Informatique Embarquée

Chap 0 : Introduction

Chap 1 : Résumé : Représentation des nombres binaires

S2

2020 – 2021

IUT A GEII Toulouse3

A. Nketsa

Introduction

Définition :

Système embarqué : c'est un système électronique et informatique autonome construit pour effectuer des tâches précises**.**

L'informatique embarquée ou **informatique industrielle** consiste à utiliser un ordinateur pour commander des dispositifs.

En général, ces ordinateurs sont spécialisés et sont constitués des composants suivants :

- Microcontrôleur et/ou microprocesseur (Chef d'orchestre)
- des périphériques :
  * Mémoires
  * Entrées (par exemple des capteurs)
  * Sorties (Par exemple des moteurs)

Informatique embarquée et informatique industrielle utilisent les mêmes techniques

Pour construire ce genre de système, on a besoin des compétences :

- en Electronique numérique pour construire la carte électronique et certains composants.

Par exemple : on doit utiliser un circuit imprimé permettant de faire les connexions entre microcontrôleur- mémoire – entrées – sorties – partie puissance.

- en programmation pour les systèmes embarqués

- en électronique de puissance pour l'énergie

- en électronique analogique pour le conditionnement des capteurs

Les systèmes embarqués sont partout

Quelques domaines d'application

Transport

-Aéronautique :

Avion - Hélicoptère - Drone - Pilotage automatique – contrôle de navigation

- Maritime :

Navire (Pilotage automatique)

- Ferroviaire :

TGV - Metro

- Automobile :

Véhicule autonome - Assistant de conduite (GPS - ABS - Airbag – etc)

Robotique

Santé

Appareils de mesure, d'exploration et d'intervention

Equipement courant :

Téléphone portable – imprimante – copieurs – console de jeu – télévision

Equipement dans le bâtiment :

Ascenseurs – contrôle d'accès – système de surveillance, éclairage automatique

Equipement de production

Communication : Satellites

Résumé : Représentation des nombres binaires

1- Bases de numération

Tout nombre entier naturel peut être exprimé dans une base B.
Soit N un nombre entier ≥ 0**,**
N se décompose dans la base B sous la forme

$$\mathbf{N = a_{n-1}*B^{n-1} + \quad + a_0*B^0} \qquad \textbf{noté} \quad \mathbf{N = (a_{n-1} \qquad a_0)_B}$$

B est la base et **ai** ∈ [0, B-1] ai sont appelés les symboles ou digits

Exemple:

$$\mathbf{259 = 2*10^2 + 5*10^1 + 9*10^0}$$

2-Bases usuelles en informatique embarquée

Base décimale (ou encore base 10)
B = 10 et ai ∈ [0, 9] ai sont les symboles aussi appelés chiffres

Base binaire (ou encore base 2)
B = 2 et ai ∈ [0, 1] ai sont les symboles aussi appelés bits

Base hexadécimale (ou encore base 16)
B = 16 ai ∈ [0, 16-1] les ai sont appelés les symboles ou digits

Comme un symbole ne doit comporter qu'un seul élément,

Par convention, dans la base hexadécimale :
10 est noté A, 11 est noté B, 12 est noté C, 13 est noté D, 14 est noté E , 15 est noté F

d'où ai ∈ [0, F]

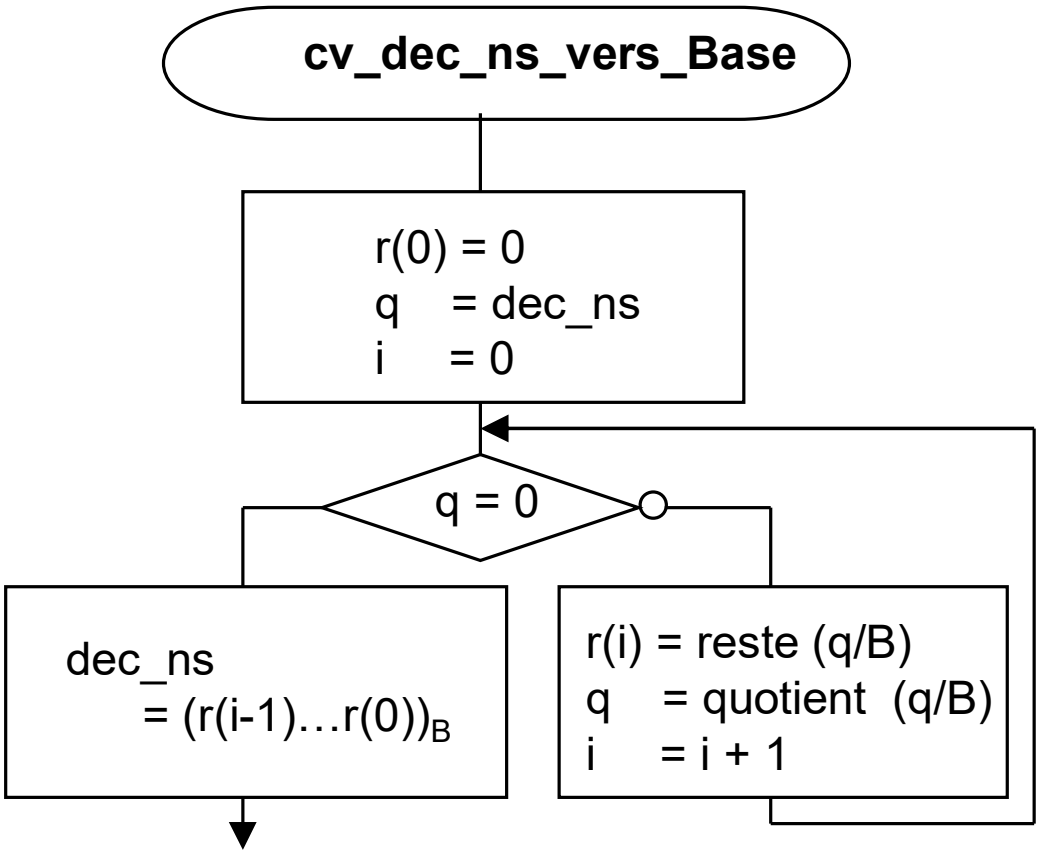
B = 16 et ai ∈ [0, 9, A, B, C, D, E, F] ai sont les symboles aussi appelés symboles hexadécimaux

