477

666

Représentation des nombres entiers

1

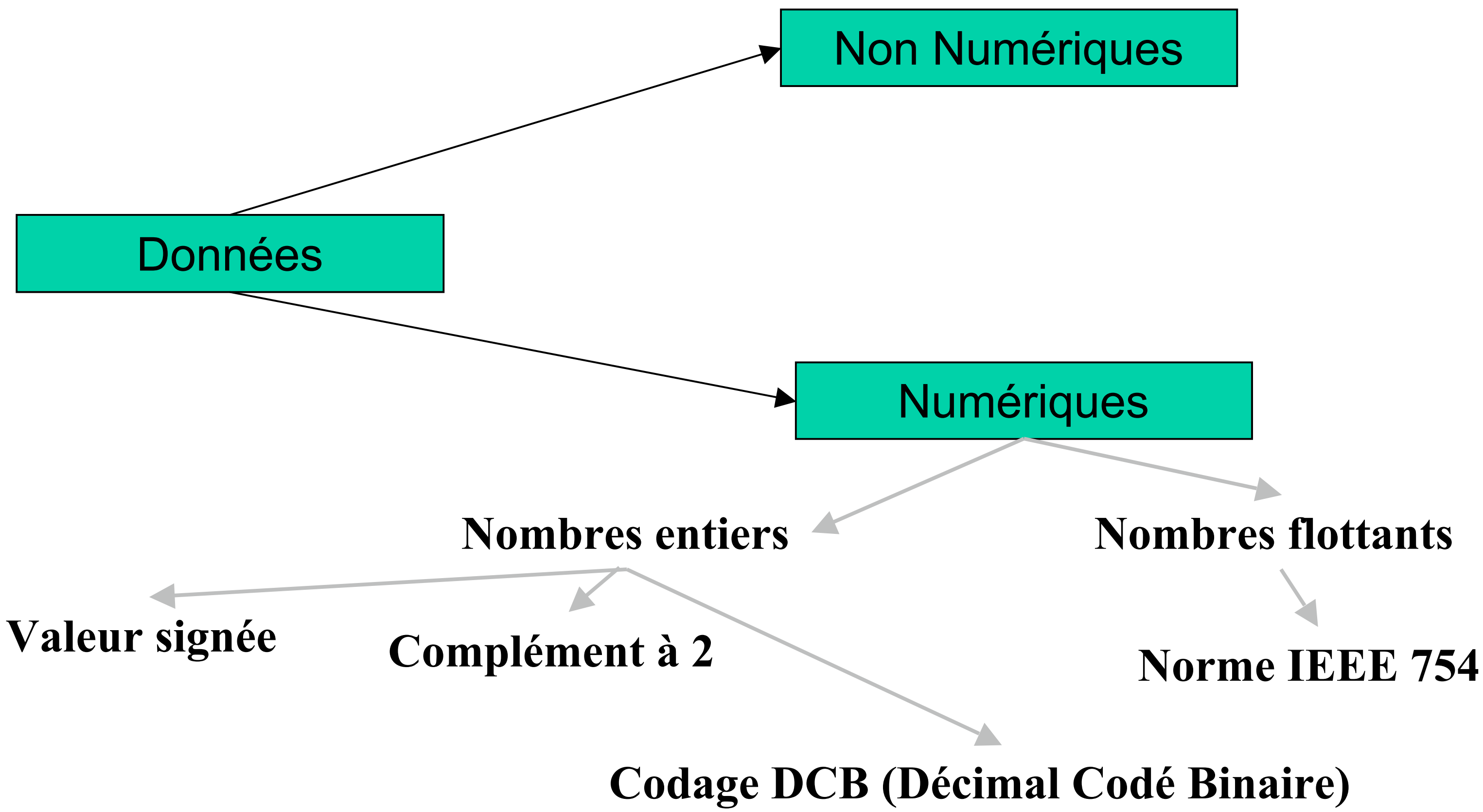
A99ACF

7652993002

1101110110111011

3419

Représentation des données



Représentation des données

- Toutes les données sont stockées sous forme binaire de tailles différentes
- Ces données peuvent être interprétées pour représenter des données de différents types et formats via un langage de programmation
  - float, char, bool, int, etc.

Représentation des nombres

- L’arithmétique utilisée par les ordinateurs
  - Précision finie (et fixe)
    - Limitations
  - Une notation binaire
- Représentation s’effectue selon une chaîne binaire d’une longueur fixée à n bits
  - Sur 8 bits, 16 bits …

Entier

- Pas de partie fractionnaire

Exemples: -2022
-213
0
1
666
54323434565434

Représentation des nombres entiers signés

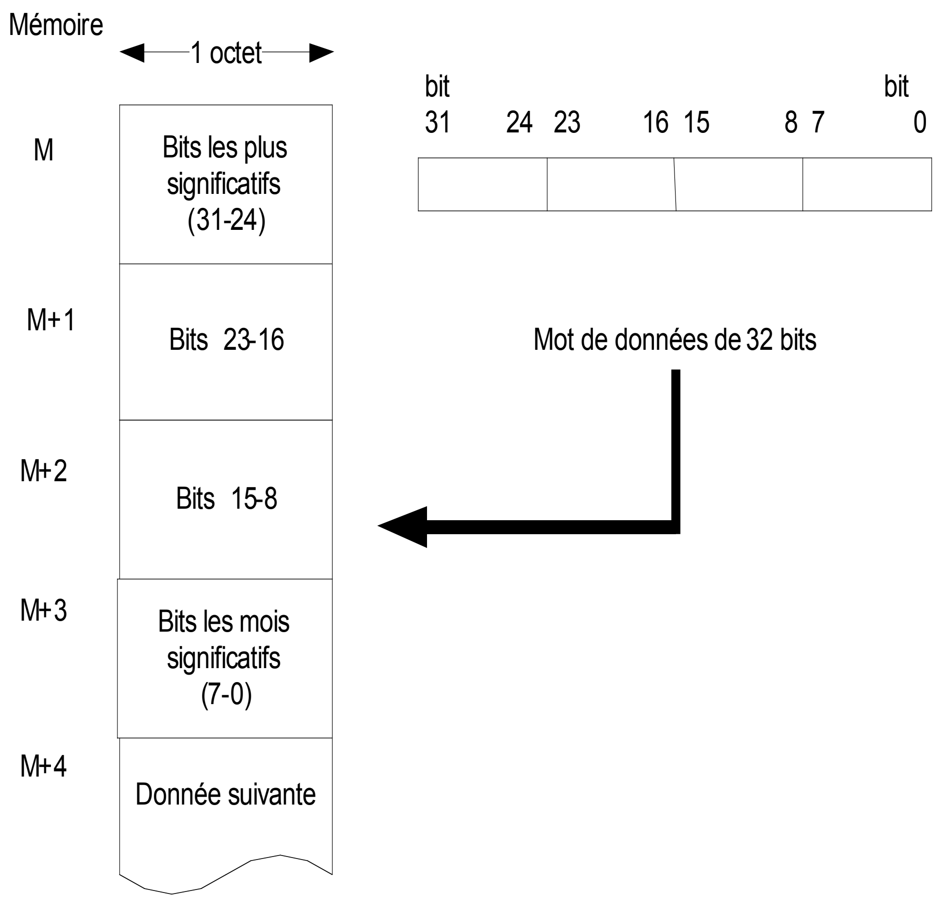
- Conventions
  - Valeur signée
  - Codage DCB (Décimal Codé Binaire)
  - Complément à 1
  - Complément à 2

