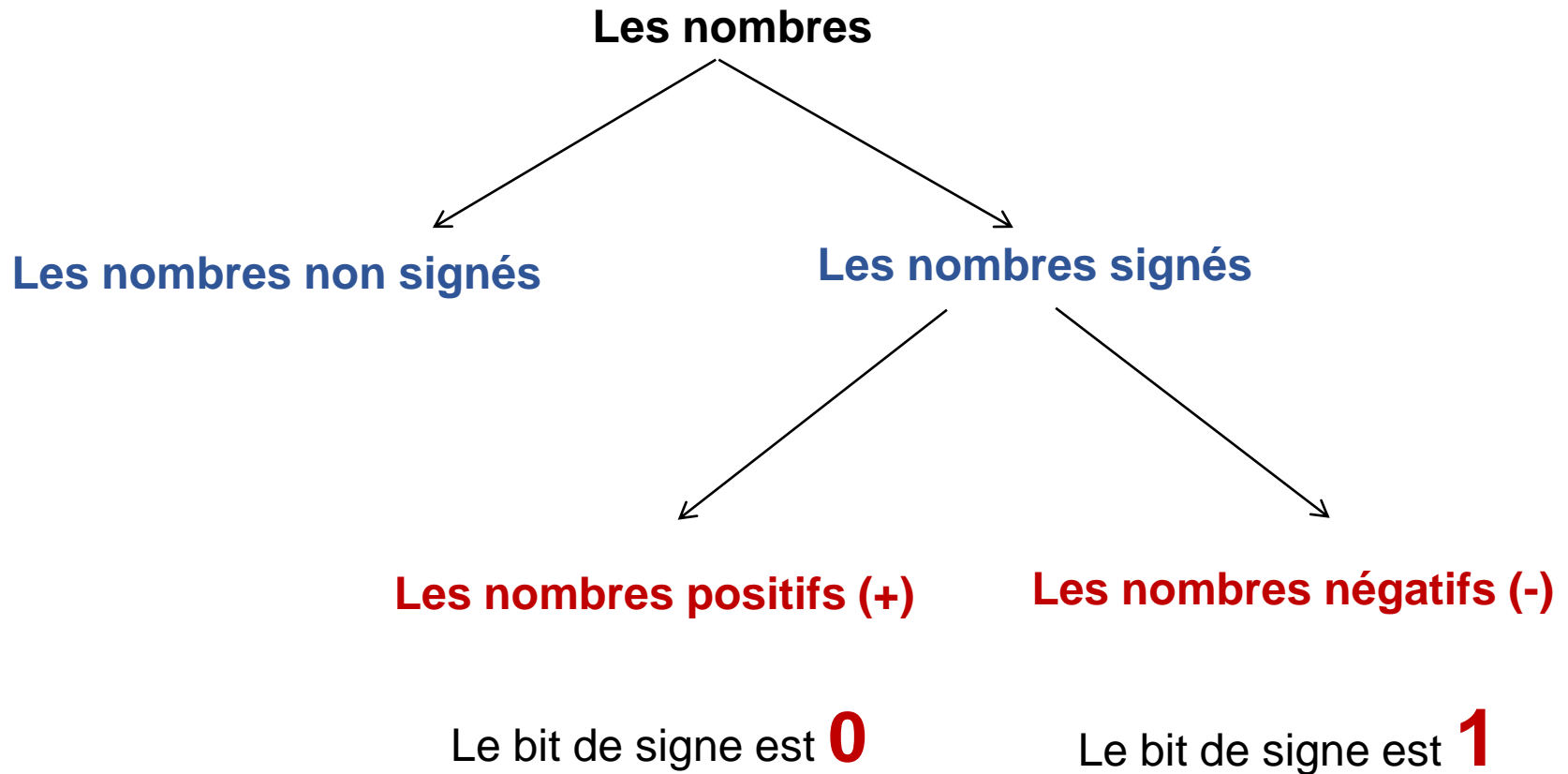


# **Suite Chapitre 1\_Partie 3: Système de numération et représentation des nombres**

## Représentation des nombres entiers:



## Représentation des nombres entiers:

Il existe trois manière pour représenter les nombres signés:

- ❑ **Signe/ valeur absolue**
- ❑ **Complément à 1**
- ❑ **Complément à 2**

## Représentation des nombres signés:

### ❑ La représentation signe / valeur absolue (S/VA)

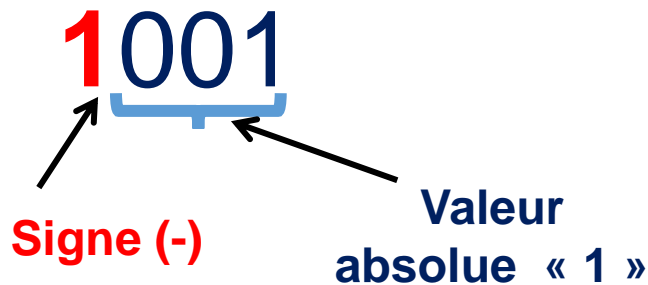
C'est une représentation assez simple dont le principe est le suivant :

