1 Problématique

Dans un ordinateur l'UAL du microprocesseur réalise les opérations arithmétiques. Pourtant nous savons qu'il n'est composé que de transistors qui laissent ou non passer le courant. Ce comportement *binaire* est la base de l'information dans la machine.

Comment alors représenter les nombres entiers dans la mémoire de l'ordinateur ?

2 Cellules mémoires

Le passage du courant ou non est représenté par les chiffres 0 ou 1. Nous parlons de **BInary DigiTS** ou plus simplement la contraction **bits**.

FIGURE 1 – 1 bit

Dans la mémoire d'un ordinateur, ces chiffres sont regroupés par paquet de huit qu'on appelle **octet** (ou **bytes** en anglais). Les octets sont organisés en *mots machines* de 2, 4 ou 8 octets. Une

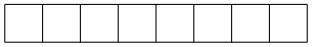

FIGURE 2 – 1 octet

machine *32 bits* manipule des mots de 4 octets ($4 \times 8 = 32$ bits) lorsqu'elle effectue des opérations.

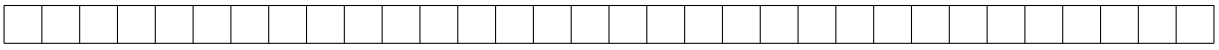

FIGURE 3 – 1 mot-machine 32 bits

Activité 1 : Chaque bit accepte 2 valeurs possibles : 0 ou 1. Avec 1 bit nous pouvons donc avoir 2 combinaisons possibles.

1. Combien de combinaisons peut-on réaliser avec 1 octet ?
2. Même question pour 1 mot-mémoire 32 bits ?

3 Encodage des entiers naturels

Afin de représenter un entier naturel en mémoire, il suffit de le convertir en base 2.

3.1 Écriture en base 10

Le principe de l'encodage en base 2 s'appuie sur le même modèle que celui en base 10. Rappelons ce principe :

$$6103 = 6 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 3 \times 10^0$$

Activité 2 :

1. Décomposer 76035 en base 10.
2. Combien d'entiers en base 10 peut-on représenter avec 4 chiffres ? Indiquer le plus petit et le plus grand.
3. Combien d'entiers en base 10 peut-on représenter avec k chiffres ? Indiquer le plus petit et le plus grand.

Christophe Viroulaud

3.2 Écriture en base 2

Afin d'éviter les ambiguïtés il est possible d'écrire un nombre en précisant sa base : 1001_2.

Activité 3 :

1. En s'appuyant sur le principe du paragraphe précédent, calculer la valeur de l'entier du nombre binaire suivant : 101101_2.
2. Écrire en base 2 les entiers de 0 à 10.
3. Combien d'entiers peut-on représenter avec 8 chiffres binaires (*soit 1 octet)* ? Indiquer le plus petit et le plus grand.
4. Combien d'entiers peut-on représenter avec k chiffres binaires ? Indiquer le plus petit et le plus grand.
5. Quel est le plus grand entier que l'on peut stocker dans un mot-mémoire 32 bits ?

3.3 Unités de mesures

Comme pour les unités de masse, de longueur, il est pratique de convertir la capacité mémoire d'un ordinateur.

1 kilooctet = 1000 octets

Activité 4 :

1. En s'aidant de la page Wikipedia : https://fr.wikipedia.org/wiki/Octet#Multiples_normalis.C3.A9s, citer les principaux multiples utilisés dans la vie quotidienne.
2. Associer chaque mémoire à un multiple de l'octet : disque dur, barette RAM, registre du processeur, clé USB, DVD.
3. Bob achète un disque dur de 500Go. Il le branche sur son ordinateur et vérifie sa capacité. Le système d'exploitation *Windows* lui annonce une capacité de 465Gio. Comment expliquer cet affichage ?

3.4 Conversion

Chaque entier est converti en base 2 avant d'être stocké en mémoire.

Activité 5 :

1. Regarder les pages :
   - http://www.elektronique.fr/cours/code/convertir_binaire-decimal.php#iii (lien court https://vu.fr/MpTd)
   - https://www.apprendre-en-ligne.net/crypto/images/bases.html (lien court https://vu.fr/2U2E)
   - Convertir 37_{10} en base 2.

3.5 Addition binaire

Une addition en base 2 applique les mêmes principes qu'en base 10 :

- $0 + 0 = 0$
- $1 + 0 = 1$
- $1 + 1 = 0$ et une retenue de 1
- $1 + 1 + 1 = 1$ et une retenue de 1

Christophe Viroulaud

Activité 6 :

1. Poser l'addition binaire : $101_2 + 111_2$
2. Poser l'addition binaire : $101101_2 + 11100_2$

3.6 Écriture en base 16

La base 16 est régulièrement utilisé pour représenter les nombres binaires plus facilement. Chaque chiffre hexadécimal est représenté par 4 bits.

Activité 7 :

1. Compléter le tableau d'écriture des chiffres hexadécimaux.

hexadécimal	bits
0	0000
1	0001
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	
D	
E	
F	

La représentation RGB (Red-Green-Blue) des couleurs du web utilise la notation hexadécimal. Par exemple, *FF0000* représente le rouge. Sa représentation en mémoire est donc :

$$\overbrace{1111}^{F}\ \overbrace{1111}^{F}\ \overbrace{0000}^{0}\ \overbrace{0000}^{0}\ \overbrace{0000}^{0}\ \overbrace{0000}^{0}$$

2. Déterminer la couleur représentée par le nombre binaire ci-après. Retrouver la nuance avec une recherche internet.

101110010010111110101000

4 Langage Python et les entiers

4.1 Découverte d'un EDI Python

Un EDI (Environnement de Développement Intégré) est un logiciel qui fournit des outils facilitant l'écriture d'un programme informatique. Il existe de nombreux EDI spécialisé pour Python tels Spyder, Pyzo, EduPython. Cette présentation s'appuiera sur *Spyder* mais il est possible d'effectuer les mêmes observations dans un autre environnement.

Activité 8 :

1. Regarder la vidéo de présentation : vidéo EDI
2. Parmi les écrans présentés, quel est le seul indispensable ?
3. Dans la console, écrire l'instruction **3+5** puis valider.
4. Écrire

```
1 print("L'addition 345+237="+str(345+237))
```

L'interpréteur lit et exécute le code ligne après ligne.

4.2 Encoder des entiers avec Python

Par défaut les nombres entiers sont encodés en base 10 en Python. Pour utiliser des nombres binaires il suffit d'ajouter le préfixe **0b**.
Également le préfixe **0x** permet de manipuler des nombres en base hexadécimale.
À l'inverse la fonction **bin()** convertit en base 2 n'importe quelle valeur.

Activité 9 : Dans la console, écrire :

1.

```
1 0b01001100
```

2.

```
1 0xAD2
```

3.

```
1 bin(76)
```

4. Convertir le nombre binaire **10101** en décimal.
5. Convertir le nombre hexadécimal **F3A** en base 2.

5 Retour sur la problématique

Un nombre entier est encodé en base 2 en mémoire. Les ordinateurs de moins de dix ans sont dits *64 bits* ce qui signifie que chaque mot-mémoire possède 64 bits.

Activité 10 :

1. Que pourrait-on déduire à propos de la taille maximale d'un entier en mémoire ?
2. Dans la console Python entrer le code suivant :

```
1 import sys
2 sys.maxsize
```

3. Avec une calculatrice (Numworks) calculer 2^{63}.
4. Comment expliquer la différence entre ce résultat et celui de la première question ?

Christophe Viroulaud