MATHÉMATIQUES

APPLIQUÉES À L'INFORMATIQUE

Avant de commencer

- Rappel des règles …

- Florian BERLIAT
- Contact : florian.berliat13@ynov.com

Présentation

Avant de commencer

Programme

- Arithmétique
  - Notion de base
  - Manipulations et calculs
- Logique Mathématique
  - Algèbre de boole et calcul propositionnel
- Les matrices
  - Le calcul matriciel et les systèmes linéaires
  - Les matrices particulières
  - Les opérations sur les matrices
  - Les systèmes linéaires
- Les ensembles
  - Caractérisation de la notion d'ensemble
  - Les relations entre les ensembles
  - Les outils d'analyse et de dénombrement
- Statistiques et probabilités
  - Probabilités conditionnelles ou non conditionnelles

Arithmétique

NOTION DE BASE

Arithmétique

Notions

Il n'y a que 10 types de gens dans le monde : ceux qui connaissent le binaire et les autres …

Arithmétique
Notions

- Les informations traitées par les ordinateurs sont de différentes natures :
  - Nombres, texte, images, sons, vidéos, programmes…
- Dans un ordinateur, elles sont toujours représentées sous forme binaire(BIT = Binary digIT)
  - Une suite de 0 et de 1

00110110 11010101 10100011 00111001 10011000 10101100
00100101 01011010 01110110 11011010 10110111 10101101

Arithmétique
Notions

- Définition : le codage de l'information
  - Le codage de l’information permet d’établir une correspondance qui permet sans ambiguïté de passer d’une représentation (dite externe) d’une information à une autre représentation (dite interne : sous forme binaire) de la même information, suivant un ensemble de règles précises.
- Exemple :
  - Le nombre 35 : 35 est la représentation externe du nombre trente cinq
  - La représentation interne de 35 sera une suite de 0 et de 1 : 00100011

Arithmétique
Notions

- En informatique, le codage de l'information s'effectue principalement en trois étapes:
  - L'information sera exprimée par une suite de nombres (Numérisation)
  - Chaque nombre est codé sous forme binaire (suite de 0 et 1)
  - Chaque élément binaire est représenté par un état physique
- État physique ??
  - Charge électrique (RAM) :
    - chargé (bit à 1)
    - non chargé (bit à 0)
  - Magnétisation (DD, disquette) :
    - polarisation Nord (bit à 1)
    - polarisation Sud (bit à 0)
  - Fréquences (Modem) dans un signal sinusoïdal :
    - Fréquence f1 (bit à 1) : $s(t) = a \sin(2\pi f_1 \, t + \Psi)$
    - Fréquence f2 (bit à 0) : $s(t) = a \sin(2\pi f_2 \, t + \Psi)$

<u>Remarque :</u> ne pas confondre nombre et chiffre

Arithmétique
Notions

- Un système de numération décrit la façon avec laquelle les nombres sont représentés
- Un système de numération est défini par :
  - Un alphabet *A* : les signes ou symboles disponibles pour la représentation des nombres.
  - Des règles d'écritures des nombres : juxtaposition de symboles. Elles définissent comment un nombre est construit à partir des symboles de l’alphabet.

Arithmétique
Notions

- Exemple 1 : la Numération Romaine

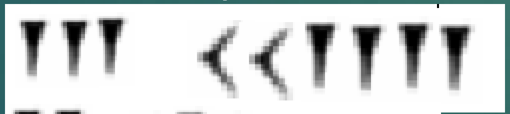
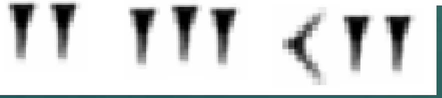
Système romain	I	V	X	L	C	D	M
Valeur décimal	1	5	10	50	100	500	1000

- Lorsqu'un symbole est placé à la droite d'un symbole plus fort ou égal à lui, sa valeur s'ajoute : CCLXXI = 271 ou plus simple VI = 6
- Un symbole placé immédiatement à la gauche d'un symbole plus fort que lui, indique que le nombre qui lui correspond doit être retranché au nombre qui suit : CCXLIII = 243 ou plus simple IV = 4
- On ne place jamais 4 symboles identiques à la suite
- Le plus grand nombre exprimable est 3999 (MMMCMXCIX)

***Remarque :* Système non adapté au calcul**

Arithmétique
Notions

- Exemple 2 : la Numération Babylonienne
- Chez les Babyloniens (environ 2000 ans avant J.C.), les symboles utilisés sont le clou pour l'unité et le chevron pour les dizaines.
- C'est un système de position.

- A partir de 60, la position des symboles entre en jeu :
  - 204 :  (3*60)+(24)
  - 7392 :  (2*3600)+(3*60)+(12) (3600=60*60)
- Le nombre 60 constitue la base du système

Arithmétique
Notions

- Exemple 3 : la Numération décimale
  - C'est le système de numération le plus pratique actuellement.
  - L'alphabet est compose de dix chiffres : $A = \{0,1,2,3,4,5,6,7,8,9\}$
  - Le nombre 10 est la base de cette numération
  - Pourquoi décimale (10) ? Car on a tous dix doigts, pardi !! Ou presque…
  - C'est un système positionnel. Chaque position possède un poids
  - Par exemple, le nombre 4134 s'écrit comme :

$$4134 = 4 \times 10^3 + 1 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

$$4000 \quad + \quad 100 \quad + \quad 30 \quad + \quad 4$$

Arithmétique
Notions

- Un système de numération positionnel à base b est défini sur un alphabet de b chiffres :

$$A = \{c_0, c_1, \ldots, c_{b-1}\} \text{ avec } 0 \le c_i < b$$

  - Soit $N = a_{n-1}\ a_{n-2} \ldots a_1\ a_0\ _{(b)}$ la représentation en base b sur n chiffres
    - a_i : est un chiffre de l'alphabet de poids i (position i).
    - a_0 : chiffre de poids 0 appelé le chiffre de ***poids faible***
    - a_{n-1} : chiffre de poids n-1 appelé le chiffre de ***poids fort***
  - La valeur de N en base 10 est donnée par :

$$N = a_{n-1}.\,b^{n-1} + a_{n-2}.\,b^{n-2} + \cdots + a_0.\,b^0_{(10)} = \sum_{i=0}^{n-1} a_i b^i$$

Arithmétique

Notions : le système binaire

- Un mot binaire de « n » bits s'écrit avec des éléments binaires prenant pour valeur 0 ou 1
- On appelle **LSB (Least Significant Bit)** le bit de poids le plus faible.
- On appelle **MSB (Most Significant Bit)** le bit de poids le plus fort.
- Exemple pour un mot de 8 bits

