

(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2006-155370

(P2006-155370A)

(43) 公開日 平成18年6月15日(2006.6.15)

| | | |
|----------------------------|----------------|-------------|
| (51) Int. Cl. | F I | テーマコード (参考) |
| G06F 9/38 (2006.01) | G06F 9/38 310A | 5B013 |
| G06F 9/40 (2006.01) | G06F 9/40 310A | 5B033 |

審査請求 未請求 請求項の数 8 O L (全 49 頁)

| | | | |
|-----------|------------------------------|----------|---|
| (21) 出願番号 | 特願2004-347124 (P2004-347124) | (71) 出願人 | 504132272 国立大学法人京都大学 京都府京都市左京区吉田本町36番地1 |
| (22) 出願日 | 平成16年11月30日(2004.11.30) | (74) 代理人 | 100080034 弁理士 原 謙三 |
| | | (74) 代理人 | 100113701 弁理士 木島 隆一 |
| | | (74) 代理人 | 100116241 弁理士 金子 一郎 |
| | | (72) 発明者 | 中島 康彦 京都府京都市左京区吉田本町 国立大学法人京都大学大学院経済学研究科内 |
| | | Fターム(参考) | 5B013 AA02 5B033 AA03 EA02 |

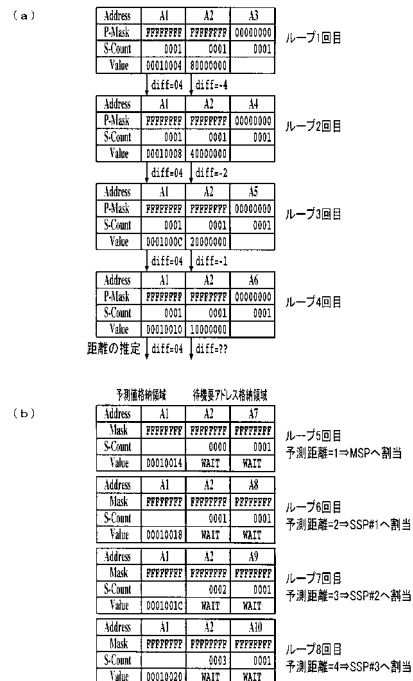
(54) 【発明の名称】 データ処理装置、データ処理プログラム、およびデータ処理プログラムを記録した記録媒体

(57) 【要約】

【課題】 主記憶手段から命令列および/または値を読み出し、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、予測的中率を向上させることによって、より効果的な命令区間の事前実行を実現するデータ処理装置を提供する。

【解決手段】 命令区間が実行された際の出力要素に対して、ストアの回数が記録されるとともに、このストアの回数に基づいて、予測入力アドレスに対してストアカウンタ(S-Count)が設定される。MSP/SSPは、予測処理部によって予測された入力要素に基づいて、該当する命令区間を事前実行するとともに、待機要アドレス格納領域に登録されている予測入力アドレスに関して、該アドレスに対応するストアカウンタ(S-Count)に基づいて、該当入力アドレスに対して行われるストアの回数を待機した上で主記憶からの読み出しを行って該当する命令区間の事前実行を行う。

【選択図】 図15



【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、複数の命令区間の実行結果としての入力パターンおよび出力パターンを記憶する入出力記憶手段とを備え、

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、上記入出力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入出力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とともに、

10

上記第 1 の演算手段による命令区間の実行結果を、上記入出力記憶手段に記憶する際に、入力パターンに含まれる入力要素のうち、予測を行うべき入力要素と予測を行う必要のない入力要素とを区別し、この区別情報を上記入出力記憶手段に登録するとともに、上記入出力記憶手段に格納される出力パターンにおける出力要素のうち、該当命令区間の実行の際にストアが行われたものについてそのストアの回数をカウントし、このカウント値を上記入出力記憶手段に格納する登録処理手段と、

上記区別情報に基づいて、上記入出力記憶手段に記憶されている入力要素のうち、予測を行うべき入力要素の値の変化の予測を行う予測処理手段と、

20

上記予測処理手段によって予測された入力要素に基づいて、該当する命令区間を事前実行するとともに、上記カウント値に基づいて該当入力要素に対して行われるストアの回数を待機した上で主記憶からの読み出しを行って該当する命令区間の事前実行を行う第 2 の演算手段とをさらに備え、

上記第 2 の演算手段による命令区間の事前実行結果が上記入出力記憶手段に記憶されることを特徴とするデータ処理装置。

【請求項 2】

上記入出力記憶手段が、上記第 1 の演算手段による命令区間の実行結果としての入力パターンおよび出力パターンを一時的に記録する入出力記録領域を備え、

上記入出力記録領域が、各出力要素に対して、ストアが行われた回数を格納するストアカウンタを有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 3】

上記入出力記憶手段が、上記第 1 の演算手段によって演算が行われた命令区間毎に過去の実行結果の履歴を格納する履歴格納領域を備え、

上記登録処理手段が、上記入出力記録領域に記録された実行結果を上記履歴格納領域に格納するとともに、上記入出力記録領域に記録された実行結果の入力パターンに含まれる入力要素のうち、履歴格納領域に前回の実行結果として登録されている出力要素と同じアドレスの入力要素に対して、対応する前回の出力要素のストアカウンタを該入力要素に対するストアカウンタとして登録することを特徴とする請求項 2 記載のデータ処理装置。

【請求項 4】

40

上記入出力記憶手段が、上記予測処理手段によって予測された入力要素を格納する予測値格納領域を備え、

上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴の間での値の変化量が一定である入力要素に関して値の予測を行い、上記予測値格納領域に格納することを特徴とする請求項 3 記載のデータ処理装置。

【請求項 5】

上記入出力記憶手段が、ストアの回数を待機した上で主記憶からの読み出しを行うべき入力要素を格納する待機要アドレス格納領域を備え、

上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴においてアドレスが変化せず、実行履歴の間での値の変化量が不定である入力要素に関して

50

、上記ストアカウンタ、および予測距離に基づく値としての待機カウンタを上記待機要アドレス格納領域に格納することを特徴とする請求項3記載のデータ処理装置。

【請求項6】

上記入出力記憶手段が、ストアの回数を待機した上で主記憶からの読み出しを行うべき入力要素を格納する待機要アドレス格納領域を備え、

上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴においてアドレス自体が変化し、それぞれのアドレスの値も、ストアが発生することにより変化する入力要素に関して、上記ストアカウンタに基づく値としての待機カウンタを上記待機要アドレス格納領域に格納することを特徴とする請求項3記載のデータ処理装置。

【請求項7】

請求項1ないし6のいずれか一項に記載のデータ処理装置が備える各手段が行う処理をコンピュータに実行させることを特徴とするデータ処理プログラム。

【請求項8】

請求項7に記載のデータ処理プログラムを記録したコンピュータ読取り可能な記録媒体。

10

【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列および/または値を読み出し、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。

20

【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。

【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。

30

【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、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。

40

【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

50

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のと同時に、これに伴う消費電力の削減も実現される。

10

【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。

20

【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

30

【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22(a)に示す。

【0011】

帯域変数(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。

40

【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 B_{in} / B_{out} のみを入出力として登録しなければならない。ここで、図22(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22(b)に示す。このメモリマップにおいて、 B_{in} / B_{out} を含まない領域はLocals-Bのみとなっている。よって、 B_{in} / B_{out} 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

50

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トの値(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。

10

(1)%sp以上の領域のうち、%sp+0~63はレジスタ退避領域、%sp+64~91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。

20

【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23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が構造体を戻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局所変数とを区別することができる。

30

【0017】

一方、(b)はB実行中の状態を示している。引数が入力、戻り値が出力、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する。

40

【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。

50

【 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。

【 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。

10

【 0 0 2 2 】

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24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。

【 0 0 2 3 】

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。

20

【 0 0 2 4 】

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。

30

【 0 0 2 5 】

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。

40

【 0 0 2 6 】

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。

【 0 0 2 7 】

以降、図24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

50

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。

【0028】

ここで、命令区間の一例として、図25に示す命令区間が、図24に示したRFおよびRBの構成によって実行され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。すなわち、命令区間の先頭が1000番地となっている。また、図26は、図25に示す命令区間が実行された場合に、RBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示しており、図27は、RBにおける実際の登録状況を示している。

【0029】

第1行目の命令(以降、単に第1の命令のように称する)において、アドレス定数A1がレジスタR0にセットされる。第2の命令において、レジスタR0の内容をアドレスとする主記憶からロードされた4バイトデータ(00110000)がレジスタR1に格納される。この場合、アドレスA1、マスク(FFFFFFFF)(マスクにおいて、Fが有効バイトを示しており、0が無効バイトを示す)、データ(00110000)は、入力としてRBにおけるInput側の第1列に登録され、レジスタ番号R1、マスク(FFFFFFFF)、およびデータ(00110000)は出力としてRBにおけるOutput側の第1列に登録される。

10

【0030】

第3の命令において、アドレス定数A2がレジスタR0にセットされる。第4の命令において、レジスタR0の内容をアドレスとする主記憶からロードされた1バイトデータ(02)がレジスタR2に格納される。この場合、アドレスA2、マスク(FF000000)、およびデータ(02)は入力としてRBにおけるInput側の第2列に登録される。この際、アドレスA2の残り3バイトについては、Don't Careを意味する「-」が格納される。レジスタ番号R2、マスク(FFFFFFFF)およびデータ(00000002)は出力としてRBにおけるOutput側の第2列に登録される。

20

【0031】

第5の命令において、アドレス(A2+R2)からロードされた1バイトデータ(22)がレジスタR2に格納されている。アドレスR2の値は(02)であったので、アドレス(A2+02)、およびデータ(22)が、入力としてRBにおけるInput側の第2列に追加登録される。この際、アドレス(A2+02)の部分に登録が行われ、アドレス(A2+01)および(A2+03)に対応する部分は、Don't Careを意味する「-」のままとなる。すなわち、アドレスA2に対応するマスクは(FF00FF00)となる。レジスタ番号R2、マスク(FFFFFFFF)、およびデータ(00000022)は、出力としてRBにおけるOutput側の第2列に上書きされる。

30

【0032】

第6の命令において、アドレス定数A3がレジスタR0にセットされる。第7の命令において、レジスタR0の内容をアドレスとする主記憶からロードされた1バイトデータ(33)がレジスタR3に格納される。この場合、アドレスA3、マスク(00FF0000)、およびデータ(33)は入力としてRBにおけるInput側の第3列に登録される。レジスタ番号R3、マスク(FFFFFFFF)、およびデータ(00000033)は出力としてRBにおけるOutput側の第3列に登録される。

40

【0033】

第8の命令において、アドレス(R1+R2)からロードされた1バイトデータ(44)がレジスタR4に格納される。この場合、アドレスR1とアドレスR2は命令区間の内部にて上書きされたレジスタのアドレスとなるので、アドレスR1およびアドレスR2は命令区間の入力とはならない。一方、アドレス(R1+R2)によって生成されたアドレスA4は命令区間の入力であるので、アドレスA4、マスク(00FF0000)、およびデータ(44)は入力としてRBにおけるInput側の第4列に登録される。レジスタ番号R4、マスク(FFFFFFFF)、およびデータ(00000044)は出力としてRBにおけるOutput側の第4列に登録される。

【0034】

50

第9の命令において、レジスタR5から値が読み出され、読み出された値に1が加えられた結果が再びレジスタR5に格納される。この場合、レジスタR5、マスク(FFFFFFFF)、およびデータ(00000100)は入力としてRBにおけるInput側の第5列に登録される。また、レジスタ番号R5、マスク(FFFFFFFF)、およびデータ(00000101)は出力としてRBにおけるOutput側の第5列に登録される。

【0035】

以上のように、命令実行時におけるメモリ/レジスタからの読み出しに際しては、以下の処理が行われる。

(1) RBにおけるOutput側が検索され、読み出されたアドレス/レジスタ番号が既登録であれば、該アドレス/レジスタ番号はInput側に登録されずに終了する。

10

(2) RBにおけるOutput側になればRBにおけるInput側が検索され、読み出されたアドレス/レジスタ番号が既登録であれば該アドレス/レジスタ番号は登録されずに終了する。

(3) RBにおけるInput側にもなければ、RBに新たにエントリが追加されて、該アドレス/レジスタ番号および値が登録される。

【0036】

また、命令実行時におけるメモリ/レジスタへの書き込みに際しては以下の処理が行われる。

(1) RBにおけるOutput側が検索され、読み出されたアドレス/レジスタ番号が既登録であれば値が更新されて終了する。

20

(2) RBにおけるOutput側になれば、新たにエントリが追加されて読み出されたアドレス/レジスタ番号および値が登録される。

【0037】

また、後述する特許文献1では、上記のような再利用を行う構成において、プロセッサを複数設け、並列事前実行を行う構成が開示されている。この並列事前実行が行われる際の入力の予測方法として、最後に出現した引数および最近出現した2組の引数の差分に基づいて、ストライド予測を行う方法が開示されている。

【0038】

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。

30

【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)

【特許文献1】特開2004-258905号公報(公開日2004年9月16日)

【発明の開示】

【発明が解決しようとする課題】

【0039】

図28は、図25に示す命令区間が繰り返し実行された場合における、RBの入力側に登録される履歴の例を示している。この例では、Timeが1~4まで変化するごとに命令区間が実行され、命令区間が実行される度に、アドレスA2の値は、(02)、(03)、(04)、(05)と変化しており、これに伴って他の入力要素における値が変化している。

40

【0040】

また、各履歴の間に示されるdiffは、対応する入力要素の値の変化量を示している。上記した従来の入力予測は、このdiffを用いて予測を行うことになる。図29は、この従来の入力予測による予測結果を示している。

【0041】

例えばループ制御変数のように、単調変化するアドレス(上記の例ではアドレスA2に対応)の内容については正確に予測することができている。しかしながら、命令区間に配列要素が含まれている場合、配列要素の添字が単調変化するにしても、配列要素値は一般に

50

単調変化するとは限らない。図 28 に示す例では、アドレス A 2 からロードした値が配列要素の添字に該当しており、この添字をアドレスとして用いる主記憶参照はアドレスが変化するために、履歴として登録される入力要素の数そのものが変化することになる。このような状況では、同一列の変化に規則性がなくなるために、図 29 におけるアドレス A 3 に対応する列に示すように、予測的中率が極めて悪化することになる。

【0042】

入力予測を行う際に、内容が変化しないアドレスに関する値の予測をすることはハードウェア資源の無駄となる。また、値の変化に規則性がない場合は、差分を 0 と仮定して予測するしかないが、無理に予測することにより、かえって的中率を下げることもある。図 29 に示す例では、A 2 + 4 に対応するアドレスについてはマスク位置そのものの変化を

10

【0043】

以上の課題はいずれも、登録された全てのアドレスを一律に扱ったことにより生じた問題である。

【0044】

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、主記憶手段から命令列および/または値を読み出し、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、予測的中率を向上させることによって、より効果的な命令区間の事前実行を実現するデータ処理装置、データ処理プログラム、およびデータ処理プログラムを記録した記録媒体を提供することにある。

20

【課題を解決するための手段】

【0045】

上記の課題を解決するために、本発明に係るデータ処理装置は、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、複数の命令区間の実行結果としての入力パターンおよび出力パターンを記憶する入出力記憶手段とを備え、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、上記入出力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入出力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とともに、上記第 1 の演算手段による命令区間の実行結果を、上記入出力記憶手段に記憶する際に、入力パターンに含まれる入力要素のうち、予測を行うべき入力要素と予測を行う必要のない入力要素とを区別し、この区別情報を上記入出力記憶手段に登録するとともに、上記入出力記憶手段に格納される出力パターンにおける出力要素のうち、該当命令区間の実行の際にストアが行われたものについてそのストアの回数をカウントし、このカウント値を上記入出力記憶手段に格納する登録処理手段と、上記区別情報に基づいて、上記入出力記憶手段に記憶されている入力要素のうち、予測を行うべき入力要素の値の変化の予測を行う予測処理手段と、上記予測処理手段によって予測された入力要素に基づいて、該当する命令区間を事前実行するとともに、上記カウント値に基づいて該当入力要素に対して行われるストアの回数を待機した上で主記憶からの読み出しを行って該当する命令区間の事前実行を行う第 2 の演算手段とをさらに備え、上記第 2 の演算手段による命令区間の事前実行結果が上記入出力記憶手段に記憶されることを特徴としている。

30

40

【0046】

上記の構成では、入出力記憶手段に、複数の命令区間の実行結果としての入力パターンおよび出力パターンが記憶されており、命令区間の実行時に、該命令区間の入力パターンと、入出力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に再利用を行う構成となっている。そして、予測処理手段によって、入出力記憶手段に記憶されている入力要素の今後の変化が予測され、この予測結果に基づいて、第 2 の演算手段が命令区間の事前

50

実行を行うようになっている。

【0047】

ここで、前記した従来技術のように、単純に入力要素の予測を行うと、予測の的中率が低くなることによって、予測による事前実行の効果が非常に低くなるという問題がある。これに対して、上記の構成によれば、まず登録処理手段が、入力パターンに含まれる入力要素のうち、予測を行うべき入力要素と予測を行う必要のない入力要素とを区別する。そして、予測処理手段は、区別処理手段によって予測を行うべき入力要素と判断された入力要素について予測を行うようになっている。したがって、予測の的中率を向上させることが可能となるので、より効果的な命令区間の事前実行を実現することが可能となる。

