## Séquence 1 - Numérisation - Encodage des nombres

# I. Numérisation de l'information

## I. 1. Définition de la numérisation - Histoire

**Définition 1 (Numérisation (ou encodage binaire))**

La numérisation est l'opération permettant de transformer et stocker une donnée (nombre, texte, son, image, vidéo...etc) en une série d'informations très simples à deux états (d'où le qualificatif de BI-naire). On note souvent 0 et 1 ces deux états.
Une fois numérisée, une donnée peut être stockée et manipulée de manière beaucoup plus aisée, en particulier par une machine programmable.

**Exemple 1 (Le code Morse)**

Ce code associe à chaque lettre et chaque chiffre un code composé d'un assemblage de signaux longs et courts. Il s'agit donc bien de numérisation car l'information d'un message est transformée en une série de signaux binaires dont les deux états sont long/court. Ces signaux longs/courts sont associés à des impulsions électriques dans le cas du télégraphe, à des signaux lumineux plus ou moins longs dans le cas de la navigation maritime et même à des clignements d'yeux !
Allez voir cette vidéo pour en savoir plus :
`https://www.youtube.com/watch?v=DhnS0agi_kI`

**Exemple 2 (Cartes perforées pour encoder de la musique)**

L'orgue de barbarie est un instrument de musique capable de jouer automatiquement de la musique en lien avec une carte perforée encodant une mélodie. Il s'agit bien d'une des premières numérisations de la musique puisque qu'une mélodie est transformée en série de signaux binaires à deux états troué/non troué.
Cliquez sur le lien suivant pour découvrir cet incroyable objet :
`https://www.youtube.com/watch?v=w5TqIAPdW_I`

**Exemple 3 (Cartes perforées pour les métiers à tisser)**

Le métier à tisser Jacquard a fasciné les pionniers de l'informatique. Il s'agit d'une machine programmable, puisque le tissage réalisé par la machine suivra un patron de tissage encodé sur des cartes en papier perforées. La manière dont la carte est perforée permet de guider la machine sur la manière de tisser.
Cliquez sur le lien suivant pour découvrir cette invention révolutionnaire... qui a bouleversé le monde ouvrier et sa sociologie :
`https://www.youtube.com/watch?v=eE5wxtaIcEY`

**Remarque.** Pour les données issues de phénomènes physiques (son, image, paramètres environnementaux tels que la température ou la pression notamment) on oppose souvent captation **analogique** et captation **numérique**.

**La captation d'une donnée physique sous forme analogique** repose sur un dispositif qui permet de suivre, de façon continue et *analogue*, l'évolution d'une grandeur physique : aiguille associée au mouvement d'une membrane vibrante pour la captation du son, niveau de mercure suivant la température dans un thermomètre, particules de bromure d'argent réagissant à la lumière dans le cas d'une photographie argentique, sonde Pitot mesurant la pression en suivant la vitesse d'écoulement d'un flux d'air...

**La captation d'une donnée physique sous forme numérique** repose sur un dispositif d'échantillonnage qui va sélectionner des instants de captation de la donnée, quantifier la donnée captée et reconstituer par de savants calculs de correction et d'interpolation la donnée cible.

Les deux types de captation ont leur intérêt, mais aujourd'hui, la captation numérique domine de plus en plus.

| Analogique | Numérique |
|---|---|
| Signal capté continu | Signal capté discret |
| Signal directement issu de la réalité physique | Signal reconstruit, synthétisé |
| Erreurs de captation (latences) | Erreurs de reconstruction (choix d'échantillonnage) |
| Donnée difficile à traiter par une machine | Donnée standardisée, facile à traiter |

# I. 2. Supports physique de stockage des données

Le stockage des données est réalisé d'un point de vue physique de différentes manières. Certains supports de données listés ci-dessous peuvent accueillir au choix des données numériques ou analogiques.

## I. 2. a. Classification des différents supports de stockage.

On peut classifier les différents supports de stockage selon les critères suivants :

**Mémoire de masse ou mémoire de travail.** Certains supports sont destinés à stocker des petites quantités d'information de manière temporaire notamment lors de l'exécution d'un programme. Il s'agit de mémoire de travail, situées à proximité du processeur, ayant une capacité limitée mais un temps d'accès très court. Ce sont souvent des mémoires électroniques. A l'inverse, certains supports de stockage sont destinés à stocker de gros volumes d'information sur la durée. On parle alors de mémoires de masse. C'est le cas des disques magnétiques, des disques SSD, des DVDs par exemple.

**Mémoire volatile ou non-volatile.** Certains supports de stockage doivent être alimentés de manière continue en électricité pour que la donnée stockée subsiste. On parle alors de **mémoire vive ou de mémoire volatile** : les informations stockées disparaissent lorsque l'alimentation électrique cesse. C'est le cas de la RAM ou des registres du micro-processeur. A l'inverse, certaines mémoires sont mortes : la donnée encodée sur ce support subsiste en l'absence de toute alimentation. C'est le cas des disques durs, des CDs et DVDs, des clés USB...

**Mémoire à accès direct ou séquentiel.** Les mémoires magnétiques et optiques sont

à accès séquentiel : il faut que la tête parcours un certain chemin pour atteindre la donnée souhaitée. Les mémoires électroniques sont à accès direct : une adresse mémoire va permettre, en irrigant certains connecteurs du circuit imprimé, de rapatrier directement la donnée souhaitée.

**Mémoire RO, WO, RW ou reprogrammable.** Une mémoire peut seulement être accessible en lecture (Read Only RO), seulement en écriture (Write Only WO), accessible en lecture et en écriture (RW). Elle peut ausi être réinscriptible mais de façon globale. C'est le cas de la ROM. On parle dans ce dernier cas de mémoire reprogrammable.

**Support amovible ou non.** Certains supports de stockage sont liés physiquement à la machine qui les utilise pour lire ou écrire des données. C'est généralement le cas du disque dur (sauf pour les disques durs externes). D'autres supports peuvent être transportés, échangés : c'est le cas des K-7, VHS, CDs, DVDs, des clés USB...

## I. 2. b. Critères de choix d'un support de stockage.

Le choix d'un support de stockage dépend de plusieurs facteurs, et en particulier de son usage :

**Temps d'accès en lecture/écriture :** extrêmement rapide pour certaines mémoires électroniques, très lent pour certaines bandes magnétiques. Le disque dur à mémoire flash (SSD) a des temps d'accès très réduits par rapport au disque dur magnétique (HDD). De façon générale, les mémoires à accès électroniques à accès direct ont des temps d'accès et de latence beaucoup plus faibles.

**Capacité de stockage :** extrêmement grande pour certains disques durs ou SSD, quelques octets pour certaines mémoires électroniques utilisés comme registres du microprocesseur

**Sécurité des données :** les incidents matériels sont beaucoup plus fréquents avec les disques durs magnétiques qu'avec les SSD

**Coût :** très peu élevé pour certaines bandes magnétiques, encore très élevé pour la mémoire flash

**Durée de vie :** une dizaine d'année pour les disques durs et les mémoires flashs (SSD ou clé USB) pour un usage grand public, 20 ans pour un CD-RW conservé dans de bonnes conditions, 20 à 30 ans pour une bande magnétique

