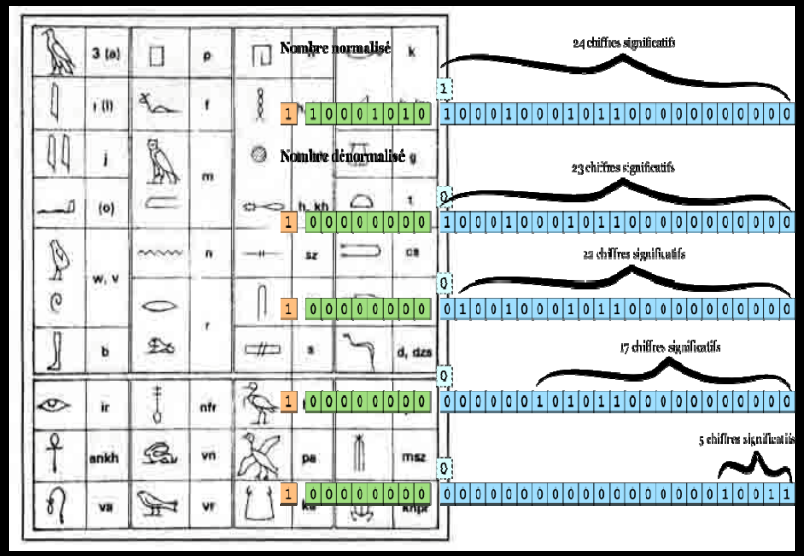
Algorithmes numériques

Chapitre 1 : codage



Stéphane Gobron
HE-Arc // hes.so

haute école
neuchâtel berne jura
arc ingénierie
saint-imier le locle delémont

2017

Algorithmes Numériques

Chapitre 1

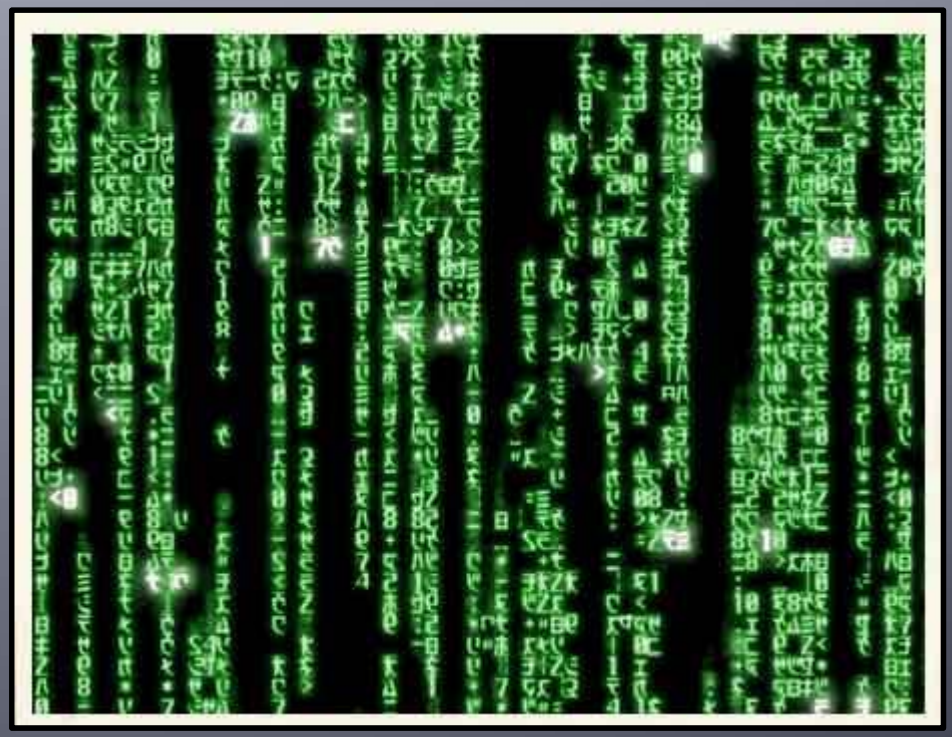
- ➔ **Codage des nombres**
- Résolution d'équations
- Systèmes linéaires
- Dérivation
- Intégration
- Equation différentielles
- Optimisation

Nuage de mots de ce chapitre

Super hero présenté dans ce chapitre !

Chapitre 1
Codage des nombres

- Notion de codes
- Problématique
- Modèle mathématique
- Valeurs entières
  - Codage
  - Arithmétique
- Valeurs réelles
  - Stockage
  - Arithmétique
  - Précision
- Labo 1



Représentation du codage de la Matrice

Algorithmes Numériques

Codage des nombres

Notion de codes

- Exemples de codage?
- Qu'est-ce qu'un codage?

En informatique

- Que code-t-on?

_____

- Pourquoi coder?

_____

- Peut-on tout coder?

_____

- Y-a-t-il un problème d'optimisation?

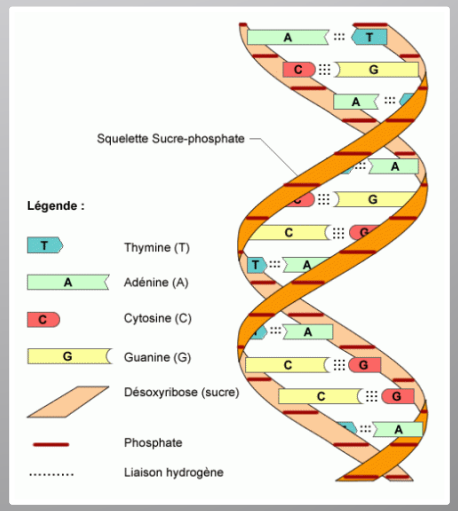
_____

Code postal	Codage en bâtonnets
51100	|||| |||| | ||| | |||| | ||
52130	|||| ||| | | ||| || ||| | ||
08400	|||| ||||| ||||| | | ||||
75006	| || | |||| ||||| | |||| ||
13007	|| || |||| |||| ||| | | |||
	|||| | ||| ||| || |||||| |

Codage postal

Ancien perse

Codes civil… code morale…?



