(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2009-282990
(P2009-282990A)
(43)公開日　平成21年12月3日(2009.12.3)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 13/12	(2006.01)	G06F	13/12	340G	5B014
G06F 12/04	(2006.01)	G06F	12/04	540A	5B060

審査請求 有　請求項の数 9　ＯＬ　外国語出願　（全 41 頁）

(21)出願番号　特願2009-149905 (P2009-149905)
(22)出願日　平成21年6月24日 (2009.6.24)
(62)分割の表示　特願2006-541181 (P2006-541181)の分割
原出願日　平成16年10月25日 (2004.10.25)
(31)優先権主張番号　10/721,879
(32)優先日　平成15年11月26日 (2003.11.26)
(33)優先権主張国　米国 (US)

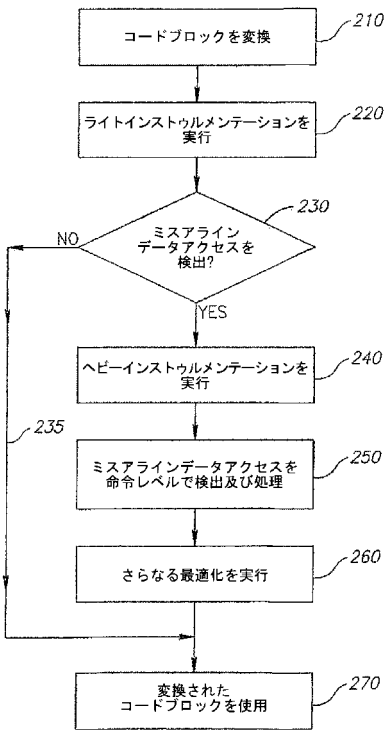
（特許庁注：以下のものは登録商標）
１．Ｌｉｎｕｘ

(71)出願人　591003943
インテル・コーポレーション
アメリカ合衆国　９５０５２　カリフォルニア州・サンタクララ・ミッション　カレッジ　ブーレバード・２２００
(74)代理人　100104156
弁理士　龍華　明裕
(72)発明者　デボアー、ハロルド、シオドー
イスラエル、３０９００　ヤコブ　ジックロン、ハショフティム　ストリート　９エー
(72)発明者　エトジォン、オルナ
イスラエル、３４７６１　ハイファ、カリブ　ストリート　５

最終頁に続く

(54)【発明の名称】ミスアラインデータアクセスの検出及び処理のためのデバイス、システム、及び方法

(57)【要約】　（修正有）
【課題】ミスアラインデータアクセス及びデータミスアラインメントに関連する問題を解決する。
【解決手段】ミスアラインデータアクセスの検出及び処理のためのデバイス、システム、及び方法。方法は、例えば、第１コンピューティングプラットフォームに適した第１フォーマットから第２コンピューティングプラットフォームに適した第２フォーマットに変換されたコードブロックの実行がもたらすミスアラインデータアクセスを検出する段階と、前記ミスアラインデータアクセスに従って前記コードブロックを修正する段階とを備える。
【選択図】図２



【特許請求の範囲】
【請求項１】
第１コンピューティングプラットフォームに適合する第１フォーマットから第２コンピューティングプラットフォームに適合する第２フォーマットに変換された１つのコードブロックの実行がもたらすミスアラインデータアクセスを検出する段階と、
前記ミスアラインデータアクセスに従って、前記コードブロックを修正する段階と
を備える方法。
【請求項２】
検出する段階は、前記コードブロックの実行が前記ミスアラインデータアクセスをもたらすか否かを検出すべく、前記コードブロックのインストゥルメンテーションを実行する段階を有する
請求項１に記載の方法。
【請求項３】
検出する段階は、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行する段階を有する
請求項２に記載の方法。
【請求項４】
検出する段階は、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行する段階を有する
請求項１に記載の方法。
【請求項５】
修正する段階は、実行が前記ミスアラインデータアクセスを処理するコードシーケンスに前記コードブロックの実行を分岐させる１つの命令を、前記コードブロックに追加する段階を有する
請求項１に記載の方法。
【請求項６】
修正する段階は、前記コードブロックの後続の実行におけるミスアラインデータアクセスを処理すべく前記コードブロックを修正する段階を有する
請求項１に記載の方法。
【請求項７】
前記コードブロックを前記第１フォーマットから前記第２フォーマットに変換する段階をさらに備える請求項１に記載の方法。
【請求項８】
検出する段階は、３２ビットベースのコンピューティングプラットフォームに適合するフォーマットから６４ビットベースのコンピューティングプラットフォームに適合するフォーマットに変換されたコードブロックの実行がもたらす、ミスアラインデータアクセスを検出する段階を有する
請求項１に記載の方法。
【請求項９】
第１コンピューティングプラットフォームに適合する第１フォーマットから第２コンピューティングプラットフォームに適合する第２フォーマットに変換された１つのコードブロックの実行がもたらすミスアラインデータアクセスを検出し、前記ミスアラインデータアクセスに従って、前記コードブロックを修正する１つのプロセッサ
を備える装置。
【請求項１０】
前記プロセッサは、前記コードブロックの実行が前記ミスアラインデータアクセスをもたらすか否かを検出すべく、前記コードブロックのインストゥルメンテーションを実行することが可能である

請求項９に記載の装置。
【請求項１１】
前記プロセッサは、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行することが可能である
請求項１０に記載の装置。
【請求項１２】
前記プロセッサは、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行することが可能である
請求項９に記載の装置。
【請求項１３】
前記プロセッサは、実行が前記ミスアラインデータアクセスを処理するコードシーケンスに前記コードブロックの実行を分岐させる１つの命令を、前記コードブロックに追加することが可能である
請求項９に記載の装置。
【請求項１４】
前記プロセッサは、前記コードブロックの後続の実行におけるミスアラインデータアクセスを処理すべく前記コードブロックを修正することが可能である
請求項９に記載の装置。
【請求項１５】
前記プロセッサは、前記ミスアラインデータアクセスを検出する前に、前記コードブロックを前記第１フォーマットから前記第２フォーマットに変換することが可能である
請求項９に記載の装置。
【請求項１６】
前記第１コンピューティングプラットフォームは３２ビットベースのコンピューティングプラットフォームであり、前記第２コンピューティングアーキテクチャは６４ビットベースのコンピューティングプラットフォームである
請求項９に記載の装置。
【請求項１７】
第１コンピューティングプラットフォームに適合する第１フォーマットから第２コンピューティングプラットフォームに適合する第２フォーマットに変換された１つのコードブロックの実行がもたらすミスアラインデータアクセスを検出し、前記ミスアラインデータアクセスに従って、前記コードブロックを修正する１つのプロセッサと、
前記プロセッサに動作可能に連携され、前記コードブロックの少なくとも一部を記憶する１つのダイナミックランダムアクセスメモリと
を備えるコンピューティングプラットフォーム。
【請求項１８】
前記プロセッサは、前記コードブロックの実行が前記ミスアラインデータアクセスをもたらすか否かを検出すべく、前記コードブロックのインストゥルメンテーションを実行することが可能である
請求項１７に記載の装置。
【請求項１９】
前記プロセッサは、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行することが可能である
請求項１８に記載の装置。
【請求項２０】
前記プロセッサは、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテ

ーションを実行することが可能である
請求項１７に記載の装置。
【請求項２１】
記憶された複数の命令のセットを備える機械可読メディアであって、前記複数の命令は、機械によって実行された場合に、前記機械に、
第１コンピューティングプラットフォームに適合する第１フォーマットから第２コンピューティングプラットフォームに適合する第２フォーマットに変換された１つのコードブロックの実行がもたらすミスアラインデータアクセスを検出する段階と、
前記ミスアラインデータアクセスに従って、前記コードブロックを修正する段階と
を備える方法を実行させる機械可読メディア。
【請求項２２】
検出する段階をもたらす前記複数の命令は、前記コードブロックの実行が前記ミスアラインデータアクセスをもたらすか否かを検出すべく、前記コードブロックのインストゥルメンテーションを実行する段階をもたらす
請求項２１に記載の機械可読メディア。
【請求項２３】
検出する段階をもたらす前記複数の命令は、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行する段階をもたらす
請求項２２に記載の機械可読メディア。
【請求項２４】
検出する段階をもたらす前記複数の命令は、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行する段階をもたらす
請求項２１に記載の機械可読メディア。
【請求項２５】
前記複数の命令は、１つのトランスレータの少なくとも一部を構成する
請求項２１に記載の機械可読メディア。
【請求項２６】
前記複数の命令は、１つの実行レイヤの少なくとも一部を構成する
請求項２１に記載の機械可読メディア。
【請求項２７】
前記複数の命令は、１つのオペレーティングシステムの少なくとも一部を構成する
請求項２１に記載の機械可読メディア。
【請求項２８】
前記複数の命令は、１つのコンパイラの少なくとも一部を構成する
請求項２１に記載の機械可読メディア。
【発明の詳細な説明】
【背景技術】
【０００１】
コンピューティングプラットフォームの分野において、ソフトウェアアプリケーションは元々、第１のコンピューティングプラットフォーム、例えば３２ビットベースのコンピューティングプラットフォーム、例えばＩｎｔｅｌ（登録商標）アーキテクチャ３２（ＩＡ－３２）によって実行されるべく記述される。いくつかのケースにおいて、ソフトウェアアプリケーションを変換して実行する適切なハードウェア及び／又はソフトウェアを用いて、第２のコンピューティングプラットフォーム、例えば６４ビットベースのコンピューティングプラットフォーム、例えばＩｎｔｅｌ（登録商標）Ｉｔａｎｉｕｍ（登録商標）プロセッサ上で、そのソフトウェアアプリケーションを実行することが可能である。
【０００２】
第２のコンピューティングプラットフォーム上でのソフトウェアアプリケーションの変

換又は実行の間、ミスアラインデータアクセス及びデータミスアラインメントに関連する問題が生じる場合がある。データミスアラインメントは、例えば、プロセッサによって効率的にアクセスされ得ない１つのメモリアドレスに存在するデータアイテムを含む。第２コンピューティングプラットフォームがミスアラインデータアイテムにアクセスしようとした場合に、望ましくないオーバーヘッド、例えば追加のプロセッシングサイクル又はプロセッシング時間が必要となり得る。いくつかのケースにおいて、第２コンピューティングプラットフォーム上でのソフトウェアアプリケーションの実行は、例えば、ソフトウェアアプリケーションが第１（すなわち、元の）コンピューティングプラットフォームによって実行された場合にも生じ得るデータミスアラインメントにより、既存のプラットフォーム問題を悪化及び増大させる。これはパフォーマンス速度を著しく低下させ、プロセッシング時間及び／又はプロセッシングサイクルの数を著しく増加させ得る。その上、いくつかのケースにおいて、ミスアラインデータアクセスイベントは、例えば第２コンピューティングプラットフォーム上で動作するアプリケーションによって又はオペレーティングシステムによってエラーとして扱われ、その結果、アプリケーションの早期の終了又は他の望ましくない結果をもたらす。

【０００３】

データミスアラインメントアクセス問題は、ミスアラインデータアクセスイベントをもたらすそれぞれの命令を置き換える比較的長いコードシーケンスを用いて、部分的に緩和され得る。しかしながら、全てのデータミスアラインメントアクセス問題を防ぐためのそのような長いコードシーケンスのアプリケーション体は非効率的であり、著しいオーバーヘッド、例えば追加のプロセッシングサイクル及び／又はプロセッシング時間を招く。

