計算機アーキテクチャ 83－7
（1990．7．18）

スーパーコンピューテング用大粒度LISPコンパイレーション

深瀬政秋(*)　　中村維男(**)
*－－－東北大学電気通信研究所
*－－－東北大学工学部

本論文では、汎用パイプラインマシン(GPPM)を用いて、完全２進木構造を有するデータを処理する。GPPMとは、非数値処理と大粒度演算パイプラインの並列処理を一台の計算機で行なうものである。本論文の目的は、再帰的に増設される並列演算大粒度汎用パイプライン(PAP)を用いて、GPPMのコンパイラをベクトル化なしに作成することである。我々の提案するコンパイレーション手法に関して、構文解析器と中間言語発生器が、LISPのS式を用いて効果的に表現される。

LISP-COMPILATION TOWARD SUPERCOMPUTING WITH LARGE GRANULARITY

Masa-aki Fukase*　　and　　Tadao Nakamura**

* --- Research Institute of Electrical Communication, Tohoku University, Sendai 980, JAPAN
**--- Department of Mechanical Engineering, Faculty of Engineering, Tohoku University, Sendai 980, JAPAN

In this paper, a general-purpose pipeline machine (GPPM) is employed for processing a complete binary tree-structured data. The GPPM is constructed primarily for non-numeric and parallel pipelined arithmetic processings with large granularity in a computer environment. The objective of this paper is toward building a compiler of the GPPM without vectorization by using a recursively extended parallel arithmetic general-purpose pipeline (PAP) with large granularity. In view of compilation with respect to our proposal, a syntax analyzer and an intermediate-code generator are written in LISP in order to make the two be active in S-expression.

1. 緒言

計算機は、一方では数値計算の高速処理を目指し、他方では記号処理の機能強化を目指して、初期の頃から開発が行なわれてきている。従って、今日の計算機が取り扱う対象は、主として数値情報と非数値情報のどちらか一方に限られる。しかし、両方の情報を一台のコンピュータで高速処理する必要性が最近急速に高まっている。

従来のスーパーコンピュータは、ソフト的にはベクトル化率を上げることにより、超高速科学演算処理を実現した。非数値処理であるバックトラック計算法に対して、CRAY-1が効果的に導入された[1]。Nクイーン問題に対するバックトラック計算法を、Intel iPSCで処理する場合の性能評価が行なわれた[2]。

知識工学の発達と共に、記号処理向きスーパーコンピュータが開発されてきた。この問題に対しては、並列処理[3]や並列性[4]のアルゴリズムに関するソフトウエアの研究がある。しかし、コネクションマシン[5]、或いは第５世代コンピュータ[6]などに、非数値情報と数値情報の超高速処理を行なわせることは困難と思われる。

本論文は、前置プロセッサと大粒度並列演算パイプラインで構成される汎用パイプラインマシン(GPPM)を想定し、そのコンパイラについて述べる。前置プロセッサは数式の構文解析を行なうコンパイラを具備し、大粒度並列演算パイプラインを制御する。演算パイプラインは、四則演算機能を有するセグメントで構成される。多種類の２項演算数式の計算を、GPPMで並列処理する場合を説明する。

２節では、多種類の２項演算数式に対するLISPの構文解析器について述べる。また、それらのデータをベクトル化せずに大粒度並列演算パイプラインで処理する形態について説明する。３節では、並列演算パイプラインのための中間言語発生器について述べる。以上のGPPMの方式について、４節で議論する。

2. 再帰的構文解析と大粒度並列演算パイプライン

従来のスーパーコンピュータによる数式のベクトル処理を図１に示す。計算の木の各ノードは、相当する小粒度の演算パイプラインの機能に対応する。演算パイプラインのチェイニングは、木の探索を並列処理することに対応する。

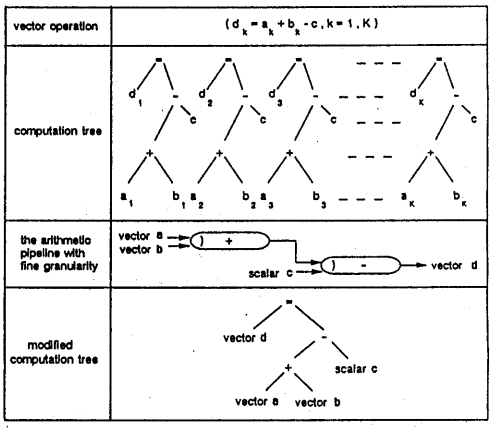

図１　ベクトルマシンの数式処理

２項演算数式の例として、

((a*(b#c))+(d#(e-f)))*(((g+h)+i)-(j-(k+l)))　　(1)

