ECOLE SUPERIEURE POLYTECHNIQUE

Technologie des ordinateurs

Ecole Supérieure Polytechnique (ESP)

Département Génie Informatique (DGI)

Diplôme Supérieur de Technicien en Informatique (DSTI)

Licence 1

Année académique 2019-2020

Dr Birahime DIOUF , dioufbira11@gmail.com

Objectifs du cours

- Ce cours a pour objectifs de permettre à l'étudiant de pouvoir :
- Décrire la représentation des informations manipulées
  - type (les bases de numération),
  - leurs conversions, ainsi que
  - leur codage
- Maitriser le traitement des informations par les composants internes d'un ordinateur à travers :
  - l'algèbre de Boole,
  - les portes logiques et
  - les circuits logiques

Chapitre 1 : Représentation de l'information : Système de numération, Opérations et Codage

Comment sont représentées les informations dans les ordinateurs ?

Plan

- Codage de l'information
- Bases d'un système de numération
- Changement de base
- Nombres signés
- Opérations binaires (Addition/Soustraction, Multiplication, Division)
- Codes pondérés/non pondérés
- Codes de détection d'erreurs

INTRODUCTION

- Les informations traitées par les ordinateurs sont de différentes natures : nombres, texte, images, sons, vidéo, programmes, …
- Questions :
  - Comment coder une image ou un texte en un fichier ?
  - Comment représenter une couleur dans un ordinateur ?
  - Comment représenter un graphe dans un ordinateur ?
  - Comment représenter une base de données dans un ordinateur ?
- L'information analogique est supportée par des grandeurs physiques telles qu'une tension ou une intensité électrique susceptible de varier de façon continue (suite de modifications arbitrairement petites de leurs valeurs)
- A l'opposé, l'information digitale ou logique est fondamentalement discontinue. Son support élémentaire est système à n états d'équilibre. Chaque état d'équilibre correspond à une valeur de l'information appelé digit.
- Dans un ordinateur, les informations sont représentées sous forme binaire (BIT : Binary digIT) une suite de 0 et 1.
- Il existe plusieurs variantes de codes binaires, ayant chacune leurs avantages et leurs inconvénients, et possédant des propriétés utilisées dans des applications spécifiques (calcul numérique, capteurs de position, mise en mémoire, détection et correction d'erreurs en transmission, ….).
- On distingue deux grandes catégories de codes suivant qu'il est possible ou non d'attribuer une signification à chacun des digits du code.
- Dans le premier cas, on parle de code pondéré et dans le second, le code est dit code non pondéré.

CODAGE DE L'INFORMATION

Système de numération

- Un système de numération décrit la façon avec laquelle les nombres sont représentés.
- Chaque représentation d'un nombre est caractérisée par la base dans laquelle elle est faite.
- La base d'un système de numération est le nombre de chiffres différents utilisés..
- Les nombres tels que nous les utilisons sont, en réalité, une convention d'écriture.
- En numérique, les systèmes les plus utilisés sont : le système binaire (base 2), le système octal (bas 8), le système décimal (base 10) et le système hexadécimal (base 16).

- Nous allons noter $(N)_b$ pour désigner le nombre N écrit en base b.

$$N = a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \cdots + a_1 b + a_0$$

- Où
- i est le *rang* et b^i le *poids* du chiffre α_i
- $\boldsymbol{b}$ est la base du système de numération, $\boldsymbol{a_i}$ le chiffre de rang i, $\boldsymbol{n}$ le rang du chiffre de poids fort

CODAGE DE L'INFORMATION

Bases d'un système de numération

Base décimal (base 10)

- C'est le système que nous utilisons tous les jours.
- Il comprend 10 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.
- De façon générale, tout nombre décimal N entier de n chiffres s'écrit :

$$(N)_{10} = (\alpha_{n-1} \dots \alpha_1 \alpha_0)_{10} = \alpha_{n-1} \cdot 10^{n-1} + \cdots + \alpha_1 \cdot 10^1 + \alpha_0 \cdot 10^0$$

- i est le *rang* et 10^i le *poids* du chiffre α_i
- Tout nombre décimal N à virgule de n chiffres pour la partie entière et m chiffres pour la partie flottante s'écrit :

$$(N)_{10} = (\alpha_{n-1} \dots \alpha_0 \, , \alpha_{-1} \dots \alpha_{-m})_{10} = \alpha_{n-1} \cdot 10^{n-1} + \cdots + \alpha_0 \cdot 10^0 + \alpha_{-1} \cdot 10^{-1} + \cdots + \alpha_{-m} \cdot 10^{-m}$$

