各種浮動小数点表現法の評価方式の実現†

森 岳志†† 中川正樹†††
髙橋延匡††† 中森眞理雄†††

本論文は，新しい浮動小数点表現法の比較・評価を行う環境を提供することを目的とし，OS/o 用言語Cコンパイラ CAT に複数通りの浮動小数点表現法を実現したことについての報告である．本研究では，(1) URR (2) IEEE 表現法 (3) MIL-STD-1750A表現法の3種類の表現法をサポートしている．さらに，表現法の選択を，(ア) コンパイル時に行う方法，(イ) 実行時に行う方法の両者を用いて実現した．後者の方法は，効率の良い評価環境を与えることを目的として考案された方式で，評価の際のコンパイルとリンクの手間を軽減し，新たに表現法を追加したとしても浮動小数点演算ライブラリの再編成だけを行えばよく，プログラムの再コンパイルを必要としないことなどの特徴を有している．定数の扱いに問題が生じたが，現時点では実行時に内部表現に変換することによって解決している．

1. はじめに

計算機の名実ともに重要な応用分野として数値計算がある．特に，数値計算専用マシンとして発達したスーパコンピュータの出現は，種々の問題解決に寄与している．

一方，パーソナルコンピュータ（以下パソコンと略す）やワークステーションと呼ばれるパーソナル・ユースの計算機が安価で手に入れられるようになった．そして，8086用数値演算用コプロセッサ8087や，68000数値演算用コプロセッサ68881などの出現，さらにはV60のような浮動小数点演算機能を組み込んだマイクロプロセッサの出現によってパソコン上でも，数値計算がある程度の速度で行える時代が到来した．

数値計算を行う際，浮動小数点演算機能は必要不可欠な道具として用いられているが，以下に示すように，浮動小数点表現法そのものに対する問題点がクローズアップされてきている．

(1) 精度の問題

スーパコンピュータなどで，従来では考えられなかったほどの演算回数をこなす場合，はたして現在の語長で精度が保たれるのか，といった問題がある．

(2) 表現領域，ひいては“あふれ”の問題

エンドユーザの飛躍的拡大を考えた場合，パーソナルコンピュータなどで“あふれ”の処理を従来どおりにしておいてよいのかといった問題がある．表現法のフォーマット，さらに“あふれ”が発生した場合のOS側の対処など再考の必要がある．

以上，スーパコンピュータとパーソナルシステムを例にあげたが，汎用大型計算機においても，ネットワークの中核プロセッサとしての位置付けと性能向上の目的から両者の問題を露呈しつつある．

近年になり，これらの問題を解決することを目的とした松井・伊理表現法[1]やURR（万能実数値表現法）[2),3)]，規格化を目的としたIEEE表現法[4]などの新しい浮動小数点表現法が提案された．

筆者の属する研究室では，1979年度から，パーソナル・ユースを前提とした，手の入れられる“自前の”研究用計算機システムの開発を行っている．我々はそのオペレーティングシステムをOS/omicron（以下OS/oと略す）と呼んでいる[5]．

OS/oでは，システム記述言語に言語Cを採用し，そのための言語CコンパイラCAT (C compiler developed at Tokyo University of Agriculture and Technology) を，1986年春までに開発した[6),7)]．OS/oでは，システムプログラムだけではなく，アプリケーションプログラムの記述にも言語Cを採用しており，浮動小数点演算機能は必要不可欠である．

以上の背景の下で，我々は，次に示す目的からCATに新しい浮動小数点表現法を実現しようと考えた．

(1) 新しい浮動小数点表現法の比較・評価を行う環境を提出すること

パーソナルな計算機上での数値計算は，大型計算機のそれと比較して，専門家ばかりでなくより広範囲の

† A Programming Environment for the Practical Evaluation of Multiple Schemes of Floating Point Representation by TAKESI MORI (Software Engineering Development Laboratory, NEC Corporation), MASAKI NAKAGAWA, NOBUMASA TAKAHASHI and MARIO NAKAMORI (Faculty of Technology, Tokyo University of Agriculture and Technology).

