(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3871883号
(P3871883)
(45)発行日　平成19年1月24日(2007.1.24)　(24)登録日　平成18年10月27日(2006.10.27)

(51)Int.Cl.　F I
G06F 9/38 (2006.01)　G06F 9/38 330B

請求項の数 23 (全 23 頁)

(21)出願番号　特願2000-571339(P2000-571339)
(86)(22)出願日　平成11年4月2日(1999.4.2)
(65)公表番号　特表2002-525741(P2002-525741A)
(43)公表日　平成14年8月13日(2002.8.13)
(86)国際出願番号　PCT/US1999/007266
(87)国際公開番号　WO2000/017745
(87)国際公開日　平成12年3月30日(2000.3.30)
審査請求日　平成18年2月7日(2006.2.7)
(31)優先権主張番号　09/157,721
(32)優先日　平成10年9月21日(1998.9.21)
(33)優先権主張国　米国(US)

(73)特許権者　591016172
アドバンスト・マイクロ・ディバイシズ・インコーポレイテッド
ＡＤＶＡＮＣＥＤ　ＭＩＣＲＯ　ＤＥＶＩＣＥＳ　ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国、９４０８８－３４５３　カリフォルニア州、サニィベイル、ピィ・オゥ・ボックス・３４５３、ワン・エイ・エム・ディ・プレイス、メイル・ストップ・６８（番地なし）
(74)代理人　100064746
弁理士　深見　久郎
(74)代理人　100085132
弁理士　森田　俊雄

最終頁に続く

(54)【発明の名称】間接分岐ターゲットを計算するための方法

(57)【特許請求の範囲】

【請求項１】

マイクロプロセッサであって、

命令バイトを受取り記憶するよう構成される命令キャッシュを含み、命令キャッシュは、フェッチアドレスを受取り、それに応答してフェッチアドレスに対応する命令バイトの１つ以上を出力するよう構成され、さらに、

複数個の選択子ビットを記憶するよう構成される分岐選択子アレイを含み、各選択子ビットは、命令キャッシュに記憶される特定の命令バイトに対応し、分岐選択子アレイは、フェッチアドレスを受取り、それに応答して予測される次のフェッチアドレスのためのソースを示す１つ以上の対応する選択子ビットを出力するよう構成され、さらに、

フェッチアドレスをフェッチブロックサイズと合計して次のシーケンシャルなアドレスを生成するよう構成されるシーケンシャルアドレス加算器と、

命令キャッシュに記憶される間接分岐命令のための予測される分岐ターゲットアドレスを記憶するよう構成される間接分岐ターゲットキャッシュとを含み、間接分岐ターゲットキャッシュは、フェッチアドレスを受取り、対応する間接分岐アドレスを出力するよう構成され、さらに、

フェッチアドレスを、命令キャッシュによって出力される命令バイトからのオフセットと合計して直接分岐アドレスを計算するよう構成される直接分岐加算器と、

分岐選択子アレイから１つ以上の対応する選択子ビットを受取るよう構成されるマルチプレクサとを含み、マルチプレクサは、選択子ビットに基づいて次のシーケンシャルなア

ドレス、直接分岐アドレス、または間接分岐アドレスのいずれかを選択するよう構成される、マイクロプロセッサ。

【請求項２】

シーケンシャルアドレス加算器、直接分岐加算器、および間接分岐ターゲットキャッシュは各々、次のシーケンシャルなアドレス、直接分岐アドレス、および間接分岐アドレスをそれぞれ並列に計算するよう構成される、請求項１に記載のマイクロプロセッサ。

【請求項３】

命令キャッシュによって出力される命令バイトからのオフセットを計算するよう構成されるオフセット計算論理ユニットをさらに含み、オフセット計算論理ユニットは、選択子ビットに基づいてオフセットを計算するよう構成される、請求項２に記載のマイクロプロセッサ。

【請求項４】

前記選択子ビットの２つが、前記命令キャッシュに記憶される各命令バイトに対応し、前記２バイトは、対応する命令が、（ｉ）分岐命令ノットテイクン、（ii）直接分岐命令テイクン、（iii）間接分岐アドレステイクン、または（iv）リターンアドレステイクンであるかを示す、請求項３に記載のマイクロプロセッサ。

【請求項５】

命令キャッシュは、命令バイトについてのプリデコード情報を記憶するよう構成され、プリデコード情報は、命令開始バイトを示すプリデコードビットを含む、請求項４に記載のマイクロプロセッサ。

【請求項６】

フェッチブロックサイズは定数である、請求項５に記載のマイクロプロセッサ。

【請求項７】

前記マルチプレクサに結合され、コール命令のすぐ後に続く命令に対する命令ポインタを記憶するよう構成されるリターンスタックをさらに含む、請求項６に記載のマイクロプロセッサ。

【請求項８】

前記分岐選択子アレイおよび前記間接分岐ターゲットキャッシュは各々、セットアソシアティブであり、もしあれば、どの記憶された選択子ビットおよび間接分岐ターゲットアドレスがフェッチアドレスに対応するかを決定するためにフェッチアドレスと比較されるタグを記憶するよう構成される、請求項２に記載のマイクロプロセッサ。

【請求項９】

マイクロプロセッサにおける予測される次のフェッチアドレスを生成するための方法であって、

現在のフェッチアドレスを受取るステップと、

命令キャッシュから現在のフェッチアドレスに対応するプリデコード情報および命令バイトを出力するステップと、

分岐選択子アレイから現在のフェッチアドレスに対応する選択子ビットを出力するステップと、

現在のフェッチアドレスをフェッチブロック定数に加算して次のシーケンシャルなアドレスを生成するステップと、

間接分岐ターゲットアドレスキャッシュから記憶された間接分岐アドレスを出力するステップと、

リターンスタックから記憶されたリターンアドレスを出力するステップと、

オフセットを計算するステップと、

現在のフェッチアドレスをオフセットに合計して次の直接分岐アドレスを生成するステップと、

予測される次のフェッチアドレスとして、（ｉ）次のシーケンシャルなアドレス、（ii）出力間接分岐アドレス、（iii）出力リターンアドレス、および（iv）次の直接分岐アドレスのいずれかを選択するステップとを含み、前記現在のフェッチアドレスを前記出力

するステップと、前記記憶された間接分岐アドレスを前記出力するステップと、前記記憶されたリターンアドレスを前記出力するステップと、現在のフェッチアドレスをオフセットに加算して次の直接分岐アドレスを生成する前記ステップとは、すべて並列に行なわれる、方法。

【請求項１０】

命令キャッシュに記憶される命令バイトについての選択子ビットを生成し記憶するステップと、

選択子ビットに基づいて予測される次のフェッチアドレスを選択するステップとをさらに含む、請求項９に記載の方法。

【請求項１１】

命令キャッシュに記憶される分岐命令の結果に基づいて選択子ビットを更新するステップをさらに含む、請求項１０に記載の方法。

【請求項１２】

前記計算するステップは、

現在のフェッチアドレスから、テイクンであると予測される命令キャッシュからの第１の直接分岐命令への第１の特定のオフセットを決定するステップと、

第１の直接分岐命令内に即値データとして記憶される第２の特定のオフセットを検出するステップとを含み、

前記合計するステップは、現在のフェッチアドレスと第１のオフセットと第２のオフセットとを合計するステップを含む、請求項１１に記載の方法。

【請求項１３】

タグを用いて選択子ビットにアクセスするステップをさらに含む、請求項１２に記載の方法。

【請求項１４】

タグを用いて間接分岐ターゲットキャッシュにアクセスするステップをさらに含む、請求項１３に記載の方法。

【請求項１５】

複数個のプリデコードビットを命令キャッシュに記憶するステップをさらに含み、プリデコードビットは、分岐命令の存在と、直接分岐命令に即値データとして記憶されるオフセットバイトの存在とを示す、請求項１４に記載の方法。

【請求項１６】

コンピュータシステムであって、

命令バイトを記憶するよう構成される命令キャッシュ手段を含み、命令キャッシュ手段は、フェッチアドレスを受取り、それに応答してフェッチアドレスに対応する命令バイトの１つ以上を出力するよう構成され、さらに、

複数個の選択子ビットを記憶するよう構成される分岐選択子アレイ手段を含み、各選択子ビットは、命令キャッシュ手段に記憶される特定の命令バイトに対応し、分岐選択子アレイ手段は、フェッチアドレスを受取り、それに応答して予測される次のフェッチアドレスのためのソースを示す１つ以上の対応する選択子ビットを出力するよう構成され、さらに、

フェッチアドレスをフェッチブロックサイズと合計して次のシーケンシャルなアドレスを生成するよう構成されるシーケンシャルアドレス加算器手段と、

命令キャッシュ手段に記憶される間接分岐命令のための予測される分岐ターゲットアドレスを記憶するよう構成される間接分岐ターゲットキャッシュ手段とを含み、間接分岐ターゲットキャッシュ手段は、フェッチアドレスを受取り、対応する間接分岐アドレスを出力するよう構成され、さらに、

フェッチアドレスを、命令キャッシュ手段によって出力される命令バイトからのオフセットと合計して直接分岐アドレスを計算するよう構成される直接分岐加算器手段と、

分岐選択子アレイ手段から１つ以上の対応する選択子ビットを受取るよう構成される選択子手段とを含み、選択子手段は、選択子ビットに基づいて次のシーケンシャルなアドレ

ス、直接分岐アドレス、または間接分岐アドレスのいずれかを選択するよう構成される、コンピュータシステム。

【請求項１７】

シーケンシャルアドレス加算器手段、直接分岐加算器手段、および間接分岐ターゲットキャッシュ手段は各々、次のシーケンシャルなアドレス、直接分岐アドレス、および間接分岐アドレスをそれぞれ並列に計算するよう構成される、請求項１６に記載のコンピュータシステム。

【請求項１８】

命令キャッシュ手段によって出力される命令バイトからのオフセットを計算するよう構成されるオフセット計算論理ユニットをさらに含み、オフセット計算論理ユニットは、選択子ビットに基づいてオフセットを計算するよう構成される、請求項１７に記載のコンピュータシステム。

【請求項１９】

前記選択子ビットの２つが、前記命令キャッシュ手段に記憶される各命令バイトに対応し、前記２バイトは、対応する命令が、（ｉ）分岐命令ノットテイクン、（ii）直接分岐命令テイクン、（iii）間接分岐アドレステイクン、または（iv）リターンアドレステイクンであるかを示す、請求項１８に記載のコンピュータシステム。

【請求項２０】

命令キャッシュ手段は、命令バイトについてのプリデコード情報を記憶するよう構成され、プリデコード情報は、命令開始バイトを示すプリデコードビットを含む、請求項１９に記載のコンピュータシステム。