Représentation des nombres entiers signés

- Le choix entre des conventions
  - Le constructeur de la machine
  - Éventuellement par le programmeur
    - Langage C
      - int – 2 octets, complément à 2
      - unsigned short – 8 bits, non signé

Entiers positifs

- Représentation des entiers positifs
  - Un approche évident
    - Codage en binaire
    - 8 bits => 256 valeurs
    - 32 bits =>
      4294967296 valeurs



En Général (binaire)

Nombre de bits	Binaire	
	Min	Max
n	0	2^n - 1

Important !!

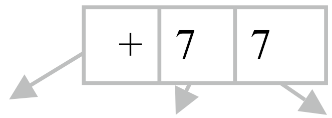
de 0 à $(2^n - 1)$ => 2^n valeurs différentes !

Convention du codage DCB

- Décimal Codé Binaire
  - Chaque chiffre du nombre N_{10} est codé par son équivalent binaire
    - 10 valeurs différentes
    - 4 bits
    - Le codage du signe peut suivre différentes conventions
      - + : 1011_2
      - - : 1101_2

Convention du codage DCB

- Exemple



$+77_{10}$: $1011\ 0111\ 0111_2$

-77_{10} : $1101\ 0111\ 0111_2$

- Préféré pour certaines applications (affaires) où il est nécessaire d'avoir une représentation exacte du nombre décimal
- Conversion DCB→caractère est facile

Intervalles de formats de données

Nb. de bits	Binaire	BCD	ASCII
1	0 – 1		
2	0 – 3		
3	0 – 7		
4	0 – 15	0 – 9	
5	0 – 31		
6	0 – 63		
7	0 – 127		
8	0 – 255	0 – 99	0 – 9
9	0 – 511		
16	0 - 65,535	0 – 9999	0 – 99
24	0 – 16,777,215	0 – 999999	0 – 999
Etc.			

Le nombre de valeurs codées en DCB est moins important qu'en binaire

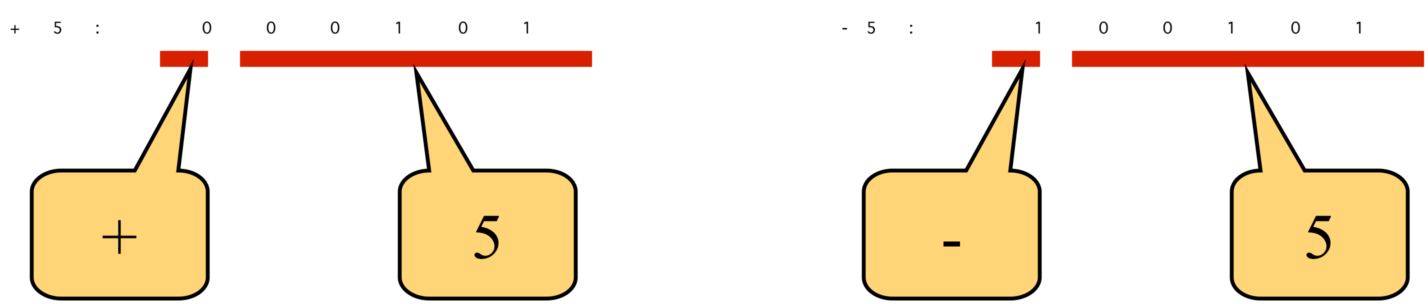
Convention du codage DCB

- Inconvénients
  - Codage ne se prête pas directement aux opérations arithmétiques
    - Résultat – un code binaire sans signification
  - L'arithmétique en DCB est plus difficile qu'en binaire et plus lente

76 →	$0111\ 0110_{bcd}$		convertir les sommes partielles
x 7 →	0111_{bcd}		
42 →	101010_{bin}	→	$0100\ 0010_{bcd}$
49 →	110001_{bin}	→	$+0100\ \ 1001_{bcd}$
4[1]32 →			0100 1101 0010
13 ← ajuster la retenue		convertir 13 en DCB	+0001 0011
532 →			0101 0011 0010
			= 532 en DCB

Convention de la valeur signée

- Réserver un bit pour le signe (le bit le plus à gauche); les autres bits codent la valeur absolue du nombre
  - 0 = « + » et 1 = « - »
- Représentation de +5 et -5 en valeur signée sur 6 bits



Convention de la valeur signée

- Difficultés: Deux représentations de la valeur zéro
  - Représentation en valeur signée sur 6 bits
    - 0 : 0 0 0 0 0 0 => +0
    - 0 : 1 0 0 0 0 0 => -0
  - La réalisation d'une opération de type soustraction nécessite un circuit particulier différent de celui permettant la réalisation des additions
  - Le système doit tester à la fin de chaque calcul pour assurer qu'il n'y a qu'un seul zéro

Intervalles des nombres

Longueur de la chaîne de bits	Intervalle en base 10			
	Non signé		Valeur signée	
	Min	Max	Min	Max
1	0	1		
2	0	3	-1	1
3	0	7	-3	3
4	0	15	-7	7
5	0	31	-15	15
6	0	63	-31	31
Etc.				

