10.3 Représentation des nombres négatifs en CP2 (Complément à 2)

Il existe trois méthodes pour calculer le complément à 2 (CP2) d'un nombre binaire.

10.3.1 Première méthode :

La première consiste à le soustraire à la puissance de 2 immédiatement supérieure. Par exemple : trouver le complément à 2 du nombre N= 1 0 0 0 1 1 1 $0_{(2)}$,

1 0 0 0 0 0 0 0 0
- 1 0 0 0 1 1 1 0
0 1 1 1 0 0 1 0

Donc CP2(N) = 0 1 1 1 0 0 1 0

10.3.2 Deuxième méthode :

Elle consiste à trouver d'abord le complément à 1 et à ajouter 1 au résultat. Par exemple :

N= 1 0 0 0 1 1 1 $0_{(2)}$, CP1(N) = 0 1 1 1 0 0 0 1

0 1 1 1 0 0 0 1
+ 1
0 1 1 1 0 0 1 0

Donc CP2(N) = 0 1 1 1 0 0 1 0

10.3.3 Troisième méthode

Consiste à conserver tous les bits à partir de la droite jusqu'au premier 1 compris et de changer les autres bits de 1 à 0 ou de 0 à 1. Pour l'exemple précédent, on obtient :

CP2(1 0 0 0 1 1 1 0)= 0 1 1 1 0 0 **1 0** → **bits conservés**

Question :

Quelle est la valeur décimale du nombre binaire suivant : 10110110 ?

Réponse :

Le bit de poids fort indique qu'il s'agit d'un nombre négatif. Donc la Valeur décimale = - CP1(10110110) = - (01001010)2= - (74)10

Question :

Quels sont les nombres qu'on peut représenter sur 04 bits ?

Valeurs En CP2				Valeurs En binaire	chiffre En décimal
0	0	0	0	0000	+0
0	0	0	1	0001	+1
0	0	1	0	0010	+2
0	0	1	1	0011	+3
0	1	0	0	0100	+4
0	1	0	1	0101	+5
0	1	1	0	0110	+6
0	1	1	1	0111	+7

Valeurs En CP2				Valeurs En binaire	Chiffre En décimal
1	0	0	0	-1000	-8
1	0	0	1	-0111	-7
1	0	1	0	-0110	-6
1	0	1	1	-0101	-5
1	1	0	0	-0100	-4
1	1	0	1	-0011	-3
1	1	1	0	-0010	-2
1	1	1	1	-0001	-1

Tableau 7 : Représentation des nombres par la méthode CP2

Sachant que le bit du poids fort est utilisé pour représenter le signe du nombre, on peut déduire que sur 4 bits :

- Le plus grand nombre positif représentable est donc 0111 ce qui représente $2^3 - 1$ soit +7
- Le plus petit négatif est codé par 1000, ce qui donne la valeur binaire -1000, soit $-8(-2^3)$ en décimal.

Réponse :
Donc, d'après le tableau 7, on constate que sur 04 bits, on peut représenter les nombres qui sont dans l'intervalle $[-8_{(10)},+7_{(10)}]$, soit $[-2^3 , +(2^3 - 1)]$

Plus généralement, si on travaille sur n bits, l'intervalle des valeurs qu'on peut représenter en CP2 est : $[-(2^{n-1}), +(2^{n-1} - 1)]$.

Avantage :
Un seul codage pour le nombre 0. Par exemple sur 8 bits :
$+0_{(10)} = 00000000_{(2)}$, $-0_{(10)}$=cp2(00000000)=$00000000_{(2)}$

10.4 Soustraction en base 2 en utilisant les compléments

Avant de faire une soustraction en complément à 1 ou 2 il faut s'assurer que le diminuteur et le diminuende ont le même nombre de bits. Cette remarque est très importante sinon on ne retrouve pas le résultat de la soustraction.

10.4.1 Soustraction par complémentation à 1.

- La soustraction par complément à 1 revient à calculer le complément à 1 du diminuteur ensuite l'ajouter au diminuende. La dernière retenue est ajoutée au résultat.

- Lorsque le dernier bit du résultat (poids fort)=1, cela veux dire que le résultat est négatif. On calcule donc le complément à 1 de ce dernier afin d'obtenir le résultat final.

