(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3714598号
(P3714598)
(45)発行日　平成17年11月9日(2005.11.9)　(24)登録日　平成17年9月2日(2005.9.2)

(51)Int.Cl.[7]	F I		
G06F 9/46	G06F	9/46	410
G06F 9/48	G06F	9/46	452B

請求項の数 9　(全 26 頁)

(21)出願番号	特願2000-517342 (P2000-517342)
(86)(22)出願日	平成10年10月14日 (1998.10.14)
(65)公表番号	特表2001-521219 (P2001-521219A)
(43)公表日	平成13年11月6日 (2001.11.6)
(86)国際出願番号	PCT/US1998/021724
(87)国際公開番号	WO1999/021089
(87)国際公開日	平成11年4月29日 (1999.4.29)
審査請求日	平成12年5月19日 (2000.5.19)
(31)優先権主張番号	08/958,718
(32)優先日	平成9年10月23日 (1997.10.23)
(33)優先権主張国	米国 (US)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100086243
弁理士　坂口　博
(74)代理人　100091568
弁理士　市位　嘉宏

最終頁に続く

(54)【発明の名称】マルチスレッド式プロセッサでのスレッド優先順位の変更

(57)【特許請求の範囲】

【請求項１】

マルチスレッド式プロセッサにおいて、複数のスレッド各々の優先順位に基いてスレッドを切替える方法において、

該プロセッサが、該プロセッサの命令ユニットに接続されたスレッド切替え論理を含み、

該切替え論理は、各スレッドの状態が格納されたレジスタであって、ハードウェアもしくはソフトウェアによって設定することができる、各スレッドの優先順位を表すビット（優先順位状態ビット）を含む、スレッド状態レジスタと、

スレッド切替えを起こすイベントを選択する、ソフトウェア・プログラム可能なレジスタであって、ソフトウェアによるスレッドの優先順位の変更を可能とするためのビット（優先順位イネーブルビット）を有する、スレッド切替え制御レジスタと、

を有し

前記優先順位イネーブルビットをイネーブル状態にすることにより、ソフトウェア上の優先順位切替え命令によって前記優先順位が変更可能となり、

これにより、スレッド切替えを起こすイベントに応答して、又は、前記優先順位切替え命令の使用を介して、スレッドの優先順位を調節することを特徴とする方法。

【請求項２】

前記スレッド切替え制御レジスタが、スレッド切替えを起こすイベントのそれぞれについて別々の制御ビットをさらに含み、該制御ビットをイネーブルにすることによって、対応

するイベントによるスレッド切替えが起る、請求項１記載の方法。
【請求項３】
スレッド切替えを起こすイベントが、キャッシュミス、仮想アドレスから実アドレスへのマッピングが格納される変換ルックアサイド・バッファ（ＴＬＢ）ミス、外部割込み、及びスレッド切替えタイムアウトを含む、請求項２記載の方法。
【請求項４】
前記優先順位切替え命令が、ノー・オペレーション命令である、請求項１～３のいずれか１項記載の方法。
【請求項５】
前記優先順位を表すビットが複数あり、該複数のビットの複数の組合わせのそれぞれが、各優先順位に対応する、請求項１～４のいずれか１項記載の方法。
【請求項６】
前記複数優先順位が、高、中または低である、請求項５記載の方法。
【請求項７】
各スレッドの状態が格納されたレジスタであって、ハードウェアもしくはソフトウェアによって設定することができる、各スレッドの優先順位を表すビット（優先順位状態ビット）を含む、スレッド状態レジスタと、
スレッド切替えを起こすイベントを選択する、ソフトウェア・プログラム可能なレジスタであって、ソフトウェアによるスレッドの優先順位の変更を可能とするためのビット（優先順位イネーブルビット）を有する、スレッド切替え制御レジスタと、
を備え、
前記優先順位イネーブルビットをイネーブル状態にすることにより、ソフトウェア上の優先順位切替え命令によって前記優先順位が変更可能である、事を特徴とするマルチスレッド式プロセッサ。
【請求項８】
命令が実行されずに発生したスレッド切替えの回数が所定の閾値と等しくなった以降には、命令が実行されない限りスレッド切替えをそれ以上発生しないようにするための、該閾値を格納するフォワード・プログレス・カウント・レジスタをさらに含むことを特徴とする請求項７記載のマルチスレッド式プロセッサ。
【請求項９】
指定されたサイクル数を待った後にスレッド切替えを強制するための、該サイクル数を格納するスレッド切替えタイムアウト・レジスタをさらに含むことを特徴とする請求項７または８記載のマルチスレッド式プロセッサ。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
関連特許出願データ
本発明は、下記の米国特許出願に関連し、その要旨は参照によって本明細書に組み込まれる：（１）米国特許出願第０８／９５８７１６号明細書、（２）米国特許出願第０８／９５７００２号明細書、（３）米国特許出願第０８／９５６８７５号明細書、（４）米国特許出願第０８／９５６５７７号明細書、（５）米国特許出願第７７３５７２号明細書、（６）米国特許出願第７６１３７８号明細書、（７）米国特許出願第７６１３８０号明細書、（８）米国特許出願第７６１３７９号明細書、（９）米国特許出願第４７３６９２号明細書、（１０）米国特許第５７７８２４３号明細書。
【０００２】
【従来の技術】
本発明は、全般的にはコンピュータ・データ処理システムのための改良された方法およびその装置に関し、具体的には、改良された高性能マルチスレッド式コンピュータ・データ処理システムと、そのプロセッサのハードウェア内で実施される方法に関する。
【０００３】

現代のコンピュータの基本構造には、外部の世界との間で情報を通信するための周辺装置が含まれ、そのような周辺装置は、キーボード、モニタ、テープ駆動装置、ネットワークに接続された通信回線などとすることができる。やはりコンピュータの基本構造に含まれるのが、外部の世界との間でこの情報を受け取り、処理し、配送するのに必要なハードウェアであり、これには、バス、メモリ・ユニット、入出力コントローラ、記憶装置、少なくとも１つの中央処理装置（ＣＰＵ）などが含まれる。ＣＰＵは、システムの頭脳である。ＣＰＵは、コンピュータ・プログラムを含む命令を実行し、他のシステム構成要素の動作を指示する。

【０００４】

コンピュータのハードウェアの観点から、ほとんどのシステムは、基本的に同一の形で動作する。プロセッサは、実際には、算術演算、論理比較、ある位置から別の位置へのデータの移動など、非常に単純な動作をすばやく実行する。これらの単純な動作を大量に実行するようにコンピュータに指示するプログラムが、コンピュータが洗練されたことを行っているという錯覚をもたらす。しかし、コンピュータの新機能または改良された機能としてユーザに知覚されるものは、実際には、同一の単純な動作をはるかに高速に実行する計算機である場合がある。したがって、コンピュータ・システムに対する改良を継続するためには、これらのシステムをさらに高速にする必要がある。

【０００５】

コンピュータ・システムの総合速度の尺度の１つをスループットとも称するが、これは、単位時間あたりに実行される動作の数として測定される。概念上、システム速度に対するすべての可能な改良のうちで最も単純なものは、さまざまな構成要素のクロック速度、特にプロセッサのクロック速度を高めることである。したがって、すべてが２倍の速度で走行するが、それ以外は正確に同一の形で機能する場合、そのシステムは、所与のタスクを半分の時間で実行することになる。以前に別個の構成要素から構成されていたコンピュータ・プロセッサは、構成要素のサイズの縮小と個数の減少とによって大幅に高速に動作するようになり、最終的には、プロセッサ全体が、単一チップ上の集積回路としてパッケージされるようになった。サイズの縮小によって、プロセッサのクロック速度を高めることが可能になり、したがって、システム速度を高めることが可能になった。

【０００６】

集積回路から得られる速度の大幅な向上にもかかわらず、さらに高速のコンピュータ・システムに対する需要がいまだに存在する。ハードウェア設計者は、より大規模の集積、回路のサイズのさらなる縮小および他の技法によって、さらなる速度の向上を得ることができた。しかし、設計者は、物理的なサイズの縮小を際限なく継続することは不可能であり、継続的にプロセッサ・クロック速度を高めることに限界があると考えている。したがって、コンピュータ・システムの総合速度をさらに改善するために、他の手法に注意が向けられた。

【０００７】

クロック速度を変更しなくても、複数のプロセッサを使用することによってシステム速度を改善することが可能である。集積回路チップにパッケージされた個々のプロセッサのコストが適度であるので、これが実用的になった。スレーブ・プロセッサを使用すると、作業をＣＰＵからスレーブ・プロセッサにオフロードすることによって、システム速度がかなり改善される。たとえば、スレーブ・プロセッサは、通常は、入出力装置の通信および制御など、反復的で単純な専用プログラムを実行する。複数のＣＰＵを単一のコンピュータ・システム、通常は複数のユーザを同時にサービスするホストベース・システム内に配置することも可能である。異なるＣＰＵのそれぞれが、異なるユーザのために異なるタスクを別々に実行することができ、したがって、複数のタスクを同時に実行するシステムの総合速度が高まる。しかし、アプリケーション・プログラムなどの単独のタスクが実行される速度を改善することは、はるかに困難である。複数のＣＰＵの間でさまざまな機能の実行および結果の配布を調整することは、注意を要する作業である。スレーブ入出力プロセッサの場合、機能が事前に定義され、制限されているので、これはそれほど困難ではな

いが、汎用アプリケーション・プログラムを実行する複数のＣＰＵの場合、システム設計者がプログラムの詳細を事前に知らないことが一因となって、機能を調整することがより困難になる。ほとんどのアプリケーション・プログラムは、プロセッサによって実行されるステップの単一の経路または流れに従う。この単一の経路を複数の並列経路に分割することが可能である場合もあるが、それを行うための万能のアプリケーションは、まだ研究中である。一般に、複数のプロセッサによる並列処理のために長いタスクを小さいタスクに分割することは、コードを記述するソフトウェア・エンジニアによって、ケースバイケースで行われる。この、その場限りの手法は、必ずしも反復的でなく、予測可能でもない商業トランザクションを実行する場合に特に問題がある。

【０００８】

したがって、複数のプロセッサによって総合的なシステム性能は改善されるが、個々のＣＰＵの速度を改善する理由はまだ多数存在する。ＣＰＵクロック速度が与えられた場合、１クロック・サイクルあたりに実行される命令の「平均」数を増やすことによって、ＣＰＵの速度すなわち１秒あたりに実行される命令の数をさらに増やすことが可能である。高性能単一チップ・マイクロプロセッサのための一般的なアーキテクチャが、高速実行のために頻繁に使用される命令の小さい簡略されたセットを特徴とする縮小命令セット・コンピュータ（ＲＩＳＣ）アーキテクチャであり、これらの単純な動作は、前に述べたものより高速に実行される。半導体技術が進歩するにつれて、ＲＩＳＣアーキテクチャの目標は、計算機の各クロック・サイクルに１つまたは複数の命令を実行できるプロセッサの開発になってきた。１クロック・サイクルあたりに実行される命令の「平均」数を増やすためのもう１つの手法が、ＣＰＵ内のハードウェアを変更することである。この命令あたりのクロック数というスループット尺度は、高性能プロセッサのアーキテクチャの特徴を表すのに一般的に使用されている。命令パイプライン化とキャッシュ・メモリが、これを達成することを可能にしたコンピュータ・アーキテクチャの特徴である。パイプライン命令実行を用いると、前に発行された命令が完了する前に、後続の命令の実行を開始できるようになる。キャッシュ・メモリは、頻繁に使用されるデータおよび他のデータをプロセッサの近くに格納し、これによって、ほとんどの場合に主記憶のフル・アクセス・タイムを待つことなしに、命令実行を継続できるようにするものである。並列に実行する命令を見つけるためのルック・アヘッド・ハードウェアを有する複数の実行ユニットを用いるいくつかの改良も示されてきた。

