計算機アーキテクチャ 52－6
（1984. 3. 7）

「日立製作所の計算機開発の歴史と展望」
――科学技術計算の高速化アーキテクチャについて――

河辺 峻　小高 俊彦
（日立製作所 神奈川工場）

1. はじめに

コンピュータを構成する基幹ハードウェア技術の中で，半導体技術の急激な進歩は，素子のコストを飛躍的に低減し，従来では実現不可能であったような大規模なコンピュータが実現しつつある。これに伴ないアーキテクチャ技術の方も安価な半導体を大量に利用して実現する新しい考え方が導入されて来ている。

この報告では，日立製作所の大型コンピュータの中で，特に科学技術計算を高速に処理するアーキテクチャ技術について述べる。

「コンピュータ・アーキテクチャ」という用語の意味は，「プログラムから見えるコンピュータの機能であり，コンピュータ内部の実現方式とは独立している」[1]という狭義の概念で考えることにする。

2. 汎用コンピュータのアーキテクチャと問題点

現在の汎用コンピュータのアーキテクチャの基本的な考え方は，2つのオペランドの演算を指示する命令を次々に実行して処理を進めて行くことである。図1.に汎用コンピュータの処理の概念図を示す。

この図において動作を指示する命令は記憶装置にある。命令の実行は，処理装置による命令語の読み出しとこれに続く命令解読によってその実行が開始される。通常の命令は2つのオペランドを指定し，この図の例では一方のデータは記憶装置，他方は処理装置内の高速の内部レジスタから読み出される。2つのオペランドは演算器で処理され内部レジスタへ戻される。これが

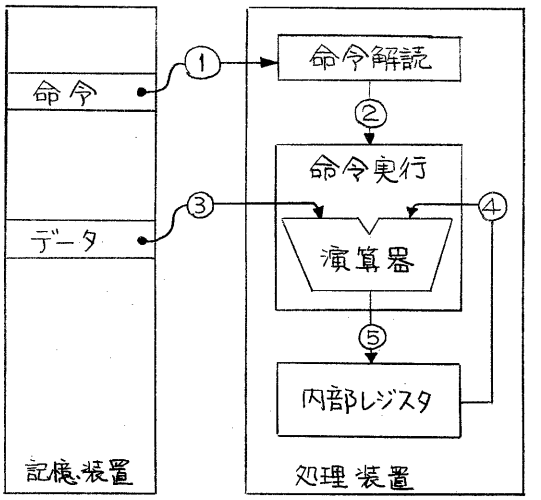

図1. 汎用コンピュータの処理の概念図

代表的な命令の処理形態であり，2つのオペランドが記憶装置あるいは内部レジスタにあるかにより，命令の処理タイプが

- レジスタ ― レジスタ
- レジスタ ― 記憶（図1の例）
- 記憶 ― 記憶

に分類される。

次に科学技術計算における汎用コンピュータの処理とその問題点について考える。科学技術計算のプログラムは主としてFORTRAN言語で記述されているが，処理時間の多くの部分はDOループの処理である。

たとえば簡単のために，

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

を例にとって考える。汎用コンピュータの命令では，このDOループはFORTRANコンパイラによって次の様

な命令語に展開される。

```
        ┌→LD   0, C(1)
        │ AD   0, B(1)
N回     │ STD  0, A(1)
        │ AR     (オペランドアドレス更新)
        └BCT     (ループ回数制御)
```

したがってこのDOループを処理するためには，記憶装置より

2×N　のオペランド読み出し
1×N　のオペランド書き込み
5×N　の命令語の読み出し

を必要とする。

今，演算器の処理能力がひじょうに高いと仮定すると，このDOループの処理時間は，記憶装置と処理装置との間のオペランドと命令語の転送能力によって決まる。すなわち

2N＋1N＋5N＝8N

のデータ（オペランド＋命令語）転送時間である。

この例において まず着目すべき点は，オペランドの転送量よりも命令語の転送量の方が多いという点である。DOループの処理という決まった手順の繰り返しに，5N回もの命令語の読み出しを行なうのはいかにももったいない。

したがってコンピュータ内部の実現方式（これを方式論理技術と呼ぶ）では，命令語が記憶装置に連続して格納されていること，および同じ命令語をたびたび頻度高く転送することなどの性質を利用して，

