MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE UNIVERSITÉ MOULOUD MAMMERI TIZI-OUZOU



FACULTÉ DE TECHNOLOGIE
DÉPARTEMENT TECHNOLOGIE

Présentée par
Dr. Mohamed DJOUDER

Cours

Structure des Ordinateurs et Applications

SOMMAIRE

I. CHAPITRE I : INTRODUCTION À L'INFORMATIQUE 1

I.1 Objectifs du chapitre 1

I.2 Définition de l'Informatique 2

I.3 Les ordinateurs 2

I.4 Un aperçu sur l'architecture d'un ordinateur 4

I.4.1 Micro-processeurs 5

I.4.2 Les Périphériques 5

I.5 Les systèmes de numération 7

I.6 Conversion d'un nombre d'un système à un autre 8

I.6.1 Conversion binaire en décimal 8

I.6.2 Conversion décimal - binaire d'un nombre entier 9

I.6.3 Conversion de la base B (2, 8, 16) vers la base 10 10

I.6.4 Conversion décimal vers une base B quelconque(B10 vers B(2,8,16)) 11

I.6.5 Conversion de la base 8 vers la base 2 12

I.6.6 Conversion de la base 16 vers la base 2 12

I.6.7 Conversion de la base 2 vers la base 8 13

I.6.8 Conversion de la base 2 vers la base 16 14

I.6.9 Conversion de la base 16 vers la base 8 14

I.7 Opérations arithmétiques de base 14

I.7.1 Opérations arithmétique en binaire (Base 2) 15

I.8 Définition 16

I.9 Propriétés de l'algèbre de Boole 16
I.10 Portes Logiques 18
I.10.1 La Négation 18
I.10.2 . La Conjonction 19
I.10.3 La Disjonction 19
I.10.4 . Fonction logique NON-ET 19
I.10.5 Fonction logique NON-OU 20
I.10.6 Fonction OU-EXCLUSIF (XOR) 20
I.10.7 Fonction NON-OU-EXCLUSIF (XNOR) 20
I.11 Fonctions logiques utilisant des interrupteurs 20
I.12 Représentation des fonctions logiques 21
I.12.1 Représentation par la table de vérité - logigrammes 21
I.13 Représentation par le tableau de Karnaugh – logigrammes 22
I.13.1 Représentation de deux variables a, b 22
I.13.2 Représentation de trois variables a,b,c 23
I.13.3 Représentation de quatre variables a,b,c,d 23
I.13.4 Adjacences des cases et regroupement de cases dans un tableau de Karnaugh 24
I.14 Formes canoniques 25

I. CHAPITRE I : INTRODUCTION A L'INFORMATIQUE

I.1 Objectifs du chapitre

Dans ce chapitre, vous allez voir :

- *L'environnement informatique*
- *Identifier les principaux composants d'un ordinateur*
- *Comprendre les systèmes de numération*
- *Convertir entre les bases*
- *Comprendre les bases de l'algèbre de Boole, y compris les opérations logiques fondamentales (ET, OU, NON).*
- *Savoir représenter des fonctions logiques sous forme de tables de vérité. Être capable de simplifier des expressions booléennes en utilisant les lois de l'algèbre de Boole.*
- *Apprendre à exprimer des fonctions logiques sous forme canonique (somme de produits, produit de sommes). Utiliser les tables de Karnaugh pour simplifier des expressions logiques.*
- *Savoir concevoir des circuits logiques en utilisant des portes logiques de base Exercices d'Applications*

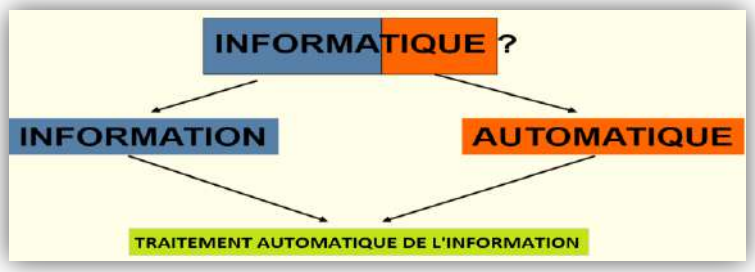
I.2 Définition de l'Informatique

L'informatique est une branche qui s'occupe du domaine du traitement automatique de l'information. Le mot ***« INFORMATIQUE »*** est composé à partir des mots ***« « INFORMATION »*** et ***« AUTOMATIQUE »***.

INFORMATIQUE = INFORmation + autoMATIQUE

L'informatique a pour rôle :

- *La conception et la construction des ordinateurs,*
- *Le fonctionnement et la maintenance des ordinateurs,*
- *Leur exploitation (utilisation des ordinateurs dans les différents domaines d'activités).*



I.3 Les ordinateurs

Par définition,

Un ordinateur est une machine qui permet de traiter l'information capable de :

- *Résoudre des problèmes ;*
- *Traiter des données en exécutant des opérations préétablies ;*
- *D'acquérir et de conserver de l'information (acquisition, stockage);*
- *D'effectuer des traitements (calcul) ;*
- *De restituer les informations stockées (restitution).*

Du point de vue d'un programmeur :

Un ordinateur peut être vu comme une machine composée, figure (1.2) :

- *D'un processeur capable d'effectuer des opérations de traitement ;*
- *D'une ou plusieurs mémoires chargées de retenir ;*
- *Manipulation des données (données d'entrée, de sortie, de travail) ;*
- *Création des programmes spécifiant les opérations à effectuer*

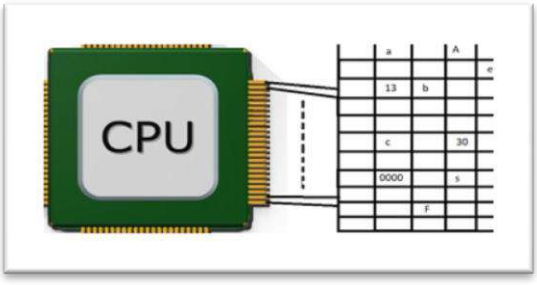


Figure I-2 :Du point de vue d'un programmeur

L'ordinateur a été créer principalement pour :

A. Calculer

C'est la tâche pour laquelle l'ordinateur a été conçu au départ. D'ailleurs, le nom anglais de l'ordinateur, ***« computer »***, signifie en français*« calculateur »*.La fonction de calcul d'un ordinateur ne se limite pas à l'utilisation de la calculette, d'un tableur ou à l'exécution de programmes de calculs scientifiques.

Il y a en fait du calcul dans toutes les opérations que réalise un ordinateur :

- *L'affichaged'une page web ou d'un document réalisé avec un traitement de texte,*
- *Le codage et le décodage des informations stockées dans les fichiers,*
- *La gestion des communications avec d'autres ordinateurs sur un réseau...*

B. Gérer des données

Lorsque vous utilisez un ordinateur, vous avez souvent besoin de conserver les résultats de votre travail. C'est par exemple le cas si vous rédigez votre CV, utilisez un outil de messagerie ou travaillez sur un logiciel de retouche d'image. Dans toutes ces situations, une fois votre travail terminé, vous souhaitez que l'ordinateur puisse enregistrer les données correspondantes, et vous les restituer ultérieurement.

Quels que soient les éléments de l'ordinateur où ce stockage aura lieu, il est nécessaire de mettre en forme ces données et de les organiser, pour que vous puissiez les retrouver au milieu de l'ensemble des autres données également stockées au même endroit.

C. Communiquer

Un ordinateur peut communiquer avec un utilisateur, avec un autre ordinateur, avec des machines…La fonction de communication ne consiste donc pas uniquement à échanger des informations sur Internet. Cette utilisation de l'ordinateur est d'ailleurs la plus récente, puisqu'elle ne s'est réellement développée qu'avec l'essor du Web, dans les années 1990.En revanche, quelle que soit la tâche que vous réalisez avec votre ordinateur, vous passez votre temps à interagir avec lui, soit pour lui donner des ordres, soit pour prendre connaissance des résultats. Dans ce cadre, un grand nombre d'éléments, matériels et logiciels, font partie de l'interface homme-machine, qui permet la communication entre l'utilisateur et l'ordinateur.

Les principales notions qui permettent de comprendre en quoi consiste une interface homme-machine sont définies dans la section ***« communication avec un ordinateur »***voire la figure (1.3).

