

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第4467094号
(P4467094)

(45) 発行日 平成22年5月26日 (2010.5.26)

(24) 登録日 平成22年3月5日 (2010.3.5)

(51) Int. Cl.

F I

G 0 6 F 11/34 (2006.01)

G 0 6 F 11/28 (2006.01)

G 0 6 F 9/38 (2006.01)

G 0 6 F 11/34 L

G 0 6 F 11/28 3 1 0 B

G 0 6 F 9/38 3 1 0 A

G 0 6 F 9/38 3 1 0 J

G 0 6 F 9/38 3 8 0 C

請求項の数 37 外国語出願 (全 38 頁)

(21) 出願番号 特願平10-375539
 (22) 出願日 平成10年11月26日 (1998.11.26)
 (65) 公開番号 特開平11-272520
 (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 08/980168
 (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)

前置審査

(73) 特許権者 398038580
 ヒューレット・パカード・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 100084009
 弁理士 小川 信夫

最終頁に続く

(54) 【発明の名称】 プロセッサパイプラインにおいて多数の潜在的に同時の命令をサンプリングする装置

(57) 【特許請求の範囲】

【請求項 1】

システムの複数の処理段を有するプロセッサパイプライン (200) において命令をサンプリングする装置であって、

パイプラインの第1段へ、サンプルビット (401) 及びサンプルレジスタファイル識別フィールド (402) を含む複数の指示フィールドをそれぞれが有する命令をフェッチする手段 (210) と；

現在実行されているフェッチされた命令のいずれかのサブセットをサンプリングのために選択された命令としてランダムに識別するのに、その選択された命令中のサンプルビット (401) をアサートすることによりランダムに識別する手段と；

選択された命令のいずれかがパイプラインのいずれかの段にある間に前記のシステムの状態情報を与える手段 (288) と；

前記状態情報を記憶するための複数のプロファイルレジスタ (300) と；

全ての選択された命令がパイプラインを出てしまったことをソフトウェアアプリケーションに通知して、ソフトウェアアプリケーションがいずれの状態情報も読み取れるようにする手段と

を備え、

前記指示フィールド (401, 402) の中の少なくとも一つは、前記の選択された命令が監視のために選択されたことを指示し、前記指示フィールド (402) は前記の選択された命令と関連する命令番号を指示することを特徴とする装置。

【請求項 2】

前記の選択された命令は、パイプライン（200）により完全に処理される有効命令と、パイプラインを出る前にアボートされる有効命令と、パイプラインを出る前に部分的に処理される無効命令とを含む請求項 1 に記載の装置。

【請求項 3】

前記の監視のための手段をアクチベートするようサンプルビット（401）をアサートする手段を備える請求項 1 に記載の装置。

【請求項 4】

それぞれの選択された命令と関連の状態情報を一つもしくはそれ以上の指示フィールドの中の少なくとも一つによりインデックスされるレジスタ（300）に蓄積する請求項 1 から 3 のいずれか 1 項に記載の装置。

10

【請求項 5】

前記の状態情報は一对の選択された命令の間で生じる事象の事象カウントを含む請求項 1 から 3 のいずれか 1 項に記載の装置。

【請求項 6】

前記の事象カウント是一对の選択された命令のフェッチ間に生じるクロックサイクルの数を含む請求項 5 に記載の装置。

【請求項 7】

前記の事象カウント是一对の選択された命令間にフェッチされた命令の数を含む請求項 5 に記載の装置。

20

【請求項 8】

前記の事象カウントは前記一对の選択された命令間にパイプラインの特定の段に入る命令の数を含む請求項 5 に記載の装置。

【請求項 9】

前記の事象カウント是一对の選択された命令間にフェッチされた分岐命令の数を含む請求項 5 に記載の装置。

【請求項 10】

前記の事象カウントはプロセッサ、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測定を含む請求項 5 に記載の装置。

【請求項 11】

30

特定の事象をカウントする複数のサンプリングカウンタ（510）と、
各サンプリングカウンタを特定の値に初期化する手段と、
特定の事象にตอบสนองしてオーバーフローする前記のサンプリングカウンタの少なくとも 1 つのオーバーフローにตอบสนองしてランダムに識別する前記手段をアクチベートする手段とを更に含んでいる請求項 1 から 10 のいずれか 1 項に記載の装置。

【請求項 12】

ランダムに識別する前記手段は前記の複数のサンプリングカウンタのどれがオーバーフローするかに基づいて別々の指示フィールドで前記の選択された命令を識別する手段を含む請求項 11 に記載の装置。

