IBM RS/6000とPOWERアーキテクチャの10年間 <前編>

日本アイ・ビー・エム（株）東京基礎研究所
寒川　光　samukawa@jp.ibm.com
川瀬　桂　kawase@jp.ibm.com

1975年10月に約20名の研究者がIBMのワトソン研の801号館に集まり，高級言語で記述されたプログラムにとって，既存のシステムよりも価格性能比の優れたシステムを目指して，ミニコンピュータ，コンパイラ，制御プログラムの設計に着手した[1]．801プロジェクトである．当時のIBMの主力製品はSystem/370（S/370）で，1964年に発売されたS/360の発展である．計算機アーキテクチャという用語はS/360の開発において最初に用いられた[1]．S/360以前の計算機では，計算機ごとに命令セットやデータ形式を用意したため，これらは計算機の付属物と考えられた．S/360の設計において，これらにハードウェアから独立した定義を与えることで，小型から大型までの幅広い機種で共通のソフトウェアが稼働し，また新たに開発する計算機へのソフトウェアの継承を容易にした．この時代のハードウェア技術は，50から60個のICモジュールがプリント基盤上にハンダ付けされ，20枚のカードがボードに差し込まれ，大型機でも6,000回路程度でプロセッサ全体が実現されていた．このような実装技術を前提とすると，アーキテクチャを支えるのは，マイクロプログラムということになる．

マイクロプログラム方式で，命令パイプラインが100%の効率で稼働するとMIPS値がマシンサイクルの逆数に達するが，一般に命令やデータの取出しの遅れ，実行時間の長い命令，分岐命令による遅れ，命令間のインターロックなどのため，実効性能はその1/3から1/4にまで低下する．801プロジェクトは，このようなS/370の命令セットとマイクロプログラム制御の反省に基づいて，命令長を固定化し，命令数，命令の形式，アドレスモードを少なくすることで，命令の実行制御をワイヤードロジック方式で可能な範囲に収め，1命令を1クロックで実行する801ミニコンピュータを完成させた（1980年）．単精度の浮動小数点加算も，ビットオペレーションを駆使して20クロックで処理できた．801はオフィス製品事業部に，IBMパーソナル・コンピュータRT（RISC Technology）として1986年に登場する製品の基礎となった．

■1980年代後半のRISC市場　1980年代になると，IBMのいくつかの開発プロジェクトが，801をミニコンピュータとして使用するようになった．たとえば，S/370の汎用大型計算機3090のI/Oプロセッサや，小型計算機9370のプロセッサに用いられた．801プロジェクトの初代のマネージャであったJ. バーンバウムは，1982年にHewlett-packard社に移り研究チームを発足し，1986年2月にHP-3000を登場させた．最初の製品の作動周波数は，市販のチップを使用した930型が8MHz，カスタムLSIを用いた950型が12.5MHzであり，浮動小数点演算コプロセッサはオプションだった．同時期にカリフォルニア大学（RISC-I, II）とスタンフォード大学でもRISCの研究が進められ，SPARCとMIPSに発展した．

■1990年代と第2世代RISC　1985年に801の開発メンバは再びアーキテクチャの検討を開始した．ここでの目標は，浮動小数点演算パイプラインを含めたILP（Instruction Level Parallelism）に向けられた[5]．このプロジェクトは，3つの半自律的なプロセッサからなるAMERICAプロセッサを開発した．801が特定のアプリケーションに対して，1クロックで1命令の処理速度を実現したのに対し，AMERICAはより一般的な浮動小数点演算を含むアプリケーションに対して，1クロックに数命令の処理速度を実現した．CMOSは当時のS/370で使われていたバイポーラ型の論理演算回路に比べると，スイッチングタイムでは劣るものの，集積度ではるかに大規模だったため，エンジニアリングワークステーションのようにコスト性能比が重要な市場で急速に

プロセッサ，機種，周波数（発表年）	TPP	Reak
POWER1 530 25MHz（1990）	42	50
RSC 220 33MHz（1992）	14	66
POWER1 580 62.5MHz（1992）	104	125
POWER2 590 66MHz（1993）	236	264
P2SC 595 135MHz（1996）	440	540
P2SC M397 160MHz（1997）	532	640
ppc604e F50 332MHz（1998）	317	664
Power3 Model 260 200MHz（1998）	642	800
Power3-II Model 270 375MHz（2000）	1236	1500
Power3 Model 260 200MHz×1	639	800
Power3 Model 260 200MHz×2	1168	1600
Power3 High 220MHz×1	684	880
Power3 High 220MHz×2	1247	1760
Power3 High 220MHz×4	2153	3520
Power3 High 220MHz×8	3516	7040

