-Correction-

1| Architecture des ordinateurs : Applications

Avant de se lancer : être à l'aise en binaire.

Il est de bon goût d'être rapide et efficace pour convertir les nombres de 0 à 15 en binaire. Mais comment savoir que $(13)_{10}=(1101)_2$? Il existe une méthode « puissante » pour les petits nombres :

- On regarde la puissance de 2 la plus proche. Dans le cas de 13, il s'agit de $8=2^3$. On sait qu'il faudra donc 3+1 =4 bits.
- On décompose en puissance de 2, en commençant par la plus proche, et en réitérant : $13=8+5=8+4+1=2^3+2^2+2^0$
- On lit les bits en mettant des 0 ou des 1 selon que les puissances de 2 soient présentes :

$$2^3+2^2+2^0=(1101)_2$$

Je vous invite à vous entrainer **mentalement** à écrire en binaire des nombres au hasard entre 0 et 15.

Attention, cette stratégie montre ses limites pour les grands nombres : personne ne sait quelle est la puissance de 2 juste avant 276 425… Il y a cependant des astuces :

Petite astuce : un nombre binaire de n-1 bits, avec uniquement des 1, vaut 2^n-1. Prouvez-le avec la formule pour la somme des termes d'une suite géométrique ! Exemple : $255=(1111111)_2$.

Autre astuce : une puissance de 2, de la forme 2^n, est codée en binaire par n+1 bits : d'abord 1, suivi uniquement de 0, par exemple $(64)_{10}=2^6=(1000000)_2$

Pour ajouter des nombres en formats binaires, on peut poser l'addition (avec des retenues, qui valent seulement 0 ou 1), comme en CE2… la règle est que « 1+1=0 et je retiens 1 ». Bon, on peut aussi les convertir en base 10…

1 Décomposer les nombres suivants dans leurs bases puis exprimer les en bases 10

$(110010)_2 = 1x2^5+1x2^4+0x2^3+0x2^2+1x2^1+0x2^0 = 50$

On peut écrire la somme de la gauche vers la droite (comme ci-dessus), mais il faut commencer par lire le nombre de bits pour voir à quelle puissance de 2 démarrer (pour n bit, on commence à 2^{n-1}).

On peut aussi l'écrire de la droite vers la gauche (c'est-à-dire lire le nombre à l'envers).

$(1001010)_2 = 1x2^6+0x2^5+0x2^4+1x2^3+0x2^2+1x2^1+0x2^0 = 74$

$(2104)_8 = 2x8^3+1x8^2+0x8^1+4x8^0 = 1\ 092$

Astuce pour les puissances de 8 (ou de 16) sans calculette : $8^3=(2^3)^3=2^9=512$ (on peut savoir par coeur que $2^8=256$, car c'est la capacité de mémoire d'un octet (8 bits)).

$(47623)_8 = 4x8^4+7x8^3+6x8^2+2x8^1+3x8^0 = 20\ 371$

$(5AF)_{16} = 5x16^2+10x16^1+15x16^0 = 1\ 455$

$(3A5DB6)_{16} = 3x16^5+10x16^4+5x16^3+13x16^2+11x16^1+6x16^0 = 3\ 825\ 078$

On ne vous demandera pas ça sans calculette. Encore que poser les calculs est tout à fait possible.

2 Utiliser la méthode de Hörner pour décomposer les nombres suivants :

$(11011)_2 = ((((1x2+1)x2+0)x2+1)x2+1) = 27$

$(1010)_2 = ((1x2+0)x2+1)x2+0 = 10$

3 Exprimer les nombres suivants dans la base binaire

Pour aller de la base 10 vers la base 2, on peut effectuer des divisions successives par 2, jusqu'à ce que le quotient fasse 0. Ensuite, on lit « à l'envers » sur les restes l'écriture en binaire.

