Exercice 1

Représentation binaire des entiers

1.1 Le Binaire

1. Rappeler les valeurs des puissances de 2, de 2^0 à 2^{10} :

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9

2. Combien vaut 64 en binaire. Puis 32 en binaire ? Comment fait-on pour diviser un nombre binaire, pair, par 2 ?

3. Le nombre de lignes d'un fichier de l'ancienne version d'un tableur très connu était limité à 65535. Combien de bits étaient nécessaires pour stocker le numero de ligne ? Et combien d'octets ?

4. Convertir en binaire les nombres 65, 129, 257 :

65	129	257

4. Convertir en binaire les nombres 63, 127, 255. Quelle remarque faite-vous ?

63	127	255

5. En remarquant que 96 est egal à 64 + 32, Convertir 96 en binaire.

6. Poser l'addition binaire 1010 1111 + 1111 0101 (resultat sur 9 bits)

```
  1010 1111
+ 1111 0110
----------------
...
```

7. Poser la soustraction binaire de 1111 1010 - 1110 0110

```
  1111 1010
- 1110 0110
----------------
...
```

1.2 Numération hexadecimale

1. Convertir en hexadecimal les nombres 65, 129, 257, 63, 127, 255

65	129	257	63	127	255

2. Deux entiers positifs ont pour écriture en base hexadecimale A8 et 94.

- Quelle est la valeur en base 10 pour chacun de ces nombres ?
- Calculer leur somme.
- Quelle est l'écriture en base 16 de leur somme ?

1.3 Représentations

1. Comment reconnaitre qu'un nombre binaire est divisible par 2 ? Par 4 ?
2. Si la représentation d'un nombre positif n en base 2 est sur 8 bits, combien de bits faudra-t-il pour $2 \times n$? Par exemple, n = 129 (8 bits) => 258 (... bits)

1.4 Grands nombres

1. Convertir en octets (ou en multiple d'octets), 1024 Mb (mega bits)
2. Convertir en bits (ou multiples), 5.6Go

1.5 Division euclidienne ou méthode rapide

La conversion utilise le principe de la division euclidienne :

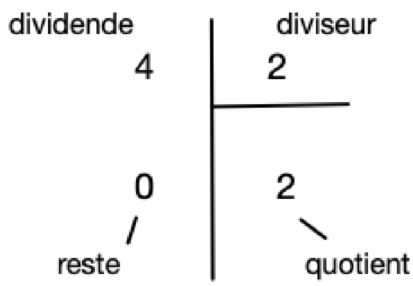

Figure 1 – division euclidienne de 4 par 2

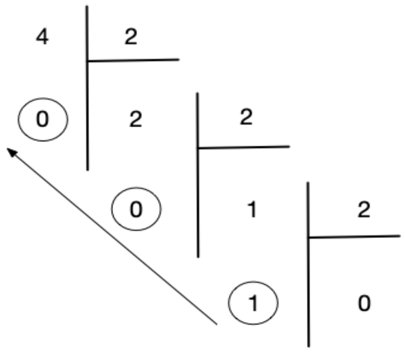

FIGURE 2 – conversion 4(10) = 100(2)

1. Utiliser la division euclidienne pour convertir 31 en binaire
2. Trouver une méthode rapide pour retrouver le même résultat (à partir d'une puissance de 2 proche de 31)

Exercice 2

Algorithmes de conversion

2.1 Base 2 vers la base 10

L'agorithme peut s'énoncer de la manière suivante :

On considère un nombre binaire constitué de 3 chiffres, a, b, c. Chacun de ces 3 chiffres pouvant être 0 ou 1.

- multiplier `a` par 4, `b` par 2 et `c` par 1. Ajouter le tout.
- Afficher le résultat

Programmer cet algorithme en Python.

```
...
...
```

2.2 Base 10 vers la base 2

On considère un nombre décimal de longueur quelconque, N.

