The University of Osaka
Institutional Knowledge Archive

Title	統合アレイプロセサについて
Author(s)	泉谷，洋三；北脇，重宗；諸木，赫夫
Citation	大阪大学大型計算機センターニュース．1982，45，p. 59-72
Version Type	VoR
URL	https://hdl.handle.net/11094/65528
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

統合アレイプロセサについて

日本電気株式会社
泉谷洋三[*1]　北脇重宗[*2]　諸木赫夫[*3]

1. はじめに

大阪大学大型計算機センターに設置された超大型コンピュータACOSシステム1000（以下S1000と略す）は、処理性能の向上のため、LSIや実装技術の改良等による、種々のハードウェア方式上の工夫がなされています。これらの改善の中で、ベクトル演算の高速化の手段として、演算処理装置中に、統合アレイプロセサを装備しています。

本稿では、統合アレイプロセサによる、高速化メカニズムについて紹介すると共に、FORTRAN77コンパイラによる利用の方法について記述します。統合アレイプロセサが大阪大学大型計算機センターで利用可能となる時期は、本年の5月13日の予定です。

2. 統合アレイプロセサ機構について

大型技術計算の高速処理のために、ベクトルや行列計算を高速処理する必要が高くなってきています。S1000では、これらの要求に応えて、技術計算に関する経験をベースに、統合アレイプロセサ（IAP：Integrated Array Processor）を開発しました。

IAPは、演算処理装置中に、標準機構として組みこまれ、ベクトル命令を処理する機構です。また、S1000では、1G（＝1000メガ）バイトセグメントが実現されたことにより、IAPは、大配列の演算を連続して処理できます。

アーキテクチャ上、配慮した主な点は、以下の通りです。

(1) 通常の命令を混在して自由に使用できること。
（標準のFORTRAN77の言語仕様のままで、ベクトル命令に展開できるFORTRAN77コンパイラが準備されており、高級言語レベルでの互換性が保たれる。）

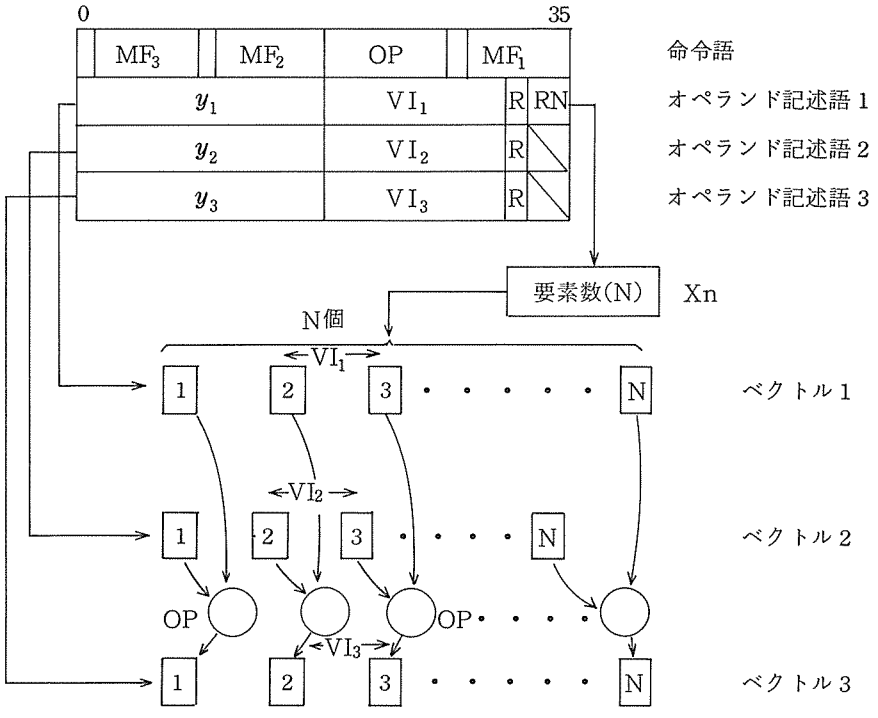
(2) 応用範囲を拡げるため、扱うデータの種類が豊富であること。
（浮動小数点単精度、倍精度、固定小数点データ、論理データが扱える。）

