(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3570442号
(P3570442)
(45)発行日　平成16年9月29日(2004.9.29)　(24)登録日　平成16年7月2日(2004.7.2)

(51)Int.Cl.[7]　F I
G06F 9/46　G06F 9/46　360F
G06F 15/177　G06F 15/177　681B

請求項の数 8　(全 15 頁)

(21)出願番号　特願平7-104357
(22)出願日　平成7年4月27日(1995.4.27)
(65)公開番号　特開平7-311740
(43)公開日　平成7年11月28日(1995.11.28)
審査請求日　平成13年12月5日(2001.12.5)
(31)優先権主張番号　234-207
(32)優先日　平成6年4月28日(1994.4.28)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100081721
弁理士　岡田　次生
(72)発明者　デイル・シー・モーリス
アメリカ合衆国カリフォルニア州メンロパーク　ギルバート・アヴェニュー　442
(72)発明者　バーナード・エル・スタンフ
アメリカ合衆国マサチューセッツ州チェイムスフォード　アクトン・ロード　50

最終頁に続く

(54)【発明の名称】コンピュータ

(57)【特許請求の範囲】
【請求項１】
中央演算処理装置（ＣＰＵ）を有するディジタル・コンピュータであって、ＣＰＵが、ロード命令および記憶命令を含む、コンピュータ・プログラムに関連するコンピュータ命令を実行し、コンピュータ命令はプログラム順であり、ＣＰＵを制御するように動作する制御システムであって、
ＣＰＵによって実行される第１のロード命令が、他のプログラムまたは他のＣＰＵとのプロセス間通信に使用される共有メモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが、当該共有メモリ領域にアクセスし当該プログラムに関連するロード命令は順序付けられる必要があることを示す特定の値を有する場合、当該共有メモリ領域にアクセスし当該プログラム順において第１のロード命令に続く第２のロード命令によって要求される第２のロード動作が第１のロード命令によって要求される第１のロード動作の完了前に完了されないようにＣＰＵを制御すると同時に、当該プログラム順において第１のロード命令に続く他の命令は、第１のロード動作の完了前に実行できるようにＣＰＵを制御し、
ＣＰＵによって実行される第２の記憶命令が、他のプログラムまたは他のＣＰＵとのプロセス間通信に使用される共有メモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが、当該共有メモリ領域にアクセスし当該プログラムに関連する記憶命令は順序付けられる必要があることを示す特定の値を有する場合、第２の記憶命令によって要求される第２の記憶動作が、当該共有メモリ領域にアクセスし当該プログラム順において第２の記憶命令に先行する第１の記憶命令によって要求される第１の記憶動作が完了するまで完了され

ないようにＣＰＵを制御すると同時に、第２の記憶動作は当該プログラム順において第２の記憶命令に先行する他の命令の前に完了することができるようにＣＰＵを制御するように動作する制御システムを含むコンピュータ。
【請求項２】
制御システムが、参照されるメモリ領域に関連するメモリ属性ビットが特定の値を有することを検査することによって、共有メモリ領域への参照を検出する請求項１に記載のコンピュータ。
【請求項３】
メモリ属性ビットが、変換索引緩衝機構に記憶される請求項２に記載のコンピュータ。
【請求項４】
プロセス状況ビットが、ＣＰＵレジスタに記憶される請求項１に記載のコンピュータ。
【請求項５】
中央演算処理装置（ＣＰＵ）を有するコンピュータであって、ＣＰＵが、ロード命令および記憶命令を含む、コンピュータ・プログラムに関連するコンピュータ命令を実行し、コンピュータ命令はプログラム順であり、ＣＰＵを制御するように動作する制御システムであって、
ＣＰＵによって実行される第１のロード命令が、他のプログラムまたは他のＣＰＵとのプロセス間通信に使用される共有メモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが、当該共有メモリ領域にアクセスし当該プログラムに関連するロード命令は順序付けられる必要があることを示す特定の値を有する場合、当該共有メモリ領域にアクセスし当該プログラム順において第１のロード命令に続く第２のロード命令によって要求される第２のロード動作が第１のロード命令によって要求される第１のロード動作の完了前に完了されず、当該プログラム順において第１のロード命令に続き、当該共有メモリ領域を参照する記憶命令によって要求される記憶動作が第１のロード命令が完了するまで完了しないようにＣＰＵを制御すると同時に、当該プログラム順において第１のロード命令に続く他の命令は、第１のロード動作の完了前に実行できるようにＣＰＵを制御し、
ＣＰＵによって実行される第２の記憶命令が、他のプログラムまたは他のＣＰＵとのプロセス間通信に使用される共有メモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが、当該共有メモリ領域にアクセスし当該プログラムに関連する記憶命令は順序付けられる必要があることを示す特定の値を有する場合、第２の記憶命令によって要求される第２の記憶動作が、当該共有メモリ領域にアクセスし当該プログラム順において第２の記憶命令に先行する第１の記憶命令によって要求される第１の記憶動作が完了するまで完了されず、第２の記憶命令によって要求される第２の記憶動作が、当該共有メモリ領域を参照し当該プログラム順において第２の記憶動作に先行するロード命令によって要求される全てのロード動作が完了するまで完了しないようにＣＰＵを制御すると同時に、第２の記憶動作は当該プログラム順において第２の記憶命令に先行する他の命令の前に完了することができるようにＣＰＵを制御するように動作する制御システムを含むコンピュータ。
【請求項６】
制御システムが、参照されるメモリ領域に関連するメモリ属性ビットが特定の値を有することを検査することによって、共有メモリ領域への参照を検出する請求項５に記載のコンピュータ。
【請求項７】
メモリ属性ビットが、変換索引緩衝機構に記憶される請求項６に記載のコンピュータ。
【請求項８】
プロセス状況ビットが、ＣＰＵレジスタに記憶される請求項５に記載のコンピュータ。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】
本発明は、バリア命令を使用しない古いプログラムを必要に応じて順次付けて実行するとともに、中央演算処理装置（以下、ＣＰＵという）が命令を非順序に完了できるようにし