【請求項 13】

40

ランダムに識別する前記手段はアクチベートされる平均頻度数を示すため各特定値をあるインターバルもしくは間隔の数字からランダムに選択する請求項 11 に記載の装置。

【請求項 14】

識別する前記手段はアクチベートされる平均頻度数を動的に変えるように前記の数字の間隔のサイズを変更する請求項 13 に記載の装置。

【請求項 15】

前記のランダムに選択される数字はソフトウェアで計算され、それからハードウェアレジスタに記憶される請求項 13 に記載の装置。

【請求項 16】

前記のランダムに選択される数字はハードウェア回路により発生される請求項 13 に記

50

載の装置。

【請求項 1 7】

前記の状態情報は前記の選択された命令を識別する情報を含む請求項 1 から 3 のいずれか 1 項に記載の装置。

【請求項 1 8】

前記の状態情報は前記の選択された命令のアドレスを含む請求項 1 7 に記載の装置。

【請求項 1 9】

前記の状態情報は前記の選択された命令を実行するプロセスの識別を含む請求項 1 7 に記載の装置。

【請求項 2 0】

前記の状態情報はアドレススペース番号を含む請求項 1 7 に記載の装置。

【請求項 2 1】

前記の状態情報はハードウェアコンテキスト識別子を含む請求項 1 7 に記載の装置。

【請求項 2 2】

前記の状態情報はスレッド識別子を含む請求項 1 7 に記載の装置。

【請求項 2 3】

前記のソフトウェアアプリケーションに通知する手段が、全ての選択された命令がパイプライン (2 0 0) を出ってしまったことを指示するため割り込みを発生する請求項 1 から 2 2 のいずれか 1 項に記載の装置。

【請求項 2 4】

全ての選択された命令がパイプラインを出してしまったことを指示するためにソフトウェアアプリケーションがポーリングできるフラグを前記のソフトウェアアプリケーションに通知する手段が含んでいる請求項 1 から 2 2 のいずれか 1 項に記載の装置。

【請求項 2 5】

前記の特定の事象は有効なフェッチされた命令である請求項 1 1 に記載の装置。

【請求項 2 6】

フェッチのレートがクロック (6 0 9) により決定され、そして前記の特定の事象は単一の潜在的なフェッチされた命令に対応するクロックサイクルの小区分である請求項 1 1 に記載の装置。

【請求項 2 7】

前記の特定の事象はパイプラインのいずれかの特定の段への命令の導入である請求項 1 1 に記載の装置。

【請求項 2 8】

前記の状態情報は前記の選択された命令の中のリタイア / アボート状態を含む請求項 1 から 3 のいずれか 1 項に記載の装置。

【請求項 2 9】

前記の状態情報は前記の選択された命令を処理する間に検出されるパイプラインの処理状態を含む請求項 1 から 3 のいずれか 1 項に記載の装置。

【請求項 3 0】

前記の状態情報は前記の選択された命令が経験する待ち時間を含む請求項 1 から 3 のいずれか 1 項に記載の装置。

【請求項 3 1】

前記の事象カウントはパイプラインの特定の段に入る一対の選択された命令間に経過するクロックサイクルの数を含む請求項 5 に記載の装置。

【請求項 3 2】

請求項 1 から 3 1 のいずれか 1 項に記載の装置を含むコンピュータシステム。

【請求項 3 3】

システムの複数の処理段を有するプロセッサパイプライン (2 0 0) において命令をサンプリングするコンピュータにより実行される方法であって、

パイプライン (2 0 0) の第 1 段へ、それぞれがサンプルビット (4 0 1) 含む複数の

10

20

30

40

50

指示フィールド及びサンプルレジスタファイル識別フィールド(402)を有する、現在実行されている命令をフェッチする段階と；

そのフェッチされた命令のいずれかのサブセットを選択された命令としてランダムに識別するのに、その選択された命令中のサンプルビット(401)をアサートすることにより識別する段階と；

前記の選択された命令のいずれかがパイプラインのいずれかの段にある間に前記のシステムの状態情報を与える段階と；

前記状態情報を複数のプロファイルレジスタ中に記憶する段階と；

全ての選択された命令がパイプラインを出てしまったことをソフトウェアアプリケーションに通知して、ソフトウェアアプリケーションがいずれの状態情報も読み取れるようにする段階と

を備え、前記指示フィールドの中の少なくとも一つは前記の選択された命令が監視のために選択されたことを指示し、前記指示フィールドは前記の選択された命令と関連する命令番号を指示することを特徴とする方法。

【請求項34】

前記指示フィールドの中の少なくとも一つによりインデックスされるレジスタにそれぞれの選択された命令と関連の状態情報を蓄積する請求項33に記載の方法。

【請求項35】

前記の識別する段階が少なくとも一つのサンプリングカウンターのオーバーフローにตอบสนองして実施され、サンプリングカウンタを特定の値に初期化し、そして特定の事象をカウントするようにする請求項33に記載の方法。

【請求項36】

前記の識別する段階が実行される平均頻度数を示すため各特定値をあるインターバルもしくは間隔の数字からランダムに選択する請求項33に記載の方法。

【請求項37】

前記の状態情報が前記の選択された命令を認識する情報を含んでいる請求項33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、多数の同時実行命令をサンプリングすることに係る。

【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。

10

20

30

40

50

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。

10

【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。

20

【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。

30

命令をアウトオブオーダー(順序ずれて)でイ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

40

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」とであると言える。

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ことを識別する。

【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

50

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十分である。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、Alpha 21164 プロセッサのようなインオーダー（順序正しい

10

20

30

40

50

）マシンにも存在する。

例えば、Alpha 21164（インオーダー）プロセッサ対ペンチウム・プロ（アウトオーダー）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。多数の種類的事象に対し、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。

