SYS2041 – Électronique numérique
Cours 1 : Introduction et numération

Alexandre BRIÈRE

Qu'est-ce que l'électronique numérique ?

Traitement des phénomènes discrets

$\Rightarrow$ Position d'un interrupteur, comptage, etc.

Immunité (partielle) aux bruits

$\Rightarrow$ Transport, stockage et utilisation des données

Calculs et manipulations aisés

$\Rightarrow$ Multiplexage, compression de données, etc.

Interface avec les ordinateurs

$\Rightarrow$ Réalisés eux-mêmes grâce à l'électronique numérique

Objectifs du module

- Comprendre les bases de la logique combinatoire et séquentielle
- Analyser et concevoir des systèmes élémentaires à base de composants numériques

Organisation du module

- 33h de cours/TD
- un partiel le 08/11/2018
- un examen en fin de semestre

Plan du cours

Plan du module

Numération en base b

Un système de numération repose sur les 2 conventions suivantes :

- Tout nombre entier strictement inférieur à b est représenté par un symbole unique d'un ensemble E
  - Système décimal
    $b = 10$ et $E = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - Système octal
    $b = 8$ et $E = \{0, 1, 2, 3, 4, 5, 6, 7\}$
  - Système binaire
    $b = 2$ et $E = \{0, 1\}$

- Tout nombre entier x admet un développement unique de la forme :

$$x = a_n b^n + a_{n-1} b^{n-1} + ... + a_1 b^1 + a_0 b^0$$
$$\text{où } a_i \in E$$

- Cette notation peut être abrégée sous la forme suivante :

$$x : (a_n a_{n-1} ... a_1 a_0)_b$$

Numération binaire

- Base : 2
- Utilisation de deux symboles : 0 et 1
- Un nombre binaire à donc la forme suivante :

$$(a_n a_{n-1} ... a_1 a_0)_2 = (a_n 2^n + a_{n-1} 2^{n-1} + ... + a_1 2^1 + a_0 2^0)_{10}$$

- Exemple :

$$(1010)_2 = (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0)_{10} = 8 + 2 = 10$$

- On lit de gauche à droite :
$\Rightarrow$ des bits de poids fort (MSB) aux bits de poids faible (LSB)

Récapitulatif

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

Numération hexadécimale

- Base : 16
- Développée pour les ordinateurs :
  1 octet (8 bits) peut représenter 256 (16×16) valeurs différentes
- Utilisation de 16 symboles : 0,...,9,A,B,C,D,E,F
- Un nombre hexadécimale à donc la forme suivante :
  $(N)_{16} = a_n a_{n-1} ... a_1 a_0$
  $(N)_{16} = (a_n 16^n + a_{n-1} 16^{n-1} + ... + a_1 16^1 + a_0 16^0)_{10}$
- Exemple :
  $$\begin{aligned}(3A)_{16} &= (3 \times 16^1 + A \times 16^0)_{10} \\ &= (3 \times 16 + 10 \times 1)_{10} \\ &= (48 + 10)_{10} \\ &= (58)_{10}\end{aligned}$$

Récapitulatif

Décimale	Binaire	Hexadécimale
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Changement de base : binaire et hexadécimale

- Binaire vers hexadécimale
  - Découpage du nombre binaire par paquet de 4
  - Exemple :

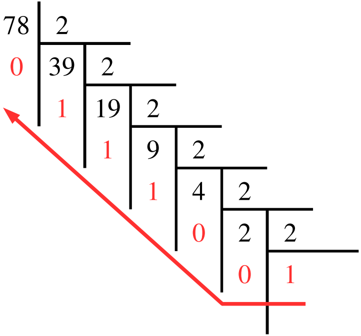
$$(10110001)_2 = \begin{array}{c|c} 1011 & 0001 \\ B & 1 \end{array} = (B1)_{16}$$

- Hexadécimale vers binaire
  - Conversion digit par digit
  - Exemple :

$$(A7BF)_{16} = \begin{array}{c|c|c|c} A & 7 & B & F \\ 1010 & 0111 & 1011 & 1111 \end{array} = (1010011110111111)_2$$

- Soit x un nombre entier : on divise x par 2, puis le quotient obtenu par 2 et ainsi de suite jusqu'à obtenir un quotient nul
- On écrit de gauche à droite, du dernier reste au premier
- Exemple :

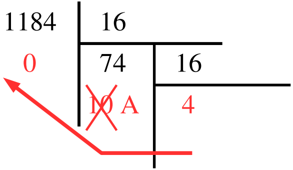
  $(78)_{10} = (1001110)_2$



Changement de base : décimale vers hexadécimale

- Soit x un nombre entier : on divise x par 16, puis le quotient obtenu par 16 et ainsi de suite jusqu'à obtenir un quotient nul
- On écrit de gauche à droite, du dernier reste au premier
- Exemple :

  $(1184)_{10} = (4A0)_{16}$



Et les nombres négatifs dans tout ça ?

Utilisation du complément à 1 et du complément à 2

- Complément à 1 :
  - On remplace les 1 par des 0 et inversement
  - Exemple : complément à 1 de 1011

    $1011 \Rightarrow 0100$

- Complément à 2 :
  - Méthode 1 : on calcul le complément à 1 et on ajoute 1
  - Méthode 2 : de droite à gauche, on garde tout les 0 et le premier 1 puis on inverse les autres bits
  - Exemple : complément à 2 de 1100

    $1100 \Rightarrow 0011 \Rightarrow 0011 + 1 = 0100$

    $1100 \Rightarrow \mathbf{0}100$

Nombres entiers signés : représentation "intuitive"

- Utilisation du bit de poids fort comme bit de signe
  - $0 \Rightarrow$ nombre positif
  - $1 \Rightarrow$ nombre négatif
- Suivi de la valeur absolue en binaire
- Exemple :
  - $+25 = \mathbf{0}11001$
  - $-25 = \mathbf{1}11001$

Exemple avec 3 bits

Binaire	Décimal
000	$+0$
001	$+1$
010	$+2$
011	$+3$
100	-0
101	-1
110	-2
111	-3

Problème : deux codages différents pour zéro !

