Vol. 24 No. 4 Jul. 1983

汎用計算機のための条件制御ベクトル演算方式†

堀越 彌†† 梅谷征雄††

より高い性能の汎用計算機を実現するため，すでに報告したように，科学技術計算用途を念頭においた内蔵ベクトル演算機構と自動ベクトル・コンパイラからなる内蔵ベクトル演算方式を開発し実用化した．しかし，この方式をより効果あらしめるには，ソース・プログラムからベクトル・コードに変換される割合，ベクトル化率をより高める必要がある．このため今回新たに，従来不可能だった IF 文(条件文)を含む DO ループをベクトル化する条件制御ベクトル演算方式を開発した．制御ベクトルの制御下にハードウェア機構としては，演算を選択的に実行する 26 の条件制御ベクトル命令と，これら制御ベクトルを取り扱う 12 の制御ベクトル生成命令を内蔵ベクトル演算機構に追加した．また，ソフトウェア機能としては，IF 文を含む DO ループを上記命令列に変換する機能，IF 文を局在化して残りの部分をベクトル化する機能などを自動ベクトル・コンパイラに付加した．この方式を HITAC M-200 H で実験したところ，50% の演算実行比率を前提にしたループ・ミクスで 2.74～3.06 倍の性能改善効果が確認できた．また，2 次元拡散プログラムの例では，従来のベクトル演算方式よりさらに 1.64 倍の性能向上が図られた．これらの結果をもとに，HITAC M-280 H で条件制御ベクトル演算方式を実用化した．

1. まえがき

より高い性能の計算機を実現する一つの手段として，とくに科学技術計算分野では，ベクトル演算機能が注目を浴びつつある．ベクトル演算機能とは，二つの演算語（ソース・オペランド）間の演算をそのつど指定する従来のスカラ命令形式に対して，演算要素群（ベクトル・オペランド）間の演算を1命令で指定するベクトル命令を導入し，ハードウェアによるより高度な並列処理の可能性を開き，より高いシステム性能を実現しようというものである．この方式はいわゆる科学技術専用計算機，スーパーコンピュータでは以前から用いられたものであるが，筆者らは汎用計算機用の内蔵ベクトル演算方式を開発し，これを HITAC M-180/200H/280H で実用化した結果をすでに報告した[1)~4)]．

さて，このようなベクトル演算方式には，専用機，汎用機を問わず共通した課題がいくつかある．その第1のものは，ベクトル命令で記述されたプログラムの部分はベクトル演算機能を有する計算機により十分高速に処理できるが，従来からのスカラ命令で記述された部分は顕著な改善がないということである．このことが一般に次の二つの技術進歩を要請することになる．

（1） より多くのプログラムがベクトル・コードに変換できるような（高いベクトル化率）アーキテクチャ技術，プログラミング技術を開発すること．

（2） より高速のスカラ・コード処理を可能とする処理装置（プロセッサ）技術を開発すること．

（2）の課題は，一般の汎用計算機の高速化を期待しているわけであり，すでに広範な努力が積み重ねられているのはいうまでもない．したがって，ベクトル演算技術の分野では（1）の高いベクトル化技術の開発が中心になるといってよい．この意図に従って，今回 DO ループ内に条件文を含む場合でもベクトル・コードに展開できる条件制御ベクトル演算方式を開発した．本稿では，条件制御ベクトル演算方式のアーキテクチャ(論理仕様)，処理装置方式，コンパイラ方式について述べ，HITAC M-200H 内蔵ベクトル演算機能(IAP 機能：Integrated Array Processor 機能とも略す）に実験的に付加した結果について報告する．

2. 条件制御の必要性

DO ループ内に IF 文を含む場合にもベクトル化の可能性を開く条件制御ベクトル演算方式の第1の目的は，前章に述べたようにベクトル演算方式にとって最も重要なベクトル化率の向上を図ることにある．文献 1) においてすでに報告したように，科学技術計算の 10～17% が条件文を含む DO ループを含む．この部分のベクトル化はそれだけベクトル化率の向上をもたらし，よりベクトル演算方式の高速性を実現することになる．条件制御ベクトル演算方式の第2の目的は，

† Conditional Vector Arithmetic Facility for General-Purpose Computer by HISASHI HORIKOSHI and YUKIO UMETANI (Central Research Laboratory, Hitachi Ltd.).

†† (株)日立製作所中央研究所

プログラムを書きやすくすることである．なぜなら，IF 文を含む DO ループがベクトル化されない従来のベクトル演算方式では，性能に敏感なプログラマは，IF 文を可能な限り DO ループの外に追い出すような配慮をするため，自然なプログラミングができなかった．条件制御ベクトル演算方式の成功によりこの欠点は取り除かれることになる．

以下では，IF 文を含む DO ループにはどのような種類があり，それぞれのベクトル化にソフトウェア，ハードウェアのどのような機構が考えられるかを明らかにする．

（1） ループ不変条件を含む DO ループ：**図1**に示す DO ループがこの種類に属し，本来この IF 文は DO ループの外にあってもよい場合である．この DO ループをベクトル化するためには，コンパイラがこの不変条件を検出し，等価的に図の下側に示すプログラムに変更し，DO ループ部をベクトル化すればよい．

（2） 端点指標条件を含む DO ループ：しばしば見られる形式の DO ループで，**図2**に示すように，指標の片端のみで演算内容が異なる場合である．端点のみループ外で演算するように変更すれば，図2に下に示すようなプログラムになり，残された DO ループはベクトル化できることになる．これもコンパイラの方式的拡張により可能である．

（3） ループ可変条件を含む DO ループ：**図3**に

```
      DO 10 I=1, N
      IF (J.EQ.K) GO TO 10
      Z(I)=X(I)+Y(I)
   10 CONTINUE
            ⇩
      IF (J.EQ.K) GO TO 20
      DO 10 I=1, N
   10 Z(I)=X(I)+Y(I)
   20 CONTINUE
```

図 1 ループ不変条件を含む DO ループの例
Fig. 1 An example of DO Loop which includes a condition independent of its loop operation.

```
      DO 10 I=1, N
      IF (I.EQ. 1) D(I)=E(I)+F(I)
      Z(I)=X(I)+Y(I)
   10 CONTINUE
            ⇩
      D(1)=E(1)+F(1)
      Z(1)=X(1)+Y(1)
      DO 10 I=2, N
   10 Z(I)=X(I)+Y(I)
```