- 一度に複数個の命令語を転送する。
- 処理装置内に高速のキャッシュ・メモリを置き，たびたび用いられる記憶装置の写しを持っておく。

などの処理を行なうことにより，記憶装置と処理装置との間の命令語の転送を実効的に減らしている。

これはアーキテクチャが要求する一見無駄な処理を，方式論理技術がカバーしている一例であるといえる。しかしながら処理装置の内部では5N回の命令の解読と実行が行なわれており，オペランドのアドレスの計算も各命令語の解読の後に毎回行なわれる。

これらの反省からベクトル命令の概念が生まれる。ベクトル命令では，決まった手順の繰り返しで処理するオペランドの転送と演算を一つの命令で済ますことができる。先程のDOループの例では，VEAD（Vector Element-wise Add Double）命令という一つのベクトル命令を処理すれば良く，処理装置は3N回のオペランドの転送と，1N回の演算（加算）に専心することができる。

これが次節に述べる内蔵ベクトル演算方式のアーキテクチャの一例である。

3. 内蔵ベクトル演算方式

内蔵ベクトル演算方式とは，科学技術計算能力を高めるために，日立製作所がHITAC M-180，200H，280Hなどの汎用大型コンピュータ上にIAP（Integrated Array Processor）機能として実用化したものである。[2)]

図2に内蔵ベクトル演算方式におけるベクトル命令処理の概念を示す。ベクトル命令は図3に示す様な仕様になっており，オ2オペランドとオ3オペランドのデータ群の間で，エレメント対応に演算を行ない，結果をオ1オペランドのデータ群に格納する。

このようにデータ群を一まとめに定義することにより，処理装置は一括して演算を行ない高速性能を得ている。図5に示すようなベクトル命令を約40種備えることによって10MFLOPS以上の実効性能を汎用コンピュータで初めて実現した。また図4に示す自動ベクトルコンパイラの働きによって，利用者は既存の標準FORTRANプログラムを用いることができる。