La moitié des codes est affectée au nombres positifs et l'autre moitié au nombres négatifs

Convention de la valeur signée

Nombre de bits	Valeur signée	
	Min	Max
n	$-(2^{n-1} - 1)$	$2^{n-1} - 1$

Convention du complément

- Complément: soustraire une valeur de la valeur base
- Complément à 1(restreint ou logique)
  - Complément à 9
- Complément à 2 (vrai)
  - Complément à 10

Complément logique

- En base 10
- Supposons
  - 3 digits décimaux
  - Diviser l'intervalle de représentation

500	999	0	499
-499_{10}	-0_{10}	0_{10}	499_{10}

  - 5xx, 6xx, 7xx, 8xx, 9xx – nombres négatifs
  - Complément → 999-Nombre

Complément logique

- Complément à 9
- Représenter -467_{10} en complément à 9 (3 digits)?

$$\begin{array}{r} 999 \\ -\ 467 \\ \hline 532 \end{array} \qquad -467_{10} \rightarrow 532$$

- Représenter -467_{10} en complément à 9 (4 digits)?

$$\begin{array}{r} 9999 \\ -\ 467 \\ \hline 9532 \end{array} \qquad -467_{10} \rightarrow 9532$$

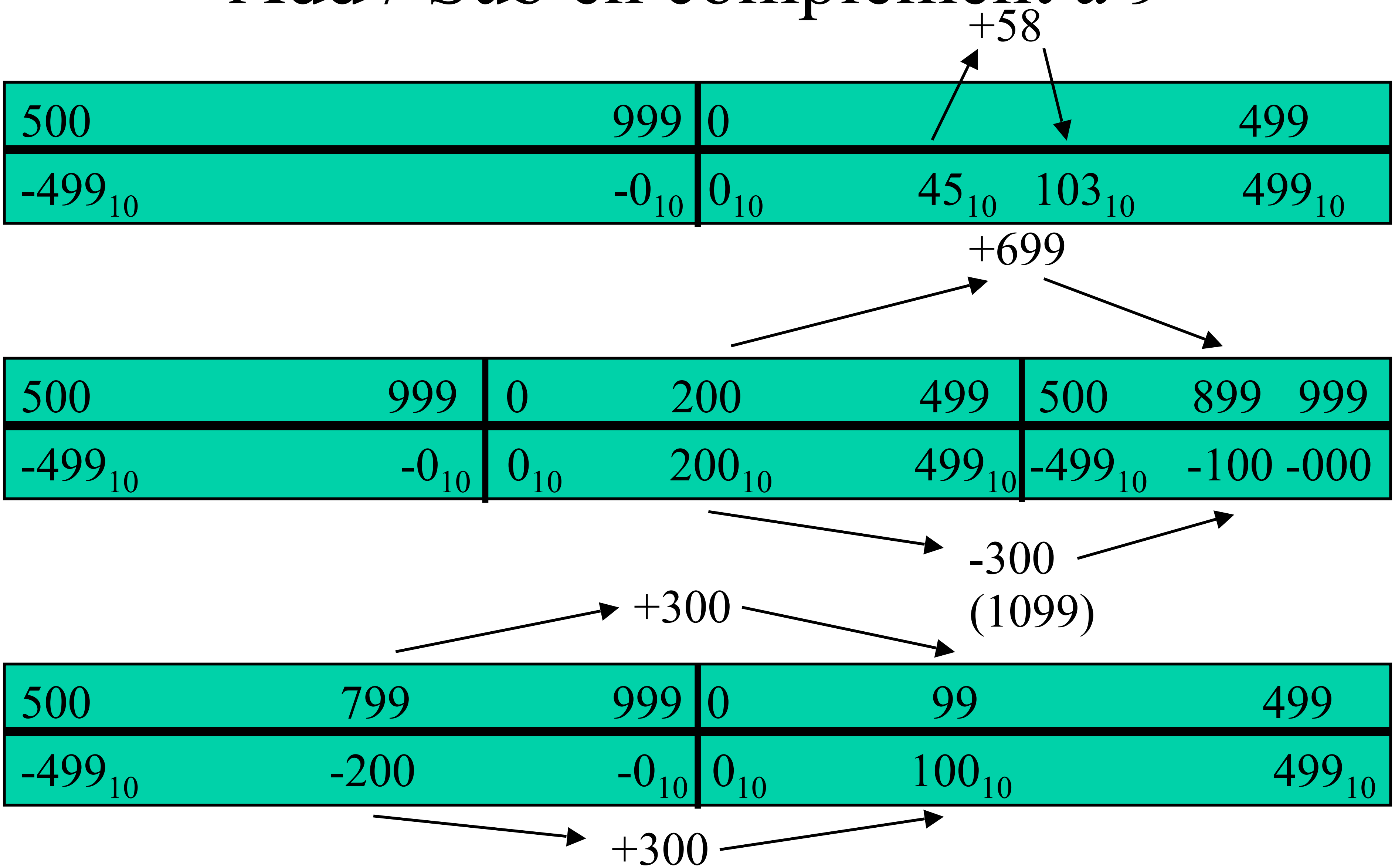
Complément logique

- Complément à 9
- Quelles sont la valeur du signe et la magnitude de 9990 lorsque celui-ci est une représentation en complément à 9 sur 4 digits?
  - Le premier digit est supérieur à 4, donc → **signe négative**

$$\begin{array}{r} \mathbf{9999} \\ \mathbf{-9990} \\ \hline \mathbf{0009} \end{array}$$

Donc, 9990 en complément à 9 sur 4 digits représente: -9

Add / Sub en complément à 9



Add / Sub en complément à 9

- En conséquence, une procédure pour additionner 2 chiffres dans le cas où le résultat s'étend au-delà du nombre maximum de digits consiste à ajouter la dernière retenue

$-200_{10} + 100_{10}$ en complément à 9 sur 3 digits

$-200_{10} + 300_{10}$ en complément à 9 sur 3 digits

$$\begin{array}{r} 799 \\ 100 \\ \hline 899 \end{array} \qquad \begin{array}{r} 799 \\ 300 \\ \hline 1099 \\ \hookrightarrow 1 \\ \hline 100 \end{array}$$

