Chapitre 1 : Représentation des données

BOULAICHE Ammar

Université de Jijel

Cours Structure Machine 1 - Première Année Informatique

Septembre 2023

Introduction

- Les ordinateurs peuvent traiter des données de différents types (textes, nombres, images, vidéos, etc.).
- Toutes ces données sont représentées au sein des composants de l'ordinateur sous forme binaire.
- Il s'agit d'une représentation à deux états symbolisés par 1 (courant) et 0 (pas de courant).
- Un codage est donc nécessaire pour convertir les données symboliques de leur forme externe (habituelle) à la forme binaire (0 et 1) exploitable par l'ordinateur.
- Pour les données numériques (nombres), cette opération est basée sur la conversion des nombres de la base décimale vers la base binaire. C'est ce qu'on appelle «le changement de base».

Numération en base b

- Une valeur numérique (un nombre) peut être représentée dans plusieurs bases différentes.
- L'expression d'une telle valeur en base b quelconque est généralement donnée par la notation positionnelle suivante.

$$(A)_b = (a_n a_{n-1} ... a_1 a_0, a_{-1} a_{-2} ... a_{-p})_b = \sum a_i \times b^i$$

- Chaque coefficient a_i est un chiffre compris entre 0 et $b-1$.

Exemple :

  - En base décimale (b=10), $a_i \in \{0,1,2,3,4,5,6,7,8,9\}$
  - En base binaire (b=2), $a_i \in \{0,1\}$
  - En base octale (b=8), $a_i \in \{0,1,2,3,4,5,6,7\}$
  - En base hexadécimale (b=16), $a_i \in \{0,1,2,3,4,5,6,7,$ $8,9,A,B,C,D,E,F\}$

- La notation $()_b$ indique que le nombre est écrit en base b.

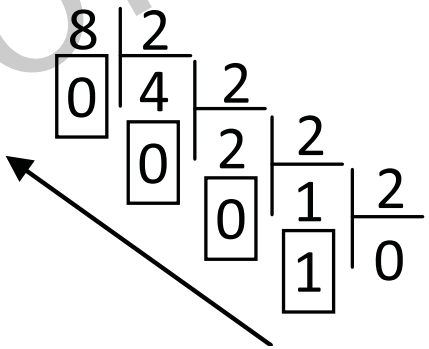
Changement de base

- Les changements de base les plus rencontrés sont les suivants.
  - Conversion d'un nombre décimal en une base b quelconque.
  - Conversion d'un nombre exprimé dans une base b quelconque en décimal.
  - Conversion d'un nombre exprimé dans une base $b = 2^n$ quelconque en binaire.
  - Conversion d'un nombre binaire en un nombre exprimé dans une base $b = 2^n$ quelconque.
  - Conversion d'un nombre exprimé dans une base $b1$ en un nombre exprimé dans une base $b2$, tel que $b1 \neq b2 \neq 10$.

Conversion de la base 10 vers la base b

- Pour la partie entière (celle avant la virgule), la conversion peut s'effectuer en appliquant la méthode des divisions successives. On divise le nombre décimal par la base b, puis on divise le quotient obtenu par la base b, et ainsi de suite jusqu'à l'obtention d'un quotient nul. On prend ensuite les restes des divisions dans l'ordre inverse de celui dans lequel ils ont été obtenus pour former le résultat de la conversion.

Exemple : La conversion du nombre décimal $(8)_{10}$ en binaire (base 2) sera donc effectuée comme suit :



$$(8)_{10} = (1000)_2$$

Conversion de la base 10 vers la base b

- Pour la partie fractionnaire (celle après la virgule), la conversion peut s'effectuer en appliquant la méthode des multiplications successives. On multiplie la partie fractionnaire du nombre décimal par la base b, puis on multiplie la partie fractionnaire du résultat obtenu par la base b, et ainsi de suite jusqu'à l'obtention d'un résultat avec partie fractionnaire nulle. On prend ensuite les chiffres des parties entières des résultats dans le même ordre que celui dans lequel ils ont été obtenus pour former le résultat de la conversion.

