Université Internationale de Casablanca

Architecture des ordinateurs

Dr. Laila DAMRI

Spécialité : MIAGE

Année universitaire : 2018-2019

Plan du cours

Partie 1

- Chapitre 1 : Codage de l'information
- Chapitre 2 : Architecture matérielle interne des machine

Partie 2

- Chapitre 3 : Algèbre De Boole
- Chapitre 4 : Logique combinatoire
- Chapitre 5 : Logique séquentielle

Organisation du cours

- Séance de 4 heures par semaine (cours et TD).
- 2 Contrôles Continus (30%).
- 1 Examen Final (50%).
- Assiduité (10%).
- Participation (10%).

Objectifs du cours

- Compréhension de l'organisation des ordinateurs.
- Modélisation du fonctionnement des ordinateurs.
- Représentation des données.
- Représentation du calcul arithmétique et logique.
- Compréhension des fondements des traitements.
- des programmes par les ordinateurs.

1

Codage de l'information

Introduction

Références

- Unités de mesure de capacité
  - Kilo = $10^3 \approx 2^{10} = 1024$
  - Méga = $10^6 \approx 2^{20} = 1\ 048\ 576$
  - Giga = $10^9 \approx 2^{30} = 1\ 073\ 74\ 824$
  - Tera = $10^{12} \approx 2^{40} = 1\ 099\ 511\ 627\ 776$
  - Peta = $10^{15} \approx 2^{50} = 1\ 125\ 899\ 906\ 842\ 624$
- Unités de mesure de temps
  - ms = milliseconde = 10^{-3} s = 0,001 s
  - µs = microseconde = 10^{-6} s = 0,000 0001 s
  - ns = nanoseconde = 10^{-9} s = 0,000 000 001 s
  - ps = picoseconde = 10^{-12} s = 0,000 000 000 001 s

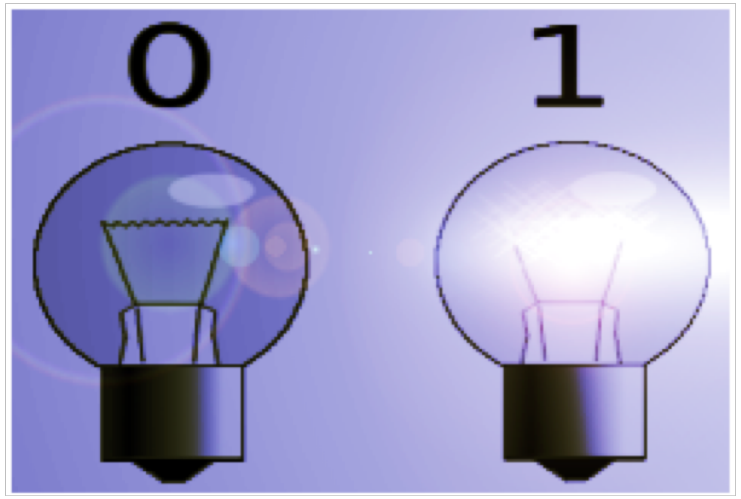
Introduction

Historique

Dans la fin des années 30, Claude Shannon, mathématicien, démontra qu'avec des interrupteurs, on pouvait effectuer des opérations logiques en associant le nombre 1 au vrai (fermé) et 0 au faux (ouvert).

Il s'est aidé des travaux de Boole. A eux deux, ils ont posé les briques de base de l'informatique.

À l'heure actuelle nos ordinateurs ne sont fait que de "transitors » qui gèrent l'état 0 ou 1.



Introduction

Historique

- L’ordinateur est né pour répondre à un besoin de calcul plus vite :
  - Automatisation du calcul

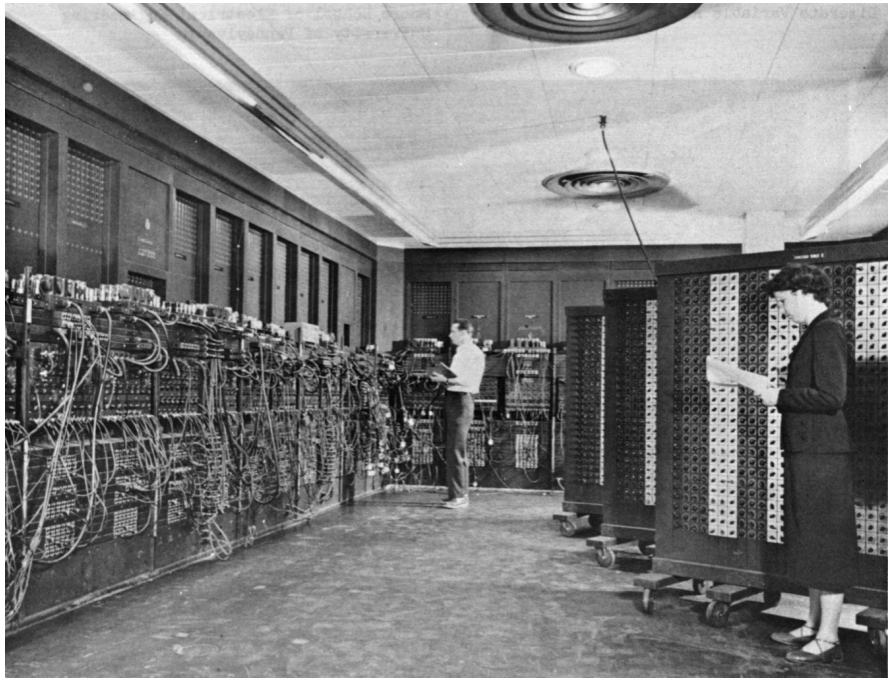
- 18ième siècle et avant : les principes fondateurs
- 19ième siècle : les calculateurs
- 20ème siècle : théorie de l'information et la machine universelle
- 1945 : Architecture de Von Neumann et naissance de l'ordinateur
- Les années 1950 : 1ere génération : tubes a vides
- Les années 1960 : 2eme génération : transistors
- Les années 1970 : 3eme génération : circuits intégrés
- Les années 1980 : 4eme génération : puces avec des millions de transistors

Introduction

Historique

Premier ordinateur (ENIAC)

- Electronic Numerical Integrator Analyser and Calculator –(ENIAC) 1945
- Construit a l'Université de Pennsylvanie
- Technologie des tubes a vide (17468), 7200 diodes, 70000 résistance, 10000 capacités
  - pesant 30 tonnes !
  - occupant167 m^2
- Construit pour être Turing-complet
  - Peut simuler toutes les machines de Turing à une Bande
- Multiplication de 2 nombres de 10 chiffres en 3ms !



