(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4986247号
(P4986247)
(45)発行日　平成24年7月25日(2012.7.25)　(24)登録日　平成24年5月11日(2012.5.11)

(51)Int.Cl.　　F I
G06F 9/445 (2006.01)　　G06F 9/06 610J
G06F 9/54 (2006.01)　　G06F 9/06 640B

請求項の数 15　(全 25 頁)

(21)出願番号　特願2009-132708 (P2009-132708)
(22)出願日　平成21年6月2日 (2009.6.2)
(65)公開番号　特開2010-282252 (P2010-282252A)
(43)公開日　平成22年12月16日 (2010.12.16)
審査請求日　平成23年11月30日 (2011.11.30)

早期審査対象出願

(73)特許権者　505381024
株式会社ユビキタス
東京都新宿区西新宿６－１０－１　日土地西新宿ビル　２０Ｆ
(74)代理人　100102406
弁理士　黒田　健二
(74)代理人　100100240
弁理士　松本　孝
(72)発明者　橋本　健一
東京都新宿区西新宿１－２５－１　新宿センタービル　株式会社ユビキタス内

審査官　稲垣　良一

最終頁に続く

(54)【発明の名称】プログラム、制御方法、並びに制御装置

(57)【特許請求の範囲】
【請求項１】
ソフトウェアの起動を制御する方法であって、メモリを管理する機能を有するプロセッサに、
所定の動作状態にあるソフトウェアの前記所定の動作に必要な物理ページに対して、全ての物理ページでページフォルトが発生するように、メモリ管理用のページテーブルエントリを書き換えるステップ、
前記書き換えるステップの後に、前記所定の動作状態にあるソフトウェアのメモリイメージを保存するステップ、および
前記ソフトウェアが起動する時、前記メモリ管理用のページテーブルエントリを使用して、各ページテーブルエントリでページフォルトを発生させ、そのページフォルトが発生したページを順次読み出すステップ
を実行させる、ソフトウェアの起動制御方法。
【請求項２】
請求項１に記載のソフトウェアの起動制御方法において、
前記メモリは、ＲＡＭおよび不揮発性メモリを含み、
前記書き換えるステップは、前記ＲＡＭが記憶しているページテーブルエントリを書き換えるステップを含み、
前記保存するステップは、前記所定の動作状態にあるソフトウェアのデータ、プログラムコード、ページテーブルエントリを書き換えられたメモリ管理用テーブル、ページフォ

ルトハンドラ、割込ベクタ、およびレジスタを、前記不揮発性メモリに記憶させるステップを含む
方法。
【請求項３】
請求項２に記載のソフトウェアの起動制御方法において、
前記書き換えるステップは、各ページテーブルエントリが示す物理ページが存在するか否か、および前記物理ページが前記ＲＡＭ内にあるかを判断するステップ、前記ＲＡＭが記憶している前記ページテーブルエントリのうち前記物理ページが存在するページテーブルエントリを、前記物理ページへのアクセスを禁止する情報に書き換えるステップ、および前記書き換えるステップにより前記物理ページへのアクセスを禁止する情報に書き換えられたことを識別するマーキングを実行するステップを含み、
前記読み出すステップは、前記ページフォルトが発生した物理ページに前記マーキングがされているかを判断し、前記マーキングがされたと判断された物理ページを読み出す
方法。
【請求項４】
請求項２または３に記載のソフトウェアの起動制御方法において、
前記読み出すステップは、
ページフォルトが発生したとき、前記割込ベクタが前記ページフォルトハンドラを呼び出し、前記ページフォルトハンドラが、ページフォルトが発生したアドレスからページを計算し、そのページを前記不揮発性メモリから読み出すステップを含む
方法。
【請求項５】
請求項１または２に記載のソフトウェアの起動制御方法において、
前記プロセッサが、組み込み型のコンピュータのプロセッサである
方法。
【請求項６】
情報処理装置であって、
メモリ、前記メモリを管理する機能を有するプロセッサ、およびプログラムコードを有し、
前記プログラムコードは、ソフトウェアの起動を制御するために前記プロセッサに、
所定の動作状態にあるソフトウェアの前記所定の動作に必要な物理ページに対して、全ての物理ページでページフォルトが発生するように、メモリ管理用のページテーブルエントリを書き換えるステップ、
前記書き換えるステップの後に、前記所定の動作状態にあるソフトウェアのメモリイメージを保存するステップ、および
前記ソフトウェアが起動する時、前記メモリ管理用のページテーブルエントリを使用して、各ページテーブルエントリでページフォルトを発生させ、そのページフォルトが発生したページを順次読み出すステップ
を実行させる、情報処理装置。
【請求項７】
請求項６に記載の情報処理装置において、
前記メモリは、ＲＡＭおよび不揮発性メモリを含み、
前記書き換えるステップは、前記ＲＡＭが記憶しているページテーブルエントリを書き換えるステップを含み、
前記保存するステップは、前記所定の動作状態にあるソフトウェアのデータ、プログラムコード、ページテーブルエントリを書き換えられたメモリ管理用テーブル、ページフォルトハンドラ、割込ベクタ、およびレジスタを、前記不揮発性メモリに記憶させるステップを含む
装置。
【請求項８】

請求項７に記載の情報処理装置において、
前記書き換えるステップは、各ページテーブルエントリが示す物理ページが存在するか否か、および前記物理ページが前記ＲＡＭ内にあるかを判断するステップ、前記ＲＡＭが記憶している前記ページテーブルエントリのうち前記物理ページが存在するページテーブルエントリを、前記物理ページへのアクセスを禁止する情報に書き換えるステップ、および前記書き換えるステップにより前記物理ページへのアクセスを禁止する情報に書き換えられたことを識別するマーキングを実行するステップを含み、
前記読み出すステップは、前記ページフォルトが発生した物理ページに前記マーキングがされているかを判断し、前記マーキングがされたと判断された物理ページを読み出す
装置。
【請求項９】
請求項７または８に記載の情報処理装置において、
前記読み出すステップは、
ページフォルトが発生したとき、前記割込ベクタが前記ページフォルトハンドラを呼び出し、前記ページフォルトハンドラが、ページフォルトが発生したアドレスからページを計算し、そのページを前記不揮発性メモリから読み出すステップを含む
装置。
【請求項１０】
請求項６または７に記載の情報処理装置において、
前記プロセッサが、組み込み型のコンピュータのプロセッサである
装置。
【請求項１１】
記憶媒体にコンピュータ読み取り可能に記憶されるプログラムであって、
メモリを管理する機能を有するプロセッサに、
所定の動作状態にあるソフトウェアの前記所定の動作に必要な物理ページに対して、全ての物理ページでページフォルトが発生するように、メモリ管理用のページテーブルエントリを書き換えるステップ、
前記書き換えるステップの後に、前記所定の動作状態にあるソフトウェアのメモリイメージを保存するステップ、および
前記ソフトウェアが起動する時、前記メモリ管理用のページテーブルエントリを使用して、各ページテーブルエントリでページフォルトを発生させ、そのページフォルトが発生したページを順次読み出すステップ
を含むソフトウェアの起動を制御する方法を実行させる、プログラム。
【請求項１２】
請求項１１に記載のプログラムにおいて、
前記メモリは、ＲＡＭおよび不揮発性メモリを含み、
前記書き換えるステップは、前記ＲＡＭが記憶しているページテーブルエントリを書き換えるステップを含み、
前記保存するステップは、前記所定の動作状態にあるソフトウェアのデータ、プログラムコード、ページテーブルエントリを書き換えられたメモリ管理用テーブル、ページフォルトハンドラ、割込ベクタ、およびレジスタを、前記不揮発性メモリに記憶させるステップを含む
プログラム。
【請求項１３】
請求項１２に記載のプログラムにおいて、
前記書き換えるステップは、各ページテーブルエントリが示す物理ページが存在するか否か、および前記物理ページが前記ＲＡＭ内にあるかを判断するステップ、前記ＲＡＭが記憶している前記ページテーブルエントリのうち前記物理ページが存在するページテーブルエントリを、前記物理ページへのアクセスを禁止する情報に書き換えるステップ、および前記書き換えるステップにより前記物理ページへのアクセスを禁止する情報に書き換え

られたことを識別するマーキングを実行するステップを含み、
前記読み出すステップは、前記ページフォルトが発生した物理ページに前記マーキングがされているかを判断し、前記マーキングがされたと判断された物理ページを読み出す
プログラム。
【請求項１４】
請求項１２または１３に記載のプログラムにおいて、
前記読み出すステップは、
ページフォルトが発生したとき、前記割込ベクタが前記ページフォルトハンドラを呼び出し、前記ページフォルトハンドラが、ページフォルトが発生したアドレスからページを計算し、そのページを前記不揮発性メモリから読み出すステップを含む
プログラム。
【請求項１５】
請求項１１または１２に記載のプログラムにおいて、
前記プロセッサが、組み込み型のコンピュータのプロセッサである
プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明はプログラム、制御方法、並びに制御装置に関し、特に、ソフトウェアの起動を制御するときに好適なプログラム、制御方法、並びに制御装置に関する。
【背景技術】
【０００２】
パーソナルコンピュータにおいて、ＯＳ（Operating System）を起動させ、所望のソフトウェアを動作させるまでには数分単位の起動時間が必要であった。これを高速に起動させる手法として、ハイバネーション（hibernation）と呼ばれる手法が存在する（例えば、特許文献１参照）。
【０００３】
特許文献１には、起動後のＣＰＵ（central processing unit）やＩ／Ｏ（input／output）レジスタ、ＲＡＭ（Random Access Memory）イメージが、ハードディスクドライブ（ＨＤＤ）やフラッシュメモリに格納されることが記載されている。そして、次に起動されるときに、格納されているＲＡＭイメージが復帰され、その後、ＣＰＵやＩ／Ｏレジスタが再設定されることが記載されている。このように起動することで、ＯＳの起動を高速化させることが、特許文献１では提案されている。このような提案に基づくハイバネーションと称される手法は、既にパーソナルコンピュータで適用されている。
【０００４】
また、組み込み型のコンピュータ、例えば、テレビジョン受像器、ハードディスクレコーダといった電子機器に組み込まれているコンピュータにおいても、ハイバネーションの手法が応用されている。
【先行技術文献】
【特許文献】
【０００５】
【特許文献１】特開２００５－１４９２２５号公報
【特許文献２】特開２００７－３３４３８３号公報
【発明の概要】
【発明が解決しようとする課題】
【０００６】
ハイバネーションを適用し、ＯＳを起動させる場合とＯＳを通常通りに起動させる場合とを比較した場合、ハイバネーションを適用してＯＳを起動させる方が、より高速に起動させることができる。しかしながら、ＲＡＭの容量増加に伴って保存すべきＲＡＭイメージのサイズも増加してしまい、起動時に、そのＲＡＭイメージの展開時間も増加してしま

