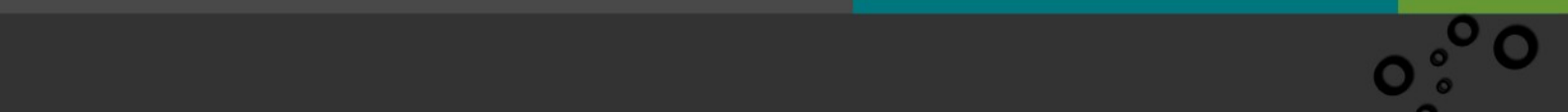


# **Généralités, les systèmes de numération et l'Arithmétique binaire**





# Table des matières

<b>Objectifs</b>	<b>5</b>
<b>Introduction</b>	<b>7</b>
<b>I - Partie I : Généralités sur le numérique</b>	<b>9</b>
A. 1. Historique.....	10
B. 2. Comparaison des signaux analogiques et les signaux numériques.....	10
C. 3. Les avantages du numérique.....	11
<b>II - Partie II : Les bases de numération et la conversion d'une base à l'autre</b>	<b>13</b>
A. 1. Les systèmes de numérations.....	14
B. 2. Conversion d'une base à une autre.....	15
C. RÉSUMÉ.....	17
D. Exercice.....	17
<b>III - Partie III : Codage des nombres</b>	<b>19</b>
A. 1. Code pondéré.....	19
B. 2. Code non pondéré.....	20
C. 3. Codes correcteur d'erreur.....	22
D. Exercice.....	22
E. Exercice.....	22
<b>IV - Partie IV : Représentation des nombres et arithmétique binaire</b>	<b>23</b>

A. 1. Représentation des nombres négatifs.....	24
B. 2. Arithmétique binaire.....	25
C. Exercice.....	26
D. Exercice.....	26
<b>V - Test de compréhension de la leçon 1</b>	<b>27</b>
<b>Solution des exercices</b>	<b>29</b>





# Objectifs

À la fin de cette leçon, vous serez capable de

- Définir les avantages et inconvénient du signal numérique
- Définir les différents systèmes de numération ;
- Faire les calculs de conversion entre les différents systèmes de numération ;
- Manipuler les différentes représentations des éléments de numérations.



# Introduction

---

Bienvenue à la première semaine du cours d'électronique numérique, dont la première leçon porte sur **les systèmes de numération et l'arithmétique en base 2**. Cette leçon comporte des ressources d'enseignement (cours et lectures) et d'apprentissage (travaux dirigés), un quiz à compléter, et quelques questions au forum.

Aujourd'hui nos **ordinateurs, téléphones** et **autres appareils** savent manipuler aussi bien des **nombres** et du texte que des images, de la vidéo ou de la musique... Mais comment **représenter**, au sein d'un **système numérique**, cette diversité des objets du **monde réel ou virtuel** ? Quelles sont les **techniques** utilisées pour représenter **numériquement** les grandeurs qui nous entourent ?

Ces différentes interrogations feront l'objet de notre leçon de la semaine



# Partie I : Généralités sur le numérique



1. Historique	10
2. Comparaison des signaux analogiques et les signaux numériques	10
3. Les avantages du numérique	11

## Objectifs

Définir les **avantages** et inconvénient du signal **numérique**

L'électronique numérique manipule des variables logiques 0 ou 1 obéissant aux règles de l'algèbre de Boole. Nous allons voir dans ce chapitre comment une information utilise des variables logiques.

## A. 1. Historique

**1900 : transmission analogique par fils, commutateurs électromécaniques.**

---



1900

**1970 : numérisation du réseau entre centraux, la transmission reste analogique entre le poste de l'abonné et le central local.**

---



1970

**1995 : Portables, GSM : transmission numérique de l'appelant à l'appelé.**

---

1995

## B. 2. Comparaison des signaux analogiques et les signaux numériques

### **Schéma de comparaison**

---

Comparaison

La plupart des **grandeurs physiques** sont **analogiques** : **température, temps, vitesse, position, pression**, etc (\*), une variable  $x$  réelle (ensemble  $\mathbb{R}$ )

### **Une grandeur numérique varie de manière discrète ( $N$ états)**

---

Toute variable qui ne peut prendre qu'un nombre fini de valeurs est une variable numérique. Ces valeurs en nombre fini sont appelées "états".

