(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3670801号
(P3670801)
(45)発行日　平成17年7月13日(2005.7.13)　(24)登録日　平成17年4月22日(2005.4.22)

(51)Int.Cl.[7]　F I
G06F　9/38　G06F　9/38　310A

請求項の数 11　(全 28 頁)

(21)出願番号　特願平9-159049
(22)出願日　平成9年6月16日(1997.6.16)
(65)公開番号　特開平11-7388
(43)公開日　平成11年1月12日(1999.1.12)
審査請求日　平成14年9月12日(2002.9.12)

(73)特許権者　000005821
松下電器産業株式会社
大阪府門真市大字門真１００６番地
(74)代理人　100090446
弁理士　中島　司朗
(72)発明者　高山　秀一
大阪府門真市大字門真１００６番地　松下電器産業株式会社内
(72)発明者　桧垣　信生
大阪府門真市大字門真１００６番地　松下電器産業株式会社内

審査官　後藤　彰

最終頁に続く

(54)【発明の名称】プロセッサ

(57)【特許請求の範囲】
【請求項１】
並列実行させるオペレーションを指定するための複数のオペレーションフィールドを１命令中に含む命令を解読し実行するＶＬＩＷプロセッサであって、
定数を格納するための定数保持手段と、
前記複数のオペレーションフィールドを並列に解読し、オペレーションを指定する代わりに、元の定数が分割された分割定数が置かれていることを検出する解読手段と、
複数のオペレーションフィールドに跨って離れて配置されている複数の分割定数を、前記解読手段により分割定数が置かれていると検出される度に、前記定数保持手段に順次格納することにより、元の定数を復元する定数格納制御手段と、
前記定数格納制御手段により前記定数保持手段内に復元された元の定数を用いてオペレーションフィールドに指定されているオペレーションを実行する実行手段とを備え、
前記定数格納制御手段は、
前記定数保持手段に、第１の分割定数を格納した後に、さらに第２の分割定数を格納する際に、
（１）前記定数保持手段に格納されている第１の分割定数を、第２の分割定数の桁数分シフトするか、又は
（２）前記定数保持手段に格納されている定数の有効桁数を示す値を参照することで、適切な位置に第２の分割定数を格納し、前記値を第２の分割定数の桁数だけ更新し、
ここで、前記定数保持手段の桁数および前記元の定数の桁数は、前記第１の分割定数が

置かれているオペレーションフィールドの桁数および前記第２の分割定数が置かれているオペレーションフィールドの桁数よりも大きく、前記第１の分割定数が置かれているオペレーションフィールドの桁数および前記第２の分割定数が置かれているオペレーションフィールドの桁数はいずれも前記実行手段において並列実行できる命令の桁数よりも小さいこと

を特徴とするＶＬＩＷプロセッサ。

【請求項２】

前記第１の分割定数が置かれているオペレーションフィールドと、前記第２の分割定数が置かれているオペレーションフィールドとの間のオペレーションフィールドに、当該第１の分割定数および当該第２の分割定数を利用しないオペレーションが、少なくとも１つ指定されていること

を特徴とする請求項１に記載のＶＬＩＷプロセッサ。

【請求項３】

前記第１の分割定数と前記第２の分割定数は、１命令中に離れて配置されていること

を特徴とする請求項１に記載のＶＬＩＷプロセッサ。

【請求項４】

前記第１の分割定数と前記第２の分割定数は、複数の命令に跨って配置されていること

を特徴とする請求項１に記載のＶＬＩＷプロセッサ。

【請求項５】

前記定数格納制御手段は、前記定数保持手段の下位桁から上位桁に向けて順次に前記分割定数を蓄積して格納していくこと

を特徴とする請求項１～４のいずれか１項に記載のＶＬＩＷプロセッサ。

【請求項６】

前記定数格納制御手段は、前記定数保持手段の上位桁から下位桁に向けて順次に前記分割定数を蓄積して格納していくこと

を特徴とする請求項１～４のいずれか１項に記載のＶＬＩＷプロセッサ。

【請求項７】

前記ＶＬＩＷプロセッサはさらに、前記元の定数又は前記分割定数に対してゼロ拡張及び符号拡張の少なくとも一方の拡張処理を行なう拡張手段を備えること

を特徴とする請求項１～４のいずれか１項に記載のＶＬＩＷプロセッサ。

【請求項８】

前記拡張手段は、前記定数保持手段に格納される前の前記分割定数に対して前記拡張処理を行なうこと

を特徴とする請求項７に記載のＶＬＩＷプロセッサ。

【請求項９】

前記拡張手段は、前記定数保持手段から読み出された前記元の定数に対して前記拡張処理を行なうこと

を特徴とする請求項７に記載のＶＬＩＷプロセッサ。

【請求項１０】

前記定数格納制御手段は、前記実行手段によって前記定数保持手段に格納された前記元の定数が読み出された直後に前記定数レジスタにゼロを格納すること

を特徴とする請求項１～４のいずれか１項に記載のＶＬＩＷプロセッサ。

【請求項１１】

前記実行手段は、前記命令に従って分岐オペレーションを実行する分岐実行部を有し、

前記定数格納制御手段は、前記分岐実行部により分岐オペレーションが実行された場合に前記定数保持手段にゼロを格納すること

を特徴とする請求項１～４のいずれか１項に記載のＶＬＩＷプロセッサ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、マイクロプロセッサに関し、特に、命令中に生じる無駄領域や未使用領域を有効化させる技術に関する。

【０００２】

【従来の技術】

近年のマイクロプロセッサ応用製品の高機能化及び高速化に伴い、コード効率の高いプログラムを実行することができるマイクロプロセッサ（以下、単に「プロセッサ」という。）が望まれている。つまり、プログラムを構成する各命令中には無駄なコードや未使用の領域が含まれないことが望ましい。

【０００３】

ところが、特に、ＶＬＩＷ（Very Long Instruction Word）の如く固定長命令の場合には、命令中にノーオペレーションコード（ｎｏｐコード）等の無駄なコードを置く必要が生じる。ＶＬＩＷは複数のオペレーションフィールドからなり、各オペレーションフィールドにおいてプロセッサが備える複数の演算ユニットに対応したオペレーションが指定されるが、オペレーションの依存関係等により、常に並列実行可能な複数のオペレーションが存在するとは限らないからである。

【０００４】

このようなｎｏｐコードの挿入に伴うコード効率の低下を回避する従来の技術として、例えば、特開平８－１６１１６９に開示されたＶＬＩＷ方式の計算機システムがある。

図２４は、上記従来技術における命令フォーマットを示す図である。

この従来技術では、例えば、オペレーションフィールド＃２にｎｏｐコードを配置する必要がある場合に、そのオペレーションフィールド＃２にはｎｏｐコードに代えて他の命令で用いられる定数を配置しておき、オペレーションフィールド＃１の一部の箇所にその旨を示す命令有効化情報を配置する。そして、プロセッサは、その命令の実行時に、命令有効化情報を参照することによってオペレーションフィールド＃２には定数だけが置かれていると解読し、それをオペランドとして演算に用いる。このようにして、命令中にｎｏｐコードを挿入することによる無駄領域の発生を回避している。

【０００５】

【発明が解決しようとする課題】

しかしながら、上記従来技術では、無駄領域を埋めることができる定数の大きさが制限されるという問題がある。

例えば、３２ビット長の１個のオペレーションフィールドにのみｎｏｐコードを置く必要がある場合に６４ビットからなる数値定数を置くことはできない。また、例えば、３２ビットの固定長命令中の８ビットのみが未使用領域である場合には、その８ビットを埋めることができる定数は８ビット以下で表現される定数に限定され、例えば３２ビットで表現される絶対番地で埋めることはできない。

【０００６】

つまり、上記従来技術は、比較的大きい桁数の無駄領域が生じる場合には効果を奏するものの、命令の桁数が例えば３２ビットの如く小さい場合には、そこに生じる無駄領域も小さいために、多くの無駄領域が定数で埋められずにそのまま残ってしまうという問題がある。

そこで、本発明はかかる問題点に鑑みてなされたものであり、命令中の無駄領域を埋めるための定数の大きさがその命令の桁数によって制限されることがないプロセッサを提供することを目的とする。これによって、たとえ命令中の小さな桁数の領域であっても無駄領域として残ってしまうことが回避され、コード効率の高いプログラムの作成が支援されることをめざす。

【０００７】

【課題を解決するための手段】

上記目的を達成するために本発明は、命令を解読し実行するプロセッサであって、定数を格納するための定数レジスタと、前記命令中に前記定数レジスタへ格納すべき定数が置かれていることを解読する解読手段と、前記解読手段により前記定数が置かれていると解読

された場合には、前記定数レジスタに定数が格納されていないと判断すると前記定数を所定位置に格納し、前記定数レジスタに定数が既に格納されていると判断すると前記定数レジスタに既に格納されている定数を残したまま新たな定数を前記定数レジスタに格納する格納手段と、前記定数レジスタに格納されている全ての定数を読み出し、その定数をオペランドとするオペレーションを実行する実行手段とを備えることを特徴とする。

【０００８】

これによって、複数の命令に跨って分割配置されていた定数の断片が定数レジスタに格納されて蓄積され、元の１個の定数に復元される。従って、命令中に小さな桁数の無駄領域が生じる場合であっても、その無駄領域を桁数の大きい定数の一部で埋めておくことが可能となり、コード効率の高いプログラムの作成が支援される。

【０００９】

【発明の実施の形態】

以下、本発明に係るプロセッサの実施の形態について、図面を用いて詳細に説明する。なお、本明細書では、「命令」とは本プロセッサが同時並列に解読し実行するコード全体を意味し、「オペレーション」とは本プロセッサが並列に実行できる数値演算、論理演算、転送、分岐等の処理単位又はその処理単位を指定するためのコードを意味する。

（命令フォーマット）

まず、本プロセッサが解読実行する命令の構造について説明する。

【００１０】

本プロセッサは、ＶＬＩＷアーキテクチャを採るプロセッサ（以下、「ＶＬＩＷプロセッサ」という。）であり、３２ビット固定長の命令を解読実行する。

図１（ａ）は、本プロセッサが実行する命令５０のフィールド構成を示す図である。

図１（ｂ）～図１（ｄ）は１６種類の命令フォーマットを示す図であり、そのうち、図１（ｂ）は３オペレーション、図１（ｃ）は２オペレーション、図１（ｄ）は１オペレーションを同時に指定できる命令フォーマットである。

【００１１】

この命令５０は、３２ビット固定長であり、４ビットずつに区切られた８個のフィールド（上位よりＰ０．０フィールド５１、Ｐ１．０フィールド５２、…、Ｐ３．２フィールド５８）からなる。なお、Ｐ２．０フィールド５３～Ｐ２．２フィールド５５のグループをまとめて第１演算フィールド５９と呼び、Ｐ３．０フィールド５６～Ｐ３．２フィールド５８のグループをまとめて第２演算フィールド６０と呼ぶ。

【００１２】

図１（ｂ）～図１（ｄ）において、“ｃｏｎｓｔ”は定数であり、これが用いられるオペレーションの種類によっては即値、絶対番地、ディスプレースメント等の数値定数や文字定数を意味する。“ｏｐ”はオペレーションの種類を指定するオペコードを、“Ｒｓ”はソースオペランドとなるレジスタを、“Ｒｄ”はデスティネーションオペランドとなるレジスタを、“ｃｃ”は本プロセッサが備える専用の３２ビットレジスタ（図３に示される定数レジスタ３６）の格納値を分岐先の絶対番地又は相対番地（ディスプレースメント）とする分岐オペレーションを指定するオペコードを意味する。

【００１３】

また、これらコードの直後に添付された数値は、第１演算フィールド５９及び第２演算フィールド６０のいずれのオペレーションのために用いられるものであるかを示す。例えば、フォーマットコードが“６”である命令フォーマットの場合であれば、Ｐ１．０フィールド５２に置かれた４ビットの定数“ｃｏｎｓｔ１”とＰ２．１フィールド５４に置かれた４ビットの定数“ｃｏｎｓｔ１”とは結合され、８ビットの定数として第１演算フィールド５９のオペコード“ｏｐ１”に対応するソースオペランドになることを意味する。

