IUT de Colmar - Département RT
1ière année.

Numération

Laurent MURA.

SOMMAIRE

1. Les différents systèmes
2. Les différentes conversions
3. Quelques systèmes de codage
4. L ’arithmétique binaire

1. Les différentes bases

Les systèmes de numération

•Les systèmes numériques traitent des informations numériques sous forme de nombres.

•Les systèmes de numération permettent d'exprimer des nombres dans différentes bases:

- •Décimale
- •Binaire
- •Octale
- •Héxadécimale.

Généralité commune à toutes les bases: Traduction des chiffres (1)

- ***Traduction en décimal:***
  - $(9872)_{10}$, est un nombre en base dix dont les chiffres 9, 8, 7, et 2 sont appelés **digits**. Ils représentent des puissances respectives de dix.
  - Ce nombre se traduit par $9*10^3+8*10^2+7*10^1+2*10^0$ où les puissances de dix sont appelées **poids**.

Représentation générale

Digit	a3	a2	a1	a0
Base	b	b	b	b
Poids	3	2	1	0
	↓	↓	↓	↓
	$a3*b^3$	$a2*b^2$	$a1*b^1$	$a0*b^0$

Résultat

→ $a3*b^3+a2*b^2+a1*b^1+a0*b^0$

Généralité commune à toutes les bases: Traduction des chiffres (2)

- ***Traduction des chiffres à virgule en décimal:***

$(a3\ a2\ a1\ a0, a\text{-}1\ a\text{-}2)_b$

$= a3 * b^3 + a2 * b^2 + a1 * b^1 + a0 * b^0 + a\text{-}1 * b^{-1} + a\text{-}2 * b^{-2}$ ou encore,

Digit	a3	a2	a1	a0	a-1	a-2
Base	b	b	b	b	b	b
Poids	3	2	1	0	-1	-2
	↓	↓	↓	↓	↓	↓
	$a3*b^3$	$a2*b^2$	$a1*b^1$	$a0*b^0$	$a\text{-}1*b^{-1}$	$a\text{-}2*b^{-2}$

Résultat

→ $a3*b^3+a2*b^2+a1*b^1+a0*b^0+a\text{-}1*b^{-1}+a\text{-}2 * b^{-2}$

- $(22 + 0.8125)_{10} = (22.8125)_{10}$

Digit	(	1	0	1	1	0	,	1	1	0	1	)	2
Poids		16	8	4	2	1		1/2	1/4	1/8	1/16		

Résultat	16		4	2			0,5	0,25		0,0625

Généralité commune à toutes les bases: Comptage

•Le comptage dans une base se fait en plaçant le digit de poids faible à 0. Puis il est incrémenté jusqu'à atteindre la valeur de la base moins 1. La valeur suivante est alors atteinte en incrémentant le digit de poids immédiatement supérieur. Et ainsi de suite…

•Une base exprimée dans son propre système s'écrit toujours $(10)_b$.

Exemple: ***Comptage binaire:***

a_2	a_1	a_0	a_2	a_1	a_0	a_2	a_1	a_0
0	0	0	0	1	1	1	1	0
0	0	1	1	0	0	1	1	1
0	1	0	1	0	1			

Le système binaire (1)

Base 2

Digits 0,1

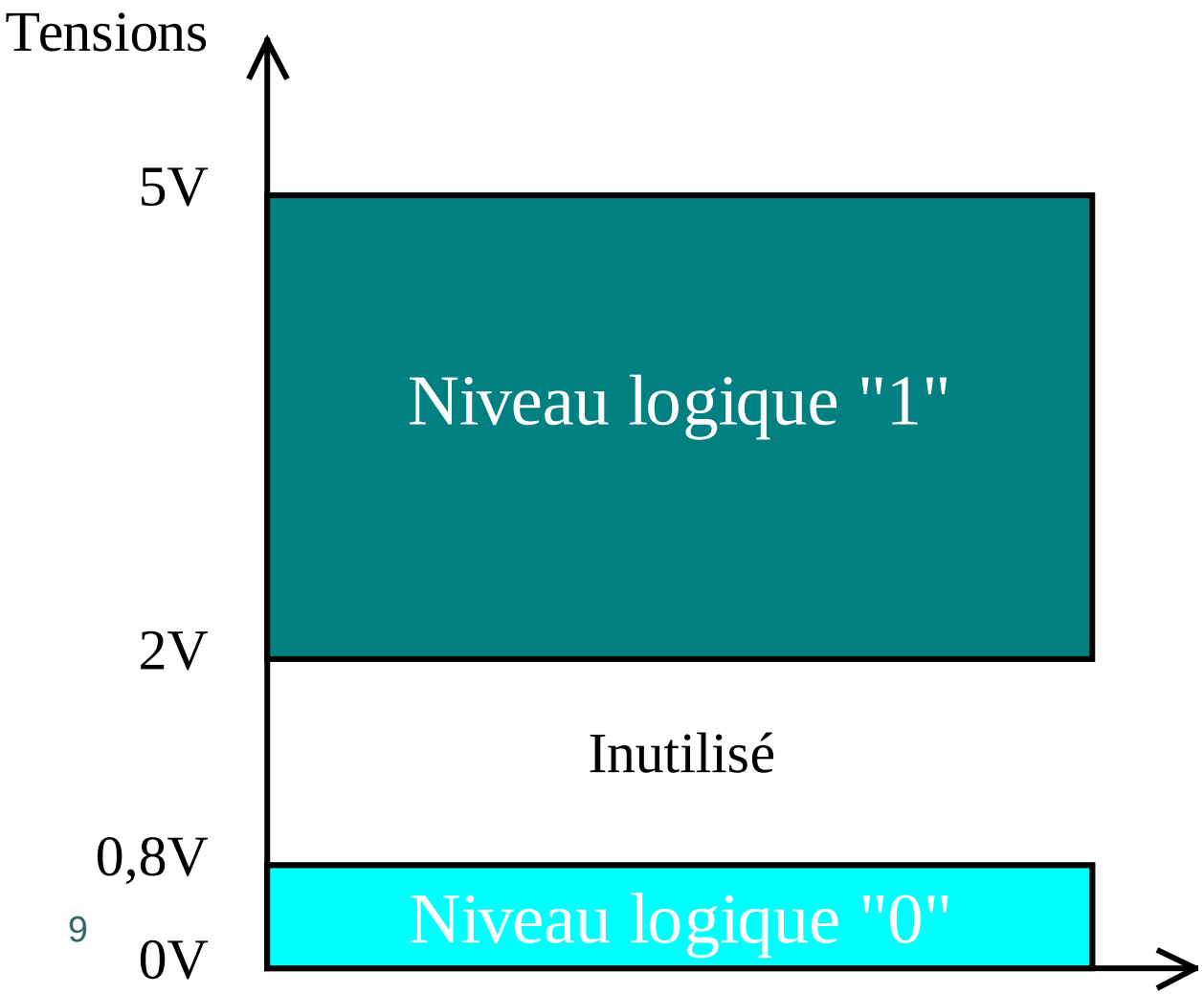
Poids 1, 2, 4, 8, 16, 32, 64, 128, 256, …