【請求項２１】

フェッチブロックサイズは定数である、請求項２０に記載のコンピュータシステム。

【請求項２２】

前記選択子手段に結合され、コール命令のすぐ後に続く命令に対する命令ポインタを記憶するよう構成されるリターンスタックをさらに含む、請求項２１に記載のコンピュータシステム。

【請求項２３】

前記分岐選択子アレイ手段および前記間接分岐ターゲットキャッシュ手段は各々、セットアソシアティブであり、もしあれば、どの記憶された選択子ビットおよび間接分岐ターゲットアドレスがフェッチアドレスに対応するかを決定するためにフェッチアドレスと比較されるタグを記憶するよう構成される、請求項１７に記載のコンピュータシステム。

【発明の詳細な説明】

【０００１】

【発明の背景】

１．技術分野

この発明は、マイクロプロセッサにおける分岐ターゲットアドレスを記憶することに関し、特に、間接分岐ターゲットアドレスをキャッシュするための方法に関する。

【０００２】

２．背景技術

Sakamoto M et alによる、「スーパースカラプロセッサにおける分岐ペナルティを低減するためのマイクロアーキテクチャサポート」（“Microarchitecture Support for Reducing Branch Penalty in a Superscalar Processor”）と題する、１９９６年１０月７日、オースティンにおける、コンピュータおよびプロセッサにおけるコンピュータ設計ＩＣＣＶＬＳＩに関する国際会議議事録（Proceedings of the International Conference on Computer Design. ICC VLSI in Computers and Processors）の第２０８から２１６頁は、動的分岐予測方式を実現し分岐履歴テーブルを有するマイクロプロセッサを開示する。分岐ターゲットバッファが、先に実行されたテイクンのジャンプ命令の結果を維持する。ジャンプ命令は、無条件ジャンプ命令、条件付き分岐命令および間接ジャンプ命令を含む。

Chang P-Y et alによる「間接ジャンプのためのターゲット予測」（“Target Prediction for Indirect Jumps”）と題する、コンピュータ・アーキテクチャ・ニュース（Computer Architecture News）、第２５巻、第２号、１９９７年５月１日、第２７４頁から２８３頁は、間接ジャンプターゲットを予測するためにターゲットキャッシュを有する予測メカニズムを採用するスーパースカラプロセッサアーキテクチャを開示する。
ＷＯ　９８／２２８７３は、分岐予測を選択するために分岐選択子を採用する分岐予測メカニズムを開示する。分岐予測：シーケンシャル、リターンスタック、分岐予測１および分岐予測２のための分岐選択子が開示される。
今日のマイクロプロセッサ設計者は、絶えず製品の性能を向上させようとしている。マイクロプロセッサの性能を増大させる方法の１つは、それが動作するクロック周波数を増大させることである。しかしながら、クロック周波数が増大すると、マイクロプロセッサ内の各機能ユニットは、その指定されたタスクを行なうのにかかる時間が少なくなる。したがって、所与のクロック周波数で性能をさらに増大させるために他の方法が採用される。並列実行はそのような方法の１つである。並列実行は、各クロックサイクル中２つ以上の命令を実行することを伴う。並列実行を実現するために、多くのマイクロプロセッサは、各々が独立に並列して実行するよう構成された並列機能ユニットを用いる。

【０００３】
所定のクロックサイクルでマイクロプロセッサの性能を増大させるための別の方法は、アウトオブオーダ実行である。たいていのプログラムは、特定の順序で実行する命令に依存する。しかしながら、プログラム内のある種の命令は、所望の機能がなお達成されるのであればプログラム順序を飛越して実行してもよい。アウトオブオーダで実行可能な命令は、そのまわりの命令に「依存」しない命令である。アウトオブオーダ実行を実現するマイクロプロセッサは、どの命令が他の命令に依存しないか、すなわちアウトオブオーダ実行が可能であるかを決定することができる。

【０００４】
命令をアウトオブオーダで実行するのには、いくつかの潜在的な欠点がある。特に、欠点の１つは、分岐命令の使用である。分岐命令は、条件付きジャンプ命令（Ｊｃｃ）、無条件ジャンプ命令（ＪＭＰ）、リターン命令（ＲＥＴ）、およびサブルーチンコール命令（ＣＡＬＬ）を含む。マイクロプロセッサは分岐命令後の命令が実行されるべきかどうかを決定できないことがあるので、これらの命令は、アウトオブオーダ実行を特に困難にする。

【０００５】
所定のクロックサイクルでマイクロプロセッサの性能を増大させるのに使用される別の方法は、パイプライン化である。パイプライン化は、命令を実行するためにマイクロプロセッサが完了しなければならないタスクを分割することを伴う。タスクは、命令処理「パイプライン」を形成する複数のステージに分割される。ステージは、典型的には、一般的な連続処理形式になる。たとえば、第１のステージは、命令キャッシュに記憶された命令を「フェッチ」することかもしれないし、第２の段階は、命令を整列させ、それをデコードユニットに運んでデコードするか、または機能ユニットに運んで実行することかもしれない。パイプラインによって、命令の長いストリームが、より少ないクロックサイクルで実行されることが可能になる。パイプライン化されたマイクロプロセッサの利点は、一度に２つ以上の命令を処理する能力による。たとえば、第２の命令が命令キャッシュからフェッチされているのと同時に、第１の命令をデコードすることが可能である。

【０００６】
しかしながら、アウトオブオーダ実行と同様、パイプライン化されたマイクロプロセッサにおいて、分岐命令の結果として性能が後退することがある。分岐命令はパイプラインを「ストール」させてしまうことがある、なぜなら、パイプライン処理は、分岐命令が実行されてしまう後までどの命令をフェッチするべきかを決定することができないからである。たとえば、条件付き分岐命令は、「テイクン」または「ノットテイクン」のいずれかである。分岐命令がノットテイクンであれば、プログラム順序で分岐命令のすぐ後に続く命

令が実行される。代わりに、分岐命令がテイクンであれば、分岐ターゲットアドレス相対するあるオフセットの異なった命令が実行される。パイプライン化されたマイクロプロセッサにおいては、分岐命令が実行を完了するまで、パイプラインの初期ステージまたはフェッチステージにおいては、「テイクン」または「ノットテイクン」に従ったターゲット命令をフェッチするかどうかを決定するのが困難である。したがって、マイクロプロセッサのパイプライン処理は、初期ステージが分岐命令の結果を待つ間、ストールするか、または「バブル」（すなわち、パイプラインを介して伝搬するアイドルクロックサイクル）を形成してしまうことがある。

【０００７】

パイプラインにおいてストールまたはバブルを形成するのを防ぐために、多くのプロセッサは、分岐予測方式を実現する。分岐予測は、分岐命令が実際に実行を完了する前に、各分岐命令がテイクンまたはノットテイクンのいずれであるかを予測することを伴う。複数の異なったアルゴリズムが使用され得るが、多くは、特定の分岐命令のテイクン／ノットテイクン履歴を記憶することに依存する。たとえば、分岐命令が実行されテイクンである場合、分岐予測ハードウェアは、その情報を記憶し、その結果、分岐命令が次にフェッチされるとき、パイプラインは、その先行のパフォーマンスに基づいて命令がテイクンであると自動的に推測するであろう。

【０００８】

分岐予測情報を記憶することは、ストールまたはバブルの形成の確立を低減するが、それは不利益も有する。不利益の１つは、分岐履歴情報を維持し更新するのに必要とされる複雑なハードウェアである。第２の不利益は、分岐履歴情報を記憶するのに必要とされるスペースである。マイクロプロセッサのダイスペースは、比較的乏しい資源であるので、分岐予測を記憶するために多量のスペースを用いることは不利益である。命令キャッシュのサイズが大きくなると、そこにより多くの分岐命令を記憶することが可能である。これは代わりに、分岐履歴予測情報の記憶空間をより多く必要とする。さらに、現在のソフトウェアは、高集中の分岐命令（たとえば、４命令ごとに１つ）を有する傾向がある。この高集中は、いくつの分岐が命令キャッシュに記憶されるかを正確に知ることができないということと合わさって、典型的には、分岐履歴情報を記憶するのに充てられるダイスペースの量を多くしてしまう。

【０００９】

これらの理由のために、分岐予測および履歴情報を効率的に記憶するための方法が所望される。特に、分岐履歴および予測情報のために必要とされる記憶空間の量を低減するための方法が特に望ましい。

【００１０】

【発明の開示】

上述した課題は、この発明に従うマイクロプロセッサおよび分岐予測ユニットによって部分的に解決可能である。ある実施例では、分岐予測情報を効率的に記憶するよう構成されたマイクロプロセッサは、命令キャッシュおよび分岐予測ユニットを含み得る。分岐予測ユニットは、命令キャッシュに結合されてもよく、間接分岐ターゲットキャッシュ、直接分岐加算器、および分岐選択子アレイを含んでもよい。間接分岐ターゲットアレイは、間接分岐命令のための予測される分岐ターゲットアドレスを記憶するよう構成可能であり、直接分岐加算器は、直接分岐命令のための分岐ターゲットアドレスを計算するよう構成可能である。分岐選択子アレイは、間接分岐ターゲットキャッシュに結合されてもよく、複数個の選択子ビットを記憶するよう構成されてもよい。ある実施例では、選択子ビットは、命令キャッシュに記憶される特定の命令バイトに対応してもよい。選択子ビットは、特定の命令バイトと関連づけられる予測される次のフェッチアドレスのためのソースを示す。

【００１１】

別の実施例では、分岐予測ユニットは、直接分岐加算器、間接分岐ターゲットキャッシュ、または別のソースのいずれかから予測される次のフェッチアドレスを選択するよう構成

されるマルチプレクサをさらに含んでもよい。マルチプレクサは、現在のフェッチアドレスに対応する選択子ビットの値に基づいてその選択をするよう構成可能である。分岐予測ユニットは、マルチプレクサに結合されるシーケンシャルアドレス加算器も含んでよい。シーケンシャルアドレス加算器は、現在のフェッチアドレスの一部と、命令キャッシュのフェッチブロックまたはキャッシュラインサイズに対応する定数とを合計することにより、予測される次のシーケンシャルなフェッチアドレスを生成するよう構成可能である。コール命令のすぐ後に続く命令に対応するフェッチアドレスを記憶するために、リターンスタックを利用してもよい。

【００１２】

マイクロプロセッサにおいて予測される次のフェッチアドレスを記憶するための方法も企図される。ある実施例では、方法は、間接分岐命令のための予測される分岐ターゲットアドレスを間接分岐ターゲットキャッシュに記憶することを含み得る。加えて、特定の命令バイトに対応する選択子ビットが、分岐選択子アレイに記憶される。直接分岐命令については、予測される直接分岐ターゲットアドレスは、「オンザフライ」で計算される。一旦、予測される間接分岐アドレスが参照され、予測される直接分岐アドレスが計算されると、予測される次のフェッチアドレスは選択子ビットに基づいて選択される。

