Chapitre 2 : Représentation de l'information dans la machine

1. **Introduction**
2. **Le codage binaire**
3. **Représentation des caractères**
4. **Représentation des nombres**
   - **Nombres entiers:**
     - Signe / valeur absolue
     - Complément à 1
     - Complément à 2
   - **Nombres réels (Fractionnaires)**
     - Virgule fixe
     - Virgule flottante (norme IEEE 754)

1

Introduction

- Les informations traitées par un ordinateur peuvent être de différents types (texte, nombres, etc.) mais elles sont toujours représentées et manipulées par l'ordinateur sous forme binaire.
- Toute information sera traitée comme une suite de **0** et de **1**. Le codage d'une information consiste à établir une correspondance entre la représentation externe (habituelle) de l'information et sa représentation interne dans la machine, qui est une suite de bits. On utilise la représentation binaire car elle est simple et facile à réaliser.

2

1. Le codage binaire

a) Le codage binaire pur.

b) Le code binaire réfléchi (Code de Gray)

c) Le code BCD (Décimal codé binaire)

d) Le code excède de trois.

3

a) Le codage binaire pur

- Le binaire pur est aussi qualifié de binaire naturel. Ce codage a déjà été abordé dans le cadre du chapitre 1 traitant des systèmes de numération.
- En effet, dans ce codage on associe à chaque entier positif la valeur qui lui correspond selon le système de numération binaire. Ainsi, en ayant ***n bits*** on pourra coder les valeurs comprises entre **[0 et 2^n[.**

4

a) Le codage binaire pur

Exemple :

- Sur **6 bits** : $(35)_{10} = (100011)_2$
- Attention la valeur **$(35)_{10}$** n'est pas représentable sur **5 bits.** En effet, rappelez-vous que pour coder une valeur sur **n bits,** elle doit être comprise entre **[0 et 2^n[,** dans notre cas (5 bits) entre **[0 et 2^5[= [0 et 32[.**

5

b) Le code binaire réfléchi (Code de Gray)

- Dans le code binaire pur le passage d'une combinaison à l'autre entraîne parfois le changement simultané de plusieurs bits. C'est par exemple le cas pour la transition de l'équivalent décimal **3** à l'équivalent décimal **4** pour laquelle les bits de poids 1 et 2 passent de 1 à 0 et le bit de poids 3 passe de 0 à 1. **(Le passage de 011 à 100 implique la modification de 3 bits)**

6

b) Le code binaire réfléchi (Code de Gray)

- **Pour éviter cet inconvénient, il est nécessaire d'imaginer des codes pour lesquels le passage d'une combinaison à la suivante n'implique que la modification d'un bit et d'un seul (les transitions s'effectuent sans ambiguïté(غموض) ,éliminant les risques d'aléas(مخاطر)). De tels codes sont appelés "codes réfléchis". Parmi ceux-ci, le code de Gray est le plus employé.**

7

b) Le code binaire réfléchi (Code de Gray)

- **Tout comme le binaire naturel, le binaire réfléchi peut coder n'importe quel nombre entier naturel.**
- **Note : Un code réfléchi est un code non pondéré (غير متوازن) ne peut être utilisé pour les opérations arithmétiques.**

8

b) Le code binaire réfléchi (Code de Gray)

- **Le lien entre un code n codé en binaire réfléchi (code de Gray) et un code N codé en binaire pur est le suivant :**

$$n = \frac{N \oplus 2N}{2}$$

9

b) Le code binaire réfléchi (Code de Gray)

Il existe d'autres méthodes pour calculer code de gray

- Le premier Bit à gauche reste le même,
- Puis, De gauche à droite faire la somme des bits adjacents sans retenue
- $(10010)_2=(11011)_{Gray}$

10

b) Le code binaire réfléchi (Code de Gray)

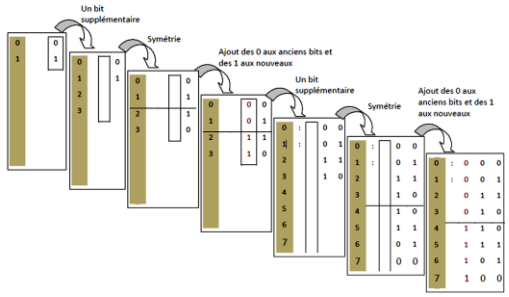
- **Il y a une autre méthode pour construire le code de gray :**

1. On établi un code de départ : zéro est codé 0 et un est codé 1.
2. Puis, à chaque fois qu'on a besoin d'un bit supplémentaire, on symétrise les nombres déjà obtenus (comme une réflexion dans un miroir)
3. On rajoute un 1 au début des nouveaux nombres et un zéro sur les anciens.

