Travaux dirigés :

Le codage IEEE 754

pour les nombres à virgule flottante

Année universitaire 2021–2022

1 Le format simple précision

1. Rappeler les différents champs qui composent le format IEEE754 simple précision ainsi que leurs tailles et leurs positions dans un mot de 32 bits.
2. Rappeler la formule permettant de calculer le nombre décimal codé par un nombre en écriture normalisée.
3. Donner la valeur du plus grand et du plus pe-

tit nombre décimal (en valeur absolue) qu'il est possible de coder en écriture normalisée.

4. Donner l'écriture normalisée de 7.78125.
5. Donner l'écriture normalisée de 0.1.

2 Précision relative en écriture normalisée

1. Donner l'écriture normalisée de 1, 2 et 4.
2. Combien de nombres différents de l'intervalle $[1, 2[$ est-il possible de coder et quelle est la distance entre ces nombres (donner la réponse en base 2 et l'ordre de grandeur en base 10)?
3. Même questions mais en considérant l'intervalle $[2, 4[$.
4. Même questions mais en considérant l'intervalle $[2^{127}, 2^{128}[$.
5. Que peut-on déduire de cet exercice concernant la précision des calculs ?

3 Nombres en écriture dénormalisée

1. Rappeler la formule permettant de calculer le nombre décimal codé par un nombre en écriture dénormalisée. Quelles sont les deux différences que l'on peut identifier par rapport à l'écriture normalisée ?
2. Donner la valeur du plus grand et du plus petit (en valeur absolue) nombre décimal différent de zéro qu'il est possible de coder en écriture dénormalisée.
3. Expliquer à quoi sert l'écriture dénormalisée et justifier les différences identifiées avec l'écriture normalisée.

4 Autres valeurs

En plus des nombres en écriture normalisée et dénormalisée, trois valeurs exceptionnelles peuvent être codées. Pour chacune, rappeler son nom, son

codage, et expliquer son intérêt.

5 Arithmétique

1. Soit le programme dont le code est donné listing 1 ci-dessous. Lorsqu'il est exécuté, la sortie produite est :

```
bsb =  0
bbs =  1e-06
```

   Expliquer.

2. Soit le programme dont le code est donné listing 2, qui calcule la racine carré de `y` par la méthode de Newton. Lorsque `y` vaut 3, l'exécution se termine et produit la sortie :

```
2
5.15
1.75
2.8662622
```

```
1.7321428
1.9564607
1.7320508
1.744921
sqrt( 3 ) =  1.7320508
```

En revanche, lorsque `y` vaut 2, le programme entre dans une boucle infinie. Les premières lignes de la sortie sont :

```
1.5
5.1
1.4166666
2.7460785
1.4142157
1.7371949
1.4142135
1.4442381
1.4142135
1.4145256
```

```
1.4142135
1.4142135
```

Ensuite, la dernière ligne se répète jusqu'à ce que l'exécution soit stoppée par l'utilisateur. Expliquer pourquoi et proposer une correction du programme qui assure que le programme se termine quelle que soit la valeur de `y`.

Listings

```
func main() {
    var big float32 = 1E6
    var small float32 = 1E-6

    var bsb float32 = (big + small) - big
    var bbs float32 = (big - big) + small

    fmt.Println("bsb = ", bsb)
    fmt.Println("bbs = ", bbs)
}
```

Listing 1 : Somme de nombres d'ordre de grandeur différents

```
func main() {
    const y float32 = 3
    var x float32 = 1
    var xnext float32 = 10

    for x*x != y {
        var tmp float32 = x - ((x*x - y) / (2 * x))
        fmt.Println(tmp)
        x = xnext
        xnext = tmp
    }
    fmt.Println("sqrt(" ,y ,") = ", x)
}
```

Listing 2 : Calcul de racine carré par la méthode de Newton