解　説

6. 命令セットアーキテクチャの具体例

6.12 スーパコンピュータ NEC SX システム†

渡　辺　貞††

1. はじめに

スーパコンピュータ SX システムは，システム構成上，制御プロセッサ (CP) と演算プロセッサ (AP) とによる機能分散構成を採用している．制御プロセッサはオペレーティングシステムの実行，入出力処理，コンパイルなど，主としてシステム全体の制御を行うプロセッサであり，アーキテクチャは汎用コンピュータ ACOS システムを，SX の制御用に拡張したものである．一方，演算プロセッサはユーザプログラムを専用に実行する科学技術計算専用プロセッサで，高速のベクトルおよびスカラ演算機能を備えている．ここでは，演算プロセッサの命令セットアーキテクチャについて，設計方針および機能について説明する．

演算プロセッサのアーキテクチャ設定においては，科学技術計算の高速化を第一の目的とした．このために，高速マシンサイクルを達成でき，演算パイプラインが有効に働き，かつ，LSI 化に適したアーキテクチャの設定を目指した．

高速マシンサイクル (6 ns) の達成のためには，必然的に，マイクロプログラム制御ではなく，ハードワイヤード制御にならざるを得なかった．さらに，LSI 化に適したアーキテクチャということで，できるだけ簡単な命令形式，命令機能，アドレッシングを採用することにした．しかしながら，命令機能をあまりに単純にすると，コンパイラ作成の複雑化や，制御機能の命令ステップ数の増加を招く．そこで，アーキテクチャとしては，科学技術計算に必要な機能を重視し，それに，必要最少限のシステム制御機能を加え，かつ，強力なベクトル計算機能のサポートということを基本にして設定した．結果として，いわゆる RISC アーキテクチャの多くの特徴が採り入れられたものとなっている．

表-1 演算プロセッサ諸元

命令	スカラ命令	83 種
	ベクトル命令	88 種
	計	171 種
データ	固定小数点	16, 32 ビット
	浮動小数点	32, 64, 128 ビット
	論　理	64 ビット
レジスタ	スカラレジスタ	64 ビット×128 個
	ベクトルレジスタ	256 語×40 個
	ベクトルマスク	256 ビット×8 個
	その他	命令実行数カウンタ，ベクトル命令数カウンタ，ベクトル要素数カウンタ，など

表-1 に，演算プロセッサの主要なアーキテクチャ諸元を示す．

2. 命令の形式と機能

2.1 命令形式とアドレッシング

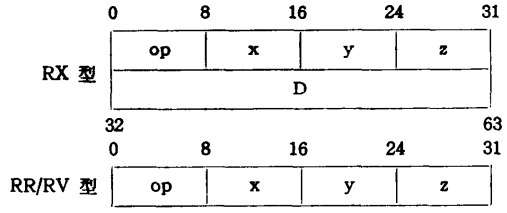
図-1 に，代表的な命令形式を示す．命令形式は，できるだけ単純にして，命令取りだし/デコードなどの制御回りが簡単になるようにした．命令長は 4 バイトと 8 バイトの 2 種である．4 バイト命令は主として演算系の命令，8 バイト命令はロード/ストア/分岐などのメモリアクセス系の命令である．

RR/RV 型命令は，4 バイト長であり，フィールド x, y, z でレジスタを指定する．RR 型はスカラ命令用，RV 型はベクトル命令用である．フィールド x, y, z の先頭ビットは制御用ビットである．たとえば，RV 型で x フィールドの先頭ビットはベクトルのマスク付き演算を指定するために使われる．フィールド y, z の先頭ビットは即値データかレジスタ指定かを示す．

すべての演算は，レジスタ上のオペランドに対して行われる．すなわち，演算は Sx←Sy op Sz の形で行われる．ここで，Sx, Sy, Sz は，おのおの x, y, z で

† An Example of Instruction Set Processor Architecture NEC SX by Tadashi WATANABE (Computer Engineering Division, NEC Corporation).

