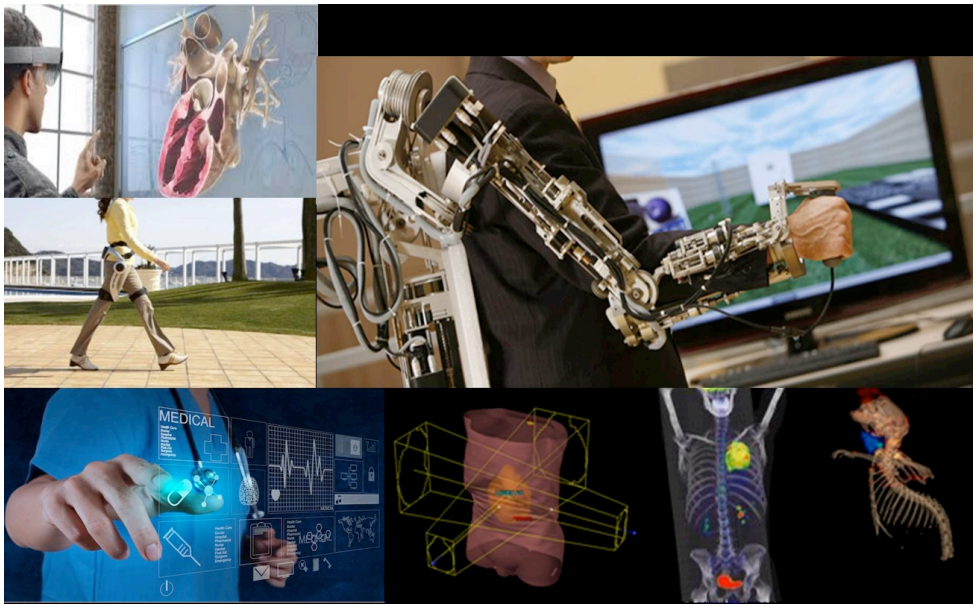
IBIOM: Ingénierie pour le Biomédical



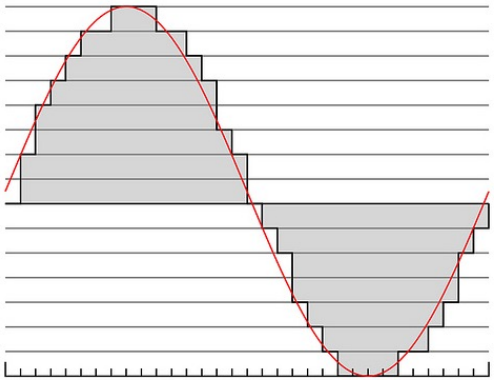
Su RUAN

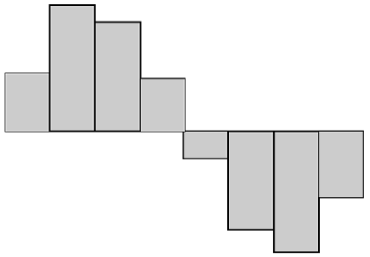
Professeure des universités

su.ruan@univ-rouen.fr

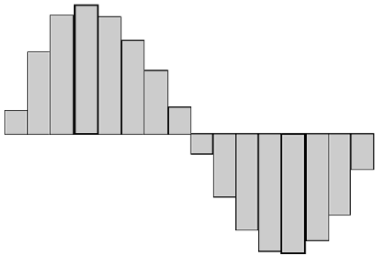
Donnes numériques

- Conversion Analog vs. Digital

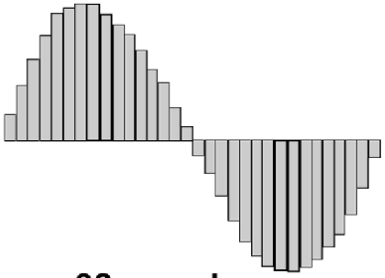




8 samples

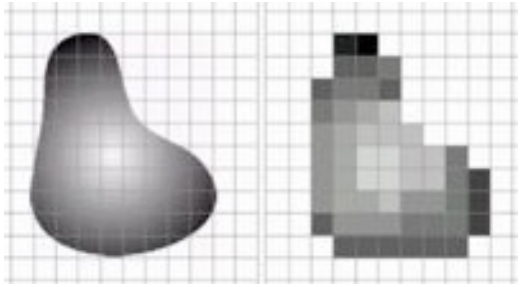


16 samples



32 samples

pixel



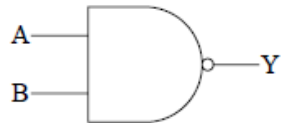
Donnes numériques

- Système numérique (Digital system)
  - Système binaire, octet
    0 et 1, 8 bits -> octet
  - Système hexadécimal
    0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - Conversion
    $12 = 8+4 = 2^3+2^2 = (1100)_2$
    $100 = 96+4 = 6*16+4 = (64)_{16}$
- Opérateurs logiques (système binaire)
  - $NON: \bar{A}$
  - OU: A+B
  - ET: AB
  - OU Exclusif: $A \oplus B = \bar{A}B + A\bar{B}$

