Travaux Dirigés n°2

Corrigé

Chapitre 2 : Représentation de l'information

Conventions : 1 Kio $= 2^{10}$ octets, 1 Mio $= 2^{20}$ octets, 1 Gio $= 2^{30}$ octets ; 1 Ko $= 10^3$ octets et 1 Mo $= 10^6$ octets. Sauf indication contraire, les entiers signés sont représentés sur 8 bits. Un indice précise la base utilisée : $(\cdots)_2$, $(\cdots)_8$, $(\cdots)_{10}$ ou $(\cdots)_{16}$.

Exercice 1 : Bits, octets et unités

1. Combien de bits contiennent 3 octets ? 2 Kio ?
2. Combien de valeurs différentes peut-on représenter avec 10 bits ?
3. Combien de bits faut-il au minimum pour coder 100 valeurs différentes ?
4. Un fichier contient 12 000 000 bits. Donner sa taille en octets puis en Mo.
5. Exprimer 1 Gio en octets.

Correction.

1. $3 \times 8 = \boxed{24 \text{ bits}}$; 2 Kio $= 2048$ octets $= 2048 \times 8 = \boxed{16\,384 \text{ bits}}$.
2. $2^{10} = \boxed{1024}$ valeurs différentes.
3. $2^6 = 64 < 100 \leq 2^7 = 128$, donc il faut $\boxed{7 \text{ bits}}$.
4. $12\,000\,000/8 = \boxed{1\,500\,000 \text{ octets}} = \boxed{1{,}5 \text{ Mo}}$.
5. 1 Gio $= 2^{30} = \boxed{1\,073\,741\,824 \text{ octets}}$.

Exercice 2 : Conversions entre les bases

Compléter le tableau suivant en détaillant les étapes de conversion : divisions successives, regroupement de bits ou décomposition en puissances de la base.

Décimal	Binaire	Octal	Hexadécimal
156			
217			
	10110111		
			3A7
		745	

Correction.

Décimal	Binaire	Octal	Hexadécimal
156	10011100	234	9C
217	11011001	331	D9
183	10110111	267	B7
935	1110100111	1647	3A7
485	111100101	745	1E5

Détails :

- $156 = 128 + 16 + 8 + 4 = (10011100)_2$; groupes de 3 bits : `010|011|100` $= (234)_8$; groupes de 4 bits : `1001|1100` $= (9C)_{16}$.
- $217 = 128 + 64 + 16 + 8 + 1 = (11011001)_2$; `011|011|001` $= (331)_8$; `1101|1001` $= (D9)_{16}$.
- $(10110111)_2 = 128 + 32 + 16 + 4 + 2 + 1 = 183$; `010|110|111` $= (267)_8$; `1011|0111` $= (B7)_{16}$.
- $(3A7)_{16} = 3 \times 256 + 10 \times 16 + 7 = 935$; `0011|1010|0111` $= (1110100111)_2$; regroupé par 3 : `001|110|100|111` $= (1647)_8$.
- $(745)_8 = 7{\times}64{+}4{\times}8{+}5 = 485$; `111|100|101` $= (111100101)_2$; regroupé par 4 : `0001|1110|0101` $= (1E5)_{16}$.

Exercice 3 : Opérations arithmétiques en binaire

Effectuer les opérations suivantes en binaire, puis vérifier les résultats en décimal.

1. $(10110)_2 + (01101)_2$
2. $(11011)_2 - (01101)_2$
3. $(1101)_2 \times (101)_2$
4. $(1100111)_2 \div (1001)_2$: donner le quotient et le reste.

Correction.

1. $10110 + 01101 = \boxed{(100011)_2}$; vérification : $22 + 13 = 35 = (100011)_2$.
2. $11011 - 01101 = \boxed{(01110)_2}$; vérification : $27 - 13 = 14 = (1110)_2$.
3. $1101 \times 101 = 1101 + 110100 = \boxed{(1000001)_2}$; vérification : $13 \times 5 = 65 = (1000001)_2$.
4. On prend $1100 \geq 1001$: $q = 1$, reste 0011 ; on abaisse un bit : $0111 < 1001$ donc $q = 0$; on abaisse un bit : $1111 \geq 1001$ donc $q = 1$, reste 0110 ; on abaisse un bit : $1101 \geq 1001$ donc $q = 1$, reste 0100. D'où $\boxed{\text{quotient} = (1011)_2,\ \text{reste} = (100)_2}$; vérification : $103 = 9 \times 11 + 4$.

