Solutions du TD pour la partie 1

I. Exemples de questions pour le 1er TD (questions sur les nombres positifs)

1. 8 2. 9 3. $10_{10} = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 1010_2$ 4. 21_{10}

5. $39_{10} = 32+0+0+4+2+1 = 10'0111_2$ 6. $64_{10} = 14_{60} = 24_{30} = 1'000'000_2 = 100_8 = 40_{16}$

7. $32{,}625_{10} = 10'0000{,}101_2 = 100'000{,}101_2 = 40{,}5_8$
   $= 0010'0000{,}1010_2 = 20{,}A_{16}$
   $128{,}75_{10} = 1000'0000{,}11_2 = 010'000'000{,}110_2 = 200{,}6_8$
   $= 1000'0000{,}1100_2 = 80{,}C_{16}$

8. La conversion s'effectue par des multiplications de puissances de 16, de 8 ou de 2 selon la base:
   $A{,}C_{16} = 10*16^0 + 12*16^{-1} = 10 + 3/4 = 10{,}75_{10}$
   $70{,}1_8 = 7*8^1 + 0*8^0 + 1*8^{-1} = 56{,}125_{10}$
   $1011{,}001_2 = 11{,}125_{10}$

II. Exercices pour le 2nd TD (exercices sur les nombres négatifs ou à virgule flottante)

1. Avant de donner les représentations de -32, il faut déterminer celle de +32:
   $+32 = 0010'0000_{2(\text{entier positif sur 8 bits})} = 0010'0000_{2(+\,/\,8)}$.
   Pour convertir en "signe + valeur absolue" (S.+V.A.), comme ici la V.A. prend moins de 8 bits (elle prend 6 bits: 10'0000; sur 7 bits: *010'0000*), il suffit de mettre à 1 le bit le plus à gauche:
   $-32 = \mathbf{1}\textit{010'0000}_{2(\text{S.+V.A. sur 8 bits})} = 1010'0000_{2(\text{S.+V.A. }/\,8)}$.
   Pour le "complément à 1" (c. à 1), par contre, il faut inverser chaque bit (du binaire pour +32) :
   $-32 = 1101'1111_{2(\text{c. à 1 sur 8 bits})} = 1101'1111_{2(\text{c.à1 }/\,8)}$
   Pour le "complément à 2" (c. à 2), on ajoute 1 à ce dernier résultat en complément à 1:
   $-32 = 1110'0000_{2(\text{c. à 2 sur 8 bits})}$ car $1101'1111_{2(\text{c. à 1 / 8 bits})} + 1 = 1110'0000_{2(\text{c. à 2 sur 8 bits})}$

   $128_{10} = 1000'0000_{2(\text{entier positif sur 8 bits})}$. La V.A. a déjà 8 bits: **le bit le plus à gauche est déjà à 1**. Donc, pour représenter -128 avec la méthode S.+V.A., il faudrait ajouter un 9ème bit. Il n'est donc pas possible de représenter -128 avec la méthode "S.+V.A.". Notez donc (même si ce n'est pas demandé dans l'énoncé de l'exercice) que l'on a:
   $128 = 0'1000'0000_{2(\text{entier positif sur 9 bits})}$
   donc: $-128 = 1'1000'0000_{2(\text{signe+V.A. sur 9 bits})}$
   et que: $127 = 0111'1111_{2(\text{ entier positif sur 8 bits})}$
   donc: $-127 = 1111'1111_{2(\text{signe+V.A. sur 8 bits})}$
   Pour la même raison (bit le plus à gauche déjà à 1), l'inversion des bits de $1000'0000_2$ (= + 128) pour trouver la représentation en complément à 1 de -128, on a $0111'1111_2$. Or, dans cette représentation, le bit le plus à gauche doit être 1, pas 0. Il n'est donc pas possible de représenter -128 en complément à 1. **Toutefois**, $0111'1111_2$ peut quand même être ré-utilisé (en y ajoutant 1) pour calculer le complément à 2 : $-128 = 1000'0000_{2(\text{c. à 2 sur 8 bits})}$
   puisque $1000'0000_2 = 1000'0000_2 + 1$
   Ces problèmes liés à la représentation de -128 sur 8 bits sont donc exactement les mêmes que ceux vus dans la section 3.2.2.2 de la partie 1 du cours pour la représentation de -4 sur 3 bits.