図 2 端点指標条件を含む DO ループの例
Fig. 2 An example of DO Loop which includes a terminal index condition.

```
      DO 10 I=1, N
      IF (A(I).GT.B(I)) THEN
          D(I)=E(I)+F(I)
        ELSE
          Z(I)=X(I)+Y(I)
      ENDIF
   10 CONTINUE
```

図 3 ループ可変条件を含む DO ループの例
Fig. 3 An example of DO Loop which includes a loop-dependent condition.

```
      DO 10 I=1, N
      IF (A(I).GT.B(I)) D(I)=E(I)+F(I)
   10 Z(I)=X(I)+Y(I)
            ⇩
      DO 10 I=1, N
   10 IF (A(I).GT.B(I)) D(I)=E(I)+F(I)
      DO 20 I=1, N
   20 Z(I)=X(I)+Y(I)
```

図 4 ループ可変条件を含むもう一つの DO ループの例
ループ分割によりベクトル化が可能になる．
Fig. 4 An example of DO Loop which includes a loop-dependent condition and needs loop-partitioning before vectorization.

示すようにループ指標の値に依存して条件の成否が変化し，それによって演算内容が変化するものである．これをベクトル化するには，条件を制御ベクトルと呼ばれる条件の成否の“0”，“1”からなるベクトルに変換するハードウェア命令と，この制御ベクトルの制御下に各ベクトル要素の演算の非実行/実行を選択するハードウェアの条件ベクトル命令が必要になる．**図4**に示す例では，上記ハードウェア命令だけでは効率的にベクトル化できないので，図の下半に示すようなプログラムにコンパイラで等価変換することによりベクトル化が可能になる．

今回開発した条件制御ベクトル演算方式は，自動ベクトル・コンパイラと内蔵ベクトル演算機構の働きにより，上記(1)～(3)のすべてがベクトル化の対象となっている．これによってベクトル化されない IF 文を含む DO ループには，整数条件を含む IF 文などがあるが，これはハードウェアがその種の命令を備えていないことによるもので，方式的に可能な範囲はほぼすべて実現しているといえる．

3. 条件制御ベクトル演算方式

3.1 方式の概要

条件制御ベクトル演算方式は，1，2章で述べてきたように，IF 文を含む DO ループをベクトル化しようというもので，ハードウェア的には演算の直接対象となる二つの演算ベクトル X, Y 以外に第3の制御ベク

Vector Elementwise Add with Control Vector
VEAE with C の動作

$$\begin{cases} Z_0 \leftarrow X_0 + Y_0 & \text{if } C_0=1 \\ Z_0 \text{ 変化せず} & \text{if } C_0=0 \end{cases}$$
$$\begin{cases} Z_1 \leftarrow X_1 + Y_1 & \text{if } C_1=1 \\ Z_1 \text{ 変化せず} & \text{if } C_1=0 \end{cases}$$
$$\vdots$$
$$\begin{cases} Z_i \leftarrow X_i + Y_i & \text{if } C_i=1 \\ Z_i \text{ 変化せず} & \text{if } C_i=0 \end{cases}$$

図 5 制御ベクトルの制御を受けるベクトル命令の動作原理
Fig. 5 Operational description of vector instructions with a control vector.

トルCを導入し，Cの対応するビット C_i の“0”,“1”により X_i, Y_i 間の演算の実行を制御する方式である．VEAE (vector elementwise add) を例としてこの命令が制御ベクトルの制御を受ける場合を示すと**図5**のようになる．すなわち，従来，単純なベクトル間の演算に限られたベクトル命令に，より複雑な制御を許すことにより，その用途を拡大し，ベクトル演算機能の利用者により高い性能を約束しようというものである．

これを可能にするためにはまずソフトウェアとハードウェアのインタフェースとしてのアーキテクチャ（論理仕様）を定義する必要がある．従来，汎用計算機向きのアーキテクチャとして開発した，HITAC M-180/200H 用の内蔵ベクトル演算機能を拡張して，条件制御ベクトル演算方式を導入した．これについて，3.2節で述べる．また，このアーキテクチャを HITAC M-200H で実験的にインプリメント（実現）し，HITAC M-280H で実用化した．これについて3.3節で述べる．また，このアーキテクチャを支援するソフトウェアとして，既存の FORTRAN コードから自動的に条件制御ベクトル命令に落とす，自動ベクトル・コンパイラを開発した．これについて3.4節で述べる．

3.2 アーキテクチャ（論理仕様）

条件制御ベクトル演算機能のアーキテクチャは，HITAC M-180/200H/280H 用の内蔵ベクトル演算機能に，制御ベクトル機能を付加することにより実現した．拡張は，**表1**に示すような(1)六つの制御ベクトルを生成する命令の新設，および，**表2**に示すような28個のベクトル演算命令のうち，(2) 26個のベクトル演算命令への条件制御の付加によって行った．制御ベクトル生成命令は表1に示すように，データの各種の大小関係より制御ベクトルを生成するもので，現在のところは浮動小数点データのみ

表 1 制御ベクトル生成命令一覧
Table 1 Summary table of newly added vector instructions to generate control vectors.

ベクトル命令の名称	略称		演算語				動作概要
	短精度	倍精度	1	2	3	C*	
Vector compare equal	CEQE	CEQD	—	S/V	S/V	g	$C_i \leftarrow$ if $X_i = Y_i$
Vector compare not equal	CNEE	CNED	—	S/V	S/V	g	$C_i \leftarrow$ if $X_i \neq Y_i$
Vector compare greater or equal	CGEE	CGED	—	S/V	S/V	g	$C_i \leftarrow$ if $X_i \geqq Y_i$
Vector compare greater than	CGTE	CGTD	—	S/V	S/V	g	$C_i \leftarrow$ if $X_i > Y_i$
Vector compare less or equal	CLEE	CLED	—	S/V	S/V	g	$C_i \leftarrow$ if $X_i \leqq Y_i$
Vector compare less than	CLTE	CLTE	—	S/V	S/V	g	$C_i \leftarrow$ if $X_i < Y_i$

* 演算語の制御ベクトル (C) 欄の g は，この命令により制御ベクトルが生成されることを示す．

表 2 コントロール・ベクトルの制御を受けるベクトル命令
Table 2 Summary table of vector instructions conditionally executed under control vectors.