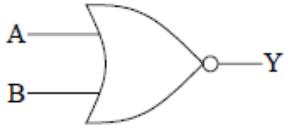
Donnes numériques

- Circuits logiques

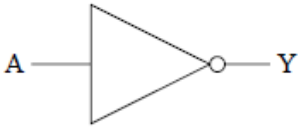
NON AND (ET)



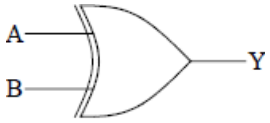
NON OR (OU)

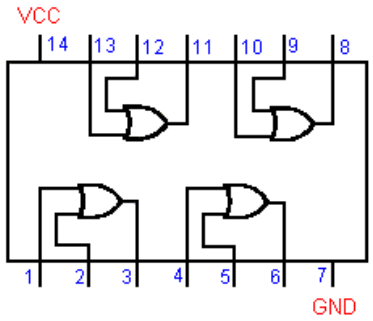


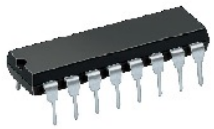
NOT (NON)



(XOR) OU exclusif

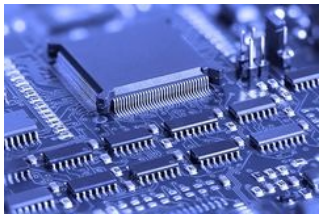


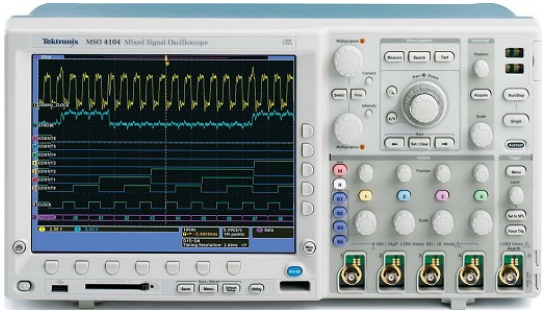




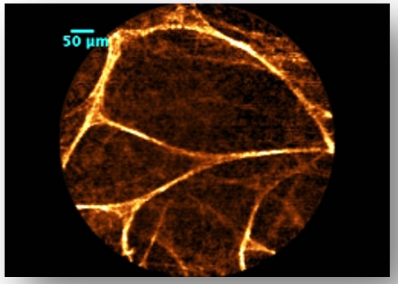
Appareils numériques

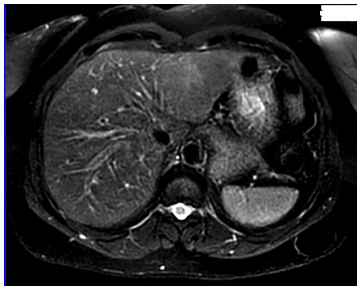
- Appareils médicaux





Instrumentation biomédicale





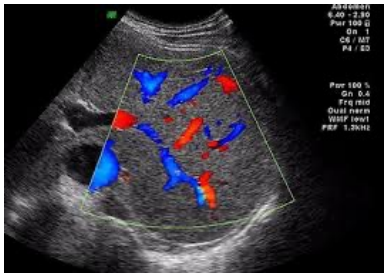


Image optique

Imagerie par résonance magnétique (IRM)

Échographie

Programmation

- Algorithme

Exemple : comparer deux variables

```
Algorithme Comparaison
  varaible x, y, max;
  debut
   saisir (x);
   saisir (y);
   si (x>y) alors
      max =x;
   sinon
      max=y;
  fin
```

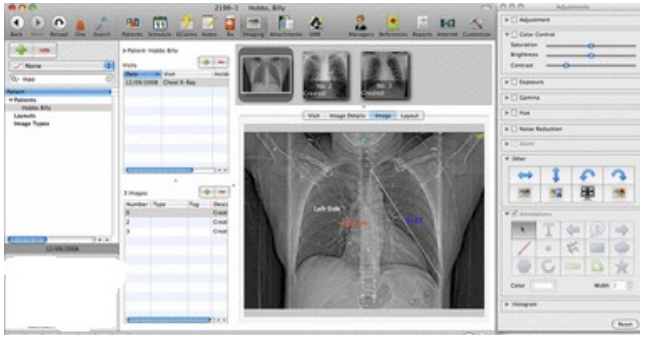
https://www.paris-friendly.fr/atelier-gratuit-de-programmation.html

