

Université Abdelhamid Mehri – Constantine 2

2020-2021. Semestre 1

Codification et représentation de l’information (CRI)

|  |
| --- |
| Chapitre 01 : Codification et représentation des nombres  Chapitre 02 : Codification α-Numérique  Chapitre 03 : Algèbre de Boole |



Image représentant le cours/chapitre

|  |  |  |  |
| --- | --- | --- | --- |
| Staff pédagogique | | | |
| **Nom** | **Grade** | **Faculté/Institut** | **Adresse e-mail** |
| Bahri Mohamed redha | MCB | Nouvelles Technologies | Reda.bahri@univ-constantine2.dz |

|  |  |  |  |
| --- | --- | --- | --- |
| Etudiants concernés | | | |
| **Faculté/Institut** | **Département** | **Année** | **Spécialité** |
| Nouvelles Technologies | MI | L1 | MI |

# Objectifs du cours

* Le premier objectif dans ce chapitre est d’initier l’étudiant à la codification et la représentation des nombres.
* Le deuxième objectif est de familiariser l’étudiant à la notion du binaire et le code machine.

1. Chapitre 1 : Codification et représentation des nombres

1.1. Introduction :

L’informatique ce trouve de nos jours dans toute activité humaine. L’ordinateur, se trouve alors comme l’outil le plus répondu au monde. Pour comprendre comment fonctionne un ordinateur, il faut d’abord comprendre comment il représente et traite l’information de manière générale.

Dans ce chapitre nous allons focaliser sur la codification et la représentation des nombres.

La machine manipule uniquement deux valeurs binaires (0 et 1), pour la représentation et le traitement de l’information et notamment des nombres, ceci motive l’intérêt qui va être porté dans ce chapitre à la codification et la représentation binaire.

1.2. Les entiers positifs :

L’ordinateur calcule en binaire, seuls les symboles 0 et 1 sont utilisés pour représenter et manipule les nombres. Par ailleurs la taille des nombres manipulésétant limitée, les espaces de stockage ont une taille limitée, ceci limite donc les capacités de l’ordinateur à exprimer les nombres.

1.2.1. Représentation des entiers positifs dans une base B

De même qu’en décimal on utilise dix chiffres pour représenter les nombres, dans une base B (B est un entier et B ≥ 2) on utilise B chiffres pour la représentation des nombres :

Soit une base B dont l’ensemble des chiffres est : E=

Chaque nombre X exprimé en base B sera représenté par une concaténation des symboles de l’ensemble E.

**Exemples :**

* en base 10 : **X = 19372** , E={0,1,2,3,4,5,6,7,8,9} ; Le nombre X est représenté par les chiffres 1,9,3,7 et 2 de la base 10.
* en base 2 : **X = 10111**, E={0,1} ; Le nombre X est représenté par les chiffres 1 et 0.

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene."Remarque : Pour ne pas avoir d’ambigüité dans la représentation des nombres il faut toujours préciser la base utilisée pour la représentation du nombre, sauf pour la base 10. |

**Exemple :**

**X = 101** en base 2 est différent de **X = 101** en base 10.

Donc, **X = 101** en base 2 sera exprimé **X=(101)2**

* + - 1. Représentation d’un entier positif dans une base B

Soit **X** un nombre entier positif et soit **B** une base (Bϵ N et B ≥ 2).

Pour obtenir la représentation de **X** dans la base **B** on applique le principe de la division euclidienne :

**X = q.B + r.**

* On divise à chaque fois le **q** par **B** et on note le reste.
* La division est répétée jusqu’à ce que le **q** soit inférieure à **B**.
* Les restes obtenus respectivement dans l’opération de la division en plus de la dernière valeur du **q** seront notés de droite à gauche, dans l’expression du nombre.

**Exemple**

Soit le nombre entier X= 44 exprimé en base 10, pour exprimer X en base 3 on réalise la division successive par 3 :

44 = 14\*3 + **2**

14 = 4\*3 + **2**

4 = **1**\*3 + **1**

Le dernier reste est **1**, **1** est inférieur à **3**, donc on arrête le processus de la division.

Ainsi le nombre décimal **44** exprimé en base **10** sera exprimé **(1122)** en base **3**

* + - 1. Valeur décimale d’un nombre exprimer en base B

La valeur décimale d’un nombre **X** exprimé en base **B** est obtenue en multipliant chaque chiffre de la base **B**, de droite à gauche dans la représentation de **X**, par une puissance croissante (commençant par zéro) de la base. Les valeurs ainsi obtenues seront additionnées.

**Exemples**

1. Soit le nombre X exprimé en base 7, X = (3425)7

La valeur décimale de x est :

, d’où

1. Soit le nombre X exprimé en base 2, X = (1101)2

La valeur décimale de x est :

, d’où

1.2.2 Les bases classiques :

L’informatique généralement se sert de quelques bases classiques notamment : la base **2 (binaire)**, la base **8 (octale)**, la base **10 (décimale)** et la base **16 (Hexadécimale)**.

Dans cette section nous présentons la conversion des nombres binaires en octale et en hexadécimale car ceci joue un rôle prépondérant dans les systèmes digitaux.

Sachant que chaque symbole binaire occupe un **bit** (**0** ou **1**), un nombre binaire et donc une collection de bits.

Sachant aussi que le nombre **8** est une puissance **3** de **2** et que le nombre **16** est une puissance **4** de **2**.

Donc, chaque chiffre octal correspond à **3 bits** et chaque chiffre hexadécimal à **4 bits**.

1.2.2.1. Conversion binaire – Octal

La conversion du binaire en octal se fait par un groupement de 3 bits de droite à gauche dans la représentation binaire du nombre, chaque groupement sera remplacé par le chiffre équivalent en base octale.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Binaire** | **000** | **001** | **010** | **011** | **100** | **101** | **110** | **111** |
| **Octale** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**Exemples :**

Soit le nombre entier X exprimé en binaire **X = (1101001)2**

La conversion de X en octal se fait par groupement de trois bits de droite à gauche, de la manière suivante :

X = **001** **101 001** donc x**=(151)8**

**1 5 1**

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene."Remarque : les bits manquants à gauche sont remplis par des zéros. |

1.2.2.2. Conversion binaire hexadécimal

La conversion du binaire en hexadécimal se fait par un groupement de 4 bits de droite à gauche, chaque collection sera remplacée par le chiffre équivalent en base hexadécimal.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Binaire** | **0000** | **0001** | **0010** | **0011** | **0100** | **0101** | **0110** | **0111** |
| **Hexadécimale** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Binaire** | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| **Héxadécimale** | 8 | 9 | A | B | C | D | E | F |

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene."Remarque : les chiffres de la base 16 sont : |

**Exemple :**

Soit le nombre entier X exprimer en binaire **X=(1101111011)2**

La conversion de X en hexadécimal se fait par groupement de quatre bits de droite à gauche, de la manière suivante :

**X=0011 0111 1011**donc X**= (37B) 16**

**3 7 B**

Les bits manquants à gauche sont remplis par des zéros.

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene."Remarque : La conversion directe d’une Base **B** vers une Base **K** est possible si l’une des deux base est une puissance entière de l’autre **(ex : B=3, K= 9)** et inversement. |

1.2.3. Arithmétique

Les opérations arithmétiques classiques, pour le autres bases, se réalisent de la même manière qu’en décimal. Dans la suite de ce cours nous exposons les opérations arithmétiques binaires à titre d’exemples illustratifs. Les détails seront exposés en travaux dirigés.