ADN

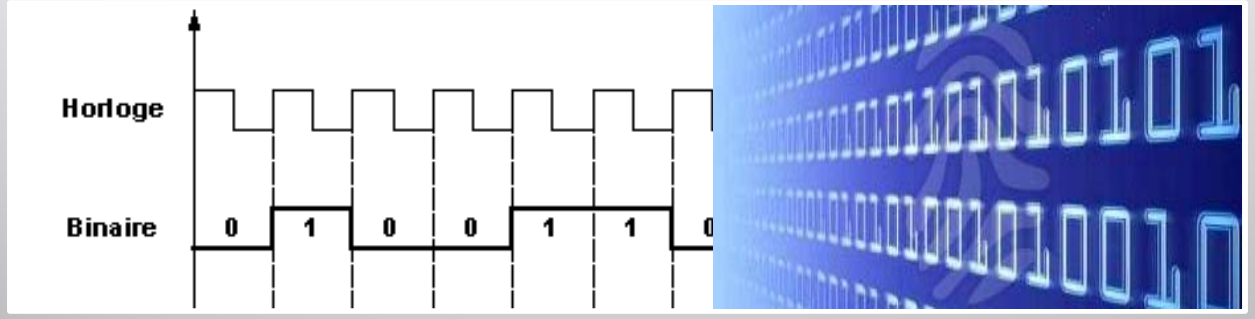
*Un **codage** permet de passer d'une représentation des données vers une autre*

Algorithmes Numériques

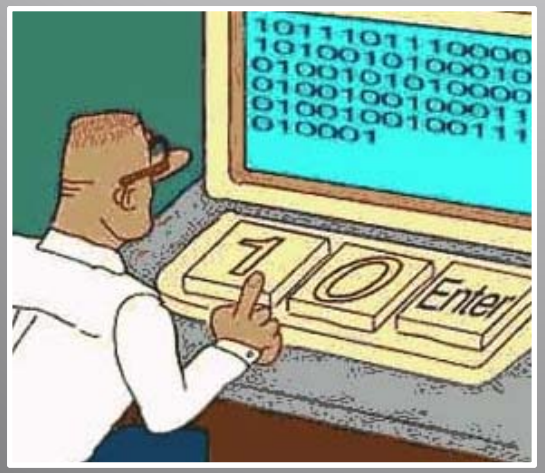
Codage des nombres

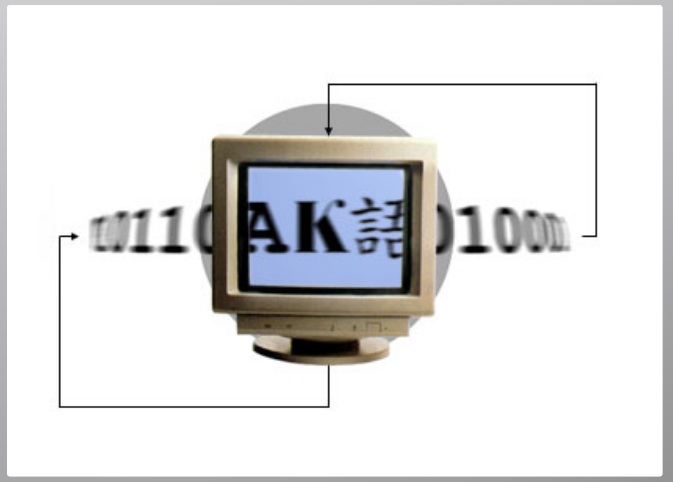
Problématique

- Comment communiquer avec quelque chose qui ne «comprend» que des zéros et des uns?
- Capacité mémoire d'un ordinateur par contrainte finie



Beaucoup, vraiment beaucoup de zéros et de uns



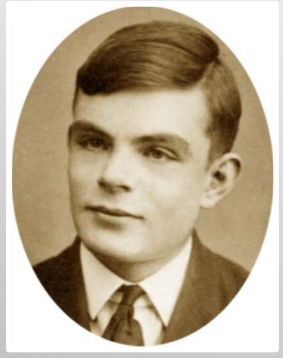


Algorithmes Numériques

Codage des nombres

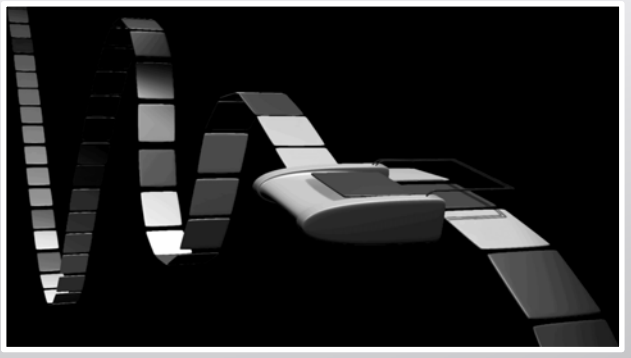
Un mot d'histoire

- Un des pères –sinon *le* père— fondateur de l'informatique
- Réussite majeure : briser le code de la machine Enigma des Nazis
- Condamné pour homosexualité puis gracié par la reine mais 60 ans après sa mort
- Mort probablement accidentellement de ses propres expér. ou en croquant une pomme empoisonnée au cyanure
- Logo d'Apple? Il n'en est rien en réalité !



