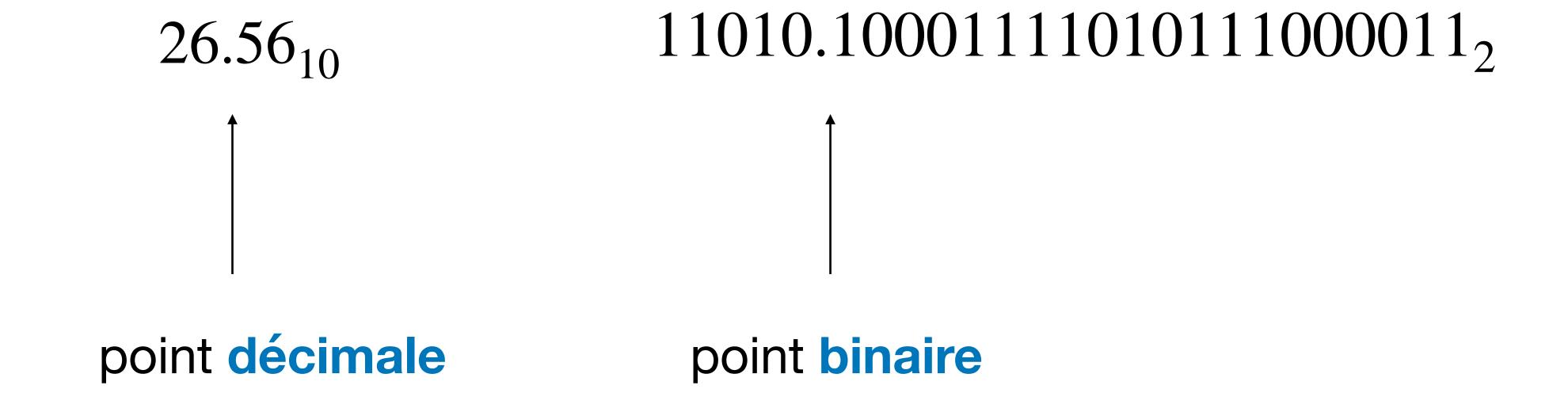
# Nombres à virgule fixe

### Point décimale et point binaire:

C'est un point qui sépare la partie entière de la partie fractionnaire



### Représentation binaire des nombres à virgule

p.ex. convertir le binaire  $11.011_2$  en décimale

4	27	26	2 <sup>5</sup>	24	$2^3$	$2^2$	21	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	2-4	$2^{-5}$	$2^{-6}$	$2^{-7}$	$2^{-7}$	
							1	1	0	1	1						

$$1 \times 2^{1} + 1 \times 2^{0} + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$$
  
 $2 + 1 + 0 + 0.25 + 0.125$ 

 $3.375_{10}$ 

# Nombres à virgule : type de données

Virgule flottante:

float

Virgule fixe:

doit être implémenté

le nombre de chiffres/ bits dans la partie fractionnaire peut varier

le nombre de chiffres/ bits dans la partie fractionnaire est fixé

## Nombres à virgule en base 10

exemple en base 10 : représentation de 1234

#### Virgule flottante:

Le nombre de **chiffres** après le point décimale flotte. Je peux choisir librement où placer la virgule grâce à l'exposant : flexibilité dans la représentation

$$1234 \times 10^0 = 1234$$

$$123.4 \times 10^1 = 1234$$

$$12.34 \times 10^2 = 1234$$

$$1.234 \times 10^3 = 1234$$

$$0.1234 \times 10^4 = 1234$$

#### Virgule fixe:

Le nombre de **chiffres** après le point décimale est fixé. Je ne peux pas choisir librement où placer la virgule : limitation dans la représentation

p.ex. Une représentation décimale d'un nombre à virgule fixe dont la partie fractionnaire a été fixé à 2 **chiffres** 

$$12.34 = FixedPointNumber$$

en sachant que la partie fractionnaire a été fixé à 2 **chiffres**, je peux décaler deux **décimales** vers la gauche et ainsi retrouver le nombre d'origine:

$$12.34 \times 10^2 = 1234$$

## Nombres à virgule en base 2

exemple en base 2 : représentation de 1234

#### Virgule flottante:

Le nombre de **bits** après le point décimale flotte.

