Pourquoi utiliser le complément à 2

5TTR 6TTR Approfondissement

C'est l'avantage du **complément à 2** : l'addition (et la soustraction en la transformant en addition) se fait avec les **mêmes circuits logiques** que pour les nombres non signés.

Principe

- En **complément à 2**, les nombres négatifs sont représentés de telle façon que l'addition fonctionne naturellement.
- On **ne doit pas traiter le signe à part** : on additionne bit par bit, on garde la retenue (carry), et on ignore la retenue finale si elle dépasse 8 bits.

Exemple 1 : Addition sans dépassement

Prenons `+5` et `+3` sur 8 bits :

```
+5 = 00000101
+3 = 00000011
----------------
=8 = 00001000
```

👉 Résultat correct : **8**.

Exemple 2 : Addition avec un nombre négatif

Calculons `5 + (-3)` sur 8 bits.

- `5` = `00000101`
- `-3` en complément à 2 :
  - `3` = `00000011`
  - Inverse : `11111100`
  - +1 : `11111101` → `-3` = `11111101`

Addition :

```
00000101  (+5)
11111101  (-3)
----------------
00000010  (+2)
```

👉 Résultat correct : **2**.

Exemple 3 : Somme négative

Calculons `-5 + (-3)` :

- `-5` = `11111011`
- `-3` = `11111101`

Addition :

```
11111011  (-5)
11111101  (-3)
----------------
11111000  (-8)
```

👉 Résultat correct : **-8**.

Exemple 4 : Dépassement (overflow)

Calculons `+120 + +10` :

- `120` = `01111000`
- `10` = `00001010`

Addition :

```
01111000  (+120)
00001010  (+10)
----------------
10000010
```

Résultat binaire : `10000010` En signé : c'est **-126** ! 👉 Problème d'**overflow** : le résultat sort de la plage [-128, 127].

À retenir

1. **Complément à 2 = addition unique** (pas besoin de règle spéciale pour les négatifs).
2. L'addition est correcte tant qu'il n'y a pas **overflow**.
3. L'overflow apparaît quand on additionne deux nombres du **même signe** et qu'on obtient un résultat du signe opposé.

Vidéo à la une à regarder sur Youtube: http://youtu.be/OMSLBhOiJns