数理解析研究所講究録
第514巻 1984年 150-172

ベクトルプロセッサのソフトウェア技術
——ＦＡＣＯＭ　ＶＰソフトウェア——

富士通㈱　　神谷　幸男（ Sachio　Kamiya ）

【１】はじめに

ベクトルプロセッサを有効に使用するためには，解くべき問題に内在する並列性を最大限引き出す必要がある。しかし，問題をコード化してゆく過程で，問題自身が持つ並列性は，種々の要因により失われてゆく[1]。ここでは，プログラムのコード化の過程と，その過程における並列性の減少要因について考えてみる。また，並列性減少要因の一つとして挙げられるプログラミング言語について，実例を基に，それらの得失について考えてみる。更に，現在主流となっている自動ベクトル化の方式を採用しているＦＡＣＯＭ　ＶＰシステムのソフトウェア概要を紹介しながら，ベクトルプロセッサを有効に使用するためのプログラミング方法について述べる。

【２】プログラムのコード化の過程と並列性減少要因

プログラムのコード化の過程は，図１のように考えることができる。また，コード化の過程で発生する並列性の減少要因としては，次のものが考えられる。

(1)　問題（物理／数学モデル）→　計算スキーム

解こうとする問題から計算スキームという離散化モデルへの変更の過程であり，この段階で並列性の大きさが確定する。

(2)　計算スキーム　→　アルゴリズム

アルゴリズムの選択と並列性の関係は，非常に深い。例えば，本質

的に並列処理に向いていない回帰演算をベースにした数値計算アルゴリズムを使用した場合（ガウス消去法を用いた三項方程式の解法等）では，現状でのベクトルプロセッサの高速化手法である同一の演算を複数のデータに対して同時に行うという方式（ＳＩＭＤ：Single Instruction Multiple Data stream）に向いていないため，この過程で大幅に並列性が失なわれることになる。

(3) アルゴリズム → プログラム

この過程では，使用するプログラミング言語によって，並列性が減少することがある。これは，使用する言語に並列処理の表現方法があるかないかによる。例えば，標準ＦＯＲＴＲＡＮ言語では並列性が直接には表現できず，並列性の度合いとしては１（すべて逐次処理）に減少することになる。

(4) プログラム → 実行

プログラムから実行の過程では，コンパイラ及びハードウェアのアーキテクチャによる制約を受ける。例として，標準ＦＯＲＴＲＡＮ言語で書かれたプログラムをベクトルプロセッサ上で実行する場合を考えてみる。翻訳過程では，コンパイラ（この場合のコンパイラは，逐次処理として書かれたプログラムから自動的に並列処理可能部分を取り出してベクトル命令を生成するというベクトル化機能を持つ）のベクトル化の認識機能の度合によって並列性が抑えられてしまう。また，ベクトル命令を含むオブジェクトモジュールをベクトルプロセッサ上で実行する場合には，計算機自体が持つ並列性でプログラムの並列性が抑えられる。

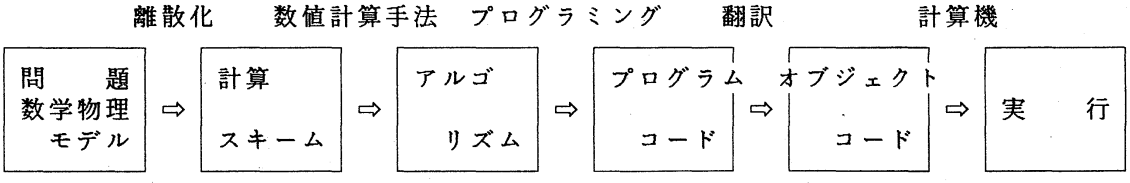


図１ プログラムのコード化の過程

これらのプログラムのコード化の過程を通じて，並列性を保持するための重要な鍵は，

・計算スキーム／アルゴリズムの選択

・プログラミング言語

である。前者の計算スキーム／アルゴリズムは，使用者の選択によって決まる要因であり，後者のプログラミング言語は，システムソフトウェアに依存する要因である。

ここでは，言語という観点からいくつかの例題を基に，並列性の保持と，それらの得失について考えてみる。

【３】ベクトルプロセッサを使用するプログラミング言語

科学技術計算の分野では，ＦＯＲＴＲＡＮ言語が多く使用されている。したがって，これまでに発表されたベクトルプロセッサを使用するための言語には，ＦＯＲＴＲＡＮ言語をベースにしたものが多い。これらを分類してみると表１のようになる。

表１　ベクトルプロセッサを使用する言語

形　式	言　語　仕　様	方　式
手続き向きプログラミング言語	標準FORTRAN 言語	自動ベクトル化コンパイラ方式
		ビルディング・ブロック方式
	FORTRAN ベース拡張言語	ベクトル（配列）処理言語
		アーキテクチャ指向言語
問題向きプログラミング言語		

【３.１】自動ベクトル化コンパイラ方式

この方式は，ベクトルプロセッサを使用する言語として標準ＦＯＲＴＡＮ言語を用い，コンパイラに並列処理性を認識させる方式である。この方式の歴史は古く，ILLIAC IV 用に開発されたIVTRAN，CRAY用のＦＯＲＴＡＮ言語ＣＦＴなどがその代表である。また，最近では，以下に示すいくつかのコンパイラが発表されている。

・ＦＯＲＴＲＡＮ77／ＶＰ[2]　（富士通）
・ＦＯＲＴ77／ＨＡＰ[2]　（日　立）
・ＦＯＲＴＲＡＮ77／ＳＸ　（日　電）

(1)　自動ベクトル化コンパイラ方式の実際例

ＣＲＡＹ用のＦＯＲＴＲＡＮであるＣＦＴ[3]は，標準ＦＯＲＴＲＡＮ言語で書かれたプログラムがベクトル化できるコンパイラである。例えば，下のようなＤＯループがプログラムにある場合，ＣＦＴはこれらの演算がベクトル演算として実行しても値が変らないことを認識し，自動的にベクトル命令を生成する。

