Ch Jaillet
Janv. 2019

Info0204

Eléments d'architecture des ordinateurs

Chapitre 2 : *Représentation des données*

1. Introduction, exemples
2. Représentation des entiers
   - entiers non signés / signés
   - opérations, gestion des débordements
3. Représentation des réels

Ch Jaillet
- URCA > UFR Sciences > Dept Maths, Méca, Info
- *christophe.jaillet@univ-reims.fr*
- *http://cosy.univ-reims.fr/~cjaillet*

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

1

1. Introduction, exemples

- Informatique
  - = traitement automatique ~~de l'information~~
  - = traitement automatique des données

- les données
  - texte
  - nombres
  - couleurs
  - images
  - *choses* !!

- représentations
  - suite de caractères => …
  - selon le type de nombre
  - …
  - combinaison de pixels => …
  - objets / structures / enregistrements
    => arrangement de champs => …

- tout est numérique
  - représentation des données élémentaires + combinaison
  - représentation informatique :
    - xxxxx IIIII V ~~IIII~~ 5
    - binaire !

- Théorème fondamental :
  - Sur *n* bits, on peut obtenir 2^n représentations différentes

  => … représenter 2^n valeurs différentes

2

2

1. Introduction, exemples

- caractères
  - n ≥ 6 => « *disons* 8… » : un octet
    - taille minimum d'un emplacement mémoire : un *mot*
      4 bits en 1971 (Intel 4004), puis 8 (octet) en avril 1972 (Intel 8008)
  - ASCII standard
    - *American Standard Code for Information Interchange*
    - norme ISO/CEI 646 : 128 caractères
      7 bits ; le 8ème bit sert à une vérification de parité
    - n° 32 : l'espace code 20 (32)
      '!' code 21
      …
      '0' .. '9' codes 30 à 39
      'A' .. 'Z' codes 41 à 5A (numéros 65 +)
      'a' .. 'z' codes 61 à 7A (décalage : 32)
    - Exercice : Codez « Bonjour !»
      Décodez 436F6F6C203F
  - ASCII étendu
  - …
  - iso-latin-xx iso-8859-1 (Fr)
  - Unicode (utilisé en Java)
    - codage universel, sur 16 bits

	000	001	002	003	004	005	006	007
0	NUL	DLE	SP	0	@	P	‘	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	−	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

1. Introduction, exemples

- couleurs
  - noir et blanc vrai/faux 1 bit ?
    niveaux de gris 32 niveaux ? 8 bits : paliers 0, 8, …
  - 256 couleurs (noir=0, …)
    palette de couleurs
    systèmes à table indexée : palette dans l'image
  - couleurs **RVB** (ou *RGB*)
    - RVB dégradé : 5 bits R, 6 bits V, 5 bits B
      - couleurs 16 bits : 65 536 couleurs
    - couleurs vraies : 1 octet / composante (0 .. 255)
      - couleurs 24 bits : 16,7 millions de couleurs
  - autres :
    - CMJN Cyan, Magenta, Jaune, Noir
    - TSL (Teinte, Saturation, Luminosité) / TSV (…)
    - Y'UV (luma, chrominance)

- images
  - -
  - GIF (*Graphics Interchange Format*)
    2 à 256 couleurs
  - RVB
    - écrans VESA
    - vectoriel
      BMP (bitmap)
      TGA 2D (PAL, NTSC)
  - autres :
    - impression quadri
    - SVG
    - video analogique
      PAL / SECAM

- JPEG
  - *Joint Photographic Experts Group*
  - vectoriel compressé par zone (quad-tree)

- MPEG / M-JPEG
  - *Moving Pictures Experts Group*
  - basé sur la redondance temporelle

1. Introduction, exemples

- **entiers**
  - Taille fixe
    - Quelle que soit la taille choisie, le nombre de valeurs différentes est limité
    - On ne peut pas représenter toutes les valeurs entières
  - Différentes longueurs selon le nb de valeurs souhaitées (nb de valeurs représentables)

nb octets	1	2	4	8	…
nb val. ≠	256	65 536	4 294 967 296	18 446 744 073 709 551 616	…

- <u>exemple :</u> tailles, en cm
  - 0,45 m … 2,40 m => moins de 200 val => 8 bits
  - codes : *100*taille - 45* => 0 pour 0,45 ; 1 pour 0,46 ; … ; xxx pour 2,40
    plus simple ? 0 pour 0,00 ; 1 pour 0,01 ; …

