1 TD : Nombres entiers non signés

1.1 Exercice

1) Donner la représentation en base 2 et sur 8 bits des entiers 14, 218, 42 et 57. Vous pouvez vérifier vos réponses avec la fonction `bin`.
2) Donner la représentation en base 16 de ces nombres. Vous pouvez utiliser l'écriture en base 2 de ces nombres. Vous pouvez vérifier en utilisant la fonction `hex`.

1.2 Exercice

Écrire en base 10 les nombres suivants écrits sur 8 bits en base 2 : $0010\ 1001_2$, $1010\ 0001_2$, $1100\ 0101_2$.

Vous pouvez vérifier dans l'interpréteur Python en tapant `0b` suivit du nombre écrit en binaire.

1.3 Exercice

Combien de bits au minimum pour écrire en base 2 les nombres suivants écrit en base 10 ?

63, 65, 129, 300, 550.

1.4 Exercice

1) Écrire en base 10 les nombres suivants écrits en base 2 :
   1_2, 10_2, 100_2, 1000_2.
2) Que remarque-t-on ?
3) Sachant que $41 = 101001_2$ Que vaut 1010010_2 en base 10 ?

1.5 Exercice

Challenge : compter sur les doigts en base deux.

Combien de valeurs différentes peut-on coder en utilisant les doigts des deux mains ?

1.6 Exercice

1) Écrire en bases 16 (hexadécimale) les nombres suivants écrits en base 2 (binaire) : $1100_2, 1001\ 0010_2, 0100\ 1101_2$ et $1010\ 1111_2$.
2) Écrire en bases 2 les nombres suivants écrits en base 16 : $B2_{16}, A3_{16}, 1C_{16}, 4D_{16}$ et $6B_{16}$.

1.7 Exercice

1) Écrire en base 16 les nombres suivants écrits en base 10 : 12, 17, 30, 31 et 32.
2) Écrire en base 10 les nombres suivants écrits en base 16 : $A2_{16}, F3_{16}, 1A_{16}, 2F_{16}$ et $5B_{16}$.

2 Nombre de bits nécessaires pour représenter une donnée

2.1 Exercice

1) Les paquets IP sont munis d'un champs appelé TTL (Time To Live, « temps restant à vivre »). Ce champs est initialisé par l'émetteur à 255 et à chaque routeur traversé décrémente ce champ de 1 jusqu'à la valeur 0, ce qui engendre la suppression du paquet IP.
   a. Combien de bits sont nécessaires pour stocker le champ TTL ?
   b. Combien de routeurs un paquet peut-il traverser au maximum ?
2) Le nombre de lignes d'un fichier de l'ancienne version d'un tableur était limité à 65 535. Combien de bits étaient nécessaires pour stocker le numéro de ligne ? Ce qui correspond à combien d'octets ?

2.2 Exercice

Le 2 avril 2020, US Federal Aviation Administration a ordonné aux utilisateurs de Boeing 787 de redémarrer leurs appareils tous les ... jours pour éviter « différents scénarios potentiellement catastrophiques ».

La raison : il semblerait que la période des compteurs de temps des processeurs embarqués est de 1,024 ms et non pas de 1 ms exactement. De plus, en supposant que le compteur de millisecondes depuis le moment du démarrage de l'avion soit codé sur 32 bits.

1) Déterminer le nombre maximal d'unités de temps que l'on peut coder avec en entier non signé sur 32 bits.
2) À l'aide des informations données, convertir ce nombre en jours.
3) Au bout de combien de jours faut-il redémarrer les ordinateurs de bord de ces avions ?

2.3 Exercice

Dans les années 60 la mémoire était très cher et des langages comme le COBOL ne permettait pas d'avoir de champs contenant plus de 80 caractères (héritage de carte perforées). Pour économiser de la place, le programmeur ont décidé de coder les années sur deux caractères ce qui a causé le « bug de l'an 2000 ». En effet en 2000 aurait été stocké 00 rendant impossible les comparaisons de dates. Par exemple La différence entre 1992 et l'an 2000 aurait été $00 - 92 = -92$ au lieu de $2000 - 1992 = 8$.