Je peux choisir librement où placer la virgule grâce à l'exposant : flexibilité dans la représentation

$$1001101001.0_2 \times 2^1 = 1234_{10}$$

$$100110100.10_2 \times 2^2 = 1234_{10}$$

$$10011010.010_2 \times 2^3 = 1234_{10}$$

$$1001101.0010_2 \times 2^4 = 1234_{10}$$

$$100110.10010_2 \times 2^5 = 1234_{10}$$

#### Virgule fixe:

Le nombre de **bits** après le point décimale est fixé. Je ne peux pas choisir librement où placer la virgule : limitation dans la représentation

p.ex. Une représentation binaire d'un nombre à virgule fixe dont la partie fractionnaire a été fixé à 2 **bits** 

$$100110100.10_2 = 308.5_{10} = FixedPointNumber$$

en sachant que la partie fractionnaire a été fixé à 2 **bits**, je peux décaler deux **bits** vers la gauche et ainsi retrouver le binaire d'origine:

$$100110100.10_2 \times 2^2 = 10011010010_2 = 1234_{10}$$

# Nombres à virgule fixe : Format Q

Dans cet exercice on veut convertir un int ou un float en format Q8 c'est-à-dire la partie fractionnaire est fixé à 8 bits.

Exemple pour un float :

$$26.56_{10} <=> 11010.10001111010111000011_{2}$$

$$11010.1000111110101111000011_{2}$$

$$11010.100011111_{2}$$

Exemple pour un int:

$$10_{10} <=> 1010_2$$

$$1010.00000000_2$$

Mais on ne peut pas dire à l'ordinateur de supprimer ou d'ajouter des bits. Cependant on peut faire stocker la valeur voulue en entier (sans la virgule) dans un int:

$$1101010001111_2 \qquad \text{stocke} \qquad 11010.10001111_2 \qquad \qquad 10100000000_2 \quad \text{stocke} \qquad 1010.0000000_2 < = > 6799_{10}$$

Il suffit donc de décaler 8 bits vers la gauche pour stocker un nombre à virgule fixe dans un int.

### Convertir un float en nombres à virgule fixe

#### Code de l'exercice :

Convertir un nombre (float ou int) en nombre à virgule fixe dont la partie fractionnaire est fixé à 8 bits et stocker la valeur dans un int

$$26.56_{10} <=> 11010.10001111010111000011_2$$

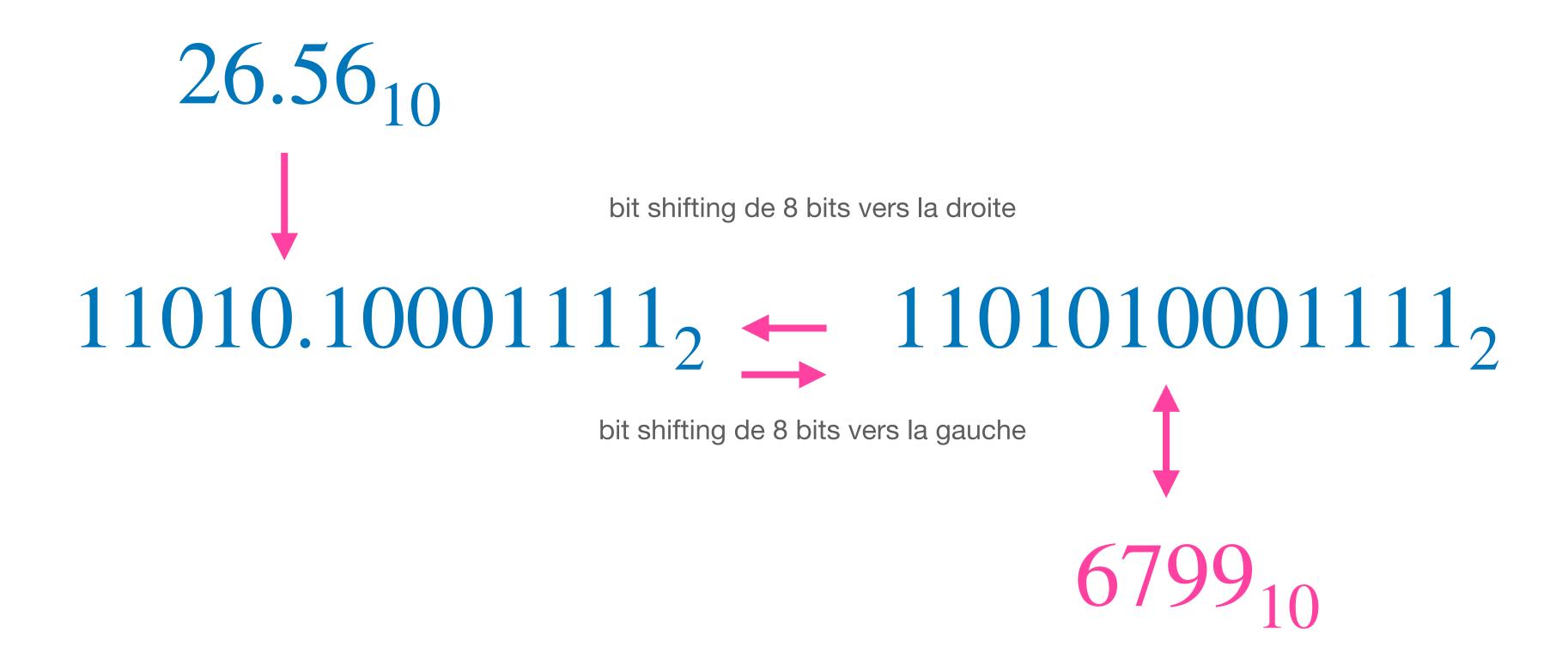
bit shifting de 8 bits vers la gauche 
$$1101010100011111.0101111000011_2 <=> 6799.36_{10}$$