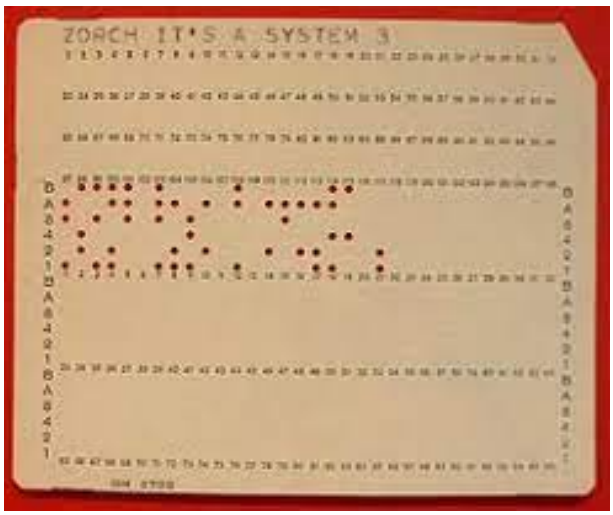
**Empreinte écologique :** les nouvelles RAM (DDR4 et 5) sont conçues pour limiter la consommation énergétique

**Disponibilité mondiale, enjeux géostratégiques...**

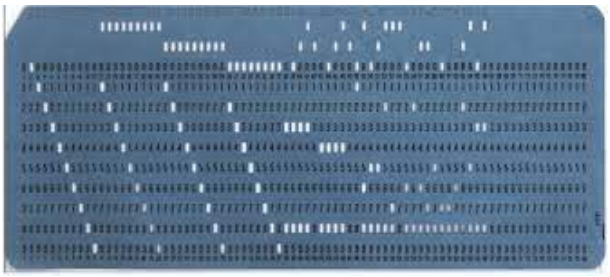
## I. 2. c. Supports papier

L'idée d'encoder des données en binaire par les deux états troué/non troué sur des supports cartonnés date du XVIII siècle. Cette méthode a été utilisée dans l'industrie textile pour programmer les métiers à tisser, puis pour programmer des machines de traitement de gros volumes de données (données démographiques, bancaires), et ceci jusque dans les années 80 !
Je vous invite à visionner les deux vidéos ci-dessous pour en apprendre davantage sur l'utilisation des cartes perforées :

— Les utilisations de la carte perforée depuis le XVIIIème siècle :
`https://www.youtube.com/watch?v=fOgONLPtfLQ`




(a) Carte perforée 48 colonnes à trous ronds type Samas



(b) Carte perforée 80 colonnes IBM

FIGURE 1 – Deux modèles historiques de cartes perforées.

— Un focus sur la trieuse automatique, dans une ambiance délicieusement vintage des années 60 :

`https://www.ina.fr/ina-eclaire-actu/video/caf97059686/la-carte-perforee`

Certains vestiges de cette époque se retrouvent encore aujourd'hui dans les limitations de certains langages de programmation (contrainte de 80 caractères par ligne en langage Fortran 77 par exemple)

## I. 2. d. Supports magnétiques

**Différents supports magnétiques.** La mémoire à tores magnétiques est l'ancêtre du disque dur et a été utilisé à partir des années 50 et jusqu'au milieu des années 70. Les supports magnétiques sous forme de bandes ont ensuite servi à stocker des signaux analogiques (son puis vidéo). Vous avez certainement déjà vu une **K7 audio** ou une **VHS** (vidéo), supports utilisés jusqu'à la fin des années 2000.
Concernant les données numériques, les **bandes magnétiques** sont utilisées pour le stockage à long terme des données numériques car la durée de vie d'une bande peut aller jusqu'à 30 ans dans de bonnes conditions de conservation. Les bandes magnétiques sont encore très utilisées de nos jours pour le stockage de données sensibles comme les données bancaires. Jusque dans les années 2000, la **disquette**[1] était également un support de stockage magnétique très utilisé.

1. appelée *floppy* en anglais, car il s'agissait d'un disque magnétique *souple*

Mais le support magnétique le plus utilisé par le grand public pour le stockage des données numériques est sans conteste le **disque dur magnétique**, qui tend d'ailleurs à disparaître depuis quelques années au profit des SSD (*Solid State Drive*) dont le principe physique est différent, voir Section **??**.

(a) K7 audio


(b) Cassette VHS vidéo

FIGURE 2 – Deux exemples de supports magnétiques utilisés pour les signaux analogiques audio et vidéo

**Principe général de fonctionnement.** Le principe de la bande magnétique date de la fin du XIXème siècle. Le principe physique est le suivant : un enduit contenant une grande quantité de petits cristaux capables d'aimantation rémanente, souvent un oxyde de fer, recouvre le support de l'enregistrement.

**Écriture de données.** L'entrefer d'un électroaimant circule au ras de l'enduit, imposant une direction au magnétisme des cristaux. La variation du signal dans le temps provoque donc une variation d'aimantation dans l'espace et va orienter les cristaux selon une certaine direction de façon pérenne.

**Lecture de données.** Quand un bobinage similaire à l'électroaimant passe au même endroit et à la même vitesse, la variation de l'aimantation provoque, comme dans un alternateur ou une dynamo mais avec une très faible puissance, une force électromotrice dans le circuit (induction). Un amplificateur électronique permet de reconstituer un signal analogue à celui de l'enregistrement. [2]

Je vous invite à aller visionner cette vidéo explicative pour comprendre le fonctionnement de l'encodage sur support magnétique :
`https://www.youtube.com/watch?v=89RZCqyGM7w`

2. Source : `https://fr.wikipedia.org/wiki/Bande_magn%C3%A9tique`

**Focus sur le disque dur magnétique HDD (*Hard Disk Drive*).** Un disque dur est usuellement constitué de plusieurs plateaux (disques) ayant le même axe (sur roulements à billes ou à huile). Les plateaux sont solidaires de cet axe central qui peut tourner à une vitesse constante (A en gris clair sur le croquis) comprise entre 3 600 et 15 000 tr/min (constante sur tous les modèles)
Chaque plateau est recouvert des deux côtés d'un oxyde magnétique. Un ensemble de bras mobiles (B en gris foncé sur le croquis) disposés en dents de peigne permet de positionner des têtes de lecture/enregistrement (C en bleu sur le croquis) de façon stable à plusieurs emplacements au-dessus des surfaces.[3] Toutes les têtes de lecture/écriture sont reliées à une armature qui se déplace à la surface des plateaux, avec une ou deux têtes par plateau (une tête par face utilisée). L'armature déplace les têtes radialement à travers les plateaux pendant qu'ils tournent, permettant ainsi d'accéder à la totalité de leur surface.
Chaque donnée binaire est repérée par la donnée de 3 informations (coordonnées CHS), voir Figure 5

**Cylinder :** numéro de cylindre (creux)

**Head :** numéro de la tête de lecture concernée (attention, il y a deux têtes par plateau)

**Sector :** secteur angulaire sur lequel se situe la donnée

Le modèle d'adresse CHS ne convient bien qu'aux disques durs et aux disquettes (pas aux bandes magnétiques ou clefs USB par exemple). De ce fait, un autre mode d'adressage logique (LBA *Logical Block Adressing*) plus général est utilisé par le BIOS ou le système d'exploitation[4]. Cependant l'adresse LBA est convertie à très bas niveau en une adresse CHS, qui correspond à la réalité physique du disque dur.

L'électronique associée contrôle le mouvement de l'armature ainsi que la rotation des plateaux, et réalise les lectures et les écritures suivant les requêtes CHS reçues. Les firmwares des disques durs récents sont capables d'organiser les requêtes de manière à minimiser le temps d'accès aux données, et donc à maximiser les performances du disque

Contrairement aux CD/DVD, ce sont d'abord les pistes périphériques (c'est-à-dire les plus éloignées du centre du plateau) qui sont écrites en premier (et reconnues comme « début du disque »), car c'est à cet endroit que les performances sont maximales : en effet, la vitesse linéaire d'un point du disque est plus élevée à l'extérieur du disque (à vitesse de rotation constante) donc la tête de lecture/écriture couvre une plus longue série de données en un tour qu'au milieu du disque.

