OUKA
The University of Osaka
Institutional Knowledge Archive

Title	スーパーコンピュータSX-1の概要(1) : アーキテクチャを中心に
Author(s)	渡辺, 貞; 近藤, 良三; 端山, 信幸 他
Citation	大阪大学大型計算機センターニュース. 1986, 60, p. 109-129
Version Type	VoR
URL	https://hdl.handle.net/11094/65682
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

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

The University of Osaka

スーパーコンピュータＳＸ－１の概要（１）

― アーキテクチャを中心に ―

渡辺　貞[+]、近藤良三[++]、端山信幸[+++]、大中幸三郎[++++]、藤井　護[++++]

1. はじめに

スーパーコンピュータＳＸシステム〔１－３〕は、大規模な科学技術計算を高速に実行するためのコンピュータシステムです。特に本システムは、これらの大規模計算で多用されるベクトル計算を超高速に実行するだけでなく、ベクトル計算ができない部分の処理、例えば、スカラ計算や入出力処理、システムの制御などにおいても高速化を図り、システム全体としての総合的な処理能力の向上が図られています。本稿では、ＳＸシステムの構成、システムを構成する主要装置の概要、特にベクトル計算の高速実行のための多重並列パイプライン、スカラ計算の高速化のための工夫、ソフトウェアの概要などについて説明します。

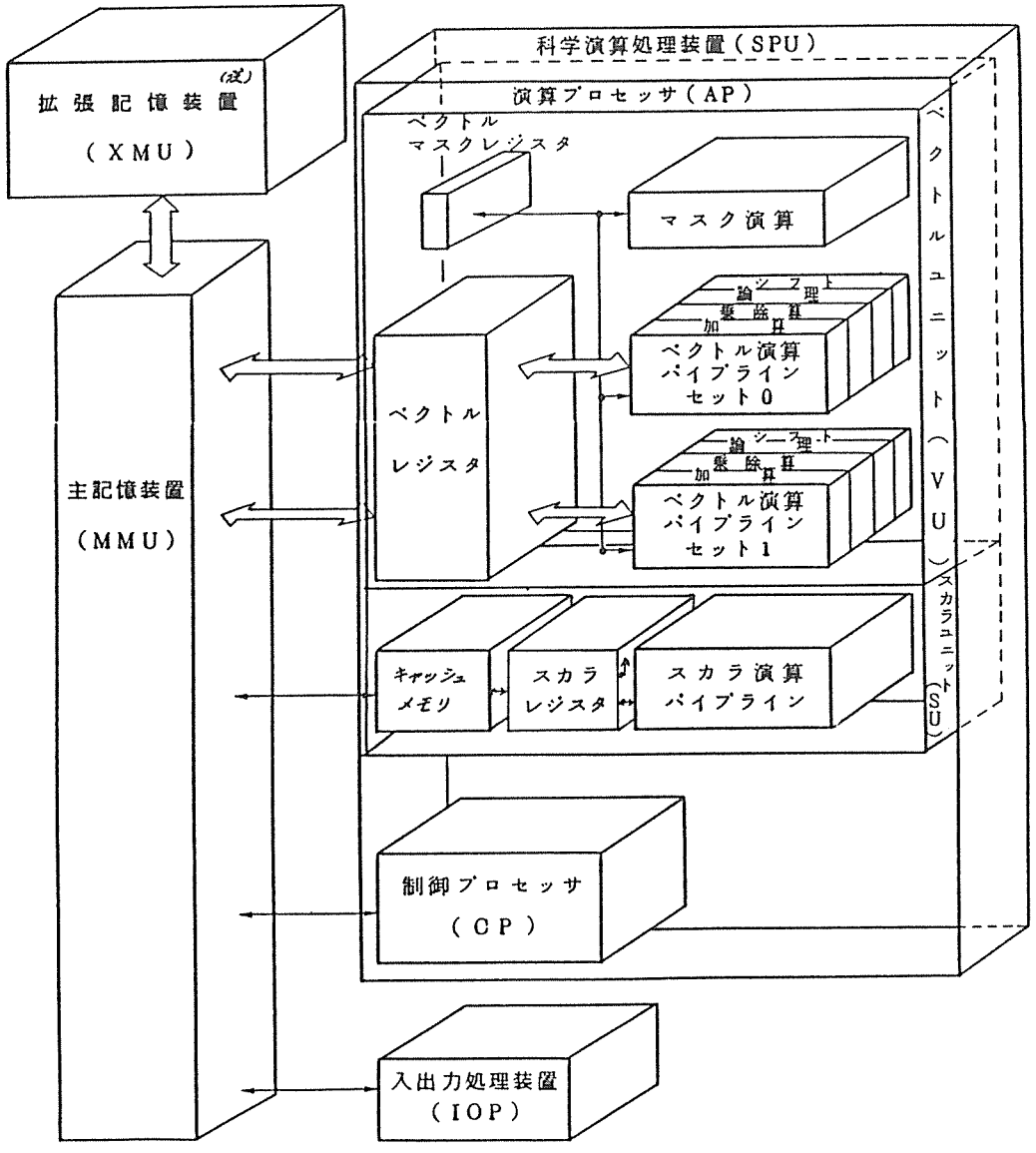
2. ハードウェアの概要

スーパーコンピュータＳＸシリーズは、三つのモデルＳＸ－２、ＳＸ－１とＳＸ－１Ｅから成り、最大ベクトル性能は、それぞれ1300MFLOPS、570MFLOPS、285MFLOPS（Million FLoating Operations Per Second，１秒間に浮動小数点命令を100万回）です。以下では本センターに導入されるＳＸ－１を中心に説明します。

2.1　システム構成

本センターに導入されるモデルであるＳＸ－１のシステム構成および内部構成を図１に、システムの諸元を表１に示します。図１において、科学演算処理装置（ＳＰＵ）は、二つの独立したプロセッサ、制御プロセッサ（ＣＰ）と演算プロセッサ（ＡＰ）から構成されています。制御プロセッサ上では、いわゆるオペレーティングシステム（ＯＳ）の機能が動作し、システム全体を制御します。また、ジョブの入出力処理、ファイル処理、タイムシェアリングシステム、FORTRANのコンパイルなどの機能も、制御プロセッサ上で実行されます。演算プロセッサは、科学技術計算専用に設計されたプロセッサで、スカラ命令およびベクトル命令を超高速に実行することを目的としています。利用者のプログラムは、演算プロセッサ上で実行されます。制御プロセッサと演算プロセッサによる機能分散構成により、演算プロセッサはベクトル処理を主体とする利用者プログラムの実行に専念することができます。

＋　日本電気㈱コンピュータ技術本部　　＋＋　日本電気㈱基本ソフトウェア開発本部
＋＋＋　関西日本電気ソフトウェア㈱　　＋＋＋＋　大阪大学大型計算機センター研究開発部



SPU : Scientific Processing Unit
AP : Arithmetic Processor
VU : Vector Unit
SU : Scalar Unit
CP : Control Processor
MMU : Main Memory Unit
XMU : Extended Memory Unit
IOP : Input Output Processor

