

N° d'ordre :.....

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Akli Mohand Oulhadj de Bouira

Faculté des Sciences et des Sciences Appliquées
Département d' *INFORMATIQUE*



Polycopié de cours

En : Mathématique et Informatique

Spécialité : Informatique

Par : Dr. Zahira CHOUIREF

*Electronique et Composants des
Systèmes Informatique*

Année : 2017/2018



UNIVERSITE DE BOUIRA

FACULTE DES SCIENCES ET DES SCIENCES APPLIQUEES

Mathématique et Informatique



Electronique et Composants des Systèmes Informatique

Présenté par
- Z. CHOUIREF -

Cours
Licence Informatique
1^{ère} année

DR. Z. CHOUIREF 2017/2018

*« Les grands esprits discutent des idées.
Les esprits moyens discutent des événements.
Les petits esprits discutent des gens. »*
— Eleanor Roosevelt

*« La seule révolution possible, c'est d'essayer de s'améliorer soi-même, en espérant que
les autres fassent la même démarche.
Le monde ira mieux alors. »*
— Georges Brassens



Table des matières

Introduction Générale -----	5
Chapitre 1 : Introduction au Système Informatique -----	6
1. <i>Définitions</i> -----	7
1.1. Information -----	7
1.2. Informatique-----	7
1.3. Programme -----	7
1.4. Ordinateur -----	8
2. <i>Système Informatique</i> -----	8
2.1. Définition-----	8
2.2. Principe de fonctionnement-----	8
2.3. Exemples -----	9
2.4. Composants de Système Informatique -----	10
Chapitre 2 : Composants matériels HARDWARE -----	11
1. <i>Introduction</i> -----	12
2. <i>Formes d'un PC</i> -----	12
3. <i>Composants matériels "Hardware"</i> -----	12
3.1. L'Unité centrale -----	12
3.1.1. Microprocesseur " μP " -----	13
3.1.2. Mémoire centrale-----	13
3.2. Les périphériques-----	13
3.2.1. Périphériques d'entrée (de commande)-----	14
3.2.2. Périphériques de Stockage-----	14
3.2.3. Périphériques de sortie (de visualisation)-----	15
3.2.4. Autres périphériques -----	15
3.2.5. Dispositifs d'énergie et de liaison -----	15
Chapitre 3 : Composants logiciels SOFTWARE -----	18
1. <i>Introduction</i> -----	19
2. <i>Définitions</i> -----	19
2.1. Instruction -----	19
2.2. Programme -----	19
2.3. Processus -----	20
2.4. Logiciel-----	20
2.5. Progiciel-----	20
3. <i>Composants Logiciels "Software"</i> -----	20
3.1. Système d'exploitation-----	20
3.1.1. Principe de fonctionnement -----	21

3.1.2. Les principales fonctions d'un SE-----	21
3.1.3. Principaux composant d'un SE :-----	22
3.2. Pilotes de périphériques -----	24
3.3. Différents logiciels et applications-----	24
3.1.4. Catégorisation fonctionnelle :-----	24
3.1.5. Catégorisation commerciale :-----	25
Chapitre 4 : Codage et représentation de l'information -----	28
1. Introduction-----	29
2. Représentation de l'information -----	30
3. Les unités de mesure en Informatique-----	30
3.1. Le bit-----	30
3.2. L'Octet-----	31
3.3. Le débit-----	31
3.4. Le Hertz -----	32
4. Systèmes de numérotation -----	32
4.1. Définition de base-----	33
4.2. Représentation des nombre entiers -----	33
4.3. Représentation des nombre fractionnaires -----	33
4.4. Transcodage (Conversion de bases)-----	34
4.4.1. Base décimale vers autres bases-----	34
4.4.2. Base Binaire vers autres bases-----	36
4.4.3. Base octale vers autres bases-----	37
4.4.4. Base Hexadécimale vers autres bases -----	37
4.4.5. Conversion d'un nombre réel (fractionnel)-----	38
4.5. Opérations arithmétiques -----	39
5. Codage de l'information-----	40
5.1. Typologie de l'information-----	40
5.2. Débordement (Overflow)-----	41
5.3. Codage des informations numériques-----	41
5.3.1. Codage des entiers non signés -----	41
5.3.2. Codage des entiers signés -----	41
5.3.3. Codage des nombre fractionnaires-----	43
5.4. Codage des caractères-----	46
5.4.1. Le code ASCII-----	46
5.4.2. Le code ASCII étendu -----	47
5.4.3. Le code UTF-8 -----	48
5.5. Alphabet, mots et langage -----	49
Chapitre 5 : Algèbre de BOOLE -----	50
1. Introduction-----	51

2. Algèbre de BOOLE -----	51
2.1. Définition-----	52
2.1.1. La fonction - ET - -----	52
2.1.2. La fonction - OU -----	52
2.1.3. La fonction - NON -----	52
3. Fonctions logiques et tables de vérité -----	52
3.1. Fonction logique-----	54
3.2. Priorité des opérateurs-----	54
3.3. Lois fondamentales de l'Algèbre de Boole-----	55
3.3.1. L'opérateur OU-----	55
3.3.2. L'opérateur ET-----	55
3.3.3. L'opérateur NON -----	55
3.3.4. Distributivité-----	55
3.3.5. Autres relations utiles -----	56
3.3.6. Dualité de l'algèbre de Boole-----	56
3.3.7. Théorème de DE-MORGANE-----	56
3.3.8. Les opérateurs NAND et NOR sont des opérateurs universels -----	56
3.4. Les portes logiques-----	57
3.4.1. Schéma d'un circuit logique (Logigramme)-----	57
3.5. Etude des fonctions logiques-----	57
3.5.1. Forme canonique-----	58
3.5.2. Forme canonique numérique -----	58
3.5.3. Passage à une Forme canonique-----	58
3.5.4. Simplification des fonctions logiques -----	61
3.5.5. La simplification Algébrique -----	61
3.5.6. La simplification Graphique-----	63
Conclusion Générale -----	68
Références bibliographiques-----	69
Annexe -----	71





Introduction Générale

Ce cours intitulé «Électronique et Composants des Systèmes Informatique/ ECSI», qui s'adresse principalement aux étudiants de 1ère année *Mathématique et Informatique (MI)*, présente de façon détaillée et avec des exemples simples les concepts des composants électroniques et systèmes informatique. Le contenu de ce cours est par conséquent est orienté particulièrement vers les étudiants qui auront pour parcours pédagogiques soit Informatique, Recherche Opérationnelle ou Mathématiques (Analyse, Algèbre, Statistiques et Probabilité), en l'élaguant au besoin. Ainsi, le système électronique, objet de l'étude de ce cours est l'ordinateur.

Ce cours comprend cinq chapitres avec des exercices complémentaires proposés en annexes. Chaque chapitre présente les connaissances essentielles permettant l'acquisition de compétences pratiques de base en ECSI.

En premier lieu l'étudiant est invité à découvrir l'environnement informatique en introduisant d'abord le fonctionnement de n'importe quel système informatique, les différents composants matériels qui le constitue avec une description des rôles et fonctionnalités de chacun de ses composants.

Ensuite nous présentons les différentes applications et les différents logiciels permettant l'exploitation convenable des ressources d'un système informatique, les spécifications de chaque type de logiciels ainsi que leur interopérabilité permettant d'assurer les différents services demandés par l'utilisateur.

Les deux derniers chapitres sont consacrés aux (i) bases fonctionnelles du traitement automatique de l'information permettant à l'étudiant de comprendre les principes et les techniques de représentation des différents types de données dans un système informatique, et de l'initier aux différentes opérations arithmétiques et logiques représentant les bases du traitement automatique de l'information, (ii) aperçu des méthodes algébriques et logiques permettant la conception et l'optimisation des solutions automatisées. Les étudiants seront initiés par introduction aux circuits logiques leur permettant d'établir une liaison entre les connaissances théoriques et la conception réelle et pratique des composants informatique.

Comme complément en annexe, cinq séries d'exercices qui touchent tous les chapitres terminent ce support de cours.

1

Chapitre 1 : Introduction au Système Informatique

1. Définitions

1.1. Information

C'est un ensemble de données ayant un sens. Elle nous permet d'avoir des renseignements sur une personne, un objet, un événement, etc.

Une information peut être présentée sous différentes formes:

- Un texte (ex.: le nom d'une personne).
- Une image (ex.: la photo d'une personne).
- Un son (ex.: la voix d'une personne).
- Une vidéo (ex.: une vidéo décrivant le comportement d'une personne).

1.2. Informatique

Dans le monde de la technologie ainsi que dans notre vie quotidienne, nous utilisons souvent le terme "**Informatique**", ça devient un synonyme du terme "**technologie**" !

L'informatique est l'abréviation des deux mots "**INFOR**mation et auto**MATI**QUE", définit la science de traitement automatique et rationnel de l'information. L'informatique est utilisée dans plusieurs domaines d'activité: scientifique, technique, industriel, éducatif, médical, etc.

Une machine ("**Ordinateurs**", Systèmes embarqués, Robots, Automates, etc.) est capable de remplir des tâches différentes selon les instructions qui lui sont adressées. Ces instructions, rédigées sous forme de programmes par les informaticiens, sont traitées par le matériel de l'ordinateur.

Cette *informatisation* permettra de réaliser un gain très considérable en *temps* et en *effort*.

1.3. Programme

Un **programme informatique** est un ensemble d'opérations destinées à être exécutées par un ordinateur. Un **programme source** est un code écrit par un informaticien dans un langage de programmation. Il peut être compilé vers une forme binaire¹, ou directement interprété. Un **programme binaire** décrit les instructions à exécuter par un microprocesseur sous forme numérique. Ces instructions définissent un langage machine.

¹ cette notion sera détaillée dans le chapitre 4.

1.4. Ordinateur

Est une machine automatique de traitement de l'information. Tous les **traitements** réalisés par un ordinateur se font via l'**exécution d'un programme** au niveau du microprocesseur ou CPU (Central Process Unit). Il peut recevoir des données en entrée «*fonction d'entrée*», effectuer sur ces données des opérations en fonction d'un ou de plusieurs programmes «*fonction de traitement*», et enfin fournir des résultats en sortie «*fonction de sortie*» tel que présenté dans la figure 1.

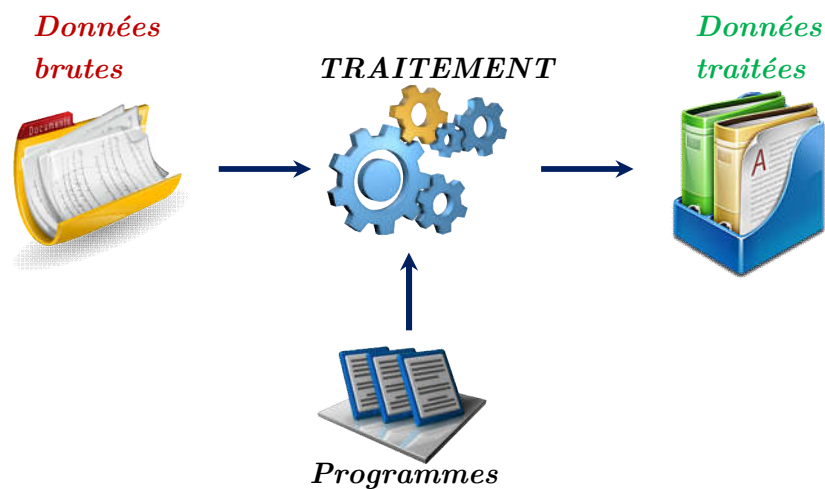


Figure 1 - Schéma fonctionnel du processus de traitement de l'information

2. Système Informatique

2.1. Définition

Un système informatique représente l'ensemble des moyens *d'acquisition, de restitution, de traitement et de stockage des données* dédié au traitement des informations. Son rôle principal est le traitement automatique de l'information en se servant des ressources matérielles (*Hardware*) et logicielles (*Software*).

2.2. Principe de fonctionnement

Un traitement informatique nécessite en général des informations en entrées (données) et délivre en sortie des résultats sous forme d'informations traitées, résultats de calculs, actions (sonneries, images, signaux, etc.).

2.3. Exemples

En se basant sur le principe de fonctionnement, plusieurs automatismes, dispositifs et systèmes peuvent être qualifiés en tant que système informatique. Citons quelques exemples en expliquant le fonctionnement de chacun de ces exemples :

Téléphone GSM : Que ce soit un téléphone mobile classique ou un Smartphone, il représente un système informatique proprement dit. Le traitement des appels téléphoniques ainsi que les messages (SMS) est un processus automatisé sur la base de la structuration des données de l'appelant (Numéro GSM, date et heure, contenu, destinataire...), qui sont regroupées et acheminées via le réseau GSM jusqu'à l'établissement de l'appel ou la réception de l'SMS.



Consoles de jeux : Ce sont des dispositifs électroniques dédiés spécialement aux jeux vidéo, leur fonctionnement repose sur l'exécution des programmes graphiques en fonction des commandes de l'utilisateur qui utilise une manette de jeux ou Joystick pour les transmettre à la console, qui à son tour exécute le programme approprié à la commande et affiche sur écran les résultats.



Xbox, Playstation et Nintendo représentent les marques des consoles les plus connues.

Pilote automatique : Les avions sont dotés de plusieurs systèmes informatiques dont le fonctionnement est très complexe. L'un de ces systèmes est le pilote automatique de l'avion. Ce dernier calcule un ensemble de facteurs (Vitesse, boussole, hauteur, position...) pour diriger et guider l'avion à sa destination. Il est même programmé pour pouvoir atterrir sans intervention humaine.



Satellite : Il intègre l'un des systèmes informatiques les plus avancés sur la planète. Selon sa fonction, le satellite traite énormément d'informations et les retransmet à une ou plusieurs stations de réception sur terre. Par exemple



les météos, qui représentent des informations relevées en continu, calculées ensuite pour en déduire l'état ultérieur de l'atmosphère, etc. D'autres satellites ont une mission de diffusion Radio et TV, télécommunication, surveillance aérienne, etc.

Robot : Utilisé souvent dans l'industrie, il représente un dispositif qui exécute des commandes instantanées, présélectionnées ou programmées d'avance pour faire des tâches précises selon sa fonction. Certains robots sont dotés d'une intelligence très avancée, pour pouvoir réagir avec le comportement humain en utilisant de la voix ou le mouvement.



2.4. Composants de Système Informatique

Pour assurer convenablement ses fonctions en termes de traitement automatique de l'information, un système informatique utilise à la fois les deux parties essentielles qui le constituent :

Partie HARDWARE : Représente l'ensemble des composants physiques dont le rôle principal est :

- L'acquisition de l'information et des commandes.
- L'exécution des différents traitements sur les informations acquises.
- L'envoi en sortie (affichage, etc.) les résultats du traitement.



Partie SOFTWARE : Représente l'ensemble de tous les programmes préinstallés ou ajoutés au système informatique, lui déterminant les instructions à exécuter, règles à respecter, ainsi que la structure et la manière dont les résultats seront transmis.



Ces deux parties seront détaillées dans les deux chapitres suivants.

2

Chapitre 2 : Composants matériels **HARDWARE**



1. Introduction

Dans tout ce qui suit, nous allons prendre le *PC (Personal Computer)* autrement dit "*Ordinateur Personnel*" comme exemple de référence du système informatique. Nous aborderons le rôle ainsi que les principales fonctionnalités de chacun de ses composants matériels.

2. Formes d'un PC

Les différents constructeurs des PC proposent plusieurs formes de PCs dédiés à plusieurs usages personnels et professionnels nous citons à titre d'exemple :

- **Le PC** de bureau classique ou habituel, dont la forme physique est composée d'une unité centrale, d'un écran et des périphériques de commande (Clavier et souris)
- **Le PC portable** : représentant une forme plus compacte du PC de bureau lui donnant l'option de mobilité en utilisant une batterie intégrée et lui assurant une autonomie d'énergie minimale de 90 mn.
- **All-in-one** : Autrement dit "Tout en Un", c'est une autre nouvelle forme de PC de bureau permettant une économie d'énergie et d'espace de travail car l'unité centrale et l'écran sont regroupés dans un seul cadre.
- etc.



3. Composants matériels "Hardware"

Tout ce qui est physique, ça concerne les circuits électriques, électroniques ainsi que les différents mécanismes. La partie *hardware* d'un PC est composée principalement de deux parties :

- Une unité centrale chargée d'effectuer les différents traitements ;
- Un ensemble de périphériques qui sont utilisés pour interagir avec la machine, pour sauvegarder, transférer, archiver, visualiser les données, etc.

3.1. L'Unité centrale

C'est à son niveau seulement que sont effectuées les différentes opérations arithmétiques et logiques du traitement automatique de l'information. Elle est composée de deux composants principaux :

3.1.1. Microprocesseur " μP "

Dit aussi CPU (Control Process Unit) ou "*Microprocesseur*", il représente le **cerveau** du PC, il est chargé d'exécuter toute les opérations de traitement des informations quelque soient leur nature.



3.1.2. Mémoire centrale

C'est la partie qui contient les programmes et les données qui seront traités par le microprocesseur. Il existe trois types de mémoires internes :

Mémoire vive (RAM-Random Access Memory) : Elle permet la lecture/écriture des données, elle contient les programmes et les informations *en cours du traitement ou d'exécution*. Les informations enregistrées sur la RAM sont perdues dès que le PC est mis hors tension.



On dit que la RAM est volatile.

Mémoire morte (ROM- Read Only Memory) : C'est une mémoire utilisée en lecture seule, elle contient les programmes de démarrage et les données de configuration de base du PC. Ceux-ci sont enregistrés une fois pour toutes dans cette mémoire et ne peuvent être ni modifiés ni effacés, même après coupure de l'alimentation électrique.

Mémoire Cache : Dite aussi "*antémémoire*" C'est une mémoire intégrée avec le *Microprocesseur*, elle est utilisée pour enregistrer temporairement des copies de données provenant d'une source (Disque dur / RAM) dont le temps d'accès est long comparant à la vitesse du CPU, afin de diminuer le temps d'un nouvel accès à ces données.

Le principe du cache est également utilisable en écriture, et existe alors en trois modes possibles : write-through, write-back et write-around.

