

---

---

# Représentation de l'information dans la machine

---

---

# Systemes de numeration

# Introduction

---

- Quelle que soit la nature de l'information traitée par un ordinateur (image, son, texte, vidéo), elle l'est toujours représentée sous la forme d'un ensemble de nombres binaires
- Une information élémentaire correspond à un chiffre binaire (0 ou 1) appelé bit. Le terme **bit** signifie « **binary digit** »
- Le codage de l'information permet d'établir une correspondance entre la représentation externe de l'information et sa représentation binaire

# Introduction

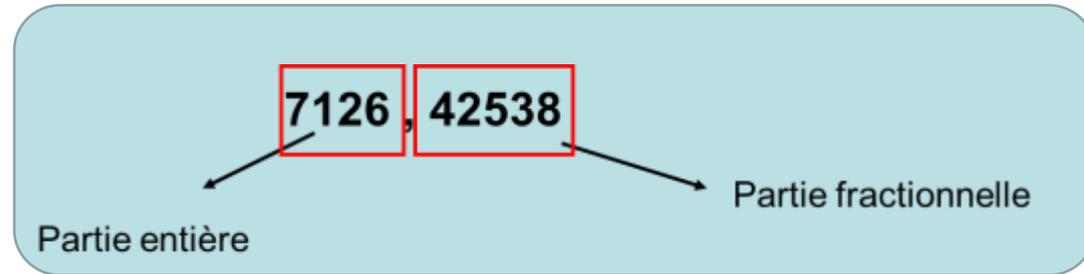
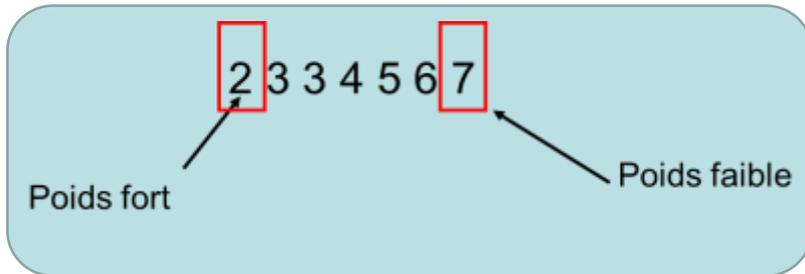
---

---

- Nous avons pris l'habitude de représenter les nombres en utilisant dix symboles différents: 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9
- Ce système est appelé le système **décimal** (déci signifie dix).
- Il existe cependant d'autres formes de numération qui fonctionnent en utilisant un nombre de symboles distincts.
  - Exemple :
    - système binaire (bi: deux),
    - le système octal (oct: huit),
    - le système hexadécimal (hexa: seize).
- Dans un système de numération : le nombre de symboles distincts est appelé **la base** du système de numération.

# Le système décimal

- On utilise dix symboles différents:  
 $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- N'importe quelle combinaison des symboles  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$  nous donne un nombre.



## Exemple:

le nombre 3213, peut être écrit sous la forme suivante :

$$3213 = 3 * 10^3 + 2 * 10^2 + 1 * 10^1 + 3 * 10^0$$

C'est la forme **polynomiale**

- Un nombre **réel** peut être écrit aussi sous la forme polynomiale:

$$3213,987 = 3 * 10^3 + 2 * 10^2 + 1 * 10^1 + 3 * 10^0 + 9 * 10^{-1} + 8 * 10^{-2} + 7 * 10^{-3}$$

# Systeme binaire ( systeme à base 2 )

## Exemple

- Sur un seul bit : 0 , 1

Sur 2 bits :

Binaire	Décimal
00	0
01	1
10	2
11	3

**4 combinaisons =  $2^2$**

Sur 3 Bits

Binaire	Décimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

**8 combinaisons =  $2^3$**

# Le système octal ( base 8 )

- 8 symboles sont utilisés dans ce système:  
 $\{ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 \}$

- **Exemple 1 :**

$$(526)_8 = 5 * 8^2 + 2 * 8^1 + 6 * 8^0$$

$$(537,235)_8 = 5 * 8^2 + 3 * 8^1 + 7 * 8^0 + 2 * 8^{-1} + 3 * 8^{-2} + 5 * 8^{-3}$$

- **Exemple 2 :**

Le nombre (1289) n'existe pas dans la base 8 puisque les symboles 8 et 9 n'appartiennent pas à la base .

# Le système hexadécimal ( base 16 )

- On utilise seize (16) symboles différents:

$$(17)_{16} = 1 * 16^1 + 7 * 16^0$$

$$(AB)_{16} = A * 16^1 + B * 16^0 = 10 * 16^1 + 11 * 1$$

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

# Résumé

---

- Dans une base  $X$ , on utilise  $X$  symboles distincts pour représenter les nombres.
- 
- La valeur de chaque symbole doit être strictement inférieur à la base  $X$ .
- Chaque nombre dans une base  $X$  peut être écrit sous sa forme polynomiale

# Conversion d'une base X à la base 10

- Cette conversion est assez simple puisque il suffit de faire le développement en **polynôme** de ce nombre dans la base **X**, et de faire la somme par la suite.

## Exemple :

$$(1101)_2 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = (13)_{10}$$

$$(1A7)_{16} = 1 * 16^2 + A * 16^1 + 7 * 16^0 = 1 * 16^2 + 10 * 16^1 + 7 * 16^0 = 256 + 160 + 7 = (423)_{10}$$

$$(1101,101)_2 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} = (13,625)_{10}$$

$$(43,2)_5 = 4 * 5^1 + 3 * 5^0 + 2 * 5^{-1} = 20 + 3 + 0,4 = (23,4)_{10}$$

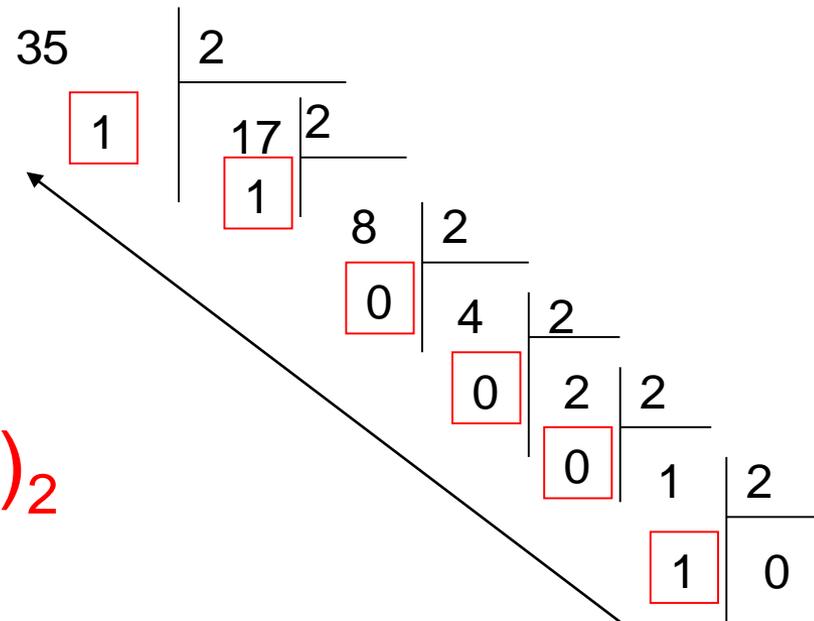
# Conversion de la base 10 à la base 2

Le principe consiste à faire des divisions successives du nombre sur 2, et prendre le reste des divisions dans l'ordre inverse.

