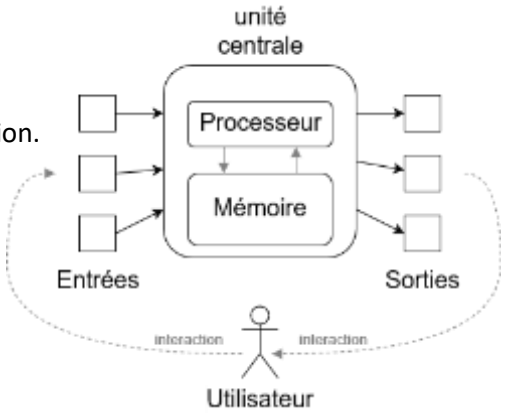
Education au numérique

Introduction : comment fonctionne un ordinateur ?

*Un **ordinateur** est un système de traitement de l'information programmable tel que défini par Turing et qui fonctionne par la lecture séquentielle d'un ensemble d'instructions, organisées en programmes, qui lui font exécuter des opérations logiques et arithmétiques*.

Typiquement, un ordinateur est composé de 4 parties :

- Le **processeur** qui va permettre le traitement de l'information.
- La **mémoire** qui permet de conserver l'information.
- Les **entrées** qui permettent à l'ordinateur de recevoir de l'information (clavier, souris, USB…).
- Les **sorties** qui permettent de ressortir de l'information (écran, imprimante, USB…).



Le processeur est composé d'un grand nombre de circuits électroniques et c'est le passage du courant électrique dans ces circuits qui fait fonctionner le processeur (et donc l'ordinateur). À travers les circuits, seules 2 informations peuvent passer : soit il y a du courant (le circuit est fermé), soit il n'y en a pas (le circuit est ouvert). Pour nous « simplifier » la vie, nous utilisons les valeurs 0 et 1 pour représenter ces informations : 0 quand le courant ne passe pas et 1 quand le courant passe.

La mémoire fonctionne sur le même mode ; il s'agit en fait de plein de circuits ouverts ou fermés, donc des 0 et des 1. Toute l'information présente dans la mémoire et toute l'information traitée par l'ordinateur n'est en fait qu'une suite de 1 et de 0 !

Représentation de l'information

Toutes les informations présentes dans l'ordinateur ne sont que des 1 et des 0. Les informations en entrée sont en fait **numérisées** (=codées) pour être manipulées par l'ordinateur. Ces informations seront ensuite décodées pour être ressorties de manière « lisible » pour l'utilisateur.

Pour retrouver une information stockée dans la mémoire, il faut connaitre son **adresse** qui indique l'endroit où se trouve cette information.

Les 0 et 1 sont appelés « **bit** » (noté b). Un bit est la plus petite unité d'information manipulable par une machine. Un **octet** (byte en anglais et noté B) est une unité d'information composée de 8 bits.

Pour nous faciliter la vie, des normes sont utilisées. Une norme permet à tous les appareils qui l'utilisent de communiquer. Sans norme, chacun coderait comme il veut ses informations en 1 et 0, ce qui rendrait l'échange d'informations très compliqué. Les normes permettent en quelque sorte de tous parler la même langue.

Les nombres

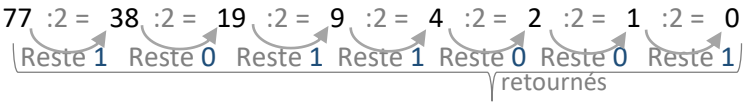
Dans la vie de tous les jours, nous comptons en décimal, également appelé « base 10 » car la base des exposants est 10 et les chiffres utilisés vont de 0 à 9, soit 10 chiffres (voir exemple)

$$275 = 5 \text{ unités} + 7 \text{ dizaines} + 2 \text{ centaines} = 5x1 + 7x10 + 2x100 = 5x10^0 + 7x10^1 + 2x10^2$$