En général un ordinateur fonctionne selon le schéma suivant :

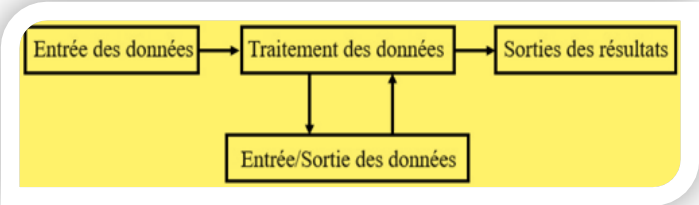


Figure I-3 : Principe de la magnétoscopie

I.4 Un aperçu sur l'architecture d'un ordinateur

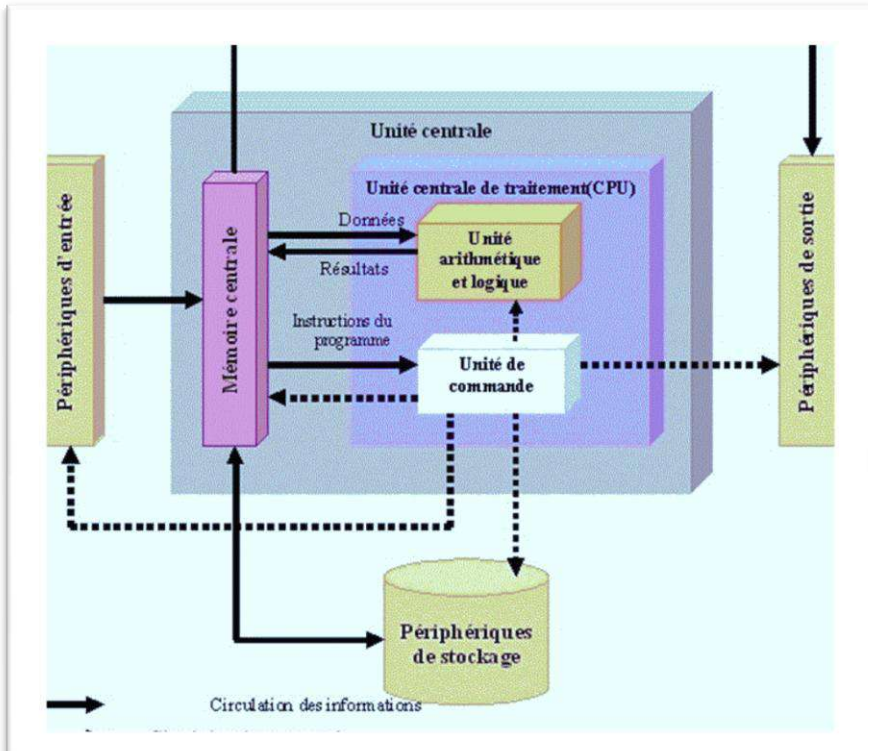
Un ordinateur est composé de deux parties :

- ***PARTIE MATERIELLE (HARDWARE)***
- ***PARTIE LOGICIELLE (SOFTWARE)***

Dans ce cours, nous nous focalisons sur la partie matérielle de l'ordinateur et expliquant son architecture et sa structure logique. Par la suite, nous présentons une structure physique simplifiée. L'ordinateur, d'une manière générale est constitué d'une ***CPU*** (***C***entral ***P***rocess ***U***nit : Unité de Traitement Centrale) et des périphériques.

La ***CPU*** est constituée d'un ***M***icro-***p***rocesseur (ou ***P***rocesseur) et d'une ***M***émoire ***C***entrale. Cette dernière également est constituée d'une mémoire vive (***RAM*** : **R**andom **A**ccess ***M***emory) et d'une mémoire morte (***ROM*** : ***R***ead ***O***nly ***M***emory).

Le processeur se compose de deux unités : ***unité de contrôle*** et ***unité arithmétique et logique***.



C.P.U: Central Process Unit (Unité de Traitement Centrale)

U.C: Unité de contrôle ou de commande

U.A.L: Unité Arithmétique et Logique

M.C: Mémoire Centrale

Figure I-4 : Principe de la magnétoscopie

Les périphériques sont composés de périphériques d'entrées, périphériques de sorties et ***périphériques de sauvegardes*** (*Mémoires externes*), Voir la figure I.4.

I.4.1 Micro-processeurs

a) L'***U.A.L***(***U**nité **A**rithmétique et **L**ogique : **C**ircuit de **C**alcul*), elle consiste en circuit de calcul qui est chargée d'effectuer toutes les opérations arithmétique et logiques. Elle contient tous les circuits logiques pour réaliser les différentes opérations arithmétique (Addition, soustraction, multiplication, division, etc.) et logiques (décalage, rotation, et logique, et binaire, ou logique, ou binaire, négation logique, complément, etc.).

b) L'***U.C*** (***U**nité de **C**ontrôle : **C**ircuit de **C**ommandes*) Elle contrôle toutes les opérations qui s'effectuent dans l'ordinateur. C'est elle qui permet l'exécution d'un programme présent en mémoire centrale (***M.C***).

c) La ***M.C*** (*La Mémoire Centrale : Circuit de Mémorisation*)

RAM

La « ***RAM*** » - ***R**andom **A**ccess **M**émory* (*Mémoire à Accès Aléatoire - Mémoire vive ou mémoire volatile*), a pour rôle de stocker les programmes à exécuter (en cours d'exécution) ainsi que les données résultats. Un programme ne peut s'exécuter que s'il est chargé en mémoire centrale. Elle est constituée de plusieurs barrettes (barrettes RAM). Cette mémoire est volatile, c'est à dire qu'elle s'efface s'il y a une coupure de courant.

ROM

La « ROM » - ***R**ead **O**nly **M**émory* (*Mémoire en lecture seule – Mémoire morte*) mémoire durable et ne s'efface pas par coupure de courant. Cette dernière sert à conserver du code et des paramètres système nécessaire au fonctionnement de l'ordinateur (***BIOS*** : ***B**asic **I**nput/**O**utput **S**ystem - Programme de base des entrées/sorties*).

La mémoire centrale se mesure actuellement par milliers de méga-octets.

On a :

- ***1 BIT : (ETAT ELECTRONIQUE 1 OU 0) = UNITE ELEMENTAIRE DE L'INFORMATION (BINARYELEMENT OU BINARY DIGIT)***
- ***1 OCTET = 1 CARACTÈRE = 8 BITS = 23 BITS.***
- ***1 KILO-OCTETS = 210 OCTETS = 1024 OCTETS***
- ***1 MEGA-OCTETS = 210 KILO-OCTETS = 220 OCTETS***
- ***1 GIGA-OCTETS = 210 MEGA-OCTETS = 220 KILO-OCTETS = 230 OCTETS***

I.4.2 Les Périphériques

a) Les mémoires externes ou auxiliaires (Support de stockage / mémorisation)

Elles servent à conserver des grandes quantités d'informations (fichiers de données, programmes, logiciels d'applications, système d'exploitation, etc.). Ceux sont des supports magnétiques, optiques ou électroniques qui ne s'effacent pas par coupures de courant. On peut en citer les disques magnétiques, disquettes, cassettes, CD-ROM, flash-disque, etc. Actuellement la capacité des disques magnétiques compte de dizaines de Giga-Octets pour les ordinateurs personnels (P.C.).

« 1 Giga-octets = 230 octets ≃ 1 milliard d'octets »

b) Les périphériques d'entrée (ou organes d'entrée)

Ils servent à transmettre les informations à l'ordinateur, plus précisément vers le CPU.

Exemple

On peut citer :

- *Clavier (il y a des claviers AZERTY ou QWERTY : introduire des textes et des commandes)*
- *Microphone (introduire des sons et de la parole)*
- *Caméra (introduire de la vidéo)*
- *Lecteur optique (introduire des données à partir de CD ou DVD)*
- *Scanner (numérisé des photos) – Tous de type de capteurs (Capteur de chaleurs, d'acidité, etc.)*

c) Les périphériques de sortie (ou organes de sortie)

Ils servent à recevoir les informations provenant du CPU∗ vers l'extérieur.

Ils permettent de traduire les informations de leurs formes codées (des 1 et 0) vers leurs formes naturelles.

