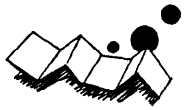
解　説



技術計算プログラムの自動ベクトル化技術†

梅谷征雄†† 高貫隆司††† 安村通晃††

1. 緒　言

近年，技術計算分野には，一般にスーパコンピュータと呼ばれる高速演算装置の進出が目覚ましいが，それらの多くは，高速半導体の使用と同時に，パイプライン処理あるいは並列処理などの論理方式上の工夫により性能を上げている[1]．これらは，種々のレベルの演算並列性を利用して高速処理を行うわけであり，逐次処理を基本とする従来プロセッサとはかなり使い勝手の異なるものになっている．しかし，一方では，逐次処理を前提にすでに20余年の歴史を有するFORTRAN言語があり，そこで培われたプログラミング習慣と膨大なプログラム財産をいかにしてスーパコンピュータに馴染ませるかが大きな課題となっている．

そのために種々のソフトウェア上の工夫が行われているわけであるが，主要なアプローチを分類すると表-1のようになる．第1は，システム関数／サブルーチン方式であり，システム側で用意した並列計算関数ないしサブルーチンをプログラマが適宜使用する方式である．これは，マクロな並列処理に向く反面，きめ細かな並列化に適さない．第2のアプローチは，従来のプログラミング言語の仕様を並列演算向きに拡張する方式であり，最も正統的であるが標準仕様からの乖離が問題となる．拡張仕様の統一化が望まれる．第3は，本解説の主題である自動並列化のアプローチであり，FORTRANなどで記述されたプログラム中の並列化可能部分をコンパイラで検出して，ハードウェア機構に適した機械命令コードを生成する方式である．従来のプログラミング環境を変えないこと，既存プログラムの並列化が可能であることに特徴があるが，プログラムの性質およびコンパイラの能力により並列化範囲が制約されることは否めない．

これらのアプローチは排他的なものではなく，実際には表-2に示すようにそれぞれの特徴を生かしながら組み合わせて用いられることが多い[4),7),8),10),12),15)~17)]．

本稿ではこのうち，第3のアプローチである自動並列化の実現に必要な技術の現況と課題について述べる．

なお，スーパコンピュータの論理方式は多様であり，それに応じて並列化方式も異なるわけであるが，本稿では話を具体化するために，現在の多くのスーパコンピュータがそれに類すると思われるベクトルプロセッサ型に話を限ることにする．これは次に示すベクトル命令と呼ぶ，一次元の並列演算制御命令を逐次実

表-1　スーパコンピュータへのソフトウェアアプローチ

方　式	プログラミング例
システム関数／サブルーチン方式	CALL VIP (S, A, B)
言語拡張方式	S=S+A.,.B
自動並列化方式	DO 10 I=1, N S=S+A(I)*B(I) 10 CONTINUE

表-2　スーパコンピュータのソフトウェア方式

機　種	メーカ	ソフトウェア方式	主要プログラミング言語
ASC[7)]	Texas Instrument	自動並列化方式 言語拡張方式	ASC-FORTRAN
STAR-100[15)]	CDC	同　上	STAR-FORTRAN
ILLIAC IV[4)]	Burroughs	同　上	IVTRAN
CRAY-1[16)]	Cray Research	自動並列化方式 システム関数方式	CRAY-FORTRAN
CYBER 203[16)]/205[17)]	CDC	自動並列化方式 言語拡張方式	Cyber 200 FORTRAN
230/75 AP[8)]	富士通	言語拡張方式 システム関数方式	AP-FORTRAN
M200H IAP[12)]	日　立	自動並列化方式	最適化FORTRAN

† Vectorization Techniques for Scientific Program by Yukio UMETANI (Central Research Laboratory, Hitachi Ltd.), Ryuji TAKANUKI (Software Works, Hitachi Ltd.) and Michiaki YASUMURA (Central Research Laboratory, Hitachi Ltd.).

†† 日立製作所中央研究所
††† 日立製作所ソフトウェア工場

行する機能を有するものである.

Z←X. op. Y

.op. 演算種別

$\left.\begin{array}{l}\mathbf{X}=(X_1, X_2, \cdots, X_n)\\ \mathbf{Y}=(Y_1, Y_2, \cdots, Y_n)\\ \mathbf{Z}=(Z_1, Z_2, \cdots, Z_n)\end{array}\right\}$ ベクトル

ベクトルプロセッサ向けの並列化を特に“ベクトル化”と呼ぶ. また, 並列化の対象言語を FORTRAN に限定することにする.

本稿では, まず第2章にて自動ベクトル化の必要技術を概括した後, 第3章で主要技術を詳述する. また, 第4章で実用システムにおける自動ベクトル化方式の実例を紹介する.

2. 自動ベクトル化技術総論

本章では, コンパイラによる自動ベクトル化の実例を紹介し, 必要技術を概括する.

2.1 自動ベクトル化の実例

自動ベクトル化の一例として, HITAC M-200 H IAP (Integrated Array Processor) における FORTRAN ソースコードとコンパイラの生成するオブジェクトコードを 図-1 に示す. M-200 H IAP は28種類のベクトル演算命令を備え[1], 一方 VOS 2/VOS 3 最適化 FORTRAN 77 コンパイラは, 最内側 DO ループ部分を並列化しベクトル命令列を生成する機能を有する. 図-1 にて下左がベクトル化を行わない場合のオブジェクトコード, 下右が自動ベクトル化の結果になるベクトル命令を含むオブジェクトコードである. 右で VEME, VEAE がベクトル命令であり, それぞれ, 一連のデータ列 $\mathbf{X}=(X(1), X(2), \cdots X(N))$, $\mathbf{Y}=(Y(1), Y(2), \cdots Y(N))$, $\mathbf{T}=(T(1), T(2), \cdots, T(N))$, $\mathbf{Z}=(Z(1), Z(2), \cdots, Z(N))$ の間で要素ごとの乗算 (T(I)←X(I)*Y(I))I=1, N および加算(Z(I)←Z(I)+T(I))I=1, N を行う. ここで **T** は演算の中間結果を置くために取られた作業ベクトルである. このように, DO ループにおける繰り返しはベクトル命令部に集約され, ハードウェアで高速に処理される. すなわち, FORTRAN で指定した演算順序を変更して実行するわけであり, 変更の方法とそれに必要なチェック機能がコンパイラの能力をきめることになる.