†† 日本電気(株)



op：命令コード
x, y, z：スカラレジスタ，ベクトルレジスタまたは即値指定
D：オペランド・アドレスの変位指定

図-1 主な命令形式

指定されるスカラレジスタ（Sレジスタ）または即値を表す．3オペランド形式にしたことで，レジスタの使用に融通性をもたせることができる．ここで分かるとおり，本アーキテクチャはRISCの一つの特徴である演算とメモリアクセスを分離したロード/ストアアーキテクチャである．これによって，コンパイラが命令リオーダリングと呼ぶ最適化技術により，命令をうまく並べ替えることによって，パイプラインを有効に働かせることができる．たとえば，次のように，

ロード S0←Memory （1）

加算 S2←S0+S1 （2）

とある場合，（1）のロードとそれをオペランドとして使用する（2）の加算命令の間に，それらとは無関係の命令を挿入することにより，メモリからのロードによるアクセス時間を無駄なくほかの命令実行に使用できる．

RX 型命令は，ロード/ストア系命令で使用する．フィールドy，zでは，ベースおよびインデックス用のSレジスタを指定する．ベース，インデックスおよび変位により生成された論理アドレスは，1M バイトのページに分割されている主記憶にマッピングされる．バッチ指向の大容量メモリを必要とするアプリケーションを効率よく実行するために，ページサイズを 1M バイトと大きくした．

2.2 スカラ演算の命令セットアーキテクチャ

ベクトル計算を高速に実行するスーパコンピュータといえども，実効的な高速計算のためには，ベクトル化できない部分の実行，すなわち，スカラ演算の高速化がきわめて重要である．高速スカラ演算を目指して，命令セットとしては，科学技術計算に必要な機能を重視し，64 ビット/語をベースとした命令セットを設定した．したがって，10 進演算命令やバイト単位のメモリ間移送命令などは備えていない．当初は，バイト単位のレジスタへのロード/ストアも止めようとしたが，ソフトの作りやすさを考慮して，結局，1バイトのロード/ストア命令および 16 ビット整数用の 2 バイトロード/ストア命令をサポートすることにした．

スカラ命令用のレジスタとして，128 個のスカラレジスタ（Sレジスタ）を用意した．演算プロセッサは，メモリの高速アクセスのために，キャッシュメモリを備えている．しかし，キャッシュメモリの物理的

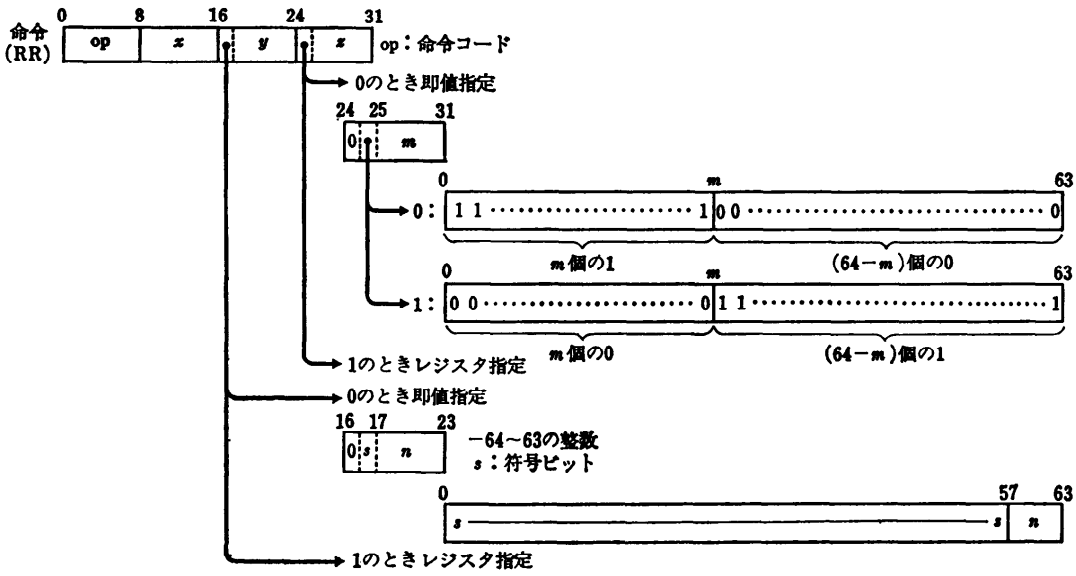


図-2 即値生成の方法

大きさやディレクトリアクセスの遅れのために，そのアクセス時間は，超高速実行のためには十分速いとはいえない．そこで，メモリアクセス頻度の減少と高速データアクセスのために，128 個のスカラレジスタを備えることにした．レジスタ指定フィールドで指定できる最大数（7 ビット）ということで，128 個のレジスタ数にした．

