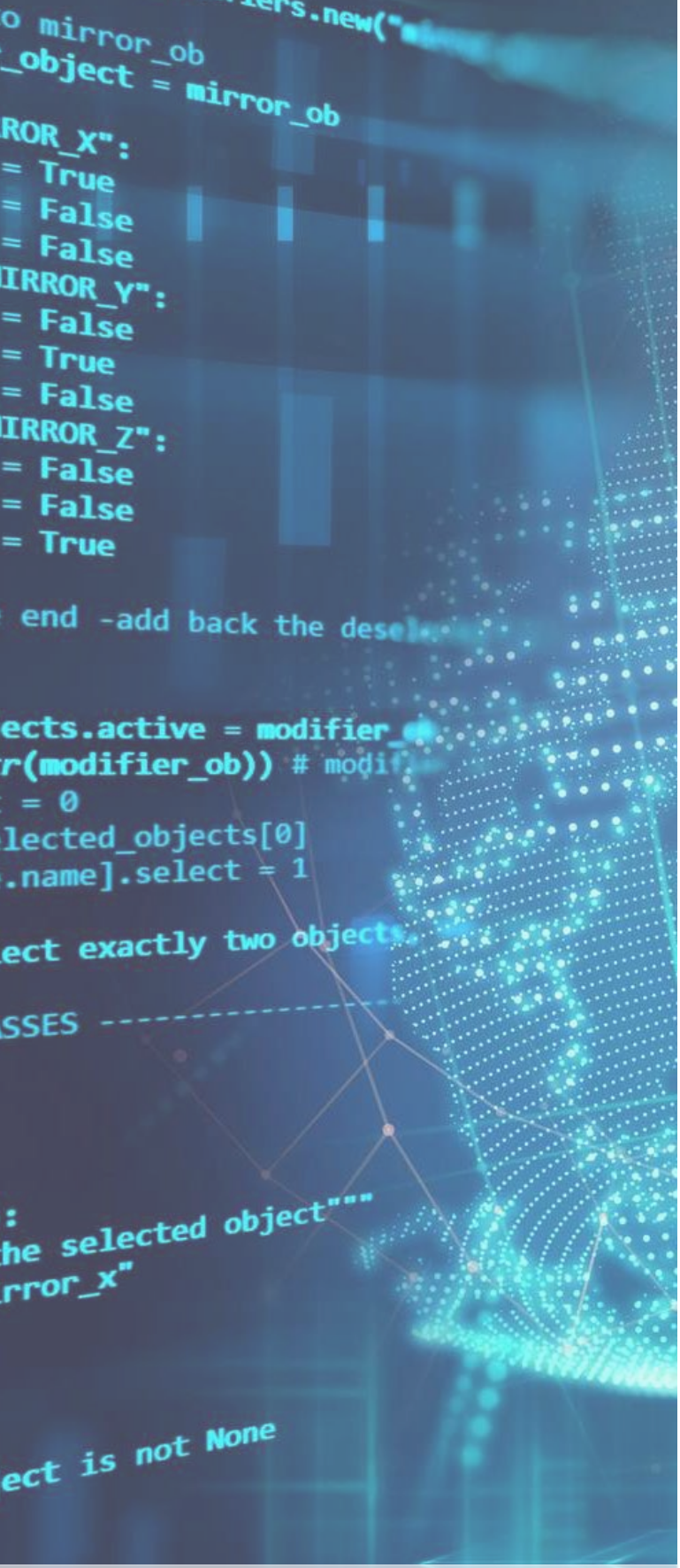


Informatique
Les bases

1ère année maturité

« Il y 10 sortes de gens : ceux qui comprennent le binaire et les autres… »

Damien Cardinaux

Gymnase de Crissier

Edition 2024-2025

Table des matières

1 Introduction

Un peu d'histoire

Le mot *informatique* est la combinaison des mots **information** et **automatique**. Il est apparu pour la première fois en 1957 dans une publication scientifique en Allemagne, puis en 1962 en France. C'est finalement en 1967 qu'il est adopté par l'Académie française.

Avant les années 60, on parle plutôt de **calculateurs** pour désigner les premiers ordinateurs. Comme souvent pour les technologies, la guerre a fait évoluer de manière rapide les innovations dans ce domaine. En effet, durant la Seconde Guerre mondiale, l'armée allemande utilisait une machine de cryptage électro-magnétique appelée **Enigma** pour communiquer. Les échanges d'information générés par Enigma étaient considérés comme indéchiffrables ! Les Alliés ont pourtant réussi grâce notamment à un pionnier de l'informatique à casser le code d'Enigma[1], sans que les Allemands ne le découvre. Il s'appelait **Alan Turing** et a été à l'origine de l'informatique moderne.

Figure 1 – Une machine Enigma

Mais la guerre a également été un vecteur important pour le perfectionnement des calculateurs pour les besoins de calculs balistiques pour les tirs d'artilleries. Il fallait pouvoir calculer beaucoup et vite, bien plus vite qu'un humain, surtout qu'il faut y intégrer des paramètres atmosphériques (vent, humidité, frottement, …).

[1] Vous pouvez visionner le film Imitation Game sorti en 2014

La machine de Turing

C'est en 1936 qu'Alan Turing conceptualise cette machine qui portera son nom. Il entend démontrer qu'une machine automatisée peut **calculer sans intervention humaine**. La **notion de calculabilité** en informatique peut se traduire par la capacité à savoir si un problème est solvable ou non par une machine.

Petit exemple

Un palindrome est un mot que l'on peut lire de la même manière dans les deux sens, comme le mot **SUGUS**.

La machine te Turing a été la première machine à résoudre automatiquement ce genre de problème.

Mais pourquoi cette machine est-elle l'ancêtre de l'ordinateur ?

Sa machine est assez rudimentaire. Elle est constituée d'un ruban à cases dans lesquels on peut inscrire uniquement deux symboles distincts ou aucun symbole. Elle ne peut faire comme opérations que lire ou écrire un symbole, effacer un symbole et déplacer le ruban d'une case vers la droite ou la gauche.

Cette machine est donc capable de résoudre des problèmes en ne connaissant que deux symboles pour représenter de l'information tout comme les 0 et les 1 d'un ordinateur. De plus, elle ne peut lire qu'une seule case du ruban à la fois de manière séquentielle.

Le premier ordinateur

La première machine considérée comme étant un ordinateur car entièrement électrifiée, mesurait 20 mètres de long et 2,5 mètres de hauteur occupant un espace de plus de 160 m^2 (taille d'un appartement de 5,5 pièces).

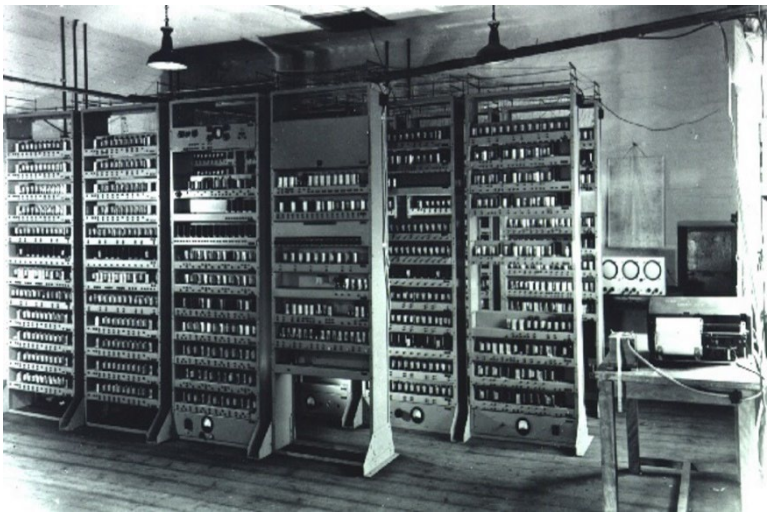


Figure 2 - L'ENIAC

Cette machine a été nommée l'ENIAC pour *Electronic Numerical Integrator and Computer*, et a été inaugurée en 1946 à l'université de Pennsylvanie aux Etats-Unis. Elle permettait d'exécuter jusqu'à 5000 calculs par secondes.

ENIAC Girls

6 femmes ont obtenu une reconnaissance en 2013 grâce à un documentaire sur leur travail de programmation durant la guerre puis sur l'ENIAC. Pour programmer l'ENIAC « elles ne disposaient d'aucun mode d'emploi pour manipuler les quelques 3000 commutateurs »[2] de la machine. Elles ont participé à l'élaboration d'un langage de programmation pour le successeur de l'ENIAC.

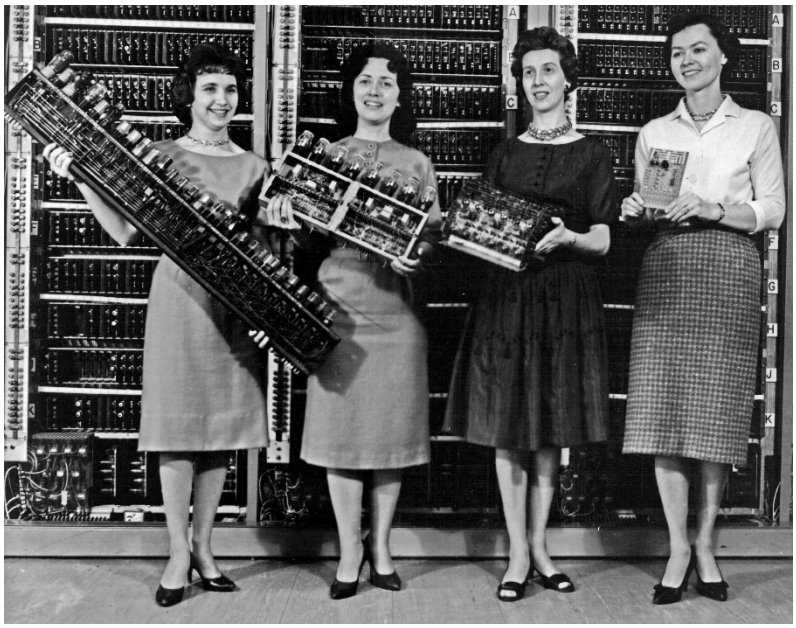


Figure 3 - Les ENIAC six sont un groupe de six programmeuses ayant participé au développement et à la programmation de l'ENIAC

[2] http://francoisehalper.fr/1946-histoire-de-leniac-premier-veritable-ordinateur/

2 Représentation de l'information

2.1 Introduction

Le mot informatique vient de la **combinaison** d'*information* et *automatique*. L'informatique désigne donc le **traitement automatique de l'information** par des machines.

Automatique signifie que la machine exécute des tâches **sans intervention humaine directe** : une fois programmée, elle suit des instructions et produit des résultats. Mais que signifie exactement le mot *information* en science informatique ?

Lorsque vous allumez un ordinateur ou un smartphone, vous voyez apparaître le bureau, avec éventuellement des documents **textes**, des **images** ou des **dossiers**. **Chacun de ces éléments est une information**, représentée sous une forme que la machine peut interpréter.

Prenons l'exemple d'une image : lorsque vous double-cliquez dessus, le fichier contient un codage numérique que l'ordinateur sait décoder pour afficher correctement la photo à l'écran. Même le simple fait de déplacer la souris ou de cliquer génère des informations codées, transmises et interprétées par le système.

> 😆 **Fun fact ! Ou pas…**
>
> Tout ce que vous dites à Siri est encodé, transmis à Apple, stocké et analysé…

Nous allons donc apprendre :

1. Comment les informations sont codées pour être comprises par la machine.
2. Comment elles sont stockées pour pouvoir être retrouvées.
3. Comment elles sont transmises d'un appareil à l'autre.

Tout cela pour que, de notre côté, il nous suffise de double-cliquer sur une icône ou d'ouvrir une page Internet pour accéder à des contenus complexes.

Enfin, puisque Internet permet à des milliards d'ordinateurs de communiquer entre eux, il est essentiel d'assurer la confidentialité des échanges. C'est pourquoi on utilise des techniques de **chiffrement** (et non "cryptage") afin que nos communications ne soient pas lisibles par n'importe qui.

Coder de l'information ce n'est pas nouveau

Depuis ses origines, l'humanité a inventé de nombreux systèmes pour **représenter**, **transmettre** et **conserver** l'information.

Des Sumériens et leur écriture cunéiforme, aux Égyptiens et leurs hiéroglyphes, en passant par les idéogrammes chinois jusqu'aux alphabets modernes, les sociétés humaines ont toujours cherché à créer des codes permettant de partager et mémoriser le savoir.

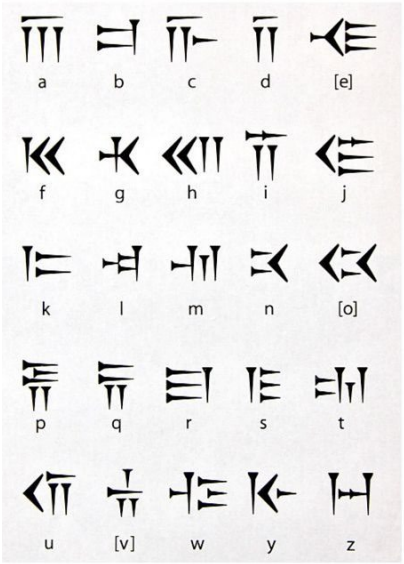


Figure 6 - Pictogrammes sumériens

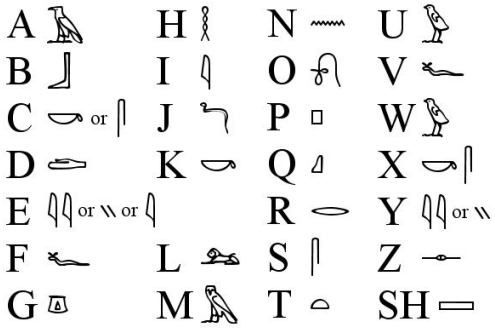


Figure 4 - Hiéroglyphes égyptiens

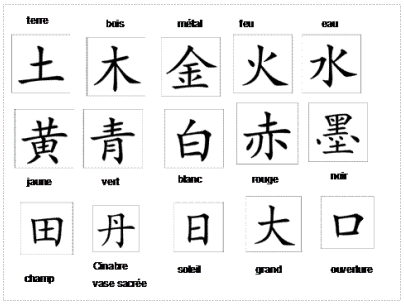


Figure 5 - Idéogrammes chinois

Systèmes de communication

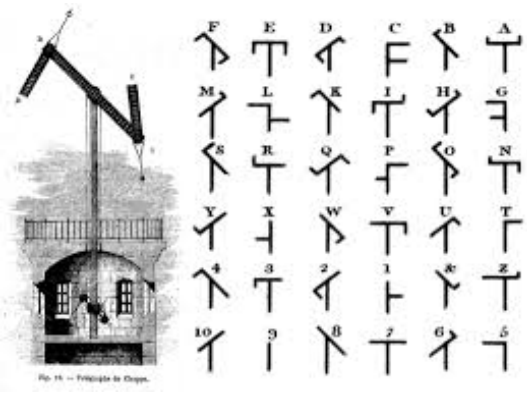
	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I,J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Même les Grecs de l'Antiquité avaient déjà inventé des systèmes de communications codés. Ils transmettaient des messages entre les cités avec un système de torches enflammées.

Cinq torches « à gauche », cinq torches « à droite », étaient séparées par un espace suffisamment grand pour être identifiables à longue distance. Une torche pouvait être soit allumée, soit éteinte. Le nombre de torches allumées à gauche, de 1 à 5, représentait la ligne d'un tableau de décodage, le nombre de torches allumées à droite représentait la colonne de ce même tableau.

Le télégraphe de Chappe

Au XVII^e^ siècle, l'invention du télescope permit d'augmenter considérablement les distances de communication visuelle. Des relais installés sur des collines, espacés d'une dizaine de kilomètres, pouvaient transmettre des messages d'une ville à l'autre à une vitesse inédite.



En 1794, **Claude Chappe**, ingénieur français, mit au point un télégraphe optique composé de bras articulés.

Ces bras formaient différentes positions, visibles à distance avec une lunette, et permettaient de coder des lettres ou des mots entiers. Le système évoquait les gestes qu'un humain ferait avec ses bras pour transmettre un signal.

Anecdote – Premier piratage des communications

Le piratage du télégraphe[3] Chappe est un détournement du réseau de télégraphie optique français entrepris par deux hommes d'affaires bordelais, Louis et François Blanc, entre 1834 et 1836, afin de connaître avant tout le monde le prix de certaines actions à la Bourse de Paris.

Le piratage a été rendu possible par la corruption d'un agent télégraphique de Tours, qui ajoutait discrètement le prix actuel des actions aux messages envoyés par l'État.

Le code Morse

Au début du XIXe siècle, la maîtrise de l'électricité ouvrit la voie à un nouveau mode de communication.

En 1832, **Samuel Morse** inventa le **code Morse**, un système de points et de traits permettant de coder l'alphabet. Transmis par impulsions électriques sur des lignes télégraphiques, il s'imposa rapidement comme un standard mondial de communication.

Le Morse pouvait aussi être adapté à d'autres supports : signaux lumineux (lampes), sonores (sifflets, cloches), ou radio.

D'un point de vue conceptuel, le code Morse est intéressant car il se rapproche du langage binaire de l'informatique : une information complexe (un texte) est traduite en une suite de symboles élémentaires (point ou trait), tout comme l'ordinateur utilise des suites de 0 et de 1.