ベクトル命令の名称	略称		演算語の S/V*			制御**ベクトル可否	動作概要
	短精度	倍精度	1	2	3		
Vector move	VME	VMD	V	S/V		C	$Z_i \leftarrow X_i$
Vector elementwise complement	VECE	VECD	V	S/V		C	$Z_i \leftarrow -X_i$
Vector elementwise add	VEAE	VEAD	V	S/V	S/V	C	$Z_i \leftarrow X_i + Y_i$
Vector elementwise subtract	VESE	VESD	V	S/V	S/V	C	$Z_i \leftarrow X_i - Y_i$
Vector elementwise multiply	VEME	VEMD	V	S/V	S/V	C	$Z_i \leftarrow X_i * Y_i$
Vector elementwise divide	VEDE	VEDD	V	S/V	S/V	C	$Z_i \leftarrow X_i / Y_i$
Vector element sum	VSME	VSMD	S	S/V		C	FPR$\leftarrow$FPR$+\Sigma X_i$
Vector element sum with complement	VSMCE	VSMCD	S	S/V		C	FPR$\leftarrow$FPR$-\Sigma X_i$
Vector inner product	VIPE	VIPD	S	S/V	S/V	C	FPR$\leftarrow$FPR $+\Sigma X_i * Y_i$
Vector inner product with complement	VIPCE	VIPCD	S	S/V	S/V	C	FPR$\leftarrow$FPR $-\Sigma X_i * Y_i$
Scalar multiply and add	VSMAE	VSMAD	V	S	S/V	C	$Z_i \leftarrow Z_i + X * Y_i$
Scalar multiply and subtract	VSMSE	VSMSD	V	S	S/V	C	$Z_i \leftarrow Z_i - X * Y_i$
First order iteration	VITRE	VITRD	V	S/V	S/V		$Z_{i+1} \leftarrow X_i + Y_i * Z_i$
Convert double to single	VCVDE		V	S/V		C	Z_i (Single) $\leftarrow X_i$ (Double)
Convert single to double		VCVDE	V	S/V		C	Z_i (Double) $\leftarrow X_i$ (Single)

* 命令語の OP 拡張部により，演算語の S (Scalar)/V (Vector) を指定できる．
** 命令語の OP 拡張部により，C (制御ベクトル) の制御を受けることを指示する．

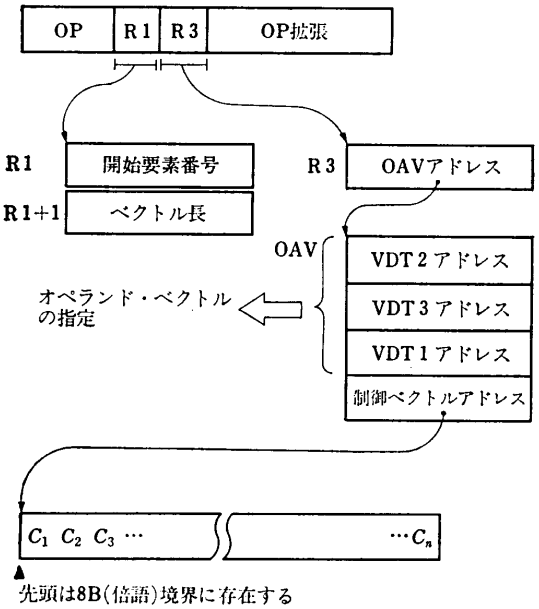


図 6 制御ベクトルを用いるベクトル命令の形式
Fig. 6 Formats of the vector instructions which are controlled by a control vector.

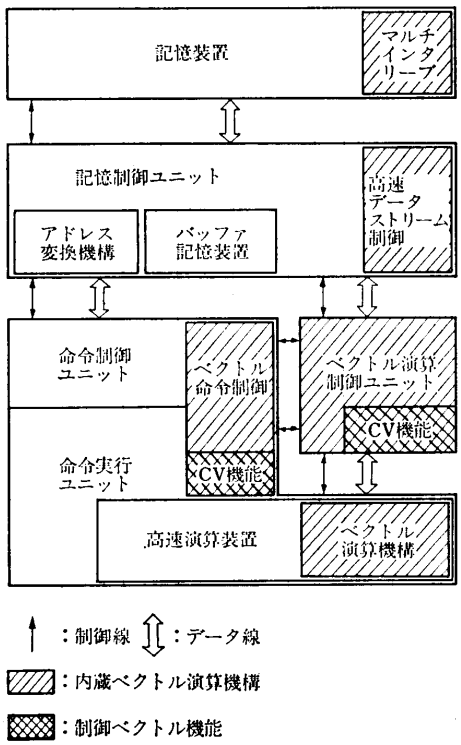


図 7 HITAC M-200 H の内蔵ベクトル演算機構と制御ベクトル機能
Fig. 7 Machine organization of the HITAC M-200 H with the integrated vector arithmetic feature and the control vector function.

許している．一方，条件制御ベクトル命令は表2に示すように，ほぼすべての既存のベクトル命令がこの機能をもつように拡大されているが，VITRE, VITRD の二つの First Order Iteration 命令だけは適用からはずされている．これは，演算の性格上，中途の演算のみ抑止しても意味がないと判断したことによる．

各命令での制御ベクトルの指定は**図6**のように行う．すなわち従来は命令のR3部で指定された汎用レジスタ内のデータを OAV (Operand Address Vector) のアドレスとして用い，記憶装置内の OAV 領域には上から VDT 2 アドレス (第2オペランドベクトルを指定する VDT 2 のアドレス，VDT: Vector Descriptor Table)，VDT 3 アドレス，VDT 1 アドレスを配列していたが，今回新たに VDT 1 アドレスの下に 32 ビットの制御ベクトル・アドレスを付加した．制御ベクトルは各ベクトル要素対応に左から1ビットずつ詰め込まれているが，ベクトル演算という高速性に配慮して8バイト (64 ビット) 単位に取り扱うこととした．

3.3 処理装置の方式

ベクトル演算に条件制御を導入するために，**図7**に

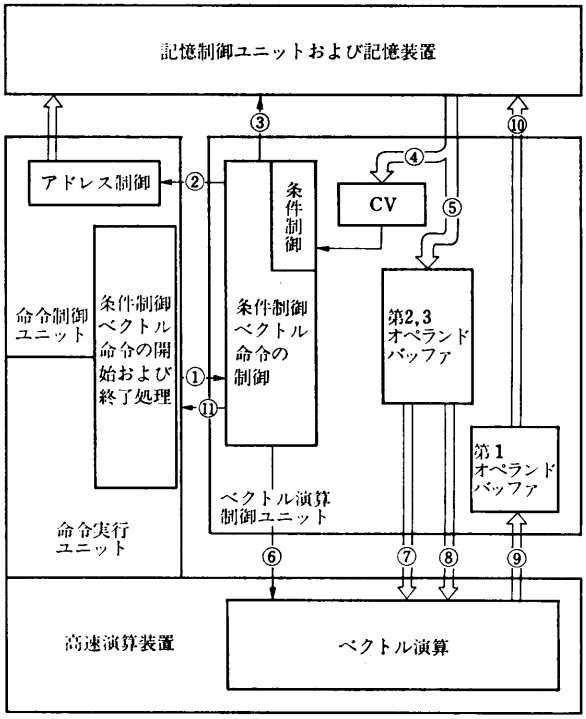


図 8 条件制御処理の動作概念図
Fig. 8 Operational description of the conditional arithmetic operation.

