

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第4654433号
(P4654433)

(45) 発行日 平成23年3月23日(2011.3.23)

(24) 登録日 平成23年1月7日(2011.1.7)

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

請求項の数 8 (全 39 頁)

(21) 出願番号	特願2004-324348 (P2004-324348)	(73) 特許権者	504132272
(22) 出願日	平成16年11月8日(2004.11.8)		国立大学法人京都大学
(65) 公開番号	特開2006-134186 (P2006-134186A)		京都府京都市左京区吉田本町36番地1
(43) 公開日	平成18年5月25日(2006.5.25)	(74) 代理人	100080034
審査請求日	平成19年6月19日(2007.6.19)		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
		審査官	三坂 敏夫

最終頁に続く

(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) 【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、

上記データ処理装置が、

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区別処理手段と、

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、上記区別処理手段が、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をセットする命令である場合に、該当アドレスに対して、予測を行うべき入力要素を示す区別情報としての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ことを特徴とするデータ処理装置。

【請求項2】

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、

上記データ処理装置が、

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区別処理手段と、

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、上記区別処理手段が、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、予測を行うべき入力要素を示す区別情報としての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上記変更フラグをセットすることを特徴とするデータ処理装置。

【請求項3】

上記区別処理手段が、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、予測を行うべき入力要素を示す区別情報としての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上記変更フラグをセットすることを特徴とする請求項1記載のデータ処理装置。

【請求項4】

上記区別処理手段が、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、予測を行うべき入力要素を示す区別情報としての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、該アドレスに対して上記履歴フラグをセット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5】

上記区別処理手段が、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、

10

20

30

40

50

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ことを特徴とする請求項4記載のデータ処理装置。

【請求項6】

上記予測処理手段が、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ことを特徴とする請求項1または2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。

10

【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。

【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。

【背景技術】

20

【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

50

投機的先行処理できる命令数を多くするには、相応のハードウェアコストを要する、という点である。第3の問題としては、命令間の依存関係が多いほど、多重に投機的先行処理をする必要が生じ、予測の正当性の検証処理、および誤った予測に基づく処理の無効化処理が極めて複雑になる、という点である。

【0007】

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

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17(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17(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17(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

50

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。

(1) %sp以上の領域のうち、%sp+0~63はレジスタ退避領域、%sp+68~91は引数退避領域であり、いずれも関数の入出力ではない。

(2) 構造体を返す場合の暗黙的引数(Implicit Arg.)は%sp+64~67に格納される。

(3) 明示的引数(Explicit Arg.)はレジスタ%o0~5、%sp+92以上の領域に置かれる。

【0014】

まず、大域変数と局所変数とを区別するために、一般的に、OSが実行時のデータサイズおよびスタックサイズの上限を決めることを利用し、次の事項を仮定する。

(1) 大域変数はLIMIT未満の領域に置かれる。

(2) %spは、LIMIT以下になることはなく、LIMIT~%spの領域は無効である。

【0015】

以上の条件を満たしながら、関数Aが関数Bを呼び出す場合の、メモリマップにおける引数およびフレームの概要を図18に示す。同図を参照しながら、以下にAの局所変数およびBの局所変数を区別する方法について説明する。

【0016】

同図において、(a)はA実行中の状態を示している。LIMIT未満の太枠部分に命令(Instructions)および大域変数(Global Vars.)が格納され、%sp以上に有効な値が格納されている。%sp+64には、Bが構造体を返り値とする場合の暗黙的引数として、構造体の先頭アドレスが格納される。Bに対する明示的引数の先頭6ワードはレジスタ%o0~5、第7ワード以降は%sp+92以上に格納される。ベースレジスタを%spとするオペランド%sp+92が出現した場合、この領域は引数の第7ワードすなわちBの局所変数である。一方、オペランド%sp+92が出現しない場合、この領域はAの局所変数である。このように、(a)の状態では、オペランドを検証することによってAの局所変数とB局所変数とを区別することができる。

【0017】

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

【0018】

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

【0019】

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

【0020】

再利用の際は、Bの局所変数は入出力から除外されるので、Bの局所変数のアドレスが一致している必要がない。このため、いかなるコンテキストであっても、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。

【0021】

Bが参照する大域変数やAの局所変数のアドレスは、そもそもBにおいて生成されるアドレス定数や、大域変数/引数を起源とするポイント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(戻り値、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

【0022】

関数再利用を実現するために、再利用表として、関数管理表(RF)および入出力記録表(RB)を設けることにする。1つの関数を再利用するために必要なハードウェア構成を図19に示す。複数の関数を再利用可能とするには、この構成を複数組用意することになる。

【0023】

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

20

【0024】

戻り値は、%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を使用する戻り値(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

【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、戻り値、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、戻り値レジスタへの書き込みは関数の出力として登録する。その他のレジスタ参照は登録する必要がない。主記憶参照も同様に、読み出しが先行したアドレスについては入力、書き込みは出力として登録する。

【0026】

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

40

【0027】

以降、図19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(戻り値、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。

50

【 0 0 2 8 】

ここで、命令区間の一例として、図 2 0 に示す命令区間が、図 1 9 に示した 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。同図において、P C は、該命令区間が開始された際の P C 値を示している。すなわち、命令区間の先頭が 1 0 0 0 番地となっている。また、図 2 1 は、図 2 0 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しており、図 2 2 は、R B における実際の登録状況を示している。

【 0 0 2 9 】

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 A 1 がレジスタ R 0 にセットされる。第 2 の命令において、レジスタ R 0 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 R 1 に格納される。この場合、アドレス A 1、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 R B における Input 側の第 1 列に登録され、レジスタ番号 R 1、マスク（FFFFFFFF）、およびデータ（00110000）は出力として R B における Output 側の第 1 列に登録される。

10

【 0 0 3 0 】

第 3 の命令において、アドレス定数 A 2 がレジスタ R 0 にセットされる。第 4 の命令において、レジスタ R 0 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 R 2 に格納される。この場合、アドレス A 2、マスク（FF000000）、およびデータ（02）は入力として R B における Input 側の第 2 列に登録される。この際、アドレス A 2 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 R 2、マスク（FFFFFFFF）およびデータ（00000002）は出力として R B における Output 側の第 2 列に登録される。