Exemple

On peut citer :

- *L'écran ;*
- *L'imprimante ;*
- *Table traçante ;*
- *Synthétiseur vocal ;*
- *Synthétiseur musical ;*
- *Haut-parleur ; ...etc.*

d) Les périphériques d'entrée/sortie (ou organes d'entrée / sortie)

C'est l'ensemble de périphériques qui permet, à la fois, de transmettre et de recevoir des données. Ils sont, également, appelés périphériques de stockage ou mémoires externes (mémoires auxiliaires) : (La disquette, le CD-ROM, la clé USB, le disque dure externe, etc.…)

Remarque : Les mémoires externes ou auxiliaires (Disques durs, disquettes, flash-disque, CD*, DVD, etc.) sont considérées comme des périphériques d'entrée et de sortie en même temps.

Partie I : Système de Numération et codification des Nombres

I.5 Les systèmes de numération

Les nombres binaires jouent un rôle essentiel dans la conception des ordinateurs numériques. Par conséquent, une discussion détaillée du système de nombres binaires est donnée dans cette section. Le système de numérotation binaire utilise deux symboles 0 et 1 et sa base est 2. Les symboles 0 et 1 sont généralement appelés BITs, qui est une contraction des deux mots chiffres binaires.

Un nombre binaire de n bits de la forme : $a_{n-1}\, a_{n-2}\ldots..\, a_1\, a_0$ où chaque a_i (i = 0, 1, …. n–1) vaut « 0 » ou « 1 » a la grandeur.

$$a_{n-1}\, 2^{n-1} + a_{n-2}\, 2^{n-2} + ...+ a_1\, 2^1 + a_0\, 2^0$$

Le bit à l'extrême gauche d'un nombre binaire a la valeur de position la plus élevée et est généralement appelé bit le plus significatif (***le plus fort***) ou ***MSB***. De même, le bit occupant la position extrême droite d'un nombre binaire donné à la valeur de position la plus faible et est appelé bit le moins significatif (***le plus faible***) ou ***LSB***.

(The bit at the extreme le of a binary number has the highest positional value and is usually called the Most Significant Bit or MSB. Similarly, the bit occupying the extreme right position of a given binary number has the least positional value and is referred to as the Least Significant Bit or LSB.)

On appelle cela le « ***langage binaire*** » ou en « ***base 2*** » car il n'y a que 2 possibilités 0 ou 1 pour faire fonctionner les composants des ordinateurs, il n'y a que 2 états 1 et 0, ***vrai=1*** ou ***faux=0***.

Les ordinateurs sont composés de « ***plusieurs Milliards de transistors*** qui ont un fonctionnent similaire a un interrupteur commandé électriquement, Quand il **laisse passer le courant** on dit que cela **correspond au 1** alors que si le **courant ne passe pas**, cela **correspond au 0**. Il n'y a que deux états possibles, voir la figure (1.5). »

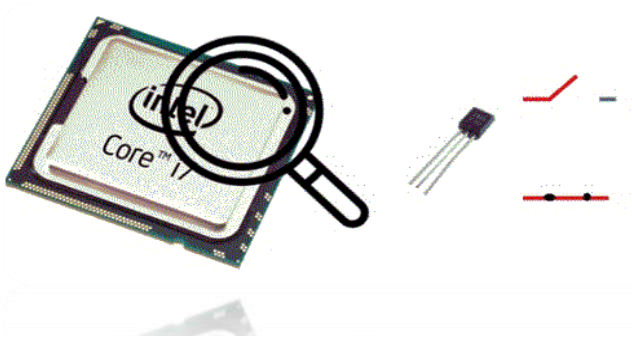


Fig1.5 Fonction d'un transistor

Remarque : **Autres systèmes de numération**

Systèmes	Romain	Décimal	Binaire	Octal	Hexadécimal
Zéro	/	0	0	0	0
Un	I	1	1	1	1
Deux	II	2	10	2	2
Trois	III	3	11	3	3
Quatre	IV	4	100	4	4
Cinq	V	5	101	5	5
Six	VI	6	110	6	6
Sept	VII	7	111	7	7
Huit	VIII	8	1000	10	8
Neuf	IX	9	1001	11	9
Dix	X	10	1010	12	A
Onze	XI	11	1011	13	B
Douze	XII	12	1100	14	C
Treize	XIII	13	1101	15	D
Quatorze	XIV	14	1110	16	E
Quinze	XV	15	1111	17	F
Seize	XVI	16	10000	20	10

I.6 Conversion d'un nombre d'un système à un autre

Les conversions de nombre interviennent pour passer d'un système de numération vers un autre :

I.6.1 Conversion binaire en décimal

Le nombre donné est exprimé comme une somme de termes dont chacun est le produit d'un bit (***0 ou 1***) et d'une puissance de ***2***.

- La puissance de **2** est déterminée à partir de la position du bit. Ainsi l'équivalent décimal d'un nombre binaire « ***a*** » la forme générale ;

$$a_{n-1}\ a_{n-2}\ ....\ a_i.....\ a_1.....\ a_0\ ,\ a_{-1}\ a_{-2}\ .....\ a_{-p}$$
$$= a_{n-1}*2^{n-1} + a_{n-2}*2^{n-2} + ... + a_i*2^i + ... + a_1*2^1 + a_0*2^0 + a_{-1}*2^{-1} + a_{-2}*2^{-2} + ... + a_{-p}*2^{-p}$$

Dans le système de nombres binaires, quelques exemples de nombres binaires et de leurs équivalents décimaux sont donnés ci-dessous :

$(101101)_2 = (1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 32+0+8+4+0+1)10 = (45)_{10}$

Les résultats ci-dessus peuvent s'exprimer plus clairement de la manière suivante :

Nombre Binaire	1	0	1	1	0	1	Nombre Décimal
Base/ Puissance	2^5	2^4	2^3	2^2	2^1	2^0	45
Équivalent en Décimal	32	16	8	4	2	1	
Produit pour chaque termes	32	0	8	4	0	1	

Autre exemple avec virgule

$(111,1011)_2 = (1\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$

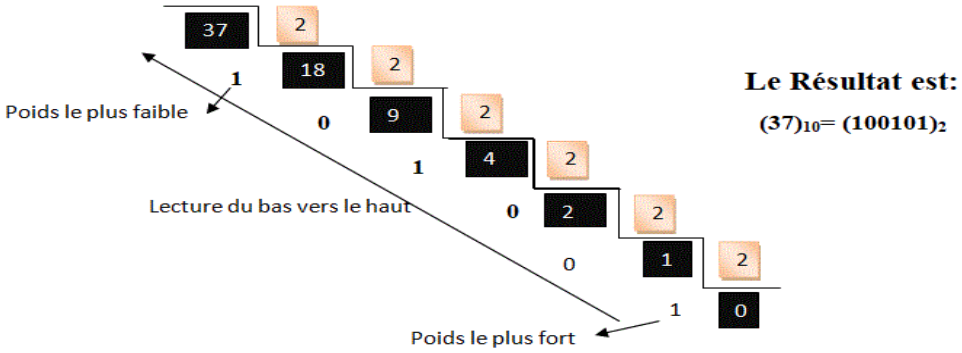
$= 4 + 2 + 1 + 0{,}5 + 0 + 0{,}125 + 0{,}0625)10 = (7{,}6875)_{10}$

I.6.2 Conversion décimal - binaire d'un nombre entier

a) Méthode de multiplication et de division :

La méthode des divisions successives consiste à diviser successivement par 2, le nombre décimal à convertir jusqu'à ce que le résultat de la division soit égal à 0. Le nombre binaire correspondant sera la succession des restes obtenus. Le bit de poids le plus élevé de ce nombre, étant le reste obtenu à la dernière division.

