Note de COURS
4 : Représentation de l'information

Entiers non-signés

Base 10

- Un chiffre en base 10 peut prendre une valeur entre 0 et 9
- Les chiffres utilisent la numération en système positionnel (la position du chiffre par rapport aux autres est une information en soit)

REPRESENTATION MATHEMATIQUE

3-chiffres

d_2 d_1 d_0

3ème chiffre 2ème chiffre 1er Chiffre

EXEMPLE

2 0 9

3ème chiffre 2ème chiffre 1er Chiffre

- Un entier à n chiffre représenté par $d_{n-1}d_{n-2}\ldots d_1d_0$ correspond à la valeur décimale

$$D = \sum_{i=0}^{n-1} d_i.10^i = d_{n-1}.10^{n-1} + d_{n-2}.10^{n-2} + \cdots + d_1.10^1 + d_0.10^0$$

- Valeur maximum :

Nombre de chiffre	Valeur max.	Intervalle
1	$9 = 10^1-1$	$0 \rightarrow 9$ $(0 \rightarrow 10^1-1)$
2	$99 = 10^2-1$	$0 \rightarrow 99$ $(0 \rightarrow 10^2-1)$
3	$999 = 10^3-1$	$0 \rightarrow 999$ $(0 \rightarrow 10^3-1)$
4	$9999 = 10^4-1$	$0 \rightarrow 9999$ $(0 \rightarrow 10^4-1)$
5	$99999 = 10^5-1$	$0 \rightarrow 99999$ $(0 \rightarrow 10^5-1)$
…		
n	$999\ldots999 = 10^n-1$	$0 \rightarrow 999\ldots999$ $(0 \rightarrow 10^n-1)$

Base 2

- La base 10 est la plus utilisé par les humains (bien qu'ils utilisent encore d'autre bases : 12, 16, 24, 60, 365)
- La base 2 et ses dérivées (principalement 8, 16) sont très utilisées dans le monde numérique
- On parle de bit pour <u>bi</u>nary digi<u>t</u> au lieu de chiffres décimaux
- Le **bit** est aussi une unité d'information qui peut être utilisée pour manipuler des variables booléennes
- Un nombre binaire fait référence à une séquence de bit en représentation positionnelle ($b_{n-1}b_{n-2}\ldots b_1b_0$)

$b_{n-1}\, b_{n-2} \;\ldots\; b_1\, b_0$

Plus significatif
Bit le + à gauche

Moins significatif
Bit le + à droite

- La conversion base-2 vers base-10 peut se faire en utilisant la formule suivante :

$$D = \sum_{i=0}^{n-1} b_i.2^i = b_{n-1}.2^{n-1} + b_{n-2}.2^{n-2} + \cdots + b_1.2^1 + b_0.2^0$$

- Pour éviter les confusions de représentation en ajoute en indice la base : $(b_{n-1}b_{n-2}\ldots b_1b_0)_2$
- Valeur maximum :

Nombre de chiffre	Valeur max.	Intervalle
1	$1_2 = 2^1-1$	$0 \rightarrow 1_2$
2	$11_2 = 2^2-1$	$0 \rightarrow 11_2$
3	$111_2 = 2^3-1$	$0 \rightarrow 111_2$
4	$1111_2 = 2^4-1$	$0 \rightarrow 1111_2$
5	$11111_2 = 2^5-1$	$0 \rightarrow 11111_2$
…		
n	$111\ldots111_2 = 2^n-1$	$0 \rightarrow 111\ldots111_2$

Base 16

- Système de représentation compact des nombres binaires
- Basé sur 16 symboles
- La conversion base 16 ⬄ base 2 est assez facile (1 chiffre base16 ⬄ 4 chiffres en base 2)
- Exemple :
  - $(AB)_{16} = (1010\ 1011)_2 = 10.16^1 + 11.16^0 = (171)_{10}$
  - $(E9)_{16} = (1110\ 1001)_2 = 14.16^1 + 9.16^0 = (233)_{10}$