3.2. Les périphériques

Ce sont des dispositifs matériels permettant d'assurer les échanges d'informations (ex.: le transport, le transfert, l'impression, la numérisation, la visualisation, etc.) en entrée et en sortie entre l'ordinateur et l'extérieur ou de stocker de manière permanente des informations.

Par fonction, les périphériques sont classés comme suite :

3.2.1. Périphériques d'entrée (de commande)

Des dispositifs permettant de commander la machine en lui transmettant les commandes à exécuter, tel que le clavier, la souris, la manette de jeux, le joystick (jeux), il existe ainsi d'autres tel que l'écran tactile, tablette de commande, etc.



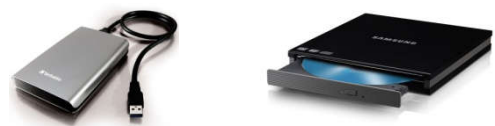
3.2.2. Périphériques de Stockage

Comme la mémoire vive perd les informations après arrêt de l'ordinateur, il est donc important d'utiliser des mémoires qui permettent de conserver, d'une façon permanente ces informations.

Cela est le rôle des périphériques de stockage : comme le Disque dur, les lecteurs/Graveurs Optique (CD, DVD, CD/DVD-RW), les clés USB (Flashdisk), les lecteurs ZIP, le lecteur de cartes mémoires, etc.



Certains périphériques de stockage sont intégrés dans la structure du PC alors que d'autres ont la caractéristique de mobilité tel que le *Flashdisk*, *Le lecteur ZIP*, *Le disque dur externe* et *le lecteur Optique (CD-DVD) externe*.



Remarque / *Les périphériques de type lecteurs utilisent des supports pour la sauvegarde des informations, données, etc. comme les CD/DVD-Rom, Disquette 3.5'', Disquette ZIP, carte mémoire, etc.*



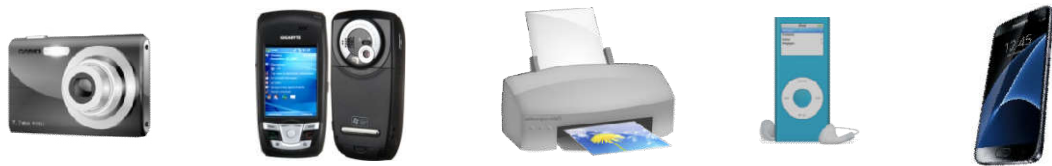
3.2.3. Périphériques de sortie (de visualisation)

Le périphérique principal d'affichage dans un PC est l'écran. La *carte graphique* est une unité d'échange. Il existe un autre périphérique d'affichage c'est le vidéoprojecteur (DataShow).



3.2.4. Autres périphériques

Ils existent plein d'autres périphériques qui peuvent être reliés à un PC, par exemple : Scanner, Imprimante, Appareils photos numérique, Lecteur MP3/MP4, Téléphone cellulaire, Modem Internet, Carte satellite, Carte TV, etc.



3.2.5. Dispositifs d'énergie et de liaison

Dans l'architecture du PC, il existe des dispositifs qui sont indispensable pour le fonctionnement du PC, à savoir :

A. La carte-mère : C'est le système nerveux du PC. Elle contient tous les connecteurs possibles pour le branchement et l'assemblage de n'importe quel composant du PC. Ainsi elle assure la liaison et l'interconnexion entre l'ensemble de ces composants.

Pour des raisons industrielles, elle intègre aussi quelques périphériques (*carte Graphique, carte réseau et carte son*).



En plus des emplacements (Socket, Connecteurs, Slots, Ports) disponibles sur la carte mère et qui permettent l'installation des mémoires RAM, Microprocesseur, Disque dur, les différents lecteurs, etc. La carte mère contient d'autres composants nécessaires pour le fonctionnement du PC, tels que :

Le Chipset / Il représente la plateforme centrale de la carte mère. Son rôle est de coordonner les échanges de données entre le processeur et les divers périphériques et composants du PC.

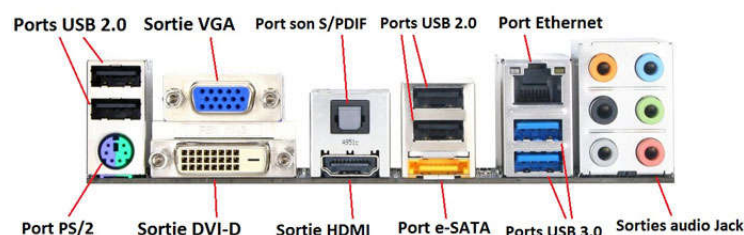
La ROM / (définie auparavant). C'est un circuit intégré sur la carte mère, contenant un programme appelé **BIOS** (*Basic Input Output System*), qui est le premier programme exécuté à chaque démarrage du PC. Ce dernier permet au PC de *booter* (démarrer) et d'initialiser (tester) les périphériques, charger le système d'exploitation (Windows, Linux...) dans la RAM avant de lui passer le relais.

La ROM est présente sur toutes les cartes-mères.

Le CMOS / (*Complementary Metal Oxide Semiconductor*) : Lorsque le PC est mis hors tension, le CMOS conserve l'heure, la date et tous les paramètres de configuration du PC, qui lui permettent de démarrer correctement une deuxième fois. Elle est alimentée par une pile installée sur la carte mère. Le CMOS est une mémoire lente mais qui consomme peu d'énergie.

Les ports d'entrée-sortie/ Un port sert à connecter des périphériques à l'ordinateur. Il existe plusieurs types dont voici quelques-uns :

- *Les ports séries* : pour la connexion de vieux périphériques DB-9 RS232 tels que le clavier et la souris.
- *Les ports PS1 et PS2* : (Personal System) utilisés pour connecter principalement le clavier et la souris.
- *Les ports parallèles LPT1* : pour la connexion, entre autres, de vieilles imprimantes (imprimante matricielle).
- *Les ports USB* : (Universal Serial Bus) pour la connexion de périphériques récents (Clavier, souris, Flashdisk, Bluetooth, etc.).
- *Les ports RJ45* (Registered Jack) : permet la connexion à un réseau informatique, Internet et autre.
- *Les ports VGA* (Video Graphics Array) : pour la connexion d'un moniteur (écran) à l'ordinateur, il existe aussi des ports DVI et HDMI pour les écrans à haute définition et qualité d'image.
- Les connecteurs IDE ou SATA 1 ou 2 pour la connexion de périphériques de stockage comme les disques durs et lecteur CD/DVD.
- Les connecteurs audio pour la connexion d'appareils audio comme des haut-parleurs ou un microphone.



Les connecteurs d'extension (Slots) / En plus des ports disponible pour la liaison des différents composants du PC, il existe des ports (Slots) supplémentaires, permettant l'ajout de nouveaux périphériques (*Carte d'extension*) tels que : Carte graphique (Non intégrée), Carte réseau, Carte FaxModem, Carte Son, Carte TV, Carte Satellite, etc. Dans l'objectif d'ajouter de nouvelles fonctionnalités ou d'améliorer les performances du PC, on peut aussi citer :

- *Connecteur ISA* : (Industry Standard Architecture) : permettant de connecter des cartes ISA.
- *Connecteur PCI* (Peripheral Component InterConnect) : permettant de connecter des cartes PCI, beaucoup plus rapides que les cartes ISA.
- *Connecteur AGP* (Accelerated Graphic Port): un connecteur rapide pour carte graphique. *Il est remplacé actuellement par le PCI Express.*
- *Connecteur PCI Express* (Peripheral Component InterConnect Express) : architecture de bus plus rapide que les bus AGP et PCI.

Les Bus / Chaque composant du PC possède une liaison directe ou indirecte avec les autres composants, cette liaison est réalisée par un ensemble de lignes appelées «**BUS**». Il existe plusieurs type de bus (Bus System FSB, Bus série, Bus parallèle, Bus USB, etc.)

Il existe des bus de données qui servent à transporter les différentes informations entre les composants, et des bus d'adresse pour la localisation des sources et destinations des informations.

Notons que la carte mère regroupe la plus grande partie des bus de liaison.

B. Le boîtier d'alimentation : Son rôle est d'alimenter en courant électrique les différents composants du PC, à savoir: Carte mère, Disque dur, Lecteurs Optiques et Lecteur de cartes mémoires.



3

Chapitre 3 : Composants logiciels **SOFTWARE**



1. Introduction

Tout système informatique comme le PC intègre une partie software qui représente la *partie logique*. Elle est constituée d'un ensemble de programmes qui définissent les règles de fonctionnement, l'ordre de traitement des données et bien d'autres fonctionnalités.

Pour se familiariser avec le domaine du software, nous présentons dans ce qui suit un ensemble de définitions de base très importantes des différents concepts Software.

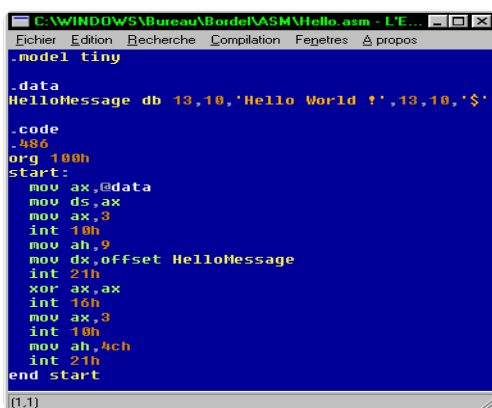
2. Définitions

2.1. Instruction

Les instructions définissent un ensemble d'opérations à exécuter par l'ordinateur. Elles sont écrites dans un *langage de programmation* approprié.

2.2. Programme

C'est un ensemble d'instructions codées (*écrites selon les règles définies par un langage de programmation*). Son rôle est de définir un ordre d'exécution de traitement des informations existantes dans le PC ou bien introduites via clavier ou souris ou autre périphériques. Dans l'objectif de réaliser une ou plusieurs tâches (*Opérations*) de calcul et de traitement. Il est aussi définit, dans un programme, la destination et la manière dont les informations traitées seront transmis à l'utilisateur (ordre d'affichage, envoi par mail, signalisation sonore ou lumineuse, etc.).



```
.model tiny
.data
HelloMessage db 13,10,'Hello World !',13,10,'$'
.code
.486
org 100h
start:
mov ax,@data
mov ds,ax
mov ax,3
int 10h
mov ah,9
mov dx,offset HelloMessage
int 21h
xor ax,ax
int 10h
mov ax,3
int 10h
mov ah,4ch
int 21h
end start
```

```
class Point {
public Point (int x, int y) { this.x = x; this.y = y; }
public void deplace (int dx, int dy) { x += dx; y += dy; }
public void affiche () { System.out.println ("Je suis en " + x + " " + y); }
protected int x, y;
}

class Pointcol extends Point {
public Pointcol (int x, int y, byte couleur) {
super(x, y); // obligatoirement comme première instruction
this.couleur = couleur;
}
public void affiche () {
super.affiche();
System.out.println ("et ma couleur est : " + couleur);
}
private byte couleur;
}

public class Poly {
public static void main (String args[]) {
Point p = new Point (8, 5);
p.affiche(); // appelle affiche de Point
Pointcol pc = new Pointcol (4, 8, (byte)2);
p = pc; // p de type Point, référence un objet de type Pointcol
p.affiche(); // on appelle affiche de Pointcol
p = new Point (5, 7); // p référence a nouveau un objet de type Point
p.affiche(); // on appelle affiche de Point
}
}
```

Figure 2 - Exemple de segments de programmes.

2.3. Processus

C'est *un programme en cours d'exécution* par l'ordinateur. Le terme processus est introduit pour faire la distinction entre un programme dans **son état brut** sous forme d'un *texte stocké dans un ou plusieurs fichiers* et **son état actif** où *les instructions sont chargées en mémoire vive (RAM) et en cour d'exécution avec les données à traiter*.

Lors de son exécution, le processus requiert un *espace d'adressage* en mémoire vive, avec des ressources telles que carte réseau, écran, imprimante, etc.

L'exécution d'un processus dure un certain temps, avec un début et une fin. Il peut être lancé (démarré) par un utilisateur par l'intermédiaire d'un périphérique (clique souris, etc.) ou bien par un autre processus.

2.4. Logiciel

Un logiciel est un programme d'une taille et complexité importante (grande), il peut intégrer plusieurs sous-programmes (*appelés aussi modules*). Son rôle est de réaliser des tâches adaptées aux besoins habituels ou spécifiques de l'utilisateur.

2.5. Progiciel

C'est un *petit logiciel*, il permet de réaliser des tâches qui ne sont pas très compliquées, mais elles sont très importantes. En général les progiciels offrent des alternatives aux utilisateurs par rapport aux services offerts par le système d'exploitation ou les logiciels d'application. Certains *progiciels* offrent plus d'option et de fonctionnalités que les logiciels habituels ou intégrés avec le SE.

Par exemple :

- On peut utiliser "*Notepad++*" pour la création de simples textes au lieu d'utiliser *Microsoft office Word*.
- L'utilitaire de gravure de CD et DVD "*Nero*" offre plus de possibilité que l'utilitaire intégré avec Windows.

3. Composants Logiciels "Software"

3.1. Système d'exploitation

C'est un type particulier de logiciel, chaque PC doit disposer d'un système d'exploitation pour fonctionner et servir les utilisateurs. Le *SE (Système d'Exploitation)* ou en anglais *OS (Operating System)* est le premier intervenant entre l'utilisateur et la

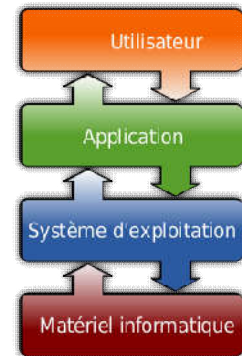
machine, il offre les outils et les accès lui permettant l'exploitation des ressources de la machine (clavier, écran, disque, imprimante, etc.).

Les SE les plus connus sont : Windows, Solaris, Linux, UNIX, Apple MacOS, etc.

3.1.1. Principe de fonctionnement

Le SE joue le rôle d'interface entre l'utilisateur et la machine (matériels). Dans ce cadre, l'utilisateur transmet ses besoins à la machine en utilisant les périphériques d'entrée (clavier, souris, etc.), via des applications (*programme*) appropriées comme un navigateur Web ou un lecteur média par exemple.

L'application utilisateur, reformule la demande de l'utilisateur sous forme d'un ensemble de commandes compréhensible par le système d'exploitation. Ce dernier invoque les modules (*Programmes/Applications*) nécessaires pour le traitement de sa demande, en incluant les "*pilotes*" de périphériques permettant l'interaction entre la machine (PC) et l'utilisateur.



3.1.2. Les principales fonctions d'un SE

Le SE tourne en permanence et contrôle la machine depuis son démarrage (boot). Il offre des fonctionnalités très importantes telles que:

1. La gestion de la mémoire centrale : Qui consiste à partager la mémoire vive entre plusieurs processus, et les données entre plusieurs utilisateurs. Le SE permet aussi d'exécuter des applications dont la taille (avec les données) est plus grande que la taille de la mémoire de la machine.

C'est le cas des jeux vidéo qui sont portés sur des DVD de 4.5 Go et qui peuvent être exécutés sur des PC à 2 Go de RAM.

2. L'exécution des commandes d'entrées-sorties : qui sont toujours lentes par rapport à la vitesse du Microprocesseur (CPU); et là se pose la question : comment synchroniser le CPU avec les périphériques d'entrées-sorties ?

3. La gestion du CPU : Cette fonction permet l'exécution de plusieurs programmes en même temps. Etant donné que le CPU est le cerveau de la machine et c'est lui seul qui a la possibilité de l'exécution des différents traitements de l'information, dans ce cas le SE intervient pour allouer le CPU aux applications exécutées simultanément.

Ainsi un utilisateur peut lire les informations sur le Web (*Navigateur*), écouter de la musique (*Lecteur média*), télécharger des documents (*Idman*), transférer des photos via Bluetooth (*Copie*), etc.

4. La gestion des mémoires secondaires (Disque dur) : On peut aussi appeler cette fonction "*Gestion de Fichiers*", pour la simple raison que l'ensemble des informations traitées par la machine sont stockées et regroupées dans des fichiers sur les mémoires secondaires (HDD, Flashdisk, etc.).

Cette fonction permet *d'organiser, enregistrer, protéger, chercher et retrouver* facilement les informations stockées. Ainsi le SE assure une gestion cohérente de données lors d'un accès simultané à une information (fichier), c'est le cas si un utilisateur désire supprimer un fichier (.pdf) alors que celui-ci est ouvert dans une application (Foxit reader / Adobe reader).

5. Fournir un environnement de travail à l'utilisateur : Cela est assuré par l'interprétation d'un langage de commande et l'enchaînement des travaux demandés, pour faciliter l'utilisation de la machine.

3.1.3. Principaux composant d'un SE :

1. Noyau et utilitaires : Le SE comporte un certain nombre de routines (sous-programmes). Les plus importantes constituent le noyau (*kernel*). Celui-ci est chargé en mémoire vive au démarrage de l'ordinateur et contient de nombreuses procédures nécessaires au bon fonctionnement du système. Il représente le cœur fonctionnel d'un SE. Le noyau d'un SE se compose de quatre parties principales :

- Le gestionnaire de tâches (ou des processus),
- Le gestionnaire de mémoire,
- Le gestionnaire de fichiers,
- Le gestionnaire de périphériques d'entrée-sortie.

Il possède également deux parties auxiliaires (des utilitaires) : le chargeur du SE et l'interpréteur de commandes IC.

2. Le gestionnaire de tâches : Sur un *système à temps partagé*, l'une des parties les plus importantes du SE est le gestionnaire de tâches (*Scheduler*) ou *ordonnanceur*. Sur un système à un seul processeur, il divise le temps en *laps* (*slices*, tranches).

Périodiquement, le gestionnaire de tâches décide d'interrompre le processus en cours et de démarrer (*ou reprendre*) l'exécution d'un autre, soit parce que le premier a épuisé son temps d'allocation du *processeur*, soit parce qu'il est bloqué (en attente d'une donnée quelconque nécessaire pour continuer).

