

(19) 日本国特許庁(JP)

(12) 特許公報(B2)

(11) 特許番号

特許第4763727号
(P4763727)

(45) 発行日 平成23年8月31日(2011.8.31)

(24) 登録日 平成23年6月17日(2011.6.17)

(51) Int.Cl. F I
G O 6 F 9/38 (2006.01) G O 6 F 9/38 3 3 O K

請求項の数 17 (全 10 頁)

(21) 出願番号	特願2007-556350 (P2007-556350)	(73) 特許権者	595020643
(86) (22) 出願日	平成18年2月17日(2006.2.17)		クアルコム・インコーポレイテッド
(65) 公表番号	特表2008-530713 (P2008-530713A)		QUALCOMM INCORPORATED
(43) 公表日	平成20年8月7日(2008.8.7)		アメリカ合衆国、カリフォルニア州 92
(86) 国際出願番号	PCT/US2006/005776		121-1714、サン・ディエゴ、モア
(87) 国際公開番号	W02006/089189		ハウス・ドライブ 5775
(87) 国際公開日	平成18年8月24日(2006.8.24)	(74) 代理人	100091351
審査請求日	平成19年9月19日(2007.9.19)		弁理士 河野 哲
(31) 優先権主張番号	11/061, 981	(74) 代理人	100088683
(32) 優先日	平成17年2月18日(2005.2.18)		弁理士 中村 誠
(33) 優先権主張国	米国 (US)	(74) 代理人	100108855
			弁理士 蔵田 昌俊
		(74) 代理人	100075672
			弁理士 峰 隆司

最終頁に続く

(54) 【発明の名称】 分岐予測ミスを訂正するシステムおよび方法

(57) 【特許請求の範囲】

【請求項 1】

パイプラインプロセッサにおいて分岐予測ミス进行处理する方法において、
分岐命令が予測ミスされたことを検出することと、
前記予測ミスを検出することに対応して、前記パイプラインから、前記分岐命令より先にパイプラインに入力された少なくとも1つの命令を消去することと、
を備えた方法。

【請求項 2】

前記分岐命令より先にパイプラインに入力された少なくとも1つのコミットされていない命令が検出されるなら、前記コミットされていない命令を消去する、請求項1の方法。

10

【請求項 3】

前記消去される命令は、少なくとも1つのコミットされていない命令であり、パイプラインにおいてストールが検出された命令である、請求項2の方法。

【請求項 4】

前記分岐予測を訂正することと、
前記分岐命令を前記パイプラインから消去することと、
をさらに備えた、請求項1の方法。

【請求項 5】

プログラムの順番に前記分岐命令および前記分岐命令より先にパイプラインに入力されたすべての消去された命令をフェッチすることをさらに備えた、請求項4の方法。

20

【請求項 6】

前記分岐命令より先にパイプラインに入力された少なくとも1つの命令を前記パイプラインから消去することは、すべてのコミットされていない命令を前記パイプラインから消去することを備えた、請求項1の方法。

【請求項 7】

少なくとも1つの命令実行パイプラインと、
条件付分岐命令のパイプライン内の評価を予測する分岐予測器と、
前記パイプライン内の命令の順番を追跡する命令順番マネージャーと、
分岐命令が予測ミスしたことを検出することに応答して、前記分岐命令より先にパイプラインに入力された少なくとも1つの命令を前記パイプラインから消去するパイプラインコントローラーと、
を備えたプロセッサ。

10

【請求項 8】

前記分岐予測器は、前記分岐命令が予測ミスされたことを検出することに応答して分岐予測した結果を無効にする、請求項7のプロセッサ。

【請求項 9】

分岐命令より先にパイプラインに入力された少なくとも1つの命令を前記パイプラインから消去することは、すべてのコミットされていない命令を前記パイプラインから消去することを備えた、請求項7のプロセッサ。

【請求項 10】

前記分岐命令が予測ミスされたことを検出することに応答して、前記分岐命令を前記パイプラインから消去することをさらに備えた、請求項7のプロセッサ。

20

【請求項 11】

プログラムの順番に前記分岐命令および前記分岐命令より先にパイプラインに入力されたすべての消去された命令をフェッチすることをさらに備えた、請求項7のプロセッサ。

【請求項 12】

パイプラインプロセッサにおいて、分岐予測ミスを訂正する方法において、
分岐命令が予測ミスされたことを検出することと、
前記分岐命令より先にパイプラインに入力された第1の命令の、待ち時間動作に関する依存状態を検出することと、
すべてのコミットされていない命令を前記パイプラインから消去することと、
を備えた方法。