Exercice 4 : Représentation des entiers signés

1. Représenter $+37$, -37 et -100 sur 8 bits selon les trois méthodes suivantes :
   - signe et valeur absolue ;
   - complément à 1 ;
   - complément à 2.
2. Donner la valeur décimale des nombres suivants, codés en complément à 2 sur 8 bits : `11001001`,

`10000000` et `01111111`.

3. Donner l'intervalle des entiers représentables en complément à 2 sur 6 bits, puis sur 12 bits. Combien de bits faut-il au minimum pour représenter tous les entiers de -300 à 300 ?

4. Calculer $45 - 70$ sur 8 bits en complément à 2, en effectuant l'addition de 45 et de -70. Donner la valeur décimale du résultat.

5. Calculer $100 + 60$ sur 8 bits en complément à 2. Interpréter le résultat obtenu et préciser s'il y a un dépassement de capacité (*overflow*).

Correction.

1. On a $37 = (00100101)_2$ et $100 = (01100100)_2$.

Valeur	Signe et val. abs.	Complément à 1	Complément à 2
$+37$	`00100101`	`00100101`	`00100101`
-37	`10100101`	`11011010`	`11011011`
-100	`11100100`	`10011011`	`10011100`

2. `11001001` : bit de poids fort égal à 1, donc négatif ; inversion : `00110110`, puis $+1$: `00110111` $= 55$; valeur $\boxed{-55}$.
`10000000` = $\boxed{-128}$ (plus petite valeur) ; `01111111` = $\boxed{+127}$ (plus grande valeur).

3. Intervalle : $-2^{n-1} \leq N \leq 2^{n-1} - 1$. Sur 6 bits : $\boxed{-32 \leq N \leq 31}$; sur 12 bits : $\boxed{-2048 \leq N \leq 2047}$.
Pour $[-300, 300]$: 9 bits donnent $[-256, 255]$ (insuffisant), 10 bits donnent $[-512, 511]$: $\boxed{\text{10 bits}}$.

4. 45 = `00101101` ; 70 = `01000110`, donc -70 = `10111010`.
`00101101` + `10111010` = `11100111` : bit de poids fort 1, valeur $-(00011000 + 1) = -25$; résultat $\boxed{-25}$.

5. 100 = `01100100` et 60 = `00111100` ; somme : `10100000`. Le bit de poids fort vaut 1 : on lirait -96 alors que $100 + 60 = 160 > 127$. Il y a donc **dépassement de capacité** (*overflow*) : deux positifs donnent un résultat « négatif », le résultat n'est pas représentable sur 8 bits signés.

Exercice 5 : Représentation des caractères

1. Donner le code ASCII, en décimal, hexadécimal et binaire sur 8 bits, des caractères `C`, `z`, `7` et de l'espace. On rappelle : `A` = 65, `a` = 97 et `0` = 48.

2. Décoder la suite d'octets ASCII suivante, donnée en hexadécimal : `48 45 4C 4C 4F`.

3. Combien de caractères distincts l'ASCII classique sur 7 bits permet-il de coder ? Quelle est la différence entre le code d'une lettre minuscule et celui de sa majuscule correspondante ?

4. En UTF-8, les caractères sont codés sur 1 à 4 octets. Calculer la taille en octets des chaînes suivantes : `Bonjour`, `Été`, un mot arabe de 5 lettres et `5€`.

   Précision : pour le mot arabe, supposer que chacune des cinq lettres est représentée sur 2 octets en UTF-8.

Correction.

1. `C` $= 67 = (43)_{16}$ = `01000011` ; `z` $= 122 = (7A)_{16}$ = `01111010` ; `7` $= 55 = (37)_{16}$ = `00110111` ; espace $= 32 = (20)_{16}$ = `00100000`.

2. $48 = \text{H}$, $45 = \text{E}$, $4C = \text{L}$, $4C = \text{L}$, $4F = \text{O}$: la chaîne est $\boxed{\texttt{HELLO}}$.
3. $2^7 = \boxed{128}$ caractères. $\texttt{a} - \texttt{A} = 97 - 65 = \boxed{32}$: un seul bit change (celui de poids 2^5).
4. `Bonjour` : 7 caractères ASCII, soit $\boxed{7}$ octets ; `Été` : $2+1+2 = \boxed{5}$ octets ; mot arabe de 5 lettres : $5 \times 2 = \boxed{10}$ octets ; `5€` : $1 + 3 = \boxed{4}$ octets (le symbole de l'euro occupe 3 octets en UTF-8).

