(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4615810号
(P4615810)
(45)発行日　平成23年1月19日(2011.1.19)　(24)登録日　平成22年10月29日(2010.10.29)

(51)Int.Cl.　F I
G06F 9/34 (2006.01)　G06F 9/36 330B

請求項の数 23 (全 25 頁)

(21)出願番号	特願2001-552212 (P2001-552212)
(86)(22)出願日	平成12年7月19日 (2000.7.19)
(65)公表番号	特表2003-519869 (P2003-519869A)
(43)公表日	平成15年6月24日 (2003.6.24)
(86)国際出願番号	PCT/US2000/019770
(87)国際公開番号	WO2001/052059
(87)国際公開日	平成13年7月19日 (2001.7.19)
審査請求日	平成19年4月24日 (2007.4.24)
(31)優先権主張番号	09/483,078
(32)優先日	平成12年1月14日 (2000.1.14)
(33)優先権主張国	米国(US)

(73)特許権者　591016172
アドバンスト・マイクロ・ディバイシズ・インコーポレイテッド
ＡＤＶＡＮＣＥＤ　ＭＩＣＲＯ　ＤＥＶＩＣＥＳ　ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国、９４０８８－３４５３　カリフォルニア州、サニィベイル、ピィ・オゥ・ボックス・３４５３、ワン・エイ・エム・ディ・プレイス、メイル・ストップ・６８（番地なし）
(74)代理人　100064746
弁理士　深見　久郎
(74)代理人　100085132
弁理士　森田　俊雄

最終頁に続く

(54)【発明の名称】６４ビットアドレス指定のための呼出ゲート拡張

(57)【特許請求の範囲】
【請求項１】
プロセッサであって、
セグメントセレクタを特定する分岐命令を実行するように構成される実行コアを含み、
前記プロセッサは、前記セグメントセレクタに応答してセグメント記述子テーブルから少なくとも第１のエントリを読出すように構成され、前記第１のエントリが呼出ゲート記述子を指示するならば、前記プロセッサは、前記第１のエントリおよび前記セグメント記述子テーブル内の第２のエントリから、前記呼出ゲート記述子を読み出すように構成されており、前記第１のエントリと前記第２のエントリとの各々は別個にセグメント記述子を記憶することが可能である、プロセッサ。
【請求項２】
前記プロセッサはさらに前記呼出ゲート記述子からオフセットを抽出するように構成され、前記オフセットの少なくとも第１の部分は前記第１のエントリ内に記憶され、前記オフセットの残りの部分は前記第２のエントリ内に記憶される、請求項１に記載のプロセッサ。
【請求項３】
前記オフセットは６４ビットである、請求項２に記載のプロセッサ。
【請求項４】
前記プロセッサは前記呼出ゲート記述子からターゲットセグメントセレクタを抽出するように構成され、前記ターゲットセグメントセレクタはターゲットセグメント記述子を識

別する、請求項１に記載のプロセッサ。
【請求項５】
前記ターゲットセグメント記述子は第１の動作モード指示を含み、前記第１の動作モード指示は３２ビットよりも大きなデフォルトアドレスサイズを確立する、請求項４に記載のプロセッサ。
【請求項６】
前記第２のエントリはタイプフィールドを含み、前記タイプフィールドは符号化されて前記第２のエントリが無効であることが示される、請求項１に記載のプロセッサ。
【請求項７】
前記プロセッサは、前記第２のエントリをセグメント記述子として読出すことに応答して、前記タイプフィールドを用いて前記第２のエントリが無効であることを検出するように構成される、請求項６に記載のプロセッサ。
【請求項８】
方法であって、
プロセッサがセグメント記述子テーブルから呼出ゲート記述子を読出すステップを含み、前記呼出ゲート記述子は前記セグメント記述子テーブル内に第１のエントリと第２のエントリとを含み、前記第１のエントリと前記第２のエントリとの各々は、別個に、セグメント記述子を記憶することが可能であり、前記方法はさらに、
前記プロセッサが前記呼出ゲート記述子からオフセットを抽出するステップを含み、前記オフセットは、実行されるべき第１の命令をターゲットコードセグメント内に位置付ける、方法。
【請求項９】
前記オフセットは６４ビットである、請求項８に記載の方法。
【請求項１０】
前記プロセッサが前記セグメント記述子テーブルの第３のエントリのみから第１のセグメント記述子を読出すステップをさらに含む、請求項８に記載の方法。
【請求項１１】
前記プロセッサが前記呼出ゲート記述子からターゲットセグメントセレクタを抽出するステップをさらに含み、前記ターゲットセグメントセレクタはターゲットセグメント記述子を識別する、請求項８に記載の方法。
【請求項１２】
前記プロセッサが前記ターゲットセグメント記述子を読出すステップをさらに含み、前記ターゲットセグメント記述子は３２ビットよりも大きなデフォルトアドレスサイズを確立する、請求項１１に記載の方法。
【請求項１３】
前記第１のエントリと前記第２のエントリとの各々はタイプフィールドを有する、請求項８に記載の方法。
【請求項１４】
前記プロセッサが前記第２のエントリ内の前記タイプフィールドをセットして無効を示すステップをさらに含む、請求項１３に記載の方法。
【請求項１５】
前記プロセッサが前記第１のエントリ内の前記タイプフィールドをセットして呼出ゲート記述子を指示するステップをさらに含む、請求項１４に記載の方法。
【請求項１６】
前記プロセッサが前記第２のエントリからセグメント記述子を読出すことを試みるステップと、
前記プロセッサが前記第２のエントリ内の前記タイプフィールドに応答して前記セグメント記述子が無効であることを判断するステップとをさらに含む、請求項１４に記載の方法。
【請求項１７】

コンピュータシステムであって、
プロセッサを含み、前記プロセッサは、
セグメントセレクタを特定する分岐命令を実行するように構成される実行コアを含み、
前記プロセッサは前記セグメントセレクタに応答してセグメント記述子テーブルから少なくとも第１のエントリを読出すように構成され、前記第１のエントリが呼出ゲート記述子を指示するならば、前記プロセッサは、前記第１のエントリおよび前記セグメント記述子テーブル内の第２のエントリから、前記呼出ゲート記述子を読み出すように構成されており、前記第１のエントリと前記第２のエントリとの各々は別個にセグメント記述子を記憶することが可能であり、前記コンピュータシステムはさらに、
入力／出力（Ｉ／Ｏ）デバイスを含み、前記入力／出力（Ｉ／Ｏ）デバイスは、前記コンピュータシステムと、前記Ｉ／Ｏデバイスが結合可能な別のコンピュータシステムとの間で通信が行なわれるように構成される、コンピュータシステム。
【請求項１８】
前記Ｉ／Ｏデバイスはモデムを含む、請求項１７に記載のコンピュータシステム。
【請求項１９】
第２のプロセッサをさらに含み、前記第２のプロセッサは、
セグメントセレクタを特定する分岐命令を実行するように構成される実行コアを含み、
前記プロセッサは、前記セグメントセレクタに応答してセグメント記述子テーブルから少なくとも第１のエントリを読出すように構成され、前記第１のエントリが呼出ゲート記述子を指示するならば、前記セグメント記述子テーブル内の第２のエントリが前記呼出ゲート記述子の残りの部分を記憶する、請求項１７に記載のコンピュータシステム。
【請求項２０】
セグメントセレクタに応答してセグメント記述子テーブルからセグメント記述子を読出すように構成される回路を含むプロセッサであって、
前記セグメント記述子は、前記セグメント記述子のタイプを識別するタイプフィールドを含み、前記セグメント記述子のタイプに応答して、前記回路は、（ｉ）前記セグメント記述子テーブルの第１のエントリおよび第２のエントリ、または、（ｉｉ）前記セグメント記述子テーブルの第１のエントリのみ、のいずれかから前記セグメント記述子を読出すように構成される、プロセッサ。
【請求項２１】
前記タイプが呼出ゲート記述子を指示するならば、前記回路は前記第１のエントリおよび前記第２のエントリを読出すように構成され、前記タイプが前記呼出ゲート記述子ではない第１のセグメント記述子を指示するならば、前記回路は前記第１のエントリのみを読出すように構成される、請求項２０に記載のプロセッサ。
【請求項２２】
前記回路は前記セグメント記述子からオフセットを抽出するように構成され、前記セグメント記述子の前記タイプが、前記セグメント記述子が前記第１のエントリおよび前記第２のエントリ内に記憶されることを指示するならば、前記オフセットの第１の部分は前記第１のエントリにあり、前記オフセットの第２の部分は前記第２のエントリにある、請求項２０に記載のプロセッサ。
【請求項２３】
前記タイプフィールドが前記第１のエントリにあり、前記第２のエントリにあるタイプフィールドと同じ位置が無効タイプにコーディングされる、請求項２０に記載のプロセッサ。
【発明の詳細な説明】
【０００１】
【発明の分野】
この発明はプロセッサの分野に関し、より具体的には、プロセッサ内のアドレスサイズおよびオペランドサイズに関する。
【０００２】

【関連技術の説明】
ｘ８６アーキテクチャ（ＩＡ－３２アーキテクチャとしても知られる）は市場で広く受け入れられ、成功を享受してきた。したがって、ｘ８６アーキテクチャに従ってプロセッサを設計することが有利である。このようなプロセッサは、ｘ８６アーキテクチャに書込まれた大量のソフトウェアから利益を得ることができる（このようなプロセッサはソフトウェアを実行でき、そのプロセッサを採用するコンピュータシステムは大量の利用可能なソフトウェアのために市場でより広く受け入れられ得るからである）。
【０００３】
コンピュータシステムは進化し続けてきたため、６４ビットアドレスサイズ（時としてオペランドサイズ）が望ましいものとなった。より大きなアドレスサイズによって、より大きなメモリフットプリント（プログラムによって動作させられるデータおよびプログラム内の命令によって占有されるメモリの量）を有するプログラムがメモリスペース内で動作することが可能となる。より大きなオペランドサイズによって、より大きなオペランド上で、より正確にはオペランド内での動作が可能となる。６４ビットアドレスおよび／またはオペランドサイズを用いると、より高性能なアプリケーションおよび／またはオペレーティングシステムが可能となり得る。
【０００４】
残念ながら、ｘ８６アーキテクチャは最大３２ビットオペランドサイズおよび３２ビットアドレスサイズに限定されている。オペランドサイズとは、プロセッサによって動作させられるビットの数（たとえば、ソースまたはデスティネーションオペランド内のビット数）を指す。アドレスサイズとは、プロセッサが生成するアドレス内のビットの数を指す。したがって、ｘ８６アーキテクチャを採用するプロセッサは、６４ビットアドレスサイズまたはオペランドサイズから利益を得ることができるアプリケーションのニーズを満たさないおそれがある。
【０００５】
【発明の概要】
上述の問題は、ここで説明されるようなプロセッサによって大部分が解決される。そのプロセッサは、アドレスサイズが３２ビットよりも大きい第１の処理モードをサポートする。アドレスサイズは公称６４ビットとして示され得るが、プロセッサの種々の実施例は、３２ビットよりも大きく６４ビット以下のあらゆるアドレスサイズを第１の処理モードで実現し得る。第１の処理モードは、制御レジスタ内のイネーブル指示をイネーブル状態にし、さらにはセグメント記述子内の第１の動作モード指示および第２の動作モード指示を予め規定された状態にセットする(set)ことによって、確立され得る。ｘ８６プロセッサアーキテクチャとの互換ができる３２ビットおよび１６ビット処理のための互換モードが提供されるように、（イネーブル指示はイネーブル状態のままで）第１の動作モード指示と第２の動作モード指示との他の組合せが用いられ得る。
【０００６】
イネーブル指示を介して第１の処理モードがイネーブルにされている間に互換モードは３２ビットまたは１６ビットコードが実行されることを可能にし得るが、第１の処理モード内で動作するコードを３２ビットまたは１６ビットコードから呼出すことが望ましいだろう。たとえば、アプリケーションプログラムが３２ビットまたは１６ビットモードで動作し得る間、オペレーティングシステムは第１の処理モードで動作し得る。セグメント記述子テーブル内で２つのエントリを占有する呼出ゲート記述子が規定される。そうでない場合には２つのエントリの各々がセグメント記述子を記憶し得る、その２つのエントリを占有することによって、呼出ゲート記述子は３２ビットを超えるアドレスを記憶するのに十分なスペースを含み得る。したがって、呼出コードセグメントが呼出ゲート記述子を参照し得るが、これはターゲットコードセグメントを参照し、さらにはターゲットコードセグメントのアドレススペース内でアドレスを提供し得る。以上のことは、たとえそのアドレスが呼出コードセグメント内でのアドレスサイズを超えていたとしても行なわれ得る。さらに、呼出ゲート記述子に２つのエントリを占有させることによって、セグメント記述子

