(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3569014号
(P3569014)
(45)発行日　平成16年9月22日(2004.9.22)　(24)登録日　平成16年6月25日(2004.6.25)

(51)Int.Cl.[7]　F I
G06F 9/38　G06F 9/38 310X
G06F 9/46　G06F 9/38 350Y
　　G06F 9/46 310Q
　　G06F 9/46 313D

請求項の数 34　(全 27 頁)

(21)出願番号　特願平6-290742
(22)出願日　平成6年11月25日(1994.11.25)
(65)公開番号　特開平8-147165
(43)公開日　平成8年6月7日(1996.6.7)
審査請求日　平成13年1月19日(2001.1.19)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100079359
弁理士　竹内　進
(74)代理人　100093584
弁理士　宮内　佐一郎
(72)発明者　木村　康則
神奈川県川崎市中原区上小田中１０１５番地　富士通株式会社内

審査官　後藤　彰

最終頁に続く

(54)【発明の名称】マルチコンテキストをサポートするプロセッサおよび処理方法

(57)【特許請求の範囲】
【請求項１】
ある意味のあるまとまった仕事をするための実行単位であるコンテキストを格納した複数のコンテキスト格納部と、
あるコンテキストの命令をパイプラインに流して実行すると共に、該パイプラインの空きを判断した場合は、実行途中にある別のコンテキストに切り替えて同時に複数のコンテキストを実行させる命令実行部と、
前記命令実行部で同時に実行している前記複数のコンテキストの各々に固有のコンテキストＩＤを設定するＩＤ設定部と、
前記命令実行部で同時に複数のコンテキストを実行する際に使用するレジスタ名を、前記ＩＤ設定部で設定された実行中のコンテキストＩＤに実行命令の指定レジスタ名を加えたレジスタ名に名前替えして物理レジスタを割り当てるレジスタ・リネーミング部と、
を備え、
前記コンテキストを構成する命令列は、プログラムコードに加え、プログラムコードの意味は変えないがプログラムコードの実行を助けるための種々の情報を格納する属性情報フィールドを持ち、該属性情報フィールドに、実行途中にある別のコンテキスト切替えの契機となる後続命令の実行に必要な命令実行情報を格納したことを特徴とするマルチコンテキストをサポートするプロセッサ。
【請求項２】
請求項１記載のマルチコンテキストをサポートするプロセッサに於いて、

前記コンテキスト切替えを発生する後続命令はロード命令であり、該ロード命令の実行でメモリからデータが届くまでのレイテンシ分のインターバルをもつ先行する命令コードの属性情報フィールドに、この命令の実行と同時に実行途中にある別のコンテキストの命令フェッチを指示する属性情報を設けたことを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項３】

請求項１記載のマルチコンテキストをサポートするプロセッサに於いて、

前記コンテキスト切替えを発生する後続命令は分岐命令であり、該分岐命令の実行でメモリからデータが届くまでのレイテンシ分のインターバルをもつ先行する命令コードの属性情報フィールドに、この命令の実行と同時に実行途中にある別のコンテキストの命令フェッチを指示する属性情報を設けたことを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項４】

請求項１記載のマルチコンテキストをサポートするプロセッサに於いて、

前記命令実行部は、

少なくとも２つの命令格納バッファと、複数のコンテキストいずれか１つから前記命令格納バッファのいずれか１つに命令をプリフェッチするプリフェッチルートを切り替える第１ルータと、

複数の命令格納バッファのいずれか１つから前記パイプラインにフェッチするフェッチルートを切り替える第２ルータと、

現在パイプラインに流しているコンテキストの命令の属性情報から実行途中にある別のコンテキストへの切替えの契機となる後続命令の実行に必要な命令実行指示情報を解読した際に、前記第１ルータのプリフェッチルートを、切替先のコンテキストから現在命令フェッチに使用されていない別の命令バッファへのプリフェッチルートに切替えて命令プリフェッチを行わせ、続いて別のコンテキスト切替えの契機となる前記後続命令の実行を判別した際に、前記第２ルータのフェッチルートを、前記第１ルータでプリフェッチルートを切り替えた側の命令バッファに切り替えてプリフェッチされている別のコンテキストの命令をパイプラインにフェッチさせるコンテキスト切替部と、

を備えたことを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項５】

請求項１記載のマルチコンテキストをサポートするプロセッサに於いて、

前記命令実行部は、

実行途中にある別のコンテキスト切替えの契機となる後続命令を、該後続命令の実行に伴うハードウェアの信号から認識してコンテキストを切り替えることを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項６】

請求項５記載のマルチコンテキストをサポートするプロセッサに於いて、

前記命令実行部は、

前記後続命令がロード命令の場合、前記コンテキスト切替えを、前記ロード命令の実行におけるキャッシュのミスヒットを示す信号から認識することを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項７】

請求項５記載のマルチコンテキストをサポートするプロセッサに於いて、

前記命令実行部は、

前記後続命令が分岐命令の場合、前記コンテキスト切替えを、前記ロード命令の実行における分岐条件成立時（taken)を示す分岐ターゲットバッファの出力から認識することを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項８】

請求項１記載のマルチコンテキストをサポートするプロセッサに於いて、

前記レジスタ・リネーミング部は、

実行中のコンテキストＩＤを格納するＩＤレジスタと、
実行中のコンテキストの命令をフェッチする命令レジスタと、
有効フラグフィールド、レジスタキーフィールド及びデータフィールドを備えた複数の物理レジスタをメモリ上にマッピングした名前替えレジスタファイルと、
前記ＩＤレジスタのコンテキストＩＤに前記命令レジスタのレジスタ指定フィールドのレジスタ名を加えたレジスタキーを作成して前記レジスタファイルの該当する物理レジスタを名前替えレジスタとして割り当てるレジスタ割付部と、
前記レジスタファイルからオーバフローした名前替えレジスタのデータを退避する実行中のコンテキスト毎に設けられたレジスタ退避部と、
前記レジスタファイルから前記レジスタ退避部に名前替えレジスタのデータを退避させる退避処理部と、
前記レジスタ割付け部による前記レジスタファイルの参照で前記レジスタキーに対応するレジスタが存在しなかった場合、前記レジスタ退避部から対応するレジスタキーのデータを前記レジスタファイルにロードさせるロード処理部と、
を備えたことを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項９】
請求項８記載のマルチコンテキストをサポートするプロセッサに於いて、
前記名前替えレジスタファイルは、前記命令レジスタのレジスタ指定フィールドで指定可能な数の物理レジスタを有することを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項１０】
請求項８記載のマルチコンテキストをサポートするプロセッサに於いて、
前記名前替えレジスタファイルは、前記レジスタ割付部による名前替えレジスタの割付けで前記有効フラグをオンして使用状態を表し、前記退避処理部によるレジスタデータの退避で有効フラグをオフして空き状態を表すことを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項１１】
請求項８記載のマルチコンテキストをサポートするプロセッサに於いて、
前記レジスタ退避部は、退避データを格納するデータフィールドに加え、有効フラグフィールドを有し、前記退避処理部によるレジスタデータの退避により有効フラグをオンしてデータ退避状態を表し、前記ロード処理部による退避データのロードで前記有効フラグをオフして前記名前替えレジスタファイルに存在することを表すことを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項１２】
請求項８記載のマルチコンテキストをサポートするプロセッサに於いて、
前記退避処理部は、前記名前替えレジスタファイルでオーバフローが発生した際に、実行待ち状態にある他のコンテキストのいずれか１つのコンテキストＩＤをもつ全ての前記物理レジスタのデータを、対応するコンテキストのレジスタ退避部に退避させることを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項１３】
請求項８記載のマルチコンテキストをサポートするプロセッサに於いて、
前記ロード処理部は、前記レジスタ退避部から実行中のコンテキストのレジスタデータを前記名前替えレジスタファイルにロードするに先立ち、前記退避処理部によって実行待ち状態にある他のコンテキストのいずれか１つのコンテキストＩＤをもつ前記物理レジスタのデータを、対応するコンテキストのレジスタ退避部に退避させることを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項１４】
請求項８記載のマルチコンテキストをサポートするプロセッサに於いて、
前記退避処理部およびロード処理部は、１つの名前替えレジスタ単位にレジスタデータの退避とロードを行うことを特徴とするマルチコンテキストをサポートするプロセッサ。

【請求項１５】
請求項１記載のマルチコンテキストをサポートするプロセッサに於いて、
前記複数のコンテキストは、異なったアドレス空間で実行される複数のプロセスであることを特徴とするマルチコンテキストをサポートするプロセッサ。
【請求項１６】
請求項１記載のマルチコンテキストをサポートするプロセッサに於いて、
前記複数のコンテキストは、同一アドレス空間で実行される複数のスレッドであることを特徴とするマルチコンテキストをサポートするプロセッサ。
【請求項１７】
請求項１６記載のマルチコンテキストをサポートするプロセッサに於いて、
前記スレッドは、数値計算プログラムのループであることを特徴とするマルチコンテキストをサポートするプロセッサ。
【請求項１８】
ある意味のあるまとまった仕事をするための実行単位であるコンテキストを複数作成する作成過程と、
あるコンテキストの命令をパイプラインに流して実行すると共に、該パイプラインの空きを判断した場合は、実行途中にある別のコンテキストに切り替えて同時に複数のコンテキストを実行させる命令実行過程と、
同時に実行している前記複数のコンテキストの各々に固有のコンテキストＩＤを設定するＩＤ設定過程と、
前記コンテキストの実行で使用するレジスタ名を、前記コンテキストＩＤに実行命令の指定レジスタ名を加えたレジスタ名に名前替えして物理レジスタを割り当てるレジスタ・リネーミング過程と、
を備え、
前記コンテキストを構成する命令列は、プログラムコードに加え、プログラムコードの意味は変えないがプログラムコードの実行を助けるための種々の情報を格納する属性情報フィールドを持ち、該属性情報フィールドに、実行途中にある別のコンテキスト切替えの契機となる後続命令の実行に必要な命令実行情報を設けたことを特徴とするマルチコンテキスト処理方法。
【請求項１９】
請求項１８記載のマルチコンテキスト処理方法に於いて、
前記コンテキスト切替えを発生する後続命令はロード命令であり、該ロード命令の実行でメモリからデータが届くまでのレイテンシ分のインターバルをもつ先行する命令コードの属性情報フィールドに、この命令の実行と同時に実行途中にある別のコンテキストの命令フェッチを指示する属性情報を設けたことを特徴とするマルチコンテキスト処理方法。
【請求項２０】
請求項１８記載のマルチコンテキスト処理方法に於いて、
前記コンテキスト切替えを発生する後続命令は分岐命令であり、該分岐命令の実行でメモリからデータが届くまでのレイテンシ分のインターバルをもつ先行する命令コードの属性情報フィールドに、この命令の実行と同時に実行途中にある別のコンテキストの命令フェッチを指示する属性情報を設けたことを特徴とするマルチコンテキスト処理方法。
【請求項２１】
請求項１８記載のマルチコンテキスト処理方法に於いて、
前記命令実行過程は、
実行中のコンテキストの命令を命令バッファにプリフェッチした後にパイプラインにフェッチして実行し、
前記実行中のコンテキストの命令の属性情報から実行途中にある別のコンテキストへの切替えの契機となる後続命令の実行に必要な命令実行指示情報を解読した際に、切替先のコンテキストから現在命令フェッチに使用されていない別の命令バッファへのプリフェッチに切り替え、

