(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3786575号
(P3786575)
(45)発行日　平成18年6月14日(2006.6.14)　(24)登録日　平成18年3月31日(2006.3.31)

(51)Int.Cl.　F I
G06F 9/42 (2006.01)　G06F 9/42 330A

請求項の数 4　(全 25 頁)

(21)出願番号　特願2000-357619(P2000-357619)
(22)出願日　平成12年11月20日(2000.11.20)
(65)公開番号　特開2002-157115(P2002-157115A)
(43)公開日　平成14年5月31日(2002.5.31)
審査請求日　平成16年3月26日(2004.3.26)

(73)特許権者　503121103
株式会社ルネサステクノロジ
東京都千代田区丸の内二丁目４番１号
(74)代理人　100089071
弁理士　玉村　静世
(74)代理人　100075096
弁理士　作田　康夫
(72)発明者　三ツ石　直幹
東京都小平市上水本町五丁目２０番１号　株式会社日立製作所半導体グループ内

審査官　後藤　彰

最終頁に続く

(54)【発明の名称】データ処理装置

(57)【特許請求の範囲】
【請求項１】
所定の命令を順次実行するデータ処理装置であって、
プログラムカウンタとスタックとレジスタ番号で指定できる複数のレジスタとを有し、
前記スタックは、データの書き込みと読み出しが可能であって、最後に格納された内容から順次読み出しを行うことができるデータ記憶手段であり、
前記データ処理装置は、前記プログラムカウンタで指定されるアドレスに格納された命令を実行し、前記プログラムカウンタの値を更新するものであり、
前記データ処理装置は、前記プログラムカウンタの内容を前記スタックに書き込む第１の命令と、前記複数のレジスタの中の指定されたレジスタの内容を前記スタックに書き込む第２の命令と、前記スタックから読み出した内容を前記プログラムカウンタに書き戻すことができるとともに前記複数のレジスタのうち前記スタックから内容を書き戻すべきレジスタの数とレジスタ番号を指定することができる第３の命令と、前記スタックから読み出した内容を前記プログラムカウンタに書き戻すことができるとともに前記複数のレジスタのうち前記スタックから内容を書き戻すべきレジスタの数とレジスタ番号を指定することができる第４の命令とを実行可能であり、
前記第３の命令と前記第４の命令は、指定できるレジスタの数の取り得る値が異なることを特徴とするデータ処理装置。
【請求項２】
請求項１記載のデータ処理装置であって、

レジスタ選択のための算術演算手段を備え、
前記第３の命令は、前記スタックから順次読み出した複数の内容を、前記レジスタの内、前記第３の命令の命令コード中に指定された値とこの値を用いて前記算術演算手段によって演算された値とに基づいて指定される複数のレジスタに書込むことができることを特徴とするデータ処理装置。

【請求項３】
請求項１又は２に記載のデータ処理装置であって、
前記データ処理装置は、さらに第５の命令を実行可能であり、
前記第５の命令は、前記複数のレジスタのうち前記スタックから内容を復帰すべきレジスタ番号を指定することができることを特徴とするデータ処理装置。

【請求項４】
請求項１乃至３の何れか１項に記載のデータ処理装置であって、
前記データ処理装置は、さらに復帰命令を実行可能であり、
前記復帰命令は、前記スタックから読み出した内容を前記プログラムカウンタに書き戻すことができることを特徴とするデータ処理装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、データ処理装置に関し、特に、半導体集積回路装置によって構成されるマイクロコンピュータにかかり、これらの中央処理装置（ＣＰＵ）など、及びこれらを用いたデータ処理システム、更に、この開発装置に利用して有効な技術に関するものである。

【０００２】

【従来の技術】

半導体集積回路装置の製造技術の高度化に伴って、半導体単結晶からなるシングルチップに、中央演算処理装置（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ；以下、単にＣＰＵと称する）、プログラムを格納するＲＯＭ（ＲｅａｄＯｎｌｙ　Ｍｅｍｏｒｙ）、書き替え可能に各種データを格納するＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ）等を含む構成素子を集積して製造した、マイクロコンピュータが広範囲に普及してきており、種々の目的のデータ処理装置として使用されてきている。このマイクロコンピュータは、ＣＰＵが同時に処理し得る情報の量によって性能が異なり、例えば４ビット、８ビット、１６ビット、３２ビット等のマイクロコンピュータとして区分されている。

【０００３】

このようなマイクロコンピュータは、アドレス空間の拡張や、命令セットの拡大、高速化等が図られてきている。また、ＣＰＵは、ソフトウェアによってその性能が定義されているから、前記のようにアドレス空間の拡張や、命令セットの拡大、高速化等を図ったマイクロコンピュータにおいても、既存のマイクロコンピュータのソフトウェア資産を有効に利用できることが望ましい。

【０００４】

このため、オブジェクトレベルで互換性を保ちつつ、アドレス空間の拡張や、命令セットの拡大、高速化等を実現した例として、例えば本発明者が先に提案した特開平６－５１９８１号公報がある。これは１６ビット単位の可変長命令で、いわゆる汎用レジスタ型、或いはロードストア型アーキテクチャを採用することが示されている。

【０００５】

プログラム用のメモリであるＲＯＭを内蔵した、シングルチップ型のものにあっては、内蔵ＲＯＭの容量が、外部にメモリを接続するのに比較して、少ないから、プログラム容量を削減することが望ましい。

【０００６】

１６ビット単位の可変長命令を採用した場合、オペレーションコードのみを持ち、実効アドレスの指定を行わず、かつ有意味な命令として、復帰（ＲＴＳ／ＲＴＥ）命令がある。かかる復帰命令を１６ビット単位の命令に割当てる場合、１６ビットの命令コードの全て

のビットが利用されていない場合が多い。
【０００７】
例えば、命令コードＨ’５４７０がサブルーチンからの復帰命令であって、Ｈ’５４７１、Ｈ’５４７２、Ｈ’５４７３が未定義（他の命令として定義されていない）とされているような場合、ビット０、ビット１は利用されていない。
【０００８】
一方、本発明者らが先に提案した特開平８－２６３２９０号公報に記載のＣＰＵは、命令を実行する実行手段を制御する制御手段に指定可能な複数の汎用レジスタの組合せを固定にし、複数の汎用レジスタのスタックに対する退避／復帰命令を有し、複数の汎用レジスタを順次待避／復帰するようにしている。汎用レジスタの組合せを固定にすることによって、汎用レジスタの指定に要するビット数を節約できる。
【０００９】
かかる退避／復帰命令は、サブルーチンや例外処理などの、処理の切れ目で、それ以前の処理の状態を保存する目的に用いられ、アドレッシングモードは、スタックポインタのプリデクリメント（待避）／ポストインクリメント（復帰）にが使用できる。
【００１０】
つまり、かかる汎用レジスタの復帰命令は、前記サブルーチンまたは例外処理からの復帰命令（ＲＴＳ／ＲＴＥ）の前に置かれることが多い。
【００１１】
いわゆるアキュムレータ型アーキテクチャのＣＰＵにおいては、サブルーチンへの分岐／サブルーチンからの復帰時に、ＰＣに加えて、アキュムレータの退避／復帰を行う場合がある。これは、データの処理が少数のアキュムレータ上で行われ、サブルーチンのように処理が変わるときには、アキュムレータを空ける必要があるためである。換言すれば、アキュムレータを自動的に退避／復帰を行っても、無駄にならないからである。
【００１２】
これに対して、汎用レジスタ型アーキテクチャのＣＰＵにおいては、汎用レジスタのいずれにデータを置くかについて、自由度が高く、また、サブルーチンへの分岐／サブルーチンからの復帰時の引数が、汎用レジスタに割当てられる場合もある。このため、所定の（全部または一部の）汎用レジスタの退避／復帰を自動的に行うと、無駄になりやすい。不所望の汎用レジスタの退避／復帰によって、実行ステート数や、スタックの使用量が増加してしまうからである。
【００１３】
マイクロコンピュータのプログラムはＣ言語で記述されることが多くなっている。Ｃ言語で多く用いられる関数は、サブルーチンとして実現される。
【００１４】
【発明が解決しようとする課題】
本発明が解決しようとする問題点は、
マイクロコンピュータなどのデータ処理装置、その論理的規模の増加を最小限にしつつ、使い勝手を向上し、プログラム容量を縮小したり、実行ステート数を短縮したりする手段を提供することにある。
【００１５】
特に、Ｃ言語などを使用したプログラムにおいて、関数を多用した場合のプログラム容量の縮小、実行ステート数の短縮を図ることである。
【００１６】
また、既存のソフトウェア資産を有効に利用可能にし、使用者の開発効率を向上するとともに、開発装置を共通に利用可能にし、開発装置の開発効率を向上するとともに、いち速く開発環境を提供可能にすることである。
【００１７】
本発明の前記ならびにそのほかの目的と新規な特長は、本発明書の記述及び添付図面から明らかになるであろう。