【図面の簡単な説明】

【０００４】

本発明に関する主題は、明細書の結びの部分に特に示され、明確に請求される。本発明は、一方で、添付の図面とともに読まれた場合に、以下の詳細な説明を参照することにより、それらの特徴及び利点とともに構成及びオペレーション方法の両方について最も良く理解されるだろう。

【０００５】

【図１】本発明のいくつかの実施形態に係る、データミスアラインメントを検出及び処理することができるコンピューティングプラットフォームの概略図である。

【０００６】

【図２】本発明のいくつかの実施形態に係る、データミスアラインメントを検出及び処理する方法のフローチャートの概略である。

【０００７】

説明の簡単さ及び明確さを目的として、図面に示される構成要素は必ずしも縮尺どおりに描かれる必要がないことが理解されるだろう。例えば、明確さを目的として、いくつかの構成要素の大きさが、他の構成要素に比較して誇張される。さらに、適切である場合には、対応する又は同様の構成要素を示すべく、参照番号が複数の図面にわたって繰り返される。

【発明を実施するための最良の形態】

【０００８】

以下の詳細な説明において、本発明の包括的な理解を提供することを目的として、多くの具体的な細部が説明される。しかしながら、本発明は、これらの具体的な細部なしで実施され得ることが当業者に理解されるだろう。他のケースでは、本発明を不明瞭にしないよう、よく知られた方法、手順、構成要素、ユニット、及び／又は回路は、詳細には説明されない。

【０００９】

図１は、本発明の典型的な実施形態に係る、データミスアラインメントを検出及び処理することができるコンピューティングプラットフォーム１１０を概略的に示す。コンピューティングプラットフォーム１１０は、例えば、データプロセッシング又は種々のソフト

ウェアアプリケーションの実行のために使用され、本発明のいくつかの実施形態に係るミスアラインデータアクセスの検出及び／又は処理を実装する。コンピューティングプラットフォーム１１０は、１つのコンピューティングデバイスを備える。例えば、コンピューティングプラットフォーム１１０は、パーソナルコンピュータ、デスクトップコンピュータ、モバイルコンピュータ、ラップトップコンピュータ、ノートブックコンピュータ、端末、ワークステーション、サーバコンピュータ、携帯情報端末（ＰＤＡ）、タブレットコンピュータ、特化された又は専用のコンピューティングデバイス、ネットワークデバイス、又は類似のものの一部を少なくとも含んでよい。コンピューティングプラットフォーム１１０は、ハードウェアコンポーネント及び／又はソフトウェアコンポーネントの任意の適切な組合せを用いて実装される。

【００１０】

図１に示された例において、コンピューティングプラットフォーム１１０は、例えば、１以上のプロセッサ１４１、１以上のメモリユニット１４２、１以上のストレージユニット１４３、１つの出力ユニット１４４、及び１つの入力ユニット１４５を備える。コンピューティングプラットフォーム１１０は、コンピューティングプラットフォーム１１０の１以上のコンポーネントに動作可能に連携する、技術的に知られた他の適切なコンポーネント又はコンポーネントセットを備えてよい。

【００１１】

プロセッサ１４１は、例えば、１つの中央処理装置（ＣＰＵ）、１つのデジタルシグナルプロセッサ（ＤＳＰ）、１以上のコントローラ、或いは任意の適切な特定用途及び／又は汎用及び／又は多目的のプロセッサ又はマイクロプロセッサ或いはコントローラを含む。メモリユニット１４２は、例えば、ランダムアクセスメモリ（ＲＡＭ）、読み出し専用メモリ、ダイナミックＲＡＭ（ＤＲＡＭ）、シンクロナスＤＲＡＭ（ＳＤ－ＲＡＭ）、又は他の適切なメモリユニットを含む。ストレージユニット１４３は、例えば、ハードディスクドライブ、フレキシブルディスクドライブ、コンパクトディスク（ＣＤ）ドライブ、ＣＤ－ＲＯＭドライブ、或いは他の適切なリムーバブル又は非リムーバブルなストレージユニット又はメモリモジュールを含む。出力ユニット１４４は、例えば、１以上のカード、アダプタ、コネクタ、及び／又はモニタに接続可能又はモニタと通信可能なコンポーネントを含む。入力ユニット１４５は、例えば、１以上のカード、アダプタ、コネクタ、及び／又はキーボード、マウス、又はタッチパッドに接続可能又はキーボード、マウス、又はタッチパッドと通信可能なコンポーネントを含む。メモリユニット１４２、ストレージユニット１４３、出力ユニット１４４、及び／又は入力ユニット１４５は、プロセッサ１４１と動作的に関連する。技術的に知られたように、プロセッサ１４１、メモリユニット１４２、ストレージユニット１４３、出力ユニット１４４、及び／又は入力ユニット１４５は、他の適切なコンポーネント及び／又は実装を有してよい。

【００１２】

いくつかの実施形態において、ソフトウェアは、例えばストレージユニット１４３又はメモリユニット１４２内に記憶されてよく、プロセッサ１４１を用いて実行されてよい。そのようなソフトウェアは、例えば、１以上のオペレーティングシステム、例えば、Ｍｉｃｒｏｓｏｆｔ Ｗｉｎｄｏｗｓ（登録商標）、Ｌｉｎｕｘ、Ｕｎｉｘ（登録商標）、Ａｐｐｌｅ ＯＳ、Ｓｏｌａｒｉｓ、Ｓｕｎ－ＯＳ、ＨＰ－ＵＸ、又は他の適切なオペレーティングシステムを含んでよい。ソフトウェアはさらに、１以上のソフトウェアアプリケーション、１以上のドライバ、コンパイラ、インタープリタ、エミュレータ、実行レイヤ、ソフトウェア環境、管理されたソフトウェア環境、トレーションレイヤ、及び／又は種々の他の適切なソフトウェアコンポーネントを含んでよい。ソフトウェアは、例えば、ソフトウェアコンポーネント、ソフトウェアアプリケーション、及び／又は本発明の実施形態に従う１以上の方法を実装または使用するソフトウェアレイヤ、及び／又は他の適切なソフトウェアコンポーネントを含んでよい。いくつかの実施形態において、ソフトウェア及び／又はメモリユニット１４２は、１以上のミスアラインデータアイテム、例えば、メモリユニット１４２内の、プロセッサ１４１によって効率的にアクセスされ得ないメモリ

アドレスにあるデータアイテムを含んでよい。

【００１３】

いくつかの実施形態において、コンピューティングプラットフォーム１１０は、場合によっては、ソフトウェアコンポーネント及び／又はハードウェアコンポーネントの任意の適切な組合せを用いて実装されるトランスレータ１５０を含んでよい。例えば、一実施形態において、トランスレータ１５０は、メモリユニット１４２及び／又はストレージユニット１４３内に記憶された、プロセッサ１４１を用いて起動又は実行されるソフトウェア及び／又は命令を含んでよい。いくつかの実施形態において、トランスレータ１５０は、ソフトウェアアプリケーション又は１以上の命令を、第１コンピューティングプラットフォームに適合する第１フォーマットから第２コンピューティングプラットフォームに適合する、例えばコンピューティングプラットフォーム１１０に適合する第２フォーマットに変換及び／又は変換してよい。一実施形態において、トランスレータ１５０は、例えば、専用ソフトウェアレイヤ、例えば１つの実行レイヤを用いて実装され得る。いくつかの実施形態において、トランスレータ１５０は、本発明の実施形態に従う、ミスアラインデータアクセスを検出及び処理する１以上のオペレーションを実行する１以上のモジュール又はコンポーネント（図示せず）、例えば、１つの検出モジュール、１つのインストゥルメンテーションモジュール、１つのライトインストゥルメンテーションモジュール、ヘビーインストゥルメンテーションモジュール、１つの変換モジュール、１つの再変換モジュール、１つの最適化モジュール、１つのコード修正モジュール、１つのコード分類モジュール、又は類似のものを含んでよい。

【００１４】

いくつかの実施形態において、トランスレータ１５０は、以下に説明するように、データミスアラインメントを検出及び／又は処理してよい。例えば、トランスレータ１５０によって又はトランスレータ１５０に従って実行される命令又はオペレーションは、本発明のいくつかの実施形態に従う、以下に説明されるようなデータミスアラインメントの検出及び処理の典型的な方法からもたらされる修正を含む修正された変換プロセスを反映する。例えば、本発明のいくつかの実施形態において、トランスレータ１５０は、以下に説明するように、データミスアラインメントに関連するパラメータを考慮する変換方法又は手順を実装してよい。

【００１５】

図２は、本発明のいくつかの実施形態に係る、データミスアラインメントを検出及び処理する一フローチャートを概略的に示す。図２の方法は、本発明の他の実施形態に係る方法とともに、例えば図１のコンピューティングプラットフォーム１１０とともに、トランスレータ１５０、プロセッサ１４１、コンピューティングプラットフォーム１１０によって実行されるソフトウェア又は命令によって、及び／又は他の適切なコンピューティングプラットフォーム、デバイス、装置、及び／又はシステムとともに使用される。いくつかの実施形態において、図２の方法は、適切なソフトウェアコンポーネント及び／又はハードウェアコンポーネントを用いて実装される、１つのコンパイラ、１つのインタープリタ、及び／又は１つのエミュレータによって使用される。

【００１６】

本明細書において使用される"コードのブロック"及び／又は"コードブロック"という語句は、例えば、１以上の命令又は命令セットを含んでよいことが注意される。例えば、１つのコードブロックは、５個の命令、又は２０個の命令等を含んでよい。

【００１７】

本発明のいくつかの実施形態に従って、"ホットブロック"は、例えば、最適化のため又はさらなる最適化オペレーションのための１つの候補として特定及び／又は分類された１つのコードブロックを含んでよい。例えば、ソフトウェアアプリケーションの実行の間に複数回、例えば多数回実行されるコードブロックが、１つのホットブロックとして特定及び／又は分類され得る。他の適切な規則又は条件が、１つのホットブロックを特定、分類、及び／又は定義すべく使用されてよい。

