(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3607701号
(P3607701)
(45)発行日　平成17年1月5日(2005.1.5)　(24)登録日　平成16年10月15日(2004.10.15)

(51)Int.Cl.[7]　F I
G06F　9/34　G06F　9/34　330

請求項の数 7　(全 24 頁)

(21)出願番号　特願平5-502403
(86)(22)出願日　平成4年7月8日(1992.7.8)
(65)公表番号　特表平6-501805
(43)公表日　平成6年2月24日(1994.2.24)
(86)国際出願番号　PCT/US1992/005720
(87)国際公開番号　WO1993/001543
(87)国際公開日　平成5年1月21日(1993.1.21)
審査請求日　平成11年7月6日(1999.7.6)
(31)優先権主張番号　726,773
(32)優先日　平成3年7月8日(1991.7.8)
(33)優先権主張国　米国(US)

(73)特許権者
トランスメタ　コーポレイション
アメリカ合衆国　カリフォルニア州　９５０５４　サンタ　クララ　フリーダム　サークル　３９４０
(74)代理人
弁理士　阿部　龍吉
(74)代理人
弁理士　蛭川　昌信
(74)代理人
弁理士　内田　亘彦
(74)代理人
弁理士　菅井　英雄
(74)代理人
弁理士　青木　健二

最終頁に続く

(54)【発明の名称】複数型レジスタ・セットを採用したＲＩＳＣマイクロプロセッサ・アーキテクチャ

(57)【特許請求の範囲】
【請求項１】
複数のモードで作動し、１ないし複数のオペランドに対して命令を実行して結果を生成するデータ処理システムに用いるデータ・レジスタ・ファイル装置であって、
複数のレジスタ・バンクとデータ処理システムのモードに対応して前記各レジスタ・バンクのアクセスの切替えを行う切替え手段とを備え、
前記各レジスタ・バンクは、
シャドウ・レジスタ・サブセットと第１整数レジスタ・サブセットと第２整数レジスタ・サブセットとを含むｎビットの整数レジスタ・セットと、
ｍビットのタイプ変更可能な浮動小数点レジスタ・セットと、
ブール・レジスタ・サブセットと１ビットのブールデータ・レジスタとを含むブール値を記憶するためのブール・レジスタ・セットと
によって構成され、
前記切替え手段は、前記モードに対応して、データ処理システムが第１モードで作動するときは前記第１整数レジスタ・サブセットと前記第２整数レジスタ・サブセットにアクセスを切替え、データ処理システムが第２モードで作動するときは前記第１整数レジスタ・サブセットと前記シャドウ・レジスタ・サブセットにアクセスを切替えることを特徴とするデータ・レジスタ・ファイル装置。
【請求項２】
前記第１整数レジスタ・サブセット、前記第２整数レジスタ・サブセット、前記タイプ変

更可能な浮動小数点レジスタ・セット、および前記ブール・レジスタ・サブセットは、それぞれオペランドまたは結果を記憶するための複数のデータ・レジスタを含み、前記複数のデータ・レジスタのそれぞれが同一構造であり、それぞれのセット内でオフセットとして個別にアドレス可能であることを特徴とする請求項１に記載のデータ・レジスタ・ファイル装置。

【請求項３】

前記モードに対応して、データ処理システムが前記第２モードに切替わるときは前記ブール・レジスタ・サブセットの前記複数のデータ・レジスタ内のデータを前記１ビットのブールデータ・レジスタに書き込み、データ処理システムが前記第１モードに切替わるときは前記１ビットのブールデータ・レジスタ内のデータを前記ブール・レジスタ・サブセットに戻すための転送手段を含むことを特徴とする請求項２に記載のデータ・レジスタ・ファイル装置。

【請求項４】

前記複数のデータ・レジスタのそれぞれが２個の書込みポートと５個の読出しポートを含むことを特徴とする請求項２に記載のデータ・レジスタ・ファイル装置。

【請求項５】

前記第１整数レジスタ・サブセット、前記ｍビットのタイプ変更可能な浮動小数点レジスタ・セット、および前記ブール・レジスタ・サブセットがそれぞれ、一定の結線論理値を有するデータ・レジスタを含むことを特徴とする請求項２に記載のデータ・レジスタ・ファイル装置。

【請求項６】

ｎ、ｍおよびｌがそれぞれ32、64、および32であることを特徴とする請求項１に記載のデータ・レジスタ・ファイル装置。

【請求項７】

前記モードに対応して、データ処理システムが前記第１モードで作動するときは前記ブール・レジスタ・サブセットにアクセスを切替え、データ処理システムが前記第２モードで作動するときは前記１ビット・ブールデータ・レジスタにアクセスを切替えるための第２の切替え手段を有することを特徴とする請求項１に記載のデータ・レジスタ・ファイル装置。

【発明の詳細な説明】

発明の背景

発明の分野

本発明は一般的にはマイクロプロセッサに関し、さらに具体的には、複数の対称レジスタ・セットを備えたRISCマイクロプロセッサに関する。

以下に列挙した米国特許出願は本件特許出願と同時に米国特許出願され、係属中のものであるが、これらの米国特許出願に開示されており、かつそれぞれに対応して出願された日本での特許出願に開示されている事項は、その出願番号を本明細書で引用することにより本明細書の一部を構成するものとする。

1.発明の名称「高性能RISCマイクロプロセッサ・アーキテクチャ」（High－Performance RISC Microprocessor Architecture）SC/SerialNo.07/727,006、1991年７月８日出願、発明者Le T.Nguyen他、およびこれに対応する特願平５－502150号（特表平６－501122号公報）。

2.発明の名称「拡張可能RISCマイクロプロセッサ・アーキテクチャ」（Extensible RISC Microprocessor Architecture）SC/SerialNo.07/727,058、1991年７月８日出願、発明者Le T.Nguyen他、およびこれに対応する特願平５－502153号（特表平６－501124号公報）。

3.「アーキテクチャ上の依存関係を隔離したRISCマイクロプロセッサ・アーキテクチャ」（RISC Microprocessor Architecture with Isolated Architectural Dependencies）SC/SerialNo.07/726,744、1991年７月８日出願、発明者Le T.Nguyen他、およびこれに対応する特願平５－502152号（特表平６－502034号公報）。

4.発明の名称「高速トラップと例外状態をインプリメントしたRISCマイクロプロセッサ・アーキテクチャ」（RISC Microprocessor Architecture Implementing Fast Trap and Exception State）SC/SerialNo.07/726,942、1991年７月８日出願、発明者Le T.Nguyen他、およびこれに対応する特願平５－502154号（特表平６－502035号公報）。
5.発明の名称「シングル・チップ・ページ・プリンタ・コントローラ」（Single Chip Page Printer Controller）SC/SerialNo.07/726.929、1991年７月８日出願、発明者Derek J.Lentz他、およびこれに対応する特願平５－502149号（特表平６－501586号公報）。
6.発明の名称「複数の異種プロセッサをサポートすることのできるマイクロプロセッサ・アーキテクチャ」（Microprocessor Architecture Capable of Supporting Multiple Heterogeneous Processors）SC/SerialNo.07/726,893、1991年７月８日出願、発明者Derek J.Lentz他、およびこれに対応する特願平５－502151号（特表平６－501123号公報）。
なお、本明細書の記述は本件出願の優先権の基礎たる米国特許出願07/726,773号の明細書の記載に基づくものであって、当該米国特許出願の番号を参照することによって当該米国特許出願の明細書の記載内容が本明細書の一部分を構成するものとする。
背景の説明
マイクロプロセッサをベースとするコンピュータ・システムは、主メモリ記憶装置および補助永続記憶装置の通常の補強に加えて、１つまたは２つ以上の汎用データ・レジスタ・１つまたは２つ以上のアドレス・レジスタ、および１つまたは２つ以上のステータス（状況）フラグをも備えているのが代表的である。従来のシステムには、整数データを格納しておくための整数レジスタと浮動小数点データを格納しておくための浮動小数点レジスタを備えているものもある。ステータス・フラグは、最近に実行されたオペレーションの結果としてのある種の条件を示すために使用されているのが代表的である。一般的には、ステータス・フラグとしては、前のオペレーションにおいて、キャリー（桁上げ）が生じたか否か、負数が生じたか否か、および／またはゼロが生じたか否かを示すものがある。
これらのフラグは、プログラム制御のフロー（流れ）内の条件付きブランチ（分岐）の結果を判定する際に役立っている。例えば、１番目の数を２番目の数と比較し、これらの２つの数が等しいことを条件として、あるサブルーチンへブランチしたい場合には、マイクロプロセッサは他方から一方を減算し、該当する条件フラグをセットまたはクリアすることによって２つの数を比較することができる。減算の結果の数値はストアしておく必要はない。そのあと、ゼロ・フラグのステータスを条件として、条件付きブランチ命令を実行させることができる。この方式は簡単に実現できるが、柔軟性と威力に欠けている。いったん比較が行われると、該当のフラグに基づく条件付きブランチを行う前に、それ以後の数値その他のオペレーションを行うことができない。もし行うと、比較の結果得た条件フラグ値を途中に置かれた命令が重ね書きするので、正しくないブランチが行われることになる。この方式は、上に示した単純な等価比較例とは異なり、もっと複雑なブランチ・テストを行うことが望ましい場合には、さらに複雑になる。
例えば、１番目の数が２番目の数より大で、３番目の数が４番目の数より小で、５番目の数が６番目の数と等しい、という条件が満たされたときだけ、プログラムをサブルーチンにブランチさせる場合を考えてみる。この場合、従来のマイクロプロセッサでは、条件付きブランチが途中に大量に配置された長い比較の列を実行する必要がある。比較とブランチがシリアルになったこの方式の特に望ましくない特徴は、命令がパイプライン化したどのマイクロプロセッサにも観察されている。
パイプライン方式マイクロプロセッサでは、どの時点においても２つ以上の命令が実行されており、複数の命令がどの瞬時においても異なる実行ステージに置かれている。これにより、スループットが大幅に向上している。代表的なパイプライン方式マイクロプロセッサでは、パイプライン・ステージは、（ａ）命令のフェッチ、（ｂ）命令のデコード、（ｃ）命令のオペランドの取得、（ｄ）命令の実行、（ｅ）結果のストアからなっている。
問題が起こるのは、条件付きブランチ命令がフェッチされるときである。そのようなケー

