Université Paris 13
Institut Galilée
Deug Mias 1ère année
2003-2004

Programmation Impérative
Polycopié de cours n° 1

Enseignants

A. Nazarenko et C. Recanati

Table des matières

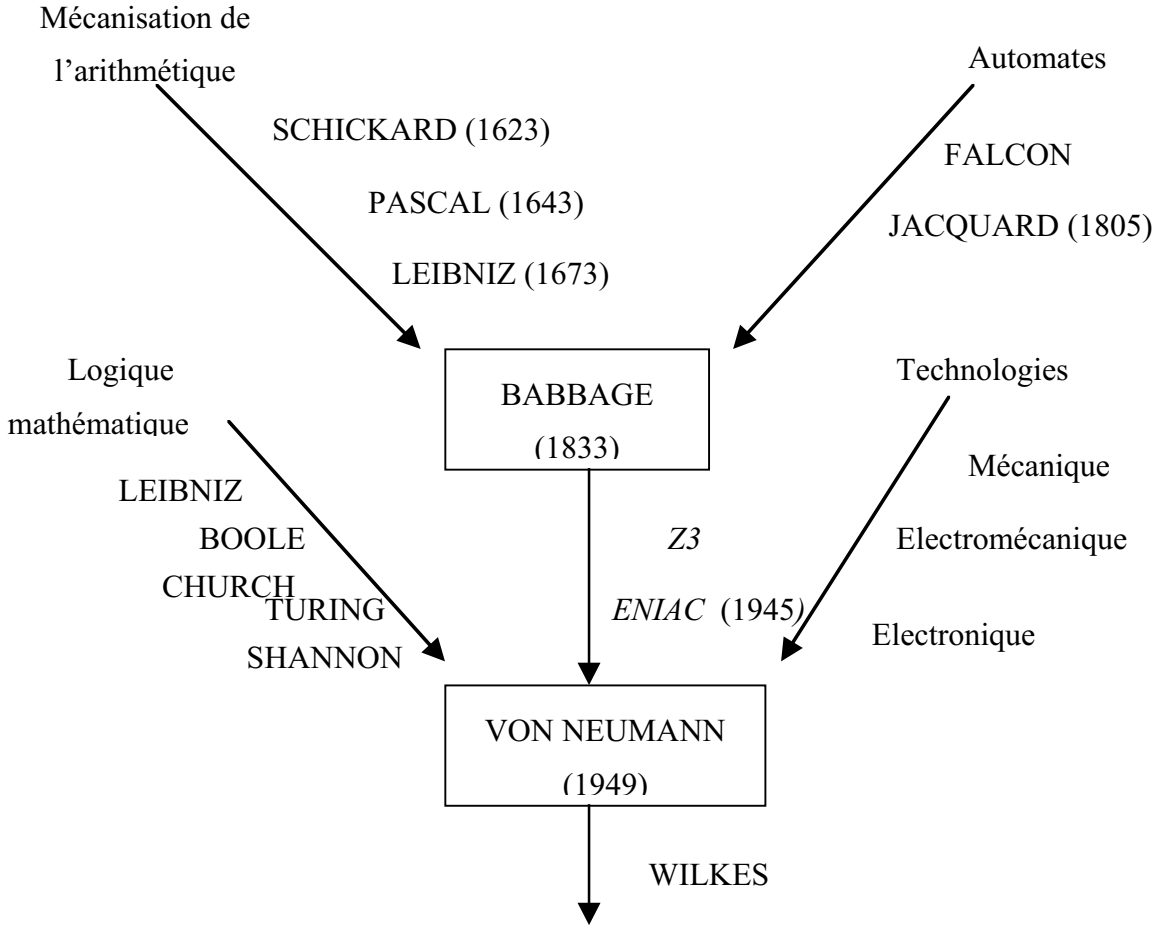
1 Une très brève histoire de l'informatique

L'informatique est la science s'intéressant au traitement automatique de l'information. Le mot « informatique », issu de la contraction des mots information et automatique, n'a été forgé qu'en 1962 : la discipline est récente. Le terme anglais généralement utilisé, *computer science*, science du calcul, renvoie à une histoire beaucoup plus longue, celle du calcul numérique d'abord, étendue ensuite à tout calcul symbolique.

On trouvera ci-dessous quelques éléments d'histoire. Pour plus d'information, on pourra consulter les deux livres indiqués en bibliographie à la fin du chapitre.

1.1 Les racines de l'informatique

On constate que l'informatique est née et a évolué au confluent de plusieurs évolutions scientifiques et techniques. La figure ci-dessous (inspirée de [1]) présente un arbre généalogique des ordinateurs actuels.



1.1.1 Les premières machines à calculer

En 1623, William Schickard inventa la première machine à calculer mécanique.

En 1642, Blaise Pascal invente la Pascaline, machine capable d'effectuer des additions et soustraction, destinée à aider son père: un percepteur de taxes.

En 1673, Gottfried Wilhelm Von Leibniz ajouta à la Pascaline la multiplication et la division. Il est le premier à préconiser l'utilisation du système binaire pour les calculs.

En 1834, Charles Babbage (1792-1871) invente la première machine à calculer mécanique. Cependant il apprend qu'une machine à tisser (métier à tisser Jacquard) est programmée à l'aide de cartes perforées, il se lance donc dans la construction d'une machine à calculer exploitant cette idée révolutionnaire. Aucune de ces deux machines ne fonctionna.