て、プログラムの実行効率の向上を期するようにしたコンピュータに関する。

【０００２】

【従来の技術】

大部分のデジタル・コンピュータは、何らかの結果を得るためにプログラム中の命令を実行する責任を負う単一のＣＰＵを組み込んでいる。このような単一ＣＰＵは一般に、非常に簡単なプロセッサ実行モデルを有する。すなわち、ＣＰＵは、ハードウェアが命令を一度に１つずつ、実行中の特定のプログラムで命令が現れる順序で実行するようにプログラマに見えるように命令を実行する。これは「プログラム順」実行と呼ばれ、この命令実行方法は今日、パーソナル・コンピュータなどで広く使用されている。

【０００３】

プログラム順に命令を実行する単一ＣＰＵコンピュータは、固有の命令スループット限界を有する。最大の限界のうちの２つは、ＣＰＵが半導体技法で許容される速度でしか動作できず、かつ特定の命令をただちに実行できないことが多いために発生する。第１の限界を解消するために、コンピュータは、各ＣＰＵがプログラムの一部を実行する複数の並列ＣＰＵを使用し始めた。第２の限界は、ＣＰＵがただちに完了できない命令を「中断」し、次の命令を実行して時間を節約できるようにすることによって、部分的に対処されている。この結果、ＣＰＵは命令をプログラム順に開始するが、それらの命令は順不同で完了することができる。ＣＰＵが命令を順不同で完了できるようにすることは利点であるが、この命令実行方法は、他の重大な問題を発生させる。

【０００４】

複数のＣＰＵを備えたコンピュータが効果的に動作するには、ＣＰＵは両方向に通信できなければならない。一般に、ＣＰＵは、相互にアクセスできる共有メモリを使用することによって通信する。共有メモリを通信に使用できる１つの例は「メールボックス」方法であり、この方法は本明細書では、本発明を例示するために使用される。複数ＣＰＵを備えたコンピュータでは、１つのＣＰＵが動作を実行し、動作の結果が他のＣＰＵによって使用されることが多い。ＣＰＵが通信する１つの方法は、「メールボックス」を「フラグ」と共に使用することによるものである。すなわち、第１のＣＰＵが、第２のＣＰＵが使用できるようにデータを置くメールボックスとしてメモリ中のある位置が指定される。第２のＣＰＵは、このＣＰＵが使用できる有効なデータがメールボックスにあることをこのＣＰＵに通知する特定のビット・パターン（フラグ）を探すために、指定されたメモリ位置を読み取る。本発明を理解するには、メールボックス・プロセスがどのように働くかを理解することが有用である。

【０００５】

図６および図７は、２つのＣＰＵ間のメールボックス通信技法を示す。図６は、プロセッサ１が実行するプログラム・コード・セクションを１０１で示しており、２つの命令が詳細に示されている。第１の命令１０３によって、プロセッサ１はメールボックス位置にデータを記憶する。第２の命令１０５によって、プロセッサ１はメールボックス・フラグをセットする。プロセッサ２が実行するプログラム・コード・セクションが１０７で示されており、４つの命令が詳細に示されている。第１の命令１０９によって、プロセッサ２はメール・フラグをロードし、第２の命令１１１によって、プロセッサ２は、フラグがセットされている（メールボックスに有効なデータがあることを示す）かどうかを調べるためにフラグを試験する。分岐命令１１３によって、プロセッサ２は、フラグがセットされていない場合（命令１１１によって判定される）、ループ・バックしてメール・フラグを再ロードする（命令１０９）。フラグがセットされている場合、プロセッサ２は分岐命令１１３を通過し続け、次の命令１１５を実行する。この命令によって、プロセッサ２はメールボックス中のデータをロードする。

【０００６】

図６に示した、プロセッサ１および２による命令の実行の一例を図７に示す。これらは「順次一貫」ＣＰＵであり、これは、これらのＣＰＵがすべての命令をプログラム順に実行し、後の動作は、前の動作がすべて完了するまで実行されないことを意味する。プロセッ

サ１は、図７の時間Ｔ４でメールボックスにデータを記憶し（命令１０３）、次いで、時間Ｔ５でメール・フラグを記憶する（命令１０５）。時間Ｔ０ないし時間Ｔ３によって示される期間中、プロセッサ１は、図示していない他の命令を実行する。この例では、プロセッサ２は、図７に示すように時間Ｔ０でメール・フラグをロードし（命令１０９）、時間Ｔ１で、フラグがセットされているかどうかを調べるための検査を行う（命令１１１）。分岐命令（１１３）は、時間Ｔ２で実行され、フラグがプロセッサ１によってセットされていないため、分岐命令によって、プロセッサ２は分岐してメール・フラグ命令１０９をロードし、時間Ｔ３でメール・フラグを再ロードする。フラグは時間Ｔ４で再び検査され、フラグがセットされていないため、分岐命令によって、プロセッサ２は再び分岐して、メール・フラグ命令をロードする。この命令は、時間Ｔ６で実行される。フラグは時間Ｔ７で再検査され、フラグがプロセッサ１によってセットされているため、時間Ｔ８で実行される分岐命令によってプロセッサ２が分岐することはなく、したがって、プロセッサ２は次に時間Ｔ９で、メールボックス中のデータをロードする（命令１１５）。

【０００７】

この例では、プロセッサ２は、プロセッサ１がメールボックスを記憶してフラグをセットするまでメールボックスからのロードを行わないので、妥当なメールボックス・データをロードする。ＣＰＵが順次一貫性を維持せず、すなわち、ＣＰＵが、前の命令がすべて実行される前に以後の命令を完了できる場合、メールボックス通信技法は失敗することがある。

