(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3574019号
(P3574019)
(45)発行日　平成16年10月6日(2004.10.6)　(24)登録日　平成16年7月9日(2004.7.9)

(51)Int.Cl.[7]　F I
G06F 9/32　G06F 9/32 380
G06F 11/28　G06F 11/28 310E

請求項の数 5　(全 22 頁)

(21)出願番号　特願平11-307008
(22)出願日　平成11年10月28日(1999.10.28)
(65)公開番号　特開2000-148482(P2000-148482A)
(43)公開日　平成12年5月30日(2000.5.30)
審査請求日　平成13年8月2日(2001.8.2)
(31)優先権主張番号　186948
(32)優先日　平成10年11月5日(1998.11.5)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT－PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100081721
弁理士　岡田　次生
(72)発明者　ヴァサンス・バラ
アメリカ合衆国01776マサチューセッツ州サッドバリー、ハント・ロード　64

審査官　後藤　彰

(56)参考文献　特開昭63－91750（JP，A）
最終頁に続く

(54)【発明の名称】分岐履歴装置およびプログラムのプロファイル作成方法

(57)【特許請求の範囲】
【請求項１】
プロセスの分岐履歴を記録する装置であって、第１ロード信号に応答して命令アドレス値を受け取ってその命令アドレス値を記憶する第１トレース開始アドレス・レジスタと、分岐履歴値を記憶する第１分岐履歴レジスタであって、シフト信号に応答して当該分岐履歴値をシフトして当該第１分岐履歴レジスタの第１位置に分岐結果信号を記憶し、上記第１ロード信号に応答して上記分岐履歴値をクリアする第１分岐履歴レジスタと、第２ロード信号に応答して上記第１トレース開始アドレス・レジスタの上記命令アドレス値および上記第１分岐履歴レジスタの上記分岐履歴値を受け取ってそれらを記憶し、バッファ選択信号に応答してデータ・バスに上記分岐履歴値を出力するバッファと、命令信号を受け取り、そのような命令信号に応答して上記シフト信号、上記第１ロード信号および上記第２ロード信号を生成するコントローラであって、少なくとも所定の条件付き分岐命令が検出された時に上記シフト信号を生成し、トレース終了条件が検出されたときに上記第２ロード信号を生成するコントローラと、を備える分岐履歴記録装置。
【請求項２】
上記第１分岐履歴レジスタが、上記第１ロード信号に応答して上記第１位置にマーカーをセットし、上記第１分岐履歴レジスタの第２位置へのマーカーのシフト動作に応答してフル信号を生成し、上記コントローラが上記トレース終了信号または上記フル信号のいずれかに応答して上記第１ロード信号および上記第２ロード信号を生成する、請求項１に記載の分岐履歴記録装置。

【請求項３】
コンピュータ上で実行されるプログラムのプロファイルを作成する方法であって、第１アドレス・レジスタに命令アドレス値を記憶するステップと、条件付き直接分岐命令の条件に応答して第１分岐履歴レジスタの第１位置に分岐結果値をシフトさせるステップと、トレース終了条件に応答して上記第１アドレス・レジスタから第２アドレス・レジスタへ上記命令アドレス値を転送するステップと、上記トレース終了条件に応答して上記第１分岐履歴レジスタから第２分岐履歴レジスタへ分岐履歴値を転送するステップと、を含む方法。
【請求項４】
上記トレース終了条件が後方分岐命令値または間接分岐命令値のいずれかに応答して発生する、請求項３に記載の方法。
【請求項５】
プロセッサにおける実行行動を記憶する装置であって、プロセッサの命令レジスタに接続され、上記命令レジスターから受け取る第１のあらかじめ定められた命令値に応答して第１制御信号を生成し、上記命令レジスターから受け取る第２のあらかじめ定められた命令値に応答して第２制御信号を生成するように構成されるコントローラと、上記コントローラおよび上記プロセッサのプログラム・カウント・レジスタに接続され、上記第１制御信号に応答して上記プログラム・カウント・レジスタから受け取るプログラム・カウント値をラッチするように構成される第１レジスタと、上記コントローラおよび上記第１レジスタに接続され、上記第１制御信号に応答して上記第１レジスタのプログラム・カウント値をラッチするように構成される第２レジスタと、コントローラに接続され、上記プロセッサの分岐結果信号を受け取るように構成されるシフト・レジスタであって、上記第１制御信号に応答して該シフト・レジスタの内容をクリアし、上記第２制御信号に応答してその内容を第１位置から第２位置の方向へシフトさせ分岐結果信号の値をその第１位置へシフトさせるシフトレ・ジスタと、上記コントローラおよび上記シフト・レジスタに接続され、上記第１制御信号に応答して上記シフト・レジスタの内容をラッチするように構成される第３レジスタと、を備える装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、一般的に、コンピュータ上で実行するソフトウェア・プログラムの実行時プロファイルに関するものである。
【０００２】
【従来の技術】
実行時プロファイル機能は、プログラムの実行時動作を理解するためのメカニズムである。実行時プロファイルは、プログラムの制御の流れの経路、すなわちどの命令が実行されたか、実行においてどの分岐が発生したかを示す情報の集合である。プロファイルに基づく最適化は、命令スケジューリング、ループ・スケジューリング、データ事前ロード、インライン関数および命令キャッシュ性能強化のため使用することができる。
【０００３】
一層大きい性能利得の可能性のあるプログラム部分に解析の努力を集中するため、プログラムの実行時プロファイルが最適化コンパイラおよび動的変換機構によって使用される。先進的コンパイラは、命令レベルの並列性を増加させ、資源の使用度を高め、キャッシュ性能を向上させるため、ブロック境界を越えて最適化を実行する。プロファイル・データは、また、ソフトウェア開発者がそのプログラムの性能を調整する際に役立つ。
【０００４】
プログラム・プロファイル機能は、典型的には、プログラムの実行の間、イベントの発生数をカウントする。測定されるイベントは、典型的には、ルーチン、コード行または分岐のようなプログラムの局所的部分である。基本ブロックおよび制御の流れに基づいて一層洗練されたプロファイルの作成も可能である。プログラムのためのプロファイル情報は、