Comme ce système n'est constitué que de deux digits, il est la base des systèmes numériques.8 Ce système permet de représenter les éléments d'une logique à deux états (vrai ou faux). En base 2 un digit porte le nom particulier de **bit** qui est une contraction de binary digit.

Exemple

Dans les systèmes électroniques de commutation, les valeurs suivantes sont généralement admises (sauf spécifications contraires des constructeurs) pour les **niveaux logiques** 0 et 1.

Le système binaire (2)



Le système octal

Base 8

Digits 0,1,2,3,4,5,6,7

Poids 1, 8, 64, 512, 4096, …

Le système octal était utilisé mais tombe à présent en désuétude. Il offre néanmoins des particularités intéressantes quant aux possibilités de conversion avec les autres bases.

Le système décimal

Base 10

Digits 0,1,2,3,4,5,6,7,8,9

Poids 1, 10, 1000,10000, …

C'est notre système numérique "natif". Rien de spécial à en dire sauf qu'il faudra pouvoir s'en passer (ne pas y revenir sans cesse pour interpréter des résultats de calculs dans d'autres bases).

Le système hexadécimal

Base 16

Digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Poids 1, 16, 256, 4096, 65536, …

C'est avec le système binaire les deux systèmes les plus utilisés dans les systèmes numériques.

En effet, les entités manipulées par les ordinateurs étant des octets, ensemble de huit bits, une façon très commode et condensée de les représenter est de découper les octets en deux "quartés", dont les valeurs peuvent alors facilement se représenter par un seul digit du système hexadécimal.

Il réalise un bon compromis pour représenter de grand nombres binaires.

Exemple

- Soit la valeur binaire suivante : $(1101010010100111)_2$.
- L'inconvénient de cette représentation est sa longueur et la lourdeur d'écriture.
- Son équivalent en hexadécimal est bien plus simple et condensé:

$$\Rightarrow (1101010010100111)_2 = (D4A7)_{16}$$

2. Les différentes conversions

Conversion binaire ⇒ décimal

- Méthode
  - Détecter le poids d'un digit
  - Multiplier ce digit par la base élevée à la puissance poids
  - Additionner ce résultat au précédent.
- Exemple
  - $(11011)_2 = 1*2^4+1*2^3+0*2^2+1*2^1+1*2^0 = 16+8+2+1 = (27)_{10}$
  - $(10110101)_2 = 1*2^7+1*2^5+1*2^4+1*2^2+1*2^0$ $=128+32+16+4+1 = (181)_{10}$