【００１３】

ある実施例では、プリデコードビットが、命令バイトとともに計算され記憶される。プリデコードビットは、予測される直接分岐ターゲットアドレスを計算するために使用されてもよく、これは、現在のフェッチアドレスからテイクンであると予測される第１の直接分岐命令への第１のオフセットを決定し、直接分岐命令内に即値データとして記憶される第２のオフセットを検出し、次に現在のフェッチアドレスと第１のオフセットと第２のオフセットとを合計することによって計算されてもよい。

【００１４】

別の実施例では、予測されるシーケンシャルなアドレスは、現在のフェッチアドレスの一部を、命令キャッシュのキャッシュラインサイズまたはフェッチブロックサイズを表わす定数と合計することにより計算されてもよい。

【００１５】

分岐予測情報を効率的に記憶するよう構成されるコンピュータシステムも企図される。ある実施例では、コンピュータシステムは、上述したようなマイクロプロセッサとバスブリッジとを含む。バスブリッジは、高速ＣＰＵバスを介してマイクロプロセッサに結合されてもよい。モデムなどの周辺装置がバスブリッジを介してマイクロプロセッサに結合されてもよい。別の実施例では、コンピュータシステムは、ＣＰＵバスを介して第１のマイクロプロセッサに結合される第２のマイクロプロセッサを含んでもよい。

【００１６】

この発明の他の目的および利点は、添付の図面を参照し以下の詳細な説明を読むと明らかになるであろう。

【００１７】

この発明は、さまざまな変形および代替の形を許すが、その具体的な実施例が図に例として示され、ここに詳細に記載される。しかしながら、図面およびその詳細な説明は、この発明を開示される特定の形に制限するものではなく、反対に、その意図は、前掲の特許請求の範囲によって定義されるこの発明の精神および範囲内にあるすべての変形、等価および代替を含むものであることが理解される。

【００１８】

【発明を実施する態様】

ここで図１を参照して、例のマイクロプロセッサ１０のある実施例のブロック図が示される。マイクロプロセッサ１０は、プリフェッチ／プリデコードユニット１２と、分岐予測ユニット１４と、命令キャッシュ１６と、命令整列ユニット１８と、複数個のデコードユニット２０Ａ～２０Ｃと、複数個のリザベーションステーション２２Ａ～２２Ｃと、複数個の機能ユニット２４Ａ～２４Ｃと、ロード／ストアユニット２６と、データキャッシュ

２８と、レジスタファイル３０と、リオーダバッファ３２と、ＭＲＯＭユニット３４とを含む。文字が後に付された特定の参照番号とともに参照される要素は、参照番号のみにより集合的に参照されることがある。たとえば、リザベーションステーション２２Ａ～２２Ｃは、リザベーションステーション２２として集合的に参照されることがある。

【００１９】

プリフェッチ／プリデコードユニット１２は、主メモリサブシステム（図示せず）から命令を受けるよう結合され、命令キャッシュ１６および分岐予測ユニット１４にさらに結合される。同様に、分岐予測ユニット１４は、命令キャッシュ１６に結合される。分岐予測ユニット１４は、命令整列ユニット１８および機能ユニット２４Ａ～Ｃにも結合される。命令キャッシュ１６はさらに、ＭＲＯＭユニット３４および命令整列ユニット１８に結合される。命令整列ユニット１８は、ロード／ストアユニット２６およびそれぞれのデコードユニット２０Ａ～Ｃに結合される。デコードユニット２０Ａ～Ｃはそれぞれリザベーションステーション２２Ａ～Ｃに結合され、これはさらにそれぞれの機能ユニット２４Ａ～Ｃに結合される。加えて、命令整列ユニット１８およびリザベーションステーション２２は、レジスタファイル３０およびリオーダバッファ３２に結合される。機能ユニット２４は、ロード／ストアユニット２６、レジスタファイル３０、およびリオーダバッファ３２にも結合される。データキャッシュ２８は、ロード／ストアユニット２６および主メモリサブシステムに結合される。最後に、ＭＲＯＭユニット３４は、命令整列ユニット１８に結合される。

【００２０】

命令は、プリフェッチ／プリデコードユニット１２によって主メモリからプリフェッチされる。プリフェッチ／プリデコードユニット１２は、可変長の命令を固定長の命令にプリデコードし、これは次に命令キャッシュ１６に記憶される。命令は、プリフェッチ方式を用いることによって実際に必要とされるより前にプリフェッチおよびプリデコードされてもよい。さまざまなプリフェッチ方式がプリフェッチ／プリデコードユニット１２によって採用され得る。

【００２１】

マイクロプロセッサ１０は、条件付き分岐命令の後の命令を投機的にフェッチするために分岐予測を採用してもよい。分岐予測ユニット１４は、分岐予測動作を行なうために含まれる。プリフェッチ／プリデコードユニット１２は、特定の命令ラインがプリデコードされるとき初期分岐ターゲットを決定する。キャッシュラインに対応する分岐ターゲットへのその後の更新は、キャッシュライン内の命令の実行のために生じ得る。命令キャッシュ１６は、分岐予測ユニット１４にフェッチされる命令アドレスの表示を与える。これにより、分岐予測ユニット１４は、分岐予測を形成するとき、どの分岐ターゲットアドレスを選択すべきかを決定することが可能になる。命令整列ユニット１８および機能ユニット２４は、分岐予測ユニット１４に更新情報を与える。命令整列ユニット１８は、分岐予測ユニット１４によって予測されなかった分岐命令を検出するよう構成され得る。機能ユニット２４は、分岐命令を実行し、予測された分岐方向が予測誤りでなかったかどうかを決定する。分岐方向は「テイクン」であるかもしれず、その場合にはその後の命令は分岐命令のターゲットアドレスからフェッチされる。反対に、分岐方向は「ノットテイクン」であるかもしれず、その場合にはその後の命令は、分岐命令に連続したメモリ場所からフェッチされる。予測誤りされた分岐命令が検出されると、予測誤りされた分岐の後の命令は、マイクロプロセッサ１０のさまざまなユニットから廃棄される。さまざまな好適な分岐予測アルゴリズムが分岐予測ユニット１４によって採用され得る。分岐予測ユニット１４および命令キャッシュ１６とのその相互作用のさらなる詳細を記載する前に、例示のマイクロプロセッサ１０の一般的局面および可能な実施例を記載する。

【００２２】

命令キャッシュ１６は、プリフェッチ／プリデコードユニット１２から受取られる命令を記憶するために設けられる高速キャッシュメモリである。記憶された命令は次に、命令キャッシュ１６からフェッチされ、命令整列ユニット１８に転送される。ある実施例では、

命令キャッシュ１６は、セットアソシアティブ構造としてもよい。命令キャッシュ１６は、アクセス時間を速めるためにウェイ予測方式をさらに採用してもよい。たとえば、命令の各ラインを特定するタグにアクセスし、タグをフェッチアドレスと比較してウェイを選択する代わりに、命令キャッシュ１６は、アクセスされるウェイを予測してもよい。この態様では、ウェイは、アレイにアクセスするより前に投機的に選択される。ウェイ予測を用いると、命令キャッシュ１６のアクセス時間は、ダイレクトマップドのキャッシュと同様になり得る。命令バイトが読出された後、ベリファイのためにタグ比較が行なわれる。ウェイ予測が正しくなければ、正しい命令バイトがフェッチされ、正しくない命令バイト（処理パイプラインのさらに下にある）は廃棄される。なお、命令キャッシュ１６は、フルアソシアティブ構成、セットアソシアティブ構成、またはダイレクトマップド構成で実現可能である。

【００２３】

ＭＲＯＭユニット３４は、「高速経路」（fast-path）命令のシーケンスを記憶するよう構成されるリードオンリメモリである。高速経路命令は、デコーダ２０Ａ～Ｃおよび機能ユニット２４Ａ～Ｃによってデコードおよび実行可能である命令である。対照的に、「ＭＲＯＭ命令」は、デコーダ２０Ａ～Ｃおよび機能ユニット２４Ａ～Ｃによって直接デコードおよび実行されるには複雑すぎる命令である。命令キャッシュ１６がＭＲＯＭ命令を出力すると、ＭＲＯＭユニット３４は、高速経路命令のシーケンスを出力することによって応答する。より具体的には、ＭＲＯＭユニット３４は、ＭＲＯＭ命令を構文解析し定義された高速経路命令のサブセットに変換して、所望の演算を実行する。ＭＲＯＭユニット３４は、高速経路命令のサブセットをデコードユニット２０Ａ～Ｃにディスパッチする。

【００２４】

命令バイトが一旦命令キャッシュ１６からフェッチされると、それらは命令整列ユニット１８に運ばれる。命令整列ユニット１８は、命令をデコードユニット２０Ａ～Ｃの１つに経路付ける。レジスタオペランド情報も検出されレジスタファイル３０およびリオーダバッファ３２に経路付けられる。加えて、命令が１つ以上のメモリ操作が行なわれることを必要とする場合、命令整列ユニット１８は、メモリ操作をロード／ストアユニット２６にディスパッチする。各デコードされた命令は、命令とともに含まれ得る変位または即値データおよびオペランドアドレス情報とともに、リザベーションステーション２２にディスパッチされる。

【００２５】

マイクロプロセッサ１０は、アウトオブオーダ実行をサポートし、したがって以下のタスクのためにリオーダバッファ３２を採用する。タスクは、レジスタ読出動作およびレジスタ書込動作のために元のプログラムシーケンスを追跡し、レジスタリネーミングを実現し、投機的命令実行および分岐予測誤り復旧を可能にし、正確な例外を容易にするものである。リオーダバッファ３２内の一時的記憶場所は、レジスタの更新を伴う命令のデコードの際に確保される。一時的記憶場所は、命令の投機的実行から生じる投機的レジスタ状態を記憶する。分岐命令が正しくなければ、予測誤りされた経路に沿った、投機的に実行された命令から生ずる結果は、それらがレジスタファイル３０に書込まれる前に、リオーダバッファ３２において無効化可能である。同様に、特定の命令が例外を引き起こした場合、例外を引き起こした命令の後の命令が廃棄され得る。この態様では、例外は「正確」である（すなわち、例外を引き起こした命令の後の命令は、例外より前に完了されない）。なお、特定の命令は、プログラム順序でその特定の命令に先行する命令より前にそれが実行されるならば、投機的に実行される。先行する命令は、分岐命令または例外を引き起こす命令であり得、その場合には投機的結果はリオーダバッファ３２によって廃棄され得る。

【００２６】

命令整列ユニット１８の出力において与えられる即値または変位データおよびデコードされた命令は、それぞれのリザベーションステーション２２に直接経路づけられる。ある実施例では、各リザベーションステーション２２は、対応する機能ユニットへの発行を待っ

