JAPLA 研究会資料 2001/4/28

Ｊの値の内部表現について
－浮動小数点数を中心に－

西川 利男

Toshio.Nishikawa@kiu.ne.jp

このところＪで数論の問題を扱い，その計算にＪが極めて強力であることを報告してきた．同時に，QBASIC，UBASIC などの言語，DERIVE，Maple Ⅴなどの数式処理をも用いて比較している．一方，伊理先生の「数値計算の常識」（p.2）[1] にもあるように，0.01 を 10000 回，加えると 100.003 になる，といったことが起きる．Ｊでは起きないが，QBASIC で単精度ではこの通りになる．これらを詳細に検討するには浮動小数点数の内部表現について知ることが必要である．

Ｊの値の内部表現についてはシステム・ユーティリティ関数 3!:x により知ることができるが，パソコンの一般の方式と異なる個所もあり，このままではわかりにくい．このためいくつかのプログラムをつくり簡単にわかるようにした．これを用いて浮動小数点数の原理とＪの方式との対応について解説する．

１．Ｊの値のデータ型と内部表現のユーティリティ関数

3!:0　値のデータ型 …… 返された数によりデータ型が示される

1	論理数（boolean）
2	文字列（literal）
4	整数（integer）
8	浮動小数点数（float）
16	複素数（complex）
32	ボックス化された値（boxed）

64　拡張整数（extended integer）

128　有理数つまり分数（rational/fraction）

3!:1　内部表現の２進表示

3!:3　内部表現の16進表示（文字として）

3!:4　整数の変換

3!:5　浮動小数点数の変換

・データ型，表示の例

```
   3!:0 (1)
1
   3!:0 ('abc')
2
   3!:0 (12)
4
   3!:0 (3.14)
8
   3!:0 (3j4)
16
   3!:0 (<2 3 4)
32
   3!:0 (12x)
64
   3!:0 (2r3)
128
```

２．Ｊの内部表現の形式

ユーティリティ関数（3!:3）で表示される内部表現は一般的に次のような形式になっている．いろいろなデータ型の値に対して内部表現を見てみよう．

単に１と入力したときは，Ｊでは論理数（真）と解される．

```
   3!:3 (1)
01000000  …… 論理数
00000000  …… 区切り
01000000  …… 要素数１
00000000  …… ０次元（アトム）
```

01000000 …… 値

文字列はつぎのようになる.

3!:3 (’abc’)

02000000 …… 文字列

00000000 …… 区切り

03000000 …… 要素の総数３

01000000 …… １行

03000000 …… ３列

61626300 …… 値，文字列 ’abc’

もっとも簡単な１つの整数ではつぎのようになる.

3!:3 (12)

04000000 …… 整数

00000000 …… 区切り

01000000 …… 要素数１

00000000 …… ０次元（アトム）

0c000000 …… 値（16 進）10 進 12 に相当

つぎは，５つの整数から成る１次元のベクトルである.

3!:3 (4, 7, 2, 10, 3)

04000000 …… 整数

00000000 …… 区切り

05000000 …… 要素の総数５

01000000 …… １次元（ベクトル）

05000000 …… 要素数

04000000 …… 値

07000000 …… 値

02000000 …… 値

0a000000 …… 値 （16進）10進10に相当

03000000 …… 値

さらに，２行３列の配列である．

```
   3!:3 (>: i. 2 3)
04000000
00000000
06000000 …… 要素の総数６
02000000 …… ２次元（配列）
02000000 …… ２行
03000000 …… ３列
01000000 …… 値
02000000 …… 値
03000000 …… 値
04000000 …… 値
05000000 …… 値
06000000 …… 値
```

浮動小数点数について形式は明らかだが，値の表現は複雑であり，後にくわしく述べる．

```
   3!:3 (0.1)
08000000 …… 浮動小数点数
00000000 …… 区切り
01000000 …… 要素数１
00000000 …… ０次元（アトム）
9a999999 …… 値
9999b93f …… 値
```