単純な実行カウント、あるいは、プログラムを実行しているコンピュータの範囲内のカウンタから集められる一層精巧な測定基準から構成されることができる。

【０００５】

プロファイル作成に対する１つの従来技術の手法は、プロファイル作成用プローブ（探り針）をプログラム・コードに加えることによってプログラム・コードに仕組みを作るものである。プロファイル作成用プローブは、そのプローブを含むコードの基本ブロックの実行を記録するための付加的命令である。典型的には、プログラムは、コードの各基本ブロックの範囲内に配置されたプロファイル用プローブを用いてコンパイルされる。次に、そのように仕組まれたコードが、いくつかのプロファイル・データのセットを得るため、異なるいくつかのテスト入力セットを使用する。その後、プログラムは、生成されたプロファイル・データを使用して再コンパイルされ、オリジナル・プログラムのプロファイルに基づくコンパイルされたプログラムが与えられる。

【０００６】

このような仕組みに基づいてプロファイル・データを収集する方法は、複雑であり、時間を要する傾向がある。そのようなコードの仕組みは、命令の追加のためコード・サイズが膨張する結果となることがある。プローブ命令の追加は、また、コードの実行を遅くし、プロファイル付きプログラム・バージョンの実行速度がオリジナル・バージョンに比較して３０倍も遅い場合もある。実行測度の低下は不便きわまりない。プロファイルに基づく最適化がユーザの間で広く使われていない主な理由が速度低下にあることは経験が示している。

【０００７】

仕組まれたプログラムに関する代表的テスト入力セットの選択は、プロファイル・データの正確度にとって重要である。入力が慎重に選択されないならば、プロファイルは実際の使用度を反映しない。例えばソート・ルーチンまたはデータベース・アプリケーションのように、データに高度に依存するプログラムは、ユーザ入力に高度に反応する分岐を持つ。ユーザの習慣を十分研究することなく、プロファイルの有効性を検証することは困難である。そのようなユーザ研究を行わない場合、プロファイル作成は、典型的には、正確なプロファイル・データを生成するために必要とされる時間を増大させる大きい入力セットを使用して行われる。

【０００８】

しかしながら、仕組に基づいたプロファイル作成に必要とされる時間を減少させるため、小さいテスト入力データのセットが使用されなければならない。しかし、小さいテスト入力セットは、生成されるプロファイル・データの正確度を減少させる。従って、プロファイル作成の正確度とプロファイル作成を実行するために必要な時間の間にトレードオフが存在する。

【０００９】

しかしながら、代表的テスト入力データに対応することが困難または不可能なプログラムが一部にはある。オペレーティング・システム（ＯＳ）のカーネルおよび組み込みシステムのようなリアルタイム・アプリケーションは、それらの実行特性によってプロファイルに基づく最適化の利益を受けることができない。データベース・システムのような長時間実行アプリケーションもプロファイル機能から除外される場合が多い。

【００１０】

更に、プロファイル・データの分析および使用は、付加的処理ステップを必要とする。プログラムは、実行できるようにされたプロファイル機能を使用してコンパイルされ、テスト入力セットを使用して実行され、次に、プロファイル・データを使用して再コンパイルされる。小さいプログラムの場合、この手順は大きなオーバーヘッドまたは困難を含まない。しかし、商用データベース・アプリケーションのような大型システムの場合、これは、構築スクリプトの大幅な修正を必要とする。多数の人手がこれらのスクリプトに投資される。それに加えて、大型システムは、解析および再コンパイルのため大量の処理時間を必要とする。結果として、ソフトウェア業者は、プロファイル型最適化の採用を躊躇する

