

マイクロプロセッサの構成の ソフトウェアシミュレータによる性能解析

山本崇夫 山口龍一 内海則夫 吉本哲朗 長尾彰文* 枝松壽一

松下電器産業株式会社 半導体研究センター
松下電子工業株式会社 システム開発センター*

大阪府守口市八雲中町3丁目1番1号
京都府長岡京市神足焼町1番地*

あらまし マイクロプロセッサのマイクロアーキテクチャに於て、以下の構成要素の違いの全体の性能への影響を定量的に解析する。
(1) スーパースカラ型の命令発行機構 (インオーダー、スコアボード、レジスタリネーミング)。
(2) いくつかの分岐予測アルゴリズムに基づく投機的命令実行。
(3) TLBの構成及びページテーブルキャッシュ。
評価はソフトウェアシミュレータを用いて仮想マイクロプロセッサを作成し、SPECベンチマークプログラムを実行させることにより行う。

和文キーワード マイクロプロセッサ、スーパースカラ、命令発行、投機的実行、TLB、性能解析

Performance Evaluation of a Microprocessor Using Software Simulation

Takao Yamamoto, Ryuichi Yamaguchi, Norio Utsumi, Tetsuro Yoshimoto,
Akifumi Nagao*, and Hisakazu Edamatsu

Semiconductor Research Center, Matsushita Electric Industrial Co., Ltd.
System Development Center, Matsushita Electronics Co., Ltd.*

Moriguchi, Osaka, 570 JAPAN
Nagaokakyo, Kyoto, 617 JAPAN*

Abstract

This paper describes the quantitative performance evaluation of a microprocessor. The analysis covers the following:

- (1) instruction issue methods for superscalar execution: in-order issue, score-boarding, and register-renaming methods,
- (2) speculative execution for the register-renaming method with various related branch prediction algorithms, and
- (3) Translation Look-aside Buffer which has various configurations with Page Table Cache.

Evaluations use software simulation to execute some SPEC programs.

英文 key words performance evaluation, microprocessor, superscalar, instruction issue, speculative execution, TLB

1 はじめに

高性能なマイクロプロセッサの設計に於いては、マイクロアーキテクチャ、コンパイラ技術、回路設計技術、デバイス技術など、それぞれの要素が効率良く機能することが不可欠である。また、マイクロアーキテクチャに於いては、それぞれの構成要素が相互に性能を発揮することによって、全体の性能が決定される。本稿では、マイクロプロセッサのマイクロアーキテクチャに於いて、スーパースカラ型の命令発行機構と、投機的命令実行機構と、TLB (Translation

Look-aside Buffer) のそれぞれについて、ソフトウェアによって仮想マイクロプロセッサを作成しベンチマークプログラムを実行させることにより、それぞれのマイクロアーキテクチャがマイクロプロセッサの性能に与える影響を解析する。

2 命令発行機構の解析

スーパースカラ技術では、単位サイクルあたり複数の命令を、マイクロプロセッサ内部の複数の演算器に発行して高性能化を達成する。命令発行の際、以下の依存関係により同時発行が阻害される。

- ・データ依存：
後続命令が先行命令の結果を参照
- ・逆依存：
先行命令の読込元と後続命令の書込先が同じ
- ・出力依存：
先行命令と後続命令の書込先が同じ
- ・制御依存：
分岐条件が先行命令の結果に依存

上記依存を解消し、より多くの命令を発行するため、種々の命令発行方式^[1]が提案されている。従来の解析^[2]では、純粋に命令の並列度を評価することを目的としていたので、命令は理想的に供給されるとしていた。しかしながら、現実のマイクロプロセッサに適用する場合、命令/データ供給能力は分岐命令とキャッシュメモリにより左右されるという問題がある。そこで、仮想的にマイクロプロセッサモデルを構築し、分岐機構とメモリ階層による性能阻害要因を組み込んで、種々の命令発行方式を実現した

場合の性能を解析した。

2.1 構成

以下の命令発行方式を比較した。

・インターロック制御：発行対象命令が先行命令に対していずれかの依存関係にある場合、当該命令と後続命令の演算器への発行を止める。したがって、命令はin-orderに発行される。

・スコアボード：発行対象命令が先行命令に対してデータ依存あるいは出力依存関係にある場合、当該命令の演算器への発行を止める。一方、後続命令は先行命令に対して依存がない場合、out-of-orderに命令発行される。ここでは文献^[1]に従った。

