

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2005-284683
(P2005-284683A)

(43) 公開日 平成17年10月13日(2005. 10. 13)

(51) Int. Cl.⁷
G06F 9/40

F I
G06F 9/40 310C

テーマコード(参考)
5B033

審査請求 未請求 請求項の数 11 O L (全 36 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)

(71) 出願人 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 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
Fターム(参考) 5B033 AA01 AA20 CA11 EA02

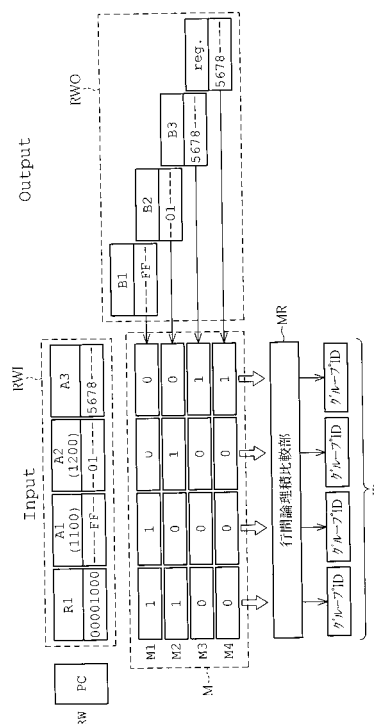
(54) 【発明の名称】 データ処理装置

(57) 【要約】 (修正有)

【課題】 関数の再利用を行う上でよりの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。

【解決手段】 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。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。

【選択図】 図10



【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、

10

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】

20

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。

30

【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、

40

(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

50

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。

10

【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。

20

【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。

30

【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、

40

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】

50

【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。

10

【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。

20

【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。

30

【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ため、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ため、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のと同時に、これに伴う消費電力の削減も実現される。

40

50

【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。

10

【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ことを明らかにし、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。

20

【0011】

大域変数 (Globals) は、関数Aの入出力 (A in / A out) および関数Bの入出力 (B in / B out) 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の戻り値 (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】

コンテキ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。

30

【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。

40

(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。

50

【 0 0 1 4 】

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。

【 0 0 1 5 】

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。

【 0 0 1 6 】

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%r0~5、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。

10

【 0 0 1 7 】

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。

20

【 0 0 1 8 】

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する。

30

【 0 0 1 9 】

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。

【 0 0 2 0 】

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、いかなるコンテキストであっても、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。

40

【 0 0 2 1 】

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。

【 0 0 2 2 】

50

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 (R F) および入出力記録表 (R B) 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、R F および R B 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。R F 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。R B 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。

10

【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R F が一括管理し、MaskおよびValueはR B が管理することにより、Readアドレスの内容とR B の複数エントリをC A M (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。

20

【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。

30

【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, H P S 5 , 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)

40

【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、R B 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、R B 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、R B 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

50

でよりの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。

10

【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。

20

【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の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。

30

【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。

40

【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

50

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。

10

【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て、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、各行成分の関係を容易に把握することが可能となる。

20

【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。

30

40

50

【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。

10

【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。

20

【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。

30

【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。

40

【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ため、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

50

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。

10

【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。

20

【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、今後入力が見込ま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ため、上記の構成による効果は高くなる。

30

【発明の効果】

40

【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の確か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。

50

【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。

10

【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。

20

【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。

30

【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。

40

【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】

50

(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に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。

10

20

30

40

50

【 0 0 6 9 】

また、R B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。

【 0 0 7 0 】

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。

【 0 0 7 1 】

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。

【 0 0 7 2 】

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、一致比較は行われず。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。

【 0 0 7 3 】

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。

【 0 0 7 4 】

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

10

20

30

40

50

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。

10

【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。

20

【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

30

【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。

40

【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

50

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。

10

【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。

20

【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。

30

【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。

40

【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

50

長のものとなる。例えばこのメモリ容量を 2 M b y t e とした場合、横が 8 w o r d、縦を 6 5 5 3 6 ラインとすることになる。

【 0 0 9 0 】

(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 R F 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、R F 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。

