Représentation des informations :

1. Petite introduction :

Les informations traitées par l'ordinateur sont de différents types (texte , images , sons ..) , mais elles sont toujours représentées sous forme binaire (seulement avec 0 et 1) ..

* codage : une fonction établissant une correspondance entre la représentation externe de l'information avec l'information interne ..

Ex ..

15 a comme représentation interne la suite de bits 1111

Bit : 0 , 1
On dit que le nombre 1010 contient 4 bits

2 - système de numération : (dans ce chapitre , on vas s'intéresser à la représentation des données numérique , où vous allez recontrer التعدادx))

* les entiers positifs :

Dans le décimal (النظام العشري) , chaque entier positif ou nul se compose des chiffres { 0,1,2 . . 9 } .

Dans un système de base b (b > 1) , chaque entier positif se compose des chiffres { 0 , 1 .. (b-1) } , par ex ! dans le système binaire chaque nombre se compose seulement de 0 et de 1 .

> la forme polynomiale : tout nombre positif N peut etre représenté par une expression de la forme :

$$N = a_n * b^n + a_{n-1} * b^{n-1} + ... + a_1 b^1 + a_0 b_0 \text{ ... } (\Omega)$$

Octet : 8 bits

A_i appartient à { 0,1, .. , (b-1)} et $a_n \neq 0$.

L'expression (Ω) est équivalente à ($a_n a_{n-1} .. a_1 a_0$)b .

Par ex :

$31=1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = (10101)_2$

$35=1*3^3 +0*3^2 + 2*3^1 + 2*3^0 = (1022)_3$

> le système binaire : toute information est codée seulement avec 0 et 1 .

Et sur n bit on peut représenter 2^n combinaison (قائمة- الاحتمالاتx))

Sur 2 bit on peut représenter 4 combinaison qui sont : 00 , 01 , 10 , 11

> aller d'une base vers une autre :

- base quelconque > décimal : on utilise la forme polynomiale

Prenons des exemples :

$(1111)_2 = 1*2^3 + 1 *2^2 + 1*2^1 + 1*2^0 = 15$

$(2103)_4 = 2*4^3 + 1*4^2 + 0*4^1 + 3*4^0 =$

- décimal > base quelconque : la division successive

Prenons de exemple !

25 /2 = 12 reste 1

12 /2 = 6 reste 0

6 / 2 = 3 reste 0

3 / 2 = 1 reste 1

1 / 2 = 0 reste 1

L'octal : base 8
L'hexadécimal : base 16

25 = $(11001)_2$ (on lit de bas en haut)

- base b > base b^n : par décomposition ! Prenons des exemples et vous aller comprendre :3 ..

On estime de faire la conversion du binaire vers l'octal d'un entier positif N , au lieu de convertir N en décimal puis en octal (on faisait comme ça au lycée) , on

Peut facilement , découper le nombre à convertir en blocs de 3 bits (car 8 = 23)

Par ex : $(1010011101)_2 = (1235)_8$

001 010 011 101

1 2 3 5

Pour aller du binaire vers l'hexadécimal , on découpe le nombre à convertir en blocs de 4 bits

Par ex :

$(1010011101)_2 = (25D)_{16}$

0010 1001 1101

2 5 D

Lorsque b>10 , on représente les chiffres de cette base avec des symboles !
10 = A
11 = B
12 = C
13 = D
Et ainsi de suite :'3

* le plus grand nombre qu'on peut représenter sur n bit est : $2^n - 1$

Par ex : sur 4 bits , le plus grand nombre qu'on peut représenter est 1111

C'est $2^{(4-1)} - 1$

($(111111....1)_2 = 2^n - 1$:3 penser à مجموع حدود متتابعة لمتتالية هندسية)

N fois 1

* sur n bit (registre de longeur n) , et pour une base b , seuls les nombres entiers positifs N tel que : $0 \leq N \leq b^n - 1$ (on dit que la capacité du registre est b^n-1)

* le bit du poids fort est celui de gauche , le bit du poids faible est celui de droite

* over-flow { dépassement } : Lorsque par exemple le résultat d'une opération sur des nombres produit un nombre plus grand que la taille du mot prévu pour le représenter

Par ex : sur 4 bit : $(1001)_2 + (1011)_2 = (10100)_2$ on peut pas représenter le résultat sur 3 bit

> Opérations arithmétiques en binaire :

addition : prenons des exemples !