3- Changement de base

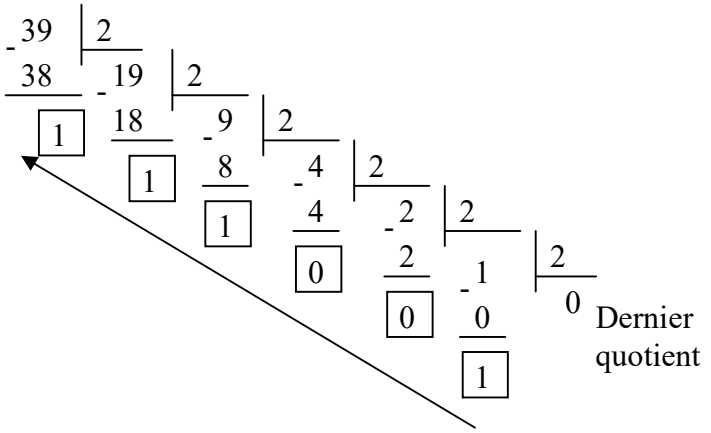
3-1 Décimal (base 10) vers base B ⇒ méthode par divisions successives

Changement de base
nombre décimal vers nombre en Base B

Le nombre décimal à convertir est dec_ns



Exemple convertir 39 en binaire

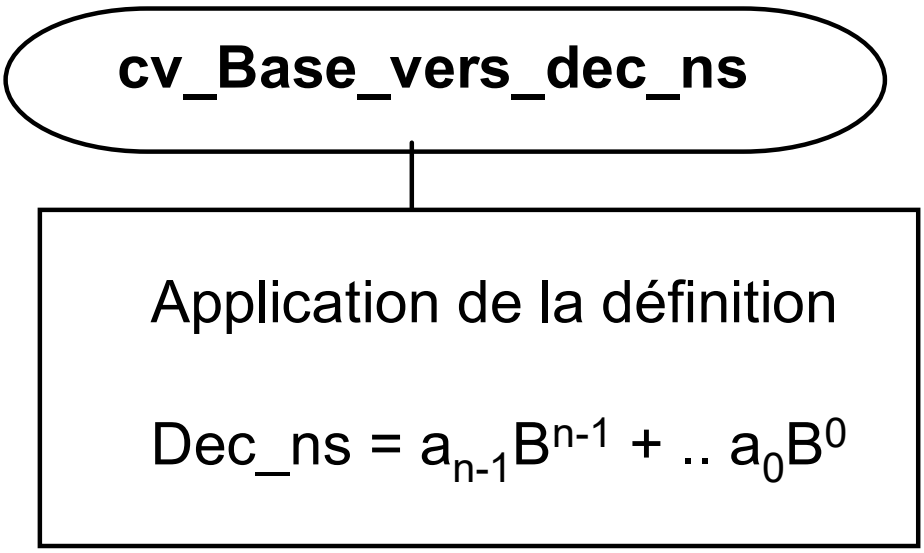


$39 = (100111)_2$

3-2 Base B vers décimal ⇒ application de la définition d'un nombre

Changement de base
nombre en Base B vers décimal

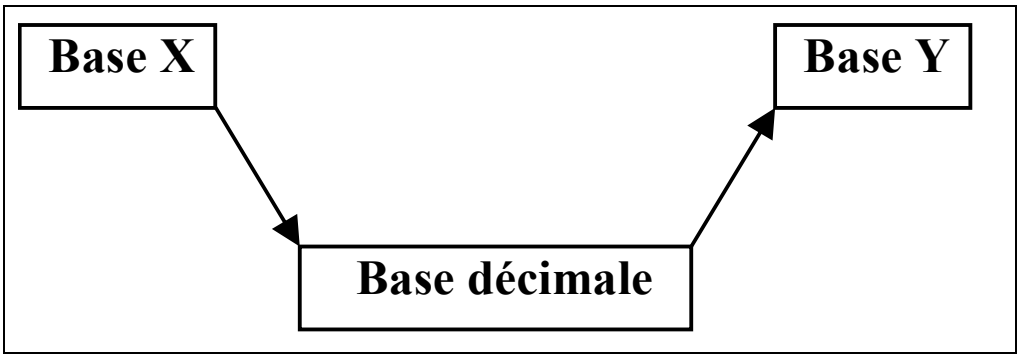
$N = (a_{n-1} \quad a_0)_B$



Exemples

$(100111)_2 = 1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = (39)_{10}$

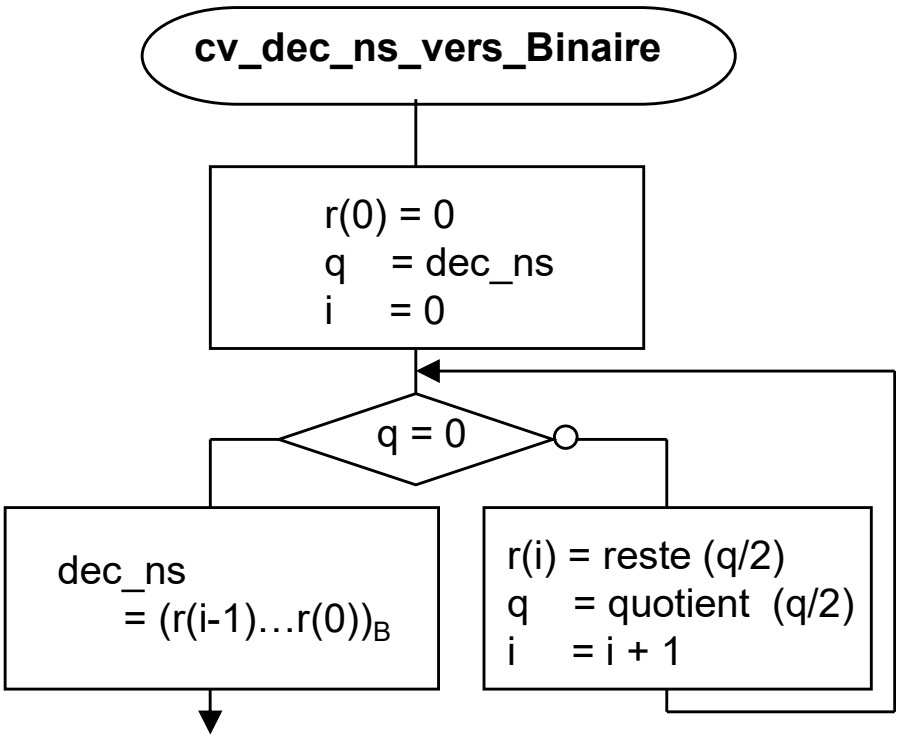
3-3 Approche générale : Changement de Base X vers base Y