（例）ベクトル化できるＤＯループ

```
FORTRAN プログラム
DO 100 I=1,N
  A(I)=X(I)*SIN(T(I))+Y(I)*COS(T(I))
  B(I)=X(I)*COS(T(I))-Y(I)*SIN(T(I))
100 CONTINUE
      ⇩
ベクトル演算
Ai=Xi*SIN(Ti)+Yi*COS(Ti)   ,i=1,2,...,N
Bi=Xi*COS(Ti)-Yi*SIN(Ti)   ,i=1,2,...,N
```

科学技術計算のプログラムでは，ＤＯループ中にＩＦ文などの制御文を含むことが多いが，ＣＦＴでは，制御文の並列処理用に組込み関数を用意して，解決を図っている。以下に制御文を含むＤＯループの例を示す。

（例）制御文を含むＤＯループの例

標準FORTRAN プログラム

```
      DO 100 I=1,N
        IF(L(I)) THEN
          X(I)=A(I)*B(I)
        ELSE
          X(I)=A(I)**2
        ENDIF
  100 CONTINUE
```

⇨

ＣＦＴプログラム

```
        DO 100 I=1,N
          X(I)=CVMGT(A(I)*B(I),
     1         A(I)**2,L(I))
  100 CONTINUE
```

(2)　自動ベクトル化コンパイラ方式の得失

自動ベクトル化コンパイラ方式の利点としては，

・使用者が慣れた言語でプログラミングできること
・これまでに貯えられてきたプログラム財産が，そのまま利用できること
・プログラムの互換性を保つことができ，流通ソフト用の言語としても利用できること

などが挙げられる。

逆に，欠点としては，並列性がコンパイラによって認識されるため，プログラムが持つ並列性がすべて認識されるかどうかは，コンパイラの認識技術に依存してしまうことである。即ち，認識レベルが低ければ，並列性を充分引き出すことができない。

【３.２】ビルディング・ブロック方式

この方式は，あらかじめ並列処理向きに作成されたライブラリを積み重ねてプログラム全体の並列性を上げようとする方式である。例えば，標準ＦＯＲＴＲＡＮ言語のＣＡＬＬ文を用いて，システムが提供する並列処理ライブラリや使用者自身が作成したライブラリを引用する。このような方式のライブラリで，発表されているものには，次のものがある。

・ＣＲＡＹＰＡＣＫ[4]　（ＢＣＳ社）
・ＡＰＭＡＴＨ[5]　（ＦＰＳ社）
・ＳＳＬⅡ／ＶＰ[6]　（富士通）
・ＭＡＴＲＩＸ／ＨＡＰ[2]　（日　立）

(1)　ライブラリの内容

ライブラリの中に含まれる内容としては，ベクトル演算／マトリックス演算といった比較的単純なライブラリから，ＦＦＴ，ソートといった算法の核となる部分を集めたもの，更には，固有値問題，常微分方程式などの解法レベルの機能を集めたものまである。

(2)　ビルディング・ブロック方式の得失

ビルディング・ブロック方式の利点は，並列度の高いライブラリが利用可能なため，比較的容易に高い並列性を持ったプログラムが作成できることである。

欠点としては，利用しようとするライブラリによっては，

・解決しようとする問題に適合したライブラリが無い場合は，その部分に対応する並列性が得られない
・ライブラリのインタフェースを合わすための補助操作が必要

などが挙げられる。

【3.3】ベクトル（配列）処理言語

ベクトル（配列）処理言語とは，ベクトルプロセッサを使用する言語としてベクトル処理や配列処理を陽に記述できるように言語仕様を定めた方式である。この種の言語としては，次のようなものが発表又は使用されている。

・ＡＰ－ＦＯＲＴＲＡＮ[7],[8]　（富士通）
・ＢＳＰ－ＦＯＲＴＲＡＮ[9]　（バロース）
・ＶＥＣＴＲＡＮ[1]　（ＩＢＭ）

更に現在では，ＡＮＳＩの規格（ＦＯＲＴＲＡＮ　８Ｘ）として，配列処理の記述が検討されている。

(1)　ベクトル（配列）処理言語の実際例

科学技術計算の問題を解くための言語仕様としては，少なくとも次の項目を含んでいる必要がある。点線の後の例は，富士通ＡＰ(Array Processor unit)-ＦＯＲＴＲＡＮ[7],[8]の場合の例である。

・ベクトル又は配列データが記述できる
　－配列全体　……　A(*,*)
　－部分ベクトル　……　B(1,*)
　－一定間隔ベクトルデータ　……　C(2*IX)
　－ランダムな位置にあるベクトルデータ　…　D(IL(*))
・ベクトル演算が記述できる
　－四則／比較／論理演算，組込み関数　……　C(*)=V(*)+U(*)
　－総和／内積演算，最大値／最小値検索　…　X=VSUM(A(*,*))
　－ベクトルデータの編集操作（圧縮等）　…　C(*)=GAT(L(*),D(*))

ここで，“＊”の文字は，配列寸法を宣言する文で指示したすべての要素の集合であることを意味する。IXは，インデックス変数といい，

要素位置の部分集合を表す変数である。また，VSUM及び GATは，ベクトル組込み関数であり，各々ベクトルデータの総和を求める関数，及びベクトルデータをある条件（前記の例では，配列Ｌの真／偽の値）によりデータを収集する関数である。

