(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4294778号
(P4294778)

(45)発行日　平成21年7月15日(2009.7.15)　(24)登録日　平成21年4月17日(2009.4.17)

(51)Int.Cl.		F I		
G06F 11/34	(2006.01)	G06F	11/34	S
G06F 9/38	(2006.01)	G06F	11/34	L
		G06F	9/38	380C

請求項の数 34　外国語出願　(全 38 頁)

(21)出願番号　特願平10-375360
(22)出願日　平成10年11月26日(1998.11.26)
(65)公開番号　特開平11-272517
(43)公開日　平成11年10月8日(1999.10.8)
審査請求日　平成17年11月9日(2005.11.9)
(31)優先権主張番号　08/980164
(32)優先日　平成9年11月26日(1997.11.26)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100059959
弁理士　中村　稔
(74)代理人　100067013
弁理士　大塚　文昭
(74)代理人　100065189
弁理士　宍戸　嘉一
(74)代理人　100096194
弁理士　竹内　英人

最終頁に続く

(54)【発明の名称】プロセッサパイプラインにより処理される相互作用の特性の統計値を推定する方法

(57)【特許請求の範囲】

【請求項１】

複数の処理段を有するコンピュータシステムのパイプラインで処理される命令間の相互作用の特性の統計値を推定するためにコンピュータが実行する方法であって、

パイプラインの第１段へ命令をフェッチし；

フェッチした命令から命令のセットをランダムに選択し、その選択された命令のセットのサブセットは互いに同時に実行され；

前記の選択された命令のサブセット間の距離を特定し；

前記の選択された命令のセットがパイプラインで処理されている間にコンピュータシステムの状態情報を記録し；

記録された状態情報をソフトウェアへ通信し；そして

前記の選択された命令の複数のセットからの前記の記録された状態情報を統計学的に分析して、前記の選択された命令のセット間の相互作用の統計値を推定する

段階を備えたことを特徴とする方法。

【請求項２】

ランダムに選択されたそれぞれのセットは少なくとも２つの命令を有する請求項１に記載の方法。

【請求項３】

前記の分析は更に、前記の記録された状態情報のファンクションに基づいて前記の複数のセットの中の一つのサブセットを選択する段階を含む請求項１に記載の方法。

【請求項４】
前記のファンクションは、選択された命令の識別情報に基づいて選択を行い、この識別情報は、選択された命令のアドレス、選択された命令を実行するプロセス、選択された命令がその中で実行するアドレススペース番号、そして選択された命令がその中で実行するハードウェアコンテクスト又はスレッド番号を含む請求項３に記載の方法。
【請求項５】
前記のファンクションは、選択された命令に関連した経路識別情報に基づいて選択を行う請求項３に記載の方法。
【請求項６】
前記のファンクションは、選択された命令の演算コードに基づいて選択を行う請求項３に記載の方法。
【請求項７】
前記のファンクションは、選択された命令に対して記録された待ち時間情報に基づいて選択を行う請求項３に記載の方法。
【請求項８】
前記のファンクションは、選択された命令に対して記録された事象情報に基づいて選択を行う請求項３に記載の方法。
【請求項９】
前記のファンクションは、選択された命令のオペランドの値に基づいて選択を行う請求項３に記載の方法。
【請求項１０】
前記のファンクションは、選択された命令のデータ依存関係に基づいて選択を行う請求項３に記載の方法。
【請求項１１】
前記のファンクションは、選択された命令が経験する事象カウントに基づいて選択を行う請求項３に記載の方法。
【請求項１２】
フェッチのレートはクロックにより決定され、そして前記の事象カウントは、選択された命令のいずれか２つのフェッチ間のサイクル数と、選択された命令のいずれか２つの間でリタイアする命令の数と、選択された命令のいずれか２つの間でパイプラインの特定の段に入る命令の数と、プロセッサ、メモリシステム、ネットワーク又はＩ／Ｏサブシステム事象の所定の測定値を含む請求項１１に記載の方法。
【請求項１３】
前記の距離は、選択された命令のセットのいずれか２つのメンバー間のクロックサイクル数として特定される請求項１に記載の方法。
【請求項１４】
前記の距離は、選択された命令の最初から最後までのフェッチ間にフェッチされた命令の数として特定される請求項１に記載の方法。
【請求項１５】
前記の距離は、フェッチされた命令の最初から最後のフェッチまでのサイクルの細分化の数として特定される請求項１２に記載の方法。
【請求項１６】
前記の距離は、一定の間隔にわたって変化する請求項１に記載の方法。
【請求項１７】
前記の距離は、一定の間隔にわたりランダムにそして均一に変化する請求項１６に記載の方法。
【請求項１８】
前記の統計学的分析は、選択された命令の特定のセットにおける選択された命令の記録された状態情報のファンクションを計算する請求項１７に記載の方法。
【請求項１９】

前記のファンクションは、特定命令の複数の実行中の平均同時性レベルを計算する請求項１８に記載の方法。
【請求項２０】
前記のファンクションは、特定の命令の複数回実行中のパイプラインの特定段の平均利用を計算する請求項１８に記載の方法。
【請求項２１】
前記のファンクションは、特定の命令の複数回実行中に浪費されるイッシュースロットの数を計算する請求項１８に記載の方法。
【請求項２２】
前記の統計学的分析は、選択された命令の特定のセットに対する記録された状態情報のファンクションを計算する請求項３に記載の方法。
【請求項２３】
前記のファンクションは、同一の選択された命令を含むセットに基づいてループ反復待ち時間を推定する請求項２２に記載の方法。
【請求項２４】
前記のファンクションは、同時発生のループ反復の数を推定する請求項２２に記載の方法。
【請求項２５】
前記のファンクションは、実行コンテクスト間の遷移の頻度を推定する請求項２２に記載の方法。
【請求項２６】
実行コンテクストはスレッド又はプロセスである請求項２５に記載の方法。
【請求項２７】
前記のファンクションはロードから使用までの待ち時間を計算する請求項１８に記載の方法。
【請求項２８】
前記のファンクションは、選択された命令の実行間の重なる量を計算する請求項１８に記載の方法。
【請求項２９】
重なりはクロックサイクルで測定される請求項２８に記載の方法。
【請求項３０】
前記の重なりはフェッチされた命令で測定される請求項２８に記載の方法。
【請求項３１】
前記の実行コンテクストはプログラムの部分である請求項２５に記載の方法。
【請求項３２】
前記の遷移は割り込み又はトラップハンドラーへの、そしてそこからのものである請求項２５に記載の方法。
【請求項３３】
前記のファンクションは、データを発生する第１の選択された命令と、発生されたデータを消費する第２の選択された命令との間の待ち時間を推定する請求項２２に記載の方法。
【請求項３４】
前記の第１の命令はロードオペレーションを実行し、そして第２の命令はそのロードオペレーションの結果を使用する請求項２９に記載の方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、一般に、コンピュータシステムの性能測定に係り、より詳細には、プロセッサパイプラインによって処理される命令間の相互作用の特性の統計値を推定することに係る。

【０００２】
【従来の技術】
コンピュータプロセッサは、益々高速になっているが、ソフトウェアアプリケーションの性能は、それに歩調が合っていない。大型の商業用途の場合に、命令当たりの平均プロセッササイクル（ＣＰＩ）値が２．５ないし３という大きさである。４ウェイ命令イッシューアプロセッサでは、ＣＰＩが３であることは、１２ごとに１つのイッシュースロットしか良好に使用されないことになる。ソフトウェアスループットがハードウェアの改良となぜ歩調が合わないかを理解することが重要である。
このような問題をメモリの待ち時間に転嫁するのが一般的であり、実際に、多くのソフトウェアアプリケーションは、データ転送が完了するのを待機して多数のサイクルを費やす。しかしながら、分岐予想ミスのような他の問題も、プロセッササイクルを浪費する。一般的な原因とは独立して、システムアーキテクチャー並びにハードウェア及びソフトウェアエンジニアは、複雑なプロセッサを組み込んだ近代的なコンピュータシステムの性能を改善するために、どの命令がストールしているかそしてなぜかを知る必要がある。
【０００３】
通常、これは、システムが動作している間にその振る舞いの「プロファイル」を発生することにより行われる。プロファイルとは、性能データの記録である。しばしば、プロファイルは、性能のボトルネックを容易に識別できるようにグラフ的に発生される。
プロファイル形成は、計装及び模擬により行うことができる。計装では、プログラムの実行中に特定事象を監視するためにプログラムに付加的なコードが追加される。模擬は、実際のシステムでプログラムを実行するのではなく、人為的な環境においてプログラム全体の振る舞いをエミュレートするように試みる。
これら２つの方法は、各々、欠点を有する。計装は、追加命令及び余計なデータ参照のためにプログラム真の振る舞いを擾乱させる。模擬は、実際のシステムにおいてプログラムを実行する場合に比して実質的な性能オーバーヘッドを犠牲にして擾乱を回避する。更に、計装又は模擬では、大規模なソフトウェアシステム全体、即ちアプリケーション、オペレーティングシステム及びデバイスドライバコードをプロファイリングすることが通常困難である。
【０００４】
プロセッサのプロファイル情報を与えるために、ハードウェア実施の事象サンプリングを使用することもできる。ハードウェアサンプリングは、模擬及び計装に勝る多数の効果を有し、即ち性能を測定するためにソフトウェアプログラムを変更する必要がない。サンプリングは、比較的低いオーバーヘッドで全システムに作用する。実際に、最近では、低いオーバーヘッドのサンプリングをベースとするプロファイリングを使用して、パイプラインストール及びそれらの原因に関する詳細な命令レベル情報を収集することができる。しかしながら、多くのハードウェアサンプリング技術は、特定の事象を測定するように設計されているので融通性に欠ける。
デジタル社のＡｌｐｈａ　ＡＸＰ２１１６４、インテル社のペンティウイム・プロ及びＭＩＰＳ１００００は、データキャッシュ（Ｄキャッシュ）ミス、命令キャッシュ（Ｉキャッシュ）ミス及び分岐予想ミスのような種々の事象をカウントすることのできる事象カウンタを形成する。これらの事象カウンタは、カウンタがオーバーフローするときに割り込みを発生し、従って、カウンタの性能データを高レベルのソフトウェアでサンプリングすることができる。
【０００５】
事象カウンタは、特定のプログラム又はその一部分を実行する間にシステムが招いた分岐予想ミスの数のような集合情報を捕獲するのに有用である。しかしながら、既知の事象カウンタは、どの分岐命令が頻繁に予想ミスを生じるかのように状態情報を個々の命令に帰属させる点で有用性が低い。これは、事象カウンタがオーバーフローしそして割り込みを生じるときには、その事象を生じた命令のプログラムカウンタ（ＰＣ）がもはや使用できないためである。

命令をアウトオブオーダー（順序ずれして）でイッシューすることのできるプロセッサの動的なオペレーションを推測することが特に問題である。実際に、アウトオブオーダープロセッサで実行されるソフトウェアプログラムの振る舞いは極めて不可解で且つ理解が困難である。その具体的な例としてアウトオブオーダーのＡｌｐｈａ２１２６４プロセッサでの命令の流れについて考える。

【０００６】

スーパースカラープロセッサアーキテクチャー

実行順序

アウトオブオーダープロセッサは、命令を正しい順序でフェッチしそしてリタイアするが、命令をそれらのデータ依存性に基づいて処理する。命令の処理は、レジスタのマッピング、命令の発生及び実行を含む。命令は、それがフェッチされたときから、それがリタイア又はアボートするときまで、「フライト中」であると言える。

各プロセッササイクル中に、プロセッサパイプラインの第１段は、命令キャッシュ（Ｉキャッシュ）から命令のセットをフェッチする。命令のセットはデコードされる。命令デコーダは、フェッチされたセットのどの命令が命令流の一部分であるかを識別する。

【０００７】

フェッチすべき次の命令のＰＣを分析するには多数のサイクルを要するので、ＰＣは、通常、分岐又はジャンププレディクタ（予想子）により前もって予想される。予想を誤ったときには、プロセッサは、「不良」実行経路を占有する予想ミス命令をアボート（中止）し、そして「良好」経路において命令のフェッチを再スタートする。

命令を順序ずれ状態で実行できるようにするために、命令のオペランドに指定されたレジスタは、「読み取り後の書き込み」及び「書き込み後の書き込み」競合を防止するように動的に名前を付け直される。この名前の付け直しは、アーキテクチャー即ち「仮想」レジスタを物理的レジスタへとマッピングすることにより達成される。従って、同じ仮想レジスタに書き込む２つの命令は、それらが異なる物理的レジスタに書き込みそして仮想レジスタの消費者が適切な値を得るので、順序ずれ状態で安全に実行することができる。

【０００８】

レジスタマップ型命令は、そのオペランドが計算されそして適当な形式の機能的「実行」ユニットが得られるまで、イッシュー待ち行列に存在する。命令によって使用される物理的なレジスタは、命令がイッシューされるサイクルで読み取られる。命令は、それらが実行された後に、リタイアの準備ができたとマークされ、そしてプログラム順序における全ての手前のリタイア準備命令がリタイアしたときに、即ち命令が正しいプログラム順序でリタイアするときに、プロセッサによりリタイアされる。リタイアの際に、プロセッサは、命令によりなされる変更をシステムのアーキテクチャー「状態」へコミットし、そして命令により消費されたリソースを解除する。

【０００９】

予想ミス

分岐が誤って予想されるようなある場合には、命令をトラップし又は破棄しなければならない。これが生じたときには、現在の推測的な構造状態が、予想ミスが生じた実行点へと戻され、正しい命令においてフェッチが続けられる。

【００１０】

遅延

多数の事象が命令の実行を遅らせる。パイプラインの前方において、フェッチユニットは、Ｉキャッシュミスのためにストールするか、又はフェッチユニットは、予想ミスのために不良経路に沿って命令をフェッチすることがある。マップ手段は、空いた物理的レジスタが欠乏するか、又はイッシュー待ち行列に空きスロットが欠乏するためにストールすることがある。イッシュー待ち行列の命令はそれらのレジスタ依存性が満足されるか又は機能的実行ユニットが使用できるようになるのを待機する。

命令は、データキャッシュミスによりストールすることがある。命令は、それらが不良経路を下るように推測的に発生されるか、又はプロセッサが割り込みを行ったためにとラッ

プされることがある。これら事象の多くは、例えば、コードの検査により静的に予想することが困難であり、それらは全てシステムの性能を低下させる。この形式の状態情報を捕獲するのに単純な事象カウンタでは不充分である。加えて、遅延の長さを厳密に測定して、どの遅延に特に注目すべきかを決定することは困難である。

【００１１】

プログラマー又は最適化ツールが、スーパースカラー及びアウトオブオーダープロセッサ、又はこの点については任意のアーキテクチャー設計のプロセッサのような複雑なコンピュータシステムのソフトウェア及びハードウェア要素の性能を改善できるように、事象を特定の命令及びマシン状態に直接的に帰属させることが強く望まれる。

【００１２】

【発明が解決しようとする課題】

公知の事象カウンタに伴う問題

既知の事象カウンタに伴う主な問題は、カウンタをオーバーフローさせた事象を生じさせた命令が、通常は、例外的ＰＣよりかなり前にフェッチされることであり、即ち、このＰＣは、オーバーフローを生じさせた命令のものではない。フェッチと割り込みとの間の遅延の長さは、一般に、予想できない量である。この予想できない事象分布は、事象を特定の命令に適切に帰属させることを困難にする。順序ずれ及び予測的実行は、この問題を増幅するが、これは、Ａｌｐｈａ２１１６４プロセッサのようなインオーダー（順序正しい）マシンにも存在する。例えば、Ａｌｐｈａ２１１６４（インオーダー）プロセッサ対ペンチウム・プロ（アウトオブオーダー）プロセッサに対してＤキャッシュ基準事象カウントを監視しながら、性能カウンタ割り込みハンドラーに与えられるプログラムカウンタ値を比較する。例示的プログラムは、ランダムメモリアクセス命令、例えば、ロード命令と、それに続く、ナルオペレーション命令（ｎｏｐ）のハンドラーとを含むループより成る。

【００１３】

インオーダー型のＡｌｐｈａプロセッサでは、全ての性能カウンタ事象（例えば、キャッシュミス）は、事象の６サイクル後に実行される命令に帰属され、ロードアクセス後の７番目の命令においてサンプルの大きなピークを生じる。このスキューした事象分布は、理想的なものではない。しかしながら、単一の大きなピークがあるために、静的な分析は、時々、このピークから後方に作用し、その事象を生じさせた実際の命令を識別することができるが、これは、非常に単純なプログラムに対する最良の推測以上のものは何もない。アウトオブオーダー型のペンティウム・プロで実行される同一のプログラムの場合に、事象サンプルは、次の２５個の命令にわたって広く分布され、スキューを示すだけでなく、著しい不鮮明さも示す。サンプルの広い分布は、特定の事象を、その事象を生じた特定の命令に帰属させるのをほぼ不可能にする。他のハードウェア事象をカウントするときにも同様の振る舞いが生じる。

