Module: logique combinatoire et séquentielle

Crédit: 2

Coefficient: 1

Unité: Méthodologique

Chapitre 1 : Systèmes de numération

- Introduction
- Système décimal
- Système binaire , octal et hexadécimal
- Conversion d'un système de numération vers un autre système .
- Opérations arithmétiques en binaire, octal et hexadécimal.

Objectifs

- Comprendre c’est quoi un système de numération .
- Apprendre la méthode de conversion d’un système à un autre .
- Apprendre à faire des opérations arithmétiques en binaire.

Introduction

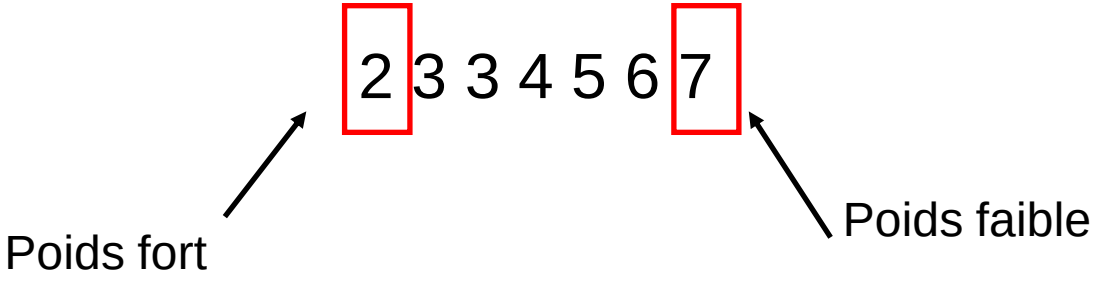
- Nous avons pris l'habitude de représenter les nombres en utilisant dix symboles différents: 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9
- Ce système est appelé le système décimal (déci signifie dix).
- Il existe cependant d'autres formes de numération qui fonctionnent en utilisant un nombre de symboles distincts.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

1 . 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.



Développement en polynôme d'un nombre dans le système décimal

- Soit le nombre 1978, ce nombre peut être écrit sous la forme suivante :

$$1978 = 1000 + 900 + 70 + 8$$
$$1978 = 1*1000 + 9*100 + 7*10 + 8*1$$
$$1978 = 1*10^3 + 9*10^2 + 7*10^1 + 8*10^0$$

Cette forme s'appelle la forme polynomiale

Un nombre réel peut être écrit aussi sous la forme polynomiale

$$1978{,}265 = 1*10^3 + 9*10^2 + 7*10^1 + 8*10^0 + 2*10^{-1} + 6*10^{-2} + 5*10^{-3}$$

Comptage en décimal

- Sur une seule position : 0 ,1,2,3,4,5,….9= 10^1-1
- Sur deux positions : 00 , 01,02, …..,99=10^2-1
- Sur trois positions 000,001,……,999=10^3-1

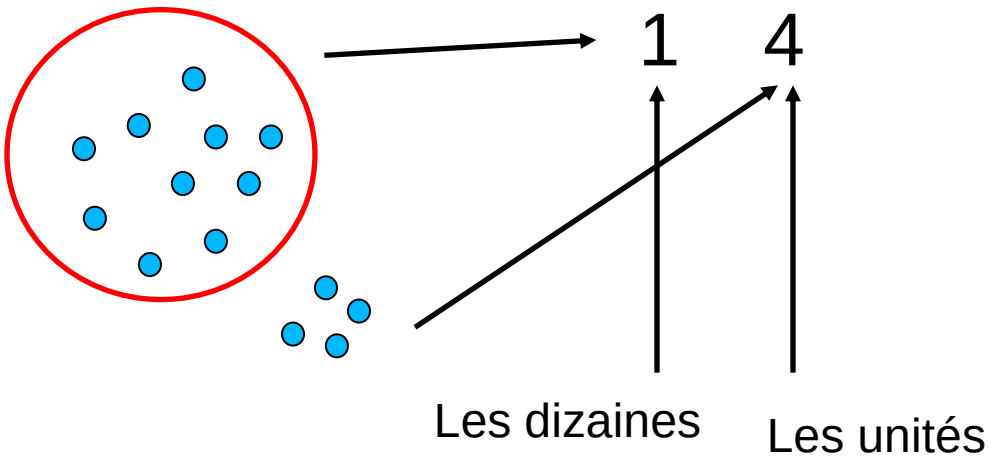
- Sur n positions : minimum 0

  maximum 10^n-1

  nombre de combinaisons 10^n

2 . Système binaire (système à base 2): exemple illustratif

Supposons qu'on a 14 jetons , si on forme des groupes de 10 jetons. On va obtenir 1 seul groupe et il reste 4 jetons.

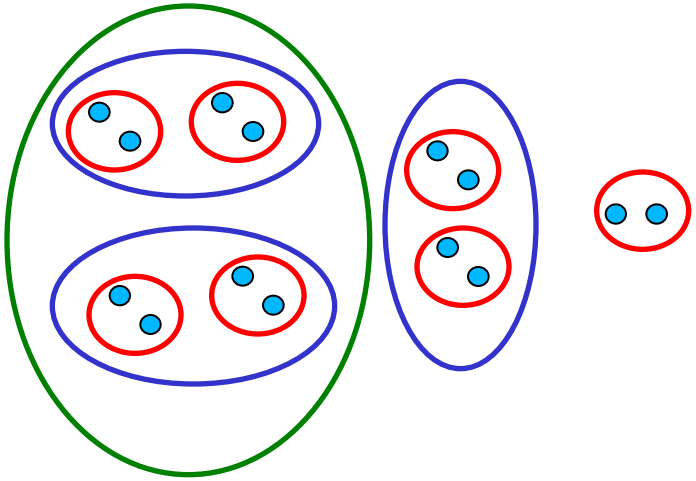


. Maintenant on va former des groupes de 2 jetons (on obtient 7 groupes)

. Par la suite on va regrouper les 7 groupes 2 à 2 (on obtient 3 groupes).

. On va regrouper ces derniers aussi 2 à 2 (on obtient 1 seul groupe)

. Le schéma illustre le principe :

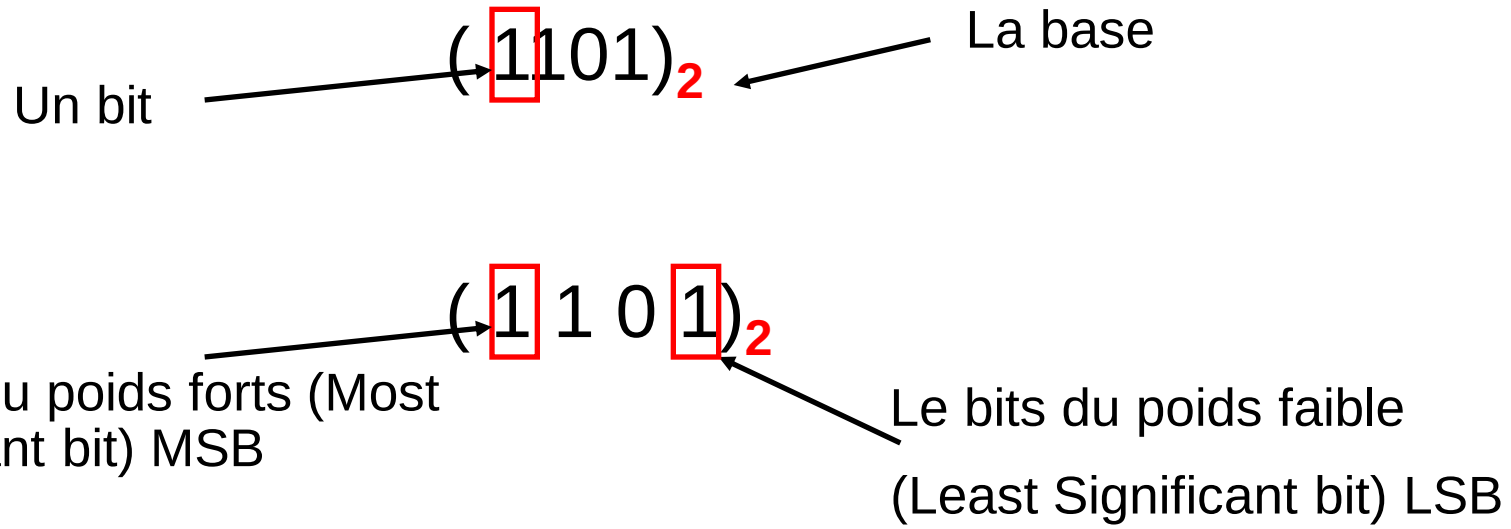


Nombre de jetons qui restent en dehors des groupes : 0
Nombre de groupes qui contiennent 2 jetons : 1
Nombre de groupes qui contiennent 2 groupes de 2 jetons : 1
Nombre de groupes qui contiennent des groupes de 2 groupes de 4 jetons : 1