テーブルは、互換モードセグメントのためのセグメント記述子を記憶し続け得る。したがって、呼出ゲート記述子および互換モードセグメント記述子はセグメント記述子テーブル内で共存し得る。加えて、呼出ゲート記述子が占有する第２のエントリ内のタイプフィールドであり得る区域を無効タイプにコーディングしてもよく、コードセグメントのための第２のエントリを誤って用いることによって、結果としてプロセッサが例外を知らせることになり得る。

【０００７】

概して、プロセッサが企図される。プロセッサは、セグメントセレクタを特定する分岐命令を実行するように構成される実行コアを含む。プロセッサは、セグメントセレクタに応答してセグメント記述子テーブルから少なくとも第１のエントリを読出すように構成され、第１のエントリが呼出ゲート記述子を指示する(indicate)と、セグメント記述子テーブル内の第２のエントリが呼出ゲート記述子の残りの部分を記憶する。加えて、コンピュータシステムが企図され、これは、コンピュータシステムとＩ／Ｏデバイスが結合可能な別のコンピュータシステムとの間の通信が行なわれるように構成される入力／出力（Ｉ／Ｏ）デバイスおよびプロセッサを含む。

【０００８】

さらに、方法が企図される。呼出ゲート記述子はセグメント記述子テーブルから読出される。呼出ゲート記述子はセグメント記述子テーブル内に第１のエントリと第２のエントリとを含み、第１のエントリと第２のエントリとの各々はセグメント記述子を記憶することが可能である。オフセットが呼出ゲート記述子から抽出される。オフセットは、実行されるべき第１の命令をターゲットコードセグメント内に位置付ける。

【０００９】

以下の詳細な説明を読み、さらには添付の図を参照することによって、この発明の他の目的および利点が明らかとなるだろう。

【００１０】

この発明には種々の変形および代替の形が可能であるが、その具体的な実施例は図における例によって示され、ここで詳細に説明される。しかし、図およびそれに対する詳細な説明は、この発明を開示されたある特定の形に限定することを意図するものではなく、逆に、追加の請求項によって規定されるようなこの発明の思想および範囲内にあるすべての変形、均等物、および代替物を包含することが意図されることが理解されるべきである。

【００１１】

【好ましい実施例の詳細な説明】

図１を参照して、プロセッサ１０のある実施例を例示するブロック図が示される。他の実施例も可能であり、企図される。図１の実施例では、プロセッサ１０は、命令キャッシュ１２、実行コア１４、データキャッシュ１６、外部インターフェイスユニット１８、メモリマネージメントユニット（ＭＭＵ）２０、およびレジスタファイル２２を含む。例示される実施例では、ＭＭＵ２０は、１組のセグメントレジスタ２４、第１の制御レジスタ２６、第２の制御レジスタ２８、ローカル記述子テーブルレジスタ（ＬＤＴＲ）３０、およびグローバル記述子テーブルレジスタ（ＧＤＴＲ）３２を含む。命令キャッシュ１２は、外部インターフェイスユニット１８、実行コア１４、およびＭＭＵ２０に結合される。実行コア１４はさらに、ＭＭＵ２０、レジスタファイル２２、およびデータキャッシュ１６に結合される。データキャッシュ１６はさらに、ＭＭＵ２０と外部インターフェイスユニット１８とに結合される。外部インターフェイスユニット１８はさらに、ＭＭＵ２０と外部インターフェイスとに結合される。

【００１２】

一般に、プロセッサ１０は、ｘ８６アーキテクチャと互換性があり、かつ６４ビット処理をサポートするための追加のアーキテクチャ特徴を含む、プロセッサアーキテクチャを採用する。プロセッサ１０は、現在実行されているコードに対応するコードセグメント記述子内に記憶された情報に応答して、さらには１つ以上の制御レジスタ内に記憶された１つ以上のイネーブル指示に応答して、動作モードを確立するように構成される。ここで用い

られるように、「動作モード」はプログラム的に選択可能な種々のプロセッサ属性のためのデフォルト値を指定する。たとえば、動作モードは、デフォルトオペランドサイズとデフォルトアドレスサイズとを指定し得る。デフォルトオペランドサイズは、命令の符号化がデフォルトをオーバーライドしない限りは、命令のオペランド内のビット数を指定する。デフォルトアドレスサイズは、命令の符号化がデフォルトをオーバーライドしない限りは、命令のメモリオペランドのアドレス内のビット数を指定する。デフォルトアドレスサイズは、メモリオペランドの少なくとも仮想アドレスサイズを指定し、また物理アドレスサイズも指定し得る。代替的には、物理アドレスサイズは、デフォルトアドレスサイズから独立していてもよく、代わりに以下で説明されるＬＭＥビットに依存し得るか（たとえば、物理アドレスは、ＬＭＥビットがクリア(clear)ならば、３２ビットであり得るが、ＬＭＥビットがセットされている(set)ならば、３２ビットよりも大きく６４ビットよりも小さい実現例によって異なるサイズであり得る）、または別の制御ビット（たとえば、別の制御レジスタ内の物理アドレス拡張ビットまたはＰＡＥビット）に依存し得る。ここで用いられるように、「仮想アドレス」とは、メモリにアクセスするために実際に用いられるアドレスである「物理アドレス」へとアドレス変換メカニズム（たとえば、ページングメカニズム）によって変換される前に生成されるアドレスである。加えて、ここで用いられるように、「セグメント記述子」とは、メモリのセグメントのための状況およびアクセスコントロールを規定するためにソフトウェアが作成し、かつプロセッサが用いるデータ構造である。「セグメント記述子テーブル」とは、多数のエントリを有するメモリ内のテーブルであり、各エントリはセグメント記述子を記憶することが可能である。

【００１３】

例示される実施例では、ＭＭＵ２０は動作モードを生成し、その動作モードを実行コア１４に伝達する。実行コア１４は、動作モードを用いて命令を実行する。より具体的には、実行コア１４は、（メモリオペランドがキャッシュ可能でデータキャッシュ１６でヒットするならばデータキャッシュ１６を通して、またはメモリオペランドがキャッシュ不可能でデータキャッシュ１６でミスとなるならば外部インターフェイスユニット１８を通して）レジスタファイル２２またはメモリからデフォルトオペランドサイズを有するオペランドを取出すが、以上は、ある特定の命令の符号化がデフォルトオペランドサイズをオーバーライドする場合以外のことであり、オーバーライドする場合には、オーバーライドする(overriding)オペランドサイズが用いられる。同様に、実行コア１４は、アドレスがデフォルトアドレスサイズを有する、メモリオペランドのアドレスを生成するが、これはある特定の命令の符号化がデフォルトアドレスサイズをオーバーライドしない限りにおいてのことであり、オーバーライドする場合には、オーバーライドするアドレスサイズが用いられる。他の実施例では、動作モードを用いるプロセッサ１０の部分（たとえば、実行コア１４）内で、動作モードを生成するために用いられる情報を局所的にバックアップ(shadow)することもでき、動作モードは局所的なシャドーコピー（shadow copies)から決定され得る。

【００１４】

上述のように、ＭＭＵ２０は、実行されているコードに対応するコードセグメント記述子に応答して、さらには制御レジスタ内の１つ以上の値に応答して、動作モードを生成する。コードセグメント記述子からの情報がセグメントレジスタ２４のうちの１つ（ＣＳまたはコードセグメントと呼ばれるレジスタ）の中に記憶される。加えて、制御レジスタ２６は、デフォルトアドレスサイズが３２ビットよりも大きい動作モード（「３２／６４モード」）と、３２ビットおよび１６ビット動作モードのためのある特定の互換モードとをイネーブルにするために用いられるイネーブル指示（ＬＭＥ）を記憶する。デフォルトオペランドサイズは３２／６４モードでは３２ビットであり得るが、所望ならば命令が６４ビットオペランドサイズでデフォルト３２ビットオペランドサイズをオーバーライドし得る。ＬＭＥ指示がイネーブル状態ならば、３２ビットモードおよび１６ビットモードに加えて３２／６４モードが用いられ得る。ＬＭＥ指示がディセーブル状態ならば、３２／６４ビットはディセーブルにされる。ある実施例では、３２／６４モードでのデフォルトアド

レスサイズは、実現例に依存するものであり得るが、６４ビット以下のいかなる値であってもよい。さらに、仮想アドレスのサイズは、所与の実現ではその実現での物理アドレスのサイズと異なり得る。

【００１５】

イネーブル指示は、イネーブル状態がビットのセットされた状態であり、かつディセーブル状態がビットのクリアにされた状態であるビットとしてここで説明され得ることが注目される。しかし、多数のビットが用いられる符号化と、イネーブル状態がクリア状態でありかつディセーブル状態がセット状態である符号化とを含む、他の符号化も可能である。したがって、この説明の残りの部分は、制御レジスタ２６内のＬＭＥ指示をＬＭＥビットとして呼び、ここではイネーブル状態がセットであり、ディセーブル状態がクリアであり得る。しかし、上述のように、ＬＭＥ指示の他の符号化も企図される。

【００１６】

セグメントレジスタ２４は、プロセッサ１０が実行しているコードが現在使っているセグメント記述子からの情報を記憶する。上述のように、ＣＳは、セグメントレジスタ２４のうちの１つであり、メモリのコードセグメントを指定する。コードセグメントは実行されているコードを記憶する。他のセグメントレジスタが種々のデータセグメントを規定し得る（たとえば、ＳＳセグメントレジスタが規定するスタックデータセグメント、ならびにＤＳセグメントレジスタと、ＥＳセグメントレジスタと、ＦＳセグメントレジスタと、ＧＳセグメントレジスタとが規定する最大４つのデータセグメント）。図１は例示的なセグメントレジスタ２４Ａの内容を例示し、これは、セレクタフィールド２４ＡＡおよび記述子フィールド２４ＡＢを含む。セレクタフィールド２４ＡＡは、実行コア１４が実行するある特定のセグメントロード命令に応答してセグメントセレクタでロードされてある特定のセグメントを活動化させる。セグメントセレクタは、メモリ内のセグメント記述子テーブルの中のセグメント記述子を識別する。より具体的には、プロセッサ１０は２つのセグメント記述子テーブル、すなわち、ローカル記述子テーブルとグローバル記述子テーブルとを採用し得る。ローカル記述子テーブルの基底アドレスがＬＤＴＲ３０内に記憶される。同様に、グローバル記述子テーブルの基底アドレスがＧＤＴＲ３２内に記憶される。セグメントセレクタ内のあるビット（テーブルインディケータビット）が記述子テーブルを選択し、セグメントセレクタの残りが、選択されたテーブルへのインデックスとして用いられる。ある命令がセグメントセレクタをセグメントレジスタ２４のうちの１つにロードすると、ＭＭＵ２０は、対応するセグメント記述子を選択されたセグメント記述子テーブルから読出し、セグメント記述子からの情報をセグメント記述子フィールド（たとえば、セグメントレジスタ２４Ａのためのセグメント記述子フィールド２４ＡＢ）に記憶する。セグメント記述子フィールド内に記憶される情報は、所望ならば、すべてのセグメント記述子を含む好適なあらゆるセグメント記述子のサブセットを含み得る。加えて、所望ならば、セグメント記述子または他のソースから引き出された他の情報がセグメント記述子フィールド内に記憶され得る。たとえば、ある実施例は、コードセグメント記述子から動作モード指示を復号化することもでき、動作モード指示の元の値というよりはむしろ復号化された値を記憶し得る。ある命令によってＣＳがセグメントセレクタでロードされると、コードセグメントは変化して、したがってプロセッサ１０の動作モードが変化し得る。セグメント記述子テーブルは以下でより詳細に説明される。