【０００８】

メールボックス通信技法が失敗する簡単な例を図８に示す。この図８は、図６で詳細に説明した命令が、命令を必ずしも順次一貫した方法で実行するわけではないＣＰＵによって実行されることを示す。プロセッサ１は、２つの動作を実行する。第１の動作１０３は、時間Ｔ４中にメールボックスにデータを記憶することであり、第２の動作は、時間Ｔ５中にメール・フラグ１０５を記憶することである。プロセッサ２は、時間Ｔ０でメール・フラグをロードし（１０９）、次いで、時間Ｔ１中に、メール・フラグがセットされているかどうかを調べるための検査を行う（１１１）。しかし、メール・フラグは時間Ｔ５にならない限りプロセッサ１によって記憶されず、したがって、プロセッサ２によって実行されるフラグ試験は失敗し、時間Ｔ２で実行される分岐命令１１３によって、プロセッサ２はメール・フラグのロードにループ・バックする。プロセッサがメール・フラグをただちにロードできず、たとえば、メール・フラグがキャッシュではなくメイン・メモリにある場合、プロセッサは、メール・フラグのロードの実行を中断する。しかし、プロセッサ２は、次のメールボックス・データのロード（１１５）を実行できるので、これを実行する（ＣＰＵは、この情報を後で実行する必要はないと「仮定」し、したがって、このロードを実行して時間を節約する）。ＣＰＵは、時間Ｔ４でメール・フラグの再ロードを完了し、時間Ｔ５でフラグを再試験する。この第２回目の試みで、プロセッサ２はフラグがセットされていることを知り、したがって、時間Ｔ６で実行される分岐命令によってプロセッサ２が分岐することはない。プロセッサ２はすでに時間Ｔ３中にメールボックスからのデータのロードを完了しているので、ロードを再実行せず（仮定が正しかったと仮定する）、引き続き、プログラム中の次の命令１１７を実行する。しかし、プロセッサ２はメールボックス中の無効なデータを読み取ることになる。これは受け入れられない結果である。

【０００９】

ＣＰＵが、命令を非順次に実行し、同時に、ＣＰＵ間通信の失敗を防止できるようにするために、「バリア」命令と呼ばれる特殊な命令が使用されている。このような命令によって、ＣＰＵは、バリア命令よりも先に進む前に前の命令を終了しておくことを強制される。コンピュータ・プログラマは、このような命令を使用して、プログラム順に実行する必要があるプログラム・コードの領域がＣＰＵによってプログラム順に実行されるようにする。このように、ＣＰＵは、プログラムの比較的小さな部分の命令をプログラム順に実行することを強制されるに過ぎず、プログラム命令の残りの部分は最大効率で実行することができる。

【００１０】
図９および図１０は、バリア命令を使用して、図８に示した問題が発生するのを防ぐことを示す。図９は、各ＣＰＵが実行するプログラムの一部をリストしたもので、前記で論じたのと同じものであり、「メールボックス中のデータのロード命令」１１５の前に追加バリア命令４０１が置かれている。プロセッサ１は、時間Ｔ４および時間Ｔ５中にそれぞれ、動作１０３および１０５を実行する。プロセッサ２は、図３に関連して論じたのと同じ動作を時間Ｔ０ないし時間Ｔ２で実行する。しかし、分岐命令１１３の後で、メールボックス中のデータのロード命令１１５の前に、バリア命令４０１が追加されている。したがって、プロセッサ２がフラグを試験して、フラグがセットされていないことを知った後、バリア命令によって、プロセッサ２が「仮定を行い」メールボックス中のデータをロードすることが妨げられ、プロセッサ２は、次の命令に進む前にフラグのロード・試験動作を完了しておくことを強制される。メール・フラグの再ロードが遅延し、プロセッサがメールボックス・データをロードできる場合でも、バリア命令は、次の命令に進む前にすべての未処理命令を完了しておくことをプロセッサに強制する。メールボックスからのデータのロードは最終的に、メールボックス・データが有効なものである時間Ｔ１２で達成される。バリア命令を使用することによって、プロセッサ２がメールボックスから無効なデータをロードすることが防止された。

【００１１】
【発明が解決しようとする課題】
非順次実行に関する問題を「ロード」動作に関して例示したが、同じ問題が「記憶」動作にも存在する。バリア命令を使用することによって、非順次「記憶」動作に関する問題も防止される。しかし、プログラマは「バリア」命令を使用して新しいプログラムを書くことができるが、今日使用されている大多数のプログラム・コードは、「バリア」命令を使用しない古いコードである。１つのＣＰＵが（バリア命令を有する）新しいコードを実行し、他のＣＰＵが（順次実行を保証することができない）古いコードを実行している場合、メールボックス通信技法は失敗する。２つのＣＰＵは共に、通信命令を順次に実行しなければならない。

【００１２】
命令を非順次に実行できるＣＰＵは、すべての命令を順次に実行することを強制できることが多い。したがって、「古い」コードを実行するＣＰＵがプログラム全体を順次に実行する（命令を順次に開始して完了する）ことを強制された場合、ＣＰＵ間通信の成功が保証される。しかし、ＣＰＵ間通信は一般に、プログラムにおいてあまり実行されない機能なので、すべての命令を順次に実行することをＣＰＵに強制すると、あまり実行されないタスクを成功させるために高性能が犠牲になる。

【００１３】
業界で必要とされているものは、必要に応じて、特殊なバリア命令を使用せずに、ロード命令および記憶命令を選択的に順序付けて実行するコンピュータである。このように、古いプログラムを効果的に使用し、同時に、ＣＰＵが命令を非順次に完了できるようにすることによって得られる性能上の利点の大部分を得ることができる。

【００１４】
【課題を解決するための手段】
本発明によって構成されたコンピュータは、バリア命令を使用しない古いプログラムを、必要に応じて順序付けて実行する。その他の場合、このコンピュータは、コンピュータに最も効率的な方法でプログラムを実行する。このコンピュータは、アドレス可能なメモリ領域と、単一のメモリ・アドレスまたはメモリ・アドレスのグループに関連するメモリ属性ビットとを含む。メモリ属性ビットが「セット」されているときは（論理真）、そのビットに関連するメモリ領域が２つ以上のプログラムまたは複数のＣＰＵによって共有されることを示す。コンピュータ中の各ＣＰＵはプロセス状況ビットも含む。この状況ビットがセットされているときは、ＣＰＵが現在、バリア命令や、必要に応じてプログラム順実行を強制するためのその他の手段を使用していないことを示す。