【００１４】

また、数値を伴わない定数“ｃｏｎｓｔ”は、本プロセッサが備える専用の３２ビットレジスタ（図３に示される定数レジスタ３６）に格納される定数を示す。例えば、フォーマットコードが“０”である命令フォーマットにおけるＰ１．０フィールド５２に置かれた

４ビットの定数“ｃｏｎｓｔ”は、暗黙的に指定された定数レジスタ３６に格納される定数である。

【００１５】

図２は、図１で用いられている３種類のオペコード“ｃｃ”、“ｏｐ１”及び“ｏｐ２”それぞれによって指定される具体的なオペレーションを説明する図である。

４ビットのオペコード“ｃｃ”は、１６種類の分岐オペレーションの中の一つを指定する。１つの分岐オペレーションは、分岐条件と分岐形式によって特定される。分岐条件には、等しい（“ｅｑ”）、等しくない（“ｎｅｑ”）、より大きい（“ｇｔ”）等がある。分岐形式には、上記定数レジスタ３６の格納値を分岐先の絶対番地として分岐する形式（ニーモニック表示において“ｉ”が添付されていないもの）と相対番地として分岐する形式（ニーモニック表示において“ｉ”が添付されているもの）とがある。例えば、オペコード“ｅｑ”は、直前の比較結果が等しい場合に絶対番地指定による分岐を行なうオペレーションを意味し、オペコード“ｅｑｉ”は、直前の比較結果が等しい場合に相対番地指定による分岐を行なうオペレーションを意味する。

【００１６】

４ビットのオペコード“ｏｐ１”は、“ａｄｄ”（加算）、“ｓｕｂ”（減算）、“ｍｕｌ”（乗算）、“ａｎｄ”（論理積）、“ｏｒ”（論理和）等の算術論理演算に属するオペレーションの一つを指定する場合と、“ｍｏｖ”（ワード（３２ビット）データの転送）、“ｍｏｖｈ”（ハーフワードデータの転送）、“ｍｏｖｂ”（バイトデータの転送）等のレジスタ・レジスタ間転送に属するオペレーションの一つを指定する場合とがある。

【００１７】

４ビットのオペコード“ｏｐ２”は、上記オペコード“ｏｐ１”と同様の算術論理演算及びレジスタ・レジスタ間転送に加えて、“ｌｄ”（メモリからレジスタへの１ワードデータのロード）、“ｓｔ”（レジスタからメモリへのワードデータのストア）等のレジスタ・メモリ間転送に属するオペレーションの一つを指定する場合がある。

【００１８】

次に、図１（ａ）に示された各フィールド５１、５２、５９、６０の特徴を説明する。

Ｐ０．０フィールド５１は、この命令５０のフォーマットを特定する４ビットのフォーマットコードを置くためのフィールドであり、具体的には、図１（ｂ）～図１（ｄ）に示された１６種類の命令フォーマットの一つを特定する。

【００１９】

Ｐ１．０フィールド５２は、定数又は分岐用のオペコードを置くためのフィールドである。

このＰ１．０フィールド５２に定数が置かれた場合（フォーマットコード＝０、１、４～９の場合）には、その定数は、定数レジスタ３６に格納する対象となる場合（フォーマットコード＝０、１、４、５の場合）と、第１演算フィールド５９又は第２演算フィールド６０のオペランドの一部を構成する場合（フォーマットコード＝５、７、８、９、Ｂの場合）とがある。さらに、定数レジスタ３６に格納する対象となる場合には、その４ビットの定数のみが格納される場合（フォーマットコード＝０、１の場合）と、第１演算フィールド５９又は第２演算フィールド６０に置かれた１２ビットの定数と共に格納される場合（フォーマットコード＝４、５の場合）とがある。

【００２０】

一方、このＰ１．０フィールド５２に分岐用のオペコード“ｃｃ”が置かれた場合（フォーマットコード＝２、３、Ａの場合）には、定数レジスタ３６に格納された定数を分岐先の絶対番地として、又は、相対番地（ディスプレースメント）として分岐することを意味する。

第１演算フィールド５９は、本プロセッサと外部（メモリ）とのデータの転送を伴わないオペレーション（算術論理演算、レジスタ間転送）を指定するためのオペコードとオペランド（ソース及びデスティネーション）との組又は定数が置かれる。

【００２１】

第２演算フィールド６０は、上記第１演算フィールド５９の場合に加えて、本プロセッサと外部（メモリ）とのデータの転送を伴うペレーション（レジスタ・メモリ間転送）を指定するためのオペコードとオペランドとの組が置かれることもある。
なお、以上のようなオペレーションの種類の各フィールドへの割当ては、ノイマン型の本プロセッサにおいては２つ以上の分岐オペレーションを同時に実行する必要がないこと、本プロセッサと外部（メモリ）とのオペランドの入出力ポート（図３におけるオペランドアクセス部４０）を１つに限定していること等に基づく。
【００２２】
ここで、図１（ｂ）～図１（ｄ）に示された命令フォーマットには以下の特徴がある。
第１に、定数“ｃｏｎｓｔ”に着目して判るように、定数レジスタ３６に定数を格納させる命令フォーマットは次の３通りである。
（１）フォーマットコードが“０”又は“１”の場合：
この命令では、Ｐ１．０フィールド５２に置かれた４ビットの定数が定数レジスタ３６に格納される。
（２）フォーマットコードが“４”の場合：
この命令では、Ｐ１．０フィールド５２～Ｐ２．２フィールド５５に置かれた１６ビットの定数が定数レジスタ３６に格納される。
（３）フォーマットコードが“５”の場合：
この命令では、Ｐ１．０フィールド５２とＰ３．０フィールド５６～Ｐ３．２フィールド５８に置かれた１６ビットの定数が定数レジスタ３６に格納される。
【００２３】
第２に、本プロセッサでは、１個の命令に最大３つのオペレーションを指定することができるが、その場合には、図１（ｂ）に示された３オペレーション用の命令フォーマットから判るように、それら３つのオペレーションの種類は次のいずれかの組み合わせになる。
（１）４ビットの定数を定数レジスタ３６にセットするオペレーションと２個の汎用オペレーション（フォーマットコードが“０”、“１”の場合）
（２）定数レジスタ３６にセットされた値を絶対番地又は相対番地として分岐するオペレーションと２個の汎用オペレーション（フォーマットコードが“２”、“３”の場合）
このように、本プロセッサの命令は、わずか３２ビット長でありながら最大３つのオペレーションを同時に指定することができるコード効率の高いフィールド構成を有している。
（プロセッサのハードウェア構成）
次に、本プロセッサのハードウェア構成を説明する。
【００２４】
図３は、本発明に係るプロセッサのハードウェア構成を示すブロック図である。
本プロセッサは、上述したように、最大３つのオペレーションを並列実行するＶＬＩＷプロセッサであり、大きく分けて、命令レジスタ１０、解読部２０及び実行部３０から構成される。
【００２５】
命令レジスタ１０は、命令フェッチ部３９から送られてきた１個の命令を保持する３２ビットのレジスタである。
解読部２０は、命令レジスタ１０に保持された命令を解読し、その解読結果に応じた制御線を実行部３０に出力するものであり、大きく分けて、フォーマットデコーダ２１と命令デコーダ２２とからなる。
【００２６】
命令デコーダ２２はさらに、Ｐ１．０フィールド１２に保持されたオペコード“ｃｃ”を解読しその結果に基づいてＰＣ部３３を制御する分岐デコーダ２３と、Ｐ２．０フィールド１３に保持されたオペコードを解読しその結果に基づいて第１演算部３７を制御する第１演算デコーダ２４と、Ｐ３．０フィールド１６に保持されたオペコードを解読しその結果に基づいて第２演算部３８及びオペランドアクセス部４０を制御する第２演算デコーダ２５とからなる。

【００２７】
フォーマットデコーダ２１は、Ｐ０．０フィールド１１に保持された４ビットのフォーマットコードをデコードすることによって命令レジスタ１０に保持された命令のフォーマットが図１（ｂ）～図１（ｄ）に示された１６種類のうちのいずれであるかを特定し、その結果に応じて分岐デコーダ２３、第１演算デコーダ２４及び第２演算デコーダ２５による解読動作を許可又は禁止したり、実行部３０の定数レジスタ制御部３２を動作させたりする。
【００２８】
なお、上記デコーダ２１、２３～２５は、基本的には１サイクルに１つのオペレーションを解読し、実行部３０に制御信号を与える。また、命令レジスタ１０と実行部３０を接続する２６ビットの定数信号線２６は、命令レジスタ１０に置かれた定数やオペランドを実行部３０に転送するためのバスである。
実行部３０は、解読部２０での解読結果に基づいて、最大３つのオペレーションを並列実行する回路ユニットであり、実行制御部３１、ＰＣ部３３、レジスタ群３４、第１演算部３７、第２演算部３８、命令フェッチ部３９及びオペランドアクセス部４０からなる。なお、この実行部３０のうち定数レジスタ制御部３２、ＰＣ部３３及び定数レジスタ３６については、別の図面においてさらに詳細な構成を示している。
【００２９】
実行制御部３１は、解読部２０での実行結果に基づいて実行部３０の各構成要素３３～４０を制御する制御回路や配線の総称であり、通常のプロセッサが備える構成要素（タイミング制御、動作許可禁止制御、ステータス管理、割り込み制御等の回路）の他に本プロセッサに特有の定数レジスタ制御部３２を有する。定数レジスタ制御部３２は、フォーマットデコーダ２１からの指示に基づいて命令レジスタ１０に保持された４ビット又は１６ビットの定数（ｃｏｎｓｔ）を定数レジスタ３６に格納する制御を行なう。
【００３０】
ＰＣ（プログラムカウンタ）部３３は、分岐デコーダ２３による制御の下で、次に解読実行すべき命令が置かれている図示されていない外部メモリ上のアドレスを命令フェッチ部３９に出力する。
命令フェッチ部３９は、３２ビットのＩＡ（インストラクションアドレス）バス及び３２ビットのＩＤ（インストラクションデータ）バスを通じて図示されていない外部メモリから命令ブロックをフェッチし、内部の命令キャッシュに保持すると共に、ＰＣ部３３から出力されたアドレスに相当する命令を命令レジスタ１０に供給する。
【００３１】
レジスタ群３４は、１５個の３２ビット汎用レジスタ３５と１個の３２ビット定数レジスタ３６から構成される。これら１６個のレジスタ３５、３６に格納された値は、第１演算デコーダ２４及び第２演算デコーダ２５での解読結果に基づいて、第１演算部３７及び第２演算部３８に転送され、ここで演算が施され、又は、ここを単に通過した後に、レジスタ群３４又はオペランドアクセス部４０に送られる。なお、定数レジスタ３６に格納された値は、第１演算部３７及び第２演算部３８での演算に用いられる他に、ＰＣ部３３にも転送され、ここで分岐先となる有効アドレスを生成するために用いられる。
【００３２】
第１演算部３７は、２個の３２ビットデータに対して算術論理演算を行なうＡＬＵと乗算を行う乗算器とを内部に有し、第１演算デコーダ２４による制御の下で２種類のオペレーション（算術論理演算とレジスタ間転送）を実行する。
第２演算部３８も、第１演算部３７と同様に、２個の３２ビットデータに対して算術論理演算を行なうＡＬＵと乗算を行う乗算器とを内部に有し、第２演算デコーダ２５による制御の下で２種類のオペレーション（算術論理演算とレジスタ間転送）を実行する。
【００３３】
オペランドアクセス部４０は、第２演算デコーダによる制御の下でレジスタ群３４と図示されていない外部メモリとの間でオペランドの転送を行なう回路であり、そのオペランド