〈FORTRAN ソースコード〉

```
 ISN        SOURCE STATEMENT

00001           DIMENSION X(100),Y(100),Z(100)
00002           DO 10 I=1,N
00003        10 Z(I)=Z(I)+X(I)*Y(I)
00004           STOP
00005           END
```

〈オブジェクトコード〉

非ベクトル化コード

```
100000   L     9,=F'4'
         L     7,=F'1'
         L     6,N
         LR    2, 6
         SR    2, 7
         AR    2, 7
         LR    3, 2
         ST    2,I0001
         LTR   2, 2
         BC    13,#100001
100002   LR    2, 9
         LR    10, 3
         LR    11, 2
    ┌──→10 LE    2,X
    │    ME    2,Y
    │    AE    2,Z
    │    STE   2,Z
    │    AR    11, 9
    └──  BCT   10,#10
100004   LR    8, 6
         AR    8, 7
100001   L     15,=V(F#RCON)
         BAL   14,  52( 0,15)
```

ベクトル化コード

```
100000   L     10,=F'1'
         L      2,N
         SR     2,10
         AR     2,10
         BC    13,#100001
      10 SR     0, 0
         L      1,N
         LA    15,=0A001
         VEME   0,15
         LA    15,=0A002
         VEAE   0,15
100004   L     11,N
         AR    11,10
100001   L     15,=V(F#RCON)
         BAL   14,  52( 0,15)
```

図-1 自動ベクトル化の実例

表-3 自動ベクトル化に必要な確認項目とコンパイル技術

項番	確 認 項 目	コンパイル技術
1	データ参照関係	データ参照解析技術
2	制 御 構 造	制御構造解析技術
3	添 字 形 式	指標解析技術
4	演 算 種 類	意味解析技術
5	データ形式	
6	ル ー プ 長	

2.2 自動ベクトル化に必要な技術

逐次的な実行を前提としたプログラムにベクトル命令を適用するためには, コンパイラは適用可能性を確認する必要がある. そのための確認項目と, 確認に必要なコンパイル技術をまとめると 表-3 のようになる.

まずベクトル化とはデータ間の参照関係を変えないようにして演算の実行順序を変更することであるから, そのような順序変更を許容するようなデータ参照関係にあることを確認する必要がある. そのためのデータ参照解析技術が基本的に重要

である．

また，ベクトル命令は一般に一連のデータに対して同一の演算を要求するので，DOループ内の制御の流れが乱れる場合にはベクトル命令の適用が難しくなる．それを避けるための制御構造解析技術が必要である．

さらに，一連の配列データをベクトルとして認識するためには配列の添字形式を確認する必要があり，そのための指標解析技術が要求される．この技術はまた，配列データに対してデータ参照解析を行う場合にも基幹技術として必要となる．

そのほか，ベクトル命令のレパートリ制約による演算種類とデータ形式のチェックや，一般に性能の悪い短ループへのベクトル命令の適用を抑止するためのループ長チェックなどが必要であり，そのための意味解析技術が要求される．

本稿では，これらの技術のうち特に重要な，データ参照解析技術とその一環である指標解析技術，および制御構造解析技術を取り上げ，次章で詳しく述べることにする．

3. 自動ベクトル化技術各論

3.1 データ参照* 解析技術

データ参照解析は，2章で述べたとおりベクトル命令の使用に伴う演算順序の変更により，データ参照順序が変わらないことを保証するために必要である．

図-2 にベクトル化を許容しないデータ参照関係と許容する関係の例を示す[3]．図-2は，横軸にループ制御変数Iの値，縦軸に文の番号をとり，ベクトル化時の演算順序を本来の演算順序と対照させて示すものである．

例1がベクトル化を許容しないケースであるが，A(I) と A(I+1) の関係が問題となる．本来はループ制御変数Iをまず固定して文番号を動かし次いでループ制御変数を動かすことから黒矢印で示す演算順序となる．一方ベクトル化時は，各文ごとにループ制御変数値を動かすことから白矢印の順序となる．したがって，文番号2と3における要素 A(2) の参照順序を比較すると，本来はまず I=1 にて文番号3の引用が行われた後に I=2 にて文番号2の定義が行われるのに対して，ベクトル化時には図のとおりその順序が逆転して正しい結果を与えなくなる．

他方，例2のケースでは，いずれの場合でも参照順序が変わらないことから結果の同一性が保証されることになる．

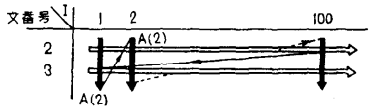
〈ソースプログラム〉

文番号	例　1
1	DO 10 I=1, 100
2	A(I)=B(I)+C(I)
3	10 E(I)=A(I+1)*D(I)

文番号	例　2
1	DO 10 I=1, 100
2	A(I+1)=B(I)+C(I)
3	10 E(I)=A(I)*C(I)

〈DO ループの演算順序〉

例1（ベクトル化を許容しないデータ参照関係）



例2（ベクトル化を許容するデータ参照関係）

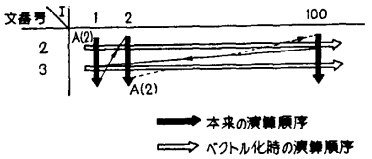


図-2 ベクトル化による演算順序の変更

このように，DO ループ内に同一配列ないし変数の参照が2回以上ある場合に，相互の位置関係および配列であれば添字の関係が問題になるわけであり，少なくとも一方が定義側にあるすべての対に関してベクトル化により参照順序の逆転が起きないことをコンパイラでチェックする必要がある．そのために要求される技術をデータ参照解析技術と呼んでいる．

データ参照解析の課題を整理すると **表-4** のようになる．

データは配列と単純変数に分類されるが，配列データに関しては添字式の関係でベクトル化の適否がきまるので参照対間の添字比較が中心となる．添字式は，

表-4 データ参照解析の課題

分　類	課　題
配列データの参照解析	(a) 指標解析による添字式処理. (b) 添字比較による要素参照順序の判定. (c) 再帰的データ参照関係の検出と処理.
変数データの参照解析	(a) 再帰的データ参照関係の検出と処理. (b) 中間変数の配列化と終値保証.
そ　の　他	明示されないデータ参照関係の処理.