表-1　Linpack TPP性能値（Mflop/s）

	POWER1		POWER2	
	論理	メモリ	論理	メモリ
ICU	200	550	547	2,277
FXU	250	250	583	848
FPU	360	60	1,001	315
DCU×4	700	3,800	1,117	16,600

ICU: Instruction Cache Unit　FXU: Fixed Point Unit
FPU: Floating Point Unit　DCU: Data Cache Unit

表-2　チップセットの回路数（単位Kトランジスタ）

成長した．この現象は1990年代のダウンサイジングを暗示していた．801とAMERICAの父と呼ばれるJ. コックはこの点を，1988年に行われたチューリング賞受賞記念講演で「回路やメモリの密度向上（たとえばトランジスタを100万個のせたチップや，4メガビットのチップなど）のおかげで，ほんの少数のCMOSチップで，多くの重要な問題に対してベクトル方式の計算機の性能に匹敵するような，強力な科学計算機を作ることができる」と述べた[2]．このAMERICAプロセッサがテキサス州のオースチン研に渡り，1990年にRISC System/6000（RS/6000）として登場するのである．2年後にDECがAlpha AXPアーキテクチャとAlpha 21064プロセッサを発表し，第2世代の主要なRISC製品が出揃った．POWERの最初の製品は専用のチップを用いて20から30MHzで稼働した．

■RS/6000ファミリー　RS/6000は製品ファミリーの名称で，POWER（Performance Optimized with Enhanced RISC）アーキテクチャをサポートする．その後このアーキテクチャはPowerPC（Power Performance Chip，1991年秋発表）に発展し，多重プロセッシングと64ビットアドレスに対応した．POWERアーキテクチャをサポートする初期のプロセッサは複数のチップで構成され，最初のプロセッサもアーキテクチャ同様POWERと呼ばれたが，本稿では曖昧さを避けるため，POWER1と呼ぶことにする．

1993年9月にPOWER2が発表されたが，これも複数のチップで実装された．本格的な1チップによるサポートは1996年10月発表のP2SCからである．一方PowerPCは初めから1チップでサポートされ，プロセッサとしては601，603，604などが登場した．Power3は1997年に発表され，これによりPOWER2の系列とPowerPCの系列が統合された．現在は2001～2002年に出荷される製品群に搭載されるPower4チップが開発中である．

■Linpack性能値　**表-1**に代表機種の，次数1,000の連立1次方程式を解くLinpackベンチマークの性能値を掲げた．ここではプログラムの改変が許されているToward Peak Performance（TPP）を示す．改変は元のプログラムをブロック化し，SMP（Symmetric Multiprocessing）用にはさらに並列化している．浮動小数点演算量は$2n^3/3$なので，1秒間で解ければ667Mflop/sになる．

最初のPOWER1のデスクサイド型の530型では42Mflop/sであるが，最新のPower3のデスクサイド型である270型では1236Mflop/sなので，10年間で性能は30倍になった．

多重プロセッシングはPowerPCとPower3でサポートされ，表の2重線の下の2つはPower3の2ウェイおよび8ウェイの性能値である．

RS/6000の製品ファミリーには，RS/6000のプロセッサをノードとして，これらを相互結合網で接続した分散メモリ型並列計算機が存在する．最初の発表は1993年2月で，Scalable POWERparallel 1（SP1）と呼ばれ，POWER2をノードプロセッサとした時代はSP2と呼ばれたが，現在はRS/6000SPと呼ばれている．

POWER1で最大の論理回路数を搭載したチップは浮動小数点ユニットで，トランジスタ数に換算して36万である（**表-2**）．Power4チップは1億7千万トランジスタに達する[4]．先に引用したコックの講演は「科学計算プロセッサの性能追求」と題するもので，この中で博士は1988年の時点で，「過去25年間に単一プロセッサのコンピュータの性能は100倍も向上した．一方，コストは

1万分の1になった．来る25年間も同様のコストの改善が見られるだろう」と述べている[2]．プロセッサは速くなるより，より急速に安くなるのであるが，これはこの10年間について考えると，CMOSの高集積度によるところが大きい．論理回路の集積度をアプリケーションの性能に伝えるには，アーキテクチャ，実装技術，コンパイラ技術，プログラミング技術と，複数の技術の連携が必要である．特にRISC型の計算機においては，プロセッサの設計とコンパイラ技術の連携は重要である．本稿では2回にわたって，RS/6000とPOWERアーキテクチャの10年間の発展を，アーキテクチャ，ハードウェア，コンパイラの関係を俯瞰する視点で振り返ってみたい．本号で，POWER1，POWER2とP2SCについて述べ，次号で，PowerPC，Power3，Power4について述べる．

