ensiie
école nationale supérieure d'informatique
pour l'industrie et l'entreprise

Numération-Codage

G.Berthelot

Plan

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- sytèmes de numération binaire et hexadécimal
- conversion
- codage
- complément à 1, complément à 2
- représentation des "réels"
- quantité d'information
- codage de Huffman

introduction

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- rappel : informatique = science du traitement automatisé de l'information.
- Qu'est ce que l'information : connaissance d'un choix parmi un ensemble de valeurs. Ces valeurs peuvent être représentée à l'aide de suites de symboles

- unité élémentaire d'information : le bit
  (un bit permet de faire un choix entre 2 valeurs possibles)

- kilobit 10^3 kibi bit $2^{10} = 1024$
- megabit 10^6 mebi bit $2^{20} = 1024^2$
- gigabit 10^9 gibi bit $2^{30} = 1024^3$
- terabit 10^{12} tibi bit $2^{40} = 1024^4$
- petabit 10^{15} pebi bit $2^{50} = 1024^5$
- exabit 10^{18} exbi bit $2^{60} = 1024^6$

nombres entiers

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- opérations principales
  - addition, soustraction
  - multiplication
  - division
  - modulo
- propriétés des opérateurs
  - division entière interne dans N
  - soustraction non interne dans N
  - soustraction interne dans Z
  - division réelle non interne
  - modulo non définie dans R
  - division par 0 non définie

nombres entiers

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Numération (représentation des entiers)
  - décimale
  - hexadécimale
  - octale
  - binaire

- représentation (formule indépendante de la base b):

$$...a_2a_1a_0,q_1q_2.... = \sum_{i=0}^{\infty} a_i b^i + \sum_{i=1}^{\infty} q_i b^{-i}$$

- Remarques : les règles de l'arithmétique sont indépendantes des bases, seules les tables changent.

nombres entiers

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Base 10
  - a,q ∈ {0,1,2,3,4,5,6,7,8,9}
  - b=10
  - arithmétique
    - table d'addition:

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0^1
2	2	3	4	5	6	7	8	9	0^1	1^1
.										
.										
8	8	9	0^1	1^1	2^1	3^1	4^1	5^1	6^1	7^1
9	9^1	0^1	1^1	2^1	3^1	4^1	5^1	6^1	7^1	8^1

    - + tables pour les autres opérations

nombres entiers

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Base 8 (octal)
  - a,q ∈ {0,1,2,3,4,5,6,7}
  - b=8
  - arithmétique
    - table d'addition:

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0^1
2	2	3	4	5	6	7	0^1	1^1
3	3	4	5	6	7	0^1	1^1	2^1
4	4	5	6	7	0^1	1^1	2^1	3^1
5	5	6	7	0^1	1^1	2^1	3^1	4^1
6	6	7	0^1	1^1	2^1	3^1	4^1	5^1
7	7	0^1	1^1	2^1	3^1	4^1	5^1	6^1

nombres entiers

- Base 16
  - a,q ∈ {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
  - b=16
  - table d'addition:

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

nombres entiers

- Base 2
  - a,q ∈ {0,1}
  - b=2
  - table d'addition:

+	0	1
0	0	1
1	1	0^1

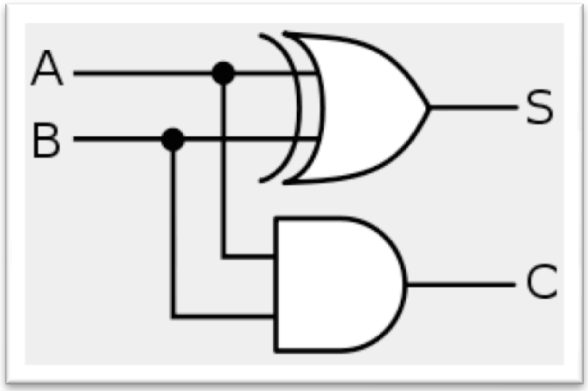
avec retenue entrante et sortante:

x_1	x_2	re	s	rs
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

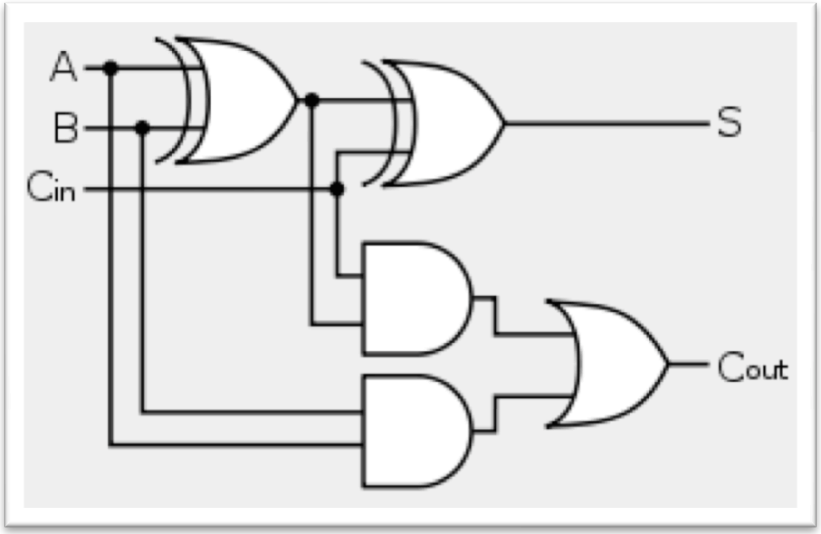
nombres entiers

- Demi-additionneur

A	B	S=A+B	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



- Additionneur complet



nombres entiers

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Remarque: pour la réalisation d'une colonne d'un d'additionneur
  - 100 cas à considérer pour la base 10
  - 4 cas à considérer pour la base 2
  - 16 x 16 = 256 cas pour la base 16
  - il faut considérer aussi la retenue entrante, donc le nombre de cas est multiplié par 2

- Conclusion
  - les opérateurs sont plus simples à réaliser en binaire, même s'il faut plus de chiffres pour représenter un nombre (répétition seulement)

Conversion entiers

- conversion
  - hexadécimal
  - binaire
  - octal

  } → décimal

calcul du polynome selon les coefficients et base utilisée

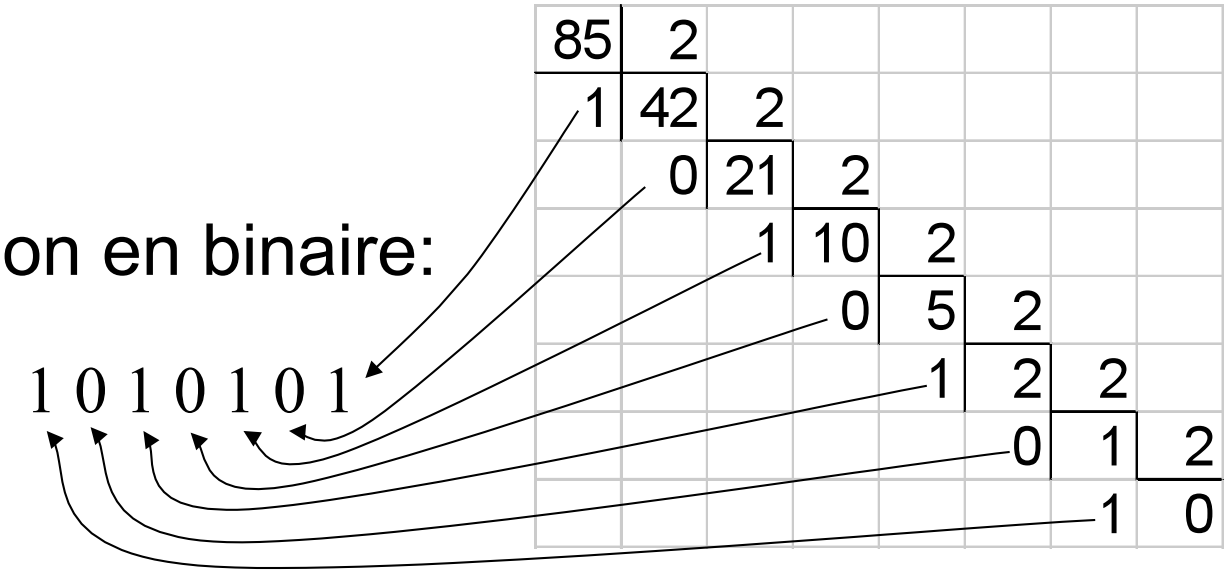
(calcul en décimal)