†† 日本電気(株)ソフトウェア生産技術開発本部
††† 東京農工大学工学部数理情報工学科

人々が，すなわち，計算機システムに関し十分な知識を持たない人たちまでが，それぞれ多様な目的で数値計算を行うという面で異なっている．例えば，オーバフロー回避のために余儀なくされる，いわゆる“生活の知恵”のような式の変更をユーザから解放する必要がある．

そのようなパソコン上での数値計算という新しい計算環境を考えると，数値計算のためには必須のデータ構造の浮動小数点表現法を根本から見直す必要がある．そこで，我々はCATに複数通りの浮動小数点表現法を実現し，それらを比較・検討するための環境を与えることにした．

（2） 新しい浮動小数点表現法の数値計算に及ぼす影響を調査すること

実際に数値実験を試みることによって，各種浮動小数点表現法の比較・評価を行い，数値計算に及ぼす影響を調査する．今まで提案されている表現法，あるいは今後考案されるかもしれない表現法をそれ自体だけでは理論的に評価することは極めて難しい．表現能力，精度，計算速度，コストなどは各種アプリケーションに応じて利害得失があり，実践的に容易に比較検討できる環境が必要となる．

（3） OS/o 上で動作するアプリケーションプログラムのために，浮動小数点演算機能を実現すること

当研究室では，オンライン手書き文書認識，LISP処理系などの研究・開発も行われている．それらの開発はCATを用いて行われるために，浮動小数点演算機能は必要不可欠な機能となる．

次章では，CATに組み込まれた浮動小数点表現法の種類とCATの言語仕様について述べ，第3章では，作成した浮動小数点演算の命令セットについて述べる．第4章で，CATに複数通りの浮動小数点表現法を組み込む場合の問題点とその解決法について述べる．第5章では，その方式において問題となった点を解決した新しい評価方式の設計と，標準OSであるCP/M-68K上でのプロトタイプの実現について述べる．第6章では，プロトタイプの問題点と本評価方式のOS/oへの移行について述べ，最後に本研究の成果と今後の課題について述べる．

2. CATにおける浮動小数点表現法

CATでは，第1章で述べたような新しい浮動小数点表現法の数値計算に及ぼす影響を調査したいと考え，複数通りの浮動小数点表現法を採用している．

2.1 組み込まれた浮動小数点表現法

現在，CATは次に示す3種類の浮動小数点表現法を組み込んでいる．

（1） URP[2),3)] (図 1)

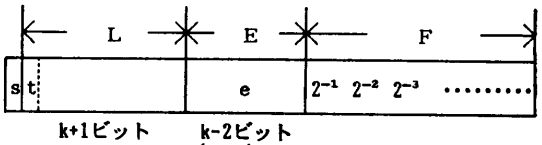
オーバフロー，アンダフロー問題の解消を目的として提案された表現法である．その他，データ長に独立であること，同じビットパターンのままで固定小数点数と見なした場合と大小比較が等しいこと，∞を除くとあらゆるビットパターンが“数”として意味をもつことなど数多くの利点を備えている．

（2） IEEE 表現法[4)] (図 2)

浮動小数点表現法の規格化を目的として提案された．オーバフローやアンダフロー後の処理を規定していること，非数の概念を導入していることなど，現在の浮動小数点表現法に対する問題解決の意図がうかがえる．

（3） MIL-STD-1750A 表現法 (図 3)

伝統的な表現法で，表現可能範囲や表現誤差より，



k+1ビット　k-2ビット (k≠1)

L --- 000···01 または 111···10
kはL部中の0または1の連なる数

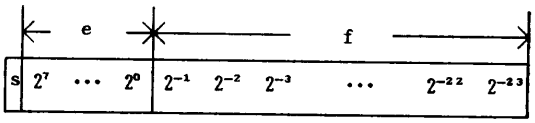
$V=f' \times 2^{e}$

$f' = 1+f$ (s=0のとき)
$=-2+f$ (s=1のとき)

$e' = 2^{k-2}+e$ (s=0,t=1のとき)
$=-2^{k-1}+e$ (s=0,t=0のとき)
$=-2^{k-2}+e-1$ (s=1,t=1のとき)
$= 2^{k-1}+e-1$ (s=1,t=0のとき)