30

【請求項 13】

前記分岐予測ミスを訂正することをさらに備えた、請求項12の方法。

【請求項 14】

プログラムの順番で、前記分岐命令と前記分岐命令より先にパイプラインに入力されたすべての消去された命令をフェッチすることをさらに備えた、請求項13の方法。

【請求項 15】

パイプラインプロセッサにおいて分岐予測ミスを訂正する方法において、
分岐命令が予測ミスされたことを検出することと、
前記分岐命令が前記パイプライン内の最後のコミットされていない命令かどうか決定することと、
前記分岐命令が前記パイプライン内の最後のコミットされていない命令なら、前記分岐命令をコミットし、すべてのコミットされていない命令を前記パイプラインから消去することと、

40

前記分岐命令が前記パイプライン内の最後のコミットされていない命令でないなら、前記分岐命令より先にパイプラインに入力された命令が待ち時間動作により前記パイプラインにおいてストールしているかどうか決定することと、

前記分岐命令より先にパイプラインに入力された命令が待ち時間動作によりパイプライン

50

ンにおいてストールしているなら、前記分岐命令および全ての他のコミットされていない命令を前記パイプラインから消去することと、
を備えた方法。

【請求項 16】

前記分岐予測ミスを訂正することをさらに備えた、請求項 15 の方法。

【請求項 17】

プログラムの順番に、前記分岐命令と前記分岐命令より先にパイプラインに入力されたすべての消去された命令をフェッチすることをさらに備えた、請求項 15 の方法。

【発明の詳細な説明】

【技術分野】

10

【0001】

この発明は一般にプロセッサの分野に関し、特にブランチ予測ミスにตอบสนองしてプロセッサ実行パイプラインからコミットされていない(uncommitted)命令を消去する方法に関する。

【背景技術】

【0002】

マイクロプロセッサは多種多様なアプリケーションにおいて計算タスクを実行する。改良されたプロセッサ性能は、ソフトウェアの変更を介してより高速な動作および/または増大された機能性を可能にするためにほとんどの場合望ましい。ポータブル電子装置のような多くの埋め込まれたアプリケーションにおいて、電力を節約することはまたプロセッサの設計および実施において、重要な考慮すべき事柄である。

20

【0003】

ごく最近のプロセッサはパイプラインアーキテクチャを採用する。この場合、各々が複数の実行ステップを有するシーケンシャルな命令が実行において重なっている。性能を最大にするために、命令はパイプラインを介して連続的に流れなければならない。しかしながら、命令間のデータの依存度、メモリアクセスに関連する遅延、十分なパイプラインリソースを命令に割り当てることが不能であるといようないろいろな理由により命令はしばしばパイプラインにおいてストールする。パイプラインのストールを最小にすることおよびそれらを効率的に解消することは改良されたプロセッサ性能を得る上で重要なファクターである。

30

【0004】

現実世界のプログラムは条件分岐命令を含む。その分岐動作は一般には、命令がパイプラインにおいて深く評価されるまで知られていない。一般に現代のプロセッサは、分岐予測の種々の形態を採用する。それにより条件分岐命令の分岐動作がパイプラインにおいて早期に予測される。また、プロセッサは、推論的にパイプラインリソースを割り当ておよび/またはフェッチし、分岐予測に基づいて推論的に命令を実行する。実際の分岐動作が決定されると、分岐が予測ミスされたなら、推論的にフェッチされた命令はパイプラインから消去しなければならず、新しい命令が正しい分岐目標アドレスからフェッチされる。予測ミスされた分岐は逆にプロセッサ性能および電力消費に影響を与える。

【0005】

40

一般に、予測ミスされた分岐命令を処理する際に、分岐命令より古いすべての命令、すなわち、分岐命令よりも先にパイプラインに入力された命令は、推論的にフェッチされた命令が消去される前に実行を完了することが可能である。1つ以上のより古い命令が長い待ち時間動作によりパイプラインにおいてストールすると、パイプラインを消去する前に解決される依存状態を待つことは、予測ミスされた分岐のパフォーマンスに不利な条件を悪化させる。