【0048】

また、登録処理手段は、入出力記憶手段に格納される出力パターンにおける出力要素のうち、該当命令区間の実行の際にストアが行われたものについてそのストアの回数をカウントし、このカウント値を入出力記憶手段に格納する。そして、予測処理手段は、上記カウント値に基づいて該当入力要素に対して行われるストアの回数を待機した上で主記憶からの読み出しを行って該当する命令区間の事前実行を行うようになっている。したがって、例えば値の変化が不定となる出力要素に関しては予測を行うことが困難であり、この場合に上記のようにカウントされたストアの回数を待機した上で主記憶読み出しが行われることによって、適切な入力要素の値を設定した状態で事前実行を行うことが可能となる。

【0049】

以上のような構成により、よりの確な事前実行を実現することが可能となる。このような事前実行が行われることによって、次に、同じ命令列が出現し、予測入力値と同じ入力が行われた場合には、入出力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可能性を一段と高めることができる。

【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、上記入出力記憶手段が、上記第1の演算手段による命令区間の実行結果としての入力パターンおよび出力パターンを一時的に記録する入出力記録領域を備え、上記入出力記録領域が、各出力要素に対して、ストアが行われた回数を格納するストアカウンタを有する構成としてもよい。

【0051】

上記の構成によれば、入出力記憶手段に入出力記録領域が設けられており、この入出力記録領域に、各出力要素に対して、ストアが行われた回数を格納するストアカウンタが設けられている。これにより、第1の演算手段によって命令区間の実行が行われた際に、該命令区間の実行に際して、各出力要素に対して行われたストアの回数を的確に記録することが可能となる。

【0052】

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、上記入出力記憶手段が、上記第1の演算手段によって演算が行われた命令区間毎に過去の実行結果の履歴を格納する履歴格納領域を備え、上記登録処理手段が、上記入出力記録領域に記録された実行結果を上記履歴格納領域に格納するとともに、上記入出力記録領域に記録された実行結果の入力パターンに含まれる入力要素のうち、履歴格納領域に前回の実行結果として登録されている出力要素と同じアドレスの入力要素に対して、対応する前回の出力要素のストアカウンタを該入力要素に対するストアカウンタとして登録する構成としてもよい。

【0053】

上記の構成によれば、まず入出力記録領域に記録された実行結果が順次命令区間毎に設けられた履歴格納領域に格納される。そして、入出力記録領域から履歴格納領域に格納される入力パターンに含まれる入力要素のうち、履歴格納領域に前回の実行結果として登録されている出力要素と同じアドレスの入力要素に対して、対応する前回の出力要素のストアカウンタが該入力要素に対するストアカウンタとして登録される。ここで、履歴格納領域に格納される入力要素のうち、前回の実行結果としての出力要素と同じアドレスとなる入力要素は、前回の実行結果に影響を受ける入力要素となる。すなわち、このような入力

10

20

30

40

50

要素に対して上記のようにストアカウンタを設定することによって、該当入力要素に対して予測を行う際に、待機すべきストアの回数を的確に設定することが可能となる。

【0054】

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、上記入出力記憶手段が、上記予測処理手段によって予測された入力要素を格納する予測値格納領域を備え、上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴の間での値の変化量が一定である入力要素に関して値の予測を行い、上記予測値格納領域に格納する構成としてもよい。

【0055】

上記の構成によれば、まず入出力記憶手段に予測値格納領域が設けられている。そして、予測処理手段が、実行履歴の間での値の変化量が一定である入力要素に関して値の予測を行い、予測値格納領域に格納する。ここで、履歴における命令区間の実行結果の間での値の変化量（差分）が一定である入力要素は、今後もその変化量が一定である可能性が高いものであるため、これに基づいて予測を行うことが可能である。このようにして予測を行った結果を予測値格納領域に格納することによって、予測中の可能性の高い予測値を設定することが可能となる。

10

【0056】

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、上記入出力記憶手段が、ストアの回数を待機した上で主記憶からの読み出しを行うべき入力要素を格納する待機要アドレス格納領域を備え、上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴においてアドレスが変化せず、実行履歴の間での値の変化量が不定である入力要素に関して、上記ストアカウンタ、および予測距離に基づく値としての待機カウンタを上記待機要アドレス格納領域に格納する構成としてもよい。

20

【0057】

上記の構成によれば、まず入出力記憶手段に待機要アドレス格納領域が設けられている。そして、予測処理手段が、実行履歴においてアドレスが変化せず、実行履歴の間での値の変化量が不定である入力要素に関して、上記ストアカウンタ、および予測距離に基づく値としての待機カウンタを待機要アドレス格納領域に格納する。ここで、予測距離とは、該当命令区間が今後繰り返し実行された場合の、現時点からの実行回数を示している。実行履歴においてアドレスが変化せず、実行履歴の間での値の変化量が不定である入力要素とは、該当アドレスに対して、命令区間が繰り返し実行される毎にストアが行われるものである。よって、待機カウンタを、上記のようにストアカウンタおよび予測距離に基づいて設定することによって、待機すべき回数を適切に設定することが可能となる。

30

【0058】

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、上記入出力記憶手段が、ストアの回数を待機した上で主記憶からの読み出しを行うべき入力要素を格納する待機要アドレス格納領域を備え、上記予測処理手段が、上記履歴格納領域に格納されている入力要素のうち、実行履歴においてアドレス自体が変化し、それぞれのアドレスの値も、ストアが発生することにより変化する入力要素に関して、上記ストアカウンタに基づく値としての待機カウンタを上記待機要アドレス格納領域に格納する構成としてもよい。

40

【0059】

上記の構成によれば、まず入出力記憶手段に待機要アドレス格納領域が設けられている。そして、予測処理手段が、実行履歴においてアドレス自体が変化し、それぞれのアドレスの値も、ストアが発生することにより変化する入力要素に関して、上記ストアカウンタに基づく値としての待機カウンタを上記待機要アドレス格納領域に格納する。実行履歴においてアドレス自体が変化し、それぞれのアドレスの値も、ストアが発生することにより変化する入力要素とは、命令区間が繰り返し実行される毎にアドレスが変化し、また、値の変化量も不定となるものである。よって、待機カウンタを、上記のようにストアカウンタのみに基づいて設定することによって、待機すべき回数を適切に設定することが可能となる。

50

【0060】

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、上記登録処理手段が、入力に用いられた上記レジスタの各アドレスに対して、スタックポインタまたはフレームポインタとして用いられる場合、および、該アドレスに対する書き込み命令が定数セット命令である場合に、該当アドレスに対して区別情報として定数フラグをセットし、上記以外の場合に、該当アドレスに対して上記定数フラグをリセットする構成としてもよい。

【0061】

上記の構成によれば、入力に用いられたレジスタのアドレスのうち、アドレスが固定しており、かつ、値が単調変化すると予測されるアドレスに定数フラグをセットすることが可能となる。よって、定数フラグがセットされているレジスタのアドレスに基づく入力要素に対して予測を行うようにすることによって、予測的中率を向上させることが可能となる。

10

【0062】

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、上記登録処理手段が、入力要素が新規に上記入出力記憶手段に記憶される際に、該入力要素のアドレスに対して、区別情報として変更フラグをリセットし、上記入出力記憶手段に記憶された後に、該当アドレスに対してストア命令が実行された場合に、該当アドレスに対して変更フラグをセットする構成としてもよい。

【0063】

上記の構成によれば、入出力記憶手段に記憶されたものの、その後一度も書き込みが行われないアドレスに対しては、変更フラグがリセットされた状態となる。このようなアドレスに記憶されている内容は変化していないことになるので、該アドレスに対して予測を行う必要はないことになる。すなわち、上記のような変更フラグが入力要素のアドレスに設けられることによって、予測が必要なアドレスのみに対して予測を行うことが可能となる。よって、予測処理のためのハードウェア資源を有効に利用することが可能となる。

20

【0064】

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、上記登録処理手段が、入力要素が新規に上記入出力記憶手段に記憶される際に、該入力要素のアドレスに対して、区別情報として履歴フラグをリセットし、該アドレスに対するロード命令実行時に、該アドレスを生成したレジスタアドレスに上記定数フラグがセットされている場合に、該アド

30

【0065】

上記の構成によれば、入出力記憶手段に記憶されている入力要素のアドレスに対するロード命令実行時に、該アドレスを生成したレジスタアドレスに上記定数フラグがセットされている場合に、該アドレスに対して履歴フラグがセットされるようになっている。ここで、定数フラグがセットされているレジスタアドレスとは、上記のように、アドレスが固定しており、かつ、値が単調変化すると予測されるアドレスとなっている。よって、このようなレジスタアドレスに基づいて生成されたアドレスに関して予測を行うことによる予測的中率は高くなることが予想される。すなわち、上記のような履歴フラグを設けることによって、予測すべきアドレスを適切に設定することが可能となる。

40

【0066】

なお、履歴フラグとしては、各アドレスに文字通りのフラグをたてるようにしてもよいし、複数のバイトデータからなるアドレスのうち、履歴保存対象とするバイト位置を示すマスクといった形式で履歴フラグを実現するようにしてもよい。

【0067】

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、上記登録処理手段が、入力要素が新規に上記入出力記憶手段に記憶される際に、該入力要素のアドレスに対して、区別情報として変更フラグをリセットし、上記入出力記憶手段に記憶された後に、該当アドレスに対してストア命令が実行された場合に、該当アドレスに対して変更フラグをセットするとともに、上記予測処理手段が、上記入出力記憶手段に記憶されている入力要素の

50

アドレスのうち、上記変更フラグがセットされ、かつ、履歴フラグがセットされているアドレスに関して、入力要素の変化の予測を行う構成としてもよい。

【0068】

ここで、変更フラグがセットされているアドレスとは、上記したように、予測を行うことによる効果が期待できるアドレスとなる。また、履歴フラグがセットされているアドレスとは、上記したように、予測的中率が高いことが期待できるアドレスとなる。したがって、上記の構成によれば、予測を行うことによる効果が高いと予想されるアドレスに関してのみ予測が行われることになる。よって、予測処理のためのハードウェア資源を有効に利用することが可能となる。

【0069】

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、上記予測処理手段が、上記入出力記憶手段に記憶されている入力要素のうち、該入力要素の履歴における値の変化量が0ではない入力要素のみに対して、入力要素の値の変化の予測を行う構成としてもよい。

【0070】

上記の構成によれば、履歴における値の変化量が0ではない入力要素のみに対して、入力要素の値の変化の予測が行われることになる。ここで、履歴における値の変化量が0となっている入力要素とは、変化がないことが予想される入力要素であるので、該入力要素に対して予測を行う必要はないことになる。すなわち、上記の構成によれば、予測が必要なアドレスのみに対して予測を行うことが可能となる。よって、予測処理のためのハードウェア資源を有効に利用することが可能となる。

【0071】

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、第2の演算手段が主記憶手段から値を読み出す際に、上記予測値格納領域において、ストアカウンタ値がセットされておらず、予測値が有効である場合に該予測値を読み出し値とし、ストアカウンタが0よりも大きい場合にはストアカウンタが0になるまで待機し、ストアカウンタが0になった時点で値を取り出す構成としてもよい。

【0072】

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、第2の演算手段が主記憶手段へ値を書き込む際に、他の第2の演算手段に対して書き込みアドレスおよび値を通知するとともに、該通知を受信した他の第2の演算手段は、予測値格納領域に同一アドレスが登録されている場合に、該入力要素のストアカウンタを1だけ減じて書き込み値を格納し、ストアカウンタが既に0である場合には何も行わない構成としてもよい。

【発明の効果】

【0073】

以上のように、本発明に係るデータ処理装置は、上記第1の演算手段による命令区間の実行結果を、上記入出力記憶手段に記憶する際に、入力パターンに含まれる入力要素のうち、予測を行うべき入力要素と予測を行う必要のない入力要素とを区別し、この区別情報を上記入出力記憶手段に登録するとともに、上記入出力記憶手段に格納される出力パターンにおける出力要素のうち、該当命令区間の実行の際にストアが行われたものについてそのストアの回数をカウントし、このカウント値を上記入出力記憶手段に格納する登録処理手段と、上記区別情報に基づいて、上記入出力記憶手段に記憶されている入力要素のうち、予測を行うべき入力要素の値の変化の予測を行う予測処理手段と、上記予測処理手段によって予測された入力要素に基づいて、該当する命令区間を事前実行するとともに、上記カウント値に基づいて該当入力要素に対して行われるストアの回数を待機した上で主記憶からの読み出しを行って該当する命令区間の事前実行を行う第2の演算手段とをさらに備え、上記第2の演算手段による命令区間の事前実行結果が上記入出力記憶手段に記憶される構成である。

【0074】

これにより、予測的中率を向上させることが可能となるとともに、適切な入力要素の

10

20

30

40

50

値を設定した状態で事前実行を行うことが可能となるので、より効果的な命令区間の事前実行を実現することが可能となる。このような事前実行が行われることによって、次に、同じ命令列が出現し、予測入力値と同じ入力が行われた場合には、命令列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という効果を奏する。

【発明を実施するための最良の形態】

【0075】

本発明の実施の一形態について図1ないし図21に基づいて説明すれば、以下のとおりである。

【0076】

(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に示す。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(入出力記憶手段)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。

【0077】

命令区間記憶部2は、プログラムにおける関数およびループを再利用するためのデータを格納するメモリ手段であり、RF、RB、RB登録処理部(登録処理手段)2A、および予測処理部(予測処理手段)2Bを備えた構成となっている。この命令区間記憶部2におけるRFおよびRBの詳細、ならびにRB登録処理部2Aおよび予測処理部2Bの詳細については後述する。

【0078】

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。

【0079】

MSP1Aは、RW(再利用記憶手段)4A、演算器(第1の演算手段)5A、レジスタ6A、Cache7A、および通信部9Aを備えた構成となっている。また、SSP1Bは、RW(再利用記憶手段)4B、演算器(第2の演算手段)5B、レジスタ6B、Cache/Local7B、判定部8B、および通信部9Bを備えた構成となっている。

【0080】

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およびRBの各エントリ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における特定のエントリをアクティブにする制御線の集合によって構成される。

【0081】

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。

【0082】

判定部8Bは、後述する事前実行の起動開始後の主記憶読み出しが行われる際に、RBにおける入出力記録行(後述)、予測値格納領域(後述)、待機用アドレス格納領域(後

10

20

30

40

50

述)、およびCache/Local7Bのうち、どこから値を読み出すかを判定するブロックである。この判定処理の詳細については後述する。この判定部8Bは、SSP1B内に設けられた小さなプロセッサによって実現される。

【0083】

通信部9A・9Bは、MSP1AまたはSSP1Bによって主記憶書き込みが行われる場合に、その旨をその他の全てのSSP1B...またはMSP1Aに対して通知するブロックである。この通信部9A・9Bは、MSP1AまたはSSP1B内に設けられた小さなプロセッサによって実現される。

【0084】

(RF/RBの構成)

図1は、本実施形態における命令区間記憶部2におけるRFおよびRBの構成の概要を示している。同図に示すように、RFは、複数のエントリを格納しており、各エントリに対して、該エントリが有効であるか否かを示すV、エントリ入れ替えのヒントを示すLRU、関数の先頭アドレスを示すStart、参照すべき主記憶アドレスを示すRead/Write、および、関数とループとを区別するF/Lを保持している。

【0085】

また、RBは、RFに格納されているエントリに対応して複数のエントリを格納しており、各エントリに対して、該エントリが有効であるか否かを示すV、エントリ入れ替えのヒントを示すLRU、関数またはループを呼び出す際の直前のスタックポイント%spを示すSP、引数(Args.)(V:有効エントリ、Val.:値)、主記憶値(C-FLAG:Readアドレスの変更フラグ、P-Mask:Readアドレスの履歴マスク、Mask:Read/Writeアドレスの有効バイト、Value:値、S-Count:Read/Writeアドレスのストアカウンタ)、返り値(Return Values)(V:有効エントリ、Val.:値)、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、および、引数や返り値以外のレジスタおよび条件コード(Regs.,CC)を保持している。また、RBは、1つ以上のレジスタアドレスに対応して定数フラグ(Const-FLAG)を格納するメモリ領域を保持している。なお、定数フラグ(Const-FLAG)の詳細については後述する。

【0086】

上記のRFおよびRBにおける各項目についてより詳細に説明する。上記Vは、上記のようにエントリが有効であるか否かを示すものであるが、具体的には、未登録時には「0」、登録中である場合には「2」、登録済である場合には「1」の値が格納されるようになっている。例えば、RFまたはRBを確保する際に、未登録エントリ(V=0)があれば、これを使用し、未登録エントリがなければ、登録済エントリ(V=1)の中からLRUが最小のものを選択して上書きすることになる。登録中エントリ(V=2)は使用中であるので上書きすることはできない。

【0087】

上記LRUは、一定時間間隔で右へシフトされていくシフトレジスタの中の「1」の個数を示したものである。RFの場合、このシフトレジスタは、該当エントリに関して、再利用のための登録を行ったか、もしくは再利用を試みた場合に、左端に「1」が書き込まれるようになっている。したがって、該当エントリが頻繁に使用されれば、LRUは大きな値となり、一定期間使用されなければ、LRUの値は0となる。一方、RBの場合、シフトレジスタには、該当エントリが再利用された場合に「1」が書き込まれるようになっている。したがって、該当エントリが頻繁に使用されれば、LRUは大きな値となり、一定期間使用されなければ、LRUの値は0となる。

【0088】

上記RBにおける主記憶値のMaskについて説明する。一般に、アドレスとデータとを1バイトずつ管理することにすれば管理が可能であるが、実際には、4バイト単位でデータを管理の方がキャッシュ参照を高速に行うことができる。そこで、RFでは、主記憶アドレスを4の倍数で記憶するようになっている。一方、管理単位を4バイトとする場合、1バイト分だけをロードすることに対応できるようにするために、4バイトのうちでどの