1.1.2 Les ordinateurs programmables électromécaniques

Vers la fin des années 30, Claude Shannon démontra qu'à l'aide d'interrupteurs fermés pour "vrai" et ouverts pour "faux" on pouvait effectuer des opérations logiques (algèbre de Boole) en associant le nombre " 1 " pour "vrai" et "0" pour "faux".

En 1936, Turing propose un modèle formel de calculateur (la machine de Turing).

En 1938, Konrad Zuse (1910-1995) invente le premier ordinateur programmable universel (non spécialisé) qui fonctionne grâce à des relais électromécaniques: le Z3. Cet ordinateur est le premier à utiliser le binaire au lieu du décimal.

En 1937, Howard Aiken met au point un ordinateur programmable mesurant 17m de long et 2.5 mètres de hauteur, permettant de calculer 5 fois plus vite que l'homme : c'est le Mark I d'IBM. Il est alors constitué de 3300 engrenages, 1400 commutateurs reliés par 800km de fil électrique. En 1947, le Mark II voit le jour, ses engrenages sont remplacés par des composants électroniques.

1.1.3 Les ordinateurs programmables électroniques

En 1946, le premier ordinateur ne comportant plus de pièces mécaniques est créé grâce à J.Mauchly et J.Presper Eckert: l'ENIAC (Electronic Numerical Integrator And Computer). Il occupe une place de 1500 m2. Il fut utilisé pour des calculs servant à mettre au point la bombe H. Son principal inconvénient est sa programmation : il est uniquement programmable manuellement avec des commutateurs ou des câbles à enficher. La première erreur informatique est due à un insecte qui, attiré par la chaleur, était venu se loger dans les lampes et avait créé un court-circuit. Ainsi un insecte s'appelant bug en anglais, le nom est resté pour nommer une erreur informatique. En effet, les tubes étant de médiocres conducteurs, ils nécessitaient une grande quantité d'énergie électrique qu'ils dissipaient en chaleur.

Mauchly, Eckert, et John von Neumann (1903-1957) travaillaient à la conception d'un ordinateur électronique, l'EDVAC. Le premier rapport de Von Neumann sur l'EDVAC eut beaucoup d'influence; on y trouve de nombreuses idées encore utilisées dans les ordinateurs les plus modernes.

En Angleterre, Maurice Wilkes construisit en 1948 l'EDSAC (à partir de l'EDVAC) le premier ordinateur à programme en mémoire (architecture dite de Von Neumann).

1.1.4 La naissance de l'informatique

On considère généralement que l'informatique est née en 1946, au moment où sont énoncés, à Princeton aux Etats-Unis, les principes de fonctionnement d'une machine qu'on appellera ensuite ordinateur. On parle de « machine de Von Neumann » parce qu'on en attribue

traditionnellement la paternité à Von Neumann (les plans de cette machine résultaient en fait d'un travail collectif).

Cette machine se distingue des « machines à calculer » antérieures en ce que sa mémoire permettait non seulement de stocker des données et des résultats mais aussi d'enregistrer le programme lui-même. C'est ce qui fait qu'on parlera de « machine universelle » : ce n'est pas une machine dédiée à un type de calcul particulier, on peut modifier le programme pour faire de nouveaux calculs.

Les caractéristiques suivantes de cette machine sont à la base de tous les ordinateurs modernes:

- Machine universelle contrôlée par programme ;
- Instructions et données sont codées sous forme numérique (binaire) et enregistrées en mémoire ;
- Instructions du programme sont normalement exécutées en séquence ;
- Il existe des instructions permettant les ruptures de séquence.

1.2 Les trois grandes périodes de l'informatique

L'histoire de l'informatique est traditionnellement découpée en plusieurs générations (on parle des ordinateurs de 1ère, de 2ème... génération) mais ce découpage reflète davantage l'histoire de composants électroniques que celle de l'informatique considérée dans son ensemble.

On constate en réalité que l'informatique est née et a évolué au confluent de plusieurs évolutions scientifiques et techniques :

- l'invention de la logique binaire par G. Boole à la fin du XIXème siècle;
- l'évolution technique depuis les machines à calculer manuelles ou mécaniques jusqu'aux machines à transistors ;
- le souci du cryptage et le développement de la communication à la fin du XXème siècle et l'émergence de la notion d'information ;
- les travaux sur la calculabilité et l'émergence de la notion d'algorithme (ou de programme enregistré): le traitement d'un problème se ramène à l'exécution d'une séquence finie de calculs arithmétiques ou logiques écrits sous la forme d'un programme ;
- l'histoire et l'évolution des différents langages de programmation.

On retrouve ces différents axes entremêlés dans l'histoire de l'informatique. Nous distinguerons ici trois étapes principales.

1.2.1 La première informatique

Cette période s'étend de 1945 jusque vers le milieu des années soixante. C'est l'époque où les principes fondateurs sont mis en place (codage binaire, algorithme, théorie de l'information) même si cette période est largement centrée sur la machine et ses évolutions. L'essor de

l'informatique est tiré par les marchés militaires. Dans le civil, les besoins sont encore embryonnaires.

