

# CHAPITRE 3 : LA REPRÉSENTATION DE L'INFORMATION (INFORMATION REPRESENTATION)

# Introduction

2

- Les informations traitées par un ordinateur peuvent être de différents types (texte, nombres, images, son, vidéos, etc.) mais elles sont toujours représentées et manipulées par l'ordinateur sous forme binaire.
- En fait, toute information sera traitée comme une suite de 0 et de 1. L'unité d'information est donc les **chiffres binaires** (0 et 1) que l'on appelle bit (pour **binary digit** ).

# Codage binaire

3

- ❑ Le codage d'une information consiste à établir une correspondance entre la représentation externe (habituelle) de l'information (texte, image, ...etc), et sa représentation interne dans la machine c.à.d. le code binaire qui est toujours une suite de bits.
- ❑ Le code binaire, plus généralement appelé système binaire, est un système de numération utilisant la base 2 qui utilise exclusivement des 0 et des 1.
- ❑ Le code binaire, est à la base du numérique et de l'informatique. Il est utile de comprendre son principe pour mieux maîtriser ordinateurs et logiciels.

# Codage binaire

4

- En binaire, on distingue quatre principaux systèmes de codage :
  - Code Binaire **pur (naturel)**,
  - Code Binaire **BCD (Binary Coded Decimal)**,
  - Code Binaire **réfléchi** (code de **Gray**),
  - Code **Excess de trois**.

# Codage binaire

5

## Code binaire pur :

- Le codage **binaire pur** ou **binaire naturel** consiste à une conversion traditionnelle d'un nombre vers le système binaire sur un nombre de bits fixé.
- **Exemple :**
  - $(37)_{10}$  sur 8 bits =  $(00100101)_2$
  - $(37)_{10}$  sur 16 bits =  $(000000000000100101)_2$
  - $(37)_{10}$  sur 4 bits = **ERREUR** (Overflow)
- Pour un nombre composé de plusieurs chiffres : sa représentation binaire dépend de ce nombre (pas de chiffre à la fois)

# Codage binaire

6

## Code binaire **DCB** ou **BCD**

- ❑ Le codage binaire **DCB** (Décimal Codé Binaire) en français ou **BCD** (**B**inary **C**oded **D**ecimal) en anglais, ce code permet de simplifier la conversion avec la notation décimale.
- ❑ C'est un système de numération utilisé en électronique numérique et en informatique pour coder des nombres en se rapprochant de la représentation humaine , en base 10.

# Codage binaire

7

## Code binaire DCB ou BCD

- ❑ Le code BCD consiste à convertir chaque chiffre d'un nombre en base 10 par son équivalent binaire sur 4 bits (comme montré dans le tableau à droite), le résultat est le regroupement des codes binaires.

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

# Codage binaire

8

## Code binaire DCB ou BCD

- ❑ Les combinaisons restantes ne sont pas utiliser en BCD

Chiffre décimal	Code BCD
/	1010
/	1011
/	1100
/	1101
/	1110
/	1111

Non utilisés



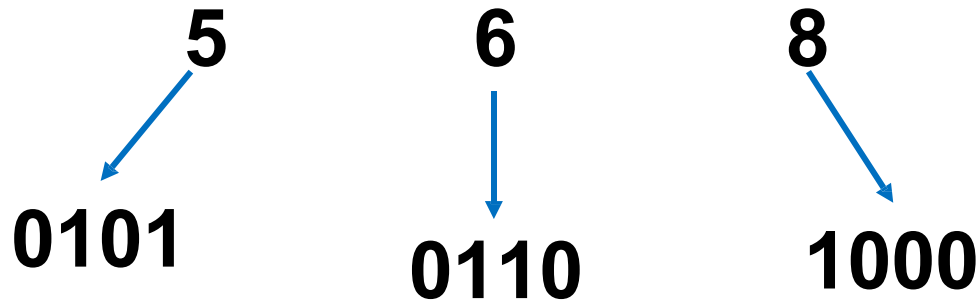
# Codage binaire

9

## Code binaire DCB ou BCD

### Exemple Conversion du Décimal vers le BCD :

- ❑  $(568)_{10} = ( ? )_{BCD}$ . L'indication  $(.....)_{BCD}$  signifie qu'il s'agit d'un nombre Décimal Codé en Binaire.
- ❑ on code chaque digit (chiffre) décimal par un ensemble de quatre chiffres binaires.



$$(568)_{10} = ( 010101101000 )_{BCD}$$

# Codage binaire

10

## Code binaire DCB ou BCD

### Exemple Conversion du BCD vers le Décimal :

- ❑  $(010101101000)_{\text{BCD}} = (?)_{10}$
- ❑ Pour la conversion de BCD vers décimal on remplace chaque bloc de 4 bits (en partant de la droite) par son équivalent décimal.