1.2.3.1. L’addition binaire :

L’addition de deux bits binaire se réalise selon la spécification suivante :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bit1** |  | **Bit2** |  | **Résultat** | **Retenue** |
| **0** | **+** | **0** | **=** | **0** | **0** |
| **0** | **+** | **1** | **=** | **1** | **0** |
| **1** | **+** | **0** | **=** | **1** | **0** |
| **1** | **+** | **1** | **=** | **0** | **1** |

L’addition binaire de deux nombres s’effectue bit à bit de droite à gauche, en reportant les retenues

**Exemple :**

Soit les deux nombres **X = (1101101)2** et **Y = (10101011)2**

L’opération d’addition se réalise de la même manière que dans la base 10

10111101111101

+ **X+Y = (100011000)2**

1 01 0 1 0 11

10 00 1 1 0 00

1.2.3.2. la soustraction binaire :

La soustraction de deux bits binaire se réalise selon la spécification suivante :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bit1** |  | **Bit2** |  | **Résultat** | **Retenue** |
| **0** | **-** | **0** | **=** | **0** | **0** |
| **0** | **-** | **1** | **=** | **1** | **1** |
| **1** | **-** | **0** | **=** | **1** | **0** |
| **1** | **-** | **1** | **=** | **0** | **0** |

La soustraction binaire de deux nombres s’effectue bit à bit de droite à gauche, en reportant les retenues

NB : plus de détailles seront effectués en travaux dirigés.

1.2.3.3. La multiplication binaire :

La multiplication de deux bits binaire se réalise selon la spécification suivante :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit1** |  | **Bit2** |  | **Résultat** |
| **0** | **X** | **0** | **=** | **0** |
| **0** | **X** | **1** | **=** | **0** |
| **1** | **X** | **0** | **=** | **0** |
| **1** | **X** | **1** | **=** | **1** |

**Exemple :**

Soit les deux nombres **X= (110 10)2** et **Y= (1010)2**

L’opération de multiplication X \* Y se réalise de la même manière que dans la base 10

11010

x 1010

= 00000

+ 11010 **.**

+ 00000 **. .**

+11010 **. . .**

= 100000100

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene."Remarques :  * Le produit de deux nombres positifs est toujours plus grand que les nombres multipliées, ceci pose un problème pour la représentation interne des nombres (taille et capacité). * La division binaire est une opération assez complexe généralement on utilise le même principe que le décimal (division par soustraction). * Le développement d’arithmétique dans les différentes bases sera exposé dans les séances des travaux dirigés. |

1.3.Les entiers négatifs

Dans le calcul binaire les nombres entiers positifs sont représentés uniquement en 0 et 1. Le signe négatif précédant les valeurs entières pose un problème de représentation sur machine.

Plusieurs méthodes sont utilisées pour représenter les nombres négatifs dans un ordinateur, parmi lesquelles nous citons : la représentation en signe et valeur absolue **(SVA)**, le complément à 1 **(CP1)** et le complément à 2 **(CP2)**.

1.3.1. Représentation des nombres entiers négatifs en SVA (Signe et valeur absolue)

1.3.1.1. Principe

* Les nombres entiers en SVA seront codés en binaire par ± valeur absolue, le bit le plus fort du nombre représenté, sera réservé pour le signe (**1** pour le signe **‘-‘** et **0** pour le signe **‘+’**).
* Un nombre entier codé sur n bits aura une valeur absolue **V** :

1.3.1.2. Propriété

Soit un nombre entier X codé en SVA sur n bits, la plage de la représentation de X est :.

Sachant que sur n bits nous avons **2n** représentations, et de **0** à **2n-1-1** nous avons **2n-1**nombres. Donc, le zéro compte 2 fois **(+0)** et **(-0)**.

Le zéro possède alors deux représentations : **(-(2n-1-1),……-1,-0,+0,1,2,…,(2n-1-1))**.

**Exemple1 :**

Sur **4** bits nous aurons 16 nombres entiers :

**(-7,-6,-5,-4,-3,-2,-1,-0,+1,+1,+2,+3,+4,+5,+6,+7)**.

***NB*** : pour un nombre codé sur un mot de n bits nous aurons réellement 2n-1 nombre différents (le zéro possède deux codes)

**Exemple2 :**

Le nombre entier **X=-20** sera codé en **SVA** sur **6** bits minimum comme suit :

**20 =(10100)2**

**X sera codé par : 110100.**

**Exemple3 :**

Le nombre entier **X=+ 20** sera codé en SVA sur **6 bits** comme suit :

**20 = (10100)2**

**X sera codé par : 010100.**

**Exemple4 :**

Le nombre entier **X=-20** sera codé en SVA sur **8 bits** comme suit :

**20 =(10100)2**

**X sera codé par : 10010100.**

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene."Remarque : En **SVA**, les opérations d’addition et de soustraction sont compliquées car le bit de signe doit être traité à part. Cette méthode n’est pas utilisée par les constructeurs de calculateur pour la réalisation des opérations d’addition et de soustraction. Les applications d’arithmétique SVA seront plus détaillées en travaux dirigées. |

**1.3.2. Représentation des nombres entiers négatifs en complément à 1 (CP1)**

* + - 1. **Principe :**

Un nombre entier négatif est obtenu en complément à 1(complément logique) par remplacement de chaque bit à 0 par 1 et inversement, dans le code binaire du nombre positif. Le bit le plus fort dans le code du nombre est réservé pour le signe (**1** pour le signe ’**-**‘ et 0 pour le signe ‘**+**’).

**Exemple**:

Soit le nombre entier **X = - 5** codé sur 4 bits ;

Nous avons : +5 =**(0101)2** en binaire (le bit le plus fort est un bit de signe, 0 pour le signe positif).

Le **CP1 (+5)= - 5**, donc X **= 1010** en**CP1.**

* + - 1. **Propriétés :**
* En complément à 1, le**CP1(X)** et le symétrique de **(X).**
* Donc, les nombres entiers codés sur **n** bits en CP1seront aux nombres de**2n-1** nombre positifet**2n-1** nombre négatif.
* la plage de la représentation d’un entier X codé sur n bits est : .
* Sachant que sur **n** bits nous avons **2n** représentations, et de **0** à **2n-1-1** nous avons **2n-1** nombres. Donc, le **zéro** compte 2 fois **(+0)** et **(-0)**.

***NB****:* la représentation en CP1 est facile à réaliser électroniquement.

* + - 1. **ARITHMETIQUE**
* **L’Addition en CP1 :**

**Principe** :

* Dans l’addition une retenue générée par le bit de signe doit être ajoutée au résultat obtenue car on effectue l’addition des compléments y compris le bit de signe (report de la retenue). Une condition nécessaire pour ajouter la retenue, est que une retenue doit être aussi générer par le bit juste avant signe.
* Si une retenue est générée exclusivement par les bits de signe ou par le bit juste avant signe, l’addition est impossible a réalisée. une erreur de dépassement de capacités est déclarée.

**Exemple1** :

Soit les deux entiers **X= 7** et **Y = 6**,

On veut réaliser L’opération **X-Y** sur **4 bits** en **CP1**

7-6 = (7) + (-6) donc, 7- 6 = 7 + CP1(6).