1.2.2 La deuxième informatique

Cette période court jusqu'à la fin des années soixante-dix. C'est l'époque où l'informatique se constitue comme discipline. On assiste aux débuts de l'intelligence artificielle, de la traduction automatique. Sur le plan technique, cette période est marquée par les circuits intégrés et l'apparition des mini-ordinateurs à côté des gros ordinateurs scientifiques. C'est également le moment où se développe le marché civil de l'informatique, avec en corollaire la professionnalisation des informaticiens. On voit apparaître les premiers réseaux militaires.

a. Une évolution technique très rapide

En 1948, le transistor est créé par la firme Bell Labs (grâce aux ingénieurs John Bardeen, Walter Brattain et William Shockley). Il permet dans les années 50 de rendre les ordinateurs (**deuxième génération**) moins encombrants, moins gourmands en énergie électrique donc moins coûteux: c'est la révolution dans l'histoire de l'ordinateur!

En 1960, l'IBM 7000 est le premier ordinateur à base de transistor.

Jack Kilby (Texas Instruments) et Robert Noyce (Fairchild Semiconductor) inventèrent les circuits intégrés en 1959. Le circuit intégré permet de réduire encore la taille et le coût des ordinateurs (**troisième génération**) en intégrant sur un même circuit électronique plusieurs transistors sans utiliser de fil électrique.

C'est en 1971 qu'apparaît le premier micro-ordinateur.

En 1971, le premier microprocesseur, l'Intel 4004, voit le jour. En 1973, le processeur 8080 d'Intel garnit les premiers micro-ordinateurs : le Micral et le Altair 8800 (Ce fût le premier ordinateur de Bill Gates...). A la fin de l'année 1973, Intel commercialisait déjà des processeurs 10 fois plus rapides que le précédent (le Intel 8080).

Les années 1970 virent aussi naître les super-ordinateurs. Seymour Cray (né en 1925) conçut le CRAY-1, qui apparut en mars 1976; il pouvait exécuter 160 millions d'opérations par seconde. Le Cray XMP sortit en 1982. Cray Research continue à construire des ordinateurs géants.

Au début des années 80, on arrive à faire des circuits intégrant des centaines de milliers de transistors dans une puce de quelques millimètres carrés. On parle de **quatrième génération**.

b. Naissance de nombreux langages de programmation

Cette période voit naître de nombreux langages de programmation.

On voit apparaître une première famille de langages dans les années 50 :

- John Backus et son équipe écrivent le premier compilateur FORTRAN en avril 1957.
- LISP (List Processing), un langage de traitements de listes pour l'intelligence artificielle, fut inventé par John McCarthy vers 1958.
- Alan Perlis, John Backus, Peter Naur et leurs associés développèrent Algol (Algorithmic Language) en 1959.

Le langage BASIC a été développé vers 1964 par John Kenneny et Thomas Kurtz.

C'est au cours des années 1970 que le système d'exploitation Unix fut développé aux Bell Laboratories par Ken Thompson et Dennis Ritchie. Brian Kernigham et Ritchie développèrent en même temps le langage C, important langage de programmation que nous utiliserons dans le reste de ce cours, et qui servit de base à l'implémentation du système Unix.

On vit ensuite apparaître de nouveau langages, tels que Pascal (inventé par Niklaus Wirth) et Ada (réalisé par une équipe dirigée par Jean Ichbiab).

c. Essor de l'algorithmique (au sens large)

Les années 1960 connurent de nombreuses avancées dans ce domaine :

- Edsger Dijkstra trouva un algorithme efficace pour résoudre le problème des plus courts chemins dans un graphe, à titre de démonstration pour l'ARMAC en 1956. Il trouva aussi un algorithme efficace de recherche d'un arbre afin de minimiser le câblage du X1.
- On voit émerger la théorie des automates et des langages formels (Noam Chomsky et Michael Rabin)
- On commence aussi à utiliser des méthodes formelles pour prouver la correction des programmes. Les travaux de Tony Hoare (l'inventeur de l'algorithme de tri appelé Quicksort) jouèrent un rôle important.
- Donald Knuth, auteur du traité *The Art of Computer Programming*, posa les fondements arithmétiques rigoureux pour l'analyse des algorithmes dans les années 1960.

Il y eut aussi des progrès importants en algorithmique et en théorie de la complexité au cours des années 1970 :

- Les travaux d'Edgar Codd sur les bases de données relationnelles permirent une avancée majeure dans la théorie des bases de données.
- En 1971, Steve Cook publia un article fondamental sur la NP-complétude, et peu après, Richard Karp montra que de nombreux problèmes combinatoires étaient NP-complets.
- Whit Diffie et Martin Hellman publièrent un article fondant la théorie de la cryptographie à clef publique ; le système de cryptage RSA fut inventé par Ronald Rivest, Adi Shamir, et Leonard Adleman.

d. Les prémisses d'Internet

Vers la fin de la décennie des années 60, on commença à construire ARPAnet, un réseau précurseur d'Internet.