【 0 0 9 1 】

まず、図 4 (b) に、プログラムカウンタ (P C) 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、R B の「 0 1 」のラインにおいて、P C および Reg. が Value に登録されており、R F の「 0 1 」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「 A 1」、親ノード番号を示す UP が「 F F」となっている。また、R B の「 0 3 」のラインでは、Value 値なしで、Key が「 0 1」となっており、R F の「 0 3 」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「 A 2」、親ノード番号を示す UP が「 F F」となっている。以降、同様に、R B および R F における「 0 5 」のラインおよび「 0 7 」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。

10

20

【 0 0 9 2 】

この状態で、ある命令区間の実行が検出されると、P C および Reg. が R B に入力され、マッチラインとして、R B における「 0 1 」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 R B における番地である「 0 1 」が、エンコード結果として R F に伝達され、キー 0 1 に対応する R F におけるラインが参照される。キー 0 1 に対応する R F 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 A 1 について比較を行う必要はないことがわかる。

【 0 0 9 3 】

したがって、図 4 (a) の木構造に示すように、P C および Reg. の一致が S 1 において確認されると、T r 1 に示すノードのように、主記憶アドレス A 1、A 2、A 3 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。

30

【 0 0 9 4 】

R F および R B がこの状態である場合に、主記憶アドレス A 2 に対して書き込みが行われたとする。この場合、R F および R B における入力パターンの登録時には主記憶アドレス A 2 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 A 2 が変更されることによって、主記憶アドレス A 2 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 R F および R B が変更されることになる。

【 0 0 9 5 】

まず、内容が変更された主記憶アドレスである A 2 をキーにして、R F 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、R F における「 0 3 」のラインが選択される。そして、選択された「 0 3 」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。

40

【 0 0 9 6 】

次に、「 0 3 」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「 0 1 」のラインが認識される。そして、「 0 1 」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 A 2 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、「 0 1 」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「 0 3 」が書き込まれる。

50

【 0 0 9 7 】

以上のように R F および R B 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、P C および Reg. が R B に入力される。そして、R B において、連想検索により、入力されたこれらの値と、R B 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、R B における「0 1」のラインがマッチラインとして選択される。

【 0 0 9 8 】

次に、マッチラインとして選択されたラインの R B における番地である「0 1」が、エンコード結果として R F に伝達され、キー 0 1 に対応する R F におけるラインが参照される。キー 0 1 に対応する R F 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 A 1 となっている。すなわち、主記憶アドレス A 1 に関しては、一致比較を行う必要はないことがわかる。

10

【 0 0 9 9 】

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 A 2 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「0 3」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 A 2 の値が C a c h e 7 A を介して読み出され、R B 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「0 3」となっているラインが検索される。

【 0 1 0 0 】

図 5 (b) に示す例では、Key が「0 3」となっているラインは「0 4」および「0 5」の 2 つあるが、主記憶 3 から読み出された値が「0 0」であるので、「0 5」のラインがマッチラインとして選択され、R F に対して、エンコード結果としてキー 0 5 が伝達される。キー 0 5 に対応する R F 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、R O 1 および R O 2 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 6 A および主記憶 3 に対して出力される。

20

【 0 1 0 1 】

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、R F 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 A 1 の内容とキー 0 1 による検索をスキップして、主記憶アドレス A 2 の内容とキー 0 3 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。

30

【 0 1 0 2 】

(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 R F および R B 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、R O 1 および R O 2 が設けられている。

40

【 0 1 0 3 】

出力値は、R F および R B から出力されるアドレスに基づいて、出力値を記憶する R A M 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。

【 0 1 0 4 】

入力パターンに関しては、R F および R B において木構造によって登録されている。そ

50

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ト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ト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、ポイント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、ポイント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、ポイント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、順次ポイント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ト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。

10

20

30

40

50

そして、R Bにおいて、連想検索により、入力されたこれらの値と、R B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、R B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5 Aによる演算処理が行われる。そして、該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、R B、R F、R O 1、必要に応じてR O 2に登録される。ここで、R BおよびR F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、R Fの終端フラグを「E」とし、入力パターンの登録を終了する。

10

【0113】

一方、入力されたP CおよびReg.の値に一致するものが、R B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的一致比較が行われる。このようにして、R BおよびR Fに登録されている入力パターンと、該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R BおよびR F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、R F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R Fにおけるラインに対応する、R O 1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R O 1に登録しきれない場合には、ポインタを用いてR O 2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。