Voici quelques exemples de variables numériques :

- une variable désignant une lettre de l'alphabet (26 états)
- une variable désignant une carte dans un jeu de cartes (52 états)
- l'état d'un feu rouge (3 états)
- une variable désignant le jour de l'année (365 états)
- une variable désignant un signe du zodiaque (12 états)
- une variable  $x$  entière (ensemble  $\mathbb{N}$  ou  $\mathbb{Z}$ , pour peu qu'on se limite à un intervalle)

Il existe en fait énormément de variables numériques dans la vie courante...

## C. 3. Les avantages du numérique

### ***a- En numérique, l'information ne se dégrade pas...***

---

En électronique analogique, tout traitement de l'information (y compris une simple copie) dégrade cette information.

En électronique numérique au contraire, l'information peut être traitée ou copiée (même un grand nombre de fois) sans être dégradée (\*).

Cette "non-dégradation" apporte un avantage décisif à l'électronique numérique, notamment dans le fait que l'information peut être dupliquée et propagée beaucoup plus facilement et de manière quasiment illimitée.

### ***b- le traitement peut être très complexe***

---

Le second avantage d'une représentation numérique (=sous forme de bits et d'octets) de l'information, c'est qu'on peut faire subir à cette information des traitements beaucoup plus complexes qu'en analogique.

- en analogique, le nombre de traitements successifs applicables à un signal est limité puisque chacun de ces traitements dégrade un peu l'information,
- en numérique au contraire, le nombre de traitements successifs est virtuellement illimité, ce qui permet des traitements beaucoup plus complexes qu'en analogique.

### ***c- on peut programmer!***

---

Programmer

Notre propos ici n'est pas d'affirmer que programmer est amusant, mais plutôt d'insister sur le fait qu'en **électronique numérique** il existe des circuits qui peuvent être programmés (entre autres : **les Microprocesseurs**) ...

Un "**programme**", c'est un **ensemble d'instructions** (c'est-à-dire d'informations) qui définissent la **fonction d'un circuit**.

Un **circuit** qui peut être programmé n'a donc pas de **fonction fixe**, ce qui lui confère une **flexibilité énorme** en **termes d'applications**.

Cette possibilité (même si elle est devenue banale) est remarquable : en **numérique**, on peut donc indiquer via un **programme**, c'est-à-dire après sa fabrication, ce qu'un **circuit doit faire**.

### ***Codage binaire***

---

Le codage binaire est un puissants outils mathématiques (algèbre de Boole)

- simplification de systèmes
- équivalence de systèmes





- correction d'un signal erroné

Beaucoup de systèmes ont deux états physiques

- {présence, absence} de courant
- système {ouvert, fermé}
- surface avec {creux, bosses}
- aimantation {nord, sud}



# Partie II : Les bases de numération et la conversion d'une base à l'autre

1. Les systèmes de numérations	14
2. Conversion d'une base à une autre	15
RÉSUMÉ	17
Exercice	17

## Objectifs

Définir les différents systèmes de numération

Le système de numération le mieux adapté pour effectuer des calculs est le **système binaire, ou à base 2**, qui ne comprend que **deux caractères 0 et 1**.

Dans un système numérique quelconque, les informations circulent sous la forme de **mots binaires** formés de **suites de 1 et de 0**. On fixe à l'avance le nombre d'élément de ces mots (**un octet est un mot de huit éléments**) et la manière de les écrire appelée **code**.

Chaque élément de ces mots binaires est appelé **élément binaire** (eb) ou plus communément **bit**, contraction de l'expression "**binary digit**".

**Il existe plusieurs façons d'écrire les mots binaires car, selon les besoins, on a développé plusieurs codes.**

## A. 1. Les systèmes de numérations

### Forme générale

---

Les nombres **entiers ou décimaux** peuvent être représentés dans **plusieurs bases** différentes.

De manière générale l'expression d'un nombre en base B est de la forme :



#### Définition : Le système à base « 10 » ou décimale

---

C'est le système utilisé dans la vie courante.

Il comporte **10 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.**

Un mot (ou chiffre) est une combinaison de ces 10 caractères.

Le nombre se décompose sous la forme :



#### Définition : Le système à Base « 2 » ou binaire

---

Tout nombre binaire peut s'écrire comme un développement suivant les puissances de 2.

