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

Mme KAZI TANI ADILA

STRUCTURE MACHINE 1

Introduction

- 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 **b**inary dig**it**).

Codage binaire

- 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

- En binaire, on distingue quatre principaux systèmes de codage :
  - Code Binaire **pur (naturel)**,
  - Code Binaire **BCD** (**B**inary **C**oded **D**ecimal),
  - Code Binaire **réfléchi** (code de **Gray**),
  - Code **Excess de trois**.

Codage binaire

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 = $(0000000000100101)_2$
  - $(37)_{10}$ sur 4 bits = ERREUR (Overflow)
- Pour un nombre composé de plusieurs chiffres : sa représentation binaire dépends de ce nombre (pas des chiffres qui le composent)

Codage binaire

Code binaire DCB ou BCD

- Le codage binaire **DCB** (**D**écimal **C**odé **B**inaire) 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

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

Code binaire DCB ou BCD

- Les combinaisons restantes ne sont pas utiliser en BCD

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

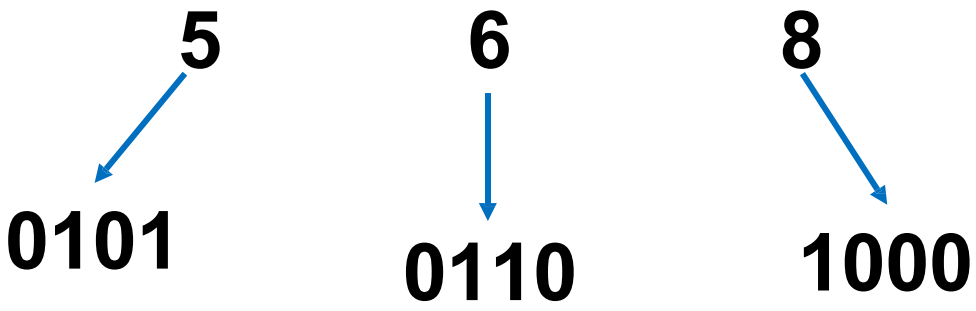
Non utilisés

Codage binaire

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.



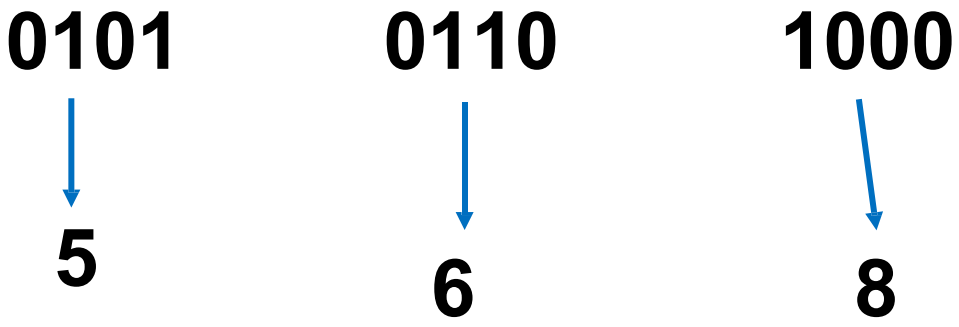
$(568)_{10}$ = (**010101101000**) $_{BCD}$

Codage binaire

Code binaire DCB ou BCD

Exemple Conversion du BCD vers le Décimal :

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



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

Codage binaire

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

Code binaire DCB ou BCD

Addition BCD :

- L'addition BCD respecte les règles suivantes :
  - Si une somme de 4 bits est ≤ 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 additionnez 6 (0110) au groupe de 4 bits qui causé l'erreur.

Codage binaire

Code binaire DCB ou BCD

Exemple 1 Addition BCD :

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

	43			0100	0011
+	35		+	0011	0101
=	78		=	0111	1000

Codage binaire

Code binaire DCB ou BCD

Exemple 2 Addition BCD :

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

	89	1000 1001
+	85	1000 0101
=	174	1 0000 1110

1

Cette somme produit un report à gauche

> 9

+	110	110
=	1 0111	0100

Codage binaire

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 quiest > 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

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 certaine applications .
- Ce code est utilisé dans :
  - Les tableaux de Karnaugh.
  - Dans des circuits d'entré/sortie, notamment dans les codeurs optiques.
  - Dans certain convertisseur analogique/numérique

Codage binaire

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

Code binaire réfléchi (code de GRAY)

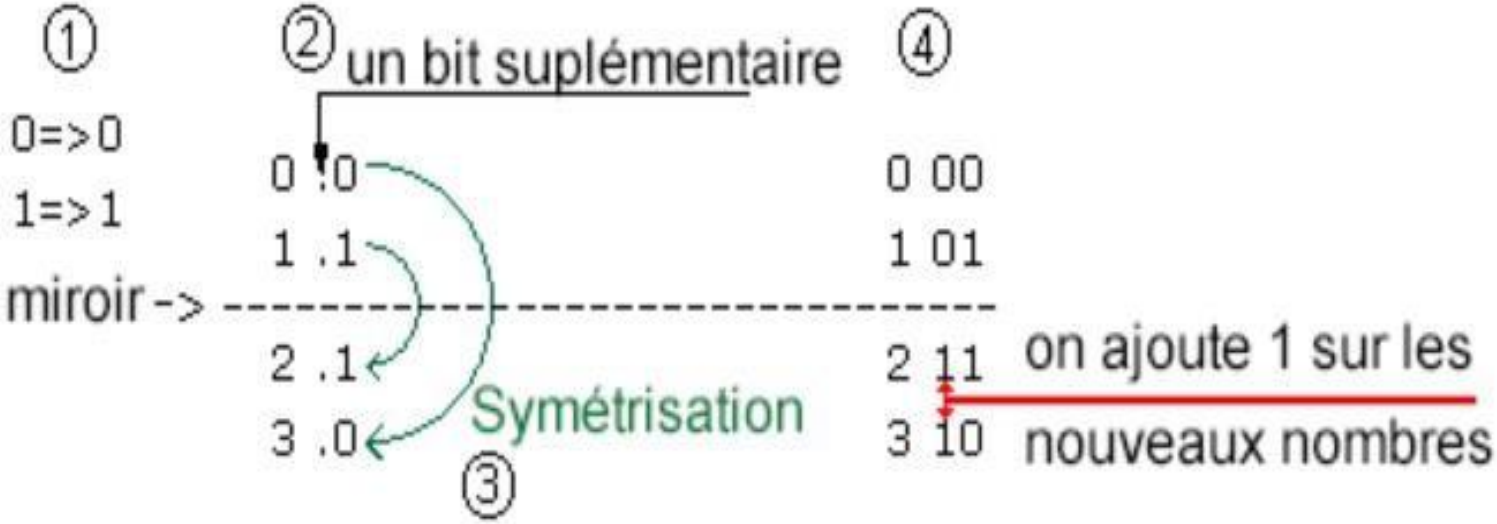
Principe : Le nom de code binaire réfléchi, vient de la méthode de construction basée sur la **réflexion** de blocs de codes déjà construits :

- 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

Code binaire réfléchi (code de GRAY)

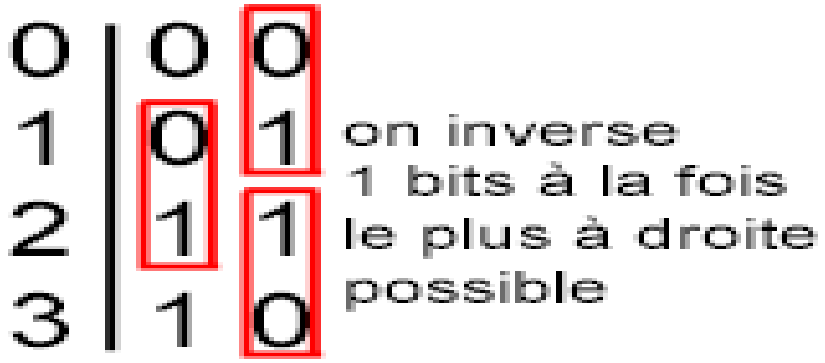
Exemple 1 : pour les 4 premiers chiffres décimaux 0 à 3.



Codage binaire

Code binaire réfléchi (code de GRAY)

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

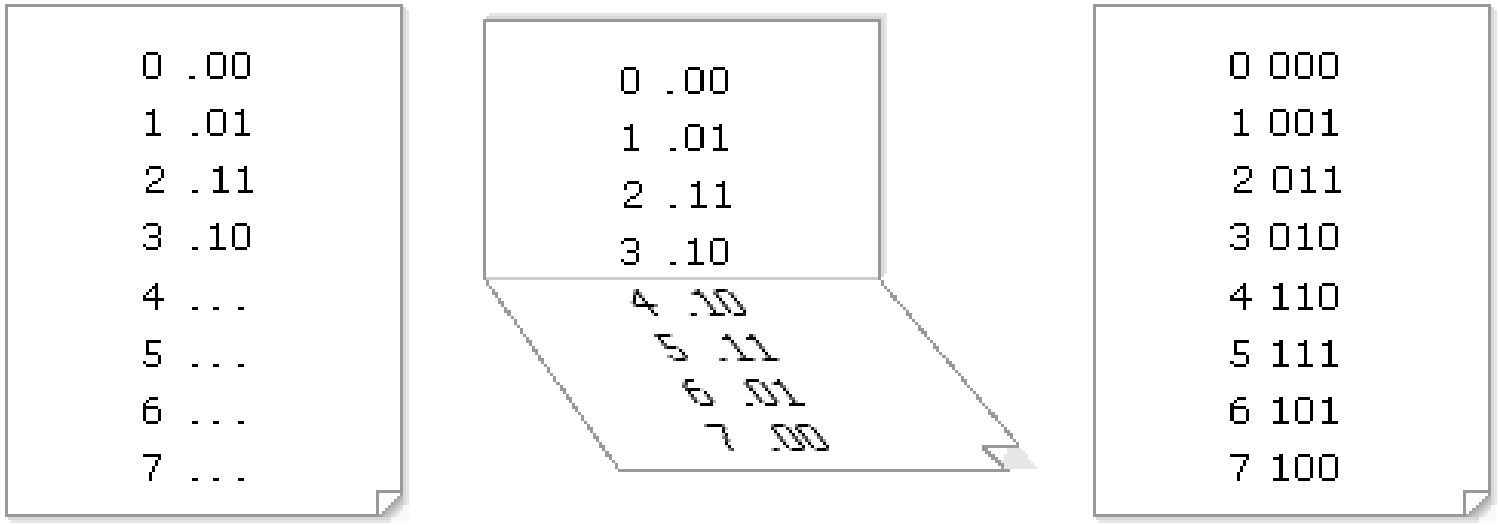


Construction du code Gray : nous avons inversé 1 bits à la fois.

Codage binaire

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 valeur binaires.



Codage binaire

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 valeur binaires.

	b_2	b_1	b_0
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

axe miroir

Construction du code Gray sur 3 bits

Codage binaire

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

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

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** (**L**east **S**ignificant **B**it en anglais et le bit du poids le plus faible) et du bit adjacent supérieur le plus proche du nombre binaire.

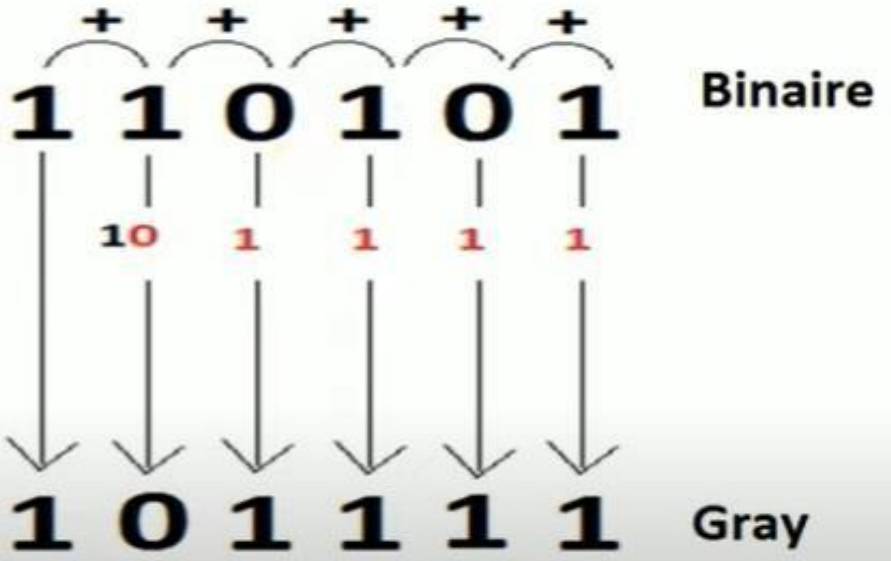
Codage binaire

Code binaire réfléchi (code de GRAY)

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

Exemple :

$(110101)_2 = ?$



$(110101)_2 = (101111)$ Gray

Codage binaire

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

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.

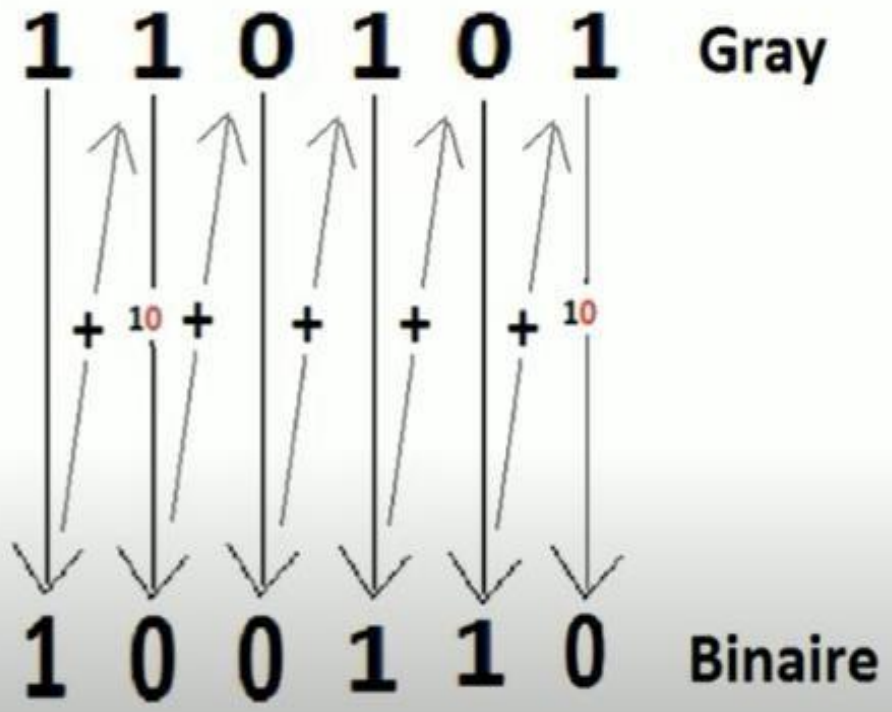
Codage binaire

Code binaire réfléchi (code de GRAY)

Conversion du binaire réfléchi vers binaire pur :

- **Exemple :**

$(110101)_{Gray} = ?$



$(110101)_{Gray} = (100110)2$

Codage binaire

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. chiffre+3).