スとして、オペランドをまだパイプラインに残っているオペレーションから得るものとする場合に、オペランドがまだ計算されていないので、条件付きブランチの条件がまだテストできない場合がある。この結果、「パイプライン停止」（pipeline stall）が起こり、プロセッサの速度が大幅に低下することになる。

従来のマイクロプロセッサ・ベースのシステムのもう１つの欠点は、いかなるデータ・タイプ（データ型）であっても、レジスタ・セットが１つしか備わっていないことである。従来のアーキテクチャでは、どのデータ・タイプにおいても、レジスタの個数を増やすことが望ましいために、どのタイプのレジスタ・セットの場合であっても、そのセットを大きくすることだけが解決策であった。この結果、アドレス指定上の問題、アクセス衝突という問題、および対称上の問題が起こり得る。

同じように注目すべきことは、従来のアーキテクチャでは、どのレジスタ・セットの場合も、数値データ・タイプが１つに限定されていることである。種々の従来システムでは、汎用レジスタに格納できるのは、数値データかアドレス「データ」のどちらかである。しかし、本明細書で用いている「データ」の用語にはアドレスは含まれていない。そのようにした意図を理解しやすくするために、２つの従来システムを参照して説明する。インテル8085マイクロプロセッサは、数値データの２バイトか１個の２バイト・アドレスのどちらかを格納するために使用できるレジスタ・ペア“HL"を備えている。本発明による改善はこの問題を対象にしていない。もう１つはインテル80486マイクロプロセッサであり、これは整数データ型の汎用レジスタ・セットと浮動小数点レジスタ・セットを備えているが、各セットはそれぞれのデータ・タイプが限定されており、少なくとも算術論理演算ユニットが直接にレジスタを使用することを目的としている。

このことは、マイクロプロセッサが両方のデータ・タイプと係わりがないオペレーション（演算）を実行するときは、使用可能なチップ・エリアなどの、マイクロプロセッサの資源を無駄に消費することになる。例えば、ユーザ・アプリケーションは専ら整数型オペレーションと係わりがあることが多く、浮動少数点型オペレーションを実行することはまったくない。この種のユーザ・アプリケーションが浮動少数点レジスタを備えた従来のマイクロプロセッサ（80486など）で実行されるときは、これらの浮動小数点レジスタはその全実行期間中遊んだままになっている。

従来のマイクロプロセッサのレジスタ・セット・アーキテクチャのもう１つの問題は、ユーザ・アプリケーションとオペレーティング・システム・カーネルのようにアクセス特権レベルが高いエンティティとの間でコンテキスト・スイッチングまたはステート・スイッチングを行うときに観察されている。マイクロプロセッサ内の制御がコンテキスト、モード、またはステートをスイッチするとき、制御が渡されたオペレーティング・システム・カーネル（kernel）または他のエンティティは、ユーザ・アプリケーションがオペレーションに使用したのと同じデータについてオペレーションしないのが普通である。従って、データ・レジスタには、制御を受け取った新しいエンティティには無用なデータ値が格納されるが、その値はユーザ・アプリケーションに制御が返されるまで残しておかなければならないのが普通である。カーネルは、一般的には、独自に使用するレジスタをもっていなければならないが、どのレジスタが現在ユーザ・アプリケーションによって使用中であるかをカーネルに知らせる方法がない。自身のデータ用にスペースを確保するためには、カーネルはスワップ・アウトするか、さもなければ、レジスタのあらかじめ定めたサブセットの内容をストアしなければならない。この結果、特に、カーネルが制御を繰返し要求し、制御を保持している期間が短時間の場合には、オーバヘッドとして負担すべき処理時間の損失が膨大なものとなる。

上記に関連して注目すべきことは、従来のマイクロプロセッサでは、「大規模な」コンテキスト・スイッチを行う必要があるとき、マイクロプロセッサが一般的に大多数の処理サイクルを含めて、さらに大量の処理資源を拡張し、全データとステート情報をセーブしてからスイッチを行う必要があったことである。コンテキストをスイッチ・バックするときは、システムを以前の状態に復元するために、パフォーマンスを犠牲にするという代価を払わなければならなかった。例えば、マイクロプロセッサが２つのユーザ・アプリケーシ

ョンを実行する場合、各アプリケーションが各データ・タイプのレジスタの完全な補強を必要とし、しかも条件設定オペレーションや数値計算の種々のステージに置かれていると、一方のユーザ・アプリケーションから他方へスイッチするたびに、スワッピングを行うか、さもなければシステム内のすべてのデータ・レジスタとステート・フラグの内容をセーブしておく必要がある。この結果、オペレーションに伴うオーバヘッドが大量に発生することは明らかであり、特に、レジスタをセーブしておく必要のある主記憶装置や補助記憶装置がマイクロプロセッサ自体よりも著しく低速の場合には、大幅な性能低下をもたらすことになる。

以上の事実から明らかになったことは、複合条件（complex condition）を構成する各種条件を途中に条件付きブランチを介在させないで計算することを可能にする、改良型マイクロプロセッサ・アーキテクチャを開発することが望ましいことである。さらに、明らかになったことは、多数の単純な条件を並列に計算できるようにすることが望ましく、そうすれば、マイクロプロセッサのスループットが向上することである。

また、明らかになったことは、どのデータ・タイプの場合も、レジスタ・セットを複数にすることが望ましいことである。

さらに、望ましいことは、使用可能な整数レジスタが必要とする整数データ量を最適に格納するのに不十分であった場合に、マイクロプロセッサの浮動小数点レジスタを整数レジスタとして使用できるようにすることである。特に、明らかになったことは、このようなタイプ変更をユーザ・アプリケーションには完全に見えない（transparent）ようにすることが望ましいことである。

また、非常に望ましいことは、ユーザ・レジスタのサブセットを使用するのではなく、カーネルが使用するために予約した専用レジスタ・サブセットを備えたマイクロプロセッサを開発し、この新しいレジスタ・セットを、これらのレジスタによって代用されたレジスタ・サブセットと全く同じようにアドレス指定できるようにすることであり、そうすれば、カーネルがユーザ・アプリケーションと同じレジスタ・アドレス指定方式を使用できることである。さらに明らかになったことは、マイクロプロセッサの資源を最大限に利用するためには、２つのレジスタ・サブセット間の切替えをマイクロプロセッサのオーバヘッドとなるサイクルを必要としないで行うことが望ましい。

もう１つ明らかになったことは、「大規模な」コンテキスト・スイッチを最小のオーバヘッドで行えるようにするマイクロプロセッサ・アーキテクチャにすることが望ましいことである。これに関連して望ましいことは、各タイプのレジスタ・セットのバンクを複数にすることを可能にするアーキテクチャにすることであり、そうすれば、２つまたはそれ以上のユーザ・アプリケーションをマルチタスキング環境で、あるいはその他の「同時実行」モードで、稼働させることができ、各ユーザ・アプリケーションは少なくともレジスタの１バンク全体に独占的にアクセスできることである。また、明らかになったことは、レジスタ・アドレス指定方式を、レジスタ・バンク間ではなく、ユーザ・アプリケーション間で同じにすることが望ましく、そうすれば、ユーザ・アプリケーションを可能な限り単純化することができ、また、レジスタ・バンク間の切替えをハードウェアでサポートしたシステムにすれば、ユーザ・アプリケーションは現在使用中のレジスタ・バンクがどれであるかを意識しないで済み、また他のレジスタ・バンクや他のユーザ・アプリケーションの存在さえも意識しないで済むことである。

本発明の上記およびその他の利点は、添付図面を参照して詳述する本発明の説明および請求の範囲から理解されるはずである。

発明の概要

本発明によるレジスタ・ファイル・システムは、整数レジスタの第１サブセットと第２サブセットおよびシャドウ・サブセット（shadow subset）を含む整数レジスタ・セット、整数レジスタまたは浮動小数点レジスタとして個別的に使用できるタイプ変更可能（re－typable）なレジスタ・セットおよび個別的にアドレス可能なブール・レジスタのセットを備えている。

本発明は整数機能ユニット（integer functional unit）と浮動小数点機能ユニット（flo

ating point functional unit）を備え、これらのユニットは整数レジスタ・セットをアクセスする整数命令を実行し、複数のモードで動作する。どのモードにおいても、命令は、整数レジスタの第１サブセットへの通常アクセス権が許可されている。第１モードでは、命令は、第２サブセットへの通常アクセス権が許可されている。しかし、第２モードでは、第２サブセットへのをアクセスを試みる命令には、第２サブセットではなく、シャドウ・サブセットをアクセスする許可が与えられる。これは、命令には見えない（transparent）形で行われる。従って、どのモードで実行されるかを意識しないでルーチンを書くことができ、システム・ルーチン（これは第２モードで実行される）は、少なくとも第２サブセットを見かけ上自由に使用することができるので、第２サブセットの内容（第１モードで実行されているユーザ・プロセスが使用中の場合がある）をセーブするときに生じるオーバヘッドを負担しないで済む。

本発明によれば、さらに複数の整数レジスタ・セットが用意されている。これらのレジスタ・セットは、命令中のフィールドで指定されるようにして、個別的にアドレス指定が可能である。レジスタ・セットは読取りポートと書込みポートを含んでおり、これらはマルチプレクサによってアクセスされる。この場合、マルチプレクサはレジスタ・セットの内容、つまり、命令中のフィールドの指定によって制御される。

これらの整数レジスタ・セットの１つは、浮動小数点レジスタ・セットとしても使用可能である。本発明の１実施例では、このセットは倍精度浮動小数点データを収容するために64ビット幅になっているが、下位の32ビットだけが整数命令によって使用される。