Si on regroupe les différents chiffres on obtient : 1110

1110 est la représentation de 14 dans la base 2

- Dans le système binaire, pour exprimer n'importe quelle valeur on utilise uniquement 2 symboles : { 0 , 1}



. Un nombre dans la base 2 peut être écrit aussi sous la forme polynomial

$$(1110)_2 = 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = (14)_{10}$$

$$(1110{,}101)_2 = 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3} = (14{,}625)_{10}$$

Comptage en binaire

- Sur un seul bit : 0 , 1

  $0 <= N <= 2^1 - 1$

  $0 <= N <= 1$

- Sur 2 bits :

Binaire	Décimal
00	0
01	1
10	2
11	3

$0 <= N <= 2^2 - 1$

$0 <= N <= 3$

4 combinaisons= 2^2

L'intervalle sur n bits est $0 <= N <= 2^n - 1$

Sur 3 Bits

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

$0 <= N <= 2^3 - 1$

$0 <= N <= 7$

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 :**

$$(127)_8 = 1*8^2 + 2*8^1 + 7*8^0$$

$$(127{,}65)_8 = 1*8^2 + 2*8^1 + 7*8^0 + 6*8^{-1} + 5*8^{-2}$$

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

Décimal	Binaire	Octal	Hexadécimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Résumé

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

$$a_n\ b^n + a_{n-1}\, b^{n-1} + \cdots + a_2\, b^2 + a_1\, b + a_0 + a_{-1}\, b^{-1} + a_{-2}\, b^{-2} + \cdots + a_{-m}\, b^{-m}$$

3. 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 :

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

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

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

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

Exercice

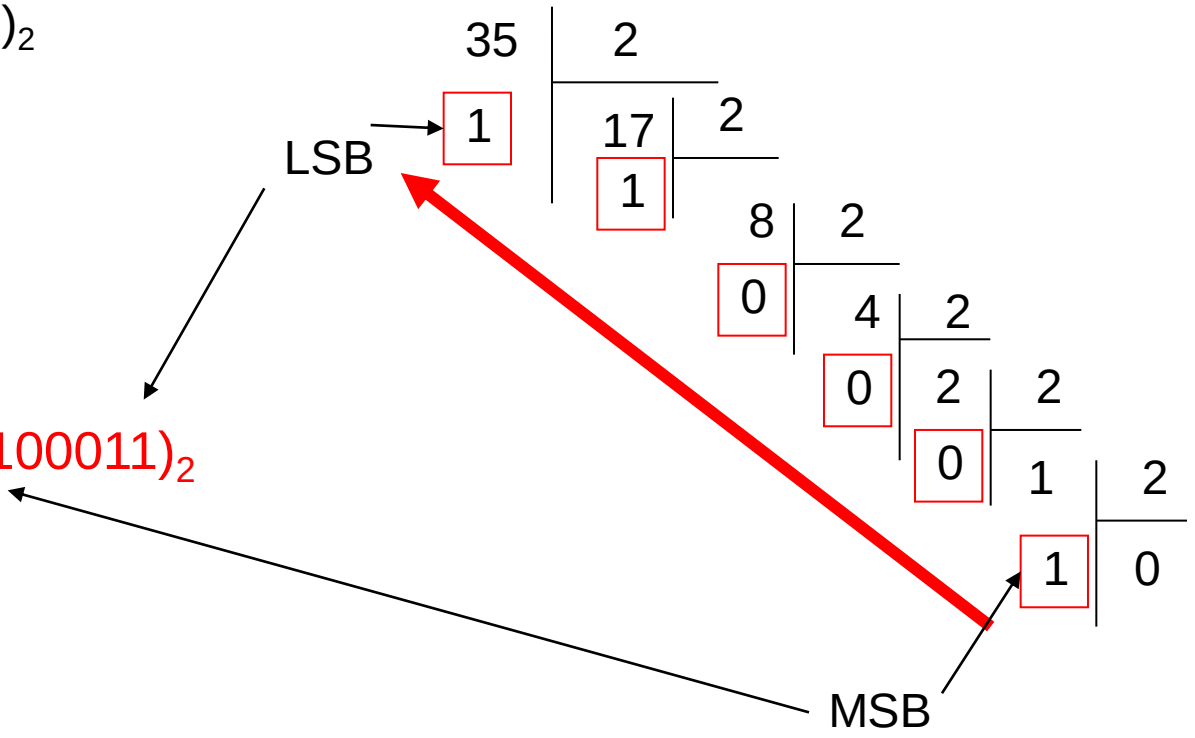
- Effectuer les transformations suivantes à la base 10 ?
  - $(123)_6=(?)_{10}$
  - $(45,76)_8=(?)_{10}$
  - $(1100,11)_2=(?)_{10}$
  - $(1ABC)_{16}=(?)_{10}$

Conversion du décimal à une base X

a) 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

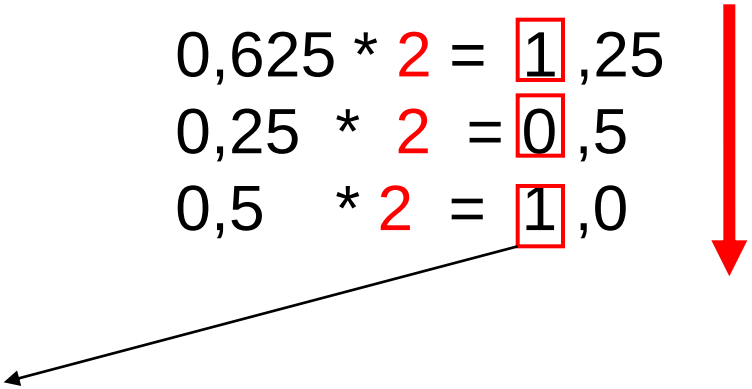
- Un nombre réel est constitué de deux parties : la partie entière et la partie fractionnelle.
- La partie entière est transformée en effectuant des divisions successives.
- La partie fractionnelle est transformée en effectuant des multiplications successives par 2 .

Exemple : $(35,625)_{10}=(?)_2$

P.E= 35 = $(100011)_2$

PF= 0,625 = $(?)_2$

0,625 * 2 = 1 ,25
0,25 * 2 = 0 ,5
0,5 * 2 = 1 ,0



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

Donc 35,625=$(100011,101)_2$

- **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

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

Remarque :

Le nombre de bits après la virgule va déterminer la précision .

Exercice :

Effectuer les transformations suivantes :

$(23,65)=(?\ )_2$

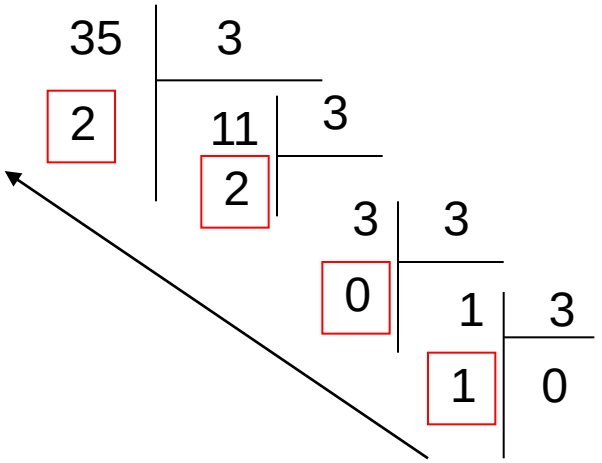
$(18,190)=(?)_2$

Conversion du décimal à la base X

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

Exemple :