Add / Sub en complément à 9

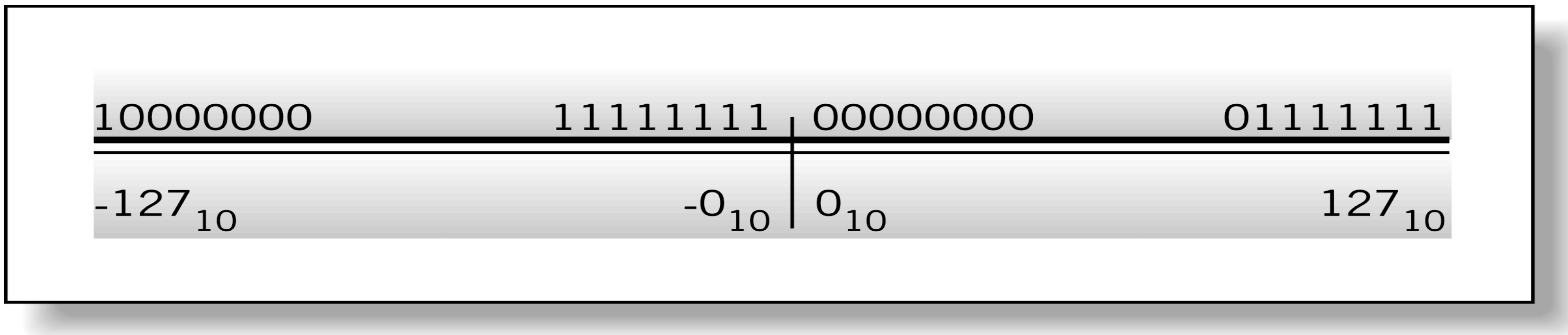
- Pour soustraire, on prend le complément du chiffre que l'on doit soustraire et on réalise l'addition
  - Possibilité de débordement (*overflow)*
    - Exemple: 300 + 300 = 600 (-399)?
  - Si les deux entrées de l'addition ont le même signe et le signe du résultat est différent alors on a un problème de débordement

Convention du complément à 1

- Convention du complément à 1
    - 0 dans le bit le plus à gauche => « + »
    - 1 => « - »
  - Nombre positif
    - Représentation binaire sur n bits
    - 6 : 0 0 0 1 1 0 (6 bits)
  - Nombre négatif
    - Inverser tous les bits 0 → 1 et 1 → 0
    - -6 : 1 1 1 0 0 1 (6 bits)

Convention du complément à 1

- Intervalle des nombres représentables en complément à 1 sur 8 bits



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 4, Figure 04-10

- Cette méthode est aujourd'hui obsolète

Convention du complément à 1

- Inconvénient important
  - Deux représentation distinctes de la valeur 0

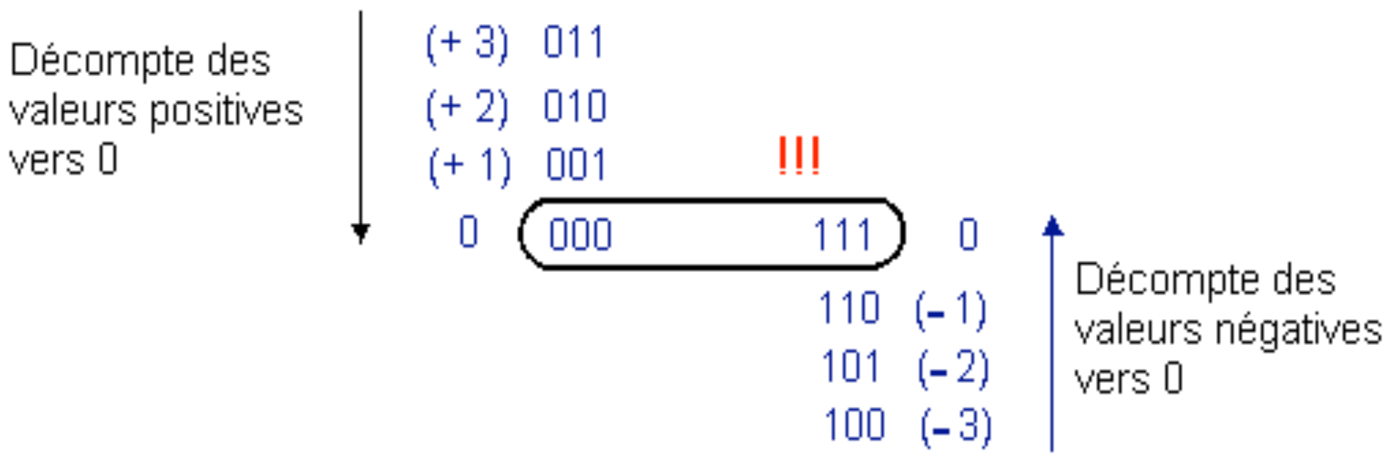


Fig. 28. - Double représentation possible du zéro.

Add / Sub en complément à 1

+58

10000000	11111111	00000000			01111111
-127_{10}	-0_{10}	0_{10}	45_{10}	103_{10}	127_{10}

$$
\begin{array}{l}
00101101 = 45_{10} \\
00111010 = 58_{10} \\
\hline
01100111 = 103_{10}
\end{array}
$$

Add / Sub en complément à 1

+106

10000000		11111111	00000000		01111111
-127_{10}	-2_{10}	-0_{10}	0_{10}	103_{10}	127_{10}

$$01101010 = 106_{10}$$

$$11111101 = -2_{10}$$

$$1\ 01100111 = 103_{10}$$

$$+1$$

$$01101000 = 104_{10}$$

Complément arithmétique (vrai)

- En base 10
  - Supposons
    - 3 digits décimaux
    - Diviser l'intervalle de représentation

500	999	0	499
-500_{10}	-001_{10}	0_{10}	499_{10}

  - 5xx, 6xx, 7xx, 8xx, 9xx – nombres négatifs
  - Trouver un complément sur 3 digits, 2 méthodes:
    - 1) 1000-Nombre
    - 2) Complément à 9 sur 3 digits + 1

Complément vrai

- Complément à 10
- Représenter -467_{10} en complément à 10 (3 digits)?

$$\begin{array}{r} 1000 \\ -\ 467 \\ \hline 533 \end{array} \qquad \begin{array}{r} 532 \\ +\ \ 1 \\ \hline 533 \end{array} \qquad -467_{10} \rightarrow 533$$

- Représenter -467_{10} en complément à 10 (4 digits)?

$$\begin{array}{r} 10000 \\ -\ 467 \\ \hline 9533 \end{array} \qquad \begin{array}{r} 9532 \\ +\ \ 1 \\ \hline 9533 \end{array} \qquad -467_{10} \rightarrow 9533$$

Complément vrai

- Complément à 10
- Quelles sont la valeur du signe et la magnitude de 9990 lorsque celui-ci est une représentation en complément à 10 sur 4 digits?
  - Le premier digit est supérieur à 4, donc → **signe négative**

10000	**0009**
-9990	**+ 1**
0010	**0010**

Donc, 9990 en complément à 10 sur 4 digits représente: -10

Complément vrai

- Complément à 10
- Additions simples!