を考える。ポーランド記法は演算順序を一意に決定するので、並列処理には向いていない。計算の木では、演算順序が一意に決まらないので並列処理に向いている。この方法でパイプライン処理をするための規則性を考える。

LISPのリスト処理機能を使うと、２項演算数式の階層的な分解が容易である。このことは並列パイプライン処理に非常に適している。式(1)は、

(*(+(*a(#bc))(#d(-ef)))(-(+(+gh)i)(-j(+kl))))　　(2)

なるS式で表わされる。図２に、このＳ式のリスト構造を示す。

定義１　Ｓ式の或る要素の左側に存在する、（と）の数の差を、その要素の「レベル」という。

定義２　Ｓ式の要素のレベルの最大(最深)値を、Ｓ式の「深さ」という。

Ｓ式に対応するリスト構造に対しても、レベルと深さが同様に定義される。図２の場合、右下にレベルを記入してある。このリスト構造の深さは４である。データ構造は階層的で再帰的である。要素は、下位のレベルから上位のレベルの順序で評価されなければならない。同じレベルの全ての要素は、並列

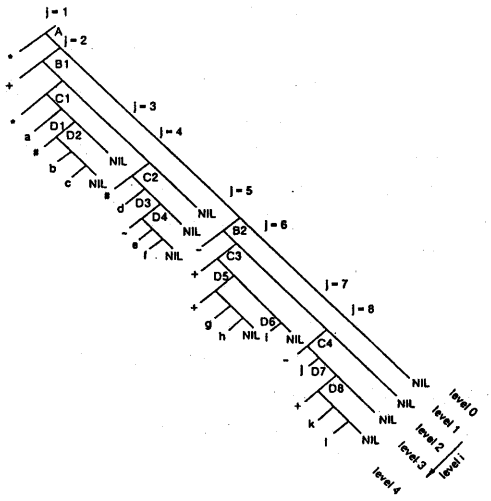

図２　式(2)のリスト構造

評価が可能である。

(2)のＳ式は、四則演算の能力を有するセグメントで構成され、一般の演算パイプラインよりも粒度の大きい演算パイプラインで処理できる。

定義３　セグメント毎に複数の演算機能を有する演算パイプラインを、「大粒度演算パイプライン」という。

図３に大粒度演算パイプラインの概念を示す。このパイプライン１本で多種類の２項演算数式を評価できる。図３は、汎用パイプライン[7]の最も簡単な例である。汎用パイプラインは、MISD型の最初の構築とみなされている。各セグメントの機能は、そこで処理されるデータに付随するプログラムにより供給される。その性能を評価することにより、汎用パイプラインの有効性が示された[8]。

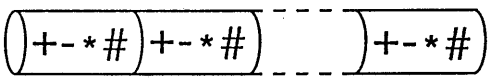

図３　大粒度演算パイプライン

任意のレベルｉの要素は、演算子とオペランドとレベルi+1の要素を再帰的に含む要素からなる。演算子は、それを値の一部として含みレベルi-1に属する要素の評価に使われる。レベルｉで評価される要素は、リストとオペランドである。従って、図２のリスト構造は分割攻略を並列に進めることが可能である。

定義４　複数のパイプラインで構成されるひとつのパイプラインシステムを、「並列パイプライン」という。

定義５　大粒度演算パイプラインで構成される並列パイプラインを、「大粒度並列演算パイプライン(Parallel Arithmetic Pipeline with large granularity:PAP)」という。

各々の数式の再帰的分解によって得られる要素は、PAPで処理することに適している。図４では、図２の階層的なアトムを相当する個々のパイプラインに振り分けている。１個のセグメントが１個のアトムを評価する。後のセグメントで評価されるアトムは、前のセグメントを素通りする。後のセグメント程、データサイズは小さい。

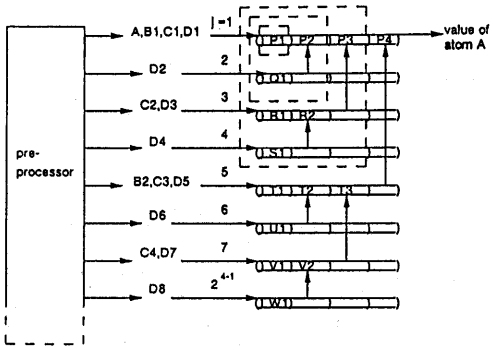

図４　PAPによる、図２のリスト構造の処理

図４では最初に、j=1なるパイプラインの最初のセグメントP1が使われる。j=2なる他のパイプラインの最初のセグメントQ1が使われた後で、これらの出力はj=1なるパイプラインの２番目のセグメントP2に入力される。P1、Q1とP2の間の接続形態は、R1、S1とR2の間の接続形態と同じである。このようにして、個々のパイプラインのセグメントはPAPの中で再帰的に活用される。図５では、図４の動作の軌跡を時間空間座標に示した。$n_1\tau$、$n_3\tau$は、各々＋と＊、＃の演算時間である。