(注) 拡張記憶装置(XMU)は本センターには導入されません。

図1 システム構成

表１　システム諸元

			ＳＸ－１
科学演算処理装置（ＳＰＵ）	最大ベクトル性能		570 MFLOPS
	クロック		7 ナノ秒
	レジスタ	ベクトルレジスタ	40 Kバイト
		ベクトルマスクレジスタ	128 ビット×８　個
		スカラレジスタ	128 個
	データ形式	固定小数点	32 ビット
		浮動小数点	32/64/128 ビット*（*スカラ命令のみ）
		論理	64 ビット
	命令数		171 個
	ベクトル演算パイプライン		2 セット
	ベクトル演算器		8 個
	スカラ演算パイプライン		1 セット
	マスク付きベクトル処理		可
	リストベクトル処理		可
	キャシュメモリ		64 Kバイト
	アドレス変換機構		1 Mバイトページ単位
主記憶装置（ＭＭＵ）	容量		64/128/256 Mバイト（注１）
	インタレース		256/512 ウェイ（注２）
	記憶素子		64 KビットＭＯＳスタティックＲＡＭ
入出力処理装置（ＩＯＰ）	総合データ転送能力		50 Mバイト／秒
	入出力チャネル本数		最大 32 本
拡張記憶装置（注３）（ＸＭＵ）	容量		128 Mバイト～ 2Gバイト
	転送速度		1.1 Gバイト／秒
	記憶素子		256 KビットＭＯＳダイナミックＲＡＭ

（注１）本センターでは 128Mバイト（その他に制御プロセッサメモリ 64 Mバイト）。
（注２）本センターでは 256ウェイ。
（注３）本センターでは導入されません。

2.2　多重並列パイプライン

ＳＸ－１の演算プロセッサは、多重並列パイプライン方式を採用しています。

図２に種々のパイプライン構成を示します。図２（a）は、基本のパイプライン構成であり、パイプラインにデータを連続的に入力することにより、計算機の処理単位である１クロックごとに、一つの結果が得られます。これに対し、同一構成のパイプラインをm本用意すると、m倍の性能となります。ＳＸ－１では、図２（b）に示すように、同一構成のパイプラインを２本備え、各パイプラインはそれぞれ、同一ベクトルの一つおきのベクトル要素の演算を分担します。２本のパイプラインは、クロックごとに、まったく同一の動作を行ないます。これによって、１クロックに同時に二つの結果を得るので、図２（a）に示した単一パイプラインに比べ、２倍の性能となります。この方式は、複数のパイプラインをＳＩＭＤ（Single Instruction Multiple Data stream）方式で動作させるので、並列パイプラインと呼びます。ベクトルの各要素のパイプラインへの割り付けは、ハードウェアが自動的に行なっています。したがって、利用者は、２本のパイプラインから構成されていることを、まったく意識する必要はありません。

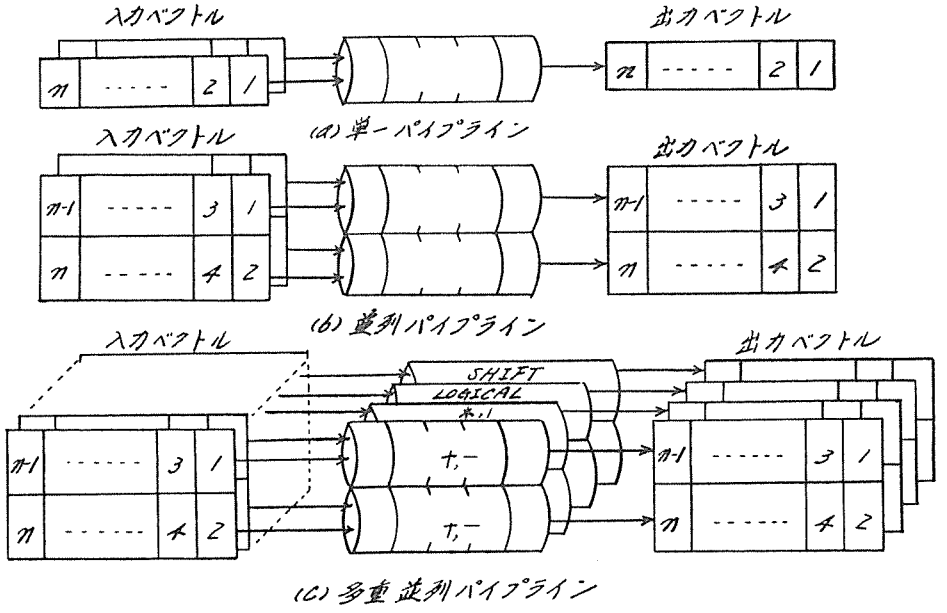


図２　種々のパイプライン方式

ＳＸ－１の演算プロセッサは、図２（c）に示したように、加算、乗除算、論理、シフトの４種のベクトル演算パイプラインを備えています。これら４種のパイプラインは、独立に動作できるので、多重パイプラインと呼びます。また、４種のパイプラインの各々は、前述のように、２本の並列パイプラインを構成しているので、ベクトル演算パイプラインは、合計８本です。これら８本のパイ

プラインは、すべて同時に動作が可能です。したがって、浮動小数点演算を行う加算および乗除算パイプラインのほかに論理演算、シフト演算パイプラインの性能も合わせると最大性能は1140MOPS（Million Operations Per Second, 1秒間に命令を100万回）となります。

2.3 メモリ構成と性能

パイプライン処理において、連続的に結果を得るためには、パイプラインにデータを充分な速さで供給し、かつ取り出すことが必要です。しかしながら、主記憶に使用するメモリ素子は、大容量ではあっても、演算速度に比べ遅いのが普通です。このために、単一のメモリ装置では、高いデータ転送能力をもたせることは不可能です。そこで、主記憶は、通常、バンクと呼ぶ複数個の独立して動作するモジュールで構成し、これに、図3で示したように、アドレスをバンク間にまたがるように割り振って構成します。このような方式をインタリーブ方式またはインタレース方式と言います。バンク数が多いほど、独立に動作可能なモジュールが増えるので、データ転送能力も高くなります。本センターに導入されるＳＸ－１の主記憶は、256個のバンクから構成されています。

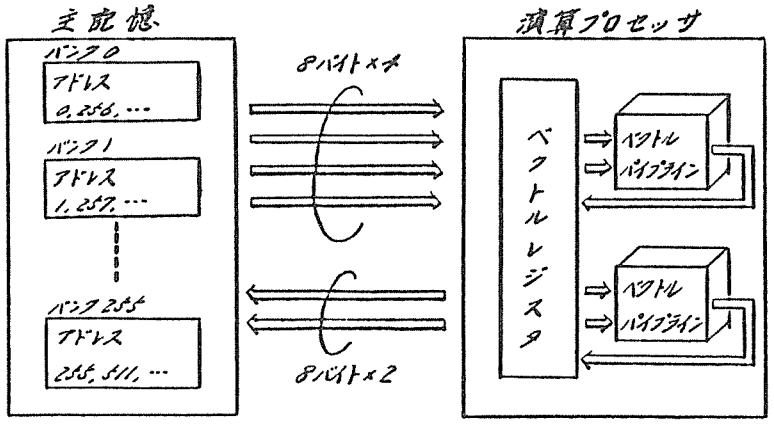


図3　主記憶構成とデータ転送路

主記憶とプロセッサ間のデータ転送能力を高めるには、さらに、データ転送路の幅を太くする必要があります。これは、パイプの中を流れる水の流量を増やすのと同じ考えです。ＳＸ－１では、図3に示したように、主記憶から演算プロセッサへのベクトルデータの読み出しパスは、8バイトのデータパスが4本、逆に、演算プロセッサから主記憶への書き込みパスは、8バイトのデータパスが2本となっています。これによって、演算プロセッサは、クロック毎に、4個の8バイト（倍精度）データ、即ち、毎秒4.6Gバイトのデータを受け取ることができます。

