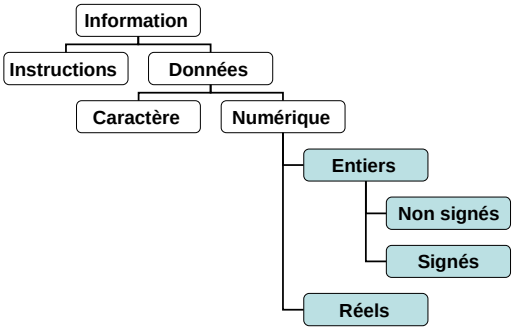
## Chapitre 2 : Représentation de l'information dans la machine

- Introduction
- Représentation des nombres négatifs
  - Signe / valeur absolue
  - Complément à 1
  - Complément à 2
- Représentation des nombres réels
  - Virgule fixe
  - Virgule flottante
- Le codage BCD et EXCESS3
- Représentation des caractères

## Introduction




## 1. Représentation des nombres entiers

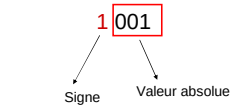
- Il existe deux types d'entiers :
  - les entiers non signés ( positifs )
  - et les entiers signés ( positifs ou négatifs )
- **Problème :** Comment indiquer à la machine qu'un nombre est négatif ou positif ?

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

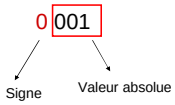
## 1.1 Représentation signe / valeur absolue ( S/VA )

- 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

**Remarque 2**

- 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 | 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 .

•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

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

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

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

## 1.3 Complément à 2 ( complément à vrai )

- Si on suppose que **a** est un nombre sur **n** bits alors :

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

et si on prend le résultat sur n bits on va obtenir la même valeur que a .

$$\boxed{a + 2^n = a}$$

Exemple : soit a = 1001 sur 4 bits

$2^4= 10000$

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

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

•Si on prend deux nombres entiers **a** et **b** sur **n** bits , on remarque que la soustraction peut être ramener à une addition : **a – b = a + (-b)**

• Pour cela il suffit de trouver une valeur équivalente à **-b** ?

$a - b = a + 2^n - b = a + (2^n - 1) - b + 1$

On a $b + CA1(b) = 2^n - 1$ donc $CA1(b) = (2^n - 1) - b$

Si on remplace dans la première équation on obtient :

$a - b = a + CA1(b) + 1$

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

**CA1(b)+1 = CA2(b)**

Et enfin on va obtenir : $a - b = a + CA2(b)$ → transformer la soustraction en une addition .

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.


**Remarque 2**

- 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 | 000 | + 0 |
| 001 | 001 | + 1 |
| 010 | 010 | + 2 |
| 011 | 011 | + 3 |
| 100 | - 100 | - 4 |
| 101 | - 011 | - 3 |
| 110 | - 010 | - 2 |
| 111 | - 001 | - 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

$-4 \leq N \leq +3$

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

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

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

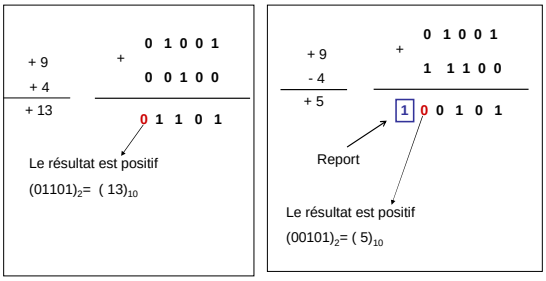
Si on travail sur n bits , l'intervalle des valeurs qu'on peut représenter en CA2 :

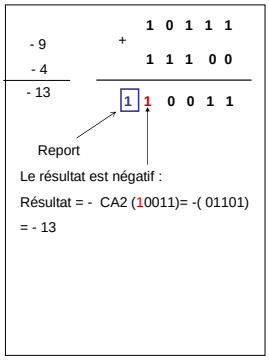
$-(2^{(n-1)}) \leq N \leq +(2^{(n-1)} - 1)$

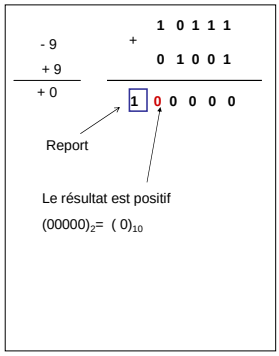
**La représentation en complément à deux ( complément à vrai ) est la représentation la plus utilisée pour la représentation des nombres négatifs dans la machine.**

## Opérations arithmétiques en CA2

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



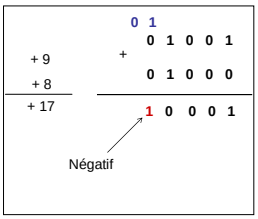


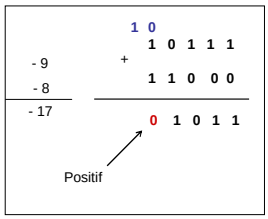




## La retenue et le débordement

- 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.

