Chapitre 12 : Les entiers

I) La numérisation

a) Représentation des entiers

Pour écrire des nombres entiers, nous sommes habitués à utiliser une notation de position en base 10 en utilisant des chiffres compris entre 0 et 9. Exemple : $3507 = 3 \times 1000 + 5 \times 100 + 0 \times 10 + 7 \times 1$

Un entier naturel à $n+1$ chiffres peut donc s'écrire :

$(c_n c_{n-1} \dots c_1 c_0)_{10} = c_n 10^n + c_{n-1} 10^{n-1} + \dots + c_1 10^1 + c_0 * 10^0$

En suivant le même principe, nous pouvons envisager une notation positionnelle en base b, où $b \geq 2$ est un entier en utilisant des chiffres c_i de 0 à $b-1$: $(c_n c_{n-1} \dots c_1 c_0)_b = c_n b^n + c_{n-1} b^{n-1} + \dots + c_1 b^1 + c_0 b^0$

Si $b = 2$, on obtient le système binaire et les deux chiffres c_i (0 ou 1) sont appelés bits : $(c_n c_{n-1} \dots c_1 c_0)_b = c_n 2^n + c_{n-1} 2^{n-1} + \dots + c_1 2^1 + c_0 2^0$

c_0 est le poids du bit le plus faible (LSB) et c_n est le poids du bit le plus fort. Une suite composée de 0 et de 1 s'appelle un *mot* dont les *lettres* sont donc 0 ou 1 et *l'alphabet* est {0,1}

b) Pourquoi le binaire ?

C'est ce système binaire qui est utilisé car il est simple à obtenir sur les différentes couches physiques :

- Dans les microprocesseurs ou clé USB, ces deux états sont électriquement simples à réaliser avec l'état haut (5V) ou bas (0V) d'une porte logique (transistor MOS),
- Sur le disque dure, il s'agit de l'orientation du moment magnétique de petites cellules.
- Sur les CD, DVD, il s'agit de la présence ou de l'absence de gravures
- Sur les fibres optiques, il s'agit de jouer sur l'état de l'intensité lumineuse transmise.

II) Représentation des entiers naturels en binaire

a) Les entiers naturels ou non signés :

Avec n bits, on peut représenter 2^n entiers k compris entre 0 et $2^n - 1$ tels que : $k = \sum_{i=0}^{n-1} b_i 2^i$ *avec* $b_i \in \{0,1\}$.

L'écriture en base 10 est constituée des restes obtenus dans les divisions euclidiennes successives par 10 jusqu'à obtenir un quotient nul, par exemple : $5326 = 6 \times 10^0 + 2 \times 10^1 + 3 \times 10^2 + 5 \times 10^3$

Le principe est le même en base 2 :

11	2				
1	5	2			
	1	2	2		
		0	1	2	
			1	0	

$11 = (1011)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$ avec le bit de poids fort plus à gauche et le bit de poids faible à droite. En python, pour convertir en binaire un entier A sur python : $bin(A)$.

5326	10				
6	532	10			
	2	53	10		
		3	5	10	
			5		

b) Entiers relatifs ou signés

Pour décrire les entiers négatifs k avec n bits, on utilise le complément à 2 qui consiste à coder $k < 0$ de la manière suivante :

- On code $|k|$ sur les n bits
- On inverse les bits
- Et on ajoute 1

Par exemple, sur 3 bits : $-1 \rightarrow (001)_2 \rightarrow (110)_2 \rightarrow (111)_2$

Ce qui permet de représenter les nombres compris entre -2^{n-1} et $2^{n-1} - 1$, le MSB est 0 pour les nombres positifs ou nuls et 1 pour les nombres négatifs. Exemple sur 8 bits :

uint 8	int 8
$0 \leq k \leq 255$	$-128 \leq k \leq 127$

c) Entiers-multi-précision

Avec le langage python, le nombre de bits avec lequel on souhaite faire la conversion numérique est paramétrable.Par exemple en photo, on limite la numérisation d'un nombre à 8 bits en uint8 (les valeurs sont 256 périodiques et des risques de dépassement de capacités-overflow sont possibles) :

-256	…	….	-1	0	1…	….	…	255	256	…	…	511		
0			255	0				255	0			255		

- 255+1=0 en effet 1111 1111 + 1 = 0000 0000 avec la perte de bit de retenue
- 255+256=255
- 0-1 =255

Par défaut, on a un codage sous 64 bits, si le nombre à coder dépasse 2^64 alors on stocke sur d'autres "paquets de 64 bits.Le plus grand entiers numérisable est en fait limité par la capacité de la mémoire vive. Si 8Go de mémoire vive, on a donc 8 * 10^9 * 8 bits soit presque 10^10 bits et un maximum de l'ordre de 2^(10^10) ce qui donne en bas 10 : 2^(10^10) = 10^k soit k = 10^10 * log10(2) Soit un max de 10^(10^10)

Commenté [AM1]: Ou notation positionnelle

Commenté [AM5]: Cette astuce s'explique :
-l'idée est que 3+(-3)=0
- donc si $(3)_2 = (011) sur\ 3\ bits$
- alors $(-3)_2 = (101) sur\ 3\ bits$ permet d'avoir $(000)_2\ sur\ 3\ bits$

Commenté [AM2]: Pour les entiers en base 10, on écrit :
$x = c_0 10^0 + c_1 10^1 + \dots + c_n 10^n$
Pour les entiers en base 2, on écrit :
$x = c_0 2^0 + c_1 2^1 + \dots + c_n 2^n$
Pour les nombres décimaux en base 10:
$x = 5{,}1234. * 10^n$
En base 2 :
$x = 2{,}1012 * 2^n$

Commenté [AM3]: On a en effet 2*2*…*2 =2^n combinaisons possibles.
La valeur maximale est donnée par :
$2^0 + 2^1 + \dots + 2^{n-1} = 1 + q^1 + q^2 + \dots q^{n-1} = \frac{1-q^n}{1-q} = \frac{1-2^n}{1-2} = 2^n - 1$

Commenté [AM4]: La capacité de stockage des systèmes de mémoire s'exprime en octet et ses multiples en 2^n
1kio = 1024o = 2^10 o
1Mio= 2^20 o
1Gio = 2^30