続いて別のコンテキスト切替えの契機となる前記後続命令の実行を判別した際に、プリフェッチを切替え先の命令バッファにプリフェッチされている別のコンテキストの命令をパイプラインにフェッチさせることを特徴とするマルチコンテキスト処理方法。

【請求項２２】

請求項１８記載のマルチコンテキスト処理方法に於いて、

前記命令実行過程は、実行途中にある別のコンテキストへの切替えの契機となる後続命令を、該後続命令の実行に伴うハードウェアの信号から認識してコンテキストを切り替えることを特徴とするマルチコンテキスト処理方法。

【請求項２３】

請求項２２記載のマルチコンテキスト処理方法に於いて、

前記命令実行過程は、前記後続命令がロード命令の場合、前記コンテキスト切替えを、前記ロード命令の実行におけるキャッシュのミスヒットを示す信号から認識することを特徴とするマルチコンテキスト処理方法。

【請求項２４】

請求項２２記載のマルチコンテキスト処理方法に於いて、

前記命令実行過程は、前記後続命令が分岐命令の場合、前記コンテキスト切替えを、前記ロード命令の実行における分岐条件成立時（taken)を示す分岐ターゲットバッファの出力から認識することを特徴とするマルチコンテキスト処理方法。

【請求項２５】

請求項１８記載のマルチコンテキスト処理方法に於いて、

前記名前替え過程は、

実行中のコンテキストＩＤをＩＤレジスタに格納し、

実行中のコンテキストの命令を命令レジスタにフェッチし、

実行中のコンテキストＩＤに、前記命令レジスタにフェッチしている命令中のレジスタ指定フィールドのレジスタ名を加えたレジスタキーを作成し、

有効フラグフィールド、レジスタキーフィールド及びデータフィールドを備えた複数の物理レジスタを有する名前替えレジスタファイルの該当する物理レジスタを、前記レジスタキーで検索して名前替えレジスタとして割り付け、

前記レジスタファイルのオーバフロー発生時に、前記名前替えレジスタのデータを、実行中のコンテキスト毎に設けられたレジスタ退避部に退避し、

前記レジスタファイルの参照で前記レジスタキーに対応するレジスタが存在しなかった場合、前記レジスタ退避部から対応するレジスタキーのデータを前記レジスタファイルにロードさせることを特徴とするマルチコンテキスト処理方法。

【請求項２６】

請求項２５記載のマルチコンテキスト処理方法に於いて、

前記名前替えレジスタファイルは、前記命令レジスタのレジスタ指定フィールドで指定可能な数の物理レジスタを有することを特徴とするマルチコンテキスト処理方法。

【請求項２７】

請求項２５記載のマルチコンテキスト処理方法に於いて、

前記物理レジスタに名前替えレジスタを割り付けた際に、前記有効フラグをオンして使用状態を表し、前記レジスタデータの退避で前記有効フラグをオフして空き状態を表すことを特徴とするマルチコンテキスト処理方法。

【請求項２８】

請求項２５記載のマルチコンテキスト処理方法に於いて、

前記レジスタ退避部は、退避データを格納するデータフィールドに加え、有効フラグフィールドを有し、前記名前替えレジスタファイルからのレジスタデータの退避により有効フラグをオンしてデータ退避状態を表し、前記名前替えレジスタファイルへの退避データのロードで前記有効フラグをオフして前記名前替えレジスタファイルに存在することを表すことを特徴とするマルチコンテキスト処理方法。

【請求項２９】

請求項２５記載のマルチコンテキスト処理方法に於いて、
前記名前替えレジスタファイルでオーバフローが発生した際に、実行待ち状態にある他のコンテキストのいずれか１つのコンテキストＩＤをもつ全ての前記物理レジスタのデータを、対応するコンテキストのレジスタ退避部に退避させることを特徴とするマルチコンテキスト処理方法。
【請求項３０】
請求項２５記載のマルチコンテキスト処理方法に於いて、
前記レジスタ退避部から実行中のコンテキストのレジスタデータを前記名前替えレジスタファイルにロードするに先立ち、実行待ち状態にある他のコンテキストのいずれか１つのコンテキストＩＤをもつ前記物理レジスタのデータを、対応するコンテキストのレジスタ退避部に退避させることを特徴とするマルチコンテキスト処理方法。
【請求項３１】
請求項２５記載のマルチコンテキスト処理方法に於いて、
１つの名前替えレジスタ単位にレジスタデータの退避とロードを行うことを特徴とするマルチコンテキスト処理方法。
【請求項３２】
請求項１８記載のマルチコンテキスト処理方法に於いて、
前記複数のコンテキストは、異なったアドレス空間で実行される複数のプロセスであることを特徴とするマルチコンテキスト処理方法。
【請求項３３】
請求項１８記載のマルチコンテキスト処理方法に於いて、
前記複数のコンテキストは、同一アドレス空間で実行される複数のスレッドであることを特徴とするマルチコンテキスト処理方法。
【請求項３４】
請求項３３記載のマルチコンテキスト処理方法に於いて、
前記スレッドは、数値計算プログラムのループであることを特徴とするマルチコンテキスト処理方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】
本発明は、スーパスカラのアーキテクチャをもつプロセッサ及び処理方法に関し、特に、複数のコンテキストを同時にパイプラインに流して実行するマルチコンテキストをサポートするプロセッサ及び処理方法に関する。
最初に、本発明で使用するマルチコンテキスト（Ｍｕｌｔｉ－Ｃｏｎｔｅｘｔ） を説明する。まずコンテキストを、「ある意味のあるまとまった仕事をするための計算機上の実行単位」と定義する。コンテキストには、例えばユニックス（ＵＮＩＸ） のアドレス空間の異なるプロセスや、マッシュ（ＭＡＣＨ） でいう同一アドレス空間のスレッドを含む。即ち、複数のコンテキストは、独立したメモリ空間を持っているか、同じメモリ空間を持っているかを問題にせず、両方を包含する。
【０００２】
したがって、マルチコンテキストとは、同時に実行可能な複数のコンテキストの集合である。本発明のマルチコンテキストをサポートするプロセッサは、複数のコンテキストをパイプラインに効率良く投入して実行することを目指している。
【０００３】
【従来の技術】
従来、スーパスカラプロセッサのパイプラインには、一般に単一のユーザプログラムから単一の命令列にコンパイルされたコードが投入され、実行時にハードウエアによって並列性が抽出されて、並列に実行されている。しかし、この方法では、命令間に依存関係がある場合や、キャッシュのミスヒット、分岐命令が実行された場合などには、実行が中断してパイプラインに空きが出来てしまう。これをパイプラインのストール、あるいはパイプ

ラインのバブル発生という。このためプロセッサが本来持っている性能をフルに引き出すことはできない。

【０００４】

そこで、パイプラインのストールを解消するため、複数のプロセスの命令列を、ある一定の間隔でパイプラインに投入し、お互いの命令列から生じるパイプラインのストールを相殺する方法がある。しかし、この方法では、命令列の切替えが間隔が一定であるため、プログラムの性質を使ったスケジューリングができず、ストール解消の効果が小さい。

【０００５】

また複数のコンテキストを同時に実行させるためには、各コンテキストの実行環境を保持するためのレジスタなどをコンテキスト分持たなければならず、資源を大量に使ってしまう。逆に、ある一定の資源をコンテキスト分に分割して使う場合には、一つのコンテキストが使える資源が減ってしまうという欠点があった。

【０００６】

この方法の一例として、ＨＥＰを説明する。ＨＥＰは、図１３のように、複数の命令ストリームを機械的に一命令ずつパイプラインに投入するものであり、メカニズムとしては簡単である。図１３は、ｎ個のプログラムを同時に一つのパイプラインに投入している。ここでプログラム１～ｎの各々が本発明のコンテキストである。実行時には、

プログラム１の命令１
プログラム２の命令１
・・・・
プログラムｎの命令１
プログラム１の命令２
・・・

の順で命令がパイプライン２００に投入され、実行される。パイプライン２００は４段で構成される。フェッチ段Ｆは命令をキャッシュから取り出す。デコード段Ｄは命令を解読する。実行段ＥＸ（Ｅｘｅｃｕｔｅ　）は命令を実行する。ストア段Ｓ（Ｓｔｏｒｅ　）は実行結果を格納する。

【０００７】

いま、プログラム１の命令１がロード命令で、プログラム１の命令２が命令１でロードした結果を使うような場合、プログラム１だけで実行している場合には、命令１と命令２の実行の間にパイプラインのストールが生じる。一般にロード命令はメモリをアクセスするため、直後のサイクルでロード結果を使うことはできず、パイプラインのストールが生じる。

【０００８】

図１３のＨＥＰの場合、プログラム１の命令１と命令２の実行の間には、プログラム１～プログラムｎの命令１の実行が挟まれるため、プログラム１の命令１がロード命令であった場合のパイプライン２００のストールは実効的に無くなり、プログラム実行のスループットを上げることができる。

【０００９】

【発明が解決しようとする課題】

しかし、パイプライン２００への命令投入に当たって、複数のプログラム１～ｎの命令ストリーム間の関係を考慮していないため、パイプライン２００のストール発生を常に減少できるかどうかの保証はない。また、実効時には同時に動くコンテキスト分（プログラム分）だけ作業レジスタを用意しなければならず、レジスタ数に対する要求は大きくなる。

【００１０】

逆の言い方をすると、ハードウェアで用意できるレジスタ数が固定の場合には、１つのコンテキストが使えるレジスタの数が減ることになる。図１３のＨＥＰの例では、単純にはｎ個のプログラム分のレジスタセットを一時に用意しなければならない。更に、このような単純な命令ストリームの投入方法では、命令キャッシュ、データキャッシュのヒット率が悪くなる可能性も大きい。

【００１１】
一方、パイプラインのストールを解消する別の方法として、単一のプログラムからスレッドと呼ばれる並列に実行可能なプログラム単位を切り出し、これを同時にパイプラインに流して実行する方法もある。しかし、この方法はフォートラン（ＦＯＲＴＲＡＮ）などで書かれた数値計算プログラムのループを一つのスレッドとするような、単純な場合にのみ適用可能であることが多く、一般のプログラムに対してコンパイラでスレッドを抽出することは極めて困難であった。
【００１２】
本発明の目的は、スパースカラプロセッサのパイプラインのストールを減らし、複数のコンテキストを同時に効率良くパイプラインに流して性能を向上させるマルチコンテキストをサポートするプロセッサとその処理方法を提供する。
【００１３】
【課題を解決するための手段】
図１は本発明の原理説明図である。まず、ある意味のあるまとまった仕事をするための実行単位である複数のコンテキスト５２－１，５２－２が、キャッシュメモリなどのコンテキスト格納部（メモリ空間）１６－１に格納されている。命令実行部２５は、パイプライン１５に、あるコンテキスト５２－１の命令を流して実行すると共に、パイプライン１５の空きを判断した場合は、実行途中にある別のコンテキスト５２－２に切り替えて同時に複数のコンテキストを実行させる。
【００１４】
ＩＤ設定部６２－１，６２－２は、命令実行部１０で同時に実行している複数のコンテキスト５２－１，５２－２の各々に固有のコンテキストＩＤとしてＣＩＤｉ（ｉ＝０，１）を設定する。
レジスタ・リネーミング部（レジスタリネーミング部）２５は、命令実行部１０で同時に複数のコンテキスト５２－１，５２－２を実行する際に使用するレジスタ名を、ＩＤ設定部６２－１，６２－２で設定された実行中のコンテキストＩＤであるＣＩＤｉに実行命令の指定レジスタ名　Ｒｊ　（ｊ＝１，２，３，・・・ｍ）を加えたレジスタ名ＣＩＤｉ－Ｒｊ　に名前替えして物理レジスタを割り当てる。
【００１５】
このレジスタ・リネーミング部２５は、マルチ・リネーミング機能によって、複数のコンテキストを同時にパイプラインに流したときの１つのコンテキスト分の物理レジスタだけで、全てのコンテキストの実行に必要なマルチ・レジスタリネーミングを行う。
コンテキストの切替えは、コンテキストをハードウェアとソフトウェアの組合せ、又はハードウェアのみによって行われる。
【００１６】
ハードウェアとソフトウェアの組合せでコンテキストを切り替える場合、コンテキスト５２－１，５２－２を構成する命令列は、プログラムコードに加え、プログラムコードの意味は変えないがプログラムコードの実行を助けるための種々の情報を格納する属性情報フィールドを持つ。属性情報フィールドには、実行途中にある別のコンテキスト切替えの契機となる後続命令の実行に必要な命令実行情報が格納される。
【００１７】
コンテキスト切替えの契機となる後続命令には、ロード命令と分岐命令がある。後続命令はロード命令の場合、ロード命令の実行でメモリからデータが届くまでのレイテンシ分のインターバルをもつ先行する命令コードの属性情報フィールドに、この命令の実行と同時に実行途中にある別のコンテキストの命令フェッチを指示する属性情報を設ける。後続命令が分岐命令の場合も、同様に、分岐命令の実行でメモリからデータが届くまでのレイテンシ分のインターバルをもつ先行する命令コードの属性情報フィールドに、この命令の実行と同時に実行途中にある別のコンテキストの命令フェッチを指示する属性情報を設ける。
【００１８】

