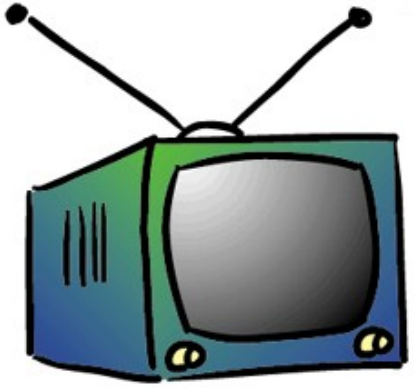
Comment coder le monde en 0 et 1 ? Partie 1



M. Lancelot - Professeur de Technologie

Le monde en 0 et en 1

Dans un monde géré par des ordinateurs, la plupart des informations que nous recevons, que nous émettons ou que nous stockons sont transmises en utilisant le système binaire.

Cela concerne le téléphone, la télévision, internet, la photographie, la musique…

Pour coder l'information, que ce soient des nombres, du texte, des images, de la vidéo ou des sons, un ordinateur utilise uniquement des 0 et des 1.

Techniquement, un ordinateur traite l'information binaire (bit) par groupe de 8 bits appelé octet.

Un octet peut prendre 256 valeurs différentes :

de 0000 0000 à 1111 1111

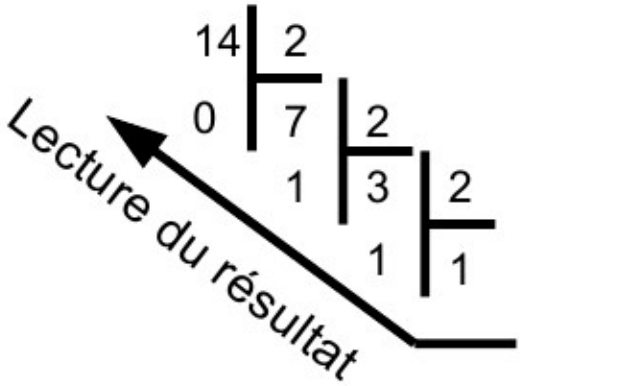
Représentation d'un nombre

Par conversion décimal vers binaire :

1

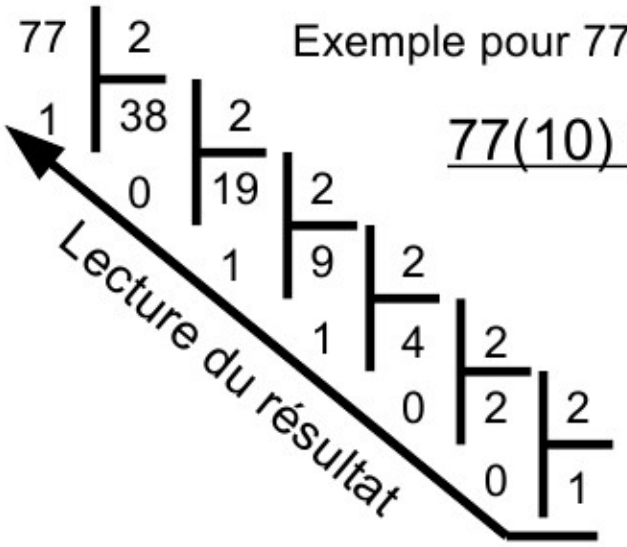
Rappel :

Exemple pour 14(10)



14(10) = 00001110(2)

Exemple pour 77(10)



77(10) = 01001101(2)

Avec un octet :
le nombre maximum en binaire est donc : 1111 1111(2)
le nombre maximum en décimal est donc : 255(10)

Pour des nombres plus grands, il est nécessaire d'utiliser plusieurs octets.

Addition sur les nombres binaires

L'addition de nombres binaires est identique qu'en notation décimale; elle est juste simplifiée parce qu'il n'y a que deux chiffres 0 et 1.

