Chapitre I
Structure de l'Ordinateur & Systèmes de Numération

Sommaire

Cours de Structure des Ordinateurs & Applications :
https://elearning.univ-bejaia.dz/course/view.php?id=14316

Cours des bases d'Algorithmique sur Elearning :
https://elearning.univ-bejaia.dz/course/view.php?id=7944

Page facebook :
https://www.facebook.com/InitiationAlgoProgrammation/

La chaîne Youtube :
https://www.youtube.com/c/AlgoProgrammation1èreAnnéeTechnologie
https://www.youtube.com/@algo-prog

Adapté par : Redouane OUZEGGANE
rouzeggane@gmail.com - redouane.ouzeggane@univ-bejaia.dz

Chapitre I : Représentation & Codification des nombres.

I.1. Définition de l'Informatique

L'informatique est une branche qui s'occupe du domaine du traitement automatique de l'information. Le mot « INFORMATIQUE » est composé à partir des mots « INFORMATION » et « AUTOMATIQUE ». L'informatique a pour rôle :

- La conception et la construction des ordinateurs,
- Le fonctionnement et la maintenance des ordinateurs,
- Leur exploitation (utilisation des ordinateurs dans les différents domaines d'activités).

On écrit :

INFORMATIQUE = INFORmation + autoMATIQUE

L'informatique est la science du traitement automatique et rationnel de l'information, en tant que support de connaissances et des communication ; c'est aussi l'ensemble des applications de cette science, mettant en œuvre du matériels (ordinateur, imprimantes, scannair, cables réseaux, …) et du logiciels (système d'exploitation, programmes d'application : Word, Excel, My-PASCAL, My-C, …).

I.2. Aperçu sur l'architecture d'un ordinateur

Un ordinateur est composé de deux parties : partie matérielle (hardware) et la partie logicielle (matérielle). Dans ce cous, nous nous focalisons sur la partie matérielle de l'ordinateur et expliquant son architecture et sa structure logique. Par la suite, nous présentons une structure physique simplifié.

Comme indiqué sur la *figure I.1,* un ordinateur, d'une manière générale, est constitué d'un *CPU* (Central Process Unit : Unité de Traitement Centrale) et des périphérique. Le *CPU* est constitué d'un *Micro-Processeur* (ou *Processeur*) et d'une *Mémoire Centrale.* Cette dernière est constituée d'une mémoire vive (*RAM* : Random Access Memory) et d'une mémoire morte (*ROM* : Read Only Memory).

Le processeur se compose de deux unités : unité de contrôle et unité arithmétique et logique. Et les périphérique sont composés de périphériques d'entrée, périphériques de sortie et et périphérique de sauvegardes (Mémoires externes).

Les différents composants illustrés par la figure en page suivante, sont comme suit :

I.2.1. Micro-Processeur

I.2.1.1. L'U.A.L (Unité Arithmétique et Logique : Circuit de Calcul)

Elle consiste en circuit de calcul qui est chargée d'effectuer toutes les opérations arithmétique et logiques. Elle contient tous les circuits logique pour réaliser les différentes opérations arithmétique (Addition, soustraction, multiplication, division, etc.) et logiques (décalage, rotation, et logique, et binaire, ou logique, ou binaire, négation logique, complément, etc.).

I.2.1.2. L'U.C. (Unité de contrôle : Circuit de commandes)

Elle contrôle toutes les opérations qui s'effectuent dans l'ordinateur. C'est elle qui permet l'exécution d'un programme présent en mémoire centrale (M.C.)