(2)　ベクトル（配列）処理言語の得失

ベクトル（配列）処理言語をベクトルプロセッサ用の言語として導入する方法は，最も素直な方法である。また，この方法では，使用者が直接並列性の記述ができるため，言語による並列性の減少といった問題は解決される。更に，ＤＯループという逐次処理用の手続きが不要になるため，プログラムの生産性が向上するという利点がある。

しかし，言語を新しくすることは，次の点で欠点となる。

・国際的な規格が今のところないため，互換性の面で問題となる
・財産として貯えられているプログラムは，そのままの形では使用できない

【３.４】アーキテクチャ指向言語

アーキテクチャ指向言語は，目標とする計算機のハードウェア・アーキテクチャを有効に使用できるように仕様を決めた言語である。科学技術計算分野では，ＦＯＲＴＲＡＮが多用されているため，これらの計算機を使用するための言語として，ＦＯＲＴＲＡＮ-like な記述を採用しており，形式的にはベクトル（配列）処理言語と同様な記述ができるようになっている。

この種の言語としては，

・ＣＦＤ[11]　　(ILLIAC IV)
・ＤＡＰ－ＦＯＲＴＲＡＮ[1]　　（ＩＣＬ社）

などが有名である。

(1)　アーキテクチャ指向言語の実際例

ILLIAC IV は，６４個（計画では２５６個）のプロセッサエレメント（ＰＥ）を並列に動作させる形のアーキテクチャを持ったシステムである。ＣＦＤは，このシステムを効率良く使用するために開発された言語であり，流体力学の問題を解くためのいくつかの拡張が採り入れられている。以下に一例を示す。

（例）　ベクトルデータの記述方法

A(*)　……　６４個の配列要素（各ＰＭにあるデータ）

A(*+1)　……　６４個の配列要素（各ＰＭにあるデータをサイクリックにシフトして定義／引用するための記述方法。この例では，プロセッサの番号が１つ大きいプロセッサにあるＰＭのデータを意味する）

（注）ＰＭ：プロセッサメモリ

(2)　アーキテクチャ指向言語の得失

(1)の例でも分るように，並列性表現は，完全に使用者に任されている。この点では，非常に優れたものであるが，プログラマは陽にハードウェア・アーキテクチャを意識しなければならないため，

・他のシステムでは使用できない

・プログラミングが難しい

という欠点がある。特定の目的に的を絞ったシステムに適用される方式である。

【３.５】問題向きプログラミング言語

プログラムのコード化の過程のより高位のところで言語を定義する方法である。即ち，数学モデル，計算スキーム（又はアルゴリズム名）

を記述することにより，プログラムの生産性及び並列性を向上させようというねらいを持つ言語である。

この方法では，ある特定の分野に限って，次に示すような言語が開発されている。

・ＥＬＬＰＡＣＫ[12]　　（Purdue大，楕円型ＰＤＥ）
・ＣＡＰ／ＰＦＤＲ[13]　　（電中研，拡散方程式）
・ＤＥＱＳＯＬ[14]　　（日立，偏微分方程式）

(1)　問題向きプログラミング言語の得失

問題向きプログラミング言語は，並列性が高く，かつ高い生産性を持つ方式である。しかし，この種のシステムでは，あらゆる問題が解決できるという汎用性には欠けており，

・適用範囲が限定されること
・プログラムの互換性が得られない

などの欠点がある。

【３.６】各プログラミング言語の評価

これまでに述べてきた各種のプログラミング言語の得失をまとめると表２のようになる。

各言語方式には様々な特徴があり，目的に応じて使い分けられているのが現状である。現在では，自動ベクトル化技術が向上したため，自動ベクトル化方式を採用するシステムが主流となっている。更に，自動ベクトル化方式は，

・プログラム財産がそのまま活かせる
・プログラムが書き易い

という利点を考えると，表２に示した方式の中では，今のところ最も良い方式であるといえる。

表2　各プログラミング言語の評価

言語方式	互換性 プログラム財産の利用	性能	適用範囲	生産性
自動ベクトル化コンパイラ方式	○	○	○	○
ビルディング・ブロック方式	△/○	○	△/○	○
ベクトル（配列）処理言語	× （→○）	◎	○	○
アーキテクチャ指向言語	×	◎	△/○	△/○
問題向きプログラミング言語	×	○	△/○	◎

◎：非常に良い　○：良い　△：やや悪い　×：悪い

【4】ＦＡＣＯＭ　ＶＰ用コンパイラＦＯＲＴＲＡＮ77／ＶＰ

前章でも述べたように，ベクトルプロセッサを使用するための言語方式としては，自動ベクトル化コンパイラ方式が主流を占めている。ここでは，自動ベクトル化コンパイラ方式を採用しているＦＡＣＯＭ　ＶＰシステムのコンパイラであるＦＯＲＴＲＡＮ77／ＶＰを例にとり，自動ベクトル化技術の現状を紹介する。

【4.1】自動ベクトル化技術[15],[16]

(1)　ベクトル化技術の基本

自動ベクトル化は，【3.1】(1)でも述べたように，ＦＯＲＴＲＡＮプログラム中に出現するＤＯループが，ベクトル演算として実行しても実行結果が変らないかどうかを調べ，実行結果が変らないことが分かれば，ベクトル命令に変換することである。

```
      DO 100 I=1,N                        DO 100' I=1,N
         A(I)=B(I)/C(I+1)                    A(I)=B(I)/C(I+1)
                                  ⇨
         C(I)=A(I)+D(I)                      DO 100" I=1,N
                                                C(I)=A(I)+D(I)
  100 CONTINUE                        100" CONTINUE
```

図２　ベクトル化できるＤＯループ

```
      DO 100 I=1,N                        DO 100' I=1,N
         A(I)=B(I)/C(I)                      A(I)=B(I)/C(I)
          a1        c1                100' CONTINUE
                                  ⇏
         C(I+1)=A(I)+D(I)                    DO 100" I=1,N
          c2       a2                         C(I+1)=A(I)+D(I)
  100 CONTINUE                        100" CONTINUE
```