- Exemple : $\mathrm{N} = (2538)_{10} = 2 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 8 \cdot 10^0$

$$\mathrm{M} = (297{,}45)_{10} = 2 \cdot 10^2 + 9 \cdot 10^1 + 7 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}.$$

CODAGE DE L'INFORMATION

Bases d'un système de numération

Base binaire (base 2)

- Ce système comprend 2 chiffres appelés bits : 0 et 1.
- Exemple : $N = (10001)_2$
- **Remarque** : Quelques conversions
  - $1\ \text{octet}\ (1\text{o}) = 8\ \text{bits}, 1\ \text{Ko} = 1024\ \text{octets}, 1\ \text{Mo} = 1024\ \text{Ko}, 1\ \text{Go} = 1024\ \text{Mo}$
- **NB :**
  - le bit de poids le plus fort est appelé MSB (*Most Significant Bit*) et
  - le bit de poids le plus faible est appelé LSB (*Least Significant Bit*).

Base octal (base 8)

- Il comprend 8 chiffres : 0, 1, 2, 3, 4, 5, 6, 7.
- Exemple : $N = (273)_8$

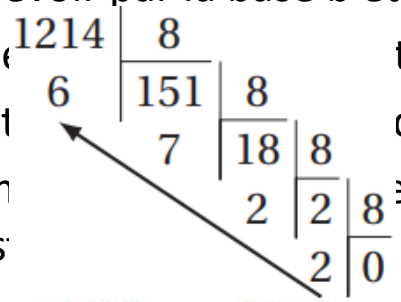
Base hexadécimal (base 16)

- Il est composé de 16 symboles (10 chiffres et 6 lettres) : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F.
- Exemple : $N = (\text{F4C})_{16}$

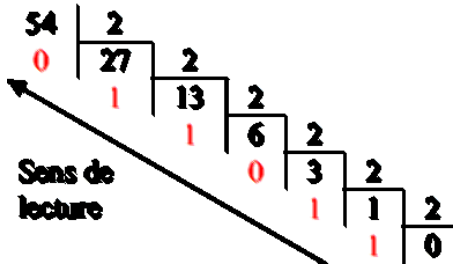
CODAGE DE L'INFORMATION

Changement de base

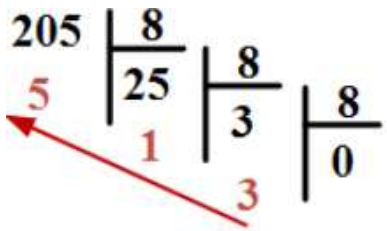
- C'est la conversion de la représentation d'un nombre dans une base donnée vers la représentation de ce même nombre dans une autre base.
- Conversion base 10 vers autres bases (2, 8, 16)
  - On divise le nombre décimal à concevoir par la base b et on conserve le reste.
  - Le quotient obtenu est divisé par b e… te.
  - Répéter l'opération sur chaque quot… ce qu'il soit nul.
  - Les restes successifs sont écrits, en… dernier, de la gauche vers la droite pour former l'expression de N dans le sys…
  - Exemple : Effectuer les conversions s…



