

(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2016-212639
(P2016-212639A)

(43) 公開日 平成28年12月15日(2016. 12. 15)

(51) Int. Cl.			F I			テーマコード (参考)
G06F	9/46	(2006.01)	G06F	9/46	410	
G06F	9/48	(2006.01)	G06F	9/46	452F	

審査請求 未請求 請求項の数 10 O L (全 18 頁)

(21) 出願番号	特願2015-95939 (P2015-95939)	(71) 出願人	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
(22) 出願日	平成27年5月8日 (2015.5.8)	(74) 代理人	100079049 弁理士 中島 淳
		(74) 代理人	100084995 弁理士 加藤 和詳
		(74) 代理人	100099025 弁理士 福田 浩志
		(72) 発明者	田▲邨▼ 優人 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		(72) 発明者	中島 耕太 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

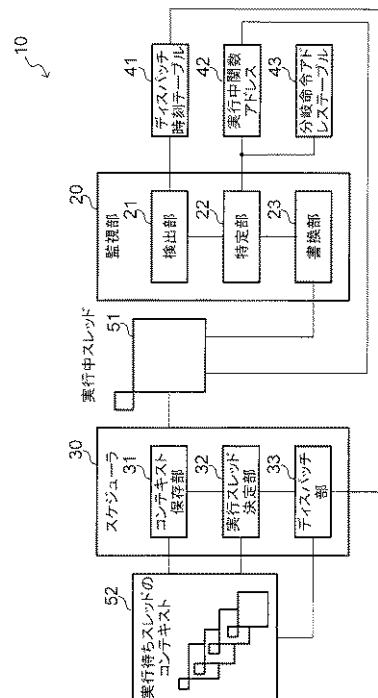
(54) 【発明の名称】 演算装置、スレッド切替方法、及びマルチスレッドプログラム

(57) 【要約】

【課題】 マルチスレッディングにおけるスレッドの切り替え時間を短縮する。

【解決手段】 マルチコアCPUが有する各コアで複数のスレッドを切り替えながらプログラムを実行するマルチスレッド処理において、検出部21が、実行中のスレッドの各々の実行状態に基づいて、実行待ちのスレッドへの切り替え対象となるスレッドを検出する。そして、特定部22が、検出部21により検出された切り替え対象となるスレッドで実行中又は過去に実行された命令に基づいて、切り替え対象となるスレッドで今後実行されることが推定される命令の位置情報を特定する。さらに、書換部23が、特定部22により特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるための命令に書き換える。

【選択図】 図3



【特許請求の範囲】**【請求項 1】**

マルチコアCPUが有する各コアで複数のスレッドを切り替えながらプログラムを実行するマルチスレッド処理において、実行中のスレッドの各々の実行状態に基づいて、実行待ちのスレッドへの切り替え対象となるスレッドを検出する検出部と、

前記検出部により検出された切り替え対象となるスレッドで実行中又は過去に実行された命令に基づいて、該切り替え対象となるスレッドで今後実行されることが推定される命令の位置情報を特定する特定部と、

前記特定部により特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるための命令に書き換える書換部と、

を含む演算装置。

10

【請求項 2】

前記特定部は、前記切り替え対象のスレッドで呼び出された関数に含まれる命令を、前記今後実行される命令として推定する請求項 1 記載の演算装置。

【請求項 3】

前記プログラムのコンパイル時に、関数の呼び出しを実行する前又は後に、呼び出す関数の位置情報が保存されるように設定された第 1 の記憶領域と、

前記プログラムのコンパイル時に取得された該プログラムに含まれる各関数に含まれる命令の位置情報が記憶された第 2 の記憶領域と、をさらに含み、

前記特定部は、前記第 1 の記憶領域を参照して、前記切り替え対象のスレッドで呼び出された関数を特定すると共に、前記第 2 の記憶領域を参照して、特定した前記関数に含まれる命令の位置情報を特定する

請求項 2 記載の演算装置。

20

【請求項 4】

前記第 2 の記憶領域には、前記各関数に含まれる命令のうち、分岐命令の位置情報が記憶される請求項 3 記載の演算装置。

【請求項 5】

前記特定部は、前記切り替え対象のスレッドで直近に実行された分岐命令の位置情報であって、前記切り替え対象のスレッドに割り当てられているコアで取得された分岐命令の位置情報を、前記今後実行されることが推定される命令の位置情報として特定する請求項 1 記載の演算装置。

30

【請求項 6】

前記検出部は、前記スレッドの各々の実行状態として、該スレッドがコアを占有している時間を取得し、前記コアを占有している時間が予め定めた閾値以上となるスレッドを、前記切り替え対象となるスレッドとして検出する請求項 1 ~ 請求項 5 のいずれか 1 項記載の演算装置。

【請求項 7】

実行されているスレッドの各々に対してコアが割り当てられた時刻が記憶された第 3 の記憶領域をさらに含み、

前記検出部は、前記第 3 の記憶領域に記憶された時刻を参照して、前記スレッドがコアを占有している時間を取得する

請求項 6 記載の演算装置。

40

【請求項 8】

前記書換部は、前記特定部により特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるためのスケジューラを呼び出す命令に書き換える請求項 1 ~ 請求項 7 のいずれか 1 項記載の演算装置。

【請求項 9】

マルチコアCPUが有する各コアで複数のスレッドを切り替えながらプログラムを実行するマルチスレッド処理において、実行中のスレッドの各々の実行状態に基づいて、実行待ちのスレッドへの切り替え対象となるスレッドを検出し、

50

検出された切り替え対象となるスレッドで実行中又は過去に実行された命令に基づいて、該切り替え対象となるスレッドで今後実行されることが推定される命令の位置情報を特定し、

特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるための命令に書き換える

ことを含む処理を演算装置に実行させるスレッド切替方法。

【請求項 10】

マルチコア CPU が有する各コアで複数のスレッドを切り替えながらプログラムを実行するマルチスレッド処理において、実行中のスレッドの各々の実行状態に基づいて、実行待ちのスレッドへの切り替え対象となるスレッドを検出し、

10

検出された切り替え対象となるスレッドで実行中又は過去に実行された命令に基づいて、該切り替え対象となるスレッドで今後実行されることが推定される命令の位置情報を特定し、

特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるための命令に書き換える

ことを含む処理を演算装置に実行させるためのマルチスレッドプログラム。

【発明の詳細な説明】

【技術分野】

【0001】

開示の技術は、演算装置、スレッド切替方法、及びマルチスレッドプログラムに関する

20

【背景技術】

【0002】

従来、複数のスレッドを切り替えながらプログラムを実行するマルチスレッディングに関する技術として、タイムアウトレジスタを有するマルチスレッド式プロセッサが提案されている。このマルチスレッド式プロセッサでは、アクティブ・スレッドの実行がプログラム可能な時間期間を超える時にスレッド切り替えが強制される。

【先行技術文献】

【特許文献】

【0003】

30

【特許文献 1】特表 2001 - 521216 号公報

【特許文献 2】特表 2001 - 521215 号公報

【発明の概要】

【発明が解決しようとする課題】

【0004】

マルチスレッディングにおいて、I/O 処理などにより暇になる CPU 資源を最大限活用するための手法であり、各スレッドあたりの処理が比較的少ない軽量スレッド (LWT : Light Weight Thread) を実行させる方法がある。LWT では、ハードウェアで同時に実行可能なスレッド数よりも遥かに多いスレッドを生成し、実行されるスレッドを切り換えながら高多重に処理を進めるため、スレッドの切り替え頻度が高い。