本発明によれば、ブール演算を実行するための機能ユニットが備わっており、さらに、ブール演算の結果を保持しておくためのブール・レジスタ・セットが用意されているので、専用の固定ロケーション・ステータス・フラグが不要になっている。整数機能ユニットと浮動小数点機能ユニットは数値比較命令を実行し、これらの命令は比較の結果を収めておくブール・レジスタをそれぞれ指定している。ブール機能ユニットはブール組合せ命令（Boolean combinational instruction）を実行し、そのソースと宛先はブール・レジスタ・セットの中の指定されたレジスタになっている。従って、本発明によれば、複合（complex）ブール機能の１つの結果だけで条件付きブランチを実行できるので、複合ブール機能の基本的部分間に条件付きブランチを介在させる必要がなく、データ・プロセッサにおけるパイプライン混乱（pipeline disruption）を最小にすることができる。

最後に、システムには複数の同種レジスタ・バンクが設けられている。バンクはどのプロセスまたはルーチンにも割り振ることができるので、ルーチン内の命令はどのバンクで実行されるかを指定しないで済むようになっている。

【図面の簡単な説明】

第１図は、本発明のマイクロプロセッサの命令実行ユニットを示すブロック図であり、レジスタ・ファイルの要素を示している。

第２図～第４図は、それぞれ第１図に示した命令実行ユニットの浮動小数点、整数およびブール部分を示す簡略ブロック図である。

第５図～第６図は、それぞれ浮動小数点および整数部分を示す詳細図であり、レジスタ・セット間の選択手段を示している。

第７図は、第１図に示した命令実行ユニットによって実行可能なマイクロプロセッサ命令ワード例のフィールドを示す図である。

好適実施例の詳細な説明

I.レジスタ・ファイル

第１図は、本発明によるRISC（縮小命令セット計算）の命令実行ユニット（IEU）10の基本構成要素を示す図である。IEU 10はレジスタ・ファイル12と実行エンジン14を備えている。レジスタ・ファイル12は１つまたは２つ以上のレジスタ・バンク16－０～16－ｎを含んでいる。明らかなように、各レジスタ・バンク16は他のレジスタ・バンク16のすべてと同一構造になっている。従って、以下では、レジスタ・バンク16－０だけを説明することにする。レジスタ・バンクはレジスタ・セットA 18、レジスタ・セットFB 20、およびレジスタ・セットC 22を含んでいる。

一般的には、CISC（複合命令セット計算）命令をCISCプロセッサが実行するには、従来のレジスタ・ファイルを使用すれば十分であるのに対し、本発明によるRISCマイクロプロセッサは、RISC命令を実行する際に使用するのに最適に構成されたレジスタ・ファイルを備えたことを特徴としている。レジスタ・ファイルを特別に適応したものにすると、マイクロプロセッサのIEUの実行エンジン（execution engine）は、資源利用効率とロー・スループット（raw throughput）の面で性能を向上させることができる。一般的考え方はレジスタ・セットをRISC命令に合わせてチューン（調整）することであるが、実施例によっては、アーキテクチャのどのレジスタ・セットでも対象とすることができる。

A.レジスタ・セットＡ

レジスタ・セットA 18は整数レジスタ24（RA［31:0］を含んでおり、その各々は整数値のデータを収めるように適応化されている。１実施例によれば、各整数は32ビット幅になっている。RA［］整数レジスタ24は複数の第１整数レジスタ26（RA［23:0］）と複数の第２整数レジスタ28（RA［31:24］）から構成されている。RA［］整数レジスタ24は各々が同一構成になっており、整数レジスタ・セット24内の固有アドレス（unique address）による場合でも、各々が同じ方法でアドレス可能になっている。例えば、第１整数レジスタ30（RA［０］）は整数レジスタ・セットA18内のゼロ・オフセットにアドレス指定することが可能である。

RA［０］は常に値がゼロになっている。これは、ユーザ・アプリケーションや他のプログラムは他の定数値よりも定数値ゼロを使用することが多いことが観察されてきたためである。従って、クリア、比較その他の目的のために、ゼロがいつでも即時に使用できるようになっていることが望ましい。特定の値に関係なく、任意のレジスタに一定の配線（hard－wired）した値を入れておくと得られるもう１つの利点は、その任意のレジスタを、結果をセーブしておく必要のない命令の宛先として使用できることである。

また、このことは、固定レジスタ（fixed register）はデータに依存する遅延の原因となることがないことも意味する。データ依存関係が起こるのは、「スレーブ」命令がそのオペランドの１つまたは２つ以上のために、「マスタ」命令の結果を必要とするときである。パイプライン方式プロセッサでは、これはパイプラインを停止（stall）させる原因となる。例えば、マスタ命令は、コード列の中にスレーブ命令よりも早く現れる場合であっても、実行時間が著しく長くなることがある。このことから容易に理解されるように、スレーブの「インクリメントとストア」命令がマスタの「４倍ワード整数除算」命令の結果データに基づいて実行される場合は、スレーブ命令がフェッチされ、デコードされたあと、マスタ命令が実行を終えるまで、多数のクロック・サイクルの間、実行を待たされることになる。しかし、状況によっては、マスタ命令の数値結果は必要とされず、マスタ命令は条件コード・フラグをセットするといった、他の目的に実行される場合もある。マスタ命令の宛先がRA［０］ならば、数値結果は事実上破棄されることになる。IEU 10のデータ依存関係チェッカ（図示せず）は、マスタ命令の最終結果、つまり、ゼロはすでに分かっているので、スレーブ命令を遅延させる原因とはならない。

整数レジスタ・セットA18はシャドウ・レジスタ32（RT［31:24］）も備えている。各シャドウ・レジスタは整数値を保持することができ、１実施例では、32ビット幅にもなっている。各シャドウ・レジスタは、各整数レジスタがアドレス指定できるのと同じように、オフセットとしてアドレス指定することができる。

最後に、整数レジスタ・セットA18はIEUモード整数スイッチ（mode integer switch）34を備えている。このスイッチ34は、他の同種の要素と同じように、対応する論理的機能がレジスタ・セット内に用意されている限り、物理的にスイッチとして実現する必要はない。IEUモード整数スイッチ34はライン36を介して整数レジスタの第１サブセット26に接続され、ライン38を介して整数レジスタの第２サブセット28に、ライン40を介してシャドウ・レジスタ32に接続されている。レジスタ・セットA 18へのアクセスはすべてライン42上のIEUモード整数スイッチ34を通して行われる。第１サブセットRA［23:0］内のレジスタを読み書きするためのアクセス要求はIEUモード整数スイッチ34を通して自動的に渡される。しかし、第１サブセットRA［23:0］の外側のオフセットを使用した整数レジスタへの

アクセスは、実行エンジン14の動作モードに応じて第２サブセットRA［31:24］に向けられるか、シャドウ・レジスタRT［31:24］に向けられる。
IEUモード整数スイッチ34は実行エンジン14内のモード制御ユニット44の制御を受けて動作する。モード制御ユニット44はIEU 10に関する該当ステートまたはモード情報を、ライン46経由でIEUモード整数スイッチ34へ送る。実行エンジンがカーネル（kernel）・モードへの移行といったコンテキスト・スイッチを実行すると、モード制御ユニット44はIEUモード整数スイッチ34を次のように制御する。つまり、第２サブセットRA［31:24］への要求は、整数セット内で要求した同じオフセットを使用してシャドウRT［31:24］へリダイレクト（re－directed）される。従って、オペレーティング・システム・カーネルやそのとき実行中の他のエンティティは、見かけ上は第２サブセットRA［31:24］をアクセスすることができるので、第２サブセットRA［31:24］の内容を主メモリにスワップ・アウトしたり、第２サブセットRA［31:24］をスタック上にプッシュしたりするとき生じるオーバヘッドがなくなり、また他の従来のレジスタ格納手法を使用しないで済むことができる。
実行エンジン14が通常ユーザ・モードに戻って、制御が当初に実行中のユーザ・アプリケーションに渡されると、モード制御ユニット44は、アクセスが再び第２サブセットRA［31:24］に向けられるようにIEUモード整数スイッチ34を制御する。一実施例では、モード制御ユニット44は、IEU 10における割込み許可の現在の状態に応じて動作するようになっている。また、一実施例では、実行エンジン14はプロセッサ・ステータス・レジスタ（PSR）（図示せず）を備えており、このレジスタは、割込みが許可されているか、禁止されているかを示した１ビット・フラグ（PSR［７］）をもっている。従って、IEUモード整数スイッチ34とPSR内の割込み許可フラグとは、ライン46で結合するだけでよい。割込みが禁止されている間は、IEU 10は整数RA［23:0］へのアクセス権を保持しているので、ユーザ・アプリケーションの各種データの分析を行うことができる。これにより、デバッグ、エラー報告、またはシステム・パフォーマンス分析を向上させることができる。
B.レジスタ・セットFB
タイプ変更可能な（re－typable）レジスタ・セットFB 20とは、浮動小数点レジスタ48（RF［31:0］）および／または整数レジスタ50（RB［31:0］）を含んでいるものと考えてよい。いずれかのデータ・タイプの場合も、一方が他方を除外することを意味しないときは、本明細書では、RFB［］という用語を用いることにする。一実施例では、浮動小数点レジスタRF［］は、整数レジスタRB［］と同じ物理的シリコン・スペースを占めている。また、一実施例では、浮動小数点レジスタRF［］は64ビット幅に、整数レジスタRB［］は32ビット幅になっている。このことから理解されるように、倍精度浮動小数点数が必要でなければ、レジスタ・セットRFB［］を32ビット幅の構成にすると、各浮動小数点レジスタの余分の32ビット用に必要になるシリコンの面積を節約できるという利点がある。
レジスタ・セットRFB［］中の個々のレジスタの各々は、浮動小数点値と整数値のいずれかを保持することができる。レジスタ・セットRFB［］には、オプションとして、浮動小数点値が整数値であるものとして、あるいは整数値が浮動小数点値であるものとして、誤ってアクセスされるのを防止するハードウェアを設けることが可能である。しかし、一実施例では、レジスタ・セットRFB［］を単純化するために、個々のレジスタが誤った使い方をされるのを防止することは、ソフトウェア設計者に任すようにしている。従って、実行エンジン14は、レジスタ・セットRFB［］までのオフセットを指定して、ライン52上にアクセス要求を出すだけであり、あるオフセットに置かれたレジスタが浮動小数点レジスタとして使用されるものか、整数レジスタとして使用されるものかは指定しない。実行エンジン14内では、各種エンティティはレジスタ・セットRFB［］に用意されている64ビット全部を使用することも、例えば、整数演算や単精度浮動小数点演算において、下位32ビットだけを使用することもできる。
最初のレジスタRFB［０］51は、RB［０］が32ビット整数のゼロ（0000_{hex}）となり、RF［０］が64ビット浮動小数点のゼロ（00000000_{hex}）となる形式で、定数値のゼロを格納する。このようにすると、RA［０］に対して上述したのと同じ利点が得られる。