Exemple 1:
Effectuons sur 5 bits, l'opération (+8) + (- 9).

Solution:
Les nombres doivent être sur 5 bits y compris le bit de signe

$(+8) = 01000_{(2)}$
$(+9) = 01001_{(2)}$

Le complément à 1 de 01001 est 10110 = $-9_{(10)}$

```
  0 1 0 0 0   ←  diminuende
+ 1 0 1 1 0   ←  diminuteur
= 1 1 1 1 0
```

Le bit du signe =1 → résultat négatif → le résultat = - Cp1(11110).
Dans ce cas, calculons le complément à 1 du résultat

Cp1(11110)=$00001_{(2)}$. Cela veux dire que le résultat = $-1_{(2)}$

Exemple 2: toujours sur 8 bits effectuons l'opération : (-8) + (-9)

Solution :
Dans ce cas chaque nombre est représenté par son complément à 1 :
(+8) = 01000, (-8)= cp1(01000)=10111
(+9) = 01001, (-9)= cp1(01001)=10110

```
1  1 0 1 1 1
   1 0 1 1 0
   0 1 1 0 1
+            1
   0 1 1 1 0
```

Remarque 1:
Nous remarquons que le dernier bit du résultat = zero. Ce qui veux dire que le résultat est positif. Hors, l'addition de deux nombres négatifs ne peut donner qu'un nombre négatif !!!. Ce cas s'appelle « dépassement » , en anglais, « overflow ». Il résulte du fait que sur 5 bits on ne peux représenter que les nombres qui sont dans l'intervalle [-15 , 15] (voir la section 8.2) alors que -8-9 donne -17 qui est hors intervalle.

Remarque 2
Dans l'addition arithmétique signée, par exemple (A+B), on dit qu'il y a débordement si et seulement si les deux opérandes A et B sont de même signe et le résultat S est de signe différent.

10.4.2 Soustraction par complément à 2

- Complémenter le diminuteur ensuite l'ajouter au diminuende. La dernière retenue est ignorée.
- Tout comme la méthode du complément à 1, lorsque le dernier bit du résultat (poids fort)=1, cela veux dire que le résultat est négatif. On calcule donc le complément à 2 de ce dernier afin d'obtenir le résultat final.

Exemple 3:
Effectuons la même opération en complément à 2. (+8) - (+9).

Solution
(+8) = $01000_{(2)}$, (+9) = $01001_{(2)}$
Le complément à 2 de 01001 est 10111 = -9

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

Le bit du signe =1 → résultat négatif → le résultat = - Cp2(11111).

Dans ce cas, calculons le complément à 1 du résultat :
Cp2(11111)=00001(2). Cela veux dire que le résultat = -1(2)

11 Les Nombres Réels

11.1 Représentation en virgule fixe

Soit un nombre N tel que N= - $1010{,}1001_{(2)}$. Le codage du nombre en virgule fixe consiste à définir la position de la virgule selon un format donné, c'est-à-dire la taille de la partie entière ainsi que la taille de partie fractionnaire. Les bits à gauche de la virgule représentent la partie entière signée du nombre tandis que la partie droite représente la parie fractionnaire. Dans l'exemple ci-dessous (tableau 8), la partie entière signée est sur 8 bits et la partie fractionnaire sur 8 bits.

1	0	0	0	0	1	0	1	0	1	0	0	1	0	0	0	0
signe	Partie entière								Partie fractionnaire							

Tableau 8 : Représentation d'un nombre réel par la méthode de la virgule fixe

Exemple :

1. Représenter le nombre réel (-6.125) en format virgule fixe (1 bit de signe, 8 bit pour la partie entière et 7 bits pour la partie fractionnaire.
2. Quelle est le plus petit nombre positif représentable dans ce format.
3. Quelle est le plus grand nombre positif représentable dans le même format.

