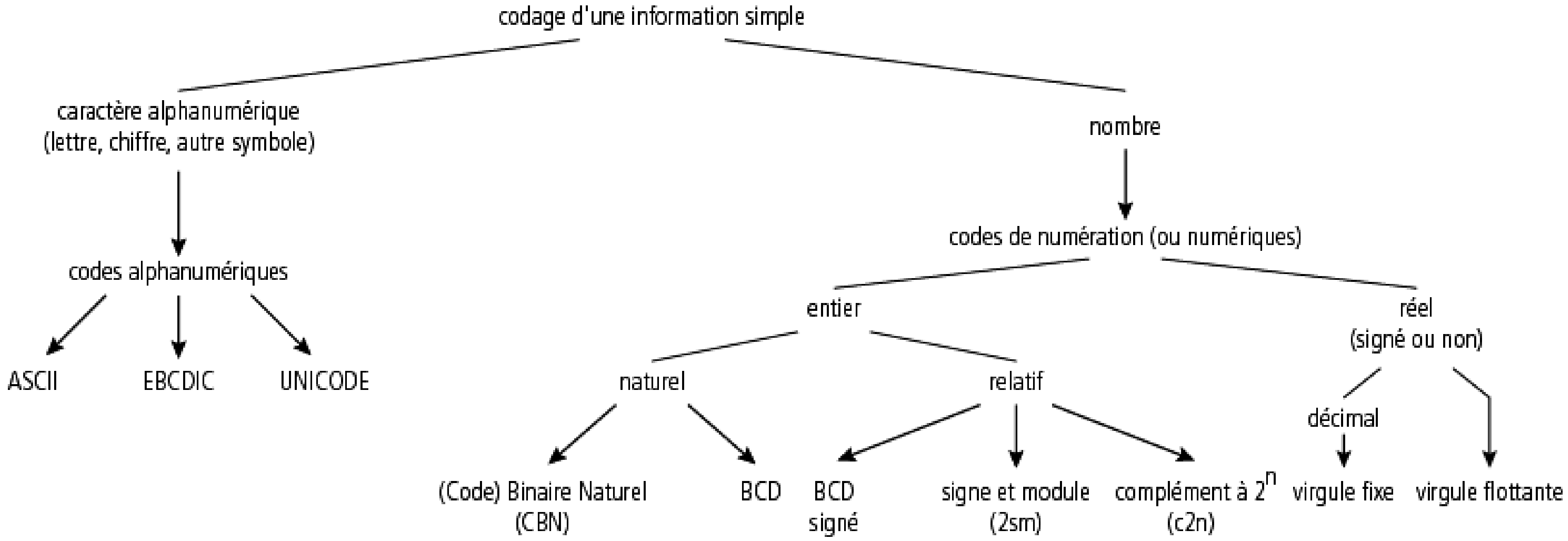
Architecture des ordinateurs

6 - Représentation des entiers en machine

Philippe Darche
IUT Paris Descartes

Représentation de l'information en machine (rappel)

- Des 0 et des 1 !
  - B = 2 (base binaire)



Codes de numération (ou numériques)

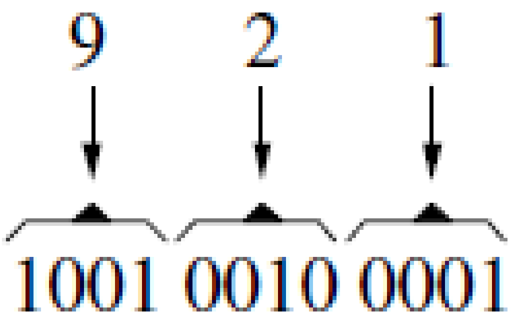
- Codage uniquement des nombres
- Plusieurs familles
  - ensemble des entiers naturels ($\mathbb{N}$) et relatifs ($\mathbb{Z}$)
  - ensemble des décimaux ($\mathbb{D}$)
  - ensemble des réels ($\mathbb{R}$)

Représentations d'un entier naturel

- (Code) Binaire Naturel (ou CBN)
  - voir cours sur la numération
- Décimal Codé Binaire (DCB)
  ou BCD (*Binary Coded Decimal*) ou code 8421
  - compacté ou condensé (*packed*)
    - un chiffre décimal = un quartet (demi-octet)
  - non compacté, non condensé ou étendu (*unpacked*)
    - un chiffre décimal = un octet !
  - à ne pas confondre avec le binaire naturel