【０００９】

通常のＲＩＳＣプロセッサの性能は、スーパースカラ・コンピュータおよびＶＬＩＷ（Very Long Instruction Word）コンピュータでさらに高めることができ、これらのコンピュータの両方が、１プロセッサ・サイクルあたりに複数の命令を並列に実行する。これらのアーキテクチャでは、複数の機能ユニットまたは実行ユニットを設けて、複数のパイプラインを並列に走行させる。スーパースカラ・アーキテクチャでは、命令は、インオーダー（in-order）またはアウトオブオーダー（out-of-order）で完了できる。インオーダー完了とは、ある命令の前にディスパッチされたすべての命令が完了するまでは、その命令を完了できないことを意味する。アウトオブオーダー完了とは、事前に定義された規則が満たされる限り、ある命令の前のすべての命令が完了する前に、その命令が完了することを許可されることを意味する。

【００１０】

スーパースカラ・システムでの命令のインオーダー完了とアウトオブオーダー完了の両方について、パイプラインは、ある状況の下でストールする。前にディスパッチされ、まだ完了していない命令の結果に依存する命令が、パイプラインのストールを引き起こす可能性がある。たとえば、必要なデータがキャッシュ内にない、すなわち、キャッシュ・ミスを引き起こすロード／ストア命令に依存する命令は、そのデータがキャッシュ内で使用可能になるまでは実行できない。継続実行のため、また、高いヒット率すなわちデータがキャッシュ内ですぐに使用可能である回数と比較したデータに対する要求の回数を維持するために必要な、キャッシュ内の必要不可欠なデータを維持することは、特に大きいデータ

構造を伴う計算の場合には簡単ではない。キャッシュ・ミスは、数サイクルにわたるパイプラインのストールを引き起こす可能性があり、データがほとんどの時間に使用可能でない場合には、メモリ待ち時間の総量が厳しくなる。主記憶に使用される記憶装置は、高速になりつつあるが、そのようなメモリ・チップとハイエンド・プロセッサの間の速度ギャップは、ますます大きくなりつつある。したがって、現在のハイエンド・プロセッサ設計での実行時間のかなりの量が、キャッシュ・ミスの解決を待つのに費やされ、これらのメモリ・アクセスの遅れが、プロセッサ実行時間のうちで占める比率がますます大きくなっている。

【００１１】

ＣＰＵ内のハードウェアの効率を改善するためのもう１つの技法が、処理タスクを、スレッドと称する独立に実行可能な命令のシーケンスに分割することである。この技法は、スレッドが同一のプロセッサによって実行される点を除いて、異なるプロセッサによる独立実行のために大きいタスクを小さいタスクに分割することに関連する。ＣＰＵが、複数の理由のいずれかのために、これらのスレッドのうちの１つの処理または実行を継続できない時には、ＣＰＵは、別のスレッドに切り替え、そのスレッドを実行する。コンピュータ・アーキテクチャ界で定義される用語「マルチスレッディング」は、複数の関連するスレッドに分割された１つのタスクを意味するソフトウェアでのこの用語の使用と同一ではない。アーキテクチャ的定義では、スレッドは、独立とすることができる。したがって、この２つの用語の使用を区別するために、「ハードウェア・マルチスレッディング」が使用されることがしばしばである。本発明の文脈では、用語「マルチスレッディング」は、メモリ待ち時間を許容するためのハードウェア・マルチスレッディングを含む。

【００１２】

マルチスレッディングを用いると、プロセッサのパイプラインが、現在のスレッドに関してパイプライン・ストール状態が検出された時に、異なるスレッド上の有用な作業を行えるようになる。マルチスレッディングを用いると、非パイプライン・アーキテクチャを実施するプロセッサが、現在のスレッドに関してストール状態が検出された時に、別のスレッドに関して有用な作業を行えるようにもなる。マルチスレッディングには、２つの基本形態がある。従来の形態では、プロセッサ内でＮ個のスレッドまたは状態を保ち、サイクル単位でスレッドをインターリーブする。これによって、単一のスレッド内の命令が分離されるので、すべてのパイプライン依存性が除去される。本発明によって考慮される、マルチスレッディングのもう１つの形態では、いくつかの長い待ち時間のイベントの際にスレッドをインターリーブする。

【００１３】

マルチスレッディングの従来の形態では、スレッドごとのプロセッサ・レジスタの複製が用いられる。たとえば、商品名PowerPC（商標）の下で販売されるアーキテクチャを実施するプロセッサがマルチスレッディングを実行するためには、プロセッサは、Ｎ個のスレッドを走行させるためにＮ個の状態を維持しなければならない。したがって、汎用レジスタ、浮動小数点レジスタ、条件レジスタ、浮動小数点状態および制御レジスタ、カウント・レジスタ、リンク・レジスタ、例外レジスタ、セーブ／リストア・レジスタおよび特殊目的レジスタが、Ｎ回複製される。さらに、セグメント・ルックアサイド・バッファなどの特殊バッファを複製することができ、また、各項目にスレッド番号のタグを付けることができ、タグを付けない場合にはスレッド切り替えのたびにフラッシュしなければならない。また、一部の分岐予測機構、たとえば相関レジスタとリターン・スタックなども、複製しなければならない。幸い、レベル１命令キャッシュ（Ｌ１　Ｉ－キャッシュ）、レベル１データ・キャッシュ（Ｌ１　Ｄ－キャッシュ）、命令バッファ、ストア待ち行列、命令ディスパッチャ、機能ユニットまたは実行ユニット、パイプライン、変換ルックアサイド・バッファ（ＴＬＢ）および分岐履歴テーブルなどのプロセッサのより大きい機能のうちのいくつかは、複製する必要がない。あるスレッドが遅延に遭遇した時に、プロセッサは、すばやく別のスレッドに切り替える。このスレッドの実行は、第１のスレッドでのメモリ遅延とオーバーラップする。

【００１４】
既存のマルチスレッディング技法では、キャッシュ・ミスまたはメモリ参照の際のスレッド切り替えが説明される。この技法の主な例は、アガーワル（Agarwal）他著、「Sparcle : An Evolutionary Design for Large-Scale Multiprocessors」、IEEE Micro Volume 13、No.3、pp.48-60、1993年6月に記載されている。ＲＩＳＣアーキテクチャで適用される際には、通常は関数呼出しのサポートに使用される複数のレジスタ・セットが、複数のスレッドを維持するように変更される。８つのオーバーラップするレジスタ・ウィンドウが、４つのオーバーラップしないレジスタ・セットになるように変更され、各レジスタ・セットは、トラップおよびメッセージ処理のために予約される。このシステムでは、リモート・メモリ要求をもたらす第１レベルのキャッシュ・ミスのそれぞれで発生するスレッド切替えが開示される。このシステムは、当技術分野での進歩を表すが、現代のプロセッサ設計では、プロセッサに付加される複数のレベルのキャッシュ・メモリまたは高速メモリが使用されることがしばしばである。プロセッサ・システムは、周知のアルゴリズムを使用して、その主メモリ記憶域のどの部分がキャッシュの各レベルにロードされるかを決定し、したがって、第１レベルのキャッシュ内に存在しないメモリ参照が発生するたびに、プロセッサは、第２レベルまたはさらに上位のレベルのキャッシュからのそのメモリ参照の取得を試みなければならない。

【００１５】
【発明が解決しようとする課題】
したがって、本発明の目的は、マルチスレッド式データ処理システムで実施されるハードウェア論理およびレジスタを使用する、マルチレベル・キャッシュ・システムでのメモリ待ち時間に起因する遅延を減らすことのできる、改良されたデータ処理システムを提供することである。

【００１６】
【課題を解決するための手段】
本発明は、命令の２つのスレッドの間で実行を切り替える能力を有するマルチスレッド式プロセッサと、任意選択のスレッド切替え条件のソフトウェア・オーバーライドを有するハードウェア・レジスタで実施されるスレッド切替え論理とを提供することによって、この目的に対処する。命令のさまざまなスレッドを処理することによって、スレッドの間でのプロセッサの使用を最適化できるようになる。プロセッサが命令の第２のスレッドを実行できるようにすることによって、キャッシュ、メモリ、外部入出力、直接アクセス記憶装置などのさまざまな記憶要素から第１スレッドのために必要なデータまたは命令を取り出している時に、そうでなければ遊休状態になるプロセッサ利用度が高まる。スレッド切替えの条件は、スレッドごとに異なるものとすることができ、また、ソフトウェア・スレッド制御マネージャの使用によって処理中に変更することができる。

【００１７】
第１スレッドが、キャッシュ・ミスなどの完了に多数のサイクルを必要とする待ち時間イベントを有する時に、第２スレッドを処理することができるが、この待ち時間イベント中に、第２スレッドが、はるかに短い時間で完了できるキャッシュ・ミスを、同一のキャッシュ・レベルまたは異なるキャッシュ・レベルで経験する可能性がある。

【００１８】
命令実行なしでスレッドを切り替えるサイクルの反復で各スレッドがロックするスラッシングは、本発明によって、フォワード・プログレス・カウント・レジスタを実施し、プログラム可能な最大回数までのスレッド切替えを許容し、それを超えた時点であるスレッドが実行可能になるまでプロセッサがスレッドの切替えを停止するようにする方法を実施することによって解決される。フォワード・プログレス・カウント・レジスタとその閾値によって、命令が実行されずに発生したスレッド切替えの回数が監視され、その回数が閾値と等しくなった時には、命令が実行されない限りスレッド切替えがそれ以上発生しなくなる。フォワード・プログレス・カウント・レジスタの追加の長所は、外部コンピュータ・ネットワークへのアクセスなどの非常に長い待ち時間イベント用の閾値と、キャッシュ・

ミスなどの短い待ち時間イベント用の別のフォワード・プログレス閾値など、待ち時間イベントに合わせてレジスタと閾値をカストマイズできることである。

【００１９】

スレッド切替えタイムアウト・レジスタで指定されたサイクル数を待った後にスレッド切替えを強制することによって、あるスレッドに対するコンピュータ処理が、長すぎる時間にわたって非アクティブになることが防止される。コンピュータ処理システムは、共用資源の競合から生じるハングを経験しなくなる。スレッド間のプロセッサ・サイクル割当の公平さが達成され、外部割込みおよび他のプロセッサ外部のイベントに対する最大応答待ち時間が制限される。

【００２０】

すばやいスレッド切替えは、スレッドの状態、スレッドの優先順位およびスレッド切替え条件を記憶するハードウェア・レジスタによって達成される。

【００２１】

プロセッサ内の１つまたは複数のスレッドの優先順位を、スレッド切替えハードウェア・レジスタを使用して変更することができる。割込み要求からの信号またはソフトウェア命令のいずれかを使用して、状態レジスタの、各スレッドの優先順位を表すビットを変更する。その後、各スレッドの優先順位に応じて、スレッド切替えを行って、高い優先順のスレッドがより多くの処理サイクルを得られるようにすることができる。優先順位変更の長所は、スレッド切替えの頻度を変更でき、クリティカルなタスクの実行サイクルを増やすことができ、スレッド切替え待ち時間のために高い優先順位のスレッドが失う処理サイクル数を減らすことができることである。

【００２２】

本発明のもう１つの態様は、それぞれが優先順位を有する命令の複数のスレッドのうちの少なくとも１つを、少なくとも１つのメモリ・ユニットに接続されたマルチスレッド式プロセッサ内で実行し、複数のスレッドのうちの１つまたは複数の優先順位を変更することによって、データ処理システム内でマルチスレッド式動作を実行するためのコンピュータ方法である。すべてのスレッドの優先順位に基づいて、多重プロセッサは、あるスレッドから別のスレッドに切り替えることができる。優先順位は、割込みからの信号または、スレッド切替えマネージャと称するハードウェアおよびソフトウェアの配置のいずれかによって変更することができる。スレッドの優先順位から、あるスレッドが別のスレッドより高い優先順位を有することができることが暗示されるが、複数のスレッドが等しい優先順位を有することも可能である。各スレッドが３つの優先順位のうちの１つを有することのできる優先順位方式を提示する。より低い優先順位の第１スレッドからの切替えは、第２スレッドが作動可能状態である時に発生する。より高い優先順位の第１スレッドからの切替えは、高い優先順位のスレッドが、レベル１キャッシュまたはすばやい応答時間を有する他のメモリ・ユニットからのミスを有する時に使用不能にすることができる。