3-4 Changement de bases usuelles et cas particuliers

a) Base 2 : Décimal (base 10) vers binaire (base 2) ⇒ divisions successives

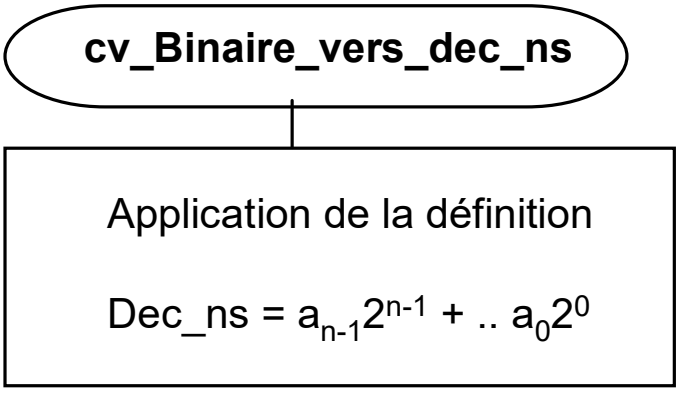
Changement de base
Nombre en Base 10 vers nombre en Base 2



Exemple déjà traité

b) Binaire (Base 2) vers décimal ⇒ définition d'un nombre

Changement de base
nombre en Base 2 vers nombre en base 10

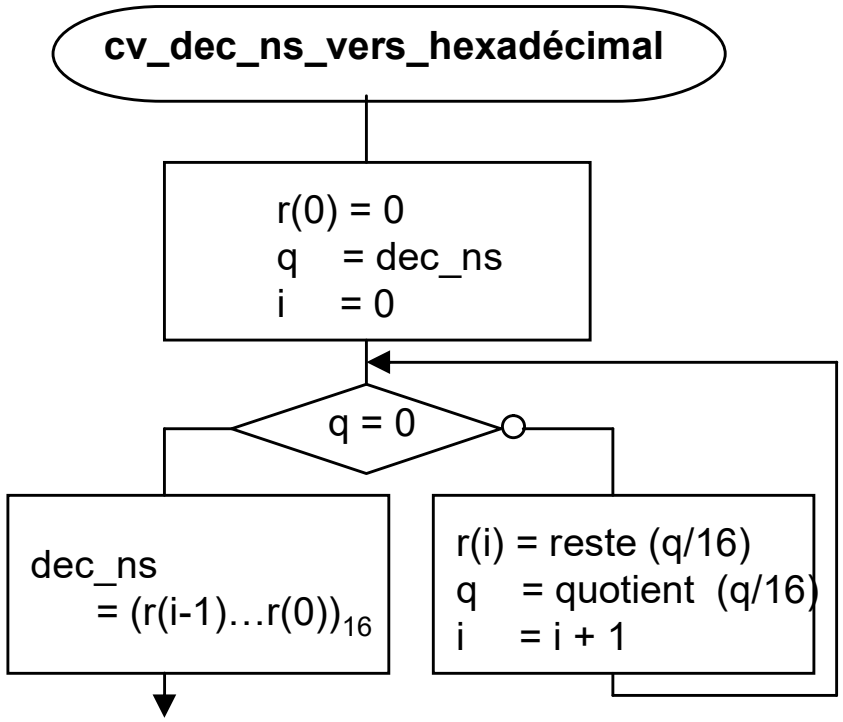


Exemple déjà traité

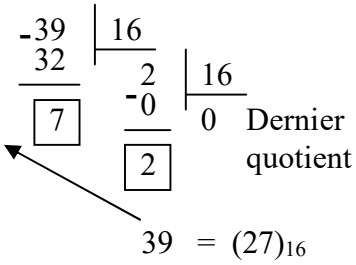
c) Base hexadécimale (base 16) :

Décimal (base 10) vers hexadécimal (base 16) ⇒ divisions successives

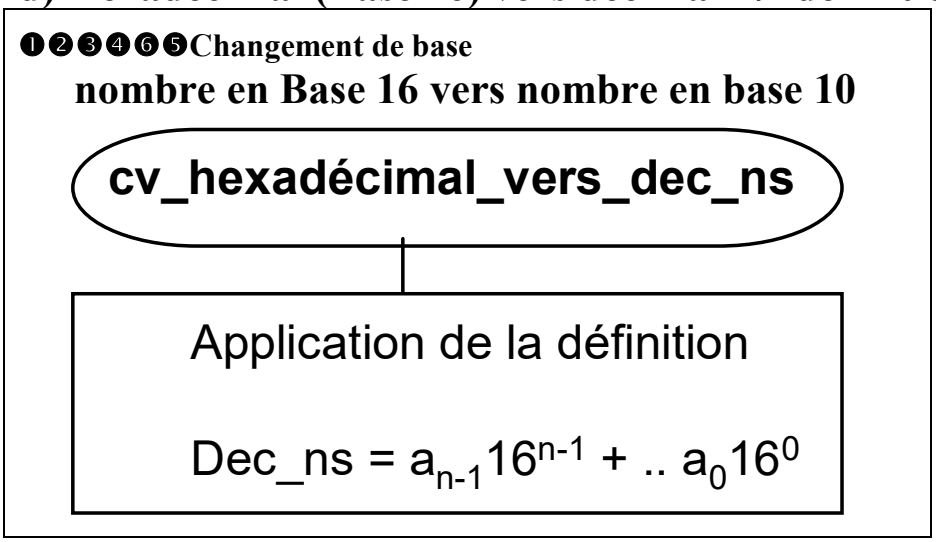
Changement de base
nombre en Base 10 vers nombre en base 16



Exemple convertir 39 en hexadécimal



d) Hexadécimal (Base 16) vers décimal ⇒ définition d'un nombre



Exemple :

$(27)_{16} = 2*16^1 + 7*16^0 = 2*16 + 7*1 = 39$

Cas particuliers :

a) Binaire (base 2) vers hexadécimal (base 16)

Règle :	**Exemple**
1- On regroupe les bits de droite à gauche par paquet de 4bits 2- On convertit chaque paquet de 4bits en hexadécimal	**$(1101)_2 = (D)_{16}$**

b) Hexadécimal (Base 16) vers Binaire (base 2)

Règle :	**Exemple**
On convertit chaque symbole hexadécimal en binaire sur 4bits	**$(A)_{16} = (1010)_2$**

Remarques importantes :

- La base hexadécimale est la notation compactée de la base binaire

- Il est conseillé de connaître par cœur

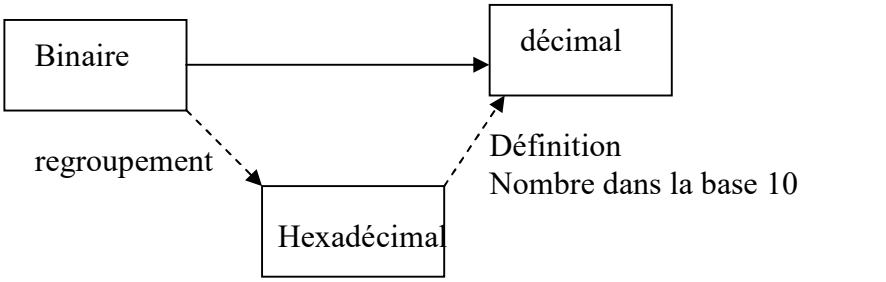
la correspondance binaire ↔ hexadécimal

Réduction du nombre de divisions ou de multiplications pour les conversions

binaire ↔ décimal

- Pour réduire le nombre de divisions successives ou de multiplications, il est conseillé de passer par la base 16, on réduit ainsi le nombre d'opérations dans un rapport de 4.

