COURS: Logique Combinatoire et Séquentielle

Youssou FAYE

Université Assane Seck de Ziguinchor

7 juillet 2015

Chapitre 1 : Systèmes de Numération et Représentation des Nombres

1. Les Systèmes de Numération et de Codage
2. Arithmétique dans un Système Positionnel
3. Représentation des Nombres

Partie 1

LES SYSTEMES DE NUMERATION ET DE CODAGE

Les Systèmes de Numération et de Codage

Système de numération

- Défini comme un ensemble de règles permettant de représenter le nombre états d'un système
  - Il est composé d'un alphabet muni d'un certain nombre d'opérateurs permettant de lier les éléments de l'alphabet
  - Dans un Système de numération positionnel, la valeur du chiffre dépend de sa position dans la représentation du nombre ;
  - Exemple : Système de numération décimal est positionnel
  - Système de numération romain est non positionnel

Numération de position

- Mathématiquement, la valeur d'un nombre N est représentée sous forme d'un plolynôme par n chiffres dans la base b.
  - N=$a_{n-1}b^{n-1}+a_{n-2}b^{n-2}+.....a_1b^1+a_0b^0$
  - Exemple en base 10 (décimale) : $3254 = 3.10^3 + 2.10^2 + 5.10^1 + 4.10^0$
  - Un décalage à gauche multiplie un nombre par sa base
  - Un décalage à droite divise un nombre par sa base

Numération binaire

- L'alphabet est composé de deux symbôles {0, 1} appelés éléments binaires ou bit pour Binary digIT
- La base est 2, le système est pondéré par 2, c'est à dire les poids sont des puissances de 2
- L'addition et la multiplication sont les opérations de base
- Exemple de représentation d'un nombre en binaire
  - 10010_2 où le 2 en indice indique la base binaire
  - $10010_2 = 1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 0x2^0$
  - Le bit le plus significatif, le bit le plus à gauche est appelé *bit de poids fort* ou MSB (Most Significant Bit).
  - Le bit le moins significatif, le bit le plus à droite est appelé bit de poids faible ou LSB (Less Significant Bit)
- Si on utilise n bits, on peut représenter 2^n valeurs différentes, de 0 à 2^{n-1}
- Exemple pour N=8 : 00000000 à 11111111

Numération en base 5

- L'alphabet est composé de 5 symbôles {0, 1, 2, 3, 4}
- La base est 5, le système est pondéré par 5, c'est à dire les poids sont des puissances de 5
- L'addition et la multiplication sont les opérations de base
- Exemple de représentation d'un nombre en base 5
  - 13042_5 où le 5 en indice indique la base
  - $13042_5 = 1x5^4 + 3x5^3 + 5x5^2 + 7x5^1 + 2x5^0$

Numération Octale

- L'alphabet est composé de 8 symbôles {0, 1, 2, 3, 4, 5, 6, 7}
- La base est 8 ou base octale, le système est pondéré par 8, c'est à dire les poids sont des puissances de 8
- L'addition et la multiplication sont les opérations de base
- Exemple de représentation d'un nombre en octal
  - 13762_8 où le 8 en indice indique la base
  - $13762_8 = (1x8^4 + 3x8^3 + 7x8^2 + 6x8^1 + 2x8^0)_{10}$
- L'intérêt de ce système est que la base 8 est une puissance de 2 ($8 = 2^3$), donc les poids sont aussi des puissances de 2.
- Chaque symbole de la base octale peut être exprimé sur 3 éléments binaires

Numération hexadécmale

- L'alphabet est composé de 16 symbôles {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
- La base est 16, le système est pondéré par 16, c'est à dire les poids sont des puissances de 16
- L'addition et la multiplication sont les opérations de base hexadécmale
- Exemple de représentation d'un nombre en hexadécmale
  - $1A57F_{16}$ où le 16 en indice indique la base hexadécmale
  - $1A57F_{16} = (1x16^4 + 10x16^3 + 5x16^2 + 7x16^1 + 15x16^0)_{10}$
- L'intérêt de ce système est que la base 16 est une puissance de 2 (16 $=2^4$), donc les poids sont aussi des puissances de 2.
- Chaque symbole de la base hexadécmale peut être exprimé sur 4 éléments binaires

Conversion d'un système de numération à un autre (1)