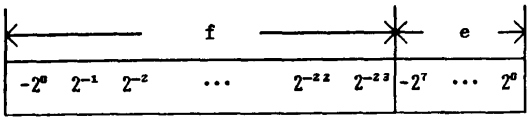
ただしk=1のときは
$e' = 0$ (s≠tのとき)
$=-1$ (s=tのとき)

図 1 URR
Fig. 1 URR for floating point representation.



e=0 & f=0 のとき $V = (-1)^s \times 0$
e=0 & f≠0 のとき $V = (-1)^s \times f \times 2^{-126}$
0<e<255 のとき $V = (-1)^s \times (1+f) \times 2^{e-128}$
e=255 のとき V = NAN(非数)

図 2 IEEE 表現法
Fig. 2 IEEE format for floating point representation.



$V = f \times 2^e$

図 3 MIL-STD-1750A 表現法
Fig. 3 MIL-STD-1750A format for floating point representation.

実行速度に重点をおいた表現法である．その意味で，URR とは対照的な表現法であるといえる．

2.2 CAT の言語仕様

CAT は基本的には言語C参照マニュアル[8]に準拠する立場をとった[6]．これは言語の性格上，互換性を考慮したためである．しかし，浮動小数点数に関しては，第1章の研究目的のために，以下に述べる追加・修正を行った．

（1） データ型の追加

参照マニュアルでは浮動小数点数に対するデータ型は float (32 ビット) と double (64 ビット) である．しかし，我々はこれらに加えて short 型 (16 ビット) を用意した．これは，浮動小数点数としてデータ長に独立という URR の長所を生かすためである．型の宣言は次のように行う．

例 short float x;

（2） 算術型変数，定数の変換規則の変更

参照マニュアルでは，すべての浮動小数点演算について，単精度が式中に現れると，その小数部に 0 が付け加えられて倍精度にのばされることになっている．しかし，CAT では MC68000 のアーキテクチャを生かし，また単精度で十分なアプリケーションも多いと考え，単精度を強制的に倍精度に変換することはしていない．

（3） 定数表記の追加

上述の(1)と(2)の変更により，これらに対する型の演算が新たに加わった．その結果，ソースプログラムの式中にそれらの定数データを表記する必要が生じた．

(i) 倍精度浮動小数点数を指定するために，定数の後ろに d/D を付け加える．

(ii) short 型浮動小数点数を指定するために，定数の後ろに s/S を付け加える．

次に定数表記の例を示す．デフォルトは単精度浮動小数点数である．

例 1.0D, 1.0S

3. 浮動小数点演算に対する命令セット

以下に示す浮動小数点演算命令を用意した．

（1） 指数部と仮数部の切り出し
（2） 指数部と仮数部の結合
（3） 四則演算
（4） 整数型から浮動小数点方式の内部表現への変換，およびその逆変換
（5） 10 進文字列から浮動小数点方式の内部表現への変換，およびその逆変換
（6） 絶対値
（7） 比較
（8） ネガティブ
（9） 整数化
(10) 平方根
(11) 指数関数
(12) 対数関数
(13) 三角関数
(14) 逆三角関数

指数部と仮数部が固定長である浮動小数点表現法と比較して，URR ではそれらが可変長であるために，処理方法によっては，指数部と仮数部の分離，結合の処理に大きな手間を要する．そのため，分離・結合の処理の，さまざまな高速化を試み[9]，基本命令として用意した．

4. CAT への浮動小数点演算機能の組込み

複数通りの浮動小数点演算の実現で問題になるのは，定数の内部表現への変換と演算ルーチンの切り換えである．本章では，まず，定数変換をコンパイル時に行う方式の設計と実現を述べる．