- entiers < 0 ?
  - entiers entre -15 et 150 ?
    - …

- Problème : les calculs !
  - comparaison : ok si la repr. conserve l'ordre
  - opérations : addition, soustraction, …

1. Introduction, exemples

- **réels**
  - Taille fixe : …
  - Au sens mathématique,
    - l'ensemble des entiers est infini
    - l'ensemble des réels est infini
    - l'ensemble des réels ente 0 et 1 (par exemple) est infini
  - **Représentation dégradante :**
    - un réel, représenté informatiquement, est a priori faux

- <u>Exemples :</u>
  - 1,625 = 1 + 1/2 + 1/8 = $(1{,}101)_2$
  - 0,6 = 1/2 + 1/16 + . . . (?)
    - en fait 0,6 = $(0{,}1001\underline{1001}....)_2$

- Idée de représentation :
  <u>*virgule flottante*</u>
  - base 10 : $-518{,}29 = -5{,}1829.10^2$
  - base 2 : $13 = (1101)_2 = (1{,}101)_2.2^3$

Ch Jaillet

Info0201

Introduction à la programmation orientée objet

Chapitre 2 : *Représentation des données*

7

2. Représentation des entiers

- Taille de la représentation ?
  - 2, 4, … , 256, … valeurs différentes (représentations)
- Choix de la taille :
  - selon le nombres de valeurs différentes ont on a besoin
- ? signé / non signé

8

8

2. Représentation des entiers
a. entiers non signés, ici sur un octet

- [choix] un octet => valeurs de 0 à 255 exclusivement
- codage
  - $220 = \ldots = (11011100)_2 \leftrightarrow [11011100]_2$
  - $13 = (1101)_2 \leftrightarrow [00001101]_2$
- décodage
  - $[10011011]_2 \leftrightarrow (10011011)_2 = \ldots = 145$
  - $[00101110]_2 \leftrightarrow (101110)_2 = \ldots = 46$

valeurs, écritures (mathématiques)
≠
représentation (informatique)

- NB : autres écritures de la représentation
  - $13 = (1101)_2 \leftrightarrow [00001101]_2 = [0d]_{16}$
  - $[233]_8 = [10011011]_2 \leftrightarrow (10011011)_2 = \ldots = 145$
    $[233]_8 \leftrightarrow (233)_8 = \ldots = 145$

9

9

Ch Jaillet

Info0201

Introduction à la programmation orientée objet

Chapitre 2 : *Représentation des données*

1. Introduction, exemples
2. Représentation des entiers
   a. entiers non signés
   b. entiers signés
   c. opérations, gestion des débordements
3. Représentation des réels

10

2. Représentation des entiers
b. entiers signés, ici sur un octet

- Introduction :
  - 1 octet : 256 codes : *a priori* 128 valeurs positives
    128 négatives
  - représentation naturelle des valeurs positives

- Historique :
  - 76 => 0 1 0 0 1 1 0 0
  - -76 ?
    - ? 1 1 0 0 1 1 0 0 *signe et valeur absolue* (SVA)
    - ? 1 0 1 1 0 0 1 1 *complément à 1* (c1)
    - ? 1 0 1 1 0 1 0 0 *…* " *complément à 2* " (c2)
      en fait " *complément à 1, +1* "

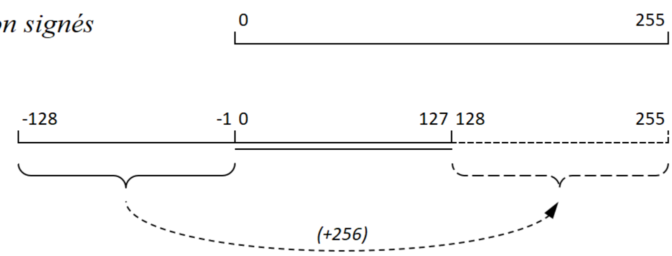
bit de signe

2. Représentation des entiers
b. entiers signés, ici sur un octet (2)

- Représentation en complément à 2 (sur un octet)
  - valeurs entre -128 et 127 (exclusivement)
  - codage naturel des valeurs positives
  - "translation" des valeurs négatives