0101



5

0110



6

1000



8

$(010101101000)_{\text{BCD}} = (568)_{10}$

# Codage binaire

11

## Code binaire DCB ou BCD

### Addition BCD :

- ❑ Comme tout autre système de numération dans l'arithmétique l'**Addition BCD** est une opération qui peut être nécessaire.

# Codage binaire

12

## Code binaire DCB ou BCD

### Addition BCD :

- ❑ L'addition BCD respecte les règles suivantes :
  - ❑ Si une somme de 4 bits est  $\leq 9$ , le résultat est un nombre BCD.
  - ❑ Si une somme de 4 bits est  $> 9$ , ou une retenue est créée à partir d'un groupe de 4 bits, le résultat est non valide. Il faut additionner 6 (0110) au groupe de 4 bits qui causé l'erreur.

# Codage binaire

13

## Code binaire DCB ou BCD

### Exemple 1 Addition BCD :

❑ faite l'opération suivante en BCD : 43 + 35

$$\begin{array}{r} 43 \\ + 35 \\ \hline = 78 \end{array}$$

$$\begin{array}{r} 0100 \quad 0011 \\ + 0011 \quad 0101 \\ \hline = 0111 \quad 1000 \end{array}$$

# Codage binaire

14

## Code binaire DCB ou BCD

### Exemple 2 Addition BCD :

❑ faite l'opération suivante en BCD : 89 + 85

$$\begin{array}{r} + 89 \\ + 85 \\ \hline = 174 \end{array}$$
  
$$\begin{array}{r} + \quad \quad \quad 1000 \quad 1001 \\ \quad \quad \quad 1000 \quad 0101 \\ \hline = \quad 1 \quad \underbrace{0000} \quad \underbrace{1110} \end{array}$$

Cette somme produit un report à gauche

> 9

$$\begin{array}{r} + \quad \quad \quad 110 \quad 110 \\ \quad \quad \quad 0111 \quad 0100 \\ \hline = \quad 1 \quad \underbrace{0111} \quad \underbrace{0100} \end{array}$$

# Codage binaire

15

## Code binaire DCB ou BCD

### Soustraction BCD :

- ❑ Comme l'Addition BCD, la **Soustraction BCD** est une opération qui peut être nécessaire aussi et suit les mêmes règles que l'addition c.à.d. que s'il y a une différence de 4 bits qui est  $> 9$ , ou une retenue est créée à partir d'un groupe de 4 bits, le résultat est non valide. Il faut soustraire 6 (0110) au groupe de 4 bits qui causé l'erreur.

# Codage binaire

16

## Code binaire réfléchi (code de GRAY)

- ❑ Le **code binaire réfléchi**, également appelé **code Gray**, est un type de codage binaire permettant de ne modifier qu'un seul bit à la fois quand un nombre est augmenté d'une unité, cette propriété est importante pour certaines applications .
- ❑ Ce code est utilisé dans :
  - ❑ Les tableaux de Karnaugh.
  - ❑ Dans des circuits d'entrée/sortie, notamment dans les codeurs optiques.
  - ❑ Dans certains convertisseurs analogique/numérique



# Codage binaire

17

## Code binaire réfléchi (code de GRAY)

❑ Le code binaire réfléchi est différent du code binaire pur .

Décimale	Binaire	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

# Codage binaire

18

## Code binaire réfléchi (code de GRAY)

**Principe** : Le nom de code binaire réfléchi, vient de la méthode de construction basée sur la réflexion de blocs de codes déjà construits :