40

【0005】

従来技術のように、タイムアウト時に割り込みを発生させる手法は、スレッドの切り替えに時間がかかる。特に、スレッドの切り替え頻度が高い LWT では、プログラムの実行時間全体に対して、スレッドの切り替え時間が重い負荷となる。

【0006】

開示の技術は、一つの側面として、マルチスレッディングにおけるスレッドの切り替え時間を短縮することを目的とする。

【課題を解決するための手段】

【0007】

開示の技術は、一つの態様として、マルチコア CPU が有する各コアで複数のスレッド

50

を切り替えながらプログラムを実行するマルチスレッド処理に関する。開示の技術は、実行中のスレッドの各々の実行状態に基づいて、実行待ちのスレッドへの切り替え対象となるスレッドを検出する検出部を備える。また、開示の技術は、前記検出部により検出された切り替え対象となるスレッドで実行中又は過去に実行された命令に基づいて、該切り替え対象となるスレッドで今後実行されることが推定される命令の位置情報を特定する特定部を備える。さらに、開示の技術は、前記特定部により特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるための命令に書き換える書換部を備える。

【発明の効果】

【0008】

一つの側面として、マルチスレッディングにおけるスレッドの切り替え時間を短縮することができる、という効果を有する。

10

【図面の簡単な説明】

【0009】

【図1】マルチスレッディングにおけるスレッドの切り替えを説明するための図である。

【図2】割り込み処理を利用したスレッドの切り替えを説明するための図である。

【図3】第1実施形態に係るマルチスレッド処理装置の概略構成を示すブロック図である。

【図4】第1実施形態に係るマルチスレッド処理装置として機能するコンピュータの概略構成を示すブロック図である。

【図5】監視処理の一例を示すフローチャートである。

20

【図6】検出部での処理を説明するための図である。

【図7】特定部での処理を説明するための図である。

【図8】書換部での処理を説明するための図である。

【図9】スケジューラの呼び出しを説明するための図である。

【図10】切替処理の一例を示すフローチャートである。

【図11】ディスパッチ部での処理を説明するための図である。

【図12】第2実施形態に係るマルチスレッド処理装置の概略構成を示すブロック図である。

【図13】第2実施形態の変形例に係るマルチスレッド処理装置の概略構成を示すブロック図である。

【発明を実施するための形態】

30

【0010】

以下、図面を参照して、開示の技術に関する実施形態の一例を詳細に説明する。

【0011】

まず、実施形態の詳細を説明する前に、軽量スレッド(LWT: Light Weight Thread)によるプログラムの実行、及びLWTにおけるスレッド切り替えの問題点について説明する。

【0012】

例えば、マルチコアCPUによりLWTを実行させる場合、LWTでは、コア数よりも遥かに多いスレッドを生成し、各コアで実行されるスレッドを切り替えながらプログラムを実行する。例えば、図1に示すように、あるコアでスレッドAが実行されており、スレッドB、スレッドC、・・・が実行待ちであるとする。スレッドAにおいて、I/O処理が発生するなどして、コアがメモリやディスクでの処理待ち状態となる場合がある。この場合、このコアを有効活用するために、そのコアにおけるスレッドAの実行を停止し、そのコアを実行待ちのスレッドのいずれか(図1の例では、スレッドB)に割り当てる。

40

【0013】

ここで、例えば、スレッドAがあるコアを長時間占有して実行されている場合には、他の実行待ちのスレッドの実行が滞ってしまうため、強制的にスレッドを切り替える必要がある。このような場合のスレッドの切り替え方法としては、図2に示すように、タイマで設定された所定タイミングなどで割り込み信号を発生させ、強制的にスレッドを切り替える方法が考えられる。この方法では、例えば、以下の手順により、スレッドの切り替えが

50

実現される。

【 0 0 1 4 】

- 1 . 割り込み信号を受信したコアが、割り込みハンドラを呼び出し
- 2 . 割り込みハンドラが、スレッド A のコンテキストをコンテキスト保存領域に保存
- 3 . 割り込みハンドラが、スケジューラを呼び出すためのシグナルを生成
- 4 . 割り込みハンドラが、生成したシグナルを送出して、スケジューラを呼び出し
- 5 . スケジューラが、次に実行するスレッドを決定し (スレッド B)、ディスパッチャが、スレッド B のコンテキストをコンテキスト保存領域から読み出し
- 6 . ディスパッチャが、読み出したコンテキストをコアに設定

【 0 0 1 5 】

これにより、対象のコアにおいて、スレッド B が実行される。すなわち、対象のコアにおいて実行されるスレッドが、スレッド A からスレッド B に切り替わる。

【 0 0 1 6 】

しかし、割り込み処理によるスレッドの切り替えは、上記のように手順が多く、時間がかかる。LWT は、スレッド切り替え頻度が高いため、スレッドの切り替えに多くの時間を要すると、LWT により実行されるプログラム全体の性能が低下する。

【 0 0 1 7 】

そこで、以下の各実施形態では、コアを長時間占有して実行されているスレッドで今後実行される命令箇所を、スケジューラ呼出命令に書き換えることで、スレッド切り替えの手順を少なくし、スレッド切り替え時間の短縮を図る。以下、各実施形態について詳述する。以下の各実施形態では、マルチコア CPU を実装したコンピュータにおいて、LWT によるマルチスレッディングでアプリケーションプログラムを実行する場合に、開示の技術を適用した例について説明する。

【 0 0 1 8 】

< 第 1 実施形態 >

図 3 に示すように、第 1 実施形態に係るマルチスレッド処理装置 1 0 は、監視部 2 0 と、スケジューラ 3 0 とを含む。

【 0 0 1 9 】

スケジューラ 3 0 は、LWT 用に最適化されたスケジューラであり、実行されるスレッドと同じプロセス内でメモリを切り替えることなく呼び出せるユーザレベルのスケジューラである。スケジューラ 3 0 は、コンテキスト保存部 3 1 と、実行スレッド決定部 3 2 と、ディスパッチ部 3 3 とを含む。

【 0 0 2 0 】

コンテキスト保存部 3 1 は、スケジューラ 3 0 を呼び出した対象のコアで実行されているスレッド (実行中スレッド 5 1) のコンテキストを保存する。具体的には、コンテキスト保存部 3 1 は、対象のコアのレジスタに記憶されたデータをスタック領域に保存する。また、コンテキスト保存部 3 1 は、対象のコアのスタックポインタレジスタに記憶されたスタックポインタを専用の保存領域に保存する。

【 0 0 2 1 】

実行スレッド決定部 3 2 は、コンテキスト保存領域に保存された実行待ちのスレッドのコンテキスト 5 2 を参照して、次に実行するスレッドを決定する。具体的な決定方法は、実行されるアプリケーションプログラムによる。なお、コンテキスト保存領域とは、上記のスタック領域及びスタックポインタが保存された専用の保存領域である。