【００１８】
【課題を解決するための手段】
本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。
【００１９】
本発明のデータ処理装置は、サブルーチンから、または例外処理からの復帰命令において、当該命令コードに複数の汎用レジスタを指定する情報を含み、当該命令コードの実行時に、プログラムカウンタの復帰に加えて、前記指定した汎用レジスタの復帰を行うようにする。
【００２０】
更に、指定する汎用レジスタの組合せを連続したものにする。
【００２１】
同様の指定を可能にした、汎用レジスタの退避命令を実行可能にする。
【００２２】
上記した手段によれば、
サブルーチンから、または例外処理からの復帰命令と、汎用レジスタの復帰命令を同一の命令（例えば１６ビット）とすることにより、使い勝手を向上し、プログラム容量を短縮できる。
【００２３】
１６ビット単位の可変長の命令コード体系を採る場合にも、命令プリフェッチや命令デコードを簡易にしつつ、命令コードを有効に利用して、全体的なプログラム効率を向上できる。
【００２４】
同様の指定を可能にした、汎用レジスタの退避命令を持ち、これをサブルーチン乃至例外処理ルーチンの先頭で実行し、最後に、本発明の復帰命令を実行することによって、より一層の使い勝手を向上できる。
【００２５】
指定する汎用レジスタの組合せを固定にすることによって、汎用レジスタを指定する情報を短くでき、限られた命令コード（例えば１６ビット）であっても、指定可能にできる。
【００２６】
汎用レジスタの復帰と、プログラムカウンタなどの復帰を連続的に行うことによって、内部の動作を容易にし、高速化できる。
【００２７】
【発明の実施の形態】
図１に、本発明の適用されたマイクロコンピュータのブロック図を示す。
【００２８】
マイクロコンピュータは、シングルチップ型であり、全体の制御を司るＣＰＵ、割込コントローラ（ＩＮＴ）、ＣＰＵの処理プログラムなどを格納するメモリであるＲＯＭ、ＣＰＵの作業領域ならびににデータの一時記憶用及びスタック用のメモリとすることができるＲＡＭ、タイマ、シリアルコミュニケーションインタフェース（ＳＣＩ）、Ａ／Ｄ変換器、第１乃至第９入出力ポート（ＩＯＰ１～ＩＯＰ９）、クロック発振器（ＣＰＧ）の機能ブロック乃至はモジュールから構成され、公知の半導体製造技術により１つの半導体基板上に形成される。
【００２９】
かかるマイクロコンピュータは、電源端子として、グランドレベル（Ｖｓｓ）、電源電圧レベル（Ｖｃｃ）、アナロググランドレベル（ＡＶｓｓ）、アナログ電源電圧レベル（ＡＶｃｃ）、アナログ基準電圧（Ｖｒｅｆ）、の他専用制御端子として、リセット（ＲＥＳ）、スタンバイ（ＳＴＢＹ）、モード制御（ＭＤ０、ＭＤ１）、クロック入力（ＥＸＴＡＬ、ＸＴＡＬ）端子を有する。
【００３０】

ＣＰＧの端子ＥＸＴＡＬ、ＸＴＡＬに接続される水晶発振子またはＥＸＴＡＬ端子に入力れる外部クロックに基づいて生成される基準クロック（システムクロック）に同期して、マイクロコンピュータは動作を行う。この基準クロック１周期をステートと呼ぶ。

【００３１】

マイクロコンピュータの機能ブロックは、内部バスによって相互に接続さる。バスの制御を行う、図示はされないバスコントローラを内蔵している。内部バスはアドレスバス・データバスのほか、リード信号・ライト信号・バスサイズ信号を含み、これらはバスコマンドとしてコード化される。

【００３２】

かかる機能ブロックやモジュールは内部バスを介して、ＣＰＵによってリード／ライトされる。内部バスのデータバス幅は３２ビットとする。内蔵ＲＯＭ、ＲＡＭリード／ライトを１ステートでリード／ライト可能とする。

【００３３】

タイマ、ＳＣＩ、Ａ／Ｄ変換器、ＩＯＰ１～ＩＯＰ９、ＳＹＳＣ、ＩＮＴ、ＣＰＧなどが有する制御レジスタを総称して、内部Ｉ／Ｏレジスタと呼ぶ。

【００３４】

各入出力ポートは、アドレスバス、データバス、バス制御信号或いはタイマ、ＳＣＩ、Ａ／Ｄ変換器の入出力端子と兼用されている。即ち、タイマ、ＳＣＩ、Ａ／Ｄ変換器は、それぞれ入出力信号を有し、入出力ポートと兼用にされた端子介して、外部と入出力されるものである。例えばＩＯＰ５、６は、タイマの入出力端子と兼用、ＩＯＰ７はＳＣＩの入出力端子と兼用にされている。アナログデータの入出力（ＡＩＮ０～ＡＩＮ７）端子は、ＩＯＰ８と兼用にされている。

【００３５】

外部割込み入力は、ＩＯＰ９と兼用にされている。

【００３６】

かかるマイクロコンピュータにリセット信号ＲＥＳが与えられると、ＣＰＵを始めとし、マイクロコンピュータはリセット状態になる。このリセットが解除されると、モード制御で指定された動作モード（シングルチップ、外部バス拡張、内蔵ＲＯＭ有効など）とされ、ＣＰＵは所定のアドレス（ベクタアドレス）からスタートアドレスをリードして、このスタートアドレスから命令のリードを開始するリセット例外処理を行う。この後、ＣＰＵは逐次、ＲＯＭなどから命令をリードし、解読して、その解読内容に基づいて、フラグやビットの判定や操作を含む、データの処理或いはＲＡＭ、タイマ、ＳＣＩ、Ａ／Ｄ変換器、入出力ポートなどとのデータ転送を行う。即ち、ＣＰＵは、タイマ、ＳＣＩ、Ａ／Ｄ変換器、入出力ポートなどから入力されるデータ或いは機器の状態や指示（スイッチ、ボリュームなど）を参照しつつ、ＲＯＭなどに記憶されている命令に基づいて処理を行い、その結果に基づいて、入出力ポート、タイマなどを使用しつつ、外部に信号を出力し、機器の制御を行うものである。

【００３７】

割込みコントローラ（ＩＮＴ）は、タイマ、ＳＣＩ、Ａ／Ｄ変換器、外部入力の各割込み信号を入力して、割込み要求信号をＣＰＵに出力する。

【００３８】

ＣＰＵに割込要求信号が与えられると、ＣＰＵは実行中の処理を中断して、割込み例外処理状態を経て、所定の処理ルーチンに分岐し、所望の処理を行い、割込要因をクリアしたりする。所定の処理ルーチンの最後には、通常復帰命令がおかれ、この命令を実行することによって前記中断した処理を再開する。

【００３９】

割込み例外処理の実行時には、後述のプログラムカウンタ（ＰＣ）やコンディションコードレジスタ（ＣＣＲ）などがＲＡＭなどに退避され、復帰命令実行時に上記プログラムカウンタ（ＰＣ）やコンディションコードレジスタ（ＣＣＲ）などが復帰される。

【００４０】

更に、割込み処理ルーチンでは所要の処理を行うため、汎用レジスタを使用する必要がある。割込み発生直前の状態を保存するため、割込み処理ルーチンの先頭で、所要の汎用レジスタをスタックに退避する。これは、例えば、
ＳＴＭ　ＥＲ０－ＥＲ３，@－ＳＰ
と記述される。割込み処理ルーチンの最後に、退避した汎用レジスタを復帰してから、復帰命令（ＲＴＥ）を実行して、割込み発生直前のプログラムを継続する。これは、従来技術では、例えば、
ＬＤＭ　@ＳＰ＋，ＥＲ０－ＥＲ３
ＲＴＥ
と記述される。

【００４１】
本発明によれば、上記ＬＤＭとＲＴＥ命令を１命令で実行して、
ＲＴＥ／４　ＥＲ０－ＥＲ３
と記述すればよいようにする。

【００４２】
同様に、サブルーチンにおいても同様に、所要の処理を行うため、汎用レジスタを使用する。例えば、サブルーチンの先頭で、
ＳＴＭ　ＥＲ０－ＥＲ１，@－ＳＰ
と記述し、最後に、
ＬＤＭ　@ＳＰ＋，ＥＲ０－ＥＲ１
ＲＴＳ
と記述される。

【００４３】
本発明によれば、上記ＬＤＭとＲＴＥ命令を１命令で実行して、
ＲＴＳ／２　ＥＲ０－ＥＲ１
と記述すればよいようにする。

【００４４】
図２に、ＣＰＵに内蔵されている汎用レジスタ及び制御レジスタの構成例（プログラミングモデル）を示す。

【００４５】
ＣＰＵは、３２ビット長の汎用レジスタを８本持っている。汎用レジスタは、すべて同機能を持っており、アドレスレジスタとしてもデータレジスタとしても使用することができる。

【００４６】
データレジスタとしては３２ビット、１６ビット及び８ビットレジスタとして使用きる。アドレスレジスタ及び３２ビットレジスタとしては、一括して汎用レジスタＥＲ（ＥＲ０～ＥＲ７）として使用する。１６ビットレジスタとしては、汎用レジスタＥＲを分割して汎用レジスタＥ（Ｅ０～Ｅ７）、汎用レジスタＲ（Ｒ０～Ｒ７）として使用する。これらは同等の機能を持っており、１６ビットジスタを最大１６本まで使用することができる。なお、汎用レジスタＥ（Ｅ０～Ｅ７）を、特に拡張レジスタと呼ぶ場合がある。８ビットレジスタとしては、汎用レジスタＲを分割して汎用レジスタＲＨ（Ｒ０Ｈ～Ｒ７Ｈ）、汎用レジスタＲＬ（Ｒ０Ｌ～Ｒ７Ｌ）として使用する。これらは同等の機能を持っており、８ビットレジスタを最大１６本まで使用することができる。各レジスタ独立に使用方法を選択することができる。