【００１５】
ＣＰＵがメモリ・アドレスにアクセスすると、プロセス状況ビットおよびメモリ属性ビットは、それらが共に「セット」されているかどうかを検出するために検査される。両方のビットがセットされている場合、ＣＰＵは、メモリ属性ビットに関連するメモリ領域を参照するロード命令および記憶命令を特定の「順序」で実行する。すなわち、記憶命令は、前に開始された記憶命令がすべて完了するまで実行されない。これを「順序付け記憶」命令と呼ぶ。両方のビットがセットされている場合、「ロード」命令は、次のロード命令が完了する前にＣＰＵによって完了される。これを「順序付けロード」命令と呼ぶ。
【００１６】
この新しいアーキテクチャは、古いプログラムによって必要とされる同期を提供し、同時に、従来技術の順次実行コンピュータで得られる効率に勝る改良された効率を提供する。
【００１７】
【実施例】
コンピュータ業界は、マルチＣＰＵコンピュータと、性能上の理由から厳密なプログラム順とは異なる順序で命令を完了するＣＰＵを使用するようになっている。しかし、今日使用されている大部分のプログラムは、バリア命令などを使用しないので、このような「古い」プログラムが正しく動作できるようにするには何らかの適応を施さなければならない。古いプログラムが単一のＣＰＵによって実行され、そのプログラムが他のＣＰＵやプログラムと通信しない場合、ＣＰＵが順不同で命令を完了する場合でも問題は発生しないことに留意されたい。それは、ＣＰＵは、複数の未処理の命令を有することができ、かつ最初に開始された命令よりも前に、いくつかの後の命令を完了することができるが、実行中のプログラムがプログラム順に実行されているように見えるようにするからである。
【００１８】
しかし、このようにプログラム順に見えることは、特定のＣＰＵによって実行されているプログラムにしか当てはまらない。第１のＣＰＵによって実行されている命令のシーケンスを第２のＣＰＵまたはプログラムから見ると（２つのＣＰＵまたはプログラムは特定のメモリ領域－共有メモリ－内で相互作用する）、第２のＣＰＵまたはプログラムに見えるのは順不同実行シーケンスである。この順不同の実行は、上述の通信障害を発生させるものである。したがって、プログラム順実行が問題になるのは、古いプログラムが他のプログラムまたはＣＰＵと相互作用するときだけである。本発明は、メモリ属性ビットおよびプロセス状況ビットを使用して、ＣＰＵがいつ、通信に使用されるメモリ中の位置を参照する命令をプログラム順に（完全な命令を順次に）実行しなければならないかを判定する。
【００１９】
オペレーティング装置（以下、ＯＳという）とは、作業がホスト・コンピュータによってどのように行われるかを制御する責任を負うソフトウェア・プログラムである。ＣＰＵ時間、ディスク空間、およびメモリを、コンピュータ上で実行する様々なプログラムに割り振ることは、ＯＳによって実行される典型的なタスクである。ＯＳは、どのページが共有されているかに注目することによって、どのページが通信のために異なるＣＰＵ（またはプログラム）によって使用されているかを「知る」。ＯＳは、メモリ・ページが共有されていることを動的に知ると、そのメモリ・ページに関連するメモリ属性ビットをセットする。この属性ビットは、コンピュータのメモリ管理装置によって維持され、特定のページがコンピュータのメイン・メモリに存在し、かつ参照されるときに、命令実行手段としての変換索引緩衝機構（以下、ＴＬＢという）の一部となる。ＣＰＵは、メモリのページを参照するとき、そのページに関連するメモリ属性ビットを検査して、そのページが通信に使用されるかどうかを判定する。
【００２０】
ＣＰＵは、通信に使用されるメモリ・ページが参照されていることを知ることができるが、ＣＰＵによって実行されているプログラムが、ＣＰＵが自動的に、そのページを参照する命令をプログラム順に実行することを必要とするかどうかを知る必要がある。たとえば

、バリア命令を使用する「新しい」プログラムはすでに、正しい通信を保証するために必要な準備を行っている。これに対して、「古い」プログラムは、通信障害を防ぐためにＣＰＵが命令の必要な順序付けを行うことを必要とする。したがって、ＣＰＵによって実行されている特定のプログラムに関連する「プロセス状況ビット」がＣＰＵによって維持される。このプロセス状況ビットは、ＣＰＵのプロセッサ状況レジスタに維持される。プログラムがバリア命令などを使用するかどうかを判定するために使用される必要なプロセス情報は多数の方法で生成することができる。たとえば、プログラムをコンパイルまたはアセンブルする場合、コンパイラまたはアセンブラは、プログラムが「新しい」プログラムであることをコンピュータに通知する特殊なコードを生成することができる。プログラムがコンピュータにロードされたとき、ＯＳは、この特殊コードを参照し、それに応じてプロセス状況ビットをセットする。

【００２１】

したがって、プログラムがメモリ・ページを参照するとき、ＣＰＵは、メモリ属性ビットとプロセス状況ビットを共に検査して、ＣＰＵがその特定のページを参照するすべての命令をプログラム順に実行（開始して完了）しなければならないかどうかを判定する。図１、図２、図３は、それぞれ、単一のプログラムを実行する、２つのＣＰＵを含む、可能な組合せを示す。

【００２２】

図１は、２つのＣＰＵ５０１および５０３と、３つのメモリ・ページ５０５、５０７、５０９を示す。各ＣＰＵ５０１および５０３はそれぞれ、プロセス状況ビット５１１および５１３を有し、各メモリ・ページ５０５、５０７、５０９はそれぞれ、メモリ属性ビット５１５、５１７、５１９を有する。プロセス状況ビット５１１と５１３は共に、ＣＰＵ５０１と５０３が共に、正しいプログラム実行を保証するためにバリア命令などを使用する「新しい」プログラムを実行していることを意味する「０」である。メモリ・ページ５０５と５０９は共に、それぞれのメモリ属性ビットが、これらのメモリ・ページがＣＰＵ間で共有されておらず、したがって、これらのメモリ・ページを参照する命令を順序付ける必要がないことを示す「０」にセットされている。メモリ・ページ５０７に関連するメモリ属性ビットは、このメモリ・ページがＣＰＵ５０１，５０３間で共有されており、したがって、このページを参照する命令を順序付ける必要があることを示す「１」にセットされている。

【００２３】

ＣＰＵ５０１がメモリ・ページ５０５にアクセスし、あるいは、ＣＰＵ５０３がメモリ・ページ５０９にアクセスするとき、どちらのメモリ・ページも共有されていないので、これらのＣＰＵは命令を明示的に順序付ける必要がない。ＣＰＵ５０１は、メモリ・ページ５０７にアクセスした場合、メモリ属性ビット５１７がセットされている（「１」）ことを検出する。プロセス状況ビット５１１はセットされていない（「０」）ので、ＣＰＵ５０１が、メモリ・ページ５０７を参照する命令を明示的に順序付けることを強制されることはない。すなわち、ＣＰＵ５０１は、ＣＰＵ５０１が命令を明示的に順序付けることを必要としないプログラムを実行している。というのは、そのようなプログラムは、通信の問題を回避するように書かれているからである。同じことがこの例でのＣＰＵ５０３にも当てはまる。