Exemple



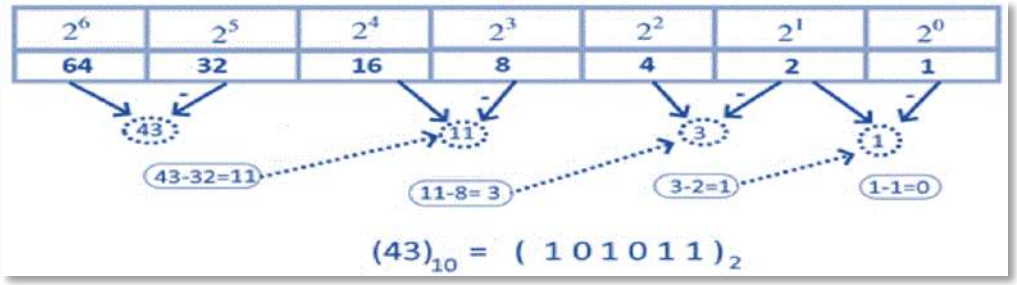
b) Méthode des soustractions successives

Pour utiliser cette méthode∗, on doit déterminer en premier lieu, les valeurs successives des puissances de la base (2_i, i=0.....n). On déterminera ensuite, entre quelles valeurs successives de 2i se situe le nombre à convertir. La borne inférieure est alors soustraite au nombre. On procède de la même manière avec le reste obtenu, jusqu'à ce que l'on obtienne 0, comme reste de la soustraction.

La valeur binaire sera 1 à la position du poids utilisé dans la soustraction, et 0 à la position des poids non utilisés.

Exemple

En utilisant la méthode des soustractions successives, convertir le nombre décimal ***43*** en binaire. On situe ***43*** dans la table des puissances successives de ***2***; ensuite on lui soustrait la borne inférieure. On fait de même avec le reste trouvé et ainsi de suite jusqu'à trouver **0**



On a utilisé dans les différentes soustractions 2^5 , 2^3 , 2^1 et 2^0 . Pour récupérer le résultat on va donc mettre des 1, dans les positions correspondant aux poids 5, 3, 1 et 0, et des 0 dans les positions qu'on n'a pas utilisées (c'est-à-dire les poids 2 et 4). Donc la conversion de $(43)_{10}$ de la base 10 est : $(101011)_2$ en base 2.

I.6.3 Conversion de la base B (2, 8, 16) vers la base 10

Pour convertir un nombre $N_b = (a_{n-1}\, a_{n-2} \ldots a_1 a_0\,)_b$ de la base b vers la base 10, on effectue le calcul suivant :

$$(N_b)_b = (a_{n-1} * b^{n-1} + a_{n-2} * b^{n-2} \ldots + a_1 * b^1 + a_0 * b^0\,)_{10}$$

$$(Nb)_B = \sum_{i=0}^{n-1} a_i * b^i$$

Elle correspond à la somme des produits de chaque symbole du nombre par le poids correspondant. Le poids des symboles : il s'obtient en numérotant les symboles à partir de la droite vers la gauche, en commençant du 0.

Quelques exemples d'applications :

EXEMPLE 1 : $(1367)_8 = (?\,)_{10}$

$(1367)_8 = (1 * 8^3 + 3 * 8^2 + 6 * 8^1 + 7 * 8^0\,)_{10}$

$(1367)_8 = (759)_{10}$

EXEMPLE 2 : $(2B3\,)_{16} = (?\,)_{10}$

$(2B3\,)_{16} = (2 * 16^2 + B * 16^1 + 3 * 16^0\,)_{10}$

$(2B3\ )_{16} = (691)_{10}$

EXEMPLE 3 : $(1011101\ )_2 = (?\ )_{10}$

$(1011101)_2 = (1 * 2^6 + 0 * 2^5 + 1 * 2^4 + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0\ )_{10}$

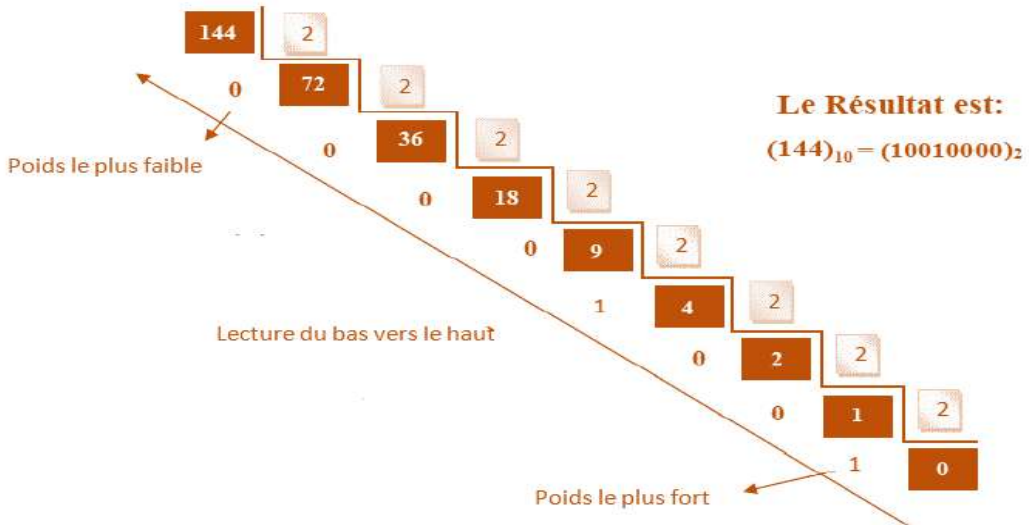
$(1011101)_2 = (93)_{10}$

I.6.4 Conversion décimal vers une base B quelconque(B10 vers B(2,8,16))

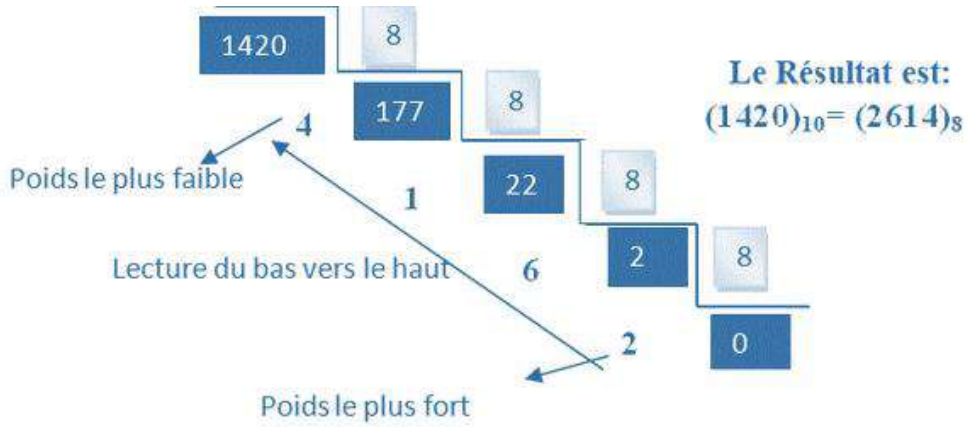
Pour la conversion décimale à un autre système de numération, le procédé est identique (divisions successives ou soustractions successives).

On remplace seulement 2 par B ; B étant la base de numération du système considéré, les restes obtenus auront des valeurs comprises entre 0 et B-1

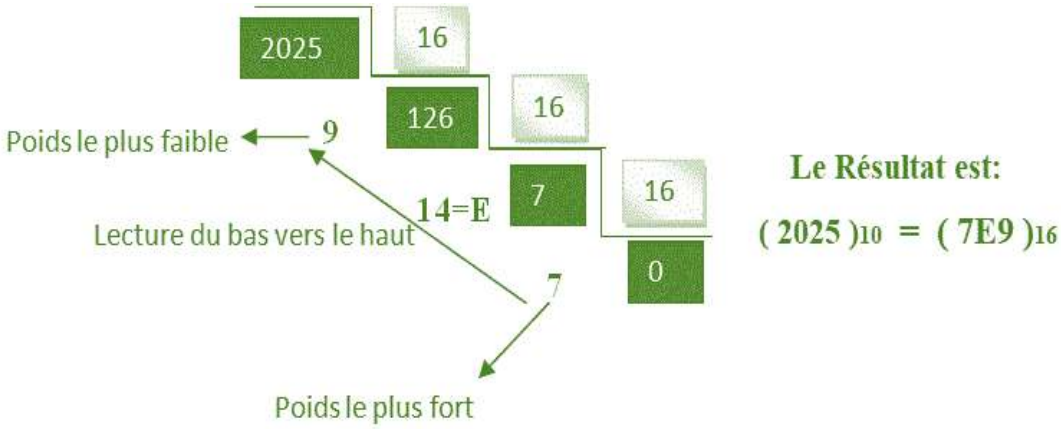
EXEMPLE 1 : $(144)_{10}= (?)_2$



EXEMPLE 2 : $(1420)_{10}= (?)_8$



EXEMPLE 2 : $(2025)_{10}= (?)_{16}$



I.6.5 Conversion de la base 8 vers la base 2

Pour convertir un nombre N_b exprimé en base 8 vers la base 2, nous procédons comme suit : $8 = 2^3$ Il faut donc utiliser 3 bits pour exprimer un seul chiffre octal en binaire. La représentation des chiffres de la base 8 vers le binaire est comme suit :