20

【 0 0 3 1 】

第 5 の命令において、アドレス（A 2 + R 2）からロードされた 1 バイトデータ（22）がレジスタ R 2 に格納されている。アドレス R 2 の値は（02）であったので、アドレス（A 2 + 02）、およびデータ（22）が、入力として R B 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（A 2 + 02）の部分に登録が行われ、アドレス（A 2 + 01）および（A 2 + 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 A 2 に対応するマスクは（FF00FF00）となる。レジスタ番号 R 2、マスク（FFFFFFFF）、およびデータ（00000022）は、出力として R B における Output 側の第 2 列に上書きされる。

30

【 0 0 3 2 】

第 6 の命令において、アドレス定数 A 3 がレジスタ R 0 にセットされる。第 7 の命令において、レジスタ R 0 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 R 3 に格納される。この場合、アドレス A 3、マスク（00FF0000）、およびデータ（33）は入力として R B における Input 側の第 3 列に登録される。レジスタ番号 R 3、マスク（FFFFFFFF）、およびデータ（00000033）は出力として R B における Output 側の第 3 列に登録される。

【 0 0 3 3 】

第 8 の命令において、アドレス（R 1 + R 2）からロードされた 1 バイトデータ（44）がレジスタ R 4 に格納される。この場合、アドレス R 1 とアドレス R 2 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 R 1 およびアドレス R 2 は命令区間の入力とはならない。一方、アドレス（R 1 + R 2）によって生成されたアドレス A 4 は命令区間の入力であるので、アドレス A 4、マスク（00FF0000）、およびデータ（44）は入力として R B における Input 側の第 4 列に登録される。レジスタ番号 R 4、マスク（FFFFFFFF）、およびデータ（00000044）は出力として R B における Output 側の第 4 列に登録される。

40

【 0 0 3 4 】

第 9 の命令において、レジスタ R 5 から値が読み出され、読み出された値に 1 が加えら

50

れた結果が再びレジスタ R 5 に格納される。この場合、レジスタ R 5、マスク (FFFFFFFF)、およびデータ (0000100) は入力として R B における Input 側の第 5 列に登録される。また、レジスタ番号 R 5、マスク (FFFFFFFF)、およびデータ (0000101) は出力として R B における Output 側の第 5 列に登録される。

【 0 0 3 5 】

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) R B 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。

(2) R B における Output 側になれば R B 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) R B における Input 側にもなければ、R B 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。

【 0 0 3 6 】

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) R B 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。

(2) R B 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。

【 0 0 3 7 】

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。

【 0 0 3 8 】

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。

【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 2 0 0 2 年 9 月 1 5 日)

【特許文献 1】特開 2 0 0 4 - 2 5 8 9 0 5 号公報 (公開日 2 0 0 4 年 9 月 1 6 日)

【発明の開示】

【発明が解決しようとする課題】

【 0 0 3 9 】

図 2 3 は、図 2 0 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している。この例では、Time が 1 ~ 4 まで変化するととも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 A 2 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。

【 0 0 4 0 】

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 2 4 は、この従来の入力予測による予測結果を示している。

【 0 0 4 1 】

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 2 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 2 3 に示す例では、アドレス A 2 からロードした値が配列

10

20

30

40

50

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24におけるアドレスA3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がある。図24に示す例では、A2+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予測すべきであるが、マスク位置の変化まで予測することは困難である。この場合には、

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区別処理手段と、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

【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実行を行うようになっている。

40

【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、まず区別処理手段によって、入力パターンに含まれ

50

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が区別される。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。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。

【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、上記区別処理手段が、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。

10

【0049】

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。

【0050】

20

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、上記区別処理手段が、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。

【0051】

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。

30

【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、上記区別処理手段が、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、該アドレスに対して履歴フラグをセットする構成としてもよい。

【0053】

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

【0054】

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

50

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。

【0055】

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、上記区別処理手段が、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。

【0056】

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。

【0057】

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。

【0058】

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。

【発明の効果】

【0059】

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区別処理手段と、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。

【0060】

これにより、予測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。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】

【0061】

本発明の実施の一形態について図1ないし図16に基づいて説明すれば、以下のとおりである。

【0062】

(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

10

20

30

40

50

タ処理装置は、M S P (Main Stream Processor) 1 A、S S P (Shadow Stream Processor) 1 B、再利用表としてのR F / R B (命令列記憶手段) 2、および主記憶(主記憶手段)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、S S P 1 B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。

【0063】

R F / R B 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、R B登録処理部(区別処理手段)2 Aおよび予測処理部(予測処理手段)2 Bを備えた構成となっている。このR F / R B 2の詳細、ならびにR B登録処理部2 Aおよび予測処理部2 Bの詳細については後述する。

10

【0064】

主記憶3は、M S P 1 AおよびS S P 1 Bの作業領域としてのメモリであり、例えばR A M (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、M S P 1 AおよびS S P 1 B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。

【0065】

M S P 1 Aは、R W (再利用記憶手段) 4 A、演算器(第1の演算手段) 5 A、レジスタ6 A、およびC a c h e 7 Aを備えた構成となっている。また、S S P 1 Bは、同様に、R W (再利用記憶手段) 4 B、演算器(第2の演算手段) 5 B、レジスタ6 B、およびC a c h e / L o c a l 7 Bを備えた構成となっている。

20

【0066】

R W 4 A・4 B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R FおよびR B(後述する)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R W 4 A・4 Bは、実際のハードウェア構造としては、R F / R B 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。

【0067】

演算器5 A・5 Bは、レジスタ6 A・6 B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、A L U (arithmetic and logical unit)と呼ばれるものである。レジスタ6 A・6 Bは、演算器5 A・5 B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5 A・5 B、およびレジスタ6 A・6 Bは、S P A R Cアーキテクチャに準じたものとする。C a c h e 7 A・7 Bは、主記憶3と、M S P 1 AおよびS S P 1 B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、S S P 1 Bでは、C a c h e 7 Bには、局所メモリとしてのL o c a l 7 Bが含まれているものとする。

30

【0068】

(R F / R Bの構成)

図1は、本実施形態におけるR F / R B 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、R F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。

40

【0069】

また、R Bは、R F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S P、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)、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

50

タおよび条件コード(Regs.,CC)を保持している。また、R B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。

【0070】

上記のR FおよびR B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、R FまたはR B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。

10

【0071】

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。R F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、R B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。

