(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5131269号
(P5131269)

(45)発行日　平成25年1月30日(2013.1.30)　(24)登録日　平成24年11月16日(2012.11.16)

(51)Int.Cl.　　F I
G06F 9/46 (2006.01)　　G06F 9/46 ３５０
G06F 9/48 (2006.01)　　G06F 9/46 ４５５Ｚ

請求項の数 6　(全 15 頁)

(21)出願番号　特願2009-505014 (P2009-505014)
(86)(22)出願日　平成19年3月20日 (2007.3.20)
(86)国際出願番号　PCT/JP2007/055670
(87)国際公開番号　WO2008/114415
(87)国際公開日　平成20年9月25日 (2008.9.25)
審査請求日　平成21年4月9日 (2009.4.9)

(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　100104190
弁理士　酒井　昭徳
(72)発明者　上方　輝彦
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

審査官　田中　幸雄

最終頁に続く

(54)【発明の名称】マルチプロセッシングシステム

(57)【特許請求の範囲】
【請求項１】
マスタＣＰＵと、
スレーブＣＰＵと、
タスクと当該タスクに対応したオペレーションシステム（以下、「ＯＳ」という）との組み合わせからなるＯＳタスクセットを複数記憶する記憶部と、
前記マスタＣＰＵが、前記スレーブＣＰＵの実行指示に応じて、前記記憶部に記憶されている複数のＯＳタスクセットのうち、前記実行指示の対象となった第１タスクと当該第１タスクに対応した第１ＯＳとの組み合わせからなる第１ＯＳタスクセットを、前記記憶部からメモリに格納する格納部と、
前記スレーブＣＰＵが、前記メモリに格納された前記第１ＯＳタスクセットを参照し、当該第１ＯＳタスクセットを構成する前記第１ＯＳをロードし、前記第１タスクを実行するタスク実行部と、
を備えることを特徴とするマルチプロセッシングシステム。
【請求項２】
前記マスタＣＰＵは、前記スレーブＣＰＵに実行させる前記第１タスクを変更する場合、前記メモリに格納した前記第１ＯＳタスクセットを、前記記憶部に記憶されている第２タスクと当該第２タスクに対応した第２ＯＳとを組み合わせた第２ＯＳタスクセットに入れ替えることを特徴とする請求項１に記載のマルチプロセッシングシステム。
【請求項３】

前記スレーブＣＰＵは、実行する前記第１または前記第２タスクに応じた前記第１または前記第２ＯＳタスクセットを格納した前記メモリの物理アドレスと、前記第１または前記第２ＯＳタスクセットの論理アドレスとの対応をあらわすメモリマップとを作成し、当該メモリマップを参照して前記論理アドレスを前記物理アドレスに変換するアドレス変換部を備え、

前記マスタＣＰＵからの前記実行指示に示された前記第１または前記第２ＯＳタスクセットの論理アドレスを、前記アドレス変換部によって物理アドレスに変換して、前記メモリから前記第１または前記第２ＯＳタスクセットを読み出すことを特徴とする請求項２に記載のマルチプロセッシングシステム。

【請求項４】

前記マスタＣＰＵは、前記第１または前記第２ＯＳタスクセットの格納場所が変更になると、前記アドレス変換部に、前記メモリマップにおける前記第１または前記第２ＯＳタスクセットの物理アドレスを更新させることを特徴とする請求項３に記載のマルチプロセッシングシステム。

【請求項５】

前記マスタＣＰＵは、前記スレーブＣＰＵが前記実行指示に応じて指定されたタスクの処理を開始すると、当該タスクのつぎに前記スレーブＣＰＵに処理させるタスクのＯＳタスクセットを構成するＯＳをロードすることを特徴とする請求項１～４のいずれか一つに記載のマルチプロセッシングシステム。

【請求項６】

前記マルチプロセッシングシステムは、ＡＭＰ（Ａｓｙｍｍｅｔｒｉｃａｌ Ｍｕｌｔｉ－Ｐｒｏｃｅｓｓｉｎｇ）システムもしくはＬＣＭＰ（Ｌｏｏｓｅｌｙ Ｃｏｕｐｌｅｄ Ｍｕｌｔｉ Ｐｒｏｃｅｓｓｏｒ）システムによって構成されることを特徴とする請求項１～５のいずれか１つに記載のマルチプロセッシングシステム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、静的ロードによって指定されたタスクを処理するマルチプロセッシングシステムに関する。

【背景技術】

【０００２】

従来より、組み込みシステムやシステムＬＳＩにおいて、１つのチップ上に複数のＣＰＵコアやＤＳＰコアを組み込んだシステムが設計されている。これらのシステムをマルチプロセッシングシステムといい、対称型マルチプロセッシング（Ｓｙｍｍｅｔｒｉｃ Ｍｕｌｔｉ－Ｐｒｏｃｅｓｓｉｎｇ：ＳＭＰ）システムと、非対称型マルチプロセッシング（Ａｓｙｍｍｅｔｒｉｃａｌ Ｍｕｌｔｉ－Ｐｒｏｃｅｓｓｉｎｇ：ＡＭＰ）システムとの２種類に大別される。

【０００３】

（ＳＭＰシステム）

図９は、ＳＭＰシステムの構成を示すブロック図である。ＳＭＰシステム９００では、基本的に複数のＣＰＵ９３０（ＣＰＵ０～ＣＰＵ３）が、それぞれ同等なものとして振舞うことができる。したがって、ＣＰＵ９３０がタスク９１０を処理する際には、それぞれ、同じプログラムコードを利用することができる。ＳＭＰシステム９００のＣＰＵ９３０がおこなう処理は、カーネルプロセスやユーザプロセスなどの区別なく、なおかつ、複数のＣＰＵ９３０を使って同時に実行させることもできる。

【０００４】

また、ＳＭＰシステム９００は、ＳＭＰシステムをサポートするＳＭＰ対応ＯＳ９２０が必要となる。さらに、ＳＭＰシステム９００において共有化されたプログラムコードは、マルチプロセッサであっても利用可能である必要がある。なお、ＳＭＰシステム９００では、シングルプロセッサのシステムで、割り込み禁止とタスク優先度に依存した排他制

