IUT de Lens – BUT Info S1
R1.03 – Introduction à l'architecture des ordinateurs

TD n^{o}1

codage des entiers naturels

L'objectif du cours R1.03 est de comprendre le fonctionnement d'un ordinateur et de connaître ses composants. Dans ce TD, nous étudions le binaire, et plus généralement comment les entiers sont représentés et manipulés par la machine.

CONSIGNES

Les calculatrices font aisément les calculs qu'on vous demande de faire. De ce fait, elles sont interdites dans ce cours ! Il faudra faire les calcul "à la main" ou de tête.

NOMBRES, CHIFFRES, BASE

Un nombre s'écrit avec des chiffres, tout comme un mot s'écrit avec des lettres. Habituellement, on utilise les chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. Il y a 10 chiffres, on dit qu'on compte en base 10 (décimal). Vous savez que dans un nombre, le chiffre des unités doit être multiplié par 1 (soit 10^0), le chiffre des dizaines doit être multiplié par 10 (soit 10^1), le chiffre des centaines doit être multiplié par 100 (soit 10^2), le chiffre des milliers doit être multiplié par 1000 (soit 10^3), et ainsi de suite. Par exemple, le nombre qui s'écrit 54321 est égal à $5 * 10^4 + 4 * 10^3 + 3 * 10^2 + 2 * 10^1 + 1 * 10^0$.
Vous savez bien sûr qu'un ordinateur fonctionne en binaire, c'est à dire en base 2. Il n'utilise que 2 chiffres (0 et 1) qui correspondent aux deux états possibles d'un transistor (passant ou bloqué). En informatique, on utilise aussi couramment les bases 8 (octal), et 16 (hexadécimal (de hexa=6 et décimal=10)).

Exercice 1 : Donnez les valeurs des puissances de 2, de 2^0 à 2^{16}. Autrement dit, donnez la valeur de 2^i pour i allant de 0 jusque 16. Les puissances de 2 jusque 2^{10} doivent être connues par cœur. Il est utile de connaître aussi les puissances jusque 2^{16}.

Exercice 2 : La valeur d'un nombre change-t-elle quand on ajoute des zéros à gauche de ce nombre (1, 01, 001, ...) ? La valeur d'un nombre change-t-elle quand on ajoute des zéros à droite de ce nombre (1, 10, 100, ...) ? Comment appelle-t-on les zéros qui ne comptent pas ?

BASE

Quand un nombre est écrit en base b, le chiffre le plus à droite doit être multiplié par b^0, le chiffre immédiatement à sa gauche par b^1, puis le chiffre suivant par b^2 et ainsi de suite. Par exemple, en base b, le nombre $c_4c_3c_2c_1c_0$ (où chaque c_i est un chiffre) est égal à $c_4 * b^4 + c_3 * b^3 + c_2 * b^2 + c_1 * b^1 + c_0 * b^0$. En base b, il y a au total b chiffres, de 0 jusque $b-1$.
En base 16, on utilise les chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F. Le chiffre A correspond à la valeur 10, B à 11 et ainsi de suite jusque F qui correspond à 15.
Pour indiquer qu'un nombre est écrit en base b, on le mettra entre parenthèses et on indiquera en indice la valeur de b, par exemple $(c_4c_3c_2c_1c_0)_b$. Dans ce cours, on utilisera aussi les notations usuelles en C++ : un nombre qui commence par 0b est écrit en **b**inaire (base 2), un nombre qui commence par un 0 (non significatif) est écrit en base 8 (octal) et un nombre qui commence par 0x est écrit en base 16 (he**x**adécimal). En C++, quand il y a beaucoup de chiffres, on peut les grouper et séparer les groupes par un guillemet simple, par exemple 0x1234'ABCD.

Exercice 3 : Montrez que 0b1010'0011 = $(10100011)_2$ = 0xA3 = $(A3)_{16}$ = 163 = $(163)_{10}$ = 0243 = $(243)_8$. Écrire un nombre en base b ou dans une autre base b' change-t-il la valeur de ce nombre ?

Exercice 4 : Donnez en base 10 la valeur de $(1235321)_4$.