【００１８】
本発明のいくつかの実施形態に従って、"コールドブロック"は、例えば、ホットブロックとして特定及び／又は分類されない１つのコードブロックを含んでよい。例えば、コールドブロックは、ソフトウェアアプリケーションの実行の間に１回だけ又は少数回実行されるコードブロックを含んでよい。いくつかの実施形態において、例えば、コードブロックは、初期の変換オペレーションの間にコールドブロックとして特定及び／又は分類され、さらなる変換及び／又は最適化オペレーションの間にホットブロックとして再び分類されてよいことが注意される。
【００１９】
ブロック２１０で示されるように、この方法は、１つのアプリケーションソフトウェアの１つのコードブロックを、第１コンピューティングプラットフォームに適合した第１フォーマットから、第２コンピューティングプラットフォームに変換又はコンバートしてよい。第１フォーマットにおけるコードブロックは、"元のコードブロック"と呼ばれ、第２フォーマットにおけるコードブロックは"変換されたコードブロック"として呼ばれる。いくつかの実施形態において、例えば、１つのコードブロックは、Ｉｎｔｅｌ（登録商標）アーキテクチャ３２（ＩＡ－３２）からＩｎｔｅｌ（登録商標）アーキテクチャ６４（ＩＡ－６４）に変換され得る。変換は、例えば、トランスレータ１５０を用いて行われてよい。一実施形態において、当該変換は、１つのコンパイラソフトウェア又は１つのインタープリタソフトウェアを用いて行われてよい。変換は、例えば、動的及び／又はリアルタイムに、例えば実質的に実行時に又は実行時の近くで行われてよい。いくつかの実施形態において、変換は、予め、例えば実行時に先立って、行われてよい。
【００２０】
本明細書で説明されるように、本発明の方法の実施形態は、複数のインストゥルメンテーションオペレーションを含んでよい。本発明のいくつかの実施形態に従って、インストゥルメンテーションは、例えば、コードブロック又は命令の振る舞い又はオペレーションをトラッキング、診断、デバッグ、及び／又は解析することを目的として、１以上の命令をあるコードブロック又はある命令に追加することを含む。
【００２１】
本明細書で使用される"ライトインストゥルメンテーション"という語句は、例えば、一コードブロックのレベルで行われるインストゥルメンテーションを含む。一実施形態において、例えば、"ライトインストゥルメンテーション"は、１つのコードブロックの実行がミスアラインデータアクセスをひき起こすか否か、又は１つのコードブロックの実行が、ミスアラインデータアクセスイベントをもたらすか否かを検出するインストゥルメンテーションを含んでよい。いくつかの実施形態において、"ライトインストゥルメンテーション"を実行することは、例えば、与えられたコードブロックの実行がミスアラインデータアクセスイベントを含むか否かを決定することをもたらす。
【００２２】
本明細書で使用される"ヘビーインストゥルメンテーション"という語句は、例えば、一命令のレベルで行われるインストゥルメンテーションを含む。一実施形態において、例えば、"ヘビーインストゥルメンテーション"は、１つの命令の実行がミスアラインデータアクセスをひき起こすか否か、又は１つの命令の実行がミスアラインデータアクセスイベントをもたらすか否かを検出するインストゥルメンテーションを含んでよい。他の実施形態において、"ヘビーインストゥルメンテーション"は、例えば、実行することがミスアラインデータアクセスをもたらす１以上の命令を特定するインストゥルメンテーションを含む。いくつかの実施形態において、"ヘビーインストゥルメンテーション"を実行することは、例えば、実行することがミスアラインデータアクセスイベントをもたらす命令の位置の決定をもたらす。
【００２３】
いくつかの実施形態において、本実施形態で使用される"ミスアラインデータアクセスを検出すること"という語句は、例えば、変換されたコードブロックの実行または続く実

行がミスアラインデータアクセス又はミスアラインデータアイテムへのアクセスを含むことを検出することを含む。いくつかの実施形態において、"ミスアラインデータアクセスを検出すること"という語句は、例えば、実行された場合に、ミスアラインデータアクセス又はミスアラインデータアイテムへのアクセスをもたらす命令の位置を検出することを、付加的に又は代替的に含む。

【００２４】

いくつかの実施形態において、本明細書で使用される"ミスアラインデータアクセスを処理"、"ミスアラインデータアクセスを回避"、及び／又は"ミスアラインデータアクセスを防止"という語句は、例えば、修正された変換されたコードブロックを生成すべく、変換されたコードブロックを修正すること、又は修正された変換されたコードブロックを生成すべく元のコードブロックを再び変換することを含む。いくつかの実施形態において、修正された変換されたコードブロックは、実行された場合に、ミスアラインデータアイテムに、修正されない変換されたコードブロックの実行の間における当該ミスアラインデータへの対応するアクセスに比べて、より効率的に、より高速に、より短い時間で、及び／又はより少量のプロセッシングサイクルを用いて、アクセスし得る。いくつかの実施形態において、"ミスアラインデータアクセスを処理"、"ミスアラインデータアクセスを回避"、及び／又は"ミスアラインデータアクセスを防止"は、例えば、ミスアラインデータアイテムに、複数に分けて、例えば、２、４、８、１６に分けて、又は他の適切な数に分けて、複数のアクセスステージで、複数のアクセス命令を用いて、複数のロード命令を用いて、１つの専用のコードシーケンス、例えば本明細書で詳説されるようなＣｏｄｅ１を用いて、又は本発明の実施形態に従う他の適切な方法を用いて、アクセスすることを含む。

【００２５】

いくつかの実施形態において、"ミスアラインデータアクセスを処理"、"ミスアラインデータアクセスを回避"、及び／又は"ミスアラインデータアクセスを防止"は、例えば、ミスアラインデータアクセスイベントを処理、回避、防止、及び／又は矯正する他の適切な方法を含む。例えば、"ミスアラインデータアクセスを処理"、"ミスアラインデータアクセスを回避"、及び／又は"ミスアラインデータアクセスを防止"は、例えば、データアイテムをアラインすること、データアイテムを第１メモリ位置から第２メモリ位置に移動すること、データアイテムを第１メモリ位置から第２メモリ位置にコピーすること、又は他の適切な方法によって、データアイテムのミスアラインメントを矯正することを含む。

【００２６】

ブロック２２０に示されるように、変換されたコードブロックに、ライトインストゥルメンテーション又は解析を行う。いくつかの実施形態において、実行がミスアラインデータアクセスを必要とする、実質的に全ての命令又は少なくともいくつかの命令が、ライトインストゥルメントされる。ライトインストゥルメンテーションは、例えば、修正された命令が、予め定められた条件の発生、又は予め定められた規則が合致した場合を示すよう、１以上の命令の修正を含む。例えば、いくつかの実施形態において、変換されたコードブロックにおいてミスアラインデータアクセスイベントが検出された場合に、１つのシグナルが生成される。一実施形態において、例えば、当該シグナル又はインジケーションは、トランスレータ１５０内の１つの検出モジュールによって、トランスレータ１５０内の１つの変換モジュールに提供される。

【００２７】

一実施形態において、ブロック２２０のライトインストゥルメンテーションの目的又は結果は、例えば、もしミスアラインデータアクセスイベントを処理、矯正、回避、又は防止することが必要とされる場合に、さらなる解析及び最適化オペレーションが適用できるように、変換されたコードブロックの実行がミスアラインデータアクセスイベントをもたらすか否かを検出することを含む。ブロック２３０で示されるように、方法は、ミスアラインデータアクセスイベントが、変換されたコードブロック内に検出されたか否かをチェックしてよい。そのチェックが否である場合、ブロック２７０に至る矢印２３５によって示されるように、変換されたコードブロックは維持及び／又は使用される。そのチェック

が否である場合、ブロック２４０及び先に示されるように、変換されたコードブロックはさらに調査、解析、及び／又は最適化される。

【００２８】

さらなる調査、解析、及び最適化は、ブロック２４０に示されるように、変換されたコードブロックのヘビーインストゥルメンテーション又は解析を含む。いくつかの実施形態において、その実行がミスアラインデータアクセスを含む実質的に全ての命令が、ヘビーインストゥルメントされる。ヘビーインストゥルメンテーションは、例えば、ミスアラインデータアクセスイベントにおいて、ミスアラインデータアクセスについて情報が提供及び／又は登録されるよう１以上の命令の修正を含んでよい。いくつかの実施形態において、その情報は、例えば、ミスアラインデータアクセスをもたらす変換されたコードブロック内の命令又は複数の命令のインジケーションを含む。その情報は、例えば、ミスアラインデータアクセスのタイプまたはプロパティ、例えばミスアラインメントの粒度のインジケーションを含む。例えば一実施形態において、ミスアラインデータアクセスイベントをもたらす８ビットデータアクセスがヘビーインストゥルメントされて、ミスアラインデータアクセスが１バイト又は４バイトの粒度であることを示す。ミスアラインデータアクセスの種々の他のプロパティ、特性、属性、及び／又は特性が特定、検出、登録、及び／又は解析されてよい。

【００２９】

２５０に示されるように、ブロック２４０のヘビーインストゥルメンテーションは、命令レベルでのデータアクセスミスアラインメントの検出及び処理のために使用される。いくつかの実施形態において、ミスアラインデータアクセスをもたらすと検出された命令のいくつか或いは全て又は実質的に全ては、再生成、再変換、修正、最適化、及び／又は置換されて、データアクセスミスアラインメントを処理、矯正、回避、及び／又は防止し、ミスアラインデータアイテムに比較的より効率的にアクセスすることを可能にし、ミスアラインデータアイテムの比較的高速なアクセスを可能にし、又は、ミスアラインデータアイテムにアクセスする代替の又は適格な方法を用いることを可能にする。

【００３０】

一実施形態において、ブロック２５０で示された検出及び処理オペレーションは、例えば、ホットブロックの変換の間に行われてよい。例えば、ホットブロックが特定され、ブロック２５０で示される検出及び処理オペレーションは、ホットブロック内に含まれる１以上のコールドブロックに行われてよい。

【００３１】

いくつかの実施形態において、ブロック２５０で示される検出及び処理オペレーションは、例えば、以下の仮想コードを用いて行われてよい。

```
// test bit0 to see if address is 2-byte aligned.
// Predicates p.mis and p.al set appropriately.
// Will use p.mis and p.al to predicate the following instructions
tbit p.mis,p.al     = r.addr, 0
// 2-byte load if aligned
(p.al)  ld2 r.val   = [r.addr]
// if misaligned, load each byte separately
(p.mis) ld1 r.val   = [r.addr]
(p.mis) add r.addrH = 1, r.addr
(p.mis) ld1 r.valH  = [r.addrH]
// combine the separately loaded bytes
(p.mis) dep r.val   = r.valH, r.val, 8, 8
```

Code 1

【００３２】

Ｃｏｄｅ１は具体的な目的のみのために提示され、本発明の実施形態はこの件に限定されず、Ｃｏｄｅ１に加えて又はＣｏｄｅ１の代わりに、他の適切な命令、命令セット、オ

ペレーション、仮想コード、又はアルゴリズムが本発明の実施形態に従って使用され得ることが指摘される。

【００３３】

本発明のいくつかの実施形態において、例えば、ブロック２５０で示される検出及び処理オペレーションは、Ｃｏｄｅ１に対する多様な変更を含み得る代替のコードを使用してよい。いくつかの実施形態において、例えば、ミスアラインデータアクセスイベントの初期の検出の１以上の結果は、ミスアラインデータアクセスイベントの、結果として生ずる１又はいくつかの検出を強化及び／又は予防すべく、登録及び／又はトラッキングされてよい。例えば、一実施形態において、例えばホットブロックの解析を通じてミスアラインデータアクセスが検出されたメモリアドレスの位置が、登録及び／又はトラッキングされてよい。登録又はトラッキングは、例えば、適切なリスト、トラッキングリスト、スタック、ルックアップテーブル、配列、データベース、変数、レジスタ、共用体、又は他の適切な任意の方法を用いて行われてよい。以前の検出の登録、トラッキング、及び／又は記憶は、例えば、プロセッサ１４１、メモリユニット１４２、ストレージユニット１４３、トランスレータ１５０、プロセッサ１４１を用いて実行される適切なソフトウェア、及び／又はコンピューティングプラットフォーム１１０の他の適切なコンポーネントを用いて、行われてよい。