C.レジスタ・セットC
レジスタ・セットC 22は、複数のブール・レジスタ（RC［31:0］）から構成されている。RC［］は「条件ステータス・レジスタ」（CSR）とも呼ばれている。ブール・レジスタはいずれも構造とアドレス指定方式が同じになっている。ただし、各レジスタはRC［］内の固有アドレスまたはオフセットに個別的にアドレスすることが可能である。
一実施例では、レジスタ・セットCはさらに「旧条件ステータス・レジスタ」（PCSR）60を含んでおり、レジスタ・セットCはCSRセレクタ・ユニット62も備えている。このユニットはモード制御ユニット44に応答して、CSR 54とPCSR 60を交互に選択する。この実施例では、CSRは割込みが許可されているとき使用され、PCSRは割込みが禁止されているとき使用される。CSRとPCSRは他の点ではすべて同じである。また、この実施例では、割込み禁止とセットされると、CSRセレクタ・ユニット62はCRSの内容をプッシュしてPCSRに入れ、PCSRの旧内容に重ね書きする。割込みが再び許可されると、CSRセレクタ・ユニット62はPCSRの内容をポップしてCSRに戻す。他の実施例では、RA〔31:24〕およびRT［31:24］で行われるのと同じように、CSRとPCSRの間でアクセスを交互に行うことが望ましい場合がある。いずれの場合も、PCSRは常に32ビット「特殊レジスタ」として使用することが可能である。
ブール・レジスタは、従来公知のマイクロプロセッサにおけるブール・レジスタと異なり、いずれも専用条件フラグになっていない。つまり、CSR 54は、専用キャリー・フラグも、専用マイナス・フラグも、比較が一致したことまたは減算結果がゼロであることを示す専用フラグも含んでいない。その代わりに、どのブール・レジスタも任意のブール・オペレーションのブール結果の宛先となることができる。他のレジスタ・セットの場合と同じように、最初のブール・レジスタ58（RC［０］は常に値ゼロが入るので、RA［０］で上述したのと同じ利点が得られる。好適実施例では、各ブール・レジスタは１ビット幅で、１つのブール値を示すようになっている。
II.実行エンジン
実行エンジン14は１つまたは２つ以上の整数機能ユニット66、１つまたは２つ以上の浮動小数点機能ユニット68、および１つまたは２つ以上のブール機能ユニット70を備えている。これらの機能ユニットは、以下に説明するように命令を実行する。バス72、73、75はIEU 10の種々エレメントを結んでおり、それぞれはデータ経路、アドレス経路、および制御経路を表しているものとする。
A.命令の形式
第７図は、実行エンジン14に実行させることができる整数命令の形式（フォーマット）を示す一例である。理解されるように、すべての命令が図示の形式に厳密に従う必要はなく、データ処理システムには命令フェッチャとデコード（図示せず）が含まれており、これらは形式の異なる命令を処理するように構成されている。第７図には、理解を容易にするために、１つの例だけが示されている。本明細書全体を通して、符号［］は命令の各ビットを示すために用いられている。I［31:30］は、実行エンジン14の将来の実装に備えて予約されている。I［29:26］は特定の命令の命令クラスを示している。表１は、本発明によって実行される命令の各種クラスを示すものである。

表１
命令のクラス

クラス	命令
0-3	整数および浮動小数点レジスタ間命令
4	即値定数ロード(immediate constant load)
5	予約
6	ロード
7	ストア
8-11	制御フロー
12	修飾子（Modifier)
13	ブール演算
14	予約
15	アトミック（拡張）

本発明で特に重要な命令クラスは、クラス０－３のレジスタ間命令とクラス13のブール・オペレーションである。他のクラスの命令もレジスタ・ファイル12を処理するが、これらのクラスを詳しく説明しなくても、本発明を十分に理解できると思われるので、説明を省略する。

Ｉ［25］はB0と名付けられ、これは宛先レジスタがレジスタ・セットＡにあるか、レジスタ・セットＢにあるかを示している。Ｉ［24:22］は、ある命令クラス内でどの特定機能を実行すべきかを指定した命令コード（opcode）である。例えば、レジスタ間命令クラス内では、命令コードは「加算」を指定することができる。Ｉ［21］は、命令の実行時に使用すべきアドレス指定モード（addressing mode）、つまり、レジスタ・ソースのアドレス指定または即値ソース（immediate source）のアドレス指定を指定している。Ｉ［20:16］は、宛先レジスタをB0で指定したレジスタ・セット内のオフセットとして指定している。Ｉ［15］はB1と名付けられ、これは第１オペランドがレジスタ・セットＡから得られるか、またはレジスタ・セットＢから得られるかを示している。Ｉ［14:10］は、第１オペランドを得るときのレジスタがどれだけオフセットしているかを指定している。Ｉ［9:8］は命令コードＩ［24:22］の延長部分であり、機能の選択を指定している。Ｉ［7:6］は予備である。Ｉ［５］はB2と名付けられ、命令の第２オペランドがレジスタ・セットＡから得られるか、またはレジスタ・セットＢから得られるかを指定している。最後に、Ｉ［4:0］は、第２オペランドを得るときのレジスタがどれだけオフセットしているかを指定している。

第１図に示すように、整数機能ユニット66と浮動小数点機能ユニット68はそれぞれ整数比較命令と浮動小数点比較命令を実行する機能を備えている。比較命令の命令形式は第７図

に示されているものとほぼ同じであるが、各フィールドは多少異なる名前を付けて区別しておくと便利である。Ｉ［20:16］は結果をストアしておくべき宛先レジスタを指定しているが、アドレス指定モード・フィールドＩ［21］はレジスタ・セットＡとＢとの間の選択を行わない。その代わりに、アドレス指定モード・フィールドは比較の第２ソースがレジスタに入っているか、即値データ（immediate data）であるかを示している。比較はブール型命令であるので、宛先レジスタは常にレジスタ・セットＣに置かれている。他のフィールドの働きは第７図に示されている。整数および浮動小数点機能ユニットでブール・オペレーションを実行するとき、命令コードと機能選択フィールドは２オペランドを比較する際にどのブール条件をテストするかを指定している。整数および浮動小数点機能ユニットは数値比較に関するIEEE標準を完全にサポートしている。

IEU 10はロード／ストア・マシン（load/store machne）である。つまり、あるレジスタの内容がメモリにストアされるか、またはメモリから読み出されるとき、どのメモリ・ロケーションをストアまたはロードのソースまたは宛先とするかを判断するためにアドレス計算をしなければならない。そのような場合には、宛先レジスタ・フィールドＩ［20:16］はロードまたはストアの宛先またはソースとなるレジスタを指定している。ソース・レジスタ１フィールドＩ［14:10］は、メモリ・ロケーションのベース・アドレスを収めているレジスタがセットＡのものか、あるいはセットＢのものかを指定している。一実施例では、ソース・レジスタ２フィールドＩ［4:0］はインデックスまたはベースからのオフセットを収めているレジスタがセットＡのものか、またはセットＢのものかを指定している。ロード／ストア・アドレスはベースにインデックスを加えると求まる。別のモードでは、Ｉ［7:0］はインデックスとしてベースに加えるべき即値データを収めている。

B.命令実行ユニットとレジスタ・セットのオペレーション

この分野に精通したものならば理解されるように、整数機能ユニット66、浮動小数点機能ユニット68、およびブール機能ユニット70は、実行しようとする現命令の命令クラス・フィールド、命令コード・フィールド、および機能選択フィールドの内容に応じて動作する。

1.整数オペレーション

例えば、命令クラス、命令コード、および機能選択が整数レジスタ間加算を実行することを示しているときは、整数機能ユニットはそれを受けて指示されたオペレーションを実行するのに対し、浮動小数点機能ユニットとブール機能ユニットはそれを受けてもそのオペレーションを実行しない。しかし、本件特許出願と同時に出願された係属中の米国特許出願（冒頭に列挙）から理解されるように、浮動小数点機能ユニット68は、浮動小数点と整数の両方のオペレーションを実行する機能を備えている。また、これらの機能ユニットはいずれも２つ以上の命令を同時に実行する構成になっている。

整数機能ユニット66は整数演算機能だけを備えている。整数演算を行うには、一般的に、第１ソース、第２ソース、および宛先が必要になる。どの整数型命令も、実行すべき特定のオペレーションを１つまたは２つ以上のソース・オペランドで指定し、整数オペレーションの結果を特定の宛先にストアすることを指定する。ロード／ストア・オペレーションで使用されるアドレス計算のような、ある種の命令では、ソースはベースおよびインデックスとして使用される。整数機能ユニット66は第１バスに接続され、このバスを介して整数機能ユニット66はスイッチングとマルチプレクシング制御（SMC）ユニットA 74およびSMCユニットB 76に接続されている。整数機能ユニット66によって実行される各整数型命令はそのソースと宛先がレジスタ・セットＡに置かれているか、レジスタ・セットＢに置かれているかを指定する。

以下では、IEU 10が整数型レジスタ間加算を実行する命令を機能フェッチ・ユニット（図示せず）から受け取った場合を想定して説明する。各種実施例において、この命令はレジスタ・バンクを指定することができ、各ソースと宛先ごとに別のバンクを指定することもできる。一実施例では、命令Ｉ［］は長さが32ビットまでに制限されているので、どのレジスタ・バンク16－０～16－ｎが命令に関係しているかを示す標識は収められない。その代わりに、バック・セレクタ・ユニット78がどのレジスタ・バンクが現在アクティブであ