* 本稿にて“参照”の用語はデータの“引用”と“定義”の両者に対して用いる．

ループの進行とともに規則的にその値を変える指標変数により表現されているので，その種々の形式に対する指標解析と標準化が第1の課題となる．第2に指標解析の結果にもとづく添字比較と要素参照順序の決定が必要であり，順序関係判定の精緻さが課題となる．さらに，一般にはベクトル化を許容しないデータ参照関係の特別な場合として内積計算などで用いられる再帰的データ参照関係があり，これらは実計算にてしばしば出現し，ハードウェア命令の備えられているケースも多いのでその処理方式が重要な課題である．

単純変数については添字を持たないので，位置関係と“引用”，“定義”の組み合わせのみによりベクトル化の適否が決定される．引用が定義に先行する場合には再帰的データ参照となるので配列の場合と同様その処理が問題となる．定義が引用に先行する場合には，ベクトル化にあたって当該変数の作業配列への置き換えが必要であり，またDOループから抜けた後に当該変数が引用される場合には，作業配列の終値を当該変数に保証する必要がある．

配列，単純変数を通じてそのほかの考慮事項としては，Equivalence，副作用などによる明示されないデータ参照関係の検出と処理が挙げられる．

以上，データ参照解析における課題を概括したが，特に配列データの参照解析にて重要な指標解析技術，添字比較技術，再帰的データ参照処理技術を取り上げて次にのべることとする．

（1）　指標解析技術

指標解析とは，配列添字に含まれる種々の指標変数の性質および添字式の性質を解析するものであり，後にのべる添字比較の基礎となる．

図-3 に，実プログラムで用いられる指標変数の種類を分類して示す．第1に(a)に示すDO制御変数があり，最も一般的に用いられる．第2に(b)に示すように DO ループ中の再帰的代入文で定義される再帰

(a) DO 制御変数: I

```
   DO 10 I=1,N
   A(I+1)=…
10 CONTINUE
```

(b) 再帰型指標変数: ID

```
   DO 10 I=1,N
   A(ID)=…
10 ID=ID+2
```

(c) 置き換えられた指標変数: IR

```
   DO 10 I=1,N
   IR=N+1-I
   A(IR)=…
10 CONTINUE
```

(d) 巡回型指標変数: IC

```
   DO 10 I=1,N
   A(IC)=…
10 IC=I
```

図 3　指標変数の種類

```
ISN        SOURCE STATEMENT
00001            DIMENSION A(100),B(100),C(100)
00002            DO 10 I=1,N
00003            I1=N1-I
00004            I2=I1-1
00005            INDJ=IND+J
00006            A(I1)=A(I2)
00007            B(IND)=C(INDJ)-3(IND-1)
00008            IND=IND+2
00009         10 CONTINUE
00010            STOP
00011            END
```

図-4　非標準指標変数例

型指標変数があり，DO 制御変数とは異なる初期値，増分値を取らせる一手段としてしばしば用いられる．同様な目的で，(c)に示すようなループ内での指標変数の置き換えも行われる．さらに特殊な場合として，(d)に示すような巡回型指標変数が用いられることもある．このような指標表現の多様さは旧来FORTRANにおける添字式，DO 制御範囲の表現形式の制約に起因するところが大きく，この点は FORTRAN 77 の普及に伴って今後改善されてゆくものと思われる．

実際のプログラムでは，以上の形式が組み合わせて用いられるわけであり，その例を **図-4** に示す．図-4にて，Iが DO 制御変数，IND が再帰型指標変数，I1, I2, INDJ はいずれも置き換えられた指標変数である．ここで，A(I1) と A(I2) の参照関係を得るためには I2=I1−1 の関係を知る必要があり，B(IND) と B(IND−1) が共通データを参照しないことを確認するためには IND=IND+2 の関係式を解析せねばならず，DO ループ内のデータの流れを徹底して解析する必要がある．そのための処理が指標解析である．

指標解析の課題と必要技術を整理すると **表-5** のとおりである．

指標解析の課題としては，再帰的指標変数や巡回型指標変数の定義文の検出および DO ループ中での指標変数の置き換えの追跡が挙げられるが，このためには広義のデータフロー解析が必要である．また添字式間の比較を容易にするために，たとえば 図-4 にて I1 を N1−I I2 を N1−I−1 として保持するなど，指標

表-5　指標解析の課題と必要技術

項番	課　　題	必　要　技　術
1	再帰型指標変数，巡回型指標変数の検出	データフロー解析技術
2	指標変数の，定義／引用関係の追跡	
3	指標変数と添字式の標準化	数式処理技術

変数，添字式の標準化の操作が必要であり，このための数式処理技術が要求される．

（2） 添字比較技術

添字比較技術は対象とする添字表現の範囲ならびに比較の正確さによりその程度がはかられる．

図-2 に示したとおり，一般に配列データについて，同名配列の（引用，定義）あるいは（定義，定義）の組における，同一要素に対する参照順序が問題となる．上方での配列参照を I=I1 回目，下方での参照を I=I2 回目とする時，I1≦I2 であればベクトル演算の導入による順序変更と整合するので問題なく，I1>I2 の時に都合が悪い．ただし，両者が共通要素を参照しない場合には問題がない[3]．

