＜論文＞

32ビットFORTHによる多数桁数演算：完成版

貞　方　一　也*

抄　録：多数桁数とは、任意の桁数の整数部と任意の桁数の小数部を持つ数のことである。今回多数桁数を広範囲に取り扱うことのできるFORTH32の新版（第2.8版）を完成した。このFORTH32第2.8版では、多数桁数のスタック演算と四則演算に加えて、各種の多数桁数関数演算が可能である。

FORTH32の多数桁数を用いてEulerの定数を小数点以下1000桁まで求める計算を行った。この計算のプログラムは巨大数を用いる計算のプログラムよりも明解であるが、計算時間は4倍位の長さとなる。カオスに関するMayの方程式 y = ax（1－x）でm周期解（m＝4, 8, 16, …, 2＾24）の起きるパラメーターaの値を求める計算も行った。この計算のプログラムは、以前の浮動小数を用いる計算より明解であり、精度を自由に設定でき、しかも実行も高速である。

キーワード：32ビットFORTH、 FORTH32、巨大数、多数桁数、Eulerの定数、Mayの方程式

1．はじめに

我々が開発を続けている32ビットFORTH（FORTH32と呼ぶ）は、浮動小数点演算ができる[1]。この浮動小数は、96ビットであり、そのうち仮数部は80ビットを占め、10進で23桁の精度を持っている。

近年数値計算の分野で巨大数が関心を持たれている。巨大数とは、何千桁から何万桁、…、何億万桁にもなる巨大な整数のことである。例えば、1000！（1000の階乗）、[π×（10の千乗）] の整数部分などである。このような巨大数はすでにFORTH32に取り入れられている[2]。FORTH32の巨大数は10万進法で表され、その1つの桁には、00000から99999までの数が入る。例えば、123456789012345は3桁の巨大数となる。FORTH32では、巨大数のスタック演算や四則演算が可能である。

多数の桁数の整数部と小数部を持つ数は多数桁数と呼ばれ、FORTH32ver.2.7で導入された[3]。そこでは多数桁数の関数演算が取り上げられたが肝心のスタック演算や四則演算ができなかった。今回、多数桁数のスタック演算、四則演算などを取り入れ、関数演算その他を整備し、多数桁数を容易に扱えるFORTH32を作成することができた。このようなFORTH32を第2.8版のFORTH32と呼ぶ。

以下2～6では、第2.8版のFORTH32について説明する。また、7では多数桁数によるEulerの定数γの計算について、8ではカオスにおけるMayの方程式の周期解に関する計算について述べる。

2．FORTH32第2.8版

多数桁数の十分な取り扱いが可能となったFORTH32第2.8版であるがその陣容、起動方法は旧版と違いはない。

2.1　FORTH32の陣容と起動

FORTH32には次の2種類がある。

（1）FORTH32W

WindowsのDOS窓で動くプログラムで、次のように入力して起動する。

c>forth32w

FORTH32Wは、60000Hバイトの DOSメモリーを占有するが、すばやく起動され、軽快に動作する。

（2）FORTHEX

WindowsのDOS窓で動くプログラムで、自作の補助プログラムであるEXE32を使って次のように入力して起動する。

c>exe32 forthex

FORTHEXは、1000000Hバイトの拡張メモリーを使用

＊人間基礎科学講座

し、起動には時間がかかるが、巨大な桁数の計算が可能である。

2.2 ヴォキャビュラリ

ヴォキャビュラリFPA-VOCの中に、浮動小数関連のワードと巨大数関連のワードがある。今回の多数桁数に関しても、関連するワードはすべてFPA-VOCの中に納められている。

3．多数桁数のデータ型

FORTH32の数値データのある浮動小数を簡単にfnumberと呼び、１つのfnumberを#fのように表している。今回の新しい数値データである多数桁数をgnumberと呼び、１つのgnumberを&gのように表すこととする。この節では、gnumberの定義、入力方法、出力方法について述べる。なお、１セルは４バイト＝32ビットのメモリーのこととする。