Les systèmes électriques et électroniques sont caractérisés par deux états :

- **interrupteur : ouvert ou fermé**
- **transistor : bloqué ou saturé**

De cette constatation est née l'idée d'utiliser le système à **base 2** ou **système binaire**.

La base 2 n'utilise que deux symboles : **0 et 1.**

L'équivalence avec les circuits électriques se fera ainsi :

- **0 = interrupteur ouvert aucun courant ne peut circuler, la lampe est éteinte correspond au transistor bloqué.**
- **1 = interrupteur fermé le courant peut circuler, la lampe est allumée correspond au transistor saturé.**



#### Exemple : Décomposition du nombre binaire 1100101

---

Le nombre binaire peut se décomposer comme suit :

**LSB (Least significant bit)** = bit de poids le **plus faible** (à droite)

**MSB (Most significant bit)** = bit de poids le **plus fort** (à gauche)

#### Définition : Étendus binaire

---

Un **mot** binaire est un ensemble de **plusieurs bits**.

Avec n bits, on peut coder de **0 à 2n-1** valeurs binaires, l'intervalle des valeurs est : **[0 ... 2n-1]**.

#### Application

- 1 **octet** (byte en anglais) = 8 bits, permet de représenter **28 = 256 valeurs**, soit

un intervalle **[0 ... 28-1] = [0 ... 255]**,

- 1 **kilo-octet** (Ko) = 210 octets = 1024 octets
- 1 **méga-octet** (Mo) = 210 kilo-octets = **220 octets**
- 1 **giga-octet** (Go) = 210 méga-octets = **230 octets**
- 1 **téra-octet** (To) = 210 giga-octets = **240 octets**



### **Exemple : Capacités des supports de stockage**

DD : 1 To = 240 octets de données possibles

Clé USB : 3 Go = 3 X 230 octets

### **Définition : Le système à base « 16 » ou hexadécimale**

Il est **très employé**, surtout en **informatique**. C'est un système numérique ayant pour **base 16**. On l'utilise pour **l'écriture condensée de nombres binaires**.

Le seul inconvénient est l'utilisation de nouveaux symboles pour les **chiffres supérieurs à 9**.

Les 16 symboles sont les suivants :

- **Dix chiffres de 0 à 9.**
- **Six lettres majuscules : A à F**

Les symboles hexadécimaux **A à F** correspondent aux valeurs décimales **10 à 15**.

Un **caractère hexadécimal** représente un **mot binaire de 4 bits**.

Cette écriture est de loin plus pratique qu'une suite de 1 et de 0.

**Caractère hexa code 4 bits (ou quartet)**

**Le principal avantage de ce code est de pouvoir codé sur un mot court, un chiffre binaire important**

### **Exemple : Conversion**

$0110\ 1011_2 = 6B_{16}$  ;  $1111\ 1111_2 = FF_{16}$

## **B. 2. Conversion d'une base à une autre**

### **Conversion décimale <-> binaire ou hexadécimale**

Pour toutes conversion à partir du système **décimal** vers une **quelconque base** de numération, nous utilisons **la méthode des divisions successives par la base**.

**Principe :**

- on **divise** le **nombre décimal** par la **base** (2, 16 ou autres), puis le **quotient** obtenu est de **nouveau divisé** par la **base** (2, 8 ou autres) jusqu'à ce qu'il devienne **NUL**.
- Les **restes successifs** sont lus de **BAS EN HAUT** représentent le nombre dans la base en question.

### **Exemple : Transformer le nombre décimal 88(10) en binaire (2)**

Division successive

Alors :  $88_{10} = 1011000_2$

### **Exemple : Transformer le nombre décimal 423(10) en Hexadécimale**

Division successive par 16 :

$423_{10} = 1A7_{16}$

### **Partie fractionnaire :**

Pour la partie fractionnaire, on procède par **produits successifs par la base (2, 16 ou autres)**.

**Principe :**

-> Le nombre décimal **fractionnaire** est **multiplié par la base** (2, 16, ou autres).

-> La **partie entière** de ce produit représente **le premier (bit ou chiffre dans la nouvelle base)** après la virgule.

-> La partie **fractionnaire** de ce premier produit est à **son tour multipliée par la base** (2, 16, ou autres).

-> La partie **entière** de ce produit représente le **deuxième** (bit ou chiffre dans la nouvelle base) après la virgule.