20

【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。

30

【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R W 4 A・4 Bによって生成され、生成された入出力セットに基づいて、R B、R F、R O 1、およびR O 2への上記したような登録処理が行われる。R W 4 A・4 B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R W 4 A・4 B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、R W 4 Aについて説明するが、R W 4 Bについても同様である。

【0117】

(R Wの第1構成例)

図12は、R W 4 Aのメモリ構成の概略を示す図である。同図に示すように、R W 4 Aは、命令区間のP C値を格納するP C、入力アドレスおよび入力値を格納するR W I、および、出力アドレスおよび出力値を格納するR W O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R W 4 A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。

40

【0118】

まず、ある命令区間の実行が開始されると、そのP C値がR W 4 AにおけるP C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】

50

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、RW0に登録されているか否かが検索される。RW0に登録されている場合には、既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ため、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、RW0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ため、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、RW0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、RW0に登録されているか否かが検索される。RW0に登録されている場合には、既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ため、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、RW0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RW0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RW0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RW0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RW0に登録されていないので、アドレスreg.および値(----FF--)がRW0に登録される。

【0130】

10

20

30

40

50

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。

10

【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。

20

【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。

30

【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、ノードに登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。

40

【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】

50

ここで、C A M / R A M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。

10

【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。

20

【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ため、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。

30

【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ため、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。

40

【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。

50

【 0 1 5 6 】

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。

【 0 1 5 7 】

(BW1)書き込みが行われたレジスタ/メモリのアドレスが、RW0に登録されているか否かが検索される。RW0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。

【 0 1 5 8 】

この時、RW0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、RW0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。

【 0 1 5 9 】

(BW2)書き込みが行われたレジスタ/メモリのアドレスが、RW0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RW0に登録する。

【 0 1 6 0 】

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、RW0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。

【 0 1 6 1 】

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。

【 0 1 6 2 】

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RW0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。

【 0 1 6 3 】

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。

【 0 1 6 4 】

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RW0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。

【 0 1 6 5 】

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。

【 0 1 6 6 】

また、この時点では、アドレスreg.はRW0に登録されていないので、アドレスreg.および値(----FF--)がRW0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 [1 1 0 0] に置き換えられる。その後、暫定行列 A (x) が初期化される。

【 0 1 6 7 】

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 B 1 に値 (----FF--) が書き込まれる。この時点では、アドレス *reg* . は R W O に登録されているので、R W O への登録は行われぬ。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) [1 1 0 0] が記憶される。

【 0 1 6 8 】

また、アドレス B 1 は R W O に登録されていないので、アドレス B 1 および値 (----FF-- 10
-) が R W O に登録される。

【 0 1 6 9 】

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 [1 1 0 0] に置き換えられる。その後、暫定行列 A (x) が初期化される。

【 0 1 7 0 】

次に、第 3 行目において、レジスタにおけるアドレス R 1 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 A 2 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 R 1 は R W I に既に登録されているので、R W I への登録は行われぬ 20
。

【 0 1 7 1 】

この時、アドレス R 1 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) [1 0 0 0] が記憶される。

【 0 1 7 2 】

また、アドレス A 2 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 A 2 は R W O および R W I のいずれにも登録されていないので、アドレス A 2 および値 (--01----) が R W I に登録される。

【 0 1 7 3 】

この時、アドレス A 2 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 30
、その他のメモリ要素を 0 とした暫定行列 A (2) [0 0 1 0] が記憶される。

【 0 1 7 4 】

また、この時点では、アドレス *reg* . は R W O に登録されており、この R W O 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 [1 0 1 0] に置き換えられる。その後、暫定行列 A (x) が初期化される。

【 0 1 7 5 】

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 B 2 に値 (--01----) が書き込まれる。この時点では、アドレス *reg* . は R W O に登録されているので、R W O への登録は行われぬ。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) [1 0 1 0] が記憶される。 40

【 0 1 7 6 】

また、アドレス B 2 は R W O に登録されていないので、アドレス B 2 および値 (--01----) が R W O に登録される。

【 0 1 7 7 】

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 [1 0 1 0] に置き換えられる。その後、暫定行列 A (x) が初期化される。 50

