(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3578682号
(P3578682)
(45)発行日　平成16年10月20日(2004.10.20)　(24)登録日　平成16年7月23日(2004.7.23)

(51)Int.Cl.[7]　F I
G06F 7/00　G06F 7/00　101W
G06F 9/305　G06F 9/30　340E

請求項の数 7　(全 21 頁)

(21)出願番号　特願平11-278471
(22)出願日　平成11年9月30日(1999.9.30)
(65)公開番号　特開2000-122847(P2000-122847A)
(43)公開日　平成12年4月28日(2000.4.28)
審査請求日　平成13年8月2日(2001.8.2)
(31)優先権主張番号　169483
(32)優先日　平成10年10月10日(1998.10.10)
(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　ジェローム・シー・ハック
アメリカ合衆国94303カリフォルニア州パロ・アルト、タリズマン・ドライブ　851

最終頁に続く

(54)【発明の名称】浮動小数点数処理記憶方法

(57)【特許請求の範囲】
【請求項１】
浮動小数点計算ユニット、メモリ、および浮動小数点レジスタを有するコンピュータ内で浮動小数点数を処理し記憶する方法であって、上記浮動小数点レジスタ内にレジスタ形式で浮動小数点数を保持するステップであって、それぞれの上記浮動小数点数は、符号、指数および仮数を含み、上記レジスタ形式の上記浮動小数点数はそれぞれ上記メモリ内で１ワードより多くのバイトを有する、ステップと、エンディアン制御ビットの第１状態に応じて、上記符号および上記指数が完全に第１メモリワードに含まれ、上記仮数が完全に第２メモリワードに含まれるよう、ビッグ・エンディアン形式で第１および第２の連続するメモリワードに浮動小数点数を記憶するステップと、エンディアン制御ビットの第２状態に応じて、上記仮数が完全に上記第１メモリワードに含まれ、上記符号および上記指数が完全に上記第２メモリワードに含まれるよう、リトル・エンディアン形式で上記第１および第２の連続するメモリワードに浮動小数点数を記憶するステップと、を含む、浮動小数点数を処理し記憶する方法。
【請求項２】
それぞれの上記メモリワードは、８バイトをもち、上記レジスタ形式は、８バイト仮数、２バイト指数、および１バイト符号を含む、請求項１に記載の浮動小数点数を処理し記憶する方法。
【請求項３】
上記浮動小数点数を記憶するステップは、上記レジスタ形式からの変換なく実行される、

請求項１に記載の浮動小数点数を処理し記憶する方法。
【請求項４】
上記ビッグ・エンディアン形式で浮動小数点数を記憶するステップは、上記指数および上記符号を含むメモリワードの最上位バイト位置にゼロを記憶することを含む、請求項１に記載の浮動小数点数を処理し記憶する方法。
【請求項５】
上記リトル・エンディアン形式で浮動小数点数を記憶するステップは、上記指数および上記符号を含むメモリワードの最下位バイト位置にゼロを記憶することを含む、請求項１に記載の浮動小数点数を処理し記憶する方法。
【請求項６】
上記ビッグ・エンディアン形式で浮動小数点数を記憶するステップは、上記第１および第２のメモリワードの連続するバイト位置に、上記符号、上記指数および上記仮数を記憶することを含む、請求項１に記載の浮動小数点数を処理し記憶する方法。
【請求項７】
上記リトル・エンディアン形式で浮動小数点数を記憶するステップは、上記第１および第２のメモリワードの連続するバイト位置に、上記符号、上記指数および上記仮数を記憶することを含む、請求項１に記載の浮動小数点数を処理し記憶する方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
この発明は、コンピュータ内の浮動小数点ユニットのアーキテクチャに関し、特に、浮動小数点プロセッサ内でバイエンディアン（ｂｉ－ｅｎｄｉａｎ）・ワードを処理し記憶する方法および装置に関する。
【０００２】
【従来の技術】
一般に、コンピュータ内で浮動算術演算を実施するための浮動小数点ユニットは、浮動小数点計算ユニットと、オペランド、中間結果および同類のものを保持するための１組の浮動小数点レジスタと、浮動小数点ステータスレジスタを含む。浮動小数点計算ユニットは、一般にパイプライン化されており、異なる浮動小数点計算の異なる演算を同時に実施することができる。浮動小数点ステータスレジスタは、精度制御および丸め制御のような浮動小数点計算を制御する制御情報を含むことができる。さらに、浮動小数点ステータスレジスタは、浮動小数点計算の間に発生した例外を記録するオーバフローおよびゼロ除算フラグのようなフラグ情報を含むことができる。ある実施例で、浮動小数点ユニットは、個々のステータスおよび制御情報を含むことができる。
【０００３】
浮動小数点ユニットは、典型的に、ＡＮＳＩ／ＩＥＥＥ浮動小数点標準第７５４－１９８５号に準拠するよう構成される。この標準は、浮動小数点データタイプ、多様な算術演算およびその他の演算、並びに例外の処理を指定している。すべての点でＩＥＥＥ浮動小数点標準を満たし、かつ、先行技術の欠点を克服し性能を向上させる追加の機能をもつ浮動小数点ユニットを提供することが望ましい。
【０００４】
従来の浮動小数点ステータスレジスタは、上述したように制御情報およびフラグ情報を含むハードウェア・レジスタである。制御情報は、ソフトウェアによって設定され、フラグ情報は、浮動小数点計算の実行に応じて設定される。従来、制御情報は、浮動小数点ステータスレジスタの内容を汎用レジスタにコピーし、汎用レジスタの内容を変更し、汎用レジスタの内容を浮動小数点ステータスレジスタに戻すことによって変更される。浮動小数点ステータスレジスタ内のフラグ情報も、同様の方法でクリアすることができる。従って、制御情報を更新しフラグ情報をクリアする演算は、比較的時間がかかる。その上、浮動小数点ステータスレジスタ内の制御情報が更新されるとき、パイプライン化された浮動小数点計算ユニットをフラッシュする必要があり、これにより部分的に完了された計算をア

ボートし性能を低下させることになる。同様に、読込み－変更－書込み演算が、フラグ情報のみをクリアし、制御情報に影響を与えない場合、フラッシュが行われて性能をさらに低下させることになる。これらの欠点のため、浮動小数点ステータスレジスタの頻繁な更新は一般に回避される。

【０００５】

投機的（推測的）な演算の実行は、プロセッサ性能を向上させるための既知の技法である。プロセッサを最大限に利用するため、実行に必要なオペランドが利用可能であれば、プログラムの中の後に現れる命令を前の命令と並行に実行するようにスケジュールすることができる。通常、プログラム内には分岐命令があるので、命令が実行を要求するかどうかを前もって判断できないことがある。しかし、プロセッサの資源が使用されていない状態であれば、たとえそれらの命令の実行が後で不要であると判断されるとしても、命令を投機的に実行することによってプロセッサの性能を改善することができる。投機的に実行された命令の実行をプログラムが最終的に必要とする場合、性能の改善が得られる。投機的な命令の実行が必要とされない場合、結果は捨てられる。

【０００６】

浮動小数点ステータスレジスタは、フラグビット形式のフラグ情報、つまり「フラグ」を含む。フラグは、浮動小数点計算の実行の間に生じる例外を記録する。さらに例外は、割込みを生成することがある。投機的な実行の場合、投機的な実行の結果は後で捨てられることがあるので、例外を直ちに報告することは望ましくない。それにもかかわらず、一般に、浮動小数点ユニットは、投機的な演算に関するフラグを投機的でない演算と同様に処理する。

【０００７】

フラグ情報内に記録される例外の１つとして、計算結果の中の指数（ｅｘｐｏｎｅｎｔ）が指定されたレンジを越えるオーバフロー例外がある。レンジは、浮動小数点数を記憶するために使用されるメモリ形式によって、または結果のユーザによって設定することができる。しかし、浮動小数点ユニットは、オーバフロー例外の報告を生じさせるような、レンジを越える浮動小数点数を処理することができる。これは、オーバフロー例外の不必要な報告という問題を生じる。例えば、浮動小数点計算は典型的にいくつかの演算を伴う。ある計算の中で、計算が続行することが許される場合、最終の結果がオーバフロー例外を生じさせなくても、中間演算の結果がオーバフロー例外を生じさせることがある。実行を遅らせたり終わらせることがあるので、例外の不必要な報告は避けることが望ましい。

【０００８】

浮動小数点計算中に浮動数を処理する別の側面は、「ビッグ・エンディアン（ｂｉｇ　ｅｎｄｉａｎ）」および「リトル・エンディアン（ｌｉｔｔｌｅ　ｅｎｄｉａｎ）」形式に関連する。ビッグ・エンディアン形式で、データワードは、その最上位バイトがメモリワードの最上位バイトに対応するようにメモリに記憶される。リトル・エンディアン形式では、データワードは、その最下位バイトがメモリワードの最上位バイトに対応するようにメモリに記憶される。プロセッサが両方の形式を効率的に処理することが要求されうる。

【０００９】

【発明が解決しようとする課題】

上記の欠点の１つまたは複数を軽減し、または排除する浮動小数点アーキテクチャを提供することが望ましい。

【００１０】

【課題を解決するための手段】

この発明の第１の側面によると、浮動小数点計算ユニット、メモリ、および浮動小数点レジスタを有するコンピュータにおいて浮動小数点数を処理し記憶する方法が提供される。浮動小数点数は、レジスタ形式の浮動小数点レジスタに保持され、それぞれの浮動小数点数は、符号（ｓｉｇｎ）、指数（ｅｘｐｏｎｅｎｔ）、および仮数（ｓｉｇｎｉｆｉｃａｎｄ：有効数字）を含む。レジスタ形式のそれぞれの浮動小数点数は、メモリ内の１ワードより多くのバイトをもつ。エンディアン制御ビットの第１状態に応じて、浮動小数点数