Codage binaire

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

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
1000	0100	0101

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

Codage binaire

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 soustrairede chaque chiffre le nombre 3 pour obtenir le résultat final qui est le nombre décimal.

Codage binaire

Code binaire à excès de trois

- **Conversion du binaire Excess-3 vers le Décimal**

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

1000	0100	0101
8	4	5
-3	-3	-3
=5	=1	=2

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

Représentation des caractères

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



Représentation des caractères

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





Représentation des caractères

- 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.



Représentation des caractères

- 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

- 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

- 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

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

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			
146	92	k	210	D2	K	64	40	blank
147	93	l	211	D3	L	76	4C	<
148	94	m	212	D4	M	77	4D	(
149	95	n	213	D5	N	78	4E	+
150	96	o	214	D6	O	79	45	|
151	97	p	215	D7	P	80	50	&
152	98	q	216	D8	Q	90	5A	!
153	99	r	217	D9	R	91	5B	$
						92	5C	*
162	A2	s	226	E2	S	93	5D	)
163	A3	t	227	E3	T	94	5E	;
164	A4	u	228	E4	U	96	60	-
165	A5	v	229	E5	V	97	61	/
166	A6	w	230	E6	W	107	6B	,
167	A7	x	231	E7	X	108	6C	%
168	A8	y	232	E8	Y	109	6D	_
169	A9	z	233	E9	Z	110	6E	>
						111	6F	?
122	7A	:	125	7D	'			
123	7B	#	126	7E	=			
124	7C	@	127	7F	"			

Représentation des caractères

Codage ASCII

- Le code ASCII signifie « **American Standard Code for Information Interchange** » (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

Codage ASCII

- Chaque **caractère** à un code **ASCII**



Représentation des caractères

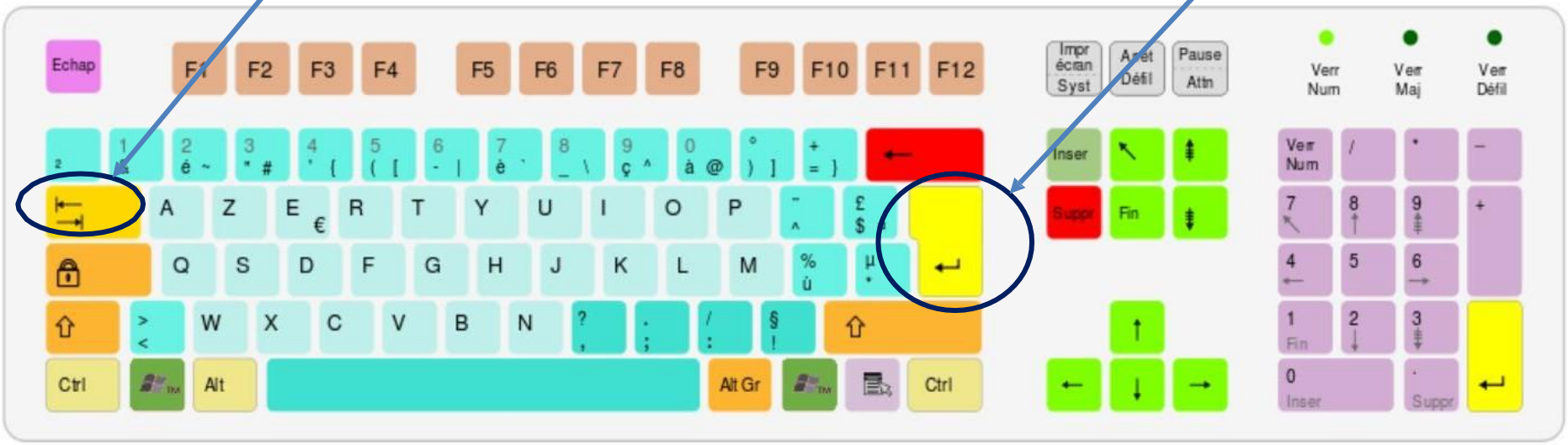
Codage ASCII

- 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 8ème bit est 0.

Représentation des caractères

Codage ASCII

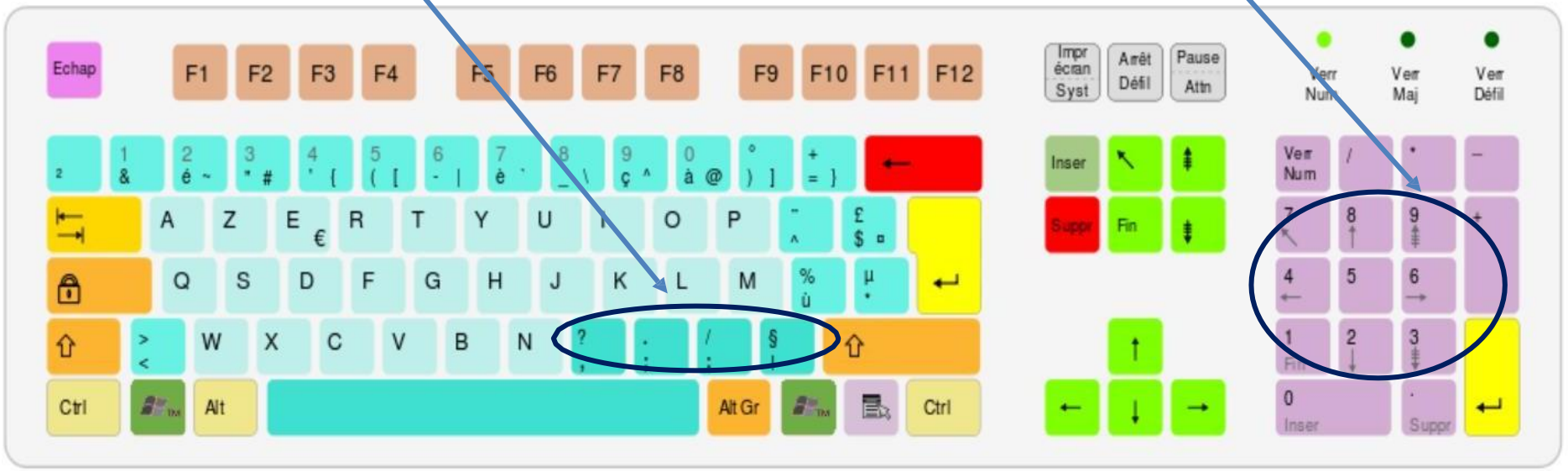
- Ce codage sur 7 bits donne 128 codes différents répartis comme suit :
  - 0 à 31 : caractères de contrôle (retour à la ligne, Tabulation, etc.…)



Représentation des caractères

Codage ASCII

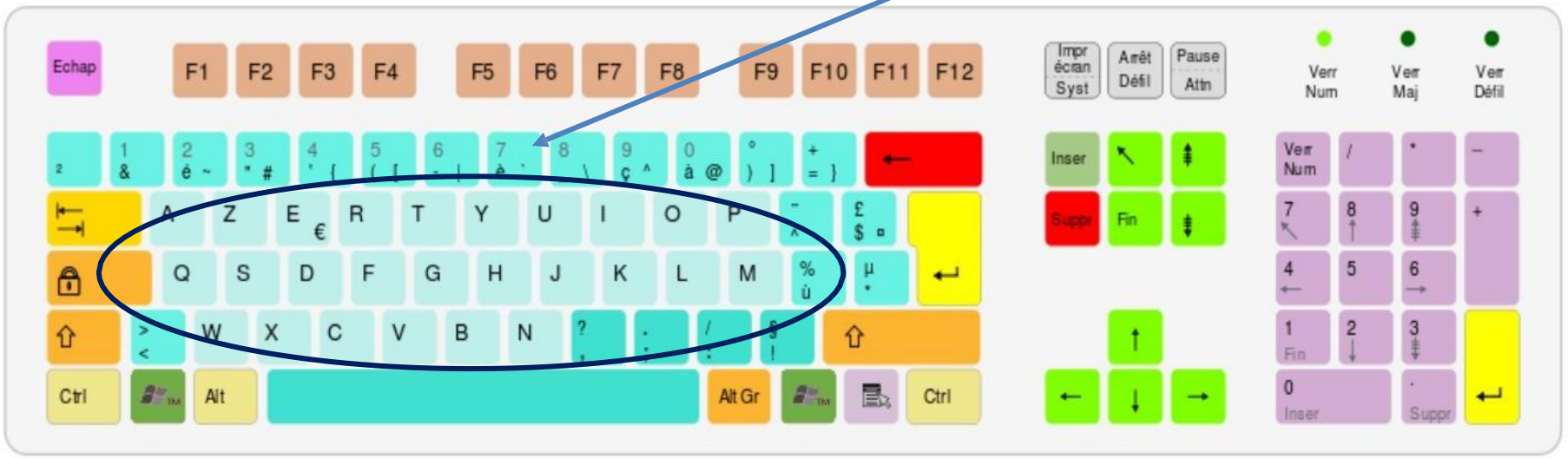
- À partir du code 32, suivent des signes de ponctuation et quelques symboles mathématiques comme ! ou + ou /, puis les chiffres arabes de 0 à 9



Représentation des caractères

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

Codage ASCII

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

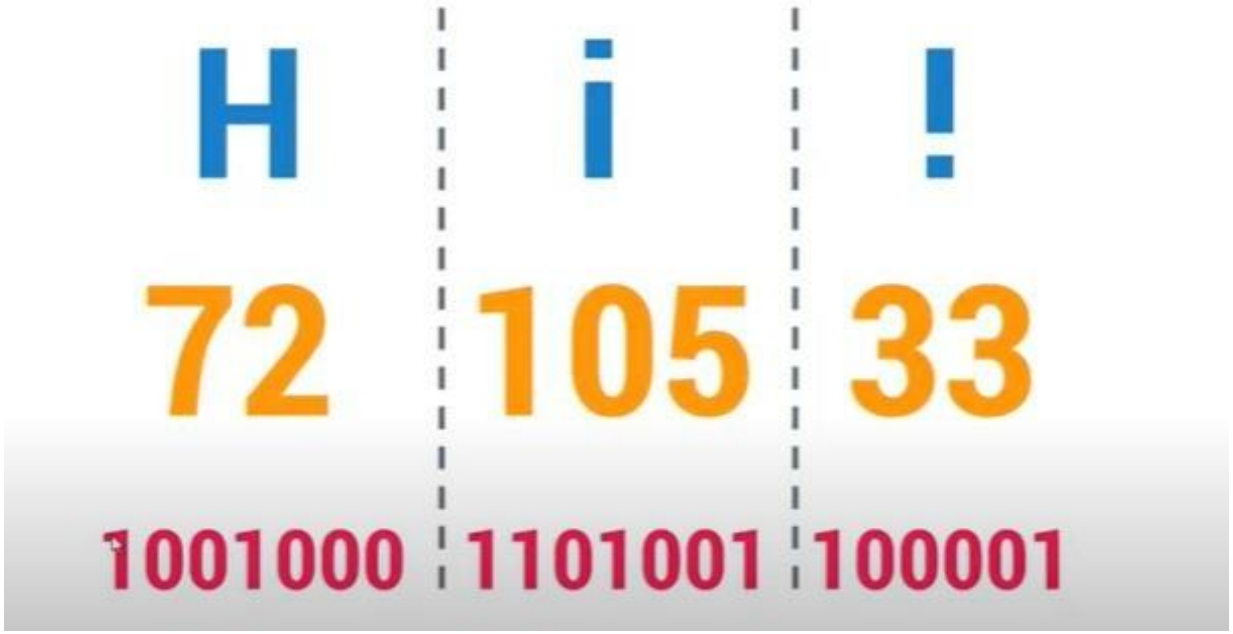


Représentation des caractères

Codage ASCII

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

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



Représentation des caractères

Codage ASCII

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

01001000 01101001 00100001

Représentation des caractères

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

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

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

Codage ASCII

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

65	109	105	83
A	m	i	S

Représentation des caractères

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….



Représentation des caractères

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 donner naissance à l'**UNICODE**.

Représentation des caractères

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

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 « **U**nicode **T**ransformation **F**ormat ». Il s'agit d'une famille de normes pour coder le jeu de caractères Unicode en binaire.

Représentation des caractères

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

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.

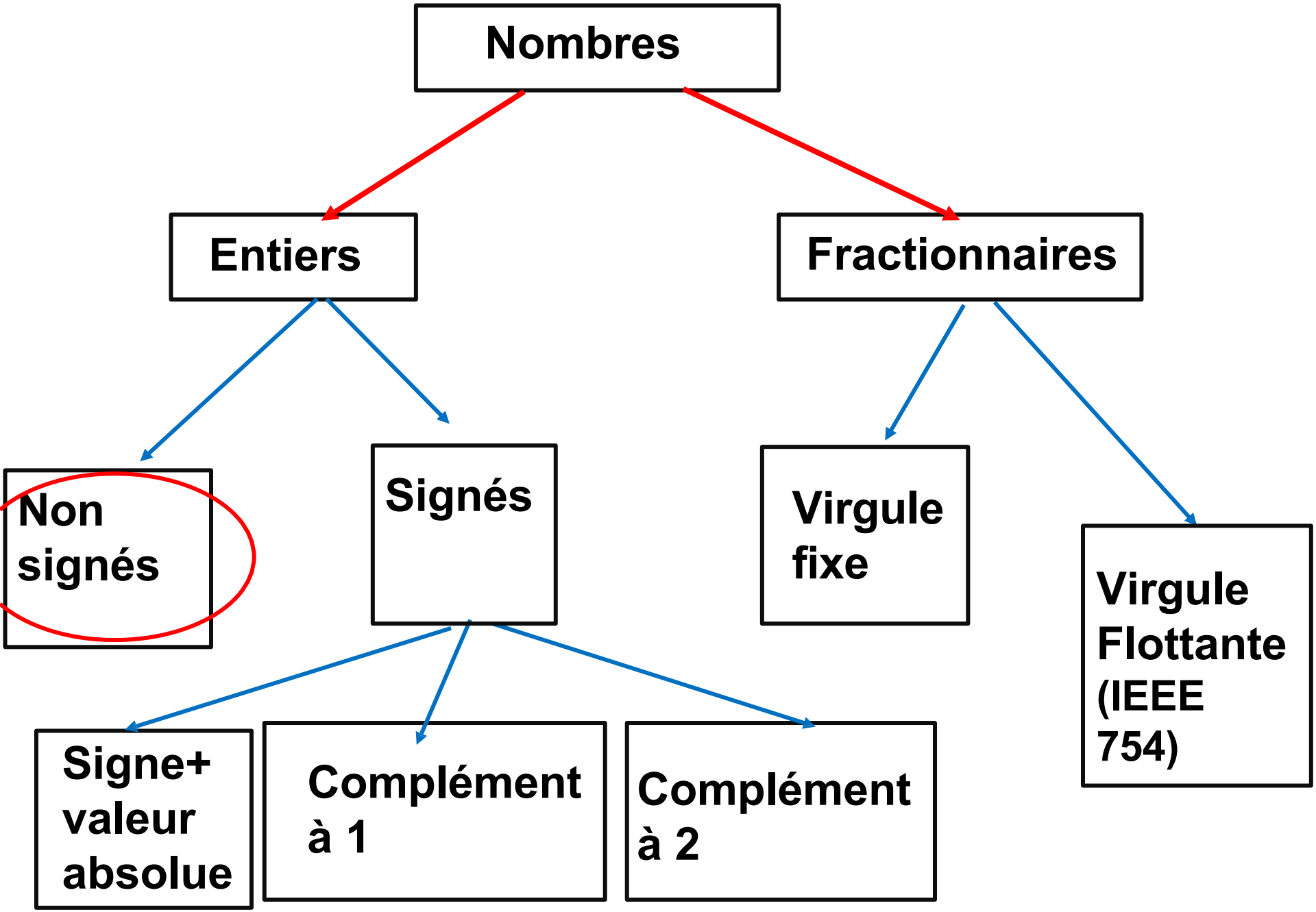
Représentation des caractères

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.

