Introduction à la numération binaire

bstorm TRAINING

Qu’est-ce que la numération?

- Système utilisé pour représenter les nombres
- Ensemble de règles et de symboles pour écrire des nombres.

Parmi les plus populaires:

- Décimal ou numération de base 10
- Octale ou numération de base 8
- Hexadécimal ou numération de base 16
- Binaire ou numération de base 2

La numération binaire

- Elle n'utilise que 2 chiffres: 0 et 1
- Chaque chiffre ou digit s'appelle un bit
- Les bits sont regroupés par 8 et constituent un byte (ou octet) => 11001010
- La lecture se fait de droite à gauche

Décoder un byte (octet)

- Chaque position dans le byte correspond à une puissance de 2

Byte: 1 1 0 0 1 0 1 0

Position: [7][6][5][4][3][2][1][0]

Calcul: $2^7 + 2^6 + 0^5 + 0^4 + 2^3 + 0^2 + 2^1 + 0^0 =$

$128 + 64 + 0 + 0 + 8 + 0 + 2 + 0 = 202$

Représentation binaire

- Nous pouvons nous simplifier la représentation binaire d'un byte en exprimant pas les zéros excédentaires à gauche

Exemples:

00001101 => XXXX1101 => 1101

00101001 => XX101001 => 101001

(*Les X symbolisent les zéros supprimés visuellement — seul le résultat final compte, cette simplification est utile en lecture humaine*)

Conversion simplifiée

- Sur base de la représentation simplifiée nous pouvons traduire les bytes de la manière suivante:

$\Rightarrow \quad 1101 = 2^3 + 2^2 + 2^0 = 8 + 4 + 0 + 1 = 13$

$\Rightarrow 11111 = 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 16 + 8 + 4 + 2 + 1 = 31$

$\Rightarrow \qquad 10 = 2^1 = 2$

A votre tour…!

Effectuez les conversions binaire => décimal suivantes:

11010101 =>	00110011 =>	11111111 =>
01101011 =>	11001100 =>	01101101 =>
10101010 =>	10010011 =>	10100101 =>
01010101 =>	01110111 =>	00010010 =>
11110000 =>	11101110 =>	11101010 =>

L’addition binaire

- Addition mathématique de mots binaires/ de bytes
- S’effectue bit par bit, de droite à gauche
- Attention à la notion de retenue.

Règles d'addition

Il y a 4 règles de calcul à retenir:

- 0 + 0 = 0
- 1 + 0 = 1
- 0 + 1 = 1
- 1 + 1 = 0 (avec une retenue de 1)

Le mécanisme de retenue consiste à ajouter 1 au bit de gauche suivant suite à l'addition de 2 bits à 1.

Exemple d'addition

```
  1101      (13)
+ 0001      (  1)
----------
  1110      (14)
```

1 + 1 = 0 (retenue de 1)

0 + 0 (+ 1 retenu) = 1

1 + 0 = 1

1 + 0 = 1

Addition de deux bytes

Reportons cette addition simplifiée à l'échelle d'un byte:

```
  00001101      (13)
+ 00000001      (  1)
 -------------
  00001110      (14)
```

Vous remarquez que le résultat reste inchangé. Les deux bytes ont cependant été correctement écrits.

Les cas de dépassements

Que se passe-t-il si nous souhaitons ajouter un bit à un byte déjà plein?

```
  11111111      (255)
+ 00000001      (   1)
 -------------
1 00000000      (256)
```

Nous nous retrouvons en situation de dépassement (**overflow**): le byte a atteint sa capacité maximum, l'ajout de 1 bit engendre la génération d'un 9^{e} bit ce qui dépasse la taille normale d'un byte.

Comment sont gérés les cas de dépassements?

Cela dépend du **contexte d'exécution** :

- **Processeur (CPU)** : certains déclenchent un **drapeau d'overflow** dans le registre d'état.
- **Langage de programmation** :
  - En **C** : l'overflow peut être silencieux (valeur tronquée à 8 bits).
  - En **Python** : les entiers ne sont pas limités en taille par défaut.
  - En **Java** : les types ont des tailles fixes, et l'overflow est "circulaire".
- **Système mémoire** : peut allouer un nouveau byte ou **changer de type** si besoin (ex : de uint8 à uint16).

A vous de jouer…!

Effectuez les additions binaires suivantes et traduisez le résultat obtenu en base 10:

00010101 + 01101101 =

11011011 + 01110010 =

01100010 + 01000010 =

10001010 + 00111011 =

01101111 + 00001100 =

11010100 + 11011010 =

01010101 + 10101010 =

11001100 + 11001100 =

10011001 + 01111011 =

11110001 + 00111011 =

La soustraction binaire

- Soustraction mathématique de mots binaires/ de bytes
- S'effectue bit par bit, de droite à gauche
- Attention à la notion d'emprunt.

Règles de soustraction

Comme pour l'addition, il y a **4 règles de calcul** à retenir et à appliquer:

- 0 - 0 = 0
- 1 - 0 = 1
- 0 - 1 = 1 (avec un emprunt de 1 bit à gauche)
- 1 - 1 = 0

Le mécanisme d'emprunt consiste à prendre un bit à gauche afin de pouvoir effectuer l'opération 0 - 1

Exemple de soustraction

```
  1101      (13)
- 0001      (  1)
----------
  1100      (12)
```

1 - 1 = 0

0 - 0 = 0

1 - 0 = 1

1 - 0 = 1

Exemple de soustraction avec emprunt de bit

```
  1101      (13)
- 0011      ( 3)
----------
  1010      (10)
```

1 - 1 = 0

1 0 - 1 = 1 (1 est emprunté à gauche)

0 - 0 = 0 (0 car bit emprunté à l'étape précédente)

1 - 0 = 1

Les cas de dépassements

Que se passe-t-il si nous souhaitons soustraire un byte d'une valeur supérieure au byte initial

```
  00000001    (1)
- 00000010    (2)
--------------
```

???