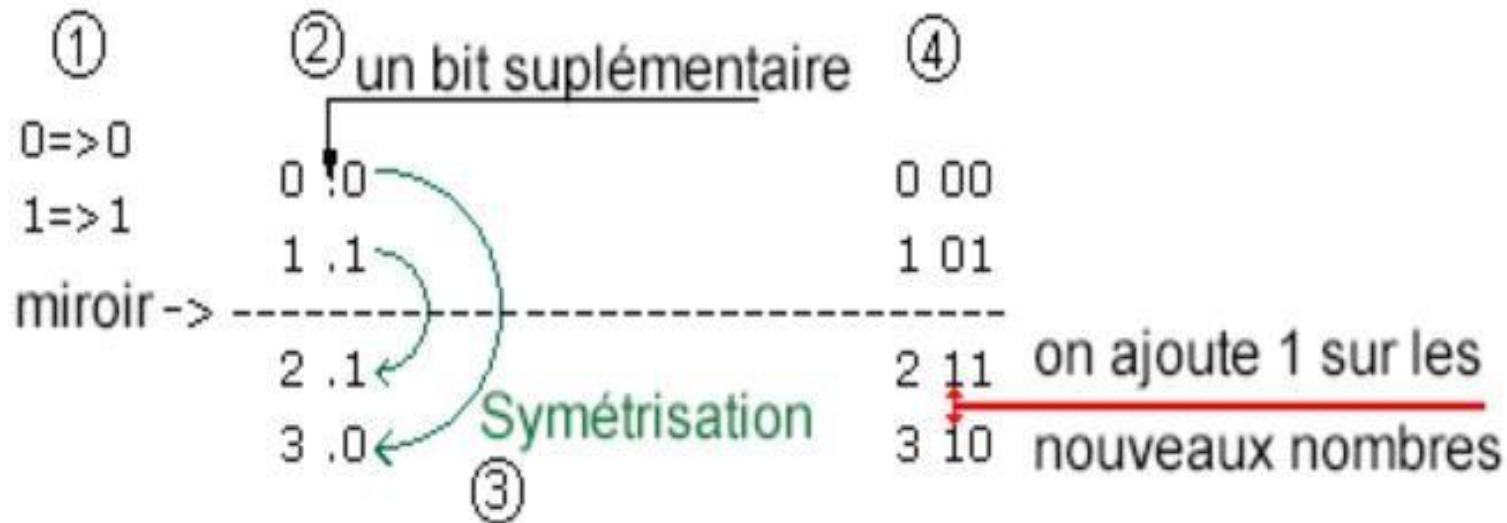
- ❑ On choisit un code de départ : zéro est codé 0 et *un* est codé 1,
- ❑ Puis, à chaque fois qu'on a besoin d'un bit supplémentaire, on symétrise la liste des codes déjà obtenus (comme une *réflexion* dans un miroir),
- ❑ Enfin, on rajoute un 0 puis un 1 au début (à gauche) de chacun des codes. On a ainsi doublé le nombre de codes formés.

# Codage binaire

19

## Code binaire réfléchi (code de GRAY)

**Exemple 1** : pour les 4 premiers chiffres décimaux 0 à 3.



# Codage binaire

20

## Code binaire réfléchi (code de GRAY)

**Exemple 1** : pour les 4 premiers chiffres décimaux 0 à 3.

0		0	0	
1		0	1	on inverse
2		1	1	1 bits à la fois
3		1	0	le plus à droite possible

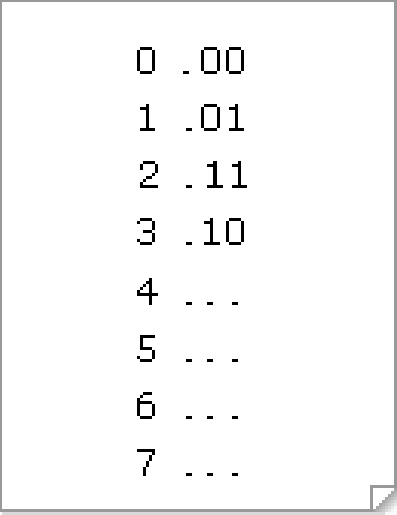
*Construction du code Gray : nous avons inversé 1 bits à la fois.*

# Codage binaire

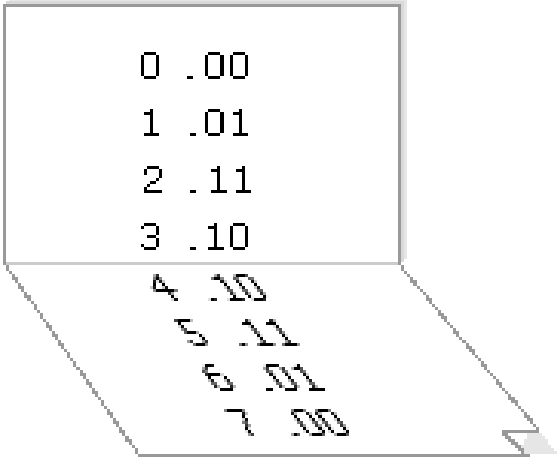
21

## Code binaire réfléchi (code de GRAY)

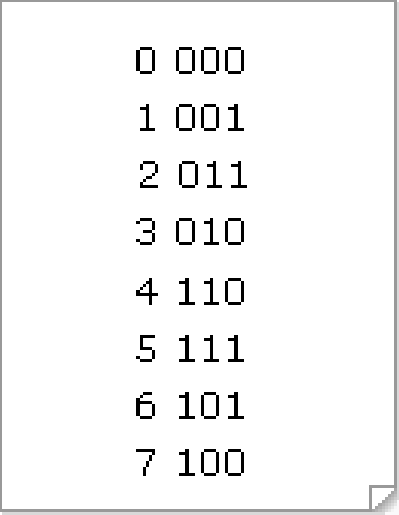
**Exemple 2** : pour les 8 premiers chiffres décimaux 0 à 7.  
Nous avons besoin de 3 bits pour représenter 8 valeurs binaires.