【００４７】
汎用レジスタＥＲ７には、汎用レジスタとしての機能に加えて、スタックポインタ（ＳＰ）としての機能が割当てられており、例外処理やサブルーチン分岐などで暗黙的に使用される。例外処理は前記割込み処理を含む。

【００４８】
ＰＣは２４ビットのカウンタで、ＣＰＵが次に実行する命令のアドレスを示す。特に制限

されないもののＣＰＵの命令は、すべて２バイト（ワード）を単位としているため、最下位ビットは無効であり、命令リード時には最下位ビットは０とみなされる。

【００４９】

ＣＣＲは８ビットのレジスタで、割込みマスクビット（Ｉ）とハーフキャリ（Ｈ）、ネガティブ（Ｎ）、ゼロ（Ｚ）、オーバフロー（Ｖ）、キャリ（Ｃ）の各フラグを含む。ハーフキャリ（Ｈ）、ネガティブ（Ｎ）、ゼロ（Ｚ）、オーバフロー（Ｖ）、キャリ（Ｃ）の各フラグは転送命令や演算命令実行時に、データを検査して、その状態を反映する。ハーフキャリ（Ｈ）は１０進補正用にのみ用いられる。

【００５０】

図３に、ＣＰＵの構成例を示す。

【００５１】

ＣＰＵは、制御部と、前記汎用レジスタＥＲ０～ＥＲ３１、プログラムカウンタＰＣ、コンディションコードレジスタＣＣＲを含む実行部から構成される。

【００５２】

制御部は、命令レジスタＩＲ、命令デコーダＤＥＣ、レジスタセレクタＲＥＳＬ、割込受け付け回路ＩＮＴを含む。命令デコーダＤＥＣは、例えば、マイクロＲＯＭ或ＰＬＡ（Ｐｒｏｇｒａｍｍａｂｌｅ Ｌｏｇｉｃ Ａｒｒａｙ）または布線論理で構成される。命令デコーダＤＥＣの出力の一部が命令デコーダＤＥＣにフィードバックされている。これは各命令コード内の遷移に用いるステージコード（ＴＭＧ）と、命令コード間に用いる制御信号（ＭＯＤ）を含む。即ち、前記ＳＴＭ命令の第１ワードは汎用レジスタの本数を指定する制御信号を発生し、割込みマスク信号を出力して、割込みの受付けを禁止する。

【００５３】

割込み制御部ＩＮＴは、後述の図１の割込みコントローラの出力する割込み要求信号を受け付ける。また、命令デコーダの出力する割込みマスク信号を参照して、割込みがマスクされていなければ、命令変更部ＣＨＧに割込みを指示し、割込み例外処理用の命令コードの実行を行わせる。

【００５４】

実行部には、更に、テンポラリレジスタＴＲ、算術論理演算器ＡＬＵ、インクリメンタＩＮＣ、リードデータバッファＤＢＲ、ライトデータバッファＤＢＷ、アドレスバッファＡＢを含む。これらのブロックはＧＢバス、ＤＢバス、ＷＢバスによって相互に接続されている。

【００５５】

算術論理演算器ＡＬＵは、命令によって指定される各種の演算や実効アドレスの計算など用いる。

【００５６】

インクリメンタＩＮＣは、主にＰＣの加算やアドレス計算に用いられる。

【００５７】

ＤＢＲは、ＲＯＭ、ＲＡＭ、内部Ｉ／Ｏレジスタ、或いは図示はされない外部メモリから、リードした命令コードやデータを一時的に格納する。ＤＢＷはＲＯＭ、ＲＡＭ、内部Ｉ／Ｏレジスタ、或いは外部メモリへのライトデータを一時的に格納する。ＤＢＲ、ＤＢＷによってＣＰＵ内部動作と、ＣＰＵ外部のリード／ライト動作のタイミングを調整している。

【００５８】

アドレスバッファＡＢは、ＣＰＵがリード／ライトするアドレスを一時的に格納する。

【００５９】

図４に、レジスタセレクタＲＥＳＬの一部の詳細なブロック図を示す。

【００６０】

レジスタセレクタＲＥＳＬは、レジスタフィールドを保持するＩＲから出力されるレジスタフィールド情報と、命令デコーダＤＥＣの制御信号出力を入力して、各汎用レジスタに対応するレジスタ選択信号を発生する。

【００６１】
レジスタ指定フィールドは２つある（ｒ１、ｒ２）。
【００６２】
第２のレジスタ指定フィールド（ｒ２）は、ＩＲのビット３～０と、命令デコーダの制御信号出力（ｉｎｃ、ｄｅｃ）から生成される。
【００６３】
上記を入力したインクリメンタ（ＲＩＮＣ）の出力が、システムクロックの反転φ＃でラッチされ、第２のレジスタ指定フィールド（ｒ２）となる。
【００６４】
例えば、
ＳＴＭ　ＥＲ０－ＥＲ３，@－ＳＰ
の場合、ｒ２の初期値として、０が与えられる。その後、後述の所定のタイミングで、インクリメント信号（ｉｎｃ）によって、ｒ２は０→１→２→３とされる。
【００６５】
また、
ＬＤＭ　@ＳＰ＋，ＥＲ０－ＥＲ１
または、
ＲＴＳ／２　ＥＲ０－ＥＲ１
の場合、ｒ２の初期値として、１が与えられる。その後、後述の所定のタイミングで、インクリメント信号（ｄｅｃ＝ｒｄ－－）によって、ｒ２は１→０とされる。
【００６６】
これによって、レジスタ選択回路自体をそのほかの命令と共通化することが、論理的規模の増加を抑止できる。
【００６７】
ｒ２のビット３は、これらの命令では使用しないため、インクリメンタ（ＲＩＮＣ）は３ビットでよい。インクリメンタ（ＲＩＮＣ）を持つことにより、さほど論理規模を大きくすることなく、連続した任意の汎用レジスタの組合せを指定可能にできる。
【００６８】
なお、その他のＲＥＳＬの詳細については省略する。
【００６９】
図５に、複数汎用レジスタの復帰を含む第１の復帰命令（ＲＴＳ／ｎ）、第２の復帰命令（ＲＴＥ／ｎ）の命令フォーマットを示す。
【００７０】
第１の復帰命令（ＲＴＳ／ｎ）は、サブルーチンからの復帰命令であり、第２の復帰命令（ＲＴＥ／ｎ）は、例外処理ルーチンからの復帰命令である。
【００７１】
ビット１５～７はオペレーションフィールドであり、復帰命令を指定する。
【００７２】
ビット６が０のとき、ビット５、４によって、汎用レジスタの本数を指定する。０１のときは２本、１０のときは３本、１１のときは４本とする。汎用レジスタの選択は、連続した２、３、４本の選択が可能である。例えば、４本の場合には、ＥＲ０－３、ＥＲ１－４、ＥＲ２－５、ＥＲ３－６、が指定可能である。ＥＲ７は、スタックポインタと兼用であり、対象としない。
【００７３】
ビット６が１のときは、汎用レジスタの復帰を伴わない。即ち、従来技術の復帰命令と同一であり、共通の命令コード（Ｈ’５４７０）とする。
【００７４】
ビット３～０で最初に使用する汎用レジスタを指定する。ビット３は予約されているものとし、０とする。
【００７５】

例えば、ＲＴＳ／４　ＥＲ０－３のときは、ＥＲ３から復帰されるので、Ｈ’５４３３の命令コードとなる。

【００７６】

第２の復帰命令（ＲＴＥ／ｎ）も、オペレーションフィールドが異なるのみであり、前記同様である。

【００７７】

ＳＴＭ命令の命令フォーマットは、２ワード命令であり、第２ワードのビット３～０で最初に使用する汎用レジスタを指定する。これは上記復帰命令と共通化されている。

【００７８】

ＬＤＭ命令の命令フォーマットも同様である。

【００７９】

図６に、複数汎用レジスタの復帰を含む第１の復帰命令（ＲＴＳ／ｎ）、第２の復帰命令（ＲＴＥ／ｎ）のスタックの構成の例を示す。

【００８０】

スタックはスタックポインタ（ＳＰ：ＥＲ７）によって、スタックされた最後のアドレスが指示されている。

【００８１】

サブルーチン分岐命令（例えば、ＢＳＲ命令）によって、プログラムカウンタがスタックに退避される。その後、サブルーチンの先頭で、所要の汎用レジスタが退避される（例えば、ＳＴＭ　ＥＲ０－ＥＲ１，@－ＳＰ）。