御や実行順序制御をおこなうようなプログラムは正常に動作しなくなる可能性がある。

【０００５】

（ＡＭＰシステム）

図１０～１２は、ＡＭＰシステムの構成例１～３を示すブロック図である。ＡＭＰシステムとは、ＳＭＰシステムではないマルチプロセッシングシステムであり、ＯＳのカーネルを実行するＣＰＵ、ユーザアプリケーションのある機能を実行するＣＰＵというように、各ＣＰＵにはそれぞれの役割が決定されている。また、ＡＭＰシステムは、図１０～１２のように様々な構成例がある。

【０００６】

図１０のＡＭＰシステム１０００は、複数のＣＰＵ１０３０（ＣＰＵ０～ＣＰＵ３）を備えている。ＣＰＵ０～ＣＰＵ３は、同種であっても異種であってもよく、各ＣＰＵに、それぞれ利用するＯＳ１０２０（ＯＳ０～ＯＳ３）が設定されている。また、ＡＭＰシステム１０００の場合、上述したようにＣＰＵ１０３０ごとに役割が決められているため、処理対象のタスク１０１０も、ＣＰＵ０によって処理するタスクＡ（タスク１～３）、ＣＰＵ１によって処理するタスクＢ（タスク４～６）というようにＣＰＵ１０３０によって処理するタスク１０１０が割り振られている。このような場合、各ＣＰＵ１０３０は、割り振られたタスクにしかアクセスできない。

【０００７】

また、図１１のＡＭＰシステム１１００は、複数のＣＰＵ１１３０（ＣＰＵ０～ＣＰＵ３）を備えているが、各ＣＰＵ１１３０は同種類のＣＰＵとする。このように、複数のＣＰＵ１１３０が同種であれば、利用するＯＳは、同一の種類のＯＳ１１２０を利用することができる。したがって、同一のＯＳ１１２０を利用して各ＣＰＵ１１３０が割り振られたタスク１１１０を処理する。

【０００８】

また図１２のＡＭＰシステム１２００は、複数のＣＰＵ１２３０（ＣＰＵ０～ＣＰＵ３）を備えており、ＣＰＵ０とＣＰＵ０は同種のＣＰＵとする。このような場合、同種のＣＰＵ０とＣＰＵ１とは、共通のＯＳ１２２０（ＯＳ０）を利用してタスク１２１０の中のタスクＡを処理する。その他のＣＰＵ１２３０（ＣＰＵ２、ＣＰＵ３）は、それぞれ対応するＯＳ１２２０を利用して割り振られたタスク１２１０（タスクＢ、Ｃ）を処理する。

【０００９】

また、マルチプロセッシングシステムの構成方式には、たとえば、メインメモリを複数のＣＰＵによって共有し、クロスバースイッチなどによってメインメモリとＣＰＵとの間を密結合するＴＣＭＰ（Ｔｉｇｈｔｌｙ　Ｃｏｕｐｌｅｄ　Ｍｕｌｔｉ　Ｐｒｏｃｅｓｓｏｒ）システムや、メインメモリを複数のＣＰＵによって共有しても、共有しなくてもよく、メインメモリとＣＰＵとを疎結合するＬＣＭＰ（Ｌｏｏｓｅｌｙ　Ｃｏｕｐｌｅｄ　Ｍｕｌｔｉ　Ｐｒｏｃｅｓｓｏｒ）システムがある。

【００１０】

一般的に、組み込みシステムにおいて、ある既存のＯＳによって開発したソフトウェア資産／プログラム資産があると、新しいシステムに変更した場合であっても、ソフトウェア資産／プログラム資産を利用させたい。したがって、既存のＯＳによってプログラムをロードする必要がある。特に、新しいシステムがＡＭＰシステムの場合、スレーブＣＰＵで実行するプログラムは既存のプログラムをそのまま既存のＯＳで動作させたい場合がある。このような場合には、ソフトウェア資産／プログラム資産を利用する際に既存のＯＳをロードしてプログラムを実行させる。

【００１１】

既存のＯＳをロードしてプログラムを実行させる際、プログラムを静的にリンクする手法と、動的にリンクする手法とがある。静的にリンクするとは、プログラム作成時、プログラムをモジュールに分割し、コンパイルした後に、オブジェクトファイルを汎用ライブラリと一緒にリンクして実行可能形式のロードモジュール（バイナリ）を作成する手法である。また、動的なロードとは、プログラムの実行開始時、または、実行中にルーチンが

呼び出されたときに、初めて他のモジュールやライブラリと結合する手法である（たとえば、下記特許文献１参照。）。

【００１２】

静的にプログラムをロードする場合は、上述のように、スレーブＣＰＵでは、実行する可能性があるすべてのプログラムをロードしておく必要がある。一方、ＯＳがプログラムを動的にロードして、リンクして実行できる仮想記憶をサポートしたＯＳ（Ｗｉｎｄｏｗｓ（登録商標）やＵＮＩＸ（登録商標）やＬｉｎｕｘ等）の場合、プログラムのうち、複数のタスクから共有されるコードは、位置独立コード（ＰＩＣ）で記載されており、共有ライブラリ（ダイナミックリンクライブラリ）として、実行時に動的にリンクする。

【００１３】

【特許文献１】特開２００２－９９４３９号公報

【発明の開示】

【発明が解決しようとする課題】

【００１４】

しかしながら、上述したＡＭＰシステムやＬＣＭＰシステムにおいては、動的にプログラムをロードができない既存のＯＳを使用するものも多い。動的にプログラムをロードできない場合、スレーブＣＰＵは、あるＯＳ１をロードする際には、ＯＳ１を利用して実行させる可能性のあるすべてのプログラム（タスク）をあらかじめメインメモリにロードしておかなければならない。そのために、メインメモリは、他の処理に利用するためのメモリ領域が小さくなってしまう。したがって、プログラムのロードによって各ＣＰＵの処理速度の低下を招く恐れがあるという問題があった。

【００１５】

上述のような問題が生じた場合、一般的には、共有ライブラリ、ダイナミックリンクローダ（ＤＬＬ）といった技術が用いられている。この技術を実現するシステムは、ＯＳが仮想アドレス空間（Ｖｉｒｔｕａｌ　Ａｄｄｒｅｓｓ　Ｓｐａｃｅ）と実アドレス空間（Ｒｅａｌ　Ａｄｄｒｅｓｓ　Ｓｐａｃｅ）との対応をページ単位で管理するＭＭＵ（Ｐａｇｅｄ　Ｍｅｍｏｒｙ　Ｍａｎａｇｅｍｅｎｔ　Ｕｎｉｔ）を使用するため、仮想記憶（Ｖｉｒｔｕａｌ　Ｍｅｍｏｒｙ）の仕組みを備えている必要があった。