ここで，a1,a2,c1,c2 はオペランドを意味する記号である。

図３　ベクトル化できないＤＯループ

ベクトル演算とは，一般的にＳＩＭＤ型の演算であり，同種の演算を複数のデータに対して同時に施すことである。従って，あるＤＯループがベクトル化できるか否かは，次のようなＤＯループの変形に対して結果が同じかどうかを調べることにより判断することができる。

例えば，図２においては，左のＤＯループと右のＤＯループは同じ結果を得ることができるので，この例はベクトル化ができるループである。一方，図３における配列要素Ｃ（３）の内容を調べてみると，左のＤＯループのc1の引用では，ＤＯの制御変数が２のときのc2の結果を用いているが，右のループの場合では，更新される前の値を引用しており，結果が異なる。従って，図３の左のＤＯループの例は，ベクトル化できないＤＯループといえる。

この二つのＤＯループの内容を見ても分かるように，ベクトル化できるか否かは配列要素データのアクセスの方法（定義／引用の関係：データの依存関係という）と大いに関係がある。データの依存関係は，配列添字の値の変化の様子に依存しているため，添字の形式を調べることによりベクトル化の可否が決定できる。

データの依存関係は矢印“→”を用いて表現することができる。二つのオペランドの定義ｆ及び引用（又は定義）ｇに対して，ｆを先行して実行しなければならないとき，ｆ→ｇと書くことにする。図３の例では，c2→c1という関係と，配列要素Ａ（Ｉ）の関係からa1→a2という関係がある。そして，この二つの関係から，どちらの文をベクトル演算として先に実行しても，結果が異なることになる。即ち，データの依存関係がループを形成している場合は，その部分がベクトル演算として実行できないことを示している。逆に，データの依存関係がループを形成していなければベクトル化ができる。図４は，多少複雑なＤＯループであるが，データの依存関係がループを形成していないためベクトル化ができたループの例である。

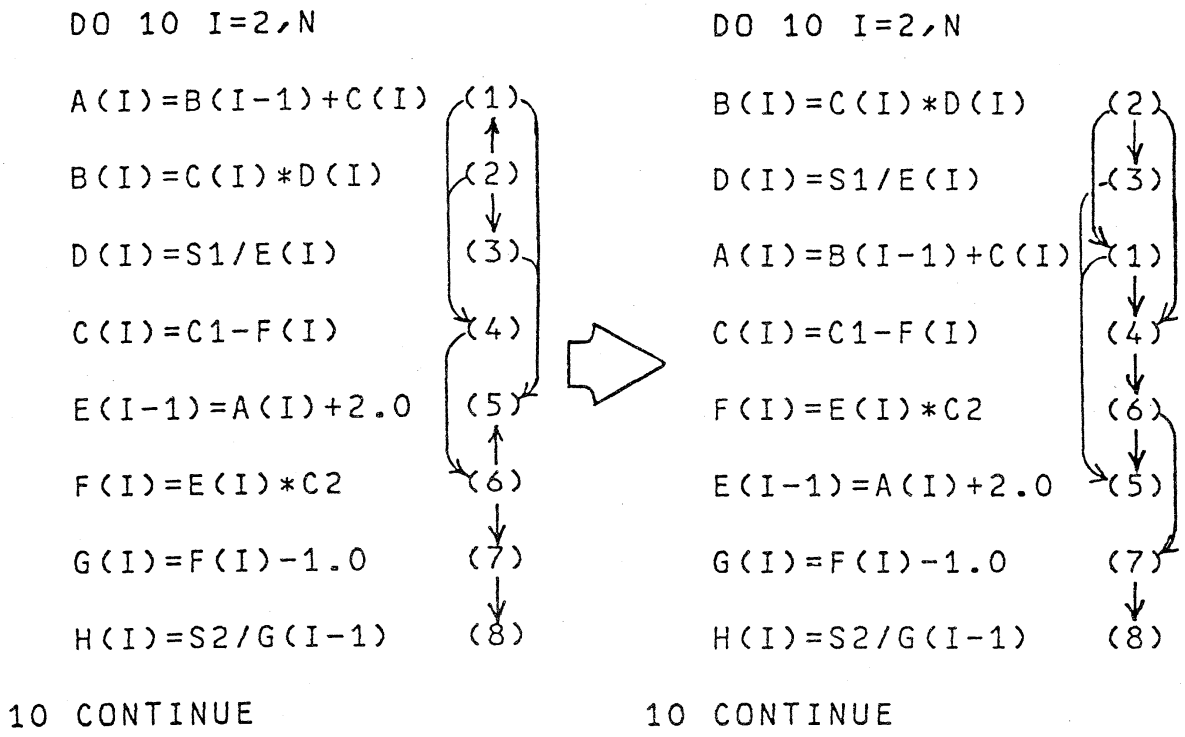


図４　データの依存関係を用いたベクトル化

(2)　ベクトルマクロ演算の自動ベクトル化技術

科学技術計算プログラムでは，ベクトルの四則演算で代表される単純な（各要素毎に並列に実行できる）演算ばかりでなく，ベクトルデータから目的とするデータを一つ作り出すという演算がある。この形の演算をベクトルマクロ演算という。ベクトルマクロ演算の代表的なものは，ベクトルデータの内積／総和を求める演算や最大値／最小値の検索などである。図５に内積を求める演算の例を示す。この形の演算は，同種の演算を同時に実行するというタイプの演算ではなく，演算順序を変更して，その中から並列性のある演算を取り出す必要がある。例えば，$X=X+\sum_{i=1}^{n} Ti$というベクトルデータ $\{Ti, i=1,2,\ldots,n\}$ の総和を求める演算では，図６のような演算順序の変更を行えばよい。
ＦＡＣＯＭ　ＶＰシステムでは，このような演算を並列に実行する命令を用意し，コンパイラでは，図５のような演算をパターンにより認識し，ベクトルマクロ演算の高速化を図っている。

```
      DO 100 I=1,N
        X=X+A(I)*B(I)
100   CONTINUE
```

