Codage des nombres
et
Logique binaire

Le codage de l'information

- Dans la vie courante, on utilise le système décimal pour calculer.
- Pour pouvoir fonctionner, le processeur traite les informations contenues en mémoire de façon binaire: il ne sait traiter que des suites de 0 et de 1.

  ... aussi bien les nombres, que les instructions.
- Pour la représentation machine, il faut effectuer un changement de base pour les représenter en base 2 en mémoire.

Les bases de la numération

- Habitude de représenter les nombres en base 10 (convention pratique)
- La représentation en base 10 que nous utilisons n'est qu'une des multiples représentations possibles
  - *Ex de codage ancestral : traits, bâtons, cailloux !!*
- Notion de base : regroupement par paquets de même taille. La taille des paquets = base
- si la taille est 10 on parle de base 10

Quelques bases de numération

- Base 2 : logique booléenne à la base de l'électronique numérique
  - 2 symboles: le BIT (BInary digiT) {0,1}.
  - 8 bits constitue 1 octet.
  - 256 valeurs possibles (= 2^8) Ex : de 0 à 255
- Base 8 : système octal : {0,1,2,3,4,5,6,7}
- Base 16: système hexadécimal pour la représentation des mots machines: {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

La base 2

- On décompose les nombres en puissance de 2. Ex. de codage de 204

$2^7=128$	$2^6=64$	$2^5=32$	$2^4=16$	$2^3=8$	$2^2=4$	$2^1=2$	$2^0=1$	
128 +	64 +	32	16	8 +	4 =	2	1	
1	1	0	0	1	1	0	0	= 204

- Notation : par convention, lorsqu'il y a un risque de confusion sur la base utilisée, on écrit la valeur de la base en indice.
  - Exemple : 12_{10}(base 10),14_8 (base 8), 1100_2 (base 2).

Le codage octal et hexadécimal

- Les codages octal (base 8) et hexadécimal (base 16) sont aussi des codages très utilisés en informatique :
  - codages beaucoup plus compacts (plus courts),
  - les conversions entre bases 16, 8 et 2 sont très simples : il suffit de regrouper les bits par paquets de 4, ou 3 ou 1.
- Utilisé en informatique pour coder les couleurs, caractères
  - Exemple: code RGB (Red Green Blue) utilise 256 valeurs d'intensité pour chacune des 3 couleurs sur 6 positions
  - 000000 = noir; FFFFFF= blanc; FF0000 = rouge; 00FF00 = bleu; FF99A0 = rose etc.
- Notations :
  - $A_{16} = 10_{10}$; $B_{16} = 11_{10}$; $C_{16} = 12_{10}$; $D_{16} = 13_{10}$; $F_{16} = 15_{10}$;
  - $FF_{16} = 255_{10}$
  - Nombre noté généralement suivi de H (ou X): 0DH

Quelques bases de numération

Binaire	Octal	Hexa	Décimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7

En base 2, sur 4 bits

$0001 = 0 * 2^3 + 0*2^2 + 0*2^1 + 1*2^0$
$= 1$

$0111 = 0 * 2^3 + 1*2^2 + 1*2^1 + 1*2^0$
$= 7$

Quelques bases de numération

Binaire	Octal	Hexa	Décimal
1000	10	8	8
1001	11	9	9
1010	12	A	10
1011	13	B	11
1100	14	C	12
1101	15	D	13
1110	16	E	14
1111	17	F	15

En base 2, sur 4 bits

$1001 = 1 * 2^3 + 0*2^2 + 0*2^1 + 0*2^0$
$= 8$

$1111 = 1 * 2^3 + 1*2^2 + 1*2^1 + 1*2^0$
$= 15$

Arithmétique binaire

Addition en base 2

- Même principe qu'en base 10
- Retenue dès que le résultat dépasse 1

a	b	s=a+b	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Exemple :

$$\begin{array}{cccccc} & 1 & 1 & 1 & 1 & \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \quad + \\ \hline 1 & 1 & 0 & 0 & 0 & 0 \end{array}$$

Arithmétique binaire

Multiplication base 2

- Multiplication par 2^n : décalage de *n* positions vers la gauche :

  011101 x 000010 = 111010

  011101 x 000100 = 1110100

  Débordement : si la taille des mots mémoire est limitée, les bits sortant des limites sont perdus.