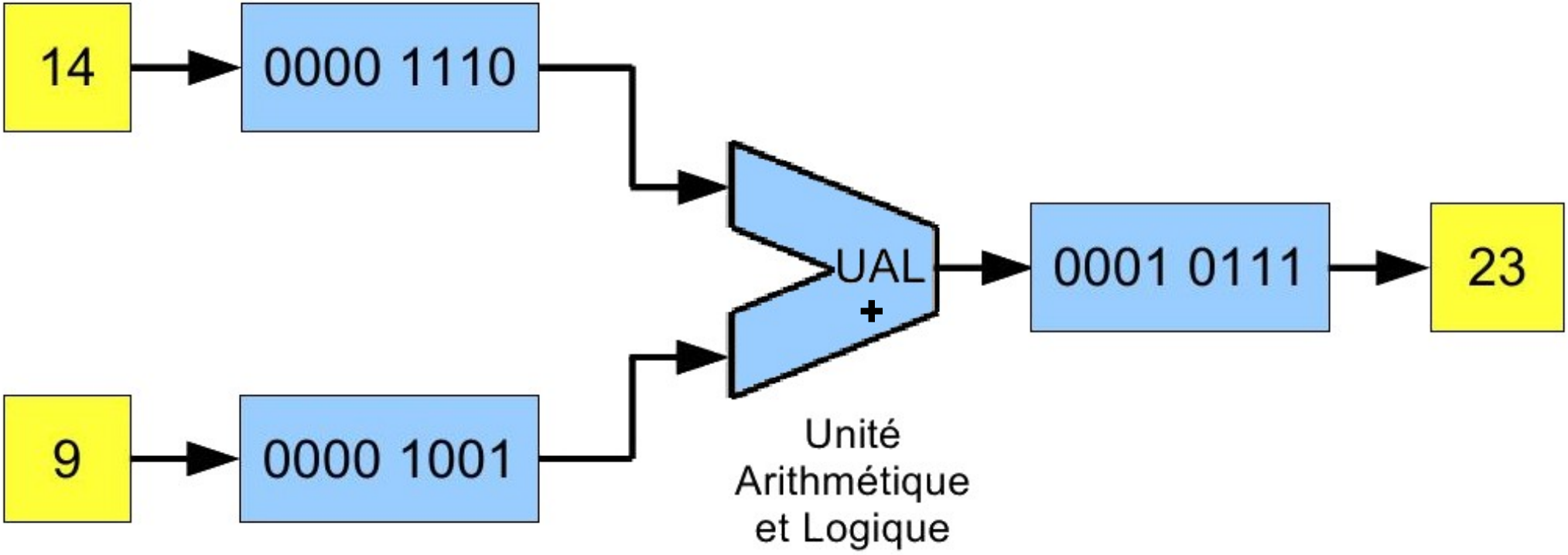
Il n'y a qu'une seule table d'addition en binaire :

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 avec une retenue
1 + 1 + 1= 1 avec une retenue

Exemple de calcul binaire avec 0001110 soit $14_{(10)}$ + 00001001 soit $9_{(10)}$

```
  0 0 0 0 1 1 1 0
+ 0 0 0 0 1 0 0 1
-----------------
  0 0 0 1 0 1 1 1
```

Les additions dans un ordinateur



Rappel :

$14_{(10)} = 0000\ 1110_{(2)}$

$9_{(10)} = 0000\ 1001_{(2)}$

$0001\ 0111_{(2)} = 23_{(10)}$

Représentation d'un nombre négatif

Pour compléter la représentation des entiers, il faut pouvoir écrire des entiers négatifs.

La première idée est d'utiliser le bit le plus à gauche de l'octet pour représenter le signe, les autres bits donnant une valeur absolue :

Exemples :