-> L'opération de conversion continue de la même manière jusqu'à ce que le **produit obtenu soit égal à 1.0 pour le binaire ou 0 après la virgule pour les autres bases**.

### **Exemple : conversion du nombre décimal fractionnaire : 0.84375 (10) en binaire.**

$0.84375_{10}$  :

### **Exemple : Conversion du nombre décimal fractionnaire : 0.2265625 10 en hexadécimal**

$0.2265625 \times 16 = 3.625$

$0.625 \times 16 = 10.0$

**$0.2265625_{10} = 0.3A_{16}$**

### **Conversions particulières ( hexadécimale => binaire )**

Cette conversion est très simple.

Chaque **symbole hexadécimal** est remplacé par son équivalent **binaire de 4 bits**.

### **Exemple : 9F3(16) à convertir en binaire**

$9F3_{16} = 100111110011_2$

### Conversion binaire => Hexadécimale

Dans ce cas :

- on divise le nombre binaire par tranches de 4 chiffres depuis la droite,
- puis on substitue à chaque groupe son équivalent hexadécimal.

**Exemple : 110110100111(2) à convertir en hexadécimal**

$110110100111_2 = DA7_{16}$

**Exemple : soit le nombre binaire : 10011010101.10001(2) à convertir en hexadécimal**

$10011010101.10001_2 =$

## C. RÉSUMÉ

### Format de données

Un système numérique se trouve limité par le format de données qu'il est capable de traiter.

Limites

## D. Exercice

[Solution n°1 p 31]

Quel est la valeur en décimale des codes suivants :  $111001_2$ ,  $4DD_{16}$ ,  $7A1F_{16}$ ?

☐ je sais pas

☐ 57, 125, 31273

54, 1247, 9871

57, 1245, 31263

59, 4356, 32456



# Partie III :

## Codage des nombres

1. Code pondéré	19
2. Code non pondéré	20
3. Codes correcteur d'erreur	22
Exercice	22
Exercice	22

### Objectifs

Faire les calculs de conversion entre les différents systèmes de numération

Un **code** est une **correspondance arbitraire** entre un ensemble de **symboles** et un **ensemble d'objets**. Les symboles peuvent être des **lettres**, des **chiffres**, des **signes de ponctuation** ... Certains codes permettent d'effectuer des **opérations arithmétiques**, d'autres permettent de **détecter des erreurs** lors d'une transmission de données voir de les corriger.

Nous verrons dans cette partie les différents codes utilisé pour coder les informations en informatique.

## A. 1. Code pondéré

### *Décimale, Binaire, hexadécimal*

Les **codes** décrits dans la partie précédente sont des **codes pondérés**. En effet, dans une **base donnée**, chaque **bit** est affecté d'un **poids proportionnel à sa position**.

Il existe d'autres types de codes pondérés notamment les représentations décimales codées binaire.



### Code BCD (Binary Coded Decimal ou décimal codé binaire)

Il s'agit d'une représentation des nombres en base 10, où chaque chiffre est codé en binaire. Il faut **4 bits** pour représenter les 10 chiffres de la base 10 et chaque bit d'un groupe de 4 est affecté de son poids naturel.

On écrit ainsi par exemple :  $421_{10} = 0100\ 0010\ 0001_{\text{BCD}}$

En binaire naturel on obtient après les divisions successive par la base 2 :  $110100101_2$ .

En code **BCD** un nombre de n chiffres occupe toujours **4 x n bits**.



#### Remarque : Conversion BCD $\Leftrightarrow$ binaire naturel

<< Le passage d'un code binaire naturel au code BCD (ou inversement) nécessite le passage absolu par la valeur décimale.>>



#### Méthode : Conversion

- (Valeur BCD)  $N_{\text{BCD}} \Rightarrow$  convertir en décimal  $\Rightarrow$  convertir en binaire ( $N_2$ )
- (Valeur Binaire naturel)  $N_2 \Rightarrow$  convertir en décimal  $\Rightarrow$  l'exprimer en BCD ( $N_{\text{BCD}}$ ).

Tableau de Concordance

## B. 2. Code non pondéré

### Le code ASCII (American Standard Code for Information Interchange).

C'est le code le plus utilisé dans les **transmissions** entre une **unité centrale** et ses **périphériques**.

$\Rightarrow$  Il sert à coder des **lettres**, des **chiffres** et un **certain nombre** d'ordres qui correspondent souvent aux **touches du clavier** (par exemple la touche **ENTREE**).

