(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4254954号
(P4254954)

(45)発行日　平成21年4月15日(2009.4.15)　(24)登録日　平成21年2月6日(2009.2.6)

(51)Int.Cl.　　F I
G06F 9/40 (2006.01)　　G06F 9/40 310C
G06F 9/38 (2006.01)　　G06F 9/38 390

請求項の数 11　(全 37 頁)

(21)出願番号　特願2004-97197(P2004-97197)
(22)出願日　平成16年3月29日(2004.3.29)
(65)公開番号　特開2005-284683(P2005-284683A)
(43)公開日　平成17年10月13日(2005.10.13)
審査請求日　平成18年10月19日(2006.10.19)

(73)特許権者　503360115
独立行政法人科学技術振興機構
埼玉県川口市本町４丁目１番８号
(74)代理人　110000338
特許業務法人原謙三国際特許事務所
(74)代理人　100080034
弁理士　原　謙三
(72)発明者　中島　康彦
京都府京都市左京区吉田神楽岡町５－２８－１０２

審査官　久保　正典

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】
主記憶手段から命令区間を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、
上記主記憶手段から読み出した命令区間に基づく演算を行う第１の演算手段と、上記第１の演算手段による上記主記憶手段に対する読み出しおよび書き込み時に用いられるレジスタと、上記第１の演算手段によって命令区間の演算が行われたときの入力パターンおよび出力パターンからなる入出力グループを生成する入出力生成手段と、上記入出力生成手段によって生成された入出力グループを記憶する命令区間記憶手段とを備え、
上記第１の演算手段が、命令区間を実行する際に、該命令区間の入力パターンと、上記命令区間記憶手段に記憶されている入力パターンとが一致した場合、該入力パターンと対応して上記命令区間記憶手段に記憶されている出力パターンをレジスタおよび／または主記憶手段に出力する再利用処理を行い、
上記入出力生成手段が、
出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかを示す依存関係格納部と、
上記依存関係格納部に格納されている情報に基づいて、１以上の上記出力要素を含む出力パターンと、１以上の上記入力要素を含む入力パターンとからなる入出力グループを設定する入出力グループ設定手段とを備えていることを特徴とするデータ処理装置。
【請求項２】

上記入出力グループ設定手段が、ある第１の出力要素の起源となる入力要素の組が、他の第２の出力要素の起源となる入力要素の組に全て含まれている場合に、第２の出力要素の起源となる入力要素の組を入力パターン、第１の出力要素および第２の出力要素を出力パターンとする入出力グループを設定することを特徴とする請求項１記載のデータ処理装置。

【請求項３】

上記入出力グループ設定手段が、ある第１の出力要素の起源となる入力要素の組と、他の第２の出力要素の起源となる入力要素の組との間で、共通の入力要素が存在しない場合に、第１の出力要素の起源となる入力要素の組を入力パターン、第１の出力要素を出力パターンとする第１の入出力グループ、および、第２の出力要素の起源となる入力要素の組を入力パターン、第２の出力要素を出力パターンとする第２の入出力グループをそれぞれ設定することを特徴とする請求項１記載のデータ処理装置。

【請求項４】

上記依存関係格納部が、上記各出力要素を行成分、上記各入力要素を列成分とする２次元配列メモリによって構成され、該２次元配列メモリの各メモリ要素が、該メモリ要素の行成分に対応する出力要素が、該メモリ要素の列成分に対応する入力要素を起源とするか否かの情報を保持していることを特徴とする請求項１記載のデータ処理装置。

【請求項５】

上記第１の演算手段によって命令区間の演算が行われる際に、レジスタおよび／または主記憶手段から読み出しが行われた場合に、上記入出力生成手段が、

（１）読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として依存関係格納部に登録されている場合、該出力要素に対応する依存関係格納部の行成分からなる暫定行列を一時記憶する処理、

（２）読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素としては依存関係格納部に登録されておらず、入力要素として依存関係格納部に登録されている場合、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列を一時記憶する処理、および、

（３）読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素および入力要素のいずれとしても依存関係格納部に登録されていない場合には、該アドレスおよび値を入力要素として依存関係格納部に登録するとともに、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列を一時記憶する処理を行い、

レジスタおよび／または主記憶手段への書き込みが行われた場合に、上記入出力生成手段が、

（４）書き込みが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されている場合、登録されている出力要素に対応する出力値を、書き込みが行われた値に更新するとともに、

既に登録されている出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている全ての暫定行列の論理和に置き換え、その後、一時記憶されている暫定行列を初期化する処理、および、

（５）書き込みが行われたレレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されていない場合、該アドレスおよび値を出力要素として依存関係格納部に登録するとともに、該出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている全ての暫定行列の論理和に置き換え、その後、一時記憶されている暫定行列を初期化する処理を行うことを特徴とする請求項４記載のデータ処理装置。

【請求項６】

上記入出力グループ設定手段が、上記２次元配列メモリにおける各行成分間の論理積演算を行う行間論理積比較部を含んでおり、

上記入出力グループ設定手段が、依存関係格納部において、ある第１行成分の反転と、ある第２行成分との論理積が全て０になる行成分の組を抽出し、抽出された行成分の組の

うち、入力要素の組を最も多く含む行成分以外の行成分を、入出力グループの対象外として設定することを特徴とする請求項４記載のデータ処理装置。
【請求項７】
上記入出力グループ設定手段が、上記２次元配列メモリにおける各行成分間の論理積演算を行う行間論理積比較部を含んでおり、
上記入出力グループ設定手段が、依存関係格納部において、他のどの行成分に対しても論理積が全て０になる行成分を、それぞれ入出力グループとして設定することを特徴とする請求項４記載のデータ処理装置。
【請求項８】
上記命令区間記憶手段が、複数の上記入力パターンを、一致比較すべき項目をノードとみなした木構造として記憶する入力パターン記憶手段を備えていることを特徴とする請求項１記載のデータ処理装置。
【請求項９】
上記入力パターン記憶手段が、上記入力パターンにおいて一致比較すべき項目の値と、次に比較すべき項目とを対応させて格納することによって、上記木構造を実現することを特徴とする請求項８記載のデータ処理装置。
【請求項１０】
上記入力パターン記憶手段が、連想検索手段と、付加記憶手段とを備え、
上記連想検索手段が、一致比較すべき項目の値を格納する値格納領域と、該項目を識別するキーを格納するキー格納領域とを有する１つ以上の検索対象ラインを備え、
上記付加記憶手段が、上記検索対象ラインに対応した対応ラインごとに、次に連想検索を行うべき項目を格納する検索項目指定領域を有していることを特徴とする請求項９記載のデータ処理装置。
【請求項１１】
少なくとも１つの第２の演算手段をさらに備え、
上記第２の演算手段が、上記第１の演算手段によって処理が行われている命令区間に関して、今後入力が予想される予測入力値に基づいて該命令区間の演算を行い、その結果を上記命令区間記憶手段に対して登録することを特徴とする請求項１～１０のいずれか一項に記載のデータ処理装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、主記憶手段から命令区間を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置に関するものである。
【背景技術】
【０００２】
従来、ＣＰＵ(Central Processing Unit)を始めとするマイクロプロセッサにおいて、演算速度の高速化技術に関する研究開発が盛んに行われている。高速化技術としては、例えばパイプライン、スーパースケーラ、アウトオブオーダー実行、および、レジスタリネーミングなどが挙げられる。
【０００３】
パイプラインは、命令の実行処理を数段階に分解し、複数の命令を流れ作業的に同時処理を行う技術である。スーパースケーラは、命令の実行回路を２組以上用意し、複数の命令を同時に並行して実行する技術である。アウトオブオーダー実行は、命令の記述順序を無視して、いくつかの連続する命令の中から先に実行可能なものを探して先行処理を行う技術である。レジスタリネーミングは、例えばＣＩＳＣ(Complex Instruction Set Computer)タイプのプロセッサにおいて、従来のプロセッサにおける命令の互換性を保ちながら、汎用レジスタの数を増やすことによって並行処理が行われる確率を増大させる技術である。
【０００４】

このように、マイクロプロセッサにおける演算速度の高速化を図る際には、命令の実行を並行して行うことが重要となっている。しかしながら、プログラム中には、ある命令の結果に応じて異なる命令が行われるような依存関係、言い換えれば分岐が含まれている場合がほとんどである。このような分岐が含まれている場合、並行処理によって先行して処理を行っていると、分岐の結果によって先行処理した内容が無駄になるという状況が発生することになり、演算速度の高速化の効果が小さくなるという問題がある。

【０００５】

そこで、プログラム中に分岐がある場合に、分岐先を予測することによって先行処理が無駄になる確率を低減し、並行処理の効果を向上させる技術、いわゆる分岐予測に関する研究が数多く行われている。

【０００６】

しかしながら、分岐予測に基づいて投機的先行処理を行う場合には、一般的に次のような問題がある。第１の問題としては、予測の正当性を常に検証する必要があるので、先行命令列の実行時間そのものを削減することはできない、という点である。第２の問題としては、誤った予測に基づく一連の先行演算結果を全て無効化する必要があるので、一度に投機的先行処理できる命令数を多くするには、相応のハードウェアコストを要する、という点である。第３の問題としては、命令間の依存関係が多いほど、多重に投機的先行処理をする必要が生じ、予測の正当性の検証処理、および誤った予測に基づく処理の無効化処理が極めて複雑になる、という点である。

【０００７】

一方、分岐予測とは異なる高速化技術として、値再利用という技術も提案されている。この値再利用とは、プログラムの一部分に関する入力値および出力値を再利用表に登録しておき、同じ箇所を再度実行する際に、入力値が再利用表に登録されているものである場合には、登録されている出力値を出力する、という技術である。この値再利用による効果としては次のようなものが挙げられる。（１）入力値が、再利用表に登録されている入力値と一致すれば、実行結果を検証する必要がない。（２）入力値および出力値の総数によってのみハードウェアコストが決定され、省略可能な命令列の長さが制約されない。（３）命令間の依存関係の多少は、再利用機構の複雑さに影響を与えない。（４）冗長なロード／ストア命令を削減することができるとともに、これに伴う消費電力の削減も実現される。

【０００８】

後記する非特許文献１には、プログラムにおける関数に関して値再利用を行う技術が示されている。この従来技術では、一般的にロードモジュールがＡＢＩ(Application Binary Interface)に従って作られることを利用しており、特に、ＳＰＡＲＣ(Scalable Processor ARChitecture)　ＡＢＩを利用している。そして、このＡＢＩにおいて関数の入出力を特定することによって値再利用を実現している。すなわち、値再利用のためのコンパイラによる専用命令の埋め込みが不要となっており、既存ロードモジュールへの適用が可能となっている。

【０００９】

また、関数の多重構造を動的に把握することにより、関数内局所レジスタやスタック上の局所変数を値再利用における入出力値から除外するようにしており、これによって効率を向上させている。特に関数については、関数の複雑さに拘わらず、最大６のレジスタ入力、最大４のレジスタ出力、および、局所変数を含まない最小限の主記憶値の登録による再利用および事前実行が可能となっている。この従来技術について以下に詳細に説明する。

【００１０】

まず、単一の関数を対象として、何が入力で何が出力であるかを明らかにし、１レベルの再利用を行うために必要な機構について説明する。プログラムにおいては、一般的に関数は多重構造を形成している。関数Ａ（Function-A）が関数Ｂ（Function-B）を呼び出す構造を図１５（ａ）に示す。

【００１１】
大域変数（Globals）は、関数Ａの入出力（Ａｉｎ／Ａｏｕｔ）および関数Ｂの入出力（Ｂｉｎ／Ｂｏｕｔ）になりうるものである。関数Ａの局所変数（Locals-A）は、関数Ａの入出力ではないが、ポインタを通じてＢの入出力になりうるものである。また、関数Ａから関数Ｂへの引数（Args）は、関数Ｂへの入力となりうるものであり、関数Ｂから関数Ａの返り値（Ret.Val.）は、関数Ｂからの出力となりうるものである。なお、関数Ｂの局所変数（Locals-B）は、関数Ａおよび関数Ｂの入出力には含まれない。
【００１２】
コンテクストに依存せずに関数Ｂを再利用するには、関数Ｂの実行時に、関数Ｂの入出力Ｂｉｎ／Ｂｏｕｔのみを入出力として登録しなければならない。ここで、図１５（ａ）に示すプログラム構造を実行する際の主記憶におけるメモリマップを図１５（ｂ）に示す。このメモリマップにおいて、Ｂｉｎ／Ｂｏｕｔを含まない領域はLocals-Bのみとなっている。よって、Ｂｉｎ／Ｂｏｕｔを識別するには、GlobalsとLocals-Bとの境界、および、Locals-BとLocals-Aとの境界をそれぞれ確定しなければならない。前者については、一般的にＯＳ(Operating System)が実行時のデータサイズおよびスタックサイズの上限を決めることを利用し、ＯＳが設定する境界(LIMIT)に基づいてGlobalsとLocals-Bとの境界を確定することができる。後者については、Ｂが呼び出される直前のスタックポインタの値（SP in A）を用いることによって、Locals-BとLocals-Aとの境界を確定することができる。
【００１３】
次に、与えられた主記憶アドレスが、大域変数であるか、または、どの関数の局所変数であるかを識別する方法について説明する。ロードモジュールは、ＳＰＡＲＣ　ＡＢＩに規定されている以下の条件を満たすと仮定する。なお、%fpはフレームポインタ、%spはスタックポインタを意味するものとする。
（１）%sp以上の領域のうち、%sp＋０～６３はレジスタ退避領域、%sp＋６８～９１は引数退避領域であり、いずれも関数の入出力ではない。
（２）構造体を返す場合の暗黙的引数(Implicit Arg.)は%sp＋６４～６７に格納される。
（３）明示的引数(Explicit Arg.)はレジスタ%o０～５、%sp＋９２以上の領域に置かれる。
【００１４】
まず、大域変数と局所変数とを区別するために、一般的に、ＯＳが実行時のデータサイズおよびスタックサイズの上限を決めることを利用し、次の事項を仮定する。
（１）大域変数はLIMIT未満の領域に置かれる。
（２）%spは、LIMIT以下になることはなく、LIMIT～%spの領域は無効である。
【００１５】
以上の条件を満たしながら、関数Ａが関数Ｂを呼び出す場合の、メモリマップにおける引数およびフレームの概要を図１６に示す。同図を参照しながら、以下にＡの局所変数およびＢの局所変数を区別する方法について説明する。
【００１６】
同図において、（ａ）はＡ実行中の状態を示している。LIMIT未満の太枠部分に命令(Instructions)および大域変数(Global Vars.)が格納され、%sp以上に有効な値が格納されている。%sp＋６４には、Ｂが構造体を返り値とする場合の暗黙的引数として、構造体の先頭アドレスが格納される。Ｂに対する明示的引数の先頭６ワードはレジスタ%o０～５、第７ワード以降は%sp＋９２以上に格納される。ベースレジスタを%spとするオペランド%sp＋９２が出現した場合、この領域は引数の第７ワードすなわちＢの局所変数である。一方、オペランド%sp＋９２が出現しない場合、この領域はＡの局所変数である。このように、（ａ）の状態では、オペランドを検証することによってＡの局所変数とＢの局所変数とを区別することができる。
【００１７】

一方、（ｂ）はＢ実行中の状態を示している。引数が入力、返り値が出力、大域変数およびＡの局所変数が入出力となりうる。ただし、Ｂは可変長引数を受け入れる場合があるので、一般に％fp＋９２以上の領域がＡの局所変数の領域となるかＢの局所変数の領域となるかは判断できない。

【００１８】

局所変数を区別するには、まず、（ａ）の時点において引数の第７ワード以降を検出した関数呼び出しは再利用の対象外とし、第７ワード以降を検出しない関数呼び出しに関して、直前に％sp＋９２の値を記録しておくようにする。なお、第７ワード以降を使用する関数呼び出しの出現頻度が低いと予想されることから、第７ワード以降を使用する関数を再利用の対象外とする制限による性能低下は軽微なものと考える。