さらに，即値データの生成を工夫することによって，メモリアクセス頻度と命令ステップ数の削減を図った．図-2 に，即値データの生成方法を示す．命令の y, z フィールドで即値データを指定できるが，y フィールドでは，−64〜63 の整数，z フィールドでは論理定数を生成できる．これを組み合わせることによって，たとえば，1 バイトデータを抽出するためのマスクデータなどは，メモリをアクセスすることなく，1 命令で生成できる．このほかにも，浮動小数点データ f の絶対値 $|f|$，符号反転 $(-f)$ なども，即値データと f との論理演算命令によって求めることができる．これらの代表的な例を図-3 に示す．

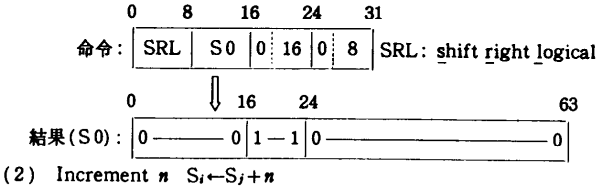
また演算プロセッサは，比較結果を示すためのいわゆる条件コードを備えていない．条件分岐命令は S レジスタの内容を調べ，その内容に従って分岐する．条件コードによって分岐するアーキテクチャの場合，条件コードを変化させる命令 1 とそれに従って分岐する条件分岐命令 2 とは不可分であり，通常，1 の命令の直後に 2 の命令を実行しなければならない．これに対して，レジスタの内容で分岐するアーキテクチャの場合，レジスタの内容を変える命令 3 とその内容によって分岐する条件分岐命令 4 とは，通常の演算命令とその結果を使用する命令とまったく同じ関係となっている．したがって，コンパイラは，演算パイプラインを有効に働かせるために，3 と 4 の命令の間に独立な命令を挿入することにより，最適化を図っている．

2.3 ベクトル演算の命令セットアーキテクチャ

スーパコンピュータを特徴づけるものは，ベクトル演算の高速実行である．高速実行のために重要なことは，パイプライン化されたベクトル演算器を高速クロックで動作させること，それらを並列に実行させること，さらに，ベクトルデータを高速にベクトル演算器に出し入れすることである．このためにアーキテクチャ上配慮したことの一つは，ベクトルデータの高速転送のために，ベクトルレジスタを用意したことである．一つのベクトルレジスタは，モデルによって異なるが，最大 256 語（1 語＝64 ビット）のベクトルデータを保持することができる．ベクトルレジスタは，合計 40 個あり，このうち 8 個はベクトル演算専用のレジスタ，32 個はベクトル演算の中間データなどを保持するバッファとしての役割をもつ．ベクトルレジスタを備えることにより，メモリとプロセッサ間のデータ転送量を少なくし，メモリ負荷を軽減していると同時に，ベクトルデータの高速アクセスに大きな効果がある．

プロセッサ性能の向上のためには，ベクトル演算の高速化のほかに，プログラムのベクトル化可能部分（ベクトル化率）を高めることがきわめて重要である．ベクトル化率の向上は，ベクトル計算向きのアルゴリズムやプログラミングの工夫，コンパイラの改善によって得られる

(1) 1 バイトのマスクデータ生成



(2) Increment n $S_i \leftarrow S_j + n$
ADD S_i, S_j, n $S_i \leftarrow S_j + n$

(3) Decrement n $S_i \leftarrow S_j - n$
ADD $S_i, S_j - n$ $S_i \leftarrow S_j - n$

(4) Floating Point Absolute $S_i \leftarrow |S_j|$
AND S_i, S_j, (1)0 $S_i \leftarrow S_j \wedge$ '011.....1'

(5) Floating Point Complement $S_i \leftarrow -S_j$
XOR S_i, S_j, (1)1 $S_i \leftarrow S_j \oplus$ '100.....0'

(6) Normalize $S_i \leftarrow$ Normalize (S_j)
FAD $S_i, S_j, 0$ $S_i \leftarrow S_j + 0.0$

図-3 即値を使用した演算例

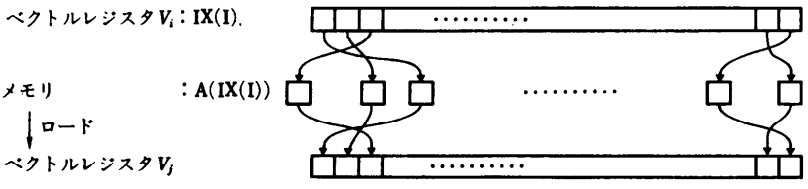


図-4 リストベクトル (Vector Gather 命令の動作)