やオペランドアドレスを保持するバッファを内部に有する。具体的には、例えば、命令レジスタ１０のＰ３．１フィールド１６にオペコード“ｌｄ”が置かれていた場合には、外部メモリに置かれていた１ワードのデータがオペランドアクセス部４０を経てレジスタ群３４のいずれかのレジスタにロードされ、また、オペコード“ｓｔ”が置かれていた場合には、レジスタ群３４のいずれかのレジスタの格納値が外部メモリにストアされる。

【００３４】

上記ＰＣ部３３、レジスタ群３４、第１演算部３７、第２演算部３８及びオペランドアクセス部４０は、図示されるように、内部バス（Ｌ１バス、Ｒ１バス、Ｌ２バス、Ｒ２バス、Ｄ１バス、Ｄ２バス）で接続されている。なお、Ｌ１バス及びＲ１バスはそれぞれ第１演算部３７の２つの入力ポートに、Ｌ２バス及びＲ２バスはそれぞれ第２演算部３８の２つの入力ポートに、Ｄ１バス及びＤ２バスはそれぞれ第１演算部３７及び第２演算部３８の出力ポートに接続されている。（定数レジスタ３６及びその周辺回路の詳細な構成）

次に、定数レジスタ３６及びその周辺回路について詳細に説明する。

【００３５】

図４は、定数レジスタ３６及びその周辺回路の詳細な構成と接続関係を示すブロック図である。なお、図中の固定値（“０”）２７は、定数“０”を示す４本の信号線の固定的な配線を意味する。

定数レジスタ制御部３２は、５個の３入力セレクタ３２ａ～３２ｅと３個の４入力セレクタ３２ｆ～３２ｈとからなり、定数レジスタ３６は、８個の４ビット幅レジスタ３６ａ～３６ｈからなる。なお、各入出力データは並列４ビットである。

【００３６】

定数レジスタ制御部３２は、フォーマットデコーダ２１及び命令デコーダ２２からの制御信号に従って上記８個の入力セレクタ３２ａ～３２ｈを制御することで、以下に示す４通りの格納方法のいずれかの方法により、命令レジスタ１０に保持された定数又はゼロを定数レジスタ３６に格納させる。

図５（ａ）～図５（ｄ）は、その４通りの格納方法を説明する図である。

【００３７】

図５（ａ）は、フォーマットデコーダ２１によってＰ０．０フィールド１１に保持された値が“０”又は“１”であると解読された場合の格納方法を示す。これは、Ｐ１．０フィールド１２に置かれた４ビットの定数のみを定数レジスタ３６に格納する場合に相当する。

具体的には、定数レジスタ３６に保持されたデータを４ビット単位で上位にシフトさせると同時に、命令レジスタ１０のＰ１．０フィールド１２に保持された４ビットの定数を定数レジスタ３６の最下位の４ビットレジスタ３６ｈに格納する。

【００３８】

図５（ｂ）は、フォーマットデコーダ２１によってＰ０．０フィールド１１に保持された値が“４”であると解読された場合の格納方法を示す。これは、Ｐ１．０フィールド１２～Ｐ２．２フィールド１５に置かれた１６ビットの定数を定数レジスタ３６に格納する場合に相当する。

具体的には、定数レジスタ３６の下位１６ビット３６ｅ～３６ｈに保持されたデータを上位１６ビット３６ａ～３６ｄにシフトさせると同時に、命令レジスタ１０のＰ１．０フィールド１２～Ｐ２．２フィールド１５に保持された１６ビットの定数を定数レジスタ３６の下位１６ビット３６ｅ～３６ｈに格納する。

【００３９】

図５（ｃ）は、フォーマットデコーダ２１によってＰ０．０フィールド１１に保持された値が“５”であると解読された場合の格納方法を示す。これは、Ｐ１．０フィールド１２とＰ３．０フィールド１６～Ｐ３．２フィールド１８に置かれた１６ビットの定数を定数レジスタ３６に格納する場合に相当する。

具体的には、定数レジスタ３６の下位１６ビット３６ｅ～３６ｈに保持されたデータを上位１６ビット３６ａ～３６ｄにシフトさせると同時に、命令レジスタ１０のＰ１．０フィ

ールド１２とＰ３．０フィールド１６～Ｐ３．２フィールド１８に保持された１６ビットの定数を定数レジスタ３６の下位１６ビット３６ｅ～３６ｈに格納する。

【００４０】

図５（ｄ）は、フォーマットデコーダ２１によってＰ０．０フィールド１１に保持された値が“２”、“３”及び“Ａ”のいずれかであると解読された場合又は命令デコーダ２２によってＰ２．１フィールド１４、Ｐ２．２フィールド１５、Ｐ３．２フィールド１７及びＰ３．３フィールド１８の少なくとも一つに定数レジスタ（Ｒ１５）が指定されている場合の格納方法を示す。これは、Ｐ１．０フィールド１２に置かれた分岐オペレーション、第１演算フィールド５９及び第２演算フィールド６０の少なくとも一つのオペレーションによって定数レジスタ３６の格納値が使用された（読み出された）後に、定数レジスタ３６にオールゼロを格納する（定数レジスタ３６をクリアする）場合に相当する。

【００４１】

具体的には、定数レジスタ３６の格納値がＰＣ部３３、第１演算部３７及び第２演算部３８のいずれかに読み出された直後に、３２ビットの定数“０”を定数レジスタ３６に格納する。

なお、定数レジスタ３６の使用後にクリアしておくのは、定数レジスタ３６には常にゼロ拡張された値が格納されていることを保証するためである。ここで、ゼロ拡張とは、ある数値の有効桁数が一定の桁数に満たない場合に、その有効桁より上位の桁全てをゼロで埋める処理をいう。

【００４２】

以上のように、命令レジスタ１０のＰ０．０フィールド１１の値が“０”、“１”、“４”、“５”の場合には、定数レジスタ３６に既に格納された定数をシフトさせながら新たな定数が定数レジスタ３６に格納される。また、定数レジスタ３６は、その格納値が一旦読み出されて使用されると、その内容は消去される。このようにして、定数レジスタ３６は、その内容が読み出されるまで、次々に格納される定数を蓄積していくことができる。

（ＰＣ部３３の詳細な構成）

次に、ＰＣ部３３の詳細な構成を説明する。

【００４３】

図６は、ＰＣ部３３の詳細な構成を示すブロック図である。

ＰＣ部３３は、定数“４”を示す固定的な配線である固定値（“４”）３３ａ、２入力セレクタ３３ｂ、加算器３３ｃ、次に解読実行すべき命令のアドレスを保持するＰＣ３３ｄ及び４入力セレクタ３３ｅから構成される。

このＰＣ部３３では、解読部２０からの制御信号に従ってセレクタ３３ｂ、３３ｅが動作することにより、以下の３通りの値のいずれかが有効アドレスとしてセレクタ３３ｅから命令フェッチ部３９に出力される。

（１）ＰＣ３３ｄの内容に“４”を加算した値

これは、分岐しないで順次に実行する場合、即ち、解読実行された命令に分岐オペレーションが指定されていない場合に相当する。なお、“４”を加算するのは、１つの命令の長さが４バイト（３２ビット）であることによる。

（２）ＰＣ３３ｄの内容に定数レジスタ３６の内容を加算した値

これは、定数レジスタ３６の内容を相対番地として分岐する場合、例えば、Ｐ１．０フィールド１２によって相対番地による分岐が指定されていると分岐デコーダ２３が解読した場合が該当する。

（３）定数レジスタ３６の内容

これは、定数レジスタ３６の内容を絶対番地として分岐する場合、例えば、Ｐ１．０フィールド１２によって絶対番地による分岐が指定されていると分岐デコーダ２３が解読した場合が該当する。

【００４４】

以上のように、このＰＣ部３３は、専用の加算器３３ｃを備え、定数レジスタ３６に保持された値を直接用いる構成となっているので、第１演算部３７や第２演算部３８での演算

とは独立並行して、定数レジスタ３６の格納値を絶対番地又は相対番地として分岐する実行制御を行なうことができる。
（プロセッサの動作）
次に、具体的な命令を解読実行した場合の本プロセッサの動作について説明する。
【００４５】
図７は、３２ビットの定数を扱う処理の一例を示すフローチャートである。
本図には、レジスタＲ０とＲ１との格納値の差を求め（ステップＳ８０）、その結果にレジスタＲ２の格納値を掛け（ステップＳ８１）、さらにその結果に３２ビットの定数“０ｘ８７６５４３２１”（１６進数の“８７６５４３２１”）を加え（ステップＳ８２、Ｓ８３）、最後にレジスタＲ３をクリアしておく（ステップＳ８５）という処理が示されている。
【００４６】
図８は、図７に示された処理内容を本プロセッサに行なわせるプログラムの例を示す図である。
このプログラムは、３個の命令７１～７３から構成されている。１行が１個の命令に相当し、各命令の内容は各フィールドに置かれたニーモニックで表現されている。なお、定数は全て１６進数で表現されている。また、“ｆｍｔｎ（ｎ＝０～Ｆ）”はフォーマットコード“ｎ”を示し、“Ｒｎ（ｎ＝０～１５）”はレジスタ群３４の中の１つのレジスタを示す。なお、“Ｒ１５”は定数レジスタ３６を意味する。
【００４７】
図９は、図８に示されたプログラムを実行した場合の本プロセッサの動作を示すタイミングチャートである。
本図には、クロックサイクル、汎用レジスタＲ０～Ｒ３及び定数レジスタＲ１５の内容、４つのバスＬ１、Ｒ１，Ｌ２，Ｒ２を流れるデータが示されている。
上記図８及び図９を用いて、各命令７１～７３ごとの本プロセッサの動作を説明する。
（命令７１）
命令７１が命令レジスタ１０にロードされると、本プロセッサは図９のクロックサイクルｔ０～ｔ１に示された動作をする。
【００４８】
フォーマットデコーダ２１は、命令レジスタ１０のＰ０．０フィールド１１の値（“ｆｍｔ４”）から、この命令はフォーマットコードが“４”の２オペレーション命令であると判断し、以下の２つのオペレーションが並列実行されるように実行部３０を制御する。
（１）第１のオペレーション
定数レジスタ制御部３２は、内部の８個のセレクタ３２ａ～３２ｈを制御することで、図５（ｂ）に示された格納方法により、Ｐ１．０フィールド１２～Ｐ２．２フィールド１５に保持された１６ビットの定数（０ｘ８７６５）を定数レジスタ３６の下位１６ビットに格納する。その結果、図９のクロックサイクルｔ０～ｔ１に示されるように、定数レジスタＲ１５の内容は、それまでの“０ｘ００００００００”から“０ｘ００００８７６５”に変化する。
（２）第２のオペレーション
第２演算部３８は、汎用レジスタＲ０の内容(“０ｘ３３３３３３３３”）と汎用レジスタＲ１の内容（“０ｘ２２２２２２２２”）とを入力とし、ここで減算した後に、その結果を再び汎用レジスタＲ０に格納する。その結果、図９のクロックサイクルｔ０～ｔ１に示されるように、汎用レジスタＲ０の内容は、それまでの“０ｘ３３３３３３３３”から“０ｘ１１１１１１１１”に変化する。
（命令７２）
次に、命令７２が命令レジスタ１０にロードされると、本プロセッサは図９のクロックサイクルｔ１～ｔ２に示された動作をする。
【００４９】
フォーマットデコーダ２１は、上記命令７１の場合と同様に、命令レジスタ１０のＰ０．