【００１６】

しかしながら、μＩＴＲＯＮ仕様のＯＳなどに代表される組み込みプロセッサ用のＯＳには、仮想記憶（Ｖｉｒｔｕａｌ　Ｍｅｍｏｒｙ）の仕組みを備えていないものが多々存在する。μＩＴＲＯＮ仕様のＯＳでは、通常、ＣＰＵ上で実行されるＯＳと、ＣＰＵ上で実行される「複数のタスクを構成する関数」は、静的にリンクされている。このため、ロードモジュールは絶対アドレスになっていることが多い。

【００１７】

上述のように、仮想記憶の仕組みを備えていない場合、アプリケーションプログラムは、そのタスク自身が使用する（関数呼び出しをする）すべてのライブラリを静的にリンクしなければならない。アプリケーションプログラムを動的にロードしようとすると、アプリケーションプログラムごとに、そのタスク自身が使用するすべてのライブラリを静的にリンクしなければならない。すなわち、Ｃライブラリなどの１つのアプリケーションプログラムごとに存在することになる。したがって、複数のアプリケーションプログラムを実行させようとすると、同じライブラリが、アプリケーションプログラムの数だけ、メインメモリ上のアドレス空間に存在しなければならず、メインメモリが多く必要となるという問題があった。

【００１８】

また、新しいＯＳの上で、すでに開発されているアプリケーションプログラムをＤＬＬに対応させるには、アプリケーションプログラムの変更と、ＯＳの変更が必要になり、個別にプログラムを開発しなければならず、開発コストがかってしまうという問題があった。

【００１９】

本発明は、上記に鑑みてなされたものであって、メモリに負担をかけることなく、タスクに応じてＯＳを切り替えて、適切にタスクを実行させることを目的とする。

【課題を解決するための手段】

【００２０】

上述した課題を解決し、目的を達成するために、本発明にかかるマルチプロセッシングシステムは、実行指示の対象となった第１タスクと当該第１タスクに対応した第１オペレーションシステム（以下、「ＯＳ」という）との組み合わせからなる第１ＯＳタスクセットをメモリに格納する格納部と、前記メモリに格納された前記第１ＯＳタスクセットを参照し、当該第１ＯＳタスクセットを構成するＯＳをロードし、前記実行指示の対象となった第１タスクを実行するタスク実行部とを備えることを特徴とする。

【００２１】

また、上記発明において、前記マルチプロセッシングシステムは、マスタＣＰＵとスレーブＣＰＵとを備え、前記格納部は前記マスタＣＰＵによって構成され、前記タスク実行部は、前記スレーブＣＰＵによって構成されてもよい。

【００２２】

これらの発明によれば、これから実行するタスクと、実行対象となるタスクを処理するために必要なデータのみがメモリにロードされる。

【００２３】

また、上記発明において、前記マスタＣＰＵは、前記スレーブＣＰＵに実行させる前記第１タスクを変更する場合、前記メモリに格納した前記第１ＯＳタスクセットを、第２タスクと当該第２タスクに対応したＯＳとを組み合わせた第２ＯＳタスクセットに入れ替えてもよい。

【００２４】

この発明によれば、タスク実行に必要ないデータは、メインメモリから除外される。

【００２５】

また、上記発明において、前記スレーブＣＰＵは、実行する前記第１または前記第２タスクに応じた前記第１または前記第２ＯＳタスクセットを格納した前記メモリの物理アドレスと、前記第１または前記第２ＯＳタスクセットの論理アドレスとの対応をあらわすメモリマップとを作成し、当該メモリマップを参照して前記論理アドレスを前記物理アドレスに変換するアドレス変換部を備え、前記マスタＣＰＵからの前記実行指示に示された前記第１または前記第２ＯＳタスクセットの論理アドレスを、前記アドレス変換部によって物理アドレスに変換して、前記メモリから前記第１または前記第２ＯＳタスクセットを読み出てもよい。

【００２６】

さらに上記発明において、前記マスタＣＰＵは、前記第１または前記第２ＯＳタスクセットの格納場所が変更になると、前記アドレス変換部に、前記メモリマップにおける前記第１または前記第２ＯＳタスクセットの物理アドレスを更新させてもよい。

【００２７】

この発明によれば、タスクの実行に必要な情報を高速にメモリにロードすることができる。

【００２８】

また、上記発明において、前記マスタＣＰＵは、前記スレーブＣＰＵが前記実行指示に応じて指定されたタスクの処理を開始すると、当該タスクのつぎに前記スレーブＣＰＵに処理させるタスクのＯＳタスクセットを構成するＯＳをロードしてもよい。