図５　内積を計算するＤＯループの例

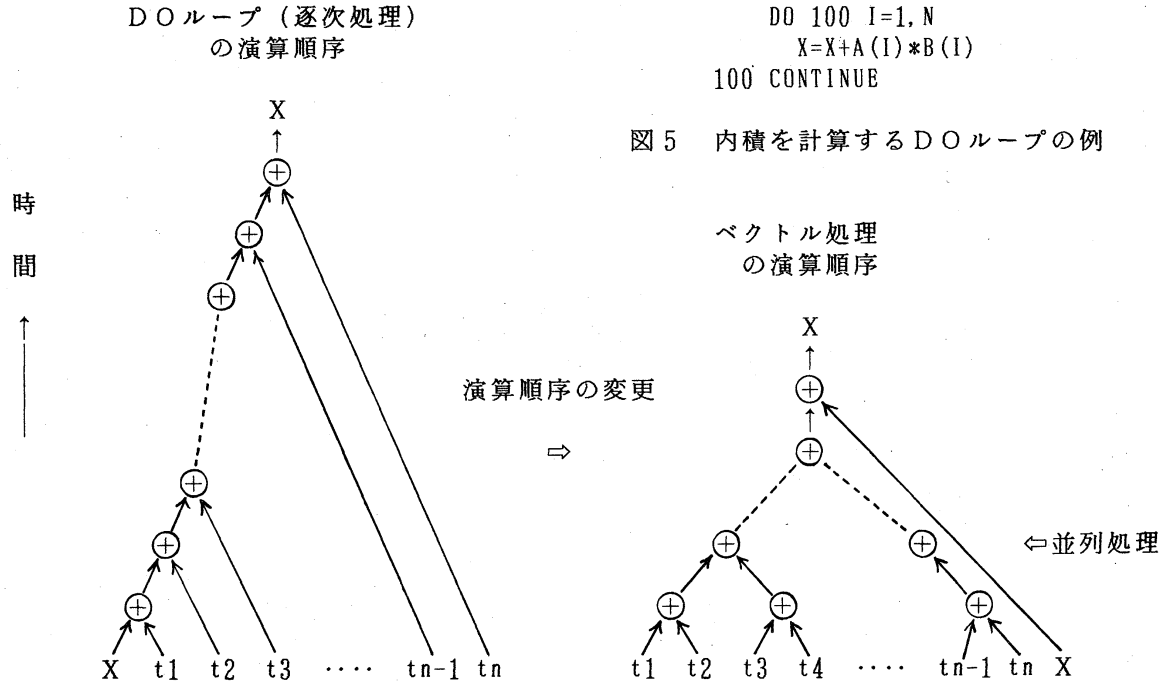


図６　ベクトルマクロ演算の演算順序変更

(3)　制御文を含むＤＯループのベクトル化技術

(1), (2)で紹介した技術は，ループ中の構造が単純な場合のベクトル化技術であるが，一般的には，ＤＯループ中にＩＦ文等の制御文を含むことが多い。制御文によって発生する手続きの流れの変化は，(1)で示したようなＤＯループの分割では，並列処理化することができない。しかし，見方を変えれば，手続きの流れの変化は並列処理化能なベクトル要素の実行／非実行に置き換えることができる。図７にその概要を示す。図７で“：mi”は条件付き演算機能（マスク付き演算機能）といい，miの内容（真／偽）によって左の実行文の実行／非実行を制御することを意味する。このことを利用して，ＦＡＣＯＭ　ＶＰシステムでは，ＩＦ文等の制御文を含むＤＯループもベクトル化できるようにしている。

```
      DO 100 I=1,N
        IF(A(I).GE.EPS) THEN
          B(I)=A(I)*C(I)
          A(I)=B(I)-D(I)
        ENDIF
  100 CONTINUE
              ⇩　ベクトル化
      mi=Ai.GE.EPS                ,i=1,2,...,N
      Bi=Ai*Ci          : mi      ,i=1,2,...,N
      Ai=Bi-Di          : mi      ,i=1,2,...,N
```

