ベクトルプロセッサとＦＯＲＴＲＡＮベクトル拡張言語

平林俊弘　　高嶋秀夫
（富士通株式会社）

1．はじめに

最近の商用スーパーコンピュータは，ベクトル演算パイプラインを備えたベクトルプロセッサが主流となっている．その利用言語としては，標準ＦＯＲＴＲＡＮが多い．11),12),13) 標準ＦＯＲＴＲＡＮで配列データを扱うには通常ＤＯループによる繰り返しを用いるが，これらをベクトルプロセッサ向けに変換する自動ベクトル化技術の発展がその一因となっている．しかし科学技術計算分野での配列計算の需要は多く，配列演算などをそのままプログラミングできることも望まれている．ベクトルプロセッサの普及にともない，標準ＦＯＲＴＲＡＮにおける配列処理の必要性も強まっているといえる．1),6),7)

本稿では，ＦＡＣＯＭ　ＶＰシリーズの有効利用とＦＯＲＴＲＡＮの配列処理強化をねらって開発した，ＦＯＲＴＲＡＮ７７／ＶＰベクトル拡張言語のベクトル処理を紹介する．また，ＡＮＳＩのＸ３Ｊ３委員会で検討されているＦＯＲＴＲＡＮ８Ｘ（次の標準規格案）に提案されている配列処理についても簡単に紹介する．

なお本稿では，単一であることを示すためにスカラ，またスカラの集りを示すためにベクトルという用語を用いる．例えばスカラデータ，スカラ命令，ベクトルデータ，ベクトル命令など．またベクトル処理と配列処理はスカラデータをまとめて扱う点で同じだが，ベクトル処理は一次元データまでであり，多次元データの扱いを含めた配列処理とは区別して用いることにする．

2．ＦＡＣＯＭ　ＶＰのベクトル処理　8),9),10)

汎用コンピュータでは，一つの命令で一つのデータを処理する．これをスカラ処理という．これに対してベクトルプロセッサは，複数のスカラデータを一つのベクトル命令でまとめて処理することができる．これをベクトル処理という．

a）ベクトル命令

ベクトル命令は，スカラ命令の繰り返しを一度に処理するものと考えることができる．ベクトル命令には，ロード／ストア命令，四則演算，比較，論理演算などスカラ命令に対応する基本的なものをはじめ，ベクトルデータ特有の扱いを実現するマクロ命令，編集命令など８３命令ある．表２．１に代表的なベクトル命令を示す．

表２．１代表的なベクトル命令

分類	命令数	簡略コード	名称	
ロード／ストア命令	14	ＶＬ	VECTOR LOAD	
		ＶＬＤ	VECTOR LOAD LONG	
		ＶＬＥ	VECTOR LOAD SHORT	
		ＶＬＭ	VECTOR LOAD MASK	など
四則演算命令	18	ＶＡＤ	VECTOR ADD LONG	
		ＶＳＤ	VECTOR SUBTRACT LONG	
		ＶＭＤ	VECTOR MULTIPLY LONG	
		ＶＤＤ	VECTOR DIVIDE LONG	など
比較命令	5	ＶＣＤ	VECTOR COMPARE LONG	など
論理演算命令	12	ＶＮ	VECTOR AND	
		ＶＸＭ	VECTOR EXCLUSIVE OR MASK	など
マクロ命令	13	ＶＦＸＤ	VECTOR FIND MAXIMUM LONG	
		ＶＳＭＤ	VECTOR SUM LONG	など
編集命令	7	ＶＣＰ	VECTOR COMPRESS	
		ＶＥＸ	VECTOR EXPAND	など
変換命令	5	ＶＣＩＤ	VECTOR CONVERT INTEGER TO LONG	
		ＶＣＤＩ	VECTOR CONVERT LONG TO INTEGER	など
制御命令	9	ＶＬＶＬ	LOAD VECTOR LENGTH REGISTER	
		ＶＰＴ	POST	
		ＶＷＴ	WAIT	など
合計	83			

b）ベクトルデータ

メモリ上のベクトルデータとベクトルレジスタとの転送は，ベクトルロード／ストア命令によって行う．ベクトルロード／ストア命令では，連続，等間隔及びランダムな位置をとるスカラデータ群をアクセスできる．

c）ベクトル命令の実行

ベクトル命令は複数のスカラデータをまとめて処理する．ベクトル命令が扱うスカラデータの個数をベクトル長と呼び，これはベクトル制御命令によって設定される．このベクトル長に従って，ベクトルロード／ストア命令はスカラデータを転送する．また，ベクトル演算命令はベクトルレジスタ内のスカラデータを演算処理する．