Exemple 1 :  $(35)_{10} = (?)_2$

Après division :

on obtient :  $(35)_{10} = (100011)_2$



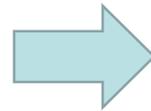
# Conversion de la base 10 à la base 2 cas d'un nombre réel

- Un nombre réel est constitué de deux parties : la partie entière et la partie fractionnelle.
- La partie entière est transformée en effectuant des divisions successives.
- La partie fractionnelle est transformée en effectuant des multiplications successives par 2 .

Exemple :  $35,625 = (?)_2$

Partie entière =  $35 = (100011)_2$

Partie fractionnelle =  $0,625 = (?)_2$



**Partie fractionnelle**

$$0,625 * 2 = \boxed{1},25$$

$$0,25 * 2 = \boxed{0},5$$

$$0,5 * 2 = \boxed{1},0$$



$$(0,625) = (0,101)_2$$

$$\text{Donc } 35,625 = (100011,101)_2$$

# Conversion de la base 10 à la base 2 cas d'un nombre réel

- **Exemple 2:**  $(0,6)_{10} = (?)_2$

$$0,6 * 2 = 1,2$$

$$0,2 * 2 = 0,4$$

$$0,4 * 2 = 0,8$$

$$0,8 * 2 = 1,6$$



$$(0,6) = (0,1001)_2$$

Le nombre de bits après la virgule va déterminer la précision

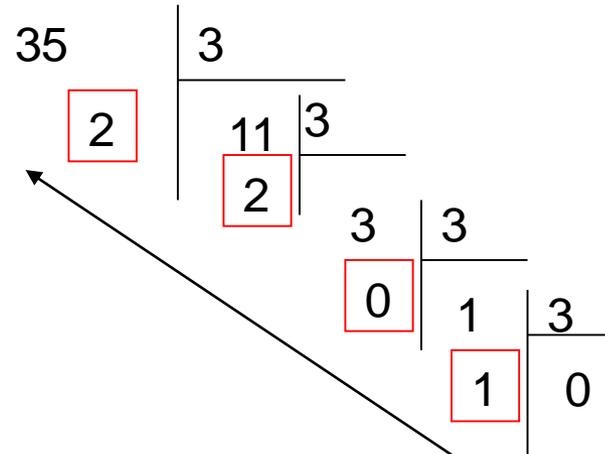
# Conversion du décimal à une base X

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

Exemple :

$$(35)_{10} = (?)_3$$

$$(35)_{10} = (1022)_3$$



# Conversion du décimal à une base X

- **Question** : Effectuer les transformations suivantes :

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

43

	2							
1		21	2					
		1	10					
		0	5	2				
			1	2	2			
				0	1	2		
					1	0		

**(101011)<sub>2</sub>**

43

	5					
3		8	5			
		3	1	5		
			1			

**(133)<sub>5</sub>**

43

	8			
3		5	8	
		5	0	

**(53)<sub>8</sub>**

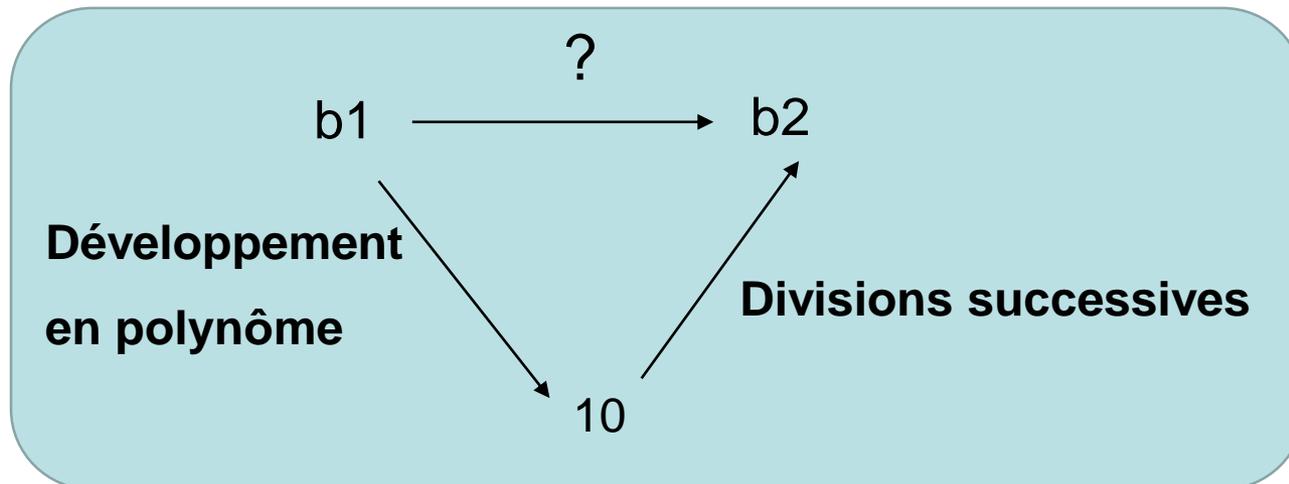
43

	16			
11		2	16	
		2	0	

**(2B)<sub>16</sub>**

# Conversion d'une base $b_1$ à une base $b_2$

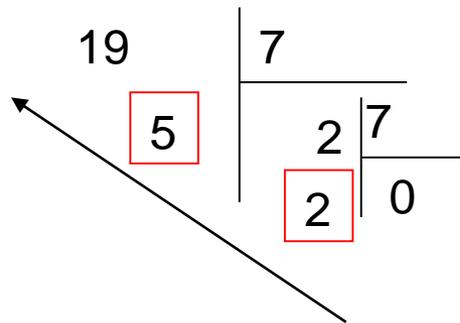
- Il n'existe pas de méthode pour passer d'une base  $b_1$  à une autre base  $b_2$  directement.
- L'idée est de convertir le nombre de la base  $b_1$  à la base 10, en suit convertir le résultat de la base 10 à la base  $b_2$ .



# Conversion d'une base b1 à une base b2

Exemple :  $(34)_5 = (?)_7$

$$(34)_5 = 3 * 5^1 + 4 * 5^0 = 15 + 4 = (19)_{10} = (?)_7$$



$$(19)_{10} = (25)_7 \longrightarrow (34)_5 = (25)_7$$

# Conversion : Octal → binaire

. En octal chaque, symbole de la base s'écrit **sur 3 bits en binaire**.

. L'idée de base est de remplacer chaque symbole dans la base octal par sa valeur en binaire sur 3 bits ( faire des éclatement sur 3 bits ).