さらに、ＳＸ－１は、ベクトルデータを保持するために40Kバイトの、ベクトルレジスタを備えています。ベクトルレジスタは、高速の素子で構成されているので、パイプライン演算器に対する

データ転送能力を高めています。ベクトルレジスタには、ベクトル演算の中間結果を保持しておくことができ、主記憶と演算プロセッサ間のデータ転送頻度を軽減するのに役立っています。

2.4　ベクトル化率と性能

ベクトル処理能力がいくら高くても、プログラム中のベクトル命令で実行できる部分の割合が多くなくては、その性能を十分に引出すことができません。図4に示したように、ベクトル命令で実行できる割合、即ち、ベクトル化率βを次のように定義します。ベクトル化率βとは、あるプログラムをすべてスカラ命令で実行したときの実行時間をtとし、その内、ベクトル命令で実行できる時間をtvとすると、$\beta = tv/t$ とします。このとき、このコンピュータのスカラ性能psとベクトル性能pvの性能比αを$\alpha = pv/ps$ とすると、ベクトル化することによるスカラ性能（すべてスカラ命令で実行したときの性能）に対する性能比pは、$p = \alpha/(\beta + \alpha(1-\beta))$ となります。これを、pとβの関係として図示したのが、図5です。これからわかるように、十分なベクトル性能を引出すためには、ベクトル化率が十分高いことが必要です。ベクトル化率を高めるための工夫としては、次の3つが重要です。第1は、ベクトル化に適したアルゴリズムを採用したり、プログラミングを工夫するという利用者からのアプローチ、第2は、ベクトル化可能部分を見つけ出し、その部分を自動的にベクトル化するコンパイラの自動ベクトル化技術の向上、第3は、ハードウェアの方式としてベクトル化できる範囲を広げるために、種々のベクトル化機能を用意することです。ＳＸ－1では、ベクトル化率を高めるためのハードウェアの機能として、通常の四則演算に加えて、１Ｆ文を含むループのベクトル化のためのマスク付きベクトル演算機能、ベクトルの圧縮・伸張機能、リスト（間接指標）ベクトルを扱うための収集・拡散機能を備えています。図6に、これらの機能を示します。

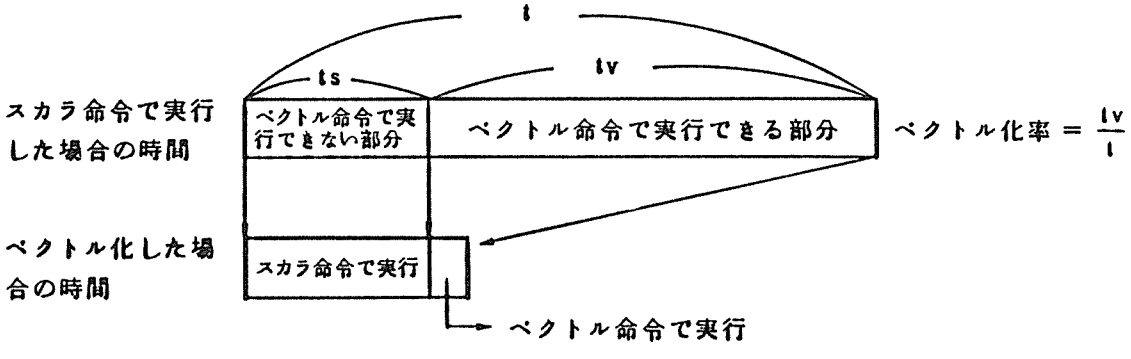


図4　ベクトル化率

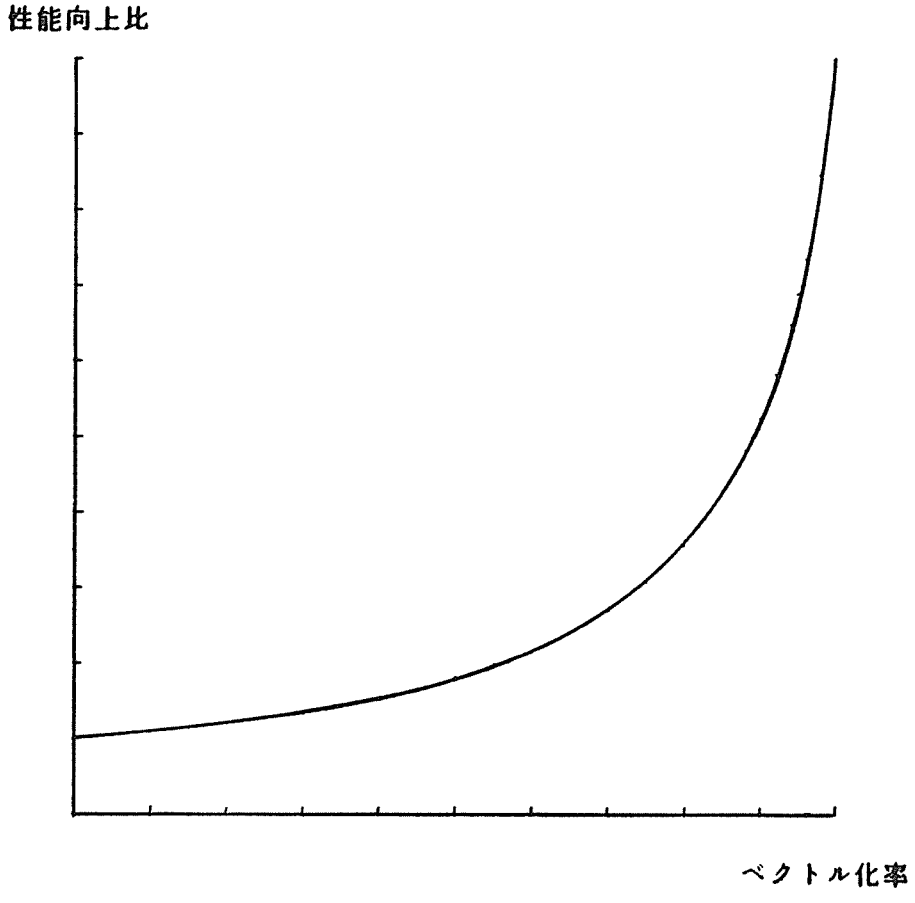
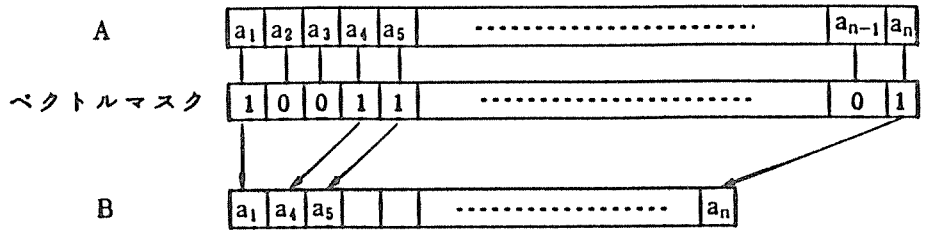
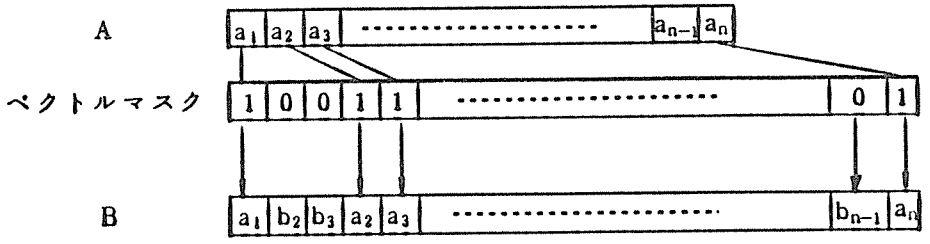