【発明の開示】

【0006】

この発明はパイプライン化されたプロセッサにおいて分岐予測ミスを処理する方法に関する。分岐予測ミスが検出され、分岐命令より古い少なくとも1つの命令が、予測ミス

50

を検出することに対応してパイプラインから消去される。

【 0 0 0 7 】

また、この発明はプロセッサに関する。プロセッサは命令実行パイプラインおよび条件分岐命令のパイプラインにおける評価を予測する分岐予測器を含む。また、プロセッサはパイプライン内の命令の順番を追跡するとともにパイプライン内の命令間の依存状態を追跡する命令順番マネージャーを含む。プロセッサはさらに、分岐命令が予測ミスしたことを検出することに対応して、パイプラインからの分岐命令より古い少なくとも1つの命令を消去するパイプラインコントローラーを含む。

【 0 0 0 8 】

さらに、この発明は、パイプライン化されたプロセッサにおいて分岐予測ミスを訂正する方法に関する。分岐命令予測ミスが検出される。分岐命令がパイプライン内の最後のコミットされない命令かどうか決定される。分岐命令がパイプライン内の最後のコミットされない命令であるなら、分岐命令はコミットされずすべてのコミットされない命令はパイプラインから消去される。分岐命令がパイプライン内の最後のコミットされない命令でないなら、分岐命令より古い命令が長い待ち時間動作によりパイプライン内でストールするかどうか決定される。分岐命令より古い命令が長い待ち時間動作によりパイプライン内で行き詰るなら、分岐命令およびすべての他のコミットされない命令はパイプラインから消去される。

【発明を実施するための最良の形態】

【 0 0 0 9 】

図1はプロセッサ10の機能ブロック図を描画する。プロセッサ10は制御ロジック13に従って命令実行パイプライン12内の命令を実行する。パイプライン12は12aおよび12bのような複数の並列パイプラインを有したスーパースケラであってもよい。パイプライン制御ロジック14は、分岐予測器13および命令順番マネージャー15を含んでいてもよい。パイプライン12a、12bはパイプステージに組織された種々のレジスタまたはラッチ16と、1つ以上の算術論理演算ユニット(ALU) 18を含む。汎用レジスタ(GPR) ファイル20は、メモリヒエラルキーの最上層を備えたレジスタを提供する。パイプライン12a、12bは、命令側変換索引バッファ(I T L B) 24により管理されるメモリアドレッシングおよび許可を用いて命令キャッシュ22から命令をフェッチする。データは、メイン変換索引バッファ(T L B) 28により管理されるメモリアドレッシングおよび許可を用いてデータキャッシュ26からアクセスされる。種々の実施形態において、I T L B は、T L B の部分のコピーを備えていてもよい。

【 0 0 1 0 】

あるいは、I T L B とT L B は統合されてもよい。同様に、プロセッサ10の種々の実施形態において、I キャッシュ22およびD キャッシュ26は、集積されてもよいまたは統合されてもよい。I キャッシュ22および/またはD キャッシュ26におけるミスは、メモリアクセス30の制御の下にメイン(オフチップ) メモリ32へのアクセスを生じる。プロセッサ10は種々の周辺装置36へのアクセスを制御する入力/出力(I / O) インターフェース34を含んでいてもよい。当業者は、プロセッサ10の多数の変形物が可能であることが認識されるであろう。例えば、プロセッサ10は、I キャッシュおよびD キャッシュのどちらからまたは両方のための第2レベル(L 2) キャッシュを含んでいてもよい。さらに、プロセッサ10内に描画された1つ以上の機能ブロックは、特定の実施形態から省略されてもよい。

【 0 0 1 1 】

パイプラインングはよく知られたプロセッサ実施技術である。これにより複数の命令は実行において同時にオーバーラップされる。典型的なアーキテクチャの各命令は、フェッチ、デコード、実行、メモリアクセス、およびライトバックのような複数の実行ステップにおいて実行される。プロセッサパイプライン12は複数の「パイプステージ」から構成される。パイプステージの各々はロジックと記憶エレメント16を備える。これは命令の実行ステップまたは実行ステップの一部を完了する。パイプステージは一緒に接続さ

10

20

30

40

50

れてパイプライン 1 2 を形成する。命令はパイプライン 1 2 に入り、ステージを介して連続的に処理される。前の命令が実行を完了する前に新しい命令がパイプライン 1 2 に入り、それゆえ、いつでも複数の命令がパイプライン 1 2 内で処理されてもよい。シーケンシャル命令ストリームにおける命令間の並列処理を利用するこの能力は改良されたプロセッサ性能に著しく貢献する。理想的な条件の下で、および 1 サイクルで各パイプステージを完了するプロセッサ 1 0 において、パイプライン 1 2 を満たす簡単なイニシャルプロセスに続いて、命令はサイクル毎に実行を完了してもよい。