示すように命令実行ユニットおよびベクトル演算制御ユニットに制御ベクトル（以下CVと略す）機能を実験的に付加した．**図8**に条件制御処理の処理の流れを示す．図8を用いてより詳細に動作を説明する．

（1） 命令制御および命令実行ユニットにおいて，ベクトル命令の開始が完了すると，ベクトル演算制御ユニットに制御が渡される．

（2） ベクトル演算制御ユニットでは読みだすべきベクトル，CV等のアドレスを命令制御ユニットに準備させ，

（3） 同時に記憶制御ユニット，および記憶装置に読みだしを指示する．

（4） 読みだされたCVはベクトル演算制御ユニット内の64ビットのCVレジスタに，

（5） ベクトル・オペランドは第2，第3用のオペランド・バッファに格納される．

（6） ベクトル演算制御ユニットからは，ベクトル演算実行の指示が出る．CVの対応ビットが“0”の場合には実際の演算は何も行わない．一方，CVの対応ビットが“1”の場合には演算の実行が指令され，

（7） 第2オペランド

（8） 第3オペランドが送出される．

（9） 演算結果は第1オペランド・バッファに累積される

(10) 結果が第1オペランド・バッファに到着すると，ベクトル演算制御ユニットは，(2),(3)と同様の手順で，記憶制御ユニットおよび記憶装置にベクトル・オペランドの書込みを指示し，データを送出する．対応するベクトル要素の処理中になんらかの例外要因（ページ・フォルト，オーバ・フロー等）が発生した場合には，そのフラッグがデータに付随して転送されており，この最後の段階で割込みを発生する．この場合，一般には結果の格納は抑止される．一方，CVの対応ビットが“0”であった場合には，結果として得られる第1オペランドも存在しないので書込みは抑止される．また，このベクトル要素に関係して発生していた割込み要因もあったとしても捨てられることになる．

3.4 自動ベクトル・コンパイラの条件文処理方式

自動ベクトル・コンパイラは，FORTRANやFORTRAN 77言語で記述されたプログラムのDOループ部分を内蔵ベクトル演算機構用のベクトル命令に自動的に変換する．従来の自動ベクトル・コンパイラが条件文による制御分岐を内部に含むDOループのベクトル化を行わなかったのに対し，条件制御機能を有するコンパイラは制御ベクトルなどを用いてベクトルコードを生成し，ベクトル化の範囲を拡大している．

図9にFORTRAN 77言語により記述した条件文を含むDOループ例と，コンパイラの出力するオブジェクトコードを示す．オブジェクトコードの中ほどにあるCGTEが，文番号4の条件文に対応する制御ベクトル生成命令であり，その5行下のVME＊が，文番号5に対応する条件制御ベクトル命令である．その他の命令は，ループ長制御，ベクトル長計算，ベクトル命令セットアップなどのためのスカラ命令である．**表3**にはこのようなベクトル化が可能であるためのFORTRAN 77言語における条件を示す．＊印が条件文の処理に関連する制約である．

条件制御を行う場合注意を要する点は，制御ベクトル下に演算を行うベクトル命令の性能が，一般には制

ソース・プログラム

```
00001                PROGRAM TEMP1
00002                REAL A(5000),B(5000),C(5000)
00003                DO 10 I=2,N
00004                IF(C(I).GT.0.0) THEN
00005                B(I)=A(I)
00006                END IF
00007             10 CONTINUE
00008                STOP
00009                END
```

オブジェクト・プログラム

```
58A0 D33C                  100000    L     10,N
5890 D31C                            L     9,=F'1'
5870 D320                            L     7,=F'2'
182A          00003                  LR    2,10
1B27                                 SR    2, 7
1A29                                 AR    2, 9
47D0 C05E                            BC    13,#100001
180A                       100002    LR    0,10
1809                                 SR    0, 9
5000 D360                            ST    0,.A0009
1800          00004        100003    SR    0, 0
5810 D360                            L     1,.A0009
41F0 D2E8                            LA    15,.DA001
A40F 024B                            CGTE  0,15
1800          00005        100004    SR    0, 0
5810 D360                            L     1,.A0009
58E0 D32C                            L     14,=F'8192'
41F0 D2F8                            LA    15,.DA002
A40F E060                            VME*  0,15
18BA          00008        100006    LR    11,10
1AB9                                 AR    11, 9
58F0 C008                  100001    L     15,=V(F*RCON)
45E0 F034                            BAL   14,  52( 0,15)
05                                   DC    XL1'05'
```

図9 自動ベクトル・コンパイラによる条件文の処理例
Fig. 9 An example of conditional statement processed by the automatic vectorization compiler.

表 3 条件制御機能を有する自動ベクトル・コンパイラにおける DO ループのベクトル化条件
Table 3 Conditions which have to be satisfied for a DO loop to be vectorized.