Représentation des nombres

- La représentation (codage) des nombres est nécessaire afin de les stocker et de les manipuler par un ordinateur.
- Le principal problème est la limitation de la taille du codage : un nombre mathématique peut prendre des valeurs arbitrairement grandes, tandis que le codage dans la machine doit s'effectuer sur un nombre fixe de bits.
- Il y a deux types de nombre à représenter : les nombres **entiers** et les nombres **fractionnaires.**


Nombres
Entiers
Fractionnaires
Non signés
Signés
Virgule fixe
Virgule Flottante (IEEE 754)
Signe+ valeur absolue
Complément à 1
Complément à 2

Représentation des nombres

Représentation des entiers :

1. Représentation des entiers non signés :

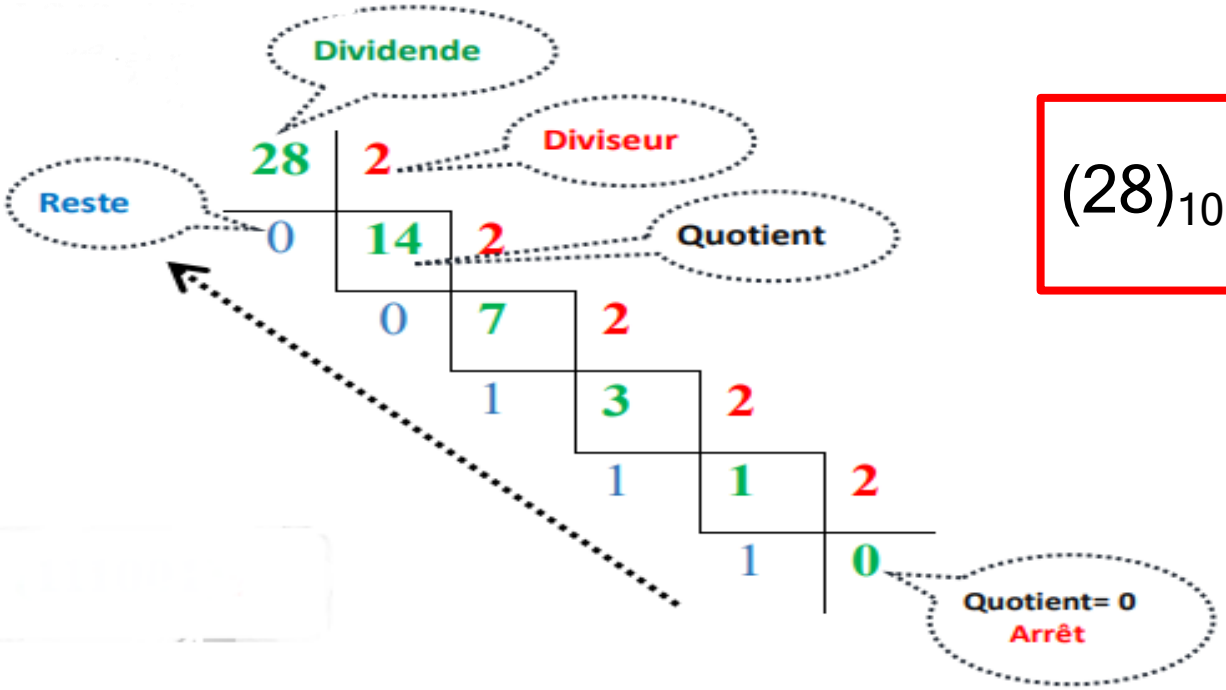
Le codage des entiers non signés en binaire se fait comme on l'a vu précédemment c'est-à-dire division successives par 2 jusqu'à ce que le quotient soit nul et récupérer les restes de bas en haut ensuite les écrire de gauche à droite (conversion du décimal vers le binaire).

Représentation des nombres

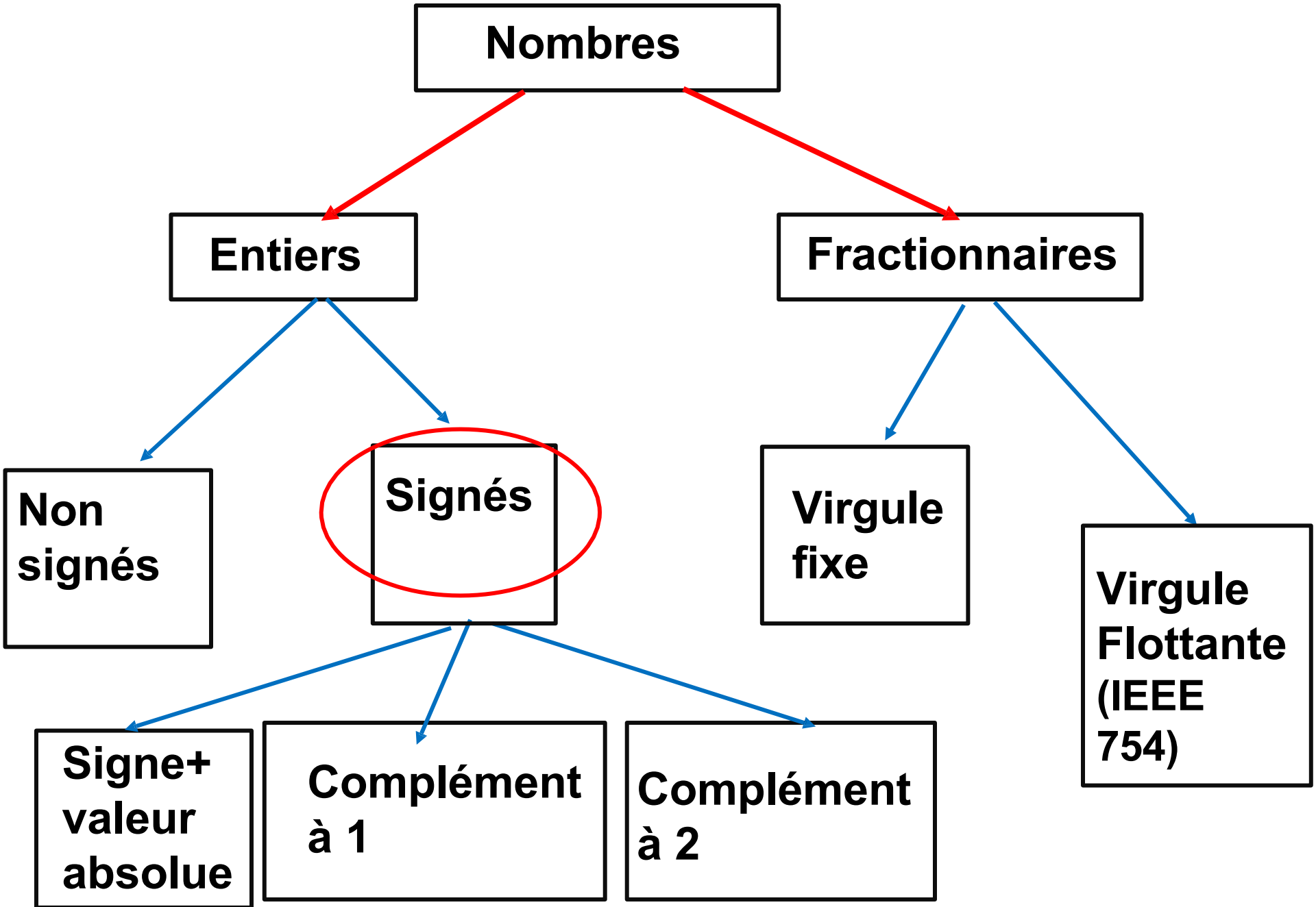
Représentation des entiers :

1. Représentation des entiers non signés :

Exemple : Coder le nombre 28 sur 8 bits



$(28)_{10} = (00011100)_2$


Nombres
Entiers
Fractionnaires
Non signés
Signés
Virgule fixe
Virgule Flottante (IEEE 754)
Signe+ valeur absolue
Complément à 1
Complément à 2

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

- Nous avons jusqu'à présent parlé de nombres entiers naturels (pas de signe). Ils ne peuvent par nature qu'être positifs ou nuls.
- Envisageons maintenant les nombres entiers signés , qui sont des nombres munis d'un signe ‘+’ ou ‘-’.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

- En décimal,
  - +1, +2, +3 etc. sont des nombres positifs
  - -1, -2, -3 etc. sont des nombres négatifs.
- De même en binaire,
  - +1, +10, +11, +100, +101 etc. sont des nombres binaires positifs,
  - -1, -10, -11, -100, -101 etc. sont des nombres binaires négatifs.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

- Le problème est que les circuits électroniques ne peuvent enregistrer que des 0 ou des 1 mais pas de signes + ou -.
- La solution est alors de réserver **un bit pour indiquer le signe**. Reste à déterminer le bit qui dans un nombre binaire conviendrait le mieux pour symboliser le signe et quelle valeur de ce bit (0 ou 1) pour représenter le signe "+" ou "-".

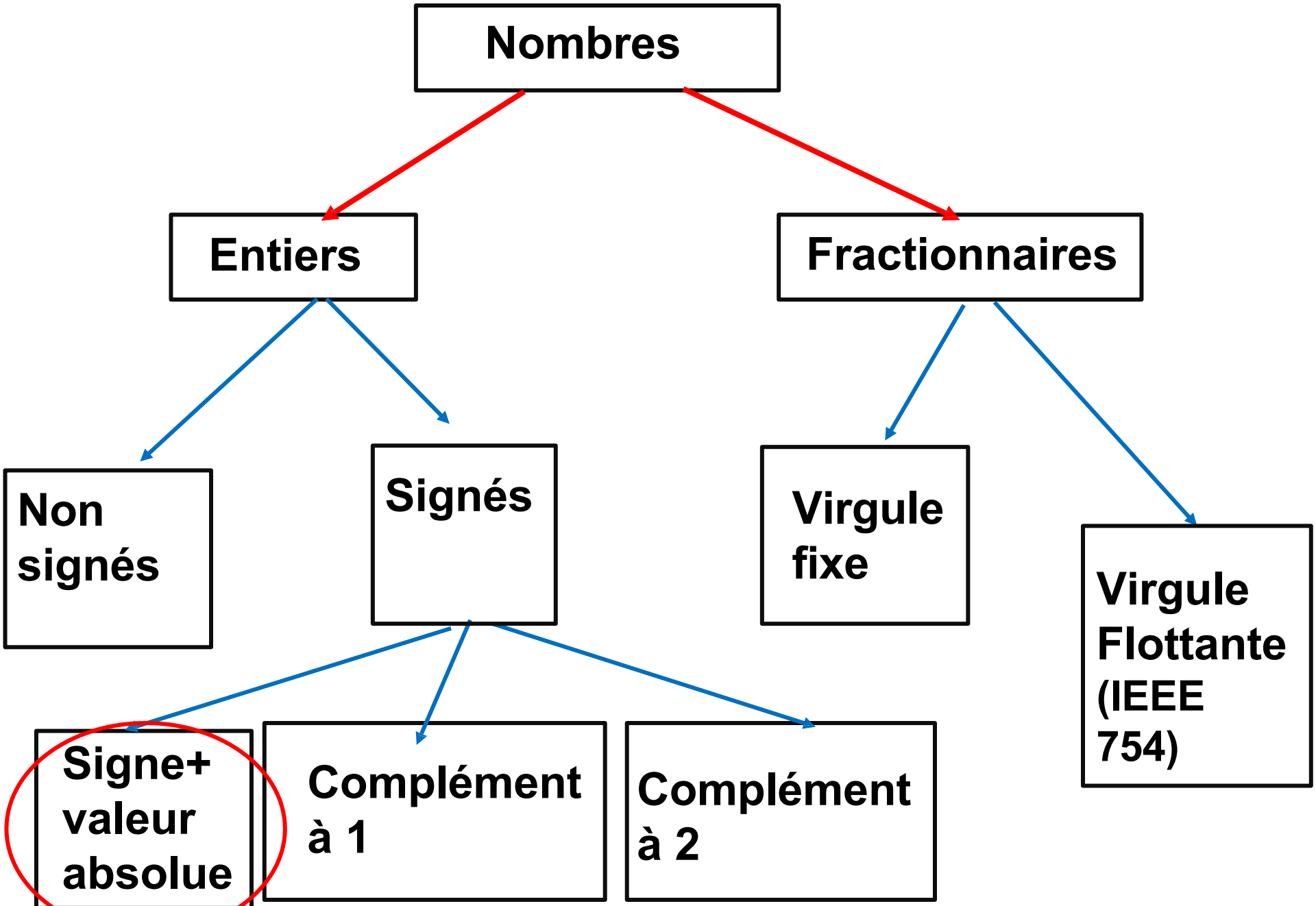
Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Le codage des entiers signés (positifs et négatifs) peut se faire de trois manières :

- Binaire **signé** (ou signe+valeur absolue).
- Binaire **complément à 1** (ou complément restreint ou logique).
- Binaire **complément à deux** (ou complément vrai ou arithmétique)


Nombres
Entiers
Fractionnaires
Non signés
Signés
Virgule fixe
Virgule Flottante (IEEE 754)
Signe+ valeur absolue
Complément à 1
Complément à 2

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire signé :

Dans cette représentation sur un ensemble de bits on réserve le bit le plus à gauche (le bit du poids le plus fort ou MSB) pour le signe et d'attribuer par convention la valeur 1 au signe « - » et la valeur 0 au signe « + » et le reste des bits pour représenter la valeur absolue du nombre en binaire.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire signé :

Exemple : représenter sur 8 bits en binaire signé les valeurs +83, -51, -128.

Sur 8 bits donc 1er bit pour le signe, les 7 bits restants pour la valeur Absolue.