【００８２】

これに対応して、第１の復帰命令（例えば、ＲＴＳ／２　ＥＲ０－ＥＲ１）は、まず、指定した汎用レジスタの復帰として、スタックからのリードを行い、指定した汎用レジスタ（ＥＲ１、ＥＲ０）に順次格納する。引続き、プログラムカウンタの復帰として、スタックからのリードを行い、プログラムカウンタに格納し、この内容に従って、命令のリードを行う。

【００８３】

同様に、例外処理によって、プログラムカウンタとコンディションコードレジスタがスタックに退避される。プログラムカウンタが２４ビットであり、コンディションコードレジスタと組合せて、スタックに退避される。その後、サブルーチンの先頭で、所要の汎用レジスタが退避される（例えば、ＳＴＭ　ＥＲ０－ＥＲ３，@－ＳＰ）。

【００８４】

これに対応して、第２の復帰命令（例えば、ＲＴＥ／４　ＥＲ０－ＥＲ３）は、まず、指定した汎用レジスタの復帰として、スタックからのリードを行い、指定した汎用レジスタ（ＥＲ３、ＥＲ２、ＥＲ１、ＥＲ０）に順次格納する。引続き、プログラムカウンタとコンディションコードレジスタの復帰として、スタックからのリードを行い、プログラムカウンタとコンディションコードレジスタに所定の内容を格納するとともに、命令のリードを行う。

【００８５】

図７に、第１の復帰命令（ＲＴＳ／ｎ）の動作フローを示す。

【００８６】

ステップ１で、制御信号Ａで、スタックポインタ（ＳＰ）の内容がアドレスバス（ＩＡＢ）に出力され、ロングワードのリードが指示される。また、ＳＰの内容のインクリメント（＋４）が指示される。制御信号Ｃで、リードしたデータのＤＢＲへの格納が指示される。

【００８７】

ステップ２～５で、制御信号Ｂで、前のステップでリードした内容の汎用レジスタへの書込みが指示される。具体的には、一旦、内部バスＧＢに出力し、ＡＬＵを経由して、内部バスＷＢに出力し、指定された汎用レジスタに書込まれる。レジスタセレクタＲＥＳＬに対してレジスタフィールドのデクリメント（－１）が指示される。前記同様に、制御信

号Ａで、スタックポインタ（ＳＰ）の内容のアドレスバス（ＩＡＢ）への出力と、ロングワードのリードが指示され、ＳＰの内容のインクリメント（＋４）が指示される。制御信号Ｃで、リードしたデータのＤＢＲへの格納が指示される。ステップ２～４は、指定した汎用レジスタの本数（ｎ）に従って、所定回数が実行される。

【００８８】

ステップ６で、制御信号Ｂで、前のステップでリードした内容のプログラムカウンタ（ＰＣ）への書込みが指示される。

【００８９】

ステップ７で、制御信号Ａで、命令リードとして、プログラムカウンタ（ＰＣ）の内容がアドレスバス（ＩＡＢ）に出力され、メモリワードリードが指示される。また、ＰＣの内容のインクリメント（＋２）が指示される。制御信号Ｃで、リードされた命令をＩＲに格納する。

【００９０】

ステップ８で、同様に、制御信号Ａで、プログラムカウンタ（ＰＣ）の内容のアドレスバス（ＩＡＢ）への出力と、メモリワードリードが指示され、ＰＣの内容のインクリメント（＋２）が指示される。制御信号Ｂで、前のステップでリードした命令（ＩＲ）の命令デコーダへの入力が指示される。次の命令の実行が開始される。

【００９１】

ＰＣの復帰時、スタックからリードした内容を、一旦、ＰＣに格納するものとしたが、これを行わず、ステップ７で、ＰＣの代わりに、ＤＢＲから読み出すようにしてもよい。

【００９２】

汎用レジスタの復帰を行わない復帰命令の場合は、ステップ２～５をスキップすればよい。換言すれば、従来技術の復帰命令に、ステップ２～５を追加すればよい。

【００９３】

かかる動作フローに従って、ハードウェア記述言語（ＨＤＬ）を記述し、命令デコーダＤＥＣを生成することができる。第１の復帰命令（ＲＴＳ／ｎ）に関する部分は、以下の通りである。

【００９４】

```
casex(opcode)                                    (1)
16'b01010100_0001????: begin                     (2)
 case(tmg)                                       (3)
 4'b0001: begin                                  (4)
 nexttmg = 4'b0100;                              (5)
 bcmd    = long_read;                            (6)
 ...
   end                                           (7)
 4'b0100: begin                                  (8)
 nexttmg = 4'b0101;                              (9)
 bcmd    = long_read;                            (10)
 ...
   end                                           (11)
 ...
  end                                            (12)
16'b01010100_0010????: begin                     (13)
 case(tmg)                                       (14)
 4'b0001: begin                                  (15)
 nexttmg = 4'b0010;                              (16)
 ...
 4'b0010: begin                                  (17)
 nexttmg = 4'b0100;                              (18)
 ...
```

本記述は、ＩＥＥＥ１３６４として標準化されているＨＤＬ記述言語に従っている。かかるＨＤＬ記述言語自体については、１９９６年７月ＣＱ出版株式会社発行の『入門Ｖｅｒｉｌｏｇ－ＨＤＬ記述』などに記載されている。

【００９５】

ｃａｓｅｘ及びｃａｓｅは、それに続く（）内の変数（ｏｐｃｏｄｅ、ｔｍｇ）に応じて、後続の値（１６’ｂ０１０１０１００＿０００１？？？？：、４’ｂ０００１：など）に一致した動作を実行する。ｂｅｇｉｎ～ｅｎｄで１つの動作が記述される。

【００９６】

（１）で、命令デコーダの入力（ｏｐｃｏｄｅ）による条件を指示する。当該条件で、（２）～（１２）でＲＴＳ／２に相当する命令コードの動作が記述される。更に、各命令コード毎に、ステップに相当する命令デコーダの入力（ｔｍｇ）による条件を指示する。ＲＴＳ／２の第１ステップの動作は、（４）～（７）に記述される。（５）で、次のステップ（ｎｅｘｔｔｍｇ）が第４ステップであることが指定される。ｎｅｘｔｔｍｇは１ステート後に、ｔｍｇとして命令デコーダに入力される。（６）で、バスコマンドがロングワードリードとされる。ｌｏｎｇ＿ｒｅａｄはパラメータとして、所定のコードが割当てられるようにしておく。この後に、そのほかの制御信号が記述される。同様に、第４ステッ

プの動作は、（８）～（１１）に記述される。第５～第８ステップの動作は、これ以降（１２）までに記述される。

【００９７】

同様に、命令デコーダの入力（ｏｐｃｏｄｅ）による条件で、（１３）以降でＲＴＳ／３に相当する命令コードの動作が記述される。第１ステップの動作記述において、（１５）で、次のステップ（ｎｅｘｔｔｍｇ）が第２ステップであることが指定される。また、第２ステップの動作は、（１７）以降に記述される。更に、第４～第８ステップの動作も記述される。

【００９８】

更に、その後、ＲＴＳ／４に相当する命令コードの動作が記述され、第２の復帰命令（ＲＴＥ／ｎ）、複数レジスタの退避命令（ＳＴＭ）や、その他の命令コードも記述される。

【００９９】

図８に、第１の復帰命令（ＲＴＳ／ｎ）の実行タイミングを示す。

【０１００】

ｎ＝２の場合の例である。

【０１０１】

特に制限はされないものの、内蔵ＲＯＭ、ＲＡＭのリード／ライトを１ステートでリード／ライト可能とし、内蔵ＲＯＭ上にプログラム、内蔵ＲＡＭ上にスタックを配置した場合のタイミングを示す。

【０１０２】

Ｔ０のφ＃（＃は反転論理を示す）で、ＣＰＵのアドレスバッファ（ＭＡＢ）かアドレスがＩＡＢに出力される。また、命令デコーダから、命令フェッチ（ｉｆ）を示す、バスコマンド（ＢＣＭＤ）が出力される。

【０１０３】

Ｔ１のφで、ＩＡＢの内容がＰＡＢに出力され、バスコマンドに基づき、リードサイクルが開始され、ＰＤＢにデータが出力される。φ＃でＰＤＢのリードデータが内部データバスＩＤＢに得られ、これをＴ２のφでＩＲにラッチする。以上の動作は以前の命令の実行の制御によって行われる（プリフェッチ）。ここで、内蔵ＲＯＭ、ＲＡＭは、ＰＡＢ及びＰＤＢに接続されていないが、ＰＡＢ、ＰＤＢ相当の動作をモジュール内で行い、本タイミング図には、このモジュール内の動作を示している。

【０１０４】

直前の命令の実行が終了すると、最も早く命令の実行が開始される場合には、Ｔ２のφで命令コードがＤＥＣに入力されて、命令の内容が解読される。解読結果に従って、制御信号を出力して、各部の制御を行う。命令の一部（レジスタ指定フィード：ｒ）がレジスタセレクトに与えられる。

【０１０５】

前記のステップ１として、Ｔ２のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。ロングワードのリードのバスコマンドが出力される。

【０１０６】

Ｔ３のφで、ＩＮＣでインクリメント（＋４）された結果が、内部バスＷＢを経由してＳＰにライトされる。Ｔ３のφ＃から、スタックからリードしたデータがＩＤＢに出力され、Ｔ４のφで、ＤＢＲに格納される。