るかを制御する。一実施例では、バンク・セレクタ・ユニット78は、IEU 10内のステータス・ワード（図示せず）中の１または２つ以上のバンク選択ビットに応答して動作する。整数型加算命令を実行するために、整数機能ユニット66は第１および第２ソース・レジスタのＩ［14:10］およびＩ［4:0］内のID（識別）に応動する。整数機能ユニット66は第１および第２ソース・レジスタのIDをそれぞれポートS1およびS2から出力し、SMCユニットA 74とSMCユニットB 76に接続された整数機能ユニット・バス72上に送出する。一実施例では、SMCユニットＡとＢはそれぞれ命令Ｉ［］からB0－２を受け取るように接続されている。一実施例では、それぞれのBnに入っているゼロはレジスタ・セットＡを示し、１はレジスタ・セットＢを示している。ロード／ストア・オペレーション時には、整数と浮動小数点機能ユニット66および68のソース・ポートは、それぞれ、ベース・ポートＢおよびインデックス・ポートＩとして使用される。

指示されたレジスタ・セットから第１および第２オペランドをバス72上に得ると、下述するように、整数機能ユニット66はこれらのオペランドについて指示されたオペレーションを実行し、その結果をポートＤから出力し、整数機能ユニット・バス72上に送出する。SMCユニットＡとＢはB0に応答して、その結果を該当するレジスタ・セットＡまたはＢあてに送る。

SMCユニットＢはさらに命令クラス、命令コード、および機能選択に応答し、オペランドを浮動小数点レジスタRF［］から読み取るか、整数レジスタRB［］から読み取るか（あるいか結果をそのどちらにストアするか）を制御する。上述したように、一実施例では、レジスタRF［］は64ビット幅にできるのに対し、レジスタRB［］はわずか32ビット幅である。従って、SMCユニットＢはレジスタ・セットRFB［］にワードを書くか、またはダブル・ワードを書くかを制御する。SMCユニットＡには、バス42上のデータ転送の幅を制御する手段を設ける必要はない。

バス42上のすべてのデータは32ビット幅であるが、レジスタ・セットＡには他の種類の複雑さが存在する。IEU 10は実行エンジン14のモード制御ユニット44の制御に応答し、バス42上のデータをバス42経由でバス36、バス38またはバス40に接続するかどうか、およびその逆に接続するかどうかを制御する。

IEUモード整数スイッチ34はさらにＩ［20:16］、Ｉ［14:10］、およびＩ［4:0］に応答する。指示されたある宛先またはソースがRA［23:0］にあれば、IEUモード整数スイッチ34は自動的にデータをライン42と36との間に結合する。しかし、レジスタRA［31:24］に対しては、IEUモード整数スイッチ24はライン42上のデータがライン38またはライン40に接続されているかどうか、およびその逆に接続されているかどうかを判断する。割込みが許可されているときは、IEUモード整数スイッチ34はSMCユニットＡを整数レジスタRA［31:24］の第２サブセット28に接続する。割込みが禁止されているときは、IEUモード整数スイッチ34はSMCユニットＡをシャドウ・レジスタRT［31:24］に接続する。従って、整数機能ユニット66内で実行される命令は、RA［31:24］をアドレス指定すべきか、またはRT［31:24］をアドレス指定すべきかを意識する必要はない。このことから理解されるように、SMCユニットＡは、それが整数機能ユニット66によってアクセスされるか、または浮動小数点機能ユニット68によってアクセスされるかに関係なく、同じように動作できるいう利点がある。

2.浮動小数点オペレーション

浮動小数点機能ユニット68は、命令のクラス、命令コード、および機能選択フィールドを受けて動作し、浮動小数点オペレーションを実行する。S1、S2、およびＤポートは整数機能ユニット66で上述したように動作する。SMCユニットＢは、バス52経由で浮動小数点レジスタRF［］からの浮動小数点オペランドに応答し、そして数値浮動小数点結果を浮動小数点レジスタRF［］に書き込む。

3.ブール・オペレーション

SMCユニットC 80は命令Ｉ［］の命令クラス、命令コード、および機能選択フィールドを受けて動作する。SMCユニットＣは、比較オペレーションが数値機能ユニット66または68の一方によって実行されたことを検出すると、SMCユニットＣはその比較を実行した機能

ユニットのDポートにおいて示されたブール・レジスタにバス56を介してブール・オペレーション結果を書き込む。
ブール機能ユニット70は、整数および浮動小数点機能ユニット66と68とは異なり、比較命令を実行しない。その代わりに、ブール機能ユニット70は、ブール・レジスタ内容のビット単位の論理的組合せを、表２に列挙されたブール機能に従って実行するのに使用されるだけである。

表２
ブール機能

I[23,22,9,8]	ブール演算結果計算
0000	ZERO
0001	S1 AND S2
0010	S1 AND (NOT S2)
0011	S1
0100	(NOT S1) AND S2
0101	S2
0110	S1 XOR S2
0111	S1 OR S2
1000	S1 NOR S2
1001	S1 XNOR S2
1010	NOT S2
1011	S1 OR (NOT S2)
1100	NOT S1
1101	(NOT S1) OR S2
1110	S1 NAND S2
1111	ONE

複数の同種ブール・レジスタを用意し、その各々が個別的にブール・オペレーションの宛先としてアドレス指定できるようにすると得られる本発明の利点について、表３～５を参照して説明する。表３は、条件付きブランチを複合（complex）ブール機能に基づいて実行するコード・セグメントの例を示すものである。複合ブール機能は論理和（OR）がとられる３つの部分からなっている。最初の部分はさらに２つの部分からなり、これらは論理積（AND）がとられる。

表３
複合ブール機能の例

```
1 RA[1] := 0;
2 IF((RA[2] = RA[3]) AND (RA[4] > RA[5])) OR
3    (RA[6] < RA[7]) OR
4    (RA[8] <> RA[9])) THEN
5     X()
6 ELSE
7     Y();
8 RA[10] := 1;
```

表４は、従来マイクロプロセッサが表３の機能を実行するときの１つの類似方法を疑似アセンブリ形式で示すものである。表４のコードは、表３のコードを処理する少なくとも通常インテリジェント機能をもつコンパイラによって生成されるものとして書かれている。つまり、コンパイラは、３部分のいずれかが真であると、表３の２行目から４行目に表されている条件が渡されることを認識する。

表４
ブール・レジスタ・セットを使用しない
複合ブール機能の実行

```
 1 START      LDI  RA[1],0
 2 TEST1      CMP  RA[2],RA[3]
 3            BNE  TEST2
 4            CMP  RA[4],RA[5]
 5            BGT  DO_IF
 6 TEST2      CMP  RA[6],RA[7]
 7            BLT  DO_IF
 8 TEST3      CMP  RA[8],RA[9]
 9            BEQ  DO_ELSE
10 DO_IF      JSR  ADDRESS OF X()
11            JMP  PAST_ELSE
12 DO_ELSE    JSR  ADDRESS OF Y()
13 PAST ELSE  LDI  RA[10],1
```

表３の１行目の割当ては表４の１行目の「即値ロード」ステートメントによって実行される。表３の２行目に表されている複合ブール条件の最初の部分は表の４の２行目～５行目のステートメントで表される。RA［２］がRA［３］に等しいかどうかをテストするには、表４の２行目の比較ステートメントは、どのようにコーディングしたかに応じて、RA［３］からRA［２］を減算することあるいはその逆を実行し、減算の結果をストアする場合とストアしない場合がある。比較ステートメントによって実行される重要な機能は、ゼロ、マイナス、およびキャリー・フラグがその結果に応じてセットまたはクリアされることである。

表４の３行目の条件付きブランチ・ステートメントは、RA［２］がRA［３］と等しくなかったことを条件として、コードの後続部分へブランチする。２つが等しくないと、ゼロ・フラグがクリアされるので、２番目のサブ部分を実行する必要がない。表４の３行目に条件付きブランチ・ステートメントが存在するので、２行目の比較結果が分かるまでは、表４の後続ステートメントのフェッチ、デコードおよび実行が禁止されるために、パイプライン停止（pipeline stall）が起こることになる。最初の部分（TEST1）の最初のサブ部分が渡されると、表４の４行目の第２サブ部分はRA［４］をRA［５］と比較するので、この場合も、該当ステータス・フラグがセットまたはクリアされることになる。

RA［２］がRA［３］と等しく、RA［４］がRA［５］より大きければ、複合ブール機能の中の残りの２部分（TEST2とTEST3）をテストする必要がなく、表４の５行目のステートメントは条件付きでラベルDO_IFへブランチし、表４の“IF"の内側のオペレーションが実行さ

れる。しかし、テストの最初の部分が失敗すると、“IF"と“ELSE"部分のどちらを実行させるかを判定するために追加の処理が必要になる。
ブール機能の第２部分は表４の６行目でRA［６］をRA［７］と比較するものであり、この場合も、該当ステータス・フラグがセットまたはクリアされる。「より小さい」の条件がステータス・フラグによって指示されていると、複合ブール機能が渡され、その実行は即時にDO_IFラベルへブランチすることができる。従来の各種マイクロプロセッサでは、「より小さい」条件はマイナス・フラグを検査することでテストすることが可能になっている。RA［７］がRA［６］より小でなかったときは、テストの第３部分を実行する必要がある。表４の８行目のステートメントはRA［８］をRA［９］と比較する。この比較が失敗したときは、“ELSE"コードを実行させる必要がある。そうでなければ、実行は表４の10行目の“IF"コードへフォールスルー（fall through）するだけであり、そのあと“ELSE"コードの前後で別のジャンプが行われることになる。表４の３、５、７および９行目の条件付きブランチはそれぞれが別々のパイプライン停止を引き起こすので、この複合ブール機能を処理するために必要な処理時間が大幅に増加することになる。
本発明のブール・レジスタ・セットＣを採用するとスループットが大幅に向上することは、表５を特に参照すれば容易に理解されるはずである。