【 0 0 1 7 】

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。

【 0 0 1 8 】

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。

【 0 0 1 9 】

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ツツ氏等の「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。

【 0 0 2 0 】

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・インターナショナルコンファレンス・オン・アーキテクチャサポート・フォア・プログラミंग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パーシャド氏等の「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。

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

10

20

30

40

50

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。

【 0 0 2 1 】

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の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。

10

【 0 0 2 2 】

【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。

20

【 0 0 2 3 】

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。

30

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。

【 0 0 2 4 】

順序ずれて実行する(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。

40

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。

【 0 0 2 5 】

50

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ィンドウが動的に定義される。命令のウィ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場所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。

10

【0026】

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。

20

【0027】

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

30

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。

【0028】

より詳細には、システムのプロセッサパイプラインにおいて多数の同時実行命令をサンプリングするための装置が提供される。パイプラインは、複数の処理段を有する。この装置は、パイプラインの第1段へ命令がフェッチされたときに選択される多数の命令を識別する。多数の選択された命令のサブセットは、パイプラインにおいて同時に実行される。多数の選択された命令のいずれかがパイプラインのいずれかの段にある間にシステムの状態情報がサンプリングされる。全ての選択された命令がパイプラインを出るときにソフトウェアに通知して、ソフトウェアがいずれの状態情報も読み取れるようにする。

40

【0029】

【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Alpha 21264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

50

トを含む多数の実行パイプライン 1 1 1、命令キャッシュ (I キャッシュ) 1 1 2 及びオンチップデータキャッシュ (D キャッシュ) 1 1 3 として実施することができる。又、プロセッサチップ 1 1 0 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 1 1 9 も備えている。

オフチップメモリ 1 2 0 は、汎用キャッシュ (B キャッシュ又は S R A M) 1 2 1 と、揮発性メモリ (D R A M) 1 2 2 と、永続的メモリ (ディスク) 1 2 3 とを含むハイラキー構成をとることができる。I / O 1 3 0 は、システム 1 0 0 に対してデータを入力及び出力するのに使用できる。

【 0 0 3 0 】

オペレーション

システム 1 0 0 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 1 2 0 に記憶される。命令及びデータは、既知のコンパイラ、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 1 1 2 - 1 1 3 を経て 1 つのプロセッサ 1 1 0 の実行パイプライン 1 1 1 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。

【 0 0 3 1 】

プロセッサパイプライン

図 2 a は、図 1 の 1 つのプロセッサ 1 1 0 の実行パイプライン 2 0 0 を示すもので、これは、例えば、フェッチ、マップ、イッシュ、実行及びリタイアユニット、各々、2 1 0、2 2 0、2 3 0、2 4 0 及び 2 5 0 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 2 0 0 が情報 (データ及び命令) を処理するレートは、ライン 2 0 1 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 2 0 0 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。

【 0 0 3 2 】

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 2 0 0 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。

【 0 0 3 3 】

フェッチユニット

B キャッシュ 1 2 1 は、データ項目を各々 I キャッシュ 1 1 2 及び D キャッシュ 1 1 3 に転送する。フェッチユニット 2 1 0 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 (T L B) 2 0 5 を使用して、実行されるべき次の命令を I キャッシュ 1 1 2 からフェッチする。I キャッシュ 1 1 2 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 2 0 0 は、

10

20

30

40

50

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。

10

【0034】

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。

【0035】

イシューユニット

次の段において、フェッチされた命令は、イシューユニット230によって順序付けされる。イシューユニット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。

20

【0036】

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。

30

【0037】

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、又はト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。

40

【0038】

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

50

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。

10

【0039】

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。

20

【0040】

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でもできる。

【0041】

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。

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ドは、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。

40

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。

50

【 0 0 4 2 】

レジスタマップユニット 2 2 0 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 2 3 0 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 10 が適当な実行ユニット 2 4 0 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。

【 0 0 4 3 】

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 2 1 0 が I キャッシュ 1 1 2 のミスによりストールするか又はフェッチユニット 2 1 0 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 2 2 0 は、空きの物理的レジスタの欠乏、又はイッシュユニット 2 3 0 における空きスロットの欠乏により 20 ストールすることがある。

イッシュユニット 2 3 0 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 2 4 0 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 1 0 0 の性能を低下させる。これにより、ライン 2 8 8 に得られる情報をサンプリングすることが重要となる。

【 0 0 4 4 】

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 3 0 0 が設けられる。メモリ 3 0 0 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 3 0 0 で直接識別される。レジスタファイル 3 0 0 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 3 0 0 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。 30

ファイル 3 0 0 は、図 2 b のライン 2 8 8 によりパイプライン 2 0 0 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 2 0 0 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 3 0 0 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。 40

【 0 0 4 5 】

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 (6 4 ビット)、サイクルカウント、即ち待ち時間 (8 又は 1 0 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 3 0 0 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 3 1 0 は、選択された命令の PC を記憶する。以下に述べるように、プロファイルされている命令は、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 3 1 0 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 50