$-200_{10} + 100_{10}$ en complément à 10 sur 3 digits

$-200_{10} + 300_{10}$ en complément à 10 sur 3 digits

$$\begin{array}{r} 800 \\ +\ 100 \\ \hline 900 \end{array} \qquad\qquad \begin{array}{r} 800 \\ +\ 300 \\ \hline 1100 \end{array}$$

On laisse tomber la retenue

- Toute retenue au-delà du nombre de digit n'est pas prise en compte

Convention du complément à 2

- **Convention la plus utilisée**
  - 0 dans le bit le plus à gauche signifie le nombre positif => « + »
  - 1 => « - »

- Nombre positif
  - Représentation binaire sur ***n*** bits
  - 6 : 000110 (6 bits)
- Nombre négatif –***N***
  1. Soustraire la valeur au modulus
  2. Complément à 1 de son équivalent positif ,+*N*, et ajouter 1
     - Inverser tous les bits 0 → 1 et 1 → 0 dans la représentation binaire de +*N* sur ***n*** bits et ajouter la valeur 1

Convention du complément à 2

- Exemple
  - 6 (6 bits): +6 => 000110

  - -6 (6 bits):

    1) 000110 Nombre positif 6 sur 6 bits

    2) 111001 Complément à 1

    3) +1 Ajouter 1

    111010 Complément à 2

1.

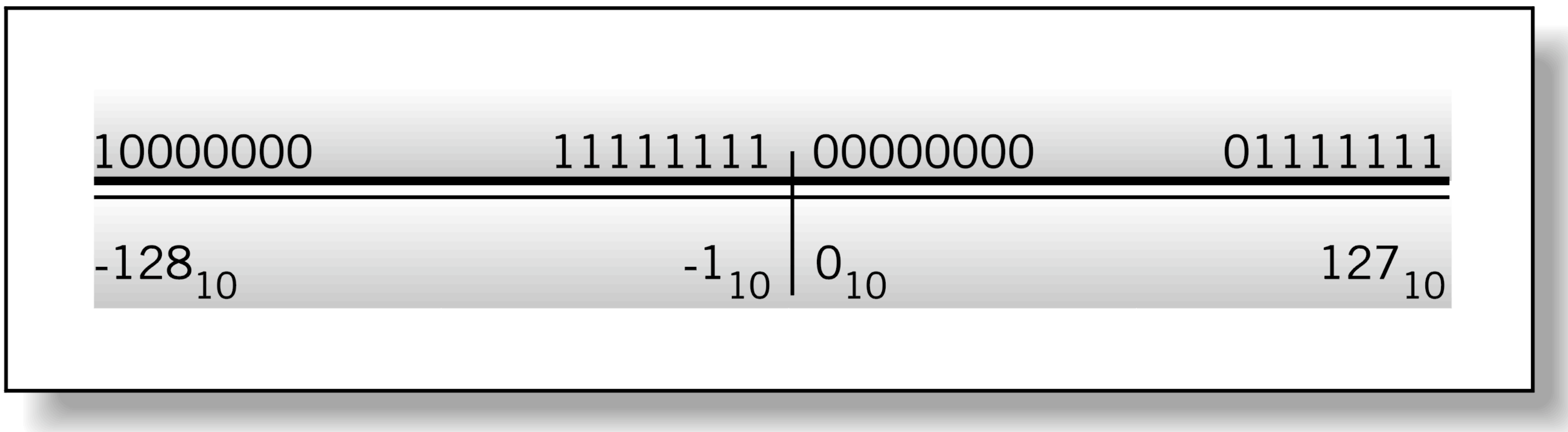
2.

1000000
-000110
111010

-6 c-à-2 sur 6 bits => 111010

Convention du complément à 2

- Intervalle des nombres représentables en complément à 2 sur 8 bits



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition Chapter 4, Figure 04-12

Signe

- Convention du complément à 2, le bit de poids fort (MSB) :
  - 0 = nombre positif
  - 1 = nombre négatif