＊1　コンピュータ技術本部第一技術部
＊2　基本ソフトウェア開発本部共通ソフトウェア開発部
＊3　情報処理官庁システム事業部第二システム部

(3) 高性能化できる範囲を拡げるため、命令セットが豊富であること。

（四則演算、論理演算、内積計算、累積、最大値／最小値のサーチなどが行える。）

ベクトル命令の命令形式を図1.に示します。



OP ： 命令コード

MFi： ベクトルiの先頭要素に対するアドレス修飾子

yi ： ベクトルiの先頭要素のアドレス・ディスプレースメント

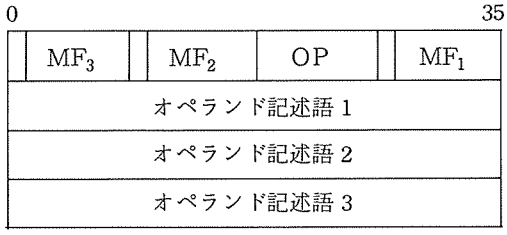
VIi ： ベクトルiの要素間隔（Rの指定により、レジスタの内容を要素間隔とすることもできる。）

RN ： 要素数を保持するレジスタ

R ： 指定レジスタの有無

図1. ベクトル命令の命令形式

この命令の形式は、ACOSアキテクチャの可変長オペランドを扱う複数語命令と同じです。図2.に複数語命令の標準形式を示します。



MFi（i=1～3）　：オペランド記述語 i に対するアドレス修飾の指定
OP　：命令コード
オペランド記述語 i　：命令語とは独立に、データ形式、アドレス、データの長さなどを指定する。

図2.　複数語命令の標準形式

図1.に従って、ベクトル命令の仕様を以下に説明します。

オペランド記述語のyiは、アドレス修飾子MFiで指定されるインデックスレジスタにより修飾されて、各ベクトルiの先頭アドレスを指定します。ベクトル要素の間隔は、ベクトルごとに、VIiで指定します。ベクトルの要素数は、オペランド記述語1のRNが指定するインデックスレジスタXnの内容で指定します。演算の内容は、OPによって指定します。

ベクトル命令の各要素ごとの演算は、通常の命令がパイプラインで処理されるのと同じように、パイプライン化されて処理されます。このパイプライン処理により、基本演算器、浮動小数点加減算器、高速乗除算器などの演算器の使用効率を高めています。

次に、FORTRAN77のDOループを例にとってベクトル命令による改善の様子を図3.に説明します。図3.においてFORTRAN77で記述した累計プログラムX＝X＋A(I)を左欄に、ベクトル命令を使用しない場合の可能な最も少ない基本命令数で実現した場合を中央欄に、ベクトル命令を使用した場合を右欄にそれぞれ示しています。また、基本命令とベクトル命令で実現した場合の処理の流れを示します。

基本命令だけで実現する場合は、浮動小数点加算（FAD：Floating Add)と、オペランド読み出し用アドレス加算、つまりインデックスレジスタの加算（ADLXm：Add Logic to Xm)、そして、ベクトルの要素数の回数だけループさせるための分岐命令（TRCTn：Transfer on Count n)から構成されます。なお、DOループ内では、中間結果は内部レジスタ上へ累積されますので、主記憶へ格納するストア命令（FST：Floating Store)が付加されます。