タ 3 1 0 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 3 1 0 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キ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 3 1 0 がハードウェアコンテキ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。

【 0 0 4 6 】

プロファイル有効アドレスレジスタ 3 2 0 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 6 4 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 2 0 0 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。

【 0 0 4 7 】

プロファイル事象カウンタ 3 3 0 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 3 3 0 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。

【 0 0 4 8 】

プロファイル経路レジスタ 3 4 0 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 3 5 0 は、選択された命令が、例えば、パイプライン 2 0 0 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 3 5 0 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。

【 0 0 4 9 】

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 2 0 0 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 3 5 0 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 3 5 0 の数は、特定の実施形態におけるパイプライン 2 0 0 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 3 5 0 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、イシュー対完了である。この最

10

20

30

40

50

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。

【0050】

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 응답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。

【0051】

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。

【0052】

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。

【0053】

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

10

20

30

40

50

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。

【0054】

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。

【0055】

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、さもなければ、記録されないことを示す。

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。

【0056】

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。PSW 520 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。

【0057】

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が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

10

20

30

40

50

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。

【0058】

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。

【0059】

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。

【0060】

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。

【0061】

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ตอบสนอง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

10

20

30

40

50

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。

【0062】

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。

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。

【0063】

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。

【0064】

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-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。

【0065】

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、或いはオペコード、オペランド、アドレス、事象及び待ち時間の他のブル組合せのみをサンプルリングする。

【 0 0 6 6 】

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、Alpha 21264 プロセッサのようなアウトオブオーダーイッシュ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。

【 0 0 6 7 】

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。

【 0 0 6 8 】

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においてサンプルリングされる。サンプル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ルリングし、より詳細な情報を抽出することもできる。

【 0 0 6 9 】

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ル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

10

20

30

40

50

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キスト識別子、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。

【0070】

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キ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。

【0071】

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。

【0072】

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十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

10

20

30

40

50

がまだ分からないので、実際上不十分である。

【 0 0 7 3 】

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。

【 0 0 7 4 】

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、十分な距離に離れた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を時間的に関連させることができねばならない。

【 0 0 7 5 】

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。

【 0 0 7 6 】

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。

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执行は、潜在的同時性のウィンドウを増加することができる。

Alpha 21264 プロセッサにおいて同時性を検討するために、命令 I (810) の周りのウィンドウ 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、ウィンドウの適度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 のウィ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 < I 1、I 2 > (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 I 1 と第 2 の選択された命令 I 2 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。

10

【0077】

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 I 1 から時間的に前方にそして第 2 命令 I 2 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 < I 1、I 2 > 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関連させるためには、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 < I 1、I 2 > 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。

20

【0078】

浪費イ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 < I 1、I 2 > 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 1、I 2 > 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、< I 1、I 2 > の形態のサンプルされた対を考える。

30

対ごとの各サンプル < I 1、I 2 > 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 I 1 及び I 2 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 を関連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 < I 1、I 2 > がリタイアするかどうかとも指示する。

【0079】

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シュースロットの平均数である。

40

浪費したイ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 I 2 がリタイアするような形式 < I 1、I 2 > のサンプルの数を F 1 とし、サンプルと共に記録される待ち時間は、I 及び I 2 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 F 1 で表される。同様に、ステップ 920 において、I 及び I 2 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 < I 1、I 2 > のサンプルの数を B 1 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。

【0080】

50

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ィンドウのサイズ 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、即ち形成されるイシュースロットの数は、 $W \times (F1 + B1)$ となる。

イ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、Alpha 21264 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。

$$WIS = L1 - (W \times (F1 + B1))$$

値 WIS は、命令 I の実行当たりの浪費イ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 日に出版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。

【0081】

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。

【0082】

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。

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ィンドウを制御する。

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。

【0083】

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。

【0084】

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。

10

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。

20

【0085】

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

30

【0086】

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。

40

【0087】

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使用できる。以下の説明は、プログラマー及びコンパイラ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。

【0088】

ハードウェアの最適化

50

アウトオブオーダーのスーパー scaler 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。

【0089】

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。

【0090】

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。

【0091】

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當である。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。

【0092】

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、移動ウィンドウとして、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。

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キストスイッチは、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。

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。

【0093】

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。

【0094】

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る(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)。

【0095】

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。

【0096】

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。

10

【0097】

選択されたビットは、M×B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1321-1324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。

20

【0098】

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。

30

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。

【0099】

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十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。

【0100】

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。

40

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

50

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。

【0101】

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ccess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。

【0102】

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、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ccess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ccess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ccessに関する正確な情報が不十分であるから、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。

【0103】

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。

【0104】

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

10

20

30

40

50

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、その組のスレッドと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、それらと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)。

【0105】

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、全キャッシュシミュ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。

【0106】

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ブチマイザーは、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。