5

Complément à 2 de 5

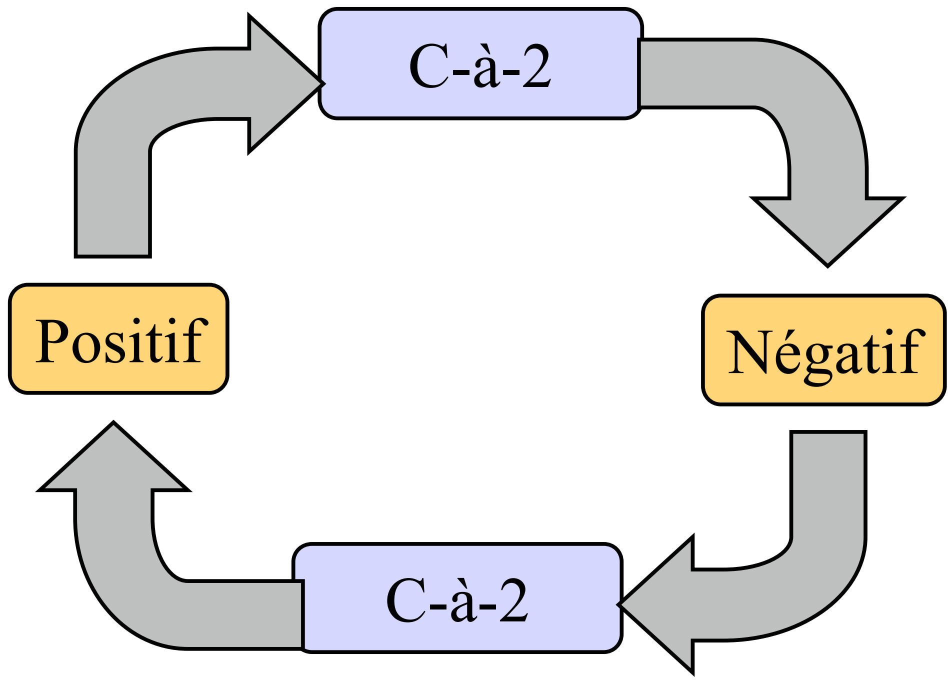
+5 : 0 0 0 1 0 1

-5 : 1 1 1 0 1 1

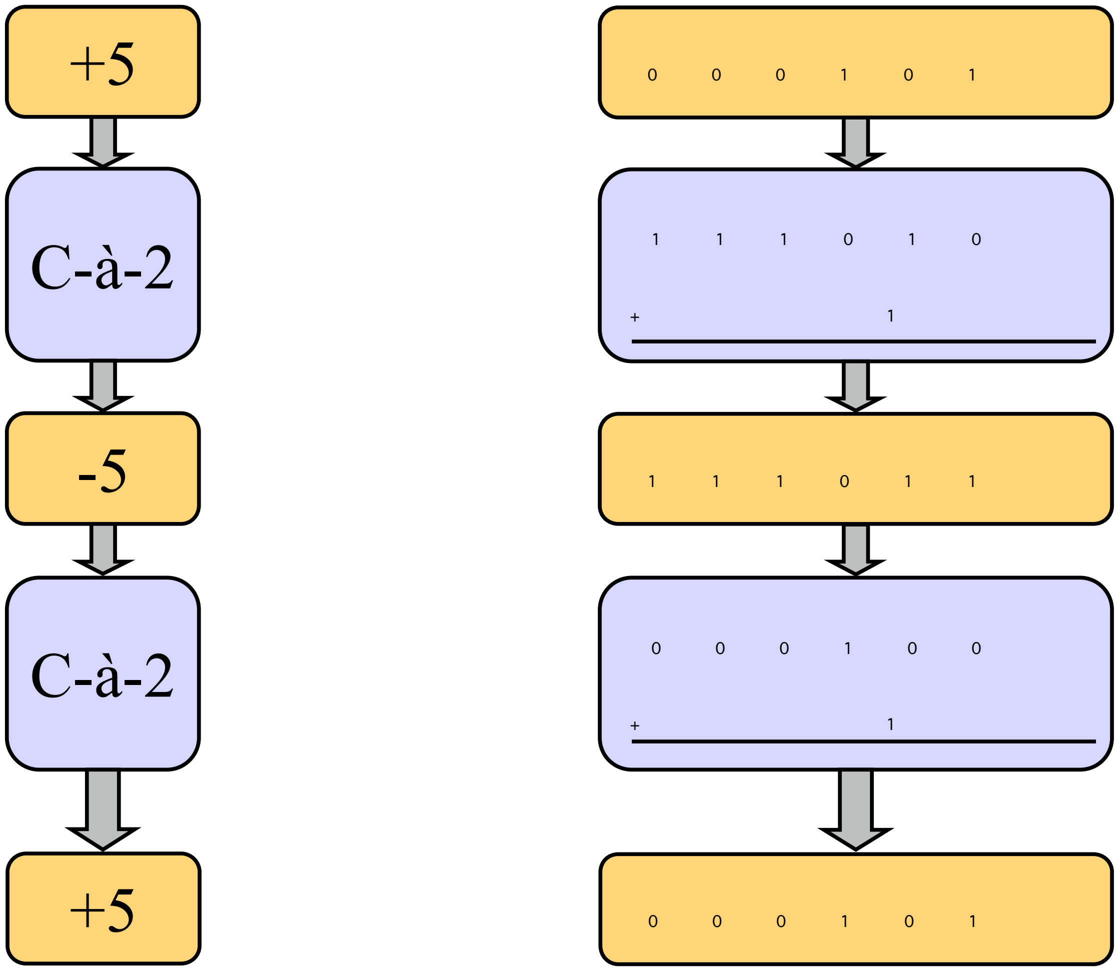
positif

négatif

Notion de “Complément”



Exemple



Exercice – Conversion en C-à-2

- Représenter -20_{10} en c-à-2 sur 8-bits
  Réponse:

- 1100011 est une représentation en c-à-2 sur 7-bits. Donnez la valeur?

  Réponse :

Exercice – Conversion en C-à-2

Réponse

- Représenter -20_{10} en c-à-2 sur 8-bits

  Réponse: <u>11101100</u>

- 1100011 est une représentation en c-à-2 sur 7-bits. Donnez la valeur?