【０１０７】

ステップ４として、Ｔ４のφ及びφ＃で、ＤＢＲの内容が内部バスＧＢ、ＡＬＵ、内部バスＷＢを介して、指定（ｒ）した汎用レジスタに書込まれる。レジスタ指定フィールドがデクリメントされる。前記同様に、Ｔ３のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。ロングワードのリードのバスコマンドが出力される。Ｔ４のφで、ＩＮＣでインクリメント（＋４）された結果が、内部バスＷＢを経由してＳＰにライトされる。Ｔ４のφ＃から、スタックからリ

ードしたデータがＩＤＢに出力され、Ｔ５のφで、ＤＢＲに格納される。
【０１０８】
ステップ５として、Ｔ５のφ及びφ＃で、ＤＢＲの内容が内部バスＧＢ、ＡＬＵ、内部バスＷＢを介して、指定（ｒ）した汎用レジスタに書込まれる。レジスタ指定フィールドがデクリメントされる。前記同様に、Ｔ４のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。ロングワードのリードのバスコマンドが出力される。Ｔ５のφで、ＩＮＣでインクリメント（＋４）された結果が、内部バスＷＢを経由してＳＰにライトされる。Ｔ５のφ＃から、スタックからリードしたデータがＩＤＢに出力され、Ｔ６のφで、ＤＢＲに格納される。
【０１０９】
ステップ６として、Ｔ６のφ及びφ＃で、ＤＢＲの内容が内部バスＧＢ、ＡＬＵ、内部バスＷＢを介して、ＰＣに書込まれる。
【０１１０】
ステップ７として、Ｔ６のφ＃で、ＰＣの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。Ｔ７のφで、ＩＮＣでインクリメント（＋２）された結果が、内部バスＷＢを経由してＰＣにライトされる。
【０１１１】
ステップ８として、Ｔ７のφ＃で、ＰＣの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。Ｔ８のφで、ＩＮＣでインクリメント（＋２）された結果が、内部バスＷＢを経由してＰＣにライトされる。Ｔ８のφでリードした命令コードが、ＩＲを経由して、ＤＥＣに入力されて、次の命令が実行される。
【０１１２】
図９に、第２の復帰命令（ＲＴＥ／ｎ）の動作フローを示す。
【０１１３】
第１の復帰命令（ＲＴＳ／ｎ）に対して、ステップ６で、ＰＣとともに、ＣＣＲへの書込みが行われるようにされる。そのほかの動作は同様にできる。
【０１１４】
図１０に、第２の復帰命令（ＲＴＥ／ｎ）の実行タイミングを示す。
【０１１５】
ｎ＝４の場合の例である。
【０１１６】
前記同様に、Ｔ２のφで命令コードがＤＥＣに入力されて、命令の内容が解読される。解読結果に従って、制御信号を出力して、各部の制御を行う。命令の一部（レジスタ指定フィード：ｒ）がレジスタセレクトに与えられる。
【０１１７】
前記のステップ１として、Ｔ２のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。ロングワードのリードのバスコマンドが出力される。
【０１１８】
Ｔ３のφで、ＩＮＣでインクリメント（＋４）された結果が、内部バスＷＢを経由してＳＰにライトされる。Ｔ３のφ＃から、スタックからリードしたデータがＩＤＢに出力され、Ｔ４のφで、ＤＢＲに格納される。
【０１１９】
ステップ２として、Ｔ４のφ及びφ＃で、ＤＢＲの内容が内部バスＧＢ、ＡＬＵ、内部バスＷＢを介して、指定（ｒ）した汎用レジスタに書込まれる。レジスタ指定フィールドがデクリメントされる。前記同様に、Ｔ３のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。ロングワードのリードのバスコマンドが出力される。Ｔ４のφで、ＩＮＣでインクリメント（＋４）された結果が、内部バスＷＢを経由してＳＰにライトされる。Ｔ４のφ＃から、スタックからリードしたデータがＩＤＢに出力され、Ｔ５のφで、ＤＢＲに格納される。

【０１２０】
ステップ３として、Ｔ５のφ及びφ＃で、ＤＢＲの内容が内部バスＧＢ、ＡＬＵ、内部バスＷＢを介して、指定（ｒ）した汎用レジスタに書込まれる。レジスタ指定フィールドがデクリメントされる。前記同様に、Ｔ４のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。ロングワードのリードのバスコマンドが出力される。Ｔ５のφで、ＩＮＣでインクリメント（＋４）された結果が、内部バスＷＢを経由してＳＰにライトされる。Ｔ５のφ＃から、スタックからリードしたデータがＩＤＢに出力され、Ｔ６のφで、ＤＢＲに格納される。
【０１２１】
ステップ４として、Ｔ６のφ及びφ＃で、ＤＢＲの内容が内部バスＧＢ、ＡＬＵ、内部バスＷＢを介して、指定（ｒ）した汎用レジスタに書込まれる。レジスタ指定フィールドがデクリメントされる。前記同様に、Ｔ５のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。ロングワードのリードのバスコマンドが出力される。Ｔ６のφで、ＩＮＣでインクリメント（＋４）された結果が、内部バスＷＢを経由してＳＰにライトされる。Ｔ６のφ＃から、スタックからリードしたデータがＩＤＢに出力され、Ｔ７のφで、ＤＢＲに格納される。
【０１２２】
ステップ５として、Ｔ７のφ及びφ＃で、ＤＢＲの内容が内部バスＧＢ、ＡＬＵ、内部バスＷＢを介して、指定（ｒ）した汎用レジスタに書込まれる。レジスタ指定フィールドがデクリメントされる。前記同様に、Ｔ６のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。ロングワードのリードのバスコマンドが出力される。Ｔ７のφで、ＩＮＣでインクリメント（＋４）された結果が、内部バスＷＢを経由してＳＰにライトされる。Ｔ７のφ＃から、スタックからリードしたデータがＩＤＢに出力され、Ｔ６のφで、ＤＢＲに格納される。
【０１２３】
ステップ６として、Ｔ６のφ及びφ＃で、ＤＢＲの内容が内部バスＧＢ、ＡＬＵ、内部バスＷＢを介して、ＣＣＲとＰＣに書込まれる。ＤＢＲのビット３１～２４がＣＣＲに、ビット２３～０がＰＣに書込まれる。
【０１２４】
ステップ７として、Ｔ８のφ＃で、ＰＣの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。Ｔ９のφで、ＩＮＣでインクリメント（＋２）された結果が、内部バスＷＢを経由してＰＣにライトされる。
【０１２５】
ステップ８として、Ｔ９のφ＃で、ＰＣの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。Ｔ１０のφで、ＩＮＣでインクリメント（＋２）された結果が、内部バスＷＢを経由してＰＣにライトされる。Ｔ１０のφでリードした命令コードが、ＩＲを経由して、ＤＥＣに入力されて、次の命令が実行される。
【０１２６】
図１１に、複数汎用レジスタの退避命令（ＳＴＭ）の動作フローを示す。
【０１２７】
ステップ０は、命令コードの第１ワードで制御される。制御信号Ａで、プログラムカウンタ（ＰＣ）の内容のアドレスバス（ＩＡＢ）への出力と、メモリワードリードが指示され、ＰＣの内容のインクリメント（＋２）が指示される。制御信号Ｂで、命令コードの第２ワードの命令デコーダへの入力と、第１ワードで指定される汎用レジスタ本数の情報（ＭＯＤ）が指示される。
【０１２８】
ステップ１で、制御信号Ａで、スタックポインタ（ＳＰ）の内容のデクリメント（－４）が指示される。ステップ２～５で、制御信号Ａで、スタックポインタ（ＳＰ）の内容のアドレスバス（ＩＡＢ）への出力と、ロングワードのライトが指示される。制御信号Ｂで

、汎用レジスタの内容のＤＢＷへの格納が指示される。制御信号Ｃで、ＤＢＷの内容のＩＤＢへの出力が指示される。レジスタセレクタＲＥＳＬに対してレジスタフィールドのインクリメント（－１）が指示される。ステップ２～４は、ＳＰの内容のデクリメント（－４）が指示される。ステップ２～５は、指定した汎用レジスタの本数（ｎ）に従って、所定回数が実行される。

【０１２９】

ステップ６で、制御信号Ａで、プログラムカウンタ（ＰＣ）の内容のアドレスバス（ＩＡＢ）への出力と、メモリワードリードが指示され、ＰＣの内容のインクリメント（＋２）が指示される。制御信号Ｂで、前のステップでリードした命令（ＩＲ）の命令デコーダへの入力が指示される。次の命令の実行が開始される。

【０１３０】

ステップ２～５において、汎用レジスタの内容を、ＤＢＷに転送後、当該汎用レジスタの内容を０にクリアするようにするとよい。汎用レジスタを退避した後、０にクリアし、その内容を確定することで、その後の処理を簡略化することができる。