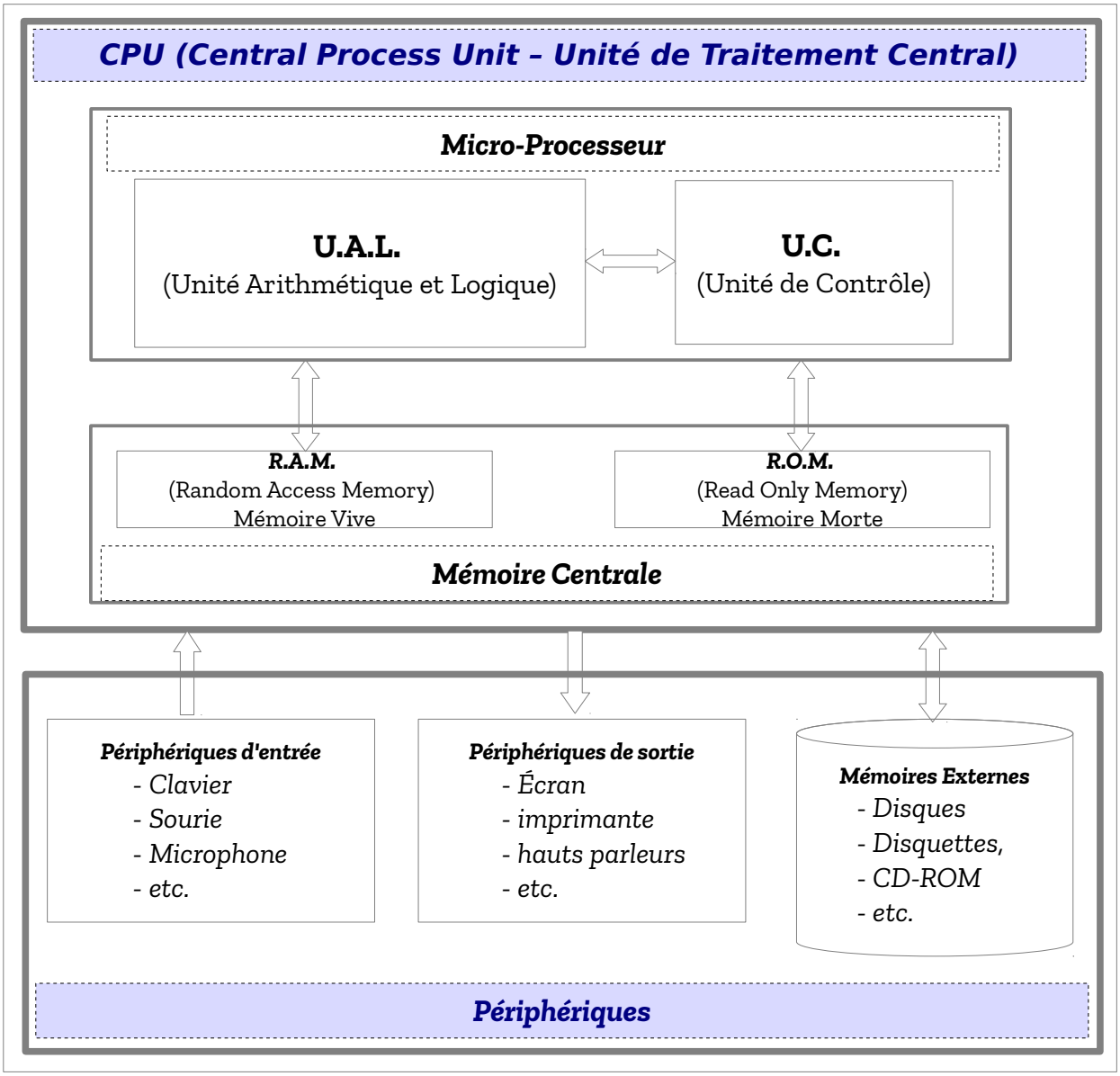


Figure I.1 : *Structure d'un Ordinateur*

C.P.U : Central Process Unit (Unité de Traitement Centrale)

U.C. : Unité de contrôle ou de commande

U.A.L. : Unité Arithmétique et Logique

M.C. : Mémoire Centrale.

I.2.2. La M.C. (La mémoire centrale : circuit de mémorisation)

I.2.2.1. RAM

La RAM, pour Random Access Mémory (Mémoire à Accès Aléatoire - Mémoire vive ou mémoire volatile), a pour rôle de stocker les programmes à exécuter (en cours d'exécution) ainsi que les données résultats. Un programme ne peut s'exécuter que s'il est chargé en mémoire centrale. Elle est constituée de plusieurs barrettes (barrettes RAM) . Cette mémoire est volatile, c'est à dire qu'elle s'efface s'il y a une coupure de courant.

I.2.2.2. ROM

La ROM (Read Only Mémory – Mémoire en lecture seule – Mémoire morte) mémoire durable et ne s'efface pas par coupure de courant. Cette dernière sert à conserver du code et des paramètres système nécessaire au fonctionnement de l'ordinateur (BIOS : Basic Input/Output System - Programme de base des entrées/sorties).