属性情報を利用したコンテキスト切替えのため、命令実行部１０は、少なくとも２つの命令バッファ５６－１，５６－２を持つ。また複数のコンテキスト５２－１，５２－２いずれか１つから命令バッファ５６－１，５６－２のいずれか１つに命令をプリフェッチするプリフェッチルートを切り替える第１ルータ５４と、命令格納バッファ５６－１，５６－２のいずれか１つからパイプライン１５にフェッチするフェッチルートを切り替える第２ルータ５８を設ける。

【００１９】

コンテキスト切替部６０は、現在パイプライン１５に流しているコンテキスト５２－１の命令の属性情報から、実行途中にある別のコンテキストへの切替えの契機となる後続命令の実行に必要な命令実行指示情報を解読した際に、第１ルータ５４のプリフェッチルートを、切替先のコンテキスト５２－１から現在命令フェッチに使用されていない別の命令バッファ５６－１へのプリフェッチルートに切り替えて命令プリフェッチを行わせる。

【００２０】

続いてコンテキスト切替部６０は、別のコンテキスト５２－２への切替えの契機となる後続命令の実行を判別した際に、第２ルータ５８のフェッチルートを、第１ルータ５４でプリフェッチルートを切り替えた側の命令バッファ５６－１に切り替えてプリフェッチされている別のコンテキスト５２－２の命令をパイプライン１５にフェッチさせる。

【００２１】

またコンテキストの切替えをハードウェアで行う場合、命令実行部１０は、実行途中にある別のコンテキスト切替えの契機となる後続命令を、この後続命令の実行に伴うハードウェアの信号から認識してコンテキストを切り替える。例えば後続命令がロード命令の場合、コンテキスト切替えを、ロード命令の実行におけるキャッシュのミスヒットを示す信号から認識する。また後続命令がロード命令の場合、コンテキスト切替えを、ロード命令の実行における分岐条件成立時（ｔａｋｅｎ）を示す分岐ターゲットバッファの出力から認識する。

【００２２】

またマルチリネーミング機能を実現するレジスタ・リネーミング部３５は、実行中のコンテキストＩＤを格納するＩＤレジスタ７６と、実行中のコンテキストの命令をフェッチする命令レジスタ７８と、有効フラグフィールド、レジスタキーフィールド及びデータフィールドを備えた複数の物理レジスタ８４－１～８４－ｋを有する名前替えレジスタファイル８２を備える。

【００２３】

レジスタ割付部は、ＩＤレジスタ７６のコンテキストＩＤであるＣＩＤｉに、命令レジスタ７８のレジスタ指定フィールド８０のレジスタ名Ｒｊと加えたレジスタキーＣＩＤｉ－Ｒｊ　を作成してレジスタファイル８２の該当する物理レジスタを名前替えレジスタとして割り付ける。

レジスタファイル８２からオーバフローした名前替えレジスタのデータは、退避処理部９２－１によって退避実行中のコンテキスト毎に設けられたレジスタ退避部９８－１，９８－２に退避される。またロード処理部９２－２は、レジスタ割付部によるレジスタファイル８２の参照で、レジスタキーＣＩＤｉ－Ｒｊ　に対応するレジスタが存在しなかった場合、レジスタ退避部９８－ｉから対応するレジスタキーのデータをレジスタファイル８２にロードさせる。

【００２４】

レジスタファイル８２は、命令レジスタ７８のレジスタ指定フィールド８０で指定可能な数の物理レジスタ８４－１～８４－ｍを有する。

レジスタファイル８２の各物理レジスタ８４－１～８４－ｍは、名前替えレジスタの割付けで有効フラグ８６をオンして使用状態を表し、レジスタデータの退避で有効フラグ８６をオフして空き状態を表す。

【００２５】

またレジスタ退避部９８－１，９８－２は、退避データを格納するデータフィールドに加

え、有効フラグフィールドを有する。有効フラグは、レジスタデータの退避によりオンしてデータ退避状態を表し、退避データのロードで有効フラグをオフしてレジスタファイル８２に存在することを表す。
また退避処理部９２－１は、レジスタファイル８２でオーバフローが発生した際に、実行中以外のコンテキストのいずれか１つのコンテキストＩＤをもつ全ての前記物理レジスタのデータを、対応するコンテキストのレジスタ退避部９８に退避させる。
【００２６】
ロード処理部９２－２は、レジスタ退避部９８から実行中のコンテキストのレジスタデータをレジスタファイル８２にロードするに先立ち、退避処理部９２－１によって、実行中以外のコンテキストのいずれか１つのコンテキストＩＤをもつ物理レジスタのデータを、対応するコンテキストのレジスタ退避部９８に退避させる。
【００２７】
またレジスタデータの退避とロードはコンテキスト単位ではなく、レジスタ単位に行ってもよい。実行対象となる複数のコンテキストは、異なったアドレス空間で実行される複数のプロセスである。また同一アドレス空間で実行される複数のスレッドであってもよい。スレッドは、数値計算プログラムのループである。また本発明はマルチコンテキストをサポートするマルチコンテキスト処理方法を提供する。この処理方法は、
ある意味のあるまとまった仕事をするための実行単位であるコンテキストを複数作成する作成過程；
パイプラインにあるコンテキストの命令を流して実行すると共に、パイプラインの空きを判断した場合は、実行途中にある別のコンテキストに切り替えて同時に複数のコンテキストを実行させる命令実行過程；
同時に実行している複数のコンテキストの各々に固有のコンテキストＩＤを設定するＩＤ設定過程；
前記コンテキストの実行で使用するレジスタ名を、前記コンテキストＩＤに実行命令の指定レジスタ名を加えたレジスタ名に名前替えして物理レジスタを割り当てるレジスタ・リネーミング過程と、
を有する。この処理方法の詳細も装置構成と基本的に同じになる。
【００２８】
【作用】
このような本発明のマルチコンテキストをサポートするプロセッサとその処理方法によれば、マルチリネーミング機能による命令間の依存関係を解消し、またロード命令の実行によるキャッシュのミスヒットや分岐命令が実行等を判断してパイプラインに流すコンテキストを切り替えることで、パイプラインストールを減らし、プロセッサの性能を向上する。
【００２９】
また柔軟なコンテキストの同時実行をスケジューリングすることで、スーパスカラのアーキテクチャをもつプロセッサの潜在性能をフルに引き出す。
またマルチリネーミングにおいて、同時に実行される複数のコンテキストが使うレジスタの数を、単一のコンテキストを走らせた場合と同じ数にでき、資源の利用効率を下げない。
【００３０】
更に、コンテキストを切り替える場合に、切替前のコンテキストで使用していた物理レジスタを、切替後のコンテキストで使用するレジスタに瞬時に切り替えず、必要なレジスタずつ徐々に切り替えることで、コンテキスト切替時のオーバヘッドを小さくする。
さらに、コンテキストは、ＤＯループのような小さな粒度のコンテキストばかりでなく、ユーザプロセスのような大きな粒度のコンテキストも扱えるようにする。
【００３１】
【実施例】
＜目　次＞

１．動作環境
２．コンテキストの切替え
３．マルチレジスタ・リネーミング
４．本発明の適用例
１．動作環境
図２は、本発明のマルチコンテキストをサポートするプロセッサの動作環境となるプロセッサマシンのハードウェアである。このプロセッサマシンは、ＣＰＵ１０からのバス１２に主記憶制御ユニット１４を介して主記憶ユニット１６を接続している。また、入出力バスアダプタ２２－１，２２－２に対し、外部の入出力バス２４－１，２４－２を接続している。

【００３２】
入出力バス２４－１，２４－２には、入出力制御装置２６－１，２６－２を介して入出力装置３０－１，３０－２が接続される。入出力装置３０－１，３０－２は、磁気ディスクユニット、光ディスクユニット、磁気テープユニットなどが使用される。また、システムバスアダプタ３２－１，３２－２により外部のシステムバス３４－１，３４－２に接続している。

【００３３】
システムバス３４－１，３４－２には、システム記憶ユニット３６－１，３６－２を接続している。更に、システムバス３４－１，３４－２に対し図示のプロセッサマシンと同じ複数のプロセッサマシンを接続することで、マルチプロセッサシステムが構築される。
プロセッサマシンのＣＰＵ１０には、浮動小数点演算制御ユニット３８、浮動小数点演算実行ユニット４０、整数演算制御ユニット４２、スーパスカラ整数演算実行ユニット４４、メモリ管理ユニット４６、命令キャッシュメモリ４８、データキャッシュメモリ５０およびバスインタフェースユニット５２が設けられる。

【００３４】
浮動小数点演算実行ユニット４０は、浮動小数点演算制御ユニット３８の制御のもとに浮動小数点演算を実行する。スーパスカラ整数演算実行ユニット４４はパイプラインを備え、整数演算制御ユニット４２の制御のもとに、動的に並列実行可能な整数演算命令を判別し、並列的に整数演算を実行する。即ち、本発明にあっては、整数演算制御ユニット４２とスーパスカラ整数演算実行ユニット４４によって複数のコンテキストを並列的に実行する処理が行われる。

【００３５】
命令キャッシュメモリ４８は、メモリ管理ユニット４６によるキャッシュ制御を受ける。命令キャッシュメモリ４８には、浮動小数点演算制御ユニット３８および整数演算制御ユニット４２で使用する主記憶制御ユニット１６からフェッチしたプログラムコードが格納される。この内、整数演算制御ユニット４２およびスーパスカラ整数演算実行ユニット４４でサポートされる本発明のマルチコンテキストの処理については、実行を開始した複数のマルチコンテキストの命令列が命令キャッシュメモリ４８に格納される。