1.2.3 La troisième informatique

C'est la période que nous vivons depuis les années 1980. Elle est marquée par de nombreux phénomènes :

- La miniaturisation qui permet à l'informatique d'envahir le quotidien ;

- La généralisation de la culture informatique avec une appropriation individuelle, l'apparition d'amateurs compétents et un souci accru des utilisateurs ;
- La conjonction des mondes de l'informatique et des télécommunications avec notamment l'essor des réseaux. Les ordinateurs peuvent être connectés aux réseaux de télécommunications. Les données sont transmises à grande vitesse par câble, fibre optique, ligne téléphonique ou satellite d'un ordinateur à un autre sur la planète. Ainsi, un grand nombre d'ordinateurs sont reliés entre eux offrant la possibilité de communiquer et de partager un énorme potentiel d'informations (accès à distance à d'autres ordinateurs, partage de fichiers, courrier électronique...). Internet est l'ensemble de tous les réseaux interconnectés.

En 1979, trois étudiants de Caroline du Nord développèrent un serveur de nouvelles distribué qui finalement devint Usenet.

La décennie 1980 vit apparaître le micro-ordinateur personnel, grâce à Steve Wozniak et Steve Jobs, fondateurs de Apple Computer. En 1981, l'Osbonne 1 fut le premier ordinateur vraiment portable. En 1984, Apple commercialisa le MacIntosh, avec un nouveau système d'exploitation d'interface conviviale fondée sur le multi-fenêtrage.

En 1987, l'US National Science Fondation démarra NSFnet, qui devait devenir une partie de l'Internet actuel.

Parallèlement, le système de fenêtrage X-Window destiné aux machines Unix, prend sa source en 1985 à partir d'un système de fenêtrage appelé W développé à l'université Stanford en 1982. La version 11 (encore utilisée aujourd'hui) sort en 1988 et permet aux constructeurs de machines d'implanter des interfaces graphiques portables.

1.3 Les ordinateurs aujourd'hui : utilisation et environnement matériel

Le terme ordinateur recouvre des types de machines très différentes les unes des autres. De même, pour chaque type de machine, on peut concevoir diverses utilisations, chaque utilisation pouvant à son tour nécessiter des types de logiciels distincts.

1.3.1 Types d'utilisation

Voici quelques grands types d'utilisation des ordinateurs :

- Bureautique (rédaction de document, feuilles de calculs, dessin …)
- Communication (messagerie, échange de données à distance, navigation sur internet…)
- Loisirs (jeux, dessin…)
- Programmation
- Gestion (de stocks, comptabilité…)
- Calculs numériques (météo, simulation de systèmes physiques comme des voitures…, résolution d'équations, traitements statistiques de données…)
- Résolution de problèmes numérique (optimisation d'un parcours de livraison…) ou symbolique (emploi du temps, preuve automatique, traduction automatique…)

1.3.2 Types de logiciels

Chacune de ces utilisations fait appel à un logiciel ou à une classe de logiciels particuliers. Nous donnons ici en exemple quelques grandes classes de logiciels :

- Outils de bureautique : éditeur et traitement de texte permettant l'édition et la mise en page de documents, tableur pour faire des feuilles de calculs, logiciel de dessin
- Outils de gestion d'informations permettant le stockage des données, le traitement statistique de celles-ci et la recherche d'information dans ces bases de données (systèmes de gestion de bases de données).
- Interface graphique : programme permettant l'affichage ergonomique des informations.
- Système d'exploitation : ensemble de programmes gérant les éléments physiques de la machine et aidant à l'exécution des programmes par un ordinateur.
- Compilateur : programme permettant la traduction en langage machine (directement exécutable par la machine) de programmes écrits en langage de haut niveau (compréhensible et manipulable « simplement » par un humain).

Naturellement, nous ne sommes généralement pas conscients, quand nous utilisons un ordinateur, du nombre de logiciels auxquels nous faisons appel.

1.3.3 Types de machines

L'utilisation ou l'application que l'on vise détermine le choix des logiciels mais aussi le type de matériel à utiliser. On distingue notamment les types d'ordinateurs suivants :

- Puce à programme fixe : la puce est physiquement « programmée » pour exécuter un petit ensemble d'actions (dans une machine à laver, par exemple) ;
- Calculatrice : elle permet de traiter des opérations arithmétiques élémentaires ;
- **Calculatrice programmable** : elle permet de calculer des fonctions numériques complexes ;
- **Micro-ordinateur** : machine ayant l'architecture et les composants d'un ordinateur et généralement dédiée à la bureautique et aux calculs numériques ;
- **Ordinateur individuel / station de travail** : machine adaptée à des calculs numériques puissants ou à des simulations graphiques conséquentes ;
- Super-calculateur : machine dédiée à des calculs numériques très importants.

1.3.4 Description d'un ordinateur

Ce qui est dit dans ce cours se rapporte plus particulièrement aux types de machines écrits en caractères gras dans la liste précédente.

La configuration de base d'un ordinateur comprend :

- Une Unité centrale (UC) contenant le processeur, c'est-à-dire le "moteur" de l'ordinateur qui exécute les instructions, et, la mémoire centrale qui contient les programmes en exécution ou prêts à s'exécuter et les données à traiter.

– Un écran (aussi appelé moniteur),

– Un clavier,

De plus en plus souvent les ordinateurs se voient dotés d'une *souris*, qui permet à l'utilisateur de transmettre certaines commandes à l'ordinateur de façon plus ergonomique.

Les composants précédents peuvent se présenter sous forme d'éléments distincts aussi bien qu'en un seul bloc (comme dans un ordinateur portable par exemple).