La mémoire centrale se mesure actuellement par milliers de méga-octets. On a :

- 1 bit : (état électronique 1 ou 0) = unité élémentaire de l'information (Binary Element ou Binary Digit)
- 1 octet = 1 caractère = 8 bits = 2^3 bits.
- 1 Kilo-octets = 2^{10} octets = 1024 octets
- 1 Mega-octets = 2^{10} Kilo-octets = 2^{20} octets
- 1 Giga-octets = 2^{10} Mega-octets = 2^{20} Kilo-octets = 2^{30} octets

I.2.3. Les Périphériques

I.2.3.1. Les mémoires externes ou auxiliaires (Support de stockage / mémorisation)

Elle servent à conserver des grandes quantités d'informations (fichiers de données, programmes, logiciels d'applications, système d'exploitation, *etc.*). Ceux sont des supports magnétiques, optique ou électroniques qui ne s'effacent pas par coupures de courant. On peut en citer les disques magnétiques, disquettes, cassettes, CD-ROM, flash-disque, *etc.*

Actuellement la capacité des disques magnétiques compte de dizaines de Giga-octets

pour les ordinateurs personnels (P.C.)

$$1 \text{ Giga-octets} = 2^{30} \text{ octets} \simeq 1 \text{ milliard d'octets}$$

I.2.3.2. Les périphériques d'entrée (ou organes d'entrée)

Ils servent à transmettre les informations à l'ordinateur, plus précisément vers le CPU . Exemples :

- Clavier (il y a des claviers AZERTY ou QWERTY : introduire des textes et des commandes)
- Microphone (introduire des sons et de la parole)
- Caméra (introduire de la vidéo)
- Lecteur optique (introduire des données à partir de CD ou DVD)
- Scanner (numérisé des photos)
- Tous de type de capteurs (Capteur de chaleurs, d'acidité, *etc.*)

I.2.3.3. Les périphériques de sortie (ou organes de sortie)

Ils servent à recevoir les information provenant du CPU vers l'extérieur. Ils permettent de traduire les informations de leurs formes codées (des 1 et 0) vers leurs formes naturelles. Exemple :

- L'écran
- L'imprimante
- Table traçante
- Synthétiseur vocal
- Synthétiseur musical
- Haut parleur, *etc.*

Remarque :

Les mémoires externes ou auxiliaires (Disques durs, disquettes, flash-disque, CD, DVD, etc.) sont considérées comme des périphériques d'entrée et de sortie en même temps.

La structure physique (réelle) d'un ordinateur est illustrée par la figure ci-dessous :