d）マスク付きベクトル演算

ベクトル命令の処理で，要素となるスカラ処理をマスク値で制御することができる．これをマスク付きベクトル演算という．マスクデータはマスクレジスタにおかれ，オンとなる要素に対応するスカラ処理のみが実行される．

e）配列演算のベクトル命令による実行

標準ＦＯＲＴＲＡＮにおける配列演算は，配列要素に対する繰り返し演算である．配列要素はスカラデータであり，その集りとしての配列演算は，そのほとんどをベクトル命令によって処理することが可能である．14),15),16),17) 例えば，次のＤＯループは，

```
      ＤＩＭＥＮＳＩＯＮ　Ａ（１０），Ｂ（１０），Ｃ（１０）
      ＲＥＡＬ＊８　Ａ，Ｂ，Ｃ
      ＤＯ　１０　Ｉ＝１，１０
１０　　Ａ（Ｉ）＝Ｂ（Ｉ）＋Ｃ（Ｉ）
```

以下のベクトル命令列で実行できる．

```
      ＶＬＶＬ　　，１０
      ＶＬＤ　　ＶＲ００１，Ｂ
      ＶＬＤ　　ＶＲ００２，Ｃ
      ＶＡＤ　　ＶＲ００３，ＶＲ００１，ＶＲ００２
      ＶＳＴＤ　ＶＲ００３，Ａ
```

ベクトル制御命令（ＶＬＶＬ）で，ベクトル長１０を設定する．ベクトルロード命令（ＶＬＤ）でベクトルＢとベクトルＣをベクトルレジスタ（ＶＲ００１とＶＲ００２）にロードし，ベクトル加算命令（ＶＡＤ）で結果を求める．加算はベクトルレジスタ内の１０個の要素対応に行われる．結果のベクトルレジスタ（ＶＲ００３）をベクトルＡにストアして，このＤＯループの実行は完了する．

３．ＦＯＲＴＲＡＮ７７／ＶＰベクトル拡張言語

ＦＡＣＯＭ　ＶＰハードウェアを有効に利用するアプローチには現状では以下のものがあげられる．

－　自動ベクトル化機能による標準ＦＯＲＴＲＡＮでの利用
－　ＶＰ向け数値計算ライブラリによる利用
－　ベクトル処理の記述によるハードウェアの直接利用

このうち，自動ベクトル化機能，数値計算ライブラリによる利用は，既存のＦＯＲＴＲＡＮ資産を修正することなく処理効率を向上させる基本的アプローチで既に実現している．さらに，有効利用の拡大をねらったアプローチとして，標準ＦＯＲＴＲＡＮにベクトルデータの扱いを直接表現するベクトル処理を追加した．

３．１　ベクトル処理拡張の概要

ベクトルデータを扱う基本機能として，ベクトルデータの記法，ベクトル演算，条件付きベクトル演算，ベクトル組込み関数を追加した．表現が標準ＦＯＲＴＲＡＮの自然な拡張となるようＦＯＲＴＲＡＮ８Ｘ配列処理を意識した．また，ＦＡＣＯＭ　ＶＰの利用を意図して，ベクトル処理で書いたところは必ずベクトル命令で処理するよう対応ずけた．

組込み演算子にはないが，プログラムで利用可能とおもわれる標準的なベクトル命令の機能は，ベクトル操作関数として組込み関数の形式で実現した．レジスタの使用などハードウェア依存するところは，標準的記述の実現の観点からむしろコンパイラの処理に任せるのが良いと考え導入しなかった．

ベクトル処理拡張の概要を図３．１に示す．

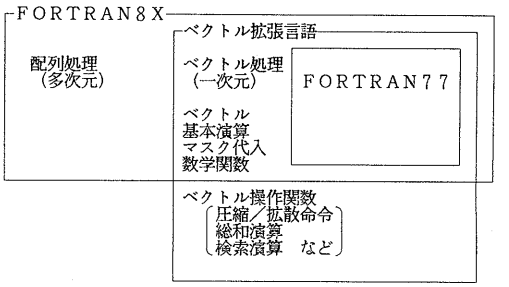


図３．１　ベクトル拡張言語の概要

３．２　言語仕様　18)

a）スカラデータとベクトルデータ

スカラデータは，変数，配列要素，定数など単一のデータのことである．ベクトルデータは，スカラデータの集りである．ベクトルデータの要素の個数をベクトル要素数という．要素となるスカラデータには順番があり，１からベクトル要素数までの番号が振られる．これをベクトル要素番号という．ベクトルデータは以下のように示すことができる．ここでａｉはスカラデータを示す．

```
{ a1, a2, ・・・, ai, ・・・, an }
  ↑   ↑          ↑          ↑    ベクトル要素数n
  1   2          i          n……ベクトル要素番号
```

b）ベクトルデータの書き方

ベクトル拡張言語では，配列全体及び配列の部分をベクトルデータとして扱うことができる．ベクトルデータの要素となる配列要素は，特定の次元の添字集合を示すことで選択される．配列要素を引用するときのように，親となる配列名にベクトルデータを示す添字を付加することで配列要素が選択される．要素の選択は，原則として一つの次元に限られる．

array (・・・, index, ・・・)