Pour représenter les nombres uniquement avec des 0 et des 1, nous allons compter en « base 2 ». Ainsi, l'exposant sera 2 et les chiffres utilisés seront entre 0 et 1, soit 2 chiffres (ça tombe bien). Pour décoder un nombre en binaire, c'est en fait assez simple, c'est la même chose que le décimal sauf que la base de l'exposant est 2. Par exemple, 11010 en base 2 (noté 11010_2) vaut 26 en base 10 (26_{10}).

$$11010 = 0x2^0 + 1x2^1 + 0x2^2 + 1x2^3 + 1x2^4 = 0 + 2 + 0 + 8 + 16 = 26$$

Pour coder un nombre en binaire (de décimal vers binaire), il faut simplement faire une suite de divisions par **2** (car binaire = base **2**). Les restes de ces divisions mis côte à côte puis retournés seront le nombre en binaire.



Nombre en binaire = 1001101_2

Ecrit sur 1 octet = 01001101_2

De la même manière, vous pouvez compter en n'importe quelle base.

Pour compter dans une base plus grande que 10, il nous faudra plus de 10 chiffres. Par exemple, en base 16, nous aurons besoin de 16 chiffres. Pour cela, nous utilisons les lettres de l'alphabet. A vaudra 10, B 11, C 12 et ainsi de suite.

$$F4A_{16} = Ax16^0 + 4x16^1 + Fx16^2 = 10x1 + 4x16 + 15x256 = 10+224+3840 = 4074_{10}$$

Exercices

Traduire en base 10		Traduire en base 2
101_2 =	1101010_2 =	10_{10} =
101_8 =	10_{16} =	89_{10} =
101_{16} =	10_{43} =	
101_{32} =	10_{2458} =	

Pour les nombres à virgule

Le premier chiffre à gauche du point est multiplié par la base exposant 0. Plus on va vers la gauche, plus l'exposant augmente (à chaque fois de 1). Jusqu'ici c'est comme avant. Pour les chiffres après la virgule, plus on va vers la droite plus l'exposant diminue (à chaque fois de 1). Par exemple :

En base 10 : $16.625 = 1x10^1 + 6x10^0 + 6x10^{-1} + 2x10^{-2} + 5x10^{-3}$
En base 2 : $10000.101 = 1x2^4 + 0x2^3 + 0x2^2 + 0x2^1 + 0x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3}$
$= 16 + 0 + 0 + 0 + 0 + 0.5 + 0 + 0.125$
$= 16.625$

Il existe certaines limites…
Par exemple, 0.347_{10} a seulement 3 chiffre après la virgule. Son équivalent binaire est $0.01011000110\ldots_2$ et a un nombre infini de chiffres derrière la virgule… Ce qui peut poser des problèmes d'arrondi.

La norme IEEE754

Cette norme définit la façon de coder un nombre réel. Elle propose de coder le nombre sur 32 bits et le découpe en trois parties :

- Le signe (1 bit) : 0 si + et 1 si -
- L'exposant (8 bits)
- La mantisse (23 bits)

Les étapes à suivre pour coder un nombre réel sont les suivantes : par exemple, -6,625

- Le signe sera 1 car négatif
- Coder la valeur absolue en binaire : $6,625_{10} = 110,101_2$
- Mettre ce nombre sous la forme 1,mantisse : 110,101 = 1,10101 x 2^2 (l'exposant correspond au nombre de fois que la virgule doit être déplacée à droite pour obtenir l'original)
- Etendre la mantisse à 23 bits en rajoutant des 0 derrière (comme en décimal, rajouter des 0 derrière la virgule ne change pas le nombre) : 101 0100 0000 0000 0000 0000
- Ajouter 127 à l'exposant : 2+127 = 129
- Traduire ce nombre en binaire : $129_{10} = 1000\ 0001_2$
- Mettre tout bout à bout (signe, exposant, mantisse) : 1100 0000 1101 0100 0000 0000 0000 0000

Les étapes à suivre pour décoder un nombre réel sont les suivantes :