う。結果として、ＲＡＭの大容量化に伴い高速起動が困難になってしまう。

【０００７】

また、パーソナルコンピュータは、ＣＰＵの性能も比較的高いため、仮にＲＡＭイメージのサイズが増加してしまっても、そのＲＡＭイメージを処理する性能が確保される。しかしながら、組み込み型のコンピュータの場合、ＣＰＵの性能が比較的低いものが使われることが多い。そのため、組み込み型のコンピュータの場合、ＲＡＭイメージが増加すれば、ハイバネーションの手法を適用しても、起動時の速度は低下してしまう。すなわち、組み込み型のコンピュータの場合、ＲＡＭイメージの増加による速度の低下は、より顕著に表れてしまう。

【０００８】

また、ＲＡＭイメージを圧縮することで、ＲＡＭイメージのサイズを小さくすることも提案されているが、起動時に伸張する処理が必要となる。この伸張処理にかかるＣＰＵへの負荷や、伸張処理にかかる時間を考慮すると、起動の高速化という点では効果的な方法ではない。

【０００９】

このようなことを考慮し、特許文献２では、ハイバネーションのイメージの全ての転送を完了する前に、ＯＳの実行を開始する手法が提案されている。しかしながら、この手法では、特別なハードウエアを搭載し、先行転送するページを予め特定しておく必要があるため、その特別なハードウエアの分だけコストが高くなるなどの問題点があった。

【００１０】

本発明は、このような状況に鑑みてなされたものであり、起動時間を短縮することができるようにするものである。

【課題を解決するための手段】

【００１１】

本発明の一側面のプログラムは、メモリを管理する機能を有する制御装置に、所定のソフトウェアの動作に必要なページに対して、全てのページでページフォルトが発生するようにページテーブルエントリを書き換え、前記ソフトウェアが起動時に、前記ページテーブルエントリで、ページフォルトが発生し、そのページフォルトが発生したページを順次読み出すステップを含む。

【００１２】

前記所定のソフトウェアが起動された後、前記ページテーブルエントリを書き換え、その起動時のデータ、プログラムコード、テーブル、ページフォルトハンドラ、割り込みベクタ、およびレジスタを、前記メモリに記憶させるようにすることができる。

【００１３】

前記メモリのうち、書き換えの対象となる前記ページテーブルエントリを記憶しているのはＲＡＭであり、順次読み出される前記ページを記憶しているのは不揮発メモリであるようにすることができる。

【００１４】

組み込み型のコンピュータが読み込むようにすることができる。

【００１５】

本発明の一側面の制御方法は、メモリを管理する機能を有する制御装置の制御方法において、所定のソフトウェアの動作に必要なページに対して、全てのページでページフォルトが発生するようにページテーブルエントリを書き換え、前記ソフトウェアが起動時に、前記ページテーブルエントリで、ページフォルトが発生し、そのページフォルトが発生したページを順次読み出すステップを含む。

【００１６】

本発明の一側面の制御装置は、メモリを管理する機能を有する制御装置において、所定のソフトウェアの動作に必要なページに対して、全てのページでページフォルトが発生するようにページテーブルエントリを書き換える書き換え手段と、前記ソフトウェアが起動時に、前記ページテーブルエントリで、ページフォルトが発生し、そのページフォルトが

発生したページを順次読み出す読み出し手段とを備える。

【００１７】

本発明の一側面のプログラム、制御方法、並びに制御装置は、所定のソフトウェアの動作に必要なページに対して、全てのページでページフォルトが発生するようにページテーブルエントリが書き換えられ、ソフトウェアが起動時に、ページテーブルエントリで、ページフォルトが発生し、そのページフォルトが発生したページが順次読みだされる。

【発明の効果】

【００１８】

本発明の一側面によれば、ＯＳの起動時間を短縮することが可能となる。

【図面の簡単な説明】

【００１９】

【図１】本発明を適用した情報処理装置の一実施の形態の構成を示す図である。

【図２】ＭＭＵのモデルを示す図である。

【図３】ディスクリプタについて説明するための図である。

【図４】物理ページの読み込みについて説明するための図である。

【図５】物理ページの読み込みについて説明するための図である。

【図６】物理メモリマップについて説明するための図である。

【図７】起動の処理について説明するためのフローチャートである。

【図８】起動の処理について説明するためのフローチャートである。

【図９】起動の処理について説明するためのフローチャートである。

【図１０】起動の処理について説明するためのフローチャートである。

【図１１】起動の処理について説明するためのフローチャートである。

【発明を実施するための形態】

【００２０】

以下に、本発明の実施の形態について図面を参照して説明する。

【００２１】

まず、本発明の概略を説明する。本発明は、Memory Management Unit(以下、ＭＭＵと略記する)を搭載したＣＰＵ（central processing unit）上で動作するＯＳ（Operating System）やアプリケーション等のソフトウェアを高速に起動させるための手法である。

【００２２】

高速起動の対象となるソフトウェアが、一度、通常通りの方法で起動され、その状態のＲＡＭ（Random Access Memory）イメージが不揮発メモリ等に保存される。ＲＡＭイメージが、不揮発メモリ等に保存されるとき、ＭＭＵのテーブルが書き換えられ、全てのページでページフォルトが発生するように変更される。対象とされるソフトウェアには、ページフォルトハンドラが用意されており、ページフォルトが発生した場合、そのページフォルトが発生したページのみが、不揮発メモリからロードされる。

【００２３】

任意のソフトウェアが起動され、プログラムコードが実行されたり、そのソフトウェアが動作に必要なデータを読み出すために、ＲＡＭにアクセスしたりした場合、毎回ページフォルトが発生し、必要なページが不揮発メモリからＲＡＭに逐次ロードされることになる。これにより、従来のハイバネーション起動とは異なり、全てのＲＡＭイメージを不揮発メモリからメインＲＡＭに予めロードする必要が無くなり、動作に必要な必要最小限のＲＡＭイメージのみをロードすることが可能になる。よって、高速に所望のソフトウェアを起動・動作させることが可能となる。

【００２４】

本発明を使用することで、従来は数十秒～数分を要していたＯＳやソフトウエア（以下、単にソフトウェアとする）の起動時間を数秒程度に短縮することが可能となることが、本出願人により確認されている。以下に、具体的に説明する。

【００２５】

［情報処理装置の構成について］

図１は、本発明を適用した情報処理装置の一実施の形態の構成を示す図である。本発明を適用した情報処理装置は、パーソナルコンピュータ（ＰＣ）に適用できることは勿論のこと、組み込み型のコンピュータを有する装置にも適用できる。組み込み型のコンピュータを含む装置としては、テレビジョン受像器、ハードディスクレコーダといった電子機器がある。ここでは、ハードディスクレコーダに対して本発明を適用したときを例にあげて説明する。

【００２６】

図１は、本発明を適用した情報処理装置としてのハードディスクレコーダの構成を示す図である。図に示したハードディスクレコーダ１００は、ＣＰＵ１０１、ＲＡＭ１０２、ＲＯＭ（Read Only Memory）１０３、不揮発メモリ１０４、ＭＰＥＧ（Moving Picture Experts Group）エンコード・デコード部１０５、チューナ１０６、ＨＤＤインターフェース１０７、ＨＤＤ１０８、Ｉ／Ｏ部１０９、起動モード切替部１１０を備える。

【００２７】

ＣＰＵ１０１は、ハードディスクレコーダ１００の各部を制御する。このＣＰＵ１０１は、Memory Management Unit(以下、ＭＭＵと記述する)を搭載し、ＲＡＭ１０２を小分割して小単位（ページ）で管理できる仕組みを有している。なお、ここでは、ＭＭＵ１３１がＣＰＵ１０１に含まれるとして説明を続けるが、ＭＭＵ１３１が、ＣＰＵ１０１に含まれず、外部に備えられている構成とすることも可能である。また、ＭＭＵ１３１の形式には特に制限はないが、ページ単位にアクセスの許可・禁止の属性が設定可能で、アクセス禁止のページにアクセスした場合は、ページフォルトの例外が発生できる構成とされている。また、ＣＰＵ１０１のＭＭＵ１３１は、４キロバイト（以下、４KBと記述する）を１ページとして管理するとして説明を続ける。

【００２８】

ＲＡＭ１０２は、ＳＲＡＭ（Static Random Access Memory）やＤＲＡＭ（Dynamic Random Access Memory）などで構成することが可能である。ＲＡＭ１０２は、ＣＰＵ１０１が使用する主記憶装置として機能し、そのような機能を有していれば、ＲＡＭ１０２として用いることが可能である。

【００２９】

ＲＯＭ１０３は、FLASH ROMやMask ROMなどの読み込み専用のメモリである。ＲＯＭ１０３は、ＯＳやアプリケーションソフトウェアが格納され、そのような格納が行えれば、どのような種類のＲＯＭが用いられても本発明においては良い。

【００３０】

不揮発メモリ１０４は、ハードディスクレコーダ１００の電源が切られても、記憶している内容が保持されるメモリである。例えば、FLASH ROMや、バックアップ機能付きのＳＲＡＭ、ＤＲＡＭなどで構成することが可能である。

【００３１】

不揮発メモリ１０４には、後述するソフトウェアが起動した後に、ＲＡＭ１０２に格納されたソフトウェアのメモリイメージが格納される。そのため、不揮発メモリ１０４の容量は、ＲＡＭ１０２の容量以上の容量とされることが好ましい。ただし、データ圧縮等によりデータを縮小するようにした場合、不揮発メモリ１０４の容量は、ＲＡＭ１０２の容量以下の容量でも良い。また、不揮発メモリ１０４は、ＨＤＤ１０８と兼用することも可能である（ＨＤＤ１０８を不揮発メモリ１０８として利用することも可能である）。

【００３２】

ＭＰＥＧエンコード・デコード部１０５は、動画の圧縮・伸張を行う。動画は、チューナ１０６を介して供給される。チューナ１０６は、ユーザの指示に基づき、複数の番組（動画）から、１つの動画を選択し、ＭＰＥＧエンコード・デコード部１０５に供給する。ＭＰＥＧエンコード・デコード部１０５は、必要に応じ、ＨＤＤインターフェース１０７を介して、ＨＤＤ１０８にチューナ１０６からのデータを供給したり、ＨＤＤ１０８からのデータをＨＤＤインターフェース１０７を介して、受信したりする。また、その際、必要に応じ、エンコードまたはデコードの処理を実行する。