11

b) Le code binaire réfléchi (Code de Gray)

Illustration :



12

b) Le code binaire réfléchi (Code de Gray)

Exercice:

- $(22)_{10}=(?)_{Gray}$
- $(18)_{10}=(?)_{Gray}$
- $(101010)_2=(?)_{Gray}$
- Donner en code de Gray la représentation des nombres décimaux inférieurs à 16

13

b) Le code binaire réfléchi (Code de GRAY)

Tableau de conversion Décimal ⟶ Code Gray

Décimal	Code binaire pur	Code Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

14

c) Le code BCD (Binary Coded Decimal)

- Pour passer du décimal au binaire , il faut effectuer des divisions successives. Il existe d'autres méthode simplifiée pour le passage du décimal au binaire (le code BCD, EXCESS 3,…..) .
- Le BCD est le code le plus utilisé. son principe consiste à faire des éclatement sur 4 bits et de remplacer chaque chiffre décimal par sa valeur binaire correspondante.
- Les combinaisons supérieures à 9 sont interdites

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

15

c) Le code BCD (Binary Coded Decimal)

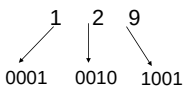
Exemple : Le nombre 512 en décimal équivaut à $(1000000000)_2$.
En code BCD ce nombre sera codé comme suit $(0101\ 0001\ 0010)_{BCD}$,la conversion de code est établie comme suit:

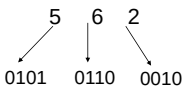
- Au premier chiffre (qui est 2) correspond sur quatre bits la valeur $(0010)_2$
- Au second chiffre (qui est 1) correspond sur quatre bits la valeur $(0001)_2$
- Au dernier chiffre (qui est 5) correspond sur quatre bits la valeur $(0101)_2$
- Le code BCD du nombre **512** sera la concaténation dans le même ordre des chiffres décimaux de leur correspondants en binaire, c'est-à-dire **$(0101\ 0001\ 0010)_{BCD}$**

16

c) Le code BCD (Binary Coded Decimal)

Exemple 2:





$129 = (0001\ 0010\ 1001)_{BCD}$ $562 = (0101\ 0110\ 0010)_{BCD}$

17

c) Le code BCD (Binary Coded Decimal)

Remarque

Dans le code BCD :

- Pour l'addition, s'il y a une retenue ou le résultat obtenu n'appartenant pas au BCD (le résultat est >9) on doit y ajouter $6 = (0110)_2$

18

c) Le code BCD (Binary Coded Decimal)

Exemples:

```
    16        0001 0110
+   25        0010 0101
=             0011 1011   > 9
        +      0110       +6
              0100 0001
```

```
137=  0001 | 0011 | 0111
+ 99 + 0000 | 1001 | 1001
       0001 | 1101 | 0000
            + 0110 | 0110
       0010 | 0011 | 0110
=        2    3    6
```

19

c) Le code BCD (Binary Coded Decimal)

Exercice:

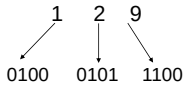
- Convertir les nombres décimaux 19 et 21 en BCD puis faire la somme

20

d) Le codage EXCESS3 (BCD+3)

Chaque chiffre décimal est codé séparément en son équivalent binaire + 3.

