NSI 1ère - Données

Complément à deux - Travaux dirigés

qkzk

2021/03/19

V/F

1. Tous les entiers relatifs admettent une unique représentation en complément à deux.
2. En complément à deux sur un octet, on peut représenter des nombres de -128 à 127.
3. L'addition en complément à deux sur un octet de 100 et 110 donne un résultat conforme.

1. Additions d'entiers relatifs

1. Donner la représentation binaire de 27 et 33.
2. Donner la représentation en binaire signé sur un octet de -27 et 33.
3. Vérifier que l'addition des deux représentation précédentes ne donne pas la représentation de 6.
4. Recommencer en utilisant le complément à deux sur un octet et vérifier que cette fois le résultat est conforme.

2. Tailles minimales et maximales

1. On encode des entiers sur 2 octets.
   a. Les entiers sont tous positifs. Quels sont les plus petits et plus grands entiers qu'on puisse encoder ?
   b. Les entiers sont encodés en complément à 2. Même question.
2. Dans un programme les entiers sont encodés en complément à deux. La taille utilisée a été perdue.
   On sait que `111010101` est un entier négatif.
   a. Quelle est la taille employée ?
   b. Quels sont les plus petits et plus grands entiers qu'on puisse encoder ?

3. Complément à deux sur un octet.

On encode les entiers en complément à 2 sur un octet.

1. Compléter le tableau suivant : *Lorsque c'est impossible, écrire une croix*

Entier	Complément à 2 sur un octet
1	
127	
	`1111 1111`
-12	
-93	
101	
-139	
	`0101 1100`
	`1101 0011`

2. Réaliser les additions des éléments du tableau en complément à 2, deux lignes à la fois :
   premier + second, second + troisième, troisième + quatrième.
3. Vérifier les résultats obtenus.

4. Complément à deux, un programme

Dans cet exercice nous allons écrire une fonction qui réalise le complément à deux en Python.

- La foncion `int` permet d'obtenir la représentation décimale d'un entier depuis n'importe quelle base $b \leq 36$

```
>>> int('10101', 2) # en binaire
21
>>> int("azerjsdkfjlqdkfj", 36) # les "chiffres" 0-9 et a-z
2428190859766979995068079
```

- La fonction `bin` donne la représentation binaire d'un entier naturel

```
>>> bin(120)
'0b1111000'
```

1. Écrire une fonction qui inverse un bit :

```
>>> inverser('0')
'1'
>>> inverser('1')
'0'
```

2. Python permet de transformer une chaîne de caractère en tableau avec la fonction `list`

```
>>> list("bonjour")
['b', 'o', 'n', 'j', 'o', 'u', 'r']
```

Écrire une fonction Python :

1. qui prend un entier positif (`int`),
2. le converti en binaire,
3. enlève `0b` au début,
4. Le converti en un tableau de chaînes de caractères.

Exemple :

```
>>> vers_tableau(123)
['1', '1', '1', '1', '0', '1', '1']
```

3. Écrire une fonction python qui ajoute des `'0'` au début d'un tableau jusqu'à ce qu'il ait la taille souhaitée :

```
>>> aouter_zeros(['1', '1'], 4)
['0', '0', '1', '1']
>>> aouter_zeros(['1', '1', '1', '1'], 4)
['1', '1', '1', '1']
```

4. Écrire une fonction python qui retourne l'indice du dernier `'1'` dans un tableau qu'on lui passe en paramètre :

```
>>> dernier_1(['0', '1', '0'])
1
```

On supposera que le tableau contient toujours un '1'.

5. La méthode `join` des chaînes de caractères permet de convertir une liste de chaîne en une chaîne :

```
>>> '|'.join(['1', '3', '5'])
'1|3|5'
>>> ''.join(["b", "o", "n"])
'bon'
```

Utiliser les fonctions précédentes et la méthode `join` pour écrire le complément à 2 sur une taille donnée.

5. Compléments

Ces exercices progressifs permettent de vérifier la compréhension des élèves sur la représentation des entiers en complément à deux, en passant de la conversion de nombres décimaux simples en binaire signé, à des opérations plus complexes avec des nombres à virgule fixe.

Voici une série d'exercices progressifs sur la représentation des entiers en complément à deux, faisables sans ordinateur:

1. Conversion de nombres décimaux en binaire signé:

Convertir les nombres décimaux suivants en binaire signé en utilisant le complément à deux: 5, 10, -3, -7.

2. Conversion de nombres binaires signés en décimaux:

   Convertir les nombres binaires signés suivants en décimaux: 0101, 1100, 1010, 1111.

3. Addition de nombres binaires signés:

   Ajouter les paires de nombres binaires signés suivantes en utilisant la méthode de complément à deux: 0011 + 0100, 1101 + 1001, 0110 + 1001.

4. Soustraction de nombres binaires signés:

   Soustraire les paires de nombres binaires signés suivantes en utilisant la méthode de complément à deux: 0100 - 0011, 1111 - 1100, 1001 - 0110.

5. Conversion de nombres décimaux à virgule fixe en binaire signé:

   Convertir les nombres décimaux suivants en binaire signé à virgule fixe avec une précision de 4 bits: 3,25, -2,75, 7,375, -6,125.

6. Opérations arithmétiques avec des nombres à virgule fixe en binaire signé:

   Ajouter ou soustraire les paires de nombres à virgule fixe suivantes en utilisant la méthode de complément à deux et en respectant la précision de 4 bits: 0011,0101 + 1101,0010, 1010,1100 - 0100,0110.