Codage de l’information

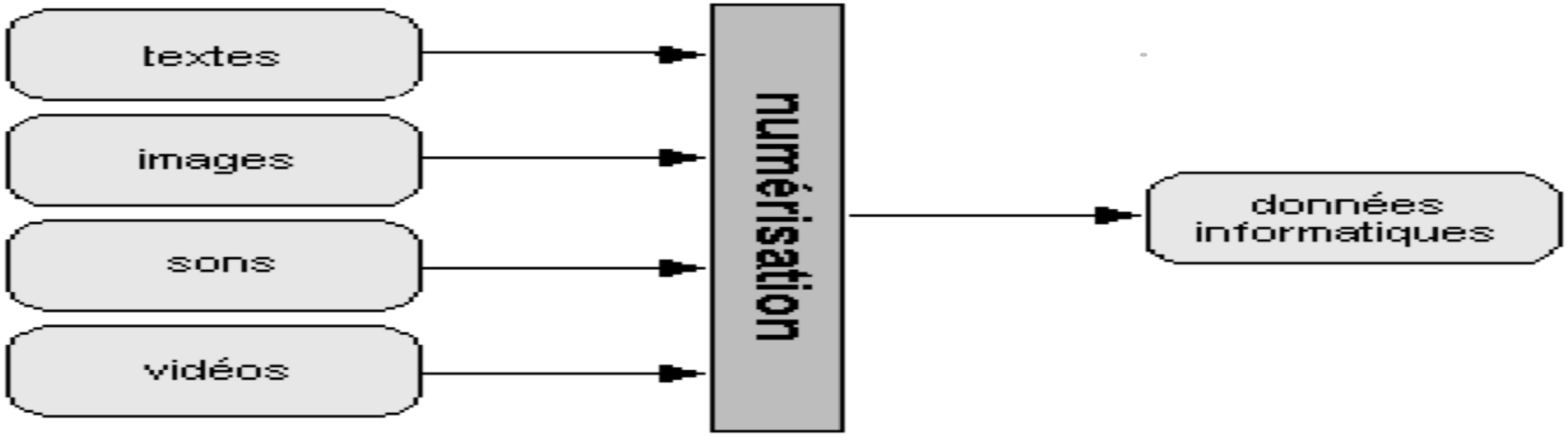
Présentation

- Opération d'identification et de représentation d'un ensemble d'informations à l'aide d'un code, informatique ou non.
- Très utilisé en informatique, le codage permet de créer, manipuler et faire circuler avec des outils informatiques génériques des objets hétérogènes (son, texte, image fixe ou animée).
- Ce terme fait référence aussi bien au codage de données en mode caractère (codage ASCII), en mode image, ...

Codage de l’information

Définition

- Le **codage de l’information** concerne les moyens de formaliser l'information afin de pouvoir la manipuler, la stocker ou la transmettre. Il ne s'intéresse pas au contenu mais seulement à la forme et à la taille des informations à coder.



Codage de l’information

Codage de données non numériques

Principe

A chaque caractère correspond un nombre binaire qui lui est propre.

Les alphabets étant différents d’un pays à un autre (nombre et type de caractères), nous allons trouver de nombreux codages différents.

le codage ASCII

Les codage ASCII étendu
- **la norme ISO8859**
- **la norme ANSI**

Unicode
- **USC-2**
- **UTF-8**

Codage de l'information

Codage de données non numériques

Code ASCII

American Standard Code for Information Interchange

7 bits (128 caractères)

26 lettres majuscules A – Z

26 lettres minuscule a – z

10 chiffres 0 à 9

caractères de ponctuation

sp,! " #$%& ' ()*+,-. /< = >?@ [ ] ^_` { | } ~

caractères de contrôle :

