(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4164452号
(P4164452)
(45)発行日　平成20年10月15日(2008.10.15)　(24)登録日　平成20年8月1日(2008.8.1)

(51)Int.Cl.　F I
G06F 12/08 (2006.01)　G06F 12/08 523C
G06F 12/08 517B
G06F 12/08 531C
G06F 12/08 541Z

請求項の数 9　(全 10 頁)

(21)出願番号　特願2004-25349(P2004-25349)
(22)出願日　平成16年2月2日(2004.2.2)
(65)公開番号　特開2005-216220(P2005-216220A)
(43)公開日　平成17年8月11日(2005.8.11)
審査請求日　平成18年11月29日(2006.11.29)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(72)発明者　小川　武志
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

最終頁に続く

(54)【発明の名称】情報処理方法及び装置

(57)【特許請求の範囲】
【請求項１】
メモリの有する同一のメモリエリアを、複数の異なるアドレスセットでＣＰＵによりアクセス可能に構成したメモリ回路と、
前記複数の異なるアドレスセットの一方のアドレスセットをキャッシャブル用のアドレスセットとし、他方をアンキャッシャブル用のメモリアドレスセットとしてキャッシュ制御するキャッシュ手段と、
前記メモリに所定サイズのエリアが要求されると、当該要求された所定サイズにキャッシュ単位であるラインサイズの２倍のサイズを付加して、前記メモリの前記アンキャッシャブル用のメモリアドレスセットによりアクセス可能なメモリエリアに割当てる割当て手段と、
を有することを特徴とする情報処理装置。
【請求項２】
更に、前記メモリに確保されたエリアに対してキャッシュのフラッシュ及びクリアを行う手段を有することを特徴とする請求項１に記載の情報処理装置。
【請求項３】
前記所定サイズのエリアの前後それぞれに、前記キャッシュ単位であるラインサイズのデータエリアが付加されることを特徴とする請求項１に記載の情報処理装置。
【請求項４】
前記メモリの前記所定サイズのエリアのデータをＤＭＡ転送するＤＭＡ制御手段を更に

有することを特徴とする請求項１に記載の情報処理装置。
【請求項５】
ＣＰＵ、メモリ、キャッシュメモリ及びキャッシュコントローラを具備する情報処理装置で実行される情報処理方法であって、
前記メモリに所定サイズのエリアが要求されると、当該要求された所定サイズにキャッシュ単位であるラインサイズの少なくとも２倍のサイズを付加して前記メモリに前記エリアを確保する工程と、
前記エリアを前記メモリのアンキャッシャブル用のメモリエリアに割当てる工程と、
を有することを特徴とする情報処理方法。
【請求項６】
前記確保する工程では、前記所定サイズのエリアの前後それぞれに、前記キャッシュ単位であるラインサイズのデータエリアが付加されることを特徴とする請求項５に記載の情報処理方法。
【請求項７】
更に、前記メモリに確保されたエリアに対してキャッシュのフラッシュ及びクリアを行う工程を有することを特徴とする請求項５に記載の情報処理方法。
【請求項８】
前記メモリの前記所定サイズのエリアのデータをＤＭＡ転送するＤＭＡ制御工程を更に有することを特徴とする請求項５に記載の情報処理方法。
【請求項９】
請求項５乃至８のいずれか１項に記載の情報処理方法を、コンピュータに実行させることを特徴とするプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、キャッシュ機能を備えた情報処理装置及びその方法に関するものである。
【背景技術】
【０００２】
マイクロコンピュータの主記憶として、安価で大容量なＤＲＡＭを利用する場合、ＤＲＡＭを含むバスよりもＣＰＵチップ内のバスを高速に設計することが可能であり、ＣＰＵ内部バスのクロックを高くして設計することでコンピュータの処理能力を高めることができる。このようなＣＰＵを備えたコンピュータシステムにおいて、ＣＰＵから外部バスへのメモリアクセスを効率的に行うために、キャッシュメモリのハードウェアを内部バスと外部バスの間に介在させるのが一般的である。
【０００３】
このようなキャッシュメモリは、ＣＰＵの内部バスと外部バスの間に存在しているので、ＣＰＵ側から見ると透過的だが、例えばＤＭＡ等において外部バスを介して外部メモリの読み書きを行った場合は、キャッシュ内にあるデータと主記憶内にあるデータとが一致しない場合が発生する。このような不具合を避けるために、ＤＭＡ転送を行う前後に、そのＤＭＡ転送対象のメモリエリアに対してキャッシュメモリのクリアやフラッシュ（キャッシュ内容をメモリに書き戻す）を行う方法と、ＤＭＡ転送の対象となるメモリエリアをアンキャッシャブルにしておく方法がある（例えば、特許文献１）。本願発明は、後者のアンキャッシャブル領域を割り当てる方法に関する。
【０００４】
ＤＭＡ転送に使用するメモリ領域を動的に割り当てるＡＰＩを持った組み込み用のＯＳが広く利用されている。このようなＯＳはメモリマネージャからメモリエリアを取得し、キャッシュコントローラを制御して、その取得したメモリエリアをアンキャッシャブルに設定できる。一般にキャッシュコントローラはメモリエリアの設定機能を備えており、その機能を用いて、メモリエリアに対してキャッシャブルエリアとアンキャッシャブルエリアを複数指定できる。しかしながら、このようなエリアを指定するためのレジスタの数が