条件の種類	ベクトル化のための必要条件
(1) 文の種類 (*印は条件制御機能)	代入文, CONTINUE 文, GO TO 文*, 論理 IF/算術 IF/ブロック IF/ELSE/ELSE IF/ENDIF の各文*
(2) 制御構造*	制御移動 (GO TO) による内部ループがない* 逆方向およびループ外への制御移動がない* 多重ブロック IF で表現可能なもの*
(3) データ型式	実数型および複素数型の単精度または倍精度
(4) 配列添字型式	定添字または線形添字
(5) データ参照関係	添字関係がベクトル化に整合していること

御ベクトルのパタンに依存せずほぼ一定の時間を示すことである．したがって実際の演算がほとんど行われないような多方向分岐ではむしろスカラ命令のほうが有利になる可能性がある．この問題には次の方針で対処した．

(a) ループ展開による端点条件式の除去

I をループ制御に用いられる指標変数 (DO 10 I=1, N の I など) とすると，図2のように指標変数の初期値ないし終値に依存する条件式 (IF(I.EQ.1) など) がしばしば用いられる．このようなループ端点条件式に対しては，ループの初回ないし終回の演算を前後に展開し，ループ中の指標変数の変域を制限することにより条件の成否を確定し，条件式の除去と制御の構造の節約を図る．

(b) ループ不変条件式の分岐命令によるサポート

図1のように条件の成否がループの進行にかかわらず確定しているループ不変条件式は制御ベクトルを生成しても全ビット'1'ないし'0'であり，無駄であるので，従来と同様に判定と分岐命令による分岐制御を行う．

(c) 多重のループ可変条件部のベクトル化抑止

制御ベクトルの合成の必要なループ可変条件が多重に働くような部分はベクトル演算による性能改善はあまり期待できないので，ベクトル化を抑止する．

図10に自動ベクトル・コンパイラの主要構成を示す．ベクトル化モジュールおよびその子モジュールがベクトル化全般の機能を司り，二重枠で囲った制御構造解析モジュール，ループ展開モジュール，ループ分割モジュール，ベクトル中間語生成モジュールがとくに条件制御機能に重要な役割を演じている．

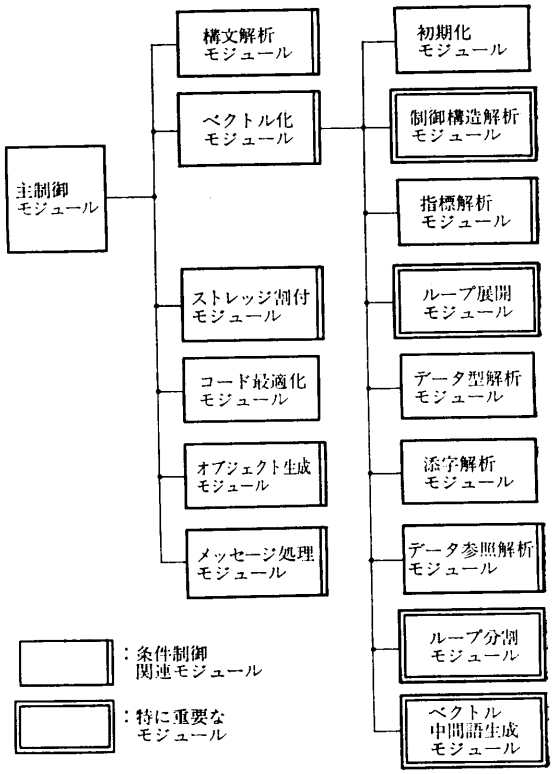


図 10 条件制御機能を有する自動ベクトル・コンパイラの構成
Fig. 10 Module structure of the Automatic Vectorization Compiler with conditional control function.

制御構造解析モジュールは DO ループ内の制御の流れと制御の支配関係の解析を，ループ展開モジュールは端点条件式除去のためのループ展開を，ループ分割モジュールはベクトル化不能の部分を他から分離し残る部分のベクトル化の促進を，ベクトル中間語生成モジュールは制御ベクトルなどを用いた中間コードの生成をそれぞれ行う．このうち，制御構造解析，ループ展開，ベクトル中間語生成部の条件文処理の大略を図

```
1       DO 10 I=1, N
2           IF (C(I).GT.0.0) THEN
3               IF (C2(J).GT.1.0) THEN
4                   A(I)=B(I)+D(I)
5               ENDIF
6           ELSE
7               IF (I.EQ.1) THEN
8                   E(I)=F(I)*G(I)
9               ENDIF
10              X(I)=Y(I)+Z(I)
11          ENDIF
12  10  CONTINUE
```

図 11 条件文を含む DO ループの例 (FORTRAN 77)
Fig. 11 An example of DO loop which includes conditional statements.

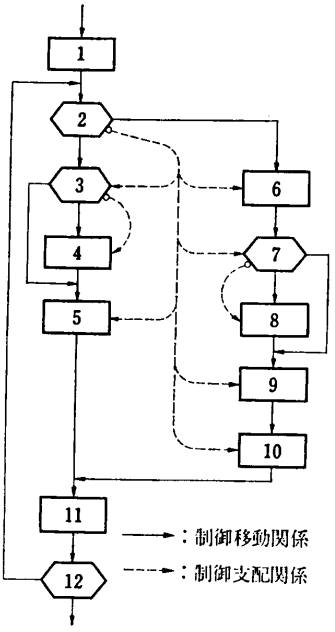


図 12 例題プログラムの制御構造解析結果
Fig. 12 Control structure of the previous program resulted from control flow analysis of the Automatic Vectorization Compiler.

11 の例題に即して述べる．図11は三つの条件文を含むが，文番号2は制御ベクトルの必要なループ可変条件文，文番号3はループ不変条件文，文番号7はループ端点条件文である．

（1） 制御構造解析

制御構造解析モジュールでは，構文解析の結果として得られる中間語 (Intermediate Text) と制御テーブル類を入力として，中間語ブロックとよぶ制御単位を類別し，また制御単位間の制御移動関係と支配関係を解析し，**図12**のような中間語ブロックの関連図を記憶装置上のテーブルに生成する．

図12にて，枠内の数字はブロックを同定する文番号を，実線の矢印は制御の移動関係を，点線の矢印は制御の支配関係を表現している．このモジュールではまた，内部ループあるいは，ループ外分岐などのベクトル化を阻害する制御構造も検出する．

（2） ループ展開

ループ展開モジュールでは，ループ中に端点条件式がある場合に，端点処理のループ外への展開とそれに伴う端点条件式の消去，制御構造の簡約化を実施する．**図13**を用いてこの手順を述べる．

この例で消去の対象になるのはブロック7の条件式である．初回，$I=1$ の処理を中間語の複写によりループの外へ展開することにより 7 は 9 への無条件 GO TO に変わり，ブロック8への制御パスが断ち切られる((a)→(b))．この影響を伝えるために全ブロックを操作して制御移動元のないブロックを捜し，そこから制御移動先へのパスを切る((b)→(c))．この例では，ブロック8のみが消去され(c)の状態が得られる．制御移動係の更新に合わせて制御支配関係も更新する．

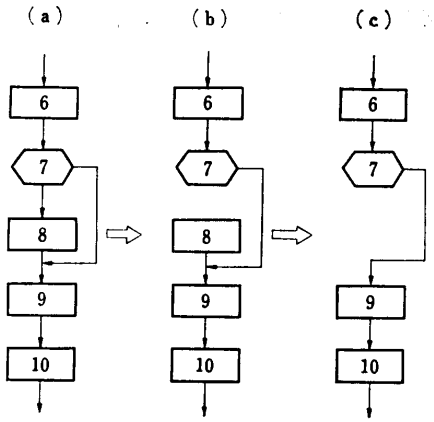


図 13 ループ展開処理による制御構造の簡約化
Fig. 13 Simplification of the control structure by the loop expansion process of the Automatic Vectorization Compiler.