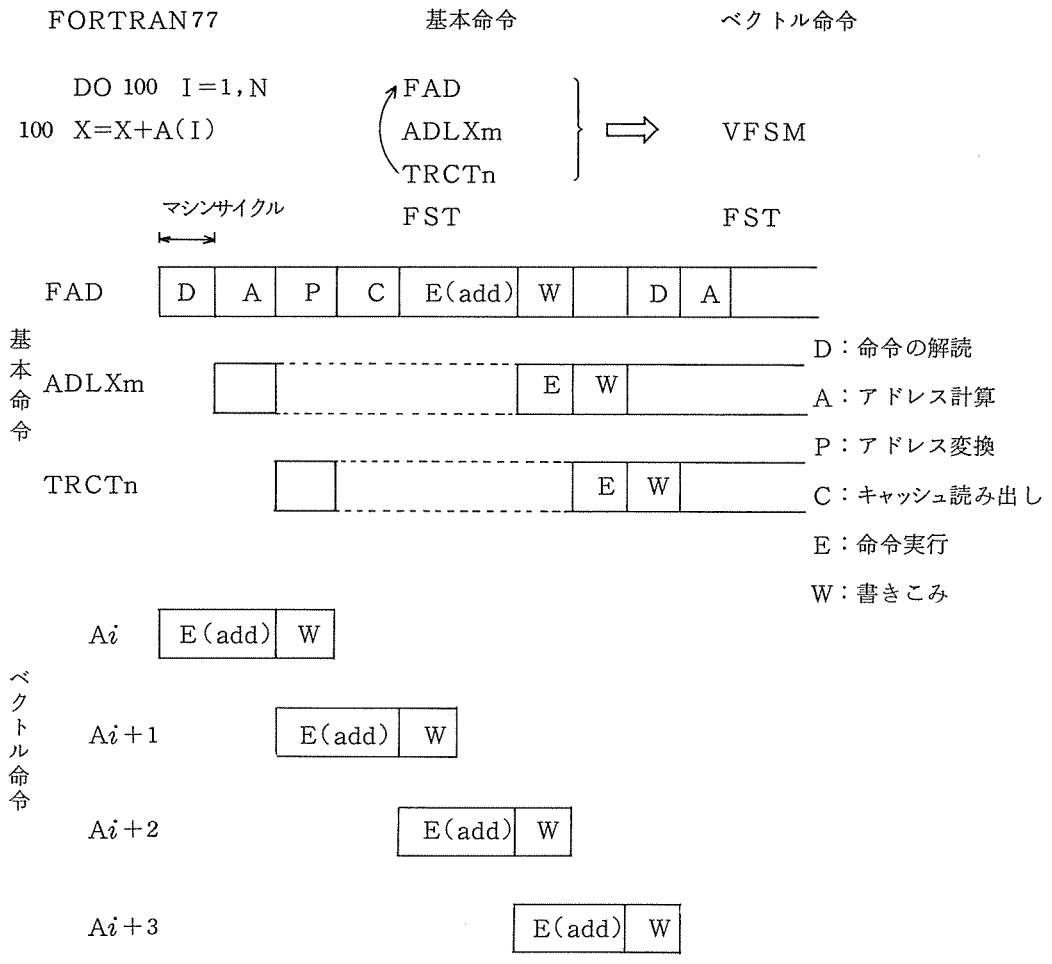


図3. ベクトル命令の動作例

図3.に示すように、TRCTn命令の次のFAD命令の解読とアドレス計算（図3.中のD, Aサイクル）は、前のADLXm命令の終了を待つ必要があります。すなわち、ADLXn命令によって加算されたインデックスレジスタの内容を用いて、FAD命令で使用するオペランドのアドレス計算を行いますが、ADLXm命令のインデックスレジスタへの書き込み（Wサイクル）が終了しないと、その内容を使用できないためです。前の命令の動作待ちとなるこのような現象をパイプラインのインターロックと呼びます。このインターロックにより最高性能が発揮できない場合があります。

一方、ベクトル命令（VFSM）の場合は、先行制御部は、オペランドアドレスによるキャッシュメモリのアクセスと同時に、次のオペランドアドレスの加算を行い、更に、同一サイクルでル

ープ数のカウントを行う機能を持っているので、パイプラインのインターロック現象は起りません。

このように、ベクトル命令を用いることによって、ベクトル命令を用いない場合に比して、飛躍的に処理速度が改善される場合が多くあります。

ベクトル命令で取り扱われるデータは、大配列の場合があるため、演算過程でページ不在割り込みが起ることがあります。このときには、そのページが主記憶へ割り付けられてからの再実行動作、いわゆるデマンドページングが必要になります。また、ひとつのジョブに長時間連続してEPUを割り当てると、入出力処理を終えEPUでの処理待ちのジョブ実行の応答時間が長くなり、システムの運用上不都合を生じます。このような不都合を回避するために、大配列の演算過程で、適当な間隔で外部割り込みを受けつける必要があります。S1000のIAPでは、ベクトル命令の実行途中で割り込みを受けつけることができるようになっており、外部割り込みが、長時間にわたって保留されることはありません。