は、符号および指数が完全に第１メモリワードに含まれ、仮数が完全に第２メモリワードに含まれるよう、ビッグ・エンディアン形式で第１および第２の連続するメモリワードに記憶される。エンディアン制御ビットの第２状態に応じて、浮動小数点数は、仮数が完全に第１メモリワードに含まれ、符号および指数が完全に第２メモリワードに含まれるよう、リトル・エンディアン形式で第１および第２の連続するメモリワードに記憶される。浮動小数点数は、レジスタ形式からの変換なく記憶されることが好ましい。

【００１１】

ビッグ・エンディアン形式で浮動小数点数を記憶するステップは、指数および符号を含むメモリワードの最上位バイト位置にゼロを記憶することを含むことができる。リトル・エンディアン形式で浮動小数点数を記憶するステップは、指数および符号を含むメモリワードの最下位バイト位置にゼロを記憶することを含むことができる。

【００１２】

この発明の別の側面によると、浮動小数点数を処理し記憶する装置が提供される。この装置は、浮動小数点計算ユニット、メモリ、および浮動小数点レジスタを有するコンピュータを備える。さらに装置は、浮動小数点レジスタにレジスタ形式で浮動小数点数を保持する手段を含み、それぞれの浮動小数点数は、符号、指数および仮数を含む。レジスタ形式の浮動小数点数はそれぞれ、メモリ内に１ワードより多くのバイトをもつ。さらに装置は、エンディアン制御ビットの第１状態に応じて、符号および指数が完全に第１メモリワードに含まれ、仮数が完全に第２メモリワードに含まれるよう、ビッグ・エンディアン形式で第１および第２の連続するメモリワードに浮動小数点数を記憶する手段と、エンディアン制御ビットの第２状態に応じて、仮数が完全に第１メモリワードに含まれ、符号および指数が完全に第２メモリワードに含まれるよう、リトル・エンディアン形式で第１および第２の連続するメモリワードに浮動小数点数を記憶する手段を備えることができる。

【００１３】

【発明の実施の形態】

この発明をより良く理解するため、添付の図面が参照されこの中で取り入れられる。

【００１４】

図１に、浮動小数点ユニットおよび関連するプロセッサ要素の簡略ブロック図を示す。図１に示される要素は、中央処理装置（ＣＰＵ）の一部を構成する。命令シーケンサ１０は、実行されているプログラムの命令を処理し復号化する。命令は、メモリ１２内のオペランドのアドレスを含む。実行されている命令のタイプに依存して、メモリ１２からのオペランドは、汎用レジスタ（ＧＲ）１４または浮動小数点レジスタ（ＦＲ）１６に供給される。プロセッサは、典型的に、多くの浮動小数点レジスタを含む。浮動小数点オペランドは、浮動小数点計算を実施するため、浮動小数点レジスタ１６から浮動小数点計算ユニット２０および２２に供給される。浮動小数点計算の結果は、浮動小数点レジスタ１６に返される。プロセッサは、１つまたは複数の浮動小数点計算ユニットを含むことができる。浮動小数点ステータスレジスタ２４は、浮動小数点計算を制御する制御情報、および浮動小数点計算の間に生じた例外を記録するフラグ情報を含む。浮動小数点レジスタ１６、浮動小数点計算ユニット２０、２２、および浮動小数点ステータスレジスタ２４は、プロセッサの浮動小数点ユニットを構成する。

【００１５】

図２に、浮動小数点ステータスレジスタ２４の適当な形式の一例を示す。６４ビット・アーキテクチャを示している。浮動小数点ステータスレジスタ２４は、トラップ・フィールド４０、メイン・ステータスフィールド４２（ｓｆ０）、および少なくとも１つの代替ステータスフィールドを含む。図２の例で、浮動小数点ステータスレジスタ２４は、トラップ・フィールド４０、メイン・ステータスフィールド４２（ｓｆ０）、代替ステータスフィールド４４（ｓｆ１）、４６（ｓｆ２）および４８（ｓｆ３）、および予約ビット５０を含む。以下に述べるように、メイン・ステータスフィールド４２および代替ステータスフィールド４４、４６および４８は、異なる命令または命令グループに関連することができ、浮動小数点演算の制御および監視を向上させる。

【００１６】
一例として、トラップ・フィールド４０は、ビット位置０に無効な演算の浮動小数点例外フォールトを、ビット位置１にデノーマル（ｄｅｎｏｒｍａｌ）／非正規（ｕｎｎｏｒｍａｌ）オペランドの浮動小数点例外フォールトを、ビット位置２にゼロ除算浮動小数点例外フォールトを、ビット位置３にオーバフロー浮動小数点例外トラップを、ビット位置４にアンダーフロー浮動小数点例外トラップを、ビット位置５に不正確な浮動小数点例外トラップを含むことができる。それぞれのケースで、それぞれのビットが設定されているとき、フォールトまたはトラップは禁止（ディスエーブル）される。
【００１７】
図３に、それぞれのステータスフィールドｓｆ０、ｓｆ１、ｓｆ２およびｓｆ３の適当な形式の一例を示す。それぞれのステータスフィールドは、浮動小数点計算を制御するための制御情報、および浮動小数点計算の間に生じた状況を示すためのフラグ情報を含む。
【００１８】
図４に、適当なビット割当ての例を示す。制御ビットは、ゼロにフラッシュ・モード（ビット０）、最も広いレンジ指数（ビット１）、精度制御（ビット３：２）、丸め制御（ビット５：４）、およびトラップ禁止（ビット６）を含む。フラグビットは、無効な演算（ビット７）、デノーマル／非正規オペランド（ビット８）、ゼロ除算（ビット９）、オーバフロー（ビット１０）、アンダーフロー（ビット１１）および不正確（ビット１２）を含む。
【００１９】
代替ステータスフィールドの数、メインおよび代替ステータスフィールド内のビット数、ならびにステータスフィールド内のビット割当ては、この発明の範囲内で変更することができることが分かるであろう。さらに、ステータスレジスタは、より多くのまたはより少ない予約ビットを含むことができ、予約ビットは、任意の所望の方法で分布させることができる。
【００２０】
この発明の機能に従って、浮動小数点命令セットは、浮動小数点ステータスレジスタの指定されたフィールドを更新するための命令を含む。それぞれのステータスフィールド内の制御情報に関して、ｆｓｅｔｃ．ｓｆｘで示される浮動小数点制御設定命令を使用して、ステータスフィールドｘ（ｓｆｘ）内の制御ビットを更新する。ここで、ｘはステータスフィールドの番号を表す。より具体的に述べると、指定されたステータスフィールドの制御ビットは、メイン・ステータスフィールドｓｆ０の制御ビットを命令内のマスクフィールドと論理積演算し、メイン・ステータスフィールドｓｆ０の制御ビットを命令内の第２マスクフィールドと論理和演算することによって、初期化することができる。こうして、制御フィールド内の指定されたビットが設定され、またはクリアされ、メイン・ステータスフィールドｓｆ０からコピーを作ることができる。
【００２１】
さらに、浮動小数点命令セットは、ｆｃｌｒｆ．ｓｆｘで示される浮動小数点フラグクリア命令を含む。フラグクリア命令は、ステータスフィールドｘ内のフラグをクリアする。上述したように、ステータスフィールド内のフラグビットは、浮動小数点計算の間に生じる例外の結果として設定することができる。フラグビットは、それらがクリアされるまで設定されたままであることを意味する「スティッキービット」とみなされる。浮動小数点制御設定命令および浮動小数点フラグクリア命令は、浮動小数点ステータスレジスタの指定されたステータスフィールド内のビットが、他のステータスフィールドまたは同じステータスフィールドの他のサブセットに影響を及ぼすことなく設定されまたはクリアされることを可能にする。こうして、読み取り－変更－書き込み演算を必要とすることなく、また、浮動小数点パイプラインを中止したりフラッシュすることなく、浮動小数点ステータスレジスタを更新することができる。
【００２２】
浮動小数点制御設定命令および浮動小数点フラグクリア命令は、浮動小数点ステータスレ

ジスタの指定されたサブフィールドが設定され、またはクリアされることを可能にする命令として記述されうる。上で述べたように、指定されたステータスフィールドの制御ビットを設定し、またはクリアすることができ、指定されたステータスフィールドのフラグビットをクリアすることができる。より一般的には、指定されたステータスフィールドの１つもしくは複数のビット、または指定されたステータスフィールドのサブフィールドを設定し、またはクリアすることができる。以下に述べるように、このように浮動小数点ステータスレジスタを更新する能力は、性能の向上をもたらす。

【００２３】

図５および図６に関して、複数の浮動小数点ステータスフィールドの動作および利用を示す例を述べる。図５は、投機的な実行を利用しないプログラムセグメントの例を示す。図６および図７は、性能を向上させるため投機的な実行を利用する図５のプログラムセグメントの異なるバージョンを示す。図５ないし図７は、プログラムセグメントの簡略図であり、実際のプログラムセグメントは、コンピュータ制御のための追加の命令を含むことができることが分かるであろう。

【００２４】

図５で、分岐命令１０２が浮動小数点命令１００の後に続く。分岐命令１０２の結果に依存して、プログラムセグメントは命令１０６および１０８に進むことができ、または命令１０４に分岐することができる。