ている最大３つのペンディング命令についての命令情報（すなわち、デコードされた命令ならびにオペランド値、オペランドタグおよび／または即値データ）を保持することができる。なお、図面に示す実施例では、各リザベーションステーション２２は、専用の機能ユニット２４と関連づけられる。したがって、３つの専用の「発行位置」がリザベーションステーション２２および機能ユニット２４によって形成される。言い換えれば、発行位置０は、リザベーションステーション２２Ａおよび機能ユニット２４Ａによって形成される。リザベーションステーション２２Ａに整列されディスパッチされる命令は、機能ユニット２４Ａによって実行される。同様に、発行位置１は、リザベーションステーション２２Ｂおよび機能ユニット２４Ｂによって形成され、発行位置２は、リザベーションステーション２２Ｃおよび機能ユニット２４Ｃによって形成される。

【００２７】

特定の命令のデコードの際、必要とされるオペランドがレジスタ場所であるならば、レジスタアドレス情報はリオーダバッファ３２およびレジスタファイル３０に同時に経路づけられる。ｘ８６レジスタファイルは、８個の３２ビット実レジスタ（すなわち、典型的には、ＥＡＸ、ＥＢＸ、ＥＣＸ、ＥＤＸ、ＥＢＰ、ＥＳＩ、ＥＤＩおよびＥＳＰと呼ばれる）を含む。ｘ８６マイクロプロセッサアーキテクチャを採用するマイクロプロセッサ１０のある実施例では、レジスタファイル３０は、３２ビットの実レジスタの各々について記憶場所を含む。さらなる記憶場所が、ＭＲＯＭユニット３４によって使用されるレジスタファイル３０内に含められてもよい。リオーダバッファ３２は、これらのレジスタの内容を変更する結果のための一時的記憶場所を含み、それによりアウトオブオーダ実行を可能にする。リオーダバッファ３２の一時的記憶場所は、デコードされると、実レジスタの１つの内容を変更するよう決定される各命令のために確保される。したがって、特定のプログラムの実行中のさまざまな時点で、リオーダバッファ３２は、所定のレジスタの、投機的に実行された内容を含む１つ以上の場所を含み得る。

【００２８】

所定の命令のデコードの後に、リオーダバッファ３２が所定の命令におけるオペランドとして用いられたレジスタに割当てられた先の場所を有していることが決定されれば、リオーダバッファ３２は、対応するリザベーションステーションに、１）最も最近に割当てられた場所の値、または２）値が、やがて先の命令を実行するであろう機能ユニットによってまだ発生されていなければ、最も最近に割当てられた場所についてのタグ、のいずれかを転送する。リオーダバッファ３２が所与のレジスタのために確保された場所を有していれば、オペランド値（またはリオーダバッファタグ）は、レジスタファイル３０からではなくリオーダバッファ３２から与えられる。リオーダバッファ３２中に、必要とされるレジスタのために確保された場所がなければ、値はレジスタファイル３０から直接とられる。オペランドがメモリ場所に対応していれば、オペランド値はロード／ストアユニット２６を介してリザベーションステーションに与えられる。

【００２９】

ある特定の実施例では、リオーダバッファ３２は、ユニットとして、同時にデコードされた命令を記憶し操作するよう構成される。この構成を、ここでは「ライン志向の」と呼ぶ。いくつかの命令を併せて操作することによって、リオーダバッファ３２内に採用されるハードウェアは簡素化され得る。たとえば、この実施例に含まれるライン志向のリオーダバッファは、１つ以上の命令が命令整列ユニット１８によってディスパッチされるたびに、３つの命令に属する命令情報のために十分な記憶場所を割当てる。対照的に、実際にディスパッチされる命令の数に依存して、可変量の記憶装置が、従来のリオーダバッファにおいては割当てられる。可変量の記憶装置を割当てるには、比較的多数の論理ゲートが必要とされ得る。同時にデコードされた命令の各々が実行すると、命令結果は同時にレジスタファイル３０に記憶される。記憶装置は、同時にデコードされた命令の別のセットへの割当てのために解放される。加えて、制御論理はいくつかの同時にデコードされた命令にわたって償却されるので、命令当り採用される制御論理回路の量は低減される。特定の命令を特定するリオーダバッファタグは、２つのフィールドである：ラインタグおよびオフ

セットタグに分割されてもよい。ラインタグは、特定の命令を含む同時にデコードされた命令のセットを特定し、オフセットタグは、セット内のどの命令が特定の命令に対応するかを特定する。なお、命令結果をレジスタファイル３０に記憶し対応する記憶装置を解放することを、命令を「リタイア」するという。さらになお、いかなるリオーダバッファ構成をマイクロプロセッサ１０のさまざまな実施例において採用してもよい。

【００３０】

前述したように、リザベーションステーション２２は、命令が対応する機能ユニット２４によって実行されるまで命令を記憶する。（ｉ）命令のオペランドが与えられ、かつ（ii）同じリザベーションステーション２２Ａ～２２Ｃ内にあり、かつプログラム順序で命令より先にある命令についてオペランドがまだ与えられていなければ、命令が実行のために選択される。なお、命令が機能ユニット２４の１つによって実行されると、その命令の結果は、その結果を待っているリザベーションステーション２２のいずれかに直接送られ、同時に結果はリオーダバッファ３２を更新するために送られる（この技術は、一般的には、「結果転送」と言われる）。命令は、関連づけられた結果が転送されるクロックサイクル中に、実行のために選択され機能ユニット２４Ａ～２４Ｃに送られることが可能である。リザベーションステーション２２は、この場合には、転送された結果を機能ユニット２４に経路付ける。

【００３１】

ある実施例では、各機能ユニット２４Ａ～Ｃは、加算および減算、ならびにシフト、ローテート、論理演算、および分岐動作を行なうよう構成される。なお、浮動小数点演算を可能にするために浮動小数点ユニット（図示せず）を採用してもよい。浮動小数点ユニットは、コプロセッサとして動作し、ＭＲＯＭユニット３４から命令を受取り、その後にリオーダバッファ３２と交信して命令を完了し得る。さらに、機能ユニット２４は、ロード／ストアユニット２６によって行なわれるロードメモリ操作およびストアメモリ操作のためにアドレス生成を行なうよう構成され得る。

【００３２】

機能ユニット２４の各々はまた、条件付き分岐命令の実行に関する情報を分岐予測ユニット１４に与える。分岐予測が正しくなかったならば、分岐予測ユニット１４は、命令処理パイプラインに入った予測誤りされた分岐の後の命令をフラッシュ（flush）し、命令キャッシュ１６または主メモリからの必要とされる命令のフェッチをもたらす。なお、そのような状況では、投機的に実行され、ロード／ストアユニット２６およびリオーダバッファ３２に一時的に記憶されたものを含む、予測誤りされた分岐命令の後に起こる元のプログラムシーケンスにおける命令の結果は廃棄される。

【００３３】

機能ユニット２４によって発生された結果は、レジスタ値が更新されているならばリオーダバッファ３２に、メモリ場所の内容が変更されているならばロード／ストアユニット２６に送られる。結果がレジスタに記憶されるべきものであれば、リオーダバッファ３２は、命令がデコードされたとき、レジスタの値のために確保された場所に結果を記憶する。複数個の結果バス３８が、機能ユニット２４およびロード／ストアユニット２６からの結果の転送のために含まれる。結果バス３８は、生成された結果、および実行された命令を特定するリオーダバッファタグを運ぶ。

【００３４】

ロード／ストアユニット２６は、機能ユニット２４とデータキャッシュ２８との間のインターフェイスを与える。ある実施例では、ロード／ストアユニット２６は、ペンディングのロードまたはストアについてのデータおよびアドレス情報のために８個の記憶場所を有するロード／ストアバッファで構成される。バッファが一杯になると、命令整列ユニット１８は、ロード／ストアユニット２６がペンディングのロードまたはストア要求情報のための空きを有するまで待つ。ロード／ストアユニット２６はまた、ペンディングのストアメモリ操作に対するロードメモリ操作の依存性チェックを行ない、データコヒーレンシが確実に維持されるようにする。メモリ操作は、マイクロプロセッサ１０と主メモリサブシ

ステムとの間のデータの転送である。メモリ操作は、メモリに記憶されるオペランドを利用する命令の結果であってもよく、データ転送をもたらすが他の操作をもたらさないロード／ストア命令の結果であってもよい。加えて、ロード／ストアユニット２６は、セグメントレジスタと、ｘ８６マイクロプロセッサアーキテクチャによって定義されるアドレス変換メカニズムに関連する他のレジスタなどの特殊レジスタのための特有のレジスタ記憶場所とを含み得る。

【００３５】

ある実施例では、ロード／ストアユニット２６は、ロードメモリ操作を投機的に行なうよう構成される。ストアメモリ操作は、プログラム順序で行なわれてもよいが、予測されたウェイに投機的に記憶されてもよい。予測されたウェイが正しくなければ、ストアメモリ操作の前のデータは、その後に、予測されたウェイにリストアされ、ストアメモリ操作は正しいウェイに行なわれる。別の実施例では、ストアは投機的に実行されてもよい。投機的に実行されたストアは、更新の前のキャッシュラインのコピーとともに、ストアバッファ内に置かれる。投機的に実行されたストアが、分岐予測誤りまたは例外のために後に廃棄されるならば、キャッシュラインはバッファに記憶された値にリストアされてもよい。なお、ロード／ストアユニット２６は、投機的実行なしを含む、いかなる投機的実行を行なうようにも構成されてよい。

【００３６】

データキャッシュ２８は、ロード／ストアユニット２６と主メモリサブシステムとの間で転送されるデータを一時的に記憶するために設けられる高速キャッシュメモリである。ある実施例では、データキャッシュ２８は、８ウェイのセットアソシアティブ構造で最大１６キロバイトを記憶する容量を有する。命令キャッシュ１６と同様に、データキャッシュ２８は、ウェイ予測メカニズムを採用し得る。データキャッシュ２８は、セットアソシアティブ構成およびダイレクトマップト構成を含む、さまざまな特有のメモリ構成で実現可能であることが理解される。

【００３７】

ｘ８６マイクロプロセッサアーキテクチャを採用するマイクロプロセッサ１０のある特定の実施例では、命令キャッシュ１６およびデータキャッシュ２８は線形にアドレスされる。線形アドレスは、命令によって特定されるオフセットと、ｘ８６アドレス変換メカニズムのセグメント部分によって特定されるベースアドレスとから形成される。線形アドレスは、任意で、主メモリにアクセスするために物理的アドレスに変換されてもよい。線形から物理的への変換は、ｘ８６アドレス変換メカニズムのページング部分によって特定される。なお、線形アドレスのキャッシュは、線形アドレスタグを記憶する。１組の物理的タグ（図示せず）を採用して、線形アドレスを物理的アドレスにマッピングし、かつ変換エイリアスを検出してもよい。加えて、物理的タグブロックが、線形から物理的へのアドレス変換を行なってもよい。