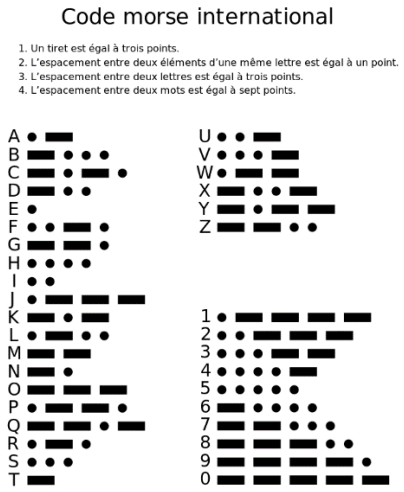


Figure 7 - Coadge de l'alphabet en Morse

[3] https://fr.wikipedia.org/wiki/Piratage_du_télégraphe_Chappe

Les prémices de l'encodage efficace d'information

Si vous observez le code Morse[4], vous remarquerez que les signaux utilisés pour représenter les lettres ne suivent pas simplement l'ordre de l'alphabet, puisqu'il est plus économique de coder les lettres les plus fréquentes avec les codes les plus courts, comme le E.

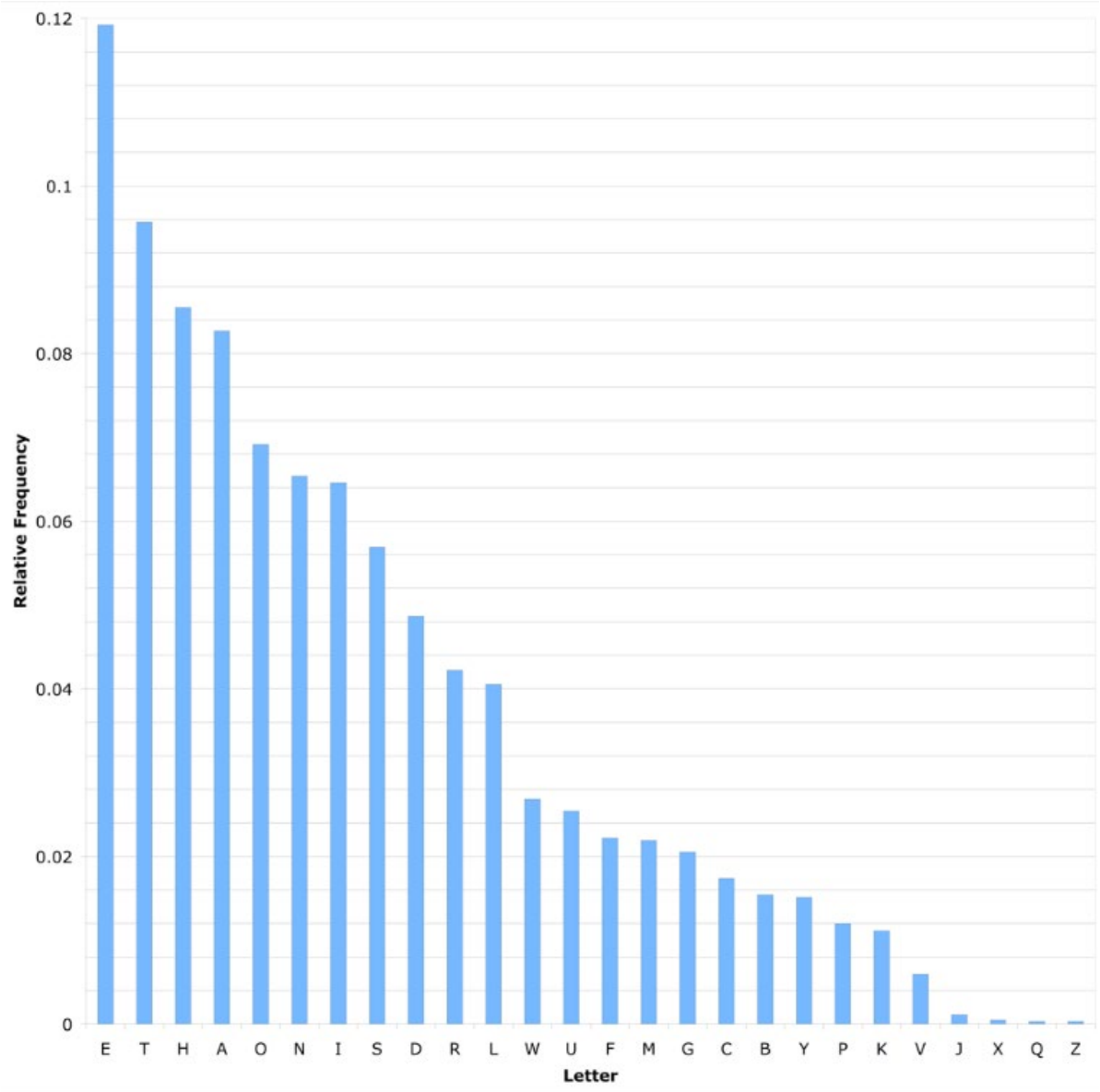


Figure 8 - Représentation de la fréquence moyenne de distribution des lettres en anglais

[4] https://fr.wikipedia.org/wiki/Code_Morse_international

2.2 Le code binaire

Avec l'invention du Morse, l'humanité disposait déjà d'un système de codage et de transmission de l'information par l'électricité. Il ne manquait plus que deux éléments essentiels pour aller vers l'ordinateur moderne :

1. Une pièce technologique capable de contrôler le passage du courant électrique : le **transistor**.
2. Un **système de codage** plus général que le Morse, permettant de représenter tous types d'informations à partir de **seulement deux états électriques** :
    - courant / pas de courant,
    - allumé / éteint
    - vrai / faux,
    - 1 / 0.

Ce système, c'est le **code binaire**.

En utilisant uniquement des 0 et des 1, il permet de représenter n'importe quel type d'information : chiffres, texte, images, sons, vidéos, etc.

Une machine électronique ne comprend en effet que deux états : courant présent ou courant absent. Les informaticiens ont choisi de les symboliser par 0 et 1, tout comme le Morse utilisait le • et le — .

Exemple

Imaginons que l'on souhaite coder trois réponses possibles : {OUI, NON, PEUT-ÊTRE}.

Il existe plusieurs façons de construire un dictionnaire binaire :

	OUI	**NON**	**PEUT-ÊTRE**
1	1	0	10
2	01	00	10
3	10	1	0
4	0001	0010	0011

Un dictionnaire est l'ensemble des symboles utilisés pour représenter les données.

Dans le codage n°1, le dictionnaire est {0, 1, 10}. Chaque élément de ce dictionnaire est appelé un mot.

- Exemple : le mot « 10 » **est de longueur 2**, car il contient deux symboles binaires.

Il existe donc deux grandes familles de codages :

- **Codage à taille fixe** : tous les mots ont la même longueur (exemple n°4).
- **Codage à taille variable** : certains mots sont plus courts que d'autres (exemple n°1).

L'enjeu est toujours de trouver le codage le plus efficace et économe en ressources pour représenter, stocker et transmettre les informations.

A retenir

- Le code binaire repose sur **deux états** : 0 et 1.
- **Tout peut être représenté** en binaire (texte, son, image, vidéo).
- Un **dictionnaire** associe des données (mots, symboles) à des combinaisons binaires.
- On distingue :
    - les **codages à taille fixe**, simples mais parfois gourmands en espace,
    - et les **codages à taille variable**, plus compacts mais plus complexes à gérer.

Exemple

En binaire pour représenter l'alphabet de couleurs {rouge ; bleu ; jaune ; vert}, les mots doivent être au minium de longueur 2. On peut par exemple, représenter ce dictionnaire comme suit :

	rouge	**bleu**	**jaune**	**vert**
Mots de longueur variable	0	1	10	11
Mots de longueur fixe	11	10	00	01

Remarque : d'autres propositions auraient pu être faites et seraient valables.

Exercice 2.1

a) Imaginez un système de codage permettant de coder en binaire le dictionnaire {A ; B ; C ; D}

A	B	C	D

b) Quelle est la longueur maximale d'un mot de votre dictionnaire ?

c) Est-ce que tous les mots ont la même longueur ?

d) Peut-on décoder sans ambiguïté le code 110110. Dans les deux cas, pourquoi ?

e) Si on ajoute les symboles E et F au dictionnaire, que devient votre codage ?

f) En codant sur une longueur maximale de 3, combien de lettre pourrions-nous ajouter ?

Exercice 2.2

Soit le dictionnaire suivant :

A	B	E	G	H	L	O	S
000	001	010	011	100	101	110	111

Décoder le code binaire suivant : 100010101101110 ?

Donnez deux raisons pour lesquelles ce dictionnaire permet de décoder sans ambiguïté les codes binaires.

1)

2)

Exercice 2.3

Complétez le tableau suivant qui décrit la taille d'un dictionnaire à taille fixe.

Longueur des mots	Nombre de mots possibles	Dictionnaire
1	2	{0 ; 1}
2	4	{00 ; 01 ; 10 ; 11}
3		
4		
n		

Bits, octets et plus

Nous venons de comprendre qu'un ordinateur peut encoder de l'information grâce à deux symboles uniques : **0 et 1**.

C'est la plus petite unité d'information en informatique, appelée bit (abréviation de **bi**nary digi**t**).

Avec un seul bit, on ne peut pas coder beaucoup d'information. Par exemple:

- 0 = NON et 1 = OUI,
- compter de 0 à 1,
- représenter deux couleurs (0 = BLEU, 1 = ROUGE).

Mais on peut faire mieux en regroupant plusieurs bits. Par exemple, avec des **mots de 2 bits**, on obtient **4 combinaisons possibles**.

Mots de 2 *bits*
00
01
10
11

En pratique, les ordinateurs utilisent des **paquets de 8 bits**, appelés **octets** (ou **bytes** en anglais). Cette convention existe depuis longtemps : **lire un octet revient à lire un caractère**.

Des bits au giga

Mais quand vous achetez les tout dernier iPhone, on vous propose la version 128 Go, 256 Go ou 512 Go. Mais qu'est-ce exactement que ces **Go** ?

Dans le **Système international d'unités** (SI), les préfixes se basent sur des multiples de 1000, car on travaille dans le système décimal (base 10).

Exemple en poids

- 1000 g = 1 kg
- 1000 kg = 1 tonne

Appliqué aux octets, cela donne :

- 1 kilooctet (Ko) = **1000 octets**
- 1 mégaoctet (Mo) = **1000 Ko = 1 000 000 octets**
- 1 gigaoctet (Go) = **1000 Mo = 1 000 000 000 octets**

C'est cette définition que les **constructeurs** (Apple, Samsung, fabricants de disques durs, etc.) utilisent pour annoncer les capacités.

En informatique, les ordinateurs travaillent en binaire. La valeur proche de 1000 la plus pratique est **2^{10} = 1024**.

Les unités binaires normalisées (IEC)

En informatique, comme tout est basé sur le binaire, les puissances de 2 sont souvent plus pratiques.

La valeur la plus proche de 1000 est **2^{10} = 1024**.

Pour éviter les confusions, l'IEC (Commission Électrotechnique Internationale) a introduit des unités binaires :

- 1 **kibioctet** (Kio) = 1024 octets
- 1 **mébioctet** (Mio) = 1024 Kio = 1 048 576 octets
- 1 **gibioctet** (Gio) = 1024 Mio = 1 073 741 824 octets
- 1 **tébioctet** (Tio) = 1024 Gio = 1 099 511 627 776 octets

Exemple

Un disque vendu comme 512 Go (décimal) correspond en réalité à environ 476 Gio (binaire) une fois affiché par l'ordinateur.

Nom	*Abréviation SI*	*Taille (SI)*	*Abréviation binaire*	*Taille (binaire)*	*Exemple d'usage*
Octet	o / B	8 bits	o / B	8 bits	
Kilooctet	Ko / KB	1000 o	Kio	1024 o	Un mail sans pièce jointe
Megaoctet	Mo / KB	1000 Ko	Mio	1024 Kio	Une petite photo
Gigaoctet	Go / GB	1000 Mo	Gio	1024 Mio	20 min de film
Teraoctet	To / TB	1000 Go	Tio	1024 Gio	Disque de stockage du PC

Analogie visuelle

- **Bit** (b) : un seul **grain de riz** → la plus petite unité (0 ou 1)
- **Octet** (B) : une petite **cuillère de riz** → 8 grains regroupés
- **Kilooctet** (Ko / Kio) : un **bol** de riz → environ 1000 cuillères
- **Mégaoctet** (Mo / Mio) : un **sac** de riz → environ 1000 bols
- **Gigaoctet** (Go / Gio) : une **palette** de sacs → environ 1000 sacs
- **Téraoctet** (To / Tio) : un **conteneur** entier → environ 1000 palettes

A retenir

- Les **Go, Mo, Ko** (SI) sont utilisés dans le commerce, et ont un facteur 1000 entre eux.
- Les **Gio, Mio, Kio** (binaires) sont utilisés par les ordinateurs, et ont un facteur 1024 entre eux.
- La différence explique pourquoi la capacité affichée est toujours un peu plus faible que celle annoncée.

Exercice 2.4

a) Combien d'octets font 128 bits ?

b) Si une information est encodée sur 49'152 bits, combien de Ko cela représente-t-il ? Et en kibioctets ?

c) Convertissez 2 Go en Kio.

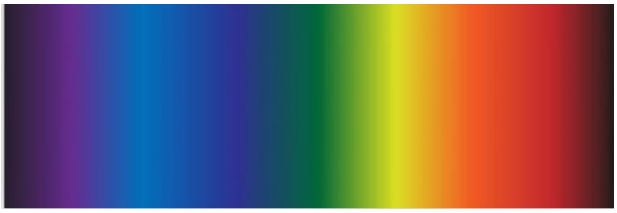
d) Le premier Super Mario Bross sur NES sorti en 1987 faisait 40'960 octets. Le jeu Call of Duty de 2019 sur PC en faisait 235 GB. Combien de fois plus lourd est ce dernier jeu par rapport au jeu NES ?

Peut-on encoder toutes les informations ?

On sait encoder en binaire des nombres, des lettres, des images, du son, de la vidéo, des données biométriques, des données physiques. Mais…

Reprenons l'exemple de notre nouvel iPhone 13 en version 128Go. Ce chiffre représente la taille de son disque de stockage. Sans entrer dans les détails maintenant, on comprend que l'espace de stockage n'est pas infini. Cependant, la nature, elle, ne connait pas de limite !

Prenons un exemple simple, le nombre de couleurs dans la nature est infini or dans un ordinateur on ne peut pas représenter une infinité d'élément. Il faut alors limiter le nombre de couleur que l'on va représenter.



Il se trouve que l'œil humain ne « peut » voir que quelques millions de couleurs. Donc si on est capables de représenter « que » quelques millions de couleurs, on pourra se rapprocher le plus possible de la vraie couleur, et le cerveau humain ne verra pas la différence.

Exercice 2.5

a) Comment pourrait-on représenter les 10 chiffres de notre système décimal humain : {0 ;1 ;2 ;3 ;4 ;5 ;6 ;7 ;8 ;9}

b) Et le nombre 10 ?

Exercice 2.6

Une photo prise par un téléphone fait environ 5 MB. SI votre téléphone dispose d'un stockage de 128 Go mais que le système prend déjà 30% de l'espace, combien de photo pourriez-vous stocker ?

Exercice 2.7

Classez dans l'ordre ces tailles de fichiers :

4.5 Mo – 450 Ko – 4'194'304 o – 1024B – 0.5 Go – 1024b

2.3 Les entiers

Promis ça sera la seule partie qui ressemble à des maths… mais il faut bien un petit peu parler de la base.

L'ordinateur, lui, n'a pas de doigts mais utilise l'électricité. Par conséquent, il ne connaît que deux types d'informations : il y a du courant, il n'y a pas de courant ; allumé, éteint ; vrai, faux ; 1 ou 0.

Pour nous, humains, la numération habituelle est le **système décimal**, ou **base 10**. Pour l'ordinateur, c'est le **système binaire**, ou **base 2**.

Le système décimal

Ce système que vous utilisez au quotidien a un alphabet de de 10 symboles :

{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

La valeur d'un chiffre dépend à la fois de ce qu'il représente et de sa position.

Exemple : dans le nombre 1934, le chiffre 3 vaut 30, tandis que dans 3008, le 3 vaut 3000.

C'est ce qu'on appelle une **représentation positionnelle**.

Pour décomposer un nombre en base 10, on utilise sa **forme canonique**.

Exemple

La décomposition du nombre 3528 est :

Milliers	Centaines	Dizaines	Unités
$10^3 = 1000$	$10^2 = 100$	$10^1 = 10$	$10^0 = 1$
3	5	2	8

Sa forme canonique est alors :

$$3 \bullet 10^3 + 5 \bullet 10^2 + 2 \bullet 10^1 + 8 \bullet 10^0$$

Le système binaire

Le système binaire (base 2) fonctionne exactement de la même manière, mais avec seulement **deux symboles : 0 et 1**.

Un élément binaire se nomme un bit et un ensemble de bits peut représenter un entier en utilisant le même principe que pour le système décimal.

Avec le système décimal, qui contient 10 symboles, on représente les nombres entiers en ajoutant 1 unité à chaque fois ; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Lorsqu'on atteint neuf, le dernier et dixième symbole, on ajoute alors 1 aux dizaines puis on remet à 0 les unités, ce qui donne 10.