【００２９】

この発明によれば、スレーブＣＰＵが連続してタスクの実行する際の待機時間を短縮させることができる。

【００３０】

また、上記発明のマルチプロセッシングシステムは、ＡＭＰシステムもしくはＬＣＭＰシステムによって構成されてもよい。

【発明の効果】
【００３１】
本発明にかかるマルチプロセッシングシステムは、メモリに負担をかけることなく、タスクに応じてＯＳを切り替えることによって適切にタスクを実行させることができるという効果を奏する。
【図面の簡単な説明】
【００３２】
【図１】図１は、本実施の形態にかかるマルチプロセッシングシステムにおけるメインメモリの利用形態を示す説明図である。
【図２】図２は、本実施の形態にかかるマルチプロセッシングシステムのハードウェアの構成の一例を示すブロック図である。
【図３】図３は、マルチプロセッシングシステムのタスク実行処理の手順を示すフローチャートである。
【図４】図４は、マルチプロセッシングシステムのアドレス変換処理を示す説明図である。
【図５】図５は、ＡＭＰシステムにおけるマスタＣＰＵとスレーブＣＰＵとの構成を比較する説明図である。
【図６】図６は、アドレス変換機構の構成を示すブロック図である。
【図７】図７は、論理アドレスと物理アドレスの対応の変化例を示すブロック図である。
【図８】図８は、ＯＳタスクセットの切り替えタイミングを示すタイミングチャートである。
【図９】図９は、ＳＭＰシステムの構成例を示すブロック図である。
【図１０】図１０は、ＡＭＰシステムの構成例１を示すブロック図である。
【図１１】図１１は、ＡＭＰシステムの構成例２を示すブロック図である。
【図１２】図１２は、ＡＭＰシステムの構成例３を示すブロック図である。
【符号の説明】
【００３３】
２００　マルチプロセッシングシステム
２０１　マスタＣＰＵ
２０２　ＤＭＡＣ
２０３　スレーブＣＰＵ
２０４　ＲＡＭ
２０５　割り込みコントローラ
２０６　リセットコントローラ
【発明を実施するための最良の形態】
【００３４】
以下に添付図面を参照して、この発明にかかるマルチプロセッシングシステムの好適な実施の形態を詳細に説明する。
【００３５】
（メインメモリの利用形態）
まず、本発明にかかるマルチプロセッシングシステムの特徴となるメインメモリの利用形態について説明する。図１は、本実施の形態にかかるマルチプロセッシングシステムにおけるメインメモリの利用形態を示す説明図である。
【００３６】
図１のように、本発明にかかるマルチプロセッシングシステムでは、メインメモリ１００にＯＳタスクセット１１０をロードする。ＯＳタスクセットとは、スレーブＣＰＵにロードするＯＳと、このＯＳによって実行させる複数のプログラムのタスクとの組み合わせである。ＯＳタスクセットは、通常、マルチプロセッシングシステムからアクセス可能な記録媒体のファイルシステム１２０に格納されている。また、ＯＳタスクセットを構成するＯＳおよびタスクは、静的にリンクされている。

【００３７】
メインメモリ１００に格納されるＯＳタスクセット１３０は、タスクの実行時に、マスタＣＰＵなど所定プロセッサからの指示に応じて、ファイルシステム１２０から読み出される。また、頻繁に実行されるタスクの場合、メインメモリ１００の中に該当するＯＳタスクセットが読み出される。
【００３８】
（マルチプロセッシングシステムのハードウェア構成）
つぎに、上述のようなメインメモリの利用形態を適用させたマルチプロセッシングシステムのハードウェア構成について説明する。図２は、本実施の形態にかかるマルチプロセッシングシステムのハードウェアの構成の一例を示すブロック図である。図２のように、マルチプロセッシングシステム２００は、マスタＣＰＵ２０１と、ＤＭＡＣ２０２と、スレーブＣＰＵ２０３と、ＲＡＭ２０４と、割り込みコントローラ２０５と、リセットコントローラ２０６とを含んで構成される。
【００３９】
マスタＣＰＵ２０１は、マルチプロセッシングシステム１００全体を制御する。また、処理対象となっているタスクをスレーブＣＰＵ２０３に分散して実行させる。ＤＡＭＣ２０２は、マスタＣＰＵ２０１やスレーブ２０２を介すことなく、直接ＲＡＭ２０４にアクセスしてデータ転送おこなう送受信用メモリ・コントローラである。スレーブＣＰＵ２０３は、マスタＣＰＵ２０１の指示に応じて指定されたタスクを実行する。なお、マスタＣＰＵ２０１およびスレーブＣＰＵ２０３の機能については詳しく後述する。
【００４０】
ＲＡＭ２０４は、高速アクセス可能な記録媒体である。また、ＲＡＭ２０４は、マルチプロセッシングシステム２００のメインメモリとして機能し、マスタＣＰＵ２０１およびスレーブＣＰＵ２０３のワークエリアとして利用される。
【００４１】
割り込みコントローラ２０５は、マスタＣＰＵ２０１からスレーブＣＰＵ２０３のタスク実行指示に対して、割り込み処理をおこなう。割り込みコントローラ２０５は、割り込み指示を受け付けると、割り込み指示の優先度やマスク状態から判断してマスタＣＰＵ２０１に割り込み要求信号を発生させる。リセットコントローラ２０６はリセット指示を受け付けると、対象となったスレーブＣＰＵ２０３の処理をリセットする。
【００４２】
つぎに、マスタＣＰＵ２０１の機能について詳細に説明する。マスタＣＰＵ２０１は、図１に示したようなＯＳタスクセットを利用してスレーブＣＰＵ２０３にタスクを実行させるため、（１）スレーブＣＰＵの実行開始と実行停止を制御する機能、（２）スレーブＣＰＵで動作するＯＳタスクセットを退避・復元する機能、（３）スレーブＣＰＵにタスクセットの変更が必要なことを通知する機能の３種類の機能を備えている。
【００４３】
（１）スレーブＣＰＵの実行開始と実行停止を制御する機能とは、具体的には、スレーブＣＰＵ２０３にリセットを印加および解除する処理と、スレーブＣＰＵ２０３の実行を一時停止させる処理と、スレーブＣＰＵ２０３の指定したアドレスに格納されたタスクを実行させる処理を制御する機能である。
【００４４】
また、（２）スレーブＣＰＵ２０３によって実行させるＯＳタスクセットを退避・復元する機能とは、具体的には、マスタＣＰＵ２０１からアクセスできるメモリ領域とスレーブＣＰＵ２０３がアクセスできるメモリ領域の間におけるデータ転送処理を制御する機能である。たとえば、マスタＣＰＵ２０１からのロード・ストア機能、ＤＭＡデータ転送機能、ＬＡＮなどを介したデータ転送機能が挙げられる。
【００４５】
また、（３）スレーブＣＰＵ２０１にタスクセットの変更を通知する機能とは、具体的には、マスタＣＰＵ２０１とスレーブＣＰＵ２０３間の割り込み要求処理、マスタＣＰＵ

２０１とスレーブＣＰＵ２０３間のメッセージ通信処理を制御する機能である。メッセージ通信処理としては、たとえば、マスタＣＰＵ２０１とスレーブＣＰＵ２０３からアクセス可能な共有メモリもしくはマスタＣＰＵ２０１とスレーブＣＰＵ２０３間のメッセージデータの通信や、マスタＣＰＵ２０１とスレーブＣＰＵ２０３上で動作する割り込み処理プログラムの通信が挙げられる。

【００４６】

つぎに、スレーブＣＰＵ２０３の機能について説明する。スレーブＣＰＵ２０３は、マスタＣＰＵ２０１の指示に応じてタスクを実行させるため、（１）実行状態および実行可能状態のプロセス・タスクを知る機能、（２）実行状態および実行可能状態のプロセス・タスクのリストを保持する機能、（３）実行状態および実行可能状態のプロセス・タスクの実行を一時停止・再開する機能、（４）ＯＳ起動時に、プロセス・タスクのリストにあるプロセス・タスクを再開もしくは初期起動させる機能の４種類の機能を備えている。