- Base B vers la base 10
  - $(a_n.....a_0)_B = a_n.B^n + ... + a0.B^0 = (a'_m....a'_0)_{10}$
  - Exemple
    - $(1001)_2 = 1.2^3 + 0.2^2 + 0.2^1 + 1.2^0 = 9_{10}$
    - $(A12)_{16} = 10.16^2 + 1.16^1 + 2.16^0$
- Base 10 vers base B
  - Méthode à soustractions successives
  - Elle consiste à soustraire successivement la plus grande puissance de B multiplié par un élément de la base. on note l'élément de la base, et on continue de la même manière jusqu'à la plus petite puissance de B.
  - Exemple1 : Base 10 vers base 2 : $135_{10} = ...2$
    - De 135 on peut (1) retirer 128 reste 7 ->$135 = 2^7 + 7$ (on met 1 en position 7 de la suite binaire)
    - De 7 on peut (1) retirer 4 reste 3 ->$7 = 2^2 + 3$ (on met 1 en position 2 de la suite binaire)
    - De 3 on peut (1) retirer 2 reste 2 ->$3 = 2^1 + 1$ (on met 1 en position 1 de la suite binaire)
    - De 1 on peut (1) retirer 1 reste 0 ->$1 = 2^0 + 0$ (on met 1 en position 0 de la suite binaire)
    - $135_{10} = 10000111_2$

Conversion d'un système de numération à un autre (2)

- Base 10 vers base B
  - Méthode à soustractions successives
  - Exemple 2 : Base 10 vers base 8 239_{10}=.....$_8$
    - 239= 3.8^2 + 47 –> 3 en position 2
    - 47= 5.8^1 + 7 –> 5 en position 1
    - 7= 7.8^0 + 0 –> 7 en position 0
    - 239_{10}=357_8
  - Méthode à divisons successives
  - Elle consiste à diviser successivement par B autant de fois que cela est nécessaire pour obtenir un quotient nul. Ensuite on écrit les restes dans l'ordre inverse de celui dans lequel ils ont été obtenus.
  - Exemple : base 10 vers base 2 : $20,4_{10}$=.....$_2$
    - Partie entière - - - - - - - - - - - - - Partie décimale
      20/2 = 10 reste **0** - - - - - - - - - - 0,4x2= **0** + 0,8
      10/2 = 5 reste **0** - - - - - - - - - - - 0,8x2= **1** + 0,6
      5/2 = 2 reste**1**- - - - - - - - - - - - - 0,6x2= **1** + 0,2
      2/2 = 1 reste **0**
      1/2 = 0 reste **1**
    - $20,4_{10}$=**10100**,**011**$_2$

Conversion d'un système de numération à un autre (3)

Base 2^n vers base 2

- Chaque symbole de la base B $= 2^n$ peut être représenté par n éléments binaires.
- Exemple 1 : Base 16$= 2^4$ vers base 2
  $3A9_{16} = 001110101001_2$
- Exemple 2 : Base 8$= 2^3$ vers base 2
  $742,5_8 = 111100010,101_2$

Base 2 vers base 2^n

- Il suffit de regrouper les éléments binaires par paquets de n.
- Exemple : $1011011_2 =$ .....$_8$
- $1011011_2 = \underbrace{001}_{1}\ \underbrace{011}_{3}\ \underbrace{011}_{3} = 133_8$
- $1011011_2 = \underbrace{0101}_{5}\ \underbrace{1011}_{B} = 5B_{16}$

Conversion d'un système de numération à un autre (4)

- Base i vers base j
  - si i et j sont des puissances de 2, on utilise la base 2 comme relais
  - Exemple : Base 8 –> base 2 –> base 16
  - sinon, on utilise la base 10 comme relais
  - Exemple : Base 5 –> base 10 –> base 2

Les Systèmes de Codage

Codes Numériques pondérés

Le code binaire pur est un code pondéré par des puissances de 2, utilisé en arithmétique binaire. Ses dérivées sont le code octal et le code hexadécimal.

- Exemples BCD (Décimal codé en Binaire), le code Aiken

Codes Numériques non pondérés

Dans ces types de code, aucun poids est affecté à la position d'un bit. On convient simplement d'un tableau de correspondances entre les objets à coder et une représentation.