En binaire c'est exactement la même chose mais avec deux symboles. Ce qui donne :

Binaire	Décimal
0	0
1	1
10	2
11	3
100	4
101	5

On voit ici que le principe est donc le même que celui qu'on a appris pour compter dans la vie de tous les jours, mais avec deux symboles au lieu de 10.

En effet, prenons le passage de 1 à 10, soit de 1 à 2 en décimal.

Instinctivement, nous voudrions dire que si on ajoute 1 à 1, on obtient 2. Or dans le système binaire le symbole 2 n'existe pas. Donc on doit mettre à un 0 dans la colonne de droite et un 1 dans la colonne de gauche.

Exemple visuel :

$$\begin{array}{rr} & {\scriptstyle 1} \\ & 11 \\ + & 1 \\ \hline & 100 \end{array}$$

Remarque n°1

En mathématique, on appelle cela les bases. Notre système décimal est la base 10, car nous comptons à dix doigts. Le système binaire est la base 2. Mais il existe toutes les bases imaginables. On peut alors compter en base 3, 5, 7 ou même 16, ce qu'on nomme l'hexadécimal. Ce dernier est très utilisé également en informatique.

Base 10	**Base 2**	**Base 3**	**Base 7**	**Base 16**
9	1001	100	12	9
10	1010	101	13	A
15	1111	120	21	F
26	11010	222	35	1A

Remarque n°2

Un mot binaire, par exemple 11011001, peut représenter un nombre entier mais peut aussi représenter une tout autre information. Un ordinateur sait à chaque fois qu'il interprète un octet, ce que celui-ci représente.

C'est identique à un humain qui lit ce document : il est capable d'interpréter aussi bien le texte, les nombres que les images.

Exercice 2.8

a) Ecrivez en binaire de 0 jusqu'à 20.

Décimal	Binaire
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Décimal	Binaire
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

b) Que pouvez-vous observer sur la parité des nombres binaires ?

Conversion du binaire au décimal

Tout comme dans le système décimal, la valeur de chaque chiffre dépend toujours de sa place, qui représente, cette fois, une puissance de 2.

Exemple

La décomposition du nombre $1101_{(2)}$ est :

$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
1	1	0	1

Sa forme canonique est alors :

$$1 \bullet 2^3 + 1 \bullet 2^2 + 0 \bullet 2^1 + 1 \bullet 2^0$$
$$= 8 + 4 + 0 + 1$$
$$= \mathbf{13_{(10)}}$$

*Notes : pour différencier les deux systèmes on note (2) ou (10) en **indice** pour savoir dans quel système on se trouve. Par exemple $11_{(10)}$ ou $11_{(2)}$.*

On peut donc utiliser un tableau de conversion facile à écrire pour convertir un nombre binaire en décimal :

Coefficient	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeur	256	128	64	32	16	8	4	2	1
Bit									

Si l'on souhaite convertir 01101000(2), on remplit donc le tableau ci-dessus et on additionne les valeurs des coefficients là où les bits sont à 1 :

Coefficient	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeur	256	128	**64**	**32**	16	**8**	4	2	1
Bit			**1**	**1**	0	**1**	0	0	0

$$1101000_{(2)} = 64 + 32 + 8 = 104_{(10)}$$

On appelle **bit de poids faible** le bit le plus **à droite**, car il a la valeur la plus faible, tout comme l'unité pour le système décimal. Il faut donc **écrire le nombre binaire de droite à gauche** dans le tableau de conversion.

Exercice 2.9

Donnez la représentation décimale des nombres binaires suivants :

Binaire	Décimal
1111	
10	
101011	
0001010	
100000000	
100000010	

Exercice 2.10

Quel est le plus grand nombre entier décimal que l'on peut représenter sur 16 bits ?

Conversion du décimal au binaire

L'opération inverse est un plus complexe mais deux méthodes existent : tableau de conversion ou un système plus mathématique.

Méthode 1

On part du même tableau de conversion qu'avant :

Coefficient	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Valeur	1024	512	256	128	64	32	16	8	4	2	1
Bit											

1. Déterminer le coefficient maximum dont la valeur est plus petite que l'entier à convertir.
2. Le bit associé à ce coefficient est 1.
3. Soustraire la valeur de ce coefficient à l'entier à convertir pour obtenir le nouveau nombre à convertir.
4. Recommencer à l'étape 1 tant que le nombre est différent de 0.
5. En commençant par le plus grand coefficient utilisé, le nombre binaire correspondant est composé de la suite des bits où des 0 représentent les coefficients non utilisés.

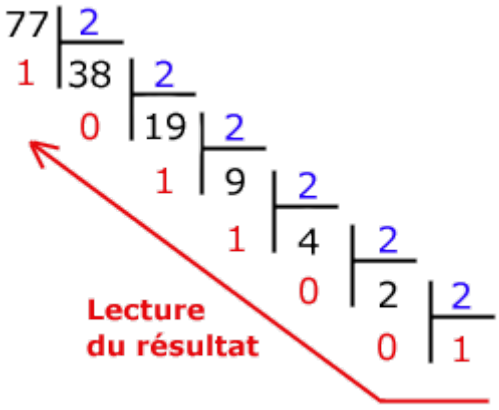
Par exemple, la conversion du nombre décimal 666 en binaire s'obtient avec les étapes suivantes :

$$\begin{aligned} 666 &= 512 + \mathbf{154} \\ 154 &= 128 + \mathbf{26} \\ 26 &= 16 + \mathbf{10} \\ 10 &= 8 + \mathbf{2} \\ 2 &= 2 + 0 \end{aligned}$$

Coefficient	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Valeur	1024	512	256	128	64	32	16	8	4	2	1
Bit		1	0	1	0	0	1	1	0	1	0

Méthode 2

Il s'agit de faire une suite de division par 2 et le résultat sera la combinaison des restes.



Donc $77_{(10)}$ vaut $1001101_{(2)}$.

Exercice 2.11

Donnez la représentation binaire des nombres décimaux suivants :

Décimal	Binaire
14	
89	
129	
1204	

Encodage avec des octets

Nous avons vu que de manière généralisée on encode par bloc de 8 bits, soit 1 octet, les données en informatique. Mais alors le nombre le plus grand que l'on pourrait représenter serait combien ?

Soit le nombre **1111 1111**, sur 1 octet. Sa valeur décimale est alors :

$$11111111_{(2)} = 2^8 - 1 = 255_{(10)}$$

Cela serait bien peu pour effectuer tous les calculs dont nous avons besoin. C'est alors que chaque système doit prendre des décisions lors de sa conception sur la manière de représenter les nombres, par exemple. On par exemple choisir de coder les entiers avec 4 octets, soit 32 bits.

Le plus grand nombre alors que l'on peut représenter sera $2^{32} - 1 = 4\ 294\ 967\ 295$, soit un peu plus de 4 milliards.

Exercice 2.12

Le clip YouTube le plus visionné est le titre Baby Shark de Pinkfong sorti en 2016. Il a cumulé plus de 15 milliards de vues !!!
Combien d'octets sont-ils nécessaires pour encoder ce nombre de vues ?

Dépassement de capacité

Nous avons vu jusqu'ici comment les nombres sont codés dans un ordinateur, qu'aujourd'hui nous regroupons en octet les paquets d'information et nous savons additionner des nombres binaires entre eux.

Admettons qu'un système choisisse d'encoder les entiers sur 1 octet. Le plus grand nombre que l'on peut représenter est donc $11111111_{(2)}$ soit le nombre $255_{(10)}$.

Que se passe-t-il maintenant si on ajoute 1 à ce nombre binaire ?

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1} \\ 11111111 \\ +\qquad\qquad 1 \\ \hline 100000000 \end{array}$$

Le nombre obtenu ici est bien $256_{(10)}$ mais il nécessite 9 bits. Or si le système est conçu sur 8 bits le nombre final sera en réalité $00000000_{(2)}$ soit le nombre zéro !

C'est ce qu'on appelle un **dépassement de capacité** car le système à atteint sa limite et nous avons dépassé sa capacité de stockage.

Pourquoi Gangnam Style a cassé le compteur de YouTube

En 2014, la vidéo du chanteur PSY Gangnam Style affichait un nombre de vue qui n'avait aucun sens. Le site ne pouvait plus comptabiliser les vues après avoir dépassé très exactement les 2.147.483.647. En effet, YouTube avait opté pour un encodage 32 bits pour les compteurs de vues, ce qui est le nombre le plus grand que l'on peut représenter sur 32 bits.

Depuis, ils ont opté pour un codage sur 64 bits soit plus de 9 milliards de milliards. D'ici là, on aura tous oublié ce titre et YouTube…

Et les nombres entiers négatifs ?

La première idée pour représenter un entier relatif est d'utiliser **un bit pour représenter le signe**. En effet, un bit peut uniquement prendre deux valeurs, 0 ou 1, comme le signe, + ou -. Les bits restants étant utilisés pour représenter un entier positif.

Considérons l'encodage sur un octet (8 bits), nous réservons le bit de poids fort (la puissance de 2 la plus grande) pour le signe : **0 indique un nombre positif** et **1 un nombre négatif**.

Avec cette approche, un entier relatif est représenté par sa valeur absolue (entier naturel) auquel on associe un signe. Le domaine couvert se trouve donc divisé par deux, un bit étant utilisé pour le signe.

Ainsi, avec un octet, 7 bits sont utilisés pour encoder la valeur absolue, soit [0 ; 127], ce qui permet de représenter les entiers relatifs dans l'intervalle [-127 ; 127].

Exemple

La représentation de -33 est alors :

Signe	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeur	*64*	*32*	*16*	*8*	*4*	*2*	*1*
1	0	1	0	0	0	0	1

Bien que cette approche soit simple et semble convenir, elle pose deux problèmes majeurs :

- Le premier est d'avoir deux représentations pour zéro (+0 et -0),
- Le second apparait lorsque l'on additionne un nombre et son opposé. Le résultat de cette opération devrait être 0, mais, sans rentrer dans le détail de l'arithmétique binaire qui sera abordée plus tard, l'addition bit à bit avec cette représentation donne $(-2|x|)$. En effet, l'addition des bits de signe donne toujours 1 et le reste des bits représente l'addition de deux fois la valeur absolue.

Pour remédier ces problèmes, une autre représentation des entiers relatifs a été mise au point, il s'agit de la représentation en complément à deux.

Complément à deux

La représentation en complément à deux reprend l'idée du **signe encodé par le bit de poids fort** et conserve la représentation des entiers positifs. Donc tous les entiers positifs ont une représentation en binaire qui commence par un 0 et le plus grand entier représenté sur un octet est 127.

Les entiers négatifs sont représentés grâce au complément à deux dont voici le principe :

1. Écrire la valeur absolue du nombre en binaire
2. Inverser tous les bits, les 0 deviennent des 1 et vice-versa. Le résultat obtenu s'appelle le complément à 1 du nombre de départ.
3. On ajoute 1, la dernière retenue est ignorée le cas échéant.

Exemple

La représentation de -33 en complément à deux :

1. Valeur absolue en binaire : 00010001
2. Complément à 1 : on inverse tous les bits : 11101110
3. Ajouter 1 : 11101111

Exercice 2.13

Donnez la représentation binaire sur 5 bits des nombres décimaux suivants :

Décimal	Bit de signe	Complément à 2
-12		
-3		
-32		
-55		

Exercice 2.14

Avec un octet, quel est l'ensemble des entiers couvert en utilisant le complément à deux ?

2.4 Les caractères

On sait désormais encoder des nombres entiers avec des 0 et des 1. Mais comment faire maintenant pour encoder des caractères pour par exemple envoyer un email ou faire un commentaire sur Instagram ?

La solution est comme nous l'avions vu dans l'introduction, on associe chaque caractère à un code binaire sur un octet.

Caractère	Décimal	Binaire
A	65	01000001
B	66	01000010
C	67	01000011
…	…	…
Z	90	01011010
0	48	00110000
1	49	00110001
9	57	00111001

Chaque caractère frappé sur le clavier est représenté par le code correspondant dans ce tableau.

Chacun des caractères de la phrase que vous lisez (qu'on nomme **chaîne de caractères**) a ainsi été stocké, transmis et manipulé par l'ordinateur sous la forme d'une séquence de 0 et 1.

Lorsqu'il s'agit de représenter ce texte à l'écran ou à l'impression, les logiciels utilisent la table dans l'autre sens pour trouver le caractère correspondant au nombre binaire.

En plus des lettres, les caractères qui représentent **les chiffres sont eux-mêmes listés dans la table de conversion**, étant donné qu'ils se trouvent sur notre clavier. Contre-intuitivement, la valeur binaire du caractère représentant un chiffre ne correspond pas au chiffre lui-même.

Ces tables donnent également une représentation des caractères de **ponctuation** et des **symboles mathématiques**, ainsi que des **caractères non-imprimables** comme le retour à la ligne.

Table ASCII

Au début de l'informatique, aucun constructeur ne s'échangeait d'information sur la manière dont ils codaient les données dans leurs machines. C'était la course chacun pour soi, et donc il existe des dizaines de tables de conversion. Cela a vite conduit au fait que les machines de différents constructeurs ne pourraient jamais communiquer entre-elles…

Le code américain normalisé pour l'échange d'information ASCII (pour American Standard Code for Information Interchange) est apparu dans les années 1960. Malgré sa large acceptation, avec ses **7 bits par caractère**, cette table avait pour principal défaut de ne pas prendre en compte les caractères qui n'existent pas dans la langue anglaise, ne serait-ce que les lettres accentuées. On y voit une valeur décimale, sa valeur binaire et le caractère que cela représente. *Notez que sur un octet, le code ASCII commence toujours par un 0.*

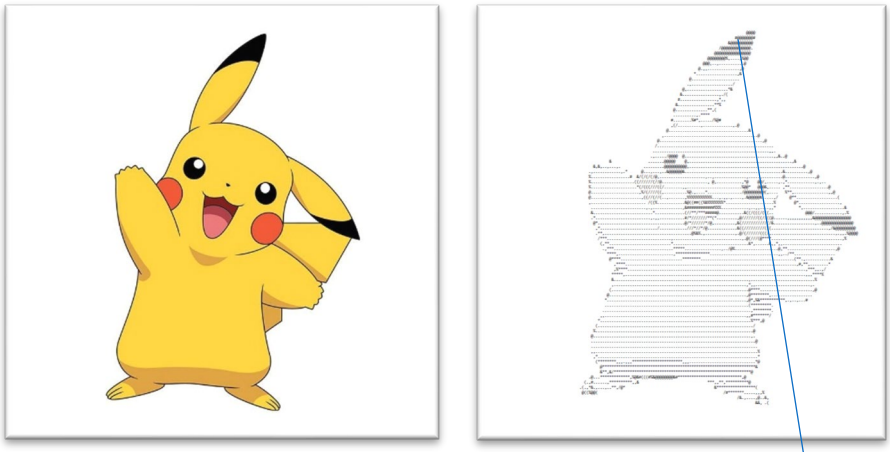
ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

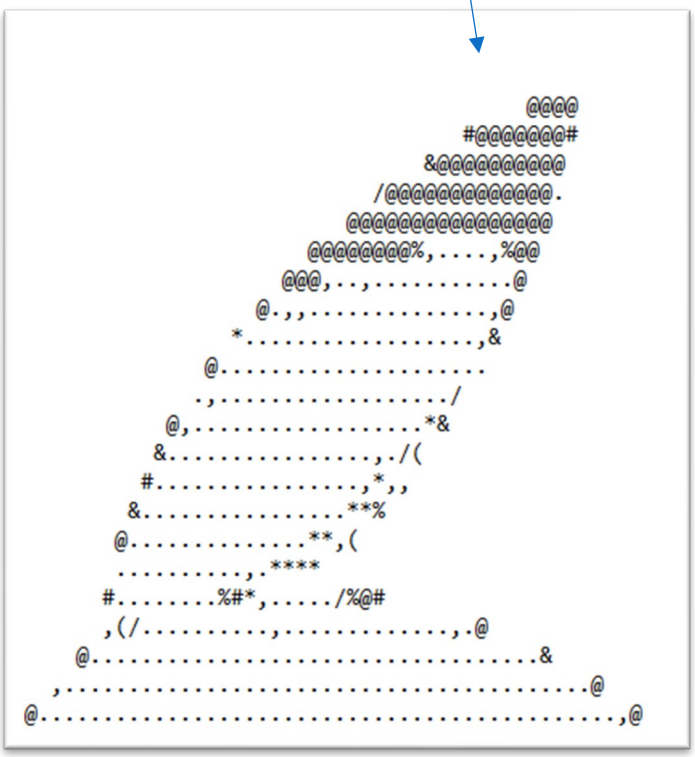
L'ASCII Art

Depuis l'introduction de l'encodage ASCII jusqu'à aujourd'hui, une forme d'art s'est répandue que l'on appelle l'ASCII ART. Il s'agit de créer une image uniquement avec les caractères de la table ASCII. Aujourd'hui certain vont bien plus loin en créant même des animations.

Voilà un exemple de transformation d'une image en ASCII Art



Ci-contre un zoom sur l'oreille de Pikachu. On voit que c'est l'utilisation de caractères comme @, /, &, * qui permettent de faire des nuances de gris sur l'image.



UTF : la standardisation

Des tables multiples, mutuellement incompatibles, ont alors émergé : une table pour les européens, une autre pour les Japonais et ainsi de suite.