20

【0072】

上記R B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、R F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バイトのうちどのバイトが有効であるかを示す必要がある。すなわち、Maskは、4バイトのうちどのバイトが有効であるかを示す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)の詳細については後述する。

30

【0073】

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。

40

【0074】

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。

【0075】

以上のように、R F / R B 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。

50

【 0 0 7 6 】

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、そのエントリに対応する信号がONとなるように動作す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。

【 0 0 7 7 】

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が実現されることになる。

【 0 0 7 8 】

(再処理の概略)

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

【 0 0 7 9 】

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、登録中の入出力表エントリを有効にする。

【 0 0 8 0 】

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。全ての入力が一致した場合に、(5)RBに登録済の出力(戻り値、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。

【 0 0 8 1 】

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。

【 0 0 8 2 】

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。全ての入力が一致した場合に、(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。

【 0 0 8 3 】

再利用した場合、R B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、R FおよびR Bへの登録を開始する。

【 0 0 8 4 】

(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。

【 0 0 8 5 】

(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、S 1 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。S 1 において Y E S、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、R Wに登録されている登録中 R B エントリを全て無効化し、S 6 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。

【 0 0 8 6 】

一方、S 1 において N O、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 R F および R B に登録されているか否かを検索する (S 2)。S 2 において Y E S、すなわち、該関数呼び出しおよび入力値が R F および R B 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。

【 0 0 8 7 】

S 2 において N O、すなわち、該関数呼び出しおよび入力値が R F および R B に登録されていないと判定された場合、該関数のための R F エントリおよび R B エントリを確保しようと試み、(1)既存の R F 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 R F エントリ以外に、使用可能な R F 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 R B エントリ以外に、使用可能な R B エントリがあるかを判定する (S 3)。

【 0 0 8 8 】

S 3 において N O、すなわち、使用可能な R F ・ R B エントリがないと判定された場合には、登録を開始せず、R Wに登録されている R B を全て無効化し (S 5)、R W を空にする。一方、S 3 において Y E S、すなわち、使用可能な R F ・ R B エントリがあると判定された場合には、該関数のための R F エントリおよび R B エントリを確保し、R W に登録する (S 4)。ここで、R W に登録した際に、R W に登録されている R W エントリが溢れた際には、最も古い R W エントリを削除し、対応する R B 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。

【 0 0 8 9 】

一方、S 2 において Y E S、すなわち、該関数呼び出しおよび入力値が R F および R B 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、R B 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 R W 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 R B エントリの内容のうち必要なものを R W に登録されているエントリに追加する (S 9)。ここで、R W の T O P から順に登録し、途中で R B があふれた場合には、以降、R W の B O T T O M までに対する R B を無効化し、R W 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 1 0)、処理を終了する。

【 0 0 9 0 】

(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

10

20

30

40

50

トを参照しながら以下に説明する。S 1 1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 1 2)。ここでループに関するRBが検出されると(S 1 2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 1 3)。

【0091】

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 1 4)。ここで、該関数に対応するRF/RBが検出されると(S 1 4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 1 5)。

【0092】

その後、復帰命令を実行し(S 1 6)、処理を終了する。

10

【0093】

(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 2 1)。S 2 1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 2 4のステップに移行する。

【0094】

一方、S 2 1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(S 2 2)。S 2 2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S 2 4のステップに移行する。

20

【0095】

S 2 2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(S 2 3)、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。

【0096】

次に、S 2 4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。S 2 4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S 3 0のステップに移行する。

30

【0097】

一方、S 2 4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(S 2 5)。

【0098】

S 2 5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(S 2 6)、RWを空にする。その後、S 2 9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。

40

【0099】

一方、S 2 5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(S 2 7)。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(S 2 8)、それに対応するRBを無効化する。その後、S 2 9にお

50

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。

【0100】

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。

【0101】

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。

【0102】

(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、すなわち、関数に対応するRBを検出したと判定された場合、S46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。

【0103】

S42においてNO、すなわち、関数に対応するRB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RB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(S43)。S43においてNO、すなわち、該後方分岐命令に対応するRF/RBを検出していないと判定された場合、S46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。

【0104】

一方、S43においてYES、すなわち、該後方分岐命令に対応するRF/RBを検出したと判定された場合、RWのTOPから該RBの手前までのRBを全て無効化し(S44)、RWから削除する。また、該RBエントリを有効化し、かつtaken=0とし、RWから削除する(S45)。その後、S46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。

【0105】

(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、RW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RWに登録されているRBに対して登録する。以下では、(1)汎用レジスタREADの場合、(2)汎用レジスタWRITEの場合、(3)浮動小数点レジスタREADの場合、(4)浮動小数点レジスタWRITEの場合、(5)条件コードレジスタICC-READの場合、(6)条件コードレジスタICC-WRITEの場合、(7)浮動小数点条件コードレジスタFCC-READの場合、(8)浮動小数点条件コードレジスタFCC-WRITEの場合、(9)主記憶READの場合、(10)主記憶WRITEの場合についてそれぞれ説明する。

【0106】

(1)汎用レジスタREADの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして、(1-1)該RBがループ関数かつ%0-6の場合、または該RB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RWをたどり、該RBが関数の場合、処理を終了する。一方、該RBが関数ではな

10

20

30

40

50

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 0 7 】

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。

10

【 0 1 0 8 】

（ 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 がリーフ関数かつ%o0-5の場合、または該 R B が非リーフ関数かつ%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。

20

【 0 1 0 9 】

一方、（ 2 - 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=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 R W エントリに進む。

【 0 1 1 0 】

（ 3 ）浮動小数点レジスタ 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（ 3 - 1 ）該 R B が関数の場合、何もせずに処理を終了する。一方、（ 3 - 2 ）該 R B 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。

30

【 0 1 1 1 】

（ 4 ）浮動小数点レジスタ 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（ 4 - 1 ）該 R B 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 R W 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。

40

【 0 1 1 2 】

一方、（ 4 - 2 ）該 R B 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。

【 0 1 1 3 】

（ 5 ）条件コードレジスタ I C C - 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（ 5 - 1 ）該 R B が関数の場合、何もせずに処理を終了する。一方、（ 5 - 2 ）該 R B 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。

50

【0114】

(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。

【0115】

(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

10

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。

【0116】

(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。

【0117】

(9) 主記憶 READ の場合

まず、RWのTOPからBOTTOMまで順にたどる。そして、RBにWRITE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、RBにREAD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。

