Chapitre 3

Codage de l'information

Table des matières

1 Base b

Définition 1 :

Soit b un entier (avec $b \geq 2$).
Tout entier naturel n peut s'écrire d'une façon unique sous la forme d'une somme de termes de la forme $x_i b^i$ où les nombres i sont des entiers naturels et les nombres x_i sont des entiers naturels compris entre 0 et $b-1$.
Les entiers entre 0 et $b-1$ constituent les chiffres de la base b.
On note alors :

$$n = x_n x_{n-1} ... x_1 x_{0_{(b)}} = x_n b^n + x_{n-1} b^{n-1} + ... + x_1 b + x_0$$

Exemple 1 :

Pour $n = 952$. On a :

- $n = 12302_{(5)}$
- $n = 1022021_{(3)}$
- $n = 2530_{(7)}$
- $n = 952_{(10)}$

2 Système binaire

Définition 2 :

Le système binaire définit la représentation d'un nombre en base 2.
Cette représentation utilise donc uniquement des 0 et des 1, nommé **bit**(bit est l'abréviation de BInary Digit (chiffre binaire)).
Un **octet** (byte en anglais) est une suite de 8 bits.
Un groupe de bits est appelé un **mot**.
Dans un mot binaire le bit le plus à gauche est le **MSB** (Most Significant Bit = bit de poids fort) et le bit le plus à droite est le **LSB** (Least Significant Bit = bit de poids faible)

Exemple 2 :

- $5_{(10)} = 101_{(2)}$
- $15_{(10)} = 1111_{(2)}$
- $12_{(10)} = 1100_{(2)}$
- $256_{(10)} = 100000000_{(2)}$

Exercice 1 : Déterminer la représentation en base 2 des entiers suivants :

(a) $n = 57$

(b) $n = 58$

(c) $n = 85$

(d) $n = 9872$

3 Système hexadécimal

Définition 3 :

La numération hexadécimale utilise 16 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F.
Le A représente donc le nombre 10.
Le F représente le nombre 15.

Exemple 3 :

- $AA_{(16)} = 170_{(10)}$
- $15_{(16)} = 21_{(10)}$
- $ABCD_{(16)} = 43981_{(10)}$
- $FFF_{(16)} = 4095_{(10)}$

Exercice 2 : Déterminer la représentation en base 16 des entiers suivants :

(a) $n = 157$

(b) $n = 256$

(c) $n = 1024$

(d) $n = 10000$

4 Représentation des entiers relatifs

4.1 Le complement à 1

Définition 4 :

On fixe un nombre de bits N.
Le complement à 1 de la représentation binaire d'un entiers consiste à inverser tous les bits.

Exemple 4 :

Sur 8 bits, le complément à 1 de $10011010_{(2)}$ est $01100101_{(2)}$.

Exercice 3 : On travaille ici avec 8 bits.

1. Déterminer la représentation binaire de $n = 154$.
2. Déterminer son complément binaire p.
3. Déterminer la somme de $n + p$.

4.2 Le complement à 2

Définition 5 :

Si on utilise des mots de N bits, la méthode du complément à 2 permet la représentation des entiers relatifs entre -2^{N-1} et $2^{N-1} - 1$.
La représentation d'un nombre négatif se fait en deux étapes :

- Calcul du complément à 1 de la représention binaire de la valeur absolue du nombre.
- Calcul du complément à 2 : Ajouter 1 au complément à 1.

Dans cette représentation, le bit de poids fort d'un entier positif est 0 et le bit de poids fort d'un entier négatif est 1. On parle alors du bit de signe.

Exemple 5 :

Pour représenter le nombre (-19) avec 8 bits :

- On a $19 = 00010011_{(2)}$, son complément à 1 est 11101100.
- On ajoute 1 : 11101101

On obtient donc : $-19 = 11101101$.

Exercice 4 : Déterminer la représentation binaire sur 4 bits par la méthode du complément à 2 des entiers suivants :

(a) $n = -1$

(b) $n = -3$

(c) $n = -7$

5 Représentation des nombres à virgule

5.1 Nombre à virgule en base 2

En base 10 on a :
$47,159_{(10)} = 4 \times 10^1 + 7 \times 10^0 + 1 \times 10^{-1} + 5 \times 10^{-2} + 9 \times 10^{-3}$

Par analogie, en base 2 :
$10,101_{(2)} = 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

Définition 6 :

On appelle nombre dyadique tout nombre réel qui admet une écriture en base 2 donc la partie décimale est finie.