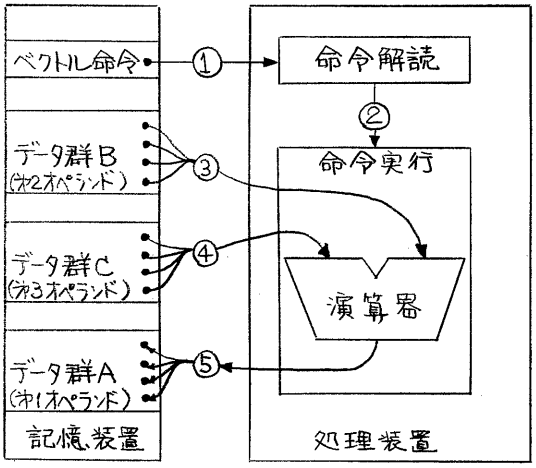

図2. 内蔵ベクトル演算方式におけるベクトル命令処理の概念図

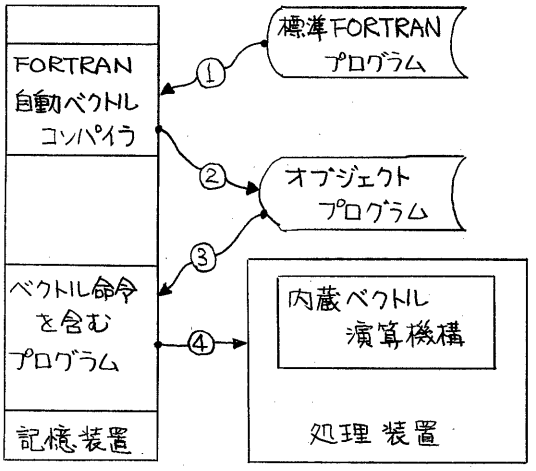

図4. 内蔵ベクトル演算方式の使い方の一例

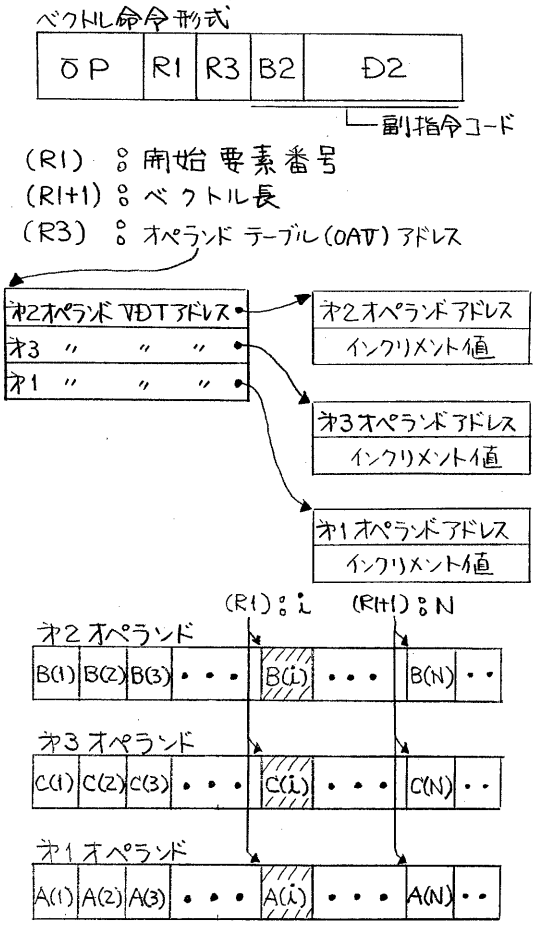

図3. ベクトル命令仕様の例

	命令名称	略称	動作概要
1	Move	VME/D	$Z_i \leftarrow X_i$
2	Complement	VECE/D	$Z_i \leftarrow -X_i$
3	Add	VEAE/D	$Z_i \leftarrow X_i + Y_i$
4	Subtract	VESE/D	$Z_i \leftarrow X_i - Y_i$
5	Multiply	VEME/D	$Z_i \leftarrow X_i * Y_i$
6	Divide	VEDE/D	$Z_i \leftarrow X_i / Y_i$
7	Sum	VSME/D	$FPR \leftarrow FPR + \Sigma X_i$
8	Sum with Complement	VSMCE/D	$FPR \leftarrow FPR - \Sigma X_i$
9	Inner Product	VIPE/D	$FPR \leftarrow FPR + \Sigma X_i * Y_i$
10	Inner Product with Complement	VIPCE/D	$FPR \leftarrow FPR - \Sigma X_i * Y_i$
11	Scalar Multiply and Add	VSMAE/D	$Z_i \leftarrow Z_i + X * Y_i$
12	Scalar Multiply and Subtract	VSMSE/D	$Z_i \leftarrow Z_i - X * Y_i$
13	First Order Iteration	VITRE/D	$Z_{i+1} \leftarrow X_i + Y_i * Z_i$
14	Convert D→S	VCVDE	Z_i(Single) $\leftarrow X_i$(Double)
15	Convert S→D	VCVED	Z_i(Double) $\leftarrow X_i$(Single)
16	Compare Equal	CEQE/D	$C_i \leftarrow X_i = Y_i$
17	Compare Not Equal	CNEE/D	$C_i \leftarrow X_i \neq Y_i$
18	Compare Greater or Equal	CGEE/D	$C_i \leftarrow X_i \geq Y_i$
19	Compare Greater Than	CGTE/D	$C_i \leftarrow X_i > Y_i$
20	Compare Less or Equal	CLEE/D	$C_i \leftarrow X_i \leqq Y_i$
21	Compare Less Than	CLTE/D	$C_i \leftarrow X_i < Y_i$

図5. ベクトル命令レパートリ

この方式の一つの特徴は，図2に示すように，すべてのベクトル命令のオペランドが記憶装置上に存在し，内部レジスタを用いていない点である。これにより，ＴＳＳ環境においてベクトル命令を処理中に，処理を中断して割込を受付けることが可能である。処理したエレメント数は，図3に示すベクトル命令のR1フィールドで指定された汎用レジスタに格納され，割込処理後，再びこのベクトル命令がのこりのエレメントの処理に戻る。またこのベクトル命令のオペランドが記憶装置上に存在しないとき（仮想記憶においてマッピング・フォールトが発生したとき）も，処理したエレメント数のところで，割込が発生しオペランドが記憶装置上にページ・インした後，処理が再開される。

この様に，汎用コンピュータ並みのきめ細かな割込制御を可能にすることにより，既存のオペレーティング・システムとの共存を実現し，ベクトル命令方式の用途を拡大することができた。

内蔵ベクトル演算方式は，現在のところ演算内容の規則性が高い科学技術計算分野に適用されているが，汎用コンピュータとの共存方式を考えると，他の分野においてもこれらの考え方が利用できよう。