20

【0118】

その後、再度RWのTOPからBOTTOMまで順にたどる。そして、(a)アドレスが、RB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、WRITE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READ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、READ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、READデータとしての登録が必要であるので、RFに主記憶READアドレスを確保し、READデータとして登録する。RF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RWエントリからBOTTOMまでに対応するRBエントリを全て無効化する。

30

【0119】

(10) 主記憶 WRITE 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。

40

【0120】

その後、RWのTOPからBOTTOMまで順にたどる。そして、(a)アドレスが、RB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、WRITE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WRITEデータに更新する。(d)上記の(c)でない場合、WRITEデータとしての登録が必要であるので、RFに主記憶WRITEアドレスを確保し、WRITEデータとして登録する。RFに主記憶アドレ

50

スを確認できなかった場合には、登録不能であるため、そのRWエントリからBOTTOMまでに対応するRBエントリを全て無効化する。

【0121】

(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17(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。

10

【0122】

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。

【0123】

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。

20

【0124】

浮動小数点レジスタはレジスタウィンドウに含まれないので、出力された%f0~1は、関数Aを含む全階層の出力となる。一方、その他のレジスタ入出力は、関数を超えて影響がおよぶことはない。すなわち、ループF内部における入出力、すなわち、レジスタ入力としての%i6~7、%g,l,o、%f0~31、%icc、%fcc、およびレジスタ出力としての%i2~7、%g,l,o、%f2~31、%icc、%fccの影響範囲はループEまでとなる。主記憶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。

30

【0125】

以上のことから、多重再利用を実現するには、前述したRFおよびRB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(RW)を装備することによって、現在実行中かつ登録中であるRFおよびRB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、RW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。

40

【0126】

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、RWを用いて、そのエントリに対応するRBおよび上位のRBを特定し、登録を中止することができる。

【0127】

50

なお、RWの深さは有限であるものの、一度に登録可能な多重度を超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、RWの深さを超えるAの多重再利用も可能となる。

【0128】

（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、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くなる。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

【0129】

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。

20

【0130】

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、RF/RB2、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RF/RB2に対して入出力を登録するパスを示している。

【0131】

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。

30

【0132】

（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RF/RB2、また、その他の局所的な参照にはSSP1Bごとに設けた局所メモリとしてのLocal7Bを使用することとし、Cache7Bおよび主記憶3への書き込みを不要としている。なお、MSP1Aが主記憶3に対して書き込みを行った場合には、対応するSSP1Bのキャッシュラインが無効化される。

40

【0133】

具体的には、RB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RB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事は、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。

【0134】

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

50

が Local 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。

【 0 1 3 5 】

(予測機構)

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、R B の使用履歴に基づいて将来の入力を予測し、SSP 1 B へ渡す必要がある。このために、RF / R B 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、RF 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、MSP 1 A や SSP 1 B とは独立して入力予測値を求めるものである。

10

【 0 1 3 6 】

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。

【 0 1 3 7 】

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

20

【 0 1 3 8 】

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。

【 0 1 3 9 】

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

30

【 0 1 4 0 】

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 / R B 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)

40

【 0 1 4 1 】

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、RF / R B 2 に設けられている R B 登録処理部 2 A によって行われる。この R B 登録処理部 2 A 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。

【 0 1 4 2 】

(命令区間の実行例)

50

ここで、命令区間の一例として、図20に示す命令区間が、図1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10は、図20に示す命令区間が実行された場合のRBにおける実際の登録状況を示している。

【0143】

第1の命令において、アドレス定数A1がレジスタR0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R0に対応する定数フラグ(Const-FLAG)がセットされる。

【0144】

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3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)、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。

【0145】

また、アドレスとして用いたレジスタR0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A1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A1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R1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R1に対応する定数フラグ(Const-FLAG)はリセットされる。

【0146】

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。この命令は、定数をセットする命令であるので、レジスタR0に対応する定数フラグ(Const-FLAG)がセットされる。

【0147】

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3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。

【0148】

また、アドレスとして用いたレジスタR0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A2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(02)の1バイトデータであるので、これに対応して、アドレスA2に対応する履歴マスク(P-Mask)には(FF000000)がセットされる。そして、レジスタR2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R2に対応する定数フラグ(Const-FLAG)はリセットされる。

【0149】

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。

【0150】

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) がリセットされているので、アドレス (A 2 + 02) に対応する履歴マスク (P-Mask) はセットされない。すなわち、アドレス A 2 に対応する履歴マスク (P-Mask) は (FF000000) のままとなる。そして、レジスタ R 2 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 2 に対応する定数フラグ (Const-FLAG) はリセットされる。

【 0 1 5 1 】

第 6 の命令において、アドレス定数 A 3 がレジスタ R 0 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 0 に対応する定数フラグ (Const-FLAG) がセットされる。

【 0 1 5 2 】

第 7 の命令において、レジスタ R 0 の内容をアドレスとする主記憶 3 からロードされた 1 バイトデータ (33) がレジスタ R 3 に格納される。この場合、アドレス A 3、マスク (00FF0000)、およびデータ (33) は入力として R B における Input 側の第 3 列に登録される。レジスタ番号 R 3、マスク (FFFFFFF)、およびデータ (00000033) は出力として R B における Output 側の第 3 列に登録される。

【 0 1 5 3 】

また、アドレスとして用いたレジスタ R 0 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 3 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 (33) の 1 バイトデータであるので、これに対応して、アドレス A 3 に対応する履歴マスク (P-Mask) には (00FF0000) がセットされる。そして、レジスタ R 3 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 3 に対応する定数フラグ (Const-FLAG) はリセットされる。

【 0 1 5 4 】

第 8 の命令において、アドレス (R 1 + R 2) からロードされた 1 バイトデータ (44) がレジスタ R 4 に格納される。この場合、アドレス R 1 とアドレス R 2 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 R 1 およびアドレス R 2 は命令区間の入力とはならない。一方、アドレス (R 1 + R 2) によって生成されたアドレス A 4 は命令区間の入力であるので、アドレス A 4、マスク (00FF0000)、およびデータ (44) は入力として R B における Input 側の第 4 列に登録される。レジスタ番号 R 4、マスク (FFFFFFF)、およびデータ (00000044) は出力として R B における Output 側の第 4 列に登録される。