$100_h \rightarrow 1 \times 16^2 + 0 \times 16^1 + 0 \times 16^0 = 256_d$

$2B3_h \rightarrow 2 \times 16^2 + 11 \times 16^1 + 3 \times 16^0 = 691_d$

Conversion entiers

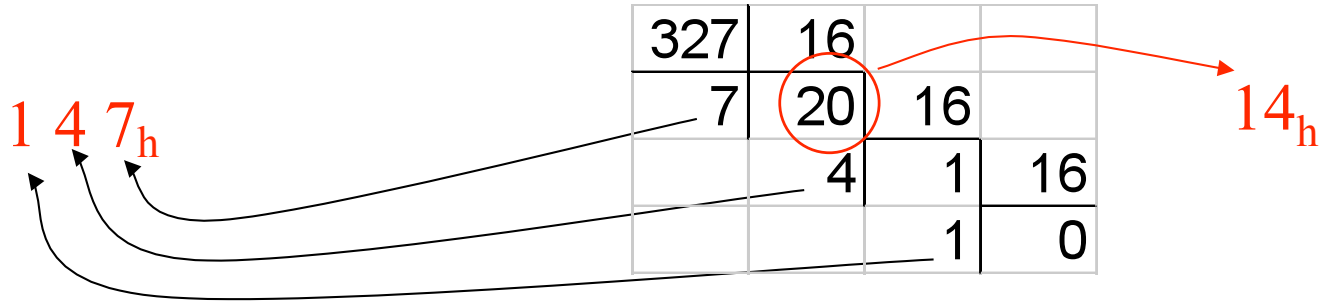
- algorithme de base
  - les restes des divisions successives par la base constituent la suite des chiffres à partir de la droite
- conversion décimal → binaire
  - soit un nombre N = 85
  - b = 2
  - représentation en binaire:



Conversion entiers

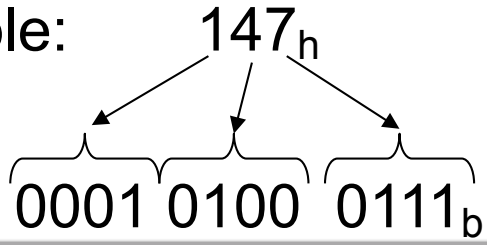
- conversion décimal => hexadécimal
  - exemple N = 327

    (division en décimal)



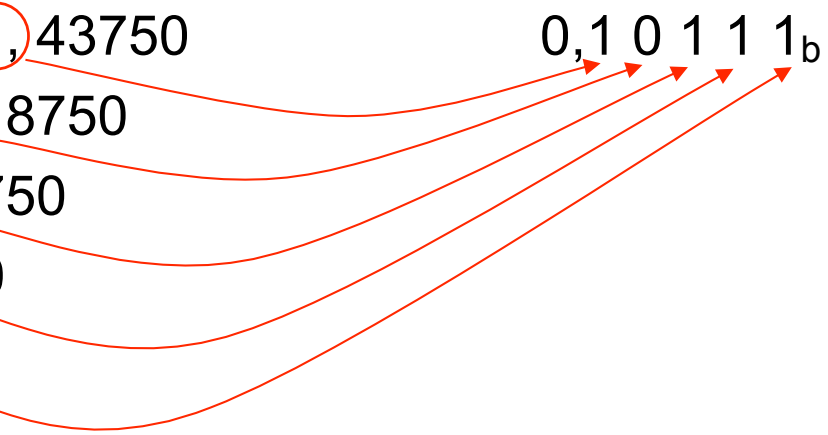
- conversion hexadécimal → binaire
  - chaque chiffre hexadécimal est remplacé par sa valeur binaire sur 4 chiffres

    exemple: 147_h



Conversion réels

- conversion partie entière : idem nombres entiers
- converstion partie fractionnaire
  - on procéde à des multiplications successives par la base pour obtenir les chiffres à partir de la virgule
  - exemple $0{,}71875_d \rightarrow$ binaire
    - 0,71875 x 2 = 1, 43750
    - 0,4375 x 2 = 0, 8750
    - 0,875 x 2 = 1, 750
    - 0,75 x 2 = 1, 50
    - 0,5 x 2 = 1, 0



  - A quelle condition la suite obtenue est-elle finie ?

