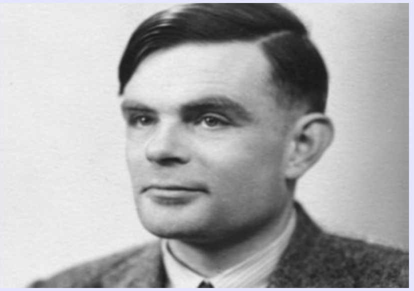
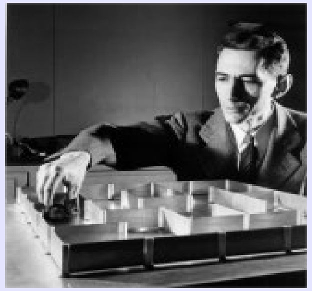
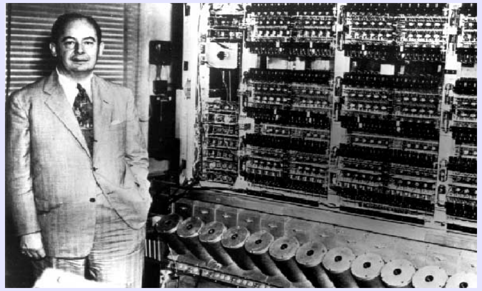






Information, Calcul et Communication

Théorie: Représentation de l'Information (1)

R. Boulic

Par quels moyens peut on représenter des nombres entiers positifs ou négatifs ?

Plan

Lien avec les leçons précédentes
- **Rappel des domaines d'applications**
- **Une représentation est une convention**
- **Vers l'unité élémentaire d'information (exercices)**

Manipulation sur les nombres entiers
- **Opérations et domaine couvert**

La virgule flottante: Pourquoi ? Comment ?
- **Un exemple qui pose problème**

Retour à la représentation des symboles
- **De l'alphabet aux idéogrammes**

Comment représenter un nombre entier?

Commençons par les entiers naturels (=positifs et nul)

Rappel: tout nombre peut être représenté à l'aide d'un ensemble d'éléments binaires.

Définition: une suite de 0 et de 1 est appelé un ***motif binaire***

un motif binaire isolé est insuffisant
pour comprendre ce qui est codé

- Il faut en plus une méthode d'interprétation du motif binaire en tant que ***donnée*** **(*data*)**
- Une solution: la notation positionnelle des nombres

Notation positionnelles des nombres

Exemple d'un nombre entier en base 10 : 703

Le nombre 703 est la notation abrégée de l'expression:

$$7 \,.\, 10^2 + 0 \,.\, 10^1 + 3 \,.\, 10^0$$

- Le chiffre de droite est toujours associé à la puissance 0 de la base 10
- La puissance de la base augmente d'une unité de chiffre en chiffre, en allant de la droite vers la gauche

Cette convention de **notation positionnelle** peut être exploitée dans n'importe quelle base

Représentation *positionnelle* en base 2

repose sur les mêmes conventions qu'en base dix (décimal)

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

poids forts à gauche, **poids faibles** à droite

Pratique: Conversions

Du binaire vers le décimal: additionner les puissances de 2 présentes dans le motif binaire

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	1	0	1	1

$$0 + 0 + 0 + 0 + 8 + 0 + 2 + 1 = 11$$

Du décimal vers le binaire :
décomposer un nombre entier X en une somme de puissances de 2 par division entières successives tant que le quotient ≥ 2

$$\begin{aligned} 11 &= 2 \cdot 5 + 1 \\ &= 2 \cdot (2 \cdot 2 + 1) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot 1 + 0) + 1) + 1 \\ &= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 1011 \end{aligned}$$

EPFL

Entiers: domaine couvert

Une représentation destinée à une machine est associée à une capacité fixe exprimée en nombre de bits (d'octets).