添字表現は比較の容易さにより，点添字，線形添字，多重線形添字，非線形添字などに分類される．各区分の例と添字比較の課題を整理すると **表-6** のようになる．

まず，点添字とは，ループの進行につれて値を変えない添字のことであり，点添字同志の比較については A(J) と A(J1) のようにループ内で関係の明示されない添字間の比較が問題となる．

次いで，線形添字とはループの進行につれて値が定値で変化する添字のことであり，最も一般に用いられる．点添字も変化値がゼロの線形添字に含めて考えるのが普通である．線形添字間の比較方式は従来から最も良く研究されている[2)~4),7)]．

一般に線形添字は，各次元に対して〔初期値，増分値，終値〕の3つ組で表現されるので，初期値を IN，増分値を C とする時，ループ制御変数 I の初期値をゼロ，増分値を1に標準化して取ることにより，C*I+IN と表現できる．

表-6 添字式の分類と添字比較の課題

分類	例	添字比較の課題
点添字	DO 10 I=1, N 10 A(J)=…+A(J+1)	変数を含む添字式の比較
線形添字	DO 10 I=1, N 10 A(I)=…+A(I+1)	・指標変数の変域の考慮 ・増分値と初期値差の組み合わせの考慮 ・多次元添字式の比較 ・非標準指標変数を含む添字式の比較 ・外側ループにおけるデータ関係の考慮
非線形添字	DO 10 I=1, N I1=IV(I) I2=IW(I) 10 A(I1)=…+A(I2)	（一般に比較困難）

単一次元を取り出した場合，C1*I+IN1, C2*I+IN2 間の添字比較の一方式は，C1=C2 の場合に IN1 と IN2 の大小比較を行うことである．すなわち，IN1≧IN2 であれば C1*I+IN1≧C2*I+IN2 が成り立つことから，C1*I1+IN1=C2*I2+IN2 なる I1, I2 があるとすれば I1≦I2 である．そこで，各次元に対して上記の比較を行い，すべての次元で I1≦I2 である場合に全体として I1≦I2 と判定し，次元ごとに大小関係が異なれば共通要素を参照しないと判断する[4)]．

これに対し，I≧0 における I と 2*I+1 のように，増分値は等しくないがループ制御変数の変域を考慮すると実は共通要素を参照しない場合や，2*I と 2*I+1 のように増分値と初期値差の組み合わせにより共通要素を参照しない場合，(I, I) と (I+1, I+2) のように次元ごとに添字値を等しくする制御変数対が異なるために共通要素を持たない場合などへの配慮を行うために，さらにループ制御変数の値域，増分値と初期値差間の組み合わせ，次元間の相関などを考慮した添字比較方式も報告されている[3)]．

これらの方式を，図-4 に示すような非標準指標変数を含む添字式に適用するためには，Ci, INi などを数式として保持し処理する必要がある．

また，次に示すガウス消去の計算ループ例

```
      DO 10 K=1, N-1
      DO 10 J=K+1, N
      C=A(J, K)/A(K, K)
      DO 10 I=K, N
   10 A(J, I)=A(J, I)-C*A(K, I)
```

にて，最内側の A(J, I) と A(K, I) が共通要素を参照しないことを確認するためには，外側ループに逆のぼって J と K の関係を明確にする必要がある．線形添字の比較に関しては，以上の点を考慮しなければならない．

さらに，表-6 に示すように，添字値の増分が一定でない非線形添字もスパース配列参照の場合などに用いられるが，これらの添字比較は一般にきわめて困難である．

（3） 再帰的データ参照処理技術

再帰的データ参照関係は，次のように計算結果がそれ以降のループの繰り返しにおいて再使用されるような関係にあるものを言う．

```
      DO 10 I=1, N
        =……+A(I-1)+……
   10 A(I)=…
```

表-7　再帰的演算の分類とベクトル化技術

再帰距離		例	ベクトル化技術
文間隔	指標間隔		
0	1	〈内積〉 DO 10 I=1, N 10 SUM−SUM+B(I)∗C(I) 〈一階逐次計算〉 DO 10 I=1, N 10 A(I+1)=B(I)∗A(I)+C(I)	◦ 添字比較技術 ◦ 構文解析技術
	2	〈二階逐次計算〉 DO 10 I=1N, 10 A(I+2)=B(I)∗A(I+1) +C(I)∗A(I)+D(I)	
1以上	—	DO 10 I=1, N =⋯+A(I)⋯ 10 A(I+1)=⋯	◦ 添字比較技術 ◦ 構文解析技術 ◦ 数式処理技術

特に内積計算などは，計算結果を同一文で再使用している点が特徴的である．

```
DO 10 I=1, N
10 SUM=SUM+A(I)*B(I)
```

これらのデータ参照関係は一般にベクトル化に適さないわけであるが，定義と引用の間隔すなわち再帰距離が短い場合には内積命令などの再帰的演算命令の使用によるベクトル化の可能性がある．

表-7 に，再帰距離による演算の分類とベクトル化の必要技術を整理して示す．

再帰距離は文間隔と指標間隔により測られるが，文間隔がゼロのケースはベクトル化が比較的容易である．指標間隔の演算としては，内積計算，一階逐次計算などがあり，これらをベクトル化するためには，定義部と引用部の添字差を計算するための添字比較技術，引用部の式内位置を確認するための構文解析技術などが必要となる．指標間隔がひらくにつれて構文解析は複雑になる．

一方，文間隔が1以上の場合には，さらに数式処理により1文に縮約して標準化をはかることになるが，一般に困難な仕事である．

3.2　制御構造解析技術

DO ループ中に条件文，GO TO 文による制御移動が含まれる場合には，データ参照解析に先立って制御構造解析を行い，その構造がベクトル化を許容するか否かを判断する必要がある．また許容する場合にはさらに，データ参照解析時に比較すべきデータ間の制御構造上の位置関係を考慮する必要がある．非標準指標変数の検出においても，定義文の制御構造上の位置を考慮せねばならない．