$(83)_{10} = (1010011)_2$

$(83)_{10} = (\mathbf{0}1010011)_{BS}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire signé :

Exemple : représenter sur 8 bits en binaire signé les valeurs +83, -51, -128.

$(-51)_{10} = (?)_{BS}$

$|-51| = (51)_{10} = (0110011)_2$

$(-51)_{10} = (10110011)_{BS}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire signé :

Exemple : représenter sur 8 bits en binaire signé les valeurs +83, -51, -128.

$(-128)_{10} = (?)_{BS}$

$|-128| = (128)_{10} = (10000000)_2$ 

Dépassement de capacité , (128) nécessite au moins 8 bits pour le représenter en plus du bit de signe donc il nous faut un total de 9 bits.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire signé :

- L'Intervalle des nombres : dans cette représentation sur n bits on peut coder les nombres entre $-(2^{n-1}-1)$ et $+(2^{n-1}-1)$.

Exemple : sur 4 bits : entre -7 et +7

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire signé :

Cette représentation présente deux inconvénients:

1- Deux représentations différentes pour le 0 :

par exemple sur 4 bits :

$(+0)_{10} = (0000)_{BS}$ et

$(-0)_{10} = (1000)_{BS}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire signé :

2 - Conflits lors des opérations arithmétiques :

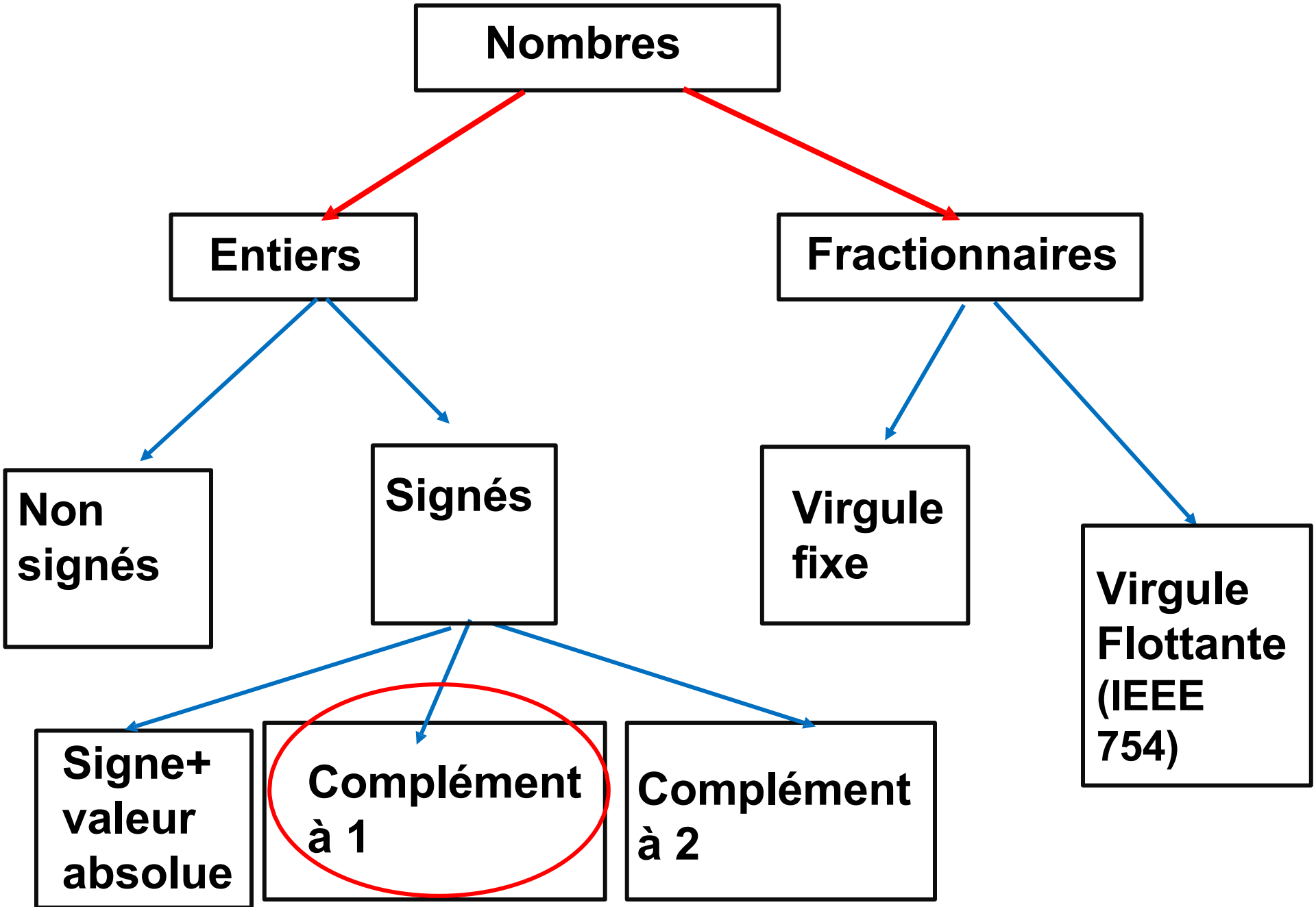
Exemple : Faite l'opération suivante en binaire signé sur 8 bits :

3 – 4 = 3 + (- 4)

3 0000 0011

-4 1000 0100 ⇒

-1 1000 0111 $(-7)_{10}$ faux


Nombres
Entiers
Fractionnaires
Non signés
Signés
Virgule fixe
Virgule Flottante (IEEE 754)
Signe+ valeur absolue
Complément à 1
Complément à 2

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1: (Complément Logique ou **Complément Restreint**):

Les nombres positifs sont codés de la même façon qu'en binaire pur.

Un nombre négatif est codé en inversant chaque bit de la représentation de sa valeur absolue et le bit le plus à gauche est utilisé pour représenter le signe du nombre : s'il est = 1 alors le nombre est négatif sinon le nombre est positif.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1:

Exemple :

Coder les nombres $(+33)_{10}$ et $(-33)_{10}$ en complément à un sur 8 bits :

$(33)_{10}=(100001)_2$

- $(+33)_{10}=(\mathbf{0}0100001)_{BS}= (\mathbf{0}0100001)_{CA1}$
- $(-33)_{10} =(\mathbf{1}0100001)_{BS} = (\mathbf{1}1011110)_{CA1}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1:

- L'Intervalle des nombres : dans cette représentation sur n bits on peut coder les nombres entre $-(2^{n-1}-1)$ et $+(2^{n-1}-1)$.

Exemple : sur 4 bits : entre -7 et +7

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1:

Cette représentation possède aussi deux inconvénients :

1- Deux représentations différentes pour le 0 : par exemple sur 4 bits : $(+0)_{10} = (0000)_{CA1}$ et $(-0)_{10} = (1111)_{CA1}$

2- Opérations arithmétiques difficiles.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1: Addition

L'addition en complément à 1 est effectuée comme le binaire pur en suivant les règles suivante :

- Deux nombres de **signes différents** : pas de dépassement de capacité et si on trouve une retenue on l'ajoute au résultat.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1: Addition

• Deux nombres de **même signes « positifs »** : risque de dépassement de capacité et si on trouve une retenue on l'ajoute au résultat, si le bit du signe est 1 alors Overflow

• Deux nombres de **même signes « négatifs »** : risque de dépassement de capacité et si on trouve une retenue on l'ajoute au résultat, si le bit du signe est 0 alors Overflow.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1: Addition

- **Exemple** : Sur 4 bits effectuer les opérations suivantes en CA1 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

$2\text{-}5 = (+2)_{10} + (-5)_{10} = (0010)_{CA1} + (1010)_{CA1}$
$= (1100)_{CA1} = (-3)_{10}$

$2+5 = (+2)_{10} + (+5)_{10} = (0010)_{CA1} + (0101)_{CA1}$
$= (0111)_{CA1} = (+7)_{10}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1: Addition

- **Exemple** : Sur 4 bits effectuer les opérations suivantes en CA1 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

$$-2+5 = (-2)_{10} + (+5)_{10} = (1101)_{CA1} + (0101)_{CA1}$$
$$= (0010 + 1_{\text{de retenue}})_{CA1}$$
$$= (0011)_{CA1} = (+3)_{10}$$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1: Addition

- **Exemple :** Sur 4 bits effectuer les opérations suivantes en CA1 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

$-2-5 = (-2)_{10} + (-5)_{10}$

$= (1101)_{CA1} + (1010)_{CA1}$

$= (0111 + 1_{\text{de retenue}})_{CA1}$

$=(1000)_{CA1}$

$= (-7)_{10}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1: Addition

- Exemple : Sur 4 bits effectuer les opérations suivantes en

CA1 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

$5+3 = (+5)_{10} + (+3)_{10}$

$= (0101)_{CA1} + (0011)_{CA1}$

$= (1000)_{CA1}$ -> Overflow

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 1: Addition

- Exemple : Sur 4 bits effectuer les opérations suivantes en

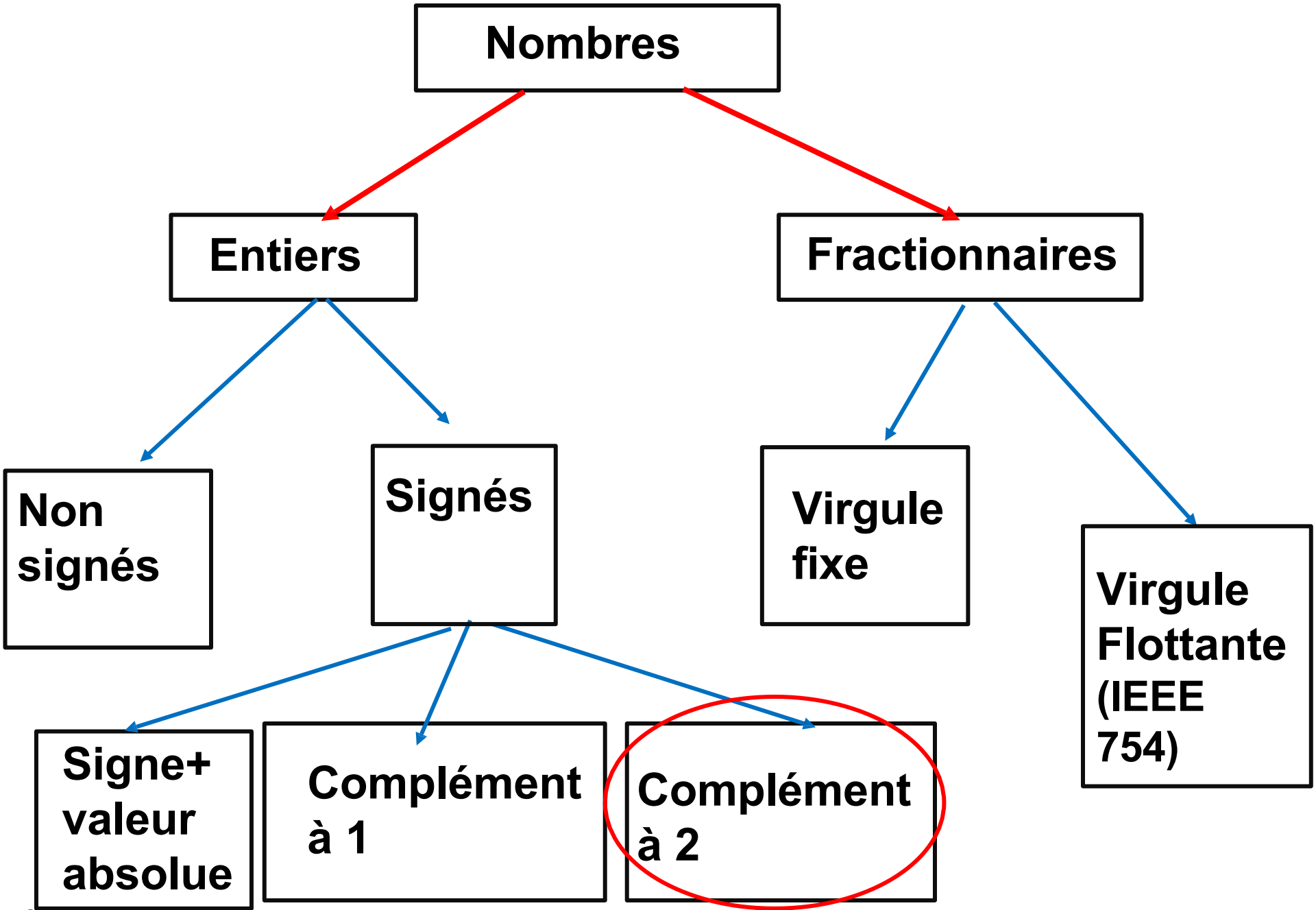
CA1 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

- $5-3 = (-5)_{10} + (-3)_{10}$

$= (1010)_{CA1} + (1100)_{CA1}$

$= (0110 + 1_{\text{de retenue}})_{CA1}$

$= (0111)_{CA1}$ -> Overflow


Nombres
Entiers
Fractionnaires
Non signés
Signés
Virgule fixe
Virgule Flottante (IEEE 754)
Signe+ valeur absolue
Complément à 1
Complément à 2

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: (Complément vrais ou **complément arithmétique) :**

Les nombres **positifs** sont codés de la même façon qu'en binaire pur.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: (Complément vrais ou **complément arithmétique) :**

Un nombre **négatif** est codé en inversant chaque bit de la représentation de sa valeur absolue puis on ajoute 1 (c.à.d. : **CA2=CA1+1**) et le bit le plus à gauche est utilisé pour représenter le signe du nombre : s'il est = 1 alors le nombre est négatif sinon le nombre est positif.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2:

Exemple : Coder les nombres $(+33)_{10}$ et $(-33)_{10}$ en complément à deux sur 8 bits :

$(+33)_{10} = (00100001)_{BS}$

$= (00100001)_{CA1}$

$= (00100001)_{CA2}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2:

Exemple : Coder les nombres $(+33)_{10}$ et $(-33)_{10}$ en complément à deux sur 8 bits :

$(-33)_{10} = (10100001)_{BS}$

$= (11011110)_{CA1} + 1$

$= (11011111)_{CA2}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2:

Une autre méthode pour calculer le complément à 2 consiste à démarrer du bit le plus faible :

- Si le bit est un zéro, on recopie le « 0 » jusqu'au premier 1 rencontré.
- Si c'est un « 1 » on garde ce premier 1 ensuite on inverse tous les bits restant .

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2:

Exemple : représenter $(-33)_{10}$ sur 8 bits en Cà2

$(-33)_{10} = (10100001)_{BS}$

$= (11011111)_{CA2}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2:

Un seul codage pour 0. Par exemple sur 8 bits : +0 ou -0 = $(00000000)_{CA2}$

L'intervalle des nombres : dans cette représentation sur n bits on peut coder les nombres entre $-(2^{n-1})$ a $(2^{n-1}-1)$

Exemple : Sur 8 bits on peut coder des nombres de -128 a +127.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: Addition

L'Addition est effectuée comme le binaire pur en suivant les règles suivantes :

• Deux nombres de **signes différents** : il n'y a pas de dépassement de capacité et si on trouve une retenue on l'ignore et le résultat est en CA2.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: Addition

- Deux nombres **négatifs** : si on trouve une retenue on l'ignore, si le bit du signe du résultat est 0 alors Overflow et le résultat est en CA2.
- Deux nombres **positifs** : si on trouve une retenue on l'ignore et si le bit du signe du résultat est 1 alors Overflow et le résultat est en CA2.