Chiffre en Octal	Chiffre équivalent en Binaire		
	2^2	2^1	2^0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

$(4)_8=(1.2^2+0.2^1+0.2^0)_2=(100)_2$

$(7)_8=(1.2^2+1.2^1+1.2^0)_2=(111)_2$

$(2)_8=(0.2^2+1.2^1+0.2^0)_2=(010)_2$

Exemples tirés à partir de la table :

$(743)_8 = (111\ 100\ 011)_2$
$(134)_8 = (001\ 011\ 100)_2$
$(3476)_8 = (011\ 100\ 111\ 110)_2$

I.6.6 Conversion de la base 16 vers la base 2

Pour convertir un nombre N_b exprimé en base 16 vers la base 2, nous procédons comme suit : $16 = 2^4$ Il faut donc utiliser 4 bits pour exprimer un seul chiffre hexadécimal en binaire. La représentation des chiffres de la base 16 vers le binaire est comme suit :

Chiffre en Héxadécimal	Chiffre équivalent en Binaire				
	2^3	2^2	2^1	2^0	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	$(9)_{16}=(1.2^3 + 0.2^2 +0.2^1 +1.2^0)_2=(1001)_2$
4	0	1	0	0	
5	0	1	0	1	$(A)_{16}=(1.2^3 + 0.2^2 +1.2^1 +0.2^0)_2=(1010)_2$
6	0	1	1	0	
7	0	1	1	1	$(F)_{16}=(1.2^3 + 1.2^2 +1.2^1 +1.2^0)_2=(1111)_2$
8	1	0	0	0	
9	1	0	0	1	
10 | A	1	0	1	0	
11 | B	1	0	1	1	
12 | C	1	1	0	0	
13 | D	1	1	0	1	
14 | E	1	1	1	0	
15 | F	1	1	1	1	

Exemples tirés à partir de la table :

$(743\ )_{16} = (0111\ 0100\ 0011)_2$

$(134\ )_{16} = (0001\ 0011\ 0100)_2$

$(A6E9\ )_{16} = (1010\ 0110\ 1110\ 1001)_2$

I.6.7 Conversion de la base 2 vers la base 8

Pour trouver l'équivalent d'un « nombre binaire en octal », il suffit de former des groupes de 3 bits chacun (Puisque $8 = 2^3$), en commençant du poids le plus faible (à partir de la droite), si le dernier groupe formé possède moins de 3 bits, il suffit de rajouter des 0, puis calculer l'équivalent en octal de chaque groupe.

EXEMPLE DE CONVERSION DU BINAIRE VERS L'OCTAL

- $(10110101010\ )_2 = (\ ?\ )_8$

2 6 5 2

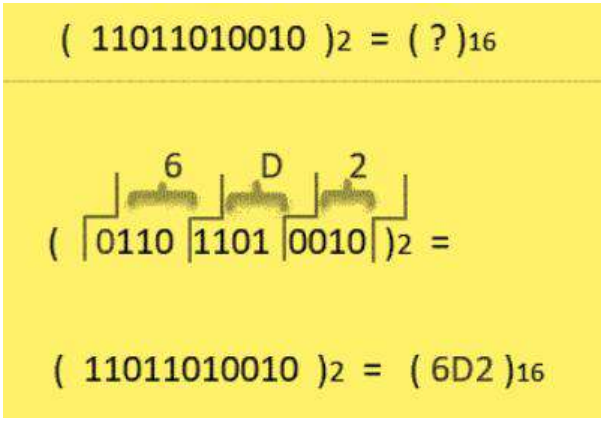
$(\ 010\ 110\ 101\ 010\ )_2 = (\ 2652\ )_8$

$(10110101010\ )_2 = (\ 2652\ )_8$

I.6.8 Conversion de la base 2 vers la base 16

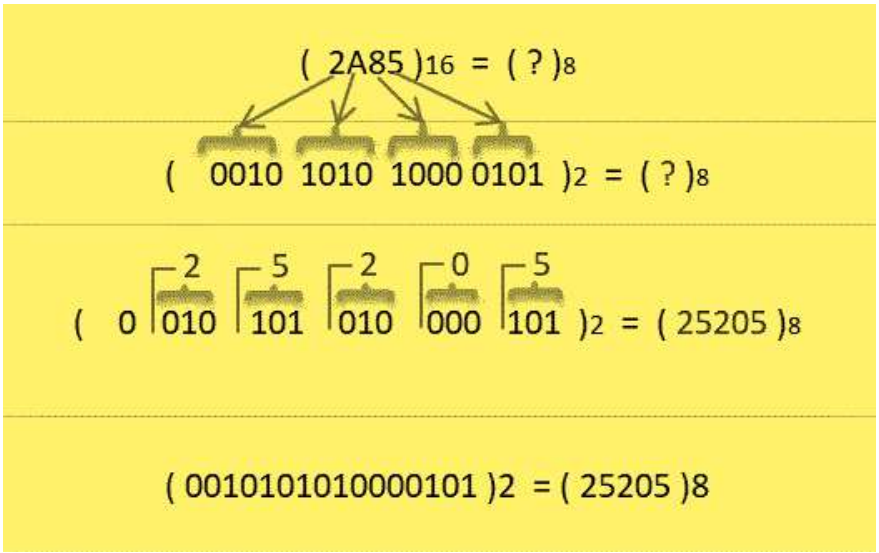
Pour trouver l'équivalent d'un « nombre binaire en Hexadécimal », il suffit de former des groupes de 4 bits chacun (Puisque $16 = 2^4$), en commençant du poids le plus faible (à partir de la droite), si le dernier groupe formé possède moins de 4 bits, il suffit de rajouter des 0, puis calculer l'équivalent en Hexadécimal de chaque groupe.

EXEMPLE DE CONVERSION DU BINAIRE VERS L'HEXADECIMAL



I.6.9 Conversion de la base 16 vers la base 8

Pour convertir un nombre N_b exprimé en base 16 vers la base 8 ou vice versa, nous devons passer par une base intermédiaire tel que le décimal ou le binaire, mais le passage par le binaire est « beaucoup plus simple ».



I.7 Opérations arithmétiques de base

Dans ce qui suit, nous allons voir comment les opérations arithmétiques de base sont réalisé sur les systèmes de numération base 2, 8 et 16.

I.7.1 Opérations arithmétique en binaire (Base 2)

a) Addition binaire

L'addition en base 2 fonctionne comme l'addition en décimal, mais attention car en binaire, 1+ 1 = 10 on sait que $(2)_{10} = (10)_2$: il faut donc placer 0 et mettre une retenue de 1 sur le bit suivant.

Les règles de l'addition

0 + 0 = 0	***0 + 1 = 1***
1 + 0=1	***1 + 1 = 10 {On écrit "0" et on reporte "1" sur le bit de rang supérieur (rang de gauche) }***

b) Soustraction binaire

Dans la soustraction binaire, on procède comme en décimal. Quand la quantité à soustraire est supérieure à la quantité dont on soustrait, on emprunte 1 au voisin de gauche. En binaire, ce 1 ajoute 2 à la quantité dont on soustrait, tandis qu'en décimal il ajoute 10.

Les règles de la soustraction

0 - 0 = 0	***0 - 1 = {on emprunte "1" ce qui fait 10-1, on écrit "1" et on retient 1 }***
1 - 0 = 1	***1 - 1 = 0***

c) Multiplication binaire

La multiplication binaire s'effectue selon le principe des multiplications décimal, on multiplie donc le multiplicande par chacun des bits du multiplicateur. On décale les résultats intermédiaires obtenus et on effectue ensuite l'addition de ses résultats partiels.

Les règles de la multiplication

0 x 1 = 0	***0 x 0 = 0***
1 x 0 = 0	***1 x 1 = 1***

d) Division binaire

La division binaire s'effectue à l'aide de soustractions et de décalages, comme la division décimale, sauf que les digits du quotient ne peuvent être que 1 ou 0. Le bit du quotient est 1 si on peut soustraire le diviseur, sinon il est 0.