## I. 2. e. Supports optiques

Les CD, les DVD et les Blu-ray sont les disques optiques les plus connus.
Sur ces supports, l'information binaire est encodée grâce à la gravure de creux et de bosses en sillon spiralaire sur un support réfléchissant. Pour lire les données encodées, un laser balaye ces séries de creux et de bosses et le rayon réfléchi va être analysé : lors d'un changement (creux vers bosse ou bosse vers creux) des interférences destructives vont atténuer le signal réfléchi et cela correspondra à un état 0. Sur une zone continue (intérieur d'un creux ou interne d'une bosse) le signal est plus intense et cela encodera un état 1.
Le CD a une capacité de stockage d'environ 700 Mo, le DVD (double couche) peut stocker jusqu'à 8.7 Go. Un DVD Blu-Ray peut stocker jusqu'à 50 Go. Le terme Blu-Ray fait référence au laser bleu utilisé pour leur conception. Contrairement au laser rouge employé

3. Pour une disquette le principe est exactement le même mais il n'y a qu'un seul plateau souple.
4. https ://fr.wikipedia.org/wiki/Logical_block_addressing

pour les CD et les DVD, ce laser permet d'obtenir une meilleure résolution à la fois pour l'image et pour le son.
La vidéo suivante donne des détails supplémentaire sur cette technologie :
`https://www.youtube.com/watch?v=Uzqo-cXFNk8`

## I. 2. f. Supports micro-électronique

Ce sont les supports de données qui nous intéresserons le plus cette année puisque toutes les mémoires de travail (registres, mémoires caches, RAM) des ordinateurs actuels sont de ce type. Ce sont des mémoires :

**volatiles :** les données stockées disparaissent lorsque l'alimentation électrique est coupée.

**à accès direct :** ce qui permet d'obtenir des temps d'écriture et de lecture extrêmement rapides, indispensable pour le travail du microprocesseur et l'exécution rapide de programmes complexes

**RW :** elles sont accessibles en lecture et en écriture le plus souvent

**peu ou non amovibles :** elles sont généralement conçues sous forme de circuits intégrés liés à la carte mère de l'ordinateur.

Elles sont construites avec des transistors MOSFET (*Metal-Oxyde Semi-Conductor Field Effect Transistors*). Ce sont des composants électroniques qui possèdent trois broches, des pattes métalliques sur lesquelles on connecte des fils électriques. On peut appliquer une tension électrique sur ces broches, qui peut représenter soit 0 soit 1. Sur ces trois broches, il y en a deux entre lesquelles circule un courant (la source et le drain), et une troisième qui commande le courant (la grille). Le transistor s'utilise le plus souvent comme un interrupteur commandé par sa troisième broche, la grille. Le courant qui traverse les deux premières broches passe ou ne passe pas selon le courant appliqué sur la grille, voir Figure 7.
L'association astucieuse de ces transistors sous forme de bascules mémoires pilotables elles-mêmes associées sous forme de registres permet de créer des mémoires statiques adressables à accès direct en lecture et en écriture (*SRAM Static Random Access Memory*).

Il existe deux types de mémoires RAM : les mémoires SRAM vues auparavant, et les mémoires DRAM (*Dynamic Random Access Memory*). Les données d'une mémoire statique ne s'effacent pas tant qu'elles sont alimentées en courant. Par contre, les données des mémoires dynamiques s'effacent en quelques millièmes ou centièmes de secondes si l'on n'y touche pas. Il faut donc réécrire chaque bit de la mémoire régulièrement, ou après chaque lecture, pour éviter qu'il ne s'efface. On dit qu'on doit effectuer régulièrement un rafraîchissement mémoire. Le rafraîchissement prend du temps, et a tendance à légèrement diminuer la rapidité des mémoires dynamiques.

Les mémoires SRAM sont surtout utilisées dans les registres du processeur ou pour des mémoires de petite taille. Par contre, les mémoires DRAM sont utilisées pour des mémoires de plus grande taille, comme la mémoire de travail (RAM) de l'ordinateur.

Outre les mémoires vives, il existe des mémoires qui sont elles aussi électroniques, adressables, mais dans lesquelles on ne peut pas écrire : ce sont les mémoires ROM (*Read-Only Memory*). Si on ne peut pas écrire dans une ROM, certaines permettent cependant de réécrire intégralement leur contenu : on dit qu'on reprogramme la ROM. Insistons sur la différence entre reprogrammation et écriture : l'écriture permet de modifier un byte sélectionné/adressé, alors que la reprogrammation efface toute la mémoire et la réécrit en totalité. De plus, la reprogrammation est généralement beaucoup plus lente

qu'une écriture, et doit parfois se faire physiquement/chimiquement par ultraviolets. Les mémoires ROM sont souvent utilisées pour stocker des programmes essentiels, par exemple pour le démarrage de l'ordinateur.
**Mémoires électroniques non volatiles.** Jusqu'à il y a peu, ce type de mémoire était volatile mais grâce à l'avènement de la mémoire flash, il existe maintenant des mémoires non-volatiles, y compris des mémoires de masse (https ://fr.wikipedia.org/wiki/SSD *Solid-State Drive* ou disque statique à semi-conducteurs)

### I. 2. g. Supports quantiques

Ce sont peut-être les supports de stockage de demain... Je vous laisse aller lire l'article de Pour La Science sur ces mémoires encore à l'état expérimental mais faisant l'objet d'intenses recherches au niveau international :
`http://www.lkb.upmc.fr/quantumnetworks/wp-content/uploads/sites/26/2015/10/PLS_Memoires_Laurat_2010.pdf`

## I. 3. Choix de numérisation. Formats

Numériser une donnée nécessite de **faire des choix** et de **se mettre d'accord** (trouver un convention) sur la façon de procéder pour l'encodage. Mais il faut également être capable de garder une trace de ce processus d'encodage pour récupérer les données. En effet, une fois encodée, l'information numérique n'est rien d'autre qu'une suite d'états binaires (0 ou 1). Si l'on souhaite récupérer ces informations, il faut savoir comment regrouper ces 0 et 1 et savoir comment les interpréter.

**Définition 2 (Format de données)**

> Un format informatique est une **convention pour représenter une donnée** sous forme numérique (série d'états binaires). Le choix d'un format permet d'encoder une donnée en une série de 0 et de 1. La connaissance du format dans lequel la donnée a été encodée permet de décoder la série de 0 et de 1 et de retrouver l'information qui se cache derrière la série d'états binaires.

Un format de données est ainsi une convention (éventuellement normalisée) utilisée pour représenter des données - des informations représentant un texte, une page, une image, un son, un fichier exécutable, etc. C'est un gabarit où les données sont placées à des endroits particuliers pour que l'outil qui lit ce format trouve les données où il s'attendait à les trouver. Lorsque ces données sont stockées dans un fichier, on parle de format de fichiers. Une telle convention permet d'échanger des données entre divers programmes informatiques ou logiciels, soit par une connexion directe, soit par l'intermédiaire d'un fichier.

**Définition 3 (Interopérabilité)**

> On appelle **interopérabilité** cette possibilité d'échanger des données entre différents logiciels en s'appuyant sur des conventions de formats.

**Un format est dit spécifié** lorsqu'il est suffisamment décrit pour en développer une implémentation complète. La spécification est souvent trouvée sous la forme d'un fichier au format pdf ou texte, en une ou plusieurs langues. Elle contient des informations qui nécessitent le plus souvent une bonne connaissance en informatique. Il n'y

a pas d'adresse particulière qui regroupe toute les spécifications. Elles se trouvent le plus souvent sur le site internet du propriétaire du format ou sur celui de l'organisme qui a édité une norme à son sujet. Des ingénieurs experts des grandes sociétés de l'informatique et de l'électronique se réunissent régulièrement lors de grands consortiums pour inventer de nouveaux formats (plus efficaces, moins gourmands en mémoire, plus facile à encoder ou décoder) ou améliorer et faire évoluer les formats existants. A l'issue de ces réunions, l'ensemble des acteurs d'une filière numérique se mettent d'accord et publient une norme décrivant le protocole d'encodage des données. Cet accord permet d'assurer l'interopérabilité entre les différents acteurs : fabricants de matériel électronique, développeurs de logiciels, utilisateurs.