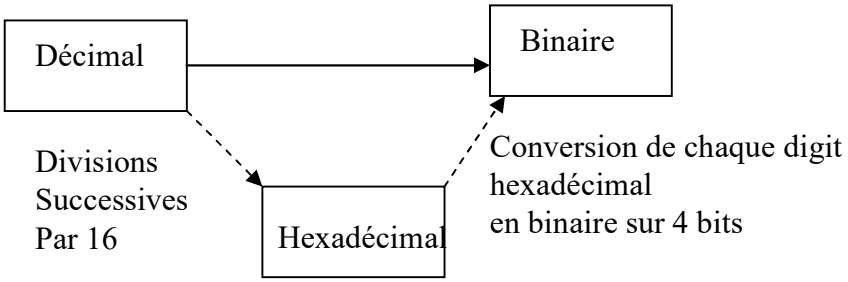
Binaire vers décimal



11110011 → F3 → $15*16^1 + 3*16^0 = 243$

au lieu de $1*2^7 + 1*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

Décimal vers binaire



243 → F3 → 1111 0011

au lieu de 8 divisions par 2

3-5 Notation

- les nombres décimaux (base 10) seront notées comme habituellement. dddd
  d ∈ [0, 9]

- les nombres binaires seront notés avec le suffixe b xxxxx**b**
  x ∈ [0, 1]

- les nombres hexadécimaux : 2 notations : avec préfixe **0x**hhhh
  h ∈ [0, A, B, C, D, E, F]

  avec suffixe yyyy**h**
  y ∈ [0, A, B, C, D, E, F]

4- Représentation et interprétation des nombres binaires

En informatique industrielle ou embarquée,

- l'information de base est binaire.

Donc la base naturelle est binaire

- la même configuration binaire peut être interprétée de plusieurs manières.

En binaire, on peut représenter :

- les nombres binaires non signés (notés ns)
- les nombres binaires signés (notés sg)
- les nombres au format BCD (simple ou packed) noté BCD
- les symboles (codes ASCII) noté entre apostrophes 'A' lu code ASCII de A
- les nombres fractionnaires
- les nombres en virgule fixe non signés et signés
- les nombres en virgule flottante

4-1 Nombres binaires non signés

Ce sont des nombres ≥ 0 .

Ce sont les nombres binaires naturels

Avec n bits, on peut représenter les nombres ≥ 0 allant de **0 à 2^n-1**

4-2 Nombres binaires signés

Ce sont des nombres négatifs ou positifs ou nul

Il existe plusieurs représentations possibles.

Mais nous ne parlerons que de celle qui est utilisée dans les calculateurs.

C'est la représentation en complément à deux (notée cpl2).

Représentation en complément à deux

Avec n bits, on peut représenter les nombres de -2^{n-1} à $+(2^{n-1})-1$

Caractéristiques

- le nombre de bits de la représentation doit être fixé
- le signe est le bit de poids fort (bit le plus à gauche du nombre)
- par convention le bit de signe,

  SF = 0 => nombre positif ou nul
  SF = 1 => nombre négatif

- l'opposé, x', d'un nombre, x, est tel que x' + x = 0 sur le nombre de bits de la représentation.

  x' est complément à 2 de x

  Règle :

  Le complément à 2 d'un nombre binaire s'obtient :

  - en faisant le complément à 1 = complémenter chaque bit
  - en ajoutant 1 au complément à 1

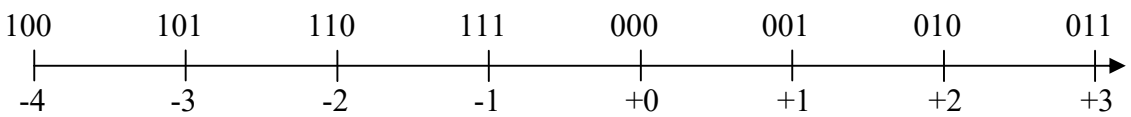
- le bit de signe fait partie intégrante du nombre signé. On ne peut pas dissocier le signe du nombre pour calculer sa valeur absolue.
- zéro est son propre opposé
- le plus petit nombre négatif (= **-2^{n-1}**) est aussi son propre opposé.

Exemple : le nombre de bits de la représentation est 3

3 bits $\Rightarrow$ 8 combinaisons binaires

codage des nombres de -2^2 à $+2^2-1$ $\Rightarrow$ -4 à +3

Valeur binaire	111	110	101	100	011	010	001	000
Valeur décimale non signée	7	6	5	4	3	2	1	0
Valeur décimale signée	-1	-2	-3	-4	3	2	1	0