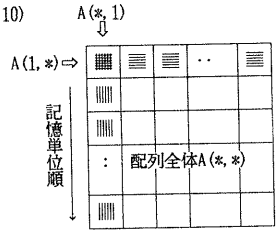
ここでarrayは配列名，indexは添字集合を示す選択子である．indexの書き方によって，連続要素，等間隔要素，ランダム要素の選択ができる．

i）連続要素の選択

連続要素の選択には，アスタリスクを用いる．この指定にかぎって，連続する次元に複数指定しても良い．

例：連続要素から構成されるベクトル

DIMENSION A(10, 10)　　A(*, 1)



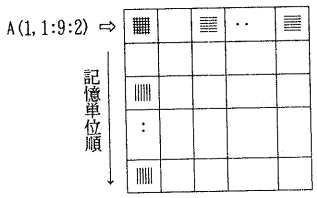
ii）等間隔要素の選択

等間隔要素を選択する場合，次の形式の添字範囲を指定する．

初期値：終値：増分値

例：等間隔要素から構成されるベクトル

DIMENSION A(9, 9)　　A(1:9:2, 1)



iii）ランダムな要素の選択

ランダムな要素を選択する場合，一次元整数型ベクトルを指定する．指定された一次元整数型ベクトルの値が，添字値を意味している．

例：ランダムな要素から構成されるベクトル

```
DIMENSION A(10)
DIMENSION IDX(5)
INTEGER IDX
DATA IDX/1, 5, 6, 7, 9/
```

A ⇨ ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩

A(IDX(*)) ⇨ ① ⑤ ⑥ ⑦ ⑨

b）ベクトル演算及びベクトル式

ベクトル演算は，オペランドとなるベクトルの要素対応のスカラ演算を意味している．要素対応のスカラ演算は，ベクトル要素番号の順に従って実行される．オペランドのベクトル要素数は一致しなければならない．二項演算においていずれか一方がスカラオペランドの場合，それは対応するベクトルオペランドのベクトル要素数のベクトルとして扱われる．

スカラ演算子はそのままベクトル演算子として拡張される．オペランドがベクトルデータであるとき，ベクトル演算子として作用する．ベクトル演算子として拡張されるのは以下のものである．

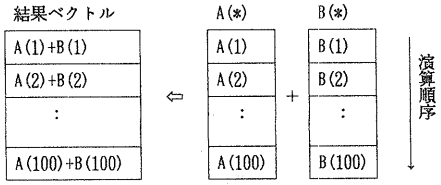
- 算術演算子　：　＋　－　＊　／　＊＊
- 関係演算子　：　.LT.　.LE.　.EQ.　.NE.　.GT.　.GE.
- 論理演算子　：　.NOT.　.AND.　.OR.　.EQV.　.NEQV.

ベクトル演算子の追加によってベクトル式も追加される．ベクトル式の結果は，ベクトルとなる．

例：ベクトル演算

DIMENSION A(100), B(100)

A(*)+B(*)



c）ベクトル代入文

代入文の左辺にベクトルを記述するとベクトル代入文となる．ベクトル演算と同様，右辺の式の結果をベクトル要素番号の順に従って要素対応に代入する．

例：ベクトル代入

```
DIMENSION A(100)
A(*)=1.0
```

ベクトル代入文において右辺に出現するベクトルの要素をその引用に先立って左辺で定義することはできない．これをベクトルの回帰的参照という．配列要素に対する回帰的な代入にはＤＯループを用いる．

例：ベクトルの回帰的参照

```
DIMENSION A(100)
A(2:100)=A(1:99)
```

d）条件付きベクトル演算

ベクトル演算，ベクトル代入文における要素対応のスカラ演算を論理型ベクトルの真偽の値でマスク制御することができる．この論理型ベクトルはＷＨＥＲＥ条件とよばれ，論理ＷＨＥＲＥ文で設定される．論理ＷＨＥＲＥ文は論理ＩＦ文とおなじ形をしており，設定されたＷＨＥＲＥ条件でトレーラ部のベクトル代入文をマスク制御する．