Alan Turing
1912~1954

→ Définit les concepts de **programme** et de **programmation**



Vue d'artiste d'une Machine de Turing sans la table de transition

→ modèle abstrait du fonctionnement des appareils mécaniques de calcul, tel un ordinateur et sa mémoire

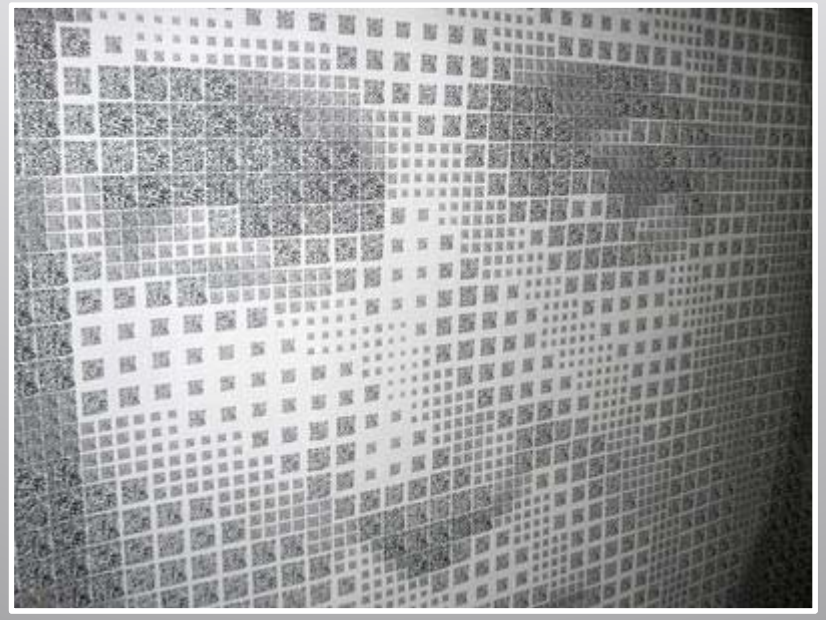
Ce concept permettant de tendre à une définition précise au concept d'algorithme est toujours largement utilisé en informatique théorique, en particulier dans les domaines de la **complexité** algorithmique et de la **calculabilité**.

Algorithmes Numériques

Codage des nombres

Modèle mathématique

- Changement de base
- Discrétisation
- Il n'y a pas de modèle mathématique puisque nous traitons un problème purement informatique! ☺



Art et discrétisation – code barres

Design informatique

Valeurs entières

- 1 octet = 8 bits

- Anglais

➔ « Octet » ➔ « byte »

➔ « Bit » ➔ « bit »

Entier sur 1 octet

-127	11111111
-1	10000001
-0	10000000
$+0$	00000000
$+1$	00000001
$+127$	01111111

avec signe et valeur absolue

-128	10000000
-1	11111111
± 0	00000000
$+1$	00000001
$+127$	01111111

avec signe et complément

Entier sur 2 octets

	Adr	$Adr+1$
-32768	00000000	10000000
-1	11111111	11111111
± 0	00000000	00000000
$+1$	00000001	00000000
$+32767$	11111111	01111111

Grand à la fin

	Adr	$Adr+1$
-32768	10000000	00000000
-1	11111111	11111111
± 0	00000000	00000000
$+1$	00000000	00000001
$+32767$	01111111	11111111

Petit à la fin

Codages possibles

- Pour le C et C++
- Tableau récapitulatif
- Attention : pour les labos, qui seront à réaliser en HTML, tout est *float*!

Type de donnée	Signification	Taille (en octets)	Plage de valeurs acceptée
char	Caractère	1	-128 à 127
unsigned char	Caractère non signé	1	0 à 255
short int	Entier court	2	-32 768 à 32 767
unsigned short int	Entier court non signé	2	0 à 65 535
int	Entier	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	-32 768 à 32 767 -2 147 483 648 à 2 147 483 647
unsigned int	Entier non signé	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	0 à 65 535 0 à 4 294 967 295
long int	Entier long	4	-2 147 483 648 à 2 147 483 647
unsigned long int	Entier long non signé	4	0 à 4 294 967 295
float	Flottant (réel)	4	$-3.4*10^{-38}$ à $3.4*10^{38}$
double	Flottant double	8	$-1.7*10^{-308}$ à $1.7*10^{308}$
long double	Flottant double long	10	$-3.4*10^{-4932}$ à $3.4*10^{4932}$

Design informatique

Ce qui est « vrai »

- Arithmétique
- Très simple en passant par le codage Booléen
- Mais attention aux calculs avec des « int »

En informatique,
avec un codage sur des entiers :

7 / 2 = 3

et

(7 / 2) x 2= 6

Choix des équipes

- Vous avez quelques minutes pour former des équipes de 3 ou 4 étudiants

Indices!

- Regroupez-vous avant tout pour réaliser une équipe homogène

- **Trois critères**

- Éloignement
- Affinité
- Complémentarité

Algorithmes Numériques
Codage des nombres

Développement info.