1 0 1 0 1 0 0 **1**

MSB LSB

$1\,0\,1\,0\,1\,0\,0\,1 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$128 + 32 + 8 + 1 = 169$

Arithmétique
Notions

- Bases de numération : Binaire, Octale, Hexadécimale
  - Système binaire (b=2) utilise deux chiffres : {0,1}
    - C'est avec ce système que fonctionnent les ordinateurs
  - Système Octale (b=8) utilise huit chiffres :{0,1,2,3,4,5,6,7}
    - Utilisé il y a un certain temps en Informatique
    - Il permet de coder 3 bits par un seul symbole
  - Système Hexadécimale (b=16) utilise 16 chiffres : {0,1,2,3,4,5,6,7,8,9,A=$10_{(10)}$,B=$11_{(10)}$,C=$12_{(10)}$,D=$13_{(10)}$,E=$14_{(10)}$,F=$15_{(10)}$}
    - Cette base est très utilisée dans le monde de la micro informatique
    - Il permet de coder 4 bits par un seul symbole.

Arithmétique
Notions

- Le transcodage (ou conversion de base) est l'opération qui permet de passer de la représentation d'un nombre exprimé dans une base à la représentation du même nombre, mais exprimé dans une autre base.

Arithmétique

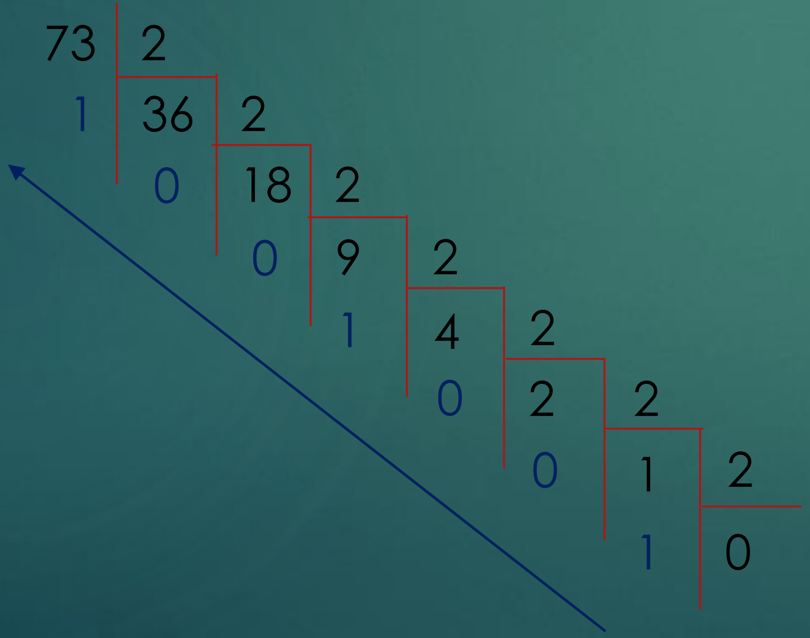
Notions : conversion de la base 10 vers une base b

- La règle à suivre est celle des divisions successives :
  - On divise le nombre par la base b
  - Puis le quotient par la base b
  - Ainsi de suite jusqu'à l'obtention d'un quotient nul
  - La suite des restes correspond aux symboles de la base visée.
  - On obtient en premier le chiffre de poids faible et en dernier le chiffre de poids fort

Arithmétique

Transcodage: conversion décimale vers binaire

- Soit N le nombre d'étudiants d'une classe représenté en base décimale par : $N = 73_{(10)}$
- Quelle est sa représentation binaire? En utilisant l'algorithme précédent :



$\mathbf{73_{(10)} = 1001001_{(2)}}$

Arithmétique

Transcodage: conversion décimale vers octale

- Soit N le nombre d’étudiants d’une classe représenté en base décimale par : $N = 73_{(10)}$
- Quelle est sa représentation octale? En utilisant l'algorithme précédent :

$$
\begin{array}{r|l}
73 & 8 \\
\hline
1 & 9 \;\big|\; 8 \\
 & 1 \;\;\; 1 \;\big|\; 8 \\
 & \quad\;\; 1 \;\;\; 0
\end{array}
$$

$$\mathbf{73_{(10)} = 111_{(8)}}$$

Arithmétique

Transcodage: conversion décimale vers hexadécimale

- Soit N le nombre d’étudiants d’une classe représenté en base décimale par : N = $73_{(10)}$
- Quelle est sa représentation hexadécimale? En utilisant l'algorithme précédent :

73 | 16
9 | 4 | 16
4 | 0

$73_{(10)} = 49_{(16)}$

Arithmétique

Exercices

- Donner l'écriture en base 2 des nombres suivants :

  $M = 19_{(10)}$ $N = 31_{(10)}$ $O = 256_{(10)}$ $P = 729_{(10)}$

  0001 0011 0001 1111 0001 0000 0000 0010 1101 1001

- Donner l'écriture en base 8 des nombres suivants :

  $Q = 18_{(10)}$ $R = 76_{(10)}$ $S = 729_{(10)}$

  22 114 1 331

- Donner l'écriture en base 16 des nombres suivants :

  $T = 70_{(10)}$ $U = 471_{(10)}$ $V = 718_{(10)}$ $W = 51727_{(10)}$

  46 1D7 2CE CA0F

Arithmétique

Notions : conversion de la base 2 vers une base b

- Il existe deux solutions selon les cas :
- Solution1 :valable dans tous les cas
  - Convertir le nombre en base binaire vers la base décimale, puis convertir ce nombre en base 10 vers la base b
- Solution 2 : 2, 8, 16
  - Binaire vers décimale : par définition $\sum_{i=0}^{n-1} a_i b^i$
  - Binaire vers octale : regroupement des bits en des sous-ensembles de trois bits puis remplacer chaque groupe par le symbole correspondant dans la base 8
  - Binaire vers Hexadécimale : regroupement des bits en des sous ensembles de quatre bits puis remplacer chaque groupe par le symbole correspondant dans la base 16

Arithmétique

Transcodage: conversion binaire vers décimale

- Soit N un nombre représenté en base binaire par : N = $1010011101_{(2)}$
- Représentation décimale ?
- Solution :

N = 1 0 1 0 0 1 1 1 0 1

$N = 1 \times 2^9 + 0 \times 2^8 + 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 512 + 0 + 128 + 0 + 0 + 16 + 8 + 4 + 0 + 1$

$= 669_{(10)}$

$\mathbf{1010011101_{(2)} = 669_{(10)}}$

Arithmétique

Notions : correspondance binaire/octale

Suite binaire	Symbole octale
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Arithmétique