Programmation

- Algorithme

Exemple : comparer deux variables

```
Algorithme Comparaison
  varaible x, y, max;
  debut
   saisir (x);
   saisir (y);
   si (x>y) alors
      max =x;
   sinon
      max=y;
  fin
```



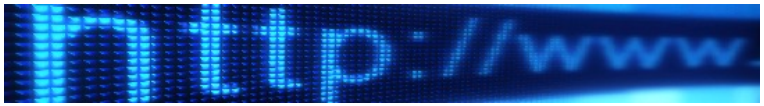
- En java ou en C

```
float  x=5, y=10,  max=0;

if (x>y) {
       max=x;
       System.out.println ("x est plus grand que y");
       }
else {
     max=y;
     System.out.println ("y est plus grand que x ou égal à x");
}
```

Programmation

- Web

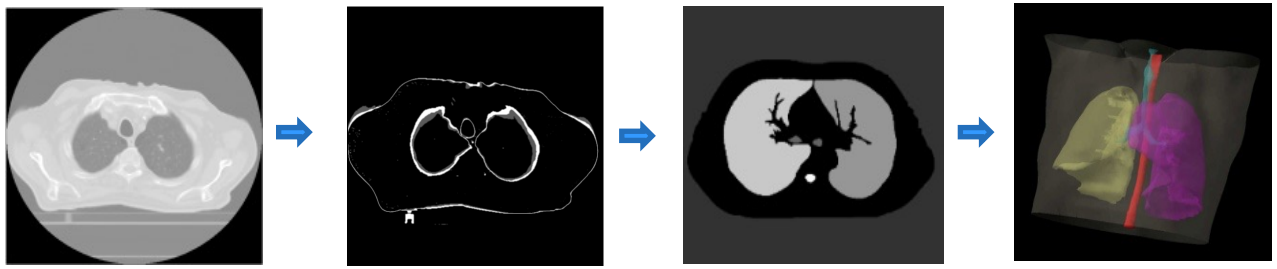


Traitement de l'information

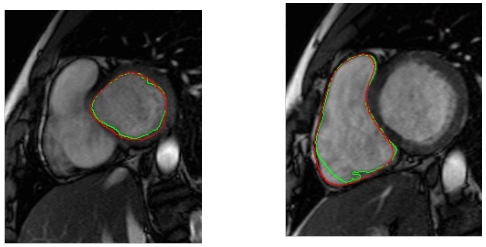
- Instrumentation biomédicale
- Traitement d'images
- Traitement du signal
- Traitement de grandes quantités de données (Big data, Bio-Informatique)
- Intelligence Artificielle en Biologie et Médecine