【００３６】
データキャッシュメモリ５０は、浮動小数点演算実行ユニット４０およびスーパスカラ整数演算実行ユニット４４の演算に使用する主記憶制御ユニット１６から読み出したデータを格納する。データキャッシュメモリ５０のキャッシュ制御も、メモリ管理ユニット４６により行われる。
２．コンテキストの切替え
本発明のマルチコンテキストをサポートするプロセッサにおけるコンテキスト切替えの基本的なアーキテクチャは、図１３の従来例のように機械的に切り替えるのではなく、パイプラインにストールが発生する可能性のある命令を実行したときにコンテキストを切り替える。また１つのコンテキストが継続して実行可能である場合には、できるだけそのコンテキストを連続して実行する。これによって、コンテキストの切替えに伴う不要なオーバヘッドを低減し、キャッシュのヒット率の減少を回避する。

【００３７】
具体的には、現在実行中のコンテキストでロード命令または分岐命令を実行したときをコンテキストの切替タイミングとする。ロード命令の実行によりキャッシュメモリをアクセスした結果、ミスヒットした場合には、アクセス対象となるデータが主記憶メモリユニットから届くまでに数十サイクル分のレイテンシが存在する。
【００３８】
また分岐命令の実行の結果、分岐条件が成立して分岐した場合（ｔａｋｅｎ）、分岐先の命令をフェッチするためのパイプラインにストールが発生する可能性がある。したがって本発明にあっては、実行中のコンテキストでロード命令または分岐命令の実行を契機に、実行途中にある他のコンテキストへの切替えのタイミングとする。本発明によるコンテキスト切替えの実現方法としては、ハードウェアのみで行う方法とコンパイラとハードウェアの両方で行う方法の２種類がある。
【００３９】
まずコンテキストの切替えをハードウェアのみで行う場合を説明する。ハードウェアのみでロード命令の実行におけるキャッシュメモリのミスヒットの検出には、キャッシュタグの一致／不一致の信号をＣＰＵ側に入力して使うことができる。また分岐命令の実行に伴う分岐成立／分岐不成立（ｔａｋｅｎ／ｎｏｔ　ｔａｋｅｎ　）の判定には、分岐ターゲットバッファの出力を使うことができる。
【００４０】
このようなロード命令の実行に伴うキャッシュミスヒットの検出および分岐命令の実行による分岐条件成立時の検出は、図２のＣＰＵ１０に設けたスーパスカラのアーキテクチャをもつ整数演算制御ユニット４２およびスーパスカラ整数演算実行ユニット４４の機能として一般的に実現されており、検出結果を確保するレジスタの割当てなどの僅かな処理の変更で対応できる。
【００４１】
またハードウェアのみで実現するコンテキストの順番は、現在実行対象となっている複数のコンテキストについてラウンドロビン方式により予め定めた順番で切り替えればよい。
次に、コンパイラによるソフトウェアとハードウェアの両方で実現される本発明のコンテキスト切替えを説明する。
図３は、ソフトウェアとハードウェアの両方でコンテキスト切替えを行う場合の現在実行しているあるコンテキストの命令コードである。このコンテキストにあっては、命令１，命令２，・・・の順にパイプラインに流して実行される。ロード／分岐命令は、ここにロード命令または分岐命令が存在することを示している。
【００４２】
このプロクラムコードに特徴的なのは、通常の命令列を格納する命令フィールド６４に加え、属性情報フィールド６６をもっている点である。属性情報フィールド６６には属性情報が格納される。属性情報フィールド６６は、プログラムコードの意味は変えないがプログラムコードの高速実行を助けるための種々の属性情報を格納するためのフィールドと定義される。この属性情報フィールド６６は、コンパイラにおけるディレクティブ（ｄｉｒｅｃｔｉｖｅ）に相当する。
【００４３】
このコンテキストのプログラムコードにあっては、命令２の属性情報フィールド６６に属性情報「ＳＷ」を置いている。この属性情報「ＳＷ」は、「命令２の実行と同時に別のコンテキストの命令フェッチを開始せよ」という内容を指示する。その後、ロード／分岐命令の部分で、ロード命令の実行でキャッシュのミスヒットや分岐条件の成立で分岐した場合には、パイプラインにストールを発生する。
【００４４】
そこで、属性情報「ＳＷ」の指示によって既にフェッチしている別のコンテキストの命令列の実行に直ちに移る。この結果、ロード命令の実行によるキャッシュのミスヒットあるいは分岐命令の実行を契機として、実行中のコンテキストから実行途中にある他のコンテ

キストにオーバヘッドなしで切り替えることができる。

【００４５】

プログラムコードの属性情報フィールド６６に対する属性情報「ＳＷ」は、コンパイル後のプログラムコードを解析することにより埋め込むことができる。この作業は、コンパイルが済んだプログラムコードの中のロード命令あるいは分岐命令のオペコードを探し、ロード命令あるいは分岐命令に対しフェッチに必要な所定サイクル分のインターバルだけ先行した命令フィールドの属性情報フィールド６６に属性情報「ＳＷ」を置けばよい。

【００４６】

図４は、図３の構造をもつプログラムコードを用いた複数のコンテキストの切替えを実現する本発明の機能ブロックであり、図２のＣＰＵ１０に設けた整数演算制御ユニット４２およびスーパスカラ整数演算実行ユニット４４で実現される。まず、メモリ空間１６－１とＣＰＵ空間１０－２に分けられる。メモリ空間１６－２は、図２の命令キャッシュメモリ４８で実現され、もしキャッシュに命令が存在しなければ外部の主記憶ユニット１６に存在する。ＣＰＵ空間１０は、整数演算制御ユニット４２およびスーパスカラ整数演算実行ユニット４４で実現される空間である。

【００４７】

メモリ空間１６－１には、この実施例にあっては、４つのコンテキスト５２－１～５２－４が設けられている。コンテキスト５２－１～５２－４のそれぞれは、図３のようにコンパイルされたプログラムコードの命令ストリームである。コンテキスト５２－１～５２－４に対しては、コンテキストＩＤとして例えば＃０～＃３を設定するＩＤマッピングレジスタ６２－１～６２－４が設けられている。

【００４８】

コンテキスト５２－１～５２－４の命令ストリームは、コンテキストリード部６５－１～６５－４のそれぞれで読み出される。コンテキストリード部６５－１～６５－４の各々は、コンテキスト５２－１～５２－４の命令ストリームを読み出す際に、ＩＤマッピングレジスタ６２－１～６２－４で設定されたコンテキストＩＤ＃０～＃３を付けて読み出す。

【００４９】

ＣＰＵ空間１０－１には、第１ルータ５４、命令バッファ５６－１，５６－２、第２ルータ５８、命令実行部２５、マルチ・リネーミング部３５、コンテキスト切替部６０が設けられる。ルータ５４はコンテキスト５２－１～５２－４のいずれか１つを選択し、選択したコンテキストから読み出された命令ストリームを命令バッファ５６－１または５６－２の一方に格納する。ルータ５４からの命令ストリームを命令バッファ５６－１，５６－２のどちらに格納するかは、コンテキスト切替部６０が命令ストリーム切替信号Ｅ３によって制御する。

【００５０】

第２ルータ５８は、命令バッファ５６－１，５６－２のいずれかから命令を取り出して命令実行部２５に実行させる。ルータ５８による命令バッファ５６－１，５６－２からの命令取出しも、コンテキスト切替部６０からの命令ストリーム切替信号Ｅ４により制御される。

命令実行部２５には命令レジスタ７８が設けられ、第２ルータ５８で命令バッファ５６－１または５６－２から取り出された命令コードを格納し、パイプライン１５に供給する。パイプライン１５は４段構成であり、フェッチ段Ｆ、デコーダ段Ｄ、実行段ＥＸ、およびストア段Ｓを備える。更に命令実行部２５には、属性情報判別部１０４と命令実行判別部１０６が設けられる。

【００５１】

属性情報判別部１０４は、命令レジスタ７８に投入された命令について、図３の属性フィールド６６の属性情報を判別し、属性情報「ＳＷ」を判別すると、ＳＷ情報検出信号Ｅ１をコンテキスト切替部６０に出力する。コンテキスト切替部６０は、属性情報判別部１０４よりＳＷ情報検出信号Ｅ１を受けると、切替先となる別のコンテキストからの命令ストリームを、空き状態にある命令バッファ５６－１，５６－２のいずれかに格納するための

第１ルータ５４の切替えを行う。
【００５２】
命令実行判別部１０６は、属性情報判別部１０４で属性情報「ＳＷ」を判別した後のロード命令の実行によるキャッシュメモリのミスヒットあるいは分岐命令の実行による分岐条件の成立（ｔａｋｅｎ）を検出したとき、ロード命令または分岐命令の実行検出信号Ｅ２をコンテキスト切替部６０に出力する。
この検出信号Ｅ２を受けたコンテキスト切替部６０は、第２ルータ５８に命令ストリーム切替信号Ｅ４を出力し、属性情報「ＳＷ」の判別で、第１ルータ５４の切替えで切替先となる別のコンテキストからの命令をフェッチしている側の命令バッファ５６－１，５６－２に切り替え、パイプライン１５にストールを発生させないようにする。
【００５３】
ここで第１ルータ５４は、コンテキスト５２－１～５２－４からの命令ストリームを命令バッファ５６－１，５６－２のいずれかに予め格納するプリフェッチルートを切り替えるものである。これに対し第２ルータ５８は、命令バッファ５６－１，５６－２のいずれかから命令実行部２５の命令レジスタ７８に対する命令のフェッチルートを切り替える機能をもつといえる。
【００５４】
更に、命令実行部２５に対してはレジスタ・リネーミング部３５が設けられている。レジスタ・リネーミング部３５の詳細は後の説明で明らかにされる。
次に、図４のコンテキスト切替動作を具体的に説明する。いま図５のようにコンテキスト５２－１が実行中であったとする。この場合には、第１ルータ５４がコンテキストリード部６５－１を選択し、コンテキスト５２－１の命令ストリームにＩＤマッピングレジスタ６２－１によるコンテキストＩＤ＃０を付けた命令ストリームが、命令ルート６８によって例えば命令バッファ５６－２に供給される。
【００５５】
また第２ルータ５８は、コンテキスト５２－１からの命令ストリームが供給されている命令バッファ５６－２を選択しており、命令バッファ５６－２からコンテキスト５２－１のプログラムコードを取り出して命令レジスタ７８に投入し、パイプライン１５で実行している。
このようなコンテキスト５２－１の実行中に、図３に示した属性情報「ＳＷ」が属性情報フィールド６６に現われると、命令実行部２５の属性情報判別部１０４が属性情報「ＳＷ」を判別し、ＳＷ情報検出信号Ｅ１をコンテキスト切替部６０に出力する。ＳＷ情報検出信号Ｅ１を受けたコンテキスト切替部６０は、図６のように、第１ルータ５４に命令ストリーム切替信号Ｅ３を出力し、切替先となる別のコンテキスト５２－２のコンテキストリード部６５－２に切り替え、同時に命令バッファ５６－１に切り替える。
【００５６】
このため、実行対象となってはいるが現在停止しているコンテキスト５２－２の命令ルート７２による命令ストリームが、命令バッファ５６－１にプリフェッチされることになる。このとき第２ルータ５８は切り替えられず、現在実行中のコンテキスト５２－１の命令バッファ５６－２に格納された命令を命令ルート７０によって順番に取り出して、命令実行部２５で実行している。
【００５７】
この図６の切替状態で、命令実行部２５が命令バッファ５６－２に残っているコンテキスト５２－１の命令ストリームを実行する中でロード命令を実行すると、ロード命令の実行が命令実行判別部１０６で判別され、データキャッシュメモリでミスヒットとなったことを条件にロード命令実行検出信号Ｅ２をコンテキスト切替部６０に出力する。
【００５８】
ロード命令実行検出信号Ｅ２を受けたコンテキスト切替部６０は、第２ルータ５８に命令ルート切替信号Ｅ４を出力し、直ちに命令バッファ５６－１に命令ルートを切り替える。
この結果、ロード命令を実行した後の次のサイクルでは、コンテキスト５２－２の命令コ

