Cette activité de groupe doit être comprise et réalisée en **autonomie**. Chacun de vous doit être capable de faire et d'expliquer les exercices d'applications

Encodage des nombres relatifs

1. Conversions entier relatif $\leftrightarrow$ binaire

Dans cette activité, nous allons nous intéressé à la représentation réelle en machine des nombres entiers relatifs.

ACTIVITÉ 1

Par exemple, le but est de comprendre *comment est représenté dans l'ordinateur le nombre -65*.

Spoil : sur 8 bits, $-65_{10} = 10111111_2$

Méthode d'*encodage des entiers relatifs*

1. la **longueur** en bits d'un binaire relatif est fixée à l'avance
2. le **signe** est représenté par le bit de poids fort (le premier bit à gauche) :

   – `0` pour un nombre positif ou nul
   – `1` pour un nombre négatif

3. un entier **positif** : codage classique des entiers naturels
4. un entier **négatif** : codage en **complément à deux**

Technique du *Complément à deux* pour déterminer l'opposé d'un entier relatif

1. **inverser** les bits
2. **additionner** 1 au résultat

CORRECTION

Comment encoder -65 sur 8 bits ?

(1) on se prépare à utiliser 8 bits donc `-65 = .... ....`

(2) -65 est négatif et est l'**opposé de 65**. Appliquons la technique du *complément à deux* sur 65 pour passer du codage binaire de 65 au codage binaire de -65 :

`codage_bin`(65) $\rightarrow$ complément à deux $\rightarrow$ `codage_bin`(-65)

Ce qui donne :

$$\underset{65_{10}}{0100\ 0001_2} \rightarrow \underset{\text{complément à deux}}{\begin{array}{|llll|} 1. & 0100\ 0001 & \rightarrow & 1011\ 1110 \\ 2. & 1011\ 1110\ +\ 1 & = & 1011\ 1111 \end{array}} \rightarrow \underset{-65_2}{1011\ 1111_2}$$

On a donc la réponse attendue (et on peut vérifier avec le bit de poids fort que c'est bien un nombre négatif) :

$$-65_{10} = 10111111_2$$

2. Exemples

Voici quelques exemples à étudier.

Exemple

Sur **2 octets** : $2021_{10} = 0000\ 0111\ 1110\ 0101_2$ car...

comme 2021 est positif, on utilise la conversion binaire classique :

$$2021 = 1024 + 512 + 256 + 128 + 64 + 32 + 4 + 1$$

ce qui donne

$$1024 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^2 + 2^0.$$

$2021_{10} = 11111000101_2$ et donc sur 16 bits :

$$2021_{10} = 0000\ 0111\ 1100\ 0101_2$$

Exemple

Sur **2 octets** : $-15_{10} = 1111111111110001_2$ car…

comme -15 est négatif, on va appliquer le *complément à deux* sur le codage binaire du nombre positif 15 :

- codage binaire de 15 sur 16 bits :

  1. $15_{10} = 1111_2 = 0000\ 0000\ 0000\ 1111_2$

- complément à deux :

  2. `0000 0000 0000 1111     → 1111 1111 1111 0000`
  3. `1111 1111 1111 0000 + 1 = 1111 1111 1111 0001`

Ainsi le codage relatif sur 2 octets devient : `-15 → 1111 1111 1111 0001`

Exemple

Sur 2 octets : $1111\ 1100\ 1010\ 1001_2 = -855$ car…

`1111 1100 1010 1001` est un nombre négatif puisque son bit de poids fort vaut `1`. Utilisons le **complément à deux** pour déterminer son **op-**

posé :

- complément à deux :
  1. `1111 1100 1010 1001` → `0000 0011 0101 0110`
  2. `0000 0011 0101 0110 + 1 = 0000 0011 0101 0111`
- codage décimal de l'oposé (qui est positif) :
  3. $0000\,0011\,0101\,0111_2 = 512_{10} + 256_{10} + 64_{10} + 16_{10} + 4_{10} + 2_{10} + 1_{10} = 855_{10}$

Donc $1111\,1100\,1010\,1001_2$ est l'opposé de 855_{10}. Ainsi :

$$1111\,1100\,1010\,1001_2 = -855_{10}.$$

3. Exercice d'application

ACTIVITÉ 2

Convertir en **binaire relatif sur un octet** les nombres 42 ; -42 ; 0.

Convertir en décimal les binaires relatifs codés sur 1 octet : 0011 1010 ; 1011 1010 ; 1111 1111.

ACTIVITÉ 3

Convertir en **binaire relatif sur deux octet** les nombres -1 ; -2022.

Convertir en décimal le binaire relatif codé sur 2 octet 0000 0111 1110 0110.