例：論理ＷＨＥＲＥ文

```
DIMENSION A(100),B(100),C(100),L(100)
LOGICAL L

WHERE(L(*))A(*)=B(*)+C(*)
```

次のＤＯループと同じ結果となる．

```
DO 10 I=1,100
10 IF(L(I))A(I)=B(I)+C(I)
```

ＷＨＥＲＥ条件によって複数のベクトル代入文をマスク制御することもできる．これにはＷＨＥＲＥブロックを使う．

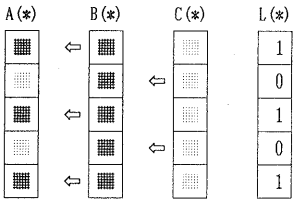
例：ＷＨＥＲＥブロック

```
DIMENSION A(5),B(5),C(5),D(5)

WHERE(L(*))
   A(*)=B(*)
ELSEWHERE
   A(*)=C(*)
ENDWHERE
```

ＥＬＳＥＷＨＥＲＥ文はＷＨＥＲＥ条件の反転を意味しており，これは次の表現と同じ結果となる．

```
WHERE(L(*))A(*)=B(*)
WHERE(.NOT.L(*))A(*)=C(*)
```



e）組込み関数

標準ＦＯＲＴＲＡＮの組込み関数は，ほとんどベクトルの実引数が指定できるように拡張されている．ベクトルの実引数が指定された場合，実引数ベクトルの要素となるスカラデータに対してのスカラ組込み関数演算が行われる．表３．２に拡張された標準ＦＯＲＴＲＡＮ組込み関数の総称名を示す．

例：組込み関数の引用

```
DIMENSION A(10),B(10)

A(*)=SIN(B(*))
```

次のＤＯループと同じ結果となる．

```
DO 10 I=1,10
10 A(I)=SIN(B(I))
```

表３．２　拡張された標準ＦＯＲＴＲＡＮ組込み関数

機能	総称名	機能	総称名
型変換	ＩＮＴ	正弦	ＳＩＮ
	ＲＥＡＬ	余弦	ＣＯＳ
	ＤＢＬＥ	正接	ＴＡＮ
	ＣＭＰＬＸ	余接	ＣＯＴＡＮ
	ＤＣＭＰＬＸ	逆正弦	ＡＳＩＮ
四捨五入	ＮＩＮＴ	逆余弦	ＡＣＯＳ
絶対値	ＡＢＳ	逆正接	ＡＴＡＮ
剰余	ＭＯＤ		ＡＴＡＮ２
倍精度化乗算	ＤＰＲＯＤ	双曲線正弦	ＳＩＮＨ
符号の付替え	ＳＩＧＮ	双曲線余弦	ＣＯＳＨ
超過分	ＤＩＭ	双曲線正接	ＴＡＮＨ
最大値	ＭＡＸ	誤差関数	ＥＲＦ
最小値	ＭＩＮ		ＥＲＦＣ
虚部	ＩＭＡＧ	ガンマ関数	ＧＡＭＭＡ
共役複素数	ＣＯＮＪＧ		ＬＧＡＭＭＡ
平方根	ＳＱＲＴ	論理否定	ＮＯＴ
立方根	ＣＢＲＴ	論理積	ＩＡＮＤ
指数	ＥＸＰ	論理和	ＩＯＲ
	ＥＸＰ２	排他的論理和	ＩＥＯＲ
	ＥＸＰ１０		
対数	ＬＯＧ		
	ＬＯＧ１０		
	ＬＯＧ２		

f）ベクトル操作関数

標準の組込み関数の他に，ベクトル処理固有の組込み関数としてＦＡＣＯＭ　ＶＰハードウェア命令の機能を活かしたベクトル操作関数がある．各操作関数の総称名とともに，その機能をＤＯループ表現で示す．

ｉ）ＡＰＲＯＧ　：　等差数列ベクトルの生成

```
DIMENSION STR(100)
INTEGER STR

STR(*)=APROG(1,100,1)      ⇨   DO 10 I=1,100
                                10   STR(I)=I
```

ｉｉ）ＶＳＵＭ　：　ベクトルの要素和

```
INTEGER SUM,DATA(100)
LOGICAL MASK(100)

SUM=VSUM(DATA(*),MASK(*)) ⇨   SUM=0
                              DO 10 I=1,100
                                 IF(MASK(I))THEN
                                    SUM=SUM+DATA(I)
                                 ENDIF
                              10 CONTINUE
```

iii) ＩＦＭＸ，ＦＭＸＶ　：　最大要素位置，最大要素値

```
INTEGER MAXIDX,MAXVAL
INTEGER DATA(100)

MAXIDX=IFMX(DATA(*))        ⇨  MAXIDX=1
MAXVAL=FMXV(DATA(*))           MAXVAL=DATA(1)
                               DO 10 I=2,100
                               IF(DATA(I).GT.MAXVAL)THEN
                                  MAXIDX=I
                                  MAXVAL=DATA(I)
                               ENDIF
                            10 CONTINUE
```

iv) ＩＦＭＮ，ＦＭＮＶ　：　最小要素位置，最小要素値

iii）を参照.