Progressivement, notamment avec l'émergence du Web au cours des années 1990, l'augmentation de l'interconnexion des ordinateurs personnels a amené au début des années 2000 à la mise en place d'une énorme table intégrant le contenu de toutes les tables existantes, via le standard UTF.

Le standard **Unicode UTF** (Universal Character Set Transformation Format) s'est imposé pour l'échange, car il permet d'agréger sur 8 bits, 16 bits ou 32 bits par caractère la totalité des caractères utilisés dans toutes les langues.

Avec 32 bits on peut encoder 2^{32} caractères soit plus de **4,3 milliards** de signes !

Les caractères liés à l'édition des partitions de musique ou les émojis sont également intégrés.

Variantes

Pour éviter de consommer 32 bits par caractère, des variantes plus compactes ont été mises à disposition.

La plus connue – des européens, puisqu'elle regroupe les caractères qui nous concernent – est la table **UTF-8**[5]. C'est un encodage des caractères **de longueur variable qui utilise de 1 à 4 octets par symbole**.

L'idée principale a été de partir de la table ASCII et ses 128 signes et de l'augmenter de tous les signes nécessaires de toutes les langues ainsi que d'autres caractères comme les Emoji. Par exemple, le *é* a comme valeur décimale 233.

[5] Annexe 1

COMMENT S'OPÈRE LE CODAGE SUR PLUSIEURS OCTETS ?

Nous appelons **point de code** la valeur numérique dans la table UTF d'un caractère.

Caractère	Décimal	Binaire	Nb. De bit nécessaire
A	65	1000001	7
é	233	11101001	8
€	8364	10 0000 1010 1100	14
😂	128514	1 1111 0110 0000 0010	17

On a dit que l'UTF-8 code les caractères sur une longueur de 1 à 4 octets ceci afin de gagner en place. En effet, pourquoi utiliser 4 octets pour encoder la lettre A qui s'encode sur 7 bits.

Il faut alors définir des règles afin que la machine puisse comprendre si elle doit lire 1, 2, 3 ou 4 octets par caractère.

Représentation binaire UTF-8	Signification
0xxxxxxx	1 octet codant 1 à 7 bits
110xxxxx **10**xxxxxx	2 octets codant 8 à 11 bits
1110xxxx **10**xxxxxx **10**xxxxxx	3 octets codant 12 à 16 bits
11110xxx **10**xxxxxx **10**xxxxxx **10**xxxxxx	4 octets codant 17 à 21 bits

Ce codage multi-octet est structuré de telle manière qu'en lisant de gauche à droite chaque octet on peut savoir s'il s'agit d'un octet qui code un point de code sur un ou plusieurs octets, ou s'il s'agit de la continuation du codage d'un point de code. En clair si l'octet commence par :

- **0** c'est un code à un seul octet
- **110** c'est que le message est codé sur 1+1=2 octets et que donc un deuxième octet le suit
- **1110** c'est que le message est codé sur 1+1+1=3 octets et que donc deux octets le suivent
- **1111** c'est que le message est codé sur 1+1+1+1=4 octets et que donc quatre octets le suivent
- **10** c'est que c'est un octet qui constituant le code

Tous les xxxxx représentent ce qu'on appelle la **charge utile**, c'est-à-dire **l'encodage du caractère** a proprement parlé.

Exemple 1

Le caractère **A** a comme valeur décimale dans la table UTF **65**.

En binaire 65 s'écrit 1000001, soit sur 7 bits.

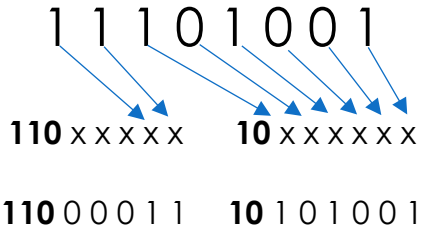
Donc le code UTF8 sera sur un seul octet commençant par un 0, puis la représentation binaire du caractère, ce qui donne **01000001**.

Exemple 2

Le caractère **é** a comme valeur décimale dans la table UTF **233**.

En binaire 233 s'écrit 11101001, soit sur 8 bits. Il faut alors 2 octets pour l'encoder en UTF8 sous la forme **110**xxxxx **10**xxxxxx

Les 11 x vont être remplacé par les 8 bits de du caractère, complétés par des 0 devant.



Exemple 3

Le caractère **€** a comme valeur décimale dans la table UTF **8364**.

En binaire 8364 s'écrit **100000 10101100**, soit sur 14 bits. Il faut alors 3 octets pour l'encoder en UTF8 sous la forme **110**xxxxx **10**xxxxxx **10**xxxxxx, soit 11100010 10000010 10101100

Puisqu'on peut dire quel type d'octet on regarde à partir des premiers bits du premier octet à gauche, alors même si quelque chose est altéré quelque part, la séquence entière n'est pas perdue : ce codage est appelé ***codage auto-synchronisant***.

Conclusion

Ce chapitre avait pour but de vous montrer que derrière un simple message WhatsApp, un système particulièrement complexe opère.

La phrase « Tu peux être à 15h? »[6] doit être d'abord codée sur votre téléphone en :

01010100 01110101 00100000 01110000 01100101 01110101 01111000 00100000 11000011 10101010 01110100 01110010 01100101 00100000 11000011 10100000 00100000 00110001 00110101 01101000 00111111

Puis cette information doit être transmise à un serveur par WiFi, qui transmettra lui-même cette information à un serveur aux USA qui renverra cela sur un réseau en Suisse pour finalement être décoder par le smartphone de votre ami.e. Tout cela en moins de 5 secondes.

Exercice 2.15

1. À l'aide de la table ASCII, codez en binaire votre prénom

2. Décodez cette exclamation en binaire : 01000010 01110010 01100001 01110110 01101111 00100001.

3. Peut-on coder en binaire la phrase « Un âne est-il passé par là ?» à l'aide de la table ASCII (justifiez la réponse) ?

[6] Coder en ligne des messages : https://fr.planetcalc.com/9029/

Exercice 2.16

Le symbole Ø correspond à la valeur décimale 8709.

1. Convertissez cette valeur en binaire.

2. Combien d'octets doit-on utiliser en UTF-8 pour coder ce nombre convenablement (les moitiés d'octet sont interdites) ?

3. Donnez le codage UTF-8 correspondant.

Exercice 2.17

Trouvez les deux erreurs de transmission dans ce code UTF8 et justifiez.

01000100 11000011 00101000 01110011 10100000 00110001
00110000 00100000 01100001 01101110 01110011

2.5 Les images

Depuis des siècles l'être humain conservent des traces visuelles de son environnement, depuis les grottes de Lascaux à Instagram, en passant par la Camera Obscura.

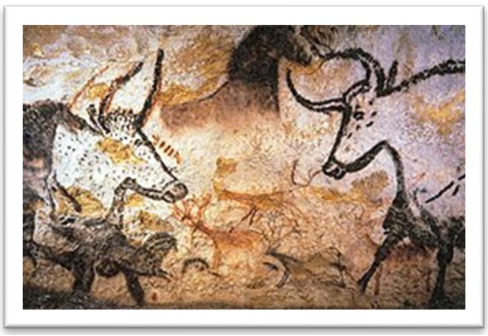


Figure 9 - Fresque de Lascaux

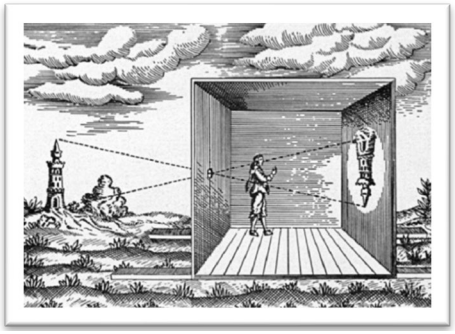


Figure 10 - Principe de la Camera Obscura

Notre œil fonctionne comme la camera obscura : la lumière est réfléchie sur un objet, puis son faisceau est envoyé à notre œil qui passe par une lentille, le cristallin, et l'image se forme sur notre rétine. L'information est ensuite envoyée au cerveau qui la décode.

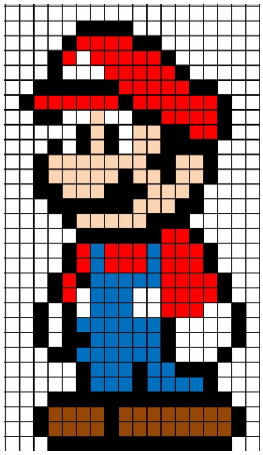
Pour une machine cela va se passe de la même manière. Mais avant tout c'est quoi une image numérique ?

Zoom sur une image numérique

Sur votre téléphone portable on ne peut pas zoomer assez mais faites le test sur un ordinateur. Zoomer au maximum sur une photo. Voici ce que cela donne :

Figure 11 - Georges Seurat, La Seine à la Grande Jatte.
Printemps (huile sur toile, 1888)

A ce niveau de zoom on se rend compte que notre image est en réalité constituée de d'une multitude de petits carrés colorés. C'est ce qu'on appelle des pixels, mot qui vient de l'anglais « **pic**ture **el**ement », ou élément d'image en français.



Donc avec ce que l'on a vu auparavant, on peut stocker et coder une image très facilement dans un ordinateur, sachant qu'elles sont toutes rectangulaires.

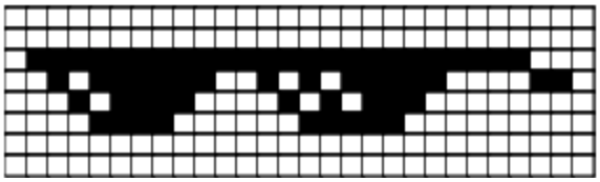
Une image est donc un tableau, ou matrice, constitué de **colonnes** et de **lignes** ; on parle alors d'**image matricielle**. Chaque élément de cette matrice constitue donc un pixel, et ce pixel contient une information sur sa couleur, codée bien sûr en binaire.

Les images noir-blanc

Nous savons que le *bit* est l'unité d'information la plus petite. Donc l'image la plus simple que l'on peut coder est une image en noir et blanc.

Blanc = 1 Noir = 0

Voici une représentation d'une image numérique en NB, et sa matrice binaire correspondante.



```
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0],
[0,0,1,0,1,1,1,1,1,1,0,0,1,0,1,0,1,1,1,1,1,1,0,0,0,0,1,1,0],
[0,0,0,1,0,1,1,1,1,1,0,0,0,0,1,0,1,0,1,1,1,0,0,0,0,0,0,0,0],
[0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
```

Donc dans une image noir-blanc, **chaque pixel détient 1 bit d'information**. Les premières consoles d'arcade étaient en NB, comme le fameux jeu PONG était alors facile à programmer, mais notre Mario n'a pas un rendu très satisfaisant en NB.

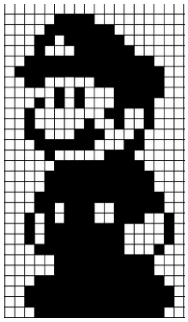
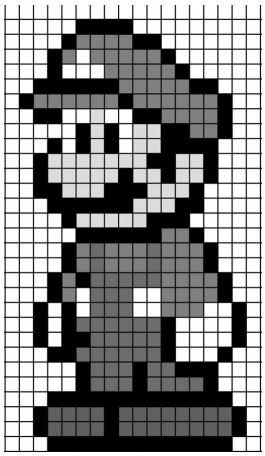


Image en nuance de gris



Si on veut faire un peu mieux, il nous faut des nuances de gris. D'ailleurs aujourd'hui le mode Noir-Blanc des appareils photos de nos smartphones font des nuances de gris, ce qui donne souvent un effet artistique très apprécié.

Voici notre Mario en nuances de gris. On peut déjà mieux voir les détails.

Dans ce type d'image seul le niveau de l'intensité est codé sur un **octet** (256 valeurs). Par convention, **la valeur 0 représente le noir** (intensité lumineuse nulle) et **la valeur 255 le blanc** (intensité lumineuse maximale).

0	*8*	*16*	*32*	*56*	*72*	*90*	*104*	*112*	*128*
136	*144*	*160*	*176*	*192*	*208*	*224*	*244*	*248*	*255*

Pour encoder une image en niveaux de gris, chaque pixel nécessite donc 8 bits, soit 1 octet.

Image en couleurs

Et comment fait-on alors pour les images en couleurs ? Une fois de plus on s'est inspiré des autres sciences ou arts. En peinture on pour créer l'ensemble des couleurs on utilise la synthèse soustractive avec le cyan, le magenta et le jaune. Ces pigments vont absorber une partie de la lumière et donc soustraire une part de couleur. Le mélange des trois donne du noir, et si on ôte totalement le jaune on obtient du bleu.

Pour les écrans, on utilise également trois couleurs, le **Rouge**, le **Vert** et le **Bleu**, les couleurs dites **primaires**. On utilise alors la **synthèse additive** pour produire toutes les couleurs. Le mélange des trois donne du blanc et en mélangeant du rouge et du vert on obtient du jaune. En informatique ce système de couleur s'appelle **RGB** pour Red-Green-Blue.

Un pixel doit donc contenir l'information des trois couleurs et sera donc codé sur 3 octets, un pour chaque couleur.

Exemple

Couleur	R	G	B
Rouge	255	0	0
Vert	0	255	0
Bleu	0	0	255
Jaune	255	255	0
Orange	255	165	0
Rose	255	192	203

Par convention on écrit les couleurs sous la forme **rgb(255,192,203)**, pour un rose par exemple. C'est donc comme si on disposait de 255 gouttes de chaque couleur que l'on peut mélanger. C'est ce qu'on appelle les **vraies couleurs** en informatique et une couleur est donc encodée sur 24 bits.

On peut donc représenter 256^3 = 16 millions de couleurs.

Cependant, on pourrait utiliser moins de couleur pour représenter une image. Il existe des codages différents, par exemple sur 16 bits ce qui correspond à 65'000 couleurs, ou à l'inverse utiliser 32 ou 64 bits pour disposer d'encore plus de nuances. On appelle cet encodage la **profondeur de l'image**.

Code Hexadécimal

Vous verrez très probablement les couleurs écrites parfois sous une autre forme, comme **#FFC0CB** pour notre rose ci-dessus. Dans Word, Photoshop, sur Internet.

Ceci est le code hexadécimal. C'est la représentation de 255, 192 et 203 en base 16. Cette base est énormément utilisée par les informaticiens pour simplifier la lecture du code binaire. Pour ceux que cela intéresse, je vous laisse trouver cela sur internet, mais au lient de puissance de 2, on aura des puissances de… 16.

Définition, dimensions et taille d'une image

La définition d'une image n'est rien d'autre que son nombre de pixels. Une image de **dimension** 640 pixels de large par 320 de haut a une **définition** de

640 x 320 = 204'800 pixels

Cela nous permet de connaitre ensuite sa **taille**, ou son **poids** numérique, dépendamment du type d'image que nous avons (NB, nuances de gris, couleurs).

Exemple

Cette même image aura :

	Nb. De bit/pixel	Taille [b]	Taille [Ko]
NB	1	204'800	25
Nuances de gris	8	1'638'400	200
Couleur	3 x 8 = 24	4'915'200	600

Exercice 2.18

Calculer, pour chaque définition d'image et chaque couleur, la taille mémoire nécessaire à l'affichage.

Largeur	Hauteur	Définition	Noir Blanc	256 Gris	Vraie couleur
320	200				
3000	4000				

Trouvez la taille des photos prises par votre smartphone. Est-ce que la taille correspond au calcul ? Si non pourquoi selon vous ?

Résolution d'une image

Attention ce terme est très souvent confondu avec la définition d'une image. La résolution traduit la **qualité de l'image**. Elle est exprimée en DPI (Dots Per Inch) ou PPP (Point Par Pouce), un pouce représentant **2.54 cm**. La résolution permet ainsi d'établir le **rapport entre le nombre de pixels d'une image et la taille réelle de sa représentation sur un support physique**. Une résolution de 300 dpi signifie donc 300 colonnes et 300 rangées de pixels sur un pouce carré ce qui donne donc 90000 pixels sur un pouce carré.

ATTENTION : **un pixel n'a pas de dimension a priori**, c'est juste un élément qui stock une information de couleur. Sa dimension va dépendre du support sur lequel on va imprimer l'image.

On peut traduire la résolution par le calcul suivant :

Résolution = Nombre de pixels en largeur/ Taille de l'image en largeur

Pour mieux se figurer ce que cela veut dire, il faut imaginer imprimer une image sur du papier. Pour que la qualité, donc le détail de l'image, soit bonne, pour une même dimension d'impression, il faut pouvoir avoir plus de pixels par centimètre, donc une résolution plus haute. Sinon l'image sera pixelisée.

C'est comme le pointillisme en Art. Si ce tableau avait été créer avec des pinceaux larges, l'artiste aurait pu mettre moins de points par centimètre donc la qualité du rendu aurait été mauvaise voire catastrophique.

Les standards de résolution pour l'impression sont de 300 dpi alors qu'une image destinée au web (ou smartphone) nécessite que 72 dpi.

Résolution YouTube

Résolution	Largeur	Hauteur
144p	256	144
480p	640	480
720p	1280	720
1080p	1920	1080
4K	3840	2160

Les formats d'image

Pour ces images matricielles, il existe beaucoup de format différent que l'on peut repérer au nom du fichier et à son extension. Par exemple, pour un fichier « Image.jpeg » son extension et JPEG ce qui nous indique que c'est une image.