entiers non signés

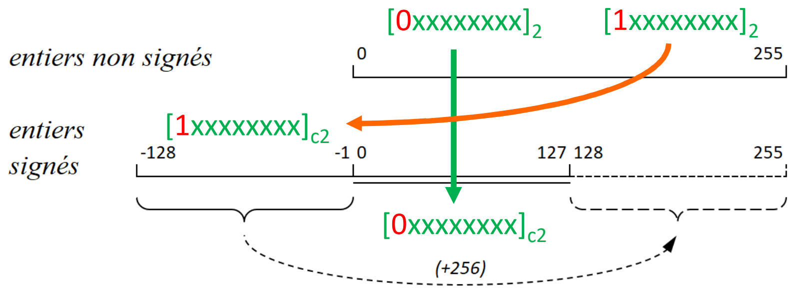
entiers signés



2. Représentation des entiers
b. entiers signés, ici sur un octet (3)

Ch Jaillet (URCA)
Info0204 – Ch. 2
Repr. données

- Représentation en complément à 2 (sur un octet)
  - notion de bit de signe



  - positif *ssi* $[0xxxxxxx]_{c2}$
    négatif *ssi* $[1xxxxxxx]_{c2}$ } **bit de signe !** (bit de *poids fort*)
  - Attention : notation hexa. => on ne voit pas le bit de signe
    => écrire la représentation en binaire

13

13

2. Représentation des entiers
b. entiers signés, ici sur un octet (4)

Ch Jaillet (URCA)
Info0204 – Ch. 2
Repr. données

- codage
  - $13 = (1101)_2 \leftrightarrow [00001101]_{c2}$
  - -116 ?
    - $-116 + 256 = 140 = \ldots = (10001100)_2 \leftrightarrow [10001100]_2$ *définition*
      donc $-116 \leftrightarrow [10001100]_{c2}$ *calcul indirect*
    - ou $-116 = -128 + 12 = -128 + 8 + 4 \leftrightarrow [10001100]_{c2}$ *direct*
- décodage
  - $[00101110]_{c2} \leftrightarrow (101110)_2 = \ldots = 46$
  - $[10011011]_{c2}$?
    - $(10011011)_2 = 128 + 16 + 8 +2 + 1 = 155$ *définition*
      donc $[10011011]_{c2} \leftrightarrow 155 - 256 = -101$ *calcul indirect*
    - ou $[10011011]_{c2} \leftrightarrow -128 + 16 + 8 +2 + 1 = -101$ *direct*
- **Attention :**
  - codage et décodages *naturels* pour les valeurs positives !!

14

2. Représentation des entiers
b. entiers signés, ici sur un octet (5)

- Valeurs négatives : les différentes techniques
  1. **Décodage :** $[11010011]_{c2}$
     a. définition $(11010011)_2 \leftrightarrow 128+64+16+2+1 = 211$
        or le bit de signe est à 1 donc la valeur est négative :
        $[11010011]_{c2} \leftrightarrow 211-256 = -45$
     b. direct $[11010011]_{c2} \leftrightarrow (128+64+16+2+1)-256$ car le bit de signe est à 1
        $= 211-256 = -45$
     c. (équivalent) $[11010011]_{c2} \leftrightarrow -128+64+16+2+1$ car le bit de signe est à 1
        $= -128+83 = -45$
  2. **Codage :** -24
     a. définition -24 < 0 donc on considère -24+256=232
        or 232 = … = $(11101000)_2$ donc $-24 \leftrightarrow [11101000]_{c2}$
     b. direct $-24 = -128+104 = -128+64+40 = -128+64+32+8$
        $\leftrightarrow [11101000]_{c2}$
     c. méthode du complément à 1 : négation bit à bit, puis ajouter 1
        $24 = 16+8 = (11000)_2 \leftrightarrow [00011000]_{c2}$
        donc $\sim 24 \leftrightarrow [11100111]_{c2}$
        et $-24 \leftrightarrow [11101000]_{c2}$

$$\begin{array}{r} 11100111 \\ +\quad\quad 1 \\ \hline 11101000 \end{array}$$

15

15

2. Représentation des entiers
b. entiers signés, ici sur un octet (6)

- Valeurs négatives : les différentes techniques
  1. **Décodage : …**
  2. **Codage :** -24
     a. définition …
     b. direct …
     c. méthode du complément à 1 : négation bit à bit, puis ajouter 1
        $24 = 16+8 = (11000)_2 \leftrightarrow [00011000]_{c2}$
        donc $\sim 24 \leftrightarrow [11100111]_{c2}$
        et $-24 \leftrightarrow [11101000]_{c2}$