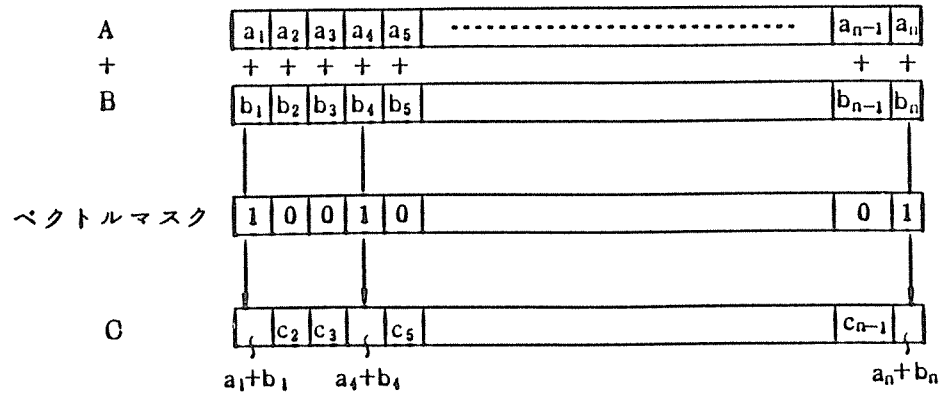
図5　ベクトル化率と性能



(a) ベクトルの圧縮 (compress)



(b) ベクトルの伸長 (expand)



(c) マスク付きベクトル演算

図6　1F文を含むループのためのベクトル機能

2.5　スカラ計算の高速化

プログラムの性質や計算アルゴリズムによっては、必ずしもベクトル化率を高くできない場合があります。また、ベクトル性能が高くなればなるほど、ジョブ全体としてはスカラ演算の実行時間の割合が高くなります。このような場合、実質的な性能を左右するのは、スカラ演算の実行時間になります。したがって、実効性能の向上には、スカラ演算の高速化が重要な要素です。図7にベクトル化率と性能比の関係を、スカラ性能が0.5倍の場合と比較して示します。

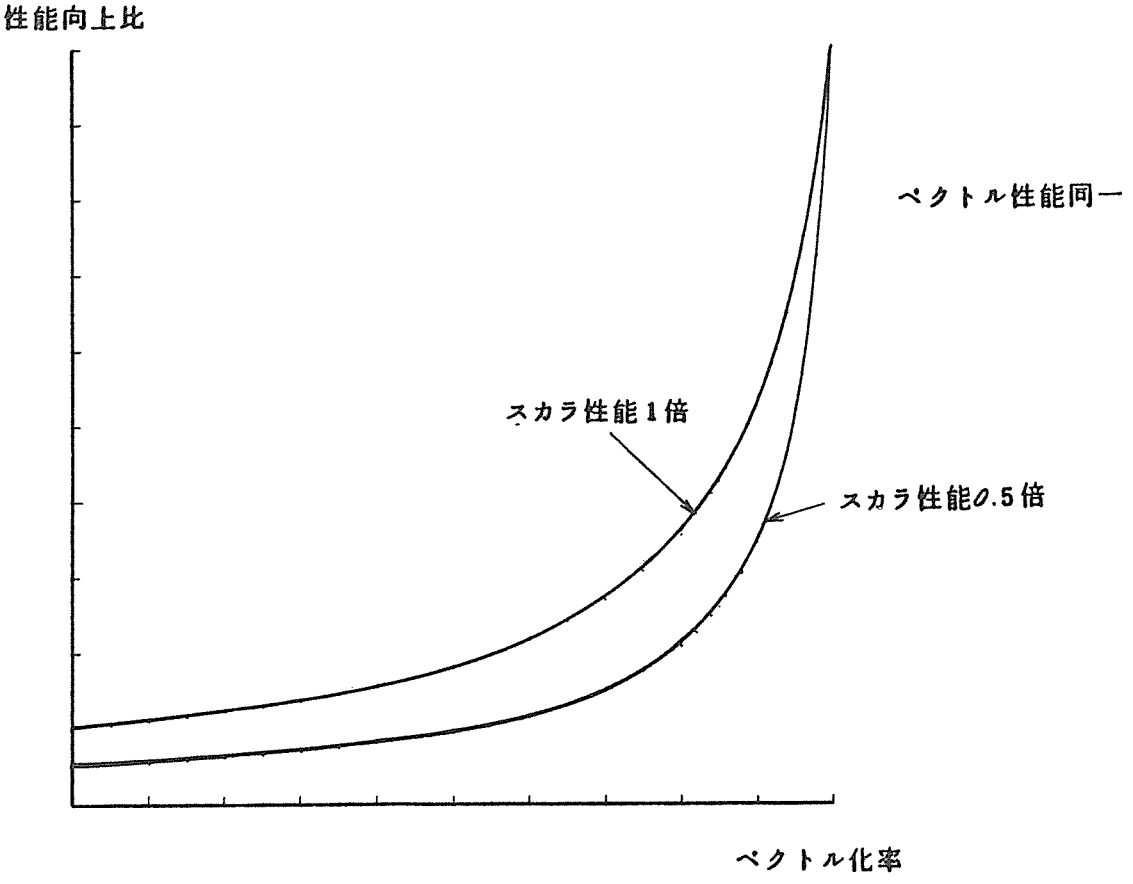


図7　ベクトル化率と性能　(2)

ＳＸ－１の演算プロセッサでは、スカラ計算の高速化のために、命令形式を単純化し、科学技術計算に頻繁に使用される命令だけを用意することによって、命令制御を簡単にし、スカラ命令においても、ベクトル命令と同様7ナノ秒のマシンサイクルで動作するようになっています。さらに、128個ものスカラレジスタが用意され、スカラ演算だけでなくアドレス計算にも使えるようにして、メモリアクセス頻度の減少と共に、レジスタ使用の融通性の増大が図られています。