【００２５】

図５のプログラムセグメントを実行する際、プロセッサの性能を向上させるため投機的な実行を利用することが望ましい。命令が投機的に実行されるとき、命令は、それらがプログラム・シーケンス内に現れるよりも早く実行されるので、結果は、プログラム・シーケンス内の適切な箇所で利用できる。投機的な実行は、プロセッサ資源が利用可能であって他で使用されていない期間に有利である。また投機的な実行は、命令が実行に複数サイクルを必要とし、通常に実行される場合にプログラム実行の遅延をもたらすところでも有利である。１つの例はメモリアクセス命令である。投機的に実行される命令は、一般に分岐命令の後に続くので、それらは、プログラムの実行を完了するために最終的に要求されないことがある。例えば、命令１０６が投機的に実行され、分岐命令１０２が命令１０４に分岐することになる場合、命令１０６は要求されない。しかし、分岐命令１０２が、命令１０６を実行することになる場合、命令１０６の投機的な実行によって性能が高められる。図５で、命令１００、１０４、１０６および１０８はそれぞれ、浮動小数点ステータスレジスタ内のメイン・ステータスフィールドｓｆ０を利用する。

【００２６】

ここで図６を参照して、投機的な実行を利用するため、図５のプログラムセグメントが変更されている。図５および図６の中の同じ命令は同じ参照数字をもつ。図６に示すように、図５の命令１０４および１０６にそれぞれ対応する命令１０４Ａおよび１０６Ａは、分岐命令１０２の前に実行される。命令１０４Ａおよび１０６Ａは、分岐命令１０２の直前に実行されるように示されている。しかし、命令１０４Ａおよび１０６Ａは、これらの命令の実行に必要なデータが利用可能である分岐命令１０２の前のどの時間に実行されてもよいことが分かるであろう。なお、１つの命令または命令シーケンスを投機的に実行してもよい。

【００２７】

投機的でない命令である命令１００は、メイン・ステータスフィールドｓｆ０に割り当てられる。投機的な命令１０４Ａおよび１０６Ａは、浮動小数点ステータスレジスタ内の代替ステータスフィールドｓｆ２およびｓｆ３にそれぞれ割り当てられる。代替ステータスフィールドｓｆ２およびｓｆ３内のフラグビットが命令１０４Ａおよび１０６Ａの実行の前にクリアされることを保証するため、ｆｃｌｒｆ．ｓｆ２およびｆｃｌｒｆ．ｓｆ３で示されるフラグクリア命令１１０を使用して、代替ステータスフィールドｓｆ２およびｓｆ３内のフラグビットをクリアする。

【００２８】

それぞれの浮動小数点命令は、その命令によって使用されおよびそれに関連するステータスフィールドを指定するフィールドを含むことが好ましい。浮動小数点ステータスレジスタが、メイン・ステータスフィールドおよび３つの代替ステータスフィールドを含む場合、浮動小数点ポイント命令内の２ビット・フィールドが、ステータスフィールドを指定する。

【００２９】

図５および図６を参照して、図５の命令１０４が、図６のフラグ検査命令１２０と置き換えられていることが分かるであろう。さらに、図５の命令１０６は、図６のフラグ検査命令１２２と置き換えられている。ｆｃｈｋｆ．ｓｆ２で示されるフラグ検査命令１２０は、代替ステータスフィールドｓｆ２内のフラグビットを検査する。同様に、ｆｃｈｋｆ．ｓｆ３で示されるフラグ検査命令１２２は、代替ステータスフィールドｓｆ３内のフラグビットを検査する。

【００３０】

ステータスフィールド内に設定されるフラグビットはどれも、関連する命令または命令シーケンスの実行中に生じた例外を示す。例外は、フラグビット内に記録され、プログラム割込みを生じさせる潜在性をもっている。しかし、投機的な実行の場合、投機的な命令が実際に必要であると判断されるまで、メイン・ステータスフィールド内のフラグビットの設定またはプログラム割込みは行われるべきではない。投機的に実行された命令が捨てられる場合、割込みは、プロセッサ演算に不必要な遅延を与え、不適当にプログラムに割り込むことになる。以下に述べるように、代替ステータスフィールドは、投機的な実行の間の例外を記録し、投機的に実行された命令が確約される時間になるまで従来の例外の報告を遅らせる機構を提供する。

【００３１】

図２ないし図４に図示し上述した浮動小数点ステータスレジスタの構成は、投機的な実行の間に生じる割込みが遅延され、または完全に回避されることを可能にする。再び図３および図４を参照して、制御ビットは、トラップ禁止（ｔｒａｐ　ｄｉｓａｂｌｅ：ＴＤ）ビットを含む。ＴＤビットが、１つまたは複数の代替ステータスフィールド内に設定されているとき、そのステータスフィールドに関連する命令の実行の間の例外は、割込みを生じさせない。投機的に実行された命令が必要とされ、または確約されるとき、ｆｃｈｋｆ．ｓｆｘで示される浮動小数点フラグ検査命令が、代替ステータスフィールドｘ内のフラグビットを検査する。より具体的には、浮動小数点フラグ検査命令は、代替ステータスフィールドｘ内のフラグビットを、メイン・ステータスフィールドｓｆ０内のフラグビットおよび浮動小数点ステータスレジスタ内のトラップ許可（イネーブル）ビットと比較する。回復演算への分岐は、１）代替ステータスフィールドｓｆｘに設定されているフラグビットが、浮動小数点ステータスレジスタ内の許可されるトラップに対応する場合、または、２）代替ステータスフィールドｓｆｘに設定されているフラグビットが、メイン・ステータスフィールドｓｆ０内に設定されていない場合に生じる。

【００３２】

再び図６を参照して、命令１０４Ａおよび１０６Ａの投機的な実行の間、トラップ禁止ビットが、代替ステータスフィールドｓｆ２およびｓｆ３に設定される。代替ステータスフィールドｓｆ２およびｓｆ３内のフラグビットは、命令１０４Ａおよび１０６Ａの実行の間に生じる例外の結果として設定することができる。しかし、ＴＤビットが、プログラム割込みを防ぐ。プログラムが、命令１２０または命令１２２に進むとき、浮動小数点フラグ検査命令が実行される。詳しく述べると、分岐命令１０２は、投機的に実行された命令１０６Ａが確約されることを要求すると仮定する。プログラムは、分岐命令１０２からフラグ検査命令１２２に進む。命令１２２は、代替ステータスフィールドｓｆ３内のフラグビットを、メイン・ステータスフィールドｓｆ０内のフラグビットおよび浮動小数点ステータスレジスタ内のトラップ許可ビットと比較する。回復演算１２６への分岐は、１）代替ステータスフィールドｓｆ３に設定されているフラグビットが、浮動小数点ステータスレジスタ内の許可されるトラップに対応する場合、または、２）代替ステータスフィール

ドｓｆ３に設定されているフラグビットが、メイン・ステータスフィールドｓｆ０に設定されていない場合、に生じる。これらの条件のいずれも満たされない場合、回復演算１２６は不要であり、プログラムは直接命令１０８へ進む。

【００３３】

回復演算１２６は、投機的に実行された１つまたは複数の命令、この例では命令１０６Ａの再実行を伴う。再実行の間、再実行される命令は、メイン・ステータスフィールドｓｆ０に関連する。１つまたは複数のフラグビットが、メイン・ステータスフィールドｓｆ０に設定されうる。対応するトラップが許可される場合、割込みを行うことができる。こうして、投機的に実行された命令１０６Ａが確約されるまで、例外の報告は遅らされる。分岐命令１０２が命令１２０の実行を必要とする場合に、命令１０６Ａに関連する例外の報告が回避される。

【００３４】

分岐命令１０２が、投機的に実行された命令１０６Ａではなく命令１０４Ａが確約されることを要求する場合、プログラムは、上述したのと同様にフラグ検査命令１２０に進む。命令１２０は、代替ステータスフィールドｓｆ２内のフラグビットを、メイン・ステータスフィールドｓｆ０内のフラグビットおよび浮動小数点ステータスレジスタ内のトラップ許可ビットと比較する。回復演算１２４への分岐は、１）代替ステータスフィールドｓｆ２に設定されているフラグビットが、浮動小数点ステータスレジスタ内の許可されるトラップに対応する場合、または、２）代替ステータスフィールドｓｆ２に設定されているフラグビットが、メイン・ステータスフィールドｓｆ０に設定されていない場合、に生じる。これらの条件のいずれも満たされない場合、回復演算１２４は不要であり、プログラムは命令１０８へジャンプする。

【００３５】

回復演算１２４は、投機的に実行された１つまたは複数の命令、この例では命令１０４Ａの再実行を含む。再実行の間、再実行される命令は、メイン・ステータスフィールドｓｆ０に関連する。１つまたは複数のフラグビットが、メイン・ステータスフィールドｓｆ０に設定されうる。対応するトラップが許可される場合、割込みを行うことができる。

【００３６】

ＴＤビットは、メイン・ステータスフィールドｓｆ０内では使用されず、投機的でない命令は、典型的にメイン・ステータスフィールドｓｆ０に割り当てられる。メイン・ステータスフィールドｓｆ０に割り当てられた命令が実行されるとき、例外は直ちに報告される。

【００３７】

いくつかの例で、実行されている投機的な命令または命令シーケンスの数が、浮動小数点ステータスレジスタ内で利用可能な代替ステータスフィールドの数を越えることがある。その場合、２以上の投機的に実行される命令または命令シーケンスは、浮動小数点ステータスレジスタ内の１つの代替ステータスフィールドによって共有することができる。この状況で、１つまたは複数のフラグビットが、共有される代替ステータスフィールド内に設定されるとき、代替ステータスフィールドを共有する命令のうちのどれが１つまたは複数のフラグビットを設定させたかに関して曖昧さがある。この困難を克服するための技法を図７を参照して説明する。