**7= (0111)2** et **6= (0110)2** avec **CP1(6) = 1001**

Alors, **1** 01 11 1 1

+

1 0 0 1

= **1** 0 0 0 0 une retenue est générée par le bit de

signe et le bit avant signe

+ 1

= 0 0 0 1 le résultat est correcte 7 – 6 = 1

**Exemple2** :

Soit les deux entiers **X= 7** et **Y = 6**

On veut réaliser l’opération **X+Y** sur **4 bits** en **CP1**

**7= (0111)2** et **6= (0110)2**

Alors, **1** 0 11 1 1

+

0 1 1 0

= 1 1 0 1 **Résultat incorrecte**

Une retenue est générée uniquement par le bit avant signe.

L’addition est incorrecte (7 + 6 = 13, le nombre signé +13 nécessite 5 bits pour la codification. Un dépassement de capacités doit être signalé dans ce cas).

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene."Remarque : En CP1 la soustraction d’un nombre se réduit à l’addition de son complément à 1. (le circuit de soustraction et inutile). En plus le CP1 ne traite pas le bit de signe à part. |

* + 1. Représentation en complément à 2 (CP2)
       1. **Principe :**
* Le complément à 2 (complément arithmétique) d’un nombre entier négatif est obtenu en rajoutant +1 à sa représentation CP1.
* Cette représentation donne une configuration de plus car le zéro aura une seul représentation.

**Exemple**:

Soit le nombre entier **X = - 5** codé sur **4**bits ;

Nous avons : **X = 1010** en **CP1.**

Le **CP2(+5)= CP1(+5) +1, donc X = 1011 en CP2.**

* + - 1. **Propriétés**
* La plage de la représentation d’un entier X codé sur n bits en CP2 est :
* Sur **n** bits on a : **X+CP2(X)=2n** (d’où le nom complément à 2)
* La valeur d’un nombre représenter en CP2 revient à lire un nombre binaire

Xn-1Xn-2…X0de la façon suivante :

Donc**, -**

* Le CP2 (CP2(X)) = X pour toute valeur entière X.
  + - 1. **ARITHMETIQUE**

**L’Addition en CP2 :**

***Principe****:*

* dans l’addition en CP2 une retenue générée par le bit de signe sera ignorée. Une condition nécessaire pour accepter le résultat ainsi obtenu, est qu’une retenue doit être aussi générer par le bit juste avant signe.
* Si nue retenue est générée exclusivement par les bits de signe ou par le bit juste avant signe, l’addition est impossible a réalisée. une erreur de dépassement de capacités est déclarée.

***Exemple1****:*

Soit les deux entiers **X= 7** et **Y = 6**

On veut réaliser l’opération **X-Y** sur **4 bits** en **CP2**

7- 6 = (7)+(-6) donc, 7- 6 = 7 + CP2(6).

**7= (0111)2** et **6= (0110)2 avec CP1(6) = 1001** donc**CP2(6) = 1010**

Alors, **1**0 11 1 1

+

1 0 1 0

= **1** 0 0 0 1

Une retenue est générée par le bit de signe et le bit avant le bit de signe. Elle sera ignorée et le résultat est correcte (7 – 6 = 1).

***Exemple2****:*

Soit les deux entiers **X= 7** et **Y = 6,**

On veut réaliser l’opération **X+Y** sur **4 bits** en **CP2**

**7= (0111)2** et **6= (0110)2**

Alors, **1** 0 11 1 1

+

0 1 1 0

= 1 1 0 1 **Résultat incorrecte**

Une retenue est générée uniquement par le bit avant signe.

L’addition est incorrecte (7 + 6 = 13, le nombre signé +13 nécessite 5 bits pour la codification et la représentation. Un dépassement de capacités doit être signalé dans ce cas).

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque :  * Soustraire un nombre reviens à additionner son opposé c-à-d son complément à 2. * La multiplication de deux nombres revient à multiplier les valeurs absolues des 2 nombres et de prendre l’opposé du résultat si les nombres sont de signe différent. * Pour diviser, il faut passer par la division des valeurs absolues et repositionner les signes du quotidien est du reste si nécessaires |

1.4. Les Nombres réels

L’ordinateur manipule uniquement des bits, il faut donc définir une représentation des nombres décimaux adaptés aux calculs. En plus l’espace de stockage d’un nombre sur une machine est limité, ce qui induit à une limite de la précision des calculs.

L’idée de base pour la représentation des nombres décimaux est de définir l’emplacement d’une virgule séparant la partie entière de la partie décimale et de considérer que les chiffres à droite de la virgule correspondent aux puissances négatives de la base (comme pour les décimaux).

1.4.1. Représentation des nombres réels en virgule fixe :

L’objectif du calcul fractionnaire est d’avoir le maximum possible de précision dans l’écriture des nombres fractionnaires. L’emplacement de la virgule dans l’écriture du nombre permet de déterminer cette précision. Malheureusement, dans un ordinateur la taille de la représentation est limitée, ainsi la gestion de la position de la virgule pose un réel problème.

1.4.1.1. Conversion base B base décimale

Dans la représentation d’un nombre ou virgule fixe, on choisit une position dans les chiffres du nombre pour séparer la partie fractionnaire à droite de la position choisie, et la partie entière partant de la position choisie vers la gauche.

La conversion de la partie fractionnaire est obtenue par multiplications successives par des puissances négatives et décroissantes la base B utilisée en commençant par (-1).La partie entière est obtenue selon le principe des Nombres entiers.

Ainsi, pour coder un nombre réel sur n positions, on peut par exemple définir une virgule après les K chiffres de poids faible, donc,

Pour X= (an-K-1…a1a0**,**a-1 …a1-k a-k)B

La valeur décimale de X sera donnée par la formule :

**Exemple :**

Soit le nombre réel x codé en binaire en virgule fixe :**X = (1101 , 01)2**

La valeur décimale de x est obtenue comme suit :

X= (1\* 2-2 ) + (0\* 2-1 ) + (1\* 20 ) + (0\* 21 ) + (1\* 22 ) + (1\* 23)

Donc, X= 0.25 + 0 + 1 + 4 + 8.

D’où **X = 13.25.**

1.4.1.2. Conversion base décimale base B

Pour transformer un nombre décimal fractionnaire en base B, la partie entière est transformée selon le même principe des nombres entiers. La partie fractionnaire est obtenue par multiplication successive de la partie fractionnaire par la base B. on note dans chaque opération la partie entière résultante de la multiplication. Le processus de multiplication et répété jusqu’à disparition de la partie fractionnaire ou apparition d’une période de chiffres.

**Exemple** :

**X=7,125 on veut exprimer X en base binaire :**

* Conversion de la partie entière de X en base 2 : **7=(111)2**
* Conversion de la partie fractionnaire de X en base 2 :

0,125 x 2 = 0,25 = 0 + 0,25

0.25 x 2 = 0,5 = 0 + 0,5

0,5 x 2 = 1,0 = 1 + 0,0

**0,125 =(0,001)2**

Ainsi : **7,125 =(111,001)2**

* + - 1. **Arithmétique**

Les opérations arithmétiques en virgule fixe dans une base B, se réalisent de la même manière qu’en base 10.