## Exemples :

$$(345)_8 = (\underline{011} \ \underline{100} \ \underline{101})_2$$

$$(65,76)_8 = (\underline{110} \ \underline{101}, \ \underline{111} \ \underline{110})_2$$

$$(35,34)_8 = (\underline{011} \ \underline{101}, \ \underline{011} \ \underline{100})_2$$

Octal	Binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

## Remarque :

le remplacement se fait de droit à gauche pour la partie entière et de gauche à droite pour la partie fractionnelle .

# Conversion : binaire → octal

---

- L'idée est de faire des **regroupements** de **3 bits** à partir du poids faible.
- Par la suite **remplacer** chaque regroupement par la valeur octal correspondante

## Exemple :

$$(11001010010110)_2 = (\underline{011} \ \underline{001} \ \underline{010} \ \underline{010} \ \underline{110})_2 = (31226)_8$$

$$(110010100,10101)_2 = (\underline{110} \ \underline{010} \ \underline{100} \ , \ \underline{101} \ \underline{010})_2 = (624,51)_8$$

## Remarque :

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

# Conversion : hexadécimal → binaire

- En Hexa chaque symbole de la base s'écrit sur 4 bits.
- L'idée de base est de remplacer chaque symbole par sa valeur en binaire sur 4 bits ( faire des éclatement sur 4 bits ).

## Exemple :

$$(345B)_{16} = (\underline{0011} \ \underline{0100} \ \underline{0101} \ \underline{1011})_2$$

$$(AB3,4F6)_{16} = ( \ \underline{1010} \ \underline{1011} \ \underline{0011} \ , \ \underline{0100} \ \underline{1111} \ \underline{0110} \ )_2$$

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

# Conversion : binaire → hexadécimal

---

L'idée est de faire des **regroupements de 4 bits** à partir du poids faible.

Par la suite remplacer chaque regroupement par la valeur Héxa correspondante .

## Exemple :

$$(11001010100110)_2 = (\underline{0011} \ \underline{0010} \ \underline{1010} \ \underline{0110})_2 = \mathbf{(32A6)_{16}}$$

$$(110010100,10101)_2 = (\underline{0001} \ \underline{1001} \ \underline{0100}, \underline{1010} \ \underline{1000})_2 = \mathbf{(194,A8)_{16}}$$

# Opérations arithmétiques en binaire

---

---

$$\begin{array}{r} + 0 \\ 0 \\ \hline \end{array}$$

0

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

1

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

1

$$\begin{array}{r} + 1 \\ 1 \\ \hline \end{array}$$

1 0

1 1

1 1 0 0 0 1 1

+

1 0 0 0 1 0 1 1

---

1 1 1 0 1 1 1 0

# Opérations arithmétiques en hexadécimal

## Opérations en Octal

$$\begin{array}{r} \boxed{1} \quad \boxed{1} \\ 4 \quad 3 \quad 6 \quad 5 \\ + \quad \quad 4 \quad 5 \quad 1 \\ \hline \boxed{5} \quad \boxed{8} \quad \boxed{11} \quad \boxed{6} \\ \text{En octal 8 s'écrit 10} \quad \text{En octal 11 s'écrit 13} \\ \quad \boxed{0} \quad \quad \boxed{3} \end{array}$$

Le résultat final :  $(5036)_8$

## Opérations en hexadécimal

$$\begin{array}{r} \boxed{1} \\ 4 \quad 8 \quad 6 \quad 5 \\ + \quad \quad 7 \quad A \quad 5 \quad 1 \\ \hline \boxed{12} \quad \boxed{18} \quad \boxed{11} \quad \boxed{6} \\ \text{En hexa 18 s'écrit 12} \quad \text{En hexa 11 s'écrit B} \\ \quad \boxed{2} \quad \quad \boxed{B} \\ \text{C} \end{array}$$

Le résultat final :  $(C2B6)_{16}$

# Exercice

- Effectuer les transformations suivantes à la base 10 ?

$$- (123)_6 = (?)_{10}$$

$$- (45,76)_8 = (?)_{10}$$

$$- (1100,11)_2 = (?)_{10}$$

$$- (1ABC)_{16} = (?)_{10}$$

- Effectuer les transformations suivantes

$$(43)_6 = (?)_5 = (?)_8$$

$$(2A)_{16} = (?)_9$$

- Effectuer les opérations suivantes et transformer le résultat au décimal à chaque fois:

$$(1101,111)_2 + (11,1)_2 = (?)_2$$

$$(43)_8 + (34)_8 = (?)_8$$

$$(43)_6 + (34)_6 = (?)_6$$

$$(AB1)_{16} + (237)_8 = (?)_{16}$$

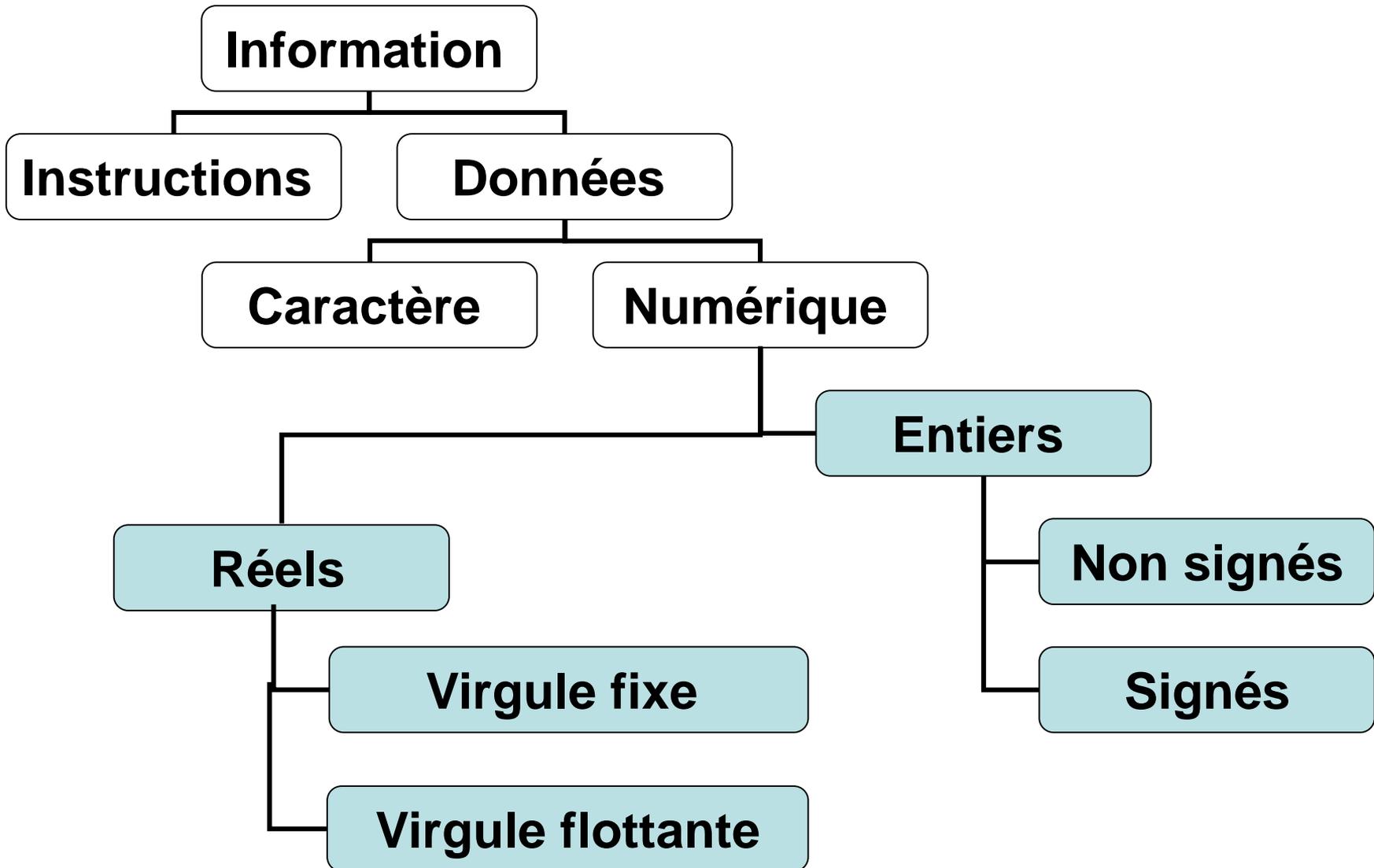
# Représentation de l'information

# Introduction

---

- Les machines numériques utilisent le **ystème binaire**.
- Dans le système binaire : uniquement 2 symboles sont utilisés : 0 et 1.
- C'est **facile de représenter ces deux symboles** dans les machines numériques.
- Le 0 et le 1 sont représentés par **deux tensions** .

# Introduction



# Représentation des nombres entiers

---

- Il existe deux types d'entiers :
  - les entiers non signés ( positifs )
  - et les entiers signés ( positifs ou négatifs )
- **Problème** : Comment indiquer à la machine qu'un nombre est négatif ou positif ?
- Il existe 3 méthodes pour représenter les nombres négatifs :
  - Signe/ valeur absolue
  - Complément à 1 ( complément restreint )
  - Complément à 2 ( complément à vrai )

# Représentation signe / valeur absolue

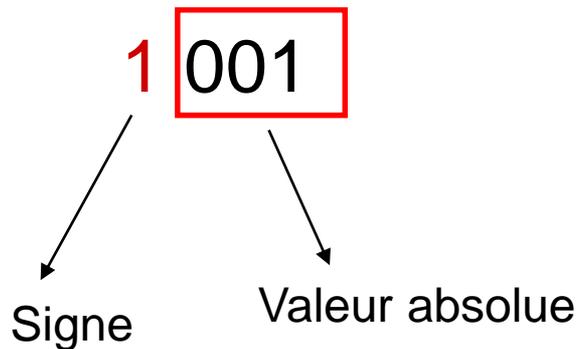
- Sur  $n$  bits , alors le bit du poids fort est utilisé pour indiquer le signe :
  - 1 : signe négatif
  - 0 : signe positif
- Les autres bits (  $n - 1$  ) désignent la valeur absolue du nombre.

(**1** 1 0 **1**)<sub>2</sub>

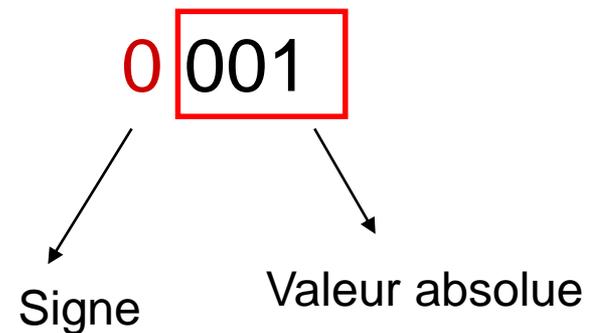
Le bits du poids forts

Le bits du poids faible

Exemple :



1001 est la représentation de - 1



0001 est la représentation de + 1

# Représentation signe / valeur absolue

Sur 3 bits on obtient :

signe	VA	valeur
0	00	+ 0
0	01	+ 1
0	10	+ 2
0	11	+ 3
1	00	- 0
1	01	- 1
1	10	- 2
1	11	- 3

- Les valeurs sont comprises entre -3 et +3

$$-3 \leq N \leq +3$$

$$-(4-1) \leq N \leq +(4-1)$$

$$-(2^2-1) \leq N \leq +(2^2-1)$$

$$-(2^{(3-1)}-1) \leq N \leq +(2^{(3-1)}-1)$$

Sur **n** bits , l'intervalle des valeurs qu'on peut représenter en système en valeur absolue:

$$-(2^{(n-1)}-1) \leq N \leq +(2^{(n-1)}-1)$$

# Représentation signe / valeur absolue

---

## Avantages et inconvénients:

- Représentation assez simple .
- Le zéro possède deux représentations +0 et -0 ce qui conduit à des difficultés au niveau des opérations arithmétiques.
- Pour les opérations arithmétiques il nous faut deux circuits : l'un pour l'addition et le deuxième pour la soustraction .

⇒ L'idéal est d'utiliser un seul circuit pour faire les deux opérations, puisque :

$$X - Y = X + ( -Y )$$

# Représentation en complément à un

- On appelle **complément à un** d'un nombre  $N$  un autre nombre  $N'$  tel que :

$$N + N' = 2^n - 1$$

**n** : est le nombre de bits de la représentation du nombre  $N$  .

## Exemple :

Soit  $N=1010$  sur 4 bits donc son complément à un de  $N$  :

$$N' = (2^4 - 1) - N$$

$$N' = (16 - 1)_{10} - (1010)_2 = (15)_{10} - (1010)_2 = (1111)_2 - (1010)_2 = 0101$$

	1	0	1	0
+	0	1	0	1
	1	1	1	1

# Représentation en complément à un

Sur  $n$  bits , l'intervalle des valeurs qu'on peut représenter en CA1 :

$$-(2^{(n-1)} - 1) \leq N \leq +(2^{(n-1)} - 1)$$

Exemple : sur 3 bits, les valeurs sont comprises entre -3 et +3

$$\begin{aligned} -3 &\leq N \leq +3 \\ - ( 4-1 ) &\leq N \leq + ( 4 -1 ) \\ -(2^2 -1) &\leq N \leq +(2^2-1) \\ -(2^{(3-1)} -1) &\leq N \leq +(2^{(3-1)} -1) \end{aligned}$$

# Représentation en complément à un

Sur 3 bits :

Valeur en CA1	Valeur en binaire	Valeur décimale
000	000	+ 0
001	001	+ 1
010	010	+ 2
011	011	+ 3
100	- 011	- 3
101	- 010	- 2
110	- 001	- 1
111	- 000	- 0

• Dans cette représentation , le bit du poids fort nous indique le signe: 0 : positif , 1 : négatif ).

• **On remarque que dans cette représentation le zéro possède aussi une double représentation ( +0 et -0 ) .**

# Représentation en complément à un

---

Exemple :

Quelle est la valeur décimale représentée par la valeur 101010 en complément à 1 sur 6 bits ?