【 0 0 1 2 】

そのような理想的な条件は、命令間のデータ依存状態（データハザード）、分岐のような制御依存状態（制御ハザード）、プロセッサリソース割り当て衝突（構造ハザード）、割り込み、キャッシュミス、ページフォルト等を含むさまざまなファクターにより実際には実現されない。命令が 2 つのオペランド上で算術または論理演算を実行するとき典型的なデータハザードに遭遇する。この場合、1 つ以上のオペランドが実行を完了しておらず、従って必要とされるオペランドを発生していなかった先行する命令の結果である。より古い命令は他の算術または論理演算がもしれず、またはキャッシュ 2 2、2 6 においてミスをし、メモリアインターフェース 3 0 にオフチップメモリアクセス動作を強制する命令のようなメモリアクセスがもしれない。データハザードはパイプライン 1 2 にストールすることを強制する。

【 0 0 1 3 】

パイプライン化されたプロセッサ 1 0 で遭遇した典型的な制御ハザードは予測ミスした分岐命令である。条件分岐命令は「起きる」かまたは「起きない」かである。起きる場合、命令は異なるプログラムポイントに制御フローを指示する。起きない場合、命令実行はシーケンシャルに進む。分岐条件の評価は、実行パイプステージの期間、パイプライン 1 2 に深く生じる。分岐命令が評価されるまで、プロセッサ 1 0 はどの命令をフェッチしどの命令（すなわち、次のシーケンシャル命令または分岐目標アドレスにおける命令）を次に実行するかを知らない。分岐条件が評価されるまで待つ際の遅延はパイプライン 1 2 にストールを生じる。従って、多くのプロセッサは、例えば従前の条件分岐命令の実行に基いてどのように分岐条件が評価するかを予測するであろう。プロセッサ 1 0 は予測されたアドレスにおいて始まるパイプライン 1 2 に命令をフェッチし、推論的に命令を実行する。予測が正しいとき、パイプラインのストールは回避される。

【 0 0 1 4 】

いくつかの分岐命令は予測された分岐条件と反対の分岐条件を評価するであろう。これは、ここでは、「分岐予測ミス」または「予測ミスされた分岐」と呼ばれる。分岐予測ミスが検出されると、分岐命令よりもより新しいすべての命令（すなわち、分岐予測に基いてフェッチされたすべての命令）はパイプライン 1 2 から消去されなければならない。単一のパイプラインにおいて、どの命令が予測ミスした分岐よりもより新しいかを決定することは正攻法である - 分岐の「後ろの」すべてのパイプステージは消去されなければならない。

【 0 0 1 5 】

図 2 は 2 つの並列実行パイプライン 1 2 a および 1 2 b を有したスーパースケラーパーパイプラインアーキテクチャを描画する。図 2 に描画された状況において、パイプライン 1 2 a 内の命令 A は、オペランド発生、メモリアクセス、またはその他の長い待ち時間動作のような命令 X の依存状態によりストールする。また、命令 A のデータハザードは命令 B をストールさせた。従って、命令 C、D および E が命令キャッシュ 2 2 からフェッチされパイプライン 1 2 b にロードされる。スーパースケラーパープロセッサ 1 0 において、命令実行の順番を実行するために並びに命令間の依存状態を追跡するのにある機構が必要である。

【 0 0 1 6 】

ほとんどのスーパースケラーパープロセッサ 1 0 は、パイプライン制御ロジック 1 4 の一部として順番マネージャ 1 5 を含む。順番マネージャ 1 5 はパイプラインを介した

10

20

30

40

50

命令実行の順番を追跡する。すなわちどの命令が所定の命令よりもより古いかまたはより新しいかを追跡する。順番マネージャー 15 はさらに命令の依存状態を追跡し、例外処理において役に立つ。