**A contrario, un format non-spécifié** est par exemple un format que l'on peut déduire de la forme produite par un logiciel mais dont la description n'est pas explicitement donnée et est souvent intimement mêlée au code du logiciel. Quand aucune forme ne peut être appliquée, on parle généralement de fichier binaire car c'est la seule connaissance que l'on a du fichier.

Un format peut-être :

**ouvert :** d'après l'article 4 de la loi française n°2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique : « *On entend par standard ouvert tout protocole de communication, d'interconnexion ou d'échange et tout format de données interopérable et dont les spécifications techniques sont publiques et sans restriction d'accès ni de mise en œuvre.* » Ainsi, un format « ouvert » est un format non seulement « spécifié » mais, de plus, « accessible » et « interopérable ». Ainsi, n'importe quel programmeur ou utilisateur peut donc lire et utiliser les spécifications du format pour manipuler les données à sa guise et rendre ses logiciels compatibles (interopérables) avec ce format (possibilité de lire des données stockées dans ce format et de stocker des données en suivant les spécifications de ce format)

**libre :** Les notions de format ouvert et de format libre sont très proches. Cependant, un format sera qualifié de libre uniquement si aucune restriction juridique ne lui est applicable. Cela va au-delà des restrictions d'accès et de mise en œuvre auxquelles l'article 4 cité plus haut fait référence.

**fermé :** les spécifications ne sont pas publiques, seules les personnes ayant rédigé les spécifications sont en mesure de savoir en détail comment les données stockées dans ce format sont rangées en mémoire.

Une autre distinction s'opère entre un format :

**normalisé** , faisant l'objet d'une normalisation par une institution publique ou internationale (ISO, W3C)

**quelconque** , qui peut devenir un standard du fait de sa popularité.

Un format est dit propriétaire s'il a été élaboré par une entreprise, dans un but essentiellement commercial. Un format propriétaire peut être ouvert (le format PDF d'Adobe par exemple) s'il est publié, ou fermé (le format Doc de Microsoft par exemple). Mais même lorsque des spécifications sont rendues publiques, les entreprises à l'origine de formats propriétaires tentent d'en conserver le contrôle soit en proposant régulièrement de nouvelles versions plus élaborées (contrôle par maintien d'une avance technologique), soit en utilisant des moyens juridiques comme le brevet.

Formats de fichiers

| Catégorie | Formats |
|---|---|
| Images | PNG, MNG, TIFF, JPEG, GIF, TGA, OpenEXR, BMP, FITS (en) |
| Dessin vectoriel | VML, SVG, Silverlight, SWF, AI, EPS, DXF |
| 3D | XCF, BLEND, SKP, (SKB), DXF, 3DS Max, C4D, VRML, X3D, IFC, DWG |
| Son | OGG, FLAC, MP3, WAV, WMA, AAC |
| Vidéo | MPEG, OGM (DVD, DivX, XviD), AVI, Theora, FLV |
| Page | PDF, PostScript, HTML, XHTML, XML, PHP |
| Document de traitement de texte | ODT, TXT, DOC, RTF |
| Exécutable | BIN, ELF, EXE, SDC, BAT |
| Archives (fichiers généralement compressés) | 7Z, TAR, GZIP, ZIP, LZW, ARJ, RAR, SDC |
| Archives pour bandes dessinées (formats identiques aux formats d'archive sur lesquels ils sont basés: seul l'extension du fichier diffère) | CB7 (.cb7), basé sur 7z<br>CBA (.cba), basé sur ACE<br>CBR (.cbr), basé sur RAR<br>CBT (.cbt), basé sur TAR<br>CBZ (.cbz), basé sur ZIP |

Critères pour le choix et l'élaboration d'une nouvelle norme de format :

**Critères techniques :**
- dégradation de la donnée (échantillonnage, compression...)
- espace mémoire occupé par le fichier encodé (algorithmes de compression)
- durée/performance d'encodage
- durée/performance de décodage
- disponibilité (ouvert) et clarté de la spécification
- complexité du format
- complexité des algorithmes d'encodage
- complexité des algorithmes de décodage

**Critères juridiques :** format propriétaire, breveté, restrictions d'usage ou de mise en œuvre, format fermé ou ouvert

**Critères éthiques :** format ouvert voire libre ou non

# II. Encodage des textes

L'encodage des textes consiste plus ou moins à attribuer à chaque caractère du texte (caractères blancs et ponctuation compris) un code sur un certain nombre de bits. Cette table de correspondance qui, à chaque caractère, associe son encodage binaire est appelée *jeu de caractères*

## II. 1. Le format d'encodage ASCII : un standard historique anglo-saxon

Seuls les caractères utiles en langue anglaise ont été pris en compte dans la création de ce format, car ce format a été discuté et adopté uniquement à l'initiative des grandes firmes américaines dans les années 70.
Ce jeu de caractères n'inclut donc aucun caractère accentué, aucun idéogramme asiatique, aucune lettre arabe par exemple. Malgré tout, ce format, très simple, est devenu un standard et est encore utilisé aujourd'hui. La Figure **??** décrit le jeu de caractères de ce format.

Ce format définit un jeu de 128 caractères numérotés de 0 à 127, voir Figure 8. Sept bits suffisent donc à coder tous les caractères pris en compte dans ce format. Cependant, comme la plupart des ordinateurs travaillent sur des données représentées sur un nombre de bits multiple de huit, les caractères ASCII sont codés sur 8 bits (1 octet), le bit de poids fort étant 0.

**Exemple 4**

On considère le texte suivant « J'aime la prepa »

1. Combien faut-il d'octets pour stocker ce texte en ASCII ? [a]
2. Écrire l'encodage binaire mémoire correspondant. [b]

*a.* Réponse : 15 caractères × 8 bits = 80 bits = 10 octets.

*b.* Réponse : En hexadécimal :4A 27 61 69 6D 65 20 6C 61 20 70 72 65 70 61 soit, en binaire : 01001010 00100111 01100001 01101001 01101101 01100101 00100000 01101100 01100001 00100000 01110000 01110010 01100101 01110000 01100001

Dans `emacs` `Alt +x` puis taper *hexl−mode* dans la console emacs pour observer l'encodage binaire (en hexadécimal) du texte tapé. La Figure 9 nous montre comment vérifier ainsi les réponses de l'exercice précédent. Pour sortir de ce mode, taper `Ctrl+c` deux fois.

