【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成23年2月17日(2011.2.17)

【公表番号】特表2008-530713(P2008-530713A)
【公表日】平成20年8月7日(2008.8.7)
【年通号数】公開・登録公報2008-031
【出願番号】特願2007-556350(P2007-556350)
【国際特許分類】
Ｇ０６Ｆ　9/38　(2006.01)
【ＦＩ】
Ｇ０６Ｆ　9/38　３３０Ｋ

【誤訳訂正書】
【提出日】平成22年12月22日(2010.12.22)
【誤訳訂正１】
【訂正対象書類名】特許請求の範囲
【訂正対象項目名】全文
【訂正方法】変更
【訂正の内容】
【特許請求の範囲】
【請求項１】
パイプラインプロセッサーにおいて分岐予測ミスを処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応答して、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。
【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。
【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ー。
【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のプロセッサー。
【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のプロセッサー。
【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のプロセッサー。
【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のプロセッサー。
【請求項１２】
パイプラインプロセッサーにおいて、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。
【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ー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、
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。
【誤訳訂正２】
【訂正対象書類名】明細書

【訂正対象項目名】全文
【訂正方法】変更
【訂正の内容】
【発明の詳細な説明】
【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
【技術分野】
【０００１】
この発明は一般にプロセッサーの分野に関し、特にブランチ予測ミスに応答してプロセッサー実行パイプラインからコミットされていない(uncomitted)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ーは多種多様の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ー性能は、ソフトウエ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ー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。
【０００３】
ごく最近のプロセッサー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ー性能を得る上で重要なファクターである。
【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ーは、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、プロセッサーは、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ー性能および電力消費に影響を与える。
【０００５】
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ー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、予測ミスを検出することに応答してパイプラインから消去される。
【０００７】
また、この発明はプロセッサーに関する。プロセッサー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

サー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。プロセッサーはさらに、分岐命令が予測ミスしたことを検出することに応答して、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。

【０００８】

さらに、この発明は、パイプライン化されたプロセッサー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が決定される。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が決定される。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】

【０００９】

図１はプロセッサー１０の機能ブロック図を描画する。プロセッサー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ーラーであってもよい。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ーまたはラッチ１６と、１つ以上の算術論理演算ユニット（ＡＬＵ）１８を含む。汎用レジスター（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ーを提供する。パイプライン１２ａ、１２ｂは、命令側変換索引バッファ（ＩＴＬＢ）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（ＴＬＢ）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、ＩＴＬＢは、ＴＬＢの部分のコピーを備えていてもよい。

【００１０】

あるいは、ＩＴＬＢとＴＬＢは統合されてもよい。同様に、プロセッサー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、集積されてもよいし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、メモリインターフェース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。プロセッサー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、プロセッサー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、プロセッサー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（Ｌ２）キャッシュを含んでいてもよい。さらに、プロセッサー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。

【００１１】

パイプライニングはよく知られたプロセッサー実施技術である。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。プロセッサーパイプライン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れてパイプライン１２を形成する。命令はパイプライン１２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１２に入り、それゆえ、いつでも複数の命令がパイプライン１２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

サー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ー１０において、パイプライン１２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。

【００１２】

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）、プロセッサー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、ページフォー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かもしれず、またはキャッシュ２２、２６においてミスをし、メモリインターフェース３０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かもしれない。データハザードはパイプライン１２にストールすることを強制する。

【００１３】

パイプライン化されたプロセッサー１０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１２に深く生じる。分岐命令が評価されるまで、プロセッサー１０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１２にストールを生じる。従って、多くのプロセッサーは、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。プロセッサー１０は予測されたアドレスにおいて始まるパイプライン１２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。

【００１４】

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１２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。

【００１５】

図２は２つの並列実行パイプライン１２ａおよび１２ｂを有したスーパースケーラ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１２ａ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２２からフェッチされパイプライン１２ｂにロードされる。スーパースケーラープロセッサー１０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。

【００１６】

ほとんどのスーパースケーラープロセッサー１０は、パイプライン制御ロジック１４の一部として順番マネージャー１５を含む。順番マネージャー１５はパイプラインを介した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。

【００１７】

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、ＴＬＢ２８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、プロセッサー１０は、パイプライン１２（またはスーパースケーラーアーキテクチャ内のパイプライン１２ａおよび１２ｂ）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、プロセッサー１０は、例外を生じる命令とパイプライン１２ａおよび１２ｂからのすべてのより新しい命令を消去する。次にプロセッサー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。

【００１８】

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。

【００１９】

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１２ａ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１２ｂ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。

【００２０】

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１２ａおよび１２ｂ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１２Ａ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１２ａおよび１２ｂ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１２ａおよび１２ｂに供給される。例外を生じる命令がパイプライン１２ａおよび１２ｂ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、プロセッサー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。

【００２１】

類似の手続は、スーパースケーラープロセッサー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ず、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ー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、プロセッサーは、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。従ってプロセッサー性能を改良する。

【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）、プロセッサーは、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、プロセッサーは、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、プロセッサーは、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。

【００２６】

従来のプロセッサー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ー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。

【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はプロセッサーの機能ブロック図である。

【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は分岐予測ミスを処理する方法のフロー図である。