Systèmes d'exploitation

Généralités et numération

Définitions

- **Information** : ce qui peut être transmis par un signal ou une combinaison de signaux (message) selon un code commun.
- **Code** : système de symboles destinés à représenter une information.
- **Codage** : représenter des informations de “haut niveau” sur une machine.
- **Décodage** : l'opération inverse du codage
- **Transcodage** : Le passage d'un code (base) vers un(e) autre.

Généralités et numération

L'ordinateur ne traite que les informations numérisées (des 0 et des 1 sans signification particulière pour la machine) et seul l'interprétation qu'en fait l'utilisateur permet de lui donner un “sens”.

Codage de l'information

On code l'information pour la transmettre et la traiter et en général on utilise :

- des alphabets (grec, latin, russe …) comportant des lettres,
- des chiffres,
- des symboles,
- des pictogrammes (images, code de la route …)
- des idéogrammes (Egypte, Chine …)

ce qui nous permet d'écrire des textes comportant des mots, des nombres, des symboles, des formules, etc.

Code barre

Ce principe de codage, apparu au début des années 80, est largement utilisé sur le marché et surtout pour les produits en commerce. Par principe le marquage comportait des barres verticales ainsi que 13 chiffres :

- Le 1er chiffre désigne le pays d'origine : 3=Fr, 4=Ger, etc

…

- Les cinq suivants sont ceux du code « fabricant »,
- Les six autres sont ceux du code de l'article,
- Le dernier étant une clé de contrôle
- Les barres représentent le codage de ces chiffres sur 7 bits , à chaque chiffre est attribué un ensemble de 7 espace blancs ou noirs (0 ou 1).

Représentation d'un entier dans une base B

On choisit une base B et B caractères dont les valeurs sont comprises entre 0 et B-1

On écrit un nombre sous la forme d'une suite de caractères $N = a_n a_{n-1} a_{n-2} \dots a_1 a_0.$

Les valeurs de $a_n, a_{n-1}, a_{n-2}, \dots a_1 et a_0$ sont comprises entre 0 et $B-1$ on calcule la valeur de N par la formule :

$N = B^n \times a_n + B^{n-1} \times a_{n-1} + B^{n-2} \times a_{n-2} \dots B^1 \times a_1 +$

Code binaire

- On mémorise dans un ordinateur l'information à l'aide de circuits électroniques qui ont 2 états notés 0 et 1.
- C'est donc une représentation binaire.
- La base étant 2 nous utiliserons 2 caractères (le 0 et le 1) pour représenter les nombres.
- Par exemple : 11001010 est un nombre en base 2

Représentation dans la base 2

l'algorithme d'Euclide pour représenter 13 :

$$13 = 6 \times 2 + 1$$

$$6 = 3 \times 2 + 0$$

$$3 = 1 \times 2 + 1$$

$$1 = 0 \times 2 + 1$$

ce qui donne 1101 (l'ordre est donné du bas vers le haut du reste!!)

Calcul dans d'autres bases : 8 et 16

Toujours avec l'algorithme d'Euclide :

- codage 409 dans la base 8 :

$$409 = 51 \times 8 + 1$$

$$51 = 6 \times 8 + 3$$

$$6 = 0 \times 8 + 6$$

ce qui donne 631

Exemple de codage en Hexa

- Codage de 409 en base 16 :

$$525 = 32 \times 16 + 13$$

$$32 = 2 \times 16 + 0$$

$$2 = 0 \times 16 + 2$$

ce qui donne 20D

Décodage : revenir à la base 10

On utilise la décomposition vue précédemment :

$$N = B^n \times a_n + B^{n-1} \times a_{n-1} + \ldots + B^1 \times a_1 + B^0 \times a_0$$

Exemples :

- N_1 = 1100111 dans la base 2 –> N_1 =103 dans la base 10
- N_2= 5301 dans la base 8 –> N_2=2753 dans la base 10

Passage d'une base à une autre dans le cas

Prenons l'exemple suivant :

$13 = 2^3 + 2^2 + 0 \times 2^1 + 2^0$

1101 dans la base 2

$8^1 + 5 \times 8^0$

15 dans la base 8 (regrouper par trois et décoder dans la base 2 car $8 = 2^3$)

0D dans la base 16 (regrouper par quatre et décoder dans la base 2 car $16 = 2^4$)

Exemple d'utilisation

Retrouver le nombre écrit dans des bases différentes alors qu'il manque des chiffres dans l'écriture.

```
                    _  _  9          base 10

_  1  _  _   _  _  _  _  0  _          base 2

                    _  1  _          base 16

                 1  _  _  _          base 8
```

Code ASCII

- Le code ASCII est un code de 8 bits utilisant 2 caractères héxadécimaux : 7 bits + 1 bit de parité
- On code de 00 à $7F_H$ les caractères spéciaux, les chiffres, les lettres majuscules et minuscules.
- Le bit de parité est 0 ou 1 suivant que la parité est paire ou impaire (vu en calcul booléen).