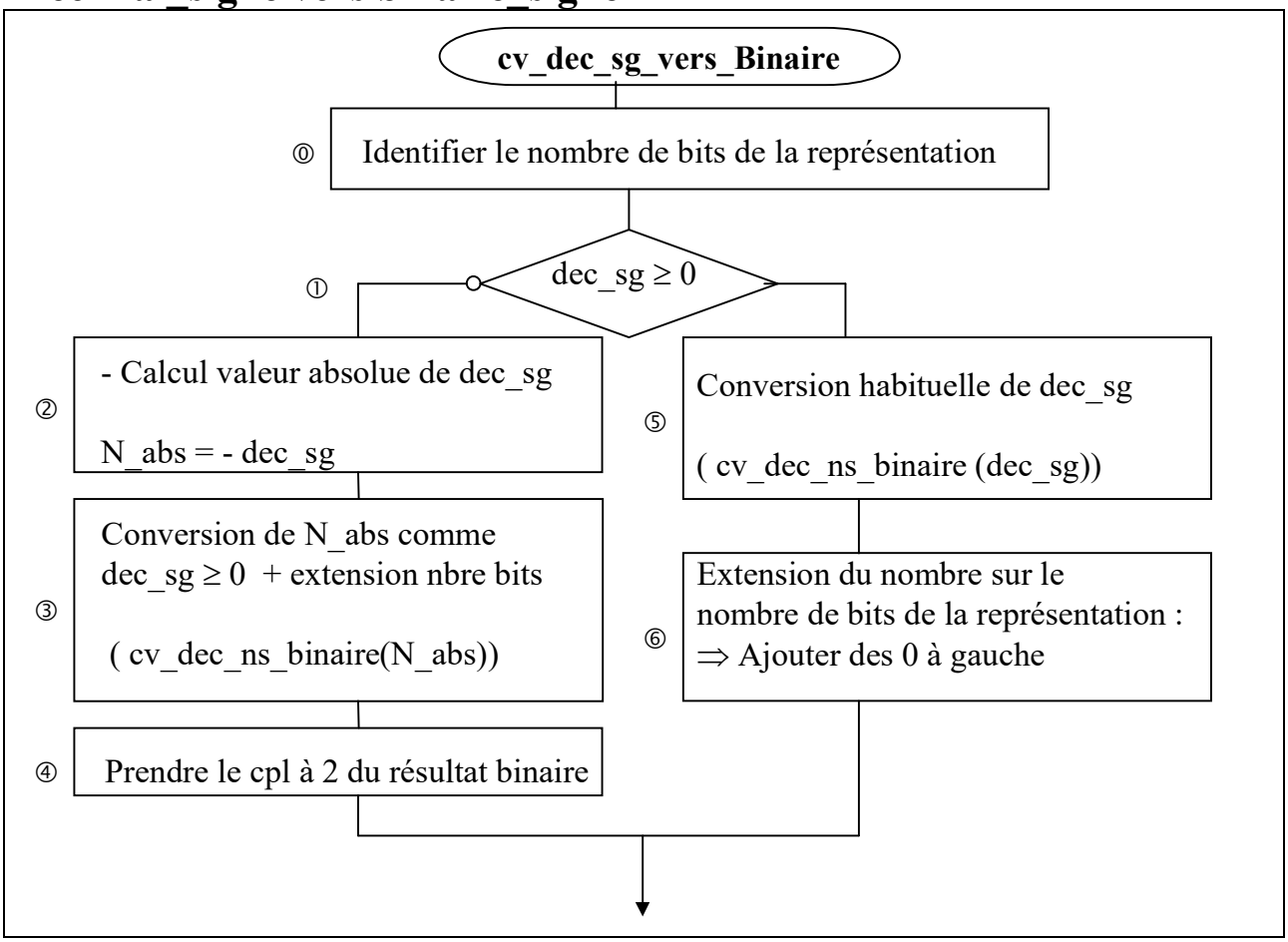


Conversions de nombres signés

Remarque importante :

Avant toute conversion de nombre binaire signé, on doit obligatoirement fixer le nombre de bits du mot. Ceci permet de déterminer la position du signe et sa valeur.

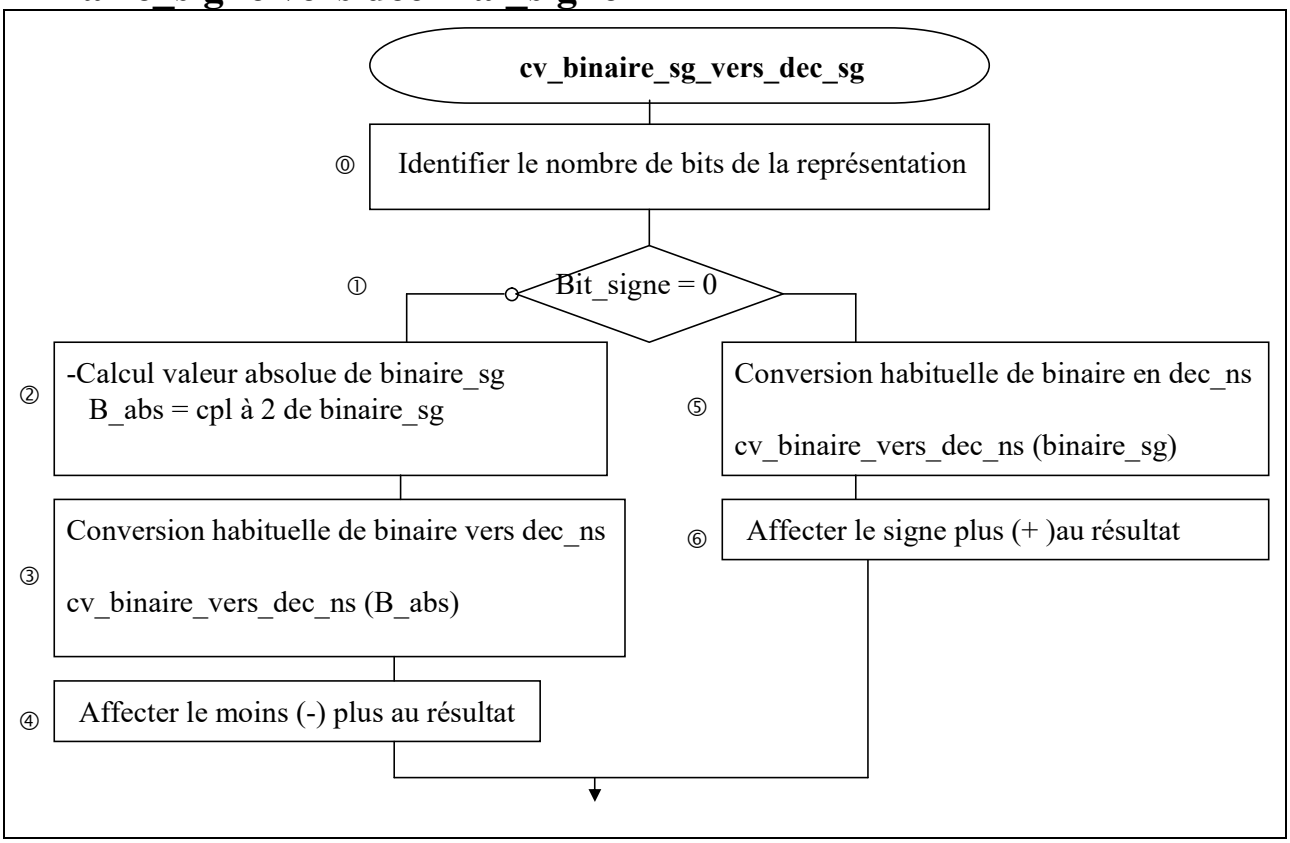
Décimal_signé vers binaire_signé



Exemple

Convertir en binaire signé en complément à 2 sur 8bits	
a) -102	**b) +102**
⓪ nombre de bits = 8	**⓪ nombre de bits = 8**
① -102 < 0	**① 102 ≥ 0**
② Valeur absolue |-102| = 102	**⑤ positif ou nul ⇒ conversion habituelle décimal binaire naturel 102 = 70H = 01110000b**
③ Conversion en binaire avec extension 102 = 70h = 01110000b	
④ Complément à deux de 01110000 10001111 + 1 10010000	
Résultat -102 = 10010000b	**Résultat +102 = 01110000b**