Les règles de la division

0 : 0 =/	***1 : 0 =/***
0 : 1 = 0	***1 : 1 = 1***

L'Algèbre de BOOLE

I.8 Définition

Algèbre de Boole permet de simplifier des systèmes binaires qui eux-mêmes servent à concevoir et implanter des circuits logiques, s'intéresse aux opérations et aux fonctions sur les variables logiques, dans la conception des circuits électronique. Permet de modéliser des raisonnements logiques, en exprimant un « état » en fonction de conditions. Par exemple : Communication = Émetteur ET Récepteur Communication est « VRAI » si Émetteur actif ET Récepteur actif (c'est une fonction logique dépendant des variables Émetteur et Récepteur).

Liste des notations booléennes communément utilisées et de leurs énoncés équivalents :

Symboles	Enoncés équivalent
+	OR / OU
.	AND / ET
$\bar{A}$	NOT A ou bien NON A
S=A	S est vrai si A est Vrai
S=A+B	S est vrai si A OU B est Vrai
S=A . B	S est vrai si A ET B sont Vrai

I.9 Propriétés de l'algèbre de Boole

Si A et B sont des variables Booléennes, alors A+B, A.B sont aussi des variables Booléennes.

1) Associativité

Comme avec les opérations habituelles, certaines parenthèses sont inutiles :

(a + b) + c = a + (b + c) = a + b + c

(a.b).c = a.(b.c) = a.b.c

2) Commutativité

L'ordre est sans importance :

a + b = b + a

a.b = b.a

3) Distributivité

Comme avec les opérations habituelles, il est possible de distribuer :

a.(b + c) = a.b + a.c

Attention : comportement différent par rapport aux opérateurs + et * habituels :

a + (b.c) = (a + b).(a + c)

4) Idempotence

a + a + a + [...] + a = a

a.a.a.[...].a = a

5) Élément neutre

a + 0 = a

a.1 = a

6) Élément nul

0.a = 0

1 + a = 1

7) Absorption

a + a.b = a

a.(a + b) = a

8) Simplification

A+$\bar{A}$. b = A + B

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

9) Redondance

A . B + $\bar{A}$. C = A . b + $\bar{A}$. C + B . C

10) Complémentarité

a = a (« La lampe est allumée » =« la lampe n'est pas non allumée »)

a + a = 1 (« VRAI » Si « lampe allumée » OU Si « lampe _non allumée » → c'est toujours le cas → vrai dans tous les cas → toujours VRAI, donc =1)

a .a = 0 (« VRAI » Si « lampe allumée » ET Si « lampe _non allumée » → impossible → faux dans tous les cas → toujours FAUX donc =0)

11) Priorité

Pour faciliter leur compréhension, il a été décidé que ces opérations seraient soumises aux mêmes règles que les opérations, la fonction ET est ainsi prioritaire par rapport à la fonction OU; pour s'aider, on peut placer des parenthèses dans les opérations.

Lois ou théorème de De Morgan

Le théorème de De MORGAN s'exprime par les deux relations :

- $\overline{A+B}=\overline{A}.\overline{B}$
- $\overline{A.B}=\overline{A}+\overline{B}$

Exemple sur la priorité :

Dans cet exemple on suppose **a = 0** ; **b = 1** ; **c = 1** et On cherche à déterminer **a .b + c = ?**
Ce qu'il faut faire d'abord

- On calcule « **a .b** » : car la multiplication logique ET est prioritaire devant la somme logique OU.
- Donc : **a .b = 0 . 1**d'où le résultat de **0.1 = 0**
- Puis, on calcule **0 + c** qui est **0 + 1 = 1**
- Le résultat final de **a .b + c = (a . b) + c = 1**

I.10 Portes Logiques

Les circuits de commutation constituent les éléments de base des systèmes numérique et sont appeler des « ***portes*** ». Ce nom vient de leur aptitude à commander une sortie par une combinaison d'états logique à l'entrée de la porte et des fonctions de commutation exécutées décrite par des tables de vérités.

Les fonctions logiques peuvent être représentées de plusieurs façons :

- ***Équations logiques***
- ***Tables de vérités***
- ***Logigrammes***
- ***Diagrammes échelle (LADDER qui est un langage en automatisme)***

I.10.1 La Négation

Le contraire de « **a** » est VRAI si et seulement si **a** est FAUX.

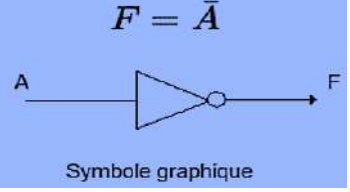
Le contraire de « **a** » est noté par :

- ***NON-a***
- « **!** » dans quelques langages de programmation (C, C++, ...)
- « **NOT** » dans certains langages de programmation (ASM, ...)

Représentation de la fonction NON :

Table de vérité

Entrée	Sortie
A	F
0	1
1	0



I.10.2 . La Conjonction

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 par :

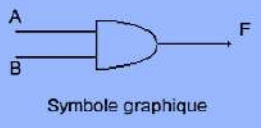
- « **.** » « **&**» ou « **&&** » dans quelques langages de programmation (Perl, C, PHP...)
- « **AND** » dans certains langages de programmation (Ada, Pascal, Python, PHP ...)

Représentation de la fonction AND ou ET :

F=A.B ou F= A AND B

Table de vérité

Entrée		Sortie
B	A	F
0	0	0
0	1	0
1	0	0
1	1	1



I.10.3 La Disjonction

Elle est définie de la manière suivante : « **a OU b** » est VRAI si et seulement si a est VRAI ou b est VRAI. (si a est vrai et que b est vrai aussi, alors « **a OU b** » est vrai.)

Cette loi est aussi notée par :

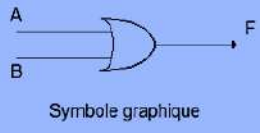
- « + » « | » ou« || » dans quelques langages de programmation
- « **OR** » dans certains langages de programmation

Représentation de la fonction OU ou OR :

F=A+B ou F= A OR B

Table de vérité

Entrée		Sortie
B	A	F
0	0	0
0	1	1
1	0	1
1	1	1



I.10.4 . Fonction logique NON-ET

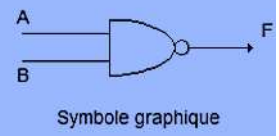
Elle est définie de la manière suivante : « **a NON-ET b** » est Vrai si et seulement si **a** est FAUX **OU b** est FAUX.

Représentation de la fonction NON-ET :

Table de vérité

Entrée		Sortie
B	A	F
0	0	1
0	1	1
1	0	1
1	1	0

$$F = \overline{A.B}$$



I.10.5 **Fonction logique NON-OU**

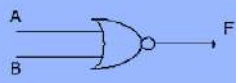
Elle est définie de la manière suivante : « **a NON-OU b** » est VRAI si et seulement si **a** est FAUX **ET b** est FAUX.

Représentation de la fonction NON-OU :

Table de vérité

Entrée		Sortie
B	A	F
0	0	1
0	1	0
1	0	0
1	1	0

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



Symbole graphique

I.10.6 **Fonction OU-EXCLUSIF (XOR)**

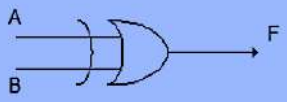
Représentation de la fonction XOR :

Représentation de la fonction XOR :

Table de vérité

Entrée		Sortie
B	A	F
0	0	0
0	1	1
1	0	1
1	1	0

$$F = A \oplus B = \overline{A}B + A\overline{B}$$



Symbole graphique

I.10.7 **Fonction NON-OU-EXCLUSIF (XNOR)**

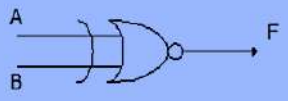
Représentation de la fonction XOR :

Représentation de la fonction XOR :

Table de vérité

Entrée		Sortie
B	A	F
0	0	1
0	1	0
1	0	0
1	1	1

$$F = \overline{A \oplus B} = \overline{A}\overline{B} + AB$$



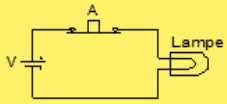
Symbole graphique

I.11 Fonctions logiques utilisant des interrupteurs

- En électronique, on représente les fonctions logiques avec des logigrammes.
- En automatisation, on utilise des interrupteurs et des relais pour représenter les fonctions logiques.