Ces symboles sont codés en binaire sur 7 bits ce qui permet  $2^7 = 128$  possibilités.

#### Exemple : Code ASCII

Les caractères ASCII zéro **0** et **espace** (touche espace du clavier) sont respectivement codé en Hexadécimal par **30<sub>16</sub>** et par **20<sub>16</sub>**.

(cf. asciiexemp p 21)Exemple Code ASCII

Une table plus complète est donnée ci-dessous :

(cf. ascii p 21)Tableau des codes ASCII

La transmission d'une information se fait en réalité sur **8 bits**. Le **dernier bit** est en principe **un bit de parité** servant à la **détection des erreurs** :

- Il est mis à **0** si le **nombre de bits** du signal est **pair**,
- il est mis à **1** dans le cas **inverse**.

On peut ainsi détecter une erreur se produisant sur un bit.

En réalité le code ASCII, qui a été mis au point pour la langue anglaise, ne contient pas de caractères accentués ni de caractères propres à une langue.

Le **8ème bit** est donc souvent utilisé pour transmettre ces caractères. On parle alors de **code ASCII étendu**.

### Code Adjacents

Lorsque deux chiffres ou nombres consécutifs ont toujours des représentations **qui ne diffèrent que par un seul bit**, on dit qu'il s'agit d'un **code adjacent**.

Si **l'adjacence est complète** (avec retour au point de départ) on parle de **code cyclique**.

Il a été élaboré pour éviter les risques d'erreur de détection d'une information dans les systèmes réels.

C'est le cas des **codes binaires réfléchit** aussi appelé **code GRAY**.

Règle lorsque l'on compte :

### Conversion binaire naturel <=> binaire réfléchit

- Le **bit de gauche** du **code Gray** est le **même** que le **bit de gauche** du **nombre binaire**.
- Ajouter le **MSB** du **nombre binaire** à son **voisin immédiat** et **reporter la somme en négligeant une retenue** éventuelle sur la ligne inférieure correspondante au code Gray.
- Continuer **l'addition des bits** à **leur voisin de droite** et **reporter les sommes** ainsi obtenues jusqu'à **atteindre le LSB**.
- Le nombre en **code Gray** comportera toujours le **même nombre de bits** que le **binaire original**.

**Exemple : soit à convertir le binaire naturel ; 10110 (2) en code Gray**

binaire Gray

### Conversion binaire réfléchit <=> binaire naturel

- Le **bit de gauche** du **nombre binaire** est le **même** que le **bit de gauche**

**du code Gray.**

- Ajouter le MSB du nombre binaire obtenu au voisin de droite immédiat du code Gray.
- Continuer les additions jusqu'à atteindre le LSB.

**Exemple : soit à convertir le code Gray  $NG = 1001G$  en binaire naturel**

conversion binaire Gray

## C. 3. Codes correcteur d'erreur

### code

Les transmissions numériques nécessitent des rapports signal/bruit beaucoup plus faibles que les transmissions analogiques. Cela étant, même si un rapport signal/bruit de 1 est acceptable il est toujours possible qu'un bit soit modifié lors d'une transmission de données. Des codes ont donc été développés pour détecter et éventuellement corriger ces erreurs

## D. Exercice

[Solution n°2 p 31]

1. Quelles sont les valeurs : binaire naturel, BCD et binaire réfléchi du nombre décimal 145 ?

	je sais pas
	$1110001_2$ ; $100101000101_{BCD}$ ; $11011001_G$
	$10011101_2$ ; $010101000101_{BCD}$ ; $1101100101_G$
	$10010001_2$ ; $000101000101_{BCD}$ ; $11011001_G$
	$101110001_2$ ; $1101000101_{BCD}$ ; $1101100111_G$

## E. Exercice

[Solution n°3 p 32]

Quelles sont les valeurs : binaire naturel, Hexadécimal, décimale et BCD du code binaire réfléchi  $110011011_G$  ?