【００１７】

ある実施例では、ＣＳセグメントレジスタのみが３２／６４モードで用いられる。データセグメントレジスタは無視される。１６ビットモードおよび３２ビットモードでは、コードセグメントおよびデータセグメントが活動状態になり得る。さらに、制御レジスタ２８内の第２のイネーブル指示（ＰＥ）がＭＭＵ２０の動作に影響を及ぼし得る。ＰＥイネーブル指示を用いて保護モードをイネーブルにすることもでき、ここではセグメンテーションおよび／またはページングアドレス変換メカニズムが用いられ得る。ＰＥイネーブル指示がディセーブル状態にある場合には、セグメンテーションおよびページングメカニズムがディセーブルにされ、プロセッサ１０は「実モード」となる（ここでは、実行コア１４

が生成するアドレスは物理アドレスである）。ＬＭＥ指示と同様に、ＰＥ指示は、イネーブル状態はセットされたビットで、かつディセーブル状態はクリアなビットである、ビットであり得る。しかし、上述のように他の実施例も企図される。

【００１８】

ＭＭＵ２０は、所望であれば、さらなるハードウェアメカニズムを採用し得ることが注目される。たとえば、ＭＭＵ２０は、仮想アドレスから物理アドレスへのページングアドレス変換を実現するためのページングハードウェアを含み得る。ページングハードウェアは、ページ変換を記憶するための変換ルックアサイドバッファ（ＴＬＢ）を含み得る。

【００１９】

制御レジスタ２６および２８はアーキテクトされた制御レジスタとして実現され得る（たとえば、制御レジスタ２６はＣＲ４で、制御レジスタ２８はＣＲ０であり得る）ことが注目される。代替的には、制御レジスタのうちの１つまたは両者がモデル専用レジスタとして実現されて３２／６４モードに干渉することなしにアーキテクトされた制御レジスタの他の用途が可能にされてもよい。

【００２０】

一般に、命令キャッシュ１２は、命令バイトを記憶するための高速キャッシュメモリである。実行コア１４は、実行のために命令キャッシュ１２から命令を取出す。命令キャッシュ１２は、直接マッピング、セットアソシエーティブ、および完全にアソシエーティブな構成を含む、好適なあらゆるキャッシュ機構を採用し得る。命令の取出が命令キャッシュ１２内でミスとなると、命令キャッシュ１２は外部インターフェイスユニット１８と通信してミッシングキャッシュラインを命令キャッシュ１２へと入れる(fill)こともできる。加えて、命令キャッシュ１２はＭＭＵ２０と通信して命令キャッシュ１２から取出された仮想アドレスのための物理アドレス変換を受取り得る。

【００２１】

実行コア１４は命令キャッシュ１２から取出された命令を実行する。実行コア１４は、レジスタファイル２２からレジスタオペランドを取出し、レジスタファイル２２内のデスティネーションレジスタを更新する。レジスタオペランドのサイズは、動作モードと、ある特定の命令のための動作モードのすべてのオーバーライドとによって制御される。同様に、実行コア１４は、メモリオペランドのキャッシュ性(cacheability)およびデータキャッシュ１６内でのヒットを条件として、データキャッシュ１６からメモリオペランドを取出し、データキャッシュ１６内のデスティネーションメモリロケーションを更新する。メモリオペランドのサイズも同様に、動作モードと、ある特定の命令のための動作モードのすべてのオーバーライドとによって制御される。さらに、実行コア１４が生成するメモリオペランドのアドレスのサイズは、動作モードと、ある特定の命令のための動作モードのすべてのオーバーライドとによって制御される。

【００２２】

実行コア１４は好適なあらゆる構造を採用し得る。たとえば、実行コア１４は、スーパーパイプラインコア、スーパースカラコア、またはそれらの組合せであり得る。実行コア１４は、設計上の選択に従って、アウトオブオーダ投機的実行またはインオーダ実行を採用し得る。

【００２３】

レジスタファイル２２は６４ビットレジスタを含み得るが、これは、プロセッサ１０の動作モードおよびある特定の命令のためのあらゆるオーバーライドによって指示されるように６４ビットレジスタ、３２ビットレジスタ、１６ビットレジスタ、または８ビットレジスタとしてアクセスされてもよい。ある実施例のレジスタフォーマットが以下で図７に関して説明される。レジスタファイル２２内に含まれるレジスタは、ＬＥＡＸレジスタ、ＬＥＢＸレジスタ、ＬＥＣＸレジスタ、ＬＥＤＸレジスタ、ＬＥＤＩレジスタ、ＬＥＳＩレジスタ、ＬＥＳＰレジスタ、およびＬＥＢＰレジスタを含み得る。レジスタファイル２２はさらにＬＥＩＰレジスタを含み得る。代替的には、実行コア１４は、レジスタファイル２２内のあらゆるレジスタがアーキテクトされたレジスタにマッピングされ得るレジスタ

リネームの形を採用してもよい。レジスタファイル２２内のレジスタの数は、このような実施例については実現例によって異なり得る。

【００２４】

データキャッシュ１６は、データを記憶するように構成された高速キャッシュメモリである。データキャッシュ１６は、直接マッピング、セットアソシエーティブ、および完全にアソシエーティブな構成を含む、好適なあらゆるキャッシュ機構を採用し得る。データの取出または更新がデータキャッシュ１６内でミスとなると、データキャッシュ１６は外部インターフェイスユニット１８と通信してデータキャッシュ１６へとミッシングキャッシュラインを入れることもできる。加えて、データキャッシュ１６がライトバックキャッシュポリシーを採用する場合には、データキャッシュ１６から外に出される(cast out）更新されるキャッシュラインが外部インターフェイスユニット１８に伝達されてメモリにライトバックされ得る。データキャッシュ１６はＭＭＵ２０と通信してデータキャッシュ１６に与えられる仮想アドレスのための物理アドレス変換を受取り得る。

【００２５】

外部インターフェイスユニット１８は、プロセッサ１０の外部のシステムの部分と通信を行なう。外部インターフェイスユニット１８は、上述のような命令キャッシュ１２およびデータキャッシュ１６のためにキャッシュラインを伝達し、ＭＭＵ２０とも通信を行ない得る。たとえば、外部インターフェイスユニット１８は、ＭＭＵ２０に代わってセグメント記述子テーブルおよび／またはページングテーブルにアクセスし得る。

【００２６】

所望ならば、プロセッサ１０は統合レベル(integrated level)２（Ｌ２）キャッシュを含み得ることが注目される。さらに、外部インターフェイスユニット１８は、システムとの通信に加えてバックサイドキャッシュと通信するように構成され得る。

【００２７】

図２を参照して、３２／６４モードのためのコードセグメント記述子４０のある実施例のブロック図が示される。他の実施例も可能であり、企図される。図２の実施例では、コードセグメント記述子４０は８バイトを含み、最下位４バイトの上方に最上位４バイトが例示される。最上位４バイトは最下位４バイトよりも数の上でより大きなアドレスに記憶される。４バイトの各グループの最上位ビットは、図２（および以下の図３）でビット３１として示され、最下位ビットはビット０として示される。（図２および図３の両方において）４バイト内の短い垂直の線は各々のビットを区切り、長い垂直の線もビットを区切るが、それはまたフィールドも区切る。

【００２８】

以下の図３で示される３２ビットおよび１６ビットコードセグメント記述子とは異なって、コードセグメント記述子４０は基底アドレスまたはリミット(limit)を含まない。プロセッサ１０は（３２ビットおよび１６ビットモードで採用されるセグメント化されたリニアアドレススペースというよりはむしろ）３２／６４モードのためにフラット(flat)仮想アドレススペースを採用する。したがって、それ以外の場合には基底アドレスおよびリミットを記憶するであろうコードセグメント記述子４０の部分は、セグメント記述子４０内でリザーブされる。セグメンテーションを通して提供される仮想アドレスはここで「リニアアドレス」とも呼ばれ得ることが注目される。「仮想アドレス」という用語は、セグメント化されていないアーキテクチャ内で生成される他の仮想アドレスおよびリニアアドレスを含む、メモリをアドレス指定するために実際に用いられる物理アドレスへと変換メカニズムを通して変換されるすべてのアドレスを含む。

【００２９】

セグメント記述子４０は、Ｄビット４２、（３２／６４モードコードセグメントのために１にセットされる）Ｌビット４４、使用可能(available)ビット（ＡＶＬ）４６、現在の（Ｐ）ビット４８、記述子特権レベル（ＤＰＬ）５０、およびタイプフィールド５２を含む。以下の図５で示されるように、Ｄビット４２とＬビット４４とはプロセッサ１０の動作モードを決定するために用いられる。ＡＶＬビット４６は、システムソフトウェア（た

とえば、動作システム）によって用いられることのために使用可能である。Ｐビット４８が用いられてセグメントがメモリ内に存在するか否かが示される。Ｐビット４８がセットされていると、セグメントは存在し、コードがセグメントから取出され得る。Ｐビット４８がクリアであると、セグメントは存在せず、例外が生じて（たとえば、ディスク記憶デバイスから、またはネットワーク接続を通して）セグメントがメモリへとロードされる。ＤＰＬはセグメントの特権レベルを示す。プロセッサ１０は、（ＤＰＬフィールド内で０から３として符号化され、かつレベル０が最も特権を与えられたレベルである）４つの特権レベルを採用する。ある特定の命令およびプロセッサリソース（たとえば、構成レジスタおよび制御レジスタ）は、より特権を与えられたレベルでのみ実行可能またはアクセス可能であり、より低い特権レベルでこれらの命令を実行すること、またはこれらのリソースにアクセスすることを試みることによって、結果として例外が生じる。コードセグメント４０からの情報がＣＳセグメントレジスタにロードされると、ＤＰＬがプロセッサ１０の現在の特権レベル（ＣＰＬ）となる。タイプフィールド５２はセグメントのタイプを符号化する。コードセグメントでは、タイプフィールド５２の最上位ビット２ビットがセットされ得る（最上位ビットはシステムセグメントからコードまたはデータセグメントを区別し、第２の最上位ビットはデータセグメントからコードセグメントを区別する）。残りのビットはさらなるセグメントタイプ情報（たとえば、実行専用、実行および読出、または実行および読出専用、コンフォーミング、およびコードセグメントがアクセスされたか否か）を符号化し得る。

【００３０】

コードセグメント記述子内のいくつかの指示は、セットされた値およびクリアされた値が規定の意味を有するビットとして説明されるが、他の実施例は所望であれば反対の符号化を採用してもよく、多数のビットを用いてもよいことが注目される。したがって、たとえば、Ｄビット４２とＬビット４４との各々は、上述のイネーブル指示の考察と同様に、所望であれば１つ以上のビットであり得る動作モード指示の例であり得る。