  Exemple : La conversion du nombre décimal $(0.375)_{10}$ en binaire (base 2) sera donc effectuée comme suit :

$$\begin{aligned} 0.375 \times 2 &= \boxed{0}.75 \Rightarrow a_{-1} = \boxed{0} \\ 0.75 \times 2 &= \boxed{1}.50 \Rightarrow a_{-2} = \boxed{1} \\ 0.50 \times 2 &= \boxed{1}.00 \Rightarrow a_{-2} = \boxed{1} \end{aligned} \quad \longrightarrow \quad (0.375)_{10} = (0.011)_2$$

- Le résultat de la conversion du nombre décimal $(8.375)_{10}$ en binaire sera donc : $(1000.011)_2$.

Conversion de la base b vers la base 10

- La conversion de la base b vers la base 10 peut s'effectuer en faisant la somme des produits des chiffres du nombre à convertir et de la base b pondérée par la position de ces chiffres.

  Exemple : La conversion des nombres $(1101.01)_2$, $(1B2.8)_{16}$, $(45.3)_8$ $(431.3)_5$ en base 10 sera donc effectuée comme suit :

$$(1101.01)_2 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-3} = (13.125)_{10}$$

$$(1B2.8)_{16} = 1 \times 16^2 + 11 \times 16^1 + 2 \times 16^0 + 8 \times 16^{-1} = (434.5)_{10}$$

$$(45.3)_8 = 4 \times 8^1 + 5 \times 8^0 + 3 \times 8^{-1} = (37.375)_{10}$$

$$(431.3)_5 = 4 \times 5^2 + 3 \times 5^1 + 1 \times 5^0 + 3 \times 5^{-1} = (116.6)_{10}$$

Conversion de la base b = 2^n vers la base 2

- La conversion d'un nombre de la base $b = 2^n$ vers la base 2 peut s'effectuer en convertissant chaque chiffre de ce nombre à part sur n bits et en concaténant les résultats obtenus pour former le nombre correspondant en binaire.

  Exemple : La conversion des nombres $(312.2)_4$, $(457.6)_8$, $(5B2.C)_{16}$ en base 2 sera donc effectuée comme suit :

$$(312.2)_4 = \underbrace{11}_{3}\,\underbrace{01}_{1}\,\underbrace{10}_{2}.\underbrace{10}_{2} = (110110.10)_2$$

$$(457.6)_8 = \underbrace{100}_{4}\,\underbrace{101}_{5}\,\underbrace{111}_{7}.\underbrace{110}_{6} = (100101111.110)_2$$

$$(5B2.C)_{16} = \underbrace{0101}_{5}\,\underbrace{1011}_{B}\,\underbrace{0010}_{2}.\underbrace{1100}_{C} = (010110110010.1100)_2$$

Conversion de la base 2 vers la base b = 2^n

- La conversion d'un nombre de la base binaire vers la base $b = 2^n$ peut se faire facilement en découpant les chiffre de ce nombre en paquets de n bits et en convertissant chaque paquet à part. Le découpage doit commencer à partir de la virgule vers les deux sens (vers la gauche pour la partie entière et vers la droite pour la partie fractionnaire). Lorsqu'un paquet est incomplet, on le complète avec des 0.

Exemple : La conversion du nombre binaire $(10011101011.11001)_2$ vers les bases 8 et 16 sera donc effectuée comme suit :

$$(10011101011.11001)_2 = \underbrace{010}_{2}\underbrace{011}_{3}\underbrace{101}_{5}\underbrace{011}_{3}.\underbrace{110}_{6}\underbrace{010}_{2} = (2353.62)_8$$

$$(10011101011.11001)_2 = \underbrace{0100}_{4}\underbrace{1110}_{E}\underbrace{1011}_{B}.\underbrace{1100}_{C}\underbrace{1000}_{8} = (4EB.C8)_{16}$$

Conversion de la base b1 vers la base b2 ($b1 \neq b2 \neq 10$)

- **Règle générale** : La conversion de la base $b1$ vers la base $b2$, tel que $b1 \neq b2 \neq 10$ peut se faire en passant par la base 10 comme base intermédiaire.

  Exemple : La conversion du nombre $(75.3)_8$ de la base 8 vers la base 6 sera donc effectuée comme suit :

  $(75.3)_8 = 7 \times 8^1 + 5 \times 8^0 + 3 \times 8^{-1} = (61.375)_{10}$
  $(61.375)_{10} = 1\times6^2+4\times6^1+1\times6^0+2\times6^{-1}+1\times6^{-2}+3\times6^{-3} = (141.213)_6$

- **Cas particulier** : Dans le cas où les bases $b1$ et $b2$ sont tous les deux des puissances de 2 ($b1{=}2^n$ et $b2{=}2^m$, tel que $n{\neq}m$), on peut passer par la base 2 comme base intermédiaire au lieu de la base 10.