【００３３】
Ｉ／Ｏ部１０９は、起動モード切替部１１０の状態をＣＰＵ１０１が読みとるために設けられている。以下に説明するソフトウェアは、高速起動イメージを取得するための通常起動モードと、高速起動イメージが取得された後の起動モードとしての高速起動モードを有する。Ｉ／Ｏ１０９と起動モード切替部１１０は、これらの起動モードの切り替えに用いられる。通常起動モードと高速起動モードの切り換えのために、起動モード切替部１１０をスイッチで構成することが可能である。また、通常起動モードと高速起動モードの切り換えのために、起動モード切替部１１０を、ブートローダ等からコマンドで切り替える構成とすることも可能である。

【００３４】
また、一度高速起動のイメージが作成されれば、通常起動モードを不要とすることができるため、高速起動モードのみが実装され、通常起動モードと高速起動モードの切り換えを不要とする構成としても良い。そのような構成とした場合、Ｉ／Ｏ部１０９と起動モード切替部１１０を省略した構成とすることも可能である。

【００３５】
［ＭＭＵについて］
図２は、ＣＰＵ１０１が装備するＭＭＵ１３１のモデルを示す図である。図２に示したＭＭＵ１３１は、３２ｂｉｔクラス以上のＣＰＵ１０１が装備するモデルである。ＭＭＵ１３１の構成や、物理アドレス、仮想アドレス、各テーブルのインデックスに使用するbit数、テーブルの段数等は、ＣＰＵ１０１のメーカに依存するものであるが、特にメーカのアーキテクチャに依存はしない。ここでは説明の都合上、３２bit等の具体的な数字を例にあげて説明するが、その数値は、本発明の適用範囲の限定を示すものではない。

【００３６】
ＭＭＵ１３１は、最終的な物理ページを指し示すテーブル内エントリの属性にアクセス許可もしくはアクセス許可相当の機能を実装し、不許可の場合にアクセスしたときには、ページフォルトもしくはページフォルト相当の例外処理が発生できる機能を少なくとも有する。

【００３７】
ＭＭＵレジスタ２００は、ＭＭＵ１３１に装備されるレジスタである。このレジスタにレベル１ディスクリプタテーブル２０１の先頭アドレスが代入される。レベル1ディスクリプタテーブル２０１は、メモリ上に置かれたレベル1メモリテーブルである。物理アドレス空間が３２bitの場合、1つのアドレスを指し示すのに３２bit、即ち４バイトが使用されるため、レベル１ディスクリプタテーブル２０１のサイズは、１２bit空間＝４KB空間×４バイトで１６KBのサイズとなる。

【００３８】
仮想アドレス２０２は、レベル1メモリテーブルのインデックスとして使われる仮想アドレスの３１bitから２０bitを示す。ＶＡはVirtual Address（仮想アドレス）を意味する。所定の仮想アドレスにアクセスがあった場合、レベル1ディスクリプタテーブル２０１の先頭アドレスから仮想アドレスの３１bitから２０bitがインデックスとされ、レベル1ディスクリプタ２０３にアクセスがされる。

【００３９】
物理アドレス空間が３２bitのＣＰＵ１０１である場合、１つのアドレスを示すのに３２bit、即ち４バイトが使用されるため、レベル1ディスクディスクリプタのアドレスは以下のような式になる。

レベル1ディスクリプタ２０３のアドレス
＝ レベル1ディスクリプタテーブル２０１の先頭アドレス
＋ 仮想アドレス２０２(VA[31:20]) ×４

【００４０】
レベル1ディスクリプタ２０３は、レベル２ディスクリプタテーブル２０４の先頭アドレスを指し示すポインタや属性から構成されるディスクリプタである。レベル２ディスク

リプタテーブル２０４の先頭アドレスから仮想アドレスの１９bitから１２bitがインデックスとされ、レベル２ディスクリプタ２０６にアクセスされる。

【００４１】

物理アドレス空間が３２bitのＣＰＵ１０１である場合、１つのアドレスを示すのに３２bit、即ち４バイトが使用されるため、レベル２ディスクディスクリプタのアドレスは以下のような式になる。

レベル２ディスクリプタ２０６のアドレス
＝ レベル２ディスクリプタテーブル２０４の先頭アドレス
＋ 仮想アドレス２０５(VA[19:12]) ×４

【００４２】

レベル２ディスクリプタ２０６は、４KBの物理ページ２０７を示すポインタや属性から構成されるディスクリプタである。物理ページ２０７は、最終的に仮想アドレスから物理アドレスに変換された１ページの物理メモリである。物理ページ２０７の４KB内のアドレスは、仮想アドレス２０８(VA[11:0])で指定される。

【００４３】

図３は、レベル1ディスクリプタ２０３やレベル２ディスクリプタ２０６の一例である。図３におけるディスクリプタはあくまでも一例であり、本発明は、このような一例で示した特定のＣＰＵ１０１やアーキテクチャに依存することを示すものではない。

【００４４】

ベースアドレス３０１は、次のテーブルや物理ページの先頭アドレスを指すポインタである。属性３０２乃至３０４は、それぞれ、実行可・不可、特権モード・ユーザモードといった属性を示す属性bitである。アクセス許可bit３０５は、このディスクリプタが示す物理ページに対してのアクセスが許可されているか否かを示すbitである。このアクセス許可bit３０５で、アクセスを禁止するように設定されている物理ページにアクセスがあった場合、一般的には、ページフォルトと呼ばれる例外処理、即ち割込み処理が実行され、そのような機構が必要である。よって、ＣＰＵ１０１のアーキテクチャに依存はしないが、ページフォルト、もしくはページフォルト相当の機能は、ＣＰＵ１０１に装備されている必要がある。

【００４５】

［物理ページについて］

図４は、物理ページ２０７が、ＲＡＭ１０２（図１）上に並んでいる状態を擬似的に示している。ＲＡＭ１０２には、１ページ分の物理ページ２０７が、物理ページ２０７－０から順に、物理ページ２０７－ｎまで配列されている。ＭＭＵ１３１搭載のＣＰＵ１０１上でソフトウェアが動作される場合、１ページの物理ページ２０７は、このように４KB乃至６４KB単位で１ページとして管理されるように構成されていることが多い。

【００４６】

図５は、ソフトウェアが所定の動作状態であるときの物理ページの使用状況について説明するための図である。図５中、数字が記載されているページが使用されているページであり、数字が記載されていないページは使用されていないページを示す。ソフトウェアが使うプログラムコードやデータが、図４に示したように、全ての領域を使っていたとしても、ソフトウェアの状態をある単位時間でみた場合、図５で示すように使われていることが多い。すなわち、使用されているページと使用されていないページとがあり、全てのページが使用されているわけではない。

【００４７】

このソフトウェアが、所定の動作状態のときには、物理ページ２０７－０、物理ページ２０７－２、物理ページ２０７－４、物理ページ２０７－５、物理ページ２０７－９、物理ページ２０７－１６、および物理ページ２０７－１８が使用される。すなわち、図４に示したように、ＲＡＭ１０２には、１ページ分の物理ページ２０７が、物理ページ２０７－０から順に、物理ページ２０７－ｎまで配列されているが、そのうち所定のソフトウェアが、所定の動作状態のときには、全てのページが使われるわけではなく、図５に示した

ように、複数のページのみが使用される。

【００４８】

従来のハイバネーション起動による起動の場合、ソフトウェアが動作する前に、図４に示したように、物理ページ２０７－０から物理ページ２０７－ｎまで、順次読み出しが行われてから復帰動作が始まり、所定の動作状態にされる。しかしながら実際に、そのソフトウェアが所定の動作状態になるためには、図５に示したように、所定の複数の物理ページ２０７が読み出されるだけでよい。そこで、本発明においては、後述するように読み出しを制御することで、図５に示したように、必要とされる物理ページ２０７のみが読み出されるようにする。

【００４９】

従来のハイバネーション起動による起動の場合、図４に示したように物理ページ２０７－０から順次読み出しが行われるため、換言すれば、必要のない物理ページ２０７も読み出されるため、読み出しに時間がかかり、結果として、所定のソフトウェア（ＯＳなども含む）の起動が遅くなるということがあった。しかしながら、本発明によれば、図５に示したように、必要とされる物理ページ２０７のみが読み出されるため、読み出しにかかる時間を短縮することができ、所定のソフトウェア（ＯＳなども含む）の起動を早くすることが可能となる。

【００５０】

図６は、ソフトウェアの物理メモリマップを示す図である。なお、以下に説明する処理は、ＣＰＵ１０１やＯＳに依存するもので、これらの機能が実装されていれば構成やメモリ配置に制限はなく、以下の説明だけに、本発明が適用されることを示すものではない。

【００５１】

不揮発メモリ１０４は、電源を切っても記憶内容が保持されるメモリである。図６に示した例では、不揮発メモリ１０４としてFLASH ROMをイメージしている。不揮発メモリ１０４は、メインメモリ上にマッピングされているが、電源を切っても内容が保持され、且つ、ＲＡＭ１０２以上の容量を装備している。しかしながら、本発明を適用できる不揮発メモリ１０４は、Ｉ／Ｏ経由でのアクセス等、必ずしもメモリマップ上にマッピングされている必要はなく、そのアーキテクチャに制限はない。

【００５２】

データ４０１は、プログラムコード４０２が使用する読み書きができるデータ領域である。データ４０１は物理ページ２０７として、ある特定の大きさに分割されて格納される。データ４０１は、読み書きできる必要があるのでＲＡＭ１０２上に置かれることが好ましい。

【００５３】

プログラムコード４０２は、起動・実行させる所望のプログラムを示す。一般的なパーソナルコンピュータでWindows（登録商標）やLinuxと言ったＯＳが搭載され、その上で動作するソフトウェアである場合、プログラムとは、そのＯＳとソフトウェアを含む。プログラムコード４０２は、物理ページ２０７として、ある特定の大きさに分割されて格納される。このプログラムコード４０２は、ＲＡＭ１０２またはＲＯＭ１０３上に置かれる。

【００５４】

ＭＭＵテーブル４０３は、図２に示したレベル1ディスクリプタテーブル２０１およびレベル２ディスクリプタテーブル２０４を示している。ページフォルトハンドラ４０４は、ＭＭＵ１３１のレベル２ディスクリプタ２０６がアクセス禁止の属性であり、ページフォルトが発生したとき、割込ベクタ経由で例外処理を行うためのプログラムである。ここではプログラムコード４０２と分けて、ページフォルトハンドラ４０４を記述しているが、プログラムコード４０２に含まれる場合もある。