【 0 0 2 2 】

ディスパッチ部 3 3 は、実行スレッド決定部 3 2 で決定されたスレッドに対象のコアをディスパッチする。具体的には、ディスパッチ部 3 3 は、決定されたスレッドのコンテキストをコンテキスト保存領域から読み出し、対象のコアにセットする。より具体的には、ディスパッチ部 3 3 は、決定されたスレッドについて、専用の保存領域に記憶されたスタックポインタを読み出し、対象のコアのスタックポインタレジスタにセットする。また、ディスパッチ部 3 3 は、決定されたスレッドについてスタック領域に記憶されたデータを

10

20

30

40

50

読み出して、対象のコアのレジスタにセットする。

【 0 0 2 3 】

また、ディスパッチ部 3 3 は、現在実行中のスレッドに対してコアがディスパッチされた時刻を記録するディスパッチ時刻テーブル 4 1 に、決定されたスレッドに対象のコアをディスパッチした時刻を記録する。ディスパッチ時刻テーブル 4 1 は、メモリ領域の所定の記憶領域（第 3 の記憶領域）に保存される。

【 0 0 2 4 】

監視部 2 0 は、アプリケーションプログラムを実行するコアとは別に、専用のコアで実行される機能部である。監視部 2 0 は、検出部 2 1 と、特定部 2 2 と、書換部 2 3 とを含む。なお、監視部 2 0 は、開示の技術の演算装置の一例である。

10

【 0 0 2 5 】

検出部 2 1 は、ディスパッチ時刻テーブル 4 1 を常時又は十分短い周期で繰り返し参照し、ディスパッチ時刻テーブル 4 1 に記録された時刻と現在時刻との差が、規定時間以上となるスレッドを、長時間コアを占有しているスレッドとして検出する。検出部 2 1 は、検出したスレッドを、実行待ちの他のスレッドへ切り替える対象となる切り替え対象スレッドとして、特定部 2 2 に通知する。

【 0 0 2 6 】

特定部 2 2 は、検出部 2 1 から通知された切り替え対象スレッドで実行中又は過去に実行された命令に基づいて、切り替え対象スレッドで今後実行される命令を推定し、推定した命令が保存されているメモリ領域のアドレスを特定する。

20

【 0 0 2 7 】

ここで、実行中のアプリケーションプログラムでは、アプリケーションプログラムに含まれる関数の呼び出し命令の実行前又は実行後に、実行中関数アドレス 4 2 を第 1 の記憶領域に保存する。実行中関数アドレス 4 2 とは、実行中のスレッドにおいて直近で呼び出された関数が保存されているメモリ領域のアドレスである。アプリケーションプログラムのコンパイル時に、実行中関数アドレス 4 2 を第 1 の記憶領域に保存するように設定することができる。また、アプリケーションプログラムのコンパイル時に、アプリケーションプログラムに含まれる各関数に含まれる分岐命令が保存されているメモリ領域のアドレスが取得される。取得された分岐命令のアドレスは、その分岐命令を含む関数のアドレスに対応付けて、分岐命令アドレステーブル 4 3 として、メモリ領域の第 2 の記憶領域に保存される。

30

【 0 0 2 8 】

そこで、特定部 2 2 は、第 1 の記憶領域に保存された実行中関数アドレス 4 2 を参照して、切り替え対象スレッドで呼び出された関数のアドレスを特定する。また、特定部 2 2 は、第 2 の記憶領域に保存された分岐命令アドレステーブル 4 3 を参照して、特定した関数に含まれる分岐命令のアドレスを特定する。特定部 2 2 は、特定した分岐命令のアドレスを、書換部 2 3 に通知する。

【 0 0 2 9 】

なお、切り替え対象スレッドで今後実行される命令として分岐命令を特定するのは、分岐命令により処理がループしていることにより、切り替え対象スレッドがコアを長時間占有する結果になっているとの考えに基づく。分岐命令により処理がループしている場合には、再びその分岐命令が実行される可能性が高いため、今後実行される命令として、分岐命令を推定するものである。ただし、特定した関数に含まれる命令であれば、切り替え対象スレッドで今後実行される可能性はあるため、分岐命令以外の命令を切り替え対象スレッドで今後実行される命令として特定してもよい。また、上記のように、処理のループを想定して切り替え対象スレッドで今後実行される命令を推定するため、分岐命令の中でもループエンドに限定して特定するようにしてもよい。

40

【 0 0 3 0 】

書換部 2 3 は、特定部 2 2 により特定された分岐命令のアドレスに基づいて、メモリ領域において特定された分岐命令が保存されている箇所を、スケジューラ 3 0 の呼び出し命

50

令に書き換える。特定部 2 2 で複数の分岐命令のアドレスが特定されている場合には、書換部 2 3 は、特定された全ての分岐命令を書き換えてもよいし、特定された一部の分岐命令を書き換えてもよい。一部の分岐命令を書き換える場合は、例えば、特定された分岐命令のうち、実行順が後ろの方から順に所定個の分岐命令を書き換えることができる。実行順が前の方の分岐命令は、既に実行中のスレッドにおいて実行済みである可能性があり、実行順が後ろの方が、今後実行される分岐命令である可能性が高いからである。

【 0 0 3 1 】

マルチスレッド処理装置 1 0 は、例えば、図 4 に示すコンピュータ 6 0 で実現することができる。コンピュータ 6 0 は、複数のコア 6 1 0、6 1 1、・・・を有する CPU 6 1、一時記憶領域としてのメモリ 6 2、及び不揮発性の記憶部 6 3 を備える。また、コンピュータ 6 0 は、表示装置及び入力装置等の入出力装置 6 8 が接続される入出力インターフェース (I / F) 6 4 を備える。また、コンピュータ 6 0 は、記録媒体 6 9 に対するデータの読み込みと書き込みとを制御する read / write (R / W) 部 6 5、及びインターネット等のネットワークに接続されるネットワーク I / F 6 6 を備える。CPU 6 1、メモリ 6 2、記憶部 6 3、入出力 I / F 6 4、R / W 部 6 5、及びネットワーク I / F 6 6 は、バス 6 7 を介して互いに接続される。

10

【 0 0 3 2 】

記憶部 6 3 は、HDD (Hard Disk Drive)、SSD (solid state drive)、フラッシュメモリ等によって実現できる。記憶媒体としての記憶部 6 3 には、コンピュータ 6 0 をマルチスレッド処理装置 1 0 として機能させるためのマルチスレッドプログラム 7 0 が記憶される。

20

【 0 0 3 3 】

マルチスレッドプログラム 7 0 は、CPU 6 1 のいずれかのコアを監視部 2 0 として機能させるための監視プログラム 7 1 を含む。また、マルチスレッドプログラム 7 0 は、CPU 6 1 のいずれかのコアをスケジューラ 3 0 として機能させるための切替プログラム 7 2 と、アプリケーションプログラム 7 3 とを含む。本実施形態では、監視部 2 0 として機能するコアをコア 6 1 0、アプリケーションプログラムを実行し、かつスケジューラ 3 0 として機能するコアを 6 1 1 とする。