【００２４】

図２は、図１と同じＣＰＵおよびメモリ・ページを示す。しかし、この例では、ＣＰＵ５０１が、必要に応じて命令が順次に実行されることを保証しない「古い」プログラムを実行していることを示す、ＣＰＵ５０１のプロセス状況ビット６０１がセットされている（「１」）。したがって、ＣＰＵ５０１は、メモリ・ページ５０７にアクセスするとき、そのメモリ・ページを参照する命令を明示的にプログラム順で実行させなければならない。ＣＰＵ５０１は、メモリ・ページ５０７にアクセスするとき、プロセス状況ビット６０１とメモリ属性ビット５１７が共に「１」であると判定し、したがって、メモリ・ページ５０７を参照するすべての命令を順序付けて実行する。ＣＰＵ５０１は、メモリ・ページ５

０５にアクセスするとき、メモリ属性ビット５１５が「０」であることによって示されるように、そのメモリ・ページが共有されていないので、命令を順序付けない。ＣＰＵ５０３は、図１に関連して説明したように動作する。

【００２５】

図３は、図１および図２と同じＣＰＵおよびメモリを示すが、ＣＰＵ５０３のプロセス状況ビット７０１がセットされている（「１」）。これは、ＣＰＵ５０３が「古い」プログラムを実行しており、共有メモリ・ページを参照する命令を順序付けなければならないことを示す。ＣＰＵ５０１またはＣＰＵ５０３がメモリ・ページ５０７にアクセスするとき、これらのＣＰＵは、共有メモリ・ページ５０７を参照する命令が、順序付けて実行されるようにしなければならない。

【００２６】

このアーキテクチャによって、「古い」プログラムを実行するコンピュータは、プログラム順以外の順序で命令を完了できるようにすることから得られる大部分の速度上の利点を達成することができる。また、本発明によって構成されたコンピュータは、通信障害や、ＣＰＵまたはプログラム間でのメモリ・ページの共有に関係するその他の類似の問題の恐れなしに、「古い」プログラムと「新しい」プログラムを混ぜて実行することができ、好都合である。

【００２７】

図４は、ＣＰＵが必要に応じて共有メモリ・ページを参照する命令を順序付けるようにする簡略化された制御装置を示す。プロセス状況ワード８０１、命令制御装置８０３、およびＣＰＵの内部のアンド・ゲート８０５が示されている。ＴＬＢ８０７、および複数のＣＰＵによって共有されるメモリ装置の一部となるメモリ・ページ８０９も示されている。プロセス状況ワード８０１では、関連するＣＰＵが、共有メモリ・ページにアクセスする際に明示的な命令順序付けを必要とする「古い」プログラムを実行していることを示すプロセス状況ビット８１１がセットされている（「１」）。命令制御装置８０３では、２つの記憶命令８１３および８１５が待機している。

【００２８】

記憶命令８１３は、２つの命令のうちの第１の命令であり、したがって、命令制御装置８０３によって最初に開始される。記憶命令８１３は、レジスタ「１」（図示せず）の内容を仮想メモリ位置「Ａ」に記憶する（ＳＴ）よう命令制御装置８０３に命令する。ＣＰＵハードウェアは、ＴＬＢ８０７中の仮想アドレス「Ａ」を検索する。ＴＬＢは、「実」アドレス「Ａ１」に関連する項目「Ａ」８１７を有する。ＣＰＵハードウェアは次いで、メモリ・ページ８０９に位置する実アドレス「Ａ１」にレジスタ「１」の内容を記憶する。ＴＬＢ変換・記憶動作を実行するためのハードウェアおよび技法は、マイクロプロセッサ業界の当業者によって周知である。

【００２９】

ＴＬＢ「Ａ」項目にはメモリ属性ビット８１９も関連付けられている。このビットは、メモリ・アドレスＡ１が共有メモリ・ページ８０９内にあることを示す（「１」）にセットされている。プロセス状況ビット８１１と、メモリ属性ビット８１９が共にセットされている場合（「１」）、アンド・ゲート８０５は命令制御装置８０３に制御信号８２１を送る。この制御信号は、記憶命令８１３が完了するまで命令制御装置が記憶命令８１５を実行するのを妨げる。しかし、命令制御装置は、記憶命令８１３が完了するまですべての命令の実行を停止させておく必要はない。記憶命令８１５が共有メモリ・ページ８０９外のメモリ位置を参照する場合、記憶命令８１３が完了する前に記憶命令８１５を開始して完了することができる。これは、記憶命令８１５に関連する記憶動作が共有メモリ・ページ８０９では行われないため、第２のＣＰＵまたはプログラムが、この記憶動作の結果に依存しないからである。しかし、図のように、記憶命令８１５は共有メモリ・ページ８０９内の位置を参照しており、したがって、命令制御装置８０３は、記憶命令８１３が完了するまで、記憶命令８１５に関連する記憶動作を実行しない。

【００３０】

必要なハードウェアを実施する多数の方法があるが、所望の結果は、共有メモリ・ページへの第２の記憶命令８１５が、共有メモリ・ページへの前の記憶命令が完了するまで行われないことである。それによって、データがメールボックス位置に置かれる前にメール・フラグがセットされることが防止される。

【００３１】

図５は、図４で説明したＣＰＵおよびコンピュータ・ハードウェアを示すが、記憶命令が実行される代わりに、２つのロード命令９０１および９０３が命令制御装置８０３で待機している。この動作は、図４に関連して説明したのと同様であり、所望の結果は、ロード命令９０１がロード命令９０３よりも前に完了することである。