。
【００１１】
プログラム・コードに仕組みを設けることに対する代替手法は、統計的プログラム・カウント（略してＰＣ）サンプリングを使用するものである。プロファイルが作成されつつあるコードの周囲で包み込みの役をするプロファイラ・プログラムの制御の下で、実際のプログラムが実行される。プロファイラ・プログラムは、オペレーティング呼び出しを行って、事前に定められた秒当たりＸ回という頻度でプロファイラ・プログラムに渡されるべきタイマ割込みをセットする。プロファイラ・プログラムは、また、オペレーティング・システムの範囲内のこの割り込みに関する”ハンドラ・プロシージャ”を登録する。次に、実際のプログラムの実行がテスト入力セットによって開始および駆動される。
【００１２】
タイマ割込みが発生すると、ハンドラ・プロシージャが起動され、実行プログラムは停止される。この時点において、マシンの状態（言い換えると、プロセスのプログラム・カウント）がプロファイラ・プログラムに記録のため渡される。また、ハンドラ・プロシージャが割り込みの時点におけるプロセッサのレジスタの多数の値を記録する場合がしばしばある。統計的ＰＣサンプリングのオーバーヘッドは、選択されているサンプリング頻度Ｘによって決定される。オーバーヘッドおよび速度はサンプリング頻度によって決定される。サンプリング頻度が少なくなれば、オーバーヘッドは減少し、速度は増加する。しかし、プロファイル・データの正確度もまたサンプリング頻度によって決定され、サンプリング頻度が多くなれば、正確度は増加する。従って、サンプリング頻度を選択する時オーバーヘッドと正確度の間にトレードオフが存在する。
【００１３】
更に、典型的には、上述の統計的ＰＣサンプリング手法はあまりに精細なレベルの細分性を生む。統計的ＰＣサンプリングは、また、制御の流れを実際に追跡しないので、コードを最適化するプロセスにおいてそれを使用するため高いレベルの解析を必要とする。プログラム・コードの最適化を実行するため、コードのどの部分がホットであるか（換言すれば頻繁に実行されるコード部分）を標示するプロファイル情報がプログラムの制御の流れに対応付けされる必要がある。プロファイル・データが任意の間隔でとられる多数のプログラム・カウント値に関係する時、これは困難である。また、必要とされる高いレベルの解析のため、プロファイル・データの解析は、通常、プログラムの実行の後、実行される。これは、動的リンク・ライブラリ（ＤＬＬ）の範囲内のプログラムの実行時制御の流れのような動的アドレス指定情報の一部が失われる可能性があるという欠点を持つ。それに加えて、実行後の解析の必要条件によって、動的最適化のための統計的サンプリングの使用が妨げられる。
【００１４】
代替的には、コンパイラの仮定に基づき、コードの仕組みまたは実行割り込みを介して得られるプロファイル・データの使用を必要とせず、コードの再コンパイルを必要としない静的方法が存在する。しかし、そのような静的推定は、典型的には、プロファイル機能ほど正確ではない。例えば、静的推定が分岐動作を予測するため使用される場合、その予測の不正確度は、プロファイル情報に基づく予測のほぼ２倍である。更に、動的に結合されたプロシージャの範囲内の制御の流れは静的に推定を行うのが難しい。
【００１５】
別の手法は、プロファイル作成の速度を速めるため、既存の分岐処理ハードウェアを使用するものである。オーバーヘッドを減らすハードウェアの使用は、統計的ＰＣサンプリングの場合のような一層低いプロファイル作成オーバーヘッドに関するトレードオフの必要性を克服する。ハードウェアの使用は、また、コードに必要とされる仕組みのレベルを減らし、仕組まれたプログラムにおいて発生するコード膨張および実行速度低下を回避する。
【００１６】
プログラムのプロファイルを統計的に作成するハードウェアに支援された方法は、典型的

には、そのプログラムの動作の分岐履歴を保持することを含む。基本ブロックの範囲内でとられた各分岐に関して１を、分岐が発生しなかった各分岐に関してゼロをそれぞれ分岐履歴バッファに記憶することによってコードのブロックにおける分岐動作の履歴を記憶するバッファを使用して、分岐履歴が取得される。

【００１７】

商用プロセッサにおいて既存の分岐処理ハードウェアを使用するハードウェア支援プロファイル作成手法の１例が、Ｃｏｎｔｅ、ＰａｔｅｌおよびＣｏｘ著”Ｕｓｉｎｇ Ｂｒａｎｃｈ Ｈａｎｄｌｉｎｇ Ｈａｒｄｗａｒｅ ｔｏ Ｓｕｐｐｏｒｔ Ｐｒｏｆｉｌｅ－Ｄｒｉｖｅｎ Ｏｐｔｉｍｉｚａｔｉｏｎ”（ＭＩＣＲＯ２７, Ｎｏｖｅｍｂｅｒ１９９４）に提唱されている。そこに記述されている方式は、分岐予測ハードウェアにおける使用に関して実行速度の低下はわずかに０．４％－４．６％でありながら高い精度を持つプロファイルを取得する。

【００１８】

既存の分岐予測ハードウェアにおいて分岐を予測するため、分岐命令のアドレスが分岐履歴の現在値と組み合わせられる。これは、最後のｋ個の分岐結果の大域分岐履歴であるか、あるいは、分岐毎履歴すなわち同じ分岐の最後のｋ個の結果を含むテーブルである。この結果の値を使用して、予測を読み取るため予測値テーブルへインデックスする。分岐の実際の実行の後、分岐（０／１）の結果が分岐履歴バッファにシフトされる。分岐履歴バッファは、実行するあらゆる分岐の結果を記録する大域バッファであるか、あるいは、同じ分岐の過去の履歴だけを記録する分岐毎バッファである。分岐履歴レジスタのビットは単にシフト送りされるだけで、レジスタが一杯か否かは検査されない。直接分岐だけが分岐予測ハードウェアによって取り扱われ、間接分岐は予測されない。

【００１９】

上記引用文献の方式は、分岐予測が非常に低いオーバーヘッドで実行中プログラムに関するプロファイル情報を取得することができるように典型的には最近のマイクロプロセッサにおいて使用される分岐予測ハードウェアを使用する、この方式は次のように動作する。（１）プロファイルが作成されるべきプログラムは、制御流れグラフ（すなわちＣｏｎｔｒｏｌ Ｆｌｏｗ Ｇｒａｐｈのことで以下ＣＦＧと略称される）の枝のテーブルによって強化される。ＣＦＧの１例が図１に示されている。枝はコード・ブロックＡ－Ｆの間で矢印として表されている。ＣＦＧ構成は、プログラムをコンパイルするコンパイラによって決定されるプログラムの静的制御の流れを表す。（２）実行の間、オペレーティング・システム・カーネルは、分岐予測バッファに記録されている分岐履歴情報を周期的に読み取り、それを使用してＣＦＧの枝に関連するカウンタを増分する。このプロセスは、ＣＦＧを加重制御流れグラフ（すなわちＷｅｉｇｈｔｅｄ Ｃｏｎｔｒｏｌ Ｆｌｏｗ Ｇｒａｐｈであって以下ＷＣＦＧと略称される）へ変換するものと見なされることができる。なぜなら、ＣＦＧの枝は、枝に関連するカウンタの値によって識別（または”加重”）されるからである。オーバーヘッドを低く保つため、ＣＦＧ枝カウントがメモリにおいて更新され、プログラムが実行を完了した後全体のＷＣＦＧがディスクに書き込まれる。