・レジスタリネーミング：発行対象命令が先行命令に対してデータ依存にある場合のみ、当該命令の演算器への発行が止められる。後続命令はout-of-orderに命令発行される。

2.2 解析モデル

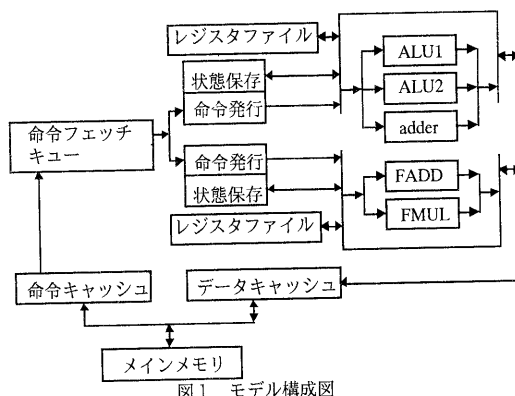


図1 モデル構成図

図1に想定したモデル構成図を示す。命令は、命令フェッチキューによりメモリ階層から読み出され、命令発行部で一旦保持される。メモリから命令発行部へは1サイクル当り最大4命令が送られ、整数命令と浮動小数点命令それぞれ最大16命令が保持される。命令発行部では、命令間の依存関係を調べ、実行可能命令を演算器に発行する。演算器は、整数演算用にALUと、メモリアクセスのアドレス計算用の加算器と、両方の機能を兼ねた演算器の計3個を設定した。浮動小数点演算用には、加算器と乗算器の計2個を設定した。以上の演算器を使用する

ことで、最大5命令が同時に演算器に発行される。命令は、SPARC命令セット^[3]を使用する。命令発行部は前記3種のアルゴリズムをモデル化した。また状態保存部は、命令間の依存関係の解消に使用した。

解析には上記構成をモデル化し、SPARC実行形式のオブジェクトを実行できる専用ソフトウェアシミュレータを開発した。

2.3 シミュレーション結果

ベンチマークプログラムには、SPECベンチマークの一部（整数演算系：gccとespresso、浮動小数点系：doducとfpppp）を使用した。

分岐命令の頻度を表1に示す。図2、図3、図4、図5にシミュレーション結果を示す。図2はgccとespresso、図3はdoducとfppppそれぞれを実行した場合に命

表1 分岐命令頻度

ベンチマーク	分岐命令頻度
gcc	0.183
espresso	0.178
doduc	0.064
fpppp	0.015

令発行部で保持される平均命令数を示す。横軸は命令発行方式、縦軸は保持される命令数を示す。図4はgccとespresso、図5はdoducとfppppそれぞれを実行した場合の命令発行機構と単位時間あたりの命令実行数（IPC）の関係を示す。横軸は命令発行方式、縦軸はIPCを示す。キャッシュサイズは8/16/32/∞（すべてキャッシュヒット）Kbyteと変化させた。