Codage et décodage binaire

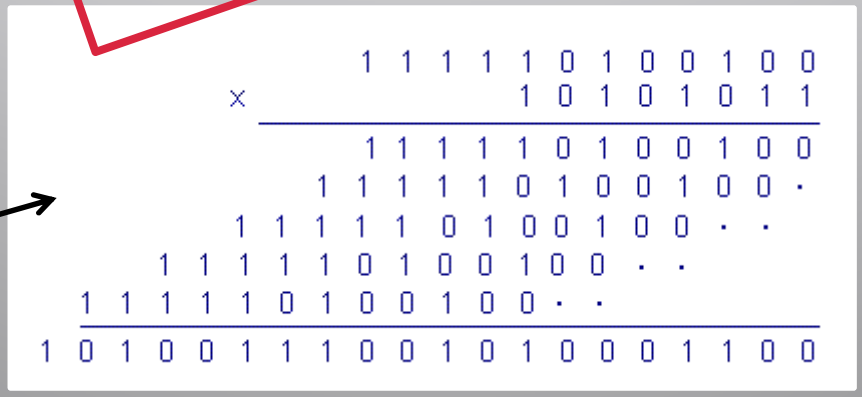
- Objectif principal : Développer un programme HTML 5, donc JS, permettant le codage et le décodage entier ⇔ binaire sur *n* bits, avec *n* valeur entrée par l'utilisateur

- Objectif(s) secondaire(s) :

→ Addition
→ Soustraction
→ Multiplication

/!\ Ce travail ne sera pas noté mais forme la base fondamentale du labo 1

Pré-labo en vue du labo1



Fin de la 1ère partie du chapitre 1

Merci!

Partie 2 du chapitre 1:

Codage « flottant »

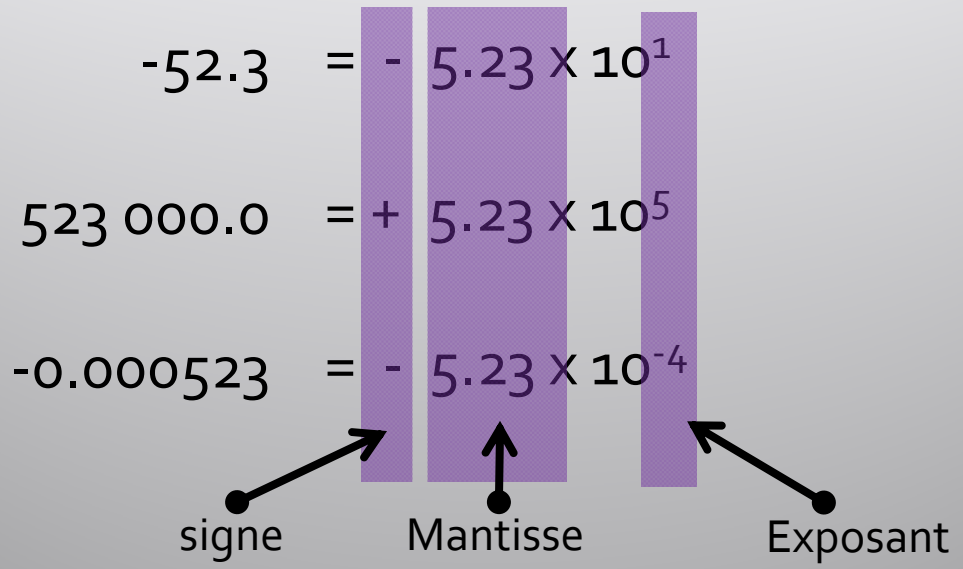
```
.titanic {
  float: none;
}
```

Interprétation négationniste du concept

Problématique

Pb de la «,»

- Comment coder un nombre à virgule avec des '0' et des '1'?
- Le signe reste le 1er bit
- Pour la virgule, on code le chiffre sous la forme de puissance! ☺
- **Idée principale** : pour tout nombre, réécrivons le en produit d'une puissance avec un nombre compris entre 1 et 10



Mais attention nous sommes ici en unité décimale, il faut tout réinterpréter en binaire donc en puissance de 2!

Problématique

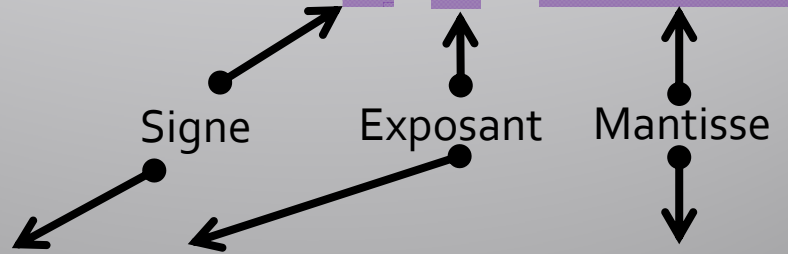
Interprétation en base 2

- Tout nombre peut aussi être interprété sous la forme du produit d'une puissance de 2 et d'un nombre entre 1 et 2

$-52.3 = -2^a \times x = -2^5 \times 1.634375$ (A)

$523\,000.0 = +2^b \times y = +2^{18} \times 1.995086\ldots$ (B)

$-0.000523 = -2^c \times z = -2^{-11} \times 1.071104$ (C)



$A_b = 1 \mid 10000100 \mid 10100010010011001100110$

$B_b = 0 \mid 10010010 \mid 11111010111100000000000$

$C_b = 1 \mid 01110100 \mid 00010010001100111011111$

Valeurs réelles et signe

- Pour toute valeur réelle codée « x »
- « s » est le bit de signe
- Pour trouver « S » avec le bit de signe « s » :
  si s=1 => négatif donc $S=-1$
  si s=0 => positif donc $S=1$
- Ou encore $S=-2s+1$
- *e': valeur en entier du code binaire « e »*

$$x = S.M.2^{e'-d}$$

Formule du codage d'un nombre réel

bit de signe → $s \neq S$ ← signe

$m \neq M$

Mantisse de bits

Valeur logiquement entre 0.0 et 1.0 **mais qui est en fait** entre 0.5 et 1.0

Mantisse et exposant

- « m » la **mantisse** de bits est faite tel que
- $M = m/2^n$, *i.e.* une valeur $0.0 < M < 1.0$