有限であり、また指定できるエリアが多いほど回路が冗長になるため、無制限にこれらエリアを指定できるわけではない。

【０００５】

一方、ＤＭＡを利用するアプリケーションソフトウェアでは、１つのＤＭＡチャネルに対して複数のバッファを用意して処理の効率化を図っている。例えば一つ目のバッファへのＤＭＡ転送中には、二つ目のバッファに対するＤＭＡの準備や後処理を行っている。

【特許文献１】特開２０００－１８１７９６号公報

【発明の開示】

【発明が解決しようとする課題】

【０００６】

上記事情に鑑みると、システムで利用できるＤＭＡのチャネル数よりも、指定できるアンキャッシャブルエリアの数が数倍あることが望ましいが、ＤＭＡのチャネル数が多いと、このようなエリアコントロールのためのレジスタの数が増えて回路規模が膨大になってしまう。

【０００７】

別の方法として、予め大きめのアンキャッシャブルエリアを用意しておき、そのエリアを管理するメモリマネージャによってアンキャッシャブルエリアを割り当てる方法もある。その場合、アンキャッシャブルメモリのエリアを静的に決めておく必要があり、メモリ全体の利用効率を低下させる。

【０００８】

本発明は上記問題点に鑑みてなされたもので、キャッシュのエリア制御を容易にした情報処理装置及びその方法を提供する。

【課題を解決するための手段】

【０００９】

本発明の情報処理装置の特徴は、メモリの有する同一のメモリエリアを、複数の異なるアドレスセットでＣＰＵによりアクセス可能に構成したメモリ回路と、前記複数の異なるアドレスセットの一方のアドレスセットをキャッシャブル用のアドレスセットとし、他方をアンキャッシャブル用のメモリアドレスセットとしてキャッシュ制御するキャッシュ手段と、前記メモリに所定サイズのエリアが要求されると、当該要求された所定サイズにキャッシュ単位であるラインサイズの２倍のサイズを付加して、前記メモリの前記アンキャッシャブル用のメモリアドレスセットによりアクセス可能なメモリエリアに割当てる割当て手段とを有することにある。

【００１０】

また本発明の情報処理方法の特徴は、ＣＰＵ、メモリ、キャッシュメモリ及びキャッシュコントローラを具備する情報処理装置で実行される情報処理方法であって、

前記メモリに所定サイズのエリアが要求されると、当該要求された所定サイズにキャッシュ単位であるラインサイズの少なくとも２倍のサイズを付加して前記メモリに前記エリアを確保する工程と、前記エリアを前記メモリのアンキャッシャブル用のメモリエリアに割当てる工程とを有することを特徴とする。