- Exemple de Codes Numériques non pondérés Code de Gray, Code ASCII

Codes Numériques pondérés

Code BCD

- Dans ce système de codage, chaque chiffre est codé par son équivalent en binaire sur quatre bits. C'est un code pondéré avec les poids 1, 2, 4, 8, 10, 20, 40, 80, 100,200,400,800,1000..
- Exemple : $1998_{10} = 1111100110_2 = 0001100110011000_{BCD}$

Code binaire d'Aiken

- Ce code est pondéré par 2,4,2,1. Il peut être constitué par les règles suivantes :
- de 0 à 4 on code en binaire pur ;
- de 5 à 9 on ajoute 6 et on code en binaire pur. (c.à.d. 5 –> 5 + 6 = 11, 6 –> 6 + 6 =12, . . .)

Code binaire d'Aiken

Décimal	Aiken
	2 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	1 0 1 1
6	1 1 0 0
7	1 1 0 1
8	1 1 1 0
9	1 1 1 1

Codes Numériques non pondérés

Code de Gray (binaire réfléchi)

- Un seul bit change entre deux nombres consécutifs. On y trouve la notion d'adjacence entre deux termes. Le code présente 4 symétries miroir. Il est cyclique : il se referme sur lui-même.

Données non numériques :Codes ASCII , UNICODE

- Ils servent à coder des chiffres, des lettres, des signes de ponctuations et des caractères spéciaux.
- Le codage est réalisé par une table de correspondance, propre à chaque code utilisé.
- ASCII (American Standard Code for Information Interchange) à 7 ou 8 bits
- UNICODE (Extended Binary Coded Decimal Internal Code) à 16 bits

Code de Gray

Décimal	Gray
0	0 0 0
1	0 0 1
2	0 1 1
3	0 1 0
4	1 1 0
5	1 1 1
6	1 0 1
7	1 0 0

Partie 2

ARITHMETIQUE DANS UN SYSTEME DE NUMERATION

Addition Binaire

L'addition binaire se fait avec les mêmes règles qu'en décimal.

- On commence par additionner les bits de poids faibles ;
- On a des retenues lorsque la somme de deux bits de même poids dépasse la valeur de l'unité la plus grande (dans le cas du binaire : 1)
- Cette retenue est reportée sur le bit de poids plus fort suivant.

La table d'addition binaire est la suivante :

Table d'addition

+	0	1
0	0	1
1	1	10

A		B		Résultat	Retenue
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

- Exemple :

$$\begin{array}{r} 1\,0\,1\,1\,0 \\ 1\,1\,1\,1 \\ \hline 1\,0\,0\,1\,0\,1 \end{array}$$

Soustraction Binaire

Dans la soustraction binaire, on procède comme en dcimal qu'en décimal.

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

La table de soustraction binaire est la suivante :

Table d'addition

-	0	1
0	0	11
1	1	0

A		B		Résultat	Retenue
0	-	0	=	0	0
0	-	1	=	1	1
1	-	0	=	1	0
1	-	1	=	0	0

- Exemple 1 : 1 0 1 , 0 / 1 1 , 1 / 0 0 1 , 1

  Exemple 2 : 0 0 0 1 1 / 0 1 1 0 0 / 1 ⟸ 1 0 1 1 1

  La règle ne marche pas pour l'exemple2
  A voir en complément à 1 ou à 2 oú la soustraction d'un nombre se réduit à l'addition de son complément

Multiplication Binaire

- La table de multiplication binaire est la suivante

:

Table d'addition

x	0	1
0	0	1
1	1	10

A		B		Résultat
0	x	0	=	0
0	x	1	=	0
1	x	0	=	0
1	x	1	=	1

- Exemple :

```
  1 0 1 1 0
      1 0 1
 ------------
    1 0 1 1 0
  0 0 0 0 0 .
1 0 1 1 0 .
 ------------
1 1 0 1 1 1 0
```