  Exemple : La conversion du nombre $(52.6)_8$ de la base 8 vers la base 16 sera donc effectuée comme suit :

$$(52.6)_8 = \underbrace{101}_{5}\underbrace{010}_{2}.\underbrace{110}_{6} = \underbrace{0010}_{2}\underbrace{1010}_{A}.\underbrace{1100}_{C} = (2A.C)_{16}$$

Arithmétique en binaire

Addition

$$\begin{array}{rr} & 1101{,}0011 \\ + & 1001{,}1110 \\ \hline = & 10111{,}0001 \end{array}$$

Soustraction

$$\begin{array}{rr} & 1111{,}0110 \\ - & 0101{,}1101 \\ \hline = & 1001{,}1001 \end{array}$$

Multiplication

$$\begin{array}{rr} & 0011 \\ \times & 1110 \\ \hline & 0000 \\ + & 0011 \\ + & 0011 \\ + & 0011 \\ \hline = & 0101010 \end{array}$$

Division

$$\begin{array}{r|l} 10001000{,}111 & 101 \\ \hline 101 & \\ \hline 00111 & \\ 101 & \\ \hline 01000 & \\ 101 & \\ \hline 00110 & \\ 101 & \\ \hline 001\ 11 & \\ 1\ 01 & \\ \hline 0\ 101 & \\ 101 & \\ \hline 000 & \end{array} \quad 11011{,}011$$

Représentation des données numériques

- La représentation des données numériques varie suivant qu'il s'agit de nombres entiers naturels (non-signés), de nombres entiers relatifs (signés) ou de nombres réels.
- Les nombres entiers naturels ne posent aucun problème, ils sont représentés en binaire sur n bits, où n représente le nombre d'unités mémoires réservées (n = 8, 16, 32, 64, etc.). On peut donc représenter des nombres allant de 0 à 2^{n-1}.
- Pour les nombres entiers relatifs et les nombres réels, le problème qui se pose est celui de la conversion du signe (- ou +) et de la virgule du nombre en binaire.
- La représentation des nombres entiers relatifs et des nombres réels se base principalement sur la définition d'une convention permettant de remplacer le signe et la virgule du nombre par des chiffres binaires.

Représentation des entiers relatifs

- Les entiers relatifs peuvent être codés selon trois méthodes :
  - **Signe et valeur absolue (S+VA)** : Dans cette méthode, on réserve le bit le plus à gauche (le poids fort) pour représenter le signe du nombre (0 : nombre positif, 1 : nombre négatif). Les bits qui restent représentent la valeur absolue du nombre.

    Exemple : Sur 8 bits, les entiers seront représentés comme suit :

    $(+21)_{10} = (00010101)_{S+VA(8bits)}$ $\quad (-21)_{10} = (10010101)_{S+VA(8bits)}$

    $(+36)_{10} = (00100100)_{S+VA(8bits)}$ $\quad (-36)_{10} = (10100100)_{S+VA(8bits)}$

  - **Complément à 1 (C1)** : Dans cette méthode, les entiers négatifs sont représentés en inversant les valeurs des bits constituant ces nombres.

    Exemple : Sur 8 bits, les entiers seront représentés comme suit :

    $(+21)_{10} = (00010101)_{C1(8bits)}$ $\quad (-21)_{10} = (11101010)_{C1(8bits)}$

    $(+36)_{10} = (00100100)_{C1(8bits)}$ $\quad (-36)_{10} = (11011011)_{C1(8bits)}$

Représentation des entiers relatifs

- **Complément à 2 (C2)** : Dans cette méthode, les entiers négatifs sont représentés en ajoutant la valeur 1 à leur représentation en complément à 1. L'avantage de cette méthode est qu'il n'y a plus qu'une seule représentation pour le 0, ce qui permet de libérer la combinaison 10...0 pour représenter le nombre -2^{n-1} qu'était irreprésentable dans les deux premières méthodes.