【 0 1 5 5 】

また、アドレスとして用いたレジスタ R 1 およびレジスタ R 2 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 4 に対応する履歴マスク (P-Mask) はセットされない。すなわち、アドレス A 4 に対応する履歴マスク (P-Mask) は (00000000) となる。そして、レジスタ R 4 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 4 に対応する定数フラグ (Const-FLAG) はリセットされる。

【 0 1 5 6 】

第 9 の命令において、レジスタ R 5 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 R 5 に格納される。この場合、レジスタ R 5、マスク (FFFFFFF)、およびデータ (00000100) は入力として R B における Input 側の第 5 列に登録される。また、レジスタ番号 R 5、マスク (FFFFFFF)、およびデータ (00000101) は出力として R B における Output 側の第 5 列に登録される。この時、レジスタ R 5 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 5 に対応する定数フラグ (Const-FLAG) はリセットされる。

【 0 1 5 7 】

その後、アドレス A 2、およびアドレス A 3 に対してストア命令が実行され、アドレス A 2、およびアドレス A 3 に対して変更フラグ (C-FLAG) がセットされたとする。

【 0 1 5 8 】

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

10

20

30

40

50

トされたマスク位置は、アドレス A 2 の第 1 バイト、アドレス A 3 の第 2 バイトのみとなる。このマスク位置のみ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。また、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。

【 0 1 5 9 】

図 1 1 は、図 2 0 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 2 の列に履歴マスク (P-Mask) として (FF000000)、アドレス A 3 の列に履歴マスク (P-Mask) として (00FF0000)、およびアドレス R 5 の列に履歴マスク (P-Mask) として (FFFFFFF) が記憶される。そして、Time 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。

10

【 0 1 6 0 】

同図に示す例では、アドレス A 2 およびアドレス R 5 の列に関しては、Time が 1 ~ 4 に変化する間における diff が全て 01 となっている。よって、これらのアドレスに対応する値は、単位時間あたりに 01 ずつ増加していくことが予想される。一方、アドレス A 3 の列に関しては、Time が 1 ~ 4 に変化する間に、diff は 00 であったり 02 であったりしている。したがって、アドレス A 3 に関しては、予測することが困難であることがわかる。

【 0 1 6 1 】

20

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。

【 0 1 6 2 】

図 1 2 は、上記の予測に基づいて、予測処理部 2 B がアドレス A 2 およびアドレス R 5 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 + 4) およびアドレス A 3 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。

【 0 1 6 3 】

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。

30

【 0 1 6 4 】

(R F / R B の第 2 の構成例)

次に、R F / R B 2 の第 2 の構成例について、図 1 3 を参照しながら以下に説明する。同図に示すように、R F / R B 2 は、R B、R F、R O 1 (第 2 出力パターン記憶手段)、および R O 2 (第 1 出力パターン記憶手段) を備えた構成となっている。

【 0 1 6 5 】

40

R B 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。

【 0 1 6 6 】

R F 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

50

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。

【0167】

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

10

【0168】

(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に登録するようになっている。

【0169】

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

【0170】

これにより、冗長性をなくし、RF/RB2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。

【0171】

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、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。

30

【0172】

次に、図14を参照しながら、RF/RB2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。

40

【0173】

次に、マッチラインとして選択されたラインのRBにおける番地である「01」が、エンコード結果としてRFに伝達され、キー01に対応するRFにおけるラインが参照される。キー01に対応するRFにおけるラインでは、比較要フラグが「0」であり、比較す

50

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になる。

【0174】

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、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

【0175】

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

【0176】

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、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、上記した図1に示す例では、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、第2の構成例によれば、RF/RB2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

【0177】

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。

【0178】

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

40

【0179】

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。

【0180】

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

50

【 0 1 8 1 】

(連想検索動作の別の例)

上記の例では、図 1 3 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構成および動作について以下に説明する。

【 0 1 8 2 】

まず、図 1 5 (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、10
 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。

【 0 1 8 3 】

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、20
 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 1 8 4 】

したがって、図 1 5 (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30
 確認されると、Tr 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。

【 0 1 8 5 】

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 1 6 (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 1 8 6 】

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 1 8 7 】

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。

【 0 1 8 8 】

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。50

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。

【0189】

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

10

【0190】

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

【0191】

図16(b)に示す例では、Keyが「03」となっているラインは「04」および「05」の2つあるが、主記憶3から読み出された値が「00」であるので、「05」のラインがマッチラインとして選択され、RFに対して、エンコード結果としてキー05が伝達される。キー05に対応するRFにおけるラインでは、終端フラグが「E」となっているので、入力パターンが全て一致したと判定され、該当命令区間は再利用可能と判断される。そして、終端フラグEが検出されたラインから「Select Output」信号が出力され、RO1およびRO2に格納されている、該ラインに対応する出力値がレジスタ6Aおよび主記憶3に対して出力される。

20

【0192】

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

30

【0193】

(出力値の格納手段)

上記では、命令区間の入力パターンをRFおよびRBに登録し、連想検索動作を行うことについて説明したが、以下では、入力パターンの一致が確認された後に、再利用として出力される出力値を格納する手段について説明する。上記において図13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。

40

【0194】

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

【0195】

入力パターンに関しては、RFおよびRBにおいて木構造によって登録されている。そして、木構造の末端となっているライン、すなわち、終端フラグEが登録されているラインにおいて、再利用が可能であると判定されることになる。したがって、終端フラグEが

50

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。

【0196】

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。

【0197】

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、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。

10

【0198】

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。

20

【0199】

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13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。

【0200】

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。よって、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。

30

【0201】

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。

40

【0202】

(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。

【0203】

まず、ある命令区間の実行が検出されると、PCおよびReg.の値がRBに入力される。そして、RBにおいて、連想検索により、入力されたこれらの値と、RBのValueの列に登録されている命令区間先頭アドレスおよびレジスタ値とが比較される。ここで、RBの

50

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

【0204】

一方、入力されたPCおよびReg.の値に一致するものが、RBのValueの列に登録されている場合には、上記した連想検索動作と同様にして、次の一致比較すべき項目についての一一致比較が行われる。このようにして、RBおよびRFに登録されている入力パターンと、該当命令区間における入力パターンとの一致比較を継続していき、一致しない項目が生じた時点で、新たにノードを追加する形で、その一致しない項目についてRBおよびRFに登録が行われる。そして、登録すべき入力パターンの最後の項目が登録されたラインにおいて、RFの終端フラグを「E」とし、入力パターンの登録を終了する。