表５
ブール・レジスタ・セットを使用した
複合ブール機能の実行

```
 1 START       LDI RA[1],0
 2 TEST1       CMP RC[11],RA[2],RA[3],EQ
 3             CMP RC[12],RA[4],RA[5],GT
 4 TEST2       CMP RC[13],RA[6],RA[7],LT
 5 TEST3       CMP RC[14],RA[8],RA[9],NE
 6 COMPLEX     AND RC[15],RC[11],RC[12]
 7             OR  RC[16],RC[13],RC[14]
 8             OR  RC[17],RC[15],RC[16]
 9             BC  RC[17],DO_ELSE
10 DO_IF       JSR ADDRESS OF X()
12             JMP PAST_ELSE
13 DO_ELSE     JSR ADDRESS OF Y()
14 PAST_ELSE   LDI RA[10],1
```

特に表５の２行目～５行目に示すように、ブール・レジスタ・セットＣを使用すると、マイクロプロセッサは、途中にブランチを介在しないで３つのテスト部分を連続して実行することができる。各ブール比較は２つのオペランド、宛先、およびテストすべきブール条

件を指定している。例えば、表５の２行目の比較はRA［２］の内容をRA［３］の内容と比較し、その内容が等しいかをテストし、比較結果のブール値をRC［11］にストアしている。表５に示すように、ブール機能の各比較はそれぞれの中間結果を別々のブール・レジスタにストアしている。冒頭に列挙した関連特許出願から理解されるように、IEU 10はこれらの比較を２つ以上同時に実行する機能を備えている。

表５の２行目～３行目の最初の２つの比較が完了すると、２つの各中間結果の論理積（AND）が表３の６行目に示すようにとられる。その後、テストの最初の部分の結果がRC〔15〕に格納される。ブール機能の第２と第３サブ部分の結果の論理和（OR）が表５の７行目に示すようにとられる。理解されるように、データ依存関係がないので、６行目のAND（論理積）と７行目のOR（論理和）は並列に実行することができる。最後に、これら２つのオペレーションの結果の論理和が表５の８行目に示すようがとられる。

表から理解されるように、表３の複合ブール機能全体が真であるか、偽であるかを示すブール値がRC［17］に収められる。その後、表５の９行目に示すように、単純条件付きブランチを実行することが可能である。表５に示すモードでは、ブール・レジスタRC［17］がクリアで、複合機能が失敗したことを示していると、“ELSE"コードへブランチする。コードの残余部分は、表４に示すようなブール・レジスタがない場合と同じにすることができる。

ブール機能ユニット70は他の機能ユニットと同じように、命令クラス、命令コード、および機能選択フィールドに応動する。従って、この場合も、表５から理解されるように、整数機能ユニットおよび／または浮動小数点機能ユニットは、１行目～５行目と13行目の命令を実行し、ブール機能ユニット70は６行目～８行目のブール・ビット単位結合命令（Boolean bitwise combination instruction）を実行する。９行目～12行目の制御フローとブランチ命令はIEU 10のエレメント（第１図には示されていない）によって実行される。

III.データ経路

第２図～第５図は、それぞれ、IEUの浮動小数点、整数、およびブール演算部分内のデータ経路を示す詳細図である。

A.浮動小数点部分のデータ経路

第２図に示すように、レジスタ・セットFB 20はマルチポート・レジスタ・セットである。一実施例では、レジスタ・セットFB 20は２つの書込みポートWFB0－１と５つの読取りポートRDFB0－４をもっている。第１図の浮動小数点機能ユニット68は第２図のALU2 102、FALU 104、MULT 106、およびNULL 108から構成されている。レジスタ・セット20とエレメント102～108を除く第２図のすべてのエレメントは第１図のSMCユニットＢを構成している。

外部双方向データ・バスEX_DATA［］はデータを浮動少数点ロード／ストア・ユニット122へ送る。即値浮動小数点データ・バスLED_IMED［］は「即値ロード」命令からのデータを送る。他の即値浮動小数点データは「即値加算」命令で必要となるものと同じバスRFF1_IMEDおよびRFF2_IMDE上を送られる。データは「特殊レジスタ移動」命令を受けると、バスEX_SR_DT［］上にも送られる。データは第３図に示す整数部分からバス114と120上を送られてくることもある。

浮動小数点レジスタ・セットの２つの書込みポートWFB0とWFB1は、それぞれ書込みマルチプレクサ110－０と110－１に結合されている。書込みマルチプレクサ110は第３図の整数部分のALU0またはSHF0;FALU;MULT;ALU2:EX_SR_DT［］またはLDF_IMED［］；およびEX_DATA［］からデータを受け取る。当業者なら理解されるように、各ポートからどの入力が選択されるかは制御信号（図示せず）から判断され、入力データをどのレジスタに書くかはアドレス信号（図示せず）から判断される。マルチプレクサの制御とアドレス指定は公知であるので、ここでは、マルチプレクサまたはレジスタ・セットに関する詳しい説明は省略する。

浮動小数点レジスタ・セットの５つの読取りポートRDFB0～RDFB4は、それぞれ読取りマルチプレクサ112－０～112－４に結合されている。読取りマルチプレクサは、それぞれ、即値ロード・バイパス・バス126上のEX_SR_DT［］またはLDF_IMED［］；外部ロード・デー

タをレジスタ・セットFBをスキップさせる外部ロード・データ・バイパス・バス127;非乗算整数オペレーションを実行するALU2 102の出力；非乗算浮動小数点オペレーションを実行するFALU 104;乗算オペレーションを実行するMULT 106;それぞれ非乗算整数オペレーションとシフト・オペレーションを実行する第３図の整数部分のALU0 140またはSHF0 144からもデータを受け取る。読取りマルチプレクサ112－１と112－３はそれぞれRFF1_IMED［］とRFF2_IMED［］からもデータを受け取る。

浮動小数点部分内の各算術演算型ユニット102～106は、第１および第２ソース・マルチプレクサS1とS2のそれぞれのセットから２つの入力を受け取る。各ユニットALU2、FALUおよびMULTの最初のソースは読取りマルチプレクサ112－０または112－２の出力から得られ、２番目のソースは読取りマルチプレクサ112－１または112－３の出力から得られる。FALUとMULTのソースはバス114経由で第３図の整数部分からも得られる。

ALU2、FALU、およびMULTの結果は書込みマルチプレクサ110へ送り返されて浮動小数点レジスタRF［］にストアされ、読取りマルチプレクサ112へも送り返されて以後のオペレーションのオペランドとして再使用される。FALUは浮動小数点比較オペレーションのブール結果を示す信号FALU_BDをも出力する。FALU_BDはFALUの内部ゼロ・フラグと符号フラグから直接に計算される。

空白（null）バイト・テスタNULL 108は、あるモードにあるとき、ALU2の第１ソース・マルチプレクサからのオペランドについて空白バイト・テスト操作を行う。NULL 108は、32ビットの最初のソース・オペランドに値がゼロのバイトが含まれているかどうかを示したブール信号NULLB_BDを出力する。

読取りマルチプレクサ112－０、112－１および112－４の出力はバス118経由で（第３図の）整数部分へ送られる。読取りマルチプレクサ112－４の出力は、STDT_FP［］ストア・データとして浮動小数点ロード／ストア・ユニット122へも送られる。

第５図はS1とS2マルチプレクサのコントロールを示す詳細図である。図示のように、一実施例では、各S1マルチプレクサは命令Ｉ［］のビットB1を受けて動作し、各S2マルチプレクサは命令Ｉ［］のビットB2を受けて動作する。S1とS2マルチプレクサは各種機能ユニット用のソースを選択する。ソースはレジスタ・ファイルのどちらかから得ることができ、これは命令自体のB1とB2ビットによって制御される。さらに、各レジスタ・ファイルは２つの読取りポートを備え、そこからソースを得ることもできるが、その制御は図示していないハードウェアで行われる。

B.整数部分のデータ経路

第３図に示すように、レジスタ・セットA 18もマルチポートになっている。一実施例では、レジスタ・セットA 18は２つの書込みポートWA0－１と５つの読取りポートRDA0－４を備えている。第１図の整数機能ユニット66は、第３図のALU0 140、ALU1 142、SHF0 144、およびNULL 146から構成されている。レジスタ・セットA 18およびエレメント140－146を除く第３図のすべてのエレメントは第１図のSMCユニットＡを構成している。

外部データ・バスEX_DATA［］はデータを整数ロード／ストア・ユニット152へ送る。バスLDI_IMED［］上の即値整数データは「即値ロード」命令を受けて送られる。他の即値整数データは「即値加算」などの非即値ロード命令を受けてバスRFA1_IMEDとRFA2_IMED上を送出される。データは「特殊レジスタ移動」命令を受けてバスEX_SR_DT［］上にも送出される。データはバス116と118経由で浮動小数点部分（第２図に図示）から送られてくる場合もある。

整数レジスタ・セットの２つの書込みマルチプレクサWA0とWA1はそれぞれ148－０と148－１に結合されている。書込みマルチプレクサ148は、（第２図の）浮動小数点部分のFALUまたはMULT;ALU0;ALU1;SHF0;EX_SR_DT［］またはLDI_IMED［］；およびEX_DATA［］からデータを受け取る。

整数レジスタ・セットの５つの読取りポートRDA0～RDA4はそれぞれ読取りマルチプレクサ150－０～150－４に結合されている。各読取りマルチプレクサは即値ロード・バイパス・バス160上のEX_SR_DT［］またはLDI_IMED［］；外部ロード・データがレジスタ・セットＡをスキップすることを可能にするロード外部データ・バイパス・バス154;ALU0;ALU1;SH

F0;および（第２図の）浮動小数点部分のFALUまたはMULTからもデータを受け取る。読取りマルチプレクサ150－１と150－３はそれぞれRFA1_IMED［］とRFA2_IMED［］からもデータを受け取る。