【００３８】

図７で、投機的な実行を利用するため、図５のプログラムセグメントが再び変更されている。図５および図７の中の同じ命令は、同じ参照数字をもつ。図７に示すように、図５の命令１０４および１０６にそれぞれ対応する命令１０４Ｂおよび１０６Ｂは、分岐命令１０２の前に実行される。

【００３９】

投機的でない命令である命令１００は、メイン・ステータスフィールドｓｆ０に割り当てられる。投機的な命令１０４Ｂおよび１０６Ｂは、浮動小数点ステータスレジスタ内の代替ステータスフィールドｓｆ２に割り当てられ、それを共有する。命令１０４Ｂおよび１

０６Ｂの実行前に代替ステータスフィールドｓｆ２内のフラグビットがクリアされることを保証するため、ｆｃｌｒｆ．ｓｆ２で示されるフラグクリア命令１３０を使用して、代替ステータスフィールドｓｆ２内のフラグビットをクリアする。

【００４０】

さらに図５および図７を参照して、図５の命令１０４は、図７のｆｃｈｋｆ．ｓｆ２で示されるフラグ検査命令１３２と置き換えられる。さらに、図５の命令１０６は、図７のｆｃｈｋｆ．ｓｆ２で示されるフラグ検査命令１３４と置き換えられる。フラグ検査命令１３２および１３４はそれぞれ、代替ステータスフィールドｓｆ２内のフラグビットを検査する。

【００４１】

図７のプログラムセグメントの中で、命令１０４Ｂおよび１０６Ｂのどちらかまたは両方の実行の間の例外は、代替ステータスフィールドｓｆ２内のフラグビットを設定することができる。トラップ禁止ビットが代替ステータスフィールドｓｆ２に設定されるので、命令１０４Ｂおよび１０６Ｂが投機的に実行されるとき、例外は報告されない。

【００４２】

分岐命令１０２に続き、フラグ検査命令１３２またはフラグ検査命令１３４が実行される。フラグ検査命令１３２または１３４は、代替ステータスフィールドｓｆ２内のフラグビットを、メイン・ステータスフィールドｓｆ０内のフラグビットおよび浮動小数点ステータスレジスタ内のトラップ許可ビットと比較する。回復演算への分岐は、１）代替ステータスフィールドｓｆ２に設定されているフラグビットが、浮動小数点ステータスレジスタ内の許可されるトラップに対応する場合、または、２）代替ステータスフィールドｓｆ２に設定されているフラグビットが、メイン・ステータスフィールドｓｆ０に設定されていない場合に、生じる。詳しく述べると、フラグ検査命令１３２は回復演算１４０に分岐し、フラグ検査命令１３４は回復演算１４２に分岐する。上記の条件のどちらも満たされない場合、それぞれの回復演算は不要であり、プログラムは直接命令１０８に進む。

【００４３】

代替ステータスフィールドｓｆ２は、この例で命令１０４Ｂおよび１０６Ｂによって共有されるので、命令１０４Ｂおよび１０６Ｂのどちらが１つまたは複数のフラグビットを設定させたかを代替ステータスフィールドｓｆ２のみから判断することはできない。この情報を判断するため、回復演算１４０および１４２は次のように動作する。初めに、確約される投機的な命令または命令シーケンスが再実行され、メイン・ステータスフィールドｓｆ０が、再実行された命令に関連する。命令の再実行が確約されている間、１つまたは複数のフラグビットが、メイン・ステータスフィールドｓｆ０に設定されうる。さらに、対応するトラップが浮動小数点ステータスレジスタ内で許可されている場合、割込みが生じる。次に、共有された代替ステータスフィールドがクリアされる。それから、代替ステータスフィールドを共有するが、このとき確約されていない命令または命令シーケンスが再実行される。代替ステータスフィールドが、確約されていない命令の再実行に関連する。確約されていない命令の再実行の間に生じる例外は、代替ステータスフィールド内のフラグビットを設定させる。回復演算のこのシーケンスは、代替ステータスフィールドから確約された命令または命令シーケンスに対するフラグ情報を効果的に除去し、それをメイン・ステータスフィールドに移す。代替ステータスフィールドは、代替ステータスフィールドを共有するが確約されなかった残りの命令に対するフラグ情報を保持する。

【００４４】

再び図７の例を参照して、投機的に実行された命令１０６Ｂが確約されることを分岐命令１０２が要求すると仮定する。プログラムは、フラグ検査命令１３４に進む。フラグ検査命令１３４は、代替ステータスフィールドｓｆ２内のフラグビットを、メイン・ステータスフィールドｓｆ０内のフラグビットおよび浮動小数点ステータスレジスタ内のトラップ許可ビットと比較する。回復演算１４２への分岐は、１）代替ステータスフィールドｓｆ２に設定されているフラグビットが、浮動小数点ステータスレジスタ内の許可されるトラップに対応する場合、または、２）代替ステータスフィールドｓｆ２に設定されているフ

ラグビットが、メイン・ステータスフィールドｓｆ０に設定されていない場合、に生じる。これらの条件のどちらも満たされない場合、回復演算１４２は不要であり、プログラムは直接命令１０８に進む。

【００４５】

回復演算１４２は、投機的に実行された命令１０６Ｂの再実行を伴う。再実行の間、１つまたは複数のフラグビットが、メイン・ステータスフィールドｓｆ０に設定されうる。対応するトラップが許可される場合、割込みを生じさせることができる。代替ステータスフィールドｓｆ２内のフラグビットがクリアされ、投機的に実行された命令１０４Ｂが再実行される。命令１０４Ｂの再実行の間、１つまたは複数のフラグビットが、代替ステータスフィールドｓｆ２に設定されうるが、代替ステータスフィールドｓｆ２内のトラップ禁止ビットが割込みを防ぐ。ここで、代替ステータスフィールドｓｆ２は、プログラム実行の間の後の時間に確約されうる投機的に実行される命令１０４Ｂに対するフラグ情報のみを含む。

【００４６】

図６および図７のプログラムセグメントは、命令の再実行を生じさせ、それによって短期的に性能を低下させることに気付くであろう。しかし、プログラムの実行のインスタンスの大部分で、例外は生じない。それらのケースで、命令の再実行は必要とされず、全体の性能は投機的な実行によって高められる。

【００４７】

投機的な実行の場合、代替ステータスフィールドは、投機的な実行の間に生じる例外が、メイン・ステータスフィールドｓｆ０に影響を及ぼすことなく記録されることを可能にする。それぞれの代替ステータスフィールド内のトラップ禁止ビットは、不適当な割込みの発生を防ぐ。それぞれの命令が確約されるとき、投機的に実行された命令に関連するフラグ情報が利用される。

【００４８】

浮動小数点ステータスレジスタ内の複数のステータスフィールドの利用を投機的な実行に関連して上述した。しかし、浮動小数点ステータスレジスタの複数のステータスフィールド構成は、通常の投機的でない実行の間に有用である。例えば、浮動小数点ステータスレジスタの従来の読み取り－変更－書き込み更新によって必要とされる遅延をまねくことなく、プログラム・シーケンス内の異なる命令に対して異なる制御ビットを利用することが望ましい。したがって、異なる浮動小数点命令について異なる丸めおよび／または切捨てモードを利用することが望ましい。これは、異なる命令を浮動小数点ステータスレジスタ内の異なるステータスフィールドと関連させることによって達成することができる。こうして、異なるステータスフィールドに関連する命令は、異なる制御ビットをもつことができる。投機的でない実行の場合、トラップ禁止ビットはリセットされたままである。したがって、例外は直ちに報告される。

【００４９】

複数のステータスフィールドについて多数の他の用途がある。これらは、例えば区間演算計算の効率的な支援のため、あるステータスフィールドは切り上げであり他は切り捨てであるように、代替のコンピューティング環境を設定することを含む。別の用途は、１つのステータスフィールドを「フラグのためのビットバケツ」およびコードシーケンスのための標準計算環境として使用することである。この利用例は、ＤＩＶＩＤＥおよびＳＱＲＴコードシーケンスにあり、その中で、反復アルゴリズムに関する算術演算は、最も近いものへの丸め／ＷＲＥ＝１モードで行われる必要があり、中間計算からのフラグ設定はユーザによって見られるべきではない（または除算はＩＥＥＥ－７５４準拠でない）。この用途は、他のコンパイラ・ランタイム・シーケンスでも同様である。正弦および余弦ルーチンは、一般に、ユーザの浮動小数点環境を保存することから始まり、自己の浮動小数点環境を設定し、自己の計算を行い、ユーザの浮動小数点環境を回復する。複数のステータスフィールドがあれば、自己の浮動小数点環境を代替ステータスフィールド内に設定することができ、それによってユーザの元の浮動小数点環境を保存し回復する費用を回避するこ

とができる。

【００５０】

回転レジスタ（ｒｏｔａｔｉｎｇ　ｒｅｇｉｓｔｅｒｓ）は、実行に複数サイクルを必要とする命令をもつプログラム・ループの実行を速めるために利用されている。回転レジスタは、プログラム・ループの複数の繰り返しが並行に実行されることを可能にする。命令がレジスタを参照するとき、レジスタは、論理的レジスタとみなされ、対応する物理的レジスタは、ループのそれぞれの繰り返しごとにインクリメントされる。したがって、いくつかの繰り返しが、異なるレジスタの中で進行中になりうる。