4.1 定数の処理

プログラム中に現れる定数は，コンパイル時に適用される浮動小数点表現の内部表現に変換する．表現法の選択は，コンパイル時のオプション機能によって行っている．

CAT は，浮動小数点定数が出現したとき，ユーザが指定した浮動小数点表現法の内部表現に定数を変換し，オブジェクトコードの中にそれを埋め込むことになる．

定数から浮動小数点表現への変換や初期化式に対する演算を行うため，コンパイラ自身に複数通りの浮動小数点演算ライブラリをリンクしている．

4.2 コンパイラの生成するコード

OS/o の環境の下で，効率の良いコードを調査するため，次に示す二つの方法を用いて浮動小数点演算機能を実現した．

方法 1 関数呼出しによる方法

浮動小数点演算に対し言語Cの関数呼出しと同等なコードを生成する方法である．

この場合，浮動小数点演算を行う関数はライブラリとして用意される．OS/o は静的リンクの環境であるため，ユーザは**図 4** に示すようにユーザプログラムと浮動小数点演算ライブラリをリンクすることによって目的となる実行形式を得る．演算の種類（四則演算の区別など）や適用される浮動小数点表現法の選択は，コンパイラが生成する関数名によって行う．

方法 2 エミュレーションによる方法

MC63000 ではコプロセッサに対応できるような特定の命令を用いることによって，ソフトウェアによるエミュレーションを行うことができる．この命令はエミュレータトラップと呼ばれ，通常のスーパバイザコールと同等の能力を持つ．

コンパイラは，浮動小数点演算に対しエミュレーション命令を生成する．実行時には CPU は浮動小数点演算をハードウェア命令の拡張とみなして実行する．

演算の選択は，MC68000 の命令体系に即したエミュレーション命令の実効アドレスによって行われる．

現在のエミュレーションの命令体系を**図 5** に示す．

ユーザはユーザプログラムの実行に先立ち，エミュレーションの割込みベクタの内容を書き換えることに

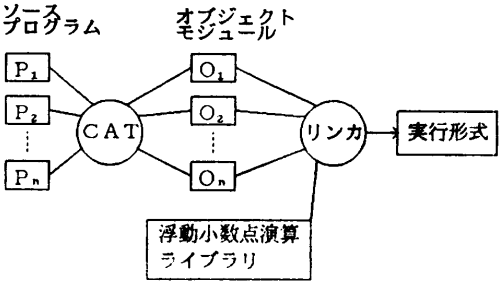


図 4 ライブラリのリンク
Fig. 4 Linkage with floating point arithmetic libraries.

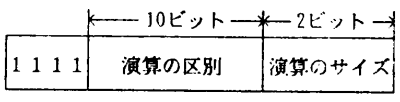


図 5 MC 68000 のエミュレーションの命令構成
Fig. 5 Instruction format for floating point arithmetic emulation.

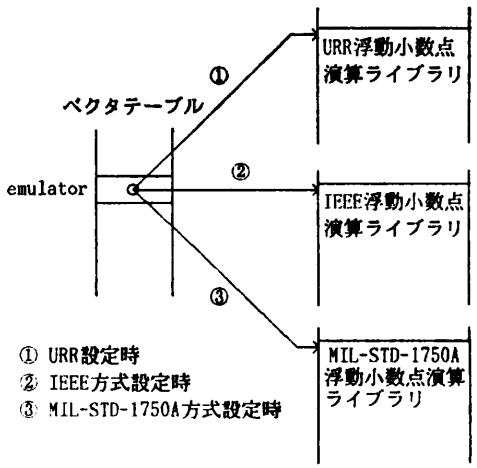


図 6 ライブラリの切替え
Fig. 6 Switch of floating point arithmetic libraries.

よって浮動小数点演算ライブラリの切り替えを実現する（**図 6**）．

関数呼出しによる方法とエミュレーションによる方法の実行速度の比較は，関数呼出しの手間と例外処理の手間との比較になる．OS/o では動的なリロケータビリティを実現するため，絶対アドレスを一切用いない．そのために，関数呼出し用に MC68000 に備えられている絶対番地によるジャンプやリターン命令（JSR, RTS 等）は用いず，多少，関数呼出しの手間が大きくなっている．それに対し，エミュレーションによる方法では例外処理に時間を費やす反面，OS/o プログラム実行環境とは異なる動作環境を設定し，JSR 等の MC68000 に備えられた関数呼出し用の命令を用いて関数呼出しを行い，速度の向上を図っている．