Tableau de conversion Décimal vers Excess-3	
Décimal	Excess-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100



21

d) Le codage EXCESS3 (BCD+3)

Remarque

Dans le code Excess3 :

- Pour l'addition, s'il y a une retenue on doit y ajouter 3 $(0011)_2$ si non on soustrait 3 $(- 0011)_2$

22

d) Le codage EXCESS3 (BCD+3)

Exemples :

```
45  =  0011 | 0111 | 1000
+90  + 0011 | 1100 | 0011
       0111 | 0011 | 1011
      -0011 | +0011 | -0011
     = 0100 | 0110 | 1000
     =   1    3      5
```

```
137 =  0100 | 0110 | 1010
+ 90  + 0011 | 1100 | 1100
        1000 | 0011 | 0110
       -0011 | +0011 | +0011
      = 0101 | 0110 | 1001
      =   2    3      6
```

23

d) Le codage EXCESS3 (BCD+3)

Exercice :

- Convertir les nombres décimaux 19 et 21 en Excédent-3 puis faire la somme.
- Effectuer en BCD puis en Excédent-3 l'opération suivante : $65_{(8)} + 23_{(8)}$

24

2. La représentation des caractères

a) Introduction

b) Le code ASCII.

c) Le code EBCDIC

d) Le code Unicode.

25

2. La représentation des caractères

a) Introduction:

- Les informations que doivent traiter les ordinateurs sont composées de nombres, lettres, chiffres ou symboles particuliers.
- On doit représenter l'information a traiter de manière a ce qu'elle puisse être utilisable par la machine.

26

2. La représentation des caractères

- Pour cela, il faut un code capable de représenter les 26 combinaisons, qui correspondent aux lettres, plus les 10 combinaisons, qui correspondent aux chiffres, soit 36 combinaisons différentes, ce qui implique un code compose au minimum de 6 bits.
- Le codage le plus utilisé est le ASCII (American Standard Code for Information Interchange)

27

2. La représentation des caractères

b) Le code ASCII:

- C'est le système de codage quasi universel. C'est un code a 7 positions, le 8ieme bit étant réserve au bit de parité ce qui fait 2^7 = 128 caractères représentables. Ce code comprend :
  - Les Chiffres,
  - Les Majuscules,
  - Les Minuscules,
  - Quelques Symboles Usuels En Informatique ($,@,*,..),
  - Des Fonctions De Commandes (Tabulations, Retour Chariot, Sonnette..),
  - Des Symboles De Ponctuations (!,",},...).

28

2. La représentation des caractères

b) Le code ASCII:

- Notons que le code ASCII, défini pour les besoins de l'informatique en langue anglaise (ne permet la représentation des caractères accentués é, è, à, ù, ...), et encore moins des caractères chinois ou arabes. Pour ces langues, d'autres codages existent, utilisant 16 bits par caractères.

29

2. La représentation des caractères

b) Le code ASCII:

		000	001	010	011	100	101	110	111
		0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	10	LF	SUB	*	:	J	Z	j	z
1011	11	VT	ESC	+	;	K	[	k	{
1100	12	FF	FS	,	<	L	\	l	|
1101	13	CR	GS	-	=	M	]	m	}
1110	14	SO	RS	.	>	N	^	n	~
1111	15	SI	US	/	?	O	-	o	DEL

TABLE DES CODES ASCII

NUL	Absence de caractère,blanc,espace
SOH	Start of Heading : début en-tête
STX	Start of Text
ETX	End of Text
EOT	End of Transmission
ENQ	Enquiry Demande
ACK	Acknowledge, accusé réception
BEL	Bell, sonnette
BS	Backspace marche arrière 1 caractère
HT	Horizontale Tabulation
LF	Line Fed retour à la avelle ligne
VT	Vertical Tabulation
FF	Form Fed, passage page suivante
CR	Carriage Return, retour chariot
SO	Shift Out caractère suivant non std
SI	Shift In retour au caractères std
DLE	DataLink Escape chgmt de signific
NAK	Negative Acknoledgment
SYN	Synchronous, caractère de synchro
ETB	End Of Transmission Block
CAN	Cancel annulation de la donnée précédente
SUB	Substitute remplacement
ESC	Escape caractère de ctrl d'extension
FS	File Separator
GS	Groupe Separator
RS	Record Separator
US	United Separator
SP	Space Espace
DEL	Delete, suppression
DC1 à DC4 : caractères de commandes	

30

2. La représentation des caractères

b) Le code ASCII:

Exercice:

- Donner les codes ASCII de D ,d et de mot Machine?