【０１３１】

例えば、その後、汎用レジスタに０を代入する場合、ＭＯＶ命令を省略できる。また、１などを代入する場合は、

ＭＯＶ．Ｌ　＃１，ＥＲｎ

を使用する代わりに、

ＭＯＶ．Ｂ　＃１，ＲｎＬ

を使用することができる。イミディエイトデータは、そのデータサイズに対応した領域を命令コードに持つから、ＭＯＶ．Ｌの命令コード長（例えば３ワード）よりも、ＭＯＶ．Ｂの命令コード長（例えば１ワード）の方が短い。これによって、プログラム容量の短縮に寄与できる。

【０１３２】

なお、プログラム中に使用する定数値は、０や１などの比較的小さい値が多いことが一般的である。

【０１３３】

図１２に、複数汎用レジスタの退避命令（ＳＴＭ）の実行タイミングを示す。

【０１３４】

ｎ＝２の場合の例である。

【０１３５】

Ｔ２のφで命令コードの第１ワード（ｓｔｍ－１）がＤＥＣに入力されて、命令の内容が解読される。解読結果に従って、制御信号を出力して、各部の制御を行う。命令の一部（レジスタ指定フィード：ｒ）がレジスタセレクトに与えられる。

【０１３６】

ステップ０として、Ｔ２のφ＃で、ＰＣの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。Ｔ２のφで、ＩＮＣでインクリメント（＋２）された結果が、内部バスＷＢを経由してＰＣにライトされる。Ｔ３のφでリードした命令コードが、ＩＲを経由して、ＤＥＣに入力されて、次の命令が実行される。

【０１３７】

Ｔ３のφで命令コードの第２ワード（ｓｔｍ－２）がＤＥＣに入力されて、命令の内容が解読される。解読結果に従って、制御信号を出力して、各部の制御を行う。命令の一部（レジスタ指定フィード：ｒ）がレジスタセレクトに与えられる。

【０１３８】

前記のステップ１として、Ｔ３のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＩＮＣに入力する。Ｔ３のφで、ＩＮＣでデクリメント（－４）された結果が、内部バスＷＢを経由してＳＰにライトされる。

【０１３９】

ステップ２として、Ｔ４のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢとＩＮ

Ｃに入力する。ＡＢからアドレスＩＡＢが出力される。ロングワードのライトのバスコマンドが出力される。Ｔ５のφで、ＩＮＣでデクリメント（－４）された結果が、内部バスＷＢを経由してＳＰにライトされる。また、指定（ｒ）した汎用レジスタの内容が内部バスＤＢを介してＤＢＷに書込まれる。レジスタ指定フィールドがインクリメントされる。Ｔ５のφ＃から、ＤＢＷの内容がＩＤＢに出力される。

【０１４０】

ステップ５として、Ｔ５のφ＃で、ＳＰの内容を内部バスＧＢに読み出して、ＡＢを介してアドレスＩＡＢが出力される。ロングワードのライトのバスコマンドが出力される。Ｔ５のφで、指定（ｒ）した汎用レジスタの内容が内部バスＤＢを介してＤＢＷに書込まれる。レジスタ指定フィールドがインクリメントされる。Ｔ５のφ＃から、ＤＢＷの内容がＩＤＢに出力される。

【０１４１】

ステップ６として、Ｔ６のφ＃で、ＰＣの内容を内部バスＧＢに読み出して、ＡＢとＩＮＣに入力する。ＡＢからアドレスＩＡＢが出力される。Ｔ７のφで、ＩＮＣでインクリメント（＋２）された結果が、内部バスＷＢを経由してＰＣにライトされる。プリフェッチした命令コード（ｎｅｘｔ）が、ＩＲを経由して、ＤＥＣに入力されて、次の命令が実行される。

【０１４２】

前記の通り、ステップ２～５において、Ｔ５、Ｔ６のφ＃で、汎用レジスタの内容を０にクリアする。これは、当該タイミングの内部バスＷＢを０とし、これを汎用レジスタに書込むようにするとよい。

【０１４３】

図１３に、本発明のＣＰＵの開発環境の概略を示す。

【０１４４】

使用者は、各種エディタなどを用いて、Ｃ言語乃至アセンブリ言語でプログラムを作成する。これは通常、複数のモジュールに分割して作成される。

【０１４５】

Ｃコンパイラは、使用者の作成したそれぞれのＣ言語ソースプログラムを入力し、アセンブリ言語ソースプログラム乃至オブジェクトモジュールを出力する。

【０１４６】

例えば、Ｃ言語でプログラムを作成する場合、

```
void  seq0 (void)
 {
...
}
```

と記述された関数ｓｅｑ０は、本発明のＣＰＵに対しては、以下のようにコンパイルされる。

【０１４７】

```
_seq0:
   STM      ER3－4, @－SP
      ...
      RTS／2     ER3－4
```

関数内で使用する汎用レジスタＥＲ３、ＥＲ４をＳＴＭ命令でスタックに退避し、０にクリアされる。復帰命令時にスタックから回復される。

【０１４８】

また、メインプログラムで上記関数ｓｅｑ０を呼び出す場合は、

ｓｅｑ０（）；
と記述され、
ＪＳＲ　＠＿ｓｅｑ０
とサブルーチン分岐命令にコンパイルされる。
【０１４９】
アセンブラは、アセンブリ言語ソースプログラムを入力し、オブジェクトモジュールを出力する。
【０１５０】
リンケージエディタは、上記Ｃコンパイラやアセンブラの生成した、複数のオブジェクトモジュールを入力して、各モジュールの外部参照や相対アドレスなどの解決を行い、１つのプログラムに結合して、ロードモジュールを出力する。
【０１５１】
ロードモジュールは、シミュレータ／デバッガに入力して、パーソナルコンピュータなどのシステム開発装置上で、ＣＰＵの動作をシミュレーションし、実行結果を表示し、プログラムの解析や評価を行うことができる。また、エミュレータに入力して、実際の応用システム上などで動作する、いわゆるインサーキットエミュレーションを行ない、マイクロコンピュータ全体としての、実動作の解析や評価を行うことができる。
【０１５２】
更に、ロードモジュールをＰＲＯＭライタなどのＲＯＭ書込み装置に入力して、マイクロコンピュータの内蔵ＲＯＭがフラッシュメモリなどの場合や、外部のフラッシュメモリなどに、作成したプログラムをロードすることができる。或いはマイクロコンピュータの製造工程で、内蔵ＲＯＭに書込むことも可能である。必要に応じて、オブジェクトコンバータなどによって、所望のフォーマットに変換する。
【０１５３】
このほかに、ライブラリアンとして、汎用的なサブルーチンなどを提供することもできる。
【０１５４】
ＣＰＵ定義情報ファイルは、コンパイル対象のＣＰＵを指定し、本発明の復帰命令を持たない（既存の）ＣＰＵと、本発明のＣＰＵとを共通の開発環境を利用可能にする。
【０１５５】
例えば、ＣＰＵ定義情報ファイルに、
ＳＥＴ　ＣＰＵ＝　ＣＰＵ１；
と記述することによって、本発明のＣＰＵを選択するようにする。
【０１５６】
コンパイル時に、かかるＣＰＵ定義情報ファイルを参照し、本発明の復帰命令を持たないＣＰＵが選択されていれば、前記関数は、

```
_ｓｅｑ０：
        ＳＴＭ       ＥＲ３－４，＠－ＳＰ
        ＭＯＶ．Ｌ    ＃０，ＥＲ３
        ＭＯＶ．Ｌ    ＃０，ＥＲ４
        ・・・
        ＬＤＭ      ＠ＳＰ＋，ＥＲ３－４
        ＲＴＳ
```

となる。ＭＯＶ．Ｌは、当該関数の処理内容に従って、変更になる場合もある。
【０１５７】
Ｃコンパイラ自体には、Ｃ言語によるプログラムを、ＣＰＵの命令に変換する機能のほか

、Ｃ＋＋言語によるプログラムのコンパイルや、モジュール間最適化などといった、ＣＰＵの命令セットとは直接関係のない機能の向上が図られているが、ＣＰＵ毎の個別のコンパイラでは、これらの機能向上を全ての個別のコンパイラに適用しなければならない。本発明のように、共通のＣコンパイラとしておけば、前記、ＣＰＵの命令セットとは直接関係のない機能向上を図ることが容易になり、また、開発効率などを向上することができる。

【０１５８】

上記実施例によれば、以下の作用効果を得るものである。

【０１５９】

単一の命令で、複数の汎用レジスタと復帰動作と、サブルーチンまたは例外処理ルーチンからの復帰動作を行うことによって、相対的に、命令コード長を短縮（例えば、２ワード短縮）し、命令のリード回数を低減して、高速化することができる。

【０１６０】

例えば、Ｃ言語などで多用される関数はサブルーチンとされるから、この数に対応する数のサブルーチンからの復帰命令が使用されることになる。関数毎に必要なデータが汎用レジスタに割当てられるから、それ以前の汎用レジスタの内容の退避／復帰を伴う。従って、かかる復帰命令による命令コード長の短縮効果も大きくなる。

【０１６１】

また、関数毎に、退避／復帰する汎用レジスタ本数も異なるから、レジスタの本数の異なる命令を複数命令サポートすることによって、或いは、汎用レジスタの復帰を伴わない復帰命令もサポートして、プログラムの作成を容易にし、使い勝手を向上することができる。