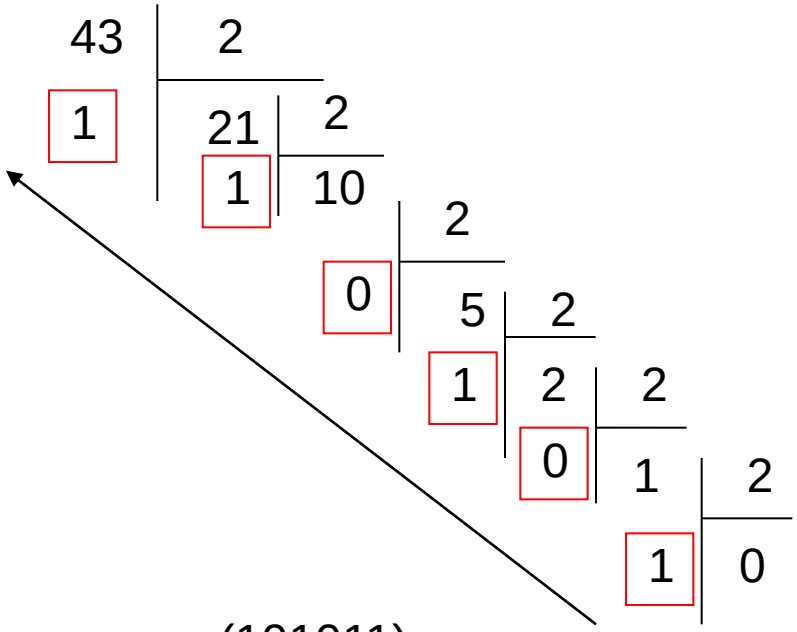
$35 = (?)_3$



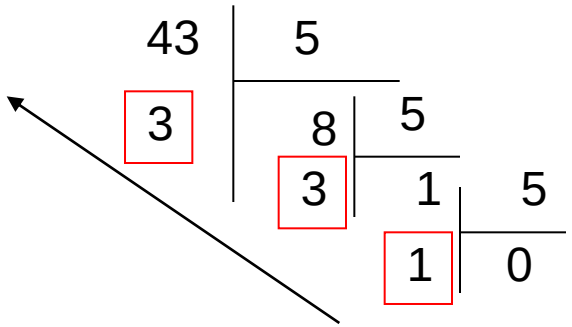
$35=(1022)_3$

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

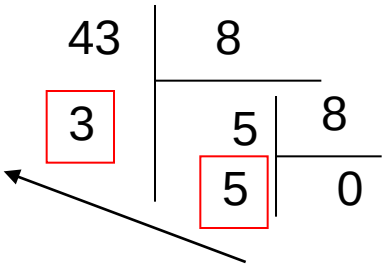
$$(43)_{10}=(?)_2=(?)_5=(?)_8=(?)_{16}$$



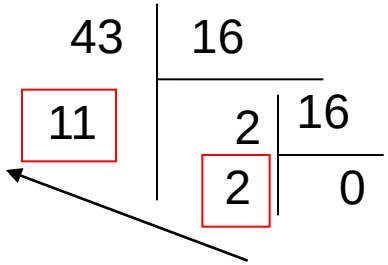
$(101011)_2$



$(133)_5$



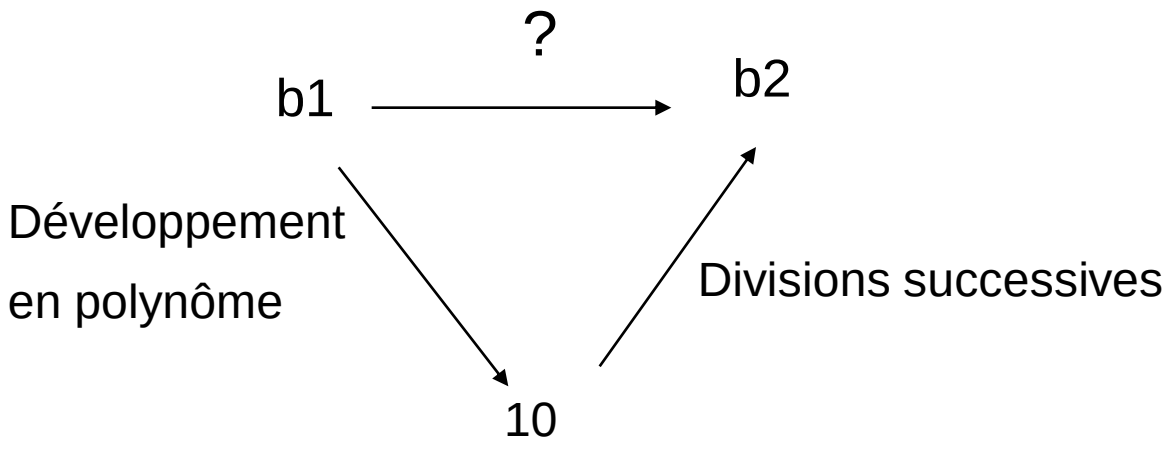
$(53)_8$



$(2B)_{16}$

Conversion d'une base b1 à une base b2

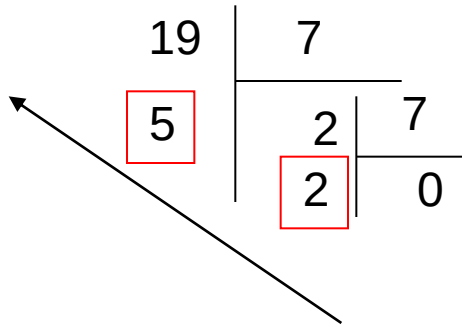
- Il n'existe pas de méthode pour passer d'une base b1 à une autre base b2 directement.
- L'idée est de convertir le nombre de la base b1 à la base 10 , en suit convertir le résultat de la base 10 à la base b2 .



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

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

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



Exercice : effectuer les transformations suivantes

$(43)_6=(?)_5=(?)_8$
$(2A)_{16}=(?)_9$

Conversion : binaire → octal

. En octal chaque, symbole de la base s'écrit sur 3 bits en binaire. $8 = 2^3$

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

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

Exemples :

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

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

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

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

Conversion : Octal → binaire

- L'idée de base 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 :

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

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

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.

$16 = 2^4$

- L'idée de base est de replacer chaque symbole par sa valeur en binaire sur 4 bits (faire des éclatement sur 4 bits).

Exemple :

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

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

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

Conversion : binaire →hexadécimal

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

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

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

Exemple :

$(1100101010011 0)_2=(0011\ 0010\ 1010\ 0110)_2=(32A6)_{16}$

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

La conversion d'une base b1= 2^N à une base b2= 2^M

Si les deux bases b1 et b2 de puissance de 2, On peut utiliser la base 2 comme passerelle .

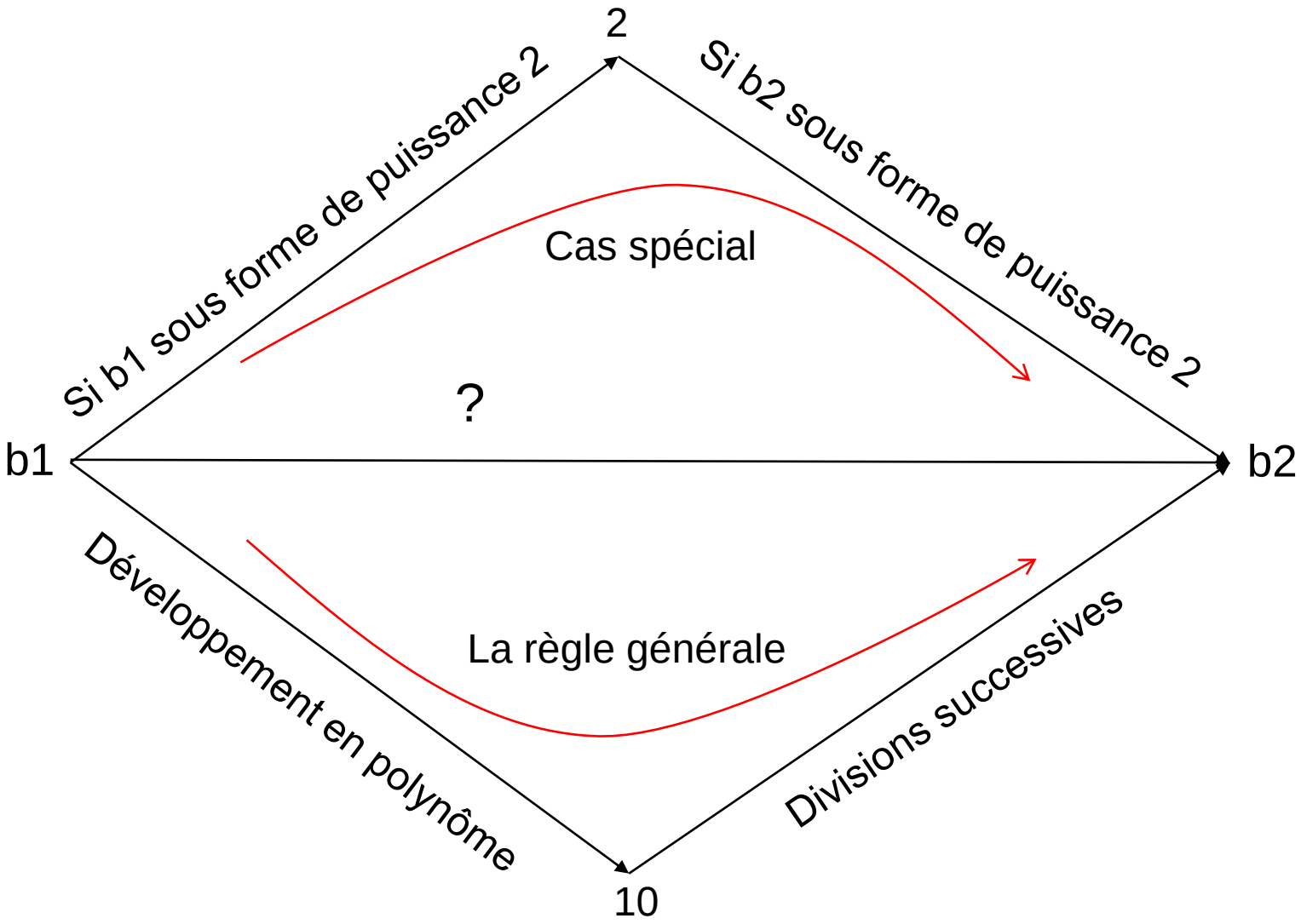
Exemple

Base 8 → base 2 → base 16

Base 4 → base 2 → base 8

$(345)_8 = (\underline{011}\ \underline{100}\ \underline{101})_2 = (\underline{0000}\ \underline{1110}\ \underline{0101})_2 = (0E5)_{16} = (E5)_{16}$