⇒ Avec *n* le nb de bits pour la mantisse

- Si « M » ∈ [0.5, 1.0[, on dit que la mantisse est normalisée
- « e'-d » est l'exposant dont seule la valeur « e' » est stockée en binaire avec *e*

⇒ d pour « décalage » aussi appelé « biais », fixé $2^{E-1}-1$ avec *E* nb de bits pour *e*

Exemple pour un codage sur 32 bits

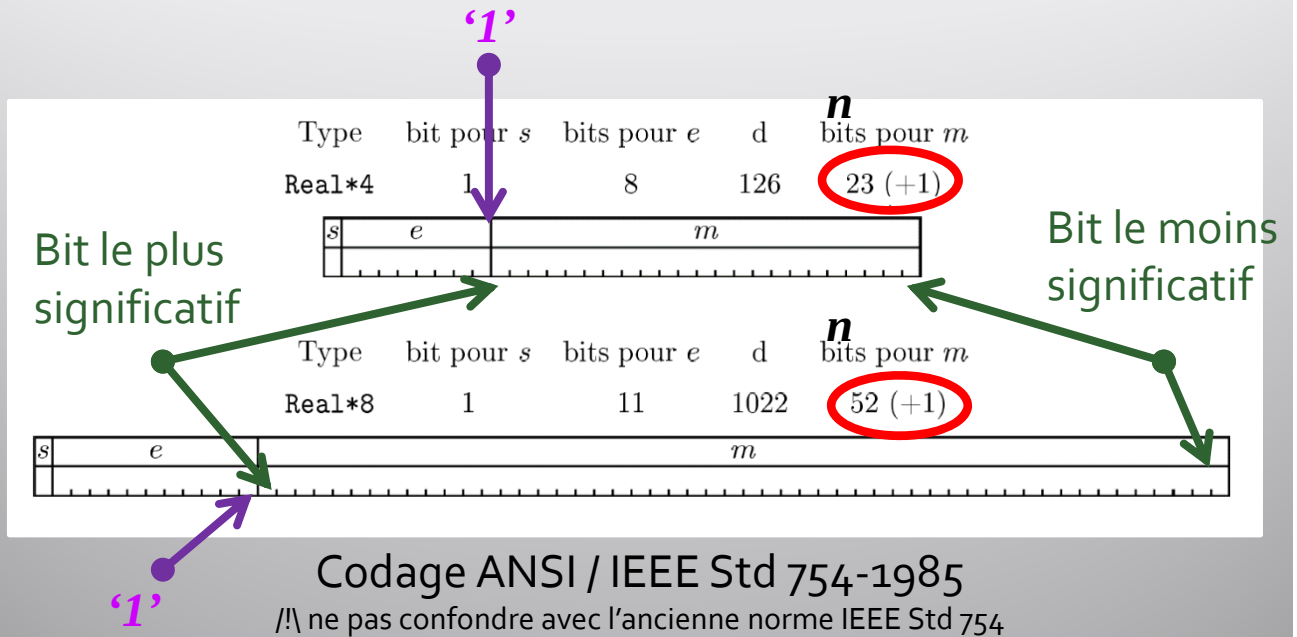
Pour 32 bits, E=8 => $d = 127$
Pour 64 bits, E=11 => $d = 1023$

Codage ANSI / IEEE Std 754-1985
/!\ ne pas confondre avec l'ancienne norme IEEE Std 754

http://fr.wikipedia.org/wiki/IEEE_754

Valeurs réelles

- Bits pour 32 et 64 bits



Codage ANSI / IEEE Std 754-1985

/!\ ne pas confondre avec l'ancienne norme IEEE Std 754

- Attention au « (+1) »

La mention (+1) pour le nombre de bits réservés à la mantisse m appelle quelques explications :

Si le nombre est normalisé, sa mantisse est dans l'intervalle [0.5 ; 1.0 [, son chiffre le plus significatif est obligatoirement un 1. On peut omettre de le mémoriser : c'est ce qu'on appelle le *bit caché*. Grâce à cette astuce, un champ de 23 ou 52 bits permet de stocker une mantisse dont on connaît 24 ou 53 bits.

n n

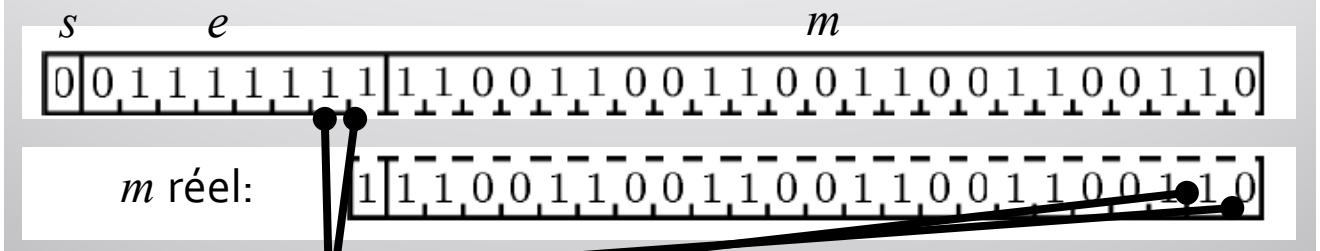
Réf.: *J-M. Blanc, "Analyse et Simulation, Chap.1, Comment un processeur calcule…« , Fribourg*

Design informatique

Valeurs réelles

- Exemple de calcul pour retrouver une valeur réelle
- Ici on cherche à trouver ce que l'ordinateur a codé pour le calcul «9 divisé par 5»

