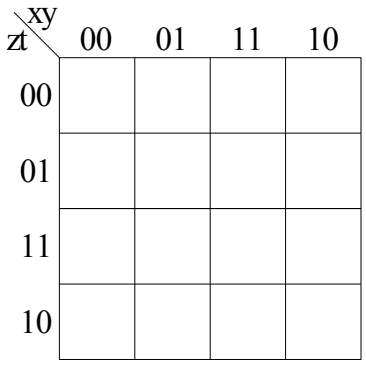


Tableau à 4 variables

Figure 15

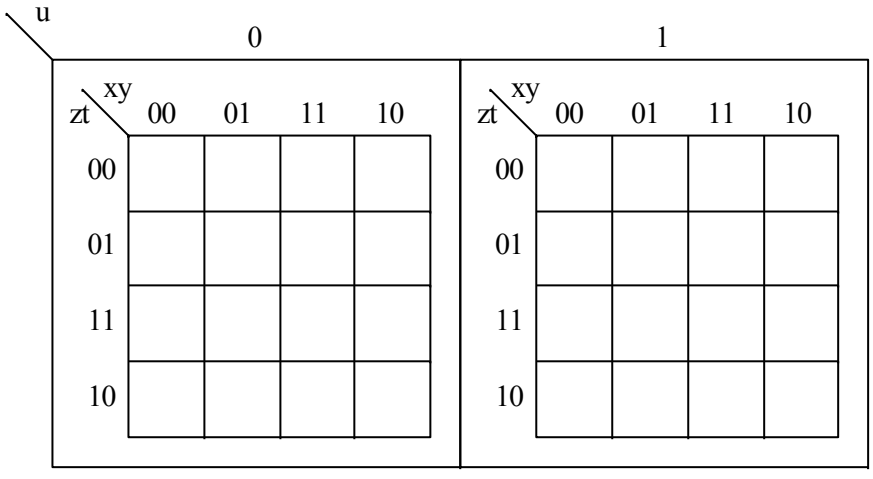


Tableau à 5 variables

Figure 16

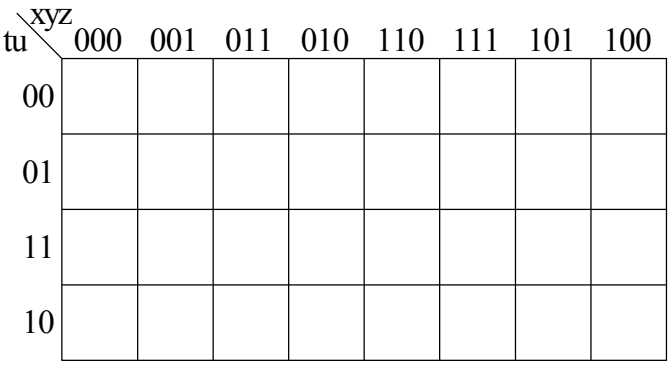


Tableau à 5 variables

Figure 17

Chaque case d'un tableau correspond au seul minterm prenant la valeur 1 pour la combinaison identifiée par la ligne et la colonne. Par exemple les trois cases coloriées dans les tableaux de la figure 18 correspondent respectivement aux produits suivants :

$x\,y\,z\,t$, $\bar{x}\,\bar{y}\,\bar{z}\,\bar{t}$ et $x\,y\,\bar{z}\,\bar{t}$

Il faut comprendre chaque ligne et chaque colonne comme une structure cyclique continue : chaque case a toujours quatre voisins qu'il faut éventuellement chercher à l'autre extrémité de la ligne ou de la colonne. Les tableaux de la figure 18 illustrent ce concept, les croix y matérialisent les voisins des cases coloriées :

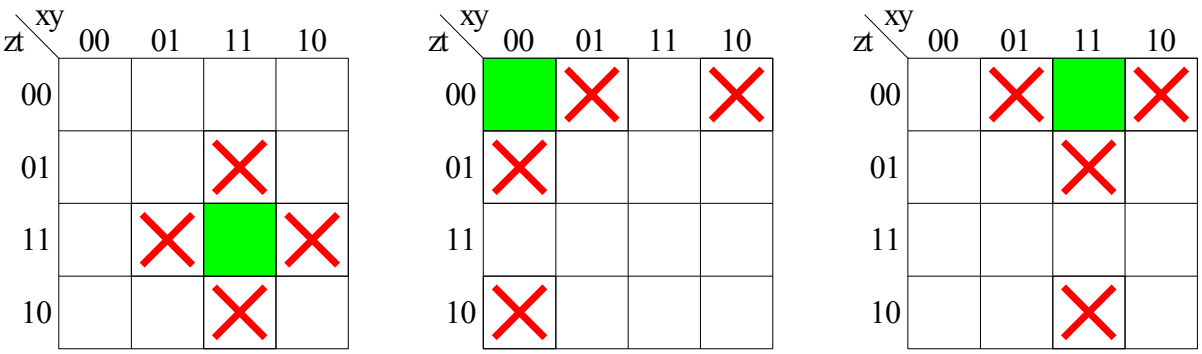


Figure 18

Dans le cas de la représentation en deux tableaux superposés chaque case a cinq voisins : les quatre dans le même plan et une dans l'autre plan.

Le passage de la table de vérité au tableau de Karnaugh consiste à remplir chaque case avec la valeur de la fonction pour le produit correspondant. Il est possible de n'indiquer que les 1.

La méthode de simplification de Karnaugh consiste à rassembler les cases adjacentes contenant des 1 par groupes de 2, 4 ou 8 termes. Considérons en effet le groupement vertical de deux cases, en rouge, de la figure 19. Il correspond à la somme de deux termes :

$G = x\,y\,t + x\,y\,\bar{t}$

Il est possible de factoriser le produit x y :

$G = x\ y\ (t + \bar{t}) = x\ y$

La variable t qui prend les deux valeurs 0 et 1 dans le groupement disparaît. Il ne reste que le produit des variables x et y, qui gardent ici la valeur 1.

Dans un groupement de deux termes on élimine donc la variable qui change d'état et on conserve le produit des variables qui ne changent pas. Dans un groupement de quatre on élimine les deux variables qui changent d'état. Dans un groupement de huit on élimine trois variables, etc…

On cherche à avoir le minimum de groupements, chaque groupement rassemblant le maximum de termes. Une même case peut intervenir dans plusieurs groupements car C + C = C. C'est le cas de la case jaune sur la figure 19.

Pour les cases isolées on ne peut éliminer aucune variable. On conserve donc le produit caractérisant la case. L'expression logique finale est la réunion des groupements après élimination des variables qui changent d'état.

Reprenons l'exemple de la fonction F définie par la table de vérité 14. La figure 19 donne le tableau de Karnaugh correspondant :

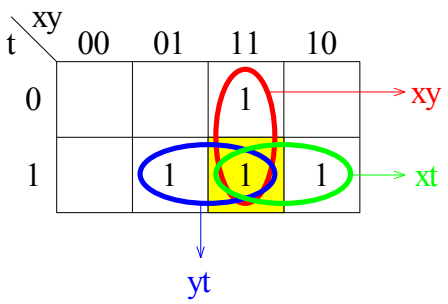


Figure 19

Nous y observons trois groupements de deux termes, nous pouvons écrire pour la fonction :

$F = x\ y + y\ z + z\ x$

Nous retrouvons le résultat précédent.

Considérons une autre fonction F de quatre variables x, y, z et t définie par la table 15. La figure 20 donne le tableau de Karnaugh équivalent. Sur cette figure nous avons également matérialisé les trois groupements possibles : deux groupements de quatre termes, dont un contenant les quatre coins, et un groupement de deux termes. Cette méthode nous permettent d'écrire :

$F = x\,\bar{y} + \bar{y}\,\bar{t} + y\,\bar{z}\,t$

x	y	z	t	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Table 15

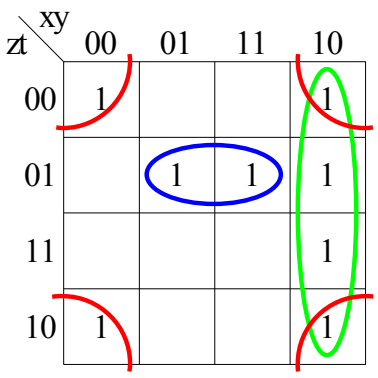


Figure 20

Circuits logiques

Dans ce chapitre nous nous intéressons à une famille de circuits logiques pour lesquels la sortie dépend uniquement des états des entrées.

1. Addition binaire

1.1. Demi-additionneur

Addition et soustraction sont deux opérations arithmétiques de base. Commençons par l'addition de deux nombres binaires, la soustraction sera étudiée dans le prochain paragraphe. En base 2 l'addition de deux bits s'écrit :

$$\begin{cases} 0+0 = 00 \\ 0+1 = 01 \\ 1+0 = 01 \\ 1+1 = 10 \end{cases}$$

Comme en décimal, nous devons donc tenir compte d'une éventuelle retenue (carry). La figure 1 montre la décomposition de l'addition de deux nombres binaires de quatre bits.

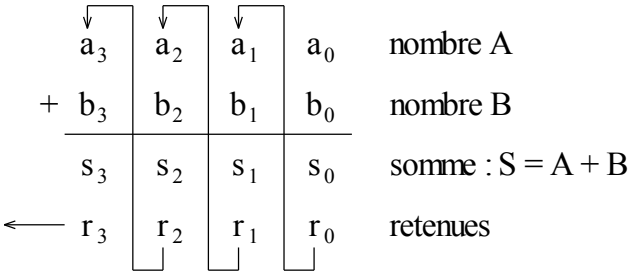


Figure 1

📖 L'addition des deux bits de bas poids (LSB : Least Significant Bit) a_0 et b_0, donne un résultat partiel s_0 et une retenue r_0. On forme ensuite la somme des deux bits a_1 et b_1 et de la retenue r_0. Nous obtenons un résultat partiel s_1 et une retenue r_1. Et ainsi de suite, nous obtenons un résultat sur quatre bits S et une retenue r_3.

Considérons la cellule symbolisée sur la figure 2, comptant deux entrées A et B les deux bits à sommer et deux sorties D le résultat de la somme et C la retenue.

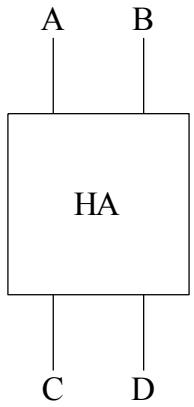


Figure 2

Ce circuit, qui permettrait d'effectuer l'addition des deux bits de plus bas poids est appelé demi-additionneur (Half-Adder). Ecrivons la table de vérité de celui-ci :

A	B	C	D
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Table 1

Si nous écrivons ces deux fonctions sous leur forme canonique il vient :

$$\begin{cases} D = \overline{A}\,B + A\,\overline{B} \\ C = A\,B \end{cases}$$

Nous reconnaissons pour la sortie D une fonction OU exclusif, donc :

$$\begin{cases} D = A \oplus B \\ C = A\,B \end{cases}$$

Ce qui peut être réalisé par le circuit schématisé sur le logigramme de la figure 3.

