ＡＩＰ－ＬＩＳＰ：（３）数値演算の実現

7P-3

河込和作°　新堀哲之°　小林勉°°
°（株）東芝　°°日本ビジネスオートメーション（株）

0．はじめに

Ｃｏｍｍｏｎ　Ｌｉｓｐは、数値演算においても、豊富なデータ型と、柔軟で強力な関数を用意している。

我々は、ＡＩＰ－Ｌｉｓｐに、Ｃｏｍｍｏｎ　Ｌｉｓｐ準拠の数値演算機構を組み込んだ。本報告では、その実現手法と、評価結果について報告する。

1．数値演算機構の設計方針

ＡＩＰ－Ｌｉｓｐの数値演算機構を設計する際に重視したのは、以下のような点である。

1）　高速性の維持

ｆｉｘｎｕｍ，ｓｈｏｒｔ－ｆｌｏａｔ演算性能への悪影響を最小化するようにｂｉｇｎｕｍ、ｒａｔｉｏ等の演算をサポートする。

2）　柔軟性の確保

数値演算実行時ルーチンにおいて、モジュール化を徹底し、数値データの内部表現、演算アルゴリズム等を、変更しやすくする。また、ｂｉｇｎｕｍやｒａｔｉｏをサポートしないシステムを容易に構築できるようにする。

2．数値演算実行時ルーチンの構成

ＡＩＰ－Ｌｉｓｐの数値演算実行時ルーチンは、図１に示すように、階層的に構成されている。

これらは、例外を直接処理する部分がアセンブラ、一部の超越関数の本体がＣ言語で記述されている他は、すべてＬｉｓｐで記述されている。

上位レベルのモジュールは、実際のデータ表現と独立に記述可能な部分であり、大部分のＣｏｍｍｏｎ　Ｌｉｓｐ関数がこれに属する。上位レベルの関数では、ジェネリック演算を利用して、簡潔で保守性の良い記述を指向している。

一方、下位レベルのモジュールは、ｂｉｇｎｕｍ演算やｓｈｏｒｔ－ｆｌｏａｔの解析など、実際の（タグ付き）データ表現を陽に取り扱う部分である。下位レベルの関数では、ジェネリックな演算を一切使用せず、常にｆｉｘｎｕｍの結果を生じるような、専用の組み込み関数だけによる記述をおこなっている。こうすることにより独立性を高め、上位モジュールとは別の意味で保守性を向上させている。

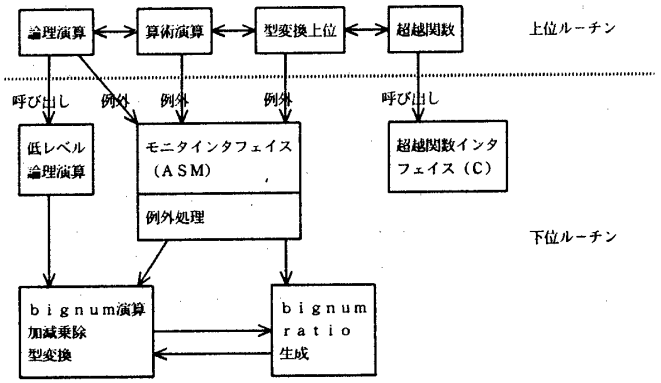

図１　数値演算ソフトウェアの構成

3．データ表現

ＡＩＰは、Ｌｉｓｐ／Ｐｒｏｌｏｇ等を効率的にサポートするため、４ビットのタグフィールドを持っている。ＡＩＰ－Ｌｉｓｐでは、数値データを表わすタグ値として、ｆｉｘｎｕｍ、ｓｈｏｒｔ－ｆｌｏａｔ、その他の数値、の３種類を割り当てている。

現在のＡＩＰ－Ｌｉｓｐは、その他の数値としてｂｉｇｎｕｍとｒａｔｉｏをサポートしている。これらは、実体の先頭にあるサブタイプコードによって識別される。

ｆｉｘｎｕｍとｓｈｏｒｔ－ｆｌｏａｔについては、即値表現（２８ビット幅）を採用しており、演算速度の向上と空間の節約に寄与している。

ｂｉｇｎｕｍの内部表現として、ベクタ型の表現を採用し、高速な処理を可能とした。ｂｉｇｎｕｍを表すベクタの各要素は全てｆｉｘｎｕｍで、最上位のワード以外のＭＳＢが０になっている。最上位ワードは符号が拡張されていて、全体としては２７ビット幅の２の補数表現と符号ビット（最上位ビットのＭＳＢ）という構成になっている。

Numerical operations of AIP-LISP
Kazusaku KAWAGOME, Satoshi SHIMBORI(TOSHIBA Corp.),
Tsutomu KOBAYASHI(JAPAN BUSINESS AUTOMATION Corp.)