Signes
0 pour +
1 pour -

0000 0010 ⟶ +2 en décimal
1000 0010 ⟶ −2 en décimal

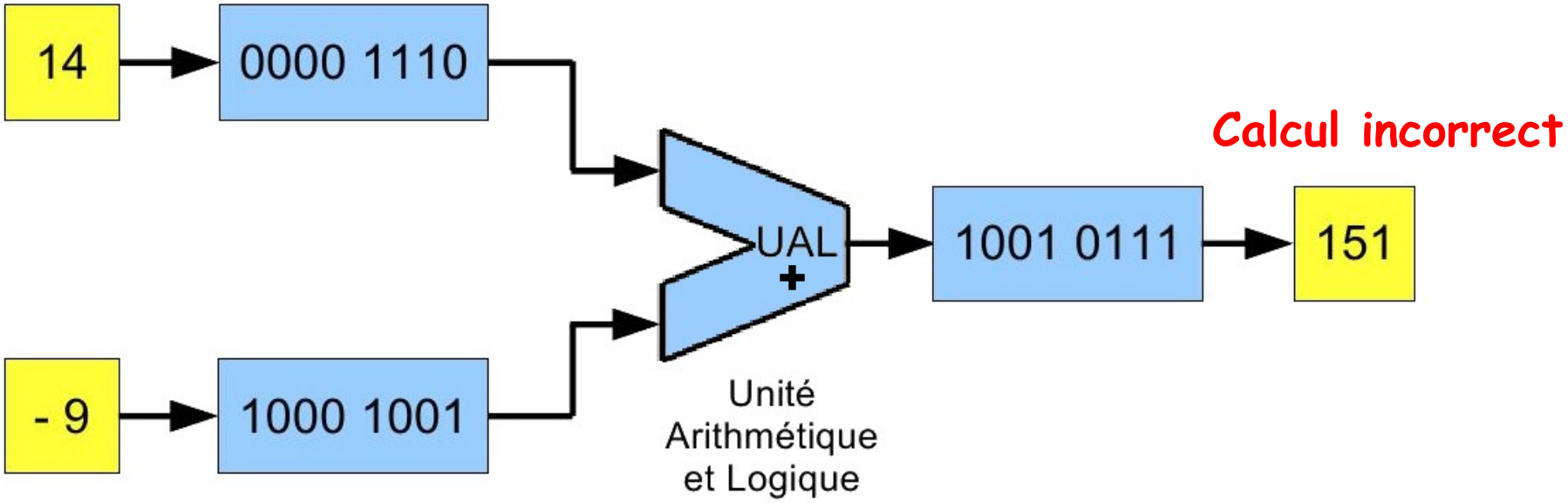
0000 1001 ⟶ +9 en décimal
1000 1001 ⟶ −9 en décimal

Ce codage nécessite d'indiquer au système que l'octet est signé.
En effet, en mode normal $1000\ 0010_{(2)} = 130_{(10)}$
$1000\ 1001_{(2)} = 137_{(10)}$

Inconvénients de ce mode de représentation

Le premier problème de cette représentation est que le zéro (0) peut s'écrire de 2 façons : 0000 0000 ou 1000 0000
respectivement égaux à +0 et –0

Le second problème est que l'addition avec cette représentation est complexe et impose de modifier la table d'addition de l'UAL si un des nombres est négatif.



Représentation d'un nombre négatif
En complément à 2

Les nombres négatifs sont codés de façon à ce que les additions puissent être effectuées facilement par l'UAL.
C'est pourquoi les nombres négatifs sont codés en complément à 2

Les nombres positifs sont représentés de manière usuelle.
Les nombres négatifs sont obtenus par deux opérations successives :
inverser les bits de l'octet, puis ajouter 1