***NB****:* Plus de détails seront exposées en travaux dirigés.

* + 1. Représentation des nombres Réels en virgule flottante :

La gestion de la position de la virgule fixe étant difficile, en général, on fait recours à la représentation en virgule flottante **[FLOATING POINT].**

* + - 1. **Principe**:

La représentation à virgule flottante consiste à représenter les nombres réels sous la forme suivante : **X =± M.BE** , Avec :

**B :** Base (2, 8, 10, 16, …)

**M**: Mantisse (un nombre purement fractionnaire)

**E**: Exposant (un nombre entier)

La mantisse est normalisée (elle comporte le maximum de chiffres significatifs) donc :

Soit

Exposant et mantisse doivent pouvoir représenter des nombres positifs et négatifs. Ils pourraient être codés en SVA, CP1 ou CP2, souvent M et sous forme SVA, E est sans signe mais décalé (Biaisé).

La représentation interne d’un nombre réel normalisé sur n bits aura la forme suivante :

|  |  |  |
| --- | --- | --- |
| **SM** | **E** | **M** |
| Signe de la maitrise | Exposant | Mantisse |

* + - 1. **Cas d’étude : représentation des nombre réels en virgule flottante selon la norme IEEE-754 en simple précision.**

Par souci de normalisation de représentation des nombres en vigueur flottante, plusieurs normes ont été proposées.

Dans la suite de ce cours nous abordons la norme IEEE-754 établit par le comité IEEE (Institut of Electrical and Electronics Engénners), en simple précision sur 32 bits.

***Principe :***

Dans cette norme un nombre flottant X s’écrit sous la forme, où :

* **M :** représente le pseudo mantisse car le 1 entier n’est pas représenté. La mantisse est codée sur 23 bits (correspondant aux puissances négatives de 2-1 à 2-23).
* **E :** est un exposant codé sur 8 bits, l’exposant est décalé de +127 car il sera représenté sans signe sur les 8 bits (Ed **= E + 127**)
* Un bit de signe. Selon le schéma suivant :

|  |  |  |
| --- | --- | --- |
| **S** | **Ed** | **M** |
| Signe | Exposant décalé | Mantisse |

Un (1) bit de signe, 8 bits pour l’exposant et 23 bits pour la pseudo mantisse

**Description** :

* + - * 1. Le bit le plus fort est un bit de signe, représentant le signe de la mantisse (1 pour le signe négatif et 0 pour le signe positif).
        2. L’exposant sur **8** bit est décalé de +**127**pour représenter les puissances positives et négatives. La plage de représentation de l’exposant est : d’où
* L’exposant décalé sera représenté sur 8 bits non signé, il sera dans la plage .
* Les valeurs acceptées pour **Ed** sont l’intervalle**.** Les valeurs zéro et 255 seront réservées à d’autres fins.
* De même, Les valeurs acceptées pour **E** sont : **.** Les valeurs -127 et +128 seront réservées à d’autres fins.
  + - * 1. La pseudo-mantisse est la partie fractionnaire elle occupe les 23 bits les plus faibles dans la représentation interne.

***Exemple***

Représentation du nombre **X= 25,127**en virgule flottante **IEEE-754** en simple précision :

1. Mettre X sous Forme

**25 = (11001)2  , 0.125 =(0,001)2**

**25,125 = +1,1001001 . 24**

1. La pseudo-mantisse **M=(0,1001001)2**
2. L’exposant biaisé **Ed = 4+127= 131**

**131 =(10000011)2**

1. le bit de signe = **0**

Donc sur 32 bit en norme IEEE-754

Le nombre X=25,125 sera représenté en interne par :

|  |  |  |
| --- | --- | --- |
| **0** | **10000011** | **10010010000000000000000** |
| S | Ed | M |

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque :  * La représentation interne du nombre réel peut être exprimée en hexadécimale, octal, etc… * Dans l’exemple précédant si on procède par groupement de 4 bits de droite à gauche, en suite nous remplaçons les groupements par leurs équivalents en binaire, nous obtiendrons la représentation interne hexadécimale du nombre réel : **X= 41C90000** |

***NB****:* L’arithmétique des réels en virgule flottantes ainsi que des applications plus détaillées de la norme IEEE-754 seront exposées en travaux dirigés.

Le tableau ci-dessous récapitule les plages des valeurs des nombres réels représentés en normes IEEE-754 simple précision.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signe** | **Exposant décalé** | **Exposant** | **Mantisse** | **Valeur** |
| **0** | **0** | **-127** | **0** | **Zéro** |
| **1** | **0** | **-127** | **0** | **Zéro** |
| **0** | **0** | **-127** | **≠ 0** | **Nombre dénormalisé** |
| **1** | **0** | **-127** | **≠ 0** | **Nombre dénormalisé** |
| **0** | **1 ≤ Ed ≤ 254** | **-126 ≤ Ed ≤ +127** | **Quelconque** | **Nombre normalisé** |
| **1** | **1 ≤ Ed ≤ 254** | **-126 ≤ Ed ≤ +127** | **Quelconque** | **Nombre normalisé** |
| **0** | **255** | **128** | **0** | **+∞** |
| **1** | **255** | **128** | **0** | **-∞** |
| **0** | **255** | **128** | **≠ 0** | **NaN** |
| **1** | **255** | **128** | **≠ 0** | **NaN** |

1. Chapitre2 : Codage α-numérique

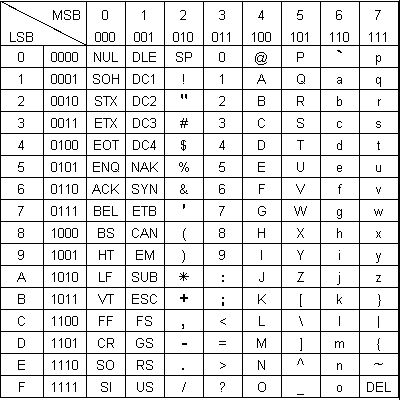
2.1. Le code ASCII :

2.1.1. Le code ASCII (07 bits) :

Le code ASCII est une abréviation de l’**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. Le code est universellement adopté pour coder et représenter l’information α-numérique.

Dans la table **ASCII**, le codage se fait sur **7 bits** et la représentation interne est réalisée sir 8 bits. Le 8ème bit (bit le plus fort) est réservé pour le système. La table ASCII est organisée en lignes et en colonnes où :

* Chaque ligne représente un état des quatre bits les plus faibles de l’octet, appelés bits de poids faible (LSB).
* Chaque colonne représente un état des trois bits les plus fort de l’octet, appelés bits de poids fort (MSB).
* Chaque case de la table représente un caractère α-numérique codé sur sept bits.



**Table ASCII -7 bits**

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque : le nombre de caractères α-numériques codés dans la table et de 128 ( 27). |

***Exemples :***

* Le code ASCII du caractère « **A »** est : 1000001 ou 41 en hexadécimale.
* Le code ASCII du caractère « **? »** est : 0111111 ou 3F en hexadécimale.
* Le code ASCII 4E représente le caractère « N ».