３．浮動小数点表示の考え方とその仕様

浮動小数点表示の原理はいわゆる科学的表記と同じである．つまり例えば6.023×10^{23}なる数は

仮数（Mantissa）6.023

基底（Base）　10

指数（Exponent）23

であるが，コンピュータ内部の値としてはこのままでは扱うことは出来ない．

すなわち，コンピュータの浮動小数点表示としてはつぎのようにする．

・基底は通常は２進である．（メインフレームで２進化１０進というのがあったが）

・値全体の正，負をあらわすため，符号ビットを置く．

・指数については正，負の数とする代わりに，適当な数を加え（バイアス（bias)方式），すべて正の整数で扱う．

・仮数は０と１との間の値となるように調整する．

・小数点を最左端に置き，小数点のすぐ下の桁が０にならないように，正規化と呼ぶ調整をおこなう．

・通常は小数点を省略した暗黙の小数点（Implied binary point)なる手法を使う．

また，コンピュータの機種，ＯＳ（ＩＢＭ，ＤＥＣ－ＶＡＸなど）によりいくつかの仕様があるが，パソコンでは標準的にＩＥＥＥの方式が使われる．

ＩＥＥＥ方式の浮動小数点表示では次のようになっている．

・ＩＥＥＥ単精度

　基底　２進（１６進）

　指数　８ビット（63 - bias)

　仮数　２３ビット

・ＩＥＥＥ倍長精度

　基底　２進（１６進）

　指数　１１ビット（127 - bias)

　仮数　５２ビット

４．Ｊにおける浮動小数点数の表示

Ｊでは先に見たように，浮動小数点数は内部では６４ビットで表現される．

そしてつぎのように割り当てられている．

64 bit = 1 bit(Sign) + 11 bit(Exponent) + 52 bit(Mantissa)

例として，57.8125 という１０進小数をとりあげてみよう．これに対するＪの内部表現はつぎのようになる．

なお，ここで使用した関数定義のリスティングは付録２にあげてある．

```
   3!:3 (57.8125)
08000000
00000000
01000000
00000000
00000000
00e84c40
```

ここで値の部分だけを取り出すには，(2) 3!:5 (57.8125) とすればよい．しかしこれはバイナリ値でありそのままでは見ることはできない．かつ，Ｊではマイクロプロッセサでよく行われるバイト逆順になっているので，これらを考慮した関数 fl で１６進数として表示するとつぎのようになる．

```
   fl 57.8125
40 4C E8 00 00 00 00 00
```

しかし，これでも初めの１０進小数との対応は明らかではないだろう．以下，浮動小数点表示の原理に基づき，解明していく．

まず，１０進小数を２進分解する．

57.8125 = 32 + 16 + 8 + 1 + .5 + .25 + .0625

= 2^5 + 2^4 + 2^3 + 2^0 + 2^_1 + 2^_2 + 2^_4

したがって２進表示ではつぎのようになる．

１１１００１．１１０１

この後，先に述べたコンピュータのための変換調整を行う．

正規化 …… 仮数が０と１との間になるように変換する．

．１１１００１１１０１＊2^6

指数のバイアス表示への変換 …… 1022 を加える

6 + 1022 = 1028 (=4 0 4 hex)

Implied Floating Point 調整 …… ２進小数第１位のビットを落とす．

．１１００ １１１０ １０００ （＝ Ｃ Ｅ ８ hex)

符号ビット …… ０（正）を指数部の最初に加える．

このようにして，６４ビットの浮動小数点，内部表示の値が組み立てられた．

40 4C E8 00 00 00 00 00

逆に与えられたＪの１０進小数をの内部表示を解読して２進表示する関数 flb を用いれば，つぎのように示される．

```
flb 57.8125
S    Exponent       Mantissa
-  -------------  ----------------------------
0  100 0000 0100  .11100  1110 1000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
```

なお，いろいろな浮動小数点数の値の内部表現の例を付録１にあげてあるので，そのよ

うすを参照，比較されたい．

文献

[1] 伊理正夫，藤野和建，「数値計算の常識」，共立出版 (1987).

[2] Anthony Ralston, Edwin D. Reilly, ″Encyclopedia of Computer Science″, 3rd ed. van Nostrand, p.81-87 (1993).