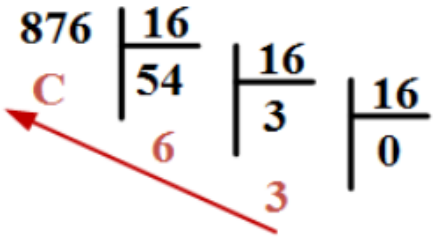
$1214_{(10)} = 2276_{(8)}$



$N = (54)_{10} = (110110)_2$



$N = (205)_{10} = (315)_8$



$N = (876)_{10} = (36C)_{16}$

CODAGE DE L'INFORMATION

Changement de base

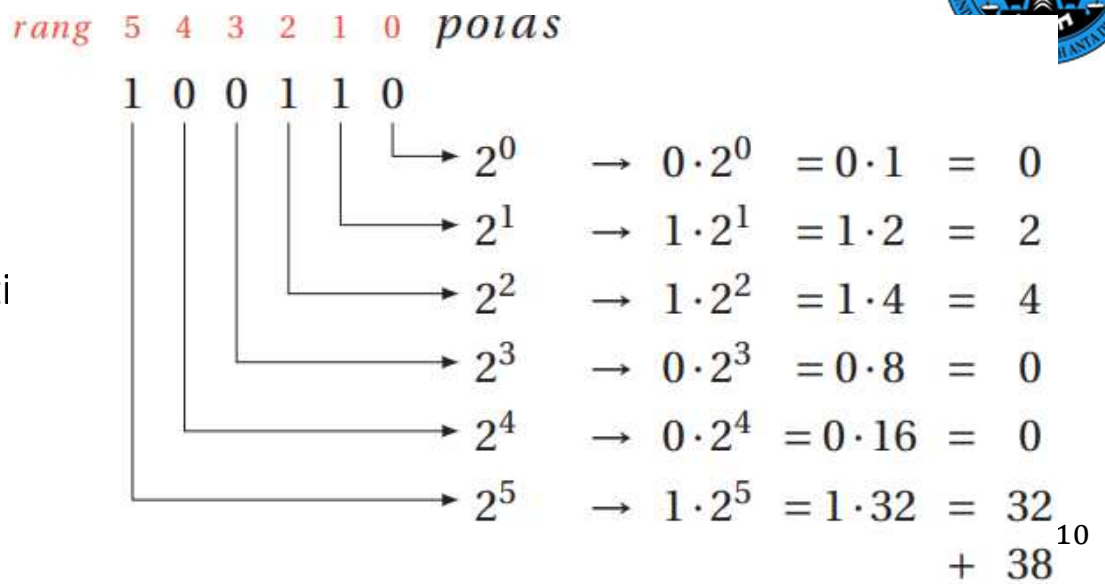
- Conversion base 10 vers autres bases (2, 8, 16)
  - Conversion d'un nombre décimal à virgule en binaire
  - Exemple : $15{,}625_{(10)} = (\ ?\ )_{(2)}$

Conversion partie entière	Conversion partie fractionnaire
15 / 2 = 7 Reste 1 ↑ 7 / 2 = 3 Reste 1 ↑ 3 / 2 = 1 Reste 1 ↑ 1 / 2 = 0 Reste 1 ↑	0,625 * 2 = 1,25 = 1 + 0,25 ↓ 0,25 * 2 = 0,5 = 0 + 0,5 ↓ 0,5 * 2 = 1 = 1 + 0 ↓
$15{,}625_{10} = 1111{,}101_2$	

CODAGE DE L'INFORMATION

Changement de base

- Conversion des bases (2, 8, 16) vers base 10
  - Pour retrouver le nombre décimal, il suffit
    chiffre appartenant au système de numérati
    au rang de ce chiffre.
- Exemple : Effectuer les conversions suivantes



- binaire vers en décimal

$$(1101{,}011)_2 = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} + 1.2^{-3} = (13{,}375)_{10}$$

- Base 8 vers base 10

$$N = (6543)_8 = 6 \cdot 8^3 + 5 \cdot 8^2 + 4 \cdot 8^1 + 3 \cdot 8^0 = (3427)_{10}$$

- Base 16 vers base 10

$$N = (AC53)_{16} = C \cdot 16^2 + 5 \cdot 16^1 + 3 \cdot 16^0 = 12 \cdot 16^2 + 5 \cdot 16^1 + 3 \cdot 16^0 = (3155)_{10}$$

CODAGE DE L'INFORMATION

Changement de base

- Table de correspondance entre les nombres des différentes base

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

CODAGE DE L'INFORMATION

Changement de base

Conversion des bases (8, 16) vers base 2

- Pour convertir un nombre octal (resp. hexadécimal) en binaire, on remplace chaque symbole du nombre écrit en base $8 = 2^3$ (resp. base $16 = 2^4$) par son équivalent écrit en binaire de 3 bits (resp. 4 bits) (voir tableau).

Exemples :

- Base 8 vers base 2

$$N = (257)_8 \Rightarrow N = (\underbrace{010}_{2}\ \underbrace{101}_{5}\ \underbrace{111}_{7})_2$$

- Base 16 vers base 2

$$N = (ECA)_{16} = (\underbrace{1110}_{E}\ \underbrace{1100}_{C}\ \underbrace{1010}_{A})_2$$

CODAGE DE L'INFORMATION

Changement de base

Conversion base 2 vers les bases (8, 16)

- Pour convertir un nombre binaire en un nombre octal (base $8 = 2^3$), on regroupe les 1 et 0 du nombre par 3 en commençant par la droite, puis chaque groupe est remplacé par le chiffre octal correspondant.
- Pour convertir un nombre binaire en base $16 = 2^4$, on fait des regroupements de 4 bits.