この方式の一つの問題点は，記憶装置上のオペランドの参照回数が増大することがある点である。たとえば，次のFORTRANのDOループの例を考えてみる。

```
      DO  20  I = 1,N
20  A(I) = B(I) + C(I) + D(I)    (例12)
```

汎用コンピュータの命令語では，この式は次の様に展開される。

```
N回 ┌ LD   0, D(I)
    │ AD   0, C(I)
    │ AD   0, B(I)
    │ STD  0, A(I)
    │ AR      (オペランドアドレス更新)
    └ BCT     (ループ回数 制御)
```

したがってこのDOループを処理するためには，記憶装置より

３×Ｎ　のオペランド読み出し
１×Ｎ　のオペランド書き込み
６×Ｎ　の命令語の読み出し

を必要とする。

一方，ベクトル命令方式では次の2命令に展開される。

```
VEAD   W ← C + D
VEAD   A ← B + W
```

すなわち，ベクトル命令語の読み出しは２回で済むが，オペランドの記憶装置に対する参照は，本来の４Ｎ回に対して６Ｎ回に増大している。図6に演算の項数と汎用コンピュータとの性能比との関係の概略図を示す。これより，演算の項数が増えるとオペランド

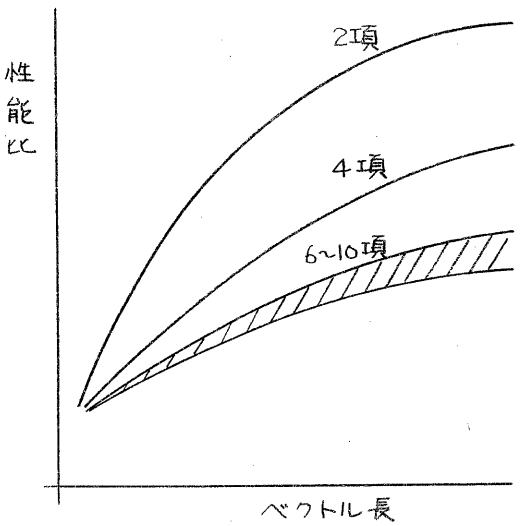

図6. 演算項数と性能の概略図
（汎用コンピュータとの性能比）

の記憶装置に対する参照が増大するため性能の向上率が低下する傾向になる。

しかしながら，全体では大きな性能向上が得られている。その理由は命令解読の回数がO(N)からO(1)に減少したこと，およびベクトル・データの構造と規則性をあらかじめ処理装置に伝えることによって高度のパイプライン演算処理が可能になったことにある。

一方，アーキテクチャ技術の点から考察すると，図3に示すような3オペランド形式のベクトル命令の仕様を，演算の項数に対応した，可変数個のオペランド形式のベクトル命令の仕様に拡張することも可能である。この場合は余分のオペランド参照回数は発生しない。(図7)

しかしながら，コンピュータ内部の実現方式技術である方式論理技術の点から考えると，性能を確保するためには，オペランド数に応じた制御回路がそれぞれ独立に必要となるため，可変数個のオペランド形式のベクトル命令を実現するのはひじょうにむつかしい。この点で，方式論理技術を考慮に入れたアーキテクチャ技術の進歩がきわめて重要である。

4. スーパーコンピュータの高速アーキテクチャ技術

内蔵ベクトル演算方式は，汎用コンピュータの環境の中で共存を図った高速処理のアーキテクチャである。これに対して性能を一層優先させたアーキテクチャが，スーパーコンピュータ用のアーキテクチャ技術である。ここでは日立製作所が最近開発したHITAC S-810に代表されるスーパーコンピュータのアーキテクチャ技術の特徴について述べる。