試作したシステムにおける四則演算の計算時間の平均は，関数呼出しによる方法では 2.538 μs，エミュレーションによる方法では 2.077 μs（12.5 MHz/クロックの CPU で測定）であった．このように，OS/o 実行環境の下では，エミュレーションによる方法の方が計算時間において有利となった．さらに，今後のハードウェア化や ROM 化への拡張性も考慮し，OS/o では浮動小数点演算に対し，エミュレーションのコードを出力している．

4.3 複数通りの浮動小数点演算に対応した CAT の実現

以上に述べた定数の処理，エミュレーションによる演算の採用により，URR, IEEE 表現法，MIL-STD-1750A 表現法のそれぞれの浮動小数点表現法に対応した 3 種類の CAT を実現した．URR を浮動小数点

表現とした高水準言語処理系は CAT が初めてである.

5. 新しい評価方式

5.1 浮動小数点評価方式としての問題点

浮動小数点表現法の比較を行うため，前章で述べた CAT を用いていくつかの数値実験を行った．その場合，図7のように同一の評価用プログラムに対し，比較を行う浮動小数点表現法の数だけコンパイルとリンクの作業が必要になる．評価を行う際には，プログラム中に記述されている定数にさまざまな値を設定し，その影響を調査することも行われるが，その度にこの作業を行うことは，大きな負担となった．

5.2 新しい評価方式の設計方針

我々は前節で述べた問題点を解決するため，以下に述べる条件を満たす新しい評価方式を考えた．

（1） 一回のコンパイルおよびリンクの作業で，複数通りの表現法が適用できること

図8のように，比較する浮動小数点表現法の数に依存せずに一回のコンパイル作業によって，さらには一つの実行形式によって複数通りの浮動小数点表現法の適用を可能にする．これによって，コンパイル，リンクの作業に要する手間を軽減し，効率の良い比較を行う環境を与えることが可能となる．

（2） 比較対象となる浮動小数点表現法の数を限定しないこと

現時点では3種類の表現法を実現しているが，将来，他の表現法（例えば松井・伊理表現法[1]など）を採用する可能性がある．拡張性を考慮し，実現される浮動小数点表現法の数は限定しない方式を採用する．

（3） 浮動小数点表現法を追加してもコンパイラの変更を必要としないこと

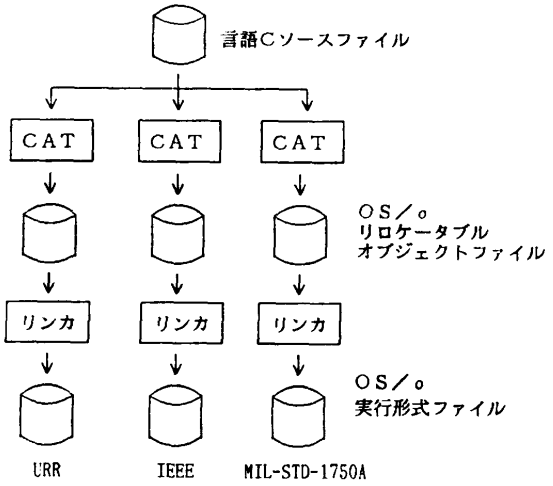


図7 コンパイル時に表現法を選択する方法
Fig. 7 Compilation and linkage of object programs in the compile time selection method.

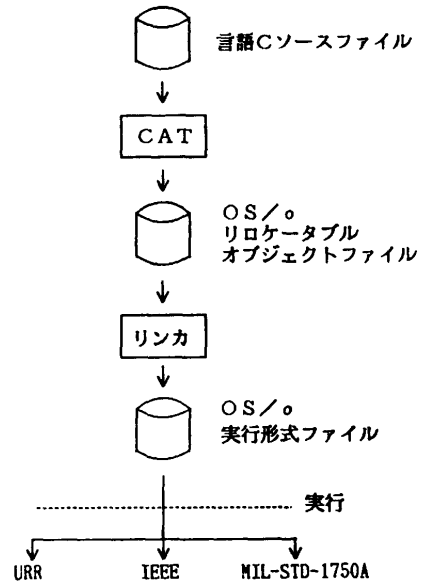


図8 実行時に表現法を選択する方法
Fig. 8 Compilation and linkage of object programs in the run time selection method.