【００３１】

図３を参照して、３２および１６ビット互換モードのためのコードセグメント記述子５４のある実施例のブロック図が示される。他の実施例も可能であり、企図される。図２の実施例と同様に、コードセグメント記述子５４も８バイトを含み、最下位４バイトの上方に最上位４バイトが示される。

【００３２】

コードセグメント記述子５４は、コードセグメント記述子４０の上の説明と同様に、Ｄビット４２、Ｌビット４４、ＡＶＬビット４６、Ｐビット４８、ＤＰＬ５０、およびタイプフィールド５２を含む。加えて、コードセグメント記述子５４は、基底アドレスフィールド（参照番号５６Ａ、５６Ｂ、および５６Ｃ）、リミットフィールド（参照番号５７Ａおよび５７Ｂ）、およびＧビット５８を含む。基底アドレスフィールドは、（ＬＥＩＰレジスタ内に記憶される）ロジカル取出アドレスに加えられる基底アドレスを記憶してある命令のリニアアドレスを形成し、これは次に任意でページング変換メカニズムを通して物理アドレスに変換され得る。リミットフィールドは、セグメントのサイズを規定するセグメントリミットを記憶する。セグメントリミットよりも大きな論理アドレスのバイトにアクセスする試みは、認められず、例外を生じさせる。Ｇビット５８はセグメントリミットフィールドのスケーリングを決定する。Ｇビット５８がセットされると、リミットは４Ｋバイトページにスケーリングされる（たとえば、１２の最下位０がリミットフィールド内のリミットに追加される）。Ｇビット５８がクリアならば、リミットはそのまま用いられる。

【００３３】

３２／６４モードが制御レジスタ２６内のＬＭＥビットを介してイネーブルにされていない時の３２および１６ビットモードのためのコードセグメント記述子は、Ｌビットがリザーブされていて０であるように規定されている以外は、コードセグメント記述子５４と同様のものであり得ることが注目される。ある実施例に従った３２ビットモードおよび１６

ビットモード（ＬＭＥビットがセットされた互換モードと、ＬＭＥビットがクリアにされたモードとの両者）では、データセグメントも用いられることがさらに注目される。データセグメント記述子は、Ｄビット４２がセグメントの上限を指示するように規定されているか、または（スタックセグメントのための）デフォルトスタックサイズを定めるように規定されていること以外は、コードセグメント記述子５４と同様のものであり得る。

【００３４】

次に図４を参照して、制御レジスタ２６内のＬＭＥビットと、３２／６４モード、ならびに３２ビットモードおよび１６ビットモードを実現する高度な柔軟性を可能にするための互換モードとの例示的な用途を例示する図が示される。ボックス６０はＬＭＥビットがセットされているときの例示的な動作を示し、ボックス６２はＬＭＥビットがクリアであるときの例示的な動作を示す。

【００３５】

ボックス６０で示されるように、ＬＭＥビットがセットされているときにサポートされる互換モードは、６４ビットオペレーティングシステム（つまり、３２ビットを超える仮想アドレススペースおよび物理アドレススペース、および／または６４ビットのデータオペランドを利用するように設計されるオペレーティングシステム）が３２ビットアプリケーションプログラム（つまり、３２ビットのオペランドサイズおよびアドレスサイズを用いて書込まれるアプリケーションプログラム）で動作することを可能にし得る。オペレーティングシステムのためのコードセグメントは、図２で示される３２／６４モードコードセグメント記述子４０によって規定され得るため、Ｌビットがセットされ得る。したがって、オペレーティングシステムは、（たとえば、セグメント記述子テーブルおよびページング変換テーブルを含む）オペレーティングシステムが維持するデータ構造およびオペレーティングシステムコードのために拡張された仮想アドレススペースおよび物理アドレススペースを利用することができる。オペレーティングシステムはまた、デフォルト３２ビットオペランドサイズをオーバーライドする命令符号化を用いて３２／６４モードで規定される６４ビットデータタイプを用いることもできる。さらに、オペレーティングシステムは、セグメント記述子テーブル内で１つ以上の３２ビット互換モードセグメント記述子（クリアにされたＬビット、セットされたＤビット、たとえば、図２で示されるセグメント記述子５４）を確立し、さらには互換モードセグメントのうちの１つへと分岐することによって、３２ビットアプリケーションプログラムを実行することもできる。同様に、オペレーティングシステムは、セグメント記述子テーブル内で１つ以上の１６ビット互換モードセグメント記述子（クリアにされたＬビット、クリアにされたＤビット、たとえば、図２で示されるセグメント記述子５４）を確立させ、さらには互換モードセグメントのうちの１つへと分岐することによって、１６ビットアプリケーションプログラムを実行することもできる。したがって、６４ビットオペレーティングシステムは、互換モードで既存の３２ビットアプリケーションプログラムおよび１６ビットアプリケーションプログラムを実行するための能力を保持することができる。ある特定のアプリケーションプログラムは、拡張された能力がそのプログラムに望まれると３２／６４モードに移され得るか、または３２ビットまたは１６ビットのままであり得る。

【００３６】

プロセッサ１０が３２ビットアプリケーションプログラムを実行している間、プロセッサ１０の動作モードは３２ビットである。したがって、アプリケーションプログラムは一般に、それが（たとえば、オペレーティングシステムも３２ビットオペレーティングシステムである時）ＬＭＥビットがクリアな状態で３２ビットモードで実行されるのと同じ態様で、実行され得る。しかし、アプリケーションプログラムは、オペレーティングシステムサービスを呼出し、例外を経験するか、または終了し得る。これらの事例ごとに、プロセッサ１０は（図４の矢印６４で示されるように）オペレーティングシステムコードの実行に戻り得る。オペレーティングシステムコードは３２／６４モードで動作するため、オペレーティングシステムサービスルーチン、例外ハンドラ等のアドレスは３２ビットを超え得る。したがって、プロセッサ１０は、オペレーティングシステムコードに戻る前に３２

ビットよりも大きなアドレスを生成する必要があり得る。ＬＭＥビットは、現在の動作モードがたとえ３２ビットであったとしてもオペレーティングシステムは３２／６４モードで動作しているかもしれないという表示をプロセッサ１０に提供し、このようにして、プロセッサ１０は、オペレーティングシステム呼出および例外のためにより大きなアドレススペースを提供し得る。

【００３７】

ある実施例では、割込セグメント記述子テーブル内に記憶される割込セグメント記述子を用いて、例外が処理される。ＬＭＥビットがセットされていると、割込セグメント記述子は、例外を処理するオペレーティングシステムルーチンの６４ビットアドレスを含む１６バイトエントリであり得る。ＬＭＥビットがクリアであると、割込セグメント記述子は、３２ビットアドレスを含む８バイトエントリであり得る。したがって、プロセッサ１０はＬＭＥ指示に応答して割込記述子テーブルにアクセスする（つまり、ＬＭＥビットがセットされていると１６バイトエントリを読出し、ＬＭＥビットがクリアであると８バイトエントリを読出す）。したがって、例外は、たとえアプリケーションプログラムが３２ビット互換モードで実行されていても、６４ビットオペレーティングシステムによって処理され得る。さらに、プロセッサ１０は、ＬＭＥビットがクリアならば３２ビット（または１６ビット）オペレーティングシステムをサポートする。

【００３８】

同様に、プロセッサ１０内の呼出メカニズムは、ＬＭＥビットの状態に基づいて異なった態様で動作し得る。オペレーティングシステムは典型的にはアプリケーションプログラムよりもより高い特権レベルで実行されるため、アプリケーションプログラムからオペレーティングシステムへの転送(transfers)は、慎重に制御されてアプリケーションプログラムは許可されたオペレーティングシステムルーチンの実行のみ可能であることが確実にされる。より一般的には、特権レベルの変化は慎重に制御される。ある実施例では、プロセッサ１０は、オペレーティングシステム呼出を行なうために少なくとも２つのメカニズムをサポートし得る。１つの方法は、（以下でより詳細に説明される）セグメント記述子テーブル内の呼出ゲートを通してのものであり得る。別の方法は、プロセッサ１０がサポートするＳＹＳＣＡＬＬ命令であり得るが、これはモデル専用レジスタをオペレーティングシステムルーチンのアドレスのソースとして用いる。モデル専用レジスタの更新は特権を与えられた動作であり、したがって、より高い特権レベルで実行されているコード（たとえば、オペレーティングシステムコード）のみが、ＳＹＳＣＡＬＬ命令によって用いられるモデル専用レジスタ内にアドレスを確立することができる。ＳＹＳＣＡＬＬ方法では、第２のモデル専用レジスタが、オペレーティングシステムルーチンのアドレスの最上位３２ビットを記憶するように規定され得る。したがって、ＬＭＥビットがセットされていると、アドレスは２つのモデル専用レジスタから読出され得る。ＬＭＥビットがクリアであると、アドレスは最下位３２ビットを記憶するモデル専用レジスタから読出され得る。代替的には、ＳＹＳＣＡＬＬ命令が用いるモデル専用レジスタは６４ビットに拡張されてもよく、アドレスは、ＬＭＥビットの状態に基づいて３２ビット（モデル専用レジスタの最下位３２ビット）または６４ビットであり得る。

【００３９】

上述のように、ＬＭＥビットをセットすることによって、オペレーティングシステムは６４ビットであり、かつ１つ以上のアプリケーションプログラムは６４ビットではない（たとえば、図示されるように３２ビットまたは上の説明と同様の様態で動作する１６ビットである）システムでプロセッサ１０が動作することが可能となり得る。加えて、ボックス６２で示されるように、ＬＭＥビットをクリアにすることによって、ｘ８６アーキテクチャと互換性がある３２ビットまたは１６ビットモードでプロセッサ１０が動作することが可能となり得る。上述のように、例外およびオペレーティングシステム呼出を処理するためのメカニズムは、セットされたまたはクリアなＬＭＥビットを処理するように設計され、したがって、たとえプロセッサ１０が３２／６４モードで動作可能であっても、３２ビットおよび１６ビットモードは変更されずに動作し得る。さらに、ＬＭＥビットがクリア

なときにｘ８６と互換性のある１６ビットおよび３２ビットモードを提供することによって（さらには、これらのモード内にリザーブされるＬビットを無視することによって）、プロセッサ１０は、Ｌビットが３２／６４モード以外の他のいくつかの目的のために規定されるシステムで動作し、ＬＭＥビットがセットされると依然として３２／６４モードをサポートし得る。したがって、３２ビットオペレーティングシステムおよび３２ビットまたは１６ビットアプリケーションプログラムを採用するシステムは、プロセッサ１０を採用することができる。続いて、システムは、プロセッサ１０を変える必要なしに６４ビットオペレーティングシステムへとアップグレードされ得る。

【００４０】

図４で例示されていないものは、ＬＭＥビットがセットされた状態で動作する６４ビットオペレーティングシステムおよび６４ビットアプリケーションプログラムである。６４ビットオペレーティングシステムおよび３２ビットアプリケーションプログラムのために上で説明されたオペレーティングシステムルーチンの呼出のためのメカニズムは、６４ビットアプリケーションプログラムにも等しく当てはまり得る。加えて、（以下でより詳細に説明されるように）６４ビットオフセットをサポートする呼出ゲートがサポートされる。

