Support du cours : Systèmes Logiques

Elaborée par: Lamia Tounsi

Lamia.tounsi@fss.usf.tn

Pour la filière: 1ère année PISI

2023-2024

Chapitre 1:Systèmes de Numération et codes

1. Généralités
2. Système décimal
3. Système binaire , octal et hexadécimal
4. Conversion d’un système de numération vers un autre système
5. Représentation des nombres signés
6. Opérations arithmétiques

1.Généralités

- On utilise les " systèmes de numération" pour compter des objets et de les représenter par des nombres
- Un système de numération se définie par deux éléments:
  - La base du système
  - Les symboles du système
- Les systèmes les plus utilisés sont

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

3

1.Généralités

- **Représentation des nombres positifs**

Un nombre **N** sera présenté dans une base **b** comme suit:

b: base du système du numération

- **a_i**: symbole du système du numération ou chiffre de rang i avec **$a_i<b$**

b^i: pondération associée à **a_i**

➔ toute base **b** est composée de **b** chiffres (digits ou symboles) tous différents

Exemples:

$N_1=(2019)_{10}$

$N_2=(10110100)_2$

2.Le système décimal

- Système **décimal**➔ car il a toujours été naturel de compter sur ses doigts
- Comporte **dix** symboles différents:**{ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 }**

Exemple

$\boxed{1\,9\,3\,8\,5\,6\boxed{2}}$

Poids fort

(Most significant bit)
MSB

Poids faible

(least significant bit)
LSB

$\boxed{345\,,\,567}$

Partie entière

Partie fractionnelle

2.Le système décimal

- **Développement en polynôme** d'un nombre dans le système décimal:

$$1968 = 1000+900+60+8 =$$

$$1x1000+9x100+6x10+8x1$$

$$=1x10^3+ 9x10^2+ 6x10^1+8x10^0$$

3.Le système binaire, octal et hexadécimal

3.3 Le système binaire (base 2)

Utilise deux symboles appelés **BIT** (**B**inary dig**IT**) : **0** et **1**

Cette base est très commode pour distinguer les **2 états logiques fondamentaux**

Dans les domaines de l'automatisme, de l'électronique et de l'informatique, nous utilisons la base 2

Exemple:

$$\mathbf{(1011)_2 = 1x2^0 + 1x2^1 + 0x2^2 + 1x2^3 = 1 + 2 + 8 = (11)_{10}}$$

Un nombre à **n** chiffres en base 2 distingue $\mathbf{2^n}$ états ou combinaisons

Les puissances successives de 2 (1, 2, 4, 8, 16, 32, …) sont appelées **poids binaires.**

3.2 Le système octal (base 8)

Dans ce système, la base vaut 8 et il y a **8 digits**: 0,1,2,3,4,5,6 et 7

Exemple:

$$(365)_8 = 5x8^0 + 6x8^1 + 3x8^2 = 5 + 48 + 192 = (245)_{10}$$

3.3 Le système hexadécimal (base 16)

Dans ce système il y a **16 digits**:

Exemple:

$$(BAC)_{16} = Cx16^0 + Ax16^1 + Bx16^2$$
$$=12 + 10x16 + 11x256$$
$$=12 + 160 + 2816$$
$$=(2988)_{10}$$

Décimal	Hexadécimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Remarque: Généralement dans le système Hexadécimal, on utilise le pos fixe h, H ou le le préfixe $, ou encore en C/C++, le préfixe 0x: **7CFh ou 7CFH ou $7CF ou 0x7CF**

4. Conversion d'un système de numération vers un autre

4.1 Conversion d'une base X à la base 10

Il faut faire le développement en polynôme du nombre dans la base **X** , et de faire la somme par la suite

Exemples:

- **$(1011)_2 = 1x2^0 + 1x2^1 + 0x2^2 + 1x2^3 = 1 + 2 + 8 = (11)_{10}$**
- **$(365)_8 = 5x8^0 + 6x8^1 + 3x8^2 = 5 + 48 + 192 = (245)_{10}$**
- **$(BAC)_{16} = 12x16^0 + 10x16^1 + 11x16^2 = (2988)_{10}$**

4.2 Conversion de la base 10 à la base X

- **Base 10 vers base 2**

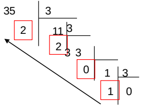
1ère méthode par divisions successives: On divise le nombre en base 10 par 2, puis, on divise successivement le quotient de chaque division par 2 jusqu'à ne plus pouvoir diviser par 2

Le nombre binaire s'obtient en relevant le reste de chaque division en partant de la dernière division vers la première (sens de lecture vers le haut)

Base 10 vers base X

La conversion se fait en prenant les restes des divisions successives sur la base X dans le sens inverse

Exemple: $35 = (?)_3$



$35=(1022)_3$

Question : Effectuer les transformations suivantes :

$(43)_{10}=(?)_2=(?)_5=(?)_8=(?)_{16}$

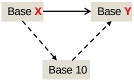
Réponse : $(43)_{10}= (101011)_2=(133)_5=(53)_8=(2B)_{16}$

12

4.3 Conversion d'une base X à une base Y

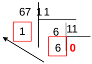
Il n'existe pas de méthode pour passer d'une base **X** à une autre base **Y** directement

L'idée est de convertir le nombre de la base **X** à la base 10 , ensuite convertir le résultat de la base 10 à la base **Y**



Exemple : $(74)_9=(?)_{11}$

$(74)_9 = 7 \times 9 + 4 = (67)_{10}$



$(67)_{10}=(61)_{11}$

➔ $(74)_9=(61)_{11}$

Conversion octal ➔ binaire

En octal chaque, symbole de la base s'écrit sur 3 bits en binaire. Remplacer chaque symbole dans la base octal par sa valeur en binaire sur 3 bits (faire des éclatements sur 3 bits).