**2.1.2. Le code ASCII étendu (08 bits) :**

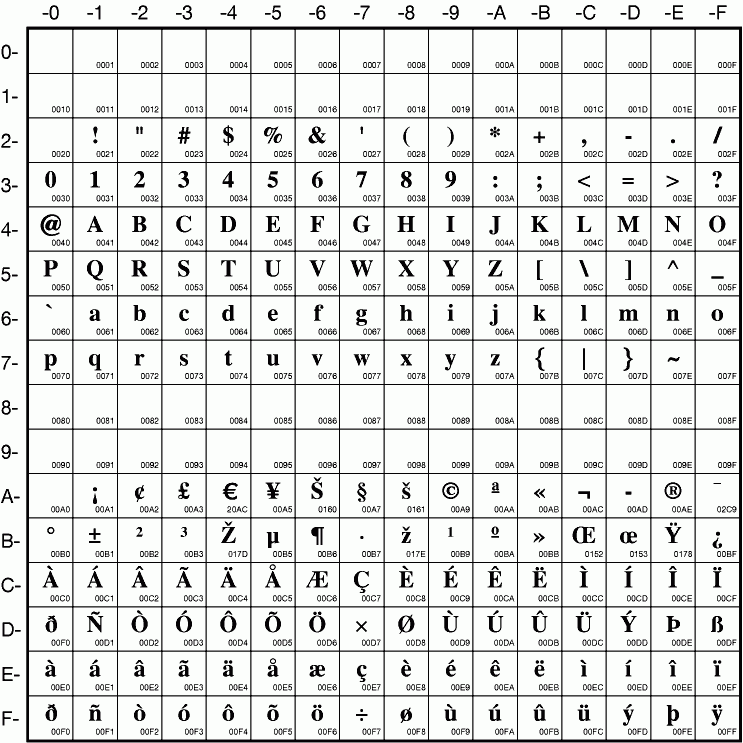
Le code ASCII a été mis au pont au départ pour la langue anglaise. Plus tard l’informatique a envahie le monde ce qui a nécessité l’introduction de codage deplusieurs autre langueset caractères spéciaux. La table du code ASCII 7 bits s’avère insuffisante pour couvrir les nouveaux codes.

Le code ASCII a été étendu à 8bits pour pouvoir coder plus de caractères.la norme ISO/CEI 8859 fournit des extensions pour le codage en incluant le 8ième bit le plus fort.

Ainsi, le code ASCII étendu utilise 8 bits pour coder les caractères.La table ASCII est organisée en lignes et en colonnes où :

* Chaque colonne représente un état des quatre bits les plus faible de l’octet, appelés bits de poids faible (LSB).
* Chaque ligne représente un état des quatre bits les plus forts de l’octet, appelés bits de poids fort (MSB).
* Chaque case de la table représente un caractère α-numérique codé sur huit bits. le nombre de caractères α-numériques codés dans la table et de 256 ( 28).

Ci-dessous un exemple de la table ASCII ISO-8859-15.



**Table ASCII ISO-8859-15.**

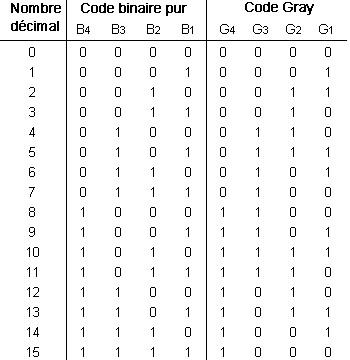
***Exemples :***

* Le code ASCII du caractère « **A »** est : 01000001ou 41 en hexadécimale.
* Le code ASCII du caractère « **? »** est : 00111111 ou 3F en hexadécimale.
* Le code ASCII 4E représente le caractère « N ».

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque :  * Le code ASCII étendu ne prend pas en charge les caractères qui ne sont pas d'alphabet latin (le chinois ou l’arabe). * L’**Unicode** tend à remplacer le code ASCII. Ce standard code chaque caractère sur 16 bits, ce qui offre 65536(216) possibilités. Cela permet de coder les différentes langues ainsi de beaucoup d’autre symboles. |

2.2. Le code Gray (binaire réfléchi)

Le **code** **Gray**, également appelé **code binaire réfléchi**, est obtenu par la modification d’un seul bit à la fois dans le code binaire d’un nombre quand il est augmenté d'une unité. Le tableau suivant représente la construction du code Gray pour les nombres de 0 à 15 :



2.2.1. Généralisation :

Soit **X** un nombre entier positif codé en binaire sur n+1 bits par : **X = Bn Bn-1…………..B1B0**

et codé en code Gray sur n+1 bits par : **X = Gn Gn-1…………..G1G0**

1. Conversion **Binaire-Gray** : le calcul du code Gray à partir d’un code binaire pur s’effectue selon la procédure suivante :

**Gn= Bn**

**Gi=Bi+1 ⊕Bi** pour i=n-1 jusqu’à i=0

1. Conversion **Gray-Binaire** : le calcul du code binaire à partir d’un code Gray s’effectue selon la procédure suivante :

**Bn = Gn**

**Bi=Gi ⊕ Gi+1 ⊕……. ⊕Gn**pour i=n-1 jusqu’à i=0

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque : le code gray est utilisé dans les capteurs numériques (angulaires ou linéaires) dans la conception des circuits logiques notamment dans les tableaux de karnaugh, etc… |

2.3. Le code BCD

Le code BCD abréviation du **B**inaire **C**odé **D**écimale, fait correspondre au chiffre de la, base 10 (de 0 à 9) l’équivalent de leurs valeurs en binaire sur quatre bits. Ainsi, chaque nombre exprimé en base 10 sera code en BCD en remplaçant chaque chiffre décimale par sonéquivalent binaire sur quatre bits.Ce code BCD est utilisé dans les systèmes numériques.

***Exemple :***

Soit le nombre X tel que X= 759,

7 = (0111)2 5 = (0111)2 9 = (0111)2

Le code BCD de X est : 011101011001

***NB*:** les détails des exercices relatifs à cette partie du chapitre seront effectués en séance de Traveaux dirigées.

1. Chapitre 3 : Algèbre de Boole
   1. Introduction :

Dans cette partie du cours nous nous intéressons aux travaux du mathématicien anglais George Boole publiés en 1854, ceux relatifs à la logique binaire appliquée à la logique des circuits.

* 1. Terminologies et opération de bases
     1. Les variable booléennes :

Une variable booléenne prend ses valeurs dans l’ensemble {0 , 1} , elle est souventreprésentée par les symboles de l’alphabet (x, y, z, etc …).

Les valeurs 0 et 1 sont des valeurs logiques : le « 0 » équivaut la valeur logique « faux » et le 1 équivaut la valeur logique « vraie».

* + 1. Les opérateurs booléens :

Les opérateurs de bases sont :

* + - 1. Le et logique noté : (\*) ou (.) Ou () :

***Exemple*:** X et y sera noté : x\*y ou x.y ou bien xy.

***Principe***: Le résultat de l’opération de x et y est vrai si et seulement si x et y sont vrais

* + - 1. Le ou logique noté : (+) :

***Exemple* :** X ou y sera noté : x+y .

***Principe***: Le résultat de l’opération de x et y est faux si et seulement si x et y sont faux

* + - 1. La négation : ( ) :

***Exemple*:** la négation de X seranotée : X .

***Principe* :** Le résultat de l’opération la négationX et la valeur booléenne inverse de X

* + 1. Expression booléenne

Une expression booléenne est formée à partir de variables booléennes connectées par des opérateurs booléens. L’expression booléenne retourne une valeur booléenne lors de son évaluation.

***Exemple :*** (X+Y).(X+Z) est une expression booléenne.

* + 1. Evaluation d’une expression booléenne

L’évaluation d’une expression booléenne respecte l’ordre de précédences des opérateurs booléens.