2. La représentation des caractères

c) Le code EBCDIC:

- Le code **EBCDIC** (**E**xtended **B**inary **D**ecimal **I**nterchange **C**ode) créé par IBM est un code à 8 éléments binaires utiles, soit 256 combinaisons possibles.

2. La représentation des caractères

c) Le code EBCDIC:

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX		PT			GE				FF	CR		
1	DLE	SBA	EUA	1C		NL				EM			DUP	SF	FM	ITB
2							ETB	ESC						ENQ		
3			SYN					EOT					RA	NAK		
4	SP										¢	.	<	(	+	
5	&										!	$	*	)	;	¬
6	-	/									¦	,	%	_	>	?
7											:	#	@	'	=	"
8		a	b	c	d	E	f	g	h	I						
9		j	k	l	m	N	o	p	q	r						
A		~	s	t	u	v	w	x	y	z						
B																
C	{	A	B	C	D	E	F	G	H	I						
D	}	J	K	L	M	N	O	P	Q	R						
E	\		S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9						
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Le code EBCDIC Standard

2. La représentation des caractères

c) L' unicode:

• Unicode est une norme informatique développée par le *Consortium Unicode* qui vise à donner à tout caractère de n'importe quel système d'écriture de langue un identifiant numérique unique, et ce de manière unifiée, quelle que soit la plate-forme informatique ou le logiciel.

2. La représentation des caractères

c) L' unicode:

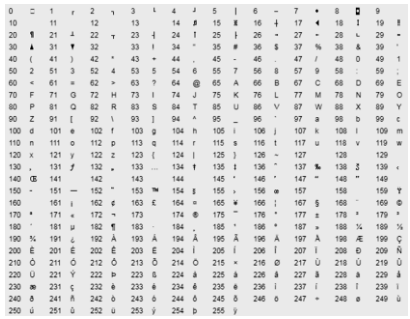
• C'est un code de 32 bits (4 milliards d'entrees possibles mais contient jusqu'à maintenant quelques millions de caractères distincts seulement).

• Chaque caractère est stocké sur 1-5 octets.

• Code multilingues : Amérique, Europe, Afrique, Asie,.. etc.

2. La représentation des caractères

c) Le codage unicode:



3. La représentation des nombres

a) La représentation des nombres entiers
- les entiers non signés
- les entiers signés (positifs ou négatifs)

b) La représentation des nombres réels (fractionnaires)
- Virgule fixe.
- Virgule flottante (norme IEEE 754)

37

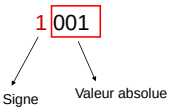
a) Représentation des nombres entiers

- Il existe deux types d'entiers :
  - les entiers non signés (positifs)
  - et les entiers signés (positifs ou négatifs)
- **Problème :** Comment indiquer à la machine qu'un nombre est négatif ou positif ?
- Il existe 3 méthodes pour représenter les nombres négatifs :
  - Signe/ valeur absolue
  - Complément à 1(complément restreint)
  - Complément à 2 (complément à vrai)

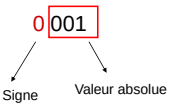
38

a.1 Représentation signe / valeur absolue (S/VA)

- Si on travail sur n bits , alors le bit du poids fort est utilisé pour indiquer le signe :
  - **1** : signe négatif
  - **0** : signe positif
- Les autres bits (n -1) désignent la valeur absolue du nombre.
- Exemple : Si on travail sur 4 bits.



1001 est la représentation de - 1



0001 est la représentation de + 1

39

Sur 3 bits on obtient :

signe	VA	valeur
0	00	+ 0
0	01	+ 1
0	10	+ 2
0	11	+ 3
1	00	- 0
1	01	- 1
1	10	- 2
1	11	- 3

• Les valeurs sont comprises entre -3 et +3

$-3 \leq N \leq +3$

$-(4-1) \leq N \leq +(4-1)$

$-(2^2-1) \leq N \leq +(2^2-1)$

$-(2^{(3-1)}-1) \leq N \leq +(2^{(3-1)}-1)$

Si on travail sur **n** bits , l'intervalle des valeurs qu'on peut représenter en S/VA :

$$-(2^{(n-1)}-1) \leq N \leq +(2^{(n-1)}-1)$$

40

Avantages et inconvénients de la représentation signe/valeur absolue

- C'est une représentation assez simple .
- On remarque que le zéro possède deux représentations +0 et -0 ce qui conduit à des difficultés au niveau des opérations arithmétiques.
- Pour les opérations arithmétiques il nous faut deux circuits : l'un pour l'addition et le deuxième pour la soustraction .