$(32231)_4 = (\underline{11}\ \underline{10}\ \underline{10}\ \underline{11}\ \underline{01})_2 = (\underline{001}\ \underline{011}\ \underline{101}\ \underline{101})_2 = (1355)_8 = (2ED)_{16}$


2
Si b1 sous forme de puissance 2
Si b2 sous forme de puissance 2
Cas spécial
?
b1
b2
La règle générale
Développement en polynôme
Divisions successives
10

Le complément à 1

pour prendre le complément à 1 est d’inverser le bit zéro par 1 et le bit 1 par zéro en prend en considération le nombre de bits de représentation. C'est-à-dire, si la représentation est sur 8 bits, il faut compléter les cases vides de gauche par des 0 avant la complémentation. Ce qui nous a donné des 1 au moment de complémentation.

Exemple: (1011100101) le complément à 1 (0100011010)

Si la représentation est sur 8 bits alors: (110100) le complément à 1 est (?)

Avant de prendre le complément il faut compléter le nombre de représentation c-t-d (00110100) donc le complément à 1 est (11001011)

Le complément à 2

Pour faire le complément à 2 : complément à 2 = complément à 1+ 1

Exemple: le complément à 2 de (11000110) =()

Le complément à 1 de (11000110) est (00111001)+1= 00111010

Une autre méthode, conserver tous bits à partir de la droite jusqu'au premier 1 compris et de changer les autres bits de 0 en 1 ou de 1 en 0.

Exemple: (11000110) le complément à 2 est (00111010) garder 10 de poids faible et inverser les autres

Représentation sur 4 bits: 1100 cà2 0100 ; 1000 cà2 1000 ; 11 cà2 1101

Représentation sur 8 bits: 1100 cà2 11110100 ; 1000 cà2 11111000 ; 11 cà2 11111101 ; 10000000 cà2 10000000

1. Représentation des nombres entiers

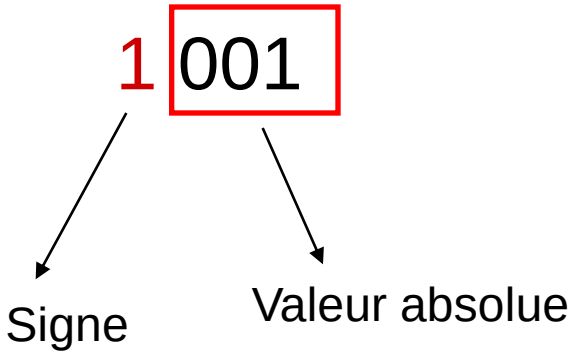
- Il existe deux types d'entiers :
  - les entiers non signés (positifs)
  - et les entiers signés (positifs ou négatifs)
- **Problème :** Comment indiquer à la machine qu'un nombre est négatif ou positif ?

- Il existe 3 méthodes pour représenter les nombres négatifs :
  - Signe/ valeur absolue
  - Complément à 1(complément restreint)
  - Complément à 2 (complément à vrai)

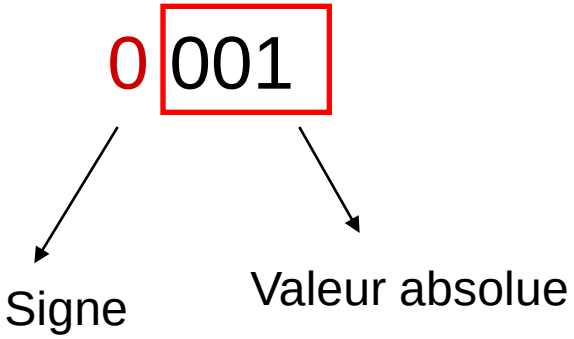
1.1 Représentation signe / valeur absolue (S/VA)

- Si on travail 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.

- Exemple : Si on travail sur 4 bits.



1001 est la représentation de - 1



0001 est la représentation de + 1

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

$-3 \leq N \leq +3$

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

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

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

Si on travail sur **n** bits , l'intervalle des valeurs qu'on peut représenter en S/VA :

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

Avantages et inconvénients de la représentation signe/valeur absolue

- C’est une représentation assez simple .
- On remarque que 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 $a- b =a + (-b)$

1.2 Représentation en complément à un (complément restreint)

- On appel 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 :

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

$$N'=(16-1)-(1010)_2= (15) - (1010)_2 = (1111)_2 – (1010)_2 = 0101$$

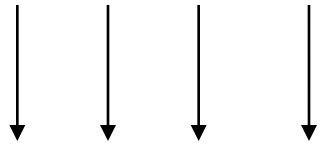
$$\begin{array}{r} 1\ 0\ 1\ 0 \\ +\ 0\ 1\ 0\ 1 \\ \hline 1\ 1\ 1\ 1 \end{array}$$

Remarque 1 :

- Pour trouver le complément à un d'un nombre, il suffit d'inverser tous les bits de ce nombre : si le bit est un 0 mettre à sa place un 1 et si c'est un 1 mettre à sa place un 0 .
- Exemple :

Sur 4 Bits

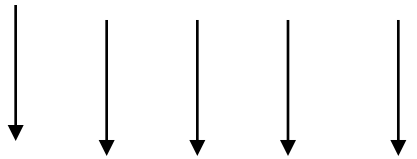
1 0 1 0



0 1 0 1

Sur 5 Bits

0 1 0 1 0



1 0 1 0 1

Méthode 2: représentation par cà1

- Dans cette représentation , le bit du poids fort nous indique le signe (0 : positif , 1 : négatif).
- Le complément à un du complément à un d'un nombre est égale au nombre lui même .

CA1(CA1(N))= N

- 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)

= - $(010101)_2$= - $(21)_{10}$

Si on travail sur 3 bits :

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

•Dans cette représentation , le bit du poids fort nous indique le signe .

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

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

$$-3 \leq N \leq +3$$
$$-(4-1) \leq N \leq +(4-1)$$
$$-(2^2-1) \leq N \leq +(2^2-1)$$
$$-(2^{(3-1)}-1) \leq N \leq +(2^{(3-1)}-1)$$

Si on travail sur n bits , l'intervalle des valeurs qu'on peut représenter en CA1 :

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

1.3 Complément à 2 (complément à vrai)

- Le complément à 2 de a représenté sur n bits est :

$$2^n - a = \text{complément à 2 de a}$$

Exemple : soit a = 1001 sur 4 bits

$2^4 = 10000$

10000- 1001=0111

Donc le complément à 2 de 1001 est 0111

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

Exemple

- Trouver le complément à vrai de : 01000101 sur 8 bits ?

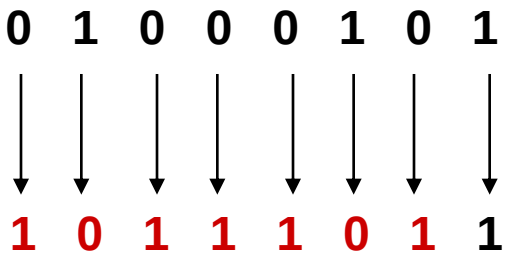
CA2(01000101)= CA1(01000101)+ 1

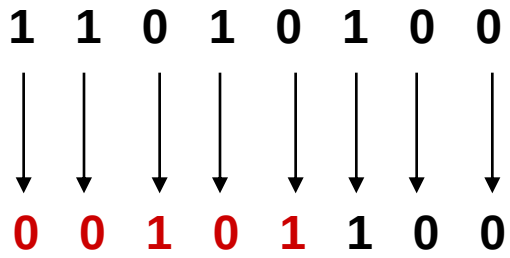
CA1(01000101)= (10111010)

CA2(01000101)=(10111010)+ 1 = (10111011)

- **Remarque 1 :**

Pour trouver le compétemment 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.





Méthode 3: représentation par Cà2

- Dans cette représentation , le bit du poids fort nous indique le signe (0 : positif , 1 : négatif).
- Le complément à deux du complément à deux d'un nombre est égale au nombre lui même .

CA2(CA2(N))= N

- Exemple :

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.
- Valeur = - CA2(101010)

= - (010101 + 1)

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

Si on travail sur 3 bits :

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

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

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

$$-4 \leq N \leq +3$$
$$-4 \leq N \leq +(4-1)$$
$$-2^2 \leq N \leq +(2^2-1)$$
$$-2^{(3-1)} \leq N \leq (2^{(3-1)}-1)$$

Si on travail sur n bits , l'intervalle des valeurs qu'on peut représenter en CA2 :

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

La représentation en complément à deux (complément à vrai) est la représentation la plus utilisée pour la représentation des nombres négatifs dans la machine.