【0107】

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。

10

20

30

40

50

ステップ 1 6 5 0 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。

【0108】

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

10

20

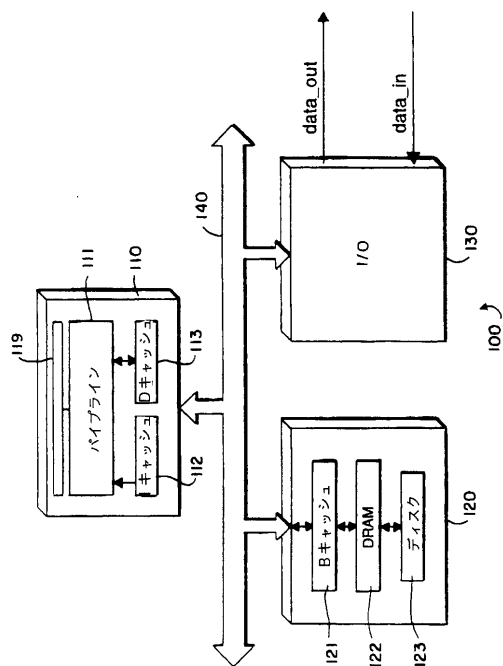
30

40

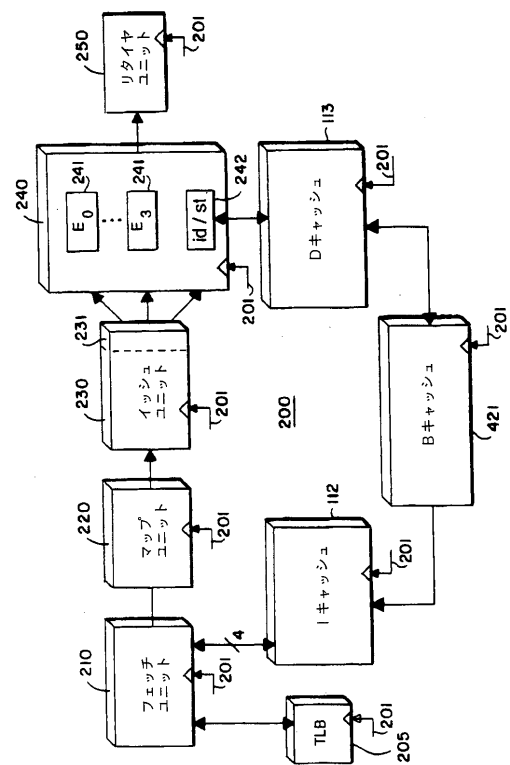
50

- 1 2 3 永続的メモリ
- 1 3 0 入力／出力インターフェイス（ I ／ O ）
- 1 4 0 バスライン
- 2 0 0 実行パイプライン
- 2 0 5 変換ルックアサイドバッファ（ T L B ）
- 2 1 0 フェッチユニット
- 2 2 0 マップユニット
- 2 3 0 イッシューユニット
- 2 4 0 実行ユニット
- 2 5 0 リタイアユニット