Exemples :

- Base 2 vers base 8

$$N = (11001101111)_2 = (\underbrace{011}_{3}\ \underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{111}_{7})_2 \Longrightarrow N = (3157)_8$$

- Base 2 vers base 16

$$N = (100001101111)_2 = (\underbrace{1000}_{8}\ \underbrace{0110}_{6}\ \underbrace{1111}_{F})_2 \Longrightarrow N = (86F)_{16}$$

CODAGE DE L'INFORMATION

Changement de base

Conversion entre base 8 et base 16

- Pour convertir un nombre octal en nombre hexadécimal, on le convertit d'abord le nombre octal en binaire puis, le nombre binaire est converti en hexadécimal.

Exemples :

- Base 8 vers base 16

$$(456)_8 = (\underbrace{100}_{4}\ \underbrace{101}_{5}\ \underbrace{110}_{6})_2 = (\underbrace{0001}_{1}\ \underbrace{0010}_{2}\ \underbrace{1110}_{E})_2 = (12\mathrm{E})_{16}$$

- Base 16 vers base 8

$$(A1\mathrm{B})_{16} = (\underbrace{1010}_{A}\ \underbrace{0001}_{1}\ \underbrace{1011}_{B})_2 = (\underbrace{101}_{5}\ \underbrace{000}_{0}\ \underbrace{011}_{3}\ \underbrace{011}_{3})_2 = (5033)_8$$

Représentation des nombres dans les machines numéri

Représentation des nombres

- Les systèmes logiques sont constitués de mécanismes qui ne permettent de noter que 2 états : « 0 » ou « 1 ». Une mémoire élémentaire est donc une unité contenant « 0 » ou « 1 ». Plusieurs de ces unités sont assemblées pour représenter un nombre binaire.
- Exemple : Mémoire à 8 bits
- Ces mémoires sont indissociables et l'ordre d'assemblage donne le poids de chaque bit.
- On a :
  - Représentation des nombres entiers positifs
  - Représentation binaire des entiers signés
  - Représentation des nombres réels dans un calculateur

Représentation des nombres dans les machines numéri

Représentation des nombres entiers signés : Nombres signés et complémentation

- La plupart des dispositifs numériques traitent également des nombres négatifs.
- Deux symboles complémentaires sont pris en compte : $+$ et $-$.
- Un bit de signe indique si le nombre est négatif ou positif : **0** représente le signe $+$ et **1** celui $-$.
- Exemple : $+(47)_{10} = \mathbf{0}0101111$ et $-(47)_{10} = \mathbf{1}0101111$
- Complément restreint ou complément à 1 (C1)
  - Pour prendre l'inverse d'un nombre, il suffit d'inverser tous ses bits (0 remplacé par 1 et 1 par 0).
  - Exemple : -47 s'écrit $\mathbf{1}0101111$ en notation exacte et son C1 est $\mathbf{1}1010000$
  - **Problème** : de nouveau, on a deux représentations différentes pour le zéro.
- Complément vrai ou complément à 2 (C2)
  - C'est la représentation la plus utilisée. Pour obtenir le C2 d'un nombre binaire, il faut prendre le C1 de ce nombre et lui ajouter 1.
  - Exemple : $-(47)_{10} = \mathbf{1}0101111$ son C1 est $\mathbf{1}1010000$, son C2 s'écrit $\mathbf{1}1010001$

Représentation des nombres dans les machines numéri

Représentation des nombres réels dans un calculateur

- Dans un calculateur, un nombre est toujours écrit sous forme d'un bloc de n éléments binaires (considéré comme un entier N).
- Pour représenter les nombres fractionnaires il est nécessaire de définir la position de la virgule.
- Pour ce faire, il existe deux méthodes :
  - la représentation en virgule fixe ;
  - la représentation en virgule flottante : virgule flottante simplifié et la représentation IEEE 754

Représentation des nombres dans les machines numéri

Représentation des nombres réels dans un calculateur

- Représentation en virgule fixe
- La virgule est toujours à une position donnée K (virgule au rang K).
- La valeur N écrite en mémoire aura les poids suivants :

$$2^{N-1-K}, \dots, 2^{0}, 2^{-1}, \dots, 2^{-K} \qquad 0 \leq N \leq (2^{n}-1)2^{-K}$$

- Exemples :
  - Nombres non signés :
    - $(56,3125)_{10} = (111000,0101)_{2}$ est représenté par

0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0

partie supra-unitaire | partie sub-unitaire

  - Nombres signés :
    - $(+56,3125)_{10} = (+111000,0101)_{2}$;

0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0

    - $(-56,3125)_{10} = (-111000,0101)_{2}$;

1	1	0	0	0	1	1	1	1	0	1	1	0	0	0	0

  - **NB** : les valeurs sont notées en C2 en mémoire

- Inconvénients :
  - Problème de gestion de la virgule dans les multiplications ;
  - Utilisation d'un grand nombre de bits de part et d'autre de la virgule pour représenter des grandeurs très faibles et des grandeurs très importantes.