図2と図3から、整数系のプログラムでは浮動

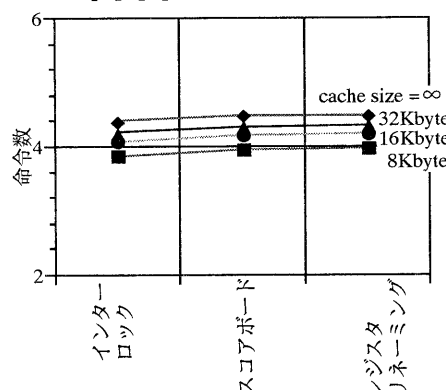


図2 整数演算系(gcc,espresso)での命令発行機構による命令発行部で保持される命令数の変化

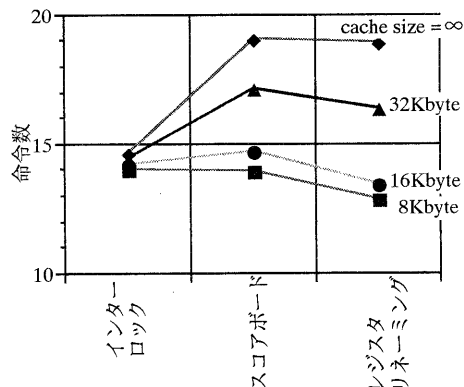


図3 浮動小数点系(doduc,fpppp)での命令発行機構による命令発行部で保持される命令数の変化

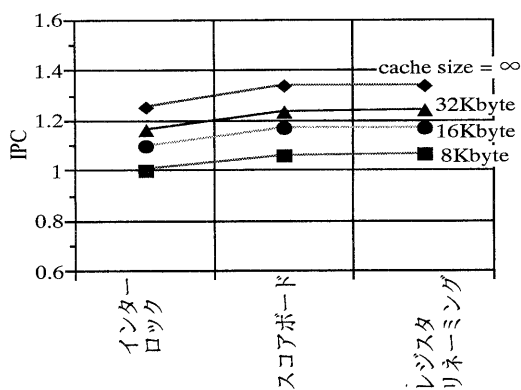


図4 整数演算系(gcc,espresso)での命令発行機構によるIPCの変化

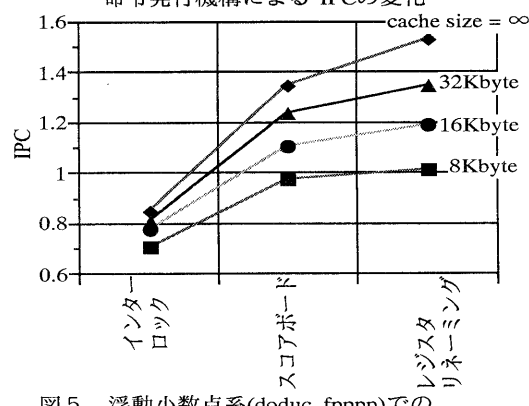


図5 浮動小数点系(doduc,fpppp)での命令発行機構によるIPCの変化

小数点系プログラムに比較して、命令発行機構に保持される命令数が極めて少ない。これは表1から整数系プログラムは浮動小数点系プログラムに比べて分岐命令の出現頻度が高いからであると推測できる。

図4と図5から、浮動小数点系ではキャッシュが大きくなるにしたがって命令発行機構の違いによるIPCの比率差が大きくなるが、整数系ではほとんど差がない。これは、整数演算系では分岐命令の出現頻度が高いために、スコアボードやレジスタリネーミング機構へ十分な命令供給が行われないためと思われる。

3 投機的命令実行機構の解析

2で行った解析から分岐命令による制御依存が性能を抑制していると推測される。制御依存を解消する方法として、投機実行がある。ここでは、投機的命令実行機構が性能に対してどのような影響を与えるかについて調べるため、まず分岐命令の解析を行い、さらに分岐予測精度の向上、命令供給能力の向上による性能向上について解析する。解析方法として、2と同様に仮想のマイクロプロセッサモデルを作成しベンチマークプログラムを流すことにより行った。

3.1 構成

以下の投機実行方式を比較した。

A：分岐成立予測

条件分岐命令が、分岐すると予測して命令をプリフェッチし、投機実行を行なう。

B：分岐不成立予測

条件分岐命令が、分岐しないと予測して命令をプリフェッチし、投機実行を行なう。

C：PC相対分岐のオフセットによる予測

PC相対条件分岐命令の、アドレスオフセットが負なら、分岐成立予測を行ない、正なら分岐不成立予測を行なう。

D：直前の分岐結果による予測

直前に実行された、条件分岐命令の結果が分岐成立なら、分岐成立予測を行ない、不成立なら分岐不成立予測を行なう。

E：分岐履歴テーブルによる予測

最新の複数の条件分岐命令の判定結果をテーブルに保持し、同じ分岐命令の直前の結果がテ

ーブルにあり、分岐成立であれば、分岐成立予測を行ない、不成立なら分岐不成立予測を行なう。また、テーブルになれば、直前の分岐結果による予測を行なう。

3.2 解析モデル

解析に使用したモデルは、2で使用したモデルのうち、レジスタリネーミングを命令発行方式として用いたものを使用する。投機的命令実行機構は命令スケジューリング部の一部であり、通常、分岐命令が終わるまで分岐命令後の命令は実行できないが、分岐命令確定以前に分岐予測先の命令を仮に実行することを可能にするものである。この解析では分岐は1個しか越えないとする。