【００４１】

次に図５を参照して、プロセッサ１０のある実施例に従った、ＬＭＥビット、コードセグメント記述子内のＬビット、およびコードセグメント記述子内のＤビットの状態と、プロセッサ１０の対応する動作モードとを例示する表７０が示される。他の実施例も可能であり、企図される。表７０が示すように、ＬＭＥビットがクリアならば、Ｌビットはリザーブされる（０と規定される）。しかし、プロセッサ１０は、ＬＭＥビットがクリアならば、Ｌビットをドントケアとして扱い得る。このようにして、ＬＭＥビットがクリアならば、ｘ８６と互換性のある１６ビットおよび３２ビットモードがプロセッサ１０によって提供され得る。ＬＭＥビットがセットされていてコードセグメント内のＬビットがクリアならば、プロセッサ１０によって互換動作モードが確立され、Ｄビットが１６ビットまたは３２ビットモードを選択する。ＬＭＥビットおよびＬビットがセットされていてＤビットがクリアならば、プロセッサ１０のために３２／６４モードが選択される。最後に、ＬＭＥビット、Ｌビット、およびＤビットがすべてセットされているならば選択されるであろうモードがリザーブされる。

【００４２】

上で言及され、以下の図６で例示されるように、３２／６４動作モードは（実現例に依存するが最大６４ビットの）３２ビットを超えるデフォルトアドレスサイズと、３２ビットのデフォルトオペランドサイズとを含む。３２ビットのデフォルトオペランドサイズは、ある特定の命令の符号化を介して６４ビットへとオーバーライドされてもよい。プログラムが行なうデータ操作の多くにとって３２ビットが十分であるプログラムのために平均命令長を最小にするために、３２ビットのデフォルトオペランドサイズが選択される（なぜならば、６４ビットへとオーバーライドすることは命令符号化に命令プレフィックスを含むことを伴い、これによって命令長が増大させられ得るためである）。（現在存在するプログラムのうちのかなりの数であり得る）このようなプログラムでは、６４ビットオペランドサイズへと移ることによって、プログラムが達成する実行性能が実際に減じられるおそれがある（つまり、実行時間が増大する）。この減少は、６４ビット値が記憶されるときにプログラムが用いるデータ構造のメモリサイズの倍増(doubling)にある程度起因し得る。３２ビットが十分ならば、これらのデータ構造は３２ビット値を記憶する。したがって、データ構造がアクセスされるときにアクセスされるバイトの数は、３２ビット値が十分であり得るときに６４ビット値が用いられると増大し、増大したメモリ帯域幅（および各値によって占有される増大したキャッシュスペース）によって、実行時間が増大するおそれがある。したがって、デフォルトオペランドサイズとして３２ビットが選択され、ある特定の命令の符号化を介してデフォルトがオーバーライドされ得る。

【００４３】

次に図６を参照して、ある特定の命令のために動作モードをオーバーライドするための命

令プレフィックスの用途のある実施例を例示する表７２が示される。他の実施例も可能であり、企図される。実行コア１４は、表７２に従ってある特定の命令のためにアドレスサイズおよびオペランドサイズを決定する。特に図６で示される実施例では、命令プレフィックスバイト（アドレスサイズオーバーライドプレフィックスバイト）が用いられてデフォルトアドレスサイズがオーバーライドされ、別の命令プレフィックスバイト（オペランドサイズオーバーライドプレフィックスバイト）が用いられてデフォルトオペランドサイズがオーバーライドされ得る。アドレスサイズオーバーライドプレフィックスバイトは（１６進法では）６７として符号化され、オペランドサイズオーバーライドプレフィックスバイトは（１６進法では）６６として符号化される。ある特定の命令でのオーバーライドプレフィックスの数によって、表の列が形成される。表の行は、動作モードおよび対応する列内のオーバーライドプレフィックスの数に基づいて、ある特定の命令のオペランドサイズおよびアドレスサイズを示す。オーバーライドプレフィックスの数は、対応するタイプのオーバーライドプレフィックスの数を指す（たとえば、アドレスサイズの行は、アドレスサイズオーバーライドプレフィックスの数に基づいたアドレスサイズであり、オペランドサイズの行は、オペランドサイズオーバーライドプレフィックスの数に基づいたオペランドサイズである）。

【００４４】

オーバーライドプレフィックスの数で「０」とラベル付けされる列は、各動作モードのためのデフォルトオペランドサイズおよびアドレスサイズを示す。３２ビットモードの行および１６ビットモードの行は、互換モード（ＬＭＥセット）と標準モード（ＬＭＥクリア）との両者を指すことが注目される。さらに、デフォルトアドレスサイズは３２／６４モードでは６４ビットであるが、アドレスビットの実際の数は、上述のように実現例に依存したものであり得る。

【００４５】

表７２で示されるように、３２／６４ビットモード内に１つのアドレスサイズオーバーライドプレフィックスを含むことによって、アドレスサイズが６４ビット（所与の実現では６４ビットよりも少ないものであり得るが、３２ビットよりは大きい）から３２ビットへと変化する。加えて、３２／６４ビットモード内に１つのオペランドサイズオーバーライドプレフィックスを含むことによって、オペランドサイズが３２ビットから６４ビットへと変化する。（たとえば、「Ｃ」プログラミング言語において短整数データ型をサポートするために）１６ビットオペランドにも備えることが所望であり得る。したがって、３２／６４モード内に２つのオペランドサイズオーバーライドプレフィックスを含むことによって、１６ビットのオペランドサイズが選択される。２つよりも多いオペランドサイズオーバーライドプレフィックスを含むことによって結果として、２つのオペランドサイズオーバーライドプレフィックスを含むときと同じオペランドサイズが得られる。同様に、１つよりも多いアドレスサイズオーバーライドプレフィックスを含むことによって結果として、１つのアドレスサイズオーバーライドプレフィックスを含むときと同じアドレスサイズが得られる。

【００４６】

３２ビットモードでは、１つのオーバーライドプレフィックスを含むことによって、デフォルト３２ビットサイズが１６ビットへと切換えられ、１つよりも多いオーバーライドプレフィックスを含むことは、１つのオーバーライドプレフィックスを含むことと同じ効果を有する。同様に、１６ビットモードでは、１つのオーバーライドプレフィックスを含むことによって、デフォルト１６ビットサイズが３２ビットに切換えられ、１つよりも多いオーバーライドプレフィックスを含むことは、１つのオーバーライドプレフィックスを含むことと同じ効果を有する。

【００４７】

図７を参照して、ＬＥＡＸレジスタ７４のある実施例を例示する図が示される。レジスタファイル２２内の他のレジスタも同様であり得る。他の実施例も可能であり、企図される。図７の実施例では、レジスタ７４は６４ビットを含み、最上位ビットはビット６３とし

てラベル付けされ、最下位ビットはビット０としてラベル付けされる。図７は、（Ａレジスタがオペランドとして選択された場合）命令のオペランドサイズに基づいてアクセスされるＬＥＡＸレジスタの部分を示す。より具体的には、オペランドサイズが６４ビットならば、（図７で「ＬＥＡＸ」とラベル付けされたブレース(brace)によって示されるように）レジスタ７４全体がアクセスされる。オペランドサイズが３２ビットならば、（図７で「ＥＡＸ」とラベル付けされたブレースによって示されるように）レジスタ７４のビット３１：０がアクセスされる。オペランドサイズが１６ビットならば、（図７で「ＡＸ」とラベル付けされたブレースによって示されるように）レジスタのビット１６：０がアクセスされる。上述のオペランドサイズは、動作モードおよびオーバーライドプレフィックスのいずれかを含むことに基づいて選択され得る。しかし、８ビットレジスタ（図７のＡＨまたはＡＬ）にアクセスするある特定の命令演算コードが規定される。

【００４８】

次に図８を参照して、グローバル記述子テーブル８０およびローカル記述子テーブル８２のある実施例を例示するブロック図が示される。他の実施例も可能であり、企図される。図８で示され、かつ上で言及されるように、グローバル記述子テーブル８０の基底アドレスはＧＤＴＲ３２によって提供され、ローカル記述子テーブル８２の基底アドレスはＬＤＴＲ３０によって提供される。したがって、仮想アドレススペース内にグローバル記述子テーブル８０とローカル記述子テーブル８２とを恣意的に置くことをサポートするために、ＧＤＴＲ３２とＬＤＴＲ３０とは６４ビット基底アドレスを記憶し得る。ＬＭＥビットがクリアならば、基底アドレスの最下位３２ビットが用いられて記述子テーブルが位置付け(locate)られ得る。

【００４９】

グローバル記述子テーブル８０とローカル記述子テーブル８２との両者は、種々の種類のセグメント記述子を記憶するように構成される。たとえば、３２／６４モードコードセグメント記述子８４、８６、および９０と、互換モード記述子９２および９４とが図８で示される。記述子８４－９４の各々は対応する記述子テーブル内でエントリを占有し、ここではエントリは１つのセグメント記述子（たとえば、図２および図３で示される実施例では８バイト）を記憶することができる。グローバル記述子テーブル８０内の別の種類の記述子はローカル記述子テーブル記述子９６であり、これは、ローカル記述子テーブル８２のためにシステムセグメントを規定し、ＬＤＴＲ３０内に記憶される基底アドレスを提供する。ＬＤＴＲ３０は、グローバル記述子テーブル８０内に記述子９６を位置付けるセグメントセレクタをオペランドとして有するＬＬＤＴ命令を用いて初期設定される。グローバル記述子テーブル８０は、所望ならば、異なったローカル記述子テーブルを位置付ける多数のＬＤＴ記述子を記憶し得る。ＬＭＥビットがセットされているとＬＤＴ記述子９６は６４ビットオフセットを記憶し得るため、ＬＤＴ記述子９６はグローバル記述子テーブル８０内に２つのエントリを占有し得る。ＬＭＥビットがクリアであると、ＬＤＴ記述子９６はグローバル記述子テーブル８０内に単一のエントリを占有し得る。同様に、各々のタスクは、記述子テーブル８０および８２のうちの１つの中にタスク状態セグメント（ＴＳＳ）記述子を有してタスクに関連するある特定の情報を記憶し得る。したがって、ＴＳＳ記述子は２つのエントリを占有してＴＳＳ情報が６４ビットアドレススペース内のどこにでも記憶されることを可能にし得る。

【００５０】

ローカル記述子テーブルおよびグローバル記述子テーブルはまた、呼出ゲート記述子を記憶することもできる。たとえば、図８は呼出ゲート記述子１００、１０２、および１０４を示す。呼出ゲート記述子は、６４ビットオフセットもサポートし、したがって、対応する記述子テーブル内で２つのエントリを占有し得る。例示的な３２／６４呼出ゲート記述子が以下の図９で示される。

【００５１】

セグメント記述子テーブル８０および８２を８バイトで維持し、かつ６４ビットオフセットを含む記述子のために２つのエントリを用いることによって、１６ビットモードおよび

３２ビットモードのための記述子は、６４ビットオフセットを含む記述子と同じテーブル内に記憶され得る。したがって、互換モードで動作するアプリケーションは、６４ビットオペレーティングシステムと同じセグメント記述子テーブル内で適切な記述子を有することができる。