【００５１】

浮動小数点ステータスレジスタ内の１つまたは複数の代替ステータスフィールドは、回転レジスタと関連することができる。回転ステータスフィールドは、異なる時間にプログラム・ループの異なる繰り返しと関連する。概して、浮動小数点ステータスレジスタは、固定される１つまたは複数のステータスフィールド、および回転する１つまたは複数のステータスフィールドを含むことができる。

【００５２】

図８を参照して、浮動小数点ステータスレジスタ２４を更新する技法の一例を述べる。浮動小数点計算ユニット２０は、ステージ２００、２０２、２０４、２０６および２０８を含むパイプライン・アーキテクチャをもつ。従来技術で知られているように、パイプライン化された計算ユニットの異なるステージは、同時に異なる演算を実施して動作速度を増加させる。１つの命令はいくつかの演算を必要とすることがあるので、計算ユニットの複数ステージによって同時に実施される演算は、１つの命令または２以上の命令に関連することができる。簡潔にするため、図８は、浮動小数点計算ユニットの各ステージにつき１つの命令を示す。さらに、図８に、メイン・ステータスフィールドｓｆ０および代替ステータスフィールドｓｆ１、ｓｆ２およびｓｆ３を含む浮動小数点ステータスレジスタ２４が示されている。先に述べたように、それぞれのステータスフィールドは、制御ビットＣおよびフラグビットＦを含む。

【００５３】

浮動小数点計算ユニット２０は、異なる浮動小数点命令がパイプライン化された計算ユニットの異なるステージで実行されているときの一時点で示されている。それぞれの浮動小数点命令は、ステータスフィールドｓｆ０、ｓｆ１、ｓｆ２およびｓｆ３の１つと関連し、またはそれを参照する。従って、例えばステージ２０４は、ｆｍｐｙで示される浮動小数点乗算命令を含み、この命令は、代替ステータスフィールドｓｆ３を参照する。浮動小数点計算ユニット２０のそれぞれのステージ２００、２０２、２０４、２０６および２０８は、参照されるステータスフィールドからの制御ビットのコピーを含む。制御ビットは、命令と共にパイプライン化された計算ユニットの中を伝搬し、その命令を制御するために使用される。これは、ステータスフィールドが動的に更新されることを可能にし、従ってプログラム・シーケンス内の異なる命令は、異なる制御情報を利用することができる。

【００５４】

ステータス情報に関する図８の装置の動作を以下に述べる。浮動小数点ステータスレジスタ２４は、浮動小数点制御設定ｆｓｅｔｃ．ｓｆｘ（ここで、ｘは参照されるステータスフィールドを示す）および浮動小数点フラグクリア（ｆｃｌｒｆ．ｓｆｘ）を含む浮動小数点命令によって更新される。浮動小数点制御設定およびフラグクリア命令は、浮動小数点プログラム・シーケンス内に組み込まれ、他の浮動小数点命令と共にパイプライン化された浮動小数点計算ユニットの中を伝搬する。上述したように、浮動小数点制御設定およびフラグクリア命令は、他のステータスフィールドに影響を及ぼすことなく、また、同じステータスフィールド内の他のサブフィールドに影響を及ぼすことなく、浮動小数点ステータスレジスタの指定されたサブフィールドが迅速に更新されることを可能にする。

【００５５】

図８の例で、フラグクリア命令２２０は、浮動小数点計算ユニット２０から出てきた。命令２２０は、メイン・ステータスフィールドｓｆ０内のフラグビットをリセットさせる。

この演算は、図示されるようにパイプライン化された計算ユニットの出力で実施することができる。命令２２０の後に続き、例えばステージ２００および２０８内の命令のようなメイン・ステータスフィールドｓｆ０を示すパイプライン内の命令は、それぞれの演算の間に生じる例外の結果として、メイン・ステータスフィールドｓｆ０に１つまたは複数のフラグビットを設定することができる。

【００５６】

ステージ２０８は、ステータスフィールドｓｆ０が制御ビット０１０１０を含むことを示す。ステージ２０２の制御設定命令は、ステータスフィールドｓｆ０の制御ビットを１１１１１に更新している。それゆえ、パイプライン内のステージ２０２から上流にあり、ステータスフィールドｓｆ０を参照するそれらの命令は、制御ビット１１１１１を利用する。対照的に、ステージ２０２の下流にあり、ステータスフィールドｓｆ０を参照するそれらの命令は、制御ビット０１０１０を使用する。このように、異なる命令は、同じステータスフィールドから異なる制御情報を利用することができることが分かるだろう。制御情報は、演算を遅延させることなく動的に更新することができる。

【００５７】

ｆｓｅｔｃ．ｓｆ２で示される制御設定命令２２４は、パイプライン化された計算ユニット２０の第１ステージ２００に入るところである。命令２２４は、ステータスフィールドｓｆ２内の制御ビットを、例えば０１００１に設定することができる。こうして、命令２２４の上流にあって、ステータスフィールドｓｆ２を参照するそれらの命令は、新しい制御ビットを利用する。

【００５８】

制御情報を必要とするそれぞれの浮動小数点命令は、それがパイプライン化された浮動小数点計算ユニットに入るとき、参照されるステータスフィールドから制御情報を読み取ることができる。参照されるステータスフィールドが、命令の完了前に更新されるとしても、その制御情報は、命令と共にパイプラインを通して運ばれ、その命令の実行を制御する。従って、例えば、ステータスフィールドｓｆ０を参照するステージ２０８の命令は、それがパイプライン化された計算ユニットに入るときに、ステータスフィールドｓｆ０から制御ビット０１０１０を取得した。ステージ２０８の命令がパイプラインから出る前にステータスフィールドｓｆ０内の制御ビットがステージ２０２の制御設定命令によって変更されるとしても、それらの制御ビットが命令の実行を制御する。この構成は、高い柔軟性を与え、ステータス情報を変更するための従来の技法に関連する遅延を避ける。

【００５９】

図８の構成は、制御設定命令がパイプライン化された計算ユニットの入力に達するとき、それは参照されるステータスフィールド内の制御ビットを更新し、フラグクリア命令がパイプライン化された計算ユニットから出るとき、それは参照されるステータスフィールド内のフラグビットをクリアするプロトコルを利用する。さらに、制御情報を必要とするそれぞれの浮動小数点命令は、それがパイプライン化された計算ユニットに入るとき、参照されるステータスフィールドを読み取る。この発明の範囲内で、異なるプロトコルを利用することができることが理解されるであろう。例えば、制御設定命令およびフラグクリア命令は、パイプライン化された計算ユニットに対して早いまたは遅い時間に浮動小数点ステータスレジスタを更新することができる。さらに、浮動小数点命令は、パイプライン化された計算ユニットに対して早いまたは遅い時間に、参照されるステータス情報を読み取ることができる。

【００６０】

ＩＥＥＥ標準に従うメモリ内の浮動小数点数の記憶装置に適した形式を図９に示す。単精度表現２５０は、８ビット指数２５２、および２３ビット仮数（ｍａｎｔｉｓｓａまたはｓｉｇｎｉｆｉｃａｎｄ）２５４を含む。２倍精度表現２６０は、１１ビット指数２６２および５２ビット仮数２６４を含む。拡張２倍表現２７０は、１５ビット指数２７２および６３ビット仮数２７４を含む。４倍精度表現２８０は、１５ビット指数２８２および１１２ビット仮数２８４を含む。それぞれの表現は、符号ビットを含む。これらの表現の中

で最も大きい指数は、１５ビットである。

【００６１】

浮動小数点レジスタ１６（図１）の中で使用されるレジスタ形式の一例を図１０に示す。レジスタ形式は、符号ビット２８８、１７ビットを有する指数フィールド２９０、６４ビットを有する仮数フィールド２９２を含む。このように、レジスタ形式の指数フィールド２９０は、メモリ表現のどの指数フィールドよりも大きい。レジスタ形式の指数フィールドは、メモリ形式で支援される最も広い指数フィールドより少なくとも２ビット広いことが好ましい。

【００６２】

一般に、従来技術の浮動小数点ユニットは、結果の指数が所与のレンジを越えるとき、オーバフロー例外を示すフラグビットがステータスレジスタに設定されるように構成される。レンジは、メモリ形式に基づいて選択される。従って、指数がメモリに記憶することができるレンジを越える場合、オーバフロー例外が発生する。同じレンジが、計算の任意の中間ステップおよび最終ステップに適用される。

【００６３】

多くの計算の中で、中間ステップは、メモリ形式の指数レンジを越える結果を生じるが、最終結果は、メモリ形式の指数レンジの範囲内にある。従って、浮動小数点レジスタを含む浮動小数点計算ユニットが、メモリ形式の指数レンジより広い指数レンジを処理する能力をもつところで、最終結果がメモリ形式の指数レンジの範囲内にある限り、中間ステップの間のオーバフロー例外の発生は不要である。

【００６４】

上記のタイプの浮動小数点計算の例を図１１に図示する。直角三角形の斜辺計算がブロック３００に示されている。斜辺Ｃは、辺ＡおよびＢの２乗の和の平方根として計算される。図示される例で、メモリ形式は、指数が９以下であり、ＡおよびＢの値が10^9に等しいことを要求する。ＡおよびＢの値は、中間ステップ３０２の斜辺の公式に代入される。追加の中間ステップ３０４、３０６および３０８は、ステップ３１０で最終結果を導く。指数は、ステップ３０４および３０６でメモリ形式のレンジを越えることに注意しなければならない。しかし、ステップ３１０の最終結果の指数は、メモリ形式のレンジの範囲内（９以下）である。浮動小数点計算ユニットは、中間ステップ３０２－３０８で生成されるレンジで指数を処理することができることを前提とすると、ステップ３１０の最終結果は、メモリ形式の指数レンジの範囲内にあるので、これらの中間ステップのオーバフロー例外の発生は不要である。詳しく述べると、最終結果のみがメモリ形式で記憶されればよい。