  Exemple : Sur 8 bits, les entiers seront représentés comme suit :

  $(+21)_{10} = (00010101)_{C2(8bits)}$ $\quad$ $(-21)_{10} = (11101011)_{C2(8bits)}$

  $(+36)_{10} = (00100100)_{C2(8bits)}$ $\quad$ $(-36)_{10} = (11011100)_{C2(8bits)}$

- La représentation en complémént à 2 est la représentation standard sur les ordinateurs pour exprimer les nombres entiers négatifs.

Arithmétique sur les entiers relatifs

- Pour les nombres entiers signés, l'opération de soustraction est remplacée par l'opération d'addition (A – B = A + (-B)).
- L'addition en complément à 1 est effectuée en additionnant les bits des deux nombres, bit par bit, y compris le bit de signe, puis s'il y a une retenue, on l'ajoute au résultat de l'addition.

  Exemple : Les additions (+5)+(-3) et (-3)+(+2) sont effectuées, sur 4 bits, comme suit :

+5	0101		–3	1100	
–3	1100		+2	0010	
+2	10001		–1	1110	C1(4bits)
	→ 1				
	0010	C1(4bits)			

Arithmétique sur les entiers relatifs

- L'addition en complément à 2 est plus facile qu'en complément à 1, il suffit d'effectuer une addition binaire bit par bit, y compris les bits de signe, et d'ignorer ensuite les éventuelles retenues.

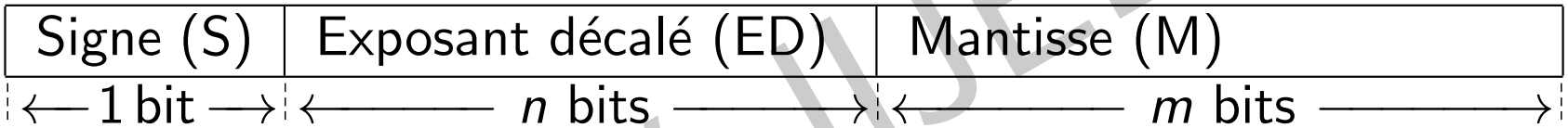
  Exemple : Les additions (+5)+(-3) et (-3)+(+2) sont effectuées, sur 4 bits, comme suit :

$$\begin{array}{r} +5 \\ -3 \\ \hline +2 \end{array} \quad \left(\begin{array}{r} 0101 \\ 1101 \\ \hline \cancel{1}\,0010 \end{array} \right)_{C2(4bits)} \qquad \begin{array}{r} -3 \\ +2 \\ \hline -1 \end{array} \quad \left(\begin{array}{r} 1101 \\ 0010 \\ \hline 1111 \end{array} \right)_{C2(4bits)}$$

- Un débordement est signalé lorsque la retenue en entrée et la retenue en sortie du bit de signe sont différentes.

Représentation des nombres réels

- La représentation des nombres fractionnaires la plus répondue dans les ordinateurs d'aujourd'hui est celle en virgule flottante selon la norme IEEE 754. Dans cette représentation les nombres fractionnaires sont représentés sous la forme suivante :



- Les champs de cette représentation sont extraits de l'écriture scientifique suivante du nombre réel.

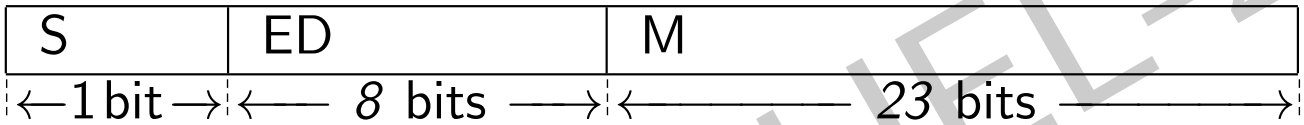
$$N = (-1)^S \times 1, M \times 2^E$$

- L'exposant décalé $ED = E + D$. Où D est le décalage (biais) de l'exposant, c'est une valeur fixe donnée par la formule suivante :

$$D = 2^{\text{Taille de ED}-1} - 1$$

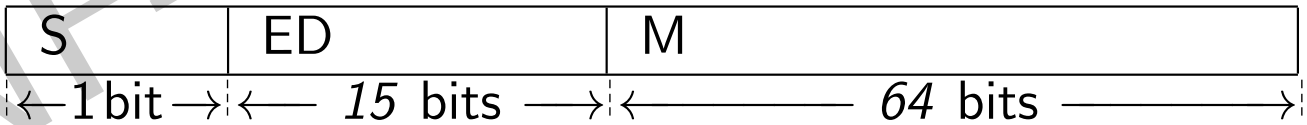
Représentation des nombres réels

- La norme IEEE 754 possède 3 formats de représentation :
  - **Format simple précision (32 bits)**



  - **Format double précision (64 bits)**

  - **Format précision étendue (80 bits)**



Représentation des nombres réels

Exemple 1 : La représentation du nombre réel $(-42.375)_{10}$ selon la norme IEEE754 en simple précision sera donc calculée comme suit :

$(-42,375)_{10} = (-101010,011)_2$.