casting en int

$$1101010001111_2$$

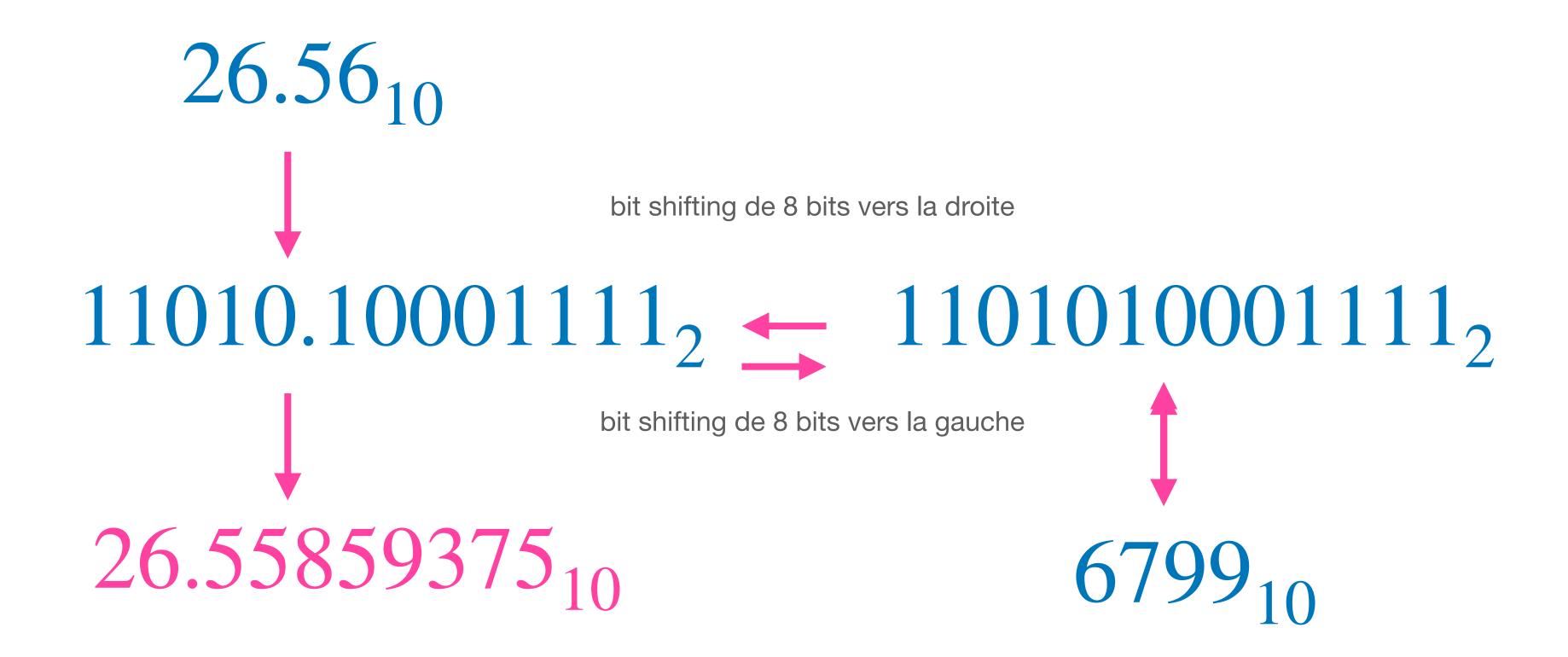
$$\langle = > 6799_{10} \rangle$$

#### Convertir un FixedPointNumber en float



On peut s'apercevoir que les deux binaires diffèrent uniquement en la position du point binaire. On peut donc considérer la représentation des entiers comme des "cas spéciaux" des nombres à point fixe où le point binaire est à la position 0.

