RÉPUBLIQUE ALGÉRIÈNNE DÉMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED BOUDIAF - M'SILA
FACULTE DE TECHNOLOGIE

DEPARTEMENT DE SCOLE COMMUN

TP Informatique 01

Dr. Bilal BOUDJELLAL

Professeur à l'université Mohamed Boudiaf – M'sila

Version 1 – Octobre 2018

bilal.boudjellal@gmail.com

Avant-propos

Les lignes qui suivent et qui concernent principalement les étudiants de 1ère année socle commun ST de l'université Mohamed Boudiaf - M'sila. Le but de ce document est d'aider les étudiants à mieux comprendre la matière "***TP - Informatique 1***".

N'hésitez pas à me contacter en cas de besoin Bon travail !

Enseignant : Dr. Bilal BOUDJELLAL

Contact : par mail au Bilal.Boudjellal@gmail.com

Disponibilité :

- **Au bureau :** Lundi, Mardi de 10h00 -11h00
- **Par émail :** toute question en relation avec le cours doit être envoie par émail, je m'engage à répondre aux questions par mail dans un délai de 48 heures qui suivent la réception du message, sauf en cas des imprévus.

Informations sur le TP :

- **Faculté :** Technologie
- **Département :** Socle Commun ST
- **Public cible :** 1ère année Licence
- **Intitulé du cours :** Informatique 01
- **Crédit :** 04
- **Coefficient :** 02
- **Durée :** 14 semaines
- **Horaire :** 01h30
- **Salle :** SI2

I.5 Représentation des nombres entiers

I.5.1 Représentation d'un entier naturel

Un ***entier naturel*** est un nombre entier ***positif*** ou ***nul***. Le choix à faire (c'est-à-dire le nombre de ***bits*** à utiliser) dépend de la fourchette des nombres que l'on désire utiliser. D'une manière générale, Avec ***N*** bits, on code 2^N valeurs, alors, un codage sur ***N*** bits pourra permettre de représenter des nombres entiers naturels compris entre ***[0*** et $2^N - 1]$.

Par exemple, avec ***8*** *bits*, on peut coder ***256*** valeurs différentes, car :

$$2^8 = 256$$

Et représenter des nombres entiers naturels compris entre ***[0*** et ***255]***, car :

$$2^8 - 1 = 255$$

Un ordinateur manipule des nombres binaires par groupe de ***8 bits = un octet***. Données codées sur un ou plusieurs octets : *8 bits*, *16 bits*, *32 bits*, *64 bits* ... etc.

I.5.2 Représentation d'un entier relatif (entier signé)

Un ***entier relatif*** est un entier pouvant être ***négatif***. Il faut donc coder le nombre de telle façon que l'on puisse savoir s'il s'agit d'un nombre positif ou d'un nombre négatif, et il faut de plus que les règles d'addition soient conservées. L'astuce consiste à utiliser un codage que l'on appelle ***complément à deux***. Cette représentation permet d'effectuer les opérations arithmétiques usuelles naturellement.

Un ***entier relatif*** positif ou nul sera représenté en binaire (base 2) comme un entier naturel, à la seule différence que le ***bit de poids fort*** (le bit situé à l'extrême gauche) représente le signe. Le bit de poids fort sera :

- ***0*** pour les nombres *positifs* ou *nul* (0 correspond à un signe positif).
- ***1*** pour les nombres *négatifs* (1 correspond à un signe négatif).

D'une manière générale, Avec ***N*** bits, on code 2^{N-1} valeurs, alors, un codage sur ***N*** bits pourra permettre de représenter des nombres entiers naturels compris entre $[-2^{N-1}$ et $2^{N-1} - 1]$.

Par exemple, si on code un ***entier relatif*** avec ***8 bits*** (*1 octet*), on peut coder ***256*** valeurs différentes, car :

$$2^8 = 256$$

Et représenter des nombres entiers naturels compris entre ***[-128*** et ***127]***, car le plus petit entier relatif positif codé sur ***8 bits*** est :

$$-2^{8-1} = -128$$

Et le plus grand entier relatif positif codé sur ***8 bits*** est :

$$2^{8-1} - 1 = 127$$

Donc, de la même manière :

- Sur ***16 bits*** (2 *octets*), l'intervalle de codage est *[-32768, 32767]*.
- Sur ***32 bits*** (*4 octets*), l'intervalle de codage est *[-2147483648, 2147483647]*.

Il existe deux méthodes pour coder ***les entiers signés*** en ***binaire***:

- La méthode du signe et valeur absolue.
- La méthode du complément à deux.

I.5.3 Méthode du signe

Le principe de cette méthode consiste à utiliser un bit de signe (le bit de poids fort) et à coder la valeur absolue. On peut facilement représenter un entier signé en binaire en utilisant cette méthode selon les étapes suivantes :

- Représenter la **valeur absolue** du nombre en ***binaire*** (*base 2*).
- Le ***bit de poids fort*** (le plus à gauche) doit être égal à ***1***.

Par exemple, si on désire coder la valeur **-27** avec ***8 bits*** (*1 octet*). Il suffit :

1. Représenter la *valeur absolue* **27** du nombre en *binaire*. Le résultat est:

   $$(27)_{10} = (00011011)_{2}$$

2. Le ***bit de poids fort*** (le plus à gauche) doit être égal à ***1***. Donc :

   $$(-27)_{10} = (10011011)_{bs}$$

Note 1 : **bs** est l'abréviation de binaire signé.

L'inconvénient de cette méthode est qu'il existe toujours deux façons de coder le ***0****.*

I.5.4 Méthode du complément à deux

La méthode du complément à deux est le plus utilisé et le standard de fait pour coder les entiers signés car il existe une seule façon de coder le 0, grâce au « +1 » qui décale l'intervalle de codage des négatifs.

Le codage d'un entier relatif négatif avec complément à deux se fera selon les étapes suivantes:

- Représenter la **valeur absolue** du nombre en ***binaire*** (*base 2*).
- Inverser les bits : Les ***1*** deviennent ***0*** et les ***0*** deviennent ***1***. On fait ce qu'on appelle ***le complément à 1*** (*complément à un*).
- On ajoute ***1*** au résultat (les dépassements sont ignorés). Dit « ***complément à 2*** »

Par exemple, si on désire coder la valeur **-27** avec ***8 bits*** (*1 octet*). Il suffit :

1. Représenter la *valeur absolue* **27** du nombre en *binaire*. Le résultat est:

   $(27)_{10} = (00011011)_{2}$

2. D'écrire son *complément à 1* :

0	0	0	1	1	0	1	1
↓	↓	↓	↓	↓	↓	↓	↓
1	1	1	0	0	1	0	0

   $(27)_{10} = (11100100)_{cà1}$

3. Et d'ajouter *1* :

   $$\begin{array}{cccccccccc} & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ + & & & & & & & & 1 \\ \hline & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$$

   $(-27)_{10} = (11100101)_{cà2}$

Note : **cà1 et cà2 sont** les du ***complément à 1*** et le ***complément à 2***.

On remarquera qu'en additionnant un nombre et son complément à deux on obtient ***0***. En effet, ***00011011 + 11100101 = 00000000*** (avec une retenue de ***1*** qui est éliminée).

Dans tous les cas :

- Si bit de poids fort = 0 : entier positif.
- Si bit de poids fort = 1 : entier négatif.