10

20

30

40

50

バイトが有効であることを示す必要がある。すなわち、Maskは、4バイトのうちでどのバイトが有効であることを示す4ビットのデータとなっている。例えば、C001番地から1バイト分をロードした結果、値がE8であった場合、R Fには、アドレスC000が登録され、R BのMaskに「0100」、Valueに「00E80000」が登録されることになる。なお、Readアドレスにおける変更フラグ(C-FLAG)および履歴マスク(P-Mask)、ならびにRead/Writeアドレスにおけるストアカウンタ(S-Count)の詳細については後述する。

【0089】

上記の引数や返り値以外のレジスタおよび条件コード(Regs.,CC)について説明する。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、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、になり、入力は、arg[0-5]、出力は、rti[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R Bにおける引数(Args.)の項で十分である。

10

【0090】

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、R BにおけるRegs.,CCには、%g0-7、%o0-7、%l0-7、%i0-7、%f0-31、ICC、FCCが登録されるようになっている。

20

【0091】

以上のように、命令区間記憶部2において、ReadアドレスはR Fが一括管理し、MaskおよびValueはR Bが管理している。これにより、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)によって一度に比較する構成を可能としている。このことについて、以下により詳しく説明する。

【0092】

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、R A Mと呼ばれるメモリである。一方、上記のC A M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、そのエントリに対応する信号がO Nとなるように動作するようになっている。通常は、C A MはR A Mとセットにして用

30

【0093】

ここで、C A MとR A Mとの連携動作について、具体例を挙げて説明する。C A M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、R A Mに、C A M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C A Mに入力すると、一致するエントリがO Nとなり、R A M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R Bが実現されることになる。

40

【0094】

なお、図2に示すように、本実施形態におけるR Bには、入出力記録行(入出力記録領域)、区間毎情報として履歴格納行(履歴格納領域)、予測値格納領域、および待機要アドレス格納領域、ならびに予測実行結果記録行が設けられている。これらの入出力記録行、履歴格納行、予測値格納領域、および待機要アドレス格納領域、ならびに予測実行結果記録行は、図1に示すR Bにおけるエントリにほぼ準じた形式で実現されるが、それぞれ格納形式が若干異なっている。これらの格納形式の詳細については後述する。

【0095】

(再利用処理の概略)

次に、関数およびループのそれぞれの場合について、再利用処理の概略について説明す

50

る。

【0096】

まず、関数の場合について説明する。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、登録中のエントリを有効にする。

【0097】

以降、図1を参照しながら説明すると、関数を呼び出す前に、(1)RFに登録されているエントリにおける関数の先頭アドレスに、該当関数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関数に関するエントリにおける引数が、呼び出す関数の引数と完全に一致するエントリを選択する。そして、(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RFからすべて参照して、(4)RBに登録されている内容と一致比較を行う。全ての入力一致した場合に、(5)RBに登録済の出力(返り値、大域変数、およびAの局所変数)を主記憶3に書き戻すことによって、関数の実行を省略する、すなわち関数の再利用を実現することができる。

10

【0098】

次に、ループの場合について説明する。ループが完了する以前に関数から復帰したり、前記した擾乱が発生したりするなど、ループの入出力登録が中止されなければ、登録中のループに対応する後方分岐命令を検出した時点で、登録中の入出力表エントリを有効にし、そのループの登録を完了する。

20

【0099】

さらに、後方分岐命令が成立する場合は、次のループが再利用可能かどうかを判断する。すなわち、図1を参照しながら説明すると、後方分岐する前に、(1)RFに登録されているエントリにおけるループの先頭アドレスに、該当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ループに関するレジスタ入力値が、呼び出すループのレジスタ入力値と完全に一致するエントリを選択する。そして、(3)関連する主記憶アドレスをRFから全て参照して、(4)RBに登録されている内容と一致比較を行う。全ての入力一致した場合に、(5)RBに登録済の出力(レジスタおよび主記憶出力値)を主記憶3に書き戻すことによってループの実行を省略する、すなわちループの再利用を実現することができる。

30

【0100】

再利用した場合、RBに登録されている分岐方向に基づいて、さらに次のループに関して同様の処理を繰り返す。一方、次のループが再利用不可能であれば、次のループを通常に実行し、RFおよびRBへの登録を開始する。

【0101】

(命令区間の実行時における処理の流れ)

次に、命令がデコードされた場合の具体的な処理の流れについて説明する。以下では、命令がデコードされた結果、関数呼び出し命令である場合、関数復帰命令である場合、後方分岐成立の場合、後方分岐不成立の場合、およびその他の命令の場合について、それぞれ処理の流れを説明する。

40

【0102】

(関数呼び出し命令である場合)

命令がデコードされた結果、関数呼び出し命令である場合の処理を図3に示すフローチャートを参照しながら以下に説明する。まずステップ1(以降、S1のように称する)において、引数の第7ワードを検出したか否かが判定される。S1においてYES、すなわち、引数の第7ワードを検出したと判定された場合には、RWに登録されている登録中RBエントリを全て無効化し、S6に移行して、プログラムカウンタを関数の先頭へ進め、処理を終了する。

【0103】

50

一方、S 1においてNO、すなわち、引数の第7ワードを検出していないと判定された場合には、該関数呼び出しおよび入力値がRFおよびRBに登録されているか否かを検索する(S 2)。S 2においてYES、すなわち、該関数呼び出しおよび入力値がRFおよびRBに登録されていると判定された場合には、後述するS 7のステップに移行する。

【0104】

S 2においてNO、すなわち、該関数呼び出しおよび入力値がRFおよびRBに登録されていないと判定された場合、該関数のためのRFエントリおよびRBエントリを確保しようと試み、(1)既存のRFエントリがあるか、(2)登録作業中につき追い出すことのできないRFエントリ以外に、使用可能なRFエントリがあるか、または(3)登録作業中につき追い出すことのできないRBエントリ以外に、使用可能なRBエントリがあるかを判定する(S 3)。

【0105】

S 3においてNO、すなわち、使用可能なRF・RBエントリがないと判定された場合には、登録を開始せず、RWに登録されているRBを全て無効化し(S 5)、RWを空にする。一方、S 3においてYES、すなわち、使用可能なRF・RBエントリがあると判定された場合には、該関数のためのRFエントリおよびRBエントリを確保し、RWに登録する(S 4)。ここで、RWに登録した際に、RWに登録されているRWエントリが溢れた際には、最も古いRWエントリを削除し、対応するRBを無効化する。S 3またはS 4が行われた後に、プログラムカウンタを関数の先頭へ進め(S 6)、処理を終了する。

【0106】

一方、S 2においてYES、すなわち、該関数呼び出しおよび入力値がRFおよびRBに登録されていると判定された場合、該関数は再利用可能であることになる。すなわち、RBから出力値を求めるとともに、レジスタおよび主記憶3にこの出力値を書き込む(S 7)。そして、登録中の関数/ループがRWに登録されているか否かを判定し(S 8)、登録されている場合には、再利用を行った関数のRBエントリの内容のうち必要なものをRWに登録されているエントリに追加する(S 9)。ここで、RWのTOPから順に登録し、途中でRBがあふれた場合には、以降、RWのBOTTOMまでに対するRBを無効化し、RWから削除する。その後、プログラムカウンタを次の命令へ進め(S 10)、処理を終了する。

【0107】

(関数復帰命令である場合)

命令がデコードされた結果、関数復帰命令である場合の処理を図4に示すフローチャートを参照しながら以下に説明する。S 11において、RWのTOPから順にたどり、関数に対応するRF/RBが検出されるまでに、ループに関するRBが検出されるか否かが判定される(S 12)。ここでループに関するRBが検出されると(S 12においてYES)、該当RBを全て無効化するとともに、RWから削除する(S 13)。

【0108】

一方、RW探索中に、該関数に対応するRF/RBを検出したか否かが判定される(S 14)。ここで、該関数に対応するRF/RBが検出されると(S 14においてYES)、該当RBエントリを有効化するとともに、RWから削除する(S 15)。

【0109】

その後、復帰命令を実行し(S 16)、処理を終了する。

【0110】

(後方分岐成立である場合)

命令がデコードされた結果、後方分岐成立である場合の処理を図5に示すフローチャートを参照しながら以下に説明する。まず、RWのTOPから順にたどり、関数に対応するRBを検出するか否かが判定される(S 21)。S 21においてYES、すなわち、関数に対応するRBを検出した場合には、後述するS 24のステップに移行する。

【0111】

一方、S 21においてNO、すなわち、関数に対応するRBを検出しない場合には、次

10

20

30

40

50

に、該後方分岐命令自身のアドレスとRB中のループ終了アドレスとが一致するか否かが判定される(S22)。S22においてNO、すなわち、該後方分岐命令自身のアドレスとRB中のループ終了アドレスとが一致しないと判定されると、後述するS24のステップに移行する。

【0112】

S22においてYES、すなわち、該後方分岐命令自身のアドレスとRB中のループ終了アドレスとが一致すると判定された場合、RWのTOPから該RBの手前までのRBを全て無効化し(S23)、RWから削除する。また、該RBエントリを有効化し、かつtaken=1とし、RWから削除する。

【0113】

次に、S24において、次ループの先頭アドレスおよび入力値がRFおよびRBに登録されているか否かが判定される。S24においてYES、すなわち、次ループの先頭アドレスおよび入力値がRFおよびRBに登録されている場合には、後述するS30のステップに移行する。

【0114】

一方、S24においてNO、すなわち、次ループの先頭アドレスおよび入力値がRFおよびRBに登録されていない場合には、次ループのためのRFエントリおよびRBエントリを確保しようと試み、(1)既存のRFエントリがあるか、(2)登録作業中につき追い出すことができないRFエントリ以外に、使用可能なRFエントリがあるか、または(3)登録作業中につき追い出すことができないRBエントリ以外に、使用可能なRBエントリがあるかが判定される(S25)。

【0115】

S25においてNO、すなわち、使用可能なRF・RBエントリがないと判定された場合には、登録を開始せずに、RWに登録されているRBを全て無効化し(S26)、RWを空にする。その後、S29において、プログラムカウンタを条件分岐先へ進め、処理を終了する。

【0116】

一方、S25においてYES、すなわち、使用可能なRF・RBエントリがあると判定された場合には、その使用可能なRF・RBエントリを確保し、確保したRF・RBをRWに登録する(S27)。また、RBにループ終了アドレス(後方分岐命令自身のアドレス)に登録する。ここで、RWへの登録を行った際にRWが溢れた場合には、最も古いRWエントリを削除し(S28)、それに対応するRBを無効化する。その後、S29において、プログラムカウンタを条件分岐先へ進め、処理を終了する。

【0117】

一方、前記したS24においてYESとなった場合、次ループは再利用可能であることになるので、RBから出力値を求め、この値をレジスタおよび主記憶3に書き込む(S30)。ここで、登録中の関数/ループがRWに登録されているか否かが判定され(S31)、登録されている場合、再利用を行ったループのRBエントリの内容のうち必要なものをRWに登録されているエントリに追加する(S32)。このとき、RWのTOPから順に登録し、途中でRBが溢れた場合、以降、RWのBOTTOMまでに対するRBを無効化し、RWから削除する。

【0118】

その後、プログラムカウンタは、次ループ先頭ではなく、該RB中のtakenの値に応じて、taken=1の場合は自命令、taken=0の場合は、RB中に記憶しておいたループ終了アドレスの次へ進める。その後、処理を終了する。

【0119】

(後方分岐不成立である場合)

命令がデコードされた結果、後方分岐不成立である場合の処理を図6に示すフローチャートを参照しながら以下に説明する。まず、RWのTOPから順に検索し(S41)、関数に対応するRBを検出したか否かが判定される(S42)。S42においてYES、す

10

20

30

40

50

なわち、関数に対応する R B を検出したと判定された場合、S 4 6 においてプログラムカウンタを次命令に進め、処理を終了する。

【 0 1 2 0 】

S 4 2 において N O、すなわち、関数に対応する R B を検出していないと判定された場合、該後方分岐命令自身のアドレスと R B 中のループ終了アドレスが一致するか否かが判定される (S 4 3)。S 4 3 において N O、すなわち、該後方分岐命令に対応する R F / R B を検出していないと判定された場合、S 4 6 においてプログラムカウンタを次命令に進め、処理を終了する。

【 0 1 2 1 】

一方、S 4 3 において Y E S、すなわち、該後方分岐命令に対応する R F / R B を検出したと判定された場合、R W の T O P から該 R B の手前までの R B を全て無効化し (S 4 4)、R W から削除する。また、該 R B エントリを有効化し、かつ taken=0 とし、R W から削除する (S 4 5)。その後、S 4 6 においてプログラムカウンタを次命令に進め、処理を終了する。

【 0 1 2 2 】

(その他の命令である場合)

次に、命令がデコードされた結果、上記以外のその他の命令である場合について説明する。その他の命令である場合、レジスタ R / W、主記憶 R / W が実行される。その際に、R W が空でなければ、以下の手順によってレジスタ R / W、主記憶 R / W を R W に登録されている R B に対して登録する。以下では、(1) 汎用レジスタ R E A D の場合、(2) 汎用レジスタ W R I T E の場合、(3) 浮動小数点レジスタ R E A D の場合、(4) 浮動小数点レジスタ W R I T E の場合、(5) 条件コードレジスタ I C C - R E A D の場合、(6) 条件コードレジスタ I C C - W R I T E の場合、(7) 浮動小数点条件コードレジスタ F C C - R E A D の場合、(8) 浮動小数点条件コードレジスタ F C C - W R I T E の場合、(9) 主記憶 R E A D の場合、(1 0) 主記憶 W R I T E の場合についてそれぞれ説明する。

【 0 1 2 3 】

(1) 汎用レジスタ R E A D の場合

まず、R W の T O P から B O T T O M まで順にたどる。そして、(1 - 1) 該 R B がループ関数かつ %o0-6 の場合、または該 R B が非ループ関数かつ %i0-6 の場合、arg[0-5].V=0 であれば、arg[0-5].V=1 に変更し、arg[0-5].Val に読み出しデータを記録する。その後、さらに R W をたどり、該 R B が関数の場合、処理を終了する。一方、該 R B が関数ではない (ループである) 場合、arg[0-5].V=0 であれば、arg[0-5].V=1 に変更し、arg[0-5].Val に読み出しデータを記録し、処理を終了する。

【 0 1 2 4 】

一方、(1 - 2) 該 R B がループの場合、(a) %g0-7 で grr[0-7].V=0 であれば、grr[0-7].V=1 に変更し、grr[0-7].Val に読み出しデータを記録し、処理を終了する。(b) %o0-7 で arg[0-7].V=0 であれば、arg[0-7].V=1 に変更し、arg[0-7].Val に読み出しデータを記録し、処理を終了する。(c) %l0-7 で lrr[0-7].V=0 であれば、lrr[0-7].V=1 に変更し、lrr[0-7].Val に読み出しデータを記録し、処理を終了する。(d) %i0-7 で irr[0-7].V=0 であれば、irr[0-7].V=1 に変更し、irr[0-7].Val に読み出しデータを記録し、次の R W エントリに進む。

【 0 1 2 5 】

(2) 汎用レジスタ W R I T E の場合

まず、R W の T O P から B O T T O M まで順にたどる。そして (2 - 1) 該 R B がループ関数かつ %o0-5 の場合、または該 R B が非ループ関数かつ %i0-5 の場合、arg[0-5].V=0 であれば、以降の読み出しは入力ではないことを示すために、arg[0-5].V=2 に変更する。さらに、%o0-1/%i0-1 について、rti[0-1].V=1 に変更し、rti[0-1].Val に書き込みデータを記録する。その後、さらに R W をたどり、該 R B が関数の場合、処理を終了する。一方、該 R B が関数ではない (ループである) 場合、arg[0-1].V=0 であれば、以降の読み出しは

入力ではないことを示すために、arg[0-1].V=2に変更し、rti[0-1].V=1に変更し、rti[0-1].Valに書き込みデータを記録し、処理を終了する。

【0126】

一方、(2-2)該RBがループの場合、(a)%g0-7でgrr[0-7].V=0であれば、grr[0-7].V=2に変更し、grr[0-7].Valに書き込みデータを記録し、処理を終了する。(b)%o0-7でarg[0-7].V=0であれば、arg[0-7].V=2に変更し、arg[0-7].Valに書き込みデータを記録し、処理を終了する。(c)%l0-7でlrr[0-7].V=0であれば、lrr[0-7].V=2に変更し、lrr[0-7].Valに書き込みデータを記録し、処理を終了する。(d)%i0-7でirr[0-7].V=0であれば、irr[0-7].V=2に変更し、irr[0-7].Valに書き込みデータを記録し、次のRWエントリに進む。

10

【0127】

(3)浮動小数点レジスタREADの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(3-1)該RBが関数の場合、何もせずに処理を終了する。一方、(3-2)該RBがループの場合、frr[0-31].V=0であれば、frr[0-31].V=1に変更し、frr[0-31].Valに読み出しデータを記録し、処理を終了する。

【0128】

(4)浮動小数点レジスタWRITEの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(4-1)該RBが関数かつ%f0-1の場合、rtf[0-1].V=1に変更し、rtf[0-1].Valに書き込みデータを記録する。さらにRWをたどり、frr[0-1].V=0であれば、以降の読み出しは入力ではないことを示すために、frr[0-1].V=2に変更し、rtf[0-1].V=1に変更し、rtf[0-1].Valに書き込みデータを記録し、処理を終了する。

20

【0129】

一方、(4-2)該RBがループの場合、frr[0-31].V=0であれば、frr[0-31].V=2に変更し、frw[0-31].V=1に変更し、frw[0-7].Valに書き込みデータを記録し、処理を終了する。

【0130】

(5)条件コードレジスタICC-READの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(5-1)該RBが関数の場合、何もせずに処理を終了する。一方、(5-2)該RBがループの場合、icr.V=0であれば、icr.V=1に変更し、icr.Valに読み出しデータを記録し、処理を終了する。