BCD compacté

- Exemple



- Propriétés
  - \+ conversion élémentaire et lecture directe
  - \- place mémoire occupée élevée par rapport à CBN
  - \- calculs arithmétiques plus complexes

BCD non compacté

- Exemple

$$\overbrace{0000\ 1000}^{8}\ \overbrace{0000\ 0001}^{1}$$

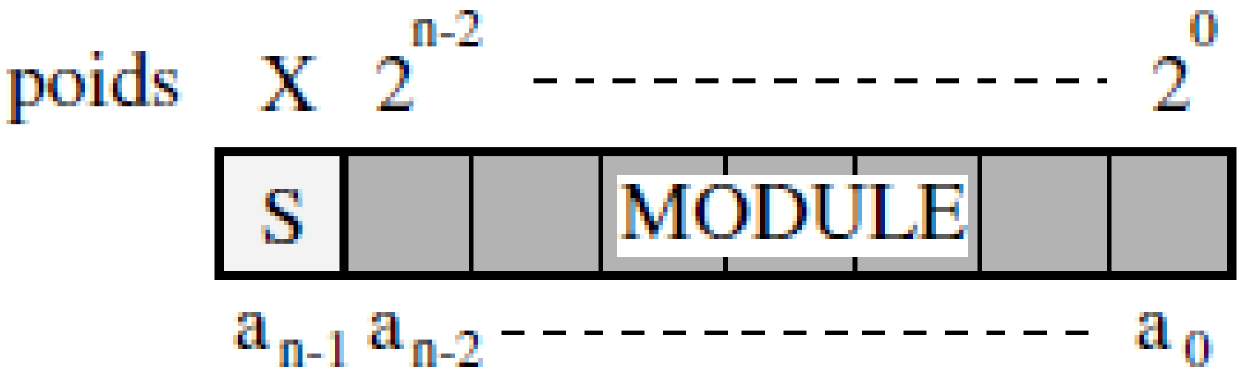
- Propriétés
  - \+ conversion élémentaire et lecture directe
  - \- place mémoire occupée encore plus grande qu’en version compactée
  - \- calculs arithmétiques plus complexes

Représentations d'un nombre entier relatif

- Entier relatif
  - signe : + ou -
  - module = valeur absolue
    - toujours positif
- Problème : comment coder le signe ?
  - la réponse : un seul bit avec - → 1 et + → 0
    - intérêt : test du signe sur un seul bit
- Représentations usuelles
  - signe et module (2sm), appelé aussi « signe et valeur absolue »
  - complément à 2^n
  - BCD signé

Représentation en signe et module (2sm)

- Constat :
  - entier relatif = entier naturel avec un signe !
- Format n bits



Exemples de codage

- Format n = 8 bits
  - +1 : 0 000 0001
  - -1 : 1 000 0001
  - +0 : 0 000 0000
  - -0 : 1 000 0000 (absurde)
- Valeurs extrêmes pour un format n = 4 bits
  - +7 : 0111
  - -7 : 1111

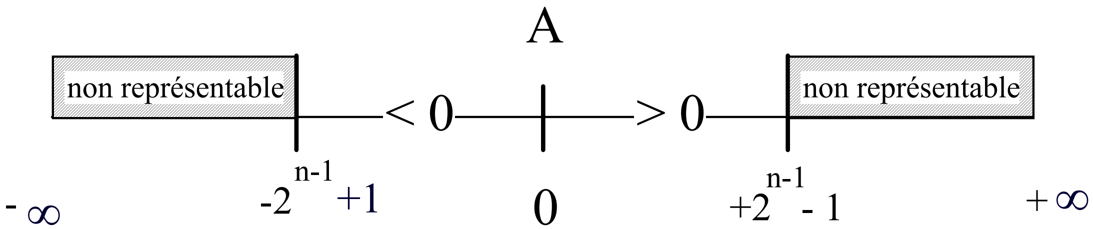
Représentation en signe et module (2sm)

- Avantages/inconvénients
  - \+ valeur directement compréhensible par l'utilisateur
  - \+ calcul simple de l'opposé arithmétique
  - \+ calcul simple du signe pour la multiplication et la division signées
  - \- le chiffre zéro a deux représentations
    - $\Rightarrow$ perte d'une valeur pour le codage à cause de la représentation de « -0 »
  - \- $(+x) + (-x) \neq 0$!
  - \- bit de signe S non pondéré