Représentation des entiers signés

Signe + valeur absolue : en décimal cela donne +14, -781 dont 14 et 781 sont les valeurs absolues et + et – les signes.

- En binaire 0 et 1 vont remplacer + et –, respectivement en tête du nombre.
- Par exemple 1101 représente -5 et 0110 représente +6.
- Cette convention (abandonnée) ne permet pas de faire automatiquement des additions et des soustractions car il faut au préalable déterminer le signe du résultat par des comparaisons.

Représentation des entiers signés (suite)

Représentation en complément à 2 : dans cette représentation on tient compte du nombre de bits utilisés pour représenter les nombres.

- On partage l'intervalle de représentation en 2 parties, l'une servant à représenter les nombres positifs et l'autre les nombres négatifs.
- Le bit de poids le plus fort détermine le signe. 0 pour + et 1 pour -.
- Exemple : sur 8 bits l'intervalle de représentation est compris entre 00000000 et 11111111 c'est-à-dire 00 et FF en hexadécimal et 0 et 255 en décimal

Complément à 2, version math

Calcul : pour tout entier relatif

$$N = -a_{n-1} \times 2^{n-1} + \sum_{i=0}^{n-2} a_i \times 2^i$$

- si $a_{n-1} = 0$ alors N est positif ou nul
- si $a_{n-1} = 1$ alors N est négatif

Exemple : sur 3 bits $010_2 = 2_{10}$ et $100_2 = -2^2 = -4_{10}$

Complément à 2 version info

- Si le nombre est positif, notation normale
- Si le nombre est négatif
  - Alors on prend le complément à 1 (*Not* bit à bit) de son opposé
  - On ajoute 1 (on élimine ce qui déborde éventuellement)

Exemple: coder -4 en complément à deux sur 8 bits

- 4 s'écrit **0**0000100
- On inverse **1**1111011
- On ajoute 1 : **1**1111100

Opérations sur des entiers relatifs

Opposé d'un entier relatif : l'opposé de X noté op_X est tel que $X + op_X = X + (\bar{X} + 1) = 0$

La représentation en complément à 2 sur n bits permet de représenter les nombres sur l'intervalle : $[-2^{n-1}, 2^{n-1} - 1]$.

Opérations sur des entiers relatifs

Exemples

- en binaire $X = 000$ alors $\bar{X} = 111$ et $op_X = 000$
- en binaire $X = 001$ alors $\bar{X} = 110$ et $op_X = 111$
- en hexa : $X = 3B$ alors $\bar{X} = C4$ et $op_X = C5$

Exercice d'application

Sur 8 bits :

1. Placer sur un axe du plus petit au plus grand les nombres hexadécimaux suivants : 3B, 87, F5, A3, 6A, FF, 12

2. Calculer leur complément à 2 puis placer les sur le même axe.

Additions et soustractions sur des entiers p

Entiers positifs

12	0001 0010	47	0100 0111
+ 28	+0010 1000	-A5	-1010 0101
---	---	---	---
3A	0011 1010	**1** A2	**1** 1010 0010

EC	1110 1100	B3	1011 0011
+2B	+ 0010 1011	-75	- 0111 0101
---	---	---	---
1 17	**1** 0001 0111	3E	0011 1110

Additions et soustractions en C_2

Entiers en complément à 2

35	0011 0101	Il s'agit ici de faire la somme de
+ 2C	+0010 1100	nombres positifs 35 et 2C. On
------	----------	obtient 61 ce qui est correct
61	0110 0001	

AC	1010 1100	Ici on soustrait 54 à 2B ce qui donne
+ 2B	+0010 1011	-29 . Le résultat est correct.
------	----------	
D7	1101 0111	

Additions et soustractions en C_2 (suite)

```
   87        1000 0111
+  91      + 1001 0001
   ------    ----------
 1 18      1 0001 1000
```

La somme de 2 nombres négatifs est négative ce qui n'est pas le cas ici le résultat est erroné.

```
   75        0111 0101
+  49      + 0100 1001
   ------    ----------
   BE        1011 1110
```

La somme de 2 nombres positifs positive ce qui n'est pas le cas ici le résultat est erroné.