ページ不在割り込みや、外部割り込みが発生すると演算が中断された時点のアドレス情報や要素数が、プログラムで指定可能なレジスタに保持されるので、再開時には、中断されたベクトル命令から再び実行すれば、引き続く演算が矛盾なく継続して実行できます。

3. ベクトル命令に関する最適化手法

S1000では、高速にベクトルデータを処理するための命令が60種用意されています。このようなベクトル機能をコンパイラ言語を通じて利用する方式として、表1.に示す3方式が考えられます。

方式としては、構文拡張方式が最も自然な考え方ですが、ループ解析方式は言語の変更がないため、ソフトウェア資産として存在するプログラムをベクトル命令を利用して効率的に実行できるという利点も捨て難く、また、現在FORTRAN77に関して国際的に次期標準言語(FORTRAN8X)に配列処理機能を組込む動きがあり、FORTRAN77に独自に仕様を拡張することは、不都合な結果になることも考えられます。

このため、FORTRAN77(V)コンパイラでは、ループ解析方式を基本とし、コンパイラによる解析の困難な最大値／最小値用命令に対しては、組込み関数を用意しています。これら組込み関数は、全てインラインコードに展開します。

FORTRAN77(V)コンパイラでは、ベクトル命令に関するループ解析処理は、コンパイラオプション〝IAP〟が指定された場合に行われます。コンパイラは算術代入文、CONTINUE文のみからなるDOループで使われているデータの型、型変換、関数引用、添字式の形が、ベクトル命令に適合するものを選び、ベクトル命令に展開します。直接ベクトル命令に分解できないDOループでも、2個以上のDOループに分解すれば、それぞれがベクトル化できる場合には、

ベクトル化します。この場合、ベクトルの作業用領域が必要となることがあります。翻訳時に作業用ベクトル領域の大きさ（DOループの実行回数）が不明な場合は、コンパイラオプション〝IAP〟で指定された大きさの作業用ベクトル領域を割り付けます。

また、プログラムのベクトル化のための診断情報として、ベクトル化できなかったDOループに対し、理由を示すメッセージを出力する機能が用意されています。

表1. ベクトル機能サポート方式の比較

方式	コーディング例	長所	短所
ループ解析方式	DIMENSION A(100),B(100),C(100),D(100) DO 10 I=1, 100 10 A(I)=B(I)*C(I)+D(I) S=0 DO 20 I=1, 100 20 S=S+A(I)	• 言語仕様の変更が不要で、既存プログラムにも有効	• コンパイラの解析に依存する • 解析困難な命令もあり、コンパイラが解析できるように、コーディングする必要があり、また逆に適用されないようにする必要が生じることもある
組込み手続き方式	DIMENSION A(100),B(100),C(100),D(100) CALL VAMUL(A,B,C,100,1,1,1) CALL VASUM(A,A,D,100,1,1,1) S=AVSUM(A,100,1)	• 組込み手続きの追加コンパイラの変更量が小さい • 言語仕様の変更がない • 利用者がコーディングした個所のみベクトル化するので、利用者の意図が十分に反映できる。	• 手続きの数機能を選定することが困難 • 既存プログラムの修正が必要
構文拡張方式	DIMENSION A(100),B(100),C(100),D(100) A=B*C+D S=SUM(A)	• 記述性の高い仕様とすることができる	• コンパイラの負担が大きい • 標準仕様から外れる • 既存のプログラムの修正が必要

4. FORTRAN77(V)による利用方法

ACOS－6 FORTRAN77は、センターニュース「FORTRAN77概説」にて紹介されているように、当センター運用においても、今後、利用の中心となって行く言語です。本稿で述べてきたIAPによる、処理性能の高速化についても、FORTRAN77(V)において、最大の効果が期待されます。