Représentation en signe et module (2sm)

- Etendue des valeurs

$$A_{10<2sm} \in [-(2^{n-1}-1), 2^{n-1}-1]$$



- Formule de décomposition

$$A_{10<2sm} = (-1)^S \times \left(a_{n-2} \times 2^{n-2} + \ldots + a_1 \times 2^1 + a_0 \times 2^0\right)$$

$$= (-1)^S \times \sum_{i=0}^{n-2} a_i \times 2^i$$

Conversion du nombre A

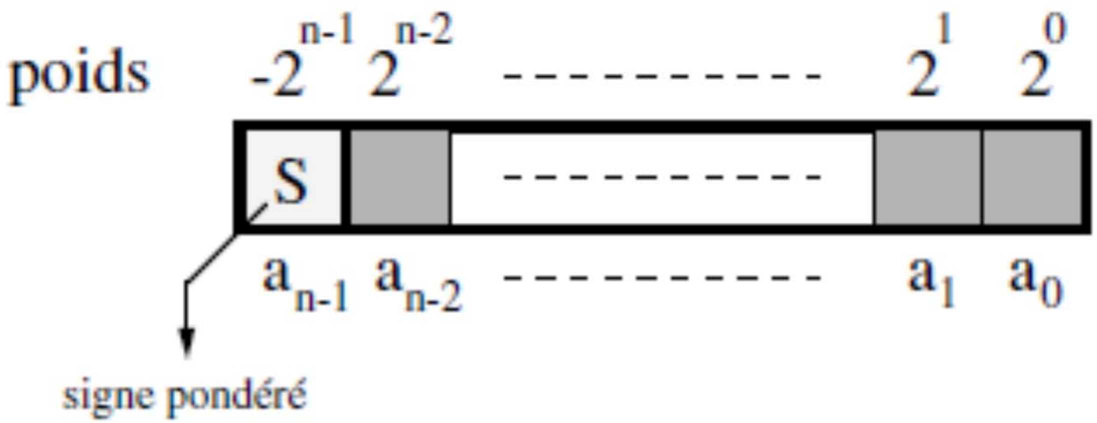
- $A_{10} \rightarrow A_{2sm}$ (au format n)
  - vérification de l'appartenance de A à l'étendue des valeurs
    - fonction du format n
  - codage du signe S (bit a_{n-1})
    - si $A \leq 0$ alors $a_{n-1} = 1$
    - si $A \geq 0$ alors $a_{n-1} = 0$
  - codage de la valeur absolue (bits a_{n-2} à a_0)
    - conversion en binaire naturel
- Conversion inverse
  - utilisation de la formule de décomposition pour le module puis ajout du signe

Opérations en signe et module

- Traiter les signes à part
- Traiter les modules à part
  - calculs en binaire naturel

Représentation en complément à 2^n

- Objectif :
  - supprimer les points négatifs du signe et module
- La réponse :
  - donner au signe un poids
- Format n bits



Exemples de codage

- Format n = 8 bits
  - +1 : 0 000 0001
  - -1 : 1 111 1111
  - 0 : 0 000 0000
- Valeurs extrêmes pour un format n = 4 bits
  - +7 : 0111
  - -8 : 1000

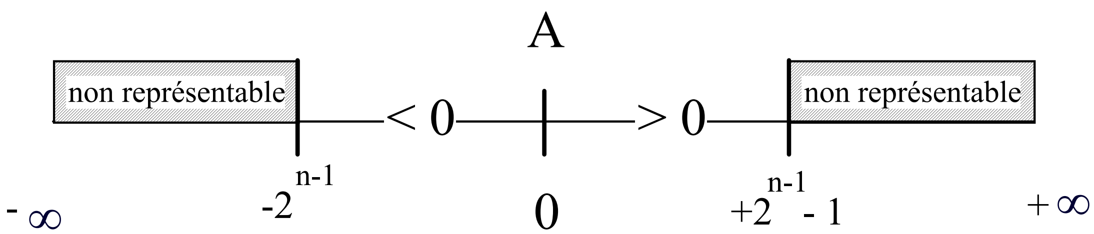
Représentation en complément à 2^n

- Avantages/inconvénients
  - \+ une seule représentation du zéro
  - \+ signe pondéré
    - pas de perte de codage
  - \+ un seul opérateur pour l'addition et la soustraction
    - utilisation de l’opposé
    - $(+x) + (-x) = 0$
  - \- valeurs négatives non compréhensibles directement

