TP – Représentation des entiers en machine

Objectif

L'objectif de ce TP est de manipuler différentes représentations binaires des entiers et de comprendre le fonctionnement du complément à deux.

Toutes les listes de bits seront représentées par des listes d'entiers `0` ou `1`, en **big-endian** (bit de poids fort en premier).

Exemple :

$$13 = (1101)_2 \quad \longleftrightarrow \quad [1, 1, 0, 1]$$

—

1 Conversion entier $\leftrightarrow$ binaire

1.1 Fonction `tobin`

Écrire une fonction :

```
tobin(n: int) -> list[int]
```

qui renvoie la représentation binaire de l'entier naturel n sous forme d'une liste de bits en big-endian.

On supposera $n \geq 0$.

Exemples

$$\texttt{tobin(0)} = [0]$$

$$\texttt{tobin(13)} = [1, 1, 0, 1]$$

—

1.2 Fonction `bin2int`

Écrire une fonction :

```
bin2int(t: list[int]) -> int
```

qui calcule l'entier naturel correspondant à la liste de bits `t` (big-endian).

Exemples

$$\texttt{bin2int([1,0,1,1])} = 11$$

—

2 Complément à deux

Dans toute la suite, on travaille sur des entiers signés codés en **complément à deux (CA2)**.

2.1 Fonction `ca2`

Écrire une fonction :

```
ca2(a: int, n: int) -> list[int]
```

qui renvoie la représentation en complément à deux de l'entier a sur n bits.

On supposera que a est représentable sur n bits, c'est-à-dire :

$$-2^{n-1} \leq a < 2^{n-1}$$

Exemples

$$\texttt{ca2(5, 8)} = [0,0,0,0,0,1,0,1]$$

$$\texttt{ca2(-3, 4)} = [1,1,0,1]$$

—

2.2 Fonction `ca2int`

Écrire une fonction :

```
ca2int(t: list[int]) -> int
```

qui renvoie l'entier relatif représenté par la liste de bits `t` en complément à deux.

Exemples

$$\texttt{ca2int([0,1,1,0])} = 6$$

$$\texttt{ca2int([1,1,0,1])} = -3$$

—

3 Addition binaire

3.1 Addition non signée

Écrire une fonction :

```
add(t1: list[int], t2: list[int]) -> list[int]
```

qui réalise l'addition binaire de deux listes de bits de même longueur, sans tenir compte d'un éventuel dépassement (le bit de retenue final est ignoré).

Exemple

$$[1,0,1,1] + [0,1,1,0] = [0,0,0,1]$$

—

3.2 Addition en complément à deux

Écrire une fonction :

```
addca(t1: list[int], t2: list[int]) -> list[int]
```

qui réalise l'addition de deux entiers codés en complément à deux sur le même nombre de bits.

On supposera que l'addition ne provoque pas de débordement.

—

4 Format minimal

4.1 Fonction `min_format`

Écrire une fonction :

```
min_format(t: list[int]) -> list[int]
```

qui renvoie la représentation en complément à deux de même valeur que `t`, mais utilisant le **plus petit nombre de bits possible**.

On rappelle qu'en complément à deux, le bit de signe peut être répété sans changer la valeur.

Exemples

$$[0, 0, 0, 1, 0, 1] \longrightarrow [1, 0, 1]$$

$$[1, 1, 1, 0, 0] \longrightarrow [1, 0, 0]$$

—