【００２０】

分岐予測ハードウェアは、典型的には、分岐命令アドレスによってインデックス付けされ、分岐動作の最近の履歴に関する情報を含むバッファから構成される。履歴情報を構成する多くの形態がある。例えば、各バッファ・エントリが、同じ分岐の前の結果（分岐五毎の履歴）のレコードを含むかもしれないし、または、各バッファ・エントリが、その分岐が実行された最後の時点にこの分岐の直前に先行した分岐のシーケンスの結果（大域分岐履歴）を含むこともできる。いずれの場合でも、この分岐履歴情報は、バッファ・エントリから取り出され、分岐の現在のインスタンスの結果を予測するために使用される。図２の（Ａ）は、分岐命令アドレスによってインデックス付けされる分岐目標バッファを構成する１つの形態を示す。再び、履歴情報を使用して予測を取得する方法はいくつかある。例えば、履歴値が分岐命令アドレスと組み合わせられ、その結果の値を使用して予測値テーブルをインデックスして現在時分岐命令に関する予測された結果が取得される。図２の

（Ｂ）は、予測値テーブル２４にインデックス付けするための分岐履歴を取得するため分岐命令アドレスでインデックス付けされる履歴レジスタ・テーブル２２の１例を示す。

【００２１】

一旦分岐命令が実際に実行されると、この分岐の実際の結果（０／１）を説明するように、分岐予測ハードウェアによって維持される分岐履歴情報が更新される。これは、典型的には、この分岐によってシフト・レジスタにインデックス付けされたバッファ・エントリから分岐履歴を取り出し、最後にこの分岐の結果をシフトし、新しい分岐履歴値をバッファ・エントリに記憶することによって行われる。図３は、３つのサンプル・バッファ・エントリを含む履歴レジスタ・テーブル２２を示している。

【００２２】

上記引用文献の方式において、オペレーティング・システムは、分岐予測ハードウェアのバッファに記録された情報をサンプリングする時、特定の分岐が実行された回数を見積もり、このカウントを分岐命令を表すＣＦＧの枝に関係づける。各分岐命令に対応する２つの可能なＣＦＧの枝が存在する。１つは（分岐履歴において１によって示される）分岐した方向に関するもので、他方は（分岐履歴においてゼロによって示される）分岐しなかった方向に関するものである。上記文献は、分岐履歴情報から１つの枝カウントを見積るためのいくつかの発見的手法を提案している。例えば、履歴の長さによって除算される分岐履歴における１の数が分岐が実行された回数の推定値を与える。

【００２３】

分岐履歴情報がそれが変わるより頻繁にサンプリングされるとすれば、枝の重みの数えすぎが発生する可能性がある。オペレーティング・システムによってサンプリングされる都度分岐履歴をすべてゼロにすれば、ゼロのエントリは分岐が実行されなかったことをも意味するので、問題を解決しない。上記文献によって提案されている解決策は、無効と有効分岐履歴の間の境界を示すため、図３に示されるように、先頭の”１”をマーカー・ビットとして使用するものである。オペレーティング・システムによって分岐履歴がサンプリングされた後、オペレーティング・システムは、履歴をゼロにし、分岐履歴の最下位ビット（ＬＳＢ）を１にセットする。その後、分岐履歴シフト送り論理が分岐履歴を更新する時、このビットが左へシフトする。マーカー・ビットがシフト・レジスタの最上位ビット（ＭＳＢ）位置に達する時を検出するため、何らかの付加的論理も必要とされる。上記文献は、マーカー・ビットがＭＳＢに達する状況が発生すると１にセットされる分岐履歴に関連する余分な”フル・ビット”を設けることを提案している。”フル・ビット”が１であれば、履歴の内容のすべてが有効であることを意味する。図３は、１に設定された”フル・ビット”を持つバッファ・エントリを図示している。しかしながら、（フル・ビットを除き）履歴の内容は左へシフトされ続け、最も左のビットが送り出される。

【００２４】

上記文献の手法の欠点は、分岐予測ハードウェアによって維持される分岐履歴情報が、プロセッサ上で実行するすべてのプログラムによって共有され、プロファイルが作成されているプログラムの状態の一部ではないことである。このように、同じプログラムの異なる分岐が同じ分岐履歴エントリにマップされるだけではなく、異なるプログラムにおける分岐もまた同じ分岐履歴エントリにマップされる。従って、分岐履歴におけるビットは、現在実行しているプログラムのいずれかにおける任意の分岐の結果に対応する可能性がある。上記文献は枝のカウントを見積ることにのみ関心があるので、これは、カウントの正確度を減らすだけで、その方式の完全性に影響を及ぼさない。しかし、この分岐履歴情報は、実行時にプログラムによって実行される分岐命令の実際のシーケンスを再構築するために使用することができない。これは、分岐履歴がプログラムの状態を実行することの一部として保持され、文脈切り替えの間オペレーティング・システムによって保存および復元される場合にのみ可能である。更に、上記文献の手法の場合プログラムにおける頻繁な実行経路を決定する唯一の方法は、ＷＣＦＧを分析して、最高の重みを持つ枝を特定して、それらを共に結合してトレースを形成することを試みることである。ＷＣＦＧを処理するために要求される高レベルの分析は、プログラムが実行中の適用を高価なものにする。