■ 計算機アーキテクチャ ■

POWERアーキテクチャの特徴は，(1) 命令を1語長(32ビット) に揃える (命令数は200弱)，(2) 命令を固定小数点系，浮動小数点系，分岐系の3種類に大別する，(3) 演算はレジスタ同士，(4) 十分な数のオペランド，(5) 豊富なレジスタ (32個の浮動小数点レジスタfp0からfp31と32個の汎用レジスタ) などである．ここでは本稿で使用する，浮動小数点データを移動する命令と算術演算命令を紹介する．

データ移動命令は有効アドレスで指定されたオペランド (倍精度) を浮動小数点レジスタにロード／ストアする．更新型の命令 (lfdu, lfdux) では，有効アドレスを (ra) + (d) または (ra) + (rb) などとして求めた後，汎用レジスタの内容 (ra) を更新する☆1．ストア命令は，ロード命令の擬似コードの“l”を“st”に置き換えたものである (lfd→stfdなどの4種類)．

浮動小数点算術演算命令には四則演算命令と乗加算命令 (fma命令など) がある．乗加算命令D＝A・B+Cは乗算と加算を融合 (MAF: Multiply Add Fused) している．融合の考え方を3桁の10進数の計算123.×321.+12300.で示すと**図-1**のようになる (乗数が3桁なので3項の和を求めれば積A・Bが得られるが，この加算にC項を含めて加える)．いずれの命令も演算結果のために独立したレジスタがあてがわれ，入力オペランドに出力を上書きすることはない．算術演算は倍精度のみで，単精度演算は，単精度ロード命令，ストア命令，単精度への変換命令によって行われる．

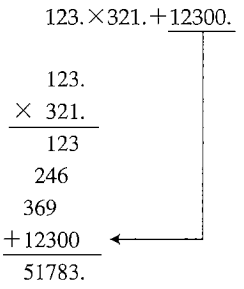

図-1　乗算と加算の融合

条件レジスタは各4ビットで8つ定義されている．分岐命令は条件コードによる分岐を採用しているので，コンパイラが分岐条件を生成する命令と分岐命令を引き離しやすくしている．

命令列を2次元座標変換プログラムを例に考察する (**図-2**)．このコードはソフトウェアパイプライニングが適用されており，1回目のループ反復の前半はループに入る前に (fp6にy (1) がロードされ，fp8にx (1) * a11が，fp7にx (1) * a21が作られ)，最後の反復の後半はループ外で処理される．a11,･･･, b2の6変数はこのループに入る前に浮動小数点レジスタfp0からfp5に置かれ，ループ内での記憶域アクセスはx (i)，y (i) のロード (lfdu) と，xx (i)，yy (i) へのストア (stfdu) の4回のみとなる．なお「記憶域」は記憶域オペランドを指し，物理的にはキャッシュかメモリかは特定しない．

このプログラムをS/370のような少数のレジスタと2オペランド算術演算命令で構成されるアーキテクチャの計算機で実行すると，記憶域アクセス回数がはるかに増える．S/370ではレジスタは少数のアキュムレータとしてしか提供されていないからである．これに対し，POWERではレジスタファイルとして存在し，また算術演算命令も入力オペランドを破壊しない．レジスタファイルの目的は，いったんレジスタにロードしたデータを，繰り返し演算に使用することにある．

■ POWER1 ■

浮動小数点算術演算は1つの乗加算器が行う．この処理は，Boothのリコーディング後の29項をWallace木☆2によって2項にまで加え合わされたところでCを含める．この3項を1段のCSA (Carry Save Adder) によって2項にしてから，CLA (Carry Lookahead Adder) で1項にする．POWER1は乗算も加算もこの乗加算器によって行い，除算はこの乗加算器を用いるマイクロコード