30

【0131】

(6)条件コードレジスタICC-WRITEの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(6-1)該RBが関数の場合、何もせずに処理を終了する。一方、(6-2)該RBがループの場合、icr.V=0であれば、icr.V=2、icw.V=1に変更し、icw.Valに書き込みデータを記録し、処理を終了する。

【0132】

(7)浮動小数点条件コードレジスタFCC-READの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(7-1)該RBが関数の場合、何もせずに処理を終了する。一方、(7-2)該RBがループの場合、fcr.V=0であれば、fcr.V=1に変更し、fcr.Valに読み出しデータを記録し、処理を終了する。

40

【0133】

(8)条件コードレジスタICC-WRITEの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(8-1)該RBが関数の場合、何もせずに処理を終了する。一方、(8-2)該RBがループの場合、fcr.V=0であれば、fcr.V=2、fcw.V=1に変更し、fcw.Valに書き込みデータを記録し、処理を終了する。

【0134】

50

(9) 主記憶 R E A D の場合

まず、R W の T O P から B O T T O M まで順にたどる。そして、R B に W R I T E データとして登録済である場合は、その値を使用する。一方、上記の場合ではなく、R B に R E A D データとして登録済である場合には、その値を使用する。さらに、いずれにも登録済でない場合は、キャッシュを経由して主記憶 3 から読み込む。

【 0 1 3 5 】

その後、再度 R W の T O P から B O T T O M まで順にたどる。そして、(a) アドレスが、R B に登録されている $sp+64$ の場合、構造体ポインタの読み出しであるので、 $arg0.V=0$ であれば、 $arg0.V=1$ に変更し、 $arg0.Val$ に読み出しデータを記録する。(b) 上記の (a) の場合でなく、アドレスが、LIMIT 以上 $sp+92$ 未満であれば、登録不要領域であるので、何もしない。(c) 上記の (b) の場合でない場合、W R I T E データとして登録済であるかどうかを検査し、そうであれば、すでに上書きされたあとの R E A D であるので登録不要であり、何もしない。(d) 上記の (c) でない場合、R E A D データとして登録済であるかどうかを検査し、そうであれば、すでに登録済であるので登録不要であり、何もしない。(e) 上記の (d) でない場合、R E A D データとしての登録が必要であるので、R F に主記憶 R E A D アドレスを確保し、R E A D データとして登録する。R F に主記憶アドレスを確保できなかった場合には、登録不能であるため、その R W エントリから B O T T O M までに対応する R B エントリを全て無効化する。

10

【 0 1 3 6 】

(1 0) 主記憶 W R I T E の場合

まず、キャッシュを経由して、主記憶 3 に書き込む。そして、ベースレジスタが $14(\%sp)$ かつオフセットが 92 以上である場合、引数の第 7 ワードを検出したことを記憶する。

20

【 0 1 3 7 】

その後、R W の T O P から B O T T O M まで順にたどる。そして、(a) アドレスが、R B に登録されている $sp+64$ の場合、構造体ポインタの読み出しであるので、 $arg0.V=0$ であれば、 $arg0.V=2$ に変更する。(b) 上記の (a) の場合ではなく、アドレスが LIMIT 以上 $sp+92$ 未満であれば、登録不要領域であるので、何もしない。(c) 上記の (b) の場合でない場合、W R I T E データとして登録済であるかどうかを検査し、そうであれば、すでにアドレスは登録済であるので、内容を新しい W R I T E データに更新する。(d) 上記の (c) でない場合、W R I T E データとしての登録が必要であるので、R F に主記憶 W R I T E アドレスを確保し、W R I T E データとして登録する。R F に主記憶アドレスを確保できなかった場合には、登録不能であるため、その R W エントリから B O T T O M までに対応する R B エントリを全て無効化する。

30

【 0 1 3 8 】

(ループを含む多重再利用)

1 レベルで上記のような再利用機構を用いた場合、図 2 2 (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。

40

【 0 1 3 9 】

図 7 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。

【 0 1 4 0 】

図 8 は、図 7 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

50

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。

10

【0141】

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、関数を越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 3 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。

【0142】

以上のことから、多重再利用を実現するには、前述した R F および R B を関数やループの入れ子構造と関連づける機構が必要である。図 9 に示すように、再利用ウィンドウ（R W）を装備することによって、現在実行中かつ登録中である R F および R B の各エントリ（図中では A、B、C と示す）をスタック構造として保持する。関数やループの実行中は、R W に登録されている全てのエントリについて、これまでに述べた方法に基づいて、レジスタおよび主記憶参照を登録していく。

20

【0143】

この際に、あるエントリに関して、（1）登録可能項目数の超過、（2）引数の第 7 ワードの検出、（3）システムコールの検出、によって再利用不可能であると判断した場合には、R W を用いて、そのエントリに対応する R B および上位の R B を特定し、登録を中止することができる。

30

【0144】

なお、R W の深さは有限であるものの、一度に登録可能な多重度を越えて関数やループを検出した場合には、外側の命令区間から順次登録を中止し、より内側の命令区間を登録対象に加えることによって、入れ子関係の動的変化に追従することができる。また、実行および登録中（例えば A）に、再利用可能な命令区間（例えば D）に遭遇した場合には、登録済の入出力をそのまま登録中エントリに追加することによって、R W の深さを越える A の多重再利用も可能となる。

【0145】

（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、R B 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、R B 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 R B 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 R B 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 R B 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。

40

【0146】

これに対して、多重再利用を行うプロセッサとしての M S P 1 A とは別に、命令区間の

50

事前実行によってR B エントリを有効にするプロセッサとしてのS S P 1 B 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。

【0147】

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、R W 4 A・4 B、演算器5 A・5 B、レジスタ6 A・6 B、キャッシュ7 A・7 B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。

【0148】

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。

【0149】

(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3に書き込む値がM S P 1 A とS S P 1 B 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、S S P 1 B は、R B 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S S P 1 B ごとに設けた局所メモリとしてのL o c a l 7 B を使用することとし、C a c h e 7 B および主記憶3への書き込みを不要としている。なお、M S P 1 A 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S S P 1 B のキャッシュインが無効化される。

【0150】

具体的には、R B 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、M S P 1 A と同様にアドレスおよび値をR B へ登録する。以後、主記憶3ではなくR B 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。

【0151】

なお、局所メモリとしてのL o c a l 7 B の容量は有限であり、関数フレームの大きさがL o c a l 7 B 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。

【0152】

(予測機構の参考例)

次に、上記の課題(2)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、R B の使用履歴に基づいて将来の入力を予測し、S S P 1 B へ渡す必要がある。このために、命令区間記憶部2には、予測処理部2 B が設けられている。この予測処理部2 B は、R F の各エントリごとに設けた小さなプロセッサによって構成され、M S P 1 A やS S P 1 B とは独立して入力予測値を求めるものである。

【0153】

前記したように、従来の入力予測では、R B における入力側に登録された全てのアドレスが一律に扱われたことによって、予測の的中率を下げる結果となっている。この問題を解決するためには、予測が的中する可能性が高いアドレスと、予想が外れる可能性が高いアドレスを区別するとともに、値の変化にも着目して必要最小限のアドレスのみを予測対象とすることが必要である。

【0154】

予測が的中することが期待できるアドレスとは、アドレスが固定しており、かつ、値が単調変化するアドレスである。このようなアドレスには、ラベルによって参照される帯域変数、および、スタックポインタやフレームポインタをベースレジスタとして参照される

10

20

30

40

50

局所変数（フレーム内変数）などがある。

【0155】

これらのアドレスを識別するために、ロード命令実行時のアドレス計算が参照するレジスタに定数フラグ（Const-FLAG）が設けられる。スタックポインタやフレームポインタとして用いるレジスタについては無条件に定数フラグがセットされるものとする。その他のレジスタについては、定数をセットする命令が実行された時に定数フラグ（Const-FLAG）がセットされるものとする。

【0156】

次に、過去に参照したアドレスのうち、一度も書き込みが行われないアドレスについては、内容が変化していないことが保証されることになり、このようなアドレスについては予測する必要がないことになる。よって、このようなアドレスを区別するために、書き込みが行われたことを示す変更フラグ（C-FLAG）が設けられる。入力要素としてのアドレスをRF/RBに新規に記録する時には、該アドレスに対応する変更フラグ（C-FLAG）がリセットされ、登録後に該アドレスに対してストア命令が実行された時に、変更フラグ（C-FLAG）がセットされる。

10

【0157】

また、入力要素としてのアドレスを履歴保存対象とするか否かを示す履歴マスク（P-Mask）が設けられる。入力要素としてのアドレスをRF/RBに新規に記録する時には、該アドレスに対応する履歴マスク（P-Mask）（履歴フラグ）がリセットされる。そして、ロード命令実行時に、該アドレスを生成したレジスタに対応する定数フラグ（Const-FLAG）がセットされている場合には、履歴マスク（P-Mask）のうちロード対象となったバイト位置がセットされる。

20

【0158】

以上の定数フラグ（Const-FLAG）、変更フラグ（C-FLAG）、および履歴マスク（P-Mask）の設定の制御は、命令区間記憶部2に設けられているRB登録処理部2Aによって行われる。このRB登録処理部2Aは、小さなプロセッサによって構成され、上記のような判断を行うことによって定数フラグ（Const-FLAG）、変更フラグ（C-FLAG）、および履歴マスク（P-Mask）の設定を行う。

【0159】

（命令区間例）

ここで、命令区間の一例として、図10（a）に示す命令区間が実行され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。すなわち、命令区間の先頭が1000番地となっている。この命令区間は、ループ構造となっており、11個の命令から構成されている。また、図10（b）は、上記命令区間が実行された場合に、RBに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示している。

30

【0160】

第1行目の命令（以降、単に第1の命令のように称する）において、アドレス定数A1がレジスタR1にセットされる。第2の命令において、レジスタR1の内容を用いて、アドレスA1の内容（00010004）がレジスタRxにロードされる。

40

【0161】

第3の命令において、アドレス定数A2がレジスタR2にセットされる。第4の命令において、レジスタR2の内容を用いて、アドレスA2の内容（80000000）がレジスタRyにロードされる。

【0162】

第5の命令において、レジスタRxの内容から4を減じた値をアドレスとするアドレスA3（00010000）の内容（0000AAAA）がレジスタRzにロードされる。第6の命令において、レジスタRxの内容に4を加えた値（00010008）がレジスタRxにセットされる。

【0163】

第7の命令において、レジスタRxの内容（00010008）が、レジスタR1の内容を用い

50

てアドレス A 1 にストアされる。第 8 の命令において、レジスタ R y の内容 (80000000) を右に 1 ビットシフトした値 (40000000) がレジスタ R y にセットされる。

【 0 1 6 4 】

第 9 の命令において、レジスタ R y の内容 (40000000) が、レジスタ R 2 を用いてアドレス A 2 にストアされる。第 1 0 の命令において、レジスタ R y の内容とレジスタ R z の内容とを加えた値 (4000AAAA) が、レジスタ R z にセットされる。

【 0 1 6 5 】

第 1 1 の命令において、レジスタ R z の内容 (4000AAAA) がレジスタ R x を用いてアドレス A 4 にストアされる。第 1 2 の命令において、ループの先頭アドレスとしての 1000 番地に処理が分岐される。

10

【 0 1 6 6 】

上記の第 1 2 の命令に引き続いて行われる第 2 回目のループ処理の例を図 1 0 (c) に示し、図 1 0 (d) に、この場合の R B に登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す。また、第 2 回目のループ処理に引き続いて行われる第 3 回目のループ処理の例を図 1 0 (e) に示し、図 1 0 (f) に、この場合の R B に登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す。

【 0 1 6 7 】

以上のように、ループ第 1 回目では、アドレス A 1 の値 (00010004)、アドレス A 2 の値 (80000000)、アドレス (00010000) の値 (0000AAAA) が入力となっており、レジスタ R x の値 (00010008)、レジスタ R y の値 (40000000)、レジスタ R z の値 (4000AAAA)、アドレス A 1 の値 (00010008)、アドレス A 2 の値 (40000000)、アドレス (00010004) の値 (4000AAAA) が出力となっている。

20

【 0 1 6 8 】

また、ループ第 2 回目では、アドレス A 1 の値 (00010008)、アドレス A 2 の値 (40000000)、アドレス (00010004) の値 (4000AAAA) が入力となっており、レジスタ R x の値 (0001000C)、レジスタ R y の値 (20000000)、レジスタ R z の値 (6000AAAA)、アドレス A 1 の値 (0001000C)、アドレス A 2 の値 (20000000)、アドレス (00010008) の値 (6000AAAA) が出力となっている。

【 0 1 6 9 】

以上の処理において、注目すべき点は、ループ第 1 回目とループ第 2 回目との間におけるデータの依存関係である。第 1 の依存関係は、定数アドレス A 1 に関するループ第 1 回目の第 7 の命令とループ第 2 回目の第 2 の命令との依存関係である。この依存関係において、定数アドレス A 1 の値の変化量は増分 4 と一定である。

30

【 0 1 7 0 】

第 2 の依存関係は、定数アドレス A 2 に関するループ第 1 回目の第 9 の命令とループ第 2 回目の第 4 の命令との依存関係である。この依存関係において、定数アドレス A 2 の値は右 1 ビットシフトのため変化量は不定である。

【 0 1 7 1 】

第 3 の依存関係は、変化するアドレス A 4 に関するループ第 1 回目の第 1 1 の命令とループ第 2 回目の第 5 の命令との依存関係である。この依存関係において、アドレス A 4 のアドレスの変化量は増分 4 と一定、また、値の変化量は不定である。

40

【 0 1 7 2 】

このようなループ構造をループ間の並列処理によって高速化するためには、データ依存関係を動的に把握し、依存関係にない部分を効率よく並列化することが必要である。

【 0 1 7 3 】

(参考例による命令区間の実行例)

次に、図 1 0 (a) に示す命令区間が、上記した参考例における R F および R B の構成によって実行された場合の例について説明する。図 1 1 は、図 1 0 (a) に示す命令区間が実行された場合の R B における実際の登録状況を示している。

50

【 0 1 7 4 】

第 1 の命令において、アドレス定数 A 1 がレジスタ R 1 にセットされる。この命令は、定数をセットする命令であるので、レジスタ R 1 に対応する定数フラグ (Const-FLAG) がセットされる。

【 0 1 7 5 】

第 2 の命令において、レジスタ R 1 の内容を用いて、アドレス A 1 の内容 (00010004) がレジスタ R x にロードされる。この場合、アドレス A 1、マスク (FFFFFFFF)、データ (00010004) は、入力として R B における Input 側の第 1 列に登録され、レジスタ番号 R x、マスク (FFFFFFFF)、およびデータ (00010004) は出力として R B における Output 側の第 1 列に登録される。なお、この時点でレジスタ番号 R x の出力として登録される値は、後の処理で書き換えられるので、図 1 1 に示す値とは異なっている。

10

【 0 1 7 6 】

また、アドレスとして用いたレジスタ R 1 に対応する定数フラグ (Const-FLAG) がセットされているので、アドレス A 1 に対応する履歴マスク (P-Mask) がセットされる。ここで、対象となるデータは (00110000) の 4 バイトデータであるので、これに対応して、アドレス A 1 に対応する履歴マスク (P-Mask) には (FFFFFFFF) がセットされる。そして、レジスタ R x は、定数がセットされるものではないことになるので、レジスタ R x に対応する定数フラグ (Const-FLAG) はリセットされる。

【 0 1 7 7 】

第 3 の命令において、アドレス定数 A 2 がレジスタ R 2 にセットされる。この命令は、定数をセットする命令であるので、レジスタ R 2 に対応する定数フラグ (Const-FLAG) がセットされる。

20

【 0 1 7 8 】

第 4 の命令において、レジスタ R 2 の内容を用いて、アドレス A 2 の内容 (80000000) がレジスタ R y にロードされる。この場合、アドレス A 2、マスク (FFFFFFFF)、データ (80000000) は、入力として R B における Input 側の第 2 列に登録され、レジスタ番号 R y、マスク (FFFFFFFF)、およびデータ (80000000) は出力として R B における Output 側の第 2 列に登録される。なお、この時点でレジスタ番号 R y の出力として登録される値は、後の処理で書き換えられるので、図 1 1 に示す値とは異なっている。

【 0 1 7 9 】

また、アドレスとして用いたレジスタ R 2 に対応する定数フラグ (Const-FLAG) がセットされているので、アドレス A 2 に対応する履歴マスク (P-Mask) がセットされる。ここで、対象となるデータは (80000000) の 4 バイトデータであるので、これに対応して、アドレス A 1 に対応する履歴マスク (P-Mask) には (FFFFFFFF) がセットされる。そして、レジスタ R y は、定数がセットされるものではないことになるので、レジスタ R y に対応する定数フラグ (Const-FLAG) はリセットされる。

30

【 0 1 8 0 】

第 5 の命令において、レジスタ R x の内容から 4 を減じた値をアドレスとするアドレス A 3 (00010000) の内容 (0000AAAA) がレジスタ R z にロードされる。この場合、アドレス A 3、マスク (FFFFFFFF)、データ (0000AAAA) は、入力として R B における Input 側の第 3 列に登録され、レジスタ番号 R z、マスク (FFFFFFFF)、およびデータ (0000AAAA) は出力として R B における Output 側の第 3 列に登録される。なお、この時点でレジスタ番号 R z の出力として登録される値は、後の処理で書き換えられるので、図 1 1 に示す値とは異なっている。