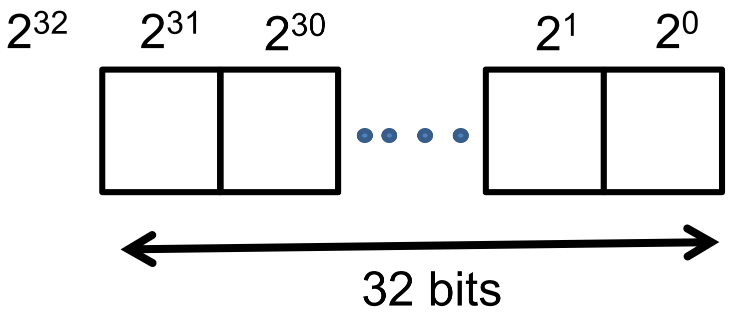
<u>Ex:</u> entier manipulé par une machine « 32 bits ». Cette machine dispose d'instructions pour réaliser très rapidement les opérations de base (addition, multiplication, etc) pour des entiers représentés sur 4 octets (max).

Donc limitation du nombre d'entiers représentables = 2^{32}

Si la représentation est totalement destinée aux nombres positifs, son **<u>domaine couvert</u>** est alors, pour 32 bits :

Min = motif binaire avec des 0 partout = zéro

Max = motif binaire avec des 1 partout = $2^{32}-1$



Entiers: domaine couvert (2)

Les calculs sur des entiers sont exacts si le résultat désiré est un entier et appartient au domaine couvert

La représentation choisie doit tenir compte de l'ensemble des résultats possibles

Plusieurs causes possibles de dépassement de capacité:

- division entière: perte de partie fractionnaire
- multiplication, addition, soustraction: propagation de retenue au delà de 2^{31}

exemples de dépassement de capacité

Exemple 1: addition de 2 entiers avec une capacité de un seul bit

0	0	1	1	
+0	+1	+0	+1	
--	--	--	--	
0	1	1	$\boxed{1}$0	la retenue est perdue

Exemple 2: addition avec une capacité de 32 bits

2^{32}	2^{31}																															2^0
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
~~1~~	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Entiers négatifs:
Clef de la représentation: tirer parti du dépassement de capacité

Question: comment tirer parti d'une capacité limitée de n bits pour en déduire une représentation des entiers négatifs permettant de remplacer la soustraction par l'addition de l'opposé ?

Rappel: n bits permettent de représenter 2^n nombres entiers positifs de 0 à $(2^n - 1)$ quand tous les bits sont à 1.

La valeur 2^n elle-même n'est pas représentable sur n bits, on a:

$(2^n - 1) + 1 = 2^n$ (en théorie)

Mais $(2^n - 1) + 1 = 0$ (sur n bits)

Conséquence: le motif binaire de $(2^n - 1)$ est une bonne représentation de -1 car on obtient 0 quand il est ajouté à 1

Représentation des entiers signés

Propriétés à vérifier: si a et b sont deux nombres opposés

Alors :

$$a + b = 0$$

de plus $\quad -(-a) = a$

Avec n bits de capacité, on pose que l'opposé d'un nombre **x** est donné par l'expression **2^n - x** appelée le Complément à 2 de **x**.

donc $\quad a + b = (2^n - b) + b = 2^n = 0 \quad$ (sur n bits)

$$-(-a) = 2^n - (2^n - a) = a$$

Pratique: comment calculer l'opposé (2^n – **x**) d'un entier **x** ?

Une transformation est nécessaire pour :
- faire apparaitre des quantité représentables sur n bits
- les manipuler à l'aide d'opérations simples

$$2^n - x$$

$$= 2^n \underline{-1 + 1} - x$$

$$= (\underline{(2^n - 1)} - x) + 1$$

1	1	1	1	1	1	1	1

Cas particulier : ((2^n -1) – **x**) est très facile à obtenir !

Il suffit d'inverser chaque bit de **x** : 0 -> 1 ou 1 -> 0

Cette valeur est appelée le Complément à 1 de **x**.

Complément à 2 de **x** = Complément à 1 de **x** + 1

Exemple1: Complément à 1 du nombre onze en binaire = (2^n-1) - onze

0	0	0	0	1	0	1	1	motif binaire de onze

	1	1	1	1	1	1	1	1	2^n-1
-	0	0	0	0	1	0	1	1	soustraction de onze

1	1	1	1	0	1	0	0	complément à 1 de onze

Exemple2: calcul de l'opposé de onze avec son complément à 2=(complément à 1)+1

	0	0	0	0	1	0	1	1	motif binaire de onze
	↓	↓	↓	↓	↓	↓	↓	↓	
	1	1	1	1	0	1	0	0	complément à 1 de onze
+	0	0	0	0	0	0	0	1	addition de + 1
=	1	1	1	1	0	1	0	1	complément à 2 de onze = opposé de onze