0	.00
1	.01
2	.11
3	.10
4	...
5	...
6	...
7	...



0	.00
1	.01
2	.11
3	.10
4	.10
5	.11
6	.01
7	.00



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

# Codage binaire

22

## Code binaire réfléchi (code de GRAY)

**Exemple 2** : pour les 8 premiers chiffres décimaux 0 à 7. Nous avons besoin de 3 bits pour représenter 8 valeurs binaires.

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

*Construction du code Gray sur 3 bits.*

# Codage binaire

23

## Code binaire réfléchi (code de GRAY)

### Conversion du binaire pur vers binaire réfléchi :

Nous passons par 4 étapes comme suit :

1. Commencez par le bit le plus à gauche appelé aussi le **MSB** (**M**ost **S**ignificant **B**it en anglais et Bit du poids le plus fort en français) du nombre binaire. Le **MSB** du code Gray est le même que le **MSB** du nombre binaire donné, c.à.d. prendre le **MSB** du binaire pur et le mettre dans sa position correspondante du code de Gray.

# Codage binaire

24

## Code binaire réfléchi (code de GRAY)

**Conversion du binaire pur vers binaire réfléchi :**

**2.** Le deuxième bit le plus significatif, adjacent au MSB, dans le numéro de code Gray est obtenu en ajoutant le MSB et le second MSB du nombre binaire et en ignorant le report éventuel. C'est-à-dire que si le bit de poids fort et le bit qui lui est adjacent sont tous deux "1", le bit de code Gray correspondant serait un "0".



# Codage binaire

25

## Code binaire réfléchi (code de GRAY)

### Conversion du binaire pur vers binaire réfléchi :

3. Le troisième bit le plus significatif, adjacent au deuxième MSB, du numéro de code Gray est obtenu en ajoutant le deuxième MSB et le troisième MSB au nombre binaire et en ignorant le report éventuel.
4. Le processus se poursuit jusqu'à l'obtention du LSB du numéro de code Gray par l'addition du **LSB** (Least Significant Bit en anglais et le bit du poids le plus faible) et du bit adjacent supérieur le plus proche du nombre binaire.

# Codage binaire

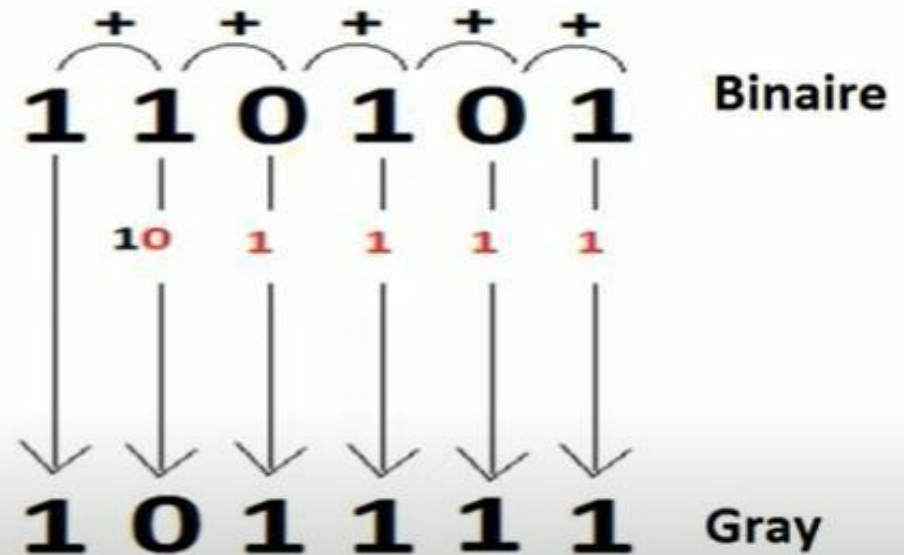
26

## Code binaire réfléchi (code de GRAY)

• Conversion du binaire pur vers binaire réfléchi :

Exemple :

$$(110101)_2 = ?$$



$$(110101)_2 = (101111)_{\text{Gray}}$$

# Codage binaire

27

## Code binaire réfléchi (code de GRAY)

### Conversion du binaire réfléchi vers binaire pur :

Nous passons par les 4 étapes suivantes :

1. Commencez par le bit le plus significatif (MSB). Le MSB du nombre binaire est identique au MSB du numéro de code Gray.
2. Le bit situé à côté du MSB (le second MSB) du nombre binaire est obtenu en ajoutant le MSB du nombre binaire au second MSB du numéro de code Gray et en ignorant le report éventuel.