【００２３】

簡単に言うと、本発明は、マルチスレッド式プロセッサ内の命令の複数のスレッドのそれぞれに、プロセッサ内のすべてのスレッドの相対優先順位に基づいてプロセッサ・サイクルを割り振ることによる、コンピュータ処理の方法である。

【００２４】

スレッド切替えマネージャは、少なくとも１つの優先順位切替え命令と、計算機状態レジスタやスレッド切替え制御レジスタなど第１レジスタ内の少なくとも１つのスレッド切替え優先順位ビットと、スレッド状態レジスタなどの第２レジスタ内の複数の優先順位状態ビットとを有し、優先順位切替え命令によって、複数の優先順位状態ビットのいずれかを変更することができる、ハードウェアおよびソフトウェアの配置である。複数のスレッドを有するマルチスレッド式処理システムでは、実行中のスレッドを、アクティブ・スレッドと称する。優先順位切替え命令は、アクティブ・スレッドまたはマルチスレッド式処理システム内の他のいずれかのスレッドに対応する優先順状態ビットを変更することができる。その後、スレッド切替え優先順位ビットがイネーブルされている場合に、マルチスレ

ッド式プロセッサは、アクティブ・スレッドまたは別のスレッドのいずれかの優先順位の変更に応答して、アクティブ・スレッドから処理を切り替える。

【００２５】

スレッド切替えマネージャに対応するように既存プロセッサを変更するために、優先順位切替え命令は、マルチスレッド式プロセッサの複数のアーキテクチャ的レジスタのいずれをも変更してはならず、不正命令割込みなしでマルチスレッド式プロセッサ上で実行されなければならず、たとえば、優先順位切替え命令は、好ましいノー・オペレーション命令とすることができる。

【００２６】

本発明は、命令の複数のスレッドを処理するための手段と、複数のスレッドのそれぞれの優先順位状態を記憶するための手段と、複数のスレッドのそれぞれの優先順位状態を変更するための手段と、それに応答して、処理手段が複数のスレッドのうちの第１スレッドから第２スレッドへ処理を切り替えられるようにするための手段とを含むコンピュータ処理システムでもある。

【００２７】

本発明の他の目的、特徴および特性と、構造の関連する要素の方法、動作および機能と、部分の組合せと、製造の経済性とは、好ましい実施例の以下の詳細な説明および添付図面から明らかになる。添付図面は、すべてが本明細書の一部を形成し、同様の符号は、さまざまな図面の対応する部分を示す。

【００２８】

本発明自体、ならびにその使用の好ましい態様、さらなる目的および長所は、下記の添付図面と共に例の実施例の詳細な説明を参照することによって最もよく理解される。

【００２９】

【発明の実施の形態】

ここで図面、具体的には図１を参照すると、本発明の方法およびシステムの実施に使用することのできる、コンピュータ・データ処理システム１０の高水準ブロック図が示されている。本発明を使用することのできるコンピュータ・データ処理システム１０の主要なハードウェア構成要素および相互接続が、図１に示されている。命令を処理するための中央処理装置（ＣＰＵ）１００は、キャッシュ１２０、１３０および１５０に結合される。命令キャッシュ１５０には、ＣＰＵ１００による実行のための命令が格納される。データ・キャッシュ１２０および１３０には、ＣＰＵ１００によって使用されるデータが格納される。キャッシュは、主記憶１４０内のランダム・アクセス・メモリと通信する。ＣＰＵ１００および主記憶１４０も、バス・インターフェース１５２を介してシステム・バス１５５と通信する。さまざまな入出力プロセッサ（ＩＯＰ）１６０ないし１６８が、システム・バス１５５に付加され、直接アクセス記憶装置（ＤＡＳＤ）１７０、テープ駆動装置１７２、遠隔通信回線１７４、ワークステーション１７６およびプリンタ１７８などのさまざまな記憶装置および入出力装置との通信をサポートする。図１は、高水準でコンピュータ・データ処理システム１０の代表的な構成要素を示す目的のものであり、そのような構成要素の数と種類を変更できることを理解されたい。

【００３０】

ＣＰＵ１００内では、プロセッサ・コア１１０に、特化した機能ユニットが含まれ、これらの機能ユニットのそれぞれが、命令のシーケンシング、整数を用いる演算の実行、実数を用いる演算の実行、アドレス可能記憶域と論理レジスタ・アレイの間の値の転送などのプリミティブ動作を実行する。図２は、プロセッサ・コア１００を示す図である。好ましい実施例では、データ処理システム１０のプロセッサ・コア１００は、単一集積回路のパイプライン式スーパースカラ・マイクロプロセッサであり、これは、たとえばＩＢＭ社によって販売されるPowerPC（商標）６０４マイクロプロセッサ・チップなど、商品名PowerPC（商標）の下で販売されるＲＩＳＣプロセッサの系列などのコンピュータ・アーキテクチャを使用して実施できる。

【００３１】

下で述べるように、データ処理システム１０には、さまざまなユニット、レジスタ、バッファ、メモリおよび他のセクションが含まれることが好ましく、これらのすべてが集積回路によって形成されることが好ましい。図では、さまざまなデータ経路が簡略化されていることを理解されたい。実際には、さまざまな構成要素から出入りする多数の別々の並列のデータ経路がある。さらに、本明細書に記載の発明に密接に関係しないさまざまな構成要素が省略されているが、追加機能のためにプロセッサに追加ユニットが含まれることを理解されたい。データ処理システム１０は、縮小命令セット・コンピューティング（ＲＩＳＣ）技法または他のコンピューティング技法に従って動作することができる。

【００３２】

図２からわかるように、データ処理システム１０のプロセッサ・コア１００には、レベル１データ・キャッシュ（Ｌ１　Ｄ－キャッシュ）１２０、レベル２（Ｌ２）キャッシュ１３０、主記憶１４０およびレベル１命令キャッシュ（Ｌ１　Ｉ－キャッシュ）１５０が含まれることが好ましく、これらのすべてが、機能的にさまざまなバス接続を使用して記憶域制御ユニット２００に相互接続される。図１からわかるように、記憶域制御ユニット２００には、Ｌ１　Ｄ－キャッシュ１２０およびＬ２キャッシュ１３０と、主記憶１４０と、複数の実行ユニットとを相互接続するための遷移キャッシュ２１０が含まれる。Ｌ１　Ｄ－キャッシュ１２０とＬ１　Ｉ－キャッシュ１５０は、プロセッサ１００の一部としてチップ上に設けられることが好ましく、主記憶１４０とＬ２キャッシュ１３０は、チップ外に設けられる。メモリ・システム１４０は、プロセッサ・コア１００の内部または外部とすることのできるランダム・アクセス・メイン・メモリ、プロセッサ・コア１００の外部の他のデータ・バッファおよびキャッシュ（存在する場合）、および、たとえば図１に示されたＤＡＳＤ１７０、テープ駆動装置１７２、ワークステーション１７６などの他の外部メモリを表す目的のものである。Ｌ２キャッシュ１３０は、主記憶１４０より高速のメモリ・システムであることが好ましく、選択されたデータをＬ２キャッシュ１３０に格納することによって、主記憶１４０への参照の結果として発生するメモリ待ち時間を最小にすることができる。図１からわかるように、Ｌ２キャッシュ１３０および主記憶１４０は、Ｌ１　Ｉ－キャッシュ１５０に直接に、また記憶域制御ユニット２００を介して命令ユニット２２０に接続される。

【００３３】

Ｌ１　Ｉ－キャッシュ１５０からの命令は、命令ユニット２２０に出力されることが好ましく、命令ユニット２２０は、本発明の方法およびシステムに従って、さまざまなサブプロセッサ・ユニット、たとえば分岐ユニット２６０、固定小数点ユニット２７０、記憶域制御ユニット２００、浮動小数点ユニット２８０および、データ処理システム１０のアーキテクチャによって指定される他のユニットによる複数のスレッドの実行を制御する。当業者は、図１に示されたさまざまな実行ユニットのほかに、現代のスーパースカラ・マイクロプロセッサ・システムに、本発明の趣旨および範囲から逸脱せずに追加することのできるそのような実行ユニットのそれぞれの複数の版が含まれることがしばしばであることを了解するであろう。これらのユニットのほとんどは、入力として、汎用レジスタ（ＧＰＲ）２７２および浮動小数点レジスタ（ＦＰＲ）２８２などのさまざまなレジスタからのソース・オペランド情報を有する。さらに、複数の特殊目的レジスタ（ＳＰＲ）２７４を使用することができる。図１からわかるように、記憶域制御ユニット２００と遷移キャッシュ２１０は、汎用レジスタ２７２および浮動小数点レジスタ２８２に直接に接続される。汎用レジスタ２７２は、特殊目的レジスタ２７４に接続される。

【００３４】

このマルチスレッド式プロセッサ１００に固有の機能ハードウェア・ユニットの中に、スレッド切替え論理４００と遷移キャッシュ２１０がある。スレッド切替え論理４００には、どのスレッドをアクティブ・スレッドまたは実行中のスレッドにするかを決定するさまざまなレジスタが含まれる。スレッド切替え論理４００は、機能的に、記憶域制御ユニット２００と、実行ユニット２６０、２７０および２８０と、命令ユニット２２０に接続される。記憶域制御ユニット２００内の遷移キャッシュ２１０は、マルチスレッディングを

実施できなければならない。記憶域制御ユニット２００と遷移キャッシュ２１０は、１スレッドあたり少なくとも１つの未処理のデータ要求を許容することが好ましい。したがって、たとえばＬ１　Ｄ－キャッシュ・ミスの発生に応答して、第１スレッドが延期される時に、第２スレッドが、そこに存在するデータについてＬ１　Ｄ－キャッシュ１２０にアクセスできるようになる。第２スレッドも、Ｌ１　Ｄ－キャッシュ・ミスをもたらす場合には、別のデータ要求が発行され、したがって、複数のデータ要求を、記憶域制御ユニット２００および遷移キャッシュ２１０内で維持しなければならない。遷移キャッシュ２１０は、参照によって本明細書に組み込まれる米国特許出願第０８／７６１３７８号明細書の遷移キャッシュであることが好ましい。記憶域制御ユニット２００と、実行ユニット２６０、２７０および２８０と、命令ユニット２２０は、すべてが機能的にスレッド切替え論理４００に接続され、スレッド切替え論理４００は、どのスレッドを実行するかを決定する。

【００３５】

図２からわかるように、バス２０５は、たとえば記憶域制御ユニット２００へのデータ要求および命令ユニット２２０へのＬ２キャッシュ１３０ミスなどの通信のために、記憶域制御ユニット２００と命令ユニット２２０の間に設けられる。さらに、変換ルックアサイド・バッファ（ＴＬＢ）２５０が設けられ、これには、仮想アドレスから実アドレスへのマッピングが格納される。図示されてはいないが、本発明では、変換ルックアサイド・バッファ２５０に類似の形で動作するセグメント・ルックアサイド・バッファなどの追加の高水準メモリ・マッピング・バッファを設けることができる。