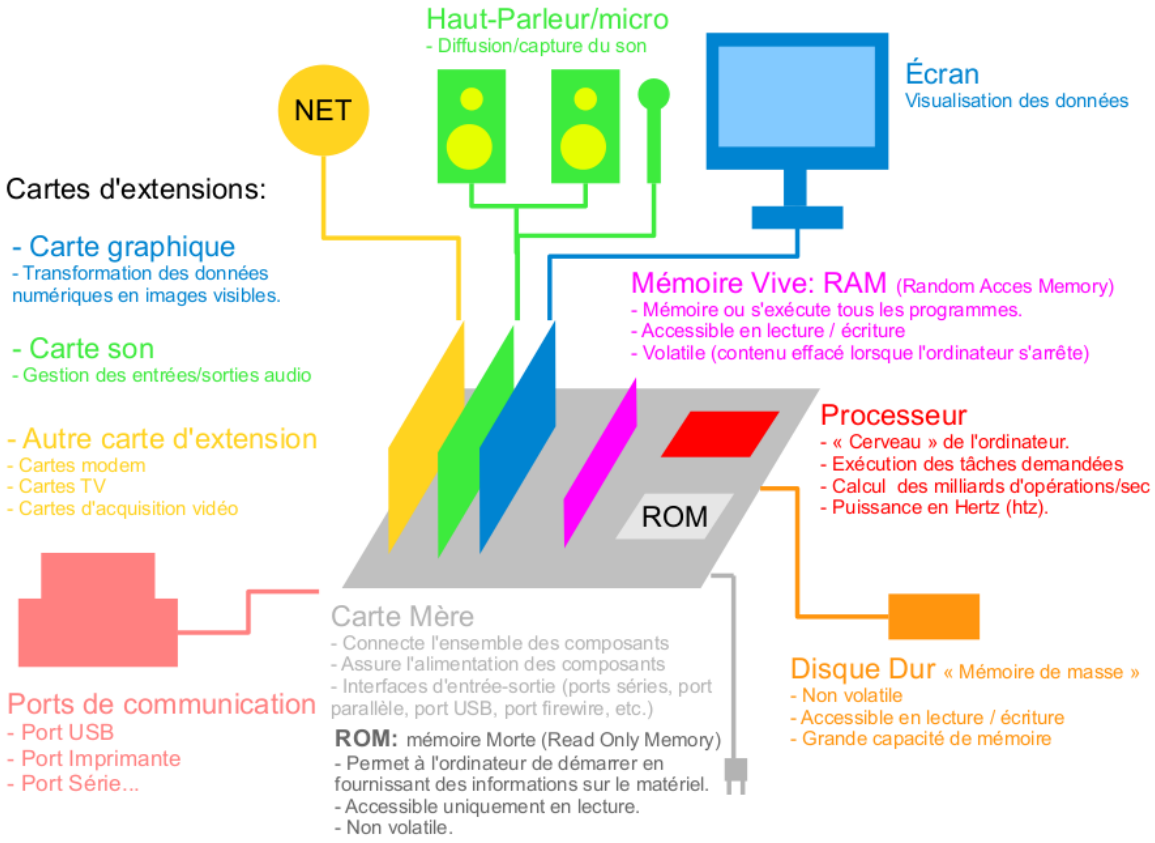


Figure I.2 : *Structure de la partie matérielle d'un ordinateur.*

I.3. Les systèmes de numération

Dans cette section, nous présentons les différents systèmes de numération à base 2, 8 et 16 et comment les nombres sont écrits et convertis d'une base à une autre. Par la suite, nous allons voir les opérations arithmétiques de bases sur les bases : 2, 8 et 16.

I.3.1. Le système de numération base 10 : système décimale

Le système de numération décimale est construite sur la base de *10* chiffres (*0, 1, 2, 3, 4, 5, 6, 7, 8, 9*). Prenons un exemple, pour illustré la décomposition d'une nombre en base 10. Le nombre 378 en numération décimale (base 10) se décompose de la façon suivante :

$378 = (378)_{10} = 8 \times 10^0 + 7 \times 10^1 + 3 \times 10^2$

D'une façons générale, un nombre N en base 10 s'écrit et se décompose comme suit :

$$N = (C_m C_{m-1} \ldots C_2 C_1 C_0)_{10} = C_0 \times 10^0 + C_1 \times 10^1 + C_2 \times 10^2 + \ldots C_{m-1} \times 10^{m-1} + C_m \times 10^m = \sum_{i=0}^{m} C_i \times 10^i$$

Tel-que : C_i ($i=\overline{1,m}$) sont les chiffres de N dans la base 10 et $0 \leq C_i < 10$.

I.3.2. Le système de numération base b

Le système de numération base *b*, tel-que *b* est un nombre naturel non-nul et supérieure à 1. Donc, on peut écrire $b \in \mathbb{N}^*\text{-}\{1\}$ (la plus petite base qui peut exister est 2). De la même façon que la base 10, on peut écrire un nombre N en base *b* comme suit :

$$N = (C_m C_{m-1} \ldots C_2 C_1 C_0)_b = C_0 \times b^0 + C_1 \times b^1 + C_2 \times b^2 + \ldots C_{m-1} \times b^{m-1} + C_m \times b^m = \sum_{i=0}^{m} C_i \times b^i$$

Tel-que : C_i ($i=\overline{1,m}$) sont les chiffres de N dans la base b et $0 \leq C_i < b$. Le résultat de formule ci-dessus, est un nombre écrit en base 10.

Remarques :

- Dans une base de numération ***b*** ($b \in \mathbb{N}^*\text{-}\{1\}$), nous avons b chiffres : 0, 1, 2, … (b-1).
- Si b = 10, on est dans la base décimale, nous aurons 10 chiffres : 0, 1, …, 9
- Si b = 2, on est dans la base binaire, nous aurons 2 chiffres : 0, 1
- Si b = 8, on est dans la base octale, nous aurons 8 chiffres : 0, 1, …, 7
- Si b = 16, on est dans la base hexadécimal, nous aurons 16 chiffres : 0, 1, …, 9, A, B, C, D, E, F

I.3.3. Le système de numération base 2 : système binaire

En remplaçant b par 2, dans la formule précédente, nous aurons :

$$N = (C_m C_{m-1} \ldots C_2 C_1 C_0)_2 = C_0 \times 2^0 + C_1 \times 2^1 + C_2 \times 2^2 + \ldots C_{m-1} \times 2^{m-1} + C_m \times 2^m = \sum_{i=0}^{m} C_i \times 2^i$$

Tel-que : Ci ($i=\overline{1,m}$) sont les chiffres de N dans la base 2 et $0 \leq Ci < 2$.