Exemple2: calcul de l'opposé de onze avec son complément à 2=(complément à 1)+1

=	1	1	1	1	0	1	0	1	complément à 2 de onze = opposé de onze
+	0	0	0	0	1	0	1	1	addition de onze
	1	1	1	1	1	1	1		
	0	0	0	0	0	0	0	0	0 sur n bits

Domaine couvert avec le complément à 2

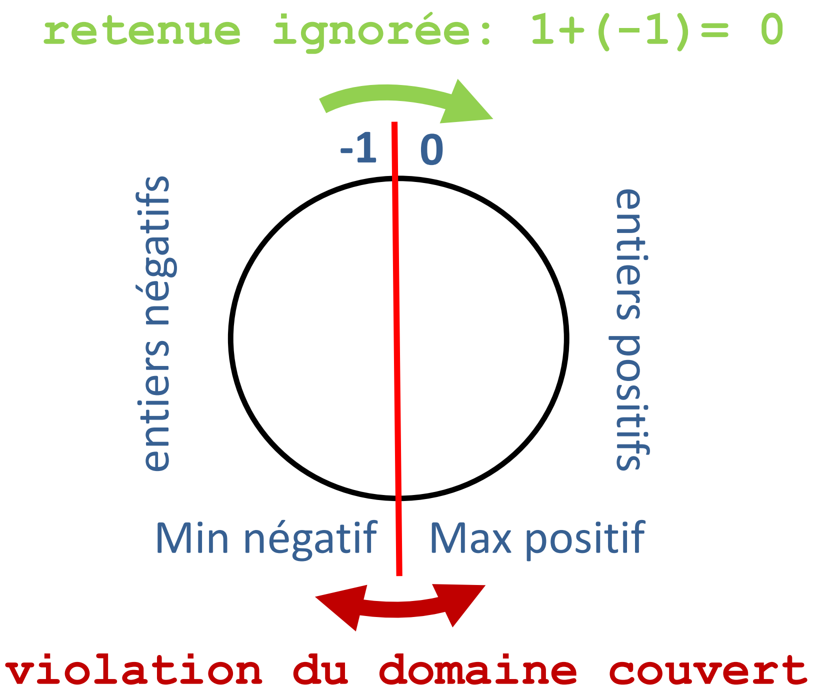
Bit de poids fort = signe

Min positif = 00000…0000 = 0

Max positif = 01111…1111

Min négatif = 10000…0000

Max négatif = 11111…1111 = -1

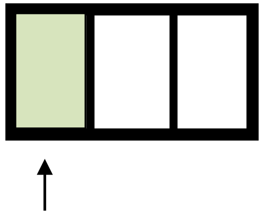


Dépassement de capacité ⇔ l'opération produit une retenue qui est ignorée ; cela ne pose aucun problème quand l'addition d'un entier positif et d'un entier négatif donne un résultat positif car *on reste dans le domaine couvert*.

Violation du domaine couvert: changement incorrect du bit de signe quand l'addition de 2 entier positifs > Max Positif (ou 2 entiers négatifs < Min négatif)