【００５２】

一般に、呼出ゲートが用いられてより小さい特権レベルを有するコードセグメントとより大きい特権レベルを有するコードセグメント（たとえば、オペレーティングシステムルーチンを呼出すアプリケーションプログラム）との間の遷移が管理される。より少ない特権を与えられたコードは、呼出または、ターゲットとしてセグメントセレクタ（および、この事例では無視されるが、セグメントへのオフセット）を特定する他の分岐命令を含む。セグメントセレクタは、記述子テーブル内の呼出ゲート記述子を識別し、これは、より大きな特権レベルコードを実行するのに必要とされる最小の特権レベルを含む。プロセッサ１０が呼出または他の分岐命令を実行するとき、プロセッサ１０はセグメントセレクタで記述子テーブルに索引をつけ、呼出ゲートを位置付ける。プロセッサ１０の現在の特権レベルと、（セグメントセレクタの一部であり、特権検査目的のために現在の特権レベルを下げるために用いられ得る）リクエスタ(requestor)特権レベルとの両者が十分な特権を反映していると（たとえば、特権レベルは呼出ゲート記述子内の最小特権レベルよりも数の上において少ないかまたはそれと等しいと）、呼出が進み得る。呼出ゲート記述子は、ターゲットセグメント（より大きな特権レベルを有するコードセグメント）のためのセグメントセレクタと、コード取出がそこで開始されるターゲットセグメント内のオフセットとを含む。プロセッサ１０は、呼出ゲート記述子からセグメントセレクタおよびオフセットを抽出し、ターゲットセグメント記述子を読出してより大きな特権レベルを有するコードの取出を開始する。一方で、現在の特権レベルまたはリクエスタ特権レベルのいずれかが呼出ゲート記述子内の最小特権レベルよりも小さい特権レベルであるならば（たとえば、現在の特権レベルまたはリクエスタ特権レベルのいずれかが最小特権レベルよりも数の上で大きいならば）、プロセッサ１０は、呼出ゲート記述子にアクセスした後、かつターゲット記述子にはアクセスすることなしに、例外を知らせる。したがって、より大きな特権レベルで実行されているコードへのアクセスは慎重に制御される。

【００５３】

上述のように、呼出ゲート記述子は、ターゲットセグメントセレクタおよびセグメント内のオフセットを含む。ターゲットセグメント記述子への参照は、呼出ゲート記述子から別の記述子への矢印として図８で例示される。たとえば、呼出ゲート記述子１００はモード記述子９０を参照し、呼出ゲート記述子１０２は３２／６４モード記述子８６を参照し、呼出ゲート記述子１０４は３２／６４モード記述子８４を参照する。図８で示されるように、呼出ゲート記述子は両方の記述子テーブル内に記憶されてもよく、他方のテーブル内または同じテーブル内の記述子を参照し得る。さらに、呼出ゲート記述子は、３２／６４モード記述子または互換モード記述子のいずれかを参照し得る。

【００５４】

一般に、プロセッサ１０がセグメントセレクタを用いて記述子テーブルのうちの１つから記述子を読出すと、１つの記述子テーブルエントリが読出される。しかし、ＬＭＥビットがセットされていて、かつプロセッサ１０によってエントリが呼出ゲート記述子、ＬＤＴ記述子、またはＴＳＳ記述子であることが検出されると、プロセッサ１０はテーブル内の次に続くエントリを読出して記述子の残りを得る。したがって、呼出ゲート記述子、ＬＤＴ記述子、およびＴＳＳ記述子は、テーブルエントリのサイズを再定義することなしに、さらには１つのエントリを占有する記述子のためにテーブルはどのように管理されるのかを再定義することなしに、異なるサイズの互換モード記述子（または標準モード記述子）とともにテーブル内で共存し得る。さらに、ＴＳＳ記述子、ＬＤＴ記述子、および呼出ゲート記述子の第２の部分はセグメント記述子としてアクセスされ得るため、以下の図９で示されるように、記述子が記述子テーブルへと記憶されると、第２の部分内での記述子のタイプフィールドであり得る記述子の部分が無効タイプにセットされる。代替的には、記

述子テーブル読出が行なわれるたびに、プロセッサ１０は２つの連続するエントリを記述子テーブルから読出すこともでき、第１のエントリが呼出ゲート、ＬＤＴ記述子タイプ、またはＴＳＳ記述子タイプであると第２のエントリが用いられ得る。

【００５５】

いずれかの動作モード（３２／６４モード、３２ビット互換モード、または１６ビット互換モード）で動作しているコードは、ＬＭＥビットがセットされていると呼出ゲート記述子を参照し得ることが注目される。したがって、３２ビットまたは１６ビットアプリケーションは、たとえルーチンのアドレスが３２ビットまたは１６ビットアドレススペースの外にあったとしても、呼出ゲートメカニズムを用いてオペレーティングシステムルーチンを呼出すことができる。加えて、呼出ゲート記述子は、あらゆる動作モードを有するコードセグメントを参照することができる。オペレーティングシステムは、（３２ビットターゲットセグメントのために）呼出ゲート内のオフセットの最上位３２ビットが０であるか、または（１６ビットターゲットセグメントのために）呼出ゲート内のオフセットの最上位４８ビットが０であることを保証し得る。

【００５６】

図９を参照して、呼出ゲート記述子１２０のある実施例のブロック図が示される。他の実施例も可能であり、企図される。図２および図３と同様に、最下位バイトの上方に最上位バイトが示される。４バイトの各グループの最上位ビットがビット３１として示され、最下位ビットがビット０として示される。４バイト内の短い垂直の線は各々のビットを区切り、長い垂直の線もビットを区切るが、それはまたフィールドも区切る。上述のように、呼出ゲート記述子は記述子テーブル内で２つのエントリを占有する。図９の水平方向の破線によって、呼出ゲート記述子１２０は（線の上の）上部と（線の下の）下部とに分割される。下部は、呼出ゲートのセグメントセレクタによって索引付けされるエントリ内に記憶され、上部は次に続くエントリ内に記憶される。

【００５７】

呼出ゲート記述子１２０は、ターゲットセグメントセレクタ（フィールド１２２）、オフセット（フィールド１２４Ａ、１２４Ｂ、および１２４Ｃ）、現在の（Ｐ）ビット１２６、記述子特権レベル（ＤＰＬ）１２８、タイプフィールド１３０、および擬似タイプフィールド１３２を含む。Ｐビットは上述のＰビット４８と同様のものである。ターゲットセグメントセレクタは、（より大きな特権レベルを有する）ターゲットセグメント記述子が記憶される記述子テーブルのうちの１つのエントリを識別する。オフセットは、コード取出が開始されるアドレスを識別する。３２／６４モードでは、コードセグメントは基底アドレスを有さず、さらにはフラットリニアアドレス指定が用いられるため、オフセットはコード取出が始まるアドレスである。他のモードでは、オフセットはターゲットセグメント記述子が規定するセグメントベースに加えられてコード取出が始まるアドレスが生成される。上述のように、オフセットはこの実施例では６４ビットを含み得る。

【００５８】

ＤＰＬ１２８は、（現在の特権レベルおよび要求された(requested)特権レベルの両方において）呼出ルーチンが有さなければならない最小特権レベルを記憶し、これは呼出ゲートをうまく通り抜け、ターゲットセグメント記述子内で特定される特権レベルにおいて呼出されたルーチンを実行し得る。

【００５９】

タイプフィールド１３０は呼出ゲート記述子タイプにコーディングされる。ある実施例では、このタイプは、ｘ８６アーキテクチャ内で規定される３２ビット呼出ゲートタイプとしてコーディングされる。代わりに、他の符号化が用いられてもよい。最後に、擬似タイプフィールド１３２が無効タイプ（たとえば、０）にコーディングされて呼出ゲート記述子１２０の上半分を記憶するセグメントテーブルエントリを識別するセグメントセレクタが与えられると(presented)、プロセッサ１０によって例外が知らされることが保証される。

【００６０】

ＬＤＴ記述子９６の下半分は３２ビットＬＤＴ記述子と同様のものであり得ることと、ＬＤＴ記述子９６の上半分は呼出ゲート記述子１２０の上半分と同様のものであり得ることとが注目される。

【００６１】

次に図１０を参照して、プロセッサ１０が実行する命令のための命令フォーマット１４０のブロック図が示される。他の実施例も可能であり、企図される。図１０の実施例では、命令フォーマット１４０は、プレフィックスフィールド１４２、演算コードフィールド１４４、ｍｏｄ　Ｒ／Ｍ（レジスタ／メモリ）フィールド１４６、ＳＩＢ（スケールインデックスベース）フィールド１４８、変位フィールド１５０、および即値フィールド１５２を含む。演算コードフィールド１４４以外の各フィールドは任意である。したがって、命令フォーマット１４０によって、可変長命令が規定され得る。

【００６２】

プレフィックスフィールド１４２は、命令のためのいかなる命令プレフィックスのためにも用いられる。上述のように、オペランドサイズオーバーライドプレフィックスおよびアドレスサイズオーバーライドプレフィックスはある命令へと符号化されてプロセッサ１０の動作モードをオーバーライドし得る。これらのオーバーライドプレフィックスがプレフィックスフィールド１４２内に含まれる。上述のように、オペランドサイズオーバーライドプレフィックスおよびアドレスサイズオーバーライドプレフィックスの各々は、バイト単位で(by bytes)プレフィックスフィールド１４２内に含まれ得る。

【００６３】

演算コードフィールド１４４は、命令の演算コード（つまり、命令セット内のどの命令が実行されているのか）を含む。いくつかの命令では、オペランドは演算コードフィールド１４４内で特定され得る。他の命令では、演算コードの一部がｍｏｄ　Ｒ／Ｍフィールド１４６内に含まれ得る。さらに、ある特定の演算コードは、オペランドとして８ビットまたは１６ビットレジスタを特定する。したがって、演算コード符号化はまた、プロセッサ１０の動作モードによって示されるデフォルトをオーバーライドするように働き得る。

【００６４】

Ｍｏｄ　Ｒ／Ｍフィールド１４６およびＳＩＢフィールド１４８は、命令のオペランドを示す。変位フィールド１５０はいかなる変位情報をも含み、即値フィールド１５２は即値オペランドを含む。

【００６５】

コンピュータシステム

図１１を参照して、バスブリッジ２０２を通して種々のシステム構成要素に結合されるプロセッサ１０を含むコンピュータシステム２００のある実施例のブロック図が示される。他の実施例も可能であり、企図される。示されるシステムでは、メモリバス２０６を通してメインメモリ２０４がバスブリッジ２０２に結合され、ＡＧＰバス２１０を通してグラフィックスコントローラ２０８がバスブリッジ２０２に結合される。最後に、ＰＣＩバス２１４を通して複数のＰＣＩデバイス２１２Ａ－２１２Ｂがバスブリッジ２０２に結合される。ＥＩＳＡ／ＩＳＡバス２２０を通した１つ以上のＥＩＳＡまたはＩＳＡデバイス２１８への電気的インターフェイスに対応する(accomodate)ように、２次的なバスブリッジ２１６がさらに設けられ得る。プロセッサ１０は、ＣＰＵバス２２４を通してバスブリッジ２０２に結合され、任意のＬ２キャッシュ２２８にも結合される。ＣＰＵバス２２４とＬ２キャッシュ２２８に対するインターフェイスとは共に、外部インターフェイスユニット１８が結合し得る外部インタフェースを含み得る。

【００６６】

バスブリッジ２０２は、プロセッサ１０、メインメモリ２０４、グラフィックスコントローラ２０８、およびＰＣＩバス２１４に接続されるデバイスの間のインターフェイスを提供する。バスブリッジ２０２に接続されるデバイスのうちの１つからある動作が受取られると、バスブリッジ２０２は動作のターゲット（たとえば、ある特定のデバイスまたは、ＰＣＩバス２１４の場合には、ターゲットはＰＣＩバス２１４上にあること）を識別する

。バスブリッジ２０２は動作をターゲットのデバイスへと経路付ける。バスブリッジ２０２は一般に、ソースデバイスまたはバスが用いるプロトコルからターゲットデバイスまたはバスが用いるプロトコルへと動作を変換する。