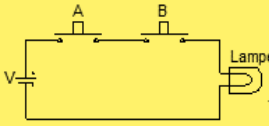
Exemples sur les fonctions logiques utilisant des interrupteurs

❖ **Fonction logique NON**
Exemple : Interrupteur normalement fermé



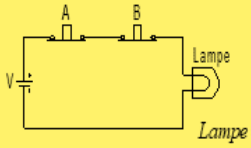
$Lampe = \overline{A}$

❖ **Fonction logique ET**
Utilise deux interrupteurs normalement ouvert en séries.



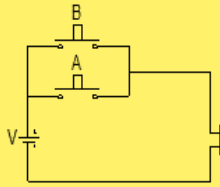
$Lampe = A \bullet B$

❖ **Fonction logique NON-OU**
Utilise deux interrupteurs normalement fermés en séries.



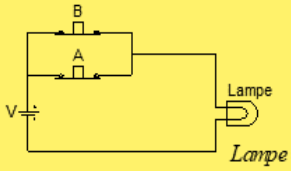
$Lampe = \overline{A+B} = \overline{A}\,\overline{B}$

❖ **Fonction logique OU**
Utilise deux interrupteurs normalement ouvert en parallèles.



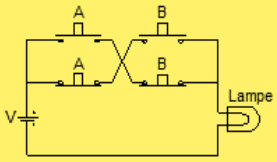
$Lampe = A + B$

❖ **Fonction logique NON-ET**
Utilise deux interrupteurs normalement fermés en parallèles.



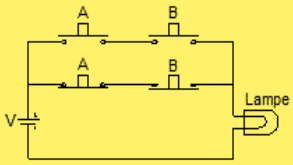
$Lampe = \overline{AB} = \overline{A} + \overline{B}$

❖ **Fonction OU-EXCLUSIF**
Utilise deux interrupteurs à deux contacts



$Lampe = A \oplus B = \overline{A}B + A\overline{B}$

❖ **Fonction NON OU-EXCLUSIF**
Utilise deux interrupteurs à deux contacts



$Lampe = \overline{A \oplus B} = \overline{A}\,\overline{B} + AB$

I.12 Représentation des fonctions logiques

I.12.1 **Représentation par la table de vérité - logigrammes**

Pour une table de vérité donnée, nous pouvons trouver l'équation logique et le logigramme (ou diagramme échelle) correspondant

- Il faut utiliser l'algèbre de Boole pour simplifier.

Exercice d'application

Questions :

1. A partir de la table de vérité ci-dessous exprimez la sortie S en fonction des entrées A,B,C
2. Tracer son logigramme en utilisant des portes logiques nécessaires.

Entrées			Sortie
C	B	A	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Solution :

1. On construit l'équation de S en écrivant tous les termes donnant S=1.

   Ainsi, S = 1 :

   - si C=0 et B=1 et A=0;
   - ou si C=0 et B=1 et A=1;
   - ou si C=1 et B=0 et A=1;
   - ou si C=1 et B=1 et A=0. 2.

On peut donc écrire :

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

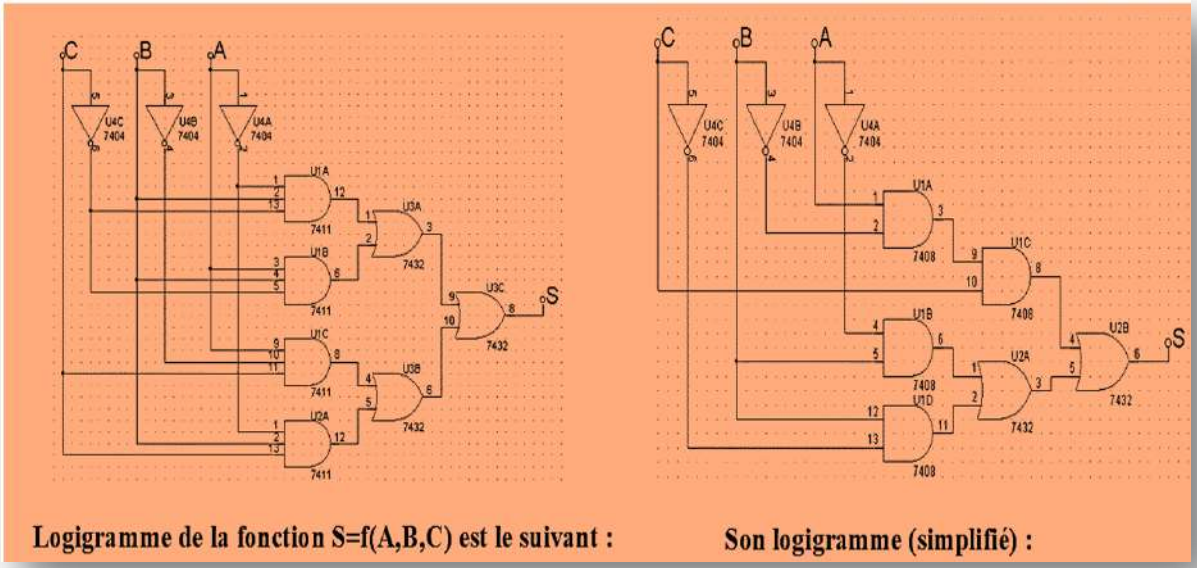
2. Tracer son logigramme en utilisant des portes logiques nécessaires.

   On peut simplifier la fonction S:

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

   Autre simplification :

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

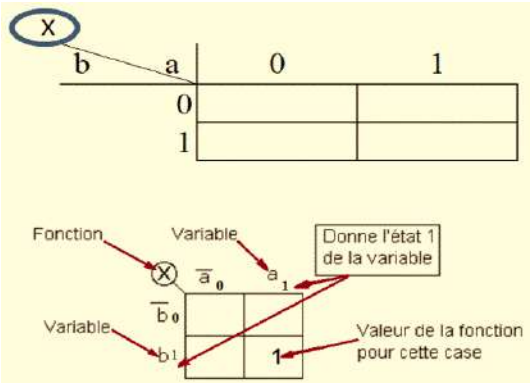


Logigramme de la fonction S=f(A,B,C) est le suivant : **Son logigramme (simplifié) :**

I.13 Représentation par le tableau de Karnaugh – logigrammes

I.13.1 **Représentation de deux variables a, b**

Un tableau de Karnaugh peut se représenter sous les formes suivantes :

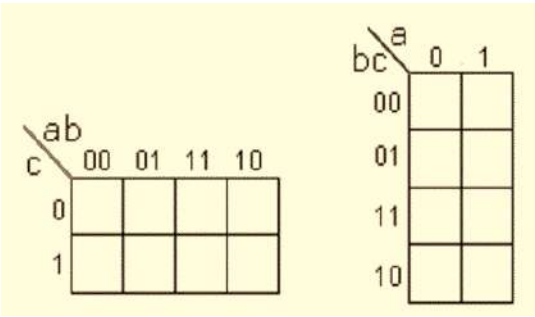


Avec X est le nom de la fonction
A et B sont des noms des variables (a, b),

L'état des variables : 0 , 1

La valeur de la fonction (1 ou 0) à l'intérieur des cases

I.13.2 **Représentation de trois variables a,b,c**



A chaque case est associé un triplet des valeurs a, b, c.

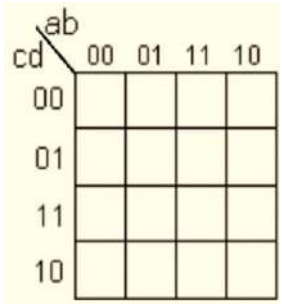
Exemple : La case 1 représentera le triplet {0,0,0} ou a = 0, b = 0 et c = 0.
Nous pouvons dire également que la case 1 correspond au produit (a · b · c).

I.13.3 **Représentation de quatre variables a,b,c,d**

A chaque case est associé un quadruplet des valeurs a, b, c, d.