null, etx, bel, bs, ht, lf, vt, ff, cr, …, del

Pas de caractères accentués

1 bit supplémentaire utilisé pour le contrôle de parité

ASCII étendu

8 bits -> 256 caractères

caractères internationaux

caractères semi-graphiques

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0	0x00	(sp)	32	40	0x20	@	64	100	0x40	`	96	140	0x60
(soh)	1	1	0x01	!	33	41	0x21	A	65	101	0x41	a	97	141	0x61
(stx)	2	2	0x02	"	34	42	0x22	B	66	102	0x42	b	98	142	0x62
(etx)	3	3	0x03	#	35	43	0x23	C	67	103	0x43	c	99	143	0x63
(eot)	4	4	0x04	$	36	44	0x24	D	68	104	0x44	d	100	144	0x64
(enq)	5	5	0x05	%	37	45	0x25	E	69	105	0x45	e	101	145	0x65
(ack)	6	6	0x06	&	38	46	0x26	F	70	106	0x46	f	102	146	0x66
(bel)	7	7	0x07	'	39	47	0x27	G	71	107	0x47	g	103	147	0x67
(bs)	8	10	0x08	(	40	50	0x28	H	72	110	0x48	h	104	150	0x68
(ht)	9	11	0x09	)	41	51	0x29	I	73	111	0x49	i	105	151	0x69
(nl)	10	12	0x0a	*	42	52	0x2a	J	74	112	0x4a	j	106	152	0x6a
(vt)	11	13	0x0b	+	43	53	0x2b	K	75	113	0x4b	k	107	153	0x6b
(np)	12	14	0x0c	,	44	54	0x2c	L	76	114	0x4c	l	108	154	0x6c
(cr)	13	15	0x0d	-	45	55	0x2d	M	77	115	0x4d	m	109	155	0x6d
(so)	14	16	0x0e	.	46	56	0x2e	N	78	116	0x4e	n	110	156	0x6e
(si)	15	17	0x0f	/	47	57	0x2f	O	79	117	0x4f	o	111	157	0x6f
(dle)	16	20	0x10	0	48	60	0x30	P	80	120	0x50	p	112	160	0x70
(dc1)	17	21	0x11	1	49	61	0x31	Q	81	121	0x51	q	113	161	0x71
(dc2)	18	22	0x12	2	50	62	0x32	R	82	122	0x52	r	114	162	0x72
(dc3)	19	23	0x13	3	51	63	0x33	S	83	123	0x53	s	115	163	0x73
(dc4)	20	24	0x14	4	52	64	0x34	T	84	124	0x54	t	116	164	0x74
(nak)	21	25	0x15	5	53	65	0x35	U	85	125	0x55	u	117	165	0x75
(syn)	22	26	0x16	6	54	66	0x36	V	86	126	0x56	v	118	166	0x76
(etb)	23	27	0x17	7	55	67	0x37	W	87	127	0x57	w	119	167	0x77
(can)	24	30	0x18	8	56	70	0x38	X	88	130	0x58	x	120	170	0x78
(em)	25	31	0x19	9	57	71	0x39	Y	89	131	0x59	y	121	171	0x79
(sub)	26	32	0x1a	:	58	72	0x3a	Z	90	132	0x5a	z	122	172	0x7a
(esc)	27	33	0x1b	;	59	73	0x3b	[	91	133	0x5b	{	123	173	0x7b
(fs)	28	34	0x1c	<	60	74	0x3c	\	92	134	0x5c	|	124	174	0x7c
(gs)	29	35	0x1d	=	61	75	0x3d	]	93	135	0x5d	}	125	175	0x7d
(rs)	30	36	0x1e	>	62	76	0x3e	^	94	136	0x5e	~	126	176	0x7e
(us)	31	37	0x1f	?	63	77	0x3f	_	95	137	0x5f	(del)	127	177	0x7f

Codage de l’information

Codage de données non numériques

Unicode

1 a 4 octets (1114112 caractères).

Codage unique quelque soit la plateforme, le logiciel, la langue.

Code universel contenant, en plus de tous les caractères connus, 42 000 caractères asiatiques. Le code ASCII est contenu dans les 128 premiers caractères d’UNICODE.

UNICODE est supporté par Windows NT, Windows 2000, Java, et certains systèmes UNIX.

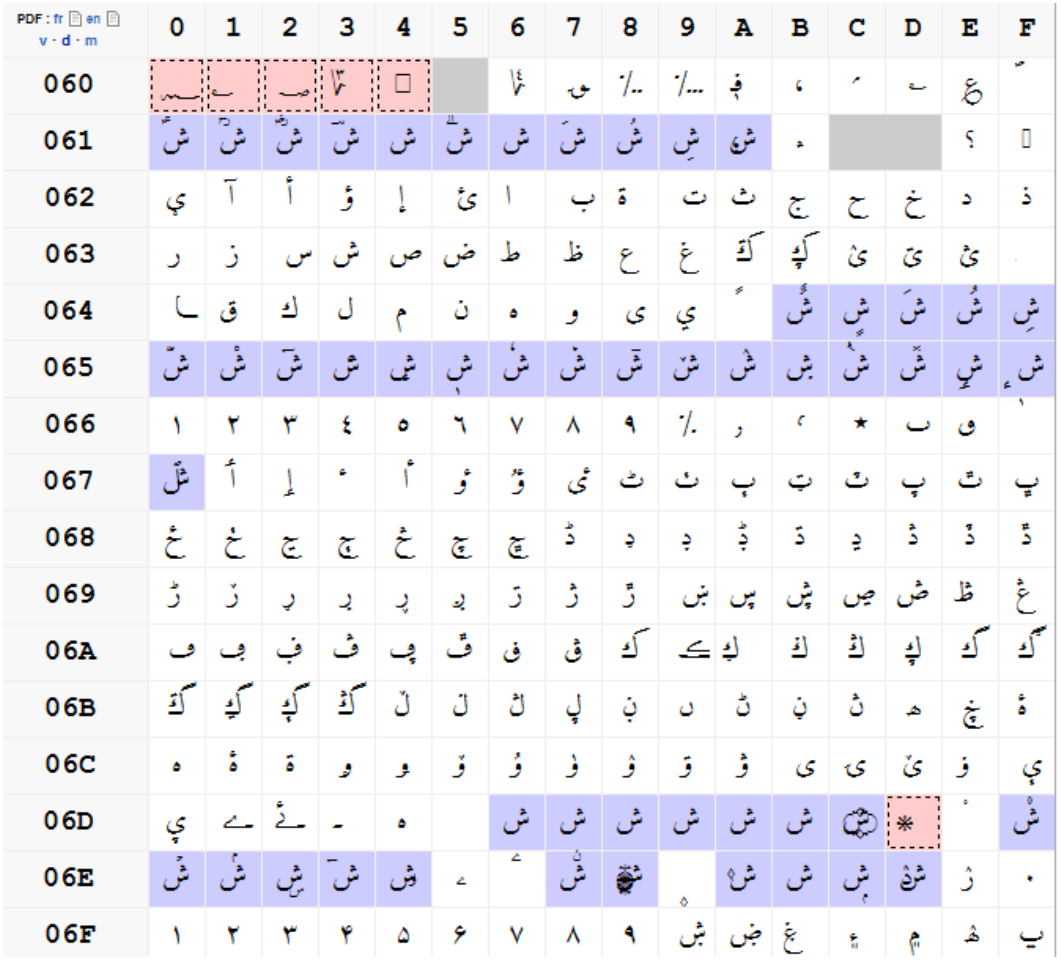
Unicode (UTF-7 , UTF-8, UTF-16, UTF- 32)

(Universal Character Set Transformation Format)

UTF-7

ï»¿Bonjour,

Je suis parti Ã l'Ã©cole



Codage de l’information

Codage de données numériques

Nombres entiers positifs ou nuls : 0 ; 1 ; 315

Nombres entiers négatifs : -1 ; -1255

Nombres fractionnaires : 3,1415 ; -0,5

Un algorithme de codage réalise la conversion en binaire. Les opérations arithmétiques (+, -, *, /) sont effectuées en arithmétique binaire.