【００３６】

図３は、記憶域制御ユニット２００を詳細に示す図であり、名前から暗示されるように、このユニットは、さまざまなキャッシュ、バッファおよび主記憶を含むさまざまな記憶ユニットからのデータおよび命令の入出力を制御する。図３からわかるように、記憶域制御ユニット２００には、機能的にＬ１　Ｄ－キャッシュ１２０、マルチプレクサ３６０、Ｌ２キャッシュ１３０および主記憶１４０に接続された遷移キャッシュ２１０が含まれる。さらに、遷移キャッシュ２１０は、シーケンサ３５０から制御信号を受け取る。シーケンサ３５０には、命令またはデータの取出要求を処理するために、複数、好ましくは３つのシーケンサが含まれる。シーケンサ３５０は、遷移キャッシュ２１０およびＬ２キャッシュ１３０に制御信号を出力し、主記憶１４０との間で制御信号を送受する。

【００３７】

図３に示された記憶域制御ユニット２００内のマルチプレクサ３６０は、Ｌ１　Ｄ－キャッシュ１２０、遷移キャッシュ２１０、Ｌ２キャッシュ１３０および主記憶１４０からデータを受け取り、データをメモリに格納する場合には、実行ユニット２７０および２８０からデータを受け取る。これらの供給源のうちの１つからのデータは、マルチプレクサ３６０によって選択され、シーケンサ３５０から受け取った選択制御信号に応答して、Ｌ１　Ｄ－キャッシュ１２０または実行ユニットに出力される。さらに、図３からわかるように、シーケンサ３５０は、第２のマルチプレクサ３７０を制御する選択信号を出力する。シーケンサ３５０からのこの選択信号に基づいて、マルチプレクサ３７０は、Ｌ２キャッシュ１３０または主記憶１４０からのデータを、Ｌ１　Ｉ－キャッシュ１５０または命令ユニット２２０に出力する。上で述べた制御信号および選択信号を作る際に、シーケンサ３５０は、Ｌ１　Ｄ－キャッシュ１２０用のＬ１ディレクトリ３２０とＬ２キャッシュ１３０用のＬ２ディレクトリ３３０にアクセスし、これらを更新する。

【００３８】

本明細書に記載のプロセッサのマルチスレッディング能力に関して、記憶域制御ユニット２００のシーケンサ３５０は、スレッド切替え論理４００にも信号を出力して、データ要求および命令要求の状態を示す。したがって、キャッシュ１２０、１３０および１５０と、主記憶１４０と、変換ルックアサイド・バッファ２５０からのフィードバックが、シーケンサ３５０に送られ、その後、スレッド切替え論理４００に通信され、スレッド切替え論理４００は、下で述べるようにスレッド切替えをもたらすことができる。マルチスレッ

ド式プロセッサ内でのスレッド切替えを引き起こすように設計されたイベントが発生する装置は、機能的にシーケンサ３５０に接続されることに留意されたい。

【００３９】

図４は、スレッドを切り替えるかどうかを判定し、切り替える場合にはどのスレッドに切り替えるかを判定するスレッド切替え論理ハードウェア４００の論理表現およびブロック図である。記憶域制御ユニット２００と命令ユニット２２０は、スレッド切替え論理４００と相互接続される。スレッド切替え論理４００は、命令ユニット２２０に組み込まれることが好ましいが、多数のスレッドがある場合には、スレッド切替え論理４００の複雑さが増し、その結果、スレッド切替え論理が命令ユニット２２０の外部になる場合がある。説明を簡単にするために、スレッド切替え論理４００は、記憶域制御ユニット２００の外部にあるものとして図示した。

【００４０】

この実施例でスレッドの切替えをもたらすいくつかのイベントは、記憶域制御ユニット２００のシーケンサ３５０からスレッド切替え論理４００へ、信号線４７０、４７２、４７４、４７６、４７８、４８０、４８２、４８４および４８６を介して通信される。他の待ち時間イベントが、スレッド切替えを引き起こす可能性があるが、このリストは、網羅的であることを意図したものではなく、スレッド切替えを実施できる方法を代表するものにすぎない。命令ユニット２２０内にない第１スレッドＴ０または第２スレッドＴ１のいずれかによる命令の要求は、それぞれ図４の符号４７０または４７２によって示されるスレッド切替えをもたらす可能性があるイベントである。信号線４７４は、Ｔ０またはＴ１のいずれかであるアクティブ・スレッドが、Ｌ１　Ｄ－キャッシュ１２０ミスを経験する時を示す。スレッドＴ０またはＴ１のいずれかに関するＬ２キャッシュ１３０のキャッシュ・ミスは、それぞれ信号線４７６または４７８によって知らされる。信号線４８０および４８２は、それぞれＴ０スレッドまたはＴ１スレッドの継続実行のためにデータが返される時にアクティブになる。変換ルックアサイド・バッファ・ミスおよびテーブル・ウォークの完了は、それぞれ信号線４８４または４８６によって示される。

【００４１】

これらのイベントは、すべてがスレッド切替え論理４００に供給され、具体的には、スレッド状態レジスタ４４０およびスレッド切替えコントローラ４５０に供給される。スレッド切替え論理４００は、スレッドごとに１つのスレッド状態レジスタを有する。本明細書に記載の実施例では、２つのスレッドが表現されるので、第１スレッド用Ｔ０のＴ０状態レジスタ４４２と、第２スレッドＴ１用のＴ１状態レジスタ４４４があり、これらを本明細書で説明する。スレッド切替え論理４００には、どのイベントがスレッド切替えをもたらすかを制御するスレッド切替え制御レジスタ４１０が含まれる。たとえば、スレッド切替え制御レジスタ４１０は、状態変化がスレッド切替えコントローラ４５０によって見られるようにするイベントをブロックし、その結果、ブロックされたイベントの結果としてスレッドが切り替えられなくすることができる。スレッド状態レジスタおよびスレッドを変更する論理と動作は、本明細書と同時に出願され、参照によって本明細書に組み込まれる米国特許出願第０８／９５７００２号明細書の主題である。スレッド切替え制御レジスタ４１０は、本明細書と同時に出願され、参照によって本明細書に組み込まれる、米国特許出願第０８／９５８７１６号明細書の主題である。フォワード・プログレス・カウント・レジスタ４２０は、スラッシングの防止に使用され、スレッド切替え制御レジスタ４１０に含めることができる。フォワード・プログレス・カウント・レジスタ４２０は、本明細書と同時に出願され、参照によって本明細書に組み込まれる米国特許出願第０８／９５６８７５号明細書の主題である。スレッド切替えタイムアウト・レジスタ４３０は、本明細書と同時に出願され、参照によって本明細書に組み込まれる米国特許出願第０８／９５６５７７号明細書の主題であり、これによって、公平さとライブロック発行が割り振られる。制限的ではないが、最後に、スレッド切替えコントローラ４５０には、スレッドを切り替えるかどうかと、どのスレッドにどの状況の下で切り替えるのかを実際に判定するすべての論理の頂点を表す無数の論理ゲートが含まれる。これらの論理構成要素とその機能

のそれぞれを、さらに詳細に説明する。

【００４２】

スレッド状態レジスタ

スレッド状態レジスタ４４０には、各スレッドの状態レジスタが含まれ、名前からわかるように、対応するスレッドの状態が格納される。この例では、Ｔ０スレッド状態レジスタ４４２とＴ１スレッド状態レジスタ４４４がある。ビットの数と、各スレッドの状態を記述するための特定のビットの割振りは、特定のアーキテクチャおよびスレッド切替え優先順位方式に合わせてカスタマイズすることができる。２つのスレッドを有するマルチスレッド式プロセッサのスレッド状態レジスタ４４２および４４４のビットの割振りの例を、下の表に示す。

スレッド状態レジスタのビット割振り

(0) 命令／データ
0=命令
1=データ

(1:2) ミス・タイプ・シーケンサ
00=なし
01=変換ルックアサイド・バッファ・ミス（I/Dのビット0を検査）
10=L1キャッシュ・ミス
11=L2キャッシュ・ミス

(3) 遷移
0=現在の状態への遷移はスレッド切替えをもたらさない
1=現在の状態への遷移はスレッド切替えをもたらす

(4:7) 予約済み

(8) 0=ロード
1=ストア

(9:14) 予約済み

(15:17) フォワード・プログレス・カウンタ
111=リセット（このスレッド中に命令が完了した）
000=命令完了なしでのこのスレッドの１回目の実行
001=命令完了なしでのこのスレッドの２回目の実行
010=命令完了なしでのこのスレッドの３回目の実行
011=命令完了なしでのこのスレッドの４回目の実行
100=命令完了なしでのこのスレッドの５回目の実行

(18:19) 優先順位（ソフトウェアによって設定可能）
00=中
01=低
10=高
11=＜不正＞

(20:31) 予約済み

(32:63) 64ビット実施形態の場合に予約済み

【００４３】

上で説明した実施例では、ビット０によって、ミスまたはプロセッサが実行をストールした理由が、命令の要求とデータの要求のどちらの結果であるかが識別される。ビット１および２は、図５の説明でさらに説明するように、要求された情報が使用可能でなかったかどうかと、使用可能でなかった場合に、どのハードウェアから使用可能でなかったか、すなわち、データまたは命令の変換されたアドレスが変換ルックアサイド・バッファ２５０になかったのか、データまたは命令自体がＬ１　Ｄ－キャッシュ１２０またはＬ２キャッシュ１３０になかったのかを示す。ビット３は、スレッドの状態の変化が、スレッド切替えをもたらすかどうかを示す。スレッドは、スレッド切替えをもたらさずに状態を変更することができる。たとえば、スレッドＴ１がＬ１キャッシュ・ミスを経験する時にスレッ

ド切替えが発生する場合に、スレッドＴ１がＬ２キャッシュ・ミスを経験する場合、Ｌ１キャッシュ・ミスの際にすでにスレッドが切り替えられているので、スレッド切替えはない。しかし、Ｔ１の状態は、まだ変化する。その代わりに、選択によって、スレッド切替え論理４００が、Ｌ１キャッシュ・ミスの際に切り替えないように構成またはプログラミングされる場合には、スレッドがＬ１キャッシュ・ミスを経験した時に、スレッドの状態が変化してもスレッド切替えはない。スレッド状態レジスタ４４２および４４４のビット８は、特定のスレッドによって要求された情報が、プロセッサ・コアにロードされるのか、プロセッサ・コアからキャッシュまたは主記憶にストアされるのかに割り当てられる。ビット１５ないし１７は、フォワード・プログレス・カウント・レジスタ４２０に関して後で説明するように、スラッシングの防止に割り振られる。ビット１８および１９は、スレッドの優先順位を示すために、ハードウェアで設定するか、ソフトウェアによって設定することができる。

【００４４】

図５は、データ処理システム１０によって処理されるスレッドの現在の実施例での４つの状態を表し、これらの状態は、スレッド状態レジスタ４４０のビット位置１：２に格納される。状態００は、「実行可能」状態すなわち、必要なすべてのデータおよび命令が使用可能であるので、スレッドの処理の準備ができていることを表す。状態１０は、スレッドがＬ１　Ｄ－キャッシュ１２０にデータが返されるかＬ１　Ｉ－キャッシュ１５０に命令が返されるかのいずれかを待っているので、プロセッサ内でのスレッドの実行がストールしているスレッド状態を表す。状態１１は、スレッドがＬ２キャッシュ１３０にデータが返されるのを待っていることを表す。状態０１は、テーブル・ウォークと称する、変換ルックアサイド・バッファ２５０でのミスがある、すなわち、仮想アドレスがエラー状態であったか、使用可能でなかったことを示す。図５には、スレッド状態の階層も示されており、スレッドの実行の準備ができていることを示す状態００が、最も高い優先順位を有する。短い待ち時間イベントには、高い優先順位を割り当てることが好ましい。