Représentation des nombres dans les machines numéri

Représentation des nombres réels dans un calculateur

- Représentation en virgule flottante
- Principe : le nombre N est représenté sous la forme :

exposant	mantisse

- **1ère Approche** :
- Soit $N = a_3a_2a_1a_0,\ a_{-1}a_{-2}a_{-3}$, peut se noter

$$\underbrace{(a_6a_5a_4a_3a_2a_1a_0)}_{\text{mantisse }(m)} \cdot \underbrace{2^{-3}}_{\exp\,(e)}$$

- m et e sont notées en C2 en mémoire
- Exemple : Soit la mémoire de taille :
  - 12 bits pour m et 4 bits pour e
  - $26{,}75_{10} = 11010{,}110_2 = (11010110)_2 \cdot 2^{-3}$

$$\Rightarrow \begin{cases} e = -3 \\ m = 11010110 \end{cases}$$

1101	000011010110

- Remarque :
  - Les ordinateurs utilisent cette représentation avec 32 bits pour la mantisse et 8 bits pour l'exposant. En général, on utilise la représentation inverse, avec le bit le plus à gauche égal à 1, soit une mantisse normalisée $\Rightarrow 0{,}5 \leq M < 1$.

- **2ème Approche** : inverse de la précédente
- Bit le plus à gauche de la mantisse a pour poids 2^{-1}
- Soit $N = a_3a_2a_1a_0,\ a_{-1}a_{-2}a_{-3}$: N peut se noter :

$$\underbrace{(0, a_{-1}a_{-2}a_{-3}a_{-4}a_{-5}a_{-6}a_{-7})}_{\text{mantisse}} \cdot \underbrace{2^4}_{\exp}$$

- Exemple : même exemple
  - $26{,}75_{10} = 11010{,}110_2 = (0{,}11010110)_2 \cdot 2^5$

0101	110101100000

Représentation des nombres dans les machines numéri

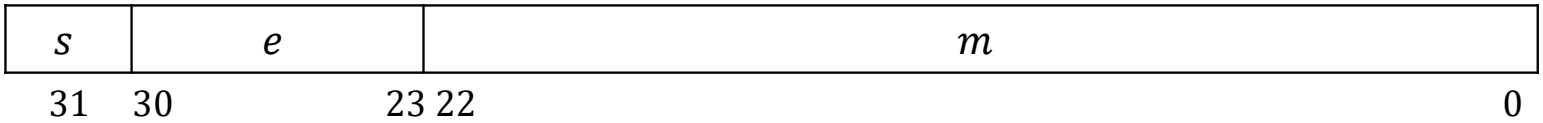
Représentation des nombres réels dans un calculateur

- Représentation IEEE 754
- Définit 3 formats :
  - simple précision sur 32 bits (1 bit : signe, 8 bit : exposant en code relatif à 127 et 23 bits : mantisse),
  - double précision sur 64 bits (1 bit : signe, 11 bit : exposant en code relatif à 1023 et 52 bits : mantisse) et
  - représentation intermédiaire sur 80 bits (principalement utilisée en interne par les processeurs pour minimiser les erreurs d'arrondi).

Représentation des nombres dans les machines numéri

Représentation des nombres réels dans un calculateur

- Représentation IEEE 754 : simple précision
- Format :
  - $N = 1,m \times 2^{e-127}$