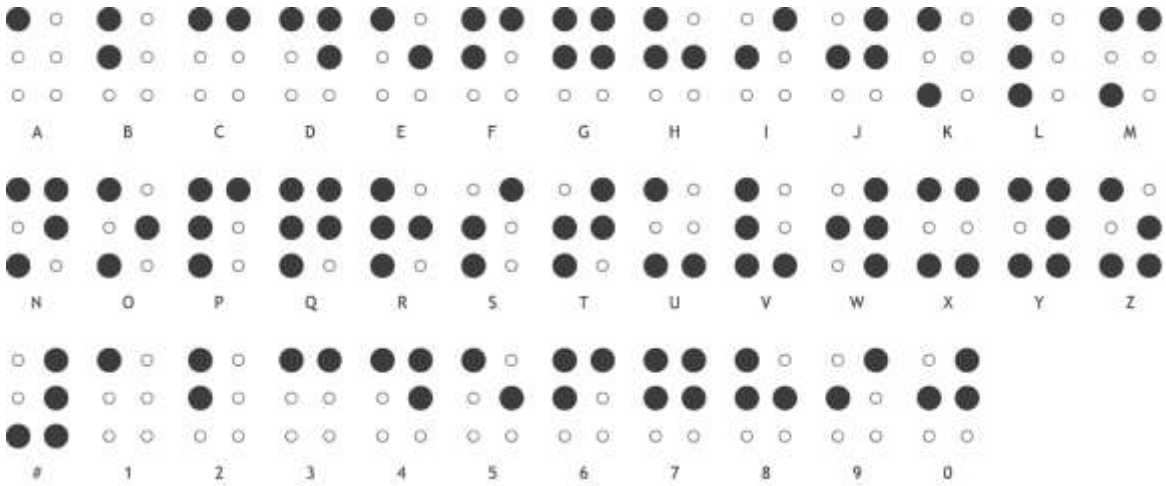
- Prendre le nombre n comme étant 1,mantisse : n = 1,101 0100 0000 0000 0000 0000
- Calculer l'exposant et lui retirer 127 : 129-127 = 2
- Utiliser ce nombre pour décaler la virgule d'autant vers la droite dans n : n = 110,10100…
- Traduire ce nombre en décimal : $110,10100\ldots_2 = 6,625_{10}$
- Rajouter le signe, - si 1 et + si 0 : -6,625

Les textes

Des codes connus

Il existe des codes dont vous avez déjà entendu parler et dont beaucoup de personnes se servent tous les jours : le code braille et le code morse par exemple.

Le braille



Le morse

A •–	J •–––	S •••	1 •———–
B –•••	K –•–	T –	2 ••–––
C –•–•	L •–••	U ••–	3 •••––
D –••	M ––	V •••–	4 ••••–
E •	N –•	W •––	5 •••••
F ••–•	O –––	X –••–	6 –••••
G ––•	P •––•	Y –•––	7 ––•••
H ••••	Q ––•–	Z ––••	8 –––••
I ••	R •–•	0 –––––	9 ––––•

Le code ASCII

Le code ASCII est une des normes les plus utilisées pour représenter en binaire les caractères de l'alphabet latin, les symboles, les signes et les chiffres. A l'origine, il se codait sur 7 bits, ce qui permettait d'avoir jusqu'à 128 caractères (0000000_2 à 1111111_2, 0_{10} à 127_{10}). Ce code a été initialement pensé pour la langue anglaise, il ne contenait donc pas de caractères accentuées. C'est pour cela qu'il a été étendu à 8 bits (1 octet). Il peut maintenant coder beaucoup plus de caractères.