Solution :

1. Il faut d'abord convertir le nombre en binaire pour pouvoir le représenter en machine. On a : $-6.125_{(10)} = -110{,}001_{(2)}$.On représente donc le nombre selon le format indiqué. On obtient donc : 1|00000110|0010000
2. le plus petit nombre positif est représenté comme suit : 0|00000000|0000001 ce qui donne la valeur $Nmin=2^{-7}$
3. le plus grand nombre positif est représenté comme suit : 0|11111111|1111111

Pour donner l'équivalent en décimale, calculons la partie entière max (PEmax) et la partie fractionnaire max (PFmax).

$PEmax=2^0+2^1+\ldots+2^7=2^8-1$

$PFmax=2^{-1}+2^{-2}+\ldots+2^{-7}=1-2^{-7}$

$Nmax=PE\ max=+PF\ max=2^8-1+1-2^{-7}=2^8-2^{-7}$

11.2 Représentation en virgule flottante

Il existe plusieurs formats de représentation en virgule flottante proposés par l'IEEE (Institute of Electrical and Electronics Engineers) et qui ont été adoptés par les fabricants de microprocesseurs, parmi ces formats, on cite :

- Le format simple précision, utilisant 32 bits.
- Le format double précision, utilisant 64 bits.
- Les formats étendus à simple et à double précision.

Le lecteur intéressé pourra se reporter aux nombreuses références sur le sujet [1-3]. En ce qui suit, nous allons décrire les différentes méthodes de représentation des nombres réels en virgule flottante.

11.2.1 Par la méthode de l'exposant réel.

Il existe deux méthodes pour représenter les nombres réels en virgule flottante. La première consiste à représenter un nombre N par un bit de signe (0 si N est positif, 1 sinon), une mantisse M (qui doit être normalisée) et un exposant Exp (entier positif ou négatif). Le nombre N est donc écrit sous la forme suivante :

$$N=\pm|M|\times 2^{Exp} \quad \text{avec} \quad 0.1_{(2)} \leq |M| < 1_{(2)}$$

Selon le type de machine, un certain nombre de bit est réservé pour la mantisse ainsi que pour l'exposant.

Exemple1 : Représenter N= - $1010,1001_{(2)}$ sur 16 bits en format virgule flottante (12 bits pour la mantisse, 4 bits pour l'exposant et 1 bit pour le signe de la mantisse).

Solution :
On commence par normaliser la mantisse :
N= - $1010,1001_{(2)}$ =-$0.10101001_{(2)}$ x 2^{+4}. (+4 représente le nombre de déplacement de la virgule vers la gauche).

Dans ce cas : M= - $0.10101001_{(2)}$ et Exp= + $4_{(10)}$ =$0100_{(2)}$

1	0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0
Signe	Exp				Mantisse											

Tableau 9 : Représentation d'un nombre réel par la méthode de la virgule flottante –exposant positif-

Exemple2 : représenter N= - $0,001001_{(2)}$ sur 16 bits en format virgule flottante (12 bits pour la mantisse, 4 bits pour l'exposant et 1 bit pour le signe de la mantisse).

Solution :
On commence par normaliser la mantisse :
N= - $0,001001_{(2)}$ =- $0.1001_{(2)}$ x 2^{-2}. (-2 représente le nombre de déplacement de la virgule vers la droite).
Dans ce cas : M=- $0.1001_{(2)}$ et Exp=-$2_{(10)}$

En virgule flottante, les exposants négatifs sont représentés par la méthode du complément à 2 (voir la section 8.3).

|Exp|= +$2_{(10)}$ =$0100_{(2)}$
Exp=-2=CP2(0100) =$1100_{(2)}$

1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
Signe (1 bit)	Exp (4 bits)				Mantisse (12 bits)											

Tableau 10 : Représentation d'un nombre réel par la méthode de la virgule flottante –exposant négatif-

Remarque : le bit le plus à gauche de l'exposant représente le bit du signe.

11.2.2 Par la méthode de l'exposant Biaisé.

La deuxième méthode de la représentation des nombres réels en format virgule flottante consiste toujours à représenter le nombre N sous la forme : N= ±|M| x 2^{Exp_biaise} tel que $0.1_{(2)} \leq |M| < 1_{(2)}$. Mais cette fois-ci, il ne s'agit pas de représenter l'exposant réel mais un autre exposant dit baisé. Ce dernier est calculé en fonction de l'exposant réel comme suit :

$$\textit{Exp\_biaisé} = \textit{Exp} + \textit{biais tel que biais} = 2^n/2 = 2^{(n-1)}.$$

n est le nombre de bit de l'exposant.
La valeur du biais ajoutée, rend la valeur de l'exposant biaisé toujours positif.