【００１９】

以上の準備により、（ｂ）における主記憶参照アドレスが、予め記録した％sp＋９２の値以上の場合はＡの局所変数、小さい場合はＢの局所変数であることがわかる。Ｂ実行時には、Ｂの局所変数を除外しながら、大域変数およびＡの局所変数を再利用表へ登録する。

【００２０】

再利用の際は、Ｂの局所変数は入出力から除外されるので、Ｂの局所変数のアドレスが一致している必要がない。このため、いかなるコンテクストであっても、入力さえ一致すれば、再利用することが可能である。ただし、Ｂが参照する大域変数やＡの局所変数については、アドレスおよびデータの両方が再利用表の内容と完全に一致する必要がある。すなわち、Ｂを実行する前に、どのようにして比較すべき主記憶アドレスを網羅するかがポイントになる。

【００２１】

Ｂが参照する大域変数やＡの局所変数のアドレスは、そもそもＢにおいて生成されるアドレス定数や、大域変数／引数を起源とするポインタに基づいているものである。よって、まず引数が完全に一致する再利用表中のエントリを選択した後に、関連する主記憶アドレスをすべて参照して一致比較を行うことにより、Ｂが参照すべき主記憶アドレスを網羅することができる。そして、全ての入力が一致した場合にのみ、登録済の出力（返り値、大域変数、およびＡの局所変数）を再利用することができる。

【００２２】

関数再利用を実現するために、再利用表として、関数管理表（ＲＦ）および入出力記録表（ＲＢ）を設けることにする。１つの関数を再利用するために必要なハードウェア構成を図１７に示す。複数の関数を再利用可能とするには、この構成を複数組用意することになる。

【００２３】

この表において、ＲＦおよびＲＢに保持されるVは、エントリが有効であるか否かを示すフラグであり、LRU(least recently used)は、エントリ入れ替えのヒントを示している。ＲＦは、上記のVおよびLRUの他に、関数の先頭アドレス(Start)、および参照すべき主記憶アドレス(Read/Write)を保持する。ＲＢは、上記のVおよびLRUの他に、関数呼び出し直前の％sp(SP)、引数(Args.)（V：有効エントリ、Val.：値）、主記憶値(Mask：Read/Writeアドレスの有効バイト、Value：値)、および、返り値(Return Values)(V：有効エントリ、Val.：値)を保持する。

【００２４】

返り値は、％ｉ０～１（リーフ関数では％ｏ０～１に読み替える）または％ｆ０～１に格納され、％ｆ２～３を使用する返り値（拡張倍精度浮動小数点数）は対象プログラムには存在しないものと仮定する。ReadアドレスはＲＦが一括管理し、MaskおよびValueはＲＢが管理することにより、Readアドレスの内容とＲＢの複数エントリをＣＡＭ(content-addressable memory)により一度に比較する構成を可能としている。

【００２５】

単一の関数を再利用するには、まず、関数実行時に、局所変数を除外しながら、引数、

返り値、大域変数および上位関数の局所変数に関する入出力情報を再利用表に登録していく。ここで、読み出しが先行した引数レジスタは関数の入出力として、また、返り値レジスタへの書き込みは関数の出力として登録する。その他のレジスタ参照は登録する必要がない。主記憶参照も同様に、読み出しが先行したアドレスについては入力、書き込みは出力として登録する。

【００２６】

関数から復帰するまでに次の関数を呼び出した場合、または、登録すべき入出力が再利用表の容量を超える、引数の第７ワードを検出する、途中でシステムコールや割り込みが発生する、などの擾乱が発生しなかった場合、復帰命令を実行した時点で、登録中の入出力表エントリを有効にする。

【００２７】

以降、図１７を参照しながら説明すると、関数を呼び出す前に、（１）関数先頭アドレスを検索し、（２）引数が完全に一致するエントリを選択し、（３）関連する主記憶アドレスすなわち少なくとも１つのMaskが有効であるReadアドレスをすべて参照して、（４）一致比較を行う。全ての入力が一致した場合に、（５）登録済の出力（返り値、大域変数、およびＡの局所変数）を書き戻すことによって、関数の実行を省略することができる。

【非特許文献１】情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム，ＨＰＳ５，pp.1-12，Sep.(2002)，“関数値再利用および並列事前実行による高速化技術”（中島康彦、緒方勝也、正西申悟、五島正裕、森眞一郎、北村俊明、富田眞治）（発行日２００２年９月１５日）

【発明の開示】

【発明が解決しようとする課題】

【００２８】

上記の従来技術では、ＲＢにおいて、各エントリは、１つの項目でも内容が異なれば、それぞれ別のエントリとして登録する必要がある。よって、ＲＢにおけるメモリの利用効率は良くないことになる。また、実行しようとしている関数の入力パターンと、ＲＢの各エントリに含まれている入力パターンとで、１つでも異なるものがあると、再利用を行うことができないことになる。

【００２９】

本発明は上記の問題点を解決するためになされたもので、その目的は、再利用を行う上でより的確な入出力グループを命令区間記憶手段に登録することを可能とするデータ処理装置を提供することにある。

【課題を解決するための手段】

【００３０】

本発明に係るデータ処理装置は、上記課題を解決するために、主記憶手段から命令区間を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、上記主記憶手段から読み出した命令区間に基づく演算を行う第１の演算手段と、上記第１の演算手段による上記主記憶手段に対する読み出しおよび書き込み時に用いられるレジスタと、上記第１の演算手段によって命令区間の演算が行われたときの入力パターンおよび出力パターンからなる入出力グループを生成する入出力生成手段と、上記入出力生成手段によって生成された入出力グループを記憶する命令区間記憶手段とを備え、上記第１の演算手段が、命令区間を実行する際に、該命令区間の入力パターンと、上記命令区間記憶手段に記憶されている入力パターンとが一致した場合、該入力パターンと対応して上記命令区間記憶手段に記憶されている出力パターンをレジスタおよび／または主記憶手段に出力する再利用処理を行い、上記入出力生成手段が、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかを示す依存関係格納部と、上記依存関係格納部に格納されている情報に基づいて、１以上の上記出力要素を含む出力パターンと、１以上の上記入力要素を含む入力パターンとからなる入出力グループを設定する入出力グループ設定手段とを備えていることを特徴としている。

【００３１】

上記の構成では、第１の演算手段が命令区間を実行する際に、該命令区間の入力パターンと、上記命令区間記憶手段に記憶されている入力パターンとが一致した場合、該入力パターンと対応して上記命令区間記憶手段に記憶されている出力パターンをレジスタおよび／または主記憶手段に出力する再利用処理を行う構成となっている。そして、命令区間記憶手段に記憶される入力パターンおよび出力パターンは、入出力生成手段によって生成されたものとなっている。

【００３２】

入出力生成手段は、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかを示す情報に基づいて、１以上の出力要素を含む出力パターンと、１以上の入力要素を含む入力パターンとからなる入出力グループを設定し、設定された１以上の入出力グループを生成するようになっている。したがって、ある命令区間が実行された際の入力パターンおよび出力パターンを単純に命令区間記憶手段に登録する場合と比較して、再利用を行う上でより的確な入出力グループを命令区間記憶手段に登録することが可能となる。よって、再利用を行う際の検索効率を向上させることができる。

【００３３】

また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、ある第１の出力要素の起源となる入力要素の組が、他の第２の出力要素の起源となる入力要素の組に全て含まれている場合に、第２の出力要素の起源となる入力要素の組を入力パターン、第１の出力要素および第２の出力要素を出力パターンとする入出力グループを設定する構成としてもよい。

【００３４】

上記の構成では、ある第１の出力要素の起源となる入力要素の組が、他の第２の出力要素の起源となる入力要素の組に全て含まれている場合に、これらが１つの入出力グループにまとめられることになる。よって、冗長な入出力グループを削除することが可能となるので、命令区間記憶手段に入出力グループを冗長に登録することを防止することができる。

【００３５】

また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、ある第１の出力要素の起源となる入力要素の組と、他の第２の出力要素の起源となる入力要素の組との間で、共通の入力要素が存在しない場合に、第１の出力要素の起源となる入力要素の組を入力パターン、第１の出力要素を出力パターンとする第１の入出力グループ、および、第２の出力要素の起源となる入力要素の組を入力パターン、第２の出力要素を出力パターンとする第２の入出力グループをそれぞれ設定する構成としてもよい。

【００３６】

上記の構成によれば、２つの入出力グループにおいて、共通の入力要素が存在しない場合には、それぞれ別の入出力グループとして設定されることになる。ここで、共通の入力要素が存在しない場合とは、それぞれの入出力グループが互いに依存関係を有さないということになる。すなわち、再利用を行う際に、以前に実行された命令区間における入力パターンおよび出力パターンのうちの一部のみが一致した場合にも、再利用を行うことが可能となるので、再利用が可能となる確率を高めることができる。

【００３７】

また、本発明に係るデータ処理装置は、上記の構成において、上記依存関係格納部が、上記各出力要素を行成分、上記各入力要素を列成分とする２次元配列メモリによって構成され、該２次元配列メモリの各メモリ要素が、該メモリ要素の行成分に対応する出力要素が、該メモリ要素の列成分に対応する入力要素を起源とするか否かの情報を保持している構成としてもよい。

【００３８】

上記の構成では、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの

入力要素を起源とするものであるかの情報を、２次元配列メモリによって示すようになっている。よって、２次元配列メモリの各メモリ要素に対して、例えば１または０を格納するという単純な処理によって上記の情報を格納することができるとともに、例えば各メモリ要素に関して論理演算を行うことによって、各行成分の関係などを容易に把握することが可能となる。

【００３９】

また、本発明に係るデータ処理装置は、上記の構成において、上記第１の演算手段によって命令区間の演算が行われる際に、レジスタおよび／または主記憶手段から読み出しが行われた場合に、上記入出力生成手段が、（１）読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として依存関係格納部に登録されている場合、該出力要素に対応する依存関係格納部の行成分からなる暫定行列を一時記憶する処理、（２）読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素としては依存関係格納部に登録されておらず、入力要素として依存関係格納部に登録されている場合、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列を一時記憶する処理、および、（３）読み出しが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素および入力要素のいずれとしても依存関係格納部に登録されていない場合には、該アドレスおよび値を入力要素として依存関係格納部に登録するとともに、該入力要素に対応する依存関係格納部の列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列を一時記憶する処理を行い、レジスタおよび／または主記憶手段への書き込みが行われた場合に、上記入出力生成手段が、（４）書き込みが行われたレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されている場合、登録されている出力要素に対応する出力値を、書き込みが行われた値に更新するとともに、既に登録されている出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている全ての暫定行列の論理和に置き換え、その後、一時記憶されている暫定行列を初期化する処理、および、（５）書き込みが行われたレレジスタおよび／または主記憶手段のアドレスが、出力要素として登録されていない場合、該アドレスおよび値を出力要素として依存関係格納部に登録するとともに、該出力要素に対応する依存関係格納部の行成分を、その時点で一時記憶されている全ての暫定行列の論理和に置き換え、その後、一時記憶されている暫定行列を初期化する処理を行う構成としてもよい。

【００４０】

上記のような処理が行われることによって、ある命令区間が実行された際の入出力関係、すなわち、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかについての情報を的確に依存関係格納部の２次元配列メモリに格納することができる。

【００４１】

また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、上記２次元配列メモリにおける各行成分間の論理積演算を行う行間論理積比較部を含んでおり、上記入出力グループ設定手段が、依存関係格納部において、ある第１行成分の反転と、ある第２行成分との論理積が全て０になる行成分の組を抽出し、抽出された行成分の組のうち、入力要素の組を最も多く含む行成分以外の行成分を、入出力グループの対象外として設定する構成としてもよい。

【００４２】

上記の構成では、各行成分の論理積を行うことによって、入力要素の組を最も多く含む行成分以外の行成分を、入出力グループの対象外として設定するようになっている。この処理によって、ある第１の出力要素の起源となる入力要素の組が、他の第２の出力要素の起源となる入力要素の組に全て含まれている場合に、これらを１つの入出力グループにまとめることが実現されることになる。したがって、冗長な入出力グループを削除することが可能となるので、命令区間記憶手段に入出力グループを冗長に登録することを防止することができる。

【００４３】
また、本発明に係るデータ処理装置は、上記の構成において、上記入出力グループ設定手段が、上記２次元配列メモリにおける各行成分間の論理積演算を行う行間論理積比較部を含んでおり、上記入出力グループ設定手段が、依存関係格納部において、他のどの行成分に対しても論理積が全て０になる行成分を、それぞれ入出力グループとして設定する構成としてもよい。
【００４４】
上記の構成では、各行成分の論理積を行うことによって、他の行成分に対して独立関係にある行成分を入出力グループとして設定するようになっている。この処理によって、共通の入力要素が存在しない、言い換えれば、互いに依存関係を有さない入出力グループを抽出することができるので、再利用を行う際に、以前に実行された命令区間における入力パターンおよび出力パターンのうちの一部のみが一致した場合にも、再利用を行うことが可能となる。
【００４５】
また、本発明に係るデータ処理装置は、上記の構成において、上記命令区間記憶手段が、複数の上記入力パターンを、一致比較すべき項目をノードとみなした木構造として記憶する入力パターン記憶手段を備えている構成としてもよい。
【００４６】
上記の構成によれば、複数の入力パターンにおいて共通する項目については１つのノードとして記憶することが可能となるので、入力パターン記憶手段における記憶内容の冗長性を低減することが可能となる。したがって、命令区間記憶手段に必要とされる記憶容量を低減することができるので、データ処理装置自体のコストを低減することが可能となる。
【００４７】
そして、入力パターン記憶手段が、例えば連想検索装置によって構成されている場合、過去の入力パターンがグループ分割されて登録される可能性が高くなっているので、同時に複数の入力パターンの検索が行われる可能性を高めることが可能となる。すなわち、一般的な連想検索装置の特性である長レイテンシ高スループットのメリットをより効果的に享受することが可能となる。また、過去の入力パターンがグループ分割されて登録される可能性が高くなることによって、再利用時の入力パターンのヒット率を向上することができる。
【００４８】
また、本発明に係るデータ処理装置は、上記の構成において、上記入力パターン記憶手段が、上記入力パターンにおいて一致比較すべき項目の値と、次に比較すべき項目とを対応させて格納することによって、上記木構造を実現する構成としてもよい。
【００４９】
この場合、一致比較すべき項目に関して順に一致比較していくことが可能となるので、一致比較すべき項目をノードとみなした木構造として入力パターンを記憶することを実現することが可能となる。
【００５０】
また、本発明に係るデータ処理装置は、上記の構成において、上記入力パターン記憶手段が、連想検索手段と、付加記憶手段とを備え、上記連想検索手段が、一致比較すべき項目の値を格納する値格納領域と、該項目を識別するキーを格納するキー格納領域とを有する１つ以上の検索対象ラインを備え、上記付加記憶手段が、上記検索対象ラインに対応した対応ラインごとに、次に連想検索を行うべき項目を格納する検索項目指定領域を有している構成としてもよい。
【００５１】
この場合、一致比較すべき項目の値が連想検索手段に入力されると、値とキーとが一致する検索対象ラインがシングルマッチし、シングルマッチした検索対象ラインに対応する付加記憶手段における対応ラインによって、次に連想検索を行うべき項目が確定するよう

になる。

【００５２】

ここで、各入力パターンは、一致比較すべき項目をノードとみなした木構造として記憶しているので、連想検索手段において、ある項目に関して一致する検索対象は、上記のように１つとなる（シングルマッチ）。シングルマッチ機構のみを有する連想検索メモリは一般的に市販されている一方、マルチマッチを、シングルマッチと同一性能によって報告可能な連想検索メモリは一般的には市販されていない。すなわち、上記の構成によれば、市販の連想検索メモリを連想検索手段として利用することができるので、より短期間かつ低コストで、本発明に係るデータ処理装置を実現することが可能となる。