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: Addition

- **Exemple** : Sur 4 bits effectuer les opérations suivantes en CA2 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

$2-5 = (+2)_{10} + (-5)_{10}$

$= (0010)_{CA2} + (1011)_{CA2}$

$= (1101)_{CA2}$

$= (-3)_{10}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: Addition

- **Exemple** : Sur 4 bits effectuer les opérations suivantes en CA2 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

$2+5 = (+2)_{10} + (+5)_{10}$

$= (0010)_{CA2} + (0101)_{CA2}$

$= (0111)_{CA2}$

$= (+7)_{10}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: Addition

- **Exemple** : Sur 4 bits effectuer les opérations suivantes en CA2 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

$-2+5 = (-2)_{10} + (+5)_{10}$

$= (1110)_{CA2} + (0101)_{CA2}$

$= (1_{\text{ignoré}}\ 0011)_{CA2}$

$= (0011)_{CA2}$

$= (+3)_{10}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: Addition

- **Exemple** : Sur 4 bits effectuer les opérations suivantes en CA2 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

$-2-5 = (-2)_{10} + (-5)_{10}$

$= (1110)_{CA2} + (1011)_{CA2}$

$= (1_{\text{ignoré}}\ 1001)_{CA2} = (1001)_{CA2}$

$= (-7)_{10}$

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: Addition

- **Exemple** : Sur 4 bits effectuer les opérations suivantes en CA2 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

$5+3 = (+5)_{10} + (+3)_{10}$

$= (0101)_{CA2} + (0011)_{CA2}$

$= (1000)_{CA2}$ -> Overflow

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Types de représentation	Intervalle de nombres	Exemples
Binaire Signé	$[-(2^{n-1}-1) , +(2^{n-1}-1)]$	•Sur 4 bits : [-7,+7] •Sur 8 bits [-127,127]
Complément à un	$[-(2^{n-1}-1) , +(2^{n-1}-1)]$	•Sur 4 bits : [-7,+7] •Sur 8 bits [-127,127]
Complément à deux	$[-2^{n-1} , +(2^{n-1}-1)]$	•Sur 4 bits : [-8,+7] •Sur 8 bits [-128,127]

Représentation des nombres

Représentation des entiers :

2. Représentation des entiers signés :

Binaire complément à 2: Addition

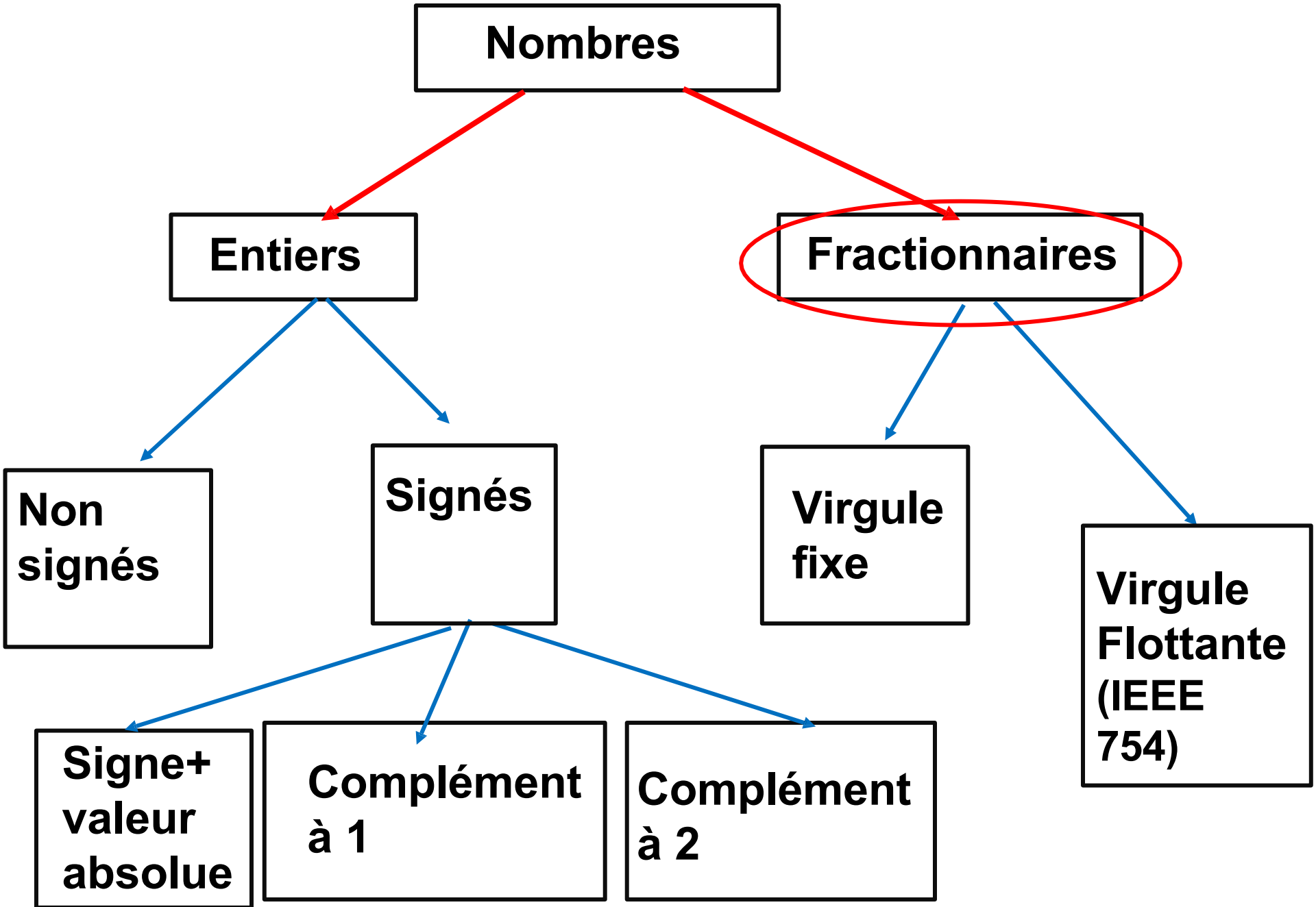
- **Exemple** : Sur 4 bits effectuer les opérations suivantes en CA2 : 2-5, 2+5, -2+5, -2-5, 5+3, -5-3

-5-3 = $(-5)_{10} + (-3)_{10}$

= $(1011)_{CA2} + (1101)_{CA2}$

= $(1_{\text{ignoré}}\,1000)_{CA2}$

= $(-8)_{10}$ Selon la propriété du CA2.


Nombres
Entiers
Fractionnaires
Non signés
Signés
Virgule fixe
Virgule Flottante (IEEE 754)
Signe+ valeur absolue
Complément à 1
Complément à 2

Représentation des nombres

Représentation des nombres fractionnaires :

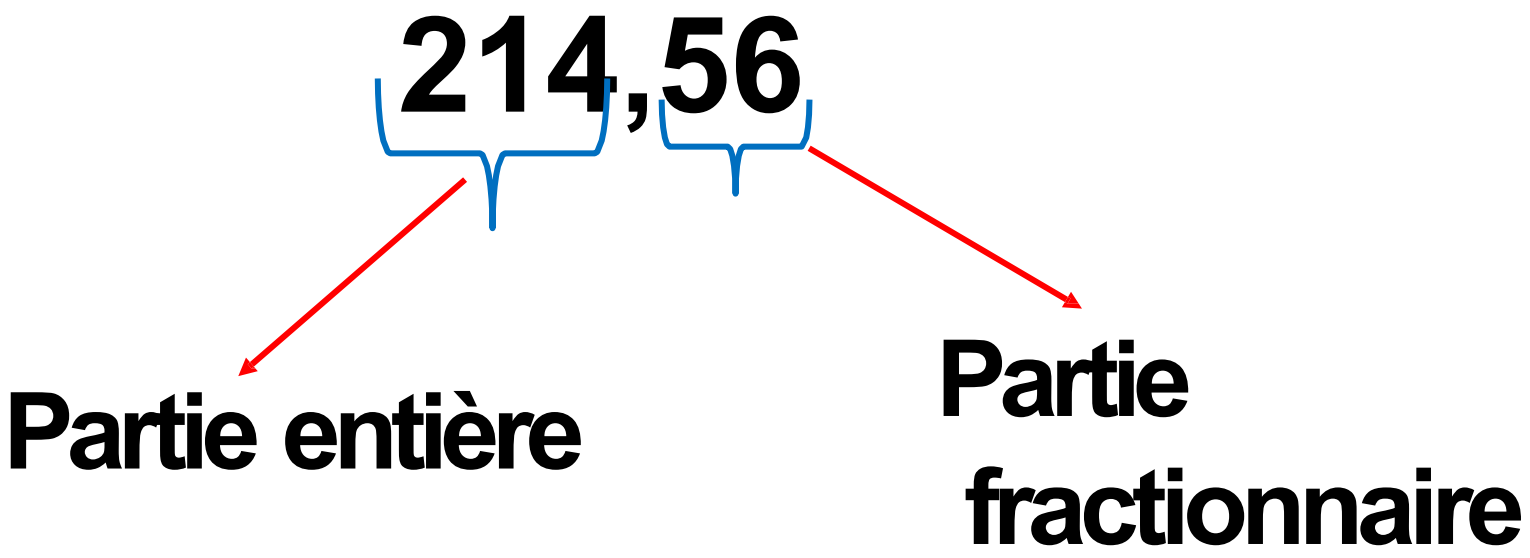
- Les nombres **fractionnaires** sont les nombres qui comportent des **chiffres après la virgule.**

- Un nombre fractionnaire est composé de deux parties :
  - Une partie **entière**.
  - Une partie **fractionnaire** ou décimale (à ne pas confondre avec le système décimal).

Représentation des nombres

Représentation des nombres fractionnaires :

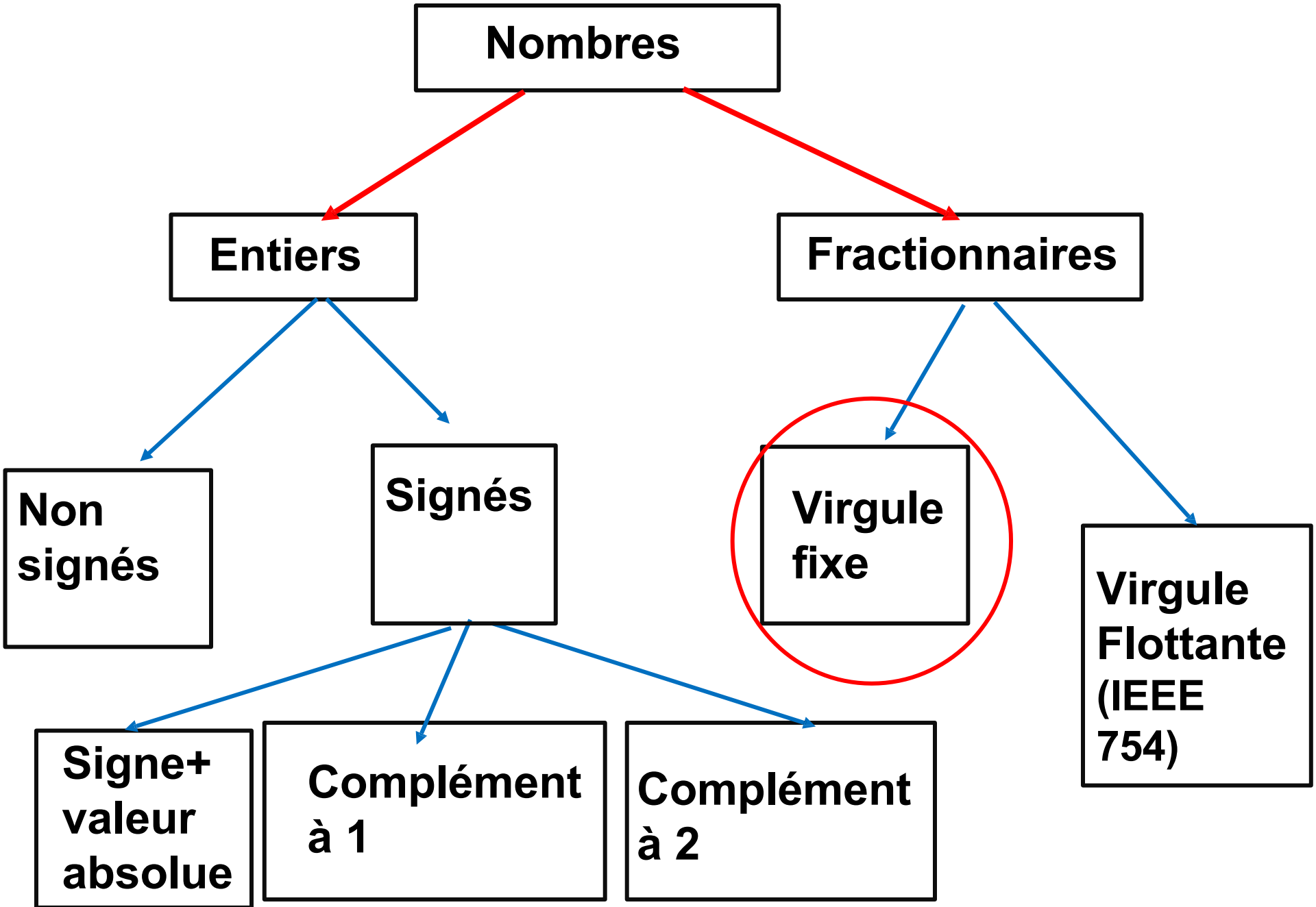
- **Exemple :**



Représentation des nombres

Représentation des nombres fractionnaires :

- Il existe deux méthodes pour représenter les nombres fractionnaires (réels) en binaire :
  - **Virgule fixe** : la position de la virgule est fixe (ne change pas).
  - **Virgule flottante**: la position de la virgule est dynamique (change).