０フィールド１１の値（“ｆｍｔ４”）から、この命令はフォーマットコードが“４”の２オペレーション命令であると判断し、以下の２つのオペレーションが並列実行されるように実行部３０を制御する。
（１）第１のオペレーション
定数レジスタ制御部３２は、内部の８個のセレクタ３２ａ～３２ｈを制御することで、図５（ｂ）に示された格納方法により、Ｐ１．０フィールド１２～Ｐ２．２フィールド１５に保持された１６ビットの定数（０ｘ４３２１）を定数レジスタ３６の下位１６ビットに格納する。その結果、図９のクロックサイクルｔ１～ｔ２に示されるように、定数レジスタＲ１５の内容は、それまでの“０ｘ００００８７６５”から“０ｘ８７６５４３２１”に変化する。
（２）第２のオペレーション
第２演算部３８は、汎用レジスタＲ２の内容（“０ｘ０００００００４”）と汎用レジスタＲ０の内容（“０ｘ１１１１１１１１”）とを入力とし、ここで乗算した後に、その結果を再び汎用レジスタＲ０に格納する。その結果、図９のクロックサイクルｔ１～ｔ２に示されるように、汎用レジスタＲ０の内容は、それまでの“０ｘ１１１１１１１１”から“０ｘ４４４４４４４４”に変化する。
（命令７３）
最後に、命令７３が命令レジスタ１０にロードされると、本プロセッサは図９のクロックサイクルｔ２～ｔ３に示された動作をする。
【００５０】
フォーマットデコーダ２１は、命令レジスタ１０のＰ０．０フィールド１１の値（“ｆｍｔ７”）から、この命令はフォーマットコードが“７”の２オペレーション命令であると判断し、以下の２つのオペレーションが並列実行されるように実行部３０を制御する。
（１）第１のオペレーション
第１演算部３７は、定数レジスタＲ１５の内容（“０ｘ８７６５４３２１”））値と汎用レジスタＲ０の内容（“０ｘ４４４４４４４４）とを入力とし、それらを加算した後に、その結果を再び汎用レジスタＲ０に格納する。その結果、図９のクロックサイクルｔ２～ｔ３に示されるように、汎用レジスタＲ０の内容は、それまでの“０ｘ４４４４４４４４”から“０ｘＣＢＡ９８７６５”に変化し、定数レジスタＲ１５の内容はクリアされる。
（２）第２のオペレーション
第２演算部３８は、Ｐ１．０フィールド１２とＰ３．１フィールド１７に分割して置かれた８ビットの定数（“０ｘ００”）を入力とし、そのまま通過させて、汎用レジスタＲ３に格納する。その結果、図９のクロックサイクルｔ２～ｔ３に示されるように、汎用レジスタＲ３の内容は、それまでの“０ｘＦＥＤＣＢＡ９８”から“０ｘ００００００００”に変化する。
【００５１】
以上のようにして、本プロセッサにおいて、３２ビットの定数“０ｘ８７６５４３２１”は、２個の命令７１、７２に跨って分割配置され、順次定数レジスタ３６にシフトされながら格納された後に、第３番目の命令７３によって利用された。このようにして、図７のフローチャートに示された処理が３個の命令７１～７３によって実行される。
次に、１６ビットの定数を扱う別のプログラムを用いて本プロセッサの動作を説明する。
【００５２】
図１０は、１６ビットの定数を扱うプログラムの例を示す図である。
このプログラムは、５個の命令７４～７８から構成されている。
各命令７１～７３ごとの本プロセッサの動作は以下の通りである。
（命令７４）
命令７４が命令レジスタ１０にロードされると、フォーマットデコーダ２１は、命令レジスタ１０のＰ０．０フィールド１１の値（“ｆｍｔ０”）から、この命令はフォーマットコードが“０”の３オペレーション命令であると判断し、以下の３つのオペレーションが並列実行されるように実行部３０を制御する。

（１）第１のオペレーション
定数レジスタ制御部３２は、内部の８個のセレクタ３２ａ～３２ｈを制御することで、図５（ａ）に示された格納方法により、Ｐ１．０フィールド１２に保持された４ビットの定数（“０ｘ８”）を定数レジスタ３６の最下位の４ビットレジスタ３６ｈに格納する。
（２）第２のオペレーション
第１演算部３７は、汎用レジスタＲ６の値を入力とし、そのまま通過させて、汎用レジスタＲ１に格納する。
（３）第３のオペレーション
同様に、第２演算部３８は、汎用レジスタＲ７の値を入力とし、そのまま通過させて、汎用レジスタＲ２に格納する。
（命令７５）
同様にして、命令７５が命令レジスタ１０にロードされると、フォーマットデコーダ２１は、この命令はフォーマットコードが“０”の３オペレーション命令であると判断し、以下の３つのオペレーションが並列実行されるように実行部３０を制御する。
（１）第１のオペレーション
定数レジスタ制御部３２は、内部の８個のセレクタ３２ａ～３２ｈを制御することで、図５（ａ）に示された格納方法により、Ｐ１．０フィールド１２に保持された４ビットの定数（“０ｘ７”）を定数レジスタ３６の最下位４ビットレジスタ３６ｈに格納する。この結果、定数レジスタ３６の下位８ビットには定数“０ｘ８７”がセットされる。
（２）第２のオペレーション
第１演算部３７は、汎用レジスタＲ０とＲ１の値を入力とし、ここで加算した後に、その結果を再び汎用レジスタＲ１に格納する。
（３）第３のオペレーション
同様に、第２演算部３８は、汎用レジスタＲ０とＲ２の値を入力とし、ここで加算した後に、その結果を再び汎用レジスタＲ２に格納する。
（命令７６、命令７７）
同様にして、命令７６、７７が実行されることにより、定数レジスタ３６の下位１６ビットには定数“０ｘ８７６５”がセットされる。
（命令７８）
命令７８が命令レジスタ１０にロードされると、本プロセッサは、図８に示された命令７３の場合と同様の動作をする。
【００５３】
以上のようにして、本プロセッサにおいては、１６ビットの定数“０ｘ８７６５”は、４個の命令７４～７７に跨って分割配置され、順次定数レジスタ３６にシフトされながら格納された後に、第５番目の命令７８によって利用された。
（通常のプロセッサとの比較）
次に、上記図８及び図１０に示されたプログラムと同一内容の処理を通常のプロセッサに行なわせた場合について説明し、本発明に係るプロセッサと比較する。なお、ここでいう通常のプロセッサとは、本発明に係るプロセッサの定数レジスタ３６や定数レジスタ制御部３２の如く、分割された定数を蓄積して格納する手段のみを有しないプロセッサをいい、３２ビット固定長の命令を実行するものとする。
【００５４】
図２１（ａ）は、この通常のプロセッサが実行する命令のフィールド定義を示し、図２１（ｂ）は、その命令のフォーマットを示す。つまり、通常のプロセッサは、３種類の２オペレーション命令１０１～１０３と１種類の１オペレーション命令１０４を実行するものとする。
図２２は、図８に示されたプログラムと同一内容の処理、即ち、図７のフローチャートに示された処理を通常のプロセッサに行なわせるプログラムの例である。
【００５５】
図２２と図８とを比較して判るように、通常のプロセッサ用のプログラムは、本発明に係

るプロセッサ用のものよりも２個の命令だけ多くなっている。
なお、命令１０５、１０６にｎｏｐコードが含まれるのは、命令１０６は命令１０５での演算結果を用いるので、これらの命令を並列に実行させることができないからである。また、１個の定数“０ｘ８７６５４３２１”を上位１６ビットと下位１６ビットの２つに分割して定数レジスタＲｉにセットしているのは（命令１０７、１０８）、３２ビットの１個の命令の中に、セット命令のオペコードと３２ビットの定数の両方を同時に配置することは不可能だからである。
【００５６】
同様に、図２３は、図１０に示されたプログラムと同一内容の処理を通常のプロセッサに行なわせるプログラムの例である。
図２３と図１０とを比較して判るように、通常のプロセッサ用のプログラムは、本発明に係るプロセッサ用のものよりも１個の命令だけ多くなっている。
以上のように、本発明に係るプロセッサによれば、１６ビットや３２ビットの定数が複数の命令に跨って分割配置されていても、それらは定数レジスタ３６に蓄積して格納されることで元の定数に復元され、分岐や算術演算等のオペレーションに使用される。つまり、命令中に生じた小さな領域であっても、定数を分割して埋めておくことができるので、通常のプロセッサに実行させる場合よりもプログラムのコードサイズは縮小される。
（定数レジスタ３６の周辺回路の変形例）
次に、図４に示された定数レジスタ３６の周辺回路についての変形例のいくつかを示す。
（第１の変形例）
図１１は、第１の変形例に係る定数レジスタ３６の周辺回路の構成及び接続関係を示すブロック図である。
【００５７】
図４の定数レジスタ制御部３２は定数レジスタ３６をシフトレジスタとして機能させたが、図１１の定数レジスタ制御部３２は定数レジスタ３６を並列入力のレジスタとして機能させている点で異なる。具体的には、定数レジスタ制御部９０とフォーマットデコーダ２１との接続及び定数レジスタ制御部９０の構成要素が図４に示されたものと異なる。
【００５８】
定数レジスタ制御部９０は、格納桁カウンタ９１と８個の８入力セレクタ９０ａ～９０ｈとからなる。格納桁カウンタ９１は、３ビットのカウンタであり、その時点において定数レジスタ３６に蓄積されている定数の有効桁数をニブル（４ビット）単位で示すものである。
定数レジスタ制御部９０は、命令レジスタ１０に保持された定数を格納する旨の指示をフォーマットデコーダ２１から受けると、その時点での格納桁カウンタ９１の値を参照することで定数レジスタ３６の適切な位置にその定数を格納し、その後に、格納桁カウンタ９１を更新しておく。また、定数レジスタ３６の格納値が何らかのオペレーションのために読み出されたした旨の通知をフォーマットデコーダ２１又は命令デコーダ２２から受けると、固定値（“０”）９２からのオールゼロを定数レジスタ３６に格納し、その後に、格納桁カウンタ９１をリセットしておく。
【００５９】
具体的には、定数レジスタ制御部９０は、Ｐ０．０フィールド１１の値が“０”又は“１”である旨の指示をフォーマットデコーダ２１から受けると、その時の格納桁カウンタ９１の値０～７に対応して、それぞれ図１２（ａ）～図１２（ｈ）に示された格納方法によりＰ１．０フィールド１２に置かれた１ニブルの定数を定数レジスタ３６に格納するようセレクタ９０ａ～９０ｈを制御し、その後に格納桁カウンタ９１を１だけインクリメントする。
【００６０】
同様に、定数レジスタ制御部９０は、Ｐ０．０フィールド１１の値が“４”である旨の指示をフォーマットデコーダ２１から受けると、その時点での格納桁カウンタ９１の値０～７に対応して、それぞれ図１３（ａ）～図１３（ｈ）に示された格納方法により、Ｐ１．

０フィールド１２～Ｐ２．２フィールド１５に置かれた４ニブルの定数を定数レジスタ３６に格納するようセレクタ９０ａ～９０ｈを制御し、その後に格納桁カウンタ９１を４だけインクリメントする。

【００６１】

同様に、定数レジスタ制御部９０は、Ｐ０．０フィールド１１の値が“５”である旨の指示をフォーマットデコーダ２１から受けると、その時点での格納桁カウンタ９１の値０～７に対応して、それぞれ図１４（ａ）～図１４（ｈ）に示された格納方法により、Ｐ１．０フィールド１２とＰ３．０フィールド１６～Ｐ３．２フィールド１８に置かれた４ニブルの定数を定数レジスタ３６に格納するようセレクタ９０ａ～９０ｈを制御し、その後に格納桁カウンタ９１を４だけインクリメントする。

【００６２】

さらに、定数レジスタ制御部９０は、Ｐ０．０フィールド１１の値が“２”、“３”及び“Ａ”のいずれかである旨の指示をフォーマットデコーダ２１からうけた場合又は定数レジスタ３６の格納値をオペランドとするオペレーションである旨の指示を命令デコーダ２２から受けた場合には、図１５に示された定数レジスタ３６のように、３２ビットの定数“０”を定数レジスタ３６に格納させ、その後、格納桁カウンタ９１をゼロにクリアする。

【００６３】

以上のように、この第１の変形例に係る定数レジスタ制御部９０によって、上記定数レジスタ制御部３２の場合と同様に、複数の命令に跨って分割配置されていた定数は定数レジスタ３６に蓄積して格納されることで元の定数に復元される。