【 0 0 3 4 】

コア 6 1 0 は、監視プログラム 7 1 を記憶部 6 3 から読み出してメモリ 6 2 に展開し、監視プログラム 7 1 が有するプロセスを順次実行する。また、コア 6 1 1 は、アプリケーションプログラム 7 3 を記憶部 6 3 から読み出してメモリ 6 2 に展開し、アプリケーションプログラム 7 3 が有するプロセスを順次実行する。また、コア 6 1 1 は、切替プログラム 7 2 を記憶部 6 3 から読み出してメモリ 6 2 に展開し、アプリケーションプログラム 7 3 が有するプロセスを順次実行する。

30

【 0 0 3 5 】

監視プログラム 7 1 は、検出プロセス 7 1 A と、特定プロセス 7 1 B と、書換プロセス 7 1 C とを有する。コア 6 1 0 は、検出プロセス 7 1 A を実行することで、図 3 に示す検出部 2 1 として動作する。また、コア 6 1 0 は、特定プロセス 7 1 B を実行することで、図 3 に示す特定部 2 2 として動作する。また、コア 6 1 0 は、書換プロセス 7 1 C を実行することで、図 3 に示す書換部 2 3 として動作する。

40

【 0 0 3 6 】

切替プログラム 7 2 は、コンテキスト保存プロセス 7 2 A と、実行スレッド決定プロセス 7 2 B と、ディスパッチプロセス 7 2 C とを有する。コア 6 1 1 は、コンテキスト保存プロセス 7 2 A を実行することで、図 3 に示すコンテキスト保存部 3 1 として動作する。また、コア 6 1 1 は、実行スレッド決定プロセス 7 2 B を実行することで、図 3 に示す実行スレッド決定部 3 2 として動作する。また、コア 6 1 1 は、ディスパッチプロセス 7 2 C を実行することで、図 3 に示すディスパッチ部 3 3 として動作する。

【 0 0 3 7 】

また、上述した実行待ちスレッドのコンテキスト 5 2 を保存するコンテキスト保存領域

50

がメモリ 6 2 に設けられる。また、ディスパッチ時刻テーブル 4 1 が保存される第 3 の記憶領域、実行中関数アドレス 4 2 が保存される第 1 の記憶領域、及び分岐命令アドレステーブル 4 3 が保存される第 2 の記憶領域もメモリ 6 2 に設けられる。

【 0 0 3 8 】

これにより、マルチスレッドプログラム 7 0 を実行したコンピュータ 6 0 が、マルチスレッド処理装置 1 0 として機能することになる。

【 0 0 3 9 】

なお、マルチスレッドプログラム 7 0 により実現される機能は、例えば半導体集積回路、より詳しくは A S I C (Application Specific Integrated Circuit) 等で実現することも可能である。

【 0 0 4 0 】

次に、第 1 実施形態に係るマルチスレッド処理装置 1 0 の作用について説明する。マルチスレッド処理装置 1 0 において、アプリケーションプログラム 7 3 の実行が開始されると、監視部 2 0 が図 5 に示す監視処理を実行する。なお、監視処理は、開示の技術のスレッド切替方法の一例である。

【 0 0 4 1 】

図 5 に示す監視処理のステップ S 1 1 で、検出部 2 1 は、図 6 に示すように、第 3 の記憶領域 6 2 3 に記憶されたディスパッチ時刻テーブル 4 1 を常時又は十分短い周期で繰り返し参照する。なお、ディスパッチ時刻テーブル 4 1 には、後述する切替処理において、スケジューラ 3 0 のディスパッチ部 3 3 により、実行中のスレッドにコアがディスパッチされた時刻が保存されている。また、図 6 に示すディスパッチ時刻テーブル 4 1 では、「コア」、「スレッド」、及び「ディスパッチ時刻」の項目を含む例を示しているが、この例に限定されず、実行中のスレッドにコアがディスパッチされてからの時間を把握可能な形態であればよい。検出部 2 1 は、ディスパッチ時刻テーブル 4 1 に記録されたディスパッチ時刻と現在時刻との差、すなわち、各スレッドにコアがディスパッチされてからの時間を取得する。

【 0 0 4 2 】

次に、ステップ S 1 2 で、検出部 2 1 が、コアを長時間占有しているスレッドが存在するか否かを判定する。上記ステップ S 1 1 で取得した時間が予め定めた規定時間以上となるスレッドが存在する場合には、検出部 2 1 が、該当のスレッドを切り替え対象スレッドとして検出し、特定部 2 2 に通知して、処理はステップ S 1 3 へ移行する。コアを長時間占有しているスレッドが存在しない場合には、処理はステップ S 1 1 に戻る。

【 0 0 4 3 】

ステップ S 1 3 では、特定部 2 2 が、図 7 に示すように、第 1 の記憶領域 6 2 1 に保存された実行中関数アドレス 4 2 を参照して、検出部 2 1 から通知された切り替え対象スレッドで呼び出された関数のアドレスを取得する。第 1 の記憶領域 6 2 1 には、アプリケーションプログラム 7 3 のコンパイル時に、実行中のスレッドにおいて直近で呼び出された関数のアドレスが保存されるよう設定されている。

【 0 0 4 4 】

次に、ステップ S 1 4 で、特定部 2 2 が、第 2 の記憶領域 6 2 2 に保存された分岐命令アドレステーブル 4 3 において、上記ステップ S 1 3 で取得した関数のアドレスに対応付けられている分岐命令のアドレスを全て取得する。特定部 2 2 は、取得した分岐命令のアドレスを、書換部 2 3 に通知する。

【 0 0 4 5 】

次に、ステップ S 1 5 で、書換部 2 3 が、図 8 に示すように、特定部 2 2 から通知された分岐命令のアドレスに基づいて、メモリ領域において特定された分岐命令が保存されている箇所を、スケジューラ 3 0 の呼び出し命令に書き換える。そして、監視処理はステップ S 1 1 に戻る。

【 0 0 4 6 】

切り替え対象スレッドにおいて、上記監視処理のステップ S 1 5 で書き換えられたスケ

10

20

30

40

50

ジューラ 30 の呼び出し命令が実行されると、図 9 に示すように、実行中のスレッドから、直接スケジューラ 30 が呼び出される。そして、スケジューラ 30 が、図 10 に示す切替処理を実行する。

【 0 0 4 7 】

図 10 に示す切替処理のステップ S 2 1 で、コンテキスト保存部 3 1 が、切り替え対象スレッドを実行している対象のコアのレジスタに記憶されたデータをスタック領域に保存する。次に、ステップ S 2 2 で、コンテキスト保存部 3 1 が、対象のコアのスタックポインタレジスタに記憶されたスタックポインタを専用の保存領域に保存する。

【 0 0 4 8 】

次に、ステップ S 2 3 で、実行スレッド決定部 3 2 が、コンテキスト保存領域に保存された実行待ちのスレッドのコンテキスト 5 2 を参照して、次に実行するスレッドを決定する。

10

【 0 0 4 9 】

次に、ステップ S 2 4 で、ディスパッチ部 3 3 が、実行スレッド決定部 3 2 で決定されたスレッドについて、専用の保存領域に記憶されたスタックポインタを読み出し、対象のコアのスタックポインタレジスタにセットする。次に、ステップ S 2 5 で、ディスパッチ部 3 3 が、決定されたスレッドについてスタック領域に記憶されたデータを読み出して、対象のコアのレジスタにセットする。