【００２５】
上記文献のプロファイル作成手法のもう１つの欠点は、プログラムが、ＣＦＧ構成をもちいて強化することによって、本質的に”仕組み”を備えなければならないことである。それに加えて、間接分岐（すなわち分岐の異なる実行のため目標が異なる可能性のある分岐）を取り扱うことができない。間接分岐は、プロファイル作成のため、間接分岐を条件つき直接分岐のシーケンスへコンパイラが変換することを求める。このような両方の問題が、遺産として存在するプログラム・バイナリ（すなわち再コンパイルできない）プログラム対してこの方式を使用することができないものにさせている。
【００２６】
ここで、図４の簡略化されたマイクロプロセッサのアーキテクチャ１００を使用して、条件付き直接分岐の動作を例示する。次ＰＣ論理１２０によってプログラム・カウント値がプログラム・カウント・レジスタ１２４にロードされる。命令を取得するため、メモリにアクセスするアドレス・バス上にプログラム・カウント値が出力される。プログラム・カウント値に対応する命令が、命令レジスタ１３８へのロードのためデータ・バス上に送られる。次に、命令は命令デコーダ１３４によって復号され、プロセッサに関するタイミング／制御論理１３０へ入力される。命令が分岐コマンドである場合、分岐目標アドレスもデータ／アドレス・レジスタ１２８へロードされる。
【００２７】
タイミング／制御論理１３０は、プロセッサのその他の機能ブロックを駆動するタイミング／制御信号を生成する。例えば、タイミング／制御論理１３０は、処理のため算術論理ユニット（ＡＬＵ）１４０へのオペランドとしての入力のため、レジスタ・ファイル１４４における１つまたは複数のレジスタの内容を選択する。タイミング／制御論理１３０は、また、次ＰＣ論理１２０を駆動して、次のプログラム・カウント値を選択してそれをプログラム・カウント・レジスタ１２４へロードさせる。
【００２８】
タイミング／制御論理１３０は、命令復号論理１３４によって復号される命令およびＡＬＵ１４０によって生成される条件フラグＮ、ＺおよびＣの状態に応答して、タイミング／制御信号を生成する。ＡＬＵ１４０は、そのアキュムレータに非ゼロの値を検出すると、非ゼロ・フラグＮをセットする。同様に、ＡＬＵ１４０は、そのアキュムレータにゼロの値を検出すると、ゼロ・フラグＺをセットする。ＡＬＵ１４０によって実行された演算が桁上げ条件を発生する場合、桁上げフラグＣがセットされる。
【００２９】
タイミング／制御論理１３０は、分岐信号の状態を決定するため、命令復号論理１３４からの情報を用いて条件フラグＮ、ＺおよびＣを統合する。例えば、ゼロ条件で分岐する命令によって、Ｚフラグが活動状態であればタイミング／制御論理１３０は活動的分岐信号を生成することとなる。次に、分岐信号によって、次ＰＣ論理１２０が、（分岐コマンドの命令レジスタ１３８へのロードと共に、目標アドレスがロードされている）データ／アドレス・レジスタ１２８からプログラム・カウント・レジスタ１２４に分岐目標アドレス値をロードすることとなる。この結果、実行制御の流れは目標アドレスへ進む。Ｚフラグが活動的でなければ、分岐信号は非活動状態のままで、分岐動作は実行されず、次ＰＣ論理１２０はプログラム・カウント値を増分して、実行シーケンスにおける次の命令を取得する。図４のプロセッサ・アーキテクチャは、プロセッサ・アーキテクチャの１つの簡略化された例である。一層複雑な次ＰＣ機能、命令復号機能および分岐条件を伴うその他のアーキテクチャも存在する。
【００３０】
【発明が解決しようとする課題】
このように、プログラムのコードに特別の仕組みを組み込むことなく、実行中プログラムに関するプロファイル・データを収集する手段が必要とされている。
【００３１】
【課題を解決するための手段】

プログラムのコードに仕組みを組み込むことなく、実行中プログラムに関するプロファイル・データを収集する方法を提供することが本発明の目的の１つである。本発明の別の目的は、コードの実行時性能への影響を最小限にとどめながらプロファイル・データを収集することである。本発明の更に別の目的は、単純で安いハードウェアを使用してプロファイル・データを収集することである。本発明の更にまた別の目的は、詳細な分析を必要とすることなく、単純明快な方法で、プロファイル・データが実行プログラムの制御の流れと動的に関連づけられることを可能にすることである。

【００３２】

本発明は、プロセスの分岐履歴を記録する装置を提供する。該装置は、命令アドレス値を受け取って第１ロード信号に応答して命令アドレス値を記憶する第１トレース開始アドレス・レジスタを含む。第１分岐履歴レジスタが、分岐履歴値を記憶する。第１分岐履歴レジスタは、シフト信号に応答して分岐履歴値をシフトさせ分岐結果信号を第１分岐履歴レジスタの第１位置に記憶する。第１分岐履歴レジスタは第１ロード信号に応答して分岐履歴値をクリアする。バッファが、第２ロード信号に応答して、第１トレース開始アドレス・レジスタの命令アドレス値および第１分岐履歴レジスタの分岐履歴値を受け取りそれらを記憶し、バッファ選択信号に応答してデータ・バスに分岐履歴値を出力する。制御論理が命令信号を受け取り、命令信号の第１のあらかじめ定められた命令値に応答してシフト信号を生成し、命令信号の第２のあらかじめ定められた命令値に応答して第１ロード信号および第２ロード信号を生成する。

【００３３】

本発明は、また、コンピュータ上で実行されるプログラムのプロファイルを作成する方法を提供する。該方法は、トレース終了条件に応答して第１アドレス・レジスタに命令アドレス値を記憶し、トレース終了条件に応答して第１分岐履歴レジスタにおける分岐履歴値をクリアし、条件つきの直接分岐条件に応答して分岐履歴レジスタの第１位置に分岐結果値をシフトすることを含む。該方法は、トレース終了条件に応答して第１アドレス・レジスタから第２アドレス・レジスタへ命令アドレス値を転送し、トレース終了条件に応答して第１分岐履歴レジスタから第２分岐履歴レジスタへ分岐履歴値を転送するための呼び出しを行う。