さらに、スカラ演算においても演算のパイプライン化を行い、命令を連続的に実行できるように

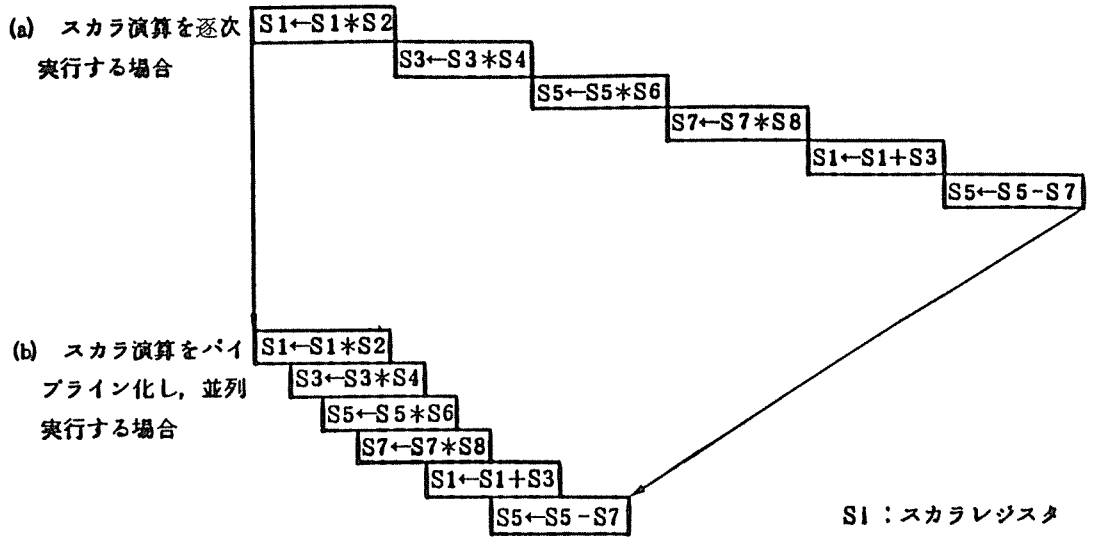


図8　スカラ演算のパイプライン化の効果

しています。従来の逐次方式と比較したときのスカラ演算パイプラインの効果を図8に示します。

図9は、演算プロセッサのスカラユニットの機能的な構成を示したものです。この図に示したように、スカラユニット内には、スカラデータおよび命令の高速アクセスのために、64Kバイトのキャッシュメモリと2Kバイトの命令バッファを備えています。命令バッファは、さらに、分岐履歴バッファを備え、分岐命令の分岐方向を記憶する分岐予測機構を備えることによって、分岐命令の高速実行が図られています。

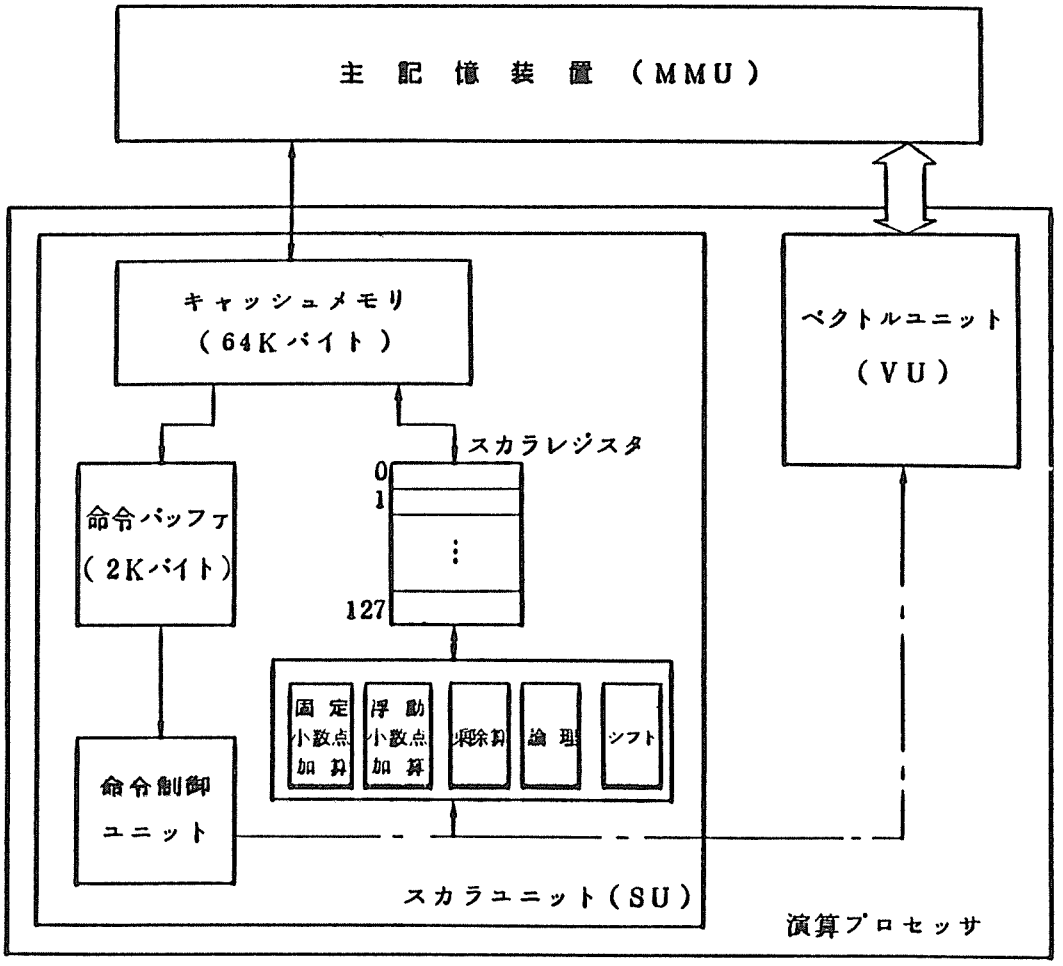


図9　スカラユニットの機能的構成

2.6　制御プロセッサ

制御プロセッサは、科学演算処理装置を構成する1つのプロセッサであり、利用者プログラムの演算処理を専用に実行する演算プロセッサを制御すると共に、システム全体の制御を行ないます。

制御プロセッサでは、ジョブの入力や出力編集処理、ファイル処理、メモリや入出力装置などの資源管理、ジョブのスケジューリング等の制御プログラムを実行します。また、演算プロセッサ上の利用者プログラムの実行と平行して、制御プロセッサ上でプログラムのコンパイル、結合編集、種々の性能向上支援ツールなども実行します。

その主な特長は、次の通りです。

①　プロセス制御による効率のよい多重プログラム制御

② セグメンテーションとページング方式による効果的な記憶保護と仮想記憶

制御プロセッサは、高速大容量の演算プロセッサ専用のメモリである演算プロセッサメモリとは別に、制御プロセッサ専用のメモリ（制御プロセッサメモリ）を持ち、オペレーティングシステム機能の大部分は、このメモリ上で動作します。したがって、ファイル処理やタイムシェアリング処理、プログラムのコンパイルなどは、ベクトル計算を行なう利用者プログラムの実行性能に影響を与えることなく、実行されます。

2.7 システム利用形態

ＳＸシステムは、大規模な処理能力を実現するための複合システムファシリティ（Multi System Facility, MSF）により、ＡＣＯＳ１０００などの既設コンピュータシステムから、バックエンドプロセッサ（Back End Processor, BEP）構成でＳＸシステムを利用することができます。また、スタンドアロン構成でも、十分に実用システムとして構成できます。本センターでは、ＢＥＰ構成を取りますので、以下では、この構成について説明します。