L’ordre de précédence des opérateurs booléens est défini du plus faible au plus fort comme suit :

**OU ET NON ()**

**Faible Fort**

***Exemple* :** soit l’expression booléenne suivant (X + Y).( X + Z + Y).

L’évaluation de cette expression pour les valeurs respective 0, 0 et 1 de X, Y et Z est la suivante :

**(X + Y) . ( X + Z + Y).**

**1 2 3 4**

**5**

**Le résultat de l’évaluation est : 0**

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque : Lorsque deux opérateurs de la même priorité se succèdent, la priorité revient à celui de gauche. |

* + 1. La table de vérité

La table de vérité est un outil mathématique permettant de représenter les expressions booléennes, en utilisant les différentes combinaisons des variables constituants cette expressions.

La table de vérité est un ensemble de colonnes, chaque variable dans l’expression occupe une colonne et l’expression elle-même occupe une colonne.

Dans chaque ligne de la table en attribue la valeur de l’expression booléenne pour les valeurs inscrit de ses variables.

Le nombre de lignes de la table de vérité est égale à 2n, tel que n représente le nombre de variables dans la table.

***Exemple*** : soit l’expression booléenne x.y+z, la table de vérité associée à cette expression est donnée par la figure ci-dessous :

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **X.Y+Z** |
| **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** | **1** |
| **1** | **1** | **1** | **1** |

**Table de vérité**

* 1. Les fonctions booléennes
     1. Définition :

Une fonction booléenne est formée d’un nom pour la fonction, du symbole d’égalité et d’ une expression booléenne formée par les variables de la fonction.

***Exemple* : F(X,Y,Z) = X.Y+Z** est une fonction à trois variables X, Y et Z.

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque :  * Une fonction booléenne est évaluée au même titre qu’une expression booléenne. * Les fonctions booléennes prennent des valeurs booléennes lors de leurs évaluations |

* + 1. Représentations des fonctions booléennes :

Une fonction booléenne peut être représentée par une table de vérité telle qu’une expression booléenne, ou par une expression algébrique selon deux formes standards.

* + - 1. Représentation des fonctions booléennes sous forme de SOP (Somme de Produit)

***Exemple* :**

Soit la fonction booléenne F(X, Y, Z) = X Y + X Y Z.

* La fonction booléenne regroupe deux termes (X Y) et (X Y Z), dans chaque terme les variables sont connectées par l’opérateur (et) (produit).
* Les deux termes de la fonction sont connectés par l’operateur (ou) (somme).
* Les termes de la fonction sont alors en sommes de produits Sop.
  + - 1. Représentation des fonctions booléennes sous forme de POS (Produit de Somme)

***Exemple :***

Soit la fonction booléenne F(X, Y, Z) = (X+ Y). (X +Y+ Z).

* La fonction booléenne regroupe deux termes (X+ Y) et (X+ Y+ Z), dans chaque terme les variables sont connectées par l’opérateur (ou) (somme).
* Les deux termes de la fonction sont connectés par l’operateur (et) (produit).
* Les termes de la fonction sont alors en produit de somme Pos.
  + - 1. Représentation des fonctions booléennes sous forme canonique

***Exemple :***

Soit la fonction booléenne F(X, Y, Z) = XY+ XYZ.

* La fonction booléenne F regroupe deux termes XY et XYZ.
* Le terme XY ne contient pas la variable Z, par conséquent il est dit non canonique.

***Définition :*** *une fonction est dite sous forme canonique Sop ou Pos si et seulement si dans chaque terme de la fonction figure toute les variables de la fonction.*

***Exemple :*** soit la fonction booléenne  , F est sous forme de Sop canonique.

* + 1. Procédures d’extractions des formes Sop et Pos d’une fonction booléenne en utilisant une table de vérité
       1. Principe :

Dans une table de vérité chaque ligne regroupe un code binaire pour les viables de la fonction représentée, ainsi que le code binaire de la fonction elle-même.

* Pour retrouver une valeur de 1 de la fonction dans une ligne de la table de vérité en construit un terme en conjonction des variables de la fonction tel que chaque variable apparait non complémentée si sa valeur est 1 et complémentée si sa valeur est 0. Le terme constitué est appelé **« min-terme »**
* Pour retrouver une valeur de 0 de la fonction dans une ligne de la table de vérité en construit un terme en disjonction des variables de la fonction tel que chaque variable apparait non complémentée si sa valeur est 0 et complémentée si sa valeur est 1. Le terme constitué est appelé **« Max-terme »**
  + - 1. Procédure permettant d’obtenir la forme Sop.

Pour obtenir la forme Sop canonique d’une fonction booléenne, on construit une disjonction de tous les min-termes de la fonction extraits de la table de vérité.

***Exemple :***

Soit la foction booleene F definie par la table de verité suivante :

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **X.Y+Z** |
| **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** | **1** |
| **1** | **1** | **1** | **1** |

* La fonction regroupe cinque min –termes.
* Pour la ligne 2 de la table de verité F(0,0,1)=1 , le min-terme associé est :
* F sous forme de Sop :

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque : Pour simplifier l’utilisation de la forme canonique Sop, on utilise un format compact appelé ***Sop Numérique***  Le format Sop numérique est obtenu par remplacement de chaque min-terme par la valeur décimale du code binaire des variables de la fonction qui génère ce min-terme.  Dans l’exemple précédant,   * Le terme sera remplacé par la valeur 1 (001 en base 2 = 1 en base10), * Le terme sera remplacé par la valeur 6 (110 en base 2 = 6 en base10),   Donc : |

* + - 1. Procédure permettant d’obtenir la forme Pos.

Pour obtenir la forme Pos canonique d’une fonction booléenne, en construit une conjonction de tous les Max-termes de la fonction extraits la table de vérité.

***Exemple :***

Soit la foction booleene F definie par la table de verité suivante :

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **X.Y+Z** |
| **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** | **1** |
| **1** | **1** | **1** | **1** |

* La fonction regroupe Trois Max–termes.
* Pour la ligne 3 de la table de verité F(0,1,0)=0, le Max-terme associé est :( X + Y+ Z )
* F sous forme de Pos :

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque : Pour simplifier l’utilisation de la forme canonique Pos, on utilise un format compact appelé ***Pos Numérique***  Le format Pos numérique est obtenu par remplacement de chaque Max-terme par la valeur décimale du code binaire des variables de la fonction qui génère ce Max-terme.  Dans l’exemple précédant,  Le terme sera remplacé par la valeur 4 (100 en base 2 = 4 en base10),  Donc : |

* + 1. Fonction non complètement définie :

Les fonctions vues précédemment sont des fonctions complètement définies, c-à-d, à chaque combinaison des variables de la fonction correspond une valeur (0 ou 1) pour la fonction.

Une fonction est dite non complètement définie s’il existe au moins une combinaison de variables de la fonction pour laquelle la valeur de la fonction booléenne est indéterminée.

***Exemple :***

On veut trier des objets selon le poids P de la manière suivante :

Les objets qui seront acceptés doivent avoir un poids 400g ≤ P ≤ 500g et soit F la fonction qui permet d’accepter ou de rejeter l’objet.

|  |  |  |
| --- | --- | --- |
| **X** | **Y** | ***F(x,y)*** |
| **0** | **0** | **0** |
| **0** | **1** | **?** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

*F(x,y)* =1 si 400g ≤ P ≤ 500g

*F*(0,1) est non définie car et *P*≥500g en même temps.