Conversion réels

- conversion partie fractionnaire binaire→ décimal
  - calcul du polynome selon les coefficients et les puissances décroissantes de la base utilisée (calcul en décimal)
  - exemple $0,1\ 0\ 1\ 1\ 1_b$ → décimal
    - $1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$
    - $= 0,5 + 0 + 0,125 + 0,0625 + 0,03125$
    - $0,71875_d$
  - A quelle condition la suite obtenue est-elle finie ?

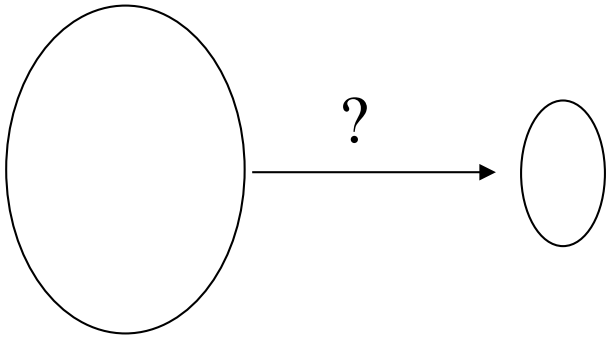
caractères

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- nombreuses variantes locales
  - latins non accentués (cultures anglophones)
  - latins accentuées (europe occidentale, amérique centrale et sud, vietnam)
  - grecs/cyrilliques (europe orientale)
  - arabes
  - persans
  - indiens (sanscrit)
  - thaïlandais
  - chinois (mandarin)
  - japonais (kanji)
  - ...

codage

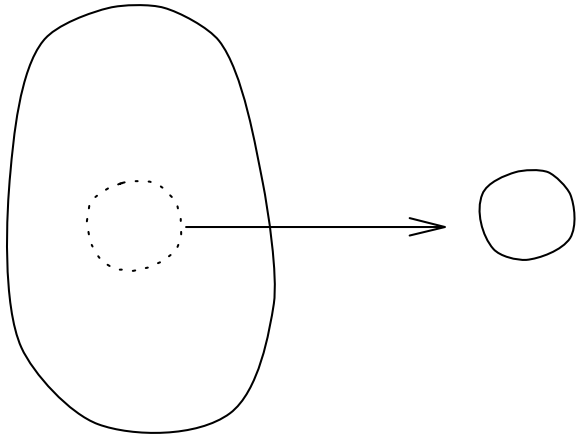
- système de représentation de valeurs à l'aide de suite de symboles
- pour des raisons pratiques (caractéristiques des processeurs) on souhaite souvent avoir des suites de longueurs fixes
- problème du codage des nombres: les ensembles sont infinis !



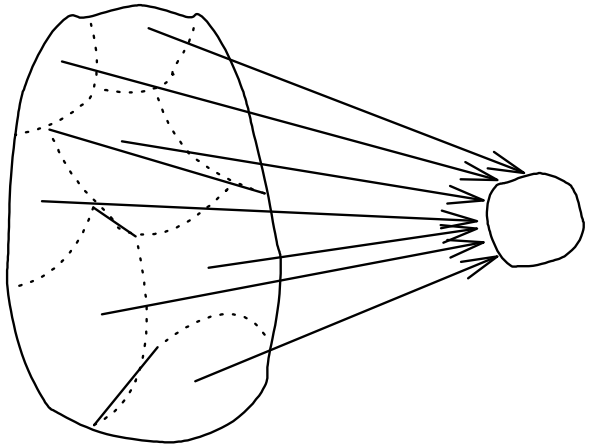
  - réduction
  - approximation

codages

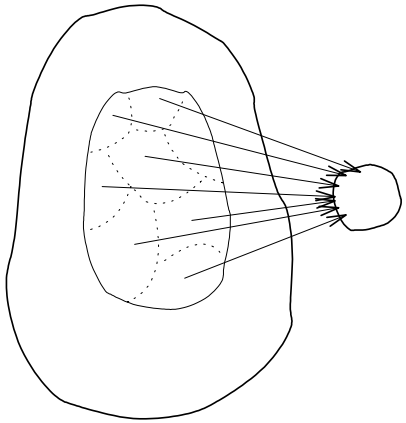
- réduction



approximation



- réduction + approximation



- => les "réels" informatiques sont des fractionnaires !