【００５５】

割込ベクタ４０５は、一般的なＣＰＵが持つ割込ベクタである。ページフォルトが発生した場合、この割込ベクタ内のページフォルトにプログラムコードがジャンプし、結果的にページフォルトハンドラ４０４が呼び出される。

【００５６】
データ４０１、プログラムコード４０２、ＭＭＵテーブル４０３、および割込ベクタ４０５の論理アドレスに対する物理アドレスは、任意のアドレスにマッピングできる。
【００５７】
イメージ保存プログラム４０６は、所望のプログラムが起動された後、所望の状態でメモリイメージが不揮発メモリ１０４に保存されるようにするためのプログラムである。イメージ保存プログラム４０６の論理アドレスと物理アドレスは、同一アドレスにマッピングされる必要がある。
【００５８】
イメージ復帰プログラム４０７は、イメージ保存プログラム４０６の処理により保存された物理メモリイメージを、必要に応じて物理ページ単位に、データ４０１やプログラムコード４０２を、対応する物理ページに、不揮発メモリ１０４から読み込み、復帰させるためのプログラムである。イメージ復帰プログラム４０７の論理アドレスと物理アドレスは、同一アドレスにマッピングされる必要がある。
【００５９】
ブートローダ４０８は、電源投入もしくはリセット後に最初に起動されるブートローダである。主にブートローダ４０８は、起動に必要な最低限のＩ／Ｏの初期化を行う。このような構成をソフトウェアは有する。
【００６０】
［ソフトウェアの動作について］
次に、本発明を適用したソフトウェアの動作について説明する。まず、概要を説明し、その後詳細を説明する。本発明によれば、ソフトウェアの局所性を利用し、ソフトウェアを高速に起動することができる。ソフトウェアにはＯＳ等も含まれる。例えば、容量４GBのＲＡＭを備えるハードウエア上で、所定のソフトウェアを動作させると仮定する。その所定のソフトウェアのプログラムおよびデータの容量の合計が、仮に４GBであったとする。一般的にソフトウェアは、様々なモードや機能を有し、所定の単一の機能だけで、全容量の４GBを使う可能性は極めて低い。
【００６１】
例えばソフトウェアが、起動後、特定の状態でユーザからのキー入力を待つとする。一般的にはハードウエアのリセット後、ブートローダが起動され、このソフトウェアが立ち上がってユーザからのキー入力待ちの状態とされる。このソフトウェアを既知の技術であるハイバネーションを適用して高速に起動させた場合、メモリイメージを作成する準備として、ユーザからのキー入力待ちの状態でＣＰＵや各Ｉ／Ｏのレジスタが保存され、プログラムコードやデータの合計４GBが、何らかの不揮発メモリに格納されることになる。起動時はこの逆で、通常の起動プロセスは通らず、４GBのメモリが展開され、ＣＰＵやＩ／Ｏのレジスタが復帰され、キー入力の処理に戻ることになる。
【００６２】
上記ソフトウェアにおいて、「ユーザからの入力を待つ」という状態を考える。この状態では、キー入力の処理が繰り返されており、このようなキー入力に係わる処理に係わるプログラムコードやデータは比較的小さい。本発明は、この原理を利用して高速起動を実現する。動作は大きく分けると以下のようになる。
【００６３】
（Ａ）通常起動モードでＯＳや所望のソフトウェアが起動され、そのソフトウェアが所望の状態にされる
（Ｂ）イメージ保存プログラムが起動され、その起動されたイメージ保存プログラムにより、ＭＭＵ１３１の全てのページテーブルに対してのアクセスを禁止するために、アクセス禁止を示す情報に所定の情報が書き換えられた後、上記（Ａ）の状態のメモリイメージがレジスタに保存されて終了される
（Ｃ）次回以降は、高速起動モードにされることで、（Ａ）の所望の状態で起動される
【００６４】

このような高速起動を実現するにあたり、基本的な準備として、通常通りにソフトウェアが起動され、そのソフトウェアが所望の状態にされ、その後、メモリイメージやレジスタが保存される。高速起動時は、ハイバネーション起動とは異なり、全てのメモリイメージがメインメモリに展開されるのではなく、実際に使われるメモリ、即ちプログラムコードやデータが、必要に応じて一部分のみ小刻みに展開される。

【００６５】

小刻みにプログラムコードやメモリが、メインメモリに展開される方法として、本実施の形態としてはＣＰＵが持つMemory Management Unit(ＭＭＵ１３１)が利用される例をあげて説明している。ＯＳが、ＭＭＵ１３１を使用する場合もあるが、ＯＳが利用する前に、本発明が適用されたソフトウェアが元に戻し、ＯＳは、本発明が適用されたソフトウェアがＭＭＵ１３１を操作したことには関知しない。

【００６６】

具体的には、メモリイメージが保存される前に、ＭＭＵ１３１のテーブルの内容が書き換えられ、全てのページがアクセス禁止に設定される。また、本発明が適用されたソフトウェアは、前記ページをアクセス禁止にしたというマークを付与する機能も有する。

【００６７】

仮に、所望のソフトウェアがＯＳ上で動作し、そのＯＳがＭＭＵ１３１を使用する場合も同じである。高速起動時は、ＭＭＵ１３１のテーブルおよびＣＰＵのレジスタのみ先行で復帰される。そして、メモリイメージが作成された後のアドレスに戻される。ＭＭＵ１３１のテーブルは、全てアクセス禁止に設定されているので、戻りアドレスにジャンプした時にページフォルトが発生する。データアクセス時も同じである。ページフォルトを処理するページフォルトハンドラ４０４は、ページフォルトが発生したアドレスからページを計算し、且つ、本発明が適用されたソフトウェアが、マークしたマークをチェックし、そのページを不揮発メモリ１０４からメインメモリ（例えば、ＲＡＭ１０２）へ読み込み、ＭＭＵ１３１のテーブルを書き換え前の元の通りに書き戻す。

【００６８】

このような処理が繰り返されることで、大容量のメモリイメージであっても、所望の状態に復帰させるには最低限のメモリイメージの読み込みだけで済み、高速に起動することが可能となる。

【００６９】

［動作の詳細について］

上記してきたように、本発明によれば、所望のソフトウェアをある状態まで通常の起動と比較して高速に起動することが可能である。その高速起動を行う手順を大きく分けると、簡便に上記したが、（Ａ）、（Ｂ）、（Ｃ）の３つに分類される。さらに（Ａ）、（Ｂ）、（Ｃ）を、フローチャートを参照した説明の前に、さらに説明を加える。（Ａ）、（Ｂ）の一連の流れは、一度実行されれば、毎回行う必要がない処理である。通常、（Ｃ）から実行されるようにすることで、高速起動することが可能となる。

【００７０】

（Ａ）通常起動

（Ａ－１）　起動モード切替部１１０を通常起動モードに設定し、通常の方法により、ＯＳや所望のプログラムが起動される。

（Ａ－２）　所望のプログラムが起動された後、ソフトウェアが操作されて、ソフトウェアが所望の状態にされる。高速起動時は、この状態で起動される。

【００７１】

（Ｂ）状態の保存

（Ｂ－１）　何らかのキーやコマンドなどによりイメージ保存プログラム４０６が起動される。この起動に関する起動方法には、特に制限はない。

（Ｂ－２）　イメージ保存プログラム４０６は、次回以降に高速に起動したい状態で、メモリイメージとレジスタを保存する。具体的には、イメージ保存プログラム４０６は、ＭＭＵ１３１のテーブルを全てアクセス禁止状態に設定し、その時点でのデータ４０１、

プログラムコード４０２、ＭＭＵテーブル４０３、ページフォルトハンドラ４０４、割込ベクタ４０５、およびレジスタ類を不揮発メモリ１０４に保存させる。

【００７２】

（Ｃ）高速起動

（Ｃ－１）　起動モード切替部１１０が、高速起動モードに設定される。ブートローダ４０８が、起動モードを判断し、高速起動モードの場合は、イメージ復帰プログラム４０７を呼び出す。イメージ復帰プログラム４０７は、イメージ保存プログラム４０６が保存したＭＭＵテーブル４０３、ページフォルトハンドラ４０４、割込ベクタ４０５を復帰させる。

（Ｃ－２）　Ｂ－１の処理でイメージ保存プログラム４０６が起動された後のアドレスに戻る、即ちジャンプする。ＭＭＵ１３１が全てアクセス禁止状態に設定されているので、プログラムコード４０２やデータ４０１にアクセスされる毎に、対応するアドレスにてページフォルトが発生し、ページフォルトハンドラ４０４が呼ばれる。

（Ｃ－３）　ページフォルトハンドラ４０４は、対応する物理ページ２０７を不揮発メモリ１０４から１ページ分読み出し、ＭＭＵ１３１を元に戻す。

（Ｃ－４）　ページフォルトが次々に発生し、Ａ－２の状態になるまで、必要なページフォルトが発生し続ける。

（Ｃ－５）　Ａ－２の状態になるまで物理ページ２０７が読み込まれる。この処理で読み込まれる物理ページ２０７は、実行されるソフトウェアや、その状態にも依存するが、極めて少なく、従来のハイバネーション技術で行われていたように、全ての物理ページ２０７を読み込むのと比較して起動時間は著しく短くすることが可能となる。

【００７３】

図７乃至１１のフローチャートを参照し、上記（Ａ）、（Ｂ）、（Ｃ）の各動作についてさらに説明を加える。

【００７４】

図７のフローチャートは、上記（Ａ）、（Ｂ）の処理に該当する。すなわち、主に、電源投入からイメージ保存までの処理に係わるフローチャートである。ステップＳ１０１において、ハードディスクレコーダ１００（図１）の電源が投入、もしくはリセットが発生してシステムが起動される。

【００７５】

ステップＳ１０２において、ブートローダ４０８（図６）が起動される。このステップＳ１０２で起動されるブートローダ４０８は、ＯＳや所望のソフトウェアを動作させるために最低限のハードウエアの初期化や、必要に応じてＲＯＭ１０３やＨＤＤ１０８に格納されているソフトウェアをＲＡＭ１０２に転送させたりする処理を想定し、そのような処理を実行できるブートローダであればよい。ブートローダ４０８は、システムに依存するものであり必須のものではない。よって、システムによっては、このステップＳ１０２が省略される場合もある。

【００７６】

ステップＳ１０３において、起動モード切替部１１０の状態がチェックされ、通常起動モードか、高速起動モードかの遷移が切り替えられる（通常起動スイッチがＯＮであるか否かが判断される）。起動モード切替部１１０が通常起動モードの場合、ステップＳ１０４に処理が進められ、高速起動モードの場合、ステップＳ１６１（図９）に進められる。