【 0 1 7 8 】

次に、第 5 行目において、アドレス A 3 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 A 3 は R W O および R W I のいずれにも登録されていないので、アドレス A 3 および値 (5678----) が R W I に登録される。

【 0 1 7 9 】

この時、アドレス A 3 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) [0 0 0 1] が記憶される。

【 0 1 8 0 】

また、この時点では、アドレス *reg* . は R W O に登録されており、この R W O 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 [0 0 0 1] に置き換えられる。その後、暫定行列 A (x) が初期化される。

【 0 1 8 1 】

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 B 3 に値 (5678----) が書き込まれる。この時点では、アドレス *reg* . は R W O に登録されているので、R W O への登録は行われぬ。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) [0 0 0 1] が記憶される。

【 0 1 8 2 】

また、アドレス B 3 は R W O に登録されていないので、アドレス B 3 および値 (5678----) が R W O に登録される。

【 0 1 8 3 】

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 [0 0 0 1] 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 1 0 に示す R W 4 A の入出力セットが生成される。

【 0 1 8 4 】

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。

【 0 1 8 5 】

(R s 1) 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。

【 0 1 8 6 】

(R s 2) ある行成分 M a 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 M a に対応する出力アドレス / 値である。

【 0 1 8 7 】

(R s 3) 「ある行成分 M a の反転」と「ある行成分 M b 」との論理積が全て 0 である場合、M a における 1 のパターンは、M b における 1 のパターンを包含する。すなわち、M a 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、M a に対応する出力アドレス / 値、および、M b に対応する出力アドレス / 値となる。

【 0 1 8 8 】

(R s 4) 「ある行成分 M a 」と「ある行成分 M b 」との論理積が全て 0 である場合、M a に属する入力アドレス / 値と、M b に属する入力アドレス / 値とは互いに独立している。

【 0 1 8 9 】

以上の情報に基づいて、R W 4 A 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 (R s 3) 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 M a の反転」と「ある行成分 M b 」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 M R 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。

【 0 1 9 0 】

次に、下位行成分が削除された状態において、上記の (R s 4) に関連する処理として、「ある行成分 M a 」と「ある行成分 M b 」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 M R 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。

10

【 0 1 9 1 】

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。

【 0 1 9 2 】

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。

20

【 0 1 9 3 】

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 M R が、各入出力グループにグループ ID を付与し、R W I 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 I D に格納する。これにより、グループ ID 格納部 I D 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。

30

【 0 1 9 4 】

以上のように、R W 4 A 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。

40

【 0 1 9 5 】

なお、本実施形態においては、R W 4 A 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はいるが、これに限定されるものではない。すなわち、R W 4 A 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 R W 4 A を適用することが可能である。

【 0 1 9 6 】

50

(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、S P A R C 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。S P A R C - A B I 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 R B、および R O 1 / R O 2 に登録されることになる。

10

【0197】

一方、S P A R C - A B I 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 R B 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。

20

【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。

30

【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。

40

【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 %l2 ~ 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

50

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。

10

【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。

20

【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。

30

【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)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のキャッシュが無効化される。

40

【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

50

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、SSP1BがN台の場合には、用意する入力予測値は、 $B + D \times 2$ から $B + D \times (N + 1)$ 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。