但し、上記定数レジスタ制御部３２の場合と異なり、後で実行される命令に置かれた定数が定数レジスタ３６の上位桁に格納される。従って、この変形例のプロセッサに図７のフローチャートに示された処理を実行させるためには、図８に示されたプログラムにおいて、命令７１の定数“０ｘ８７６５”と命令７２の定数“０ｘ４３２１”とを入れ替えておく必要がある。

（第２の変形例）

次に、第２の変形例を示す。

【００６４】

図１６は、第２の変形例に係る定数レジスタ３６の周辺回路の構成及び接続関係を示すブロック図である。

この第２の変形例は、定数レジスタ３６の格納値をシフトさせながら新たな定数を蓄積していく点で図４に示されたものと同じであるが、定数レジスタ３６への格納に先立って定数を符号拡張している点及び定数レジスタ３６の格納値が読み出された場合であっても定数レジスタ３６をクリアしない点で異なる。

【００６５】

なお、符号拡張とは、ある数値の有効桁数が一定の桁数に満たない場合に、その数値の最上位ビットを符号ビットとみなし、その符号ビットと同じ論理値で上位桁を埋める処理をいう。

符号拡張制御部１９１は、命令レジスタ１０に置かれた定数が、定数レジスタ３６の格納値が読み出された後の初めての格納に係るものである場合には、その定数を符号拡張した後に定数レジスタ３６に格納するよう定数レジスタ制御部１９０を制御し、そうでない場合には、その定数を符号拡張することなく定数レジスタ３６に格納するよう定数レジスタ制御部１９０を制御する。そのために、符号拡張制御部１９１は、内部に１ビットの読み出しフラグを記憶しておくための読み出しフラグ記憶部１９２を備え、フォーマットデコーダ２１及び命令デコーダ２２からの指示に基づいて読み出しフラグ記憶部１９２を更新する。

【００６６】

図１７は、読み出しフラグ記憶部１９２の値の変化を示す状態遷移図である。符号拡張制御部１９１は、定数レジスタ３６の格納値が読み出された旨の通知をフォーマットデコー

ダ２１又は命令デコーダ２２から受けると、その読み出し後に読み出しフラグ記憶部１９２の値を“１”にセットし、命令レジスタ１０に置かれた定数を定数レジスタ３６に格納する旨の通知をフォーマットデコーダ２１から受けると、その格納後に読み出しフラグ記憶部１９２の値を“０”にセットする。そして、符号拡張制御部１９１は、新たな定数を定数レジスタ３６に格納させる際に読み出しフラグ記憶部１９２を参照し、その値が“１”である場合には、その定数を符号拡張した後に定数レジスタ３６に格納させる。

【００６７】

なお、図１６において、セレクタ１９０ｈ及びセレクタ１９０ｅの出力から分岐されて他のセレクタに配線されている信号線は、それぞれの出力４ビットのうちの最上位ビットに相当する信号線だけである。

図１８（ａ）～図１８（ｆ）は、この第２の変形例における定数レジスタ３６及び読み出しフラグ記憶部１９２の値の変化を示す図である。

【００６８】

図１８（ａ）は、定数レジスタ３６に格納されていた定数“０ｘ８７６５４３２１”が読み出された直後におけるそれらの内容を示し、図１８（ｂ）～図１８（ｅ）は、その後に４ビットの定数“０ｘ８”、“０ｘ７”、“０ｘ６”、“０ｘ５”が順次に格納された直後におけるそれらの内容を示し、図１８（ｆ）は、図１８（ｅ）における定数“０ｘＦＦＦＦ８７６５７”が読み出された直後におけるそれらの内容を示す。

【００６９】

このようにして、本第２の変形例において、符号拡張制御部１９１が読み出しフラグ記憶部１９２によって符号拡張の必要性の有無を管理することで、複数の命令に跨って分割配置されていた定数は、定数レジスタ３６に蓄積して格納され、符号拡張された元の定数として復元される。

（第３の変形例）

次に、第３の変形例を示す。

【００７０】

図１９は、第３の変形例に係る定数レジスタ３６の周辺回路の構成及び接続関係を示すブロック図である。

この第３の変形例は、上記第２の変形例と比較し、定数レジスタ３６の格納値をシフトさせながら新たな定数を蓄積していく点及び符号拡張を行なう点において共通するが、定数レジスタ３６に定数を格納する際に拡張処理を行なうのではなく、定数レジスタ３６から定数を読み出す際に拡張処理を行なう点で異なる。

【００７１】

また、図４に示された周辺回路及び接続関係と比較すると、この第３の変形例では、定数レジスタ３６にオールゼロを格納するための構成は設けられておらず、新たに拡張制御部２９１及びゼロ／符号拡張部２９３が付加されている。

定数レジスタ制御部２９０は、定数レジスタ３６をクリアすることがない点を除いて、図４に示された定数レジスタ制御部３２と同様の動作を行なう。

【００７２】

拡張制御部２９１は、定数レジスタ３６に格納されていた定数が読み出される際に、その定数が適切にゼロ拡張又は符号拡張が行われるように、ゼロ／符号拡張部２９３を制御する。そのために、拡張制御部２９１は、内部に３ビットのカウンタからなる格納桁カウンタ２９２を有する。格納桁カウンタ２９２は、定数レジスタ３６に蓄積されている定数の有効桁数をニブル単位で記憶しておくためのものである。拡張制御部２９１は、新たな定数が定数レジスタ３６に格納された旨の通知をフォーマットデコーダ２１から受けた場合には、その定数の桁数（ニブル単位の桁数）だけ格納桁カウンタ２９２をインクリメントし、定数レジスタ３６の格納値が読み出された旨の通知をフォーマットデコーダ２１又は命令デコーダ２２から受けた場合には、格納桁カウンタ２９２をリセットする。

【００７３】

この符号拡張制御部２９１は、命令デコーダ２２からの指示に基づいて拡張形式（ゼロ拡

張か符号拡張か）を指定する１ビットの制御信号と、格納桁カウンタ２９２の値を示す３ビットの制御信号とをゼロ／符号拡張部２９３に出力する。
ゼロ／符号拡張部２９３は、定数レジスタ３６から出力された３２ビットの格納値を入力とし、拡張制御部２９１から通知される拡張形式と有効桁数（ニブル単位の桁数）に基づいて、ゼロ拡張又は符号拡張を行なう。具体的には、符号拡張制御部２９１から指定された拡張形式が符号拡張の場合であれば、入力された格納値に対して、拡張制御部２９１から通知された有効桁数の最上位に相当するビットの論理値をその有効桁数より上位の全ての桁にコピーして得られる値を出力し、一方、ゼロ拡張の場合であれば、それら上位の全ての桁をゼロとする値を出力する。
【００７４】
例えば、入力された格納値が“０ｘ８７６５４３２１”であり、拡張制御部２９１から通知された拡張形式が符号拡張であって有効桁数が４ニブルである場合には、ゼロ／符号拡張部２９３は“０ｘ００００４３２１”を出力する。
このように、本第３の変形例によって、複数の命令に跨って分割配置されていた定数は、定数レジスタ３６に蓄積して格納されることで元の定数に復元され、ゼロ拡張又は符号拡張された後に分岐や演算等のオペレーションに利用される。
（第４の変形例）
次に、第４の変形例を示す。
【００７５】
図２０は、第４の変形例に係る定数レジスタ３６の周辺回路の構成及び接続関係を示すブロック図である。
この第４の変形例は、図４に示されたものと比較し、定数レジスタ３６の格納値をシフトさせながら新たな定数を蓄積していく点において共通するが、シフトさせる方向と定数レジスタ３６への格納位置が異なる。具体的には、定数レジスタ３６の格納値を上位桁から下位桁へシフトさせると共に、命令レジスタ１０に置かれた新たな定数を定数レジスタ３６の上位桁に格納する。なお、定数レジスタ３６の格納値が読み出された直後に定数レジスタ３６をクリアする点は図４のものと同じである。
【００７６】
図１０と図４を比較して判るように、これらは、接続関係が桁方向に対称的になっており、用いられている構成要素は同一である。
この第４の変形例によって、より少ない格納回数によって下位桁全てをゼロとする数値を定数レジスタ３６にセットすることが可能となる。例えば、定数レジスタ３６がクリアされた後に新たな３２ビットの定数“０ｘ８７６５００００”をセットする場合に、図４に示された接続であれば、２個の１６ビット定数“０ｘ８７６５”と“０ｘ００００”を格納する必要があるが、図２０に示された接続であれば、１個の１６ビット定数“０ｘ８７６５”を格納するだけでよい。小数点以下を全てゼロとする固定小数点数値をセットする場合に便利である。
以上、本発明に係るプロセッサについて、実施形態及び変形例に基づいて説明したが、本発明はこれら実施形態及び変形例に限られないことは勿論である。即ち、
（１）定数レジスタ３６に定数を格納する位置として、下位桁から埋めていく方式（例えば、図４に示された方式）と、上位桁から埋めていく方式（例えば、図２０に示された方式）がある。
【００７７】
また、新たな定数を定数レジスタ３６に格納する際に、既に格納されている定数をシフトさせていく方式（例えば、図４に示された方式）と、シフトさせずに異なる桁位置に格納していく方式（例えば、図１１に示された方式）とがある。
また、定数の上位桁の拡張方式として、ゼロ拡張を行なう方式（例えば、図４に示された方式）と、符号拡張を行なう方式（例えば、図１６に示された方式）とがある。
【００７８】
また、ゼロ拡張や符号拡張を行なうタイミングとして、定数レジスタ３６への格納前に行

なう方式（例えば、図１６に示された方式）と、定数レジスタ３６からの読み出し後に行なう方式（例えば、図１９に示された方式）とがある。
また、ゼロ拡張を行なう具体的な実現方式として、定数レジスタ３６の格納値が読み出された直後にオールゼロを定数レジスタ３６に格納しておく方式（例えば、図４に示された方式）と、定数の格納時や読み出し時に固定値“０”を挿入する方式（例えば、図１９に示された方式）とがある。
【００７９】
従って、これら格納位置、シフトの有無、拡張方式、拡張のタイミング、読み出し後の自動クリアについての各方式を組み合わせることで、さらに多くのバリエーションを構築することは容易である。
さらに、上記実施形態では、上記各方式の一方だけを備える例が示されたが、これに限定されるものではない。例えば、定数のゼロ拡張を行なうための構成と符号拡張を行なうための構成の両方を備え、命令に置かれたオペコードによって、いずれかを選択的に動作させる方式であってもよい。
（２）また、上記実施の形態では、数値定数を扱う例が示されたが、文字定数であってもよいことは言うまでもない。複数の命令に跨って分割配置された文字定数であっても、定数レジスタ３６への複数回の格納によって、桁数の長い元の文字定数が復元されるからである。
（３）また、上記実施の形態では、図１（ｂ）～図１（ｄ）の命令フォーマットから判るように、１個の命令によって定数レジスタ３６に格納させることができる定数の桁数は４ビット及び１６ビットのいずれかであったが、本発明はこの桁数に限定されるものではない。例えば、１２ビットや２８ビットの定数を定数レジスタ３６に格納するための命令フォーマットを定義してもよい。そのためには、定数レジスタ３６の周辺回路の接続関係を変更すればよい。
（４）また、上記実施の形態のプロセッサは２個の演算部３７、３８を備えるＶＬＩＷプロセッサであったが、本発明は、１個の演算部のみを備え１個の命令中に１個のオペレーションだけを指定する単一オペレーション命令を実行するＶＬＩＷアーキテクチャを採らないプロセッサに対しても適用することができるのは言うまでもない。
【００８０】
特に、固定長命令の場合には未使用領域を持つ命令が多く定義されることがある。例えば、ＭＩＰＳ社製のＲＩＳＣプロセッサ“Ｒ２０００”は３２ビット固定長命令を実行するが、このプロセッサの命令セットには、未使用領域を持つ命令（例えば、ＡＤＤ命令は５ビットの未使用領域を有する。）が定義されている。本発明によれば、このような単一オペレーション命令中に生じる無駄領域の発生が回避される。
（５）また、上記実施の形態では、４ビットのオペコード“ｃｃ”は、暗黙的に定数レジスタ３６の格納値を参照する間接分岐オペレーションを意味したが、本発明はこれに限定されるものではなく、オペコード“ｃｃ”として、固定的なディスプレースメントの番地分だけ相対的に分岐するオペレーションや、汎用レジスタ３５の格納値を用いた間接分岐オペレーション等を含ませることもできる。この場合には、定数レジスタ制御部３２は、定数レジスタ３６の格納値が読み出された直後に定数レジスタ３６をクリアするだけでなく、オペコード“ｃｃ”に伴う分岐オペレーションが実行された直後にも定数レジスタ３６をクリアすればよい。これらのことは、どのようなオペレーションを４ビットのオペコード“ｃｃ”に割り当てるかの設計事項であり、分岐デコーダ２３と定数レジスタ制御部３２の変更によって容易に実現できることは言うまでもない。
【００８１】
【発明の効果】
以上の説明から明らかなように、本発明は、命令を解読し実行するプロセッサであって、定数を格納するための定数レジスタと、前記命令中に前記定数レジスタへ格納すべき定数が置かれていることを解読する解読手段と、前記解読手段により前記定数が置かれていると解読された場合には、前記定数レジスタに定数が格納されていないと判断すると前記定