Suivant le type d'utilisation de la machine, d'autres éléments viennent compléter la configuration de base précédente. Citons, entre autres : une imprimante, un scanner, un modem, différents périphériques de stockage.

[1] Architecture et technologie des ordinateurs, P. Zanella, Y. Ligier, Dunod, 1998.

[2] Ainsi naquit l'informatique – histoire des hommes et des techniques, R. Moreau, Dunod, 1987.

2 Codage des données

Un ordinateur fonctionne grâce à des signaux électriques. Il en comprend deux : « marche » symbolisé par 1 et « arrêt » par 0. En conséquence, l'alphabet d'un ordinateur se compose de deux lettres 1 et 0 sur la base desquelles il faut représenter à la fois les données stockées en machine et les instructions exécutées par la machine. Données (numériques, textuelles, sonores, images animées ...) et instructions doivent donc être traduites en mots composés uniquement de 1 et de 0.

2.1 Changement de base

Nous avons l'habitude de représenter les nombres en base 10. Cependant, cette représentation n'est qu'une des multiples représentations possibles.

2.1.1 Codage binaire

a. Définition

Tout entier positif ou nul n peut être codé en base 2 de manière unique par le codage $a_k a_{k-1} a_{k-2} ... a_1 a_0$ quand les a_i sont des entiers qui vérifient :

- $a_i \in \{0,1\}\ \forall i=0,1,\ldots,k$
- $n= a_k 2^k + a_{k-1} 2^{k-1} + a_{k-2} 2^{k-2} + ... a_1 2^1 + a_0 2^0$

Par convention, lorsqu'il y a risque de confusion sur la base utilisée, on écrit la valeur de la base en indice comme par exemple 101_2 (= 5) et 101_{10} (= 101).

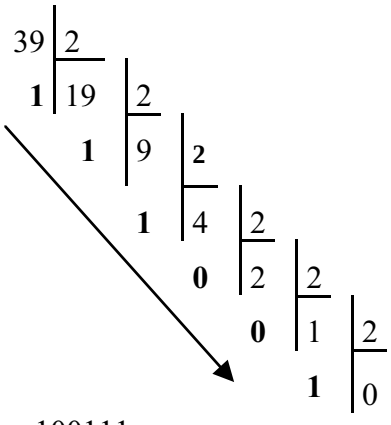
b. Conversion binaire↔ décimal

Les conversions de binaires à décimal ne posent pas de problème puisqu'il suffit de faire une somme. Par exemple :

$100111_2 = 1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 1.2^0 = 32 + 4 + 2 + 1 = 39$

Pour obtenir le codage en base 2 d'un nombre codé en base 10, on utilise la méthode des divisions successives par 2.

Exemple



$39_{10} = 100111_2$

Pour obtenir le codage en base 2, il suffit d'effectuer des divisions successives par 2 jusqu'à ce que le quotient soit égal à 0. Le premier reste obtenu est le chiffre de poids faible (le plus à droite) du codage en base 2, le dernier reste obtenu est le chiffre de poids fort (le plus à gauche).

2.1.2 Codage octal et hexadécimal

Les codages octal (base 8) et hexadécimal (base 16) sont aussi des codages très utilisés en informatique.

En base 2, l'alphabet est constitué de deux symboles : 1 et 0. En base 10, l'alphabet se compose de 10 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. De façon générale, nous avons besoin de *b* symboles distincts pour coder de façon unique un nombre en base *b*.

En base 8, l'alphabet est composé de 8 symboles : 0, 1, 2, 3, 4, 5, 6 et 7.

En base 16, l'alphabet est composé de 16 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F avec :

$A_{16} = 10_{10}$

$B_{16} = 11_{10}$

$C_{16} = 12_{10}$

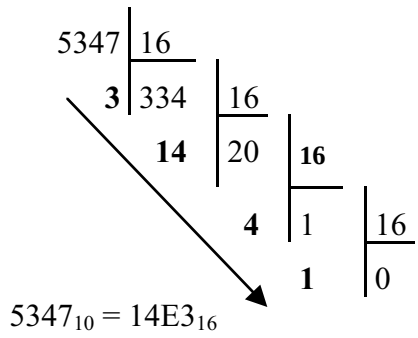
$D_{16} = 13_{10}$

$E_{16} = 14_{10}$

$F_{16} = 15_{10}$

Les conversions de base 10 en base 8 ou 16 se font en utilisant la même méthode que pour les conversions de base 10 en base 2, c'est-à-dire avec des divisions successives par 8 ou 16. Les restes de ces divisions successives forment la représentation en base 8 ou 16.

Exemple



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

Les conversions entre la base 2 et les bases 8 et 16 sont très faciles à réaliser. Par exemple, on dispose du codage en base 2 d'un entier *n*, codage comportant 8 éléments a_i :

$n_2 = a_7a_6a_5a_4a_3a_2a_1a_0$

$n_{10} = a_7 2^7 + a_6 2^6 + a_5 2^5 + a_4 2^4 + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$

$n_{10} = (a_7 2 + a_6)8^2 + (a_5 2^2 + a_4 2 + a_3)8^1 + (a_2 2^2 + a_1 2^1 + a_0 2^0)8^0$

$n_8 = (a_7 2 + a_6)(a_5 2^2 + a_4 2 + a_3)(a_2 2^2 + a_1 2^1 + a_0 2^0)$

Tous les chiffres de ce codage en base 8 sont bien inférieurs à 8 (puisque la valeur maximale est $2^2 + 2^1 + 2^0 = 7$).