図４のPAPは、任意の数式を処理できる。その特徴は、Q1-P2間、S1-R2間、R2-P3間などの接続が固定であることと、２番目以降のパイプラインは、後段の

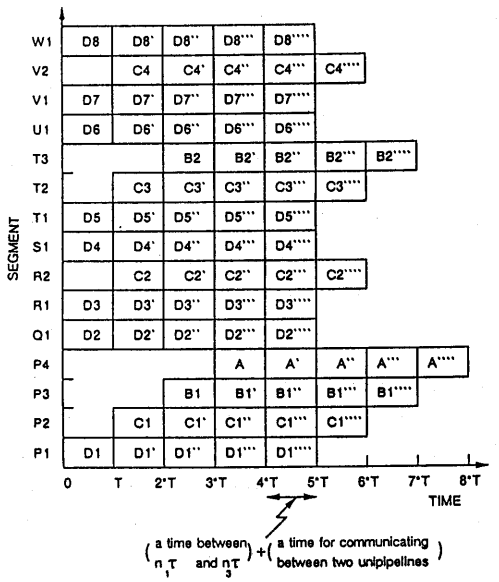

図５　図４の時間-空間チャート

セグメントを使わないことである。以下、PAPによる多種類の２項演算数式のＳ式の評価について述べる。用いられる主要な記号とそれらの意味を、表1にまとめる。

notation	meaning
T	the segment time.
N	the total number of the segment of the first unipipeline of a PAP.
j	the number of each unipipeline of a PAP.
J	the total number of the unipipeline of a PAP.
i	the level number of an element in an S-expression or corresponding list structure.
I	the depth of the S-expression or corresponding list structure.
l[j]	the level of the atom pushed just prior to the number of a NEBO in the jth stack.
j_{pop}	the number of the unipipeline satisfying the condition that its input atom has the value whose 3rd element is the input atom of the j+1th unipipeline.
$i_{pop}[j_{pop}]$	the atom's number counted from the top atom, which are stored in the j_{pop} th stack.

表１　本論文で用いる主要な記号の意味

命題１　再帰的に増設されたPAPは、Ｉ≦Ｎなる種々の２項演算数式のＳ式の評価を可能とする。

証明は、完全２進木に対応するリスト構造の基本原理より明かである。完全２進木のリスト構造は、再帰的である。一番深いレベルの要素は１個の演算子と２個のオペランドからなる。その上の任意のレベルの要素は、１個の演算子とその下のレベルで評価される２個の要素からなる。全ての数式のＩがＮに等しい時、PAPの稼働効率は一番良い。図５はこの場合に対応している。

任意の２項演算数式を数値処理する場合、レベルｉにおける評価の対象は、先にも述べた様に演算子以外の要素である。再帰的に増設されたPAPは、この問題に対して十分な本数のパイプラインと、適切な接続法を有していることが導かれる。

命題２

$$J=2^{(N-1)} \qquad (3)$$

は、PAPを再帰的に増設するための必要条件である。

命題３　２項演算数式のＳ式のレベルｉに属する演算子でない要素の個数は、最大2^{i}個である。

命題2、3の証明は省略する。命題１より、再帰的に増設されたPAPは、Ｎ以下の任意の深さを有するリスト構造に対処できることが明かである。I=Nの場合、命題2、3より、演算子を除くレベルI-1の各要素が個々のパイプラインの一番目のセグメントに割り付けられる。I-2と０の間のレベルの演算子以外の要素は、２番目以降のセグメントに過不足なく割り付けられる。I<Nの場合は、後段のセグメントと番号の大きいパイプラインは使われない。従って、本論文で提案する方式は、マッピング問題を完全に解決しているとは言えない。逆に、この問題に対する対策はコンパイルの段階で施され、実行の段階ではプロセスマイグレーション[3]のような余分な操作は不要である。