---

Partie III : Codage des nombres

je sais pas

$1110001110_2$  ;  $A34_{16}$  ;  $6571_{10}$  ;  $100101000101_{BCD}$

$100010010_2$  ;  $112_{16}$  ;  $274_{10}$  ;  $001001110100_{BCD}$

$1110101110_2$  ;  $AC4_{16}$  ;  $571_{10}$  ;  $100101110101_{BCD}$

$10001110_2$  ;  $ADD4_{16}$  ;  $6571_{10}$  ;  $100101000101_{BCD}$



# Partie IV : Représentation des nombres et arithmétique binaire

1. Représentation des nombres négatifs	24
2. Arithmétique binaire	25
Exercice	26
Exercice	26

## Objectifs

**Manipuler les différentes représentations des éléments de numérations.**

Dans un circuit électronique (par exemple un ordinateur), les nombres sont représentés par des bits 0 ou 1 stockés dans des mémoires ou des registres (physiquement cela correspond à des états différents de tension électrique ou d'aimantation). Un groupe de 8 bits forme un octet.

Les circuits électroniques manipulent des mots formés de plusieurs octets. Les ordinateurs actuels, par exemple, utilisent des mots de 8 octets c'est-à-dire 64 bits.

Les nombres représentables sont donc en nombre fini. On ne peut pas tout représenter ni tout calculer. Les mots de 64 bits manipulés par les ordinateurs autorisent 264 représentations différentes soit quelques milliards.

Dans ces conditions se pose la question de la représentation la plus appropriée pour les nombres négatifs.

## A. 1. Représentation des nombres négatifs

### *mode de représentation*

Pour les nombres exprimés en **code binaire naturel** il existe au moins **trois types de représentations** :

- Représentation par un bit de signe et une valeur absolue
- Représentation par le complément à 1 ou complément restreint (CR).
- Représentation par le complément à 2

### *Représentation par un bit de signe et une valeur absolue*

Le premier bit indique le signe :

- **0 pour le signe + (positif)**
- **1 pour le signe – (négatif)**

Le reste des bits représente la valeur absolue (en base 2)

**Exemple : Par exemple avec 3 bits on représente les nombres :**

Nombre signé

L'inconvénient de cette convention est double ! Il y a deux représentations possibles pour le nombre 0.

Par ailleurs la **soustraction** vue comme une **addition bit à bit ne fonctionne pas**.

**Exemple : Soustraction**

En effet :

### *Représentation par le complément à 1 ou complément restreint (CR).*

Le complément à 1 d'un nombre binaire s'obtient en **changeant chaque 0 par un 1 et chaque 1 par un 0**. Autrement dit, en complémentant chaque bit du nombre.

Voici une illustration de cette marche à suivre :

1 0 1 1 0 1 : nombre **binaire initial**

0 1 0 0 1 0 : **complément de chaque bit pour obtenir le complément à 1**

On dit que le complément à 1 de 101101 est 010010.

Le complément à 1 d'un nombre est donc l'inversion de chaque bit à l'aide de la fonction logique NON. Nous pouvons donc exprimer le complément à 1 par l'équation ci-dessous.

Complément à 1 de N : **CR(N) = not N**.

=> Cette représentation permet d'effectuer des **soustractions en binaire**.

## **Représentation par le complément à 2**

---

Le complément à 2 est très largement utilisé car c'est la **représentation naturelle des nombres négatifs**.

Si nous faisons la soustraction de  $2 - 3$ ,

Nous obtenons immédiatement -1 représenté en complément à 2.

Nous allons voir que "1111" est la représentation du nombre -1 sur 4 bits.

Le complément à 2 d'un nombre binaire s'obtient simplement en prenant le complément à 1 de ce nombre et en ajoutant 1 au bit de son rang de poids le plus faible.

Voici une illustration de cette conversion pour le cas  $101101_2 = 45_{10}$ .

Le complément à 2 d'un nombre est donc l'inversion de chaque bit à l'aide de la fonction logique NON, puis l'addition de 1. Nous pouvons donc exprimer le complément à 2 par l'équation ci-dessous.

Complément à 2 de N :  **$C2(N) = C1(N) + 1 = \text{not } N + 1$**

Voici la valeur du nombre -1 sur 4 bits.

Nous commençons par exprimer le nombre 1 sur 4 bits puis nous appliquons la règle de calcul du complément à 2.

## **B. 2. Arithmétique binaire**

### **Addition Binaire**

---

=> on commence par additionner les chiffres du rang de **poids faible**.

### **Soustraction Binaire**

---

On distingue deux cas pour la soustraction en binaire :

- Le cas des **calculs simple** est résumé dans le tableau suivant :
- Le cas des calcul **complexes**, on utilise le **complément à 2** du nombre à soustraire. Dans ce cas, il peut aussi intervenir le **bit de signe**.