Représentation en complément à 2^n

- Etendue des valeurs

$$A_{10<c2n} \in [-2^{n-1}, 2^{n-1} - 1]$$



- Formule décomposition

$$A_{10<c2n} = \left(a_{n-1} \times -2^{n-1}\right) + \left(a_{n-2} \times 2^{n-2} + \ldots + a_1 \times 2^1 + a_0 \times 2^0\right)$$

$$= \left(a_{n-1} \times -2^{n-1}\right) + \left(\sum_{i=0}^{n-2} a_i \times 2^i\right)$$

Relation fondamentale

- Calcul de l'opposé
  - utile pour la conversion d'un nombre négatif en complément à 2^n

$$A_{c2^n} \xrightarrow{\overline{A}_{c2^n} + 1} -A_{c2^n}$$

$$A_{c2^n} \xleftarrow[\overline{-A}_{c2^n} + 1]{} -A_{c2^n}$$

Conversion du nombre A

- $A_{10} \rightarrow A_{c2n}$
  - vérification de l'appartenance de A à l'étendue des valeurs
    - fonction du format n
  - **Si** $A \geq 0$

    Alors codage de A en binaire naturel

    en respectant le format n

    Sinon

    codage en binaire naturel de $|A|$, complémentation

    et incrémentation (*i.e.* + 1)
- Conversion inverse
  - utilisation de la formule de décomposition

Arithmétique en c2n

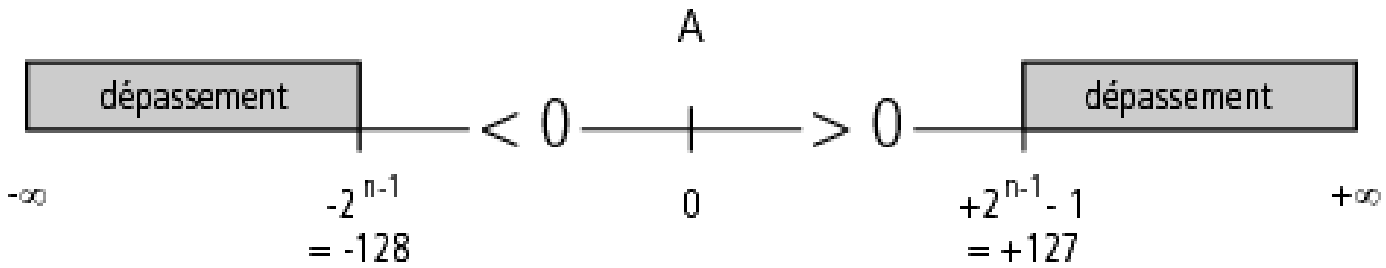
- Un exemple d'addition algébrique
  - $n = 8 \Rightarrow$ nombre $\in$ [-128,+127])

$$\begin{array}{r} -128 \\ + \quad -1 \\ \hline -129 \end{array} \qquad \begin{array}{r} +127 \\ + \quad +1 \\ \hline +128 \end{array} \qquad \begin{array}{r} -128 \\ + +127 \\ \hline -1 \end{array}$$

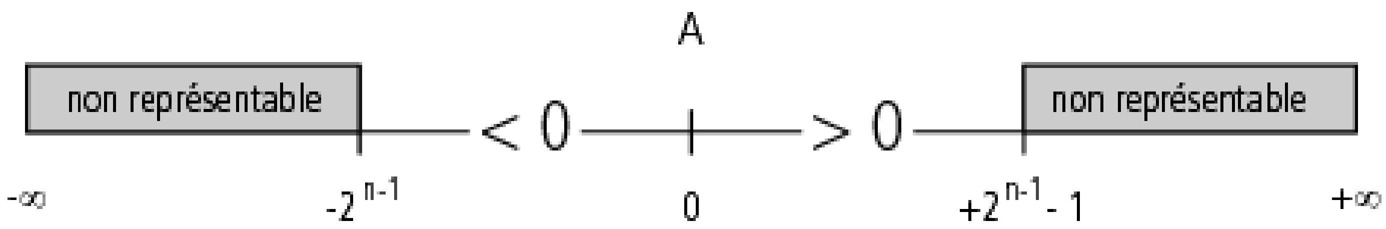
$\Rightarrow$ (sur-)dépassement de capacité (*overflow*) positif ou négatif