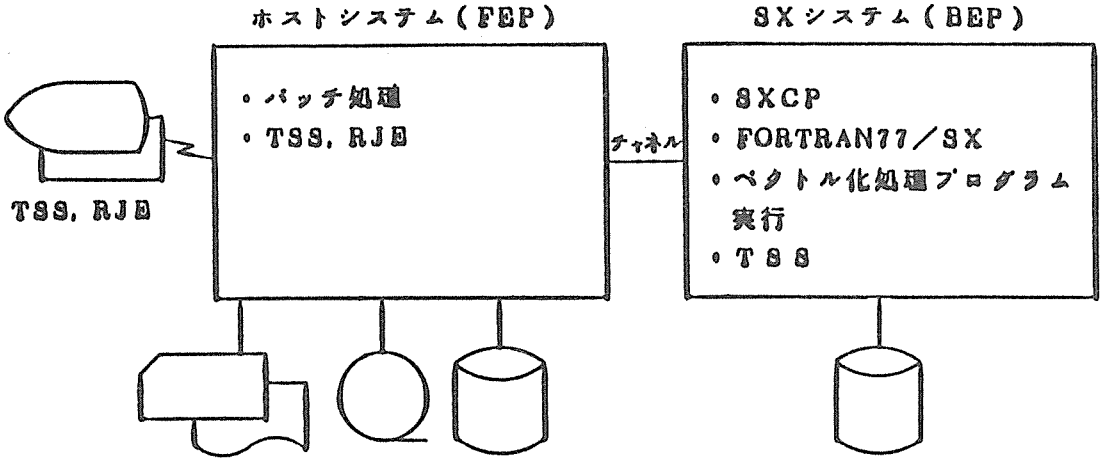


図10 バックエンドプロセッサ構成

ＢＥＰ構成は、既設のホストコンピュータをフロントエンドプロセッサ（Front End Processor, ＦＥＰ）とし、ＳＸシステムをバックエンドプロセッサ（ＢＥＰ）とする方式です（図10参照）。この方式は、ジョブ転送機能、対話処理機能により、ＦＥＰからローカル／リモートバッチ処理、対話処理のどのモードでもＳＸを利用可能です。また、ファイル転送機能を利用することにより、ファイルをホストコンピュータ側でも、ＳＸシステム側でも管理することができます。したがって、この構成での運用当初には、ＳＸ側では、ベクトル化処理プログラムのコンパイル、結合、実行およびプログラム性能改善ツールの利用などのＳＸでなければ出来ないものに限定し、ＦＥＰ側では、

それら以外の処理プログラムの編集、デバッギングなどを行ない、このシステムの運用に慣れていくにつれて、徐々にＳＸの利用を広げ、深めていくことが可能です。

3. ソフトウェアの概要

本章では、ＳＸシステムのソフトウェアの中で、利用者に直接関係するFORTRAN77/SX およびANALYZER/SX、VECTORIZER/SXなどを中心として紹介します。これらのソフトウェアは、容易にＳＸシステムを使用でき、しかも、ＳＸシステムの性能を十分に引き出せるようにするという目標のもとに開発されています。

3.1　言語処理系

大規模科学技術計算プログラムは、現状ではそのほとんど全てが FORTRAN で書かれています。ＳＸシステムにおいても、プログラミングは、FORTRANによって行ないます。図11は、FORTRAN

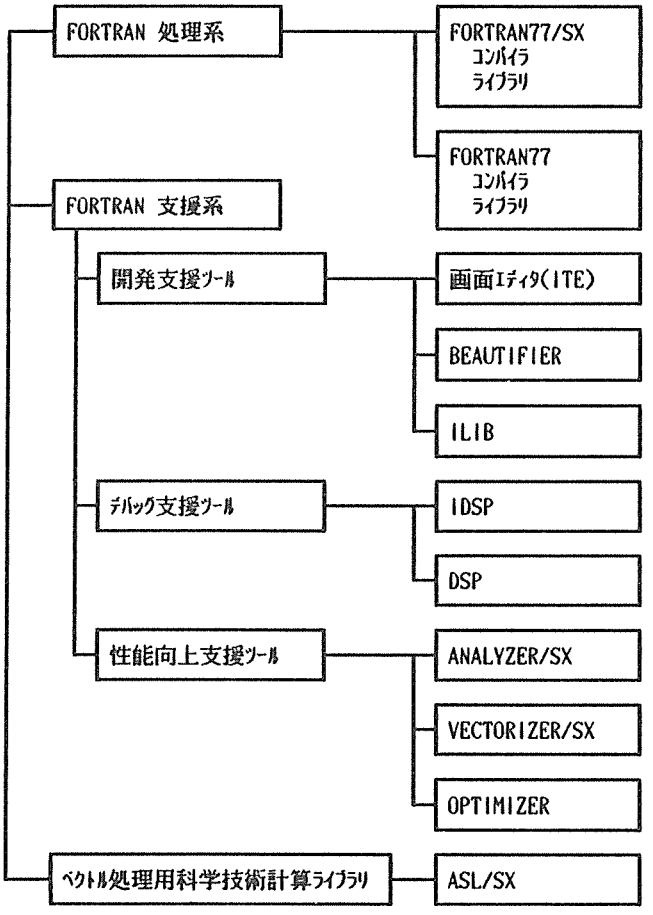


図 11　FORTRAN77/SXを中心とした体系図

を中心としたＳＸシステムの言語処理系の体系図です。

FORTRAN77/SXが演算プロセッサのベクトル性能およびスカラ性能を引き出すためのFORTRANであるのに対し、FORTRAN77は、後述のDSP/IDSPとともに、制御プロセッサ上でのFORTRANプログラムのデバッグに使用するFORTRANです。

BEAUTIFIERは、原始プログラムの清書処理を行うツールです。ILIB(Integrated Librarian)は、プログラムの開発過程で用いられる各種ライブラリについて、容易に管理を行なうためのツールです。

IDSP(Interactive Debugging Support Program)およびDSP(Debugging Support Program)は、FORTRANプログラムのデバッグを支援するツールで、原始プログラム中の行番号や英字名を使用したシンボリックデバッグを可能にしています。

3.2　ＦＯＲＴＲＡＮ77／ＳＸ

FORTRAN77/SXは、ＳＸシステム専用に開発されたFORTRANです。最新のJIS FORTRAN上位水準に準拠した言語仕様を持つとともに、自動ベクトル化機能および最新化機能を備えていますので、ＳＸシステムの性能を引き出すことができます。また、多くの拡張機能を持つとともに、汎用機のFORTRAN77との互換性を考慮しています。

3.2.1　自動ベクトル化機能

ＳＸシステムは、ベクトルデータ間の演算を高速に実行することができる命令(ベクトル命令)を多数備えています。ＳＸシステムの性能を引き出すためには、通常の命令(スカラ命令)のかわりに、できるだけベクトル命令を使用する、すなわちベクトル化する必要があります。コンパイラがプログラムを解析してベクトル命令で実行可能な部分を自動的に検出し、その部分に対してベクトル命令を生成することを、自動ベクトル化といいます。

FORTRAN77/SXコンパイラは、ＡＣＯＳ１０００等の統合アレイプロセッサ(ＩＡＰ)向けの自動ベクトル化機能を強化発展させた、自動ベクトル化機能を備えています。表２は、FORTRAN77/SXの自動ベクトル化機能の概要をまとめたものです。なお、拡張ベクトル化機能は、ベクトル化の対象範囲を広げるために、そのままではベクトル化が困難なループを変形することによりベクトル化したり、生成されるベクトル命令の効率を向上させたりする機能です。以下に、その中の主な機能について述べます。