5.3 実現上での問題点

前節で述べた評価方式を実現する場合，プログラム中に出現する定数の処理とダイナミックな浮動小数点表現法の選択が問題となった．

上述の問題点に対する解決法を次に述べる．

（1） プログラム中に現れる定数の処理

本評価方式を実現するためには，コンパイル時には定数を浮動小数点表現法の内部表現に変換できない．そこで，プログラム中に記述されている定数は実行形式の中には文字列で残し，プログラムのロード時，または実行時に，適用される浮動小数点表現法の内部表現に変換すればよいと考えた．

（2） ダイナミックな浮動小数点表現法の選択

マルチタスクの環境において，ダイナミックな浮動小数点表現法の切り替えは OS にその機能を加える必要がある．

5.4 新しい評価方式の試作

前節で述べた評価方式のプロトタイプをCP/M-68Kのもとで実現した．ただし，CP/M-68K はシングルタスクであるため(2)の機能は実現していない．

CP/M-68K 上で実現したのは，標準 OS であるの

で，各大学の専門家に配布し，本方式をより深く検討してもらいたいと考えたためである．

プログラム中に現れる定数は，プロトタイプでは実行時に変換している．すなわち，定数については，完全にインタプリティブな処理をしている．そのため，URR の実行速度に悪影響を及ぼさないように，プログラム中の定数は URR の内部表現と文字列定数の両者をオブジェクト内に持つことにした．この方法を採用することによって，URR の実行速度は，コンパイル時に定数を変換する方法，すなわち従来の評価方式とほぼ同等になり，さらに方針(2)，(3)は達成される．

```
                    テストプログラムの実行
E>test1 <-----------
selected URR !!
k=0000010    result=+2.5937425E+0
k=0000050    result=+2.6915880E+0
k=0000100    result=+2.7048138E+0
k=0000500    result=+2.7155690E+0
k=0001000    result=+2.7169242E+0
k=0005000    result=+2.7180056E+0      URRの出力
k=0010000    result=+2.7181410E+0
k=0050000    result=+2.7181488E+0
k=0100000    result=+2.7184156E+0
k=0500000    result=+2.7189322E+0

E>fset MIL <---------- MILフラグのセット
set MIL flag

E>test1
selected MIL !!
k=0000010    result=+2.5937400E+0
k=0000050    result=+2.6915870E+0
k=0000100    result=+2.7048120E+0
k=0000500    result=+2.7156944E+0
k=0001000    result=+2.7167315E+0
k=0005000    result=+2.7184548E+0      MILの出力
k=0010000    result=+2.7153563E+0
k=0050000    result=+2.7219005E+0
k=0100000    result=+2.7217975E+0
k=0500000    result=+2.5898509E+0

E>fset IEEE <---------- IEEEフラグのセット
set IEEE flag

E>test1
selected IEEE !!
k=0000010    result=+2.5937428E+0
k=0000050    result=+2.6915858E+0
k=0000100    result=+2.7048109E+0
k=0000500    result=+2.7155337E+0
k=0001000    result=+2.7170494E+0
k=0005000    result=+2.7184610E+0      IEEEの出力
k=0010000    result=+2.7185957E+0
k=0050000    result=+2.7219350E+0
k=0100000    result=+2.7219107E+0
k=0500000    result=+2.7532434E+0

E>
```

図 9 出力例
Fig. 9 An example of output.

5.5 出 力 例

図 9 に新しい評価方式を用いた数値演算システムの使用例を示す．この例は自然対数の底 e の値を求めるプログラムである．図9より，一つの実行形式で，URR, IEEE 表現法，MIL-STD-1750A 表現法の3種類の表現法が適用されていることがわかる．

5.6 その他の特徴

現在，この新しい評価方式を用いた浮動小数点演算システムでは，種々の丸めを用いた演算ライブラリを提供している．現時点では，0捨1入 (RN)，$+\infty$ 方向の切り上げ (RP)，$-\infty$ 方向の切り捨て (RM) の3種類をサポートしている．丸めの指定は，使用する浮動小数点表現法の指定と同時に行う．