- Calculer le quotient et le reste de la division de N par 2.
- Afficher le reste
- Placer le quotient dans N
- Continuer tant que le quotient est >0

Le nombre binaire issu de la conversion est celui affiché par cet algorithme, mais lu à l'envers
Programmer cet algorithme en Python.

```
...
...
...
...
```

Exercice 3

Entiers signés

3.1 Complément à 2

1. Quel est le nombre le plus grand et le nombre le plus petit que l'on peut représenter sur 8 bits en complement à 2 ?
2. Même question sur 16 bits

3.2 Entier relatif codé en complément à 2

1. calculer le nombre binaire associé à -12
2. calculer le nombre binaire associé à 12
3. Vérifier que l'addition binaire de ces 2 nombres donne 0
4. calculer le nombre binaire associé à -33
5. Quel est l'entier relatif codé en complément à 2 sur un octet par le code binaire 1111 1111 ?

$$\begin{array}{llll} (1) & -127 & (2) & 127 \\ (3) & -1 & (4) & 1 \end{array}$$

3.3 Représentation sur 16 bits

2. Vérifier que la représentation binaire sur 16 bits du nombre 2023 est 0000011111100111 ?
3. En déduire la représentation du complement à 2 du nombre -2023.

3.4 Association

Associer chacun des codes binaires suivants en complément à deux sur 8 bits au nombre qu'il représente :

$$\begin{array}{llllll} (1) & 00001111 & * & * & (a) & -127 \\ (2) & 10000001 & * & * & (b) & -86 \\ (3) & 11110000 & * & * & (c) & -16 \\ (4) & 01010101 & * & * & (d) & -15 \\ (5) & 11110001 & * & * & (e) & 15 \\ (6) & 10101010 & * & * & (f) & 85 \end{array}$$

3.5 Débordement

Quelles additions des nombres suivants provoquent un dépassement de capacité lorsque l'on utilise un codage sur 8 bits ?

1. 111 - 240
2. 113 + 15
3. 112 - 240
4. 112 + 15
5. -128 + 128
6. 256 - 200
7. -113 - 15

Exercice 4

Nombres fractionnaires

4.1 Conversions

1. Convertir $3{,}375_{10}$ en binaire (cahier NSI p 23)
2. Association

Associer le code binaire de la partie décimale d'un nombre fractionnaire représenté par un codage à virgule fixe utilisant 8 bits de partie fractionnaire avec le nombrre correspondant.

$$\begin{array}{cccccc}
(1) & 1000\,0000 & * & * & (a) & 0,9375 \\
(2) & 1111\,0000 & * & * & (b) & 0,875 \\
(3) & 0101\,0000 & * & * & (c) & 0,5 \\
(4) & 1110\,0000 & * & * & (d) & 0,375 \\
(5) & 0110\,0000 & * & * & (e) & 0,3125
\end{array}$$

4.2 Norme IEEE 754 simplifiée

voir l'exercice en ligne sur la page *codage des nombres* du site allophysique

Exercice 5

Algorithmes

5.1 Ex 4.1 : Utiliser l'algo de multiplication et suivre l'evolution des variables pour verifier que celui-ci réalise bien la multiplication de 7 par 4

Rappel de l'algorithme de multiplication

```
r <- 0
faire b fois:
    r <- r + a
# le resultat est r
```

avancée	a	b	r	itération n°
debut	7	4	0	avant iteration
dans la boucle	7	4	7	fin de la 1ere

5.2 Ex 4.2 : Utiliser l'algorithme de division pour a = 39 et b = 8

Rappel de l'algorithme de division :

```
r <- a
i <- 0
tant que r >= b, faire:
    r <- r - b
    i <- i + 1
# le resultat est i
```

avancée	a	b	r	i
debut	39	8	39	0
dans la boucle (1ere it)	39	8	31	1

…

Exercice 6

Corrections

Ex 4.1 :

avancée	a	b	r	itération n°
debut	7	4	0	avant iteration
dans la boucle	7	4	7	fin de la 1ere
	7	4	14	2
	7	4	21	3
	7	4	**28**	4

le résultat est 28.

Ex 4.2 :

avancée	a	b	r	i
debut	39	8	39	0

avancée	a	b	r	i
dans la boucle (1ere it)	39	8	31	1
dans la boucle (2)	39	8	23	2
dans la boucle (3)	39	8	15	3
dans la boucle (4)	39	8	7	4