【００６７】

ＩＳＡ／ＥＩＳＡバスに対するインターフェイスをＰＣＩバス２１４に提供することに加えて、所望ならば、２次的なバスブリッジ２１６はさらに付加的な機能を組込んでもよい。２次的なバスブリッジ２１６の外側にあるか、またはそれと一体化されている入力／出力コントローラ（図示せず）もコンピュータシステム２００内に含まれてもよく、所望ならば、キーボードおよびマウス２２２、ならびに種々のシリアルポートおよびパラレルポートに動作サポートを提供し得る。他の実施例では、外部キャッシュユニット（図示せず）がさらにプロセッサ１０とバスブリッジ２０２との間のＣＰＵバス２２４に結合され得る。代替的には、外部キャッシュはバスブリッジ２０２に結合されてもよく、外部キャッシュのためのキャッシュ制御ロジックはバスブリッジ２０２と一体化され得る。Ｌ２キャッシュ２２８がさらに、プロセッサ１０に対する後方構成で示される。Ｌ２キャッシュ２２８は、プロセッサ１０から分離されていてもよく、プロセッサ１０を備えるカートリッジ（たとえば、スロット１またはスロットＡ）と一体化されていてもよく、またはプロセッサ１０を備える半導体基板とさえ一体化されていてもよいことが注目される。

【００６８】

メインメモリ２０４は、アプリケーションプログラムが記憶され、さらにはそこからプロセッサ１０が主に実行されるメモリである。好適なメインメモリ２０４はＤＲＡＭ（ダイナミックランダムアクセスメモリ）を含む。たとえば、ＳＤＲＡＭ（同期ＤＲＡＭ）またはランバスＤＲＡＭ（ＲＤＲＡＭ）の複数のバンクも好適であり得る。

【００６９】

ＰＣＩデバイス２１２Ａ－２１２Ｂは、たとえば、ネットワークインターフェイスカード、ビデオアクセラレータ、オーディオカード、ハードディスクドライブまたはハードドライブコントローラ、フロッピー（Ｒ）ディスクドライブまたはフロッピー（Ｒ）ドライブコントローラ、ＳＣＳＩ（小型コンピュータシステムインターフェイス）アダプタ、および電話カード等の種々の周辺デバイスを例証する。同様に、ＩＳＡデバイス２１８は、モデム、サウンドカード、ならびにＧＰＩＢまたはフィールドバスインターフェイスカード等の種々のデータ収集カード等の種々の種類の周辺デバイスを例証する。

【００７０】

グラフィックスコントローラ２０８が設けられてディスプレイ２２６上のテキストおよび画像のレンダリングが制御される。グラフィックスコントローラ２０８は、メインメモリ２０４へと、さらにはそこから効果的にシフトされ得る三次元データ構造をレンダリングするような、当該技術分野で一般に公知の典型的なグラフィックスアクセラレータを具体化し得る。したがって、グラフィックスコントローラ２０８は、バスブリッジ２０２内のターゲットインターフェイスへのアクセスを要求し、かつそれを受取ってメインメモリ２０４へのアクセスを得ることができるという点で、ＡＧＰバス２１０のマスタであり得る。専用グラフィックスバスは、メインメモリ２０４からのデータの迅速な検索に対応する。ある特定の動作では、グラフィックスコントローラ２０８はさらに、ＡＧＰバス２１０上でＰＣＩプロトコルトランザクションを生成するように構成され得る。したがって、バスブリッジ２０２のＡＧＰインターフェイスは、ＡＧＰプロトコルトランザクションならびにＰＣＩプロトコルターゲットとイニシエータトランザクションとの両者をサポートするための機能を含む。ディスプレイ２２６は、画像またはテキストが示され得るすべての電子ディスプレイである。好適なディスプレイ２２６は、陰極線管（“ＣＲＴ”）、液晶ディスプレイ（“ＬＣＤ”）等を含む。

【００７１】

ＡＧＰ、ＰＣＩ、およびＩＳＡまたはＥＩＳＡバスが上の説明で例として用いられてきたが、所望であればいかなるバスアーキテクチャも代わりになり得ることが注目される。コンピュータシステム２００は、追加のプロセッサ（たとえば、コンピュータシステム２０

０の任意の構成要素として示されるプロセッサ１０ａ）を含む多重処理コンピュータシステムであってもよいことがさらに注目される。プロセッサ１０ａはプロセッサ１０と同様のものであり得る。より具体的には、プロセッサ１０ａはプロセッサ１０の同一コピーでもあり得る。プロセッサ１０ａは、（図１１で示されるように）独立したバスを介してバスブリッジ２０２に接続されてもよく、またはプロセッサ１０とともにＣＰＵバス２２４を共有してもよい。さらに、プロセッサ１０ａは、Ｌ２キャッシュ２２８と同様の任意のＬ２キャッシュ２２８ａに結合され得る。

【００７２】

図１２を参照して、コンピュータシステム３００の別の実施例が示される。他の実施例も可能であり、企図される。図１２の実施例では、コンピュータシステム３００は、いくつかの処理ノード３１２Ａ、３１２Ｂ、３１２Ｃ、および３１２Ｄを含む。各処理ノードは、それぞれの処理ノード３１２Ａ－３１２Ｄ内に含まれるメモリコントローラ３１６Ａ－３１６Ｄを介してそれぞれのメモリ３１４Ａ－３１４Ｄに結合される。加えて、処理ノード３１２Ａ－３１２Ｄは、処理ノード３１２Ａ－３１２Ｄ間の通信のために用いられるインターフェイスロジックを含む。たとえば、処理ノード３１２Ａは、処理ノード３１２Ｂと通信を行なうためのインターフェイスロジック３１８Ａ、処理ノード３１２Ｃと通信を行なうためのインターフェイスロジック３１８Ｂ、およびさらなる別の処理ノード（図示せず）と通信を行なうための第３のインターフェイスロジック３１８Ｃを含む。同様に、処理ノード３１２Ｂは、インターフェイスロジック３１８Ｄ、３１８Ｅ、および３１８Ｆを含み、処理ノード３１２Ｃは、インターフェイスロジック３１８Ｇ、３１８Ｈ、および３１８Ｉを含み、処理ノード３１２Ｄは、インターフェイスロジック３１８Ｊ、３１８Ｋ、および３１８Ｌを含む。処理ノード３１２Ｄは、インターフェイスロジック３１８Ｌを介して複数の入力／出力デバイス（たとえば、デイジーチェーン構成のデバイス３２０Ａ－３２０Ｂ）と通信するように結合される。他の処理ノードも同様の態様で他のＩ／Ｏデバイスと通信し得る。

【００７３】

処理ノード３１２Ａ－３１２Ｄは、処理ノード間の通信のためにパケットベースリンクを実現する。この実施例では、リンクは何組かの単方向ラインとして実現される（たとえば、ライン３２４Ａが用いられて処理ノード３１２Ａから処理ノード３１２Ｂへとパケットが伝送され、ライン３２４Ｂが用いられて処理ノード３１２Ｂから処理ノード３１２Ａへとパケットが伝送される）。図１２で示されるように、他の組のライン３２４Ｃ－３２４Ｈが用いられて他の処理ノード間でパケットが伝送される。一般に、ライン３２４の各々の組は、１つ以上のデータラインと、データラインに対応する１つの以上のクロックラインと、伝達されているパケットの種類を示す１つ以上のコントロールラインを含み得る。リンクは、処理ノード間の通信のためにキャッシュコヒーレント態様で動作され得るか、または処理ノードとＩ／Ｏデバイス（または、ＰＣＩバスまたはＩＳＡバス等の従来の構造のＩ／Ｏバスへのバスブリッジ）との間の通信のために非コヒーレント態様で動作され得る。さらに、リンクは、図示されるようなＩ／Ｏデバイス間のデイジーチェーン構造を用いて非コヒーレント態様で動作され得る。ある処理ノードから別のものへと伝送されるパケットは１つ以上の中間ノードを通り抜け得ることが注目される。たとえば、図１２で示されるように、処理ノード３１２Ａによって処理ノード３１２Ｄへと伝送されるパケットは、処理ノード３１２Ｂまたは処理ノード３１２Ｃのいずれかを通り抜け得る。好適なあらゆる配線アルゴリズムが用いられ得る。コンピュータシステム３００の他の実施例は、図１２で示される実施例よりもより多くの、またはより少ない処理ノードを含み得る。

【００７４】

一般に、パケットは、ノード間のライン３２４上の１つ以上のビットタイム(bit times)として伝送され得る。ビットタイムは、対応するクロックライン上のクロック信号の立上がり端縁または立下がり端縁であり得る。パケットは、トランザクションを開始するためのコマンドパケット、キャッシュコヒーレンシを維持するためのプローブパケット、およびプローブとコマンドとに応答するための応答パケットを含み得る。

【００７５】
メモリコントローラおよびインターフェイスロジックに加えて、処理ノード３１２Ａ－３１２Ｄは１つ以上のプロセッサを含み得る。概して、処理ノードは、少なくとも１つのプロセッサを含み、所望であればメモリと通信するためのメモリコントローラおよび他のロジックを任意で含み得る。より具体的には、各処理ノード３１２Ａ－３１２Ｄは、プロセッサ１０の１つ以上のコピーを含み得る。外部インターフェイスユニット１８は、ノード内にインターフェイスロジック３１８およびメモリコントローラ３１６を含み得る。
【００７６】
メモリ３１４Ａ－３１４Ｄは、好適なあらゆるメモリデバイスを含み得る。たとえば、メモリ３１４Ａ－３１４Ｄは、１つ以上のＲＡＭＢＵＳ　ＤＲＡＭ（ＲＤＲＡＭ）、同期ＤＲＡＭ（ＳＤＲＡＭ）、スタティックＲＡＭ等を含み得る。コンピュータシステム３００のアドレススペースはメモリ３１４Ａ－３１４Ｄ内で分割される。処理ノード３１２Ａ－３１２Ｄの各々は、どのアドレスがどのメモリ３１４Ａ－３１４Ｄにマッピングされ、したがってどの処理ノード３１２Ａ－３１２Ｄにある特定のアドレスのためのメモリ要求が経路付けされるべきなのかを判断するために用いられるメモリマップを含み得る。ある実施例では、コンピュータシステム３００内のアドレスのためのコヒーレンシポイントは、アドレスに対応するメモリ記憶バイトに結合されたメモリコントローラ３１６Ａ－３１６Ｄである。言換えると、メモリコントローラ３１６Ａ－３１６Ｄは、対応するメモリ３１４Ａ－３１４Ｄへのメモリアクセスの各々がキャッシュコヒーレント態様で起こることを保証することを担当する。メモリコントローラ３１６Ａ－３１６Ｄは、メモリ３１４Ａ－３１４Ｄとインターフェイスをとるための制御回路を含み得る。加えて、メモリコントローラ３１６Ａ－３１６Ｄは、メモリ要求を列に並ばせるための要求待ち行列を含み得る。
【００７７】
一般に、インターフェイスロジック３１８Ａ－３１８Ｌは、リンクからパケットを受信し、さらにはリンク上で伝送されるべきパケットをバッファリングするための種々のバッファを含み得る。コンピュータシステム３００は、パケットを伝送するための好適なあらゆるフロー制御メカニズムを採用し得る。たとえば、ある実施例では、各インターフェイスロジック３１８は、そのインターフェイスロジックが接続されるリンクの他端のレシーバ内に各々の種類のバッファの数のカウントを記憶する。インターフェイスロジックは、受信インターフェイスロジックがパケットを記憶するための解放されたバッファを有さない限りは、パケットを伝送しない。パケットを前方に(onward)経路付けることによって受信バッファが解放されると、受信インターフェイスロジックは送信インターフェイスロジックへとメッセージを伝送してバッファが解放されたことを示す。このようなメカニズムは「クーポンベース」システムと呼ばれ得る。
【００７８】
Ｉ／Ｏデバイス３２０Ａ－３２０Ｂは、好適なあらゆるＩ／Ｏデバイスであり得る。たとえば、Ｉ／Ｏデバイス３２０Ａ－３２０Ｂは、ネットワークインターフェイスカード、ビデオアクセラレータ、オーディオカード、ハードディスクドライブまたはハードドライブコントローラ、フロッピー（Ｒ）ディスクドライブまたはフロッピー（Ｒ）ドライブコントローラ、ＳＣＳＩ（小型コンピュータシステムインターフェイス）アダプタおよび電話カード、モデム、サウンドカード、およびＧＰＩＢまたはフィールドバスインターフェイスカード等の種々のデータ収集カードを含み得る。
【００７９】
上述の開示が完全に理解されると、多くの変更および変形が当業者に明らかとなるだろう。前掲の請求項はこのようなすべての変更および変形を包含すると解釈されることが意図される。
【図面の簡単な説明】
【図１】　プロセッサのある実施例のブロック図である。
【図２】　３２／６４モードのためのセグメント記述子のある実施例のブロック図である。