Traitement d'images médicales

- Segmentation d'image



Poumon



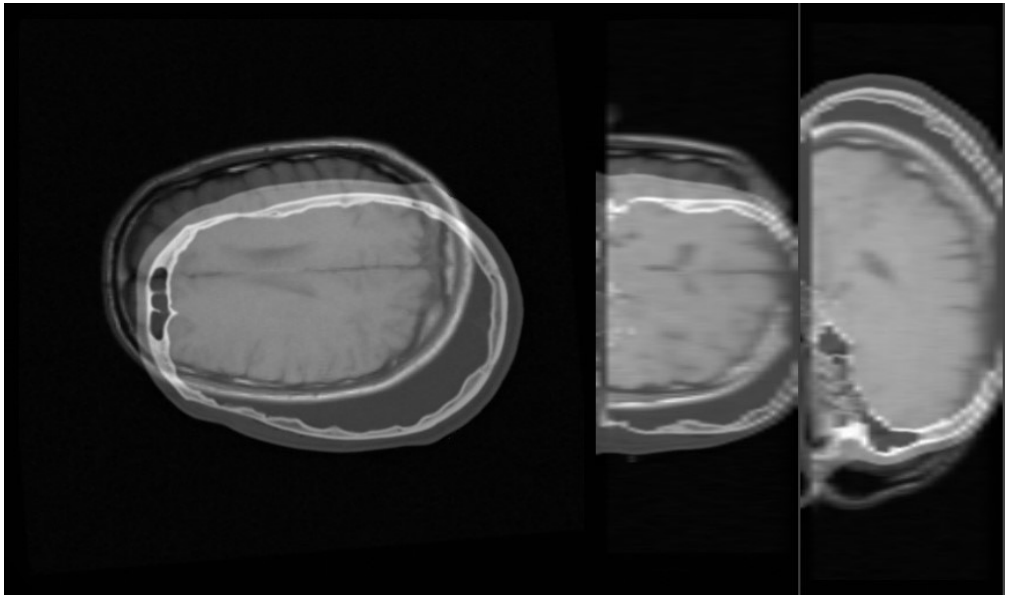
Cœur

Tumeur

Exemple: https://youtu.be/TWhyRUDKhuI

Traitement d'images médicales

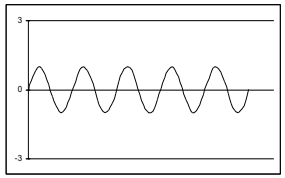
- Recalage des images



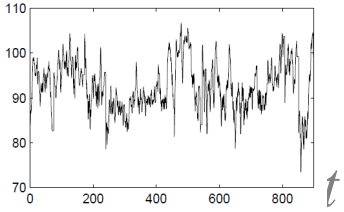
https://itk.org/Wiki/SimpleITK/Tutorials/MICCAI2015