【００７７】

ステップＳ１０３において、通常起動スイッチがＯＮであると判断されると、通常起動モードで起動するため、通常起動フラグがＯＮに設定される。通常起動フラグがＯＮに設定されると、ステップＳ１０５に処理が進められ、ＯＳが搭載されているシステムであれば、ＯＳが起動される。一般的なシステムの場合、ＯＳが起動される時にＭＭＵ１３１が初期化され、図２のＭＭＵ１３１のテーブルが作成される。本発明においては、ＯＳの搭載は必須ではないが、仮にＯＳを搭載しないシステムの場合、ＭＭＵ１３１の初期化を行う必要がある。また、ＯＳ搭載の場合であっても、ＯＳの種類等に制限はない。

【００７８】
ステップＳ１０６において、高速起動させたい所望のソフトウェアが起動される。ステップＳ１０７において、起動されたソフトウェアが動作する。この処理は、上記したＡ－２の処理に該当する。高速起動により起動した状態と同じ状態にソフトウェアを遷移させる。例えば、所望のソフトウェアに複数のモードがあったと仮定し、その中である特定のモードで高速起動させたいのであれば、ソフトウェアを操作し、そのモードにまで遷移させる。例えば、ハードディスクレコーダ１００の場合、予約するモード、再生するモード、設定するモードなどがあるが、ユーザが再生するモードをよく利用する場合、再生モードにまで遷移される。
【００７９】
ステップＳ１０８において、イメージ保存プログラム４０６（図６）の処理が開始されたか否かが判断される。この処理は、上記したＢ－１の処理に該当する。コマンドやキー操作、スイッチ等によってイメージ保存プログラム４０６による処理が開始される。このイメージ保存プログラム４０６の実行手段に制限はない。ステップＳ１０８において、イメージ保存プログラム４０６（図６）の処理は開始されていないと判断された場合、ステップＳ１０７に処理が戻され、それ以降の処理が繰り返される．すなわちこの場合、ソフトウェアの動作が継続される。
【００８０】
一方、ステップＳ１０７において、イメージ保存プログラム４０６（図６）の処理が開始されたと判断された場合、換言すれば、ソフトウェアの動作は終了したと判断された場合、ステップＳ１０９に処理が進められる。以下のステップＳ１０９乃至Ｓ１１６の処理は、上記したＢ－２の処理に該当する。また、ステップＳ１０９乃至Ｓ１１６の処理は、イメージ保存プログラム４０６が実行する処理である。
【００８１】
ステップＳ１０９において、図１に示したＩ／Ｏ部１０９のレジスタが保存される。基本的には、設定されている値が取得され、保存される。Ｉ／Ｏに関しては、全てのレジスタが読み込める仕様のＩ／Ｏとは限らないので、その場合は個別に対応する必要がある。なお、Ｉ／Ｏの種類や仕様は任意であり、特に本発明を適用するうえでの制限はない。
【００８２】
ステップＳ１１０において、ＣＰＵ１０１のレジスタが保存される。基本的にはＣＰＵ１０１の全レジスタが保存される。ＣＰＵ１０１やレジスタの種類は任意であり、特に本発明を適用するうえでの制限はない。
【００８３】
ステップＳ１１１において、アドレス空間の切り換えが行われる。ＣＰＵ１０１は、通常、仮想アドレスモードで動作している。このモードが、仮想アドレスモードから物理アドレスモードに遷移される。仮想アドレスモードから物理アドレスモードへの遷移方法は、ＭＭＵ１３１のアーキテクチャに依存するため、本発明を適用するうえでの遷移方法に制限はない。また、仮想アドレスモードから物理アドレスモードに遷移させ場合、アドレス空間が変化するため、ステップＳ１１１における処理では、論理アドレスと物理アドレスとで同一アドレス空間にマッピングされる必要がある。
【００８４】
ステップＳ１１２において、キャッシュフラッシュが実行される。ＣＰＵ１０１が、ＴＬＢ（Translation Look-aside Buffer）、一次キャッシュ、二次キャッシュを搭載し、それらが有効だった場合、ＴＬＢおよびキャッシュがフラッシュされる必要がある。これは、次のステップＳ１１３において、ＲＡＭ１０２上に置かれたＭＭＵテーブル４０３の内容を書き換える必要があり、キャッシュに格納されているデータが、全てＲＡＭ１０２に反映されている必要があるからである。このステップＳ１１２におけるキャッシュフラッシュの処理は、必要に応じて行われ、場合によっては省略されても良い処理である。
【００８５】
ステップＳ１１３において、ＭＭＵ１３１のＭＭＵテーブル４０３が、全ての物理ペー

ジ２０７へのアクセスを禁止する情報に書き換えられる。このステップＳ１１３におけるＭＭＵテーブル書き換え処理については、図８のフローチャートを参照して後述する。

【００８６】

ステップＳ１１３において、ＭＭＵ１３１のＭＭＵテーブル４０３が書き換えられると、ステップＳ１１４に処理が進められる。ステップＳ１１４において、キャッシュフラッシュが実行される。ＣＰＵ１０１がＴＬＢ、一次キャッシュ、二次キャッシュを搭載し、それらが有効だった場合、ＴＬＢおよびキャッシュがフラッシュされる必要がある。これは前段のステップＳ１１３の処理で書き換えたＭＭＵ１３１のＭＭＵテーブル４０３の内容を、確実にＲＡＭ１０２へ反映させるためである。このステップＳ１１４におけるキャッシュフラッシュの処理は、必要に応じて行われ、場合によっては省略されても良い処理である。

【００８７】

ステップＳ１１５において、不揮発メモリ１０４に対して、ＲＡＭ１０２の全容量の内容が全て保存される。ＲＡＭ１０２のアドレスに対する不揮発メモリ１０４の相対的なアドレス位置が一致している必要がある。例えば、ＲＡＭ１０２の物理アドレスが 0x10000000 から 0x1fffffff にマッピングされていたとする。この場合、例えば不揮発メモリ１０４に対しては、0x40000000 から 0x4fffffff のアドレスにより、データが読み込める必要がある。

【００８８】

この例の場合、ＲＡＭ１０２のアドレスに対する不揮発メモリ１０４のオフセットは 0x30000000 となるため、ＲＡＭ１０２に対するアドレスであっても、0x30000000 のオフセットを加えるだけで不揮発メモリ１０４内のアドレスに変換することが可能となる。不揮発メモリ１０４は、必ずしもメモリマップ上にマッピングされる必要はない。上記オフセットを加えたアドレスをキーにして読み込みができれば良い。また、不揮発メモリ１０４に対する保存方法は、アーキテクチャに依存するが、本発明の適用するうえで、その保存方法に制限はない。

【００８９】

ステップＳ１１６で、イメージ保存プログラム４０６での処理が終了される。イメージ保存プログラム４０６での処理が終了されることで、電源がＯＦＦまたはリセット(RESET)することができる状態となる。

【００９０】

図７に示したフローチャートにおいて、ステップＳ１０３において、通常起動スイッチがＯＮではないと判断されたときの処理と、ステップＳ１１３でのＭＭＵテーブル書き換え処理についての詳細な説明が残っているが、まずここでは、図８のフローチャートを参照し、ステップＳ１１３でのＭＭＵテーブル書き換え処理についての詳細な説明を行う。

【００９１】

図８に示したフローチャートに基づく処理は、ＭＭＵ１３１のＭＭＵテーブル４０３が、図２で示した構成になっており、このＭＭＵテーブル４０３を書き換える処理である。

【００９２】

ステップＳ１３１において、ＭＭＵ１３１のＭＭＵテーブル４０３の書き換えが開始されると、まず、変数レベル1ディスクリプタポインタに、レベル１ディスクリプタテーブル２０１の先頭アドレスが代入される。ステップＳ１３２において、変数レベル１ディスクリプタポインタが指すアドレスからレベル1ディスクリプタ２０３が取得される。

【００９３】

ステップＳ１３３において、ステップＳ１３２の処理で取得されたレベル１ディスクリプタ２０３内にレベル２ディスクリプタテーブル２０４へのポインタが存在するか否かが判断される。ステップＳ１３３において、レベル１ディスクリプタ２０３内にレベル２ディスクリプタテーブル２０４へのポインタが存在すると判断された場合、ステップＳ１３６に処理が進められ、レベル１ディスクリプタ２０３内にレベル２ディスクリプタテーブル２０４へのポインタが存在しないと判断された場合、ステップＳ１３４に処理が進めら

れる。

【００９４】

ステップＳ１３４において、変数レベル１ディスクリプタポインタが次のレベル1ディスクリプタポインタのアドレスに移動される。そして、ステップＳ１３５に処理が進められ、レベル1ディスクリプタポインタが最終に到達したか否かが判断される。

【００９５】

ステップＳ１３５において、レベル1ディスクリプタポインタが最終に到達したと判断されるまで、ステップＳ１３４に処理が戻され、変数レベル１ディスクリプタポインタが次のレベル1ディスクリプタポインタのアドレスに移動されるといった処理が繰り返される。そして、ステップＳ１３５において、レベル1ディスクリプタポインタが最終に到達したと判断されると、処理はステップＳ１１４(図７)に進められる。すなわち、ＭＭＵテーブルの書き換えが終了したと判断され、図７に示したフローチャートの処理に、処理が戻される。

【００９６】

一方、ステップＳ１３３において、レベル１ディスクリプタ２０３内にレベル２ディスクリプタテーブル２０４へのポインタが存在すると判断された場合、ステップＳ１３６に処理が進められる。ステップＳ１３６において、変数レベル２ディスクリプタポインタに、レベル２ディスクリプタテーブル２０４の先頭アドレスが代入される。

【００９７】

ステップ１３７において、変数レベル２ディスクリプタポインタが指すアドレスからレベル２ディスクリプタ２０６が取得される。ステップＳ１３８において、ステップＳ１３７の処理で取得されたレベル２ディスクリプタ２０６内に物理ページ２０７が存在するか否かが判断される。ステップＳ１３８において、取得されたレベル２ディスクリプタ２０６内に物理ページ２０７が存在すると判断された場合、ステップＳ１３９に処理が進められ、取得されたレベル２ディスクリプタ２０６内に物理ページ２０７は存在しないと判断された場合、ステップＳ１４３に処理が進められる。

【００９８】

ステップＳ１３９において、ステップＳ１３７の処理で取得されたレベル２ディスクリプタ２０６内の物理ページ２０７が、ステップＳ１１５(図７)にて保存の対象となるＲＡＭ１０２内のアドレスの範囲内であるか否かが判断される。ステップＳ１３９において、レベル２ディスクリプタ２０６内の物理ページ２０７が、保存の対象となるＲＡＭ１０２内のアドレスの範囲内であると判断された場合、ステップＳ１４０に処理が進められ、保存の対象となるＲＡＭ１０２内のアドレスの範囲内ではないと判断された場合、ステップＳ１４３に処理が進められる。