ードがパイプライン１５に投入され、コンテキスト５２－１のロード命令の実行でキャッシュミスヒットとなっても、パイプライン１５にストールを発生することなく、別のコンテキスト５２－２の実行にオーバヘッドなしで切り替えることができる。

【００５９】

ここで図４の実施例にあっては、同時に実行可能なコンテキストの数を４としているが、この数は便宜的なものであり、実際にはコンテキストＩＤに使うビット長、同時に実行させるのに効率的なコンテキストの数に基づいて決められる。またコンテキストの切替えは、単純にラウンドロビンで行うことが効果的である。即ち、コンテキスト＃０，＃１，＃２，＃３の順番に循環的に切り替える。

【００６０】

図７のフローチャートは、図４の機能ブロックにおけるコンテキスト切替処理である。まずステップＳ１で、実行対象となるコンテキストの初期設定を行う。この初期設定は、ＣＰＵ１０の命令処理罫では不可能なので、ＯＳを呼び出して、実行対象となるコンテキストの割当てを行う初期設定を受ける。

次にステップＳ２で、実行対象となったコンテキストの命令ストリームを命令バッファに供給し、ステップＳ３で、命令バッファの命令を取り出してパイプラインに流すことで実行する。コンテキストの命令実行中にあっては、ステップＳ４で、命令の属性情報からコンテキスト切替要求が発生したか否かチェックしている。即ち、属性情報「ＳＷ」の判別をチェックしている。

【００６１】

属性情報「ＳＷ」を判別するとコンテキスト切替要求が発生したものと判断し、ステップＳ５で、現在実行途中のコンテキストが存在するか否かチェックする。実行途中のコンテキストが存在すれば、ステップＳ６で、命令ストリームの読出しを別の実行途中にあるコンテキストに切り替え、別の命令バッファに供給する。

【００６２】

続いてステップＳ７で、切替前の命令バッファに残っている現在実行中のコンテキストの命令を実行し、ステップＳ８で、分岐命令またはロード命令を実行したか否かチェックしている。ステップＳ８で分岐命令またはロード命令の実行を判別すると、ステップＳ９で、実行命令の取出バッファをステップＳ６で別のコンテキストに切り替えて、命令ストリームを格納している命令バッファに切り替え、これによってコンテキスト切替えが成立する。コンテキスト切替えが済むとステップＳ３に戻り、切替後の命令ストリームを対象とした命令の実行を繰り返し、以下、同様にして、コンテキスト切替要求が発生するごとに切替処理を行う。

【００６３】

一方、ステップＳ４でコンテキスト切替要求が発生した際に、ステップＳ５で実行途中のコンテキストが存在しなかった場合には、ステップＳ１０に進み、並列に実行可能なコンテキストを取り出し、ステップＳ１１で、取り出したコンテキストに番号を与え、コンテキストＩＤレジスタとコンテキスト配置テーブルに登録する。

【００６４】

このステップＳ１０，Ｓ１１の処理は、コンテキストの命令実行系では処理ができないので、ＯＳを呼び出して、新たなコンテキストの割当てとコンテキスト番号を付与を受けることになる。このＯＳによる支援を受けるステップＳ１０，Ｓ１１の処理は、オーバヘッドが比較的大きくなる処理であるが、頻繁に起こることはないため、その影響は小さいといえる。

３．マルチレジスタ・リネーミング

現在使用されている多くのスーパスカラ・プロセッサでは、命令間の依存を実行時に動的に解消するため、レジスタ・リネーミングというアルゴリズムを採用している。図８は、レジスタ・リネーミングの概略である。図８（Ａ）の変更前のプログラムの依存関係を見ると、１行目と２行目のレジスタＲ１は真の依存関係▲１▼、１行目と３行目のレジスタＲ１は出力依存▲２▼、１行目と４行目のレジスタＲは逆依存▲３▼の関係にある。

【００６５】
この図８（Ａ）のレジスタ・ネーミングの状態では、１行目と２行目のレジスタＲ１が出力依存▲２▼にあり、また１行目と４行目のレジスタＲ３が逆依存▲３▼の関係にあるため、１行目、３行目、４行目を同時に実行することはできない。そこで図８（Ｂ）のように、３行目のレジスタＲ１を新たなレジスタＲ７に変更し、また４行目のレジスタＲ３を新たなレジスタＲ８に変更するリネーミング（名前の付け替え）を行うことで、出力依存▲２▼と逆依存▲３▼を解消することができる。
【００６６】
このため、図８（Ｂ）のレジスタ・リネーミングにより１行目、３行目、４行目の命令を同時に実行することが可能となる。このように、レジスタ・リネーミングにより、出力依存と逆依存にある命令間の依存関係を解消して同時に実行することを可能とする。勿論、真の依存はリネーミングであっても解消できない。図８（Ａ）（Ｂ）のリネーミングは、単一の命令ストリームの中での命令の依存を解消する手法であるが、本発明にあっては、このレジスタ・リネーミングの手法を複数のコンテキストの同時実行を対象とした複数の命令ストリームについての命令依存の対象に拡張する。即ち、複数のコンテキストの実行による複数の命令ストリームでのレジスタ・リネーミングを実現するため、次の要件を定める。
【００６７】
▲１▼レジスタのぶつかりを避けるために、レジスタの名前替えの際に、命令中のレジスタ番号に加えコンテキストのＩＤを利用する。
▲２▼１つのコンテキストが使うレジスタ数を増やすために、レジスタ名前替え用のレジスタファイルを
（命令中のレジスタ番号）＋（コンテキストＩＤ）
のキーで検索できるようにアソシエイティブなメモリで構成する。このようなメモリを以下、リネーミングバッファと呼ぶ。また、リネーミングバッファで実現する複数の物理レジスタをコンテキストの数で等分して使用するようなことはせず、物理レジスタに対し仮想的なレジスタ割当てを行う。
【００６８】
▲３▼コンテキスト切替えの際に、リネーミングバッファから古くなったコンテキストのレジスタを退避するようなことはしない。コンテキストの切替えに関係なく、リネーミングバッファの中の利用できる物理レジスタがなくなった時点で古いコンテキストに割り当てているレジスタを退避する。即ち、リネーミングバッファからのレジスタの退避をルーズに行う。
【００６９】
このような▲１▼～▲３▼の内容をもったレジスタの名前替えをマルチレジスタ・リネーミングと呼ぶ。このようなマルチレジスタ・リネーミングの構築によって、複数のコンテキストを同時にパイプラインに流すことが可能となる。即ち、あるコンテキスト＃０でレジスタＲ１を使っていたとする。この実行中のコンテキスト＃０と同時に実行させたいと考える別のコンテキスト＃１も同じレジスタＲ１を使っていたとする。
【００７０】
この場合、本発明のマルチレジスタ・リネーミングによれば、２つのコンテキスト＃０，＃１で使用しているレジスタＲ１には、コンテキストＩＤの付加により「００－Ｒ１」「１０－Ｒ１」にリネーミングされる。このため、コードを変更したりハードウェアで同期をとったりする必要がなく、同じレジスタ名をもった複数のコンテキストを同時に実行することができる。また、実行中でないコンテキストのリネーミングレジスタの情報をメモリ空間の別の領域に退避させていることで、複数のコンテキストのレジスタ情報を同時にハードウェア上の物理レジスタに依存させることができる。
【００７１】
図９は、本発明のマルチレジスタ・リネーミングを実現するための機能ブロックであり、図４のレジスタ・リネーミング部３５の詳細となる。レジスタ・リネーミング部３５は、

ＣＰＵ空間１０－１とメモリ空間１６－１を使用して構築される。ＣＰＵ空間１０－１にはコンテキスト・マッピング・テーブル６２が設けられる。コンテキスト・マッピング・テーブル６２には、全く新しいコンテキストの実行を開始する場合に、コンテキスト番号ＣＮの登録がＯＳの支援により行われる。

【００７２】

この実施例は、同時に走行するコンテキストを４つとした場合を例にとっていることから、コンテキスト・マッピング・テーブル６２にはコンテキストＩＤ００，０１，１０，１１が割り当てられ、このコンテキストＩＤに対し、実行対象となったコンテキスト番号ＣＮ０として＃０をコンテキストＩＤ００に登録し、コンテキスト番号＃１をコンテキストＩＤ１０に登録している。

【００７３】

更にＣＰＵ空間１０－１には、実行中コンテキストＩＤレジスタ７６、命令レジスタ７８、リネーミング・バッファ８２、退避／ロード処理部９２が設けられる。実行中コンテキストＩＤレジスタ７６には、現在実行中のコンテキスト、例えばコンテキスト番号＃１のコンテキストＩＤ＝１０が格納される。命令レジスタ７８にはリネーミング対象となるレジスタ名をレジスタ指定フィールド８０に格納した命令コードが格納される。

【００７４】

リネーミング・バッファ８２は、命令のレジスタ指定フィールド８０で指定可能な物理レジスタ部８４－１～８４－１６のレジスタ領域に分けられている。この実施例は、物理レジスタの指定数を１６としているが、３２，６４など、適宜の数に定めることができる。リネーミング・バッファ８２の物理レジスタ部８４－１～８４－１６のそれぞれは、有効フラグフィールド８６、キーフィールド８８およびデータフィールド９０を有する。有効フラグフィールド８６は、その物理レジスタがリネーミング・レジスタとして使用されている場合に有効フラグをオンしており、未使用である場合には有効フラグがオフとなっている。したがって、物理レジスタ部８４－１～８４－１６の有効フラグ８６を調べることで、リネーミングに使用可能な物理レジスタが存在するか否か判断できる。

【００７５】

キーフィールド８８には、実行中コンテキストＩＤレジスタ７６に格納したＣＩＤｉと命令レジスタ７８のレジスタ指定フィールド８０のレジスタ名Ｒｊを合わせたキー（ＣＩＤｉ－Ｒｊ）がリネーミング・レジスタ名を示すキーコードとして格納される。このため同じレジスタ名Ｒｊであっても、コンテキストＩＤであるＣＩＤｉによって区別することができる。

【００７６】

一方、メモリ空間１６－１にはコンテキスト・マッピング・テーブル６２に登録されたコンテキスト番号＃０，＃１ごとにコンテキスト・コントロール・ブロック９４－１，９４－２が確保されている。このコンテキスト・コントロール・ブロック９４－１，９４－２は、特定領域をレジスタ退避領域９８－１，９８－２に割り当てている。レジスタ退避領域９８－１，９８－２は、基本的にはリネーミング・バッファ８２の物理レジスタ部８４－１～８４－１６と同じ数の領域を確保している。レジスタ退避領域９８－１の各レジスタ領域は、有効フラグフィールド１００とレジスタデータフィールド１０２に分けられている。有効フラグフィールド１００の有効フラグは、レジスタデータフィールド１０２にレジスタデータが退避されるとオンし、レジスタデータをリネーミング・バッファ８２にロードするとオフとなる。