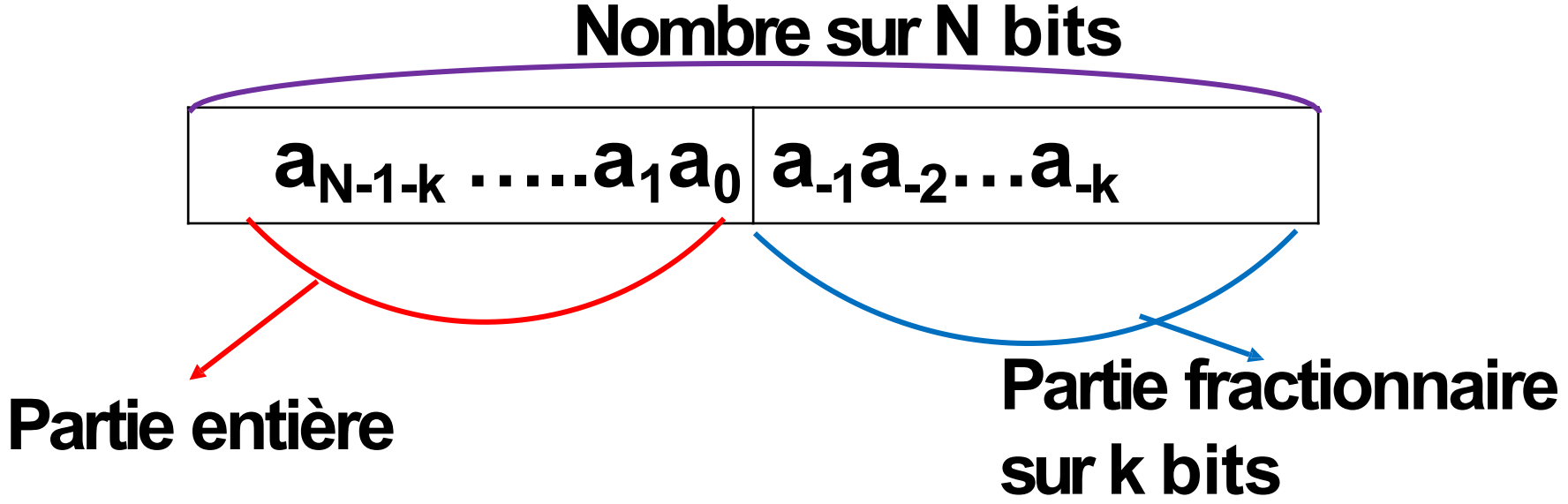

Nombres
Entiers
Fractionnaires
Non signés
Signés
Virgule fixe
Virgule Flottante (IEEE 754)
Signe+ valeur absolue
Complément à 1
Complément à 2

Représentation des nombres

Représentation des nombres fractionnaires :

1. Représentation binaire en virgule fixe :

- La représentation en virgule fixe (ou format fixe) d'un nombre binaire sur **N bits** comprend une **partie entière** suivie d'une **partie fractionnaire**.



Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

1. Conversion du décimal vers une base B :

- Pour la partie **entière**,on fait comme pour les entiers (**divisions successives** par B)
- Pour la partie **décimale** :

– On **multiplie** la partie entière par B.
– On note la partie entière obtenue.
– On recommence avec la partie décimale restante.

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

1. Conversion du décimal vers une base B :

– On s'arrête quand la partie décimale est nulle ou quand la précision souhaitée est atteinte.

- La partie décimale est la concaténation des parties entières obtenues dans l'ordre de leur calcul.

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

1. Conversion du décimal vers une base B:

Exemple 1 : conversion du décimal au binaire

$(4.25)_{10} = (?)_2$

- $(4)_{10}=(100)_2$
- 0.25 * 2 = **0.**5 | 0
- 0.5 * 2 = **1.**0 | 1 , partie fractionnaire nulle donc arrêt .

$(4.25)_{10} = (100.01)_2$

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

1. Conversion du décimal vers une base B:

Exemple 2 : conversion du décimal au binaire

coder $(27.87)_{10}$ en binaire avec 6 bits pour la partie entière et 4 bits pour la partie fractionnaire

- $(27)_{10}=(011011)_2$
- 0.87 * 2 = **1.**74 1
- 0.74 * 2 = **1.**48 1
- 0.48 * 2 = **0.**96 0
- 0.96 * 2 = **1.**92 1 (le nombre de bits pour la partie fractionnaire est atteint)

$(27)_{10}=(011011.11O1)_2$

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

1. Conversion du décimal vers une base B:

Exemple 2 : coder $(27.87)_{10}$ en binaire avec 6 bits pour la partie entière et 4 bits pour la partie fractionnaire

Donc $(27.87)_{10} = (011011.1101)_2$

Nous remarquons que nous pouvons obtenir une suite périodique (ou une suite illimitée non périodique) lors de la conversion en binaire d'un nombre décimal.

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

1. Conversion du décimal vers une base B:

Exemple 3 : conversion du décimal vers l'octal

Coder $(46.73)_{10}$ en octal avec 4 chiffres dans la partie fractionnaire

$(46)_{10} = (56)_8$

0.73 x 8 = **5**.84

0.84 x 8 = **6**.72

0,72 x 8= **5**.76

0.76 x 8 = **6**.08

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

1. Conversion du décimal vers une base B:

Exemple 3 : conversion du décimal vers l'octal

Coder $(46.73)_{10}$ en octal avec 4 chiffres dans la partie fractionnaire

le nombre de chiffres dans la partie fractionnaire est atteint , donc on arrête

$(46.73)_{10} = (56.5656)_8$

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

1. Conversion du décimal vers une base B:

Exemple 4 : conversion du décimal vers l'hexadécimal

Coder $(46.73)_{10}$ en hexadécimal avec 4 chiffres dans la partie fractionnaire

$(46)_{10} = (2E)_{16}$

0.73 x 16 = **11**.68 -> B

0.68 x 16 = **10**.88 -> A

0,88 x 16= **14**.08 -> E

0.08 x 16 = **1**.28 -> 1

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

1. Conversion du décimal vers une base B:

Exemple 4 : conversion du décimal vers l'hexadécimal

Coder $(46.73)_{10}$ en hexadécimal avec 4 chiffres dans la partie fractionnaire

le nombre de chiffres dans la partie fractionnaire est atteint , donc on arrête

$(46.73)_{10} = (2E.BAE1)_{16}$

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

2. Conversion d'une base B vers la décimal :

- Soit N un nombre fractionnaire dans une base B comme suit :

  $(N)_B = (a_{n-1} \ldots\ldots a_1 a_0 . a_{-1} a_{-2} \ldots a_{-p})_B$

- Le passage d'un nombre fractionnaire de la base B (2,8 ou 16) vers la base 10 est défini par la formule polynomiale suivante :

$$(N)_{10} = (a_{n-1}b^{n-1} + \ldots + a_1 b^1 + a_0 b^0 + a_{-1}b^{-1} + a_{-2}b^{-2} \ldots + a_{-p}b^{-p})_B$$

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

2. Conversion d'une base B vers la décimal :

Exemple 1 : conversion du binaire au décimal

$(101.011)_2 = (\ ?\ )_{10}$

$$\begin{array}{cccccccc} & \mathbf{2} & \mathbf{1} & \mathbf{0} & \mathbf{-1} & \mathbf{-2} & \mathbf{-3} & \\ (1 & 0 & 1 & . & 0 & 1 & 1 &)_2 \end{array}$$

$(101.011)_2 = 1*2^2+0*2^1+1*2^0+0*2^{-1}+1*2^{-2}+1*2^{-3}$

$= 4+0+1+0+0.25+0.125 = (5.375)_{10}$

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

2. Conversion d'une base B vers la décimal :

Exemple 2 : conversion de l'octal au décimal

$(123.26)_8 = (\ ?)_{10}$

$$\begin{array}{cccccc} & \mathbf{2} & \mathbf{1} & \mathbf{0} & \mathbf{-1} & \mathbf{-2} \\ (& 1 & 2 & 3 . & 2 & 6 \ )_8 \end{array}$$

$(123.26)_8 = 1*8^2+2*8^1+3*8^0+2*8^{-1}+6*8^{-2}$

$=64+16+3+0.25+0.09375= (83.34375)_{10}$

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

2. Conversion d'une base B vers la décimal :

Exemple 3 : conversion de l'hexadécimal au décimal

$(AB3.E5)_{16} = (?)_{10}$

$$\begin{array}{cccccc} & 2 & 1 & 0 & -1 & -2 \\ & A & B & 3 & . & E & 5 \end{array}_{16}$$

$$(AB3.E5)_{16} = A*16^2+B*16^1+3*16^0+E*16^{-1}+5*16^{-2}$$
$$=10*16^2+11*16^1+3*16^0+14*16^{-1}+5*16^{-2}$$
$$=2560+176+3+0.875+0.01953125$$
$$\approx (2\,739.895)_{10}$$

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

3. Conversion d'une base 2 vers la base 8 ou 16 et inversement :

Le principe de regroupement des bits est appliqué :

• Dans la partie **entière** le regroupement se fait de **droite à gauche** et s'il manque des bits on ajoute des 0 à gauche.

• Dans la partie **fractionnaire** le regroupement se fait de **gauche à droite** et s'il manque des bits on ajoute des 0 à droite.

Représentation des nombres

Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

3. Conversion d'une base 2 vers la base 8 ou 16 et inversement :

Exemple : Coder $(1101101.110011)_2$ en octal et en hexadécimal

- $(1101101.110011)_2 = (\ 001\ 101\ 101\ .\ 110\ 011\ )_2$

$1\quad 5\quad 5\ .\quad 6\quad 3$

$(1101101.110011)_2 = (155.63)_8$

Représentation des nombres

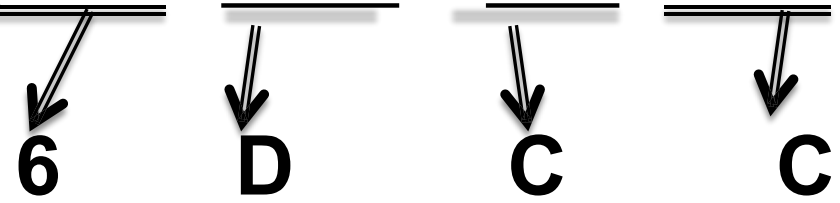
Représentation des nombres fractionnaires :

Conversion des nombres fractionnaires:

3. Conversion d'une base 2 vers la base 8 ou 16 et inversement :

Exemple : Coder $(1101101.110011)_2$ en octal et en hexadécimal

- $(1101101.110011)_2 = (\ 0110\ 1101.\ 1100\ 1100\ )_2$



$(1101101.110011)_2 = (6D.CC)_{16}$

Représentation des nombres

Représentation des nombres fractionnaires :

1. Représentation binaire en virgule fixe :

- Le problème qui se pose est que les réels peuvent être, positifs ou négatifs, c'est la convention **complément à deux** qui est utilisé pour représenter ces nombres en binaire.
- Pour convertir la partie fractionnaire (décimal en binaire ou inversement), on utilise les **puissances de 2 négatives.**

Représentation des nombres

Représentation des nombres fractionnaires :

1. Représentation binaire en virgule fixe :

1. Conversion du décimal au binaire :

Exemple 1 : Coder le nombre $(+3.5)_{10}$ en binaire en virgule fixe sur 8 bits (3 bits pour la partie entière et le reste pour la partie décimale)

$(+3)_{10} = (011)_{Cà2}$

0.5 * 2 = 1.0

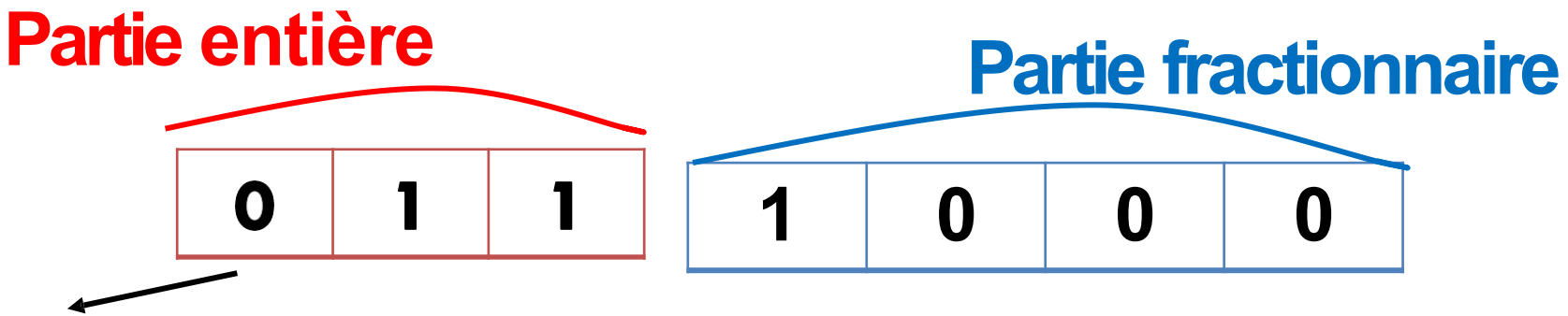
Représentation des nombres

Représentation des nombres fractionnaires :

1. Représentation binaire en virgule fixe :

1. Conversion du décimal au binaire :

Exemple 1 :



Bit de signe est 0 : nombre positif

$$(+\ 3.5)_{10} = (011.1000)_2$$

Représentation des nombres

Représentation des nombres fractionnaires :

1. Représentation binaire en virgule fixe :

1. Conversion du décimal au binaire :

Exemple 2 : Coder le nombre $(-6.625)_{10}$ en binaire en virgule fixe sur 8 bits (4 bits pour la partie entière et le reste pour la partie décimale)

$(-6)_{10} = (1010)_{Cà2}$

- 0.625 * 2 = **1.**25
- 0.25 * 2 = **0.**5
- 0.5 * 2 = **1**.0
- $(-6.625)_{10}$ **=1010.1010**

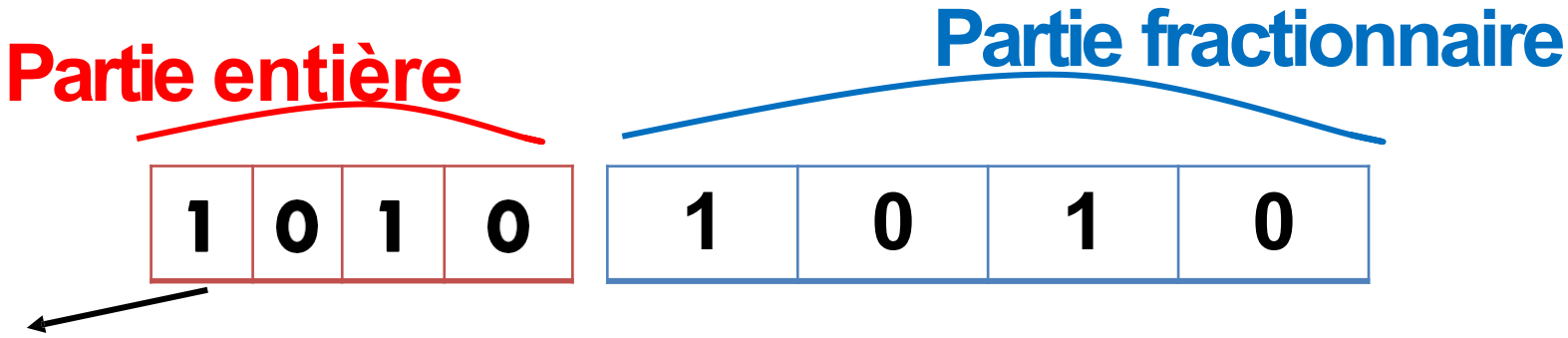
Représentation des nombres

Représentation des nombres fractionnaires :