000	(nul)	016	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	☺ (soh)	017	◄ (dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	☻ (stx)	018	↕ (dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥ (etx)	019	‼ (dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦ (eot)	020	¶ (dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005	♣ (enq)	021	§ (nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠ (ack)	022	▬ (syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	• (bel)	023	↨ (etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	◘ (bs)	024	↑ (can)	040	(	056	8	072	H	088	X	104	h	120	x
009	○ (tab)	025	↓ (em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	◙ (lf)	026	→ (eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♂ (vt)	027	← (esc)	043	+	059	;	075	K	091	[	107	k	123	{
012	♀ (np)	028	∟ (fs)	044	,	060	<	076	L	092	\	108	l	124	¦
013	♪ (cr)	029	↔ (gs)	045	-	061	=	077	M	093	]	109	m	125	}
014	♫ (so)	030	▲ (rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	☼ (si)	031	▼ (us)	047	/	063	?	079	O	095	_	111	o	127	⌂
128	Ç	144	É	160	á	176	░	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	ø	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	∈	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Il existe également **l'Unicode** qui est plus complexe mais aussi plus complet et donc plus souvent utilisé.

Exercice : convertir ce texte

66 111 110 106 111 117 114 44 10 86 111 105 99 105 32 117 110 32 116 101 120 116 101 32 115 105

109 112 108 101 46 46 46 32 67 39 101 115 116 32 133 32 118 111 117 115 32 100 101 32 108 101

32 100 130 99 111 100 101 114 32 33 32 11

……

……

……

Enigma

Enigma est une machine électromécanique portable servant au codage et au décodage de l'information. Elle fut utilisée principalement par les Allemands et leurs alliés pendant la Seconde Guerre mondiale. La machine étant réputée inviolable selon ses utilisateurs.

Enigma chiffre les informations en faisant passer un courant électrique à travers une série de composants, notamment des roues appelées rotors. Les rotors sont constitués d'une sorte de labyrinthe de fils électriques. Lorsqu'une lettre du clavier est pressée, le courant est transmis et il traverse le réseau complexe

de fils puis allume une lampe qui indique la lettre chiffrée. Dès qu'un rotor tourne d'un cran, le réseau de fil change complètement et éclaire une autre lettre.

Pour pouvoir décoder un message codé, il fallait connaitre la configuration exacte des rotors de la machine.

(Voir film « Imitation Game » 2014 par Morten Tyldum)

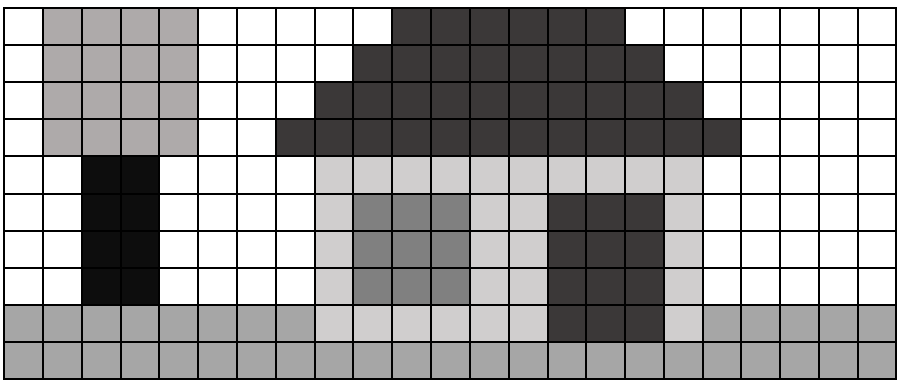
Les images

Les logimages

Le logimage est un jeu de réflexion solitaire, qui consiste à découvrir un dessin sur une grille à partir de chiffres laissés sur le bord de la grille. C'est déjà en quelque sorte une manière de coder des images.

	8	1 1	1 4 1 1	1 4 1 1	1 4 3	1 4 1 1	1 4 1 1	1 4 3	1 4 1 1	1 4 1 1	1 1	8		10	1 1	1 1 1 2 1	1 1 1 2 1	1 1	10
12 6																			
1 1 1 1																			
1 8 1 1 2 1																			
1 8 1 1 1																			
1 8 1 1 2 1																			
1 8 1 1 1																			
1 1 1 2 1																			
1 2 1 2 1																			
1 1 1 1																			
8 6																			

Un peu de la même manière que les logimages, les images sont stockées par l'ordinateur sous forme de grille[1]. On appelle ces images « matricielles » ou « bitmap » (en anglais, « carte de bit »). Chaque case de cette grille est un pixel et chaque pixel a une seule couleur. Il n'y a plus qu'à associer à chaque couleur un nombre qui sera codé en binaire.