v) ＩＯＮＣ　：　真要素の個数

```
INTEGER COUNT
LOGICAL LGDATA(100)

COUNT=IONC(LGDATA(*))       ⇨  COUNT=0
                               DO 10 I=1,100
                                  IF(LGDATA(I))THEN
                                     COUNT=COUNT+1
                                  ENDIF
                            10 CONTINUE
```

vi) ＩＦＢＯＮ　：　先頭真要素の位置

```
INTEGER TOPIDX
LOGICAL LGDATA(100)

TOPIDX=IFBON(LGDATA(*))     ⇨  TOPIDX=0
                               DO 10 I=1,100
                                  IF(LGDATA(I))THEN
                                     TOPIDX=I
                                     GOTO 20
                                  ENDIF
                            10 CONTINUE
                            20 CONTINUE
```

vii) ＧＡＴ　：　ベクトルの圧縮

```
INTEGER COMPRS(100)
INTEGER DATA(100)
LOGICAL PTRN(100)

J=IONC(PTRN(*))             ⇨  J=0
COMPRS(1:J)=                   DO 10 I=1,100
-  GAT(PTRN(*),DATA(*)            IF(PTRN(I))THEN
                                     J=J+1
                                     COMPRS(J)=DATA(I)
                                  ENDIF
                            10 CONTINUE
```

viii) ＳＣＡＴ　：　ベクトルの拡散

```
INTEGER EXPAND(100)
INTEGER DATA(100),BASE(100)
LOGICAL PTRN(100)

EXPAND(*)=                  ⇨  J=0
-  SCAT(PTRN(*),BASE(*)           DO 10 I=1,100
-              ,DATA(*))             IF(PTRN(I))THEN
                                        J=J+1
                                        EXPAND(I)=DATA(J)
                                     ELSE
                                        EXPAND(I)=BASE(I)
                                     ENDIF
                            10 CONTINUE
```

ix) ＩＤＸＬ　：　マスクからリストへの変換

```
INTEGER IXLIST(100)
LOGICAL PTRN(100)

J=IONC(PTRN(*))
IXLIST(1:J)=IDXL(PTRN(*)) ⇨  J=0
                              DO 10 I=1,100
                                 IF(PTRN(I))THEN
                                    J=J+1
                                    IXLIST(J)=I
                                 ENDIF
                           10 CONTINUE
```

x) ＢＩＴＳ　：　リストからマスクへの変換

```
INTEGER IXLIST(100)
LOGICAL PTRN(100)

PTRN(*)=BITS(IXLIST(*),100)
                              ⇨  DO 10 I=1,100
                              10    PTRN(I)=.FALSE.
                                    DO 20 I=1,100
                              20    PTRN(IXLIST(I))=.TRUE.
```

xi) ＭＡＳＫ　：　マスクによるベクトルの混合

```
INTEGER MG(100),ONDATA(100),OFDATA(100)
LOGICAL PTRN(100)

MG(*)=MASK(PTRN(*),ONDATA(*),OFDATA(*))

                              ⇨  DO 10 I=1,100
                                    IF(PTRN(I))THEN
                                       MG(I)=ONDATA(I)
                                    ELSE
                                       MG(I)=OFDATA(I)
                                    ENDIF
                           10 CONTINUE
```

3．3　プログラミング

基本的なベクトル計算などは，ベクトル処理の導入によって紙面上の論理に近い形でコーディングできるようになる．ＤＯループの数，文番号が減ることによって，プログラムもより簡潔になる．

ベクトルデータに対する条件付き処理は，リストベクトル（ランダムな要素の選択によるベクトル），ＷＨＥＲＥ文及び圧縮・拡散のベクトル操作関数などが選択でき，ハードウェアの特性を考慮していろいろな表現をとることができる．

さらにＦＡＣＯＭ　ＶＰのベクトル命令機能を表現したベクトル操作関数によってプログラムをより簡潔に表現できる．

例：Ｎ個のデータ処理

Ｎ個のデータ	DIMENSION DATA(N)
平均	AVRG=VSUM(DATA(*))/N
標準偏差	S=SQRT(VSUM((DATA(*)-AVRG)**2)/N)
平均以上のデータ数	U=IONC(DATA(*).GT.AVRG)

例：ベクトル操作関数の組合せ

```
DIMENSION A(100),B(100),C(100),IDX(100)

DO 10 I=1,100
   IF(B(I).GT.C(I))THEN
      T=B(I)
   ELSE
      T=C(I)
   ENDIF
   S=S+T*A(I)
10 CONTINUE
```