【００５３】

また、本発明に係るデータ処理装置は、以上のように、第２の演算手段が、上記第１の演算手段によって処理が行われている命令区間に関して、今後入力が予想される予測入力値に基づいて該命令区間の演算を行い、その結果を上記命令区間記憶手段に対して登録する構成となっていてもよい。この場合、第２の演算手段によって、その時点で第１の演算手段によって処理が行われている命令区間に関して、予測入力値に基づく演算が行われ、その結果が命令区間記憶手段に記憶されることになる。よって、次に、同じ命令区間が出現し、予測入力値と同じ入力が行われた場合には、命令区間記憶手段に記憶されている値を再利用することが可能となる。例えば、入力値が単調に変化するような命令区間の場合には、予測入力値が的中する可能性が高いので、上記の構成による効果は高くなる。

【発明の効果】

【００５４】

本発明に係るデータ処理装置は、以上のように、上記入出力生成手段が、出力パターンに含まれる各出力要素が、入力パターンに含まれるどの入力要素を起源とするものであるかを示す依存関係格納部と、上記依存関係格納部に格納されている情報に基づいて、１以上の上記出力要素を含む出力パターンと、１以上の上記入力要素を含む入力パターンとからなる入出力グループを設定する入出力グループ設定手段とを備えている構成である。これにより、ある命令区間が実行された際の入力パターンおよび出力パターンを単純に命令区間記憶手段に登録する場合と比較して、再利用を行う上でより的確な入出力グループを命令区間記憶手段に登録することが可能となる。よって、再利用を行う際の検索効率を向上させることができるという効果を奏する。

【発明を実施するための最良の形態】

【００５５】

本発明の実施の一形態について図１ないし図１４に基づいて説明すれば、以下のとおりである。

【００５６】

（データ処理装置の構成）

本実施形態に係るデータ処理装置の概略構成を図２示す。同図に示すように、該データ処理装置は、ＭＳＰ(Main Stream Processor)１Ａ、ＳＳＰ(Shadow Stream Processor)１Ｂ、再利用表としての命令区間記憶部（命令区間記憶手段）２、および主記憶（主記憶手段）３を備えた構成となっており、主記憶３に記憶されているプログラムデータなどを読み出して各種演算処理を行い、演算結果を主記憶３に書き込む処理を行うものである。なお、同図に示す構成では、ＳＳＰ１Ｂを１つ備えた構成となっているが、２つ以上備えた構成となっていてもよい。また、同図に示す構成では、ＳＳＰ１Ｂを備えた構成となっているが、ＳＳＰ１Ｂを備えていない構成としてもかまわない。ＳＳＰ１Ｂを備えた場合の作用・効果については、後述する。

【００５７】

命令区間記憶部２は、プログラムにおける関数やループなどの命令区間を再利用するためのデータを格納するメモリ手段である。この命令区間記憶部２の詳細については後述する。

【００５８】

主記憶３は、ＭＳＰ１ＡおよびＳＳＰ１Ｂの作業領域としてのメモリであり、例えばＲＡＭ(Random Access Memory)などによって構成されるものである。例えばハードディスクなどの外部記憶手段や、外部のＩ／Ｏ(input/output)装置などの外部装置からプログラムやデータなどが主記憶３に読み出され、ＭＳＰ１ＡおよびＳＳＰ１Ｂは、主記憶３に読み出されたデータに基づいて演算を行うことになる。また、ＭＳＰ１Ａによる演算結果が主記憶３に書き込まれ、この演算結果が上記外部装置に送出されることになる。

【００５９】

ＭＳＰ１Ａは、再利用記憶手段としてのＲＷ（入出力生成手段）４Ａ、演算器（第１の演算手段）５Ａ、レジスタ６Ａ、およびＣａｃｈｅ７Ａを備えた構成となっている。また、ＳＳＰ１Ｂは、同様に、再利用記憶手段としてのＲＷ（第２の演算手段）４Ｂ、演算器（第２の演算手段）５Ｂ、レジスタ６Ｂ、およびＣａｃｈｅ／Ｌｏｃａｌ７Ｂを備えた構成となっている。

【００６０】

ＲＷ４Ａ・４Ｂは、再利用ウィンドウであり、現在実行中かつ登録中であるＲＦ（付加記憶手段）およびＲＢ（連想検索手段）（後述する）の各ラインをリング構造のスタックとして保持するものである。このＲＷ４Ａ・４Ｂは、実際のハードウェア構造としては、命令区間記憶部２における特定のラインをアクティブにする制御線の集合によって構成される。また、詳細は後述するが、ＲＷ４Ａ・４Ｂは、実行された命令区間に関して入出力パターンを生成し、この生成された入出力グループを命令区間記憶部２に対して実行結果として登録する処理を行う。

【００６１】

演算器５Ａ・５Ｂは、レジスタ６Ａ・６Ｂに保持されているデータに基づいて演算処理を行うものであり、ＡＬＵ（arithmetic and logical unit）と呼ばれるものである。レジスタ６Ａ・６Ｂは、演算器５Ａ・５Ｂによって演算を行うためのデータを保持する記憶手段である。なお、本実施形態では、演算器５Ａ・５Ｂ、およびレジスタ６Ａ・６Ｂは、ＳＰＡＲＣアーキテクチャに準じたものとする。Ｃａｃｈｅ７Ａ・７Ｂは、主記憶３と、ＭＳＰ１ＡおよびＳＳＰ１Ｂとの間でのキャッシュメモリとして機能するものである。なお、ＳＳＰ１Ｂでは、Ｃａｃｈｅ７Ｂには、局所メモリとしてのＬｏｃａｌ７Ｂが含まれているものとする。

【００６２】

（命令区間記憶部の構成）

図１は、本実施形態における命令区間記憶部２によって実現される再利用表を示している。同図に示すように、命令区間記憶部２は、ＲＢ、ＲＦ、ＲＯ１（第２出力パターン記憶手段）、およびＲＯ２（第１出力パターン記憶手段）を備えた構成となっている。

【００６３】

ＲＢは、比較すべき値であるレジスタ値または主記憶入力値を格納するValue（値格納領域）、およびキー番号を格納するKey（キー格納領域）を備えており、ValueおよびKeyの組み合わせのラインを複数備えている。

【００６４】

ＲＦは、次に比較すべきレジスタ番号または主記憶アドレスがないことを示す終端フラグＥ、次に比較すべきレジスタ番号または主記憶アドレスの内容が更新されたことを示す比較要フラグ、次に比較すべき対象がレジスタか主記憶かを示すＲ／Ｍ、次に比較すべきレジスタ番号または主記憶アドレスを示すAdr.（検索項目指定領域）、直前に参照したライン番号を示すUP（親ノード格納領域）、次に比較すべきレジスタ番号または主記憶アドレスよりも優先して比較すべきレジスタ番号または主記憶アドレスを示すAlt.（比較要項目指定領域）、および、優先して比較する際に必要なキーを示すDN（比較要キー指定領域）を備えており、これらはＲＢにおける各ラインに対応して設けられている。

【００６５】

ＲＯ１およびＲＯ２は、ＲＢおよびＲＦによる検索結果により、再利用が可能であると判定された場合に、主記憶および／またはレジスタに出力する出力値を格納するものであ

る。ＲＯ１は、ＲＦの各ラインに１対１で対応して出力値および出力すべきアドレスを格納している。ＲＯ２は、ＲＯ１のみでは出力値を格納しきれない場合に、格納しきれない分の出力値および出力すべきアドレスを格納している。ＲＯ２からも出力値を読み出す必要がある場合には、ＲＯ１における該当ラインに、ＲＯ２における出力値が格納されているポインタが示されており、このポインタを用いてＲＯ２から出力値の読み出しが行われる。

【００６６】

また、ＲＢおよびＲＦは、それぞれＣＡＭ(content-addressable memory)およびＲＡＭ(Random Access Memory)によって構成されている。一般的に、アドレスが与えられると、そのアドレスに格納された値を参照することができるメモリは、ＲＡＭと呼ばれるメモリである。一方、上記のＣＡＭとは、連想メモリと呼ばれるメモリであり、検索すべき内容が与えられると、その内容に一致するラインが選択されるようになっている。通常は、ＣＡＭはＲＡＭとセットにして用いられる。

【００６７】

ここで、ＣＡＭとＲＡＭとの連携動作について、具体例を挙げて説明する。ＣＡＭに、「５，５，５，５，５」、「１，３，１，１，１」、「１，３，３，５，２」、「６，６，６，６，６」というデータ列がエントリとして登録されており、ＲＡＭに、ＣＡＭにおける各データ列に対応して、「５，５」、「１，１」、「１，２」、「６，６」というデータが登録されているとする。ここで、検索すべきデータ列として、「１，３，３，５，２」をＣＡＭに入力すると、一致するエントリがＯＮとなり、ＲＡＭに登録されている該当するデータ「１，２」が出力されることになる。この具体例と同様の構成および動作によって、上記ＲＢおよびＲＦが実現されることになる。

【００６８】

（比較例）

ここで、比較例として、図８に示すような構成のＲＦおよびＲＢによる動作について説明する。同図に示すように、ＲＦは、エントリが有効であるか否かを示す状態表示フラグＶ、エントリ入れ替えのヒントを示すLRU、関数とループとを区別するF/L、命令区間の先頭アドレスを示すStart、命令区間の終了アドレスを示すEnd、参照すべき主記憶入力アドレスに関する情報を示すRead、および、参照すべき主記憶出力アドレスに関する情報を示すWriteを保持している。

【００６９】

また、ＲＢは、エントリが有効であるか否かを示す状態表示フラグＶ、エントリ入れ替えのヒントを示すLRU、命令区間を呼び出す際の直前のスタックポイント％spを示すSP、ループの終了アドレス(End)、ループ終了時の分岐方向を示すtaken/not、レジスタ入力値としての引数(Args.)（Ｖ：有効エントリ、Val.：値）および引数以外のレジスタ入力値および条件コード(Regs.,CC)、主記憶入力有効バイトMask、主記憶入力値Value、主記憶出力有効バイトMask、主記憶出力値Value、および、レジスタ出力値としての返り値Return Valuesおよび返り値以外のレジスタ出力値および条件コードRegs.,CC(Ｖ：有効エントリ、Val.：値)を保持している。

【００７０】

関数またはループを実行する際に、以前に実行した命令区間が再利用可能であるか否かを判断する際には、次の手順で行われる。まず、（１）ＲＦに登録されている関数またはループのエントリの先頭アドレスStartに、該当関数またはループの先頭アドレスと一致するものがあるかを検索する。一致するものがある場合には、（２）ＲＢに登録されている該当エントリのうち、有効エントリを示す状態表示フラグＶが登録済状態にセットされているエントリであって、かつ、該エントリにおける引数args.およびRegs.,CCが、呼び出す関数またはループの対応する値と完全に一致するエントリを１つまたは複数選択する。そして、選択したエントリにおいて、（３）関連する主記憶アドレス、すなわち、少なくとも１つのMaskが有効であるReadアドレスを用いて主記憶を順に参照し、（４）該当関数またはループの主記憶入力値と、ＲＢに登録されている主記憶入力値との比較を行う。

そして、全ての入力が一致する場合に、（５）ＲＢに記憶されているReturn Valuesをレジスタに書き込み、主記憶出力アドレスに対して、順次、各有効フラグMaskがセットされている主記憶出力値Valueを書き込む。以上により、関数またはループの再利用が実現されることになる。

【００７１】

以上のような比較例における動作を、図９を参照しながらより具体的に説明する。まず、プログラムカウンタ（ＰＣ）と、ＲＦに登録された命令区間先頭アドレス（Region）とが比較され、さらに、レジスタの内容（Reg.）と、ＲＢに登録されているレジスタ入力値（Args.、Regs.,CC）とを比較する。この時点で、ＲＢにおけるエントリ０１～０４のうち、エントリ０３およびエントリ０４が一致すると判定されたとする。すなわち、この時点では、マルチマッチとなっている。

【００７２】

次に、主記憶アドレスＡ１に関して比較することになるが、主記憶アドレスＡ１に対しては、ＲＦにおいて、一致比較を行う必要がないことを示すフラグ（０）が示されているので、一致比較は行われない。すなわち、エントリ０３およびエントリ０４が候補として残ったままとなる。

【００７３】

次に、主記憶アドレスＡ２に関して比較が行われる。ここで、ＲＦにおいて、主記憶アドレスＡ２に関しては一致比較を行う必要があることを示すフラグ（１）が示されているので、一致比較が行われる。この結果、内容が「００」であるエントリ０３のみが候補として残ることになる。その後、一致比較を行う項目として主記憶アドレスＡ３およびＡ４があるが、これらはどちらも一致比較を行う必要がないことを示すフラグが示されているので、エントリ０３は、比較が必要な全ての項目が一致したことになる。よって、エントリ０３に対応する出力値としての主記憶出力値およびレジスタ出力値が主記憶およびレジスタに出力される。

【００７４】

この比較例における動作のポイントは次の通りである。（ａ）ＲＢに登録されている各値と再利用対象となっている関数またはループにおける対応する値とを比較する際に、ＲＢにおける縦の列を順に一致確認していくことになるが、内容が一致するエントリが複数存在する（マルチマッチ）ことを許容している。（ｂ）検索途中においてマルチマッチを許容しているが、最終的に１つのエントリが選択されればよい。（ｃ）ＲＢにおける列を一致確認していく順番は任意であるので、例えばレジスタ入力値を最初にまとめて比較する、ということを行うことが可能である。

【００７５】

また、この比較例の場合、次のような問題がある。（ｄ）ＲＢにおいて、各エントリにおける項目数（横の長さ）は固定となっている。よって、登録されている項目以外の項目を追加することはできないようになっている。また、逆に、使用しない項目に対応するメモリ領域は空き領域となるが、これを有効利用することはできない。（ｅ）各エントリは、１つの項目でも内容が異なれば、それぞれ別のエントリとして登録する必要がある。よって、ＲＢにおけるメモリの利用効率は良くないことになる。

【００７６】

なお、以上のような比較例の場合、ＲＦおよびＲＢを構成するメモリとしては、構造が横長のものとなる。例えばこのメモリ容量を２Ｍｂｙｔｅとした場合、横が２Ｋｗｏｒｄ、縦を２５６エントリとすることになる。

【００７７】

（入力パターンを木構造として登録する第１構成例）

上記の比較例では、ＲＢにおける各エントリとしての横の行は、一致比較を行うべき入力値の項目を全て含んだものとなっている。すなわち、全ての入力パターンをそれぞれ１つのエントリとしてＲＢに登録するようになっている。

【００７８】

これに対して、本第１構成例では、一致比較を行うべき入力値の項目を短い単位に区切り、それぞれの比較単位をノードとしてとらえ、入力パターンを木構造としてＲＦおよびＲＢに登録するようになっている。そして、再利用を行う際には、一致するノードを順次選択することによって、最終的に再利用可能かを判断するようになっている。別の言い方をすれば、複数の入力パターンに共通する部分を１つにまとめて、ＲＦおよびＲＢの１行に対応づけるようになっている。

【００７９】

これにより、冗長性をなくし、命令区間記憶部２を構成するメモリの利用効率を向上させることが可能となる。また、入力パターンを木構造としているので、１つの入力パターンをＲＢにおける１つの行としてのエントリに対応付ける必要がないことになる。よって、一致比較を行うべき入力値の項目の数を可変にすることが可能となっている。

【００８０】

また、ＲＦおよびＲＢは、入力パターンを木構造として登録しているので、一致比較を行う際には、マルチマッチが行われないことになる。つまり、命令区間記憶部２としては、シングルマッチ機構を有する連想検索メモリであれば実現可能となる。ここで、シングルマッチ機構のみを有する連想検索メモリは一般的に市販されている一方、マルチマッチをシングルマッチと同一性能によって報告可能な連想検索メモリは一般的には市販されていない。すなわち、本第１構成例における命令区間記憶部２によれば、市販の連想検索メモリを利用することができるので、より短期間かつ低コストで、本実施形態に係るデータ処理装置を実現することが可能となる。