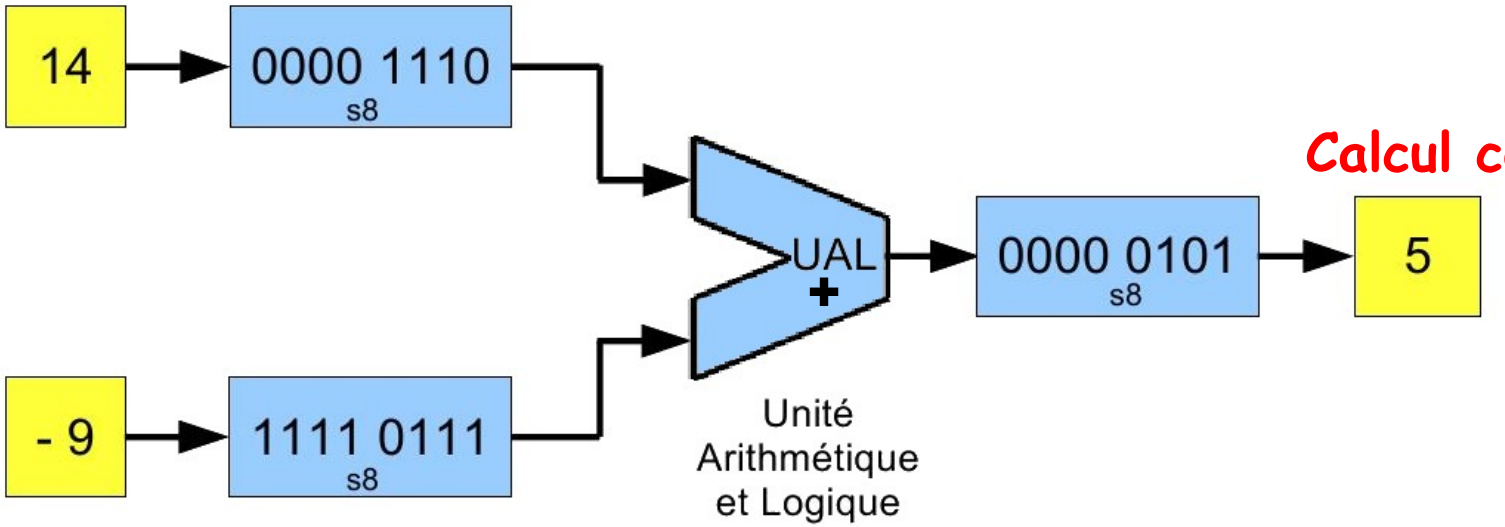
Par exemple pour obtenir –9 en complément à 2 (s8) :

$9_{(10)}$ = 0000 1001

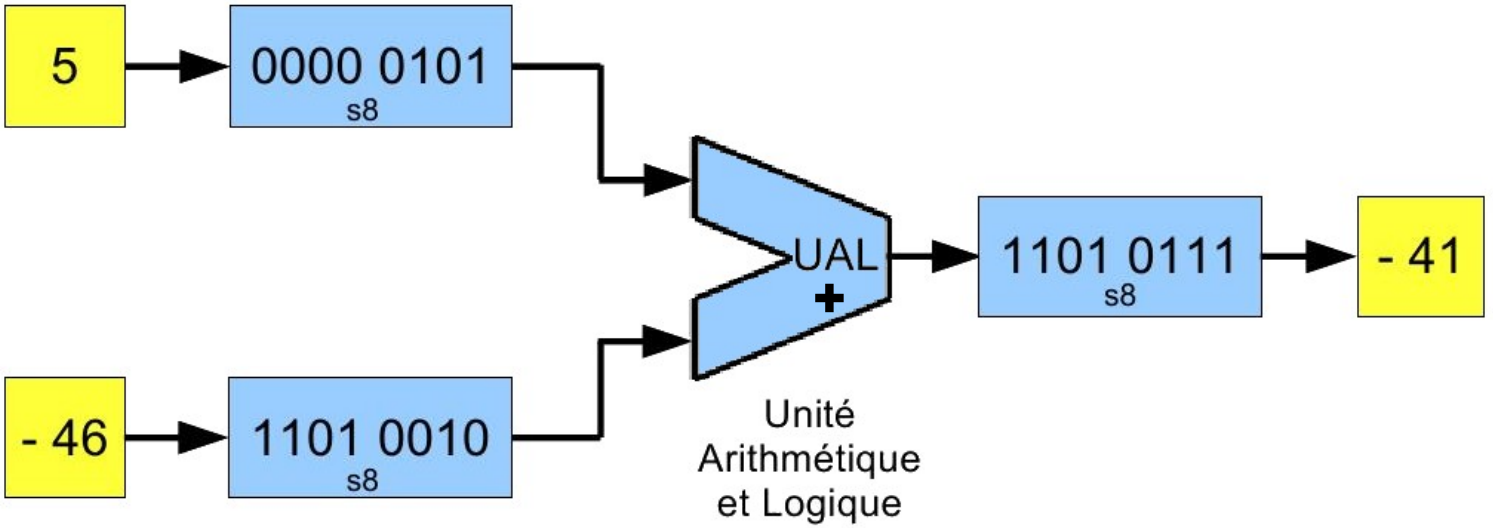
$\overline{9_{(10)}}$ = 1111 0110 inversion des bits