$$\begin{array}{r} 11100111 \\ +\quad\quad 1 \\ \hline 11101000 \end{array}$$

     d. **technique** du complément à 2 :
        - $24 = (11000)_2 \leftrightarrow [0001|1000]_{c2}$
          donc $-24 \leftrightarrow [0001|1000]_{c2}$, par la technique du complément à 2
        - En partant de la droite,
          - conserver tous les bits jusqu'au 1[er] '1' (compris)
          - puis inverser tous les suivants
        - -22 ? -122 ?

16

16

Ch Jaillet

Info0201

Introduction à la programmation orientée objet

Chapitre 2 : *Représentation des données*

17

---

2. Représentation des entiers
c. opérations, gestion des débordements

- Addition

```
                     1   1 1 1
 |0,1,0,1,1,1,1,0|  ---->    1 0 1 1 1 1 0
 |1,1,0,1,0,1,0,1|        +  1 1 0 1 0 1 0 1
                          -----------------
(1) |0,0,1,1,0,0,1,1| <-- 1 0 0 1 1 0 0 1 1
```

- **Vérification**, ici pour des entiers non signés (sur 8 bits) :
  - $[01011110]_2 \leftrightarrow (1011110)_2 = \ldots = 94$
  - $[11010101]_2 \leftrightarrow (11010101)_2 = \ldots = 213$
  - $[00110011]_2 \leftrightarrow (110011)_2 = \ldots = 51$
  - $94 + 213 = 305 \equiv 51 \pmod{256}$
    donc l'opération est juste (à 256 près)

18

2. Représentation des entiers
c. opérations, gestion des débordements

- **Soustraction**

```
            |0|1|0|1|1|1|1|0|   ----->   1 0 1 0 1 0 1 0 1 1 1 1 1 1 0
            |1|1|0|1|0|1|0|1|                 -   1 1 0 1 0 1 0 1
                                         ... 1 1                1
                                         ----------------------------
(...111)    |1|0|0|0|1|0|0|1|   <-----   ... 1 1 1 0 0 0 1 0 0 1
```

- **Vérification,**
  ici pour des entiers signés repr. en cplt à 2 sur 8 bits :
  - $[01011110]_2 \leftrightarrow (1011110)_2 = . . . = 94$
  - $[11010101]_2 \leftrightarrow -128+64+16+4+1 = -128+85 = -43$
  - $[10001001]_2 \leftrightarrow -128+8+1 = -119$
  - $94 – (-43) = 94 + 43 = 137 \equiv -119 \pmod{256}$
    donc l'opération est juste (à 256 près)

19

19

2. Représentation des entiers
c. opérations, gestion des débordements

- Multiplication

```
|0|1|0|1|1|1|1|0|   ----->              1 0 1 1 1 1 0
|1|1|0|1|0|1|0|1|               x     1 1 0 1 0 1 0 1
                       ------------------------------
                        x  x xx xx xxx xx xx x  x  x
                                      | 1 0 1 1 1 1 0
                                    1 | 0 1 1 1 1 0 .
                                1 0 1 | 1 1 1 0 .
                            1 0 1 1 1 | 1 0 .
                          1 0 1 1 1 1 | 0
                       ------------------------------
|0|0|1|1|0|1|1|0|      1 0 0 1 1 1 0  | 0 0 1 1 0 1 1 0
```

20

20

Ch Jaillet

Info0201

Introduction à la programmation orientée objet

Chapitre 2 : *Représentation des données*

21

Ch Jaillet (URCA)
Info0204 – Ch. 2
Repr. données

2. Représentation des réels
a. principe

- Idée :
  - 234,56 = 2,3456 . 10^2
  - -13,75 = - (8 + 4 + 1 + ½ + ¼) = - $(1101,11)_2$ = - $(1,10111)_2 . 2^3$
    = - $(0,110111)_2 . 2^4$
- propriété (base B) :
  - si $x = (0,b_1b_2 \ldots b_k)_B$ alors $B^*x = (b_1,b_2 \ldots b_k)_B = b_1 + (0,b_2 \ldots b_k)_B$
  - => extraire un chiffre ? multiplier par B et garder la partie entière
- **algorithme (base 2) :**
  - séparer la partie entière et la partie fractionnaire
    => reste = partie fractionnaire
    - doubler ; garder la partie entière ; conserver le reste
    - doubler ; garder la partie entière ; conserver le reste
    - …
  - écriture à virgule en base 2 (*diadique*)
  - décaler la virgule (virgule *flottante*)
- autres exemples : 49,7 ; - 10^{-2}