- Le bit poids fort indique qu'il s'agit d'un nombre négatif.
- Valeur = - CA1(101010)  
= - (010101)<sub>2</sub> = - (21)<sub>10</sub>

---

---

# Représentation en complément à 2

---

---

# Représentation en complément à 2

- Soit **X** un nombre sur **n** bits alors :

$$X + 2^n = X \text{ modulo } 2^n$$

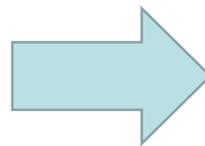
Le résultat sur **n** bits  la même valeur que **X** :

$$X + 2^n = X$$

Exemple : soit  $X = 1001$  sur 4 bits

$$2^4 = 10000$$

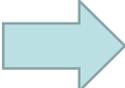
$$\begin{array}{r} \phantom{+} \phantom{1} \phantom{0} \phantom{0} \phantom{1} \\ + \phantom{1} \phantom{0} \phantom{0} \phantom{0} \phantom{0} \\ \hline 1 \phantom{1} \phantom{0} \phantom{0} \phantom{1} \end{array}$$



Si on prend le résultat sur 4 bits on trouve la même valeur de **X = 1001**

# Représentation en complément à 2

• Si on prend deux nombres entiers **X** et **Y** sur **n** bits, on remarque que la soustraction peut être ramener à une addition :

**$X - Y = X + (-Y)$**   trouver une valeur équivalente à **-Y** ?

$$X - Y = (X + 2^n) - Y = X + (2^n - 1) - Y + 1$$

$$\text{On a } Y + \text{CA1}(Y) = 2^n - 1 \text{ donc } \text{CA1}(Y) = (2^n - 1) - Y$$

On obtient :

$$X - Y = X + \text{CA1}(Y) + 1$$

# Représentation en complément à 2

---

La valeur  $CA1(Y)+1$  s'appelle le complément à deux de  $b$  :

$$CA1(Y)+1 = CA2(Y)$$

Et enfin on va obtenir :

$$X - Y = X + CA2(Y) \rightarrow \text{transformer la soustraction en une}$$

addition .

# Représentation en complément à 2

Si on travail sur 3 bits :

Valeur en CA2	Valeur en binaire	valeur
000	000	+ 0
001	001	+ 1
010	010	+ 2
011	011	+ 3
100	- 100	- 4
101	- 011	- 3
110	- 010	- 2
111	- 001	- 1

• Sur 3 bits on remarque que les valeurs sont comprises entre -4 et +3

$$\begin{aligned} -4 &\leq N \leq +3 \\ -4 &\leq N \leq +(4 - 1) \\ -2^2 &\leq N \leq +(2^2 - 1) \\ -2^{(3-1)} &\leq N \leq (2^{(3-1)} - 1) \end{aligned}$$

Si on travail sur  $n$  bits ,  
l'intervalle des valeurs qu'on  
peut représenter en CA2 :  
 $-(2^{(n-1)}) \leq N \leq +(2^{(n-1)} - 1)$

- Dans cette représentation , le bit du poids fort nous indique le signe .
- On remarque que le zéro n'a pas une double représentation.

# Représentation en complément à 2

- Astuces:**

Pour trouver le complément à 2 d'un nombre : il faut parcourir les bits de ce nombre à partir du poids faible et garder tous les bits avant le premier 1 et inverser les autres bits qui viennent après.

0	1	0	0	0	1	0	1
↓	↓	↓	↓	↓	↓	↓	↓
1	0	1	1	1	0	1	1

1	1	0	1	0	1	0	0
↓	↓	↓	↓	↓	↓	↓	↓
0	0	1	0	1	1	0	0

Le complément à deux du complément à deux d'un nombre est égale au nombre lui-même .

$$CA2(CA2(N))= N$$

# Représentation en complément à 2

---

## Exemple 1:

Trouver le complément à 2 : 01000101 sur 8 bits ?

## Exemple 2 :

Quelle est la valeur décimale représentée par la valeur 101010 en complément à deux sur 6 bits ?

Le bit poids fort indique qu'il s'agit d'un nombre négatif.

$$\text{Valeur} = - \text{CA2}(101010)$$

$$= - (010101 + 1)$$

$$= - (010110)_2 = - (22)$$

# Représentation en complément à 2

## Exemple 3:

Effectuer les opérations suivantes sur 5 Bits....

$$\begin{array}{r} + 9 \\ + 4 \\ \hline + 13 \end{array} \quad \begin{array}{r} + \\ 0\ 1\ 0\ 0\ 1 \\ 0\ 0\ 1\ 0\ 0 \\ \hline 0\ 1\ 1\ 0\ 1 \end{array}$$

Le résultat est positif

$$(01101)_2 = (13)_{10}$$

$$\begin{array}{r} + 9 \\ - 4 \\ \hline + 5 \end{array} \quad \begin{array}{r} + \\ 0\ 1\ 0\ 0\ 1 \\ 1\ 1\ 1\ 0\ 0 \\ \hline 1\ 0\ 0\ 1\ 0\ 1 \end{array}$$

Report

Le résultat est positif

$$(00101)_2 = (5)_{10}$$

# Représentation en complément à 2

## Exemple 4:

Effectuer les opérations suivantes sur 5 Bits

			1	0	1	1	1	
- 9	+							
- 4			1	1	1	0	0	
<hr/>								
- 13			1	1	0	0	1	1

Report

Le résultat est négatif :

Résultat = - CA2 (10011) = -(01101)

= - 13

				1	0	1	1	1
- 9	+							
+ 9				0	1	0	0	1
<hr/>								
+ 0			1	0	0	0	0	0

Report

Le résultat est positif

$(00000)_2 = (0)_{10}$

# Représentation en complément à 2

---

---

## La retenue et le débordement

- On dit qu'il y a une **retenue** si une opération arithmétique génère un report .
- On dit qu'il y a un **débordement (Over Flow )** ou **dépassement de capacité**: si le résultat de l'opération sur  $n$  bits est faux .
  - Le nombre de bits utilisés est insuffisant pour contenir le résultat
  - Autrement dit le résultat dépasse l'intervalle des valeurs sur les  $n$  bits utilisés.

# Représentation en complément à 2

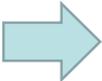
## Cas de débordement

		0	1				
			0	1	0	0	1
+ 9	+						
+ 8			0	1	0	0	0
<hr/>							
+ 17			1	0	0	0	1

Négatif

		1	0				
			1	0	1	1	1
- 9	+						
- 8			1	1	0	0	0
<hr/>							
- 17			0	1	0	1	1

Positif

- Un débordement  - si la somme de deux nombres positifs donne un nombre négatif .
- ou la somme de deux nombres négatifs donne un Nombre positif
- Il y a jamais un débordement si les deux nombres sont de signes différents.