40

【 0 1 8 1 】

また、アドレスとして用いたレジスタ R x に対応する定数フラグ (Const-FLAG) がリセットされているので、アドレス A 3 に対応する履歴マスク (P-Mask) には (00000000) がセットされる。そして、レジスタ R z は、定数がセットされるものではないことになるので、レジスタ R z に対応する定数フラグ (Const-FLAG) はリセットされる。

【 0 1 8 2 】

50

第6の命令において、レジスタR xの内容に4を加えた値(00010008)がレジスタR xにセットされる。ここで、レジスタR xはR BにおけるOutput側に既に登録されているので、R BにおけるInput側には登録されない。そして、R BにおけるOutput側に登録されているレジスタR xに対応する値が(00010008)に更新される。

【0183】

第7の命令において、レジスタR xの内容(00010008)が、レジスタR 1の内容を用いてアドレスA 1にストアされる。ここで、レジスタR xはR BにおけるOutput側に既に登録されているので、R BにおけるInput側には登録されない。アドレスA 1、マスク(FFFF FFFF)、およびデータ(00010008)は、出力としてR BにおけるOutput側の第4列に登録される。また、R BにおけるInput側にはアドレスA 1が既に登録されているので、アドレスA 1に対応する変更フラグ(C-FLAG)がセット(図ではchangeと表示)される。

10

【0184】

第8の命令において、レジスタR yの内容(80000000)を右に1ビットシフトした値(40000000)がレジスタR yにセットされる。ここで、レジスタR yはR BにおけるOutput側に既に登録されているので、R BにおけるInput側には登録されない。そして、R BにおけるOutput側に登録されているレジスタR yに対応する値が(40000000)に更新される。

【0185】

第9の命令において、レジスタR yの内容(40000000)が、レジスタR 2を用いてアドレスA 2にストアされる。ここで、レジスタR yはR BにおけるOutput側に既に登録されているので、R BにおけるInput側には登録されない。アドレスA 2、マスク(FFFFFFFF)、およびデータ(40000000)は、出力としてR BにおけるOutput側の第5列に登録される。また、R BにおけるInput側にはアドレスA 2が既に登録されているので、アドレスA 2に対応する変更フラグ(C-FLAG)がセット(図ではchangeと表示)される。

20

【0186】

第10の命令において、レジスタR yの内容とレジスタR zの内容とを加えた値(4000 AAAA)が、レジスタR zにセットされる。ここで、レジスタR yおよびレジスタR zはR BにおけるOutput側に既に登録されているので、R BにおけるInput側には登録されない。そして、R BにおけるOutput側に登録されているレジスタR zに対応する値が(4000 AAA)に更新される。

30

【0187】

第11の命令において、レジスタR zの内容(4000AAAA)がレジスタR xを用いてアドレスA 4にストアされる。ここで、レジスタR xはR BにおけるOutput側に既に登録されているので、R BにおけるInput側には登録されない。アドレスA 4、マスク(FFFFFFFF)、およびデータ(4000AAAA)は、出力としてR BにおけるOutput側の第6列に登録される。

【0188】

第12の命令において、ループの先頭アドレスとしての1000番地に処理が分岐される。後方分岐が検出された時点で、分岐先と登録を開始した命令区間先頭アドレス(1000)とが比較され、一致した場合に、該命令区間の入出力登録が完了する。

40

【0189】

以上の結果、変更フラグ(C-FLAG)がセットされ、かつ、履歴マスク(P-Mask)がセットされたマスク位置は、アドレスA 1、およびアドレスA 2となる。このマスク位置に対応するアドレス、マスク、および値が、予測対象として、命令区間ごとに過去の入力履歴を保持する履歴情報として、R Bのエントリに記録される。なお、上記の例では出現しなかったが、R Bの入力パターンに登録されたレジスタについては無条件に予測対象として履歴として記録される。

【0190】

図12(a)は、図10(a)に示す命令区間が繰り返し実行された場合における、履歴としてR Bに登録された例を示している。同図に示すように、R Bには、アドレスA 1

50

の列に履歴マスク (P-Mask) として (FFFFFFFF)、および、アドレス A 2 の列に履歴マスク (P-Mask) として (FFFFFFFF) が記憶される。そして、ループの回数が 1 ~ 4 に変化する間に、各アドレスにおける履歴マスク (P-Mask) に対応する値が変化することになる。各履歴の間に示される diff は、対応する入力要素の値の変化量 (差分) を示している。この diff は、予測処理部 2 B によって算出される。

【0191】

同図に示す例では、アドレス A 1 の列に関しては、ループの回数が 1 ~ 4 に変化する間における diff が全て 04 となっている。よって、このアドレスに対応する値は、1 回のループあたりに 04 ずつ増加していくことが予想される。一方、アドレス A 2 の列に関しては、ループの回数が 1 ~ 4 に変化する間に、diff の値が不定となっている。したがって、アドレス A 2 に関しては、予測することが困難であることがわかる。

10

【0192】

以上より、予測処理部 2 B は、履歴において、差分が一定となっているアドレスに関して、該差分がその後も継続するものと仮定して予測を行うとともに、差分が一定でない、または差分が 0 となっているアドレスに関しては予測を行わないようにする。

【0193】

図 1 2 (b) は、上記の予測に基づいて、予測処理部 2 B がアドレス A 1 の値に関して予測を行った場合の、予測エントリとして R B に記録される入力要素の状態を示している。同図において、アドレス A 2 およびアドレス A 7 ~ A 1 0 に関しては、予測値を求めずに直接主記憶 3 を参照することによって得られたものとなっている。

20

【0194】

このように入力要素の予測値が算出されると、SSP 1 B が、この予測入力要素に基づいて命令区間を実行することによって出力要素が算出され、この予測出力要素が予測エントリとして R B に記憶される。その後、MSP 1 A によって命令区間が実行され、予測エントリとして R B に記憶されている予測入力要素と同じ入力値が入力された場合に、それに対応する予測出力要素を出力することによって再利用が実現されることになる。

【0195】

(参考例における課題)

例えばループ制御変数のように、単調変化するアドレス (上記の例ではアドレス A 1 に対応) の内容については正確に予測することができている。しかしながら、命令区間に配列要素が含まれている場合、配列要素の添字が単調変化していても、配列要素値は一般に単調変化するとは限らない。図 1 0 (a) に示す例では、アドレス A 1 からロードした値が配列要素の添字に該当しており、この添字をアドレスとして用いる主記憶参照 (アドレス A 3 ~ A 1 0) はアドレスが変化するために、予測的中率が極めて悪化することになる。ループ間にデータ依存関係がない場合は、キャッシュを直接参照することによって並列処理効果を維持することが可能であるが、例えば図 1 0 (a) に示すプログラム例のように、ループ間に依存関係がある場合には、上記したような予測による効果を得ることができない。図 1 3 は、参考例による予測に基づいて、ループ処理の 2 回目および 3 回目における事前実行を行った結果を示している。同図に示すように、値が確定しないアドレスや、実際の値とは異なる値となっているアドレスが出現しており、予測の効果が薄いことがわかる。

30

40

【0196】

(予測機構)

R B に対する入出力パターンの登録に關与するアドレスは次のように分類することができる。

(1) 第 1 のタイプのアドレスは、内容が変化しない定数アドレスである。この第 1 のタイプのアドレスは、内容が変化しないので、再利用の際に過去の値と内容と比較する必要がなく、したがって、内容を予測する必要がないアドレスである。

(2) 第 2 のタイプのアドレスは、内容の変化量が一定となっている定数アドレスである。この第 2 のタイプのアドレスは、内容の変化量が一定であるので、予測を行うことが可

50

能なアドレスである。上記の例では、アドレス A 1 が第 2 のタイプのアドレスに該当する。

(3) 第 3 のタイプのアドレスは、内容の変化量が不定である定数アドレスである。この第 3 のタイプのアドレスは、予測が困難であるので、書き込みを待ち合わせる必要がある。上記の例では、アドレス A 2 が第 3 のタイプのアドレスに該当する。

(4) 第 4 のタイプのアドレスは、アドレス自体は変化するものの、それぞれのアドレスの内容は変化しないアドレスである。すなわち、ストアが発生しないアドレスであり、結果的に内容が変化しないアドレスである。この第 4 のタイプのアドレスは、内容が変化しないので、再利用の際に過去の値と内容と比較する必要がなく、したがって、内容を予測する必要がないアドレスである。

(5) 第 5 のタイプのアドレスは、アドレス自体が変化し、それぞれのアドレスの内容も、ストアが発生することにより変化するアドレスである。この第 5 のタイプのアドレスは、内容の変化量が一定であることは期待できず予測は困難であるので、書き込みを待ち合わせる必要がある。上記の例では、アドレス A 3 ~ A 10 が第 5 のタイプのアドレスに該当する。

【0197】

本実施形態に係る予測機構は、命令区間の実行時に、上記の第 1 および第 4 のタイプのアドレスを除外し、第 2、第 3、および第 5 のタイプのアドレスについて動的に分類を行うことを可能としている。また、第 5 のタイプのアドレスに関しては、事前実行を行う複数のプロセッサ (MSP1A、SSP1B) 間でデータの待ち合わせを行うようにしている。これを実現するために、上記した参考例における RB に、さらにストアカウンタ (S-Count) という項目が設けられている。図 14 (a) は、RB における入出力記録行の例を示しており、図 14 (b) は、履歴格納行の例を示している。

【0198】

まず、RB における、MSP1A または SSP1B による命令区間の実行中の入出力パターンを記録する行としての入出力記録行において、出力要素としてのアドレス、すなわち Write アドレスにストアカウンタ (S-Count) が設けられている。なお、入出力記録行は、MSP1A および SSP1B のそれぞれに対応して設けられている。

【0199】

このストアカウンタ (S-Count) は、MSP1A または SSP1B によって該当アドレスに対してストアが行われた回数を示している。すなわち、MSP1A または SSP1B によって該当アドレスに対してストアが 1 回行われる毎に、RB 登録処理部 2A が、該当エントリのストアカウンタ (S-Count) を 1 増加させる。

【0200】

また、RB における、各命令区間に対応する履歴エントリを格納する行としての履歴格納行において、Write アドレスにストアカウンタ (S-Count) が設けられている。後方分岐命令の実行時に、入出力記録行に対する命令区間の入出力登録が完了すると、該入出力記録行に登録された内容が、該命令区間に対応する履歴格納行に追加される。この際に、入出力記録行に登録されている各出力要素の Address、Mask、およびストアカウンタ (S-Count) が履歴格納行の Write 側に登録される。

【0201】

また、RB における履歴格納行において、入力要素としてのアドレス、すなわち Read アドレスにもストアカウンタ (S-Count) が設けられている。RB における入出力記録行に登録された入力要素のうち、変更フラグ (C-FLAG) がセットされ、かつ、履歴マスク (P-Mask) がセットされた入力要素が、該命令区間に対応する履歴格納行に追加される。この際に、入出力記録行に登録されている Address、履歴マスク (P-Mask)、および Value が履歴格納行の Read 側に登録される。さらに、RB における入出力記録行に登録された入力要素の全てのアドレスのうち、該当命令区間の前回の実行時における入出力パターンが記憶されている履歴格納行の Write アドレスに含まれているアドレスと一致するアドレスが、該命令区間に対応する履歴格納行に追加される。この際に、入出力記録行に登録されてい

10

20

30

40

50

る該当入力要素のAddress、履歴マスク (P-Mask)、およびストアカウンタ (S-Count) が履歴格納行のRead側に登録される。ここで登録されるストアカウンタ (S-Count) の値は、該当入力要素のアドレスと一致する、前回の命令区間実行時の入出力パターンが記憶されている履歴格納行のWriteアドレスにおけるストアカウンタ (S-Count) 値となる。

【0202】

(アドレスの分類手法)

以上のような構成のRBによって、上記した第2、第3、および第5のタイプのアドレスをどのように分類するかについて以下に説明する。図15(a)は、図10(a)に示す命令区間が繰り返し実行された場合における、履歴格納行の登録例を示しており、図15(b)は、図15(a)に示す履歴に基づいて、予測処理部2Bが以下に示す予測処理を行った際の、予測値格納領域および待機要アドレス格納領域の例を示している。

10

【0203】

各命令区間に対応する履歴格納行に登録されている入力要素に履歴マスク (P-Mask) がセットされている場合、予測処理部2Bは、Addressの変化量およびValueの変化量を求める。Addressの変化量が一定の場合には、予測処理部2Bは、今後も変化量が一定であるものとして予測される外挿値を、該当入力要素に対応する予測Addressとして予測値格納領域に格納する。一方、Addressの変化量が不定である場合には、予測処理部2Bは、最後に出現したAddressを該当入力要素の予測Addressとして予測値格納領域に格納する。

【0204】

Valueの変化量が一定の場合には、予測処理部2Bは、今後も変化量が一定であるものとして予測される外挿値を、該当入力要素に対応する予測Valueとして設定する。そして、RBにおける予測値格納領域に、該当するAddress、Mask、およびValueを格納する。以上の処理により、上記の第2のタイプのアドレスに関する予測機構が実現される。なお、図15(a)および図15(b)に示す例では、アドレスA1が、Addressの変化量が0で一定、Valueの変化量が04で一定となっており、これに基づいて、第2のタイプのアドレスとして予測値格納領域に登録されている。

20

【0205】

一方、Valueの変化量が不定の場合には、予測処理部2Bは、RBにおける待機要アドレス格納領域に、該当するAddress、およびMaskを格納するとともに、ストアカウンタ (S-Count) (待機カウンタ) には、予測距離から1を減じた値に、履歴格納行の該当入力要素に対応するストアカウンタ (S-Count) 値を乗じた値を格納する。なお、予測距離とは、該当命令区間が今後繰り返し実行された場合の、現時点からの実行回数を示している。以上のように待機要アドレス格納領域におけるストアカウンタ (S-Count) を設定することによって、待機すべきストアの回数を的確に設定することが可能となる。これにより、上記の第3のタイプのアドレスに関する予測機構が実現される。なお、図15(a)および図15(b)に示す例では、アドレスA2が、履歴マスク (P-Mask) がセットされ、かつ、Valueの変化量が不定となっており、これに基づいて、第3のタイプのアドレスとして待機要アドレス格納領域に登録されている。

30

【0206】

なお、上記した例では、予測処理部2Bは、ストアカウンタ (S-Count) には、予測距離から1を減じた値に、履歴格納行の該当入力要素に対応するストアカウンタ (S-Count) 値を乗じた値を格納するようになっているが、次のような処理を行ってもよい。すなわち、予測処理部は、RBにおける予測値格納領域に、該当するAddress、およびMaskを格納するとともに、ストアカウンタ (S-Count) には、履歴格納行の該当入力要素に対応するストアカウンタ (S-Count) 値を格納するとともに、予測距離が1だけ短い前回の予測値に基づいて事前実行を開始したSSP1Bを特定する情報を格納してもよい。このようにすれば、全てのSSP1Bによる実行通知のうち、該当するSSP1Bからの実行通知が受信された場合にのみストアカウンタ値を減少させることによって、待機すべきストアの回数を的確に設定することが可能となる。

40

【0207】

50

各命令区間に対応する履歴格納行に登録されている入力要素に履歴マスク (P-Mask) がセットされていない場合、予測処理部 2 B は、上記と同様に、Address の変化量および Value の変化量を求める。Address の変化量が一定の場合には、予測処理部 2 B は、今後も変化量が一定であるものとして予測される外挿値を、該当入力要素に対応する予測 Address として待機要アドレス格納領域に格納する。一方、Address の変化量が不定である場合には、予測処理部 2 B は、最後に出現した Address を該当入力要素の予測 Address として待機要アドレス格納領域に格納する。

【0208】

Value の変化量が一定であることは期待できないので、予測処理部 2 B は、R B における待機要アドレス格納領域に、該当する Address、および Mask を格納するとともに、ストアカウンタ (S-Count) には、履歴格納行の該当入力要素に対応するストアカウンタ (S-Count) 値を格納する。なお、この場合には、アドレスが変化しているため、ストアカウンタ (S-Count) を設定する際に、予測距離を考慮する必要はない。これにより、上記の第 5 のタイプのアドレスに関する予測機構が実現される。なお、図 15 (a) および図 15 (b) に示す例では、アドレス A7 ~ A10 が、履歴マスク (P-Mask) がセットされておらず、かつ、Value の変化量が不定となっており、これに基づいて、第 5 のタイプのアドレスとして待機要アドレス格納領域に登録されている。

10

【0209】

(MSP / SSP による事前実行)

上記のように予測処理部 2 B による処理によって生成された予測値格納行に基づく MSP 1 A / SSP 1 B による事前実行について説明する。SSP 1 B による事前実行の起動開始後の主記憶読み出しは次のように行われる。

20

【0210】

まず Cache / Local 7 B が参照されるとともに、以下に示す処理が行われる。

【0211】

最初に、該当 SSP に対応する入出力記録行のうち、読み出し対象となる主記憶アドレスと同じアドレスが Write 側に登録されているかを SSP 1 B における判定部 8 B が判定する。登録されている場合には、登録されている Value が読み出し対象となる主記憶アドレスの Value として読み出される。Write 側に登録されていない場合には、該当 SSP に対応する入出力記録行のうち、読み出し対象となる主記憶アドレスと同じアドレスが Read 側の Value に登録されているかを SSP 1 B における判定部 8 B が判定する。登録されている場合には、登録されている Value が読み出し対象となる主記憶アドレスの Value として読み出される。Read 側に登録されていない場合には、読み出し対象となる主記憶アドレスと同じアドレスが予測値格納領域に登録されているかを SSP 1 B における判定部 8 B が判定する。登録されている場合には、登録されている Value が読み出し対象となる主記憶アドレスの Value として読み出される。予測値格納領域に登録されていない場合には、読み出し対象となる主記憶アドレスと同じアドレスが待機要アドレス格納領域に登録されているかを SSP 1 B における判定部 8 B が判定する。登録されている場合、ストアカウンタ (S-Count) 値が 0 より大きい場合には、ストアカウンタ (S-Count) 値が 0 になるまで主記憶読み出しを保留し、Value に有効な値がセットされた後に Value を参照する。以上のい