Numérotation

Base décimale (Base 10)

- « humain » nombres (base 10)
- chiffres compris entre 0-9
- Nombres ‘ successions de chiffres ’
- chiffre positif pour le poids
- (puissance de 10 en général)

$$1234 = 1x1000 + 2x100 + 3x10 +4 = 1x10^3 + 2x10^2 + 3x10^1 + 4x10^0$$

$$\mathbf{d_n d_{n-1} \ldots d_2 d_1 d_0 = \Sigma\, d_i \times 10^i}$$

Numérotation

Base binaire (Base 2)

- Un système de numération utilisant la base 2. Toutes les informations sont codées avec des 0 et des 1.
  - 1 bits : 2^1 possibilités = 0, 1
  - 2 bits : 2^2 possibilités = 00, 01, 10, 11
  - 3 bits : 2^3 possibilité = 000, 001, 010, 011, 100, 101, 110, 111
  - n bits : 2^n possibilités
- x
- Un mot = un ensemble de bit avec un poids 2^n; 2^{n-1}...2^1; 2^0

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

Numérotation

Base binaire (Base 2)

- Le binaire

0	0	10	1010	31	1 1111
1	1	11	1011	32	10 0000
2	10	12	1100	63	11 1111
3	11	13	1101	64	100 0000
4	100	14	1110	127	111 1111
5	101	15	1111	128	1000 0000
6	110	16	1 0000	255	1111 1111
7	111	17	1 0001	256	1 0000 0000
8	1000	18	1 0010		
9	1001	19	1 0011		
		20	1 0100		
		24	1 1000		

Numérotation

Base binaire (Base 2)

- En décimal, avec **n** chiffres, on obtient 10^n combinaisons possibles, i.e. on peut compter de 0 à 10^n-1.

  Exemple : Avec 3 chiffres, on a 10^3 = 1000 combinaisons possibles et on peut compter de 000 à 999.

- En binaire, avec **n** bits, on obtient 2^n combinaisons possibles, i.e. on peut compter de 0 à 2^n-1

  Exemple : avec 8 bits, on a 2^8 = 256 combinaisons possibles et on peut compter de 00000000 à 11111111, i.e. de 0 à 255.

Numérotation

Base octale (Base 8)

- Ce système permet la représentation d'un nombre à partir des huit symboles suivants :

{ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 }

- N'importe quelle combinaison de ces symboles nous donne un nombre octal.

Exemple

147 ; 12 ; 4367 sont des combinaisons de plusieurs symboles pour construire un nombre.

Attention !! 89 n'est pas un nombre octal.

Numérotation

Base hexadécimal (Base 16)

- Ce système permet la représentation d'un nombre à partir des seize symboles suivants :

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

- N'importe quelle combinaison de ces symboles nous donne un nombre hexadécimal.

Exemple

342 ; 12B ; 4EF7 sont des combinaisons de plusieurs symboles pour construire un nombre.

Numérotation

Conversion entre les bases

- Les ordinateurs ne sont toujours capables que d'une seule chose : faire des calculs, et uniquement cela.
- Lorsqu'un ordinateur traite du texte, du son, de l'image, de la vidéo, il traite en réalité des nombres. En fait, dire cela, c'est déjà lui faire trop d'honneur. Car même le simple nombre « 3 » reste hors de portée de l'intelligence d'un ordinateur.
- Un ordinateur manipule exclusivement des **informations binaires**, dont on ne peut même pas dire sans être tendancieux qu'il s'agit de nombres.

Numérotation

Conversion entre les bases

Conversion de la base X vers la base 10

Il suffit de faire le développement en polynôme de ce nombre dans la base X , et de faire la somme par la suite.

Exemple

$$(1101)_2 = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = (13)_{10}$$

$$(1A7)_{16} = 1*16^2 + A*16^1 + 7*16^0 = 1*16^2 + 10*16^1 + 7*16^0 = 256 + 160 + 7 = (423)_{10}$$

$$(1101{,}101)_2 = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3} = (13{,}625)_{10}$$

Exercice d'application

Convertir vers le décimal les nombres suivants :

a. $(1AB)_{16}$ b.$(57)_8$ c.$(1101)_{16}$ d.$(10110)_2$

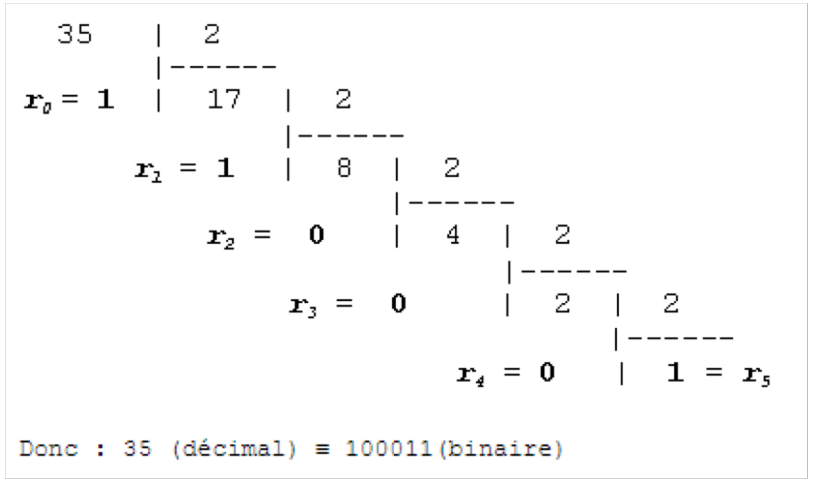
Numérotation

Conversion entre les bases

Conversion de la base 10 vers la base X

La conversion se fait en prenant les restes des divisions successives sur la base X dans le sens inverse.