（3） ベクトル化中間語生成

ベクトル化中間語の生成は，小反復化とよぶ制御構造の変更操作と中間語の生成に論理的に分けられる．実際の処理は両者を同時に進めるが説明の都合上分けて述べることにする．

小反復化とは，本来のループ演算をベクトル命令を適用するためにブロック単位の小ループの列に分解する操作を言う．ループ可変条件ブロックと演算を含む非条件ブロックが小反復の対象となる．すなわち**図14**において，ブロック2には新ブロック 20, 21 とループ終了の判定ブロック 12 を追加し，制御ベクトルを生成するための小ループを形成する．同様に，非条件演算ブロック 4, 10 の前後にも初期ブロック終端ブロックを挿入し，小ループを形成する．ループ不変ブロック 3, 7 と非演算ブロック 5, 6, 9, 11 のまわりの制御構造は不変とする．また最後に位置していた終端ブロックを除去する．ブロック2から6への制御移動は5から行うように変更する．小反復化に基づきブロック単位，または複数のブロックを含む小ループ単位にベクトル化中間語を作る．ブロック2〜12の群に対しては，ベクトル C1 と 0.0 の比較により，成立側の制御情報を代表する制御ベクトル CV1 を生成する中

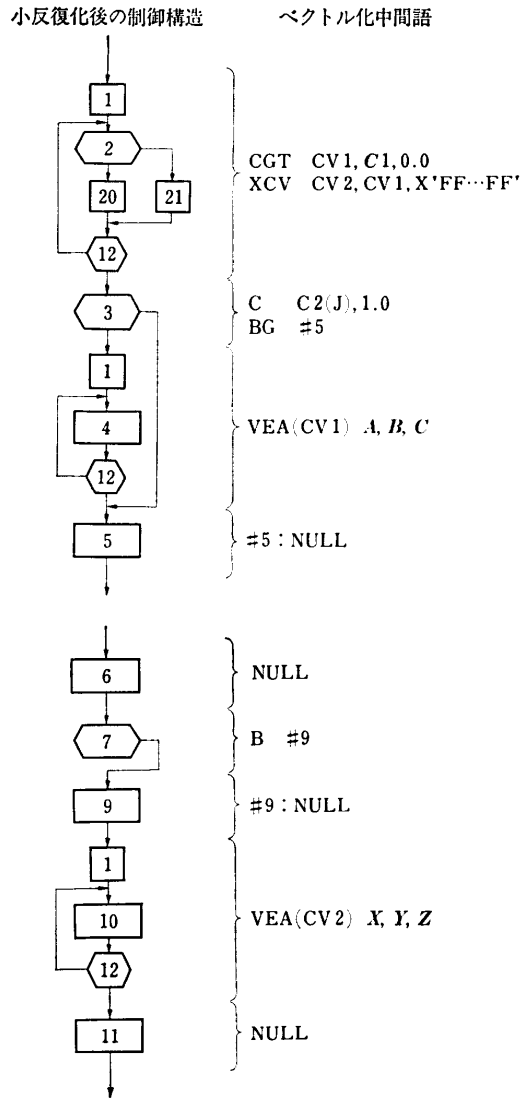


図 14 小反復化とベクトル化中間語の生成
Fig. 14 Intermediate language generation by introducing minor DO loop structure.

間語 CGT，およびその補数をとり非成立側の制御ベクトル CV2 を生成する中間語 XCV を作る．ブロック 3, 7 に対しては従来どおりのスカラ命令による比較，分岐中間語を作り，4 群と 10 群に対してはそれぞれを支配する制御ベクトルを用いたベクトル演算中間語を作る．そのほかの非演算ブロックに対してはダミー中間語 (NULL) を作る．こうしてベクトル化中間語の生成を終える．

以上，条件制御機能を有する自動ベクトル・コンパイラの機能，設計指針，構成，処理方式の大略を述べた．

今後は，プロセッサ演算性能の向上に合わせて多重ループ可変条件のベクトル化抑止を解くと同時により広い制御構造に対してもベクトル化を実現する必要がある．

4. 条件制御ベクトル演算方式の評価

4.1 基本性能の評価

IF 文を含む DO ループのベクトル化を特徴とする，条件制御ベクトル演算方式の一般的評価は必ずしも容易ではないが，まず基本的効果を把握するための，ベクトル演算の性能評価用に設定したループ・ミクスをこの用途に拡張することとした．ループ・ミクスは 2 項，内積，3 項，4 項の DO ループの各 1 回当りの演算時間を求め，これを 20, 35, 15, 30% の重みで平均化し，1 ループ当りの平均実行時間を求めようというものである．条件制御評価用のループ・ミクスでは，これに**表 4** に示すように IF 文を追加した．おのおのの IF 文の成立比は，A(I)，B(I) の値を

表 4 条件ループ・ミクスの定義
Table 4 Definition of the Conditional Loop Mix.

ループ名	重み(%)	FORTRAN 文
2項	20	DO 10 *I*=1, *N* IF (*A*(*I*).GT.*B*(*I*)) GO TO 10 *Z*(*I*)=*X*(*I*)+*Y*(*I*) 10 CONTINUE
内積	35	DO 10 *I*=1, *N* IF (*A*(*I*).GT.*B*(*I*)) GO TO 10 SUM=SUM+*X*(*I*)＊*Y*(*I*) 10 CONTINUE
3項	15	DO 10 *I*=1, *N* IF (*A*(*I*).GT.*B*(*I*)) GO TO 10 *Z*(*I*)=*Z*(*I*)+*X*(*I*)＊*Y*(*I*) 10 CONTINUE
4項	30	DO 10 *I*=1, *N* IF (*A*(*I*).GT.*B*(*I*)) GO TO 10 *Z*(*I*)=*Z*(*I*)−(*E*＊*X*(*I*)+*F*＊*Y*(*I*)) 10 CONTINUE

表 5 条件ループ・ミクスの測定値
Table 5 Performance evaluation by the Conditional Loop Mix.