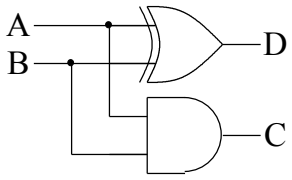


Figure 3

1.2. Additionneur

Il faut en fait tenir compte de la retenue des bits de poids inférieurs, un circuit additionneur doit donc comporter trois entrées et deux sorties, comme représenté sur la figure 4.

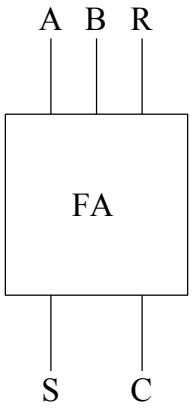


Figure 4

Ce serait possible en combinant deux demi-additionneurs comme présentés par la figure 5. En pratique pour minimiser le nombre de composants, ou de portes dans un circuit intégré, un tel additionneur est réalisé directement.

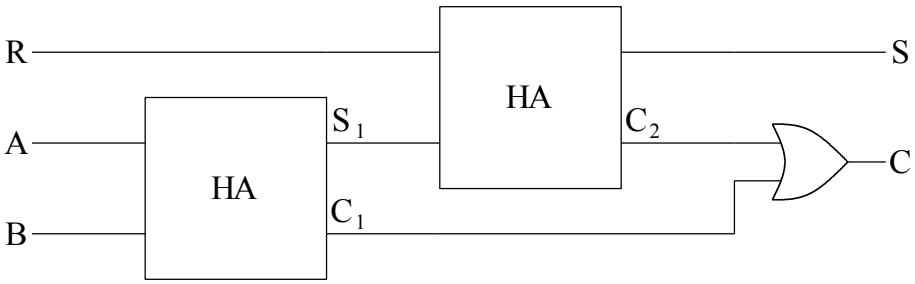


Figure 5

Les entrées A et B représentent les bits à additionner et R le report de la retenue de l'addition des bits de poids inférieurs. La sortie S représente le résultat de la somme et C la retenue. La table de vérité de ce circuit est la suivante :

A	B	R	S	C
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

Table 2

A partir de cette table nous pouvons écrire pour S et C les expressions booléennes suivantes :

$$\begin{cases} S = \overline{A}\,\overline{B}\,R + \overline{A}\,B\,\overline{R} + A\,\overline{B}\,\overline{R} + A\,B\,R \\ C = \overline{A}\,B\,R + A\,\overline{B}\,R + A\,B\,\overline{R} + A\,B\,R \end{cases}$$

Nous pouvons simplifier l'expression de C en utilisant un tableau de Karnaugh :

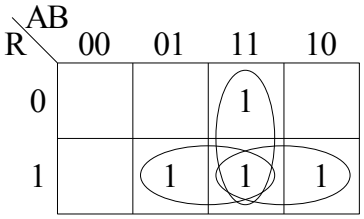


Figure 6

Nous en déduisons :
$C = A\,B + A\,R + B\,R$

Le bit de carry est égal à 1 si au moins deux des entrées sont à 1. D'autre part, nous pouvons remarquer qu'intervertir les 0 et les 1 dans la table 2 revient à permuter les lignes 1 et 8, 2 et 7, 3 et 6, 4 et 5. La table de vérité reste globalement invariante par inversion des entrées et des sorties, nous avons donc :
$\overline{C} = \overline{A}\,\overline{B} + \overline{A}\,\overline{R} + \overline{B}\,\overline{R}$

A partir de cette relation, nous pouvons écrire :

$$\begin{cases} A\,\overline{C} = A\,\overline{B}\,\overline{R} \\ B\,\overline{C} = \overline{A}\,B\,\overline{R} \\ R\,\overline{C} = \overline{A}\,\overline{B}\,R \end{cases} \Rightarrow \quad (A+B+R)\,\overline{C} = A\,\overline{B}\,\overline{R} + \overline{A}\,B\,\overline{R} + \overline{A}\,\overline{B}\,R$$

Ce qui nous permettons de réécrire l'expression de S :
$S = (A+B+R)\,\overline{C} + A\,B\,R$

La figure 7 donne un exemple de réalisation d'un additionneur 1 bit basé sur deux portes AOI (AND OR INVERT), c'est-à-dire un ensemble de portes ET suivies d'une porte NON-OU.

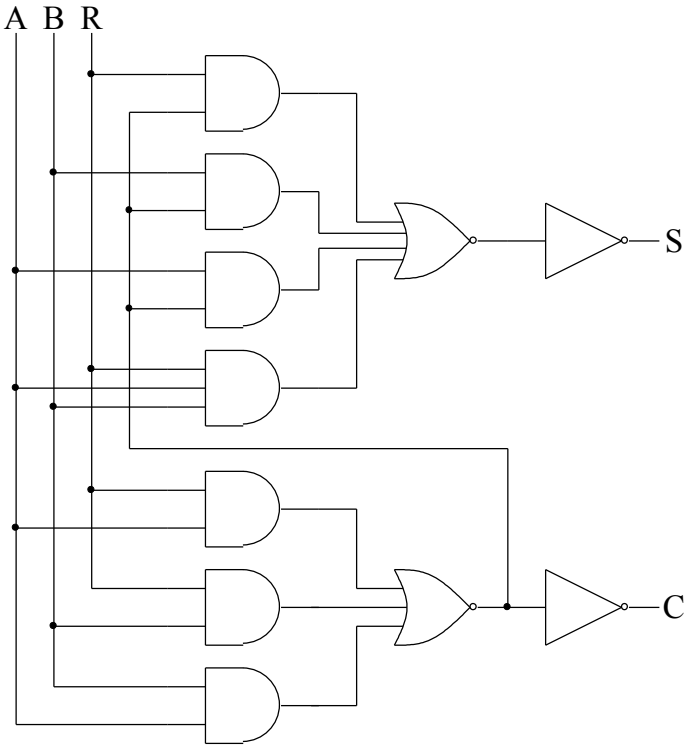


Figure 7

1.3. Addition en parallèle

L'addition de nombres comptant plusieurs bits peut se faire en série (bit après bit) ou en parallèle (tous les bits simultanément). La figure 8 montre l'exemple d'un additionneur 4 bits comptant quatre "Full Adders", comparables à celui schématisé figure 7, montés en parallèle ou en cascade. Chaque additionneur FA_i est affecté à l'addition des bits de poids i. L'entrée correspondant au report de retenue pour FA_0 est imposée à 0 (en logique positive). La retenue finale C indique un dépassement de capacité si elle est égale à 1. Le temps d'établissement du résultat correspondant au temps de propagation des retenues au travers des diverses cellules. Si δt est le temps réponse d'une cellule, la sortie S_0 et la retenue R_0 sont valables après un retard δt, la sortie S_1 et la retenue R_1 ne sont correctes qu'après un retard $2\ \delta t$, et ainsi de suite. La figure 9 présente un exemple de réalisation logique d'un additionneur de deux mots de 2 bits.

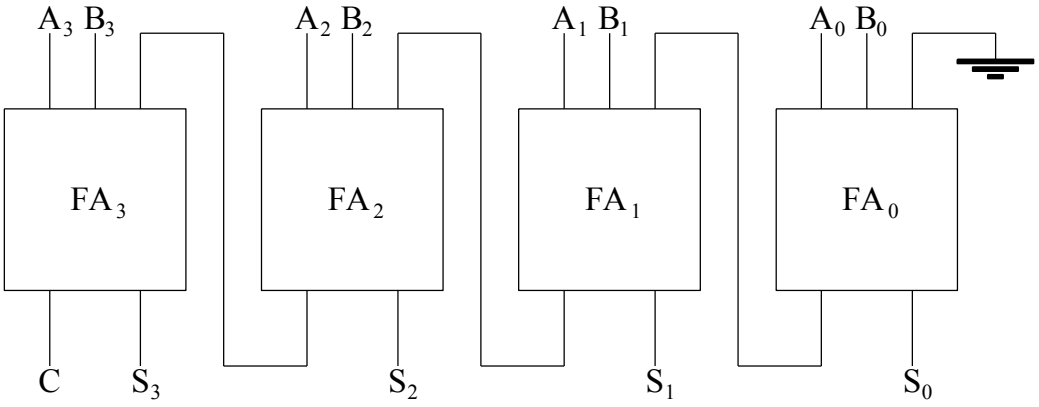


Figure 8

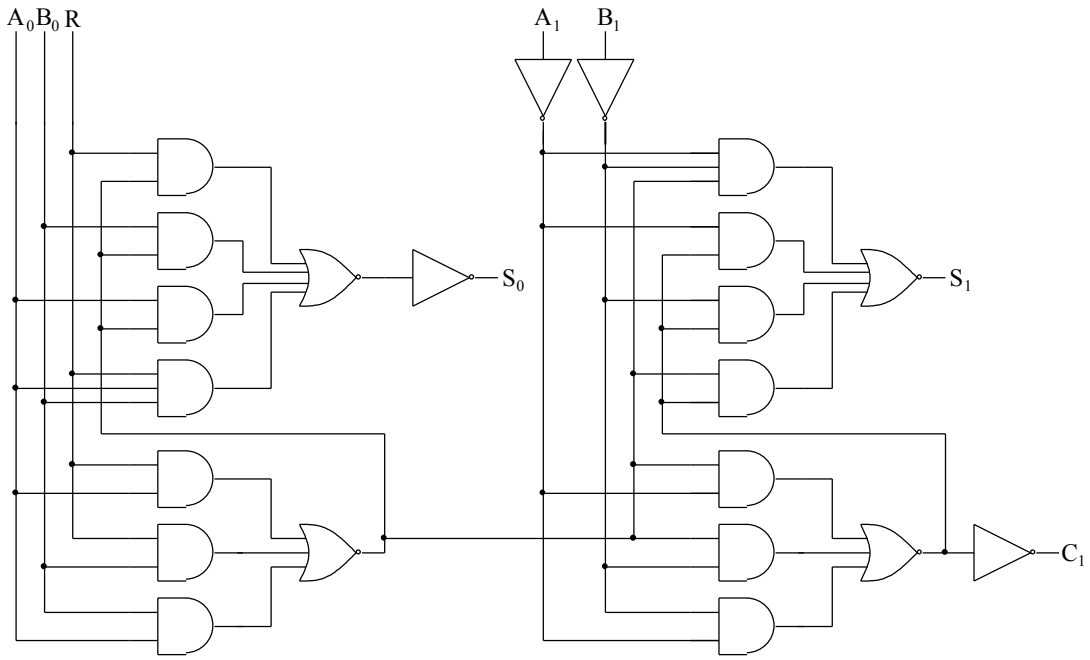


Figure 9

1.4. Addition séquentielle

Dans un additionneur séquentiel chacun des nombres A et B est représenté par un train d'impulsions (figure 10) synchrones par rapport à un signal d'horloge. L'ordre chronologique d'arrivée des impulsions correspond à l'ordre croissant des poids : le bit le moins significatif se présentant le premier. Ces impulsions sont injectées sur les deux lignes d'entrée d'un additionneur (figure 11). A chaque cycle d'horloge, la retenue provenant des bits de poids inférieurs doit être mémorisée (par exemple, à l'aide d'une bascule D qui sera étudiée dans le chapitre suivant).