【 0 0 1 7 】

パイプステージがインストラクションステップの実行を完了できないときは、例外または割り込みが起きる。例えば、TLB 28 ルックアップが、メモリページがリードオンリであることを示すなら、データをメモリに書く記憶命令は例外を生じるかもしれない。他のタイプの例外は技術的によく知られている。例外に遭遇すると、プロセッサ 10 は、パイプライン 12 (またはスーパースケラアーキテクチャ内のパイプライン 12 a および 12 b) 内のすべての以前のまたはより古い命令を実行しなければならない。また、プロセッサ 10 は、例外を生じる命令とパイプライン 12 a および 12 b からのすべてのより新しい命令を消去する。次にプロセッサ 10 は割り込み処理コードをフェッチし実行する。順番マネージャー 15 はどの命令が「確認され」、どの命令が「コミットされたか」を追跡することによりこのプロセスを支援する。

10

【 0 0 1 8 】

パイプラインハザードがその実行を妨害しないであろう、すなわち、命令がストールしないであろうと決定されたとき、命令が確認される。例えば、両方のオペランドが以前の命令から発生されたことをまたはメモリからフェッチされたことをまた、さもなければ利用可能であることを知られているとき、算術または論理演算を実行する命令が確認されてもよい。

20

【 0 0 1 9 】

その命令およびすべてのより古い命令が確認されると、命令はコミットされる。パイプラインハザードはコミットされた命令(命令自体が確認される)またはコミットされた命令に先行する任意の命令(全てのより古い命令が確認される)を妨害しないので、コミットされた命令は実行を完了することができることが知られている。図 2 を参照すると、命令 A は命令 X の結果の依存状態により確認されない。命令 B はパイプライン 12 a におけるそのような早期のステージにおいて確認される可能性が無い。パイプライン 12 b 内の命令 C は確認されてもよい。これは、ハザードが実行を完了することから命令 C を排除しないことを意味する。しかしながら、命令 C は、命令 C より古いすべての命令すなわち、命令 A および B が確認されるまでコミットすることはできない。

30

【 0 0 2 0 】

例外処理期間中における従来のルールは、例外を生じる命令が「最後のコミットされない命令」であるとき、パイプライン 12 a および 12 b は消去されるというものである。例えば、命令 D が例外を発生するとしたら、命令 X に依存する命令 A の依存状態は、A が確認することを可能にするために、解消されなければならない。A が確認すると、(命令 X が完了すると仮定して) A の前にコミットされていない命令がないなら、A もまたコミットされるであろう。命令 B がパイプライン 12 A を介して進むので、命令 B もまた確認しコミットするなら、命令 A、B および C は確認されるので命令 C がコミットされるであろう。従って、D は最後のコミットされない命令であり、パイプライン 12 a および 12 b からのすべてのより新しい命令(例えば、E)と共に消去されるであろう。コミットされた命令 A、B および C がパイプラインを介して進み実行を完了するので、次に例外処理命令がフェッチされ、パイプライン 12 a および 12 b に供給される。例外を生じる命令がパイプライン 12 a および 12 b 内の最後のコミットされない命令であるように強制することによりプログラム実行における突然の中断は保証される。すなわち、割り込み処理命令がエラーを解消し、プロセッサ 10 の状態を回復すると、命令 D で始まるプログラム実行を再開してもよく、正しい結果を発生するであろう。

40

【 0 0 2 1 】

類似の手続きは、スーパースケラプロセッサ 10 内の予測ミスされた分岐を処理するために適用可能であるように見えるであろう。例えば、図 2 の命令 C はその分岐条件を評価したそして予測ミスされたことが発見された条件分岐命令であると仮定する。命令 D

50

およびEは誤りのある分岐予測に基いてフェッチされた、そしてパイプライン12bから消去されなければならない、正しい分岐目標アドレスからフェッチされた命令と交換される。例外処理ルールの下で、予測ミスされた分岐Cは、Cが最後のコミットされた命令となるまで、すなわち、Xに依存するAの依存状態が解消され、DおよびEを消去する前にAおよびBが確認されコミットされるまで待つであろう。しかしながら、XへのAの依存状態は解消するのに多少の時間を必要とし、予測ミスされた分岐に続く適切な次の命令がフェッチされ実行されるまで時間を遅らせるかもしれない。さらに、パイプライン12aを介してAが再び進むときまでに、AとBがDおよびEと共に消去され再フェッチされたなら、依存状態は解決され、Aが迅速に確認することを可能にする。