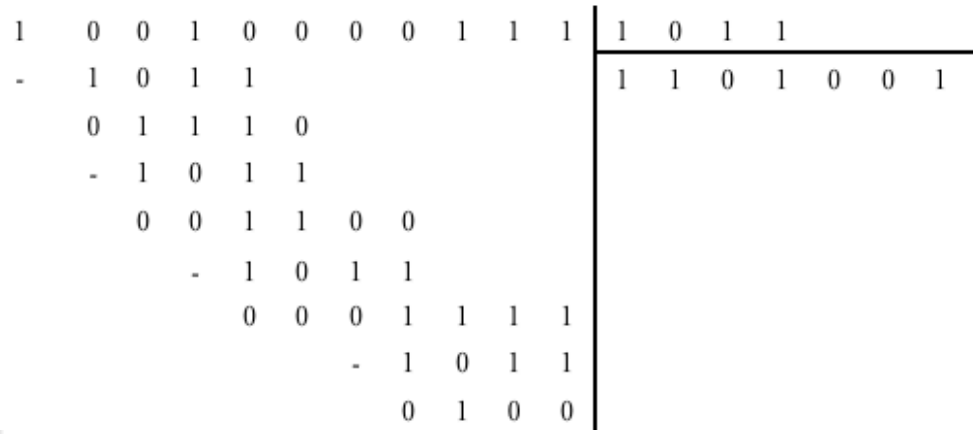
Division Binaire

La table de la division binaire est la suivante

Table d'addition

/	0	1
0	?	1
1	?	10

A		B		Résultat
0	/	0	=	impossible
0	/	1	=	0
1	/	0	=	impossible
1	/	1	=	1



Base 5

Table d'addition

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	21
4	4	10	11	12	13

Table de multiplication

x	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	11	13
3	0	3	11	14	22
4	0	4	13	22	31

Base 8

Table d'addition

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Table de multiplication

x	0	1	2	3	4	5	6	7
0	0	0	2	3	4	5	6	7
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Base 16

Table d'addition

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Base 16

Table de multiplication

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	CA	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Addition en BCD

L'addition de deux nombres cods en DCB revêt une certaine particularité que nous examinons à travers des exemples.

Résultat inférieur à 9

- Pas de problème tant que le résultat est inférieur ou égal à 9

$$\begin{array}{r} 0\ 1\ 0\ 0 \quad 0\ 1\ 0\ 1 \\ + \\ 0\ 1\ 0\ 0 \quad 0\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0 \quad 1\ 0\ 0\ 0 \end{array} \qquad \begin{array}{r} 4\ 5 \\ + \\ 4\ 3 \\ \hline 8\ 8 \end{array}$$

Quand le résultat est supérieur à 9

- Apporter une correction en additionnant 6, afin d'obtenir une réponse valide. Ceci est dû au fait que l'on représente un nombre modulo 10 avec un code modulo 16 : 16-10 = 6.

- Exemple 2 :

$$\begin{array}{r} 0\ 1\ 1\ 0 \\ + \\ 0\ 1\ 1\ 1 \\ = \\ 1\ 1\ 0\ 1 \\ + \\ 0\ 1\ 1\ 0 \\ \hline 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1 \\ \underbrace{}_{1}\ \underbrace{}_{3} \end{array}$$

> 9, on ajoute 6

- Exemple 3 :

$$\begin{array}{r} 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \\ + \\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0 \\ + \\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0 \\ \hline 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0 \\ \underbrace{}_{9}\ \underbrace{}_{4}\ \underbrace{}_{3}\ \underbrace{}_{8} \end{array} \qquad \begin{array}{r} 8\ 7\ 9\ 5 \\ + \\ 6\ 4\ 3 \\ \hline 9\ 4\ 3\ 8 \end{array}$$

Soustraction en BCD

Résultat inférieur à 9

- Dans l'addition, on ajoute 6 quand la somme de deux motifs de 4 bits dépasse 9. Pour la soustraction :
  - lorsque le résultat de la soustraction DCB est inférieure à 9, on ne change pas le résultat
  - lorsque le résultat de la soustraction DCB est sup'erieure à 9, on soustrait 6 résultat
  - lorsqu'il y a une retenue soustractiv,, on soustrait également 6 au résultat obtenu, même si la valeur est inférieure à 9