【００９９】

ステップＳ１４０において、ステップＳ１３７で取得されたレベル２ディスクリプタ２０６のアクセス許可bit（図３のアクセス許可bit３０５)がチェックされ、その物理ページ２０７がアクセス許可にされているか否かが判断される。ステップＳ１４０において、物理ページ２０７へのアクセスが許可されていると判断された場合、ステップＳ１４１に処理が進められ、物理ページ２０７へのアクセスは許可されていないと判断された場合、ステップＳ１４３に処理が進められる。

【０１００】

ステップＳ１４１において、ステップＳ１３７の処理で取得されたレベル２ディスクリプタ２０６のアクセス許可bit３０５がアクセス禁止を表すbitに書き換えられる。そして、ステップＳ１４２において、書き換えられたレベル２ディスクリプタ２０６に対してマーキングが実行される。この処理は、ステップＳ１３７の処理で取得されたレベル２ディスクリプタ２０６のアクセス許可bit３０５が、本発明が適用されたソフトウェアにより書き換えられたのか、他のソフトウェア、例えば、本来のＯＳ等の動作によって書き換えられたのかを識別するための情報を保存する（マーキングする）ために行われる。

【０１０１】

ステップＳ１４２におけるマーキングの仕方については、アーキテクチャに依存し、本発明を適用するうえでの制限はない。例えば、レベル２ディスクリプタ２０６に使われていない空きbitが存在するのであれば、その空きbitをマーキングの情報を埋め込むbitとして使用することができる。また、別途テーブルを持たせてマーキングされたところとされていないところとが管理されるようにしても良い。いずれにせよ、仮にＯＳ等が搭載されたシステムで、かつ、ＯＳがこれらのbitを使用していた場合は、共存するような仕組みにすることで、本発明を実施することが可能である。

【０１０２】

ステップＳ１４３において、変数レベル２ディスクリプタポインタが、次のレベル２ディスクリプタ２０６のポインタのアドレスに移動される。このステップＳ１４３への処理には、ステップＳ１３８において、レベル２ディスクリプタ２０６内に物理ページ２０７が存在しないと判断された場合、ステップＳ１３９において、レベル２ディスクリプタ２０６がＲＡＭ１０２を指していないと判断された場合、または、ステップＳ１４０において、物理ページ２０７へのアクセスは許可されていないと判断された場合にも来る。

【０１０３】

ステップＳ１４４において、レベル２ディスクリプタポインタが最終に到達したか否かが判断される。ステップＳ１４４において、レベル２ディスクリプタポインタが最終に到達したと判断されるまで、ステップＳ１３７に処理が戻され、それ以降の処理が繰り返される。一方、ステップＳ１４４において、レベル２ディスクリプタポインタが最終に到達したと判断された場合、ステップＳ１３４に処理が進められる。ステップＳ１３４以降の処理については既に説明したので、その説明は省略する。

【０１０４】

このようにして、ＭＭＵ１３１のＭＭＵテーブル４０３が書き換えられる。

【０１０５】

次に、高速起動時の処理について説明する。高速起動は、ステップＳ１０３において、通常起動スイッチがＯＮになっていないと判断されたとき、すなわち、高速起動にスイッチが切り替えられていると判断されたときに実行される。図９のフローチャートは、ステップＳ１０３において、通常起動スイッチがＯＮにはなっていないと判断されたときに処理が進められるフローチャートであり、高速起動時の処理について説明するためのフローチャートである。

【０１０６】

ステップＳ１６１において、高速起動モードで起動するため、通常起動フラグがＯＦＦ（高速起動フラグがＯＮ）に設定される。ステップＳ１６２において、必要に応じて、割込ベクタ４０５、ページフォルトハンドラ４０４、ＭＭＵテーブル４０３が、イメージが保存された時と同じＲＡＭ１０２のアドレスに読み込まれる。

【０１０７】

ステップＳ１６３において、ＭＭＵ１３１のＭＭＵテーブルが読み込まれる。このステップＳ１６３において実行されるＭＭＵテーブル読み込み処理については、図１０のフローチャートを参照し、後述する。

【０１０８】

ＭＭＵテーブルの読み込みが終わると、ステップＳ１６４に処理が進められる。ステップＳ１６４において、ＣＰＵ１０１のアドレス空間が、物理アドレスモードから仮想アドレスモードに遷移される。物理アドレスモードから仮想アドレスモードに遷移させる場合、アドレス空間が変化するため、ステップＳ１６４における処理では、論理アドレスと物理アドレスが同一アドレス空間にマッピングされる。

【０１０９】

ステップＳ１６５において、ステップＳ１１０(図７)で保存されたＣＰＵ１０１のレジスタの値が不揮発メモリ１０４から読み出され、ＣＰＵ１０１に対して復帰される。ＣＰＵ１０１やレジスタの種類は任意であり、本発明を適用する上での制限はない。

【０１１０】

ステップＳ１６６において、ステップＳ１０９(図７)で保存されたＩ／Ｏのレジスタの値が、不揮発メモリ１０４から読み出され、Ｉ／Ｏ部１０９に対して復帰される。Ｉ／Ｏの種類や仕様は任意であり、本発明を適用する上での制限はない。

【０１１１】

このように、レジスタなどが復帰されると、ステップＳ１０７(図７)に処理が進められる。ステップＳ１０７において、ソフトウェアが動作する。この場合、ステップＳ１０４乃至Ｓ１０６の処理が実行されずに、ステップＳ１０７においてソフトウェアが動作開始となる。よって、ステップＳ１０４乃至Ｓ１０６の処理が実行される分だけ、少なくともソフトウェアが動作開始できるまでにかかる時間が短縮できることになる。特に、ステップＳ１０５におけるＯＳの起動やＭＭＵの初期化にかかる時間、およびステップＳ１０６におけるソフトウェアの起動にかかる時間をなくすことができることで、大幅な時間の短縮を期待することができる。

【０１１２】

図９のフローチャートの説明に戻り、ステップＳ１６３で実行されるＭＭＵテーブル読み込み処理の詳細について、図１０のフローチャートを参照して説明する。

【０１１３】

ステップＳ１８１において、ＭＭＵテーブル４０３の読み出しが開始されると、まずレベル1ディスクリプタテーブル２０１が読み出される。このレベル1ディスクリプタテーブル２０１は、ステップＳ１１５(図７)の処理で、不揮発メモリ１０４にＲＡＭ１０２の内容が保存されたが、その不揮発メモリ１０４に保存されている内容から、レベル1ディスクリプタテーブル２０１のみが読み出される。

【０１１４】

ステップＳ１８２において、変数レベル1ディスクリプタポインタに、レベル1ディスクリプタテーブル２０１の先頭アドレスが代入される。ステップＳ１８３において、変数レベル1ディスクリプタポインタが指すアドレスからレベル1ディスクリプタ２０３が取得される。ステップＳ１８４において、ステップＳ１８３の処理で取得されたレベル1ディスクリプタ２０３内にレベル２ディスクリプタテーブル２０４へのポインタが存在するか否かが判断される。

【０１１５】

ステップ１８４において、取得されたレベル1ディスクリプタ２０３内にレベル２ディスクリプタテーブル２０４へのポインタが存在すると判断された場合、ステップＳ１８７に処理が進められ、取得されたレベル1ディスクリプタ２０３内にレベル２ディスクリプタテーブル２０４へのポインタは存在しないと判断された場合、ステップＳ１８５に処理が進められる。

【０１１６】

ステップＳ１８５において、変数レベル1ディスクリプタポインタが、次のレベル1ディスクリプタポインタのアドレスに移動される。そして、ステップＳ１８６において、レベル1ディスクリプタポインタが、最終に到達したか否かが判断される。ステップＳ１８６において、レベル1ディスクリプタポインタが、最終に到達したと判断された場合、ステップＳ１６４（図９）に処理が進められる。すなわちこの場合、ＭＭＵテーブル４０３の読み込みが完了されたため、次の処理へ処理が進められる。

【０１１７】

一方、ステップＳ１８６において、レベル1ディスクリプタポインタ２０３は、最終に到達していないと判断された場合、ステップＳ１８３に処理が戻され、それ以降の処理が繰り返される。ステップＳ１８３乃至Ｓ１８６が繰り返され、ステップＳ１８４において、取得されたレベル1ディスクリプタ２０３内にレベル２ディスクリプタテーブル２０４へのポインタが存在すると判断されると、ステップＳ１８７に処理が進められる。

【０１１８】

ステップＳ１８７において、ステップＳ１８３で取得されたレベル1ディスクリプタ２０３内にある、レベル２ディスクリプタテーブル２０４へのポインタが、ＲＡＭ１０２を

指しているか否かが判断される。ステップＳ１８７において、レベル２ディスクリプタテーブル２０４へのポインタが、ＲＡＭ１０２を指していると判断された場合、ステップＳ１８８に処理が進められ、レベル２ディスクリプタテーブル２０４へのポインタは、ＲＡＭ１０２を指していないと判断された場合、ステップＳ１８５に処理が進められ、それ以降の処理が繰り返される。

【０１１９】

ステップＳ１８８において、レベル２ディスクリプタテーブルが読み出される。このレベル２ディスクリプタテーブル２０４は、ステップＳ１１５(図７)の処理で、不揮発メモリ１０４にＲＡＭ１０２の内容として保存され、その不揮発メモリ１０４に保存されている内容から、レベル２ディスクリプタテーブル２０４のみが読み出される。その後、処理は、ステップＳ１８５に進められ、それ以降の処理が繰り返される。

【０１２０】

このようにして、ＭＭＵテーブルの読み出しが行われる。

【０１２１】

次に、図１１のフローチャートを参照し、ページフォルトが発生したときに実行される処理について説明を加える。ステップＳ２０１において、ページフォルトが発生すると、割込ベクタ４０５にジャンプされる。すなわち、割込ベクタ４０５から、実際のページフォルト処理を行う割込ハンドラへのジャンプが実行される。

【０１２２】

一般的なＣＰＵ１０１は、ページフォルトが発生した場合、割込処理として特定の割込ベクタにジャンプし、その割込ハンドラとして処理する。図１１に示したページフォルト発生時の処理に係わるフローチャートは、ページフォルトの割込を処理する割込ハンドラを想定しているが、これらはＣＰＵ１０１のアーキテクチャに依存する。なお、本発明の適用は、ＣＰＵ１０１のメーカや型番により、制限が加えられることはない。