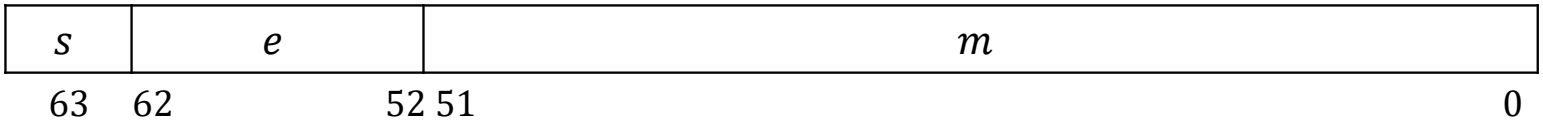
- Exemple :
  - $1\ 10000001\ 01000000000000000000000$ représente :
    - signe = 1 ⇒ nombre négatif
    - $e = (10000001)_2 = 129 \Rightarrow e - 127 = 2$
    - $m = (0{,}01)_2 = 0{,}25$
    - donc le nombre représenté est $-1{,}25 \times 2^2 = -5$.
  - $+0{,}25 = (0{,}01)$ est représenté par :
    - nombre positif ⇒ signe = 0
    - $(0{,}01)_2 = 1{,}0 \times 2^{-2} = 1{,}0 \times 2^{(125-127)}$
    - $\Rightarrow e = 125\ et\ m = 0$;
    - donc $+0{,}25$ est représenté par $0\ \ 01111101\ \ 00000000000000000000000$

NB :
Conditions à respecter pour les exposants :
• e = 00000000 est interdit.
• e = 11111111 est interdit; on s'en sert pour signaler des erreurs, on l'appelle NaN (Not a number).

Représentation des nombres dans les machines numéri

Représentation des nombres réels dans un calculateur

- Représentation IEEE 754 : double précision
- Format :
  - $N = 1, m \times 2^{e-1023}$



- Interprétation complète des codes possibles :

e	m	$représente$
0	0	± 0
0	$\neq 0$	$\pm 0, m \times 2^{-127} ou * \pm 0, m \times 2^{-1023}$
$0 < e < e_{max}$	$\forall m$	$\pm 1, m \times 2^{e-127} ou * \pm 1, m \times 2^{e-1023}$
e_{max} $(255 ou * 2047)$	0	$\pm \infty$
e_{max} $(255 ou * 2047)$	$\neq 0$	$NaN (Not a Number)$

(*simple ou double précision)

OPERATIONS

Addition binaire

- L'addition de nombres signés est utilisée dans la plupart des circuits numériques.
- La méthode consiste à écrire les nombres positifs en notation exacte et remplacer les nombres négatifs par leur complément à deux avant d'additionner.
- Si le résultat est positif, il est en notation exacte, s'il est négatif, il est en notation complément à 2.
- Addition de deux nombres positifs
  - Algorithme de l'adition binaire :

Opérations	Résultats
0 + 0	0
0 + 1	1
1 + 0	1
1 + 1	0 avec un report (retenue) de 1
1 + 1 + 1	1 avec un report de 1

  - Exemple: Effectuer les additions $(+17)_{10}$ et $(+12)_{10}$

$$(+17)_{10} \quad \mathbf{0}10001$$
$$+$$
$$(+12)_{10} \quad \mathbf{0}01100$$
$$\mathbf{0}11101$$

↑ Bit de signe

OPERATIONS

Addition binaire

Addition de deux nombres de signes contraires (soustraction)

- Algorithme de la soustraction binaire :

Opérations	Résultats
0 - 0	0
0 - 1	1 avec un report (retenue) de 1
1 - 0	1
1 - 1	0
0 - 0	0

- Exemple: Effectuer les additions

$(+17)_{10}$ $\quad$ $\mathbf{0}10001$

$+$

$(-12)_{10}$ $\rightarrow$ C2 $\quad$ $\mathbf{1}10100$

$\mathbf{1}\mathbf{0}00101$

↑ Bit de signe

↑ débordement à éliminer

Le résultat en notation exacte est $\mathbf{0}00101 = (+5)_{10}$

$(-17)_{10}$ $\rightarrow$ C2 $\quad$ $\mathbf{1}01111$

$+$

$(+12)_{10}$ $\quad$ $\mathbf{0}01100$

$\mathbf{1}11011$

↑ Bit de signe

Le C1 de $\mathbf{1}11011$ est $\mathbf{1}00100$ et son C2 est $\mathbf{1}00101$

Le résultat en notation exacte est $\mathbf{1}00101 = (-5)_{10}$

OPERATIONS

Addition binaire

Addition de deux nombres négatifs

- Exemple: Effectuons l'addition de $(-17)_{10}$ et $(-12)_{10}$.
- Les compléments à 2 s'écrivent :
- $(-17)_{10} \rightarrow \mathbf{1}01111$ et $(-12)_{10} \rightarrow \mathbf{1}10100$

$$
\begin{array}{lll}
(-17)_{10} & & \mathbf{1}01111 \\
 & & + \\
(-12)_{10} & \rightarrow C2 & \mathbf{1}10100 \\
 & & \hline
 & & \mathbf{10}00101
\end{array}
$$