$0_{hex} = 0_{dec} = 0_{oct}$	0	0	0	0
$1_{hex} = 1_{dec} = 1_{oct}$	0	0	0	1
$2_{hex} = 2_{dec} = 2_{oct}$	0	0	1	0
$3_{hex} = 3_{dec} = 3_{oct}$	0	0	1	1
$4_{hex} = 4_{dec} = 4_{oct}$	0	1	0	0
$5_{hex} = 5_{dec} = 5_{oct}$	0	1	0	1
$6_{hex} = 6_{dec} = 6_{oct}$	0	1	1	0
$7_{hex} = 7_{dec} = 7_{oct}$	0	1	1	1
$8_{hex} = 8_{dec} = 10_{oct}$	1	0	0	0
$9_{hex} = 9_{dec} = 11_{oct}$	1	0	0	1
$A_{hex} = 10_{dec} = 12_{oct}$	1	0	1	0
$B_{hex} = 11_{dec} = 13_{oct}$	1	0	1	1
$C_{hex} = 12_{dec} = 14_{oct}$	1	1	0	0
$D_{hex} = 13_{dec} = 15_{oct}$	1	1	0	1
$E_{hex} = 14_{dec} = 16_{oct}$	1	1	1	0
$F_{hex} = 15_{dec} = 17_{oct}$	1	1	1	1

Unité d'information

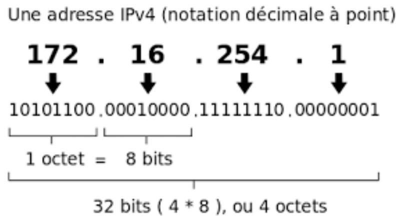
- Octet = (Byte en anglais) = 8 bits
- Dans le système internationale, Kilo, Mega, Giga, Tera font référence à des multiples en base 10 (ex: Kilomètre = 10^3).
- Dans le domaine informatique on utilise les Ko, Mo, Go, To correspondant à des puissances de 2.

Multiples de l'octet :
préfixes décimaux du SI et mésusages

Nom	Symbole	Valeur	Mésusage [a]
kilooctet	ko	10^3	2^{10}
mégaoctet	Mo	10^6	2^{20}
gigaoctet	Go	10^9	2^{30}
téraoctet	To	10^{12}	2^{40}
pétaoctet	Po	10^{15}	2^{50}
exaoctet	Eo	10^{18}	2^{60}
zettaoctet	Zo	10^{21}	2^{70}
yottaoctet	Yo	10^{24}	2^{80}

Exemples d'application

Protocole internet

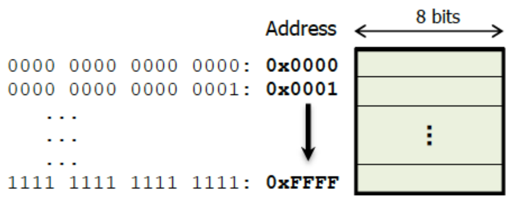


- Les adresses IPv4 (*Internet Protocol*) sont l'équivalent du numéro de téléphone dans l'internet (ex : 192.168.0.1).
- Elles sont représentées sur 32 bits (4 octets) mais représentées par convention sous la forme de 4 nombres décimaux compris entre 0 et 255.
- Permets d'adresser 2^{32} = 4 294 967 296 périphériques différents.
  Il n'y plus d'adresses disponibles depuis quelques années. Nous sommes passées à IPv6 qui représente les adresses sur 128 bits

Représentation de niveau de gris

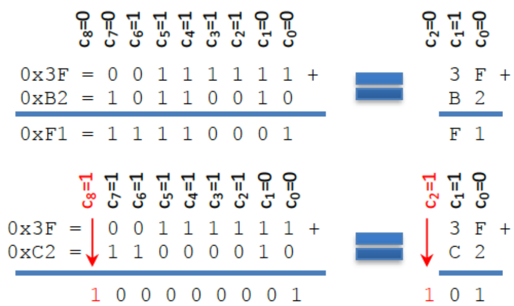
- Les niveaux de gris sont souvent représentés sur 8 bits.
- Chaque pixel d'une image en niveau de gris est stocké sur 1 octet.
- La valeur 0 correspond au noir et 255 au blanc.

Adresses mémoires



- Il existe une relation entre la taille du bus et la quantité de mémoire adressable.
- Exemple :
  Si l'on a un processeur avec un bus sur 16 bits, il pourra donc faire référence à 2^{16} adresses différentes.
  Si chaque adresse mémoire contient 1 octet, le processeur pourra gérer jusqu'à 2^{16} octets = 64Ko. Une représentation graphique avec des adresses en hexa est donnée dans le dessin suivant.

