

LifLF – Architecture matérielle

*Sylvain Brandel*

2021 – 2022

[sylvain.brandel@univ-lyon1.fr](mailto:sylvain.brandel@univ-lyon1.fr)

CM 2

# **CODAGE DES DONNÉES EN MACHINE**

***PARTIE 1 – CODAGE DES ENTIERS***

# Information

- (Transistors : plus tard)
- Détection de **deux états**
  - Haut  $\geq$  réf. haute
  - Bas  $\leq$  réf. basse
- Physique : différence de 1V (ordre de grandeur)
- Convention
  - L'un : 0
  - L'autre : 1
- Représentation **binaire**
  - bit
  - Par mots de 4 bits : représentation hexadécimale
  - Mot de 8 bits : octet (Byte)
    - Mb : Mega bit      MB : Mega Byte

# Codage des entiers naturels

## *Notation positionnelle*

- $\beta \in \mathbb{N}, \beta > 1$  : **base**
- **Représentation positionnelle en base  $\beta$**  de  $n \in \mathbb{N}$  :

$$(x_{p-1}x_{p-2} \dots x_1x_0)_\beta := \sum_{i=0}^{p-1} x_i \beta^i$$

- $x_i \in \{0, 1, \dots, \beta - 1\}$  : **chiffres** de l'écriture de  $n$  en base  $\beta$ 
  - $\beta = 2$  : chiffres 0 et 1
  - $\beta = 10$  : chiffres de 0 à 9
  - $\beta = 16$  : **chiffres** de 0 à F
- $p$  : nombre de chiffres nécessaires pour écrire  $n$
- Ex :  $(5134)_{10} = 5 \cdot 10^3 + 1 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$

# Codage des entiers naturels

## Changement de base

- Avec notation positionnelle
- $\beta \in \mathbb{N}, \beta > 1$  : base de départ,  $\gamma \in \mathbb{N}, \gamma > 1$  : base d'arrivée
- Toujours possible :
  - Conversion  $x_i$  et  $\beta$  vers écriture en base  $\gamma$
  - Calcul de  $\sum_{i=0}^{p-1} x_i \beta^i$  avec opérations en base  $\gamma$
- Ecriture de  $n$  en base  $\gamma$  : calcul dans la base d'arrivée  $\gamma$
- Ex : Conversion **binaire vers décimal** de  $n = (10100)_2$  :
  - Ajout des puissances de 2 correspondant aux bits non nuls

Chiffre	1	0	1	0	0
Position	4	3	2	1	0
Poids	$2^4$	0	$2^2$	0	0

– Donc  $(10100)_2 = 2^4 + 2^2 = 20$

- Ex : Conversion **décimal vers binaire** de  $n = (95)_{10}$  et  $(423)_{10}$

# Codage des entiers naturels

## Changement de base

- Avec **divisions euclidiennes successives**
- Reste de la division euclidienne de  $n$  par  $\beta$  :
  - Chiffre de poids faible dans l'écriture de  $n$  en base  $\beta$

$$\begin{aligned}n &= x_{p-1}\beta^{p-1} + x_{p-2}\beta^{p-2} + \dots + x_2\beta^2 + x_1\beta^1 + x_0 \\ &= \underbrace{(x_{p-1}\beta^{p-2} + x_{p-2}\beta^{p-3} + \dots + x_2\beta^1 + x_1)}_{\text{quotient}}\beta + \underbrace{x_0}_{\text{reste}}\end{aligned}$$

avec  $0 \leq x_0 \leq \beta$

- En d'autres termes,  $n \bmod \beta = x_0$ 
  - Les chiffres de  $n$  sont obtenus par divisions euclidiennes successives
  - Arrêt au premier quotient nul. Chiffres de poids faible d'abord !
- Ex : Conversion **décimal vers binaire** de  $(95)_{10}$  et  $(423)_{10}$
- Ex : Conversion **décimal vers octal** de  $(3452)_{10}$

# Codage des entiers naturels

## *Changement de base*

- Entre bases 2, 8, 16 : **conversions directes**
- $8 = 2^3 \rightarrow$  chiffre octal : entier sur trois bits

$$(x_8x_7x_6x_5x_4x_3x_2x_1x_0)_2 = \underbrace{(x_8x_7x_6)_2}_{y_2}8^2 + \underbrace{(x_5x_4x_3)_2}_{y_1}8^1 + \underbrace{(x_2x_1x_0)_2}_{y_0}8^0 = (y_2y_1y_0)_8$$