(1)　ＩＦ文を含むループのベクトル化

ＤＯループにＩＦ文などの条件文が含まれている場合、ベクトルマスク生成命令や、マスク付きベクトル演算命令を用いてベクトル化します。図12に例を示します。

表２　FORTRAN77/SXの自動ベクトル化の概要

(1) FORTRAN77/SXの自動ベクトル化の対象

対象となるループ	DOループ
対象となるデータの型	整数型、論理型 実数型（単精度、倍精度） 複素数型（単精度、倍精度）
対象となる文	代入文、CONTINUE文 IF文（算術、論理、ブロック） ELSE IF 文、ELSE文 END IF文 単純GO TO 文
対象となる演算	加減乗除算、べき算 型変換 組込み関数 内積、総和、累積 漸化式 最大、最小 サーチ
対象となるベクトル	連続、等間隔ベクトル 間接指標ベクトル 指標変数

(2) FORTRAN77/SXの拡張ベクトル化機能
(a) ベクトル化の対象範囲の拡大

ベクトル化不能部分を含むループ	ループ分割によりベクトル化
入れ子をなすループ	外側ループもベクトル化
外部関数の引用を含むループ	中継ルーチンの挿入により疑似ベクトル関数化してベクトル化
データの参照関係が適合しないループ	文の入れ換えまたは作業用ベクトルの導入によりベクトル化
ベクトル化の可否が判断できないループ	ベクトル化指示行の指示によりベクトル化

(b) ベクトル化効率の向上

密な多重ループ	一重化またはループの入れ換え
境界判定を含むループ	ループ変形により判定を除去
その他	ベクトル化指示行による種々の指示

```
      DO 10 I=1,N
        IF(A(I).GE.0.0) THEN
          X(I)=SIN(B(I)+C(I))
        END IF
10    CONTINUE
```

↓

```
Mi  ←  1 (if Ai≧0.0)          ベクトルマスク 生成
       0 (if Ai<0.0)
T1i ←  Bi+Ci(if Mi=1)         マスク 付きベクトル演算
T2i ←  compress(T1i):Mi       ベクトル圧縮
T3i ←  SIN(T2i)
T4i ←  expand(T3i):Mi         ベクトル伸長
Xi  ←  T4i(if Mi=1)
```

図 12　ＩＦ文を含むループのベクトル化例

(2)　ベクトル添字を含むループのベクトル化

スパース行列（非零のデータがわずかしかない行列）を扱う時などに頻繁に現れる、ベクトルを添字として持つ配列要素（これは間接指標ベクトルという）を含むループは、ベクトル収集・拡散命令を用いてベクトル化します（図13(1)参照）。

(3)　指標変数を含むループのベクトル化

ＤＯループ中の式にＤＯ変数などの指標変数が直接現れる場合にも、数列生成機能を用いることにより、ベクトル化します（図13(2)参照）。

(4)　マクロ演算のベクトル化

内積、総和、累積、漸化式、最大、最小、サーチなどのマクロ演算も、コンパイラはそれを認識してベクトル化します（図13(3)参照）。

(1) 間接指標ベクトルを含むループ

```
      DO 10 I=1,N
        K=IDX(I)
        A(K)=B(I)*X(K)
10    CONTINUE
```

(2) 指標変数を含むループ

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

(3) 最大値を求めるループ

```
      DO 10 I=1,N
        IF(A(I).GT.AMAX) THEN
          AMAX=A(I)
          MXIDX=I
        END IF
10    CONTINUE
```

図 13　ベクトル化されるループの例

(5)　ループ分割によるベクトル化

ＤＯループの中に、ベクトル化を阻害するような文あるいは要素が含まれている場合、その前後でループを分割し、その後、ベクトル化可能な部分をベクトル化します。

(6)　外側ループのベクトル化

ベクトル化は、基本的には最深ＤＯループを対象に行ないます。しかし、定められた条件を満た

せば、最深ＤＯループの前後でループを分割することにより、その最深ループの外側のループもベクトル化することが可能です。図14にその例を示します。

(7) ベクトル化指示行

原始プログラムの情報だけでは、コンパイラはベクトル化に関する十分な情報を得られない場合があります。ベクトル化指示行は、そのような場合に、利用者がコンパイラにベクトル化に関する情報を与えるためのものです。その情報には、ベクトル化の可否に関するもの、効率のよいベクトル命令を生成するためのものなどがあります。

```
      DO 10 I=1,N
        A(I)=B(I)+C(I)
        DO 20 J=1,M
          X(J,I)=Y(J,I)+Z(J,I)
20      CONTINUE
10    CONTINUE
```

↓

```
      DO 101 I=1,N
101     A(I)=B(I)+C(I)
      DO 102 I=1,N
        DO 20 J=1,M
          X(J,I)=Y(J,I)+Z(J,I)
20      CONTINUE
102   CONTINUE
```

↓

それぞれの部分をベクトル化

図 14　外側ループのベクトル化例

3.2.2　最適化機能

FORTRAN77/SX コンパイラは、共通式の削除や不変式のループ外への移動など、汎用機において行っていた最適化を適用するとともに、ＳＸのハードウェアを効率よく動作させるための最適化技術も取り入れています。しかも、自動ベクトル化や最適化の処理に先立って、プログラム全体のデータの流れや制御の流れを解析し、その結果を利用しながらこれらの処理を行なうため、効率のよい目的プログラムを生成することができます。ここでは、ＳＸのハードウェア向きの最適化技術のうち、主要なものについて述べます。

(1) 命令の並べ換え

ＳＸハードウェアは、高速化のために、スカラ処理においては演算パイプラインを採用したり、ベクトル処理においては同時に動作可能な複数の演算器を置いたりしています。これらの工夫を活かすためには、直前の命令の演算結果を直後の命令で参照しないようにしたり、同一の演算器を使用する命令が連続しないようにするなど、命令の順序の最適配置が必要になります。この処理を、命令の並べ換えと呼びます。コンパイラは、ハードウェアの動きをシミュレートしながら、命令の並べ換え処理を行ないます。この最適化は、128個という多数のスカラレジスタと相まって、特にスカラ処理において効果を発揮します。

(2) 不要終値保証の除去

ＤＯループ中で変数が定義されている場合、通常その変数は作業用ベクトルによって置き換えられますが、ループを出た後でその変数が引用される可能性があるため、ループの処理後、その変数

にループ終了時の最終的な値を設定する必要がありました。しかし、FORTRAN77/SX では、ループを出た後で、もしその変数の値が使用されていない場合には、その変数に対する終値保証を行なわないようにします。

(3) レジスタ割り当ての最適化

128個のスカラレジスタ、最大80 Kバイト（ＳＸ－１では40 Kバイト）のベクトルレジスタおよび8個のマスクレジスタに対する、ロード・ストア回数を最小にし、またレジスタ競合を最小にするべく割り当てます。

3.2.3 入出力の高速化

大規模科学技術計算では、計算機の高速化とともに解くべき問題の規模も大きくなり、主記憶容量を大幅に越えるような大量のデータを扱うものも、決して稀ではありません。このようなプログラムでは、計算速度の高速化ばかりではなく、入出力処理の高速化も重要です。FORTRAN77/SXでは、そのための高速入出力機能を用意していますが、ここでは省略します。

3.3 性能向上支援ツール

すでに述べたように、ＳＸシステムの性能を十分に引き出すためには、ベクトル化率を高くすることが必要です。FORTRAN77/SX のコンパイラは、ベクトル化能力を備えており、ベクトル化に適合したプログラムの場合には、そのまま翻訳するだけで高いベクトル化率を得ることができます。しかし、その場合でも、プログラムをわずかに手直しするだけで、ベクトル化率をさらに高め、性能を一層向上させることが可能です。また、ベクトル化に適合していないプログラムの場合には、適合するように書き直せば、ベクトル化率の改善により、性能を向上させることができます。