2. **1er cas:** $377_{8(\text{c. à } \mathbf{1} \text{ sur 8 bits})} = 11'111'111_{2(\text{c.à}\mathbf{1}\ /\ 8)} = -0$ et $-0 + 1 = 1$

$377_{8(\text{c.à}\mathbf{1}\ /\ 8)} + 001_8 = \mathbf{(1)00'000'000_{2(\text{c.à}\mathbf{1}\ /8\ )} = 0_{2(+\ /\ 8)}} + 1 = 1$ et $-0 + 1 = 1$

(**p.23:** "en complément à 1 sur n bits, un dépassement au delà de n bits est **ajouté** au résultat")

Donc, que l'on soit en décimal ou en binaire, l'addition que l'on fait est : $-0 + 1 = 1$

2nd cas: $377_{8(\text{c. à } \mathbf{2} \text{ sur 8 bits})} = 11'111'111_{2(\text{c.à}\mathbf{2}\ /\ 8)} = -1$ et $-1 + 1 = 0$

$377_{8(\text{c.à}\mathbf{2}\ /\ 8)} + 001_8 = \mathbf{(1)00'000'000_{2(\text{c.à}\mathbf{2}\ /\ 8)} = 0_{2(+\ /\ 8)}} = 0 = -1 + 1$

(car **p.23:** "en complément à 2 sur n bits, un dépassement au delà de n bits est **ignoré**").

177_8 $= 01'111'111_{2(+\ /8)} =$ ***127***

1er cas: $200_{8(\text{c.à } \mathbf{1} \text{ sur 8 bits})} = 10'000'000_{2(\text{c.à}\mathbf{1}\ /\ 8)} = -127$ et $-127 +$ ***127*** $= 0$

$200_{2(\text{c.à}\mathbf{1}\ /\ 8)} +$ ***177_8*** $= 11'111'111_{2(\text{c.à}\mathbf{1}\ /\ 8)} = -0 = -127 +$ ***127***

(ici, il n'y a pas de dépassement mais le résultat est en c.à1 / 8, donc à interpréter : comme pour tout c.à1, il faut inverser les bits pour trouver le nombre positif correspondant).

2nd cas: $200_{8(\text{c.à } \mathbf{2} \text{ sur 8 bits})} = 10'000'000_{2(\text{c.à}\mathbf{2}\ /\ 8)} = -128$ et $-128 +$ ***127*** $= -1$

$200_{8(\text{c.à}\mathbf{2}\ /\ 8)} +$ ***177_8*** $= 1111'1111_{2(\text{c.à}\mathbf{2}\ /\ 8)} = -1 = -128 +$ ***127***

(ici, il n'y a pas de dépassement mais le résultat est en c.à2/8, donc à interpréter : il faut enlever 1 pour trouver le c.à1 puis inverser les bits pour trouver le nombre positif).

Explication moins détaillée mais plus graphique et focalisée sur les dépassements :

Il est souvent plus facile d'effectuer des opérations arithmétiques sur des nombres signés en binaire plutôt qu'en octal ou en hexadécimal, car on voit directement la valeur du bit de signe.

$377_8 = 11111111_2$ $\quad$ $001_8 = 00000001_2$

$177_8 = 01111111_2$ $\quad$ $200_8 = 10000000_2$

– calcul de $377_8 + 001_8 = ?$

complément à 1

```
  11111111
+ 00000001
----------
100000000on reporte
 |_____1 la retenue
----------
 00000001
```

on obtient donc +1.

complément à 2

```
  11111111
+ 00000001
----------      on laisse tomber
100000000       la retenue

----------
 00000000
```

on obtient donc 0.

– calcul de $177_8 + 200_8 = ?$

complément à 1

```
  01111111
+ 10000000
----------
 11111111
```

on obtient –0

complément à 2

```
  01111111
+ 10000000
----------
 11111111
```

on obtient –1

3. $276'3200'0000_8 = 10'111'110'011'010'000'000'000'000'000'000_{2(\text{sur 32 bits})}$
$= 1'01111100'11010000000000000000000_{2(\text{sur 32 bits})}$

Signe de la mantisse ("a"): 1 (bit 31); la mantisse est donc négative.

Mantisse en valeur absolue: $0,1101_2 = 2^{-1} + 2^{-2} + 2^{-4} = 0,5 + 0,25 + 0,0625 = 0,8125$

Le biais d'un exposant est la moitié de sa plage de valeurs (- 1 dans le format IEEE 754 mais pas dans ce TD comme précisé p.25 du cours et en plus la question précise qu'ici le biais est une puissance de 2). Donc, pour un exposant sur 8 bits, le biais est $2^{8-1} = 2^7 = 128$.

Exposant biaisé: $0111'1100_2 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 = 64 + 32 + 16 + 8 + 4 = 124$.

Exposant réel = exposant biaisé - biais = 124 -128 = -4.