S1000のもつ、ベクトル命令を含めた強力な新命令および高度なパイプライン制御等の特徴を十分に発揮するため、FORTRAN77(V)では、最適化手法のひとつとしてベクトル機能に関する機能の強化が図られています。以下に、FORTRAN77(V)でサポートされるベクトル機能について説明します。

FORTRAN77(V)でIAPは、(1)DOループに対するベクトル命令処理 (2)組込み関数によるベクトル命令のサポートがなされます。

(1) DOループに対するベクトル命令サポート

DOループに対して、ベクトル命令使用の要・不要及びベクトル処理化診断メッセージ出力の要・不要は、次のコンパイラオプションにより指定します。

IAP〔(a)〕：コンパイラは最深ループに対し、ベクトル処理命令使用の可・否を分析する。可能ならば、ベクトル命令を利用したコード生成を行う。
本オプションは、最適化オプションOPT＝2，OPT＝3 のときのみ意味を持つ。

(a) ：ループの実行回数が不明のときに仮定されるループの実行回数を指定する。aは1≦a≦256(単位はK回)であり、省略した場合、5(K)が仮定される。コンパイラは、1個の作業用ベクトル領域が不明のとき、これを用いてベクトル領域を確保する。

DIAG ：ベクトル処理化診断メッセージを出力する。
本オプションは、IAPオプションが指定された場合に意味を持つ。

ベクトル命令が、DOループに対して適用される条件は以下の場合です。

① 最深DOループである。

② DOループの文は、算術代入文、CONTINUE文のみから成っている。

③ 整数型、実数型、倍精度実数型の演算である。ただし、整数値のとき、除算を含んではいけない。

④ 実数型または倍精度実数型と整数型との型変換は含まない。

⑤ 関数副プログラムの引用がない。
組込み関数の引用は、次のもののみが許される。

SIN，COS，EXP，SQRT，ALOG，ALOG10，SNGL，DBLE，
DSIN，DCOS，DEXP，DSQRT，DLOG，DLOG10，IABS，
ABS，DABS，IAND，AND，IOR，OR，IEXCLR，XOR

⑥ 互いに、EQUIVALENCEされた変数、配列要素がDOループ内に現われた場合、これらは、引用だけか、または、定義があっても他の定義引用が同じ記憶単位に対するものでない。

⑦ 添字式は、次の形式である。

(a) 整数型の式

(b) 演算子は、＋，－，＊だけである。

(c) 変数名、定数の引用である。このとき変数は、インデックス変数（制御変数または、ループ毎に値が一定値づつ変化する変数）、ループ相対定数（ループ内で値の変化しない変数）のいずれかである。

(d) インデックス変数は、各次元に1個のみである。

⑧ データの定義、引用関係がベクトル命令を適用しても矛盾が生じない関係にある。DOループ内に同じ変数あるいは、配列要素が2度以上出現している場合、ベクトル命令を適用してもデータ引用関係に矛盾が生じないのは、次の条件のときである。

(a) 添字の値が、規則的に変化する配列要素のとき

(i) 定義がなく引用のみである。

(ii) 添字の値域に共通部分がない。

(iii) 前の文に現われる配列の添字の値が、後の文の添字の値より大きいか等しい。

(iv) 代入文の右辺の添字の値は、左辺の添字の値よりも大きいか、または等しい。

(b) 変数または添字の値が、DOループ内で、一定な配列要素のとき

(i) 定義がなく引用のみである。

(ii) 添字の値が異なる。

(iii) 定義よりも前に引用がない。

⑨ ⑧の条件には合わないが、次の形式になっているときは、ベクトル命令を適用する。

(a) S＝S±A（I）＊B（I）のとき

VFIP，VFIPN，VDFIP，VDFIN，VIIP，VIIPN

(b) S＝S±A（I）のとき

VFSM，VFSN，VDFSM，VDFSN，VISM，VISN

(c) C（I）＝B（I）±S＊A（I）のとき

VFMA，VFMS，VDFMA，VDFMS，VIMA，VIMS

(d) A(I+1)=B(I)+C(I)*A(I) のとき

VFIT, VDFIT, VIFIT

⑩ ループ回数が10回以上である。

(2) ベクトル組込み関数でのベクトル命令サポート

ベクトル関数の引数並びの一般形は、次のとおりです。

(X, N, Ix)