【００３４】

ブロック２５０で示される検出及び処理オペレーションにおいて、いくつかの実施形態は、例えば、第１の命令に対するインストゥルメンテーションを解析及び／又は実行してよい。結果として、例えば、第１命令によって第１メモリアドレスをアクセスすることがミスアラインデータアクセスをもたらすことが検出される。そのような場合、第１メモリアドレスの位置は、例えばメモリユニット１４２及び／又はストレージユニット１４３の中に記憶されるレジストリ又はトラッキングリストを例えば用いて、登録及び／又はトラッキングされてよい。第２命令が第２メモリアドレスをアクセスする必要がある場合、データミスアラインメント検出を目的として、本発明の方法は、第２メモリアドレスの位置が第１メモリアドレスの位置と同じであるか否かを、例えばメモリユニット１４２及び／又はストレージユニット１４３の中に記憶されるトラッキングリスト又はレジストリを用いてチェックしてよい。例えば、第１命令の解析が第１メモリアドレスをミスアラインデータアクセスの原因として既に特定している場合、肯定のチェック結果は、第２命令を解析及び／又はインストゥルメントする必要を取り除き得る。いくつかの実施形態において、データミスアラインメント検出について、異なるメモリアドレスも全く同様であることが指摘される。例えば第１及び第２メモリアドレスの間の差又は間隔がＮバイトである場合、データミスアラインメント検出を目的としては、第１及び第２メモリアドレスは、Ｎバイトのミスアラインメント及び／又はＮの因数であるＫバイトのミスアラインメントについて全く同様であり得る。例えば、一実施形態において、メモリアドレスＬ及びメモリアドレスＬ＋８は、８バイトのデータミスアラインメントを検出する目的として、或いは４バイト又は２バイトのデータミスアラインメントを検出する目的として、全く同様である。

【００３５】

同様に、いくつかの実施形態において、ブロック２５０で示される検出及び処理オペレーションを通じて、与えられたミスアラインデータアクセスの検出に使用されるパラメータ、属性、プロパティ、変数、特性、結果、チェック結果、及び／又は計算結果のいくつか又は全ては、後続の検出において使用又は再使用されてよい。例えば、Ｃｏｄｅ１の２バイトロードの例は、ミスアラインデータアクセスの１以上の以前の検出又は解析においてセット、決定、及び／又は計算された１以上の属性を使用し得る。

【００３６】

いくつかの実施形態において、ブロック２５０で示される検出及び処理オペレーションを行うことは、例えばミスアラインデータアクセスが検出された場合、例えばデータアクセスを実行するための１つのコードシーケンス又は命令のセットを使用することを含む。

当該コードシーケンス又は命令のセットは比較的長いか比較的多数の命令又はオペレーションを含んでよい。いくつかの実施形態において、そのようなコードシーケンス又は命令のセットは、ソフトウェアアプリケーションの中の適切な位置に移されてよい。例えば、一実施形態において、スケジューラプロセスは、これらの命令のいくつか又は全てを、変換されたコードブロックの外部又はリモートの位置に、例えばサブルーチン、プロシージャ、関数、又は同種のものとして、移してよい。実行中、変換されたコードブロックの中の命令は、そのポイントで実行が外部又はリモートの命令に進み得る"分岐"命令に到達するまで実行されてよい。外部又はリモートの命令は、その実行が、変換されたコードブロックの実行が外部又はリモートの命令に"分岐された"ポイントから再開される実行をひき起こす最後の命令を含む。

【００３７】

ブロック２６０で示されるように、必要に応じて、いくつかの実施形態は、例えば、上記のオペレーションを終了した後又はホットブロックが変換された後に発生し得るミスアラインデータアクセスを検出及び処理すべく、変換されたコードブロックをさらに最適化又は解析してよい。例えば、一実施形態において、ミスアラインデータアクセスは、相対的に最適化された変換されたコードが生成された後に表れる場合がある。いくつかの実施形態において、その結果は、例えばブロック２５０で示される検出及び処理オペレーションとともに又は他の適切な時に使用されるさらなる最適化オペレーションを使用して、排除、処理、回避、又は緩和される。例えば、ブロック２５０で示されるオペレーションの間、１つの命令はミスアラインデータアクセスイベントを生成又はひき起こさないが、当該命令が、実行中にミスアラインデータアクセスをもたらす有力候補であることが検出される場合がある。そのような命令は、"候補命令"と呼ばれる。候補命令の特定、検出、分類、登録、及び／又は評価が、例えば１以上の適切な規則又は条件、例えば命令のタイプ、ミスアラインデータアクセスを命令がもたらす見込みの可能性、又はミスアラインデータアクセスを命令がもたらす著しい可能性の検出に関して実行されてよい。

【００３８】

本発明のいくつかの実施形態に従って、ブロック２６０で示されるオペレーションは、例えば候補命令を検出すると、候補命令に対するインストゥルメンテーションを実行することを、必要に応じて含んでよい。ミスアラインデータアクセスが、使用されたインストゥルメンテーションを用いて検出された場合、変換されたブロックは破棄されてよい。さらに、方法は、元のコードブロックが変換された場合に、元のコードブロックの中の実質的に全ての候補命令は、例えばＣｏｄｅ１又は任意の他の適切なコードシーケンスを用いて、ミスアラインデータアクセスを検出して処理するように変換されるインジケーションを登録してよい。登録されたインジケーションは、例えば、トランスレータ１５０、メモリユニット１４２、及び／又はストレージユニット１４３の中に記憶されてよい。いくつかの実施形態において、インジケーションは、候補命令だけでなく、候補命令のタイプの実質的に全ての命令にも関連してよい。元のコードブロックの変換において、登録されたインジケーションが検出又はトラッキングされ、そしてそれに応じて、元のコードブロックの中の実質的に全ての候補命令、又は元のコードブロックの中の候補命令のタイプの全ての命令が、例えばＣｏｄｅ１又は任意の他の適切なコードシーケンスを用いて、ミスアラインデータアクセスを検出して処理するように変換されてよい。

【００３９】

いくつかの実施形態において、上記の１以上のオペレーション、例えばブロック２４０のデータミスアラインメント検出及び処理オペレーション、ブロック２４０に関連して説明されたコード修正、又はブロック２６０の最適化オペレーションは、１以上の適切なソフトウェア及び／又はハードウェアコンポーネントによって実行されてよい。例えば、いくつかの実施形態において、コード修正は、トランスレータ１５０、或いはトランスレータ１５０の外部のモジュール又はコンポーネントによって実行されてよい。一実施形態において、コードの修正は、例えば、その修正がコード修正をひき起こし得るトランスレータ１５０の修正、或いは他の適切なコンポーネント又はモジュールの修正によって行われ

てよい。同様に、データミスアラインメントの検出及び／又は処理は、トランスレータ１５０によって、或いはトランスレータ１５０の内部又は外部にある１以上のモジュールによって実行されてよい。

【００４０】

本発明のいくつかの実施形態において、上記の１以上のオペレーション、例えば、ブロック２１０の変換オペレーション、ブロック２２０のライトインストゥルメンテーションオペレーション、ブロック２３０のデータミスアラインメントチェック、ブロック２４０のヘビーインストゥルメンテーションオペレーション、ブロック２５０のデータミスアラインメント検出及び処理オペレーション、ブロック２６０の最適化オペレーション、又は他の適切なオペレーションが繰り返される、或いは１度以上、又は多様なオペレーションの順番で実行されてよい。さらに、いくつかの実施形態において、１以上のこれらのオペレーションの結果として、変換されたコードブロックは破棄、削除、置換、調査、解析、修正、最適化、再生成、再生産、又は再変換されてよい。ブロック２７０で示されるように、上記の１以上又は全てのオペレーション、或いは上記の１以上のオペレーションの何度かのイタレーション又は繰り返しの後、結果として得られる変換されたコードブロックは保持及び使用されてよく、或いは保持されて使用され得る変換されたコードブロックとして登録されてよい。

【００４１】

本発明の実施形態は、多様な利益を与える。例えば、いくつかの実施形態において、データミスアラインメント検出及び処理は、プロセッシング時間及び／又はプロセッシング時間を著しく減少させ、パフォーマンスを著しく向上させ得る。一実施形態において、例えば、完了するのにデータミスアラインメント検出及び処理なしで約１２３６秒を要する作業負荷が、本発明の実施形態に従うデータミスアラインメント検出及び処理を用いて完了するのに約１３３秒しか要しない。本発明のいくつかの実施形態は、説明された利益に加えて又は代わりに、多様な他の利益を与える。

【００４２】

本発明のいくつかの実施形態は、例えば、装置、例えばコンピューティングプラットフォーム１１０、プロセッサ１４１、又は他の適切な機械によって実行された場合に、機械に、本発明の実施形態に従う方法及び／又はオペレーションを実行させる命令又は命令のセットを記憶する機械可読メディア又は製品を用いて実装されてよい。そのような機械は、例えば、任意の適切なプロセッシングプラットフォーム、コンピューティングプラットフォーム、コンピューティングデバイス、プロセッシングデバイス、コンピューティングシステム、プロセッシングシステム、コンピュータ、プロセッサ、又は同種のものを含んでよく、ハードウェア及び／又はソフトウェアの任意の適切な組合せを用いて実装されてよい。機械可読メディア又は製品は、例えば、任意のタイプのメモリユニット（例えば、メモリユニット１４２）、メモリデバイス、メモリ製品、メモリメディア、ストレージデバイス、ストレージ製品、ストレージメディア及び／又はストレージユニット（例えば、ストレージユニット１４３）、例えば、メモリ、リムーバブル又はノンリムーバブルメディア、消去可能又は消去不可能なメディア、書き込み可能又は再書き込み不可能なメディア、デジタル又はアナログメディア、ハードディスク、フレキシブルディスク、コンパクトディスクリードオンリーメモリ（ＣＤ－ＲＯＭ）、コンパクトディスクレコーダブル（ＣＤ－Ｒ）、コンパクトディスクリライタブル（ＣＤ－ＲＷ）、光ディスク、磁気メディア、種々のタイプのデジタル多目的ディスク（ＤＶＤ）、テープ、カセット、或いは同種のものを含んでよい。命令は、任意の適切なタイプのコード、例えば、ソースコード、コンパイル済みコード、インタープリタコード、実行コード、静的コード、動的コード、又は同種のものを含んでよく、任意の適切な高レベル、低レベル、オブジェクト指向、視覚的、コンパイル型、及び／又はインタープリタ型プログラミング言語、例えば、Ｃ、Ｃ＋＋、Ｊａｖａ（登録商標）、ＢＡＳＩＣ、Ｐａｓｃａｌ、Ｆｏｒｔｒａｎ、Ｃｏｂｏｌ、アッセンブリ言語、マシンコード、又は同種のものを用いて実装されてよい。