☆1 ra, rbは汎用レジスタの内容，dはアドレス変位である．有効アドレスの上位4ビットで16個のセグメントレジスタの1つを選択し，その内容 (24ビット) を修飾することで仮想アドレス (52ビット) が生成される．
☆2 CSAを木状に並べて，1時点で加えられるビットを可能なかぎり並列に加算することで，$O(\log_{3/2} n)$ 時間で乗算する．

```
do i=1,n
  xx(i)=b1+a11*x(i)+a12*y(i)
  yy(i)=b2+a21*x(i)+a22*y(i)
enddo

fma  fp9=fp8+fp6*fp1 ! y(i)*a22
fma  fp10=fp7+fp0*fp6 ! a12*y(i)
lfdu  fp6,y(i+1)
fma  fp7=fp2+fp4*fp11 ! a11*x(i+1)
fma  fp8=fp3+fp11*fp5 ! x(i+1)*a21
lfdu  fp11,x(i+2)
stfdu  fp10,xx(i)
stfdu  fp9,yy(i)
bc  -32
```

(I) FXデコード
(II) FX実行
(III) キャッシュ
(1) プリデコード
(2) リネーミング
(3) FPデコード
(4) FP実行1
(5) FP実行2
(6) FP書込み
(7) FPSQ

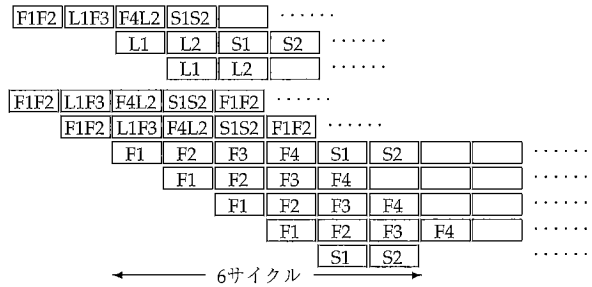

図-2　2次元座標変換プログラムのダイアグラム

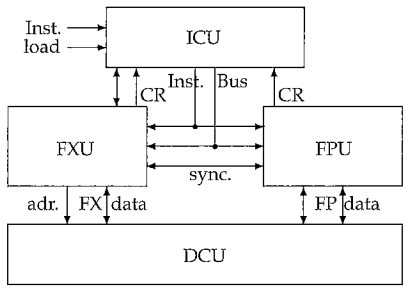

図-3　POWER1 の機能ユニット

がニュートン法で行う．

乗算と加算を融合すると，乗算に続く加算は見かけ上処理時間ゼロ（乗算が2クロック，乗加算も2クロック）と速くなるが，乗加算器の中で積がA・Bという値でビット構成されるタイミングがなくなるので，乗算と加算に分離した場合とビット単位に同一の結果を得るように，数値を丸める機会もなくなる．精度的にはA・Bを4倍精度の被加数としたまま，これに倍精度の数Cを加えてから丸めた場合と同じになるが，IEEEの規格には合致しなくなる☆3．乗加算パイプラインは3段であるが，フォワーディング経路により，多くの場合，実質的に2段の振舞いとなる．したがって依存性のない連続する乗加算命令は，先行する命令が後段に入ると，次の命令の前段が始められるので，見かけ上1サイクルで1命令が処理される．これを2クロックレイテンシ，1クロックスループットという[5]．

POWER1では，機能ユニットはFXU，FPU，分岐ユニットを持つICUとしてそれぞれが1チップで構成された（表-2）．ICUは命令を解読（デコード）し，FXUとFPUの担当分を分離し，それらをFXUとFPUに送り込む．分岐命令はICU内部のブランチユニットで処理される．また命令キャッシュとそのディレクトリやTLB（Translation Lookaside Buffer）☆4を持つ．FXUは固定小数点演算とロード／ストア命令を実行する．またデータキャッシュのディレクトリやTLBを持つ．FPUはレジスタファイルを持ち，浮動小数点算術演算を実行する．また浮動小数点ストア命令の丸め処理を行う．これらの機能ユニットの関係を**図-3**に示した．データ，命令，アドレスを送る経路は，矢印1本が32ビット幅である．CRは命令の実行結果をCR（コンディション・レジスタ）に返すため，また“sync.”は後述するFXUとFPUの同期のために使用される．2次元座標変換プログラムを例に，機能ユニットが行う命令パイプライン処理を図-2のダイアグラムに沿って説明する☆5．ここではlfdu命令を“L”，fma命令を“F”，stfduを“S”で記す．ICUは分岐系の命令を処理するとともに，命令列（F1, F2, L1, F3, F4, L2, S1, S2）を4命令／サイクルの速度でFXUとFPUに送り込む．