Exercice 5 : Les nombres ci-dessous sont donnés en binaire. Donnez leur valeur en base 10.

1. 0b100
2. 0b1001
3. 0b1'0111
4. 0b10'0011
5. 0b0101'0010
6. 0b1011'0110
7. 0b0100'0011'0111
8. 0b1010'0101'0010'1111

BIT, QUARTET, OCTET, ...

Un chiffre en binaire s'appelle un bit (contraction de binary digit). Un quartet est un nombre qui s'écrit sur 4 bits. Un octet est un nombre qui s'écrit sur 8 bits. Un seizet (terme peu usité) est un nombre qui s'écrit sur 16 bits. De manière générale, on parle de mot de n bits pour un nombre qui s'écrit sur n bits.

Exercice 6 : Énumérez à partir de 0 les 11 premiers nombres en base 2, 3, 5 et 10. Énumérez à partir de 8 les 11 premiers nombres en base 16. De manière générale, dans une base b, comment fait-on pour passer d'un nombre à son successeur, autrement dit pour incrémenter un nombre (incrémenter signifie ajouter 1, le nom correspondant est incrémentation) ?

Exercice 7 : Quel est le successeur du nombre 0xFFFE'FFFF ?

Exercice 8 : En utilisant la méthode précédente pour passer d'un nombre à son successeur (incrémentation), montrez que $2^0 + 2^1 + \ldots + 2^{n-2} + 2^{n-1} = 2^n - 1$ (pensez à écrire le terme de droite en base 2 et à l'incrémenter). Peut-on généraliser ce résultat en base b ? Connaissez vous un autre moyen de prouver ce résultat ?

Notez que de ce fait, le poids du bit d'indice n (2^n) est strictement supérieur à la somme des poids de tous les bits d'indices allant de 0 à $n - 1$.

Exercice 9 : Quelle est la plus grande valeur possible pour un quartet, un octet, un seizet ou plus généralement un mot de n bits ?

DIVISION ENTIÈRE

Vous savez que dans une division entière (division Euclidienne), on divise un nombre entier n appelé dividende par un nombre entier d appelé diviseur pour obtenir les uniques deux nombres q (quotient) et r (reste) qui vérifient $n = q * d + r$ avec $r < d$. Par exemple, 1234 divisé par 10 donne un quotient de 123 et un reste de 4.

Exercice 10 : Quand on divise le nombre n =723489 par 10, quel quotient et reste obtient-on ? Quel est le rapport entre ce quotient, ce reste et l'écriture du nombre n en base 10 ?

Exercice 11 : On veut généraliser le résultat de l'exercice précédent. Montrez que, dans n'importe quelle base b, la division du nombre $n = (c_n \ldots c_2 c_1 c_0)_b$ par le diviseur b donne un quotient égal à $(c_n \ldots c_2 c_1)_b$ et un reste égal à c_0.

Exercice 12 : Utilisez le résultat de l'exercice précédent pour obtenir une méthode permettant de convertir n'importe quel nombre n dans une base b (en procédant par divisions successives). Pour information, un ordinateur doit utiliser un algorithme de ce genre chaque fois qu'il faut afficher un nombre à l'écran.

Exercice 13 : Utilisez la méthode définie dans l'exercice qui précède pour convertir en binaire les nombres ci-dessous (donnés en base 10).

1. 25
2. 109
3. 232
4. 511
5. 498
6. 1000
7. 1024
8. 9000

CONVERSION BINAIRE PAR SOUSTRACTION

Pour de petits nombres (typiquement un octet), il peut être plus simple de convertir en binaire en faisant uniquement des soustractions. Le but est de décomposer un nombre en une somme de puissances de 2. Voici l'algorithme à utiliser pour convertir un nombre compris entre 0 et 255 (octet).

```
foreach p ∈ {128, 64, 32, 16, 8, 4, 2, 1} par ordre décroissant do
    if n ≥ p then
        n ← n − p ;
        afficher 1;
    else
        afficher 0;
```

Exercice 14 : Utilisez la méthode par soustraction pour convertir en binaire les nombres ci-dessous (donnés en base 10).

