Documenter le calcul du log base 2

Processus : Concevoir, Développer et Industrialiser

000_000_00Y-00

HENRY Erick

Documenter le calcul du log base 2

Descriptif du document.

Processus propriétaire.

Concevoir, Développer et Industrialiser.

Objet du document.

Ce document décrit la méthode de calcul du log base 2 d'un nombre. Cette méthode a pour objectif d'éviter l'emploi des bibliothèques standard qui calcule le log dans le monde des flottants et donc utilise un code complexe.
Elle calcule ce log dans le monde des entiers.

Domaine d'application.

Ce document a été fait pour un PIC24 programmé en C.

Sommaire

Certificat arrivé à expiration

X 

Erick HENRY
-
Signé par : erick.henry@safrangroup.com .[1]

[1] Nota : Un trait vertical dans la marge à gauche ou un surlignage indique une mise à jour du texte par rapport à la version précédente.

Principe

En mathématiques, le logarithme binaire ($\log_2 n$) est le logarithme de base 2. Il sera noté « lb ».
C'est la fonction réciproque de la fonction puissance de deux : $x \mapsto 2^x$. Le logarithme binaire de x est la puissance à laquelle le nombre 2 doit être élevé pour obtenir la valeur x, soit : $lb(x) = a \Leftrightarrow x = 2^a$

Ainsi, le logarithme binaire de 1 est 0, le logarithme binaire de 2 est 1, le logarithme binaire de 4 est 2, le logarithme binaire de 8 est 3.

Soit (e; m) la représentation virgule flottante binaire d'un nombre entier non nul x, où e est un entier porteur de l'ordre de grandeur, et m un significande tel que $1 \leq m < 2$.

On a :

$x = m \times 2^e$ entraîne $lb(x) = e + lb(m)$.

Ainsi le calcul de lb(x) se ramène ainsi au domaine [1, 2[

Par exemple, $10 = 2^3 \times 1,25 \rightarrow lb(10)= 3 + lb(1,25)$ où lb(1,25) est la partie fractionnaire du logarithme cherché.

Chaque bit de lb(1,25) peut se calculer directement bit à bit à l'aide des relations :

- $lb(x) = lb(x^2)/2$

- $lb(x/2) = lb(x) - lb(2)$
- $lb(2) = 1$
  - $lb(x)= lb(x/2) + 1$

Quand on cherche un nouveau bit de x ($1 \leq x < 2$) :

1. On élève x au carré
2. Si x est ≥ 2,
   - On note 1, on divise x par 2 et on poursuit

3. Sinon, on note 0 et on poursuit.

Par exemple :

lb(10)	$= 11_2 + lb(1,25)$	
	$= 11_2 + lb(1,5625)/2$	$x < 2$ donc 0
	$= 11.0_2 + lb(2,44140625)/4$	$x \geq 2$ donc 1 et x / 2
	$= 11.01_2 + lb(1,220703125)/4$	
	$= 11,01_2 + lb(1,490116119)/8$	$x < 2$ donc 0
	$= 11.010_2 + lb(2,220446049)/16$	$x \geq 2$ donc 1 et x / 2
	$= 11.0101_2 + lb(1,110223025)/16$	
	$= 11.0101_2 + lb(1,232611)/32$	$x < 2$ donc 0
	$= 11.01010_2 + lb(1,519330)/64$	$x < 2$ donc 0
	$= 11.010100_2 + lb(2,308)/128$	$x \geq 2$ donc 1 et x / 2 …

Or $11.0101001_2 = 3.3203125$ et on a déjà $2^{3.3203125} = 9.988807$

Calcul de e (e est un entier porteur de l'ordre de grandeur)

Déclaration des variables :

- une variable de même type que x (ici x_temp)
- une variable 8 bits pour stocker e initialisée à 0

Logigramme :

- Copier x dans x_temp

- Boucler tant que x_temp déculé d'un bit à droite est différent zéro : while (x_temp >>= 1)
  - Incrémenter e : e++

Fin :

- e contient tel que lb(x) = e+lb(m)

Exemple :
- x =1
  - e = 0

- x =2
  - e = 1

- x =10 (décimal)
  - e = 3

- x =0 (valeur interdite)
  - e = 0

Calcul de m (m est un significande tel que 1 ≤ m < 2)

Rappel : On calcule d'abord e après ce calcul l'objectif est d'avoir le significante.

Dans notre cas, pour notre précision, on récupère ceci dans un entier 8 bits. On fait ceci en décalant les bits de x.
Remarques : Pour la suite, il est intéressant de mettre ce nombre dans un entier 16bits.