Les formats connus sont *jpeg* ou *jpg*, *gif*, *png*, *bmp*, ou *tiff*.

Mais pourquoi y en a-t-il autant ? Le format originel est le BMP ou bitmap. Mais on a vite été confronté à un besoin de réduire la taille des fichiers images pour des questions de transmissions. Votre téléphone prend des photos de dimension 4000x3000 pixels. Si l'on reprend ce qu'on a vu :

	Nb. De bit/pixel	**Taille [Mo]**
NB	1	1.4
Nuances de gris	8	11.4
Couleur / BMP-8	24	**34.3**

Or sur votre téléphone, cette image même en couleur ne pèse en fait que 3Mo ! L'image a été **compressée** !

Chacun de ses formats, mis à part le BMP, compresse les images selon des algorithmes qui leurs sont propres. La compression peut être :

- sans perte : l'image comprimée est parfaitement identique à l'originale. On va optimiser le codage pour utiliser le moins de bit que possible.
- avec perte : l'image est plus ou moins dégradée, selon le taux de compression souhaité.

Par exemple, si on utilise le système dit des **vraies couleurs**, soit 16 millions de couleurs, ce qui peut être bien trop pour représenter ce que l'œil humain peut distinguer. La compression peut donc transformer l'encodage sur 24 bits en 16 bits et gagner en poids, tout en perdant inévitablement de l'information.

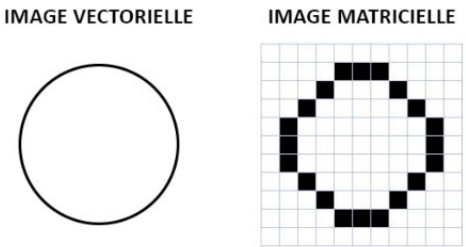
Ces formats apportent différentes propriétés à l'image. Par exemple, le format PNG permet de gérer la transparence d'une image. Un fond d'image transparent en PNG sera rempli en blanc dans les autres formats.

Le format GIF permet d'obtenir des images animées. Le format actuellement le plus utilisé par les appareils numériques est le **JPEG** car il a un très bon taux de compression tout en ne dégradant que très peu l'image.

Les images vectorielles

Il existe une autre manière de représenter les images que par une matrice de pixels. Il est en effet également possible de définir une image comme une collection d'objets graphiques élémentaires (un segment, un carré, une ellipse…) sur un espace plan : c'est le principe des images vectorielles.

L'image vectorielle est dépourvue de matrice. Elle est en fait créée à partir d'équations mathématiques. Cette image numérique est composée d'objets géométriques individuels, des primitives géométriques (segments de droite, arcs de cercle, polygones, etc.), définies chacune par différents attributs (forme, position, couleur, remplissage, visibilité, etc.).



À l'inverse de l'image matricielle composée de pixels, l'image vectorielle peut être redimensionnée sans pour autant perdre en qualité. Elle est contenue dans un **fichier beaucoup plus léger** qu'une image pixelisée, **indépendamment**

de sa taille et de sa résolution. En revanche, chaque forme d'une image vectorielle est remplie d'une seule couleur dite solide ou d'un dégradé de couleurs. Elle reste donc limitée en termes de réalisme, et donc inutilisable en photographie par exemple. De plus une image vectorielle ne peut être créée qu'à partir d'un logiciel dédié, et n'est pas reconnue par les navigateurs internet.

Les formats vectoriels les plus courants sont Postscript (.ps) et Encapsulé Postscript (.eps), Adobe Illustrator (AI), Portable Document Format (PDF), WMF (format Windows).

Exercice 2.19

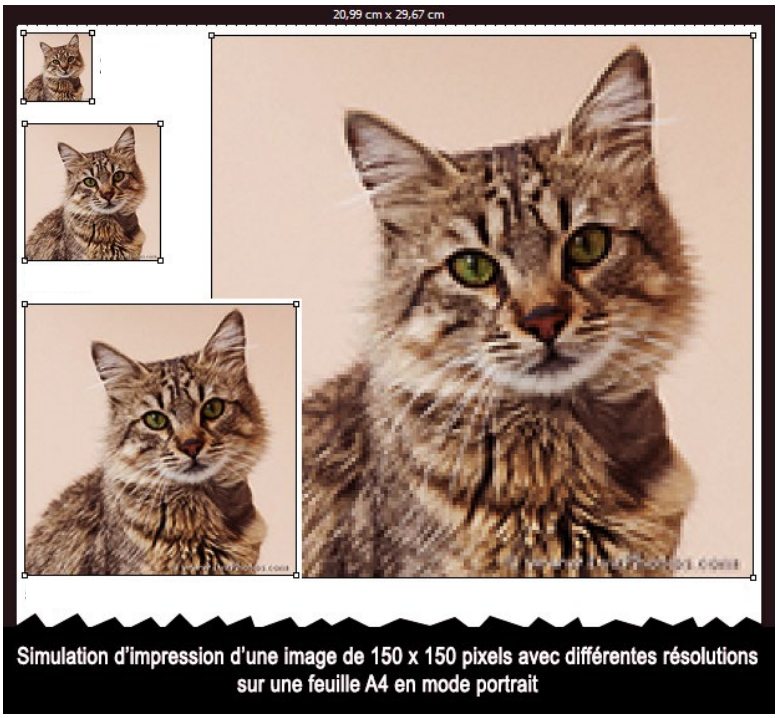
Calculez la dimension d'un pixel en millimètres et en micromètres pour une résolution de 300 dpi et pour une de 72 dpi, lors de l'impression d'une image.

Exercice 2.20

1. Une image numérique de définition 1024×768 mesure 30 cm de large et 20 cm de haut. Déterminez les dimensions des pixels.

2. On a une photographie de 10 cm sur 5 cm que l'on scanne avec une résolution de 300 ppi. Quelle sera alors la définition de l'image (en nombre de pixels) ?

3. Ci-dessous, une simulation d'impressions d'une image de 150x150 pixels à différentes résolutions, sur une page A4. Les résolutions sont de 25, 50, 100 et 200 dpi. Indiquez à côté des images, quelle résolution lui correspond. Pourquoi la grande est-elle pixellisée ?



Simulation d'impression d'une image de 150 x 150 pixels avec différentes résolutions sur une feuille A4 en mode portrait

2.6 Le son

Le numérique a permis bien évidemment de calculer, de simuler, de résoudre des problèmes, puis on a pu remplacer les machines à écrire, représenter des images. Il ne manque plus qu'à voir comment on peut encoder et stocker le son puis pouvoir le transmettre via des applications comme YouTube ou Spotify.

Mais c'est quoi le son en physique ?

Un son est une histoire d'énergie et de vibrations. Un son émerge quand des molécules subissent une pression initiale, ce qui va les amener à avancer et entrainer ce mouvement sur les molécules devant immédiatement voisines en leur transmettant une grande partie de cette énergie. Suite à ce nouveau mouvement, elles repartent en arrière pour retrouver leur position d'équilibre ayant transmis cette énergie initiale aux molécules voisines qui à leur tour vont se comporter de la même manière.

Figure 12 – Ondes de l'eau

On peut voir le son comme une onde d'eau[7] lorsqu'on jette un caillou dans l'eau.

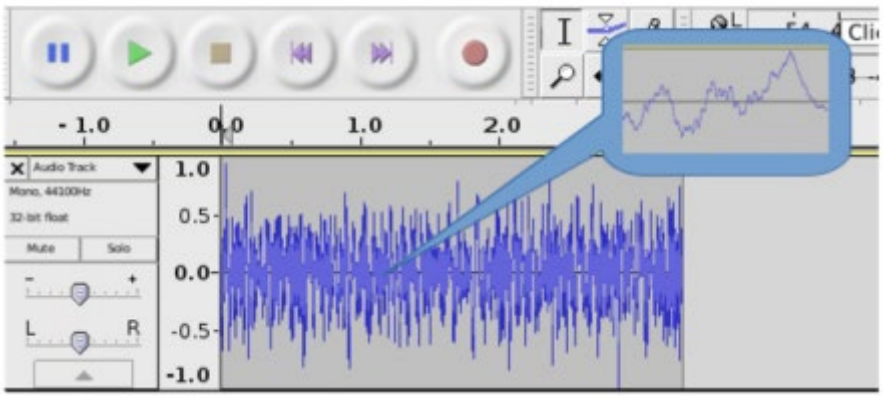
Le son est donc une **vibration mécanique**, nécessitant un **milieu matériel** : s'agissant des sons que nous entendons tous les jours, le milieu matériel est bien évidemment l'air ambiant.

On appelle **fréquence** du son, la vitesse avec laquelle ces molécules vibrent. Plus la vibration des molécules est rapide, plus le son est aigu : on parle de fréquence élevée. Inversement, plus la vibration est lente, plus basse est la fréquence. Une corde de guitare détendue vibre moins vite que sa voisine très tendue, elle va produire un son plus grave avec une oscillation bien plus lente.

Le niveau sonore correspond lui à la hauteur de l'oscillation : on parle **d'amplitude**.

[7] Figure 9 - https://www.futura-sciences.com/sciences/actualites/physique-bientot-communications-fil-travers-eau-72564/

Ce phénomène physique d'oscillation des molécules dans l'air est capté par notre oreille en mettant en vibration nos organes qui vont convertir cette pression reçue en signaux électriques transmis au cerveau. Votre musique préférée est donc **une addition de sons avec des fréquences et amplitudes différentes** qui vont vous fait vibrer au sens propre… comme au figuré !



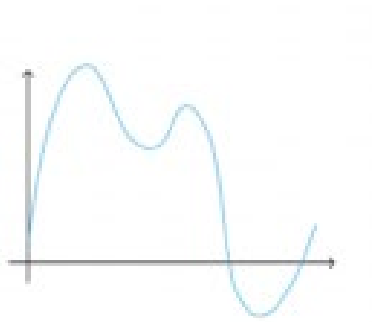
Entre phénomène physique et organe sensoriel, le son physique (on parle également de son **analogique**) va être un ensemble d'oscillations, de vibrations, définies par des fréquences et des amplitudes.

Chaque « son élémentaire » peut ainsi être assimilé à une courbe comme celle décrite dans la vidéo : on parle de **courbe sinusoïdale**, ou encore de sinusoïde. Les sons ou la musique que vous écoutez n'est autre qu'une somme de ces courbes « convenablement » arrangées.

L'oreille humaine ne peut pas entendre tous les sons, donc toutes les fréquences. Seules les fréquences comprises en 20 et 20'000 Hz sont perceptibles pour un humain.

Du son analogique au son numérique

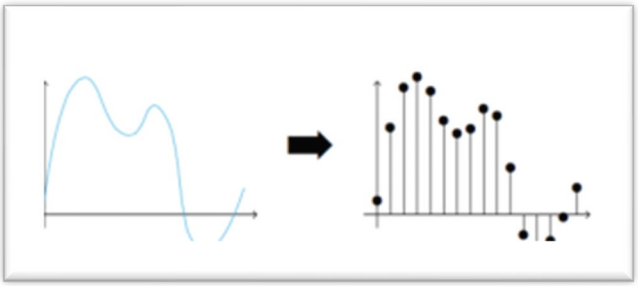
En réalité le son est donc une courbe sinusoïdale continue qui peut ressembler à ceci (en zoomer) :



On passera ici les détails du matériel pour capter le son dans un ordinateur. Ce qui nous intéresse c'est comme le coder avec des 0 et des 1.

On va tout d'abord appliquer ce qu'on appelle **l'échantillonnage**. Ce procédé découpe l'onde en petit morceaux, des échantillons, un certain nombre de fois par seconde et mesure l'amplitude. Cet intervalle de temps, appelé fréquence d'échantillonnage s'exprime en Herz (Hz). Une fréquence de 400Hz signifie qu'on va prendre 400 mesures en 1 seconde.

Ci-dessous un exemple d'échantillonnage.



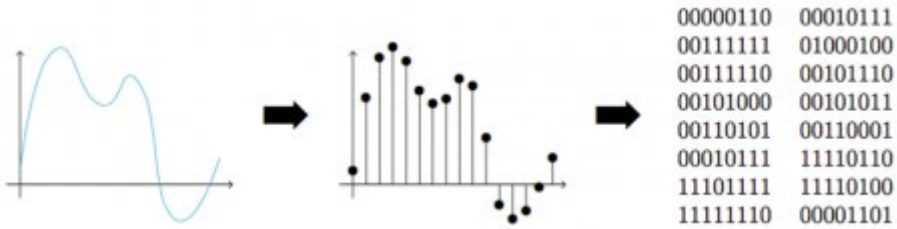
Fréquence de Nyquist

Harry Nyquist a pu démontrer que pour un signal de fréquence F, il fallait une fréquence d'échantillonnage f_e égale au double pour reconstituer sans ambiguïté le signal.

Sachant que l'oreille humaine ne perçoit globalement pas les fréquences supérieures à 20000 Hz, une fréquence d'échantillonnage supérieure à 40 kHz permettra de restituer l'ensemble de l'information physiologiquement perceptible par l'oreille humaine.

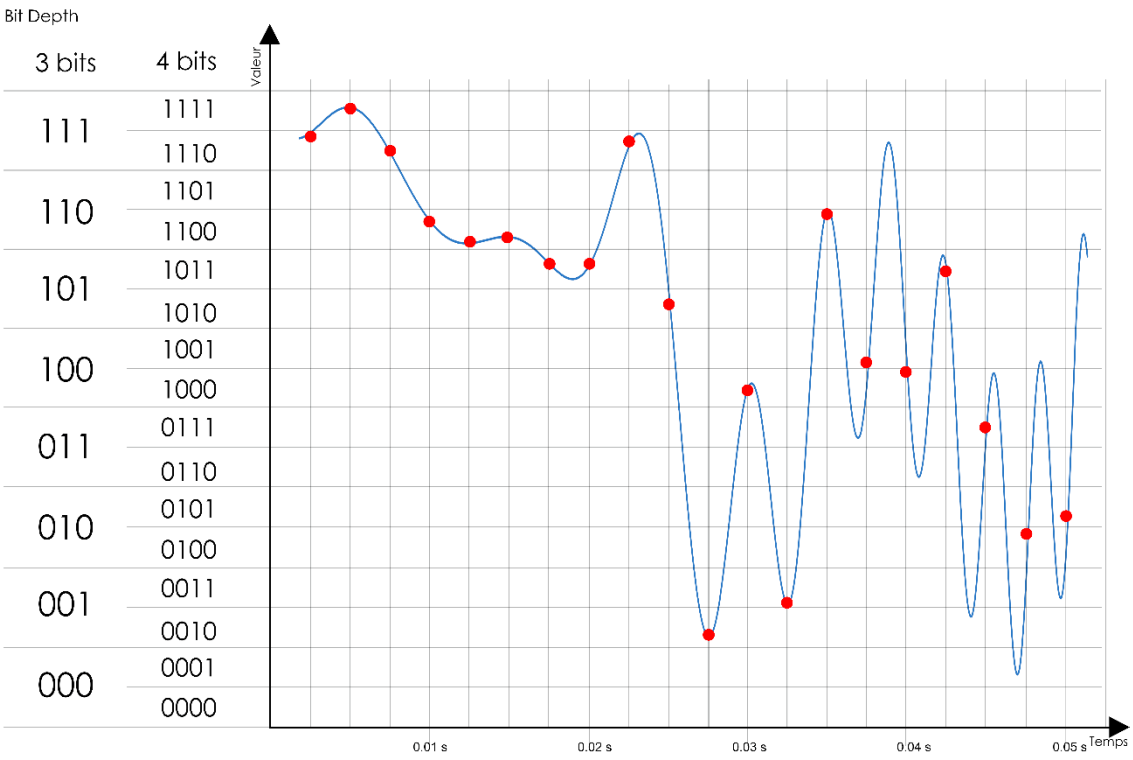
Quantification

Une fois le signal échantillonné, il peut maintenant être codé en binaire, c'est la **quantification**.



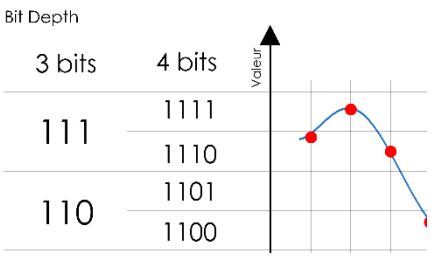
La quantification d'une valeur échantillonnée requiert de déterminer la **précision** de chaque échantillon, ce qui détermine le volume de données générées. Cela découle de la représentation des nombres par les ordinateurs.

La précision de l'encodage est donnée par la **profondeur de l'échantillonnage (bit depth)** exprimée en bits (binary digits). Comme pour l'échantillonnage, plus la profondeur de l'échantillonnage est importante, plus la quantité d'information générée est importante.



Pour faire simple, l'échantillonnage est une découpe horizontale du signal, et la quantification une découpe verticale.

Dans la figure précédente, il est représenté deux types de quantification, une sur 3 bits et une sur 4 bits, soit une découpe en 8 éléments ou 16 éléments.



De ce fait, si on regarde les trois premières mesures, selon la quantification choisie, on les codera les trois de la même manière (111 111 111) ou le 2ème sera différents (1110 1111 1110).

Remarques

Lorsque l'ensemble de la plage des valeurs possibles est utilisé pour l'encodage, la profondeur de l'échantillonnage définit la plage dynamique disponible. Elle est définie entre la valeur encodée la plus petite (0, par exemple) et la valeur encodée la plus élevée (pour une valeur encodée sur n bits, par exemple). Elle correspond également à une idée de précision ou de discrimination des échantillons.