1. Représentation binaire en virgule fixe :

1. Conversion du décimal au binaire :

Exemple 2 :



Bit de signe est 1 : nombre négatif

$$(-6.625)_{10} = (1010.1010)_2$$

Représentation des nombres

Représentation des nombres fractionnaires :

1. Représentation binaire en virgule fixe :

1. Conversion du binaire au décimal :

Exemple : Coder le nombre $(1001.11)_2$ en binaire en virgule fixe sur 8 bits (4 bits pour la partie entière et le reste pour la partie décimale)

- **Partie entière** : $(1001)_{Cà2} = (-111)_2 = (-7)_{10}$
- **Partie fractionnaire :**

$0.11 = 1*2^{-1}+1*2^{-2}$

$=0.5+0.25$

$= 0.75$

$(1001.11)_2 = (-7.75)_{10}$

Représentation des nombres

Représentation des nombres fractionnaires :

1. Représentation binaire en virgule fixe :

La représentation en virgule fixe présente de nombreux défauts :

– L'espace réservé à la partie « fractions » limite le nombre de bits réservés à la partie entière. Ce qui est **gênant pour représenter de très grands nombres**,

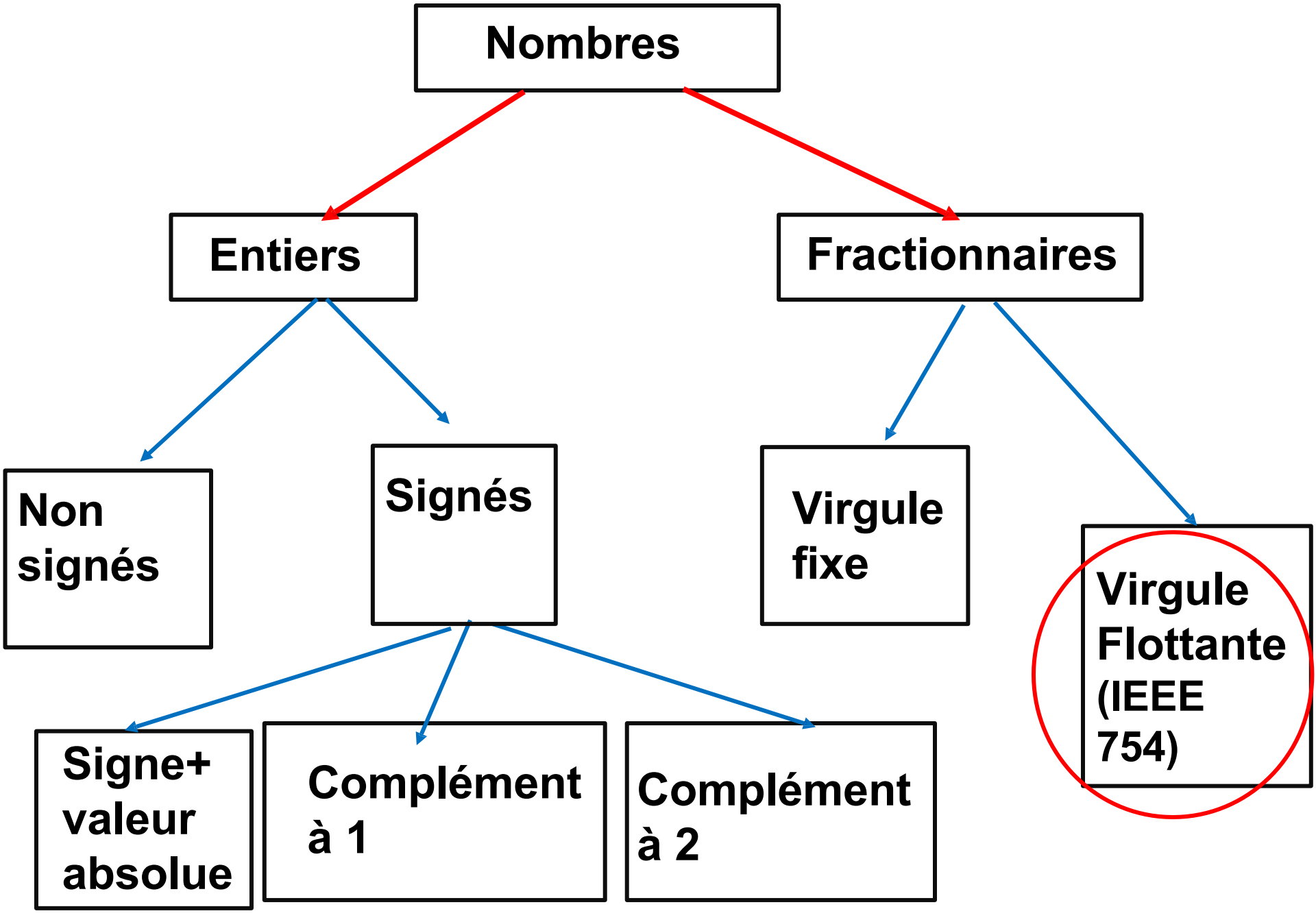
– Inversement, la **précision sur de très petits nombres est limitée** par le manque d'espace dans la partie fractionnaire alors que pour œs nombres, la partie entière ne contient que des zéros.

Représentation des nombres

Représentation des nombres fractionnaires :

1. Représentation binaire en virgule fixe :

D'où l'idée de ne pas fixer la position de la virgule, on parle alors de représentation à **virgule flottante** ou de **nombre flottant**.


Nombres
Entiers
Fractionnaires
Non signés
Signés
Virgule fixe
Virgule Flottante (IEEE 754)
Signe+ valeur absolue
Complément à 1
Complément à 2

Représentation des nombres

Représentation des nombres fractionnaires :

2. Représentation binaire en virgule flottante :

En informatique, pour représenter les nombres à virgule flottante, on utilise une représentation similaire à la «notation scientifique» des calculatrices, sauf qu'elle est en base deux et non en base dix. Il s'agit de la représentation en virgule flottante, d'où le type float.

Représentation des nombres

Représentation des nombres fractionnaires :

2. Représentation binaire en virgule flottante :

Un tel nombre est représentésous la forme suivante :

$$Smx2^{ne}$$

S : **Signe** du nombre

m : **Mantisse** du nombre comprise $(1 \leq m < 2)$

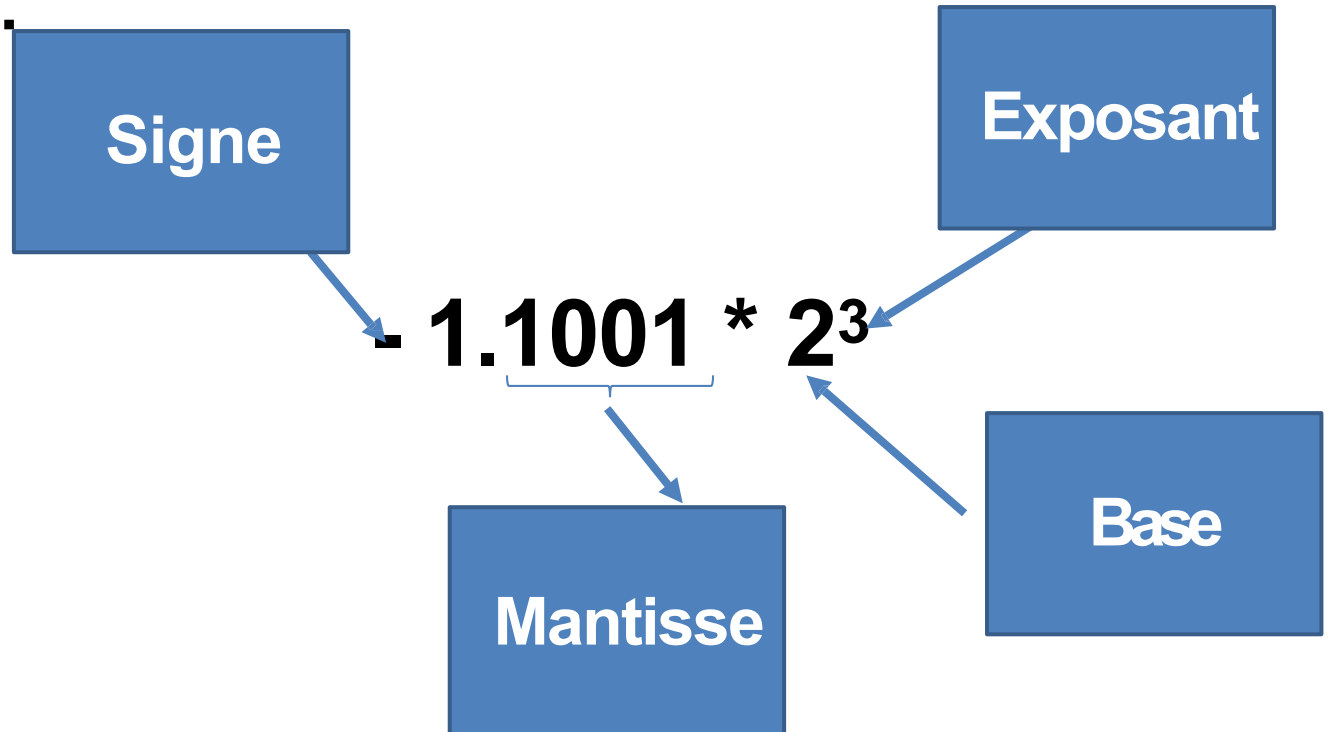
e **Exposant** du nombre (entier relatif)

Représentation des nombres

Représentation des nombres fractionnaires :

2. Représentation binaire en virgule flottante :

Un tel nombre est représenté sous la forme suivante :



Représentation des nombres

Représentation des nombres fractionnaires :

2. Représentation binaire en virgule flottante :

- Inspiré de l'écriture scientifique
- **Exemple:** $173{,}95 = +\ 1{,}7395 \times 10^2$
- Généralisation: soit x un réel

$$x = \text{signe mantisse} \times 10^e$$

- **Avantage:** permet de représenter des nombres très grands et très petits sans s'encombrer de zéros

Représentation des nombres

Représentation des nombres fractionnaires :

2. Représentation binaire en virgule flottante :

- Appliqué en base 2 l'écriture devient alors:

signe mantisse x 2^e

Avec la mantisse et l'exposant en binaire

- Pour écrire un nombre flottant en respectant la norme IEEE754, il est nécessaire de commencer par écrire le nombre sous la forme 1,XXXXX.2^e(avec e l'exposant), il faut obligatoirement qu'il y ait un seul chiffre à gauche de la virgule et il faut que ce chiffre soit un "1".

Représentation des nombres

Représentation des nombres fractionnaires :

2. Représentation binaire en virgule flottante :

- Appliqué en base 2 l'écriture devient alors:

signe mantisse x 2^e

Avec la mantisse et l'exposant en binaire

Exemple 1 :

1100.101 = 1.100101 * 2^3 (déplacement de la virgule de trois positions vers la **gauche** c'est pourquoi on multiplie le nombre binaire résultant par 2 qu'on a élevé à la **puissance positive** 3)

Représentation des nombres

Représentation des nombres fractionnaires :

2. Représentation binaire en virgule flottante :

- Appliqué en base 2 l'écriture devient alors:

signe mantisse x 2^e

Avec la mantisse et l'exposant en binaire

Exemple 2 :

$(0.011)_2$ = 1.1*2^{-2} (déplacement de la virgule de deux positions vers la **droite** c'est pourquoi on multiplie le nombre binaire résultant par 2 qu'on a élevé à la **puissance négative** 2)

Représentation des nombres

Représentation des nombres fractionnaires :

2. Représentation binaire en virgule flottante :

• A la fin des années 70, chaque ordinateur avait sa propre représentation pour les nombres à virgule flottante. Il y a donc eu la nécessité de normaliserle codage des nombres flottants , afin que la représentation ne varie pas d'un matériel à l'autre → apparition de la norme **IEEE 754.**

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

• IEEE signifie «Institute of Electrical and Electronics Engineers » ou en français « l'Institut des ingénieurs électriciens et électroniciens », qui est une association professionnellequi a pour mission promouvoir la connaissance de l'ingénierie liée à l'électricité et l'électronique y compris les télécommunications et l'informatique.

•Cette norme à été identifié par le numéro 754, d'où le nom **IEEE 754.**

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 : En général le codage d'un nombre réel en virgule flottante se présente comme suit

signe mantisse x 2^e

Signe: + est représenté par 0 (positif) et le signe − par 1 (négatif)

• **Mantisse** appartient à l'intervalle [1; 2[, contient la partie fractionnaire du nombre écrit avec la notation scientifique (Le seul chiffre avant la virgule étant toujours **1**, il n'est pas représenter)

• **e** : **Exposant** est un entier relatif et il est établi de manière à ce que la mantisse soit de la forme « 1,… »

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 : Plusieurs formats:

- **Format Simple précision :** 32 bits (soit 4 octets)
  - 1 bit de signe,
  - 8 bits d'exposant,
  - 23 bits de mantisse
- **Format Double précision :** 64 bits (soit 8 octets)
  - 1 bit de signe,
  - 11 bits d'exposant,
  - 52 bits de mantisse

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Simple précision :



Utilisé pour le type "float" (simple précision)

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 : Simple précision :

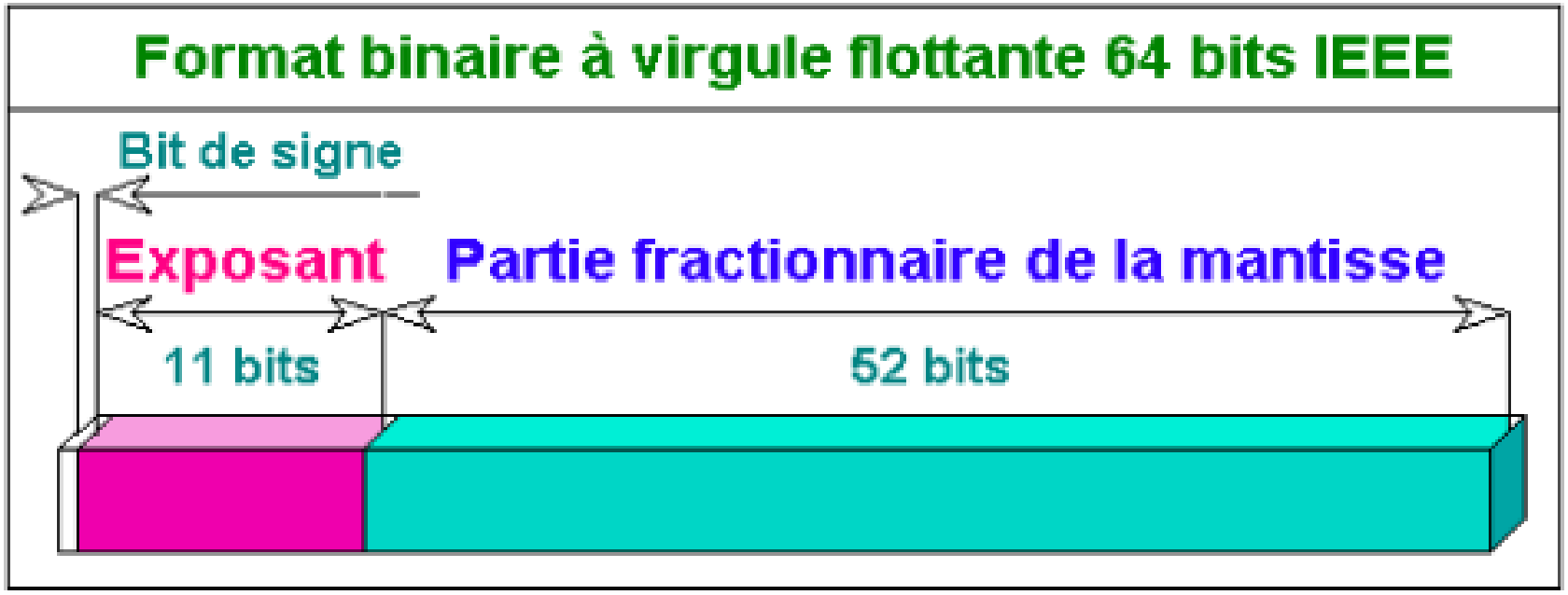
- **Bit de signe**: Le signe est codé sur **1 bit** ayant le poids fort :
  - Le signe – : bit 1
  - Le signe + : bit 0
- **L'exposant** codé sur **8 bits** et contient la valeur de la puissance qui élève le nombre 2.
- **La mantisse** codé sur **23 bits** et contient la partie fractionnaire du nombre écrit avec la notation scientifique.