(I) FXUデコードの後，(II) 実行ステージでEAを生成し，セグメントレジスタによる修飾，FXU内部に置かれたキャッシュディレクトリの検索を行い，オペランドがキャッシュに存在するかどうか調べられる．FPUはFXUの実行ステージと同時に (2) リネーミングのステージを実行する．ここでロード命令は，アーキテクチャ上のレジスタに対応する物理レジスタを割り当てられ，ストア命令は，その時点でのアーキテクチャ上のレジスタに対応する物理レジスタを教えられる．(III) キャッシュのステージで，オペランドがキャッシュに存在すればロード，不在なら処理はラッチされる．(3) FPデコードのステージでレジスタファイルが読まれ，同時に依存性が検出される．(4) (5) 実行の2つのステージが乗加算器による実行で，(6) FP書込みのステージで結果をレジスタファイルに書き込む．仮想記憶方式はキャッシュと実装上の相性がよい．特にアドレス変換でTLBヒットした場合は，(II) の有効アドレスからキ

☆3 これにはコンパイルオプションにより乗加算を乗算と加算に分割することで対処できる．
☆4 TLBはページ表のキャッシュと考えればよい．POWER1での構成は16セット×2連想性である．
☆5 命令パイプライン（instruction pipeline）は命令の実行部分に先立って，命令のフェッチ，解読などを，先行する命令の実行とオーバーラップして処理する方法を指す．

ャッシュラインを見つけ出す一連の処理は半クロックで行われる[3]．TLBをミスした場合は，仮想ページアドレスを逆ページ表で求める．

この例ではコンパイラの最適化機能によって，直前の命令の出力を入力として使用することをなくしている．ただし浮動小数点レジスタに着目すると，L1がF1やF2の実行が完了する前にfp6にy(i+1)をロードしているので，図のダイアグラムどおりでは正しく計算されそうにない．これはレジスタリネーミング機能により解決される．この機能はアーキテクチャ上の浮動小数点レジスタ(32個)を，それよりも数の多い物理レジスタ(38個)に動的に対応させる．対応関係は新たなデータがレジスタにロードされるときに更新される．これによりストア命令や演算命令の直後で同じレジスタにロードする命令が，これらの先行する命令の完了を待つ必要がなくなる．

浮動小数点ロード命令はFXUによって実行され，そのデータを使用する演算命令はFPUによって実行されるので，FXUとFPUは緊密に連携して動かなければならない．このためロード命令がFXUで実行されるサイクルとFPUのリネーミングのサイクルを一致させるなど，細かな制御を行っている．科学計算には少量のデータを順次アクセスしながら多量の浮動小数点演算を繰り返すループが多く用いられる．このようなループではユニット間に必要とされるやりとりや同期化が少なくてすむので，POWER1の設計には適している．

データキャッシュはストアイン方式で，置換方式はLRUアルゴリズム，ラインサイズは128Bで，セット数が128で，4重のセット連想写像方式を用いている．容量は128×128×4＝64KBである．64KBの容量は16KBずつ4つのチップに分けて搭載される．1ラインは32B(1ラインの4分の1)単位で4重のインターリーブ方式によって，キャッシュとメモリ間を移動する☆6, [5]．

メモリとキャッシュ間のデータ幅は毎周期128ビット，キャッシュとFPU間のデータ移動性能は毎周期64ビット(lfdまたはstfd命令が毎周期1命令)である．データ経路を図-4に示した．FPUとキャッシュ間のデータ幅は演算・移動性能比1を満たし，この性能の2倍をキャッシュとメモリ間で確保している☆7．これまで述べたモデルはすべてPOWERアーキテクチャを複数のチップで実装したものであるが，RSC(RISC Single Chip)と呼ばれる単一のチップでPOWERをサポートする220型も提供された(1992年1月発表)．この型では統合キャッシュ(容量8KB)を用い，演算パイプラインの機能も簡略化し，レジスタリネーミングも備えないなどの方法により，回路数を減らして1チップ化した(FPUは半分のビット数のWallace木を2回実行することで倍精度演算を行う)．TPP性能値は最大性能の21%である(表-1)．