Conversion décimal ⇒ binaire: Première méthode

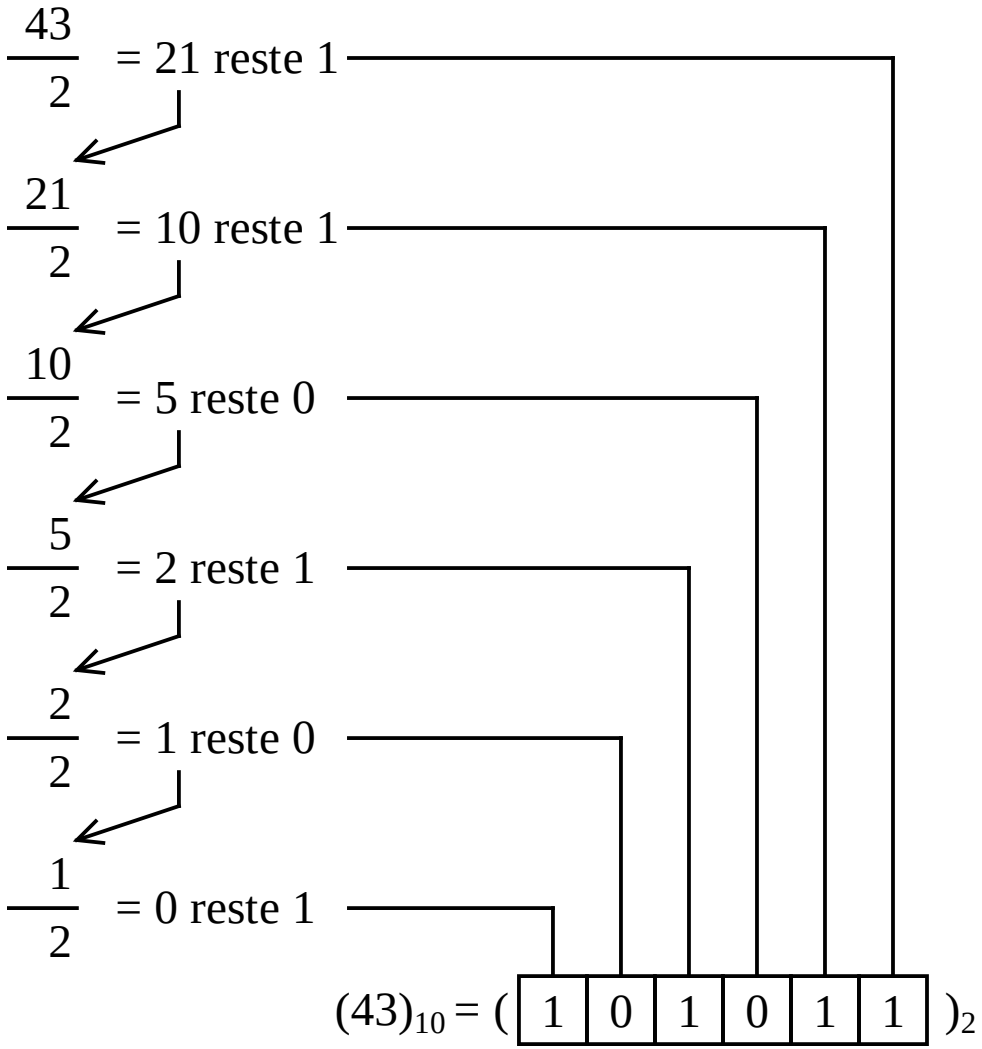
•Pour cette conversion deux méthodes existent. La première revient à faire la démarche inverse de celle énoncée auparavant. Un nombre décimal est alors exprimé en une somme de puissances de deux, puis transcrit en 0 et 1.

•Exemple

$(48)_{10} = 32+16 = (110000)_2$

Les positions qui ne sont pas utilisées sont affectées d'un zéro et celle qui le sont se voient affectées d'un un. Cette façon de faire peut convenir pour les petits nombres décimaux à convertir en binaire.

Conversion décimal ⇒ binaire: Divisions successives par 2



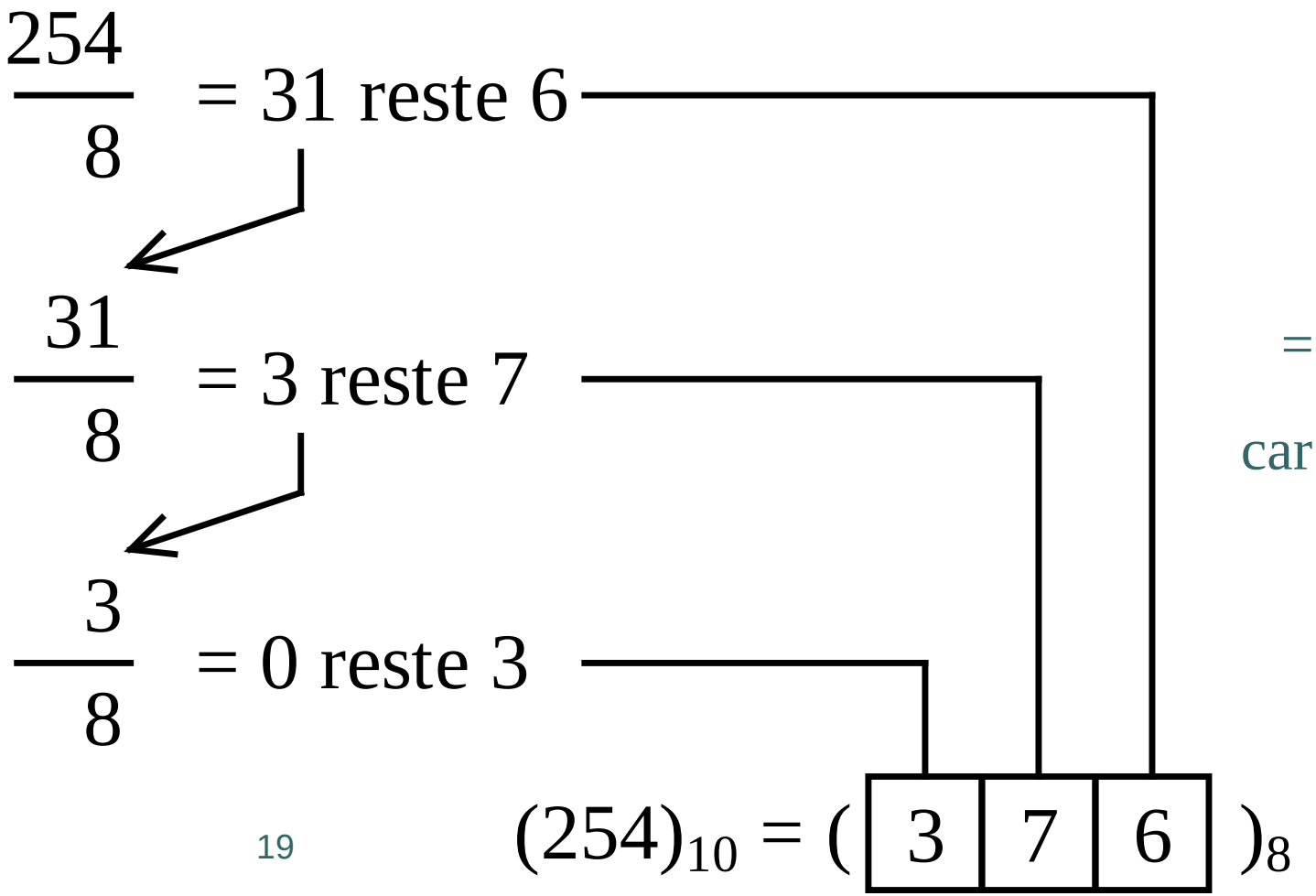
Conversion octal ⇒ décimal

•Cette conversion se fait tout à fait classiquement (comme pour toute base différente de 10).