## 2. La représentation des nombres réels

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

## 2.1 La 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 | P.E | P.F | 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

## 2.2 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 :**

  $15,6 = 0,156 * 10^{+2}$

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

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

**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.

- 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 .

| Signe mantisse | Exposant | Mantisse normalisée |
|---|---|---|
| 1 bit | p bits | k bits |

- Pour la représentation de l'exposant on utilise :
  - Le complément à deux
  - Exposant décalé ou biaisé

## Représentation de l'exposant en complément à deux

- 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 en CA2 | Mantisse normalisée |
|---|---|---|
| 1 bit | 4 bits | 8 bits |

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

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

| 0 | 1 0 1 1 | 1 1 0 1 0 0 0 0 |
|---|---|---|
| 1 bit | 4 bits | 8 bits |

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

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

| 1 | 0 1 0 0 | 1 1 0 1 0 0 0 0 |
|---|---|---|
| 1 bit | 4 bits | 8 bits |

**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

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

Exposant Biaisé = Exposant réel + Biais

## 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 normalisé : 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 |

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

Signe mantisse : négatif ( 1)
Mantisse normalisée : 0,1101000001
Exposant réel = + 4
Calculer le biais : b= $2^{4-1} = 8$
Exposant Biaisé = 4 + 8 = +12 = $( 1100)_2$

| 1 | 1100 | 1 1 0 1 0 0 0 0 0 1 0 |
|---|---|---|
| 1 bit | 4 bits | 11 bits |

## Opérations arithmétiques en virgule flottante

- Soit deux nombres réels N1 et N2 tel que
  $N1=M1*b^{e1}$ et $N2=M2*b^{e2}$
- On veut calculer N1+N2 ?
- Deux cas se présentent :
  - Si e1 = e2 alors N3= (M1+M2) $b^{e1}$
  - Si e1 <> e2 alors élevé au plus grand exposant et faire l'addition des mantisses et par la suite normalisée la mantisse du résultat.

## Exemple

- Effectuer l'opération suivante : $(0,15)_8+ (1,5)_8=(?)$ :

**$(0,15)_8$= (0,001101) = 0,1101 *$2^{-2}$**
**$(1,5)_8$= (001, 1 01) = 0,1101 *$2^1$**
**$(0,15)_8+ (1,5)_8$ = 0,1101 *$2^{-2}$ + 0,1101 *$2^1$**
**= 0,0001101 *$2^1$ + 0,1101 *$2^1$**
**= 0, 1110101 * $2^1$**

| 0 | 0001 | 111010 |
|---|---|---|
| 1 bit | 4 bits | 6 bits |

## Exercice

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

| Signe mantisse | Exposant biaisé (décalé) | Mantisse normalisée |
|---|---|---|
| 1 bit | 5 bits | 10 bits |

- Calculer **N2-N1** ?.

Avant de représenter les deux nombres on doit calculer le biais (décalage)
B = $2^{5-1} = 2^4 = 16$
N1 = $(-0,014)_8 = (-0,000001100)_2= (-0,1100)_2 . 2^{-5}$
ExpB= -5 + 16 = 11 = $(01011)_2$
N2 = $(0,14)_8 = (0,001100)_2= (0,1100)_2. 2^{-2}$
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)$_{16}$ |
| N2 | 0 | 01110 | 1100000000 | 0011 10 11 0000 0000 (3B00)$_{16}$ |

N2 - N1 = 0,14 – (-0,014) = 0,14 + 0,014
N2 - N1 = $(0,1100)_2. 2^{-2}$ + $(0,1100)_2 . 2^{-5}$
= $(0,1100)_2. 2^{-2}$ + $(0,0001100)_2 . 2^{-2}$
= $(0,1101100)_2. 2^{-2}$

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

N2 - N1 = $(0,1100)_2. 2^{14}$ + $(0,1100)_2 . 2^{11}$
= $(0,1100)_2. 2^{14}$ + $(0,0001100)_2 . 2^{14}$
= $(0,1101100)_2. 2^{14}$

Exposant biaisé = 14
Exposant réel = Exposant biaisé – Biais
Exposant réel = 14 – 16 = -2

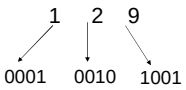
Donc on trouve le même résultat que la première opération.

## 3. Le codage BCD (Binary Coded Decimal )

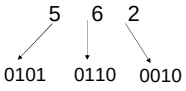
- Pour passer du décimal au binaire , il faut effectuer des divisions successives. Il existe une autre méthode simplifiée pour le passage du décimal au binaire.
- Le principe consiste à faire des éclatement sur 4 bits et de remplacer chaque chiffre décimal par sa valeur binaire correspondante .
- Les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



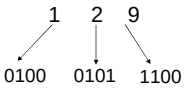

$129 = (0001\ 0010\ 1001)_2$




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

## 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 |




## 4. 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)
- 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) correspond au caractère **a**
    - Le code 58 (00111010 )correspond au caractère **:**
- Actuellement il existe un autre code sur 16 bits , se code s'appel **UNICODE** .