【０１２３】

ステップＳ２０２において、通常起動フラグがＯＮであるか否かがが判断される。通常起動フラグは、例えば、ステップＳ１０４(図７)の処理でＯＮに設定される。ステップＳ２０２において、通常起動フラグがＯＮであると判断された場合、換言すれば、通常起動であると判断された場合、ステップＳ２０７に処理が進められる。一方、ステップＳ２０２において、通常起動フラグがＯＮではないと判断された場合、換言すれば、高速起動であると判断された場合、ステップＳ２０３に処理が進められる。

【０１２４】

ステップＳ２０３において、対象となる物理ページ２０７、即ちページフォルトが発生したアドレスに対応する物理ページ２０７が、マーキングされた物理ページ２０７であるか否かが判断される。マーキングは、ステップＳ１４２（図８）の処理で実行されたマーキングである。すなわち、マーキングされている物理ページ２０７は、本発明を適用したソフトウェアにより、アクセス禁止に書き換えられた物理ページ２０７である。

【０１２５】

なお、一般的に物理ページ２０７とアドレスには、以下の計算式が満たされるが、これらはＣＰＵ１０１のアーキテクチャに依存するので、特にこの計算式に本発明の適用範囲が限定されるものではない。

物理ページ = アドレス／ページサイズ（例えば、上記した例では４KB）

この式が示すように物理ページは、アドレスをページサイズで除算したものとなる。

【０１２６】

ステップＳ２０３において、ページフォルトが発生したアドレスに対応する物理ページ２０７は、マーキングされた物理ページ２０７であると判断された場合、ステップＳ２０４に処理が進められ、マーキングされていない物理ページ２０７であると判断された場合、ステップＳ２０７に処理が進められる。

【０１２７】

ステップＳ２０４において、対象となる物理ページ２０７、即ちページフォルトが発生

したアドレスに対応する物理ページ２０７が、ステップＳ１１５（図７）の処理で不揮発メモリ１０４に保存されたイメージから４KB分のみ読み出される。

【０１２８】

ステップＳ２０５において、対象となる物理ページ２０７、即ちページフォルトが発生したアドレスに対応する物理ページ２０７の、レベル２ディスクリプタ２０６のアクセス許可bit３０５がアクセス許可に書き換えられる。ステップＳ２０６において、ステップＳ１４２（図８）の処理でマーキングした識別情報が解除される。

【０１２９】

このようにして、ページフォルトが発生されたときの処理が実行されることで、高速起動が実現される。

【０１３０】

一方、ページフォルトが発生したが、ステップＳ２０２において、通常起動フラグがＯＮであると判断された場合、または、ステップＳ２０３において、対象ページはマーキングされていないと判断された場合、ステップＳ２０７に処理が進められる。ステップＳ２０７において、標準のページフォルトの処理が実行される。すなわち、通常起動のときや、本発明が適用されたソフトウェア以外のソフトウェア（ＯＳなど）が、アクセスを不許可に設定していたような場合、通常起動や、アクセス不許可時の処理が実行される。

【０１３１】

本発明を適用したシステムに、ＯＳ等が搭載されていた場合、標準的なページフォルトハンドラが通常実装されている。それに対して、上記したようなページフォルト機能がシステムに実装された場合、ＯＳ本来が行うページフォルト処理が、改めて実行される必要があるときもある。このステップＳ２０７の処理は、ＯＳ等のシステムに依存するものであり必須の処理ではないため、本実施の形態として省略することも可能である。

【０１３２】

このようにして、ページフォルトが発生したときの処理が実行されることで、高速起動が実現される。

【０１３３】

［効果］

上記したように、Memory Management Unit(ＭＭＵ)、もしくは、ＭＭＵ相当のメモリ管理機能を搭載したコンピュータシステム上で、ＭＭＵのテーブルに対し、ソフトウェアの動作に必要なＲＡＭの最小単位、いわゆるページに対して、全てのページにてページフォルトが発生するようにページテーブルエントリを書き換え、起動時は、アクセスするＲＡＭに対して発生したページフォルトに対し、本来ＯＳ等が有する既知の技術であるページイン・ページアウトとしての機能だけでなく、発生したページフォルトの機能を保存したメモリイメージのページ単位の読み込みに用いることで、以下のような効果がある。

【０１３４】

まず、必要最小限のメモリイメージの読み込み容量を実現することが可能となる。このことにより、例えば、パーソナルコンピュータの起動時間を短縮することが可能となる。具体的には、従来、数十秒から数分を要していた起動時間を、数秒以内で起動させることができるようになる。

【０１３５】

また、デジタル家電の起動時間を短縮することが可能となる。テレビジョン受像器やハードディスクレコーダなどのデジタル家電（電子機器）には、ＯＳ（所定のソフトウェア）を搭載した機種がある。所定のソフトウェアを搭載した機器の場合、起動時間が長くなることがあるが、本発明を適用することで、これらのデジタル家電における起動時間を短縮できるようになる。

【０１３６】

また、バッテリの寿命を延命化することが可能となる。従来の高速起動を実現させる方法として、ＣＰＵやメモリを省電力モードに移行させる方法があった。この方法では、省電力モードとは言え、電力は必要であり、バッテリで動く機器にとってはその消費電力は

無視できない。本発明を適用することで、不揮発メモリに起動イメージを保存することが可能となり、電源をＲＡＭに供給したまま停止させる、いわゆるサスペンド（従来の省電力モードに対応）を使う必要がなくなる。よって、結果的に、バッテリ寿命を著しく伸ばすことができる。

【０１３７】

さらに、家電製品の省エネルギー化を実現することが可能となる。テレビジョン受像器やハードディスクレコーダなどは一般的に起動が遅く、そのために「高速起動モード」なるモードを持つ機種も存在している。しかしながら、この「高速起動モード」は、家電製品を高速に起動させるため、常に電源を投入していることで、高速起動を実現している。そのため電力は電源投入時と同等に消費されている。

【０１３８】

しかしながら、本発明を適用することで、起動時間を短くすることができるため、「高速起動モード」のときにかかる起動時間と同等、またはそれ以上に短い起動時間で起動させることができるようになり、「高速起動モード」を設ける必要がなくなる。したがって、「高速起動モード」のときに、常に電源を投入してなくてはならないといった状態をなくすことが可能となるため、結果として省エネルギー化を実現することが可能となる。

【０１３９】

上記したコンピュータが実行するプログラムは、本明細書で説明する順序に沿って時系列に処理が行われるプログラムであっても良いし、並列に、あるいは呼び出しが行われたとき等の必要なタイミングで処理が行われるプログラムであっても良い。また、専用のハードウエアで構成することもできる。また、本明細書において、システムとは、複数の装置により構成される装置全体を表すものである。

【０１４０】

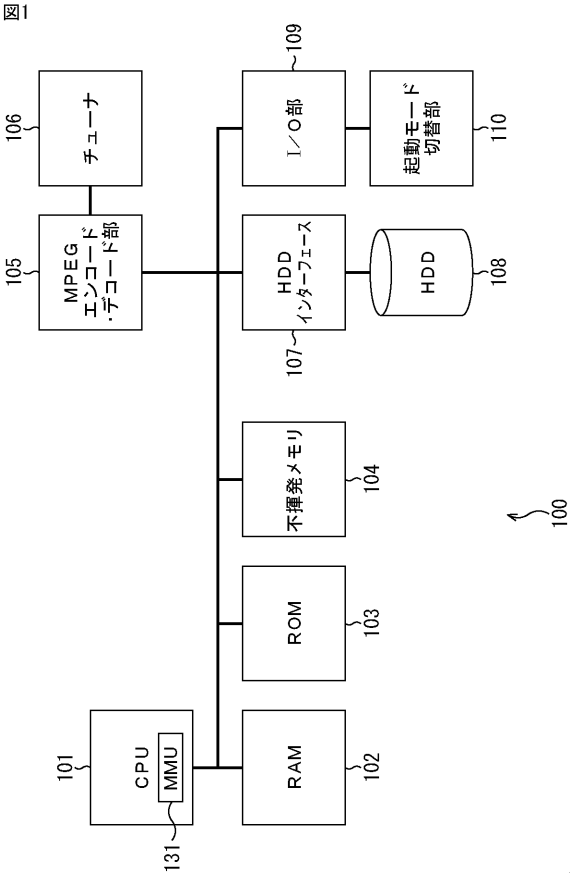
なお、本発明の実施の形態は、上述した実施の形態に限定されるものではなく、本発明の要旨を逸脱しない範囲において種々の変更が可能である。

【符号の説明】

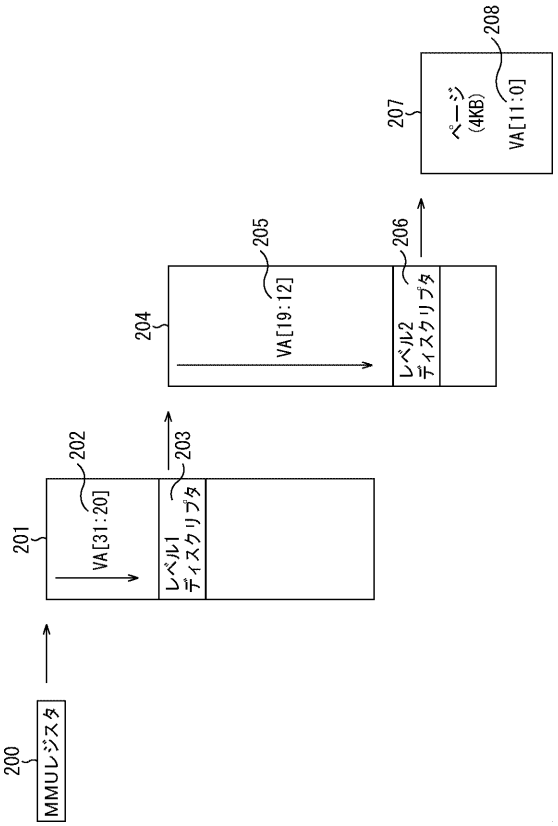
【０１４１】

１００　ハードディスクレコーダ，　１０１　ＣＰＵ，　１０２　ＲＡＭ，　１０３　ＲＯＭ，　１０４　不揮発メモリ，　１０５　ＭＰＥＧエンコード・デコード部，　１０６　チューナ，　１０７　ＨＤＤインターフェース，　１０８　ＨＤＤ，　１０９　Ｉ／Ｏ部，　１１０　起動モード切替部