- Exemple 1 :

$$\begin{array}{r} 20 \\ -\ 8 \\ \hline 12 \end{array} = \begin{array}{r} 00100000 \\ -\ \ \ \ 1000 \\ \hline 00011000 \\ -\ \ \ \ 0110 \\ \hline 00010010 \end{array}$$

emprunt, on soustrait 6

$\underbrace{0001}_{1}\ \underbrace{0010}_{2}$

- Exemple 2 :

$$\begin{array}{r} 22 \\ -\ 3 \\ \hline 12 \end{array} = \begin{array}{r} 00100010 \\ -\ \ \ \ 0011 \\ \hline 00011111 \\ -\ \ \ \ 0110 \\ \hline 00011001 \end{array}$$

emprunt et résultat supérieur à 9, on soustrait 6

$\underbrace{0001}_{1}\ \underbrace{1001}_{9}$

Code de Gray

- Pour convertir un nombre en code binaire naturel (CBN) vers un nombre en code binaire réechi (CBR) :
  - On ajoute le CBN trouvé à lui-même décalé dn rang vers la gauche, sans tenir compte de l'éventuelle retenue
  - On abandonne dans le résultat le bit de poids faible

- Exemple :

$$\begin{array}{r} 1\ 0\ 0\ 1 \\ 1\ 0\ 0\ 1 \\ \hline 1\ 1\ 0\ 1\ 1 \end{array} \Longleftarrow \text{bit à ignoré}$$

Partie 3

REPRESENTATION DES NOMBRES

- Représentation binaire des Nombres entiers positifs
- Représentation binaire des Nombres entiers signés
- Représentation binaire des Nombres réels

Représentation binaire des Nombres entiers positifs

Représentation binaire des Nombres entiers positifs

- Les nombres sont représentés en binaire sur n bits : n=nombre d'unités mémoires (n = 8, 16, 32, 64, ...) On peut représenter des nombres allant de 0 à $2^n - 1$.

Représentation binaire des Nombres entiers signés

- Traditionnellement on met un signe " - " pour représenter les nombres négatifs. Mais les systèmes logiques ne permettent de présenter qu'un des deux symboles " 0 " et " 1 ", il faut chercher une convention pour remplacer le " - "

Représentation binaire des Nombres entiers signés

- Représentation en module et signe
  - Un élément binaire est ajouté à gauche du module pour représenter le signe. Ainsi un nombre commençant par un " 0 " sera positif alors qu'un nombre commençant par un " 1 " sera négatif.
  - Exemple : Avec 4 éléments binaires les valeurs vont de -7 à+ 7.

Signe	Module	Valeur	Signe	Module	Valeur
1	111	-7	0	111	7
1	110	-6	0	110	6
1	101	-5	0	101	5
1	100	-4	0	100	4
1	011	-3	0	011	3
1	010	-2	0	010	2
1	001	-1	0	001	1
1	000	0	0	000	0

- Problème : on a ici deux représentations pour le zéro

Représentation binaire des Nombres entiers négatifs

- Représentation en complément restreint (CR) ou complément à 1(C1) ou complément logique
  - $-A = \overline{A}$:
  - Pour prendre l'inverse d'un nombre, il suffit de le complémenter (inversion de tous ses bits) comme dans le cas précédent, la nature du premier bit donnera le signe : 0 = +, 1= -.
  - Avec 4 bits : $\begin{cases} +5 = 0101 \\ -5 = 1010 \end{cases}$
  - Problème : de nouveau,on a ici deux représentations pour le zéro
- Représentation en complément vrai (CV) ou complément à 2 (C2) ou ou complément arithmétique
  - C'est la représentation la plus utilisée. Le bit le plus à gauche est encore le bit de signe : 0=+ et 1= -.
  - $-A = \overline{A}+1$
  - Une seule représentation pour le 0
  - avec des mots de n bits, on obtient 2^n valeurs différentes, de 0 à 2^{n-1}-1 pour les valeurs positives, et de -1 à -2^{n-1} pour les valeurs négatives ;

Représentation binaire des Nombres entiers négatifs

- Représentation en complément vrait (CV) ou complément à 2 (C2) ou ou complément arithmétique (suite)
  - Exemple avec 4 bits :
    - Valeurs positives de 0 à 7
    - Valeurs négativeses de -1 à -8
  - Exemple avec 3 bits :

décimal	Module et signe	complément à 1	complément à 2
+3	011	011	011
+2	010	010	010
+1	001	001	001
+0	000	000	000
-0	100	111	....
-1	101	110	111
-2	110	101	110
-3	111	100	101
-4	.....	.....	100