•Exemple

$$(247)_8 = 2*8^2+4*8^1+7*8^0 = 128+32+7 = (167)_{10}$$

Conversion décimal ⇒ octal: Divisions successives par 8



254/8

= 31.75

= 31 + 6/8,

car 0.75*8 = 6

Conversion octal ⇒ binaire

•Cette conversion s'effectue simplement en convertissant chaque chiffre du nombre octal en nombre binaire:

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

•Exemple:

⇨ $(6571)_8 = (110101111001)_2$

Conversion binaire ⇒ octal

•Cette conversion se fait d'une manière juste opposée à celle que nous venons de voir.

•Exemple:

$$(100011111000010)_2 = (43702)_8$$

•Si le nombre binaire ne permet pas de former des groupes de trois bits, il suffit alors de la compléter par des zéros.

$$(1101110011)_2 = (1563)_8$$

Conversion héxadécimal ⇒ décimal

- Exemple:

$$(4305)_{16} = 4*16^3+3*16^2+5*16^0$$
$$= 16384+768+5$$
$$= (17157)_{10}$$

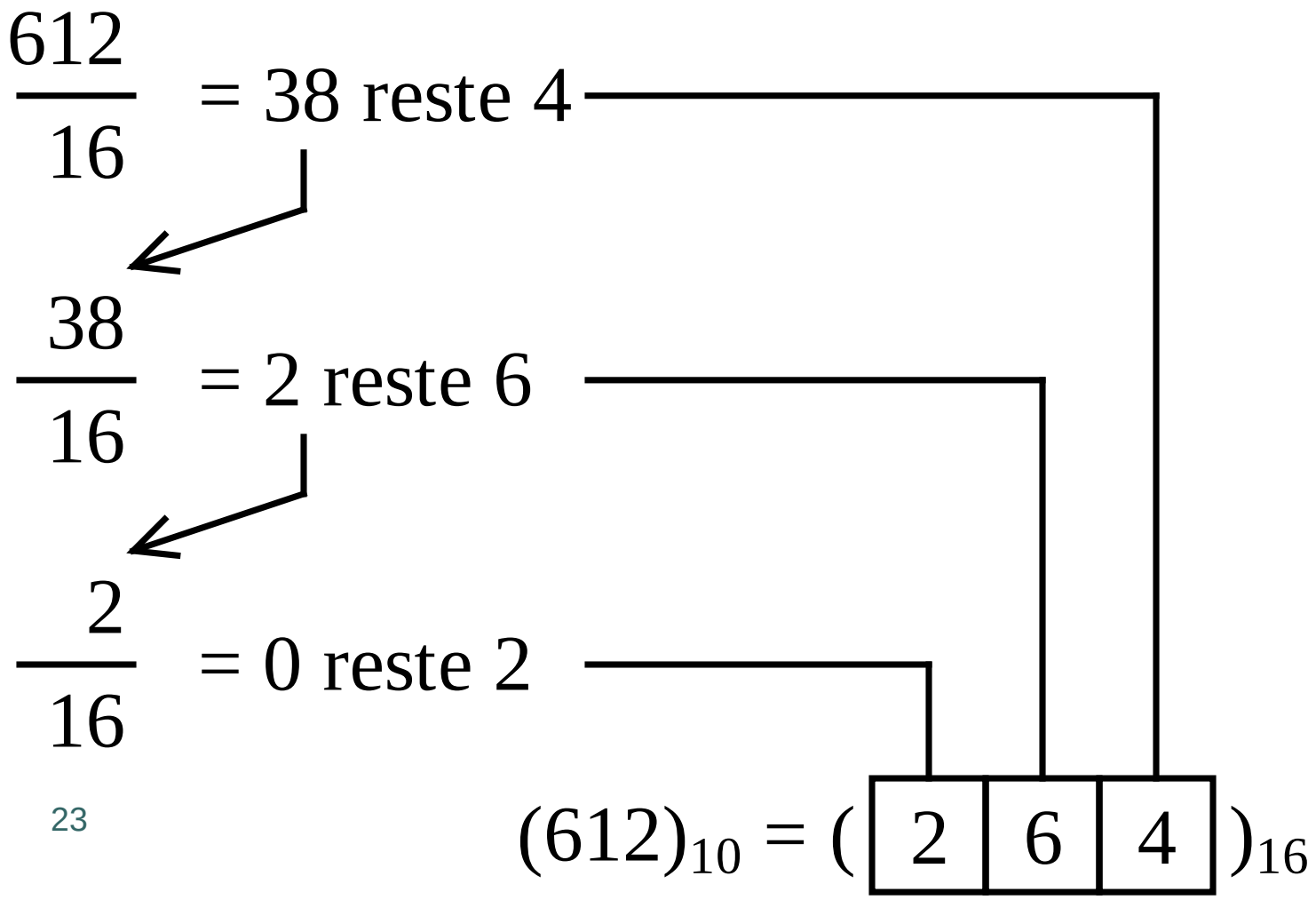
- Tableau de conversion:

Hexa	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- Exemple:

$$(4ED0)_{16} = 4*16\text{^}3+14*16\text{^}2+13*16\text{^}1$$
$$= 16384+3584+208$$
$$= (20176)_{10}$$

Conversion décimal ⇒ héxadécimal: Divisions successives par 16



Conversion héxadécimal ⇒ binaire

- Tableau de conversion:

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

- Exemple:

$$(4DE0)_{16} = (0100110111100000)_2$$

Conversion binaire ⇒ héxadécimal

•La conversion dans l'autre sens se fait en regroupant les bits par quatre, en complétant à gauche si nécessaire par des zéros.