30

40

【0212】

また、MSP 1 A / SSP 1 B による事前実行の起動開始後の主記憶書き込みは次のように行われる。

【0213】

MSP 1 A または SSP 1 B によってストア命令が実行される場合、その旨が通信部 9 A または通信部 9 B によって、その他の全ての SSP 1 B ... または MSP 1 A に対して通知される。各 SSP 1 B において、待機要アドレス格納領域の中に、通知されたアドレスと同一のアドレスが登録されている場合、該アドレスのストアカウンタ (S-Count) を 1

50

だけ減じてValueに書き込み値を格納する。ただし、ストアカウンタ (S-Count) が既に 0 である場合には何も行わない。

【0214】

以上のようにしてSSP1Bによって予測事前実行が行われた結果は、RBにおける予測実行結果記憶行に格納される。

【0215】

(命令区間の実行例)

上記のようにして予測値が生成された後に、予測値に基づく事前実行を行う場合の例について図16を参照しながら以下に説明する。ここで、予測値は、ループ処理が4回繰り返された結果に基づいて生成されたものとする。また、この例では、SSP1Bを2台利用して実行される例を想定している。この2台のSSP1Bを、それぞれSSP#1、およびSSP#2と称する。

10

【0216】

はじめに、MSP1Aがループ5回目の実行を開始し、同時にSSP#1およびSSP#2が、それぞれループ6回目およびループ7回目の予測値を受け取って実行を開始したとする。SSP#1は、SSPのための予測値格納領域にアドレスA1および値(00010018)を保持し、待機要アドレス格納領域に、アドレスA2およびストアカウンタ(S-Count)値としての(0001)、ならびに、アドレスA8およびストアカウンタ(S-Count)値としての(0001)を保持している。同様に、SSP#2は、SSPのための予測値格納領域にアドレスA1および値(0001001C)を保持し、待機要アドレス格納領域に、アドレスA2およびストアカウンタ(S-Count)値としての(0002)、ならびに、アドレスA9およびストアカウンタ(S-Count)値としての(0001)を保持している。

20

【0217】

SSP#1は、第2の命令において、レジスタR1を用いてアドレスA1の内容をレジスタRxにロードしている。この時、前記した主記憶読み出し手順に従って、SSPのための予測値格納領域からアドレスA1の値(00010018)を得ている。また、第4の命令において、レジスタR2を用いてアドレスA2の内容をレジスタRyにロードしている。この時、前記した主記憶読み出し手順に従って、待機要アドレス格納領域からアドレスA2のストアカウンタ(S-Count)値が(0001)であることを認識し、待機する。

【0218】

SSP#2は、第2の命令において、レジスタR1を用いてアドレスA1の内容をレジスタRxにロードしている。この時、前記した主記憶読み出し手順に従って、SSPのための予測値格納領域からアドレスA1の値(0001001C)を得ている。また、第4の命令において、レジスタR2を用いてアドレスA2の内容をレジスタRyにロードしている。この時、前記した主記憶読み出し手順に従って、待機要アドレス格納領域から、アドレスA2のストアカウンタ(S-Count)値が(0001)であることを認識し、待機する。

30

【0219】

その後、MSP1Aが第9の命令を実行し、アドレスA2およびストア値(04000000)をSSP#1、およびSSP#2に通知する。SSP#1では、待機要アドレス格納領域のうち、アドレスA2のストアカウンタ(S-Count)値が1だけ減じられることによって0となり、ストア値(04000000)がValueに格納される。これにより、待機状態が終了して第4の命令の実行が完了する。SSP#2では、待機要アドレス格納領域のうち、アドレスA2のストアカウンタ(S-Count)値が1だけ減じられることによって1となり、ストア値(04000000)がValueに格納されるものの、待機状態は継続する。

40

【0220】

SSP#1は、第5の命令において、レジスタRxを用いてアドレスA8の内容をレジスタRxにロードしている。この時、前記した主記憶読み出し手順に従って、待機要アドレス格納領域から、アドレスA8のストアカウンタ(S-Count)値が(0001)であることを認識し、待機する。

【0221】

50

その後、MSP1Aが第11の命令を実行し、アドレスA8およびストア値(7C00AAAA)をSSP#1、およびSSP#2に通知する。SSP#1では、待機要アドレス格納領域のうち、アドレスA8のストアカウンタ(S-Count)値が1だけ減じられることによって0となり、ストア値(7C00AAAA)がValueに格納される。これにより、待機状態が終了して第5の命令の実行が完了する。SSP#2では、待機要アドレス格納領域に該当アドレスがないため、何も実行されず、待機状態が継続する。

【0222】

その後、SSP#1が第9の命令を実行し、通知部9Bが、アドレスA2およびストア値(02000000)を全てのSSP1B(SSP#2)に通知する。SSP#2では、待機要アドレス格納領域のうち、アドレスA2のストアカウンタ(S-Count)値が1だけ減じられることによって0となり、ストア値(02000000)がValueに格納される。これにより、待機状態が終了して第4の命令の実行が完了する。

10

【0223】

さらに、SSP#1が第11の命令を実行し、通知部9Bが、アドレスA9およびストア値(7E00AAAA)を全てのSSP1B(SSP#2)に通知する。SSP#2では、待機要アドレス格納領域のうち、アドレスA9のストアカウンタ(S-Count)値が1だけ減じられることによって0となり、ストア値(7E00AAAA)がValueに格納される。これにより、待機状態が終了して第5の命令の実行が完了する。

【0224】

(RF/RBの第2の構成例)

20

次に、命令区間記憶部2の第2の構成例について、図17を参照しながら以下に説明する。同図に示すように、命令区間記憶部2は、RB、RA、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。

【0225】

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。

【0226】

RA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。

30

【0227】

RO1およびRO2は、RBおよびRA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、RA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。また、RBおよびRAは、それぞれCAMおよびRAMによって構成されている。

40

【0228】

(第2の構成例における連想検索動作)

次に、第2の構成例における連想検索動作について説明する。図1に示した構成では、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つの行としてRBに登

50

録するようになっている。

【0229】

これに対して、第2の構成例では、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、アドレス管理表としてのRA、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、RAおよびRBの1行に対応づけるようになっている。

【0230】

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。

【0231】

また、RAおよびRBは、入力パターンを木構造として登録しているため、一致比較を行う際には、マルチマッチが行われなくなる。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、第2の構成例によれば、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。

【0232】

次に、図18を参照しながら、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。

【0233】

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RAに伝達され、キー01に対応するRAにおけるラインが参照される。キー01に対応するRA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。

【0234】

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RAに伝達され、キー03に対応するRAにおけるラインが参照される。キー03に対応するRA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14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、RAに対して、エンコード結果としてキー05が伝達される。

【0235】

以上のような処理が繰り返され、RA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、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。

【 0 2 3 6 】

以上のように、第 2 の構成例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、R B 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、R B と R A 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、上記した図 1 に示す例では、R B においてマルチマッチが許容されているので、R B 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、第 2 の構成例によれば、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的大幅に低減することが可能となる。

10

【 0 2 3 7 】

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きているので、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。

【 0 2 3 8 】

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 R B および R A 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きている。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。

20

【 0 2 3 9 】

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。

30

【 0 2 4 0 】

なお、以上のような構成の場合、R A および R B 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 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 2 4 1 】

(連想検索動作の別の例)

上記の例では、図 1 7 に示した R A 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 A 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構成および動作について以下に説明する。

40

【 0 2 4 2 】

まず、図 1 9 (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 A 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 A 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 A における「 0 5 」のライ

50

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。

【0243】

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RAに伝達され、キー01に対応するRAにおけるラインが参照される。キー01に対応するRA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。

10

【0244】

したがって、図19(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。

【0245】

RA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、RA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20(b)に示すようにRAおよびRBが変更されることになる。

20

【0246】

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、RA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、RA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。

【0247】

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。

30

【0248】

以上のようにRA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。

【0249】

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RAに伝達され、キー01に対応するRAにおけるラインが参照される。キー01に対応するRA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。

40

【0250】

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

50

ンが検索される。

【0251】

図20(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、RA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RA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。

10

【0252】

以上のような連想検索動作によれば、RA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。

【0253】

(出力値の格納手段)

上記では、命令区間の入力パターンをRA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について説明する。上記において図17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。

20

【0254】

出力値は、RA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。

30

【0255】

入力パターンに関しては、RA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。

【0256】

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。

40

【0257】

そこで、本実施形態では、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、RA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RA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。

【0258】

しかしながら、このように、出力値格納手段を、RA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、RAにおける、終端フラグEが登録

50

されていないR Aのラインに対しても、R O 1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R Aの全てのラインに対応して、R O 1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、R O 1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。

【0259】

この問題を解消するために、R O 1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17の例では2つ)とともに、R O 1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R O 2に登録するようにしている。

10

【0260】

R O 2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、R O 1における複数のラインから、R O 2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、R O 2における格納情報を、R O 1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、R O 2においては、格納内容の冗長性を低くすることができる。

【0261】

以上のように、出力値格納手段としてR O 1およびR O 2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R O 1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R O 2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。

20

【0262】

(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R A、R B、R O 1、およびR O 2に登録する際の動作について説明する。

30

【0263】

まず、ある命令区間の実行が検出されると、P CおよびReg.の値が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 A、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 A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 Aの終端フラグを「E」とし、入力パターンの登録を終了する。

40

【0264】

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の一致比較が行われる。このようにして、R BおよびR A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 A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 Aの終端フラグを「E」とし、入力パターンの登録を終了する。

50

【0265】

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RA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。

【0266】

(第2の構成例における予測機構)

図21は、第2の構成例を適用した場合のデータ処理装置の概略構成を示している。図2に示す構成と異なる点としては、RW4A・4Bに入出力記録行が設けられている点、命令区間記憶部2において、RFに区間毎情報として履歴格納行、予測値格納領域、および待機要アドレス格納領域が設けられている点、および上記した第2の構成例におけるRB、RA、W1が設けられている点である。なお、W1は、上記したRO1・RO2に相当するものである。その他の構成については、図2に示す構成と同様であるので、ここではその説明を省略する。

10

【0267】

第2の構成例では、命令区間の実行時における入出力パターンを一時的に格納する場所としての入出力記録行は、上記のようにRW4A・4Bとなる。ここで、前記した第1の構成例では、命令区間の実行時における入出力パターンはRBに直接登録されていたので、RW4A・4BはRBの各行に対するポインタによって実現されていた。これに対して、第2の構成例では、RAおよびRBが木構造によって構成されているので、RW4A・4Bが直接RBの行をポイントすることができない。すなわち、第2の構成例では、RW4A・4Bは、RBの各行に対するポインタとして機能するものではなく、命令区間の実行時における入出力パターンを一時的に格納する実質的なメモリとして機能することになる。

20

【0268】

また、図17においては図示していないが、第2の構成例においても、所定の命令区間が繰り返し実行された場合における入力パターンの履歴エントリ、および予測エントリを格納する一時格納メモリ領域として、図1に示すようなRFおよびRBがRFとして設けられている。ただし、この場合には、RBにおけるエントリの行は、履歴エントリを格納する履歴格納行、予測値格納領域、および待機要アドレス格納領域としての数行によって構成されることになる。

30

【0269】

命令区間が実行されると、その入力要素がRW4A・4Bに順次格納され、全ての入力要素が揃い、演算が行われることによって出力要素が確定すると、この入出力パターンが、上記履歴格納行に格納されるとともに、上記のような木構造の入出力パターン格納機構に格納されることになる。

【0270】

また、所定の命令区間が繰り返し実行された場合には、履歴格納行に順次格納され、所定の数の履歴が格納された時点で、上記のように予測処理部2Bによって予測が行われ、予測に基づいてSSP1Bによって実行された結果は、上記のような木構造の入出力パターン格納機構に格納されることになる。

40

【0271】

(本発明の適用例)

「LIMIT」などによって大域変数領域とスタック領域とを区別できるプログラム実行環境があるとした上で、本発明に係るデータ処理装置を他の命令セットアーキテクチャにも適用するためには、スタックフレーム上の変数が、上位/下位関数のどちらの局所変数であるかを区別する手段が必要である。特に、引数を格納するレジスタが不足し、引数をスタックに格納する場合、呼ばれた関数側ではこの区別をすることができないことになる。

【0272】

本実施の形態で取り上げたSPARCプロセッサでは、引数の先頭6ワードを汎用レジ

50

スタに格納しており、6ワード以上の引数を扱う関数は出現頻度が高くないことと、引数がスタックに溢れた時点で再利用ができなくなることの両方を利用することによって、関数/ループの再利用を実現している。S P A R Cプロセッサ同様に、32本以上の汎用レジスタを有する多くのR I S Cプロセッサでも、同様の判断をすることによって、本発明のような関数/ループの再利用を実現することが可能である。

【産業上の利用可能性】

【0273】

本発明に係るデータ処理装置は、上記したようにS P A R Cプロセッサに適用することが可能である。また、S P A R C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R I S Cプロセッサにも適用することが可能である。また、このようなプロセッサを備えたゲーム機器、携帯型電話機、および情報家電などに適用することができる。

10

【図面の簡単な説明】

【0274】

【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におけるR FおよびR Bの構成の概要を示す図である。

【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】命令がデコードされた結果、関数呼び出し命令である場合の処理の流れを示すフローチャートである。

【図4】命令がデコードされた結果、関数復帰命令である場合の処理の流れを示すフローチャートである。

20

【図5】命令がデコードされた結果、後方分岐成立である場合の処理の流れを示すフローチャートである。

【図6】命令がデコードされた結果、後方分岐不成立である場合の処理の流れを示すフローチャートである。

【図7】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8】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9】R Wと、R F・R Bとの関係を示す図である。

【図10】同図(a)は、命令区間の一例を示す図であり、同図(b)は、同図(a)に示す命令区間が実行された場合に、R Bに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す図であり、同図(c)は、同図(a)に示す命令区間に引き続いて行われる第2回目のループ処理の例を示しており、同図(d)は、同図(c)におけるR Bに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す図であり、同図(e)は、同図(c)に示す命令区間に引き続いて行われる第3回目のループ処理の例を示しており、同図(f)は、同図(e)におけるR Bに登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す図である。

30

【図11】図10(a)に示す命令区間が実行された場合のR Bにおける実際の登録状況を示す図である。

【図12】同図(a)は、図10(a)に示す命令区間が繰り返し実行された場合における、履歴としてR Bに登録された例を示す図であり、同図(b)は、予測処理部がアドレスA1の値に関して予測を行った場合の、予測エントリとしてR Bに登録される入力要素の状態を示す図である。

40

【図13】参考例による予測に基づいて、ループ処理の2回目および3回目における事前実行を行った結果を示す図である。

【図14】同図(a)は、R Bにおける入出力記録行の例を示す図であり、同図(b)は、履歴格納行の例を示す図である。

【図15】同図(a)は、図10(a)に示す命令区間が繰り返し実行された場合における、履歴格納行の登録例を示す図であり、同図(b)は、同図(a)に示す履歴に基づいて、予測処理部2Bが以下に示す予測処理を行った際の、予測値格納領域および待機要ア

50

ドレス格納領域の例を示す図である。

【図 1 6】予測値に基づく事前実行を行う場合の実行例を示す図である。

【図 1 7】命令区間記憶部の第 2 の構成例の概略を示す図である。

【図 1 8】図 1 7 に示す R F / R B における連想検索動作の具体例を示す図である。

【図 1 9】同図 (b) は、図 1 7 に示す R F / R B 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 2 0】同図 (b) は、図 1 7 に示す R F / R B 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。

10

【図 2 1】第 2 の構成例を適用した場合のデータ処理装置の概略構成を示す図である。

【図 2 2】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 2 3】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 2 4】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。

【図 2 5】命令区間の一例を示す図である。

【図 2 6】図 2 5 に示す命令区間が実行された場合に、R B に登録される入力アドレスおよび入力データ、並びに出力アドレスおよび出力データを簡略化して示す図である。