3. Le gestionnaire de mémoire : La mémoire est une ressource importante qui doit être gérée avec prudence. Le gestionnaire de mémoire doit à tout moment connaître les parties libres et les parties occupées de la mémoire, allouer de la mémoire aux processus qui en ont besoin, récupérer la mémoire utilisée par un processus lorsque celui-ci se termine et traiter le va-et-vient (*Swapping*, ou *pagination*) entre le disque (*Mémoire virtuelle*) et la mémoire principale (RAM) lorsque cette dernière

ne peut pas contenir tous les processus. Ainsi l'indexation et la gestion de la *mémoire cache* fait partie des tâches du gestionnaire de mémoire.

4. Le gestionnaire de fichiers : Une des tâches fondamentales du SE est de masquer les spécificités des disques et des autres périphériques d'entrée-sortie et d'offrir aux utilisateurs un modèle agréable, organisé, cohérent et facile à gérer. Ceci se fait à travers la notion de fichier. Généralement le gestionnaire de fichiers fournit une interface graphique pour travailler avec les fichiers informatique. Les utilisations les plus communes sont : la création, l'ouverture, la visualisation, l'impression, la lecture, le renommage, le déplacement, la copie, la suppression, et la recherche de fichiers.

5. Le gestionnaire de périphériques : Le contrôle des périphériques d'entrée-sortie (E/S) de l'ordinateur est l'une des fonctions primordiales d'un SE. Ce dernier doit envoyer les commandes aux périphériques, intercepter les *interruptions*, traiter les erreurs, et retourner les résultats de communication avec le matériel aux utilisateurs du système. Il doit aussi fournir une interface simple et facile d'emploi entre les périphériques et le reste du système qui doit être, dans la mesure du possible, la même pour tous les périphériques, c'est-à-dire indépendante du périphérique utilisé.

6. Le chargeur du système d'exploitation : Lorsque l'ordinateur est mis sous tension, une suite d'actions est exécutée comme suite :

- L'exécution du BIOS, ce dernier se charge en mémoire RAM.
- Le BIOS initialise les périphériques et composants matériels,
- Le BIOS lance le **Pré-chargeur** du SE, un *petit programme* qui se trouve sur un secteur défini du disque dur.
- Le **Pré-chargeur** lance à son tour le **Chargeur du SE**.
- Ce dernier s'occupe du chargement du noyau et des différents gestionnaires et modules du SE dans la mémoire vive (RAM).

7. L'interpréteur de commandes : Le SE proprement dit est le processus maître qui permet de définir les appels système. Les programmes système tels que les éditeurs de texte, les compilateurs, les assembleurs, les éditeurs de liens et les interpréteurs de commandes ne font pas partie du SE.

```
OS Sys'thematic : liste des commandes disponibles sur notre Shell 00:00:00
clear          Efface la console
echo <on>     Si <on> vaut 1 : active l'affichage du retour clavier
              Si <on> vaut 0 : desactive cet echo
exit          Sortie du shell
Fichiers      Commandes de gestion des fichiers disponibles
help         Affiche cette aide
infotest     Information sur les tests ajoutés
kill <pid>   Termine le processus de pid <pid>
linfo       Informations sur les files de messages du processus
mkdir <nb>  Creation d'une file de <nb> messages
mdir <fid>  Suppression de la file de messages <fid>
mrecv <fid> Reception d'une valeur sur la file <fid>
msend <fid> Envoi d'une valeur sur la file <fid>
ps          Affiche l'état des processus du systeme
reboot     Redemarre le systeme
shell      Ouvre un nouveau shell
sleep <s>   S'endort pendant <s> secondes
switch <n> Le shell <n> devient actif
time      Nb d'IT horloge depuis le demarrage du systeme
test_all  Lance tous les tests
test <nom> Lance le test de nom <nom>
zombie_clear Libere les processus fils zombie du shell courant
Sys' shell 00
```

Cependant l'interpréteur de commandes (**Shell**) est souvent considéré comme en faisant partie. Sous sa forme la plus simple, l'interpréteur de commandes affiche une invite de commande pour que l'utilisateur saisisse et exécute les commandes qu'il veut.

3.2. Pilotes de périphériques

Un pilote informatique, souvent abrégé en *pilote* (*Driver*), est un programme informatique destiné à permettre au SE d'interagir avec un périphérique. En général, chaque périphérique a son propre pilote.

De manière simpliste, un pilote d'imprimante est un logiciel qui explique à l'ordinateur comment utiliser l'imprimante, la structure des données à lui transmettre ainsi que le protocole de communication. Sans pilote, l'imprimante, la carte graphique, l'interface Bluetooth ou la Webcam par exemple ne pourraient pas être utilisées. Certains SE comme Windows proposent leurs propres pilotes génériques (de base) censés fonctionner de manière satisfaisante avec la plupart des périphériques pour une utilisation courante.

3.3. Différents logiciels et applications

Il existe des milliers de logiciels, applications et utilitaires (*Petits logiciels*), qui sont conçus pour pouvoir les utiliser sur un PC. On peut les catégoriser selon leurs fonctions, ou leurs politiques de développement et de commercialisation, comme suite :

3.4. Catégorisation fonctionnelle :

1. **Logiciel de développement** : En général ce sont des langages de programmation permettant eu même la création et l'édition de différent logiciels et applications. Ils intègrent leurs propres éditeurs de code, et *compilateurs* (**Environnement de programmation**).



Ex : *Java, Delphi, VisualC++, etc.*

2. **Les suites bureautiques**: Elles représentent un ensemble de logiciels intégrés et rassemblés dans une suite, offrant plusieurs services d'automatisation des tâches de la bureautique (document, calcul, présentation, gestion de projets, publication, etc.)



Ex : *Microsoft Office : Word, Excel, PowerPoint, Publisher, etc.*
Open Office : Writer, Calc, Impress, Draw, etc.

3. **Les logiciels de traitements d'image** : Ce sont des logiciels un peu plus compliqués que ceux des suites bureautiques, parce qu'ils permettent d'effectuer des traitements (création, édition et modification) très complexes sur les images.



Ex : *Adobe Photoshop, etc.*

4. **Les logiciels de traitement de vidéo** : Logiciels d'édition, de modification et de création de vidéos.



Ex : *Adobe Aftereffect, Pinnacle studio, etc.*

5. Les logiciels de gestion des bases de données : Ce sont des logiciels utilisés pour la création et la gestion de bases de données.



Ex : *Oracle, PostgreSQL, ACSSSES, MySQL, etc.*

6. Logiciels de conception 3D : Ce sont en général des logiciels dédiés à un usage très spécifique comme les logiciels de développement, sauf que le rôle est la conception de modèles à Trois Dimensions (3D). Comme les vues architecturales (Maquettes 3D) ou des vues réalistes dédiées au développement des jeux vidéo, ou la réalisation de films en 3D.

Ex : *Maya3D, AutoCAD, Archicad, 3DS Max, CATIA, Blender, etc.*

7. Les jeux vidéo : Ce sont des applications de loisir, mais qui sont très complexes à développer ce qui est justifié par leurs taille qui peut y aller jusqu'à plusieurs Giga-octet (4 et 8 DVD-Rom).

Ex : *PES, FIFA, Call of Duty, Need For Speed, etc.*

On peut trouver aussi, différents types d'utilitaires (*Progiciels*), c'est le cas de :

- **Applications de téléchargement sur Web** (Internet): *Idman* (*Internet Download Manager*),
- **Les lecteurs média** (Winamp, VLC, JetAudio, PowerDVD, etc.),
- **Les navigateurs Web** (Opera, FireFox, Internet Explorer, Safari, etc.),
- **Les moteurs graphiques** (DirectX, OpenGL, etc.),
- **Les antivirus** (AVG, Kaspersky, Avast, etc.),
- **La messagerie électronique** (Skype, MSN, YM, etc.) et d'autres utilitaires.

3.1.4. Catégorisation commerciale :

Selon la politique de développement, distribution et commercialisation, Il existe trois (3) grands types de logiciels informatique et ce quelque soit le système d'exploitation (GNU/Linux, Windows ou Mac).

- Les logiciels propriétaires et payants
- Les logiciels gratuits
- Les logiciels libres

A. Les logiciels propriétaires et payants

Ces logiciels fabriqués par des sociétés de logiciels sont vendus à l'utilisateur lors de l'achat de son ordinateur, en boîte dans les magasins spécialisés ou sur Internet. Mais ce qui est vendu, ce n'est pas le droit de propriété de l'utilisateur sur le logiciel mais seulement un droit d'utilisation, *c'est une licence utilisateur.*

Exemple : *L'ensemble des logiciels de la firme Microsoft, le système d'exploitation Windows, la suite bureautique Microsoft office, l'environnement de développement VisualC++, etc.*

La deuxième spécification de ce type de logiciels est qu'ils sont sous **copyright**, autrement dit *le code source du logiciel est fermé et protégé*, il n'est pas possible pour un utilisateur quelque soit son niveau de compétence en informatique de modifier le logiciel pour ses besoins ou de rajouter des potentialités nouvelles. La société détentrice de la licence de propriété a ce droit, d'où la genèse commerciale d'innombrables mises à jour payantes.

Le prix des licences d'utilisation peut être fortement augmenté selon que la licence d'utilisation concerne un ou plusieurs postes en réseau ou pas, etc.

Les Sharewares :

Ce sont des logiciels dont on offre gratuitement une version d'évaluation (Essai) pour une période donnée (30, 60, 90 jours...). Dans cette période, l'utilisateur exploite les différentes fonctionnalités sans engagement financier (sans rien payer).

A l'issue de cette période le logiciel cessera de fonctionner, ou bien gardera un nombre minime de fonctionnalités ce qui oblige l'utilisateur à acheter une licence d'utilisation pour la version complète ou désinstaller le logiciel s'il n'est pas intéressé.

Certains éditeurs de ce type de logiciels offre des versions d'essai très limités en termes de fonctionnalités. L'objectif est d'inciter les utilisateurs gentiment à acheter leurs produits

Exemple : *Adobe Photoshop, NitroPDF, Daementools, UltraISO, Idman, etc.*

B. Les logiciels gratuits (freewares)

Ces logiciels déployés par des sociétés de logiciels ou des auteurs indépendants sont gratuits pour l'utilisateur qui peut les télécharger sur Internet ou les procurer sur CD-Rom auprès des kiosques...etc, sans acheter une licence d'utilisation. Ainsi il est possible d'en faire des copies et de les faire circuler entre utilisateurs.

Cependant le code source n'est pas modifiable, ce qui implique que le logiciel n'évolue que si son auteur le fait évoluer. Certains logiciels gratuits représentent *une carte de visite* de son développeur, raison pour laquelle la distribution est gratuite dans l'objectif de se faire connaître.

D'autres logiciels, ne sont gratuits que pour des périodes transitoires, une fois connus, leurs développeurs les rendent payant.

Exemple : Google earth, *Windows messenger*, Foxit Reader, *Adobe reader*, *Google chrome*, *Skype*, *VLC*...

C. Les logiciels libres

Les logiciels libres relèvent d'une logique très différente, l'objectif d'un logiciel libre est la garantie de la liberté de l'utilisateur dans son utilisation d'un logiciel. Ils sont très souvent gratuits pour l'utilisateur. Il est possible d'en faire des copies et de les faire circuler entre utilisateurs.

Un avantage très important d'un logiciel libre est le fait que *son code source est ouvert* et peut-être modifié en toute légalité. Ainsi le partage des *modifications/améliorations* entre les utilisateurs et développeur lui permet un développement très rapide en mettant en profit les expériences des uns et des autres dans un *développement collaboratif* de logiciels libres. Ce qui fait qu'un logiciel libre s'améliore sans cesse !

Le code source ouvert est tout de même protégé par une licence spéciale, la GPL (*General Public Licence*) qui stipule que si on utilise tout ou une partie du code source, alors le résultat doit proposer une version complète et libre du code source modifié, qui à son tour peut être utilisé selon le même principe.

Exemple : *Le système d'exploitation Linux (RedHat, Debian, Ubuntu, Fedora...).*

Ce système d'exploitation très robuste, fiable et performant fait fonctionner des millions d'ordinateurs (70 millions) dans le monde, et de très nombreux programmeurs/utilisateurs s'en servent et l'améliorent chaque jour via Internet.

Il existe bien d'autres logiciels libres et qui sont très utilisées :

Mozilla Firefox, Filezilla, Notepad++, TexStudio, OpenOffice, InkScape, PDFCreator, eMule, 7Zip, EasyPHP, PostgreSQL...

4

Chapitre 4 : Codage et représentation de l'information

1. Introduction

Quelle que soit la nature de l'information traitée par un ordinateur (image, son, texte, vidéo), elle l'est toujours sous la forme d'un ensemble de nombres écrits en *base 2* (*Binaire*), par exemple 01001011. Le terme **bit** (*b* minuscule dans les notations) signifie «**binary digit**», c'est un chiffre dont la valeur ne peut être que « 0 ou 1 » en numérotation binaire.

Dans un ordinateur, toute l'information est sous forme de bits qui sont regroupés en octets (1 octet est équivalent à 8 bits).

Ce concept exige qu'il y ait un codage de cette information. Ce codage dépend bien sûr du type des données. Cette partie décrit les codages les plus utilisés pour les types de base, c'est-à-dire les entiers, les nombres flottants et les caractères.

2. Représentation de l'information

Au niveau circuits électronique, la machine ne comprend pas les caractères ou chiffres tels que A, B, C, 2, 3, 4, 5...etc. Même chose que pour les 1 et 0 (*La machine (PC) n'est pas dotée d'une conscience morale*) ! Les différents chiffres et caractères sont stockés ou transmis sous forme d'impulsions et signaux électriques.

Exemple :

Si on considère k un chiffre **binaire** alors $k = 1$ ou $k = 0$.

Si on considère x un chiffre **décimal** alors si $x = 5$ alors $x = 101$ en base 2

et on note: $(x = 5)_{10}$ alors $(x = 101)_2$ ou simplement : $x = (5)_{10} = (101)_2$.

3. Les unités de mesure en Informatique

Les grandeurs physiques, mathématiques et informationnelles les plus utilisées dans le domaine de l'informatique sont :

3.1. Le bit

Utilisé pour exprimer ou quantifier la taille des mots mémoires, des registres, case mémoires et d'autres, etc. Le «*bit*» représente l'unité élémentaire de représentation et de stockage de l'information dans un système informatique.

Un bit ou élément binaire peut représenter aussi bien une alternative **logique**, exprimée par **faux** et **vrai (Oui / Non)**, qu'un chiffre du système binaire. *Il s'agit de la plus petite unité d'information manipulable par une machine numérique.*

Il est possible de représenter physiquement cette information binaire par un signal électrique ou magnétique, dont la grandeur physique du courant électrique (*tension*) détermine la valeur de l'information stockée dans un bit.

Théoriquement on associe :

- 0V (Volt) à un 0 (Bit dont l'information stockée est "0").
- 5V (Volt) à un 1 (Bit dont l'information stockée est "1").

Cependant, cela n'est pas logique et n'est pas aussi pratique ! La transmission d'une information élémentaire telle qu'un bit dont la valeur est "0" nécessite l'envoi d'un signal électrique correspondant au "0". Comme l'envoi d'un signal électrique avec une tension égale à 0 Volt n'est pas possible, on a introduit la notion d'intervalles, qui est décrite comme suite :

- Un bit = 0 si tension $\in [0.8, 2.4]$ Volts
- Un bit = 1 si tension $\in [2.8, 4.8]$ Volts.

3.2. L'Octet

Représente l'unité de base ou de référence pour exprimer la capacité de stockage, de sauvegarde des différentes mémoires ou dispositifs de stockage.

L'*octet* (en anglais **byte** ou **B** majuscule dans d'autres notations) est une unité d'information composée de 8 bits. Il permet par exemple de représenter un caractère comme une lettre ou un chiffre. Une unité d'information composée de **16 bits** est généralement appelée *mot* (en anglais *word*). Une unité d'information de 32 bits de longueur est appelée *mot double* (en anglais *double word*, d'où l'appellation *dWord*).

Cette capacité est donnée souvent par l'une des expressions suivantes :

Expression	Valeur	En Octet
Octet	8 bits	1
Ko : Kilo Octet	1024 Octet	2^{10}
Mo : Méga Octet	1024 Ko	2^{20}
Go : Giga Octet	1024 Mo	2^{30}
To : Téra Octet	1024 Go	2^{40}
Po : Péta Octet	1024 To	2^{50}
Eo : Exa Octet	1024 Po	2^{60}
Zo : Zetta Octet	1024 Eo	2^{70}
Yo : Yotta Octet	1024 Zo	2^{80}

Tableau 1 - Multiple de l'Octet.

Dans certains cas on utilise le "bit" pour exprimer des caractéristiques physique ou technologique d'un composant, c'est le cas du Microprocesseur, actuellement existe en **64 bits (x64)**, ultérieurement en **32bits (x86 ou x32)**. Il s'agit d'une spécification très importante d'un CPU, parce qu'elle décrit la capacité du traitement et donc une performance capitale de celui-ci.

3.3. Le débit

Est utilisé pour calculer la quantité d'informations transférées par seconde (s). L'unité de mesure associée est le *bit par seconde (bps)*.

Reprenant l'exemple précédent : Avec un rythme de 64 T/s (Transferts par seconde), si Pour chaque transfert on envoie 8 bits, cela nous donne un *débit de transmission de 512 bps*.

Habituellement nous utilisons le terme "*débit*" pour l'expression de la vitesse de transmission des données information que ce soit entre les différents composants d'un système informatique, ou entre équipements (PC) dans un réseau.

3.4. Le Hertz

L'unité de mesure des **fréquences**, qui permet d'exprimer le nombre d'événements quelconques par seconde. En informatique, cette unité est utilisée pour exprimer la fréquence (*Vitesse*) du processeur (*qui est bien traduite comme sa capacité de traitement*), les bus systèmes et Mémoires, le rafraichissement de l'écran...etc.

Exemple :

- CPU Octa-Cœur (8 cœurs) à **4.0 GHz**.
- Mémoire RAM à **1600 MHz** et 4Go d'espace.
- Bande WIFI à **2.4 GHz**, et qui support un débit de 100 *Mbps*.
- Un écran avec une taille de 24", et **60 Hz** comme fréquence de rafraichissement.