### ❑ Le bit du poids le plus fort correspond au bit de signe :

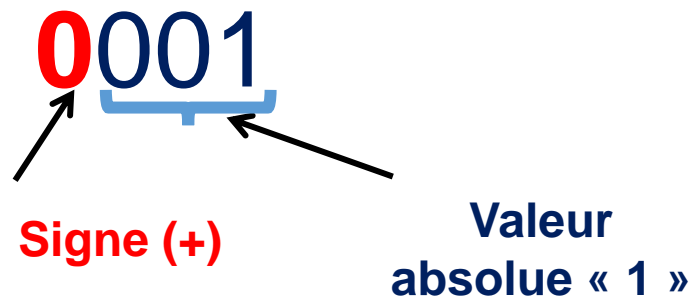
- Si le nombre est positif donc le bit de signe est mis à **0**
- Si le nombre est négatif donc le bit de signe est mis à **1**

### ❑ Les autres bits codent la valeur absolue du nombre

### Exemples:



1001 est la représentation S/VA de -1



1001 est la représentation S/VA de +1

## Représentation des nombres signés:

- ❑ La représentation en complément à 2 'CA2'

# Représentation des nombres entiers:

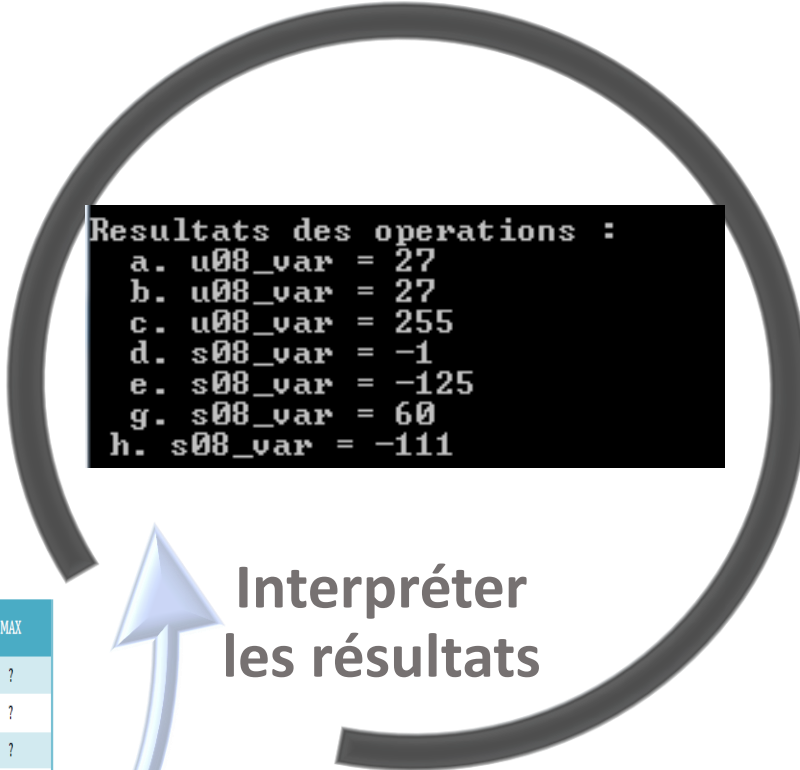
□ La représentation en complément à 2

« Mise en situation »



Compléter le  
tableau

Type	Taille n (bits)	Nombre de valeurs représentées (2 <sup>n</sup> )	MIN	MAX
char	?	?	?	?
unsigned char	?	?	?	?
short	?	?	?	?
unsigned short	?	?	?	?
int	?	?	?	?
unsigned int	?	?	?	?