### **Exemple : Premier cas de figure**

---

Cas complexe



### Exemple : Deuxième cas de figure

Cas négatif

## C. Exercice

[Solution n°4 p 32]

Quelles sont les valeurs : compléments à 1 et complément à 2 du nombre décimal 145 ?

	je sais pas
	$1110001_2$ ; $100101000101_2$
	$01101110_2$ ; $01101111_2$
	$100100011_2$ ; $00010100001_2$
	$101110001_2$ ; $1101000101_2$

## D. Exercice

[Solution n°5 p 32]

Quelles sont les résultats en utilisant le complément à 2, des opérations décimales suivantes :  $20 - 15$ ,  $125 - 207$  ?

	je sais pas
	$1110001110_2$ ; $100101000101_2$
	$1110101110_2$ ; $100101110101_2$
	$0101_2$ ; $1010\ 1110_2$
	$0111_2$ ; $0011\ 0001_2$



# Test de compréhension de la leçon 1

## Exercice 1

[Solution n°6 p 33]

Quelles sont les valeurs Hexadécimal puis binaire du nombre décimale  $(7852)_{10}$

<input type="checkbox"/>	Je ne sais pas
<input type="checkbox"/>	$(ABEAC)_{16}$ , $(1\ 0000\ 1010\ 1100)_2$
<input type="checkbox"/>	$(1EAC)_{16}$ , $(1\ 1110\ 1010\ 1100)_2$
<input type="checkbox"/>	$(FEBC)_{16}$ , $(1\ 1110\ 1010\ 1111)_2$
<input type="checkbox"/>	$(EEAC)_{16}$ , $(1\ 0100\ 1010\ 1100)_2$

## Exercice 2

[Solution n°7 p 33]

Quelles sont les valeurs Hexadécimal puis décimale du nombre binaire :  $(1101001011)_2$

<input type="checkbox"/>	Je ne sais pas
<input type="checkbox"/>	$(353B)_{16}$ , $(1233)_{10}$
<input type="checkbox"/>	$(34CC)_{16}$ , $(9443)_{10}$
<input type="checkbox"/>	$(234B)_{16}$ , $(6743)_{10}$
<input type="checkbox"/>	$(34B)_{16}$ , $(843)_{10}$

## Exercice 3

[Solution n°8 p 33]

On représente des entiers signés sur 16 bits.

Quel est le plus grand entier positif et le plus petit entier négatif que l'on puisse écrire?

<input type="checkbox"/>	$(0111111111111111)_2$ , $(1111111111111111)_2$
<input type="checkbox"/>	$(1111111111111111)_2$ , $(1101111111110111)_2$
<input type="checkbox"/>	$(000111111111)_2$ , $(1100111111111111)_2$

### Exercice 4

[Solution n°9 p 33]

Donner les compléments à 1 et 2 de l'entier le plus grand de l'exercice précédent

	Je ne sait pas
	$(1\ 000\ 0000\ 0000\ 0000)_2$ , $(1\ 000\ 0000\ 0000\ 0001)_2$
	$(1\ 000\ 0000\ 11111\ 0000)_2$ , $(1\ 1111\ 0000\ 0000\ 0001)_2$
	$(1\ 000\ 0000\ 0000\ 0000)_2$ , $(1\ 000\ 0000\ 0000\ 0001)_2$





# Solution des exercices

## > Solution n°1 (exercice p. 17)

☐ je sais pas

☐ 57, 125, 31273

54, 1247, 9871

☒ 57, 1245, 31263

59, 4356, 32456

La conversion en décimale consiste à la décomposition des nombres dans les base respectives.

Exemple : Base 2 ;  $111001 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1 \times 32 + 1 \times 16 + 1 \times 8 + 0 + 1 = \mathbf{57}$ .