•Exemple:

$$(1110001111110000)_2 = (E3F0)_{16}$$

Conversion héxadécimal ⇒ octal et octal ⇒ héxadécimal

•La solution la plus simple et la plus directe est de passer par le binaire.

•C'est à dire créer des groupes de trois bits puis des groupes de quatre.

•Ou pour la conversion inverse des groupes de quatre puis des groupes de trois.

3. Quelques systèmes de codage

Les systèmes de codage

- Définition:
  - Le codage est une manière de représenter des données numériques.
  - Dans le plupart des cas, il n'est pas possible de faire des calculs avec des données codées.
  - Les codes sont utilisés pour leurs propriétés qui permettent de détecter / corriger des erreurs etc....

Le codage DCB (1)

- DCB est l'abréviation de Décimal Codé Binaire.
- Ce codage fait correspondre à chaque digit d'un nombre décimal son équivalent binaire codé sur **quatre bits**.
- Bien évidemment toutes les combinaisons ne sont pas utilisées. On utilise la table suivante:

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

Le codage DCB (2)

- Exemple:

$$(874)_{10} = (100001110100)_{DCB}$$

De même

$$(0110010101001001)_{DCB} = (6549)_{10}$$

- Remarque

Il est important de remarquer que le nombre exprimé en DCB n'est pas écrit en binaire, en hexadécimal ou en octal, c'est un nombre exprimé en DCB. Le DCB n'est pas un système de numération, c'est un **système de codage**.

- Exemple:

$$(143)_{10} = (10001111)_2 = (000101000011)_{DCB}$$

Le codage « majoré de trois »

- Ce codage est très proche du codage DCB.
- On s'en sert pour effectuer certains calculs arithmétiques.
- Le principe est le même que celui du DCB mais on ajoute trois à chaque digit avant sa conversion en binaire.
- Ce qui donne la table de conversion suivante:

Décimal	0	1	2	3	4	5	6	7	8	9
DCB	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
Maj. de trois	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100

- <u>Exemple</u>

$(52)_{10} = 5+3 \mid 2+3 = (1000\mathit{0101})_{M3}$

- <u>Remarque:</u> seuls dix des seize combinaisons possibles sont utilisées.

Le codage GRAY (1)

- Le codage Gray est un codage à distance minimale.
- Son principe est d'incrémenter les valeurs en binaire mais en ne changeant qu'un seul bit à la fois pour passer d'un nombre à un autre.
- A la différence de ce que nous avons vus jusqu'à présent, les digits du code Gray ne sont affectés d'aucun poids.
- On peut établir une table de conversion comme suit:

Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Gray	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000

Le codage GRAY (2):

•Ce code dont les digits ne portent pas de poids (c'est à dire qu'il est impossible de faire des calculs avec cette représentation) est souvent utilisé pour les entrées sorties de convertisseurs analogiques numérique.

- •Le passage de la valeur $(0111)_2$ à la valeur $(1000)_2$ nécessite le changement de quatre bits. Dans certains processus pour lesquels le temps de commutation est primordial, une telle conversion peut d'une part durer trop longtemps et engendrer trop d'états transitoires.
- •Pour ce qui est des conversions analogique digital par exemple, on sait que les valeurs qui se suivent ne sont séparées les unes des autres que par une différence d'un seul bit.

Le codage ASCII (1):

•ASCII est l'abréviation de American Standard Code for Information Interchange.

•Nous ne donnerons ici qu'une liste partielle du codes ASCII.

Caractère	A	B	C	D	E	F	G	H	I	J
ASCII (7∈)	100 0001	100 0010	100 0011	100 0100	100 0101	100 0110	100 0111	100 1000	100 1001	100 1010
Hexadécimal	41	42	43	44	45	46	47	48	49	4A
Décimal	65	66	67	68	69	70	71	72	73	74

Caractère	a	b	c	d	e	f	g	h	i	j
ASCII (7∈)	110 0001	110 0010	110 0011	110 0100	110 0101	110 0110	110 0111	110 1000	110 1001	110 1010
Hexadécimal	61	62	63	64	65	66	67	68	69	6A
Décimal	97	98	99	100	101	102	103	104	105	106

Caractère	0	1	2	3	4	5	6	7	8	9
ASCII (7∈)	011 0000	011 0001	011 0010	011 0011	011 0100	011 0101	011 0110	011 0111	011 1000	011 1001
Hexadécimal	30	31	32	33	34	35	36	37	38	39
Décimal	48	49	50	51	52	53	54	55	56	57

Le codage ASCII (2)

•Le code ASCII est le code alphanumérique le plus répandu dans les micro-ordinateurs.

•C'est un code à sept bits, c'est à dire qu'on peut coder 2^7 = 128 éléments.