【００３２】

プロセス状況ビット８１１またはメモリ属性ビット８１９がセットされていない場合（「０」）、命令制御装置は引き続き、記憶命令８１３またはロード命令９０１が完了する前にそれぞれ、記憶命令８１３またはロード命令９０１に続くロード命令９０３を実行することができる。同期が必要とされず、あるいは、同期が、バリア命令を使用することなど何らかの他の手段によって提供されると仮定されている。この新しいコンピュータ・アーキテクチャを、限られた１つの使用法、すなわち、メールボックス通信で示したが、このアーキテクチャによって、以前は実施できなかったハードウェア設計手法が可能になる。また、このアーキテクチャによって、ユーザは、引き続き「古い」プログラムを使用し、同時に、厳密にプログラム順に命令を処理しない新しいＣＰＵによって与えられる大部分の利点を得ることができる。もちろん、説明した実施例はメモリ属性ビットおよびプロセス状況ビット使用して、ＣＰＵがどのモード（順次または順序付け）を使用して命令を実行するかを選択するが、他のモードも有利である。たとえば、あるモードは、「古い」命令を実行するＣＰＵによる共有メモリ領域への「順序付けロード」が、次に実行されるロード命令、または共有メモリ領域を参照する記憶命令よりも前に実行されるように、命令の実行を「順序付ける」ことができる。同様に、あるモードは、前の記憶命令がすべて完了し、かつ共有メモリ領域を参照する前のロード命令がすべて完了しない限り、「順序付け記憶」が実行されないように、命令を「順序付ける」ことができる。他の多数の有利な動作モードも可能である。

【００３３】

本発明の他の実施例は、本明細書で開示した本発明の仕様または実施態様を検討することによって当業者に明らかになろう。仕様および例は実施例としてのみ検討されるものであり、本発明の真の範囲および趣旨は特許請求の範囲によって示される。

【００３４】

以上、本発明の各実施例について詳述したが、ここで各実施例の理解を容易にするために、各実施例ごとに要約して以下に列挙する。

【００３５】

１．　中央演算処理装置（ＣＰＵ）を有するディジタル・コンピュータであって、ＣＰＵが、ロード命令および記憶命令を含むコンピュータ命令を実行する命令実行手段を有し、命令実行手段が、命令実行手段によって開始された命令の順序以外の順序で命令を完了するように動作し、あるいは、命令実行手段によって開始された命令の順序で命令を完了するように動作し、

ＣＰＵによって実行される第１のロード命令が共有メモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが特定の値を有する場合、第１のロード命令に続く第２のロード命令が、命令実行手段によって第１のロード命令が完了されない限り実行されないように、制御手段が命令実行手段を制御し、

ＣＰＵによって実行される記憶命令が共有メモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが特定の値を有する場合、その記憶命令が、命令実行手段によって開始された前の記憶命令がすべて完了しない限り実行されないように、制御手段が実行命令手段を制御するように、

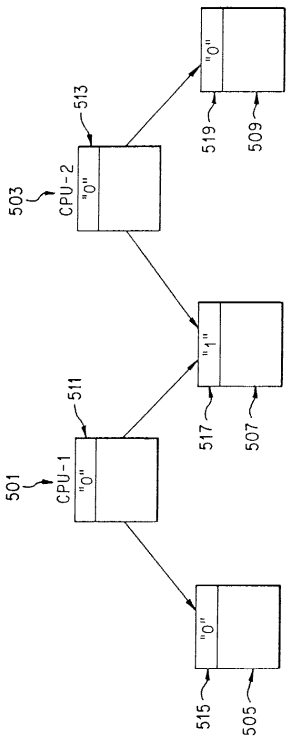
ＣＰＵを制御するように動作する制御手段を備えることを特徴とするコンピュータである

。
【００３６】
２．　制御手段が、参照されるメモリ領域に関連するメモリ属性ビットが特定の値を有することを検査することによって、共有メモリ領域への参照を検出することを特徴とする上記１に記載のコンピュータである。
【００３７】
３．　メモリ属性ビットが、変換索引緩衝機構に記憶されることを特徴とする上記２に記載のコンピュータである。
【００３８】
４．　プロセス状況ビットが、ＣＰＵレジスタに記憶されることを特徴とする上記１に記載のコンピュータである。
【００３９】
５．　中央演算処理装置（ＣＰＵ）を有するコンピュータであって、ＣＰＵが、ロード命令および記憶命令を含むコンピュータ命令を実行する命令実行手段を有し、命令実行手段が、命令実行手段によって開始された命令の順序以外の順序で命令を完了するように動作し、あるいは、命令実行手段によって開始された命令の順序で命令を完了するように動作し、
ＣＰＵによって実行される第１のロード命令が共有メモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが特定の値を有する場合、第１のロード命令に続く第２のロード命令が、命令実行手段によって第１のロード命令が完了されない限り実行されず、かつ共有メモリを参照する記憶メモリは第１のロード命令が完了しない限り実行されないように、制御手段が命令実行手段を制御し、
ＣＰＵによって実行される記憶命令が共有メモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが特定の値を有する場合、その記憶命令が、命令実行手段によって開始された前の記憶命令がすべて完了し、かつ共有メモリ領域を参照する前のロード命令がすべて完了しない限り実行されないように、制御手段が実行命令手段を制御するように、
ＣＰＵを制御するように動作する制御手段を備えることを特徴とするコンピュータである。
【００４０】
６．　制御手段が、参照されるメモリ領域に関連するメモリ属性ビットが特定の値を有することを検査することによって、共有メモリ領域への参照を検出することを特徴とする上記５に記載のコンピュータである。
【００４１】
７．　メモリ属性ビットが、変換索引緩衝機構に記憶されることを特徴とする上記６に記載のコンピュータである。
【００４２】
８．　プロセス状況ビットが、ＣＰＵレジスタに記憶されることを特徴とする上記５に記載のコンピュータである。
【００４３】
９．　中央演算処理装置（ＣＰＵ）を有するコンピュータであって、ＣＰＵが複数の命令実行モードを有し、
ＣＰＵが、それに関連するプロセス状況ビットと、そのＣＰＵによって実行される命令によって参照されるメモリ領域に関連するメモリ属性ビットとに応答して、特定の命令実行モードで動作するように、ＣＰＵを制御するように動作する制御手段を備えることを特徴とするコンピュータである。
【００４４】
１０．　メモリ属性ビットが、変換索引緩衝機構に記憶されることを特徴とする上記９に記載のコンピュータである。
【００４５】
１１．　プロセス状況ビットが、ＣＰＵレジスタに記憶されることを特徴とする上記９に