Traitement des signaux médicaux

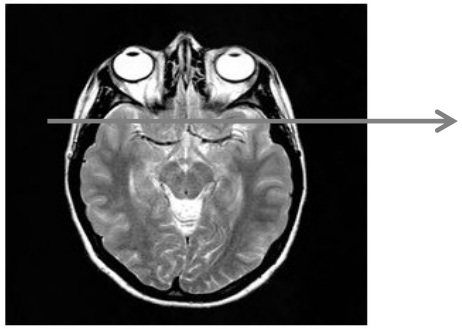
- Signaux

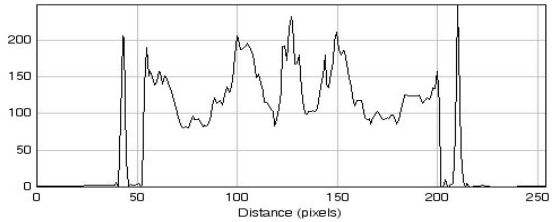


t

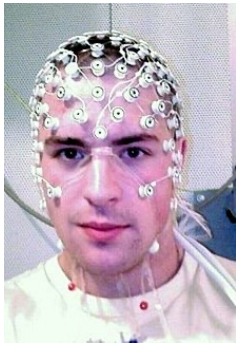


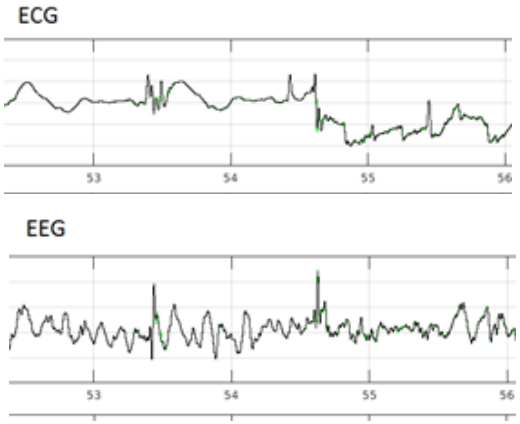
t





n



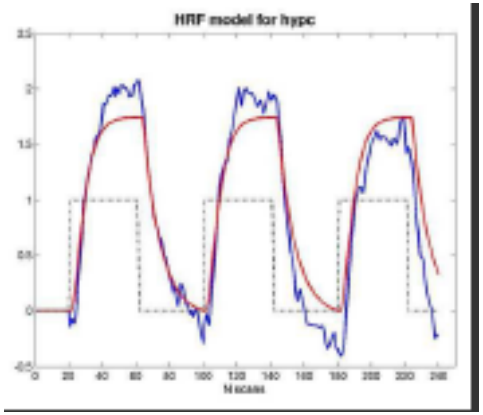


https://fr.wikipedia.org/wiki/%C3%89lectroenc%C3%A9phalographie

Traitement des signaux médicaux

- IRM fonctionnelle

  - Paradigme :
    - deux états
      - repos et activé



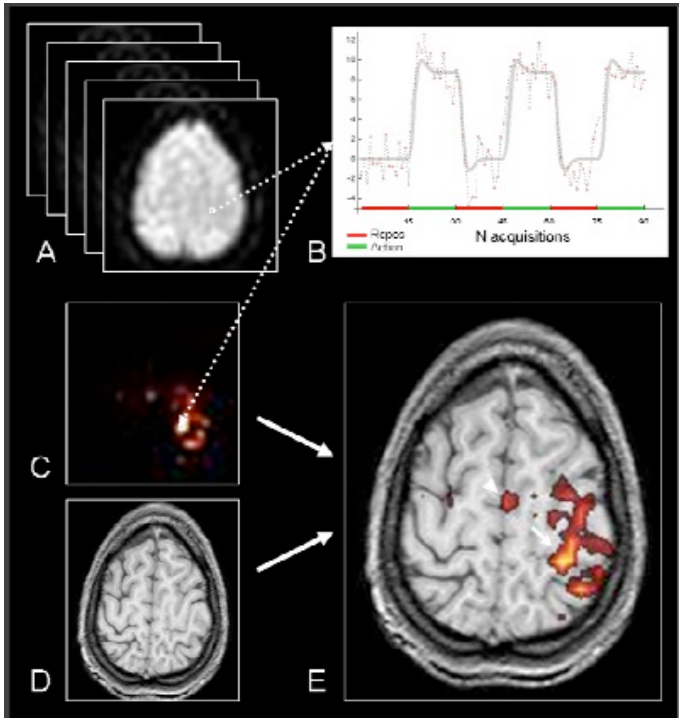


Image de A. Krainik, J. Warnking

Systèmes de numération

- Système décimal

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

- Système binaire

  0 et 1

- Système hexadécimal

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Systèmes de numération

- Système décimal

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

- Système binaire et Système octal

  0 et 1; 0-7

- Système hexadécimal

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- Conversion
  - un nombre exprimé en décimal peut se décomposer en base ***b***

$$(A)_{10} = \sum_{i=0}^{n} a_i b^i = a_n \times b^n + a_{n-1} b^{n-1} + \cdots + a_1 \times b^1 + a_0 \times b^0$$

  - la conversion d'un nombre exprimé en base ***b*** en décimal

$$(a_n a_{n-1} \cdots a_1 a_0)_b = \sum_{i=0}^{n} a_i b^i$$

Systèmes de numération

- Exemples

$(524)_8 = (\ )_{10} = 5x8^2+2x8+4 = 320+16+4 = 340$

Poids fort Poids faible

$(35)_{10} = (\ )_2$

$(35)_{10} = (\ )_8$

$(35)_{10} = (\ )_{16}$

Codage de l'information

- Coder l'ensemble d'informations ***I*** à l'aide de A = {a_1,a_2,… a_n} consiste à faire correspondre à chaque élément de ***I*** un mot construit sur A.

- Codage binaire naturel $(a_n a_{n-1} \cdots a_1 a_0)_2 = \sum_{i=0}^{n} a_i 2^i$
  - Le codage utilisé de façon native par les ordinateurs.
    b = 2, A = {0, 1}.
    Exemple: un codage de longueur 3 permet de coder les entiers de 0 à 7
  - Un mot binaire de longueur 8 est appelé **octet**

entier représenté	a_2	a_1	a_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

- Exemple:

$(1101101001001)2 = (0001\ 1011\ 0100\ 1001)_2 = (1b49)_{16}$

Codage de l'information

- Codage hexadécimal $(a_n a_{n-1} \cdots a_1 a_0)_{16} = \sum_{i=0}^{n} a_i 16^i$
  - une représentation plus compacte des nombres

    b = 16, A={0,1,… 9, A,B,C,D,F)

    lettres de A à F valant respectivement de 10 à 15

mot de 4 bits	chiffre hexadécimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	a
1011	b
1100	c
1101	d
1110	e
1111	f

entier représenté	code
16	10
17	11
18	12
...	...
25	19
26	1a
27	1b
...	...
31	1f
32	20
33	21
...	...
100	64
...	...
256	100
...	...

Codage de l'information

- Décimal codé binaire (DCB)
  - un code binaire dédié à la représentation des nombres exprimé en décimal.
  - Les chiffres du codage décimal sont les chiffres de 0 à 9. Ces chiffres peuvent être représentés par un mots de 4 bits allant de 0000 à 1001.
  - Les combinaisons allant de 1010 à 1111 ne sont pas utilisées dans le codage DCB
- Code de Gray
  - Le code de Gray est un codage construit de telle sorte que la représentation de deux entiers consécutifs ne diffère que par un seul bit.

Codage de l'information

- Exemples

codage DCB sur 8 bits

Entier décimal	code DCB sur 8 bits
0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0011
⋮	⋮
9	0000 1001
10	0001 0000
11	0001 0001
⋮	⋮
19	0000 1001
20	0010 0000
21	0010 0001
⋮	⋮
99	1001 1001

Code Gray sur 4 bits

Entier décimal	code de Gray sur 4 bits
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1100
13	1101
14	1001
15	1000

Codage de l'information

- Représentation des entiers relatifs
  - Bit de signe
    - Dans ce codage sur n bits, on utilise 1 bit pour coder le signe de l'entier relatif et n-1 bits pour coder la valeur absolue.
    - Le bit de signe vaut 0, l'entier représenté est positif; le bit de signe vaut 1, l'entier représenté est négatif.
    - Exemple: - 5 -> 1101; 5 -> 0101
  - Convention du complément à deux
    - le codage de la valeur absolue est différent selon que le nombre est positif ou négatif.
      - Positif: bit de signe = 0, la valeur absolue en binaire naturel
      - Négatif: on inverse tous les bits, puis ajoute 1 au dernier bit
      - Exemple: 5-> 0101, -5 ->1011;
    - n bits permettent de coder les entiers relatifs de -2^{n-1} à $2^{n-1}-1$.
      - Exemple: n=4, entiers relatifs : -8,-7……6,7