  Réponse : <u>-29</u>

Détails pour -20 -> 11101100

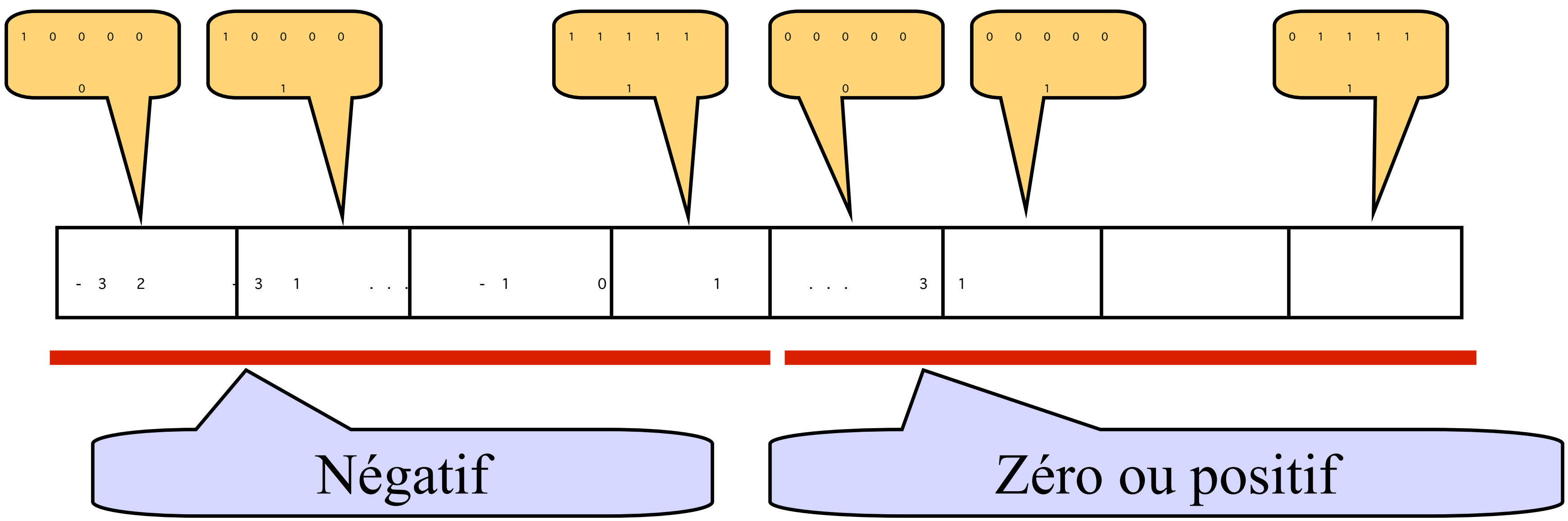
-20_{10}:	Valeur positive =		**00010100**
"Inverser":		Complément à 1	**11101011**
Ajouter 1:			+ 1
			11101100

Détails pour 1100011 -> - 29

C-à-2: Nombre négatif	**1100011**
"Inverser":	**0011100** (Complément à 1)
Ajouter 1:	+ 1
Valeur absolue	**0011101 = 29**
Nombre: =	**- 29**

Intervalle des nombres représentables en complément à 2

- 6 bits



Intervalles des nombres

Nb. de bits	Binaire					
	Non signés		Valeur signée		C-à-2	
	Min	Max	Min	Max	Min	Max
1	0	1				
2	0	3	-1	1	-2	1
3	0	7	-3	3	-4	3
4	0	15	-7	7	-8	7
5	0	31	-15	15	-16	15
6	0	63	-31	31	-32	31
Etc.						

En Général (intervalles)

Nb. de bits	Binaire					
	Non signés		Valeur signée		Complément à 2	
	Min	Max	Min	Max	Min	Max
n	0	$2^n - 1$	$-(2^{n-1} - 1)$	$2^{n-1}-1$	-2^{n-1}	$2^{n-1} - 1$

Addition en complément à 2

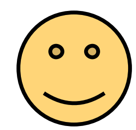
- Facile
- Pas des règles spéciales
- Simplement additionner

-5 plus +5?

- Zéro, bien sûr, mais on va voir?

Valeur signée		
-5 :		1 0 0 0 0 1 0 1
+5 :	+	0 0 0 0 0 1 0 1
-10 :		1 0 0 0 1 0 1 0

C-à-2		
		1 1 1 1 1 1 1 1
-5 :		1 1 1 1 1 0 1 1
+5 :	+	0 0 0 0 0 1 0 1
		0 0 0 0 0 0 0 0