# Codage binaire

28

## Code binaire réfléchi (code de GRAY)

### Conversion du binaire réfléchi vers binaire pur :

3. Le troisième MSB du nombre binaire est obtenu en ajoutant le deuxième MSB du nombre binaire au troisième MSB du numéro de code Gray. Encore une fois, la retenue, le cas échéant, doit être ignoré.
4. Le processus continue jusqu'à l'obtention du LSB du nombre binaire.

# Codage binaire

29

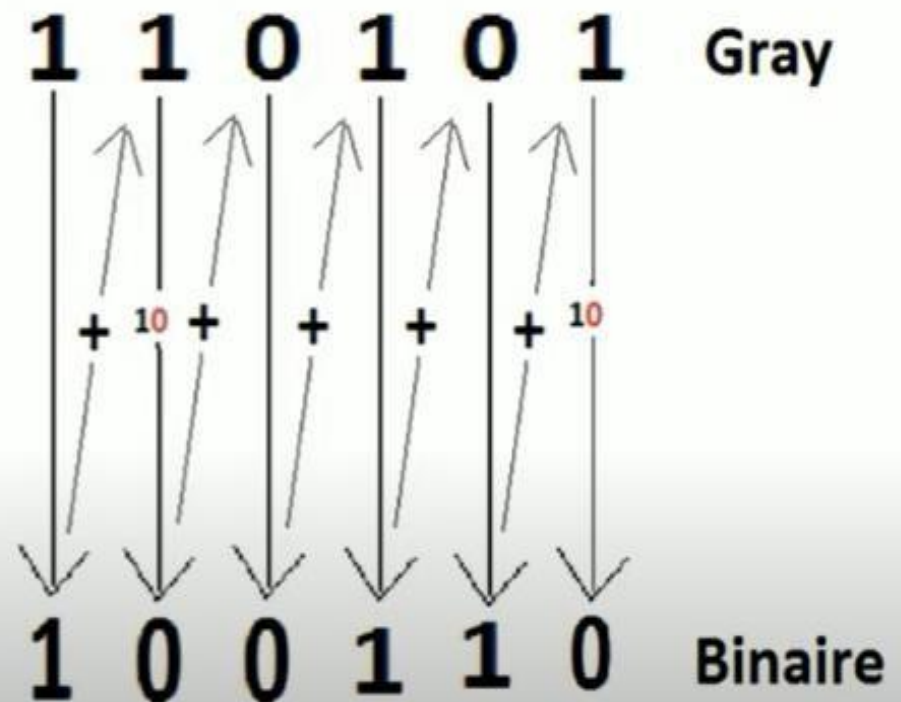
# Code binaire réfléchi (code de GRAY)

## Conversion du binaire réfléchi vers binaire pur :

- **Exemple :**

$(110101)_{\text{Gray}} = ?$

$$(110101)_{\text{Gray}} = (100110)_2$$



# Codage binaire

30

## Code binaire Excess-3

- ❑ Le code **Excess-3 (XS-3)** ou code plus 3 connu aussi sous le nom de **code de STIBITZ** du nom de son inventeur, Ce code ressemble beaucoup au code BCD, utilisé principalement par d'anciens processeurs pour la représentation des nombres en base 10.
- ❑ Son principe est basé sur le fait d'associer à chaque chiffre décimal son équivalent binaire additionné de 3 (c.à.d.  $\text{chiffre} + 3$ ).

# Codage binaire

31

## Code binaire Excess-3

Chiffre Décimal	Code XS3
0	0011
1	0100
2	0101
3	0110
4	0111

Chiffre Décimal	Code XS3
5	1000
6	1001
7	1010
8	1011
9	1100

# Codage binaire

32

## Code binaire à excès de trois

- **Conversion du Décimal vers le binaire Excess-3**  
On prends chaque chiffre du nombre décimal on lui ajoute 3 puis on le code en binaire sur 4 bits.

**Exemple :**  $(512)_{10} = ( ? )_{XS3}$

5	1	2
+3	+3	+3
=8	=4	=5
<u>1000</u>	<u>0100</u>	<u>0101</u>

$$(512)_{10} = (1000\ 0100\ 0101)_{XS3}$$



# Codage binaire

33

## Code binaire à excès de trois

- **Conversion du binaire Excess-3 vers le Décimal**  
On découpe le code XS-3 en des groupes de 4 bits en partant de la droite vers la gauche, ensuite convertir chaque groupe par son équivalent décimal et enfin soustraire de chaque chiffre le nombre 3 pour obtenir le résultat final qui est le nombre décimal.

# Codage binaire

34

## Code binaire à excès de trois

- Conversion du binaire Excess-3 vers le Décimal

**Exemple :**  $(100001000101)_{XS3} = ( ? )_{10}$

<u>1000</u>	<u>0100</u>	<u>0101</u>
8	4	5
-3	-3	-3
=5	=1	=2

$$(1000 \ 0100 \ 0101)_{XS3} = (512)_{10}$$

# Représentation des caractères

35

- ❑ Nous avons appris jusqu'à présent comment convertir des nombres entiers en binaire. Cela reste une conversion entre des nombres.