数を所定位置に格納し、前記定数レジスタに定数が既に格納されていると判断すると前記定数レジスタに既に格納されている定数を残したまま新たな定数を前記定数レジスタに格納する格納手段と、前記定数レジスタに格納されている全ての定数を読み出し、その定数をオペランドとするオペレーションを実行する実行手段とを備えることを特徴とする。

【００８２】

これによって、コンパイラ等によって複数の命令に跨って分割配置されていた定数の断片が定数レジスタに格納されて蓄積され、元の１個の定数に復元される。従って、命令中に小さな桁数の無駄領域が生じる場合であっても、その無駄領域を桁数の大きい定数の一部で埋めておくことが可能となり、コード効率の高いプログラムの作成が支援される。

【００８３】

ここで、前記格納手段は、前記定数レジスタに既に格納されている定数をシフトさせながら前記新たな定数を格納していくとすることもできる。

これによって、定数レジスタの同じ位置に定数を格納するだけで定数が蓄積されるので、次に格納すべき定数レジスタの位置を管理する必要がなくなる。

また、前記格納手段は、前記定数レジスタに既に格納されている定数を移動させることなく、その定数に隣接する位置に前記新たな定数を格納していくとすることもできる。

【００８４】

これによって、一旦、定数レジスタに格納された定数はその桁位置が変動されることがないので、１個の定数を複数の命令の無駄領域に分割して配置するためのコンパイラにおけるスケジューリングが容易になる。

また、前記格納手段は、前記定数レジスタの下位桁から上位桁に向けて順次に定数を蓄積して格納していくとすることもできる。

【００８５】

これによって、定数レジスタは下位桁から埋められていくので、有効桁が可変長の定数を定数レジスタにセットするのに好適なプロセッサが実現される。

また、前記格納手段は、前記定数レジスタの上位桁から下位桁に向けて順次に定数を蓄積して格納していくとすることもできる。

これによって、定数レジスタは上位桁から埋められていくので、下位の桁全てをゼロとする定数を定数レジスタにセットするのに好適なプロセッサが実現される。

【００８６】

また、前記プロセッサはさらに、前記定数に対してゼロ拡張及び符号拡張の少なくとも一方の拡張処理を行なう拡張手段を備えるとすることもできる。

これによって、定数レジスタに格納された定数がオペランドとして利用される場合には、そのオペランドはゼロ拡張又は符号拡張が施された適切な定数となっていることが保証される。

【００８７】

また、前記拡張手段は、前記定数レジスタに格納される前の定数に対して前記拡張処理を行なうとすることもできる。

これによって、定数レジスタに格納されている定数は常に拡張処理が行われた適切な定数であることが保証され、定数レジスタから定数を読み出した後に拡張処理を行なう必要がなくなるので、定数レジスタに格納された定数を用いたオペレーションに要する時間が短縮される。

【００８８】

また、前記拡張手段は、前記定数レジスタから読み出された定数に対して前記拡張処理を行なうとすることもできる。

これによって、定数を定数レジスタに格納する際における拡張処理が不要となるので、定数レジスタへの格納に要する時間が短縮される。

また、前記格納手段は、前記実行手段によって前記定数レジスタに格納された定数が読み出された直後に前記定数レジスタにゼロを格納するとすることもできる。

【００８９】

これによって、定数を定数レジスタに格納するだけで定数レジスタに格納されている定数は常にゼロ拡張された適切な定数であることが保証され、定数レジスタの格納値を使用する度に定数レジスタをクリアしておくための独立した消去命令は不要となる。
また、前記実行手段は、前記命令に従って分岐オペレーションを実行する分岐実行部を有し、前記格納手段は、前記分岐実行部により分岐オペレーションが実行された場合に前記定数レジスタにゼロを格納するとすることもできる。

【００９０】

これによって、分岐オペレーションの実行に伴って定数レジスタの格納値はクリアされるので、定数レジスタに不要な格納値が残されることによる不具合が回避される。
また、並列実行する複数のオペレーションからなる命令を実行するＶＬＩＷプロセッサにおいて、並列実行できるオペレーションがない場合、オペレーションの代わりに後続の命令で使用する定数が配置された命令を実行するとすることもできる。

【００９１】

これによって、単一オペレーションを指定する命令中に生じる無駄領域だけでなく、１個の命令中に２個以上のオペレーションを指定することができるＶＬＩＷ中に生じる無駄領域に対しても、定数を埋めておくことが可能となる。
また、並列実行する複数のオペレーションからなる命令を実行するＶＬＩＷプロセッサにおいて、並列実行できるオペレーションがない場合、オペレーションの代わりに定数が配置された命令のうち前記定数以外のオペレーションを並列実行するとともに、前記定数は順に格納しておき後続の命令で使用するとすることもできる。

【００９２】

これによって、ＶＬＩＷの一部のオペレーションのみに定数が置かれている場合であっても、その定数の格納と他のオペレーションとは並列に実行される。そして、その定数は蓄積されて格納され、後続の命令で利用されるので、桁数の大きい定数であっても、それを分割して複数の命令に埋めておくことが可能となる。

【００９３】

また、命令のフォーマットを指定するフォーマットコードが置かれるフォーマットフィールドと、並列実行させるオペレーションを指定する複数のオペレーションフィールドとを含む命令を解読し実行するＶＬＩＷプロセッサであって、定数を格納するための定数レジスタと、前記フォーマットコードを参照することにより、少なくとも１つの前記オペレーションフィールドに定数が置かれていることを解読する解読手段と、前記解読手段により前記定数が置かれていると解読された場合には、前記定数レジスタに定数が格納されていないと判断すると前記定数を所定位置に格納し、前記定数レジスタに定数が既に格納されていると判断すると前記定数レジスタに既に格納されている定数を残したまま新たな定数を前記定数レジスタに格納する格納手段と、前記定数レジスタに格納されている全ての定数を読み出し、その定数をオペランドとするオペレーションを実行する実行手段とを備えるとすることもできる。

【００９４】

これによって、プロセッサに実行させるＶＬＩＷ列を生成する際に桁数の大きい定数を複数のＶＬＩＷに跨って分割配置しておくことが可能となる。そして、定数が置かれている命令中の位置（オペレーションフィールド）はフォーマットコードによって明示的に指定されるが、定数の格納先（定数レジスタ）や格納位置（定数レジスタ上の桁位置）は暗黙的に指定されたものであるので、命令中に置かれた定数を定数レジスタの特定位置に格納するための明示的なオペコードは不要になる。

【００９５】

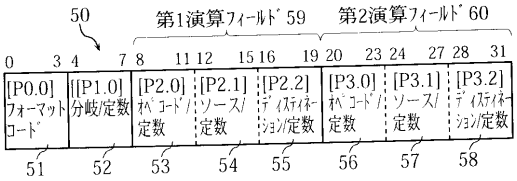
以上のように、本発明によって、コンパイラ等によって機械語命令列を生成する際に、それら命令列中に小さな無駄領域を設ける必要が生じた場合であっても、後続の命令で使用される定数を予め分割して埋めておくという最適化スケジューリングが可能となるので、プログラムのコードサイズは縮小化され、特に組み込み用途のプロセッサとしてその実用的価値は大きい。

【図面の簡単な説明】
【図１】図１（ａ）は、本発明に係るプロセッサが実行する命令のフィールド構成を示す図である。
図１（ｂ）～図１（ｄ）は、１６種類の命令フォーマットを示す図である。
図１（ｂ）は３オペレーション、図１（ｃ）は２オペレーション、図１（ｄ）は１オペレーションを同時に指定できる命令フォーマットである。
【図２】図１で用いられている３種類のオペコード“ｃｃ”、“ｏｐ１”及び“ｏｐ２”それぞれによって指定される具体的なオペレーションを説明する図である。
【図３】同プロセッサのハードウェア構成を示すブロック図である。
【図４】同プロセッサの定数レジスタ３６及びその周辺回路の詳細な構成を示すブロック図である。
【図５】図４に示された定数レジスタ制御部３２による定数の格納方法を示す図である。
図５（ａ）はフォーマットコードが“０”又は“１”である場合、図５（ｂ）はフォーマットデコードが“４”である場合、図５（ｃ）はフォーマットデコードが“５”である場合、図５（ｄ）はフォーマットコードが“２”、“３”及び“Ａ”のいずれかである場合又は定数レジスタ３６の格納値がオペランドとして指定されている場合の格納方法を示す。
【図６】同プロセッサのＰＣ部３３の詳細な構成を示すブロック図である。
【図７】３２ビットの定数を扱う処理の一例を示すフローチャートである。
【図８】図７に示された処理を同プロセッサに行なわせるプログラムの例を示す図である。
【図９】図９は、図８に示されたプログラムを実行した場合の本プロセッサの動作を示すタイミングチャートである。
【図１０】１６ビットの定数を扱う処理を同プロセッサに行なわせるプログラムの例を示す図である。
【図１１】第１の変形例に係る定数レジスタ３６の周辺回路の構成を示すブロック図である。
【図１２】フォーマットコードが“０”又は“１”である場合の定数レジスタ制御部９０による定数の格納方法を示す図である。
図１２（ａ）～図１２（ｈ）は、それぞれ格納桁カウンタ９１の値が０～７のときの格納方法を示す。
【図１３】フォーマットコードが“４”である場合の定数レジスタ制御部９０による定数の格納方法を示す図である。
図１３（ａ）～図１３（ｈ）は、それぞれ格納桁カウンタ９１の値が０～７のときの格納方法を示す。
【図１４】フォーマットコードが“５”である場合の定数レジスタ制御部９０による定数の格納方法を示す図である。
図１４（ａ）～図１４（ｈ）は、それぞれ格納桁カウンタ９１の値が０～７のときの格納方法を示す。
【図１５】フォーマットコードが“２”、“３”及び“Ａ”のいずれかである場合又は定数レジスタ３６の格納値がオペランドとして指定されている場合の定数レジスタ制御部９０による定数の格納方法を示す図である。
【図１６】第２の変形例に係る定数レジスタ３６の周辺回路の構成を示すブロック図である。
【図１７】図１６に示された読み出しフラグ記憶部１９２の値の変化を示す遷移図である。
【図１８】同変形例における定数レジスタ３６及び読み出しフラグ記憶部１９２の値の変化を示す図である。
図１８（ａ）は、定数レジスタ３６に格納されていた定数“０ｘ８７６５４３２１”が読み出された直後における内容を示し、図１８（ｂ）～図１８（ｅ）は、その後に４ビット