## II. 2. Évolution des formats d'encodage de texte.

Petit à petit, la nécessité de prendre en compte les caractères utilisés dans d'autres langues s'est faite sentir. Plusieurs formats normalisés ont émergé prenant en compte de plus en plus de caractères. L'article suivant décrit très bien ces évolutions : `http://sdz.tdct.org/sdz/comprendre-les-encodages.html`

- ISO 8859-n latin-1 qui est une extension du jeu de caractères ASCII permettant d'écrire davantage de langues européennes, en prenant en compte les caractères accentués, et certains caractères spéciaux (mais pas tous !) que l'on trouve dans les langues européennes
- ISO 2022 pour la prise en compte des idéogrammes et des alphabets asiatiques
- ISO 10646 qui introduit le jeu de caractères universel (UCS pour *Universal Character Set*, JUC en français) Unicode

La norme Unicode est une norme développée par le Consortium Unicode publiée pour la première fois en 1991 avec des versions qui évoluent depuis. On peut la voir comme une surcouche (une extension) d'ISO 10646. En fait, les deux normes sont développées parallèlement et synchronisées en permanence. Là où ISO 10646 liste simplement les caractères du jeu et leur assigne un nom et un code, Unicode décrit également :

- des algorithmes de traitement, notamment pour la gestion des différents sens d'écriture
- des encodages permettant de transcrire le JUC, notamment :
  - l'encodage UTF-8 (UCS Transformation Format, 8 bits)
  - l'encodage UTF-16 (UCS Transformation Format, 16 bits)

Il existe une notation officielle pour désigner n'importe quel caractère Unicode : U+xxxx, où xxxx est le code hexadécimal (jusqu'à 6 chiffres). Par exemple, U+0041 correspond à la lettre A majuscule. [5]

5. Une astuce : Certaines distributions Linux offrent le raccourci Ctrl+Shift+U+code pour insérer directement n'importe quel caractère Unicode dont vous spécifiez le code hexadécimal.

# III. Écriture des nombres entiers

## III. 1. Rappels de CE1

Lorsque l'on écrit le nombre 137, sans précision particulière, on suppose, dans nos sociétés, qu'il est écrit en base 10. [6]

$$137 = 1 \times 10^2 + 3 \times 10^1 + 7 \times 10^0$$

On parle de numération positionnelle additive en base 10.

**en base 10 :** car elle repose sur les puissances de 10 (10, $10^2 = 100$, $10^3 = 1000$...) ;

**additive :** car on additionne les contributions des différentes puissantes de 10 ;

**positionnelle :** car la position de chaque symbole (ici des chiffres de 0 à 9) indique la puissance de 10 qui va être multipliée par la quantité correspondant à ce symbole.

Comme on est en base 10, les symboles rouges devant les puissances de 10 désignent des quantités allant de 0 à 9 (si on allait au delà de 9, l'écriture ne serait plus unique, en lien avec la notion de division euclidienne et les théorèmes d'arithmétique que vous verrez cette année)

## III. 2. Écriture dans une base quelconque, formalisme

La *base*, dans un système de numération, est le nombre de symboles (habituellement les chiffres) qui pourront servir à exprimer des nombres.

**Définition 4 (Écriture d'un nombre en base $b$)**

Soit $x$ un nombre entier.
En base $b$, on dispose de $b$ symboles pour écrire les nombres. On note $S$ l'ensemble des symboles

$$S = \{s_0;\ s_1;\ \dots\ s_{b-1}\}$$

L'écriture de $x$ dans la base de numération $b$ est une série de $N$ symboles, c'est-à-dire

$$x = x_{N-1}x_{N-2}\cdots x_1x_0,$$

où $x_i \in S$ de sorte que :

$$x = \sum_{i=0}^{N-1} x_i \times b^i. \tag{1}$$

L'humain s'est habitué à compter en base 10, le système *décimal*, où les symboles sont :

$$S = \{0;\ 1;\ \dots\ 9\}$$

En informatique, on utilise les 3 systèmes suivants :

6. Attention, la numération positionnelle additive en base 10 n'a pas toujours été la norme ! Elle s'est construite au fil des civilisations et s'est stabilisée en Europe au Moyen-Age avec la fusion de la numération additive égyptienne, de la numération positionnelle indienne et du zéro arabe. Les mésopotamiens comptaient en base 60, les maya en base 20... `http://www.ac-grenoble.fr/ecole/74/maths.sciences74/IMG/pdf/311.pdf`

**binaire :** les ordinateurs calculent et stockent les données en base 2. Dans cette base, il n'y a que 2 symboles :

$$S = \{0; 1\}$$

On appelle ces symboles des **bits**[7]

**octal :** on utilise l'écriture en base 8 car elle correspond aux mots mémoires des anciens ordinateurs (les octets). Les symboles 8 dans cette écriture sont $S = \{0, 1, \ldots, 7\}$.

**hexadécimal :** pour des raisons de lisibilité, on utilise souvent en informatique la base 16. Dans ce dernier système, les symboles utilisés pour les quantités 10 à 15 sont généralement les 6 premières lettres A à F. Les 16 symboles utilisés en hexadécimal sont donc :

$$S = \{0, 1, \ldots, 8, 9, A, B, C, D, E, F\}$$

En informatique, quand un nombre est écrit en hexadécimal à l'écran, il est souvent précédé de `0x` ou de `$`.

**Remarque.** Lorsque le contexte ne permet pas de déterminer avec certitude la base d'un nombre, celle-ci est indiquée. Par exemple, $10011_2$, aussi parfois noté $\overline{10011}^2$ est le nombre binaire 10011.

**Exemples 5**

**1.** Soit le nombre décimal 348. En décimal, selon la notation ci-dessus, on a $s_0 = 8$, $s_1 = 4$, $s_2 = 3$ et $b = 10$. En effet :

$$348 = 3 \times 10^2 + 4 \times 10^1 + 8 \times 10^0.$$

**2.** En binaire, 348 s'écrit :

$$\begin{aligned} 101011100_2 = {} & 1 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 \\ & + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 \\ & + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0. \end{aligned}$$

**3.** En octal :

$$534_8 = 5 \times 8^2 + 3 \times 8^1 + 4 \times 8^0.$$

**4.** En hexadécimal :

$$15\mathrm{C}_{16} = 1 \times 16^2 + 5 \times 16^1 + \underbrace{12}_{\mathrm{C}} \times 16^0.$$

Des représentations ci-dessus, l'hexadécimale est la plus compacte : elle permet de représenter avec un seul symbole un nombre binaire comptant jusqu'à quatre chiffres. C'est, entre autres, pourquoi c'est une représentation populaire en informatique. □

**Propriété 1**

Si l'on dispose de $N$ chiffres ou symboles en base $b$, on peut représenter $b^N$ nombres entiers différents

7. BIT vient de l'anglais *BInary digiT* qui signifie *symbole binaire*

**Propriété 2**

Le plus grand entier que l'on peut représenter avec $N$ chiffres ou symboles en base $b$ est $b^N - 1$

**Démonstration**

$$\begin{aligned} x_{\max} &= \sum_{i=0}^{m-1} (b-1) b^i \\ &= (b-1) \sum_{i=0}^{m-1} b^i \\ &= (b-1) \left( \frac{b^m - 1}{b-1} \right) \\ &= b^m - 1 \end{aligned}$$

**Propriété 3**

Le plus grand entier positif représentable sur $N$ bits est donc $2^N - 1$.

**Propriété 4**

Un nombre entier non nul $k$ peut être encodé avec $\lfloor \log_2(k) \rfloor + 1$ bits

**Démonstration**

Le nombre minimal $N$ de bits nécessaire pour encoder un entier naturel non nul $q$ est $\lfloor \log_2(q) \rfloor + 1$
$N$ étant le nombre de bits minimal pour encoder $q$, le bit de poids fort est à 1 car sinon, on pourrait l'enlever et l'on aurait besoin que de $N-1$ bits, ce qui contredirait la minimal de $N$. Donc q est nécessairement tel que :

$$q \geq (100\ldots000)_2 = 2^{N-1}$$

Par ailleurs, $q$ est au maximum égal à $(11\ldots111)_2$ sur $N$ bits car sinon, cela signifierait qu'il lui faudrait plus de $N$ bits pour pouvoir être encodé. Ainsi :

$$q \leq (11\ldots111)_2 = 2^N - 1 < 2^N$$

On en déduit le premier encadrement :

$$2^{N-1} \leq q < 2^N$$

$N$ est alors le nombre de bits minimal (optimal) pour l'écriture binaire de $q$.
Comme la fonction logarithme népérien est croissante sur $]0; +\infty[$, les inégalités sont préservées par passage au logarithme :

$$\begin{aligned} & \ln\left(2^{N-1}\right) \leq \ln(q) < \ln\left(2^N\right) \\ \Leftrightarrow \quad & (N-1)\ln(2) \leq \ln(q) < N\ln(2) \\ \Leftrightarrow \quad & N - 1 \leq \frac{\ln(q)}{\ln(2)} < N \\ \Leftrightarrow \quad & N - 1 \leq \underbrace{\log_2(q) < N}_{\text{inégalité 1}} \end{aligned}$$