  Preuve par l'addition :

  011101

  011101 +

  111010

- Pour les autres nombres il faut passer par une décomposition en puissance de 2 :

  001100 x 000101 =

  001100 x (000100+000001) = 110000 + 001100

  = 111100

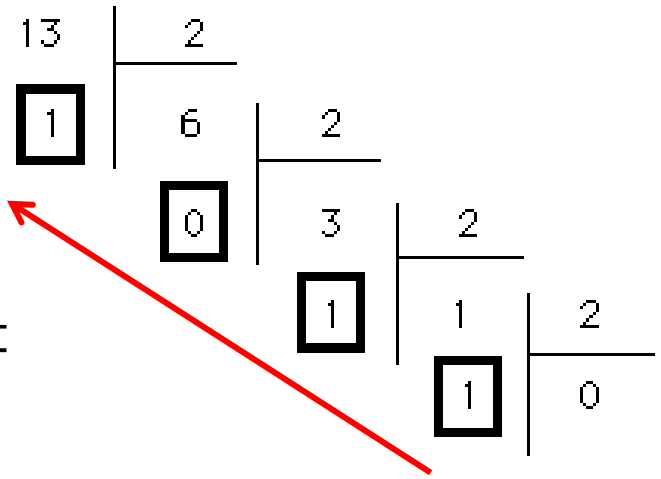
Le changement de base

Base 10 -> Base 2

- Pas de difficultés pour la conversion de binaire à décimal.

L'inverse est moins sûr ! Conversion de base 10 en base 2 par divisions successives par 2.

- Le premier reste obtenu est le chiffre de poids faible (le plus à droite) de la représentation en base 2, le dernier reste obtenu est le chiffre de poids fort (le plus à gauche). Les divisions s'arrêtent lorsque le dernier quotient est 0.



- Exemple : $13_{10} = 1101_2$.

Le changement de base

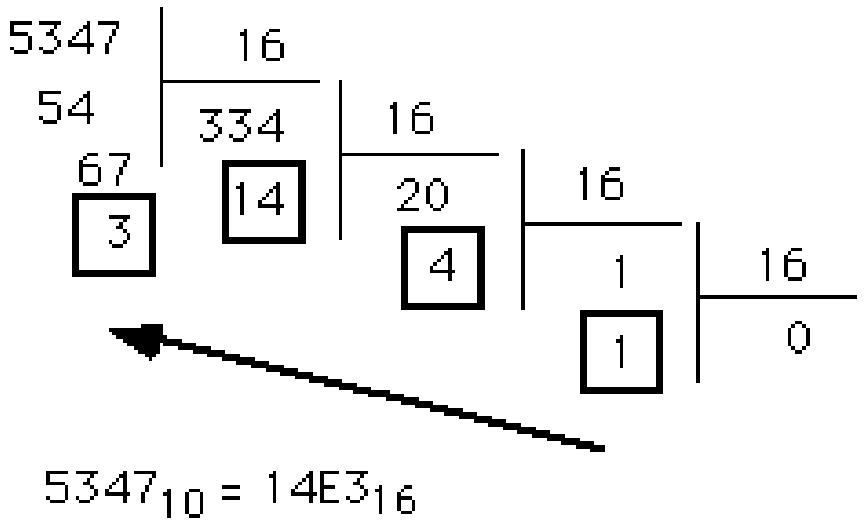
Base 16 -> Base 10 et Base 10 -> Base 16

- De L'hexadécimal au décimal :

  $837 = 8 * 10^2 + 3 * 10^1 + 7 * 10^0$

  $D5H = 13 * 16^1 + 5 * 16^0 = 13*16 + 5*1 = 208 + 5 = 213$

- Les conversions de base 16 en base 10 se font en utilisant le même principe que pour les conversions de base 2 vers base 10 : divisions successives par 16, dont les restes forment la représentation en base 16.



$5347_{10} = 14E3_{16}$

- Exemple : Conversion de 5347_{10} en base 16.

Le changement de base

- Base 2 -> Base 16
  - $1111_2 = F_{16}$
  - Il est facile de faire la conversion, en séparant le nombre binaire en 2 parties
  - $1101\ 0101_2$ = D5 H
    D 5

Représentation en mémoire

- Le nombre de bits disponible pour cette représentation est limitée par la taille des mots mémoire que l'UC est capable de traiter (16 bits, 32 bits ou 64 bits).
- Même s'il est possible d'utiliser plusieurs mots mémoire pour représenter une information donnée, il faut fixer au départ le nombre de mots réservé à la représentation de chaque type d'information (donner un type précis).
- Une des conséquences de cette restriction est qu'on ne peut pas représenter n'importe quel nombre qu'il soit entier ou réel.