L'idéal est d'utiliser un seul circuit pour faire les deux opérations, puisque $a - b = a + (-b)$

41

a.2 Représentation en complément à un (complément restreint)

- On appel complément à un d'un nombre N un autre nombre N' tel que :

$$N+N'=2^n-1$$

n : est le nombre de bits de la représentation du nombre N .

Exemple :

Soit N=1010 sur 4 bits donc son complément à un de N :

$$N' = (2^4 - 1) - N$$

$N' = (16-1) - (1010)_2 = (15) - (1010)_2 = (1111)_2 - (1010)_2 = 0101$

$$\begin{array}{r} 1\ 0\ 1\ 0 \\ +\ 0\ 1\ 0\ 1 \\ \hline 1\ 1\ 1\ 1 \end{array}$$

42

Remarque 1 :

- Pour trouver le complément à un d'un nombre, il suffit d'inverser tous les bits de ce nombre : si le bit est un 0 mettre à sa place un 1 et si c'est un 1 mettre à sa place un 0 .
- Exemple :

Sur 4 Bits

1 0 1 0
↓ ↓ ↓ ↓
0 1 0 1

Sur 5 Bits

0 1 0 1 0
↓ ↓ ↓ ↓ ↓
1 0 1 0 1

Remarque 2

- Dans cette représentation , le bit du poids fort nous indique le signe (0 : positif , 1 : négatif).
- Le complément à un du complément à un d'un nombre est égale au nombre lui même .

CA1(CA1(N))= N

- Exemple :

  Quelle est la valeur décimale représentée par la valeur 101010 en complément à 1 sur 6 bits ?
- Le bit poids fort indique qu'il s'agit d'un nombre négatif.
- Valeur = - CA1(101010)

  $= - (010101)_2 = - (21)_{10}$

Si on travail sur 3 bits :

Valeur en CA1	Valeur en binaire	Valeur décimal
000	000	+ 0
001	001	+ 1
010	010	+ 2
011	011	+ 3
100	-011	- 3
101	-010	- 2
110	-001	- 1
111	-000	- 0

•Dans cette représentation , le bit du poids fort nous indique le signe .

•On remarque que dans cette représentation le zéro possède aussi une double représentation (+0 et – 0) .

• Sur 3 bits on remarque que les valeurs sont comprises entre -3 et +3

$$-3 \le N \le +3$$
$$-(4-1) \le N \le +(4-1)$$
$$-(2^2-1) \le N \le +(2^2-1)$$
$$-(2^{(3-1)}-1) \le N \le +(2^{(3-1)}-1)$$

Si on travail sur n bits , l'intervalle des valeurs qu'on peut représenter en CA1 :

$$-(2^{(n-1)}-1) \le N \le +(2^{(n-1)}-1)$$

a.3 Complément à 2 (complément à vrai)

- Si on suppose que **a** est un nombre sur **n** bits alors :

$$a + 2^n = a \text{ modulo } 2^n$$

et si on prend le résultat sur n bits on va obtenir la même valeur que a .

$$\boxed{a + 2^n = a}$$

Exemple : soit a = 1001 sur 4 bits

$2^4 = 10000$

```
    1 0 0 1
+   1 0 0 0 0
-----------
  1 1 0 0 1
```

Si on prend le résultat sur 4 bits on trouve la même valeur de a = 1001

•Si on prend deux nombres entiers **a** et **b** sur **n** bits , on remarque que la soustraction peut être ramener à une addition : **a – b = a + (-b)**

• Pour cela il suffit de trouver une valeur équivalente à **-b** ?

$$a - b = a + 2^n - b = a + (2^n - 1) - b + 1$$

On a $b + CA1(b) = 2^n - 1$ donc $CA1(b) = (2^n - 1) - b$

Si on remplace dans la première équation on obtient :

$$a - b = a + CA1(b) + 1$$

La valeur CA1(b)+1 s'appelle le complément à deux de b :

CA1(b)+1 = CA2(b)

Et enfin on va obtenir : a - b = a+ CA2(b) → transformer la soustraction en une addition .