表-8　制御構造とベクトル化の難易

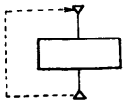
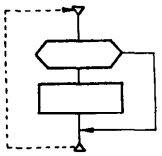
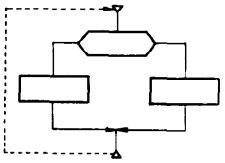
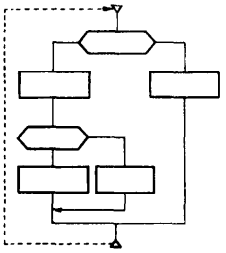
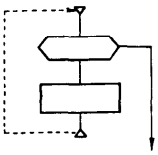
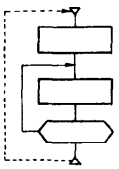
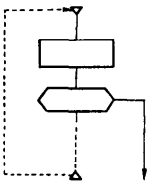
分　類	制　御　構　造	ベクトル化の難易
単純構造		易
マスク構造		やや困難
選択構造		やや困難
網目構造		困　難
探索構造		きわめて困難
内部ループ構造		不可能
離脱構造		不可能

表-8 に，自動ベクトル化の容易さから見た制御構造の分類を示す．最もベクトル化の容易なものは，DO ループ中に条件文を含まぬ単純構造であり，制御構造解析を必要としない．続く，マスク構造，選択構造，網目構造は，データ参照関係がベクトル化に矛盾せず，ループごとの演算並列性が保証されれば，適切なハードウェア機構を備えることによりベクトル化が可能である．さらに，比較結果による中途からループ外への分岐を含む探索構造になると，ループ回数が演算結果により変わることから特殊なベクトル命令が必要となる一方，自動ベクトル化の困難さも増してくる．最後に，DO ループ中にさらにループを含む内部ループ構造，あるいは演算結果による中途でのループ外分岐を含む離脱構造などは，原理的にベクトル化不可能と思われる．

3.3 そのほかのベクトル化技術

以上，ベクトル化の判定に必要な技術を中心に述べたが，ベクトル命令列を生成する実用的見地からすれば，そのほかに次のような技術が必要となる．

（1） 中間ベクトル領域の最少化

ベクトル演算の中間結果は，レジスタないし主記憶上に置かれるが，その所要量を最少化するために，レジスタないし主記憶領域の再利用を推し進める必要がある．

（2） 配列記憶域の割り付け技術

ベクトルプロセッサでは主記憶装置，演算装置間のデータ転送能力を高めるために，主記憶装置を数多くのバンクに分け並列アクセスを可能とする事が多いが，配列添字の増分値によっては単一バンクにアクセスが集中し転送能力の落ちるケースがある．このような危険をできるかぎり減らすために記憶域の割り付け方式が重要な技術である．

（3） ベクトル長の予測

一般にベクトルプロセッサは，ベクトル命令開始時にスタートアップの時間を必要とすることから，短ベクトルに対する性能が悪くなる．そこで短ベクトルの適用を抑止するための，コンパイル時のループ長予測が重要な技術となる．

4. 事例紹介

4.1 ILLIAC IVにおける自動ベクトル化方式[4)~6)]

ILLIAC IVは，64 個の PE (Processing Element) が同一命令を同時に実行する並列処理型のアレイプロセッサである．CU (Control Unit) が 64 個の PE の実行を制御する．**図-5** に ILLIAC IV の基本構成を示す．ILLIAC IV は，128 K 語（1 語は 64 ビット）のメモリをもつ．CU は，128 K 語すべてをアクセスできるが，各々の PE は，2 K 語しかアクセスできない．ILLIAC IV のような並列処理型のアレイプロセッサでは，PE による並列処理の比率を高くし，同時に実行する PE の個数をできる限り多くしないと，高性能を期待できない．これらは，ソフトウェアに課せられた課題である．

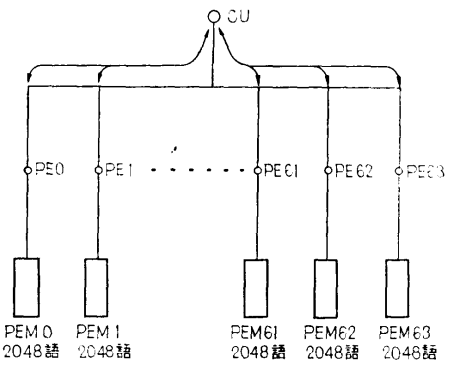


図-5 ILLIAC IV の基本構成

ILLIAC IV のソフトウェアとしては，IVTRAN と呼ばれる FORTRAN コンパイラが提供されている．IVTRAN は，IBM と CDC の FORTRAN を融合したような言語仕様を持ち，さらに並列演算を表現するために次のような拡張を行っている．

（1） DO FOR ALL 文

DO FOR ALL 文は，次の形をもつ．

DO k FOR ALL $(i_1, \cdots, i_n)/s$

ここで k は DO の端末文番号，i_j は DO の制御変数，s は制御変数の変域を示す．DO FOR ALL 文と端末文の間の文は，s で示された変域のすべての値に対して同時に実行される．

```
      DO 10 FOR ALL (I)/{1,2…10)
    1 A(I)=B(I)+C(I)
    2 E(I)=A(I)+D(I)
   10 CONTINUE
```

I の値の 1 から 10 に対して文 1 を並列に実行し，次に同じく I の値の 1 から 10 に対して文 2 を並列に実行する．

（2） メモリ割り付け宣言

前述のように，ILLIAC Ⅳ のメモリは，2 K 語単位で PE にくくりつけられている．このため，配列の

PEM 番号 / アドレス	1	2	3	4	5
1	11	21	31		
2		12	22	32	
3			13	23	33
⋮					
2048					

(a)

11	12	13
	21	22	23
		31	32	33

(b)

11	12	13
21	22	23
31	32	33

(c)

11	12	13	21	22	23	31	32	33

(d)