Ici encore, l'oreille humaine ne peut percevoir ni les intensités les plus faibles (inférieures au bruit émis par l'individu lui-même) ni les intensités au-delà du seuil de douleur.

Une précision minimale (environ 8 bits) est ainsi nécessaire pour restituer agréablement un enregistrement respectant les subtilités de l'expression orale (entre voix posée et criée, par exemple).

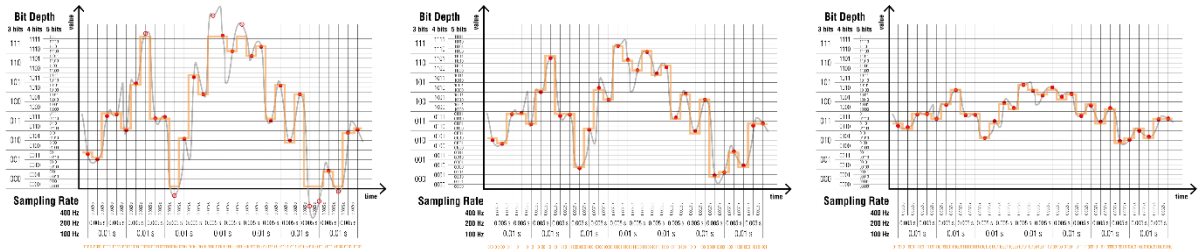
Au-delà de 16 bits, une profondeur d'échantillonnage supérieure engendre une plage dynamique qui n'a pas d'application pertinente pour la restitution des sons pour la plupart des humains, au coût d'une plus grande quantité d'information collectée.

À l'inverse, il est nécessaire de gérer correctement la plage d'amplitude dans laquelle la numérisation du signal se déroule. Cela s'opère en agissant sur le paramètre de **gain** du signal.

La **distorsion** découle d'un signal dont l'amplitude dépasse les capacités d'encodage du système. Dans ces conditions, un ajustement du gain d'entrée

est nécessaire pour rester au plus proche des limites du système, sans les franchir.

La numérisation d'un signal dont l'amplitude serait par trop réduite débouche au contraire sur un encodage qui contient moins d'information, ce qui limite les opérations réalisables numériquement par la suite sans détériorer la qualité du signal.



Exercice 2.21

Complétez le tableau suivant :

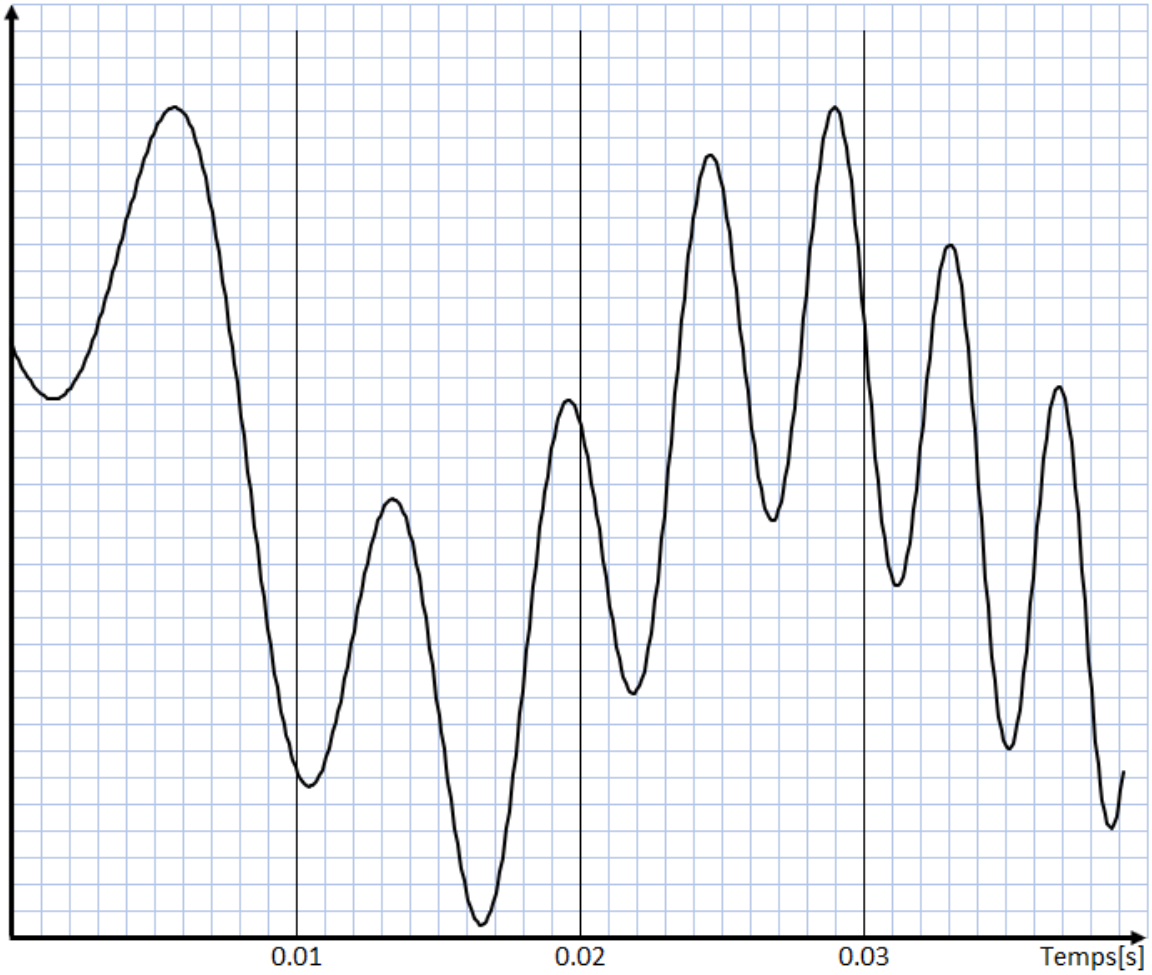
Nb. d'échantillon	Temps	Fréquence
200	1 [s]	
40'000	0.01 [s]	
	20 [s]	20 Hz
	1.5 [min]	40 KHz

Exercice 2.22

Selon les échantillonnages suivants codés sur 5 bits, donnez la taille d'un morceau de musique de 3min et 30 sec ?

Échantillonnage	Taille en Mo
9'000 Hz	
20 KHz	
44,1 KHz	

Exercice 2.23



Dessinez sur le graphe un échantillonnage de 2000 Hz et marquez d'un point tous les échantillons.

Dessinez sur le graphe les quantifications, de profondeurs (bit depth) 2, 3 et 4 bits.

Encodez les points de mesures situés entre 0.02 et 0.03 secondes.

Que peut-on dire sur ces encodages et que pourrait-on améliorer ?

2.7 Les formats

Prenons un exemple de code binaire : 0100010 1001011 0100110 00001111

Comment l'ordinateur peut-il savoir ce qu'il est en train de décoder ? Est-ce une image, du son, un fichier texte ?

Les extensions de fichiers permettent de rajouter ce type d'information. Tout comme pour les images où nous avons vu différents types d'images, tous les fichiers d'un ordinateur ont une extension.

Type de données	**Exemple d'extensions**
Images	Bmp / jpg / gif / png
Son	Mp3 / wav / aiff
Texte	Doc / docx / txt / csv

Bien évidemment, le format du fichier indique comment il doit être lu, donc comment il a été encodé.

Exercice 2.24

Que se passe-t-il si on modifie manuellement l'extension d'un fichier, par exemple, de jpeg à png ?

3 Architecture des ordinateurs

Au chapitre précédent, nous avons vu que le mot informatique traduit la notion du traitement automatique de l'information. Ce traitement d'informations se fait au travers d'un ordinateur depuis les années 1960. Mais qu'est-ce qu'un ordinateur ?

Définition

Un ordinateur est une machine électrique capable de **manipuler automatiquement** des informations et de les **mémoriser**, et que l'on peut **programmer**.

Dans ce chapitre, nous allons donc nous intéresser aux concepts fondamentaux de la structure interne d'un ordinateur en termes de composants, mais aussi de fonctionnement. Nous allons faire le lien entre l'encodage des informations en binaire et leur traitement dans un ordinateur.

Ordinateur au sens large

Au vu de la définition donnée d'un ordinateur, aujourd'hui bon nombre d'appareils intègrent ou sont des ordinateurs. Par exemple, nos SmartPhone sont des ordinateurs, tout comme une montre connectée ou une voiture.

On parle d'ailleurs aujourd'hui, de l'**Internet des Objets**, l'IoT, qui fait donc référence aux objets connectés et qui donc doivent fonctionner de la même manière qu'un **ordinateur personnel** (**PC**).

Le terme PC, qu'on associe aux ordinateurs Windows, veut en réalité dire Personal Computer, soit ordinateur personnel. Ce n'est donc rien d'autre qu'on ordinateur portable ou fixe avec écran, clavier et souris.

Il existe bien d'autres types d'ordinateurs comme les **serveurs** qui hébergent aussi bien des sites internet, des services Cloud ou des données d'entreprise, ou encore ce qu'on appelle des **supercalculateurs** qui sont utilisés et souvent la propriété de grosses universités.

Le point commun entre tous ces différents ordinateurs c'est qu'ils sont remplis de **transistors**, de **puces**, de **processeurs** et de **disques durs**.

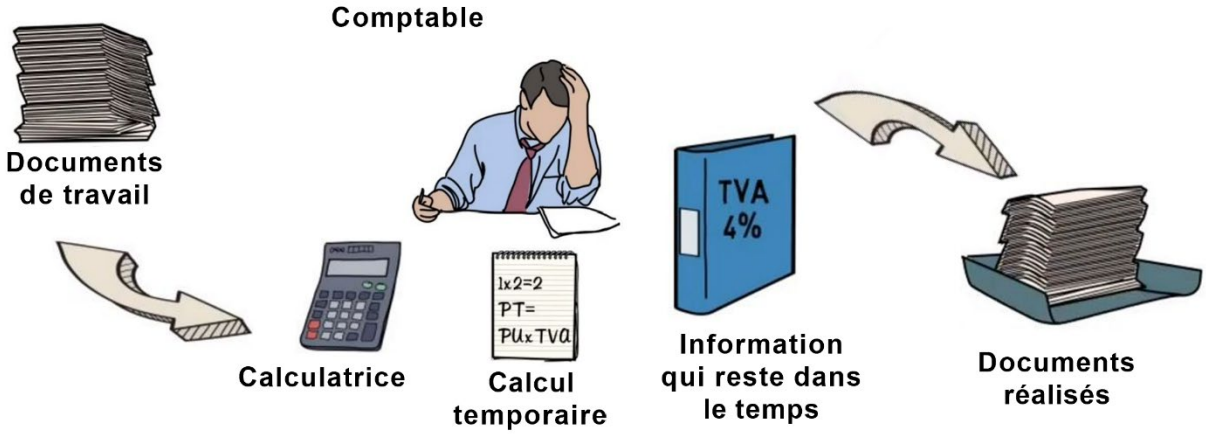
La robotique

La robotique est un domaine à part entière et est donc un sous-domaine de l'informatique. Donc les robots peuvent être considérés comme des ordinateurs mais sont dotés, en plus, de **capteurs**[8], d'**actuateurs**[9] et bien souvent de **systèmes mécaniques** leur permettant de ressembler soit à des humains soit à des animaux.

[8] Composant qui arrive à détecter une donnée physique, par exemple la température ou la distance à un objet
[9] Les actuateurs sont des dispositifs qui permette de créer un mouvement mécanique

3.1 Comment fonctionne un ordinateur ?

Partons d'une analogie simple : un comptable. Cette personne reçoit des **documents** (factures, salaires, frais, etc…) qu'il doit traiter pour créer et vérifier un budget. Pour faire des **calculs intermédiaires**, il s'aide d'une **calculatrice**, tout en **réfléchissant** avec son cerveau. Il va **lire** les documents, **écrire** des calculs, **chercher** dans des sources des informations **externes**, puis **réaliser** et **imprimer** des documents finaux, le tout contrôler par son système nerveux.



Si l'on traduit ce schéma en terme informatique, nous allons facilement retrouver tous les éléments d'un ordinateur.

Données d'entrée (inputs) :	les documents de travail
Unité de calcul (UC) :	la calculatrice
Unité centrale de traitement (CPU) :	le cerveau du comptable
Mémoire vive (RAM) :	la feuille de brouillon pour les calculs
Mémoire morte (ROM) :	les données externes
Données de sortie (outputs) :	les documents imprimés
Carte mère ou contrôleur I/O :	le système nerveux

Visuellement, on peut représenter ces éléments comme le schéma simplifié ci-dessous. Nous détaillerons plus loin ces composants mais cela peut être considéré comme le minima requis pour un ordinateur.

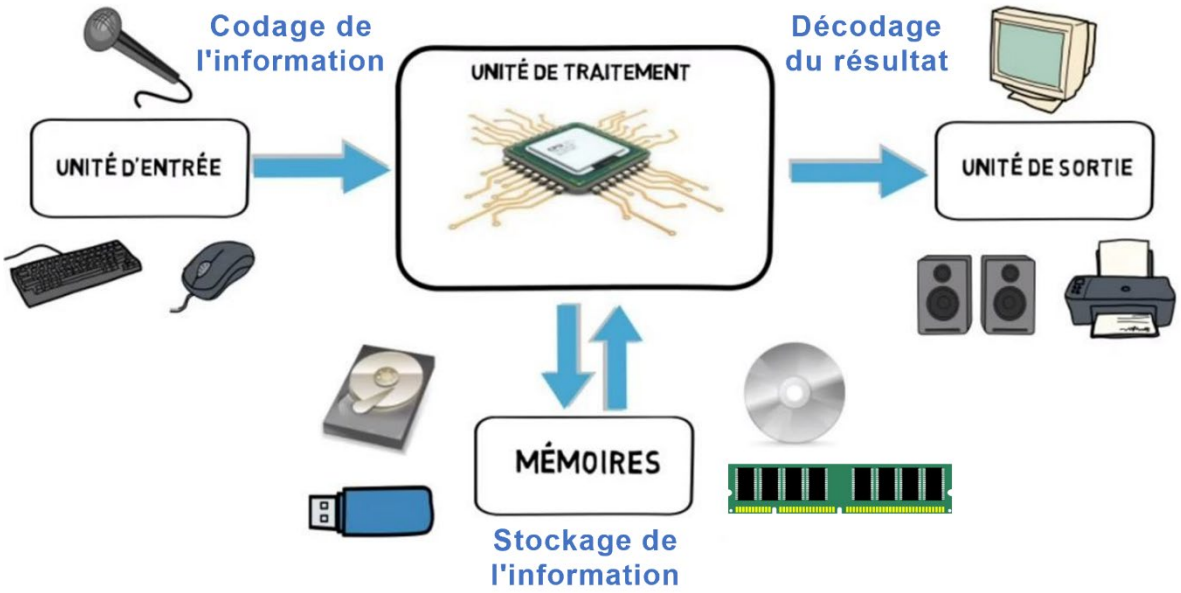


Figure 13 - Représentation imagée d'un ordinateur

3.2 Les périphériques

Le rôle d'un ordinateur est donc de traiter de l'information de manière automatique et rapide. Il y a donc des informations qui doivent lui être fournies et cela peut se faire par ce qu'on appelle des **périphériques**. Ce sont des **composants** électroniques qui sont ajoutés au système principal.

Nous allons avoir des périphériques d'**entrée** ou de **sortie**. Ce que l'on nomme en général input/output (**I/O**).

Ces composants ont pour rôle soit de fournir en entrée des données, comme, par exemple une souris, ou de produire un rendu physique en sortie, comme les haut-parleurs.

Il existe aussi des périphériques qui peuvent avoir les deux fonctions en même temps, comme les supports de données externes (clef USB, carte SD, …) ou même certaines manettes de jeux vidéo qui vont vibrer dans certaines situations.

3.3 Machine de Von Neumann

Depuis les années 60, nous concevons des ordinateurs sur la même architecture, que ce soient nos PC, smartphones ou autres IoT. L'architecture est la manière de concevoir le système principal, soit la manière dont interagissent les composants électroniques.

La communauté informatique a fait de gros efforts également pour créer des standards afin que tous les systèmes puissent communiquer les uns avec les autres.

Le premier ordinateur considéré comme tel a été l'ENIAC (*Electronic Numerical Integrator and Computer*). Développé aux Etats-Unis sous l'influence d'un certain John von Neumann, il sera la première machine électrique programmable. Cet ingénieur donnera son nom à l'architecture que nous utilisons encore aujourd'hui.