【 0 0 5 0 】

次に、ステップ S 2 6 で、ディスパッチ部 3 3 が、図 11 に示すように、第 3 の記憶領域 6 2 3 に記憶されたディスパッチ時刻テーブル 4 1 に、決定されたスレッドに対象のコアをディスパッチした時刻として、現在時刻を記録する。そして、切替処理は終了する。

20

【 0 0 5 1 】

以上説明したように、第 1 実施形態に係るマルチスレッド処理装置 10 によれば、長時間コアを占有しているスレッドで今後実行される命令箇所を、スケジューラ呼び出し命令に書き換える。そして、書き換えられた命令が実行されることにより、切り替え対象のスレッドから直接スケジューラを呼び出すことができる。これにより、割り込み処理によりスレッドの切り替えを実行する場合に比べ、スレッド切り替えのための処理が少ないため、マルチスレッディングにおけるスレッド切り替え時間を短縮することができる。

【 0 0 5 2 】

30

< 第 2 実施形態 >

次に、第 2 実施形態について説明する。第 2 実施形態では、切り替え対象スレッドで今後実行される命令の特定方法が、第 1 実施形態と異なる。なお、第 2 実施形態に係るマルチスレッド処理装置について、第 1 実施形態に係るマルチスレッド処理装置 10 と同様の部分については、同一符号を付して、詳細な説明を省略する。

【 0 0 5 3 】

図 12 に示すように、第 2 実施形態に係るマルチスレッド処理装置 2 1 0 は、監視部 2 2 0 と、スケジューラ 3 0 とを含む。監視部 2 2 0 は、検出部 2 1 と、特定部 2 2 2 と、書換部 2 3 とを含む。

【 0 0 5 4 】

40

ここで、第 2 実施形態におけるマルチスレッド処理装置 2 1 0 では、マルチコア CPU が有する各コアは、図 12 に示すように、ハードウェア構成として、LBR (Last Branch Record) 用レジスタ 9 1 と、PEBS (Precise Event Based Sampling) 機能部 9 2 と、PEBS 用カウンタ 9 3 とを含む。LBR 用レジスタ 9 1 には、該当のコアで実行中のスレッドで直近に実行された分岐命令のアドレスが、LBR 機能により取得され、保存される。PEBS 機能部 9 2 は、ハードウェア領域に記憶された情報を取り出し、ソフトウェア領域で参照可能にする機能部である。PEBS 用カウンタ 9 3 は、PEBS 機能部 9 2 で取り出すデータを指定する情報 (ここでは、LBR 用レジスタ 9 1 を指定する情報) が保存される。この LBR 機能と PEBS 機能との連携により、該当のコアで実行中のスレッドで直近に実行された分岐命令のアドレスが定期的に取り得られ、LBR 情報 4 4 とし

50

て、メモリ領域の所定領域（第４の記憶領域）に保存される。

【００５５】

そこで、特定部２２２は、第４の記憶領域に保存されたＬＢＲ情報４４を参照して、ＬＢＲ情報が示すアドレスを、切り替え対象スレッドで今後実行される命令のアドレスとして特定する。

【００５６】

マルチスレッド処理装置２１０は、例えば、図４に示すコンピュータ２６０で実現することができる。コンピュータ２６０は、複数のコア２６１０、２６１１、・・・を有するＣＰＵ２６１、メモリ６２、及び記憶部６３を備える。また、コンピュータ２６０は、入出力装置６８が接続される入出力Ｉ／Ｆ６４、記録媒体６９に対するデータの読み込みと書き込みとを制御するＲ／Ｗ部６５、及びネットワークＩ／Ｆ６６を備える。ＣＰＵ２６１、メモリ６２、記憶部６３、入出力Ｉ／Ｆ６４、Ｒ／Ｗ部６５、及びネットワークＩ／Ｆ６６は、バス６７を介して互いに接続される。なお、コア２６１０、２６１１、・・・の各々は、上述したように、ＬＢＲ用レジスタ９１と、ＰＥＢＳ機能部９２と、ＰＥＢＳ用カウンタ９３とを含む。本実施形態では、監視部２２０として機能するコアをコア２６１０、アプリケーションプログラムを実行し、かつスケジューラ３０として機能するコアを２６１１とする。

10

【００５７】

記憶部６３には、コンピュータ２６０をマルチスレッド処理装置２１０として機能させるためのマルチスレッドプログラム２７０が記憶される。マルチスレッドプログラム２７０は、監視プログラム２７１と、切替プログラム２７２と、アプリケーションプログラム２７３とを含む。監視プログラム２７１は、検出プロセス２７１Ａと、特定プロセス２７１Ｂと、書換プロセス２７１Ｃとを有する。コア２６１０は、特定プロセス２７１Ｂを実行することで、図１２に示す特定部２２２として動作する。その他のプロセスについては、第１実施形態におけるマルチスレッドプログラム２７０と同様である。

20

【００５８】

また、上述したＬＢＲ情報４４が保存される第４の記憶領域がメモリ６２に設けられる。これにより、マルチスレッドプログラム２７０を実行したコンピュータ２６０が、マルチスレッド処理装置２１０として機能することになる。

【００５９】

なお、マルチスレッドプログラム２７０により実現される機能は、例えば半導体集積回路、より詳しくはＡＳＩＣ等で実現することも可能である。

30

【００６０】

第２実施形態に係るマルチスレッド処理装置２１０の作用は、切り替え対象スレッドで今後実行される命令の特定方法が第１実施形態と異なるだけである。具体的には、第２実施形態では、第１実施形態における監視処理（図５）のステップＳ１３及びＳ１４において、特定部２２２が、第４の記憶領域に保存されたＬＢＲ情報４４を参照して、ＬＢＲ情報が示すアドレスを、切り替え対象スレッドで今後実行される命令のアドレスとして特定する。

【００６１】

以上説明したように、第２実施形態に係るマルチスレッド処理装置２１０では、ハードウェア構成が有する既存の機能を利用して、切り替え対象スレッドで今後実行される命令を特定することで、第１実施形態と同様の効果を得られる。

40

【００６２】

第２実施形態の変形例として、図１３に、ハードウェア構成が有する機能を利用した他の例を示す。

【００６３】

図１３に示すマルチスレッド処理装置２１０Ａの監視部２２０Ａは、検出部２１と、特定部２２２Ａと、書換部２３とを含む。

【００６４】

50

また、スレッドを実行するコア 2661A は、ハードウェア構成として、LBR用レジスタ 91 を含む。また、監視部 220A として機能するコア 2660A は、ハードウェア構成として、他のコアが有するレジスタに保存されたデータを参照する参照部 95 を含む。参照部 95 は、本変形例では、スレッドを実行中のコア 2661A 内の LBR用レジスタ 91 を参照する。

【0065】

特定部 222A は、検出部 21 から切り替え対象スレッドを通知されると、参照部 95 を機能させ、切り替え対象スレッドを実行しているコア 2661A 内の LBR用レジスタ 91 を参照し、LBR情報 44 を取得する。他の処理については、第 2 実施形態と同様である。