が，ベクトル化ができるような機能をもたせることがアーキテクチャ上のキーポイントとなる．このために，取り扱えるベクトルデータとしては，メモリ上に連続して配置されている連続ベクトルのほかに，要素間の間隔が一定で飛び飛びのベクトル（ストライドベクトル）およびメモリにランダムに配置されているリストベクトルを扱えるようにしている．図-4 にリストベクトルのアクセス方法を示す．図から分かるように，メモリ上のベクトルデータは，ベクトルレジスタ中のデータをアドレスとしてアクセスされる．

ベクトル化率を高めるためには，Fortran の IF 文を含む DO ループをベクトル化することが必要である．このためのベクトル機能として，マスク付き演算を用意している．マスク付き演算とは，図-5 に示したように，ベクトル同士の比較結果を格納するベクトルマスクレジスタを備え，このベクトルマスクレジスタの内容に従って演算を行うかどうかを制御するベクトル機能のことである．8 個のベクトルマスクレジスタを備え，これらのレジスタ間の論理演算命令をサポートすることによって，ネストした IF 文や IF 文の中に多項の比較文を含む場合も容易にベクトル化できるようにしている．なお，1 個のベクトルマスクレジスタの長さは，ベクトルレジスタの大きさに対応して，最大 256 ビットである．さらに，IF 文の分岐の方向に従って効率のよいベクトル命令列を生成するための機能として，ベクトルの圧縮 (compress)/伸長 (expand) 機能をサポートしている．

このほかのベクトル機能として，$A(I)=B(I)+A(I-1)*C(I)$ のような 1 次漸化式をベクトル化するベクトル命令，FFT の高速化のためのベクトルビットリバース命令なども用意した．

2.4 その他の機能

SX では，大量の入出力データを高速に転送するために，半導体記憶で構成される拡張記憶装置 (XMU) を設けている．XMU と主記憶とは，演算プロセッサ中の 1 k バイトを単位とする移送命令によって，データ転送を実行する．アーキテクチャ上は入出力命令ではなく移送命令であるが，Fortran プログラムからは入出力装置にみえるように制御プログラムでシミュレートしている．

スーパコンピュータは高速のベクトル演算機能によって高速性を達成している．さきにも述べたように，ベクトル化率が十分高くなければプログラムの高速実行は期待できない．ベクトル化率が低ければチューニングによってプログラムを修正することも必要となる．チューニングのためには，プログラムを解析して