Importance de l'étendue des valeurs

- Le format et le code détermine l'ensemble des valeurs numériques codables
  - n = 8



  - généralisation



Formule et exemple

- $S = A - B = A + (-B) = A + \overline{B} + 1$
- Exemple (n = 8 bits) :
  - Soit à calculer S = 8 – 10 = 8 + (-10) = - 2
  - A = 0000 1000
  - B = +10 = 0000 1010

$\Rightarrow$ -B = 1111 0101 + 1 = 1111 0110

$\Rightarrow$ S = 0000 1000 + 1111 0110 = 1111 1110

Résultats d'une addition algébrique

- Règles sur les opérandes
  - signes opposés : aucun dépassement possible
  - signes identiques : risque de dépassement positif ou négatif
- Le CPU indique la validité du résultat à chaque addition
  - indicateur binaire ou drapeau OF (*Overflow Flag*) dans son registre d’état (dépassement de format pour opération sur nombres signés)

Multiplication et division en c2n

- Le plus simple : traiter les signes à part et réaliser les opérations en binaire naturel !

Signe du multiplicande A	Signe du multiplicateur B	Signe du résultat R
-	-	+
-	+	-
+	-	-
+	+	+

Signe du dividende N	Signe du diviseur D	Signe du résultat Q	Signe du reste R
-	-	+	-
-	+	-	-
+	-	-	+
+	+	+	+

Multiplication en c2n

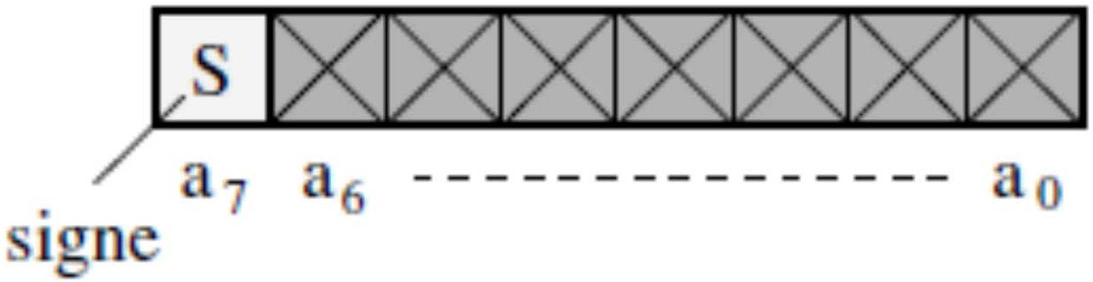
- Algorithme de [Baugh and Wooley 73]

$$P = \left(1 \times 2^{2n-1}\right) + \left(\overline{a}_{n-1} + \overline{b}_{n-1} + a_{n-1}\, b_{n-1}\right) 2^{2n-2} + \sum_{i=0}^{n-2} \sum_{j=0}^{n-2} \left(a_i \times b_j \times 2^{i+j}\right)$$

$$+ \sum_{i=0}^{n-2} \left(\overline{a}_i \times b_{n-1} \times 2^{n-1+i}\right) + \sum_{j=0}^{n-2} \left(a_{n-1} \times \overline{b}_j \times 2^{n-1+j}\right) + \left(a_{n-1} + b_{n-1}\right) 2^{n-1}$$

BCD signé

- Un octet est ajouté à gauche du module contenant le signe
- Même convention de codage du signe que pour les entiers relatifs sauf cas particuliers



BCD signé normalisé

- Standard IEEE 754
  - octet le plus à gauche
    - un bit pour le signe avec sept bits de bourrage (*stuffing bit*)
  - 18 quartets pour les chiffres du nombre