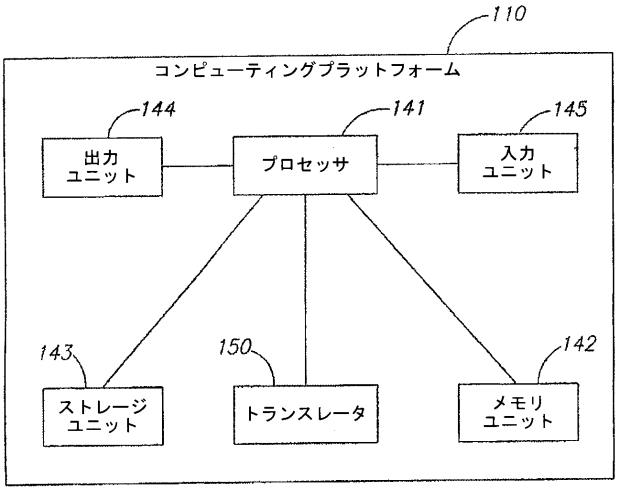
【００４３】

本発明のいくつかの実施形態は、特定の用途又は特定の設計要件に適するように、ソフトウェア、ハードウェア、又はソフトウェア及び／又はハードウェアの任意の組合せによって実装されてよい。本発明の実施形態は、全体的に又は部分的に、互いに切り離された、又は互いに一体化された複数のユニット及び／又は複数のサブユニットを含んでよく、技術的に知られたように、特定の、多目的の、又は汎用のプロセッサ又はデバイスを用いて実装されてよい。本発明のいくつかの実施形態は、データの一時的又は長期記憶のため、又は具体的な実施形態のオペレーションを促進すべく、複数のバッファ、複数のレジスタ、複数のストレージユニット、及び／又は複数のメモリユニットを含んでよい。

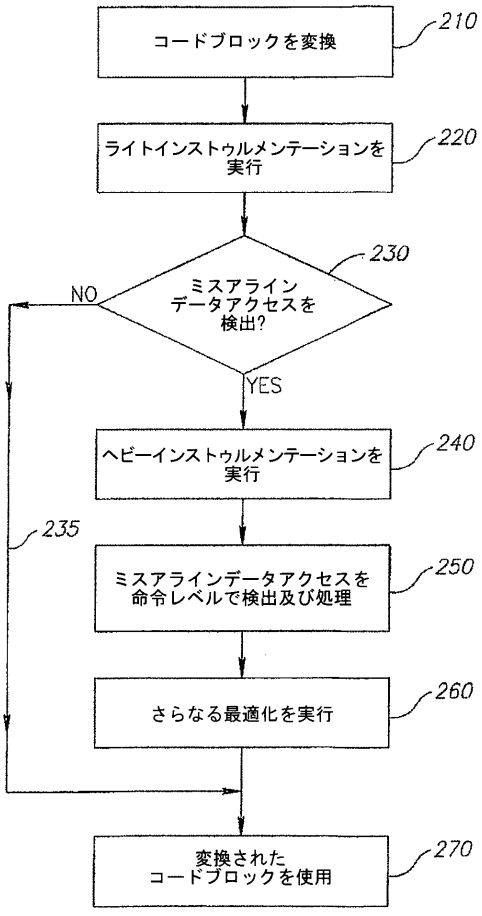
【００４４】

本発明のいくつかの機能が本明細書に示され及び説明されたが、多くの変更、置き換え、改変、及び／又は均等物が、当業者に考えられる。したがって、添付の請求項は、全てのそのような変更及び／又は改変を含むことが意図されていることが理解されるべきである。

【図１】



【図２】



【手続補正書】
【提出日】平成21年7月23日(2009.7.23)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
ミスアラインデータアクセスを検出して処理するコンピューティングプラットフォームであって、
データアクセス命令に対するインストゥルメンテーション実行して、ミスアラインアクセスをもたらす命令に関する情報を生成する手段と、
前記情報を用いて、コードブロックを、第１コンピューティングプラットフォームに適合するＩＡ－３２フォーマットから、第２コンピューティングプラットフォームに適合する６４ビットフォーマットに変換する手段と、
変換された前記６４ビットフォーマットのコードブロックを、前記第２コンピューティングプラットフォームで実行する手段と、
ミスアラインデータアクセスを処理するために、変換された前記６４ビットフォーマットのコードブロックを処理関数に分岐させる手段と
を備えるコンピューティングプラットフォーム。
【請求項２】
ミスアラインデータアクセスを検出して処理する方法であって、
データアクセス命令に対するインストゥルメンテーション実行して、ミスアラインアクセスをもたらす命令に関する情報を生成する段階と、
前記情報を用いて、コードブロックを、第１コンピューティングプラットフォームに適合するＩＡ－３２フォーマットから、第２コンピューティングプラットフォームに適合する６４ビットフォーマットに変換する段階と、
変換された前記６４ビットフォーマットのコードブロックを、前記第２コンピューティングプラットフォームで実行する段階と、
ミスアラインデータアクセスを処理するために、変換された前記６４ビットフォーマットのコードブロックを処理関数に分岐させる段階と
を備える方法。
【手続補正２】
【補正対象書類名】明細書
【補正対象項目名】００４４
【補正方法】変更
【補正の内容】
【００４４】
本発明のいくつかの機能が本明細書に示され及び説明されたが、多くの変更、置き換え、改変、及び／又は均等物が、当業者に考えられる。したがって、添付の請求項は、全てのそのような変更及び／又は改変を含むことが意図されていることが理解されるべきである。
（項目１）
第１コンピューティングプラットフォームに適合する第１フォーマットから第２コンピューティングプラットフォームに適合する第２フォーマットに変換された１つのコードブロックの実行がもたらすミスアラインデータアクセスを検出する段階と、
前記ミスアラインデータアクセスに従って、前記コードブロックを修正する段階と
を備える方法。
（項目２）

検出する段階は、前記コードブロックの実行が前記ミスアラインデータアクセスをもたらすか否かを検出すべく、前記コードブロックのインストゥルメンテーションを実行する段階を有する
項目１に記載の方法。
（項目３）
検出する段階は、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行する段階を有する
項目２に記載の方法。
（項目４）
検出する段階は、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行する段階を有する
項目１に記載の方法。
（項目５）
修正する段階は、実行が前記ミスアラインデータアクセスを処理するコードシーケンスに前記コードブロックの実行を分岐させる１つの命令を、前記コードブロックに追加する段階を有する
項目１に記載の方法。
（項目６）
修正する段階は、前記コードブロックの後続の実行におけるミスアラインデータアクセスを処理すべく前記コードブロックを修正する段階を有する
項目１に記載の方法。
（項目７）
前記コードブロックを前記第１フォーマットから前記第２フォーマットに変換する段階をさらに備える項目１に記載の方法。
（項目８）
検出する段階は、３２ビットベースのコンピューティングプラットフォームに適合するフォーマットから６４ビットベースのコンピューティングプラットフォームに適合するフォーマットに変換されたコードブロックの実行がもたらす、ミスアラインデータアクセスを検出する段階を有する
項目１に記載の方法。
（項目９）
第１コンピューティングプラットフォームに適合する第１フォーマットから第２コンピューティングプラットフォームに適合する第２フォーマットに変換された１つのコードブロックの実行がもたらすミスアラインデータアクセスを検出し、前記ミスアラインデータアクセスに従って、前記コードブロックを修正する１つのプロセッサ
を備える装置。
（項目１０）
前記プロセッサは、前記コードブロックの実行が前記ミスアラインデータアクセスをもたらすか否かを検出すべく、前記コードブロックのインストゥルメンテーションを実行することが可能である
項目９に記載の装置。
（項目１１）
前記プロセッサは、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行することが可能である
項目１０に記載の装置。
（項目１２）
前記プロセッサは、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位

置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行することが可能である
項目９に記載の装置。
（項目１３）
前記プロセッサは、実行が前記ミスアラインデータアクセスを処理するコードシーケンスに前記コードブロックの実行を分岐させる１つの命令を、前記コードブロックに追加することが可能である
項目９に記載の装置。
（項目１４）
前記プロセッサは、前記コードブロックの後続の実行におけるミスアラインデータアクセスを処理すべく前記コードブロックを修正することが可能である
項目９に記載の装置。
（項目１５）
前記プロセッサは、前記ミスアラインデータアクセスを検出する前に、前記コードブロックを前記第１フォーマットから前記第２フォーマットに変換することが可能である
項目９に記載の装置。
（項目１６）
前記第１コンピューティングプラットフォームは３２ビットベースのコンピューティングプラットフォームであり、前記第２コンピューティングアーキテクチャは６４ビットベースのコンピューティングプラットフォームである
項目９に記載の装置。
（項目１７）
第１コンピューティングプラットフォームに適合する第１フォーマットから第２コンピューティングプラットフォームに適合する第２フォーマットに変換された１つのコードブロックの実行がもたらすミスアラインデータアクセスを検出し、前記ミスアラインデータアクセスに従って、前記コードブロックを修正する１つのプロセッサと、
前記プロセッサに動作可能に連携され、前記コードブロックの少なくとも一部を記憶する１つのダイナミックランダムアクセスメモリと
を備えるコンピューティングプラットフォーム。
（項目１８）
前記プロセッサは、前記コードブロックの実行が前記ミスアラインデータアクセスをもたらすか否かを検出すべく、前記コードブロックのインストゥルメンテーションを実行することが可能である
項目１７に記載の装置。
（項目１９）
前記プロセッサは、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行することが可能である
項目１８に記載の装置。
（項目２０）
前記プロセッサは、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行することが可能である
項目１７に記載の装置。
（項目２１）
記憶された複数の命令のセットを備える機械可読メディアであって、前記複数の命令は、機械によって実行された場合に、前記機械に、
第１コンピューティングプラットフォームに適合する第１フォーマットから第２コンピューティングプラットフォームに適合する第２フォーマットに変換された１つのコードブロックの実行がもたらすミスアラインデータアクセスを検出する段階と、

前記ミスアラインデータアクセスに従って、前記コードブロックを修正する段階と
を備える方法を実行させる機械可読メディア。
（項目２２）
検出する段階をもたらす前記複数の命令は、前記コードブロックの実行が前記ミスアラインデータアクセスをもたらすか否かを検出すべく、前記コードブロックのインストゥルメンテーションを実行する段階をもたらす
項目２１に記載の機械可読メディア。
（項目２３）
検出する段階をもたらす前記複数の命令は、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行する段階をもたらす
項目２２に記載の機械可読メディア。
（項目２４）
検出する段階をもたらす前記複数の命令は、実行が前記ミスアラインデータアクセスをもたらす１つの命令の位置を検出すべく、前記コードブロックの中の少なくとも１つの命令のインストゥルメンテーションを実行する段階をもたらす
項目２１に記載の機械可読メディア。
（項目２５）
前記複数の命令は、１つのトランスレータの少なくとも一部を構成する
項目２１に記載の機械可読メディア。
（項目２６）
前記複数の命令は、１つの実行レイヤの少なくとも一部を構成する
項目２１に記載の機械可読メディア。
（項目２７）
前記複数の命令は、１つのオペレーティングシステムの少なくとも一部を構成する
項目２１に記載の機械可読メディア。
（項目２８）
前記複数の命令は、１つのコンパイラの少なくとも一部を構成する
項目２１に記載の機械可読メディア。
【手続補正書】
【提出日】平成21年8月24日(2009.8.24)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
ミスアラインデータアクセスを検出して処理するコンピューティングプラットフォームであって、
データアクセス命令に対するインストゥルメンテーション実行して、ミスアラインアクセスをもたらす命令に関する情報を生成する手段と、
前記情報を用いて、コードブロックを、第１コンピューティングプラットフォームに適合するＩＡ－３２フォーマットから、第２コンピューティングプラットフォームに適合する６４ビットフォーマットに変換する手段と、
変換された前記６４ビットフォーマットのコードブロックを、前記第２コンピューティングプラットフォームで実行する手段と、
ミスアラインデータアクセスを処理するために、変換された前記６４ビットフォーマットのコードブロックを処理関数に分岐させる手段と
を備えるコンピューティングプラットフォーム。