図７　制御文を含むＤＯループのベクトル演算化

更に，ＦＡＣＯＭ　ＶＰシステムには，条件付きの演算を効率良く動作させるためにマスク付き演算機能の他に，次の二つの機能がある。

・圧縮／拡散機能

・リストベクトル機能

これら三つの機能の概要を図８に，特徴を表３に示す。コンパイラは，条件付きの演算をこれら三つの方法から，最も効果的なものを自動的に選択している。

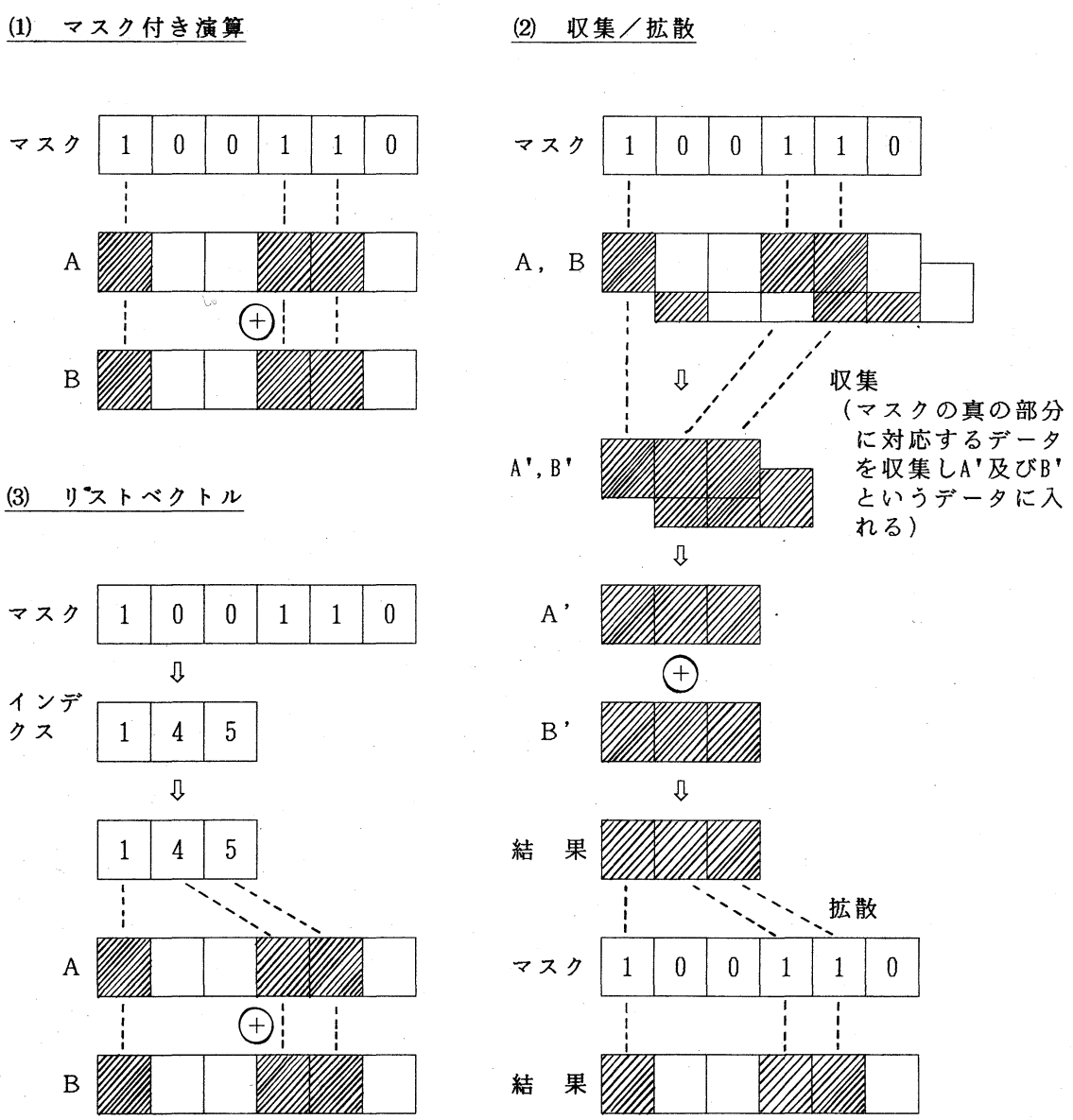


図８　条件付き演算の３機能の概要

表３　条件付き演算の３機能の特徴

NO	機　能	メリット	デメリット	選　択　基　準（場合）
(1)	マスク付き 演算	補助操作なし	全ベクトル 要素が演算対象	補助操作もなく最も広く使える （(2)(3)条件に該当しない場合）
(2)	収集／拡散	対象ベクトル 要素のみ演算	対象ベクトル要素の 収集／拡散という 補助操作を伴う	集めたベクトル要素を複数回利用又はそれを基に縮小した要素数で多くの演算を行う（関数）
(3)	リスト ベクトル	対象ベクトル 要素のみ演算	対象ベクトル要素の アクセスのための インデクスリスト 生成補助操作を伴う	演算対象ベクトル要素が少ない （条件式の真率が小さい）

(4)　その他のベクトル化技術

その他のベクトル化技術としては，

・ループ内の文を並列処理可能な部分と不可能な部分に自動的に分ける『部分ベクトル化技術』

・並列処理不可能な外部手続きの呼出しを無くすために，手続き自体を呼出し側のプログラムに組込み，ベクトル化を促進させる『外部手続き組込み技術』

などがある。

【４.３】ＶＰ向きプログラムへのチューニング

ベクトルプロセッサを有効に利用するためには，ベクトルプロセッサの特徴を活かすようなプログラムのチューニングが必要になることがある。これは，コンパイラの自動ベクトル化技術がいくら強力であっても，実行時に決まる値によってベクトル化の可否が決定されることがあるなど，自動ベクトル化には限界があるからである。また，使用者がプログラムを工夫することにより，もっと効率のよいプログラムにすることができることもある。ここでは，ベクトルプロセッサを

有効に使用するためのチューニングの基本事項と，ＦＡＣＯＭ　ＶＰシステムが提供しているチューニング機能について紹介する。

(1)　チューニングの基本事項

ベクトルプロセッサを有効に使用するために考慮すべき基本事項は，

・できる限りベクトル処理できる部分（ベクトル化率という）を大きくすること
・ベクトル処理できる部分でも，効率の良いベクトル処理が行えるようにすること

である。前者は，ベクトル化の可否を調べることにより比較的容易に知ることができるが，後者は更に，ベクトルプロセッサの次の特徴を利用したプログラミングを行うことにより実現できる。

・ベクトル要素数（ＤＯループの繰り返し回数）は，できる限り大きい方が効果的である
・ベクトルデータは，連続アドレスをもつ方が効果的である
・ベクトル演算で実行する部分は，まとめて記述した方が効果的である