【００４５】

図５には、データがさまざまな供給源から取り出される時の状態の変化も示されている。スレッドＴ０の正常に割り込まれない実行は、ブロック５１０で状態００として表されている。Ｌ１　Ｄ－キャッシュまたは Ｉ－キャッシュのミスが発生した場合、スレッドの状態は、記憶域制御ユニット２００からの信号線４７４（図４）または命令ユニット２２０からの信号線４７０（図４）で送られる信号に従って、ブロック５１２に示されている状態１０に変化する。要求されたデータまたは命令が、Ｌ２キャッシュ１３０内にあり、取り出される場合には、ブロック５１０のＴ０の正常実行が再開される。同様に、図５のブロック５１４は、Ｌ２キャッシュ・ミスを表し、これによって、Ｔ０またはＴ１のいずれかのスレッドの状態が、記憶域制御ユニット２００が信号線４７６または４７８（図４）でミスの信号を送る時に、状態１１に変化する。信号線４８０および４８２（図４）に示されているように、Ｌ２キャッシュ内の命令またはデータが、主記憶１４０から取り出され、プロセッサ・コア１００にロードされる時には、状態は、やはりブロック５１０の状態００に戻る。要求された情報の仮想アドレスが変換ルックアサイド・バッファ２５０内で使用可能でない時には、ブロック５１６に示されるように、ＴＬＢミスまたは状態０１として、記憶域制御ユニット２００が、信号線４８４（図４）を介してスレッド・レジスタ４４０に通信する。そのアドレスが使用可能になる時または、信号線４８６（図４）上で記憶域制御ユニット２００によって送られるデータ記憶域割り込み命令がある場合には、スレッドの状態は、状態００に戻り、実行の準備ができる。

【００４６】

状態の数と、各状態が表すものは、コンピュータ設計者が自由に選択できる。たとえば、あるスレッドが、Ｌ１　Ｉ－キャッシュ・ミスとＬ１　Ｄ－キャッシュ・ミスなど、複数のＬ１キャッシュ・ミスを有する場合には、キャッシュ・ミスのタイプのそれぞれに別々の状態を割り当てることができる。その代わりに、単一のスレッド状態を割り当てて、複数のイベントまたはできごとを表すことができる。

【００４７】
等しい優先順位を有する２つのスレッドについて、スレッドを切り替えるかどうかを判定するスレッド切替えアルゴリズムの例を示す。このアルゴリズムは、本発明の教示に従って、より多くのスレッドおよびスレッド切替え条件のためにそれ相応に拡張し、変更することができる。スレッド切替えアルゴリズムによる、スレッド状態レジスタ４４０（図４）に格納された各スレッドの状態と各スレッドの優先順位との間の相互作用は、各サイクルに動的に問い合わされる。アクティブ・スレッドＴ０がＬ１ミスを有する場合に、このアルゴリズムは、休止スレッドＴ１がＬ２ミスの解決を待っている場合を除いて、休止スレッドＴ１へのスレッド切替えを引き起こす。切替えが発生せず、アクティブ・スレッドＴ０のＬ１キャッシュ・ミスがＬ２キャッシュ・ミスになった場合には、このアルゴリズムは、プロセッサに、Ｔ１の状態に無関係に休止スレッドＴ１に切り替えるように指示する。両方のスレッドがＬ２キャッシュ・ミスの解決を待っている場合には、最初にＬ２ミスを解決されたスレッドが、アクティブ・スレッドになる。すべての切替え決定時に、行われる処置は、最も可能性の高い事例に合わせて最適化され、最良の性能をもたらす。Ｌ２キャッシュ・ミスから生じるスレッド切替えは、性能の低下をもたらす余分なスレッド切替えが発生しない場合に、他方のスレッドの状態次第である。

【００４８】
スレッド切替え制御レジスタ
どのマルチスレッド式プロセッサにも、スレッド切替えに関連する待ち時間と性能のペナルティが存在する。本明細書で説明する好ましい実施例のマルチスレッド式プロセッサでは、この待ち時間に、現在のスレッドに割り込むことができ、現在のスレッドが次に呼び出された時に正しく再始動できる点まで現在のスレッドの実行を完了するのに必要な時間と、スレッド固有のハードウェア機能を現在のスレッドの状態から新しいスレッドの状態に切り替えるのに必要な時間と、新しいスレッドを再始動し、その実行を開始するのに必要な時間が含まれる。本発明と共に動作可能なスレッド固有のハードウェア機能には、上で説明したスレッド状態レジスタと、参照によって本明細書に組み込まれる米国特許第５７７８２４３号明細書に記載のメモリ・セルが含まれることが好ましい。粒度の粗いマルチスレッド式データ処理システムで最適の性能を達成するために、スレッド切替えを生成するイベントの待ち時間は、通常の単一スレッド・モードに対して、マルチスレッド・モードでのスレッド切替えに関連する性能コストより大きくなければならない。

【００４９】
スレッド切替えを生成するのに使用されるイベントの待ち時間は、ハードウェアとソフトウェアの両方に依存する。たとえば、マルチスレッド式プロセッサの特定のハードウェア検討事項には、プロセッサ・チップの外部のＬ２キャッシュの実施に使用される外部ＳＲＡＭの速度が含まれる。Ｌ２キャッシュのＳＲＡＭが高速になると、Ｌ１ミスの平均待ち時間が減るが、ＳＲＡＭが低速になると、Ｌ１ミスの平均待ち時間が増える。したがって、あるスレッド切替えイベントが、スレッド切替えのペナルティより大きい外部Ｌ２キャッシュ・データ・アクセス待ち時間を有するハードウェアのＬ１キャッシュ・ミスとして定義される場合に、高性能が得られる。特定のソフトウェア・コードの特性が、スレッド切替えイベントの待ち時間にどのように影響するかの例として、コードのＬ２キャッシュのヒット対ミス比すなわち、データがＬ２キャッシュにないので主記憶から取り出さなければならない回数と比較した、データが実際にＬ２キャッシュ内で使用可能である回数を検討されたい。Ｌ２ヒット対ミス比が高いと、Ｌ１キャッシュ・ミスが、より長い待ち時間のＬ２ミスをほとんどもたらさないので、Ｌ１キャッシュ・ミスの平均待ち時間が減る。Ｌ２ヒット対ミス比が低いと、より長い待ち時間のＬ２ミスをもたらすＬ１ミスが増えるので、Ｌ１ミスの平均待ち時間が増える。したがって、実行中のコードが高いＬ２ヒット対ミス比を有する場合には、Ｌ２キャッシュ・データ・アクセス待ち時間がスレッド切替えペナルティより小さいので、スレッド切替えイベントとしてのＬ１ミスを使用不能にすることができる。低いＬ２ヒット対ミス比を有するソフトウェア・コードを実行する時には、Ｌ１キャッシュ・ミスが、より長い待ち時間のＬ２キャッシュ・ミスになる可能性

が高いので、Ｌ１キャッシュ・ミスをスレッド切替えイベントとして使用可能にすることになる。

【００５０】

いくつかのタイプの待ち時間イベントは、簡単には検出できない。たとえば、いくつかのシステムでは、キャッシュ・ミスが発生した時に、Ｌ２キャッシュが、命令ユニットに信号を出力する。しかし、他のＬ２キャッシュは、たとえばＬ２キャッシュ・コントローラがプロセッサとは別のチップ上にあり、したがって、プロセッサが状態変化を簡単に判定できない場合に、そのような信号を出力しない。これらのアーキテクチャでは、プロセッサに、未処理のＬ１キャッシュ・ミスごとに１つのサイクル・カウンタを含めることができる。所定のサイクル数の前にミス・データがＬ２キャッシュから返されない場合には、プロセッサは、Ｌ２キャッシュ・ミスがあったかのように動作し、スレッドの状態をそれ相応に変更する。このアルゴリズムは、複数の別個のタイプの待ち時間が存在する他の場合にも適用可能である。例のみとして、多重プロセッサでのＬ２キャッシュ・ミスの場合、主記憶からのデータの待ち時間は、別のプロセッサからのデータの待ち時間と大きく異なる場合がある。これらの２つのイベントに、スレッド状態レジスタ内で異なる状態を割り当てることができる。これらの状態を区別する信号が存在しない場合には、カウンタを使用して、スレッドがＬ２キャッシュ・ミスに遭遇した後に、スレッドがどの状態にならなければならないかを推定することができる。

【００５１】

スレッド切替え制御レジスタ４１０は、スレッド切替えを生成するイベントを選択するソフトウェア・プログラム可能レジスタであり、定義されたスレッド切替え制御イベントのそれぞれについて別々のイネーブル・ビットを有する。本明細書で説明する実施例では、スレッドごとに別々のスレッド切替え制御レジスタ４１０は実施されないが、スレッドごとに別々のスレッド切替え制御レジスタ４１０を実施して、より多くのハードウェアおよび複雑さという犠牲と引き換えにより高い柔軟性と性能をもたらすことができる。さらに、あるスレッド切替え制御レジスタ内のスレッド切替え制御イベントは、他のスレッド切替え制御レジスタのスレッド切替え制御イベントと同一である必要はない。

【００５２】

スレッド切替え制御レジスタ４１０は、米国特許第５０７９７２５号明細書に開示された動的走査通信インターフェースなどのソフトウェアを用いるサービス・プロセッサによるか、ソフトウェア・システム・コードを用いてプロセッサ自体によって、書き込むことができる。スレッド切替え制御レジスタ４１０の内容は、スレッド切替えの生成を使用可能または使用不能にするために、スレッド切替えコントローラ４５０によって使用される。レジスタ４１０内の１の値によって、そのビットに関連するスレッド切替え制御イベントが使用可能にされて、スレッド切替えが生成される。スレッド切替え制御レジスタ４１０内の０の値によって、そのビットに関連するスレッド切替え制御イベントが、スレッド切替えの生成を禁止される。もちろん、実行中のスレッド内の命令によって、その特定のスレッドまたは他のスレッドのスレッド切替え条件のうちのいくつかまたはすべてを使用不能にすることができる。下の表に、スレッド切替えイベントと、レジスタ４１０内のイネーブル・ビットの間の関連を示す。

スレッド切替え制御レジスタのビット割当

(0) Ｌ１データ・キャッシュ取出ミスに対するスイッチ
(1) Ｌ１データ・キャッシュ・ストア・ミスに対するスイッチ
(2) Ｌ１命令キャッシュ・ミスに対するスイッチ
(3) 命令ＴＬＢミスに対するスイッチ
(4) Ｌ２キャッシュ取出ミスに対するスイッチ
(5) Ｌ２キャッシュ・ストア・ミスに対するスイッチ
(6) Ｌ２命令キャッシュ・ミスに対するスイッチ
(7) データＴＬＢ／セグメント・ルックアサイド・バッファ・ミスに対するスイッチ
(8) Ｌ２キャッシュ・ミスおよび休止スレッド非Ｌ２キャッシュ・ミスに対するスイ

ッチ
(9)　スレッド切替えタイムアウト値到達時のスイッチ
(10)　Ｌ２キャッシュ・データが返された時のスイッチ
(11)　入出力外部アクセスに対するスイッチ
(12)　ダブルＸストア：２つのうちの１番目でのミスに対するスイッチ＊
(13)　ダブルＸストア：２つのうちの２番目でのミスに対するスイッチ＊
(14)　複数／列ストア：すべてのアクセスでのミスに対するスイッチ
(15)　複数／列ロード：すべてのアクセスでのミスに対するスイッチ
(16)　予約済み
(17)　ダブルＸロード：２つのうちの１番目でのミスに対するスイッチ＊
(18)　ダブルＸロード：２つのうちの２番目でのミスに対するスイッチ＊
(19)　計算機状態レジスタ（問題状態）ビット、msr(pr)=1の場合のor 1,1,1命令に対するスイッチ。msr(pr)と独立のソフトウェア優先順位変更を可能にする。ビット１９が１の場合、or 1,1,1命令によって低優先順位が設定される。ビット１９が０の場合、or 1,1,1命令が実行される時にmsr(pr)=0の場合に限って優先順位が低に設定される。後で説明する、ソフトウェアによる優先順位の変更を参照されたい。
(20)　予約済み
(21)　スレッド切替え優先順位イネーブル
(22:29) 予約済み
(30:31) フォワード・プログレス・カウント
(32:63) ６４ビット・レジスタ実施形態で予約済み
＊ダブルＸロード／ストアとは、基本ハーフワード、ワードまたはダブル・ワードの、ダブルワード境界をまたぐロードまたはストアを指す。この文脈でのダブルＸロード／ストアは、複数ワードまたはワードの列のロードまたはストアではない。