Exemple : Conversion de 11110110101_2 en base 8

On regroupe les chiffres du codage en base 2 par paquets de 3 ($2^3 = 8$) en allant de droite à gauche. Le dernier paquet peut évidemment contenir moins de 3 chiffres.
$11110110101_2 = 11\ 110\ 110\ 101_2$
Pour obtenir le codage en base 8, on remplace chaque paquet par la valeur en base 8 qui lui correspond.
$11\ 110\ 110\ 101_2 = 3\ 6\ 6\ 5 = 3665_8$

Exemple : Conversion de 11110110101_2 en base 16

On regroupe les chiffres du codage en base 2 par paquet de 4 ($2^4 = 16$) en allant de droite à gauche. Le dernier paquet peut évidemment contenir moins de 4 chiffres.
$11110110101_2 = 111\ 1011\ 0101_2$
Pour obtenir le codage en base 16, on remplace chaque paquet par la valeur en base 16 qui lui correspond.
$111\ 1011\ 0101_2 = 7\ B\ 5 = 7B5_{16}$

2.1.3 Autres codages

De façon générale, tout entier positif ou nul n peut être codé en base b de manière unique par le codage $a_k a_{k-1} a_{k-2} ... a_1 a_0$ quand les a_i vérifient :

- a_i est un symbole de l'alphabet du codage en base b
- $n = \sum_{i=0}^{k} a_i b^i$

2.1.4 L'addition avec un codage en base *b*

Pour additionner deux nombres codés en base b, on additionne un à un chacun des chiffres du codage de droite à gauche et, on fait une retenue dès que la somme des deux chiffres est supérieure ou égale à b (en base 10 quand la somme vaut 10, en base 2 quand la somme vaut $10_2 = 2_{10}$). En base 2, pour 1 + 1 qui vaut 10_2, on pose le 0 de $1\underline{0}_2$ et on fait une retenue correspondant au 1 de $\underline{1}0_2$.

Exemple : Addition de 14 et 19 codés en base 2

```
 1111
  1110 = 14₁₀
 10011 = 19₁₀
100001 = 33₁₀
```

2.2 Représentation des données en machine

On doit pouvoir représenter en machine des données tant numériques (entiers, réels ...) que non numériques. Pour cela, une donnée doit être écrite comme une suite de 0 et de 1. En machine, un chiffre binaire (0 ou 1) est stocké sur 1 bit (Binary Digit). On appelle **mot mémoire** une succession de bits possédant une adresse mémoire unique. Par exemple, un **octet** (*byte* en anglais) est un mot mémoire de 8 bits. Une donnée est stockée sur un mot mémoire. Or, la taille d'un mot mémoire étant limitée, il faut donc écrire chaque donnée comme une suite de taille fixée de 0 et de 1.

2.2.1 Représentation des entiers

L'ensemble des entiers est un ensemble infini. Il est donc impossible de le représenter en machine dans sa totalité. Le nombre d'entiers qu'il est possible de représenter dépend de la taille du mot mémoire choisi pour le coder.

a. Entiers naturels (non signés)

Un entier naturel, ou non signé, est représenté en machine dans un mot mémoire par son codage binaire.

Par exemple, si on stocke en mémoire les entiers non signés dans des mots mémoire contenant 8 bits (= un octet), 45_{10} (= 101101_2) est représenté par le mot 00101101. Par contre, dans ce cas, 300_{10} dont le codage en base 2 est 100101100_2 ne peut être représenté en machine dans un mot mémoire de taille 8.

La seule difficulté est donc de s'assurer que l'entier que l'on veut représenter peut être codé sur un nombre fixé de bits.

Le plus grand entier non signé représentable sur *p* bits est 2^p-1.

En effet, le plus grand entier représentable sur *p* bits est celui dont le codage n'est constitué que de 1. Par exemple pour $p = 4$:

$1111_2 = 15_{10} = 10000_2 - 1 = 2^4 - 1$

Le plus petit entier représentable sur *p* bits est 0 (son codage n'est constitué que de 0). En conséquence, **le nombre de valeurs distinctes représentables sur *p* bits est 2^p**.

Aussi, si on choisit de représenter sur 1 octet des entiers non signés, on peut coder 256 valeurs entières distinctes, celles comprises entre 0 et 255.

b. Entiers relatifs (signés)

Lorsqu'on code des entiers relatifs (on parle alors d'entiers signés), il faut représenter le signe.

Un codage possible est celui qui consiste à prendre le bit de poids fort comme le bit de signe (qui vaut 1 si le nombre est négatif et 0 sinon) et de représenter la valeur absolue de l'entier.

Si on code les entiers relatifs sur 3 bits avec cette convention, on obtient :

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

Le problème ici est, entre autres, que le 0 admet deux représentations distinctes !

Pour qu'une représentation soit satisfaisante, il faut d'une part que la machine puisse tester très simplement le signe de l'entier et, d'autre part, que la représentation choisie ne complique pas la mise en œuvre des opérations arithmétiques : les circuits réalisant les opérations sur les entiers non signés doivent pouvoir servir à effectuer les opérations sur les entiers signés. Le codage qui semble le mieux satisfaire à ces deux exigences est le **codage en complément à deux** : c'est le codage qui est quasiment universellement utilisé.