10

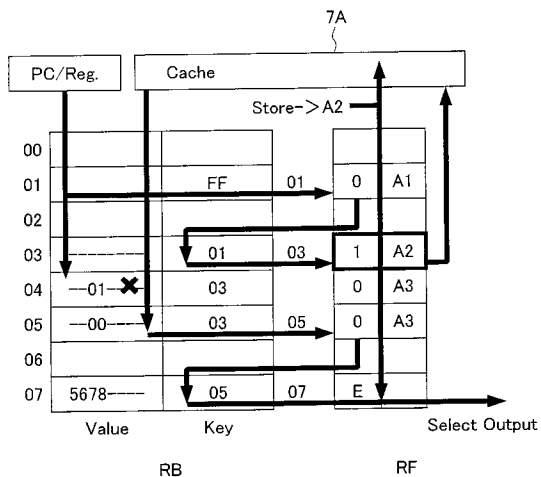
20

30

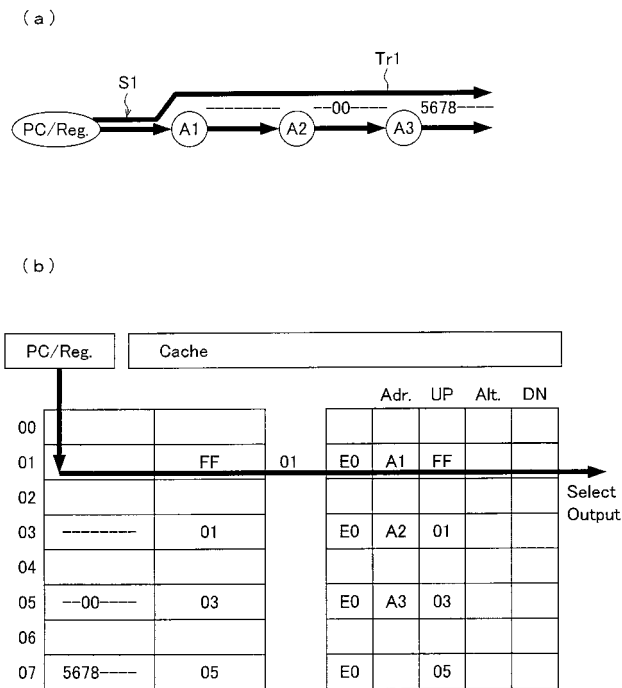
40

50

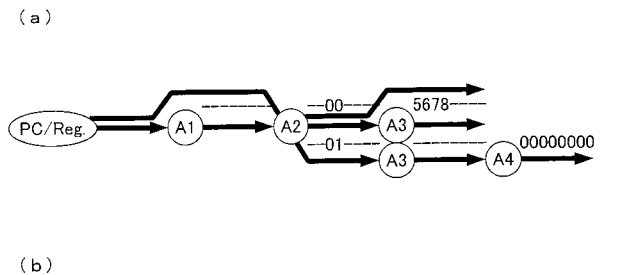
【 図 3 】



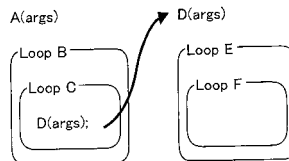
【 図 4 】



【 図 5 】



【 図 6 】



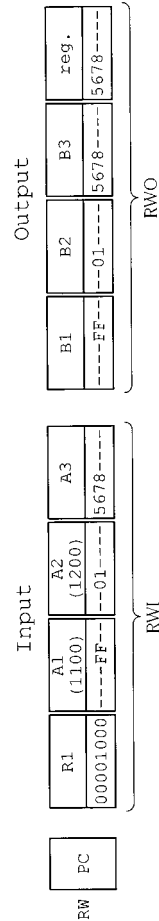
【 図 1 1 】

```

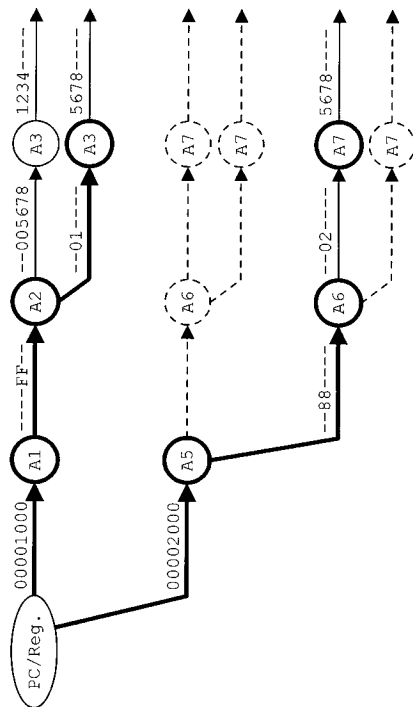
PC:load [R1+100] (----FF--> reg.
store reg.                -> B1(----FF-->
load [R1+200] (--01----> reg.
store reg.                -> B2(--01---->
load A3(5678----> reg.
store reg.                -> B3(5678---->

```

【 図 1 2 】



【 図 1 3 】



【 図 1 4 】