4. *Systèmes de numérotation*

Habituellement nous utilisons le système décimal (base 10) dans nos activités quotidiennes. Ce système est basé sur dix (10) symboles, de 0 à 9, avec des unités supérieures (dizaine, centaine, etc.) à chaque fois que dix unités sont comptabilisées.

C'est un système positionnel, c'est-à-dire que l'endroit où se trouve le symbole définit sa valeur. Ainsi, le 2 de **523** n'a pas la même valeur que le 2 de **132**. En fait, 523 est l'abréviation de $5.10^2 + 2.10^1 + 3.10^0$. On peut selon ce principe imaginer une infinité de systèmes numériques fondés sur des bases différentes.

En informatique, outre la base 10, on utilise très fréquemment le **système binaire** (base 2) puisque **l'algèbre booléenne** est à la base de l'électronique numérique. Deux symboles suffisent : 0 et 1. On utilise aussi très souvent le **système hexadécimal** (base 16) du fait de sa simplicité d'utilisation et de représentation pour les mots machines (il est bien plus simple d'utilisation que le binaire). Parfois l'utilisation du **système Octal** (base 8) est possible aussi.

Avant d'aborder la représentation et le codage des différents types de données (caractères, symboles, nombres naturels, nombres réels), il convient de se familiariser avec la représentation des nombres dans une base quelconque.

4.1. Définition de base

Un Système de numérotation désigne le mode de représentation des nombres à l'aide des symboles appelés chiffres. Le nombre de chiffre utilisés pour représenter les valeurs (nombres) est appelé "**base b**".

Exemple :

Soit : b une base de numérotation.

x un nombre écrit dans la base b ,

x est composé de n chiffres a_i de la base b

$$x = a_n a_{n-1} \dots a_1 a_0$$

- En Binaire, $b = 2$ $a_i \in \{0, 1\}$;
- En base 4, $b = 4$ $a_i \in \{0, 1, 2, 3\}$;
- En Octal, $b = 8$ $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$;
- En Décimal, $b = 10$ $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$;
- En Hexadécimal, $b = 16$ $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$;

En base 16 (*Hexadécimale*) On utilise les 6 premières lettres de l'alphabet "**En majuscules**" comme des chiffres,

A = 10, B = 11, C = 12, D = 13, E = 14, F = 15

4.2. Représentation des nombre entiers

En base 10, la représentation du nombre 2016 peut être décrite comme suite :

$$2016 = 2.10^3 + 0.10^2 + 1.10^1 + 6.10^0$$

$$2016 = 2000 + 0 + 10 + 6$$

Dans le cas général un nombre $x = a_n a_{n-1} \dots a_1 a_0$ est donné par la formule suivante :

$$a_n a_{n-1} \dots a_1 a_0 = \sum_{i=0}^n a_i b^i$$

b^0 représente le poids faible, et b^n le poids fort

En base 2 le nombre $(101)_2 = 1.2^2 + 0.2^1 + 1.2^0$
 $= 4 + 0 + 1 = (5)_{10}$

4.3. Représentation des nombre fractionnaires

Un nombre fractionnaire comporte des chiffres décimaux après la virgule.

En base 10 : $12.346 = 1.10^1 + 2.10^0 + 3.10^{-1} + 4.10^{-2} + 6.10^{-3}$

Dans le cas général, en base b nous avons :

$$\{a_n a_{n-1} \dots a_1 a_0\}, \{a_{-1} a_{-2} \dots a_{-p}\} = \{a_n b^n + a_{n-1} b^{n-1} + \dots + a_0 b^0\} + \{a_{-1} b^{-1} + a_{-2} b^{-2} + \dots + a_{-p} b^{-p}\}$$

Exemple :

$$(11,101)_2 = 1.2^1 + 1.2^0 + 1.2^{-1} + 0.2^{-2} + 1.2^{-3}$$

$$= 2 + 1 + 0,5 + 0 + 0,125 = (3.625)_{10}$$

Exercice

Donner une représentation des nombres suivants :

$(512)_{10}$, $(2167)_{10}$, $(1432.45)_8$, $(0.785)_{10}$, $(10110101)_2$, $(11101.101)_2$, $(753)_8$, $(92)_8$

4.4. Transcodage (Conversion de bases)

Nous présentons dans ce qui suit les différentes méthodes les plus répandues utilisées pour représenter les nombres dans différentes bases (Système de numérotation).

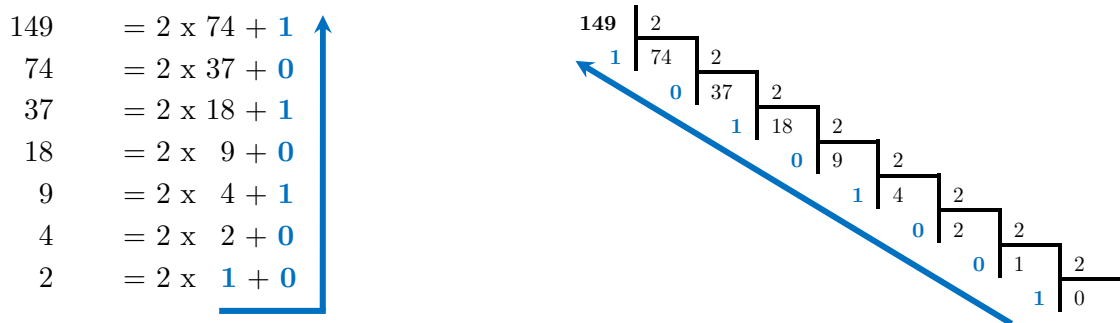
4.4.1. Base décimale vers autres bases

A. Passage de la base 10 vers la base 2 :

Nous avons le choix entre deux méthodes lors de la conversion d'un nombre entier de la base 10 vers la base 2. La première méthode se base sur la division successive du nombre décimale par 2 (base 2), Le résultat est obtenu en concevant seulement les restes de cette division commençant par le dernier reste au premier (du bas en haut).

Exemple : $(149)_{10} = (10010101)_2$

Une deuxième méthode est basée sur l'analyse du nombre décimal en prenant les restes commençant par le bas, comme suite :



On peut aussi utiliser la suite des puissances de 2 (Ecrit de droite à gauche) pour convertir n'importe quel nombre en base décimale vers un autre en base binaire comme suite :

...	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
...	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

Le principe est d'effectuer une addition des nombres figurant dans la suite (*seulement ceux qui figurent dans la suite*), pour avoir le nombre décimal à convertir, commençant par le plus grand nombre inférieur à ce dernier. A chaque fois qu'un nombre est considéré dans l'addition on le marque par un (1) en dessous sinon il sera marqué par zéro (0). Reprenons l'exemple précédant $(149)_{10}$.

...	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
...	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
						1	0	0	1	0	1	0	1

- On commence l'addition par le nombre "**128**", c'est le plus grand nombre inférieur à 149. On marque 128 par "**1**". (Résultats : 1)
- On ajoute "**64**" on aura un nombre **supérieur** à 149 ($128 + 64 = 192 > 149$).
On marque 64 par "**0**". (Résultats : 10)
- On ajoute "**32**" on aura un nombre **supérieur** à 149 ($128 + 32 = 160 > 149$).
On marque 32 par "**0**". (Résultats : 100)
- On ajoute "**16**" on aura un nombre **inférieur** à 149 ($128 + 16 = 144 < 149$).
On marque 16 par "**1**". (Résultats : 1001)
- On ajoute "**8**" on aura un nombre **supérieur** à 149 ($128 + 16 + 8 = 152 > 149$).
On marque 8 par "**0**". (Résultats : 10010)
- On ajoute "**4**" on aura un nombre **inférieur** à 149 ($128 + 16 + 4 = 148 < 149$).
On marque 4 par "**1**". (Résultats : 100101)
- On ajoute "**2**" on aura un nombre **supérieur** à 149 ($128 + 16 + 4 + 2 = 150 > 149$).
On marque 2 par "**0**". (Résultats : 1001010)
- On ajoute "**1**" on aura un nombre **supérieur** à 149 ($128 + 16 + 4 + 1 = 149$).
On marque 1 par "**1**". (Résultats : 10010101)

$$149 = 128 + 16 + 4 + 1$$

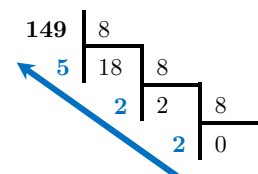
$$(149)_{10} = (10010101)_2$$

B. Passage de la base 10 vers la base 8

On peut utiliser le même principe de division successive sauf que la division est faite par 8 au lieu de 2. **Ex :** $(149)_{10} = (225)_8$

Une autre méthode consiste à suivre les étapes suivantes :

1. Convertir le nombre décimal en nombre binaire.
2. Regrouper les bits résultants par 3 **commençant par la droite** (*compléter par des "0" à gauche si nécessaires*).
3. Convertir chaque groupe de 3 bits en décimal séparément des autres bits.

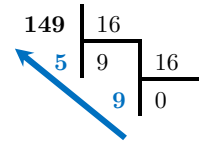


Exemple :

$$\begin{aligned}(149)_{10} &= (10010101)_2 \\ &= (\mathbf{010}) (010) (101) \\ &= 2 \quad 2 \quad 5 = (225)_8\end{aligned}$$

C. Passage de la base 10 vers la base 16 (Hex)

La méthode de la division successive nous permet de convertir un nombre décimal en un nombre écrit dans la base Hexadécimale (*hex*).



$$(149)_{10} = (95)_{16}$$

Tout comme la conversion vers la base octale (8), on peut utiliser la méthode de regroupement des bits sauf que cette fois-ci on utilise 4 bits par groupe.

Exemple :

$$\begin{aligned}(149)_{10} &= (10010101)_2 &= (1001) (0101) \\ & &= 9 \quad 5 &= (95)_{16} \\ (3784)_{10} &= (111011001000)_2 &= (1110) (1100) (1000) \\ & &= E \quad C \quad 8 &= (EC8)_{16}\end{aligned}$$

4.4.2. Base Binaire vers autres bases

D. Passage de la base 2 vers la base 10

Pour passer de la base 2 à la base 10 on utilise la formule de représentation décrite en "Section 4.2" comme suite :

$$\begin{aligned}(10010101)_2 &= 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 128 + 0 + 0 + 16 + 0 + 4 + 0 + 1 = (149)_{10}\end{aligned}$$

On peut aussi utiliser la même suite des puissances de 2 (décrite ci-dessus) pour reconstruire le nombre décimal en faisant l'addition seulement des nombres marqués par "1" dans la suite.

E. Passage de la base 2 vers la base 8

La méthode la plus simple est d'utiliser le regroupement par 3 des bits commençant par la droite ensuite convertir chaque ensemble de 3 bits séparément des autres.

$$\begin{aligned}(149)_{10} &= (10010101)_2 &= (\mathbf{010}) (010) (101) \\ & &= 2 \quad 2 \quad 5 &= (225)_8\end{aligned}$$

F. Passage de la base 2 vers la base 16

Le même principe précédent peut être appliqué lors du passage du binaire vers l'hexadécimal sauf que le regroupement se fait par des ensembles de 4 bits.

$$\begin{aligned}(3784)_{10} &= (111011001000)_2 &= (1110) & (1100) & (1000) \\ & &= E & C & 8 & = (EC8)_{16}\end{aligned}$$

4.4.3. Base octale vers autres bases

A. Passage de la base 8 vers la base 2

La première méthode repose sur le principe de la division successive par 2, et la retenue des restes de la division inversement au sens de la division. Une autre méthode plus simple est de convertir chaque chiffre " a_i " du nombre octal en bits en commençant par la droite.

On peut ajouter des "0" à gauche pour compléter les ensembles de moins de 3 bits.

$$\begin{aligned}(225)_8 &= 2 & 2 & 5 \\ &= (010) & (010) & (101) & = (10010101)_2\end{aligned}$$

B. Passage de la base 8 vers la base 10

Deux alternatives existent, soit qu'on utilise la formule générale de représentation des nombres, ou bien passer par la base binaire avant de convertir en base 10 :

$$\begin{aligned}\text{base 8} &\rightarrow \text{base 2} \rightarrow \text{base 10} \\ (225)_8 &= 2 \cdot 8^2 + 2 \cdot 8^1 + 5 \cdot 8^0 = 128 + 16 + 5 = (149)_{10}\end{aligned}$$

C. Passage de la base 8 vers la base 16

Deux alternatives existent, soit qu'on utilise la formule générale de représentation des nombres, ou bien passer par la base binaire avant de convertir en base 16 :

$$\begin{aligned}\text{Base 8} &\rightarrow \text{base 2} \rightarrow \text{base 16} \\ (225)_8 &= (10010101)_2 = 1001 & 0101 & = (95)_{16}\end{aligned}$$

4.4.4. Base Hexadécimale vers autres bases

1. Passage de la base 16 vers la base 2

On utilise la *représentation de chaque chiffre hexadécimal sur 4 bits* :

$$(EC8)_{16} = E \quad C \quad 8$$

$$= (1110) \quad (1100) \quad (1000) \quad = (111011001000)_2$$

2. Passage de la base 16 vers la base 8

On utilise la *représentation de chaque chiffre hexadécimal sur 4 bits* ensuite on regroupe par 3 les bits avant de les convertir séparément pour avoir les chiffres octal :

$$\begin{aligned} (EC8)_{16} &= \quad E \quad \quad C \quad \quad 8 \\ &= (1110) \quad (1100) \quad (1000) \quad = (111011001000)_2 \\ &= 111 \quad 011 \quad 001 \quad 000 \\ &= 7 \quad \quad 3 \quad \quad 1 \quad \quad 0 \quad = (7310)_8 \end{aligned}$$

3. Passage de la base 16 vers la base 10

On utilise directement la formule générale de représentation de données pour passer de la base 16 à la base 10. L'autre alternative consiste à passer par l'intermédiaire de la base 2.

Exemple :

$$\begin{aligned} (EC8)_{16} &= E \cdot 16^2 + C \cdot 16^1 + 8 \cdot 16^0 \\ &= 14 \cdot 16^2 + 12 \cdot 16^1 + 8 \cdot 16^0 \quad = (3784)_{10} \end{aligned}$$

Les différentes opérations de transcodage sont résumées par le schéma suivant :

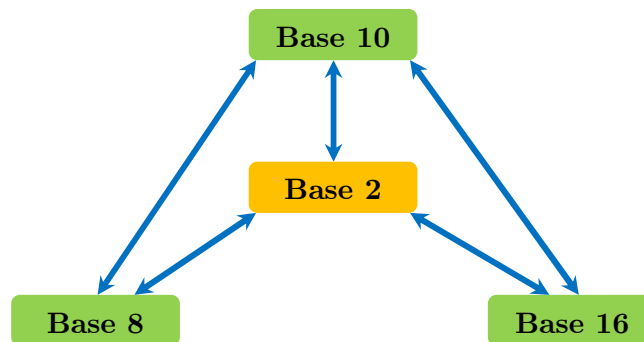


Figure 1 - Schéma général du transcodage (Conversion).

4.4.5. Conversion d'un nombre réel (fractionnel)

Le passage de base 10 en base 2 est plus subtil. La partie entière se transforme comme décrit précédemment au 4.4.1, alors que la conversion de la partie décimale se fait en la multipliant cette partie par 2, et on continue jusqu'à ce que le résultat soit "0".

Ensuite le résultat est obtenu par l'extraction du chiffre entier à gauche.

Exemple : $(149,347)_{10}$

La partie entière est donnée par : $(149)_{10} = (10010101)_2$

La partie décimale est obtenue comme suite :

$0,347 \cdot 2 = 0,694$	$0,347 = 0,0\dots$
$0,694 \cdot 2 = 1,388$	$0,347 = 0,01\dots$
$0,388 \cdot 2 = 0,766$	$0,347 = 0,010\dots$
$0,766 \cdot 2 = 1,552$	$0,347 = 0,0101\dots$
$0,552 \cdot 2 = 1,104$	$0,347 = 0,01011\dots$
$0,104 \cdot 2 = 0,208$	$0,347 = 0,010110\dots$
$0,208 \cdot 2 = 0,416$	$0,347 = 0,0101100\dots$
$0,416 \cdot 2 = 0,832$	$0,347 = 0,01011000\dots$
$0,832 \cdot 2 = 1,664$	$0,347 = 0,010110001\dots$
$0,664 \cdot 2 = 1,328$	$0,347 = 0,0101100011\dots$

$$(149,347)_{10} = (10010101,0101100011\dots)_2$$

On continue jusqu'à ce que la partie décimale soit nulle, ou bien atteindre la précision désirée (*Le nombre de chiffres après la virgule*).

4.5. Opérations arithmétiques

Les différentes opérations arithmétiques (+ - * /), s'effectuent en base quelconque b avec les mêmes méthodes qu'en base 10. Une retenue ou un report apparaît lorsque l'on atteint ou dépasse la valeur b de la base.

Exemple : En base 10 : $(5 + 4 = 9)_{10}$

En base 8 : $(5 + 4 = 11)_8$

En base 2 : $(1+1 = 10)_2$

En base 16 : $(9 + 9 = 12)_{16}$

5. Codage de l'information

Le rôle de la machine est de traiter des informations numériques représentant des grandeurs mathématiques, physiques, etc. telles que la lumière, la température, la tension, la vitesse, le son, la pression, etc. Mais avant de traiter une grandeur il faut d'abord la convertir en une *reproduction numérique représentative de la grandeur* (l'information) pouvant être manipulée au niveau des circuits électroniques.

On utilise pour cela des *mécanismes de codage de l'information*, qui traduisent fidèlement et sans ambiguïté l'information dans son état brute en une succession de bits appelée *mot-codé*.

Ainsi, ils existent plusieurs problèmes relatifs au codage de l'information selon :

- **Sa nature** : qui peut être un nombre (signé, non signé, réel...) ou caractère (alphabétique, symbole, spécial, mathématique...).
- **Son rôle** : Qui peut être une instruction, une description, un simple texte, une commande, un signal d'interruption...etc.
- **Sa taille** : Les registres d'un ordinateur (*Au niveau CPU*) sont de taille fixe (*32 à 128 bits*), ce qui limite la valeur numérique à représenter.

Exemple :

Comment représenter les entiers négatifs (signés) ?

Comment représenter les caractères alphabétiques ?

Comment différencier entre "a" et "A"?

Comment représenter les symboles "/", *, @, etc.?