10

【0066】

この変形例によれば、切り替え対象スレッドを実行しているコア 2661A 内の LBR用レジスタ 91 を直接参照して、今後実行される命令を特定することができるため、LBR情報 44 を保存するための記憶領域をメモリ領域に確保する必要がない。

【0067】

なお、上記では、マルチスレッドプログラム 70、270 が記憶部 63 に予め記憶（インストール）されている態様を説明したが、これに限定されない。開示の技術に係るプログラムは、CD-ROM、DVD-ROM、USBメモリ等の記録媒体に記録された形態で提供することも可能である。

【0068】

以上の各実施形態に関し、更に以下の付記を開示する。

20

【0069】

（付記 1）

マルチコア CPU が有する各コアで複数のスレッドを切り替えながらプログラムを実行するマルチスレッド処理において、実行中のスレッドの各々の実行状態に基づいて、実行待ちのスレッドへの切り替え対象となるスレッドを検出する検出部と、

前記検出部により検出された切り替え対象となるスレッドで実行中又は過去に実行された命令に基づいて、該切り替え対象となるスレッドで今後実行されることが推定される命令の位置情報を特定する特定部と、

前記特定部により特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるための命令に書き換える書換部と、
を含む演算装置。

30

【0070】

（付記 2）

前記特定部は、前記切り替え対象のスレッドで呼び出された関数に含まれる命令を、前記今後実行される命令として推定する付記 1 記載の演算装置。

【0071】

（付記 3）

前記プログラムのコンパイル時に、関数の呼び出しを実行する前又は後に、呼び出す関数の位置情報が保存されるように設定された第 1 の記憶領域と、

40

前記プログラムのコンパイル時に取得された該プログラムに含まれる各関数に含まれる命令の位置情報が記憶された第 2 の記憶領域と、をさらに含み、

前記特定部は、前記第 1 の記憶領域を参照して、前記切り替え対象のスレッドで呼び出された関数を特定すると共に、前記第 2 の記憶領域を参照して、特定した前記関数に含まれる命令の位置情報を特定する

付記 2 記載の演算装置。

【0072】

（付記 4）

前記第 2 の記憶領域には、前記各関数に含まれる命令のうち、分岐命令の位置情報が記憶される付記 3 記載の演算装置。

50

【 0 0 7 3 】

(付 記 5)

前記特定部は、前記切り替え対象のスレッドで直近に実行された分岐命令の位置情報であって、前記切り替え対象のスレッドに割り当てられているコアで取得された分岐命令の位置情報を、前記今後実行されることが推定される命令の位置情報として特定する付記 1 記載の演算装置。

【 0 0 7 4 】

(付 記 6)

前記特定部は、前記切り替え対象のスレッドに割り当てられているコアにおいて、Last Branch Record、及びPrecise Event Based Sampling機能を利用して取得された分岐命令の位置情報が記憶された第 4 の記憶領域を参照して、前記今後実行されることが推定される命令の位置情報を特定する付記 5 記載の演算装置。

10

【 0 0 7 5 】

(付 記 7)

前記検出部は、前記スレッドの各々の実行状態として、該スレッドがコアを占有している時間を取得し、前記コアを占有している時間が予め定めた閾値以上となるスレッドを、前記切り替え対象となるスレッドとして検出する付記 1 ~ 付記 6 のいずれか 1 項記載の演算装置。

【 0 0 7 6 】

(付 記 8)

実行されているスレッドの各々に対してコアが割り当てられた時刻が記憶された第 3 の記憶領域をさらに含み、

20

前記検出部は、前記第 3 の記憶領域に記憶された時刻を参照して、前記スレッドがコアを占有している時間を取得する

付記 7 記載の演算装置。

【 0 0 7 7 】

(付 記 9)

前記書換部は、前記特定部により特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるためのスケジューラを呼び出す命令に書き換える付記 1 ~ 付記 8 のいずれか 1 項記載の演算装置。

30

【 0 0 7 8 】

(付 記 1 0)

付記 1 ~ 付記 9 のいずれか 1 項記載の演算装置と、

前記書換部により書き換えられた命令により呼び出される前記スケジューラと、

を含むマルチスレッド処理装置。

【 0 0 7 9 】

(付 記 1 1)

前記スケジューラは、実行されているスレッドの各々に対してコアが割り当てられた時刻を、前記第 3 の記憶領域に記録する記録部を含む付記 1 0 記載のマルチスレッド処理装置。

40

【 0 0 8 0 】

(付 記 1 2)

マルチコア CPU が有する各コアで複数のスレッドを切り替えながらプログラムを実行するマルチスレッド処理において、実行中のスレッドの各々の実行状態に基づいて、実行待ちのスレッドへの切り替え対象となるスレッドを検出し、

検出された切り替え対象となるスレッドで実行中又は過去に実行された命令に基づいて、該切り替え対象となるスレッドで今後実行されることが推定される命令の位置情報を特定し、

特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるための命令に書き換える

50

ことを含む処理を演算装置に実行させるスレッド切替方法。

【 0 0 8 1 】

(付 記 1 3)

前記切り替え対象のスレッドで呼び出された関数に含まれる命令を、前記今後実行される命令として推定する付記 1 2 記載のスレッド切替方法。

【 0 0 8 2 】

(付 記 1 4)

前記今後実行されることが推定される命令の位置情報を特定する際に、前記プログラムのコンパイル時に、関数の呼び出しを実行する前又は後に、呼び出す関数の位置情報が保存されるように設定された第 1 の記憶領域を参照して、前記切り替え対象のスレッドで呼び出された関数を特定し、

10

前記プログラムのコンパイル時に取得された該プログラムに含まれる各関数に含まれる命令の位置情報が記憶された第 2 の記憶領域を参照して、特定した前記関数に含まれる命令の位置情報を特定する

付記 1 3 記載のスレッド切替方法。

【 0 0 8 3 】

(付 記 1 5)

前記第 2 の記憶領域には、前記各関数に含まれる命令のうち、分岐命令の位置情報が記憶される付記 1 4 記載のスレッド切替方法。

【 0 0 8 4 】

(付 記 1 6)

前記切り替え対象のスレッドで直近に実行された分岐命令の位置情報であって、前記切り替え対象のスレッドに割り当てられているコアで取得された分岐命令の位置情報を、前記今後実行されることが推定される命令の位置情報として特定する付記 1 2 記載のスレッド切替方法。

20

【 0 0 8 5 】

(付 記 1 7)

前記今後実行されることが推定される命令の位置情報を特定する際、前記切り替え対象のスレッドに割り当てられているコアにおいて、Last Branch Record、及びPrecise Event Based Sampling機能を利用して取得された分岐命令の位置情報が記憶された第 4 の記憶領域を参照して、前記今後実行されることが推定される命令の位置情報を特定する付記 1 6 記載のスレッド切替方法。

30

【 0 0 8 6 】

(付 記 1 8)

前記スレッドの各々の実行状態として、該スレッドがコアを占有している時間を取得し、前記コアを占有している時間が予め定めた閾値以上となるスレッドを、前記切り替え対象となるスレッドとして検出する付記 1 2 ~ 付記 1 7 のいずれか 1 項記載のスレッド切替方法。