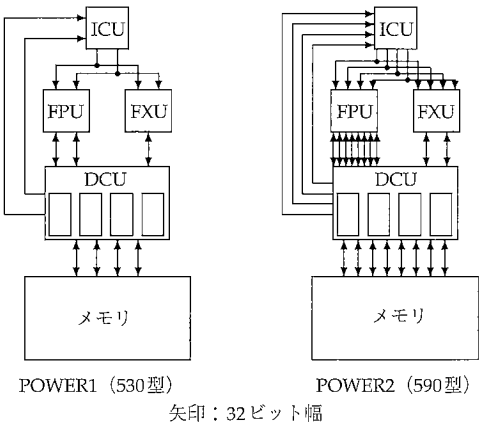

図-4　POWER1とPOWER2のデータ経路

■ POWER2とP2SC ■

POWER2はPOWER1のFPU，FXUの機能を2つずつ備えたFPU，FXUを持つ．最初のモデル(590型など)は1993年9月に発表された[7]．この時代のCMOSの集積度は，POWER1の時代の約2倍に向上した(表-2)．当時のRISCプロセッサの設計方針は，パイプラインを細分化して高周波数化するSpeed Demons派と，1クロック内に処理する仕事を増やすBrainiacs派に分かれていたが，POWER2は典型的なBrainiacs派である．つまり，FXUとFPUの同期をチップとチップの間に残したので，クロック周期はあまり短縮できなかったが，1周期で処理可能な命令数を約2倍とした(1クロックで最大6命令：分岐，条件レジスタ，2つの固定小数点，2つの浮動小数点命令)．デスクトップ型同士で比べると，POWER1の最終版は62.5MHzの580型，最初のPOWER2は66.5MHzの590型で，周波数は6%ほどしか違わないが，最大性能は125Mflop/sから266Mflop/sと2倍以上になった．

POWER2の後半のモデルは1996年10月に発表されたP2SC(POWER2 Super Chip)で，0.29ミクロンの設計ルールを用いて単一のチップで構成された(595型)．複数のチップに分けて搭載されていた機能ユニットが，単一のチップに集められた．図-3の機能ユニット間の矢印がチップ外に出ることなく伝えられるので，動作周

☆6 メモリを4バンクに分け，1ラインを32Bずつ4つのバンクに置くことで4バンクに並列にアクセスする方法を，4重のインターリーブ方式という．
☆7 "演算性能を乗加算(2flops)を計算する速度，データ移動性能を倍語(64ビット)のデータを移動する速度としたとき，前者を後者で割った商"を演算・移動性能比と定義する．

波数は135MHzに上がった．最終型の397型（1997年10月）は0.25ミクロンの設計ルールで160MHzに達した．POWER2では4倍語ロード／ストア命令，浮動小数点数→整数変換命令，平方根命令が追加された．4倍語ロード命令（lfqなど）は，連続する16バイトのデータを連続する2つのレジスタにロードする．

2重化されたFPU，FXUの機能を表／裏と呼ぶことにする．命令の表／裏への振分けは，コンパイル時や命令のディスパッチ時には決めず，これを実行の直前ぎりぎりになってから動的に行う．2つの命令は2つの乗加算器によってリネーミングされた後も，1つの命令キューに置かれ，実行の直前に表／裏が選択される．命令キューには6命令が蓄えられ，表／裏の選択時にこれらの命令のオペランドが分析され，依存性のない命令同士を表／裏に振り分ける．また表／裏のパイプラインの出力は，相互にフィードバック経路を備える（表の計算結果を裏が即座に使用できるようにフォワード経路を持つ）[7]．この結果プログラムから見た性能の振舞いは，パイプラインの段数が4で，クロック周期が半分になったように見える．

図-4に初期のPOWER2機種のデータ経路を示した[7]．デスクサイド型の590型のFPUはPOWER機種のFPUと比較すると，クロック周期あたり2倍の演算性能を持っているが，FPUとキャッシュ間のデータ幅は（lfq命令を使用した場合に限るが）4倍に強化されているので，演算・移動性能比は0.5になる．メモリとキャッシュ間も同じデータ幅を確保している．

データキャッシュの構成は，1ラインサイズが256B，セット数が256，4重のセット連想写像方式を用いているので，容量はPOWER1の4倍の256KBである．

P2SC（595型）では演算性能が590型の約2倍に強化されたが，データ移動性能は強化されなかったので，演算・移動性能比は1に戻った．クロック周期とメモリアクセス時間の開きが大きくなったので，2段キャッシュが採用された．1次キャッシュはプロセッサチップ内に収められ，容量は小さくなった．2次キャッシュは外付けで，1から16MBの容量が選択可能である．