Exemple 6 :

- $3,125$ est dyadique, en effet : $3,125 = 2 + 1 + 2^{-3} = 11,001_{(2)}$
- $\frac{1}{5}$ n'est pas dyadique : $\frac{1}{5} = 0,2_{(10)} = 0,0011\ 0011\ ..._{(2)}$

Exercice 5 :

(a) Déterminer l'écriture en base 10 des nombres suivants :

- $a = 101,101_{(2)}$
- $b = 100,1101_{(2)}$
- $c = 0,10101010..._{(2)}$

(b) Déterminer l'écriture en base 2 des nombres suivants :

- $a = 7,25$
- $b = 15,15625$
- $c = 0,15$

5.2 Norme IEEE 754

Définition 7 :

Un nombre à virgule x est représenté sous la forme suivante :

$$x = s.m.2^n$$

avec :

- s est le signe du nombre (+ ou -).
- n l'exposant, un entier relatif.
- m la mantisse, un nombre à virgule compris entre 1 inclus et 2 exclu.

Exemple 7 :

- $0.25 = 1, 0.2^{-2}$
- $9,625 = 2^3 + 2^0 + 2^{-1} + 2^{-3} = 1, (001101)_{(2)}.2^{11_{(2)}}$

Par exemple, quand on utilise 64 bits pour représenter un nombre à virgule, on utilise :

- 1 bit pour le signe.
- 11 bits pour l'exposant
- 52 bits pour la mantisse.

Le signe + est représenté par 0 et le signe - par 1.
L'exposant n est un entier relatif compris entre -1 022 et 1 023 ; on le représente comme l'entier naturel n + 1 023, qui est compris entre 1 et 2 046. Les deux entiers naturels 0 et 2 047 sont réservés pour des situations exceptionnelles.
La mantisse m est un nombre binaire à virgule compris entre 1 inclus et 2 exclu, comprenant 52 chiffres après la virgule. Comme cette mantisse est comprise entre 1 et 2, elle a toujours un seul chiffre avant la virgule et ce chiffre est toujours un 1 ; il est donc inutile de le représenter et on utilise les 52 bits pour représenter les 52 chiffres après la virgule

6 Représentation d'un texte

6.1 Code ASCII

L'encodage ASCII (American Standard Code for Information Interchange) est créé en 1961. Le jeu de caractères ASCII utilise 7 bits et dispose donc de 128 caractères numérotés de 0 à 127.
En python, pour récupérer le code d'un caractère on utilise la fonction `ord` :
Exemple :

```
>>> ord('p')
>>> 112
>>> for i in 'Bonjour' :
  print(hex(ord(i)))
0x42
0x6f
0x6e
0x6a
0x6f
0x75
0x72
```

Table de conversion en code ASCII :

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

6.2 Code ISO-8859-1

Le code ASCII ne prévoit pas d'encoder les lettres accentuées. C'est pour répondre à ce problème qu'est née la norme ISO-8859-1. Cette norme reprend les mêmes principes que l'ASCII, mais les nombres binaires associés à chaque caractère sont codés sur 8 bits, ce qui permet d'encoder jusqu'à 256 caractères.
Cette norme va être principalement utilisée dans les pays européens puisqu'elle permet d'encoder les caractères utilisés dans les principales langues européennes (la norme ISO-8859-1 est aussi appelée "latin1" car elle permet d'encoder les caractères de l'alphabet dit "latin")

6.3 Unicode

Unicode est une table qui regroupe tous les caractères existants au monde, il ne s'occupe pas de la façon dont les caractères sont codés dans la machine. Unicode accepte plusieurs systèmes de codage : UTF-8, UTF-16, UTF-32.

Pour encoder les caractères Unicode, UTF-8 utilise un nombre variable d'octets : les caractères "classiques" (les plus couramment utilisés) sont codés sur un octet, alors que des caractères "moins classiques" sont codés sur un nombre d'octets plus important (jusqu'à 4 octets). Un des avantages d'UTF-8 c'est qu'il est totalement compatible avec la norme ASCII : Les caractères Unicode codés avec UTF-8 ont exactement le même code que les mêmes caractères en ASCII.