Compilation et Analyse Lexicale
TD 1 - Correction

Licence 3 Informatique (2022-2023)

Jairo Cugliari, Guillaume Metzler
Institut de Communication (ICOM)
Université de Lyon, Université Lumière Lyon 2

jairo.cugliari@univ-lyon2.fr guillaume.metzler@univ-lyon2.fr

Exercice 1

Représentez les nombres 28_{10}, 129_{10}, 147_{10}, 255_{10} sous leur forme binaire. À partir de cette représentation binaire, vous en déduirez leur représentation hexadécimale.

On rappelle que pour passer de la représentation décimale à la représentation binaire, il suffit d'effectuer une suite de division euclidienne par 2. La représentation binaire est ensuite obtenue en lisant les restes du bas vers le haut.
On va le faire pour les deux premiers nombres et on vous laisse faire les autres.

$$\begin{aligned} 28 &= 14 \times 2 + 0, \\ 14 &= 7 \times 2 + 0, \\ 7 &= 3 \times 2 + 1, \\ 3 &= 1 \times 2 + 1, \\ 1 &= 0 \times 2 + 1. \end{aligned}$$

La représentation du nombre 28 en base 2 est donc

$$0001\ 1100.$$

On prendra l'habitude de regrouper l'écriture d'un nombre en base 2 par blocs de 4 bits (utile pour la suite !!).

On procède de la même façon pour le nombre 129 que l'on souhaite écrire en binaire.

$$\begin{aligned} 129 &= 64 \times 2 + 1, \\ 64 &= 32 \times 2 + 0, \\ 32 &= 16 \times 2 + 0, \\ 16 &= 8 \times 2 + 0, \\ 8 &= 4 \times 2 + 0, \\ 4 &= 2 \times 2 + 0, \\ 2 &= 1 \times 2 + 0, \\ 1 &= 0 \times 2 + 1, \end{aligned}$$

La représentation du nombre 28 en base 2 est donc

$$1000\ 0001.$$

De la même façon, nous avons les nombres 147 et 255 s'écrivent respectivement en base 2 :

$$1001\ 0011 \quad \text{et} \quad 1111\ 1111.$$

Pour traduire un nombre binaire en héxadécimal, il suffit simplement de regrouper les bits en paquet de 4 afin d'avoir une traduction immédiate. Rappellons que le système héxadécimale se base sur les chiffres 0 à 9 et les lettres A à F soit 16 caractères. Or avec 4 bits, nous sommes capables de décrire 16 valeurs différentes.

Ainsi, la transformation est immédiate et nous avons les représentations des nombres $28, 129, 147$ et 255 qui, en héxédécimal, sont respectivement données par

$$1C_{16}, \quad 81_{16}, \quad 93_{16} \quad \text{et} \quad FF_{16}.$$

Exercice 2

1. Les nombres 11000010_2, 10010100_2, 11101111_2, 10000011_2, 10101000_2 sont-ils pairs ou impairs ?

   Pour identifier la parité d'un nombre dont la représentation est donnée en binaire, il suffit de regarder la valeur du bit le plus à droite.

   - Si le dernier bit est égal à 1 : le nombre est **impair**.

- Si le dernier bit est égal à 0 : le nombre est **pair**.

Ainsi, les cinq nombres représentés en base 2 sont respectivement :

pair - pair - impair - impair - pair.

2. Lesquels sont divisibles par 4, 8 ou 16 ?

   Pour déterminer si un nombre écrit en base 2 est divisble par 2^k, il faut et il suffit que les k premiers bits (en partant de la droite) soient tous égaux à 0).

   Illustrons cela sur les deux premières représentations binaires :

   $$1100\ 0010 = 2 + 2^6 + 2^7 = 2 \times (1 + 2^5 + 2^6).$$

   Ce qui montre que ce premier nombre est bien divisible par 2 et seuleument par 2. Pour ce nombre, seul son dernier bit est nul.

   Vérifons que le deuxième nombre dont les deux derniers bits sont égaux à 0 est bien divisible par 4 (et par conséquent 2).

   $$1001\ 0100 = 2^2 + 2^4 + 2^7 = 2 \times (2 + 2^3 + 2^6) = 4 \times (1 + 2^2 + 2^5).$$

   Ce qui montre bien que ce nombre est divisible par 4 et 2.

   De la même façon nous avons :

   - 1110 1111 qui n'est pas divisible par 2, par conséquent il n'est pas divisible par 4 et 8. En effet, il s'agit d'un nombre impair.
   - 1000 0011 est un nombre impair, comme dans la cas précédent.
   - 1010 1000 voit ses trois premiers bits égaux à 0, il est donc divisible par $2^3 = 8$, et donc par 4 et 2 également.