Si on code les entiers relatifs en complément à 2 sur 3 bits on obtient :

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

Les caractéristiques du codage en complément à 2 sont :

- Le bit le plus à gauche dans le mot mémoire (appelé bit de signe ou bit de poids fort) représente le signe : 0 pour un entier positif, 1 pour un entier négatif,
- Si on note rep(x) la représentation en complément à deux de l'entier signé x sur p bits, on a :

  si $x \geq 0$, rep(x) = x

  si $x < 0$, rep(x) = 2^p - $|x|$ = 2^{p-1} (bit de signe à 1) + 2^{p-1} - $|x|$

Ainsi, si on code les entiers signés sur p bits, on aura :

- Les entiers compris entre 0 et 2^{p-1}-1 sont représentés en binaire sur les (p-1) bits de droite, le bit de signe valant 0,
- Les entiers compris entre -2^{p-1} et -1 sont représentés comme suit :

$1\ \mathbf{0}\ \ldots\ 0\ 0 = -2^{p-1}$

$1\ 0\ \ldots\ 0\ 1 = -2^{p-1}+1$

$1\ 0\ \ldots\ 1\ 1 = -2^{p-1}+2$

...

$1\ 1\ \ldots\ 1\ 1 = -1$

En conséquence, dans un mot mémoire composé de p bits, on peut représenter 2^p entiers signés distincts variant de -2^{p-1} à 2^{p-1}-1. Par exemple, dans 1 octet (8 bits), on peut représenter 256 valeurs distinctes variant de -128 à 127.

Voici une première méthode pour coder un entier $x < 0$ sur p bits :

1. calculer la représentation binaire de $|x|$ sur p bits,
2. complémenter tous les bits de cette représentation binaire, c'est-à-dire inverser chacun des bits,
3. ajouter 1.

Le codage ainsi obtenu est la représentation en complément à deux de x.

Les phases 2 et 3 permettent de représenter l'opposé (ou le complément à 2) du nombre codé en binaire.

Exemple : représentation en complément à deux de -2 sur un mot mémoire de 4 bits

La représentation binaire de 2 sur 4 bits est 0010.
On complémente les 4 bits et on obtient : 1101.
Enfin, on ajoute 1 : 1101 + 0001 = 1110.
On a donc : $-2_{10} = 1110_2$.

Une seconde méthode pour coder un entier $x < 0$ sur p bits est :

1. calculer la représentation binaire de $|x|$ sur p bits,
2. parcourir la chaîne de bits de droite à gauche jusqu'au premier bit à 1,
3. inverser tous les bits à gauche du bit trouvé (ce bit ne doit pas être modifié).

Exemple : représentation en complément à deux de -2 sur un mot mémoire de 4 bits

La représentation binaire de 2 sur 4 bits est 0010.

On complémente tous les bits de droite à gauche à partir du premier bit de valeur 1et on obtient : 1110.
On a donc : $-2_{10} = 1110_2$.

Exemple : représentation en complément à deux de -200 sur un mot mémoire de 8 bits

La représentation binaire de 200 sur 8 bits est 11001000.
On complémente tous les bits de droite à gauche à partir du premier bit de valeur 1 et on obtient : 00111000.
Or, 00111000 ne peut pas être la représentation en complément à 2 de -200 car le bit de signe est positif. Le problème provient simplement du fait que -200 ne peut pas être codé dans un mot mémoire de 8 bits !

Posons-nous maintenant le problème inverse de retrouver la valeur d'un entier signé x à partir de son codage. Deux cas doivent être envisagés :

- soit le bit de signe vaut 0 et il s'agit d'un entier positif. Il suffit donc d'effectuer une conversion binaire vers décimal standard ($\text{rep}(x) = x$),
- soit le bit de signe vaut 1 et il s'agit d'un entier négatif. Sa valeur s'obtient en prenant son opposé (= son complément à deux), puis en effectuant une conversion binaire vers décimal classique et en ajoutant le signe - devant la valeur obtenue.

Exemple : valeur de 10011011

Le complément à 2 est : 01100101. Et $01100101_2 = 2^6 + 2^5 + 2^2 + 2^0 = 101_{10}$.
En conséquence, la valeur de 10011011_2 est -101_{10}.

Dans le cas où le bit de signe vaut 1, on peut directement calculer la valeur décimale de la représentation en complément à deux pour obtenir x comme suit :

$x = -2^p + \text{rep}(x)$ (car on a par définition : $\text{rep}(x) = 2^p - |x| = 2^p + x$ car $x<0$).

Exemple : valeur de 10011011

$10011011_2 = 2^7 + 2^4 + 2^3 + 2^1 + 2^0$
$x = -2^8 + 2^7 + 2^4 + 2^3 + 2^1 + 2^0 = -2^7 + 2^4 + 2^3 + 2^1 + 2^0 = -101_{10}$.

c. Opérations arithmétiques sur les entiers

L'un des grands avantages de la notation en complément à deux est qu'elle permet d'additionner des entiers de n'importe quel signe en traitant leurs représentation comme des nombres binaires positifs.

Exemple : Addition de 12 et -19.