- Ex : Conversion **octal vers binaire** de  $(34521)_8$
- Ex : Conversion **hexadécimal vers binaire** de  $(9A6E)_{16}$

# Entiers naturels

## Représentation *machine*

- Nombre de bits  $p$  fixé pour chaque format de codage (8, 16, 32, 64)
- Si résultat sur plus de  $p$  bits :
  - Obtenu :  $p$  bits de poids faible du résultat exact
  - Drapeau de dépassement de capacité de l'UAL
- Les calculs continuent !
- $m$  codé sur  $q \geq p$  bits

$$m = \sum_{i=0}^{q-1} m_i 2^i = \underbrace{\left( \sum_{i=p}^{q-1} m_i \right) 2^p}_{\text{quotient de } m \text{ par } 2^p} + \underbrace{\sum_{i=0}^{p-1} m_i 2^i}_{\text{reste}}$$

- Opération arithmétique sur entiers naturels
  - Résultat  $m$  placé sur  $p$  bits
  - Du coup résultat obtenu :  $m \bmod 2^p$

# Codage des entiers relatifs

- Mots de n bits  $\rightarrow$  différents états  $w = b_{n-1} \dots b_2 b_1 b_0$
- Non signés :
  - Notation positionnelle :  $\llbracket w \rrbracket = \sum_{i=0}^{n-1} b_i 2^i$
- Signés :
  - Signe + valeur absolue :  $\llbracket w \rrbracket = (-1)^{b_{n-1}} \sum_{i=0}^{n-2} b_i 2^i$
  - Complément à 1 :  $\llbracket v \rrbracket = -\llbracket w \rrbracket = \overline{b_{n-1}} \dots \overline{b_2} \overline{b_1} \overline{b_0}$
  - Complément à 2 :  $\llbracket w \rrbracket = -b_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i$
  - Biais N :  $\llbracket w \rrbracket = \sum_{i=0}^{n-1} b_i 2^i - N$



# Codage des entiers relatifs

## Complément à 2

- $n$  entier relatif,  $-2^{p-1} \leq n \leq 2^{p-1} - 1$ , à coder sur  $p$  bits
- Notation en complément à 2 sur  $p$  bits

$$n = (c_{p-1}c_{p-2} \dots c_1c_0)_2$$

$$(c_{p-1}c_{p-2} \dots c_1c_0)_2 := -c_{p-1}2^{p-1} + \sum_{i=0}^{p-2} c_i 2^i$$

- Propriétés :
  - $n \geq 0$  ssi  $c_{p-1} = 0$
  - $n \leq 0$  ssi  $c_{p-1} = 1$

# Codage des entiers relatifs

## Complément à 2

- Interprétation
  - Considérer le bit le plus à gauche de **poids négatif** ( $-2^{p-1}$ )

- Ex :  $p = 8$ , **valeur décimale** codée par  $(10000011)_2$

$$(10000011)_2 = -128 + 3 = (-125)_{10}$$

- Ex :  $p = 8$ , coder  $(-120)_{10}$  en **complément à 2**

$$(-120)_{10} = -128 + 8 = (10001000)_2$$

# Entiers relatifs

## Complément à 2

- $m = (c_m)_{\bar{2}}$  et  $n = (c_n)_{\bar{2}}$  en complément à 2 sur p bits
- **Addition**
  - Sans dépassement : codage de  $m + n = \text{codage de } (c_m + c_n) \bmod 2^p$  en tant qu'**entier naturel**
  - Avec dépassement : résultat faux
  - Dépassement
    - m et n de signes opposés : dépassement impossible
    - m et n de même signe : dépassement ssi signe résultat  $\neq$  signe de n
- **Opposé**
  - Sans dépassement : codage de  $-n$  en complément à 2 sur p bits
$$(\overline{c_{p-1}} \dots \overline{c_1} \overline{c_0})_2 + 1 \bmod 2^p$$
comme **entier naturel**
- Ex :  $p = 7$ ,  $(36)_{10} = (0100100)_{\bar{2}}$      $(-36)_{10} = (1011100)_{\bar{2}}$