↑ Bit de signe

↑ débordement à éliminer

Le résultat en notation exact est $\mathbf{1}11101 = (-29)_{10}$

OPERATIONS

Multiplication binaire

- Algorithme de la multiplication binaire :

Opérations	Résultats
0 x 0	0
0 x 1	0
1 x 0	0
1 x 1	1
0 x 0	0

- Exemple :

$$\begin{array}{r} 101101 \\ \times \quad 10011 \\ \hline 101101 \\ 101101\cdot \\ 101101\cdots \\ \hline 1101010111 \end{array}$$

- NB :
  - Les circuits numériques n'additionnent pas l'ensemble des produits partiels directement mais deux à deux : le premier avec le deuxième puis la somme obtenue avec le troisième et ainsi de suite si nécessaire.
  - Si les nombres sont négatifs, ce sont leurs compléments à 2 qui sont pris en compte avant la multiplication et le produit est en notation exacte.
  - Si l'un des nombres est négatif, on prend son complément à 2 avant la multiplication, le résultat est le complément à 2 du produit cherché.

OPERATIONS

Division binaire

- La méthode est identique à celle d'une division avec les nombres décimaux.
- Exemple :

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

```
1  0  0  1  0  0  0  0  1  1  1 | 1  0  1  1
−  1  0  1  1                   |-------------
   0  1  1  1  0                | 0  1  1  0  1  0  0  1
   −  1  0  1  1                |
      0  0  1  1  0  0          |
         −  1  0  1  1          |
            0  0  0  1  1  1  1 |
               −  1  0  1  1    |
                  0  1  0  0    |
```

- Soit : $(110110)_2 = (1010)_2 \times (101)_2 + (100)_2$ ou encore, en base 10 : $54 = 10 \times 5 + 4$.
- Si les nombres sont signés, la division s'effectue en appliquant les mêmes règles que la multiplication.

CODES

Codes pondérés

- Code DCB (Décimal Codé Binaire) ou BCD (Binary Coded Decimal)
- Le code binaire du chiffre décimal le plus élevé 9, étant 1001, mot de quatre bits, chaque chiffre en notation binaire est codé sur 4 bits.
- D'où le tableau suivant :

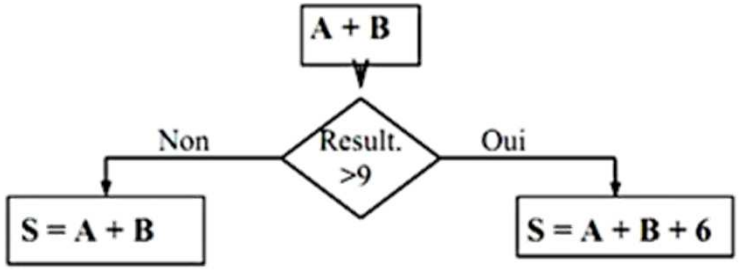
Code décimal	0	1	2	3	4	5	6	7	8	9
Code DCB	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

- Exemple : le nombre décimal 2583 s'écrit, en utilisant le code DCB
- Remarque :
  - Il y a 16 mots binaires de quatre chiffres. Le code DCB n'en retient que 10.
  - Les autres (1010 1011 1100 1101 1110 1111) ne sont pas utilisés, l'apparition de l'un d'entre eux signifie qu'une erreur s'est produite.

CODES

Codes pondérés

- Code DCB (Décimal Codé Binaire) ou BCD (Binary Coded Decimal) : Opérations
- *Algorithme:*
- **NB :** 6 est appelé **facteur de correction**



- Exemple : $(148)_{10} + (34)_{10}$ en DCB

$$\begin{array}{r} 148 \\ +\ 34 \\ \hline 182 \end{array} \qquad \begin{array}{r} 0001\ \ 0100\ \ 1000 \\ +\ \ 0000\ \ 0011\ \ 0100 \\ \hline 0001\ \ 0111\ \ 1100 \\ (1\quad\ 7\quad\ \text{"?"})_{10} \end{array}$$

$$\begin{array}{r} 148 \\ +\ 34 \\ \hline 182 \end{array} \qquad \begin{array}{r} 0001\ \ 0100\ \ 1000 \\ +\ \ 0000\ \ 0011\ \ 0100 \\ \hline 0001\ \ 0111\ \ 1100 \\ +\ \ \ \ \ \ \ \ \ \ \ \ 0110 \\ \hline 0001\ \ 1000\ \ 0010 \\ (1\quad\ 8\quad\ 2)_{10} \end{array}$$