1. 22

2. 37
3. 220
4. 101

PLUS FORT/PLUS FAIBLE POIDS

Dans un nombre $(c_n \dots c_2 c_1 c_0)_b$, le chiffre le plus à gauche (c_n) est appelé chiffre de plus fort poids (il doit être multiplié par le plus fort poids b^n). Le chiffre le plus à droite (c_0) est appelé chiffre de plus faible poids (il doit être multiplié par le plus faible poids b^0). De manière générale, le chiffre c_i a pour poids b^i. En anglais, on utilise les termes most significant/least significant (le plus significatif, le moins significatif).

Exercice 15 : Dans une base b quelconque, quelle méthode (algorithme) doit-on utiliser pour déterminer quel est le plus grand de 2 nombres (exemple : 0xfe2bd23 et 0xFE2AE34) ?

Exercice 16 : En utilisant la définition d'un nombre en base b et en factorisant le nombre b autant de fois que nécessaire, montrez que $(c_5 c_4 c_3 c_2 c_1 c_0)_b = (((((c_5 * b + c_4) * b) + c_3) * b + c_2) * b + c_1) * b + c_0$. Déduisez en une méthode (un algorithme) pour calculer facilement la valeur d'un nombre écrit en base b. Pour information, cet algorithme est utilisé chaque fois que vous tapez un nombre sur le clavier d'un ordinateur.

Exercice 17 : Utilisez l'algorithme précédent pour donner la valeur en base 10 des nombres ci-dessous.

1. 0777
2. 0xAF12
3. $(3423)_5$
4. 0b1011'1011
5. 0xBB
6. 0x10000
7. $(3210)_4$
8. 0x256

CONVERSIONS BINAIRE, OCTAL, HEXADÉCIMAL

Un chiffre en base 8 (octal) correspond à 3 bits. Un chiffre hexadécimal correspond à 4 bits. Les tableaux ci-dessous donnent la conversion entre octal/hexadécimal et binaire. Pour passer de l'octal/hexadécimal au binaire, il suffit de remplacer chaque chiffre par son équivalent binaire. Pour passer du binaire à l'octal, il suffit de regrouper les bits par groupes de 3 (en commençant par les bits de poids faibles, donc en commençant par la droite) et de convertir chaque groupe selon le tableau. Pour passer du binaire à l'hexadécimal, il suffit de regrouper les bits par groupes de 4 (en commençant par les bits de poids faibles, donc en commençant par la droite) et de convertir chaque groupe selon le tableau.

octal	binaire
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

hexadécimal	binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

hexadécimal	binaire
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Ces tableaux doivent être connus par cœur, ou retrouvés très rapidement. Par exemple, en connaissant les conversions de A, C, F et en utilisant les successeurs/prédecesseurs, on retrouve facilement les autres conversions.

Exercice 18 : Les nombres ci-dessous sont donnés en binaire, octal ou hexadécimal. Convertissez chaque nombre dans les 2 autres bases.

1. 0b0100'1011
2. 0x3A7C9
3. 0725
4. 0b1100'1010'1111'1101
5. 0xbd7ef
6. 07654321

KILO/KIBI, MÉGA/MÉBI, …

Les préfixes kilo, méga, giga, téra, etc., correspondent aux mêmes multiplicateurs que dans tous les autres domaines : des puissances de 10.

- 1 kilooctet (ko) = 10^3 octets = 1 000 octets
- 1 mégaoctet (Mo) = 10^6 octets = 1 000 ko
- 1 gigaoctet (Go) = 10^9 octets = 1 000 Mo
- 1 téraoctet (To) = 10^{12} octets = 1 000 Go
- 1 pétaoctet (Po) = 10^{15} octets = 1 000 To
- 1 exaoctet (Eo) = 10^{18} octets = 1 000 Po
- 1 zettaoctet (Zo) = 10^{21} octets = 1 000 Eo
- 1 yottaoctet (Yo) = 10^{24} octets = 1 000 Zo

Depuis 1998, il convient d'utiliser les préfixes kibi pour "kilo binaire", mébi pour "méga binaire", gibi pour "giga binaire", tébi pour "téra binaire", etc.