【００７７】

コンテキスト番号＃０のレジスタ退避領域９８－１の場合、先頭の有効フラグフィールド１００のみが０で他の有効フラグは１となっており、先頭のレジスタデータのみがリネーミング・バッファ８２にあることを表わしている。これに対し、現在実行中のコンテキスト番号＃１のレジスタ退避領域９８－２については、全ての有効フラグが０とオフしており、リネーミング・レジスタは全てリネーミング・バッファ８２に存在することを表わしている。

【００７８】
なお、レジスタ退避領域９８－１，９８－２以外のコンテキスト・コントロール・ブロック９４－１，９４－２の領域は、通常のコントロールエリア１１０－１，１１０－２として使用されている。
退避／ロード処理部９２は、リネーミング・バッファ８２がオーバフローした際のメモリ空間１６－１に対するレジスタ退避処理と、実行中のコンテキストで使用するリネーミング・レジスタがリネーミング・バッファ８２に存在しないときのメモリ空間１６－１からのレジスタデータのロードを行う。
【００７９】
図１０のフローチャートは、図９におけるマルチ・リネーミング処理である。まずステップＳ１で、命令レジスタ７８にリネーミング対象となる命令が投入されたか否かチェックしている。リネーミング対象となる命令が投入されると、ステップＳ２で、実行中コンテキストＩＤレジスタ７６に格納している実行中のコンテキストＩＤｉ＝１０と、命令レジスタ７８のレジスタ指定フィールド８０のレジスタ番号例えばＲ１でキー「１０－Ｒ１」を作成し、リネーミング・バッファ８６のキーフィールド８８を検索する。
【００８０】
キーフィールド８８の検索で該当するレジスタがあれば、ステップＳ５に進み、リネーミング・レジスタＣＩＤｉ－Ｒｊ＝１０－Ｒ１を使用した命令の実行を行う。一方、ステップＳ３で、リネーミング・バッファ８２に該当レジスタがなかった場合には、ステップＳ４に進み、メモリ空間１６－１のコンテキスト番号＃１のレジスタ退避領域９８－２から対応するレジスタデータのロード処理を行う。このレジスタロード処理の詳細は後の説明で明らかにされる。
【００８１】
図１１のフローチャートは、図９のリネーミング・バッファ８２がオーバフローした際のレジスタ退避処理である。あるコンテキストを実行中にリネーミング・バッファ８２がオーバフローすると、そのコンテキストの実行を続けるためには別のコンテキストのリネーミング・バッファ８２上に存在するレジスタデータをメモリ空間１６－１の退避領域に退避させなければならない。
【００８２】
この場合のリネーミング・バッファ８２のオーバフローは、物理レジスタ部８４－１～８４－１６の有効フラグが全てオン（ビット１）で且つ実行中コンテキストレジスタ７６のＣＩＤｉと命令レジスタ７８のレジスタ指定フィールド８０のレジスタ名Ｒｊで作ったキー「ＣＩＤｉ－Ｒｊ」によるキーフィールド８８の検索で該当するレジスタが存在しなかった場合である。
【００８３】
リネーミング・バッファ８２のオーバフロー発生に伴うレジスタ退避処理にあっては、まずステップＳ１で、リネーミング・バッファ８２より退避するコンテキストを決定する。退避するコンテキストの決定は、コンテキスト・マッピング・テーブル６２にＬＲＵアルゴリズムの機能を付け加えて決めればよい。即ち、複数のコンテキストの中の最も実行した時点の古いコンテキストを、退避するコンテキストに決定する。
【００８４】
続いてステップＳ２で、退避するコンテキストのＩＤを使用してリネーミング・バッファ８２における各物理レジスタ８４－１～８４－１６のキーフィールドを検索し、該当するレジスタをアクセスする。続いて、アクセスした物理レジスタの有効フラグがオンか否かチェックする。
有効フラグがオンであれば、ステップＳ４に進み、退避コンテキストの番号をラベルしたメモリ空間１６－１のコンテキスト・コントロール・ブロック内の対応するレジスタ番号の領域に退避する。退避が済むと、ステップＳ５で、レジスタ退避領域の有効フラグをオンする。次にステップＳ６で、リネーミング・バッファ８２の退避済みのレジスタの有効フラグをオフする。

【００８５】
続いてステップＳ７で、命令レジスタ７８のレジスタ指定フィールド８０で指定可能な例えば１６個のレジスタ分の処理を行ったか否かチェックする。全てのレジスタの処理を終了するまで、ステップＳ２～Ｓ６の処理を繰り返す。勿論、ステップＳ３で、退避コンテキストＩＤをもつレジスタを検索しても、有効フラグがオフであった場合には、ステップＳ４～Ｓ６の処理は不要である。
【００８６】
このオーバフロー時のレジスタ退避処理によって、リネーミング・バッファ８２に残っている最も実行時点の古いコンテキストで使用したレジスタ内容が、対応するメモリ空間１６－１のレジスタ退避領域に退避され、実行中のコンテキストのリネーミングのためにリネーミング・バッファ８２に空きレジスタを確保することができる。
【００８７】
図１２のフローチャートは、図９の退避／ロード処理部９２によるレジスタデータ・ロード処理である。図１１のレジスタ退避処理によって、実行途中にあるコンテキストのレジスタをメモリ空間１６－１のコンテキスト・コントロール・ブロックに退避してしまうと、その後にそのコンテキストの実行がスケジュールされたとき、レジスタのデータがリネーミング・バッファ８２上にないことがある。このような場合に、図１２のレジスタデータ・ロード処理が行われる。
【００８８】
まずステップＳ１で、レジスタデータのロード処理に先立ち、現在リネーミング・バッファ８２に格納されている別のコンテキストのレジスタ群のレジスタデータを退避する。これは、新たなレジスタデータをロードするときにオーバフローが起きないことを保証するためである。このレジスタデータの退避処理は、図１１のレジスタ退避処理を行うことになる。勿論、レジスタロード処理に先立ってリネーミング・バッファ８２の有効フラグから十分な空きレジスタが存在する場合には、ステップＳ１の他のコンテキストのレジスタデータの退避処理は行う必要がない。
【００８９】
次にステップＳ２で、実行中のコンテキストに関するメモリ空間１６－１のコンテキスト・コントロール・ブロックの対応するレジスタ退避領域の内容を読み出す。次にステップＳ３で、読み出したレジスタ退避領域の有効フラグがオンか否かチェックする。オンであればレジスタデータは退避領域に存在することから、ステップＳ４で、ロードするコンテキストのＩＤを使用してリネーミング・バッファ８２にレジスタ退避領域の読出内容をロードする。
【００９０】
レジスタロードが済むと、ステップＳ５で、コンテキスト・コントロール・ブロックのロードが済んだレジスタ領域の有効フラグをオフする。続いてステップＳ６で、リネーミング・バッファ８２のロード済みのレジスタの有効フラグをオンする。一方、ステップＳ３で、レジスタ退避領域の有効フラグがオフであれば、そのレジスタはリネーミング・バッファ８２上に確保されていることから、ステップＳ４～Ｓ６のロード処理は行わない。
【００９１】
１つの退避データのロードが済むと、ステップＳ７で、命令レジスタ７８のレジスタ指定フィールド８０で指定できる退避領域のレジスタ分の処理の終了の有無をチェックし、処理を終了するまで、ステップＳ２～Ｓ７の処理を繰り返す。
以上のマルチ・リネーミング・レジスタの処理にあっては、一度、レジスタの退避またはロードが起きると対象となったコンテキストに属する全てのレジスタデータの退避またはロードを行う場合を例にとっているが、１つずつ処理してもよい。オーバヘッドの観点からは、レジスタ退避については退避対象となったコンテキストに含まれる全てのレジスタデータを退避させ、ロードは必要に応じて１つずつ行うことが最も効率的と考えられる。
【００９２】
またリネーミング・バッファ８２内のレジスタデータの置替えは、キャッシュメモリにお

けるＬＲＵアルゴリズムに類似している。このため基本的なレジスタデータの置替えアルゴリズムは、キャッシュメモリにおけるＬＲＵアルゴリズムを適用することができる。但し、キャッシュメモリのＬＲＵアルゴリズムを適用する場合には、このための処理専用のレジスタを数個、リネーミング・バッファ８２以外に用意する必要がある。

４．本発明の適用例

（１）小さなコンテキストの場合

小さなコンテキストとは、所謂同じアドレス空間内で複数のコンテキストを実行する場合である。この場合には、従来のマルチ・スレッドの実行と同様のコンテキストの切替えが可能である。同じアドレス空間内で複数のコンテキストに分けるのは、基本的にはコンパイラの仕事になる。これに加え本発明にあっては、コンテキストの切替タイミングについて図３～図７に示したロード命令または分岐命令の実行を契機として行うことで、より柔軟なコンテキスト切替えが実現できる。

【００９３】

また、同じアドレス空間内で複数のコンテキストを実行する際のマルチ・レジスタ・リネーミングについては、図９～図１２に示したとおりであるが、ハードウェアに関しては、図９のＣＰＵ空間１０－１に設けたコンテキスト・マッピング・テーブル６２とメモリ空間１６－１のコンテキストごとのコンテキスト・コントロール・ブロックは省略できる。メモリ空間１６－１については、コンテキスト・コントロール・ブロックを設ける代わりにリネーミング・バッファ８２がオーバフローしたときのレジスタ退避領域を各コンテキストごとに設ければよい。

（２）大きなコンテキストの場合

大きなコンテキストとは、アドレス空間を共用しない複数のコンテキストを実行する場合である。この例の１つには、ユニックスのプロセスのような実行がある。また、並列計算機のプロセッサ・エレメントとして本発明のプロセッサを適用する場合には、他のプロセッサ・エレメントからきたメッセージの処理や、こちらから他のプロセッサ・エレメントに出すメッセージの構築などを、プロセッサ内の命令の実行と平行して行うにことにより、メッセージ送出と受取りのためのオーバヘッドを軽減することができる。

【００９４】

この機能は、本発明のプロセッサをメッセージベースの通信を主とする並列計算機や分散計算機のプロセッサ・エレメントとして適用する場合、極めて有効な特徴である。また、この場合のマルチ・レジスタ・リネーミングのレジスタ退避に使用するコンテキスト・コントロール・ブロックは、通常のプロセッサ・コントロール・ブロックをそのまま利用できる。

（３）マルチ・メディア処理

近年、画像や音声などのデータ・ストリームを扱う所謂マルチ・メディア処理が注目で浴びている。マルチ・メディア処理は、大量のデータ・ストリームをしかもリアルタイムに近い時間で処理しなければならない。これに大量のデータ・ストリームをリアルタイムで処理するためには、デジタル・シグナル・プロセッサやある種の処理を行う専用プロセッサをコ・プロセッサとしてメインのプロセッサに設けるというアーキテクチャが考えられる。

【００９５】

このようにメインプロセッサに専用のコ・プロセッサを設けた場合、メインプロセッサとコ・プロセッサ（Ｃｏ　Ｐｒｏｃｅｓｓｅｒ）の間の制御のやり取りが実行スピードを決め、またユーザに対するレスポンスにも影響する。特に、ユーザに対するレスポンスの高速化を図るため、本発明のプロセッサのアーキテクチャをメインプロセッサとコ・プロセッサによる並列処理に適用することは、非常に有効である。

【００９６】

即ち、コ・プロセッサを制御するプログラムを実行するときに、その都度、メインプロセッサで実行中のコンテキストの実行環境を全て退避したり、再開に当たっての復旧したりする手間を必要としなくなるからである。

