UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

Ecole Supérieure Polytechnique

Département Génie Informatique

Présentation du cours

Rappels : Langage machine et logique combinatoire

DIC 1 Informatique 2016-2017

M. Ousmane KHOUMA

Nombres signés (1/2)

La plupart des dispositifs numériques traitent également des nombres négatifs. Deux symboles complémentaires sont pris en compte : $+$ et $-$.

Un bit indiquant si le nombre est négatif ou positif : c'est le bit de signe.

Le bit de signe est le premier à gauche de l'ensemble de bits représentant le nombre binaire.

Exemple : $+(47)_{10} = 0101111$ et $-(47)_{10} = \mathbf{1}101111$.

➢ Complément à 1 d'un nombre binaire

Complémenter un chiffre binaire à 1 revient à remplacer 0 par 1 et 1 par 0. Exemple -47 s'écrit $\mathbf{1}101111$ en notation exacte et sont complément à 1 est $\mathbf{1}010000$, le bit de signe étant inchangé.

Nombres signés (2/2)

➢ Complément à 2 d'un nombre binaire

Pour obtenir le complément à 2 d'un nombre binaire, il faut prendre le complément à 1 de ce nombre et lui ajouter 1.

Exemple : $-(47)_{10} = \mathbf{1}101111$ son complément à 1 est $\mathbf{1}010000$, son complément à 2 s'écrit $\mathbf{1}010001$.

Nombres en virgule (1/5)

➢ Représentation en virgule fixe

Un nombre décimal à virgule au rang k (k chiffres après la virgule) est défini par :

$$(N)_{10} = (\alpha_{n-1} \dots \alpha_1 \alpha_0, \beta_{-1} \dots \beta_{-k})_2$$
$$= \alpha_{n-1} \cdot 2^{n-1} + \cdots + \alpha_1 \cdot 2^1 + \alpha_0 \cdot 2^0 + \beta_{-1} \cdot 2^{-1} + \cdots + \beta_{-k} \cdot 2^{-k}$$

➢ Représentation en virgule flottante

Le nombre N est représenté sous la forme :

exposant	mantisse

1ière approche

Soit $N = a_3a_2a_1a_0, a_{-1}a_{-2}a_{-3}$: N peut se noter : $(a_6a_5a_4a_3a_2a_1a_0).2^{-3}$

$$\begin{cases} exposant = -3 \\ mantisse = a_6a_5a_4a_3a_2a_1a_0 \end{cases}$$

L'exposant est noté en complément à 2 en mémoire du calculateur.

Nombres en virgule (2/5)

Exemple 1 :

Soit la mémoire de taille suivante :

4 bits	12 bits
exposant	**Mantisse**

Coder la valeur 26,75 en virgule flottante.

$$(26{,}75)_{10} = (11010,110)_2$$

$$(11010,110)_2 = (11010110).2^{-3}$$

1101	**000011010110**

$exposant = -3$ et $mantisse = 000011010110 = 214$

$$26,75 = 214.2^{-3}$$

Nombres en virgule (3/5)

2ième approche

C'est la méthode inverse de la précédente : on considère que le bit le plus à gauche de la mantisse a pour poids 2^{-1}.

Soit $N = a_3a_2a_1a_0, a_{-1}a_{-2}a_{-3}$: N peut aussi se noter : $(0, a_{-1}a_{-2}a_{-3}a_{-4}a_{-5}a_{-6}a_{-7}).2^4$

Exemple 2 :

Même exemple que précédemment :

$(26{,}75)_{10} = (11010,110)_2 \longrightarrow (0{,}11010110).2^5$

0101	**110101100000**

Remarque 1 :

Les ordinateurs utilisent cette représentation avec 32 bits pour la mantisse et 8 bits pour l'exposant. En général, on utilise la représentation inverse, avec le bit le plus à gauche égal à 1, soit une mantisse normalisée $\Rightarrow 0{,}5 \leq M < 1$.

Nombres en virgule (4/5)

➢ Représentation IEEE 754

Le standard IEEE 754 définit deux formats : les nombres en ***simple précision*** sur 32 bits, les nombres en ***double précision*** sur 64 bits.

Un nombre N de 32 bits est représenté sous la forme :

s	Exposant	mantisse

Où le signe « s » est codé sur 1 bit, l'exposant est codé sur 8 bits en code relatif à 127, et la mantisse sur 23 bits.

Un nombre de 64 bits (double précision) utilise la même représentation à ceci près que la taille de l'exposant est portée à 11 bits en code relatif à 1023, et celle de la mantisse à 52 bits.