Exemple : Prenons le même exemple précédent.
$N = -0{,}001001_{((2)} = -0.1001_{(2)} \times 2^{-2}$
$M = -0.1001_{(2)}$ et $Exp = -2_{(10)}$

Calculons maintenant la valeur de l'exposant biaisé :

$$Exp\_biaisé = -2 + (2^4/2) = 2 + 8 = 10_{(10)} = 1010_{(2)}$$

Ce nombre est représenté comme suit dans le format suivant : 12 bits pour la mantisse, 4 bits pour l'exposant biaisé et 1 bit pour le signe de la mantisse.

1	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
Signe (1 bit)	Exp_biaisé (4 bits)				Mantisse (12 bits)											

Tableau 11 : Représentation d'un nombre réel par la méthode de la virgule flottante –exposant biaisé-

Remarque : *Exp_biaisé* est toujours positif donc à l'inverse du cas précédent, le bit le plus à gauche de l'exposant biaisé ne représente pas le bit du signe mais il fait parti de la valeur.

Démonstration

On a déjà démontré dans la section 8.3 que dans méthode du complément à 2, l'intervalle des valeurs qu'on peut représenter sur n bits est:

$$-2^{(n-1)} \leq N \leq 2^{(n-1)} - 1$$

Si on rajoute la valeur $2^{(n-1)}$ à tous les termes de cette inégalité, on obtient :

$-2^{(n-1)} + 2^{(n-1)} \leq N + 2^{(n-1)} \leq 2^{(n-1)} - 1 + 2^{(n-1)} = 0 \leq N + 2^{(n-1)} \leq 2^{n-1}$

• On pose $N' = N + 2^{(n-1)}$, on aura donc : $0 \leq N' \leq 2^{n-1}$. Dans ce cas on obtient que des valeurs positives. La valeur 2^{n-1} s'appelle le biais ou le décalage.

Remarque :

Si la mantisse est sur k bits et si elle est représenté sur la machine sur k' bits tel que k> k' , alors la mantisse sera tronquée : on va prendre uniquement k' bits.

11.3 Arithmétique

Soit deux nombres réels N1 et N2 tel que : $N1 = M_1 * 2^{exp1}$ et $N2 = M_2 * 2^{exp2}$. On veut calculer N1+N2 ? Deux cas se présentent :

- Si exp1 = exp2 alors $N3 = (M_1 + M_2)\, b^{exp1}$

- Si exp1 ≠ exp2 alors élever au plus grand exposant et faire l'addition des mantisses et par la suite normaliser la mantisse du résultat.

Exemple : soient les deux nombres suivants :

$$N1=0,001101(2) \quad , \quad N2=1, 101(2)$$

Représenter N1+N2 en format virgule flottante selon le format suivant : (12 bits pour la mantisse, 4 bits pour l'exposant et 1 bit pour le signe de la mantisse).

Solution

On normalise d'abord les mantisses pour cela :

$N1=0,1101 *2^{-2}$, $N2=0,1101 *2^{1}$

$N1+N2= 0,1101 *2^{-2} +0,1101 *2^{1}$

$= 0,0001101 *2^{1} + 0,1101 *2^{1}$

$= 0, 1110101 * 2^{1}$

Dans ce cas, N1+N2 est représenté comme suit :

0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0
Signe(1 bit)	Exposant (4 bits)				Mantisse (12 bits)											

Tableau 12 : Addition de deux nombres réels en virgule flottante

La soustraction est faite de la même manière.