【発明の効果】

【００１１】

本発明によれば、キャッシュのエリア制御を容易にできる。

【発明を実施するための最良の形態】

【００１２】

以下、添付図面を参照して本発明の好適な実施の形態を詳しく説明する。

【００１３】

図１は、本発明の実施の形態に係る情報処理装置におけるＣＰＵ周りのハードウェア構成を説明する図で、入出力部や表示部などの構成を省略して示している。

【００１４】

図１において、１０１はＣＰＵで、メインメモリ１０４に記憶されたＯＳやアプリケー

ションプログラムに従って各種動作を制御している。キャッシュコントローラ１０２は、キャッシュメモリ１０３を使用したキャッシュ制御を実行している。メインメモリ１０４は、ＣＰＵ１０１により実行されるプログラムやデータ等を記憶している。１０５はＤＭＡコントローラ（ＤＭＡＣ）で、ＣＰＵ１０１を介することなく、例えばハードディスクコントローラ（ＨＤＣ）１０６とメインメモリ１０４との間でのデータ転送を行うことができる。ハードディスク（ＨＤ）は大容量の外部メモリ装置で、ここにインストールされているＯＳや各種アプリケーションがメインメモリ１０４にロードされてＣＰＵ１０１により実行される。ここでキャッシュメモリ１０３は高速のＳＲＡＭを使用し、キャッシュコントローラ１０２の制御の下に大量のデータを蓄積し、古くて使用頻度の低いデータアドレスを破棄しながらキャッシュメモリの有効利用を図っている。

【００１５】

ここでキャッシュメモリ１０３とＣＰＵ１０１はワードやバイトといった単位でデータ転送を行うのに対して、キャッシュメモリ１０３とメインメモリ１０４との間でのデータ転送はライン単位で行う。即ち、１ビットのデータが必要な場合でも、ライン全体のデータがやり取りされる（ラインフィル）。このためにキャッシュメモリ１０３を備える装置では、キャッシュメモリ１０３とメインメモリ１０４間のデータ転送はラインサイズ（例えばバス幅が６４ビットの場合には２５６ビット）を最小単位として行われる。

【００１６】

＜シャドウメモリ＞

図２は、ＣＰＵ１０１とメインメモリ１０４との間のアドレス配線を説明する図である。

【００１７】

ＣＰＵ１０１から出ているアドレス信号の内、上位の数ビットを配線しないことにより、メモリ空間を小さく抑えることが可能である。図において、ＣＰＵ１０１は、アドレスビットＡ０～Ａ２２で定義されるメモリ空間、即ち、アドレス0番地～7xFFFFF番地までのメモリ空間を有しており、メインメモリ１０４は、２０ビットのアドレス（Ａ０～Ａ１９）、即ち、アドレス0番地～FFFFF番地までのメモリ空間を有している。ここでＣＰＵ１０１のメモリアドレスの内、上位アドレス（アドレスＡ２１，Ａ２２）のアドレス線をメモリ１０４のアドレスに接続しておらず、またアドレスＡ２０は、メインメモリ１０４のＣＥ（チップイネーブル）に接続されている。

【００１８】

図３は、図２の配線の場合のＣＰＵ１０１のメモリマップを説明する図である。

【００１９】

ＣＰＵ１０１からみたとき、メモリ１０４の0番地から0x0FFFFF番地までエリア３０１の内容と同じものが、例えばアドレス0x200000番地から0x2FFFFF番地までのメモリエリア３０２でもみえることになる。この場合、アドレス0x200000番地～0x2FFFFF番地は、0番地から0xFFFFF番地のシャドウという。

【００２０】

本実施の形態に係る情報処理装置では、このシャドウ側のメモリエリア３０２をアンキャッシャブルに設定して利用する。例えば、0番地から0xFFFFF番地までのエリア３０１をキャッシャブルとし、0x200000番地から0x2FFFFF番地までのエリア３０２をアンキャッシャブルにする。このアンキャッシャブルエリアの設定がキャッシュコントローラ１０２に対して与えられると、これ以降キャッシュコントローラ１０２は、そのメモリエリアへのアクセス時には、キャッシュ動作を実行しないように制御する。

【００２１】

＜汎用メモリマネージャの利用＞

本実施の形態に係る情報処理装置では、アンキャッシャブルメモリ領域３０２の割り当てに、汎用のメモリマネージャを利用する。この汎用のメモリマネージャとは、例えばC言語の標準関数であるmalloc関数として提供される。このmalloc関数によって確保したメモリエリアは、そのままではキャッシャブルエリア３０１に設定される。従って、確保し