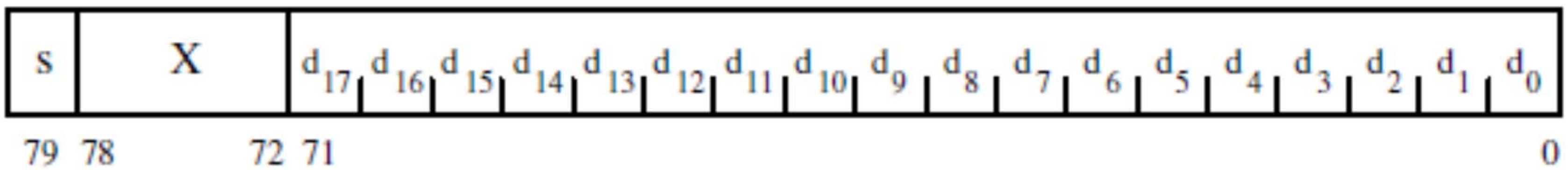
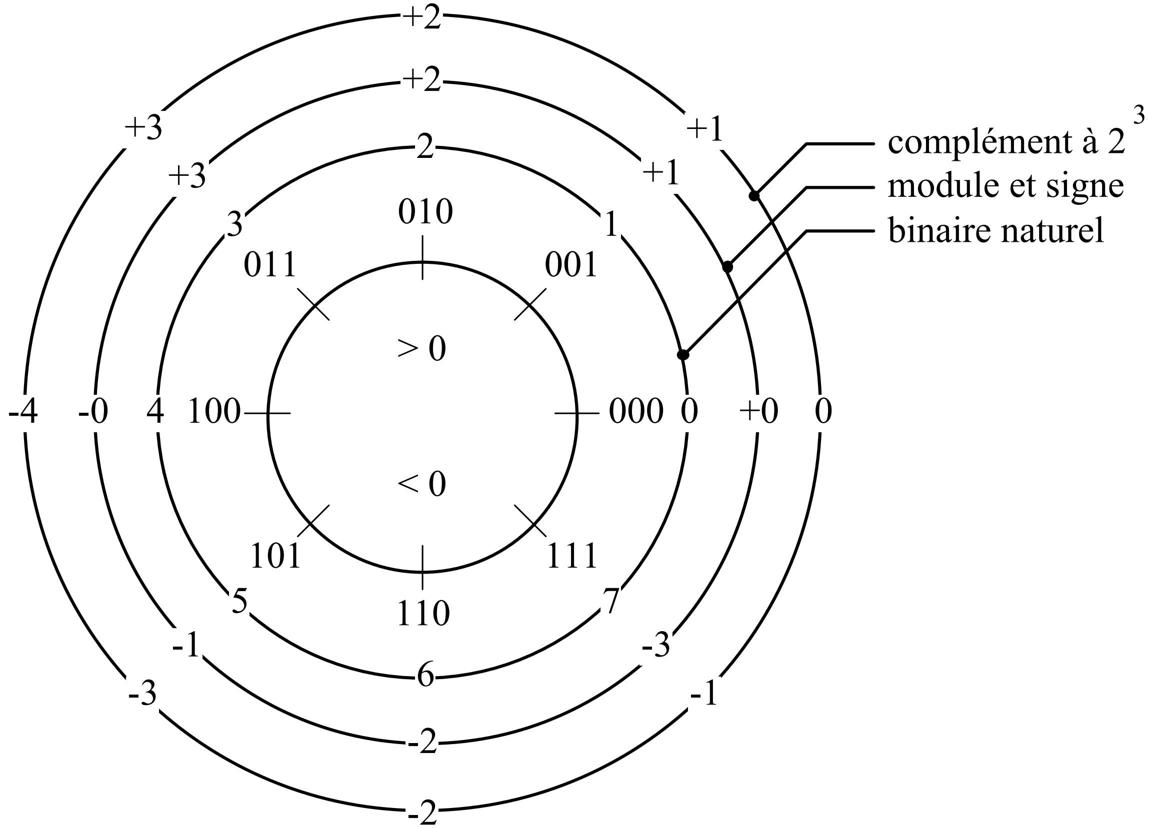


Tableau de correspondance entre représentations d'entiers (n = 4 pour B = 2)

A_2	$A_{10<bn}$	$A_{10<c2n}$	$A_{10<2ms}$
0000	0	0	+0
0001	1	+1	+1
0010	2	+2	+2
0011	3	+3	+3
0100	4	+4	+4
0101	5	+5	+5
0110	6	+6	+6
0111	7	+7	+7
1000	8	-8	-0 !
1001	9	-7	-1
1010	10	-6	-2
1011	11	-5	-3
1100	12	-4	-4
1101	13	-3	-5
1110	14	-2	-6
1111	15	-1	-7