の定数“０ｘ８”、“０ｘ７”、“０ｘ６”、“０ｘ５”が順次に格納された直後における内容を示し、図１８（ｆ）は、図１８（ｅ）における定数“０ｘＦＦＦＦ８７６５７”が読み出された直後における内容を示す。
【図１９】第３の変形例に係る定数レジスタ３６の周辺回路の構成を示すブロック図である。
【図２０】第４の変形例に係る定数レジスタ３６の周辺回路の構成を示すブロック図である。
【図２１】図２１（ａ）は、通常のプロセッサが実行する命令のフィールド定義を示す図である。図２１（ｂ）は、同命令フォーマットを示す図である。
【図２２】図８に示されたプログラムと同一内容の処理を上記通常のプロセッサに行なわせるプログラムの例を示す図である。
【図２３】図１０に示されたプログラムと同一内容の処理を上記通常のプロセッサに行なわせるプログラムの例を示す図である。
【図２４】従来技術における命令フォーマットを示す図である。
【符号の説明】
１０　命令レジスタ
２０　解読部
２１　フォーマットデコーダ
２２　命令デコーダ
２３　分岐デコーダ
２４　第１演算デコーダ
２５　第２演算デコーダ
３０　実行部
３１　実行制御部
３２　定数レジスタ制御部
３２ａ～３２ｈ　セレクタ
３３　ＰＣ部
３３ａ　固定値“４”
３３ｂ、３３ｅ　セレクタ
３３ｃ　加算器
３３ｄ　ＰＣ
３４　レジスタ群
３５　汎用レジスタＲ０～Ｒ１４
３６　定数レジスタＲ１５
３６ａ～３６ｈ　４ビット幅レジスタ
３７　第１演算部
３８　第２演算部
３９　命令フェッチ部
４０　オペランドアクセス部
４１　セレクタ
５０　命令
５１～５８　命令フィールド
５９　第１演算フィールド
６０　第２演算フィルド
９０、１９０、２９０、３９０　変形例に係る定数レジスタ制御部
９１、２９２　格納桁カウンタ
１９１　符号拡張制御部
２９１　拡張制御部
１９２　読み出しフラグ記憶部
２９３　ゼロ／符号拡張部

【図１】

(a)



(b)

0	const	op1	Rs1	Rd1	op2	Rs2	Rd2
1	const	op1	const1	Rd1	op2	Rs2	Rd2
2	cc	op1	Rs1	Rd1	op2	Rs2	Rd2
3	cc	op1	const1	Rd1	op2	Rs2	Rd2

(c)

4	const	const	const	const	op2	Rs2	Rd2
5	const	op1	Rs1	Rd1	const	const	const
6	const1	op1	const1	Rd1	op2	Rs2	Rd2
7	const2	op1	Rs1	Rd1	op2	const2	Rd2
8	const1	op1	const1	Rd1	op2	const2	Rd2
9	const2	op1	const1	Rd1	op2	const2	Rd2
A	cc	const2	const2	const2	op2	const2	Rd2

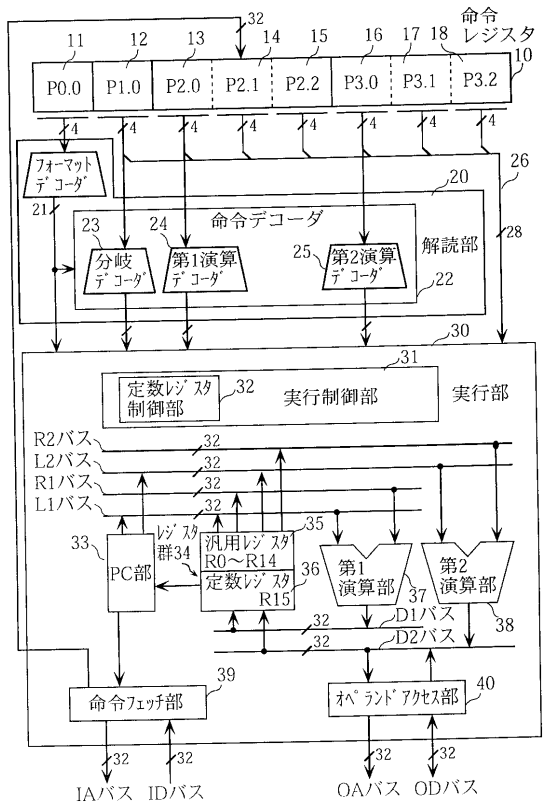
(d)

B	const2	const2	const2	const2	op2	const2	const2
C	reserved						
D	reserved						
E	reserved						
F	reserved						

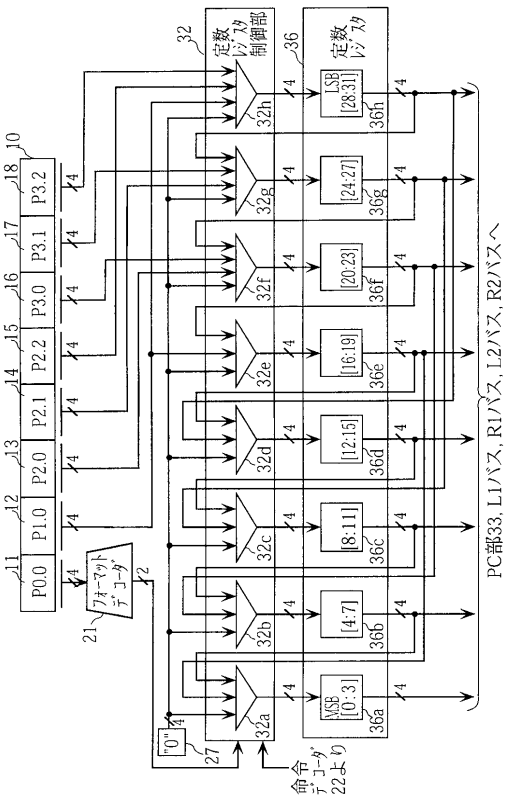
【図２】

記号	オペレーション	ニーモニック表示
cc	分岐	eq, eqi, ne, nei, gt, gti, ・・・
op1	算術論理演算	add, sub, mul, and, or, ・・・
	レジスタ間転送	mov, movh, movb
op2	算術論理演算	add, sub, mul, and, or, ・・・
	レジスタ間転送	mov, movh, movb
	レジスタ・メモリ間転送	ld, ldh, ldb, st, sth, stb

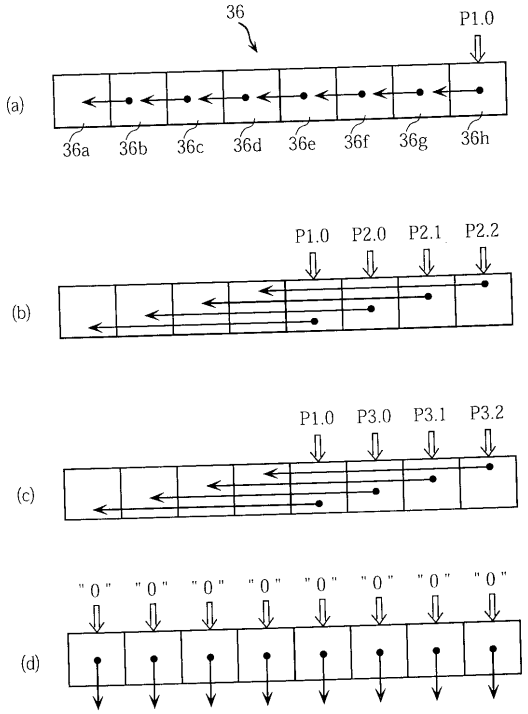
【図３】



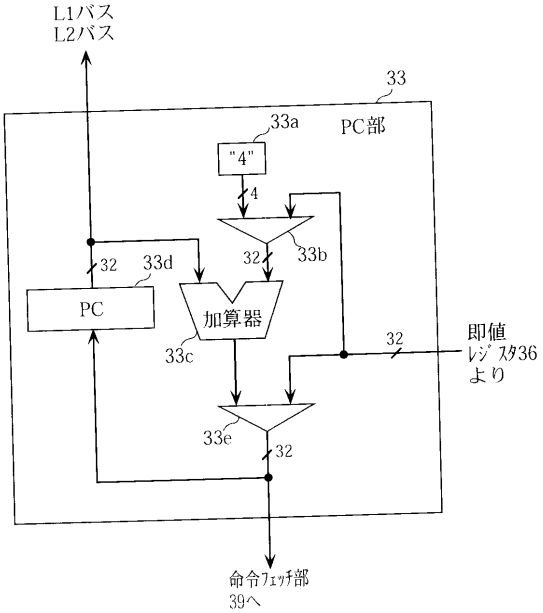
【図４】



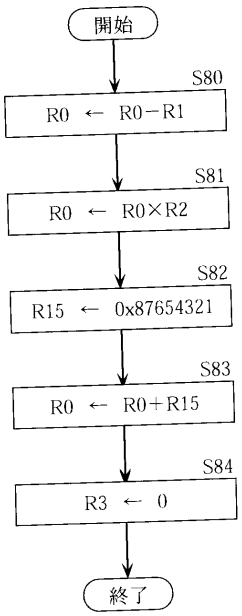
【図５】


36
P1.0
(a)
36a
36b
36c
36d
36e
36f
36g
36h
P1.0 P2.0 P2.1 P2.2
(b)
P1.0 P3.0 P3.1 P3.2
(c)
"0" "0" "0" "0" "0" "0" "0" "0"
(d)

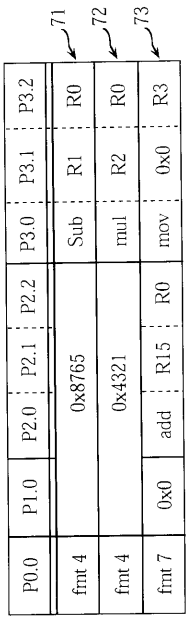
【図６】


L1バス
L2バス
33
PC部
33a
"4"
4
32
33b
32
33d
PC
加算器
33c
32
即値
ﾚｼﾞｽﾀ36
より
33e
32
命令ﾌｪｯﾁ部
39へ

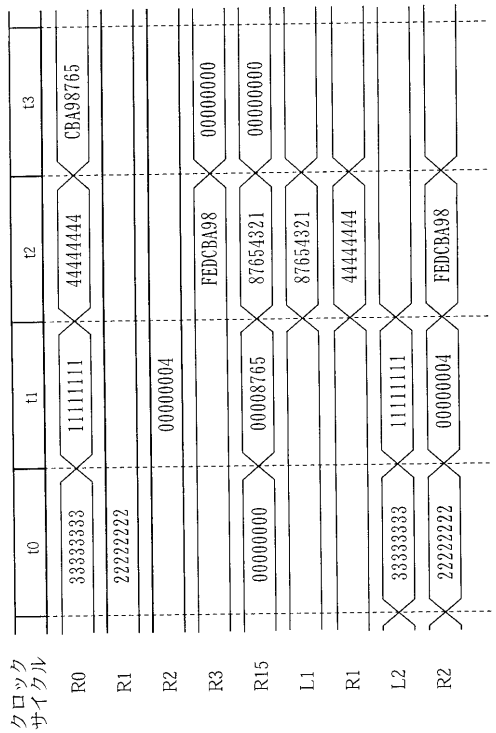
【図７】


開始
S80
R0 ← R0−R1
S81
R0 ← R0×R2
S82
R15 ← 0x87654321
S83
R0 ← R0+R15
S84
R3 ← 0
終了