整数部分内の各算術演算型ユニット140－144は第１および第２ソース・マルチプレクサS1とS2のそれぞれのセットから２つの入力を受け取る。ALU0の第１ソースは読取りマルチプレクサ150－２の出力、または32ビット幅の定数ゼロ（0000_{hex}）、または浮動小数点読取りマルチプレクサ112－４のいずれかから得られる。ALU0の第２のソースは読取りマルチプレクサ150－３または浮動小数点読取りマルチプレクサ112－１のいずれかから得られる。ALU1の第１ソースは読取りマルチプレクサ150－０またはIF_PC［］のいずれかから得られる。IF_PC［］は、IEUが命令を順序外（out－of－order）の順序で実行できる機能を備えているために、命令フェッチ・ユニット（図示せず）が必要とするリターン・アドレスを計算する際に使用される。ALU1の第２ソースは読取りマルチプレクサ150－１またはCF_OFFSET［］のいずれかから得られる。CF_OFFSET［］は、同じく順序外で実行できるために、CALL命令のリターン・アドレスを計算する際に使用される。

シフタSHF0 144の第１ソースは、浮動小数点読取りマルチプレクサ112－０または112－４またはいずれかの整数読取りマルチプレクサ150から得られる。SHF0の第２ソースは、浮動小数点読取りマルチプレクサ112－０または112－4;または整数読取りマルチプレクサ150－０、150－２、または150－４から得られる。SHF0は第３の入力をシフト量（shift amount－SA）マルチプレクサから得る。第３の入力はどれだけシフトさせるかを制御するもので、SAマルチプレクサによって浮動小数点読取りマルチプレクサ112－１、整数読取りマルチプレクサ150－１または150－３または５ビット幅定数32（11111_2または31_{10}）から取得される。シフタSHF0はサイズ・マルチプレクサ（Ｓ）から第４の入力を要求する。第４の入力はどれだけのデータをシフトさせるかを制御するもので、Ｓマルチプレクサによって読取りマルチプレクサ150－１、読取りマルチプレクサ150－３、または５ビット幅定数16（10000_2または16_{10}）から取得される。

ALU0、ALU1、およびSHF0の結果は書込みマルチプレクサ148へ送り返されて整数レジスタRA［］にストアされ、および読取りマルチプレクサ150へも送り返されて以後の演算のオペランドとして再使用される。ALU0またはSHF0のいずれの出力はバス120を経由して第２図の浮動小数点部分へ送られる。ALU0およびALU1は、それぞれ、整数比較オペレーションのブール結果を示した信号ALU0_BDおよびALU1_BDをも出力する。ALU0_BDとALU1_BDはそれぞれの機能ユニットのゼロ・フラグと符号フラグから直接に計算される。ALU0は信号EX_TADR［］およびEX_VM_ADRをも出力する。EX_TADR［］は絶対ブランチ命令に対する生成されたターゲット・アドレスであり、ターゲット命令をフェッチするためにIFU（図示せず）へ送られる。EX_VM_ADR［］は、メモリからロードし、メモリへストアするとき使用される仮想メモリであり、アドレス変換のためにVMU（図示せず）へ送られる。

空白バイト・テスタNULL 146は、第１ソース・マルチプレクサからのオペランドについて空白バイト・テスト操作を行う。一実施例では、このオペランドはALU1から得られる。NULL 146は、32ビットの第１ソース・オペランドに値がゼロのバイトが含まれているかどうかを示したブール信号NULLA_BDを出力する。

読取りマルチプレクサ150－０および150－１の出力はバス114を経由して（第２図の）浮動小数点部分へ送られる。また、読取りマルチプレクサ150－４の出力は、STDT_INT［］ストアデータとして整数ロード／ストア・ユニットへ送られる。

制御ビットPSR［７］はレジスタ・セットA 18へ送られる。第１図において、モード制御ユニット44からライン46上をIEUモード整数スイッチ34へ送られるのは、この信号である。IEUモード整数スイッチは第３図に示すようにレジスタ・セットA 18の内部に置かれている。

第６図はS1およびS2マルチプレクサのコントロールの詳細を示した図である。

C.ブール部分のデータ経路

第４図に示すように、レジスタ・セットC 22もマルチポートを備えている。一実施例では、レジスタ・セットC 22は２つの書込みポートWC0－１と５つの読取りポートRDA0－４を

備えている。レジスタ・セットC22とブール組合せユニット（Boolean combinational unit）70を除く第４図のすべてのエレメントは第１図のSMCユニットＣを構成している。
ブール・レジスタ・セットの２つの書込みポートWC0およびWC1は、それぞれ、書込みマルチプレクサ170－０および170－１に結合されている。書込みマルチプレクサ170はブール組合せオペレーションのブール結果を示しているブール組合せユニット70の出力；整数比較のブール結果を示している第３図の整数部分からのALU0_BD;浮動小数点比較のブール結果を示している第２図の浮動小数点部分からのFALU_BD;ALU1における比較命令の結果を示しているALU1からのALU1_BD－Ｐまたは整数部分における空白バイトを示しているNULL 146からのNULLA_BD;およびALU2における比較オペレーションの結果を示しているALU2からのALU2_BD_Pまたは浮動小数点部分における比較オペレーションの結果を示しているNULL 108からのNULLB_BDからデータを受け取る。あるひとつのモードでは、ALU0_BD、ALU1_BD、ALU2_BD、およびFALU_BD信号はデータ経路から受け取られないで、PSR内のゼロ・フラグ、マイナス・フラグ、キャリー・フラグ、その他の条件フラグに応じて計算される。１実施モードでは、どの瞬時においても最大８命令までがIEUで実行できるので、IEUは最大８個のPSRを維持している。
ブール・レジスタ・セットＣはバスEX_SR_DT［］にも結合されて、「特殊レジスタ移動」命令で使用される。CSRは、１つの32ビット・レジスタであるかのように全体として読み書きされる。この結果、ある種の重大なシステム・エラーが起こったときや、ある種の大規模なコンテキスト・スイッチングを行ったとき、必要に応じて、マシン・ステート情報を高速にセーブし復元することができる。
ブール・レジスタ・セットの４つの読取りポートRDC0～RDC3はそれぞれ読取りマルチプレクサ172－０～170－３に結合されている。読取りマルチプレクサ172は書込みマルチプレクサ170が受け取るのと同じセットの入力を受け取る。ブール組合せユニット70は、読取りマルチプレクサ170－０と170－１から入力を受け取る。読取りマルチプレクサ172－２および172－３は、それぞれ、信号BLBP_CPORTおよびBLBP_DPORTを出力する。BLBP_CPORTはIEUで条件付きブランチ命令の基礎として使用される。BLBP_DPORTは「ブールによる加算」命令で使用され、ＡまたはＢセットの中の整数レジスタを、Ｃセットの中のレジスタの内容に応じて、ゼロか１（先行ゼロ付き）にセットする。読取りポートRDC4は現時点では未使用であり、IEUのブール演算機能を将来強化する場合に備えて予約されている。

IV.結論

本発明の特徴および利点を、本発明の特定実施例を参照して種々説明してきたが、本発明は上述した実施例に限定されるものではない、本発明の範囲は請求の範囲に明確化されている通りである。

【図１】

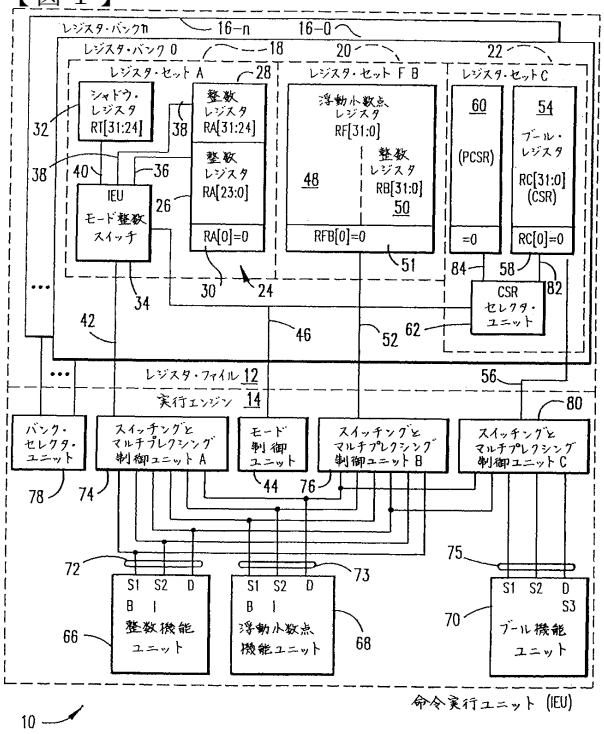

FIG.1

【図２】

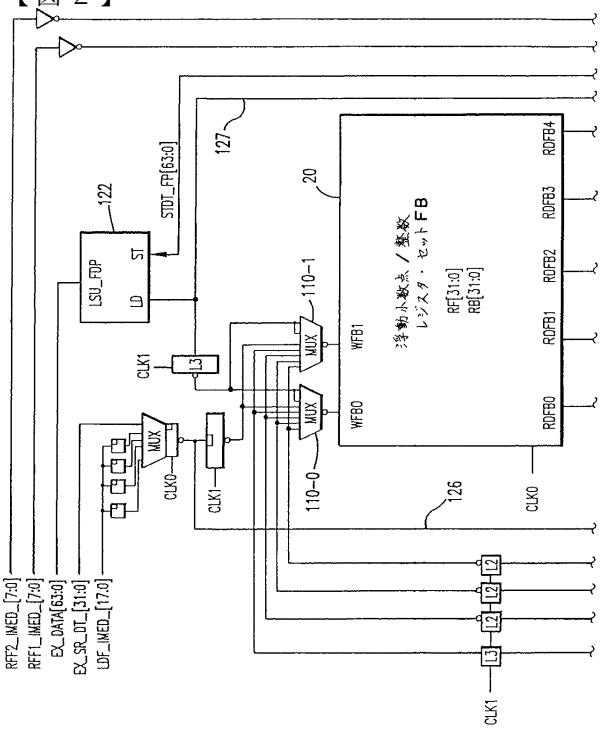

FIG.2

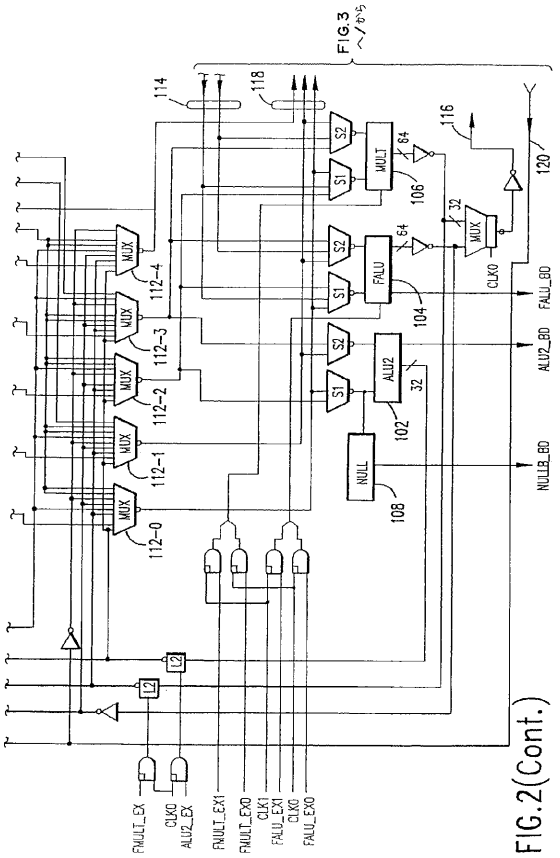

FIG.2(Cont.)