【請求項２】
ソフトウェアアプリケーションの実行の間に複数回実行される前記コードブロックを決定する手段
をさらに備える請求項１に記載のコンピューティングプラットフォーム。
【請求項３】
ホットコードブロックである前記コードブロックを決定する手段
をさらに備える請求項１に記載のコンピューティングプラットフォーム。
【請求項４】
前記変換する手段は、インタプリタソフトウェアを有する請求項１から３のいずれかに記載のコンピューティングプラットフォーム。
【請求項５】
前記変換する手段は、トランスレータソフトウェアを有する請求項１から３のいずれかに記載のコンピューティングプラットフォーム。
【請求項６】
前記変換する手段は、コンパイラソフトウェアを有する請求項１から３のいずれかに記載のコンピューティングプラットフォーム。
【請求項７】
ミスアラインデータアクセスを検出して処理する方法であって、
データアクセス命令に対するインストゥルメンテーション実行して、ミスアラインアクセスをもたらす命令に関する情報を生成する段階と、
前記情報を用いて、コードブロックを、第１コンピューティングプラットフォームに適合するＩＡ－３２フォーマットから、第２コンピューティングプラットフォームに適合する６４ビットフォーマットに変換する段階と、
変換された前記６４ビットフォーマットのコードブロックを、前記第２コンピューティングプラットフォームで実行する段階と、
ミスアラインデータアクセスを処理するために、変換された前記６４ビットフォーマットのコードブロックを処理関数に分岐させる段階と
を備える方法。
【請求項８】
前記処理関数に分岐させる段階の後に、対応するミスアラインデータアクセス命令シーケンスを実行し、前記ミスアラインデータアクセスを処理する段階
をさらに備える請求項７に記載の方法。
【請求項９】
前記処理関数から分岐して、変換された前記６４ビットフォーマットのコードブロックの実行を再開する段階
をさらに備える請求項７または８に記載の方法。

フロントページの続き

(72)発明者　チェン、ジャン－ピン
中華人民共和国、２００３３６、シャンハイ、バイファ　ストリート　レーン　３４５、ディンジャン　ユアン　ビルディング　２９、ルーム　２０２

Ｆターム(参考) 5B014 GC15 GD41
5B060 AC01 DA01 DA09

【外国語明細書】

DEVICE, SYSTEM AND METHOD FOR DETECTION AND HANDLING OF MISALIGNED DATA ACCESS

BACKGROUND OF THE INVENTION

[0001] In the field of computing platforms, a software application may be originally written to be executed by a first computing platform , for example, a 32-bit based computing platform, e.g., Intel (RTM) Architecture 32 (IA-32). In some cases, it may be possible to execute the software application on a second computing platform, for example, a 64-bit based computing platform, e.g., Intel (RTM) Itanium (RTM) processor, using suitable hardware and/or software to translate and execute the software application.

[0002] During translation or execution of the software application on the second computing platform, misaligned data access and problems associated with data misalignment may occur. Data misalignment may include, for example, a data item residing at a memory address that may not be efficiently accessed by a processor. Undesired overhead, e.g., additional processing cycles or processing time, may be required when the second computing platform attempts to accesses a misaligned data item. In some cases, execution of a software application on the second computing platform may compound and intensify existing performance problems, e.g., due to data misalignment, that are also experienced when the software application is executed by the first (i.e., the original) computing platform. This may significantly decrease performance speed and may significantly increase processing time and/or the number of processing cycles. Furthermore, in some cases, a misaligned data access event may be treated as an error by an application or by an operating system, e.g., running on the second computing platform, as an error, and may consequently cause early termination of an application or other undesired results.

[0003] Data misalignment access problems may be partially mitigated using a relatively long code sequence to replace each instruction that may result in a misaligned data access event. However, bulk application of such long code sequences to prevent all data misalignment access

problems may be inefficient and may incur significant overhead, e.g., additional processing cycles and/or processing time.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The subject matter regarded as the invention is particularly pointed out and distinctly claimed in the concluding portion of the specification. The invention, however, both as to organization and method of operation, together with features and advantages thereof, may best be understood by reference to the following detailed description when read with the accompanied drawings in which:

[0005] FIG. 1 is a schematic illustration of a computing platform capable of detecting and handling data misalignment, in accordance with some embodiments of the invention; and

[0006] FIG. 2 is a schematic flow-chart of a method of detecting and handling data misalignment in accordance with some embodiments of the invention.

[0007] It will be appreciated that for simplicity and clarity of illustration, elements shown in the figures have not necessarily been drawn to scale. For example, the dimensions of some of the elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference numerals may be repeated among the figures to indicate corresponding or analogous elements.

DETAILED DESCRIPTION OF THE INVENTION

[0008] In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the invention. However, it will be understood by those of ordinary skill in the art that the invention may be practiced without these specific details. In other instances, well-known methods, procedures, components, units and/or circuits have not been described in detail so as not to obscure the invention.

[0009] FIG. 1 schematically illustrates a computing platform 110 capable of detecting and handling data misalignment, in accordance with exemplary embodiments of the invention. Computing platform 110 may be used, for example, for data processing or for execution of various software applications, and may implement detection and/or handling of misaligned data access in accordance with some embodiments of the invention. Computing platform 110 may include a computing device. For example, computing platform 110 may include at least part of a personal computer, a desktop computer, a mobile computer, a laptop computer, a notebook computer, a terminal, a workstation, a server computer, a Personal Digital Assistant (PDA) device, a tablet computer, a specialized or dedicated computing device, a network device, or the like. Computing platform 110 may be implemented using any suitable combination of hardware components and/or software components.

[0010] In the example shown in FIG. 1, computing platform 110 may include, for example, one or more processors 141, one or more memory units 142, one or more storage units 143, an output unit 144, and an input unit 145. Computing platform 110 may include other suitable components or sets of components, as are known in the art, which may be operably associated with one or more components of computing platform 110.

[0011] Processor 141 may include, for example, a Central Processing Unit (CPU), a Digital Signal Processor (DSP), one or more controllers, or any suitable specific and/or general and/or multi-purpose processor or micro-processor or controller. Memory unit 142 may include, for example, a Random Access Memory (RAM), a Read Only Memory (ROM), a Dynamic RAM (DRAM), a Synchronous DRAM (SD-RAM), or other suitable memory units. Storage unit 143 may include, for example, a hard disk drive, a floppy disk drive, a Compact Disk (CD) drive, a CD-ROM drive, or other suitable removable or non-removable storage units or memory modules. Output unit 144 may include, for example, one or more cards, adapters, connectors and/or components able to connect to or communicate with a monitor. Input unit 145 may include, for example, one or more cards, adapters, connectors and/or components able to connect to or communicate with a keyboard, a mouse, or a touch-pad. Memory unit 142, storage unit

143, output unit 144 and/or input unit 145 may be operably associated with processor 141. It is noted that processor 141, memory unit 142, storage unit 143, output unit 144 and/or input unit 145 may include other suitable components and/or implementations as is known in the art.

[0012] In some embodiments, software may be stored, for example, in storage unit 143 or memory unit 142, and may be executed using processor 141. Such software may include, for example, one or more operating systems, for example, Microsoft Windows, Linux, Unix, Apple OS, Solaris, Sun-OS, HP-UX, or other suitable operating systems. The software may further include one or more software applications, one or more drivers, compilers, interpreters, emulators, execution layers, software environments, managed software environment, translation layers, and/or various other suitable software components. The software may include, for example, software components, software applications and/or software layers implementing or using one or more methods in accordance with embodiments of the invention, and/or other suitable software components. In some embodiments, the software and/or memory unit 142 may include one or more misaligned data items, e.g., a data item residing at a memory address in memory unit 142 that may not be efficiently accessed by processor 141.

[0013] In some embodiments, computing platform 110 may optionally include a translator 150, which may be implemented using any suitable combination of software components and/or hardware components. For example, in one embodiment, translator 150 may include software and/or instructions stored in memory unit 142 and/or in storage unit 143, which may be run or executed using processor 141. In some embodiments, translator 150 may translate and/or convert a software application or one or more instructions from a first format adapted to a first computing platform to a second format adapted to a second computing platform, e.g., adapted to computing platform 110. In one embodiment, translator 150 may be implemented, for example, using a dedicated software layer, e.g., an execution layer. In some embodiments, translator 150 may include one or more modules or components (not shown), to perform one or more operations of detecting and handling misaligned data access in accordance with embodiments of the invention, for example, a detection module, an instrumentation module, a light instrumentation module, a heavy instrumentation module, a translation module, a re-translation

module, an optimization module, a code modification module, a code classification module, or the like.

[0014] In some embodiments, translator 150 may detect and/or handle data misalignment as described below. For example, instructions or operations executed by or according to translator 150 may reflect a modified translation process in accordance with some embodiments of the invention, which may include modifications resulting from exemplary methods of detecting and handling data misalignment, as described below. For example, in some embodiments of the invention, translator 150 may implement a translation method or procedure that takes into account parameters related to data misalignment, as described in detail below.

[0015] FIG. 2 schematically illustrates a flow-chart of a method of detecting and handling data misalignment in accordance with some embodiments of the invention. The method of FIG. 2, as well as methods in accordance with other embodiments of the invention, may be used, for example, with computing platform 110 of FIG. 1, by translator 150, by processor 141, by software or instructions executed by computing platform 110, and/or with various other suitable computing platforms, devices, apparatuses and/or systems. In some embodiments, the method of FIG. 2 may be used by a compiler, an interpreter and/or an emulator, which may be implemented using suitable software components and/or hardware components.

[0016] It is noted that the phrases "block of code" and/or "code block" as used herein may include, for example, one or more instructions or sets of instructions. For example, a code block may include five instructions, or 20 instructions, etc.

[0017] In accordance with some embodiments of the invention, a "hot block" may include, for example, a code block identified and/or classified as a candidate for optimization or for further optimization operations. For example, a code block that may be performed multiple times, e.g., a large number of times, during execution of a software application, may be identified and/or classified as a hot block. Other suitable criteria or conditions may be used to identify, classify, and/or define a hot block.

[0018] In accordance with some embodiments of the invention, a "cold block" may include, for example, a code block that is not identified and/or classified as a hot block. For example, a cold block may include a code block that may be performed only once or a small number of times during execution of a software application. It is noted that in some embodiments, for example, a code block may be identified and/or classified as a cold block during initial translation operations, and may be re-classified as a hot block during further translation and/or optimization operations.

[0019] As indicated at block 210, the method may translate or convert a code block of a software application, from a first format adapted to a first computing platform to a second format adapted to a second computing platform. The code block in the first format may be referred to as "original code block", and the code block in the second format may be referred to as "translated code block". In some embodiments, for example, a code block may be translated from Intel (RTM) Architecture 32 (IA-32) to Intel (RTM) Architecture 64 (IA-64). The translation may be performed, for example, using translator 150. In one embodiment, the translation may be performed using a compiler software or an interpreter software. The translation may be performed, for example, dynamically and/or in real time, e.g., substantially at or near execution time. In some embodiments, the translation may be performed in advance, e.g., ahead of execution time.