【００１４】

スキュー又は不鮮明さのある事象サンプル分布に加えて、従来の事象カウンタは、付加的な問題で悩まされている。通常、事象カウンタより多くの当該事象があり、全ての当該事象を同時に監視することは、不可能でないまでも、困難である。プロセッサの複雑さが増すと、この問題が一層悪化する。

加えて、事象カウンタは、事象が発生したという事実しか記録せず、その事象に関する付加的な状態情報を与えない。多数の種類の事象に対し、キャッシュミス事象にサービスする待ち時間のような付加的な情報が極めて有用である。

更に、公知のカウンタは、一般に、事象をコードの「ブラインドスポット」に帰属させることができない。ブラインドスポットとは、割り込み権が与えられるまで事象が確認されないために、高優先順位システムルーチン及びＰＡＬコードのような割り込み不能コードである。そのときまでに、プロセッサの状態は著しく変化し、おそらく偽の情報を与える。

【００１５】

ストール対ボトルネック

パイプライン式のインオーダープロセッサにおいて、パイプライン段で１つの命令がストールすると、その後の命令がそのパイプライン段に通過することが妨げられる。それ故、インオーダープロセッサでは「ボトルネック」命令を識別することが比較的容易であり、即ちボトルネック命令は、パイプラインのどこかでストールする傾向がある。インオーダープロセッサの場合、命令が各パイプライン段を通るときにその待ち時間を測定し、そしてその測定された待ち時間を、各パイプライン段におけるその命令の理想的な待ち時間と比較することにより、ストールを識別することができる。命令は、ある段を通過する最小待ち時間より長い時間を必要とするときに、その段においてストールしたと仮定することができる。

【００１６】

しかしながら、アウトオブオーダープロセッサでは、あるパイプライン段でストールした命令に対して他の命令がそのパイプライン段を通過することがある。実際に、ストールした命令の付加的な待ち時間は、他の命令の処理によって完全にマスクされ、実際に、ストールした命令は、観察されるプログラム完了を遅延しないことがある。

インオーダーシステムにおいても、あるパイプライン段のストールは、別のパイプライン段がボトルネックであるときにはプログラムの全実行時間に影響しない。例えば、メモリ集中のプログラムの実行中には、Ｄキャッシュミスにより遅延される実行ユニットからの「バックプレッシャー」のために、命令パイプラインのフェッチ手段及びマップ手段がしばしばストールすることがある。

【００１７】

理想的には、キャッシュミスを生じるメモリオペレーションを一次ボトルネックとして分類する。フェッチ手段及びマップ手段のストールは、実際には、キャッシュミスによる遅延の非兆候状態であり、即ち二次ボトルネックである。

ストールが他の命令によりマスクされない命令を識別し、そしてそれらを真のボトルネックとして識別することが望ましい。更に、プログラムの振る舞いを改善するためには、非兆候（二次）ボトルネックよりもカジュアル（一次）のボトルネックに焦点を合わせることが必要である。このようにパイプライン段のボトルネックをカジュアル及び非兆候と分類することは、パイプラインの状態並びにフライト中命令のデータ及びリソース依存性を詳細に知ることが必要であるが、これらは、良く知られたように、単純な事象カウンタから得ることができない。

【００１８】

１９９２年９月２９日付のウェスコット氏等の「命令サンプリング手段(Instruction Sampling Instrumentation)」と題する米国特許第５，１５１，９８１号は、アウトオブオーダーの実行マシンにおいて命令ベースのサンプリングを行うハードウェアメカニズムを提案している。ウェスコット氏等の解決策には多数の欠点がある。第１に、この解決策は、サンプリングされるコードの長さ及びサンプリングレートに基づいて命令サンプルの流れをバイアスし得る。第２に、このシステムは、リタイアした命令のみをサンプリングし、フェッチした全ての命令をサンプリングするのではなく、その幾つかがアボートされる。第３に、ウェスコット氏等のメカニズムにより収集される情報は、例えば、キャッシュミスのような個々の事象属性に集中するが、命令間の関係を決定するための有用な情報を与えるものではない。

【００１９】

最近、「ロード通知(informing loads) 」と称するハードウェアメカニズムが提案されている。これについては、１９９６年５月２２日のプロシーディングズ第２３アニュアルインターナショナルシンポジウム・オン・コンピュータアーキテクチャー、第２６０－２７０ページに掲載されたホロイッツ氏等の「インフォームドメモリオペレーション：近代的なプロセッサにおけるメモリ性能フィードバックの供給(Informed memory operations: Providing memory performance feedback in modern processors)」を参照されたい。この場合は、メモリオペレーションに続いて、そのメモリオペレーションがキャッシュにおい

てミスした場合及びその場合にのみ条件分岐オペレーションを行うことができる。プロファイリングについては特に設計されていないが、このメカニズムは、特にＤキャッシュミスの事象情報のみを収集するのに使用できる。

【００２０】

キャッシュミスルックアサイド（ＣＭＬ）バッファと称する他の特殊なハードウェアにおいては、高いレベル２のキャッシュミスレートに悩まされる仮想メモリページが識別される。この詳細な説明については、１９９４年１０月４日のプロシーディングズ・オブ・ザ・シックスス・インターナショナルコンファレンス・オン・アーキテクチャルサポート・フォア・プログラミングランゲッジ・アンド・オペレーティングシステム、第１５８－１７０ページに掲載されたバーシャド氏等の「大型の直接マップ式キャッシュにおける競合ミスの動的な回避(Avoiding conflict misses dynamically in large direct-mapped caches)」を参照されたい。

【００２１】

インテル社のペンティウムのようなプロセッサは、分岐プレディクタの分岐ターゲットバッファ（ＢＴＢ）の内容をソフトウェアで読み取ることができる。ソフトウェアでＢＴＢを周期的に読み取ることにより、コンテ氏等は、プログラムの限界実行頻度を推定するための非常にオーバーヘッドの低い技術を開発した。これについては、１９９４年１１月３０日のプロシーディングズ・オブ・第２７アニュアルインターナショナルシンポジウム・オン・マイクロアーキテクチャ、第１２－２１ページに掲載された「プロファイル駆動の最適化をサポートするための分岐ハンドリングハードウェアの使用(Using branch handling hardware to support profile-driven optimization) 」を参照されたい。

【００２２】

この解決策は、関連サンプリング情報を記憶する「プロファイル記録」に含まれた分岐方向情報を追跡することにより得られるものと同様の情報を形成する。最近、コンテ氏等は、分岐が実行される回数及び実行されない回数をカウントするプロファイルバッファと称する付加的なハードウェアの断片を提案している。これについては、１９９６年１２月２日のプロシーディングズ・オブ・第２９アニュアルインターナショナルシンポジウム・オン・マイクロアーキテクチャー、第３６－４５ページに掲載された「プロファイルバッファを使用する正確且つ実際的なプロファイル駆動の編集(Accurate and practical profile-driven compilation using the profile buffer)」を参照されたい。

【００２３】

【課題を解決するための手段】

本発明によれば、プロセッサのオペレーションを測定するための装置及び方法であって、従来のメカニズムとは異なる装置及び方法が提供される。事象をカウントし、そして事象カウンタがオーバーフローしたときにプログラムカウンタをサンプリングするのではなく、本発明の装置及び方法は、命令をランダムに選択し、そしてその選択された命令に対して詳細な状態情報をサンプリングすることに依存する。

周期的に、プロセッサの動作中に、プロファイリングされるべき命令がランダムに選択され、そして命令の実行中に何が起きたかのプロファイル記録がプロセッサの内部プロファイルレジスタのセットに累積される。選択された命令の処理が終了し、例えば、命令がリタイアし、アボートし又はトラップした後に、割り込みが発生される。パイプラインにおいて命令がいかに処理されたかの詳細を特徴付ける記録情報を内部プロファイルレジスタからソフトウェアによりサンプリングすることができる。

【００２４】

プロファイルレジスタは、命令の実行に関する多数の有用な事実を記録することができる。性能情報は、例えば、選択された命令が実行パイプラインの各段において費やしたサイクルの数、即ち段の待ち時間、命令がＩキャッシュ又はＤキャッシュミスを受けたかどうか、メモリオペランドの有効アドレス又は分岐／ジャンプターゲット、そして命令がリタイア又はアボートされたかどうかを含むことができる。

順序正しく実行する（インオーダー型）プロセッサにおいては、サンプルされた命令のフ

ェッチ－リタイア待ち時間が与えられたときに各命令に起因する全ストールサイクル数を推定することができる。これは、１つのストールした命令が別のストールした命令とオーバーラップすることがないので、ボトルネックを識別するのに充分である。

【００２５】

順序ずれして実行する（アウトオブオーダー型）プロセッサにおいては、ほとんどのストールがおそらくオーバーラップし、そしてそのストールした命令の周りで順序ずれして発生される他の命令によりマスクされる。これは、ストールした命令の識別を困難なものにする。更に、ボトルネックを識別するためには、各命令が実行される間に同時性の平均レベルに関する情報を収集することが必要となる。

特殊目的のハードウェアは、プロファイリングされた命令が実行される間に発生する命令の数をカウント及び記録して、同時実行のレベルを測定することができる。しかしながら、これは、発生するがアボートされ、従って、リタイアしない命令を考慮に入れるものではない。そこで、有用な同時性の量の測定値が与えられる。有用な同時性は、並列に発生しそして所与の命令で首尾良くリタイアする命令の平均数である。発生するがその後にアボートされる命令は、有用ではない。従って、ストールが有用な同時性によりマスクされない命令をボトルネックとして分類することができる。この別の方法を説明するために、アウトオブオーダープロセッサにおいて性能ボトルネックの位置を正確に示すための重要なメトリックは、所与の命令が実行される間に費やされた発生スロットの数である。

【００２６】

従って、有用な同時性を測定するために、「対ごとのサンプリング(pair-wise sampling)」と称する技術が提供される。基本的な考え方は、ネスト形態のサンプリングを実行することである。ここでは、第１のプロファイリングされた命令と同時に実行できる命令のウインドウが動的に定義される。命令のウインドウからプロファイリングするために第２の命令がランダムに選択される。プロファイリングされた及び第２の命令は、プロファイル情報を収集できるところのサンプル対を形成する。

対ごとのサンプリングは、各命令に起因する費やされた発生スロットの数を容易に決定すると共に、ボトルネックの位置を既知の技術よりもかなり正確に指示する。一般に、対ごとのサンプリングは、非常に融通性があり、種々様々な当該同時性及び利用メトリックを決定することのできる分析の基礎を形成する。

【００２７】

より詳細には、プロセッサのパイプラインにより処理される１つ以上の命令を周期的に且つランダムに選択し、そして実行パイプラインの段を経て命令が進行する間にプロファイル情報を収集するための装置及び方法が提供される。高レベルのソフトウェアは、次いで、この情報を種々の仕方で後処理することができ、例えば、同じ命令の多数の実行から情報を収集することにより後処理することができる。

捕獲することのできる情報は、例えば、命令のアドレス（プログラムカウンタ即ちＰＣ）、命令が命令キャッシュミスを受けたかどうか、及びミスにサービスするために被る待ち時間を含む。命令がメモリオペレーションを実行する場合には、命令がデータキャッシュミスを受けたかどうか決定し、そしてメモリ要求を満足するための待ち時間を測定する。更に、命令が各パイプライン段において費やす時間の長さを測定することができる。又、プロファイル情報は、命令がリタイアしたかアボートしたかを指示すると共に、後者の場合には、どんな種類のトラップが命令の実行をアボートしたかも指示することができる。

【００２８】

命令が実行パイプラインを経て進行するときにプロファイリングレジスタのセットに情報が収集される。命令の実行が終了すると、それがリタイアするか又はアボートするために、上位レベルのソフトウェアに割り込みが与えられる。次いで、ソフトウェアは、プロファイリングレジスタに存在する情報を種々の方法で処理することができる。

サンプリングされる性能情報は、プロファイルで指示される最適化にとって非常に有用であるが、事象の発生を集合的にカウントするようなハードウェア事象カウンタとしても多数の使い方がある。

ここに開示する技術は、既存の性能監視ハードウェアに対する改良であり、そして命令を順序ずれして発生できる近代的なマイクロプロセッサにおいて比較的低いハードウェアコストで効率的に実施することができる。

【００２９】

より詳細には、複数の処理段を有するコンピュータシステムのパイプラインで処理される命令間の相互作用の特性の統計値を推定する方法が提供される。

パイプラインの第１段へと命令がフェッチされる。フェッチされた命令から命令のセットがランダムに選択され、その選択された命令のセットのサブセットが互いに同時に実行される。選択された命令のセット間の距離が特定され、そして選択された命令のセットがパイプラインにより処理される間にコンピュータシステムの状態情報が記録される。

記録された状態情報は、ソフトウェアへ通信され、そこで、選択された命令の複数のセットに対して統計学的に分析されて、選択された命令のセット間の相互作用の統計値を推定する。

【００３０】

【発明の実施の形態】

システムの概要

図１は、ここに開示するサンプリング方法及び装置を使用することのできるコンピュータシステム１００を示す。このシステム１００は、バスライン１４０で接続された１つ以上のプロセッサ１１０、オフチップメモリ１２０及び入力／出力インターフェイス（Ｉ／Ｏ）１３０を備えている。プロセッサ１１０は、例えば、デジタルイクイップメント社のＡｌｐｈａ２１２６４プロセッサのように、集積半導体チップにおいて、機能的実行ユニットを含む多数の実行パイプライン１１１、命令キャッシュ（Ｉキャッシュ）１１２及びオンチップデータキャッシュ（Ｄキャッシュ）１１３として実施することができる。又、プロセッサチップ１１０は、以下に詳細に述べるように、選択された命令に対してプロセッサ状態をサンプリングするためのハードウェア１１９も備えている。

【００３１】

オフチップメモリ１２０は、汎用キャッシュ（Ｂキャッシュ又はＳＲＡＭ）１２１と、揮発性メモリ（ＤＲＡＭ）１２２と、永続的メモリ（ディスク）１２３とを含むハイアラーキー構成をとることができる。Ｉ／Ｏ１３０は、システム１００に対してデータを入力及び出力するのに使用できる。

【００３２】

オペレーション

システム１００のオペレーション中に、ソフトウェアプログラムの命令及びデータがメモリ１２０に記憶される。命令及びデータは、既知のコンパイラー、リンカー及びローダー技術を使用して従来のやり方で発生される。命令及びデータは、キャッシュ１１２－１１３を経て１つのプロセッサ１１０の実行パイプライン１１１に転送される。パイプラインにおいて、命令が実行のためにデコードされる。ある命令は、データに作用する。他の命令は、プログラムの実行流を制御する。

命令を実行しながら詳細な性能データを収集することが所望される。性能データは、メモリオペレーション及び実行流に関連付けることができる。

【００３３】

プロセッサパイプライン

図２ａは、図１の１つのプロセッサ１１０の実行パイプライン２００を示すもので、これは、例えば、フェッチ、マップ、イッシュー、実行及びリタイアユニット、各々、２１０、２２０、２３０、２４０及び２５０としてシリアルに構成された複数の段を有する。パイプライン２００が情報（データ及び命令）を処理するレートは、ライン２０１上のシステムクロック信号、即ちいわゆるクロック「サイクル」により制御される。

各クロックサイクルは、パイプライン２００の段が個々の量の処理を実行できるときの「スロット」即ち時間間隔を定義する。処理スロットは、通常、順方向命令を搬送し、そして以下に述べる実行ユニットの場合は、以下一般に「データ項目」と称するデータを搬送

する。例えば、分岐予想ミス又はキャッシュミス或いはパイプラインストールのような場合には、クロックはサイクルを続けるが、有意義な命令は順方向に送られない。

【００３４】

１つの効果として、本発明の装置及び方法は、「廃物(garbage) 」即ち非有効データを搬送するプロセッサスロットに関する状態情報をサンプリングすることができる。これらは、「浪費(wasted)」スロットとして知られている。浪費スロットを識別しそしてサンプリングすることは、タスクを最適化するための重要な先駆手段である。というのは、浪費スロットは、有効に機能せず、従って、システム性能を低下するからである。それ故、一般に、ここでサンプリングされるものは、公知技術のように単なる「事象」又は「命令」ではなく、プロセッサスロットが有効な命令に関連したものであるか無効の命令に関連したものであるかに関わりなくパイプライン２００を経てプロセッサスロットをプッシュすることに関連した状態情報をである。