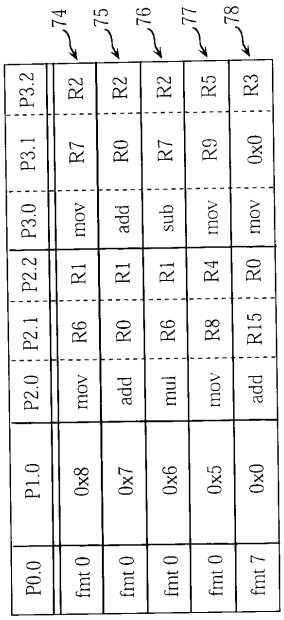
【図８】


71
72
73
P3.2
P3.1
P3.0
P2.2
P2.1
P2.0
P1.0
P0.0
R0
R1
Sub
0x8765
fmt 4
R0
R2
mul
0x4321
fmt 4
R3
0x0
mov
R0
R15
add
0x0
fmt 7

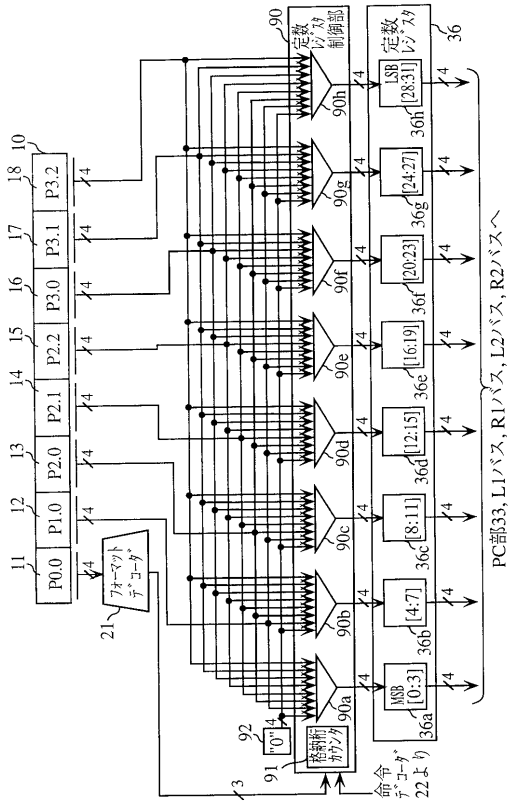
【図９】



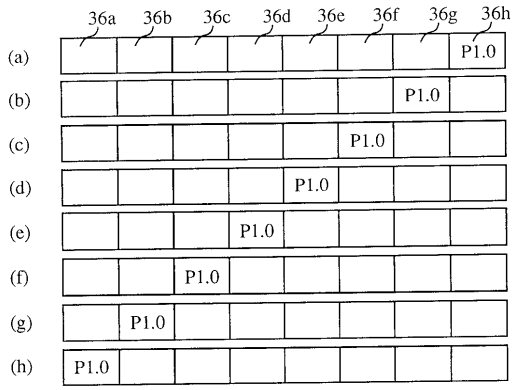
【図１０】



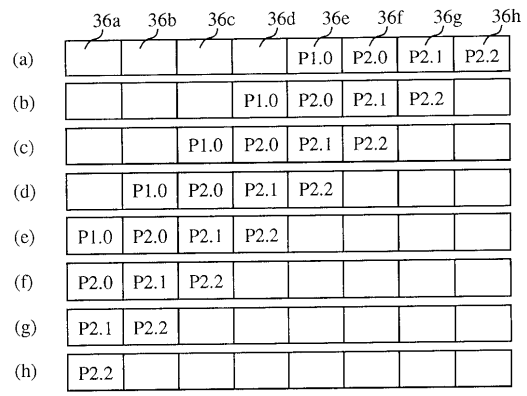
【図１１】



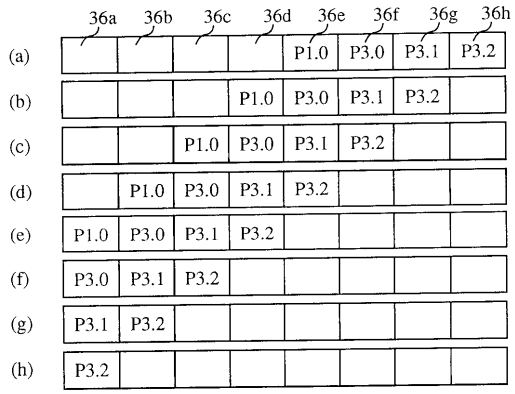
【図１２】



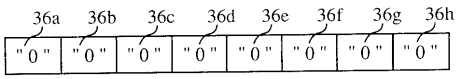
【図１３】



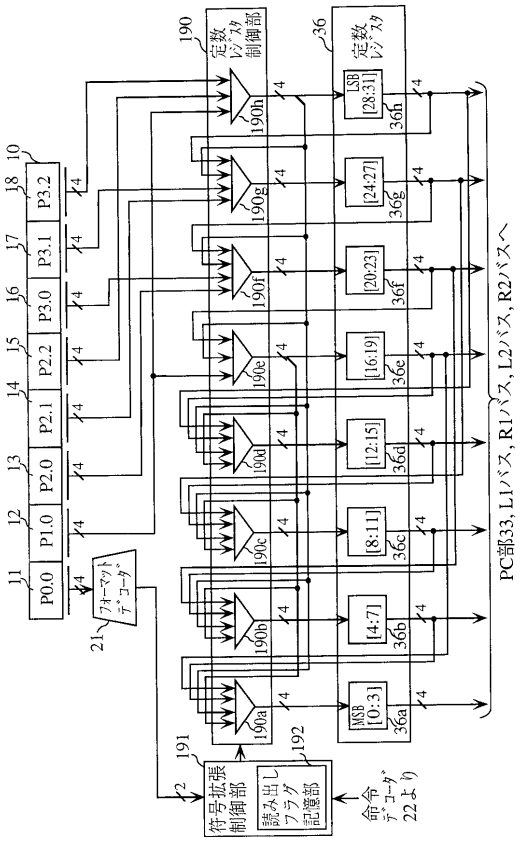
【図１４】



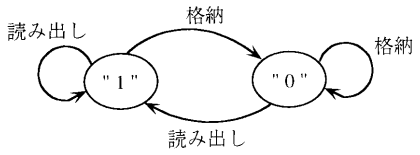
【図１５】



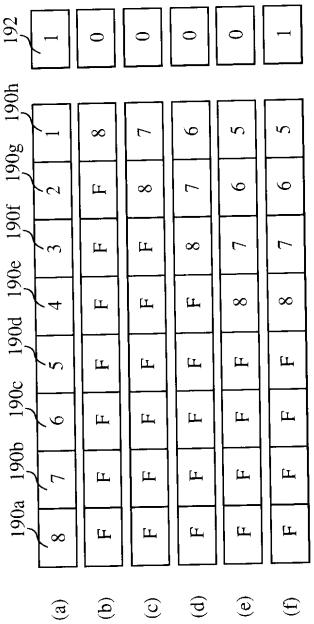
【図１６】



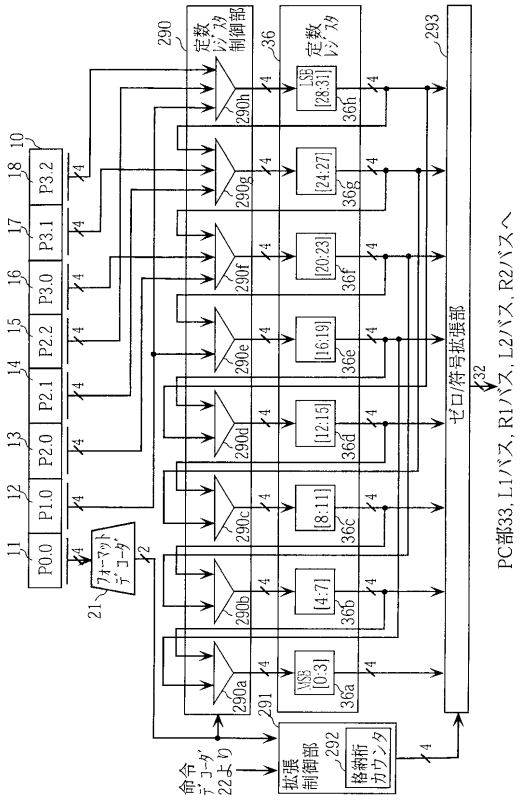
【図１７】



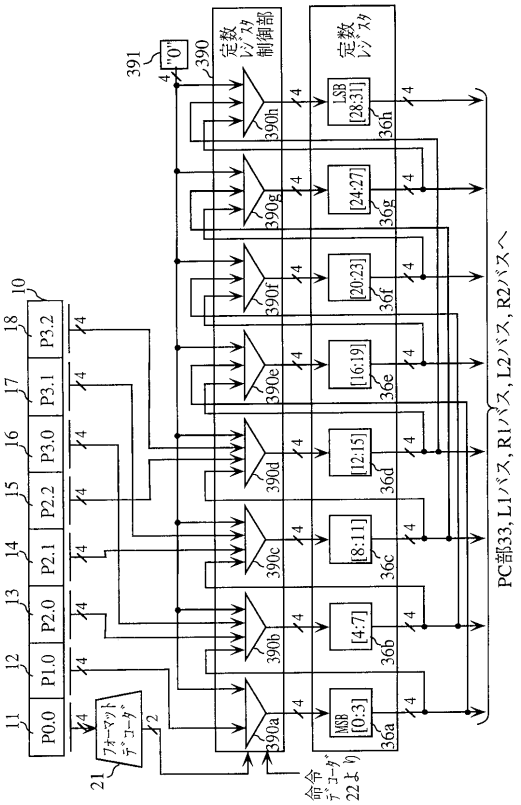
【図１８】



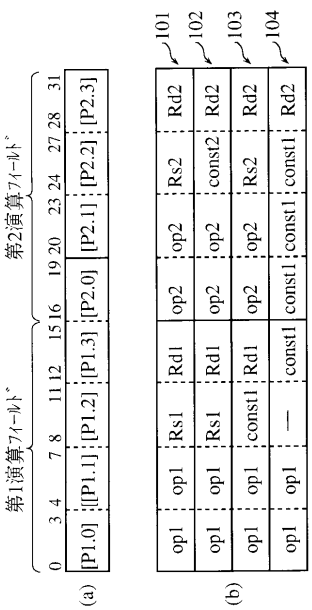
【図１９】



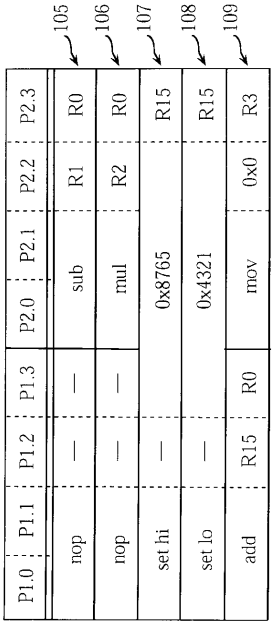
【図２０】



【図２１】



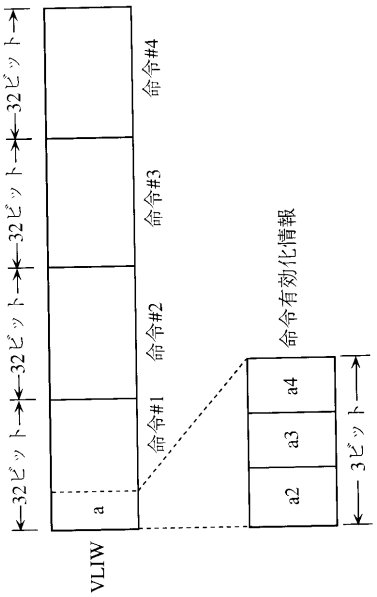
【図２２】



【図２３】

	P1.0	P1.1	P1.2	P1.3	P2.0	P2.1	P2.2	P2.3
110		mov	R6	R1		mov	R7	R2
111		add	R0	R1		add	R0	R2
112		mul	R6	R1		sub	R7	R2
113		mov	R8	R4		mov	R9	R5
114		add	—	—	0x8765			R0
115		nop	—	—		mov	0x0	R3

【図２４】



フロントページの続き

(56)参考文献 特開平５－１９７５４５（ＪＰ，Ａ）
特開平５－１４３３３３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 9/38