Vue circulaire

- Format n = 3 bits



Conversions entre représentations en machine d'entiers

	Représentations			
Nombre	Binaire naturel	Signe et module	Complément à 1	Complément à 2
A > 0	= pour un format n-1	=	=	=
A = 0	=	= sauf pour -0 !	= sauf pour -0 !	=
A < 0	non codable	•	•	•

Autres représentations entières signées

- Complément à la base diminué
  - appelé aussi complément à 1
- Représentation à chiffre signé (*Signed-Digit Number Systems* (SDNS), *Signed-Digit Number Representations* (SDNR) ou *Redundant Number Systems*)
  - BSDNS (*binary* SDNS) → chiffres -1, 0 et +1
- Etc.

Extension de format

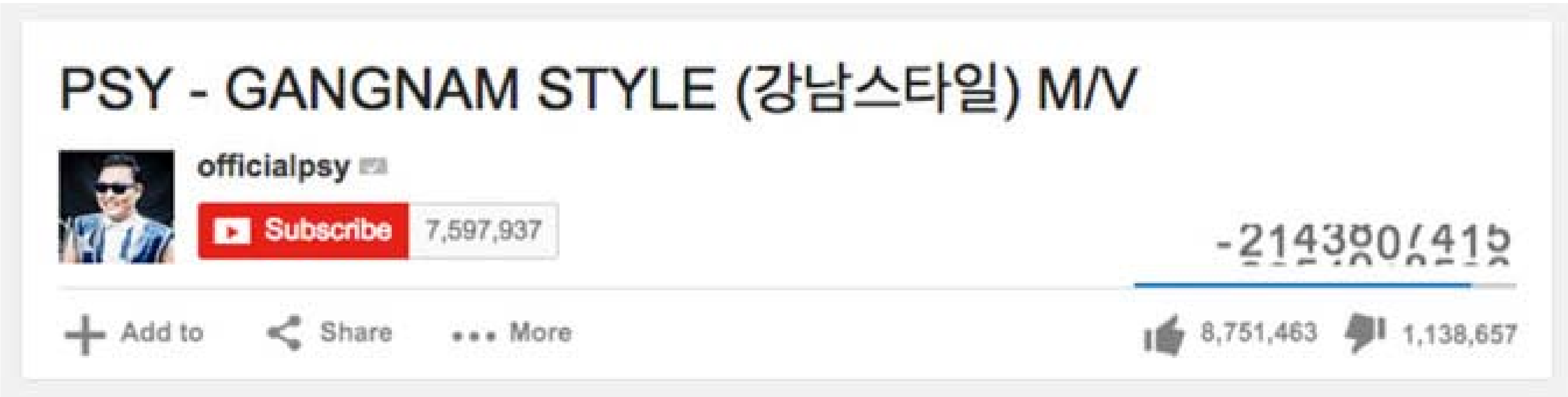
- Passage d’un format n à un format n’ (n’ > n)
  - CBN
    - ajout de 0 non significatif à gauche
  - 2sm
    - ajout de 0 entre le signe et le module
  - complément à 2^n
    - extension du bit de signe
      - = recopie du signe

Exemple de déclarations de type d’entier dans un langage évolué (C)

- Forme signée (implicite) ou non signée (*unsigned*)
- Type `char`
  - entier naturel codé en (code) binaire naturel
    (si préfixe *unsigned*)
    ou entier relatif codé en complément à 2^n
    (préfixe *signed* optionnel)
  - format n = 8
- Type `int`
  - entier relatif en complément à 2^n
  - format n = 16 ou 32 selon le MPU

Juste une question pour terminer

- Que s'est-il passé chez YouTube le 3 décembre 2014 ?



Conclusion sur la représentation de l'information en machine

0	1	0	1	0	0	0	0

$(50)_{16}$

b_7 b_6 - - - - - - - - - - - - b_0

etc.

"P"
+ bit d'imparité

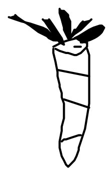
code ASCII

$(80)_{10}$

binaire naturel

(+80)

complément à 2
format n = 8



des carottes !!

⇒ une information est caractérisée par sa représentation et son format n