ループ名		重み(%)	(μs/loop) Scalar	(μs/loop) Vector	相対比 S/V
単精度	2 項	20	0.940	0.232	4.04
	内 積	35	1.015	0.263	3.86
	3 項	15	1.029	0.340	3.02
	4 項	30	1.156	0.507	2.28
	平 均	100	1.044	0.342	3.06
倍精度	2 項	20	1.095	0.321	3.41
	内 積	35	1.177	0.315	3.73
	3 項	15	1.233	0.441	2.79
	4 項	30	1.395	0.698	2.00
	平 均	100	1.234	0.450	2.74

注) ループ長: 64，アドレス連続，条件成立比 50% の場合

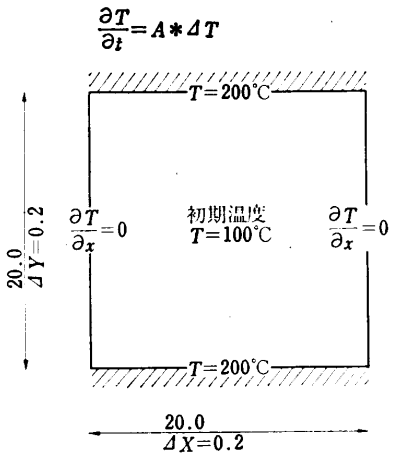
変更することにより調整可能である．**表5**は HITAC M-200H の内蔵ベクトル演算機構に実験的に組み込んだ条件制御機構と，やはり条件制御機能を有する自動ベクトル・コンパイラを用いて実測した結果である．条件ループ・ミクスは，一般のループ・ミクスと同様に，

（1） ベクトル長 (DO ループの繰返し回数)，

（2） ベクトルが記憶装置上の連続領域に格納されているかいないか，などによりその性能が影響を受けるが，さらに，

（3） 条件成立比の影響も加わる．

表5は，ループ長64，連続アドレス，条件成立比50% の場合の測定結果を，浮動小数点データが単精度 (32ビット)，倍精度 (64ビット)の場合について，それぞれ示したものである．ベクトル化により，条件ループ・ミクス値が単精度の場合で 1.044 μs から 0.342 μs に 3.06 倍，倍精度の場合で 1.234 μs から 0.450 μs に 2.74 倍の改善となっており，満足のいく



時間 (0, 10.0) の内部領域の温度分布を時間刻み，$\Delta t=0.02$ で求める．

図 15 条件制御ベクトル演算方式の効果を評価するための2次元熱拡散問題

Fig. 15 A two-dimensional thermal diffusion problem to evaluate effectiveness of the conditional vector arithmetic facility.

```
SOURCE STATEMENT

      DIMENSION DT(100,100),DXY(100,100)
      T=0.0
      DX=0.2
      DY=0.2
      DTIME=0.02
      TE=10.0
      A=0.62
      DO 20 I=1,100
         DO 10 K=2,99
            DXY(I,K)=100.0
   10    CONTINUE
         DXY(I,1)=200.0
         DXY(I,100)=200.0
         DT(I,1)=200.0
         DT(I,100)=200.0
   20 CONTINUE
  100 DO 40 K=2,99
         DO 30 I=1,100
          IF (I.EQ.1) GO TO 31
          IF (I.EQ.100) GO TO 32
          DT(I,K)=DXY(I,K)+DTIME*A*((DXY(I+1,K)-2.0*DXY(I,K)+DXY(I-1,K)
     1          )/DX**2+(DXY(I,K-1)-2.0*DXY(I,K)+DXY(I,K+1))
     2       /DY**2)
          GO TO 30
   31     DT(I,K)=DXY(I,K)+DTIME*A*((DXY(I+1,K)-2.0*DXY(I,K)+DXY(I+1,K)
     1          )/DX**2+(DXY(I,K-1)-2.0*DXY(I,K)+DXY(I,K+1))
     2       /DY**2)
          GO TO 30
   32     DT(I,K)=DXY(I,K)+DTIME*A*((DXY(I-1,K)-2.0*DXY(I,K)+DXY(I-1,K)
     1          )/DX**2+(DXY(I,K-1)-2.0*DXY(I,K)+DXY(I,K+1))
     2       /DY**2)
   30    CONTINUE
   40  CONTINUE
       DO 90 K=2,99
          DO 80 I=1,100
             DXY(I,K)=DT(I,K)
   80    CONTINUE
   90 CONTINUE
      T=T+DTIME
      IF(T.LT.TE) GO TO 100
      WRITE(6,70) T,(DXY(50,K),K=1,91,10)
   70 FORMAT(1X,'TIME = ',F9.2,1X,9F9.2)
      STOP
      END
```

図 16 2次元熱拡散問題のソース・プログラム

Fig. 16 2-dimensional thermal diffusion program.

ものといえる.

4.2 実プログラムによる評価

条件制御ベクトル演算方式の効果を大きなプログラム全体で把握するには，あまりにジョブの特性に依存しすぎるので，1例として**図15**に示す2次元熱拡散問題をとりあげることにする．この問題は，$X=(0:20.0)$, $Y=(0:20.0)$ の矩形領域において，左，右辺は断熱壁，上，下辺は 200℃，内部初期温度は 100℃の前提の元に，時間 $(0:10.0)$ の内部領域の温度分布を求めるものである.

例題としたプログラム例を**図16**に示す．この問題は2章で述べた端点指標条件を含む DO ループの典型的例であり，従来はベクトル化できなかった.

条件制御機能を有する自動ベクトル・コンパイラを用いて HITAC M-200H で実測した結果を**表6**に示す．スカラ処理方式による実行時間，16.08 秒が 8.95 秒に 1.80 倍改善されている．部分的にベクトル化されていた従来の自動ベクトル・コンパイラのオブジェクトでの実行時間 14.67 秒からも 1.64 倍の大きな改善になっている．この問題の場合，ハードウェアの制御ベクトル機能は用いないですんでいるのでこの効果はコンパイラの拡張によるベクトル化の恩恵であり，ベクトル化率の向上が，ベクトルプロセッサにとっていかに重要であるかを示しているといえる.