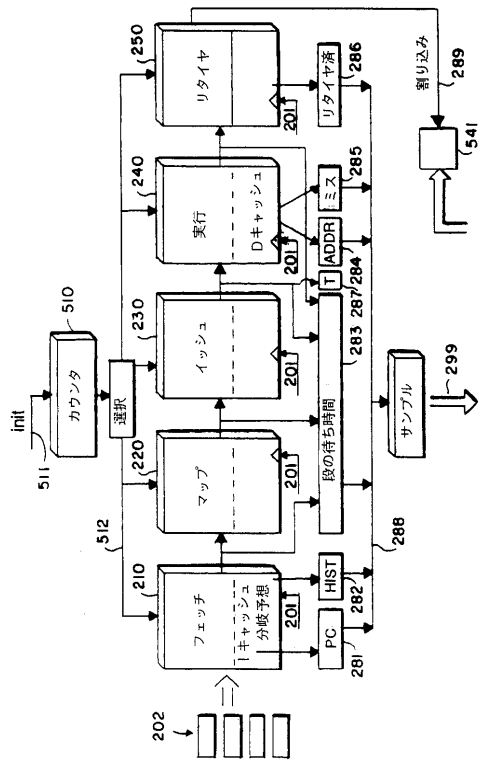
【図 1】



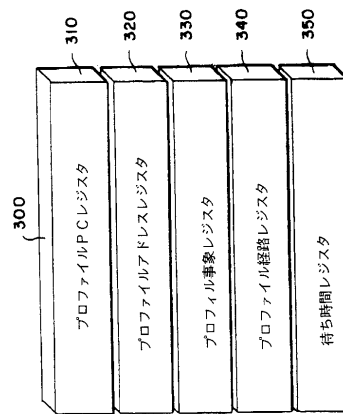
【図 2 a】



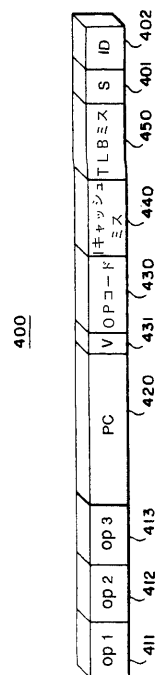
【 図 2 b 】



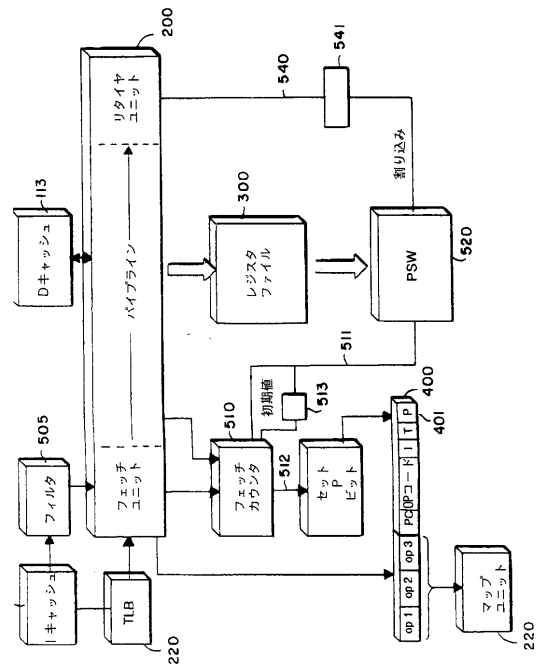
【 図 3 】



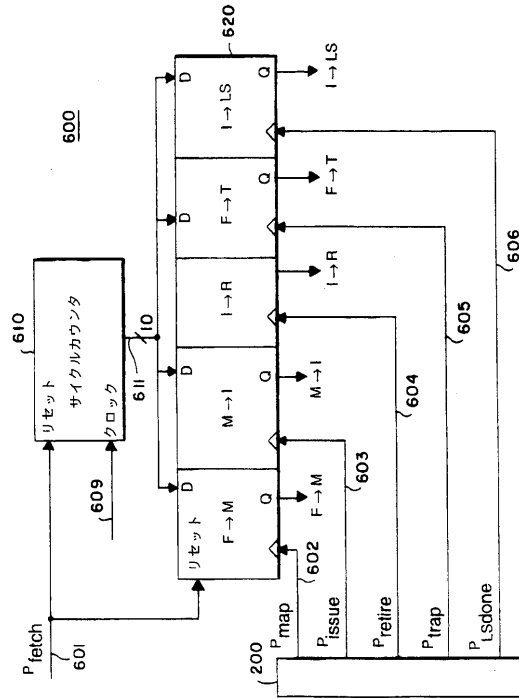
【 図 4 】



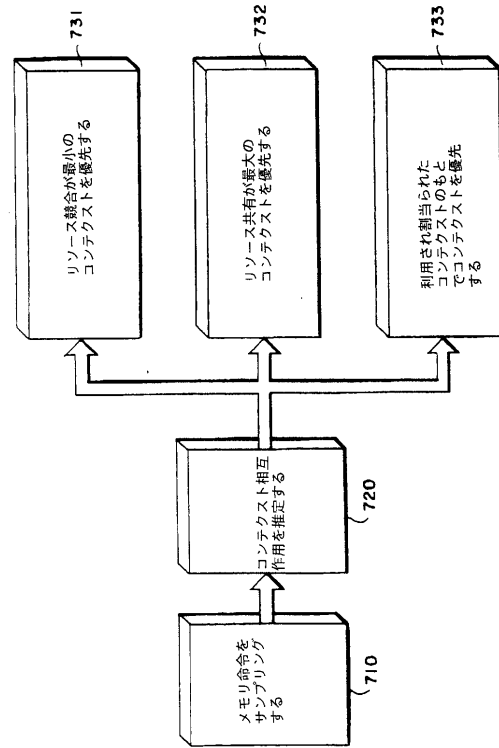
【 図 5 】



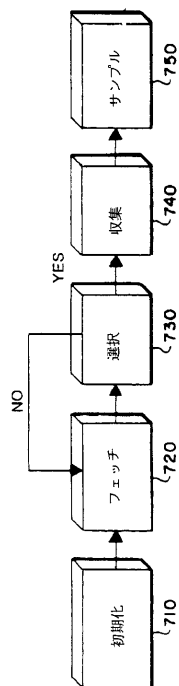