3.1 gnumberの定義

スタック上のgnumberは、次のような可変サイズの数値部と２セルのヘッダ部からなる。

```
□ □・・□      □・・□     □      □
n+2    |m|+3  |m|+2  ③    ②      ①
                          n      m
```

スタック・トップ①と２番目の②がヘッダである。①の中身は（gnumberの整数部桁数）×（gnumberの符号）であり、それをmとする。なお、単に桁数と記すとき、それは10万進法の桁数を意味するものとする。スタックの２番目②は、gnumberの整数部桁数と小数部桁数とを合わせた桁数となる。それをnとすると、３番目③から|m|+2番目までの|m|個のセルには整数部が入り、|m|+３番目からn+２番目までn-|m|個のセルには小数部が入る。なお、整数部は必ず１桁必要であり、次の制限がある。

$$n \geqq |m| \geqq 1$$

また、gnumberにおいては10万進法が使われ、スタックの３番目からn+２番目のセルに入る数値は、０以上99999以下の値でなければならない。

＜例１＞多数桁数123456.7890123456をスタックに置くには次の数値を順にスタックにプッシュする。

23456　78901　23456　00001　4　2

（２がスタック・トップ）

＜例２＞多数桁数－0.141592535をスタックに置くには次の数値を順にスタックにプッシュする。

26535　14159　00000　3　－1

（－１がスタック・トップ）

3.2 gnumberの入力

多数桁数をスタックに置くには、3.1の例のように該当する数値を順にスタックにプッシュすればよい。

もっと簡単な方法は、多数桁数のデータ記号&を先頭に付けて数値を入力することである。

＜例１＞多数桁数123456.7890123456をスタックに置くには次のように入力する。

&123456.7890123456

＜例２＞多数桁数-0.141592535をスタックに置くには次のように入力する。

&－0.141592535

指数形式を用いた次のような入力も可能である。

＜例３＞　&12.3456e10

＜例４＞　&12345e-10

なお、&を用いる多数桁数の入力に際しては、数値部の桁数が10進法で24桁以下という制限がある。

3.3 gnumber定数と変数

gnumber定数が４個定義されている。

&pi（ ---&g ）・・・πがプッシュされる。

なお、ここで示したようなFORTHのワードの説明方法に関しては付録Aを参照のこと。

同様に、&e（Napier数）、&ln２（２の自然対数）、&ln10（10の自然対数）が定義済みである。

gnumber定数、四則演算の結果、関数の戻り値などの小数部桁数を一括して設定するのが次のワードである。なお、既定として10進法で35桁（10万進法で７桁）に設定されている。

set-gdigits（ n--- ）・・・小数部桁数（10進法）をn以上でnに最も近い５の倍数に設定する。

次のgnumber定数ではその場限りの小数部桁数設定が可能である。

&pi>（ n---&g ）・・・m桁の小数部を持つgnumberがプッシュされる。ただし、mはn以上でもっともnに近い100の倍数とする。

同様に、&e>、&ln２>、&ln10>が定義されている。

小数部桁数を指定してgnumber変数を定義できる。

gnumvariable（ n--- ）・・・指定した名前で小数部n桁（10万進法）、整数部５桁分のメモリー領域を確保する。

gnum!（ &g, addr --- ）・・・addrのメモリーにスタックの&gをストアする。

gnum@（ addr ---&g ）・・・addrにあるgnumberをスタックにプッシュする。

＜例＞　7 gnumvariable xx

&pi xx gnum!

（変数xxの値を読み出して表示するには）

xx gnum@ g.

3.4 gnumberの表示

スタック上のgnumberを次のワードにより表示することができる。

g.（ &g ----)・・・スタック上のgnumberをポップして表示する。ただし、5桁ずつ空白で区切って表示する。また、整数部と小数部の間には小数点を置く。

g.r（ &g , n ----)・・・スタック上のgnumberの整数部とn桁分の小数部を表示する。

＜例1＞ &pi g.