Un additionneur parallèle est plus rapide mais nécessite plus de composants.

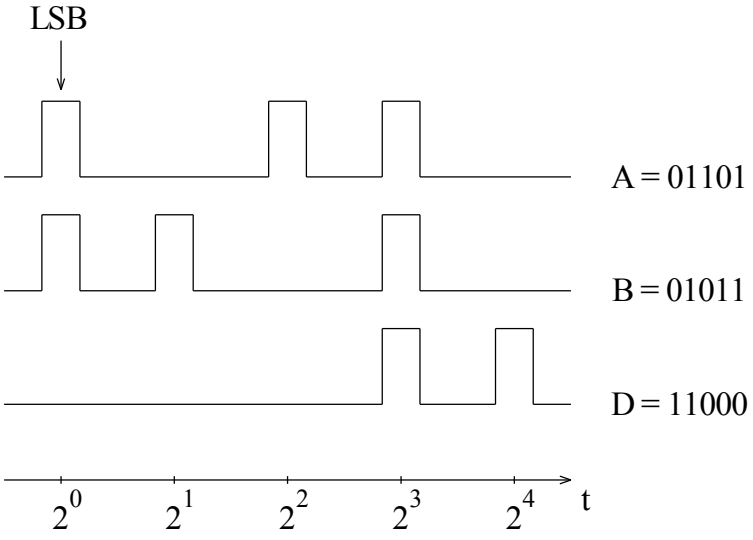


Figure 10

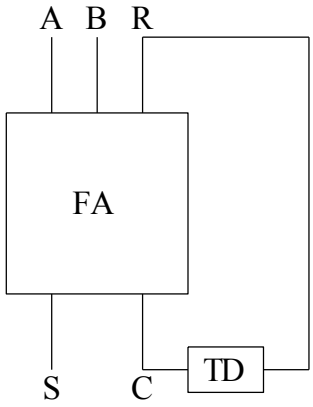


Figure 11

2. Soustraction

2.1. Demi-soustracteur

La table de vérité pour un demi-soustracteur (ne tenant pas compte d'une éventuelle retenue provenant des bits de poids inférieurs) est la suivante :

A	B	D	C
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Table 3

Où D représente le résultat de la soustraction A □ B et C la retenue. Nous en déduisons les expressions logiques définissant D et C :

$$\begin{cases} D = \overline{A}\,B + A\,\overline{B} = A \oplus B \\ C = \overline{A}\,B \end{cases}$$

et le schéma correspondant :

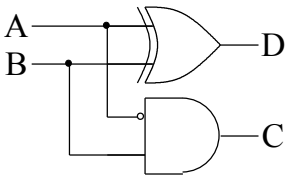


Figure 12

🕮 Nous pourrions maintenant étudier un soustracteur prenant en compte la retenue. Nous allons plutôt tirer parti de certaines propriétés de la numération binaire pour traiter de la même manière l'addition et la soustraction.

2.2. Additionneur-soustracteur

Nous savons qu'avec un mot de n bits nous pouvons représenter un entier positif dont la valeur est comprise entre 0 et 2^n - 1. Le complémentaire d'un mot de n bits est obtenu entre prenant le complément de chacun des n bits. Ainsi, si nous sommons un nombre et son complément nous obtenons un mot dont tous les bits sont à 1. C'est-à-dire :

$$A + \overline{A} = 2^n - 1$$

- dans ce paragraphe le signe + représente l'opération addition et non la fonction logique OU. Nous pouvons encore écrire :

$$-A = \overline{A} + 1 - 2^n$$

Mais sur n bits l'entier 2^n est identique à 0 :

$$2^n \equiv 0 \quad (\text{n bits})$$

C'est-à-dire qu'il est possible d'écrire un nombre entier négatif comme le "complément à 2" de sa valeur absolue :

$$-A = \overline{A} + 1$$

Nous reviendrons sur les divers codages des entiers signés plus tard. Nous pouvons utiliser cette propriété pour écrire la soustraction de deux mots de n bits sous la forme suivante :

$$A - B = A + \overline{B} + 1 - 2^n \equiv A + \overline{B} + 1 \quad (\text{n bits})$$

Ce résultat conduit au schéma de principe présenté sur la figure 13 combinant les fonctions addition et soustraction. Celui-ci est basé sur l'emploi d'un additionneur n bits et d'un multiplexeur à deux lignes d'entrée. Nous étudierons ce type de circuit un peu plus loin dans ce chapitre. Selon le code opération O (0 pour une addition et 1 pour une soustraction) ce multiplexeur permet de sélectionner une des deux entrées, B ou son complémentaire. Le code opération est également injecté sur l'entrée report de retenue de l'additionneur. Pour simplifier le schéma et éviter de représenter n lignes de connexion parallèles, on ne matérialise qu'une seule ligne. Celle-ci est barrée et accompagnée d'une valeur qui indique le nombre réel de connexions.

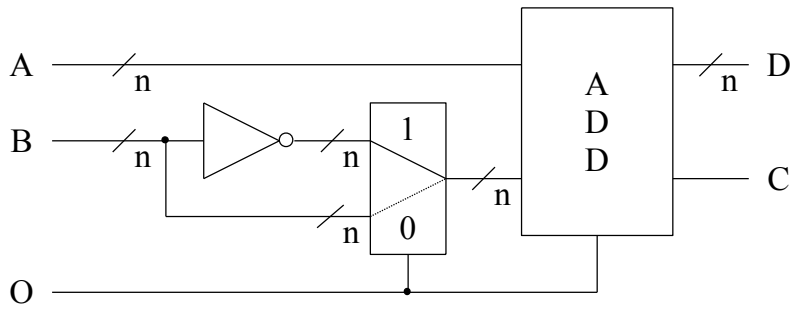


Figure 13

3. Comparaison

On rencontre très souvent la nécessité de comparer deux entiers (A = B, A > B ou A < B). Ecrivons la table de vérité correspondant à ces trois fonctions de comparaison de 2 bits. La fonction C doit être égale à 1 si et seulement si A > B, la fonction D si et seulement si A < B et la fonction E si et seulement si A = B. Ce qui nous donne :

A	B	C (A > B)	D (A < B)	E (A = B)
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Table 5

Nous en déduisons les expressions logiques de C, D et E :

$$\begin{cases} C = A\,\overline{B} \\ D = \overline{A}\,B \\ E = \overline{A \oplus B} = \overline{A\,\overline{B} + \overline{A}\,B} = \overline{C + D} \end{cases}$$

La figure 14 présente le diagramme d'un bloc logique comparant deux bits A et B.

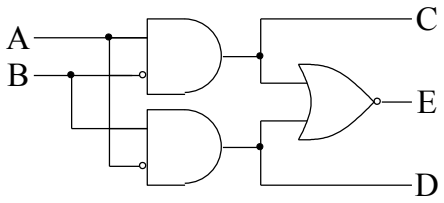


Figure 14

4. Contrôle de parité

La parité d'un mot binaire est définie comme la parité de la somme des bits, soit encore :

- parité paire (ou 0) : nombre pair de 1 dans le mot;
- parité impaire (ou 1) : nombre impair de 1 dans le mot.

La fonction OU-exclusif donne la parité d'un sous-ensemble de deux bits. Le contrôle de parité est basé sur la constatation que le mot de n+1 bits formé en adjoignant à un mot de n bits son bit de parité est toujours de parité 0. La figure 15 représente le diagramme logique d'un générateur-contrôleur de parité pour 4 bits. Si l'entrée P' est imposée à 0 ce circuit fonctionne comme générateur de parité : la sortie P représente la parité du mot composé par les bits A, B, C et D.

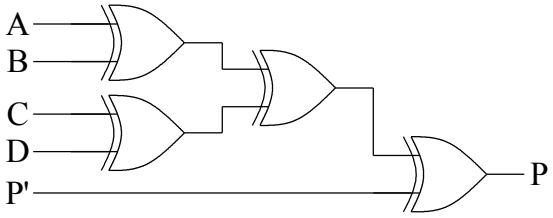


Figure 15

Le contrôle de la parité est utilisé, par exemple, pour augmenter la fiabilité d'un système de transmission ou de stockage de données. La figure 16 montre l'utilisation du circuit précédent en générateur de parité du côté de l'émission et contrôleur de parité du côté de la réception. La sortie P_2 doit être à 0 pour chaque mot transmis, sinon cela indique un problème de transmission.

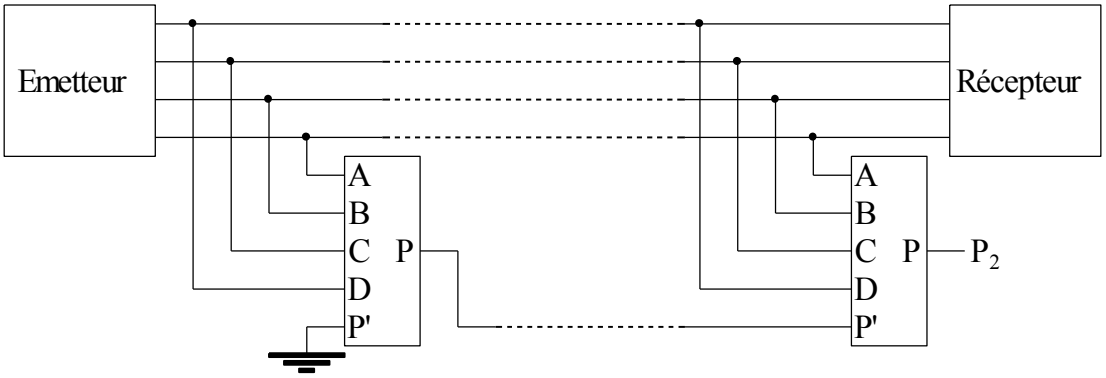


Figure 16

🕮 Remarquons cependant que la parité ne permet de détecter qu'un nombre impair de bits en erreur dans un mot. Par ailleurs il ne permet pas corriger les erreurs détectées. Pour ce faire il faut utiliser des codes correcteurs d'erreur qui nécessitent plusieurs bits supplémentaires.

5. Décodage

Dans un système numérique les instructions, tout comme les nombres, sont transportées sous forme de mots binaires. Par exemple un mot de 4 bits peut permettre d'identifier 16 instructions différentes: l'information est codée. Très souvent l'équivalent d'un commutateur à 16 positions permet de sélectionner l'instruction correspondant à un code. Ce processus est appelé décodage. La fonction de décodage consiste à faire correspondre à un code présent en entrée sur n lignes une seule sortie active parmi les $N = 2^n$ sorties possibles. A titre d'exemple, nous allons étudier le décodage de la représentation DCB des nombres.

5.1. Représentation DCB (Décimale Codée Binaire)

Le code DCB (ou en anglais BCD : Binary Coded Decimal) transforme les nombres décimaux en remplaçant chacun des chiffres décimaux par 4 chiffres binaires. Cette représentation conserve donc la structure décimale : unités, dizaines, centaines, milliers, etc… Chaque chiffre est codé sur 4 bits selon le code de la table 6 :

Décimal	DCB
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Table 6

Par exemple le nombre décimal 294 sera codé en DCB : 0010 1001 0100. Ce type de codage permet, par exemple, de faciliter l'affichage en décimal du contenu d'un compteur. Pour ce faire on peut utiliser des tubes de Nixie, contenant 10 cathodes ayant chacune la forme d'un chiffre (fig. 17) ou des afficheurs lumineux à sept segment (fig. 18).