【図 6】



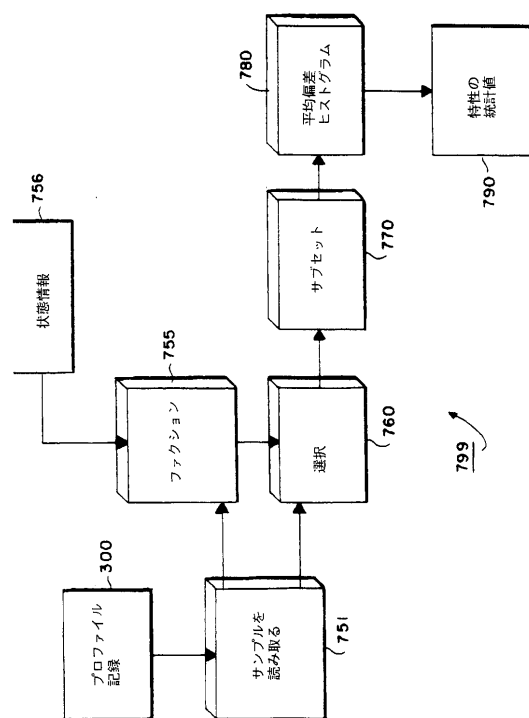
【図 7】



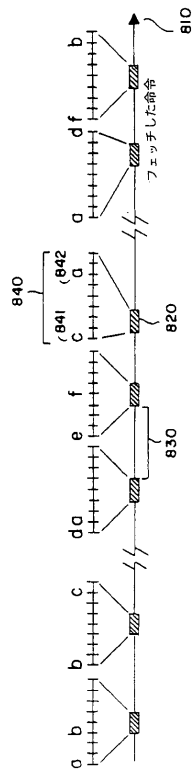
【図 7 a】



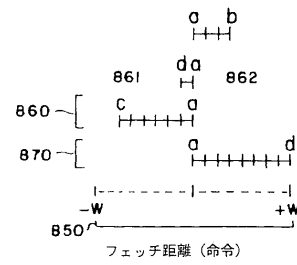
【図 7 b】



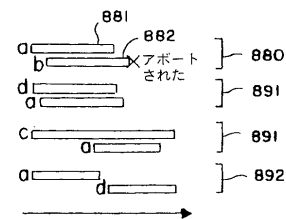
【図 8 a】



【図 8 b】

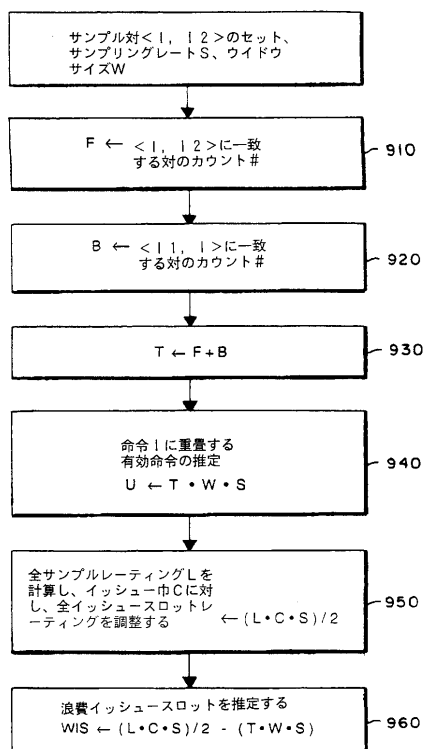


【図 8 c】

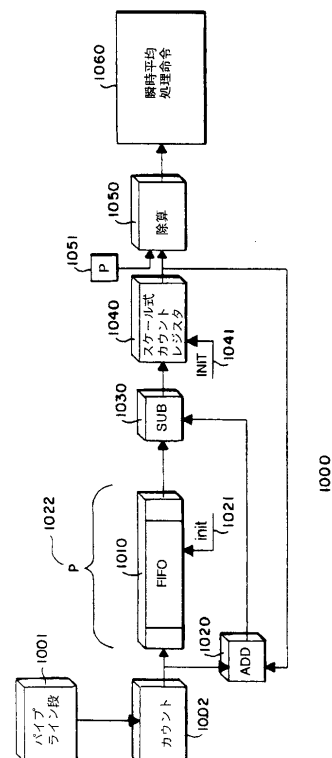


時間的重畳 (サイクル)