Par exemple :

$(101101)_2 = 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 1 \times 2^5 = 1 + 4 + 8 + 32 = 45$

I.3.4. Autres Systèmes de numération

- Système de numération octale (base *8*) : est utilise les *8* chiffres : *0, 1, 2, 3, 4, 5, 6, 7*
- Système de Numération hexadécimale (base *16*) : elle utilise les *16* chiffres et lettres suivants : *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F*

avec :

$A = (10)_{10} = (1010)_2$ $B = (11)_{10} = (1011)_2$ $C = (12)_{10} = (1100)$

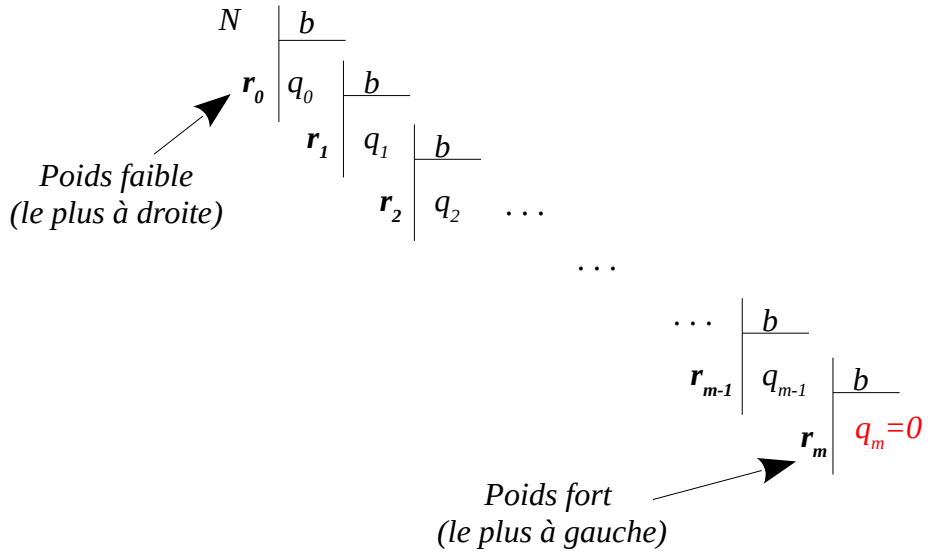
$D = (13)_{10} = (1101)_2$ $E = (14)_{10} = (1110)_2$ $F = (15)_{10} = (1111)_2$

I.3.5. Conversion d'un nombre d'un système à un autre

a) Conversion Décimale → Base b

Pour convertir un nombre N écrit en base ***10*** vers la base ***b*** (b entier positif et b >=2), on réalise des divisions euclidiennes successives sur b, jusqu'à avoir un quotient nul, et on prends les restes de divisions comme chiffres du nombre N dans la base ***b***, tel-que le premier reste de division sera à droite (la poids faible) et le dernier reste de division sera à gauche (le poids fort).

Ces divisions successives seront est réalisées comme indiqué ci-dessous :



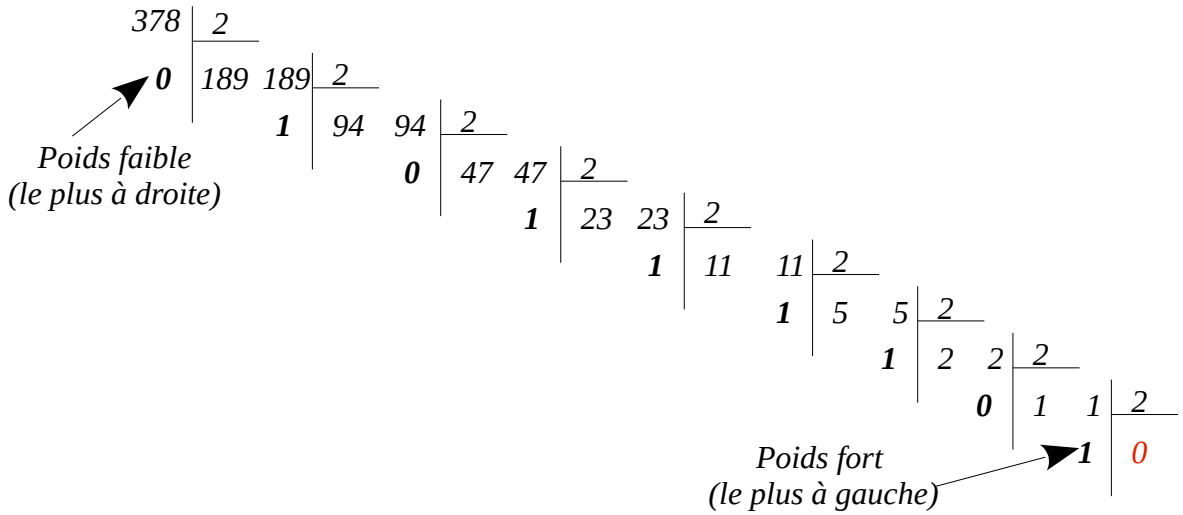
$$N = (N)_{10} = (r_m\, r_{m-1} \ldots r_2 r_1 r_0)_b$$