3.3 分岐命令の解析

上記のモデルに対して2で使用したベンチマークプログラムを実行した。分岐一分岐命令間隔を図6に、分岐命令確定サイクルを図7にそれぞれ示す。図6では分岐一分岐命令間隔が3以下である割合が整数演算系では40%、浮動小数点系では20%程度あり、分岐を1個しか越えないで投機的命令実行する場合の供給できる最大命令数に上限があることがわかる。

図7の分岐命令確定サイクルを見ると、総分岐

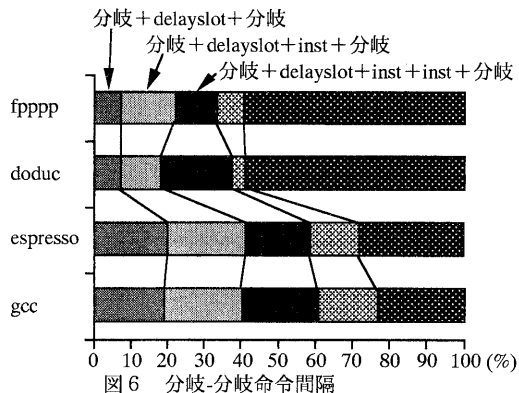
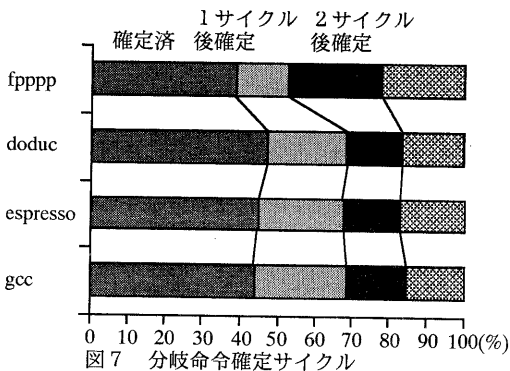


図6 分岐-分岐命令間隔

命令の45%近くが命令スケジューリング部に入った時点で分岐確定している。これは投機的命令実行をする必要がないことを示しており、プログラム間の差は余りない。

また残り40%程度の方岐命令も命令スケジューリング部に投入後、2サイクルの間で実行で

きている。これは分岐命令確定以前に分岐予測先の命令を供給できるサイクルが短いことを示している。



また、表 1 より整数演算系の方が浮動小数点系より多くの分岐命令を含むので投機的命令実行の効果が高くなることが予想できる。

3.4 シミュレーション結果

投機的命令実行機構による性能向上は、分岐命令確定前に分岐予測に従って投機的に実行される命令数を増やすことで計れるはずである。

分岐命令の解析により、投機的命令実行機構の効率を上げるためには次の 2 つの方法が考えられる。

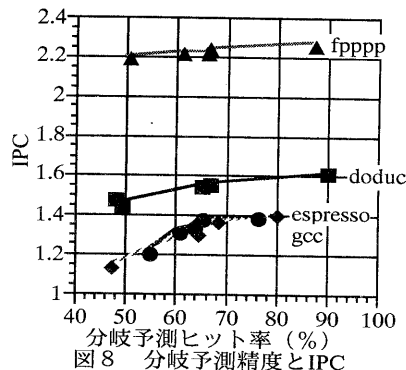
- (1) 分岐予測精度の向上
- (2) 命令供給能力の向上

3.4.1 分岐予測制度の向上

分岐予測の方式として 3. 1 に示した 5 種類を用いてシミュレーションを行い、その分岐予測精度と性能向上の関係を図 8 に示す。

図 8 によれば分岐予測精度の向上と IPC の関係はおおよそ比例しているが、整数演算系では分岐予測ヒット率が 70 % 辺りまでの傾きに比べてそれ以降の傾きが小さく、浮動小数点系の傾きと同等であることが分かる。これは、整数演算系のプログラムは分岐命令の頻度が大きいので投機的命令実行の効果が大きい、分岐予測ヒット率が 70 % 程度を越えると命令供給能力が命令処理能力に追いついたためであると考えられる。また、全てのプログラムにおいて分岐履歴テーブルによるものの分岐予測ヒット率が一番高く、IPC も高い。しかし、分岐履歴

テーブルはハードウェアとしてはかなり大きい。したがって、70 % 程度の分岐予測ヒット率が得られる分岐成立予測が効率的であると言える。さらにコンパイラなどにより分岐予測ビットの付加等の分岐予測が実現されるならば、より高い分岐予測ヒット率が得られ、投機的命令実行機構の効果を高めることができる。