$(-101010,011)_2 = (-1)^1 \times 1,01010011 \times 2^5$.

$ED = E + D = 5 + 127 = (132)_{10} = (10000100)_2$

1	10000100	01010011000000000000000

Exemple 2 : La valeur décimale représentée en virgule flottante par le code $(C26D000H)_{IEEE754}$ sera donc calculée comme suit :

1	10000100	11011010000000000000000

$ED=(10000100)_2=(132)_{10} \Rightarrow E=ED-D=132-127=(5)_{10}$.

$N=(-1)^S \times 1,M \times 2^E=(-1)^1 \times 1,1101101 \times 2^5=(-111011,01)_2$

$(-111011,01)_2 = (-59,25)_{10}$

Représentation des nombres réels

- **Cas particuliers :** Certaines configurations de la norme IEEE 754 sont réservées pour représenter des valeurs spéciales.
  - | x | $00\ldots0$ | $00\ldots0$ | code la valeur zéro ($x = 0$ ou 1).
  - | 0 | $11\ldots1$ | $00\ldots0$ | code la valeur plus infini ($+\infty$).
  - | 1 | $11\ldots1$ | $00\ldots0$ | code la valeur moins infini ($-\infty$).
  - | x | $00\ldots0$ | $M \neq 00\ldots0$ | code un nombre dénormalisé.
  - | x | $11\ldots1$ | $M \neq 00\ldots0$ | code un nombre indéterminé (NaN).

Représentation des données alphanumériques

- Les données alphanumérique sont des données non numériques incluant des lettres alphabétiques majuscules et minuscules, des symboles de ponctuation (& @ , . ; # ”, etc.), et des chiffres.
- La représentation des données alphanumériques est réalisée à partir d'une table de correspondance indiquant la configuration binaire représentant chaque caractère.
- Les deux codes les plus courants dans la représentation des données alphanumériques sont le code ASCII et le code UNICODE.
- Le codage en code ASCII est basé sur une table contenant les caractères les plus utilisés en langue anglaise. Chaque caractère est représenté sur 7 bits (ce qui donne $2^7 = 128$ combinaisons possibles).
- Les codes ASCII compris entre 0 et 31 représentent les caractères de contrôles (caractères non affichables). Ils sont utilisés pour indiquer des actions comme : passer à la ligne (CR, LF), émettre un bip sonore (BEL), etc.

Le code ASCII

- **Table du 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

- Dans cette table, la lettre A se trouve à l'intersection de la colonne 100 et de la ligne 0001. Le code ASCII de la lettre A est donc 1000001 (41 en hexadécimal).

Le code UNICODE

- L'UNICODE code les caractères sur 16 bits, ce qui permet donc de coder 65 536 (2^{16}) caractères différents.
- Il a été proposé essentiellement pour unifier les différents encodages de caractères existants mais aussi pour permettre d'incorporer tous les alphabets existants (latin, arabe, chinois, cyrillique, bengali, etc.) dans une seule table d'encodage.
- Il est compatible avec le code ASCII. Les 256 premiers caractères Unicode sont les mêmes que ceux de la table ASCII (basique et étendu).
- La table suivante montre les plages Unicode de quelques alphabets.

Alphabet	Plage des codes	Alphabet	Plage des codes
Latin	[U+0000 à U+024F]	Hébreu	[U+0590 à U+05FF]
Grec et Copte	[U+0370 à U+03FF]	Arabe	[U+0600 à U+06FF]
Cyrillique	[U+0400 à U+052F]	Maldivien	[U+0780 U+07BF]
Arménien	[U+0530 à U+058F]	Éthiopien	[U+1200 à U+137F]

Autres codes numériques

- Il existent de nombreux autres codes numériques dont chacun a un domaine d'application particulier.
- Parmi tous ces codes, nous citons :
  - **Le code BCD** : utilisé, par exemple, pour communiquer avec des dispositifs décimaux, comme certains afficheurs.
  - **Le code Gray** : utilisé, par exemple, pour résoudre les problèmes liés aux codeurs de position absolue (le passage d'une position à l'autre n'entraine alors le changement que d'un seul bit).
  - **Le code Johnson** : utilisé, par exemple, pour détecter les erreurs qui pourraient se produire lors de la transmission des données d'un dispositif à l'autre.