\+ 0000 0001 on ajoute 1

$-9_{(10)}$ = 1111 0111 (s8) complément à 2



Calcul correct



Pour obtenir la valeur décimale à partir d'un format s8

1 1 0 1 0 1 1 1 (s8) le bit à gauche indique que le nombre est négatif

- 0 0 0 0 0 0 0 1 on enlève 1

1 1 0 1 0 1 1 0

0 0 1 0 1 0 0 1 inversion des bits 0010 1001(2) = 41(10)

Représentation d'un nombre négatif
En complément à 2

Avec un octet et avec le codage en complément en 2 :
le nombre maximum positif en binaire est donc : 0111 1111(2) soit +127(10)
le nombre maximum négatif en binaire est donc : 1000 0000(2) soit -128(10)

Le 0 ne peut s'écrire que 0000 0000

Ce codage nécessite d'indiquer
au système que l'octet est signé.

Nombre décimal négatif	Nombre binaire complément à 2
+127	0111 1111
...	...
+3	0000 0011
+2	0000 0010
+1	0000 0001
0	0000 0000
-1	1111 1111
-2	1111 1110
-3	1111 1101
...	...
-127	1000 0001
-128	1000 0000

Représentation binaire des nombres dans les calculateurs

La représentation des décimaux de grandes valeurs (positifs ou négatifs) ou des nombres fractionnaires (à virgule) nécessite l'utilisation de plusieurs octets :
1 octet soit 8 bits de 0 à 255 ou de -128 à +127 en complément à 2
2 octets soit 16 bits de 0 à 62535 ou de -32768 à +32767 en complément à 2
Cela peut même aller jusqu'à 10 octets (soit 80 bits=nombre de 25 chiffres).

Remarque :
L'arithmétique utilisée par les ordinateurs est d'une précision finie et fixe.
Comme les données sont stockées sous forme binaire de tailles différentes, le système informatique est renseigné du format de la donnée :
Exemple : u8 pour 1 octet non signé (unsigned)
s8 pour 1 octet signé (signed)
u16 ou s16
u32 ou s32
etc...

Opérations binaires

Additions binaires :

12(10) = _ _ _ _ _ _ _ _ (2)
- 12(10) = + _ _ _ _ _ _ _ _ (s8)

_ _ _ _ _ _ _ _ (s8)
= _ _ _ (10)

77(10) = _ _ _ _ _ _ _ _ (2)
- 12(10) = + _ _ _ _ _ _ _ _ (s8)

_ _ _ _ _ _ _ _ (s8)
= _ _ _ (10)

77(10) = _ _ _ _ _ _ _ _ (2)
- 46(10) = + _ _ _ _ _ _ _ _ (s8)

_ _ _ _ _ _ _ _ (s8)
= _ _ _ (10)

12(10) = _ _ _ _ _ _ _ _ (2)
- 77(10) = + _ _ _ _ _ _ _ _ (s8)

_ _ _ _ _ _ _ _ (s8)
= _ _ _ (10)

Opérations binaires

Additions binaires :

12(10) = 0 0 0 0 1 1 0 0 (s8)
- 12(10) = + 1 1 1 1 0 1 0 0 (s8)

over:1 ← 0 0 0 0 0 0 0 0 (s8)
= 0 (10)

77(10) = 0 1 0 0 1 1 0 1 (s8)
- 12(10) = + 1 1 1 1 0 1 0 0 (s8)

over:1 ← 0 1 0 0 0 0 0 1 (s8)
= 65 (10)

77(10) = 0 1 0 0 1 1 0 1 (s8)
- 46(10) = + 1 1 0 1 0 0 1 0 (s8)

over:1 ← 0 0 0 1 1 1 1 1 (s8)
= 31 (10)

12(10) = 0 0 0 0 1 1 0 0 (s8)
- 77(10) = + 1 0 1 1 0 0 1 1 (s8)

1 0 1 1 1 1 1 1 (s8)

Nombre négatif

Nécessité de convertir ce nombre complément à 2

1 0 1 1 1 1 1 1 - 1 =

1 0 1 1 1 1 1 0

Inversion : 0 1 0 0 0 0 0 1

= - 65 (10)

Représentation d'un texte

Pour écrire, nous avons besoin :
- de lettres majuscules A B C ,
- de lettres minuscules a b c ,
- de chiffres 4 5 6 ,
- de signes + / # ,
- de divers controles (retour à la ligne par exemple).