- 1 kibioctet (Kio) = 2^{10} octets = 1 024 octets
- 1 mébioctet (Mio) = 2^{20} octets = 1 024 Kio
- 1 gibioctet (Gio) = 2^{30} octets = 1 024 Mio
- 1 tébioctet (Tio) = 2^{40} octets = 1 024 Gio
- 1 pébioctet (Pio) = 2^{50} octets = 1 024 Tio
- 1 exbioctet (Eio) = 2^{60} octets = 1 024 Pio
- 1 zebioctet (Zio) = 2^{70} octets = 1 024 Eio
- 1 yobioctet (Yio) = 2^{80} octets = 1 024 Zio

En anglais, on utilise B pour byte (octet), et b pour bit.

La mémoire d'un ordinateur est normalement mesurée en Gio ou Mio (facteur 1024). La taille des disques durs est souvent donnée en Go ou To (facteur 1000). Le débit d'un réseau est donné en Mb/s ou Gb/s.

Exercice 19 : Une clef USB affiche une capacité de 32 GB. 5% de sa capacité est réservé pour remplacer des cellules défaillantes et n'est pas utilisable. Donnez le calcul pour obtenir la taille utilisable en GiB.

Exercice 20 : Un fichier de 1 Gio est transmis sur un réseau à 100 Mb/s. Pour être transmis sur le réseau, le fichier est découpé en paquets de 1460 octets, et il faut ajouter 40 octets à chaque paquet avant de le transmettre sur le réseau. Donnez le calcul pour déterminer combien de temps il faudra pour transmettre le fichier ?

Exercice 21 : Au primaire, vous avez appris à additionner, soustraire, multiplier et diviser des nombres en base 10. Les algorithmes que vous avez appris restent valables dans une base b quelconque. En binaire, la multiplication et la division sont même plus simples, simplement parce qu'on n'utilise que les chiffres 0 et 1, et multiplier par 0 ou 1 est une opération particulièrement facile ($0.x = 0$ et $1.x = x$).

Donnez les algorithmes (méthodes) à utiliser pour additionner, soustraire, multiplier et diviser des nombres en binaire.

Exercice 22 : Calculez le résultat de ces opérations.

1. 0b1010'1100 + 0b0011'0111
2. 0b0000'1111 + 0b1111'0001
3. 0b1001'0000 - 0b0010'0111
4. 0b1000'0000 - 0b0000'0001
5. 0x1010'0011 * 0b1'1001
6. 0x110'0100 * 0b101'1010
7. 0x110'0100 / 0b1010
8. 0x110'0100'0011 / 0b1110

Exercice 23 : Dans certains cas, on veut transmettre des données binaires (c'est à dire avec n'importe quelle valeur possible pour des octets) via des protocoles qui ne sont prévus que pour transmettre du texte et qui en plus peuvent n'accepter qu'un sous ensemble du jeu de caractères ASCII.

Dans ce cas, on peut écrire les données en base 64. On utilise dans l'ordre les symboles 'A' à 'Z', 'a' à 'z', '0' à '1', '+' et '/'. Chaque chiffre en base 64 correspond directement à 6 bits. Pour coder 3 octets (24 bits), il faut donc 4 chiffres en base 64. On regroupe les données par groupes de 3 octets et chaque groupe est codé par 4 chiffres. Le chiffre 'A' correspond à 0, 'a' à 26, '0' à 52, '+' à 62 et '/' à 63.

Sans entrer dans les détails, le symbole '=' est utilisé à la fin du codage quand le nombre d'octets n'est pas un multiple de 3. S'il n'y a aucun '=' à la fin, c'est que le dernier groupe contenait 3 octets. S'il y a un seul '=' à la fin, cela veut dire que le dernier groupe ne contenait en fait que 2 octets. S'il y a deux fois '=' à la fin, cela veut dire que le dernier groupe ne contenait en fait que 1 octet.

On a codé en base 64 des données et on a obtenu "QUJDREVG". Donnez en hexadécimal les octets qui ont été codés.