内蔵ベクトル演算方式では，ベクトル命令の導入により1演算当りの命令数が減り，命令の読み出しおよび解読

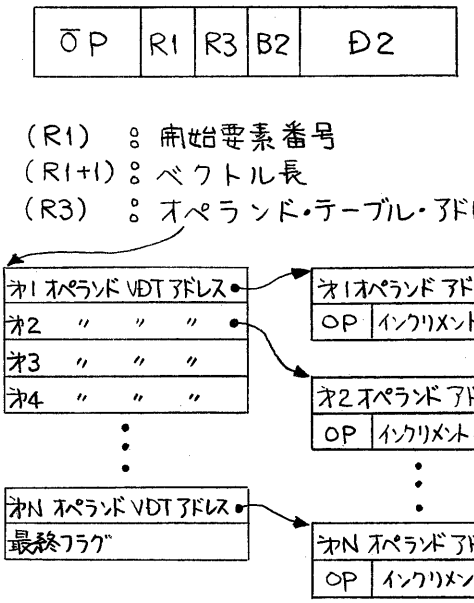

図7. 拡張ベクトル命令の仕様例

に要する処理装置，記憶装置の負荷が軽くなる。さらに現在のスーパーコンピュータでは，図8に示すように多量(数十KB)のベクトル・レジスタを処理装置内部に設けて記憶装置とのデータ転送量を大幅に減らしている。図8の例は図2の例と同じく，

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

(⊗は任意の演算を表わす)

を例にしているので，両者に差異はない。むしろ図8の方がベクトル・レジスタをデータが至由する分だけ複雑になっている。

ところが，例2で示すDOループの場合は，オペランドの記憶装置に対する参照は，内蔵ベクトル演算方式では6N回であったが，スーパーコンピュータのアーキテクチャでは本来の4N回に減少する。これは演算の中間結果をベクトル・レジスタには格納するが，記憶装置には格納しないためである。

現在の代表的なスーパーコンピュー

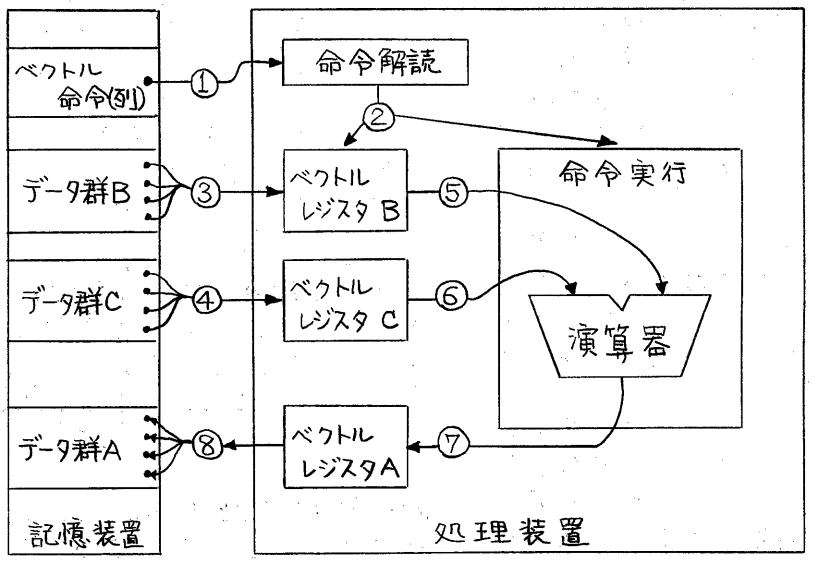

図8. ベクトル・レジスタ方式におけるベクトル命令処理の概念図

タはすべてこの考え方を土台にして設計されている。このようにすれば，記憶装置とのデータの転送量は必要最低限で済むことになり，性能上のあい路になり易い主記憶装置とのデータ転送は，アーキテクチャ技術により解決される。

スーパーコンピュータの処理装置の設計は，記憶装置とのデータ転送能力が最大限に発揮できるように行なわれる。このため処理装置内部の実現方式である方式論理技術では；

- 並列演算パイプライン方式
- 複数命令の並列演算実行方式

などを用いて，複数のベクトル命令の演算の実行が複数個のパイプライン化された演算器を用いて行なわれる。

この結果，コンピュータ内部の処理は必然的にデータフロー風に実行され，性能は記憶装置とのデータ転送能力に大きく依存することになる。

図9にS-810のベクトル命令の形式と命令の種類数を示す。

OP	R1	R2	R3

ベクトル命令形式

命令種類の分類	S-810 種類数
ロード/ストア	18
転送	7
四則演算	11
マクロ演算	10
フォーマット変換	2
比較	12
検索	7
論理演算	7
マスク演算	7
制御	2
命令種類数合計	83 種