【０１６２】

ＣＰＵの有する汎用レジスタ本数が多い場合にも、復帰する汎用レジスタを指定可能にしているから、汎用レジスタの使用方法の自由度を損なうことが少ない。また、不所望の汎用レジスタの退避／復帰を行ったりするこがない。

【０１６３】

複数の汎用レジスタを指定する固定の組合わせにすることによって、命令コード長を短縮でき、更に、各命令の実行ステート数を固定にすることにより、内部の条件分岐を行うことを少なくし（命令デコーダの入力ｏｐｃｏｄｅに応じて、デコーダを記述すればよい）、内部論理を簡潔にし、論理規模を縮小できる。マイクロプログラムによらず、ワイアードロジックなどでも容易に実現できる。マイクロプログラムによらず、ワイアードロジックなどとすることにより、論理回路の高速化に寄与することができる。レジスタセレクタにインクリメンタを持つことによって、論理的規模の増加を最小限にして、連続した任意の組合せを指定可能にできる。

【０１６４】

同様の指定を可能にした、汎用レジスタの退避命令（ＳＴＭ）を持ち、これをサブルーチン乃至例外処理ルーチンの先頭で実行し、最後に、本発明の復帰命令を実行することによって、より一層の使い勝手を向上できる。汎用レジスタの退避命令実行時に、退避後に汎用レジスタをクリアすることにより、更に、命令コード長を短縮し、実行ステート数を短縮できる。

【０１６５】

また、データのリード／ライトを連続して行うことによって、メモリに対するバースト動作などやバス幅を拡張する場合にも、これらを有効に利用しやすい。

【０１６６】

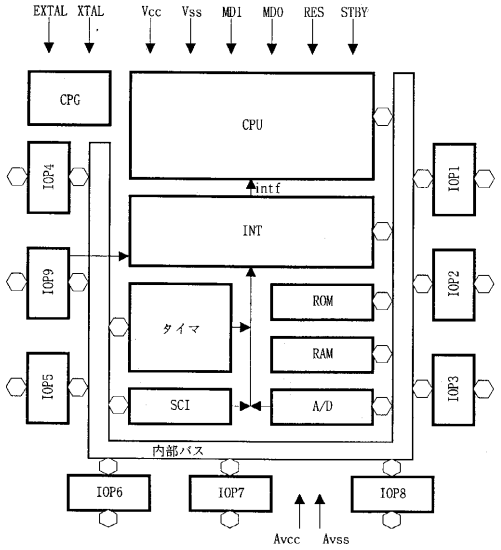
ソースプログラムレベルまたはオブジェクトプログラムレベルで、既存のＣＰＵの命令セットを包含した上で、上記命令を追加することによって、ソフトウェア資産を有効に利用することができ、使用者のソフトウェア開発効率を向上することができる。ソースプログラムレベルまたはオブジェクトプログラムレベルで互換性を保つことによる利点と前記転送命令を追加することの利点の双方を享受することができる。

【０１６７】
以上本発明者等によってなされた発明を実施例に限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能である。
【０１６８】
命令コードは一例であり、種々変更が可能である。既存の命令セットに追加するほか、新規の命令セットを開発する場合に適用することが可能であることは言うまでもない。
【０１６９】
例外処理におけるスタックの構造は、ＣＣＲとＰＣを組合せて３２ビットとしたが、これらは、１６ビットと３２ビットに分割したり、そのほかのコントロールレジスタを追加したりすることもできる。ＰＣも３２ビットとしてもよい。
【０１７０】
汎用レジスタの構成や本数など、或いは復帰命令で復帰する汎用レジスタの本数などは任意に変更できる。復帰する汎用レジスタの本数は１本でもよい。スタックに格納される順序なども任意に変更できる。汎用レジスタを復帰する場合、ＰＣ（及びＣＣＲ）と連続してリードする方が都合がよい。
【０１７１】
インクリメンタ（ＲＩＮＣ）も、インクリメント／デクリメントを行うほか、命令デコーダの出力との加算／減算を行うようにしてもよい。本発明の復帰命令の場合、命令コードのレジスタ指定フィールドに対して、１／２／３を順次減算するようにしてもよい。命令コード中に含まれるレジスタ指定フィールドも、最初に指定する汎用レジスタに対応している必要は必ずしもない。前記加算／減算と組合せて、所望の汎用レジスタを指定できればよい。
【０１７２】
復帰する汎用レジスタとして、スタックポインタであるＥＲ７が指定された場合は、ＥＲ７への書込みを禁止するとよい。スタックポインタの退避／復帰は意味がなく、所定のポイント動作を行えばよいからである。
【０１７３】
汎用レジスタの退避命令実行時に、退避後に汎用レジスタをクリアする命令と、保持する命令の両方を持ってもよい。当該命令コードの第１ワードの所定のビットでこれを指定して、制御信号（ＭＯＤ）を発生して、第２ワードの動作を変更するようにすればよい。
【０１７４】
汎用レジスタのクリアの内容は０に限定されず、他の値にしてもよい。クリアする内容を指定する手段を別に設けてもよい。
【０１７５】
命令の基本単位も１６ビットに限定されない。８ビット単位、３２ビット単位でも、本発明の復帰命令は適用可能である。
【０１７６】
簡単のために、バス幅を３２ビットとし、命令リードを１６ビット単位としたが、命令リードを３２ビット単位にして高速化することができる。このような例は、特願平１１－１６７８１２に記載されている。また、バス幅を１６ビットとして、ロングワードのリード／ライトを、ワード単位の２回のリード／ライトで実行するようにしてもよい。このような例は、特願平１１－３２０５１８に記載されている。
【０１７７】
マイクロコンピュータのその他の機能ブロックについても何等制約されない。
【０１７８】
以上の説明では主として本発明者によってなされた発明をその背景となった利用分野であるマイクロコンピュータに適用した場合について説明したが、それに限定されるものではなく、その他のデータ処理装置にも適用可能であり、本発明は少なくとも、複数のレジスタ手段を有するデータ処理装置に適用することができる。
【０１７９】

【発明の効果】
本題において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記の通りである。
【０１８０】
サブルーチンから、または例外処理からの復帰命令において、当該命令コードに複数の汎用レジスタを指定する情報を含み、当該命令コードの実行時に、プログラムカウンタの復帰に加えて、前記指定した汎用レジスタの復帰を行うようにすることにより、使い勝手を向上し、プログラム容量と実行ステート数を短縮できる。
【０１８１】
更に、指定する汎用レジスタの組合せを連続したものにすることにより、命令コード長を短縮でき、更に、各命令の実行ステート数を固定にすることにより、内部の条件分岐を行うことを少なくし、内部論理を簡潔にし、論理規模を縮小できる。１６ビット単位の可変長の命令コード体系を採る場合にも、命令プリフェッチや命令デコードを簡易にしつつ、命令コードを有効に利用して、全体的なプログラム効率を向上できる。
【０１８２】
同様の指定を可能にした、汎用レジスタの退避命令を持ち、これをサブルーチン乃至例外処理ルーチンの先頭で実行し、最後に、本発明の復帰命令を実行することによって、より一層の使い勝手を向上できる。
【０１８３】
汎用レジスタの復帰と、プログラムカウンタなどの復帰を連続的に行うことによって、内部の動作を容易にし、高速化できる。
【０１８４】
レジスタの本数の異なる復帰命令を複数サポートすることによって、また、連続した汎用レジスタの指定を可能にして、更に、プログラムの作成を容易にし、使い勝手を向上することができる。
【図面の簡単な説明】
【図１】本発明の適用されたマイクロコンピュータのブロック図。
【図２】ＣＰＵに内蔵されている汎用レジスタ及び制御レジスタの構成例の図。
【図３】ＣＰＵの構成例の図。
【図４】レジスタセレクタＲＥＳＬの一部の詳細ブロック図。
【図５】複数汎用レジスタの復帰を含む第１の復帰命令（ＲＴＳ／ｎ）、第２の復帰命令（ＲＴＥ／ｎ）の命令フォーマット図。
【図６】複数汎用レジスタの復帰を含む第１の復帰命令（ＲＴＳ／ｎ）、第２の復帰命令（ＲＴＥ／ｎ）のスタックの構成例の図。
【図７】第１の復帰命令（ＲＴＳ／ｎ）の動作フローの図。
【図８】第１の復帰命令（ＲＴＳ／ｎ）の実行タイミングの図。
【図９】第２の復帰命令（ＲＴＥ／ｎ）の動作フローの図。
【図１０】第２の復帰命令（ＲＴＥ／ｎ）の実行タイミングの図。
【図１１】複数汎用レジスタの退避命令（ＳＴＭ）の動作フローの図。
【図１２】複数汎用レジスタの退避命令（ＳＴＭ）の実行タイミングの図。
【図１３】本発明のＣＰＵの開発環境の概略図。

【図１】

図1



【図２】

図2

汎用レジスタ

	15	07	07	0
ER0	E0	R0H	R0L	
ER1	E1	R1H	R1L	
ER2	E2	R2H	R2L	
ER3	E3	R3H	R3L	
ER4	E4	R4H	R4L	
ER5	E5	R5H	R5L	
ER6	E6	R6H	R6L	
ER7(SP)	E7	R7H	R7L	