【 0 0 8 7 】

(付 記 1 9)

前記切り替え対象となるスレッドを検出する際、実行されているスレッドの各々に対してコアが割り当てられた時刻が記憶された第 3 の記憶領域に記憶された時刻を参照して、前記スレッドがコアを占有している時間を取得する付記 1 8 記載のスレッド切替方法。

40

【 0 0 8 8 】

(付 記 2 0)

特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるためのスケジューラを呼び出す命令に書き換える付記 1 2 ~ 付記 1 9 のいずれか 1 項記載のスレッド切替方法。

【 0 0 8 9 】

(付 記 2 1)

50

マルチコアCPUが有する各コアで複数のスレッドを切り替えながらプログラムを実行するマルチスレッド処理において、実行中のスレッドの各々の実行状態に基づいて、実行待ちのスレッドへの切り替え対象となるスレッドを検出し、

検出された切り替え対象となるスレッドで実行中又は過去に実行された命令に基づいて、該切り替え対象となるスレッドで今後実行されることが推定される命令の位置情報を特定し、

特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるための命令に書き換える

ことを含む処理を演算装置に実行させるためのマルチスレッドプログラム。

【0090】

10

(付記22)

前記切り替え対象のスレッドで呼び出された関数に含まれる命令を、前記今後実行される命令として推定する付記21記載のマルチスレッドプログラム。

【0091】

(付記23)

前記今後実行されることが推定される命令の位置情報を特定する際に、

前記プログラムのコンパイル時に、関数の呼び出しを実行する前又は後に、呼び出す関数の位置情報が保存されるように設定された第1の記憶領域を参照して、前記切り替え対象のスレッドで呼び出された関数を特定し、

前記プログラムのコンパイル時に取得された該プログラムに含まれる各関数に含まれる命令の位置情報が記憶された第2の記憶領域を参照して、特定した前記関数に含まれる命令の位置情報を特定する

20

付記22記載のマルチスレッドプログラム。

【0092】

(付記24)

前記第2の記憶領域には、前記各関数に含まれる命令のうち、分岐命令の位置情報が記憶される付記23記載のマルチスレッドプログラム。

【0093】

(付記25)

前記切り替え対象のスレッドで直近に実行された分岐命令の位置情報であって、前記切り替え対象のスレッドに割り当てられているコアで取得された分岐命令の位置情報を、前記今後実行されることが推定される命令の位置情報として特定する付記21記載のマルチスレッドプログラム。

30

【0094】

(付記26)

前記今後実行されることが推定される命令の位置情報を特定する際、前記切り替え対象のスレッドに割り当てられているコアにおいて、Last Branch Record、及びPrecise Event Based Sampling機能を利用して取得された分岐命令の位置情報が記憶された第4の記憶領域を参照して、前記今後実行されることが推定される命令の位置情報を特定する付記25記載のマルチスレッドプログラム。

40

【0095】

(付記27)

前記スレッドの各々の実行状態として、該スレッドがコアを占有している時間を取得し、前記コアを占有している時間が予め定めた閾値以上となるスレッドを、前記切り替え対象となるスレッドとして検出する付記21～付記26のいずれか1項記載のマルチスレッドプログラム。

【0096】

(付記28)

前記切り替え対象となるスレッドを検出する際、実行されているスレッドの各々に対してコアが割り当てられた時刻が記憶された第3の記憶領域に記憶された時刻を参照して、

50

前記スレッドがコアを占有している時間を取得する付記 2 7 記載のマルチスレッドプログラム。

【 0 0 9 7 】

(付記 2 9)

特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるためのスケジューラを呼び出す命令に書き換える付記 2 1 ~ 付記 2 8 のいずれか 1 項記載のマルチスレッドプログラム。

【 0 0 9 8 】

(付記 3 0)

マルチコア CPU が有する各コアで複数のスレッドを切り替えながらプログラムを実行するマルチスレッド処理において、実行中のスレッドの各々の実行状態に基づいて、実行待ちのスレッドへの切り替え対象となるスレッドを検出し、

検出された切り替え対象となるスレッドで実行中又は過去に実行された命令に基づいて、該切り替え対象となるスレッドで今後実行されることが推定される命令の位置情報を特定し、

特定された命令の位置情報が示すプログラムの箇所を、スレッドを切り替えるための命令に書き換える

ことを含む処理を演算装置に実行させるためのマルチスレッドプログラムを記憶した記憶媒体。

【 符号の説明 】

【 0 0 9 9 】

1 0、2 1 0、2 1 0 A	マルチスレッド処理装置	
2 0、2 2 0、2 2 0 A	監視部	
2 1	検出部	
2 2、2 2 2、2 2 2 A	特定部	
2 3	書換部	
3 0	スケジューラ	
3 1	コンテキスト保存部	
3 2	実行スレッド決定部	
3 3	ディスパッチ部	
4 1	ディスパッチ時刻テーブル	
4 2	実行中関数アドレス	
4 3	分岐命令アドレステーブル	
4 4	L B R 情報	
5 1	実行中スレッド	
5 2	実行待ちスレッドのコンテキスト	
6 0、2 6 0	コンピュータ	
6 1、2 6 1	C P U	
6 1 0、6 1 1、2 6 6 0、2 6 6 1、2 6 6 0 A、2 6 6 1 A	コア	
6 2	メモリ	
6 3	記憶部	
7 0、2 7 0	マルチスレッドプログラム	
7 1、2 7 1	監視プログラム	
7 2	切替プログラム	
7 3	アプリケーションプログラム	
9 1	L B R 用レジスタ	
9 2	P E B S 機能部	
9 3	P E B S 用カウンタ	
9 5	参照部	
6 2 1	第 1 の記憶領域	