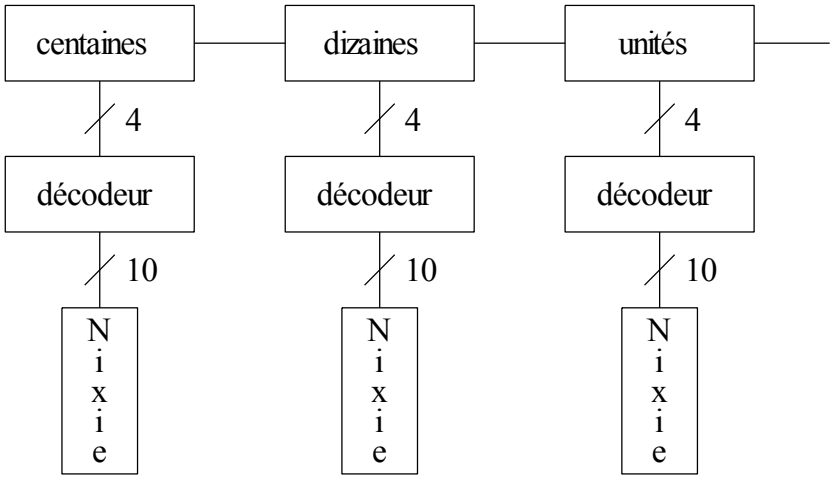


Figure 17

La fonction de chaque décodeur est d'activer une des dix lignes en sortie et une seule en fonction du code présent sur les quatre entrées. Par exemple, si ce code est égal à 5, la 6ème ligne de sortie est mise dans l'état 1 et le chiffre 5 est affiché par le tube de Nixie.

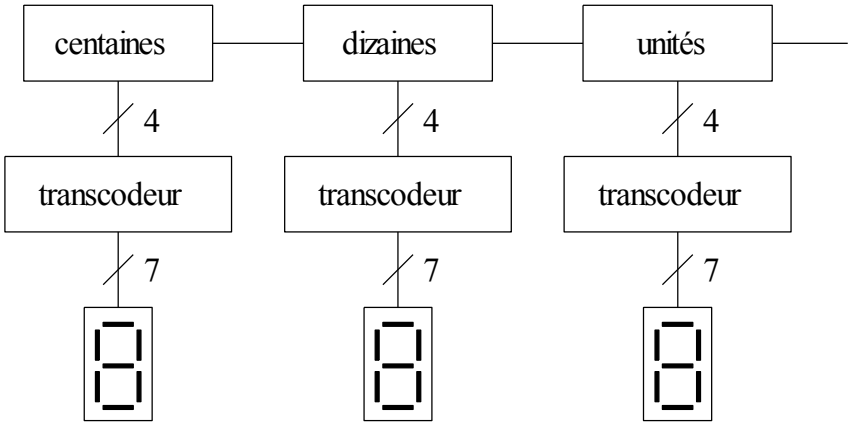


Figure 18

🕮 La fonction de chacun des transcodeurs est de positionner à 1 les lignes de sortie correspondant aux segments à allumer selon de code porté par les quatre lignes d'entrée. De manière générale, un transcodeur fait correspondre à un code A en entrée sur n lignes, un code B en sortie sur m lignes.

5.2. Décodeur DCB-décimal

Nous allons étudier l'exemple d'un décodeur DCB-décimal. La table de vérité de ce décodeur est très simple :

D	C	B	A	L_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9
0	0	0	0	1									
0	0	0	1		1								
0	0	1	0			1							
0	0	1	1				1						
0	1	0	0					1					
0	1	0	1						1				
0	1	1	0							1			
0	1	1	1								1		
1	0	0	0									1	
1	0	0	1										1

Table 7

A chacune des lignes de sortie nous pouvons associer un produit prenant en compte chacune des quatre entrées ou leur complément. Ainsi la ligne 5 correspond à :

$$A\,\overline{B}\,C\,\overline{D}$$

D'autre part, on souhaite souvent n'activer les lignes de sortie qu'en présence d'un signal de commande global (strobe ou enable). Ce signal S est mis en coïncidence sur chacune des dix portes de sortie. Dans l'exemple suivant, si S est dans l'état 0 le décodeur est bloqué et tous les sorties sont également dans l'état 0.

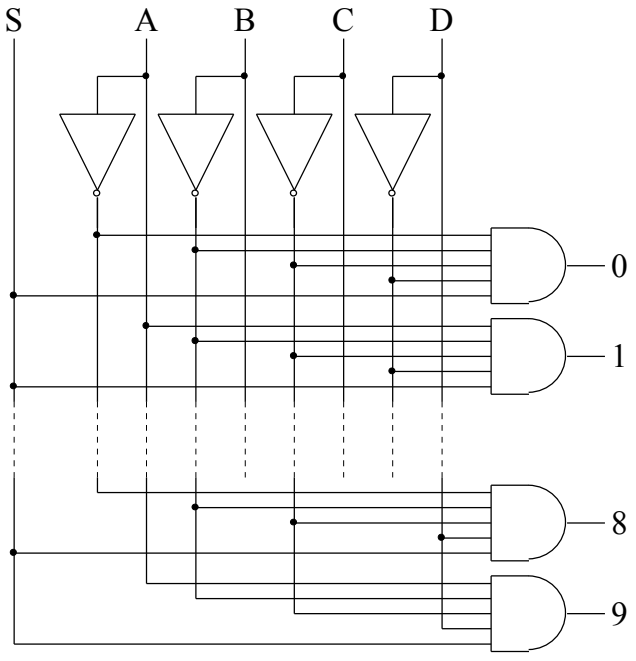


Figure 19

6. Multiplexage

Le multiplexage est un dispositif qui permet de transmettre sur une seule ligne des informations en provenance de plusieurs sources ou à destination de plusieurs cibles. La figure 20 en présente une analogie mécanique avec deux commutateurs à plusieurs positions. Choisir une ligne revient à définir l'angle du levier ou une adresse.

Figure 20

6.1. Démultiplexeur

Un démultiplexeur est un circuit comptant une entrée et N sorties et qui met en relation cette entrée avec une sortie et une seule. Pour pouvoir sélectionner cette sortie il faut également des lignes d'adressage : le code porté par ces lignes identifie la ligne de sortie à utiliser. Ce circuit est très proche d'un décodeur. Considérons un démultiplexeur avec quatre lignes de sortie. Il faut deux lignes d'adresse. Supposons que nous souhaitons également valider les données avec un signal de contrôle E (pour laisser par exemple le temps aux niveaux d'entrée de se stabiliser). Par convention nous choisissons de prendre en compte les données pour E = 0.

E	B	A	Y_0	Y_1	Y_2	Y_3	Produit
0	0	0	D	0	0	0	$\overline{A}\,\overline{B}\,\overline{E}\,D$
0	0	1	0	D	0	0	$A\,\overline{B}\,\overline{E}\,D$
0	1	0	0	0	D	0	$\overline{A}\,B\,\overline{E}\,D$
0	1	1	0	0	0	D	$A\,B\,\overline{E}\,D$
1	0	0	0	0	0	0	
1	0	1	0	0	0	0	
1	1	0	0	0	0	0	
1	1	1	0	0	0	0	

Table 8

De la table 8 nous déduisons le logigramme suivant :

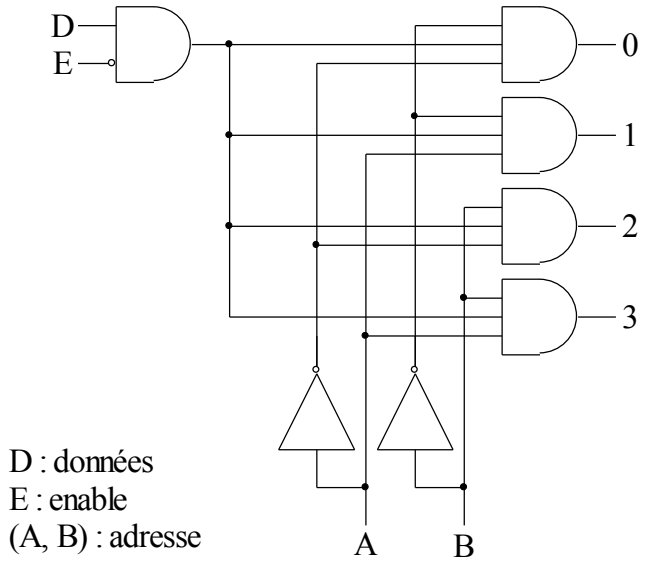


Figure 21

Il existe sous forme de circuits intégrés des démultiplexeurs avec 2, 4 ou 16 lignes de sortie. Pour constituer des démultiplexeurs d'ordre supérieur on peut être amené à cascader des démultiplexeurs. Par exemple un démultiplexeur avec 32 sorties peut être réalisé avec un "tronc" de 4 sorties et 4 "branches" de 8 sorties :

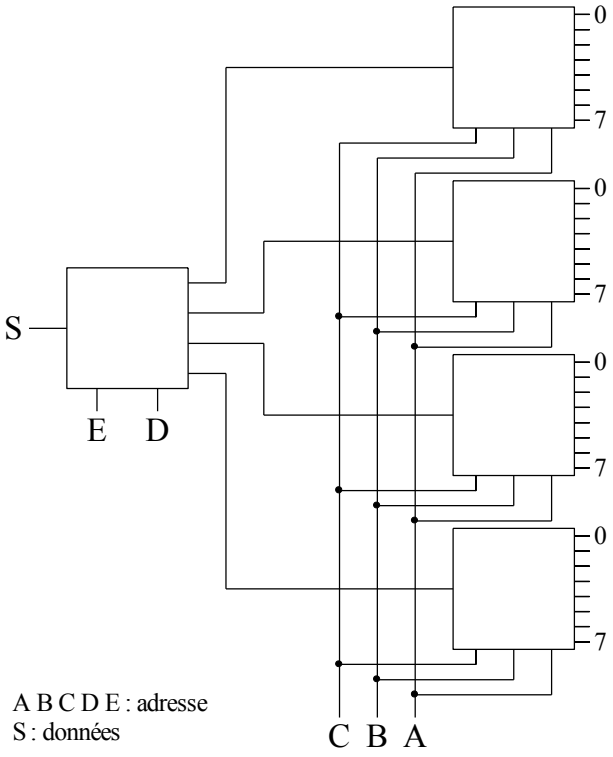


Figure 22

6.2. Multiplexeur

Un multiplexeur, réalise l'opération inverse. Il sélectionne une entrée parmi N et transmet l'information portée par cette ligne à un seul canal de sortie. Considérons un multiplexeur à quatre entrées, donc deux lignes d'adressage, et une ligne de validation. La table de vérité de ce circuit est donnée par la table 9. De cette table nous déduisons une expression logique pour la sortie :

$$Y = \overline{A}\,\overline{B}\,\overline{E}\,X_0 + A\,\overline{B}\,\overline{E}\,X_1 + \overline{A}\,B\,\overline{E}\,X_2 + A\,B\,\overline{E}\,X_3$$

Cette expression correspond au schéma présenté sur la figure 23.