【００５３】
スレッド切替えタイムアウト・レジスタ
上で述べたように、粒度の粗いマルチスレッド式プロセッサは、スレッド切替えをトリガするために、長い待ち時間のイベントに頼る。実行中に、多重プロセッサ環境内のプロセッサまたはマルチスレッド式アーキテクチャのバックグラウンド・スレッドが、単独の所有者だけを有することのできる資源の所有権を有し、別のプロセッサまたはアクティブ・スレッドが、フォワード・プログレスを行う前にその資源へのアクセスを必要とする場合がある。その例には、メモリ・ページ・テーブルの更新またはタスク・ディスパッチャからのタスクの取得が含まれる。アクティブ・スレッドが資源の所有権を得ることができなくても、スレッド切替えイベントはもたらされないが、スレッドは、有用な作業を行うことができないループを回り続ける。この場合、資源を保持しているバックグラウンド・スレッドは、プロセッサへのアクセスを得ず、その結果、スレッド切替えイベントに遭遇せず、アクティブ・スレッドにならないので、資源を解放することができない。

【００５４】
スレッドの間での処理サイクルの割振りが、もう１つの問題である。あるスレッド上で走行するソフトウェア・コードが、同一のプロセッサ内の他のスレッド上で走行するソフトウェア・コードと比較して長い待ち時間の切替えイベントにほとんど遭遇しない場合には、そのスレッドは、処理サイクルの公平な割当分以上の処理サイクルを得る。最大の許容可能な時間を超える可能性があるもう１つの過度な遅延が、限られた時間期間内に外部割込みをサービスするために待機するかプロセッサの外部の他のイベントを待機する非アクティブ・スレッドの待ち時間である。したがって、有用な処理が達成されていない場合に、システムがハングしないようにするために、ある時間の後に休止スレッドへのスレッド切替えを強制的に行うことが好ましくなる。

【００５５】
ある時間期間の後にスレッド切替えを強制するための論理が、スレッド切替えタイムアウト・レジスタ４３０（図４）、デクリメンタおよび、減分された値を保持する減分レジス

タである。スレッド切替えタイムアウト・レジスタ４３０は、スレッド切替えタイムアウト値を保持する。この実施例で使用されるスレッド切替えタイムアウト・レジスタ４３０の実施形態を、次の表に示す。

スレッド切替えタイムアウト・レジスタのビット

(0:21) 予約済み

(22:31) スレッド切替えタイムアウト値

【００５６】

本明細書で説明する本発明の実施例では、スレッドごとに別々のスレッド切替えタイムアウト・レジスタ４３０が実施されないが、柔軟性を高めるためにそれを行うことは可能である。同様に、複数のスレッドがある場合に、各スレッドが同一のスレッド切替えタイムアウト値を有する必要はない。スレッド切替えが発生するたびに、スレッド切替えタイムアウト・レジスタ４３０からのスレッド切替えタイムアウト値が、ハードウェアによって減分レジスタにロードされる。減分レジスタは、減分レジスタ値が０に等しくなるまで各サイクルに１回減分され、０になった時に、スレッド切替えコントローラ４５０に信号が送られ、スレッド切替えコントローラ４５０は、命令を処理する準備ができている他のスレッドがない場合を除いて、スレッド切替えを強制する。たとえば、システム内の他のすべてのスレッドが、キャッシュ・ミスで待機状態になっており、命令を実行する準備ができていない場合には、スレッド切替えコントローラ４５０は、スレッド切替えを強制しない。減分レジスタの値が０に達した時に、命令を処理する準備ができている他のスレッドが存在しない場合には、別のスレッドが命令を処理する準備ができるまで、減分された値は０で凍結され、準備ができた時点で、スレッド切替えが発生し、減分レジスタに、そのスレッドのスレッド切替えタイムアウト値が再ロードされる。同様に、減分レジスタは、簡単に増分レジスタと命名することができ、スレッドが実行中である時に、そのレジスタをある所定の値まで増分することができ、その値に達した時にスレッド切替えが強制される。

【００５７】

スレッド切替えタイムアウト・レジスタ４３０は、上で述べたようにサービス・プロセッサによって書き込むか、ソフトウェア・コードを用いてプロセッサ自体によって書き込むことができる。スレッド切替えタイムアウト・レジスタ４３０にロードされるスレッド切替えタイムアウト値は、特定のハードウェア構成または特定のソフトウェア・コードに従って、不要なスレッド切替えから生じる浪費サイクルを最小にするためにカスタマイズすることができる。スレッド切替えタイムアウト・レジスタ４３０の値が大きすぎると、アクティブ・スレッドが別のスレッドによって保持されている資源を待っている場合と、外部割込みまたは他のプロセッサ外部のイベントの応答待ち時間が長すぎる場合に、性能低下がもたらされる可能性がある。また、値が大きすぎると、一方のスレッドが多数のスレッド切替えイベントを経験し、もう一方のスレッドがそうでない場合に、公平さが損なわれる可能性がある。スレッド切替えを引き起こす、最も頻繁な最長の待ち時間イベント、たとえば主記憶へのアクセスより２倍ないし数倍長いスレッド切替えタイムアウト値が、推奨される。スレッド切替えタイムアウト・レジスタ４３０で指定されたサイクル数だけ待った後にスレッド切替えを強制することによって、共用資源の競合に起因するシステム・ハングが防止され、スレッド間のプロセッサ・サイクル割振りの公平さが実施され、外部割り込みおよび他のプロセッサ外部のイベントに対する最大応答待ち時間が制限される。

【００５８】

フォワード・プログレスの保証

スレッド切替えが発生し、新しいスレッドがアクティブになるたびに、少なくとも１つの命令が実行されなければならないことは、単一の命令によって複数のキャッシュ・アクセスまたは複数のキャッシュ・ミスが発生する時など、いくつかの状況では制限が強すぎる。たとえば、取出命令は、要求された命令がキャッシュ内にない場合にＬ１　Ｉ－キャッシュ１５０ミスを引き起こす可能性があるが、その命令が返された時に、必要なデータが

Ｌ１　Ｄ－キャッシュ１２０内で使用可能でない可能性がある。同様に、変換ルックアサイド・バッファ２５０でのミスが、データ・キャッシュ・ミスをももたらす可能性がある。したがって、フォワード・プログレスを厳密に実施する場合には、後続アクセスでのミスは、スレッド切替えをもたらさない。第２の問題は、一部のキャッシュ・ミスが、完了に大量のサイクルを必要とする可能性があり、その時間の間に、別のスレッドが、同一のキャッシュ・レベルで、はるかに短い時間で完了できるキャッシュ・ミスを経験する可能性があることである。第１のスレッドに戻る時に、厳密なフォワード・プログレスが実施される場合には、プロセッサは、より短いキャッシュ・ミスを有するスレッドに切り替えることができない。

【００５９】

各スレッドが、命令実行を伴わないスレッド切替えの反復サイクルでロックされるスラッシングの問題を救済するために、フォワード・プログレス・カウント・レジスタ４２０（図４）が存在し、これによって、フォワード・プログレス閾値と称するプログラム可能な最大回数までのスレッド切替えが許容される。そのスレッド切替えの最大回数の後は、命令を完了しなければ切替えは発生しない。この形で、スラッシングが防止される。フォワード・プログレス・カウント・レジスタ４２０は、実際には、スレッド切替え制御レジスタ４１０のビット３０：３１とするか、プロセッサのためのソフトウェア・プログラム可能フォワード・プログレス閾値レジスタとすることができる。フォワード・プログレス・カウント論理は、スレッドの状態を示し、命令実行なしでスレッドが経験したスレッド切替えの回数のために割り振られる、スレッド状態レジスタ４４２および４４４のビット１５：１７を使用する。これらのビットは、フォワード・プログレス・カウンタを含むことが好ましい。

【００６０】

スレッドの状態が変化し、スレッド切替えアルゴリズムが呼び出される時に、アクティブ・スレッド内で少なくとも１つの命令が完了している場合、アクティブ・スレッドのフォワード・プログレス・カウンタは、リセットされ、スレッド切替えアルゴリズムは、プロセッサ内のスレッドの間でのスレッド状態の比較を継続する。完了した命令がない場合、アクティブ・スレッドのスレッド状態レジスタ内のフォワード・プログレス・カウンタ値が、フォワード・プログレス閾値と比較される。カウンタ値が、閾値と等しくない場合には、スレッド切替えアルゴリズムは、プロセッサ内のスレッドのスレッド状態の評価を継続する。その後、スレッド切替えが発生した場合に、フォワード・プログレス・カウンタが増分される。しかし、カウンタ値が閾値または状態と等しい場合には、命令を実行できるまで、すなわち、フォワード・プログレスが発生するまで、スレッド切替えは発生しない。閾値レジスタが値０を有する場合には、別のスレッドに切り替える前に、アクティブ・スレッド内で少なくとも１つの命令が完了しなければならないことに留意されたい。各スレッド切替えが、３プロセッサ・サイクルを必要とし、２つのスレッドが存在し、スレッド切替え論理が、５回の試行の値にスレッド切替えの試行を停止するようにプログラミングされている場合、プロセッサのスラッシングが発生する最大サイクル数は、３０サイクルである。当業者は、一方でフォワード・プログレスが行われないのでスレッド切替えを禁止することと、他方でタイムアウト・カウントを超えたのでスレッド切替えを強制することの間に潜在的な衝突が存在することを了解するであろう。このような衝突は、アーキテクチャおよびソフトウェアに従って簡単に解決することができる。

【００６１】

図６は、スラッシングを防止する、スレッド切替え論理４００のフォワード・プログレス・カウント機能の流れ図である。ブロック６１０で、スレッドＴ０に関係するスレッド状態レジスタ４４２のビット１５：１７が、状態１１１にリセットされる。ブロック６２０で、このスレッドの実行を試み、状態が０００に変化する。スレッドＴ０で命令が成功裡に実行された場合、スレッドＴ０の状態は、１１１に戻り、そのままにとどまる。しかし、スレッドＴ０が命令を実行できない場合には、スレッドＴ１または、プロセッサ・アーキテクチャで３つ以上のスレッドが許容される場合には別のバックグラウンド・スレッド

へのスレッド切替えが発生する。Ｔ１または他のバックグラウンド・スレッドからのスレッド切替えが発生し、実行がスレッドＴ０に戻った時に、ブロック６３０で、スレッドＴ０を実行する２回目の試みが行われ、スレッドＴ０の状態は００１になる。やはり、スレッドＴ０がスレッド切替えイベントに遭遇した場合に、プロセッサの制御は、スレッドＴ０から別のスレッドに切り替えられる。同様に、たとえばＴ１などの他のスレッドからスレッドＴ０へのスレッド切替えが発生した時に、Ｔ０の状態は、Ｔ０実行の３回目の試みでは０１０に変化し（ブロック６４０）、Ｔ０実行の４回目の試みでは０１１に変化し（ブロック６５０）、Ｔ０実行の５回目の試みでは状態１００に変化する（ブロック６６０）。