Exercice 6 : Nombres réels : virgule fixe et virgule flottante

1. Convertir en décimal les nombres binaires suivants :

$$(110{,}101)_2 \qquad \text{et} \qquad (1001{,}011)_2.$$

2. Convertir en binaire :
   — $(10{,}375)_{10}$, en donnant une représentation exacte ;
   — $(0{,}3)_{10}$, en donnant les 8 premiers bits de la partie fractionnaire, la valeur approchée obtenue et l'erreur absolue.
3. On utilise une représentation en virgule fixe sur 8 bits, avec 4 bits pour la partie entière et 4 bits pour la partie fractionnaire.
   a) Donner la plus petite valeur positive non nulle, la plus grande valeur représentable et le pas de quantification.
   b) Coder 5,3 dans ce format et calculer l'erreur absolue de représentation.
4. En IEEE 754 simple précision, un nombre est codé sur 32 bits : 1 bit de signe, 8 bits d'exposant avec un biais de 127 et 23 bits de fraction. Représenter 5,75 puis $-13{,}25$ en binaire, puis donner leur représentation hexadécimale sur 32 bits.
5. Décoder le nombre IEEE 754 simple précision suivant : `0x41200000`.
6. Expliquer pourquoi le calcul $0{,}1 + 0{,}2$ peut donner un résultat légèrement différent de 0,3 sur un ordinateur.

Correction.

1. $(110{,}101)_2 = 4 + 2 + 0 + 0{,}5 + 0 + 0{,}125 = \boxed{6{,}625}$; $(1001{,}011)_2 = 8 + 1 + 0{,}25 + 0{,}125 = \boxed{9{,}375}$.
2. — $10 = (1010)_2$; $0{,}375 \times 2 = 0{,}75$ (0), $0{,}75 \times 2 = 1{,}5$ (1), $0{,}5 \times 2 = 1{,}0$ (1) donc $0{,}375 = (0{,}011)_2$ et $\boxed{(10{,}375)_{10} = (1010{,}011)_2}$.
   — 0,3 : 0,6 (0), 1,2 (1), 0,4 (0), 0,8 (0), 1,6 (1), 1,2 (1), 0,4 (0), 0,8 (0)… ; donc $0{,}3 = (0{,}01\overline{0011})_2$ (développement infini). Avec 8 bits : $(0{,}01001100)_2 = \frac{1}{4} + \frac{1}{32} + \frac{1}{64} = 0{,}296875$, erreur absolue $= 0{,}3 - 0{,}296875 = \boxed{0{,}003125}$.
3. a) Plus petite valeur positive non nulle = pas $= 2^{-4} = \boxed{0{,}0625}$; plus grande valeur : $(1111{,}1111)_2 = 15 + \frac{15}{16} = \boxed{15{,}9375}$.
   b) $5{,}3 \times 16 = 84{,}8 \approx 85 = (01010101)_2$, soit $(0101{,}0101)_2 = 5 + \frac{5}{16} = 5{,}3125$; erreur absolue $= 5{,}3125 - 5{,}3 = \boxed{0{,}0125}$.
4. $5{,}75 = (101{,}11)_2 = (1{,}0111)_2 \times 2^2$: $S = 0$; $E = 2 + 127 = 129 = \texttt{10000001}$; $F = \texttt{01110000000000000000000}$.
   Mot : `0 10000001 01110000000000000000000` $= \boxed{\texttt{0x40B80000}}$.
   $13{,}25 = (1101{,}01)_2 = (1{,}10101)_2 \times 2^3$: $S = 1$; $E = 3 + 127 = 130 = \texttt{10000010}$; $F =$

`10101000000000000000`.
Mot : `1 10000010 10101000000000000000` = $\boxed{\texttt{0xC1540000}}$.

5. `0x41200000` = `0100 0001 0010 0000 0000 0000 0000 0000` : $S = 0$; exposant `10000010` = 130, donc $E = 130 - 127 = 3$; fraction `01000...`, donc mantisse $(1{,}01)_2 = 1{,}25$.
   Valeur : $1{,}25 \times 2^3 = \boxed{10}$.

6. Les nombres 0,1, 0,2 et 0,3 n'ont pas de représentation binaire finie (développements périodiques) : ils sont arrondis dans la mémoire. La somme des deux valeurs arrondies $0{,}1 + 0{,}2$ peut donc différer de la valeur arrondie de 0,3 (écart de l'ordre de 10^{-17} en double précision).