Exemples :

$(745)_8=(\underline{111\ 100\ 101})_2$

Conversion binaire ➔ octal

Faire des regroupements de 3 bits puis remplacer chaque regroupement par la valeur octale correspondante

Exemples :

$(11001010110)_2=(\underline{011\ 001\ 010\ 110})_2=(3126)_8$

$(110010100,10101)_2=(\underline{110\ 010\ 100}\ ,\ \underline{101\ 010})_2=(624,52)_8$

Remarque : le regroupement se fait de droite à gauche pour la partie entière et de gauche à droite pour la partie fractionnelle.

Conversion hexadécimal ➔ binaire

En hexadécimal chaque, symbole de la base s'écrit sur 4 bits en binaire.
Remplacer chaque symbole dans la base hexadécimale par sa valeur en binaire sur 4 bits (faire des éclatements sur 4 bits).

Exemples :

$(B49)_{16}=(\underline{1011}\ \underline{0100}\ \underline{1001})_2$ $(1A,2F)_{16}=(\underline{0001}\ \underline{1010}\ ,\ \underline{0010}\ \underline{1111})_2$

Conversion binaire ➔ hexadécimal

Faire des regroupements de 4 bits puis remplacer chaque regroupement par la valeur hexadécimale correspondante

Exemples :

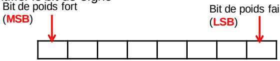
$(11011011110)_2=(\underline{0110\ 1101\ 1110})_2=(6DE)_{16}$

$(11000100,10101)_2=(\ \underline{1100\ 0100}\ ,\ \underline{1010\ 1000})_2=(C4,A8)_{16}$

Remarque : le regroupement se fait de droite à gauche pour la partie entière et de gauche à droite pour la partie fractionnelle.

5. Représentation des nombres signés

La plupart des dispositifs numériques traitent également les nombres négatifs. Le signe (+) ou (-) est identifié par un bit, dit le bit de signe et le nombre ainsi formé est dit signé. D'où donc l'importance de fixer le format de représentation des nombres pour identifier le bit de signe



Dans une base **b** et pour un format donné le MSB= bit de signe

- Si N=

b-1							

➔ N = nombre **négatif**

- Si N=

0							

➔ N = nombre **positif**

16

5.1 Complément à b-1 et Complément à b

On appelle **complément à b-1** d'un nombre N un autre nombre N_{Cb-1} tel que:

$$N+N_{Cb-1}=b^n-1$$

n : est le nombre de bits de la représentation du nombre N dans la base b

Le **complément à b** d'un nombre N noté N_{Cb} est donné par:

$$N_{Cb}=N_{Cb-1} + 1$$

Rajouter au bit LSB

Exemple

s: En binaire dans un format de 8 bits: $(12)_{10}=(00001100)_2$

le complément à 1 = $(11110011)_{C1}$

le complément à 2 = $(11110100)_{C2}$

- En octal dans un format de 2 bits: $(29)_{10}=(35)_8$

le complément à 7 = $(42)_{C7}$

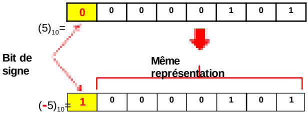
le complément à 8 = $(43)_{C8}$

Il y a trois principales façons de représenter les nombres négatifs:

5.2 La représentation en valeur absolue et signe

On adopte la même représentation pour un nombre positif et le nombre négatif correspondant. Seul le bit de signe change.

Exemple: représentation de 5 et -5 en binaire dans un format de 8 bits:



Donc dans un format de n bits en binaire:

➔ Le bit MSB=bit de signe (0 ou 1)

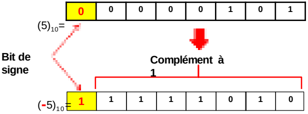
➔Valeur absolue sur $(n-1)$ bits

➔Intervalle de valeurs représentées : $[-2^{n-1}+1, 2^{n-1}-1]$

5.3 La représentation en complément à b-1

- Même principe pour le signe: **MSB**=bit de signe
- La représentation d'un nombre **positif** est i**dentique** à la représentation en valeur absolue et signe
- La représentation d'un nombre **négatif** correspond au **complément à b- 1** de son nombre positif correspondant

Exemple: représentation de 5 et -5 en binaire dans un format de 8 bits:



19

5.4 La représentation en complément à b

- Même principe que la représentation en complément à b-1 sauf que la représentation d'un nombre **négatif** correspond au **complément à b** de son nombre positif correspondant

Exemple: représentation de 5 et -5 en binaire dans un format de 8 bits:

$(5)_{10}=$	0	0	0	0	0	1	0	1	
	1	1	1	1	1	0	1	0	C1
+								1	
$(-5)_{10}=$	1	1	1	1	1	0	1	1	

Remarque:

- La représentation en complément à 2 est la représentation la plus utilisée pour la représentation des nombres négatifs dans la machine
- Utilisée pour les opérations arithmétiques
- Intervalle de valeurs représentées : **$[-2^{n-1}, 2^{n-1}-1]$**
- Le complément à 2 du complément à 2 d'un nombre est le nombre lui même