【図３】

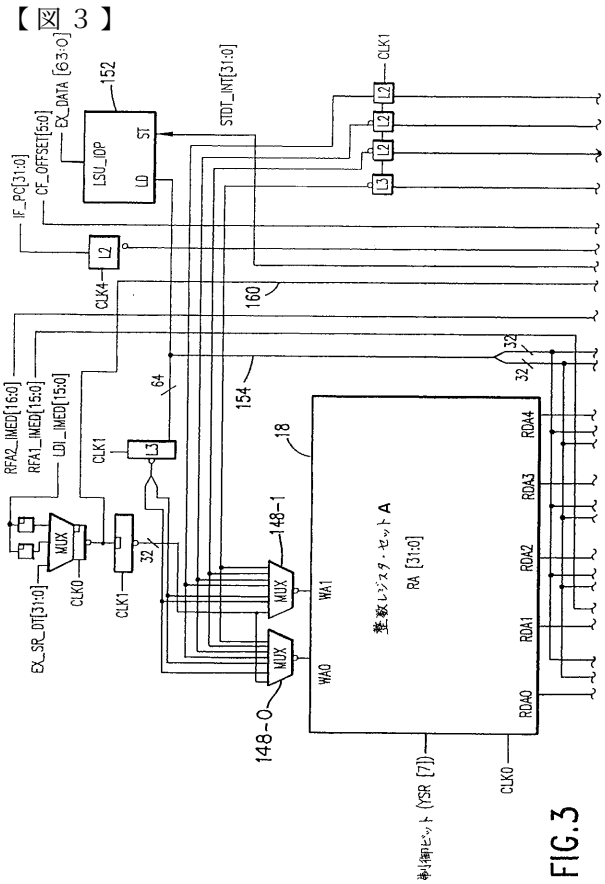

FIG.3

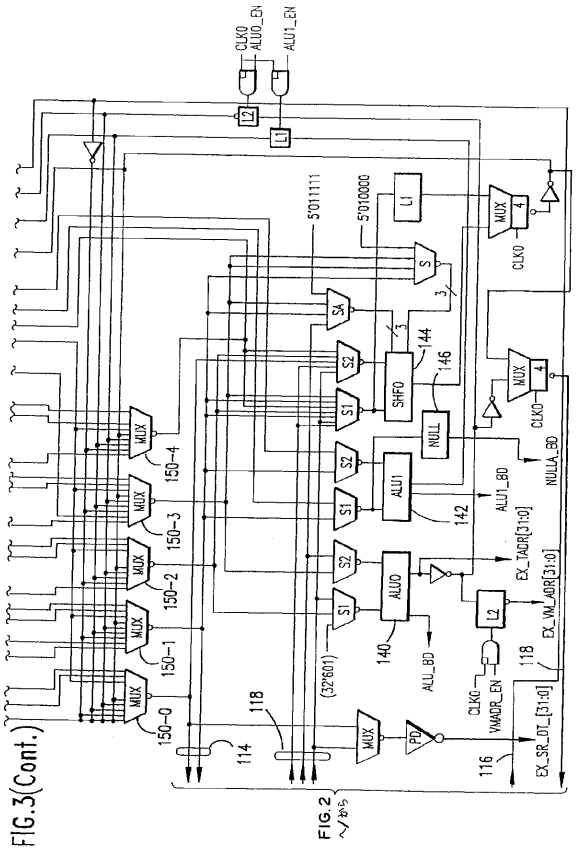

FIG.3(Cont.)

【図４】

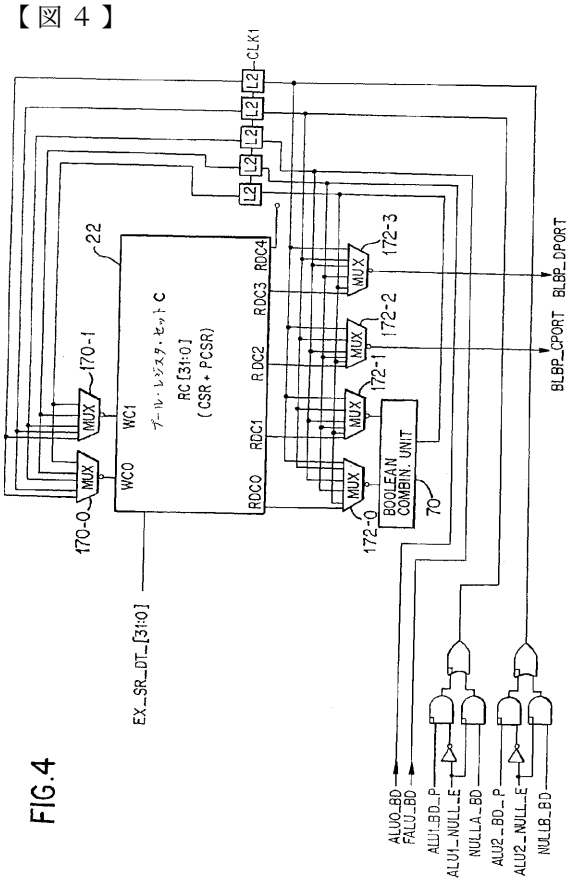

FIG.4

【図５】

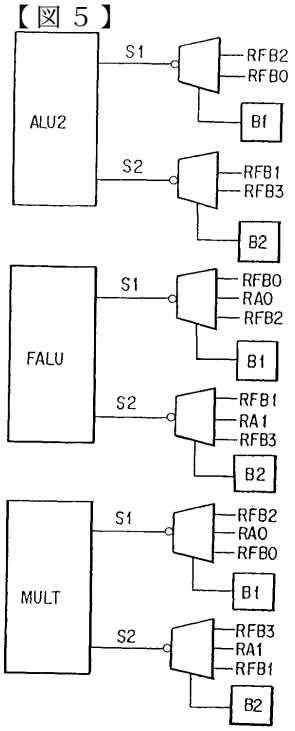

FIG.5

【図６】

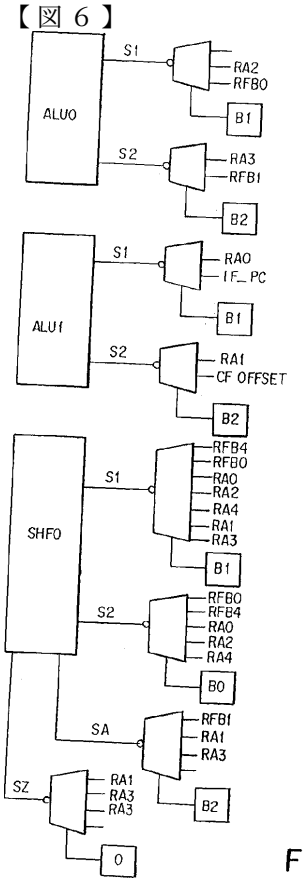

FIG.6

【図７】

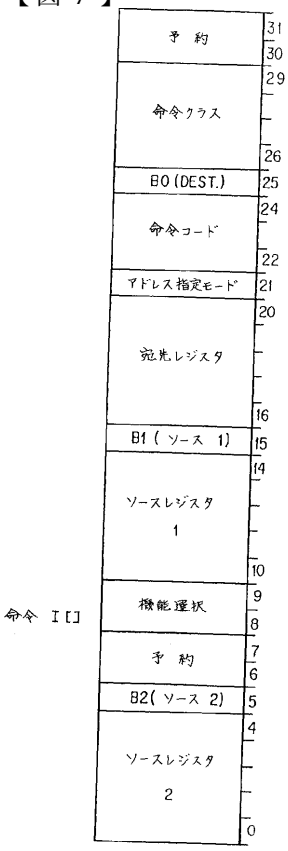
予 約
31
30
命令クラス
29
26
B0 (DEST.)
25
命令コード
24
22
アドレス指定モード
21
宛先レジスタ
20
16
B1 (ソース 1)
15
ソースレジスタ
1
14
10
機能運択
9
8
予 約
7
6
B2(ソース 2)
5
ソースレジスタ
2
4
0
命令 I[]

FIG. 7

フロントページの続き

(74)代理人
弁理士 韮澤 弘
(74)代理人
弁理士 米澤 明
(72)発明者 ガルグ, サンジブ
アメリカ合衆国 ９４５３９ カリフォルニア州 フリーモント センティネル ドライブ ４６８２０
(72)発明者 レンツ, デレク ジェイ.
アメリカ合衆国 ９５０３２ カリフォルニア州 ロス ゲイトス フィリップス アヴェニュー １７４００
(72)発明者 グエン, レ トロン
アメリカ合衆国 ９５０３０ カリフォルニア州 モンテ セレノ ダニエル プレイス １５０９６
(72)発明者 チェン, ショ ロン
アメリカ合衆国 ９５０７０ カリフォルニア州 サラトガ キート ロード １４４１１

審査官 後藤 彰

(56)参考文献 特開平２－２２６３４２（ＪＰ，Ａ）
特開平２－１１８７５７（ＪＰ，Ａ）
特開昭６２－２４２２４３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 9/34
G06F 9/46