La norme ASCII (American Standard Code for Information Interchange soit Code américain normalisé pour l'échange d'information) est la plus répandue et elle établit une correspondance entre une représentation binaire et le texte.

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]

Decimal	Hexadecimal	Binary	Octal	Char
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	=

Représentation d'un texte

Historiquement, l'ASCII définit 128 caractères codés en binaire sur 7 bits (de 000 0000 à 111 1111).

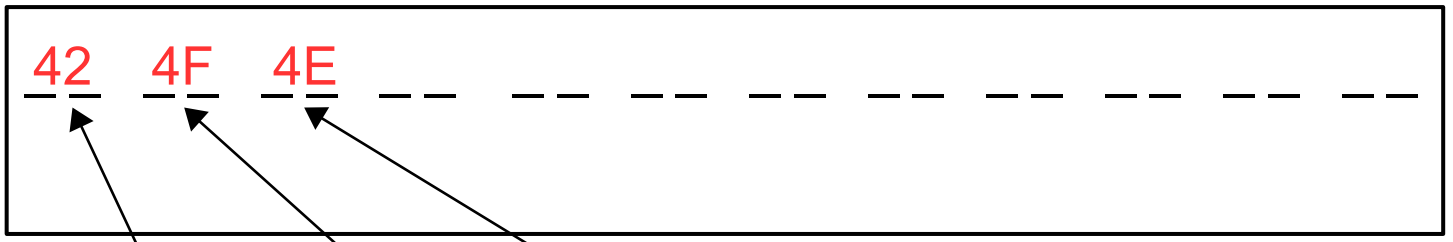
Le code ASCII a été mis au point pour la langue anglaise, il ne contient donc pas de caractères accentués, ni de caractères spécifiques à une langue. Pour coder ce type de caractère il faut recourir à un autre code. Le code ASCII a donc été étendu à 8 bits (un octet) pour pouvoir coder plus de caractères (on parle d'ailleurs de code ASCII étendu...).