3. Donnez le quotient et le reste d'une division entière par 2, 4 et 8 de ces nombres.

   On notera Q et R respectivment le quotient et le reste des divisions euclidiennes. Nous donnerons ces valeurs en base 10. Il suffit de faire la conversion en base 10 et de faire les divisions euclidiennes.

	Division par 2		Division par 4		Division par 8	
	Q	R	Q	R	Q	R
1100 0010	97	0	48	2	24	2
1001 0100	74	0	37	0	18	4
1110 1111	119	1	59	3	29	7
1000 0011	65	1	32	3	16	3
1010 1000	84	0	42	0	21	0

4. En généralisant, que suffit-il de faire pour obtenir le quotient et le reste d'une division entière d'un nombre binaire par 2^k ?

   Imaginons que l'on dispose écrit sur un n bits et considérons sa division euclidienne par 2^k où $k < n$.

   Nous avons alors les règles suivantes

   - **Quotient :** il est égal au nombre représenté par les $n - k$ premiers bits (en partant de la gauche).
   - **Reste :** il est égal au nombre représenté par les k derniers bits (en partant de la gauche).

   Reprenons l'exemple avec la représentation binaire 1110 1111 dont on souhaite déterminer le reste de la division euclidienne par $8 = 2^3$.

   $$1110\ 1111 = \underbrace{1\ 1101}_{\text{Quotient}}\ \underbrace{111}_{\text{Reste}}.$$

   Le quotient est donc égal à 0001 1101 en base 2 soit $1 + 4 + 8 + 16 = 29$ en base 10. Le reste est égal à 0111 en base 2 soit $1 + 2 + 4 = 7$ en base 10. C'est ce que nous avions précédemment trouvé.

5. Si l'on souhaite multiplier un nombre binaire quelconque par une puissance de 2, quelle méthode peut-on utiliser afin d'éviter la multiplication ?

   Prenons un petit exemple et considérons le nombre 5 écrit en base 2 : 0101. Regardons un peu le résultat de quelques multiplications par des puissances de 2

   $$\begin{aligned} 5 \times 1 = 5 &\rightarrow 0000\ 0101, \\ 5 \times 2 = 10 &\rightarrow 0000\ 1010, \\ 5 \times 2^2 = 20 &\rightarrow 0001\ 0100, \end{aligned}$$

$$5 \times 2^3 = 40 \quad \rightarrow \quad 0010\ 1000.$$

Ainsi, on remarque que si l'on souhaite multiplier un nombre par $2^k, k > 0$, il suffit, pour son écriture binaire, de rajouter k fois le bit 0 à l'extrêmité **droite** de son écriture binaire.
C'est la même règle que l'on applique lors de multiplication par des puissances de 10 sur des nombres écrits en base 10.

6. Si l'on souhaite multiplier un nombre binaire quelconque par 3 ou par 10, quelle méthode peut-on utiliser pour éviter la multiplication ?

   - Lorsque l'on souhaite multiplier un nombre par 3, il suffit d'utiliser le fait que $3 = 2 + 1$ et on pourra alors appliquer la règle utilisée pour ma multiplication par 2 (*i.e.* décalage des bits d'un rang vers la droite) et de faire ensuite l'addition avec le nombre de départ.

   - Lorsque l'on souhaite multiplier un nombre par 10, on va simplement utiliser le fait que $10 = 2^3 + 2^2$ et faire l'addition en utilisant les règles vues pour pour la multiplication par des puissances de 2. Il restera l'addition à effectuer.

Exercice 3

Donnez les valeurs décimales, minimales et maximales, que peuvent prendre des nombres signés et non signés codés sur $4, 8, 16, 32$ et n bits.

Résumons les résultats pour les deux cas

- **Cas non signé :** ici, il n'y a pas de règle particulière.

Nombre de bits	4	8	16	32	n
Valeur minimale	0	0	0	0	0,
Valeur maximale	$2^4 - 1$	$2^8 - 1$	$2^{16} - 1$	$2^{32} - 1$	$2^n - 1$

- **Cas signé :** ici, il faut prendre en compte le bit qui va nous servir à savoir si le chiffre est positif ou négatif. Si le premier bit (à l'extrême gauche) est égal à 1, alors le nombre est négatif, s'il est égal à 0, il est positif.
  Ainsi, un nombre écrit sur n bits signé pourra prendre des valeurs allant de -2^{n-1} à $2^{n-1} - 1$. Ce choix de représentation s'appelle **complément à deux**.