$(46)_{10}$ = 46 = 23x2 + 0
23 = 11x2 + 1
11 = 5x2 + 1
5 = 2x2 + 1
2= 1x2 + 0
1 = 0x2 + 1 $(46)_{10} = (101110)_2$

Comme dit dans l'intro de cette correction, on peut aussi chercher la puissance de 2 la plus proche de 2 en dessous, la retirer, etc… par exemple :

46=32+14=32+8+6=32+8+4+2 d'où l'écriture en binaire ci-dessus.

Mais si je vous demande de convertir 25 765 en binaire, connaissez-vous assez de puissances de 2 ? par ailleurs, l'algorithme des « divisions successives par 2» est facile à implémenter en code.

$(123)_{10} =$ $(1111011)_2$

$(48D)_{16} =$ On écrit les chiffres en binaires (à 4 bits) : $4 = (0100)_2$; $8 = (1000)_2$; $D = 13 = (1101)_2$ puis on les « concatène » (et à la fin on enlève le premier 0 qui ne sert à rien ici) :
$(48D)_{16} = (10010001101)_2$

Cette méthode fonctionne pour aller de la base 16 vers la base 2, et repose sur le fait que $16=2^4$. Elle ne marche pas pour aller de la base 10 vers la base 2. Par contre, elle marche pour aller de la base octale (base 8) vers la base 2, en groupant par paquet de 3 bits.

Plus généralement, formalisons : cette astuce permet de passer d'une base b vers une base b^q en groupant des paquets de q bits.

$(901A)_{16} =$ $(1001000000011010)_2$

4 Exprimer les nombres suivants dans la base hexadécimale

On peut passer de la base 10 vers la base 2 (méthode des divisions euclidiennes), puis de la base 2 vers la base 16 (astuce du groupement). On peut aussi effectuer directement les divisions successives par 16. Cela peut paraître pénible de diviser 725 par 16, mais comme on le voit ci-dessous, cela va VITE ensuite : c'est comme si vous divisiez 4 fois d'un coup par 2 !

$(725)_{10}$ = (1011010101)2 = 2D5 ou : 725 = 45x16 + 5
45 = 2x16 + 13 (D en hexa)
2 = 0x16 + 2

$(1010110)_2$ = $(56)_{16}$

De la base 2 vers la base 16? On groupe par paquet de 4 bits : on a $(0110)_2=(6)_{16}$ et $(101)_2=(0101)_2=(5)_{16}$

5 Quel nombre entier naturel maximal peut-ont coder sur 5 bits ? et sur 2 octets ?

On rappelle qu'avec n bits, on peut coder 2^n nombres. Mais n'oubliez pas qu'il faut aussi coder 0 …

5 bits : on peut coder 2^5 -1 = 31 nombres (-1 car on enlève le 0 !)

1 octets vaut 8 bits donc 2 octets valent 16 bits et on peut coder 2^16 -1 = 65 535 nombres

6 Combien de bits/octets sont nécessaires au minimum pour représenter les nombres $(75)_{10}$, $(255)_{10}$, $(256)_{10}$?

On peut chercher à la main la puissance de 2 juste avant, ou résoudre analytiquement.

75 : il suffit de résoudre $2^n-1 \geq 75$ ce qui donne $\log(2^n) \geq 76$ car la fonction log est croissante.

Ensuite, puisque log(2)>0, on peut diviser : $n \geq \log(76)/\log(2) = 6.25$ et donc n = 7 < 8 donc 1 octet !

255 : 8 bits, 1 octet

256 : 9 bits, 2 octets

7 Calculer le nombre de bits minimum nécessaire pour représenter -11 puis donner la représentation en complément à deux (en utilisant les 2 méthodes). Idem avec -7.

On rappelle qu'avec n bits, par méthode du complément à deux, on peut représenter les nombres de -2^{n-1} à 2^{n-1}- (il y a bien 2^n nombres entre les deux).

La puissance de 2 avant -11 est $-16=-2^{5-1}$ donc il faut 5 bits.