20

【図 2 7】R B における実際の登録状況を示す図である。

【図 2 8】図 2 5 に示す命令区間が繰り返し実行された場合における、R B の入力側に登録される履歴の例を示す図である。

【図 2 9】従来の入力予測による予測結果を示す図である。

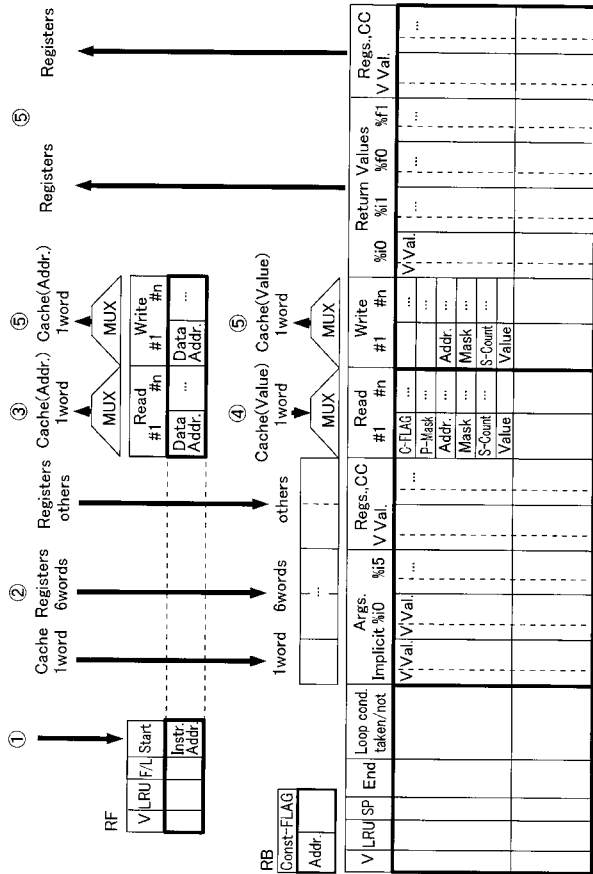
【符号の説明】

【 0 2 7 5 】

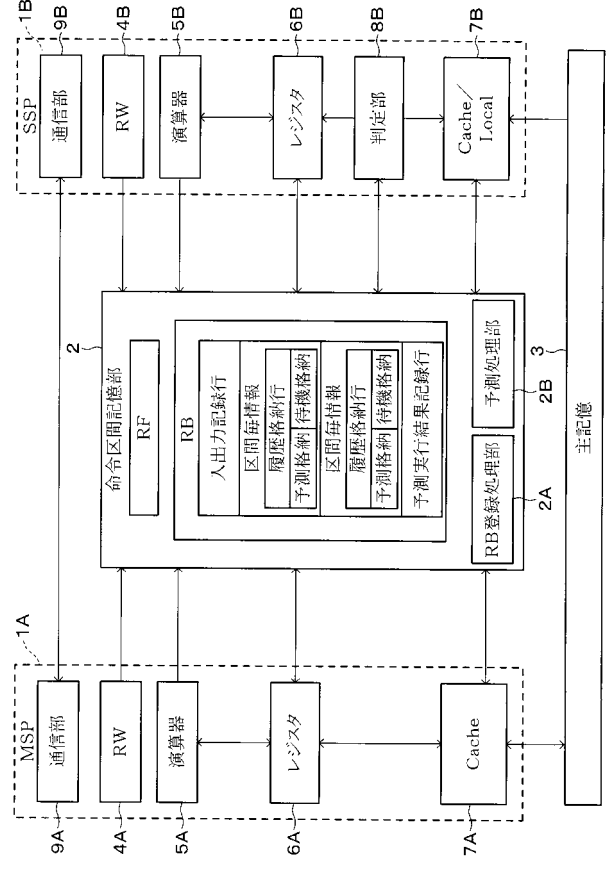
- 1 A M S P
- 1 B S S P
- 2 命令区間記憶部 (入出力記憶手段)
- 2 A R B 登録処理部 (登録処理手段)
- 2 B 予測処理部 (予測処理手段)
- 3 主記憶 (主記憶手段)
- 4 A ・ 4 B R W
- 5 A ・ 5 B 演算器 (第 1 ・ 第 2 の演算手段)
- 6 A ・ 6 B レジスタ
- 7 A ・ 7 B C a c h e
- 8 B 判定部
- 9 A ・ 9 B 通信部

30

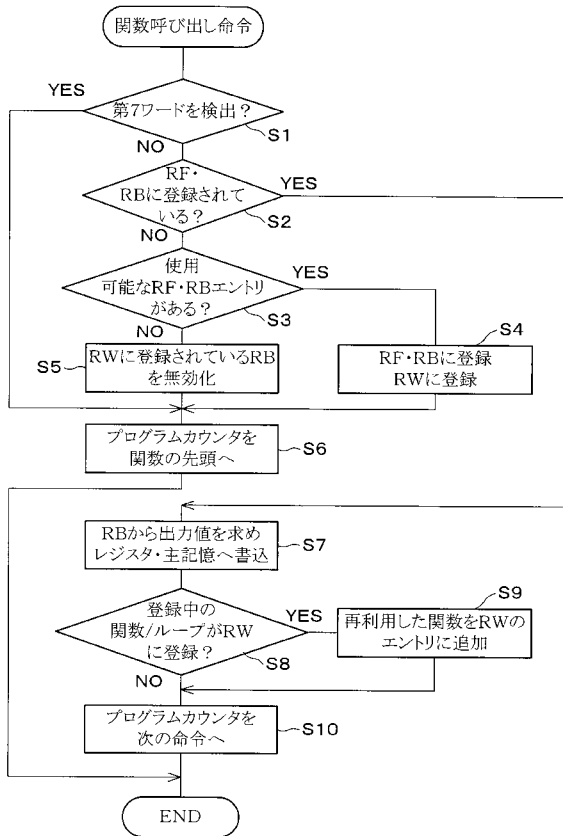
【 図 1 】



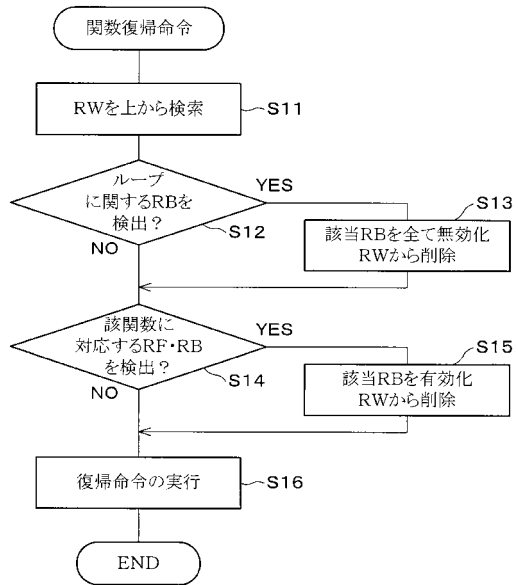
【 図 2 】



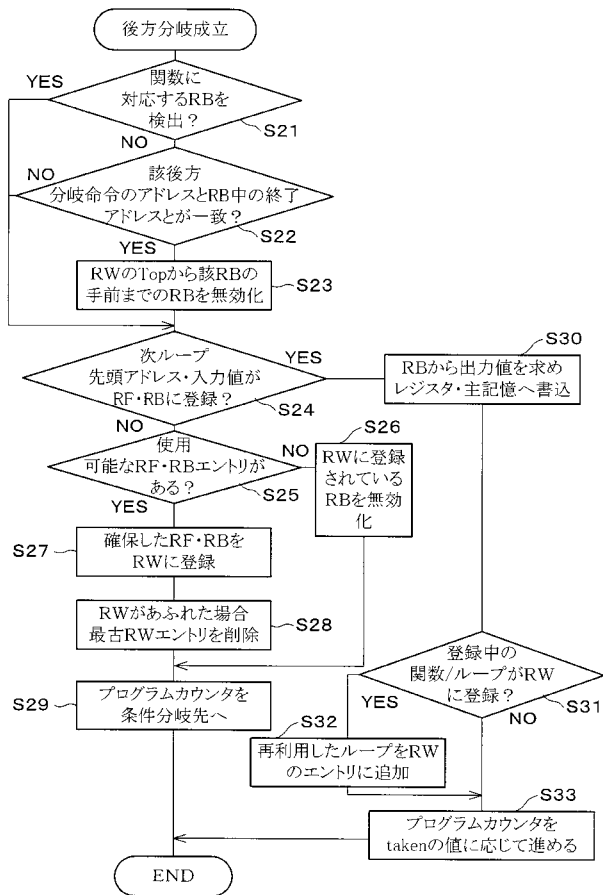
【 図 3 】



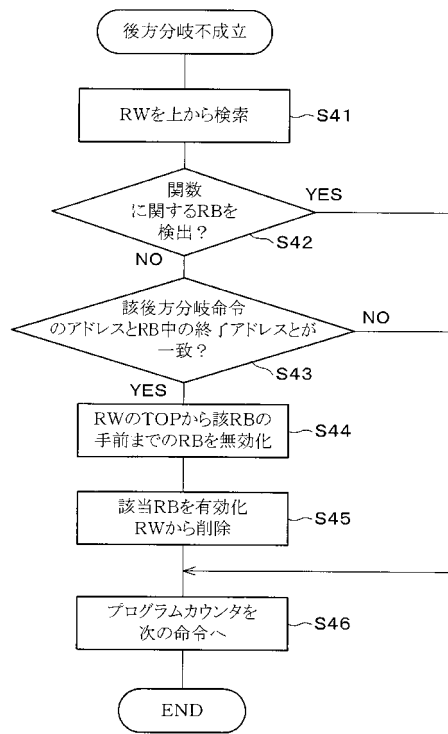
【 図 4 】



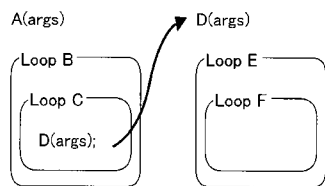
【 図 5 】



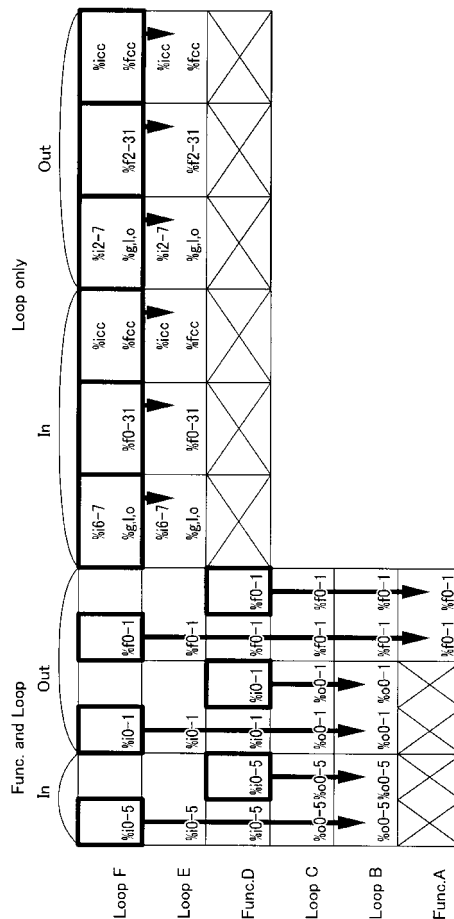
【 図 6 】



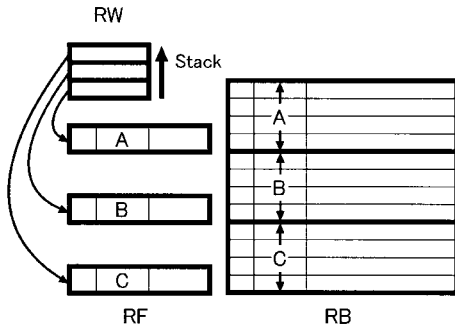
【 図 7 】



【 図 8 】



【 図 9 】



【 図 10 】

(a)

```

loop(PC=1000)
1: set A1 > R1
2: ld (A1-R1) > R2 ... (0010004)
3: set A2 > R2
4: ld (A2-R2) > R3 ... (0000000)
5: ld (A3-Rx-4) > R4 ... (0000aaaa)
6: add Rx+4 > Rx ... (0010008)
7: st Rx > (A1-R1) ... (0001008)
8: shift Ry > Ry ... (0000000)
9: st Ry > (A2-R2) ... (0000000)
10: add Ry+Rz > Rz ... (0000aaaa)
11: st Rz > (A4-Rx) ... (0000aaaa)
12: br loop
    
```

(b)

(c)

```

loop(PC=1000)
set A1 > R1
ld (A1-R1) > Rx ... (00010008)
set A2 > R2 ... (0000000)
ld (A2-R2) > Ry ... (0000000)
ld (A4-Rx-4) > Rz ... (0000aaaa)
add Rx+4 > Rx ... (0001000c)
st Rx > (A1-R1) ... (0001000c)
shift Ry > Ry ... (0000000)
st Ry > (A2-R2) ... (0000000)
add Ry+Rz > Rz ... (0000aaaa)
st Rz > (A5-Rx) ... (0000aaaa)
br loop
    
```

(d)

(e)

```

loop(PC=1000)
set A1 > R1
ld (A1-R1) > Rx ... (0001000c)
set A2 > R2 ... (0000000)
ld (A2-R2) > Ry ... (0000000)
ld (A5-Rx-4) > Rz ... (0000aaaa)
add Rx+4 > Rx ... (00010010)
st Rx > (A1-R1) ... (00010010)
shift Ry > Ry ... (0000000)
st Ry > (A2-R2) ... (0000000)
add Ry+Rz > Rz ... (0000aaaa)
st Rz > (A6-Rx) ... (0000aaaa)
br loop
    
```

(f)

【 図 11 】

| RegisterMemory Read | | RegisterMemory Write | |
|---------------------|----------|----------------------|----------|
| change | value | change | value |
| CP-FLAG | 00000000 | CP-FLAG | 00000000 |
| R1 | 00000000 | R1 | 00000000 |
| R2 | 00000000 | R2 | 00000000 |
| R3 | 00000000 | R3 | 00000000 |
| R4 | 00000000 | R4 | 00000000 |
| R5 | 00000000 | R5 | 00000000 |
| R6 | 00000000 | R6 | 00000000 |
| R7 | 00000000 | R7 | 00000000 |
| R8 | 00000000 | R8 | 00000000 |
| R9 | 00000000 | R9 | 00000000 |
| R10 | 00000000 | R10 | 00000000 |
| R11 | 00000000 | R11 | 00000000 |
| R12 | 00000000 | R12 | 00000000 |
| R13 | 00000000 | R13 | 00000000 |
| R14 | 00000000 | R14 | 00000000 |
| R15 | 00000000 | R15 | 00000000 |
| R16 | 00000000 | R16 | 00000000 |
| R17 | 00000000 | R17 | 00000000 |
| R18 | 00000000 | R18 | 00000000 |
| R19 | 00000000 | R19 | 00000000 |
| R20 | 00000000 | R20 | 00000000 |
| R21 | 00000000 | R21 | 00000000 |
| R22 | 00000000 | R22 | 00000000 |
| R23 | 00000000 | R23 | 00000000 |
| R24 | 00000000 | R24 | 00000000 |
| R25 | 00000000 | R25 | 00000000 |
| R26 | 00000000 | R26 | 00000000 |
| R27 | 00000000 | R27 | 00000000 |
| R28 | 00000000 | R28 | 00000000 |
| R29 | 00000000 | R29 | 00000000 |
| R30 | 00000000 | R30 | 00000000 |
| R31 | 00000000 | R31 | 00000000 |

【 図 12 】

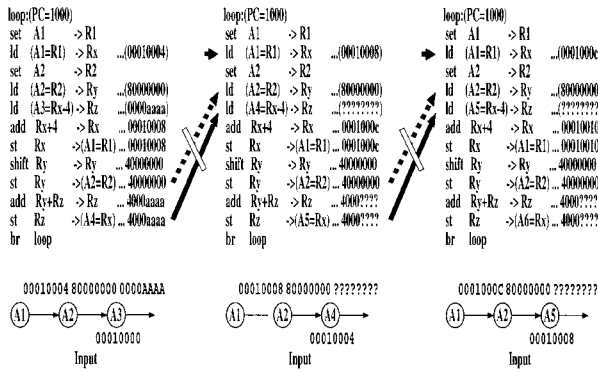
(a)

| Address | A1 | A2 | |
|---------|----------|----------|--------|
| P-Mask | FFFFFFFF | FFFFFFFF | |
| Value | 00010004 | 80000000 | ループ1回目 |
| | diff=04 | diff=-4 | |
| Address | A1 | A2 | |
| P-Mask | FFFFFFFF | FFFFFFFF | |
| Value | 00010008 | 40000000 | ループ2回目 |
| | diff=04 | diff=-2 | |
| Address | A1 | A2 | |
| P-Mask | FFFFFFFF | FFFFFFFF | |
| Value | 0001000C | 20000000 | ループ3回目 |
| | diff=04 | diff=-1 | |
| Address | A1 | A2 | |
| P-Mask | FFFFFFFF | FFFFFFFF | |
| Value | 00010010 | 10000000 | ループ4回目 |
| 距離の推定 | diff=04 | diff=?? | |

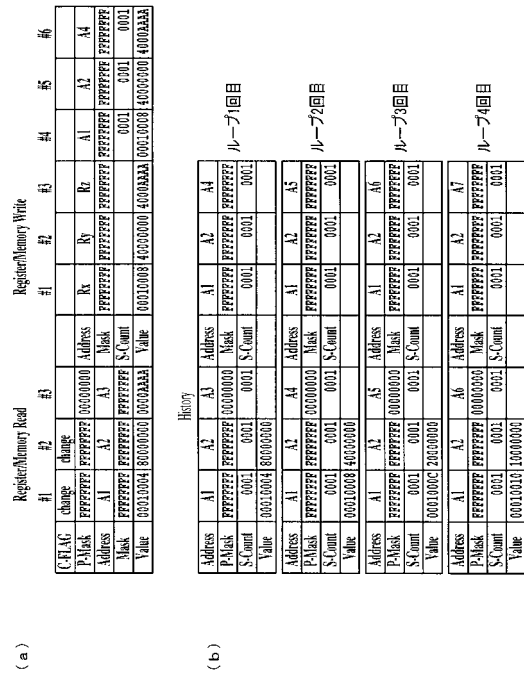
(b)

| 予測値格納領域 | | 待機要アドレス格納領域 | | |
|---------|----------|-------------|----------|-----------------|
| Address | A1 | A2 | A7 | |
| Mask | FFFFFFFF | FFFFFFFF | FFFFFFFF | ループ5回目 |
| Value | 00010014 | 10000000 | ???????? | 予測距離=1⇒MSPへ割当 |
| Address | A1 | A2 | A8 | |
| Mask | FFFFFFFF | FFFFFFFF | FFFFFFFF | ループ6回目 |
| Value | 00010018 | 10000000 | ???????? | 予測距離=2⇒SSP#1へ割当 |
| Address | A1 | A2 | A9 | |
| Mask | FFFFFFFF | FFFFFFFF | FFFFFFFF | ループ7回目 |
| Value | 0001001C | 10000000 | ???????? | 予測距離=3⇒SSP#2へ割当 |
| Address | A1 | A2 | A10 | |
| Mask | FFFFFFFF | FFFFFFFF | FFFFFFFF | ループ8回目 |
| Value | 00010020 | 10000000 | ???????? | 予測距離=4⇒SSP#3へ割当 |