X ：配列名でありベクトルを示す。

N ：整数型の変数名、配列要素名、定数または定数名の引用であり、ベクトル要素数(繰り返し数)を示す。

Ix ：整数型の変数名、配列要素名、定数または定数名の引用であり、ベクトル要素間隔(単位は要素数)を示す。

表中、型を示す場合に、次の略号を用いる。

I ：整数型

R ：実数型

D ：倍精度実数型

Q ：4倍精度実数型

表2. ベクトル組込み関数

ベクトル関数	意味	総称名	関数名	引数並びの形式	引数の型	関数の型
最大値	Xiの最大値	VMX	IVMX	(X,N,Ix)	X：I	I
			AVMX	(X,N,Ix)	X：R	R
			DVMX	(X,N,Ix)	X：D	D
			QVMX	(X,N,Ix)	X：Q	Q
最小値	Xiの最小値	VMN	IVMN	(X,N,Ix)	X：I	I
			AVMN	(X,N,Ix)	X：R	R
			DVMN	(X,N,Ix)	X：D	D
			QVMN	(X,N,Ix)	X：Q	Q
絶対値の最大値	|Xi| の最大値	VMXA	IVMXA	(X,N,Ix)	X：I	I
			AVMXA	(X,N,Ix)	X：R	R
			DVMXA	(X,N,Ix)	X：D	D
			QVMXA	(X,N,Ix)	X：Q	Q
絶対値の最小値	|Xi| の最小値	VMNA	IVMNA	(X,N,Ix)	X：I	I
			AVMNA	(X,N,Ix)	X：R	R
			DVMNA	(X,N,Ix)	X：D	D
			QVMNA	(X,N,Ix)	X：Q	Q

ベクトル関数	意　　味	総称名	関 数 名	引数並びの形式	引数の型	関数の型
最大要素の要素番号	X_i の最大値の要素番号 1≦NVMX≦(N−1)*I_x+1	NVMX	NIVMX	(X,N,I_x)	X：I	I
			NAVMX	(X,N,I_x)	X：R	I
			NDVMX	(X,N,I_x)	X：D	I
			NQVMX	(X,N,I_x)	X：Q	I
最小要素の要素番号	X_i の最小値の要素番号 1≦NVMN≦(N−1)*I_x+1	NVMN	NIVMN	(X,N,I_x)	X：I	I
			NAVMN	(X,N,I_x)	X：R	I
			NDVMN	(X,N,I_x)	X：D	I
			NQVMN	(X,N,I_x)	X：Q	I
絶対値最大の要素番号	$\|X_i\|$ の最大値の要素番号 1≦NVMXA≦(N−1)*I_x+1	NVMXA	NIVMXA	(X,N,I_x)	X：I	I
			NAVMXA	(X,N,I_x)	X：R	I
			NDVMXA	(X,N,I_x)	X：D	I
			NQVMXA	(X,N,I_x)	X：Q	I
絶対値最小の要素番号	$\|X_i\|$ の最小値の要素番号 1≦NVMNA≦(N−1)*I_x+1	NVMNA	NIVMNA	(X,N,I_x)	X：I	I
			NAVMNA	(X,N,I_x)	X：R	I
			NDVMNA	(X,N,I_x)	X：D	I
			NQVMNA	(X,N,I_x)	X：Q	I

備考 (1) ベクトル関数名を実引数として使用することはできない。
(2) 4倍精度実数型に関しては、ベクトル命令を使用していない。

5. FORTRAN77(R)による利用法

FORTRAN77(R)では、ベクトル命令に対して、ベクトル関数及びベクトルサブルーチンの呼び出しによってのみ利用できます。すなわち、FORTRAN77(V)のようにコンパイラオプションIAPによって利用することはできません。