Binaire_signé vers decimal_signé



Exemple

Convertir en décimal signé le nombre binaire en complément à 2 sur 8bits	
a) 11111111b	**b) 00111111b**
⓪ nombre de bits = 8	**⓪ nombre de bits = 8**
① bit de signe = 1 donc nbre < 0	**① bit de signe = 0 donc nbre ≥ 0**
② Valeur absolue **| 11111111b| = Cpl2 de 11111111b** **= 00000001b**	**⑤ positif ou nul ⇒ conversion habituelle décimal vers binaire naturel** **00111111b=3FH = 63**
③ Conversion en décimal de l avaleur absolue **00000001h = 01H = 1**	**⑥ Résultat +63**
④ -1	
Résultat 11111111b = -1	**Résultat 00111111b = +63**

4-4 Nombre BCD ou Code BCD

Définition

Un nombre BCD (Binary coded decimal) ou en français (DCB =décimal codé binaire) est un **nombre binaire sur quatre bits** représentant le code binaire d'un chiffre décimal.
On peut mettre plusieurs nombres BCD côte à côte pour avoir l'équivalent d'un nombre décimal, on dit alors que c'est **BCD packed**.

Correspondance BCD simple

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

BCD Packed

45H = 01000101b = 45 en décimal

Remarques importantes :

1- Tout nombre BCD est binaire

Exemple

Le nombre binaire 100100001000b est BCD car les groupes de 4bits de droite à gauche 1001 0000 1000b ont tous leur valeur décimale correspondante ≤ 9

2- Tout nombre binaire n'est pas BCD

Exemple

Le nombre binaire 101100001000b n'est pas BCD car parmi les groupes de 4bits de droite à gauche 1001 0000 1011b au moins un groupe > 9

Code ascii
Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	⌂

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	|	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	□

4-5 Représentation des nombres binaires en virgule fixe et en flottant

4-5-1- Nombres binaires fractionnaires

Un nombre binaire fractionnaire est un nombre ayant deux parties séparées par une virgule (,) : la partie entière et la partie fractionnaire.
La partie entière vaut 0
La partie fractionnaire est décrite avec des puissances négatives de la base

Formule générale :	$N_{fractionnaire} = \sum_{i=1}^{j=m} aj * 2^{-j}$

Exemple

$(0,01)_2 = 0*2^{-1} + 1*2^{-2}$
$= 0*0,5 + 1*0,25 = 0,25$

Conversions

Partie fractionnaire

a) Décimal – binaire

La partie fractionnaire est convertie comme un nombre binaire non signé par divisions successives mais comme on divise par 2^{-1} cela revient à multiplier les résultats par 2 puis on garde pour le codage la partie entière qui peut être 0 ou 1 et on reprend la partie fractionnaire pour la multiplication suivante.

Exemple : Convertir 0,62 en binaire

Il faut noter que la partie fractionnaire peut être illimitée. Dans la pratique, on doit cependant fixer le nombre bits.

0,62 * 2 = 1,24	on garde	1 puis on recommence le processus avec 0,24
0,24 * 2 = 0,48	on garde	0 puis on recommence le processus avec 0,48
0,48 * 2 = 0,96	on garde	0 puis on recommence le processus avec 0,96

On arrête par exemple à la troisième décimale
D'où 0,62 = 0,100b

b) Binaire – décimal

On applique la formule d'expression de la partie fractionnaire. La partie fractionnaire est une somme pondérée des puissances négatives de 2 avec comme coefficients de pondération ai ∈ [0,1].

Exemple convertir en décimal le nombre binaire fractionnaire 0,1101b
$0*2^0 + 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4} = 0,8125$

Remarque :

On peut éviter les puissances négatives de 2 en procédant de la façon suivante :

- **On compte le nombre de bits, x, de la partie fractionnaire**
- **On convertit en binaire sans tenir compte de la virgule**
- **On divise le résultat obtenu par 2^x**

Application :

0,1101	**on considère la partie fractionnaire**	**1101**
	On a 4 bits et 1101 = 13	
	Donc la partie fractionnaire vaut	**$13/2^4 = 0,8125$**

4-5-2- Nombres binaires en virgule fixe

4-5-2-1 Nombres binaires positifs ou nul en virgule fixe

Un nombre binaire en virgule fixe comporte deux parties :

- une parie entière, placée avant la virgule
- une partie fractionnaire placé après la virgule

$(a_{n-1}..a_0 , e_{-1}..e_{-m})_{virgule\_fixe}$

Formule générale :	$N_{réel} = \sum_{i=-m}^{i=n-1} bi * 2^i = \sum_{i=-0}^{i=n-1} ai * 2^i + \sum_{i=-m}^{i=-1} ei * 2^i$ n est le nombre de bits avant la virgule m est le nombre de bits après la virgule

Comme on ne peut pas introduire la virgule comme un symbole supplémentaire en binaire, et comme son nom l'indique, pour en faire une utilisation cohérente, il est conseillé de fixer par convention la position de la virgule (,).

Conversions

Il est conseillé de convertir chaque partie individuellement puis de transcrire les résultats en les séparant par une virgule (,).
Se reporter aux nombres binaires fractionnaires

Exemple

a) Décimal vers virgule fixe

Convertir 47,70 en binaire virgule fixe

On convertit la partie entière et la partie fractionnaire
(47) = 101111b

On convertit la partie fractionnaire

0,70 * 2 = 1,40	bit -1 = 1	on recommence avec 0,40
0,40 * 2 = 0,80	bit -2 = 0	on recommence avec 0,80
0,80 * 2 = 1,60	bit -3 = 1	on recommence avec 0,60
0,60 * 2 = 1,20	bit -3 = 1	on recommence avec 0,20

Nous arrêtons ici parce que la suite peut être très longue.
0,70 = (0,1011b)

47,70 = (101111,1011b)

b) Virgule fixe vers décimal

Convertir en décimal le nombre en virgule fixe 1100,11101b
On convertit chaque partie :

Partie entière
1100b = 12

Partie fractionnaire
$0,11101b = 0*2^0 + 1*2^{-1} + 1*2^{-2} + 1*2^{-3} + 0*2^{-4} + 1*2^{-5} = 0,90625$

Ou (11101) = 29 / 2^5 = 0,90625

1100,11101b = 12,90625

4-5-2-2 Nombres binaires signés en virgule fixe

Mêmes règles que pour les nombres binaires signés sans virgule

- Bit de signe
- Règles de conversion en décimal

Remarque importante

On fait le calcul du complément à deux sans tenir compte de la virgule.
On replace la virgule après le calcul.