	10111	23
+	10010	18
=	101001	41

	1111	15
+	1111	15
=	11110	30

0+0=0
0+1=1
1+1=0 , et une retenue
1+1+1=1 et une retenue

Soustraction :

	10111	23
-	10010	18
=	00101	5

	10101	21
-	01011	11
=	01010	10

0-0 = 0
1-0=1
0-1=1 et une retenue
1-1 = 0

Multiplication : c'est la plus facile :3 .. prenons un exemple !

```
      1100
    × 1010
----------
      0000
     1100 .
    0000 . .
   1100 . . .
----------
= 01111000
```

Remember :
If u want to be more than average , u have to work more than average people <33 ..

>> je vous conseille de faire bcp d'exercices sur ce chapitre , pour avoir de la réflexion et de la rapidité !! (la calculatrice est interdite dans qlq cis :3)

*** les entiers négatifs :** les entiers négatifs peuvent être codés selon 3 méthodes :

(S+VA) : signe et valeur absolue

(CR) : complément à 1

(CV) : complément à 2

> signe et valeur absolue : ±

On sacrifie un bit pour représenter le signe le bit du poids (n-1) , 0 pour le signe + et 1 pour le signe -

Par ex : 0110 = +6   1110 = -6

Les inconvénients de cette méthode :

- le zéro a deux représentation , 0...000 et 1...000 (soit –0 et +0)
- l'addition et la multiplication sont compliquées à cause de bit de signe .

> complément à 1 (CR) : pour obtenir le nombre négatif on inverse les bits de sa valeur absolue

Par ex : 0110=+6 , 1001=-6 (0 pour le signe + et 1 pour le signe -)

Les inconvénients de cette méthode :

- 2 représentations du zéro : 000..000 et 1111..11
- la retenue est ajoutée au résultat , prenons des exemples !!

0111	1111
+ 1001	+ 0101
10000	10100
1	1
= 0001	= 0101

> complément à 2 (CV) :

- on ajoute 1 au complément à 1 , CV = CR + 1

Par ex :

+6=0101 , -6=$(1010)_{CR}$ en ajoutant un 1 au CR , -6=$(1011)_{CV}$

- il y a une autre méthode plus rapide ! En partant du chiffre du poids faible , on réecrit les chiffres tels quels jusqu'au premier '1' inclus , après on écrit les compléments des chiffres restants

Par ex :

(01**1**0) au binaire pure s'écrit (**101**0) au complément à 2

(01110) au binaire pure s'écrit (10010) au complément à 2

- les avantages de cette méthode :
    - Une seule représentation du zéro;
    - la soustraction se réduit à l'addition de son complément;

- Pas de report du bit de retenue pour l'addition .

C à 1 : on ajoute la retenue au résultat
C à 2 : on ignore la retenue

> Intervalle de représentation des entiers N (sur n bit)

- S + VA : $-(2^{n-1}-1) \leq N \leq +(2^{n-1}-1)$
- C à 1 (CR): $-(2^{n-1}-1) \leq N \leq +(2^{n-1}-1)$
- C à 2 (CV): $-2^{n-1} \leq N \leq +(2^{n-1}-1)$ (une seule représentation pour le 0)

*** nombres réels (fractionnaires) :** les nombres fractionnaires sont les nombres qui comportent une partie inférieure à 1

On a deux méthodes pour représenter les nombres réels :

- virgule fixe
- virgule flottante

> virgule fixe :

N = partie entière , partie décimale

N = an-1 an-2 … a1 a0 , a-1 a-2 … a-p

a^{n-1} représente le bit de plus fort poids

a^{-p} représente le bit de plus faible poids

n nombre de chiffres de la partie entière

p nombre de chiffres de la partie fractionnaire

Prenons des exemples :

$(+0.001)_2 = 0.\ 0*2^{-1} + 0*2^{-2} + 1*2^{-3} = (+0.125)_{10}$

$(+0.125)_{10} = 0.\ 0*2^{-1} + 0*2^{-2} + 1*2^{-3} = (+0.001)_2$

$(+3.25)_{10} = (\ +11.01)_2$ car : $+3 = 11_2$ et $0.25=0*2^{-1} + 1*2^{-2}$

(je vous conseille encore une fois de faire bcp d'exercices sur ce chapitre , vous allez trouver plusieurs astuces non mentionnées en cour)

> virgule flottante : la représentation en virgule flottante consiste à représenter le nombre sous la forme suivante :

$$N = \pm M \times B^{E}$$

3:الكتابة العلمية اتفكروا

Avec : B : base (2 , 3 , ….)