32 bits => $d=127$



$$M = (0^0+2^1+2^2+0^3+0^4+2^5+2^6+0^7+0^8+2^9+2^{10}+0^{11}+0^{12}+2^{13}+2^{14}$$
$$+0^{15}+0^{16}+2^{17}+2^{18}+0^{19}+0^{20}+2^{21}+2^{22}+2^{23})/\ 2^{23}$$
$$= (2+4+32+64+512+1'024+8'192+16'384+131'072$$
$$+262'144+2'097'152+4'194'304+8'388'608)\ /\ 2^{23}$$
$$= 15'099'494\ /\ 2^{23}$$

$$M \approx 1.799999952 \ldots \approx 1.8$$

Et pour $e' = 1+2+4+8+16+32+64 = 127$

$$x = (-2 \times 0+1) \times M \times 2^{e'-d}$$
$$= 1 \times 1.8 \times 2^{127-127} = 1.8$$

http://www.h-schmidt.net/FloatConverter/IEEE754.html

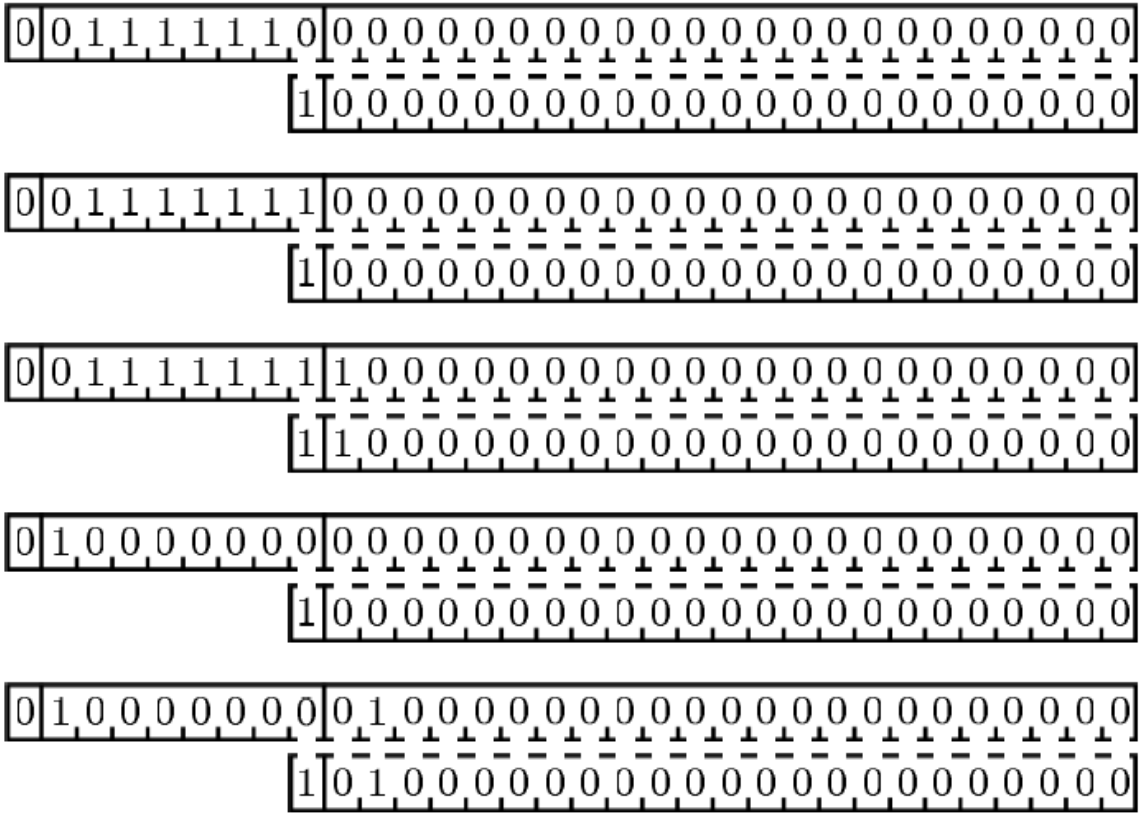
En résumé

nom	définition	exemple
x	*valeur en réel codée*	*1.8*
S	*signe*	*+ ou -*
s	*signe binaire*	*0' positif et '1' négatif*
M codé	*valeur en réel de la mantisse, toujours comprise entre dans l'intervalle [0.5, 1.0[*	*0.899999976*
M réel	*Et avec le bit caché de valeur '1' entre [1.0, 2.0[*	*1.799999952*
m	*valeur de la mantisse en binaire*	*11001100110011001100110*
e	*valeur de l'exposant en binaire*	*01111111*
e'	*valeur en entier de l'exposant*	*127*
E	*nb de bits pour l'exposant*	*8*
d	*décalage permettant le codage de nombre à virgule*	*127*
n	*nb de bits pour la mantisse*	*32-1-8=23*

Interaction

Petits exercices

- Que valent ces codes?