***N.B :*** La valeur non définie d’une fonction booléenne est représentée soit par « d », « X » ou bien «­ ».

* 1. Algèbre de Boole : Théorèmes et postulats
     1. Définition :

Une algèbre de Boole est un système algébrique fermé, contenant un ensemble K de deux ou plusieurs éléments et deux opérateurs binaires « + » (ou) et « . » (et) tel que :

En plus, les postulats suivants doivent être satisfaits :

***P1 : Existence de 1 et 0 :***

***P2 : Commutativité :***

***P3 : Associativité :***

***P4 : Distributivité :***

***P5 : Complémentation :***

est le complément de *x*.

***N.B :*** Deux expressions sont dites équivalentes si l’une peut remplacer l’autre.

* + 1. Dualité :

Le dual d’une expression est obtenu en remplaçant chaque « + » dans l’expression par « . », chaque « . » par « + », chaque 1 par 0 et chaque 0 par 1.

***Exemple :***

Dual :

***N.B:*** Dans la définition de l’algébre de Boole (postulats), pour chaque postulat le (b) est toujours le dual de (a).

* + 1. Théorèmes :

***T1 : Idempotence :***

***T2 : Propriété de 1 et 0 :***

***T3 : Absorption :***

***T4 : Demi-Absorption :***

***T5 : Loi de Demorgan :***

***T6 : Consensus :***

* 1. Simplification des fonctions booléennes :
     1. Pourquoi la simplification :

Dans l’algèbre classique, la simplification signifie la recherche d’une forme d’écriture d’une fonction assez dense et facilement évaluée.

***Exemple :***

Dans l’algèbre de Boole, la simplification signifie aussi la recherche d’une forme d’écriture d’une fonction assez dense et qui demande un nombre réduit d’opérateurs logiques (et, ou, not, …), afin que la réalisation matérielle (circuit) ne nécessite pas un nombre important de composants.

* + 1. Méthodes de simplification :
       1. Simplification algébrique :

La simplification algébrique d’une fonction booléenne nécessite l’application des théorèmes de l’algèbre de Boole.

***Exemple :***

(Factorisation)

(Complémentation)

(Existence de 1)

* + - 1. Simplification par la méthode de Karnaugh :

La méthode de Karnaugh offre une procédure simple de minimisation des fonctions booléennes. Elle est connue sous l’appellation de Table de Karnaugh ou diagramme de VEITCH.

1. ***Description de la table de Karnaugh :***

La table est un diagramme formé de carrés. Chaque carré représente un min-terme marqué par « 1 » (les « 0 » de la table représentent les max-termes) de la fonction booléenne.

Pour indicer la table, chaque variable remplit un ensemble de cases adjacentes. Un produit de deux variables se traduit par l’intersection des cases affectées à chacune des variables. Aux N cases de Karnaugh correspondent les N lignes de la table de vérité représentant la fonction booléenne.

***Exemple 1 :*** Table de Karnaugh à deux variables

*Table de vérité :*

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **F** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

*Table de Karnaugh :*

|  |  |  |
| --- | --- | --- |
| b  a | 0 () | 1 () |
| 0 () |  | 1 |
| 1 () | 1 | 1 |

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque : Deux cases de la table de Karnaugh sont dites adjacentes si et seulement si les deux min-termes des deux cases diffèrent d’une seule variable. |

***Exemple 2*** : Table de Karnaugh à trois variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *bc*  *a* | 00 | **01** | **10** | 11 |
| 0 |  |  |  |  |
| 1 |  |  |  |  |

Deux cases non adjacentes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *bc*  *a* | 00 | **01** | **11** | 10 |
| 0 |  |  |  |  |
| 1 |  |  |  |  |

Toutes les cases sont adjacentes

***N.B :*** La table de Karnaugh est considérée comme un cylindre fermé horizontalement et verticalement.

1. ***Procédure de simplification :***

*Etape 1 :* Passage de la table de vérité (ou directement de l’expression de la fonction sous forme de Sop) en plaçant des 1 dans les carrés correspondants aux min-termes de la fonction.

*Etape 2 :* Grouper tous les carrés ADJACENTS contenant des 1 de manière maximale, c-à-d former les groupements des carrés ADJACENTS de taille maximale égale à une puissance de deux décroissante

*Etape 3 :* Former un minimum de groupements de 1 afin d’obtenir une fonction simplifiée sans termes redondant appelée : Base minimale non redondante pour la fonction booléenne où chaque groupement est appelé « monôme premier ».

* + - 1. Exemples :

***Exemple 1 :***

Soit la fonction booléenne F :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *bc*  *a* | 00 | 01 | 11 | 10 |
| 0 | 0  1 | 1 | 3 | 2 |
| 1 | 4  1 | 5  1 | 7  1 | 6 |

(Base complète minimale non redondante pour *F*)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *cd*  *ab* | 00 | 01 | 11 | 10 |
| 00 | 0  1 | 1 | 3 | 2  1 |
| 01 | 4  1 | 5  1 | 7  1 | 6  1 |
| 11 | 12  1 | 13  1 | 15  1 | 14  1 |
| 10 | 8 | 9  1 | 11 | 10 |

(Base complète minimale non redondante pour *F*)

|  |
| --- |
| Résultat de recherche d'images pour "icon nota Bene." Remarque : La table de Karnaugh est utile pour simplifier les fonctions booléennes telles que le nombre de variables logiques soit ≤ 6. |

**Conclusion**

Les méthodes des simplifications booléennes servent à réécrire les fonctions dans un format plus compacte dans l’objectif de réaliser les circuits combinatoires associés à moindre cout.

Dans ce chapitre sont exposées deux méthodes de simplification (simplification algébrique et la méthode de karnaugh) nous comptons enrichir le chapitre par la présentation de la méthode de Quine McClusky incessamment.

Annexe1 :

Série1 des travaux dirigés (Codification et représentation des nombres)

**Exercice 1 :**

1- Exprimer les nombres suivant en base Décimale :

(472)8 (3132)4  (560)7(101101)2 (1A8DF)16

2- Exprimer le nombre décimal X= 327 en base 2, 3, 7, 8, et 16.

3- Faire la conversion Binaire/Décimale des nombres suivants :

101, 11101, 111101101, 11111111.

**Exercice 2 :**

1. Soit le nombre décimal X= 4a5  + 2a3 + a + 5 tel que a est un entier (a>5).

Exprimer X en base a

1. exprimer les nombres décimaux X, Y, Z en base a (a est un entier/ a>1)

X= a, Y= a2, Z=a3

**Exercice 3 :**

Déterminer les couples des entiers (x,y) tel que :

(x y) 7 = (y x)10

**Exercice 4 :**

1. Soit le nombre décimal X= 512, exprimer X en base 2, 4, 8 et 16.
2. Soit le nombre Y = ( 11010110101 )2

Exprimer directement et sans passer par la base 10 le nombre Y en base 4, 8, 16.

1. Exprimer directement en base 2 et sans passer par la procédure de division les nombres : X = (1323)4 , Y= (3765)8 , Z= (AB1F9)16

**Exercice 5 :**

1. Faire l’addition des nombres binaires suivants : X=(101101)2  et Y= (110110)2
2. Réaliser en base 8 l’addition des deux nombres X=(735)8 et Y=(132)8
3. Réaliser en hexadécimale l’addition des deux nombres X=(A1F)16 et Y=(9BC)16

**Exercice 6:**

Soit les nombres entiers X et Y tel que : X= 18 et Y= 30.

