Vol. 36 No. 5　　情　報　処　理　　May 1995

解　説

# RISCプロセッサとキャッシュ†

松　原　健　二††

## 1. はじめに

'90年代に入ってからRISCプロセッサの処理性能は飛躍的に向上している．High-endを指向しているRISCアーキテクチャは5種類もあり(**表-1**)，しばらくは群雄割拠の状態が続きそうである．それぞれのアーキテクチャとも，より高い性能を発揮するためにコンパイラ，論理方式，デバイスなどの最新テクノロジーを取り込みながら変化し続けている．RISCが現われた当初は，シンプルな機能を持つ命令セット＝シンプルな論理方式という短絡な考え方も見られたが，今ではメインフレームに匹敵する高度な方式となっている．たとえば複数の命令を並列に実行するスーパスカラ方式，さらに命令の順序を追い越して実行するout of order実行方式などがあげられる．キャッシュについても，どのような構成を採用するかにより処理性能に与える影響が大きくなってきている．特に実用的なプログラムになればなるほど，キャッシュが処理性能に与える影響は大きい．ここでは，次章でキャッシュが処理性能に与える影響について触れる．以降の章では最近のRISCプロセッサに見られるキャッシュの方式について述べる．

## 2. キャッシュと処理性能の関係

処理性能の指標は，評価対象となるプログラムの実行時間T (time) であり，これは実行命令数PC (Program Count)，平均命令実行サイクル数CPI (Cycles Per Instruction)，マシンサイクルMC (Machine Cycle) の積で表される．

$$T = PC \times CPI \times MC$$

Tの値が小さいほど性能は良いことになる．各々の値を決める主な要素は以下のとおり．

PC ：命令セットアーキテクチャ，コンパイラ，プログラムのアルゴリズム
CPI：論理方式，命令セットアーキクチャ
MC ：デバイス，論理方式

これらはPCを小さくしようとするとCPIが増大するといった具合に微妙な関係を持っている．どのようにバランスをとってTを小さくするかは，RISCアーキテクチャによって思想が異なる．PowerPCではCPIを抑え，MCはほどほど(PPC 604は100 MHz[1])であり，AlphaではMCをできるだけ短くし（Alpha 21164は300 MHz[2])，その分CPIは多少犠牲になっている．

キャッシュは通常PCには影響しない．性能に影響を与える要素として，

- アクセスタイム
- ミスペナルティサイクル（＝ミス率×ミス処理サイクル数)

がある．アクセスタイムはマシンサイクルMCに影響し，キャッシュのアクセスが遅いとMCが大きくなり性能が低下する．アクセスタイムは，デバイスの能力によって決定される．高速なデバイスを用い，アクセスの短いキャッシュを作ることは高性能プロセッサにとって欠かすことのできない技術である．ミスペナルティサイクルはキャッシュがミスすることによりメインメモリからデータを転送することで生じるペナルティである．このとき転送される一連のデータをライン，この転送をライン転送と呼ぶ（ラインの代りにブロックと呼ぶこともある)．キャッシュのミスペナルティはCPIに含まれるため，小さければそれだけ性能が向上する．何がミスペナルティを決めるかを一つに絞ることは難しい．たとえば，コンパイラがアドレスを生成する際に，キャッシュのエントリの競合を減らすようにすれば，ミス率を抑えることができる．またキャッシュの容量を

† Cache Organization of RISC Processors by Kenji MATSUBARA (General Purpose Computer Division, Hitachi Ltd.).