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Double précision :



Utilisé pour le type "double" (double précision)

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 : Double précision :

- **Bit de signe**: Le signe est codé sur **1 bit** ayant le poids fort :
  - Le signe – : bit 1
  - Le signe + : bit 0
- **L'exposant** codé sur **11 bits** et contient la valeur de la puissance qui élève le nombre 2.
- **La mantisse** codé sur **52 bits** et contient la partie fractionnaire du nombre écrit avec la notation scientifique.

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754 :

Soit N un nombre réel qu'on veut l'écrire avec la norme IEEE 754, il y a 5 étapes à suivre :

1. Détecter le **signe** de N.
2. **Convertir** N en binaire (partie entière : division successives par 2 et partie fractionnaire multiplication par 2).
3. **Normaliser** le nombre $(N)_2$ c.à.d. l'écrire sous forme d'une écriture scientifique.
4. **Calculer l'exposant biaisé (décalé) :**
   - Eb=**127**+e pour la **simple précision**
   - Eb=**1023**+e pour la **double précision**
5. Faire **remplir les champs** selon la précision utilisée.

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Pourquoi l'exposant est biaisé ?

• Pour le format **Simple précision** (respectivement **Double précision**) l'exposant est codé sur 8 bits (respectivement **11 bits**) donc la valeur minimale est 00000000 ce qui vaut 0 en décimal (respectivement 00000000000 ce qui vaut 0 en décimal) et la valeur maximale est 11111111 ce qui vaut 255 en base 10 (respectivement 11111111111 ce qui vaut 2047) donc pas de nombres négatifs.

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Pourquoi l'exposant est biaisé ?

• L'idée consiste à diviser l'intervalle en 2, tous les nombres qui se trouvent droites sont considérés positifs et ceux qui se trouvent à gauche sont considérés négatifs.

• La valeur du milieu est calculé par la formule suivante : **$D = 2^{n-1}-1$ (avec n le nombre de bits de l'exposant)**

Représentation des nombres

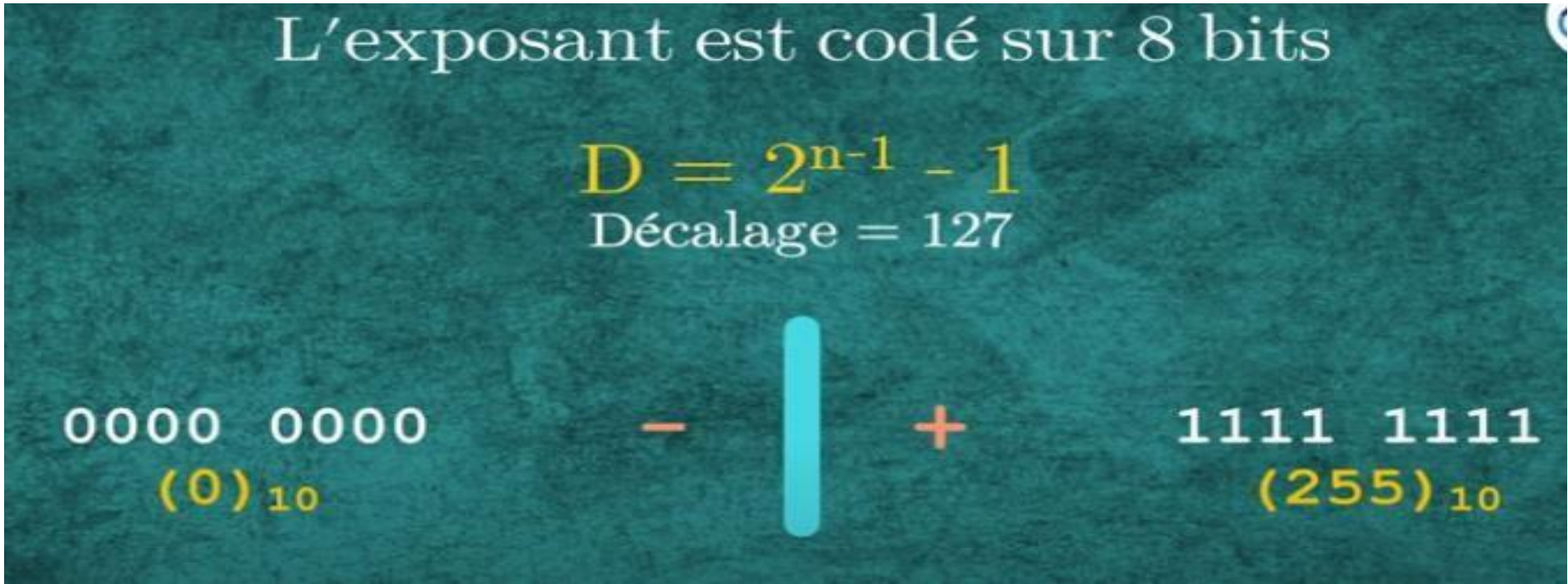
Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Pourquoi l'exposant est biaisé ?

- La valeur du milieu est calculé par la formule suivante : **$D = 2^{n-1}-1$ (avec n le nombre de bits de l'exposant)**



Représentation des nombres

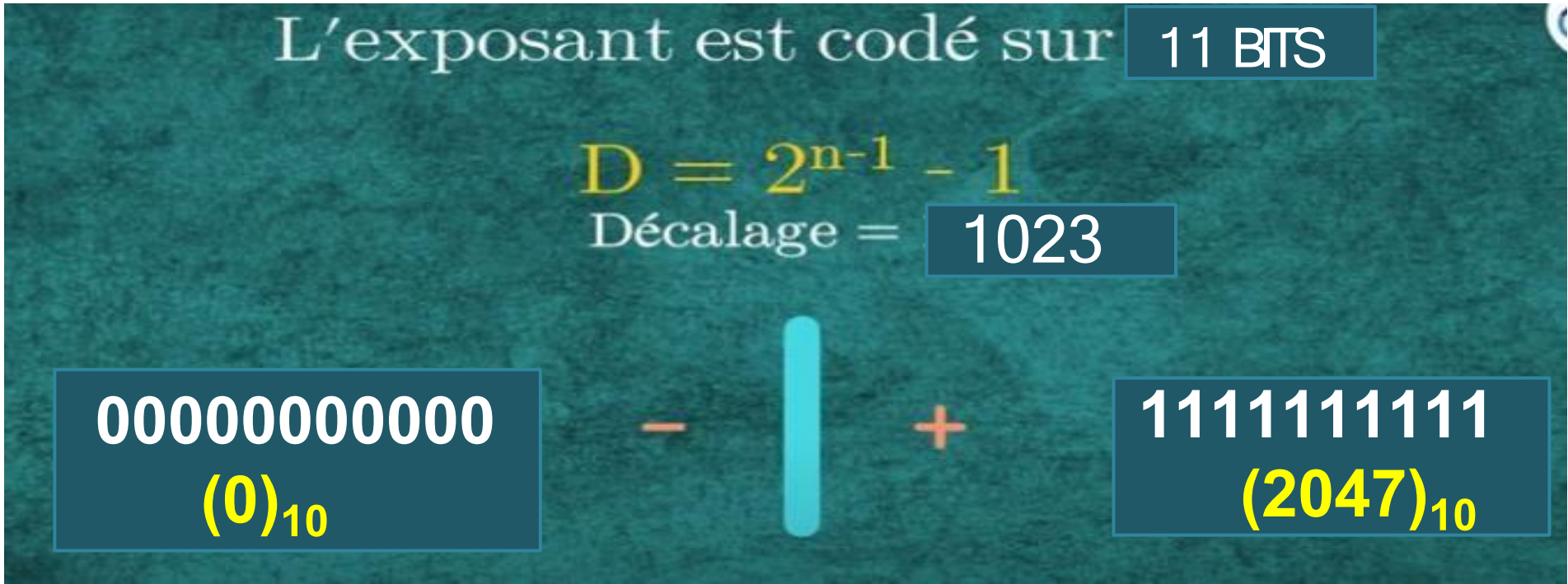
Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Pourquoi l'exposant est biaisé ?

- La valeur du milieu est calculé par la formule suivante : **$D = 2^{n-1}-1$ (avec n le nombre de bits de l'exposant)**



Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple1 : Simpleprécision: Représenterle nombreN = 12.625 on utilisant la norme IEEE 754 à simple précision.

- **Etape 01:** Détecter le signe de N.

12.625 est un nombre positif donc Signe=0

- **Etape 02 :** Ecrire N en binaire.

$(12.625)_{10}=(1100.101)_2$ (représentation réelle en virgule fixe)

- **Etape 03 :** Normaliser le nombre

$(12.625)_{10} = 1.100101 * 2^3$ (représentation scientifique)

Il faut toujours laisser un 1 dans la partie entière.

- **Etape 04 :** Calculer l'exposant biaisé Eb.

On a e = 3 d'où Eb = 127 + e = 127 + 3 = 130.

$130 = (10000010)_2$

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple1: Simple précision: Représenter le nombre N = 12.625 on utilisant la norme IEEE 754 à simple précision.

- **Etape 05:**

Signe	Exposant biaisé	Mantisse
0	10000010	10010100000000000000000

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple1: Simple précision: Représenter le nombre N = 12.625 on utilisant la norme IEEE 754 à simple précision.

- **Etape 05:**

0100 0001 0100 1010 0000 0000 0000 0000

$(414A0000)_{16}$

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple2 : Simpleprécision: Représenterle nombreN = 0.375 on utilisant la norme IEEE 754 à simple précision.

- **Etape 01:** Détecter le signe de N.

0.375 est un nombre positif donc Signe=0

- **Etape 02 :** Ecrire N en binaire.

$(0.375)_{10}=(0.011)_2$ (représentation réelle en virgule fixe)

- **Etape 03 :** Normaliser le nombre

$(0.375)_{10} = 1.1 * 2^{-2}$ (représentation scientifique)

Il faut toujours laisser un 1 dans la partie entière.

- **Etape 04 :** Calculer l'exposant biaisé Eb.

On a e = -2 d'où Eb = 127 + e = 127 +(-2) = 127 – 2 =125.

$125 = (1111101)_2$

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple2 : Simpleprécision: Représenterle nombreN = 0.375 on utilisant la norme IEEE 754 à simple précision.

- **Etape 05:**

Signe	Exposant biaisé	Mantisse
0	01111101	10000000000000000000000





Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple 1 : Double précision : Représenterle nombreN = 12.625 on utilisant la norme IEEE 754 à double précision.

- **Etape 01:** Détecter le signe de N.

12.625 est un nombre positif donc Signe=0

- **Etape 02 :** Ecrire N en binaire.

$(12.625)_{10}=(1100.101)_2$ (représentation réelle en virgule fixe)

- **Etape 03 :** Normaliser le nombre

$(12.625)_{10} = 1.100101 * 2^3$ (représentation scientifique)

Il faut toujours laisser un 1 dans la partie entière.

- **Etape 04 :** Calculer l'exposant biaisé Eb.

On a e = 3 d'où Eb = 1023 + e = 1023 + 3 = 1026.

$1026 = (10000000010)_2$

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple1: Double précision: Représenter le nombre N = 12.625 on utilisant la norme IEEE 754 à double précision.

- **Etape 05:**

S	Exposant biaisé	Mantisse
0	10000000010	1001010000000000000000000000000000000000000000000000

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple1: Double précision: Représenter le nombre N = 12.625 on utilisant la norme IEEE 754 à double précision.

- **Etape 05:**



Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple 2 : Double précision : Représenter le nombre N = - 12.625 on utilisant la norme IEEE 754 à double précision.

- **Etape 01:** Détecter le signe de N.

-12.625 est un nombre négatif donc Signe=1

- **Etape 02 :** Ecrire N en binaire.

$(12.625)_{10}=(1100.101)_2$ (représentation réelle en virgule fixe)

- **Etape 03 :** Normaliser le nombre

$(12.625)_{10} = 1.100101 * 2^3$ (représentation scientifique)

Il faut toujours laisser un 1 dans la partie entière.

- **Etape 04 :** Calculer l'exposant biaisé Eb.

On a e = 3 d'où Eb = 1023 + e = 1023 + 3 = 1026.

$1026 = (10000000010)_2$

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple 2: Double précision : Représenter le nombre N = - 12.625 on utilisant la norme IEEE 754 à double précision.

- **Etape 05:**

S	Exposant biaisé	Mantisse
1	10000000010	10010100000000000000000000000000000000000000000000000000

Représentation des nombres

Représentation des nombres fractionnaires :

2. La norme IEEE 754 :

Conversion du décimal vers la norme IEEE754

Exemple 2: Double précision : Représenterle nombre N=- 12.625 on utilisant la norme IEEE 754 à double précision.

- **Etape 05:**



Conclusion

Dans ce chapitre nous avons vu les notions suivantes :

- **Codage binaire** : code binaire pur , code BCD et code de Gray (binaire réfléchi), excess de trois.
- **Codage des caractères** : EBCDIC, ASCII, UNICODE.
- **Représentations des nombres signés :** Binaire Signé (Signe+valeur absolue), Complément à un, Complément à deux et l'addition en CA1 et l'addition en CA2.
- **Représentation des nombres réels :** virgule fixe, virgule flottante norme IEEE 754 en format simple précision et format double précision.