⇩

```
S=VSUM(MASK(B(*).GT.C(*),B(*),C(*))*A(*))
```

3．4　処理系

a）翻訳

ベクトル拡張言語の翻訳は，これまでの自動ベクトル化コンパイラと同じ使い方ができるように，オプション機能として働くようにした．ベクトル拡張言語の翻訳と同時に，既存の自動ベクトル化機能，ＶＰ向けの最適化機能も有効である．

b）デバッグ機能

ベクトル処理のためにデバッグ機能を用意した．翻訳時にオプション指定すると，以下の検査を行うオブジェクトを生成する．

- ベクトル定義の値表示
- ベクトルの添字オーバフロー検査
- 未定義ベクトルの引用検査
- ベクトル演算における長さ検査

c）目的プログラム生成

ベクトル拡張言語の翻訳では，ＶＰ向けの目的プログラムだけを生成する．汎用機上でのデバッグなどプログラムのポータビリティを保つために，標準ＦＯＲＴＲＡＮへのソースプログラム変換機能をサポートした．

d）標準ＦＯＲＴＲＡＮソースへの変換機能

ＦＯＲＴＲＡＮソースプログラムの中でベクトル拡張言語で書かれた部分は，変換機能によって標準ＦＯＲＴＲＡＮの記述に変換することができる．ベクトル演算，マスク付きベクトル演算など基本的なものから，複雑なベクトル操作関数まで変換できる．ベクトルの回帰的参照を検出していないので１００％変換とはいかないが，プリプロセッサとして使用しても実用上問題がない機能を実現している．

4．ＦＯＲＴＲＡＮ８Ｘ配列処理　2),3),4),5)

現在ＡＮＳＩ　Ｘ３Ｊ３委員会では，次期ＦＯＲＴＲＡＮ規格を検討中である．検討されている提案機能のなかでも配列処理は科学技術計算向けの重要なテーマになっている．ＡＮＳＩ　Ｘ３Ｊ３からのドキュメントをみる限りにおいて配列処理に関する部分は相当かたまっており今後の検討において大幅な変更はないと思われる．

ここでは，ベクトル拡張言語のベクトル処理をベースに比較しながら，将来形としてのＦＯＲＴＲＡＮ８Ｘ配列処理を概観する．用語は英語のままとする．またＦＯＲＴＲＡＮ８Ｘは８Ｘと略す．

a）配列処理

ベクトル拡張言語では，配列の全体，一次元を基本とした配列の部分をまとめて扱うことができる．８Ｘでは，配列の全体，任意の部分をまとめて扱うことができる．一次元だけでなく多次元の部分を一度に処理できる．ベクトル拡張言語ではベクトルデータは，ベクトル代入文など限られたところに記述可能だが，８Ｘでは，言語の基本要素としてほぼどこに現れても良い．

例：配列処理

```
DIMENSION A(100,100,100),B(100,100,100),C(100,100,100)

DO 10 I=1,100
   DO 10 J=1,100
      DO 10 K=1,100
10 A(K,J,I)=B(K,J,I)+C(K,J,I)

                ⇩

A = B + C
```

b）ａｒｒａｙ

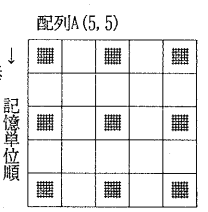
ａｒｒａｙは，ｒａｎｋ，ｓｉｚｅ，ｓｈａｐｅの性質をもつ．ｒａｎｋは次元数，ｓｉｚｅは全要素の個数，ｓｈａｐｅは，次元数と各次元のｓｉｚｅである．同じ形（ｓｈａｐｅ）の配列は，ｃｏｎｆｏｒｍａｂｌｅであり，配列演算などでオペランドとなる配列は同じ形であること（ｃｏｎｆｏｒｍａｂｉｌｉｔｙ）が必要となる．

c）ａｒｒａｙ　ｓｅｃｔｉｏｎ

配列の部分はａｒｒａｙ　ｓｅｃｔｉｏｎと呼ばれる．ａｒｒａｙ　ｓｅｃｔｉｏｎの要素は，添字の集合で選択される．次元ごとの添字集合は，ｓｅｃｔｉｏｎ　ｓｅｌｅｃｔｏｒで示される．ｓｅｃｔｉｏｎ　ｓｅｌｅｃｔｏｒには，添字の範囲を指定するものと一次元整数型配列を指定するものがある．いずれもベクトル拡張言語と同じ形式である．８Ｘの場合，複数の次元に出現して良い．