---

---

# Représentation des nombres réels

---

---

# Représentation des nombres réels

---

---

- **Problème** : comment indiquer à la machine la position de la virgule ?
- Un nombre réel est constitué de deux parties : la partie entière et la partie fractionnelle ( les deux parties sont séparées par une virgule )
- Il existe **deux méthodes** pour représenter les nombre réel :
  - 1) **Virgule fixe** : la position de la virgule est fixe
  - 2) **Virgule flottante** : la position de la virgule change ( dynamique )

# Représentation en virgule fixe

- Dans cette représentation la partie entière est représentée sur  $n$  bits et la partie fractionnelle sur  $p$  bits , en plus un bit est utilisé pour le signe.
- Exemple : si  $n=3$  et  $p=2$  on va avoir les valeurs suivantes

Signe	Partie entière	Partie fractionnelle	valeur
0	000	00	+ 0,0
0	000	01	+ 0,25
0	000	10	+ 0,5
0	000	11	+ 0,75
0	001	.00	+ 1,0
.	.	.	.
.	.	.	.

Dans cette représentation les valeurs sont limitées et nous n'avons pas une grande précision

# Représentation en virgule flottante

- Chaque nombre réel peut s'écrire de la façon suivante :

$$N = \pm M * b^e$$

**M** : mantisse  
**b** : la base  
**e** : l'exposant

- Exemple :

$$13,11 = 0,1311 * 10^{+2}$$

$$-(110,101)_2 = -(0,110101)_2 * 2^{+3}$$

$$(0,00101)_2 = (0,101)_2 * 2^{-2}$$

## Remarque :

on dit que la mantisse est **normalisée** si le **premier chiffre après la virgule** est différent de **0** et le **premier chiffre avant la virgule** est égale à **0**.

# Représentation en virgule flottante

- Dans cette représentation sur  $n$  bits :
  - La mantisse est sous la forme signe/valeur absolue
    - 1 bit pour le signe
    - et  $k$  bits pour la valeur.
  - L'exposant ( positif ou négatif ) est représenté sur  $p$  bits .

<b>Signe mantisse</b>	<b>Exposant</b>	<b>Mantisse normalisée</b>
<b>1 bit</b>	<b><math>p</math> bits</b>	<b><math>k</math> bits</b>

- Pour la représentation de l'exposant on utilise :

- 1) **Le complément à deux**
- 2) **Exposant décalé ou biaisé**

# Représentation de l'exposant en complément à deux

- Exemple:** on veut représenter les nombres  $(0,015)_8$  et  $-(15,01)_8$  en virgule flottante sur une machine ayant le format suivant :



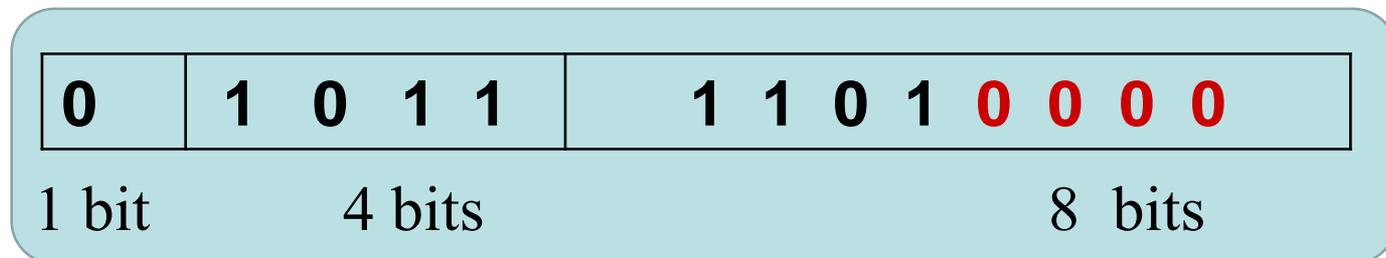
$$(0,015)_8 = (0,000001101)_2 = 0,1101 * 2^{-5}$$

**Signe mantisse** : positif (0)

**Mantisse normalisé** : 0,1101

**Exposant** = -5 → utiliser le complément à deux pour représenter le -5

Sur 4 bits **CA2(0101)=1011**



# Représentation de l'exposant en complément à deux

$$-(15,01)_8 = -(001101,000001)_2 = -0,1101000001 * 2^4$$

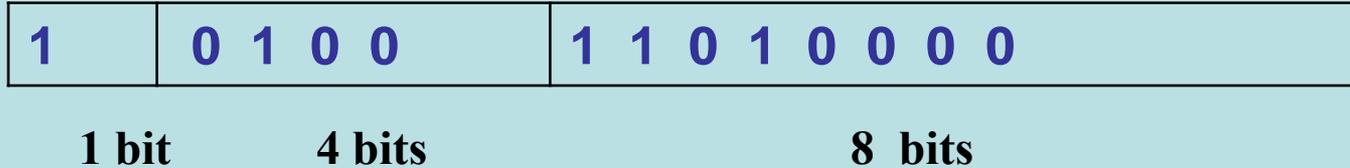
**Signe mantisse** : négatif ( 1)

**Mantisse normalisée** : 0,1101000001

**Exposant** = 4 , en complément à deux il garde la même valeur ( 0100)

On remarque que la mantisse est sur 10 bits (1101 0000 **01**), et sur la machine **seulement 8 bits** sont utilisés pour la mantisse.

Dans ce cas on va prendre **les 8 premiers bits de la mantisse**



## Remarque :

si la mantisse est sur  $k$  bits et si elle est représenté sur la machine sur  $k'$  bits tel que  $k > k'$  , alors la mantisse sera tronquée : on va prendre uniquement  $k'$  bits → **perdre dans la précision** .

# L' exposant décalé ( biaisé )

- en complément à 2, l'intervalle des valeurs qu'on peut représenter sur  $p$  bits :

$$- 2^{(p-1)} \leq N \leq 2^{(p-1)} - 1$$

Si on rajoute la valeur  $2^{(p-1)}$  à tout les terme de cette inégalité :

$$- 2^{(p-1)} + 2^{(p-1)} \leq N + 2^{(p-1)} \leq 2^{(p-1)} - 1 + 2^{(p-1)}$$

$$0 \leq N + 2^{(p-1)} \leq 2^p - 1$$

- On pose  $N' = N + 2^{(p-1)}$  donc :  $0 \leq N' \leq 2^p - 1$

Dans ce cas on obtient des valeur positives.

