Chapitre 2
Représentations des entiers positifs en machine

Objectif :

- Passer de la représentation d'un entier positif d'une base dans une autre ;

Vous avez vu que l'architecture des circuits composant un ordinateur impose l'utilisation du langage binaire. Ainsi, tout ce qui est manipulé par l'ordinateur doit être « traduit » pour être représenté dans la machine par une suite de 0 et de 1.

Vous allez découvrir dans ce chapitre comment sont représentés en machine les entiers positifs (= les entiers naturels).

Écriture d'un entier positif dans une base $b \geq 2$

Commencez par consulter le Notebook intitulé « S2_C2_Représentation_des_entiers_positifs.ipynb » qui vous présente comment on code un entier naturel dans les bases 2 et 16, les plus couramment utilisées en informatique.

Vous pouvez désormais attaquer l'activité qui suit qui a pour but d'écrire des algorithmes de conversions entre les bases décimale et binaire puis de les programmer en Python sous forme de fonctions qui pourront vous être utiles au cours de l'année scolaire.

Activité : Algorithmes de conversions

1) Conversion d'un entier en base 2

Il s'agit d'écrire ici l'algorithme en pseudo-code de la conversion en binaire par suite de divisions. On dit aussi qu'il s'agit de la méthode euclidienne (car on fait des divisions euclidiennes).

Vous avez déjà déroulé cet algorithme sur des exemples et vous êtes rendus compte que pour obtenir l'écriture binaire il fallait concaténer les restes des divisions euclidiennes.

1. Quel type de données de base permet de concaténer des éléments ?

L'algorithme consiste à répéter plusieurs fois les mêmes instructions et s'arrête lorsqu'on obtient un quotient nul.

2. Quel type de structure vous paraît adapter pour écrire cet algorithme ? Pourquoi ?

3. Ecriture de l'algorithme en pseudo-code.
Complétez l'algorithme qui suit de manière à ce qu'il corresponde à la spécification suivante :

Spécification :

- Entrées : un entier positif `n`

 Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS

- Sortie : une chaîne de caractères `D`
- Rôle : convertir `n` en base 2
- Précondition : aucune
- Postcondition : `D` contient l'écriture en base 2 de `n`

Algorithme :

```
D ← ......                  # valeur de D avant la première division
tant que n ...... faire
   r ← ......               # reste de la division euclidienne
   concaténer( ... , ... )
   ... ← n div 2            # quotient de la division
fin tant que
```

4. Programmez cet algorithme en Python et testez-le en comparant avec la fonction `bin`.

5. Transformez ce script Python en une fonction, appelée `int2bin` qui prend en paramètre l'entier `n` et qui renvoie la chaîne `D`. Vous pourrez ainsi, réutiliser cette fonction !

2) Conversion binaire → décimale

Le but est désormais d'écrire l'algorithme de la conversion inverse. Cet algorithme répond à la spécification suivante :

Spécification :

- Entrées : une chaîne de caractères `D`
- Sortie : un entier `n`
- Rôle : convertir `D` en base 10
- Précondition : `D` est formé uniquement des caractères `0` et `1`
- Postcondition : `n` est l'écriture en base 10 de `D`

L'algorithme consiste à parcourir chaque chiffre de `D`, à le multiplier par la bonne puissance de 2 et à ajouter tous ces résultats.

6. Quel type de structure permet de parcourir une chaîne de caractères très rapidement ?

7. Compléter l'algorithme suivant de manière à ce qu'il corresponde à la spécification précédente.
Attention : lorsque l'on parcoure la chaîne `D` avec une boucle `for`, on commence par le chiffre de gauche et donc par la plus grande puissance de 2.

```
n ← ......              # valeur de n avant les calculs
i ← .........           # plus grande puissance de 2 au départ
pour carac dans D faire
   n ← n + ......       # ajout du chiffre multiplié par sa puissance de 2
   ... ← ......         # la puissance diminue de 1
fin pour
```

8. Codez une fonction Python, appelée `bin2int` qui prend en paramètre une chaîne de caractères `D` (constituée de `0` et de `1`) et qui renvoie l'entier `n`. Vous pourrez ainsi, réutiliser cette fonction.

Facultatif pour les plus rapides / motivés :

9. Programmez des fonctions Python permettant de coder les conversions décimal → hexadécimal et hexadécimal → décimal.
La difficulté réside dans les conversions entre les caractères 'A', 'B', …, 'F' et leurs valeurs entières.

Ajouter et multiplier des entiers naturels en base 2

On peut poser les additions et multiplications comme en base 10 !

Exemple 1 : Calcul de $(101)_2 + (1101)_2$

On pose l'addition :

```
  1 1     ← retenues
   101
 + 1101
 -------
  10010   ← résultat
```

Vérification : $(101)_2 + (1101)_2 = 5 + 13 = 18 = (10010)_2$

Exemple 2 : Calcul de $(1110)_2 \times (101)_2$

On pose la multiplication :

```
    1110
  ×  101
 -------
   11       ← retenues pour l'addition
    1110
+   0000.
+  1110..
 -------
 1000110    ← résultat
```

Vérification : $(1110)_2 \times (101)_2 = 14 \times 5 = 70 = (100110)_2$.

Exercice 1 :

Calculer la somme $(1011)_2 + (11)_2$ et le produit $(110)_2 \times (100)_2$. Vous vérifierez vos résultats en repassant par la base décimale.

Nombre de bits nécessaires

Cas de la somme :
Ajouter deux entiers naturels ayant au plus k bits significatifs en nécessite au plus $k + 1$.

En effet : Si n et m sont deux entiers naturels ayant au plus k bits significatifs alors $n < 2^k$ et $m < 2^k$ et donc $n + m < 2^k + 2^k < 2 \times 2^k < 2^{k+1}$.

Cas du produit :
Multiplier deux entiers naturels n et m ayant respectivement au plus k et ℓ bits significatifs en nécessite au plus $k + \ell$.

En effet : $n < 2^k$ et $m < 2^\ell$ alors $n \times m < 2^k \times 2^\ell = 2^{k+\ell}$.

Exercice 2 :

Dans tout l'exercice, les nombres n et m désignent des entiers naturels ayant respectivement 4 et 5 bits significatifs.

1. Quel est le nombre maximal de bits significatifs nécessaires pour coder l'entier $n + m$? Et l'entier $n \times m$?
2. Quelles sont les valeurs binaires minimales et maximales pour n et m ?
3. Donnez deux nombres binaires n et m tels que l'écriture binaire de $n + m$ nécessite 6 bits. Même question pour 5 bits.
4. Supposons que $n = (1111)_2$. Peut-on trouver un nombre m telle que l'écriture binaire de $n + m$ nécessite 5 bits ?
5. Supposons que $m = (11000)_2$. Peut-on trouver un nombre n telle que l'écriture binaire de $n + m$ nécessite 5 bits ?
6. Supposons que $m = (10101)_2$. Quel est le nombre minimal de bits pour l'écriture binaire de $n \times m$?

Exercice 3 :

Déterminez dans chaque cas le nombre de bits nécessaires à l'écriture en base 2 de le somme $n + m$ puis du produit $n \times m$.

1. $n = 12$ et $m = 7$.
2. $n = (1011)_2$ et $m = (11100)_2$.

Ressources :

- *Synthèse de cours NSI 2019-2020*, Mickaël Barraud