†† 日立製作所汎用コンピュータ事業部

減らせば，ミス率は大きくなる．様々な選択の中から，どのようにしてミスペナルティサイクルを小さく抑えるかがアーキテクトの腕の見せ所となっている．

## 3. 階層キャッシュ

最近のRISCプロセッサは，プロセッサチップの上に小容量のキャッシュ（数十KB程度），チップの外に大容量のキャッシュ（数MB程度）という階層構成をとることが多い．前者を1次キャッシュ，後者を2次キャッシュと呼び，それぞれL1，L2と記す．従来の1階層構成キャッシュ（チップ外）との対比を**図-1**に示す．Alpha 21164のようにチップ上に2階層，チップ外に1層とあわせて3階層ものキャッシュを持つプロセッサも出てきている[2]．

階層構成をとるようになった理由を見てみよう．集積度が上がったため，浮動小数点演算回路やキャッシュをプロセッサチップ上に持つことが可能になった．そうはいっても数百KBものキャッシュをチップ上に持つことはできない．そこでチップの外に2次のキャッシュを設け容量不足を補う必要がある．キャッシュをチップの外に置くと，チップ渡りをするためアクセスタイムが長く，マシンサイクルMCを大きくしてしまうのである．

**表-1**

| アーキテクチャ | 主なプロセッサ |
|---|---|
| PowerPC | PPC 620，PPC 604 |
| PA-RISC | PA 8000，PA 7200 |
| MIPS | R 10000，R 4400 |
| Alpha | 21164，21064 A |
| SPARC | UltraSPARC，HyperSPARC |

1階層，2階層構成でのキャッシュミスペナルティは，それぞれ

$$\text{Penalty time } 1 = M_{L1} \times T_{L1}$$

$$\text{Penalty time } 2 = (M_{L1'} - M_{L2}) \times T_{L1'} + M_{L2} \times T_{L2}$$

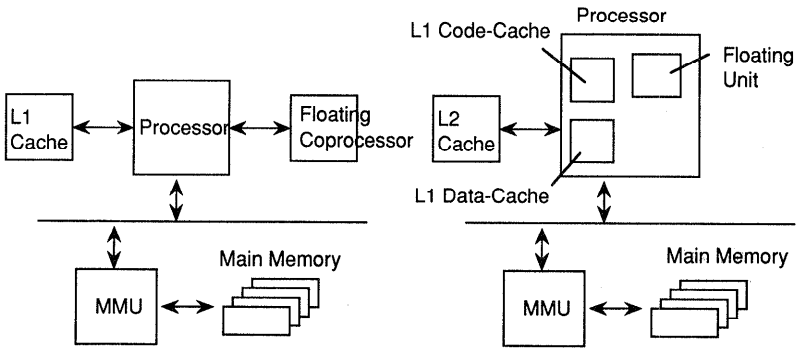
という式で表される．ここで$M_{L1}$，$M_{L1'}$，$M_{L2}$はそれぞれのキャッシュのミス率，$T_{L1}$，$T_{L2}$はライン転送に要する時間，$T_{L1'}$は2次キャッシュから1次キャッシュへの転送時間を示す．2階層の式で第一項は1次キャッシュをミスし，2次キャッシュヒットする場合のペナルティ，第二項は1次/2次キャッシュどちらもミスした場合のペナルティである．チップ外のキャッシュはほぼ同じ構成をとれるので，$M_{L1} \times T_{L1}$と$M_{L2} \times T_{L2}$は同程度となる．したがって2階層では第一項の分だけペナルティが増えることになる．しかしキャッシュをチップ上に置くことでMCが短縮され，このペナルティを帳消しにして性能が向上する．例として1階層構成にて，CPI＝2.5 cyc，$M_{L1}$＝1%，$T_{L1}$＝40 cyc，MC＝12.5 nsec（80 MHz）であるものが2階層構成にて$M_{L1'}$＝9%，$M_{L2}$＝1%，$T_{L1'}$＝4 cyc，$T_{L2}$＝40 cyc，MC＝10 nsec（100 MHz）となったとしよう．1階層から2階層になったことでミスペナルティは