Nombre de bits	4	8	16	32	n
Valeur minimale	-2^3	-2^7	-2^{15}	-2^{31}	-2^{n-1},
Valeur maximale	2^3-1	2^7-1	$2^{15}-1$	$2^{31}-1$	$2^{n-1}-1$

On peut également adopter la règle dite **signe-magnitude** qui permet de représenter des nombres allant de $-2^{n-1}+1$ à $-2^{n-1}-1$. Avec cette règle, le chiffre 0 a deux représentations possibles.

Exercice 4

Soit les deux nombres binaires suivants : 1111 1111 et 1011 0110.

1. Donnez leur représentation décimale s'ils sont codés sur 8 bits signés.

   On notera en rouge l'écriture correspondante, en base 10, avec la règle signe magnitude et en bleu avec la règle complément à deux.

   Le complément à deux consiste à utiliser le premier bit comme le bit de signe, s'il est égal à 0, on ne fait change rien par rapport à la **règle de signe**. S'il est égal 1, la valeur décimale correspondante et à changer les valeurs des autres bits ($0 \to 1$ et $1 \to 0$) et enfin d'ajouter 1. Ce qui nous donne :

   $$1111\ 1111 \to -127 \quad \text{et} \quad -1.$$

   En effet le complément à deux nous donne la représentation 000 0000 (en ne tenant pas compte du bit signé). De la même façon

   $$1011\ 0110 \to -54 \quad \text{et} \quad -74.$$

   En effet le complément à deux nous donne la représentation 100 1001 (en ne tenant pas compte du bit signé).

2. Donnez leur représentation décimale s'ils sont codés sur 16 bits signés.

   S'ils sont codés sur 16 bits signés, le bit de poids le plus fort étant 0, on manipule des entiers positifs. Ainsi les valeurs décimales associées sont respectivement égales à 255 et 182 (avec la règle du bit signé).

3. Soit le nombre entier négatif suivant : -80_{10}.

(a) On souhaite le coder sur 8 bits signés. Donnez sa représentation binaire et sa représentation hexadécimale.

On commence par donner la représentation binaire du nombre décimal 80

$$0101\ 0000.$$

On détermine ensuite le complément de ce nombre (il s'agit de switcher les valeurs 0 et 1)

$$1010\ 1111.$$

Il faut ensuite lui ajouter la valeur 1

$$\begin{array}{cccccccccc} & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ + & & & & & & & & 1 \\ \hline & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}$$

On obtient ainsi la représentation binaire $1011\ 0000$ et sa représentation hexadécimale est $D0$.

(b) On souhaite le coder sur 16 bits signés. Donnez sa représentation binaire et sa représentation hexadécimale.

On procèdera de la même façon mais on code sur 16 bits cette fois ci. On commence par donner la représentation binaire du nombre décimal 80

$$0000\ 0000\ 0101\ 0000.$$

On détermine ensuite le complément de ce nombre (il s'agit de switcher les valeurs 0 et 1)

$$1111\ 1111\ 1010\ 1111.$$

Il faut ensuite lui ajouter la valeur 1

$$\begin{array}{cccccccccccccccc} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ + & & & & & & & & & & & & & & & & 1 \\ \hline & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}$$

On obtient ainsi la représentation binaire $1111\ 1111\ 1011\ 0000$ et sa représentation hexadécimale est $FFB0$.

Exercice 5

1. Donnez, en puissance de deux, le nombre de bits que contiennent les grandeurs suivantes : 128 Kib, 16 Mib, 2 Kio, 512 Gio.

   On rappelle que $Kib = 2^{10}$ bits et $Gib = 2^{30}$ bits. On rappelle que 1 octet est égal à 8 bits. De la même façon $Kio = 8 \times 2^{10} = 2^{13}$ bits et $Gio = 8 \times 2^{30}$ octets soit $(2^3)^{30} = 2^{33}$ bits On a ainsi les résultats suivants :

   - 128 Kib $= 2^{17}$ bits,
   - 16 Mib $= 2^{24}$ bits,
   - 2 Kio $= 2 \times 2^{13} = 2^{14}$ bits,
   - 512 Gio $= 2^9 \times 2^{33}$ bits soit 2^{42} bits.

2. Donnez, à l'aide des préfixes binaires (Kio, Mio ou Gio), le nombre d'octets que contiennent les grandeurs suivantes : 2 Mo, 214 bits, 226 octets, 232 octets. Vous choisirez un préfixe qui permet d'obtenir la plus petite valeur numérique entière.