図-6　IVTRAN のメモリ割り付け

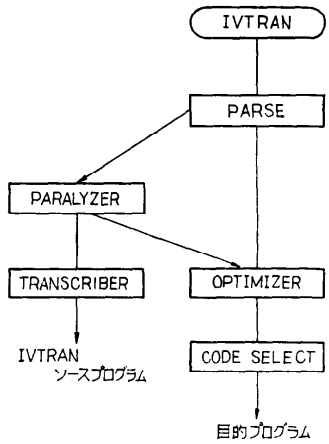


図-7　IVTRAN コンパイラの構成

各要素が並列に演算されるためには，各要素が別々の PE に対応するようにメモリ割り付けされなければならない．IVTRAN では，DIMENSION 文，型宣言文，COMMON 文で配列を宣言するときに，メモリ割り付け方法を指定することができる．いま，3×3 の配列を例にとると，標準では，**図-6**(a) のように 1 行ごとに対応する PE がひとつずつずれる形でメモリ割り付けされる．これに対し，(b)は行と列を逆にした割り付け，(c)は，行が変わっても PE がずれないようにした割り付け，(d)は，配列要素全部を一列に並べた割り付けを示す．(d)が最も効率がよい割り付け方法で，次に示すループのように DO FOR ALL 文で複数制御変数を同時に指定できる場合に割り付けできる．

```
      DO 10 FOR ALL (I, J)/(1…10). C. (1…10)
   10 A(I, J)=0. 0
```

ここで，.C. は .CROSS. の略で直積を意味し，(I, J) の変域が 1 から 10 のすべての組み合わせであることを表わしている．

(3)　DEFINE 文と OVERLAP 文

上述のように，ILLIAC IV は特殊なメモリ割り付けを行うため，FORTRAN の EQUIVALENCE 文を使用することができない．そこで，EQUIVALENCE 文にかわるものとして DEFINE 文と OVERLAP 文が用意されている．OVERLAP 文は，同時に使用しない配列をメモリ上に重複して割り付けるものである．DEFINE 文は，同一領域を数学的に同値な 2 つ以上の配列に割り付ける場合に使用するものである．

IVTRAN コンパイラの構成を **図-7** に示す．IVTRAN コンパイラは，標準の FORTRAN プログラムを入力して並列処理部分を検出する機能をもっており，図-7 の PARALYZER がこの処理を行う．PARALYZER は，構文解析フェーズ (PARSE) と最適化フェーズ (OPTIMIZER) の中間に位置し，並列処理可能な部分の中間テキストを書き換える．TRANSCRIBER を通すと，書き換えられた中間テキストに対応する IVTRAN ソースプログラムが得られるので，もとのプログラムの中で並列処理可能と判定された部分がどこか一目でわかる．

PARALYZER は，並列処理の可能性がある部分として DO ループをとりあげる．多重の DO ループの場合，隣接する DO の間に文を含まなければ，外側のループも含めて並列処理の対象とみなす．並列処理可能な DO ループとなるためには次の条件を満たす必要がある．

・隣接する DO の間に文を含まないこと．
・DO の増分は定数で指定されていること．
・ループ中の代入文は，配列要素に値を代入するものであること．
・DO の制御変数は，一つの配列参照で 2 カ所以上の添字位置に現われないこと．
・添字式は，I, C, I+C, I−C のいずれかであること (ここで I は制御変数，C はループ不変値とする)
・入出力文を含まないこと．
・ループ外へ分岐しないこと．

・DO ループ内に小ループを作らないこと．

PARALYZER は，並列処理判定に先立って DO ループの書き換えを行う．DO 文間や端末文間の代入文を最内側 DO ループ内へ移す，スカラ変数は削除するか作業用ベクトルを作るなどの書き換えにより，並列処理の対象範囲が広くなる．

書き換えの後，対象となる DO ループの各々に対して並列処理判定を行う．PARALYZER が作り出す IVTRAN プログラムが，もとの FORTRAN と同じ実行結果を得られるならば，並列処理可能であると判定する．DO ループ内に同一配列が2回以上現われ，少なくとも1回は定義されている場合に，並列処理と通常処理で結果が違ってくる可能性がある．したがって，同一配列が2回以上現われるすべてのケースで処理結果が同じであることが保証できる場合に並列処理可能と判定できることになる．

IVTRAN コンパイラは，DO ループ内の同一配列に関して，定義と定義または定義と引用の関係を〈f, g〉集合という形でまとめる．〈f, g〉集合は，2つの配列参照が同一要素を示すときの時間差（指標変数の値の差）を表現するものである．たとえば

```
      DO 9 I=1, 10
      DO 9 J=1, 10
1     A(J, I+1)=…
2       …    =…A(J+2, I)…
9     CONTINUE
```

で文1の A(J, I+1) と文2の A(J+2, I) を比較すると，両方が同じ配列要素を示すのは，文1からみるとIが1小さく，Jが2大きいときである．このときの〈f, g〉集合を (−1, +2) とする．DO ループ内で同一配列が2回以上現れるすべての場合について〈f, g〉集合を計算した後，それをもとに並列処理可能かどうか判定する．

並列処理判定アルゴリズムとして文献 2) では，The Coordinate Method と The Hyperplane Method の両者が紹介されている．The Coordinate Method は，〈f, g〉集合をもとに定義，引用の前後関係を調べ，それが並列処理と通常処理で同じであれば，並列処理可能と判定する方法である．たとえば，

```
      DO 9 I=1, 10, 1
1     A(I+2)=…
2       …    =A(I+1)
3       …    =A(I)
9     CONTINUE
```

では，文1の A(I+2) に対する値の代入は，通常処理でも並列処理でも文2，文3の参照より先行するため，結果は同じである．前後関係が逆転する場合には，文の入れ替えによって前後関係を一致させることができるかどうかが試される．The Hyperplane Method は，多重 DO ループに対してだけ適用できる方法である．本方法は，DO ループの指標変数を別のものにおきかえることによって The Coordinate Method では並列処理不可と判定されるループも並列処理可能にしようというものである．たとえば

```
      DO 5 I=1, 25, 1
      DO 5 J=2, 19, 1
      DO 5 K=2, 29, 1
      A(J, K)=0.25*(A(J+1, K)+
     1 A(J, K+1)+A(J−1, K)+A(J, K−1))
5     CONTINUE
```