Comment représenter les instructions, les signaux, etc. ?
etc. ?

5.1. Typologie de l'information

Dans un système informatique (au niveau électronique), l'information ne peut avoir que deux formes indispensables :

- **La donnée** : représente l'opérande sur laquelle porte l'opération décrite par l'instruction. Cependant les données sont aussi de deux types :
 - *Numérique* : Regroupant l'ensemble des nombres (Entier, réel...etc)
 - *Caractère* : Regroupant l'ensemble des symboles, lettres...etc.
- **L'instruction** : Chaque instruction représente ou décrit l'opération à effectuer sur une ou un ensemble de données.

5.2. Débordement (Overflow)

Du fait que la taille des registres est fixe, les retenus résultants d'un calcul mathématique au niveau du bit de poids fort engendre un dépassement de capacité du registre, ce qui est appelé un *débordement*.

Exemple :

Un ordinateur fonctionnant avec des registres de 4 bits, effectue l'opération binaire $(1110)_2 + (0110)_2 = (10100)_2$

$$\begin{array}{r} 1 \quad 1 \quad 1 \\ 1 \quad 1 \quad 1 \quad 0 \\ + \quad 0 \quad 1 \quad 1 \quad 0 \\ = \quad \boxed{0} \quad \boxed{1} \quad \boxed{0} \quad \boxed{0} \end{array}$$

Suites aux différentes contraintes précédemment citées, nous constatons qu'il ne suffit pas de convertir une information en bits pour pouvoir l'exploiter ou traiter par un système informatique. Il faut donc lui associer une signification (sens), ce qui est possible en utilisant l'une des méthodes de codage de l'information.

5.3. Codage des informations numériques

Selon la nature du nombre à coder, nous avons plusieurs méthodes de codage des nombres.

5.3.1. Codage des entiers non signés

Autrement dit entiers positifs ou en valeur absolue. Le seul problème posé est le nombre de bits nécessaires pour la représentation de ce type d'entier. Un groupe de " n " bits nous permet de représenter les nombres entier $x \in [0, 2^n-1]$.

Exemple :

$n = 5$ bits, on peut représenter les nombre appartenant à $[0, 31] / (31)_{10} = (11111)_2$

Le nombre 32 ne peut être représenté sur 5 bits.

$n = 8$ bits, on peut représenter les nombre appartenant à $[0, 255] / (255)_{10} = (11111111)_2$

Le nombre 256 ne peut être représenté sur 8 bits.

5.3.2. Codage des entiers signés

Il s'agit des nombres entiers négatifs dont il existe trois méthodes pour les représenter

A. Signe et Valeur Absolue (SAV)

Cette méthode consiste à utiliser le bit du poids le plus fort pour le signe, par convention "0" représente le signe positif (+), "1" représente le signe négatif (-). Le reste des bits est réservé pour la représentation du nombre entier en valeur absolue.

Ainsi l'intervalle des nombres représentable sur n bits soit : $x \in [-(2^{n-1}-1), +(2^{n-1}-1)]$.

La traduction d'un nombre binaire écrit en SVA en décimal se fait en convertissant les bits de la valeur absolue en décimal, ensuite précéder le résultat par le signe correspondant à la convention (0 +, 1 -).

$$(00101101)_2 = (+45)_{10} \quad (10000111)_2 = (-7)_{10}$$

Le codage en SVA est très simple mais il pose des difficultés avec les opérations arithmétiques à cause du bit de signe qui doit être traité séparément.

Exemple : Calculer en SVA $+5 +(-2)$ sur 4 bits ?

L'inconvénient de cette méthode est la double représentation du 0, (+0 -0).

B. Complément à 1 (C1)

Dans cette méthode les nombres positifs gardent le format binaire. Les nombres négatifs sont complétés bit par bit, les "1" deviennent des "0" et l'inverse. ainsi l'intervalle de nombre représentables soit $x \in [-(2^{n-1}-1), +(2^{n-1}-1)]$.

Lors de la reconstitution du nombre en base décimale, les nombre positifs sont obtenus par conversion classique. Les nombre négatifs sont obtenu en inversant tous les bits, convertir le nombre et en fin introduire le signe "-".

$$(00001100)_2 = (+12)_{10} \quad (11111000)_2 = (-7)_{10}$$

L'inconvénient de cette méthode est la double représentation du 0, (+0 -0).

L'addition en C1 dépend des deux cas suivants :

- Si aucune retenue n'est générée par le bit de signe, donc le résultat est correct et il est représenté en C1.
- Sinon, la retenue sera additionnée au résultat de l'opération, celui-ci est représenté en C1.

C. Complément à 2 (C2)

Tout comme le C1, les nombre positif garde leur format binaire, les nombres négatifs sont obtenu en calculant le complément à 1, auquel est ajouté un "1" : $C2 = C1 + 1$. L'intervalle de nombre représentables soit $x \in [-(2^{n-1}-1), +(2^{n-1}-1)]$.

La restitution du nombre décimal positif se fait par transcodage classique. Les nombres négatifs sont obtenu par :

- Attribution du signe (-) correspondant au bit du poids fort.
- Inverser tout les bits restant, ensuite ajouter 1

$$(00001100)_2 = (+12)_{10} \quad (10001000)_2 = (-120)_{10}$$

L'addition en C2 se fait comme suite :

- S'il y a une retenue générée par le bit de signe, celle-ci sera ignorée.
- Le résultat obtenu est en Complément à 2.

La représentation en C2 est la méthode la plus utilisée pour les nombres négatifs, ainsi il n'y a qu'une seule représentation du 0.

Remarque :

- En C1 & C2 les opérations arithmétiques sont avantageuses. En effet, la soustraction d'un nombre se réduit à l'addition de son complément. ainsi il n'y a pas de traitement particulier pour le bit du signe.
- il y'aura un débordement en C2 si :
 - La somme de deux nombres négatifs donne un nombre positif ou l'inverse.
 - Le résultat de l'opération dépasse l'intervalle des valeurs représentables.
- Il ne peut y avoir un débordement si les deux opérandes sont de signes différents.

Exercice

Donner en SVA, C1 et C2 le codage des nombres représentables sur 4 bits.

5.3.3. Codage des nombre fractionnaires

A. Codage en virgule fixe

Cette méthode n'est pas très pratique, au premier lieu on a proposé de ne pas prendre en considération la virgule, pour que le nombre fractionnaire soit traité comme un nombre entier. La gestion de la virgule dans ce cas est laissée au programmeur qui décidera sa position.

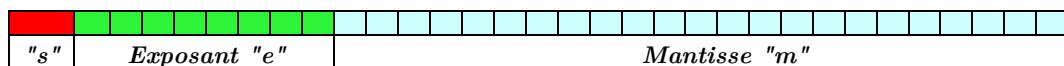
D'autre part cette méthode présente une limitation de l'intervalle des nombres représentables sur n bits.

Exemple : Soit un nombre représentable sur 6 bits tel que :

- 1 bit de signe, 3 bits pour la partie entière, 2 bits pour la partie fractionnaire.
- L'intervalle des valeurs représentables sera donc $[-7.75, +7.75]$

B. Codage en virgule flottante - Norme IEEE 754

Cette norme est proposée pour donner une représentation unifiée des nombres en virgule flottante. Elle définit trois composants de codage des nombres flottants comme décrit dans la forme suivante :



- Le signe "s" (0 si +, 1 si -) est représenté par un seul bit, celui du poids fort.
- L'exposant est codé sur les "e" bits consécutifs au signe.
- La mantisse (les bits situés après la virgule) codés sur les "m" bits restants.

Selon cette norme on peut utiliser plusieurs modes (Tailles) de précision 32, 64 et 80 bits comme suite :

Taille		Signe	Exposant	Mantisse
32	Simple précision	1	8	23
64	Double précision	1	11	52
80	Précision étendue	1	15	64

Tableau 2 - Forme du codage IEEE 754.

Cette norme définit aussi les règles et conditions suivantes :

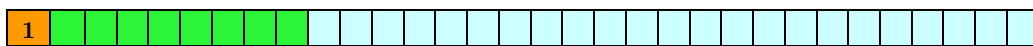
- L'exposant 00000000 est interdit.
- L'exposant 11111111 est interdit. On s'en sert toutefois pour signaler des erreurs, on appelle alors cette configuration du nombre NaN (Not a Number).
- Il faut rajouter 127 (01111111) à l'exposant pour une conversion de décimal vers un nombre réel binaire. Les exposants peuvent ainsi aller de -254 à 255.
- La formule d'expression des nombres réels est ainsi la suivante :

$$(-1)^s \times 2^E \times M = (-1)^s \times 2^{(e-127)} \times (1, m)$$

Exemple 1

Convertir le nombre -15,6875 en binaire selon la norme IEEE 754 (32 bits):

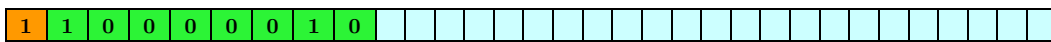
1. Représenter le signe négatif par 1.



2. Donner la représentation binaire classique $(15,6875)_{10} = (1111,1011)_2$

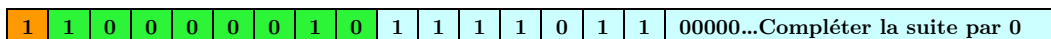
3. Normaliser : $1111,1011 = 1,1111011 \times 2^3$

4. Calculer l'exposant $e = 3 + 127 = (130)_{10} = (10000010)_2$



5. Extraire la mantisse de la représentation normalisée ($1,1111011 \times 2^3$), On ne prend que les bits situés après la virgule.

(Le bit restant de la partie entière est appelé le bit caché)



$$(-15,6875)_{10} = (110000010, 111101100000000000000000)_2 \text{ En norme IEEE754}$$

Exemple 2

Convertir le nombre 3,14 en binaire selon la norme IEEE 754 (32 bits):

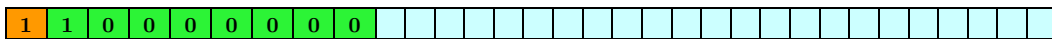
1. Représenter le signe négatif par 1.



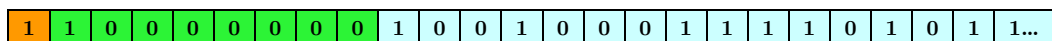
2. Donner la représentation binaire classique $(3,14)_{10} = (11,001000111101011...)_{2}$

3. Normaliser : $11,001000111101011... = 1,1001000111101011... \times 2^1$

4. Calculer l'exposant $e = 1 + 127 = (128)_{10} = (10000000)_2$



5. Extraire la mantisse de la représentation normalisée $(1,1001000111101011... \times 2^1)$, On ne prend que les bits situés après la virgule.



$$(3,14)_{10} = (110000000,1001000111101011...)_{2} \text{ En norme IEEE754}$$

Pour convertir un nombre binaire IEEE 754 en décimal, on procédera d'une manière récursive, sans avoir besoin de faire des calculs intenses ! Reprenons l'exemple 1 :

$$(110000010, 111101100000000000000000)_{2}$$

1. Extraire le premier bit (Bit du poids fort) = 1, donc le nombre est négatif (-).
2. Extraire les 8 bits suivants de la partie entière : $(10000010)_2 = (130)_{10}$
3. Soustraire 127 : $130 - 127 = 3$.
4. Ajouter le bit caché à la mantisse : $(0,111101100000000000000000 + 1)_2 = 1,111101100000000000000000$
5. Décaler la virgule de 3 bits, on obtient le résultat suivant :

$$(- 1111,10110000000000000000)_{2} = (-15, 6875)_{10}$$

Remarque

Il existe aussi d'autres mécanismes de codage tel que : STIBITZ, BCD, Aïken, Gray...etc.

5.4. Codage des caractères

Les caractères sont des données non numériques : il n'y a pas de sens à additionner ou multiplier deux caractères. Par contre, il est souvent utile de comparer deux caractères, par exemple pour les trier dans l'ordre alphabétique. On n'en parle pas seulement des caractères alphabétiques, mais aussi de tout autre symbole utilisé en mathématique (α , β ...), signe économique (\$, €...), Internet (@...), La ponctuation (!, ?, ;, :,...) les caractères spéciaux (#, &...)...etc.

N'importe quel texte est codé sous forme d'une chaîne (suite) de caractères, où chacun possède son propre code. Ainsi le code du "a" est différent du "A"... Pour cela il faudrait établir une correspondance entre un caractère et son code binaire unique.

5.4.1. Le code ASCII

Les deux codes les plus connus sont l'EBCDIC (*qui est en voie de disparition*) et le code ASCII (American Standard Code for Information Interchange). Ce dernier représente chaque caractère sur 7 bits (*on parle parfois de code ASCII étendu, utilisant 8 bits pour coder des caractères supplémentaires*). Ce codage code 127 caractères incluant les 26 lettres latines en version majuscule et minuscule, les 10 chiffres décimaux, l'espace, les symboles de ponctuation et de parenthésage.

Notons que le code ASCII original, défini pour les besoins de l'informatique en langue anglaise) ne permet pas la représentation des caractères accentués (é, è, à, ù,...), et encore moins des caractères chinois ou arabes. Pour ces langues, d'autres codages existent, utilisant 16 bits par caractères.

Plusieurs points importants à propos du code ASCII :

- Les codes compris entre 0 et 31 ne représentent pas des caractères, ils ne sont pas affichables. Ces codes, souvent nommés caractères de contrôles sont utilisés pour indiquer des actions comme retour à la ligne (CR, LF), émettre un bip sonore (BEL)...etc.
- Les lettres se suivent dans l'ordre alphabétique (codes 65 à 90 pour les majuscules, 97 à 122 pour les minuscules), ce qui simplifie les comparaisons. On passe des majuscules aux minuscules en modifiant le 5^{ème} bit, ce qui revient à ajouter 32 au code ASCII décimal.
- Les chiffres sont rangés dans l'ordre croissant (codes 48 à 57), et les 4 bits de poids faibles définissent la valeur en binaire du chiffre.

La table ASCII de base est donnée dans le tableau suivant :

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Espace	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledgde	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Bachspace	40	28	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg.Acknowledgde	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Groupe separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Tableau 3 - Table du code ASCII standard.

5.4.2. Le code ASCII étendu

Etant donné que le code ASCII ne contient pas de caractères accentués, il a donc été étendu à 8 bits pour pouvoir codé plus de caractères, on parle de l'*ASCII Etendu* (Norme ISO-8859), qui ne contient pas seulement tous les caractères accentués, mais aussi d'autres symboles et signes.

Exemple :

128	80	ç	160	A0	á	192	40	L	224	60	α
129	81	ü	161	A1	í	193	41	⊥	225	61	β
130	82	é	162	A2	ó	194	42	T	226	62	Γ
131	83	â	163	A3	ú	195	43	⊥	227	63	π

Tableau 4 - Exemple du code ASCII étendu.

5.4.3. Le code UTF-8

Il existe d'autres normes que l'ASCII, comme l'**Unicode** dont l'**UTF-8** est un membre. Ce type de codage présente l'avantage de proposer une version unifiée des différents encodages de caractères complétant l'ASCII, mais aussi de permettre l'encodage de caractères autres que ceux de l'alphabet latin. Unicode définit des dizaines de milliers de codes, mais les 128 premiers restent compatibles avec ASCII.

Dans le codage UTF-8, chaque caractère de code peut être codé sur une suite variée de 1, 2, 3 ou 4 octets. Il a été conçu pour être compatible avec certains logiciels originellement prévus pour traiter des caractères d'un seul octet. Les bits de poids fort du premier octet codant l'un des caractères indique toujours le nombre d'octets codant (*utilisés pour le codage du caractère*).

Si le caractère est codé sur 2 octets, les trois premiers bits du premier octet sont "110". S'il est codé sur 3 octets, alors les quatre premiers bits du premier octet sont "1110". Et enfin s'il est codé sur 4 octets les cinq premiers bits du premier octet sont "11110". Les deux bits de poids fort des octets qui suivent le premier octet codant sont toujours "10", Cela est résumé dans le tableau suivant :

Nbre d'octet du code	Format du code
1	0xxxxxxx
2	110xxxxx 10xxxxxx
3	1110xxxx 10xxxxxx 10xxxxxx
4	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Tableau 5 - Forme d'un code UTF-8

5.5. Alphabet, mots et langage

Le codage des différents symboles, signes et caractères, nous permet la notion *d'alphabet*, qui à la base n'est qu'un ensemble fini non vide d'éléments qui sont appelé selon le contexte "*Symboles, caractères, lettres, nombres*".

Un alphabet numérique regroupe l'ensemble des nombres composant la base du système de numérotation désigné :

- En base 10 l'alphabet numérique est : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- En Base 16 l'alphabet numérique est : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Un *mot* construit sur un alphabet représente une suite fini d'éléments de l'alphabet. Par exemple "58F9" est un mot construit à partir de l'alphabet numérique de la base 16, ainsi "Code" est un mot construit à partir de l'alphabet des lettres {a, b, ..., z}.

L'ensemble des mots créer sur un *alphabet A*, qui est défini par tout sous-ensemble A^* , constitue le *langage* associé à l'alphabet *A*. Cependant, et même si la création d'un mot est aléatoire, on a besoin de certaines règles pour définir des mots significatifs avec une sémantique logique.

Par exemple :

- "Okértsaile", est un mot défini sur l'alphabet français {a,b,...}, mais du point de vue sémantique, ce mot n'a aucun sens dans le langage Français.
- "Machine" est un mot significatif du langage français signifie un dispositif automatique.

5

Chapitre 5 : Algèbre de BOOLE

1. Introduction

L'algèbre de Boole, ou calcul booléen, est la partie des mathématiques qui s'intéresse aux opérations et aux fonctions sur les variables logiques. Elle fut inventée par le mathématicien britannique "**George BOOLE**" (1815-1864).

Aujourd'hui, l'algèbre de Boole trouve de nombreuses applications en informatique particulièrement dans la conception des circuits électroniques. Les machines numériques sont constituées d'un ensemble de circuits électroniques. Chaque circuit fournit une fonction logique bien déterminée (addition, comparaison, etc.).