【００８１】

次に、図３を参照しながら、上記第１構成例における命令区間記憶部２における連想検索動作の具体例について説明する。まず、命令区間の実行が検出されると、プログラムカウンタ（ＰＣ）およびレジスタの内容（Reg.）がＲＢに入力される。そして、ＲＢにおいて、連想検索により、入力されたこれらの値と、ＲＢのValueの列に登録されている命令区間先頭アドレスおよびレジスタ値とが比較され、値が一致する唯一の行（ライン）が候補（マッチライン）として選択される。この例では、ＲＢにおける「０１」のラインがマッチラインとして選択される。

【００８２】

次に、マッチラインとして選択されたラインのＲＢにおける番地である「０１」が、エンコード結果としてＲＦに伝達され、キー０１に対応するＲＦにおけるラインが参照される。キー０１に対応するＲＦにおけるラインでは、比較要フラグが「０」であり、比較すべき主記憶アドレスがＡ１となっている。すなわち、主記憶アドレスＡ１に関しては、一致比較を行う必要はないことになる。

【００８３】

次に、キー０１を用いて、ＲＢにおけるKeyの列に対して検索が行われる。この例では、ＲＢにおける「０３」のラインがマッチラインとして選択される。そして、エンコード結果としてキー０３がＲＦに伝達され、キー０３に対応するＲＦにおけるラインが参照される。キー０３に対応するＲＦにおけるラインでは、比較要フラグが「１」であり、比較すべき主記憶アドレスがＡ２となっている。すなわち、主記憶アドレスＡ２に関しては、一致比較を行う必要があることになる。ここで、主記憶３における主記憶アドレスＡ２の値がＣａｃｈｅ７Ａを介して読み出され、ＲＢにおいて、Valueが主記憶３から読み出された値であり、かつ、Keyが「０３」となっているラインが検索される。図３に示す例では、Keyが「０３」となっているラインは「０４」および「０５」の２つあるが、主記憶３から読み出された値が「００」であるので、「０５」のラインがマッチラインとして選択され、ＲＦに対して、エンコード結果としてキー０５が伝達される。

【００８４】

以上のような処理が繰り返され、ＲＦにおいて、次に比較すべきレジスタ番号または主記憶アドレスがないことを示す終端フラグＥが検出された場合、入力パターンが全て一致したと判定され、該当命令区間は再利用可能と判断される。そして、終端フラグＥが検出

されたラインから「Select Output」信号が出力され、ＲＯ１およびＲＯ２に格納されている、該ラインに対応する出力値がレジスタ６Ａおよび主記憶３に対して出力される。

【００８５】

以上のように、本第１構成例における命令区間記憶部２による連想検索動作は、次のような特徴を有している。まず、内容が一致したことを示すマッチラインは、ＲＢにおいて１つのラインのみとなるので、検索動作を次列へ伝搬する際にエンコードした結果を１つ伝送すればよいことになる。したがって、ＲＢとＲＦとの間を接続する信号線は、アドレスのエンコード結果である１組（Ｎ本）でよいことになる。これに対して、上記した比較例では、ＲＢにおいてマルチマッチが許容されているので、ＲＢにおける各列同士を接続する信号線は、各ラインごとに設ける（2^N本）必要があることになる。すなわち、本第１構成例の構成によれば、命令区間記憶部２を構成する連想検索メモリにおける信号線の数を大幅に低減することが可能となる。

【００８６】

また、検索途中ではシングルマッチのみが許容されるようになっているので、比較すべき項目の比較順番は、木構造における参照順に限定されることになる。すなわち、レジスタ値とメモリ内容とは、参照順に混在させながら比較する必要がある。

【００８７】

入力パターンは、各項目を参照すべきKeyという形でリンクさせることにより、木構造によってＲＢおよびＲＦに登録されている。また、入力パターンの項目は、終端フラグによってその終端が示されるようになっている。よって、入力パターンの項目数を可変とすることができるので、再利用表に登録すべき命令区間の状態に応じて、柔軟に入力パターンの項目数を設定することが可能となる。また、入力パターンの項目数が固定でないことによって、利用しない項目が無駄にメモリ領域を占有することがなくなるので、メモリ領域の利用効率を向上させることができる。

【００８８】

また、木構造によって入力パターンが登録されるので、項目の内容が重複する部分については、複数の入力パターンで１つのラインを共有することが可能となっている。よって、メモリ領域の利用効率をさらに向上させることができる。

【００８９】

なお、以上のような構成の場合、ＲＦおよびＲＢを構成するメモリとしては、構造が縦長のものとなる。例えばこのメモリ容量を２Ｍｂｙｔｅとした場合、横が８ｗｏｒｄ、縦を６５５３６ラインとすることになる。

【００９０】

（入力パターンを木構造として登録する第２構成例）

上記の例では、図１に示したＲＦにおいて、UP、Alt.、およびDNの項目は利用していないことになる。すなわち、上記の例では、ＲＦにおいて、これらの項目を設ける必要はないことになる。これに対して、UP、Alt.、およびDNの項目を利用することによって、連想検索動作をさらに高速化する第２の構成例およびその動作について以下に説明する。

【００９１】

まず、図４（ｂ）に、プログラムカウンタ（ＰＣ）およびレジスタの内容（Reg.）のみを比較し、これらが一致した場合は、主記憶値を比較することなく、区間の再利用が可能であると判断できる場合の状態を示す。この状態では、まず、ＲＢの「０１」のラインにおいて、ＰＣおよびReg.がValueに登録されており、ＲＦの「０１」のラインにおいて、終端フラグが「Ｅ」、比較要フラグが「０」、比較すべき主記憶アドレスが「Ａ１」、親ノード番号を示すUPが「ＦＦ」となっている。また、ＲＢの「０３」のラインでは、Value値なしで、Keyが「０１」となっており、ＲＦの「０３」のラインでは、終端フラグが「Ｅ」、比較要フラグが「０」、比較すべき主記憶アドレスが「Ａ２」、親ノード番号を示すUPが「ＦＦ」となっている。以降、同様に、ＲＢおよびＲＦにおける「０５」のラインおよび「０７」のラインが登録されており、それぞれ終端フラグが「Ｅ」、比較要フラグが「０」となっている。

【００９２】
この状態で、ある命令区間の実行が検出されると、ＰＣおよびReg.がＲＢに入力され、マッチラインとして、ＲＢにおける「０１」のラインが選択される。そして、マッチラインとして選択されたラインのＲＢにおける番地である「０１」が、エンコード結果としてＲＦに伝達され、キー０１に対応するＲＦにおけるラインが参照される。キー０１に対応するＲＦにおけるラインでは、終端フラグが「Ｅ」となっているので、次に比較すべき主記憶アドレスがないことがわかる。また、比較要フラグ「０」となっているので、主記憶アドレスＡ１について比較を行う必要はないことがわかる。

【００９３】
したがって、図４（ａ）の木構造に示すように、ＰＣおよびReg.の一致がＳ１において確認されると、Ｔｒ１に示すノードのように、主記憶アドレスＡ１、Ａ２、Ａ３における比較を行うことなく、対応する出力値が出力されることになる。

【００９４】
ＲＦおよびＲＢがこの状態である場合に、主記憶アドレスＡ２に対して書き込みが行われたとする。この場合、ＲＦおよびＲＢにおける入力パターンの登録時には主記憶アドレスＡ２の一致比較を行う必要はない状態であったが、主記憶アドレスＡ２が変更されることによって、主記憶アドレスＡ２の一致比較を行う必要が生じることになる。したがって、この場合には、図５（ｂ）に示すようにＲＦおよびＲＢが変更されることになる。

【００９５】
まず、内容が変更された主記憶アドレスであるＡ２をキーにして、ＲＦにおけるAdr.の列に対して検索がかけられる。これによって、ＲＦにおける「０３」のラインが選択される。そして、選択された「０３」のラインにおいて、比較要フラグが「１」に設定されるとともに、終端フラグ「Ｅ」が削除される。

【００９６】
次に、「０３」のラインにおけるUPを参照することによって、親ノードとしての「０１」のラインが認識される。そして、「０１」のラインにおいて、次に比較すべき主記憶アドレスよりも優先して比較すべき主記憶アドレスを示すAlt.に、内容が変更された主記憶アドレスであるＡ２を書き込まれるとともに、終端フラグ「Ｅ」が削除される。さらに、「０１」のラインにおいて、優先して比較する際に必要なキーを示すDNに「０３」が書き込まれる。

【００９７】
以上のようにＲＦおよびＲＢが書き換えられた場合の連想検索動作は次のようになる。ある命令区間が検出された際に、まず、ＰＣおよびReg.がＲＢに入力される。そして、ＲＢにおいて、連想検索により、入力されたこれらの値と、ＲＢのValueの列に登録されている命令区間先頭アドレスおよびレジスタ値とが比較され、ＲＢにおける「０１」のラインがマッチラインとして選択される。

【００９８】
次に、マッチラインとして選択されたラインのＲＢにおける番地である「０１」が、エンコード結果としてＲＦに伝達され、キー０１に対応するＲＦにおけるラインが参照される。キー０１に対応するＲＦにおけるラインでは、比較要フラグが「０」であり、比較すべき主記憶アドレスがＡ１となっている。すなわち、主記憶アドレスＡ１に関しては、一致比較を行う必要はないことがわかる。

【００９９】
また、次に比較すべき主記憶アドレスよりも優先して比較すべき主記憶アドレスを示すAlt.に、主記憶アドレスＡ２が登録されており、優先して比較する際に必要なキーを示すDNに「０３」が登録されていることが確認される。この場合、主記憶３における主記憶アドレスＡ２の値がＣａｃｈｅ７Ａを介して読み出され、ＲＢにおいて、Valueが主記憶３から読み出された値であり、かつ、Keyが、DNに示されている「０３」となっているラインが検索される。

【０１００】

図５（ｂ）に示す例では、Keyが「０３」となっているラインは「０４」および「０５」の２つあるが、主記憶３から読み出された値が「００」であるので、「０５」のラインがマッチラインとして選択され、ＲＦに対して、エンコード結果としてキー０５が伝達される。キー０５に対応するＲＦにおけるラインでは、終端フラグが「Ｅ」となっているので、入力パターンが全て一致したと判定され、該当命令区間は再利用可能と判断される。そして、終端フラグＥが検出されたラインから「Select Output」信号が出力され、ＲＯ１およびＲＯ２に格納されている、該ラインに対応する出力値がレジスタ６Ａおよび主記憶３に対して出力される。

【０１０１】

以上のような連想検索動作を行う第２の構成例によれば、ＲＦにおいて、次に比較すべき主記憶アドレスよりも優先して比較すべき主記憶アドレスを示すAlt.、および、優先して比較する際に必要なキーを示すDNが設けられているので、主記憶アドレスＡ１の内容とキー０１による検索をスキップして、主記憶アドレスＡ２の内容とキー０３による検索が可能となる。したがって、検索動作の処理ステップを低減することができるので、処理の高速化を図ることができる。

【０１０２】

（出力値の格納手段構成例）

上記では、命令区間の入力パターンをＲＦおよびＲＢに登録し、連想検索動作を行うことについて説明したが、以下では、入力パターンの一致が確認された後に、再利用として出力される出力値を格納する手段の構成例について説明する。上記において図１を参照しながら説明したように、命令区間記憶部２には、再利用が可能であると判定された場合に、主記憶および／またはレジスタに出力する出力値を格納する出力値格納手段として、ＲＯ１およびＲＯ２が設けられている。

【０１０３】

出力値は、ＲＦおよびＲＢから出力されるアドレスに基づいて、出力値を記憶するＲＡＭなどの記憶手段を参照することによって得ることが可能である。しかしながら、入力パターンと同様に、出力パターンについても、出力値の項目数を可変とすることが好ましいので、出力値の格納方法に関して工夫が必要である。

【０１０４】

入力パターンに関しては、ＲＦおよびＲＢにおいて木構造によって登録されている。そして、木構造の末端となっているライン、すなわち、終端フラグＥが登録されているラインにおいて、再利用が可能であると判定されることになる。したがって、終端フラグＥが登録されている各ラインに、出力すべき出力値を格納する出力値格納手段におけるポインタを登録しておくことによって、再利用の際の出力動作を行うことが可能となる。

【０１０５】

しかしながら、入力パターンが全て一致したことが確認された時点で、出力値が格納されているポインタに基づいて出力値格納手段における格納位置が特定される場合、ポインタに基づいて格納位置を特定するという変換処理が必要となり、処理速度を低下させる要因となる。

【０１０６】

そこで、本構成例では、出力値格納手段として、ＲＯ１およびＲＯ２の２つの記憶手段を設けている。そして、ＲＯ１は、ＲＦの各ラインに１対１で対応して出力値および出力すべきアドレスを格納している。すなわち、終端フラグＥが登録されているＲＦのラインにおいて再利用が可能であると判定された場合には、そのラインに対応するＲＯ１のラインが選択され、出力値が出力される。

【０１０７】

しかしながら、このように、出力値格納手段を、ＲＦの各ラインに１対１で対応して出力値および出力すべきアドレスを格納している場合、ＲＦにおける、終端フラグＥが登録されていないＲＦのラインに対しても、ＲＯ１においてメモリ領域が確保されることになる。また、終端フラグＥが登録されているＲＦの全てのラインに対応して、ＲＯ１におい

て出力値を格納するので、同じ内容が複数箇所で記憶されている、というような冗長性が存在することになる。したがって、ＲＯ１は、高速に処理を行うという面では優れているが、メモリの利用効率としてはよくないことになる。

【０１０８】

この問題を解消するために、ＲＯ１に登録可能な項目数、すなわち出力値と出力アドレスとの組の数を少なめに設定する（図１の例では２つ）とともに、ＲＯ１に登録しきれない出力値および出力アドレスの組については、ポインタを用いて格納領域が指示される構成のＲＯ２に登録するようにしている。

【０１０９】

ＲＯ２においては、ポインタによって格納領域が指示されるので、使用されないメモリ領域はほとんど生じないことになる。また、複数の出力値および出力アドレスの組を登録する場合には、順次ポインタを用いてつなげていくことができるので、登録可能な出力値および出力アドレスの組の数を可変にすることが可能である。さらに、ＲＯ１における複数のラインから、ＲＯ２における同じ格納位置を示すポインタを指示することも可能となるので、ＲＯ２における格納情報を、ＲＯ１における複数のラインで共有することも可能となる。よって、ＲＯ２においては、格納内容の冗長性を低くすることができる。

【０１１０】

以上のように、出力値格納手段としてＲＯ１およびＲＯ２の２つを設けることによって、出力値の項目が少ない場合にはＲＯ１のみの利用により処理の高速性を実現するとともに、出力値の項目が多い場合には、項目の数を可変とすることが可能なＲＯ２を用いることによって対応している。よって、上記の構成によれば、処理の高速性とメモリ利用効率の向上とを実現することができる。

【０１１１】

（命令区間記憶部に対する登録処理）

上記では、ある命令区間の実行に際して再利用を行う場合の動作について説明した。以下では、ある命令区間の実行に際して、再利用が行えないと判断された場合に、該命令区間による入出力をＲＦ、ＲＢ、ＲＯ１、およびＲＯ２に登録する際の動作について説明する。

【０１１２】

まず、ある命令区間の実行が検出されると、ＰＣおよびReg.の値がＲＢに入力される。そして、ＲＢにおいて、連想検索により、入力されたこれらの値と、ＲＢのValueの列に登録されている命令区間先頭アドレスおよびレジスタ値とが比較される。ここで、ＲＢのValueの列に、入力された値と一致するものがないと判定された場合、該命令区間は、再利用が不可能であると判定され、演算器５Ａによる演算処理が行われる。そして、該当命令区間の演算処理が終了するまでに用いられるレジスタ入力値、主記憶入力値、主記憶出力値、およびレジスタ出力値が、ＲＢ、ＲＦ、ＲＯ１、必要に応じてＲＯ２に登録される。ここで、ＲＢおよびＲＦに登録を行う際には、上記で示したような木構造となるように、各項目が１つのラインに対応するように登録が行われる。そして、登録すべき入力パターンの最後の項目が登録されたラインにおいて、ＲＦの終端フラグを「Ｅ」とし、入力パターンの登録を終了する。