たアドレスの内のデコードされていないビットを変更して、アンキャッシャブルメモリエリア３０２のアドレスとして、それをクライアントプログラム、即ち、malloc関数を実行したプログラムに返却する。例えば、malloc関数によって、キャッシャブルメモリ領域３０１のアドレス0x01000番地からアドレス0x014FF番地に確保されたメモリエリアを、ＣＰＵ１０１からみたアドレスビットＡ２１を「１」にすることにより、アンキャッシャブルメモリエリア３０２のアドレス0x201000番地からアドレス0x2014FF番地に確保する。

【００２２】

次に、このアンキャッシュブルメモリエリア３０２を解放する場合には、クライアントプログラムは逆に、メインメモリ１０４のアドレスの内、デコードされていないビット（上述の例ではアドレスビットＡ２１）を変更し、キャッシャブルエリア３０１のエリア（アドレス0x01000番地～アドレス0x01500番地）に変更する。そして次に、free関数を使って、メモリマネージャに、この確保しているエリアの解放を指示する。

【００２３】

このようにシャドウメモリを利用することによって、ハードウェアのエリア指定レジスタを使わずに無制限にキャッシャブルエリアを生成することが可能となる。

【００２４】

＜クリア＞

上記のようにしてアンキャッシャブルメモリとして割当てられたメモリエリア（上述の例では、アドレス0x01000番地～アドレス0x014FF番地）は、アンキャッシャブルエリアに割り当てられる直前までキャッシャブルモリとして利用されていた可能性がある。その場合、アンキャッシャブルメモリとして使用し始めてからキャッシュの吐き出しが起こると、それまでキャッシュされていた内容により、そのメモリエリアの内容が破壊されてしまう。従って、少なくとも、クライアントにアンキャッシャブルメモリとして割り当てる前に、そのメモリエリアに対してキャッシュされている情報をクリアする必要がある。

【００２５】

＜ラインサイズ＞

上記のように割り当てたアンキャッシャブルメモリは、そのメモリエリア周辺のアドレスがキャッシャブルメモリとして利用されている。一般にキャッシュは、ライン単位（キャッシングの単位で、例えば１ラインは２５６バイト）でＬＲＵ（Least Recent Used）のアルゴリズムで管理されているが、汎用のメモリマネージャは、キャッシュのラインサイズとは無関係に、メモリエリアの割り当てを行う。

【００２６】

図４を参照して、より詳しく説明する。

【００２７】

図４において、４００はクライアントプログラムに割り当てられたメモリエリアを示している。しかし、汎用のメモリマネージャを利用しているため、４０１で示すラインの一部がクライアント領域の外側と共有されている。クライアントプログラムへメモリエリアを引き渡したときに、４０２で示す部分は、まだクライアントプログラムからの書き込みが行われていない。その後、４０３で示す部分、即ち、クライアント領域とラインを共有している外側の領域を他のクライアントプログラムが書き換えると、クライアント使用領域４０１に対してダーティビットが立つ。そして次に、クライアントプログラムがクライアント使用領域４００全体のデータ４０４をＤＭＡ転送する。この時点ではメモリは破壊されていない。そしてシステムのプログラムの実行が進みＬＲＵアルゴリズムによって、４０１で示すラインがキャッシュから吐き出される時、４０５で示すように、ライン４０１全体が書き出される。これにより領域４０６が、ＤＭＡ転送前のデータで上書きされ、そのラインの内容が破壊されてしまう。

【００２８】

そこで本実施の形態では、クライアントプログラムの使用領域４１４の両端に、それぞれラインサイズ分の不使用領域を設ける。これにより、汎用のメモリマネージャを利用しても、メモリの内容が破壊されないように防止する。