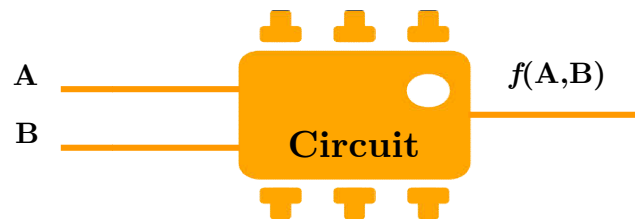


Figure 3 - Schéma fonctionnel d'un circuit électronique.

La fonction $f(A,B)$ peut être : la somme, multiplication... de A et B , ou le résultat de la comparaison de A et B ou une autre fonction. Pour concevoir et réaliser le circuit correspondant à la fonction f , on doit avoir le modèle mathématique de cette fonction. Sachant que ce modèle doit prendre en considération le système binaire.

2. Algèbre de BOOLE

L'algèbre de BOOLE a définie le principe des **systemes à deux états** qui s'excluent mutuellement, en utilisant des fonctions (*Expressions*) booléennes constituées par des variables qui ne peuvent prendre que deux valeurs possibles {FAUX, VRAI} / {NON, OUI} / {Bas, Haut} / {0, 1} ...etc. Ainsi les résultats de ces fonctions seront de même types de valeurs utilisées dans les fonctions. Sachant que ni les variables ni les résultats ne peuvent avoir les deux valeurs (Vrai et Faux) en même temps. Ce principe est bien adapté au Système binaire (0 et 1).

Exemple de systèmes à deux états :

- Un interrupteur est ouvert ou non ouvert (fermé).
- Une lampe est allumée ou non allumée (éteinte).
- Une porte est ouverte ou non ouverte (fermée).

2.1. Définition

On appelle B l'ensemble constitué de deux éléments appelés *valeurs de vérité* {FAUX, VRAI}. Cet ensemble est aussi noté $B = \{0, 1\}$, notation que l'on utilisera désormais. Sur cet ensemble on peut définir les lois **ET**, **OU** et **NON** qui représente une transformation appelée « *complémentaire* », « *inversion* » ou « *contraire* ».

2.1.1. La fonction - ET -

Elle est définie de la manière suivante : a ET b est VRAI si et seulement si a est VRAI et b est VRAI. Cette loi est aussi notée :

- $a \cdot b$
- $a \wedge b$ (dans quelques notations algébriques, ou en APL)
- $a \& b$ ou $a \&\&b$ (*Perl, C, PHP, ...*)
- a AND b (*Ada, Pascal, Python, Java, C++ ...*)

2.1.2. La fonction - OU -

Elle est définie de la manière suivante : a OU b est VRAI si et seulement si a est VRAI ou b est VRAI, ou si a et b sont vrais. Cette loi est aussi notée :

- $a + b$
- $a \vee b$ (dans quelques notations algébriques ou en APL)
- $a | b$ ou $a || b$ (*Perl, C, PHP, ...*)
- a OR b (*Ada, Pascal, Python, Java, C++ ...*)

2.1.3. La fonction - NON -

Le contraire de « a » est VRAI si et seulement si a est FAUX. Le contraire (Appelé aussi *l'inverse*) de a est noté :

- $\neg a$
- $\sim a$ (dans quelques notations algébriques ou en APL)
- $!a$ (*C, C++, Java, ...*)
- NOT a (*ASM, Pascal, ...*)

3. Fonctions logiques et tables de vérité

Une table de vérité est un tableau qui représente des entrées (en colonne) et des états binaires (0 et 1) en sortie. Le résultat, exprimé lui aussi sous forme binaire, se lit dans la dernière colonne. Le tableau suivant présente les différentes fonctions logiques, les *portes logiques* et opérations booléennes associées ainsi que les tables de vérité de chaque opération.


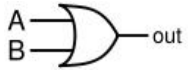
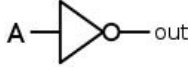
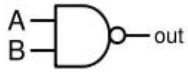
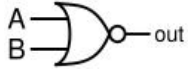
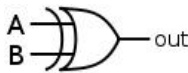
Fonctions	Porte logique	Opération booléenne	Table de vérité		
ET (AND)		$A \cdot B$	Entrées		Sortie
			A	B	A AND B
			0	0	0
			0	1	0
			1	1	1
OU (OR)		$A + B$	Entrées		Sortie
			A	B	A OR B
			0	0	0
			0	1	1
			1	1	1
NON (NOT)		\bar{A}	Entrées		Sortie
			A		NOT A
			0	1	
1	0				
NOT ET (NAND)		$\overline{A \cdot B}$	Entrées		Sortie
			A	B	A NAND B
			0	0	1
			0	1	1
			1	1	0
NOT OR (NOR)		$\overline{A + B}$	Entrées		Sortie
			A	B	A NOR B
			0	0	1
			0	1	0
			1	1	0
OU Exclusif (XOR)		$A \oplus B$ $= A \cdot \bar{B} + \bar{A} \cdot B$	Entrées		Sortie
			A	B	A XOR B
			0	0	0
			0	1	1
			1	1	0

Tableau 6 - Fonctions et portes logiques

En électronique, une porte NON est plus communément appelée *inverseur*. Le cercle utilisé sur la représentation est appelé « bulle », et montre qu'une entrée ou une sortie est inversée.

3.1. Fonction logique

C'est une fonction qui relie N variables logiques avec un ensemble d'opérateurs logiques de base (ET, OU, NON). La valeur d'une fonction logique est égale à 1 ou 0 selon les valeurs des variables logiques dont on a 2^n combinaisons possibles, qui sont représentées dans une table qui s'appelle *table de vérité* (TV).

Exemple d'une fonction logique :

$$F(A, B, C) = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \boxed{A\bar{B}C} + A.B.C$$

Terme

La fonction possède 3 variables, donc nous avons 2^3 combinaisons, elle est composée aussi de quatre (4) Termes :

$$F(0,0,0) = \bar{0}\bar{0}0 + \bar{0}0\bar{0} + 0\bar{0}0 + 0.0.0 = 0$$

$$F(0,0,1) = \bar{0}\bar{0}1 + \bar{0}0\bar{1} + 0\bar{0}1 + 0.0.1 = 1$$

$$F(0,1,0) = \bar{0}\bar{1}0 + \bar{0}1\bar{0} + 0\bar{1}0 + 0.1.0 = 0$$

$$F(0,1,1) = \bar{0}\bar{1}1 + \bar{0}1\bar{1} + 0\bar{1}1 + 0.1.1 = 1$$

$$F(1,0,0) = \bar{1}\bar{0}0 + \bar{1}0\bar{0} + 1\bar{0}0 + 1.0.0 = 0$$

$$F(1,0,1) = \bar{1}\bar{0}1 + \bar{1}0\bar{1} + 1\bar{0}1 + 1.0.1 = 1$$

$$F(1,1,0) = \bar{1}\bar{1}0 + \bar{1}1\bar{0} + 1\bar{1}0 + 1.1.0 = 0$$

$$F(1,1,1) = \bar{1}\bar{1}1 + \bar{1}1\bar{1} + 1\bar{1}1 + 1.1.1 = 1$$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

3.2. Priorité des opérateurs

On l'appelle aussi la *Précédence des opérateurs*. Pour évaluer une expression logique (fonction logique) on doit respecter les consignes suivantes :

1. On commence d'abord par l'évaluation des sous expressions entre les parenthèses "()".
2. Puis le complément (**NON**) et toutes les négations.
3. En suite le produit logique (**ET**).
4. Et enfin la somme logique (**OU**).

Exemple

$$F(A, B, C) = (\bar{A} \cdot \bar{B}) \cdot (C + B) + A \cdot \bar{B} \cdot C$$

Calculons $F(0,1,1)$, ce qui nous donne :

$$F(0,1,1) = (\bar{0}\bar{1})(1+1) + 0\bar{1}1$$

$$F(0,1,1) = (\bar{0})(1) + 0.0.1$$

$$F(0,1,1) = 1.1 + 0.0.1$$

$$F(0,1,1) = 1 + 0$$

$$F(0,1,1) = 1$$

Exercice : Donner la table de vérité de la fonction précédente ?

Solution

Pour trouver la table de vérité, il faut trouver la valeur de la fonction F pour chaque combinaison des trois variables A, B, C. Nous avons 3 variables donc $2^3 = 8$ combinaisons :

$$F(A,B,C) = (\overline{A \cdot B}) \cdot (C + B) + A \cdot \overline{B} \cdot C$$

$$F(0,0,0) = (\overline{0 \cdot 0}) \cdot (0 + 0) + 0 \cdot \overline{0} \cdot 0 = 0$$

$$F(0,0,1) = (\overline{0 \cdot 0}) \cdot (1 + 0) + 0 \cdot \overline{0} \cdot 1 = 1$$

$$F(0,1,0) = (\overline{0 \cdot 1}) \cdot (0 + 1) + 0 \cdot \overline{1} \cdot 0 = 1$$

$$F(0,1,1) = (\overline{0 \cdot 1}) \cdot (1 + 1) + 0 \cdot \overline{1} \cdot 1 = 1$$

$$F(1,0,0) = (\overline{1 \cdot 0}) \cdot (0 + 0) + 1 \cdot \overline{0} \cdot 0 = 0$$

$$F(1,0,1) = (\overline{1 \cdot 0}) \cdot (1 + 0) + 1 \cdot \overline{0} \cdot 1 = 1$$

$$F(1,1,0) = (\overline{1 \cdot 1}) \cdot (0 + 1) + 1 \cdot \overline{1} \cdot 0 = 0$$

$$F(1,1,1) = (\overline{1 \cdot 1}) \cdot (1 + 1) + 1 \cdot \overline{1} \cdot 1 = 0$$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

3.3. Lois fondamentales de l'Algèbre de Boole

3.3.1. L'opérateur OU

$(A+B)+C = A+(B+C) = A+B+C$	Associativité
$A+B = B+A$	Commutativité
$A+A = A$	Idempotence
$A+0 = A$	Elément neutre
$A+1 = 1$	Elément absorbant

3.3.2. L'opérateur ET

$(A \cdot B) \cdot C = A \cdot (B \cdot C) = A \cdot B \cdot C$	Associativité
$A \cdot B = B \cdot A$	Commutativité
$A \cdot A = A$	Idempotence
$A \cdot 1 = A$	Elément neutre
$A \cdot 0 = 0$	Elément absorbant

3.3.3. L'opérateur NON

$$\overline{\overline{A}} = A$$
$$\overline{\overline{A}} + A = 1$$
$$\overline{\overline{A}} \cdot A = 0$$

3.3.4. Distributivité

$A \cdot (B+C) = (A \cdot B) + (A \cdot C)$	Distributivité du ET sur le OU
$A + (B \cdot C) = (A+B) \cdot (A+C)$	Distributivité du OU sur le ET

3.3.5. Autres relations utiles

$$A + (A \cdot B) = A$$

$$A \cdot (A + B) = A$$

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

$$A + \bar{A} \cdot B = A + B$$

Optimisation 1

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

Optimisation 2

3.3.6. Dualité de l'algèbre de Boole

Toute expression logique reste vraie si on remplace le ET par le OU, le OU par le ET, le 1 par 0, le 0 par 1.

Exemple

$$A + 1 = 1 \rightarrow A \cdot 0 = 0$$

$$A + \bar{A} = 1 \rightarrow A \cdot \bar{A} = 0$$

3.3.7. Théorème de DE-MORGANE

La somme logique complimentée de deux variables est égale au produit des compléments des deux variables. $\overline{A + B} = \bar{A} \cdot \bar{B}$

Le produit logique complimenté de deux variables est égale au somme logique des compléments des deux variables. $\overline{A \cdot B} = \bar{A} + \bar{B}$

La généralisation du Théorème DE-MORGANE à N variables est :

$$\overline{A \cdot B \cdot C \dots} = \bar{A} + \bar{B} + \bar{C} + \dots$$

$$\overline{A + B + C + \dots} = \bar{A} \cdot \bar{B} \cdot \bar{C} \dots$$

3.3.8. Les opérateurs NAND et NOR sont des opérateurs universels

Ils sont considérés comme des opérateurs universels dans l'algèbre de BOOLE. En utilisant les NAND et les NOR on peut exprimer n'importe quelle expression (fonction) logique. Pour cela, Il suffit d'exprimer les opérateurs de base (NON, ET, OU) avec des NAND et des NOR.

Réalisation des opérateurs de base avec des NOR

$$\bar{A} = \overline{A + A} = A \downarrow A$$

$$A + B = \overline{\overline{A + B}} = \overline{A \downarrow B} = (A \downarrow B) \downarrow (A \downarrow B)$$

$$A \cdot B = \overline{\overline{A \cdot B}} = \overline{\bar{A} + \bar{B}} = \bar{A} \downarrow \bar{B} = (A \downarrow A) \downarrow (B \downarrow B)$$

Exercice : Exprimer le NON, ET et OU en utilisant des NAND ?

Propriétés des opérateurs NAND et NOR

$$A \uparrow 0 = 1$$

$$A \uparrow 1 = \bar{A}$$

$$A \uparrow B = B \uparrow A$$

$$(A \uparrow B) \uparrow C \neq A \uparrow (B \uparrow C)$$

$$A \downarrow 0 = \bar{A}$$

$$A \downarrow 1 = 0$$

$$A \downarrow B = B \downarrow A$$

$$(A \downarrow B) \downarrow C \neq A \downarrow (B \downarrow C)$$

3.4. Les portes logiques

Une porte logique est un *circuit électronique élémentaire* qui Permet de réaliser la fonction d'un opérateur logique de base (ET, OU, NON...) Voir **Tableau 6** - Page 54.

Remarque :

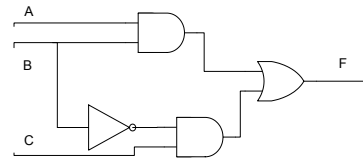
- Les portes ET, OU, NAND, NOR peuvent avoir plus que deux entrées
- Il n'existe pas de OU exclusif à plus de deux entrées

3.4.1. Schéma d'un circuit logique (Logigramme)

C'est la traduction de la fonction logique en un schéma électronique. Le principe consiste à remplacer chaque opérateur logique par la porte logique qui lui correspond.

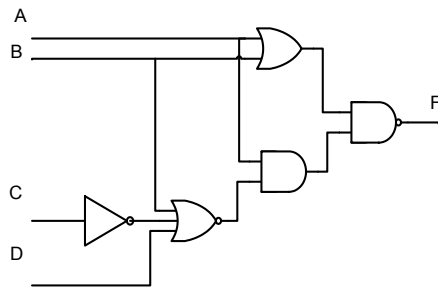
Exemple 1

$$F(A, B, C) = A.B + \bar{B}.C$$



Exemple 2

$$F(A, B, C, D) = (A + B) \cdot (B + \bar{C} + D) \cdot A$$



3.5. Etude des fonctions logiques

L'objectif de l'étude des fonctions logiques est la réalisation de circuit permettant le traitement automatique de ce type de fonction. Ainsi il faut suivre certaines étapes pour atteindre cet objectif :

- Comprendre le fonctionnement du système pour lequel nous devons réaliser le circuit.
- Définir les variables d'entrée (Une ou plusieurs).
- Définir les variables de sortie (Une ou plusieurs).
- Etablir la table de vérité.

- Ecrire les équations algébriques des sorties (à partir de la table de vérité).
- Effectuer des simplifications (algébriques ou par tableaux de **Karnaugh**).
- Faire le schéma du circuit logique (logigramme) avec un minimum de portes logiques (Optimisation).

3.5.1. Forme canonique

On appelle **Forme canonique** d'une fonction logique, la forme où chaque terme de celle-ci comporte toutes les variables constituant la fonction. Si on considère une fonction à trois variables A, B et C, Sa forme canonique peut être donnée comme suite :

$$F(A, B, C) = (\bar{A}\bar{B}C) + (\bar{A}BC) + (A\bar{B}C)$$

Il existe deux types de formes canoniques constitués de **Mintermes** et **Maxtermes** :

Mintermes : groupe des "n" variables (pouvant être complémentées) liées par des ET.

Maxtermes : groupe des "n" variables (pouvant être complémentées) liées par des OU.

Sur la base desquels on peut définir les deux formes canoniques comme suite

Première forme (Disjonctive): Représente l'union (OU) de *Mintermes* ou la somme des *Mintermes* autrement-dit.

$$F(A, B, C) = (\bar{A}\bar{B}C) + (\bar{A}BC) + (A\bar{B}C)$$

Deuxième forme (Conjonctive): Représente l'intersection (ET) de *Maxtermes* ou le produit des *Maxtermes* autrement-dit.

$$F(A, B, C) = (\bar{A} + \bar{B} + C).(\bar{A} + B + C).(A + \bar{B} + C)$$

3.5.2. Forme canonique numérique

C'est une autre représentation des formes canoniques classiques Disjonctive et Conjonctive, sous forme numérique notée :

- **R** : pour indiquer la forme disjonctive
- **P** : pour indiquer la forme conjonctive.

Exemple : $R(1,4,2) = \sum(1,4,2) = R(001,100,010) = (\bar{A}\bar{B}C) + (\bar{A}BC) + (A\bar{B}C)$

$$P(1,4,2) = \prod(2,3,6) = R(010,011,110) = (A + \bar{B} + C).(A + \bar{B} + \bar{C}).(\bar{A} + \bar{B} + C)$$

3.5.3. Passage à une Forme canonique

Quelle que soit la fonction logique, On peut toujours l'écrire dans l'une des formes canoniques. Cela revient à rajouter les variables manquantes dans les termes qui ne contiennent pas toutes les variables (les termes non canoniques), pour faire apparaître les *Mintermes* ou *Maxtermes* complets, selon la forme désirée.

Cela est possible en utilisant deux méthodes, Le calcul algébrique ou la table de vérité de la fonction.

A. Obtention de la forme canonique par calcul algébrique

Cette méthode consiste à utiliser les règles et propriétés de l'algèbre de Boole, notamment les invariants : $A.\bar{A} = 0$ et $A + \bar{A} = 1$

Ce qui consiste simplement à :

- Multiplier un terme avec une expression qui vaut 1.
- Additionner un terme avec une expression qui vaut 0.
- Par la suite faire la distribution.