＜例2＞ &pi 3 g.r

3.5 他のデータ形式との関連

FORTH32の数値データであるfnumberとgnumberの間の相互変換を行う次のワードがある。

f->g （ #f --- &g ）スタック上のfnumberをgnumberに変換する。

g->f （ &g --- &f)・・・スタック上のgnumberをfnumberに変換する。

＜例1＞ #pi f->g g.

＜例2＞ &pi g->f e.

（*）e. はスタック上のfnumberを指数形式で表示するワードである。

3.6 gnumberの末尾に0を挿入、末尾から0を削除

gnumberの小数部の末尾にあるゼロは有効桁の位置を示す役割を持っている。

例えば、3.1415926535は、小数部2桁（10万進法で）までが有効であるが、3.141592653500000は小数部3桁（10万進法で）までが有効である。

ginszeros（ &g, s --- &g')・・・末尾にs桁（10万進法で）ゼロを挿入する。

gfillzeros（ &g, s --- &g')・・・小数部桁数がs桁となるまで（10万進法で）ゼロを挿入する。

gfill1（ &g --- &g')・・・小数部が(既定＋1）桁となるようにゼロを挿入。

gdecsize（ &g, s --- &g')・・・小数部末尾のs桁（10万進法で）を削除する。

garrange（ &g, s, h --- &g')・・・末尾にs桁（10万進法で）ゼロを挿入し、整数部の最高位にh桁（10万進法で）ゼロを挿入。

3.7 gnumberの正規化

入力や計算によってスタック上に置かれたgnumberは、その整数部の最高位に無用のゼロがあることがある。それは取り除いておく必要がある。また、ゼロについては、－0は許されないので、＋0に直しておくことにする。例えば、０１－１がある場合は０１１に変更しなければならない。このような作業がgnumberの正規化である。

gnormalize（ &g --- &g')・・・gnumberの正規化を行う。

4．スタック操作

gnumberは、fnumberと同様にスタックに置かれ、計算などの数値処理を受けることになる。そのため、スタック上のgnumberに関してコピー、入れ替えなどのスタック操作が必要になる。

4.1 簡単なスタック操作ワード

スタック上のgnumberを操作する次のようなワードがある。

gdup（ &g --- &g, &g)・・・スタック・トップのgnumberのコピーをスタックにプッシュする。

gswap（ &g2, &g1 --- &g1, &g2)・・・トップと2番目を入れ替える。

gdrop（ &g ---)・・・トップを取り去る。

gover（ &g2, &g1 --- &g2, &g1, &g2)・・・2番目のコピーをプッシュする。

gnip（ &g2, &g1 --- &g1)・・・2番目を取り去る。

grot（ &g3, &g2, &g1 --- &g2, &g1, &g3 ）・・・3番目をトップに移動する。

gmrot（ &g3, &g2, &g1 --- &g1, &g3, &g2 ）・・・トップを3番目に移動する。

g2dup（ &g2, &g1 --- &g2, &g1, &g2, &g1 ）・・・トップと2番目のコピーをプッシュ。

4.2 番目指定付スタック演算ワード

何番目のgnumberを対象とするかパラメータで指定してスタック演算を行う以下のワードがある。ただし、何番目かは0番目、1番目、・・・とゼロから数えるものとする。

gpick

（ &gn, .., &g1, &g0, n --- &gn, .., &g1, &g0, &gn ）・・・n番目のコピーをプッシュする。

gnip.n

（ &gn, .., &g1, &g0, n --- &gn-1, . . , &g1, &g0 ）

・・・n番目を取り去る。

gins.n（ &gn, .., &g1, &g0, &g, n ---
&g, &gn, .., &g1, &g 0 ）・・・指定の
gnumberをn番目の前に挿入する。

＜例１＞ &2 &1 &0 2 gnip.n g. g.

＜例２＞ &2 &1 &0 &9 2 gins.n g. g. g. g.