例：ａｒｒａｙ　ｓｅｃｔｉｏｎ

```
DIMENSION A(5,5)
A(1:5:2,1:5:2)  ⇨  ▦が要素
```



d）ａｒｒａｙ　ｃｏｎｓｔｒｕｃｔｏｒ

ａｒｒａｙ　ｃｏｎｓｔｒｕｃｔｏｒは一次元の配列値を生成する．配列の定数ともいえる．ベクトル拡張言語では，ＦＯＲＴＲＡＮの文字集合に'［'がないことなどから実現しなかった．ただし組込み関数ＡＰＲＯＧで等差数列の値をとる一次元ベクトルが生成できる．

例：ａｒｒａｙ　ｃｏｎｓｔｒｕｃｔｏｒ

```
DIMENSION IDX(100)
IDX=[20[1:10:2]]
```

配列ＩＤＸの各要素に１，３，５，７，９を２０回繰り返してできる数列を代入する．

e）配列演算及び配列式

配列全体，配列の部分を構成する要素をまとめて演算することができる．組込み演算子はすべて配列演算子として拡張される．配列のオペランドをもつとき配列演算子として作用する．演算は要素対応に行われる．８Ｘの場合，要素の演算順序は規定していないので概念的には全要素同時に処理される．オペランドとなる配列のｓｈａｐｅは同じであり，演算結果の配列もオペランドと同じｓｈａｐｅとなる．

f）配列代入（ａｒｒａｙ　ａｓｓｉｇｎｍｅｎｔ）

配列代入には，単純代入の他にｍａｓｋｅｄ　ａｒｒａｙ　ａｓｓｉｇｎｍｅｎｔ，ｅｌｅｍｅｎｔ　ａｒｒａｙ　ａｓｓｉｇｎｍｅｎｔがある．

単純代入は対応要素ごとのスカラ代入であり，配列演算の場合と同様に左辺と右辺の配列のｓｈａｐｅは同じである．演算順序が規定されていないので同一配列内の重なりがある代入の場合には，右辺の結果配列は一時的に退避する必要がある．ベクトル拡張言語では演算順序を規定しており，同一配列内の重なりのあるベクトル代入文ではベクトルの回帰的参照となることがある．

例：配列の入換え

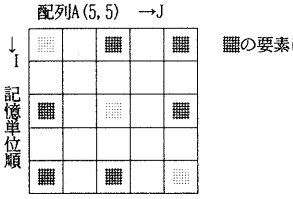
```
DIMENSION A(10000)
A(1:10000)=A(10000:1:-1)
```

ｍａｓｋｅｄ　ａｒｒａｙ　ａｓｓｉｇｎｍｅｎｔは，ＷＨＥＲＥ条件によるマスク制御であり基本的にはベクトル拡張言語と同じである．

ｅｌｅｍｅｎｔ　ａｒｒａｙ　ａｓｓｉｇｎｍｅｎｔは，ＦＯＲＡＬＬ文を用いるものであり，ｉｎｄｅｘ集合をもとめてから配列要素の代入を行う．

例：ｅｌｅｍｅｎｔ　ａｒｒａｙ　ａｓｓｉｇｎｍｅｎｔ

```
DIMENSION A(5,5),B(5,5)
FORALL(I=1:5:2,J=1:5:2,I.NE.J)A(I,J)=1.0
```



g）配列組込み関数

８Ｘでは，組込み関数が強化されて１００種類にも達している．配列組込み関数（ａｒｒａｙ　ｉｎｔｒｉｎｓｉｃ　ｆｕｎｃｔｉｏｎ）としても，ベクトル及び行列の乗算，ｒａｎｋを縮小する数値または論理計算，配列の性質の問合せ，配列の構成，配列の操作などを実現する多種多用なものが用意されている．例えばＳＵＭでは，配列のある次元要素の総和を求めることができ，結果となる配列のｒａｎｋは１減少する．このように配列を形のまま扱うことができ一般的でかつ強力な機能をもっている．

h）動的配列と別名配列

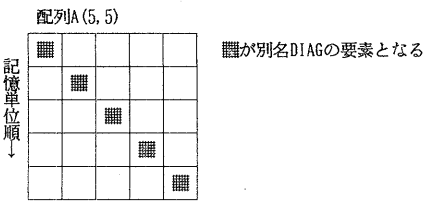
８Ｘでは，新しくａｌｌｏｃａｔａｂｌｅ　ａｒｒａｙとｉｄｅｎｔｉｆｉｅｄ　ａｒｒａｙという配列が追加されている．