Exemple

On considère un nombre en virgule fixe avec 8bits partie entière et 4 bits partie fractionnaire
Convertir en décimal le nombre 11110000,1010b

- bit de signe = 1 donc nombre < 0

- On prend la valeur absolue = cpl2 (11110000,1010)

Soit	00001111,0101
+	1
Résultat	00001111,0110

- D'où la valeur absolue 15, 375

- 11110000,1010b = -15,375

4-6-Nombres binaires en virgule flottante

La représentation en virgule flottante est un codage qui permet d'augmenter la dynamique et la précision de représentation des nombres binaires. Elle permet de se rapprocher des nombres réels.
La représentation en virgule flottante est un codage permettant de transcrire la fonction :

$N = \pm M \times B^e$

N est un nombre réel, **M** est appelé la mantisse, **B** est la base et **e** est l'exposant

Dans notre cas **B** sera égal **2** (base binaire)

En binaire, on ne peut pas ajouter des symboles, il faut définir par convention ce qui représente chaque élément de la formule.

Une solution consiste à :

- définir la taille du mot binaire représentant le nombre flottant
- définir l'organisation du mot binaire en indiquant la place et la taille de l'exposant, de la mantisse et de son signe

Exemple de mot binaire et son organisation:

Code_signe	Code_exposant	Code_mantisse
1bit	e_bit	m_bit

La base est implicite mais on doit fixer le nombre de bits (1 + e_bit + m_bit) de la représentation en indiquant le nombre de bits associé au :

- codage du signe de la mantisse, code_signe (1bit)
- codage de l'exposant, code_exposant (e_bit)
- codage de la mantisse, code_mantisse (m_bit)

Cette représentation peut être normalisée sous plusieurs formats. Il faut donc définir les relations entre la formule générale et le codage, c'est-à-dire entre :

code_signe et le signe du nombre
code_exposant et e
code_mantisse et M

Standard IEEE754

Nous allons considérer le standard IEEE 754 qui est couramment utilisé par les compilateurs de langages informatiques et en électronique numérique.

Il définit deux formats de représentations de nombre en flottant:

- une représentation sur 32bits (simple précision)
- une représentation sur 64bits (double précision)

Nous allons faire cette présentation avec le format simple précision

Flottant simple précision IEEE754

Code_signe	Code_exposant	Code_mantisse
1bit	8bits	23bits

Le codage du signe sera sur un bit avec la convention habituelle

0 = positif ou nul 1 = négatif

Le codage de l'exposant sera sur 8 bits avec un décalage (aussi appelé biais).
Le décalage (= 2^n-1) est souvent exprimé en fonction du nombre de bits de codage de l'exposant.
n est le nombre de bits de codage de l'exposant avec la relation :

e = code_exposant – décalage.

Comme n = 8, décalage = 127. Donc **e =code_exposant – 127.**

Le codage de la mantisse sera sur 23bits.
On admettra que la mantisse est de la forme 1,ffffff mais on ne stockera que la partie fractionnaire 0, ffffff et on considèrera dans le cas général que la partie entière 1 est implicite.

Nous nous limiterons aux nombres normalisés

D'où la formule générale pour les nombres normalisés

$$N_{réel} = (-1)^{code\_signe} * (1+ (code\_mantisse)_{fractionnaire}) * 2^{(code\_exposant - 127)}$$

avec code_exposant ≠ 0 et 255

Cas particuliers de la représentation

Cette représentation permet de représenter un nombre fini de possibilités. Par ailleurs, elle ne permet pas de bien représenter tous les cas de figure que nous allons traiter comme des cas particuliers qui sont définis par la convention de la norme.

a) Représentation de 0

code_exposant = 0 et code_mantisse = 0

Cette convention conduit à un 0 positif et un 0 négatif

b) Représentation des nombres dénormalisés (nombres très petits)

code_exposant = 0 et code_mantisse ≠ 0

$$\mathbf{N_{réel} = (-1)^{code\_signe} * (code\_mantisse)_{fractionnaire} * 2^{(0)}}$$

Dans ce cas exposant est par convention -126 pour la conversion

c) Représentation de l'infini

code_exposant = 255 et code_mantisse = 0 (Utilisée pour la division par 0)

donc $+\infty$ est représenté par code_signe = 0

$-\infty$ est représenté par code_signe = 1

c) Représentation d'un nombre non reconnu (utilisée pour la division ∞/∞)

code_exposant = 255 et code_mantisse ≠ 0

Quelques exemples

Nombre réel (N)	signe	code_exposant	code_mantisse	valeur
+0	0	00000000	000 0000 0000 0000 0000 0000	0.0
-0	1	00000000	000 0000 0000 0000 0000 0000	-0.0
$+\infty$	0	11111111	000 0000 0000 0000 0000 0000	$+\infty$
$-\infty$	1	11111111	000 0000 0000 0000 0000 0000	$-\infty$
NaN	*	11111111	≠ 0	Non nombre
Plus petit N ≥ 0 dénormalisé	0	00000000	000 0000 0000 0000 0000 000**1**	$(-1)^0 * 2^{-126} * 2^{-23}$ $= 2^{-149}$ $= 5{,}9*10^{-39}$
Plus grand N ≥ 0 normalisé	0	11111110	111 1111 1111 1111 1111 1111	$(-1)^0 * 2^{127} * \ldots$ $= 3{,}4028*10^{38}$
Plus petit N ≥ 0 normalisé	0	00000001	000 0000 0000 0000 0000 0000	2^{-126} $= 1{,}1754*10^{-38}$