E	B	A	Y
0	0	0	X_0
0	0	1	X_1
0	1	0	X_2
0	1	1	X_3
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Table 9

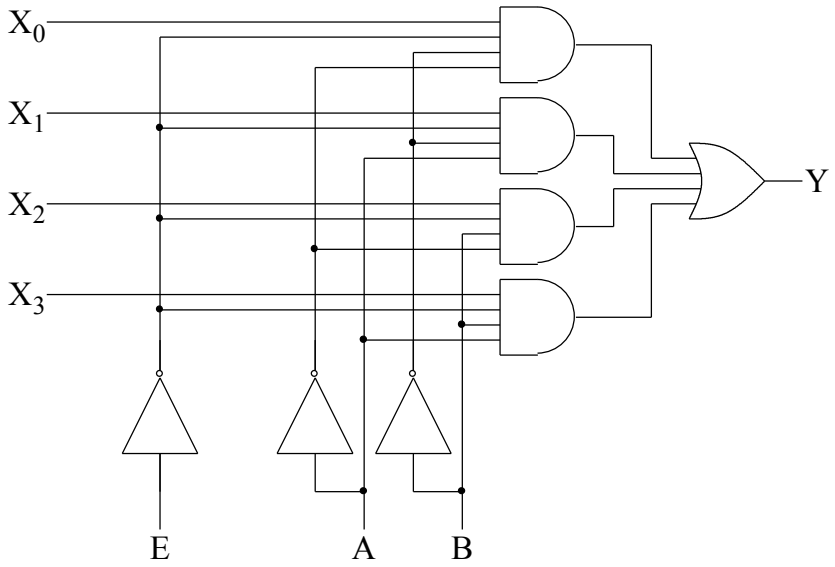


Figure 23

Tout comme pour les démultiplexeurs on peut cascader plusieurs multiplexeurs pour obtenir un multiplexeur d'ordre supérieur. La figure 24 montre comment un multiplexeur à 32 entrées peut être réalisé à partir de quatre multiplexeurs à 8 entrées et d'un multiplexeur à 4 entrées.

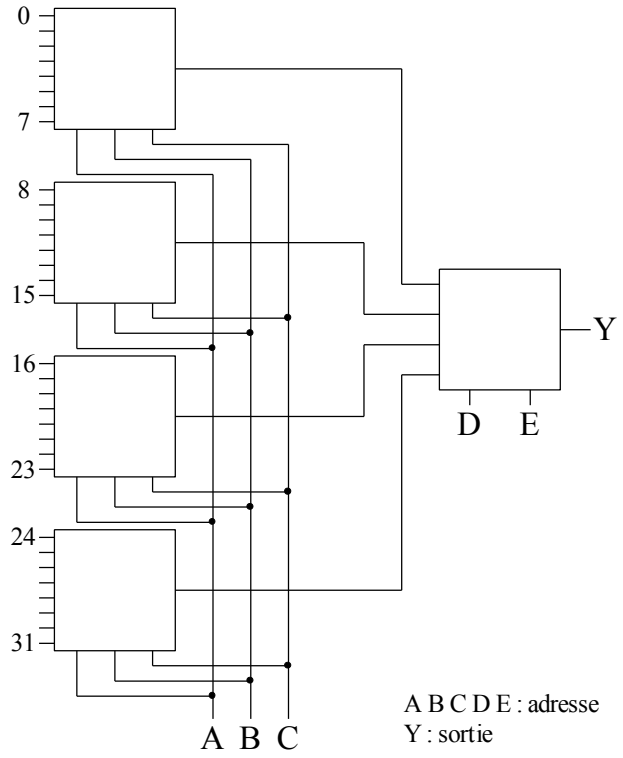


Figure 24

6.3. Conversion parallèle-série

Considérons un mot de n bits (par exemple 4) présent en parallèle sur les entrées d'un multiplexeur :

- X_0 ≡ bit correspondant à 2^0;
- X_1 ≡ bit correspondant à 2^1;
- X_2 ≡ bit correspondant à 2^2;
- X_3 ≡ bit correspondant à 2^3.

Supposons que les lignes d'adresse A et B soient connectées aux sorties d'un compteur de période T, nous aurons en fonction du temps :

t	B	A	Y
[0,T]	0	0	X_0
[T,2T]	0	1	X_1
[2T,3T]	1	0	X_2
[3T,4T]	1	1	X_3
[4T,5T]	0	0	X_0

Table 10

Les bits X_0, X_1, X_2 et X_3 se retrouvent en série dans le temps sur la sortie Y du multiplexeur.

7. Encodage

Nous venons d'étudier le principe du décodage, passons à l'opération inverse ou encodage. Un encodeur est un système qui comporte N lignes d'entrée et n lignes de sortie. Lorsqu'une des lignes d'entrée est activée l'encodeur fournit en sortie un mot de n bits correspondant au codage de l'information identifiée par la ligne activée.

Considérons un encodeur transformant un nombre décimal en son équivalent en code DCB. Il comportera donc 10 entrées (0 à 9) et 4 sorties. Nous pouvons par exemple imaginer que chacune des dix lignes d'entrée peut être reliée à une touche d'un clavier. La table 11 correspond à la table de vérité de cet encodeur. A partir de cette table nous pouvons écrire les expressions logiques définissant les sorties à partir des entrées.

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

Table 11

$$\begin{cases} Y_0 = W_1 + W_3 + W_5 + W_7 + W_9 \\ Y_1 = W_2 + W_3 + W_6 + W_7 \\ Y_2 = W_4 + W_5 + W_6 + W_7 \\ Y_3 = W_8 + W_9 \end{cases}$$

En effet Y_0 est égal à 1 quand la ligne W_1 est dans l'état 1, ou la ligne W_3, ou la ligne W_5, ou la ligne W_7, ou la ligne W_9. La ligne Y_0 est nulle dans tous les autres cas. Il est possible de réaliser ces fonctions OU avec des diodes selon le montage de la figure suivante :

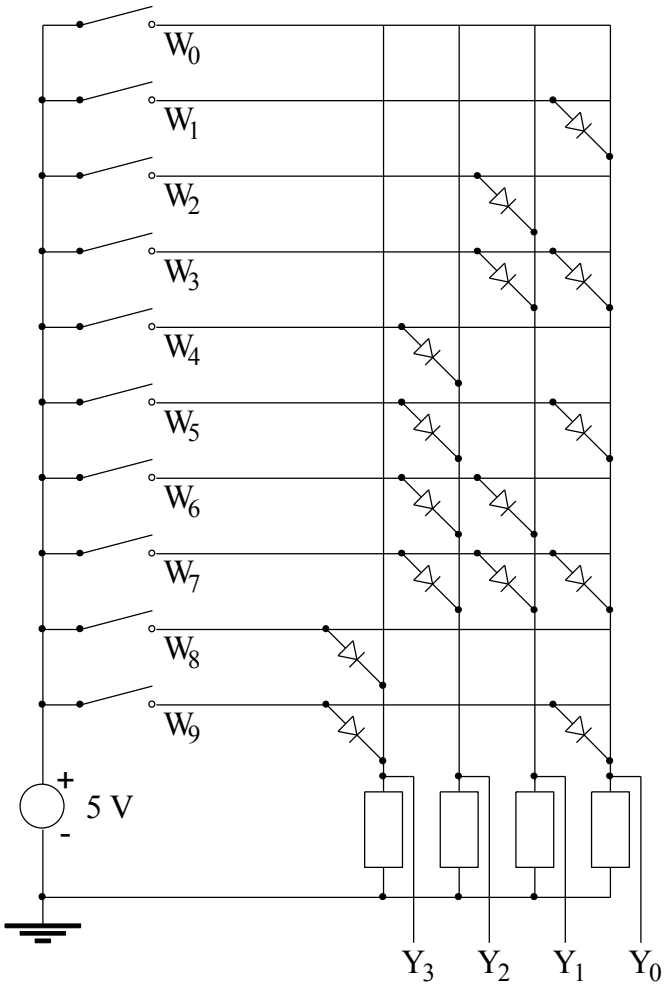
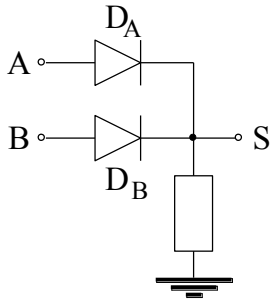


Figure 25

En effet considérons le circuit de la figure 26 :



A	B		S
0	0	D_A et D_B bloquées	0
+V	0	D_A passante/D_B bloquée	+V
0	+V	D_A bloquée/D_B passante	+V
+V	+V	D_A et D_B passantes	+V

Figure 26

Si nous traduisons la signification logique des niveaux haut et bas en logique positive, au circuit de la figure 26 correspond la table de vérité 12. La fonction réalisée est donc un OU inclusif.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Table 12

La figure 25 représente un exemple de réalisation d'un encodeur DCB réalisé avec des diodes. Le bon fonctionnement de ce codeur suppose qu'une seule ligne d'entrée peut être dans l'état 1.

Par contre, si plusieurs entrées sont actives simultanément le résultat pourra ne pas avoir de signification. Par exemple, si les deux lignes W_7 et W_8 sont dans l'état 1 (frappe simultanée des deux touches), il en sera de même pour les quatre sorties. Pour éviter ce problème on utilise un encodeur prioritaire. Pour ce type de circuit si plusieurs lignes d'entrée sont actives simultanément le résultat correspondant à une seule parmi celles-ci est affiché en sortie. La règle peut être, par exemple, de mettre en sortie le code correspondant à la ligne d'entrée d'indice le plus élevé. Par exemple, si W_7 et W_8 sont dans l'état 1 l'encodeur prioritaire donne en sortie le code correspondant à W_8. La table de vérité correspondant à ce choix est donnée par la table 13. Chaque croix indique que le code en sortie doit être indépendant de l'état de l'entrée concernée.