10

【0205】

入力パターンの登録が終了すると、終端フラグを「E」としたRFにおけるラインに対応する、RO1におけるラインに、出力値および出力アドレスの登録を行う。そして、出力値として登録すべき項目がRO1に登録しきれない場合には、ポインタを用いてRO2

20

【0206】

(第2の構成例における予測機構)

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は、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、RF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。

30

【0207】

また、図13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としての数行によって構成されることになる。

【0208】

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

40

【0209】

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。

【0210】

(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

50

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。

【0211】

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低いことと、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。SPARC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RISCプロセッサでも、同様の判断をすることによって、本発明

10

【産業上の利用可能性】

【0212】

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。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。

【図面の簡単な説明】

【0213】

【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RF/RBによって実現される再利用表を示す図である。

20

【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。

【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。

30

【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】RWと、RF・RBとの関係を示す図である。

【図10】ある命令区間が実行された場合のRBにおける実際の登録状況を示す図である。

【図11】ある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RBに登録された例を示す図である。

【図12】予測に基づいて、予測処理部がアドレスA2およびアドレスR5の値に関して予測を行った場合の、予測エントリとしてRBに記録される入力要素の状態を示す図である。

40

【図13】RF/RBの第2の構成例の概略を示す図である。

【図14】図13に示すRF/RBにおける連想検索動作の具体例を示す図である。

【図15】同図(b)は、図13に示すRF/RB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16】同図(b)は、図13に示すRF/RB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17】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、

50

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 18】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 19】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 20】命令区間の一例を示す図である。

【図 21】図 20 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。

【図 22】R B における実際の登録状況を示す図である。

【図 23】図 20 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 24】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】

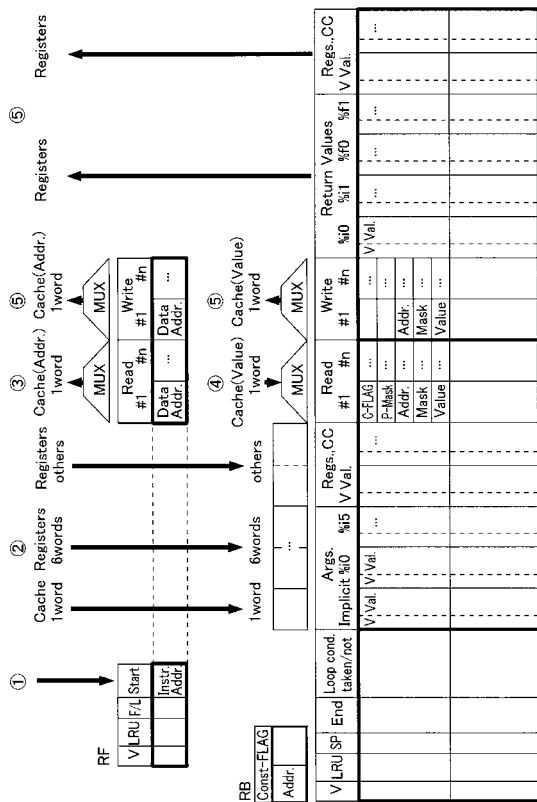
【0214】

- 1 A MSP
- 1 B SSP
- 2 RF/RB (入出力記憶手段)
- 2 A RB登録処理部 (区別処理手段)
- 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 RW
- 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 Cache