図9. ベクトル命令の形式と命令の種類数（S-810）

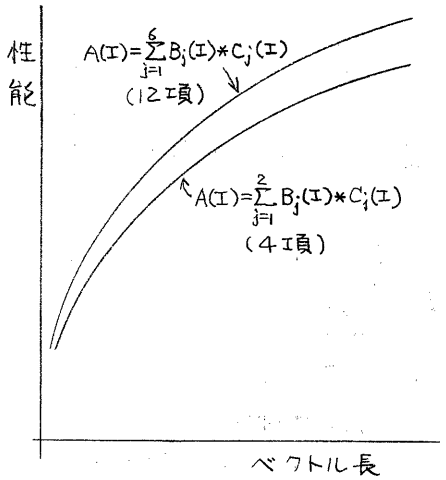

図10　S-810における演算項数と性能の概略図

内蔵ベクトル演算方式では，演算の項数が増えるとオペランドの記憶装置に対する参照が増大するため，性能の向上率が低下する傾向（図6）であった。S-810のアーキテクチャではこの問題は解決され，図10に示す様にむしろ演算の項数が多い方が演算数が多いために高い性能を示す傾向になる。

FORTRANで書かれたプログラムは，図11に示すように自動ベクトルコンパイラによってベクトル命令を含むオブジェクト・プログラムに展開される。

S-810の処理装置は，汎用コンピュータのアーキテクチャの命令（これをスカラ命令と呼ぶことにする）を処理するスカラ・プロセッサと，ベクトル命令を処理するベクトル・プロセッサを並置した形でもっている。オブジェクト・プログラムのうち，ベクトル命令はベクトルプロセッサで実行されるが，スカラ命令はスカラプロセッサで実行される。この様子を図12に示す。スカラプロセッサとベクトルプロセッサはたがいに独立に動作することが可能である。このため例えばベクトルプロセッサでベク

FORTRANのDOループ

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

自動ベクトルコンパイラ

オブジェクト	(ベクトル命令)
VLD	VR0 , B
VLD	VR4 , C
VEMD	VR8,VR0,VR4
VSTD	VR8 , A

VLD : Vector Load
VEMD : Vector Elementwise Multiply
VSTD : Vector Store
VR_i : i番ベクトルレジスタ

図11　ベクトル命令への展開例

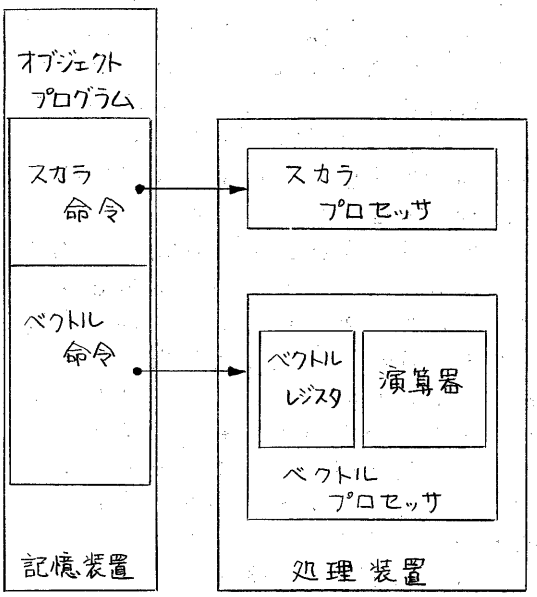

図12　S-810におけるプログラム処理の方式

トル命令を処理中に，外部より入出力装置からの終了割込みなどが発生した場合，ベクトル命令の処理を途中で中断することなく，スカラプロセッサが割込み処理を進めることができる。

したがって数十KBにおよぶベクトルレジスタの内容を退避／回復する必要もなく，割込み処理にかかるオーバヘッドを大幅に減少させることができる。

5. 性能を決定する要因

科学技術計算を高速に処理するアーキテクチャ技術について述べてきたが，実際のプログラムを実行した場合に，性能を決定する要因について考える。

(1) ベクトル化率

ユーザが作成したFORTRANプログラムは，自動ベクトルコンパイラにより図12に示すようなスカラ命令とベクトル命令から構成されるオブジェクト・プログラムに展開される。このプログラムの実行において，スカラ命令はスカラプロセッサで実行されるため，その性能は汎用大型コンピュータで処理した場合と大差がない。これに対してベクトル命令で実行された性能は大幅に向上する。このため図13に示すようにスカラ命令のみで実行した時間のうち，ベクトル命令で処理できる部分の比率をベクトル化率と呼ぶが，この部分を増やすことがきわめて重要である。