$$(0.09 - 0.01) \times 4 \text{ cyc} = 0.32 \text{ cyc}$$

増加し，2階層構成でのCPIは

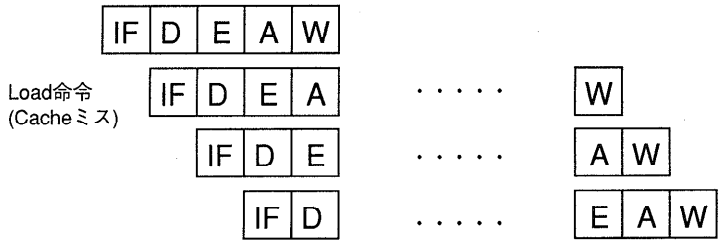
$$2.5 + 0.32 = 2.82 \text{ cyc}$$

となる．実行命令数PCは一定とすると処理時間

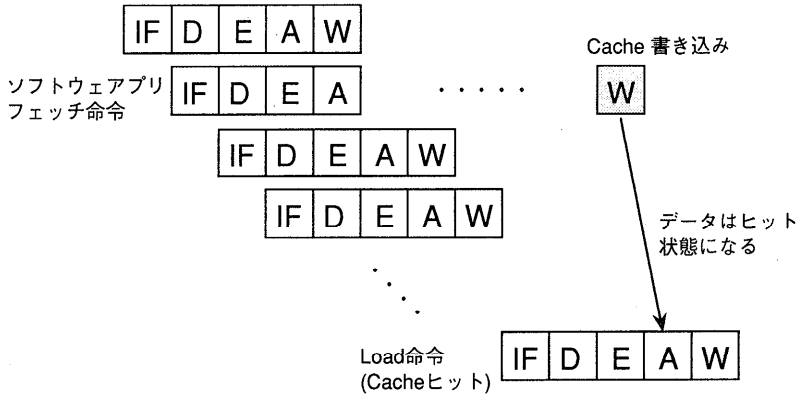



1階層キャッシュ　　2階層キャッシュ

**図-1** キャッシュの階層




(a)　ソフトウェアプリフェッチなし




(b)　ソフトウェアプリフェッチあり

**図-2**　ソフトウェアプリフェッチのパイプライン

の比は

2階層/1階層＝(2.5 cyc×12.5 nsec)
/(2.82 cyc×10 nsec)＝1.11

と求められ，11%性能が向上する．もちろんすべての場合にこれがあてはまるわけではなく，従来どおり1階層構成をとるプロセッサもある．PA 7200 プロセッサは1階層のキャッシュで 140 MHz という高い周波数を実現している[3]．

## 4.　ソフトウェアプリフェッチ

ソフトウェアプリフェッチ[4]は，キャッシュにはないことが分かっているデータを，用いる前にあらかじめライン転送する方式である*．図-2のパイプラインの流れを見てみよう．図-2 (a)，(b) はソフトウェアプリフェッチを行わない/行う場合を示している．図-2 (a) において，アクセスしたデータがキャッシュミスし，ライン転送を行う間，後続命令の実行は中断される．ラインが転送された後に実行が再開されるため，キャッシュミスの処理時間はまるまるペナルティとして現われてしまう．図-2 (b) ではキャッシュがミスした後でもパイプラインは中断せずに実行は継続される．したがってライン転送の時間が後続命令の実行とオーバラップし，ペナルティを隠すことができる．あるデータを使用することがあらかじめ分かっている場合，ライン転送に要するサイクル数だけ先立って，ライン転送を起動する命令（ソフトウェアプリフェッチ命令）を実行すれば，データを使用する時点においてはキャッシュはヒットの状態になる．

ソフトウェアプリフェッチ命令は，通常のロード命令で実現される．RISC アーキテクチャではレジスタ中に値が0固定で書き替えられないものがあり，これをロード命令のターゲットレジスタにすればソフトウェアプリフェッチ命令となる．1命令の追加でキャッシュミスペナルティを0サイクルにすることができるので，ソフトウェアプリフェッチの効果は大きい．しかしあらかじめアドレスが分かるのは，大規模データを扱う科学技術計算のプログラムに限られてしまう．オンライン処理などのビジネス用途プログラムでは，効果