Notions : correspondance binaire/hexadécimale

Suite binaire	Symbole Hexadécimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Arithmétique

Transcodage: conversion binaire vers octale

- Soit N un nombre représenté en base binaire par : $N = 1010011101_{(2)}$
- Représentation Octale ?
- Solution :

$N = 001 \quad 010 \quad 011 \quad 101$

$= \quad 1 \quad 2 \quad 3 \quad 5$

$= 1235_{(8)}$

$$\mathbf{1010011101_{(2)} = 1235_{(8)}}$$

Arithmétique

Transcodage: conversion binaire vers hexadécimale

- Soit N un nombre représenté en base binaire par : $N = 1010011101_{(2)}$
- Représentation Hexadécimale ?
- Solution :

$$N = 0010 \quad 1001 \quad 1101$$
$$= \quad 2 \quad 9 \quad D$$
$$= 29D_{(16)}$$

$$\mathbf{1010011101_{(2)} = 29D_{(16)}}$$

Arithmétique

Exercices

- Donner l'écriture des nombres suivants en base 2, puis en base 8 et 16 :

$A = 47_{(10)}$ $B = 425_{(10)}$ $C = 9019_{(10)}$ $D = 127_{(10)}$

Arithmétique
Codage des nombres

- Représentation de l'information

- Information
  - Instructions
  - Données
    - Caractère
    - Numérique
      - Entiers
        - Non signés (naturels)
        - Signés (relatifs)
      - Réels

Arithmétique
Codage des nombres

- Codage des entiers naturels :
  - Utilisation du code binaire pur :
    - L'entier naturel (positif ou nul) est représenté en base 2,
    - Les bits sont rangés selon leur poids, on complète à gauche par des 0.
  - Exemple : sur un octet, $10_{(10)}$ se code en binaire pur ?

$0\,0\,0\,0\,1\,0\,1\,0_{(2)}$

Arithmétique
Codage des nombres

- Codage des entiers naturels :
  - Étendu du codage binaire pur :
    - Codage sur n bits : représentation des nombres de 0 à $2^n – 1$
    - Sur 1 octet (8 bits): codage des nombres de 0 à $2^8 - 1 = 255$
    - sur 2 octets (16 bits): codage des nombres de 0 à $2^{16} - 1 = 65535$
    - sur 4 octets (32 bits) : codage des nombres de 0 à $2^{32} - 1 = 4\ 294\ 967\ 295$

Arithmétique
Codage des nombres

- Codage des entiers relatifs :
  - Il existe au moins trois façons pour coder :
    - Code binaire signé (par signe et valeur absolue)
    - Code complément à 1
    - Code complément à 2 (utilisé sur ordinateur)

Arithmétique
Codage des nombres

- Codage des entiers relatifs : binaire signé
  - Le bit le plus significatif est utilisé pour représenter le signe du nombre :
    - si le bit le plus fort = 1 alors nombre négatif
    - si le bit le plus fort = 0 alors nombre positif
  - Les autres bits codent la valeur absolue du nombre
  - Exemple : Sur 8 bits, codage des nombres -24 et 128
    - -24 est codé en binaire signé par : $1\,0\,0\,1\,1\,0\,0\,0_{(bs)}$

      Signe -

    - 128 ne peut pas être codé en binaire signé sur 8 bits

Arithmétique
Exercices

- Coder $100_{(10)}$ et $-100_{(10)}$ en binaire signé sur 8 bits
  - $100_{(10)} = 0110\ 0100_{(bs)}$
  - $-100_{(10)} = 1110\ 0100_{(bs)}$
- Décoder en décimal $1100\ 0111_{(bs)}$ et $0000\ 1111_{(bs)}$
  - $1100\ 0111_{(bs)} = -71_{(10)}$
  - $0000\ 1111_{(bs)} = 15_{(10)}$

Arithmétique
Codage des nombres

- Codage des entiers relatifs : complément à 1
  - Aussi appelé Complément Logique (CL) ou Complément Restreint (CR) :
    - Les nombres positifs sont codés de la même façon qu'en binaire pure
    - Un nombre négatif est codé en inversant chaque bit de la représentation de sa valeur absolue
  - Le bit le plus significatif est utilisé pour représenter le signe du nombre :
    - si le bit le plus fort = 1 alors nombre négatif
    - si le bit le plus fort = 0 alors nombre positif

Arithmétique
Codage des nombres

- Exemple : -24 en complément a 1 sur 8 bits
  - | -24 | en binaire pur   0 0 0 1 1 0 0 0 $_{(2)}$
  - Puis on inverse les bits   1 1 1 0 0 1 1 1 $_{(\text{cà1})}$
- Limitation :
  - deux codages différents pour 0 (+0 et -0)
  - Sur 8 bits : +0 = 0 0 0 0 0 0 0 0$_{(\text{cà1})}$ et -0 = 1 1 1 1 1 1 1 1$_{(\text{cà1})}$
  - La multiplication et l'addition sont moins évidentes.

Décimal	+	-
0	0000	1111
1	0001	1110
2	0010	1101
3	0011	1100
4	0100	1011
5	0101	1010
6	0110	1001
7	0111	1000

Arithmétique

Exercices

- Coder $100_{(10)}$ et $-100_{(10)}$ en complément à 1 sur 8 bits
  - $100_{(10)}$ = 0110 0100 $_{(Cà1)}$
  - $-100_{(10)}$ = 1001 1011 $_{(Cà1)}$
- Décoder en décimal 1100 0111 $_{(Cà1)}$ et 0000 1111 $_{(Cà1)}$
  - $11000111_{(Cà1)} = -56_{(10)}$
  - $00001111_{(Cà1)} = 15_{(10)}$

Arithmétique
Codage des nombres

- Codage des entiers relatifs : complément à 2
  - Aussi appelé Complément Vrai (CV) :
    - Les nombres positifs sont codés de la même manière qu'en binaire pure
    - un nombre négatif est codé en ajoutant la valeur 1 à son complément à 1
  - Le bit le plus significatif est utilisé pour représenter le signe du nombre
- Exemple : -24 en complément à 2 sur 8 bits
  - 24 est codé par : $0\ 0\ 0\ 1\ 1\ 0\ 0\ 0_{(2)}$
  - -24 est codé par : $1\ 1\ 1\ 0\ 0\ 1\ 1\ 1_{(cà1)}$
  - donc -24 est codé par : $1\ 1\ 1\ 0\ 1\ 0\ 0\ 0_{(cà2)}$