Pour convertir la valeur de N de la base b vers la base 10, on applique la formule de décomposition de la section *I.3.2.* :

$$N = (r_m\, r_{m-1} \ldots r_2 r_1 r_0)_b = \sum_{i=0}^{m} r_i \times b^i$$

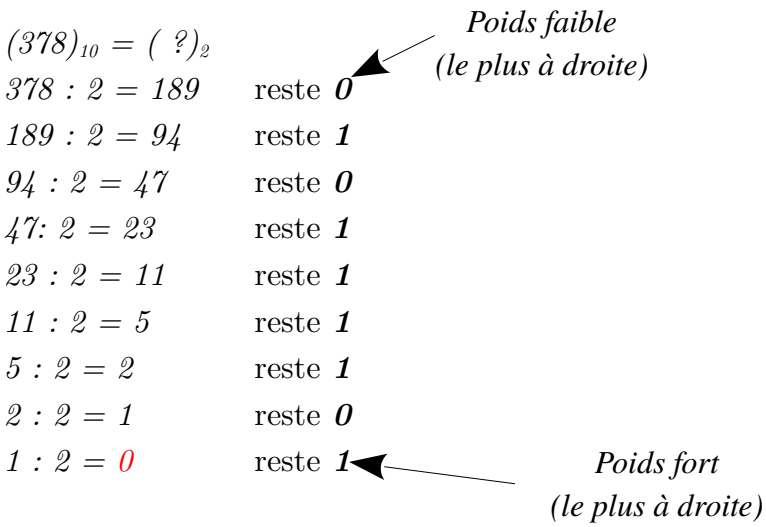
b) Conversion Décimale → Binaire

En remplaçant la base b par la valeur 2 dan les divisions euclidiennes, nous obtiendrons la valeur de N, écrit en base 10, en base 2 (en binaire). Voir l'exemple suivant :



$$(378)_{10} = (101111010)_2$$

Nous pouvons réaliser les divisions d'une façon linéaire, comme suit :



b) Conversion Binaire → Décimal

Nous utilisons la formule vue dans la section *1.3.3.* Prenons l'exemple suivant :

$$(111101)_2 = (?)_{10}$$

Chaque chiffre possède un poids (la puissance qu'il faut affecter à la base). Le chiffre du poids faible est le chiffre situé le plus à droite, et qui possède le poids **0**. Le chiffre du poids fort est le chiffre situé le plus à gauche, et qui possède le poids (**Nombre de chiffre -1**). Dans l'écriture suivante, nous représentant les poids des chiffres :

$$\begin{array}{l} \quad\ 5\ \ 4\ \ 3\ \ 2\ 1\ \ 0 \\ (1\ 1\ 1\ 1\ 0\ 1)_2 = 1\times2^0 + 0\times2^1 + 1\times2^2 + 1\times2^3 + 1\times2^4 + 1\times2^5 \\ \qquad\qquad = \quad 1 \quad + \quad 0 \quad + \quad 4 \quad + \quad 8 \quad + \quad 16 \quad + 32 \\ \qquad\qquad = \quad (61)_{10} \ = 61 \end{array}$$

c) Conversion base b1 → base b2

Pour convertir un nombre N d'une base b1 (différent de 10) vers base b2 (différent de 10 et de b1), on utilise la base 10 comme base intermédiaire. Donc, on converti, tout d'abord de la base b1 vers la base 10 (voir la section *I.3.2.*) par la suite de la base 10 vers la base b1 (voir la section *I.3.5. (a)*).

c) Conversion Octal → Binaire

Pour convertir un nombre de base 8 (octal) vers la base 2, et puisque $8 = 2^3$, ce n'est pas obligatoire de passer par la base 10. On applique une autre méthode, plus simple et plus rapide.