codages

- différents types de codage
  - codage de travail
    - objectif : faciliter les opérations sur les éléments
  - codage de stockage ou codage compressif
    - objectif : stocker en moyenne le minimum de symboles
  - codage pour transmission ou code à détection d'erreur (EDC)
    - objectif : transmettre le moins de symboles tout en gardant une certaine redondance pour détecter/corriger les erreurs de transmission
  - Codage destructif : ne peut être inversé sans perte

codages des entiers naturels

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- codage naturel
  - choix d'une longeur de représentation → exemple: 4,8,16,32,64 bits (suivant le processeur et les besoins)
  - réduction :
    - pour n bits de représentation on ne peut représenter que les entier compris entre 0 et 2^n-1
    - 4 bits 0 → 15
    - 8 bits 0 → 256
    - 16 bits 0 → 65 535
    - 32 bits 0 → 4 294 967 295
- codage d'un ensemble de M éléments : longueur ?
  - prendre le plus petit n tel que $M \leq 2^n$
  - soit : $\log_{10}(M) < \log_{10}(2^n) = n \log_{10}(2)$
- remarque : si $M = 10^m$ alors $m \log_{10}(10) \leq n \log_{10}(2) \Leftrightarrow n \geq 3{,}32m$
- approximation : $10^{3k} \approx 2^{10k}$ (1000 ≈ 1024, 1000 000 ≈ 1 048 576, …)

codages des entiers naturels

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- codage BCD (binary coded decimal) : chaque chiffre décimal est codé sur 4 bits :

chiffre→code		chiffre→code		chiffre→code	
0	0000	4	0100	8	1000
1	0001	5	0101	9	1001
2	0010	6	0110		
3	0011	7	0111		

1010,1011,1100,1101,1111 sont interdits d'emploi

- conclusion
  - facilité de traduction décimal ↔ binaire
  - perte de place (4 bits par chiffre, 5 codes inutilisés)
  - opérations arithmétiques complexes

  → peu utilisé en dehors de processeurs spéciaux

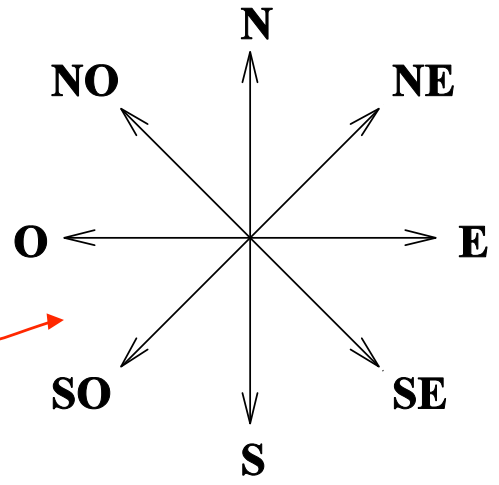
codages des entiers naturels

- codage de Gray
- Propriété : un seul bit change en passant à la valeur suivante
- Exemple sur 3 bits (valeurs de 0 à 7)

val	code	val	code	val	code	val	code
0	000	2	011	4	110	6	101
1	001	3	010	5	111	7	100

- utilisé pour indiquer des secteurs angulaires

0°-45°	→0	180°-225°	→4
45°-90°	→1	225°-270°	→5
90°-135°	→2	270°-315°	→6
135°-180°	→3	315°-360°	→7



- on peut affiner avec 4, 5,... bits

codages des nombres relatifs

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- codage naturel : idem codage des entiers plus bit de signe
  - avec n bits :

$$sa_{n-1}.a_2a_1a_0 = \sum_{i=0}^{\infty} a_i b^i$$

  - exemple sur 5 bits : $s\ a_3\ a_2\ a_1\ a_0$
  - réduction au domaine [$-2^4-1,\ +2^4-1$] soit [$-15,+15$]
  - 14 → 01110 -14 → 11110