【００２９】
これを説明したのが図４の４１０で示す部分で、４１１，４１２は、このようなデータ破壊を防ぐために両端に付加された不使用領域を示し、それぞれ１ライン分に相当するデータエリアである。
【００３０】
このように本実施の形態では、クライアントプログラムからアンキャッシャブル領域を要求されると、その要求された領域のサイズに２ライン分のサイズを上乗せしたサイズ（４１３で示す）で汎用メモリマネージャからメモリエリアを取得し、クライアントプログラムに、４１４で示す安全なエリア情報だけを使用領域として渡す。この場合のクライアントの使用可能なメモリエリアは４１４で示す領域となる。
【００３１】
図５は、本発明の実施の形態１に係る情報処理装置における処理を説明するフローチャートである。本実施の形態では、アンキャッシャブルなエリアをダイナミックに割り付けるメモリマネージメントの例で説明する。このプログラムは例えば、「cacheDmaMalloc( )」といった名前のＡＰＩで、ユーザプログラムから呼び出されてアンキャッシャブルなエリアを割り付ける。このプログラムは、ＨＤ１０７にインストールされているＯＳに含まれており、ＯＳのロードにメインメモリ１０４にロードされて実行される。このプログラムはまた既存のＯＳに対してインストール可能な実行コマンドとしても供給される。
【００３２】
まずステップＳ１で、ＡＰＩを通じてキャッシャブルエリアの確保を開始する。通常、パラメータとしてユーザが要求するメモリサイズを受け取る。次にステップＳ２で、ユーザプログラムが要求したメモリサイズに対して、キャッシュの２ライン分のサイズを付加したサイズを計算し、そのサイズで通常のメモリマネージャにメモリエリアを要求する。ここで、利用するメモリマネージャは、例えばＣ言語のインターフェースで、malloc( )関数に相当するものであり、特別なメモリマネージャではない。次にステップＳ３で、ステップＳ２での結果を検査し、メモリの確保に成功したかどうかをみる。失敗した場合はステップＳ４に進み、このサービスを終了する。この失敗する例としては、例えば、割り当てられるメモリエリアが無くなってしまった場合等が該当している。malloc( )関数の場合、メモリエリアの割り当てに失敗するとＮＵＬＬコードを返却するので、ステップＳ４でも同様に失敗であることを示す、例えばＮＵＬＬなどの値を返却する。
【００３３】
一方ステップＳ３で、メモリエリアの取得に成功するとステップＳ５に進み、その確保したメモリエリアに対してキャッシュコントローラ１０２を制御して、前述したフラッシュとクリアを行う。この時、確保したメモリエリアはメモリマネージャからみると未使用だった領域を割り当てているが、直前までユーザプログラムによって使用されていた場合もある。このため、使用中だった時に書き込んだ値がまだキャッシュ内部に残留していた場合には、後でキャッシュの吐き出しによって有効なデータを上書きして破壊する可能性があるため、キャッシュをクリアする。このクリアの前にフラッシュを行う理由は、確保したエリアとラインを共有している周辺エリアがキャッシュ内に未書き込みな有効なデータを持っている可能性があるからである。
【００３４】
こうしてステップＳ５で、メモリの準備が完了するとステップＳ６に進み、その確保したエリアの先頭番地に１ライン分を足したアドレスを計算し、そのシャドウエリアの番地を計算する。これは図４で説明したように、実際にクライアントプログラムが使用する領域は、先頭と最後の各１ラインデータを除いた領域であるためである。例えばラインサイズが前述したように２５６バイトで、図３に示すようなメモリマップの場合で説明する。ここで確保したメモリのアドレスを「0x00004010」とすると、「0x00004010」＋「0x100（ラインサイズ）」＋「0x00200000」→「0x00204110」がユーザに渡されるアドレスとなる。そしてステップＳ７で、ユーザプログラムに計算したメモリの番地を返却して終了する。