3.4.2 命令供給能力の向上

命令キャッシュからラインサイズ分の命令を読みだせる場合において、分岐命令がラインの途中のアドレスを要求する場合は、読みだしたラインの内に無効命令を含んでしまう。短いサイクルでの命令供給を増加させるために、無効命令の部分を有効命令で埋めて常にラインサイズ分の有効命令を命令供給部へ送ることを命令のアラインメント供給と呼ぶ。

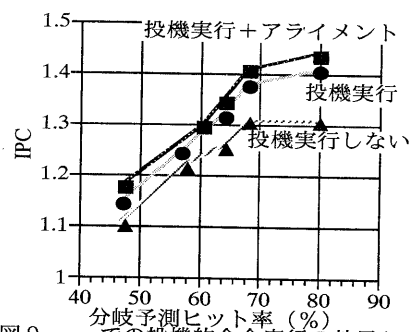


図 9 gccでの投機的命令実行の効果と命令アラインメントの効果

図 9 に gcc での命令アラインメントの効果を示す。どの分岐予測ヒット率においても 2 ~ 3 % 程度の IPC 向上が見られる。しかし、命令アラインメントを実現すると、命令キャッシュの大

きさが1.5倍程度になることを考慮にいれると、コストに見合わない。

投機的命令実行をしない場合のIPCも同時に示す。この場合の分岐予測とは、命令のプリフェッチのみを行うことであり、この効果がIPCに反映されている。IPC1.3あたりで飽和しているが、これは命令供給能力が命令の処理能力を上回るためである。この図よりIPC1.3以上を得るには投機的命令実行が有効であると言える。

投機的命令実行の効果は、投機的命令実行をしない場合に比べて最大7.6%のIPCの向上が見られる。分岐予測ヒット率が高くなるが、従って投機的命令実行の効果は大きくなるが、これは投機的実行した命令がキャンセルされる可能性が小さくなるからである。

さらなる命令供給能力の向上は、複数の分岐命令を越える投機命令実行を行うことで可能であるが、分岐確定サイクルの短さ、分岐予測ミス時のペナルティを考慮する必要がある。

4 TLBの解析

アドレス変換器(TLB)^[4]は、CAMとRAMの2つのメモリから構成され、論理アドレスを物理アドレスに変換する際のキャッシュである。従来TLBのエントリー数とアドレス変換ミス率の関係は報告されているが^[5]、TLBの構成とアドレス変換ミス率の関係、ページテーブルキャッシュ(PTC)^[6]を用いてテーブルウォークを高速化した場合の効果の報告は少ない。そこで、命令レベルシミュレータ^[6]を用い、3種類のTLBの構成について、実際のアプリケーション(SPECベンチマークプログラム^[7])を実行した時に発生するアドレス変換のペナルティを評価した。またPTCの評価を行った結果も報告する。なお、命令セットとMMUはSPARCのものを使用した。^[3]

4.1 構成

TLBの設計で重要なことは、

- (1) アドレス変換ミス率を最小にする。
- (2) アドレス変換ミスが発生した場合テーブルウォークに必要とする時間を最小にすることである。(1)のためにTLB、(2)のため

にPTCのいくつかの構成を評価に用いた。

4.1.1 TLBの構成

3つの構成を比較した。論理アドレスから物理アドレスへの変換は4Kバイトのページサイズに従う。各アドレス登録部の構成はフルアソシエティブ方式とし、リプレースアルゴリズムはFIFOとする。テーブルウォークには4レベルの変換テーブルを用いる。

・ITLB/DTLB: 命令用のTLB(ITLB)とデータ用のTLB(DTLB)を用意する。各TLBがミスするとテーブルウォークを行う。

・ITLB/DTLB/GTLB: ITLBとDTLBに加えて命令/データ共用のTLB(GTLB)を用意する。GTLBは、ITLBあるいはDTLBがミスすると補助的に各々のアドレス変換を行う。GTLBがミスするとテーブルウォークを行う。

・ITLB/GTLB: ITLBとGTLBを用意する。データアドレスの変換をGTLBで行うのに加えて、ITLBがミスすると補助的に命令アドレスの変換も行う。GTLBがミスするとテーブルウォークを行う。