各々の一覧を以下に示します。

(1) ベクトル関数

ベクトル関数の引数並びの一般形は、次のとおりです。

(X〔,Y〕,N,I_x〔,I_y〕)

X,Y　：配列名でありベクトルを示す。

N　　：整数型の変数名、配列要素名、定数または定数名の引用であり、ベクトル要素数(繰り返し数)を示す。

I_x,I_y：整数型の変数名、配列要素名、定数または定数名の引用であり、ベクトル要素間隔(単位は要素数)を示す。この引数は、X，Yのベクトルに対応して指定する。

表中、型を示す場合に、次の略号を用いる。

I　：整数型

R　：実数型

D　：倍精度実数型

Q　：4倍精度実数型

表3. ベクトル関数一覧

ベクトル関数	意　味	関数名	引数並びの形式	引数の型	関数の型
内　積	Σ Xi・Yi	IVIPD	(X,Y,N,Ix,Iy)	X,Y:I	I
		AVIPD		X,Y:R	R
		DVIPD		X,Y:D	D
総　和	Σ Xi	IVSUM	(X,N,Ix)	X:I	I
		AVSUM		X:R	R
		DVSUM		X:D	D
最大値	Xi の最大値	IVMX	(X,N,Ix)	X:I	I
		AVMX		X:R	R
		DVMX		X:D	D
		QVMX		X:Q	Q
最小値	Xi の最小値	IVMN	(X,N,Ix)	X:I	I
		AVMN		X:R	R
		DVMN		X:D	D
		QVMN		X:Q	Q
絶対値の最大値	|Xi| の最大値	IVMXA	(X,N,Ix)	X:I	I
		AVMXA		X:R	R
		DVMXA		X:D	D
		QVMXA		X:Q	Q
絶対値の最小値	|Xi| の最小値	IVMNA	(X,N,Ix)	X:I	I
		AVMNA		X:R	R
		DVMNA		X:D	D
		QVMNA		X:Q	Q
最大要素の要素番号	Xi の最大値の要素番号	NIVMX	(X,N,Ix)	X:I	I
	1≦NVMX	NAVMX		X:R	I
	≦(N-1)*Ix+1	NDVMX		X:D	I
		NQVMX		X:Q	I
最小要素の要素番号	Xi の最小値の要素番号	NIVMN	(X,N,Ix)	X:I	I
	1≦NVMN	NAVMN		X:R	I
	≦(N-1)*Ix+1	NDVMN		X:D	I
		NQVMN		X:Q	I

ベクトル関数	意　　味	関　数　名	引数並びの形式	引数の型	関数の型
絶対値最大の要素番号	$\lvert X_i \rvert$ の最大値の要素番号 1≦NVMXA≦(N−1)*Ix+1	NIVMXA NAVMXA NDVMXA NQVMXA	(X,N,Ix)	X：I X：R X：D X：Q	I I I I
絶対値最小の要素番号	$\lvert X_i \rvert$ の最小値の要素番号 1≦NVMNA≦(N−1)*Ix+1	NIVMNA NAVMNA NDVMNA NQVMNA	(X,N,Ix)	X：I X：R X：D X：Q	I I I I

備考 (1) 4倍精度実数型に関しては、ベクトル命令を使用していない。

(2) ベクトルサブルーチン

ベクトルサブルーチンの引数並びの一般形は、次のとおりです。

(Z〔,X〔,Y〔,A〕〕〕,N,Iz〔,Ix〔,Iy〕〕)

Z,X,Y　：配列名でありベクトルを示す。

A　：変数名、配列要素名、定数または定数名

N　：整数型の変数名、配列要素名、定数または定数名の引用であり、ベクトル要素数（繰り返し数）を示す。

Iz,Ix,Iy　：整数型の変数名、配列要素名、定数または定数名の引用であり、ベクトル要素間隔（単位は要素数）を示す。
この引数は、Z，X，Yのベクトルに対応して指定する。