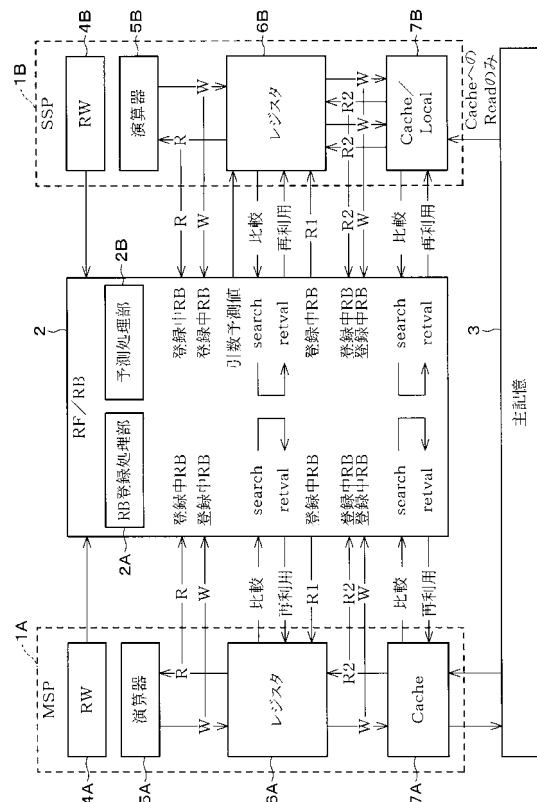
10

20

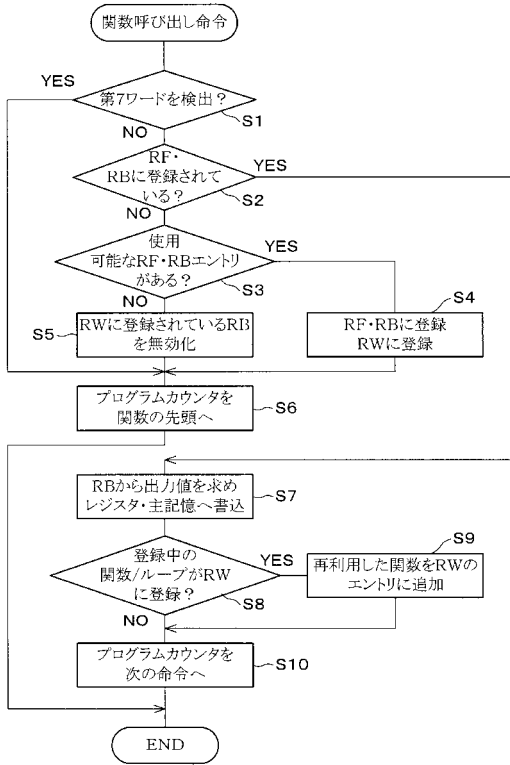
【図 1】



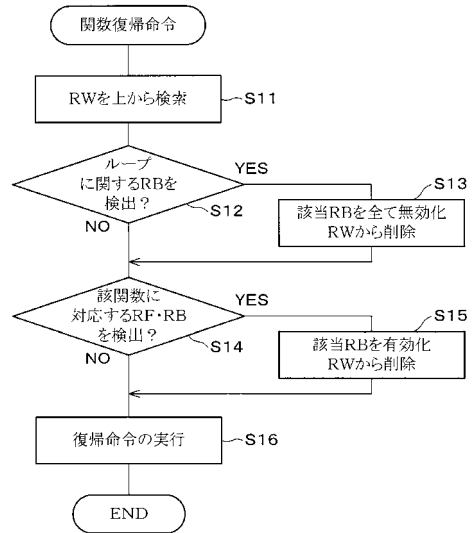
【図 2】



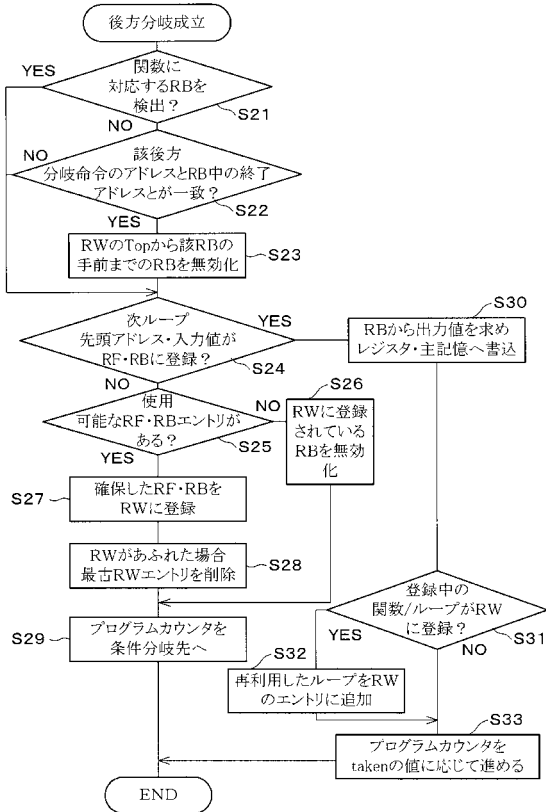
【図3】



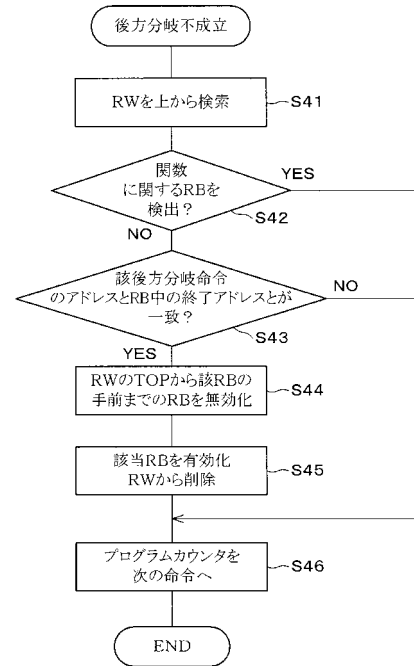
【図4】



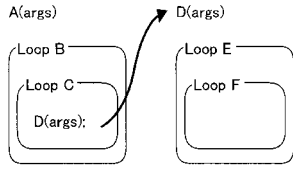
【図5】



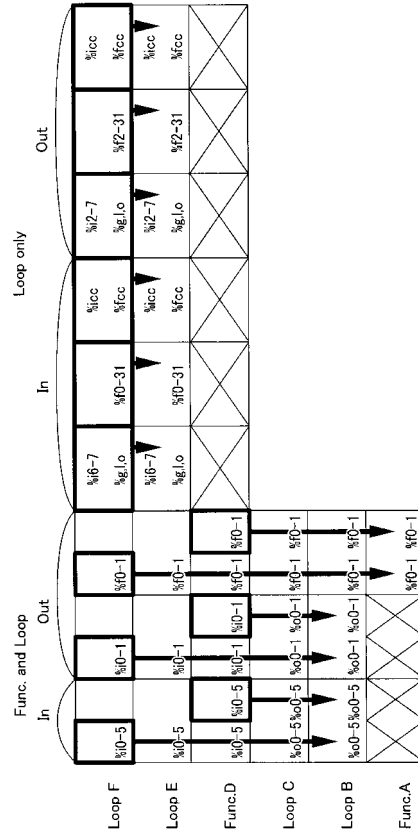
【図6】



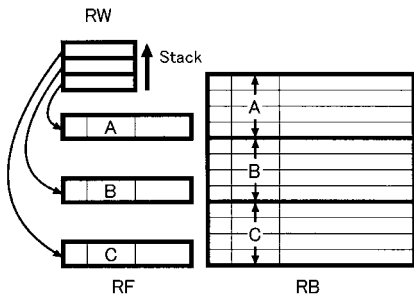
【 7 】



【 8 】



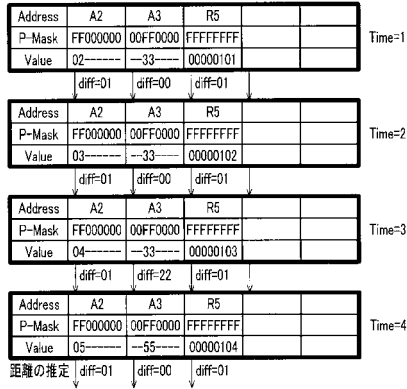
【 9 】



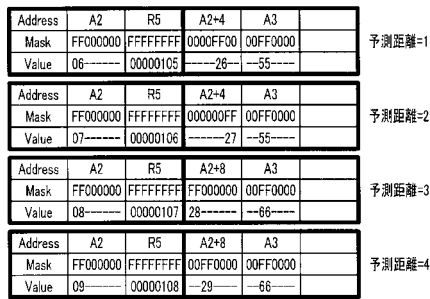
【 10 】

Const-FLAG		Register/Memory Read					Register/Memory Write								
R0	R1	R2	R3	R4	R5	#1	#2	#3	#4	#5	#1	#2	#3	#4	#5
1	0	0	0	0	0	C-FLAG	change	change							
0	0	0	0	0	0	P-Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF					
0	0	0	0	0	0	Address	A1	A2	A3	A4	R1	R2	R3	R4	R5
0	0	0	0	0	0	Mask	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
0	0	0	0	0	0	Value	00110001	02-27-1	-33-1	-44-1	00000100	00000032	00000033	00000044	00000101