【００３８】

分岐予測ユニット

ここで図２を参照すると、分岐予測ユニット１４のある実施例の詳細が示される。図が例示するとおり、この実施例では、分岐予測ユニット１４は、分岐選択子アレイ５０と、シーケンシャルアドレス加算器５２と、オフセット計算論理５４と、直接分岐加算器５６と、間接分岐ターゲットキャッシュ５８と、リターンスタック６０と、制御論理回路６２と、マルチプレクサ６４とを含む。

【００３９】

一般的には、分岐予測ユニット１４は、機能ユニット２４Ａ～Ｃからフェッチアドレスを受取り、これに応答して予測される次のフェッチアドレスを出力する。予測される次のフェッチアドレスは次に、命令キャッシュ１６に経路付けられる。予測される次のフェッチアドレスが命令キャッシュ１６をミスすれば、予測される次のフェッチアドレスはプリフェッチ／プリデコードユニット１２に経路付けられる。プリフェッチ／プリデコードユニット１２は、主メモリサブシステムにアクセスし、予測された次のフェッチアドレスに存

在する予め定められた数の命令バイトを命令キャッシュ１６に記憶するプロセスを開始する。

【００４０】

分岐予測ユニット１４の動作中、受取られたフェッチアドレスのすべてまたは一部は、複数の異なった場所に並列に経路付けられる。たとえば、図に例示される実施例では、フェッチアドレスは、分岐選択子アレイ５０、シーケンシャルアドレス加算器５２、命令キャッシュ１６、直接分岐加算器５６、および間接分岐ターゲットキャッシュ５８に運ばれる。

【００４１】

分岐選択子アレイ５０は、各々が命令キャッシュ１６内の特定の命令バイトに対応する、複数個の選択ビットを記憶するよう構成される。ある実施例では、分岐選択アレイ５０は、命令キャッシュ１６の構造をミラーリングするよう構成される。たとえば、命令キャッシュ１６が４ウェイセットアソシアティブキャッシュとして構成される場合、分岐選択子アレイ５０も、４ウェイのセットアソシアティブとして構成されてもよい。分岐選択子アレイ５０と命令キャッシュ１６とはどちらも、フェッチアドレスの同じタグまたは部分を用いてアクセス可能である。分岐選択子アレイ５０によって出力される選択子ビットは、制御論理回路６２に経路付けられる。選択子ビットに基づいて、制御論理回路６２は、４個の潜在的に可能なソースの１つから次の予測されるフェッチアドレスを選択するようマルチプレクサ６４を制御する。ある実施例では、マルチプレクサ６４は、以下の可能性がある４つのアドレス、（１）計算されたシーケンシャルなアドレス、（２）計算された直接分岐ターゲットアドレス、（３）記憶された間接分岐ターゲットアドレス、および（４）記憶されたリターンアドレス、から予測される次のフェッチアドレスを選択する。これらの４個の潜在的に可能なアドレスの各々を個別に以下に記載する。ここに使用されるフェッチブロックという言葉は、命令キャッシュ１６中でヒットするフェッチアドレスを受取るのに応答して命令キャッシュ１６によって出力される命令バイトのブロックのことをいう。いくつかの実現化例では、フェッチブロックはキャッシュラインに等しくてもよい。別の実現化例では、命令キャッシュ１６は、キャッシュラインの一部（たとえば、キャッシュラインの半分）を出力してもよく、その場合は、キャッシュラインは複数のフェッチブロックを含み得る。

【００４２】

４つの潜在的に可能なアドレスの第１、すなわち計算されたシーケンシャルなアドレスは、選択子ビットが、フェッチされたキャッシュライン内のフェッチアドレスの後にテイクンの分岐がないということを示す場合、選択される。たとえば、フェッチされたキャッシュラインが分岐命令を有していなければ、プログラムは、現在のフェッチされたキャッシュラインの終わりを超えて次のシーケンシャルなキャッシュラインへ実行を続けるであろう。次のシーケンシャルなキャッシュラインのフェッチアドレスは、シーケンシャルアドレス加算器５２によって計算される。シーケンシャルアドレス加算器５２は、フェッチアドレス（またはその一部）を受取りそれを定数６６に加算するよう構成される。定数６６は、命令キャッシュ１６のフェッチブロックサイズに等しい。たとえば、ヒットするフェッチアドレスが受取られるたびに、１６個の命令バイトのブロックが命令キャッシュ１６によって出力されるならば、定数６６は１６に等しいであろう。フェッチアドレスにフェッチブロックサイズを加算することにより、次のシーケンシャルなフェッチブロックのアドレスを計算することが可能である。いくつかの実施例では、フェッチアドレスの下位ビット（たとえば、フェッチブロック内のバイトオフセットを示すインデックスビット）は、合計が行なわれるより前または後にゼロにされ得ることに注目されたい。これにより、次のシーケンシャルアドレスは確実に、キャッシュライン（または、キャッシュライン内に複数のフェッチブロックがある場合フェッチブロック）の始まりに対応するようになる。

【００４３】

４つの潜在的に可能なアドレスの第２、すなわち計算された直接分岐ターゲットアドレス

は、制御論理回路６２によって受取られる選択子ビットが、テイクンであると予測される直接分岐命令の存在を示す場合、マルチプレクサ６４によって選択される。直接分岐命令は、オフセット（すなわち、ジャンプするバイトの数）が即値データとして命令内に含まれている分岐命令である。たとえば、ｘ８６命令セットでは、即値オフセットを備える無条件ニアジャンプ命令（たとえば、ＪＭＰ　０１４Ｃ）が直接分岐命令の一例である。即値オフセット（先の例では０１４Ｃ）は、分岐命令のすぐ後に続く命令の命令ポインタ（ＥＩＰ）に加算される、またはそれから減算されるバイトの数を示す。直接分岐加算器５６は、直接分岐ターゲットアドレスを計算する。ある実施例では、予測される直接分岐アドレスを生成するために、直接分岐加算器５６は、以下の３つの構成要素、（１）フェッチアドレス、（２）現在のフェッチアドレスから、直接分岐命令のすぐ後に続く命令へのオフセット、および（３）直接分岐命令内に即値データとして記憶されるオフセット、を合計する。フェッチアドレスは、計算のための開始点としての役割を果たす。現在のフェッチアドレスから直接分岐アドレスのすぐ後に続く命令へのオフセットは、開始点に加算され、有効なＥＩＰ値を形成する。即値データとして記憶されるオフセットは次に、有効ＥＩＰ値に加算され、予測される直接分岐ターゲットアドレスを生成する。加算器５６は、任意の数の異なったタイプの加算器（たとえば、桁上げ伝搬加算器または桁上げ保存加算器）の１つとして、または異なった加算器のタイプの組合せとして実現されてもよいことに注目されたい。さらに、加算演算は、いかなる順序でまたは並列に行なわれてもよい。直接分岐加算器５６によって使用される２つのオフセット値は、命令キャッシュ１６から受取られる命令バイトからのオフセット値を決定する計算論理ユニット５４によって計算される。この計算を以下により詳細に記載する。

【００４４】

４つの潜在的に可能なアドレスの第３、すなわち記憶された間接分岐ターゲットアドレスは、制御論理回路６２によって受取られる選択子ビットが、テイクンであると予測される間接分岐命令の存在を示す場合、マルチプレクサ６４によって選択される。間接分岐命令は、オフセットが即値データとして記憶されない分岐命令である。たとえば、オフセットがメモリ（たとえば、ＪＭＰ［９９１２］）またはレジスタ（たとえば、ＪＭＰ　ＡＸ）に記憶される無条件ジャンプ命令が、間接分岐であると考えられる。直接分岐命令とは対照的に、メモリまたはレジスタから所望のオフセットを取出すのに必要とされる余分な時間のために、間接分岐について、予測される分岐ターゲットアドレスを「オンザフライ」（またはプログラム順序を飛越して）計算することはより困難であるかもしれない。したがって、予測される間接分岐アドレスは、間接分岐ターゲットキャッシュ５８に記憶される。間接分岐ターゲットキャッシュ５８は、現在のフェッチアドレスを受取り、応答して予測される間接分岐ターゲットアドレスをマルチプレクサ６４に出力するよう構成される。制御論理回路６２が、テイクンであると予測される間接分岐命令を検出すると、それは、間接分岐ターゲットキャッシュ５８によって出力される予測される間接分岐ターゲットアドレスを選択するようにマルチプレクサ６４を向ける。

【００４５】

第４の潜在的に可能なアドレスは、記憶されたリターンアドレスである。記憶されたリターンアドレスは、制御論理６２によって受取られる選択子ビットがリターン命令の存在を示す場合、マルチプレクサ６４によって選択される。リターン命令（たとえば、ｘ８６命令セットにおけるＲＥＴおよびＲＥＴＦ）は、先行するコール命令の前にそれが保持する値にＥＩＰをリストアする。コール命令およびリターン命令は、典型的には、プロシージャまたはサブルーチンにジャンプしかつそれからリターンするために使用される。コール命令は、メモリにスタックへの現在の命令ポインタを記憶（またはプッシュ）し、リターン命令は、メモリスタックから記憶された命令ポインタを読出す（またはポップする）ことにより、最終コール命令の前からのその値に命令ポインタをリストアする。図示の実施例では、リターンスタック６０は、コール命令に対応する命令ポインタを記憶するよう構成される。制御論理回路６２によって受取られる選択子ビットがリターン命令を示す場合、制御論理回路６２は、マルチプレクサ６４に、リターンスタック６０の最上部の予測さ

れたリターンアドレスを選択させる。リターンスタック６０は、予測されるリターンアドレスがマルチプレクサ６４によって選択されるたびにスタックポインタのその最上部をインクリメントして適切なスタック動作を可能にするよう構成され得る。同様に、コール命令が検出され、対応する命令ポインタがリターンスタック６０に記憶されると、スタックポインタの最上部がデクリメントされ得る。