ＳＸシステムでは、このような観点から、利用者によるベクトル化に着目した書き直し作業（チューニング）を支援するツールとして、ANALYZER/SX、VECTORIZER/SXおよびOPTIMIZERの3つのツールが用意されています。

3.3.1 ＡＮＡＬＹＺＥＲ／ＳＸ

ANALYZER/SX は、プログラムの動的特性情報および静的構造情報を解析するためのツールです。動的特性情報は、プログラムを構成する各モジュールごとの引用回数や実行コスト（プログラム全体に対する実行時間比を近似的に示したもの）、モジュール内の各ループごとの実行コストやベクトル化の可否、全体のベクトル化率などの情報で、プログラムのどの部分にチューニングの努力を集中するべきかを知るのに役立ちます。一方、静的構造情報は、プログラムのモジュール構造や、共通データのモジュール間の相互参照関係などの情報で、プログラムの大幅な書き直しが必

要になる場合に有効です。

3.3.2 VECTORIZER/SX

VECTORIZER/SXは、ANALYZER/SXが解析した動的特性情報や、ベクトル化不可理由を示す診断情報などを画面上で見ながら、直接プログラムの修正・改良を行なう画面型のベクトル化支援ツールです。VECTORIZER/SXを使用するためには、あらかじめ、ANALYZER/SXを動かし、動的解析情報をファイルに出力しておく必要があります。図15に、VECTORIZER/SXの各画面に表示される情報と、操作手順を示します。

3.3.3 OPTIMIZER

OPTIMIZERは、FORTRAN原始プログラムに最適化処理を施し、より効率のよい形に変換するツールです。SXシステム専用に開発したものではなく、汎用のツールですが、最適化項目の一つである副プログラムのインライン展開機能は、ベクトル化という観点からも有効です。図16は、外部関数のインライン展開により、ループがベクトル化可能となる例です。

```
      DO 10 I=1,N
   10 X(I)=Y(I)*FNORM(Z(I))
                 .
                 .
                 .
      FUNCTION FNORM(X)
      PARAMETER(A=0.39894228)
      FNORM=A*EXP(-0.5*X**2)
      RETURN
      END
```

↓ OPTIMIZRによる関数FNORM のインライン展開

```
      DO 10 I=1,N
      FNORM=A*EXP(-0.5*Z(I)**2)
   10 X(I)=Y(I)*FNORM
```

↓

ベクトル化

図 16　OPTIMIZERによるインライン展開の例

4. おわりに

SXシステムの構成、ハードウェアおよびソフトウェアの概要、主要な利用形態について紹介しました。本センターにおける利用法等については、文献 [4] および、それに掲げられている参考文献を参照して下さい。

(参考文献)

[1] 渡辺：「スーパーコンピュータのアーキテクチャ」、日本機械学会関西支部第 134 回講習会テキスト。

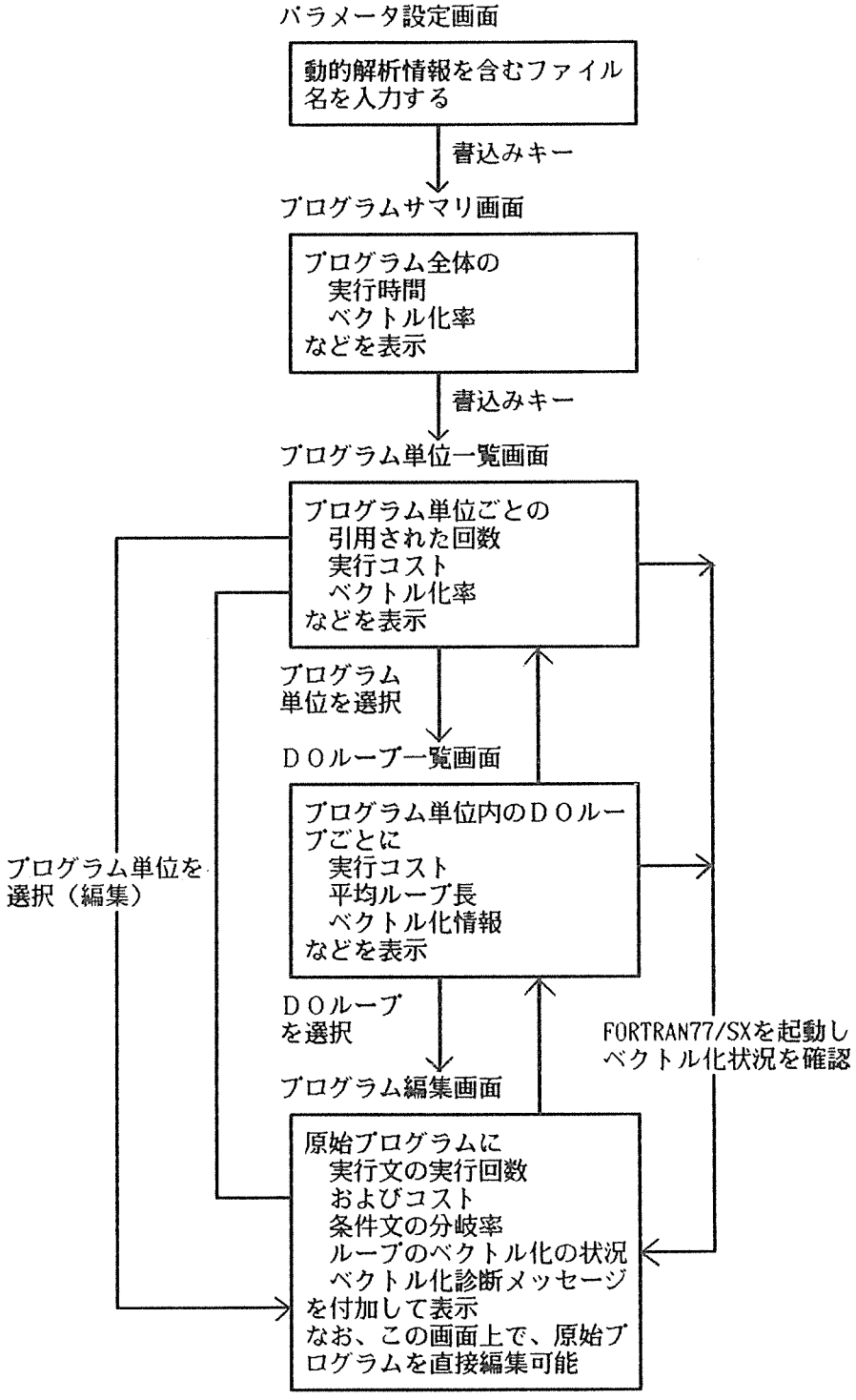


図 15　VECTORIZER／SXの画面と操作手順

〔2〕 岩屋、久保田、古勝、渡辺、近藤、辻、川村、片山、奥村、小林、仙波：「ＳＸシステムの概要」、ＮＥＣ技報、Vol.39、No.1（1986）。

〔3〕 近藤、北脇、片山、塚越、安部、石谷、松田、大橋、堀江：「ＳＸシステムの言語処理系」、ＮＥＣ技報、Vol.39、No.1（1986）。

〔4〕 藤井：「スーパーコンピュータＳＸ－１の概要(2)」、大阪大学大型計算機センターニュース、Vol.15、No.4（1986）。