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
X	1	0	0	0	0	0	0	0	0	0	0	0	1
X	X	1	0	0	0	0	0	0	0	0	0	1	0
X	X	X	1	0	0	0	0	0	0	0	0	1	1
X	X	X	X	1	0	0	0	0	0	0	1	0	0
X	X	X	X	X	1	0	0	0	0	0	1	0	1
X	X	X	X	X	X	1	0	0	0	0	1	1	0
X	X	X	X	X	X	X	1	0	0	0	1	1	1
X	X	X	X	X	X	X	X	1	0	1	0	0	0
X	X	X	X	X	X	X	X	X	1	1	0	0	1

Table 13

Alors que les expressions logiques définissant les lignes de sortie Y_i ne dépendaient que des 1 dans la table 11, il faut ici tenir compte des 0. Par exemple pour Y_0 nous avons :

$$Y_0 = W_1 \overline{W}_2 \overline{W}_3 \overline{W}_4 \overline{W}_5 \overline{W}_6 \overline{W}_7 \overline{W}_8 \overline{W}_9 + W_3 \overline{W}_4 \overline{W}_5 \overline{W}_6 \overline{W}_7 \overline{W}_8 \overline{W}_9 + W_5 \overline{W}_6 \overline{W}_7 \overline{W}_8 \overline{W}_9 + W_7 \overline{W}_8 \overline{W}_9 + W_9$$

Nous pouvons mettre le complémentaire de W_9 en facteur dans les quatre premiers termes, puis en utilisant l'identité :

$$A + \overline{A}\, B = A + B$$

Il vient, après factorisation du complément de W_8 :

$$Y_0 = \overline{W}_8 (W_1 \overline{W}_2 \overline{W}_3 \overline{W}_4 \overline{W}_5 \overline{W}_6 \overline{W}_7 + W_3 \overline{W}_4 \overline{W}_5 \overline{W}_6 \overline{W}_7 + W_5 \overline{W}_6 \overline{W}_7 + W_7) + W_9$$

Soit encore :

$$Y_0 = \overline{W}_8\ (W_1\ \overline{W}_2\ \overline{W}_3\ \overline{W}_4\ \overline{W}_5\ \overline{W}_6 + W_3\ \overline{W}_4\ \overline{W}_5\ \overline{W}_6 + W_5\ \overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\ ((W_1\ \overline{W}_2\ \overline{W}_3\ \overline{W}_4\ \overline{W}_5 + W_3\ \overline{W}_4\ \overline{W}_5 + W_5)\ \overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\ ((W_1\ \overline{W}_2\ \overline{W}_3\ \overline{W}_4 + W_3\ \overline{W}_4 + W_5)\ \overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\ (((W_1\ \overline{W}_2\ \overline{W}_3 + W_3)\ \overline{W}_4 + W_5)\ \overline{W}_6 + W_7) + W_9$$

$$Y_0 = \overline{W}_8\ (((W_1\ \overline{W}_2 + W_3)\ \overline{W}_4 + W_5)\ \overline{W}_6 + W_7) + W_9$$

Soit en réorganisant l'ordre des termes :

$$Y_0 = W_9 + \overline{W}_8\ (W_7 + \overline{W}_6\ (W_5 + \overline{W}_4\ (W_3 + W_1\ \overline{W}_2)))$$

Pour la ligne Y_1 nous avons :

$$Y_1 = W_2\ \overline{W}_3\ \overline{W}_4\ \overline{W}_5\ \overline{W}_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_3\ \overline{W}_4\ \overline{W}_5\ \overline{W}_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_7\ \overline{W}_8\ \overline{W}_9$$

Soit en factorisant :

$$Y_1 = \overline{W}_8\ \overline{W}_9\ (W_7 + \overline{W}_7\ (W_6 + \overline{W}_6\ (W_3 + W_2\ \overline{W}_3)\ \overline{W}_4\ \overline{W}_5))$$

En utilisant toujours la même identité nous pouvons simplifier cette expression, il vient en réordonnant les termes :

$$Y_1 = \overline{W}_9\ \overline{W}_8\ (W_7 + W_6 + \overline{W}_5\ \overline{W}_4\ (W_3 + W_2))$$

Pour Y_2 nous devons écrire :

$$Y_2 = W_4\ \overline{W}_5\ \overline{W}_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_5\ \overline{W}_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_6\ \overline{W}_7\ \overline{W}_8\ \overline{W}_9 + W_7\ \overline{W}_8\ \overline{W}_9$$

Soit encore :

$$Y_2 = \overline{W}_9\ \overline{W}_8\ (W_7 + \overline{W}_7\ (W_6 + \overline{W}_6\ (W_5 + \overline{W}_5\ W_4)))$$

En utilisant toujours la même identité il vient :

$$Y_2 = \overline{W}_9\ \overline{W}_8\ (W_7 + W_6 + W_5 + W_4)$$

Enfin pour Y_3 nous avons :

$$Y_3 = W_8\ \overline{W}_9 + W_9 = W_9 + W_8$$

Les supports de stockage

1. Les supports de stockage

Les supports de stockage sont des périphériques d'E /S sur lesquelles on peut effectuer les opérations de lecture/écriture. Leur rôle est la sauvegarde et l'archivage de l'information requise pour le fonctionnement de l'ordinateur, puisque la MC ne peut les garder que temporairement.

Vu leur utilité et leur importance en tantque supports permanents d'informations, ces unités de stockage ont connu une évolution considérable que ce soit dans leur technologie de construction, leur capacité de stockage et la manière de les raccorder à la machine pour garantir un accès rapide à l'information était sûrement la carte perforée, mais cette dernière a été très vite remplacée par des supports plus efficaces et plus performants, capables de stocker l'information en plus grande quantité, de manière plus durable et plus sure. Il s'agit des supports magnétiques (bande, disque, …).
L'évolution de la technologie a encore touché ce domaine et d'autres techniques ont vu le jour concernant la fabrication de ces supports, ce qui fait que sur le marché actuel, on trouve un nouveau type de support : les supports optiques.

2. Les supports magnétiques

Qu'il s'agisse de bande ou de disque, le principe d'enregistrement sur un support magnétique est le même, il se base sur les effets de magnétisation d'une surface recouverte d'une substance magnétisable.

2.1 Principe de l'enregistrement magnétique

Les supports magnétiques sont recouverts d'une substance magnétisable (oxyde de fer) composée de micro-cellules. Chaque micro-cellule peut être magnétisée par un courant électrique.
Les opérations de lecture et l'écriture sont réalisées à l'aide d'une teste constituée d'un bobinage électromagnétique enroulé sur un entrefer qui permet de magnétiser la surface du disque. Le sens de l'alimentation indique la valeur du bit enregistré.

2.2 Evolution des supports magnétiques

Bien que le principe d'enregistrement sur les supports magnétiques soit resté le même, plusieurs technologies de mise en oeuvre ont été élaborées et ce dans le but d'en améliorer l'efficacité, la capacité et le temps d'accès à l'information.
Le premier support magnétique utilisé était la bande magnétique et un peu plus tard, les disques magnétiques qui ont contribué à la révolution de l'ordinateur et de l'informatique en général.

3 Les bandes magnétiques

Une bande magnétique se présente sous la forme d'un ruban en plastique souple recouvert d'une couche magnétisable (oxyde de fer) et enroulée autour d'un support en plastique. Les bandes magnétiques sont exploitées par un dispositif spécial appelé dérouleur de bandes.

3.1 Stockage des données sur la bande magnétique : la bande magnétique est organisée en 7 ou 9 pistes. Chaque octet est représenté verticalement tel que chaque bit est stocké sur une piste. Le 9ème bit est un bit de parité.

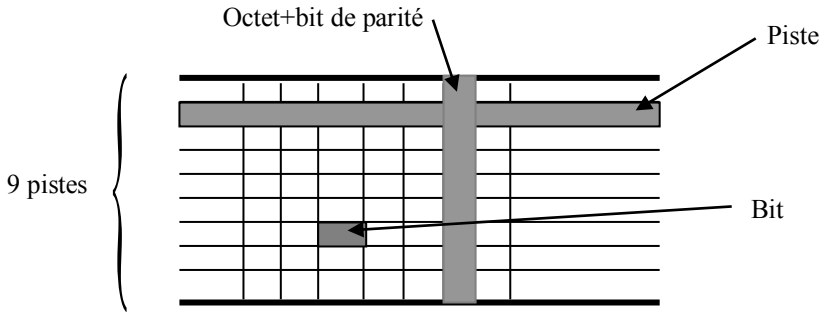


Schéma : Organisation de la bande magnétique en pistes

- Les informations sont enregistrées sur la bande sous forme de blocs, séparés par des zones non magnétisées appelées *espaces inter-blocs* ou *Gaps* selon une *densité d'enregistrement* donnée.

- La *densité d'enregistrement* est égale au nombre de bits stockés par pouce. Elle est mesurée en bit per inch (bpi).

📖 **1 octet = 1 byte = 1 caractère = 8 bits**
📖 **1 pouce = 2,54 cm**

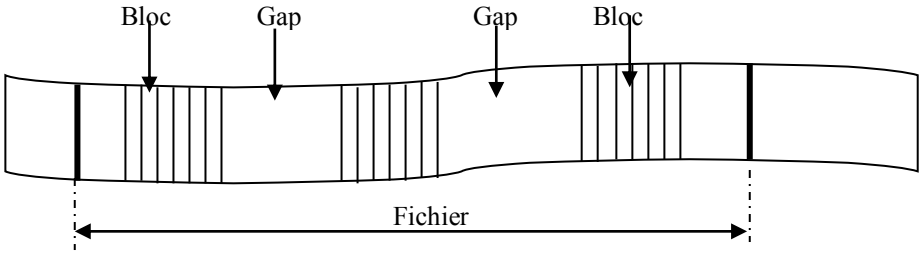


Schéma : organisation des données sous forme de blocs
Séparés par des blocs

Le stockage des enregistrements physiques sur les supports de stockage, de façon générale, peut s'effectuer de deux manières différentes :

- enregistrement groupé
- enregistrement non groupé

Cette notion de groupage (ou blocage) traduit le fait qu'un enregistrement physique contienne un ou plusieurs enregistrements logiques. Cela traduit par un paramètre important qui est *le facteur de groupage (ou facteur de blocage)*, telque :

$$F=N_{el} / N_P$$

N_{el} : nombre d'enregistrements logiques
N_P : nombre d'enregistrements physiques

Ainsi, on parle d'enregistrement groupé (ou bloqué) lorsque F=n c'est à dire qu'un enregistrement physique (ou bloc) contient plusieurs enregistrements logiques.
On parle d'enregistrement non groupé lorsque F=1 c'est à dire qu'un enregistrement physique contient un seul enregistrement logique.

a- Intérêt d'enregistrement bloqué : l'enregistrement bloqué offre plusieurs avantages, les plus importants sont :
- ✓ Gagner du temps lors des opérations de L/E.
- ✓ Gagner de l'espace sur le support

b- Choix d'un facteur de blocage : le facteur de blocage est limité par des contraintes physiques liées à la machine. Parmi ces contraintes, on cite :
- ✓ La taille du buffer d'E/S alloué en mémoire. En effet, le système d'exploitation ne peut manipuler que la quantité d'information qui peut tenir dans un buffer.
- ✓ Le nombre de buffer qui est alloués à un même fichier. En effet, le système d'exploitation permet d'allouer plusieurs buffers à un même fichier.

3.2. Exploitation d'une bande magnétique : pour une exploitation de la bande magnétique, il convient de connaître deux paramètres essentiels :
- L'espace réel de stockage
- Le temps nécessaire à la manipulation (L/E) du fichier sur la bande

a. Détermination de l'espace de stockage : pour déterminer l'espace nécessaire au stockage d'un fichier, il faut répondre à ces deux questions :
- ✓ Quel est l'espace E_b nécessaire au stockage d'un enregistrement physique ?
- ✓ Quel est l'espace E_f nécessaire au stockage de tout le fichier ?