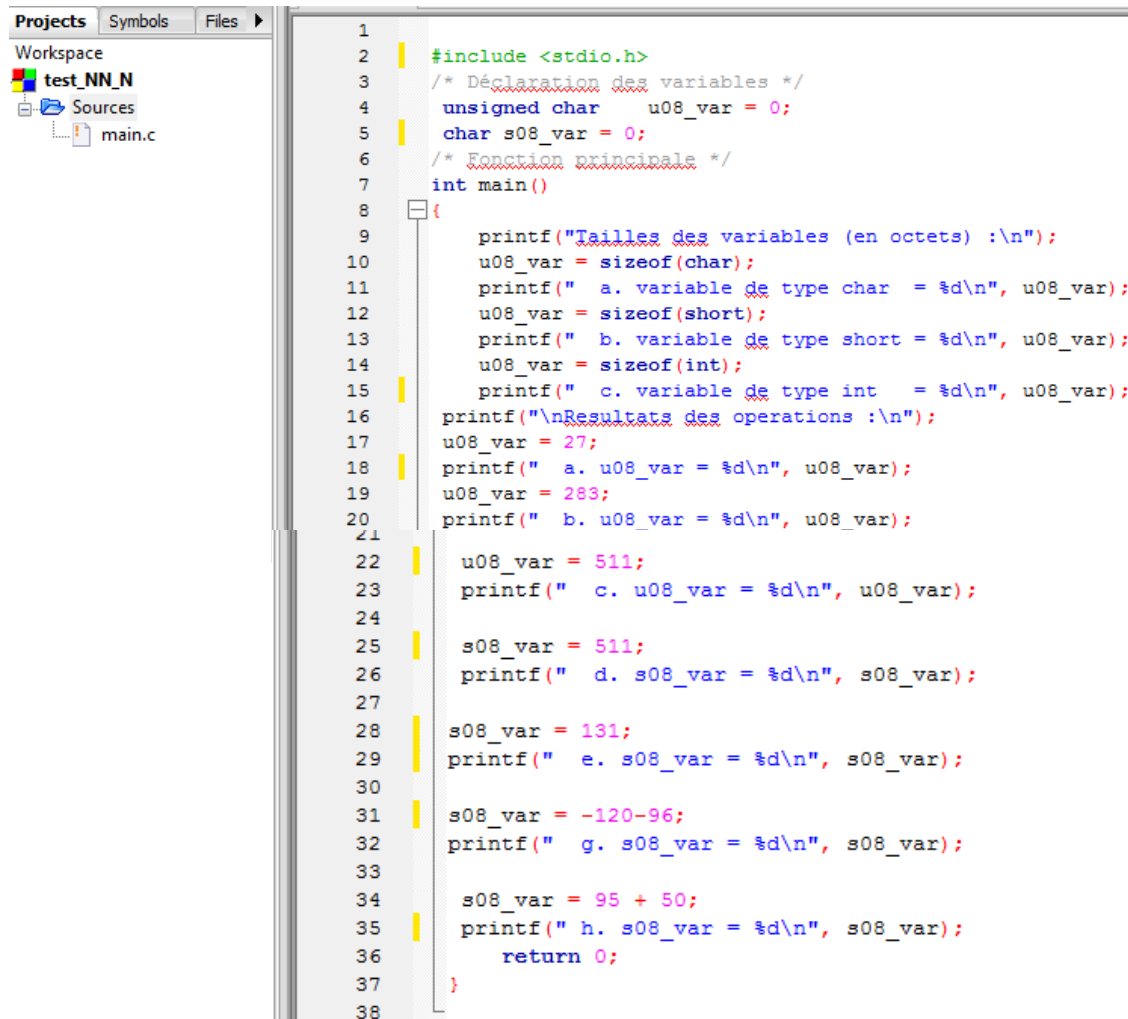


# Représentation des nombres entiers:

## ❑ La représentation en complément à 2

« Mise en situation »

Lisez et exécutez le code ci-dessus.



```
1
2  #include <stdio.h>
3  /* Déclaration des variables */
4  unsigned char  u08_var = 0;
5  char s08_var = 0;
6  /* Fonction principale */
7  int main()
8  {
9      printf("Tailles des variables (en octets) :\n");
10     u08_var = sizeof(char);
11     printf(" a. variable de type char = %d\n", u08_var);
12     u08_var = sizeof(short);
13     printf(" b. variable de type short = %d\n", u08_var);
14     u08_var = sizeof(int);
15     printf(" c. variable de type int = %d\n", u08_var);
16     printf("\nRésultats des opérations :\n");
17     u08_var = 27;
18     printf(" a. u08_var = %d\n", u08_var);
19     u08_var = 283;
20     printf(" b. u08_var = %d\n", u08_var);
21
22     u08_var = 511;
23     printf(" c. u08_var = %d\n", u08_var);
24
25     s08_var = 511;
26     printf(" d. s08_var = %d\n", s08_var);
27
28     s08_var = 131;
29     printf(" e. s08_var = %d\n", s08_var);
30
31     s08_var = -120-96;
32     printf(" g. s08_var = %d\n", s08_var);
33
34     s08_var = 95 + 50;
35     printf(" h. s08_var = %d\n", s08_var);
36     return 0;
37 }
38
```

# Représentation des nombres entiers:

□ La représentation en complément à 2

## Le résultat obtenu

Résultat 1

```
Taille des variables (en octets) :  
a. variable de type char   = 1  
b. variable de type short  = 2  
c. variable de type int    = 4
```

Résultat 2

```
Resultats des operations :  
a. u08_var = 27  
b. u08_var = 27  
c. u08_var = 255  
d. s08_var = -1  
e. s08_var = -125  
g. s08_var = 60  
h. s08_var = -111
```



# Représentation des nombres entiers:

## ❑ La représentation en complément à 2

Le tableau suivant liste les types numériques utilisés dans le langage de programmation C  
Compléter le tableau avec les informations demandées en se basant sur le résultat 1 et vos  
prérequis en informatique.