Exemple



Exercices d'application

1) Convertir vers les bases indiquées les nombres décimaux suivants :

a. $(156)_{10} \rightarrow (\ ?)_2$ b. $(57)_{10} \rightarrow (\ ?)_8$ c. $(1101)_{10} \rightarrow (\ ?)_2$ d. $(1110)_{10} \rightarrow (\ ?)_{16}$

2) Convertir les nombres suivants :

a. $(13C)_{16} \rightarrow (\ ?)_8$ b. $(57)_8 \rightarrow (\ ?)_2$ c. $(1101)_2 \rightarrow (\ ?)_{16}$ d. $(117)_8 \rightarrow (\ ?)_{16}$

Numérotation

Conversion entre les bases

Conversion de la base 10 vers la base X

Cas d'un nombre réel

- Un nombre réel est constitué de deux parties : la partie entière et la partie fractionnelle.
- La partie entière est transformée en effectuant des divisions successives.
- La partie fractionnelle est transformée en effectuant des multiplications successives par 2.

Exemples

Exemple 1. $35,625=(?)_2$

P.E= 35 = $(100011)_2$

PF= $(0,625)=(0,101)_2$

Donc $35,625=(100011,101)_2$

$$0,625 * 2 = 1,25$$
$$0,25 * 2 = 0,5$$
$$0,5 * 2 = 1,0$$

Exercice d'application

Effectuer les transformations suivantes :

a. $(23,65)_{10}=(?)_2$ b. $(18,190)_{10}=(?)_2$ c. $(2,75)_{10}=(?)_2$ d. $(46,625)_{10}=(?)_2$

Numérotation

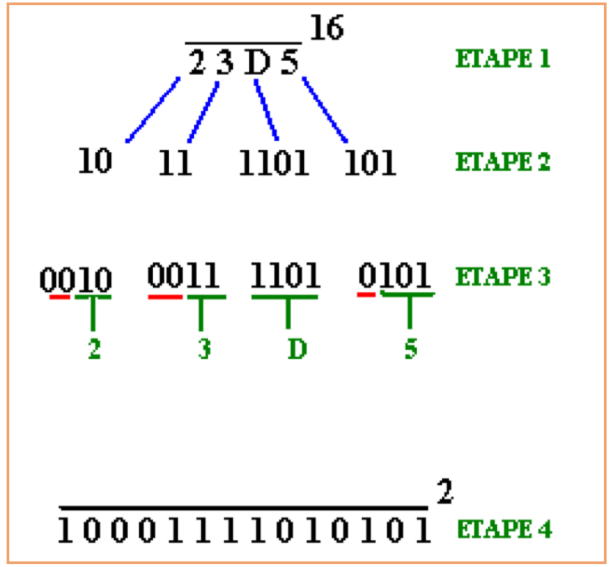
Conversion entre les bases

Conversion de la base 16 vers la base 2

- En Hexadécimal chaque symbole de la base s'écrit sur 4 bits.
- L'idée de base est de replacer chaque symbole par sa valeur en binaire sur 4 bits (faire des éclatements sur 4 bits).

Exemple

$(23D5)_{16} \rightarrow (\ ?)_2$



Exercices d'application

Convertir les nombres en binaire vers l'hexadécimal :

a. $(6E)_{16}=(?)_2$ b. $(AB,CD)_{16}=(?)_2$ c. $(79,001)_{16}=(?)_2$ d. $(101,5F4)_{16}=(?)_2$

Numérotation

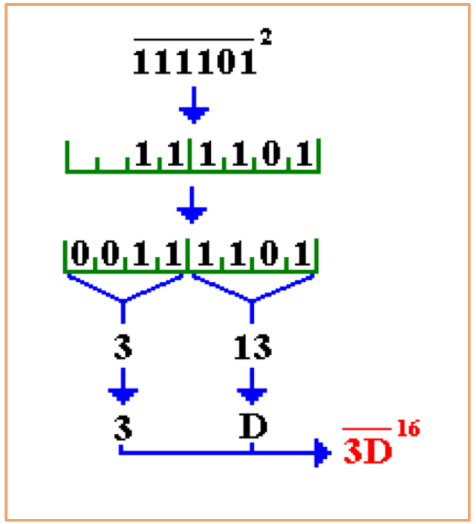
Conversion entre les bases

Conversion de la base 2 vers la base 16

- L'idée de base est de faire des regroupements de 4 bits à partir du poids faible.
- Par la suite remplacer chaque regroupement par la valeur Hexadécimal correspondante.

Exemple

$(111101)_2=(?)_{16}$



Exercices d'application

Convertir les nombres en binaire vers l'hexadécimal :

a. $(11010)_2=(?)_{16}$ b. $(1001,101)_2=(?)_{16}$ c. $(1000,001)_2=(?)_{16}$ d. $(10101,0101)_2=(?)_{16}$

Numérotation

Conversion entre les bases

Conversion de la base 8 vers la base 2

- En octal chaque, symbole de la base s'écrit sur 3 bits en binaire.
- L'idée de base est de replacer chaque symbole dans la base octal par sa valeur en binaire sur 3 bits (faire des éclatement sur 3 bits).