【００６５】

この発明の機能に従って、浮動小数点ステータスレジスタ２４（図１ないし図４）は、最も広いレンジ指数（ＷＲＥ）ビットを含む。最も広いレンジ指数ビットは、第１および第２の指数レンジと関連する。最も広いレンジ指数ビットのセット状態に対応しうる第１指数レンジは、浮動小数点計算ユニットの指数レンジ、典型的にはレジスタ形式の指数レンジに関連することができる。最も広いレンジ指数ビットのリセット状態に対応しうる第２指数レンジは、メモリ形式の指数レンジに関連することができる。ＷＲＥビットがセットされているとき、典型的に中間結果である結果が第１指数レンジに適合しない場合は例外が発生する。ＷＲＥビットがリセットされているとき、典型的に最終結果である結果が第２指数レンジに適合しない場合は例外が発生する。結果があらかじめ決められた条件を満たす場合、結果とそれぞれの指数レンジとの非適合性が確立される。例として、ＷＲＥビットがセットされているとき、結果の指数が第１指数レンジを越えていれば、オーバフロー例外が発生しうる。しかし、結果の指数が第１指数レンジに適合するが第２指数レンジを越える場合、オーバフロー例外は発生しない。ＷＲＥビットがリセットされているとき、結果の指数が第２指数レンジを越えていれば、オーバフロー例外が発生しうる。こうして、ＷＲＥビットがセットされているとき、対応する計算の指数レンジは、１または複数ビット増やすことができる。計算の中間ステップの間、指数がメモリ形式のレンジを越え

るとしても、計算の中間ステップは、オーバフロー例外を生じることなく進むことができる。再び図１１を参照して、ＷＲＥビットは、ステップ３０２、３０４、３０６および３０８でセットされ、最後のステップ３１０でリセットされる。こうして、計算は、オーバフロー例外の発生なく完了に向けて進み、最終結果の指数はメモリ形式に適合する。以下で述べるように、ＷＲＥビットを使用して、他のタイプの例外の報告を制御することができる。

【００６６】

ＷＲＥビットは、上述したように２以上の浮動小数点ステータスフィールドをもつアーキテクチャ、および１つの浮動小数点ステータスフィールドをもつアーキテクチャの中で使用することができる。浮動小数点ステータスレジスタが２またはそれ以上のステータスフィールドをもつとき、計算の中間ステップは、セットされたＷＲＥビットをもつ代替ステータスフィールドを参照し、計算の最終ステップは、リセットされたＷＲＥビットをもつメイン・ステータスフィールドｓｆ０を参照することができる。浮動小数点ステータスレジスタが１つのステータスフィールドをもつとき、ＷＲＥビットは、中間ステップの間は設定されており、計算の最終ステップの前にリセットされる。

【００６７】

ＷＲＥビットは、メモリ形式を使用してメモリに計算の最終結果を書き込むことに関連して記述された。より一般的に、ＷＲＥビットは、中間結果の指数が最終結果の指数より大きい必要がある応用に使用することができる。従って、例えば最終結果は、指定された形式で外部装置に伝送することができる。ＷＲＥビットのセットおよびリセット状態に関連する第１および第２指数レンジのサイズは、特定のアーキテクチャに依存する。上で述べたように、第１指数レンジは、第２指数レンジより少なくとも２ビット広いことが好ましい。

【００６８】

ＷＲＥビットは、オーバフロー例外を防ぐことに限定されない。ＷＲＥビットを使用して、例えばアンダーフロー例外、不正確例外、または、計算の中間ステップでは記録される必要がないが最終ステップで記録される必要がある任意の他の例外を防ぐことができる。アンダーフロー例外は、図１１の指数値を負に変更することによって説明することができる。ＷＲＥビットは、不要な例外の発生を防ぎ、それによって性能を向上させる。

【００６９】

浮動小数点ステータスレジスタ内の制御ビットとしてＷＲＥビットを説明した。別の実施例で、ＷＲＥビットは、命令形式で指定することができる。この方法の不利な点は、命令が特別なビットまたは特別な状態をもつ必要があることである。この方法の利点は、浮動小数点ステータスレジスタの更新が必要でないことである。どちらの場合も、ＷＲＥビットは、プログラマによって制御される。更に別の実施例で、指数レンジは、指数レンジのより高精度の制御を可能にする１ビットより大きい指数レンジで制御することができる。

【００７０】

この発明の更なる機能は、「ビッグ・エンディアン」（ｂｉｇ　ｅｎｄｉａｎ：ＢＥ）および「リトル・エンディアン」（ｌｉｔｔｌｅ　ｅｎｄｉａｎ：ＬＥ）形式を用いた演算に関連する。ビッグ・エンディアン形式では、データワードは、その最下位バイトがメモリワードの最上位バイトに対応するようメモリに記憶される。リトル・エンディアン形式で、データワードは、その最下位バイトがメモリワードの最下位バイトに対応するようメモリに記憶される。データは、両方の形式でメモリに記憶することができるが、それらの形式の１つだけがプロセッサのレジスタ内で許される。ビッグ・エンディアンまたはリトル・エンディアン形式は、制御レジスタ内のエンディアン制御ビットによって制御することができる。ビッグ・エンディアン／リトル・エンディアン制御ビットの状態は、ソフトウェアによって制御される。

【００７１】

上述したように、浮動小数点データは、図９に示すＩＥＥＥ形式のメモリに記憶することができ、ＩＥＥＥ形式とは異なるレジスタ形式で浮動小数点レジスタ内で処理することが

できる。いくつかのプロセッサ演算において、浮動小数点データは、レジスタ形式からＩＥＥＥ形式に変換され、メモリに記憶される。逆の場合も同様である。スピル（ｓｐｉｌｌ）／フィル（ｆｉｌｌ）演算として知られる他のプロセッサ演算では、浮動小数点データは、最初に浮動小数点数をＩＥＥＥ形式に変換することなく、レジスタ形式でメモリに書き込まれる。同様に、浮動小数点データがメモリから読み取られるとき、それはすでにレジスタ形式であり、形式変換は必要とされない。スピル／フィル演算は、例えばプログラムが割り込まれるときに実施することができる。

【００７２】

スピル／フィル演算を図１２および図１３に関して説明する。浮動小数点レジスタ形式を図１２にバイト表現で示す。詳しく述べると、レジスタ形式４００は、８バイトｓ０－ｓ７をもつ仮数（ｓｉｇｎｉｆｉｃａｎｄ：有効数字）、２バイトの指数ｅ０およびｅ１（４０４）、並びに１指数ビットおよび１符号ビットをもつ１バイトのｓｅ２（４０６）を含む。従って指数は、１７ビットをもつ。浮動小数点レジスタ形式４００は、合計８２ビットをもつ。

【００７３】

浮動小数点レジスタ形式の８２ビットは、図１３に示すように、ビッグ・エンディアンまたはリトル・エンディアン形式で１２８ビット・メモリフィールドに記憶することができる。図１３で、形式４２０は、リトル・エンディアン形式でレジスタ形式のスピル／フィル演算の結果を示す。形式４２４は、ビッグ・エンディアン形式のスピル／フィル演算の結果を示す。それぞれの形式の左側の数字は、増大するメモリアドレスを示す。リトル・エンディアン形式４２０では、仮数の最下位バイトｓ０が、１２８ビット・メモリフィールドの最も低いメモリアドレスに記憶される。ビッグ・エンディアン形式４２４では、仮数の最下位バイトｓ０が、１２８ビット・メモリフィールドの最も高いメモリアドレスに記憶される。それぞれのケースで、１２８ビットフィールドの中のゼロは、それぞれの浮動小数点数の最上位ビットである。図１３で、破線４３０、４３２および４３４は、８バイトワードの境界を示しており、第１および第２の連続するメモリワードを規定する。このように、リトル・エンディアン形式では、仮数は、ワード境界４３０と４３２の間の第１メモリワードに記憶され、指数および符号は、ワード境界４３２と４３４の間の第２メモリワードに記憶される。ビッグ・エンディアン形式では、指数および符号は、ワード境界４３０と４３２の間の第１メモリワードに記憶され、仮数は、ワード境界４３２と４３４の間の第２メモリワードに記憶される。

【００７４】

リトル・エンディアン形式４２０およびビッグ・エンディアン形式４２４は共に、仮数が８バイトアクセスでアクセスされることを可能にし、指数および符号が８バイトアクセスでアクセスされることを可能にする。すなわち、ＢＥおよびＬＥ形式のどちらの形式でも、指数および符号は一方の８バイトワードに閉じ込められ、仮数は他方の８バイトワードに閉じ込められるので、浮動小数点数のどちらかまたは両方の成分に別個にアクセスすることができる。さらに、図１３に図示し上述した形式４２０および４２４は、既存のデータに影響を及ぼすことなく、ゼロによって占有される位置の浮動小数点フィールドに制御ビットを加えることを許す。詳しく述べると、制御ビットおよび他の情報をＬＥ形式４２０のバイト１１－１５に加えることができ、そのような情報は、ＢＥ形式４２４のバイト０－４に加えることもできる。なお、個々の８バイトワードは、上述したそれらの固有の境界に合わせられることが好ましい。