【００３４】

本発明は、更に、プロセッサにおける実行行動を記録する装置を提供する。該装置は、プロセッサの命令レジスタに接続されたコントローラを備え、該コントローラは、命令レジスタから受け取った第１のあらかじめ定められた命令値に応答して第１制御信号を生成し、命令レジスタから受け取った第２のあらかじめ定められた命令値に応答して第２制御信号を生成するように構成される。第１レジスタが、コントローラおよびプロセッサのプログラム・カウンタに接続され、第１制御信号に応答してプログラム・カウンタから受け取ったプログラム・カウント値をラッチするように構成される。第２レジスタが、コントローラおよび第１レジスタに接続され、第１制御信号に応答して第１レジスタのプログラム・カウント値をラッチするように構成される。シフト・レジスタは、コントローラに接続され、プロセッサの分岐結果信号を受け取るように構成され、更に、第１制御信号に応答してその内容をクリアするように構成される。シフト・レジスタは、また、第２制御信号に応答してシフト・レジスタの内容をシフト・レジスタの第１位置から第２位置の方向へシフトし、第１位置に分岐結果信号の値を記憶するように構成される。第３レジスタが、コントローラおよびシフト・レジスタに接続され、第１制御信号に応答してシフト・レジスタの内容をラッチするように構成される。

【００３５】

【発明の実施の形態】

本発明は、図４のプロセッサのようなプロセッサ上でプログラムが実行される間のプログラムの制御の流れの動作の統計的トレース・サンプリングを対象とする。１つのトレースは、”トレース終了条件”が真である時に終了する動的分岐シーケンスである。トレース終了条件の例は、分岐目標アドレスが当該分岐命令のプログラム・カウントより低いプロ

グラム・カウントである場合のように後方に向かう分岐、あるいは、分岐目標が分岐のそのインスタンスを実行することによってのみ決定される（すなわち目標は分岐命令自体の復号によって決定されることができない）間接分岐である。

【００３６】

また、トレースに関する分岐履歴のサイズに制限を設けて、そのような制限に到達する時にトレース終了条件を生成することによって、トレース・サンプリングに伴うオーバーヘッドを制約することには利益がある。分岐履歴制限は、トレースにおいて許容される条件つき直接分岐の最大数である。分岐履歴制限は、統計的トレース・サンプリングに関係するオーバーヘッドを制限するため、トレースのサイズを制約する機能を持つ。

【００３７】

レース終了条件は、分岐が発生する時点で評価され、その時点でのコードの動作に基づく。後方分岐は、トレースの終了を標示し、分岐の目標アドレスが分岐命令アドレスより小さい時発生する。間接分岐は、分岐目標がレジスタの値によってプロセッサの範囲内に決定される場合発生し、従ってマシンの状態に依存する。間接分岐の重要な特性の１つは、同じ分岐命令の実行の各々ごとに分岐の目標が異なる可能性があることである。

【００３８】

本発明の１つの実施形態において、サンプリングの時点で完了していない”現在実行中トレース”のみならず、”最後に実行されたトレース”も追跡するため、ハードウェア補助機構が使用される。オペレーティング・システムまたは特権ソフトウェアは、ハードウェア補助機構レジスタから最後に実行されたトレースの情報を読み取ることができる。この方式のその他の部分は従来技術の統計的ＰＣサンプリングと同様である。プログラム・コードがプロファイラ・プログラムの制御の下で実行され、プロファイラ・プログラムは、オペレーティング・システム呼び出しを行ってタイマ割込みを設定し、割込みハンドラ・プロシージャを登録する。タイマ割込みが発生すると、オペレーティング・システムはハンドラを起動して、割り込みの瞬間のマシン状態および最後に実行されたトレースのデータをハンドラに与える。ハンドラはこの情報および関連ハードウェア・カウンタを記録する。このように、ハードウェア補助機構は、実施のコストが安く、プロセッサ・サイクル・タイムに対する影響を持たない。

【００３９】

本発明に従ったトレース収集論理機構５００の１つの実施形態が図５に示されている。トレース収集論理は、図４のプロセッサ１００の次ＰＣ論理１２０の出力１２２に接続される現在時トレース開始アドレス・レジスタ５６０を含む。現在時トレース開始アドレス・レジスタ５６０は、次ＰＣ値をロードするためのロード制御入力を持つ。現在時トレース開始アドレス・レジスタ５６０は、また、最後のトレース開始アドレス・レジスタ５７０に接続される出力を持つ。

【００４０】

最後のトレース開始アドレス・レジスタ５７０は、ロード制御入力を持ち、それによって、第２ロード信号５２４に応答して、最後のトレース開始アドレス・レジスタ５７０に現在時トレース開始アドレス・レジスタ５６０の内容がロードされる。最後のトレース開始アドレス・レジスタ５７０は、また、選択制御入力を持ち、それによって、図４のタイミング／制御論理１３０から受け取った選択信号１６２Ａに応答して最後のトレース開始アドレス・レジスタ５７０の内容がデータ・バス上に送り出される。

【００４１】

現在時分岐履歴レジスタ５８０は、プロセッサ１００のタイミング／制御論理１３０によって生成される分岐信号１６４に接続されるシフト・データ入力を持つシフト・レジスタを含む。最後の分岐履歴レジスタ５９０が現在時分岐履歴レジスタ５８０の出力に接続される。現在時分岐履歴レジスタ５８０は、また、シフト制御入力およびクリア／セット制御入力を持つ。シフト制御入力によって、レジスタの内容がレジスタの最上位ビット位置の方向へシフトされ、シフト・データ入力の値がレジスタの最下位ビット位置に書き込まれる。シフト制御入力はトレース・コントローラ５２０によって生成されるシフト信号５