Exemples

$(345)_8=(\underline{011}\ \underline{100}\ \underline{101})_2$

$(65,76)_8=(\underline{110}\ \underline{101}, \underline{111}\ \underline{110})_2$

$(35,34)_8=(\underline{011}\ \underline{101}\ ,\ \underline{011}\ \underline{100})_2$

Exercices d'application

Convertir les nombres en binaire vers l'octal :

a. $(716)_8=(?)_2$ b. $(101,101)_8=(?)_2$ c. $(732,01)_8=(?)_2$ d. $(77,54)_8=(?)_2$

Numérotation

Conversion entre les bases

Conversion de la base 2 vers la base 8

- L'idée de base est de faire des regroupements de 3 bits à partir du poids faible.
- Par la suite remplacer chaque regroupement par la valeur octal correspondante.

Exemple

$(1100101001011 0)_2=(\underline{011}\ \underline{001}\ \underline{010}\ \underline{010}\ \underline{110})_2=(31226)_8$

$(110010100,10101)_2= (\underline{110}\ \underline{010}\ \underline{100}\ ,\ \underline{101}\ \underline{010})_2=(624,51)_8$

Exercice d'application

Convertir les nombres en binaire vers l'octal :

a. $(11010)_2=(?)_8$ b. $(101,101)_2=(?)_8$ c. $(1000,001)_2=(?)_8$ d. $(10101,0101)_2=(?)_8$

Numérotation

Conversion entre les bases

Exercice d'application

- Effectuer les conversions suivantes :

Décimal	Binaire	Héxadécimal	Octal
1	00000001	001	001
10			
	010100101		
		065	
			764

Numérotation

Codage des entiers naturels

Présentation

Le code binaire naturel est le code dans lequel on exprime un nombre selon le système de numération binaire. C’est un entier qui ne peut être que positif ou nul, ainsi que les bits sont rangés selon leur poids.

Définitions de base

Un quartet : c'est un mot de 4 bits (0-15)

Un octet : c'est un mot de 8 bits (0-255)

Un "kilo" : unité de capacité de traitement numérique (10 bits: 0-1023)

Numérotation

Codage des entiers naturels

Opérations binaires

Opération d'addition binaire

L'addition entre deux nombres binaires se base sur les règles suivantes :

A	B	RÉSULTAT DE (A+B)	RETENUE DE (A+B)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Exemples

- Un exemple d'addition simple :

```
   1011
+  0100
   1111
```

- Un peu plus compliqué :

```
    11
   0010
+  0111
   1001
```

Codage des entiers naturels

Opérations binaires

Opération d'addition binaire

Exercices d'application

Effectuer les opérations binaires suivantes :

a. $(11010)_2+(1001)_2$ b. $(1001)_2+(1111)_2$ c. $(1001001)_2+(1111101)_2$

d. $(11111)_2+(11111)_2$

Numérotation

Codage des entiers naturels

Opérations binaires

Opération de multiplication binaire

La multiplication entre deux nombres binaires se base sur les règles suivantes :

Multiplication

0	0	0
0	1	0
1	0	0
1	1	1

Exemple

Multiplication binaire

```
      1 0 1 1  (4 bits)
  *   1 0 1 0  (4 bits)
  -------------
      0 0 0 0
    1 0 1 1 .
  0 0 0 0 .
1 0 1 1 .
-----------
0 1 1 0 1 1 1 0
```

Sur 4 bits le résultat est faux

Sur 7 bits le résultat est juste

Sur 8 bits on complète à gauche par un 0

Codage des entiers naturels

Opérations binaires

Opération de multiplication binaire

Exercices d'application

Effectuer les opérations binaires suivantes :

a. $(11010)_2*(1001)_2$ b. $(1001)_2*(111)_2$ c. $(1001001)_2*(1001)_2$

d. $(10111)_2*(111)_2$

Numérotation

Codage des entiers naturels

Opérations binaires

Opération de soustraction binaire

La soustraction entre deux nombres binaires se base sur les règles suivantes :

Soustraction

A - B				
	0	0	0	0
	0	0	1	1
	0	1	0	1
	0	(1)1	1	0

Retenu

Exemple

$$\begin{array}{cccccccc} & 1 & & 1 & 1 & & & \\ \hline & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ - & & 1 & 0 & 1 & 1 & 0 & 1 \\ \hline & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}$$

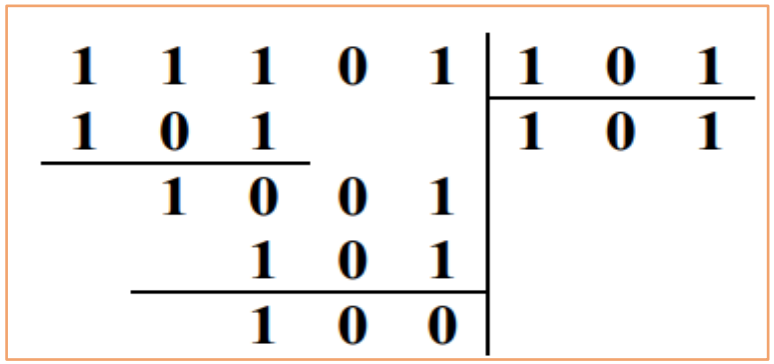
Codage des entiers naturels

Opérations binaires

Opération de division binaire

La division entre deux nombres binaires est identique à la division euclidienne.

Exemple



Il suffit en fait de soustraire 101 lorsqu'on le peut, et d'abaisser le chiffre suivant :

$$11101 = 101 \times 101 + 100$$

Codage des entiers naturels

Opérations binaires

Opération de division binaire

Exercices d'application

Effectuer les opérations binaires suivantes :

a. $(11010)_2/(1001)_2$ b. $(1001)_2/(111)_2$ c. $(1001001)_2/(1001)_2$

d. $(10111)_2/(111)_2$

Numérotation

Code binaire réfléchi (code Gray)

Présentation

Dans ce code, un seul bit change de valeur entre deux codages successifs.