5．スタック演算

gnumberの比較演算と数値演算に関するスタック演算ワードがある。

5.1 比較演算

大小などの比較ワードは、対象のgnumberを初めのままスタックに残すように定義されている。

まず、ゼロとの比較では、対象のgnumberと同じ小数部桁数を持つゼロと比較する。

&0=?（ &g --- &g , flg ）・・・ゼロに等しければ
flg＝－1，さもなくばflg＝0をプッシュ。

&0>?（ &g --- &g , flg ）・・・ゼロより大きければ
flg＝－1をプッシュ。

&0<?（ &g --- &g , flg ）・・・ゼロより小さければ
flg＝－1をプッシュ。

２数の比較では、小数部の桁数が小さい方にゼロを挿入し小数部の桁数を揃えてから比較を行う。

g=?（ &g2, &g1 --- &g2, &g1, flg ）・・・２番目とトップが等しければflg＝－1をプッシュ。

g>?（ &g2, &g1 --- &g2, &g1, flg ）・・・２番目がトップより大きければflg＝－1をプッシュ。

g<?（ &g2, &g1 --- &g2, &g1, flg ）・・・２番目がトップより小さければflg＝－1をプッシュ。

5.2 四則演算

２数の演算では、小数部の桁数と設定されている桁数のうち大きな方の桁数を小数部桁数とし、必要ならばゼロを挿入し２数の小数部の桁数がそれと同じになるように調整してから演算を行う。

g+（ &g2, &g1 --- &g ）・・・２番目とトップを取り去り、それらの加算を行い結果をプッシュする。

g-（ &g2, &g1 --- &g ）・・・２番目からトップを減算する。

g*（ &g2, &g1 --- &g ）・・・２番目にトップを乗算する。

g/（ &g2, &g1 --- &g ）・・・２番目をトップで除算する。

gmod（ &g2, &g1 --- &gr ）・・・割算を行って整数の商と余りを得て、余りをプッシュ。

正の整数との掛算・割算に関しては次のワードによって高速な演算が可能である。

gnumintdiv（ &g, n --- &g/n ）・・・正の整数nで割った値をプッシュ。

gnumintmul（ &g, n --- &g*n ）・・・正の整数nを掛けた値をプッシュ。

5.3 １項演算

１項演算はスタック上の１つのgnumberに対して行われる演算である。

gabs（ &g --- &g' ）・・・トップの絶対値をプッシュする。

gnegate（ &g --- &g' ）・・・トップの符号を変えてプッシュする。

g->gint（ &g --- &g' ）・・・トップの小数部を切り捨てる。

6．関　数

各種のgnumber関数が定義されている。

6.1 指数関数と対数関数

以下では、トップのgnumberをxとする。

gexp（ &g --- &g' ）・・・exp(x)をプッシュ。

gexp10（ &g --- &g' ）・・・10^xをプッシュ。

gln（ &g --- &g' ）・・・ln(x)をプッシュ。

glog10（ &g --- &g' ）・・・log10(x)をプッシュ。

gnumberの関数計算では、答の小数部の精度を保障するアルゴリズムが採用されている。特に問題となるのは、gexpやgexp10である。例として、gexpを取り上げると、exp(１)を小数部７桁（10万進法）まで求めるには、マクローリン展開を小数部８桁の精度で計算して最後に８桁目を切り捨てて答が得られる。ところが、例えばexp(200)＝e^200を求めるにはこのような方法ではうまく行かない。

nを大きな整数としてexp(n) を求める場合を考察する。小数部桁数をMとして、Napier数eは

$$e = e_M + \delta$$

ここで、e_Mはeの小数部をM桁まで取ったもの、残り$M+1$桁以降の部分をδとする。このとき、

$$e^n = {e_M}^n + n{e_M}^{n-1}\delta + \ldots$$

nが大きな整数であればこの式の第２項は大きな値となる。それが小数K桁の大きさとしよう。

$$n{e_M}^{n-1}\delta \approx 10^{-5K} \text{ と } \delta \approx 10^{-5M} \text{ から}$$

$$M - K \approx \{\log_{10} n + (n-1)\log_{10} e\}/5 = \Delta$$

したがって、exp(n)を小数部K桁の精度で求めるためには、まず、eを小数部M($=K+\Delta$)桁の精度で求めてから、e^nを計算し、結果の小数部の末尾をΔ桁切り捨てて小数部K桁の答とすることが必要である。