22

22

2. Représentation des réels
a. principe (2)

- Principe de représentation :
  - signe + mantisse + exposant
  - mantisse : partie après la virgule (0 avant la virgule)
    - toujours un 1 après la virgule => on peut l'ignorer
  - représentation :

± *xxxxx* *yyyyyyyyyyyyyyyyyy*

signe (1 bit) :
. 0 pour "+"
. 1 pour "-"

exposant

mantisse
. après le 1er 1 (*bit caché*)
. tronquée

  - le type de la représentation conditionne :
    - la taille (en nombre de bits) de la mantisse ; son codage
    - la taille et codage de l'exposant

2. Représentation des réels
b. norme IEEE754

- La norme IEEE754 :
  - signe + mantisse + exposant
  - mantisse m, tronquée, sur la base de $(0{,}1xxxxx)_2$: ignorer le 1er 1
  - exposant *biaisé* e' (pas l'exposant e codé en complément à 2)
    - **e' = e + *biais*** => positif => codé comme un entier non signé
    - exemple (3 bits) :
      - 8 valeurs possibles : -4 à 3
      - on en garde 7 : -3 à 3 (prévoir pour les conventions)
      - ajouter 3 pour se ramener à un entier positif
      - coder comme un entier non signé [sur 3 bits]
  - valeur 0 : exposant et mantisse à 0 + autres conventions

- deux précisions possibles :
  - simple précision = 32 bits : 1+8+23 $x = (-1)^s \times (0{,}1m)_2 \times 2^{e'-127}$
    - amplitude : -2^{127} à 2^{127} ; précision $2^{-127-24} = 2^{-151} \approx 10^{-45}$
  - double précision = 64 bits : 1+11+52

2. Représentation des réels
b. norme IEEE754 (2)

- La norme IEEE754 : exemples
  1. Donner la représentation en simple précision de x = 5,41
     - … => X = $[41\ 26\ C2\ C2]_{c1,16}$
  2. Donner une val. appr. de la val. y représentée par Y = $[A0\ D0\ 00\ 00]_{c1,16}$
     - Y = $[10100000\ 110100000\ 00000000\ 00000000]_{c1,2}$
     - négatif (bit de poids fort à 1)
     - e' = $(1000001)_2 = 65$ donc e = 65 - 127 = -62
     - M = [1010000 00000000 00000000] donc $(0{,}1m)_2 = (0{,}1101)_2$
     - $y = -(0{,}1101)_2 \,.\, 2^{-62} = -(1101)_2 \,.\, 2^{-65} = -13 \,.\, 2^{-65}$
       $= -13 \times 32 \,.\, 2^{-70} = -416 \,.\, (2^{10})^{-7}$
       $\approx -416. (10^3)^{-7} = -416 \,.\, 10^{-21} \approx -4 \,.\, 10^{-19}$

2. Représentation des réels
c. opérations

- Opérations
  - Conversions implicites
    - deux entiers => calcul entre entiers
    - deux réels => calcul entre réels
    - un entier et un réel => 2 réels : l'entier est converti en réel (codage !)
  - Recherche des valeurs nulles
  - Multiplication
    - les exposants s'ajoutent : attention au biais
    - les mantisses se multiplient : attention au 1 implicite
  - Division
    - …
  - Addition, soustraction
    - normalisation (exposant le plus haut)
    - calcul, puis normalisation du résultat

2. Représentation des réels
c. opérations (2)

- Opérations (suite)
  - attention aux arrondis
    - les opérations peuvent ne pas avoir les mêmes propriétés
      - $(1 + 2^{-24}) = 1$ donc $(1 + 2^{-24}) + 2^{-24} = 1$
        alors que $1 + (2^{-24} + 2^{-24}) = 1 + 2^{-23}$
    - *arithmétique des processeurs*
  - Exercice :
    - Effectuez-en le produit et la différence (calculs binaires)
    - Donnez les valeurs exactes de ces quantités
    - Vérifiez les valeurs obtenues pour la somme et le produit