【００３５】

フェッチユニット

Ｂキャッシュ１２１は、データ項目を各々Ｉキャッシュ１１２及びＤキャッシュ１１３に転送する。フェッチユニット２１０は、仮想アドレスを物理的アドレスへと解析するためのある形式の変換ルックアサイドバッファ（ＴＬＢ）２０５を使用して、実行されるべき次の命令をＩキャッシュ１１２からフェッチする。Ｉキャッシュ１１２からフェッチされる項目は、一般的に、実行可能な命令である。しかしながら、これらは、Ｉキャッシュが「廃物」データ即ち非命令をミスする場合のように、無効命令でもよい。

単一のプロセッササイクル中に「命令」のセットがフェッチされるのが好ましい。このセットは、例えば、４つの命令を含むことができる。換言すれば、パイプライン２００は、４スロット巾である。スロットの数は、使用可能な実行ユニットの数に基づく。他の形式のプロセッサは、単一プロセッササイクル中により少数の又はより多数の命令をフェッチすることができる。一般に、これは、各サイクルがキャッシュから４つの処理スロットを満たすことを意味する。あるスロットは、Ｉキャッシュ１１２が使用可能なデータをもたないときに浪費される。全ての処理を休止、停止するのではなく、スロットはいかなる場合にも順方向に搬送されて、サンプリングの目的で使用できるようにされるが、スロットの廃物「命令」は、実行のために発生されることがない。

フェッチ中に、選択された命令は、サンプリング又はシステムプロファイリングを許すために付加的な情報で増強することができる。増強命令は、図４を参照して以下に説明する。他の実施においては、選択された命令の増強が、イッシューユニット２３０を含むプロセッサのいかなる段でも実行できることに注意されたい。

【００３６】

マップユニット

システム１００では、パイプライン２００の次の段のマップユニット２２０を用いて命令のオペランドが物理的レジスタに動的に指定又は「マップ」される。マップユニットは、物理的レジスタをアーキテクチャー即ち「仮想」レジスタに指定する。換言すれば、仮想レジスタと物理的レジスタとの間には１対１の対応がなくてもよい。

【００３７】

イッシューユニット

次の段において、フェッチされた命令は、イッシューユニット２３０によって順序付けされる。イッシューユニット２３０は、実行されるべき次の命令のための待ち行列ヘッド(a head-of-the-queue) エントリ２３１を有するイッシュー待ち行列を備えている。命令に必要なリソースが使用できないために、イッシューユニット２３０の１つ以上の命令がストールされ得ることに注意されたい。それ故、ストールされた命令の「周り」で待ち行列２３０から他の保留中命令が順序ずれして発生される。正しい実行順序は、以下に述べるリタイアユニット２５０で確認される。

【００３８】

実行ユニット

命令は、機能的実行ユニット（Ｅ０・・・Ｅ３）２４１及びｌｄ／ｓｔユニット２４２へ発生される。実行ユニット２４１の各々は、特定形式のオペレータコード（ｏｐコード）、例えば、整数及び浮動小数点演算、分岐及びジャンプ命令等で命令を取り扱うように設計される。ｌｄ／ｓｔユニット２４２は、メモリアクセス命令を実行し、例えば、Ｄキャッシュ１１３に対してデータをロード及び記憶する。ｌｄ／ｓｔユニット２４２は、長い遅延を経験するために特別に識別される。又、長い待ち時間を伴うメモリアクセス命令は、スループットを改善するために、データがプロセッサに送り込まれるかなり前に「完了」となる。

【００３９】

リタイアユニット

命令の実行の終了は、リタイアユニット２５０により処理される。リタイアユニット２５０は、処理状態をコミットする。ある命令は、アボートするか、又はとラップされることに注意されたい。例えば、実行流は、命令がフェッチされた後に変化するか、又は命令は、例外トラップを被ることがある。このような場合に、パイプラインに既にある命令及び全ての後続命令は破棄され、そして推測的処理状態がロールバックされる。ここでの１つの効果として、破棄又は「アボート」された命令も、浪費プロセッサスロットと同様にプロファイリングされる。換言すれば、終了とは、完全に実行された有効命令をリタイアし、部分的に実行された有効命令を後処理し、或いは無効命令又は浪費スロットを破棄することを意味する。

【００４０】

本発明の技術の根底にある基本的な考え方は、パイプライン２００の段を経て進むときに、選択された「スロット」、主として命令において「データ項目」の処理を行うものである。プロファイリングハードウェアは、詳細な状態情報を動的に収集する。状態情報は、いずれのパイプライン段からでも又はシステム１００のどこからでも到来することができ、例えば、第１及び第２レベルキャッシュ又は他のサブシステムから到来することができる。状態情報は、特定事象に直接起因し得る。

ここでの設計戦略は、プロファイル記録において静的に決定することが困難な情報を収集することである。これは、プロファイル記録を性能ツールとして又はプロファイルで指令される最適化として有用なものにするか、或いはサンプリング及び分析に直接応答する動的な調整を含むオペレーティングシステム及びアプリケーションレベルソフトウェアにおけるリソース割り当てポリシー判断を行う上で有用なものにする。本発明の方法及び装置は、実際の機能的システムにおいて作用するよう設計されることを想起されたい。

【００４１】

プロファイル記録の一部分としてセーブするのにどんな状態情報に関心があるかを決定するために、図２ｂに示すように、近代的なアウトオブオーダーマイクロプロセッサのパイプライン２００の種々の段に理論的に得られる情報を検査することが有用である。

図２ｂに示すように、パイプラインの段は、フェッチ２１０、マップ２２０、イッシュー２３０、実行２４０及びリタイア２５０である。これらの段のいずれかの間に、特定の実施形態に基づき、パイプライン２００で処理されるいずれかの「フライト中」命令２０２をライン５１２によりサンプリングのために選択することができる。この選択は、カウンタ５１０の値により制御される。カウンタの値は、ライン（ｉｎｉｔ）により初期化することができる。

【００４２】

命令アドレス（ＰＣ）２８１、分岐経過ビット（ＨＩＳＴ）２８２、段の待ち時間２８３、分岐実行指示（Ｔ）２８７、データアドレス（ＡＤＤＲ）２８４、データミス（ＭＩＳＳ）２８５及びリタイア状態２８６のような状態情報は、ライン２８８においてサンプリングすることができる。選択された命令の処理が終了すると、ライン２８９に割り込み信号を発生することができる。割り込み信号２８９は、ソフトウェアでライン２９９を経て状態情報２８１－２８６をサンプリングすることができるようにする。或いは又、ソフトウェアは、内部プロセッサレジスタ５４１を経てライン２８９をポーリングすることもで

きる。

【００４３】

スーパースカラーのアウトオブオーダープロセッサアーキテクチャー

アウトオブオーダー実行プロセッサは、正しい順序で命令をフェッチ及びリタイアするが、それらのデータ依存性に基づいて命令を実行する。命令は、それがフェッチされたときから、それが終了するまで、例えば、リタイア又はアボートするまで、「フライト中」であると言える。命令は、マッピングの後、イッシューユニット２３０に入れられ、そして入力オペランドを保持するレジスタが更新されるまでそこで待機する。

【００４４】

各プロセッササイクルごとに、フェッチユニット２１０は、命令キャッシュ１１２から命令のセットをフェッチしてデコードする。フェッチユニット２１０の一部分である命令デコーダは、フェッチされたセットの中のどの命令が命令流の一部分であるかを識別する。フェッチすべき次の命令のプログラムカウンタ（ＰＣ）を分析するには多数のサイクルを必要とするので、次のＰＣは、フェッチユニット２１０の一部分である分岐又はジャンププレディクタにより予想される。予想が間違っている場合には、プロセッサは、その予想ミスした命令、即ち「不良」経路においてフェッチされた命令をアボートし、そして「良好」経路においてフェッチ命令を再スタートする。

命令を順序ずれして実行できるようにするために、レジスタはマップユニット２２０により動的に名前が付け直され、「読み取り後の書き込み」及び「書き込み後の書き込み」競合を防止する。同じ仮想レジスタに書き込む２つの命令は、順序ずれ状態で安全に実行することができる。というのは、それらは、異なる物理的レジスタに書き込みするのであり、そして仮想レジスタの消費者が適切な値を得るからである。命令は、正しい順序でフェッチされ、マップされそしてリタイアされるが、順序ずれ状態で実行することができる。

【００４５】

レジスタマップユニット２２０は、フェッチされた命令のオペランドを有効な物理的レジスタに指定する。即ち、レジスタオペランドの仮想名は、プロセッサの物理的なレジスタスペースに対して名前付けし直される。次いで、命令は命令待ち行列２３０へ送られ、そこで、実行の前に２つの事象を待機する。第１に、それらのレジスタ依存性を分析しなければならない。第２に、命令に必要なリソース、例えば、実行ユニット、レジスタ、キャッシュポート、メモリ待ち行列等が使用できねばならない。これは、現在マップされたいかなる命令に対しても、必要なリソースを再割り当てできないことを意味する。

ある命令に対してこれら２つの条件が満たされると、命令オペランドが物理的レジスタファイルにおいて探索される。次いで、オペランドレジスタの内容及び命令に関するある情報が適当な実行ユニット２４０へ送られて実行される。命令が実行を終了し、そして命令がプロセッサにおいて最も古い「非リタイア」命令であるときに、命令がリタイアする。これは、命令により使用されるリソース、例えば、物理的レジスタ及びキャッシュポートを解放する。

【００４６】

多数の事象が命令の実行を遅延させることがある。パイプラインの前方では、フェッチユニット２１０がＩキャッシュ１１２のミスによりストールするか又はフェッチユニット２１０が予想ミス経路の命令をフェッチすることがある。マップユニット２２０は、空きの物理的レジスタの欠乏、又はイッシューユニット２３０における空きスロットの欠乏によりストールすることがある。

イッシューユニット２３０における命令は、それらのレジスタ依存性が満足されるのを待機するか、又は実行ユニット２４０が使用できるのを待機する。命令は、Ｄキャッシュにおけるミスによりストールすることがある。命令は、それらが不良経路に沿って推測的に発生されるか、又はプロセッサが不法なオペレーション又はメモリアドレスのような割り込みを行ったためにトラップされることがある。これら条件の多くは、コンパイル時に予想することが困難であり、それらは全てシステム１００の性能を低下させる。これにより、ライン２８８に得られる情報をサンプリングすることが重要となる。

【００４７】
プロファイル情報レジスタ
それ故、図３に示すように、サンプリングされる各命令ごとにプロファイル情報を記憶するためのメモリ３００が設けられる。メモリ３００は、レジスタファイル又はバッファの形態でよい。換言すれば、サンプリングされる選択済み命令は、レジスタファイル３００で直接識別される。レジスタファイル３００は、複数のレジスタを含むことができる。或いは又、ファイル３００は、多数のフィールドをもつ単一のインデックス可能なレジスタとして実施することができる。
ファイル３００は、図２ｂのライン２８８によりパイプライン２００の要素に接続され、従って、選択された命令に関連した性能情報をパイプライン２００の各段に対して捕獲することができる。プロファイルレジスタ３００は、公知技術で見られる単純な「事象」カウンタ以上のものであり、ここでは、これらレジスタは、特定の既知の命令及び事象に起因する性能情報を収集することに注意されたい。
【００４８】
図３において、各レジスタに対して割り当てられるビットの数は、そこに記憶される情報の形式、例えば、命令アドレス（６４ビット）、サイクルカウント、即ち待ち時間（８又は１０ビット）、個別事象（１ビット／事象）等々に依存している。これらの数は単なる指針に過ぎない。他の実施形態は、種々のレジスタ３００に対して異なるビット数を使用することができ、これは設計上の選択肢である。
好ましい実施形態では、プロファイルＰＣレジスタ３１０は、選択された命令のＰＣを記憶する。以下に述べるように、プロファイリングされている命令は、アサートされた「プロファイル」ビットを有する。又、ＰＣレジスタ３１０は、選択された命令のｏｐコードを含むこともできる。更に、マルチスレッド式実行を許すプロセッサについては、レジスタ３１０の付加的なビットがスレッドの識別子を記憶することができる。レジスタ３１０の他のフィールドは、プロセス識別子、アドレススペース番号、ＣＰＵ番号、及び実行されている命令の命令番号（ｉｎｕｍ）を記憶することができる。更に、多数の論理レジスタセット、即ちハードウェアコンテクスト及び同時実行スレッドを有するプロセッサでは、レジスタ３１０がハードウェアコンテクスト及びスレッド識別子である。この情報を記憶することにより、プロファイル情報を特定の命令に直接起因させることができる。更に、サンプリングされた情報は、アドレスの範囲、ｏｐコード、実行スレッド、アドレススペース、等々に基づいてフィルタすることができる。
【００４９】
プロファイル有効アドレスレジスタ３２０には、選択された命令に関連したアドレスがロードされる。命令がロード又は記憶のようなメモリアクセス命令である場合には、有効な６４ビット仮想メモリアドレスが捕獲される。命令がジャンプ又は分岐である場合には、ターゲットＰＣが記録される。
本発明のサンプリング技術の１つの効果として、パイプライン２００によって処理される全ての「命令」は、サンプリングレートに関わりなく、サンプリングのために選択される確率が等しい。命令は、有効な命令、無効の命令、非割り込み命令、又は「廃物」命令である。従って、捕獲された有効アドレスは、プログラムの全体的な振る舞いを統計学的に表す。サンプリングされた命令の有効アドレスを捕獲することにより、メモリアクセス及び実行流を、実際の動的な実行に正確に関連付けることができる。
【００５０】
プロファイル事象カウンタ３３０は、例えば、１ビットフィールドに区画化される。１ビットフィールドは、選択された命令に対する事象を記録する。命令が最初に選択されるときに、レジスタがクリアされる。事象は、キャッシュミス、分岐予想ミス、リソース競合、トラップ及び例外条件、リタイア／アボート／無効、ＴＬＢミス、実行／非実行、データ依存性ストール、リソース依存性ストール、等々を含む。この実施形態では、多数の事象を単一の命令に起因させることができる。リタイア及びアボートの両命令に対して事象情報が収集されることに注意されたい。事象レジスタ３３０のサイズを減少するために、

あるビットフィールドを使用して、命令のｏｐコードに基づき異なる形式の相互に排他的な事象を記録することができる。

【００５１】

プロファイル経路レジスタ３４０は、分岐経過テーブルから最近の分岐実行／非実行情報を捕獲するのに使用される。分岐経過テーブルは、他の用途に対して良く知られている。グローバルな分岐実行経過は、選択された命令をフェッチした実行経路を指示するのに使用できる。命令は、この情報を有効なものにするために分岐命令である必要はないことに注意されたい。経路情報の使用は、以下で詳細に説明する。

待ち時間レジスタ３５０は、選択された命令が、例えば、パイプライン２００の種々の段間をフライト中である間に、チェックポイントにおいて得られたタイミング情報を記憶する。チェックポイントは、命令がストールされて、ある事象又はリソースを待機する場所に基づいて、プロセッサごとに異なる。各待ち時間レジスタ３５０は、２つのチェックポイント間で命令が費やすサイクル数をカウントする。

【００５２】

選択された命令がチェックポイントを通過し、即ちパイプライン２００の次の段に入るときに、それに対応する待ち時間レジスタ３５０が最初にクリアされ、そして１サイクル当たり１回増加され、やがて、命令が次のチェックポイントを通過し、次の待ち時間レジスタが初期化されそしてカウントを開始する。待ち時間レジスタ３５０の数は、特定の実施形態におけるパイプライン２００の段数に基づく。命令がアボート又はリタイアするときには、待ち時間レジスタ３５０に完全な待ち時間プロファイルが記憶される。

収集すべき潜在的に有用な待ち時間のリストは、フェッチ対マップ、マップ対データレディ、データレディ対実行、実行対リタイアレディ、リタイアレディ対リタイア遅延を含む。メモリ命令（ロード及び記憶）の場合、待ち時間は、イッシュー対完了である。この最後の待ち時間は、あるメモリオペレーションは、それらが作用するデータが実際にプロセッサに送られる前にリタイアの準備ができるという点で、他の待ち時間とは異なる。これらの待ち時間は、レジスタ３５０で直接カウントすることもできるし、或いはレジスタが生のサイクルカウントを収集することもでき、この場合に、プロファイリングソフトウェアは、次々の段に対する生のカウント間の差を計算して、実際の待ち時間を決定する。例えば、パイプライン待ち時間クロックサイクルをカウントする回路は、図６を参照して以下に詳細に説明する。