Exemple

- Trouver le complément à vrai de : 01000101 sur 8 bits ?

CA2(01000101)= CA1(01000101)+ 1
CA1(01000101)= (10111010)
CA2(01000101)=(10111010)+ 1 = (10111011)

- **Remarque 1 :**

Pour trouver le compétemment à 2 d'un nombre : il faut parcourir les bits de ce nombre à partir du poids faible et garder tous les bits avant le premier 1 et inverser les autres bits qui viennent après.

0 1 0 0 0 1 0 1 → 1 0 1 1 1 0 1 1

1 1 0 1 0 1 0 0 → 0 0 1 0 1 1 0 0

Remarque 2

- Dans cette représentation , le bit du poids fort nous indique le signe (0 : positif , 1 : négatif).
- Le complément à deux du complément à deux d'un nombre est égale au nombre lui même .

CA2(CA2(N))= N

- Exemple :

Quelle est la valeur décimale représentée par la valeur 101010 en complément à deux sur 6 bits ?

- Le bit poids fort indique qu'il s'agit d'un nombre négatif.
- Valeur = - CA2(101010)

= - (010101 + 1)

= - $(010110)_2$= - (22)

Si on travail sur 3 bits :

Valeur en CA2	Valeur en binaire	valeur
000	000	+ 0
001	001	+ 1
010	010	+ 2
011	011	+ 3
100	- 100	- 4
101	- 011	- 3
110	- 010	- 2
111	- 001	- 1

•Dans cette représentation , le bit du poids fort nous indique le signe .
•On remarque que le zéro n'a pas une double représentation.

•Sur 3 bits on remarque que les valeurs sont comprises entre -4 et +3

$-4 \leq N \leq +3$
$-4 \leq N \leq +(4-1)$
$-2^2 \leq N \leq +(2^2-1)$
$-2^{(3-1)} \leq N \leq (2^{(3-1)}-1)$

Si on travail sur n bits , l'intervalle des valeurs qu'on peut représenter en CA2 :

$-(2^{(n-1)}) \leq N \leq +(2^{(n-1)}-1)$

La représentation en complément à deux (complément à vrai) est la représentation la plus utilisée pour la représentation des nombres négatifs dans la machine.

Opérations arithmétiques en CA2

Effectuer les opérations suivantes sur 5 Bits , en utilisant la représentation en CA2

+ 9 + 4 + 13 : 0 1 0 0 1 + 0 0 1 0 0 = 0 1 1 0 1

Le résultat est positif

$(01101)_2$= (13)$_{10}$

+ 9 - 4 + 5 : 0 1 0 0 1 + 1 1 1 0 0 = 1 0 0 1 0 1

Report

Le résultat est positif

$(00101)_2$= (5)$_{10}$

- 9 - 4 - 13 : 1 0 1 1 1 + 1 1 1 0 0 = 1 1 0 0 1 1

Report

Le résultat est négatif :

Résultat = - CA2 (10011)= -(01101)

= - 13

- 9 + 9 + 0 : 1 0 1 1 1 + 0 1 0 0 1 = 1 0 0 0 0 0

Report

Le résultat est positif

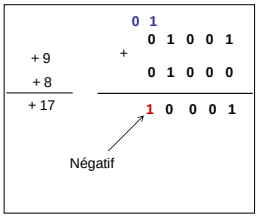
$(00000)_2$= (0)$_{10}$

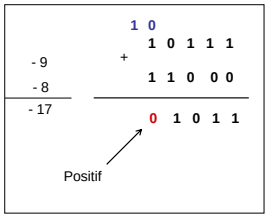
La retenue et le débordement

- On dit qu'il y a une retenue si une opération arithmétique génère un report .
- On dit qu'il y a un débordement (Over Flow) ou dépassement de capacité: si le résultat de l'opération sur n bits et faux .
  - Le nombre de bits utilisés est insuffisant pour contenir le résultat
  - Autrement dit le résultat dépasse l'intervalle des valeurs sur les n bits utilisés.