【００７５】

この発明の好ましい実施例として現在考えられるものを図示し説明したが、当業者であればこの発明の範囲から逸脱することなく多様な変更および修正を行うことができることが分かるであろう。

【００７６】

本発明は例として次の実施態様を含む。

【００７７】

（１）浮動小数点計算ユニット、メモリ、および浮動小数点レジスタを有するコンピュータ内で浮動小数点数を処理し記憶する方法であって、
上記浮動小数点レジスタ内にレジスタ形式で浮動小数点数を保持するステップであって、それぞれの上記浮動小数点数は、符号、指数および仮数を含み、上記レジスタ形式の上記浮動小数点数はそれぞれ上記メモリ内で１ワードより多くのバイトを有する、ステップと、
エンディアン制御ビットの第１状態に応じて、上記符号および上記指数が完全に上記第１メモリワードに含まれ、上記仮数が完全に上記第２メモリワードに含まれるよう、ビッグ・エンディアン形式で第１および第２の連続するメモリワードに浮動小数点数を記憶するステップと、
エンディアン制御ビットの第２状態に応じて、上記仮数が完全に上記第１メモリワードに含まれ、上記符号および上記指数が完全に上記第２メモリワードに含まれるよう、リトル・エンディアン形式で上記第１および第２の連続するメモリワードに浮動小数点数を記憶するステップと、
を含む、浮動小数点数を処理し記憶する方法。

【００７８】

（２）それぞれの上記メモリワードは、８バイトをもち、上記レジスタ形式は、８バイト仮数、指数、および符号を含む、上記（１）に記載の浮動小数点数を処理し記憶する方法。

【００７９】

（３）上記浮動小数点数を記憶するステップは、上記レジスタ形式からの変換なく実行される、上記（１）に記載の浮動小数点数を処理し記憶する方法。

【００８０】

（４）上記ビッグ・エンディアン形式で浮動小数点数を記憶するステップは、上記指数および上記符号を含むメモリワードの最下位バイト位置にゼロを記憶することを含む、上記（１）に記載の浮動小数点数を処理し記憶する方法。

【００８１】

（５）上記リトル・エンディアン形式で浮動小数点数を記憶するステップは、上記指数および上記符号を含むメモリワードの最上位バイト位置にゼロを記憶することを含む、上記（１）に記載の浮動小数点数を処理し記憶する方法。

【００８２】

（６）上記ビッグ・エンディアン形式で浮動小数点数を記憶するステップは、上記第１および第２のメモリワードの連続するバイト位置に、上記符号、上記指数および上記仮数を記憶することを含む、上記（１）に記載の浮動小数点数を処理し記憶する方法。

【００８３】

（７）上記リトル・エンディアン形式で浮動小数点数を記憶するステップは、上記第１および第２のメモリワードの連続するバイト位置に、上記符号、上記指数および上記仮数を記憶することを含む、上記（１）に記載の浮動小数点数を処理し記憶する方法。

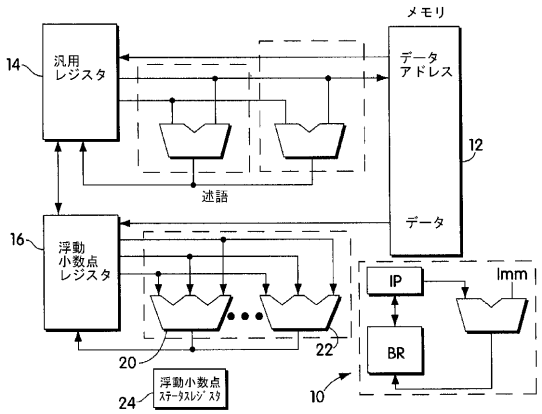
【００８４】

（８）浮動小数点計算ユニット、メモリ、および浮動小数点レジスタを有するコンピュータと、
上記浮動小数点レジスタ内にレジスタ形式で浮動小数点数を保持する手段であって、上記浮動小数点数のそれぞれが符号、指数および仮数を含み、上記レジスタ形式の上記浮動小数点数がそれぞれ上記メモリ内に１ワードより多くのバイトをもつ手段と、
エンディアン制御ビットの第１状態に応じて、上記符号および上記指数が完全に上記第１メモリワードに含まれ、上記仮数が完全に上記第２メモリワードに含まれるよう、ビッグ・エンディアン形式で第１および第２の連続するメモリワードに浮動小数点数を記憶する手段と、
エンディアン制御ビットの第２状態に応じて、上記仮数が完全に上記第１メモリワードに含まれ、上記符号および上記指数が完全に上記第２メモリワードに含まれるよう、リトル

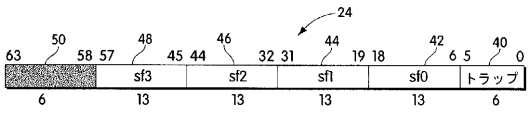
・エンディアン形式で上記第１および第２の連続するメモリワードに浮動小数点数を記憶する手段と、
を備える、浮動小数点数を処理し記憶する装置。
【００８５】
（９）上記メモリワードはぞれぞれ、８バイトをもち、上記レジスタ形式は、８バイト仮数、指数、および符号を含む、上記（８）に記載の浮動小数点数を処理し記憶する装置。
【００８６】
（１０）上記浮動小数点数は、上記レジスタ形式からの変換なく記憶される、上記（８）に記載の浮動小数点数を処理し記憶する装置。
【００８７】
（１１）上記ビッグ・エンディアン形式で浮動小数点数を記憶する手段は、上記指数および上記符号を含むメモリワードの最下位バイト位置にゼロを記憶する手段を含む、上記（８）に記載の浮動小数点数を処理し記憶する装置。
【００８８】
（１２）上記リトル・エンディアン形式で浮動小数点数を記憶する手段は、上記指数および上記符号を含むメモリワードの最上位バイト位置にゼロを記憶する手段を含む、上記（８）に記載の浮動小数点数を処理し記憶する装置。
【００８９】
（１３）上記ビッグ・エンディアン形式で浮動小数点数を記憶する手段は、上記第１および第２メモリワードの連続するバイト位置に、上記符号、上記指数、および上記仮数を記憶する手段を含む、上記（８）に記載の浮動小数点数を処理し記憶する装置。
【００９０】
（１４）上記リトル・エンディアン形式で浮動小数点数を記憶する手段は、上記第１および第２メモリワードの連続するバイト位置に、上記符号、上記指数、および上記仮数を記憶する手段を含む、上記（８）に記載の浮動小数点数を処理し記憶する装置。
【００９１】
【発明の効果】
本発明によれば、エンディアン制御ビットに応じて、レジスタ形式の浮動小数点数を、レジスタ形式からの変換なく両方のエンディアン形式でメモリに記憶することができる。
【図面の簡単な説明】
【図１】浮動小数点ユニットおよび関連するプロセッサ要素の簡略ブロック図。
【図２】浮動小数点ステータスレジスタの形式の一例を示す図。
【図３】それぞれの浮動小数点ステータスフィールドの形式の一例を示す図。
【図４】それぞれの浮動小数点ステータスフィールド内のビット割当ての一例を示す図。
【図５】投機的な演算を伴わないプログラムセグメントの一例を示すフローチャート。
【図６】第１の投機的実行方法に従う図５のプログラムセグメントの実行を示すフローチャート。
【図７】第２の投機的実行方法に従う図５のプログラムセグメントの実行を示すフローチャート。
【図８】それぞれのステージごとに制御ビットのコピーをもつパイプライン化された浮動小数点計算ユニットの簡略ブロック図。
【図９】異なる浮動小数点データタイプを記憶するために使用されるメモリ形式を示す図。
【図１０】ビット表現で浮動小数点レジスタ形式の一例を示す図。
【図１１】最も広いレンジの指数制御ビットがオーバフロー例外の報告を回避する浮動小数点計算の例を示すフローチャート。
【図１２】バイト表現で浮動小数点レジスタ形式の一例を示す図。
【図１３】スピルおよびフィル演算の間、ビッグ・エンディアンおよびリトル・エンディアン形式で浮動小数点数を記憶するためのメモリ形式を示す図。
【符号の説明】

１６　浮動小数点レジスタ
２０，２２　浮動小数点計算ユニット
２４　浮動小数点ステータスレジスタ
４２　メイン・ステータスレジスタ
４４，４６，４８　代替ステータスレジスタ
４００　レジスタ形式
４２０　リトルエンディアン形式
４４０　ビッグエンディアン形式