•Pour coder des éléments supplémentaires, en fonction de chaque pays (lettres accentuées, etc.…) il lui est souvent adjoint un huitième bit, ce qui porte ses capacités de codage à 256 éléments.

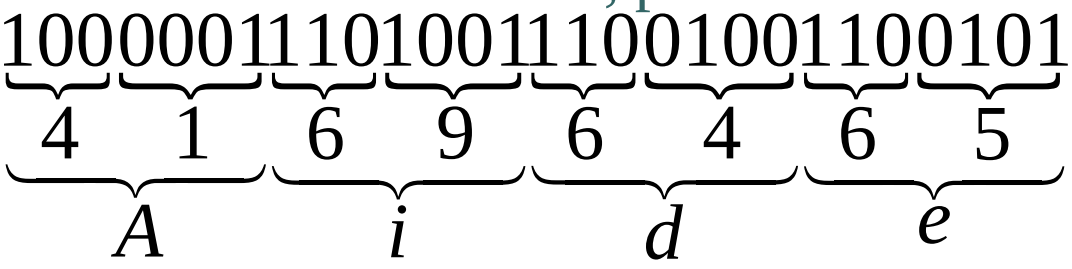
Le codage ASCII (3)36

•Exemple:

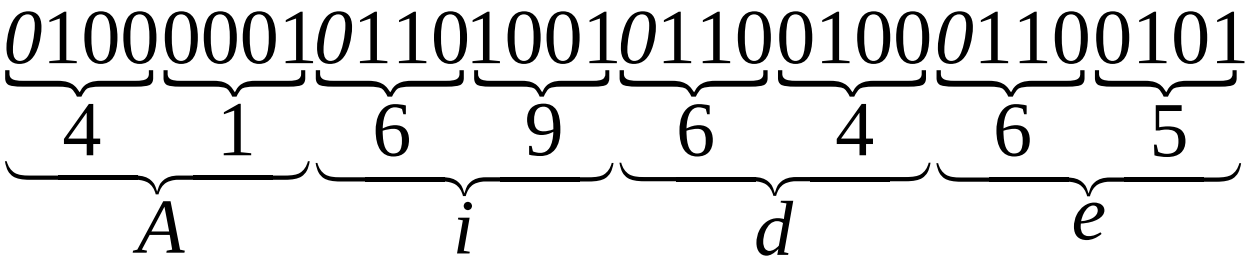
Que signifie le code ASCII suivant:

1000001 1101001 1100100 1100101

Solution : conversion en hexadécimal, puis identification:



•On peut éventuellement effectuer les mêmes opérations en complétant le premier groupe de trois bits de chaque code par un zéro.



Les codes détecteurs d ’erreurs et/ou auto-correcteurs

Problème de parasites ou perturbations lors de la transmission de données numériques

=>

utilisation des codes détecteurs voir correcteurs d'erreurs

Le code de la parité: Définition (1)

- ***La méthode de la parité paire***

  On fixe le bit de parité pour que le nombre total de un dans la représentation codée (en tenant compte du bit de parité) soit un nombre pair.

Exemple:

Le code à transmettre est 1000011 (codes ASCII du caractère 'C'):

1 1 0 0 0 0 1 1

Cas du code ASCII du caractère 'A':

0 1 0 0 0 0 0 1

Le code de la parité: Définition (2)

- ***La méthode de la parité impaire***

  Le principe en est tout à fait le même. On fixe le bit de parité pour que le nombre total de un dans la représentation codée (en tenant compte du bit de parité) soit un nombre impair.

Exemple

Le code à transmettre est 1000011 (codes ASCII du caractère 'C)

0 1 0 0 0 0 1 1

Cas du code ASCII du caractère 'A':

1 1 0 0 0 0 0 1

Le code de la parité: Application

On utilise la méthode de la parité impaire.

Emetteur	**Récepteur**
11000001 → *envoi n*^{lle} *donnée*	*mauvaise réception* → 11000000 ← *message d'erreur*
11000001 → r*envoi anc. donnée*	*bonne réception* → 11000001 ← *accusé de réception*
11001101 → *envoi n*^{lle} *donnée*	*bonne réception* → 11001101 ← *accusé de réception*

Le code de la parité: Limitation

•Cette méthode est inutile lorsqu'une erreur survient sur deux bits en même temps.

•Cette méthode est surtout utilisée lorsque la probabilité d'erreur mono bit est faible et celle d'avoir deux erreurs est nulle.

•Aucune parité n'est meilleure que l'autre bien que l'on rencontrera plus souvent un contrôle de la parité paire.

4. L'arithmétique binaire

L ’addition binaire (2)

•Nous n'avons que quatre cas en binaire:

Opération	**Résultat**	**Retenue**
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1

•Exemple:

0111 *(+7)*	1010 *(+10)*	11,011 *(+3,375)*
+ 0010 *(+2)*	+ 1111 *(+15)*	+ 10,110 *(+2,750)*
1001 *(+9)*	11001 *(+25)*	110,001 *(+6,125)*

Les nombres signés: Convention

- Les nombres négatifs et positifs sont différenciés par un **bit de signe (qui est le MSB - Most Significative Bit -):** *par convention:*
  - **zéro** si **positif,**
  - **un** si **négatif**.
- Si un nombre de 8 bits est signé:
  - 2^{8-1} = 128 éléments négatifs,
  - 2^{8-1} = 128 éléments positifs.
- Si un nombre est non signé:
  - 2^{8} = 256 éléments positifs.