Codage de l'information

- Exemples:
  - Selon la convention du complément à deux,
    1) représenter les nombres:
       69, -69
    2) quels entiers représentent :
       01010100, 10101101

Codage de l'information

- Représentation des nombres réels
  - Représentation en virgule fixe
    - une partie du codage est dévolue à la partie entière du nombre (avec éventuellement utilisation du bit de signe) et l'autre à sa partie décimale.

$$(a_n, a_{n-1}, , , , a_0, a_{-1}, \ldots a_{-m})_2 \Rightarrow \sum_{i=-m}^{n} a_i 2^i$$

$$(00101011) = 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$$

    - Pour le codage d'un réel, on code d'une part en binaire naturel sa partie entière puis sa partie décimale. Pour sa partie décimale, cela se fait en multipliant par 2 de façon successive la partie décimale du nombre et en inscrivant dans le codage la partie entière du nombre obtenu.
    - Exemple: 2,69, selon le codage par virgule fixe avec 2 bits pour la partie entière et 6 bits pour la partie décimale.

      Ex: 0,69 se code 101100 sur 6 bits, le nombre 2,69 se code alors 10101100

      -> Un des problèmes majeurs de ce codage est la précision.

Codage de l'information

- Représentation des nombres réels
  - Représentation en virgule flottant
    - Soit X un réel à coder. Pour une base **b** donnée, il est possible de trouver une infinité de couples (m; e) tels que :

$$X = m' \times b^e$$

avec ***e*** entier relatif. ***m'*** est appelée mantisse et ***e*** est appelé exposant.

  - Norme IEEE754
    - Il est possible de coder les réels sur 32 bits en simple précision et sur 64 bits en double précision.
    - La valeur absolue de ***m'*** est comprise entre 1 inclus et 2 exclus.

$$X = (-1)^s (1 + m) \times 2^{e - offset}$$

    - Les réels en simple précision sont codés sur **32** bits. Le premiers bit code ***s*** (bit de signe). ***m*** est codé sur 23 bits dont les poids vont de -1 à -23. **e** est codé sur 8 bits et offset vaut 127 (pour que l'exposant soit toujours positif).

Codage de l'information

- Représentation des nombres réels
  - Exemple
    - Soit le mot de 32 bits suivant à décoder selon la norme IEEE754 :

1100 0001 1010 1100 0000 0000 0000 0000

s e m

s : 1 (le nombre codé est négatif)

e : $(10000011)_2 = (131)_{10}$

m : 0101 1000 0000 0000 0000 000

$$\begin{aligned} m &= 2^{-2} + 2^{-4} + 2^{-5} \\ &= 0,25 + 0,0625 + 0,03125 \\ &= 0,34375 \end{aligned}$$

Le réel codé vaut donc: $(-1)^1(1 + 0,34375) \times 2^{131-127} = -21,5$

- Soit le réel 51,7 à coder en simple précision selon la norme IEEE754.

Codage de l'information

- Codage des informations alphanumériques
  - ASCII (American Standard Code for Information Interchange)

    Un mot de 7 bits (128 combinaisons)
  - Codage unicode
    - compatible ASCII
    - code les caractères sur 16 bits et donc la possibilité de coder 65536 caractères

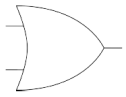
Poids fort (octal) Poids faible	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	‘	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	TAB	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Fonctions logiques

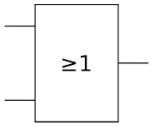
- Fonctions binaires et algèbre de Boole
  - Un système binaire est un système qui ne peut exister que dans deux états.
    - circuit ouvert ou fermé
    - tension au niveau haut ou au niveau bas
  - Logique : vrai et faux; oui et non
    - numérique : 0 et 1

- Fonctions logiques de base

La fonction OU: y=a+b



Symbole de la porte OU
(Norme anglo-saxonne)



Symbole de la porte OU
(Norme européenne)

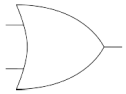
Table de vérité de la fonction OU

a	b	$y = a + b$
0	0	0
0	1	1
1	0	1
1	1	1

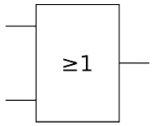
Fonctions logiques

- Fonctions binaires et algèbre de Boole
  - Un système binaire est un système qui ne peut exister que dans deux états.
    - circuit ouvert ou fermé
    - tension au niveau haut ou au niveau bas
  - Logique : vrai et faux; oui et non
    - numérique : 0 et 1