【００５３】

レジスタ３００における情報の更新は、遅延が受け入れられた直後に行う必要はない。必要とされるのは、選択された命令が完了した（リタイア又はアボートした）ことを知らせる割り込みを、レジスタファイル３００の全ての情報が更新されるまで遅延するか、或いは割り込みハンドラーを、プロファイルファイル３００が更新されるまでストールできるようにすることだけである。

プロファイルレジスタファイル３００を複写できることに注意されたい。プロファイルレジスタファイルの多数のコピーがある場合には、シリアルに又は同時にプロファイリングするために多数の命令を選択することができる。この場合には、各選択された命令が、以下に述べるように、特定のレジスタファイルで明確に識別される。オーバーヘッドの量を減少するために単一の割り込み信号に応答して多数のレジスタファイルをサンプリングすることができる。

【００５４】

増強命令

図４に示すように、各命令４００はサンプルフィールドを含む。例えば、このサンプルフィールドは、「サンプル」ビット（Ｓ）４０１と称する１ビットタグである。サンプルビット４０１がアサートされると、サンプリングのために命令が選択される。ビット４０１をアサートすると、プロファイル情報を収集するサンプリングハードウェアが作動されると共に、選択された命令が完了した（リタイア又はアボートされた）ときに割り込みを生じさせる。或いは又、フェッチされた各「命令」を「ｉｎｕｍ」値で連続的に番号付けす

ることもできる。この場合には、特定のｉｎｕｍ値をもつ命令を選択することができる。命令を選択するメカニズムについては、以下に述べる。

【００５５】

プロファイルレジスタファイル３００は、フィールドが更新されそして割り込み信号が発生されたときに読み取ることができる。割り込み信号は、特権付きのプロファイリングソフトウェア（ＰＳＷ）がプロファイルレジスタ３００の内容を処理できるようにする。多数のサンプルが記録される場合には、単一の割り込みで、多数の選択された命令に対して性能データをサンプリングできることに注意されたい。

実施形態に基づき、増強命令４００は、次の付加的なフィールド、即ち３つまでの命令オペランド（ｏｐ１、ｏｐ２及びｏｐ３）４１１－４１３と、プログラムカウンタ（ＰＣ）４２０と、オペレータコード（ｏｐコード）４３０とを含むことができる。有効フィールド（Ｖ）４３１は、１ビットフィールドを真又は偽にセットすることにより、選択されたスロットにおける「命令」が有効であるかどうか指示することができる。フィールド４４０及び４５０は、命令に関連したＩキャッシュ及びＴＬＢミスを各々指示するために指定することができる。単一の命令が多数のオペランドを含み得るので、その命令に対して多数のミスが考えられることに注意されたい。

【００５６】

プロファイルレジスタファイルＩＤ

若干複雑な設計では、多数の命令を同時にプロファイルすることができる。この実施形態では、複数のレジスタファイル３００、或いはサブフィールドを伴う単一の大きなレジスタがあり、ファイル３００の数は、同時にプロファイルすることのできるフライト中命令の数に対応する。このケースを取り扱うために、命令４００は、サンプルレジスタファイル識別子（ＩＤ）フィールド４０２も含むように増強される。これは、多数のレジスタファイル３００の１つにプロファイル情報を直接リンクできるようにする。上記したように、ここでは、選択された命令とプロファイルレジスタとの間に直接的な関連がある。それ故、レジスタに収集されるプロファイル情報は、特定の命令に直接起因し得る。

一度に１つのフライト中命令しかプロファイリングされないときでも、ファイル即ちレジスタ３００をＩＤフィールド４０２でインデックスして、プロファイリングソフトの割り込みハンドラーのコストを多数の命令サンプルにわたり償還できるようにするのが有用である。命令セット内の命令が選択された命令であるかどうかを決定することは、「ワイヤドＯＲ」オペレーションを用いて行うことができる。

【００５７】

ランダムサンプリング

本発明のプロファイリングのオーバーヘッドは、同時にプロファイリングすることのできる命令の数を制限することにより減少され、例えば、ビット４０１がセットされる。プログラム又はプログラムの一部分において各命令をプロファイリングするのではなく、ここでは、プロファイリングされるべき命令が、プロセッサパイプライン２００の特定の段階中に、例えば、フェッチの間に選択され、そしてその選択された命令がサンプルビット４０１のアサートによりタグ付けされる。サンプルビット４０１がアサートされた場合には、パイプライン２００の要素がプロファイル情報をプロファイルレジスタファイル３００へ送る。

ここに記載する命令レベルプロファイリングをサポートする詳細について以下に述べる。

【００５８】

フライト中状態

第１に、プロセッサパイプライン２００を通過する各デコードされた命令状態は、上記のように、付加的な情報で増強される。命令は、それがフェッチされたときから、それがリタイア又はアボートするときまで、フライト中であるとみなされる。上述したように、命令は、少なくとも１つのサンプルビット４０１で増強される。サンプルビット４０１は、各フライト中命令及びキャッシュ／メモリ要求の状態の一部分である。ビット４０１がアサートされると、このビットは、この命令に対してプロファイリング情報が記録されるこ

とを示し、さもなくば、記録されないことを示す。

【００５９】

簡単な設計においては、一度に１つのフライト中命令のみが、そのサンプルビット４０１をアサートすることが許される。サンプルビット４０１は、選択された命令に対し、その命令がリタイアするか又はアボートされるまで、アサートされたままとなる。多数のレジスタファイル３００をもつ更に複雑な設計では、多数のフライト中命令を個々にプロファイリングすることができ、そして付加的なビットをアサートすることができる。

プロファイルされた命令の選択及びサンプリング

フェッチ段の実施について図５に示したように、プロファイリングされるべき命令の選択及びプロファイル情報のサンプリングは、次のように行われる。フェッチカウンタ５１０は、例えば、特権付きプロファイリングソフトウェア（ＰＳＷ）５２０によりライン５１１を経て初期化される。ＰＳＷ５２０は、所定サイズを有する値の間隔からランダムに選択された値でカウンタ５１０を初期化することができる。従って、サンプリングされた命令は、命令の実行における特定のパターンと相関しない。間隔のサイズは、サンプリングの平均頻度を決定する。カウンタ５１０の値を初期化するための他のランダム化技術（ハードウェアを含む）も使用できる。

【００６０】

例えば、公知技術の場合のように命令が固定頻度でサンプリングされるときのように、ランダムサンプリングが行われないと、例えば、システム１００の収集オペレーションのように、フェッチされた全ての命令の統計学的に正しいプロファイルを発生することができない。これは、サンプリングレートに対して比較的重要でない多数の命令を含む実行ループを有する実行スレッド、例えば、命令を有しそしてサンプリング間隔が６５５３６個の命令であるループに対して、特に言えることである。他の通常のサンプリングも同じ問題を有する。そこで、２つの命令の一方のみからのサンプルが常に収集される。１つの効果として、ランダムに選択された命令は、サンプリング間隔の長さとは独立した相関を発生する。各命令４００がフェッチされるたびに、カウンタ５１０がパイプライン２００のフェッチユニット２１０によりその初期値から増加されるか、或いは別の実施形態では、減少される。カウンタ５１０が、その実施形態に基づいて、オーバーフローするか又はアンダーフローしたときに、現在フェッチされた命令がそのサンプルビット４０１をアサートし、そしてＩＤフィールド４０２は、多数の命令がサンプリングのために選択されたときにも初期化することができる。

【００６１】

別の実施形態では、カウンタ５１０は、各命令がフェッチされるたびではなく各サイクルごとに増加され、例えば、カウンタ５１０は、フェッチの機会をカウントし、実際にフェッチされる命令をカウントするのではない。例えば、フェッチユニット２１０が各クロックサイクル中にＩキャッシュ１１２から４つの項目をフェッチできる場合には、４つのフェッチ機会がある。Ｉキャッシュからの１つ以上のフェッチがミスとなるか又は「不良」命令をフェッチすることがある。ミスの場合には、ミスした命令に対して使用できるスロットが「廃物」を含み、命令を無効とマークすることが必要になる。不良命令は、不良の実行経路に存在するものであるか、又はさもなくば、アボートされる。

フェッチされた命令ではなくサイクルをカウントする場合には、設計を効果的に簡単化する。フェッチされた有効な命令のみをカウントする場合には、かなり複雑なものとなる。というのは、制御流が、フェッチされた命令のグループに向かって又はそこから分岐することができ、従って、全ての命令をデコードしてどれが有効であるかを決定することが必要となり、もはや、カウンタを４だけ増加するだけの簡単なことではなくなるからである。

【００６２】

１つの効果として、サイクル中にＩキャッシュからフェッチされた全てのもの（良好な命令、不良の命令、廃物命令）をサンプリングのために選択し、Ｉキャッシュ１１２及びパイプライン２００の真の性能を決定することができる。ここでは、バイアスはなく、従っ

て、システム性能の統計学的に正しい推定値が得られる。
これは、短い固定の時間周期中に又は離間された固定の間隔で各有効な命令のみを選択する既知の技術とは区別されるものである。何れの場合にも、オーバーヘッドを最小にすることが戦略である。システム全体の性能データを捕獲することのできる技術はない。
【００６３】
命令のフィルタ動作
選択されたものは、フィルタ５０５によりフィルタすることができる。フィルタ動作は、命令ｏｐコード、オペランド、或いは例えば、ある時間周期内で第１形式の命令の後に別の形式の命令が続くといったより複雑なフィルタ基準に基づいて行うことができる。パイプライン２００への入力においてフィルタ動作を行う場合には、カウンタ５１０をリセットすることができる。これを行う方法は、多数ある。１つの方法では、カウンタ５１０の現在初期値が初期値(init)レジスタ５１３に記憶される。命令がフィルタされるときには、初期値レジスタ５１３に記憶された値がカウンタ５１０に再ロードされ、初期のランダム化選択が想起される。
【００６４】
命令が増強された後に、パイプライン２００は、図２ｂのプロファイル情報２８１－２８６をレジスタファイル３００（１つ又は複数）に供給する。リタイアユニット２５０は、命令の完了又はアボートに応答して、プロファイル情報をファイリングを完了し、そしてライン５４０に割り込み信号を発生して、ＰＳＷ５２０がプロファイル情報をサンプリングできるようにする。
或いは、ＰＳＷ５２０は、内部プロセッサレジスタ又はメモリ位置（５４１）を経てライン５４０をポーリングすることもできる。本発明の技術の１つの特徴として、公知のあるプロファイリング技術とは対照的に、たとえ本発明の技術がプロセッサにわたる状態に関する正確な情報を与えるものであっても、プロセッサのサイクルタイムに何ら影響を与えない。唯一の時間制約は、プロファイルレジスタ３００がサンプリングされる前に全てのプロファイル情報を記録しなければならないことである。
【００６５】
待ち時間カウンタ
図６は、例示的な待ち時間、フェッチ対マップ（ＦＭ）、マップ対イッシュー（ＭＩ）、イッシュー対リタイア（ＩＲ）、フェッチ対トラップ（ＦＴ）、及びイッシュー対ｌｄｓｔ（ＩＬＳ）をカウントするための回路６００を示す。この回路６００は、ライン６１１によりラッチ６２０に接続されたサイクルカウンタ６１０を備えている。
サイクルカウンタ６１０及びラッチ６２０は、ライン６０１上の信号Ｐｆｅｔｃｈにより初期化される。この信号は、プロファイリングされるべき命令がフェッチされるときに発生され、例えば、サンプルビット４０１から導出される信号である。カウンタ６１０は、ライン６０９のクロック信号により増加される。各クロック信号は、１つのプロセッササイクルに対応する。
【００６６】
命令４００がパイプライン２００の段を経て進行するときに、パイプライン２００の段遷移がライン６０２－６０６の信号、各々、Ｐｍａｐ、Ｐｉｓｓｕｅ、Ｐｒｅｔｉｒｅ、Ｐｔｒａｐ及びＰＬＳｄｏｎｅをトリガーする。対応するラッチ６２０は、図３のプロファイル待ち時間レジスタ（又はフィールド）３５０に記憶するためにライン６１２－６１６において読み取ることができる。
【００６７】
プロファイリングアプリケーション
上記のプロファイリングハードウェアは、種々の異なる方法で使用することができる。本発明の技術は、個々の命令の実行に関する非常に詳細な情報を与えるので、１つのアプリケーションで非常に多数の命令をプロファイリングすることができる。サンプル情報はメモリバッファに記憶され、プロファイリングツールにより後で処理されて、詳細な命令レベル情報を形成することができる。

この情報は、例えば、各ロード命令に対するロード待ち時間のヒストグラム、命令実行時間のヒストグラム、及びおそらくは各命令に対するパイプライン状態の適度に包括的な分析を発生するのに使用できる。この解決策により与えられる情報の量は、おそらく、かなり多くなるので、本発明の技術の全メモリオーバーヘッドも、相当の量のメモリトラフィックが含まれるために、かなり大きなものとなる。例えば、１秒当たり１０億の命令がフェッチされ、そして各１万のフェッチされる命令ごとにサンプリングが実行される場合には、プロファイル情報のデータレートが１秒当たり約２．４ＭＢとなる。
以下、プロファイル情報を収集することにより帯域巾を減少するためのソフトウェア実施方法について説明する。

【００６８】

出力プロファイル情報をフィルタすることによるデータの減少

サンプリングされるデータの量は、プロファイル記録のあるフィールド、例えば、プロファイルレジスタ３００のデータを、それらが明確に要求されるときを除いて、無視することにより、減少することができる。システム１００のユーザは、異なるレベルのプロファイリングを望むことがある。最低のオーバーヘッドモードでは、プロファイリングアプリケーションソフトウェアは、ＰＣ及びリタイアー遅延フィールドのみを用いてプログラムの全部又は一部分に対してプロファイルレポートを発生することができる。実行されるべき最適化に基づき、平均化又は他の統計学的メトリック、例えば、最小、最大又は標準偏差の計算により他のＰＣごとの(per-PC)値を要約することができる。データを処理するための更なる時間が与えられると、プロファイリングアプリケーションは、種々の命令待ち時間のヒストグラムを形成することができる。

【００６９】

有効なメモリアドレス、分岐ターゲットアドレス及び分岐経過サンプルは、おそらく、他のフィールドよりも経費のかかる処理を必要とする。これらのフィールドは、おそらく、特定の最適化タスクを実行するためにデータを収集するとき以外は無視することができる。命令と命令との間の命令間フェッチ距離がサイクルで与えられると、プロファイリングアプリケーションは、同時性のレベルに関する情報も収集することができる。
又、プロファイリング情報のフィルタ動作は、例えば、マスクレジスタ又はプログラマブルロジックのようなハードウェア手段により行うこともできる。例えば、キャッシュミスがあったとき又は命令がリタイアしたときにのみサンプルリングするか、或いはｏｐコード、オペランド、アドレス、事象及び待ち時間の他のブール組合せのみをサンプリングする。

【００７０】

ハードウェアオペレーションの決定

本発明のプロファイリング技術は、Ａｌｐｈａ２１２６４プロセッサのようなアウトオブオーダーイッシュープロセッサの内部動作の正確な理解を得るために使用することができる。この形式のマシン編成に関して注目すべき第１の事柄の１つは、パイプライン２００において命令がストールする場所が多数ありそしてストールする理由が非常に多数あることである。
例えば、ある命令は、イッシューユニット２３０においてストールすることがある。というのは、そのオペランドの幾つかがデータレディでなく、選択された命令の実行に必要なリソースの幾つかが使用できず、又はその命令に先立って他の命令が実行されるべく選択されるからである。

【００７１】

ある命令は、仮想－物理的レジスタマッピングを行うマップ段においてストールすることがある。というのは、マシンが物理的レジスタからのものであり、フライト中の命令が非常に多数あり、或いはイッシューユニット２３０がいっぱいである（実行されようとしている命令を入れる場所がないことを意味する）ためである。或いは又、ある命令は、リタイアユニットにおいてストールすることがある。というのは、プログラム順に既にイッシューされた命令がまだ完了していないからである。

命令がどこでストールされたか、なぜストールされたかそしてどれほどの時間ストールされたかを正確に決定することは、主に、その命令が実行されるときのマシンの正確な状態によって左右される。プロセッサがこのように動的であるために、ソフトウェア性能ツールでこの状態を静的に決定することは困難である。