Les nombres signés: Exemple

•Pour un nombre à trois bits:

Non signé		Signé	
binaire	**décimal**	**binaire**	**décimal**
000	0	**000**	**0**
001	1	**001**	**1**
010	2	**010**	**2**
011	3	**011**	**3**
100	4	**100**	**-4**
101	5	**101**	**-3**
110	6	**110**	**-2**
111	7	**111**	**-1**

Notation en complément à 1

- On complémente chaque bit:

Bit	0	1
Compl. a 1	1	0
Σ	1	1

- Exemple:

Nb. binaire	1	1	0	0	1	0	0
Compl. à 1	0	0	1	1	0	1	1

- Le complément à un de $(1100100)_2$ est $(0011011)_{cpl.1}$

Notation en complément à 2

•Le complément à deux d'un nombre binaire s'obtient en prenant le complément à un de ce nombre et en ajoutant un au bit de poids le plus faible (LSB).

•Exemple:

Nb. binaire	**1**	**1**	**0**	**0**	**1**	**0**	**0**
Compl. à 1	0	0	1	1	0	1	1
Add. de 1							1
Σ	0	0	1	1	1	0	0

•Le complément à deux de $(1100100)_2$ est $(0011100)_{cpl.2}$

Nombres binaires négatifs et notation en complément à 2

•CAS NOMBRE BINAIRE POSITIF: Exemple:

0	1	0	1	1	0	1
Bit de signe	Grandeur du nombre = grandeur exacte					

$(0101101)_{cpl.2} = (0101101)_2 = (45)_{10}$

•CAS NOMBRE BINAIRE NEGATIF: Exemple:

1	1	0	1	1	0	1
Bit de signe	Grandeur du nombre = complément à 2 de la valeur					

$(1101101)_{cpl.2} = (-19)_{10}$

•La notation en complément à deux permet de réaliser une soustraction en effectuant une simple addition => un circuit additionneur peut faire des additions et des soustractions.

Soustraction binaire

•RAPPEL:

La notation en complément à deux permet de réaliser une soustraction en effectuant une simple addition.

•Nous allons donc passer en revue les quatre types *d'additions* possibles entre deux nombres signés dont les parties négatives sont exprimées en complément à 2.

Addition binaire: 2 nombres positifs

•Cela revient à faire une addition binaire simple telle que nous l'avons déjà vue.

•<u>Exemple</u>

	0 1001	$(+9)_{10}$ (cumulande)
+	0 0100	$(+4)_{10}$ (cumulateur)
	0 1101	$(+13)_{10}$(somme)

•<u>Remarque</u>

Il faut, en notation complément à deux faire attention que le cumulande et le cumulateur aient toujours le même nombre de bits de *grandeur*.

Addition binaire: Un nombre positif et un nombre négatif plus petit

- Exemple

0 1001	$(+9)_{10}$	(cumulande)
+ 1 1100	$(-4)_{10}$	(cumulateur)
1 0 0101	$(+5)_{10}$	(somme)

- Remarque

Le report du bit de signe est toujours rejeté.

Addition binaire: Un nombre positif et un nombre négatif plus grand

- Exemple

1 0111	$(-9)_{10}$(cumulande)
+ 0 0100	$(+4)_{10}$(cumulateur)
1 1011	$(-5)_{10}$(somme)

- Comme le bit de signe indique un nombre négatif sa valeur est déterminée par son complément à deux (11011 ⇨ 00100+1 ⇨ 0101 = 5). Le résultat est bien $(-5)_{10}$.

Addition binaire: Deux nombres négatifs

•Exemple

1 0111	$(-9)_{10}$	(cumulande)
+ 1 1100	$(-4)_{10}$	(cumulateur)
1 1 0011	$(-13)_{10}$	(somme)

•Comme précédemment, le bit de signe indique un nombre négatif sa valeur est déterminée par son complément à deux (110011 ⇨ 001100+1 ⇨ 1101 = 13). Le résultat est bien $(-13)_{10}$.

•Remarque

Le report du bit des signes est toujours rejeté.

Addition binaire: Deux nombres égaux et opposés

Exemple

0 1001	$(+9)_{10}$	(cumulande)
+ 1 0111	$(-9)_{10}$	(cumulateur)
1 0 0000	$(0)_{10}$	(somme)

•Le bit de signe indique un nombre positif. Sa valeur est à lecture immédiate soit 0000 = $(0)_{10}$.

•Remarque

Les report du bit des signe est toujours rejeté.

Addition binaire: Les dépassements (1)

•Prenons l'addition suivante :

0 1001	$(+9)_{10}$	(cumulande)
+ 0 1000	$(+8)_{10}$	(cumulateur)
1 0001		(somme)

•Le bit de signe indique une réponse négative ce qui est manifestement une erreur...

•La réponse devrait être $(+17)_{10}$.

•Pour exprimer cette grandeur en binaire il faut plus de quatre bits, c'est à dire que dans le cas de calcul il y a un **dépassement**.

•Un dépassement donne toujours lieu à une erreur.

•Il peut être détecté en comparant le bit de signe des éléments à additionner et celui du résultat.

Addition binaire: Les dépassements (2)

•Une solution manuelle consiste à coder les données sur un nombre plus grand de bits.

•Exemple

0 01001	$(+9)_{10}$	(cumulande)
+ 0 01000	$(+8)_{10}$	(cumulateur)
0 10001	$(+17)_{10}$	(somme)

•Il va sans dire que cette solution n'est pas applicable aux circuits numériques d'une machine, ceux-ci ayant leur nombre de bits utiles pour les traitements figés.

Multiplication binaire des nombres purs (1)

•La multiplication de nombres binaires se fait de la même façon que celle des nombres décimaux.