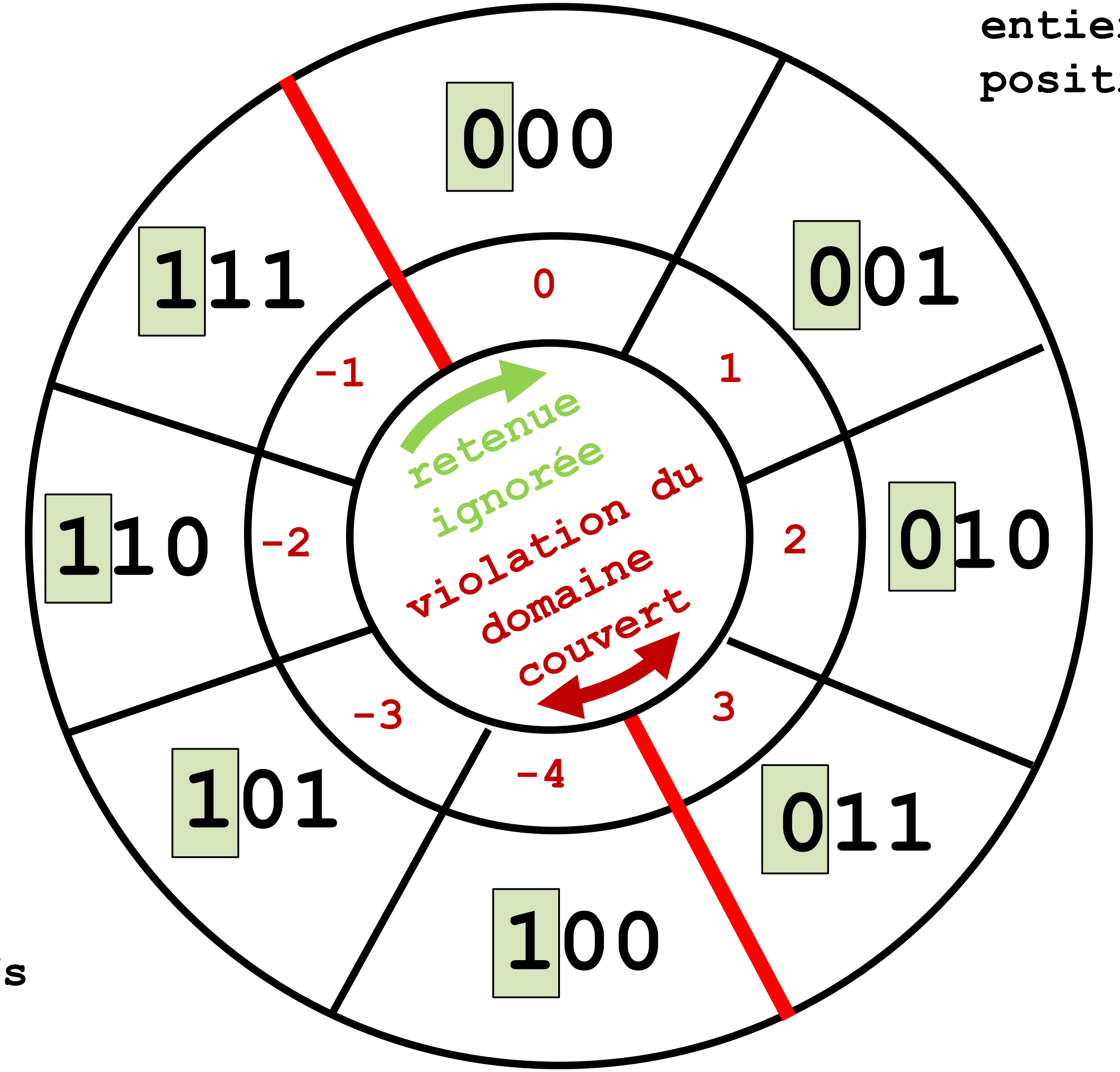
Entiers signés: domaine couvert

Exemple:
sur 3 bits



signe

entiers positifs



entiers négatifs

Questions:

Comment décoder un entier positif représenté en complément à 2 ?

Faire la somme des puissances de 2 dont le bit vaut 1

Comment décoder un entier négatif représenté en complément à 2 ?

Remarquer que son bit de signe vaut 1

Obtenir sa valeur absolue en calculant son opposé => son complément à 2

Décoder le motif binaire obtenu comme pour un entier positif

SpeakUp: avec cette représentation sur 3 bits, quel résultat décimal obtient-on pour **2+3** ? **A**: 5 , **B**: −5 , **C**: -1 , **D**: −3

Résumé

Est il possible de construire une représentation exacte du monde réel ?

Les calculs avec la représentation positionnelle **entière** donnent des **résultats exacts** pour autant qu'on reste dans le **domaine couvert**.

Conséquence:

- Cette propriété qui semble évidente et banale est en fait très importante
- Si un problème peut se résoudre en travaillant avec des entiers alors c'est l'approche à retenir. Exemples:

  - Pendant longtemps l'arithmétique financière a été effectuée en *décimal-codé-binaire* sur les entiers entre 0 et 9 de nombres en base dix.

  - Représentation d'un rationnel par une paire d'entier (numérateur, dénominateur) pour préserver la quantité exacte (e.g. 1/3).