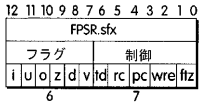
【図１】



【図２】



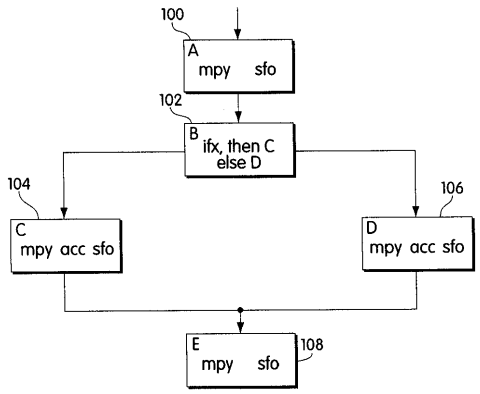
【図３】



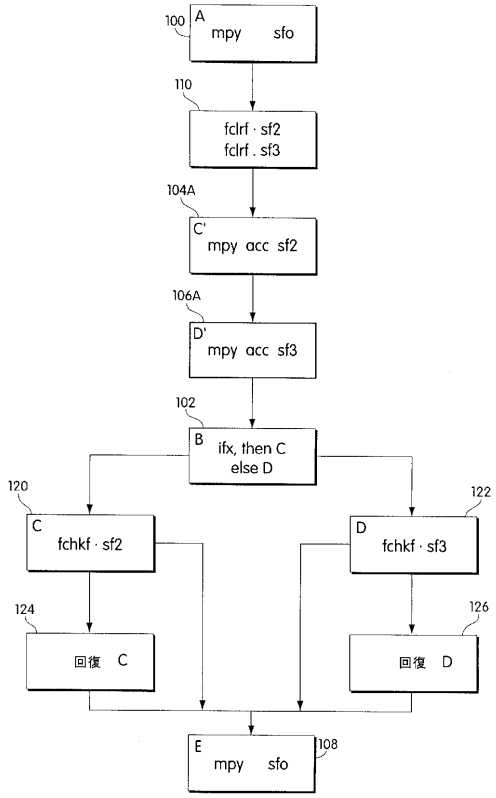
【図４】

フィールド	ビットレンジ	説明
ftz	0	ゼロへのフラッシュ・モード
wre	1	最も広い指数
pc	3:2	精度制御
rc	5:4	丸め制御
td	6	トラップ禁止
v	7	無効な演算(IEEE)
d	8	デノーマル/非正規オペランド
z	9	ゼロ除算(IEEE)
o	10	オーバフロー(IEEE)
u	11	アンダーフロー(IEEE)
i	12	不正確(IEEE)

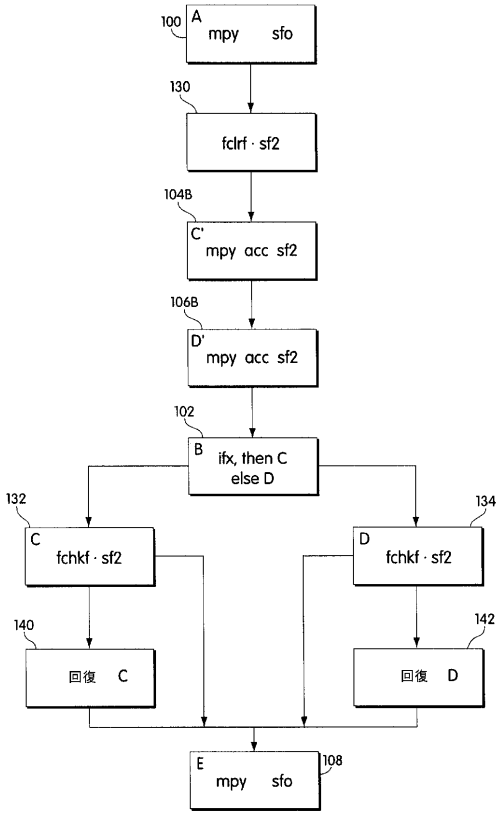
【図５】



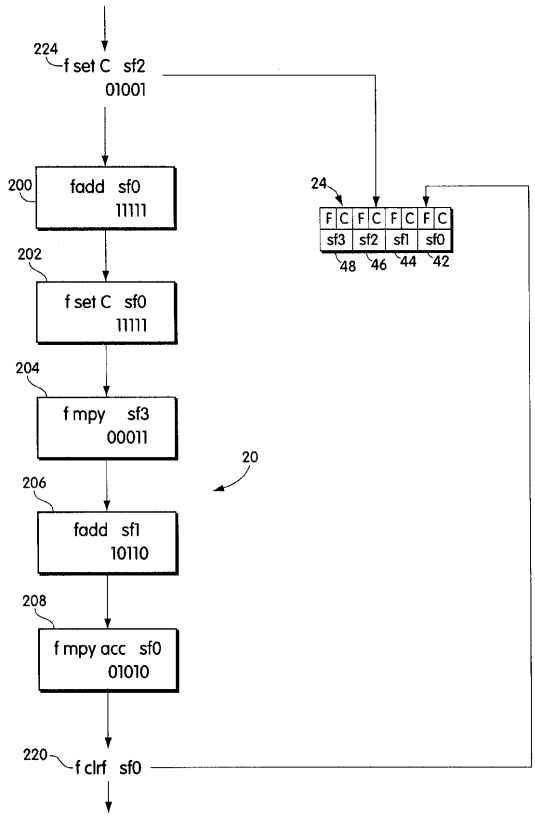
【図６】



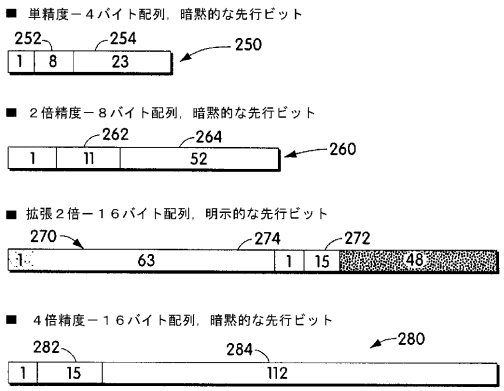
【図７】



【図８】



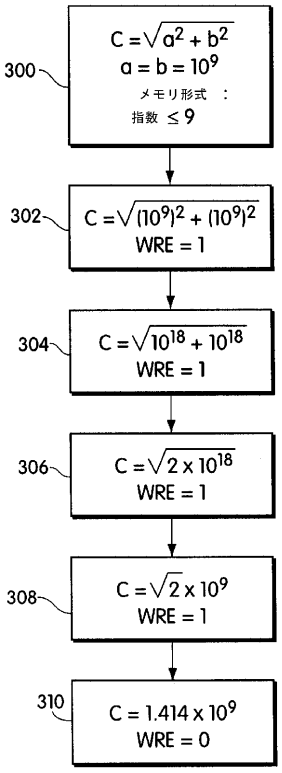
【図９】



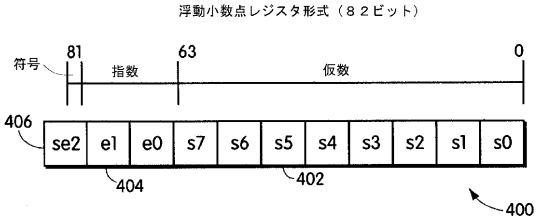
【図１０】



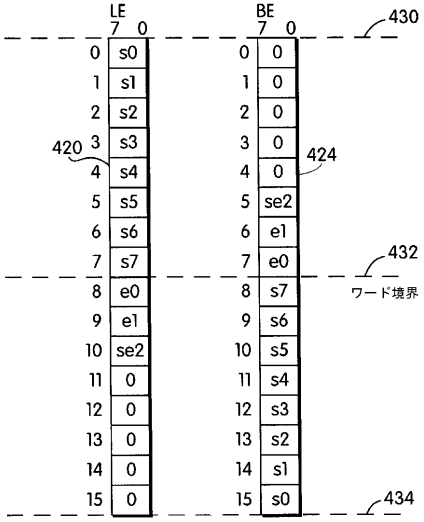
【図１１】


300
C = √(a² + b²)
a = b = 10⁹
メモリ形式 ：
指数 ≤ 9
302
C = √((10⁹)² + (10⁹)²)
WRE = 1
304
C = √(10¹⁸ + 10¹⁸)
WRE = 1
306
C = √(2 x 10¹⁸)
WRE = 1
308
C = √2 x 10⁹
WRE = 1
310
C = 1.414 x 10⁹
WRE = 0

【図１２】


浮動小数点レジスタ形式（８２ビット）
符号
81
指数
63
仮数
0
406
se2
e1
e0
s7
s6
s5
s4
s3
s2
s1
s0
404
402
400

【図１３】


LE
7 0
BE
7 0
430
420
424
432
ワード境界
434
0 s0 0 0
1 s1 1 0
2 s2 2 0
3 s3 3 0
4 s4 4 0
5 s5 5 se2
6 s6 6 e1
7 s7 7 e0
8 e0 8 s7
9 e1 9 s6
10 se2 10 s5
11 0 11 s4
12 0 12 s3
13 0 13 s2
14 0 14 s1
15 0 15 s0

フロントページの続き

(72)発明者　ピーター・マークスタイン
アメリカ合衆国９４０６２カリフォルニア州ウッドサイド、レッドランズ・ロード、１６０、エスアールエックス　９６
(72)発明者　グレン・ティー・コロンーボネット
アメリカ合衆国８０５２６コロラド州フォート・コリンズ、アントラーズ・コート　４３０４
(72)発明者　アラン・エイチ・カープ
アメリカ合衆国９４３０６カリフォルニア州パロ・アルト、イリマ・コート　８３７
(72)発明者　ロジャー・ゴリヴァー
アメリカ合衆国９７００８オレゴン州ビーヴァートン、サウスウェスト・ヴァイオレット・コート　１３３４０
(72)発明者　マイケル・モリソン
アメリカ合衆国９４０８６カリフォルニア州サニーヴェイル、コーンフラワー・コート　１０７２
(72)発明者　ゴータム・ビー・ドシ
アメリカ合衆国９４０８６カリフォルニア州サニーヴェイル、マデラ・アヴェニュー、４４２、ナンバー　１０

審査官　田中　友章

(56)参考文献　特開昭５９－２１４９４７（ＪＰ，Ａ）
米国特許第０５７２９７２４（ＵＳ，Ａ）
特開平０４－３６３７３６（ＪＰ，Ａ）
特開２０００－１２２８４９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F　7/00
G06F　9/305
G06F 12/04
G06F　5/00