* ハードウェアプリフェッチというものもあり，これはライン転送による効果を表す．データにローカリティがある場合，参照されたデータを含むラインをキャッシュに転送することで，後続の参照がキャッシュヒットとなる．

を得ることは難しい．

図-2を見るとアイデア自体は単純であり，これまで実現されていなかったことが不思議に見えるかもしれない．筆者の意見では，これはコンパイラが有効に使えるソフトウェアプリフェッチ機能というのがあまり解明されていなかったためである．また，VLSIテクノロジーの進歩により，ハードウェア量の増加，マシンサイクル増加等の影響を小さく抑えることができるようになったこともあげられる．

## 5. ノンブロッキングキャッシュ

この方式はソフトウェアプリフェッチに似ており，キャッシュがミスした場合であっても，パイプラインをストールさせることなく，後続命令を実行する．ソフトウェアプリフェッチはメインメモリからキャッシュにラインを転送するだけであるのに対し，ノンブロッキングキャッシュではさらにレジスタにもデータを格納する．ノンブロッキングキャッシュはヒットアンダーミスと呼ばれることもある．

この方式では，ターゲットレジスタはデータが転送されるまでの間使用できない状態になる．したがってターゲットレジスタに対する読み出し/書き込みが起きると，データ転送が終わるまでパイプラインがストールしてしまい，ペナルティを隠せなくなる．

ライン転送のサイクル数が大きい場合，使用できない状態のターゲットレジスタが多くなり，演算に用いるためのレジスタが不足する．これを補うためには，アーキテクチャ上十分な数のレジスタを用意するか，またはレジスタリネーミングと呼ばれる方式を用いて実質的に多くのレジスタを用意しなければならない．

このことより，あるアドレスのデータを使用することがあらかじめ分かっている場合は，ソフトウェアプリフェッチの方が安価に性能を改善できる．逆に，あらかじめアドレスが分からない場合はノンブロッキングキャッシュが有効となる．このような場合，レジスタに格納するデータをただちに演算で使用することが多い．上で示したように，この演算の実行はデータ到着まで待たされるため，ライン転送のサイクル数が大きいと，待っているサイクルが長くなり，ペナルティを隠すことができない．

以上よりノンブロッキングキャッシュが最も効果を発揮するのは，2階層のキャッシュでL1のミスペナルティが比較的短く，L2のミス率が小さい場合ということになる．

## 6. キャッシュヒント

### 6.1 書き込みヒント

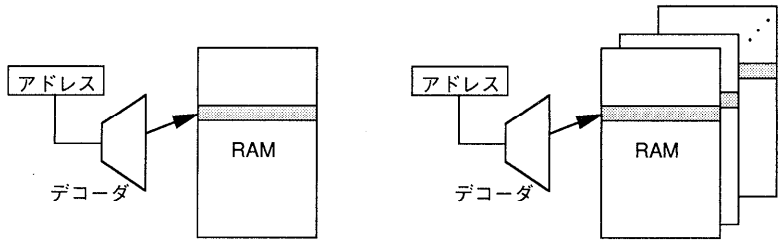
この方式はデータの書き込み（＝ストア）を行う命令にて，キャッシュミスした場合にライン転送をしなくても構わないというヒントを与える方式である．プログラムは，そのアドレスを含むライン全部に書き込みを行うことを保証する必要がある．この方式はたとえばメモリのクリアに用いられる．クリア動作のためには古いデータは必要でなく，キャッシュミスした場合のライン転送は冗長である．書き込みヒントを用いることで余分なライン転送を省くことができる．