- Fonctions logiques de base
  - La fonction OU: y=a+b



Symbole de la porte OU
(Norme anglo-saxonne)



Symbole de la porte OU
(Norme européenne)

a + a = a

a + 0 = a

a + 1 = 1

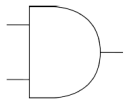
Table de vérité : écrire de façon explicite la valeur prise par la fonction pour chaque combinaison possible de ses variables

Table de vérité de la fonction OU

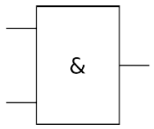
a	b	$y = a + b$
0	0	0
0	1	1
1	0	1
1	1	1

Fonctions logiques

- Fonctions logiques de base
  - La fonction ET: y= a·b=ab
    - L'opérateur ET est prioritaire sur l'opérateur OU.



Symbole de la porte ET
(Norme anglo-saxonne)



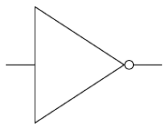
Symbole de la porte ET
(Norme européenne)

a·a=a
a·0=0
a·1=a

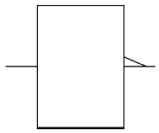
Table de vérité de la fonction ET

a	b	$y = a.b$
0	0	0
0	1	0
1	0	0
1	1	1

  - La fonction NON: y= $\bar{a}$



Symbole de la porte NON
(Norme anglo-saxonne)



Symbole de la porte NON
(Norme européenne)

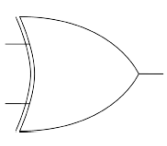
$a + \bar{a} = 1$
$a \cdot \bar{a} = 0$
$a + \bar{a}$·b=a+b

Table de vérité de la fonction NON

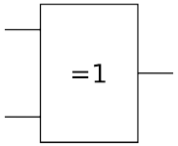
a	$\bar{a}$
0	1
1	0

Fonctions logiques

- Fonctions logiques de base
  - La fonction OU EXCLUSIF (XOR) : y= $a \oplus b = \bar{a}\, b + a\bar{b}$
    - L'opérateur ET est prioritaire sur l'opérateur OU.



Symbole de la porte ET
(Norme anglo-saxonne)



Symbole de la porte ET
(Norme européenne)

$$a \oplus a = 0$$
$$a \oplus \bar{a} = 1$$
$$a \oplus 1 = \bar{a}$$
$$a \oplus 0 = a$$

Table de vérité de la fonction XOR

a	b	$y = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

- Théorèmes de De Morgan

$$\overline{a + b + c + \cdots} = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \ldots$$

$$\overline{a \cdot b \cdot c \cdot \cdots} = \bar{a} + \bar{b} + \bar{c} \cdot \ldots$$

Fonctions logiques

- Autres fonctions logiques
  - La fonction NON ET (NAND): y= $\overline{ab}$

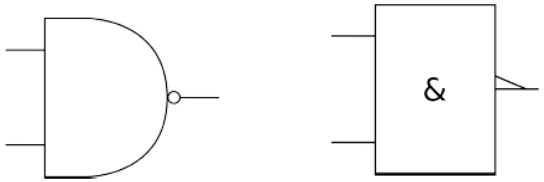


Table de vérité de la fonction NAND

a	b	$y = \overline{a.b}$
0	0	1
0	1	1
1	0	1
1	1	0

  - La fonction NON OU (NOR): y= $\overline{a+b}$

Table de vérité de la fonction NOR

a	b	$y = \overline{a+b}$
0	0	1
0	1	0
1	0	0
1	1	0

Fonctions logiques

- Écritures des fonctions logiques
  - Somme canonique de produits

    L'écriture algébrique en somme canonique de produits peut être obtenue depuis la table de vérité de la fonction en repérant les combinaisons d'entrées pour lesquelles la fonction vaut 1.

Table de vérité

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$f = \overline{a}\overline{b}\overline{c} + \overline{a}bc + a\overline{b}c + ab\overline{c}$$

  - Produit canonique de sommes

    À chaque combinaison des variables pour lesquelles la fonction vaut zéro correspond un facteur du produit. Ce facteur est composé de la somme des variables, complémentées si elles valent 1.