[0020] As detailed herein, embodiments of the methods of the present invention may include instrumentation operations. In accordance with some embodiments of the invention, instrumentation may include, for example, adding one or more instructions to a certain code block or to a certain instruction, in order to track, examine, debug and/or analyze the behavior or the operations of that code block or instruction.

[0021] The phrase "light instrumentation" as used herein may include, for example, instrumentation which may be performed at a level of a code block. In one embodiment, for example, "light instrumentation" may include instrumentation which may detect whether or not

an execution of a code block may cause a misaligned data access, or whether or not an execution of a code block may result a misaligned data access event. In some embodiments, performing a "light instrumentation" may result in, for example, a determination whether or not an execution of a given code block includes a misaligned data access event.

[0022] The phrase "heavy instrumentation" as used herein may include, for example, instrumentation which may be performed at a level of an instruction. In one embodiment, for example, "heavy instrumentation" may include instrumentation which may detect whether or not an execution of an instruction block may cause a misaligned data access, or whether or not an execution of an instruction may result a misaligned data access event. In an alternate embodiment, "heavy instrumentation" may include, for example, instrumentation which may identify one or more instructions which execution may result a misaligned data access. In some embodiments, performing a "heavy instrumentation" may result in, for example, a determination of a location of an instruction whose execution results in a misaligned data access event.

[0023] In some embodiments, the phrase "detecting a misaligned data access" as used herein may include, for example, detecting that an execution or a subsequent execution of a translated code block includes a misaligned data access, or an access to a misaligned data item. In some embodiments, the phrase "detecting a misaligned data access" may additionally or alternatively include, for example, detecting a location of an instruction which, when executed, may result in a misaligned data access, or an access to a misaligned data item.

[0024] In some embodiments, the phrases "handling a misaligned data access", "avoiding a misaligned data access" and/or "preventing a misaligned data access" as used herein may include, for example, modifying a translated code block to create a modified translated code block, or re-translating an original code block to create a modified translated code block. In some embodiments, the modified translated code block, when executed, may access a misaligned data item more efficiently, more quickly, in a shorter time and/or using a smaller amount of processing cycles relative to a corresponding access to the misaligned data during an execution of the non-modified translated code block. In some embodiments, for example, "handling a

misaligned data access", "avoiding a misaligned data access" and/or "preventing a misaligned data access" may include accessing a misaligned data item in parts, e.g., in two parts, four parts, eight parts, sixteen parts, or another suitable number of parts, in multiple access stages, using multiple access instructions, using multiple loading instructions, using a dedicated code sequence, e.g., using Code 1 as detailed herein, or using other suitable ways in accordance with embodiments of the invention.

[0025] It is noted that in some embodiments, "handling a misaligned data access", "avoiding a misaligned data access" and/or "preventing a misaligned data access" may include, for example, other suitable ways of handling, avoiding, preventing, optimizing and/or curing a misaligned data access event. For example, in one embodiment, "handling a misaligned data access", "avoiding a misaligned data access" and/or "preventing a misaligned data access" may include curing the misalignment of a data item, e.g., by aligning the data item, moving the data item from a first memory location to a second memory location, copying the data item from a first memory location to a second memory location, or other suitable ways.

[0026] As indicated at block 220, the method may perform light instrumentation or analysis on the translated code block. In some embodiments, substantially all the instructions, or at least some of the instructions, whose execution may require misaligned data access may be lightly instrumented. The light instrumentation may include, for example, modification of one or more instructions such that the modified instruction may indicate an occurrence of a pre-defined condition or when pre-defined criteria are met. For example, in some embodiments, a signal may be produced if a misaligned data access event is detected in the translated cold block. In one embodiment, for example, the signal or indication may be provided by a detection module in translator 150 to a translation module in translator 150.

[0027] In some embodiments, a purpose or a result of the light instrumentation of block 220 may include, for example, to detect whether or not an execution of a translated code block may cause a misaligned data access event, such that further analysis and optimization operations may be applied, if required, to handle, cure, avoid or prevent a misaligned data access event. As

indicated at block 230, the method may check whether a misaligned data access event was detected in the translated code block. If the check result is negative, then, as indicated by arrow 235, which leads to block 270, the translated code block may be kept and/or used. If the check result is negative, then the translated code block may be further inspected, analyzed and/or optimized, as indicated at block 240 and onward.

[0028] The further inspection, analysis and optimization may include, as indicated at block 240, heavy instrumentation or analysis of the translated code block. In some embodiments, substantially all instructions whose execution may include a misaligned data access may be heavily instrumented. The heavy instrumentation may include, for example, modification of one or more instructions such that upon a misaligned data access event, information may be provided and/or registered regarding the misaligned data access. In some embodiments, the information may include, for example, indication of the instruction or instructions in the translated code block that caused the misaligned data access. The information may include, for example, indication of a type or a property of the misaligned data access, e.g., a granularity of the misalignment. For example, in one embodiment, an 8-bit data access that results in a misaligned data access event may be heavily instrumented to indicate that the misaligned data access is of 1-byte granularity, or of 4-byte granularity. It is noted that various other properties, characteristics, attributes and/or characteristics of a misaligned data access may be identified, detected, registered and/or analyzed.

[0029] As indicated at block 250, the heavy instrumentation of block 240 may be used for detection and handling of data access misalignment at the instruction level. In some embodiments, some or all or substantially all of the instructions that have been detected to result in a misaligned data access may be regenerated, re-translated, modified, optimized and/or replaced to handle, cure, avoid and/or prevent data access misalignment, to allow a relatively more efficient access to the misaligned data item, to allow a relatively faster access to the misaligned data item, or to allow using an alternate or specific way to access the misaligned data item

[0030] In one embodiment, the detection and handling operations indicated at block 250 may be performed, for example, during translation of a hot block. For example, a hot block may be identified, and the detection and handling operations indicated at block 250 may be performed on one or more cold blocks which may be included in the hot block.

[0031] In some embodiments, the detection and handling operations indicated at block 250 may be performed, for example, using the following pseudo-code:

```
// test bit0 to see if address is 2-byte aligned.
// Predicates p.mis and p.al set appropriately.
// Will use p.mis and p.al to predicate the following instructions
tbit p.mis,p.al    = r.addr, 0
// 2-byte load if aligned
(p.al) ld2 r.val   = [r.addr]
// if misaligned, load each byte separately
(p.mis) ld1 r.val  = [r.addr]
(p.mis) add r.addrH = 1, r.addr
(p.mis) ld1 r.valH = [r.addrH]
// combine the separately loaded bytes
(p.mis) dep r.val  = r.valH, r.val, 8, 8
```

Code 1

[0032] It is noted that Code 1 is presented for exemplary purposes only, and that embodiments of the invention are not limited in this regard; other suitable instructions, set of instructions, operations, pseudo-code or algorithm may be used in accordance with embodiments of the invention, in addition to Code 1 or instead of Code 1.

[0033] In some embodiments of the invention, for example, the detection and handling operations indicated at block 250 may use alternate codes, which may include various changes

relative to Code 1. In some embodiments, for example, one or more results of earlier detection of misaligned data access events may be registered and/or tracked to enhance and/or obviate one or several consequent detections of misaligned data access events. For example, in one embodiment, the location of memory addresses for which misaligned data access is detected, e.g., during the analysis of a hot block, may be registered and/or tracked. The registration or tracking may be performed, for example, using a suitable list, tracking list, stack, lookup table, array, database, variable, register, union, or in any other suitable way. The registration, tracking and/or storage of previous detections may be performed, for example, using processor 141, memory unit 142, storage unit 143, translator 150, suitable software executed using processor 141, and/or other suitable components of computing platform 110.

[0034] In the detection and handling operations indicated at block 250, some embodiments may, for example, analyze and/or perform instrumentation on a first instruction. As a result, it may be detected, for example, that accessing a first memory address by the first instruction may result in a misaligned data access. In such case, the location of the first memory address may be registered and/or tracked, e.g., using a registry or a tracking list which may be stored, for example, in memory unit 142 and/or storage unit 143. When a second instruction needs to access a second memory address, the method of the invention may check if the location of the second memory address is identical, for the purpose of data misalignment detection, to the location of the first memory address, e.g., using the tracking list or registry stored in memory unit 142 and/or storage unit 143. A positive check result may obviate the need to analyze and/or instrument the second instruction, for example, if an analysis of the first instruction already identifies the first memory address as a cause for a misaligned data access. It is noted that in some embodiments, different memory addresses may be identical for the purpose of data misalignment detection; for example, if the difference or the distance between the first and the second memory addresses is N bytes, then the first and the second memory addresses may be identical, for the purpose of data misalignment detection, in regard to an N-byte misalignment and/or in regard to a K-byte misalignment, wherein K is a factor of N. For example, in one embodiment, a memory address L and a memory address L+8 may be identical for the purpose

of detecting an 8-byte data misalignment, or for the purpose of detecting a 4-byte or a 2-byte data misalignment.

[0035] Similarly, in some embodiments, during the detection and handling operations indicated at block 250, some or all of the parameters, attributes, properties, variables, characteristics, results, checks and/or calculations used in one or more given detections of misaligned data access, may be used or re-used in a subsequent detection. For example, the two-byte load example of Code 1 may use one or more predicates that are set, determined and/or calculated in one or more previous detections or analysis of misaligned data access.

[0036] In some embodiments, performing the detection and handling operations indicated at block 250 may include, for example, using a code sequence or a set of instructions to perform a data access, e.g., when a misaligned data access is detected; the code sequence or set of instructions may be relatively long or may include a relatively large number of instructions or operations. In some embodiments, such code sequence or set of instructions may be moved to a suitable location in the software application. For example, in one embodiment, a scheduler process may move some or all of these instructions to a location external or to or remote from the translated code block, e.g., as a subroutine, a procedure, a function, or the like. During execution, instructions in the translated code block may be executed, until a "branch out" instruction is reached, at which point execution may proceed with the external or remote instructions. The external or remote instructions may include a last instruction, whose execution may result in a resumed execution of the translated code block from the point it "branched out" to the external or remote instructions.

[0037] As indicated at block 260, optionally, some embodiments may further optimize or analyze the translated code block, for example, to detect and handle misaligned data access, which may occur after completing the above operations, or after a hot block is translated. For example, in one embodiment, misaligned data access may appear after a relatively optimized translated code is generated. In some embodiments, this result may be eliminated, handled, avoided or mitigated, for example, using further optimization operations, which may be used in

conjunction with the detection and handling operations indicated at block 250, or at another suitable time. For example, during the operations indicated at block 250, it may be detected that although an instruction does not produce or cause a misaligned data access event, the instruction is a potential candidate to result in misaligned data access during execution. Such instruction may be referred to as a "candidate instruction". Identification, detection, classification, registration and/or evaluation of a candidate instruction may be performed, for example, in relation to one or more suitable criteria or conditions, e.g., the type of the instruction, an estimated probability that the instruction will result in a misaligned data access, or a detection of a significant probability that the instruction will result in a misaligned data access.