【０１１３】

一方、入力されたＰＣおよびReg.の値に一致するものが、ＲＢのValueの列に登録されている場合には、上記した連想検索動作と同様にして、次の一致比較すべき項目についての一致比較が行われる。このようにして、ＲＢおよびＲＦに登録されている入力パターンと、該当命令区間における入力パターンとの一致比較を継続していき、一致しない項目が生じた時点で、新たにノードを追加する形で、その一致しない項目についてＲＢおよびＲＦに登録が行われる。そして、登録すべき入力パターンの最後の項目が登録されたラインにおいて、ＲＦの終端フラグを「Ｅ」とし、入力パターンの登録を終了する。

【０１１４】

入力パターンの登録が終了すると、終端フラグを「Ｅ」としたＲＦにおけるラインに対

応する、ＲＯ１におけるラインに、出力値および出力アドレスの登録を行う。そして、出力値として登録すべき項目がＲＯ１に登録しきれない場合には、ポインタを用いてＲＯ２に対して登録が行われる。以上により、命令区間の登録処理が完了する。

【０１１５】

（命令区間実行時の入出力セットの生成）

ある命令区間を実行した際に、命令区間記憶部２に対して実行結果が登録されることになるが、この実行結果は、該命令区間の実行に際して、レジスタおよび／または主記憶（以降、単にレジスタ／メモリと称する）に対して行われた入出力のセットに相当するものである。以下では、命令区間記憶部２に登録すべき入出力セットをどのように生成するかについて説明する。

【０１１６】

上記した入力パターンを木構造として登録する第１および第２構成例の場合、入出力セットはＲＷ４Ａ・４Ｂによって生成され、生成された入出力セットに基づいて、ＲＢ、ＲＦ、ＲＯ１、およびＲＯ２への上記したような登録処理が行われる。ＲＷ４Ａ・４Ｂは、ある命令区間が実行された際に行われるレジスタ／メモリからの読み出し、および／または、レジスタ／メモリへの書き込みを監視し、これに基づいて入出力セットを生成する。このＲＷ４Ａ・４Ｂによる入出力セットの生成方法について以下に説明する。なお、以下の説明では、ＲＷ４Ａについて説明するが、ＲＷ４Ｂについても同様である。

【０１１７】

（ＲＷの第１構成例）

図１２は、ＲＷ４Ａのメモリ構成の概略を示す図である。同図に示すように、ＲＷ４Ａは、命令区間のＰＣ値を格納するＰＣ、入力アドレスおよび入力値を格納するＲＷＩ、および、出力アドレスおよび出力値を格納するＲＷＯのメモリを有している。ある命令区間を実行した際の入出力セットはこのＲＷ４Ａのメモリに格納され、その後、命令区間記憶部２に登録されることになる。

【０１１８】

まず、ある命令区間の実行が開始されると、そのＰＣ値がＲＷ４ＡにおけるＰＣに格納される。その後、命令区間の実行が順次行われると、レジスタ／メモリからの読み出し、および／または、レジスタ／メモリへの書き込みが順に行われることになる。

【０１１９】

命令区間実行時にレジスタ／メモリからの読み出しが行われた場合には、ＲＷ４Ａによって次の処理が行われる。

【０１２０】

（ＡＲ１）読み出しが行われたレジスタ／メモリのアドレスが、ＲＷＯに登録されているか否かが検索される。ＲＷＯに登録されている場合には、既に出力値として入出力セットに登録されている値の読み出しが行われたものであるので、入力値として登録する必要はないことになる。すなわち、該アドレスをＲＷＩに登録せずに終了する。

【０１２１】

（ＡＲ２）読み出しが行われたレジスタ／メモリのアドレスが、ＲＷＯに登録されていない場合には、該アドレスがＲＷＩに登録されているか否かが検索される。ＲＷＩに登録されている場合には、既に入力値として入出力セットに登録されている値の読み出しが行われたものであるので、さらに入力値として登録する必要はないことになる。すなわち、該アドレスをＲＷＩに登録せずに終了する。

【０１２２】

（ＡＲ３）読み出しが行われたレジスタ／メモリのアドレスが、ＲＷＯおよびＲＷＩのいずれにも登録されていない場合には、該アドレスおよび値を入力アドレスおよび入力値としてＲＷＩに登録する。

【０１２３】

また、命令区間実行時にレジスタ／メモリへの書き込みが行われた場合には、ＲＷ４Ａによって次の処理が行われる。

【０１２４】
（ＡＷ１）書き込みが行われたレジスタ／メモリのアドレスが、ＲＷＯに登録されているか否かが検索される。ＲＷＯに登録されている場合には、既に出力値として入出力セットに登録されている値の書き換えが行われたことになるので、登録されている出力アドレスに対応する出力値を、書き込みが行われた値に更新し、終了する。
【０１２５】
（ＡＷ２）書き込みが行われたレジスタ／メモリのアドレスが、ＲＷＯに登録されていない場合には、該アドレスおよび値を出力アドレスおよび出力値としてＲＷＯに登録する。
【０１２６】
以上の処理が該命令区間の終了まで行われることによって、該命令区間の入出力セットがＲＷ４Ａによって生成されることになる。生成された入出力セットは、上記したような登録処理によって命令区間記憶部２に登録される。
【０１２７】
ここで、命令区間の一例として、図１１に示す命令区間を実行した場合の例について説明する。同図において、ＰＣは、該命令区間が開始された際のＰＣ値を示している。このＰＣ値が、ＲＷ４ＡのＰＣに格納される。
【０１２８】
その後、第１行目において、レジスタにおけるアドレスＲ１に格納されている(00001000)という値が読み込まれるとともに、この読み込まれた値に100を加える演算が行われた結果の主記憶アドレス（アドレスＡ１に相当）の値を読み出す命令が行われている。この時点では、アドレスＲ１はＲＷＯおよびＲＷＩのいずれにも登録されていないので、アドレスＲ１および値(00001000)がＲＷＩに登録される。また、アドレスＡ１の値(----FF--)が読み出され、レジスタのアドレスｒｅｇ．に格納する命令が行われている。この時点では、アドレスＡ１はＲＷＯおよびＲＷＩのいずれにも登録されていないので、アドレスＡ１および値(----FF--)がＲＷＩに登録される。
【０１２９】
また、この時点では、アドレスｒｅｇ．はＲＷＯに登録されていないので、アドレスｒｅｇ．および値(----FF--)がＲＷＯに登録される。
【０１３０】
次に、第２行目において、アドレスｒｅｇ．から値を読み出して主記憶への書き込み処理が行われ、アドレスＢ１に値(----FF--)が書き込まれる。この時点では、アドレスｒｅｇ．はＲＷＯに登録されているので、ＲＷＯへの登録は行われない。また、アドレスＢ１はＲＷＯに登録されていないので、アドレスＢ１および値(----FF--)がＲＷＯに登録される。
【０１３１】
次に、第３行目において、レジスタにおけるアドレスＲ１に格納されている(00001000)という値が読み込まれるとともに、この読み込まれた値に200を加える演算が行われた結果の主記憶アドレス（アドレスＡ２に相当）の値を読み出す命令が行われている。この時点では、アドレスＲ１はＲＷＩに既に登録されているので、ＲＷＩへの登録は行われない。また、アドレスＡ２の値(--01----)が読み出され、レジスタのアドレスｒｅｇ．に格納する命令が行われている。この時点では、アドレスＡ２はＲＷＯおよびＲＷＩのいずれにも登録されていないので、アドレスＡ２および値(--01----)がＲＷＩに登録される。
【０１３２】
また、この時点では、アドレスｒｅｇ．はＲＷＯに登録されており、このＲＷＯにおけるアドレスｒｅｇ．の値が値(--01----)に更新される。
【０１３３】
次に、第４行目において、アドレスｒｅｇ．から値を読み出して主記憶への書き込み処理が行われ、アドレスＢ２に値(--01----)が書き込まれる。この時点では、アドレスｒｅｇ．はＲＷＯに登録されているので、ＲＷＯへの登録は行われない。また、アドレスＢ２

はＲＷＯに登録されていないので、アドレスＢ２および値(--01----)がＲＷＯに登録される。

【０１３４】

次に、第５行目において、アドレスＡ３の値(5678----)が読み出され、レジスタのアドレスｒｅｇ．に格納する命令が行われている。この時点では、アドレスＡ３はＲＷＯおよびＲＷＩのいずれにも登録されていないので、アドレスＡ３および値(5678----)がＲＷＩに登録される。

【０１３５】

また、この時点では、アドレスｒｅｇ．はＲＷＯに登録されており、このＲＷＯにおけるアドレスｒｅｇ．の値が値(5678----)に更新される。

【０１３６】

最後に、第６行目において、アドレスｒｅｇ．から値を読み出して主記憶への書き込み処理が行われ、アドレスＢ３に値(5678----)が書き込まれる。この時点では、アドレスｒｅｇ．はＲＷＯに登録されているので、ＲＷＯへの登録は行われない。また、アドレスＢ３はＲＷＯに登録されていないので、アドレスＢ３および値(5678----)がＲＷＯに登録される。以上の処理によって、図１２に示すＲＷ４Ａの入出力セットが生成される。

【０１３７】

以上のようにして生成された入出力セットは、図１３に示すような木構造として、命令区間記憶部２に登録される。この木構造において、登録されている入力パターンは、ルートノードからリーフへ至る１本のパスとして命令区間記憶部２に保持される。以降、命令区間を実行する前に、該命令区間の入力パターンが、登録されている入力パターンと同じであるかを判断するために、図３に示したように、ルートノードから順に、ノードに記録されているアドレスを参照し、得られた値と一致するノードを連想検索機構を用いて選択することを繰り返すことになる。

【０１３８】

（木構造連想検索の問題）

上記の木構造の場合、入力パターンを１つずつ順に読み出して連想検索を行い、一致するノードが見つかった後に、次のノードの選択を行うことになる。すなわち、先行するノードの検索が完全に終了してから次のノードの検索が開始されることになる。

【０１３９】

ここで、ＣＡＭ／ＲＡＭで構成される連想検索装置は、一般的に長レイテンシ高スループットの特性を有している。すなわち、一般的な連想検索装置は、１つの検索入力が行われてから出力されるまでの期間は比較的長いものであるが、複数の検索入力を同時に処理して出力することが可能であるという特性を有している。これに対し、上記のように、先行するノードの検索が完全に終了してから次のノードの検索が開始される、というような検索が行われる場合、連想検索装置における高スループットの能力を利用することができないことになり、連想検索装置の能力を十全に発揮することができないことになる（問題１）。

【０１４０】

また、上記の木構造の場合、命令区間の入力パターンが参照順に一本のパスとして実現されており、入力パターン全体が一致しなければ出力を再利用することができないことになる。ここで、次のような例を想定する。まず、ある命令区間を実行した際の入力パターンのうち、前半がパターンＡ１、後半がパターンＡ２となっており、パターンＡ１に対応する出力がＸ１、パターンＡ２に対応する出力がＸ２となっていたとする。また、別の命令区間を実行した際の入力パターンのうち、前半がパターンＢ１、後半がパターンＢ２となっており、パターンＢ１に対応する出力がＹ１、パターンＢ２に対応する出力がＹ２となっていたとする。その後、ある命令区間を実行しようとした時の入力パターンのうち、前半がパターンＡ１、後半がパターンＢ２となっていた場合、入力パターンの前半および後半のそれぞれについては再利用が可能であるものの、入力パターン全体としては過去に同一パターンが出現していないので、実際には再利用することができないことになる（問

題２）。

【０１４１】

（ＲＷの第２構成例）

上記の２つの問題は、ある呼び出し時点における命令区間の入力パターンをルートノードからリーフへ至る１本のパスによる表現したことによって生じたものである。これらの問題を解決するためには、入力パターンをグループ分割し、各グループ毎に過去の入力パターンを保持する木構造を構成し、さらに、複数木構造の同時探索を可能とすることによって連想検索装置を有効に利用できるようにすることが必要である。例えば、図１３に示すような木構造を、図１４に示すような複数の木構造に分割して、ルートノードからリーフに至るパスに対応する入力グループ毎に独立に再利用が行われるようにすればよい。

【０１４２】

上記のように、木構造の分割を実現するためには、各入力グループ同士の間でデータ依存関係がないことが必要である。すなわち、ある入力パターンをグループＡとグループＢとに分割した場合において、グループＡの入力がグループＢの入力に依存する場合、あるいは、グループＢの入力がグループＡの入力に依存する場合には、グループ分割したとしても、各グループを独立に再利用できる可能性は極めて低くなる。

【０１４３】

データ依存関係がないグループに分割するには、入力パターンを生成する際に、データ依存関係の解析を行う必要がある。すなわち、ＲＷ４Ａが、データ依存関係の解析を行った上で、入力パターンをデータ依存関係がないグループに分割して入出力セットを生成するようにすればよいことになる。

【０１４４】

図１０は、上記を実現する第２構成例としてのＲＷ４Ａの概略構成を示している。同図に示すように、ＲＷ４Ａは、命令区間のＰＣ値を格納するＰＣ、入力アドレスおよび入力値を格納するＲＷＩ、出力アドレスおよび出力値を格納するＲＷＯ、依存関係格納部Ｍ、行間論理積比較部（入出力グループ設定手段）ＭＲ、およびグループＩＤ格納部ＩＤを有している。

【０１４５】

依存関係格納部Ｍは、２次元配列のメモリであり、各メモリ要素には０または１が記憶されるようになっている。また、依存関係格納部Ｍにおいて、各列はＲＷＩに登録されている各入力アドレスおよび入力値に対応しており、各行はＲＷＯに登録されている各出力アドレスおよび出力値に対応している。そして、依存関係格納部Ｍは、各出力アドレスおよび出力値が、どの入力アドレスおよび入力値を起源とするものであるかを示している。

【０１４６】

行間論理積比較部ＭＲは、依存関係格納部Ｍに格納されている各行成分間の論理積演算を行い、１以上の出力アドレスおよび出力値を含む出力パターンと、１以上の入力アドレスおよび入力値を含む入力パターンとからなる入出力グループを設定する演算部である。この行間論理積比較部ＭＲによる論理積演算の詳細については後述する。

【０１４７】

グループＩＤ格納部ＩＤは、行間論理積比較部ＭＲによる論理積演算結果に基づいて、依存関係格納部Ｍにおける各列に対応する入力アドレスおよび入力値に対して付与されるグループＩＤを格納するメモリである。このグループＩＤの詳細については後述する。

【０１４８】

ある命令区間の実行が開始されると、まず依存関係格納部Ｍにおける各メモリ要素の初期値として、全て０に設定される。そして、該命令区間のＰＣ値がＲＷ４ＡにおけるＰＣに格納される。その後、命令区間の実行が順次行われると、レジスタ／メモリからの読み出し、および／または、レジスタ／メモリへの書き込みが順に行われることになる。