【００７２】

オペレーションの概要

図７ａに示すように、プロファイリング方法７００は、次のステップを含むことができる。プロファイリング状態は、ステップ７１０において初期化される。ここで、レジスタがクリアされ、そしてカウンタに初期値が指定される。ステップ７２０において、命令がフェッチされそしてカウントされる。ステップ７３０において、初期化以来フェッチされた命令の数が所定のランダム数に等しいときに命令が選択される。選択された命令は、その選択を指示するよう増強される。

選択された命令が実行パイプライン２００を経て進むときに、ステップ７４０においてプロファイル情報が収集される。完了（リタイア又はアボート）時に、収集された情報がステップ７４０においてサンプリングされる。サンプリングされた情報は、その後の処理のためにバッファすることができる。又、特定のプロファイリング状態をサンプリングし、より詳細な情報を抽出することもできる。

【００７３】

処理された命令の特性の統計値の推定

図７ｂに示されたように、プロセス７９９は、パイプライン２００により処理される命令の特性の統計値を推定する。プロセス７９９は、次のステップを含むことができる。ステップ７５１は、ステップ７５０において上記したようにサンプリングされたプロファイル記録３００を読み取る。記録は、選択された命令が完了したときに読み取られる。ステップ７６０において、サンプルは、システムの状態情報を考慮するファンクション７５５に基づいて選択又は破棄される。

例えば、ファンクション７５５は、選択された命令のアドレス、プロセス識別子、アドレススペース番号、ハードウェアコンテクスト識別子、又はスレッド識別子のような状態情報７５６を入力として得る。又、ファンクション７５５は、経路識別情報、ｏｐコード、オペランド、待ち時間、又は選択された命令により経験する事象のような状態情報も使用することができる。事象情報は、リタイア／アボート／無効状態、キャッシュヒット／ミス、分岐予想ミス、トラップ状態ＴＬＢヒット／ミス、及びデータリソース依存性状態、等々である。

【００７４】

ステップ７６０は、ファンクション７５５に基づいてサンプルのサブセットを発生する。ステップ７８０において、統計値７９０が決定される。これら統計値は、サンプリングされた命令の特性の平均値、標準偏差、ヒストグラム（分布）及びエラー限界を含むことができる。例えば、特定の事象が発生する平均レートや、命令実行の平均待ち時間や、メモリアクセスがある。又、プロセス、スレッド又はハードウェアコンテクストの実行レートの平均値も決定できる。ヒストグラムは、命令実行、メモリアクセスレート又は待ち時間のような分布を示すことができる。

エラーの限界は、サンプリングされている特定の特性に対してサンプルの数の平方根の逆数で近似することができる。

【００７５】

Ｎ個ごとのサンプリング

ここに開示するプロファイリング技術は、Ｎ個ごとの(N-wise)サンプリングを実行するのにも使用できる。ここで、多数の同時実行命令間の相互作用の動的な状態を捕獲することができる。単一のフライト中命令をプロファイリングするのではなく、２つ以上の個別の命令が同時にプロファイリングされる。選択された命令間の動的な「距離」は、フェッチされた命令の数、又はフライト中の命令を「分離」するプロセッササイクルの数として測定することができる。カウンタ５１０によりカウントされる事象のいずれかを用いて、選

択された命令間の距離、例えば、クロックサイクル、フェッチされた命令等を測定することができる。
Ｎ個ごとのサンプリングされた命令に対するプロファイル情報は、多数の考えられる用途を有する。第１に、情報を分析して、有用な同時性レベルを測定することができる。これは、真のボトルネックを探索できるようにする。真のボトルネックは、長いストールが低い同時性で結合されることを特徴とする。又、Ｎ個ごとのサンプルは、経路のプロファイリングを容易にすると共に、経路に沿った少なくとも２つのポイントを含むように経路を制限することにより実行経路候補を明確化することができる。更に、Ｎ個ごとのサンプリングから、詳細なプロセッサパイプライン状態を統計学的に再構成することもできる。ここで、命令のグループの選択は、命令間のある類似性の尺度、例えば、最近の分岐経過、ストール、命令形式、又は他の最近の状態経過をベースとすることができる。

【００７６】

有効な同時性の測定

アウトオブオーダープロセッサにおいて性能のボトルネックを正確に位置決めするには、ストール時間及び同時性レベルの両方に関する詳細な情報を必要とする。インオーダープロセッサとは対照的に、長い待ち時間の命令がストールされる間にプロセッサを効率的に利用するに充分な同時性があるときには、長い待ち時間の命令が問題とならない。
同時性情報を得るための１つの解決策は、全パイプライン状態のスナップショットを得ることである。これは、同時実行命令のセットが所与の時点でパイプラインの段のどこにあるかを直接的に露呈する。しかしながら、全パイプラインの状態をサンプリングレジスタ及びバッファに「ダンプ」することは、時間及びスペースの両面で非常に経費がかかる。更に、発生される多量のデータは、おそらく、サンプリングのコストを償還するように効率的に収集することができない。更に悪いことに、この解決策は、リタイアする命令しか「有効」としてカウントされず、そしてフェッチされた命令がアボートするところの情報がまだ分からないので、実際上不充分である。

【００７７】

ネスト状の対ごとのサンプリング

Ｎ個ごとのサンプリングの１つの形式は、単一命令プロファイリングと全パイプラインスナップショットとの間の妥協を最小にする。ここで、統計学的な対ごとの(pair-wise) サンプリングがネスト状に行われ、所与の選択された命令に対して、同時に実行し得る別の命令が直接サンプリングされる。

ネスト状のＮ個ごとのサンプリングに対するハードウェアサポート

Ｎ個ごとのサンプリングは、次のハードウェア特徴を含む。第１に、ハードウェアは、少なくとも２つの同時フライト中命令に対しプロファイル情報を捕獲できねばならない。プロファイルレジスタのセットは、プロファイル記録の多数の個別のセットをサポートするために複写されねばならず、そして単一サンプルビット４０１は、より一般的なＩＤフィールド４０２へと増強されねばならない。第２に、ハードウェアは、サンプリングレートの変更により、選択された命令間の距離を動的に変更できねばならない。これは、ハードウェア又はソフトウェアによって行うことができる。同時サンプリング命令（Ｎ個ごと、但しＮ＞１）のセットのサイズは、カウンタ及びレジスタの付加的な複写でより大きくすることができる。

【００７８】

例えば、特権付きプロファイリングソフトウェア５２０は、対ごとのケースでは２つのフェッチカウンタ５１０の初期値がランダムに選択されるところの間隔のサイズを動的に変更することができる。これは、一対の命令に対するサンプル間フェッチ距離を同時に特定できるようにする。ハードウェアは、ソフトウェアレベルでの最大の融通性を得るために比較的大きなサンプル間フェッチ距離をサポートすることができる。
第２のフェッチカウンタがコアフェッチ命令カウンタ５１０と同じサイズであって、充分な距離に離れた２つの独立した命令を選択できるのが理想的である。Ａｌｐｈａ２１２６４プロセッサの場合には１０ビットカウンタで充分である。フェッチ命令をカウントする

ときに同時性を測定するには、それより小さなカウンタで充分であり、サイクルがカウントされる場合には、それより大きなカウンタが必要とされる。ネスト状のＮ個ごとのサンプリングについては、ハードウェアは、サンプル間フェッチ・対・フェッチ待ち時間もサイクルで測定して、多数の待ち時間レジスタ３５０を時間的に相関させることができねばならない。

【００７９】

ネスト状のＮ個ごとのサンプリングアプリケーション

高レベルアプリケーションソフトウェアは、ネスト状のＮ個ごとのサンプリングを用いて、有効な同時性を測定することができる。ここでの重要な考え方は、潜在的に同時に実行し得る命令セットのサンプリングを許すことである。ネスト状のサンプリングは、通常のサンプリングを正当化する同じ統計学的引数に基づくもので、即ちサンプリングが繰り返し適用される。Ｎ個ごとのサンプリングは２つのサンプリングレベルを含むので、著しく実行されるコードについては最も効果的である。明らかに、これは、最も重要なところでもある。

定義された同時性

図８に示すように、４巾のパイプラインにおける所与の選択された命令Ｉ（８１０）に対し、潜在的に同時の命令とは、ある動的な実行中に命令Ｉと共にプロセッサパイプライン２００に共存する命令である。これは、命令Ｉがフェッチされる前に種々の実行段に存在する命令と、命令Ｉがリタイア又はアボートされる前にフェッチされる命令とを含む。

【００８０】

例えば、Ａｌｐｈａ２１２６４プロセッサは、８０個のフライト中命令を許すものである。しかしながら、実際には、同時即ちフライト中命令の実数は、おそらく、ハードウェアでサポートされるピーク値より相当に小さい。他方、予想ミス又は不良経路に沿った推測的実行は、潜在的同時性のウインドウを増加することができる。

Ａｌｐｈａ２１２６４プロセッサにおいて同時性を検討するために、命令Ｉ（８１０）の周りのウインドウＷ８２０の適度なサイズが約１００個の命令を含まねばならないことが提案された。他の実施形態については、ウインドウの適当なサイズを実験で決定することができる。

例えば、約１００個の潜在的な同時命令であるサイズＷのウインドウが与えられると、選択される命令間のフェッチ距離をランダム化することにより非バイアスのサンプリングを行うことができる。例えば、対ごとの各サンプル＜Ｉ１、Ｉ２＞（８３１及び８３２）に対し、サンプル間フェッチ距離は、１とＷとの間に均一に分布した擬似ランダム数にセットされる。このように、第１の選択された命令Ｉ１と第２の選択された命令Ｉ２との間でサンプル間距離をランダムに変更すると、命令が実際に時間的に重畳するところの多量の統計学的情報が捕獲される。

【００８１】

同時重畳の分析

種々のサンプル間フェッチ距離をもつ同時選択される命令のセットに対するプロファイル情報は、有効な同時性統計値を直接的に表す。対応するサンプル情報の各セットを使用し、第１命令Ｉ１から時間的に前方にそして第２命令Ｉ２から時間的に後方に見ることにより同時性情報を決定することができる。

各Ｎ個ごとの選択された命令に対して記録されるプロファイル情報は、両命令＜Ｉ１、Ｉ２＞が所与の時間にプロセッサパイプライン２００に存在するようなインスタンスを正確に考慮する待ち時間を含まねばならない。更に、待ち時間レジスタのセットを相関させるためには、サンプル間フェッチ待ち行列が記録されねばならない。又、ネスト状のプロファイリングは、放棄した実行経路において命令＜Ｉ１、Ｉ２＞が完了したときを指示することもできる。この詳細な情報を統計学的に収集して、有効な同時性レベルを反映する種々のメトリックを形成することができる。

【００８２】

浪費イッシュースロットの測定

種々のサンプル間フェッチ距離をもつ対ごとの命令サンプル＜Ｉ１、Ｉ２＞の収集は、有効な同時性統計値を直接的に表す。対ごとの各サンプルを使用して、第１命令から時間的に前方にそして第２命令から時間的に後方に見ることにより同時性情報を計算する。命令Ｉの後にフェッチされた命令に対する性能情報を測定するために、＜Ｉ、Ｉ２＞の形態の対を考える。命令Ｉの前にフェッチされた命令に対する性能を測定するために、＜Ｉ１、Ｉ＞の形態のサンプルされた対を考える。

対ごとの各サンプル＜Ｉ１、Ｉ２＞に対して記録されるプロファイルデータは、待ち時間レジスタ３５０に記憶される値であって、各時点にプロセッサパイプライン２００のどこにＩ１及びＩ２があるかを指示する値と、２セットの待ち時間レジスタ３５０を相関させることのできるサンプル間フェッチ待ち時間とを含む。又、プロファイル記録は、対＜Ｉ１、Ｉ２＞がリタイアするかどうかも指示する。

【００８３】

この詳細な情報を統計学的に収集して、有効な同時性レベルを反映する種々のメトリックを形成することができる。例えば、命令Ｉに対する１つの関心のある同時性の尺度は、Ｉがフライト中である間に浪費したイッシュースロットの平均数である。

浪費したイッシュースロットの数は、図９に示すように決定できる。Ｉ及びＩ２がリタイアするような形式＜Ｉ、Ｉ２＞のサンプルの数をＦ１とし、サンプルと共に記録される待ち時間は、Ｉ及びＩ２の実行が重畳することを指示する（ステップ９１０）。それ故、有効な順方向重畳を伴うサンプル対の全数をカウントし、これはＦ１で表される。同様に、ステップ９２０において、Ｉ及びＩ２の両方がリタイアしそしてそれらの実行が重畳するような形式＜Ｉ１、Ｉ＞のサンプルの数をＢ１とする。即ち、第２の命令が第１の命令に対してサンプリングされる場合には順方向に、そして第１の命令が第２の命令に対してサンプリングされる場合には逆方向に、各対が２回考慮される。

【００８４】

次いで、ステップ９３０において、一致するサンプルの数Ｆ１＋Ｂ１に潜在的な同時性のサンプルウインドウのサイズWを乗算することにより、命令Ｉがフライト中である間にイッシューされる有効命令の数を統計学的に推定し、即ち形成されるイッシュースロットの数は、Ｗｘ（Ｆ１＋Ｂ１）となる。

イッシュースロットで測定される命令Ｉの累積的待ち時間Ｌ１、例えば、Ａｌｐｈａ２１２６４プロセッサで持続できる４／サイクルを付加的に決定することにより、ステップ９４０において、命令Ｉの実行中に浪費したイッシュースロット（ＷＩＳ）の全数を次のように要約することができる。

ＷＩＳ＝Ｌ１－（Ｗｘ（Ｆ１＋Ｂ１））

値ＷＩＳは、命令Ｉの実行当たりの浪費イッシュースロットの割合又は平均数を表すように容易に拡張することができる。好都合にも、この平均に寄与する値を増分的に収集し、データ収集中にコンパクトな記憶を行うことができる。又、これは、１９９７年３月３日に出願されたウエイル氏等の「プロセッサ性能カウンタの高頻度サンプリング(High Frequency Sampling of Processor Performance Counters)」と題する米国特許出願第０８／８１２，８９９号に開示されたような効率的なデータ減少技術を可能にする。

【００８５】

命令Ｉがフライト中である間にリタイアした命令の数、又は命令Ｉの周りでイッシューされた命令の数のような他の同時性メトリックも同様に決定することができる。

最終的に、命令Ｉが特定のパイプライン段にある間の特定の実行ユニット２４０の平均的な利用のような更に詳細な情報も抽出又は収集することができる。

単一プロセッササイクル中にパイプライン段により処理される命令の瞬時平均数の決定

図１０に示すように、異なる形式の多路サンプリングを使用して、固定サイズのプロセッササイクル数にわたりパイプラインにより処理される命令の平均数を決定することができる。図１０は、例えば、リタイアされる命令の瞬時平均数を決定するための回路を示す。プロセッササイクル中に、パイプライン２２０のいずれの段１００１についても、同様の

回路を使用して、フェッチ、マップ、イッシュー又は実行される命令の平均数を決定することができる。

【００８６】

装置１０００において、先入れ先出し（ＦＩＦＯ）待ち行列１０１０及びＮ容量の加算器１０２０の各々は、単一のプロセッササイクル中にパイプラインの特定の段１００１により処理される命令の数（カウント１００２）を受け取り、例えば、フェッチ、マップ、イッシュー又は実行される命令の数を受け取る。ＦＩＦＯ待ち行列１０１０におけるエントリの数（Ｐ）１０２２は、平均値が決定されるところのサイクルの数を決定する。Ｐは、ハードウェアで設定されてもよいし、ソフトウェアで設定されてもよい。値Ｐは、平均値が決定されるところのサイクルのウインドウを制御する。

加算器１０１０はスケール型カウントレジスタ１０４０に接続され、従って、このレジスタ１０４０は、Ｎ個のサイクル中にリタイアした命令の全数を累積することができる。ＦＩＦＯ待ち行列１０２０及びレジスタ１０４０は、ライン１０２１及び１０４１を経て初期化することができる。減算器１０３０は、それまでのＮ－１サイクルにリタイアした命令の数をレジスタ１０４０から減算し、例えば、ＦＩＦＯ待ち行列１０１０のヘッドエントリに記憶されたカウントを減算する。レジスタ１０４０の出力は追跡されたサイクルの数（Ｐ）で除算され（１０５０）、段１００１で処理された実際の命令の動特性即ち瞬時平均数１０６０を形成する。瞬時平均値は、プロファイルレジスタ３００に捕獲されるか、或いはソフトウェアで読み取り可能なプロセッサレジスタ又はメモリ位置に記憶される。