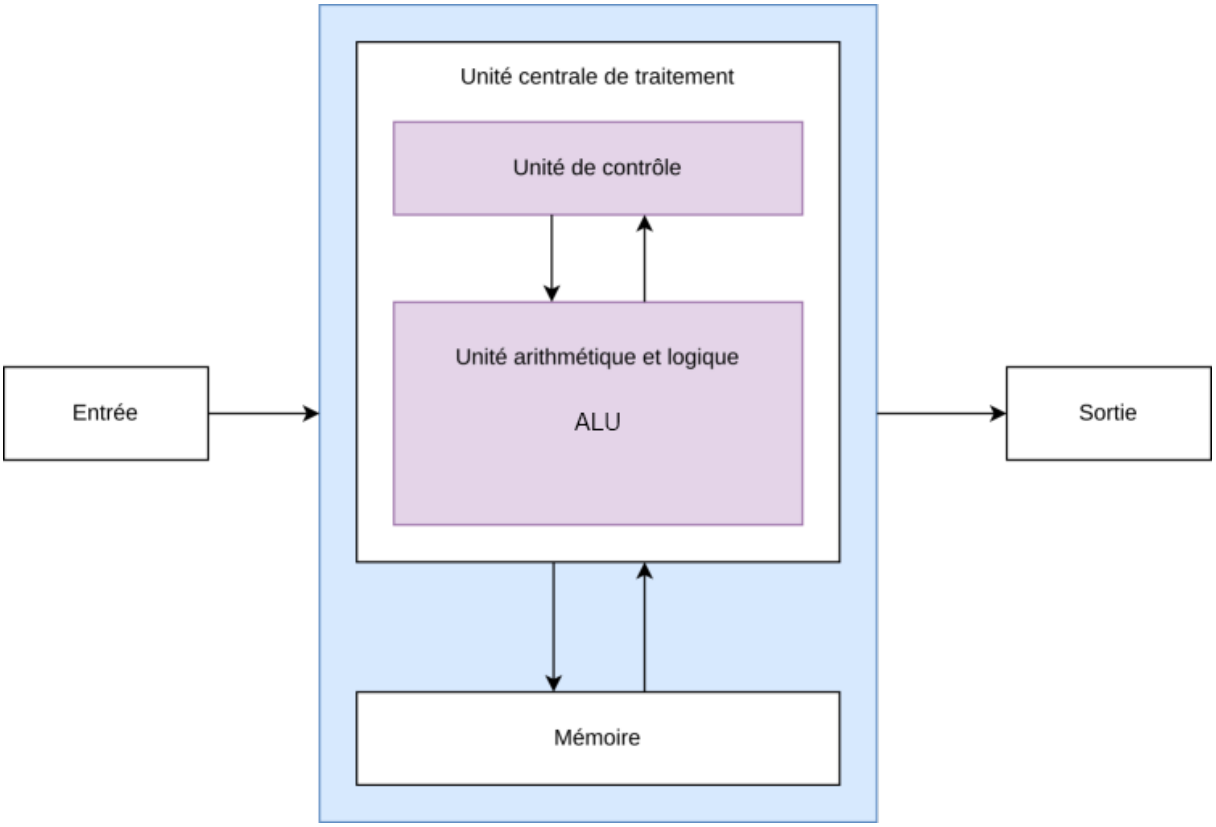


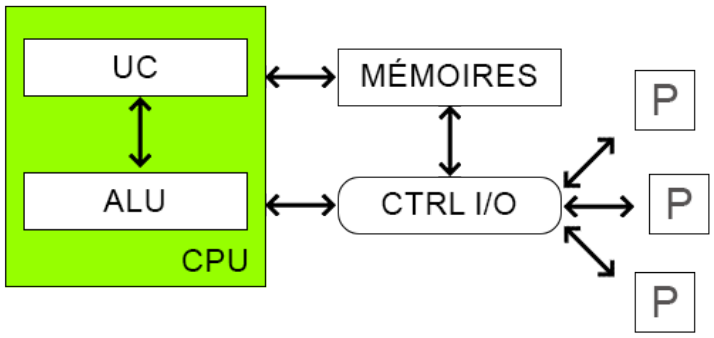
Figure 14 - Architecture de von Neumann

Le principe est relativement simple. Il faut pouvoir prendre des données en entrée (**inputs**), avoir une unité centrale de calculs (**CPU ou processeur**) divisée en deux parties, l'unité de contrôle (**UC**) et l'unité de logique et d'arithmétique (**ALU**). Ce CPU doit pouvoir accéder à des **mémoires** et finalement produire des résultats en sortie (**outputs**), le tout interconnecté et géré par un contrôleur (**carte mère** ou **Ctrl I/O** à lire Controller Inputs/Outputs).

On retrouve ici les mêmes éléments qu'avec l'analogie du comptable.

3.4 Composants d'un ordinateur

Maintenant que le principe fonctionnel d'un ordinateur est posé, il faut définir tous les composants qui vont constituer l'ordinateur. En reprenant la représentation de la Figure 13 et le concept de von Neumann, on peut poser un schéma simple d'un ordinateur :

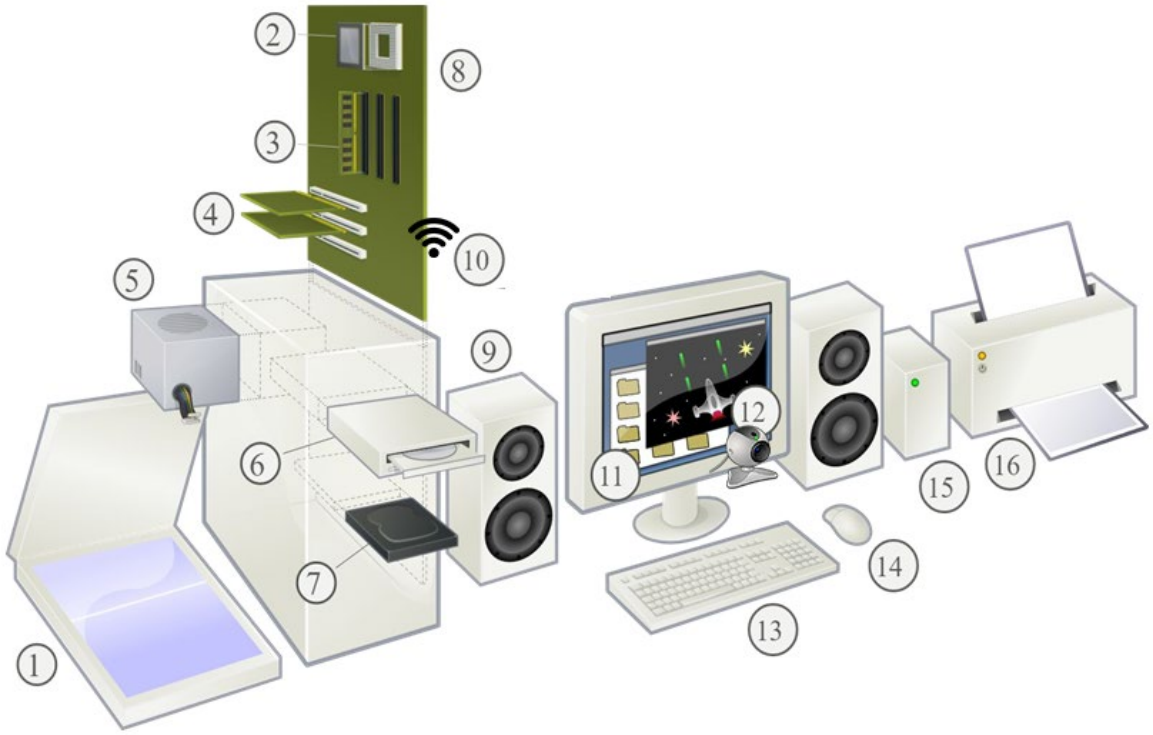


On retrouve bien le CPU et ses éléments, les mémoires, le contrôleur d'entrées/sorties et les périphériques.

Détaillons maintenant, tous ces composants en gardant à l'esprit ce schéma ci-dessus ainsi que l'analogie du comptable.

Exercice 3.1

Nommez les différents composants que vous connaissez sur ce schéma ?



1. 9.
2. 10.
3. 11.
4. 12.
5. 13.
6. 14.
7. 15.
8. 16.

3.4.1 Processeur ou CPU

Le terme anglais CPU (Central Processing Unity) désigne l'unité centrale de calculs qui est en réalité le **cerveau** de l'ordinateur. Il est composé de millions de transistors[10] gravés sur un puce de silicium. Les transistors actuels sont gravés à une taille de 3 à 5 nanomètres, donc plus fin qu'un cheveu !

Figure 15 - Processeur Intel Core i7

Il a principalement trois rôles :

1. Effectuer les **calculs** binaires
2. Exécuter des **instructions** : il donne des ordres aux autres composants ou prend des décisions sur quoi faire à quel moment
3. **Ecrire** ou **lire** dans une mémoire

On parle de **puissance** d'un ordinateur souvent en faisant référence à son processeur. Il fonctionne à une certaine cadence qu'on appelle la **fréquence d'horloge**. Elle est donnée en **Hertz [Hz]**, qui est utilisée en physique, et traduit le **nombre d'opérations par seconde** qu'il peut effectuer. Plus la fréquence est élevée, plus il est capable de calculer rapidement et de faire plusieurs tâches simultanées.

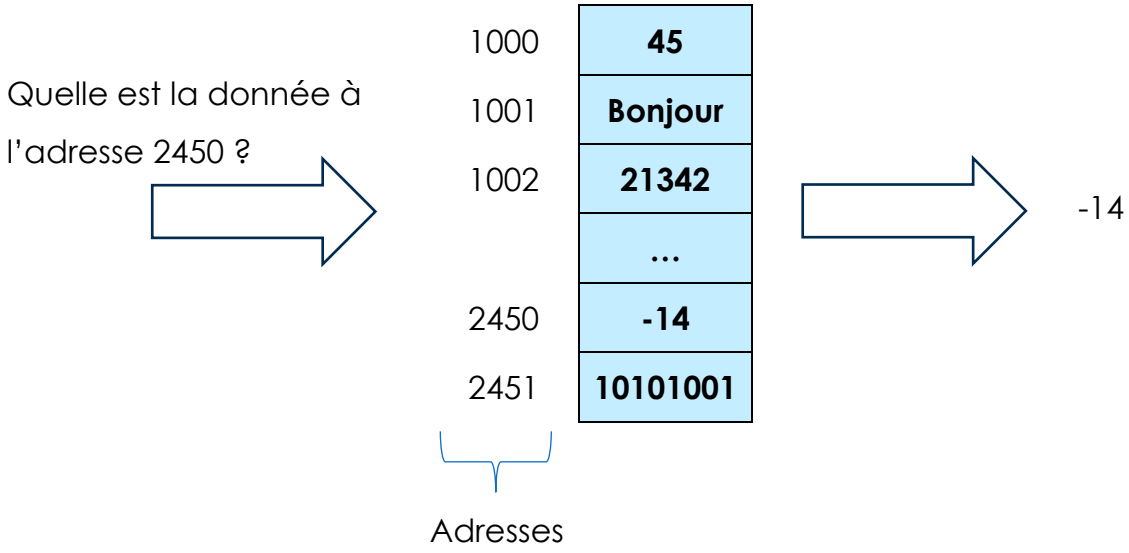
iPhone 15 Pro

Le dernier iPhone 15 Pro est doté de la puce Apple A17 Pro cadencée à 3,78 GHz, ce qui veut dire qu'il peut effectuer 3'780'000 d'opérations par secondes !!!

[10] Un transistor est un élément électronique qui fonctionne comme un interrupteur permettant ainsi de contrôler le flux électrique dans un circuit. Pour plus de détails, référez-vous au cours de physique ou via cet article : https://couleur-science.eu/?d=775902--cest-quoi-un-transistor-comment-ca-marche

3.4.2 Les mémoires

Il existe plusieurs types de mémoires dans un ordinateurs mais toutes peuvent être représentées de la même manière. On peut imaginer un **tableau** dans lequel chaque cellule est numérotée par une **adresse** et peut contenir une **donnée** binaire.



Ci-dessus, un exemple général de mémoire que le CPU, par exemple, souhaite lire. Il demande à la mémoire la donnée se trouvant à l'adresse 2450 et la mémoire répond « -14 ».

On trouve trois types de mémoires :

- La mémoire **vive**
- La mémoire **morte**
- La mémoire **de masse**

Mémoire vive

La mémoire vive, appelée **RAM** (Random Access Memory) est une mémoire **très rapide** d'accès. C'est la mémoire principale du système et le processeur peut aussi bien y **lire** qu'y **écrire**. On pourrait la comparer à la mémoire à court terme d'un humain. L'ordinateur va y stocker des données temporaires mais utiles pour un travail en cours.

Figure 16 - Barrettes de RAM

La taille de cette mémoire est évaluée en giga-octet (Go) aujourd'hui et peut-être choisie lors de l'achat d'un ordinateur personnel, contrairement aux smartphones où cette mémoire est définie au départ.

On la trouve sous forme de barrettes qui peuvent être insérée sur la carte mère. C'est pourquoi si votre ordinateur commence à être plus lent, il est possible de changer cette mémoire et donc d'augmenter sa capacité.

Une de ses caractéristiques est que lorsque la machine s'éteint, tout **son contenu est perdu**.

Mémoire morte

La mémoire morte est appelée ROM (Read Only Memory) et est, comme son acronyme l'indique, accessible **uniquement en lecture**. Le processeur ne peut donc pas y écrire quelque chose. Le stockage y est donc **permanent**. Ces mémoires sont intégrées directement à la construction de l'ordinateur et permettent surtout le démarrage du système ou des programmes spécifiques.

Figure 17 - Une mémoire ROM soudée à la carte mère

Mémoire de masse

Ces mémoires sont les plus connues. Ce sont les **disques durs**, les **clefs USB** ou pour certains les CD/DVD, bien que ces supports soient voués à disparaître.

Ce sont des mémoires moins rapides que la mémoire vive mais son de très grandes capacités. On peut jusqu'à plusieurs dizaines de tera-bytes.

Bien évidemment, une fois l'ordinateur éteint, les données sont maintenues sur le support et son accès peut se faire en lecture ou en écriture.

Aujourd'hui on parle quasiment plus que de disque **SSD** (Solid State Drive) qui sont peu coûteux, rapides et peu volumineux. Mais on trouve encore des disques mécaniques rotatifs qui inscrivent les données de manière magnétique.

Figure 18 - Disque dur rotatif magnétique

Figure 19 - Disques SSD avec ou sans boitier

3.4.3 Carte mère

La carte mère (motherboard)est le composant principal car c'est elle qui r**elie tous les autres entre eux**, leur permettant donc de communiquer.

On va pouvoir y fixer donc le CPU, les barrettes de RAM, le disque SSD et elle aura déjà des ports pour connecter des périphériques externes comme des composants USB.

Les cartes PCI

Les cartes PCI (Preripheral Component Interconnect) sont des cartes d'extension de périphériques. Cela veut dire qu'elles vont se connecter sur la carte mère et vont pouvoir faire communiquer la carte mère et un nouveau périphérique externe.

Voici une liste non-exhaustive de cartes PCI :

- Ports USB
- Carte réseau (WiFi ou ethernet)
- Carte son
- Carte graphique

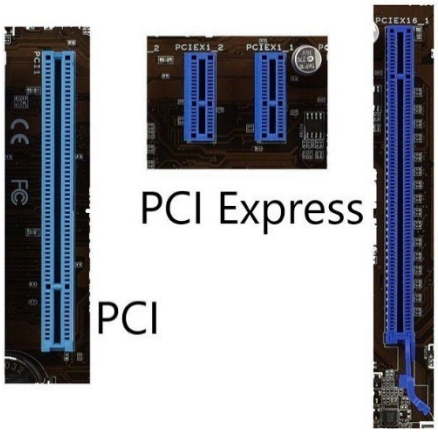


Figure 20 - Ports PCI fixés sur une carte mère

3.4.4 Carte graphique

Ce composant est surtout présent dans les ordinateurs personnels car nous avons besoin de voir les contenus de manière graphique sur un écran. En effet, sur certains serveurs ou supercalculateur, on accède seulement à distance à la machine.

Ce composant dispose de sa propre unité de calcul et évite ainsi au CPU de le faire. C'est ce qu'on appelle la **GPU**, Graphics Processing Unit, et est un paramètre très apprécié des *gamers*.

On trouvera des **GPU intégrée**, directement sur la carte mère et qui utilise la mémoire centrale, ou des **GPU discrète** qui va se fixer à la carte mère et possède sa propre mémoire.

Figure 21 - Carte graphique qui s'intègre à la carte mère

> Aujourd'hui, la compagnie Nvidia est un des leaders du marché de cartes graphiques tant pour le PC que pour des supercalculateurs, notamment ceux dans le domaine de l'IA générative.
>
> En Suisse, à Lugano, nous disposons d'un supercalculateur au Centre national suisse de calcul scientifique qui investit dans un réseau de 10'000 GPU Nvidia, au prix de 30 à 40'000 CHF l'unité…

3.4.5 Alimentation

Jusqu'ici nous avons parlé de beaucoup de composants électroniques et donc qui fonctionnent à l'électricité. Il faut donc bien les alimenter via une alimentation. Son rôle va être de transformer le courant 220 [V] de la prise électrique en courant continu de 3.3, 5 ou 12 [V] dépendamment des besoins de chaque composant.

Figure 22 - Alimentation d'un PC

L'alimentation va avoir une dizaine de connecteurs pour se fixer aux différents composants.

Bien que les smartphones soient conçus sur la même architecture, l'alimentation se fait uniquement par une batterie. C'est aussi une des raisons qui fait la complexité de ce genre d'appareil car tout est miniaturisé et les composants plus fragiles.

3.5 Périphériques

Maintenant que nous avons défini les composants minimaux pour parler d'ordinateur, on peut s'intéresser aux composants accessoires qui peuvent être utiles notamment pour qu'un humain puisse utiliser la machine.

On va distinguer les périphériques d'entrée et de sortie, soit ceux qui apportent des données au CPU et ceux qui utilisent les données du CPU pour produire un rendu physique.