【図３】　互換モードのためのセグメント記述子のある実施例のブロック図である。
【図４】　図１で示されるプロセッサのある実施例に従った、互換モードおよびレガシーモードにおける動作を示すブロック図である。
【図５】　セグメント記述子および制御レジスタ値の関数としての動作モードの１つの実施例を例示する表である。
【図６】　デフォルト動作モードをオーバーライドするための命令プレフィックスの用途のある実施例を例示する表である。
【図７】　レジスタのある実施例を示すブロック図である。
【図８】　グローバル記述子テーブルおよびローカル記述子テーブルのある実施例を例示する図である。
【図９】　３２／６４呼出ゲート記述子のある実施例のブロック図である。
【図１０】　命令フォーマットのブロック図である。
【図１１】　図１で示されるプロセッサを含むコンピュータシステムのある実施例のブロック図である。
【図１２】　図１で示されるプロセッサを含むコンピュータシステムの別の実施例のブロック図である。

【図１】

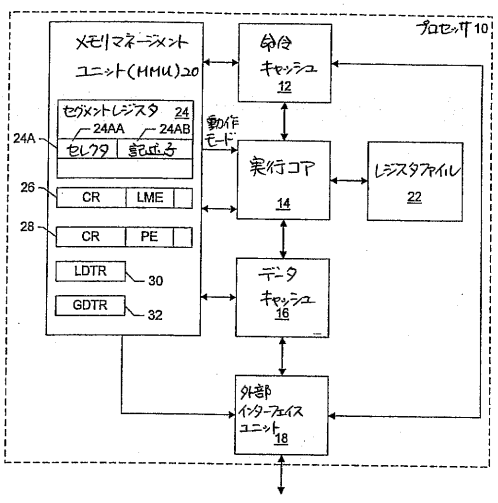

Fig. 1

【図２】

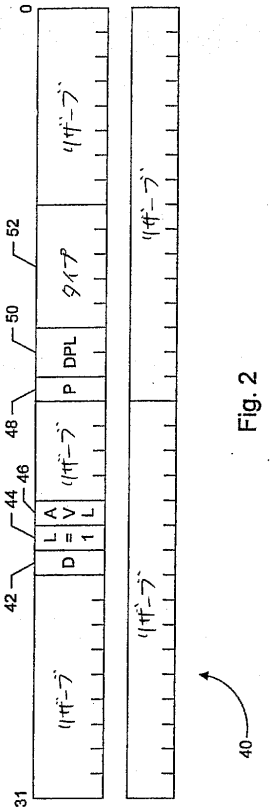

Fig. 2

【図３】

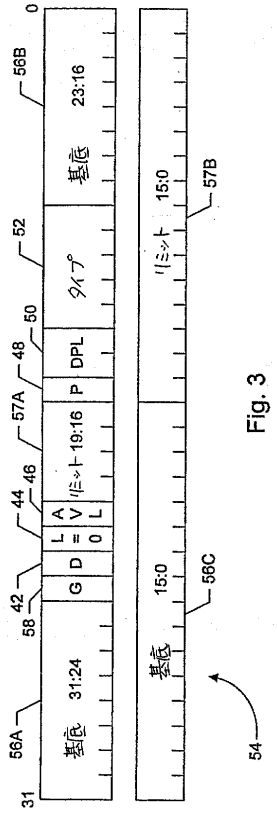

Fig. 3

【図４】

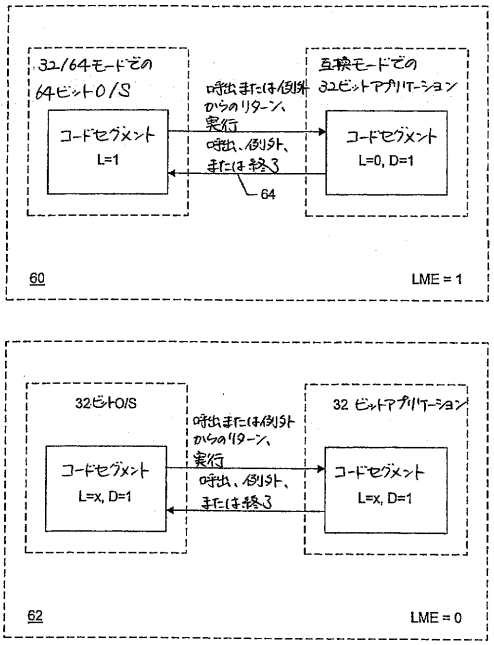

Fig. 4

【図５】

LME	CS Lビット	CS Dビット	動作モード
0	x	0	16 ビットモード
0	x	1	32 ビットモード
1	0	0	16 ビット互換モード
1	0	1	32 ビット互換モード
1	1	0	32/64 モード
1	1	1	リザーブ

70

Fig. 5

【図６】

オーバーライドプレフィックスの数(66H オペランド、67H アドレス)		0	1	2+
32/64 モード	オペランドサイズ	32 ビット	64 ビット	16 ビット
	アドレスサイズ	64 ビット	32 ビット	32 ビット
互換モードを含む 32ビットモード	オペランドサイズ	32 ビット	16 ビット	16 ビット
	アドレスサイズ	32 ビット	16 ビット	16 ビット
互換モードを含む 16ビットモード	オペランドサイズ	16 ビット	32 ビット	32 ビット
	アドレスサイズ	16 ビット	32 ビット	32 ビット

72

Fig. 6

【図７】

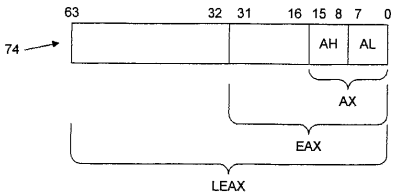

Fig. 7

【図８】

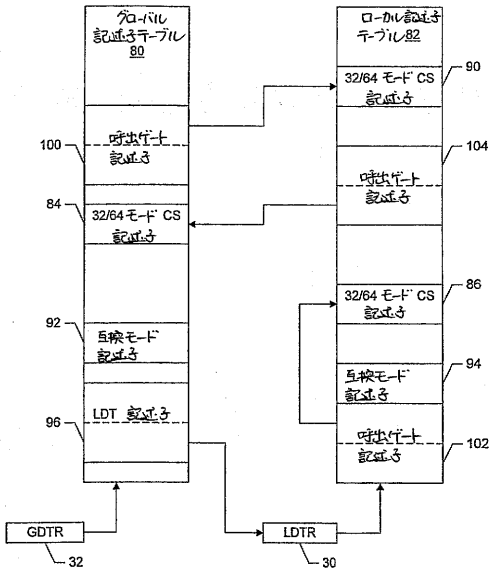

Fig. 8

【図９】

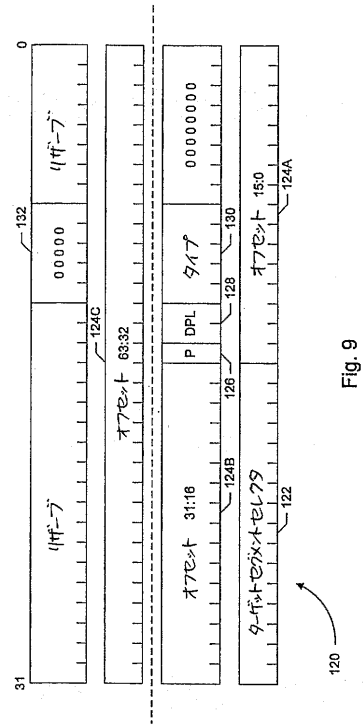

Fig. 9

【図１０】

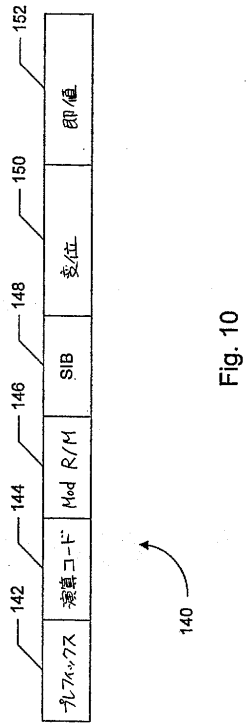

Fig. 10

【図１１】

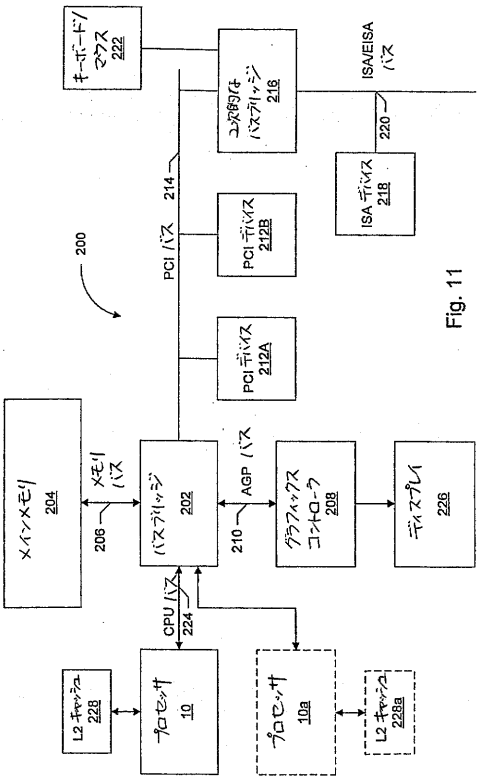

Fig. 11

【図１２】

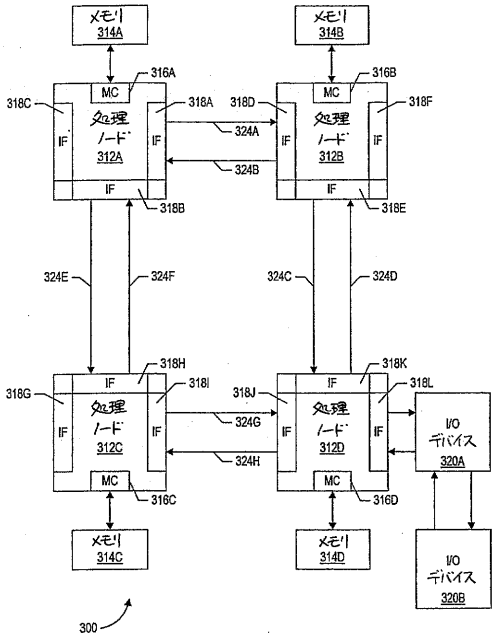

Fig. 12

フロントページの続き

(74)代理人 100083703
弁理士 仲村 義平
(74)代理人 100096781
弁理士 堀井 豊
(74)代理人 100098316
弁理士 野田 久登
(74)代理人 100109162
弁理士 酒井 將行
(72)発明者 マクグラス, ケビン・ジェイ
アメリカ合衆国、９５０３３ カリフォルニア州、ロス・ガトス、ハチンソン・ロード、２２８７６

審査官 三坂 敏夫

(56)参考文献 米国特許第０５４８１６８４（ＵＳ，Ａ）
特開平０６－２０２９４５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/34