サンプルされた命令がリタイアした命令であるときには、コンピュータにより行われた実際の「真」の有効作業を計算することができる。これは、相対的なプロセッサ性能を指示するためにしばしば引用される「生」の命令フェッチレートよりも良好な指示である。例えば、特定のアーキテクチャーは、大きなフェッチレートをもつことができるが、パイプラインにおけるストールが性能を低下することがある。

【００８７】

命令のクラスター化

又、サンプルされた状態情報を使用し、同時性情報を収集しながら当該ケースを識別することもできる。例えば、命令Ｉがキャッシュの１つにおいて「ヒット」するときに平均同時性レベルを計算し、次いで、平均同時性レベルを、命令Ｉがキャッシュミスを招く場合と比較することが有用である。変化する同時性レベルと相関するために検討すべき他の当該特徴は、レジスタ依存性ストール、キャッシュミスストール、分岐予想ミスストール、及び最近の分岐経過を含む。

【００８８】

一般に、Ｎ個ごとのサンプリングは、Ｗ個の命令のウインドウにわたりＦ（Ｉ１、Ｉ２）と表すことのできるファンクションの値をサンプリングすることにより種々の異なるメトリックを統計学的に計算できるようにする著しい融通性を与える。対応する公知のハードウェアメカニズムとは対照的に、ここに与えられる融通性は、Ｎ個ごとのサンプリングを、複雑なプロセッサに関する同時性情報を捕獲するための非常に優れた選択肢にする。これは、新規なメトリック及び分析技術の設計を可能にするためである。

標準的なＳＰＥＣベンチマークソフトウェアを実行するプロセッサでの実験では、統計学的に収集されたサンプルをベースとするメトリックは、低いオーバーヘッドの完全な情報で得られた値に収斂することが示されている。

【００８９】

経路プロファイル

命令のクラスターをプロファイリングする付加的な効果は、経路プロファイルが得られることである。経路プロファイルは、多数のコンパイラー最適化及びトレーススケジューリングに有用である。

更に、最近の分岐実行経過と共にプログラムの実行経路に沿った多数のポイントを制限することにより、経路プロファイルが明確化される。この明確化は、Ｎ個ごとのサンプリン

グとで改善され、即ちＮが増加するにつれて、明確化が改善される。著しく実行されるコードの場合には、同時プログラムが、全ての実行命令に対しパイプライン２００の各段において命令の相対的な実行順序を示すことができる。従って、ここでは、オペレーティングシステムにおける実行パイプライン２００の実際のオペレーションを統計学的に再構成することができる。

【００９０】

ランダムにサンプルされるプロファイル情報の他のアプリケーション

マイクロプロセッサの最新の世代は、考えられる最高の性能を与えるためにコンピュータアーキテクチャーが許す全ての策略を利用する。これらのマイクロプロセッサは、サイクル当たり多数の命令をフェッチし、イッシューしそしてコミットする。更に、これらのプロセッサは、命令を順序ずれして実行する。それらのあるものは、メモリオペレーションも順序ずれして実行する。

不都合なことに、プロセッサにより使用される多数の発見的メカニズムが命令及びメモリオペレーションを順序ずれしてイッシューするので、性能特性がかなり変化し得る。１つの効果として、ここに述べるプロファイリング技術は、システム１００の性能を自動的に改善できるように、システムがプログラムの性能を充分詳細に測定できるようにすることである。

【００９１】

最適化

又、本発明のプロファイリング技術は、システム１００の最適化を実行するのにも使用できる。以下の説明は、プログラマー及びコンパイラーで指令されるソフトウェアプログラムの最適化を手引きするよう意図されたものである。

【００９２】

ハードウェアの最適化

アウトオブオーダーのスーパースカラーマイクロプロセッサは、データ及びリソースの利用状態に基づいて命令をスケジューリングし直すので、コンパイル－時間命令スケジューリングは、構造的に簡単なプロセッサの場合よりも重要性がかなり低い。ここでは、主なボトルネックは、命令フェッチ及びメモリオペレーションによるものである。

より詳細には、分岐又はジャンプ予想ミス、オンチップキャッシュミス、及びＴＬＢ欠陥によりプロセッサパイプライン２００においてサイクルが失われる。これらは、静的に推測することが不可能でないまでも困難な状態である。又、高レベルオフチップオペレーションにおける遅延に対しても、キャッシュミス、リソーストラップ及び順序づけトラップのために、サイクルが失われる。失われたサイクルは、時間を浪費する。

従来の事象カウンタでは、これらの性能低下事象の合計数を測定することはできるが、失われたサイクルをプログラムの特定の命令に起因させることは不可能ではないまでも非常に困難である。ここに述べるプロファイリング技術は、ユーザが主な性能問題を測定して、それら問題を特定の命令に相関させることができるようにする。

【００９３】

フロントエンド最適化

性能の助けとなる１つのフロントエンド最適化は、基本的ブロックにおいて命令をそして手順において基本的ブロックを順序付けし直すことである。基本的ブロックとは、１つの単位として直線的に実行されるか又は全く実行されない命令のセットとして定義される。手順とは、一般に、コール命令を経て到達する基本的ブロックの凝集セットである。手順は、多数の基本的ブロックを含むことができる。基本的ブロックにおいて命令をそして手順において基本的ブロックを順序付けし直すことは、ページ及びキャッシュの一時的な位置を最適化すると共に、分岐の数を減少するように実行流及びデータアクセスを変更できるようにする。分岐は、実行流しか再指令せずそしてデータにおいて有効に作用しないので、サイクルを浪費する。この最適化は、入力として、制御流グラフエッジ周波数を知る必要がある。

【００９４】

トレースの形成

同様に、命令のスケジューリングを追跡するために、コンパイラーは、制御流グラフのエッジ又は経路周波数を必要とする。トレーススケジューラは、各基本的ブロック又はより大きな実行経路を実行するのにどれほどの時間を要するかの推定値を有するときは非常の良好なジョブを行うことができる。アルタ・ビスタサーチエンジンのような大規模な動作システムの場合には、これを従来のツールでリアルタイムに測定することが困難である。

【００９５】

ホット／コールド最適化及び経路情報

トレーススケジューリング及びホット／コールド最適化のような多数のコンパイラー最適化は、プログラムによりどの実行経路が頻繁にとられるかを知ることに依存している。これらは「ホット」経路と称する。最近まで、計装又は模擬のいずれかによりプログラムをプロファイリングすることにより、頻繁に実行される経路が推測されて、基本的なブロック又はエッジカウントが収集され、そしてこれらのカウントを用いて、ホット及びコールド経路が間接的に推測される。

最近、経路情報を直接収集するための技術が使用されている。これらの技術は正確な経路情報を与えるが、非常に高いオーバーヘッドをもつ傾向があり、アクティブな大規模コンピュータシステムを測定するには不適当である。本発明のプロファイリングでは、経路情報を最小のオーバーヘッドでランダムに捕獲することができ、そして実際の実行流の統計学的に正しい概観を依然として表すことができる。

【００９６】

分岐経過レジスタ

ほとんどの近代的なマイクロプロセッサは、グローバルな分岐経過レジスタにおいて最後のＮ個の分岐の方向を追跡する。分岐経過レジスタは、移動ウインドウとして、最近の分岐予想を観察し、そしてそれに応じて将来の命令フェッチに作用を及ぼすことができる。命令のＰＣがサンプリングされると共に、このレジスタの内容を命令フェッチ時間に捕獲することにより、時には、制御流グラフの静的な分析を使用して、プロセッサがとらねばならない最後のＮ個の分岐により厳密な経路を仮定することができる。

【００９７】

しかしながら、従来の経過レジスタは、通常、分岐の方向しか含まず、実際のターゲット行先を含まないので、情報が不正確なものとなる。特に、制御流の合流は、実際にとられた経路を識別する上であいまいさを招く。

又、分岐コードの実行を生じさせる非同期事象、例えば、割り込み又はコンテクストスイッチは、分岐経過ビットを汚染することがある。しかしながら、これらの事象は、比較的稀であり、そしてオペレーティングシステムにおけるそれらの発生は、コードにわたってランダムに分布されねばならない。頻度の高い経路を識別するのが目的であるから、予想不能な非同期事象により発生される「ノイズ性」の分岐経過ビットにより生じるものを含む頻度の低い経路を無視することができる。

【００９８】

図１１に示す命令シーケンスについて考える。ＰＣアドレスＡ－Ｅ（１１０１－１１０５）に命令がある。アドレスＡ及びＣにおける命令１１０１及び１１０３は、分岐型の命令である。ＥのＰＣをもつ命令１１０５があって、グローバルな分岐経過における最後のビットが１である場合には、ＣＤＥで終わるいかなる経路も除外することができる。というのは、このような経路の最後の分岐が失敗に終わり、それ故、グローバルな分岐経過に対応しないからである。しかしながら、ポイントＥにおける異なる制御経路の合体により、実行された真の経路がＡＥ（１１１０）又はＡＢＣＥ（１１１１）であったときを決定することができない。

【００９９】

制御流グラフの合流によるあいまいさ

図１２は、サンプリングされたＰＣ値を入力として使用して、プログラム流の静的な分析を実行することのできるプロセス１２００を示す。選択された命令の経路サンプルがステ

ップ１２１０において上記のように捕獲される。マシンへの影響を最小にするために、サンプリングされた命令はランダムに選択されるのが好ましい。各「経路」サンプル１２２０は、サンプリングされた第１命令Ｉ１のＰＣ１と、命令Ｉ１までの最後のＮ個の条件付き分岐によりとられる方向（ＢＲＡＮＣＨ　ＨＩＳＴ）とを含む。

【０１００】

任意であるが、サンプリングされた情報は、第１命令の直前に実行される第２命令（Ｉ２）のＰＣ２で増強することもできるし、或いは最後のＭ個の分岐のＰＣ値に適用されるあるファンクション、例えば、ある数の下位ビット又はハッシュ関数を用いて決定されたビットを選択するファンクションに基づいて選択された情報で増強することもできる。

ステップ１２４０において、経路サンプルを使用して、プログラムの制御流グラフの逆方向分析を実行する。この分析は、サンプリングされたデータに一致する実行経路を識別することができ（１２５０）、そしてこの情報を収集して、最適化から更に効果が得られる頻繁に実行される経路を識別することができる（１２６０）。

【０１０１】

例えば、図１１を参照すれば、命令Ｅにおいて、１の分岐経過長さが与えられると、経過ビット「１」により、ソフトウェアツールは、経路セグメントＡＥ１１１０及びＡＢＣＥ（１１０１－１１０５）を考えられる経路として識別することができる。分岐経過ビットの値が与えられたときに、静的な分析が、可能性として、単一経路セグメントしか識別できないときに、考えられる最良の成果が得られる。

又、プロセスの最近の実行経過に関する他の情報も、特定の命令に到達するためにとられた実行経路を識別する上で助けとなる。有効な情報の１つの断片は、最近実行された命令の第２のＰＣ値の知識である。おそらくＮ個ごとのサンプリングと共に多数のＰＣ値を使用することにより、１つのＰＣしか含まない経路を除外することができる。

【０１０２】

所与のクラスの最後のＭ個の命令のサンプリング

図１３に示す別の技術においては、ハードウェアは、パイプラインの任意の選択された段、例えばリタイアユニットで処理された最後のＭ個の命令の各々から少数のビット（Ｂ）を捕獲することができる。Ｂビット１３０３は、ＰＣの下位のＢビットでもよいし、或いはＢビットは、ＰＣ１３０４に適用されるハードウェア実施ファンクションＦ１３１０を使用して選択することもでき、即ちＢ←Ｆ（ＰＣ）である。ファンクション１３１０がハッシュ関数である場合には、分岐アドレスの非均一な分布が回避される。

命令のクラスは、例えば、条件分岐、コール、リターン、アクセス（ロード又は記憶）命令、間接的分岐、及び間接的コール１３２１－１３２６として識別することができる。クラスは、比較器又はマルチプレクサのような選択メカニズム１３２０によりライン１３２１を経て選択することができる。又、クラスは、パイプラインの段、例えば、フェッチ、マップ又はリタイア等により識別することもできる。クラスＩＤ１３１９は、ソフトウェアにより制御される。

【０１０３】

選択されたビットは、ＭｘＢビット巾のシフトレジスタ１３００に記憶することができる。このレジスタは、ソフトウェアの内部レジスタとして或いはメモリ位置として図５のＰＳＷ５２０へアクセスすることができる。識別されたクラスの命令１３２１－３１２４が処理されるときには、シフトレジスタ１３００は、その上位のＢビット１３０２を破棄するようにシフトされる。命令のＰＣ１３０４の選択されたＢビット１３０３は、空きビット１３０５へとシフトされる。従って、レジスタ１３００は、これら形式の命令に対し指紋即ち「経路符号」として働く。レジスタ１３００は、例えば、実行された最新のＭ個の分岐を制限する助けをする。というのは、現在経路符号に一致しない経路は、考慮対象から排除できるからである。分岐命令１３２１に対し、図２の分岐実行指示２８７を使用して、サンプリングをトリガーすることができる。

【０１０４】

経路符号により得られる精度の改善は、相当のものとなり、例えば、Ｂ＝４、Ｍ＝６のよ

うに、最後の６個の分岐から４つのビットを節約するだけでも、標準的なＳｐｅｃＩｎｔ９５ベンチマークプログラムに対する実行経路を決定する精度が２倍になる。
経路符号及びグローバルな分岐経過を使用すると、トレースを次のように分析することができる。
トレースにおいて実行される各命令に対し、次のいずれかに達するまで経路セグメントを決定するように逆方向に進行する。
ａ）グローバルな分岐経過ビットが尽きる、又は
ｂ）命令を含むルーチンの開始点に到達する。
制御流グラフの逆方向進行中に手順のコール命令に遭遇したときには、コールされた手順を通して逆方向に進行し、そして最終的に、そのコールされた全ルーチンを通して逆方向に作用するに充分な分岐経過があるときに、コール側手順に復帰する。従って、実行流のより正確な概観が与えられる。
【０１０５】
キャッシュ及びＴＬＢヒットレートの増強
キャッシュ又は変換ルックアサイドバッファ（ＴＬＢ）における高いミスレートは、システムの性能を著しく低下する。公知の解決策は、一般に、キャッシュミスアドレスを収集する特殊なハードウェア又は特殊なソフトウェア機構、例えば、ＴＬＢを周期的にフラッシュするものに依存している。観察されたミスパターンは、頻繁にアクセスされるページ即ち「ホット」ページのおおよその理解を与え、これは、仮想／物理ページマッピングポリシーに影響するように使用することができる。しかしながら、完全な分析を行うのに必要なアドレス情報は、事象が検出されるときまでに得られない。
【０１０６】
図１４は、より正確な仮想／物理ページマッピングを実行するのに使用できるプロセス１４００を示す。ステップ１４１０では、マッピングされるべきコードがシステムにおいて実行される。ステップ１４２０では、メモリをアクセスするオペレーション（ロード及び記憶）がサンプリングのために選択される。オーバーヘッドを最小にするためにサンプリングはランダムであるのが好ましい。
命令が実行される間に、有効な仮想メモリアドレスが、ステップ１４３０において、キャッシュ及びＴＬＢミスと共に識別され、従って、１つの効果として、事象及びアドレスを特定の命令に直接的に起因させることができる。同様に、ステップ１４４０において、高いアクセスレートで隣接ページを識別することができる。ステップ１４５０では、キャッシュ及びＴＬＢにおけるアクセス競合を減少するために、仮想／物理ページマッピングを調整することができる。ステップ１４６０では、隣接ページを大きな「スーパーページ」へと合成し、ページングオーバーヘッドを減少することができる。
キャッシュ又はＴＬＢにおいて捕獲され損なったメモリ参照の仮想アドレスは特定の命令に直接的に起因させて、ページマッピングポリシーを誘導するに必要な情報の形式を厳密に与えることができる。アプリケーションのメモリ参照流に関する情報を使用して、オペレーティングシステムの仮想／物理マッピングポリシーを動的に制御すると、大きな直接マップ式キャッシュにおける競合ミスを首尾良く回避し、スーパーページの形成によるＴＬＢミスレートを低減し、そしてページの複写及び移動による非均一メモリアクセス時間（ＮＵＭＡ）マルチプロセッサにおける遠隔メモリ参照の数を減少することができる。
【０１０７】
改良された命令スケジューリング
コード最適化の間に行われる１つの重要なタスクは、理想的な命令スケジューリングである。理想的な命令スケジューリングは、メモリ待ち時間による遅延を最小にするようにコードを順序付けし直す。基本的なブロックにおける隣接命令の静的な順序付けは、前世代のインオーダー型ＲＩＳＣプロセッサの場合よりも重要性が低いが、巨視的な命令スケジューリングは、アウトオブオーダー型プロセッサにおいて非常に重要である。
命令スケジューリングについての１つの非常に困難なものは、ロード及び記憶のスケジューリングである。これは、静的なスケジューラが、メモリアクセス命令を最適にスケジュ