【００４７】

（１）実行状態および実行可能状態のプロセス・タスクを知る機能、具体的には、スレーブＣＰＵ２０３上で動作するプロセス・タスクの管理領域にアクセスし、タスクの状態をリードして把握するための処理をおこなう機能である。

【００４８】

また、（２）実行状態および実行可能状態のプロセス・タスクのリストを保持する機能は、具体的には、実行状態および実行可能状態のプロセス・タスクのリストを作成して、スレーブＣＰＵ２０３からアクセス可能なメモリ上にそのリストを保存処理する機能である。

【００４９】

また、（３）実行状態および実行可能状態のプロセス・タスクの実行を一時停止・再開する機能は、具体的には、プロセス・タスクのリストにある実行状態および実行可能状態のプロセス・タスクを強制待ち状態（ＳＵＳＰＥＮＤ）処理や、プロセス・タスクのリストにある強制待ち状態（ＳＵＳＰＥＮＤ）にされたプロセス・タスクを再開（Ｒｅｓｕｍｅ）処理する機能である。

【００５０】

また、（４）ＯＳ起動時に、プロセス・タスクのリストにあるプロセス・タスクを再開もしくは初期起動させる機能は、具体的には、プロセス・タスクのリストにあるプロセス・タスクを再開するか、初期起動して実行する処理である。

【００５１】

（マルチプロセッシングシステムのタスク実行処理）

つぎに、上述したような機能を備えたマルチプロセッシングシステム１００におけるタスク実行処理の手順について説明する。図３は、マルチプロセッシングシステムのタスク実行処理の手順を示すフローチャートである。

【００５２】

図３のように、マルチプロセッシングシステム１００の場合、マスタＣＰＵ２０１からの要求をトリガとして、スレーブＣＰＵ２０３がタスク実行処理をおこなう。図３のフローチャートでは、マスタＣＰＵ２０１からＯＳ起動／再開（Ｓ３０１）、タスク再開／ロード要求（Ｓ３０５）、ＯＳタスクセットロード／入れ替え（ステップＳ３０９）の３種類のトリガある。したがって、トリガごとのスレーブＣＰＵ２０３の処理手順について説明する。

【００５３】

マスタＣＰＵ２０１からスレーブＣＰＵ２０３へＯＳ起動／再開（Ｓ３０１）が出力された場合、まず、メモリマップ設定または復元をおこない（ステップＳ３０２）、対象となるＯＳが初期化済みか否かを判断する（ステップＳ３０３）。ここで、対象となるＯＳが初期化済みでない場合（ステップＳ３０３：Ｎｏ）、ＯＳ初期化処理をおこない（ステップＳ３０４）、ＯＳ実行に移行する（ステップＳ３０６）。

【００５４】

一方、ステップＳ３０３において、対象となるＯＳが初期化済みであった場合（ステップＳ３０３：Ｙｅｓ）、動的ロード・ドライバの処理に移行する（ステップＳ３０８）。動的ロード・ドライバでは、まず、レジスタの復元をおこない（ステップＳ３１６）、一時停止タスクを再開する（ステップＳ３１７）。

【００５５】

ここで、動的ロード・ドライバ処理について説明する。これから説明する動的ロード・ドライバのステップＳ３１１～Ｓ３１５の処理は、ＯＳ実行状態（ステップＳ３０６）が、トリガとなる。まず、ステップＳ３０６の処理内容から、メインメモリに格納されているＯＳタスクセットの実行状態および実行可能状態のプロセス・タスクのリストを作成し、保存する（ステップＳ３１１）。そして、保存したリストに記載されているタスクの実行を強制待ち状態（ＳＵＳＰＥＮＤ）して、ＯＳタスクセットの実行を一時停止する（ステップＳ３１２）。

【００５６】

そして、スレーブＣＰＵ２０３は、ステップＳ３１２によってＯＳタスクセットが停止したことを通知するマスタＣＰＵ２０１に通知する（ステップＳ３１３）。その後、タスクの一時停止をレジスタに保存し（ステップＳ３１４）、必要に応じてスレーブＣＰＵ２０３を停止させ（ステップＳ３１５）、処理を終了する。

【００５７】

また、マスタＣＰＵ２０１からスレーブＣＰＵ２０３へ、タスク再開／ロード要求（Ｓ３０５）が出力された場合、ステップＳ３０５のタスク再開／ロード要求に応じて、スレーブＣＰＵ２０３は、ＯＳを実行させる（ステップＳ３０６）。そして、ＯＳタスクセットに応じて実行されたＯＳと組み合わせられたタスクを実行する（ステップＳ３０７）。なお、スレーブＣＰＵ上で対応するプログラムを含むＯＳタスクセットが実行されていないときには、マスタＣＰＵ２０１からスレーブＣＰＵ２０３へ、ＯＳタスクセットの変更が必要なことを通知する。

【００５８】

また、マスタＣＰＵ２０１からスレーブＣＰＵ２０３へＯＳタスクセットロード／入れ替えが出力された場合（ステップＳ３０９）、ＯＳタスクセットを入れ替えるために、ＯＳタスクセットをファイルシステムからメインメモリにロード、もしくは、マスタＣＰＵ２０１からアクセスできる他のメモリからスレーブＣＰＵからアクセスできるメインメモリ上にロードする（ステップＳ３１０）。なお、ＯＳタスクセットの入れ替えがあった場合には、スレーブＣＰＵ２０３がＯＳタスクセットを読み出す際のメモリマップを変更しておかなければならない。このメモリマップの変更については詳しく後述する。

【００５９】

（マルチプロセッシングシステムのアドレス変換処理）

ここで、プロセッシングシステムのアドレス変換処理について説明する。図４は、マルチプロセッシングシステムのアドレス変換処理を示す説明図である。図４のように、スレーブＣＰＵ２０３には、論理メモリマップ４００と、物理メモリマップ４１０とが格納されている。

【００６０】

論理メモリマップ４００は、スレーブＣＰＵ２０３によって与えられたアドレスに一般データ４０１や、ＯＳタスクセット４０２の呼び出し時に用いられる論理アドレスが記録されている。一方、物理アドレスマップ４１０には、アドレスに一般データ４１１や、複数のＯＳタスクセット４２０（ＯＳタスクセット４２１～４２３）が格納されている物理アドレスが記録されている。