Arithmétique
Codage des nombres

- Codage des entiers relatifs : complément à 2
  - Un seul codage pour 0. Par exemple sur 8 bits :
    - +0 est code par : $0\ 0\ 0\ 0\ 0\ 0\ 0\ 0_{(cà2)}$
    - -0 est code par : $1\ 1\ 1\ 1\ 1\ 1\ 1\ 1_{(cà1)}$
    - Donc -0 sera représenté par : $0\ 0\ 0\ 0\ 0\ 0\ 0\ 0_{(cà2)}$
  - Étendu de codage :
    - Avec n bits, on peut coder de $-(2^{n-1})$ à $(2^{n-1}-1)$
    - Sur 1 octet (8 bits), codage des nombres de -128 a 127
      - +0 = 00000000 -0 = 00000000
      - +1 = 00000001 -1 = 11111111
      - … …
      - +127= 01111111 -128=10000000

Arithmétique
Exercices

- Coder $100_{(10)}$ et $-100_{(10)}$ par complément à 2 sur 8 bits
  - $100_{(10)} = 0110\ 0100_{(Cà2)}$
  - $-100_{(10)} = 1001\ 1100_{(Cà2)}$
- Décoder en décimal $1100\ 1001_{(Cà2)}$ et $0110\ 1101_{(Cà2)}$
  - $11001001_{(Cà2)} = -55_{(10)}$
  - $01101101_{(Cà2)} = 109_{(10)}$

Arithmétique
Codage des nombres réels

- Les formats de représentation des nombres réels sont :
  - Format à virgule fixe
    - Utilisé par les premières machines
    - Possède une partie 'entière' et une partie 'décimale' séparés par une virgule. La position de la virgule est fixe d'où le nom.
      - Exemple : $54,25_{(10)}$; $10,001_{(2)}$; $A1,F0B_{(16)}$
  - Format à virgule flottante (utilisé actuellement sur machine)
    - défini par : $\pm m.b^e$
    - Un signe : + ou -
    - Une mantisse : m (en virgule fixe)
    - Un exposant : e (un entier relatif)
    - Une base : b(2,8,10,16, …)
      - Exemple : $0,5424.10^2{}_{(10)}$; $10,1.2^{-1}{}_{(2)}$; $A0,B4.16^{-2}{}_{(16)}$

Arithmétique
Codage des nombres réels : Codage en virgule fixe

- Étant donné une base b, un nombre x est représenté, en format virgule fixe, par :
  - $x = a_{n-1}a_{n-2}\ldots a_1a_0,a_{-1}a_{-2}\ldots a_{-p\ (b)}$
  - a_{n-1} est le chiffre de poids fort (MSB)
  - a_{-p} est le chiffre de poids faible (LSB)
  - n est le nombre de chiffres avant la virgule
  - p est le nombre de chiffres après la virgule
  - La valeur de x en base 10 est : $x = \sum_{-p}^{n-1} a_i b^i \ _{(10)}$
  - Exemple : $101{,}01_{(2)}=1.2^2+0.2^1+1.2^0+0.2^{-1}+1.2^{-2} = 5{,}25_{(10)}$

Arithmétique

Codage des nombres réels : Codage en virgule fixe

- Conversion de base b → 10
  - La valeur de x en base 10 est : $x = \sum_{-p}^{n-1} a_i b^i$ (10)
  - Exemple :

$101{,}01_{(2)} = 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} = 5{,}25_{(10)}$

Même chose pour les bases 8→10, 16→10, …

Arithmétique

Codage des nombres réels : Codage en virgule fixe

- Conversion de base 10 → 2
- Le passage de la base 10 à la base 2 est défini par :
  - Partie entière est codée sur p bits (division successive par 2)
  - Partie décimale est codée sur q bits en multipliant par 2 successivement jusqu'à ce que la partie décimale soit nulle ou que le nombre de bits q désiré soit atteint (i.e. que l'on obtienne la précision demandée)
- Exemple 1 : $4{,}25_{(10)} = ?_{(2)}$ format virgule fixe
  - $4_{(10)} = 100_{(2)}$
  - 0,25 x 2 = 0,5 → **0**
  - 0,5 x 2 = 1,0 → **1**
  - Donc $4{,}25_{(10)} = 100{,}01_{(2)}$

Arithmétique

Codage des nombres réels : Codage en virgule fixe

- Exemple 2 : $1234{,}347_{(10)} = ?_{(2)}$ format virgule fixe
  - $1234_{(10)} = 10011010010_{(2)}$
  - 0,347 x 2 = 0,694 → **0**
  - 0,694 x 2 = 1,388 → **1**
  - 0,388 x 2 = 0,766 → **0**
  - 0,766 x 2 = 1,552 → **1**
  - 0,552 x 2 = 1,104 → **1**
  - 0,104 x 2 = 0,208 → **0**
  - 0,208 x 2 = 0,416 → **0**
  - …
  - On continue ainsi jusqu'à la précision désirée (que le nombre de bit soit atteint)

Arithmétique

Codage des nombres réels : Codage en virgule fixe

- Exemple 3 : $13{,}4_{(10)} = ?_{(2)}$ format virgule fixe
  - $13_{(10)} = 1101_{(2)}$
  - 0,4 x 2 = 0,8 → **0**
  - 0,8 x 2 = 1,6 → **1**
  - 0,6 x 2 = 1,2 → **1**
  - 0,2 x 2 = 0,4 → **0**
  - 0,4 x 2 = 0,8 → **0**
  - 0,8 x 2 = 1,6 → **1**
  - 0,6 x 2 = 1,2 → **0**
  - …
  - La séquence se reproduit indéfiniment

Arithmétique
Codage des nombres réels : Codage en virgule fixe

- Un problème se pose : l'arrondi en binaire !!
  - Si le premier symbole abandonné est 0, on fait une troncature – on abandonne les symboles suivants (arrondi par défaut)
  - Si le premier symbole abandonné est 1, on ajoute 1 au dernier symbole conservé (arrondi par excès).