Les nombres ...

1	2	3
01	10	11

# Représentation des caractères

36

- ❑ Maintenant La question qui se pose est « **Comment coder les caractères d'un texte ?** ».

A

??

# Représentation des caractères

37

- Les **caractères** d'un texte (lettres majuscules et minuscules, caractères spéciaux, caractère de ponctuation ...), qui ne sont pas des nombres, et pourtant il faut les représenter en binaire pour que la machine puisse les comprendre, stocker et utiliser.



```
! " # $ % & ' ( ) * + , - . /  
0 1 2 3 4 5 6 7 8 9 : ; < = > ?  
@ A B C D E F G H I J K L M N O  
P Q R S T U V W X Y Z [ \ ] ^ _  
` a b c d e f g h i j k l m n o  
p q r s t u v w x y z { | } ~
```

# Représentation des caractères

38

- ❑ L'idée pour pouvoir coder les caractères est la suivante :
  - ❑ Créer une **table de caractères** qui associe à chaque caractère un nombre (un code), qui est un entier positif ;
  - ❑ Cet entier est ensuite codé par une séquence de bits en machine selon un certain encodage.

# Représentation des caractères

39

- ❑ Pour représenter et stocker les textes (**chaines de caractères**) plusieurs méthodes ont été créées, on appelle ces méthodes des **Normes internationales**.
- ❑ Les **Normes internationales** aident les organisations à comprendre les technologies de l'information et de la communication (TIC) en fournissant des outils et des méthodes qui ont fait l'objet d'un consensus international et favorisent l'interopérabilité, la sécurité et l'innovation.

# Représentation des caractères

40

- ❑ Il existe plusieurs variantes de normes pour représenter les caractères en binaire, les plus connus sont :
  - Le codage **EBCDIC**.
  - Le codage **ASCII**.
  - Le codage **Unicode**.



# Représentation des caractères

41

## Codage EBCDIC

- ❑ Le code EBCDIC signifie « **E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode » est un mode de codage des caractères sur 8 bits (c.à.d. 256 caractères) créé par IBM à l'époque des cartes perforées.
- ❑ Il existe au moins 6 versions différentes bien documentées (et de nombreuses variantes parfois créées par des concurrents d'IBM), incompatibles entre elles.

# Représentation des caractères

42

## Codage EBCDIC

- ❑ Ce mode de codage a été critiqué pour cette raison, mais aussi parce que certains caractères de ponctuation ne sont pas disponibles dans certaines versions.
- ❑ Ces disparités ont parfois été interprétées comme un moyen pour IBM de conserver ses clients captifs.

# Représentation des caractères

## : Table EBCDIC

**TABLE 6**  
**EBCDIC (IBM MAINFRAME) CHARACTER CODES**

Each code is shown in decimal, hexadecimal, and character form.

129	81	a	193	C1	A	240	F0	0
130	82	b	194	C2	B	241	F1	1
131	83	c	195	C3	C	242	F2	2
132	84	d	196	C4	D	243	F3	3
133	85	e	197	C5	E	244	F4	4
134	86	f	198	C6	F	245	F5	5
135	87	g	199	C7	G	246	F6	6
136	88	h	200	C8	H	247	F7	7
137	89	i	201	C9	I	248	F8	8
						249	F9	9
145	91	j	209	D1	J	64	40	blank
146	92	k	210	D2	K	76	4C	<
147	93	l	211	D3	L	77	4D	(
148	94	m	212	D4	M	78	4E	+
149	95	n	213	D5	N	79	45	
150	96	o	214	D6	O	80	50	&
151	97	p	215	D7	P	90	5A	!
152	98	q	216	D8	Q	91	5B	\$
153	99	r	217	D9	R	92	5C	*
						93	5D	)
162	A2	s	226	E2	S	94	5E	;
163	A3	t	227	E3	T	96	60	-
164	A4	u	228	E4	U	97	61	/
165	A5	v	229	E5	V	107	6B	,
166	A6	w	230	E6	W	108	6C	%
167	A7	x	231	E7	X	109	6D	-
168	A8	y	232	E8	Y	110	6E	>
169	A9	z	233	E9	Z	111	6F	?
122	7A	:	125	7D	,			
123	7B	#	126	7E	=			
124	7C	@	127	7F	"			

# Représentation des caractères

44

## Codage ASCII

- ❑ Le code ASCII signifie « **A**merican **S**tandard **C**ode **f**or **I**nformation **I**nterchange » (Code américain normalisé pour l'échange d'information), plus connu sous l'acronyme **ASCII** (prononcez ASKI), est une **norme informatique** pour le **codage des caractères**. En adoptant le même codage, les systèmes informatiques conçus par n'importe quel fabricant savent ainsi échanger du texte, des nombres, des signes de ponctuation et bien d'autres symboles.