On choisit de représenter ces valeurs sur 6 bits.
$12_{10} = 001100_2$
$-19_{10} = 101101_2$
L'addition s'effectue bit par bit, de droite à gauche en propageant les retenues vers la gauche.

```
   1
001100
101101
------
111001 = -2^5 + 2^4 + 2^3 + 1 = -7_10
```

Lorsqu'on additionne deux valeurs entières signées x et y représentées en complément à deux sur p bits, trois cas peuvent se produire :

- x et y positifs : rep(x) = x et rep(y) = y

  $\Rightarrow$ rep(x) + rep(y) = $x + y$ = rep($x + y$)

- x et y négatifs : rep(x) = $2^p + x$ et rep(y) = $2^p + y$

  $\Rightarrow$ rep(x) + rep(y) = $2^p + x + 2^p + y \neq$ rep($x + y$) = $2^p + x + y$ (car $x + y \leq 0$)

  $\Rightarrow$ rep($x + y$) = rep(x) + rep(y) - 2^p

- x positif et y négatif : rep(x) + rep(y) = $2^p + x + y$

  si $x + y < 0 \Rightarrow$ rep(x) + rep(y) = rep(x + y)

  si $x + y \geq 0 \Rightarrow$ rep($x + y$) = $x + y$ = rep(x) + rep(y) - 2^p

Exemple : Addition de -12 et -19

```
  111
  110100 = -12₁₀
 +101101 = -19₁₀
 -------
 1100001
```

Il faut donc éliminer la retenue se trouvant le plus à gauche et ne retenir que les 6 bits de droite significatifs :
$-12_{10} + -19_{10} = 100001_2 = -31_{10}$

Exemple : Addition de 14 et 19

```
  1111
  001110 = 14₁₀
+ 010011 = 19₁₀
 -------
  100001 = -31₁₀
```

Le problème ici est que nous avons un dépassement de capacité car 33_{10} n'est pas représentable sur 6 bits. Sur 6 bits, on peut représenter 64 valeurs distinctes variant de -32 à 31.

Un dépassement de capacité ne peut pas subvenir lorsqu'on ajoute des nombres de signes différents. Lorsqu'on additionne des nombres de même signe représentés en complément à deux, il y a dépassement de capacité si et seulement si le résultat est de signe différent.

2.2.2 Représentations des caractères

Les données non numériques correspondent aux caractères

- Alphanumériques : A, B, …, Z, …, 0, 1, …, 9
- Spéciaux : ?, !, « , $ …

La principale contrainte concernant les caractères est l'ordre alphabétique. Le caractère A doit précéder le caractère B…

Plusieurs codifications des caractères ont été proposées :

- BCD (Binary Coded Decimal) un caractère est codé sur 6 bits,
- EBCDIC (Extended Binary Coded Decimal Internal Code) sur 8 bits
- ASCII (American Standard Code for Information Interchange) sur 7 bits
- ISO (International Standard Organisation) sur 8 bits, c'est un ASCII étendu,

- UNICODE sur 16 bits, qui contient actuellement 38887 caractères de différentes langues à travers le monde,
- ISO/IEC sur 32 bits, ce codage constitue une extension de UNICODE.

Les deux derniers codes sont les plus récents (début des années 1990). En effet, $2^7 = 128$ ou $2^8 = 256$ valeurs ne suffisent pas pour représenter l'ensemble de tous les caractères de toutes les langues de la planète. Or le UNICODE permet de coder $2^{16} = 65536$ caractères différents.

Cependant, c'est le code ASCII qui est, pour l'instant, adopté de façon quasi universelle. Ses principales caractéristiques sont :

- Les caractères sont codés dans des mots-mémoire de 7 bits (ce qui permet de coder 128 symboles différents).
- Les 32 premiers codes (codes hexadécimaux de 00 à 1F) représentent des caractères de contrôle qui ne sont pas affichables (par exemple émettre un bip sonore, passer à la ligne …).
- Les lettres se suivent dans l'ordre alphabétique : codes 65 à 90 pour les majuscules,97 à 122 pour les minuscules).
- Les caractères numériques sont rangés dans l'ordre croissant (codes 48 à 57).

Remarque : Ce code étant à l'origine fait pour les besoins de l'informatique en langue anglaise, il ne permet pas de coder les caractères accentués du français, de même que les caractères spéciaux d'autres langues comme l'arabe ou le chinois par exemple.

La longueur standard minimale des mots traités par les machines actuelles étant de 8 bits, des extensions du code ASCII, utilisant le $8^{ième}$ bit, ont été proposées afin de pouvoir coder des caractères spécifiques d'autres langues, mais elles ne sont pas universellement reconnues.

Code ASCII (Hexadecimal - Caractère)

00 NUL	01 SOH	02 STX	03 ETX	04 EOT	05 ENQ	06 ACK	07 BEL
08 BS	09 HT	0A NL	0B VT	0C NP	0D CR	0E SO	0F SI
10 DLE	11 DC1	12 DC2	13 DC3	14 DC4	15 NAK	16 SYN	17 ETB
18 CAN	19 EM	1A SUB	1B ESC	1C FS	1D GS	1E RS	1F US
20 SP	21 !	22 "	23 #	24 $	25 %	26 &	27 '
28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E ~	7F DEL