というループではこのままでは並列処理不可であるが，

```
L=2*I+J+K
M=I
N=K
```

のように置き換えて，L, M, N を指標変数とする新たな DO ループに変換すると並列処理可能となる．

4.2 ASC における自動ベクトル化方式[7]

テキサスインストルメント社の ASC (Advanced Scientific Computer) は，パイプライン方式による大規模科学技術用計算機で，地震解析，気象モデル，弾道弾解析などにおける利用を目的としたものである．ASC は，最大4個までのパイプライン演算器を持つことができ，これらの並列処理によって高性能を引き出すことができる．また ASC は，スカラ命令に加えてベクトル命令をもっており，これらのベクトル命令がベクトル演算を実行する．

ASC のソフトウェアとしては，NX コンパイラと呼ばれる FORTRAN コンパイラがある．NX コンパイラは，FORTRAN プログラム中の DO ループをそれらと等価なベクトル命令に変換する．変換の過程では，文と文の相互関係をチェックする context analysis と1つの文で左辺と右辺に同一配列が現われる場合のデータ参照関係をチェックする hazard analysis により，ベクトル命令の適用可否が判定される．

ASC のベクトル命令の特徴は，3次元までの配列を1つのベクトルとして表現できることにある．3次元の配列は，次の7つの記号を使って表わせる．

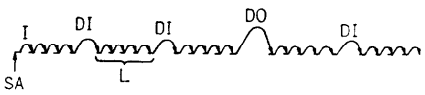


図-8 3次元配列のベクトル表現

(SA, L, NI, NO, I, DI, DO)

ここで SA は先頭アドレス，L, NI, NO はそれぞれ1次元目，2次元目，3次元目の大きさ，Iは1次元目の増分，DI, DO はそれぞれ1次元目から2次元目および2次元目から3次元目へうつるときの変位を表わす（図-8 参照）．

ASC が2つ以上のパイプライン演算器をもっているときに，NX コンパイラは，演算器が遊ぶことなくすべて有効に動作できるような目的プログラムを生成する．

NX コンパイラは，以上示したように FORTRAN の DO ループを自動ベクトル化する機能をもっているほかに，ベクトル演算用に FORTRAN 言語仕様の拡張も行っている．拡張仕様としては，SUBARRAY 文による部分配列の宣言，配列参照の添字の＊指定による配列断面の指定などがある．

4.3 CRAY-1 における自動ベクトル化方式

CRAY-1 では，FORTRAN 77 に近い標準 FORTRAN 言語で書かれたプログラムを解析して，ベクトル命令を含むオブジェクトを出力している．この節では，CRAY-1 FORTRAN (CFT)[10] のベクトル化可能条件を示して，CRAY-1 の自動ベクトル化方式を明らかにする．

まず，CRAY-1 でベクトル化の対象となるのは，最内側の DO ループで，代入文と CONTINUE 文のみからなる DO ループである．IF 文，GOTO 文，CALL 文などを含む DO ループはベクトル化されない．ユーザ定義の関数呼出しは，特別な宣言をし，かつその関数サブルーチンをアセンブラでコーディングすればベクトル化される．IF 文は，後で述べる組み込み関数を使うようにプログラムを書き換えることができれば，ベクトル化される．

DO ループ中に含まれる変数，配列要素のデータ型のうち，実数（1語 64 ビット），整数，複素数，および論理型がベクトル化の対象であり，これらのデータ間の，代入，加減乗除，べき乗の演算がベクトル化される．

基本的な組み込み関数のほとんどがベクトル化される．（以下，総称名で示す），すなわち

SIN, COS, LOG 10, SQRT, EXP, ABS, CONJG, REAL, AIMAG, MAX, MIN, FLOAT, ILIX, SNGL, COMPLX

は，ベクトル命令に変換されるし，

TAN, COT, ATAN, ATAN 2, SINH, COSH, TANH

は，擬似ベクトル命令（スカラ命令でシミュレートされる）に変換される．このほか，ベクトル和や，内積などの特殊演算用の組み込み関数，SSUM, SDOT なども用意されている．また，簡単な条件処理を記述するための一連の組み込み関数，

CVMGP, CVMGM, CVMGZ, CVMGN, CVMGT

もある．これらは，3つの引数をもつ関数で，たとえば，CVMGP の場合，第3引数が正または0ならば，第1引数を，負ならば，第2引数を結果として返す関数である．

次に，ループ中に現われる配列要素の添字は，指標変数と不変変数（ループ相対定数）から作られる以下の形の積和表現に限られる．

〔±C1＊〕INDX〔±C2〕

ここで，C1 は不変変数，C2 は不変式，INDX は指標変数であり，角カッコはオプションを示す．また，多次元の配列要素中では指標変数は高々1回しか現われてはいけない．したがって，添字値がループの進行とともに定値で変化しないいわゆるリストベクトル（間接指標ともいう）は不可能である．

この制約の下にループ内に現われる同一名の配列相互のデータ参照関係の検査が行われる．データ参照関係は，定義と定義ないし定義と引用の関係にある配列要素の間でのみ問題なのであって，どちらも引用のみの場合は検査されない．どちらかの配列要素が定義されている場合，両方の添字の値域とその変化の様子が比較される．両方の配列要素の添字の値域に重なりがなければ，ベクトル化可能になるし，値域が不明の場合はベクトル化不能となる＊．重なりがある場合には，ベクトル化して可能な場合と不能な場合に分類され