13,4 = 1101,011001100110… → 1101,0110

Premier symbole abandonné est un zéro

Arithmétique

Codage des nombres réels : Codage en virgule flottante

$$\boldsymbol{x = \pm M.2^{E}}$$

- où M est la mantisse (virgule fixe) et E l'exposant (signé).
- Le codage en base 2, format virgule flottante, revient à coder le signe, la mantisse et l'exposant.
- Exemple : Codage en base 2, format virgule flottante de (3,25)
- $3{,}25_{(10)} = 11{,}01_{(2)}$ (en virgule fixe)
- $= 1{,}101.2^{1}{}_{(2)}$
- $= 110{,}1.2^{-1}{}_{(2)}$

Un problème se pose : on a différentes manières de représenter E et M → normalisation

Arithmétique

Codage des nombres réels : Codage en virgule flottante – normalisation

- Le signe est codé sur 1 bit ayant le poids fort :
  - Le signe – : bit 1
  - Le signe + : bit 0
- Exposant biaisé (Eb)
  - Placé avant la mantisse pour simplifier la comparaison
  - Codé sur p bits et **<u>biaisé pour être positif</u>** (ajout de $2^{p-1}-1$) (pour 8 bit : $2^{8-1}-1 = 127$)

Eb = D + B
D → déplacement algébrique de la virgule
B → décalage ou biais = $2^{p-1}-1$

- Mantisse normalisé(M)
  - Normalisé : virgule est placé après le bit à 1 ayant le poids fort
  - M est codé sur q bits
- Exemple : 11,01 → 1,101 donc M =101

SM	Eb	M
1 bit	p bits	q bits

Arithmétique

Codage des nombres réels : Codage en virgule flottante – normalisation

- Notre formule devient alors :

$$\boldsymbol{x = (-1)^{S}.2^{D+B}.(1+F)}$$

- S est le bit de signe et l'on comprend alors pourquoi 0 est positif ($-1^0=1$)
- D+B = Eb (décalage ou exposant biaisé ou biais)
- F est la partie fractionnaire (mantisse)

Arithmétique
Standard IEEE 754 (1985)

- Simple précision sur 32 bits :
  - 1 bit de signe de la mantisse
  - 8 bits pour l'exposant
  - 23 bits pour la mantisse

SM	Eb	M
1 bit	8 bits	23 bits

- Double précision sur 64 bits :
  - 1 bit de signe de la mantisse
  - 11 bits pour l'exposant
  - 52 bits pour la mantisse

SM	Eb	M
1 bit	11 bits	52 bits

Arithmétique
Standard IEEE 754 (1985)

- Certaines conditions sont toutefois à respecter pour les exposants :
  - L'exposant 00000000 est interdit.
  - L'exposant 11111111 est interdit. On s'en sert toutefois pour signaler des erreurs, on appelle alors cette configuration du nombre NaN, ce qui signifie « Not a Number ».
  - Il faut rajouter 127 (01111111) à l'exposant pour une conversion de décimal vers un nombre réel binaire. Les exposants peuvent ainsi aller de −254 à 255.

Arithmétique

Conversion décimale → IEEE754 (Codage d'un réel)

- $35{,}5_{(10)} = ?$ (IEEE 754 simple précision)
- Nombre positif, donc SM = 0
- $35{,}5 \ = \ 100011{,}1_{(2)}$ (virgule fixe)
- $\quad\quad = \ 1{,}000111.2^5{}_{(2)}$ (virgule flottante)
- Exposant : Eb = D + B → Eb = 5+127, donc Eb = 132
- 1,M = 1,000111 donc M = 00011100…

$$\overbrace{0}^{SM}\ \overbrace{10000100}^{Eb}\ \overbrace{00011100000000000000000}^{M}\ _{(IEEE754)}$$

Arithmétique

Conversion IEEE754 → décimal (évaluation d'un réel)

$$\underbrace{0}_{SM}\ \underbrace{10000001}_{Eb}\ \underbrace{11100000000000000000000}_{M}\ _{(IEEE754)}$$

- S = 0, donc nombre positif
- Eb = 129, donc D = Eb – 127 = 2
- 1,M = 1,111

$$\mathbf{1,111.2^2{}_{(2)} = 111,1_{(2)} = 7,5_{(10)}}$$

Arithmétique

Exercices

- Traduire le nombre -6,625 en utilisant la norme IEEE 754
- $6{,}625_{(10)} = 110{,}1010_{(2)}$ (virgule fixe)
- On met ce nombre sous sa forme fractionnaire ***1,partie fractionnaire***
- $110{,}1010_{(2)} = \boldsymbol{1{,}101010}\text{x}2^2{}_{(2)}$ (2^2 décale la virgule de 2 chiffres vers la droite)
- La partie fractionnaire étendue sur 23 bits est donc **101 0100 0000 0000 0000 0000**
- $Eb = 2+127 = 129_{(10)} = 1000\ 0001_{(2)}$
- Le résultat est donc

$$\overbrace{0}^{SM}\overbrace{110000001}^{Eb}\ \overbrace{10101000000000000000000}^{M}{}_{(IEEE754)}$$

Arithmétique
Fin

- Pour ceux que cela intéresse, vous pouvez faire des recherches sur l'architecture des ordinateurs (les portes logiques, les circuits logiques, le langage assembleur MIPS …)

Arithmétique

MANIPULATIONS ET CALCULS

Arithmétique

Addition Binaire

- Les additions en base 2 s'effectuent comme dans le système décimal, avec la notion de retenue ou *carry* (ici en rouge) en utilisant la table d'addition suivante :

+	**0**	**1**
0	0	1
1	1	[1]0

$0 + 0 = 0$ | $0_{(10)}$
$0 + 1 = 1$ | $1_{(10)}$
$1 + 0 = 1$ | $1_{(10)}$
$1 + 1 = 10$ | $2_{(10)}$
$1 + 1 + 1 = 11$ | $3_{(10)}$

Arithmétique

Addition Binaire élémentaire

+	0	1
0	0	1
1	1	$^{1}0$

```
  1
  4 7
+ 2 3
----------
= 7 0
```

```
   1 1 1 1 1 1
  0 0 1 0 1 1 1 1
+ 0 0 0 1 0 1 1 1
----------------------
= 0 1 0 0 0 1 1 0
```

Arithmétique
Addition Binaire élémentaire

Exercice

Addition en binaire élémentaire de: $(9)_{(10)} + (4)_{(10)}$ | $(44)_{(10)} + (17)_{(10)}$ | $(43)_{(10)} + (19)_{(10)}$

```
  0 1 0 0 1        (9)(10)
+ 0 0 1 0 0        (4)(10)
----------------
= 0 1 1 0 1        (13)(10)
```

```
  1 0 1 1 0 0      (44)(10)
+ 0 1 0 0 0 1      (17)(10)
----------------
= 1 1 1 1 0 1      (61)(10)
```

```
      1 1
  1 0 1 0 1 1      (43)(10)
+ 0 1 0 0 1 1      (19)(10)
----------------
= 1 1 1 1 1 0      (62)(10)
```