M : mantisse (nombre purement fractionnaire)

E : exposant (un entier)

Par exemple :

$(+11.0101) = (+1.10101 \times 2^{1}) = (+1101.01 \times 2^{-2})$

* il y a plusieurs manières de représentation avec la virgule flottante , donc , il faut une normalisation ..

N=1,mantisse normalisée × 2

Exposant = excedent

Avec :

$eb = er + (2^{n-1} - 1)$

et eb : exposant baisé , er : exposant réel , n: le nombre de bit où l'exposant est représenté ..

Par exemple :

$(101.011)_2 = (\ 1.01011 \times 2^2\ )_2$ (et supposons que l'exposant est représenté sur 3 bit > bais = 3) , D'où : $(101.011)_2 = (1.01011 \times 2^{2+3})_2$

* **Norme IEEE754 :** Ce standard a défini au moins deux formats , l'une pour la simple précision , et l'autre pour la double précision

> simple précision : (sur 32 bit)

Signe : 0 si le nombre est positif , 1 si le nombre est négatif (sur 1 bit)

Exposant baisé : l'exposant en excédent 127 (l'exposant réel + le bais 127) (8 bit)

Mantisse normalisé : toujours sous la forme 1,bbb..bb (le 1 n'est pas écrit dans la configuration) (sur 23 bit)

signe	Exposant baisé	Mantisse normalisée

Par exemple :

$(101.011)_2 = (1.01011 \times 2^2)_2$

- Eb = 2 + 127 = 129 = (01000001)2

- Signe positif

- mantisse normalisé 01011000..00

D'où : $(101.011)_2$ = (0 01000001 01011000..00)IEEE754

Eb :	Mantisse :	Valeur :
0	0	0
0	≠ 0	Nombre dénormalisé
[1 , 254]	qlqc	Nombre normalisé
255	0	±∞
255	≠ 0	NOT A NUMBER

"I have no special talents. I am only passionately curious."
— **Albert Einstein**
- YOU'RE ENOUGH -

Exercice d'application : représenter sous le format IEEE754 , simple précision , le nombre (0.75)10

Solution:

$(0.75)_{10} = (0.11)_2 = (1.1 \times 2^{-1})_2$

Signe : positif

Exposant baisé : -1 +127 = 126 = $(01111110)_2$

Mantisse normalisée : 100...0000

D'où : $(0.75)_{10}$ = (0 01111110 100...0000)

> double précision : (sur 64 bits) - le même principe :3 -

Signe : 0 si le nombre est positif , 1 si le nombre est négatif (sur 1 bit)

Exposant baisé : l'exposant en excédent 1023 (l'exposant réel + le bais 1023) (sur 11 bit)

Mantisse normalisé : toujours sous la forme 1,bbb..bb (le 1 n'est pas écrit dans la configuration) (sur 52 bit)

* Le code BCD :

Chaque chiffre d'un nombre est codé individuellement en son équivalent binaire Sur 4 bit .

Par exemple : 21 = 0010 0001

> opérations en BCD : lorsque on effectue une addition , on la corrige en ajoutant un 6 (0110) si le résultat génère une retenue ou ci (configuration interdite n>9) .

Prenons un exemple :

  0011 1001   39
+ 0101 1000   58

1000 1 0001
     1 0110

1001 0111   97

Dans le code BCD chaque chiffre [0,9] est codé individuellement en son équivalent binaire sur 4 bit .. donc 1010 , 1011 , 1100, 1101 , 1110 , 1111 sont des configurations interdites

* **Excess 3 :** BCD + 3 à chaque chiffre

Par exemple : 21 = 0101 0011

\> opérations en Excess 3 : la correction est : si une retenue +3 sinon –3

Finalement , je vous conseille de voir les vidéos de MAHSEUR , pour bien comprendre le cour .

N'oubliez pas de faire

Les exercices corrigés

De Mr.BESSA (vous

Allez les trouver dans

Le dossier Exercice/TDs)