# Représentation des caractères

45

## Codage ASCII

- Chaque caractère à un code ASCII

A	B	C	D	E	F	G	H	I	...
65	66	67	68	69	70	71	72	73	...

# Représentation des caractères

46

## Codage ASCII

- ❑ L'ASCII code chaque caractère sur 7 bits dans la mémoire des ordinateurs, c'est-à-dire dans sept cases ne pouvant contenir que 0 ou 1, ce qui, en binaire, permettant ainsi de coder  $2^7 = 128$  caractères différents (de 0000000 à 1111111). Toutefois, les ordinateurs travaillent tous sur un multiple de huit bits (octet), donc chaque caractère est stocké dans un octet dont le 8<sup>ème</sup> bit est 0.

20/11/2023

# Représentation des caractères

47

## Codage ASCII

- ❑ Ce codage sur 7 bits donne 128 codes différents répartis comme suit :
  - ❑ 0 à 31 : caractères de contrôle (retour à la ligne, Tabulation, etc....)

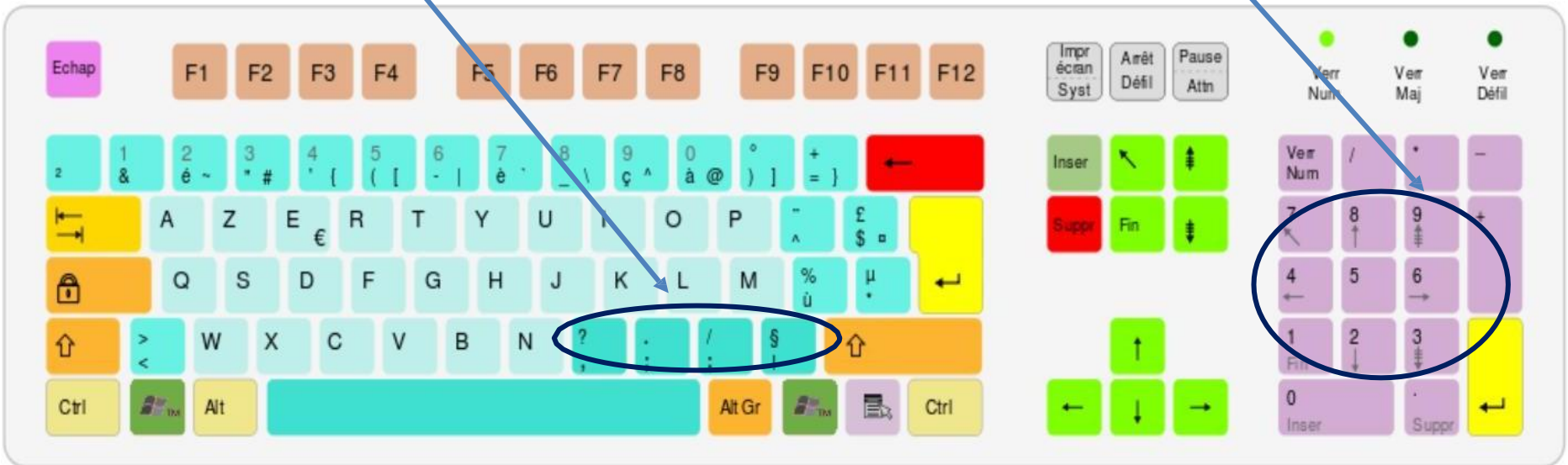


# Représentation des caractères

48

# Codage ASCII

- ❑ À partir du code 32, suivent des signes de ponctuation et quelques symboles mathématiques comme ! ou + ou / , puis les chiffres arabes de 0 à 9





# Représentation des caractères

49

## Codage ASCII

- ❑ 65 à 90 : lettres majuscules
- ❑ 97 à 122 : lettres minuscules



- ❑ Le reste des codes représente les caractères spéciaux.

# Représentation des caractères :

## Table ASCII

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

# Représentation des caractères

51

## Codage ASCII

❑ **Exemple 1** : coder la phrase « Hi! »

H	i	!
72	105	33

# Représentation des caractères

52

## Codage ASCII

❑ **Exemple 1** : coder la phrase « Hi! »

Le code **ASCII** est codé en **binaire**

H	i	!
72	105	33
1001000	1101001	100001

# Représentation des caractères

53

## Codage ASCII

- ❑ **Exemple 1** : coder la phrase « Hi! »
- ❑ Le code ASCII de la phrase « Hi! » est :  
01001000 01101001 00100001

# Représentation des caractères

54

## Codage ASCII

- ❑ **Exemple 2** : vous trouvez ci-dessous des valeurs binaires, chaque ligne représente une lettre, en lisant le mot de haut en bas. Trouvez ce mot ?