Nombres en virgule (5/5)

Exemple 3 :

$0{,}5 = 2^{-1}(1+0)$

Code (0,5) =

0	**01111110**	**000…0**
s	E	m

= ***3F0000***

$1{,}5 = 2^{0}(1+2^{-1})$

Code (1,5) =

0	**01111111**	**1000…0**
s	E	m

= ***3FC0000***

Codes pondérés (1/2)

➢ Code DCB : décimal codé binaire

Le code binaire du chiffre décimal le plus élevé 9, étant 100, mot de quatre bits, chaque chiffre en notation binaire est codé sur 4 bits. D'où le tableau suivant :

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

Exemple : le nombre décimal 2583 s'écrit, en utilisant le code DCB

0010 0101 1000 0011

Codes pondérés (2/2)

Exemple : $(148)_{10} + (34)_{10}$ en DCB

$$\begin{array}{r} 148 \\ +\ 34 \\ \hline 182 \end{array} \qquad \begin{array}{r} 0001\ \ 0100\ \ 1000 \\ +\quad 0000\ \ 0011\ \ 0100 \\ \hline 0001\ \ 0111\ \ 1100 \\ (1\qquad 7\qquad "?")_{10} \end{array}$$

Pour effectuer la correction, il faut ajouter 6 soit 0110 en DCB, d'où.

$$\begin{array}{r} 148 \\ +\ 34 \\ \hline 182 \end{array} \qquad \begin{array}{r} 0001\ \ 0100\ \ 1000 \\ +\quad 0000\ \ 0011\ \ 0100 \\ \hline 0001\ \ 0111\ \ 1100 \\ +\qquad\qquad\qquad\ \ 0110 \\ \hline 0001\ \ 1000\ \ 0010 \\ (1\qquad 8\qquad 2)_{10} \end{array}$$

NB : 6 est appelé **facteur de correction.**

Codes non pondérés (1/1)

➢ Code Alphanumérique

Un ordinateur est prévu pour traiter des informations non numériques, c'est-à-dire qu'il doit « reconnaitre » des caractères de l'alphabet, des caractères spéciaux, des chiffres, … Tous ces éléments sont associés à un code appelé alphanumérique.

Le code le plus connu est le code ASCII (American Standard Code for Information Interchange). C'est un code quasi universel pour la transmission de l'information.

Chaque symbole nécessite au moins 7 bits ce qui donne $2^7 = 128$ combinaison binaires différentes.

Le plus souvent ce code est défini avec 8 bits, le huitième étant généralement un bit de parité permettant de corriger des erreurs de transmission.

Codes de détection d'erreurs (1/1)

➢ Bit de parité pair

Le bit supplémentaire est fixé à une valeur (0 ou 1) telle que, pour chaque « mot », le nombre total de 1, y compris le bit de parité, soit pair.

Exemples : Un circuit numérique doit transmettre le caractère « , » (virgule) ; le code ASCII de ce caractère est 0101100 : il est formé de trois 1, le bit de parité doit donc être 1 pour que le nombre total de 1 soit pair. Le code ASCII du caractère « , », avec le bit de parité, est : 10101100.

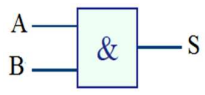
Si le caractère à transmettre est « D », dont le code ASCII est 1000100, le bit de parité doit être 0 pour que le nombre total de 1 soit pair. Le code ASCII du caractère « D », avec bit de parité, est : 01000100.

Logique combinatoire (1/8)

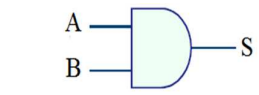
➢ L'opération ET (AND)

$S = A\,B$ ou $S = A.B$

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1



Norme IEC (nouvelle)

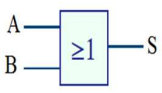


Norme IEE (ancienne)

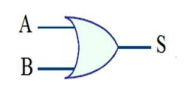
➢ L'opération ET (AND)

$S = A + B$

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1



Norme IEC (nouvelle)



Norme IEEE (ancienne)

➢ L'opération NON (NOT)

$S = \overline{A}$

A	S
0	1
1	0

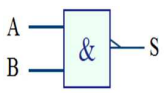
Norme IEC (nouvelle) Norme IEEE (ancienne)

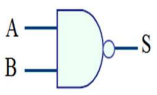
Logique combinatoire (2/8)

➢ L'opération NON-ET (NAND)

$S = (\overline{AB})$ ou $S = (\overline{A.B})$

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

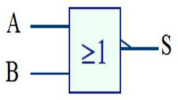


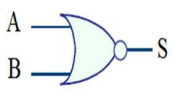


➢ L'opération NON-OU (NOR)

$S = \overline{A + B}$

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

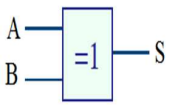


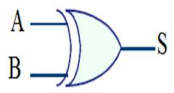


➢ L'opération OU-EXCLUSIF (XOR)