10

20

30

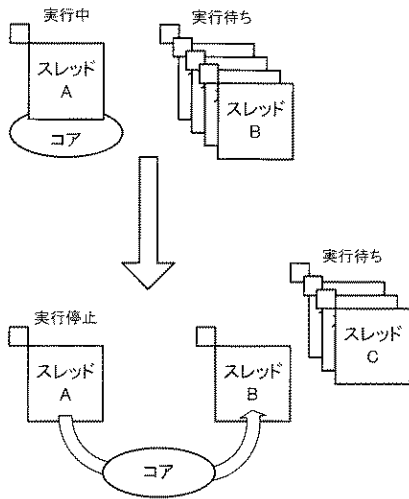
40

50

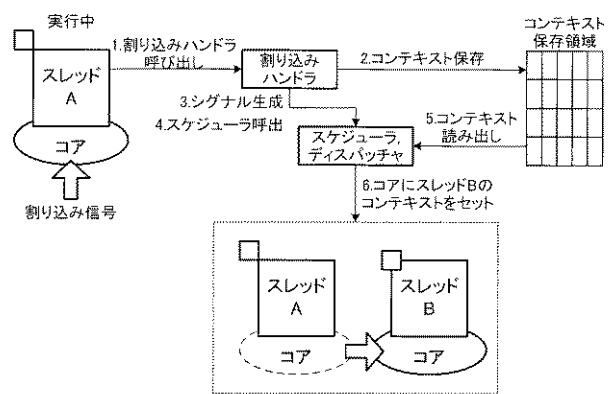
6 2 2 第 2 の 記 憶 領 域

6 2 3 第 3 の 記 憶 領 域

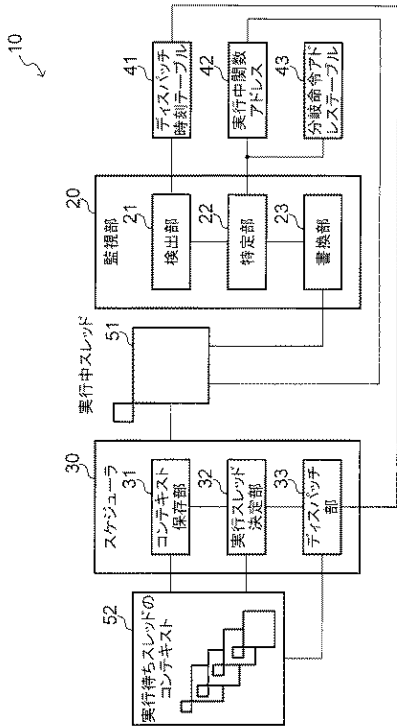
【 図 1 】



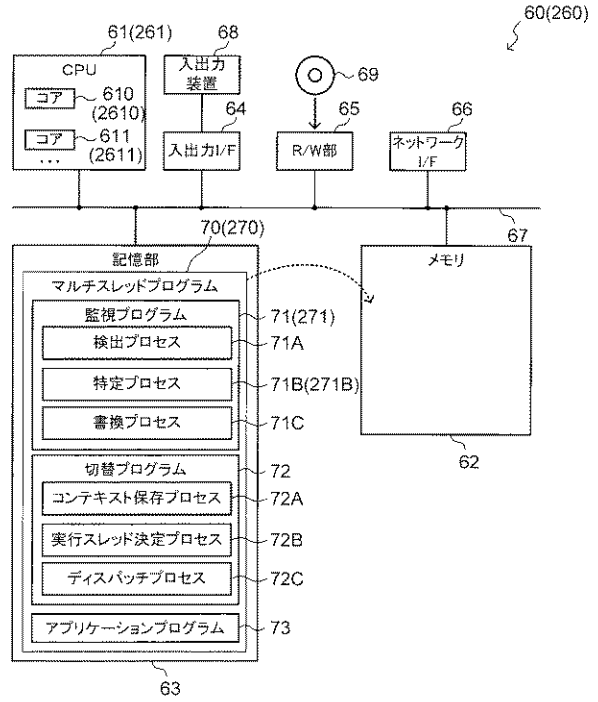
【 図 2 】



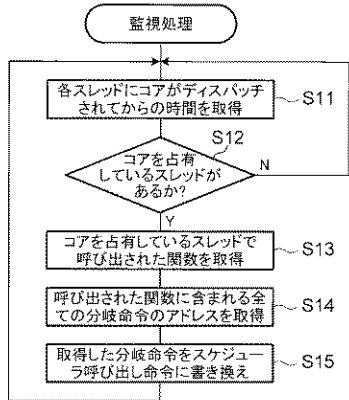
【図3】



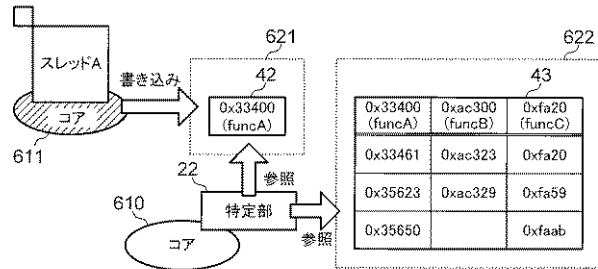
【図4】



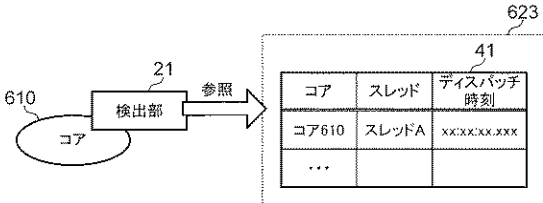
【図5】



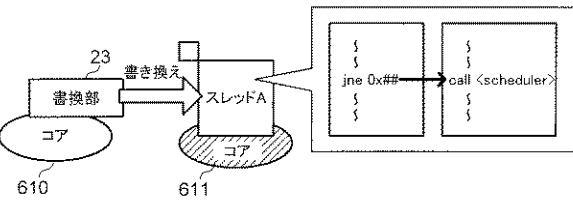
【図7】



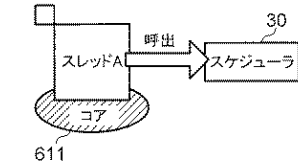
【図6】



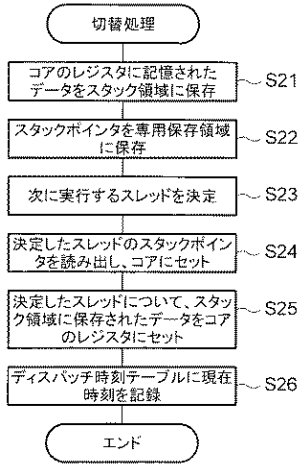
【図8】



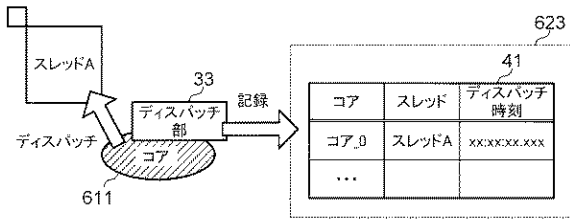
【図9】



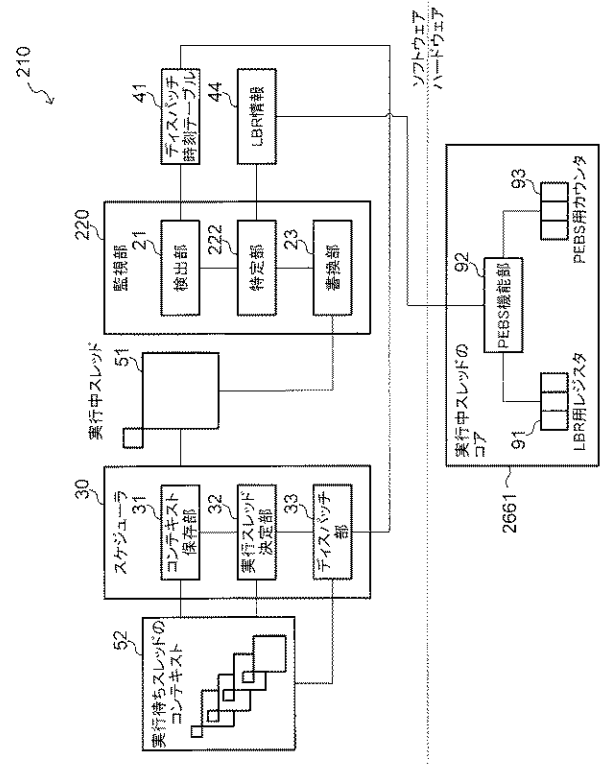
【図10】



【図11】



【図12】



【図13】