ストア動作でキャッシュをミスした場合，もともとライン転送しないという方式もある．ライン転送する場合をストアアロケート方式，しない場合をストアノンアロケート方式と呼ぶ．ストアノンアロケート方式では，ストア動作でキャッシュをミスした場合，データをメインメモリに転送する必要がある．このとき書き込みヒントを用いると，いくつかのデータをマージしてメインメモリに転送することが可能となる．

### 6.2 読み出しヒント

この方式はデータの読み出しを行う命令にて，キャッシュミスした場合にメインメモリから転送されたラインをキャッシュに書き込まなくても構わないというヒントを与える方式である．キャッシュ上のミスしたエントリに別アドレスのデータが登録されていた場合，ライン転送によって上書きされると，前のデータがキャッシュからなくなってしまう．前のデータが繰り返し参照される場合，新たなライン転送が生じて性能が低下してしまうという問題がある（これは Cache pollution と呼ばれる）．

読み出しヒントはこれを防ぐための方式であり，PA-RISCアーキテクチャにて導入されている(PA-RISC 1.1 3rd edition)．プログラムは参照頻度の小さいデータの読み出しを行うとき，読み出しヒントを指定する．このデータがキャッシ




(a) ダイレクトマップ (b) セットアソシアティブ

図-3 キャッシュの構成

ュミスした場合，転送されたラインをレジスタには格納するが，キャッシュへの上書きを抑止する．

キャッシュがダイレクトマップであるとページムーブを行う際などにCache pollutionによる性能低下が起きやすい．これは図-3 (a) に示すように，あるアドレスのデータが登録されるエントリがただ一つしかないからである．図-3 (b) のようにキャッシュがセットアソシアティブまたはフルアソシアティブであると，あるアドレスに対応するエントリは複数あるためCache pollutionによる影響は小さくなる．

PA 7200プロセッサでは，読み出しヒントが指定されるとキャッシュには格納せずに，アシストキャッシュと呼ばれる小容量のバッファに格納する[3]．

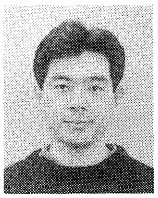
## 7. おわりに

最近のRISCプロセッサにおけるキャッシュの方式について概観した．RISCと一口にいっても，命令仕様などが相当に異なるアーキテクチャもある．それではRISCとは何かというと，コンパイラ，論理方式，デバイスなどあらゆる分野の最新テクノロジーをバランスよく吸収して，性能および機能の向上を図るアーキテクチャだと筆者は考えている．高速処理以外に，低消費電力，低コストなどもアーキテクチャ設計のターゲットに含まれる．他のComputer Science分野と同様，ここでもアメリカが圧倒的に優位な状況にある．最近では生き残りをかけてアーキテクチャ間での協調も現われてきており，これからの動向が注目されている．

## 参 考 文 献

1) Gwennap, L.: PPC 604 Powers Past Pentium, Microprocessor Report, Vol. 8, No. 5 (Apr. 1994).
2) Edmondson, J. and Rubinfeld, P.: An Overview of the 21164 Alpha AXP Microprocessor, Proc. HOT Chips VI, pp. 1-8 (Aug. 1994).
3) Kurnpanek, G., Chan, K., Zheng, J., DeLano, E. and Bryg, W.: PA 7200: A PA-RISC Processor with Integrated High Performance MP Bus Interface, Proc. the COMPCON Spring '94, pp. 375 382 (Mar. 1994).
4) Callahan, D., Kennedy, K. and Porterfield, A.: Software Prefetching, Proc. the 4 th International Conference on Architectural Support for Programming Languages and Operatiing Systems, pp. 40-52 (Apr. 1991).

(平成6年9月21日受付)



**松原 健二** (正会員)

1962年生．1984年東京大学電子工学科卒業．1986年同大学院情報工学修士課程修了．同年日立製作所に入社．Mシリーズメインフレーム開発の後，1990年よりPA-RISCプロセッサの開発に従事．どうすれば日本で高性能プロセッサが作れるようになるかに関心を持つ．