【００４６】

ここで図３を参照して、分岐予測ユニット１４のある特定の実施例のさまざまな特徴が示される。図が例示するように、分岐選択子アレイ５０、命令キャッシュ１６、および間接分岐ターゲットキャッシュ５８は各々、記憶された情報に対応するタグを記憶してもよい。間接分岐ターゲットキャッシュは、命令キャッシュ１６の構造を鏡映し、キャッシュラインまたはフェッチブロック当たり１つ以上の間接分岐ターゲットアドレスを記憶してもよい。別の実施例では、間接分岐ターゲットキャッシュ５８は、フルアソシアティブであってもよく、予測される間接分岐ターゲットアドレスの予め定められた数まで記憶してもよい。たとえば、間接分岐ターゲットキャッシュ５８は、予め定められた数の、最も最近のテイクンの間接分岐ターゲットアドレスを記憶するために、先入れ先出し（ＦＩＦＯ）置換方式を用いるよう構成されてもよい。いくつかの実施例では、間接分岐ターゲットキャッシュ５８は、間接分岐ターゲットアドレスのみを記憶してもよく、これは、間接分岐ターゲットキャッシュ５８を、間接分岐ターゲットアドレスに加えてターゲットアドレスの他のタイプを記憶する他の分岐ターゲットキャッシュよりも小さくし得る。さらに、間接分岐は直接分岐ほど頻繁に行なわれないので、これはさらに、記憶場所がより少ない分岐ターゲットキャッシュを実現するのに関連するいかなる潜在的不利益を軽減し得る。

【００４７】

図がまた例示するように、オフセット計算ユニット５４は、命令キャッシュ１６からプリデコードビットを受取るよう構成されてもよい。前述のとおり、プリデコードビットは、プリデコードユニット１２によって生成され、命令キャッシュ１６に記憶される命令の整列およびデコーディングに関連する情報を記憶するために使用されてもよい。オフセット計算ユニット５４はまた、分岐選択子アレイ５０から選択子ビットを受取るよう構成されてもよい。ある実施例では、分岐選択子アレイ５０は、フェッチアドレスの後に生じる選択子ビットのみを出力するよう構成されてもよい。別の実施例では、分岐選択子アレイ５０は、命令キャッシュ１６内の選択されたキャッシュラインまたはフェッチブロックの各バイトに対応する選択子ビットのすべてを出力するよう構成されてもよい。さらに、制御論理回路６２およびオフセット計算ユニット５４はどちらも、分岐選択子アレイ５０によって出力される選択子ビットのすべてまたは一部のみを受取るよう構成されてもよい。ある実施例では、分岐選択子アレイは、フェッチアドレスに対応する選択子ビットを、マルチプレクサ６４の制御入力に直接出力し、それにより制御論理回路６２の複雑さを低減するように（または完全になくすように）構成されてもよい。

【００４８】

分岐予測誤りが検出されると、機能ユニット２４は、正しい分岐ターゲットアドレス（および対応するテイクンまたはノットテイクン情報）を、分岐予測ユニット１４に与え得る。分岐予測ユニット１４は次に、分岐選択子アレイ５０および間接分岐ターゲットキャッシュ５８中の対応する情報を更新するよう構成され得る。分岐予測ユニット１４および分岐選択子アレイ５０はまた、命令キャッシュ１６内の置換されたキャッシュラインまたはフェッチブロックに対応するいかなる選択子ビットをもリセットするよう構成され得る。リターンスタック６０は、そのエントリの各々に関連付けられる有効ビットを有してもよい。さらに、さらなる記憶アレイを用いて、間接分岐ターゲットアレイ５８に間接分岐ターゲットアドレス全体を記憶する代わりに、オフセットの異なったサイズ（たとえば、別個の、８、１６、および３２ビットのアレイ）を記憶することが可能である。しかし、この実施例は、記憶されたオフセットを用いて間接分岐ターゲットアドレス計算を行なうためにさらなる加算器を使用する可能性がある。

【００４９】

ここで図４を参照して、分岐予測ユニット１４のある実施例の動作のさらなる詳細が示される。図が例示するように、ある実施例では、命令キャッシュ１６は、命令バイト８４と、分岐オフセットビット８６と、分岐命令バイト８８と、スタートビット９０とを記憶可能である。なお、アセンブルされていない命令９６が説明のためにのみ示され、命令キャッシュ１６内に実際には記憶されていない。分岐オフセットビット８６、分岐命令バイト８８、およびスタートビット９０はすべて、プリデコードユニット１２によって生成されるプリデコードビットである。各プリデコードビットは、例示に示されるように１命令バイトに対応する。分岐オフセットビット８６は、対応する命令バイトが直接分岐命令の即値オフセットの一部を含む場合、アサートされる。たとえば、命令ＪＭＰ２０は、次の命令（すなわちＰＯＰ　ＡＸ）の開始の２０バイト後のアドレスへの無条件ジャンプを表わす。命令ＪＭＰ２０は、Ｅ９　Ｅ０として１６進数マシンコードにアセンブルされる。ただし、Ｅ９は、即値８ビットオフセットを備える無条件ジャンプについての操作コードであり、Ｅ０は、オフセット（２０）の２の補数表現である。オフセットは、図の円７２によって示される。バイトＥ０は、直接分岐命令についての即値オフセットデータを表すので、対応する分岐オフセットビット８６はセットされる（影付きのブロックによって表わされる）。

【００５０】

同様に、対応する命令バイトが分岐命令の一部であるとき、分岐命令ビット８８はアサートされる。たとえば、命令ＪＭＰ　２０およびＲＥＴはどちらも分岐命令である。したがって、その対応する分岐命令ビット８６はアサートされる。前述のとおり、スタートビット９０は、対応する命令バイトが命令の第１のバイトである場合にのみアサートされる。

【００５１】

図はまた、選択子ビット９４についての符号化のあるタイプの一例を例示する。図において使用される符号化は、命令キャッシュ１６内に記憶される各命令バイトについて２つの選択子ビットを利用する。なお、他の実施例では、選択子ビット９４は、命令キャッシュ１６中の命令バイトのサブセットについて記憶されてもよく、たとえば命令バイト１つおきに２つの選択子ビットであってもよい。表１は、図に示す例に使用される符号化を例示する。

【００５２】

【表１】

表１ー選択子ビット符号化

00	分岐命令ノットテイクン、 シーケンシャルなアドレスを選択する
01	直接分岐命令テイクン 計算された直接分岐アドレスを選択する
10	間接分岐テイクン、間接分岐ターゲットキャッシュから間接分岐アドレスを選択する
11	リターン命令、リターンスタックからリターンアドレスを選択する

【００５３】

図では、矢印８０および８２は例としてのフェッチアドレスを表わす。例のフェッチアドレスを用いて、分岐予測ユニット１４の動作が例示され得る。フェッチアドレスは、命令キャッシュ１６、オフセット計算ユニット５４、直接分岐加算器５６、および分岐選択子アレイ５０によって受取られる。応答して、命令キャッシュ１６は、ＭＯＶ　ＡＸ，ＢＸ命令で始まる命令バイト８４を命令整列ユニット１８に出力する。並列に、対応するプリデコードビット（これも選択的に整列ユニット１８に経路付けされ得る）は、オフセット計算ユニット５４に運ばれる。並列してまた、分岐選択子アレイ５０は、選択子ビット９４（命令キャッシュ１６によって出力される命令バイトに対応する）をオフセット計算ユ

ニット５４に運ぶ。

【００５４】

フェッチアドレス、プリデコードビット、および選択子ビットを受取った後、オフセット計算ユニット５４は２つのオフセットを計算する。第１のオフセットは、フェッチアドレス８０から、（図の範囲７０によって表わされる）第１のテイクンの分岐命令の終わりまでのバイトの数である。これは、複数の方法で、たとえば、アサートされる対応する選択子ビットを有する第１のセットの分岐命令ビットに対してスキャンすることによって決定可能である。このオフセットは、直接分岐加算器５６に運ばれる。第２のオフセットは、（円７２によって表わされる）直接ジャンプ命令内の即値データとして記憶されるオフセットである。ある実施例では、オフセット計算ユニット５４は、プリデコードビットに加えて計算キャッシュ１６から命令バイトを受取ってもよい。オフセット計算ユニット５４は次に、フェッチアドレスの後の第１のテイクンの分岐命令に対応するアサートされた分岐オフセットビット８６に対してスキャンすることによって、命令バイトから即値オフセットデータ７２を選択してもよい。オフセット計算ユニット５４は、第１の「テイクン」の分岐命令に対してスキャンし、「ノットテイクン」である分岐命令が予測に悪影響を与えないようにし得る。さらに、複数の命令バイトに対応する選択子ビットがオフセット計算ユニット５４に運ばれ得るが、ある実施例では、フェッチアドレスに対応する選択子のみが制御論理回路６２（またはマルチプレクサ６４）に運ばれることに注目されたい。上に使用された例では、値０１を有する選択子ビット（フェッチアドレス８０および８２の命令バイトに対応する）は、マルチプレクサ６４に出力され、直接分岐加算器５６からの出力が選択されるべきであることを示すであろう。

【００５５】

代わりに、フェッチアドレスが命令ＰＯＰ　ＢＸのそれであったとすると、（値００を有する）選択子ビットの最終の対は、マルチプレクサ６４に運ばれ、（シーケンシャルアドレス加算器５２によって生成される）次のシーケンシャルなフェッチアドレスが選択されるべきであることを示すであろう。同様に、フェッチアドレスが命令ＲＥＴのそれであった場合、値１１を有する選択子ビットは、マルチプレクサ６４に運ばれ、リターンスタック６０の最上部に記憶されたリターンアドレスが選択されるべきであることを示すであろう。

【００５６】

なお、選択子ビット９４の他の構成もまた企図される。たとえば、ある実施例では、分岐予測ユニットは、リターンスタック６０なしに実現されてもよい。同様に、他のプリデコード情報が、上述のプリデコード情報に加えてまたはその代わりに記憶されてもよく、たとえば、対応する分岐予測情報が有効であるかどうかを示す有効ビットが記憶されてもよい。同様に、リセットが発生する場合、分岐選択子アレイ５０は、すべての選択子ビット９４をクリアする（たとえば００）よう構成されてもよい。命令キャッシュは、キャッシュラインが上書きされているときプリデコードユニット１２に制御信号をアサートするよう構成されてもよい。分岐選択子アレイ５０が命令キャッシュ１６の構造をミラーリングする構造を有するならば、分岐選択子アレイ５０は、上書きされたキャッシュラインに対応するすべての選択子ビット９４をクリアしてもよい。

【００５７】

ここで図５を参照すると、分岐予測ユニット１４の別の実施例が例示される。この実施例では、分岐選択子アレイ５０からの選択子ビットは、マルチプレクサ６４に直接経路付けされ、どのソースが次の予測されるフェッチアドレスを与えるかを選択する。選択子ビットが正しくなければ、または（後に機能ユニット２４Ａ～Ｃによって検出されるように）記憶されたアドレスの１つが正しくなければ、正しくない情報は、機能ユニット２４Ａ～Ｃから受取られた正しい情報で更新される。たとえば、選択子ビット９４が、特定の命令がテイクンであると誤って表すならば、機能ユニット２４Ａ～Ｃは、分岐予測誤りを検出すると、選択子ビットが誤っており更新される必要があるということを分岐予測ユニット１４に通知するであろう。