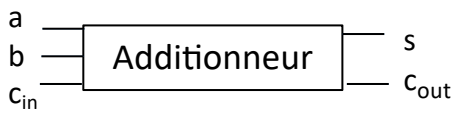
$$f = (a + b + \overline{c}).(a + \overline{b} + c).(\overline{a} + b + c).(\overline{a} + \overline{b} + \overline{c})$$

Logique combinatoire

- Addition binaire
  - Il y a deux sorties: $s=a+b+c_{in}$; c_{out}: *retenue*

Table de vérité

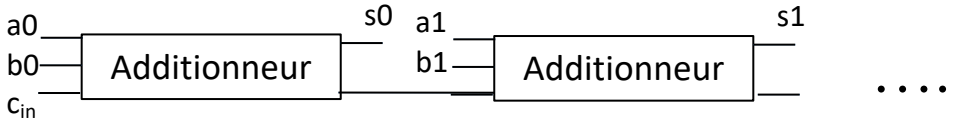
a	b	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$
\begin{aligned}
s &= a \oplus b \oplus c_{in} \\
c_{out} &= ab + ac_{in} + bc_{in} \\
&= ab + c_{in}(a+b)
\end{aligned}
$$

  - Addition parallèle

    Pour effectuer l'addition n bits, on doit disposer de n additionneurs complets. L'étage i doit disposer en entrée de la retenue produite par l'étage i-1. De ce fait, le résultat est disponible après n fois le temps de transition d'un additionneur.



Logique combinatoire

- Soustraction
  - Additionneur-soustracteur

    la soustraction B -A peut être effectuée en sommant B au complément à 2 de A.

$$B - A = B + \overline{A} + 1$$

  - Matériellement, la soustraction est effectuée en faisant l'addition B + $\bar{A}$ avec une retenue entrante valant 1 sur le premier étage de l'additionneur.

Exercices

- Codage binaire naturel

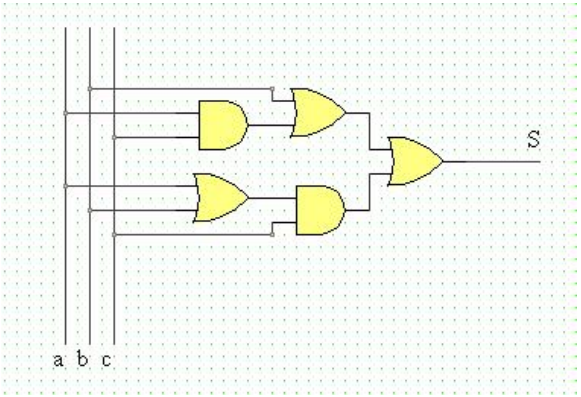
1. Coder en binaire naturel le nombre 193.
2. Le nombre précédent est-il codable sur un octet ? Quelle est donc la valeur décimale maximale codable sur un octet ?
3. Toute machine sur Internet est identifiée par une adresse appelée adresse IP représentée sur 4 octets. 193.52.145.35 est par exemple l'adresse IP attribuée par l'administrateur réseau à l'un des serveur de l'Université. Les machines entre elles ne dialoguent pas en décimal mais en binaire. Quelle est l'adresse binaire du serveur précédent ?
4. Quelle est l'écriture hexadécimale de l'adresse IP précédente ?

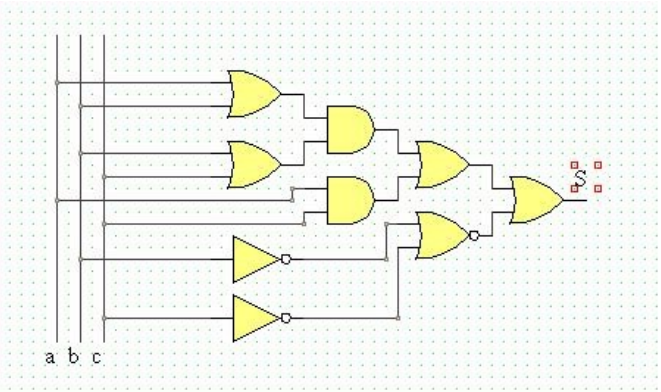
- Codage des entiers relatifs en complément à 2

1. Faire la somme 72 + (-86) sur les représentations en complément à 2. Donner la valeur décimale du résultat.
2. On veut maintenant coder les entiers sur deux octets ; que deviennent les codes de 72 et -86 ? Conclusion ?
3. On sait que $2^{10} = 1024$ donc proche de 1000 ; d'où le terme de kiloOctets pour 1024 octets. Combien de valeurs différentes pourra-t-on coder avec des mots de 32 bits ? (réponse approximative en considérant que 1024 est proche de 1000).

Exercices

1. Donnez pour chacun des 2 circuits précédents l'équation correspondant au câblage.
2. En supposant que le temps de transition d'une porte vaut T, quel sera le temps nécessaire pour obtenir une sortie valide sur chacun de ces deux circuits.
3. En établissant la table de vérité de chacun des deux circuits, montrez que ces deux circuits réalisent la même fonction.





Référence

- Cours de Pierre Heroux , https://universitice.univ-rouen.fr/