ａｌｌｏｃａｔａｂｌｅ　ａｒｒａｙは，ＡＬＬＯＣＡＴＥ文によって動的に記憶域割付けし，ＦＲＥＥ文によって割付けを解除する配列のことである．

ｉｄｅｎｔｉｆｉｅｄ　ａｒｒａｙは，ＩＤＥＮＴＩＦＹ文によって親となる（ｐａｒｅｎｔ）配列の全体及び部分に動的に別名付けする配列のことである．

例：ｉｄｅｎｔｉｆｉｅｄ　ａｒｒａｙ

```
DIMENSION A(5,5)
IDENTIFY(DIAG(I)=A(I,I),I=1,5)
```



５．まとめ

本稿では，ＦＡＣＯＭ　ＶＰシリーズの有効利用とＦＯＲＴＲＡＮ配列処理の強化をねらったＦＯＲＴＲＡＮ７７／ＶＰベクトル拡張言語で実現したベクトル処理を紹介した．

ベクトル拡張言語のベクトル処理には，ＦＯＲＴＲＡＮ８Ｘ配列処理のサブセットを組み込んでおり，要素の選択，組込み演算，マスク処理など基本機能を実現できた．またＦＡＣＯＭ　ＶＰシリーズの機能を有効利用する標準的なベクトル処理を実現できた．

ＦＯＲＴＲＡＮ８Ｘの配列処理は，ベクトル拡張言語で実現したベクトル処理にくらべて一般的でより強力なものだが，それは記述性重視の方向であり性能面を考えると問題がないとはいえない．現状の汎用コンピュータでの実現には明らかに問題があり，ベクトルプロセッサの普及によってはじめて効率的に実現する環境が整いつつあるといえる．

ベクトルプロセッサでのＦＯＲＴＲＡＮ８Ｘ配列処理の実現は，アーキテクチャの改良，配列処理向けの最適化の強化など課題が多い．今後ともＦＯＲＴＲＡＮ発展のために努力する所存である．

〔参考文献〕

〔1〕R.W.Hockney and C.R.Jesshope,"Parallel Computers",Adam Hilger Ltd.,Bristol,1981
邦訳：奥川峻史，黒住祥介，並列計算機，共立出版，1984

〔2〕ANSI working document,"Fortran8X,X3J3/S8 Version95",June 1985

〔3〕"PROPOSALS APPROVED FOR FORTRAN 8X X3J3/S6.81",ANSI,May 1982

〔4〕"Fortran Information Bulletin",ANSI,X3J3,November 1983

〔5〕菅忠義：ＦＯＲＴＲＡＮ８Ｘの紹介，ｂｉｔ12月号,1985

〔6〕棚倉，神谷：”ベクトルプロセサのソフトウェア技術動向”，航空宇宙技術研究所特別資料，航空機計算空気力学シンポジウム論文集，p25-36,1984

〔7〕RONALD H.PERROTT,DANNY CROOKES,PETER MILLIGANS,andW.R.MARTIN PURDY,"A Compiler for Array and Vector Processing Language",IEEE Trans.Software Eng.,vol.SE-11,NO.5,MAY 1985

〔8〕岡本，田村，内田：”スーパーコンピュータＦＡＣＯＭ　ＶＰのハードウェア”，雑誌FUJITSU,VOL.35 NO.4

〔9〕磯辺，神谷，黒羽：”スーパーコンピュータＦＡＣＯＭ　ＶＰのソフトウェア”，雑誌FUJITSU,VOL.35 NO.4

〔10〕平栗ほか：”マシンサイクル7.5ｎｓを達成した並列パイプライン処理方式のスーパーコンピュータＦＡＣＯＭ　ＶＰ”，日経エレクトロニクス，1983.4.11，1983

〔11〕安村通晃：”スーパーコンピュータとそのコンパイラ”，ｂｉｔ 7月号,1985

〔12〕島崎眞昭：”スーパーコンピュータ入門”，コンピュータソフトウェア，VOL.2 NO.3 JULY 1985

〔13〕唐木幸比古：”スーパーコンピュータとは”，Ｃｏｍｐｕｔｅｒ　Ｔｏｄａｙ，1984/7　NO.2

〔14〕神谷ほか：”ＦＯＲＴＲＡＮプログラムの特徴とベクトルプロセサ・アーキテクチャ”，情報処理学会第２５回（昭和57年後期）全国大会，６Ｆ－１

〔15〕高嶋ほか：”ベクトルプロセサの並列ベクトル演算方式”情報処理学会第２５回（昭和57年後期）全国大会，６Ｆ－３

〔16〕平林ほか：”ベクトルプロセサの柔軟性をもたせたベクトルレジスタ方式”，情報処理学会第２５回（昭和57年後期）全国大会，６Ｆ－４

〔17〕滝内ほか：”ベクトルプロセサの効率的条件文ベクトル化方式”，情報処理学会第２５回（昭和57年後期）全国大会，６Ｆ－６

〔18〕”FACOM OSIV/F4 MSP FORTRAN77/VPベクトル拡張言語手引書”，ＦＡＣＯＭマニュアル，1985