[1] Il existe une autre manière de stocker des images (sous forme vectorielle) mais nous ne l'aborderons pas ici.

0	1	1	1	1	0	0	0	0	0	2	2	2	2	2	2	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	2	2	2	2	2	2	2	2	0	0	0	0	0
0	1	1	1	1	0	0	0	2	2	2	2	2	2	2	2	2	2	0	0	0	0
0	1	1	1	1	0	0	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0
0	0	4	4	0	0	0	0	3	3	3	3	3	3	3	3	3	3	0	0	0	0
0	0	4	4	0	0	0	0	3	5	5	5	3	3	2	2	2	3	0	0	0	0
0	0	4	4	0	0	0	0	3	5	5	5	3	3	2	2	2	3	0	0	0	0
0	0	4	4	0	0	0	0	3	5	5	5	3	3	2	2	2	3	0	0	0	0
1	1	1	1	1	1	1	1	3	3	3	3	3	3	2	2	2	3	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	1

La **définition** d'une image matricielle est le nombre de points la composant (le nombre de pixels si c'est une image numérique). Par exemple, 200 pixels x 450 pixels signifie 200 pixels de hauteur et 450 de largeur. La **résolution** est le nombre de pixels par unité de longueur.

Figure 1 - définition 520x520, résolution 170 pixels par cm

Figure 2 - définition 32x32, résolution 32 pixels par cm

RGB

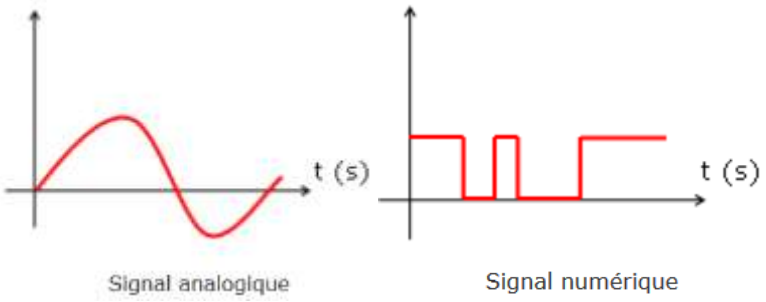
Le codage RGB est un des systèmes les plus utilisés en informatique pour coder les couleurs. RGB signifie Red (rouge), Green (vert), Blue (bleu). Ce système définit pour chaque composante (rouge, vert ou bleu) son intensité entre 0 et 255. Chaque couleur est définie par un triplet (trois nombres), le premier nombre donne l'intensité du rouge, le second donne le vert et le troisième donne le bleu. Les nombres sont ensuite traduits en binaire pour l'ordinateur.

Par exemple :

Couleur	RGB	Hexadécimal (base 16)
Rouge	{255, 0, 0}	{FF, 0, 0}
Vert	{0, 255, 0}	{0, FF, 0}
Bleu	{0, 0, 255}	{0, 0, FF}
Noir	{0, 0, 0}	{0, 0, 0}
Blanc	{255, 255, 255}	{FF, FF, FF}
Gris	{100, 100, 100}	{64, 64, 64}

Les sons

Il existe deux types de signaux : analogique et numérique. Le son est un signal analogique car il varie continuellement au cours du temps. Pour coder ce signal en binaire, il va falloir le transformer en un signal numérique c'est-à-dire qui varie discontinuellement au cours du temps.



Signal analogique Signal numérique

La transformation va se faire par « échantillonnage ». On va prélever les valeurs du signal à intervalles réguliers. De cette manière, on obtiendra une suite de valeurs dite « discrète » qui correspondra à un signal numérique. Il suffira ensuite de coder ces valeurs une par une en binaire.