Notion de débordement

On dit qu'il y a débordement quand le résultat d'une opération arithmétique est non conforme aux règles de calcul.

Pour l'addition on a :

- 2 entiers + –> résultat +
- 2 entiers - –> résultat -
- 1 entier + et 1 entier - : résultat du signe du nombre de plus grande valeur absolue.

Notion de Drapeaux (Flags)

- Les flags sont des indicateurs qui prennent la valeur 0 ou 1 lors des opérations arithmétiques et logiques (addition, comparaison …). Les flags qui vont nous intéresser ici sont :
  - S flag sign (signe)
  - C flag carry (retenue)
  - V ou O flag overflow (débordement)
  - Z zero (zéro partout)
- Le flag C (carry) indique qu'il y a débordement dans le cas de nombres entiers positifs (sauf si le flag Z est positionné)
- Le flag O (overflow) indique qu'il y a débordement dans le cas de la représentation de nombres positifs et négatifs en complément à 2.

Comparaison des entiers

18	0001 1000	
- 91	-1001 0001	C = 1 S =1 Z =0 O = 1
------	----------	(18 + 6F → + 87)
1 87	1 1000 0111	entiers a < b C_2 a > b
BD	1011 1101	
- 7A	+0111 1010	C = 0 S = 0 Z = 0 O = 1
------	----------	(- (43+ 7A) → - BD)
43	0100 0011	entiers a > b C_2 a < b

Représentation des réels en virgule flottant

- Dans un mot de n bits on réserve :
  - Le signe
  - Une partie pour l'exposant
  - Une partie pour la mantisse
- Il existe différentes répartitions suivant les constructeurs et les normes adoptées.
- **Norme IEEE754 formats de base** :
  - Simple précision : 32 bits, 8 pour l'exposant et 24 pour la mantisse (23 + 1 pour le signe). De -3.4×10^{38} à 3.4×10^{38}.
  - Double précision 64 bits 11 pour l'exposant, 54 pour la mantisse (53 + 1 pour le signe). De -1.8×10^{308} à 1.8×10^{308}.

Représentation des réels en virgule flottant

```
   S      XXXX XXXX   YYY YYYY YYYY YYYY YYYY YYYY
signe     exposant                mantisse
1 bit     8 bits                  23 bits
```

Exemple : coder -513.317 en IEEE754 simple précision□□

- Calcul de la mantisse :
  - 513 –> 10 0000 0001 □
  - 0.317 –> 0.0101 0001 0010 0110 1110 1001 □
  - 513.317 –> 10 0000 0001 . 0101 0001 0010 0110 1110 1001

Représentation des réels en virgule flottant

- Normalisation : décalage de 10 bits à gauche du point décimal. 0.1000 0000 0101 0100 0100 10 01 $\times 2^{10}$.
- La mantisse est alors 000 0000 0101 0100 0100 1001 après suppression du bit 1 de tête.□□
- Calcul de l'exposant : 7E + 0A = 88
- Calcul du signe : négatif donc le bit signe est 1.□
- Résultat : 1 10001000 000 0000 0101 0100 0100 1001
- qui se lit C4 00 54 49 en hexa (par groupe de 4)

Représentation des réels en virgule flottant

- Décoder 43 53 11 26 en IEEE754 simple précision
  - Écriture en binaire : 0100 0011 0101 0011 0001 0001 0010 0110
  - Regroupement : 0 1000 0110 101 0011 0001 0001 0010 0110
  - Signe : +
  - Exposant : 86 – 7E = 8
  - Mantisse : on rajoute 1 en tête : 1101 0011 0001 0001 0010 0110
  - ce qui donne 13832486 en décimal
  - que l'on divise par 2^{24-8} = 65536
  - au final on obtient + 211.06698

Cas particuliers pour les formats IEEE 754

Les grands exposants tels FF sur 8 bits, 7FF sur 12 bits, 3FFF 14 bits indiquent des nombres :

```
INF (infini) si la mantisse est nulle,
```

```
NaN (not a number) si la mantisse est non nulle.
```

Si l'exposant est nul, la mantisse n'est plus normalisée.

On peut avoir des nombres plus petits que l'exposant minimum.