Pour répondre à ces questions, on va considérer les deux cas (F=1) et (F=n)

Cas1 : F=1

i. L'espace E_b nécessaire au stockage d'un seul enregistrement physique :
On a **E = T/D**
$E_b = T_{el} / D + E_{ib}$
T_{el} : taille d'un enregistrement logique
D : densité d'enregistrement

ii. L'espace E_f nécessaire au stockage de tout le fichier :
$E_f = N*E_b$
N : nombre d'enregistrements
E_b : espace nécessaire au stockage d'un seul enregistrement physique + espace inter-bloc

Cas2 : F=n

i. L'espace E_b nécessaire au stockage d'un seul enregistrement physique
$E_b = F*T_{el}/D + E_{ib}$

ii. L'espace E_f nécessaire au stockage de tout le fichier :
$E_f=N_b*E_b$
N_b : nombre de bloc telque : $N_b=N_{ef}/F$

Conclusion : la formule générale permettant de déterminer l'espace réel de stockage d'un fichier sur une bande magnétique et la suivante :

$$E_f = \frac{N_{ef}}{F} * \left(E_{ib} + F * \frac{T_{el}}{D} \right)$$

N_{ef} : nombre d'enregistrements logiques dans le fichier
F : facteur de blocage
E_{ib} : espace inter-blocs
T_{el} : taille d'un enregistrement logique
D : densité d'enregistrement

Exemple :
On considère un fichier de 15000 enregistrements de 200 caractères chacun, on veut calculer l'espace nécessaire à son stockage pour F=1 puis F=20.
Densité d'enregistrement=2650bpi
Espace inter bloc=0,3pouces

1. pour F=1
   - L'espace E_b nécessaire au stockage d'un seul enregistrement physique :
     $E_{el}=T_{el}/D = 200/2650$
     $E_b=E_{el} + E_{ib} =200/2650+0,3 = 0,37547$pouces
   - L'espace E_f nécessaire au stockage de tout le fichier:
     $E_f = N_{el} * E_b = 15000*0,37547 = 5632,07547$pouces
2. pour F=20
   - L'espace E_b nécessaire au stockage d'un bloc
     $E_b = F*E_{el} + E_{ib} =20 * 200/2650 + 0,3=1,81$pouces
   - L'espace E_f nécessaire au stockage de tout le fichier :
     $E_f = N_{el}/F * E_b = 15000/20 * 1,81 =1357,5$pouces

b. Détermination du temps de traitement (L/E) d'un fichier sur bande : on peut calculer le temps de traitement d'une bande magnétique (on exclue le temps de traitement UC) :

i. Calcul la vitesse de défilement de la bande :$V_B = V_t/D$ telque :
V_B : vitesse de défilement de la bande mesurée en p/s
V_t : vitesse de transfert mesurée en c/s
D : densité d'enregistrement mesurée en bpi

ii. Calcul du temps nécessaire à la lecture d'un bloc : $t_b=T_b/V_t$ telque :
t_b : temps de lecture d'un bloc mesuré en seconde
T_b : nombre d'octets par bloc (taille d'un bloc)
V_t : vitesse de transfert mesurée en c/s

iii. Calcul du temps nécessaire au défilement d'un espace inter-bloc : $t_{ib}=T_{ib}/V_t$ ou $t_{ib}=E_{ib}/V_B$ telque :
t_{ib} : temps de défilement d'un espace inter-bloc mesuré en seconde
E_{ib} : espace inter-bloc mesuré en pouce
V_B:vitesse de la bande mesurée en p/s

Conclusion : le temps t_f nécessaire à la lecture d'un fichier est le suivant :
$t_f= N_b * (t_b + t_{ib}) = N_b * (T_b/V_t + E_{ib}/V_B)$, telque :
N_b : le nombre de blocs dans le fichier et $N_b =N_{ef}/F$

Exemple :
On considère un fichier de 15000 enregistrements de 175 caractères chacun, stocké sur une bande magnétique. Avec les informations suivantes, on voudrait calculer le temps de lecture de la bande.
Vitesse de transfert du déroulement :V_t =320000 o/s
Densité d'enregistrement : D=6250bpi
Espace inter bloc :E_{ib}=0,3p
Facteur de blocage F=30

Pour calculer le temps de lecture du fichier, on procède comme suit :

1. Calculer la vitesse de la bande :
   $V_B = V_t / D = 320000/6250 = 51,2p/s$
2. Calculer le temps de lecture d'un bloc
   $t_b = T_b / V_t = F*T_{el} /V_t = 30*175/32000 = 0,016s$
3. Calculer le temps de lecture de l'espace inter-bloc :
   $t_{ib} = T_{ib} / V_t = (T_{ib}/D) / (V_t /D) = E_{ib} / V_B = 0,3 / 51,2 = 0,0057s$

Ainsi, le temps nécessaire à la lecture du fichier complet se calcule comme suit:
$t_f = N_b * (t_b + t_{ib}) = 15000/30 * (0,016+0,0057) = 10,85s$

3.3 Caractéristiques d'une bande magnétique : la bande magnétique est caractérisée par un paramètre essentiel : l'accès séquentiel aux informations stockées : pour accéder à un bloc particulier, on est obligé de passer par tous les blocs qui le précédent. Ce paramètre joue en sa défaveur, puisque cela génère une lenteur et une perte de temps durant les accès aux données.

Avantages

- Ce sont des périphériques standard utilisables sur la plupart des systèmes.
- Elles permettent l'archivage des informations pour une longue durée.

Inconvénients

- Lenteur d'accès à l'information causé e par l'accès séquentiel
- Le montage et le démontage des bandes sur le dérouleur nécessitent un opérateur.

📖 Actuellement, les bandes magnétiques sont quasiment remplacées par des supports beaucoup plus performants qui offre à la fois une grande capacité de stockage et un accès rapide (accès aléatoire) aux informations.

4 Les disques magnétiques

Contrairement aux bandes magnétiques caractérisées par l'accès séquentiel aux données enregistrées, les disques magnétiques se distinguent par un accès direct, ce qui rend leur utilisation plus avantageuse.

Un disque magnétique est constitué par une plaque circulaire en aluminium ou en plastique de diamètre variable selon le type, recouverte d'une mince pellicule d'oxyde magnétisable.

On distingue généralement deux types de disques magnétiques : les disques durs et les disques souples (disquettes).

4.1 le disque dur : le disque dur se présente sous la forme d'un boîtier hermétique à l'intérieur duquel se trouve une pile de plateaux ou disques magnétiques superposés et regroupés autour d'un même axe. Chaque disque possède deux faces : une face supérieure et une face inférieure. A chaque face est associée une tête de L/E fixée sur un bras mobile (deux têtes pour chaque disque).

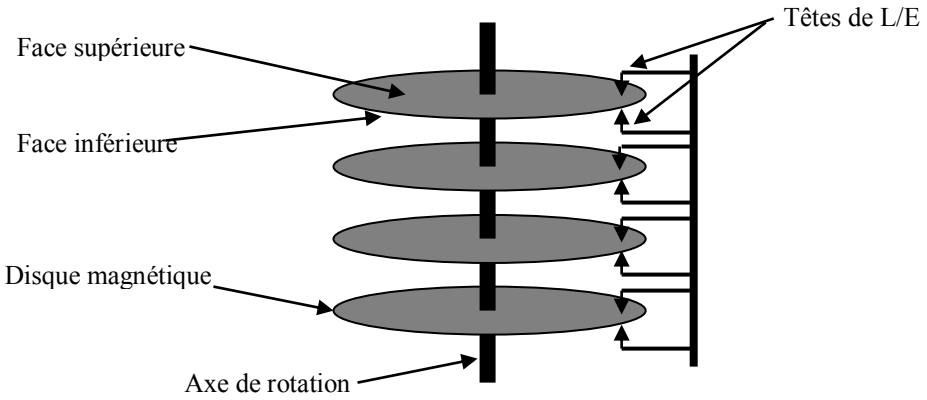


Schéma : disque dur de 4 plateaux et 8 têtes

a. Organisation du disque : les informations sont rangées dans les secteurs sous forme de bits en série. Dans un secteur, on distingue trois zones chacune réservé à des taches bien précises :

- *Zone de données utiles :* sert à recevoir les données à stocker sur le secteur.
- *Zone repérage :* elle contient le numéro du secteur.
- *Zone de contrôle :* contient un groupe de bits permettant de contrôler les données contenues dans la zone de données utiles.

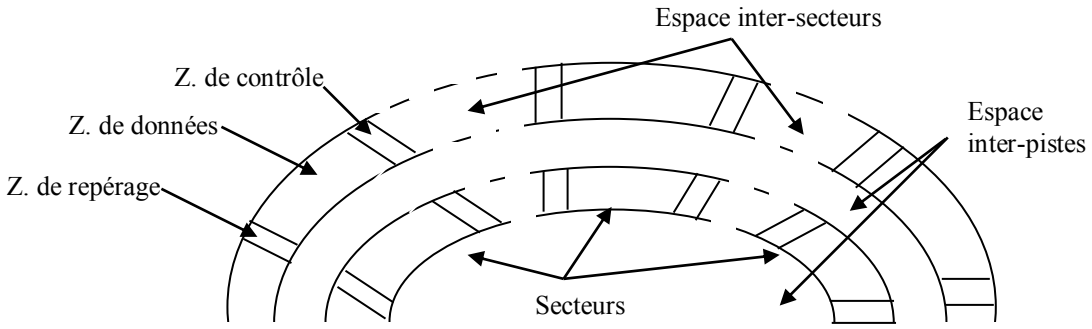


Schéma : organisation des secteurs sur les pistes

🕮 Le nombre de têtes, le nombre de cylindres et le nombre de secteurs que comporte un disque est appelé sa géométrie.

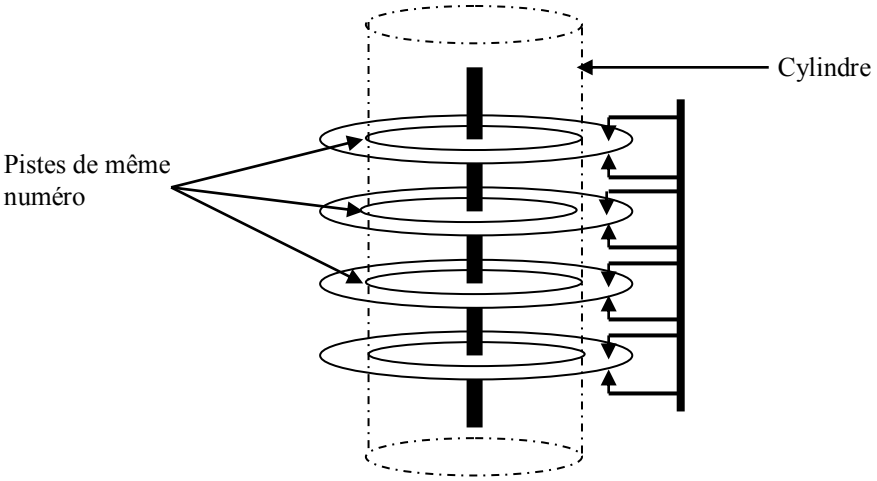


Schéma : exemple d'un cylindre

b. Densité d'enregistrement sur un disque : de manière générale, la densité est donnée par la formule suivante : $D=N_b/L$, telque :
N_b : nombre de bits (ou nombre de bytes)
L : longueur en pouces
La densité d'enregistrement linéaire par piste est calculé comme suit :
$D_L=N_{bp}/L$, telque :
N_{bp} : nombre de bits par piste (ou nombre de bytes par piste)
L : longueur d'une piste en pouce
Exemple : Soit un disque avec 64 secteurs par piste. La taille d'un secteur étant de 4096 octets. Si on considère que la longueur d'une piste L_P = 37,68cm, la densité d'enregistrement linéaire serait :
N_{bp} = 64*4096 = 262144 octets
Donc $D_L = N_{bp} / L_P$ = 262144/37,68 = 6957bpcm

c. Capacité d'un disque dur : elle désigne la quantité d'informations qu'un disque peut contenir mesurée en octet.
$C_d=C_s*N_{sp}*N_c*N_t$ telque :
C_s : capacité d'un secteur
N_{sp} : nombre de secteurs par piste
N_c : nombre de cylindres
N_t : nombre de têtes
Exemple : Considérons un disque dont la géométrie est la suivante : 14tetes, 723cylindres et 51 secteurs, la capacité d'un secteur étant de 512octets.
Donc, $C_d = C_s * N_{sp} * N_c * N_t$ = 512 * 51 * 723 * 14 = 258111ko

d. Débit de transfert : le début de transfert représente la quantité d'informations transférée par unité de temps. Il se mesure en o/s.