Conversions décimal ↔ flottant

$N_{decimal} = \pm 2^{exposant} x Mantisse$	Codage IEEE754

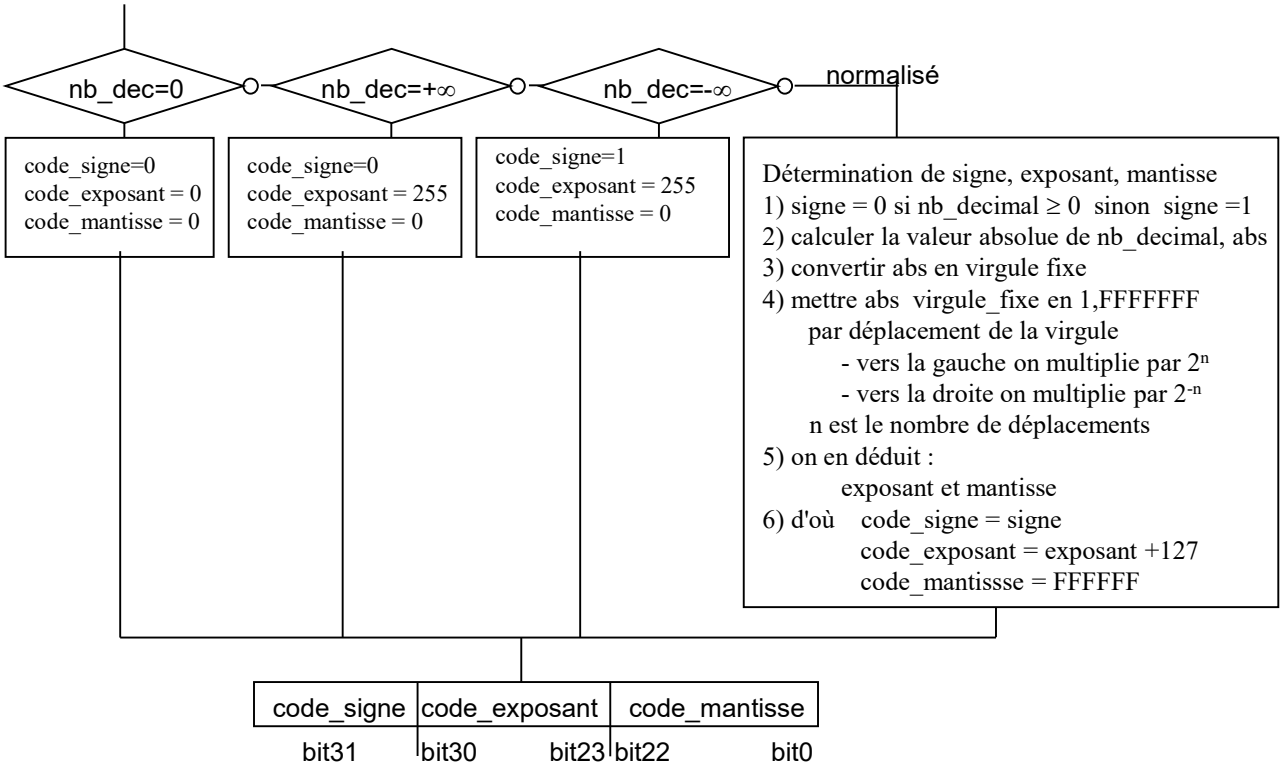
bit31	bit30 … bit23	bit22 … bit0
code_signe	code_exposant	code_mantisse

a) Conversion décimal → flottant

Elle revient :
- à calculer code_signe, code_exposant et code_mantisse
- à constituer les 32bits

Par simplification, on ne traitera que le cas des nombres normalisés
Cependant, on traitera les cas particuliers de 0 et $\pm\infty$

D'où l'organigramme



Exemple :
Convertir 12,25 en flottant
Nombre ≥ 0 **Code_signe = 0**
On procède comme en virgule fixe pour la valeur absolue
12,125 = 1100,001
On déplace la virgule pour avoir l'exposant et la mantisse
$1100,001 = 1,100001*2^3$
Exposant = 3 **Code_exposant = exposant +127 = 130**
Soit en binaire sur 8 bits 10000010b
Mantisse = 1,100001 **Code_mantisse = 10000100000…0**

12,125 en flottant 0 10000010 10000100000000000000000b

b) Conversion flottant → décimal

Elle revient

- à identifier dans les 32 bits code_signe, code_exposant et code_mantisse
- si code_signe, code_exposant et code_mantisse représentent 0 ou $\pm\infty$

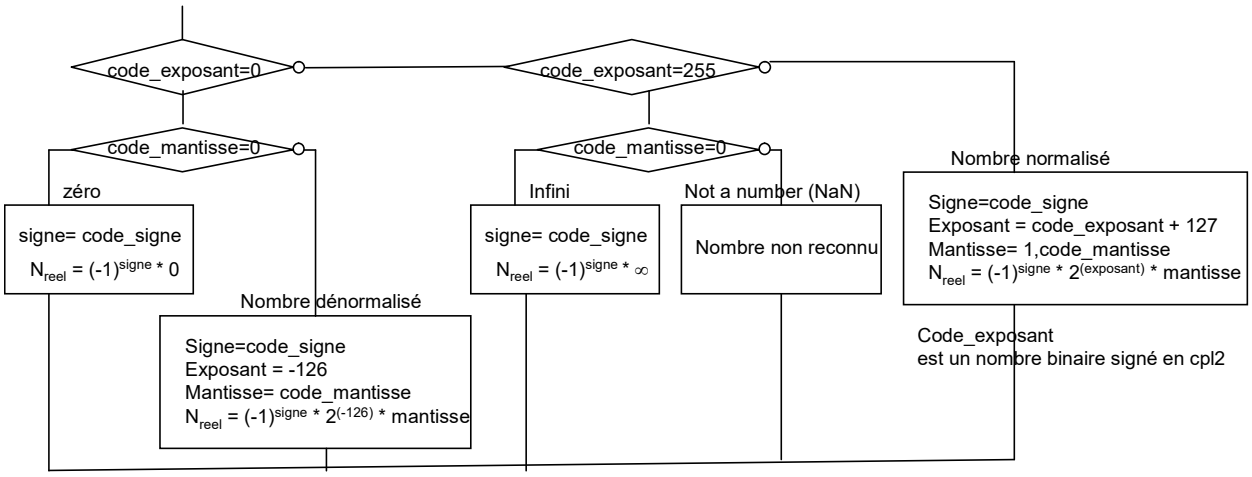
  alors

  la conversion est faite par convention

  sinon

  en déduire signe, exposant = code_exposant -127 et mantisse = 1,code_mantisse

  et par suite le $N = N_{decimal} = \pm 2^{exposant}$.Mantisse



Exemple

Convertir le nombre en flottant : C142000H

- On convertit en binaire pour identifier les champs

1100 0001 0100 0010 0000 0000 0000 0000

1 10000010 100 0010 0000 0000 0000 0000

1	**10000010**	**10000100000 0000 0000 0000**
Code_signe	**code_exposant**	**code_mantisse**
Signe = -	**exposant = code_exposant -127** **donc exposant = 130 – 127 = 3**	**Mantisse = 1,code_mantisse** **Donc matisse = 1,1000010000000**

$N = -1,1000010000 * 2^3 = -1100,0010 = -12,125$