【００６２】

この実施形態では、スレッドＴ０への切替えの試みが５回ある。５回目の試みの後と、スレッド状態レジスタ（ＴＳＲ）４４２のビット１５：１７の値が、スレッド切替え制御レジスタ（ＴＳＣ）４１０のビット３０：３１の値＋１に等しい時すなわち、ＴＳＣ（３０：３１）＋１＝ＴＳＲ（１５：１７）の時に必ず、スレッドＴ０からのスレッド切替えが発生しなくなる。５回の試みは、任意の数であり、不成功の実行を伴う切替えの許容可能な最大回数すなわち、フォワード・プログレス閾値は、プログラム可能であることが了解されよう。また、あるアーキテクチャでは、５回の切替えが多すぎ、他のアーキテクチャでは５回が少なすぎることが理解されよう。どの場合でも、命令実行なしでスレッドに切り替える試みの回数の間の関係を閾値と比較しなければならず、その閾値に達した後には、そのスレッドからのスレッド切替えが発生せず、プロセッサは、そのスレッドに関連する待ち時間が解決されるまで待機する。本明細書で説明する実施例では、スレッド状態レジスタ４４２のビット１５：１７によって表されるスレッドの状態が、スレッド切替え制御レジスタ４１０のビット３０：３１と比較される。フォワード・プログレス論理による早すぎるスレッド切替えのブロックを防ぐための、入出力装置との相互作用などの極端に長い待ち時間を有する特定のイベントのための特別な処理によって、プロセッサ性能が改善される。これらの極端に長い待ち時間のイベントを処理する方法の１つが、フォワード・プログレス・カウンタの増分をブロックするか、データが返されなかった場合のフォワード・プログレス・カウンタと閾値の間の比較の出力信号を無視することである。極端に長い待ち時間のイベントを処理するもう１つの方法は、これらの特定のイベントについて、別のより大きいフォワード・プログレス・カウントを使用することである。

【００６３】

スレッド切替えマネージャ

プロセッサにディスパッチされたすべてのソフトウェア・スレッドのスレッド状態は、前に説明したように図４のスレッド状態レジスタ４４２および４４４で維持されることが好ましい。単一のプロセッサでは、一時に１つのスレッドがその命令を実行し、他のすべてのスレッドは、休止状態になる。実行がアクティブ・スレッドから休止スレッドに切り替えられるのは、アクティブ・スレッドが、フォワード・プログレス・レジスタ４２０、スレッド切替え制御レジスタ４１０またはスレッド切替えタイムアウト・レジスタ４３０に関して上で説明した長い待ち時間のイベントに出会った時である。どのスレッドがアクティブであるかに無関係に、これらのハードウェア・レジスタでは、実行の過程の間に動的に変化しない状態が使用される。

【００６４】

スレッド切替えマネージャによってスレッド切替え条件を変更する柔軟性によって、総合的なシステム性能が改善される。ソフトウェア・スレッド切替えマネージャは、スレッド切替えの頻度を変更でき、クリティカルなタスクが使用できる実行サイクルを増やすことができ、スレッド切替え待ち時間のために失われる総合サイクルを減らすことができる。スレッド切替えマネージャは、コンパイル時または、オペレーティング・システムによる実行中のいずれかにプログラミングすることができ、たとえば、ロックしているループからスレッド切替えの頻度を変更でき、また、低い優先順位状態の休止スレッドが外部割込みを待っているか他の形で作動可能であるのでオペレーティング・システム・タスクをデ

ィスパッチすることができる。アクティブ・スレッドからのスレッド切替えを許容しないか、その頻度を減らし、その結果、現在の命令ストリームの性能が、そこからの切替えとそこに戻る切替えから生じる待ち時間を被らなくすることが有利である場合がある。その代わりに、スレッドが、総合的なシステム性能を強化するために、本質的にその優先順位を下げ、その結果として、それへの切替えの頻度を下げるか、そのスレッドからの切替えの頻度を高めることによって、その実行サイクルの一部またはすべてを捨てることができる。スレッド切替えマネージャは、スレッド切替えを無条件で強制または禁止することもでき、どのスレッドが次に実行のために選択されるかに影響することもできる。

【００６５】

複数優先順位スレッド切替え方式では、各スレッドに優先順位値を割り当てて、切替えを引き起こす条件を制限する。場合によっては、ハードウェアにスレッド優先順位を変更させることが望ましい可能性もある。たとえば、低優先順位スレッドが、あるイベントを待っており、そのイベントが発生した時に、ハードウェアが、そのスレッドの優先順位を引き上げて、外部割込みなどのイベントに対するそのスレッドの応答時間に影響することができる。スレッド間の相対優先順位またはあるスレッドの優先順位は、そのようなイベントの処理に影響する。スレッドの優先順位は、イベントに応答してハードウェアによって、または、１つまたは複数の命令の使用を介してスレッド切替えマネージャソフトウェアによって、調節することができる。スレッド切替えマネージャは、ハードウェア・スレッド切替え論理によって実行される処置を変更して、効果的にスレッドの相対優先順位を変更する。

【００６６】

３つの優先順位が、本明細書で説明する２スレッドの実施例と共に使用され、これによって、システム性能に悪影響を及ぼさずに、性能のチューニングを可能にするのに十分な、スレッドの間の区別がもたらされる。３つの優先順位を用いると、２つのスレッドが、中優先順位の同等の状況を有することができる。２つのスレッドに関する３つの優先順位の選択は、制限的であることを目的とするものではない。いくつかのアーキテクチャでは、「通常」状態が、一方のスレッドが必ず他方のスレッドより高い優先順位を有する状態であるものとすることができる。ハードウェアで設定するかソフトウェアによってプログラミングすることのできる１つまたは複数の優先順位を有する３つ以上の実行のスレッドを含むことが、本発明の範囲内であることが意図されている。

【００６７】

各スレッドの３つの優先順位は、高、中および低である。スレッドＴ０の優先順位がスレッドＴ１と同一である時には、スレッド切替え論理に対する影響はない。両方のスレッドが等しい優先順位を有するので、優先的に実行時間を与えられるスレッドは存在しない。スレッドＴ０の優先順位が、スレッドＴ１の優先順位より高い時には、Ｔ０からＴ１へのスレッド切替えイベントは、あるスレッド切替えイベントすなわち、すべてのＬ１キャッシュ・ミスすなわち、データ・ロード、データ・ストアおよび命令取出について使用不能にされる。というのは、Ｌ１キャッシュ・ミスが、Ｌ２ミスおよび変換などの他の条件よりはるかに高速に解決されるからである。どのスレッド切替えイベントでも、使用不能にし、その結果、スレッドＴ０に、スレッドＴ１より多数の実行サイクルを受け取る機会を与えることができ、これによって、スレッドＴ０が、過剰な数の実行サイクルを浪費しない限り、実行を継続できるようになる。しかし、プロセッサは、スレッドＴ０が比較的長い実行待ち時間、たとえば、Ｌ２キャッシュ・ミスまたはコンピュータ・システムの外部の供給源からのデータの取出などを経験する場合に、スレッドＴ１に制御を譲る。Ｔ１からＴ０へのスレッド切替えは、休止スレッドＴ０がスレッドＴ１をプリエンプトした場合にスレッドＴ０が作動可能になった時に切替えが発生する点を除いて、影響を受けない。この事例は、Ｌ２キャッシュ・ミスまたは変換要求が原因でスレッドＴ０からの切替えが発生し、その条件が、スレッドＴ１の実行中にバックグラウンドで解決された時に発生すると予想される。スレッドＴ０がスレッドＴ１より低い優先順位を有する場合は、上の場合でスレッドの指定を逆転したものに類似する。

【００６８】
スレッド優先順位の変更によるスレッド切替えの管理を実装することができる異なる手法が存在する。新しい命令をプロセッサ・アーキテクチャに追加することができる。所望の動作を有する副作用を有する既存のプロセッサ命令を使用することもできる。ソフトウェア制御を可能にする方法の間での選択に影響する要因には、以下が含まれる。（ａ）新規命令を含めるためのアーキテクチャ再定義の容易さと、既存プロセッサに対するアーキテクチャ変更の影響、（ｂ）異なる版のプロセッサ上で同一のソフトウェアを走行させることの望ましさ、（ｃ）新規の専用命令の使用と、既存命令を再利用し結果の副作用を定義することとの間の性能トレードオフ、（ｄ）たとえば、特定のロードまたはストアなどの一部の既存命令のすべての実行によって効果が生じるか、特にその効果を生じさせるためにストリームに命令を追加することによるさらなる制御を必要とするか、など、ソフトウェアによる制御の所望のレベル。

【００６９】
本明細書で説明するアーキテクチャは、その値によってプロセッサのアーキテクチャ的汎用レジスタが変更されない、未使用命令を活用する。この機能は、プロセッサ・アーキテクチャを更新してマルチスレッディング機能を組み込むのに非常に重要である。そうでなければ、特殊命令をコーディングすることができる。命令は、「好ましいノー・オペレーション」or 0,0,0であるが、他の命令が、効果的にノー・オペレーションとして働くことができる。ノー・オペレーションまたはnopは、その実行が、コンピュータに、動作を実行せずに次の命令の実行に進行させる命令である。好ましいアーキテクチャの実施例では、or命令の異なる版、or 0,0,0または1,1,1か、スレッド優先順位を変更するために追加の優先順位切替えの意味を付加することのできる既存命令を使用することによって、同一の命令ストリームを、不正命令割込みなどの悪影響なしにプロセッサ上で実行することができる。不正命令割込みは、不正命令の実行が試みられる時または、実施形態によって提供されない予約済みまたは任意選択の命令の実行が試みられる時に生成される。機能拡張では、計算機状態レジスタの状態を使用して、これらの命令の意味を変更する。たとえば、ユーザがこれらのスレッド優先順位命令の一部またはすべてをコーディングし、それらが提供する機能にアクセスできるようにすることは、望ましくない場合がある。それらが提供する特殊機能は、実行のあるモードだけで発生するように定義することができ、それらの命令は、他のモードでは効果がなく、通常通りノー・オペレーションとして実行される。

【００７０】
二重スレッド・マルチスレッド式プロセッサを使用する可能な実施形態の１つでは、実行中のソフトウェア自体の一部になる下記の３つの優先順位切替え命令を使用して、それ自体の優先順位を変更する。

tsop 1 or 1,1,1 休止スレッドへの切替え
tsop 2 or 1,1,1 アクティブ・スレッドを「低」優先順位に設定する
休止スレッドに切り替える
注：TSC[19]=1でなければ特権モードでのみ有効
tsop 3 or 2,2,2 アクティブ・スレッドを「中」優先順位に設定する
tsop 4 or 3,3,3 アクティブ・スレッドを「高」優先順位に設定する
注：特権モードでのみ有効

【００７１】
優先順位切替え命令tsop 1およびtsop 2は、本明細書でor 1,1,1として実施されるものと同一の命令とすることができるが、これらを別々の命令とすることもできる。これらの命令は、スレッド切替え制御レジスタ４１０のビット１９および２１と、本明細書で説明した計算機状態レジスタの問題／優先順位ビットとに相互作用する。スレッド切替え制御レジスタ４１０のビット２１が１の値を有する場合には、スレッド切替えマネージャは、そのスレッドの優先順位に、スレッド状態レジスタのビット１８：１９で表される３つの優先順位のうちの１つをセットすることができる。スレッド切替え制御レジスタ４１０のビ

ット１９が値０を有する場合には、命令tsop 2のスレッド切替えおよびスレッド優先順位の設定は、計算機状態レジスタの問題／優先順位ビットによって制御される。その一方で、スレッド切替え制御レジスタ４１０のビット１９が値１を有する場合または、計算機状態レジスタの問題／優先順位ビットが値０を有し、命令or 1,1,1がコードに存在する場合には、アクティブ・スレッドの優先順位は、低に設定され、実行は、休止スレッドがイネーブルされる場合に即座に休止スレッドまたはバックグラウンド・スレッドに切り替えられる。命令or 2,2,2では、アクティブ・スレッドの優先順位が、計算機状態レジスタの問題／優先順位ビットの値に無関係に中に設定される。命令or 3,3,3では、計算機状態レジスタの問題／優先順位ビットが０の値を有する時に、アクティブ・スレッドの優先順位に高がセットされる。スレッド切替え制御レジスタ４１０のビット２１が０の場合、両方のスレッドの優先順位に、中がセットされ、優先順位に対するor x,x,xの影響がブロックされる。外部割込み要求がアクティブであり、対応するスレッドの優先順位が低の場合、そのスレッドの優先順位は、中に設定される。