	Code binaire Naturel.....	..Code binaire réfléchi
Sur 3 bits	.....000.....	..000
	001.....	..001
	010.....	..011
	011.....	..010
	100.....	..110
	101.....	..111
	110.....	..101
	111.....	..100

Numérotation

Code binaire réfléchi (code Gray)

Présentation

Pour trouver l'expression d'un nombre binaire dans le code réfléchi, on l'additionne sans effectuer la retenue, avec le nombre obtenu en le décalant vers la gauche d'un rang et on abandonne le chiffre du plus petit poids.

Exemple

```
   1 0 1 1
+  1 0 1 1
  ---------
   1 1 1 0 1     → 1110 en code réfléchi correspond à (11)10
```

Exercices d'application

Trouver l'expression du nombre correspondante en binaire naturel :

a. $(1000)_{Gray}$ b. $(1101)_{Gray}$ c. $(1111)_{Gray}$ d. $(1001)_{Gray}$

Codage des entiers relatifs

Présentation

Il existe au moins trois façons pour coder en binaire un entier portant un signe (+ ou -) :

- Code binaire signé (par signe et valeur absolue).
- Code complément à 1.
- Code complément à 2 ou complément vrai (Utilisé sur ordinateur).

Numérotation

Codage des entiers relatifs

Code Binaire signé

Le bit le plus significatif est utilisé pour représenter le signe du nombre :

Si le bit le plus fort = 1 alors nombre négatif.
Si le bit le plus fort = 0 alors nombre positif.

Les autres bits codent la valeur absolue du nombre.

Exemple
Sur 8 bits, codage des nombres -24 et -128 en (bs)
-24 est codé en binaire signé par : $10011000_{(bs)}$.
-128 hors limite nécessite 9 bits au minimum.

Etendu de codage
Avec n bits, on code tous les nombres entre : $-(2^{n-1}-1)$ et $(2^{n-1}-1)$
Avec 4 bits : -7 et +7.

Codage des entiers relatifs

Code Binaire signé

Limitations du binaire signé

Deux représentations du zéro : + 0 et – 0

Sur 4 bits : +0 = $0000_{(bs)}$, -0 = $1000_{(bs)}$

Multiplication et l'addition sont moins évidentes.

Exercices d'application

1) Coder $(100)_{10}$ et $(-100)_{10}$ en binaire signé sur 8 bits

2) Décoder en décimal $(11000111)_{(bs)}$ et $(00001111)_{(bs)}$

Numérotation

Codage des entiers relatifs

Code Complément à 1

- Ce code est appelé aussi Complément Logique (CL) ou Complément Restreint (CR). Dans ce code :
- Les nombres positifs sont codés de la même façon qu'en binaire pure.
- Un nombre négatif est codé en inversant chaque bit de la représentation de sa valeur absolue.
- Et tout comme le binaire signé, le bit le plus significatif est utilisé pour représenter le signe du nombre :
  Si le bit le plus fort = 1 alors nombre négatif.
  Si le bit le plus fort = 0 alors nombre positif.

Exemple

-24 en complément à 1 sur 8 bits :

|-24|en binaire pur **$00011000_{(2)}$;** puis on inverse les bits **$11100111_{(cà1)}$**

Numérotation

Codage des entiers relatifs

Code Complément à 1

Limitation du complément à 1

Deux codages différents pour 0 (+0 et -0).

Sur 8 bits : $+0=00000000_{(cà1)}$ et $-0=11111111_{(cà1)}$.

Multiplication et l'addition sont moins évidentes.

Exercices d'application

1) Coder $(217)_{10}$ et $(-198)_{10}$ en complément à 1 sur 16 bits

2) Décoder en décimal $(11001111)_{(cà1)}$ et $(01001111)_{(cà1)}$

Codage des entiers relatifs

Code Complément à 2

Ce code est connu également sous le nom de Complément Vrai (CV). Les nombres positifs sont codés de la même manière qu'en binaire pure ; tandis qu'un nombre négatif est codé en ajoutant la valeur 1 à son complément à 1. Le bit le plus significatif est utilisé pour représenter le signe du nombre.

Exemple

-24 en complément à 2 sur 8 bits :

24 est codé par $00011000_{(2)}$.

-24 ➔ $11100111_{(cà1)}$.

-24 est codé par $11101000_{(cà2)}$.

Numérotation

Codage des entiers relatifs

Code Complément à 2

Avantage du complément à 2

Un seul codage pour 0

Par exemple sur 8 bits :

+0 est codé par $00000000_{(cà2)}$
-0 est codé par $11111111_{(cà1)}$
-0 sera représenté par $00000000_{(cà2)}$

Exercices d'application

1) Coder $(217)_{10}$ et $(-198)_{10}$ en complément à 2 sur 8 bits

2) Décoder en décimal $(11001110)_{(cà2)}$ et $(01001001)_{(cà2)}$

Codage des nombres réels

Présentation

Les formats de représentations des nombres réels sont :

- Format virgule fixe.
- Format virgule flottante.

Codage des nombres réels

Représentation des nombres en virgule fixe

C'est la représentation qui était utilisé par les premières machines. Elle possède une partie 'entière' et une partie 'décimale' séparée par une virgule.

La position de la virgule est fixe d'où le nom.

Exemple

$54,25_{(10)}$
$10,001_{(2)}$
$A1,F0B_{(16)}$

Codage des nombres réels

Représentation des nombres en virgule flottante

Le codage en complément à deux sur n bits ne permet de représenter qu'un intervalle de 2^n valeurs. Pour un grand nombre d'applications, cet intervalle de valeurs est trop restreint. La représentation à virgule flottante a été introduite pour répondre à ce besoin.

Exemple

Pour des mots de 32 bits : la représentation en complément à deux permet de coder un intervalle de 2^{32} valeurs, tandis que la représentation à virgule flottante permet de coder un intervalle d'environ 2^{255} valeurs.