55

Cas de débordement





•Nous avons un débordement si la somme de deux nombres positifs donne un nombre négatif .
•Ou la somme de deux nombres négatifs donne un Nombre positif
•Il y a jamais un débordement si les deux nombres sont de signes différents.

56

b. La représentation des nombres réels

- Un nombre réel est constitué de deux parties : la partie entière et la partie fractionnelle (les deux parties sont séparées par une virgule)
- Problème : comment indiquer à la machine la position de la virgule ?
- Il existe deux méthodes pour représenter les nombre réel :
  - Virgule fixe : la position de la virgule est fixe
  - Virgule flottante : la position de la virgule change (dynamique)

57

b.1 La virgule fixe

- Dans cette représentation la partie entière est représentée sur n bits et la partie fractionnelle sur p bits , en plus un bit est utilisé pour le signe.
- Exemple : si n=3 et p=2 on va avoir les valeurs suivantes

Signe	P.Entière	P.Fractionnel	valeur
0	000	00	+ 0,0
0	000	01	+ 0,25
0	000	10	+ 0,5
0	000	11	+ 0,75
0	001	00	+ 1,0
.	.	.	.
.	.	.	.

Dans cette représentation les valeurs sont limitées et nous n'avons pas une grande précision

58

b.2 Représentation en virgule flottante

- Chaque nombre réel peut s'écrire de la façon suivante :

$$N = \pm M \times b^{e}$$

  - M : mantisse ,
  - b : la base , النظام
  - e : l'exposant الأس

- **Exemple :**

  15,6 = 1,56 x 10^{+1}

  -8235,67 = -8,23567x10^{+3}

  - (110,101)$_2$ = - (1,10101)$_2$ x 2^{+2}

  (0,00101)$_2$= (1,01)$_2$ x 2^{-3}

59

b.2 Représentation en virgule flottante

Avantage:

permet de représenter des nombres très grands et très petits sans s'encombrer de zéros.

تسمح بتمثيل أعداد كبيرة وأعداد صغيرة جدا

Remarque :

on dit que la mantisse **M** (en binaire) est normalisée si le premier chiffre avant la virgule est égale à 1.

نقول أن M طبيعية(كتابة علمية) إذا كان العدد قبل الفاصلة = 1

Exemple: M=1,01101 **est normalisée**

M=0,10101 **est dénormalisée**

60

b.2 Représentation en virgule flottante

La norme IEEE 754

N=S M x 2^{er}

- Le signe (S) + est représenté par 0 et le signe − par 1
- La mantisse M appartient à l'intervalle [1; 2[
- L'exposant (er) est un entier relatif et il est établi de manière à ce que la mantisse soit de la forme « 1,… »

61

b.2 Représentation en virgule flottante

La norme IEEE 754

Il existe plusieurs formats:

- **Simple précision :** 32 bits (soit 4 octets)
  1 bit de signe, 8 bits d'exposant, 23 bits de mantisse
- **Double précision :** 64 bits (soit 8 octets)
  1 bit de signe, 11 bits d'exposant, 52 bits de mantisse
- **Quadruple précision :** 128 bits (soit 16 octets)
  1 bit de signe, 15 bits d'exposant, 112 bits de mantisse

62

b.2 Représentation en virgule flottante

La norme IEEE 754

Simple précision: les caractéristiques

- Exposant Réel (Er): de – 126 à +127 (-126 ≤Er ≤+127)

On effectue la somme Er + 127 afin de coder l'exposant en binaire

- Mantisse: de 1 à (2- 2^{-23}) 1 ≤ M < 2
- Plus petit nombre normalisé: 1,0 x 2^{-126} = 2^{-126}
- Plus grand nombre normalisé: $(1,1111\ldots.)_{(2)}$ x 2^{+127}
- Les exposants biaisés 00000000 et 11111111 sont interdits

Exposant Biaisé = Exposant réel + Biais (127)

63

b.2 Représentation en virgule flottante

La norme IEEE 754

La représentation en simple précision:

S	Eb	F
1 bit	8 bits	23 bits

S= 0 (positif) ou 1 (négatif)

Eb = Er + 127 (2^{8-1} -1)

M = 1,F

64

b.2 Représentation en virgule flottante

La norme IEEE 754

Exemple 1 : Coder N=-6,625

- 6, 625 = $110,101_{(2)}$
- 110, 1010 = $1,10101_{(2)}$ × 2^{+2}

Donc

S=1

Eb=Er+127 = 2+127 = 129 = $10000001_{(2)}$

F=$101010000000000000000000_{(2)}$

1	10000001	10101000000000000000000

- En hexadécimal : $C0D40000_{(16)}$

65

Exemple 2

- On veut représenter les nombres ($0,015)_8$ et -($15, 01)_8$en virgule flottante sur une machine ayant le format suivant :

Signe mantisse	Exposant décalé	Mantisse normalisée
1 bit	8 bits	23 bits

$(0,015)_8$=$(0,000001101)_2$= 1,101 * 2^{-6}

Signe mantisse : positif (0)
Mantisse normalisé : 1,101
Exposant réel = -6
Exposant Biaisé = -6 + 127 = +121 = ($01111001)_2$

0	01111001	1 0 1 0 0 0 0 0 0 0 0 0
1 bit	8 bits	23 bits

66

- $(15,01)_8=(001101,000001)_2= 1,101000001 * 2^3$

Signe mantisse : négatif (1)
Mantisse normalisée : 1,101000001
Exposant réel = + 3
Exposant Biaisé = 3 + 127 = +130 = $(10000010)_2$

1	10000010	10100000100000000…00000
1 bit	8 bits	23 bits

67

Opérations arithmétiques en virgule flottante

- Soit deux nombres réels N1 et N2 tel que
  $N1=M1*2^{e1}$ et $N2=M2*2^{e2}$
- On veut calculer N1+N2 ?
- Deux cas se présentent :
  - Si e1 = e2 alors N3= (M1+M2) 2^{e1}
  - Si e1 <> e2 alors élevé au plus grand exposant et faire l'addition des mantisses et par la suite normalisée la mantisse du résultat. N3= (M1+M2) $2^{\text{le plus grand exposant}}$

68

Exemple

- Effectuer l'opération suivante : $(0,15)_8+ (2,5)_8=(?)$:

$(0,15)_8$= (0,001101) = 1,101 *2^{-3}
$(2,5)_8$= (010,101) = 1,0101 *2^1
$(0,15)_8+ (1,5)_8$ = 1,101 *2^{-3} + 1,0101 *2^1
= 0,0001101 *2^1 + 1,0101 *2^1
= 1,0110101 * 2^1
M=1,0110101
Er=1 ⟶ Eb=1+127=128=$10000000_{(2)}$

1 bit	8 bits	23 bits
0	10000000	0110101000….000

69

Résumé des Différentes représentations possibles sur le standard ANSI / IEEE 754 (simple précision)

Signe	E_b	f	M	Valeur représentée
1	11111111 = 255	= 0	-	$-\infty$
0			-	$+\infty$
∀	11111111 = 255	≠ 0	-	NaNs
1	00000000 = 0	= 0	0.000….0	−0
0				+0
1	$0 < E_b < 255$ Normalisées	∀f	$M = 1.f$	$V = -M \times 2^{E_b-127}$
0				$V = +M \times 2^{E_b-127}$
1	00000000 = 0	≠ 0	$M = 0.f$	$V = -M \times 2^{-126}$
0	Dénormalisées			$V = +M \times 2^{-126}$

70

Exercice

Soit $X = -5_{(10)} \times 2^{-2}$ et $Y = 16.625_{(10)} \times 2^{-132}$

1- Représenter X et Y en virgule flottante dans le standard ANSI / IEEE-754 simple précision (32 bits) ..... **(02.5 Pts)**

2- La représentation du nombre réel Z sur le même standard est en hexadécimal comme suit

$80800000_{(16)}$

- Donner, sous la forme de $\mp a \times 2^b$, la valeur de Z. (a et b sont décimaux) .... **(01.5 Pts)**

71