■ コンパイラとブロック化 ■

801の目標は1命令を1クロックで実行するワイヤードロジックのミニコンピュータであった．これを本稿では狭義のRISCの定義と考えよう．RISCという用語は，801の開発と同時期にカリフォルニア大学で研究していたパターソン教授が使用したもので，Reduced Instruction Set Computerの略である．しかしコックらは，「801とそれに続くアーキテクチャでは，RISCを“縮小命令セット”と解釈するよりは，むしろ，“記憶階層の最速の部分を有効に使用するために注意深く選択された命令セット”である．演算命令はレジスタ対記憶域オペランド形式は使用せず，レジスタ対レジスタ形式だけを使用する．ただしレジスタを増やす必要があり，それ以外はS/370によく似ていた．このため，もしこのアーキテクチャに命名するなら“Reduced”ではなく“Regular”が適当」と言った[1]．POWERアーキテクチャは初めから文字列操作や浮動小数点数除算のように実行に複数サイクルを要する命令を持っており，狭義のRISCからは外れる．801の思想の中でPOWER1が引き継いだ最も重要な点は“コンパイラが演算器の内部での実行タイミングからデータ移動まで，幅広い最適化をすること”であり，これを広義のRISCの定義と考えよう．

プロセッサが高周波数化されても，メモリのアクセス時間は短縮が困難である．プロセッサの演算性能は年率60%で高速化されたが，メモリのアクセス時間の向上は年率10%程度であった．作動周波数を高めても，データが供給されなければ演算器は待たされる．POWER1はスーパースカラにより，1クロック乗加算を実現したが，この演算性能が発揮されるのは1クロックに1項以内の倍精度データしか記憶域参照しないループにおいてだけであり，それより多くのデータを要求するループでは，データ供給が隘路となる．両者のギャップを埋めるのがレジスタファイルを頂点とする記憶階層である．そしてこの階層を有効に働かせるためのブロック化は，コンパイラの重要な仕事である．

行列行列積$c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}$, $i = 1: n$, $j = 1: n$を例にブロック化の考えを述べる．この式に忠実な計算順序を用いると，乗加算1回につきa_{ik}とb_{kj}を1項ずつ参照するので，乗加算命令はロード命令がデータを供給するのを待つため，最大性能のたかだか半分しか発揮できない．c_{ij}を求めるのに2n項，計算全体では$2n^3$項のデータ移動が必要である．

ここで**図-5**のように，外側の2つのループにアンローリングを適用すると，4回の乗加算に対し，4項の参照になり，パイプラインは休むことなく計算できる．これがレジスタを対象としたブロック化である．そしてc_{ij}以外の項はキャッシュに存在するようにする．これにはキャッシュを対象としたブロック化を行う．定式化は，小文字（行列要素）を大文字（小行列）に書き換えることで得られる．行列A，B，Cをあらかじめm×mの小行列A_{IJ}などに分割して考える．ブロック化アルゴリズムは，$C_{IJ} = \sum_{k=1}^{N} A_{IK}B_{KJ}$, $I = 1: N$, $J = 1: N$で記述できる．ただし$N = \lceil n/m \rceil$である．ここでm×mの小行

列3つがキャッシュに収まりきるようなmを考える．するとCIJを1つ計算するのに2N個の小行列をメモリからキャッシュに移動すれば計算全体が完了するので，メモリとキャッシュ間のデータ移動は$2N^3$個となる．小行列のサイズはm^2なので$2n^3/m$項と，式に忠実な順序のm分の1になることが分かる[6]．

最内側ループを計算中の計算機内部のデータの動きを図-5に示した．c_{ij}の4項はレジスタに足し込めるので，レジスタファイルから参照され，レジスタファイルへ出力される．a_{ik}とb_{kj}はほとんどの場合キャッシュに存在し，ときどきキャッシュミスを発生しメモリ参照される．

コンパイラは，レジスタブロック化に外側ループアンローリング，キャッシュブロック化に外側ループストリップマイニングを用いる．ここではコンパイラはネストしたループプログラムでのデータの流れを分析して，いったんメモリから引き上げられたデータを記憶階層に残すことのできる計算順序（ループ構成）を作り出すことで，計算機内部のデータ移動を最適化するのである．もちろんコンパイラの仕事はこれだけではなく，演算器内部での実行のタイミングを考慮した最適化も行う．2次元座標変換プログラムでソフトウェアパイプライニングを適用した例を示したが，これを適用しないと，6サイクルで実行されるループは7サイクルに増加する．キャッシュブロック化のような大きなデータの流れに対する最適化技術は主にベクトル化の技術から，細かなコード移動のような技術はマイクロプログラムの技術から発展した．