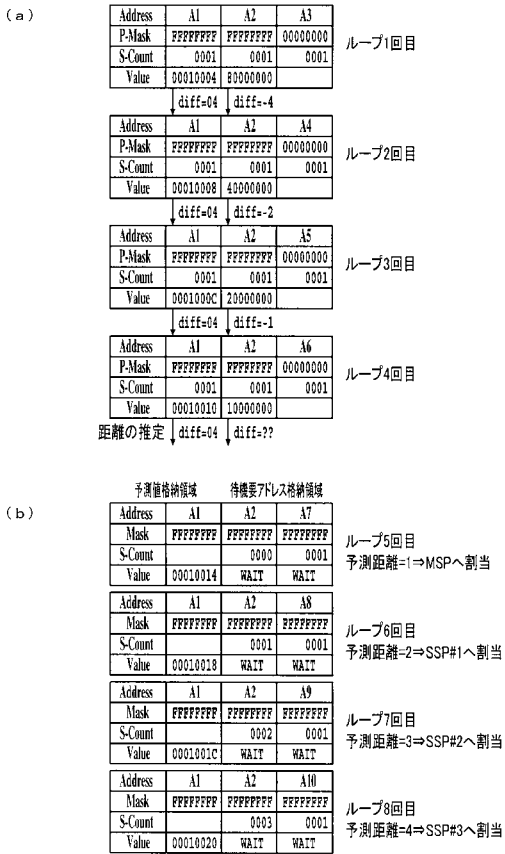
【図 13】



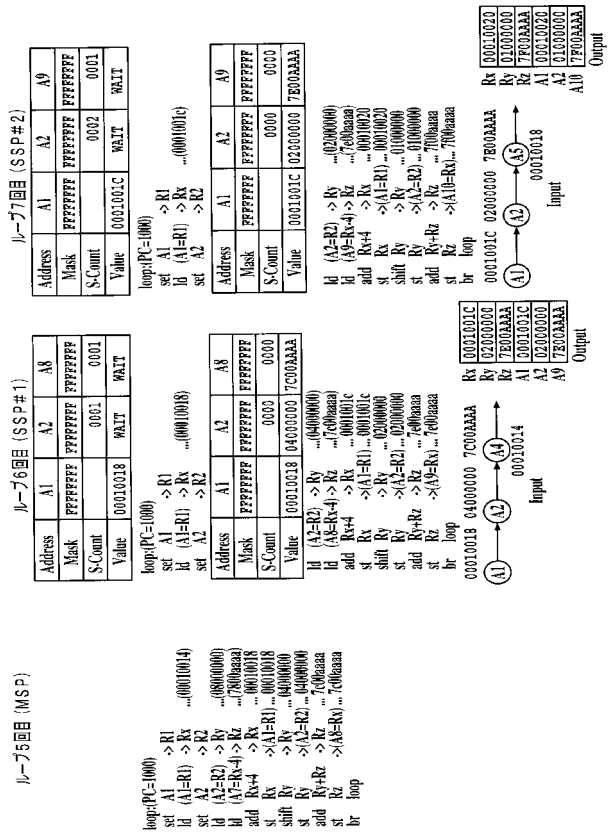
【図 14】



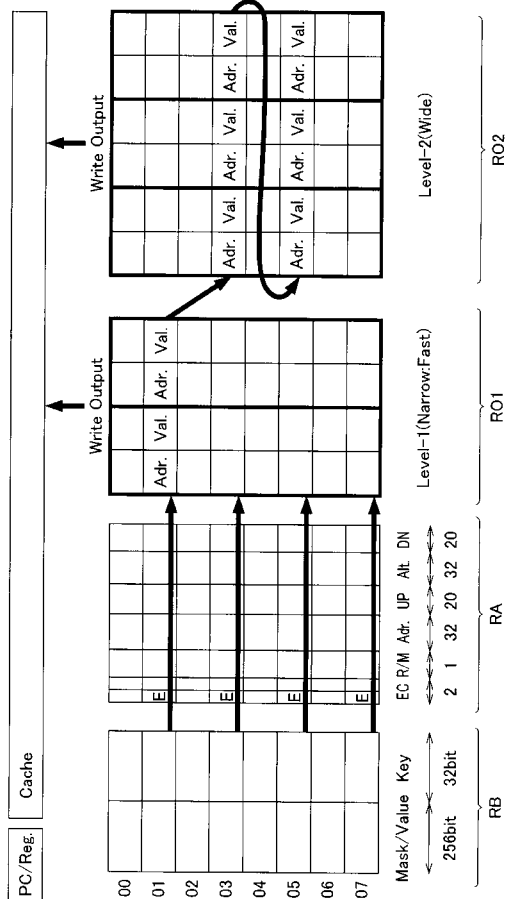
【図 15】



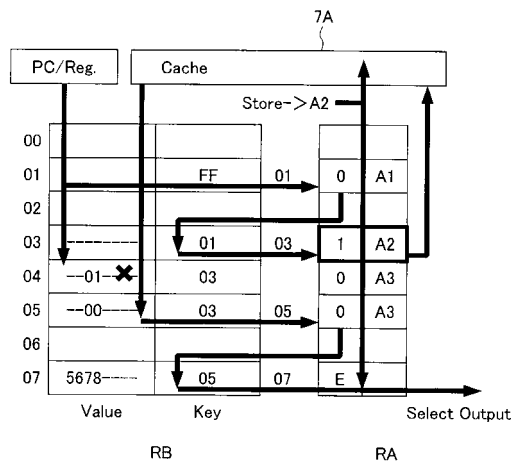
【図 16】



【 図 17 】



【 図 18 】

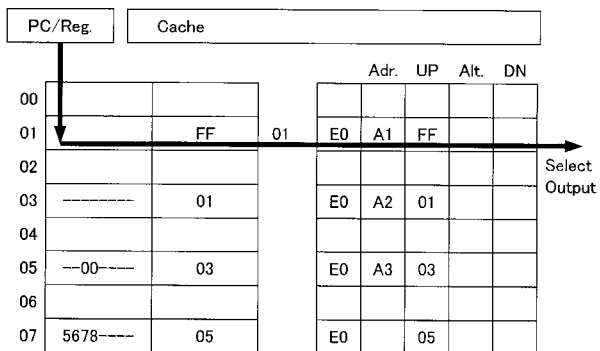


【 図 19 】

(a)

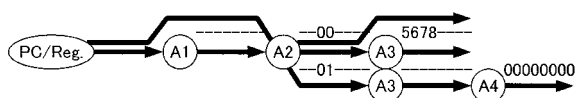


(b)

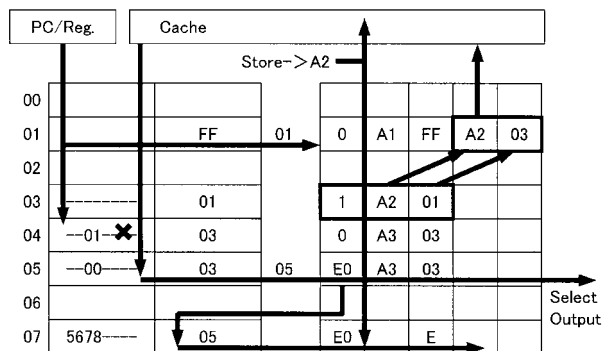


【 図 20 】

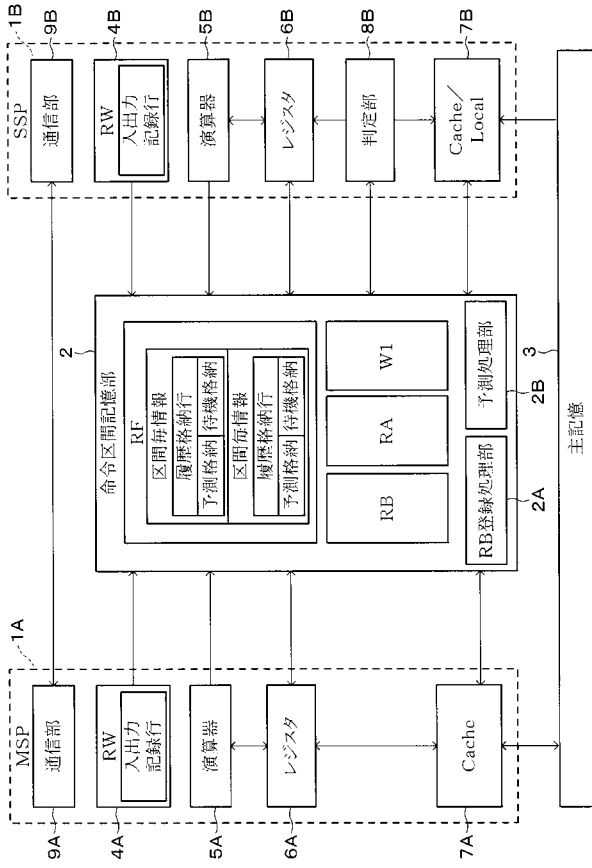
(a)



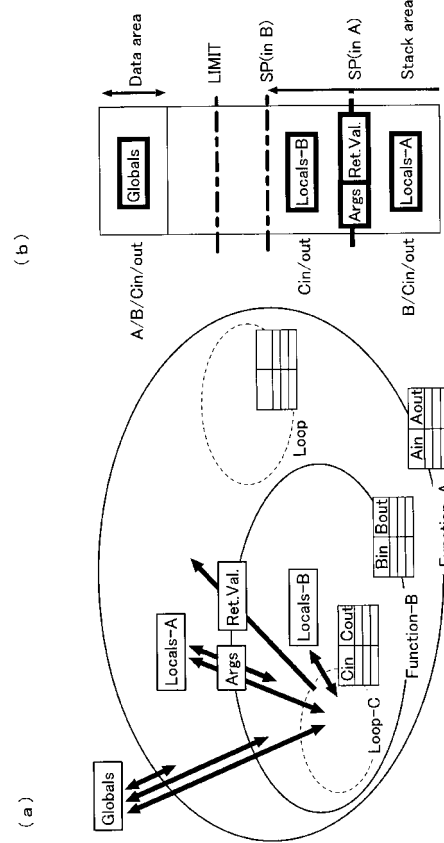
(b)



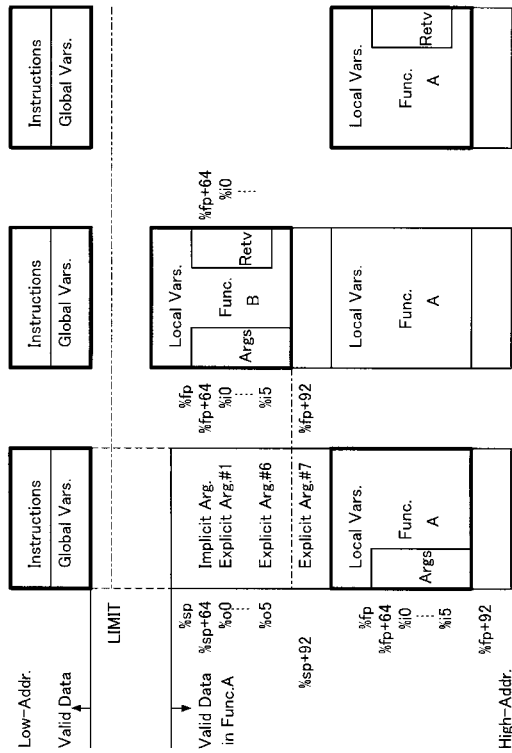
【 図 2 1 】



【 図 2 2 】



【 図 2 3 】

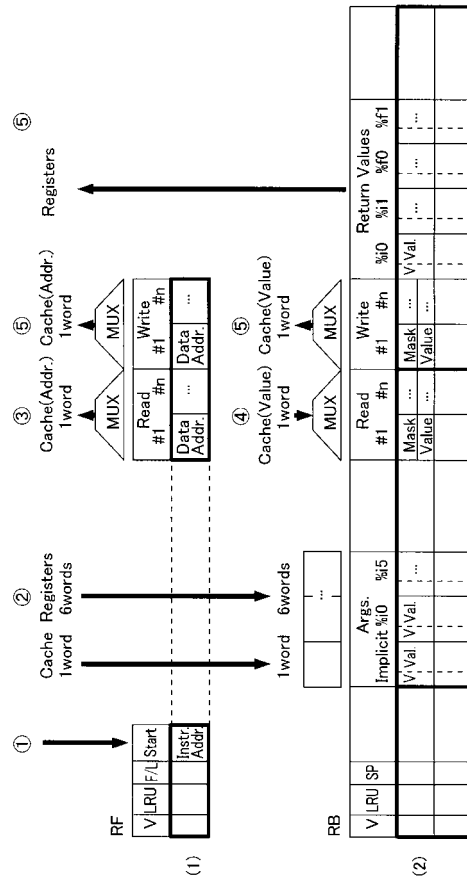


(a) Call A.

(b) Call/Return B.

(c) Return A.

【 図 2 4 】



(1)

(2)

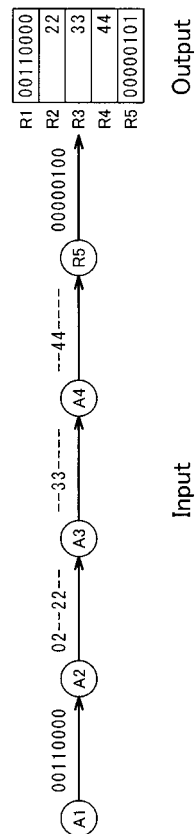
【 図 2 5 】

```

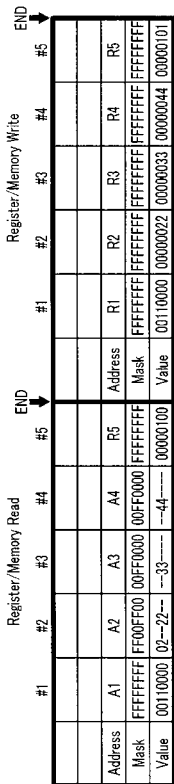
PC: 1000
set A1->R0
ld (R0)->R1
set A2->R0
ldb (R0)->R2
ldb (A2+R2)->R2
set A3->R0
ldb (R0)->R3
ldb (A4=R1+R2)->R4
add R5+1->R5

```

【 図 2 6 】



【 図 2 7 】



【 図 2 8 】

| Address | A1 | A2 | A3 | A4 | R5 | |
|---------|----------|----------|----------|----------|----------|---------|
| Mask | FFFFFFFF | FF00FF00 | 00FF0000 | 00FF0000 | FFFFFFFF | Time=1 |
| Value | 00110000 | 02--22-- | --33--- | --44--- | 00000101 | |
| | | diff=00 | diff=?? | diff=00 | diff=00 | diff=01 |

| Address | A1 | A2 | A3 | A4 | R5 | |
|---------|----------|----------|----------|----------|----------|---------|
| Mask | FFFFFFFF | FF0000FF | 00FF0000 | 00FF0000 | FFFFFFFF | Time=2 |
| Value | 00110000 | 03---23 | --33--- | --44--- | 00000102 | |
| | | diff=00 | diff=?? | diff=?? | diff=11 | diff=?? |

| Address | A1 | A2 | A2+4 | A3 | A4 | |
|---------|----------|----------|----------|----------|----------|---------|
| Mask | FFFFFFFF | FF000000 | FF000000 | 00FF0000 | 00FF0000 | Time=3 |
| Value | 00110000 | 04----- | 24----- | --33--- | --44--- | |
| | | diff=00 | diff=01 | diff=?? | diff=22 | diff=00 |

| Address | A1 | A2 | A2+4 | A3 | A4 | |
|---------|----------|----------|----------|----------|----------|---------|
| Mask | FFFFFFFF | FF000000 | 00FF0000 | 00FF0000 | 00FF0000 | Time=4 |
| Value | 00110000 | 05----- | --25--- | --55--- | --44--- | |
| | | diff=00 | diff=01 | diff=00 | diff=22 | diff=00 |

距離の推定 ↓ diff=00 ↓ diff=01 ↓ diff=00 ↓ diff=22 ↓ diff=00

【 図 2 9 】

| Address | A1 | A2 | A2+4 | A3 | A4 | |
|---------|----------|----------|----------|----------|----------|--------|
| Mask | FFFFFFFF | FF000000 | 00FF0000 | 00FF0000 | 00FF0000 | 予測距離=1 |
| Value | 00110000 | 06----- | --25--- | --77--- | --44--- | |

| Address | A1 | A2 | A2+4 | A3 | A4 | |
|---------|----------|----------|----------|----------|----------|--------|
| Mask | FFFFFFFF | FF000000 | 00FF0000 | 00FF0000 | 00FF0000 | 予測距離=2 |
| Value | 00110000 | 07----- | --25--- | --99--- | --44--- | |

| Address | A1 | A2 | A2+4 | A3 | A4 | |
|---------|----------|----------|----------|----------|----------|--------|
| Mask | FFFFFFFF | FF000000 | 00FF0000 | 00FF0000 | 00FF0000 | 予測距離=3 |
| Value | 00110000 | 08----- | --25--- | --BB--- | --44--- | |

| Address | A1 | A2 | A2+4 | A3 | A4 | |
|---------|----------|----------|----------|----------|----------|--------|
| Mask | FFFFFFFF | FF000000 | 00FF0000 | 00FF0000 | 00FF0000 | 予測距離=4 |
| Value | 00110000 | 09----- | --25--- | --DD--- | --44--- | |