２６によって駆動される。クリア／セット制御によって、レジスタ５８０の内容がクリアされ、有効な分岐履歴がレジスタの内容の範囲内で開始する位置を標示するマーカーとして最下位ビットがセットされる。クリア／セット制御は、トレース・コントローラ５２０から第１ロード信号５２２を受け取る。

【００４２】

現在時分岐履歴レジスタ５８０は、また、現在時分岐履歴レジスタ５８０が分岐履歴制限に達したことを標示するフル信号５８２を出力する。本実施形態において、フル信号はレジスタ５８０の最上位ビット（ＭＳＢ）位置のビット値に対応する。マーカー・ビットがレジスタのＭＳＢにシフトされると、フル信号が活動的となる。

【００４３】

最後の分岐履歴レジスタ５９０は、現在時分岐履歴レジスタ５８０の出力に接続される入力、および、トレース・コントローラ５２０からの第２ロード信号に応答して現在時分岐履歴レジスタ５８０の内容が最後の分岐履歴レジスタ５９０にロードされることを可能にするロード制御入力を持つ。最後の分岐履歴レジスタ５９０は、また、図４のタイミング／制御論理１３０から選択信号１６２Ｂを受け取る選択制御入力を持つ。この信号によって最後の分岐履歴レジスタ５９０の内容がデータ・バス上に送り出される。

【００４４】

トレース・コントローラ５２０は、プロセッサ１００の命令レジスタ１３８の出力１４２に接続される。トレース・コントローラ５２０は、トレース終了条件または条件付き分岐条件を検出するため、命令レジスタ１３８に記憶されている命令を復号する。後方分岐または間接分岐のようなトレース終了命令が検出されると、トレース・コントローラ５２０は、先ず、現在時トレース開始アドレス・レジスタ５６０の内容を最後のトレース開始アドレス・レジスタ５７０へ転送し、現在時分岐履歴レジスタ５８０の内容を最後の分岐履歴レジスタ５９０に転送するため、第２ロード信号５２４を切り換える。次に、第１ロード信号５２２が切り換えられて、次ＰＣ値が現在時トレース開始アドレス・レジスタ５６０へロードされ、現在時分岐履歴レジスタ５８０の分岐履歴値がクリアおよびセットされる。

【００４５】

条件つき直接分岐条件がトレース・コントローラ５２０によって検出されると、トレース・コントローラ５２０はシフト信号５２６を切り換える。シフト信号によって、レジスタ５８０の分岐履歴値がＭＳＢの方向へシフトされ、分岐命令の結果を標示する分岐信号値がレジスタのＬＳＢにロードされる。シフト信号５２６が現在時分岐履歴レジスタ５８０のＭＳＢへのマーカー・ビットの移動を引き起こすとすれば、フル信号１８２が活動状態となり、トレース終了イベントが引き起こされる。従って、トレース・コントローラは、フル信号５８２に応答して、最初に第２ロード信号５２４を切り換え、次に第１ロード信号５２２を切り換える。

【００４６】

現在時トレース開始アドレス・レジスタ５６０、最後のトレース開始アドレス・レジスタ５７０、現在時分岐履歴レジスタ５８０および最後の分岐履歴レジスタ５９０は、別個の専用レジスタであり、従来技術のマイクロプロセッサに存在する既存の分岐予測ハードウェアの一部ではない。現在時トレース開始アドレス・レジスタ５６０および現在時分岐履歴レジスタ５８０はプロファイルが作成されるプログラムの状態の一部となるが、文脈データ構造へ２つの対応するフィールドが加えられる。加えて、最後のトレース開始アドレス・レジスタ５７０および最後の分岐履歴レジスタ５９０は、プロセッサのアーキテクチャの一部となり、独立してアドレス可能な機構である。

【００４７】

本発明は、フル信号１８２がトレース終了イベントとして検出され、それによって、現在時トレース開始アドレスおよび現在時分岐履歴が、最後のトレース開始アドレス・レジスタ５７０および最後の分岐履歴レジスタ５９０にコピーされることが可能とされることを必要とする。対照的に、上記引用文献の方式が利用する分岐履歴レジスタ・テーブルにお

ける各エントリに関するフル・ビットは、分岐履歴エントリが一杯となる時セットされ、リセッされるまでセットされたままとされる。従来技術の分岐予測論理における分岐履歴バッファは、典型的には、数百の分岐履歴エントリを含む。フル・ビットがセットされた後でさえ、上記文献の分岐履歴はシフトを続行し、その結果、分岐履歴エントリの最上位ビットが失われる。このように、本発明は、完全な分岐履歴が保存されるので、実行経路が完全に再構築されることを可能にする。

【００４８】

図６には、図５のトレース・コントローラ５２０の１つの実施形態が示されている。分岐命令復号器５３０が、図４の命令レジスタ１３８の出力１４２に接続される。分岐命令復号器５３０は、条件つき直接分岐コマンドのセットのうちのいずれかの１つに応答して条件つき直接分岐信号５３２を生成する。トレース終了コマンドのセットのいずれか１つに応答して終了条件信号５３４が分岐命令復号器５３０によって生成される。

【００４９】

制御ユニット５４０が条件つき直接分岐信号５３２に応答してシフト信号５２６を生成する。制御ユニット５４０は、また、終了条件信号５３４またはフル信号５８２に応答して、第１および第２ロード信号５２２および５２４を生成する。第２ロード信号５２４は、第１ロード信号５２２が切り換えられる前に、切り換えられる。注意すべき点であるが、図４に示されるプロセッサ１００のタイミング／制御論理１３０によって生成されるタイミング／制御信号の生成と第１ロード信号、第２ロード信号およびシフト信号の生成を同期させるため、制御ユニット５４０はプロセッサ・クロック信号を受け取る。