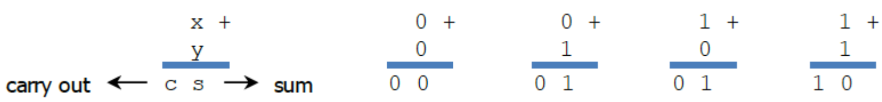
Addition de nombre non signé :



- Exemple avec 2 nombres de 8 bits, en binaire.
- Remarquez que l'addition de 2 nombres de n bits génère une retenue sortante c_n (aussi appelée *carry*), et que c_0 fait référence à la retenue entrante (généralement à 0).
- Si la retenue sortante est égale à 0, alors le résultat est représentable sur n bits, sinon (=1) il y a eu un dépassement de capacité (*overflow*) car il faudrait n+1 bit pour représenter le résultat.

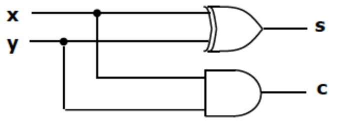
Implémentation d'un additionneur 1 bit :

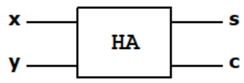
- L'addition d'un bit sans retenue entrante correspond à ce que l'on appelle un *Half-Adder (HA)*.



- L'utilisation de l'algèbre booléenne permet de concevoir le circuit du HA qui réalise x+y.

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0





- Si l'on considère qu'il y a aussi une retenue entrante c_i, alors on obtient ce que l'on appelle un *Full-Adder (FA).*

x_i	y_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

c_i \ x_iy_i	00	01	11	10
0	0	1	0	1
1	1	0	1	0

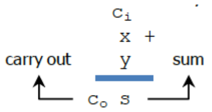
$$s_i = \overline{x_i}y_i\overline{c_i} + x_i\overline{y_i}\,\overline{c_i} + \overline{x_i}\,\overline{y_i}c_i + x_iy_ic_i$$

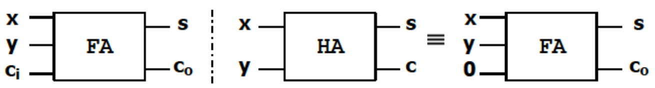
$$s_i = (x_i \oplus y_i)\overline{c_i} + \overline{(x_i \oplus y_i)}c_i$$

$$s_i = x_i \oplus y_i \oplus c_i$$

c_i \ x_iy_i	00	01	11	10
0	0	0	1	0
1	0	1	1	1

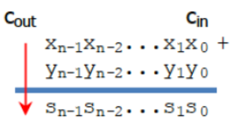
$$c_{i+1} = x_iy_i + x_ic_i + y_ic_i$$

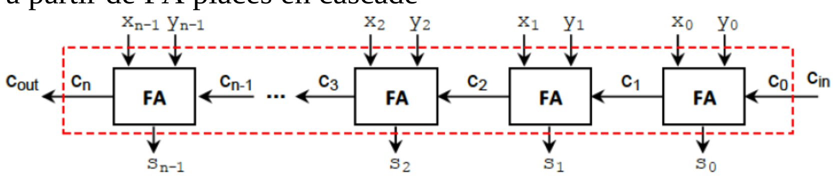




Implémentation d'un additionneur n bit :

- Généralement construit à partir de FA placés en cascade





Entiers signés

- Il existe 3 façons de représenter un nombre signé :
    - Signe magnitude
    - Complément à 1
    - Complément à 2

Représentation signe-magnitude

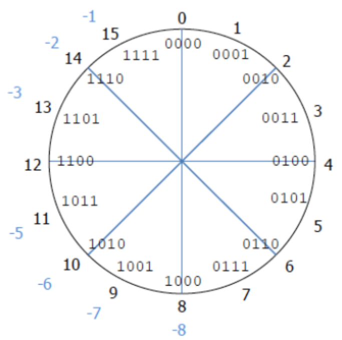
- Le signe et la magnitude (valeur) du nombre sont représentés séparément.
- Le bit de poids fort représente le signe et les autres la magnitude.
- Il faut un circuit spécifique pour la soustraction.
- Exemple :
    - 0110 = +6
    - 1110 = -6

Représentation complément à 1

- Le complément à 1 d'un nombre B se définit comme étant $K=(2^n-1)-B$ avec *n* le nombre de bit utilisé pour représenter un nombre.