3.　コンパイレーションアルゴリズム

この節では、PAPのコンパイレーションアルゴリズムについて述べる。このアルゴリズムを実行する専用の前置プロセッサを設ける。

定義６　次に述べる(i)、(ii)、(iii)の処理を、コンパイル即実行方式で行なうプロセッサを「前置プ

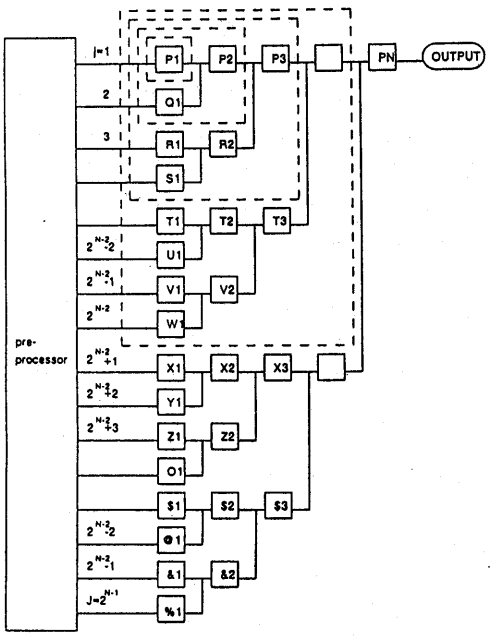

図６　PAPの機能表示

ロセッサ」という。(i)２項演算数式のＳ式の要素を分解し、機械語にコンパイルする。(ii)(i)のデータを個々のパイプラインに振り分ける。(iii)PAPの動作を制御する。

定義７　前置プロセッサと再帰的に増設されたPAPから構成されるコンピュータを、「汎用パイプラインマシン(General-Purpose Pipeline Machine:GPPM)」という。

図６はGPPMの機能表示を示す。四角は、大粒度のプロセッサである。点線で囲った部分は、PAPを再帰的に増設する際の単位である。GPPMの構成はMIMD方式である。図７は前置プロセッサのコンパイルのアルゴリズムを示す。

図７においては、最初の構文解析で、処理対象の数式を再帰的なＳ式に分解する。個々のパイプラインの入力アトムを格納するスタックを、パイプライン毎に用意する。ｊ番目のパイプラインの入力アト

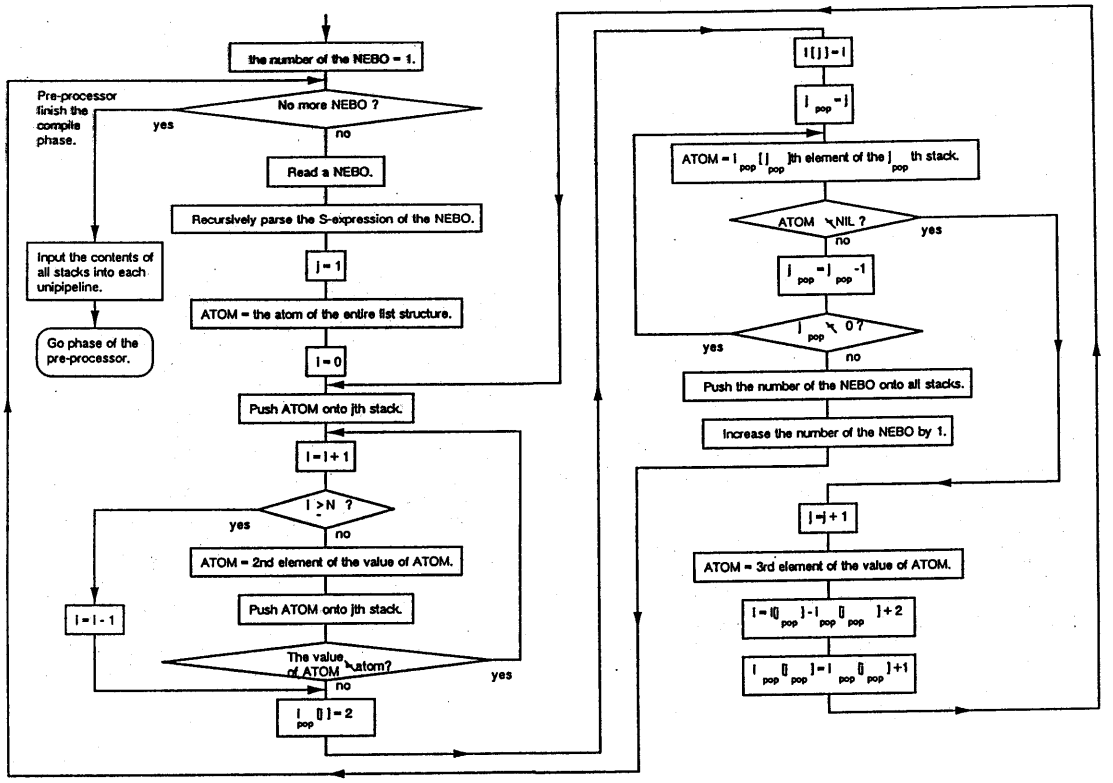

図７　LISPコンパイレーションのアルゴリズム

ムを、j番目のスタックに格納する。格納するアトムの値がアトムである場合、その時点でそのスタックへの格納を終了する。もし格納されるアトムのレベルがNに達っする場合は、命題1から明かなようにその数式は処理不可能である。しかし、図6はウエハスケールのASICに向いている。そのような場合はNは十分大きいと考えてよいので、上述のことは問題にならない。