【０１４９】

命令区間実行時にレジスタ／メモリからの読み出しが行われた場合には、ＲＷ４Ａによって次の処理が行われる。

【０１５０】
（ＢＲ１）読み出しが行われたレジスタ／メモリのアドレスが、ＲＷＯに登録されているか否かが検索される。ＲＷＯに登録されている場合には、既に出力値として入出力セットに登録されている値の読み出しが行われたものであるので、入力値として登録する必要はないことになる。すなわち、該アドレスをＲＷＩに登録せずに終了する。
【０１５１】
この時、ＲＷＯにおいて既に登録されているアドレスに対応する依存関係格納部Ｍの行成分の各メモリ要素の値が取り出され、行成分のみの１次元行列としての暫定行列Ａ（ｘ）として記憶される。ここで、ｘは暫定行列Ａが生成された順に付される番号とする。この暫定行列Ａ（ｘ）は、後述する書き込み処理が終了した時点で初期化される。なお、この暫定行列Ａ（ｘ）は、図１０では図示していないが、暫定行列Ａ（ｘ）を複数格納することができる暫定行列格納メモリに格納されることになる。
【０１５２】
（ＢＲ２）読み出しが行われたレジスタ／メモリのアドレスが、ＲＷＯに登録されていない場合には、該アドレスがＲＷＩに登録されているか否かが検索される。ＲＷＩに登録されている場合には、既に入力値として入出力セットに登録されている値の読み出しが行われたものであるので、さらに入力値として登録する必要はないことになる。すなわち、該アドレスをＲＷＩに登録せずに終了する。
【０１５３】
この時、ＲＷＩにおいて既に登録されているアドレスに対応する依存関係格納部Ｍの列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列Ａ（ｘ）が記憶される。
【０１５４】
（ＢＲ３）読み出しが行われたレジスタ／メモリのアドレスが、ＲＷＯおよびＲＷＩのいずれにも登録されていない場合には、該アドレスおよび値を入力アドレスおよび入力値としてＲＷＩに登録する。
【０１５５】
この時、新たに追加した入力アドレスおよび入力値（エントリ）に対応する依存関係格納部Ｍの列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列Ａ（ｘ）が記憶される。
【０１５６】
また、命令区間実行時にレジスタ／メモリへの書き込みが行われた場合には、ＲＷ４Ａによって次の処理が行われる。
【０１５７】
（ＢＷ１）書き込みが行われたレジスタ／メモリのアドレスが、ＲＷＯに登録されているか否かが検索される。ＲＷＯに登録されている場合には、既に出力値として入出力セットに登録されている値の書き換えが行われたことになるので、登録されている出力アドレスに対応する出力値を、書き込みが行われた値に更新し、終了する。
【０１５８】
この時、ＲＷＯにおいて既に登録されているアドレスに対応する依存関係格納部Ｍの行成分が、その時点で記憶されている全ての暫定行列Ａ（ｘ）の論理和に置き換えられる。これにより、ＲＷＯにおいて既に登録されている出力アドレス／値に対する出力の起源となる入力アドレス／値のパターンが、該出力アドレスに対応する依存関係格納部Ｍの行成分によって示されることになる。書き込み処理が終了し、暫定行列Ａ（ｘ）の論理和への置き換えが完了すると、暫定行列Ａ（ｘ）が全て初期化される。
【０１５９】
（ＢＷ２）書き込みが行われたレジスタ／メモリのアドレスが、ＲＷＯに登録されていない場合には、該アドレスおよび値を出力アドレスおよび出力値としてＲＷＯに登録する。
【０１６０】

この時、新たに追加した出力アドレスおよび出力値（エントリ）に対応する依存関係格納部Ｍの行成分が、その時点で記憶されている全ての暫定行列Ａ（ｘ）の論理和に置き換えられる。これにより、ＲＷＯに新たに登録した出力アドレス／値に対する出力の起源となる入力アドレス／値のパターンが、該出力アドレスに対応する依存関係格納部Ｍの行成分によって示されることになる。書き込み処理が終了し、暫定行列Ａ（ｘ）の論理和への置き換えが完了すると、暫定行列Ａ（ｘ）が全て初期化される。

【０１６１】

ここで、命令区間の一例として、図１１に示す命令区間を実行した場合の例について説明する。同図において、ＰＣは、該命令区間が開始された際のＰＣ値を示している。このＰＣ値が、ＲＷ４ＡのＰＣに格納される。

【０１６２】

その後、第１行目において、レジスタにおけるアドレスＲ１に格納されている(00001000)という値が読み込まれるとともに、この読み込まれた値に100を加える演算が行われた結果の主記憶アドレス（アドレスＡ１に相当）の値を読み出す命令が行われている。この時点では、アドレスＲ１はＲＷＯおよびＲＷＩのいずれにも登録されていないので、アドレスＲ１および値(00001000)がＲＷＩに登録される。

【０１６３】

この時、アドレスＲ１に対応する依存関係格納部Ｍの列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列Ａ（１）［１０００］が記憶される。

【０１６４】

また、アドレスＡ１の値(----FF--)が読み出され、レジスタのアドレスｒｅｇ．に格納する命令が行われている。この時点では、アドレスＡ１はＲＷＯおよびＲＷＩのいずれにも登録されていないので、アドレスＡ１および値(----FF--)がＲＷＩに登録される。

【０１６５】

この時、アドレスＡ１に対応する依存関係格納部Ｍの列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列Ａ（２）［０１００］が記憶される。

【０１６６】

また、この時点では、アドレスｒｅｇ．はＲＷＯに登録されていないので、アドレスｒｅｇ．および値(----FF--)がＲＷＯに登録される。この時、新たに追加したアドレスｒｅｇ．に対応する依存関係格納部Ｍの行成分が、その時点で記憶されている全ての暫定行列Ａ（１）～Ａ（２）の論理和［１１００］に置き換えられる。その後、暫定行列Ａ（ｘ）が初期化される。

【０１６７】

次に、第２行目において、アドレスｒｅｇ．から値を読み出して主記憶への書き込み処理が行われ、アドレスＢ１に値(----FF--)が書き込まれる。この時点では、アドレスｒｅｇ．はＲＷＯに登録されているので、ＲＷＯへの登録は行われない。この時、アドレスｒｅｇ．に対応する依存関係格納部Ｍの行成分が取り出され、暫定行列Ａ（１）［１１００］が記憶される。

【０１６８】

また、アドレスＢ１はＲＷＯに登録されていないので、アドレスＢ１および値(----FF--)がＲＷＯに登録される。

【０１６９】

この時、新たに追加した出力アドレスに対応する依存関係格納部Ｍの行成分が、その時点で記憶されている全ての暫定行列Ａ（１）の論理和［１１００］に置き換えられる。その後、暫定行列Ａ（ｘ）が初期化される。

【０１７０】

次に、第３行目において、レジスタにおけるアドレスＲ１に格納されている(00001000)という値が読み込まれるとともに、この読み込まれた値に200を加える演算が行われた結果の主記憶アドレス（アドレスＡ２に相当）の値を読み出す命令が行われている。この時点では、アドレスＲ１はＲＷＩに既に登録されているので、ＲＷＩへの登録は行われない

。
【０１７１】
この時、アドレスＲ１に対応する依存関係格納部Ｍの列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列Ａ（１）［１０００］が記憶される。
【０１７２】
また、アドレスＡ２の値(--01----)が読み出され、レジスタのアドレスｒｅｇ．に格納する命令が行われている。この時点では、アドレスＡ２はＲＷＯおよびＲＷＩのいずれにも登録されていないので、アドレスＡ２および値(--01----)がＲＷＩに登録される。
【０１７３】
この時、アドレスＡ２に対応する依存関係格納部Ｍの列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列Ａ（２）［００１０］が記憶される。
【０１７４】
また、この時点では、アドレスｒｅｇ．はＲＷＯに登録されており、このＲＷＯにおけるアドレスｒｅｇ．の値が値(--01----)に更新される。この時、更新されたアドレスｒｅｇ．に対応する依存関係格納部Ｍの行成分が、その時点で記憶されている全ての暫定行列Ａ（１）～Ａ（２）の論理和［１０１０］に置き換えられる。その後、暫定行列Ａ（ｘ）が初期化される。
【０１７５】
次に、第４行目において、アドレスｒｅｇ．から値を読み出して主記憶への書き込み処理が行われ、アドレスＢ２に値(--01----)が書き込まれる。この時点では、アドレスｒｅｇ．はＲＷＯに登録されているので、ＲＷＯへの登録は行われない。この時、アドレスｒｅｇ．に対応する依存関係格納部Ｍの行成分が取り出され、暫定行列Ａ（１）［１０１０］が記憶される。
【０１７６】
また、アドレスＢ２はＲＷＯに登録されていないので、アドレスＢ２および値(--01----)がＲＷＯに登録される。
【０１７７】
この時、新たに追加した出力アドレスに対応する依存関係格納部Ｍの行成分が、その時点で記憶されている全ての暫定行列Ａ（１）の論理和［１０１０］に置き換えられる。その後、暫定行列Ａ（ｘ）が初期化される。
【０１７８】
次に、第５行目において、アドレスＡ３の値(5678----)が読み出され、レジスタのアドレスｒｅｇ．に格納する命令が行われている。この時点では、アドレスＡ３はＲＷＯおよびＲＷＩのいずれにも登録されていないので、アドレスＡ３および値(5678----)がＲＷＩに登録される。
【０１７９】
この時、アドレスＡ３に対応する依存関係格納部Ｍの列に対応するメモリ要素を１とし、その他のメモリ要素を０とした暫定行列Ａ（１）［０００１］が記憶される。
【０１８０】
また、この時点では、アドレスｒｅｇ．はＲＷＯに登録されており、このＲＷＯにおけるアドレスｒｅｇ．の値が値(5678----)に更新される。この時、更新されたアドレスｒｅｇ．に対応する依存関係格納部Ｍの行成分が、その時点で記憶されている全ての暫定行列Ａ（１）の論理和［０００１］に置き換えられる。その後、暫定行列Ａ（ｘ）が初期化される。
【０１８１】
最後に、第６行目において、アドレスｒｅｇ．から値を読み出して主記憶への書き込み処理が行われ、アドレスＢ３に値(5678----)が書き込まれる。この時点では、アドレスｒｅｇ．はＲＷＯに登録されているので、ＲＷＯへの登録は行われない。この時、アドレスｒｅｇ．に対応する依存関係格納部Ｍの行成分が取り出され、暫定行列Ａ（１）［０００１］が記憶される。

【０１８２】
また、アドレスＢ３はＲＷＯに登録されていないので、アドレスＢ３および値(5678----)がＲＷＯに登録される。
【０１８３】
この時、新たに追加した出力アドレスに対応する依存関係格納部Ｍの行成分が、その時点で記憶されている全ての暫定行列Ａ（１）の論理和［０００１］に置き換えられる。その後、暫定行列Ａ（ｘ）が初期化される。以上の処理によって、図１０に示すＲＷ４Ａの入出力セットが生成される。
【０１８４】
以上のように依存関係格納部Ｍを生成することによって、命令区間の実行完了時には、次の情報が得られていることになる。
【０１８５】
（Ｒｓ１）依存関係格納部Ｍの行成分は、対応する出力アドレス／値の起源となる入力アドレス／値を１によって示している。
【０１８６】
（Ｒｓ２）ある行成分Ｍａにおいて１が示されている入力アドレス／値の組が１つの入力グループを形成し、該入力グループが一致した場合に再利用可能な出力アドレス／値は、行成分Ｍａに対応する出力アドレス／値である。
【０１８７】
（Ｒｓ３）「ある行成分Ｍａの反転」と「ある行成分Ｍｂ」との論理積が全て０である場合、Ｍａにおける１のパターンは、Ｍｂにおける１のパターンを包含する。すなわち、Ｍａに属する入力アドレス／値の組が１つの入力グループを形成するとともに、該入力グループが一致した場合に再利用可能な出力アドレス／値は、Ｍａに対応する出力アドレス／値、および、Ｍｂに対応する出力アドレス／値となる。
【０１８８】
（Ｒｓ４）「ある行成分Ｍａ」と「ある行成分Ｍｂ」との論理積が全て０である場合、Ｍａに属する入力アドレス／値と、Ｍｂに属する入力アドレス／値とは互いに独立している。
【０１８９】
以上の情報に基づいて、ＲＷ４Ａは、入出力セットを複数の入出力グループに分割する。まず、上記の（Ｒｓ３）に関連する処理として、依存関係格納部Ｍにおいて、「ある行成分Ｍａの反転」と「ある行成分Ｍｂ」との論理積が全て０になる行成分の組が行間論理積比較部ＭＲによって抽出される。抽出された行成分の組のうち、入力アドレス／値の組を最も多く含む行成分、すなわち、他の行成分における入力アドレス／値の組を全て含んだ行成分が上位行成分として選択される。そして、抽出された行成分のうち、上位行成分以外の下位行成分が削除される。この処理によって、冗長な入出力グループを排除することができる。
【０１９０】
次に、下位行成分が削除された状態において、上記の（Ｒｓ４）に関連する処理として、「ある行成分Ｍａ」と「ある行成分Ｍｂ」との論理積が全て０になる行成分の組が行間論理積比較部ＭＲによって抽出される。そして、抽出された行成分の組のうち、他のどの行成分に対しても論理積が全て０になる行成分がさらに抽出される。ここで抽出された行成分は、他のどの行成分に対しても依存関係を有さないことになるので、これを独立行成分と設定し、これ以外を非独立行成分と設定する。
【０１９１】
独立行成分は、それぞれ対応する入力アドレス／値の組および出力アドレス／値の組が抽出されて、１つの入出力グループとして設定される。一方、非独立行成分は、次の２つの処理のいずれかによって入出力グループとして設定される。
【０１９２】
第１の処理としては、非独立行成分の全てに含まれる入力アドレス／値の組および出力

アドレス／値の組の総和を１つの入出力グループとして設定する処理である。第２の処理としては、非独立行成分のそれぞれをそのまま入出力グループとして設定する処理である。第１の処理を行う場合、入出力グループの数を必要以上に増大させることがなくなるので、命令区間記憶部２におけるメモリ使用容量を低減することができる。一方、第２の処理を行う場合、入出力グループの数が比較的多くなり、命令区間記憶部２におけるメモリ使用容量が比較的大きくなるという問題はあるが、命令区間記憶部２において、同時に検索すべき木構造の数を増やすことができるので、連想検索装置における高スループットの能力を利用することが可能となる。

【０１９３】

以上のようにして入出力グループが設定されると、これに基づいて、行間論理積比較部ＭＲが、各入出力グループにグループＩＤを付与し、ＲＷＩに登録されている入力アドレス／値のそれぞれに対して、どのグループＩＤに含まれているものであるかを示す情報をグループＩＤ格納部ＩＤに格納する。これにより、グループＩＤ格納部ＩＤの内容を見ることによって、各入出力グループにおける入力パターンを特定することが可能となる。

【０１９４】

以上のように、ＲＷ４Ａは、１つ以上の入出力グループを生成し、生成した入出力グループを命令区間記憶部２に対して実行結果として登録する。このような処理により、１つの命令区間の実行結果が、１つ以上の入出力グループとして命令区間記憶部２に登録されることになる。よって、ある命令区間を再利用によって実行する際に、以前に実行された命令区間の入力パターンの一部しか一致していない場合でも、再利用を行うことが可能となる確率を高めることができる。また、同時に検索すべき木構造が複数存在する確率を高めることができるので、連想検索装置における高スループットの能力を利用することが可能となり、処理速度の向上を期待することができる。

【０１９５】

なお、本実施形態においては、ＲＷ４Ａによって生成された入出力グループは、入力パターンを木構造として登録する命令区間記憶部２に登録されるようになっているが、これに限定されるものではない。すなわち、ＲＷ４Ａによって生成された入出力グループを、命令区間の実行結果を再利用することが可能な形態で登録することが可能な命令区間記憶部であれば、本実施形態に係るＲＷ４Ａを適用することが可能である。

【０１９６】

（レジスタ値の詳細）

レジスタ入出力値としては、引数、返り値（Args.）、および、引数および返り値以外のレジスタおよび条件コード(Regs.,CC)が挙げられる。本実施形態では、ＳＰＡＲＣアーキテクチャレジスタのうち、汎用レジスタ%g0-7、%o0-7、%l0-7、%i0-7、浮動小数点レジスタ%f0-31、条件コードレジスタICC、浮動小数点条件コードレジスタFCCを用いるようになっている（詳細は後述する）。このうち、リーフ関数の入力は汎用レジスタ%o0-5、出力は汎用レジスタ%o0-1または%f0-1、また、非リーフ関数の入力は汎用レジスタ%i0-5、出力は汎用レジスタ%i0-1または%f0-1、になり、入力は、arg[0-5]、出力は、rti[0-1]または%rtf[0-1]に登録される。ＳＰＡＲＣ－ＡＢＩの規定では、これら以外のレジスタは関数の入出力にはならないので、関数に関しては、レジスタ入出力値としては、Args.がＲＢ、およびＲＯ１／ＲＯ２に登録されることになる。