4. Opérations arithmétiques en binaire

a) L'addition

0	0	1	1
+ 0	+ 1	+ 0	+ 1
0	1	1	10

Exemple:

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

b) Soustraction

0	0	1	1
- 0	- 1	- 0	- 1
0	1 1	1	0

Report (borrow)

Exemple:

```
  1 0 1 , 0
- 0 1 1 , 1
-----------
0 ← 0 0 1 , 1
```

Retenue

```
    0 0 0 1 1
-   0 1 1 0 0
-------------
1 ←   1 0 1 1 1
```

Report (retenue)

c) Multiplication

x 0	x 0	x 1	x 1
0	1	0	1
0	0	0	1

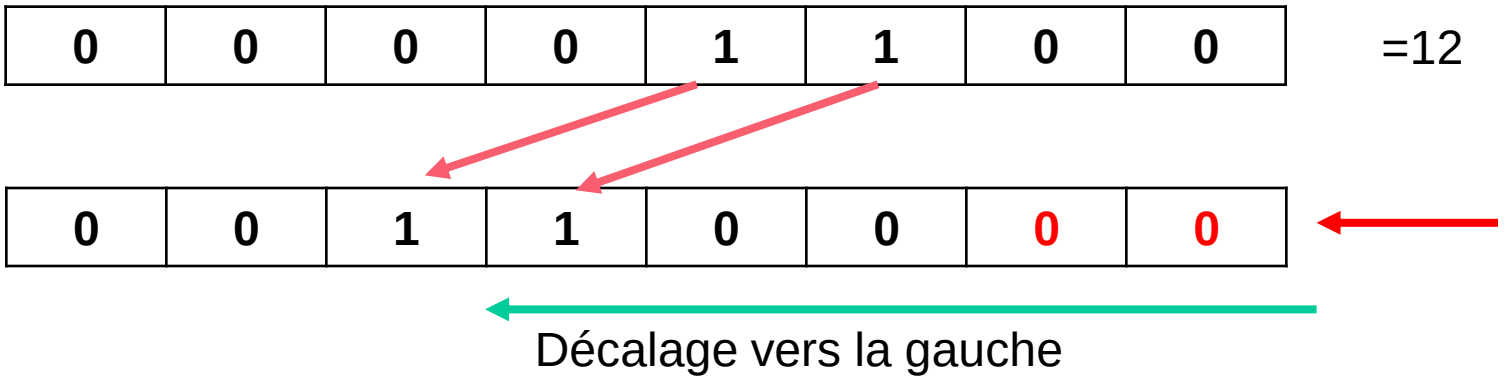
Exemple:

```
      0 1 0 1   <-- multiplicande          x  5
x     0 0 1 0   <-- multiplieur            x  2
-------------                            -----
      0 0 0 0                                
    0 1 0 1 .                                
  0 0 0 0 . .                                
0 0 0 0 . . .                                
-------------                            -----
0 0 0 1 0 1 0                               10
```

Remarque

La multiplication binaire par 2^n se traduit par un décalage de n bits vers la gauche. On introduira donc à droite n zéro.

Exemple: 12 x 4 = 12 x 2^2 ➔ donc n=2. on introduira deux zéro.

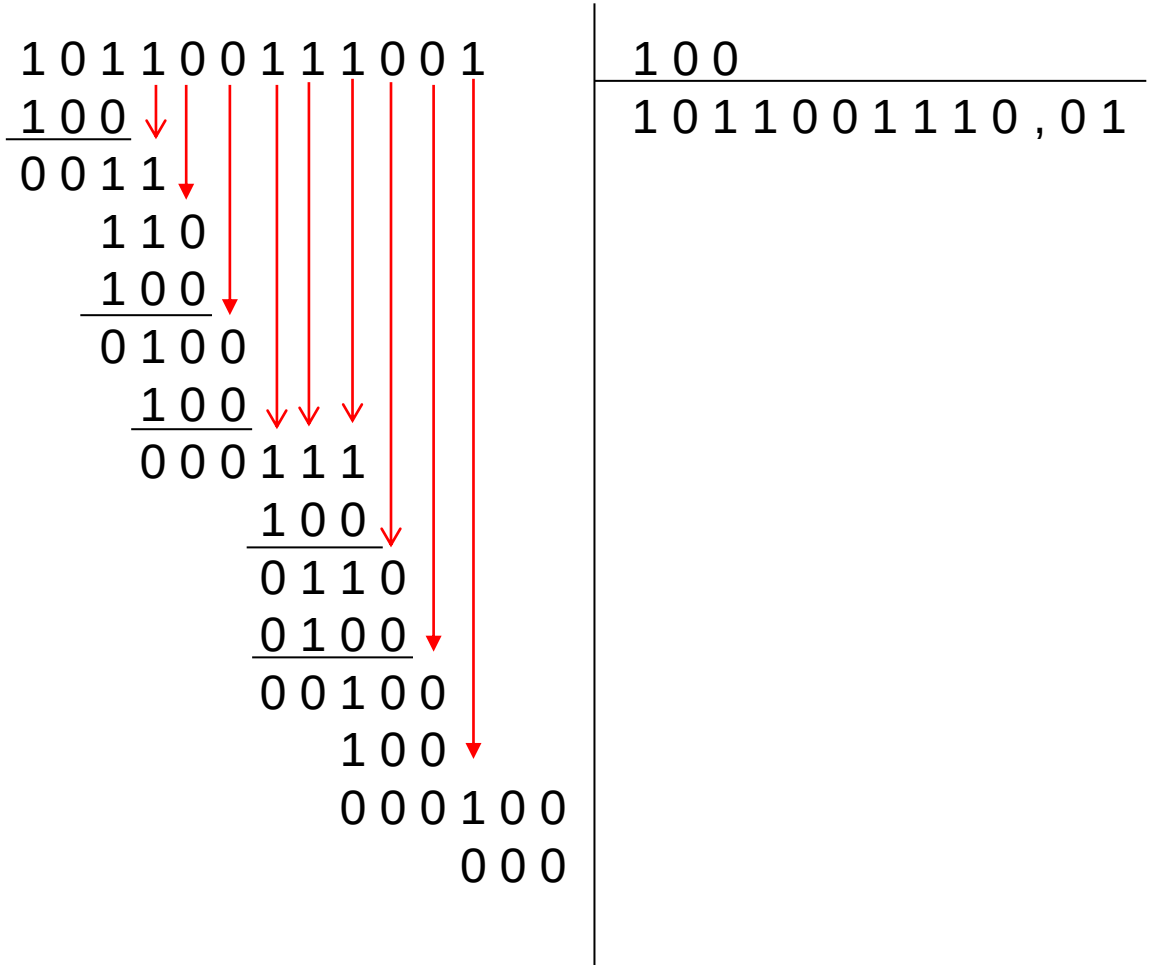


Le résultat est 48

d) Division

1	1	1	0	0	1	0	0
	1		Impossible		0		Impossible

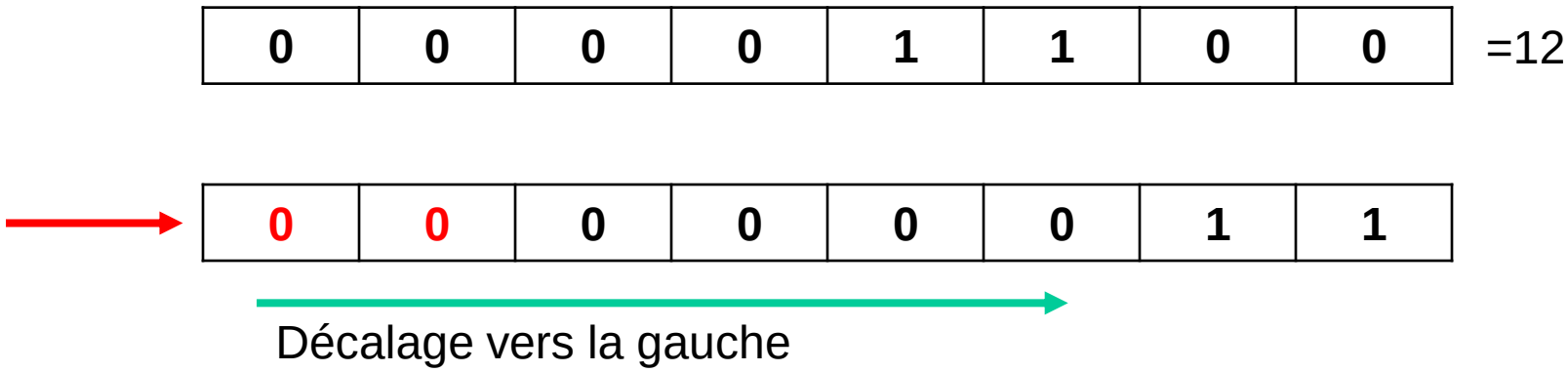
Exemple:



Remarque

La division binaire par 2^n se traduit par un décalage de n bits vers la droite. On introduira donc à gauche n zéro.