なお本発明は上記の実施例で用いた数値による限定は受けない。またスーパスカラ・プロセッサであれば、適宜のアーキテクチャをもつプロセッサに適用できる。

【００９７】

【発明の効果】

以上説明してきたように本発明によれば、複数のコンテキストをパイプラインに流して実行する場合のパイプラインのストール（バブル）を低減し、全体としてのスループットを高めることができる。

また、パイプラインのストールを回避するためのコンテキストの切替えに要する時間を大幅に短縮することができる。

【００９８】

また、複数のコンテキストを実行していても、必要とする物理レジスタは１つのコンテキストに必要なレジスタ数で済み、ハードウェア資源を増やすことなく、効率良く複数のコンテキストを実行することができる。

また、コンテキストを切り替えるタイミングをプログラムから指定できるため、より柔軟なコンテキストのスケジューリングができる。

【００９９】

更に、コンテキストとして所謂スレッドからプロセスのようなものまで、種々の大きさのコンテキストに対し適用可能である。

更にまた、本発明のプロセッサの実現に当たって拡張が必要な部分は、現在使用されているスーパスカラ・アーキテクチャの自然な拡張となっており、物理的にも論理的にもそれほど複雑な拡張を必要とすることなく、本発明によるアーキテクチャを実現することができる。

【図面の簡単な説明】

【図１】本発明の原理説明図

【図２】本発明の動作環境となるハードウェアのブロック図

【図３】コンテキスト切替機能のブロック図

【図４】コンテキストを構成する命令ストリームの構造説明図

【図５】ＣＩＤ０のコンテキスト実行時の命令ルートの説明図

【図６】属性情報からコンテキスト切替要求が発生した場合のルート切替えの説明図

【図７】本発明のコンテキスト切替処理のフローチャート

【図８】本発明のマルチ・リネーミング機能のブロック図

【図９】命令依存とレジスタ・リネーミングによる依存解消の説明図

【図１０】図８のリネーミング処理のフローチャート

【図１１】図８のレジスタ退避処理のフローチャート

【図１２】図８のレジスタデータロード処理のフローチャート

【図１３】従来のコンテキスト切替えの説明図

【符号の説明】

１０：ＣＰＵ（命令実行部）

１２：ＣＰＵバス

１４：主記憶制御ユニット

１６：主記憶ユニット

１８：２次キャッシャコントローラ

２０：２次キャッシュメモリ

２２－１，２２－２：入出力バスアダプタ

２４－１，２４－２：入出力バス

２６－１，２６－２：入出力制御装置

３０－１，３０－２：入出力装置

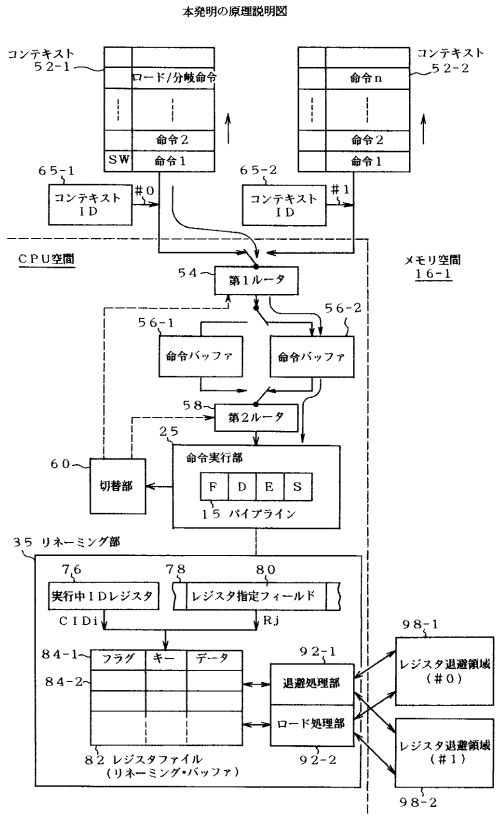
３２－１，３２－２：システムバスアダプタ

３４－１，３４－２：システムバス

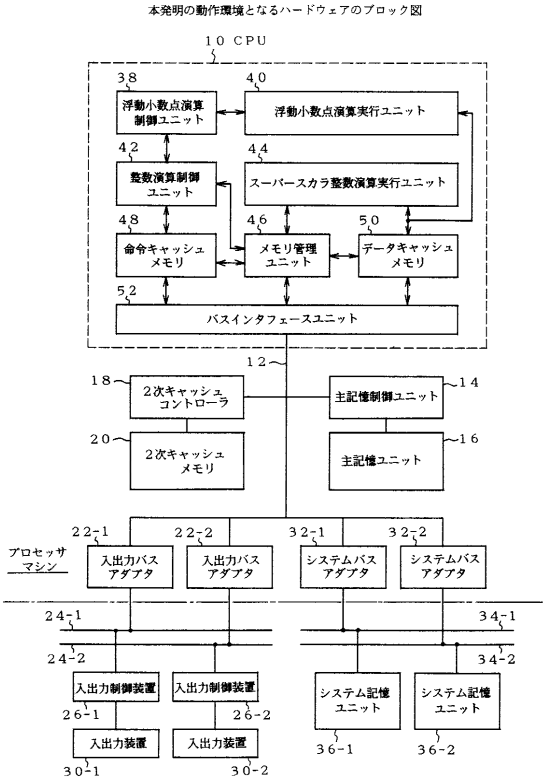
３６－１，３６－２：システム記憶ユニット

３８：浮動小数点演算制御ユニット
４０：浮動小数点演算実行ユニット
４２：整数演算制御ユニット
４４：スーパスカラ整数演算ユニット
４６：メモリ管理ユニット
４８：命令キャッシュメモリ
５０：データキャッシュメモリ
５２－１～５２－４：コンテキスト
５４：第１ルータ
５６－１，５６－２：命令バッファ
５８：第２ルータ
６０：コンテキスト切替ユニット
６２－１～６２－４：コンテキストＩＤレジスタ
６４：命令フィールド
６６：属性情報フィールド
６８，７２：プリフェッチルート
７０：フェッチルート
７４：コンテキスト・マッピング・テーブル
７６：実行中コンテキストＩＤレジスタ
７８：命令レジスタ
８０：レジスタ指定フィールド
８２：リネーミング・レジスタ・ファイル
８４－１～８４－１２：物理レジスタ
８６：有効フラグフィールド
８８：キーフィールド
９０：データフィールド
９２：退避ロード処理部
９４－１，９４－２：コンテキスト・コントロール・ブロック
９６－１，９６－２：ラベル
９８－１，９８－２：レジスタ退避領域
１００：有効フラグフィールド
１０２：データフィールド

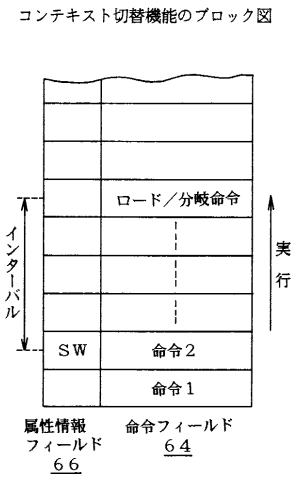
【図１】


本発明の原理説明図
コンテキスト 52-1
ロード/分岐命令
命令2
SW
命令1
コンテキスト 52-2
命令n
命令2
命令1
65-1
コンテキスト ID
#0
65-2
コンテキスト ID
#1
CPU空間
メモリ空間 16-1
54 第1ルータ
56-1
命令バッファ
56-2
命令バッファ
58 第2ルータ
25
命令実行部
F D E S
15 パイプライン
60 切替部
35 リネーミング部
76
実行中IDレジスタ
78
80
レジスタ指定フィールド
CIDi
Rj
84-1
84-2
フラグ
キー
データ
82 レジスタファイル (リネーミング・バッファ)
92-1 退避処理部
92-2 ロード処理部
98-1 レジスタ退避領域 (#0)
98-2 レジスタ退避領域 (#1)

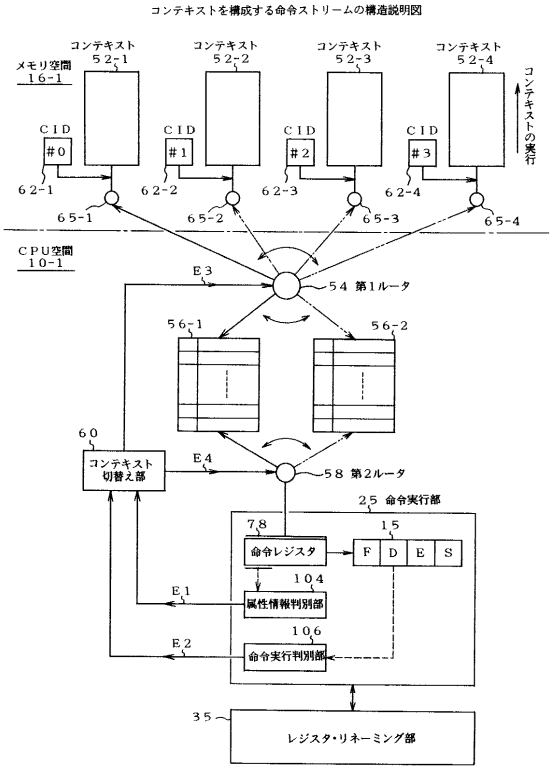
【図２】


本発明の動作環境となるハードウェアのブロック図
10 CPU
38 浮動小数点演算制御ユニット
40 浮動小数点演算実行ユニット
42 整数演算制御ユニット
44 スーパースカラ整数演算実行ユニット
48 命令キャッシュメモリ
46 メモリ管理ユニット
50 データキャッシュメモリ
52 バスインタフェースユニット
12
18 2次キャッシュコントローラ
14 主記憶制御ユニット
20 2次キャッシュメモリ
16 主記憶ユニット
プロセッサマシン
22-1 入出力バスアダプタ
22-2 入出力バスアダプタ
32-1 システムバスアダプタ
32-2 システムバスアダプタ
24-1
24-2
34-1
34-2
26-1 入出力制御装置
26-2 入出力制御装置
36-1 システム記憶ユニット
36-2 システム記憶ユニット
30-1 入出力装置
30-2 入出力装置