表2にシミュレーションを行った構成とエントリー数について示す。

表2 TLBの3つの構成

構成	エントリー数
ITLB/DTLB	32/32
	64/64
ITLB/DTLB/GTLB	8/8/128
	16/16/128
	32/32/128
ITLB/GTLB	8/64
	8/128
	16/128

4.1.2 PTCの構成

PTCは、テーブルウォークしたアドレス情報を格納しておくキャッシュである。テーブルウォークが発生すると通常メモリアクセスを4回行うが、PTCがヒットすると1回ですむ。

2つの構成を比較した。PTCのリプレースアルゴリズムはFIFOとした。

・IPTC/DPTC: 命令用(IPTC)とデータ用(DPTC)を持つ。

・GPTC：命令／データ共用（GTPC）の
みを持つ。

4.2 解析モデル

図10にモデル構成図を示す。実際のアプリケーションプログラムを、命令レベルシミュレータを用いて、実行命令及びデータアクセス命令の論理データアドレスを求め、アドレス変換シミュレータに入力する。

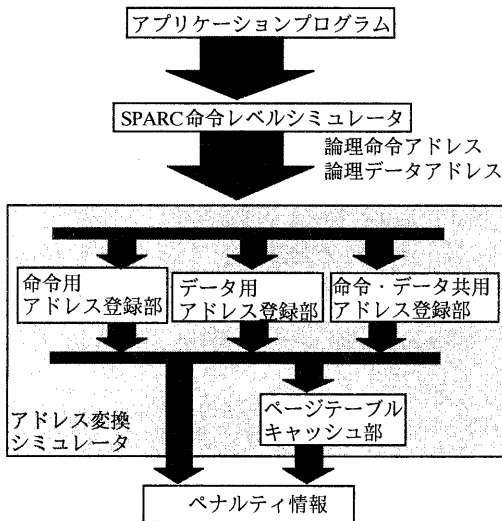


図10 モデル構成図

各部で発生するペナルティを表3に示す。

表3 ペナルティ一覧

	ペナルティ数
命令アドレス登録部ミス	1クロック
データアドレス登録部ミス	
命令データ共用アドレス登録部ミス	
ページテーブルキャッシュ部ヒット	8クロック
ページテーブルキャッシュ部ミス	32クロック

4.3 シミュレーション結果

ベンチマークプログラムには、SPECベンチマークプログラムの一部（gcc、doduc、fpppp、spice2g6）を使用した。命令及びデータアクセス数を表4に示す。

4.3.1 TLBの構成とアドレス変換ミス

図11にシミュレーション結果を示す。横軸にTLBの構成、縦軸にペナルティサイクルの

表4 命令/データアクセス数 (X1000000)

	命令	データ
gcc	25.6	7.1
doduc	65.5	35.4
fpppp	107.5	92.5
spice2g6	1224.3	298.4

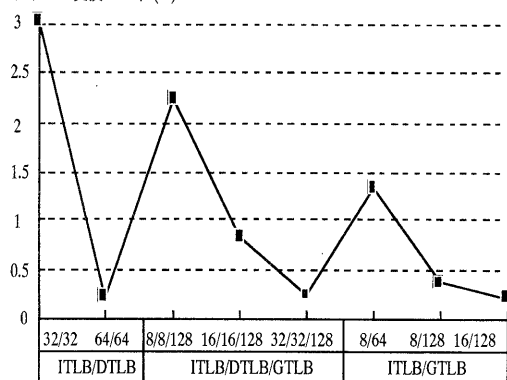
合計の命令アクセスサイクルに対する割合（アドレス変換ミス率）を示す。

[1] ITLB (64) / DTLB (64) のアドレス変換ミス率が極めて小さい。これは命令・データのアドレス情報のほとんど全てが64エンタリーに収まるためと予想される。

[2] ITLB (32) / DTLB (32) に比べてITLB (32) / DTLB (32) / GTLB (128) のアドレス変換ミス率が極めて小さい。ITLB (8) / GTLB (64) に対するITLB (8) / GTLB (128) も同様である。GTLBが、ITLBあるいはDTLBのアドレス変換ミスを補っていることが分かる。

[3] ITLB / DTLB / GTLBの構成は、GTLBのエンタリー数が等しいのかかわらず、ITLBとDTLBのエンタリー数がそれぞれ小さい構成（8あるいは16）の方がアドレス変換ミス率が高い。ITLBあるいはDTLBがミスしてGTLBにアクセスする1ク