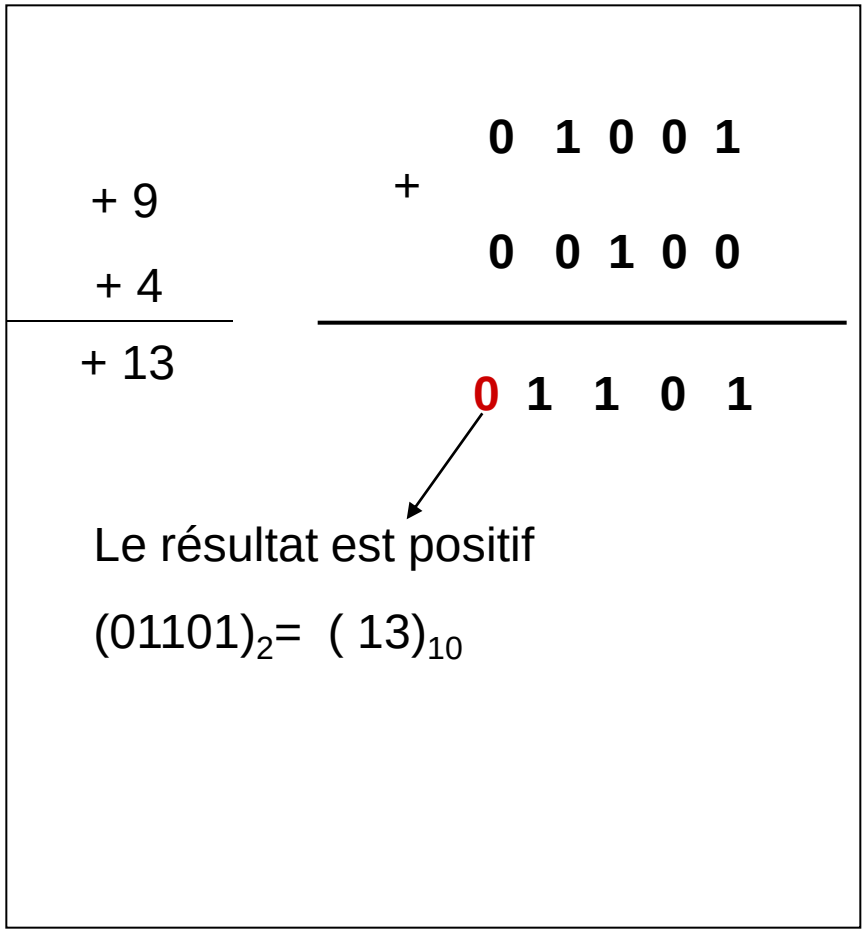
Exemple: 12 / 4 = 12 / 2^2 ➔ donc n=2. on introduira deux zéro à gauche.

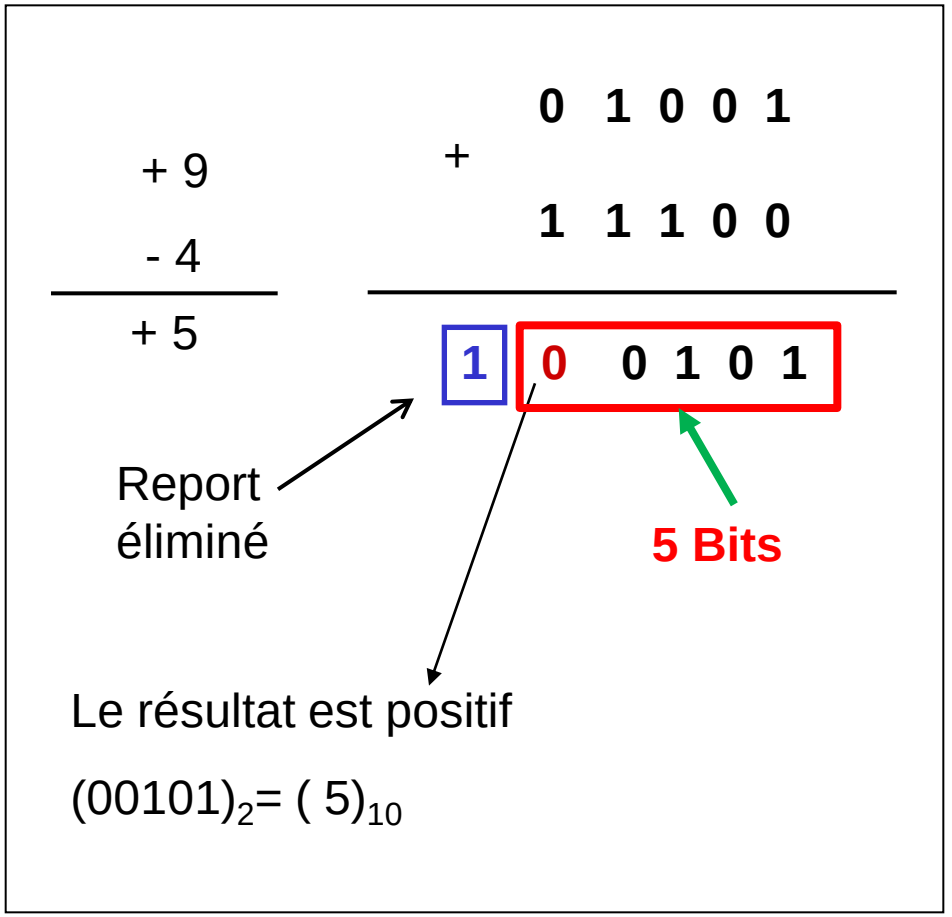


Le résultat est 03

Opérations arithmétiques en CA2

Effectuer les opérations suivantes sur **5 Bits** , en utilisant la représentation en CA2





(-9)= - Cà2 (01001) = (10111)

(-4)= - Cà2 (00100) = (11100)

- 9

- 4

- 13

1 0 1 1 1

\+

1 1 1 0 0

1 1 0 0 1 1

Report éliminé

Le résultat est négatif :