Nous avons $8 = 2^3$. donc chaque chiffre octal est replacé par 3 chiffres binaires.

Avant de convertir un nombre de base 8 vers la base 2, on établi un tableau de correspondance entre les chiffres de base 8 (octal) et les chiffres de base 2 (binaire), tel-que chaque chiffre octal sera écrit avec 3 chiffres binaires, comme suit :

Base 8	**Base 2**
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Exemples :

$(35607)_8 = (\ ?)_2$

Dans le tableau précédent nous avons les correspondances suivantes :

$3 \rightarrow 011 \quad 5 \rightarrow 101 \quad 6 \rightarrow 110 \quad 0 \rightarrow 000 \quad 7 \rightarrow 111$

Donc : $(35607)_8 = (011\ 101\ 110\ 000\ 111)_2 = (11\ 101\ 110\ 000\ 111)_2$

d) Conversion Binaire → Octal

Dans ce cas, on réalise l'opération inverse, en remplaçant chaque trois chiffres binaires par un chiffre octal, en procédant de droite vers la gauche :

$(1100101110111)_2 = (?)_8$. On regroupe par trois chiffres binaire, à partir de la droite :

$(1\ 100\ 101\ 110\ 111)_2 = (001\ 100\ 101\ 110\ 111)_2 = (?)_8$

$$\underbrace{001}_{1}\ \underbrace{100}_{4}\ \underbrace{101}_{5}\ \underbrace{110}_{6}\ \underbrace{111}_{7}$$

Donc : $(1\ 100\ 101\ 110\ 111)_2 = (14567)_8$

e) Conversion Hexadécimal → Binaire

On code par groupe de 4 chiffres binaires. Chaque chiffre hexadécimal est remplacé par *4* chiffres binaires. (puisque $16 = 2^4$). Nous utilisons le tableau de correspondance suivant :

Base 16	Base 2	Base 16	Base 2
0	0000	8	1000
1	0001	9	1001
2	0010	A=10	1010
3	0011	B=11	1011
4	0100	C=12	1100
5	0101	D=13	1101
6	0110	E=14	1110
7	0111	F=15	1111

$(1A5C0F)_{16} = (\ ?)_2$

En utilisant le tableau ci-dessus, on remplace 5 par 0101, A par 1010 et 1 par 0001, nous aurons :

$1 \rightarrow 0011 \quad A \rightarrow 1010 \quad 5 \rightarrow 0101 \quad C \rightarrow 1100 \quad 0 \rightarrow 0000 \quad F \rightarrow 1111$

$(1A5C0F)_{16} = (0001\ 1010\ 0101\ 1100\ 0000\ 1111)_2 = (11010010111000001111)_2$

f) Conversion Binaire → Hexadécimal

On code par groupe de *4* digits en commençant de droite vers la gauche. Chaque groupe de *4* chiffres binaires est remplacé par 1 chiffres hexadécimal (utiliser le tableau précédent).

$(110100101)_2 = (1\ 1010\ 0101)_2 = (0001\ 1010\ 0101)_2 = (?)_{16}$

$$\underbrace{0001}_{1}\ \underbrace{1010}_{A}\ \underbrace{0101}_{5}$$

$(0001\ 1010\ 0101)_2 = (1A5)_{16}$

I.3.6. Exemple d'application

Réaliser les conversions suivante :

$2023 = (?)_2$

$178 = (?)_{16}$

$2023 = (?)_8$

$(110001110111101011)_2 = (?)_{16}$

$(AC1D)_{16} = (?)_2 = (?)_{10}$

$(20301)_4 = (?)_2$

I.4. Opérations arithmétiques de base

Dans ce qui suit, nous allons voir comment les opérations arithmétiques de base sont réalisées sur les systèmes de numération base 2, 8 et 16.

I.4.1. Sur la base 2

Pour les quatre opérations en base 2 :

x	*y*	*x+y*	*ret.*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

x	*y*	*x-y*	*emp.*
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

x	*y*	*x*y*
0	0	0
0	1	0
1	0	0
1	1	1

x	*y*	*x/y*
0	0	/
0	1	0
1	0	/
1	1	1

Tel-que ***ret.*** est retenu de la somme et **emp.** est l'emprunte de la soustraction (le signe : 0 est positif et 1 est négatif).