3.5.1 Clavier, souris, écran

Les premiers périphériques que l'on connait sont bien évidemment le clavier, la souris et l'écran, pour un ordinateur personnel. La **souris** et le **clavier** sont des périphériques **d'entrée** étant donné qu'ils vont indiquer par exemple quelle touche a été enfoncée par l'utilisateur, ou encore quel bouton de la souris a été cliqué.

L'écran, lui, est un périphérique de **sortie** car il traduit des données binaires en images pour que l'humain le comprenne.

Périphérique d'entrée et de sortie

Il existe des cas où un périphérique a les deux rôles. Prenons l'exemple de l'écran d'un smartphone qui est tactile. Non seulement il produit une image mais il réagit lorsqu'on le touche pour indiquer où l'on a appuyé.

Certaines mannettes de jeu vidéo le sont également. Elles vibrent par exemple durant certaines actions du jeu.

Exercice 3.2

Citez 8 périphériques et annoncez s'ils sont en entrée, en sortie ou les deux

Périphérique	**Entrée**	**Sortie**
Souris (simple)	X	

3.6 Les systèmes logiques

Jusqu'ici, on a appris qu'un ordinateur fonctionne qu'avec des 0 et 1, ce qu'on appelle des bits et qui traduit le fait que ses composants fonctionnent avec de l'électricité. Un transistor, petit composé électronique, représente un bit d'information en laissant ou non passé du courant. Mais chaque composant complexe vu plus haut, comme le CPU, est conçu avec des milliers de transistors qui forment des **circuits électroniques**.

Dans ce chapitre, nous allons voir comment sont conçus ces circuits électroniques et comment un ordinateur arrive à traiter des opérations telle que l'addition binaire et manipuler les bits d'information.

3.6.1 Exemple préliminaire

L'exemple le plus simple c'est de partir de la plus petite unité d'information, soit 1 bit, représentant un nombre entier. Bien évidemment, ce nombre ne peut être que 0 ou 1.

Soit un nombre A et un nombre B que l'on additionne. Le résultat est donné dans le nombre S.

A	B	S	
0	0	0	$0 + 0 = 0$
1	0	1	$1 + 0 = 1$
0	1	1	$0 + 1 = 1$
1	1	10	$1 + 1 = 10_{(2)} = 2_{(10)}$

On constate avec le cas le plus simple, que pour additionner deux bits il faut que le résultat soit encodé sur 2 bits. Autrement, nous aurions un dépassement de capacité (voir chapitre 2.2).

La question est de savoir comment est-ce possible avec un circuit électrique de représenter cette table d'addition.

3.6.2 Portes logiques

L'idée n'est pas de comprendre la physique ou l'électronique de ces systèmes mais bien le concept informatique derrière.

Les **portes logiques** sont composants électroniques constitués de transistors, qui effectuent une **opération logique** sur une ou plusieurs **entrées** de courant et qui donne un résultat sur une ou plusieurs **sorties** de courant.

Pourquoi des portes *logiques* ?

Nous avons vu que le 0 et le 1 binaire exprime le fait d'avoir du courant ou pas dans le transistor. Mais cela est le concept physique. Or pour l'informatique, le **0** est perçu comme la valeur **FAUX** et le **1** comme la valeur **VRAI**. Ces valeurs sont dites **booléennes** car proviennent de l'algèbre de Boole, qui est une approche algébrique de la logique.

Plus simplement, prenons un exemple avec des phrases :

- Si je dis « il fait beau », la réponse peut être VRAI ou FAUX
- Si je dis « il pleut », la réponse peut être VRAI ou FAUX
- Si je dis « il pleut ET il fait beau », la réponse ne peut qu'être FAUX

C'est de la logique car s'il pleut c'est qu'il ne fait pas beau. Le ET implique que les deux propositions doivent être VRAI pour que la réponse soit VRAI or ce n'est pas possible.

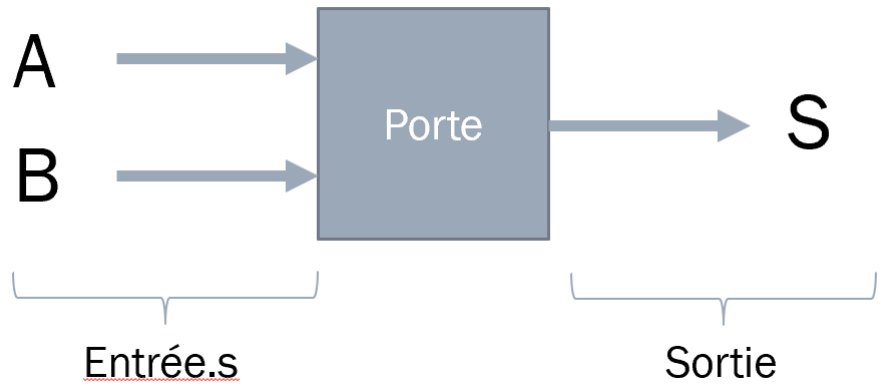
Dans ce cas on peut écrire :

VRAI et FAUX = FAUX

Donc en logique, nous n'aurons pas d'opérateur arithmétique mais des **opérateurs logiques** comme ET, OU, NON.

Et c'est uniquement avec ces opérateurs logiques que nos ordinateurs fonctionnent que soit pour taper un document Word, streamer en ligne ou jouer à un jeu vidéo !

On peut schématiser toutes les portes logiques de la manière suivante :

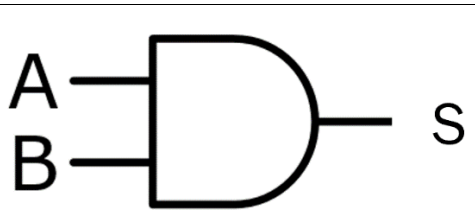


A noter qu'il peut y avoir de 1 à *n* entrées et de 1 à *n* sorties. Chaque entrée et chaque sortie sont en réalité un **fil électrique** et représente donc **1 bit**, soit un 0 ou un 1.

On leur associe une **table de vérité** qui donne pour toutes les combinaisons d'entrées, la sortie correspondante. Donc le nombre de ligne dans une table de vérité vaut 2^n où *n* est le nombre d'entrée.

On leur donne une **représentation graphique** et un nom en français bien que dans la communauté informatique c'est le **nom anglais** qui prédomine.

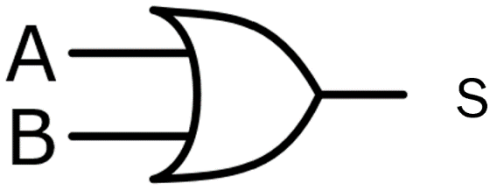
Porte ET

Définition : **La sortie S vaut 1 si et seulement si toutes les entrées valent 1**. Nom anglais : **AND**	

Cette porte peut prendre autant d'entrées que besoin. Ici l'exemple simple est avec deux entrées et sa table de vérité sera :

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Porte OU

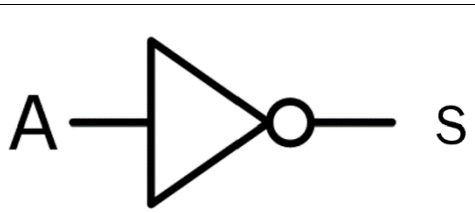
Définition : **La sortie S vaut 1 une des entrées vaut 1**. Nom anglais : **OR**	

Cette porte peut prendre autant d'entrées que besoin. Ici l'exemple simple est avec deux entrées et sa table de vérité sera :

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Contrairement au langage courant le OU logique accepte que les deux propositions soient VRAI. « *Tu veux une glace ou un bonbon ?* »

Porte NON

Définition : **La sortie S vaut l'inverse de son entrée**. Nom anglais : **NOT**	

Remarquez le petit rond avant la sortie. Il indique l'inversion de la valeur. Il se peut que vous rencontriez ce sigle à la sortie d'autres portes.

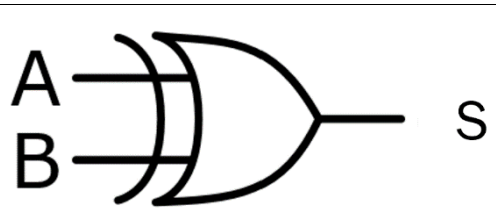
Cette porte ne prend qu'une seule entrée, car elle ne fait que d'inverser la valeur. Sa table de vérité est :

A	S
0	1
1	0

Avec ces trois portes logiques très simples, il est possible de **construire l'entier d'un ordinateur actuel** ! L'électricité se déplace extrêmement rapidement dans de bons conducteurs, c'est pourquoi on arrive à avoir des vitesses de calculs aussi élevés car même pour additionner de très grands nombres entre eux, une fois converti en binaire, il suffit de les passer dans des portes logiques.

Malgré cela, il existe bien d'autres portes logiques et des combinaisons d'entre-elles que l'on appelle des **circuits logiques**.

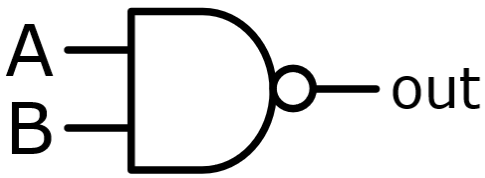
Porte OU-X (ou-exclusif)

Définition : **La sortie S vaut 1 quand A ou B vaut 1 exclusivement**. Nom anglais : **XOR**	

Cette porte peut prendre autant d'entrées que besoin. Ici l'exemple simple est avec deux entrées et sa table de vérité sera :

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Porte NON-ET

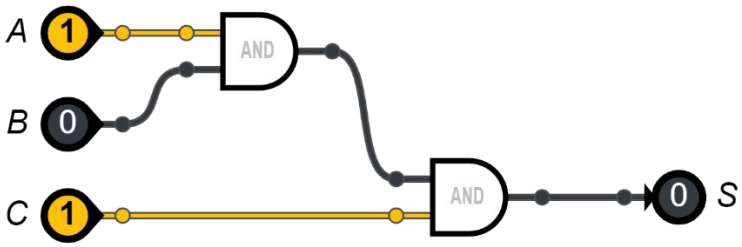
Définition : **La sortie S vaut 1 quand A ou B vaut 1 exclusivement**. Nom anglais : **NAND**	

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

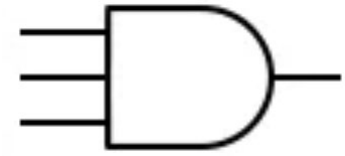
Circuits logiques

Un circuit logique est un ensemble de portes logiques.

Nous avons vu que les portes basiques sont AND et OR à deux entrées et qu'en ajoutant la NOT on pouvait créer l'ensemble des composants nécessaires. Il se trouve que pour, par exemple, créer une porte AND à 3 entrées, c'est simplement un circuit logique constitué de 2 portes AND.



Ce circuit est donc l'équivalent d'une porte AND à 3 entrées représentée comme suit :



Deux circuits sont dits **équivalents** s'ils ont la même table de vérité.

Portes universelles

Il existe deux portes dites universelles car en utilisant uniquement l'une d'entre elles, on peut alors reproduire tous les circuits logiques et toutes les portes. Ce sont les portes NAND et NOR.

Le principal avantage est économique vu qu'il suffit de produire qu'une seule porte électronique pour faire l'ensemble des composants informatique.

3.6.3 Additionneur binaire

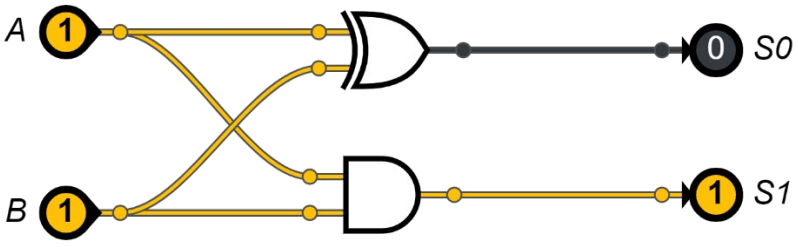
Revenons à notre exemple préliminaire afin d'additionner 2 bits. Maintenant que nous savons qu'on n'effectue que des opérations logiques comment cela est-il possible ?

Ecrivons la table de vérité de notre additionneur, sachant que 2 bits sont nécessaires en sortie :

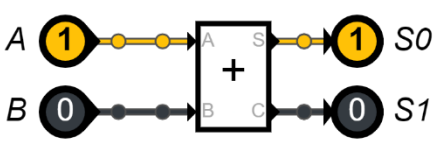
A	B	S_1	S_0
0	0	**0**	**0**
1	0	**0**	**1**
0	1	**0**	**1**
1	1	**1**	**0**

Pour rappel, en binaire 1 + 1 = 10. Le bit de poids faible est le 0 (tout à droite) et le bit de poids fort le 1 (à gauche). Ils sont respectivement notés S_0 et S_1.

On a donc un circuit logique à 2 entrées et 2 sorties. Il suffit alors d'observer la table de vérité de S_1 puis celle de S_0 pour compléter le circuit. En effet, on remarque que S_1 correspond à une porte AND, alors que S_0 correspond à une porte XOR. On obtient alors le circuit :



On appelle ce composant un demi-additionneur, car il ne peut pas additionner plus que deux bits uniques, et a une représentation spécifique :



4 Annexes

Annexe 1 : table UTF

Unicode code point	Decimal	Char	Unicode code point	Decimal	Char	Unicode code point	Decimal	Char	Unicode code point	Decimal	Char
U+0014	20	control	U+0041	65	A	U+006E	110	n	U+00E0	224	à
U+0015	21	control	U+0042	66	B	U+006F	111	o	U+00E1	225	á
U+0016	22	control	U+0043	67	C	U+0070	112	p	U+00E2	226	â
U+0017	23	control	U+0044	68	D	U+0071	113	q	U+00E3	227	ã
U+0018	24	control	U+0045	69	E	U+0072	114	r	U+00E4	228	ä
U+0019	25	control	U+0046	70	F	U+0073	115	s	U+00E5	229	å
U+001A	26	control	U+0047	71	G	U+0074	116	t	U+00E6	230	æ
U+001B	27	control	U+0048	72	H	U+0075	117	u	U+00E7	231	ç
U+001C	28	control	U+0049	73	I	U+0076	118	v	U+00E8	232	è
U+001D	29	control	U+004A	74	J	U+0077	119	w	U+00E9	233	é
U+001E	30	control	U+004B	75	K	U+0078	120	x	U+00EA	234	ê
U+001F	31	control	U+004C	76	L	U+0079	121	y	U+00EB	235	ë
U+0020	32	SPACE	U+004D	77	M	U+007A	122	z	U+00EC	236	ì
U+0021	33	!	U+004E	78	N	U+00C0	192	À	U+00ED	237	í
U+0022	34	"	U+004F	79	O	U+00C1	193	Á	U+00EE	238	î
U+0023	35	#	U+0050	80	P	U+00C2	194	Â	U+00EF	239	ï
U+0024	36	$	U+0051	81	Q	U+00C3	195	Ã	U+00F0	240	ð
U+0025	37	%	U+0052	82	R	U+00C4	196	Ä	U+00F1	241	ñ
U+0026	38	&	U+0053	83	S	U+00C5	197	Å	U+00F2	242	ò
U+0027	39	'	U+0054	84	T	U+00C6	198	Æ	U+00F3	243	ó
U+0028	40	(	U+0055	85	U	U+00C7	199	Ç	U+00F4	244	ô
U+0029	41	)	U+0056	86	V	U+00C8	200	È	U+00F5	245	õ
U+002A	42	*	U+0057	87	W	U+00C9	201	É	U+00F6	246	ö
U+002B	43	+	U+0058	88	X	U+00CA	202	Ê	U+00F7	247	÷
U+002C	44	,	U+0059	89	Y	U+00CB	203	Ë	U+00F8	248	ø
U+002D	45	-	U+005A	90	Z	U+00CC	204	Ì	U+00F9	249	ù
U+002E	46	.	U+005B	91	[	U+00CD	205	Í	U+00FA	250	ú
U+002F	47	/	U+005C	92	\	U+00CE	206	Î	U+00FB	251	û
U+0030	48	0	U+005D	93	]	U+00CF	207	Ï	U+00FC	252	ü
U+0031	49	1	U+005E	94	^	U+00D0	208	Ð	U+00FD	253	ý
U+0032	50	2	U+005F	95	_	U+00D1	209	Ñ	U+00FE	254	þ
U+0033	51	3	U+0060	96	`	U+00D2	210	Ò	U+00FF	255	ÿ
U+0034	52	4	U+0061	97	a	U+00D3	211	Ó	U+0020AC	8364	€
U+0035	53	5	U+0062	98	b	U+00D4	212	Ô	U+001F602	128514	😂
U+0036	54	6	U+0063	99	c	U+00D5	213	Õ			
U+0037	55	7	U+0064	100	d	U+00D6	214	Ö			
U+0038	56	8	U+0065	101	e	U+00D7	215	×			
U+0039	57	9	U+0066	102	f	U+00D8	216	Ø			
U+003A	58	:	U+0067	103	g	U+00D9	217	Ù			
U+003B	59	;	U+0068	104	h	U+00DA	218	Ú			
U+003C	60	<	U+0069	105	i	U+00DB	219	Û			
U+003D	61	=	U+006A	106	j	U+00DC	220	Ü			
U+003E	62	>	U+006B	107	k	U+00DD	221	Ý			
U+003F	63	?	U+006C	108	l	U+00DE	222	Þ			
U+0040	64	@	U+006D	109	m	U+00DF	223	ß			