Arithmétique

Soustraction Binaire élémentaire

$$
\begin{array}{rccc}
 & 1 & {}_{1}9 & {}_{1}5 \\
- & {}_{1} & 9_{1} & 6 \\
\hline
= & 0 & 9 & 9
\end{array}
$$

$${}_{1}9 \longrightarrow 19$$

$$9_{1} \longrightarrow 9 + 1 \longrightarrow 10$$

$$
\begin{array}{rcccccccc}
 & 1 & {}_{1}1 & {}_{1}0 & 0 & 0 & 0 & 1 & 1 \\
- & 0_{1} & 1_{1} & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
= & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1
\end{array}
$$

$${}_{1}1 \longrightarrow 11 \quad 3_{(10)}$$

$${}_{1}0 \longrightarrow 10 \quad 2_{(10)}$$

$$1_{1} \longrightarrow 1 + 1 \longrightarrow 10 \quad 2_{(10)}$$

Arithmétique
Exercices

- Effectuer les additions suivantes :
  - A = 1100 + 0011
    1111
    B = 1111 + 0101
    10100
  - C = 10101010 + 00110011
    11011101
    D = 11001101 + 11100011
    110110000
- Effectuer les soustractions suivantes :
  - E = 1111 – 0101
    1010
    F = 1100 – 0011
    1001
  - G = 10101010 – 00110011
    01110111
    H = 11001101 – 01100011
    01101010

Arithmétique

Addition et soustraction Binaire signées

- Une soustraction peut toujours, si on rend négatif son second terme, se ramener à une addition, ainsi :

$$[\boldsymbol{A} - \boldsymbol{B}] = [\boldsymbol{A} + (-\boldsymbol{B})]$$

- La méthode la plus utilisée pour rendre négatif un nombre binaire est le complément à 2
- C'est cette méthode qui est utilisée par les machines

<u>Remarque :</u> en représentation signée binaire, le MSB représente le signe (0 si + et 1 si -). Les nombres signés sont également formatés.

Arithmétique

Addition et soustraction Binaire signées

- Soit l'opération suivante : 195 – 96
  - $195_{(10)} = 11000011_{(2)}$
  - $96_{(10)} = 01100000_{(2)}$
- En représentation signée binaire, +195 doit être représenté sur plus de 8 bits si l'on veut que son bit de signe soit positif. On va donc travailler sur 9 bits pour représenter son signe
  - $+195_{(10)} = 011000011_{(2)}$
  - $+96_{(10)} = 001100000_{(2)}$
  - $-96_{(10)} = 110100000_{(Cà2)}$

```
   1 1
    0 1 1 0 0 0 0 1 1        195
+   1 1 0 1 0 0 0 0 0     +(-96)
----------------------------
= 1 0 0 1 1 0 0 0 1 1         99
```

Comme on travail sur 9 bits, cette retenue est négligée

Signe +

Arithmétique

Addition et soustraction Binaire signées

- L’opération de base des calculateurs électroniques est l’addition
- Le résultat d’une soustraction de deux nombres binaires est en fait obtenu par l’addition du premier nombre par le complément à 2 du deuxième.
- Dans le cas d’un automate programmable, les nombres entiers sont stockées dans des mots formatés généralement sur 8, 16 ou 32 bits.
- Le bit de poids le plus fort (MSB) représente le signe (0 pour positif et 1 pour négatif).
- Étudions les différents cas possibles pour l'opération D = A – B

Arithmétique

Addition et soustraction Binaire signées

- Cas 1 : deux nombres positifs
  - Exemple : A = +9 et B = +4
  - Sur 5 bits : A = 01001 et B = 00100
  - A + B
  - L'addition est immédiate

	0 1 0 0 1	+ 9
+	0 0 1 0 0	+ 4
=	0 1 1 0 1	+ 13

Bits de signe

Signe +

<u>Remarque :</u> En complément à 2, les nombres doivent toujours avoir le même nombre de bits

Arithmétique

Addition et soustraction Binaire signées

- Cas 2 : un nombre positif et un nombre négatif plus petit
  - Exemple : A = +9 et B = -4
  - Sur 5 bits : A = 01001 et B = -(00100)
  - A + B = A + (-B)
  - $B_{(Cà2)}$ = 11100

```
     1 1
      0 1 0 0 1          + 9
+     1 1 1 0 0          + (-4)
---------------------------
= 1 0 0 1 0 1            + 5
```

négligé

Signe +

Arithmétique

Addition et soustraction Binaire signées

- Cas 3 : un nombre positif et un nombre négatif plus grand
  - Exemple : A = -9 et B = +4
  - Sur 5 bits : A = -(01001) et B = 00100
  - -A + B = +(-A) + B
  - $A_{(Cà2)}$ = 10111

```
       1
       1 0 1 1 1          + (-9)
+      0 0 1 0 0          + 4
-----------------------------
=      1 1 0 1 1          - 5
```

Signe -

  - Le bit de signe de la somme est négatif, on doit complémenter à 2 le résultat : 1011 = $0101_{(Cà2)}$ = 5, comme le bit de signe est 1, on obtient -5

Arithmétique

Addition et soustraction Binaire signées

- Cas 4 : deux nombres négatifs
  - Exemple : A = -9 et B = -4
  - Sur 5 bits : A = -(01001) et B = -(00100)
  - -A - B = +(-A) + (-B)
  - $A_{(Cà2)}$ = 10111
  - $B_{(Cà2)}$ = 11100

```
     1 1 1
       1 0 1 1 1        + (-9)
+      1 1 1 0 0        + (-4)
---------------------
= 1 1 0 0 1 1           -13
```

négligé

Signe -

- Le bit de signe de la somme est négatif, on doit complémenter à 2 le résultat : 0011 = $1101_{(Cà2)}$ = 13, comme le bit de signe est 1, on obtient -13

Arithmétique

Addition et soustraction Binaire signées

- Cas 5 : deux nombres égaux et opposés
  - Exemple : A = -9 et B = +9
  - Sur 5 bits : A = -(01001) et B = 01001
  - -A + B = +(-A) + B
  - $A_{(Cà2)}$ = 10111

```
   1 1 1 1 1
     1 0 1 1 1        + (-9)
+    0 1 0 0 1        + (9)
-----------------------------
= 1  0 0 0 0 0        +0
```

négligé

Signe -

  - Le bit de signe de la somme est positif, on a donc +0

Arithmétique
Addition et soustraction Binaire signées

- L'addition de deux nombres de même signe peut donner lieu à un dépassement de capacité (OverFlow) ! Le résultat obtenu est alors faux!!
- On a un dépassement de capacité quand le bit de signe du résultat est différent de celui des deux nombres additionnés.
  - Le nombre de bits utilisés est insuffisant pour contenir le résultat
  - Autrement dit le résultat dépasse l'intervalle des valeurs sur les n bits utilisés (sur 8 bits, on dépasse l'intervalle -128 et +127)

Arithmétique

Addition et soustraction Binaire signées

- On est jamais confronté a un dépassement de capacité lorsque les signes des deux nombres sont opposés

```
 1   1 1 1 1 1 1                                      1       1 1 1
   1 0 0 1 1 0 0 1        + (-103)          0 1 1 0 0 1 1 1            +103
+  1 0 1 1 1 1 1 1        + (-65)      +    0 1 0 0 0 0 0 1            +65
-------------------------                  -------------------------
= 1 0 1 0 1 1 0 0 0   ≠   +(-168)      =    1 0 1 0 1 0 0 0     ≠     +168
        88                                       -88
```

Arithmétique

Exercices

- On dispose d'une machine travaillant sur des nombres binaires de longueur 8 (8 bits). Faire manuellement ce que l'additionneur de la machine ferait automatiquement, et donner les résultats obtenus en binaire. Eventuellement, en cas d'erreur, indiquer pourquoi.