Exemples :

- la case 4 représentera le quadruplet {1,0,0,0} ou a = 1, b = 0, c = 0 et d = 0 (a · b · c · d).
- La case 11 représentera le quadruplet {1,1,1,1} ou a = 1, b = 1, c = 1 et d = 1 (a · b · c · d).
- La case 16 représentera le quadruplet {1,0,1,0} ou a = 1, b = 0, c = 1 et d = 0 (a · b · c · d).



Exercice d'application (écriture dans le tableau de KARNAUGH)

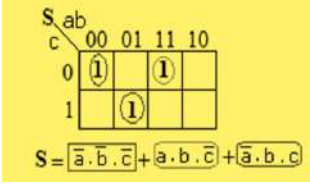
Dans cet exercice nous allons voir comment obtenir une équation de sortie simplifiée, à partir d'une table de vérité en utilisant la technique utilisée par Karnaugh.

Supposons que l'étude d'un dispositif nous ait conduits à la table de vérité ci-contre ;

La sortie S de notre dispositif doit fonctionner dans les cas suivants :

- Les 3 variables a, b et c sont simultanément à l'état 0 (fonction ET : a·b·c),
- OU si a=0, b=1, c=1 simultanément (fonction ET : a·b·c),
- OU si a=1, b=1, c=0 simultanément (fonction ET : a·b· c).

Entrées			Sortie
a	b	c	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



Ce que nous traduisons par l'équation :

$$S = \bar{a}\bar{b}\bar{c} + \bar{a}bc + ab\bar{c}$$

Dans le tableau de Karnaugh, nous mettrons un «1» dans chacune des cases correspondant aux termes

- $\bar{a} \cdot \bar{b} \cdot \bar{c}$
- $\bar{a} \cdot b \cdot c$
- $a \cdot b \cdot \bar{c}$

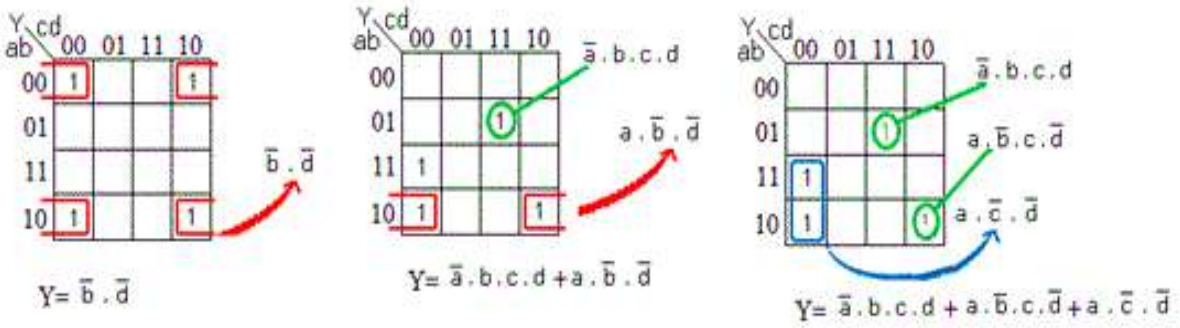
Nous placerons un «**0**» dans les cases correspondant aux autres termes. A l'aide du regroupement des case adjacente on peut écrire :

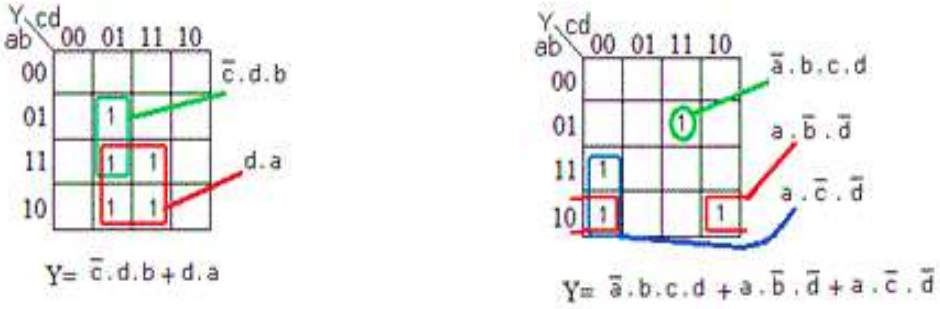
$$S = \bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot b \cdot c + a \cdot b \cdot \bar{c}$$

I.13.4 Adjacences des cases et regroupement de cases dans un tableau de Karnaugh

Dans chaque cas, l'ordre d'écriture des états des variables fait qu'entre deux cases voisines (en ligne ou en colonne) une seule variable change d'état ; on dit de telles cases qu'elles sont adjacentes voir la figure ci-dessous.

Exemples d'ordre d'écriture des états des variables fait qu'entre deux cases voisines (en ligne ou en colonne)





I.14 Formes canoniques

À partir de la table de vérité, nous pouvons avoir deux formes analytiques, dénommées formes canoniques :

1. Somme canonique de produits (1ère forme Canonique)

2. Produit canonique de sommes (2ème forme Canonique)

Exercice d'application

On prend le même exemple précédent en déduits les deux formes :

a) Pour écrire l'expression de S sous la 1ère forme Canonique

On localise les 1 dans la sortie S en fonction des états des entrées Si une entrée est 1 on prend la variable telle quelle et s'il est à 0 on prend l'inverse de cette variable.

On peut écrire :

$$S = \bar{C}.B.\bar{A} + \bar{C}.B.A + C.\bar{B}.A + C.B.\bar{A}$$ (Somme de Produit)

Entrées			Sortie
C	B	A	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

b) Pour écrire l'expression de S sous la 2ère forme Canonique

On localise les 0 dans la sortie S en fonction des états des entrées Si une entrée est 0 on prend la variable telle quelle et s'il est à 1 on prend l'inverse de cette variable

On peut écrire :

$$S = (C + B + A).(C + B + \bar{A}).(\bar{C} + B + A).(\bar{C} + \bar{B} + \bar{A})$$ (Produit de Somme)

Glossaire

BIT

Abréviation de binary digit (chiffre binaire). Plus petite unité d'information traitée par un ordinateur. Un bit représente 1 ou 0 en numération binaire, ou une condition logique vraie ou fausse. Il est représenté physiquement par un élément tel qu'une tension faible ou élevée en un point d'un circuit ou un point magnétique positif ou négatif sur un disque. Un seul bit véhicule peu d'informations susceptibles d'intéresser un utilisateur. Par contre, un groupe de 8 bits constitue un octet, une unité pouvant être utilisée pour représenter de nombreux types d'informations, tels qu'une lettre de l'alphabet, un nombre décimal ou tout autre caractère. Anglais : bit.

CD

CD réinscriptible contenant 650Mo d'espace pouvant être réécrit en théorie 1 000 fois. Anglais : Compact Disc ReWritable

CPU

CPU est un acronyme anglais pour Central ProccessingUnit.Le CPU désigne un processeur ou microprocesseur principal d'un ordinateur, on parle aussi d'unité de traitement. Le CPU est l'élément central d'une configuration informatique dont la puissance et la vitesse de traitement influencent directement les performances.

Abréviations

BIT :BInarydigiT (0 or 1)
CPU : Central Process Unit (Unité de Traitement Centrale)
INFORMATIQUE := INFORmation + autoMATIQUE
LSB : Least Significant Bit (le bit le plus faible)
MC :Mémoire Centrale
MSB : Most Significant Bit (le bit plus fort)
RAM : Random Access Memory (MémoireVive)
ROM : Read Only Memory (MémoireMorte)
UAL : Unité Arithmétique et Logique
U C : Unité de contrôle ou de commande

Références

https://fac.umc.edu.dz/ista/pdf/cours/
https://elearning.univ-bejaia.dz/course/view.php?id=14316

Bibliographie

- *ZANELLA, P. and Ligier, Y. (1989). Architecture et technologie des ordinateurs. DUNOD informatique.*
- *DUNOD. BAJARD, J. (2004). Calcul et arithmétique des ordinateurs. Traité IC2 Information - Commande - Communication : Informatique et systèmes d'information. Hermes Science Publications.*
- *BELAID, M. (2004). Architecture des ordinateurs : cours et exercices corrigés. Les Manuels de l'étudiant. Les Pages Bleues Internationales.*
- *TOCCI, R. (1992). Circuits numériques : théorie et applications.*
- *DUNOD. WACK, B. (2013). Informatique pour tous en classes préparatoires aux grandes écoles. Eyrolles.*