1. Exprimer sur six bits en SVA, CP1 et CP2 les nombres entiers X, -X, Y et -Y.

Est-ce que cette codification est possible sur Cinque bits ? Justifier.

1. Réaliser, si possible, les opérations suivantes en SVA, CP1 et CP2 :

X-Y, Y-X, -X-Y.

**Exercice 7:**

Soit les nombres entiers : X= (1101011) en CP1 , Y=(11001100) en CP2,

Z= (0100111) en CP1 et T= (0101101) en CP2.

Donner le signe et la valeur décimale de chaque nombre.

**Exercice 8 :**

1. Exprimer en binaire les nombres réels suivants : 112,125 ; 237,25 ; 128,75
2. Exprimer en décimal les nombres réels suivants : (111,01101)2 ; (101,10101)2
3. Réaliser les opérations arithmétiques suivantes : 112,125 + 237,25 et (111,01101)2+(101,10101)2

**Exercice 9 :**

1. Donner selon la norme IEEE-754 le code des nombres réels suivants :

X= 27,25  ; Y= -13,5  ; Z= 0,375

1. X est un nombre réel codé selon la norme IEEE-754, écrire X sous forme réelle.

X=(11011000011010110000000000000000)

**Exercice 10 :**

Soient les entiers positifs A, B et C tels que : A = 109 ; B = 18 et C = 36.

1. Exprimer A, B et C en base 2.
2. Réaliser les opérations arithmétiques : A+B ; A+C et B+C en SVA, CP1 et

CP2. Préciser pour chaque opération le nombre de bits nécessaire.

1. Réaliser les opérations A-B-C ; B-A-C en SVA, CP1 et CP2 sur 8 bits si possible.

**Exercice 11 :**

1. Soit le nombre réel X= 4015, 9375 *.*
2. Exprimer X en base 16, ensuite déduire X en base 2 et en base 8*.*
3. Soit le nombre réel Y = (50,1) 16

b-1 Calculer en base 16 le nombre réel Z= X+Y, déduire Z en base 10

ensuite en base2.

b-2 Exprimer les nombres  et en base 2 ensuite en base 10.

b-3 Donner la plus grande valeur de l’entier naturel n pour que le nombre

entier soit strictement supérieur à 1.

1. Soit le nombre réel A = (FAF,F) 16
2. Exprimer A sous la forme A= ±1,M. 2E
3. Donner la représentation interne en binaire de A selon la norme IEEE-754 en simple précision (sur 32 bits).
4. Donner la représentation interne en hexadécimale de A.
5. Déduire la représentation interne du nombre B = -A en hexadécimale.

Série supplémentaire des travaux dirigés

**interrogation ECRITE (2014-2015)**

**Exercice (15 Pts)**

1. Soit les nombres entiers X = (137) 8 et Y = (255) 6
2. Exprimer X et Y en base hexadécimale (16). *(02 pts)*
3. Réaliser les opérations arithmétiques X + Y et X\*Y en base hexadécimale.*(02,5 pts)*

2 -Soit les nombres entiers a et n tel que : a= n2+ 1 et n > 1.

Exprimer les nombres suivants en base a : A= n2+ 2, B= (n2+ 2)2 et

C = n (n2+ 2). *(03 pts)*

1. Soit le nombre réel X = 61/8.

Exprimer X Sous la forme X= (A, B) 2 ensuite X= (A, B) 16 (*02 pts)*

Telque : A représente la partie entière et B représente la partie fractionnaire.

1. Soit les nombres entiers X= 15 et y = - 32.
2. Coder si possible X et Y en SVA, CP1 et CP2 sur 6 bits *(03 pts)*
3. Réaliser si possible sur 6 bits les opérations : X+Y et X-Y en CP1 et en CP2. *(02,5 pts)*

**interrogation Ecrite (2017- 2018)**

**Exercice 1 : (09 Pts)**

1. Soit le nombre réel X= 2988*.*
2. Exprimer X en base 16*. (1 pt)*
3. Soit le nombre réel Y = (0, BAC) 16

b-1 Exprimer en base 10 le nombre réel Y. *(1 pt)*

b-2 déduire en base 2 le nombre réel Z= X+Y. *(1 pt)* 2- Exprimer Z sous la forme Z= ±1, M. 2E *(1 pt)*

1. Donner la représentation interne en binaire de Z selon la norme IEEE-754 en simple précision (sur 32 bits). *(2 pts)*
2. Déduire la représentation interne de Z en hexadécimale. *(1 pt)*
3. Soit T= Z. 2117

c.1- Déduire la représentation interne du nombre T en hexadécimale.*(1 pt)*

c.2- Que représente le nombre T dans la norme IEEE-754 en simple précision.

*(1 pt)*

**Exercice 2 : (06 Pts)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| SVA | 11001101 |  |  |
| CP1 |  | 01011011 |  |
| CP2 |  |  | 11000011 |

1. Compléter le tableau ci-dessus. (1,5 pts)
2. Déduire les nombres réels A, B, C. (1,5 pts)
3. Réaliser, si possible, les opérations arithmétiques suivantes sur 8 bits :
   1. A+B en SVA. (1 pt)
   2. B+C en CP1. (1 pt)
   3. A+C en CP2. (1 pt)

NB : Toutes les réponses doivent être justifiées.

**CONTROLE (2014-2015**)

**Exercice1 :**

Soit les nombres réels X,Y, et Z tel que : X = 79,25, Y= 170 et Z = (46)16

1. Calculer en base 16 le nombre T tel que : T= Y- X (1,5 Pts)
2. Représenter en SVA, CP1 et CP2 les nombre suivants :

(Z-Y), 2(Z-Y) est 16(Z-Y) (utiliser le nombre de bits nécessaire) (3 Pts)

1. Nous considérons une machine dans laquelle les nombres réels sont représentés en virgule flottante sur 24 bits selon le format X= ± 1, M. 2E, et tel que :

* Le bit le plus fort est réservé pour le signe (1 pour le signe négatif).
* Les 7 bits suivants représentent l’exposant E décalé avec la valeur (+ 63).
* Les derniers 16 bits sont réservés pour la mantisse M.

|  |  |  |
| --- | --- | --- |
| *1 bit pour le signe* | 1. *bits pour l’exposant* | 1. *bits pour la mantisse* |

1. Donner la représentation interne du nombre réel T selon la norme proposée. (03 pts)
2. Donner la représentation interne de T en Octale (base 8). (01 pts)
3. Donner la représentation interne en hexadécimale pour (- ∞) (01,5 pts)

**CONTROLE (2016-2017**)

**Exercice : (05 Pts)**

Soient les nombres entiers positifs : A, B et C tels que :

A = 109 ; B = 18 et C = 36.

1. Exprimer A, B et C en base 2.  *(01.5 pt)*
2. Réaliser les opérations arithmétiques : A+B ; A+C et B+C en SVA, CP1 et CP2. Préciser pour chaque opération le nombre de bits nécessaire. *(01.5 pt)*
3. Réaliser les opérations A-B-C ; B-A-C en SVA, CP1 et CP2 sur 8 bits si possible.

*(02 pt)*