[0038] In accordance with some embodiments of the invention, the operations indicated at block 260 may optionally include, for example, upon detection of a candidate instruction, performing an instrumentation on the candidate instruction. If a misaligned data access is detected using the instrumentation used, then the translated block may be discarded. Additionally, the method may register an indication that when the original code block is translated, substantially all candidate instructions in the original code block may be translated in a way that detects and handles misaligned data access, e.g., using Code 1 or any other suitable code sequence. The registered indication may be stored, for example, in translator 150, memory unit 142 and/or storage unit 143. In some embodiments, the indication may relate not only to the candidate instruction, but also to substantially all the instructions of the type of the candidate instruction. Upon translation of the original code block, the registered indication may be detected or tracked, and accordingly, substantially all candidate instructions in the original code block, or all instructions of the type of a candidate instruction in the original code block, may be translated in a way that detects and handles misaligned data access, e.g., using Code 1 or any other suitable code sequence.

[0039] In some embodiments, one or more of the operations described above, e.g., the data misalignment detection and handling operations of block 240, the code modification described with reference to block 240, or the optimization operations of block 260, may be performed by one or more suitable software and/or hardware components. For example, in some embodiments, code modification may be performed by translator 150, or by a module or a

component external to translator 150. In one embodiment, modification of a code may be performed, for example, by modifying translator 150 or by modifying another suitable component or module, whose modification may result a code modification. Similarly, detection and/or handling of data misalignment may be performed by translator 150, or by one or more modules which may be internal or external to translator 150.

[0040] In some embodiments of the invention, one or more of the operations described above, e.g., the translation operations of block 210, the light instrumentation operations of block 220, the data misalignment check of block 230, the heavy instrumentation operations of block 240, the data misalignment detection and handling operations of block 250, the optimization operations of block 260, or other suitable operations, may be repeated, or may be performed more than once or in various orders of operation. Furthermore, in some embodiments, as a result of one or more of these operations, a translated code block may be discarded, deleted, replaced, inspected, analyzed, modified, optimized, re-generated, re-produced, or re-translated. As indicated at block 270, after performing one or more or all of the above operations, or several iterations or repetitions of one or more of the above operations, the resulting translated code block may be kept and used, or may be registered as a translated code block which may be kept and used.

[0041] Embodiments of the invention may allow various benefits. For example, in some embodiments, data misalignment detection and handling may significantly decrease processing time and/or processing time, or may significantly improve performance. In one embodiment, for example, a workload that requires about 1,236 seconds to complete without data misalignment detection and handling, may require only about 133 seconds to complete using data misalignment detection and handling in accordance with an embodiment of the invention. Some embodiments of the invention may allow various other benefits, in addition to or instead of the benefits described.

[0042] Some embodiments of the invention may be implemented, for example, using a machine-readable medium or article which may store an instruction or a set of instructions that,

if executed by a machine, for example, by computing platform 110, by processor 141, or by other suitable machines, cause the machine to perform a method and/or operations in accordance with embodiments of the invention. Such machine may include, for example, any suitable processing platform, computing platform, computing device, processing device, computing system, processing system, computer, processor, or the like, and may be implemented using any suitable combination of hardware and/or software. The machine-readable medium or article may include, for example, any suitable type of memory unit (e.g., memory unit 142), memory device, memory article, memory medium, storage device, storage article, storage medium and/or storage unit (e.g., storage unit 143), for example, memory, removable or non-removable media, erasable or non-erasable media, writeable or re-writeable media, digital or analog media, hard disk, floppy disk, Compact Disk Read Only Memory (CD-ROM), Compact Disk Recordable (CD-R), Compact Disk Rewriteable (CD-RW), optical disk, magnetic media, various types of Digital Versatile Disks (DVDs), a tape, a cassette, or the like. The instructions may include any suitable type of code, for example, source code, compiled code, interpreted code, executable code, static code, dynamic code, or the like, and may be implemented using any suitable high-level, low-level, object-oriented, visual, compiled and/or interpreted programming language, e.g., C, C++, Java, BASIC, Pascal, Fortran, Cobol, assembly language, machine code, or the like.

[0043] Some embodiments of the invention may be implemented by software, by hardware, or by any combination of software and/or hardware as may be suitable for specific applications or in accordance with specific design requirements. Embodiments of the invention may include units and/or sub-units, which may be separate of each other or combined together, in whole or in part, and may be implemented using specific, multi-purpose or general processors, or devices as are known in the art. Some embodiments of the invention may include buffers, registers, storage units and/or memory units, for temporary or long-term storage of data or in order to facilitate the operation of a specific embodiment.

[0044] While certain features of the invention have been illustrated and described herein, many modifications, substitutions, changes, and/or equivalents may occur to those skilled in the art. It

is, therefore, to be understood that the appended claims are intended to cover all such modifications and/or changes.

CLAIMS

What is claimed is:

1 A method comprising:
detecting misaligned data access resulting from execution of a code block translated from a first format suitable for a first computing platform to a second format suitable for a second computing platform; and
modifying said code block according to said misaligned data access.

2. The method of claim 1, wherein detecting comprises performing instrumentation of said code block to detect whether execution of said code block results in the misaligned data access.

3. The method of claim 2, wherein detecting comprises performing instrumentation of at least one instruction in said code block to detect a location of an instruction whose execution results in the misaligned data access.

4. The method of claim 1, wherein detecting comprises performing instrumentation of at least one instruction in said code block to detect a location of an instruction whose execution results in the misaligned data access.

5. The method of claim 1, wherein modifying comprises adding to said code block an instruction to branch an execution of said code block to a code sequence whose execution handles the misaligned data access.

6. The method of claim 1, wherein modifying comprises modifying said code block to handle misaligned data access in a subsequent execution of said code block.

7. The method of claim 1, further comprising translating said code block from said first format to said second format.

8. The method of claim 1, wherein detecting comprises detecting a misaligned data access resulting from an execution of a code block translated from a format suitable for a 32-bit based computing platform to a format suitable for a 64-bit based computing platform.

9 An apparatus comprising:
a processor to detect misaligned data access resulting from execution of a code block translated from a first format suitable for a first computing platform to a second format suitable for a second computing platform, and to modify said code block according to said misaligned data access.

10. The apparatus of claim 9, wherein the processor is able to perform instrumentation of said code block to detect whether execution of said code block results in the misaligned data access.

11. The apparatus of claim 10, wherein the processor is able to perform instrumentation of at least one instruction in said code block to detect a location of an instruction whose execution results in the misaligned data access.

12. The apparatus of claim 9, wherein the processor is able to perform instrumentation of at least one instruction in said code block to detect a location of an instruction whose execution results in the misaligned data access.

13. The apparatus of claim 9, wherein the processor is able to add to said code block an instruction to branch an execution of said code block to a code sequence whose execution handles the misaligned data access.

14. The apparatus of claim 9, wherein the processor is able to modify said code block to handle misaligned data access in a subsequent execution of said code block.

15. The apparatus of claim 9, wherein the processor is able to, before detecting the misaligned data access, translate said code block from said first format to said second format

16. The apparatus of claim 9, wherein the first computing platform is a 32-bit based computing platform and the second computer architecture is a 64-bit based computing platform.

17. A computing platform comprising:
a processor to detect misaligned data access resulting from execution of a code block translated from a first format suitable for a first computing platform to a second format suitable for a second computing platform, and to modify said code block according to said misaligned data access; and
a dynamic random access memory operably associated with said processor to store at least a portion of said code block.

18. The apparatus of claim 17, wherein the processor is able to perform instrumentation of said code block to detect whether execution of said code block results in the misaligned data access.

19. The apparatus of claim 18, wherein the processor is able to perform instrumentation of at least one instruction in said code block to detect a location of an instruction whose execution results in the misaligned data access.

20. The apparatus of claim 17, wherein the processor is able to perform instrumentation of at least one instruction in said code block to detect a location of an instruction whose execution results in the misaligned data access.

21. A machine-readable medium having stored thereon a set of instructions that, if executed by a machine, cause the machine to perform a method comprising:
detecting misaligned data access resulting from execution of a code block translated from a first format suitable for a first computing platform to a second format suitable for a second computing platform; and
modifying said code block according to said misaligned data access.

22. The machine-readable medium of claim 21, wherein the instructions that result in detecting result in performing instrumentation of said code block to detect whether execution of said code block results in the misaligned data access.

23. The machine-readable medium of claim 22, wherein the instructions that result in detecting result in performing instrumentation of at least one instruction in said code block to detect a location of an instruction whose execution results in the misaligned data access.

24. The machine-readable medium of claim 21, wherein the instructions that result in detecting result in performing instrumentation of at least one instruction in said code block to detect a location of an instruction whose execution results in the misaligned data access.

25. The machine-readable medium of claim 21, wherein the instructions comprise at least part of a translator.

26. The machine-readable medium of claim 21, wherein the instructions comprise at least part of an execution layer.

27. The machine-readable medium of claim 21, wherein the instructions comprise at least part of an operating system.

28. The machine-readable medium of claim 21, wherein the instructions comprise at least part of a compiler.

Abstract

Device, system and method for detection and handling of misaligned data access. A method may include, for example, detecting misaligned data access resulting from execution of a code block translated from a first format suitable for a first computing platform to a second format suitable for a second computing platform, and modifying said code block according to said misaligned data access.

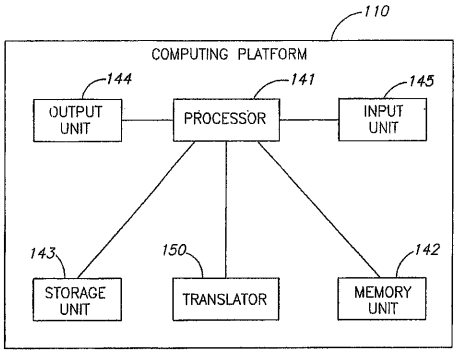

110
COMPUTING PLATFORM
144
141
145
OUTPUT UNIT
PROCESSOR
INPUT UNIT
143
150
142
STORAGE UNIT
TRANSLATOR
MEMORY UNIT

FIG.1

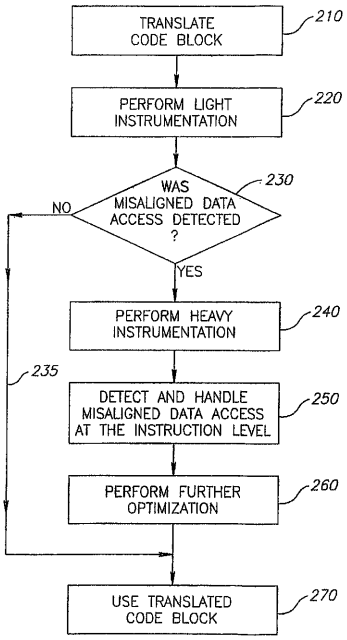

TRANSLATE CODE BLOCK
210
PERFORM LIGHT INSTRUMENTATION
220
WAS MISALIGNED DATA ACCESS DETECTED ?
230
NO
YES
PERFORM HEAVY INSTRUMENTATION
240
235
DETECT AND HANDLE MISALIGNED DATA ACCESS AT THE INSTRUCTION LEVEL
250
PERFORM FURTHER OPTIMIZATION
260
USE TRANSLATED CODE BLOCK
270

FIG.2