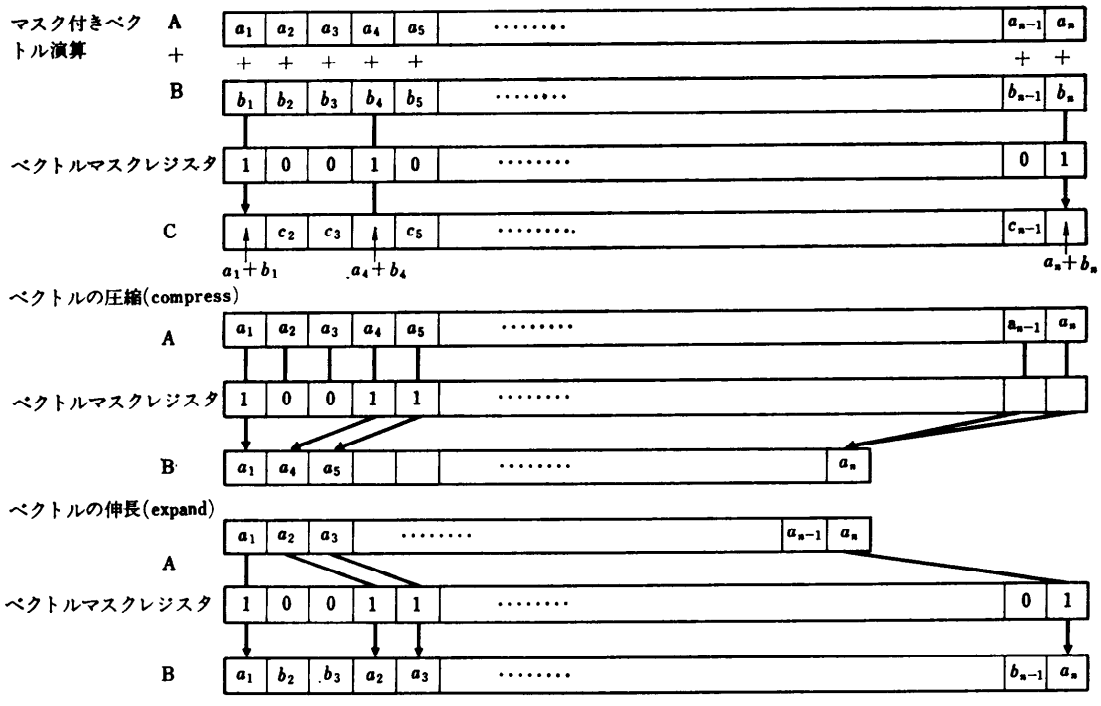


図-5 マスク付きベクトル演算例

実行頻度が高く，かつ，ベクトル化率が低い部分，チューニング部分，を調べる必要がある．このような解析/チューニングをサポートする機能として，命令実行数カウンタ，ベクトル命令数カウンタ，ベクトルエレメント数カウンタなどのカウンタを備え，プロセッサの動作プロファイルを収集するようにした．これらのカウンタは，常時動作しており，命令によって読み出すことができる．

3. ハードウェア構成・実現技術

ここまで，SX システムのソフトウェアにみえる機能について，主な特徴を述べた．以下では，このアーキテクチャを実現しているハードウェア構成技術について，特に，プロセッサの高速化を中心に主な項目を列挙する．

● 命令およびスカラデータ用に，キャッシュメモリを備えているが，このほかに小容量の命令バッファを用意して，命令の取りだし／供給の高速化を図った．この命令バッファは，分岐履歴バッファも合わせて備え，分岐予測機構を取り入れて，分岐命令を高速化した．

● スカラ演算は，ベクトル演算と同様，すべてパイプライン化した．パイプライン演算器が効率よく動作するように，レジスタの競合がないかぎり，命令を連続して発行するように制御して，命令の並列実行度を高めた．これによって，スカラ命令同士，あるいはスカラとベクトル命令も並行して実行できる．

● ベクトル演算の高速実行のために，シフト/論理/乗算/加算の 4 種のベクトル演算パイプラインを用意した．さらに，各演算パイプラインは最大 4 本のパイプライン構成とし，合計 16 本のパイプラインが同時並行して動作するように構成した．

● ベクトル演算パイプラインに対応して，ベクトルデータをメモリとの間で，高速転送するために，毎クロック，最大 8 語 (64 バイト) のデータ転送ができる太いデータ転送路を設けた．さらに，ベクトルデータの十分な転送スループットを確保するために，メモリは最大 512 個の独立したバンク構成とし，512 ウエイの高インタリーブ方式を採用した．

4. おわりに

以上，SX システムのアーキテクチャにつき，特に科学技術計算の高速化を目指して，アーキテクチャ上工夫した点を中心に説明した．

ここで述べたこと以外にも，ハードウェアの基本技術である LSI/実装技術，コンパイラを始めとするソフトウェア技術などが，実際のシステムで高速化を達成するためにはきわめて重要な技術である．SX システムの命令セットアーキテクチャは，これらさまざまな技術の動向，将来の発展性など，総合的な観点から検討を加えて，設定した．しかし，システムは使い込まれて，始めて進歩するものである．今後も，システムの使用経験を反映させて，より良い命令セットアーキテクチャへと，発展させたい．

参 考 文 献

1) 古勝，渡辺，近藤：最大性能 1.3 GFLOPS，マシンサイクル 6 ns のスーパーコンピュータ SX システム，日経エレクトロニクス，1984. 11. 19, pp. 237-272.
2) Watanabe, T. et al.: THE SUPERCOMPUTER SX SYSTEM: AN OVERVIEW, Proc. of Second International Conference on Supercomputing, pp. 50-56 (May. 1987).
3) Jippo, A. et al.: THE SUPERCOMPUTER SX SYSTEM: HARDWARE, ibid., pp. 57-64.
4) Yamamoto, M. et al.: THE SUPERCOMPUTER SX SYSTEM: OPERATING SYSTEM, ibid., pp. 65-71.
5) Tsukakoshi, M. et al.: THE SUPERCOMPUTER SX SYSTEM: FORTRAN 77/SX AND SUPPORT TOOLS, ibid., pp. 72-79.
6) Hanamura, M. et al.: VECTORIZING THE FFTS on SUPERCOMPUTER SX SYSTEM, ibid., pp. 404-412.
7) Watanabe, T.: Architecture and Performance of NEC Supercomputer SX system, Parallel Computing, Vol. 5, pp. 247-255 (1987).
8) Watanabe, T.: Design Concept for Highspeed Vector and Scalar Processing: Architecture of the NEC Supercomputer SX System, Proc. of IEEE International Conference on Computer Design, pp. 38-41 (Oct. 1986).

(昭和 63 年 9 月 8 日受付)