Méthode1 : par décalage : Recherche de $-11 + 2^5 = (21)_{10} = (10101)_2$

Donc -11 est représenté par 10101 en complément à deux

Méthode 2 : on cherche la représentation binaire de |-11| = $(11)_{10}$ = $(01011)_2$ (sur 5 bits)
On prend le complément à un (inversion des bits) : 10100 puis on ajoute 1 : 10101

-7 nécessite n=4 bits ; $-7+2^4 = 9$, -7 est représenté par $(1001)_2$
7 = (0111) d'où le complément + 1 = $(1001)_2$

8 On fixe un nombre n≥ 1 de bits. Soit un nombre entier x entre 0 et $2^n - 1$. On note a_0, …, a_{n-1} ses bits. On rappelle que son complément à deux c(x) est obtenu en inversant les bits puis en rajoutant 1.

8.1 Commencer par expliciter la décomposition en base 2 grâce aux bits. Ensuite, utiliser que inverser les bits revient à prendre comme bits $1-a_0$, …, $1-a_{n-1}$.

8.2 Utiliser une formule pour la somme des termes d'une suite géométrique.

9 Que vaut le flottant suivant, à 64 chiffres ? On rappelle que l'exposant est codé sur 11 bits, avec un décalage de E_{max} =1023, et que la mantisse est codée sur 53 bits

0100010001101001001111000001110000000000000000000000000000000000

Les différents format sont : simple précision (32 bits), double précision (64 bits) et même quadruple précision.

On commence toujours par le bit de signe, ensuite il y a les bits d'exposant puis de mantisse, avec les conventions suivantes (voir cours) :

En simple précision, on a 8 bits d'exposant, avec E_{max}=127, et 23 bits +1 caché de mantisse.

En double précision, on a 11 bits d'exposant, avec E_{max}=1023, et 52 bits +1 caché de mantisse.

En général on vous rappellera ces détails, mais pas la formule pour calculer la mantisse.

On constate 64 bits, on est donc en double précision : exposant : 11bits, Emax=1023

Réécrivons en tenant compte du format : « 1bit de signe l 11 bits d'exposant l 53 bits (dont 1 caché) » :

0 10001000110 1001001111000001110000000000000000000000000000000000

Le premier bit est le signe, les 11 suivants codent l'exposant e, et les suivants la mantisse m (avec un bit caché).

s=0 donc signe positif
e= $2^1+2^2+2^6+2^{10}$=1094 donc e – Emax = 1094-1023 = 71
m= $1 + 2^{-1}+ 2^{-4} +2^{-7} +2^{-8} +2^{-9} +2^{-10}+2^{-16} +2^{-17} +2^{-18}$ = 1,577

Finalement N = $1.577 \times 2^{71}=3{,}724.10^{21}$

10 Représentation des nombres

10.1 Equation de la forme $ax^2 + \frac{1}{a}x - a = 0$ d'où $x_1 = \frac{-1-\sqrt{1+4x^4}}{2x^2}$ et $x_2 = \frac{-1+\sqrt{1+4x^4}}{2x^2}$
Numériquement : $x_1 = -2.04 \times 10^8$ et $x_2 = 0.0$ ou 1.13×10^{-8} selon la précision de la machine utilisée.
1er problème : la valeur affichée pour x_2 dépend de la précision de la machine utilisée.
2ème problème : Les résultats sont faux car on doit avoir nécessairement $x_1 x_2 = -1$ d'après l'équation proposée.
Le principal problème vient de la soustraction effectuée dans x_2 au numérateur : Cette soustraction entre des termes d'ordre de grandeur très proche (car $1 \approx \sqrt{1+4x^4}$) entraîne une erreur de cancellation. Ce problème ne se pose pas dans x_1 qui contient une somme au numérateur.
10.2 Une méthode consisterait donc à éviter d'évaluer la racine dont le calcul nécessite une soustraction. On peut par exemple, pour lever l'erreur, exprimer x_2 par $x_2 = -\frac{1}{x_1} = 4.9 \times 10^{-9}$