【００３５】
図６は、本実施の形態に係るクライアントプログラムで取得したアンキャッシャブルなメモリエリアをＯＳに返却する処理を示すフローチャートで、例えば「cacheDmaFree( )」といった名前のＡＰＩで提供される。このプログラムも同様に、ＨＤ１０７にインストールされているＯＳに含まれており、ＯＳのロードにメインメモリ１０４にロードされて実行される。このプログラムはまた既存のＯＳに対してインストール可能な実行コマンドとしても供給される。
【００３６】
まずステップＳ１１で、クライアントプログラムから呼び出される。このときパラメータとして、クライアントプログラムに割り当てられていたメモリエリアの先頭番地を受け取る。次にステップＳ１２で、指定された番地から１ライン分引いて、そのキャッシャブルエリア側の番地を計算する。例えば図５の例によれば、（0x00204110＆0xFFFDFFFFF）－0x100→0x00004010となる。ここで求めた値は、メモリマネージャからmalloc( )関数で取得したアドレスであるため、ステップＳ１３で、free( )関数によってメモリマネージャへ返却する。そしてステップＳ１４で、このサービスを終了する。
【００３７】
以上説明したように本実施の形態によれば、アンキャッシャブルエリアの設定を、レジスタなどの数に制限されずに行うことができる。
【００３８】
また本実施の形態によれば、汎用のメモリマネージャを利用して、メモリエリアの確保及びキャッシュ制御を行うことができる。
【００３９】
［その他の実施の形態］
前述の実施の形態１では、汎用のメモリマネージャを使用していたが、キャッシュのラインサイズを意識したメモリマネージャを用意し、ユーザエリアとラインを共有する領域ができないように工夫をすることにより、前後１ライン分の不使用領域を省く方法もある。
【００４０】
また前述の実施の形態１では、フルアドレスデコードしない回路構成で説明しているが、ＭＭＵ(Memory Management Unit)等のしくみを使って、同一メモリを他のアドレスでアクセスするシャドウを構成してもかまわない。
【００４１】
本発明の目的は前述したように、本実施の形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体をシステム或は装置に提供し、そのシステム或は装置のコンピュータ（又はＣＰＵやＭＰＵ）が記憶媒体に格納されたプログラムコードを読み出し実行することによっても達成される。この場合、記憶媒体から読み出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。このようなプログラムコードを供給するための記憶媒体としては、例えば、フロッピィディスク、ハードディスク、光ディスク、光磁気ディスク、ＣＤ－ＲＯＭ，ＣＤ－Ｒ、磁気テープ、不揮発性のメモリカード、ＲＯＭなどを用いることができる。
【００４２】
また、コンピュータが読み出したプログラムコードを実行することにより、前述した実施の形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼動しているＯＳ（オペレーティングシステム）などが実際の処理の一部又は全部を行い、その処理によって前述した実施の形態の機能が実現される場合も含まれている。
【００４３】
更に、記憶媒体から読み出されたプログラムコードが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書きこまれた

後、そのプログラムコードの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるＣＰＵなどが実際の処理の一部又は全部を行い、その処理によって前述した実施の形態の機能が実現される場合も含む。

【図面の簡単な説明】

【００４４】

【図１】本発明の実施の形態に係る情報処理装置のＣＰＵ周辺の構成を説明するブロック図である。

【図２】本実施の形態に係る情報処理装置におけるＣＰＵとメモリとの接続を説明する図である。

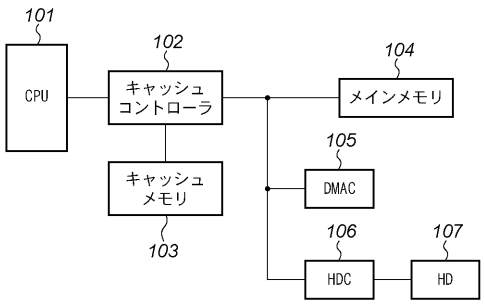
【図３】実施の形態に係るメインメモリのメモリマップを説明する図である。

【図４】従来のラインを共有することによるメモリにおけるデータの破壊と本実施の形態におけるメモリ使用領域の確保を説明する図である。

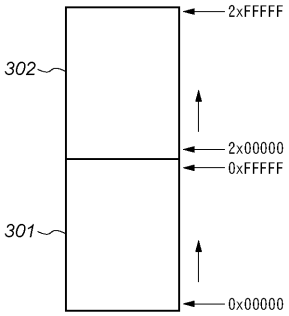
【図５】本発明の実施の形態に係る情報処理装置におけるアンキャシャブルエリアの確保処理を説明するフローチャートである。