Déclaration des variables :
- une variable 8 bits pour stocker m initialisée à 0

Logigramme :
- Tester si e est > 7
  - m est égal à la valeur de x décalé à droite de (e-7) bits
- Tester si e est < 7
  - m est égal à la valeur de x décalé à gauche de (7-e) bits
- Tester si e est = 7
  - m est égal à la valeur de x

Fin :
- m contient tel que lb(x) = e+lb(m)

Exemple :
- x =1
  - m = 128 (décimal)

- x =2
  - m = 128 (décimal)

- x =10 (décimal)
  - m = 160 (décimal) / 1010 0000(binaire)

- x =0 (valeur interdite)
  - m = 0

Calcul de lb(m) (m est un significande tel que 1 ≤ m < 2)

Hypothèse : On récupère m entier 8 bits dans un entier 16 bits.

Déclaration des variables :
- un entier pour stocker le résultat lb_m
  - la partie après la virgule du lb de x

Logigramme :

- On boucle 8 fois
    - On décale à gauche de 1 bit le résultat : lb_m
    - On élève au carré m et on place le résultat dans m
    - Si $m > 2^{15}$
        - Si oui
            - On incrémente le résultat : lb_m
            - On décale à droite de 8 bit : m (7 bits normaux + 1 pour la division par 2)
        - Si non
            - On décale à droite de 7 bit : m
    - Fin de la boucle

- Le résultat de lb de x est sur 16 bits :
    - Les 8 bits de poids forts : la partie entière
    - Les 8 bits de poids faibles : partie après la virgule

  Lb(x) = e décalé à gauche de 8 bits + lb_m;

Exemple :

- x =2
    - e = 1
    - m = 1000 0000 (binaire)

- Première boucle
    - lb_m = 0
    - $m^2 \rightarrow m : 2^{14}$
    - test : $m < 2^{15}$
        - $m = 2^7$
    - Fin de la boucle
- On boucle sur les mêmes valeurs en conséquence :
    - lb_m = 0

- x =10 (décimal)
    - $e = 11_2$
    - m = 1010 0000 (binaire)

- Première boucle
    - lb_m = 0
    - $m^2 \rightarrow$ m : 0x6400 (hexa)
    - test : $m < 2^{15}$
        - m = 0xC8
    - Fin de la boucle
- Deuxième boucle
    - lb_m = 0
    - $m^2 \rightarrow$ m : 0x9C40 (hexa)
    - test : $m \geq 2^{15}$
        - lb_m = 1
        - m = 0x9C
    - Fin de la boucle
- Troisième boucle
    - $lb\_m = 10_2$
    - $m^2 \rightarrow$ m : 0x5F10 (hexa)
    - test : $m < 2^{15}$
        - m = 0xBE
    - Fin de la boucle
- Quatrième boucle
    - $lb\_m = 100_2$
    - $m^2 \rightarrow$ m : 0x8D04 (hexa)
    - test : $m \geq 2^{15}$
        - $lb\_m = 101_2$
        - m = 0x8D

  - Fin de la boucle
- Cinquième boucle
  - lb_m = 1010_2
  - $m^2 \rightarrow$ m : 0x4DA9 (hexa)
  - test : $m < 2^{15}$
    - m = 0x9B
  - Fin de la boucle
- Sixième boucle
  - lb_m = 10100_2
  - $m^2 \rightarrow$ m : 0x5DD9 (hexa)
  - test : $m < 2^{15}$
    - m = 0xBB
  - Fin de la boucle
- Septième boucle
  - lb_m = 101000_2
  - $m^2 \rightarrow$ m : 0x8899 (hexa)
  - test : $m \geq 2^{15}$
    - lb_m = 101001_2
    - m = 0x88
  - Fin de la boucle
- huitième boucle
  - lb_m = 1010010_2
  - $m^2 \rightarrow$ m : 0x4840 (hexa)
  - test : $m < 2^{15}$
    - m = 0x90
  - Fin de la boucle
- Résultat :
  - e = $0000\ 0011_2$
  - lb_m = $0101\ 0010_2$

Attachments

Abbreviations

- Sans :
  Sans

Definitions of terms

- Logarithme base 2 :
  C’est aussi appelé logarithme binaire.
- Significande :
  Synonyme de mantisse mais l'usage du terme mantisse est cependant découragé dans ce sens par la norme internationale IEEE 754 pour la représentation des nombres à virgule flottante, qui recommande plutôt l'usage en anglais du terme significand, que l'on peut traduire en français par « significande2 ». Le risque de confusion est en effet évident, puisque ici c'est plutôt log(a) qui est la mantisse de log|x|.

Link

- Logarithme binaire (Wikipédia)
- Dean K.J., 1969, A fresh approach to binary logarithm computing, Electronic Engineering.

Revision

Index: 00

Indice actuel.