【０１９７】

一方、ＳＰＡＲＣ－ＡＢＩの規定では、ループの入出力に関しては、用いられるレジスタの種類を特定することはできないので、ループの入出力を特定するには、全ての種類のレジスタに関してＲＢに登録する必要がある。よって、ループに関しては、レジスタ入出力値として、Regs.,CCに相当する、%g0-7、%o0-7、%l0-7、%i0-7、%f0-31、ICC、FCCが登録されることになる。

【０１９８】

（多重再利用）

１レベルで上記のような再利用機構を用いた場合、図１０（ａ）に示した例で言えば、

リーフ関数としての関数Ｂや、関数Ｂの内部にあるループＣなどをそれぞれ再利用することが可能となる。これに対して、ある関数を一度実行しただけで、その関数の内部に含まれる関数やループを含む全ての命令区間が再利用可能となるように登録を行う仕組みが多重再利用である。例えば上記の例で言えば、多重再利用によれば、関数Ａを一度実行しただけで、入れ子関係にあるＡ，Ｂ，Ｃの全ての命令区間が再利用可能となる。以下に、多重再利用を実現する上で必要とされる機能拡張について説明する。

【０１９９】

図６に、一例として、関数Ａおよび関数Ｄの概念的な構造を示す。同図に示す例では、関数Ａの内部にループＢが存在しており、ループＢの内部にループＣが存在しており、ループＣにおいて関数Ｄが呼び出されるようになっている。そして、関数Ｄの内部にループＥが存在しており、ループＥの内部にループＦが存在している。

【０２００】

図７は、図６に示す関数Ａ，ＤおよびループＢ，Ｃ，Ｅ，Ｆの入れ子構造において、内側の構造のレジスタ入出力（太枠セル領域）が、外側の構造のレジスタ入出力となる影響範囲（矢印）について示している。例えば、ループＦの内部において入力として参照された％ｉ０～５は、ループＥおよび関数Ｄに対する入力でもあり、さらに、関数Ｄを呼び出したループＣおよびループＢに対する入力（ただし％ｏ０～５に読み替える）でもある。一方、関数Ａにとって％ｏ０～５は局所変数に相当するので、％ｉ０～５（％ｏ０～５）は、関数Ａに対してのレジスタ入力とはならない。すなわち、％ｉ０～５（％ｏ０～５）の影響範囲はループＢまでとなる。別の見方をすれば、関数Ｄの内部で％ｉ０～５が参照された場合には、ループＢが直接的に％ｏ０～５を参照しなくても、％ｏ０～５をループＢの入力値として登録する必要がある。ループＦ内部において出力された％ｉ０～１についても同様である。

【０２０１】

浮動小数点レジスタはレジスタウィンドウに含まれないので、出力された％ｆ０～１は、関数Ａを含む全階層の出力となる。一方、その他のレジスタ入出力は、関数を超えて影響がおよぶことはない。すなわち、ループＦ内部における入出力、すなわち、レジスタ入力としての％ｉ６～７、％g,l,o、％ｆ０～３１、％icc、％fcc、およびレジスタ出力としての％Ｉ２～７、％g,l,o、％ｆ２～３１、％icc、％fccの影響範囲はループＥまでとなる。主記憶に対する入出力については、前述した、関数呼び出し直前の％sp(SP)と比較する方法を入れ子の全階層に対して適用することにより、影響範囲を特定することができる。

【０２０２】

ここで、上記のようなＲＷ４Ａ、ＲＷ４Ｂ、および命令区間記憶部２の構成によれば、複数の命令区間の入出力を個別に記録することが可能であるので、多重再利用を実現することが可能となる。

【０２０３】

（並列事前実行）

以上に述べた、関数やループの多重再利用では、同一パラメータが出現する間隔が長い場合や、パラメータが単調に変化し続ける場合には全く効果がないことになる。すなわち、ＲＢエントリの生存時間よりも同一パラメータが出現する間隔が長い場合には、ある関数またはループがＲＢに登録されたとしても、その登録された関数またはループに関して同一パラメータが次に出現した際には、すでにその関数またはループがＲＢエントリから消えていることになり、再利用できないことになる。また、パラメータが単調に変化し続ける場合には、該当する関数やループがＲＢに登録されていても、パラメータが異なることによって再利用できないことになる。

【０２０４】

これに対して、多重再利用を行うプロセッサとしてのＭＳＰ１Ａとは別に、命令区間の事前実行によってＲＢエントリを有効にするプロセッサとしてのＳＳＰ１Ｂを複数個設けることによって、さらなる高速化を図ることができる。

【０２０５】

並列事前実行機構を行うためのハードウェア構成は、前記した図２に示すような構成となる。同図に示すように、ＲＷ４Ａ・４Ｂ、演算器５Ａ・５Ｂ、レジスタ６Ａ・６Ｂ、キャッシュ７Ａ・７Ｂは、各プロセッサごとに独立して設けられている一方、命令区間記憶部２、および主記憶３は全てのプロセッサが共有するようになっている。同図において、破線は、ＭＳＰ１ＡおよびＳＳＰ１Ｂが命令区間記憶部２に対して入出力を登録するパスを示している。

【０２０６】

ここで、並列事前実行を実現する上での課題は、（１）どのように主記憶一貫性を保つか、（２）どのように入力を予測するか、の２点が挙げられる。以下に、これらの課題に対する解決手法について説明する。

【０２０７】

（主記憶一貫性に関する課題の解決方法）

まず、上記の課題（１）どのように主記憶一貫性を保つかについて説明する。特に予測した入力パラメータに基づいて命令区間を実行する場合、主記憶に書き込む値がＭＳＰ１ＡとＳＳＰ１Ｂとで異なることになる。これを解決するために、図２に示すように、ＳＳＰ１Ｂは、ＲＢへの登録対象となる主記憶参照には命令区間記憶部２、また、その他の局所的な参照にはＳＳＰ１Ｂごとに設けた局所メモリとしてのＬｏｃａｌ７Ｂを使用することとし、Ｃａｃｈｅ７Ｂおよび主記憶３への書き込みを不要としている。なお、ＭＳＰ１Ａが主記憶３に対して書き込みを行った場合には、対応するＳＳＰ１Ｂのキャッシュラインが無効化される。

【０２０８】

具体的には、命令区間記憶部２への登録対象のうち、読み出しが先行するアドレスについては主記憶３を参照し、ＭＳＰ１Ａと同様にアドレスおよび値をＲＢへ登録する。以後、主記憶３ではなく命令区間記憶部２を参照することによって、他のプロセッサからの上書きによる矛盾の発生を避けることができる。局所的な参照については、読み出しが先行するということは、変数を初期化せずに使うことに相当し、値は不定でよいことになるので、主記憶３を参照する必要はない。

【０２０９】

なお、局所メモリとしてのＬｏｃａｌ７Ｂの容量は有限であり、関数フレームの大きさがＬｏｃａｌ７Ｂの容量を超えた場合など、実行を継続できない場合は、事前実行を打ち切るようにする。また、事前実行の結果は主記憶３に書き込まれないので、事前実行結果を使って、さらに次の事前実行を行うことはできない。

【０２１０】

（入力の予測方法）

次に、上記の課題（２）どのように入力を予測するかについて説明する。事前実行に際しては、命令区間記憶部２の使用履歴に基づいて将来の入力を予測し、ＳＳＰ１Ｂへ渡す必要がある。このために、命令区間記憶部２に記憶されている各入力パターンごとに小さなプロセッサを設け、ＭＳＰ１ＡやＳＳＰ１Ｂとは独立して入力予測値を求めるようにする。

【０２１１】

具体的には、最後に出現した引数（Ｂ）および最近出現した２組の引数の差分（Ｄ）に基づいて、ストライド予測を行う。なお、Ｂ＋Ｄに基づく命令区間の実行はＭＳＰ１Ａがすでに開始していると考える。ＳＳＰ１ＢがＮ台の場合には、用意する入力予測値は、Ｂ＋Ｄ×２からＢ＋Ｄ×（Ｎ＋１）までの範囲とする。

【０２１２】

以上のように入力予測を行えば、上記した入力パラメータが単調に変化し続けるような場合に、事前に予測しておいた結果に基づいて効果的に再利用を行うことが可能となる。

【産業上の利用可能性】

【０２１３】

本発明に係るデータ処理装置は、上記したようにＳＰＡＲＣプロセッサに適用すること

が可能である。また、ＳＰＡＲＣプロセッサと同様に、３２本以上の汎用レジスタを有する多くのＲＩＳＣプロセッサにも適用することが可能である。

【図面の簡単な説明】

【０２１４】

【図１】本発明の一実施形態に係るデータ処理装置が備える命令区間記憶部の概略構成を示す図である。

【図２】上記データ処理装置の概略構成を示すブロック図である。

【図３】上記命令区間記憶部における連想検索動作の具体例を示す図である。

【図４】同図（ｂ）は、上記命令区間記憶部における連想検索動作の他の具体例を示す図であり、同図（ａ）は、同図（ｂ）における連想検索動作を木構造として示す図である。

【図５】同図（ｂ）は、上記命令区間記憶部における連想検索動作のさらに他の具体例を示す図であり、同図（ａ）は、同図（ｂ）における連想検索動作を木構造として示す図である。

【図６】関数およびループが入れ子構造となっている状態の一例を示す図である。

【図７】関数の入れ子構造において、内側の構造のレジスタ入出力が、外側の構造のレジスタ入出力となる影響範囲を示す図である。

【図８】比較例におけるＲＦおよびＲＢの概略構成を示す図である。

【図９】比較例における検索動作の例を示す図である。

【図１０】第２構成例としてのＲＷの概略構成を示す図である。

【図１１】命令区間の一例を示す図である。

【図１２】ＲＷの第１構成例におけるメモリ構成の概略を示す図である。

【図１３】ＲＷの第１構成例によって生成された入出力セットが木構造として登録された状態を示す図である。

【図１４】ＲＷの第２構成例によって生成された入出力セットが木構造として登録された状態を示す図である。

【図１５】同図（ａ）は、関数Ａが関数Ｂを呼び出す構造を概念的に示す概念図であり、同図（ｂ）は、同図（ａ）に示すプログラム構造を実行する際の主記憶におけるメモリマップを示す図である。

【図１６】関数Ａが関数Ｂを呼び出す場合の、メモリマップにおける引数およびフレームの概要を示す図である。

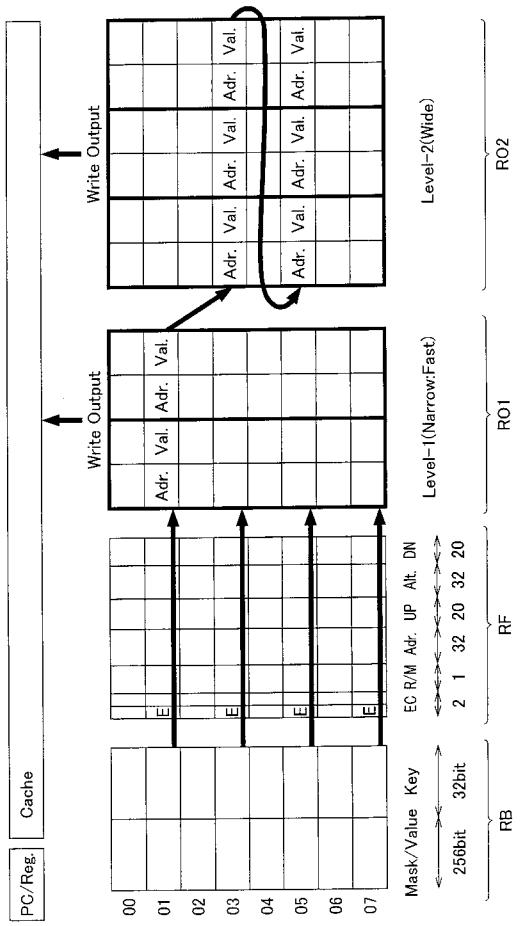
【図１７】１つの関数を再利用するための従来の再利用表を示す図である。

【符号の説明】

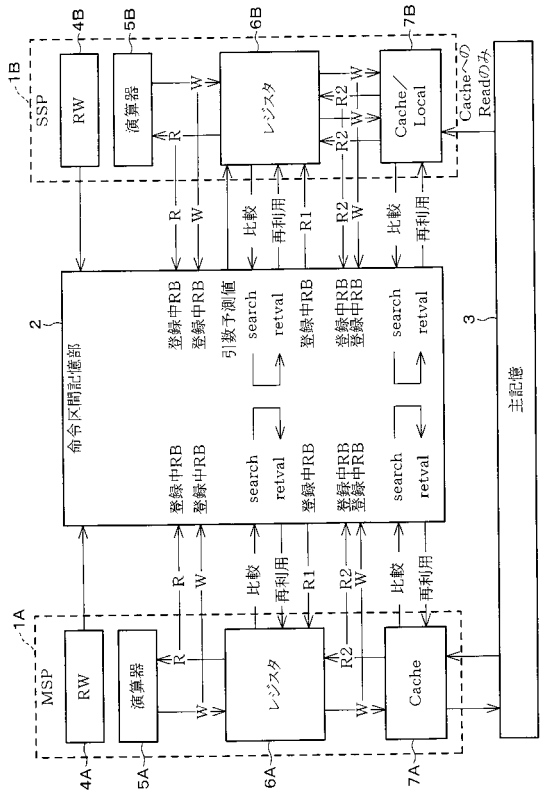
【０２１５】

１Ａ　ＭＳＰ
１Ｂ　ＳＳＰ
２　命令区間記憶部（命令区間記憶手段）
３　主記憶（主記憶手段）
４Ａ・４Ｂ　ＲＷ（入出力生成手段）
５Ａ・５Ｂ　演算器（第１・第２の演算手段）
６Ａ・６Ｂ　レジスタ
７Ａ・７Ｂ　Ｃａｃｈｅ
Ｍ　依存関係格納部
ＭＲ　行間論理積比較部（入出力グループ設定手段）

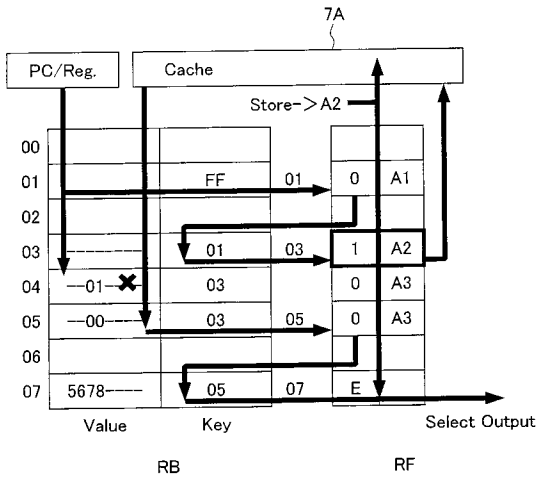
【図１】


Write Output
Write Output
Level-2(Wide)
Level-1(Narrow:Fast)
RO2
RO1
RF
RB
Cache
PC/Reg.
Val.
Adr.
E
E0 R/M Adr. UP Alt. DN
2 1 32 20 32 20
Mask/Value Key
256bit 32bit
00 01 02 03 04 05 06 07

【図２】

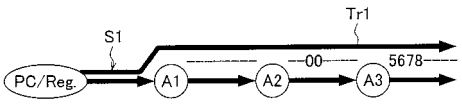

1B
SSP
RW
演算器
4B
5B
6B
7B
レジスタ
Cache／Local
Cacheへの Readのみ
2
命令区間記憶部
登録中RB
登録中RB
引数予測値
search
retval
3
主記憶
1A
MSP
RW
演算器
レジスタ
Cache
4A
5A
6A
7A
比較
再利用
R
W
R1
R2

【図３】

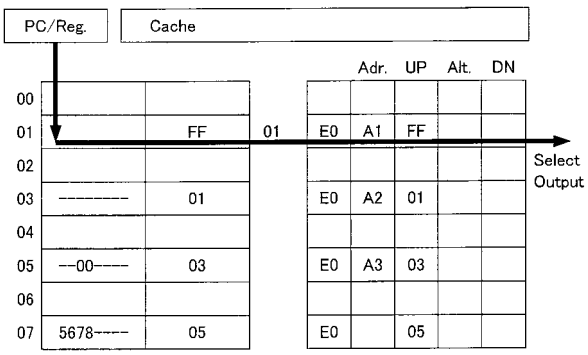

7A
PC/Reg.
Cache
Store->A2
00
01
02
03
04
05
06
07
FF
01
01
03
03
05
5678----
0
A1
1
A2
0
A3
0
A3
E
Value
Key
Select Output
RB
RF