【図１】
図1

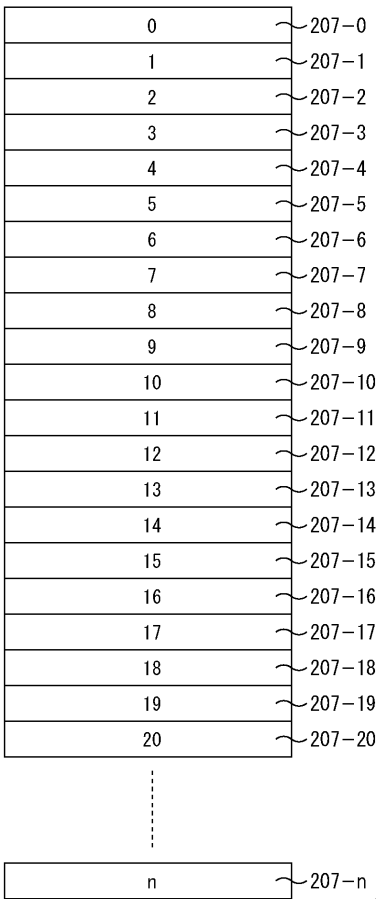


【図２】
図2



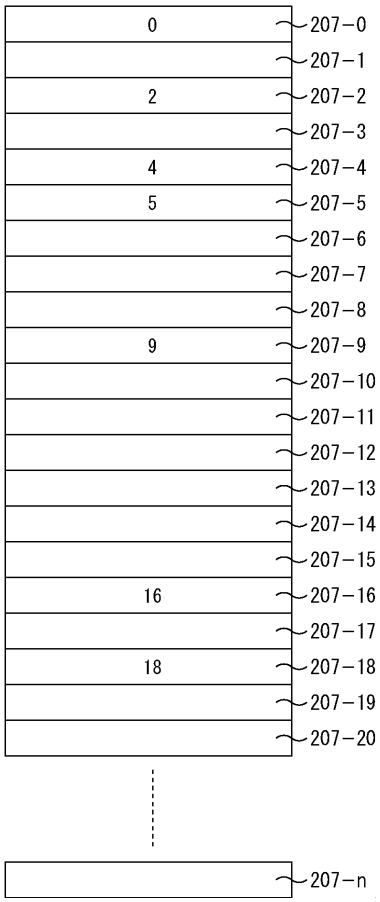
【図３】
図3

【図４】
図4



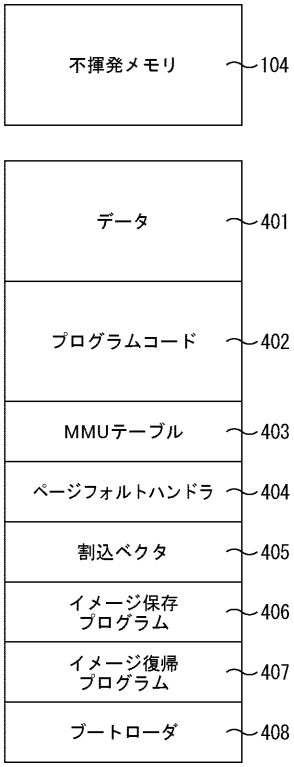
【図５】

図5



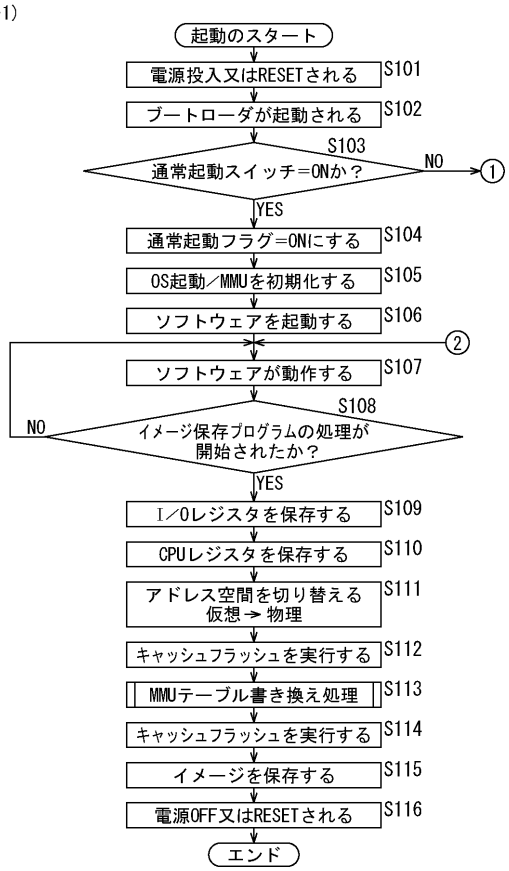
【図６】

図6



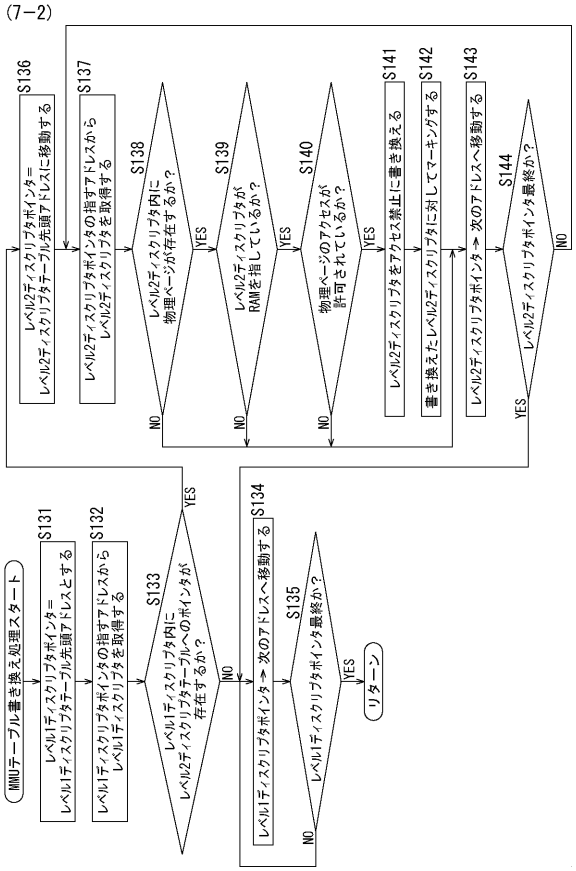
【図７】

図7
(7−1)



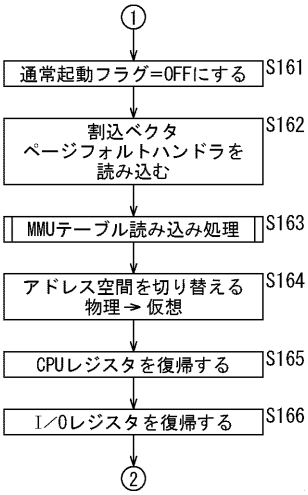
【図８】

図8
(7−2)



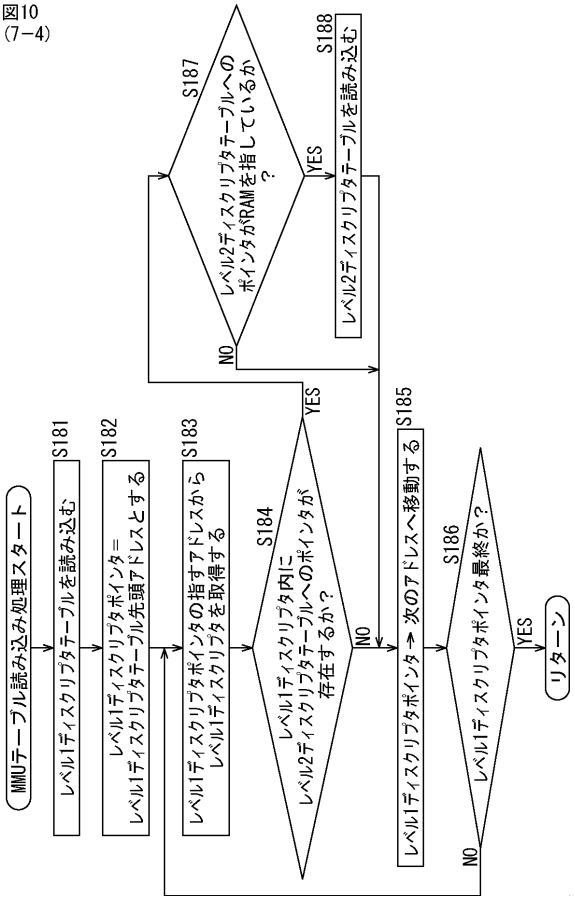
【図９】

図9
(7-3)



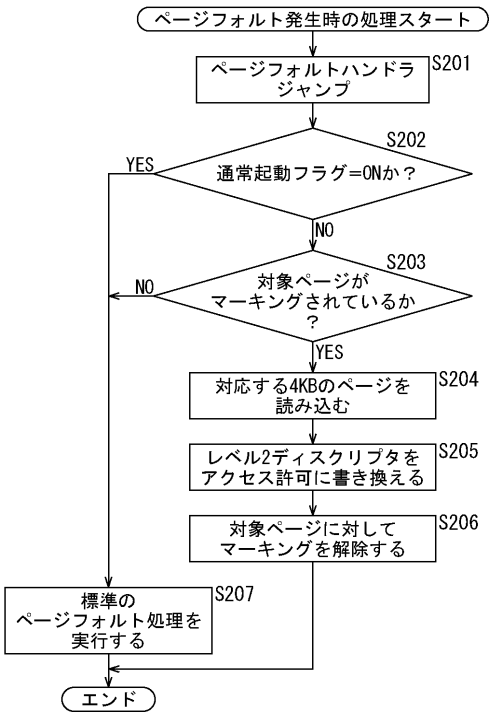
【図１０】

図10
(7-4)



【図１１】

図11



フロントページの続き

(56)参考文献 国際公開第２００６／１０７０９５（WO，Ａ１）
特開２００５－３１６８０９（ＪＰ，Ａ）
特開２００７－３３４３８３（ＪＰ，Ａ）
特開２００６－２０２２５２（ＪＰ，Ａ）
特開２００５－１４９２２５（ＪＰ，Ａ）
高橋 雅彦，Embedded Optimization (1) Fast Booting，第６回CELFテクニカルジャンボリー，２００６年 1月２０日，pp.2-11，ＵＲＬ，http://elinux.org/images/2/2c/Fastbooting-jpn20060120.pdf
内田 泰，機器設計の勘所 デジタル家電の“瞬間起動”技術 Ｌｉｎｕｘ搭載機器でハイバネーション，日経エレクトロニクス，日経ＢＰ社，２００８年 6月 2日，第９７９号，pp.128-140

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ 9／４４５
Ｇ０６Ｆ 9／５４
Ｇ０６Ｆ １２／０８ － １２／１０