使用する浮動小数点表現法と丸めを設定するため，および，使用されている表現法と丸めの方式を知るために次のコマンドを用意している．

(a) 適用される浮動小数点表現法を設定するコマンド

(b) 現在，どの浮動小数点表現法が使用されているかを知るコマンド

6. プロトタイプの問題点と OS/o での実現に向けて

プロトタイプの大きな問題点は，文字列の形式で残されている定数を使用される度に浮動小数点表現法の内部表現に変換しているので，実行速度に悪影響を与えることである．

この欠点を少しでも補うために，種々の工夫が考えられる．例えば，実現しているすべての浮動小数点表現法の内部表現に変換し，それらをオブジェクト内に持つ方法も考えられる．しかし，この方法は4.2節で述べた方針(2)，(3)に反する．また，プロトタイプのように URR の内部表現と文字列定数の両者をオブジェクト内に持つと，URR 以外の表現法を用いた場合，上述の問題は解決されていない．

上述の問題を完全に解決するためには，プログラム中の定数を，ロード時に一度だけ内部表現に変換すればよいと考えている．この方式を実現するためには，OS/o リロケータブル・オブジェクト・モジュールに，文字列の位置とそれを変換して内部表現として埋め込むデータアドレスを示すレコードを設ければよい．そのため，OS/o 用ア

センブラ，リンカとローダの変更を行う．

次に，マルチタスクの環境における，ダイナミックな浮動小数点表現法の選択の問題について述べる．当研究室で開発した OS/*o* はトランスパレントな OS として，この機能の追加は容易である．そこで，OS に手を加え，浮動小数点表現法の選択を OS に委ねることを考えた．言い換えれば，浮動小数点評価用の OS を作成するのである．

具体的には，各タスクごとに浮動小数点表現法の選択用のフラグを設置する．タスクのロード時に，OS はそのフラグの値によって割込みベクタの値を書き換え，適用される浮動小数点表現法のライブラリの選択を行う方法が考えられる．

7. おわりに

CAT に浮動小数点演算機能を組み込んだ．このことにより，第1章で述べたように，新しい浮動小数点表現法の比較・評価を行う環境を与えること，そして各種アプリケーションに最適な表現法の選択が可能となるように複数通りの浮動小数点表現法を実現した．それは，パーソナルな計算機システムの時代を迎えた現在，浮動小数点表現法そのものを見なおそうという考えに基づいている．

そして，複数通りの浮動小数点表現法を CAT でサポートするために

（1）コンパイル時に表現法の選択を行う方法

（2）実行時に表現法の選択を行う方法

の両者を用いて実現した．（2）の方法は，主に効率良く各種浮動小数点表現法の比較を行うために考案した評価方式である．その実現によって一つの実行形式で複数通りの浮動小数点表現法の適用が可能となった．さらに，評価ばかりではなく，各アプリケーションを記述するユーザがプロトタイプを作成するとき，適した浮動小数点表現法の選択を，効率良く行うことが可能となった．

OS/*o* 上での評価環境の実現が重要な残された課題である．現在は，URR のハードウェア化と CAT によるその利用方式の検討を進めている．

謝辞 本研究の一部は文部省科学研究費（試験研究（I）「新しい数値表現法の数値計算に及ぼす影響についての研究」課題番号 58840001）の援助を受けた．そこでの研究会や冬のプログラミングシンポジウム等で京都大学・一松信教授，東京大学・伊理正夫教授，立教大学・島内剛一教授，早稲田大学・中島勝也教授，日立中央研究所・浜田穂積博士には有益な討論をしていただいた．ここに心から感謝の意を表す．