- inconvénient
  - addition et soustraction sont deux opérations différentes

  → il faut constuire deux circuits différents
  - il faut soustraire les nombres négatifs au lieu de les additionner
  - deux zéros : un positif (00000) et un négatif (10000)

codages des nombres relatifs

- complément à 10
  - soit un nombre N représenté sur m digits.
  - Son complément à 10 est $\overline{N} = 10^m - N$
  - Aussi: complémenter à 9 chaque chiffre et ajouter 1 au résultat

    exemple N =14 sur 2 chiffres

    $\overline{N}$ = 100-14 = 86 = [(9-1)(9-4)]+1
- Propriété du complément à 10
  - $\overline{\overline{N}} = N$
  - $N + \overline{N} = 0$
  - avantage : addition usuelle et soustraire un nombre revient à ajouter son complément à 10. Le résultat est sur m digits

$$\begin{array}{r} 16 \\ +14 \\ \hline 30 \end{array} \qquad \begin{array}{r} 16 \\ -14 \\ \hline 02 \end{array} = \begin{array}{r} 16 \\ 86 \\ \hline 102 \end{array}$$

- ces propriétés sont les mêmes pour toutes les bases !

codages des nombres relatifs

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- complément à 2
  - soit un nombre N représenté sur m bits.
  - Son complément à 2 est $\overline{N} = 2^m - N$
  - remarque on peut aussi inverser chaque bit et ajouter 1 au résultat

    exemple $N = 14_d = 0000\ 1110$ sur 8 bits

    $\overline{N} = 1111\ 0001 + 1 = 1111\ 0010$
- Propriétés
  - addition identique au codage binaire usuel (sur m bits)
  - soustraction = addition du complément à 2
  - un seul zéro (0000 000 sur 8 bits)
  - Exemple :

$$\begin{array}{r} 0001\ 0000 \\ 1111\ 0010 \\ \hline 10000\ 0010 \end{array} \quad \begin{array}{r} 16 \\ -14 \\ \hline 02 \end{array} \qquad \begin{array}{r} 0001\ 0000 \\ +0000\ 1100 \\ \hline 0001\ 1100 \end{array} \quad \begin{array}{r} 16 \\ +14 \\ \hline 30 \end{array}$$

codages des nombres relatifs

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- complément à 2, avantages
  - domaine de valeurs pour m bits: $[-2^{n-1}, 2^{n-1}-1]$
  - exemple pour n=4

valeur	code	valeur	code
-8	1000	0	0000
-7	1001	1	0001
-6	1010	2	0010
-5	1011	3	0011
-4	1100	4	0100
-3	1101	5	0101
-2	1110	6	0110
-1	1111	7	0111

- attention aux dépassements de capacité (overflow)
  - Exemple : 6+6 ou -5-5

codages des réels

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Codage des Réels vers une partie des nombres fractionnaires par réduction et approximation
  - BCD
  - Virgule fixe
  - Virgule flottante (format IEEE)

codages des réels

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Codage BCD
  - On reprend le codage BCD des entiers et on utilise les combinaisons inutilisées pour le point et le signe
  - point : 1010
  - signe + : 1011
  - signe - :1100

  exemple :

Signe	Partie entière				Point	Partie fract.	
+ /-	8	7	3	6	.	5	7

- Avantages et inconvénients
  - Réduction : selon partie entière, approximation : selon partie fractionnaire
  - Complexité des opérations (parties fract. de longueurs différentes)
  - Adaptable à la longueur des nombre (longueur variable)
  - Même arrondis qu'en comptabilité

codages des réels

- Virgule fixe sur N bits dont M bits partie fractionnaire
  - Les M bits de droite constituent la partie fractionnaire
  - Le bit de gauche indique le signe (1 : négatif)
  - Si négatif le nombre est codé en complément à 2

    exemple (N=8, M=6):

Signe	Partie entière								Partie fract.					
1 / 0	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

  - Valeur : soit $X = \sum_{i=0}^{N-1} a_i 2^i$
    - $si \quad a_{N-1} = 0 \quad alors \quad valeur = X \times 2^{-M}$