Résultat = - CA2 (10011)$_2$= -(01101)$_2$

= (- 13)$_{10}$

- 9

+ 6

+ 0

1 0 1 1 1

\+

0 0 1 1 0

1 1 1 0 1

5 Bits

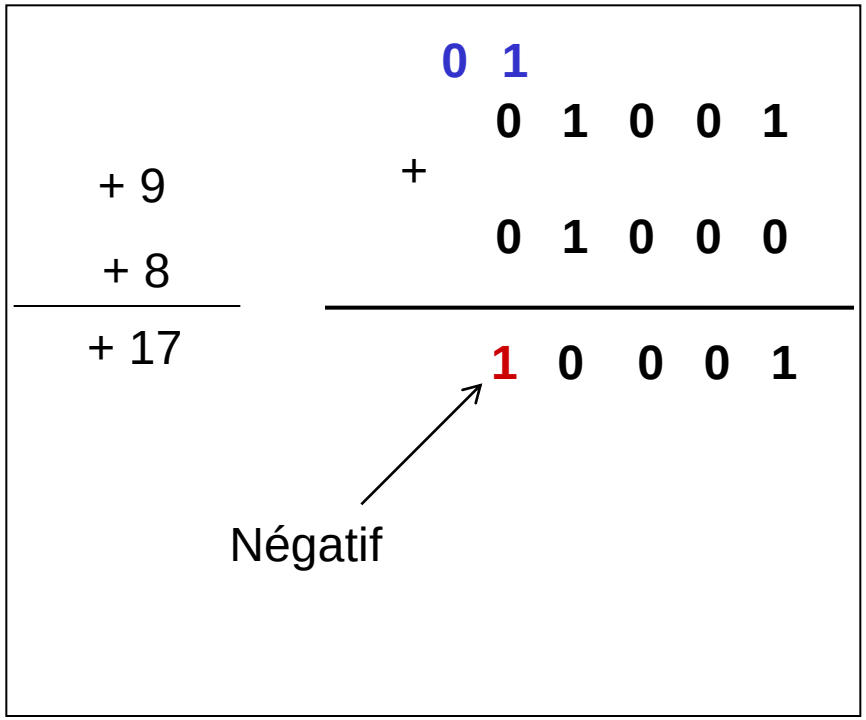
Le résultat est négatif

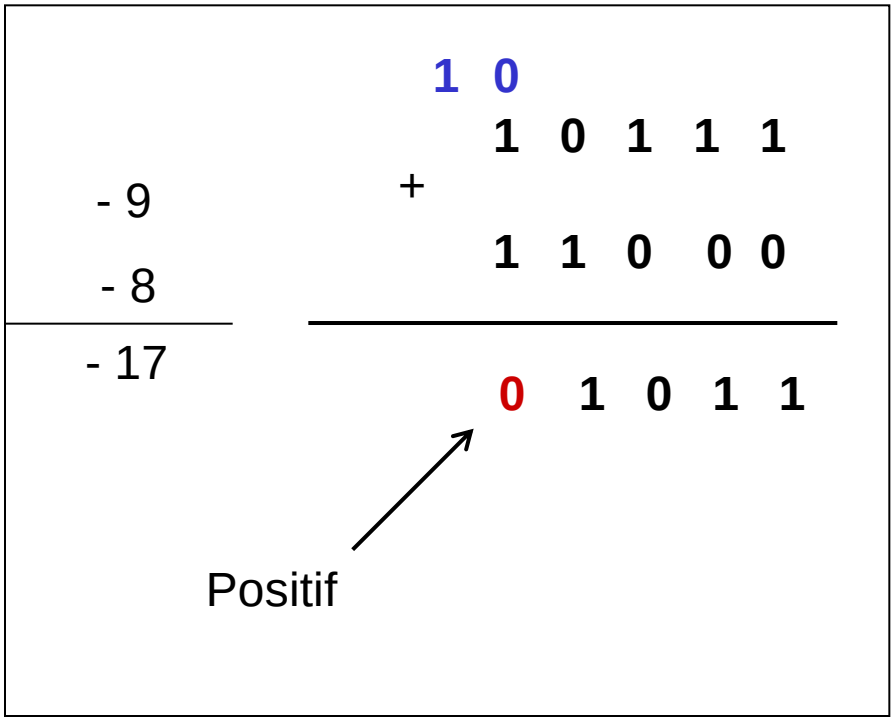
$(11101)_2$= - Cà2 $(11101)_2$= $(-3)_{10}$

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 et 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.

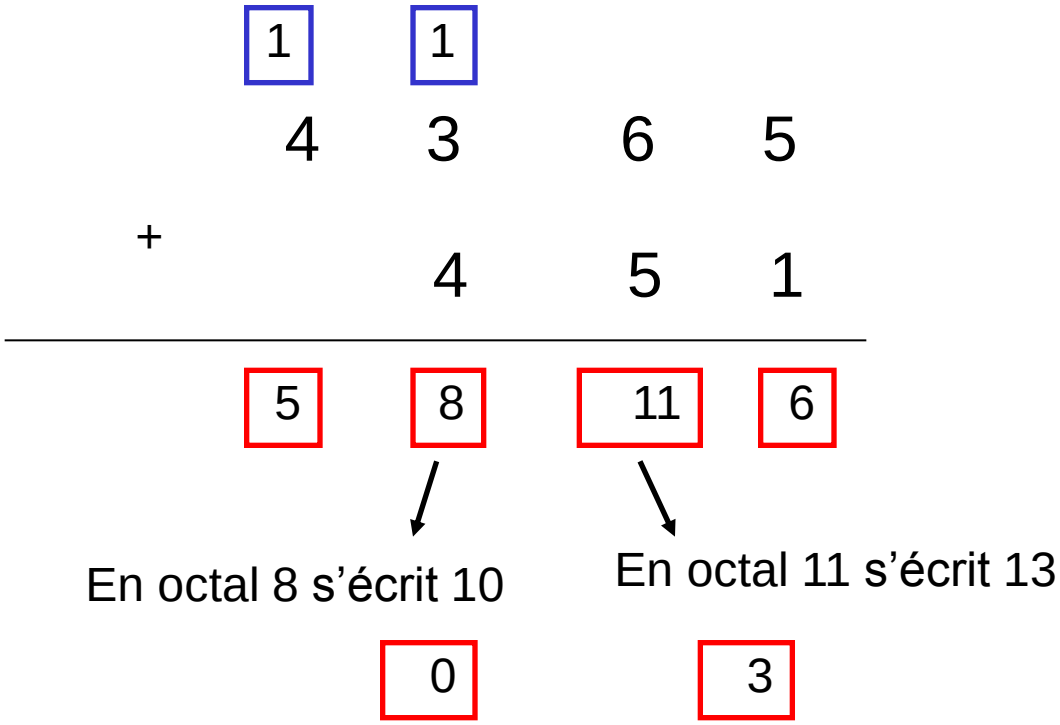
Cas de débordement





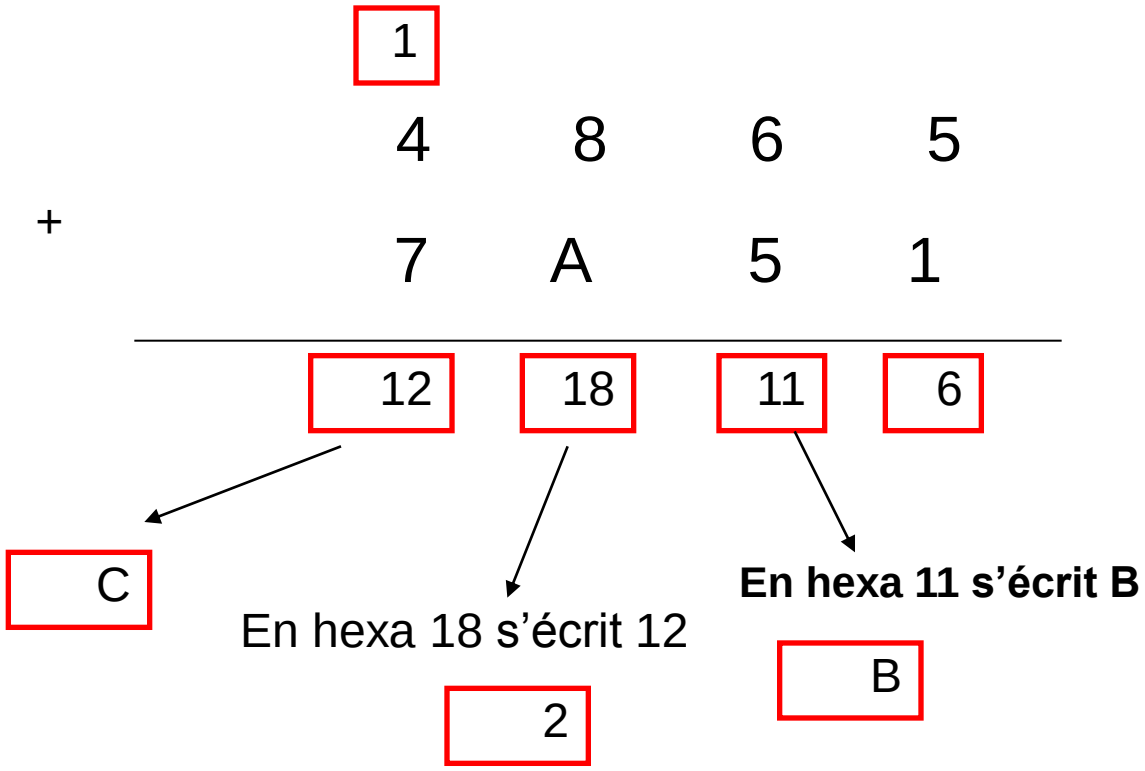
- Nous avons un débordement si la somme de deux nombres positifs donne un nombre négatif .
- Ou la somme de deux nombres négatifs donne un Nombre positif
- Il y a jamais un débordement si les deux nombres sont de signes différents.

Opérations arithmétiques en octal



Le résultat final : $(\mathbf{5036})_{\mathbf{8}}$

Opérations arithmétiques en hexadécimal



Le résultat final : $(\mathbf{C2B6})_{16}$

Exercice

- Effectuer les opérations suivantes et transformer le résultat au décimal à chaque fois:
- $(1101,111)_2+(11,1)_2=(?)_2$
- $(43)_8+(34)_8=(?)_8$
- $(43)_6+(34)_6=(?)_6$
- $(AB1)_{16}+(237)_8=(?)_{16}$

Les codes binaire

Il y a deux types de code binaire

- Le code binaire pondéré
- Le code binaire non-pondéré

Un code est dit «pondérés» quand il existe des nombres qui indiquent le poids des chiffres binaires. Le fait d'affecter des poids nous conduit à appeler le code obtenu "**code pondéré**". En multipliant ces nombres par les chiffres binaires correspondants, on obtient l'équivalence décimale. Tous les autres codes dans lesquels on ne peut repérer le poids des chiffres binaires sont appelés «**non pondérés**».