Exemple : Si pour transférer 512 octets du disque vers la mémoire centrale nécessite 5ms, le débit sera calculé comme suit :

512c ⟶ 5ms
D ⟶ 1s
D = 512/0,005 = 102400c/s = 100*1024c/s=100Ko/s
Donc, le débit de transfert est de 100Ko/s

e. Les opérations L/E : les informations stockées sur le disque sont identifiées par des adresses. Une adresse est constituée d'un numéro de tête, d'un numéro de cylindre et d'un numéro de secteur. Ainsi, pour lire une information sur le disque, l'UC fixe au contrôleur du disque la quantité d'informations à lire et lui indique l'@ correspondante sur le disque. A ce moment là, le contrôleur va effectuer les opérations suivantes :

- Il envoie les signaux nécessaires au positionnement de la tête à l'@ indiquée.
- Quand la tête est positionnée sur le secteur concerné, le disque envoi au contrôleur les informations lues en série (bit par bit).
- Le contrôleur du disque regroupe ce flot de bits et forme les mots destinés au système.

🕮 Presque le même processus est repéré pour l'écriture de données sur le disque. L'UC envoi au contrôleur les informations à stocker bloc par bloc et précise l'@ à partir de laquelle va s'affecter l'opération d'écriture. Le contrôleur va commander le positionnement de la tête et transmet les données bit par bit.

Pour accéder à une information stockée sur disque, un nombre d'opérations élémentaires sont exécutées. Chacune de ces opérations va nécessiter un certain temps pour son exécution. Le totale de ces temps constitue ce qu'on appelle temps d'accès à l'information, c'est à dire, le temps qui s'écoule entre l'ordre venant du contrôleur pour le positionnement de la tête de L/E jusqu'au transfert complet d'un secteur.

Ainsi, le temps d'accès à une information est égal à la somme des temps suivants :

- ✓ Temps nécessaire pour déplacer la tête pour la positionner sur le bon cylindre (ou temps de recherche).
- ✓ Temps de latence (ou délai de rotation) qui correspond au temps de rotation pris par le disque pour amener le bon secteur sous la tête.
- ✓ Temps nécessaire au transfert d'un secteur
- ✓ Temps du contrôleur qui est le temps pris par le contrôleur du disque pour achever l'opération d'E/S.

💣 Le disque peut faire au minimum 0 tours et au maximum 1 tour pour atteindre le bon secteur ce qui fait que le délai moyen de rotation est égal à un demi tour.

Exemple1 : si on considère qu'un disque tourne à 3600tours par minute, le délai moyen de rotation sera calculé comme suit :

3600tours ⟶ 1 minute
1/2tours ⟶ t minute
t = 0,5/3600 = 0,000138mn = 0,0083s = 8,3ms

Ainsi, le temps de rotation moyen pour atteindre le bon secteur est 8,3ms

Exemple2 : on veut calculer le temps moyen nécessaire pour lire ou écrire un secteur de 512octets pour un disque. Pour cela nous avons les informations suivantes :
Temps de recherche moyen = 9ms
Débit de transfert = 4Mo/s
Vitesse de rotation = 7200 tours/mn
Temps du contrôleur = 1ms
Le temps moyen d'accès = temps de recherche moyen + temps de rotation moyen + temps de transfert + temps du contrôleurs.

- le temps de rotation = 0,5/7200 = 4,15ms
- le temps de transfert 4Mo ⟶ 1s
  512c ⟶ ts
  t = 512 / (4*1024*1024) = 0,000122s = 0,122ms

Donc, le temps moyen d'accès = 9 + 4,15 + 0,122 + 1 = 14,272ms

📖 On a :
1 bit = 0 ou 1
1 octet = 1 caractère = 1 byte = 8 bits
1 Ko = 1024 octets
1Mo = 1024 Ko
1 Go = 1024 Mo

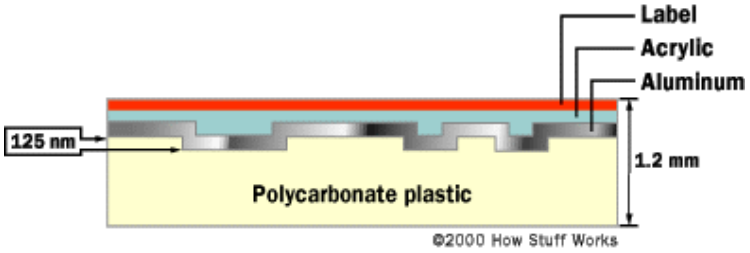
4.2 Le disque souple (disquette) : C'est un disque magnétique constitué d'un seul plateau en plastique souple enfermer à l'intérieur d'un mince boîtier de protection.

5. Le disque optique

Au début, les seuls supports utilisés pour stocker l'information étaient (et sont encore) les supports magnétiques : disque dur, disque souple. Ces dernières années, de nouveaux supports ont vu le jour, il s'agit des disques optiques (ou CD) dont les avantages majeurs sont leur grande capacité de stockage et leur amovibilité.
Exemple : CD ROM (Compact Disk Read Only Memory)
DVD (Digital Versatile Disk).

5.1 CD-ROM : Il utilise des disques portatifs de grande capacité au format pratique, de plus en plus utilisé pour la vente de logiciels. Il remplacera le lecteur de disquettes.
Le CD-ROM (Compact Disc - Read Only Memory) est un disque optique de 12 cm de diamètre et de 1mm d'épaisseur, permettant de stocker des informations numériques, c'est-à-dire correspondant à 650 Mo de données informatiques (correspondant à 300000 pages dactylographiées) ou bien jusqu'à 78 min de données audio.

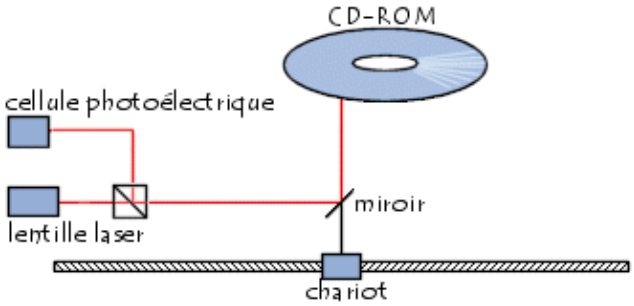
a. La composition d'un CD-ROM : Le CD est constitué de matière plastique, recouvert d'une fine pellicule métallique d'aluminium sur une des faces. Les pistes sont gravées en spirales, ce sont en fait des alvéoles d'une profondeur de 125nm et espacées de 1,6µ. Ces alvéoles forment un code binaire, une alvéole correspond à un 0, un espace à un 1.



Exemple : prenons la séquence suivante : 110010101. Celle-ci correspond sur le CD-ROM à deux espaces, deux trous, un espace, un trou, un espace, un trou, un espace.

On a ainsi une séquence binaire que le lecteur parcourt grâce à un laser ; celui-ci est réfléchi lorsqu'il rencontre un espace, il ne l'est pas lorsqu'il rencontre une alvéole.

b. Le lecteur de CD-ROM : C'est une cellule photoélectrique qui permet de capter le rayon réfléchi, grâce à un miroir semi-réflechissant comme expliqué sur le dessin suivant :

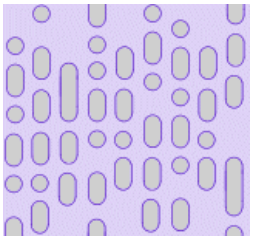


Un chariot permet de déplacer le miroir de façon à pouvoir accéder au CD-ROM en entier.
Il est ainsi possible de stocker sur ce support des musiques, des images, des vidéos, du texte et tout ce qui peut être enregistré de façon numérique.

c. Ses caractéristiques : Le lecteur CD-ROM est caractérisé :

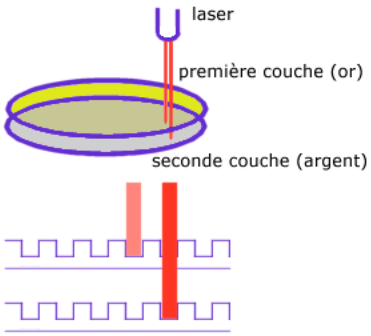
- Par sa vitesse : celle-ci est calculée par rapport à la vitesse d'un lecteur de CD-Audio (150 Ko/s). Un lecteur pouvant atteindre la vitesse de 3000 Ko/s sera caractérisé de 20X (20 fois plus vite qu'un lecteur 1X)
- Par son temps d'accès, on défini le temps moyen que le lecteur prend pour atteindre une partie du CD à une autre.

5.2. DVD-ROM : Le DVD-ROM (Digital Versatile Disc - Read Only Memory) est une variante du CD-ROM dont la capacité est largement plus grande. En effet, les alvéoles du DVD sont beaucoup plus petite (0,4µ et un espacement de 0.74µ), impliquant un laser avec une longueur d'onde beaucoup plus faible.



Les DVD existent en version "double couche", ces disques sont constitués d'une couche transparente à base d'or et d'une couche réflexive à base d'argent. Dans le but de lire ces deux couches, le lecteur dispose d'un laser à deux intensités :

- une intensité faible ; le rayon se réfléchit sur la surface dorée
- une plus grande intensité permet au rayon de traverser la première couche et de se réfléchir sur la surface argentée.



Il existe 4 types de DVD différents :

Type de support	Capacité	Temps musical équivalent	Nombre de CD équivalent
CD	650Mo	1h18 min	1
DVD simple face simple couche	4.7Go	9 h 30	7
DVD simple face double couche	8.5Go	17 h 30	13
DVD double face simple couche	9.4Go	19 h	14
DVD double face double couche	17Go	35 h	26

📖 L'intérêt du DVD touche en priorité le stockage vidéo qui demande beaucoup plus d'espace. Un DVD de 4,7 Go permet de stocker plus de deux heures de vidéo compressées en MPEG-2 (Motion Picture Experts Group), un format qui permet de compresser les images tout en gardant une très grande qualité d'image.