4.3 性能評価上の問題点

前節までは，条件制御ベクトル演算方式の基本性能と実プログラムの評価結果を述べたが，本節では条件制御ベクトル演算方式固有の性能上の問題点について述べる.

第1の問題点は条件成立比による性能の変化である．すなわち，条件制御ベクトル演算方式は，ベクトル化すれば1ループ当りの性能は条件成立比にほとんど依存しない．ところが，スカラ処理の場合は条件判定の結果演算が不要になれば，確実にその差分だけ1ループ当りの性能は改善することになる．すなわち，ループ・ミクス値は条件の成立比に大きく依存する.

表6 2次元熱拡散問題の性能評価結果
Table 6 Performance analysis of a two-dimensional thermal diffusion problem.

処理種別	Scalar (秒)	従来のベクトル処理 (秒)	条件制御ベクトル演算方式 (秒)
コンパイル	0.29	0.27	0.40
リンク	0.26	0.25	0.26
実行	16.08	14.67	8.95
合計	16.63	15.19	9.61

表7 条件成立比により性能変化
Table 7 Performance variation when ratio of taken conditions changes from 0 to 1.0.

ループミクス種別	演算の実行比率 (%)	(μs/loop) Scalar	(μs/loop) Vector	相対比 (S/V)
単精度	0	0.839	0.326	2.57
	25	0.991	0.334	2.97
	50	1.044	0.342	3.06
	75	1.086	0.346	3.14
	100	1.128	0.344	3.28
倍精度	0	0.854	0.431	1.98
	25	1.160	0.439	2.64
	50	1.234	0.450	2.74
	75	1.286	0.450	2.86
	100	1.348	0.454	2.97

注) ベクトル長：64，連続ベクトルの場合

表7は上に述べた問題を評価するため，条件の成立比，すなわち，演算実行比率を変えてループ・ミクスをスカラ処理，ベクトル処理それぞれの場合について実測したものである．単精度のスカラ処理時には，演算実行比率の 0% から 100% への変化にともない，ミクス値は 0.839 μs から 1.128 μs へ 1.34 倍変化しているが，ベクトル処理では 0.326 μs から 0.344 μs へ 1.06 倍変化するにすぎないことがわかる.

表7の結果から次の点は注目する必要がある．すなわち，演算実行比率0%の場合でも，ベクトル処理はスカラ処理に比べ十分速いことである．単精度ループ・ミクスで 2.57 倍，倍精度ループ・ミクスで 1.98 倍の改善となっている．これはスカラ処理の場合，ループ制御，条件制御などのためにかなり時間を費やしていることによるもので，この部分が減ることがベクトル演算の重要な特徴の一つであるといえる．なお，ベクトル処理の場合でも，条件成立比の差により性能に若干の差が出るのは結果の格納がなくなることにより記憶装置での競合が軽減されるためである.

条件制御ベクトル演算方式に固有の第2の問題は，条件制御を受ける演算の項数により，その効果に差が生じることである．この問題は第1の問題と理由は同じところにあり，その演算項数が多いほどベクトル処理に対し不利になる.

いま，**図17**に示すプログラムを使い，$Z(I)$=以下

```
      DO I=1, N
      IF (A(I).GT.B(I)) GO TO 40
      Z(I)=V(I)+W(I)+X(I)+Y(I)
   40 CONTINUE
```

図17 項数依存性測定プログラム（4項の場合）
Fig. 17 Performance measurement program of the term length effects.

表 8 条件制御を受ける演算式の項数による性能変化
Table 8 Performance dependency on the number of arithmetic terms which are conditionally executed by its corresponding control vector.

項数	処理時間 (μs/loop)		改善比 (S/V)
	Scalar	Vector	
1	0.504	0.165	3.05
2	0.540	0.196	2.75
3	0.579	0.294	1.97
4	0.638	0.393	1.62
5	0.676	0.496	1.36
6	0.719	0.592	1.21
7	0.770	0.689	1.12
8	0.810	0.785	1.03

注) ベクトル長: 64, 連続ベクトル, 演算実行比率: 50% の場合

の項数を変えて実測した結果が表8である. この測定値は演算実行比率を 50% に固定して行ったものであるが, 項数8の場合は, スカラ/ベクトルの改善比 1.03 にまで低下している. このような問題を解決するには, 演算を実行しない場合のハードウェアのきめ細かいベクトル処理方式を開発する必要があり今後の課題である.

5. 結び

IF 文を含む DO ループをベクトル化するために, 条件制御機能を有する内蔵ベクトル演算機構と自動ベクトル・コンパイラからなる条件制御ベクトル演算方式を開発した. HITAC M-200H 上で実験したところ, 50% の演算実行比率を前提としたループ・ミクスで 2.74〜3.06 倍の性能改善効果を確認することができた. また, 2次元熱拡散プログラムの例では, 従来のベクトル演算方式に比し 1.64 倍の性能向上が図られた. これらの結果をもとに, HITAC M-280H で条件制御ベクトル演算方式を実用化した.

ベクトル演算機能を一般化する上で最大の障害は, プログラムによりベクトル化率に差があり, したがってその効果もスカラ処理を高速化した場合ほど普遍性がないことである. この問題を解決する一つの方法は, より高度のアーキテクチャとコンパイラ技術を開発することにより, より多くのプログラムがベクトル演算の高速性の恩恵に浴するようにすることである. この研究もその一翼を担うものであるが, まだ解決すべき多くの課題を抱えている.

謝辞 本研究の推進に当たっては, 日立製作所中央研究所宮本, 安村の各研究員のご協力をいただいた. また, その実用化に際しては, 日立製作所神奈川工場, 小高主任技師のご指導をいただいた. ここに感謝の意を表します.

参考文献

1) 堀越 彌, 梅谷征雄: 汎用計算機のための内蔵ベクトル演算方式, 情報処理学会論文誌, Vol. 24, No. 2, pp. 191-199 (1983).
2) 梅谷征雄, 堀越 彌: 内蔵ベクトル演算機能のための自動ベクトルコンパイラ方式, 情報処理学会論文誌, Vol. 24, No. 2, pp. 238-248 (1983).
3) 梅谷征雄他: 技術計算プログラムの自動ベクトル化技術, 情報処理, Vol. 23, No. 1, p. 29 (1982).
4) 小高俊彦他: 超高速演算の動向, 情報処理, Vol. 21, No. 9, p. 927 (1980).

(昭和57年12月3日受付)
(昭和58年1月17日採録)