- $K = \sum_{i=0}^{n-1} k_i 2^i$ et $B = \sum_{i=0}^{n-1} b_i 2^i$
- (2^n-1) est le plus grand entier non signé.
- Remarques :
  - Les entiers positifs sont représentés normalement
  - Le codage des entiers négatifs se fait par inversion des bits du nombre positif correspondant
  - $\sum_{i=0}^{n-1} k_i 2^i = (2^n - 1) - \sum_{i=0}^{n-1} b_i 2^i \rightarrow \sum_{i=0}^{n-1} (k_i + b_i) 2^i = 2^n - 1 \rightarrow k_i + b_i = 1, \forall i \Rightarrow k_i = \overline{b_i}$
  - Les soustractions ne nécessitent pas de circuits particuliers,
  - La valeurs 0 a deux représentations (+0 et -0) (ex : avec n=4 on a 0000=+0 et 1111=-0).
- Exemple :
  - Représentation en complément à 1 de (-4) : on sait que (+4)=0100, on inverse les bits ce qui donne 1011=(-4)

Représentation complément à 2



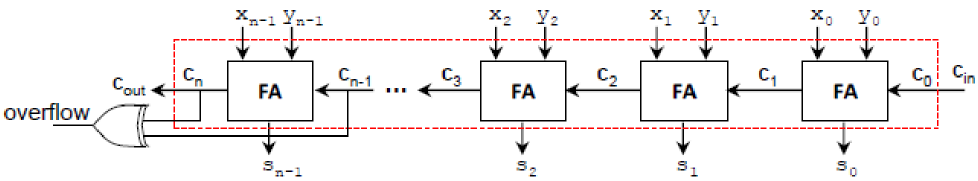
- Très utilisé car il permet d'effectuer simplement les additions et les soustractions avec unicité de la représentation (pas de +0 et -0)
- Le complément à 2 d'un nombre B se définit comme étant K=(2^n-1)-B+1 avec *n* le nombre de bit utilisé pour représenter un nombre.
- $K = \sum_{i=0}^{n-1} k_i 2^i$ et $B = \sum_{i=0}^{n-1} b_i 2^i$
- Avec n bits on peut représenter des nombres en -2^{n-1} et 2^{n-1}-1
- On constate que dans ce système on peut obtenir l'inverse en inversant tous les bits d'un nombre (comme dans le complément à 1) et en ajoutant 1.
- Exemple :
  - Représentation en complément à 1 de (-4) : on sait que (+4)=0100, on inverse les (1011) et on ajoute 1 ce qui donne 1100=(-4)

Synthèse

n=4	REPRESENTATION SIGNEE		
b3b2b1b0	Signe-magnitude	Complément à 1	Complément à 2
0 0 0 0	0	0	0
0 0 0 1	1	1	1
0 0 1 0	2	2	2
0 0 1 1	3	3	3
0 1 0 0	4	4	4
0 1 0 1	5	5	5
0 1 1 0	6	6	6
0 1 1 1	7	7	7
1 0 0 0	0	-7	-8
1 0 0 1	-1	-6	-7
1 0 1 0	-2	-5	-6
1 0 1 1	-3	-4	-5
1 1 0 0	-4	-3	-4
1 1 0 1	-5	-2	-3
1 1 1 0	-6	-1	-2
1 1 1 1	-7	0	-1
Interval pour n bits	$[-(2^{n-1}-1), 2^{n-1}-1]$	$[-(2^{n-1}-1), 2^{n-1}-1]$	$[-2^{n-1}, 2^{n-1}-1]$

Nombres signés : Addition & Soustraction

- Passe par la représentation en complément à 2, ce qui permet d'utiliser le même circuit pour l'addition et la soustraction
- Dans ce cas, le bit de retenue sortante n'indique pas nécessairement un overflow. Il y a overflow lors de l'addition de 2 nombres de n bits si les bits de retenue c_n et c_{n-1}sont différent (Overflow = $c_n \oplus c_{n-1}$)
- Attention : dans le cas de l'addition de 2 nombres de n et m bits avec $n \neq m$. Il est nécessaire de faire l'extension de signe du nombre encodé sur le plus petit nombre de bit.