このようにして、階層化したS式の全ての要素を個々のパイプラインに割り振る準備が済んだところで、1本の数式に対する構文解析とPAPの制御の準備は終わる。その数式に用いた全てのスタックに、数式を区別するためのマーカを格納する。以上の処理の中に、式(3)の関係が暗黙のうちに含まれている。引き続くすべての数式に対して、以上の手続きを完了したところで、前置プロセッサのコンパイル段階は終わる。

前置プロセッサのコンパイル段階で作られる全てのスタックの内容は、基本的に図8のようになる。前置プロセッサの実行段階では、これらのスタックの内容を対応する個々のパイプラインに流し込む。同時に、PAPの各セグメントに配置されているプロセッサの実行を制御する。数式番号は、制御情報として使用される。2つの制御情報の間にあるアトムを数値演算情報とみなす。

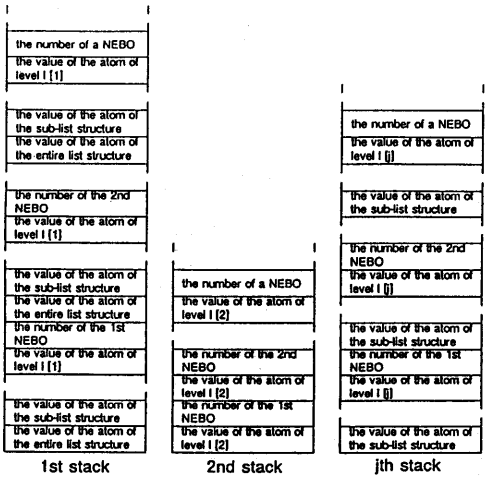

図8　図7で使用するスタックの構造

4. 結言

図6の構成はトリーマシン[9]と類似しているが、通信方式が異なる。トリーマシンは、制御プロセッサをその根に配置する。通信の流れはトップダウン方式である。一方図6の場合，前置プロセッサが制御プロセッサに相当するので、通信の流れはボトムアップ方式である。

図4の矢印はデータの流れを示しているので、図6はデータフローマシン[10]にも類似している。しかし、図6の並列システムがパイプライン動作をしていることは、図5より明かである。図6において各セグメントに配置されるプロセッサを制御しているのは、前置プロセッサである。

演算命令に関しては、本論文の方式はRISC (Riduced-Instruction-Set Computing)の思想に反している。しかし、図6に示すPAPの構造は規則的な超並列性を有する。このような構造はウエハスケールを指向したASICに向いている。

本論文は、記号処理とPAPの組合せで構成されたGPPMのコンパイルアルゴリズムに関して述べた。それは、形式が不揃いな大量の2項演算数式を、ベクトル化せずにコンパイル即実行処理する。この方式は、超並列な四則演算回路のマッピング問題に対する静的な解決法である。

引用文献

[1] L. Thiel et al., ICS87, vol. III, pp. 92-99, 1987.
[2] R. T. Mraz et al., ibid., pp. 82-91, 1987.
[3] R. Chowkwanyun et al., Parallel processing for supercomputers & artificial intelligence, McGraw-Hill Pub. Co. , pp. 325-366, 1989.
[4] R. H. Halsted, Jr., Computer, vol. 19, no. 8, pp. 35-43 IEEE, Aug. 1986.
[5] C. Stanfill et al., ibid. , pp. 494-500.
[6] K. Murakami et al., Computer, vol. 18, no. 6, pp. 76-92, IEEE, June 1985.
[7] T. Nakamura, Proc. of The IEEE Eighth International Computer Software & Applications Conference, pp. 408-414, Nov. 1984.
[8] H. Kobayashi et al., Trans. IECE Japan, vol. J68-D, no. 10, pp. 1744-1752, Oct. 1985.
[9] D. E. Shaw, 9th Int. Joint Conf. on Artificial Intelligence, pp. 61-72, 1985.
[10] Speed Issue on Data Flow Systems, Computer, vol. 15, no. 2, pp. 10-69, IEEE, Feb. 1982.