【図６】本発明の実施の形態に係る情報処理装置におけるアンキャシャブルエリアの解放処理を説明するフローチャートである。

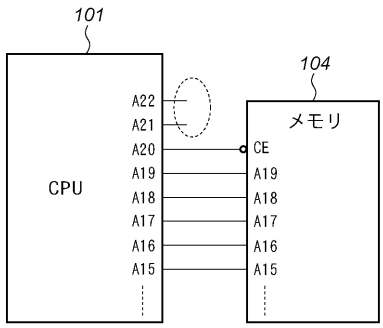
【図１】



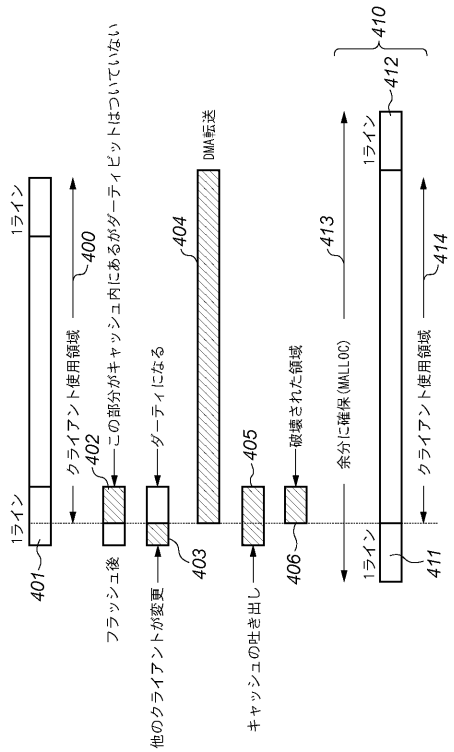
【図３】



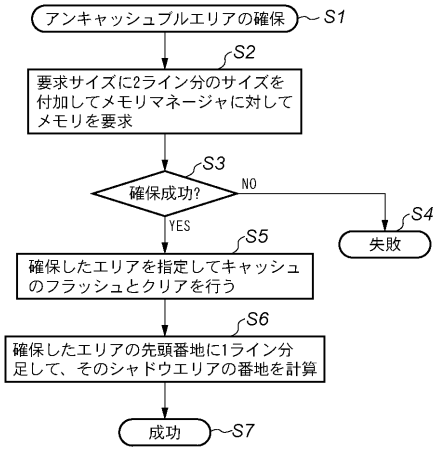
【図２】



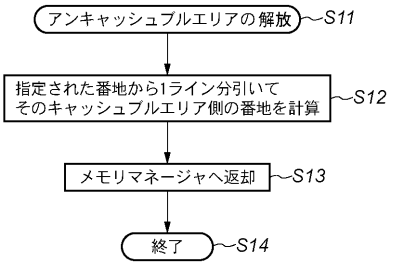
【図４】


1ライン
401
クライアント使用領域
400
この部分がキャッシュ内にあるがダーティビットはついていない
402
フラッシュ後
ダーティになる
403
他のクライアントが変更
DMA転送
404
キャッシュの吐き出し
405
破壊された領域
406
余分に確保(MALLOC)
413
410
412
1ライン
1ライン
411
クライアント使用領域
414

【図５】


アンキャッシュブルエリアの確保
S1
S2
要求サイズに2ライン分のサイズを
付加してメモリマネージャに対して
メモリを要求
S3
確保成功?
NO
YES
S4
失敗
S5
確保したエリアを指定してキャッシュ
のフラッシュとクリアを行う
S6
確保したエリアの先頭番地に1ライン分
足して、そのシャドウエリアの番地を計算
成功
S7

【図６】


アンキャッシュブルエリアの解放
S11
指定された番地から1ライン分引いて
そのキャッシュブルエリア側の番地を計算
S12
メモリマネージャへ返却
S13
終了
S14

フロントページの続き

(72)発明者　崎村　岳生
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

審査官　清木　泰

(56)参考文献　特開２００５－１９０１６１（ＪＰ，Ａ）
特開平１０－０７８９１６（ＪＰ，Ａ）
特開平０６－３４８５９３（ＪＰ，Ａ）
特開平０４－０５１３７３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ１２／０８－１２／１２