- La valeur  $2^{p-1}$  s'appelle le **biais** ou le **décalage**

# L' exposant décalé ( biaisé )

---

- Avec l'exposant biaisé on a transformé les exposants négatifs à des exposants positifs en rajoutons à l'exposant la valeur  $2^{p-1}$ .

$$\text{Exposant Biaisé} = \text{Exposant réel} + \text{Biais}$$

# L' exposant décalé ( biaisé )

## Exemple

On veut représenter les nombres  $(0,015)_8$  et  $-(15,01)_8$  en virgule flottante sur une machine ayant le format suivant :

Signe mantisse	Exposant décalé	Mantisse normalisée
1 bit	4 bits	11 bits

$$(0,015)_8 = (0,000001101)_2 = 0,1101 * 2^{-5}$$

Signe mantisse : positif ( 0 )

Mantisse normalisée : 0,1101

Exposant réel = -5

Calculer le biais :  $b = 2^{4-1} = 8$

Exposant Biaisé =  $-5 + 8 = +3 = (0011)_2$

0	0011	1 1 0 1 0 0 0 0 0 0 0
1 bit	4 bits	11 bits

# L' exposant décalé ( biaisé )

$$- (15,01)_8 = -(001101,000001)_2 = -0,1101000001 * 2^4$$

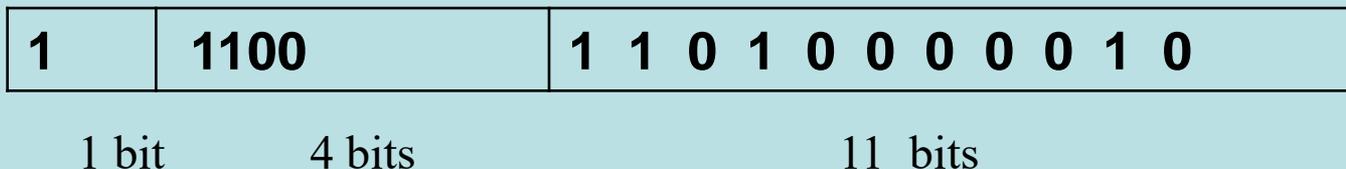
**Signe mantisse** : négatif ( 1 )

**Mantisse normalisée** : 0,1101000001

**Exposant réel** = + 4

**Calculer le biais** :  $b = 2^{4-1} = 8$

**Exposant Biaisé** =  $4 + 8 = +12 = (1100)_2$



# Opérations arithmétiques

Soit deux nombres réels  $N1$  et  $N2$  tel que

$$N1 = M1 \cdot b^{e1} \quad \text{et} \quad N2 = M2 \cdot b^{e2}$$

On veut calculer  $N1 + N2$  ?

Deux cas se présentent :

- 1) Si  $e1 = e2$  alors  $N3 = (M1 + M2) b^{e1}$
- 2) Si  $e1 \neq e2$  alors élever au plus grand exposant et faire l'addition des mantisses et par la suite normalisée la mantisse du résultat.

## Exemple

- Effectuer l'opération suivante sur la machine suivante:

$$(0,15)_8 + (1,5)_8 = (?) :$$

Signe mantisse	Exposant biaisé (décalé)	Mantisse normalisée
1 bit	4 bits	6 bits

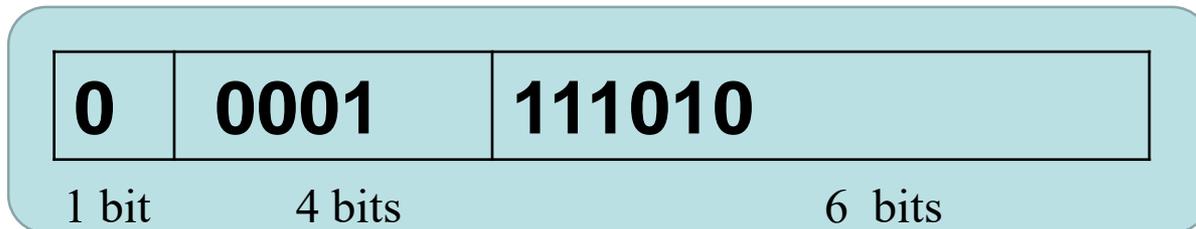
# Opérations arithmétiques

Correction:

$$(0,15)_8 = (0,001101)_2 = 0,1101 * 2^{-2}$$

$$(1,5)_8 = (001, 1 01)_2 = 0,1101 * 2^1$$

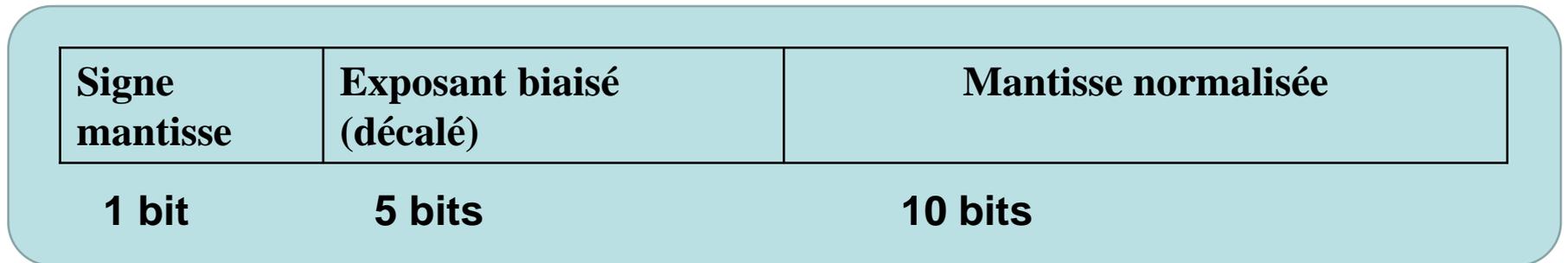
$$\begin{aligned}(0,15)_8 + (1,5)_8 &= 0,1101 * 2^{-2} + 0,1101 * 2^1 \\ &= 0,0001101 * 2^1 + 0,1101 * 2^1 \\ &= 0, 1110101 * 2^1\end{aligned}$$



# Opérations arithmétiques

## Exercice 2

1) Donner la représentation des deux nombres  $N1 = (-0,014)_8$  et  $N2 = (0,14)_8$  sur la machine suivante :



2) Calculer  $N2 - N1$  ?

# Opérations arithmétiques

## Correction:

Avant de représenter les deux nombres on doit calculer le biais (décalage)

$$N1 = (-0,014)_8 = (-0,000001100)_2 = (-0,1100)_2 \cdot 2^{-5}$$

$$B = 2^{5-1} = 2^4 = 16$$

$$\text{ExpB} = -5 + 16 = 11 = (01011)_2$$

$$N2 = (0,14)_8 = (0,001100)_2 = (0,1100)_2 \cdot 2^{-2}$$

$$\text{ExpB} = -2 + 16 = 14 = (01110)_2$$

Donc on va avoir la représentation suivante pour N1 et N2:

N1

1	01011	1100000000
---	-------	------------

1 010 1111 0000 0000  
(AF00)<sub>16</sub>

N2

0	01110	1100000000
---	-------	------------

0011 10 11 0000 0000  
(3B00)<sub>16</sub>

# Opérations arithmétiques

$$N2 - N1 = 0,14 - (-0,014) = 0,14 + 0,014$$

$$\begin{aligned} N2 - N1 &= (0,1100)_2 \cdot 2^{-2} + (0,1100)_2 \cdot 2^{-5} \\ &= (0,1100)_2 \cdot 2^{-2} + (0,0001100)_2 \cdot 2^{-2} \\ &= (0,1101100)_2 \cdot 2^{-2} \end{aligned}$$

• Si on fait les calculs avec l'exposant biaisé :

$$\begin{aligned} N2 - N1 &= (0,1100)_2 \cdot 2^{14} + (0,1100)_2 \cdot 2^{11} \\ &= (0,1100)_2 \cdot 2^{14} + (0,0001100)_2 \cdot 2^{14} \\ &= (0,1101100)_2 \cdot 2^{14} \end{aligned}$$

**Exposant biaisé = 14**

**Exposant réel = Exposant biaisé – Biais**

**Exposant réel = 14 – 16 = -2**

$$\Rightarrow N2 - N1 = (0,1101100)_2 \cdot 2^{-2}$$

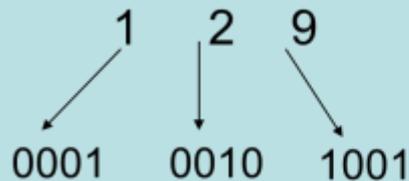
Donc on trouve le même résultat que la première opération.