ールできるようにする厳密な依存性情報を常に有していないからである。加えて、メモリアクセス命令の待ち時間を厳密に予想することが困難である。命令スケジューラは、通常、メモリアクセスに関する正確な情報が不充分であるから、一般に、Ｄキャッシュヒットを仮定してロード及び記憶をスケジュールする。或いは又、バランス型スケジューリングは、ロード当たり等しい量の待ち時間を含むスケジュールを発生するよう試みる。これは、ロード／記憶オペレーションがキャッシュにおいて常にヒットすると常時仮定することに勝る改良である。

【０１０８】

マルチスレッド型プロセッサにおけるスレッドのスケジューリング

マルチスレッド型プロセッサにおいては、上記プロファイリング方法を用いて得たスレッドのリソース利用に関する情報を使用して、全体的なリソース利用度及びスループットを最大にするようにスレッドをスケジューリングすることができる。

２つのスレッドがリソースの相補的な使い方を有し、例えば、一方のスレッドが主として整数演算ユニットを使用するが、他方のユニットは主として浮動小数点演算ユニットを使用する場合には、２つのスレッドが異なる機能的実行ユニットを使用するので、２つのスレッドを同時に動作するようにスケジューリングすることができる。同様に、２つのスレッドが競合するリソース使用を有し、例えば、両スレッドが浮動小数点演算ユニットを頻繁に使用する場合には、それらを異なる時間に動作するようにスケジューリングすることができる。

【０１０９】

図１４ｂは、プロセッサの利用度によりスレッドをスケジューリングするためのプロセスを示す。オペレーティングシステムにおいて実行されるスレッドのリソース利用度がステップ１４７０において測定される。ステップ１４７５では、リソースの利用度が収集され、そしてそのリソース利用度に基づいてスレッドがセットへと分類される。本発明のサンプリングでは、各スレッドが、プロセッサにおける各クラスのリソース、例えば、整数演算ユニット、浮動小数点演算ユニット、メモリユニット、分岐ユニット、イッシューユニット等々をいかに使用するかを決定することができる。

ステップ１４８０では、スレッドのリソース利用度を比較して、非競合実行スケジュールを決定する。所与のクラスのリソースに対する１組のスレッドの合成利用度により、そのクラスのリソースが完全利用状態より著しく多く利用されている場合には、その組のスレッドを一緒にスケジューリングしてはならず（ステップ１４９０）、逆に、合成利用度により、そのクラスのリソースが完全利用状態以下で利用されるか或いは完全利用状態より若干多めに利用される場合には、それらを一緒にスケジューリングするのが有益である（ステップ１４８５）。

【０１１０】

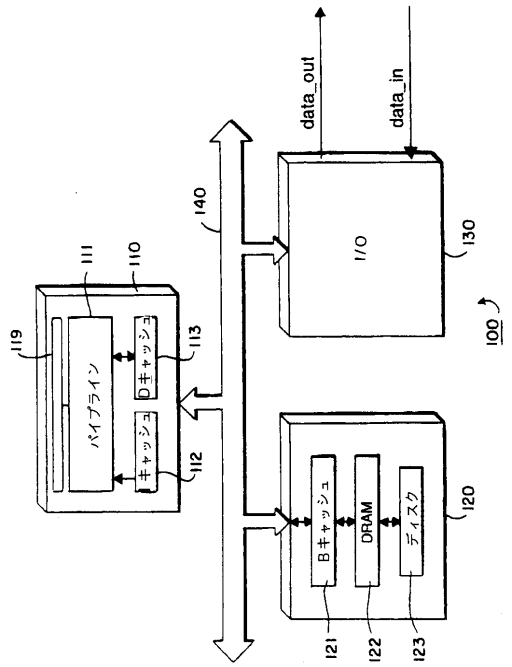
図１５は、命令のスケジューリングを実行するのに使用できるプロセス１５００を示す。マシンコード１５１０は、図１のシステム１００で実行される。コードが実行される間に、メモリオペレーション命令の待ち時間がステップ１５２０において上記のように測定される。多数の命令、例えば、命令対に対する測定値をステップ１５３０においてサンプリングすることができる。サンプリングは、オーバーヘッドを減少するためにランダムに行うことができる。同じＰＣをもつ命令に対してサンプリングされたデータは、ステップ１５４０において収集されて、例えば、待ち時間のヒストグラム（ＨＩＳＴ）１５４１が形成される。ステップ１５６０では、マシンコードが順序付けし直される。この再順序付けは、収集されたヒストグラム情報１５４１に基づく。例えば、長い待ち時間をもつメモリオペレーションは、それらに依存するオペレーションからできるだけ離れるように進められる。ステップ１５６０は、リストスケジューリング或いはトレーススケジューリングのようなスケジューリングアルゴリズムを使用することができる。

ランダムサンプリングによりロード及び記憶待ち時間を収集する場合には、各命令を待ち時間のヒストグラムに基づいてスケジューリングすることができる。本発明の技術は、全キャッシュシュミレーションの経費を被ることなく待ち時間情報を収集することにより最

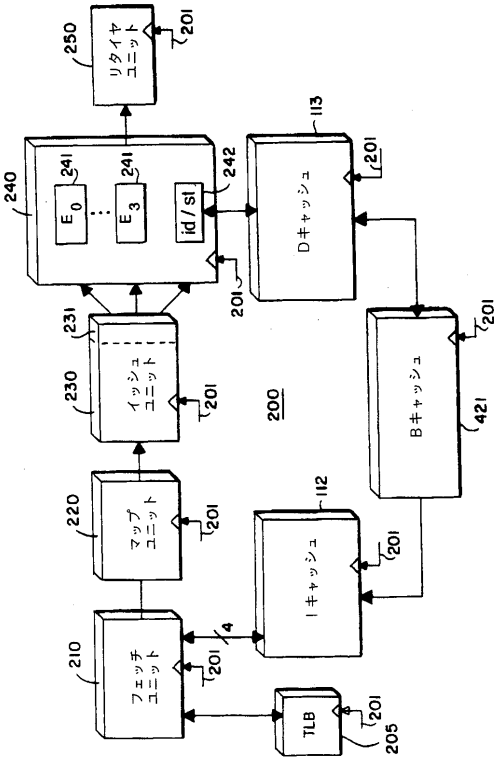
適化を導出するように使用できる。
【０１１１】
プリフェッチ命令の挿入
図１６は、測定された待ち時間に基づいてプリフェッチ命令を挿入するためのプロセスを示す。プリフェッチ命令の挿入は、メモリから返送されるべきデータを待機することにより生じるプロセッサストールを隠す上で助けとなる技術である。データが実際に必要とされる充分前にメモリシステムに要求を発生し、そして時々はデータが必要になると決定される直前にデータを要求することにより、コンパイラー及びオプチマイザーは、メモリからデータをフェッチするための待ち時間のほとんど又は全部をしばしば隠すことができる。
しかしながら、性能を実際に改善するためには、著しい待ち時間を実際に経験するメモリオペレーションに対してのみプリフェッチ命令を挿入することが望ましく、即ち長い待ち時間を実際に被らないメモリオペレーションにプリフェッチ命令を挿入すると、付加的なプリフェッチ命令を実行しなければならないためにプログラムが実際上低速化されてしまう。メモリオペレーション、特に、プリフェッチから利益を得るロードオペレーションを識別するために、プログラム内の種々のメモリオペレーションにより経験する平均待ち時間に関する統計学的データを収集することが所望される。
その一般的な構成が図１６に示されている。ステップ１６１０では、プログラム内のメモリオペレーションに対するメモリオペレーション待ち時間が測定される。ステップ１６２０では、同じプログラムカウンタ（ＰＣ）値をもつ命令に対しサンプリングされたメモリオペレーション情報が収集される。ステップ１６３０では、プリフェッチを挿入すべき大きなメモリ待ち時間をもつメモリオペレーションのサブセットが識別される。
【０１１２】
ステップ１６４０では、実行頻度情報及び測定された待ち時間情報に基づき、これらのメモリオペレーションに対してプリフェッチ命令を挿入するのに有益な位置が識別される。ステップ１６５０では、その適当な位置にプリフェッチ命令が挿入される。
待ち時間は、上記のように測定することができる。１つの方法は、サンプリングハードウェアでメモリオペレーションの待ち時間を直接測定することである。別の方法は、ロード命令が対の第１サンプルでありそしてロードからのデータの使用が対の第２サンプルであるである場合に、対構成でサンプリングを行いそして対を探索することによるものである。２つのサンプルにおいて待ち時間情報を探し、そして特に２つのサンプルのイッシュー時間の差を探すことにより、ロードオペレーションに対するメモリシステム待ち時間を推定することができる。
【０１１３】
以上、特定の実施形態について詳細に説明した。当業者であれば、上記実施形態を変更しても、幾つかの又は全ての効果が達成されることが明らかであろう。それ故、本発明の精神及び範囲内に包含されるこのような修正や変更は全て請求の範囲内に含まれるものとする。
【図面の簡単な説明】
【図１】命令駆動状態サンプリングを伴うコンピュータシステムのブロック図である。
【図２ａ】サンプリングされた命令を処理するためのマイクロプロセッサ実行パイプラインのブロック図である。
【図２ｂ】サンプリングすることのできる状態情報を示すパイプラインのブロック図である。
【図３】プロファイル情報を記憶するためのレジスタファイルのブロック図である。
【図４】増強された命令のブロック図である。
【図５】選択された命令をプロファイリングするための流れ線図である。
【図６】パイプライン待ち時間を測定するための回路を示す回路図である。
【図７】プロセスの流れ線図である。
【図７ａ】命令をサンプリングするプロセスの流れ線図である。

【図７ｂ】プロセッサパイプラインにより処理される命令の特性の統計値を推定するためのプロセスを示す流れ線図である。
【図８ａ】命令の同時実行を示す図である。
【図８ｂ】命令の同時実行を示す図である。
【図８ｃ】命令の同時実行を示す図である。
【図９】費やされる発生スロットを決定するプロセスを示す流れ線図である。
【図１０】プロセッササイクル中に処理される命令の平均数を決定するための装置のブロック図である。
【図１１】命令シーケンスの制御の流れを示すグラフである。
【図１２】制御流を識別するプロセスのデータの流れを示す図である。
【図１３】分岐経過を収集する装置のブロック図である。
【図１４ａ】ページマッピングプロセスの流れ線図である。
【図１４ｂ】スレッドスケジューリングプロセスの流れ線図である。
【図１５】メモリ待ち時間の影響を受ける命令スケジューラの流れ線図である。
【図１６】プリフェッチ命令を挿入するためのプロセッサ１６００の流れ線図である。
【符号の説明】
１００　コンピュータシステム
１１０　プロセッサ
１１１　パイプライン
１１２　データキャッシュ（Ｄキャッシュ）
１１３　命令キャッシュ（Ｉキャッシュ）
１１９　プロセッサ状態をサンプリングするハードウェア
１２０　オフチップメモリ
１２１　汎用キャッシュ
１２２　揮発性メモリ
１２３　永続的メモリ
１３０　入力／出力インターフェイス（Ｉ／Ｏ）
１４０　バスライン
２００　実行パイプライン
２０５　変換ルックアサイドバッファ（ＴＬＢ）
２１０　フェッチユニット
２２０　マップユニット
２３０　イッシューユニット
２４０　実行ユニット
２５０　リタイアユニット

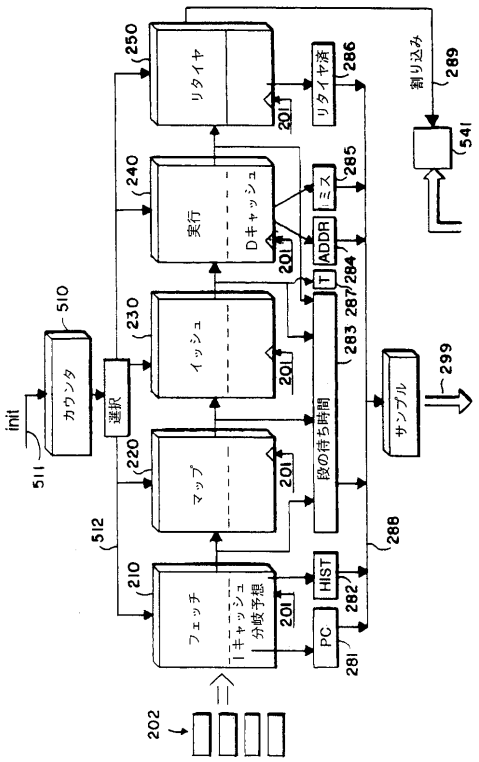
【図１】


data_out
data_in
140
I/O
130
100
111
110
119
113
パイプライン
Dキャッシュ
Iキャッシュ
112
120
Bキャッシュ
DRAM
ディスク
121
122
123

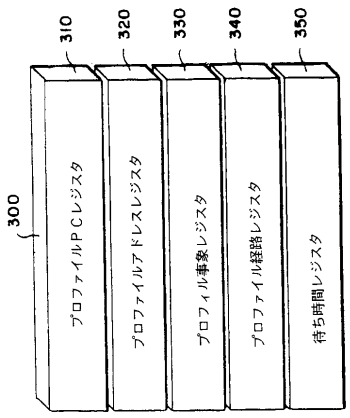
【図２ａ】


250
リタイヤユニット
201
240
241
E0
E3
id/st
242
113
Dキャッシュ
231
イッシュユニット
230
200
220
マップユニット
Bキャッシュ
421
112
Iキャッシュ
210
フェッチユニット
4
TLB
205

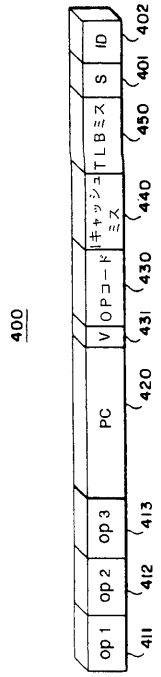
【図２ｂ】


250
リタイヤ
リタイヤ済
286
割り込み
289
541
240
実行
Dキャッシュ
ミス
285
ADDR
T
284
287
283
段の待ち時間
288
230
イッシュ
299
サンプル
510
カウンタ
選択
init
511
220
マップ
512
210
フェッチ
Iキャッシュ
分岐予想
HIST
282
PC
281
202

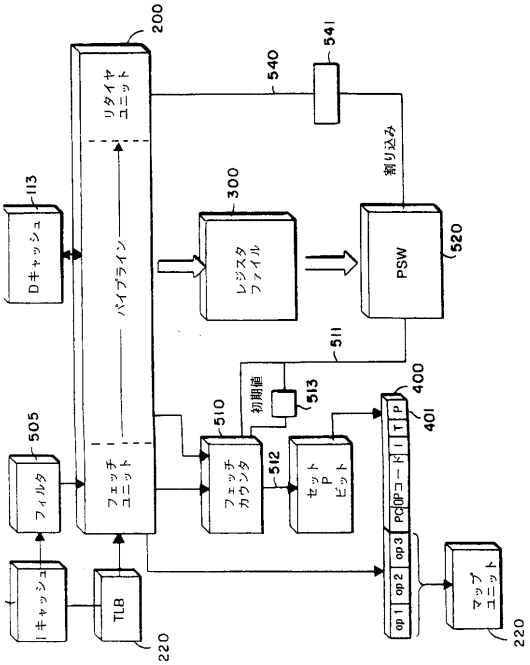
【図３】


310
320
330
340
350
300
プロファイルPCレジスタ
プロファイルアドレスレジスタ
プロフィル事象レジスタ
プロファイル経路レジスタ
待ち時間レジスタ

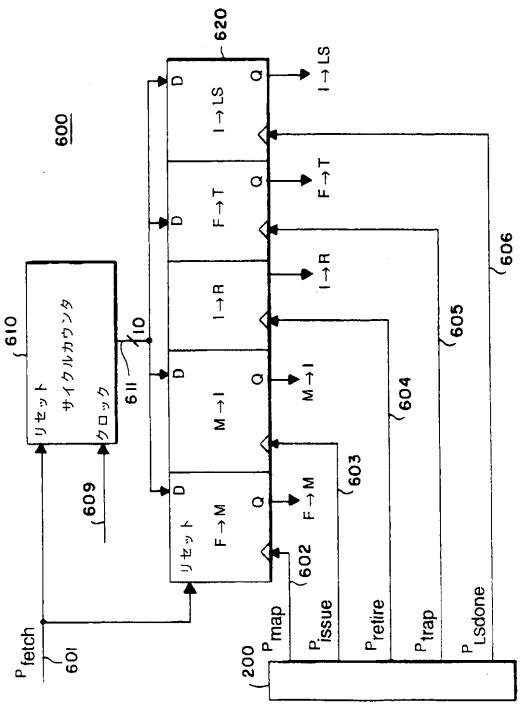
【図４】


400
op 1
op 2
op 3
PC
V
OPコード
Iキャッシュミス
TLBミス
S
ID
411
412
413
420
431
430
440
450
401
402