$S = A \oplus B = \overline{A}B + A\overline{B}$

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0



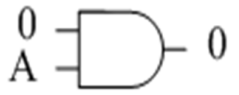


Logique combinatoire (3/8)

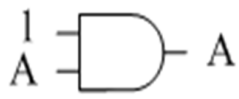
➢ Théorèmes de base

Multiplication

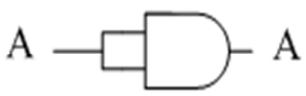
$A \cdot 0 = 0$



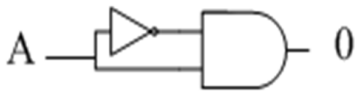
$A \cdot 1 = A$



$A \cdot A = A$



$A \cdot \bar{A} = 0$



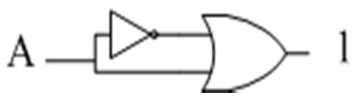
Addition

$A + 0 = A$

$A + 1 = 1$

$A + A = A$

$A + \bar{A} = 1$



Logique combinatoire (4/8)

➢ Théorèmes de Morgan

Les théorèmes de De Morgan sont utiles pour convertir des sommes en des produits, et vice-versa.

Théorèmes pour 2 variables :

$\overline{A + B} = \bar{A}.\bar{B}$ $\qquad\qquad$ $\overline{A.B} = \bar{A} + \bar{B}$

Théorèmes pour N variables :

$$\overline{\sum X_k} = \prod \bar{X}_k \qquad\qquad \overline{\prod X_k} = \sum \bar{X}_k$$

➢ Table de Karnaugh

La table de Karnaugh permet d'écrire une équation booléenne, de la simplifier et de déduire une implémentation des composants pour le montage correspondant.

Une table de Karnaugh est constituée de lignes et de colonnes en nombre tel que la table soit la plus « carrée » possible. Avec un nombre n pair de variables d'entrée ($n = 2p$), la table sera formée de 2^p lignes et 2^p colonnes ($n = 4$, nous avons 2^2 lignes et 2^2 colonnes). Avec un nombre n impair de variables d'entrée ($n = 2p + 1$), la table sera formée de 2^p lignes et 2^{p+1} colonnes ou de 2^{p+1} lignes et 2^p ($n = 3$, nous avons 2 lignes et 4 colonnes ou 4 lignes et 2 colonnes).

➢ Table de Karnaugh

✓ **Avec n = 3 : A, B et C sont les entrées**

A \ BC	$\overline{B}\,\overline{C}$ 00	$\overline{B}\,C$ 01	$B\,C$ 11	$B\,\overline{C}$ 10
$\overline{A}$ 0	0	1	3	2
A 1	4	5	7	6

✓ **Avec n = 4 : A, B, C, D sont les entrées**

AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}\,D$ 01	$B\,C$ 11	$B\,\overline{C}$ 10
$\overline{A}\,\overline{B}$ 00	0	1	3	2
$\overline{A}\,B$ 01	4	5	7	6
$A\,B$ 11	12	13	15	14
$A\,\overline{B}$ 10	8	9	11	10

Logique combinatoire (7/8)

➢ Table de Karnaugh

- ✓ A partir de la table, on simplifie en regroupant les 1 adjacents.
- ✓ Les 1 adjacents sont mis en évidence par l'ordre utilisé pour former la table.
- ✓ La taille d'un groupe est un multiple de 2^k (1,2, 4, 8, ...).
- ✓ Le groupe est soit rectangulaire ou carré.
- ✓ Former les plus gros groupes possibles.
- ✓ Un 1 peut faire partie de plusieurs groupes.

Logique combinatoire (8/8)

➢ Table de Karnaugh

$S = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,B\,C$

Entrées			Sortie
A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

A \ BC	00	01	11	10
0	1	0	1	1
1	1	0	0	0

$$\mathbf{S = \overline{B}\,\overline{C} + \overline{A}B}$$

$S = \overline{A}\,\overline{B}\,\overline{C}\,D + \overline{A}\,\overline{B}\,C\,\overline{D} + \overline{A}\,\overline{B}\,C\,D + \overline{A}\,B\,\overline{C}\,D$
$+ \overline{A}\,B\,C\,D + A\,\overline{B}\,\overline{C}\,D + A\,\overline{B}\,C\,\overline{D} + A\,B\,\overline{C}\,D$

AB \ CD	00	01	11	10
00	0	1	1	1
01	0	1	1	0
11	0	1	0	0
10	0	1	0	1

$$\mathbf{S = \overline{C}D + \overline{A}D + \overline{B}C\overline{D}}$$