参考文献

1) 松井正一，伊理正夫：あふれのない浮動小数点表示方式，情報処理学会論文誌，Vol. 21, No. 4, pp. 306-313 (1980).
2) 浜田穂積：二重指数分割に基づくデータ長独立実数値表現法，情報処理学会論文誌，Vol. 22, No. 6, pp. 521-526 (1981).
3) 浜田穂積：二重指数分割に基づくデータ長独立実数値表現法 II，情報処理学会論文誌，Vol. 24, No. 2, pp. 149-156 (1983).
4) IEEE Computer Society Microprocessor Standards Committee Task P754: A Proposed Standard for Binary Floating-Point Arithmetic, Draft 8.0 of IEEE, *Computer*, Vol. 14, No. 3, pp. 51-62 (1981).
5) 高橋延匡：OS/omicron の設計思想，第29回情報処理学会全国大会論文集，pp. 339-340 (1984. 9).
6) 篠田佳博，藤森英明，中川正樹，高橋延匡：OS/*o* のツール・セット（4）—言語Cコンパイラー，第30回情報処理学会全国大会論文集，pp. 365-366 (1985. 3).
7) 屋代 寛，森 岳志，並木美太郎，中川正樹，高橋延匡：OS/omicron 用言語 C コンパイラ cat の開発—VAX/UNIX クロスシステムからの移行—，情報処理学会ソフトウェア工学研究会資料 48-1 (1986. 6).
8) Kernighan, B. W. and Ritchie, D. M.: *The C Programming Language*, Prentice-Hall, Englewood Cliffs (1978).
9) 中森眞理雄，森 岳志，高橋延匡：OS/omicron の第一版における初等関数サブルーチン開発の計画と現状，科学研究費補助金試験研究（I）“新しい数値表現法の数値計算に及ぼす影響についての研究”報告書，pp. 114-123 (1986. 3).
10) 森 岳志，中森眞理雄，高橋延匡：OS/omicron における浮動小数点方式に関する研究，科学研究費補助金試験研究（I）“新しい数値表現法の数値計算に及ぼす影響についての研究”報告書，pp. 78-99 (1986. 3).
11) 森 岳志，中川正樹，中森眞理雄，高橋延匡：OS/omicron における複数の浮動小数点方式のサポート，第33回情報処理学会全国大国論文集，pp. 325-326 (1986. 9).

(昭和62年11月17日受付)
(昭和63年6月24日採録)

森　岳志（正会員）

昭和36年生．昭和60年東京農工大学工学部数理情報工学科卒業．昭和62年同大学院工学研究科修士課程修了．計算機における数値表現法，コンパイラの研究に従事．同年日本電気(株)入社．

中川　正樹（正会員）

昭和52年東京大学理学部物理卒業．昭和54年同大学院修士課程(物理学)修了．同在学中英国Essex大学留学 (M. Sc. in Computer Studies)．昭和54年より東京農工大学工学部数理情報助手．オンライン手書き文字認識，計算機システムソフトウェア，日本語文書処理の研究に従事．

高橋　延匡（正会員）

昭和8年生．昭和32年早稲田大学第一理工学部数学科卒業．同年(株)日立製作所中央研究所入社．HITAC 5020モニタ，TSSの開発に従事．昭和52年より東京農工大学工学部数理情報工学科教授．オペレーティング・システム，日本語情報処理，パターン認識の研究に従事．電子情報通信学会，ソフトウェア科学会，計量国語学会，ACM各会員．理学博士．

中森眞理雄（正会員）

昭和46年東京大学工学部計数卒業．昭和52年同大学院修了．工学博士．同年東京農工大学工学部数理情報講師．現在助教授．昭和60年度文部省在外研究員として西ドイツ，ボン大学で研究．組合せ的数理計画問題，グラフ・ネットワークフロー，アルゴリズム・データ構造，地理情報処理，集積回路の配置配線設計，CAI，ソフトウェア開発手法などの研究と教育に従事．

訂　　正

本誌第29巻第8号（1988）pp. 807-814に掲載されました森岳志氏ほかの論文「各種浮動小数点表現法の評価方式の実現」の中に誤りがありましたので，以下のように訂正いたします．

訂正箇所	誤	正
（1） 808ページ右段 4 行目	（1） URP[2),3)](図 1）	（1） URR[2),3)](図 1）
（2） 810ページ右段17行目	2. 538 μs,	253. 8 μs,
（3） 810ページ右段18行目	2. 077 μs	207. 7 μs
（4） 812ページ右段32行目	4. 2節で述べた方針	5. 2節で述べた方針