# Le codage BCD (Binary Coded Decimal )

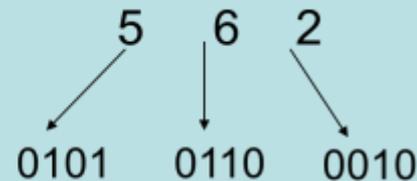
- Pour passer du décimal au binaire , il faut effectuer des divisions successives. Il existe une autre méthode simplifiée pour le passage du décimal au binaire.
- Le principe consiste à faire des éclatement sur 4 bits et de remplacer chaque chiffre décimal par sa valeur binaire correspondante .
- Les combinaisons supérieures à 9 sont interdites

Décimal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

## Exemple



$$129 = (0001\ 0010\ 1001)_2$$

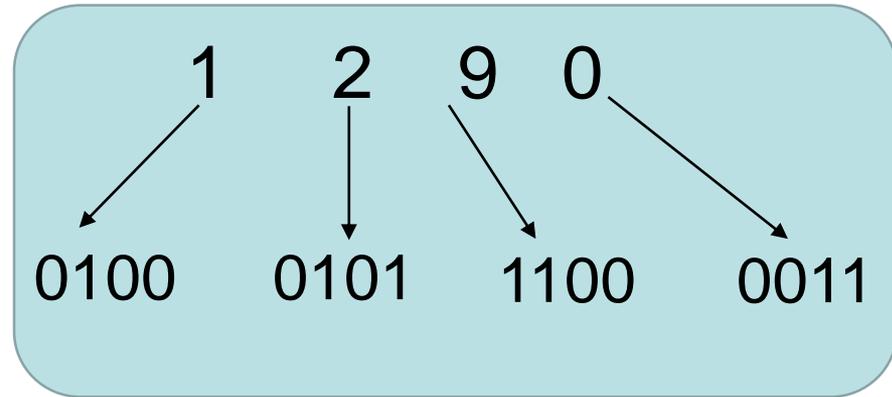


$$562 = (0101\ 0110\ 0010)_2$$

# Le codage EXCESS3 ( BCD+3 )

Décimal	BCD+3	Binaire
0	3	0011
1	4	0100
2	5	0101
3	6	0110
4	7	0111
5	8	1000
6	9	1001
7	10	1010
8	11	1011
9	12	1100

## Exemple



L'avantage principal de l'encodage XS-3 sur l'encodage BCD est qu'on peut calculer le complément à 9 d'un nombre décimal aussi facilement qu'on peut calculer le complément à 1

# Codage des caractères

---

---

- Les caractères englobent : les lettres alphabétiques ( A, a, B , b,.. ) , les chiffres , et les autres symboles ( > , ; / : ..... ) .
- Le codage le plus utilisé est le **ASCII** (American Standard Code for Information Interchange)
- 7 bits ->128 caractères :
  - 26 lettres majuscules A - Z
  - 26 lettres minuscule a - z
  - 10 chiffres 0 à 9
  - 33 caractères de ponctuation  
sp,! " # \$ % & ' ( ) \* + , - . / < = > ? @ [ ] ^ \_ ` { | } ~
  - 33 caractères de contrôle :  
null, etx, bel, bs, ht, lf, vt, ff, cr, ..., del

# Codage des caractères

- Le code **ASCII**

Chaque caractère a un « code » unique

Entier entre 0 et 255

Exemple

- E 69
- x 120
- e 101
- m 109
- p 112
- l 108
- e 101

32		64	@	96	`	48	0	80	P	112	p
33	!	65	A	97	a	49	1	81	Q	113	q
34	"	66	B	98	b	50	2	82	R	114	r
35	#	67	C	99	c	51	3	83	S	115	s
36	\$	68	D	100	d	52	4	84	T	116	t
37	%	69	E	101	e	53	5	85	U	117	u
38	&	70	F	102	f	54	6	86	V	118	v
39	'	71	G	103	g	55	7	87	W	119	w
40	(	72	H	104	h	56	8	88	X	120	x
41	)	73	I	105	i	57	9	89	Y	121	y
42	*	74	J	106	j	58	:	90	Z	122	z
43	+	75	K	107	k	59	;	91	[	123	{
44	,	76	L	108	l	60	<	92	\	124	
45	-	77	M	109	m	61	=	93	]	125	}
46	.	78	N	110	n	62	>	94	^	126	~
47	/	79	O	111	o	63	?	95	_	127	

# Codage des caractères

---

## ASCII étendu

- Dans ce codage chaque caractère est représenté sur **8 bits** .
- Avec **8 bits** on peut avoir  **$2^8 = 256$  combinaisons**
- Chaque combinaison représente un caractère
  - Exemple :
    - Le code 65  $(01000001)_2$  correspond au caractère **A**
    - Le code 97  $(01100001)_2$  correspond au caractère **a**
    - Le code 58  $(00111010)_2$  correspond au caractère **:**
- Actuellement il existe un autre code sur 16 bits , se code s'appel **UNICODE** .

# Codage des caractères

---

## Par exemple :

- ‘A’ =  $65_{10} = 0100\ 0001_2$
- ‘B’ =  $66_{10} = 0100\ 0010$
- ...
- ‘a’ =  $97_{10} = 0110\ 0101$
- ‘ ‘ =  $32_{10} = 0010\ 0000$
- ‘0’ =  $48_{10} = 0011\ 0000$
- ‘1’ =  $49_{10} = 0011\ 0001$
- ‘2’ =  $50_{10} = 0011\ 0010$
- ...
- ‘9’ =  $57_{10} = 0011\ 1001$

# Codage des caractères

---

- Actuellement il existe un autre code sur 16 bits , se code s'appel **UNICODE** .
- **UNICODE** 16 bits -> 65 536 caractères

Ce code contient -> en plus de tous les caractères européens, 42 000 caractères asiatiques.

Le **code ASCII** est contenu dans les **128 premiers** caractères d'UNICODE.

# Codage des caractères

---

Il existe deux formats :

1) 16 bits (UCS-2)

2) ou 32 bits (UCS-4)



**ISO/IEC 10646**

- UCS-2 équivalent à UNICODE 2.0
- UCS-4 inclut :
  - Caractères musicaux
  - Symboles mathématiques
  - Écritures anciennes telles que les hiéroglyphes.

---

---

**Merci pour votre attention**

---

---