以上の方法で、小数部の精度を保ってexp(x)の値を求めることができる。

6.2 3角関数

3角関数では角度の単位はradである。

gsin (&g --- &g')・・・sin(x)をプッシュする。
gcos (&g --- &g')・・・cos(x)をプッシュする。
gtan (&g --- &g')・・・tan(x)をプッシュする。
gatan (&g --- &g')・・・arctan(x)をプッシュする。

6.3 冪乗関数

g2** (&g --- &g')・・・x^2をプッシュ。
g** (&g, n --- &g')・・・x^nをプッシュ。
gsqrt (&g --- &g')・・・x^(1/2)をプッシュ。
gcbrt (&g --- &g')・・・x^(1/3)をプッシュ。
gnthroot (&g, n --- &g')・・・x^(1/n)をプッシュ。
ginv (&g --- &g')・・・1/xをプッシュ。

6.4 階乗

gn! (&g --- &g')・・・トップを整数化しその階乗を求める。
gpermu (n, r --- &g)・・・2番目(=n)とトップから(=r)、nPrを求めプッシュする。
gcombi (n, r --- &g)・・・2番目(=n)とトップから(=r)、nCrを求めプッシュする。

7．Eulerの定数の計算

Eulerの定数γは次のように定義される。

$$\gamma=\lim_{n\to\infty}\left(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n-1}-\log n\right)$$

γの値は上の定義式を直接用いるのではなく、次の漸近展開公式を使って求める。

$$\gamma=\left(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{N-1}-\log N\right)+$$
$$+\frac{1}{2N}+\sum_{k=1}^{M}\frac{B_k}{2k}\frac{1}{N^{2k}}+O\left(\frac{B_{M+1}}{2(M+1)}\frac{1}{N^{2(M+1)}}\right)$$

ここで、B_k はBernoulli数である。

例としてγを1000桁求めるとする。$N=100000$とする場合、

$$\frac{B_{M+1}}{2(M+1)}\frac{1}{N^{2(M+1)}}\approx 10^{-1005}$$より、$M=140$。

したがって、γを1000桁計算するには、$k=140$まで、

$\frac{B_k}{2k}\frac{1}{N^{2k}}$を求めればよいことがわかる。

さて、旧版のFORTH32では、巨大数を扱うことができた。γは本来小数であるが、$\gamma\times 10^{-1000}$の整数部は10進法で1000桁の数、10万進法で200桁の巨大数とみなすことができる。このように巨大数として取り扱かうことにより$\gamma\times 10^{1000}$の整数部を求めることができた[4]。

今回多数桁の小数を直接扱える新しいFORTH32が完成したのに伴い、それを用いてγの小数部1000桁を求めるプログラムを作成した。プログラムの実質的な内容は、文献[4]のプログラムと同様であるが、形式的にはgnumberを用いる明解なプログラムとなっている。しかしプログラムの実行時間に関しては、以前のプログラムと比べて4倍程度の時間がかかる。

8．Mayの方程式

FORTH32にFPA（浮動小数点演算機構）を装備した際に、FPAを用いてMayの方程式の周期解を調べた[1]。

さて、Mayの方程式のm周期解とは、関数$f(a,\ x)$をm回繰り返し用いて得られる関数$f_m(a,\ x)$の値が始めに戻るという方程式

$$f_m(a,\ x)=x$$

の解のことである。ただし、

$f(a,\ x)=ax(1-x)$, ただし、$0<a<4$, $0<x<1$
$f_1(a,\ x)=f(a,\ x)$, $f_2(a,\ x)=f(a,\ f_1(a,\ x))$
$f_3(a,\ x)=f(a,\ f_2(a,\ x))$
$\cdots$, $f_m(a,\ x)=f(a,\ f_{m-1}(a,\ x))$