Soustraction en complément à 2

- Facile
- Pas de règles spéciales
- Simplement additionner

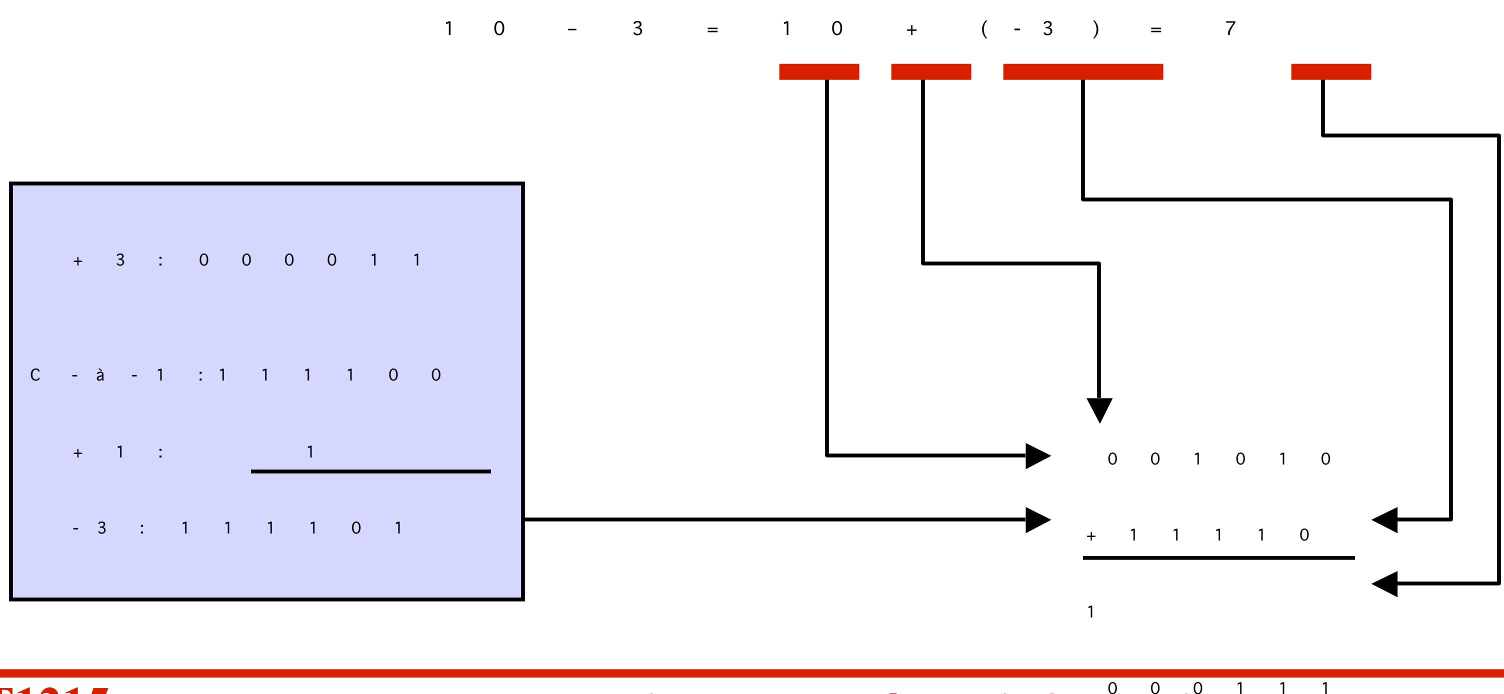
$$A - B = A + (-B)$$

additionner

Complément à 2 de B

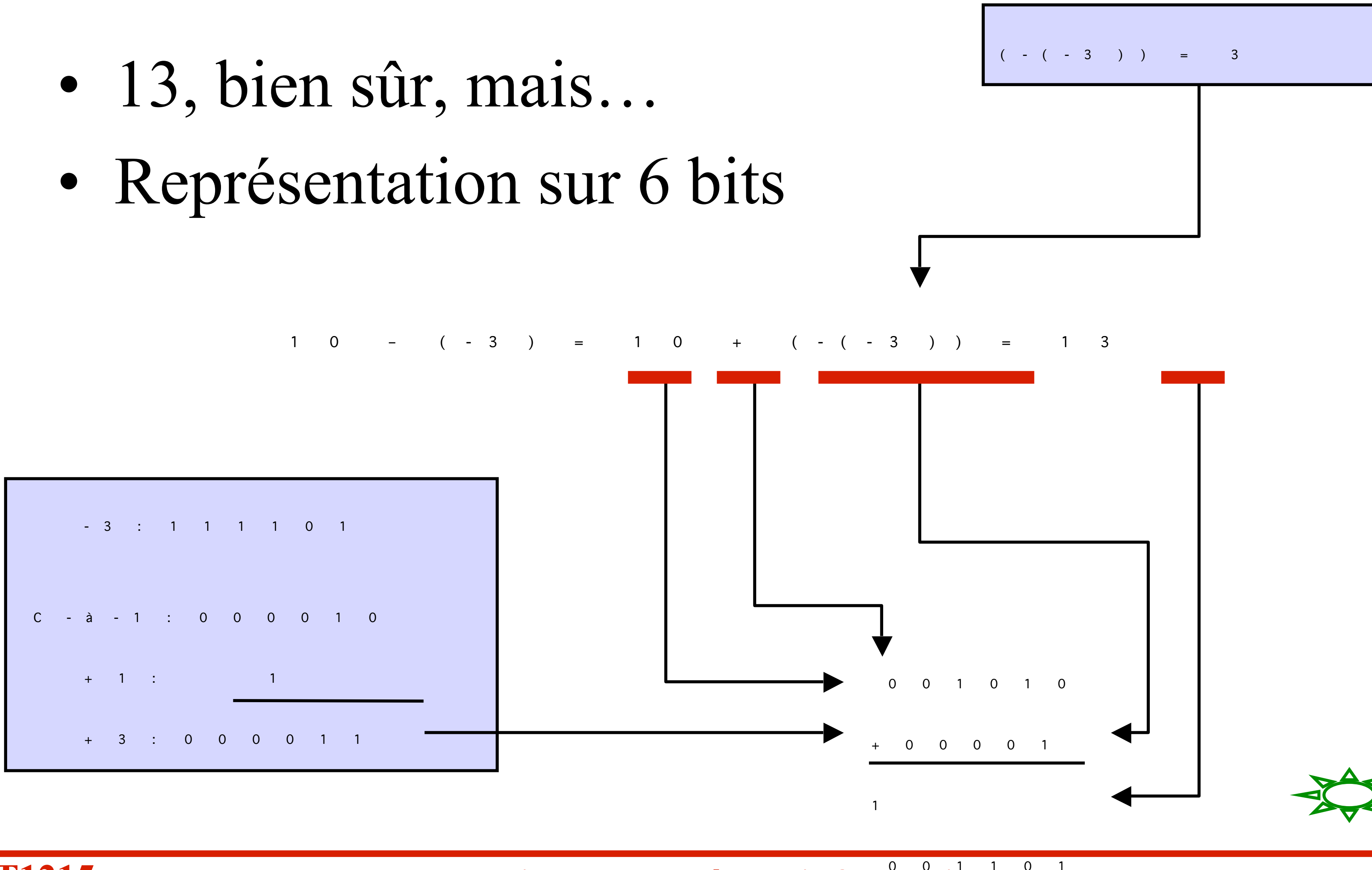
10 - 3?

- 7, bien sûr,
- On utilise une représentation sur 6-bits



10 – (-3)?

- 13, bien sûr, mais…
- Représentation sur 6 bits



Notion de carry et d'overflow

- Notion de carry = retenue
  - Lors d'une opération arithmétique effectuée sur des nombres de p bits, un p+1er bit peut être généré (bit de carry)

Convention du c-à-2 sur 8 bits

$$
\begin{array}{r}
0111\ 1111_2 \\
1111\ 1110_2 \\
\hline
1\ 0111\ 1101_2
\end{array}
$$

Notion de carry et d'overflow

- Notion d'overflow ou de dépassement de capacité
  - Lors d'une opération arithmétique mettant en jeu des nombres de p bits et de même signe, le résultat peut se révéler être trop grand ou trop petit pour être représentable par la machine
    - Résultat est en dehors de l'intervalle des nombres représentables sur p bits par la convention choisie
    - Résultat => erroné
    - Dépassement de capacité

Notion de carry et d'overflow

- Notion d'overflow ou de dépassement de capacité
  - Exemple

Convention du c-à-2 sur 8 bits

$+127_{10}$	$0111\ 1111_2$
$+2_{10}$	$0000\ 0010_2$
$+129 \neq -127_{10}$	$1000\ 0001_2$

Dépassement de capacité!!!

Convention du c-à-2 sur 8 bits => [-127_{10}, $+127_{10}$]

“Overflows” et “Carries”

Convention c-à-2 sur 4 bits

(+4) + (+2)

0100	no overflow,
<u>0010</u>	no carry
0110 = (+6)	

(+4) + (+6)

0100	overflow,
<u>0110</u>	no carry
1010 = (−6)	the result is incorrect

(- 4) + (- 2)

1100	no overflow,
<u>1110</u>	carry
11010 = (−6)	ignoring the carry,
	the result is correct

(- 4) + (- 6)

1100	overflow
<u>1010</u>	carry
10110 = (+3)	ignoring the carry,
	the result is incorrect