Nombres entiers signés – complément à 1

- Utilisation du bit de poids fort comme bit de signe
  - $0 \Rightarrow$ nombre positif
  - $1 \Rightarrow$ nombre négatif
- Complément à 1 du nombre positif pour les nombres négatifs
- Exemple :

  $+25 = \mathbf{0}11001$

  $-25 = \mathbf{1}00110$

Exemple avec 3 bits

Binaire	Décimal
000	$+0$
001	$+1$
010	$+2$
011	$+3$
100	-3
101	-2
110	-1
111	-0

Problème : deux codages différents pour zéro !

Passage du complément à 1 vers la valeur décimale

- Pour les nombres positifs
  - Même méthode que pour un nombre non-signé
  - Exemple :

    $\mathbf{0}010111 = 1 \times 16 + 1 \times 4 + 1 \times 2 + 1 \times 1 = 23$

- Pour les nombres négatifs :
  - Affecter une valeur négative au poids du bit de signe
  - Additionner les poids des bits à 1
  - Ajouter 1 au résultat
  - Exemple :

    $\mathbf{1}101000 = \underbrace{-64} + \underbrace{32 + 8} + \underbrace{1} = -23$

Nombres entiers signés – complément à 2

- Utilisation du bit de poids fort comme bit de signe
  - $0 \Rightarrow$ nombre positif
  - $1 \Rightarrow$ nombre négatif
- Complément à 2 du nombre positif
- Exemple :
  - $+25 = \mathbf{0}11001$
  - $-25 = \mathbf{1}00111$

Exemple avec 3 bits

Binaire	Décimal
000	$+0$
001	$+1$
010	$+2$
011	$+3$
100	-4
101	-3
110	-2
111	-1

Plus de problème avec zéro !

Passage du complément à 2 vers la valeur décimale

- Pour les nombres positifs
  - Même méthode que pour un nombre non-signé
  - Exemple :

    $\mathbf{0}1010110 = 64 + 16 + 4 + 2 = 86$

- Pour les nombres négatifs :
  - Affecter une valeur négative au poids du bit de signe
  - Additionner les poids des bits à 1
  - Exemple :

    $\mathbf{1}0101010 = \underbrace{-128} + \underbrace{32 + 8 + 2} = -86$

Et les nombres à virgule alors ?

Même méthode que pour les nombres entiers !

Ou presque...

- Utilisation des symboles 0 et 1
- Un nombre binaire à virgule a la forme suivante :
  $(a_n a_{n-1} ... a_1 a_0 , a_{-1} ... a_{m-1} a_m)_2 =$
  $(a_n 2^n + ... + a_1 2^1 + a_0 2^0 + a_{-1} 2^{-1} + ... + a_m 2^{-m})_{10}$
- Exemple :

$$\begin{aligned}(10,11)_2 &= (1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2})_{10} \\ &= (2 + 0,5 + 0,25)_{10} \\ &= (2,75)_{10}\end{aligned}$$

Arithmétique binaire : Addition

- $0 + 0 = 0 \Rightarrow$ somme à 0 et pas de retenue
- $0 + 1 = 1 \Rightarrow$ somme à 1 et pas de retenue
- $1 + 0 = 1 \Rightarrow$ somme à 1 et pas de retenue
- $1 + 1 = 10 \Rightarrow$ somme à 0 et retenue à 1
- Exemple :

  $11 + 1 = 100$

- $0 - 0 = 0 \Rightarrow$ différence à 0 et pas d'emprunt
- $1 - 1 = 0 \Rightarrow$ différence à 0 et pas d'emprunt
- $1 - 0 = 1 \Rightarrow$ différence à 1 et pas d'emprunt
- $(1)0 - 1 = 1 \Rightarrow$ différence à 1 avec emprunt de 1
- Exemples :

  $011 - 001 = 010$

  $011 - 010 = 001$

  $101 - 011 = 010$

Arithmétique binaire : Multiplication

- $0 \times 0 = 0$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $1 \times 1 = 1$
- Exemples :

  $011 \times 011 = 1001$

  $111 \times 101 = 100011$

Arithmétique binaire : Division euclidienne

- $0 \div 1 = 0$
- $1 \div 1 = 1$
- $0 \div 0 \Rightarrow$ INTERDIT
- $1 \div 0 \Rightarrow$ INTERDIT
- Exemples :

  $110 \div 11 = 10$

  $110 \div 10 = 11$

Autres codages binaires : BCD

Binary Coded Decimal (BCD)

- Utilisé pour l'affichage de nombre
- Chaque digit décimal est écrit en binaire
- Exemple

  $(1239)_{10} = (0001\,0010\,0011\,1001)_{BCD}$

$\Rightarrow$ Chaque octet n'est utilisé que pour coder 10 valeurs au lieux de 16

Autres codages binaires : Gray

Code de Gray

- Aussi appelé code binaire réfléchi
- On ne change qu'un bit d'un nombre au suivant

Décimal	Binaire	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Références

[1] Sébastien Gageot et Franck Crison :
SYS2041 : Systèmes numériques (Laval).

[2] Thomas Floyd :
Systèmes numériques.
Éditions Reynald Goulet, 2018.