【００６１】

スレーブＣＰＵ２０３においてタスクが実行される際には、ＯＳタスクセットは、物理メモリマップ４１０に記録された物理アドレスで呼び出される。したがって、論理アドレスには、対応する物理アドレスが記録されている。しかしながら、物理アドレスは、格納場所の移動に伴い書き換えられる。したがって、物理アドレスの書き換えは、即座に論理

メモリマップ４００の各論理アドレスに反映される。このようなアドレス変換機構によって、スレーブＣＰＵ２０３高速なＯＳタスクセットの入れ替えを実現する。

【００６２】

上述したアドレス変換処理をおこなうには、マスタＣＰＵ２０１、スレーブＣＰＵ２０３にそれぞれ後述のような機能を備える必要がある。まず、マスタＣＰＵ２０１には、スレーブＣＰＵ２０３のメモリマップを制御する機能が必要となる。スレーブＣＰＵ２０３のメモリマップを制御する機能とは、具体的には、スレーブＣＰＵ２０３にメモリマップの変更が必要な旨の通知処理をおこなう機能である。

【００６３】

また、スレーブＣＰＵ２０３には、（１）複数のＯＳタスクセットを保持し、切り替えるためのアドレス変換機能と、（２）メモリマップを変更する機能とが必要となる。（１）複数のＯＳタスクセットを保持し切り替えるためのアドレス変換機能とは、具体的には、論理アドレスと物理アドレスとを変換処理する機能である。また、（２）メモリマップを変更する機能とは、論理アドレスと物理アドレスを変換するハードウェア機構の設定を操作してメモリマップを変更する機能である。

【００６４】

つぎに、上述したマルチプロセッシングシステムの各機能を用いて具体的な実施例について説明する。

【００６５】

（実施例１：ＯＳタスクセットを利用したロード）

実施例１では、マルチプロセッシングシステムに上述したＯＳタスクセットを適用させた場合の動作について説明する。図５は、ＡＭＰシステムにおけるマスタＣＰＵとスレーブＣＰＵとの構成を比較する説明図である。

【００６６】

図５において、ＡＭＰシステム５００は、従来例の構成を示している。一方、ＡＭＰシステム５１０は、本発明にかかるマスタＣＰＵとスレーブＣＰＵとの構成を反映している。ＡＭＰシステム５００の場合、マスタＣＰＵ５０１、スレーブＣＰＵ５０２それぞれには、静的ロードＯＳが搭載されている。したがって、スレーブＣＰＵ５０２がタスクを実行する場合には、静的ロードＯＳと、静的ロード（静的リンク）プログラムとを併せた複数のプログラム５０３が一緒にメインメモリにロードされる。

【００６７】

一方、ＡＭＰシステム５１０のマスタＣＰＵ５１１には、スレーブＣＰＵ動的ロード制御モジュールが追加されている。このスレーブＣＰＵ動的ロード制御モジュールによってスレーブＣＰＵの実行制御（実行開始、実行停止）、メモリアドレス変換の制御、スレーブＣＰＵへのアクセスをおこなうことができる。

【００６８】

スレーブＣＰＵ動的ロード制御モジュールは、具体的には、スレーブＣＰＵ２０３にタスクを実行させるため、（１）スレーブＣＰＵの実行開始と実行停止を制御する機能、（２）スレーブＣＰＵで動作するＯＳタスクセットを退避・復元する機能、（３）スレーブＣＰＵにタスクセットの変更が必要なことを通知する機能の３種類の機能を実現する。

【００６９】

また、ＡＭＰシステム５１０のスレーブＣＰＵ５１２には、スレーブＣＰＵ動的ロード制御モジュールに応答するための、動的ロードモジュールが追加されている。この動的ロードモジュールによって、スレーブＣＰＵ５１２における実行中のプロセス・タスクの把握、リストの保存、タスクの実行の一時停止や再開、スレーブＣＰＵのメモリマップの変更などをおこなうことができる。

【００７０】

（実施例２：アドレス変換機構）

つぎに、実施例２では、マルチプロセッシングシステムに上述したアドレス変換機構を適用させた場合の具体的な動作について説明する。図６は、アドレス変換機構の構成を示

すブロック図である。図６に示したアドレス変換機構は、論理アドレス６１０と、物理アドレスと、サイズ６３０と、有効／無効６４０との４つのレジスタによって構成されている。なお、図６の構成は一例であり、他の構成であってもよい。

【００７１】

実施例２では、スレーブＣＰＵで動作する複数のＯＳタスクセットを格納できるメモリ領域を確保して、そのメモリ領域にＯＳタスクセットを配置する。この際に、図６のような構成のアドレス変換機構を利用することによって、高速にＯＳタスクセットの入れ替えを実現できる。

【００７２】

ここで、アドレス変換機構の各レジスタは、下記のような項目を記録する。

論理アドレス６１０：ＬＡＤＲ０…Ｎ（論理アドレスの先頭）

物理アドレス６２０：ＰＡＤＲ０…Ｎ（物理アドレスの先頭）

サイズ６３０　　：ＳＩＺＥ０…Ｎ（メモリアドレス領域のサイズｏｒアドレスマスクビット）

有効／無効６４０　：Ｖａｌｉｄ０…Ｎ：（データの有効／無効）

【００７３】

ＯＳタスクセット６０１を管理するには、たとえば、図６のようにＯＳ＿ＳＥＴ　Ａ（６０１）の　ＬＤＡＲ　ＭにＯＳタスクセットがあるべき領域の先頭アドレスを設定する。また、ＰＡＤＲ　ＭにＯＳタスクセットが格納されている物理アドレスを設定し、ＳＩＺＥ　Ｍにサイズなどを設定し、Ｖａｌｉｄ　Ｍに１などを設定することでアドレス変換を有効にする。

【００７４】

また、図７は、論理アドレスと物理アドレスの対応の変化例を示すブロック図である。図７のように、Ｖａｌｉｄ　Ｍ　や、ＰＡＭＲ　Ｍ、ＳＩＺＥ　Ｍを変更することにより、スレーブＣＰＵが実行するＯＳタスクセットの論理メモリアドレスＬＡＤＲ　Ｍから始まるメモリ領域の内容を、物理メモリアドレス領域のＰＡＤＲ　Ｍ＿ｏｌｄの内容を高速にＰＡＤＲ　Ｍ＿ｎｅｗの内容に切り替えることができる。