Code ASCII étendu – French table

ASCII	Caractère	ASCII	Caractère	ASCII	Caractère	ASCII	Caractère	ASCII	Caractère	ASCII	Caractère	ASCII	Caractère	ASCII	Caractère
0	NUL	32	Space	64	@	96	`	128	€	160		192	À	224	à
1	SOH	33	!	65	A	97	a	129		161	¡	193	Á	225	á
2	STX	34	"	66	B	98	b	130	‚	162	¢	194	Â	226	â
3	ETX	35	#	67	C	99	c	131	ƒ	163	£	195	Ã	227	ã
4	EOT	36	$	68	D	100	d	132	„	164	¤	196	Ä	228	ä
5	ENQ	37	%	69	E	101	e	133	…	165	¥	197	Å	229	å
6	ACK	38	&	70	F	102	f	134	†	166	¦	198	Æ	230	æ
7	BEL	39	'	71	G	103	g	135	‡	167	§	199	Ç	231	ç
8	BS	40	(	72	H	104	h	136	ˆ	168	¨	200	È	232	è
9	HT	41	)	73	I	105	i	137	‰	169	©	201	É	233	é
10	LF	42	*	74	J	106	j	138	Š	170	ª	202	Ê	234	ê
11	VT	43	+	75	K	107	k	139	‹	171	«	203	Ë	235	ë
12	FF	44	,	76	L	108	l	140	Œ	172	¬	204	Ì	236	ì
13	CR	45	-	77	M	109	m	141		173		205	Í	237	í
14	SO	46	.	78	N	110	n	142	Ž	174	®	206	Î	238	î
15	SI	47	/	79	O	111	o	143		175	¯	207	Ï	239	ï
16	DLE	48	0	80	P	112	p	144		176	°	208	Ð	240	ð
17	DC1	49	1	81	Q	113	q	145	‘	177	±	209	Ñ	241	ñ
18	DC2	50	2	82	R	114	r	146	’	178	²	210	Ò	242	ò
19	DC3	51	3	83	S	115	s	147	“	179	³	211	Ó	243	ó
20	DC4	52	4	84	T	116	t	148	”	180	´	212	Ô	244	ô
21	NAK	53	5	85	U	117	u	149	•	181	µ	213	Õ	245	õ
22	SYN	54	6	86	V	118	v	150	–	182	¶	214	Ö	246	ö
23	ETB	55	7	87	W	119	w	151	—	183	·	215	×	247	÷
24	CAN	56	8	88	X	120	x	152		184	¸	216	Ø	248	ø
25	EM	57	9	89	Y	121	y	153	™	185	¹	217	Ù	249	ù
26	SUB	58	:	90	Z	122	z	154	š	186	º	218	Ú	250	ú
27	ESC	59	;	91	[	123	{	155	›	187	»	219	Û	251	û
28	FS	60	<	92	\	124	|	156	œ	188	¼	220	Ü	252	ü
29	GS	61	=	93	]	125	}	157		189	½	221	Ý	253	ý
30	RS	62	>	94	^	126	~	158	ž	190	¾	222	Þ	254	þ
31	US	63	?	95	_	127	DEL	159	Ÿ	191	¿	223	ß	255	ÿ

Binary	Char
0 0 110000	0
0 0 110001	1
0 0 110010	2
0 0 110011	3
0 0 110100	4
0 0 110101	5
0 0 110110	6
0 0 110111	7
0 0 111000	8
0 0 111001	9
0 0 111010	:
0 0 111011	;
0 0 111100	<
0 0 111101	=
0 0 111110	>
0 0 111111	?
0 1000000	@
0 1000001	A
0 1000010	B
0 1000011	C
0 1000100	D
0 1000101	E
0 1000110	F
0 1000111	G
0 1001000	H
0 1001001	I
0 1001010	J
0 1001011	K
0 1001100	L
0 1001101	M
0 1001110	N
0 1001111	O
0 1010000	P
0 1010001	Q
0 1010010	R
0 1010011	S
01010100	T
01010101	U
01010110	V
01010111	W
01011000	X
01011001	Y
01011010	Z
01011011	[
01011100	\
01011101	]
01011110	^
01011111	_

Message à coder en ASCII (Héxa) → BONJOUR:2019



0 1 0 0 0 0 1 0

0 1 0 0 1 1 1 1

0 1 0 0 1 1 1 1 (N)

Binary	Char
0 0 110000	0
0 0 110001	1
0 0 110010	2
0 0 110011	3
0 0 110100	4
0 0 110101	5
0 0 110110	6
0 0 110111	7
0 0 111000	8
0 0 111001	9
0 0 111010	:
0 0 111011	;
0 0 111100	<
0 0 111101	=
0 0 111110	>
0 0 111111	?
0 1000000	@
0 1000001	A
0 1000010	B
0 1000011	C
0 1000100	D
0 1000101	E
0 1000110	F
0 1000111	G
0 1001000	H
0 1001001	I
0 1001010	J
0 1001011	K
0 1001100	L
0 1001101	M
0 1001110	N
0 1001111	O
0 1010000	P
0 1010001	Q
0 1010010	R
0 1010011	S
0 1010100	T
0 1010101	U
0 1010110	V
0 1010111	W
0 1011000	X
0 1011001	Y
0 1011010	Z
0 1011011	[
0 1011100	\
0 1011101	]
0 1011110	^
0 1011111	_

Message à décoder en ASCII (Héxa) →

48 45 4C 4C 4F 5B 57 4F 52 44 5D _ _

0 1 0 0 0 1 0 1

0 1 0 0 1 0 0 0

Message :

HELLO[WORD]