- Pour effectuer la correction, il faut ajouter 6 soit 0110 en DCB (cela correspond au fait que l'on saute les 6 combinaisons non valide), d'où

CODES

Codes non pondérés

- Avec les codes non pondérés on n'attribue pas un poids dépendant uniquement de la position du symbole.
- Code de Gray
- Souvent, dans les conversions analogiques-numériques, on a besoin d'un code dans lequel les grandeurs successives ne différent que d'un caractère : c'est le code de Gray encore appelé « binaire réfléchi », c'est-à-dire un code pour lequel un seul bit change entre deux nombres consécutifs.
- C'est un code cyclique, c'est-à-dire que la règle s'applique entre le dernier terme et le premier.
- Les positions binaires ne sont affectées d'aucun poids.

Chiffres	Code GRAY			
	G_3	G_2	G_1	G_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

CODES

Codes non pondérés

Code de Gray :

- Conversion entre code décimal et binaire réfléchi

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

- Conversion entre code binaire naturel et binaire réfléchi

	Binaire naturel				Binaire réfléchi			
	a_3	a_2	a_1	a_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

	Binaire réfléchi				Binaire naturel			
	g_3	g_2	g_1	g_0	a_3	a_2	a_1	a_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

CODES

Codes non pondérés

Code Alphanumérique

Un ordinateur est prévu pour traiter des informations non numériques, c'est-à-dire qu'il doit « reconnaitre » des caractères de l'alphabet, des caractères spéciaux, des chiffres, … Tous ces éléments sont associés à un code appelé alphanumérique.

Code ASCII

- ASCII (*American Standard Code for Information Interchange*) est quasi universel pour la transmission de l'info.
- On le retrouve pratiquement dans tous les ordinateurs et leurs organes périphériques, pour leurs dialogues (stocker, analyser et communiquer) et la représentation des textes en mémoire.
- Il a été adopté par ISO (*International Standards Organization*) qui en a fait le code ISO-7.
- Le plus souvent ce code est défini avec 8 bits, le $8^{ème}$ étant généralement un bit de parité pour corriger des erreurs de transmission et chaque symbole codé sur 7 bits ($2^7 = 128$ caractères différentes).

D'autres codes :

- ANSI : dérivé de ASCII et produit par American National Standards Institute; les 128 1ers codes correspondent aux caractères du code ASCII et les 128 autres sont une extension qui correspond à la page de code choisie.
- EBCDIC (*Extended Binary Coded Decimal Interchange Code*) : code à 8 bits crée par IBM pour ses ordinateurs.

CODES

Codes de détection d'erreurs

- Bit de parité pair
- C'est une méthode analogue à la précédente : le bit supplémentaire est fixé à une valeur (0 ou 1) telle que le nombre total de 1 dans le « mot », y compris le bit de parité, soit pair.
- Exemples :
  - Un circuit numérique doit transmettre le caractère « , » (virgule) ;
  - le code ASCII de ce caractère est 0101100 : il est formé de trois 1, le bit de parité doit donc être 1 pour que le nombre total de 1 soit pair.
  - Le code ASCII du caractère « , », avec le bit de parité, est : 10101100.
  - Si le caractère à transmettre est « D », dont le code ASCII est 1000100, le bit de parité doit être 0 pour que le nombre total de 1 soit pair.
  - Le code ASCII du caractère « D », avec bit de parité, est : 01000100.

CODES

Codes de détection d'erreurs

- Bit de parité impair
- Le bit supplémentaire est fixé à une valeur (0 ou 1) telle que, pour chaque « mot », le nombre total de 1, y compris le bit de parité, soit impair.
- Exemples :
  - Un circuit numérique doit transmettre le caractère « , » (virgule) ;
  - le code ASCII de ce caractère est 0101100 : il est formé de trois 1, le bit de parité doit donc être 0 pour que le nombre total de 1 soit impair.
  - Le code ASCII du caractère « , », avec le bit de parité, est : 00101100.
  - Si le caractère à transmettre est « D », dont le code ASCII est 1000100, le bit de parité doit être 1 pour que le nombre total de 1 soit impair.
  - Le code ASCII du caractère « D », avec bit de parité, est : 11000100.