12 Exercices :

1. Convertir $94_{(10)}$ en binaire puis compléter le tableau suivant

Chiffres du nombre en base 2								
Rang du chiffre								
Poids de chiffre								
Valeurs								

2. Déterminer x et y tel que : $(3x2, y3)_6 = (134,25)_{10}$
3. Déterminer la base B sachant que $(25)_B=(10111)_2$
4. Effectuer les conversions suivantes :
   - $18_{(10)} = N_{(2)} = N_{(8)} = N_{(16)}$
   - $1010,011_{(2)} = N_{(10)} = N_{(8)} = N_{(16)}$
   - $70C,A_{(16)} = N_{(4)} = N_{(8)} = N_{(2)} = N_{(10)}$

5. Effectuer, en binaire, les additions suivantes :
   - $7A0B_{(16)} + 56,4_{(8)}$
   - $143,12_{(5)} + 31,43_{(6)}$
6. Effectuer en octal : $326_{(8)} + 735_{(8)}$ et en hexadécimal : $3AD2_{(16)} + B0FF_{(16)}$.
7. Effectuer, en binaire, les soustractions suivantes :
   - $35_{(10)} - 25_{(10)}$
   - $5,25_{(10)} - 10,75_{(10)}$
   - $A0137_{(16)} - 33_{(4)}$

8. Effectuer en octal : 524(8) – 263(8) et en hexadécimal : 5E2(16) – 3DA(16).
9. Effectuer, en binaire les multiplications suivantes :
   - $23_{(10)} \times 29_{(10)} = N_{(2)}$
   - $2436_{(8)} \times 37_{(8)} = N_{(2)}$
   - $20{,}7_{(10)} \times 7{,}5_{(10)} = N_{(2)}$
10. Effectuer, en binaire les divisions suivantes :
    - $35_{(10)} : 7_{(10)} = N_{(2)}$
    - $237{,}5625_{(10)} : 71_{(8)} = N_{(2)}$
    - $10000111_{(2)} : 101_{(2)} = N_{(2)}$

11. Représenter sur 8 bits les nombres négatifs suivants par les 3 méthodes : Signe et valeur absolue, complément à 1 et complément à 2.

    -45 ; -128 ; -12, -36

12. Représenter en décimal les nombres binaires qui sont sous le format complément à 2

    10110110 ; 11101000

13. Soient N1, N2 et N3 trois entiers signés représentés sur 08bits. Sachant que N1= 01101110(2), N2=01011001(2) et N3=01110011(2)
    - Effectuer les opérations suivantes en complément à 1 puis en complément à 2.

      N1-N2 ; N2-N3, -N1-N3.
    - Déterminer s'il y a un dépassement (overflow)

14. Représenter le nombre réel -6.125 suivant en format virgule fixe (1 bit de signe, 8 bit pour la partie entière et 7 bits pour la partie fractionnaire.
    - Quelle est le plus petit nombre positif représentable dans ce format.
    - Quelle est le plus grand nombre positif représentable dans le même format.

15. On suppose que les nombres réels sont représentés sur 32 bits en format virgule flottante (23 bits pour la mantisse, 8 bits pour l'exposant et 1 bit pour le signe de la mantisse). Quelle est la représentation binaire du nombre réel 0.015(8) représenté avec un exposant signé:

16. Donner la valeur décimale du nombre suivant codé en format virgule flottante (23 bits pour la mantisse, 8 bits pour l'exposant et 1 bit pour le signe de la mantisse). Puis trouver le plus petit et le plus grand nombre représentables dans ce format.

    46C86000(16)

17. Supposant qu'on utilise le même format de l'exercice 17. Donner la valeur décimale des représentations binaires suivantes

    1 11111010 11110110000000000000000

    1 00000100 11000000000000000000000

    - Calculer l'addition ensuite la soustraction de ces deux nombres puis représenter le résultat.

18. Soit le format à virgule flottante: mantisse=23 bits, exposant=8 bits, calculer l'exposant biaisé du nombre n=7(10) x 2(-3). Donner la représentation binaire du nombre réel suivant représenté dans le même format avec un exposant biaisé égale à -123.75.

13 Conclusion

Dans ce chapitre, nous avons présenté les systèmes de numération les plus utilisés dans les systèmes numériques notamment le système binaire, octal et hexadécimal. Aussi, nous avons étudié les opérations de codage, décodage, conversion de codes ainsi que la réalisation des opérations arithmétiques. Nous avons étudié aussi le codage ainsi que l'arithmétique dans les nombres réels. Dans le chapitre suivant nous allons étudier d'autres codes permettant la représentation des caractères alphanumériques.