Mayの方程式の周期解とパラメーターaに関して次の(1)～(3)が知られている。

(1) 収束する増加数列a_1, a_2, a_3, …が存在する。
(2)(A) $a_1\le a$では、2周期の解が存在する($m=2$)。ただし、$a_1=3$
(B) $a_2\le a$では、4周期の解が存在する($m=4$)。ただし、$a_2=\sqrt{6}+1$
(C) $a_n\le a$では、m周期の解が存在する($m=2^n$)。
(D) a_1, a_2, a_3, …の極限をa_cとおくと$a_c\approx 3.57$。
(3) $a>a_c$では初期値x_0の値によってさまざまな周期の解が出現する。この領域はカオス領域と呼ばれる。

さて、$a_n<a$においてa_{n+1}を求めるには

$a=a_{n+1}$では、$f_m(x)=x$と$\frac{d}{dx}f_m(x)=-1$を

同時に満たすxの値が存在する（$m=2^n$）

ことを用いる。

筆者は、FORTHのFPAを用いてa_{13}までの値を求めた[1]。ただし、計算の精度は10^{-23}であった。

今回は開発したFORTH32の多数桁数を用いてa_{20}までの値を求めた。得られたa_2からa_{20}までの値のリストを付録Bに載せた。今回の計算方法は、$f_m(x)=x$と$\frac{d}{dx}f_m(x)=-1$とを同時に満たすaとxの値をNewton法を用いて求めるものである。この計算のプログラムを付録Cに載せた。なお、計算のプログラムでは多数桁数の小数部の桁数は自由に設定できるが、今回は、多数桁数の小数部は35桁（10進法）、Newton法の収束の精度は10^{-32}、リストの表示は小数点以下30桁目までと設定した。

付録A　FORTHのワードの説明方法について

ワードの説明は次のような形式で行われる。

xxx（　&g,　n ---　&g'　）　・・・説明文

ここで、xxx　はワードの名前である。次の左カッコと右カッコの間は、スタックの状態の変化を示すコメントである。カッコ中で　---　（または，　---　）の左側はワードxxxの実行前のスタックの状態、右側は実行後のスタックの状態を示している。また、・・・以降は単なる説明文である。

上の例では、スタック・トップに整数、２番目にgnumberが置かれている状態でxxxを実行すると、始めの２つが取り去られて、得られた新しいgnumberがトップに置かれることを示している。このトップを見るには次のように続いてg.を実行する必要がある。

&pi 3 xxx g.

このとき、スタックに何も結果は残っていない。結果を残しておくには、例えば次のように入力するとよい。

&pi 3　xxx gdup g.

付録B　Mayの方程式の周期解に関するパラメーターaのリスト

a_nは2^n周期解の発生するaの値である。

n	a_n					
2	3.44948	97427	83178	09819	72840	74705
3	3.54409	03595	51922	85361	59659	86604
4	3.56440	72660	95432	59777	35575	86528
5	3.56875	94195	43826	43129	82102	80025
6	3.56969	16098	01396	71428	82687	06295
7	3.56989	12593	78120	48732	02712	00585
8	3.56993	40183	73976	40118	48556	01887
9	3.56994	31760	48401	63635	44429	76162
10	3.56994	51373	42169	79431	47211	22749
11	3.56994	55573	91249	37611	71699	41756
12	3.56994	56473	52899	79134	35066	53760
13	3.56994	56666	19930	45219	39437	01151
14	3.56994	56707	46338	40814	49671	61967
15	3.56994	56716	30088	62964	34152	91546
16	3.56994	56718	19360	86406	35302	43548
17	3.56994	56718	59897	18060	46228	34728
18	3.56994	56718	68578	81826	66364	32691
19	3.56994	56718	70438	15919	65910	16302
20	3.56994	56718	70836	37308	53058	90018
21	3.56994	56718	70921	65830	57895	21755
22	3.56994	56718	70939	92378	83331	24361
23	3.56994	56718	70943	83569	53512	30531
24	3.56994	56718	70944	67350	60404	24073