  A = -61 -44   B = -61 -72 C = 99 – 35 D = 99 + 35

Arithmétique

Exercices

- 61 = 00111101 -61 = 11000011
- 44 = 00101100 -44 = 11010100
- 72 = 01001000 -72 = 10110111
- 99 = 01100011 -99 = 10011101
- 35 = 00100011 -35 = 11011101

A

```
      1 1
      1 1 0 0 0 0 1 1      +(-61)
+     1 1 0 1 0 1 0 0      +(-44)
--------------------------
= 1 1 0 0 1 0 1 1 1        +(-105)
```

On supprime le bit de trop.
Le résultat est de signe -. On complémente à 2:
0010111 = $1101001_{(Cà2)}$ = - 105

Arithmétique

Exercices

B

```
      1
        1 1 0 0 0 0 1 1        +(-61)
  +     1 0 1 1 1 0 0 0        +(-72)
  -------------------------
  = 1 **0 1 1 1 1 0 1 1**      +(-133)
```

On supprime le bit de trop.
Le résultat est de signe +. Le résultat est faux!!
Il y a débordement (overflow) : on est en dehors de la zone entre –128 et +127 correspondant aux nombres signés de 8 bits.

C

```
      1
        0 1 1 0 0 0 1 1        +(99)
  +     1 1 0 1 1 1 0 1        +(-35)
  -------------------------
  = 1 0 1 0 0 0 0 0 0          +(64)
```

On supprime le bit de trop.
Le résultat est de signe +. Le résultat est juste. C'est toujours le cas pour une addition de deux nombres de signes opposés .

D

```
        1 1         1 1
        0 1 1 0 0 0 1 1        +(99)
  +     0 0 1 0 0 0 1 1        +(35)
  -------------------------
  =     1 0 0 0 0 1 1 0        +(134)
```

On supprime le bit de trop.
Le résultat est de signe -. Le résultat est faux!!
Il y a débordement (overflow) : on est en dehors de la zone entre –128 et +127 correspondant aux nombres signés de 8 bits.

Arithmétique

Addition Binaire : flottants

Arithmétique

Addition Binaire : flottants

- Soit deux nombres réels N_1 et N_2 tel que

$$N_1 = M_1 . b^{e_1} \qquad N_2 = M_2 . b^{e_2}$$

- On veut calculer N_1+N_2 ?
- Deux cas se présentent :
  - Si $e_1 = e_2$ alors $N_3 = (M_1 + M_2) . b^{e_1}$
  - Si $e_1 \neq e_2$ alors élever au plus grand exposant et faire l'addition des mantisses et par la suite normalisée la mantisse du résultat.

Arithmétique

Addition Binaire : flottants

- Exemple 1
- Soient les nombres A = 1,5 et B = 0,5
- Représentation en IEEE :
  - $A = +1,1_{(2)}$ (virgule fixe) avec un signe + (donc S = 0)
  - $A = +1,1 \, . \, 2^0$ (virgule flottante)
  - $B = +0,1_{(2)}$ (virgule fixe)
  - $B = +1,0 \, . \, 2^{-1}$ (virgule flottante)
  - A et B > 0 ➔ $S_A = 0$ et $S_B = 0$
  - $Eb_A = 0+127 = 127$ $Eb_B = -1 + 127 = 126$
  - A = 0 01111111 10000000000000000000000 avec 1,M = 1,10000000…
  - B = 0 01111110 00000000000000000000000 avec 1,M = 1,000000000…

Arithmétique
Addition Binaire : flottants

- On ramène les deux nombres au même exposant, le plus grand des deux.
- On décale donc les bits de la mantisse du nombre ayant le plus petit exposant d'autant de bits vers la droite que la différence entre les exposants, sans oublier le 1 avant la virgule.
- Dans l'exemple on veut augmenter de 1 l'exposant du deuxième terme B. La mantisse complète du second nombre passe donc de 1.00000000000… à 0.10000000000… (on supprime le zéro le plus à droite pour rester sur 23 bits).
- On ajoute ensuite les deux mantisses, en tenant compte du signe (attention ici les mantisses ne sont pas exprimées en complément à 2) Dans l'exemple les mantisses sont de même signe on peut donc les ajouter directement :

```
     1,10000000000000000000000     x2^0
+    0,10000000000000000000000     x2^0
-----------------------------------------
=   10,00000000000000000000000     x2^0
```

Arithmétique

Addition Binaire : flottants

- On renormalise ensuite le nombre obtenu.
- Dans l’exemple, le résultat a donc pour exposant 0 et pour mantisse 10.000000000… : on renormalise la mantisse, ce qui augmente l’exposant de 1.
- On a donc le résultat final :
- 0 10000000 00000000000000000000000 = $+1{,}0 \, . \, 2^{1} = +2_{(10)}$

Arithmétique
Addition et soustraction Binaire : flottants

- Exemple 2
- Soient les nombres A = -1,5 et B = -0,5
- Représentation en IEEE :
  - A = $-1,1_{(2)}$ (virgule fixe) avec un signe + (donc S = 0)
  - A = $-1,1 \cdot 2^{0}$ (virgule flottante)
  - B = $-0,1_{(2)}$ (virgule fixe)
  - B = $-1,0 \cdot 2^{-1}$ (virgule flottante)
  - A et B < 0 ➔ $S_A = 1$ et $S_B = 1$
  - $Eb_A = 0+127 = 127$ $Eb_B = -1 + 127 = 126$
  - A = 1 01111111 10000000000000000000000 avec 1,M = 1,10000000…
  - B = 1 01111110 00000000000000000000000 avec 1,M = 1,000000000…