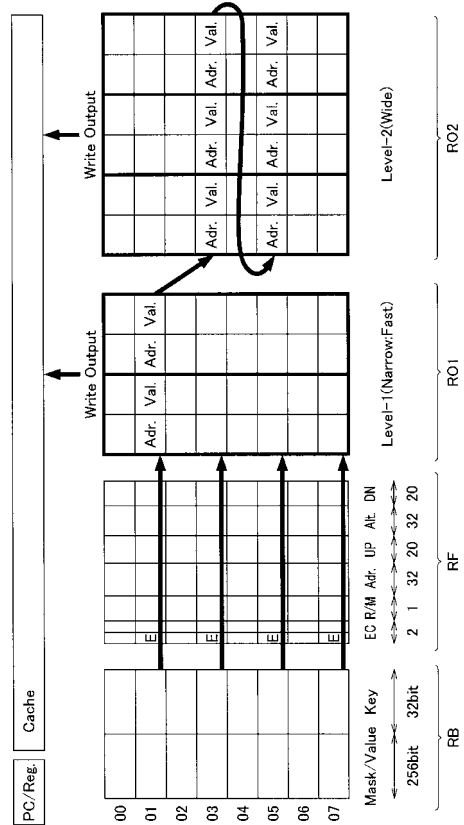
【 図 1 1 】



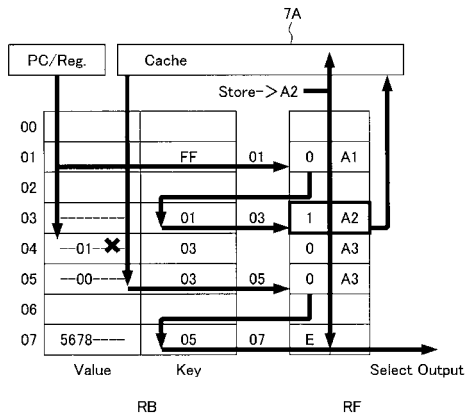
【 図 1 2 】



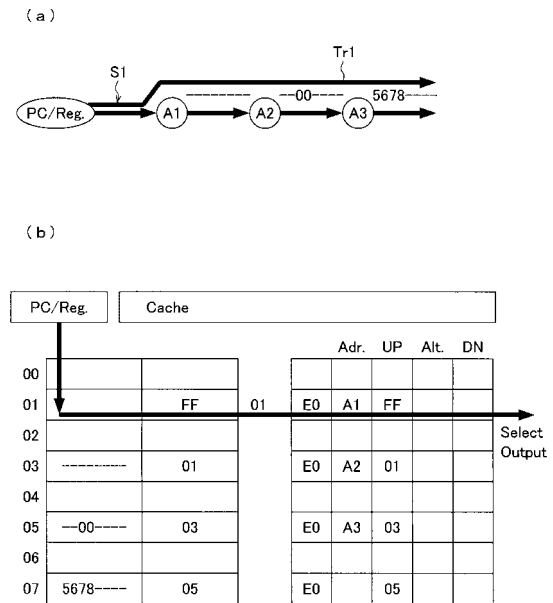
【 図 1 3 】



【 図 1 4 】



【 図 1 5 】

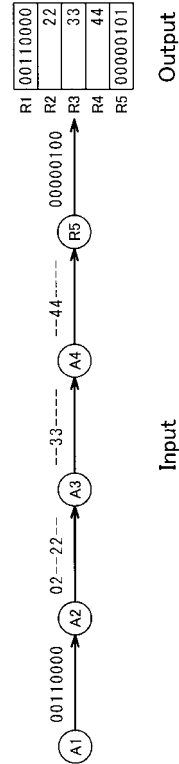


【 20 】

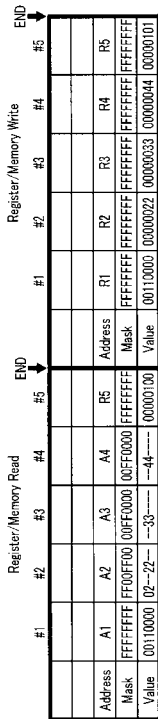
```

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
    
```

【 21 】



【 22 】



【 23 】

Address	A1	A2	A3	A4	R5	
Mask	FFFFFFF	FF00FF00	00FF0000	00FF0000	FFFFFFF	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	FFFFFFF	FF0000FF	00FF0000	00FF0000	FFFFFFF	Time=2
Value	00110000	03-23	-33	-44	00000102	
	diff=00	diff=??	diff=??	diff=11	diff=??	

Address	A1	A2	A2+4	A3	A4	
Mask	FFFFFFF	FF000000	00FF0000	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	FFFFFFF	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

【 24 】

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

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

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

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

フロントページの続き

- (56)参考文献 特開2004-258905(JP,A)
特開2005-284683(JP,A)
特開2002-312162(JP,A)
中島 康彦 他,「閾数値再利用および並列事前実行による高速化技術」,並列処理シンポジウム
2002,日本,社団法人情報処理学会,2002年 5月29日,第2002巻 第8号,Pages
269-276,ISSN:1344-0640
緒方 勝也 他,「閾数値再利用および並列事前実行による高速化技術の提案と評価」,情報処理
学会研究報告 Vol.2002 No.22 IPSJ SIG Notes,日本,社団法人情報処理学会
,2002年 3月 8日,第2002巻 第22号,Pages:163-168,ISSN:0919-6072
津邑 公暁 他,「並列事前実行機構における主記憶値テストの高速化」,情報処理学会論文誌
第45巻 No.SIG1(ACS4) IPSJ,日本,社団法人情報処理学会,2005年 1
月15日,第45巻 第SIG1(ACS4)号,Pages:31-42,ISSN:0387-5806
Youfeng Wu et al., "Better exploration of region-level value locality with integrated
computation reuse and value prediction", ACM SIGARCH Computer Architecture News, 米国
, ACM, 2001年 5月, Volume 29, Issue 2, Pages:98-108, ISSN:0163-5964

(58)調査した分野(Int.Cl., DB名)

G 0 6 F 9 / 3 8
G 0 6 F 9 / 4 0