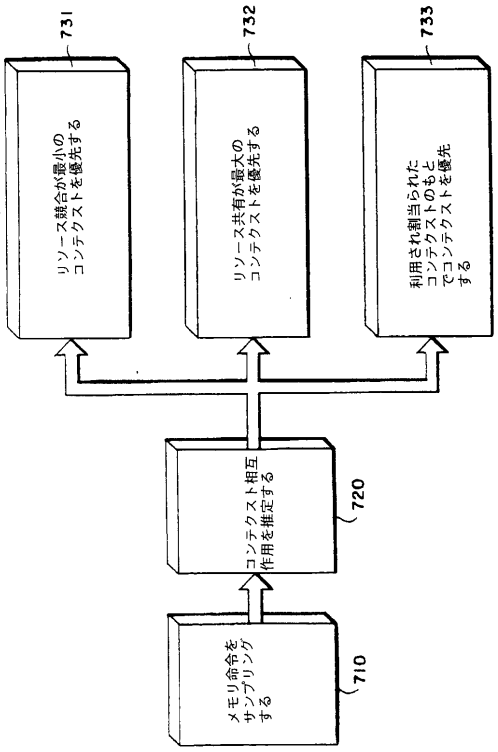
【図５】


200
リタイヤユニット
540
541
割り込み
113
Dキャッシュ
パイプライン
300
レジスタファイル
PSW
520
511
513
初期値
505
フィルタ
フェッチユニット
510
フェッチカウンタ
512
セットPビット
400
401
Iキャッシュ
TLB
220
op 1
op 2
op 3
PC
OPコード
I
T
P
マップユニット
220

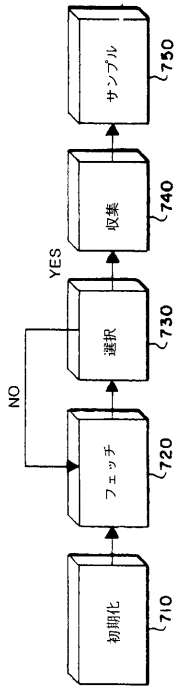
【図６】


600
610
リセット
サイクルカウンタ
クロック
609
611
10
620
D
Q
I→LS
F→T
I→R
M→I
F→M
リセット
602
603
604
605
606
601
Pfetch
200
Pmap
Pissue
Pretire
Ptrap
PLsdone

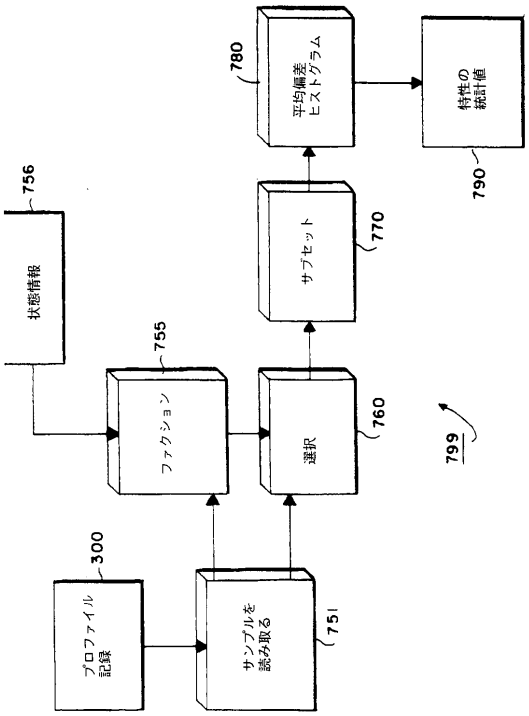
【図７】


731
リソース競合が最小のコンテクストを優先する
732
リソース共有が最大のコンテクストを優先する
733
利用され割当られたコンテクストのもとでコンテクストを優先する
720
コンテクスト相互作用を推定する
710
メモリ命令をサンプリングする

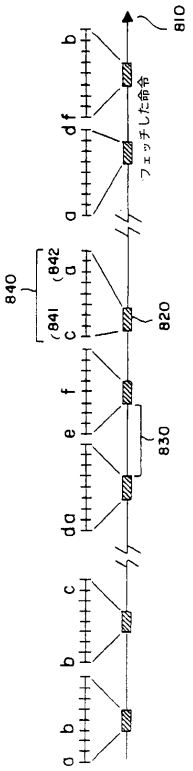
【図７ａ】



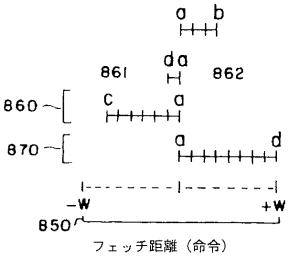
【図７ｂ】



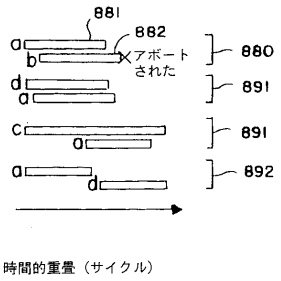
【図８ａ】



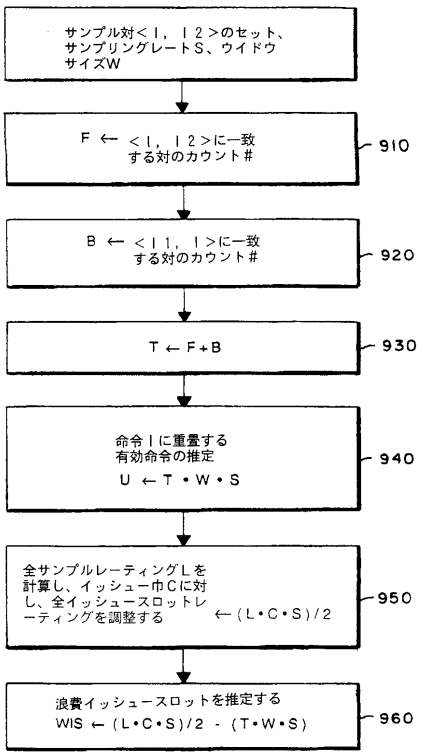
【図８ｂ】



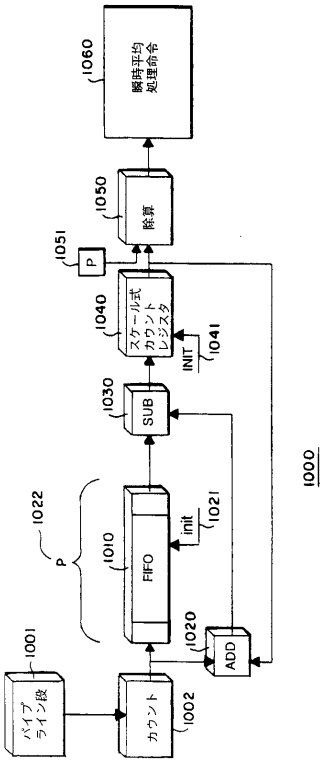
【図８ｃ】



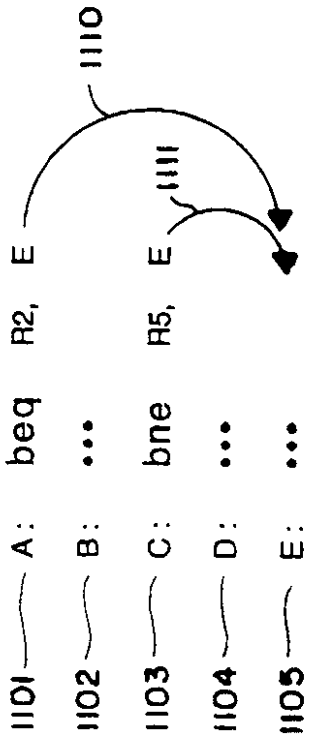
【図９】



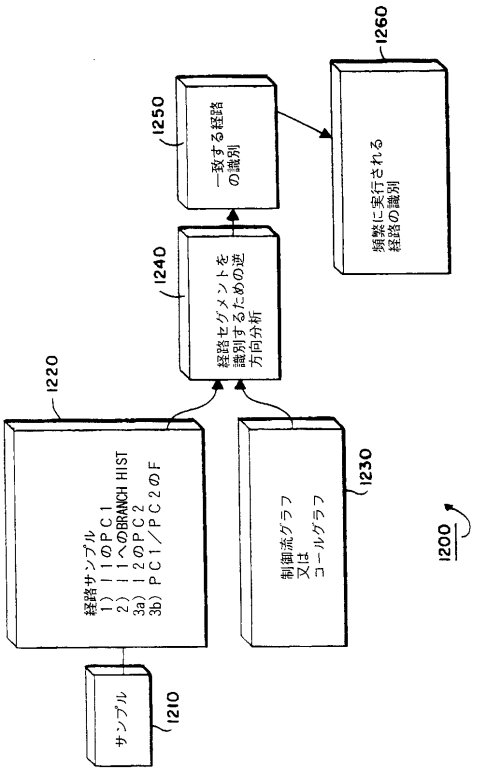
【図１０】



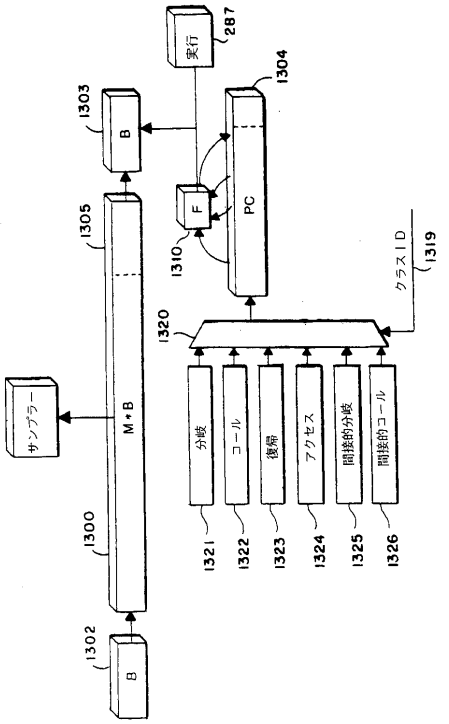
【図１１】



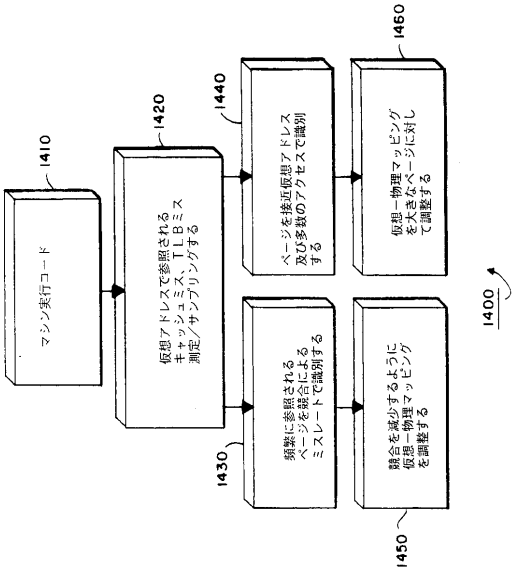
【図１２】



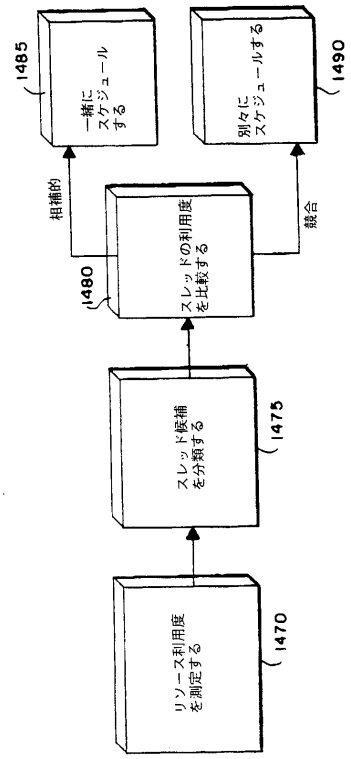
【図１３】



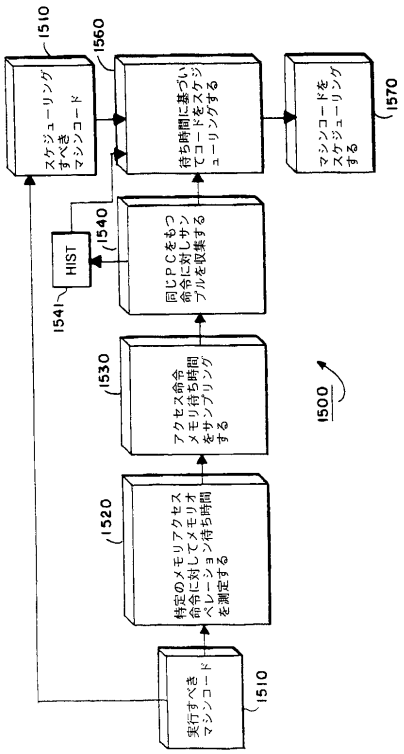
【図１４ａ】



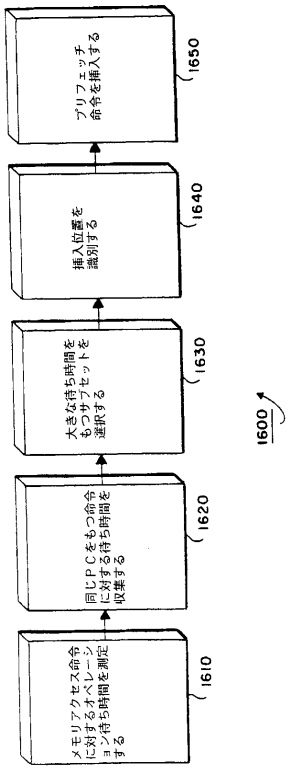
【図１４ｂ】



【図１５】



【図１６】


メモリアクセス命令に対するオペレーション待ち時間を測定する
1610
同じＰＣをもつ命令に対する待ち時間を収集する
1620
大きな待ち時間をもつサブセットを選択する
1630
挿入位置を識別する
1640
プリフェッチ命令を挿入する
1650
1600

フロントページの続き

(74)代理人 100074228
弁理士 今城 俊夫
(74)代理人 100084009
弁理士 小川 信夫
(74)代理人 100082821
弁理士 村社 厚夫
(72)発明者 ジェフリー エイ ディーン
アメリカ合衆国 カリフォルニア州 ９４０２５ メンロ パーク フィフティーンス アベニュー ８８４
(72)発明者 ジェームズ イー ヒックス
アメリカ合衆国 マサチューセッツ州 ０２１５９ ニュートン ボウ ロード ６３
(72)発明者 スティーヴン シー ルート
アメリカ合衆国 マサチューセッツ州 ０１５８１ ウェストボロー ウェスト メイン ストリート ２０１
(72)発明者 カール エイ ウォールドスパージャー
アメリカ合衆国 カリフォルニア州 ９４０２７ アサートン パーク ドライヴ ２７
(72)発明者 ウィリアム イー ウィール
アメリカ合衆国 カリフォルニア州 ９４１１４ サン フランシスコ クリッパー ストリート ２８０

審査官 坂庭 剛史

(56)参考文献 特開平０８－１７１５０５（ＪＰ，Ａ）
特開平０８－０３０４９４（ＪＰ，Ａ）
特開平０７－０６４７９９（ＪＰ，Ａ）
特開平０３－２６３１３６（ＪＰ，Ａ）
特開平０３－１７５５３３（ＪＰ，Ａ）
特開平０３－０９９３４２（ＪＰ，Ａ）
特開平０２－１５８８４７（ＪＰ，Ａ）
特開平０２－０８３７５１（ＪＰ，Ａ）
特開平０１－２２００４２（ＪＰ，Ａ）
特開昭６３－０４６５５２（ＪＰ，Ａ）
特開昭６３－０２５７４２（ＪＰ，Ａ）
特開昭６２－２４３０３５（ＪＰ，Ａ）
特開昭６２－２１４４５０（ＪＰ，Ａ）
特開昭５９－０６９８５３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 11/34
G06F 9/38