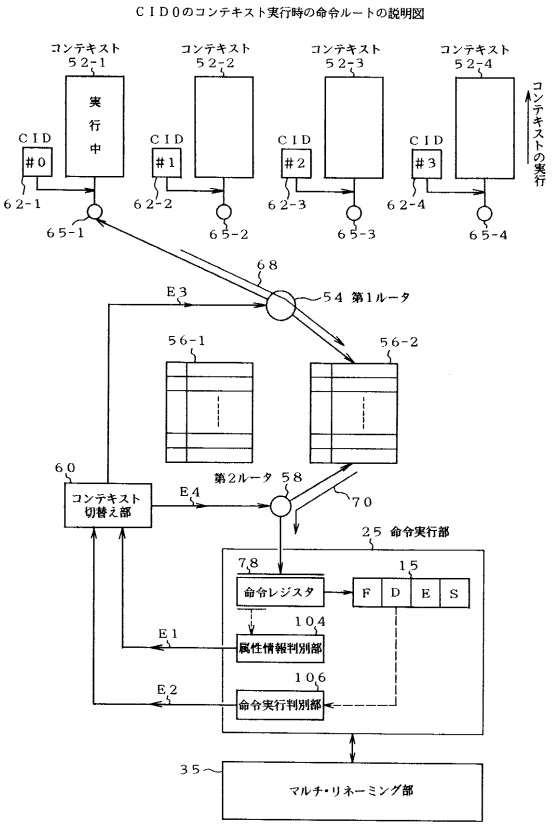
【図３】


コンテキスト切替機能のブロック図
インターバル
実行
ロード/分岐命令
SW
命令2
命令1
属性情報フィールド 66
命令フィールド 64

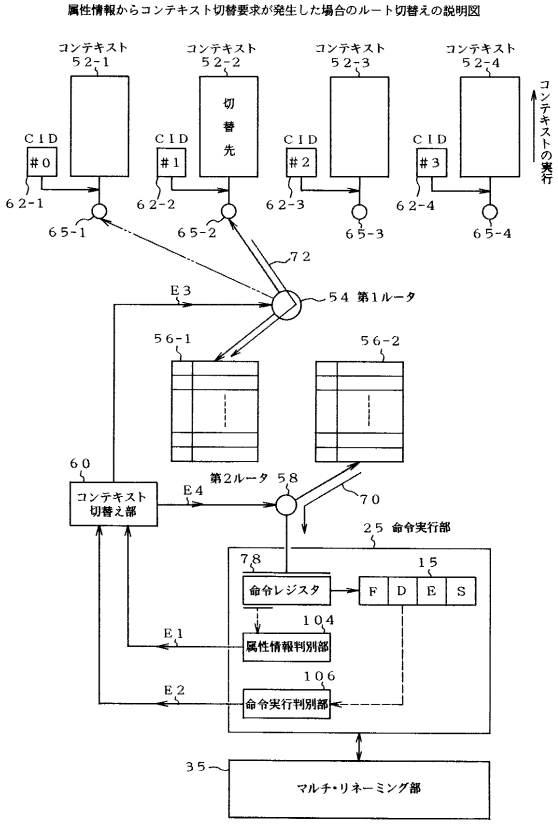
【図４】


コンテキストを構成する命令ストリームの構造説明図
メモリ空間 16-1
コンテキスト 52-1
コンテキスト 52-2
コンテキスト 52-3
コンテキスト 52-4
コンテキストの実行
CID #0
CID #1
CID #2
CID #3
62-1
62-2
62-3
62-4
65-1
65-2
65-3
65-4
CPU空間 10-1
E3
54 第1ルータ
56-1
56-2
60 コンテキスト切替え部
E4
58 第2ルータ
25 命令実行部
78
命令レジスタ
15
F D E S
104
E1
属性情報判別部
106
E2
命令実行判別部
35 レジスタ・リネーミング部

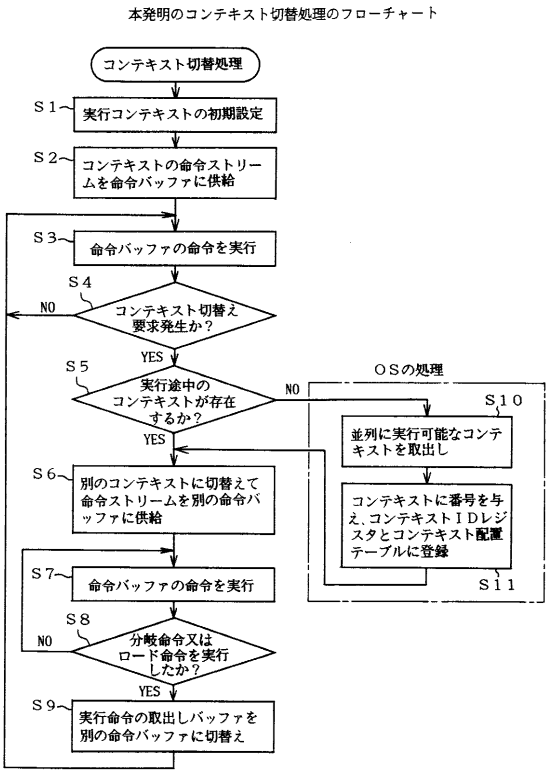
【図５】


ＣＩＤ０のコンテキスト実行時の命令ルートの説明図
コンテキスト 52-1
コンテキスト 52-2
コンテキスト 52-3
コンテキスト 52-4
実行中
ＣＩＤ
#0
#1
#2
#3
62-1
62-2
62-3
62-4
65-1
65-2
65-3
65-4
コンテキストの実行
68
E3
54 第1ルータ
56-1
56-2
60
コンテキスト切替え部
E4
第2ルータ 58
70
25 命令実行部
78
命令レジスタ
15
F D E S
104
属性情報判別部
E1
106
命令実行判別部
E2
35
マルチ・リネーミング部

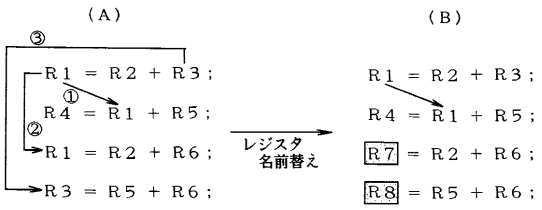
【図６】


属性情報からコンテキスト切替要求が発生した場合のルート切替えの説明図
コンテキスト 52-1
コンテキスト 52-2
コンテキスト 52-3
コンテキスト 52-4
切替先
ＣＩＤ
#0
#1
#2
#3
62-1
62-2
62-3
62-4
65-1
65-2
65-3
65-4
コンテキストの実行
72
E3
54 第1ルータ
56-1
56-2
60
コンテキスト切替え部
E4
第2ルータ 58
70
25 命令実行部
78
命令レジスタ
15
F D E S
104
属性情報判別部
E1
106
命令実行判別部
E2
35
マルチ・リネーミング部

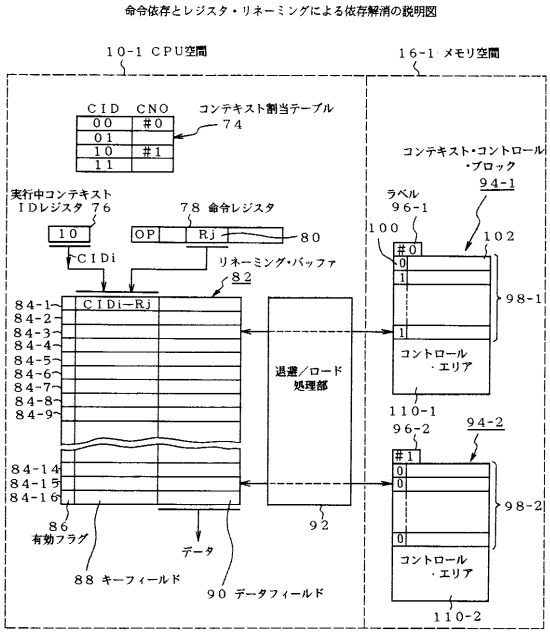
【図７】


本発明のコンテキスト切替処理のフローチャート
コンテキスト切替処理
S1
実行コンテキストの初期設定
S2
コンテキストの命令ストリームを命令バッファに供給
S3
命令バッファの命令を実行
S4
NO
コンテキスト切替え要求発生か？
YES
S5
実行途中のコンテキストが存在するか？
NO
YES
OSの処理
S10
並列に実行可能なコンテキストを取出し
コンテキストに番号を与え、コンテキストＩＤレジスタとコンテキスト配置テーブルに登録
S11
S6
別のコンテキストに切替えて命令ストリームを別の命令バッファに供給
S7
命令バッファの命令を実行
S8
NO
分岐命令又はロード命令を実行したか？
YES
S9
実行命令の取出しバッファを別の命令バッファに切替え

【図８】

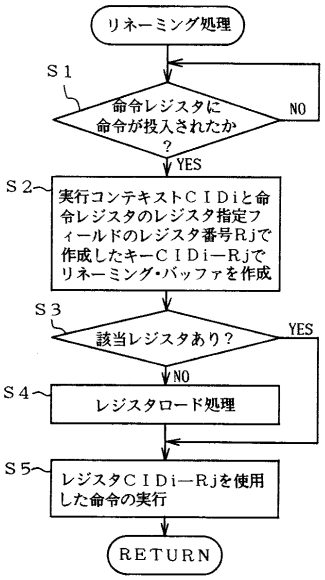

本発明のマルチ・リネーミング機能のブロック図
(A)
(B)
③
R1 = R2 + R3;
①
R4 = R1 + R5;
②
R1 = R2 + R6;
R3 = R5 + R6;
レジスタ名前替え
R1 = R2 + R3;
R4 = R1 + R5;
R7 = R2 + R6;
R8 = R5 + R6;

【図９】



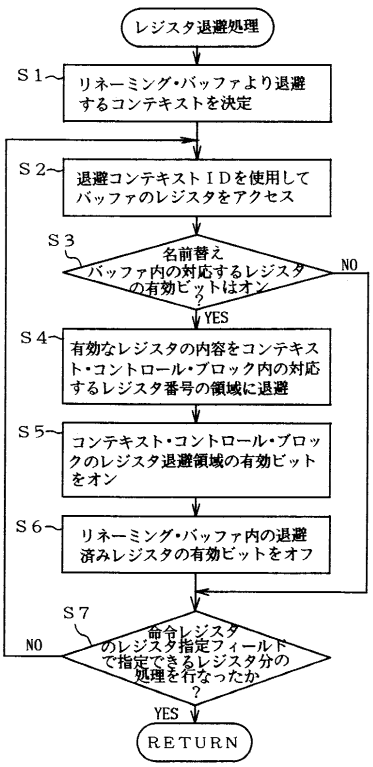
【図１０】

図8のリネーミング処理のフローチャート



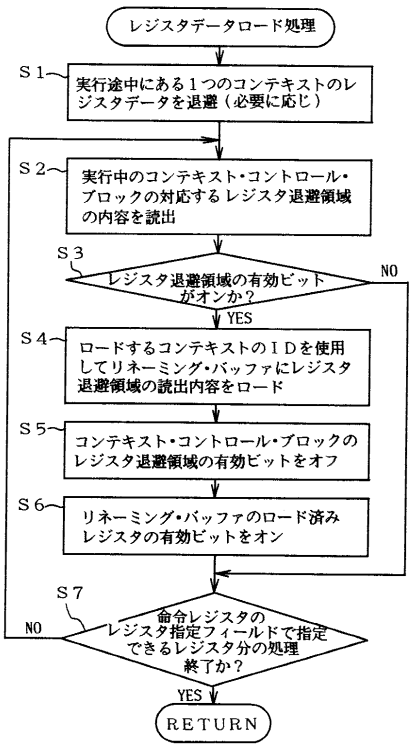
【図１１】

図8のレジスタ退避処理のフローチャート



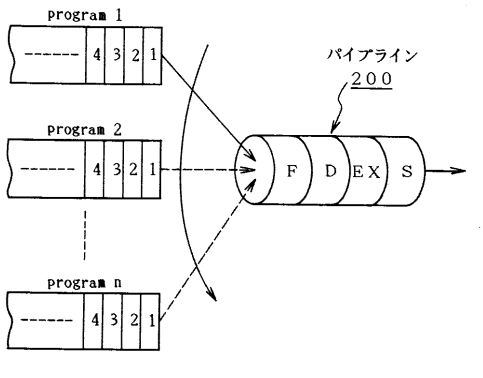
【図１２】

図8のレジスタデータロード処理のフローチャート



【図１３】

従来のコンテキスト切替えの説明図



フロントページの続き

(56)参考文献 特開平２－２２６４５８（ＪＰ，Ａ）
特開平３－１９６２２０（ＪＰ，Ａ）
特開平６－４４０８９（ＪＰ，Ａ）
特開平６－２８１８７（ＪＰ，Ａ）
特開昭５８－１４２５３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 9/38
G06F 9/46