【００７５】

（実施例３：ＯＳタスクセットの切り替え）

実施例３では、上述した実施例１、２のマルチプロセッシングシステムにおいて、タスクを効率的に実行して、処理時間を短縮させるためのＯＳタスクセットの切り替えの手法について説明する。

【００７６】

図８は、ＯＳタスクセットの切り替えタイミングを示すタイミングチャートである。タイミングチャート８１０は、通常のＯＳタスクセットの切り替えタイミングをあらわしている。スレーブＣＰＵ２０３によってＯＳタスクセットＡとＯＳタスクセットＢとを順次実行させる場合、通常、スレーブＣＰＵ２０３においてＯＳタスクセットＡが終了すると、マスタＣＰＵ２０１から、つぎのＯＳタスクセットＢのデータを転送が開始される。

【００７７】

そして、マスタＣＰＵ２０１によるＤＡＭＡＣ２０２を介したＯＳタスクセットＢのデータ転送が終了すると、この終了に応答してスレーブＣＰＵ２０３によってＯＳタスクセットＢの処理が開始される。すなわち、スレーブＣＰＵ２０３は、ＯＳタスクセットＡが終了してからつぎのＯＳタスクセットＢの処理を開始するまでＴＡは、なにも処理をおこなわない待機状態となっていた。

【００７８】

一方、実施例３を示すタイミングチャート８２０は、スレーブＣＰＵ２０３によってＯＳタスクセットＡが実行されると、マスタＣＰＵ２０１では、つぎに実行されるＯＳタスクセットＢのデータを転送している。すなわち、スレーブＣＰＵ２０３は、ＯＳタスクセットＡが終了し、マスタＣＰＵ２０１からのデータ転送が終了次第、ＯＳタスクセットＢの実行に移行できる。したがって、つぎのＯＳタスクセットＢの処理を開始するまでＴＢ

は、大きく短縮される。このように、実施例３では、タスクを効率的に実行して、処理時間を短縮させることができる。

【００７９】

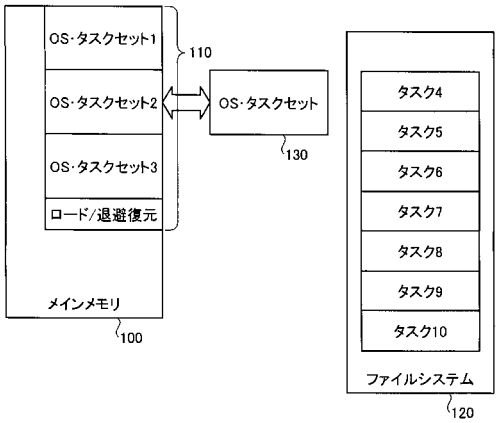
以上説明したように、本発明の実施の形態にかかるマルチプロセッシングシステムによれば、所定のＯＳを利用する場合に、対応するすべてのタスクをマインメモリにロードすることなく、必要なタスクを実行する際に、対応するＯＳをロードすればよい。したがって、メモリに負担をかけることなく、タスクに応じてＯＳを切り替えることができる。

【産業上の利用可能性】

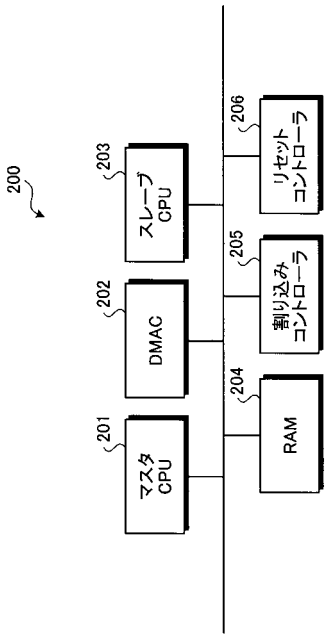
【００８０】

以上のように、本発明にかかるマルチプロセッシングシステムは、組み込み型の電機計算機、コンピュータおよびＣＰＵへの適用に有用であり、特に、非対称型マルチプロセッシングシステムに適している。