とくにスーパーコンピュータでは，ベクトル処理部分の性能向上度が10倍を越えてくるので，ベクトル化率を高めることが必須の要件である。

このために原理的にベクトル化が可能なプログラム部分をできる限りベクトル化するためのアーキテクチャ技術の開発が必要である。表1にM-280H IAPとS-810において，プログラムの計算構成要素とベクトル処理の可能性について比較する。

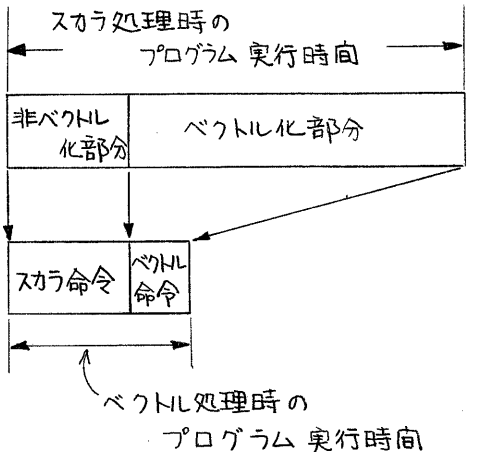

図13 ベクトル処理によるプログラムの実行時間

表1 計算構成要素とベクトル処理可能性

計算構成要素	FORTRAN例	M-280H IAP	S-810
四則演算	A(I)=B(I)⊗C(I)	○	○
内積・総和	S=S+A(I)*B(I)	○	○
条件付ループ	IF(A(I).EQ.B(I)) C(I)=D(I)	○	○
整数・論理演算	N(I)=L(I)+K(I)	×	○
間接指標ベクトル（リスト・ベクトル）	K=N(I) A(K)=B(K)+C(K)	×	○
集約・分散	A(I)=B(L(I))	×	○
関数	X(I)=EXP(Y(I))	△	○
一次巡回演算	A(I)=A(I-1)+B(I)	○	○
参照関係不適	A(I)=A(I-4)+B(I)	×	×
DOループ以外		×	×

(2) 性能の向上度

ベクトル化部分がスカラ処理時に比べ何倍高速化したかを示す値である。この値は，記憶装置とのデータの転送能力によって決まる値が，最大値を示す様に方式論理を開発するのが普通である。しかし実際には，ベクトル長やプログラムのデータ依存性などの影響により，最大値より低下する。

6. 結言

以上，科学技術計算を例にとって汎用コンピュータの抱えている幾つかの問題点と，それを解決するためのアーキテクチャ技術について述べた。

今後ともコンピュータを構成するハードウェア技術の分野では，半導体技術の進歩により，ますます素子のコストが低減して行くと考えられる。このため汎用コンピュータの性能向上は，複数個のプロセッサにより処理する方向に進むと思われる。

これに対し本報告で示したベクトル処理の考え方は，あるプログラムの処理速度を飛躍的に向上させる方式である。このアーキテクチャは，比較的データの構成と演算の規則性が高い科学技術計算分野にまず適用され実用化されたが，今後は画像処理，データベース処理といった分野に適用が拡大されることが期待できよう。

参考文献

1) G.M. Amdahl, G.A. Blaauw and F.P. Brooks, Jr, "Architecture of the IBM System/360" IBM Journal of Research and Development Vol.8 No.2 (April 1964)

2) 小高俊彦，他：HITAC M-180内蔵アレイプロセッサ，日立評論，Vol.60，No6，pp53-58 (1978)

3) 河辺峻，他：HITAC M-200H内蔵アレイプロセッサ(IAP)，電子通信学会電算機研究会，EC 80-79 pp 67-75 (1980)

4) 堀越彊，他：汎用計算機のための内蔵ベクトル演算方式，情報処理論文誌，Vol.24，No2，pp 191-199 (1983)

5) 小高俊彦，他：最大性能が630 MFLOPSで1Gバイトの半導体拡張記憶が付くスーパーコンピュータ HITAC S-810，日経エレクトロニクス，314，159-184 (1983)

6) R.M. Russell "The CRAY-1 Computer System" Communications of the ACM Vol 21, no.1, pp 63-72 (1978)

7) 堀越彊，他：コンピュータアーキテクチャ技術の動向とスーパーコンピュータ，日立評論，Vol 65，No8，pp 529-534 (1983)