Code BCD

- Dans le code BCD (Binary Coded Decimal), chaque chiffre de la représentation décimale est codé en binaire naturel sur 4 bits.

Exemples :

$$(387)_{10} = (0011\ 1000\ 0111)_{\text{BCD}}$$
$$(23)_{10} = (0010\ 0011)_{\text{BCD}}$$
$$(2175)_{10} = (0010\ 0001\ 0111\ 0101)_{\text{BCD}}$$

- La conversion du code BCD vers le décimal se fait en regroupant les bits 4 par 4. Chaque groupe sera donc converti individuellement.

Exemples :

$$(0010\ 0101\ 1000)_{\text{BCD}} = (258)_{10}$$
$$(0011\ 0010)_{\text{BCD}} = (32)_{10}$$
$$(0100\ 0000\ 1001\ 0110)_{\text{BCD}} = (4096)_{10}$$

Code Gray

- Dans le code gray (binaire réfléchi), un seul bit change quand on passe d'une valeur à la valeur suivante. On l'appelle donc le code adjacent.
- Pour convertir un nombre du binaire naturel au binaire réfléchi, il suffit de changer le bit qui précède directement un bit à 1.

  Exemples : $(110101)_2 = (101111)_{\text{Gray}}$
  $(1110011011)_2 = (1001010110)_{\text{Gray}}$

- La conversion du code gray vers le binaire naturel se fait en changeant le bit qui précède directement un bit à 1. On commence du bit le plus à gauche et on fait la mise à jour sur la source après chaque changement.

  Exemples : $(110111)_{\text{Gray}} = (100101)_2$
  $(1010011101)_{\text{Gray}} = (1100010110)_2$

Code Johnson

- Dans le code Johnson, chaque chiffre de la représentation décimale est codé sur 5 bits.

- Les 1 constituant les codes Johnson sont toujours placés en un seul groupe qui se déplace d'un chiffre décimal au chiffre suivant, comme le montre le tableau suivant.

Décimal	0	1	2	3	4	5	6	7	8	9
Johnson	00000	00001	00011	00111	01111	11111	11110	11100	11000	10000

- La conversion du décimal vers le code Johnson se fait en remplaçant chaque chiffre décimal par son équivalent Johnson.

Exemples :

$$(387)_{10} = (00111\ 11000\ 11100)_{\text{Johnson}}$$
$$(23)_{10} = (00011\ 00111)_{\text{Johnson}}$$
$$(2175)_{10} = (00011\ 00001\ 11100\ 11111)_{\text{Johnson}}$$

Références

Wladimir Mercouroff.

Architecture matérielle et logicielle des ordinateurs et des microprocesseurs.

Armand Colin, 1990. ISBN : 2-200-42007-2.

Bibliothèque centrale, Université de Jijel, Cote :002/07.

Joel Ristori, Lucien Ungaro.

Cours d'architecture des ordinateurs.

Eyrolles, 1991.

Bibliothèque centrale, Université de Jijel, Cote :002/17.

Pierre-Alain Goupille.

Technologie des ordinateurs pour les I.U.T et B.T.S informatique avec exercices.

Masson, 1993.

Bibliothèque centrale, Université de Jijel, Cote :002/01.

Références

André Poinsot.

Problèmes d'électronique logique.

Masson , 1994. ISSN : 2225844518.

Bibliothèque centrale, Université de Jijel, Cote :621/270.

Paolo Zanella, Yves Ligier.

Architecture et technologie des ordinateurs.

Dunod, 1998. ISBN : 210003801X.

Bibliothèque centrale, Université de Jijel, Cote :004/258.

Habiba Drias-Zerkaoui.

Introduction à l'architecture des ordinateurs.

Office des publications universitaires, 2006.

Bibliothèque centrale, Université de Jijel, Cote :002/08.