Résultat 1

```
Taille des variables (en octets) :  
a. variable de type char = 1  
b. variable de type short = 2  
c. variable de type int = 4
```

Tableau : Caractéristiques des principaux types numériques en C

Type	Taille (n bits )	Nombre de valeurs représentés	MIN	MAX
Char	8 bits	256	-128	127
Unsigned char	8 bits	256	0	256
short	16 bits	$2^{16}$	$-\frac{2^{16}}{2}$	$\frac{2^{16}}{2}-1$
Unsigned short	16 bits	$2^{16}$	0	$2^{16}-1$

# Représentation des nombres entiers:

□ La représentation en complément à 2

## Le résultat obtenu

Résultat 1

```
Taille des variables (en octets) :  
a. variable de type char   = 1  
b. variable de type short  = 2  
c. variable de type int    = 4
```

Résultat 2

```
Resultats des operations :  
a. u08_var = 27  
b. u08_var = 27  
c. u08_var = 255  
d. s08_var = -1  
e. s08_var = -125  
g. s08_var = 60  
h. s08_var = -111
```

# Représentation des nombres entiers:

## □ La représentation en complément à 2

Expliquer en détail le résultat 2 obtenu

Résultat 2

```
Resultats des operations :
a.  u08_var = 27  -> 27
b.  u08_var = 27  -> 27
c.  u08_var = 255 -> 511
d.  s08_var = -1  -> 511
e.  s08_var = -125 -> 131
g.  s08_var = 60  -> -132-64
h.  s08_var = -111 -> 95+50
```



```
C:\Users\nadia\Desktop\test_NN_N\main.c:19:12: warning: large integer implicitly truncated to unsigned type [-Woverflow]
    u08_var = 283;
               ^
C:\Users\nadia\Desktop\test_NN_N\main.c:22:12: warning: large integer implicitly truncated to unsigned type [-Woverflow]
    u08_var = 511;
               ^
C:\Users\nadia\Desktop\test_NN_N\main.c:25:12: warning: overflow in implicit constant conversion [-Woverflow]
    s08 var = 511;
```

# Représentation des nombres entiers:

## □ La représentation en complément à 2

Expliquer en détail le résultat 2 obtenu

Resultats des operations :

a.	u08_var	=	27	→	27
b.	u08_var	=	27	→	27
c.	u08_var	=	255	→	511
d.	s08_var	=	-1	→	511
e.	s08_var	=	-125	→	131
g.	s08_var	=	60	→	-132-64
h.	s08_var	=	-111	→	95+50

Résultat 2



# Représentation des nombres entiers:

## □ La représentation en complément à 2

```
u08_var = 27;  
printf(" a. u08_var = %d\n", u08_var);
```

### ▪ C'est quoi le type de u08\_var?

```
unsigned char    u08_var = 0;
```

### ▪ Que signifie déclaration?

**Déclarer une variable** est réserver une mémoire de taille bien précise (n bits) pour cette variable.

*Quelle est alors la taille de la mémoire réservée pour u08\_var?*

Pour u08\_var est de type caractère donc la machine a réservé uniquement 8 bits pour stocker cette variable.

### ▪ Quel est le résultat lors de l'affichage?

```
a. u08_var = 27
```



# Représentation des nombres entiers:

## □ La représentation en complément à 2

Expliquer en détail le résultat 2 obtenu

Resultats des operations :

a.	u08_var	=	27	→	27
b.	u08_var	=	27	→	283
c.	u08_var	=	255	→	511
d.	s08_var	=	-1	→	511
e.	s08_var	=	-125	→	131
g.	s08_var	=	60	→	-132-64
h.	s08_var	=	-111	→	95+50

Résultat 2



# Représentation des nombres entiers:

## □ La représentation en complément à 2-Débordement

```
u08_var = 511;  
printf("  c. u08_var = %d\n", u08_var);
```



Affichage

```
c. u08_var = 255
```

Lors de la compilation de la ligne 23 le message suivant s'affiche :

```
C:\Users\nadia\Desktop\test_NN_N\main.c:22:12: warning: large integer implicitly truncated to unsigned type [-Woverflow]  
  u08 var = 511;
```



*Que signifie Over flow ?*

# Représentation des nombres entiers:

## ❑ La représentation en complément à 2\_Débordement

Un caractère

Taille =8bits

Pour écrire 511 il nous faut 9 bits

511 = 1 1 1 1 1 1 1 1 1

9 bits



*Dépassement de la mémoire*



*La mémoire n'est pas extensible*



1 1 1 1 1 1 1 1

Débordement ou  
bien over flow

D'où l'affichage de 255

**Débordement:** C'est le dépassement de la capacité de la mémoire: le nombre de bits utilisés est insuffisant pour contenir la valeur puisque la taille de la mémoire est extensible.