Le nombre, sous la forme demandée, est donc: $-0,8125 * 2^{-4}$

4. $278 = 1'0001'0110_2 = 0,1000'1011'0_2 * 2^9$

Signe de la mantisse: 0 (bit 31); la mantisse est donc positive.

Exposant réel: 9; biais: 128. Exposant biaisé: $9 + 128 = 137 = 1000'1001_2$.

Mantisse sur 23 bits, sans le premier bit à 1 (non-représenté, comme précisé): $00010110...0_2$

En concaténant (dans l'ordre) les 3 derniers résultats: 0_2, $1000'1001_2$ et $00010110...0_2$,

on a: $0'1000'1001'00010110...0_2 = 01'000'100'100'010'110'0...0_2 = 1044'2600'000_8$.

De même, $-6,53125 = -110,10001_2 = - 0,1101'0001_2 * 2^3$

Bit 31: 1 (puisque le nombre à encoder est négatif).

Exposant réel: 3; biais: 128. Exposant biaisé: $3 + 128 = 131 = 1000'0011_2$.

Mantisse sur 23 bits, sans le premier bit à 1: $10100010..0_2$.

En concaténant (dans l'ordre) les 3 derniers résultats: 1_2, $1000'0011_2$ et $10100010...0_2$,

on a: $1'1000'0011'101000100...0_2 = 11'000'001'110'100'010'0...0_2 = 30164200000_8$.

5. a) Le plus petit nombre positif correspond à la plus petite mantisse positive et au plus petit exposant :

- les bits de la plus petite mantisse sont 1000000 (la plus petite mantisse est donc $0,1_2$);
- plus petit exposant biaisé: $0000_2 = 0$; le biais étant 2^3, le plus petit exposant réel est: $0 - 2^3 = -8$.

Donc, le plus petit nombre réel positif est: $0,1_2 * 2^{-8} = 2^{-9}$.

Le plus grand nombre réel positif correspond à la plus grande mantisse positive et au plus grand exposant:

- les bits de la plus grande mantisse sont 1111111 (→ plus grande mantisse: $0,1111111_2$)
- le plus grand exposant biaisé: $1111_2 = 15$;
- le biais étant 2^3 (= 8), le plus grand exposant réel est: 15 - 8 = 7.

Donc, le plus grand nombre réel positif est: $+(1 - 2^{-7}) * 2^{-7} = 2^7 - 2^0 = 128 - 1 = 127$.

En résumé, l'intervalle fermé est, sous la forme demandée : $[1*2^{-9}, 127*2^0] = [2^{-9}, 127]$.

b) Pour $X = AE8_{16} = 1010'1110'1000_2 = 1'0101'1101000_2$.

Bit de signe de la mantisse : 1 => mantisse négative.

Exposant biaisé = $0101_2 = 5$; biais: 8; exposant réel = 5 - 8 = -3.

Les bits de la mantisse sont 1101'000, la mantisse est: $0,1101_2$.

Donc, $X = -0,1101_2 * 2^{-3} = -1101_2 * 2^{-7} = -13 * 2^{-7}$.

Pour $Y = 9D0_{16} = 1'0011'1010000_2$.

Bit de signe de la mantisse : 1 => mantisse négative.

Exposant biaisé = $0011_2 = 3$; biais: 8; exposant réel = 3 - 8 = -5.

$Y = -0,101_2 * 2^{-5} = -101_2 * 2^{-8} = -5 * 2^{-8}$.

c) $Z = Y - X = (-5 * 2^{-8}) - (-13 * 2^{-7})$

$= (-5 * 2^{-8}) - (-26 * 2^{-8}) = 21 * 2^{-8}$.

d) $-32,625 = -10'0000,101_2 = -0,1000'00101_2 * 2^6$.

Signe négatif -> bit 11 à 1.

La mantisse normalisée ($0,100000101_2$) doit être tronquée à 7 bits : $0,1000001_2$.

Exposant réel: 6; biais: 8; exposant biaisé = 6 + 8 = 14 = 1110_2.

Le nombre -32,625 est représenté par $1'1110'1000001_2 = 111'101'000'001_2 = 7501_8$.

Il faut noter que cette valeur octale ne correspond pas exactement à 31,625 mais en est la représentation obtenue par troncation.

6. $377'2400'0000_8$ considéré comme un entier, en complément à 1, représente -11534335, en complément à 2, représente -11534336.

$377'2400'0000_8$ considéré comme un réel représenté comme dans l'exercice 7:

- exposant: biaisé = $1111'1110_2 = 254$; exposant réel = 254 -128 = 126;
- bit de signe à 1; mantisse normalisée: 1010..0, donc $0,101_2$.

Le nombre représenté est donc: $-0,101_2 * 2^{126} = -0,625 * 2^{126}$.