Pour la dernière étape, on a divisé partout par $\ln(2)$ qui est strictement positif (car $2 > 1$), donc le sens des inégalités n'est pas modifié.
Nous cherchons à déterminer l'entier $N$. Nous allons déjà manipuler les inégalités pour encadrer $N$.

$$\begin{aligned} & \log_2(q) < N \quad \text{inégalité 1} \\ \Leftrightarrow \quad & \log_2(q) - 1 < N - 1 \end{aligned}$$

En combinant avec $N - 1 \leq \log_2(q)$, on obtient l'encadrement :

$$\log_2(q) - 1 < N - 1 \leq \log_2(q)$$

Ainsi,

$$N - 1 = \lfloor \log_2(q) \rfloor$$

Dont on déduit

$$N = \lfloor \log_2(q) \rfloor + 1$$

**Exemple 6**

Par exemple, le plus grand nombre entier représentable en base 10 avec $N = 4$ symboles est

$$x_{\max} = 9999 = 10000 - 1 = 10^4 - 1,$$

alors que le plus grand entier représentable en binaire avec $N = 4$ bits est seulement :

$$x_{\max} = 1111_2 = 15_{10} = 16 - 1 = 2^4 - 1.$$

Il est indispensable de connaître sur le bout des doigts vos puissances de 2 :

| | | | | | |
|---|---|---|---|---|---|
| $2^0$ | $=$ | $1$ | $2^8$ | $=$ | $256$ |
| $2^1$ | $=$ | $2$ | $2^9$ | $=$ | $512$ |
| $2^2$ | $=$ | $4$ | $2^{10}$ | $=$ | $1024$ |
| $2^3$ | $=$ | $8$ | $2^{11}$ | $=$ | $2048$ |
| $2^4$ | $=$ | $16$ | $2^{12}$ | $=$ | $4096$ |
| $2^5$ | $=$ | $32$ | $2^{16}$ | $=$ | $65536$ |
| $2^6$ | $=$ | $64$ | $2^{32}$ | $=$ | $2^2 \times (2^{10})^3 = 4 \times (1024)^3 \approx 4 \times (10^3)^3 \approx 4 \times 10^9$ |
| $2^7$ | $=$ | $128$ | $2^{64}$ | $=$ | $2^4 \times (2^{10})^6 \approx 16 \times (1000)^6 \approx 16 \times (10^3)^6 \approx 16 \times 10^{18}$ |

## III. 3. En pratique : transformer une écriture binaire en écriture décimale

Si on a l'écriture binaire d'un nombre, il suffit d'appliquer la Formule 1 pour obtenir l'écriture du nombre en base 10.

**Exemple 7**

$$\overline{01011010}^2 = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2 = 64 + 16 + 8 + 2 = 90$$

## III. 4. En pratique : transformer une écriture décimale en écriture binaire

Voici un algorithme pour obtenir l'écriture binaire d'un nombre. Nous prenons ici comme exemple 77 en base 10.

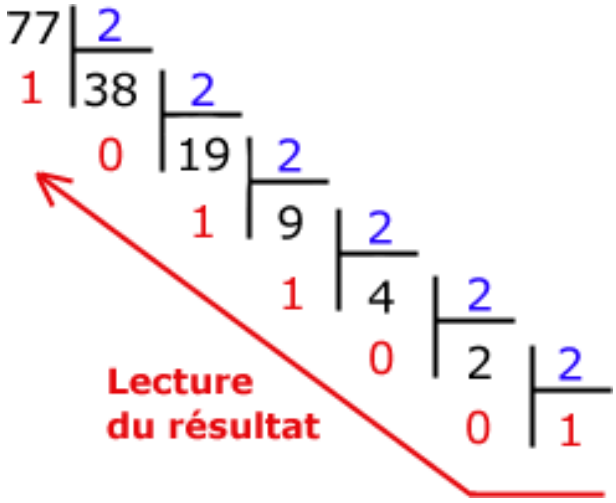
1. On démarre avec le nombre entier que l'on cherche à décomposer, ici 77
2. On cherche la plus grande puissance de 2 qui « rentre » dans 77, c'est $2^6 = 64$. Elle rentre 1 fois.
3. Il reste $77 - 1 \times 2^6 = 77 - 64 = 13$.
4. On cherche la plus grande puissance de 2 qui « rentre » dans 13, c'est $2^3 = 8$. Elle rentre 1 fois.
5. Il reste $13 - 1 \times 2^3 = 5$.

6. On cherche la plus grande puissance de 2 qui « rentre » dans 5, c'est $2^2 = 4$. Elle rentre 1 fois.
7. Il reste $5 - 1 \times 2^2 = 5 - 4 = 1$.
8. On cherche la plus grande puissance de 2 qui « rentre » dans 1, c'est $2^0 = 1$. Elle rentre 1 fois.
9. Il reste $1 - 1 \times 2^0 = 1 - 1 = 0$. L'algorithme est terminé

Ainsi 77 se décompose en :

$$\begin{aligned} 137 &= 1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 \\ &= 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \end{aligned}$$

Ainsi, l'écriture de 77 en base 2 est $\overline{0100\,1101}^2$.
On peut aussi utiliser une représentation graphique comme ci-dessous avec des divisions successives par 2




## III. 5. En pratique : transformer une écriture binaire en écriture octale ou hexadécimale

$$166_{10} = 10100110_2$$

Pour obtenir l'écriture octale, on regroupe les symboles de l'écriture binaire par groupe de 3, en partant de la droite.

$$166_{10} = \underbrace{10}_{2}\ \underbrace{100}_{4}\ \underbrace{110}_{6}$$

$$166_{10} = 246_8$$

Pour obtenir l'écriture hexadécimale, on regroupe les symboles de l'écriture binaire par groupe de 4, en partant de la droite.

$$166_{10} = \underbrace{1010}_{10 \to A}\ \underbrace{0110}_{6}$$

$$166_{10} = A6_{16}$$

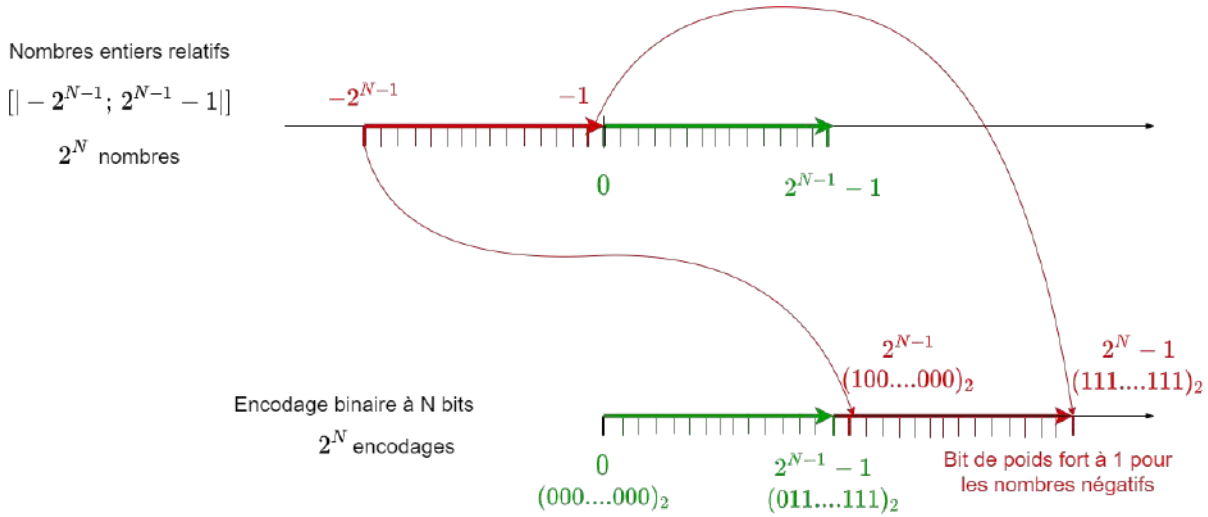
**Remarque.** En informatique, on ne parle pas d'entiers naturels pour d'entier **non signés**. Pour stocker un entier naturel selon l'encodage ci-dessous, il faut en langage C utilise le type `unsigned int`.