# Représentation des nombres entiers:

## □ La représentation en complément à 2

Expliquer en détail le résultat 2 obtenu

Resultats des operations :

a.	u08_var	=	27	→	27
b.	u08_var	=	27	→	283
c.	u08_var	=	255	→	511
d.	s08_var	=	-1	→	511
e.	s08_var	=	-125	→	131
g.	s08_var	=	60	→	-132-64
h.	s08_var	=	-111	→	95+50

Résultat 2



# Représentation des nombres entiers:

## □ La représentation en complément à 2

```
s08_var = 511;  
printf(" d. s08_var = %d\n", s08_var);
```



Affichage

```
d. s08_var = -1
```

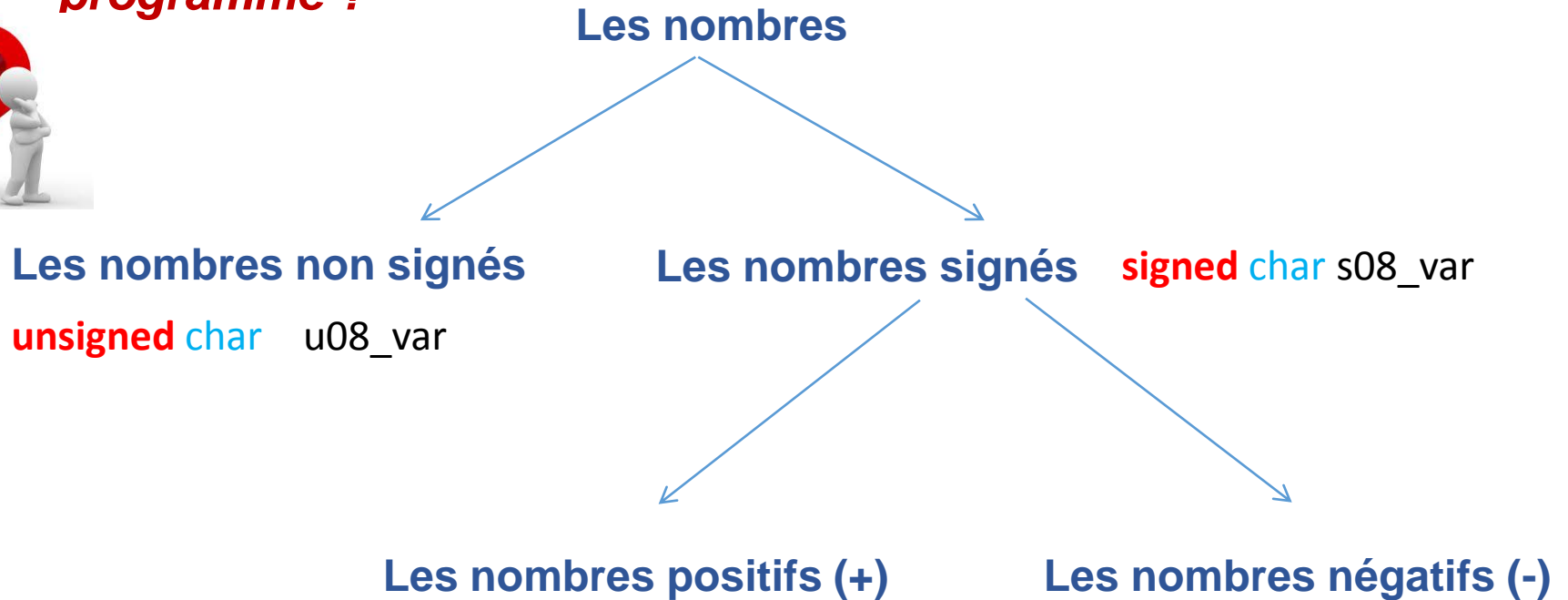
▪ *D'après le programme quel est le type de s08\_var?*

```
signed char s08_var = 0;
```

# Représentation des nombres entiers:

## □ La représentation en complément à 2

- *Quels sont les deux catégories des nombres mentionnés par le programme ?*



# Représentation des nombres entiers:

## □ La représentation en complément à 2

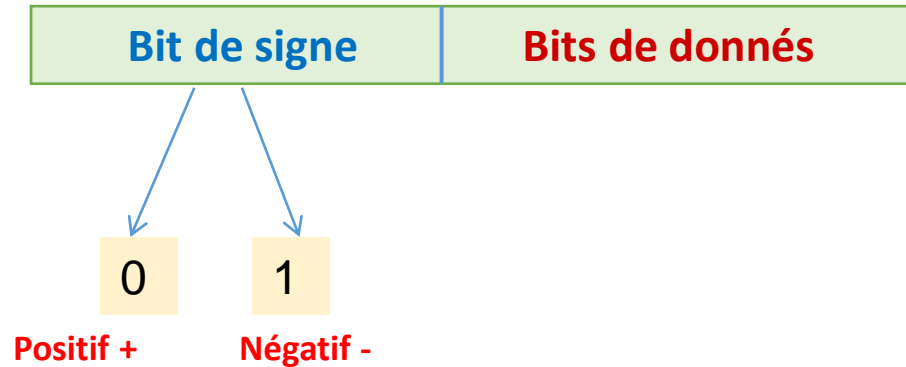


*Quelle est la différence entre les nombres signés et les nombres non signés?*

Les nombres non signés

Bits de donnés

Les nombres signés



Le bit le plus significatif est utilisé pour représenter le signe de nombre:

- Si le bit le plus fort =1 alors le nombre est **négatif**
- Si le bit le plus fort =0 alors le nombre est **positif**

# Représentation des nombres entiers:

## □ La représentation en complément à 2

511 est un nombre signé de type caractère

```
s08_var = 511;  
printf(" d. s08_var = %d\n", s08_var);
```

511 = ~~1~~ 1 1 1 1 1 1 1

**Débordement**

**Bit de  
signe**

- *Est-ce que le nombre est positif ou bien négatif ?*

Le bit de signe est 1 donc le nombre est négatif



*Comment on va coder ce nombre ?*

*Ce n'est plus le codage binaire pure comme les nombres non signés!!*

*C'est une représentation d'un nombre signé par la machine !!*



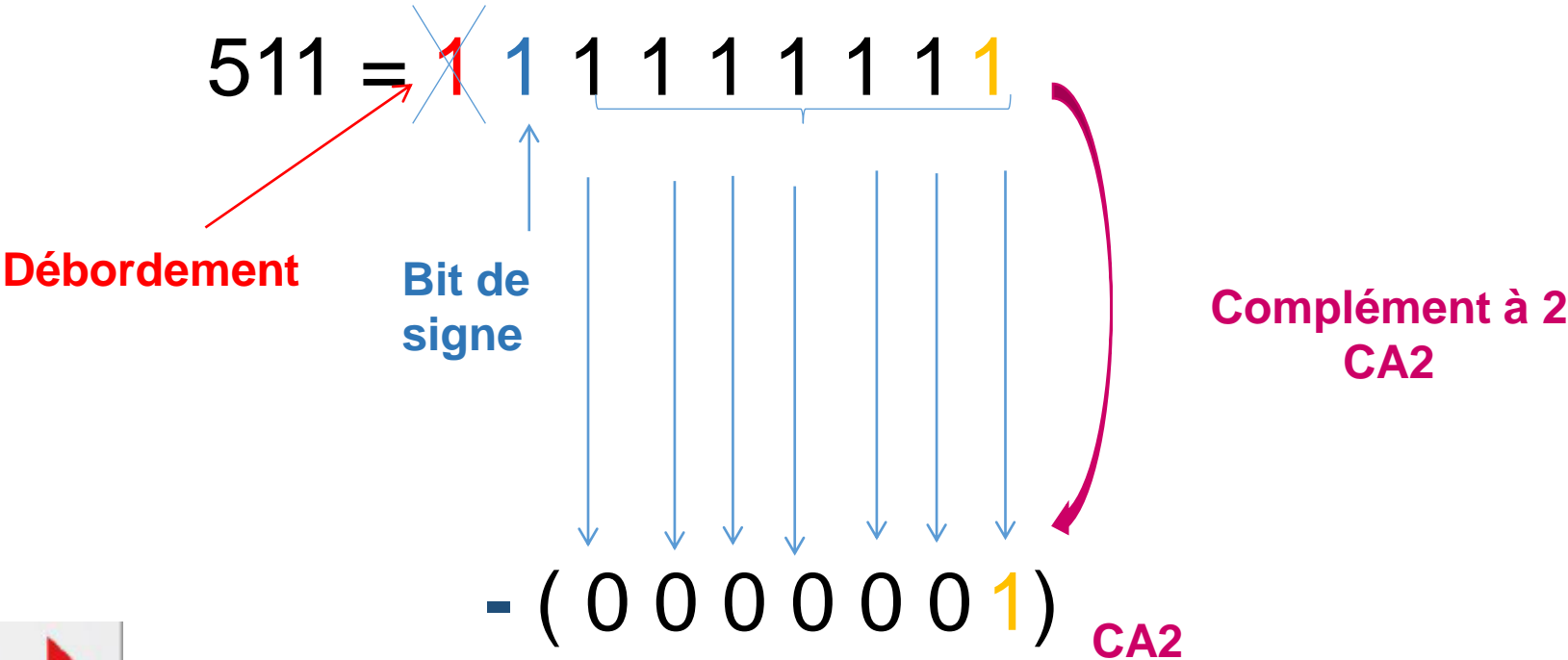
**Comment la machine traite les nombres signés?**

# Représentation des nombres entiers:

## La représentation en complément à 2

### Outil:

Essayer de parcourir les bits de donnée de ce nombre à partir de bit du poids le plus faible et garder les bits avant le premier 1 et inverser les autres bits qui viennent après.