Le programmeur des années 1970 avaient consciences de ce problème et ont décidé de représenter le temps par un nombre de secondes écoulé depuis le 1erjanvier 1970 à 0h. C'est la norme POSIX. Ce nombre était un entier signé sur 32 bits ce qui fait que le plus grand nombre possible est $01111111111111111111111111111111_2$ soit le plus grand nombre possible que l'on peut stocker sur 31 bits.

1) Quel est, en base 10, le plus grand nombre que l'on peut écrire sur 31 bits.
2) Convertir ce nombre de secondes en nombre d'années.
3) En quelle année aura lieu ce bug ?

Si le fonctionnement des ordinateurs n'est pas modifié, une seconde après cette date butoir, ils afficheraient la date du 13 décembre 1901. Ce bug est en cours de correction sur les machines anciennes. Les machines récentes représentent le temps en seconde sur 64 bits ce qui correspond à plusieurs fois l'âge de l'Univers.

3 Addition binaire

Comme nous avons que nous pouvions écrire tous les nombres entiers non signés en binaire, il est légitime de vouloir faire des opérations sans repasser par la base 10.

En binaire, la table d'addition d'un bit est la suivante :

- $0_2 + 0_2 = 0_2$
- $0_2 + 1_2 = 1_2$
- $1_2 + 0_2 = 1_2$
- $1_2 + 1_2 = 10_2$. Il y a donc une retenu uniquement dans ce cas.

Pour additionner deux nombres à plusieurs chiffres, il faut commencer par rajouter des 0 à gauche du plus court des deux et ensuite, on fait l'addition chiffre par chiffre, en partant de la droite et en faisant attention aux retenues. Voici quelques exemples :

$$\begin{array}{r} {\scriptstyle 1} \\ 0\,1 \\ +\,0\,1 \\ \hline 1\,0 \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ 1} \\ 0\,0\,1\,1 \\ +\,1\,0\,0\,1 \\ \hline 1\,1\,0\,0 \end{array} \qquad \begin{array}{r} \phantom{{\scriptstyle 1\ 1\ 1\ 1}} \\ 0\,1\,0\,1 \\ +\,1\,0\,1\,0 \\ \hline 1\,1\,1\,1 \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ 1} \\ 0\,1\,0\,1 \\ +\,0\,1\,0\,1 \\ \hline 1\,0\,1\,0 \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1} \\ 0\,1\,1\,1\,1 \\ +\,0\,0\,0\,0\,1 \\ \hline 1\,0\,0\,0\,0 \end{array}$$

3.1 Exercice

Calculer en binaire les sommes suivantes puis vérifier en convertissant en base 10.

1) $110 + 1$
2) $101 + 11$
3) $1100 + 0011$
4) $1011 + 1000$
5) $1000 + 1000$
6) $1111 + 1111$

4 Multiplication binaire

Nous allons poser la multiplication comme à l'école. Ce n'est pas la façon de faire d'un ordinateur qui a des méthodes plus performantes. La multiplication est relativement simple à poser puisqu'à chaque bit du nombre du bas, si c'est un 1, on recopie le nombre de haut, sinon on met des 0. Il n'y a donc aucune multiplication à faire, juste des additions. On peut remarquer que le nombre de bits nécessaires pour multiplier deux nombres de k bits est $2k$ bits. Il faut donc veiller a ne pas dépasser le nombre de bits possibles. Si pour écrire un nombre il n'y a pas assez de bits disponibles, il y a dépassement, overflow en anglais, et le résultat n'est plus cohérent. Voici un exemple :

$$\begin{array}{r} 1\,1\,0\,1 \\ \times\,1\,0\,1\,1 \\ \hline 1\,1\,0\,1 \\ 1\,1\,0\,1 \\ 0\,0\,0\,0 \\ 1\,1\,0\,1 \\ \hline 1\,0\,0\,0\,1\,1\,1\,1 \end{array}$$

4.1 Exercice

Calculer en binaire puis vérifier en convertissant en base 10.

1) 0011×0011
2) 1011×1001
3) 1111×1111