[3] Microsoft,「QuickBASIC Statement & Function Reference」付録Ｃ．データの記録形式，p.447-453 (1989).

付録１．　いろいろな浮動小数点数の内部表現の例

```
   flb 1.0
S    Exponent       Mantissa
-  ------------  ---------------------------
0  011 1111 1111  .10000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb 2.0
S    Exponent       Mantissa
-  ------------  ---------------------------
0  100 0000 0000  .10000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb 3.0
S    Exponent       Mantissa
-  ------------  ---------------------------
0  100 0000 0000  .11000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb 4.0
S    Exponent       Mantissa
-  ------------  ---------------------------
0  100 0000 0001  .10000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb 5.0
S    Exponent       Mantissa
-  ------------  ---------------------------
0  100 0000 0001  .10100  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb _1.0
```

```
S    Exponent       Mantissa
-  -------------  ----------------------------
1  011 1111 1111  .10000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb _2.0
S    Exponent       Mantissa
-  -------------  ----------------------------
1  100 0000 0000  .10000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb 0.5
S    Exponent       Mantissa
-  -------------  ----------------------------
0  011 1111 1110  .10000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb 0.25
S    Exponent       Mantissa
-  -------------  ----------------------------
0  011 1111 1101  .10000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb 0.125
S    Exponent       Mantissa
-  -------------  ----------------------------
0  011 1111 1100  .10000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
   flb 0.1
S    Exponent       Mantissa
-  -------------  ----------------------------
```

```
0  011 1111 1011  .11001  1001 1001  1001 1001  1001 1001  1001 1001  1001 1001
1001 1010
   flb 0.2
S    Exponent       Mantissa
-  -------------  ------------------------------
0  011 1111 1100  .11001  1001 1001  1001 1001  1001 1001  1001 1001  1001 1001
1001 1010
   flb 1.5
S    Exponent       Mantissa
-  -------------  ------------------------------
0  011 1111 1111  .11000  0000 0000  0000 0000  0000 0000  0000 0000  0000 0000
0000 0000
```

付録２．　Ｊの内部表現，表示の関数定義

```
val=: a. & i.
chr=: val ^:_1
NB. dump values in hex
hdump=: }:@,@(,"1' '"_)@(hfd"0@val f.)

NB. dec_to_bin in byte
NB.  usage: d2b 65 -> 0100 0001
d_to_b =: (8#2)&#:
d2b =: ,@((,"1' '"_)@((2,4)&$@((' '&~:)#])@(":@d_to_b))))
NB. dump values in binary
bdump=: }:@,@(,"1' '"_)@(d2b"0@val f.)


NB. J arithmetic value - internal expression
NB.   by Toshio Nishikawa 2001/3/9

NB. value type check
chk_type =: 3 : 0
select. 3!:0 y.
  case. 1 do. 'boolean'
  case. 2 do. 'literal'
  case. 4 do. 'integer'
  case. 8 do. 'floating'
  case. 16 do. 'complex'
  case. 32 do. 'boxed'
```

```
  case. 64 do. 'extented integer'
  case. 128 do. 'rational(fraction)'
end.
)




NB. J-floating value - internal expression
fl =: 3 : 'hdump |. (2) 3!:5 y.'   NB. in hex
NB. 1-sign, 11-exponent, (1-implied bp), 52-mantissa
NB.   conformed to IEEE double precision format
flb =: 3 : 0                       NB. in binary
wr 'S    Exponent       Mantissa'
wr '-  -------------  ----------------------------'
bz =. bdump |. (2) 3!:5 y.
bexp =: 15{.bz
bman =: 16}.bz
({.bz),'  ', (}.(10&{. ,11&}.)bexp),'  ', '.1',bman
)

fc =: 3 : 'bdump 5{. |. (2) 3!:5 y.'
flc =: 3 : 0
(12.9 ":y.),': ', fc y.
)
```

```
NB. J-integer value - internal expression

int =: 3 : 'hdump |. (2) 3!:4 y.'  NB. in hex

intb =: 3 : 'bdump |. (2) 3!:4 y.' NB. in binary
```