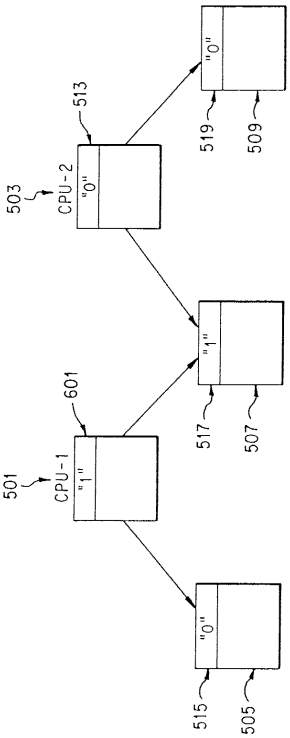
記載のコンピュータである。
【００４６】
１２．　メモリ領域が共有メモリ領域であることを特徴とする上記９に記載のコンピュータである。
【００４７】
１３．　複数の命令実行モードのうちの１つが、命令がプログラム順に実行される順次命令モードであることを特徴とする上記９に記載のコンピュータである。
【００４８】
１４．　複数の命令実行モードのうちの１つが、
ＣＰＵによって実行される第１のロード命令がメモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが特定の値を有する場合、第１のロード命令に続く第２のロード命令が、第１のロード命令がＣＰＵによって完了されない限り実行されないように、ＣＰＵが命令を制御し、
ＣＰＵによって実行される記憶命令がメモリ領域を参照し、ＣＰＵに関連するプロセス状況ビットが特定の値を有する場合、その記憶命令が、ＣＰＵによって開始された前の記憶命令がすべて完了しない限り実行されないように、ＣＰＵが命令を制御するように、
動作することを特徴とする９項に記載のコンピュータである。
【００４９】
【発明の効果】
以上のように、本発明によれば、ＣＰＵに関連するプロセス状況ビットが特定の値を有する場合に第１のロード命令に続く第２のロード命令がＣＰＵの有する命令実行手段によって第１のロード命令が完了しない限り実行されないように制御手段により命令実行手段を制御するとともに、ＣＰＵに関連するプロセス状況ビットが特定の値を有する場合に記憶命令が命令実行手段によって開始された前の記憶命令がすべて完了されないように制御手段が命令実行手段を制御するようにしたので、古いプログラムを効果的に使用し、同時にＣＰＵが命令の非順次に完了することができ、コンピュータ装置の性能を向上することができる。
【図面の簡単な説明】
【図１】共有メモリ・ページおよび専用メモリ・ページと相互作用する２つのＣＰＵを示すブロック図である。
【図２】１つのＣＰＵが、明示的なＣＰＵ命令順序付けを必要とするプログラムを実行する、共有メモリ・ページおよび専用メモリ・ページと相互作用する２つのＣＰＵを示すブロック図である。
【図３】共に、明示的なＣＰＵ命令順序付けを必要とするプログラムを実行する、２つのＣＰＵを示すブロック図である。
【図４】コンピュータ・ハードウェアが２つの記憶命令を実行する本発明の概略図である。
【図５】コンピュータ・ハードウェアが２つのロード命令を実行する本発明の概略図である。
【図６】メールボックス通信を使用する２つの独立のＣＰＵに関する部分的な命令の流れを示すフローチャートである。
【図７】図６に示したコマンドの実行を示す説明図である。
【図８】非順次一貫ＣＰＵによる、図６に示したコマンドの実行を示す説明図である。
【図９】メールボックス通信を使用し、かつバリア命令を組み込んだ、２つのＣＰＵに関する部分的な命令の流れを示すフローチャート図である。
【図１０】メールボックス・ロード動作の前にバリア命令を使用する従来技術の使用を示すフローチャートである。
【符号の説明】
５０１，５０３　　ＣＰＵ
５０５，５０７，５０９，８０９　　メモリ・ページ

５１１，５１３　　プロセス状況ビット
５１５，５１７，５１９，８１９　　メモリ属性ビット
６０１，７０１，８１１　　プロセス状況ビット
８０１　　プロセス状況ワード
８０３　　命令制御装置
８０５　　アンド・ゲート
８０７　　変換索引緩衝機構（ＴＬＢ）
８１３，８１５　　記憶命令
８１７　　項目「Ａ」
８２１　　制御信号
９０１，９０３　　ロード命令