(2)　ＦＯＲＴＲＡＮ77／ＶＰのＶＳＯＵＲＣＥリスト出力

ＦＯＲＴＲＡＮ77／ＶＰコンパイラは，リスト出力として，実行文がベクトル化されたか否かを，原始プログラム上に表示することができる。出力の例を図９に示す。図９で，各実行文の前の文字は，次の意味である。

Ｖ　……　その文がベクトル化された。
Ｍ　……　その文は，一部分がベクトル化された。
Ｓ　……　その文はベクトル化できなかった。

```
07000 S        DO 510 I=1,IPATCL
08000 M          DO 520 J=1,IPATCL
09000 V            IF( I.EQ.J) GO TO 520
10000 V              IP=KIND(I)+KIND(J)-1
11000 V              AX=QN(I,1)-QN(J,1)
12000 V              AY=QN(I,2)-QN(J,2)
13000 V              AZ=QN(I,3)-QN(J,3)
14000 V              IX=AX*RAH
15000 V              IY=AY*RAH
16000 V              IZ=AZ*RAH
17000 V              DIST=(AX-AL*IX)**2+(AY-AL*IY)**2+(AZ-AL*IZ)**2
18000 V              IF(DIST.LE.AH2) THEN
19000 V                DD=DSQRT(DIST)*RDELTA
20000 V                ND=DD
21000 V                NH=ND*RHDIST
22000 V                DH=DD-ND
23000 V                TPP(I)=TPP(I)+P(ND,IP)+0.5*DH*(P(ND+1,IP)-P(ND-1,IP)
24000         *                       +DH*(P(ND+1,IP)-2.*P(ND,IP)+P(ND-1,IP)))
25000 V                FO=F(ND,IP)+0.5*DH*(P(ND+1,IP)-F(ND-1,IP)
26000         *                     +DH*(F(ND+1,IP)-2.*F(ND,IP)+F(ND-1,IP)))
27000 V                TFX(I)=TFX(I)+FO*AX
28000 V                TFY(I)=TFY(I)+FO*AY
29000 V                TFZ(I)=TFZ(I)+FO*AZ
30000 M                JRAD(NH,IP)=JRAD(NH,IP)+1
31000 V              ENDIF
32000 V  520     CONTINUE
33000 S  510   CONTINUE
```

図９　ＶＳＯＵＲＣＥリストの出力例

(3)　ＦＯＲＴＲＡＮ77／ＶＰの最適化制御行

ＦＯＲＴＲＡＮ77／ＶＰでは，ベクトル化のための情報を原始プログラムのコメント行に書くことができる。このコメント行のことを，最適化制御行という。最適化制御行で指定できる情報としては，以下のものがある。

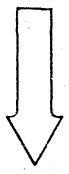
・配列要素の定義／引用がベクトル化できる関係である(NOVREC)

・ＤＯループの繰り返し回数

・ＩＦ文の条件式の真率　　など

コンパイラは，これらの情報を用いてより良いオブジェクトモジュールを生成する。最適化制御行を用いたチューニングの例を図１０に示す。

```
M          DO 100 I=1,N
V            VMIXD   = VMIX(IXL(I))*25.
S            DAFGD   = DAFG(IXL(I))*75.
S            DAFG(IXL(I)) =(DAFGD+VMIXD)/100.
V   100    CONTINUE
```



```
   *LOOP NOVREC(DAFG)
V          DO 100 I=1,N
V            VMIXD   = VMIX(IXL(I))*25.
V            DAFGD   = DAFG(IXL(I))*75.
V            DAFG(IXL(I)) =(DAFGD+VMIXD)/100.
V   100    CONTINUE
```

配列ＤＡＦＧはリストベクトルであり，添字式ＩＸＬ（Ｉ）が同じ値を持つ可能性があるため，そのままではベクトル化できない。この例は，添字式がすべて異なっていることが，陽に分っている場合に，ベクトル化できる関係であることを指示した例である。

図１０　最適化制御行を用いたチューニングの例

⑷　ＦＡＣＯＭ　ＶＰ向きプログラム改善支援システム

（会話型ベクトライザ）

ＦＯＲＴＲＡＮプログラムをベクトルプロセッサ向きにチューニングするためには，一般的に次の手順を踏む。

・第１段階　……　プログラムの動的解析

プログラムの動的な実行の様子を把握し，どの部分をチューニングすれば効果的なのかを調査する。

・第２段階　……　ベクトル化の解析

プログラムのベクトル化の可否を調査する。

・第３段階　……　チューニングの検討

第２段階で調査した結果を基にベクトル化されていない部分については，ベクトル化の促進を，ベクトル化されている部分については，更に効率の良いベクトル化ができないかを検討する。

・第４段階　……　プログラムの編集／修正

第３段階で検討した結果を基に，実際にプログラムを編集／修正する。

これらの一連の作業が効率良く実行できるように，改善支援ツールをフルスクリーン型ディスプレイを用いた対話処理システムとして実現したものが『会話型ベクトライザ』である。図１１及び１２に会話型ベクトライザの表示例を示す。図１１及び１２の画面の上半分は，チューニング情報を表示する画面であり，使用者はこの部分の情報を見ながら画面の下半分で原始プログラムの修正ができる。

```
VECTORIZE - MESSAGE LIST --- RKGS ---------------------- LINE 000015 COL 001 080
COMMAND ===>                                              SCROLL ===> PAGE

S JND237I-I LN0:00013600 - 00013600
            ARRAY ACSVAL CANNOT BE VECTORIZED BECAUSE RECURSIVE REFERENCE MAY
            TAKE PLACE.
  JND201I-I LN0:00013900 - 00014000
            THE STATEMENTS IN THIS RANGE ARE VECTORIZED BY DO VARIABLE I.
  JND401I-I LN0:00014200
            SPECIFY THE ITERATION COUNT OF THIS DO LOOP BY OPTIMIZATION
            CONTROL LINE.
  JND401I-I LN0:00014300
            SPECIFY THE ITERATION COUNT OF THIS DO LOOP BY OPTIMIZATION
            CONTROL LINE.
  JND201I-I LN0:00014400 - 00014500
            THE STATEMENTS IN THIS RANGE ARE VECTORIZED BY DO VARIABLE I.
. . . . . .
EDIT --- S2272.SYSVECT.T2024446.DATA(AARKGS) ---------------- COLUMNS 001 072
COMMAND ===>                                              SCROLL ===> PAGE
013400        330M          DO 10 I=J,N                    600S        100E
013500       3000V            IF(L(I).GT.0) THEN         30000S      10000E  50.0%
013600      19250M              ACSVAL(L(I))=            3500OS       5000E
013700                   1       ACSVAL(L(I))+B(I)          S           E
013800                         ENDIF                        S           E
```