Représentation binaire des Nombres Réels

Pour représenter les nombres fractionnaires il est nécessaire de définir la position de la virgule. Pour ce faire, il existe deux méthodes :

Représentation en virgule fixe

- On décide que la virgule soit toujours à une position donnée (un entier peut être représentatif d'un nombre fractionnaire si on connaît la place de la virgule). Tout nombre est mis sous la forme a,b avec "a " chiffre(s) avant la virgule et " b " chiffre(s) après la virgule.
- Exemple : x,y=9,75 : x=9 , y=75 ; a=5 ; b=3
  La position de la virgule est fixée.
- 9,75 =01001,110

Représentation en virgule flottante

- Le nombre N est représenté sous la forme : $\underbrace{exposant}\underbrace{mantisse}$.Deux approches existent.

Représentation en virgule flottante

- Première approche
  - Soit N=$a_3 a_2 a_1 a_0, a_{-1} a_{-2} a_{-3}$
  - N peut se noter : $\underbrace{a_6 a_5 a_4 a_3 a_2 a_1 a_0}_{mantisse} \underbrace{2^{-3}}_{exp}$. Multiplication implicite par 2^{-3}
  - $\Longrightarrow \begin{cases} exposant = -3 \\ mantisse = a_6 a_5 a_4 a_3 a_2 a_1 a_0 \end{cases}$
  - Les valeurs de la mantisse et l'exposant peuvent être notées en complément à 2.
  - Exemple : Soit la mémoire de taille suivante : $\underbrace{4bits}_{exposant} \underbrace{12bits}_{mantisse}$. Coder la valeur 26,75 en virgule flottante.
  - $(26,75)_{10} = (11010,110)_2$
    $(11010,11)_2 = (11010110).2-3$
  - $\Longrightarrow \begin{cases} exposant = -3 \\ mantisse = 11010110_2 \end{cases}$
  - $\underbrace{1101}_{exp=-3_{10}} \underbrace{000011010110}_{mantisse=214_{10}}$
  - $26,75_{10}=214.2_{10}^{-3}$ avec (-3 en complément à 2 =1101)

Représentation en virgule flottante

- La manière la plus évidente et la plus concise pour représenter un nombre en virgule flottante est donc d'employer un exposant et une mantisse signée.
- Exemple :
  $-123,45_{10} = -0,12345.10^{+3}$;$0,0000678_{10} = +0,678.10-4$
- On peut noter également qu'à priori, une représentation en virgule flottante n'est pas nécessairement unique.
- Exemple : $0,2340.10^{+2}$=$0,002340.10^{+4}$=$0,00002340.10^{+6}$
- Il nous faut donc les représenter sous une forme **normalisée** afin que la représentation ne varie pas d'un matériel à l'autre.

Représentation en virgule flottante

- Deuxième approche : Normalisation
  - C'est la méthode inverse de la précédente : on considère que le bit le plus à gauche de la mantisse a pour poids 2^{-1}
  - Ainsi, un nombre normalisé, en virgule flottante, est un nombre dans lequel le chiffre suivant la marque décimale, à gauche de la mantisse (donc à droite de la marque décimale), n'est pas un zéro alors que le nombre à gauche de la marque décimale est un zéro.
  - Soit N=$a_3a_2a_1a_0, a_{-1}a_{-2}a_{-3}$, N peut se noter : $\underbrace{0.a_{-1}a_{-2}a_{-3}a_{-4}a_{-5}a_{-6}a_{-7}}_{mantisse}\underbrace{2^{-4}}_{exp}$.
  - $(26,75)_{10}$ =$(11010,110)_2$=$(0,0011010110)_2.2^7$ n'est pas normalisé
  - Par contre $(26,75)_{10}$ =$(0,11010110)_2.2^5$ est normalisé. On peut omettre le 0 dans la représentation : $\Longrightarrow (,11010110)_2.2^5$
  - $(26,75)_{10}$ peut s'écrire : $\underbrace{0101}_{exposant}\ \underbrace{110101100000}_{mantisse}$
  - On peut trouver, en fonction des organismes de normalisation ou des constructeurs, plusieurs "normes" de représentation des nombres en virgule flottante (IEEE, IBM, ...), nous nous bornerons à présenter ici les normes IEEE.