En décimal, 1 - 2 = -1

En programmation, le comportement peut être tout autre.

Le Wrap - around

Dans le cas d'un byte non-signé, les résultats possible seront soit une erreur de sous-dépassement (underflow), soit un wrap-around, c'est-à-dire une affectation cyclique de valeur.

Passé le cap du 0, on revient à la valeur maximale du byte (255):

1 - 2 = 255,

1 - 3 = 254,

1 - 4 = 253, …

Le complément à 2

La méthode de calcul par complément à 2 est une technique de calcul visant à transformer une soustraction binaire en addition. Il permet également un élargissement du domaine de valeur en étendant le résultat à une valeure signée.

Elle sert à la fois à représenter les entiers négatifs et également à effectuer uniquement des opérations d'addition. (comportement du processeur)

Le complément à 2

- Un byte non signé a une plage de valeur qui va de 0 à 255 soit 256 valeurs possibles
- Un byte signé a une plage de valeur allant de -128 à 127
  - De 0 à 127 en POSITIF, soit 128 valeurs
  - De -1 à -128 en NÉGATIF, soit 128 valeurs

Le complément à 2

Soit la soustraction des 2 mots binaires suivant:

0100 (4)

- 0010 (2)

???

Le complément à 2

Avec la soustraction habituelle nous allons effectuer les opérations suivantes:

  0 - 0 = 0

1 0 - 1 = 1 (avec un bit emprunté)

  0 - 0 = 0

  0 - 0 = 0

Nous obtenons ainsi le résultat suivant: 0010 (2)

Le complément à 2

Le complément à 2 cherche à transformer une soustraction en addition.

Il y a 2 opérations à faire:

- Inverser les bits de l'opérande 2, cette inversion s'appelle le complément à 1 et permet l'expressivité de la négativité en binaire

- Ajouter un bit à la valeur inversée obtenue, ce bit est appelé bit de correction, il permet de régler le problème d'asymétrie entre la plage de valeurs positives et la plage de valeurs négatives

Une fois les 2 opérations effectuées il ne reste plus qu'à additionner les 2 opérandes.

Le complément à 2

Notre calcul initial va donc subir les transformations suivantes:

- Inversion des bits de 0010 => 1101
- Ajout d'un bit à la valeur inversée => 1101 + 1 => 1110
- Addition : 0100 + 1110 = 10010

Nous sommes sur 4 bits, on va donc ignorer le dernier bit de retenue => 0010 (2)

Notre calcul attendait un résultat positif. Il n'y a dès lors rien d'autre à faire.

Le complément à 2

Soit le cas d'une soustraction qui engendre un underflow:

0010 (2)

- 0100 (4)

???

Le complément à 2

Appliquons la méthode du complément à 2:

- Inversion des bits: 0100 => 1011
- Ajout de 1: 1011 + 1 => 1100

Quelle est la prochaine étape: additionner les 2 opérandes.

Le complément à 2

0010 (2)

+ 1100 (12)

1110 (14)

Comment interpréter le résultat?

Le complément à 2

Si nous traduisons le résultat 1110 en base 10 comme un nombre non signé, nous obtenons 14.

Mais nous savons d'expérience que 2 - 4 = -2 et non 14☺️

<u>Nous sommes en calcul par complément à 2.</u>

C'est là qu'intervient une notion importante: le **MSB, M**ost **S**ignificant **B**it, c'est-à-dire le bit de poids fort. Le MSB est le bit le plus à gauche de la chaîne binaire. Il est aussi appelé **bit de signe.**

Si MSB = 0, le **résultat est positif**, il n'y a rien à faire pour l'interpréter

Si MSB = 1, le **résultat est négatif** - c'est notre cas ici: 1110

Le complément à 2

Quand le résultat du calcul est négatif (MSB = 1), il faut interpréter correctement la réponse en appliquant à nouveau les deux opérations inverses du complément à 2:

- Inversion des bits : 1110 => 0001
- Ajout de 1 à la valeur inversée: 0001 + 1 => 0010

Nous obtenons ainsi la valeur décimale **2**

Puisque le MSB du résultat initial était à 1, cela indique que le nombre est **négatif**. Le résultat final est donc **-2**

A vous de jouer…!

Effectuez les soustractions suivantes en utilisant la méthode traditionnelle :

000111 - 000101 =>

111010 - 001011 =>

101100 - 110100 =>

101010 - 010101 =>

101101 - 011100 =>

001010 - 000111 =>

110110 - 010000 =>

111000 - 010111 =>

110000 - 100000 =>

000111 - 000011 =>

A vous de jouer…!

Effectuez les soustractions suivantes en utilisant la méthode du complément à 2 :

111101 - 010101 =>

011010 - 111010 =>

101001 - 000101 =>

001011 - 011001 =>

001000 - 000111 =>

101010 - 111000 =>

111011 - 111111 =>

010101 - 101111 =>

110110 - 111011 =>

0001100 - 0110100 =>