この画面で，ベクトル化メッセージを選択（Ｓの文字の挿入）すると，図１２の画面に切り変る。

図１１　会話型ベクトライザの表示例－その１－

```
---------- VECTORIZE DETAIL INFORMATION - JND2371 --------------(1/1)
  TUNING INFORMATION
   - SPECIFY NOVREC, IF THERE IS NO RECURSIVE REFERENCE OF AN ARRAY
     ACSVAL    FOR SURE , AND THE INDICES ACCESSED BY LIST VECTOR
    ARE ALL DIFFERENT.

   TUNING EXAMPLE

          .                              *VOCL LOOP,NOVREC(ACSVAL)
  +--> DO 10 I=1,NNN                     +--> DO 10 I=1,NNN
  |       .                              |       .
  |  ACSVAL    (L(I)) =                  |  ACSVAL    (L(I)) =
  |          .          =======>         |          .
  |    =  ACSVAL     (L(I))              |    =  ACSVAL     (L(I))
  |       .                              |       .
  +-- 10 CONTINUE                        +-- 10 CONTINUE
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
EDIT --- S2272.SYSVECT.T2024446.DATA(AARKGS) ---------------- COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> PAGE
013400         330M          DO 10 I=J,N                 ||      600S        100E
013500        3000V            IF(L(I).GT.0) THEN        ||    30000S      10000E  50.0%
013600       19250M              ACSVAL(L(I))=           ||    35000S       5000E
013700                     1       ACSVAL(L(I))+B(I)     ||         S          E
013800                           ENDIF                   ||         S          E
```

図１２　会話型ベクトライザの表示例－その２－

【５】まとめ

以上，ベクトルプロセッサを使用する言語の分類とそれらの得失，及び自動ベクトル化方式を採用しているFORTRAN77/VPについての概要を紹介した。現時点では，互換性及びプログラム財産の利用の観点から自動ベクトル化方式が主流を占めているが，FORTRAN 8Xが制定され，ベクトル（配列）処理記述がFORTRAN に取り入れられれば，ベクトル（配列）処理言語が主流になるものと思われる。しかし，言語はあくまでも与えられた並列性を記述するための手段に過ぎない。真にベクトルプロセッサを有効に使用するためには，

・問題のコード化の全過程を通じて並列性を追及すること

・特に，アルゴリズムの選択においては，並列処理に向いた解法を採用すること

が必要であり，より一層の並列処理向きアルゴリズムの研究開発が望まれる次第である。

《参考文献》

[1] R. W. Hockney and C. R. Jesshope, "Parallel Computers", Adam Hilger Ltd., Bristol, 1981
[2] 『日経エレクトロニクス　特集：速さを競うスーパーコンピュータ』 NO.314, 日経マグロウヒル社, 1983年
[3] CRAY-1 FORTRAN(CFT) 説明書, SM-007, センチュリリサーチセンタ㈱, 1979年
[4] "MAINSTREAM-EKS/VSP CRAYPACK Supplement to BCSLIB User's Manual", Boeing Computers Services Company, 1982
[5] "AP-120B Math Library", Publication 7288-02, Floating Point Systems Inc., 1976
[6] 「FACOM FORTRAN SSLII 使用手引書」, 99SP-0050, 富士通㈱, 1980年
[7] 「FACOM 230-75 MVII AP-FORTRAN文法書」, 75FP-0370, 富士通㈱, 1977年
[8] 「FACOM 230-75 MVII AP-FORTRAN使用手引書」, 75FP-0500, 富士通㈱, 1980年
[9] J. H. Austin Jr, "THE BURROUGHS SCIENTIFIC PROCESSOR", Infotech State of the Art Reports, SUPERCOMPUTERS, Infotech International Limited, 1979
[10] "PROPOSALS APPROVED FOR FORTRAN 8X X3J3/S6.81", ANSI, May 1982
[11] "CFD A FORTRAN-based language for ILLIAC IV", Computational Fluid Dynamics Branch, NASA, Ames Research Center, 1974
[12] J. R. Rice, "ELLPACK 77 User's Guide ", CSD-TR289, Computer Science Depertment, Purdue University, 1980
[13] 寺野他, 「移流項を含む拡散方程式の有限要素近似パッケージ CAP/PFDR 」, 情報処理学会第23回（昭和56年後期）全国大会　講演論文集, 5B-9
[14] 梅谷他, 「数値シミュレーション用プログラム言語 DEQSOL 」, 情報処理学会　数値解析研究会　資料 5-2, 1983年
[15] 「FACOM OS IV/F4 MSP FORTRAN77/VP 使用手引書」, 78SP-5680, 富士通㈱, 1983年
[16] S. Kamiya et al., "Practical vectorization techniques for the "FACOM VP" " INFORMATION PROCESSING 83, IFIP, Elsevier Science Publishers B.V., Holland, 1983