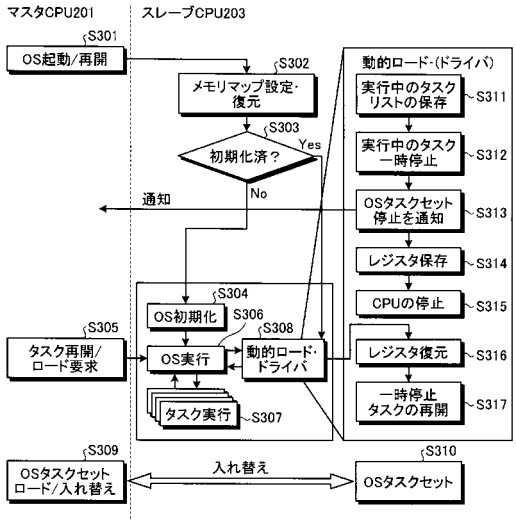
【図１】



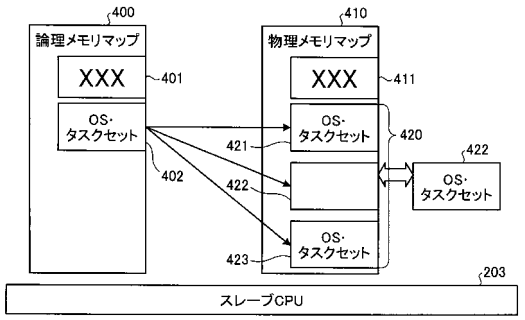
【図２】



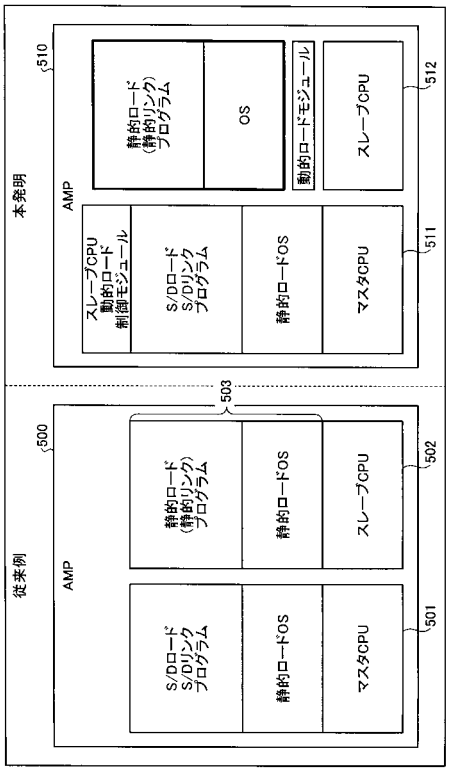
【図３】


マスタCPU201
スレーブCPU203
S301
OS起動/再開
S302
メモリマップ設定・復元
S303
初期化済？
Yes
No
通知
動的ロード・(ドライバ)
実行中のタスクリストの保存
S311
実行中のタスク一時停止
S312
OSタスクセット停止を通知
S313
レジスタ保存
S314
CPUの停止
S315
レジスタ復元
S316
一時停止タスクの再開
S317
S304
OS初期化
S306
S308
S305
タスク再開/ロード要求
OS実行
動的ロード・ドライバ
タスク実行
S307
S309
OSタスクセットロード/入れ替え
入れ替え
S310
OSタスクセット

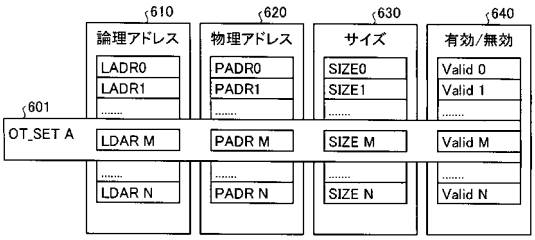
【図４】


400
論理メモリマップ
XXX
401
OS・タスクセット
402
410
物理メモリマップ
XXX
411
OS・タスクセット
420
421
422
423
422
OS・タスクセット
OS・タスクセット
203
スレーブCPU

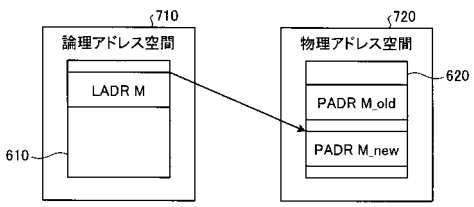
【図５】


本発明
510
静的ロード(静的リンク)プログラム
OS
動的ロードモジュール
スレーブCPU
512
AMP
スレーブCPU動的ロード制御モジュール
S/Dロード S/Dリンクプログラム
静的ロードOS
マスタCPU
511
503
従来例
500
静的ロード(静的リンク)プログラム
静的ロードOS
スレーブCPU
502
AMP
S/Dロード S/Dリンクプログラム
静的ロードOS
マスタCPU
501

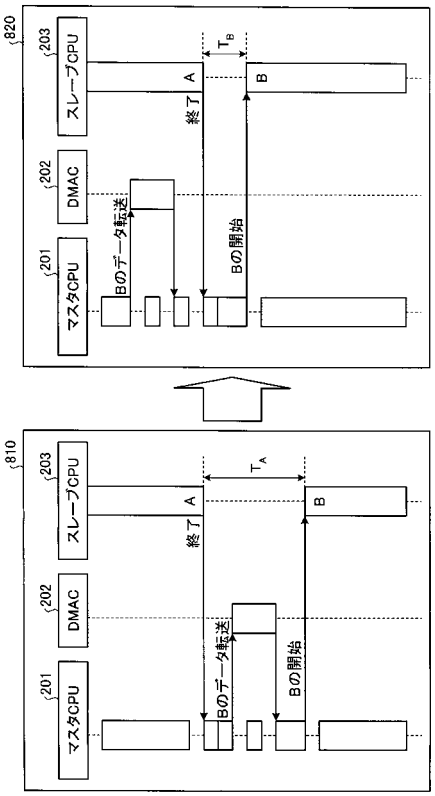
【図６】


610
論理アドレス
620
物理アドレス
630
サイズ
640
有効/無効
601
OT_SET A
LADR0
LADR1
LDAR M
LDAR N
PADR0
PADR1
PADR M
PADR N
SIZE0
SIZE1
SIZE M
SIZE N
Valid 0
Valid 1
Valid M
Valid N

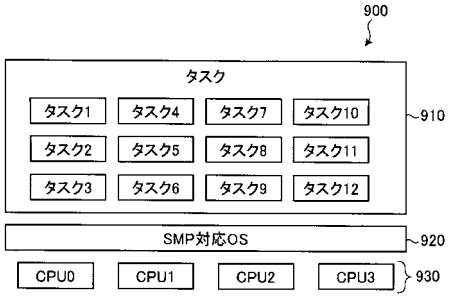
【図７】


710
論理アドレス空間
LADR M
610
720
物理アドレス空間
620
PADR M_old
PADR M_new

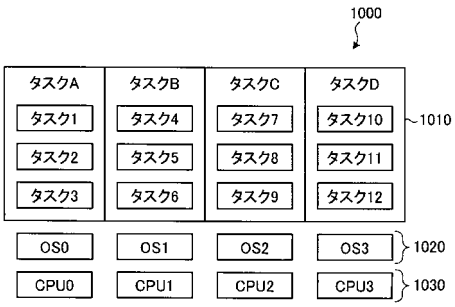
【図８】



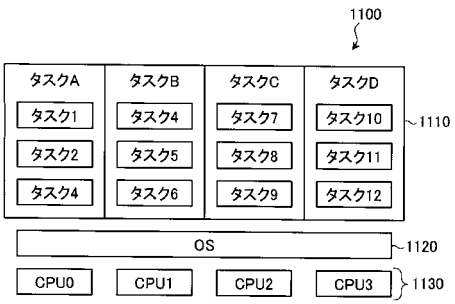
【図９】



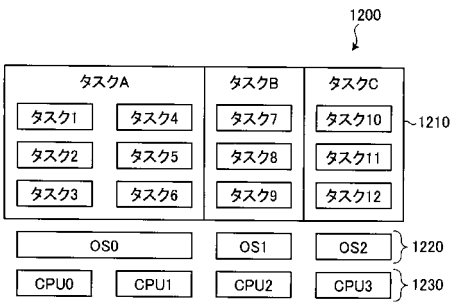
【図１０】



【図１１】



【図１２】



フロントページの続き

(56)参考文献 特開２００６－０９９３３３（ＪＰ，Ａ）
特開２００５－３４６３５８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/46
G06F 9/48