Exemple 1:

Soit la fonction à trois variables suivante : $F(A, B, C) = A.B. + \bar{B}.C + A.\bar{C}$

On remarque que l'équation est composée de **Mintermes non canonique**, où chaque terme se compose de deux variables. Donc, on doit rajouter à chaque terme la variable manquante pour écrire la fonction précédente sous la forme canonique disjonctive, et ce en procédant comme suite :

1. Commencant par le premier terme "A.B", il manque la variable C.
2. On transforme A.B comme suite :

$$A.B = A.B.(C + \bar{C}) \quad / \quad \text{Car } (C + \bar{C}) = 1$$

3. On transforme les autres termes de la même manière pour obtenir :

$$F(A, B, C) = A.B.(C + \bar{C}) + \bar{B}.C.(A + \bar{A}) + A.\bar{C}.(B + \bar{B})$$

$$F(A, B, C) = A.B.C + A.B.\bar{C} + A.\bar{B}.C + \bar{A}.\bar{B}.C + A.\bar{B}.\bar{C}$$

Exemple 2:

Concevant la même fonction pur la transformer en deuxième forme canonique (Conjonctive). Cette fois on utilise la propriété : $\overline{\bar{F}} = F$

1. On commence par le développement du complément (négation) de la fonction:

$$\overline{F(A, B, C)} = \overline{A.B + \bar{B}.C + A.\bar{C}}$$

$$\text{Ce qui nous donne : } \overline{F(A, B, C)} = \bar{A}.B + \bar{A}.B.C + \bar{A}.\bar{C} + \bar{A}.\bar{B}.\bar{C}$$

2. On transforme ensuite les termes non canoniques (terme à deux variables) :

$$\bar{A}.B + \bar{A}.\bar{C} = \bar{A}.B.(C + \bar{C}) + \bar{A}.\bar{C}.(B + \bar{B})$$

$$\text{On obtient : } \overline{F(A, B, C)} = \bar{A}.B.C + \bar{A}.\bar{B}.\bar{C} + \bar{A}.\bar{B}.\bar{C}$$

3. On développe ensuite le complément du résultat obtenu :

$$\overline{\overline{F(A, B, C)}} = \overline{\bar{A}.B.C + \bar{A}.\bar{B}.\bar{C} + \bar{A}.\bar{B}.\bar{C}}$$

$$\text{Le résultat est : } F(A, B, C) = (A + \bar{B} + \bar{C}).(A + B + C) + (A + \bar{B} + C)$$

B. Obtention de la forme canonique par la table de vérité

Cette méthode consiste à :

1. Tracer la table de vérité de la fonction F.
2. Noter les *Mintermes* et *Maxtermes* directement sur la table de vérité tel que
 Si $F(X_1, X_2, \dots, X_n) = 1$ alors $(X_1 \cdot X_2 \dots X_n)$ est un *Mintermes*,
 Qui est noté comme suite : Si $X_i = 1$ On note X_i Sinon on le note \bar{X}_i .
 Si $F(X_1, X_2, \dots, X_n) = 0$ alors $(X_1 + X_2 + \dots + X_n)$ est un *Maxtermes*,
 Qui est noté comme suite : Si $X_i = 1$ On note \bar{X}_i Sinon on le note X_i .
3. La forme Disjonctive sera le OU des *Mintermes* et la forme Disjonctive sera le ET des *Maxtermes*.

Exemple : Soit la fonction à trois variables A, B, C : $F(A, B, C) = A.B + \bar{B}.C + A.\bar{C}$

La table de vérité de cette fonction est la suivante :

A	B	C	\bar{B}	\bar{C}	A.B	$\bar{B}.C$	A. \bar{C}	F(A,B,C)	Termes canoniques
0	0	0	1	1	0	0	0	0	Maxterme $A + B + C$
0	0	1	1	0	0	1	0	1	Minterme $\bar{A}.\bar{B}.C$
0	1	0	0	1	0	0	0	0	Maxterme $A + \bar{B} + C$
0	1	1	0	0	0	0	0	0	Maxterme $A + \bar{B} + \bar{C}$
1	0	0	1	1	0	0	1	1	Minterme $A.\bar{B}.\bar{C}$
1	0	1	1	0	0	1	0	1	Minterme $A.\bar{B}.C$
1	1	0	0	1	1	0	1	1	Minterme $A.B.\bar{C}$
1	1	1	0	0	1	0	0	1	Minterme $A.B.C$

Forme Disjonctive : $F(A, B, C) = A.B.C + A.B.\bar{C} + A.\bar{B}.C + \bar{A}.\bar{B}.C + A.\bar{B}.\bar{C}$

Forme Conjonctive : $F(A, B, C) = (A + \bar{B} + \bar{C}).(A + B + C) + (A + \bar{B} + C)$

Remarque

En utilisant cette méthode, on peut aussi déduire la deuxième forme canonique (Conjonctive), en déterminant les *Mintermes* lorsque la fonction $F = 0$. Ensuite on calcul le complément du résultat pour obtenir la deuxième forme canonique.

En reprenant l'exemple précédent on remarque que :

$$F(0,0,0) = 0 \text{ Minterme } \bar{A}.\bar{B}.\bar{C}$$

$$F(0,1,0) = 0 \text{ Minterme } \bar{A}.B.\bar{C}$$

$$F(0,1,1) = 0 \text{ Minterme } \bar{A}.B.C$$

$$\text{Donc : } F(A, B, C) = (\bar{A}.\bar{B}.\bar{C}) + (\bar{A}.B.\bar{C}) + (\bar{A}.B.C)$$

Le complément de F nous donne la forme Conjonctive de la fonction F :

$$\begin{aligned} \overline{F(A, B, C)} &= \overline{(\bar{A}.\bar{B}.\bar{C}) + (\bar{A}.B.\bar{C}) + (\bar{A}.B.C)} \\ &= F(A, B, C) = (A + \bar{B} + \bar{C}).(A + B + C) + (A + \bar{B} + C) \end{aligned}$$

3.5.4. Simplification des fonctions logiques

La simplification des fonctions logiques consiste à *minimiser les termes et les variables* utilisés dans la formule de la fonction. Ainsi la représentation canonique des fonctions est correcte, mais elle peut être simplifiée.

L'objectif principal est de faciliter la réalisation de ces fonctions avec le minimum de composants électroniques (Portes logiques). Pour ce cela il existe deux méthodes de simplification :

- **Méthode Algébrique** : En utilisant les propriétés de l'Algèbre de BOOLE.
- **Méthode Graphique** : En utilisant le tableau de KARNAUGH.

3.5.5. La simplification Algébrique

Cette méthode consiste à utiliser les règles et propriétés de l'Algèbre de Boole, pour transformer la fonction en une formule plus simple. Le principe est d'utiliser les propriétés telles que *l'involution* ($\overline{\overline{X}} = X$), *l'élément absorbant*, *l'idempotence*...etc. Notant que l'utilisation de *l'involution* entraîne souvent un long calcul.

Avec l'évolution du processus de simplification d'autres simplifications peuvent être développées.

Exemple 1 : Soit $F(A, B, C) = A \cdot B \cdot C + A \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + \overline{A} \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot \overline{C}$

En factorisant on obtient :

$$F(A, B, C) = A \cdot (B \cdot (C + \overline{C}) + \overline{B} \cdot (C + \overline{C})) + \overline{A} \cdot \overline{B} \cdot C \quad \text{Sachant que } C + \overline{C} = 1$$

$$F(A, B, C) = A \cdot (B \cdot (1) + \overline{B} \cdot (1)) + \overline{A} \cdot \overline{B} \cdot C \quad \text{De même } B + \overline{B} = 1$$

$$F(A, B, C) = A + \overline{A} \cdot \overline{B} \cdot C \quad \text{Sachant que } X + \overline{X}Y = X + Y$$

La fonction simplifiée sera : $F(A, B, C) = A + \overline{B} \cdot C$

Exemple 2 : Soit $F(A, B, C) = \overline{(A + B)} \cdot \overline{C} + \overline{B} \cdot C$

Commençant par le calcul de la négation (le NON), *sans faire la distribution* :

$$F(A, B, C) = \overline{A} \cdot \overline{B} + \overline{C} + \overline{B} \cdot C$$

$$\overline{F(A, B, C)} = B \cdot C \quad \text{En utilisant l'Involution}$$

$$\overline{\overline{F(A, B, C)}} = \overline{B} + \overline{C} \quad \text{En utilisant l'Involution}$$

La fonction simplifiée sera : $F(A, B, C) = \overline{B} + \overline{C}$

Selon les exemples précédents nous remarquons que c'est plus pratique de réduire la formule d'une fonction logique. Bien qu'il n'y a pas de démarches spécifiques pour la simplification des fonctions logiques, mais un certain nombre de règles de simplification peuvent être utiles et faciliter la tâche dans ce cas (*la simplification algébrique*) :

Règle 1 : Regroupement des termes, par exemple :

$$\begin{aligned}
 F &= A.B.C + A.B.\bar{C} + A.\bar{B}.C.D \\
 F &= A.B.(C + \bar{C}) + A.\bar{B}.C.D && (C + \bar{C}) = 1 \\
 F &= A.B. + A.\bar{B}.C.D \\
 F &= A.(B + \bar{B}.(C.D)) && X + \bar{X}Y = X + Y \\
 F &= A.(B + C.D) && \text{Par distribution} \\
 \mathbf{F} &= \mathbf{A.B + A.C.D}
 \end{aligned}$$

Règle 2 : Rajouter un terme déjà existant à l'expression :

$$\begin{aligned}
 F &= A.B.C + \bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C} \\
 F &= A.B.C + \bar{A}.B.C + A.B.C + A.\bar{B}.C + A.B.C + A.B.\bar{C} \\
 \mathbf{F} &= \mathbf{B.C + A.C + A.B}
 \end{aligned}$$

Règle 3 : Supprimer un terme superflu (terme en plus) :

$$\begin{aligned}
 F &= A.B + \bar{B}.C + A.C \\
 F &= A.B + \bar{B}.C + A.C.(B + \bar{B}) && (B + \bar{B}) = 1 \\
 F &= A.B + \bar{B}.C + A.B.C + A.\bar{B}.C && \text{Distribution} \\
 F &= A.B.(1 + C) + \bar{B}.C.(1 + A) && (1 + A) = 1 \\
 \mathbf{F} &= \mathbf{A.B + \bar{B}.C}
 \end{aligned}$$

Règle 4 : Choix de la forme à simplifier

Il est souvent préférable de simplifier la forme canonique ayant le nombre minimum de termes comparant à l'autre forme. Pour simplifier l'exemple on prendra la représentation numérique des formes canoniques comme suite :

$F = R(2,3,4,5,6,7)$ Cette forme contient six (6) termes donc il est préférable de simplifier la deuxième forme contenant deux (2) termes seulement.

$$\begin{aligned}
 \bar{F} &= R(0,1) = \bar{A}.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C \\
 &= \bar{A}.\bar{B}.(C + \bar{C}) \\
 &= \bar{A}.\bar{B} \\
 \bar{F} &= \overline{A + B} \\
 \mathbf{\bar{\bar{F}} = F} &= \mathbf{A + B}
 \end{aligned}$$

Du fait de leur unicité, les formes normales (*Disjonctive et Conjonctive*) permettent de vérifier très rapidement l'égalité des équations booléennes. Mais, en faisant intervenir tous les termes, elles deviennent très lourdes à manier. Il convient donc de les simplifier pour pouvoir les exploiter. On se rend assez vite compte, qu'une simplification à l'aide des règles usuelles est très *fastidieuse*, et source de nombreuses erreurs. Ce qui convient donc à étudier un autre outil qui soit puissant pour simplifier les tables de vérités complexes : *le tableau de Karnaugh*.

3.5.6. La simplification Graphique

Cette méthode consiste à présenter les états d'une fonction logique, non sous la forme d'une table de vérité, mais en utilisant une nouvelle forme particulière de tableau à double entrée (Deux dimensions) dit : **Tableau de KARNAUGH**. L'objectif est de déployer un outil puissant, simple et rapide de simplification des fonctions logiques en évitant la simplification algébrique.

Le tableau de *Karnaugh* n'est rien d'autre qu'une forme particulière de la table de vérité, où chaque case du tableau correspond à une combinaison des variables d'entrées, donc à une ligne de la table de vérité, le tableau de *Karnaugh* contient autant de cases que la table de vérité possède de lignes.

$$\text{Nombre de lignes (Table de vérité)} = \text{Nombre de Cases (Tableau de Karnaugh)}$$

NB / Le rôle du tableau de Karnaugh n'est pas seulement cerné dans la simplification des fonctions logiques mais il nous permet aussi de déduire directement une fonction simplifiée à partir de la table de vérité d'un problème donné.

A. Principes généraux

1. **Sources** : Avant de passer au tableau de Karnaugh, il faut avoir en premier la fonction logique (sous forme brute ou canonique), ou bien la table de vérité d'un problème logique.
2. **Forme** : On commence par la représentation d'un tableau à deux dimensions en tenant compte des nombre de variables d'entrée (1, 2, 3...), pour connaître la taille du tableau.
3. **Indices** : Numéroté les lignes et les colonnes du tableau selon le **code binaire réfléchi (Code Gray)**, donc chaque fois que l'on passe d'une case à l'autre, une seule variable change d'état.

On peut numéroté les cases pour que ce soit plus facile à remplir, mais il faut faire attention à l'ordre de numérotation ! (Voir l'exemple)

4. **Regroupements** : Il s'agit d'effectuer les regroupements de cases **adjacentes** qui consiste à déterminer les **blocs rectangulaires adjacents** de taille 2^n Bits (1, 2, 4, 8...) dont la valeur des cases est 1.
5. **Déduction** : En fin, On peut déduire la fonction simplifiée dont les termes adjacents sont obtenus à partir des blocs rectangulaires adjacents et la fonction sera le OU de ces termes.

NB/ On dit que deux termes sont adjacents si seulement une variable change d'état (Valeur) entre les deux termes :

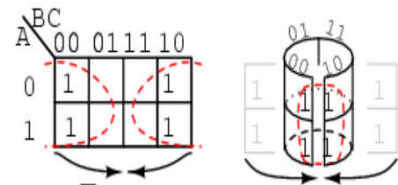
$\bar{A}.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C$ Sont des termes adjacents,
Car seulement la variable C qui change de valeur

$A.B + A.\bar{B}$ Sont des termes adjacents,
Car seulement la variable B qui change de valeur

B. Caractéristiques et méthodologie

1. La Forme :

- Chaque dimension du tableau de Karnaugh concerne une (1) ou deux (2) variables.
- Par principe, le passage d'une ligne à une autre ou d'une colonne à une autre modifie la valeur d'une seule variable.
- Le tableau se referme sur lui-même en forme de cylindre, la colonne la plus à gauche est voisine avec la colonne la plus à droite ainsi sont les lignes du haut et du bas.
- Ainsi, pour les 2 colonnes (2 lignes) extrêmes, là aussi, une seule variable doit changer de valeur entre ces 2 colonnes (lignes).
- Chaque case du tableau contient une valeur reproduite à partir de la table de vérité (Valeur de la fonction F, relative aux valeurs des variables d'entrée).



2. Regroupements :

- Tous les bits à 1 du tableau doivent être englobés dans au moins un bloc de taille 1, 2, 4, 8 ... bits.
- Un bit à 1 peut appartenir à plusieurs blocs.
- On doit créer les blocs les plus gros possibles.

3. Déduction :

Après les regroupements, on passe à l'extraction des termes correspondant comme suite :

- On ne prend pas en compte une variable qui change de valeur dans le même bloc.
- On ne conserve que les variables qui ne changent pas de valeurs dans le même bloc, tel que Si une variable A reste à **1** on la note **A**. Si elle reste à **0** on la note \bar{A} .

- Le terme logique correspondant à un bloc représente le **ET** de ces variables.
- La fonction logique simplifiée est simplement le **OU** de ces termes correspondant aux blocs.

C. Exemples

Ci après quelques exemples de tableaux de Karnaugh à 2, 3 et 4 variables.

a. Cas de deux (2) variables :

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

		A	
		0	1
B	0	0	1
	1	1	1

Dans le premier exemple où nous avons deux (2) variables, nous remarquons que:

- Nous avons deux (2) groupes de deux bits adjacents. Verticalement nous avons toujours $A=1$ donc on a le terme A . Horizontalement nous avons toujours $B=1$ donc on a le terme B .
- La fonction obtenue sera donc $F = A+B$.

b. Cas de trois (3) variables :

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

		AB			
		00	01	11	10
C	0	0	0	1	1
	1	1	0	1	1

- Tout d'abord, nous avons un grand bloc de 4 bits, La variable A reste à mais les deux autres variable B et C change de valeurs. Donc le terme retenu sera seulement A .
- Nous avons aussi un bloc minimal (*contenant un seul bit*) $A=0, B=0, C=1$. Le terme correspondant est $\bar{A}.\bar{B}.C$.

Mais : Nous savons que le tableau se referme sur lui-même donc le bit le plus à gauche et adjacent au bit le plus à droite. Ainsi on peut regrouper des bits même s'ils appartiennent à plusieurs blocs. Cela nous donne la forme du regroupement suivant :

		AB			
		00	01	11	10
F	C	0	0	1	1
	1	1	0	1	1

La variable B reste à 0, la variable C reste à 1, alors que la variable A change de valeur, donc le terme obtenu est : $\bar{B}.C$

- La fonction obtenue sera donc $F = A + \bar{B}.C$

c. Cas de Quatre (4) variables :

Dans cette exemple, Nous donnons directement le tableau de Karnaugh sans illustrer la table de vérité.

- Pour les même raisons que l'exemple précédent, nous obtiendrons trois blocs :

Bloc à deux (2) bits : $\bar{A}.B.C$

Bloc à quatre (4) bits : $\bar{B}.\bar{C}$

Bloc à huit (8) bits : D

		AB			
		00	01	11	10
F	CD	00	01	11	10
	01	1	1	1	1
	11	1	1	1	1
	10	0	1	0	0

- La fonction obtenue sera donc $F = \bar{A}.B.C + \bar{B}.\bar{C} + D$.

D. Cas des fonctions non totalement définies

Une fonction non totalement définie, est une fonction logique dont certaines combinaisons de variables d'entrée ne donnent aucun résultat (Valeur) à la fonction. On dit que l'état (Valeur) de la fonction dans ces cas est *indéterminé*. Ainsi les combinaisons relatives sont dites "*Interdites ou Impossible*".