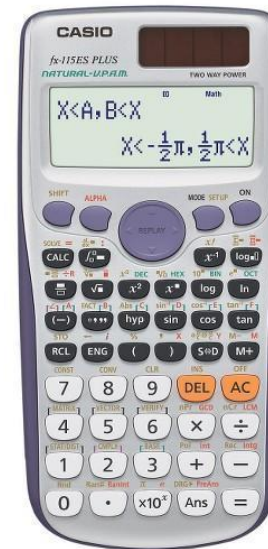
# Représentation des nombres entiers:

## □ La représentation en complément à 2



### ▪ *C'est quoi le complément à 2?*

La représentation en complément à 2 est la représentation la plus utilisée pour représenter les nombres **négatifs** dans les plus part des machines:  **$-b = \text{CA2}(b)$**   
Tel que votre pc ,votre calculatrice.



Essayer de Convertir -5 en binaire

# Représentation des nombres entiers:

## □ La représentation en complément à 2



### ▪ *Comment trouver le complément à 2 d'un nombre?*

Pour trouver le complément à 2 d'un nombre il faut parcourir les bits de ce nombre à partir de bit du poids le plus faible et garder les bits avant le premier 1 et inverser les autres bits qui viennent après.

#### Exemple:

+131

0 1 0 0 0 0 0 1 1

Bit de  
signe

CA2

-131

1 0 1 1 1 1 0 1

Bit de  
signe



+120

0 0 1 1 1 1 0 0 0

CA2

-120

1 1 0 0 0 1 0 0 0





# Représentation des nombres entiers:

## □ La représentation en complément à 2

Essayez maintenant d'appliquer le CA2 sur -1

$$511 = 1 \text{ } 1 \text{ } 1 \text{ } 1 \text{ } 1 \text{ } 1 \text{ } 1 \text{ } 1$$

$$-1 = - (0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 0 \text{ } 1)$$

$$511 = 1 \text{ } 1 \text{ } 1 \text{ } 1 \text{ } 1 \text{ } 1 \text{ } 1 \text{ } 1$$

Complément à 2  
CA2

Complément à 2  
CA2

- Le complément à 2 du complément à 2 d'un nombre N est N  
 $\text{CA2}(\text{CA2}(N))=N$

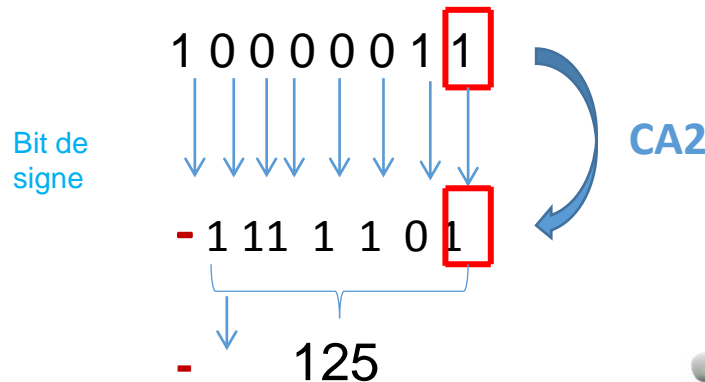
# Représentation des nombres entiers:

## □ La représentation en complément à 2

### Exemple :

L'opération e

131



```
e. s08_var = -125
```

# Représentation des nombres entiers:

## □ La représentation en complément à 2



- *D 'après vous comment la machine effectue ces opérations ?*

$$a-b=?$$

$$-a-b=?$$

$$a-b=a+CA2(b)$$
$$-a-b=CA2(a)+CA2(b)$$

# Représentation des nombres entiers:

## ❑ Opération arithmétiques en utilisant le complément à 2

### Méthode:



- 1) Convertir en binaire la valeur absolue **VA** du nombre a (a et b s'il s'agit d'une opération et essayer de les arrondir de la même taille en terme de nombre de bit)
- 2) Ajouter le bit de signe (si le nombre est signé) pour a (pour a et b s'agit d'une opération)
- 3) Appliquer le complément à 2 s'il s'agit d'un nombre négatif ( **$-b = CA2(b)$** )
- 3) Effectuer l'opération s'il s'agit d'une opération ( **$a - b = a + CA2(b)$ ,  $-a - b = CA2(a) + CA2(b)$** )
- 5) Réserver que **n bits (sa dépend de la taille de la variable)** et analyser :
  - ❑ Pour les **nombres non signés** la machine vérifie **seulement le débordement** et elle affiche alors que pour **les nombres signés** la machine vérifie **le débordement et le signe**.
  - ❑ Le complément à 2 sera applicable seulement pour les nombres signés négatifs (qui possède un bit de signe qui est égale à **1**:  **$-CA2(\text{bits des donnés})$** ).