【図４】

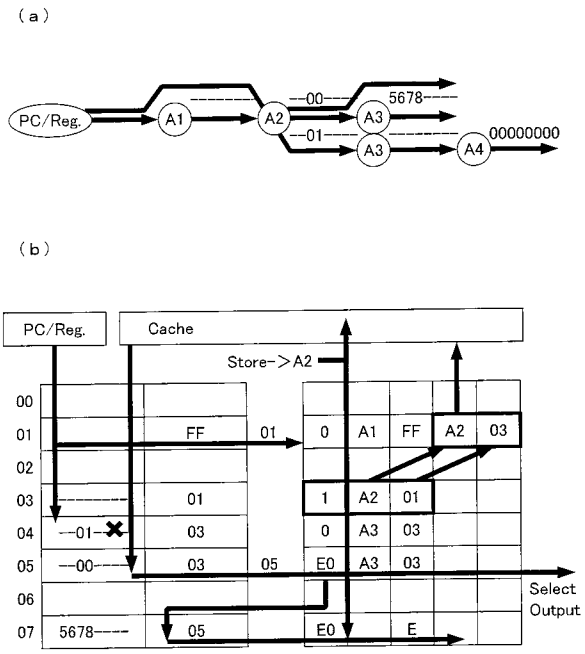
（ａ）


Tr1
S1
PC/Reg.
A1
A2
A3
00
5678

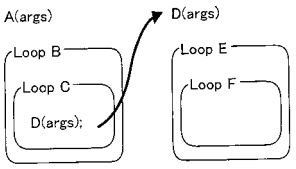
（ｂ）


PC/Reg.
Cache
Adr.
UP
Alt.
DN
00
01
02
03
04
05
06
07
FF
01
--00-----
5678----
01
03
05
E0
A1
FF
E0
A2
01
E0
A3
03
E0
05
Select Output

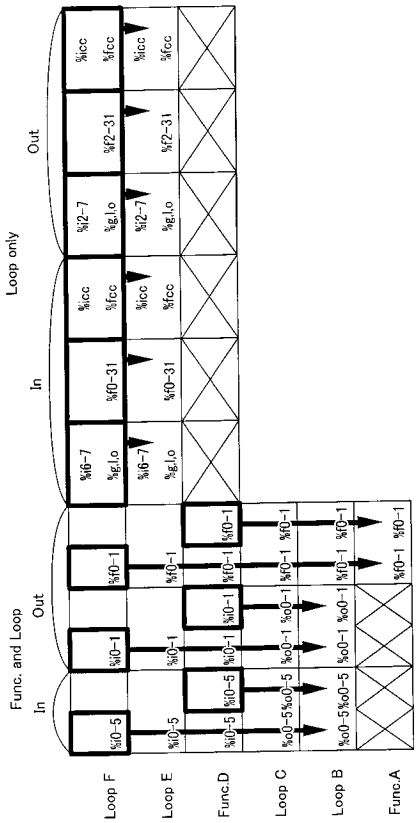
【図５】


(a)
PC/Reg.
A1
A2
A3
A4
00
01
5678
00000000
(b)
PC/Reg.
Cache
Store->A2
Select Output

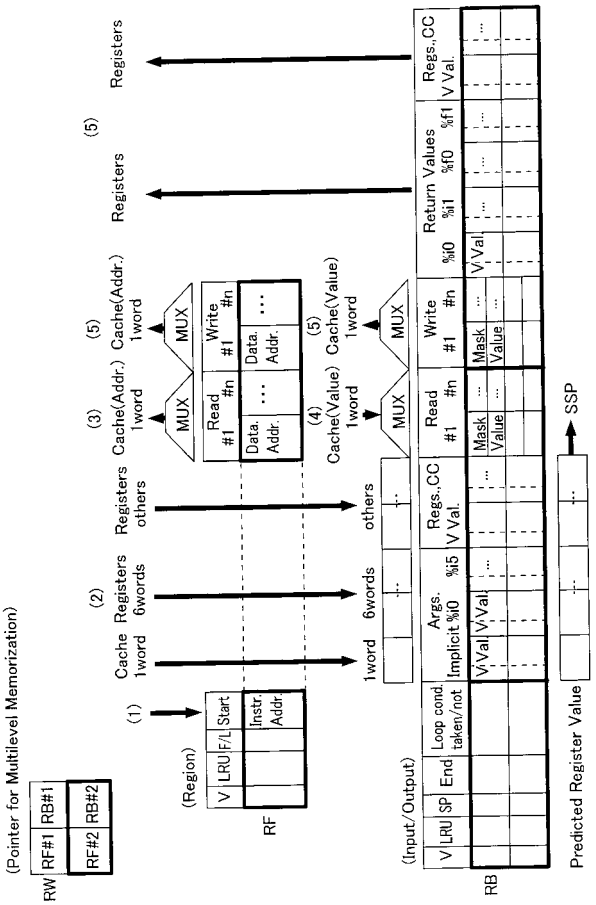
【図６】


A(args)
Loop B
Loop C
D(args);
D(args)
Loop E
Loop F

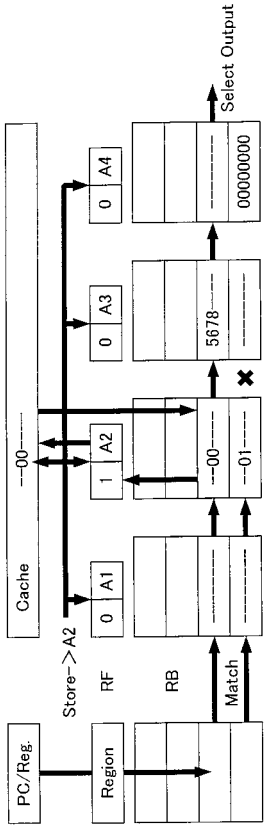
【図７】


Loop only
Out
In
Func. and Loop
Out
In
Loop F
Loop E
Func.D
Loop C
Loop B
Func.A

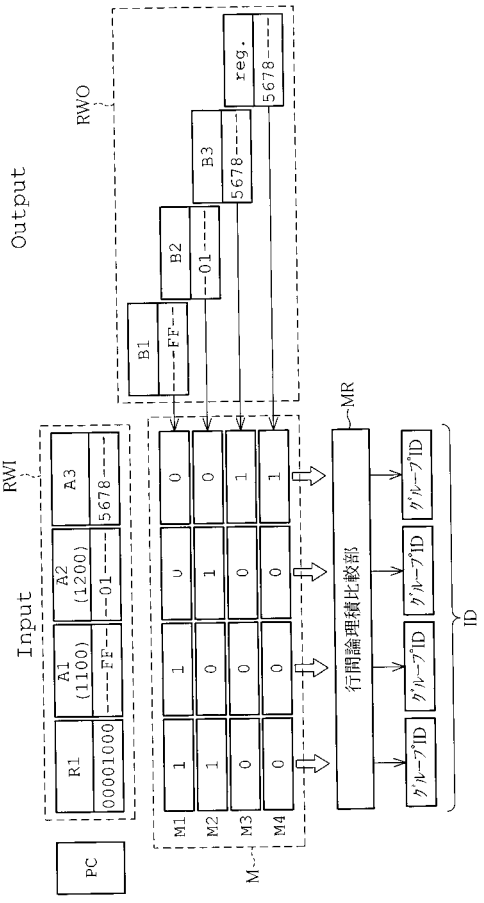
【図８】


(Pointer for Multilevel Memorization)
RW
RF#1
RB#1
RF#2
RB#2
(Region)
RF
(Input/Output)
RB
Registers
Cache(Addr.) 1word
Cache(Value) 1word
Cache 1word
Registers 6words
Registers others
Predicted Register Value
SSP

【図９】



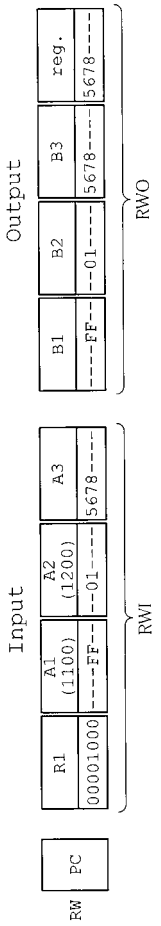
【図１０】



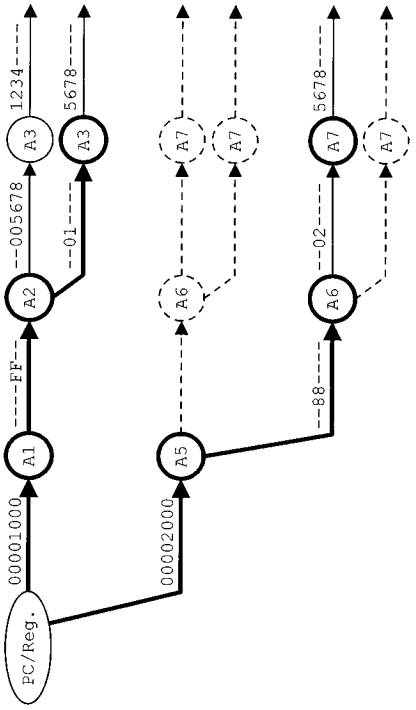
【図１１】

```
PC:load  [R1+100](----FF--)   -> reg.
   store reg.                 -> B1(----FF--)
   load  [R1+200](--01----)   -> reg.
   store reg.                 -> B2(--01----)
   load  A3(5678----)         -> reg.
   store reg.                 -> B3(5678----)
```

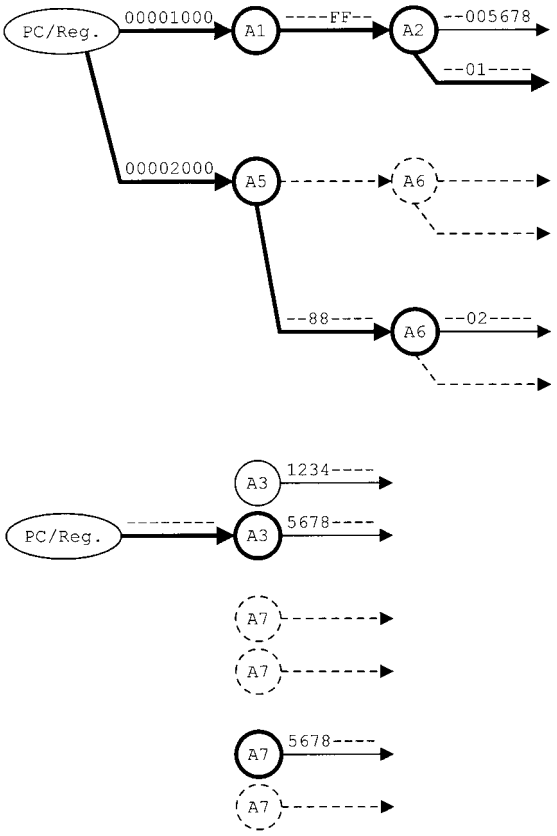
【図１２】



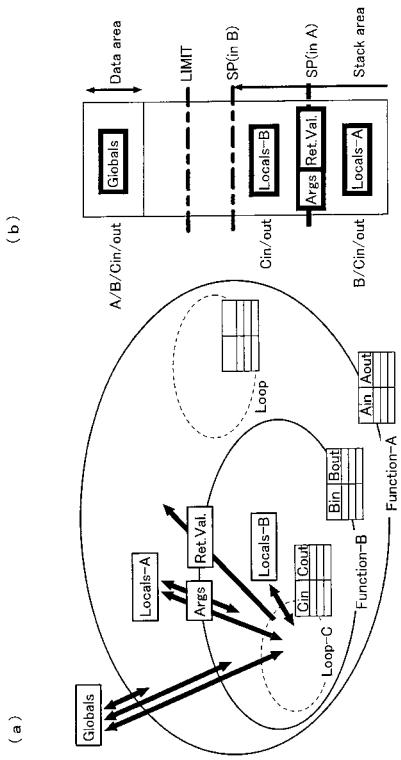
【図１３】



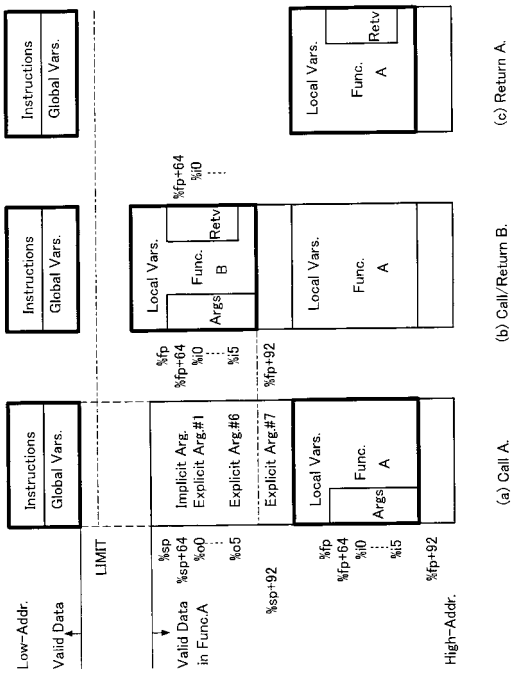
【図１４】



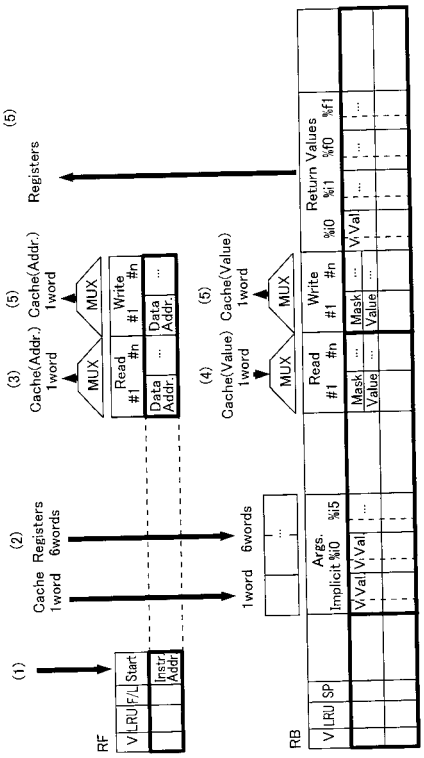
【図１５】



【図１６】



【図１７】


(1)
(2)
Cache Registers
1word 6words
(3)
Cache(Addr.)
1word
(4)
Cache(Value)
1word
(5)
Cache(Addr.)
1word
(5)
Cache(Value)
1word
(5)
Registers
MUX
Read #1 #n
Write #1 #n
Data Addr.
Mask
Value
Return Values %i0 %i1 %f0 %f1
Args. Implicit %i0 %i5
V Val
RF
RB
V LRU F/L Start
Instr Addr
V LRU SP

フロントページの続き

(56)参考文献 特開平１１－２１２７８８（ＪＰ，Ａ）
特開２００２－３１８６８８（ＪＰ，Ａ）
国際公開第９９／４５４６３（WO，Ａ１）
特開２００１－５６７８（ＪＰ，Ａ）
中島康彦 他，関数値再利用および並列事前実行による高速化技術，情報処理学会論文誌．ハイパフォーマンスコンピューティングシステム，社団法人情報処理学会，２００２年 9月，Vol.43，No.SIG_6(HPS_5)(20020915)，pp．1-12
中島康彦 他，動的命令解析に基づく多重再利用および並列事前実行，情報処理学会論文誌．コンピューティングシステム，社団法人情報処理学会，２００３年 7月，Vol.44，No.SIG_10(ACS_2)(20030715)，pp．1-16

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ 9／３４、9／３８、9／４０