付録C　ワードgetacの定義リスト

```
SCR　＃24　math2005.blk
  0 (function)
  1 : ff (a, x --- y) gdup gnegate g1+g* g* ;
  2 : ffdx g2* gnegate g1+g* ;
  3 : ffda gnip gdup gnegate g1+g* ;
  4 : ffdxa gnip g2* gnegate g1+ ;
  5 : ffdxx gdrop g2* gnegate ;
  6 : ff@ aa@ f0@ ff ;
  7 : ffdx@ aa@ f0@ ffdx ;
  8 : ffda@ aa@ f0@ ffda ;
  9 : ffdxx@ aa@ f0@ ffdxx ;
 10 : ffdxa@ aa@ f0@ ffdxa ;
 11 : --> 

SCR　＃25　math2005.blk
  0
  1 : getac
  2  &1e-32 eps g1!
  3  1 nn ! &3.5aa! &0.5 xx!
  4  23 1 do
  5  nn @ 2 * nn ! 0 flg !
  6  200 0 do
  7  xx@ f0! &1 fx0! &0 fa0! &0 fxx0! &0 fxa0!
  8  nn @ 0 do
  9  ff@ f1!
 10  ffdx@ fx0@ g* fx1!
 11  ffda@ ffdx@ fa0@ g* g+ fa1!
 12  ffdxx@ fx0@ gdup g* g* ffdx@ fxx0@ g* g+
```

```
    fxx1!
 13 ffdxa@ ffdxx@ fa0@ g* g+ fx0@ g* ffdx@ fxa0
    @g* g+ fxa1!
 14 f1@ f0! fx1@ fx0! fa1@ fa0! fxx1@ fxx0! fxa1@
    fxa0!
 15 loop -->
SCR #26  math2005.blk
  0 (cont)
  1 f1@ xx@ g- gg1! fx1@ g1+ gg2! fx1@ gone
    g- dxgg1!
  2 fa1@ dagg1! fxx1@ dxgg2! fxa1@ dagg2!
  3 dagg1@ dxgg2@ g* dxgg1@ dagg2@ g* g- dd!
  4 gg1@ dxgg2@ g* gg2@ dxgg1@ g* g- dd@ g/
    gnegate daa!
  5 gg2@ dagg1@ g* gg1@ dagg2@ g* g- dd@ g/
    gnegate dxx!
  6 aa@ daa@ g+ aa1! xx@ dxx@ g+ xx1!
  7 daa@ aa@ g/ gabs eps g1@ g<?
  8 if gdrop gdrop -1 flg! leave then
  9 aa1@ aa! xx1@ xx!
 10 loop
 11 flg @ 0<> if cr nn @ 2 * 8 .r ."  "
 12 aa@ 6 g.r
 13 else cr ." stop" leave then
 14 &0.5 xx!
 15 loop ;
```

参考文献

1）貞方一也:「FORTHと浮動小数点演算」、北海道医療大学基礎教育部論集、第13号、29ページ、1987年。

2）貞方一也:「FORTHと巨大数」、北海道医療大学基礎教育部論集、第26号、A15ページ、2000年。

3）貞方一也:「FORTHと多数桁計算」、北海道医療大学情報センター年報、第2巻、21ページ、2004年。

4）貞方一也:「Euler定数の多数桁計算」、北海道医療大学看護福祉学部紀要、第10巻、81ページ、2003年。

Many-digit number Arithmetic on 32-bit FORTH

Ichiya SADAKATA*

Abstract：A number which has the integer part of arbitrary digits and the decimal part of arbitrary digits is called a Many-digit number. Recently we have fully installed a Many-digit number Arithmetic on 32-bit FORTH. This FORTH (ver. 2.8 of FORTH32) has various Many-digit number functions in addition to Many-digit number stack manipulations and basic arithmetic operations.

By using the Many-digit number Arithmetic we have calculated Euler's constant up to1000 digits below the decimal point. The program for this calculation is simpler than the former program based on Big-number Arithmetic, but the execution time is four times that of the former program. We also have studied a m-cycle solution (m=4, 8, ...) of May's equation, y =ax(1-x) and gotten the value of the parameter a at which a m-cycle solution occurs (m= 4, 8, ..., 2^24) by the accuracy of 30 digits.

Key words：32-bit FORTH, FORTH32 Many-digit arithmetic, Euler's constant,May's equation

* Department of Integrated Human Sciences