【００５８】
この実施例のさらなる特徴は、間接分岐ターゲットがオフセットの形で記憶されることである。オフセットは次に、間接分岐加算器６８によってフェッチアドレスに加算される。この計算は、直接分岐加算器５６によって行なわれる計算と同様に行なわれ得る。さらになお、この実施例はリターンスタック６０を使用しない。
【００５９】
次に図６を参照して、マイクロプロセッサにおける予測される次のフェッチアドレスを記憶するための方法のある実施例を例示するフローチャートが示される。この実施例では、現在のフェッチアドレスが受取られた後（ステップ１２０）、以下の複数の機能が並列に行なわれ得る。（１）命令バイトおよびプリデコードビットが命令キャッシュ１６から出力される（ステップ１２２）。（２）選択子ビットが分岐選択子アレイ５０から出力される（ステップ１２４）。（３）フェッチアドレスがフェッチブロック／キャッシュラインサイズに加算され、次のシーケンシャルなフェッチアドレスが生成される（ステップ１２６）。（４）記憶された間接分岐命令アドレスが間接分岐ターゲットキャッシュ５８から出力される。（５）リターンスタック６０の最上部からのリターンアドレスが出力される。次に、オフセット計算ユニット５４は、フェッチアドレスから第１の予測されるテイクンの直接分岐命令へのオフセットを計算し、直接分岐命令内に即値データとして記憶されたオフセットを検出する（ステップ１３２）。一旦計算されると、オフセットは次にフェッチアドレスに加算され、予測される次の直接分岐命令を生成する（ステップ１３４）。最後に、次の予測されるフェッチアドレスは、分岐選択子アレイ５０からの選択子ビットに基づいて選択される（ステップ１３６）。
【００６０】
なお、例示されるステップのいくつかはオプションである（たとえばステップ１３０）。さらになお、ステップの順番は変動可能であり、ステップ内の異なった機能を異なったシーケンスで行なってもよい。たとえば、プリデコードビットは、命令バイトより前に命令キャッシュの出力から利用可能であってもよい（ステップ１２２）。したがって、各オフセットについての計算は、異なったときに開始され得る。さらになお、さらなるステップが異なった実施例においては加えられてもよい。たとえば、前述のように、いくつかの実施例は、アドレス全体の代わりにオフセットとして間接分岐ターゲットアドレスを記憶してもよい。そのような実施例では、記憶されたオフセットをフェッチアドレスおよびオフセット計算論理５４によって生成されたオフセットに加算するさらなるステップが利用されてもよい。
【００６１】
コンピュータシステムへの通用例
ここで図７を参照すると、マイクロプロセッサ１０および分岐予測ユニット１４のある実施例を利用するよう構成されたコンピュータシステム２００のある実施例のブロック図が示される。図示のシステムでは、主メモリ２０４がメモリバス２０６を介してバスブリッジ２０２に結合され、グラフィックスコントローラ２０８がＡＧＰバス２１０を介してバスブリッジ２０２に結合される。最後に、複数個のＰＣＩデバイス２１２Ａ～２１２Ｂが ＰＣＩバス２１４を介してバスブリッジ２０２に結合される。２次バスブリッジ２１６が、ＥＩＳＡ／ＩＳＡバス２２０を介して１つ以上のＥＩＳＡまたはＩＳＡデバイス２１８への電気的インターフェイスを収容するよう設けられてもよい。マイクロプロセッサ１０は、ＣＰＵバス２２４を介してバスブリッジ２０２に結合される。
【００６２】
バスブリッジ２０２は、マイクロプロセッサ１０と主メモリ２０４とグラフィックスコントローラ２０８とＰＣＩバス２１４につながれるデバイスとの間のインターフェイスとして機能する。演算がバスブリッジ２０２に接続されるデバイスの１つから受取られると、バスブリッジ２０２は演算のターゲット（たとえば、特定のデバイスまたは、ＰＣＩバス２１４の場合には、ターゲットはＰＣＩバス２１４にある）を特定する。バスブリッジ２０２は、ターゲットとされたデバイスに演算を経路付ける。バスブリッジ２０２は、一般

的には、ソースデバイスまたはバスによって使用されるプロトコルからターゲットデバイスまたはバスによって使用されるプロトコルに演算を変換する。

【００６３】

ＰＣＩバス２１４に対してＩＳＡ／ＥＩＳＡバスへのインターフェイスを与えることに加えて、２次バスブリッジ２１６はさらに、所望に応じて、さらなる機能を組込んでもよい。たとえば、ある実施例では、２次バスブリッジ２１６は、ＰＣＩバス２１４の使用の優先順位を決定するためのマスタＰＣＩアービタ（図示せず）を含む。外部からの、または２次バスブリッジ２１６と一体化されるかのいずれかの、入力／出力コントローラ（図示せず）をもコンピュータシステム２００内に含めて、所望により、キーボードおよびマスク２２０のための、ならびにさまざまなシリアルポートおよびパラレルポートのための動作サポートを与えてもよい。外部キャッシュユニット（図示せず）が、他の実施例では、マイクロプロセッサ１０とバスブリッジ２０２との間のＣＰＵバス２２４にさらに結合されてもよい。代替的に、外部キャッシュはバスブリッジ２０２に結合されてもよく、外部キャッシュのためのキャッシュ制御論理回路がバスブリッジ２０２に一体化されてもよい。

【００６４】

主メモリ２０４は、アプリケーションプログラムがこれに記憶されマイクロプロセッサ１０が主にここから実行するメモリである。好適な主メモリ２０４は、ＤＲＡＭ（ダイナミック・ランダム・アクセス・メモリ）、および好ましくはＳＤＲＡＭ（シンクロナスＤＲＡＭ）の複数個のバンクを含む。

【００６５】

ＰＣＩデバイス２１２Ａ～２１２Ｂは、たとえば、ネットワークインターフェイスカード、ビデオアクセラレータ、オーディオカード、ハードまたはフロッピィディスクドライブまたはドライブコントローラ、ＳＣＳＩ（スモール・コンピュータ・システムズ・インターフェイス）アダプタおよび電話機能カードなどのさまざまな周辺デバイスを例示するものである。同様に、ＩＳＡデバイス２１８は、モデム、サウンドカード、またはＧＰＩＢもしくはフィールドバスインターフェイスカードなどのさまざまなデータ収集カードなどのさまざまなタイプの周辺デバイスを例示するものである。

【００６６】

ディスプレイ２２６上のテキストおよび画像のレンダリングを制御するためにグラフィックスコントローラ２０８が設けられる。グラフィックスコントローラ２０８は、主メモリ２０４におよびそれから有効にシフト可能である３次元データ構造をレンダリングするために、一般的に公知の典型的なグラフィックスアクセラレータを採用してもよい。グラフィックスコントローラ２０８は、したがって、それがバスブリッジ２０２内のターゲットインターフェイスへのアクセスを要求し受取り、それにより主メモリ２０４へのアクセスを獲得することができるという点で、ＡＧＰバス２１０のマスタであり得る。専用のグラフィックスバスが、主メモリ２０４からの高速のデータ取出を可能にする。ある種の動作では、グラフィックスコントローラ２０８は、ＡＧＰバス２１０でのＰＣＩプロトコルトランザクションを生成するようさらに構成されてもよい。バスブリッジ２０２のＡＧＰインターフェイスは、したがって、ＡＧＰプロトコルトランザクションもＰＣＩプロトコルターゲットおよびイニシエータトランザクションもサポートする機能を含み得る。ディスプレイ２２６は、画像またはテキストを表わすことのできるいかなる電子ディスプレイでもある。好適なディスプレイ２２６は、陰極線管（「ＣＲＴ」）、液晶ディスプレイ（「ＬＣＤ」）などを含む。

【００６７】

なお、ＡＧＰバス、ＰＣＩバスおよびＩＳＡまたはＥＩＳＡバスが上の記載では例として使用されたが、所望によりいかなるバスアーキテクチャが置換されてもよい。さらになお、コンピュータシステム２００は、さらなるマイクロプロセッサを含むマルチプロセッシングコンピュータシステムであってもよい。

【００６８】

さらになお、この説明はさまざまな信号のアサーションに言及することがある。信号が特定の条件を示す値を運ぶ場合、信号が「アサートされる」とここでは用いられる。反対に、信号が特定の条件の欠如を示す値を運ぶ場合、信号は「デアサートされる」または「アサートされない」。信号は、論理０値を運ぶとき、または反対に、論理１値を運ぶときアサートされるものと定義され得る。加えて、上の記載では廃棄されるものとしてさまざまな値が記載された。値は複数の態様で廃棄され得るが、一般的には、値を受取る論理回路によってそれが無視されるように値を変更することを伴う。たとえば、値が１ビットを含む場合、値の論理状態は、値を廃棄するよう反転され得る。値がｎビットの値である場合、ｎビットの符号化の１つが、値が無効であることを示し得る。値を無効の符号化にセットすることにより、値が廃棄されるようになる。加えて、ｎビットの値は、セットされるとｎビットの値が有効であることを示す有効ビットを含み得る。有効ビットをリセットすることは、値を廃棄することを含み得る。値を廃棄する他の方法が同様に使用されてもよい。

【００６９】

産業上の適用性

この発明は、一般的には、マイクロプロセッサおよびコンピュータシステムに適用可能であろう。

【００７０】

さまざまな変形および変更が、上の記載を完全に理解すると当業者には明らかとなるであろう。前掲の特許請求の範囲はそのようなすべての変形および変更を含むものと解釈されるべきであることが意図される。

【図面の簡単な説明】

【図１】　マイクロプロセッサのある実施例の例示の図である。

【図２】　図１のマイクロプロセッサからの分岐予測ユニットのある実施例の例示の図である。

【図３】　図２からの分岐予測ユニットのある実施例のさらなる詳細を例示する図である。

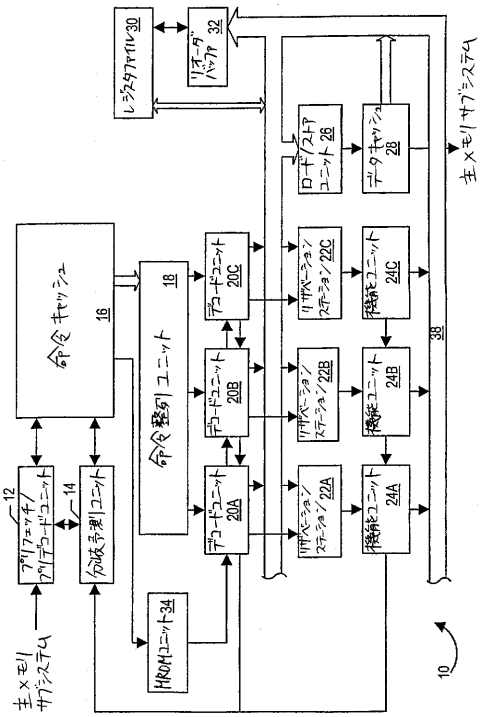
【図４】　図２からの分岐予測ユニットのある実施例のさらなる詳細を例示するブロック図である。