Représentation des entiers

- Les entiers regroupent les types *byte*, *shortint*, *integer*, *longInt* et *word*. Le type de codage employé est le même, au nombre d'octets utilisés près. Plus précisément (machine 32 bits) :

BYTE	non signé 8 bits	[0..255]
SHORTINT	signé 8 bits	[-128..127]
INTEGER	signé 16 bits	[-32768..32767]
LONGINT	signé 32 bits	$[-2^{31}..2^{31}-1]$
WORD	non signé 16 bits	[0..65535]

Représentation des entiers non signés

- En réponse à la question : "quel est le plus grand nombre entier représentable sur b bits ?"
- Le nombre de valeurs distinctes représentables sur b bits est exactement 2^b
- Le plus grand entier non signé représentable sur b bits est $2^b - 1$
- Exemple : b = 4 le plus grand nombre représentable est $1111_2 = 15_{10}$

Représentation des entiers signés

- Le bit de signe:
  - 1ère solution : rajouter un bit de signe. Si le premier bit est 0, on a un entier positif, si c'est 1, on a un entier négatif!
  - Problème d'une arithmétique compliquée : $a + (-b)$ ne se calcule pas comme $a - b$.
- Hypothèse: la représentation du signe ne doit pas compliquer la mise en oeuvre des opérations arithmétiques
- Le plus satisfaisant : codage en complément à 2, codage quasiment universellement utilisé.
  - le bit le plus à gauche représente le signe (0 pour +, 1 pour - et l'entier 0 est considéré comme un entier positif)
  - la représentation de l'opposé d'un entier signé n est obtenue en prenant le complément à 2 de sa représentation binaire

Représentation des entiers signés

- Le codage en complément à 2 obéit au principe suivant :
  - les entiers >0 sont représentés en base 2 sur les (b - 1) bits de droite, le bit le plus à gauche est mis à 0.
  - Pour les entiers < 0 notés n^-, on code en réalité la valeur $[2^b - n^-]$: le complément à 2 de n^-

- Exemple: codage avec b=3

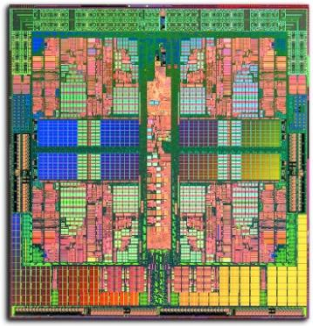
Code	Valeur
000	+0
001	+1
010	+2
011	+3
100	-4
101	-3
110	-2
111	-1

Représentation des entiers signés

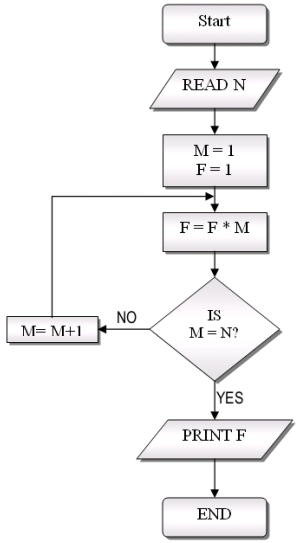
- Prendre le complément à 2 d'un nombre n^- sur b bits :
  - Calculer la représentation binaire de n^- sur b bits
  - Inverser chacun des b bits de la représentation binaire de n^-
  - Ajouter 1 au résultat
- ATTENTION : si n^- est trop grand pour être représenté sur b bits, le résultat obtenu n'est pas nécessairement la représentation de $-n^-$!
- Réaliser le codage de -17 sur 8 bits
  - 0001 0001 (étape 1)
  - 1110 1110 (étape 2)
  - 1110 1111 (étape 3)

Logique binaire

- Calculer avec des variables pouvant prendre 2 valeurs : VRAI ou FAUX
- Fonctions logiques omniprésentes dans les ordinateurs
  - Au niveau matériel
    - Un processeur est composé de milliards de portes logiques



  - Au niveau logiciel
    - Les programmes informatiques contiennent des tests logiques



Logique binaire

- Algèbre de Boole
- Deux Eléments :
  - Vrai (aussi représenté par 1)
  - Faux (aussi représenté par 0)
- Quatres opérateurs logiques :
  - ET : noté '.' ou '&'
  - OU : noté '+' ou '|'
  - OU exclusif (XOR): noté '$\oplus$' ou '^'
  - NON : noté '$\overline{a}$' ou '!'

ET	F	V
F	F	F
V	F	V

OU	F	V
F	F	V
V	V	V

XOR	F	V
F	F	V
V	V	F

NON	F	V
	V	F

Logique binaire

- Propriétés des opérateurs
  - Associativité :
    - (a+b)+c = a+(b+c)=a+b+c
    - (a.b).c = a.(b.c) = a.b.c
  - Commutativité:
    - a+b = b+a
    - a.b = b.a
  - Distributivité
    - a.(b+c) = a.b + a.c

Logique binaire

- Expression booléennes complexes
  - Composer et chainer les opérateurs logique
- Exemple:
  - La proposition suivante détermine si je vais à la plage « *si la température extérieure est supérieure à 25C ou si il ne pleut pas et que l'eau est a plus de 20C*»
  - En adoptant les conventions suivantes ....
    a : la température extérieur > 25 C
    b : il pleut
    c : la température de l'eau > 20 C
  - ... la proposition peut alors s'écrire

$$a + (\overline{b}.c)$$