➢ Les codes binaire pondérés:

1. Le code binaire pur (naturel):

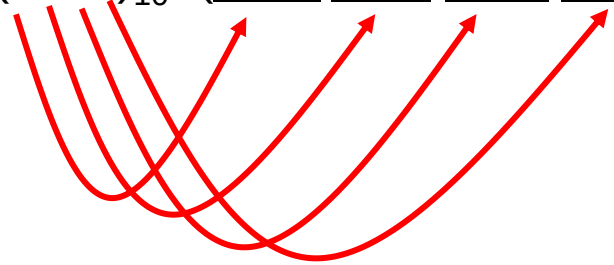
Ce code utilise l'expression naturelle du nombre en base 2 (système de numération binaire). le code binaire naturel (pur) est pondéré, les poids sont de puissance de 2: (2^0, 2^1, 2^2, 2^3, ….). Donc les déférentes puissance de 2 sont: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024.

2. Le code BCD (Binary Coded Decimal)

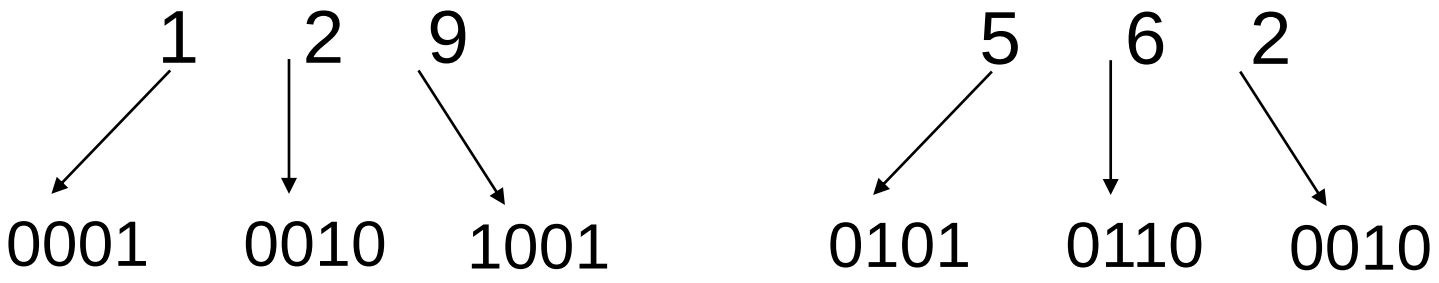
- Ce code conserve les avantages du système décimal en code binaire. Il est utilisé par les machines à calculer.
- Chaque chiffre décimal (0, 1, . . . ,9) est codé en binaire avec 4 bits. Code pondéré avec les poids 1, 2, 4, 8,10,20,40,80,... Plus facile pour coder des grands nombre, il est surtout utilisé pour l'affichage des nombres.
- Les combinaisons supérieures à 9 sont interdites (A, B, C, D, E, F)

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

Exemple: $(2763)_{10}=(\underline{0010}\ \underline{0111}\ \underline{0110}\ \underline{0011})_2$



Exemple



$129 = (0001\ 0010\ 1001)_{BCD}$ $562 = (0101\ 0110\ 0010)_{BCD}$

Remarque: Ne pas confondre BCD et code binaire pur : quand on code selon le code binaire pur on prend le nombre dans son intégralité et on le convertit ; par contre, quand on code en BCD on code chaque chiffre indépendamment les uns des autres.

$(175)_{10}=(0001\ 0111\ 0101)_{BCD}=(10101111)_2$

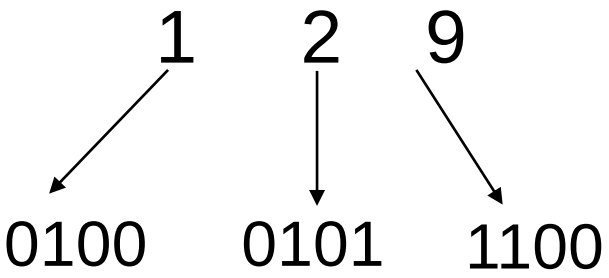
➢ Les codes binaire non pondérés:

1 Le codage EXCESS3 (BCD+3):

Le code à excès de trois (Excess 3 noté XS 3 en abrégé) s'obtient en ajoutant 3 à chaque mot-code du code BCD. Tout comme le BCD, le code à excès de 3 est un code décimal, son tableau de conversion ne concerne donc que les chiffres de 0 à 9.

Décimal	BCD+3	XS 3
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:

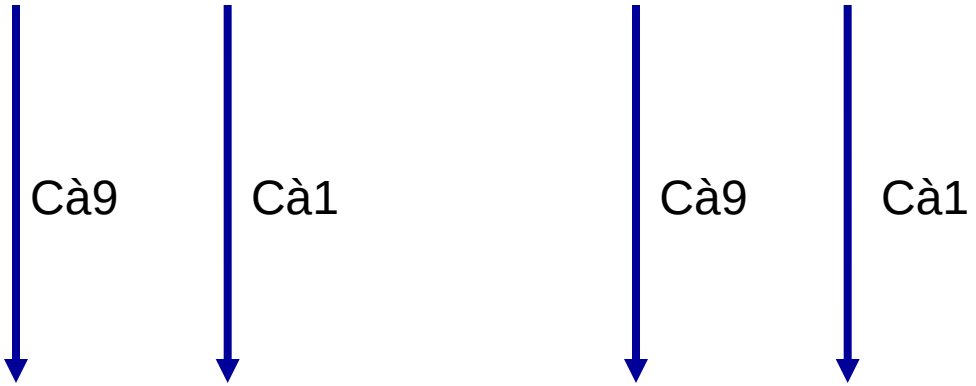


Propriété du code XS 3 :

Le code à excès de trois a été créé pour permettre la réalisation simple des opérations de soustraction. Le complément à 1 d'un mot-code représente le complément à 9 dans l'ensemble source : les codes possédant cette propriété sont appelés des codes auto-complémentaires.

Exemple:

$(5)_{10} = (1000)_{XS3}$; $(8)_{10} = (1011)_{XS3}$



$(4)_{10} = (0111)_{XS3}$; $(1)_{10} = (0100)_{XS3}$

2. Code Gray (réfléchi)

Un seul bit change entre deux nombres consécutifs (notion d'adjacence).

Ce code non pondéré est utilisé dans les tableaux de Karnaugh, dans des circuits d'entrée/sortie, et dans certains convertisseurs analogique/numérique. Il ne convient pas pour l'arithmétique binaire.

Décimal	Gray	Décimal	Gray
0	0 0 0 0	8	1 1 0 0
1	0 0 0 1	9	1 1 0 1
2	0 0 1 1	10	1 1 1 1
3	0 0 1 0	11	1 1 1 0
4	0 1 1 0	12	1 0 1 0
5	0 1 1 1	13	1 0 1 1
6	0 1 0 1	14	1 0 0 1
7	0 1 0 0	15	1 0 0 0

Une autre méthode de calcul permettant de passer d'un nombre de Gray au suivant :

- si le nombre de répétition de 1 dans le mot est pair, il faut inverser le dernier chiffre (LSB).
- si le nombre de répétition de 1 est impair, il faut inverser le chiffre situé à gauche du 1 le plus à droite.

Exemple:

- Le nombre qui suive le nombre 0110 est:

  - le nombre de 1 dans 0110 est pair, alors inverser le dernier chiffre: 0111

- Le nombre qui suive le nombre 1101 est:

  - le nombre de 1 dans 1101 est impair, alors inverser le chiffre situé à gauche du 1 le plus à droite: 1111

Codage des caractères

- Les caractères englobent : les lettres alphabétiques (A, a, B , b,..) , les chiffres (0… 9), et les autres symboles (> , ; / : ….) .
- Le codage le plus utilisé est le ASCII (American Standard Code for Information Interchange)
- Dans ce codage chaque caractère est représenté sur 8 bits (7 bits pour le caractère et un bit de parité).
- Avec 7 bits on peut avoir $2^7 = 128$ combinaisons
- Chaque combinaison représente un caractère
  - Exemple :
    - Le code 65 $(1000001)_2$correspond au caractère **A**
    - Le code 97 $(1100001)_2$ correspond au caractère **a**
    - Le code 58 $(0111010)_2$ correspond au caractère **:**
- Actuellement il existe un autre code sur 16 bits , se code s'appel **UNICODE** .

Le bit de parité est un huitième bit sert à détecter des erreurs dans la transmission des données.

Il y à deux type de parité:

1. Parité pair: égal à 1 si le nombre de 1 dans le mot-code est impaire, et égal à zéro si le nombre est pair

Exemple: 1100010 ➔ P=1 Alors 11100010

0100010 ➔ P=0 Alors 00100010

2. Parité impaire: égal à 1 si le nombre de 1 dans le mot-code est pair, et égal à zéro si le nombre est impaire

Exemple: 1100010 ➔ P=0 Alors 01100010

0100010 ➔ P=1 Alors 10100010

Exemple : changement de valeur de bit de mot $(11100010)_2=(b)_{ASCII}$ au mot $(11100011)_2=(c)_{ASCII}$ à cause des parasites et ce dernier (mot 1100011) est erroné).

Pour le reste de caractères, vous pouvez consulter le site :
www.asciitable.com