1ère lettre : 01000001

2ème lettre : 01101101

3ème lettre : 01101001

4ème lettre : 01010011



# Représentation des caractères

55

## Codage ASCII

- ❑ **Exemple 2** : On convertit chaque code binaire en décimal (utiliser la forme polynomiale)

1ère lettre : 01000001 65

2ème lettre : 01101101 109

3ème lettre : 01101001 105

4ème lettre : 01010011 83

# Représentation des caractères

56

## Codage ASCII

### Exemple 2 :

1ère lettre : 01000001 **65**

2ème lettre : 01101101 **109**

3ème lettre : 01101001 **105**

4ème lettre : 01010011 **83**

ON cherche pour  
chaque code  
décimal son caractère

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	.	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	,	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



# Représentation des caractères

57

## Codage ASCII

❑ **Exemple 2** : Le mot est : « AmiS »

65	109	105	83
A	m	i	S

# Représentation des caractères

58

## Codage ASCII

- ❑ Le code ASCII ayant été établi par des Américains, dont la langue ne comporte pas d'accents, fut très vite limitée car les langues latines contiennent les caractères accentués tel que le français, et il y a d'autres langues qui utilisent d'autres caractères tel que l'Arabe, chinois....

è, é, ê, ë

中文  
Chinese

هَيْبَرَعْلَا  
Arabic

кириллица  
Cyrillic

देवनागरी  
Devanagari

# Représentation des caractères

59

## Codage ASCII

- ❑ C'est ce qui a conduit les organismes de normalisation à définir d'autres codages plus étoffés, qui ajoutent à l'ASCII les caractères qui manquent à une langue, un pays ou une culture ce qui a donné naissance à l'**UNICODE**.

# Représentation des caractères

60

## Codage Unicode

- ❑ L'Unicode est la version courte de « *Universal Character Encoding* » en anglais, c'est-à-dire « Codage universel de caractères ».
- ❑ Ce qui fait la spécificité d'Unicode, c'est que ce standard n'est pas lié aux formats et aux codages de l'alphabet d'une langue en particulier. Au contraire, Unicode a été créé dans le but de servir de norme uniforme pour représenter **tous les systèmes d'écriture et tous les caractères** qui existent à travers le monde.

# Représentation des caractères

61

## Codage Unicode

- ❑ Unicode est uniquement une table qui regroupe tous les caractères existants dans le monde, il ne s'occupe pas de la façon dont les caractères sont codés dans la machine.
- ❑ Unicode accepte plusieurs systèmes de codage : UTF-8, UTF-16, UTF-32. Le plus utilisé, notamment sur le Web, est UTF-8
- ❑ UTF signifie « **Unicode Transformation Format** ». Il s'agit d'une famille de normes pour coder le jeu de caractères Unicode en binaire.

# Représentation des caractères

62

## Codage Unicode : UTF-8

- ❑ Le codage, UTF-8 utilise un nombre variable d'octets (de 1 à 4 octets) : les caractères "classiques" (les plus couramment utilisés) sont codés sur un octet, alors que des caractères "moins classiques" sont codés sur un nombre d'octets plus important (jusqu'à 4 octets).
- ❑ Un des avantages d'UTF-8 c'est qu'il est totalement compatible avec la norme ASCII : les caractères Unicode codés avec UTF-8 ont exactement le même code que les mêmes caractères en ASCII.

# Représentation des caractères

63

## Codage Unicode : UTF-16

- ❑ Le codage, UTF-16 utilise une seule unité de code 16 bits à largeur fixe.
- ❑ Avec 16 bits, on peut de fait représenter les caractères utilisés couramment par l'humanité, par exemple les caractères dérivés de l'alphabet latin, mais aussi les caractères dérivés de l'alphabet cyrillique, les écritures du moyen orient, sans oublier les écritures asiatiques.
- ❑ Pour les points de code à partir de 65536, UTF-16 utilise deux blocs de 16 bits.

# Représentation des caractères

64

## Codage Unicode : UTF-32

- ❑ Le codage, UTF-32 nécessite 4 octets pour coder tout caractère. Dans la plupart des cas, un document codé au format UTF-32 est presque deux fois plus volumineux que le même document codé au format UTF-16. Chaque caractère est codé dans une seule unité de code à largeur fixe en 32 bits.
- ❑ Vous utilisez UTF-32 si l'espace mémoire ne pose pas de problème et si vous souhaitez pouvoir utiliser une seule unité de code pour chaque caractère.