•Exemple:

1001 $(9)_{10}$ (multiplicande)

* 1011 $(11)_{10}$ (multiplicateur)

1001 *(produit partiel)*

1001*x* *(produit partiel)*

0000*xx* *(produit partiel)*

1001*xxx* *(produit partiel)*

1100011 $(99)_{10}$ (produit final)

Multiplication des nombres binaires purs (2)

•La multiplication binaire n'est en fait qu'une suite de rotation et d'additions. Elle peut d'effectuer comme ci-dessous :

•Opération 1

1001	(premier produit partiel)
+ 1001*x*	(second produit partiel décalé d'un bit vers la gauche)
11011	(résultat de la première addition)

Multiplication des nombres binaires purs (3)

•Opération 2

11011	(résultat de la première addition)
+ 0000*xx*	(troisième produit partiel décalé de 2 bits vers la gauche)
011011	(résultat de la seconde addition)

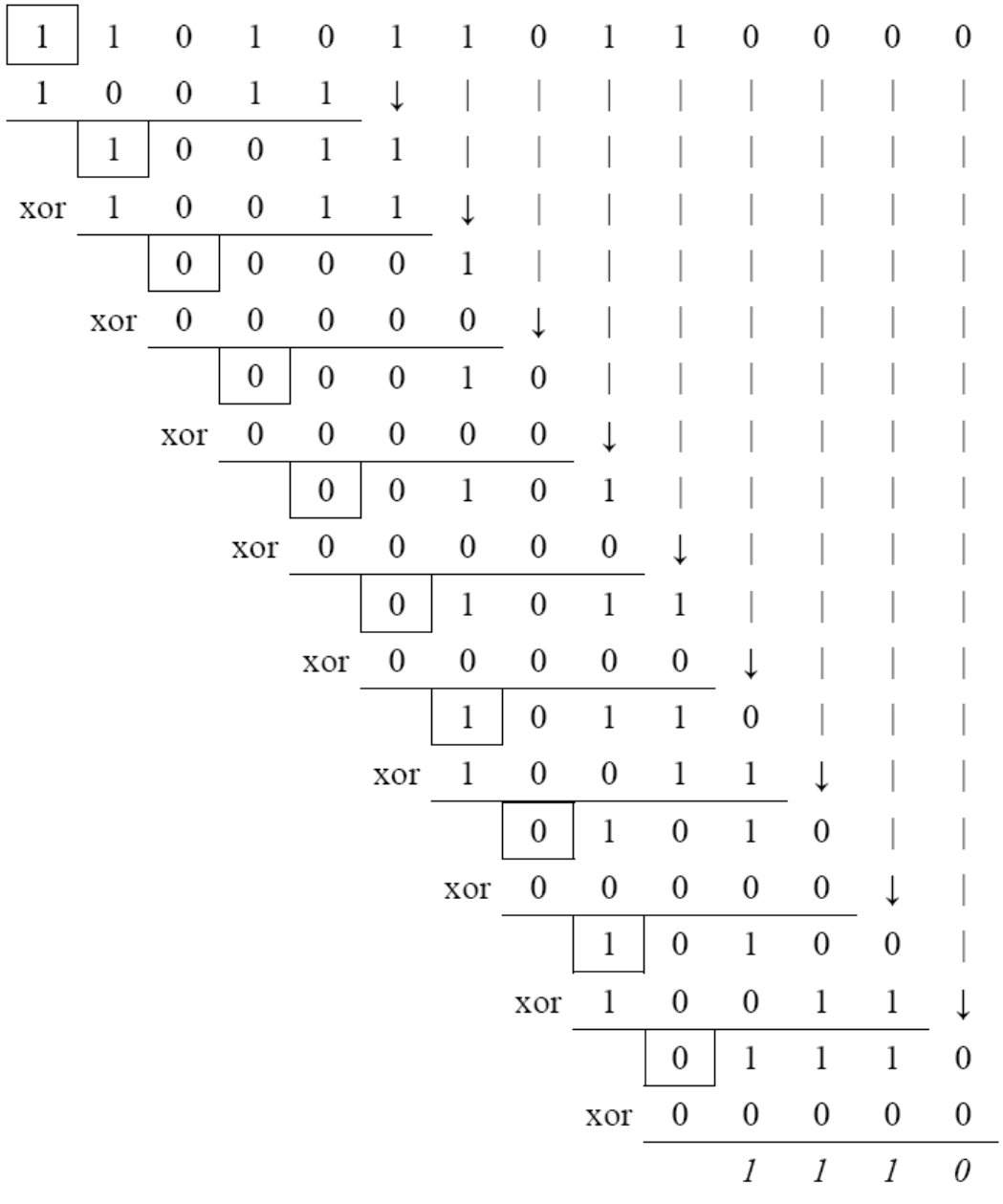
•Opération 3

011011	(résultat de la seconde addition)
+ 1001*xxx*	(quatrième produit partiel décalé de 3 bits vers la gche)
1100011	(résultat de la troisième addition)

Multiplication des nombres binaires complémentés à 2

- **CAS DE 2 NOMBRES POSITIFS:**
  - On applique la méthode précédente.
- **CAS DE 2 NOMBRES NEGATIFS:**
  - On calcule le complément à 2 de chacun des nombres (pour les rendre positifs),
  - On multiplie ces 2 nombres positifs (cas précédent),
  - Le résultat obtenu est positif.
- **CAS OU 1 DES 2 NOMBRES EST NEGATIF:**
  - On calcule le complément à 2 du nombre négatif,
  - On multiplie les 2 nombres positifs,
  - On complémente à 2 le résultat puisqu ’il est négatif.

Division binaire



1 0 0 1 1

1 1 0 0 0 0 1 0 1 0