#### Convertir un FixedPointNumber en float



La position du point binaire est importante uniquement lorsqu'on doit afficher le nombre ou appliquer une opération arithmétique.

Inconvénient des nombres à virgule fixe : perte de la précision

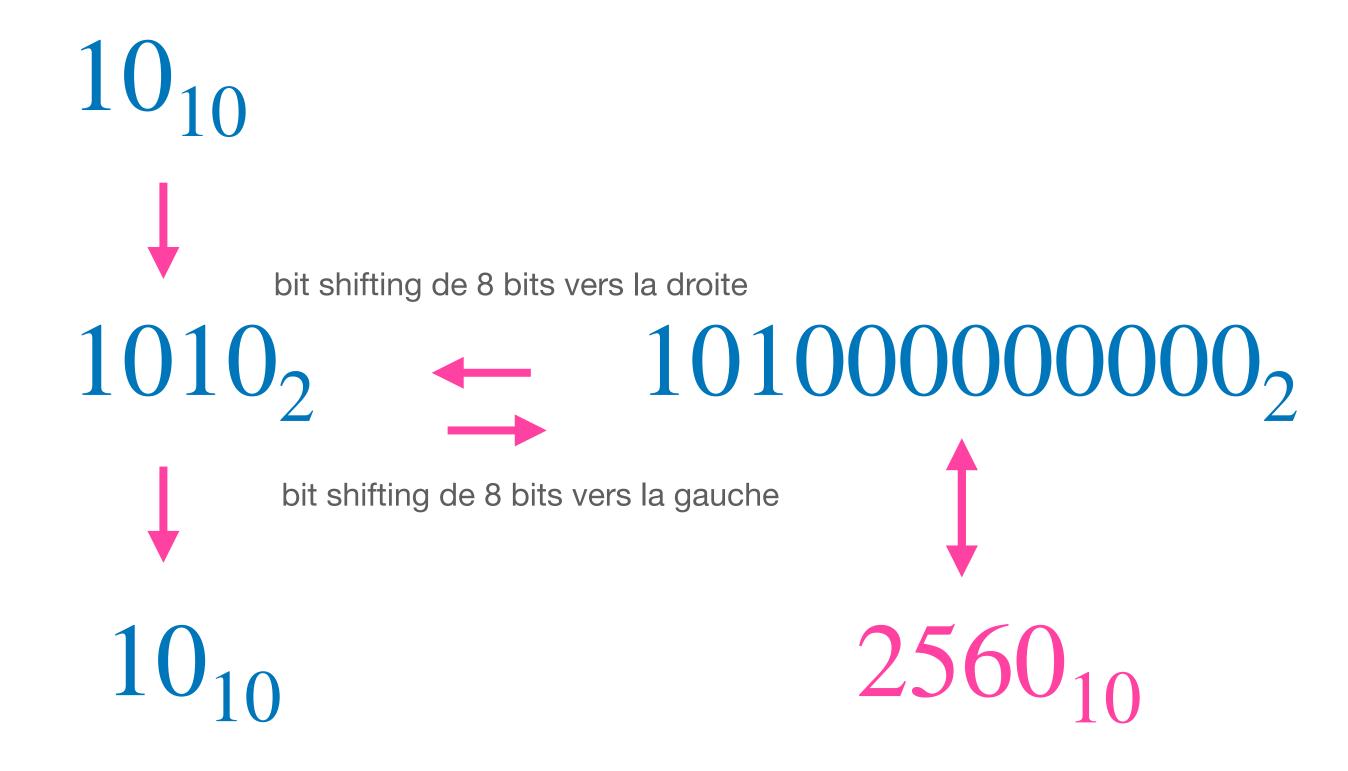
## Convertir un int en nombres à virgule fixe

#### Code de l'exercice :

Convertir un nombre (float ou int) en nombre à virgule fixe dont la partie fractionnaire est fixé à 8 bits et stocker la valeur dans un int

$$10_{10}$$
 <=>  $1010_2$ 

#### Convertir un FixedPointNumber en int



### Avantages et utilité des nombres à virgule fixe

Avantage des nombres à virgule fixe :

1) Toutes les opérations arithmétiques qu'on appliquer sur un entier, peuvent être appliquées sur le nombre à point fixe.

Les opérations d'arithmétique sur des entiers sont simples et rapides et peu coûteuse.

- 2) Permettent aux ordinateurs qui ne prennent pas en charge les float de travailler avec nombres réels (dont les nombres à virgule font partie) :
- Traitement de signal audio
- Graphiques et jeux embarqués