Voici quelques cas de figure de réalisation d'opérations arithmétiques sur la base 2 :

Exercice :

Réaliser les opérations suivantes dans la base b=2 :

$(101110)_2+(11001)_2 = (?)_2$

I.5. Codage d'information (CG)

Toute sorte d'information manipulées par un ordinateurs (numériques, textuelles, images, sons, vidéos, etc.) est représentée, sous format binaire, par des séquences de deux chiffres : **0** et **1**. Ces deux chiffres sont désignés par *BIT* (*BInary degiT*).

En informatique, plusieurs codages sont utilisés pour représenter différents types d'information.

I.5.1. Codage pondéré

Dans ce type de codage, chaque bit possède un poids selon son rang, nous avons :

a) Le code binaire pur

Le codage d'information avec le code binaire pur est exactement ce qui a été vue dans la *section I.3.3.* C'est à dire le système de numération de base 2. Ce codage est utilisé pour représenter les valeurs entières positives. En ajouter un bit (le plus à gauche) pour représenter le signe, nous pouvons représenter les valeurs entières relatives (positives et négatives.

Par exemple :

$$\begin{matrix} 5 & 4 & 3 & 2 & 1 & 0 \\ & & & & & \end{matrix}$$

$(110111)_2 = 1 + 2 + 4 + 16 + 32 = 55$

b) Le code BCD (Binary Coded Decimal)

Le codage BCD (en français : DCB : Décimal Codé Binaire) est un code qui premet de coder, en code binaire pur, les dix chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9 sur 4 bits. Chaque 4 bits est calculé avec une pondération de 8421.

Par exemple :

$(735)_{10} = (0111\ 0011\ \ 0101)_2$

Ce codage ressemble à la méthode de conversion de la base 16 vers la base 2.

c) Le code d'Aiken

Le codage d'Aiken est un code binaire pondéré par les poids 2421, il est obtenu comme suit :

Les nombres de 0 jusqu'à 4 sont codés en binaire pur

Les nombres de 5 à 9, on ajoute 6, et puis on code en binaire pur.

Voici le code des chiffres en code Aiken :

0 : 0000	5 : 1011
1 : 0001	6 : 1100
2 : 0010	7 : 1101
3 : 0011	8 : 1110
4 : 0100	9 : 1111

I.5.2. Codage non pondéré

Les bits n'ont pas de poids, il n y a pas une formule de calcule à base de poids de chiffres pour trouver la valeur en base 10. Nous avons :

a) Le code de Gray (binaire réfléchi)

Le code binaire réfléchi ou le code de Gray, dit code cyclique, puisque pour passer d'un nombre au nombre suivant, un seul bit change d'état (de 0 vers 1 ou bien de 1 vers 0). Ce code est dit aussi « code à termes adjacents), il est utilisé dans les tableaux de Karnaugh.

0 : 0000	6 : 0101	12 : 1010
1 : 0001	7 : 0100	13 : 1011
2 : 0011	8 : 1100	14 : 1001
3 : 0010	9 : 1101	15 : 1000
4 : 0110	10 : 1111	
5 : 0111	11 : 1110	

b) Le code alpha-numérique

Pour coder les différents caractères (alphabétiques, numériques, ponctuation, symboles : # @ $ …), nous utilisons des codage binaire sur un certain nombre de bits (7, 8 et 16 bits). Le code le plus utilisé est le code ASCII (American Standard Code for Information Interchange). Le code ASCII de base utilise 7 bits, et le code ASCII étendu, utilise 8 bits.

Le code ASCI sur 7 bits permet de code 128 caractères, et le code ASCII étendu (sur 8 bits), permet de coder 256 caractère. Voici quelques code de caractères :

59 3B 00111011 ; ; Semicolon
60 3C 00111100 < < Less than
61 3D 00111101 = = Equality sign
62 3E 00111110 > > Greater than
63 3F 00111111 ? ? Question mark
64 40 01000000 @ @ At sign
65 41 01000001 A A Capital A
66 42 01000010 B B Capital B
67 43 01000011 C C Capital C
68 44 01000100 D D Capital D
69 45 01000101 E E Capital E
70 46 01000110 F F Capital F
71 47 01000111 G G Capital G
...
117 75 01110101 u u Small u
118 76 01110110 v v Small v
119 77 01110111 w w Small w
120 78 01111000 x x Small x
121 79 01111001 y y Small y
122 7A 01111010 z z Small z

Pour plus d'information, voir ce lien : https://www.rapidtables.com/code/text/ascii-table.html

Bon Courage
&
Travaillez Bien.