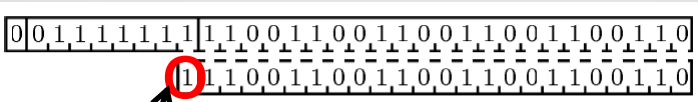
Valeurs réelles

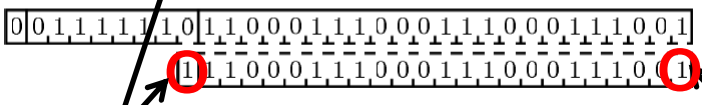
- Arithmétique
- L'addition
- Concept

➔ Pour toute variable « réelle flottante » x et y de mantisse et exposant $\{m_x, e_x\}$ et $\{m_y, e_y\}$

➔ Astuce : s'arranger pour que e_x et e_y soient les mêmes

Etape 0 : initialisation des *float* (32bits) x et y

x = 

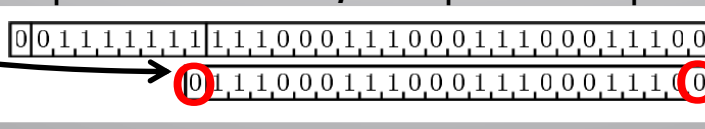
y = 

ici x = 9.f / 5.f

et y = 8.f / 9.f

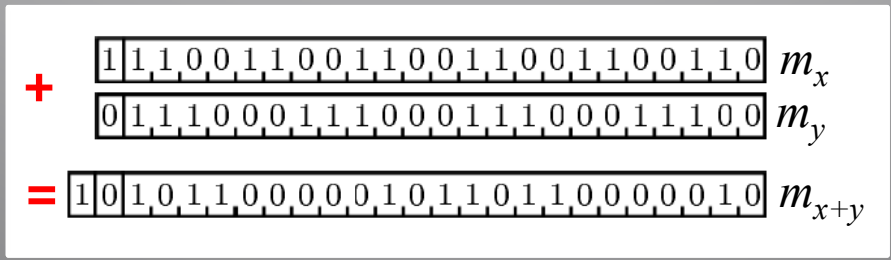
Notez bien que x et y sont normalisés

Etape 1 : décaler le plus petit exposant vers la droite
=> y n'est plus normalisé, et a perdu un peu de précision



Dans ce cas $e_y \leftarrow e_y+1$

Etape 2 : addition des mantisses

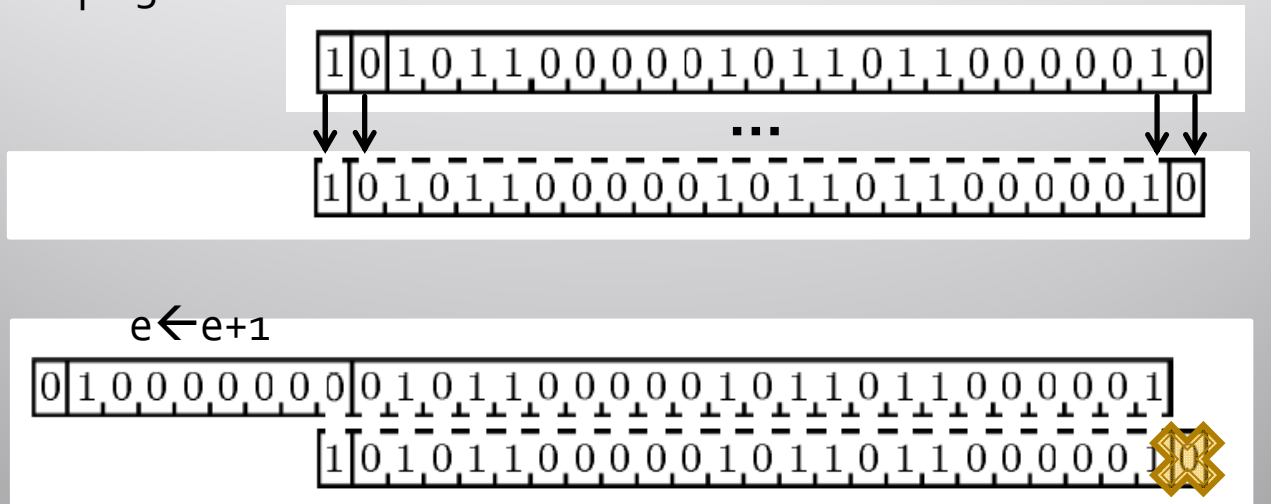


Design informatique

Valeurs réelles: l'addition

- Composition du nouveau codage

Etape 3 : normalisons la trame de bits



Etape 4 : élimination du dernier bit

En revanche, si le bit excédentaire à droite valait 1, le fait de le laisser tomber introduirait une erreur. La norme ANSI/IEEE Std 754-1985 règle le problème de l'arrondi de la manière suivante :

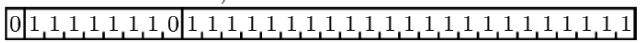
- Si le bit excédentaire vaut 0, on le laisse simplement tomber.
- Si le bit excédentaire vaut 1 et si le bit précédent vaut 0, on arrondit vers le bas : on laisse tomber le bit excédentaire.
- Si le bit excédentaire et le(s) précédent(s) valent 1, on arrondit vers le haut, avec les retenues qui s'imposent.

Réf.: *J-M. Blanc, "Analyse et Simulation, Chap.1, Comment un processeur calcule...« , Fribourg*

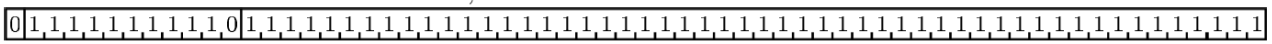
Valeurs réelles

- Précision
- La précision est toute relative
- Ainsi définir un «epsilon» peut se révéler difficile