Une représentation de nombres à virgule flottante se présente comme suit :

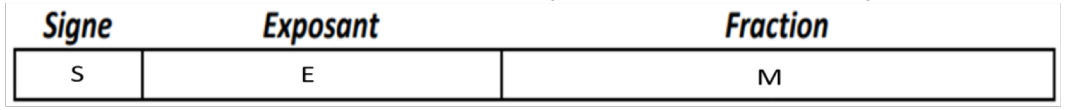
$0{,}5425 \cdot 10^{2}_{(10)}$
$10{,}1 \cdot 2^{-1}_{(2)}$
$A0{,}B4.16^{-2}_{(16)}$

Numérotation

Codage des nombres réels

Représentation des nombres en virgule flottante

La représentation en virgule flottante a été normalisée (norme IEEE 754) sous la forme suivante :



Écriture d'un nombre a virgule flottante

Un nombre en virgule flottante s'écrit sous la forme suivante :

$$x = \pm 1, M \,.\, 2^{Eb}$$

Cette forme est composé de :

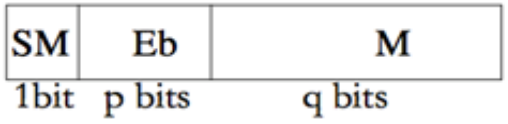
Bit de signe qui est codé sur un bit ayant le poids fort :

- Le signe - : bit 1.
- Le signe + : bit 0.

Exposant biaisé (Eb), qui est placé avant la mantisse pour simplifier la comparaison, codé sur p bits et biaisé pour être positif.

Mantisse normalisée (M) :

- Codé sur q bits et normalisé (virgule est placé après le bit 1 ayant le poids fort).
- Exemple : 11,01 → 1,101 donc M = 101



Numérotation

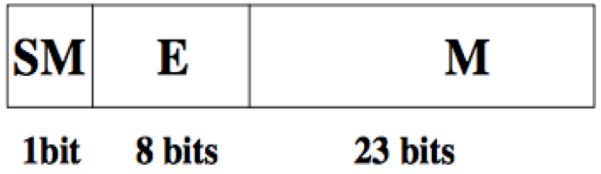
Codage des nombres réels

Représentation des nombres en virgule flottante

Il existe deux types de machines dont la représentation de la virgule flottante a été normalisée :

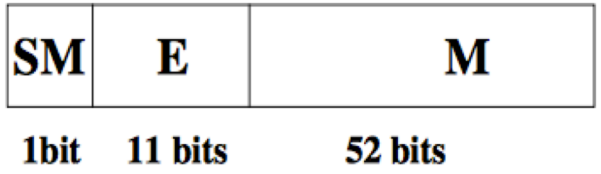
Simple précision sur 32 bits :

- 1 bit de signe de la mantisse
- 8 bits pour l'exposant
- 23 bits pour la mantisse



Double précision sur 64 bits :

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



Numérotation

Codage des nombres réels

Représentation des nombres en virgule flottante

Exemples de calcul

$35{,}5_{(10)} = ?_{\text{(IEEE 754 simple précision)}}$

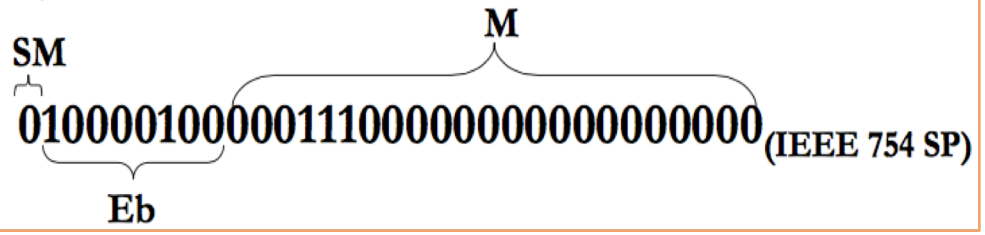
Nombre positif, donc SM = 0

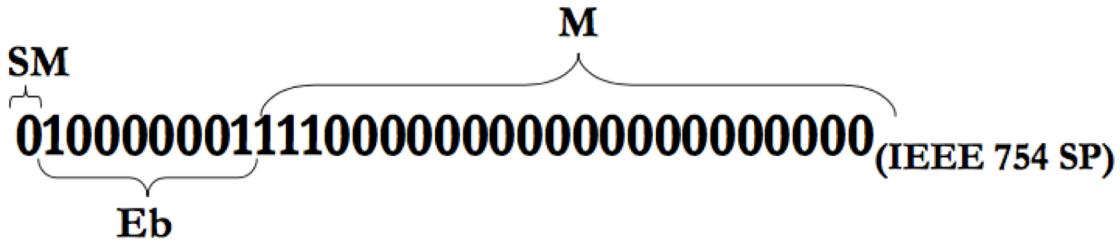
$35{,}5_{(10)} = 100011{,}1_{(2)}$ (virgule fixe)

$= 1{,}000111 \cdot 2^{5}{}_{(2)}$ (virgule flottante)

Exposant = Eb-127 = 5, donc Eb = 132

1,M = 1,000111 donc M = 00011100...





S = 0, donc nombre positif

Eb = 129, donc exposant = Eb-127 = 2

1,M = 1,111

$+ 1{,}111 \cdot 2^{2}{}_{(2)} = 111{,}1_{(2)} = 7{,}5_{(10)}$

Codage des nombres réels

Représentation des nombres en virgule flottante

Exercices d'application

1) Convertir les nombres décimaux suivant en binaire suivant le codage IEEE 754 (en virgule flottante simple précision) :

a. -143 b. -27,56 c. +135,625 d. -46,75 e. +512

1) Trouver les nombres décimaux à virgule représenté par les mots suivants :

a. 1 10000100 01010000000000000000000 b. 0 10000101 11010000000000000000000

Travaux dirigés

2

Architecture matérielle interne des machines