      $si \quad a_{N-1} = 1 \quad alors \quad valeur = -(\overline{X} \times 2^{-M})$
    - Réduction des Réels à $[-2^{N-1}/2^M, (2^{N-1}-1)/2^M]$, approximation à 2^{-M}

codages des réels

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Problème
  - Nombres très grands avec de nombreux zéros non significatif à droite
  - Nombre très petits avec de nombreux zéros non significatifs à gauche
  - Représenter ces zéros prendrait de la place
- Solution représenter la partie significative d'un coté et ajouter un exposant pour la situer dans le domaine de valeurs.

codages des réels

- IEEE 754-1985 (standard floating point format)
  - Trois champs : signe, exposant, mantisse
  - Simple précision (32 bits)

31	30 23	22 0
s	Exposant (8 bits)	Mantisse (23 bits)

  - Représentation normalisée

    représenter le nombre sous la forme :

    Non mémorisé

    $+/- 1,a_{23}a_{22}a_{21}\ldots\ldots\ldots a_0 \times 2^e$

    - Exposant = e +127 (translation pour codage exposant négatif)
    - Mantisse = $a_{23}a_{22}a_{21}\ldots\ldots\ldots a_0$
    - Approximation à 2^{-23}
    - Réduction des Réels à ?

exemple codage réel (simple précision)

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- exemple: −118.625
  - traduire en binaire : 1110110.101
  - normaliser: laisser un seul 1 avant virgule: 1.110110101×2^6
  - ne garder bits qu'après virgule et remplissage de 0: 11011010100000000000000
  - L'exposant est 6, positif, codé par 127+6= 133: 10000101
  - Le nombre est négatif: le champ *s* est 1.

codages des réels

- IEEE 754-1985 (standard floating point format)
  - Représentation dénormalisée (si valeur < à 2^{-23})
  - représenter le nombre sous la forme

    $+/- 0,a_{23}a_{22}a_{21}.....a_0 \times 2^{-126}$

    - Exposant = 0
    - Mantisse = $a_{23}a_{22}a_{21}.....a_0$
    - réduction à $]-2 \times 2^{exp-127}, -2 \times 2^{exp-127}[$
    - Approximation de 2^{-23} à 1
  - exemples

	nombre	notation utilisé	exposant	mantisse (23b)
N	1×2^{12}	$1.0 \times 2^{12=139-127}$	139	0000…….0
N	3×2^{12}	$1.1 \times 2^{13=140-127}$	140	1000…….0
D	3×2^{230}	0.0011×2^{-126}	0	00110……..0

codages des réels

- IEEE 754-1985 (standard floating point format)
  - Résumé (simple précision)

	exp.	fract.	valeur
N	>0, <255	$a_{22}..a_0$	$(1+\sum_{i=0}^{22} a_i 2^{i-23}) \times 2^{\exp-127}$
D	0	$a_{22}..a_0$	$(\sum_{i=0}^{22} a_i 2^{i-23}) \times 2^{-126}$
D	0	0	0
D	255	0	$+\infty$, $-\infty$
	255	$\neq 0$	NaN (Not a Number)

codages des réels

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- IEEE 754-1985 (standard floating point format)
  - double précision

63	62 52	51 0
s	Exposant (11 bits)	Mantisse (52 bits)

  - Exposant : 11 bits, translation : +1023
  - Mantisse: 52 bits

codages des réels

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- IEEE 754-1985 (standard floating point format)
  - Avantages et inconvénient
    - Implantation complexe mais processeurs spécialisés
    - Overflow, -∞, +∞ sont définis dans la norme
    - Les valeurs dénormalisées permettent d'approcher du zéro
    - Norme de plus en plus utilisée
    - Portabilité des programmes
  - Domaines de valeurs:

	simple préc.	double préc.
Partie fract	23 bits	52 bits
Normalisé	10^{-38} à 10^{+38}	10^{-308} à 10^{+308}
Dénormalisé	10^{-45} à 10^{+38}	10^{-324} à 10^{+308}

Codage des caractères

- Opérations sur les caractères
  - Comparaison selon l'ordre alphabétique
  - Transformation minuscule ⇔ capitales