$0,34028235\ 10^{+39}$

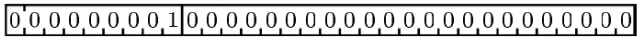


$0,1797693134862316\ 10^{+309}$

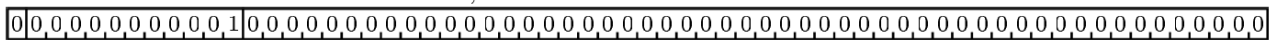


Les plus grands nombres finis à 32 et 64 bits

$0,11754944\ 10^{-37}$

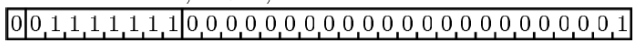


$0,2225073858507201\ 10^{-307}$

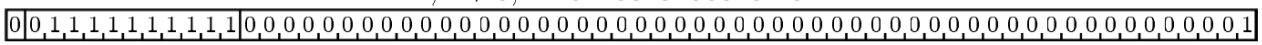


Les plus petits nombres finis à 32 et 64 bits

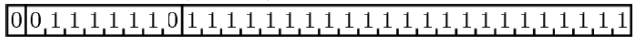
$1,0+0,11920929\ 10^{-6}$



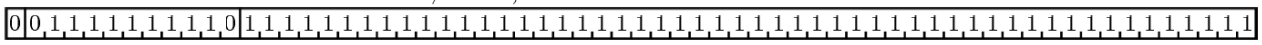
$1,0+0,2220446049250313\ 10^{-15}$



$1,0-0,59604645\ 10^{-7}$



$1,0-0,1110223024625157\ 10^{-15}$



Nombres les plus proches de 1 vers le haut et le bas

Algorithmes Numériques
Codage des nombres

Développement info.

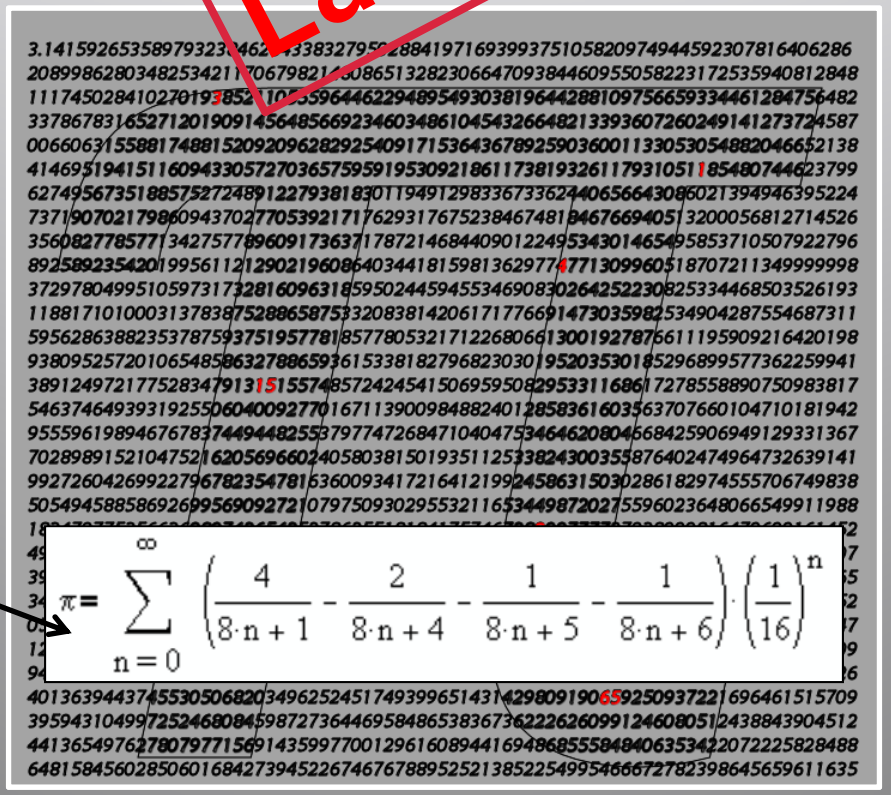
Réels sur n bits! ☺

- **Objectif principal**
  Même question que le l'avant projet pour un nombre flottant sur n bit
  /!\ La valeur de d est à trouver à la volée

- **Objectif(s) secondaire(s)**
  Implémentez l'addition

- **Super bonus** → mini projet à discuter avec STG : trouvez une approximation de π avec ce codage et cette formule

Labo 1



Réf. de chapitre

Merci! Questions?

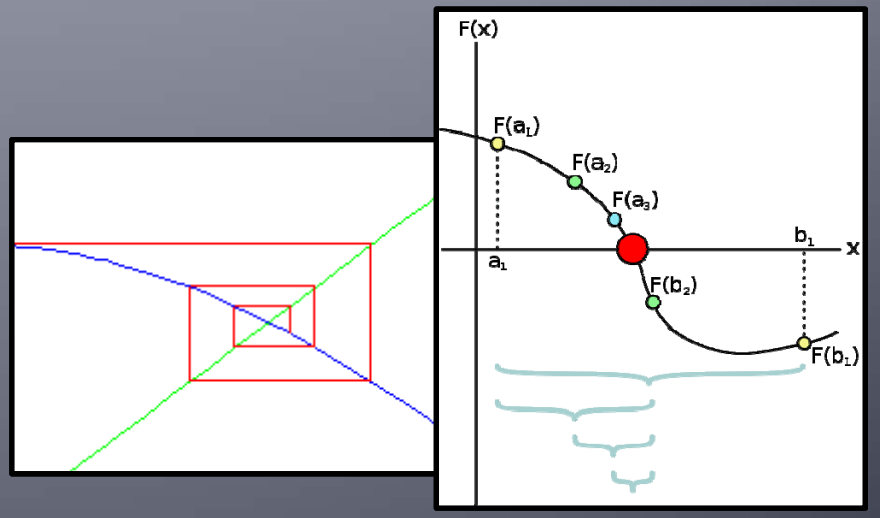
Références

- Principalement Wikipédia

Prochain cours

Résolution d'équation

- Problématique
- Modèle mathématique
- Méthode bissection
- Méthode de Newton
- Méthode du point fixe
- Labo 2



Résolution par point fixe et bissection