コントロールレジスタ

	23	0
PC		

	7	0
CCR	I	UHUNZVC

【図３】

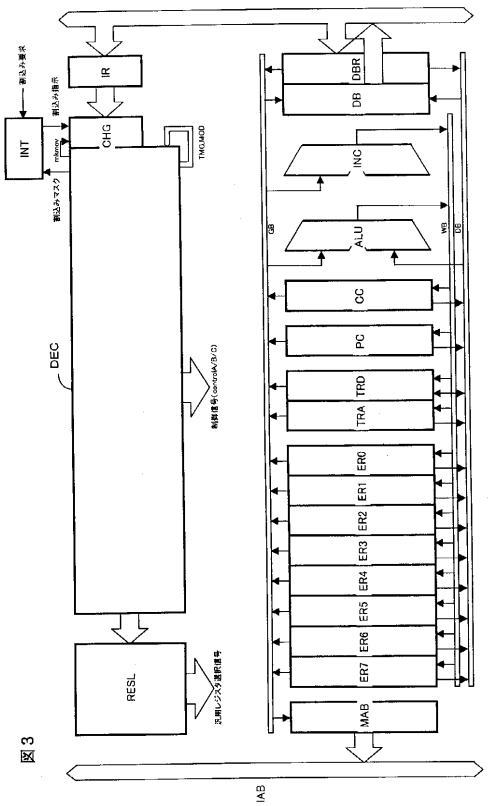


図３

【図４】

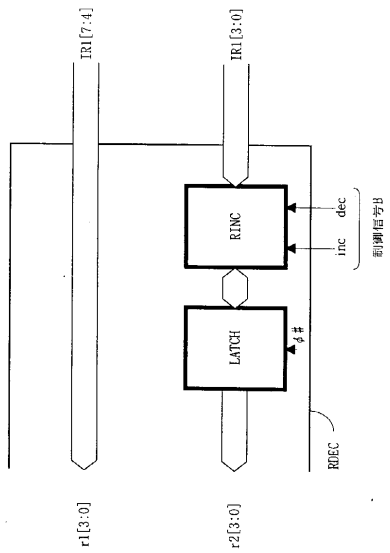


図４

【図５】

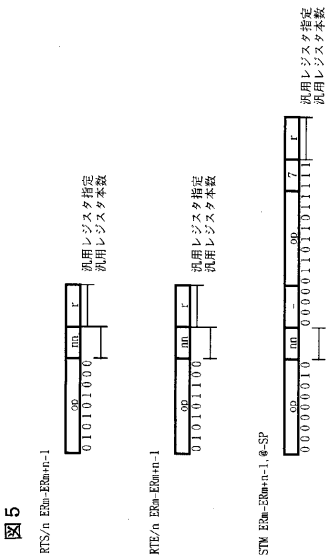


図５

【図６】

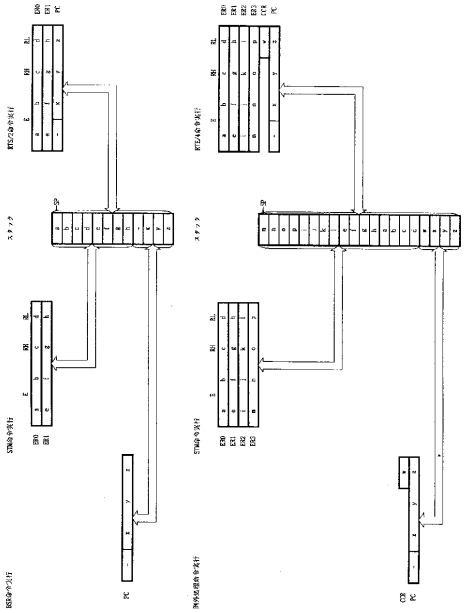
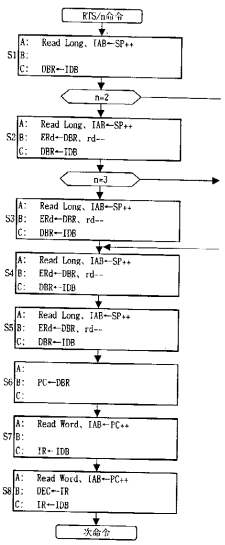


図６

【図７】

図７



【図８】

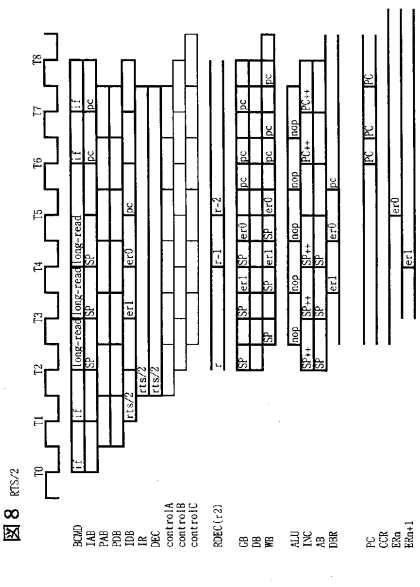
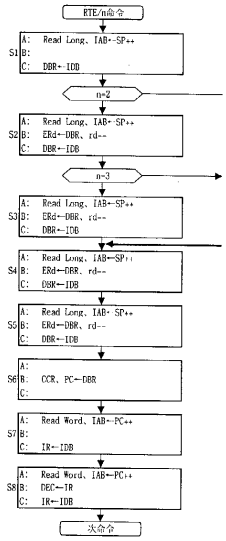


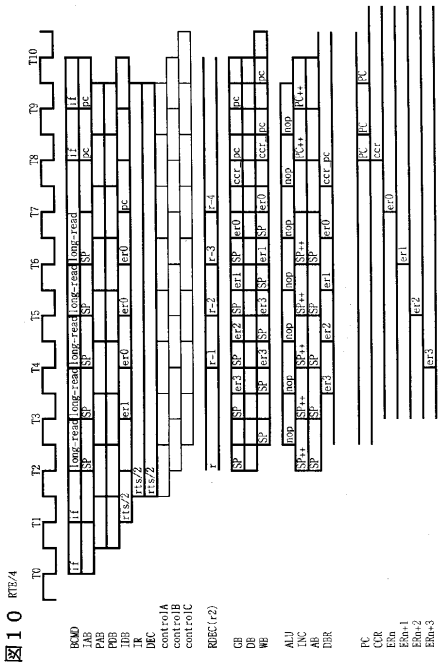
図８

【図９】

図9

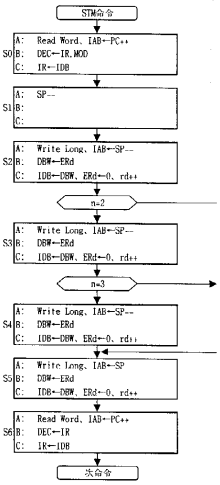


【図１０】

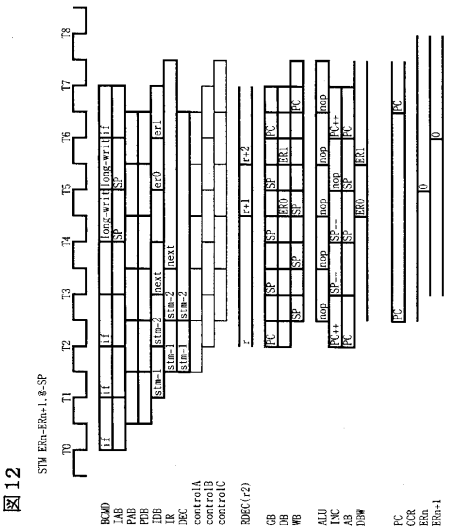


【図１１】

図11



【図１２】



【図１３】

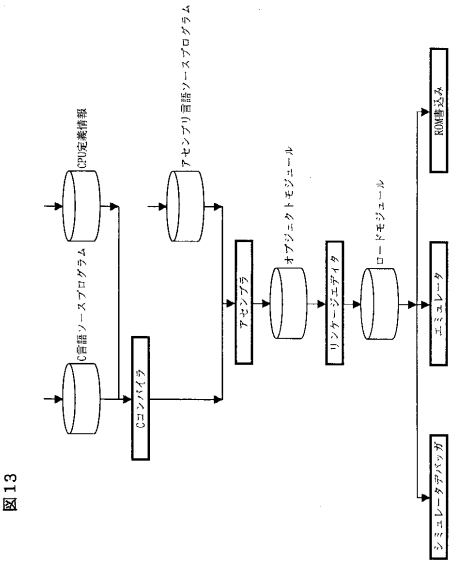

CPU定義情報
C言語ソースプログラム
アセンブリ言語ソースプログラム
Cコンパイラ
アセンブラ
オブジェクトモジュール
リンケージエディタ
ロードモジュール
結線情報
エミュレータ
シミュレータデバッガ

図13

フロントページの続き

(56)参考文献 特開平５－６２８１（ＪＰ，Ａ）
特開平８－２６３２９０（ＪＰ，Ａ）
特開平２－１０３６３２（ＪＰ，Ａ）
特開平５－１２７９０５（ＪＰ，Ａ）
特開昭５７－１０５０４５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/40 - 9/42
G06F 9/30 - 9/355