…＝A(I−1)+…	A(I−1)＝…
A(I)＝…	…＝A(I)+…
A 未完型	B 破壊型

図-9

＊ ユーザがオプションで指定すればベクトル化可能とすることもできる．

る．不能な場合は，次の2通りのケースに分けられる：

（ここで，Iは一定値で増加している指標変数とする）．同一文では，右辺の参照が左辺の定義に先行するから，たとえば，

A(I)=A(I−1)+…

は未完型のデータ参照違反に分類される．

指標変数を全く含まない配列要素や変数でループ内で定義されるものは，一時的な配列に置き換えられるか，このとき，必ず定義が先行しなければならない．

以上述べたとおり，CRAY-1では，自動ベクトル化のためのごく標準的なデータ参照関係検査とそのほかのチェックを行っている．

4.4 HITAC M-200 H IAP における自動ベクトル化方式[12]

M-200 H IAPの場合も，自動ベクトル化のコンパイル技術としては，CRAY-1と共通点が多い．

まず，対象はCRAY-1同様，最内側DOループのみで，かつ代入文とCONTINUE文のみから成るものである．対象とするデータ型は，実数型（32ビット）と倍精度型（64ビット）のみである．また，ベクトル化可能組み込み関数はSIN, COS, LOG, LOG 10, EXP, SQRTとそれらの倍精度型版，およびDBLEとSNGLであり，前者6つは擬似ベクトル命令としてサポートされる．

添字形式に関する制約もCRAY-1とほぼ同じであるが，ただ多次元配列の要素で，M 200 H IAPでは，各次元に指標変数は1回以内出現が許される．したがって，**図-10**のような，対角要素に関する演算もベクトル化可能である．

同一配列名の異なる配列要素に対するデータ参照関係の検査に関し，添字の共通部分のチェックについてM 200 H IAPでは，添字値の共通集合でみている[3]ため，**図-11**のごときループはベクトル化可能となる．

以上，M 200 H IAPの自動ベクトル化方式をみてきたが，CRAY-1に比較して両者のハードウェアアーキテクチャの相違（CRAY-1ではデータ型の守備範囲が広く，かつベクトル命令の種類も多い；一方，M 200 H IAPでは内積・ベクトル和・イテレーション等の特殊演算命令をもつ）点を除けば，基本的なベクトル化コンパイル技術（特にデータ参照関係のチェック等）では，きわめて良く似た傾向を示している．

```
DO 10 I=1, N
10 A(I, I)=A(I, I)+B(I)
```

図-10 対角要素のベクトル化

```
DO 10 I=1, N, 2
10 A(I)=A(I−1)+…
```

図-11 添字共通部分のチェック

5. むすび

以上，ベクトルプロセッサの利用技術の一環としてその重要性を増しつつある自動ベクトル化について，その方式を概括し，実例をいくつか紹介した．最近では，日立に加えて，三菱[13]，日本電気[14]などの国産メーカでも汎用計算機にベクトルプロセッサを内蔵させているとのことであり，かつてスーパコンピュータにのみ適用されていたこの技術もハードウェアコストの低廉化とともに利用度の高いものとなってきた．利用技術の側面からベクトルプロセッサに関心を持たれる方々に，本解説が幾分かのお役に立てれば幸いである．なお，本稿ではあまりふれる余裕がなかったが，自動ベクトル化とともに，ベクトルプロセッサ向きプログラミング言語ないし既存言語の拡張についても，富士通の230/75アレイプロセッサ用AP-FORTRAN[8]をはじめ，様々な提案や処理系のインプリメントが行われており，この方面では特に，FORTRAN 8Xにおける規格制定に期待されるところが大きい．将来は，仕様拡張と自動ベクトル化が相まってベクトルプロセッサの使い勝手を一層改善してゆくものと予想される．

最後に，できるかぎり事例による客観化に努めたが，筆者等の見識が至らぬために，記述に偏りのあることを懸念している．お気付きの点はご指摘いただければ幸いである．

参考文献

1) 小高俊彦他：超高速演算の動向，情報処理，Vol. 21, No. 9, p. 927 (Sep. 1980).
2) Lamport, L.: Parallel Execution of DO Loops, C. ACM, Vol. 17, No. 2, pp. 83-93 (Feb. 1974).
3) Takanuki, R. et al.: Some Compiling Algorithms for an array processor, Proc. of 3rd UJCC, pp. 273-279 (1978).
4) Presberg, D. L. et al.: The Paralyzer: IVTRAN'S Parallelism Analyzer and Synthesizer, ACM SIGPLAN Notices, Vol. 10, No. 3 pp. 9-16 (Mar. 1975).

5) Erickson, D. B.: Array Processing on an Array Processor, ACM. SIGPLAN Notices, Vol. 10, No. 3, pp. 17-24 (Mar. 1975).
6) Millstein, R. E. et al.: The ILLIAC IV FORTRAN Compiler, ACM. SIGPLAN Notices, Vol. 10, No. 3, pp. 1-8 (Mar. 1975).
7) Wedel, D.: FORTRAN for the Texas Instruments ASC System, ACM. SIGPLAN Notices, Vol. 10, No. 3, pp. 119-132 (Mar. 1975).
8) Miwa et al.: FACOM 230/75 Array Processor System (Japanese), Fujitsu (Jan. 1978).
9) Umetani, Y. et al.: An Analysis on applicability of the Vector operations to scieatific programs and the determination of an effective instruction repertoire, Proc. of 3rd UJCC pp. 331-335 (1978).
10) CRAY-1 FORTRAN (CFT) 説明書, SM-007, CRC. (1980).
11) CRAY-1 ユーザーズガイド, SM-009, CRC. (1980).
12) VOS 2/VOS 3 最適化 FORTRAN 77 使用の手引, 8080-3-258, 日立製作所, p. 351 (1980).
13) MELCOM UTS/VS 拡張 FORTRAN 使用手引書, NM-SR 00-93 A, 三菱電機 (1980).
14) 大型コンピュータ ACOS 1000 の高速化技術とアーキテクチャ, 日経エレクトロニクス (1981. 5. 11).
15) Control Data Corporation: STAR-100 & STAR-100 Mainframe, Auerbach Computer Technology Reports (1981).
16) Control Data Corporation: Cyber 203 & Cyber 203 Mainframe, Auerbach Computer Technology Reports (1981).
17) Control Data Corporation: Cyber 205, Auerbach Computer Technology Reports (1981).

(昭和56年7月24日受付)