4．数値演算例外処理機構

ＡＩＰは、ジェネリック演算を、ハードウェア的にサポートする命令を持っている。それらの命令では、ハードウェア的にオペランドの型タグを検査し、ｆｉｘｎｕｍまたはｓｈｏｒｔ－ｆｌｏａｔであればファームウェアによる演算が実行され、それ以外の型であれば例外が発生するようになっている。

また、オーバフロー等によって、ｆｉｘｎｕｍやｓｈｏｒｔ－ｆｌｏａｔでは表現できない結果が生じた場合にも例外が発生し、処理が例外処理ソフトウェアに引き継がれる。

数値演算例外処理関数は、例外及び演算ごとに用意されていて、オペランドの型の組合せによって、適切な低レベル関数を選択し演算を行なう。

例えば、ｆｉｘｎｕｍとｂｉｇｎｕｍの加算が行われた場合、ジェネリックな加算命令は加算型違反例外を発生し、モニタを通じて、例外処理関数に制御が渡る。例外処理関数は、ソフトウェア的に、オペランドの型の組合せを調べ、最終的にｆｉｘｎｕｍとｂｉｇｎｕｍの加算を行う低レベル関数を呼び出す。

その後、例外処理機構は、特殊な関数復帰処理を行い、低レベル関数が返す結果を適切なレジスタに置いて、例外を発生した次の命令から実行が再開されるようにする。

ＡＩＰ－Ｌｉｓｐでは、ｂｉｇｎｕｍやｒａｔｉｏはジェネリック演算の例外処理を通してのみ生成されるようになっている。

そのため、アプリケーションによっては必要なｂｉｇｎｕｍやｒａｔｉｏを持たない構成のシステムにも、例外処理ルーチンを、ｂｉｇｎｕｍ、ｒａｔｉｏの生成を行なわないものに差し替えるだけで対応することができる。

表１　ｆｒｐｏｌｙの結果

r	n	ＡＩＰ－Ｌｉｓｐ	ＣＬＩＳＰ
r	１０	０．１４	１．３７
r２	１０	２．２０	５．２７
r３	１０	０．１７	２．７３
r	１５	１．１５	１１．４２
r２	１５	２９．２２	７０．４５
r３	１５	１．０７	１７．４３

単位は秒、ＡＩＰのクロックは１１０ｎｓ

ＣＬＩＳＰの測定には、実装メモリ８ＭバイトのＡＳ３１６０を使用

5．評価

ＡＩＰ－Ｌｉｓｐ数値演算機構の性能を評価するため、良く知られた、Ｇａｂｒｉｅｌベンチマークの課題のひとつ、"ｆｒｐｏｌｙ"を用いて性能測定を実施した。その結果を表１に示す。

結果を見ると、ｆｉｘｎｕｍ、ｓｈｏｒｔ－ｆｌｏａｔ演算と比較して、ｂｉｇｎｕｍ演算の成績が悪いが、これは前述したような方式でジェネリック演算をサポートしているために、ｂｉｇｎｕｍ演算のたびに例外が発生するオーバヘッドがあること、および、ｂｉｇｎｕｍ演算ルーチンのチューニングがまだ不十分であることに起因すると考えられる。

6．まとめ

ＡＩＰ－Ｌｉｓｐ上に、Ｃｏｍｍｏｎ　Ｌｉｓｐ仕様に準拠した数値演算ソフトウェアを実装し、性能評価を行なった。その結果、当初の目的である、「通常の演算でオーバヘッドの少ないジェネリック演算」を実現することができたことを確認した。その反面、大きなｂｉｇｎｕｍ演算の性能が不十分であるとも判明した。

今後の課題は、より精密な性能測定と、その結果に基づく性能向上のための改良である。

参考文献

［１］河込　他："ＡＩＰ－ＬＩＳＰの実現（Ⅱ）"　情報処理学会第３６回全国大会（１９８８）

［２］五十嵐　他："ＡＩワークステーション（ＷＩＮＥ）の改良"　情報処理学会第３７回全国大会（１９８８）

［３］G.L.Steele Jr.,"Common Lisp the Language", Digital Press,1984

［４］D.E.Knuth,"The Art of Computer Programming Vol.2:Seminumerical Algorithms",Adison-Wesley,1969

［５］J.L.White,"Reconfiguiable,Retargetable Bignums:A Case Study in Efficient,Portable Lisp System Building",Proc.ACM Symposium on Lisp and Functional Programming,Aug,1986

［６］R.P.Gabriel,"Performance and Evaluation of Lisp Systems",MIT Press,1985