Exemple : Soit une serrure de sécurité s'ouvre en fonction de quatre clés A, B, C, D. Le fonctionnement de la serrure est définie comme suite :

$S(A,B,C,D) = 1$ si au moins deux clés sont utilisées. $S(A,B,C,D) = 0$ sinon.

Les clés A et D ne peuvent pas être utilisées en même temps.

Dans le ou les cas où les clés A et D sont introduites, on ne peut pas savoir quelle est la valeur de la fonction **S** ! On est dans un état indéterminé.

Solution :

Dans les cas impossible on met un **X** au lieu de 1 ou 0 dans la table de vérité de la fonction **S**. Il en sera de même pour le tableau de Karnaugh. En considérant les X comme "1" cela nous donne le Tableau de Karnaugh suivant :

(Le regroupement est illustré étape par étape).

A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	X
1	0	1	0	1
1	0	1	1	X
1	1	0	0	1
1	1	0	1	X
1	1	1	0	1
1	1	1	1	X

		AB			
		00	01	11	10
S	00	0	1	0	0
	01	X	X	1	0
	11	X	X	1	1
	10	1	1	1	0

$S = A.B + CD$

		AB			
		00	01	11	10
F	00	0	1	0	0
	01	X	X	1	0
	11	X	X	1	1
	10	1	1	1	0

$S = A.B + CD + B.D + B.C$

		AB			
		00	01	11	10
F	00	0	1	0	0
	01	X	X	1	0
	11	X	X	1	1
	10	1	1	1	0

$S = A.B + CD + B.D + B.C + \bar{A}.C$



Conclusion Générale

Le présent cours est une contribution à la présentation globale des composants de systèmes informatiques, le codage de l'information et l'algèbre de boole dès le début d'un cursus de formation des étudiants de la 1^{ère} année universitaire en MI.

Cet enseignement doit bien entendu être prolongé dans la Licence et Master par les modules d'approfondissement classiques notamment la structure machine, le système d'exploitation, le traitement automatique de l'information et la résolution logique des problèmes, etc. qui permettent de passer d'une vision essentiellement conceptuelle et externe à une vision technique et interne, indispensable aux futurs professionnels du domaine MI.

Avec cinq séries exercices, ce cours est un simple outil d'apprentissage et de référence.



Références bibliographiques

- [1]. *"Initiation à l'informatique"*.
St. Herblain, Éditions ENI, 2009.
- [2]. *"Initiation à l'informatique et à l'ordinateur"*.
Mahmadou Issoufou Tiado, Paris : Éd. l'Harmattan, 2014.
- [3]. *"Systèmes informatiques"*.
Olivier Lecarme, Licence Mathématiques-Informatique, Semestre 2, 2005–2006.
- [4]. *"Introduction aux systèmes informatiques"*.
Jacques Lonchamp, 2015.
- [5]. *"Architecture des machines et des systèmes informatiques"*.
Alain Cazes; Joëlle Delacroix, Dunod, 2015, cop. 2015.
- [6]. *"Architecture et technologie des ordinateurs"*.
P Zanella; Yves Ligier; Emmanuel Lazard. Dunod, 2013.
- [7]. *"Systèmes d'exploitation"*.
Collection: Les sélections : technologies de l'information : technologies logicielles,
Techniques de l'ingénieur, impr. 2016.
- [8]. *"Architecture des systèmes / Techniques de l'ingénieur". 2e éd*
Systèmes d'exploitation: principes et fonctions
Sacha Krakowiak, Techniques de l'ingénieur, 1996.
- [9]. *"Systèmes d'exploitation: Les concepts essentiels expliqués aux débutants"*.
Mohamed Said Ouerghi. Editions universitaires europeennes EUE, 2011.
- [10]. *"Informatique commerciale"*.
Jean-François Dhénin; et al. Rosny-sous-Bois : Bréal éd., 2004.
- [11]. *"Théorie de l'information : application aux techniques de communication"*.
Gérard Battail. Masson, 1997.

- [12]. *"Introduction aux sciences de l'information : entropie, compression, chiffrement et correction d'erreurs"*. Jean-Yves Le Boudec; Patrick Thiran; Rüdiger Urbanke. Presses polytechniques et universitaires romandes, 2015, cop. 2015.
- [13]. *"Fondements de la théorie de la transmission de l'information"*. Spătaru Alexandru. Paris : Presses polytechniques romandes, 1997.
- [14]. *"Codage et représentation de l'Information"*. Taha ZERROUKI. Université de Bouira, Mathématique et informatique. 2015.
- [15]. *"Codage de l'Information"*. Mikael.Salson. <http://www.fil.univ-lille1.fr/~salson/codage/Poly/poly.pdf> univ-lille1.fr. 2015.
- [16]. *Algèbre de Boole et machines logiques*. Pierre Naslin. Dunod, 1967.
- [17]. *"Algèbre de Boole, schémas électriques automatismes"*. R. Clément, François Degoullange. Dunod, 1968.
- [18]. *"L'Algèbre binaire de Boole et ses applications à l'informatique"*. Raoul de Palma. Dunod, 1971.
- [19]. *"Algèbre de Boole"*. Eric Cariou. Université de Pau et des Pays de l'Adour. UFR Sciences Pau - Département Informatique. 2012.



Annexe

Série d'exercices N° 1

Exercice 01. Rappel de cours.

1. Donnez une définition convenable aux mots suivants :
- Ordinateur - Informatique - Programme - Information
2. Quels sont les systèmes que l'on peut qualifier comme Système Informatique ?
Donner des exemples, expliquer ?
3. Déterminer les choses qui représentent des informations:
Fichier, SMS, MMS, email, flash disque, signal satellite, ondes radio, image, son, vibration, carte mémoire, vidéo, lumière, Bruit, mouvement physique, accident.

Exercice 02. Mémoires.

1. Classer les mémoires suivantes en mémoires principales (internes), mémoires secondaires (externes):
Carte mémoire, flash disque, Mp3 reader, DVD, CD, VCD, ROM, RAM, SDRAM, DDRAM, disque dur, EPROM, Disque SSD.
2. Que signifient les termes R.A.M. et R.O.M. ? Où utilise-t-on de la ROM ?
3. Déterminer les propriétés de chaque mémoire (RAM / ROM)
 - Lecture seule des données,
 - Lecture/écriture des données,
 - Mémoire morte
 - Mémoire vive
 - Volatile
 - Permanente
 - Random Access Memory
 - Read Only Memory
4. Dans une configuration classique, où sont stockés vos fichiers après l'extinction complète d'un ordinateur ?

5. Classez les mémoires suivantes par taille, par rapidité : RAM, registres, disques durs, cache L1, cache L2, cd-rom.
6. Pourquoi utilise-t-on des mémoires caches ?

Exercice 03. Périphériques.

1. Classer les périphériques suivants en Entrée, Sortie, Entrée/Sortie:

Carte réseau	Ecran tactile	Mannette de jeux	Lecteur/graveur de cd/dvd
Clavier	Imprimante	MODEM	Lecteur Mp3
Clé USB	Lecteur de bande	MODEM wifi	Table traçante
Data Show	Lecteur disquette	Scanner	Lecteur zip
Disque dur		Souris	Ecran

Exercice 04. Carte mère.

1. Que représente une carte mère dans un système informatique
 - Périphérique de montage
 - Système nerveux
 - Dispositif de liaison et interconnexion
 - Support de traitement de l'information
2. La carte mère contient un certain nombre d'éléments embarqués, c'est-à-dire intégrés sur son circuit imprimé. Citez tous ces éléments.
3. Quelles sont les fonctions d'un Chipset sur une carte mère d'ordinateur ?
4. Citez les différentes interfaces (connecteurs et branchements) qui se trouvent à l'intérieur de la machine avec une brève définition de chaque connecteur.
5. Pourquoi le BIOS est-il un programme en mémoire non-volatile ?

Série d'exercices N° 2

Partie I. **Partie Software.**

Exercice 01. **Notions de base**

1. Quelle est la différence entre un programme et un logiciel ?
2. Quelle est la différence entre un processus un programme ?
3. Quelle est la relation entre Processeur et processus ?
4. Le système d'exploitation dans son état brut est stocké sur ?
5. Au démarrage du PC, le SE sera chargé dans ?
6. Quelles sont les tâches principales d'un SE ?
7. Quelle est le composant du SE qui est invoqué (utilisé) dans un système multitâches ?
8. Expliquer le fonctionnement dans le cas d'un seul CPU et dans le cas de multi CPU ?
9. Quelle est la différence entre un logiciel *Libre* et *Gratuit* ?
10. Dans le cas où la mémoire ne peut pas contenir les programmes qui sont plus grands que l'espace mémoire disponible, quel est le mécanisme que le SE utilise pour gérer cette situation ?

Exercice 02. **Installer un disque dur**

1. Identifier les connecteurs d'un disque dur et les connecteurs correspondants d'une carte mère.
2. Connecter correctement un disque dur en primary master.
3. Définir le disque dur au niveau de l'architecture de la machine (setup).
4. Booter sur une disquette système afin de tester le fonctionnement et l'accès au disque dur.

Booter à partir d'un CD bootable pour lancer l'installation Windows puis Linux

5. Ajouter un disque dur à un système existant (primary slave, secondary master ou secondary slave).
-

Série d'exercices N° 3

Partie I. Les unités de mesure de l'information

Exercice 01.

L'information est un ensemble d'événements qui peuvent être communiqué à l'ordinateur.

Pour apprécier l'information on doit connaître les mécanismes de mesure de cette information.

1. Définir les unités de mesures de l'information (Capacité/Taille, Débit, Fréquence).
2. Convertir les unités suivantes :

8.5 Go =	Ko =	bits.
3,4 GHz =	MHz =	Hz.
1 Mbps =	Kops =	ops.
2 To =	Go =	Mo.

Exercice 02.

En définissant un rythme de transmission de 1000 T/s, dans une machine utilisant des mots mémoire de 16 bits et en transmettant 3 mots à chaque transfert :

- 1.1. Combien faut-il de temps pour copier un fichier de 3.24 Mo du Flashdisk vers le DD ?
- 1.2. Quel est le débit de transmission exprimé en Ko/s ?
- 1.3. Combien de mots mémoires de 16 bits faut-il envoyer à chaque transfert pour atteindre un débit de 1 Mbps ?

Un disque dur possède un taux de transfert de 61 Mo/s, sachant qu'il utilise un *baud* de **IMT/s** :

- 1.4. Quelle est la taille du paquet d'informations envoyé à chaque transfert en bit ?
- 1.5. Quel est la taille du paquet nécessaire pour copier un CD-Rom dans 2 minutes ?

Partie II . Codage de l'information

Exercice 01.

Donner le tableau de correspondance des 17 premier nombre entiers dans les bases 2, 4, 6, 8, 10 et 16.

Exercice 02.

Effectuer les conversions suivantes :

$(512)_{10} = (\dots)_8$	$(2167)_8 = (\dots)_{16}$	$(1432.45)_8 = (\dots)_{10}$
$(0,A85)_{10} = (\dots)_8$	$(10110101)_2 = (\dots)_{10}$	$(11101.101)_2 = (\dots)_{16}$
$(753)_8 = (\dots)_2$	$(92)_{16} = (\dots)_8$	$(13.25)_{10} = (\dots)_{16}$
$(84)_{10} = (\dots)_3$	$(139)_{10} = (\dots)_5$	$(305)_{10} = (\dots)_2$
$(13)_9 = (\dots)_{10}$	$(83)_8 = (\dots)_{10}$	$(FA)_{16} = (\dots)_{10}$
$(1982)_9 = (\dots)_{12}$	$(261)_7 = (\dots)_5$	$(13)_4 = (\dots)_3$
$(0,13)_9 = (\dots)_{10}$	$(0,83)_8 = (\dots)_{10}$	$(0,FA)_{16} = (\dots)_{10}$

Exercice 03.

Donner une représentation binaire des nombre suivants :

$$(512,15)_{10} \quad (62,75)_8 \quad (78,23)_9 \quad (EC,0D)_{16} \quad (163,82)_{10} \quad (12,102)_3$$

Exercice 04

Donner la base dans laquelle ces nombre sont exprimés :

$$(1A0)_x = (416)_{10} \quad (70)_x = (56)_{10} \quad (13)_x = (7)_{10} \quad (24)_x = (14)_{10}$$

Exercice 05

Convertir les nombres binaires suivants, en Octal (8), Décimal (10) et Hexadécimal (16) :

111111101000010100110	111101100011010001
110010101100011010001	111010100001100101101
111111111111110011111	000011101001101100111

Exercice 6

Effectuer les différentes opérations dans leurs bases indiquées :

Base 2 :	100101+101;	11001 + 1011;	111111+1.
Base 8 :	675 + 324;	125 + 246;	333 - 201.
Base 16 :	FA,675 + 324;	ED,125 + A0,246;	C,333 - 0,701.

Exercice 7

Effectuer les différentes opérations dans base 2:

10101101 * 1000	101011110 *	101 10111011 * 1101
10101101 ÷ 10	101011110 ÷ 110	10111011 ÷ 101

Exercice 8

Quel est le plus grand nombre qu'on peut représenter sur 12, 16, 24, 32, 48 bits ?

Donner les nombres (signés et non signés) qu'on peut représenter sur 12, 16, 24, 32, 48 bits ?

Exercice 9

Donner les représentations binaires en SAV, C1 et C2 des nombres suivant :

$$1, 2, 3, 16, 19, -1, -2, -3, -4, -16, 127.$$

Exercice 10

Effectuer l'opération suivante sur 8 bits :

$$1111 1110 + 10 = ? \text{ Discuter ?}$$

Série d'exercices N° 4

Suite . Codage de l'information

Exercice 11

Reproduire les nombres binaires suivants en décimal selon la méthode du codage :

Valeur absolue :	10001010	00001100	10000001
Complément à 1:	11110101	01110011	01111110
Complément à 2:	11110110	01110011	11111101

Exercice 12

Convertir en binaire :

15.23 12.25 63.875

Convertir en décimal :

0,11010 1101,101 100011001,00101101

Exercice 13

Convertir en binaire en utilisant la norme IEEE 754 (32 bits):

15.23 12.25 63.875

Exercice 14

Sachant qu'ils sont codés sous la norme IEEE 754 (32 bits), convertir en binaire les nombres suivant :

1 00110101, 11100101011101100100100

0 10110101, 10000101110101000000100

1 00010111, 0010010001000000000000

Exercice 15

- Coder en ASCII standard la phrase suivante : (donner le code en hexadécimal)

"Une graine d'idée...Un champ d'innovation"

- Coder votre nom en ASCII.

Exercice 16

- Coder en ASCII standard la phrase suivante : (donner le code en hexadécimal)

"Une graine d'idée...Un champ d'innovation"

- Coder votre nom en ASCII

Exercice 17

Déchiffrer le code ASCII suivant :

4C 61 20 43 69 67 61 6C 65 20 65 74 20 60 61 20

46 6F 75 72 6D 69 0A 0A 4C 61 20 43 69 67 61 6C

Série d'exercices N° 5

Exercice 1

- Montrer comment l'opérateur **ET** peut être obtenu à partir des opérateurs **OU** et **NON**. De même pour l'opérateur **OU** avec les opérateurs **ET** et **NON**.
- On note respectivement les opérateurs **OU**, **ET**, **XOR** et **NON** par "+", ".", "⊕" et "¬".
Montrer à l'aide de tables de vérité que $A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$ et que $A \oplus B = (A + B) \cdot (\bar{A} + \bar{B})$.
- Montrer que $A + (\bar{A} \cdot B) = A + B$ et que $A \cdot (\bar{A} + B) = A \cdot B$.

Exercice 2

Sachant que A, B et C sont des variables booléennes, démontrer que :

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

$$\bar{A} \cdot B \oplus \bar{A} \cdot \bar{B} = \bar{A}$$

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

Exercice 3

Donner le complément des Fonctions suivantes :

$$F = A + \bar{B} \cdot C$$

$$T = A \cdot B + B \cdot C + A \cdot C$$

$$S = \bar{C} \cdot \bar{D} + \bar{A} \cdot \bar{B} + C \cdot \bar{D} + A \cdot \bar{B}$$

$$A \cdot \bar{B}$$

$$G = \bar{A} \cdot \bar{B} + A \cdot B + A \cdot \bar{B}$$

$$H = \bar{C} \cdot D + \bar{A} \cdot B + C \cdot D + A \cdot B$$

$$L = \bar{C} \cdot \bar{D} + A \cdot \bar{B}$$

Exercice 4

A partir de la table de vérité suivante:

- Déduire les **Mintermes** et **Maxtermes**.
- Ecrire les deux formes canoniques de la fonction F.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Exercice 5

Ecrire sous les deux formes canoniques "*Conjunctive* et *Disjunctive*", les Fonctions suivantes

$$F = A + \bar{B} \cdot C$$

$$T = A \cdot B + B \cdot C + A \cdot C$$

$$S = \bar{C} \cdot \bar{D} + \bar{A} \cdot \bar{B} + C \cdot \bar{D} + A \cdot \bar{B}$$

$$A \cdot \bar{B}$$

$$G = \bar{A} \cdot \bar{B} + A \cdot B + A \cdot \bar{B}$$

$$H = \bar{C} \cdot D + \bar{A} \cdot B + C \cdot D + A \cdot B$$

$$L = \bar{C} \cdot \bar{D} + A \cdot \bar{B}$$

Exercice 6

Simplifier en utilisant les propriétés de l'Algèbre de Boole les fonctions suivantes :

$$F1 = A \cdot (A + B)$$

$$F5 = (A + \bar{B}) \cdot C + \bar{A} \cdot (\bar{B} + C) + \bar{B}$$

$$F3 = A \cdot B + \bar{C} + C(\bar{A} + \bar{B})$$

$$C(\bar{A} + \bar{B})$$

$$F4 = (A \cdot \bar{B} + C) \cdot (A + \bar{B}) \cdot C \quad F2 = (A + B) \cdot (\bar{A} + B)$$

$$F6 = (A + B + C) \cdot (\bar{A} + B + C) + A \cdot B + B \cdot C$$