【００７２】

スレッド優先順位によって変更されるイベントは、（１）データをロードする際のＬ１　Ｄ－キャッシュ・ミスに対するスイッチと、（２）データをストアする際のＬ１　Ｄ－キャッシュ・ミスに対するスイッチと、（３）命令を取り出す際のＬ１　Ｉ－キャッシュ・ミスに対するスイッチと、（４）休止スレッドが作動可能状態である場合のスイッチである。さらに、外部割込みの活動化によって、対応するスレッドの優先順位を変更することができる。下の表に、スレッド切替えを引き起こす条件に対する優先順位の影響を示す。列３および４の「ＴＳＣ」だけの項目は、スレッド切替えを開始するためにスレッド切替え制御（ＴＳＣ）レジスタ４１０に示された条件を使用することを意味する。「０として扱われるＴＳＣ［０：２］」の項目は、スレッド切替え制御レジスタ４１０のビット０：２が、そのスレッドに関してこれらのビットの値が０であるかのように扱われ、スレッド切替え制御レジスタ４１０の他のビットが、スレッド切替えを引き起こす条件の定義にそのまま使われることを意味する。列４の「スレッドＴ０作動可能時」は、スレッドＴ０が、それからのスレッド切替えを引き起こしたミス・イベントの待機を終えると同時に、スレッドＴ０への切替えが発生することを意味する。列３の「スレッドＴ１作動可能時」は、スレッドＴ１が、それからのスレッド切替えを引き起こしたミス・イベントの待機を終えると同時に、スレッドＴ１への切替えが発生することを意味する。ミス・イベントが、スレッド切替えタイムアウトである場合には、より高い優先順位のスレッドに切り替えられる前に、より低い優先順位のスレッドが命令を完了するという保証はない。

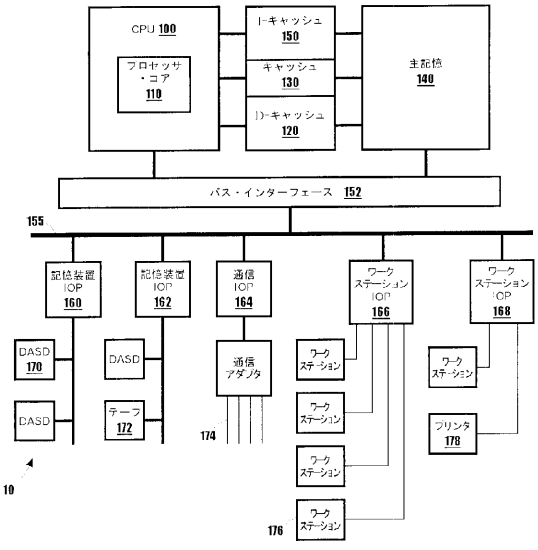
【表１】

T0 優先順位	T1 優先順位	T0 スレッド切替え条件	T1 スレッド切替え条件
高	高	TSC	TSC
高	中	0 として扱われる TSC[0:2]	TSC または T0 作動可能の場合
高	低	0 として扱われる TSC[0:2]	TSC または T0 作動可能の場合
中	高	TSC または T1 作動可能の場合	0 として扱われる TSC[0:2]
中	中	TSC	TSC
中	低	0 として扱われる TSC[0:2]	TSC または T0 作動可能の場合
低	高	TSC または T1 作動可能の場合	0 として扱われる TSC[0:2]
低	中	TSC または T1 作動可能の場合	0 として扱われる TSC[0:2]
低	低	TSC	TSC

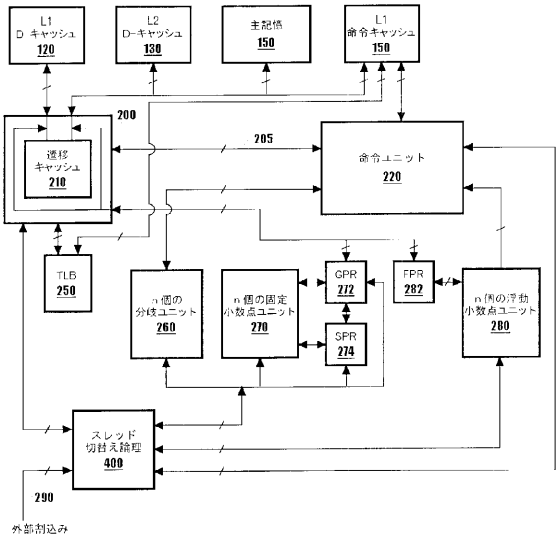
【００７３】
生産的な作業を行わないスレッドには、遊休ループ内のすべての命令がスレッド切替えを引き起こす場合であっても、性能の損失を避けるために、低優先順位を与えることが推奨される。それでも、低優先順位に設定されたスレッドに対して外部割込みが要求された場合に、ハードウェアがスレッド優先順位を変更できるようにすることが重要である。この場合、そのスレッドは、割込みに対するすばやい応答を可能にするために、中優先順位に引き上げられる。これによって、外部イベントを待っているスレッドが、それ自体を低優先順位に設定し、イベントがシグナリングされるまでその状態にとどまることが可能になる。
【００７４】
最も実用的であり好ましい実施例と現在考えられるものに関して本発明を説明してきたが、本発明は、開示された実施例に制限されず、逆に、請求項の趣旨および範囲に含まれるさまざまな修正形態および同等配置を含むことが意図されていることを理解されたい。
【図面の簡単な説明】
【図１】　本明細書に記載の発明を実施することのできるコンピュータ・システムのブロック図である。
【図２】　本発明によるマルチスレッド式データ処理システムの高水準ブロック図である。
【図３】　図２の記憶域制御ユニットのブロック図である。
【図４】　図２のスレッド切替え論理、記憶域制御ユニットおよび命令ユニットのブロック図である。
【図５】　図４に示されたスレッドが異なるスレッド切替えイベントを経験する際のスレ

ッドの状態の変化を示す図である。
【図６】　本発明のフォワード・プログレス・カウントの流れ図である

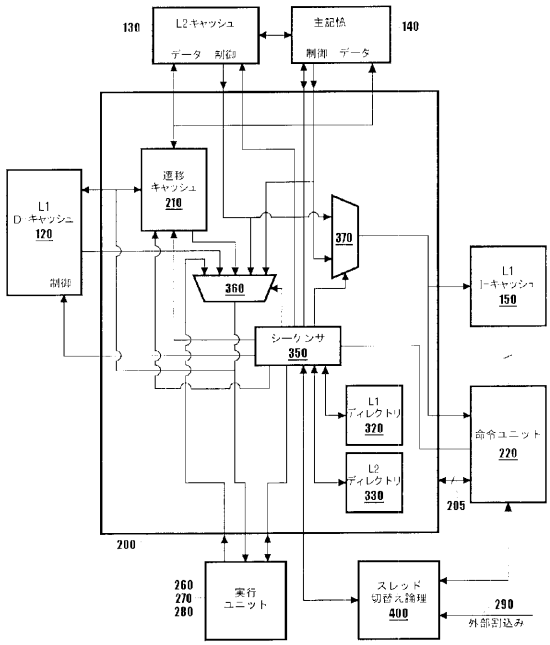
【図１】



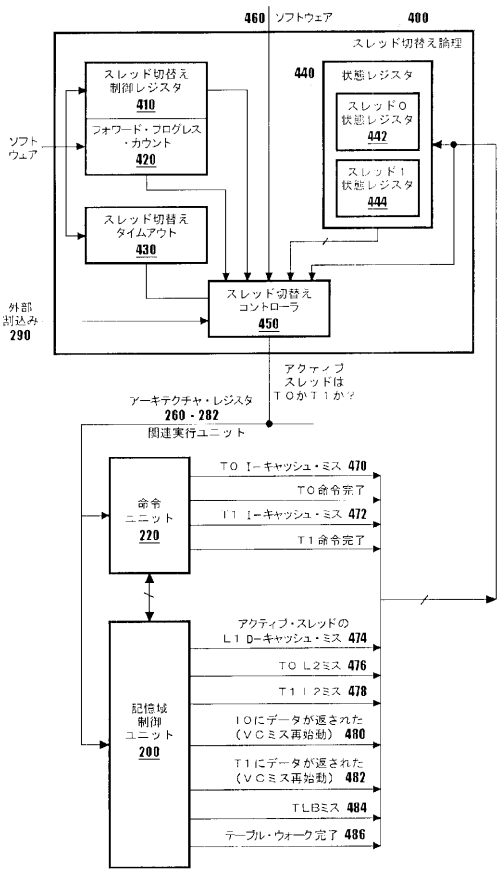
【図２】



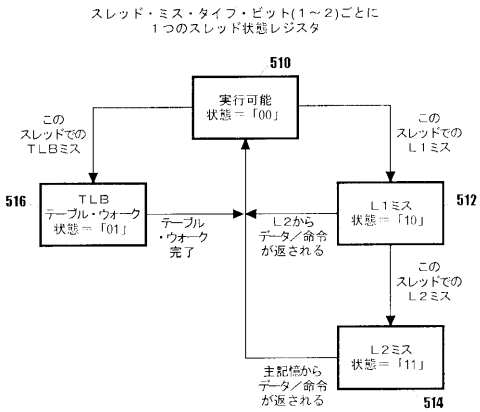
【図３】



【図４】



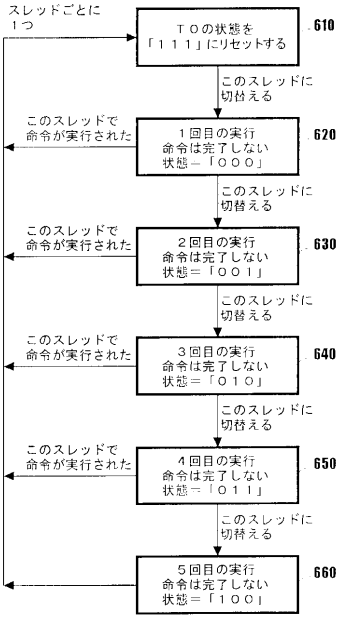
【図５】



状態の階層

最高＝状態「００」遊休
次　＝状態「１０」Ｌ１ミス
次　＝状態「１１」Ｌ２ミス
最低＝状態「０１」テーブル・ウォーク

【図６】



最大プログレス・カウント＝ TSC(30:31)+1
実行中のスレッドのTSR(15:17)が最大プログレス・カウント
以上の場合にスレッド切り替えをブロックする

フロントページの続き

(72)発明者　ボルケンハーゲン、ジョン、マイケル
アメリカ合衆国５５９０２　ミネソタ州ロチェスター　ウェストヒル・ドライブ　サウスウェスト　１３５９
(72)発明者　フリン、ウィリアム、・トーマス
アメリカ合衆国５５９０２　ミネソタ州ロチェスター　フォーティーンス・アベニュー　サウスウェスト　２５１６
(72)発明者　ウォットレング、アンドリュー、ヘンリー
アメリカ合衆国５５９０１　ミネソタ州ロチェスター　マナー・ビュー・ドライブ　ノースウェスト　４２２４

審査官　殿川　雅也

(56)参考文献　特開平０９－００６６３３（ＪＰ，Ａ）
特開平０７－０７３０５１（ＪＰ，Ａ）
柴田幸茂他，超並列計算機の要素プロセッサ向きメッセージ駆動スレッドアーキテクチャ，情報処理学会研究報告，社団法人情報処理学会，１９９５年　8月２５日，第９５巻，第８０号，第２１７－２２４頁，９５－ＡＲＣ－１１３－２８

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 9/46ー9/54