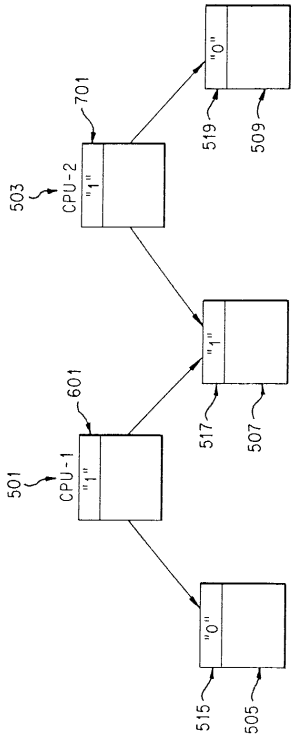
【図１】



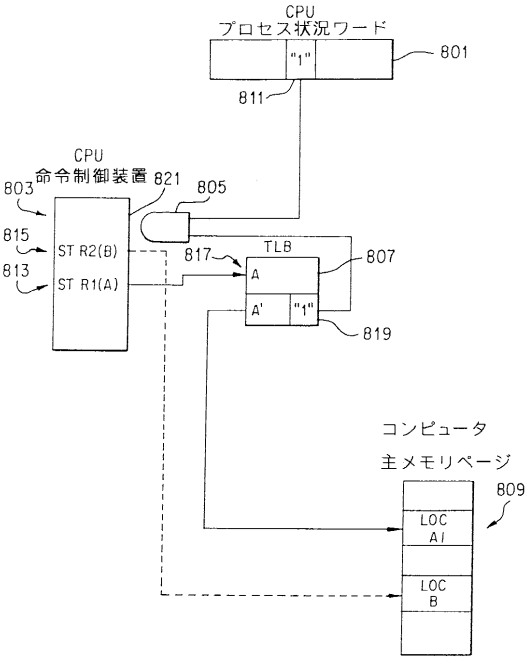
【図２】



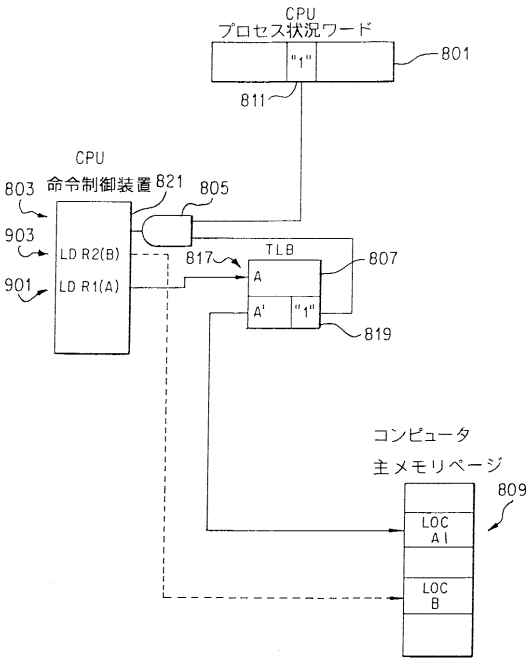
【図３】


"0"
519
509
701
503
CPU-2
"1"
"1"
517
507
601
501
CPU-1
"1"
"0"
515
505

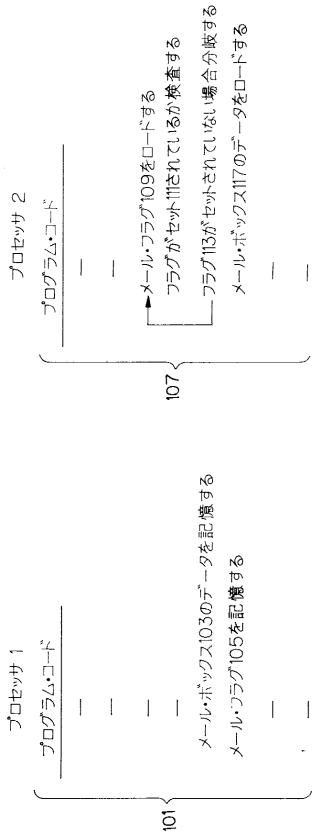
【図４】


CPU
プロセス状況ワード
"1"
801
811
CPU
命令制御装置
821
805
803
815
ST R2(B)
813
ST R1(A)
TLB
817
807
A
A'
"1"
819
コンピュータ
主メモリページ
809
LOC
A
LOC
B

【図５】


CPU
プロセス状況ワード
"1"
801
811
CPU
命令制御装置
821
805
803
903
LD R2(B)
901
LD R1(A)
TLB
817
807
A
A'
"1"
819
コンピュータ
主メモリページ
809
LOC
A
LOC
B

【図６】


プロセッサ 2
プログラム・コード
メール・フラグ109をロードする
フラグがセット111されているか検査する
フラグ113がセットされていない場合分岐する
メール・ボックス117のデータをロードする
107
プロセッサ 1
プログラム・コード
メール・ボックス103のデータを記憶する
メール・フラグ105を記憶する
101

【図７】

プログラム実行(順次一貫)

時間	命令プロセッサ 4	命令プロセッサ 2
T_0	—	メール・フラグをロードする
T_1	—	フラグがセットされているか検査する
T_2	—	(メール・フラグ・ロードに対して)分岐する
T_3	—	メール・フラグをロードする
T_4	メール・ボックスのデータを記憶する	フラグがセットされているか検査する
T_5	メール・フラグを記憶する	(メール・フラグ・ロードに対して)分岐する
T_6	—	メール・フラグをロードする
T_7	—	フラグがセットされているか検査する
T_8	—	(実行しない)分岐する
T_9	—	メール・ボックスのデータをロードする

【図８】

プログラム実行

時間	プロセッサ 1	プロセッサ 2
T_0	—	メール・フラグをロードする
T_1	—	フラグがセットされているか検査する
T_2	—	(メール・フラグ・ロードに対して)分岐する
T_3	—	メール・フラグのデータをロードする
T_4	メール・ボックスのデータを記憶する	メール・フラグをロードする
T_5	メール・フラグを記憶する	フラグがセットされている場合検査する
T_6	—	分岐する
T_7	—	命令 117

【図９】

プロセッサ 1	プロセッサ 2
プログラム・コード	プログラム・コード
—	—
—	—
—	—
メール・ボックス103のデータを記憶する	メール・フラグ109をロードする
メール・フラグ105を記憶する	フラグがセット111されているか検査する
	フラグ113がセットされていない場合分岐する
	バリア命令 401
	メール・ボックス115のデータをロードする
	—
	—
	—

【図１０】

プログラム実行

時間	プロセッサ 1	プロセッサ 2
T_0	—	メール・フラグをロードする
T_1	—	フラグがセットされている場合検査する
T_2	—	(フラグがセットされていない場合)検査する
T_3	—	バリア命令
T_4	メール・ボックスのデータを記憶する	メール・フラグをロードする
T_5	メール・フラグを記憶する	フラグがセットされている場合検査する
T_6	—	(フラグがセットされていない場合)分岐する
T_7	—	バリア命令
T_8	—	メール・フラグをロードする
T_9	—	メール・フラグがセットされている場合検査する
T_{10}	—	(フラグがセットされていない場合)分岐する
T_{11}	—	バリア命令
T_{12}	—	メール・ボックスのデータをロードする
T_{13}	—	命令 117

フロントページの続き

(72)発明者　バリー・ジェイ・フラヒヴ
アメリカ合衆国マサチューセッツ州ウエストフォード　パーカー・サークル　２
(72)発明者　ジェフリー・ディー・カーツェ
アメリカ合衆国ニューハンプシャー州ナシュア　スカイライン・ドライヴ　４
(72)発明者　ステファン・ジー・バーガー
アメリカ合衆国カリフォルニア州サンタクララ　フォーブス・アヴェニュー　２２５７
(72)発明者　ルビー・ビー・エル・リー
アメリカ合衆国カリフォルニア州ロスアルトス　アサートン・コート　１２９３３
(72)発明者　ウィリアム・アール・ブリグ
アメリカ合衆国カリフォルニア州サラトガ　ペレゴ・ウェイ　１８６３０

審査官　殿川　雅也

(56)参考文献　特開平　３－　１４０２５（ＪＰ，Ａ）
特開平　６－１１０８４５（ＪＰ，Ａ）
特開平　７－１２９５２６（ＪＰ，Ａ）
特開平　７－３０２２００（ＪＰ，Ａ）
米国特許第５０７５８４６（ＵＳ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F　9/46　-　9/54
G06F 15/16　- 15/177