# IV. Encodage des nombres entiers relatifs

En général, toutes les données de même type (entier, lettre par exemple) sont stockées sur le même nombre bits pour faciliter le traitement automatique des données. Le nombre bits de stockage utilisé pour un type est une puissance de 2 pour des raisons techniques d'adressage des données dans la mémoire.
Le type correspondant à l'encodage des entiers relatifs tel que nous allons le décrire ci-dessous est le type noté `int` [8] en langage C, OCaml et Python.
Sur les ordinateurs modernes, les entiers relatifs sont généralement stockés sur $N = 32$ bits.
Sur $N$ bits, on peut stocker $2^N$ entiers relatifs différents. Les entiers relatifs représentables iront donc de $-2^{N-1}$ à $2^{N-1} - 1$.
Ils sont ensuite translatés pour être encodés comme des nombres entiers positifs :

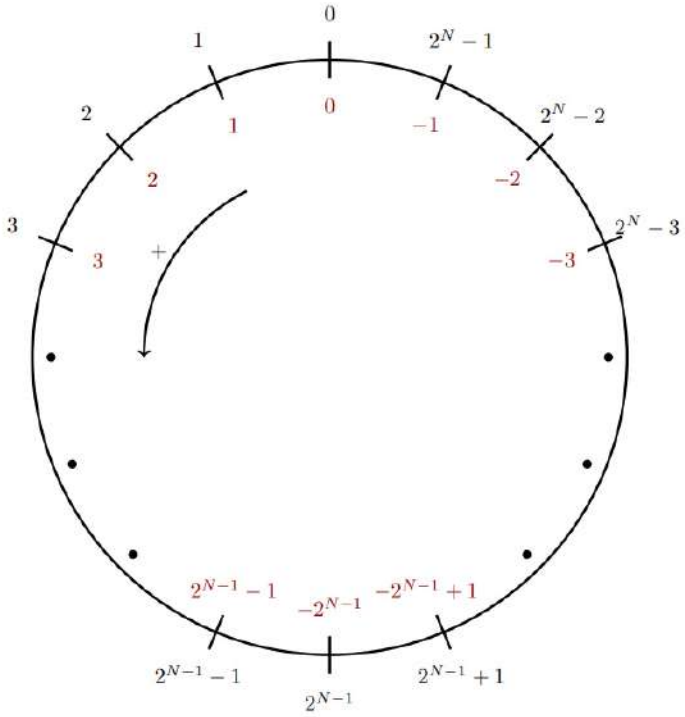
$$f : \begin{array}{lll} [\![-2^{N-1}; 2^{N-1}-1]\!] & \mapsto & [\![0; 2^N - 1]\!] \\ k & \mapsto & \begin{cases} k & \text{si } k \in [\![0; 2^N-1]\!] \\ k + 2^N & \text{si } k \in [\![-2^{N-1}; -1]\!] \end{cases} \end{array} \tag{2}$$

Le dessin ci-dessous représente cette bijection/translation $f$ :



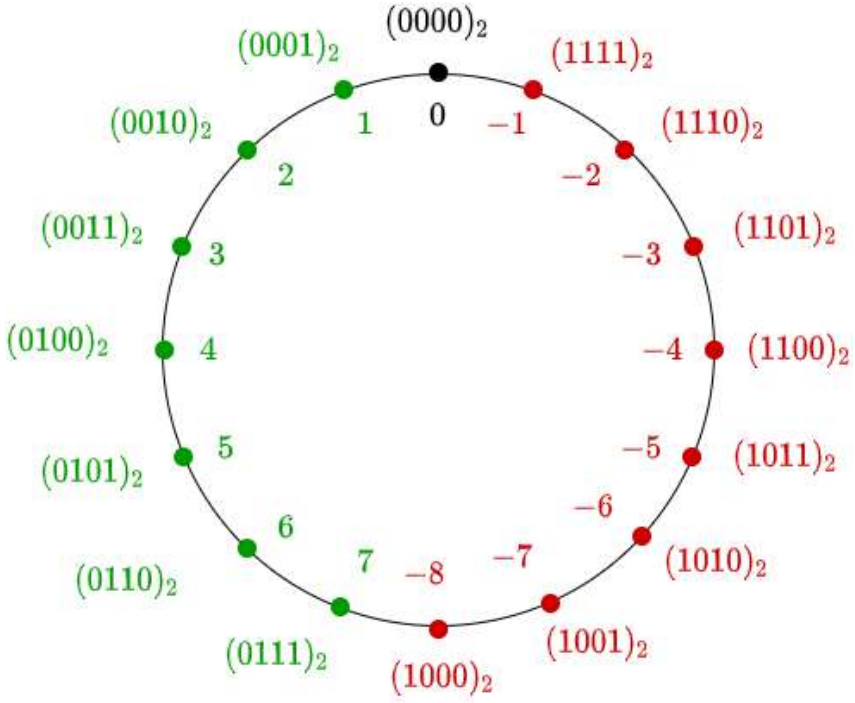

On peut aussi représenter cette association sous forme circulaire :

8. Du latin *integer* qui signifie « entier » et qui a donné les mots intègre, intégral, intégrer ou désintégrer en français.




| | | | |
|---|---|---|---|
| $0\,000\ldots0000\,0000$ | $0$ | | |
| $0\,000\ldots0000\,0001$ | $1$ | $1\,111\ldots1111\,1111$ | $-1$ |
| $0\,000\ldots0000\,0010$ | $2$ | $1\,111\ldots1111\,1110$ | $-2$ |
| $0\,000\ldots0000\,0011$ | $3$ | $1\,111\ldots1111\,1101$ | $-3$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $0\,111\ldots1111\,1110$ | $2^{N-1}-2$ | $1\,000\ldots0000\,0001$ | $-2^N+1$ |
| $0\,111\ldots1111\,1111$ | $2^{N-1}-1$ | $1\,000\ldots0000\,0000$ | $-2^N$ |

Par exemple, dans le cas d'un encodage sur 4 bits, la représentation est la suivante :



## IV. 1. En pratique : le complément à 2

En pratique, pour obtenir l'encodage binaire d'un nombre entier négatif $-k \in [\![-2^{N-1}; -1]\!]$, on ne calcule par $-k + 2^N$ :

1. On écrit l'entier $k$ positif en binaire sur $N$ bits
2. On écrit le nombre binaire complémentaire en changeant chaque bit. Si le bit était à 0, on le met à 1. S'il était à 1, on le met à 0
3. On additionne 1 au nombre binaire complémentaire, en calculant en base 2

**Exemple 8**

Je cherche l'encodage binaire de $-5$. Donc $k = 5$

1. J'écris $k = 5$ en binaire : $5_{10} = 0101_2$
2. Je complémente : $1010_2$
3. J'ajoute 1 : $1010_2 + 0001_2 = 1011_2$

L'encodage binaire de $-5$ est donc 1011.