Multiplication

- Non-signée : similaire à la multiplication scolaire
- Signée en Cà2 : attention à la gestion du signe. 2 solutions
  - Solution 1 :
    - On test le signe des 2 opérandes. Si l'une des 2 est négatives alors on inverse son signe,
    - On fait la multiplication scolaire,
    - On inverse le signe du résultat si condition en 1. vraie
  - Solution 2 :
    - On pense bien à faire l'extension de signe dans les produits partiels

(-5)					1	0	1	1
x 3					0	0	1	1
					1	0	1	1
				1	0	1	1	
			0	0	0	0		
		0	0	0	0			
33!=	0	0	1	0	0	0	0	1

Complément-à-2 direct
(ne fonctionne pas)

Inv(-5)					1	0	1	1
5					0	1	0	1
x 3					0	0	1	1
					0	1	0	1
				0	1	0	1	
			0	0	0	0		
		0	0	0	0			
15 =	0	0	0	0	1	1	1	1
Inv(15)	1	1	1	1	0	0	0	1

Solution 1

(-5)					1	0	1	1
x 3					0	0	1	1
	1	**1**	**1**	**1**	1	0	1	1
	1	**1**	**1**	1	0	1	1	
	0	**0**	0	0	0	0		
	0	0	0	0	0			
33!=	1	1	1	1	0	0	0	1

Solution 2

Code binaires

- On sait que n bits peut représenter 2^n informations différentes (nombres, caractères, couleurs …)
- Pour représenter K informations différentes il faut $\lceil log_2(K) \rceil$ bits

Exemple Code ASCII sur 7 bits

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

- Une alternative est l'encodage unicode UTF-16 qui permets de représenter plus de caractères différents

Encodage BCD

- Dans ce format les nombres décimaux sont représentés en binaires sur 4 bits

BCD	decimal #
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Code de Gray

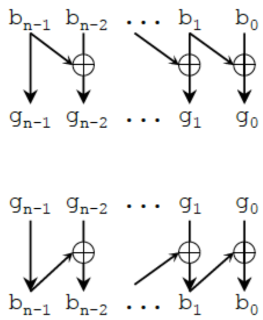
- Il est aussi appelé code binaire réfléchi.
- Dans ce format 2 nombres consécutifs ne diffèrent que d'une position.
- Exemple : 5 est codé par 0111 et 6 par 0101 (seul le 3ème bit change)

Codage décimal	Codage binaire naturel	Codage Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

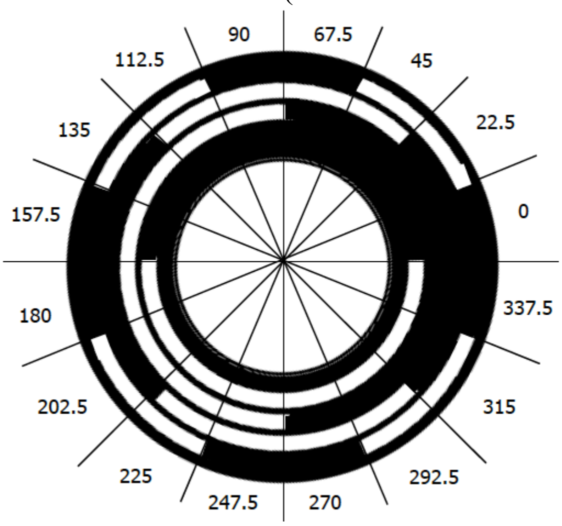
g_1g_0
0 0
0 1
1 1
1 0

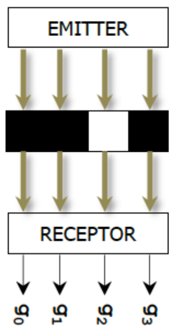
Decimal Number	$b_2b_1b_0$	$g_2g_1g_0$
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

$g_3g_2g_1g_0$
0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0
1 1 0 0
1 1 0 1
1 1 1 1
1 1 1 0
1 0 1 0
1 0 1 1
1 0 0 1
1 0 0 0



- **Exemple d'application** : capteur de mesure d'angle sur 4 bits
    - Les 360° sont divisés en 2^4=16 intervalles.
    - Un signal lumineux passe à travers maximum 4 capteurs, ce qui permets de déterminer l'origine du signal.
    - Le code de gray est préférable au code binaire standard, notamment lorsque le signal est entre 2 intervalles (minimise l'erreur de représentation)





Angle	$g_3g_2g_1g_0$
0 - 22.5	0 0 0 0
22.5 - 45	0 0 0 1
45 - 67.5	0 0 1 1
67.5 -90	0 0 1 0
90 - 112.5	0 1 1 0
112.5 - 135	0 1 1 1
135 - 157.5	0 1 0 1
157.5 - 180	0 1 0 0
180 - 202.5	1 1 0 0
202.5 - 225	1 1 0 1
225 - 247.5	1 1 1 1
247.5 - 270	1 1 1 0
270 - 292.5	1 0 1 0
292.5 - 315	1 0 1 1
315 - 337.5	1 0 0 1
337.5 - 360	1 0 0 0