Arithmétique

Addition et soustraction Binaire : flottants

- On ramène les deux nombres au même exposant, le plus grand des deux
  - M_A = 1 , 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x 2^0
  - M_B = 0 , 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x 2^0
- On ajoute ensuite les deux mantisses. Comme les deux mantisses sont de même signe, on les ajoute directement :

```
     1 , 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0        x2^0
+    0 , 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0        x2^0
-----------------------------------------------------------------------
=  1 0 , 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0        x2^0
```

- On renormalise ensuite le nombre obtenu : 1,0000… x2^1
- Le signe de la mantisse est 1, donc A + B = -2

Arithmétique

Addition Hexadécimales : entiers positifs

A	B	C	D	E	F
10	11	12	13	14	15

- Cas 1: sans retenue

```
   3 4  B  5          5 + 4 = 9
+  6 6  1  4          B + 1 = 11 + 1 = 12 = C
-----------------     4 + 6 = 10 = A
=  9 A C 9            3 + 6 = 9
```

- Cas 2 : avec la retenue

```
          1
    5 2  6  4         4 + E = 18 (16 en retenue + 2 au résultat)
+  A 3  2  E          1 + 6 + 2 = 9
-----------------     2 + 3 = 5
=  F 5   9  2         5 + A = 5 + 10 = 15 = F
```

Arithmétique

Addition Hexadécimales : entiers positifs

- Cas 2 : avec la retenue

```
   1     1
   4  B  C 3
+  2 A   2 F
----------------
=  7 5   F 2
```

3 + F = 18 (16 en retenue + 2 au résultat)
1 + C + 2 = F
B+ A = 11 + 10 = 21 (16 en retenue + 5 au résultat)
1 + 4 + 2 = 7

```
     1  1  1
     F  F  F  F
+    F  F  F  F
----------------
= 1  F  F  F  E
```

F + F = 15 + 15 = 30(16 en retenue + 14 au résultat = E)
1 + F + F = 1 + 15 + 15 = 31(16 en retenue + 15 au résultat = F)
1 + F + F = 1 + 15 + 15 = 31(16 en retenue + 15 au résultat = F)
1 + F + F = 1 + 15 + 15 = 31(16 en retenue + 15 au résultat = F)
1 en retenue à remettre au résultat

Arithmétique
Exercices

- Effectuez les additions suivantes :

DAE + F5   23CB+54BD   24A6+5FBE

EA3   7888   8464

A	B	C	D	E	F
10	11	12	13	14	15

Arithmétique

Soustraction Hexadécimales : entiers positifs

A	B	C	D	E	F
10	11	12	13	14	15

$$\begin{array}{lcccc} & 9 & B & 5 & {}_{1}4 \\ - & 6 & A & 2_{1} & 9 \\ \hline = & 3 & 1 & 2 & B \end{array}$$

4 – 9 = (4 + 16) – 9 = 11 = B
5 – 3 = 5 – (2 + 1 de retenue) = 2
B – A = 11 – 10 = 1
9 – 6 = 3

A vous de jouer:

$$\begin{array}{lcccc} & A & {}_{1}B & {}_{1}C & {}_{1}D \\ - & 2_{1} & F_{1} & F_{1} & F \\ \hline = & 7 & B & C & E \end{array}$$

D – F = 13 – 15 = (13+16) – 15 = 14 = E
C – (F+1) = 12 - (15 + 1) = (12+16) – (15+1) = 12 = C
B – (F+1) = 11 –(15+1) = (11+16) – (15+1) = 11 = B
A – (2+1) = 10 – 3 = 7

<u>Remarque</u> : Cas où le premier nombre est supérieur au second
sinon, il faut passer par le complément à 2

Arithmétique

Multiplication Binaire : entiers positifs

*	0	1
0	0	0
1	0	1

```
     1 0 1 1 0 1              45
*          1 0 1            x  5
--------------------------
=    1 0 1 1 0 1
   0 0 0 0 0 0
 1 0 1 1 0 1
--------------------------
 1 1 1 0 0 0 0 1        =    225
```

Arithmétique
Multiplication Binaire : entiers signés

- Cas de deux nombres positifs :
  - on applique la même méthode que précédemment
- Cas de deux nombres négatifs :
  - On calcul le complément à 2 de chacun des nombres (pour les rendre positifs)
  - On multiplie ces deux nombres positifs comme précédemment
  - Le résultat obtenu est positif, on ne change rien
- Cas ou l'un des deux nombres est négatif :
  - On calcul le complément à deux du nombre négatif
  - On multiplie les deux nombres positifs
  - On complémente à deux le résultat, puisqu'il est négatif

Arithmétique

Division Binaire : entiers positifs

- Dans la division en binaire, on utilise une succession de soustractions.

```
 1001100001 | 101
-101        |--------
----------- | 0001
 01001      | 00011
- 101       | 000111
----------- | 0001111
 01000      | 0001111001
- 101       |
----------- |
    110     |
-   101     |
----------- |
     1001   |
-     101   |
----------- |
      100   |
```

1. On prend le premier bit du nombre à diviser.
Si ce nombre est supérieure ou égal au diviseur, on note 1 dans le résultat et on fait la soustraction. Sinon, on note 0 dans le résultat et on test avec un bit en plus pour le nombre à diviser
2. On répète l'étape 1 tant que le nombre à diviser est inférieur au diviseur.
3. On abaisse un bit et on test si le nombre est plus grand. Si oui, on note 1 et on fait la soustraction, sinon, on note 0 et on abaisse le bit suivant. On répète, tant que le nombre est inférieur.
4. On répète le point 3 tant que le nombre de bit du nombre à diviser n'est pas atteint
5. On peut continuer on ajoutant une virgule au résultat est des zéros et on répète l'étape 3

1001100001/101 = 1111001,110011001100….

Arithmétique

Exercices

- Effectuez la division suivante :
  - 10001/100 100 reste 001
  - 11001/101 1100 reste 1001
  - 1111100/1001000111 0 reste 1111100
  - 1111111110010/1000011000 1111 reste 10001010

Arithmétique

Multiplication et division par une puissance de 2 en binaire

- Pour multiplier (respectivement diviser) par $2^n = 10\ldots0_2$ (n zéros) un nombre écrit en base 2, on décale tous ses chiffres de n rangs vers la gauche (ou la droite pour la division)
- Exemples :
  - $10010_{(2)} \times 100_{(2)} = 1001000_{(2)}$
  - $11{,}001_{(2)} \times 10 = 110{,}01_{(2)}$
  - $10010_{(2)} / 100_{(2)} = 100{,}1_{(2)}$
  - $11{,}011_{(2)} / 10_{(2)} = 1{,}1001_{(2)}$