【0022】

この発明の一実施形態によって、また図3を参照した記載によって、予測ミスされた分岐が検出されると、(ブロック40)、予測ミスされた分岐が最も古いコミットされていない命令ではないなら(ブロック42)、より古いコミットされていない命令がストールのためにチェックされる(ブロック44)。パイプラインハザード、メモリアクセス、または長い待ち時間動作によるようなストールした命令が検出されるなら、パイプラインコントローラ14は、パイプライン12a、12bからすべてのコミットされていない命令を迅速に消去する(ブロック46)。これは、予測ミスした分岐、予測ミスした分岐よりも古いすべてのコミットされていない命令、および分岐よりもより新しい全ての命令(すなわち、分岐予測ミスに基いて推論的にフェッチされた命令)を含む。分岐予測結果は無効にされ(ブロック48)、消去された、コミットされていない命令が順番に再フェッチされ実行される(ブロック50)。ストールを生じる長い待ち時間動作は、以前にストールした命令が再フェッチされ再実行されるときまでに解決されるかもしれない。しかしながら、たとえそうでなくとも、プロセッサは、正しい分岐目標から命令をフェッチしており、そうする前にストールが解決されるのを待つ必要がない。従ってプロセッサ性能を改良する。

【0023】

予測ミスされた命令が最も古いコミットされていない命令なら(ブロック42)、プロセッサは、予測ミスされた分岐命令を(消去しないように)コミットし、パイプライン12a、12bからすべてのコミットされていない命令を消去する(ブロック52)。これは、予測ミスされた分岐命令より新しいすべての命令、すなわち、誤って予測された分岐経路上の命令を消去する。次に、分岐予測機構が正確に分岐評価を反映し、命令のフェッチと実行が適切な分岐目標アドレスにおいて継続する(ブロック50)ように分岐予測が訂正される(ブロック48)。

【0024】

図3が示すように、予測ミスされた分岐(ブロック40)が最も古いコミットされていない命令(ブロック42)でないならかつより古いコミットされていない命令が長い待ち時間動作によりストールしていないなら、プロセッサは、予測ミスされた分岐をコミットし、すべてのより新しい命令を消去する(ブロック52)前に、すべてのより古い命令がコミットするのを単に待てばよい(ブロック42)。このプロセスは(分岐を消去することよりもむしろ分岐をコミットすることを除いて)上述したように、例外を処理するためにすでに存在する制御ロジックをうまく利用してもよい。

【0025】

あるいは、プロセッサは、予測ミスされた分岐を含むすべてのコミットされていない命令を単に消去し(ブロック46)、コミットされていない命令がストールした場合のように進んでもよい(ブロック48、50)。後者のオプション(図3には示されていないが、YES経路が唯一の制御フロー出口ブロック44であろう)は制御の複雑さを犠牲にして性能を最適化してもよい。ストールした命令の場合に(ブロック44)、新しい命令をコミットすることが中止され、新しい命令をコミットするタスクを同期化することと、コミットされていない命令を消去することが簡単化される。当業者は、どちらのオプションも可能であり正しい結果を生じるであろうことを容易に認識するであろう。

10

20

30

40

50

【0026】

従来のプロセッサデザインプラクティスは、例外を生じる命令または予測ミスされた分岐またはパイプライン消去を引き起こす他の命令より古いすべての命令を実行することである。この発明の例示実施形態によって、予測ミスされた分岐命令より古い1つ以上の命令がパイプラインから消去され、再フェッチされ、実行される。これは、正しくない(予測ミスされた)アドレスからの命令をフェッチすることを即座に終了し、予測ミスを訂正するためにパイプラインハザードの待ち時間を推定的に利用することによりプロセッサ性能と電力消費を改善してもよい。パイプラインハザードを解決するための時間が、ストールした命令を消去し再フェッチするのに必要な時間以上の場合、予測ミスからの回復は、パフォーマンスに不利な条件を招かない。

10

【0027】

この発明は特定の特徴、観点および実施形態に対して記載されたけれども、多数の変形物、変更および他の実施形態がこの発明の広い範囲内で可能であり、従って、すべての変形物、変更および実施形態は、この発明の範囲内に入ると見なされるべきである。それゆえ、この実施形態は、あらゆる面で例証として解釈され、制限するものではないと解釈されるべきであり、添付されたクレームの意味と等価な範囲内に入るすべての変更はクレーム内に包含されることを意図している。