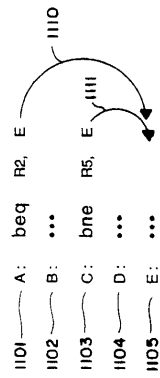
【図 9】



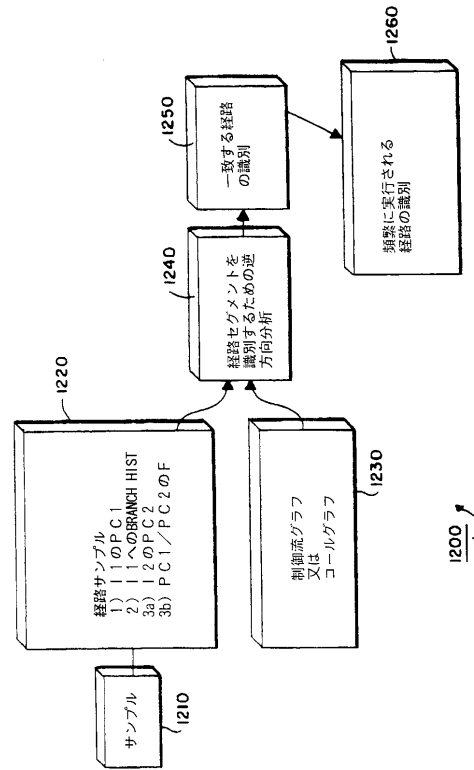
【図 10】



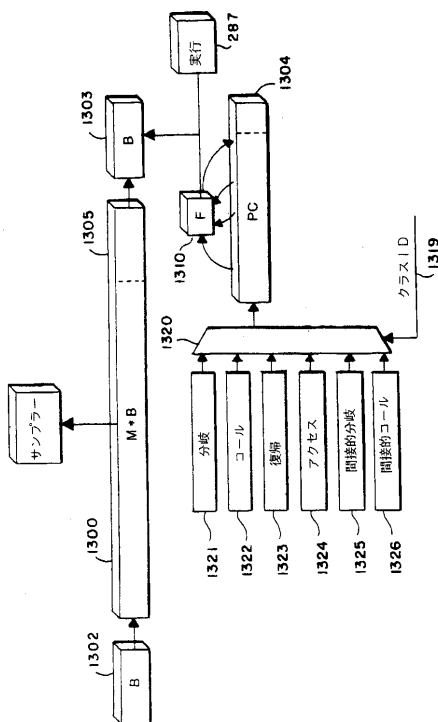
【図 1 1】



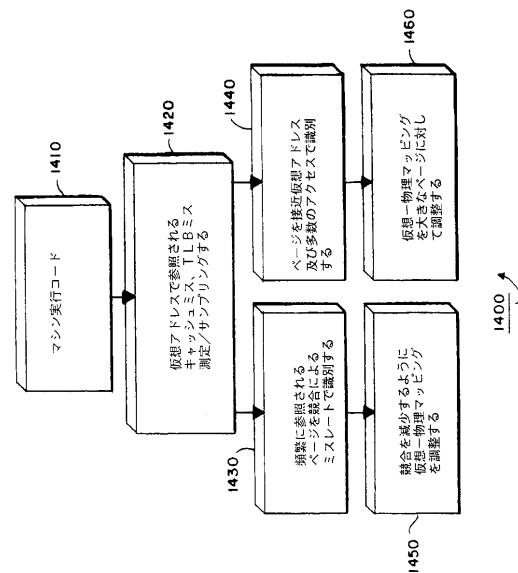
【図 1 2】



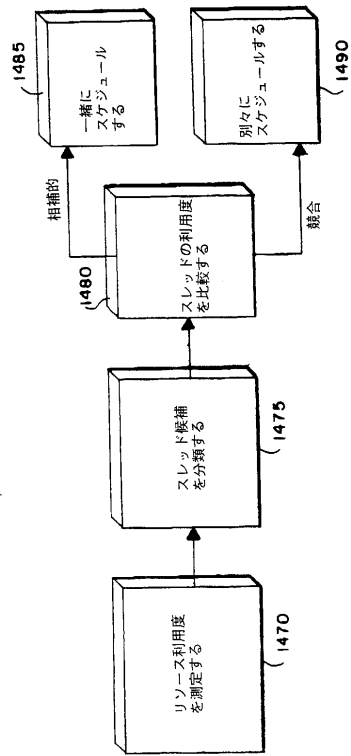
【図 1 3】



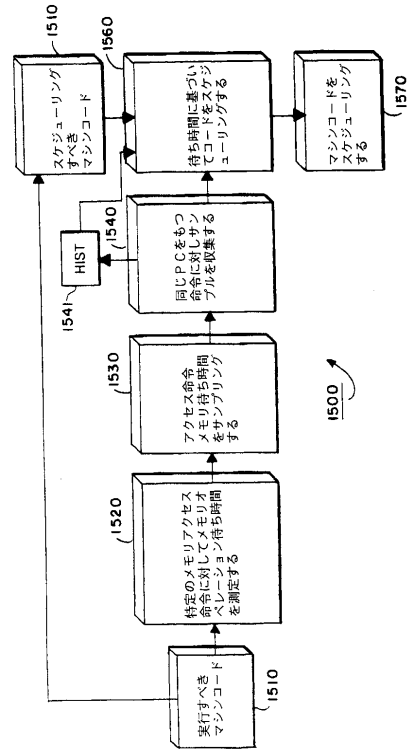
【図 1 4 a】



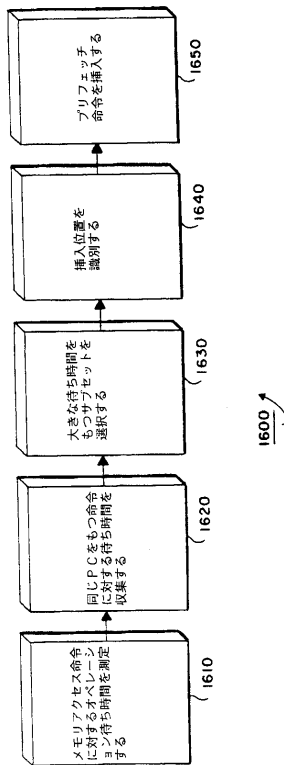
【図 14b】



【図 15】



【図 16】



フロントページの続き

- (72)発明者 ジョージ ゼット クリソス
アメリカ合衆国 マサチューセッツ州 01752 マールボロ ブライアーウッド レーン 5
1
- (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 ダニエル エル ライブホルズ
アメリカ合衆国 マサチューセッツ州 02138 ケンブリッジ リーザーヴォア ストリート 74
- (72)発明者 エドワード ジェイ マックレーラン
アメリカ合衆国 マサチューセッツ州 01746 ホーリストン ワシントン パース 5
- (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 特開昭63-046552(JP,A)
特開平03-263136(JP,A)
特開平08-030494(JP,A)
特開平08-171505(JP,A)
特開平07-064799(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63-025742(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 11/28
G06F 9/38