表中、型を示す場合に、次の略号を用いる。

I　：整数型

R　：実数型

D　：倍精度実数型

L　：論理型

表4. ベクトルサブルーチン一覧

ベクトルサブルーチン	意味	サブルーチン名	引数並びの形式	引数の型
ベクトル和	$Z_i \leftarrow X_i + Y_i$	VIADD VAADD VDADD	(Z,X,Y,N,Iz,Ix,Iy)	Z,X,Y：I Z,X,Y：R Z,X,Y：D
ベクトル差	$Z_i \leftarrow X_i - Y_i$	VISUB VASUB VDSUB	(Z,X,Y,N,Iz,Ix,Iy)	Z,X,Y：I Z,X,Y：R Z,X,Y：D
ベクトル積	$Z_i \leftarrow X_i * Y_i$	VIMUL VAMUL VDMUL	(Z,X,Y,N,Iz,Ix,Iy)	Z,X,Y：I Z,X,Y：R Z,X,Y：D
ベクトル商	$Z_i \leftarrow X_i / Y_i$	VADIV VDDIV	(Z,X,Y,N,Iz,Ix,Iy)	Z,X,Y：R Z,X,Y：D
ベクトル積和	$Z_i \leftarrow X_i + Y_i * A$	VIMA VAMA VDMA	(Z,X,Y,A,N,Iz,Ix,Iy)	Z,X,Y,A：I Z,X,Y,A：R Z,X,Y,A：D
ベクトル積差	$Z_i \leftarrow X_i - Y_i * A$	VIMS VAMS VDMS	(Z,X,Y,A,N,Iz,Ix,Iy)	Z,X,Y,A：I Z,X,Y,A：R Z,X,Y,A：D
一次漸化式	$Z_{i+1} \leftarrow Z_i + X_i * Y_i$	VIITR VAITR VDITR	(Z,X,Y,N,Iz,Ix,Iy)	Z,X,Y：I Z,X,Y：R Z,X,Y：D
移送	$Z_i \leftarrow X_i$	VIMV VAMV VDMV VADMV VDAMV	(Z,X,N,Iz,Ix)	Z,X：I Z,X：R Z,X：D Z,R,Y：D Z,D,Y：R
符号反転移送	$Z_i \leftarrow -X_i$	VIMVC VAMVC VDMVC	(Z,X,N,Iz,Ix)	Z,X：I Z,X：R Z,X：D
正値移送	$Z_i \leftarrow \lvert X_i \rvert$	VIMVA VAMVA VDMVA	(Z,X,N,Iz,Ix)	Z,X：I Z,X：R Z,X：D
負値移送	$Z_i \leftarrow -\lvert X_i \rvert$	VIMVNA VAMVNA VDMVNA	(Z,X,N,Iz,Ix)	Z,X：I Z,X：R Z,X：D
論理積	$Z_i \leftarrow X_i$ AND Y_i	VAND	(Z,X,Y,N,Iz,Ix,Iy)	Z,X,Y：IorL
論理和	$Z_i \leftarrow X_i$ OR Y_i	VOR	(Z,X,Y,N,Iz,Ix,Iy)	Z,X,Y：IorL
排他的論理和	$Z_i \leftarrow X_i$ XOR Y_i	VXOR	(Z,X,Y,N,Iz,Ix,Iy)	Z,X,Y：IorL

6. FORTRAN66による利用法

FORTRAN66(R)では、FORTRAN77(R)と同様に、ベクトル関数およびベクトルサブルーチンの呼び出しによって利用することができますが、FORTRAN66(V)では、統合アレイプロセサは、サポートされておりませんので利用することはできません。

7. おわりに

本稿の執筆に当って、大阪大学大型計算機センターニュースへの掲載の機会を与えて頂くと共に、本稿の内容について詳細にわたる懇切な助言を賜った大阪大学大型計算機センター大中幸三郎助手に対し、厚くお礼申し上げます。

8. 参考文献

1) 大中幸三郎，後藤米子：FORTRAN77概説(1), (2), (3)大阪大学大型計算機センターニュース，Vol.11, No.3 (1981)，Vol.11, No.4 (1982)，Vol.12, No.1 (1982).

2) 梅村要一他：ACOSシステム1000の高速化技術，NEC技報，Vol.35, No.5(印刷中)