【図面の簡単な説明】

【0028】

【図1】 図1はプロセッサの機能ブロック図である。

20

【図2】 図2は命令キャッシュおよび2つのパイプラインの部分の機能ブロック図である。

【図3】 図3は分岐予測ミス进行处理する方法のフロー図である。

【図1】

図1

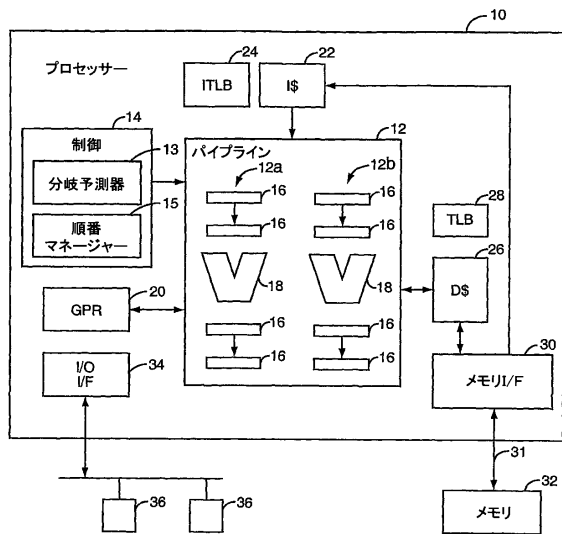


FIG. 1

【図2】

図2

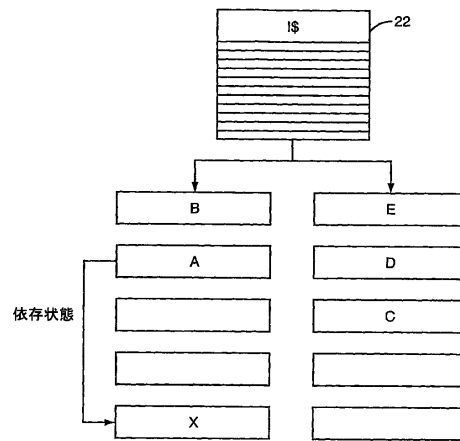


FIG. 2

【図3】

図3

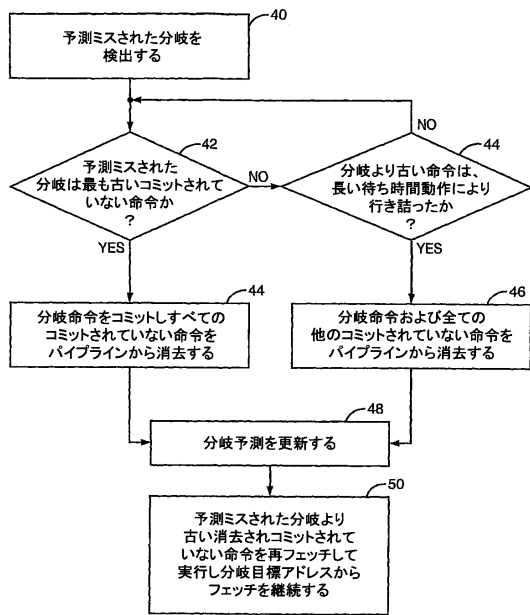


FIG. 3

フロントページの続き

- (74)代理人 100109830
弁理士 福原 淑弘
- (74)代理人 100095441
弁理士 白根 俊郎
- (74)代理人 100084618
弁理士 村松 貞男
- (74)代理人 100103034
弁理士 野河 信久
- (74)代理人 100140176
弁理士 砂川 克
- (74)代理人 100092196
弁理士 橋本 良郎
- (74)代理人 100100952
弁理士 風間 鉄也
- (72)発明者 マシルバイン、マイケル・スコット
アメリカ合衆国、ノースカロライナ州 27614、ローリー、ウェスコット・ドライブ 1300
- (72)発明者 ブリッジス、ジェフリー・トッド
アメリカ合衆国、ノースカロライナ州 27606、ローリー、ティンバーウッド・コート 3513
- (72)発明者 ディーフェンダーファー、ジェームズ・ノリス
アメリカ合衆国、ノースカロライナ州 27539、アベックス、インクベリー・コート 4000
- (72)発明者 サートリウス、トマス・アンドリュウ
アメリカ合衆国、ノースカロライナ州 27614、ローリー、オールド・チムニー・コート 1600

審査官 三坂 敏夫

- (56)参考文献 特開平07-152566(JP,A)
米国特許出願公開第2004/0187119(US,A1)
米国特許第06205542(US,B1)

- (58)調査した分野(Int.Cl., DB名)
G06F 9/38