# Représentation des nombres entiers:

□ Opération arithmétiques en utilisant le complément à 2

Exemple :

Expliquer alors le résultat de l'affichage de l'opération **g**

```
Resultats des operations :  
a. u08_var = 27  -> 27  
b. u08_var = 27  -> 283  
c. u08_var = 255 -> 511  
d. s08_var = -1  -> 511  
e. s08_var = -125 -> 131  
g. s08_var = 60  -> -132-64  
h. s08_var = -111 -> 95+50
```

Résultat 2

# Représentation des nombres entiers:

## ❑ Opération arithmétiques en utilisant le complément à 2

### Exemples:

Expliquer alors le résultat de l'affichage de l'opération **g** `g. s08_var = 60`

```
s08_var = -132-64;  
printf("  g. s08_var = %d\n", s08_var);
```

a=-132  
b= -64

**1)** Convertir en binaire la valeur absolue **VA** des nombres a et b

VA(a)=VA(-132)=132

VA(b)=VA(-64)=64

Convertir 132 en binaire

1 0 0 0 0 1 0 0

Convertir 64 en binaire

0 1 0 0 0 0 0 0

↓ Pour arrondir de la même taille

**2)** Ajouter le bit de signe (si le nombre est signé) pour a (pour a et b s'agit d'une opération)

+132      0 1 0 0 0 0 1 0 0

+64        0 0 1 0 0 0 0 0 0

# Représentation des nombres entiers:

## □ Opération arithmétiques en utilisant le complément à 2

3) Appliquer le complément à 2 s'il s'agit d'un nombre négatif  $-b = CA2(b)$  et  $-a = CA2(a)$



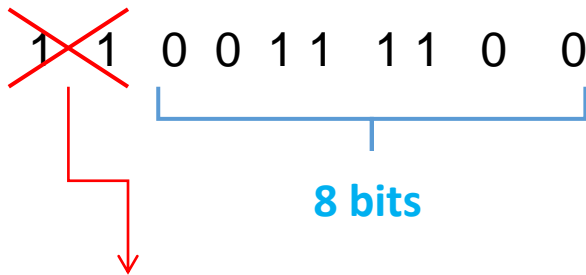
4) Effectuer l'opération s'il s'agit d'une opération  $-a - b = CA2(+132) + CA2(+64)$

$$\begin{array}{r} + \quad \begin{array}{cccccccc} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{array} \\ \quad \begin{array}{cccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \\ \hline = \quad \begin{array}{cccccccc} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{array} \end{array}$$

# Représentation des nombres entiers:

## ❑ Opération arithmétiques en utilisant le complément à 2

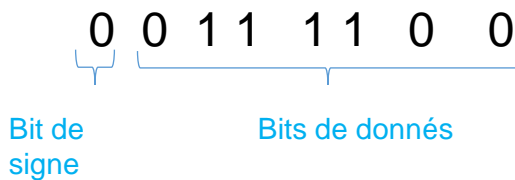
5) Réserver que **8 bits** et analyser puisqu'il s'agit d'un **caractère**:



**Débordement: Dépassement de la mémoire**

***Est-ce qu'il s'agit d'un nombre signé ou non signé?***

*Il s'agit d'un nombre signé*



***Est-ce qu'il s'agit d'un nombre positif ou négatif?***

*Le bit de signe est égale à 0 donc c'est un nombre positif → c'est la valeur exacte*  
*Affichage de 60*



# Représentation des nombres entiers:

## ❑ Opération arithmétiques en utilisant le complément à 2



▪ **Comment éviter ce message erroné et avoir un résultat correcte 'Over flow'?**



**SOLUTION** Pour avoir un affichage correcte, il faut augmenter la taille des nombre; déclarer comme short (taille =16 bits) ou bine int (taille =32 bits)

```
printf(" a. u08_var = %d\n", u08_var);  
u08_var = 283;  
printf(" b. u08_var = %d\n", u08_var);  
  
u08_var = 511;  
printf(" c. u08_var = %d\n", u08_var);  
  
s08_var = 511;  
printf(" d. s08_var = %d\n", s08_var);
```

Compilation



**Résultats**

```
b. u08_var = 283  
c. u08_var = 511  
d. s08_var = 511
```

# **Les références:**

## ➤ Livre:

« Circuits logiques combinatoires et séquentiels », Hichem TRABELSI

## ➤ Site web

<https://www.technologuepro.com/cours-systemes-logiques-3/chapitre-1-1-systeme-de-numeration-et-codage-des-informations.html>

<http://villemin.gerard.free.fr/Wwwgvm/Numerati/BINAIRE/Negatif.htm>

[https://www.rocq.inria.fr/secret/Anne.Canteaut/COURS\\_C/chapitre1.html](https://www.rocq.inria.fr/secret/Anne.Canteaut/COURS_C/chapitre1.html)