**Démonstration**

Je veux encoder l'entier négatif $-k \in [\![-2^{N-1}; -1]\!]$ sur $N$ bits.

1. J'écris $k$ en binaire :
$$k = \sum_{i=0}^{N-1} k_i \, 2^i$$
2. Je complémente l'écriture obtenue :
$$\sum_{i=0}^{N-1} (1 - k_i) \, 2^i$$
3. J'ajoute 1 :
$$1 + \sum_{i=0}^{N-1} (1 - k_i) \, 2^i = 1 + \sum_{i=0}^{N-1} 2^i - \sum_{i=0}^{N-1} k_i \, 2^i = 1 + \left(2^N - 1\right) - k = 2^N - k = f(-k)$$

où $f$ est la fonction donnée par la Définition 2.

En fait, on a le résultat suivant :

**Propriété 5**

Pour toute valeur $x$ encodée sur $N$ bits, si on note $\overline{x}$ désigne la valeur associée à la série de bits de $x$ complémentée (inversée), on a :

$$(x)_2 + (\overline{x})_2 = (1111\ldots1111)_2$$

Ce qui est équivalent, en valeurs décimales classiques, à :

$$\begin{aligned} & x + \overline{x} = 2^N - 1 \\ \Leftrightarrow \quad & 2^N = x + \overline{x} + 1 \end{aligned}$$

**Exemple 9**

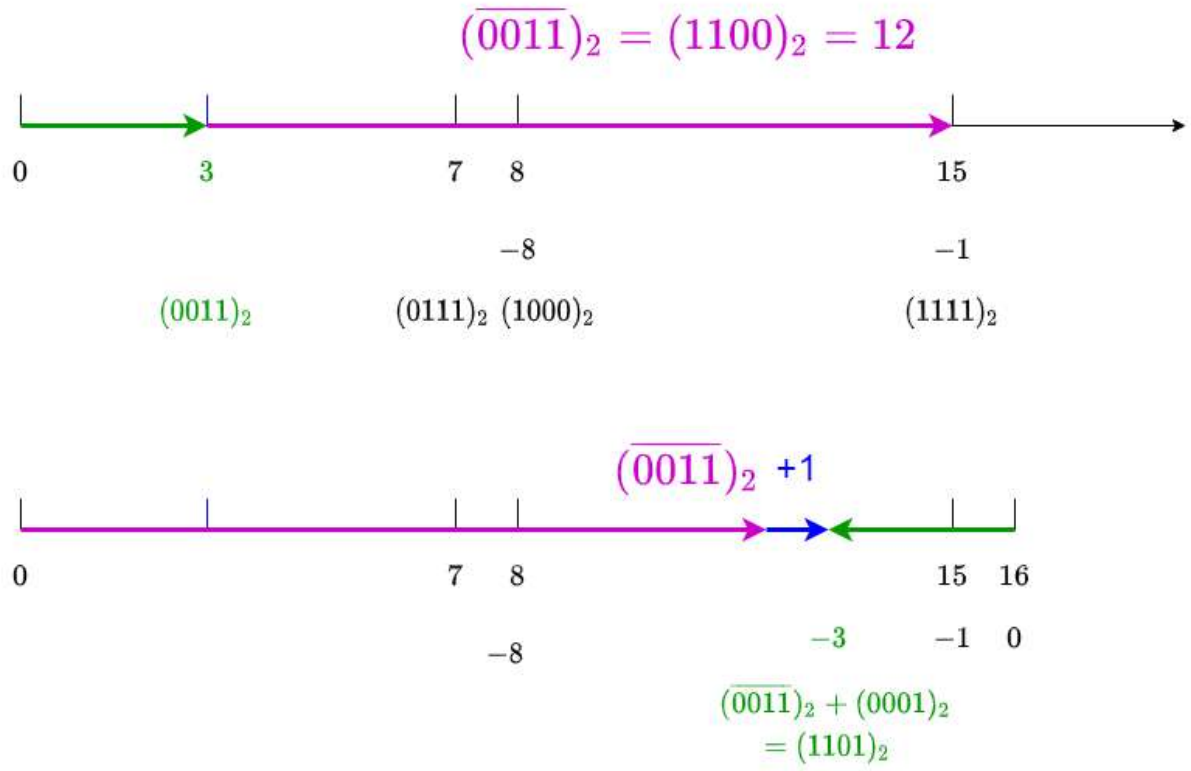
$$\begin{array}{rr} & (10100011)_2 \\ + & (01011100)_2 \\ \hline & (11111111)_2 \end{array}$$

et si on ajoute 1 à $(11111111)_2$, on obtient bien :

$$\begin{array}{rr} & (11111111)_2 \\ + & (00000001)_2 \\ \hline & (100000000)_2 = 2^N \end{array}$$

En fait, on complémentant puis en ajoutant 1, on dépasse la capacité et les $N$ premiers bits retournent à 0.

Ainsi, comme l'entier négatif $-k$ est encodé par une série de bits de valeur $2^N - k$ (voir la définition de la fonction $f$), en remplaçant $2^N$ par $k + \overline{k} + 1$, on en déduit que $-k$ est encodé par $k + \overline{k} + 1 - k$
Après simplification, on déduit que $-$ est représenté par $\overline{k} + 1$, ce qui est exactement la méthode du complément à 2.
La Figure ci-dessous donne du sens à la méthode du complément à 2 dans le cas d'entiers relatifs codés sur 4 bits. On cherche dans l'exemple de la figure à trouver la représentation de $-3$ sur 4 bits. Le calcul du complément à 2 de 3 correspond au calcul de la distance violette. La deuxième figure montre pourquoi il est nécessaire d'ajouter 1.




## IV. 2. Avantages de cet encodage

**Avantage n°1 :** Ce choix permet d'identifier très facilement un nombre négatif grâce à son bit de poids fort (le plus à gauche). S'il vaut 1, le nombre est négatif.

**Avantage n°2 :** Ce choix permet également d'effectuer comme d'habitude les opérations d'addition et de multiplication avec des entiers relatifs grâce à l'arithmétique modulaire. Les opérations binaires donnent les résultats escomptés car la congruence modulo 2 est compatible avec l'addition et la multiplication de nombres relatifs écrits en binaire.

**Exemple 10**

Pour cet exemple, on se place dans le cas où les entiers relatifs sont codés sur $N = 8$ bits. On peut donc encoder tous les nombres entiers relatifs de l'intervalle décimal $[\![-2^7;\ 2^7-1]\!] = [\![-128;\ 127]\!]$

$$(3)_{10} = (00000011)_2$$

$$(-2)_{10} = (11111110)_2$$

**Addition :** On obtient bien $(00000001)_2 = (1)_{10}$

$$\begin{array}{rr} & (00000011)_2 \\ + & (11111110)_2 \\ \hline & (00000001)_2 \end{array}$$

**Multiplication :** On obtient bien $(111111010)_2 = (-6)_{10}$

$$\begin{array}{rr} & (11111110)_2 \\ \times & (00000011)_2 \\ \hline & (11111110)_2 \\ + & (111111100)_2 \\ \hline & (111111010)_2 \end{array}$$

**Remarque.** La méthode du complément est valable dans toutes les bases. Si l'on encode les nombres relatifs $[\![-50;\ 49]\!]$ sur 2 symboles en base 10 de $[\![0;\ 99]\!]$, alors $-23$ correspond à 77.
Si l'on applique la méthode du complément à 10 sur des nombres écrits en base 10 avec deux symboles :

- Le complément à 10 de 23 est 76
- On ajoute 1
- On obtient 77, qui est bien la valeur d'encodage en base 10 correspondant à $-23$ dans ce système à 2 symboles.