  - Caractères ASCII
    - 7 bits (ISO-7) : 128 caractères
    - étendu 8 bits (ISO 8859-1 latin) : les 128 caractères ajoutés comprennent les caractères accentués
  - Caractères UNICODE (16 bits) : contiennent les caractères des alphabets les plus répandus

Codage des caractères

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- ASCII (IS0-7 bits)

American Standard Code for Information Interchange								
6,5,4,3	bits 2,1,0							
	000	001	010	011	100	101	110	111
0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
0001	BS	HT	NL	VT	NP	CR	SO	SI
0010	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB
0011	CAN	EM	SUB	ESC	FS	GS	RS	US
0100	SP	!	“	#	$	%	&	’
0101	(	)	*	+	,	-	.	/
0110	0	1	2	3	4	5	6	7
0111	8	9	:	;	<	=	>	?
1000	@	A	B	C	D	E	F	G
1001	H	I	J	K	L	M	N	O
1010	P	Q	R	S	T	U	V	W
1011	X	Y	Z	[	\	]	†	-
1100	‘	a	b	c	d	e	f	g
1101	h	i	j	k	l	m	n	o
1110	p	q	r	s	t	u	v	w
1111	x	y	z	{	|	}	~	DEL

Quantité d'information

- Lorsqu'une source émet des messages composés sur un alphabet A de N symboles on définit la quantité d'information contenue dans chaque symbole s_i par :

  $q_i = - log_2 p_i$

  où p_i est la probabilité d'émission du symbole s_i
- Remarque : puisque p_i est ≤ 1, q_i est positive
- L'unité de mesure de q_i est le bit
- un symbole s contient alors en moyenne la quantité d'information :

$$q_s = -\sum_{i=0}^{n-1} p_i \log_2 p_i$$

Quantité d'information

- Si A est un alphabet contenant N symboles équiprobables on définit la quantité d'information contenue dans chaque symbole a_i par

$$q_i = log_2 N \text{ (bits)}$$

un symbole s contient alors en moyenne la quantité d'information

$$q_s = \log_2 N$$

- Propriété : $\log_2 N \geq -\sum_{i=0}^{N-1} p_i \log_2 p_i$

- Un message M = $a_0 a_1 a_2 \ldots\ldots a_m$ contient alors en moyenne une quantité d'information : $Q_M = m \times q_s$

Quantité d'information

- conséquence
  - Soit des messages M composés de symboles de A
  - Soit I le nombre moyen de bits mesurant l'information contenue dans un symbole s
  - Soit un codage des symboles de A à l'aide de suites de bits
  - soit B le nombre moyen de bits utilisés en moyenne pour coder les symboles de A.

  - L'efficacité du codage choisi peut être évaluée en considérant la relation entre B et I

Codage compressifs

- Codage de Huffman (longueur variable)

  (symboles non équiprobables seulement)

  - Principe : assigner des codes plus courts aux symboles les plus fréquents : on diminue ainsi la quantité de bits à utiliser

  - Problème : les codes doivent être reconnus pour que le décodage se fasse sans ambiguité

    exemple : A → 1, B →10, C →11

    pas de différence entre AA et C !

Codage compressifs

- Codage de Huffman (longueur variable)
  - Exemple
    - Symbole A B C D
    - Fréquence 8/16 4/16 2/16 2/16
    - Probablité 0,5 0,25 0,125 0,125
    - Codage:

$$A \rightarrow 0 \quad B \rightarrow 10 \quad C \rightarrow 110 \quad D \rightarrow 111$$

    - Longueur moyenne d'un message contenant un symbole: $L_m = 1x0,5 + 2x0,25 + 3x0,125 + 3x0,125 = 1,75$
    - Avec le codage naturel : $L_m = 2$

Code détecteur d'erreur (CDE)

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Code redondant avec bit de parité
  - Principe : pour chaque paquet de n bits, on complète par un bit de plus dont la valeur est calculée pour que le nombre total de bits égaux à 1 soit pair.
  - Exemples :
  - 110101 → 0 110001 →1
  - Lorsqu'on relit ou reçoit le paquet complet, on recalcule le bit de parité. S'il est différent de celui reçu alors une erreur est détectée