行列行列積以外のアルゴリズムでも，ブロック化によって，メモリ参照回数をブロックサイズmで割ることのできるアルゴリズムは少なくない．Linpackベンチマークのカーネルである LU分解のアルゴリズム: A→LUは，依存性が複雑な形で埋め込まれているので，コンパイラの自動的なブロック化は困難であるが，行列行列積同様，小行列単位の定式化は可能である．またブロック化されたLU分解アルゴリズムのカーネルは行列行列積になる．そこでブロック化はプログラミングにより行い，キャッシュ容量やパイプラインの段数などの計算機の個性に依存する部分はプロセッサごとの最適化を行う．たとえばPOWER1に対しては，図-5の形のカーネルを用いるが，POWER2ではlfq命令を使用するためと，パイプラインの2重化に対応して，内側ループにもアンローリングを用いる．POWER2のLinpackのTPPは236Mflop/sと，最大性能の90%に近い性能に達する．これはデータの供給能力に余裕があるためで，P2SCでは相対的に余裕が減ったため，最大性能の80%近くに低下する（表-1）．ときどき発生するキャッシュミスにはプリフェッチが適用できれば理想的である．行列ベクトル積の例を述べる．この問題では乗加算1命令につき，メモリから演算器に1倍語を供給しなくてはならないので，通常のプログラミングでは定期的なキャッシュミスのためにFPUが待たされる．590型では，FXUの性能に余裕があるので，キャッシュミスを引き起こすデータに対して，先行してロード命令を発行することで，キャッシュミスによる遅延を軽減することができる．この操作をプログラミングで実現する方法をアルゴリズム・プリフェッチという．プリフェッチなしの行列ベクトル積の性能は，行列行列積の半分程度の130Mflop/sであるが，プリフェッチを適用することで180Mflop/sに向上する[6]．

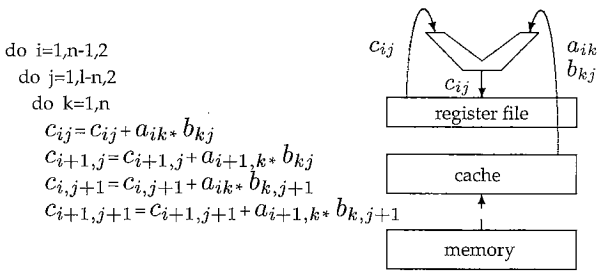

図-5　ブロック化と計算機内部でのデータの動き

■ まとめ ■

論理回路の高密度化は演算性能を非常に高め，64ビットの浮動小数点を1クロックで処理するまでになったが，相対的にデータの供給が厳しくなった．特にメモリのアクセス時間が取り残される形になったので，記憶階層を導入してその上位の階層では短いアクセス時間を実現するようにした．この方式ではレジスタやキャッシュを有効に利用できるかどうかが，実質的な性能を大きく左右する．

参考文献

1) Evolution of Information Technology 1957-1999, IBM J. of Res. Develop., Vol.44, No.1/2 (2000).
2) Cocke, J.（前川　守訳）: 1987年度ACMチューリング賞受賞記念講演，科学計算プロセッサの性能追及, bit, Vol.20, No.11, 共立出版 (1988).
3) Weiss, S. and Smith, J. E.: PowerPC 解説, POWERからPowerPCへ，インターナショナル・トムソン・パブリッシング・ジャパン (1995).
4) Diefendor, K.: Power4 Focuses on Memory Bandwidth, Microprocessor Report, Vol.13, No.13 (1999).
5) IBM RISC System/6000 Processor, IBM J. of Res. Develop., Vol.34, No.1 (1990).
6) 寒川　光: RISC超高速化プログラミング技法, 共立出版 (1995).
7) POWER2 and PowerPC Architecture and Implementation, IBM J. of Res. Develop., Vol.38, No.5 (1994).

（平成12年12月25日受付）

訂　正

本誌42巻4号（2001年4月号）pp.393-399に掲載されました連載解説「IBM RS/6000とPOWERアーキテクチャの10年間＜前編＞」に誤りがありましたので，以下の通り訂正いたします．

p.395 右段8行目
（誤）（fp6にy (1) がロードされ，fp8にx (1) *a11が，fp7にx (1) *a21が作られ），
（正）（fp6にy (1) がロードされ，fp7にb1+ x (1) *a11が，fp8にb2+x (1) *a21が作られ），

p.395 右段10行目
（誤）a11, ･･･, b2の6変数はこのループに入る前に
（正）a12, a22, b1, b2, a11, a21の6変数はこのループに入る前に

p.396 図-2
（誤）a11*x (i+1)
（正）b1+a11*x (i+1)

（誤）x (i+1) *a21
（正）b2+x (i+1) *a21