## > Solution n°2 (exercice p. 22)

je sais pas

$1110001_2$  ;  $100101000101_{\text{BCD}}$  ;  $11011001_{\text{G}}$

$10011101_2$  ;  $010101000101_{\text{BCD}}$  ;  $1101100101_{\text{G}}$

☒  $10010001_2$  ;  $000101000101_{\text{BCD}}$  ;  $11011001_{\text{G}}$

$101110001_2$  ;  $1101000101_{\text{BCD}}$  ;  $1101100111_{\text{G}}$

## > Solution n°3 (exercice p. 22)

je sais pas
$1110001110_2$ ; $A34_{16}$ ; $6571_{10}$ ; $100101000101_{BCD}$
$100010010_2$ ; $112_{16}$ ; $274_{10}$ ; $001001110100_{BCD}$
$1110101110_2$ ; $AC4_{16}$ ; $571_{10}$ ; $100101110101_{BCD}$
$10001110_2$ ; $ADD4_{16}$ ; $6571_{10}$ ; $100101000101_{BCD}$

#### > Solution n°4 (exercice p. 28)

je sais pas
$1110001_2$ ; $100101000101_2$
$01101110_2$ ; $01101111_2$
$100100011_2$ ; $00010100001_2$
$101110001_2$ ; $1101000101_2$

Il faut dans un premier temps faire la conversion du nombre décimal en binaire naturel et ensuite faire les différentes complémentations.

Pour le complément à 1, on inverse tous les bits du nombre binaire naturel obtenu après les divisions successives.

Le complément à 2 est obtenu par une simple addition de 1 au résultat du complément à 1.

#### > Solution n°5 (exercice p. 28)

je sais pas
$1110001110_2$ ; $100101000101_2$
$1110101110_2$ ; $100101110101_2$
$0101_2$ ; $1010\ 1110_2$
$0111_2$ ; $0011\ 0001_2$

La première soustraction est une opération simple, qui se fait avec la soustraction ordinaire en binaire.

Cependant la deuxième nécessite le passage obligatoire par le complément à 2.

En effet, on fait l'addition du binaire de 125 avec le binaire qui est le complément à 2 du binaire du nombre 207.

$$C2(217) = C1(217) + 1.$$

Le résultat obtenu représente le complément à 2 du résultat.

=>  $1010\ 1110$  est le complément à 2 du résultat.

=> il faut dé-complémenter pour l'obtenir.

#### > Solution n°6 (exercice p. 29)

Je ne sais pas

 $(ABEAC)_{16}$  ,  $(1\ 0000\ 1010\ 1100)_2$  $(1EAC)_{16}$  ,  $(1\ 1110\ 1010\ 1100)_2$  $(FEBC)_{16}$  ,  $(1\ 1110\ 1010\ 1111)_2$  $(EEAC)_{16}$  ,  $(1\ 0100\ 1010\ 1100)_2$ 

on applique la méthode des subdivisions successive par 16.

ce qui donne la valeur :  $(1EAC)_{16}$  , la conversion vers la base 2 est immédiate :  
 $(1EAC)_{16} = (1\ 1110\ 1010\ 1100)_2$

### > Solution n°7 (exercice p. 29)

Je ne sais pas

 $(353B)_{16}$  ,  $(1233)_{10}$  $(34CC)_{16}$  ,  $(9443)_{10}$  $(234B)_{16}$  ,  $(6743)_{10}$  $(34B)_{16}$  ,  $(843)_{10}$ 

Vers la base hexadécimale on fait un regroupement de 4 bit en commençant par la droite soit  $(34B)_{16}$

Pour la conversion en décimale, on fait un développement en polynôme de puissance de 16.

### > Solution n°8 (exercice p. 29)

 $(0111111111111111)_2$  ,  $(1111111111111111)_2$  $(1111111111111111)_2$  ,  $(1101111111110111)_2$  $(000111111111)_2$  ,  $(1100111111111111)_2$ 

Les deux valeurs sont identique en valeur absolue.

### > Solution n°9 (exercice p. 30)

Je ne sait pas

 $(1\ 000\ 0000\ 0000\ 0000)_2$  ,  $(1\ 000\ 0000\ 0000\ 0001)_2$  $(1\ 000\ 0000\ 11111\ 0000)_2$  ,  $(1\ 1111\ 0000\ 0000\ 0001)_2$  $(1\ 000\ 0000\ 0000\ 0000)_2$  ,  $(1\ 000\ 0000\ 0000\ 0001)_2$ 

Le complément à 1 s'obtient en complétant tous les bits composants le nombre soit  $1\ 000\ 0000\ 0000\ 0000$ . Le complément à 2 s'obtient en ajoutant 1 au complément à 1. On a ainsi :  $1\ 000\ 0000\ 0000\ 0001$