Représentation en virgule flottante : Norme IEEE

- Le stabdard IEEE définit trois formats de représentation des nombres réels : la simple précision (sur 32 bits), la double précision (sur 64 bits) et la précision étendue (sur 80 bits). Ce dernier est surtout destiné à réduire les erreurs d'arrondis de calculs.

1 bit	8 bits	23 bits
signe manstisse	exposant	mantisse

1 bit	11 bits	52 bits
signe manstisse	exposant	mantisse

- Chaque format commence par un bit de signe de la mantisse (celui le plus à gauche), qui vaut 0 pour les nombres positifs et 1 pour les nombres négatifs. Puis l'exposant est codé en décalage (on dit aussi en excédant) par rapport à l'exposant de référence : 127 pour la simple précision, et 1023 pour la double précision. Enfin la mantisse, est codée en binaire sur 23 ou 52 bits.

Représentation en virgule flottante : Norme IEEE

- Fonctionnement
  - Exemple : pour normaliser le nombre décimal $10,5_{10}$ en virgule flottante, format simple précision dans la base 2 (binaire), avec 64 comme exposant de référence.
  - Il conviendra donc dans un premier temps de transcrire notre nombre 10,50 en base 2, ce qui nous donne $1010,1_2$
  - Ensuite il faut "normaliser" (décaler la virgule vers la gauche de façon à trouver une forme normalisée), ce qui donne donc : $\mathbf{0},10101.2^4$.
  - L'exposant est codé en décalage (on dit aussi en excédant) par rapport à l'exposant de référence, à savoir : 64_{10}.
  - On a donc : Exposant de référence + Décalage = Exposant décalé Soit $64_{10} + 4_{10} = 68_{10}$ soit encore 1000100_2
  - Le signe du nombre étant positif, le bit représentatif du signe sera donc positionné à zéro. Nous aurons ainsi en définitive :

0	01000100	10101000000000000000000
signe manstisse	exp. code excédent 64	mantisse

Représentation en virgule flottante : Norme IEEE

- Fonctionnement (suite)
  - Exemple : normaliser le nombre 432_{10} dans la base 2 exprimé sous la forme $0,0000000000011011_2.2^{20}$ avec un exposant de référence 64_{10}
  - $0,0000000000011011_2.2^{20}$=$0,1101100000000000_2.2^{9}$
  - On a donc : Exposant de référence + Décalage = Exposant décalé Soit $64_{10} + 9_{10} = 73_{10}$ soit encore 1001001_2

0	01001001	11011000000000000000000
signe manstisse	exp. code excédent 64	mantisse

Addition en Complément à 1 ou à 2

- En complément à 1 ou à 2, la soustraction d'un nombre se réduit à l'addition de son complément.
- Dans une addition en complément à 1, une retenue générée par le bit de signe doit être ajoutée au résultat obtenu. Par contre en complément à 2, on ignore cette retenue.
- Lorsqu' on utilise ce procédé, les nombres doivent avoir le même nombre de bits, si besoin est, on ajoute des zéros à l'un des nombres.
  - Exemple 1 : soustraction sur 5 bits : 7-2= 7+(-2)
  - $7_{10} = 00111_2$ et $2_{10} = 00010_2$
  - -2 en complément à 1 =11101
  - -2 en complément à 2 =11110
  - En complément à 1 : 7+(-2)

$$\begin{array}{r} 00111 \\ + \quad 11101 \\ \hline 1 \Longleftarrow 00100 \end{array}$$

+ 1 –> retenue du bit de signe

En complément à 2 : 7+(-2)

$$\begin{array}{r} 00111 \\ 11110 \\ \hline 1 <\!\!- 00101 \end{array}$$

on ne tient pas compte de la retenue

Addition en Complément à 1 ou à 2

- Exemple 2 : soustraction sur 5 bits : 3-6= 3+(-6)
- $3_{10} = 00011_2$ et $6_{10} = 00110_2$
- -6 en complément à 1 =11001
- -6 en complément à 2 =11010
- En complément à 1 : 3+(-6)

  00011
  \+
  11001
  ———
  sans retenue 11100
  à ajouter au résultat.

  En complément à 2 : 3+(-6)

  00011
  11010
  ———
  11101 sans retenue à ignorer
- On trouve un résultat en complément à 1 ou à 2.