【図５】　図２からの分岐予測ユニットの別の実施例を例示するブロック図である。

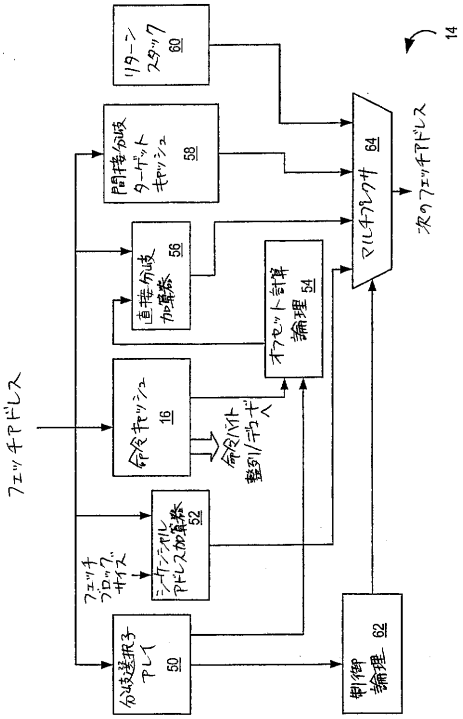
【図６】　マイクロプロセッサにおける予測される次のフェッチアドレスを記憶するための方法のある実施例を例示する流れ図である。

【図７】　図１からのマイクロプロセッサのある実施例を使用するよう構成されたコンピュータシステムのある実施例のブロック図である。

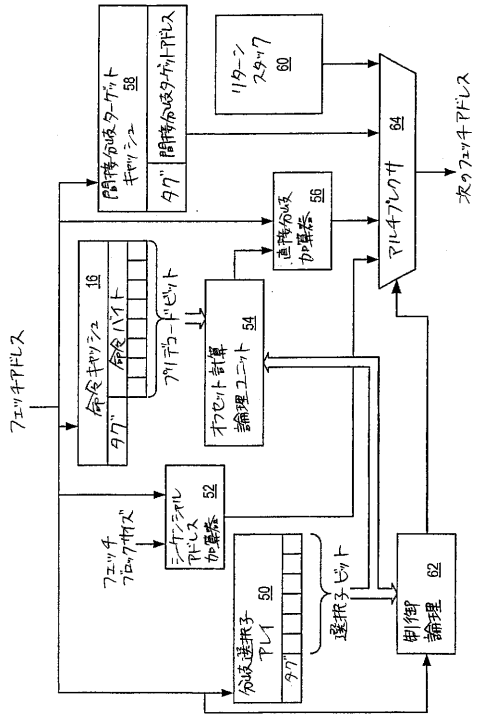
【図１】


主メモリサブシステム
プリフェッチ/プリデコードユニット
分岐予測ユニット
命令キャッシュ
命令整列ユニット
MROMユニット
デコードユニット
リザベーションステーション
機能ユニット
ロード/ストアユニット
データキャッシュ
レジスタファイル
リオーダバッファ
バス インタフェース ユニット
主メモリサブシステム

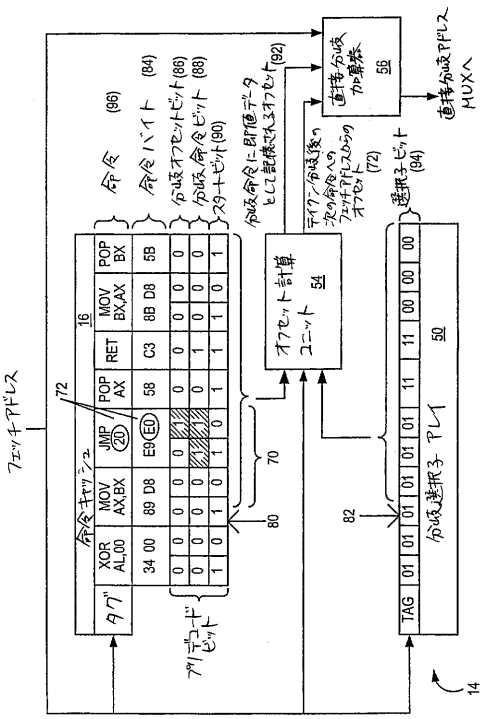
【図２】


フェッチアドレス
命令キャッシュ
命令バイト
フェッチブロックサイズ
ニーモニックデコーダ
オフセット計算論理
直接分岐加算器
間接分岐ターゲットキャッシュ
リターンスタック
マルチプレクサ
次のフェッチアドレス
分岐選択子アレイ
制御論理

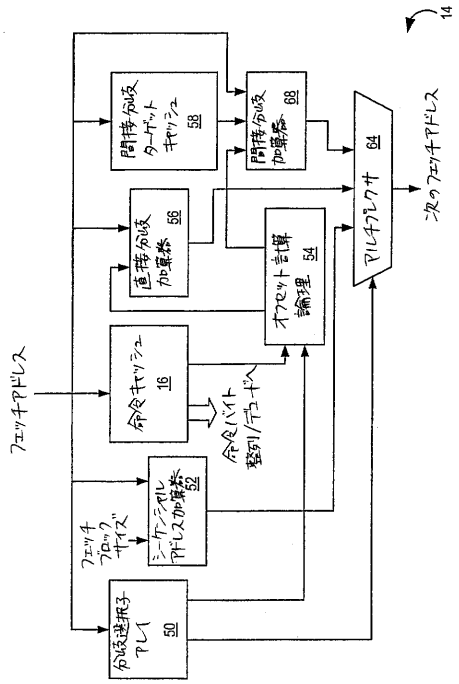
【図３】


フェッチアドレス
命令キャッシュ
タグ
命令バイト
プリデコードビット
フェッチブロックサイズ
ニーモニックデコーダ
オフセット計算論理
直接分岐加算器
間接分岐ターゲットキャッシュ
リターンスタック
マルチプレクサ
次のフェッチアドレス
分岐選択子アレイ
選択子ビット
制御論理

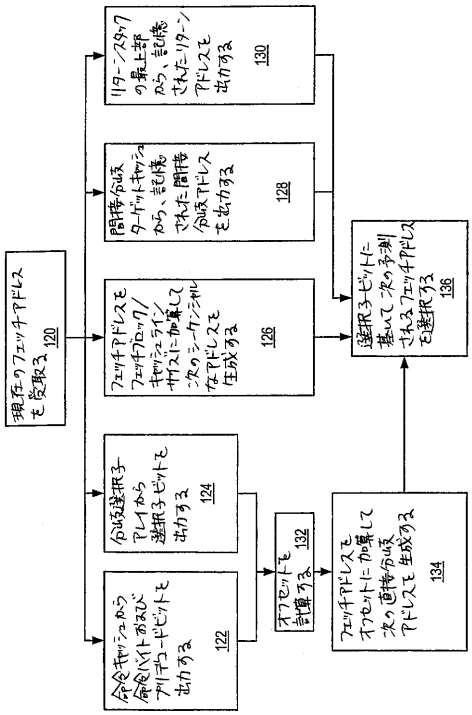
【図４】


フェッチアドレス
命令キャッシュ
タグ
命令バイト
プリデコードビット
オフセット計算ユニット
直接分岐加算器
直接分岐アドレスMUXへ
分岐選択子アレイ
TAG
選択子ビット

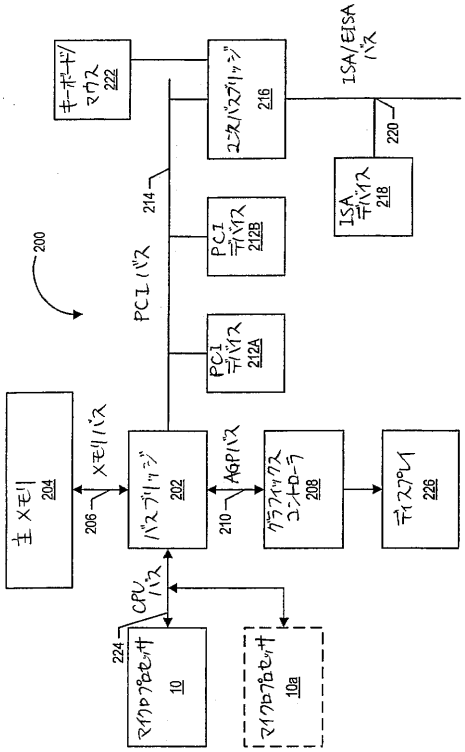
【図５】


フェッチアドレス
命令キャッシュ 16
命令バイト
整列/デコーダ
オフセット計算論理 54
直接分岐ターゲットキャッシュ 58
間接分岐ターゲットキャッシュ 68
直接分岐加算器 66
間接分岐加算器 68
マルチプレクサ 64
次のフェッチアドレス
14
分岐選択子アレイ 50
ミニ分岐テーブル 52
フェッチブロックサイズ

【図６】


現在のフェッチアドレスを受取る 120
命令キャッシュから命令バイトおよびプリデコードビットを出力する 122
分岐選択子アレイから選択子ビットを出力する 124
フェッチアドレスをフェッチブロック/キャッシュライン サイズに加算して次のシーケンシャルなアドレスを生成する 126
オフセットを計算する 132
フェッチアドレスをオフセットに加算して次の直接分岐アドレスを生成する 134
間接分岐ターゲットキャッシュから、記憶されている間接分岐アドレスを出力する 128
リターンスタックの最上部から、記憶されているリターンアドレスを出力する 130
選択子ビットに基いて次のフェッチアドレスを選択する 136

【図７】


200
主メモリ 204
メモリバス 206
バスブリッジ 202
AGPバス 210
グラフィックスコントローラ 208
ディスプレイ 226
CPUバス 224
マイクロプロセッサ 10
マイクロプロセッサ 10a
PCIバス 214
PCIデバイス 212A
PCIデバイス 212B
ISAデバイス 218
ISA/EISAバス 220
ユニバーサルバスブリッジ 216
キーボード/マウス 222

フロントページの続き

(74)代理人 100083703
弁理士 仲村 義平
(74)代理人 100091409
弁理士 伊藤 英彦
(74)代理人 100096781
弁理士 堀井 豊
(74)代理人 100096792
弁理士 森下 八郎
(72)発明者 ロバーツ，ジェイムズ・エス
アメリカ合衆国、７８７４６ テキサス州、オースティン、タマロン・ブールバード、３０５０、ナンバー・１２０５

審査官 後藤 彰

(56)参考文献 特開平８－３２０７８６（ＪＰ，Ａ）
特開平６－１６８１２０（ＪＰ，Ａ）
特開平５－１２００１３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/38