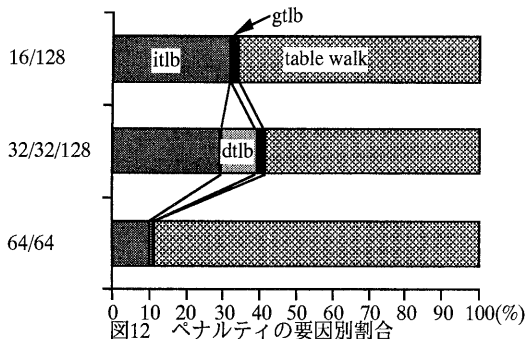
アドレス変換ミス率 (%)



構成とエンタリー数
図11 TLBのシミュレーション結果

ロックのペナルティの積み重ねが要因である。

図12にアドレス変換ミス率の低い3つの構成のアドレス変換ミス率の要因別の割合を示す。3つの構成ともアドレス変換ミス率はほぼ同じであるが、図12によるとITLB(64)/DTLB(64)の構成ではアドレス変換ミス率の約90%がテーブルウォークによるものであることが分かる。



4.3.2 PTCとテーブルウォーク数

4で示した2種類のPTCの構成をテーブルウォークがアドレス変換ミス率に占める割合の大きいITLB(64)/DTLB(64)の構成に含めて比較した。図13にPTCのエントリ数を変化させた時の、テーブルウォークのサイクル数を示す。

[1] PTCは4個用意すれば十分な効果が得られる。

[2] 同じ個数のPTCを用意した場合、GPPTCの方が優れている。これは、命令/データのTLBミスの頻度がアプリケーションの実行場所によって異なるので、ダイナミックにPT

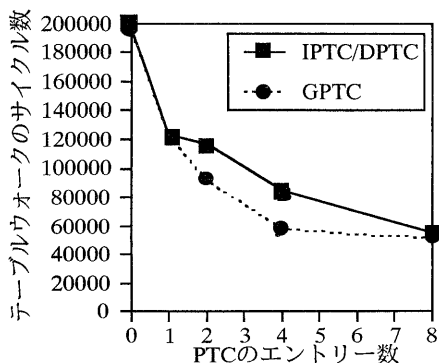


図13 PTCのシミュレーション結果 (gcc)

Cの割り振りを行ったほうがPTCミスを起こしにくいと予想される。

5 おわりに

分岐命令とメモリ階層による性能阻害要因を組み込んで命令発行機構と投機的命令実行の評価を行った。シミュレーション結果から、命令/データの供給能力と命令発行方式の命令実行数の向上の効果が定量的に評価できた。

整数系プログラムでは、分岐による命令供給阻害により、命令発行方式の違いによる性能差が小さくなるが、投機的命令実行により性能を向上することが期待できる。

また、投機的命令実行による性能向上は、gccの場合、投機的命令実行しない場合と比べて約8%見込まれることが分かった。

さらに、TLBの性能解析では、GTLBの効果を確認した。但し、実装を考えた場合はTLBのエントリ数の差によるアクセスタイムの差を考慮する必要がある。

参考文献

- [1] S.Weiss et al.: "Instruction Issue Logic in Pipelined Supercomputers", IEEE Trans. Computers, Vol.c33, No.11, p.1013, 1984.
- [2] M. Butler et al.: "Single Instruction Stream Parallelism Is Greater Than Two", 18th Annual Int. symp. on Computer Architecture, p.276, 1991.
- [3] "The SPARC Architecture Manual", Prentice-Hall, 1992.
- [4] Miyake J: "A Highly Integrated 40-MIPS (Peak) 64-b RISC Microprocessor", IEEE J. Solid-State Circuits, Vol.25, No.5, pp.1190-1198, Oct. 1990.
- [5] Clark D: "Performance of the VAX-11/780 Translation Buffer: Simulation and Measurement", ACM Translation on Computer Systems, Vol.3, No.1, pp.32-62, Feb. 1985.
- [6] Utsumi N: "Performance Evaluation of a Translation Look-aside Buffer for Highly Integrated Microprocessors", IEICE TRANS. (to be published)
- [7] SPEC newsletter, Vol. 1.1, Fall 1989.