【００５０】

注意すべき点であるが、プロセッサＩ００の命令復号器１３４から適切な情報が利用できるとすれば、命令復号器１３４の出力を使用して、トレース終了および条件つき直接分岐条件を検出することができ、分岐命令復号器５３０によって実行されるトレース・コントローラ５２０の命令復号機能は必要とされない。更に注意すべき点であるが、最後のトレース開始アドレス・レジスタ５７０および最後の分岐履歴レジスタ５９０がそれぞれ同じロード信号に応じて出力信号をラッチすることができるほど十分な時間現在時トレース開始アドレス・レジスタ５６０および現在時分岐履歴レジスタ５８０がそれらの出力を保持していれば、第１および第２ロード信号５２２、５２４を単一信号にまとめることが可能である。本発明の理念を逸脱することなく本実施形態に対する上述およびその他の修正が可能である点は当業者によって理解されることであろう。

【００５１】

本発明の統計的トレース・サンプリング方法は、開始アドレスおよび分岐履歴のペアとしてコード・トレースを符号化する。開始アドレスは、トレースが開始するプログラム・カウントまたは命令アドレスである。現在時トレースに関してトレース終了イベントが発生する時次のトレースのためのトレース開始アドレスが取得される。この際、図５のトレース・コントローラ５２０によって第１ロード信号５２２が切り換えられ、図４のプロセッサ１００の次ＰＣ論理１２０の出力１２２における次ＰＣ値が現在時トレース開始アドレス５６０にロードされる。これによって、プログラム・カウンタ１２４へロードされているものと同じプログラム・カウント値が現在時トレース開始アドレス５６０へロードされる。

【００５２】

同時に、第１ロード信号５２４が現在時分岐履歴レジスタ５８０をクリアして５８０レジスタのＬＳＢにマーカー・ビットをセットする。この状況が図７に示されている。図７に示されているように、コード・アドレス”ＡＡＡ”が現在時トレース開始アドレス・レジスタ５８０へロードされ、従ってテーブルにおいて１つのエントリとして示されている。（当然、ラベル”ＡＡＡ”は説明の目的のためここで使用されているもので、実際のアドレス値はバイナリ数である）。

【００５３】

現在値分岐履歴値がその初期化された状態にある。この例では、初期値は”０００００

０１”であるが、どのようなビット数でも良い。現在値分岐履歴レジスタ５８０のＬＳＢ位置における”１”という値は、有効な分岐履歴ビットが開始する時を標示するマーカー・ビットを表す。言い換えると、分岐履歴値における先行ゼロの右のビットは有効な分岐履歴を表す。現在時トレース開始アドレス・レジスタ５６０および現在時分岐履歴レジスタ５８０にロードされていた最後のトレース開始アドレス値”ＸＸＸ”および最後の分岐履歴値”００１００１０１”は、それぞれ、最後のトレース開始アドレス・レジスタ５７０および最後の分岐履歴レジスタ５９０に転送されている。

【００５４】

”Ｚが真であればＢＢＢへ分岐”という条件つき直接分岐命令に出会うまで、トレースのコード・セグメントが実行される。この条件付き直接分岐命令は、シフト信号５２６を切り換えるトレース・コントローラ５２０によって復号される。この例においては、プロセッサ１００のＺフラグが真であり、分岐が実行される。従って、分岐信号１６４が真で、シフト信号に応答して、論理”１”が現在時分岐履歴レジスタ５８０のＬＳＢへロードされる。従って、分岐履歴値は”００００００１１”となる。次に、制御の流れはアドレス”ＢＢＢ”に分岐し、そこで、順次実行が再開して、”Ｎが真であればＣＣＣへ分岐”という条件つき直接分岐命令に出会うまで、順次実行が続く。しかし、この例においてはＮフラグは真ではないので、分岐はとられない。従って、条件つき直接分岐命令に応答してトレース・コントローラ５２０がシフト信号５２６を切り換える時、分岐信号１６４は偽であり、論理的”０”が現在時分岐履歴レジスタ５８０のＬＳＢに書き込まれる。従って、分岐履歴値は”０００００１１０”となる。

【００５５】

”Ｃが真であればＤＤＤへ分岐”という条件つき直接分岐命令に出会うまで、制御の流れは順次続く。この例ではＣは真であり、分岐が実行される。かくして、分岐履歴値は”０００００１１０１”となり、制御の流れはアドレス”ＤＤＤ”の命令から順次進む。命令”ＷＷＷへ分岐”は間接分岐を表し、その結果、トレース・コントローラ５２０によって検出されるトレース終了条件が発生する。トレース・コントローラ５２０は、先ず、第２ロード信号５２４を切り換え、それによって、現在時トレース開始アドレス５６０にあるアドレス値”ＡＡＡ”が最後のトレース開始アドレス５７０にロードされる。同時に、現在時分岐履歴レジスタ５８０にある分岐履歴値”０００００１１０１”が最後の分岐履歴レジスタ５９０へロードされる。次に、トレース・コントローラ５２０は、間接分岐命令の目標アドレスである次ＰＣ値”ＷＷＷ”を現在時トレース開始アドレス・レジスタ５６０にロードするため第１ロード信号５２２を切り換え、次のコード・トレースの取得に備えて現在時分岐履歴レジスタ５８０を”０００００００１”に初期化する。

【００５６】

この時点で、最後のトレース開始アドレス・レジスタ５７０および最後の分岐履歴レジスタ５９０の内容がソフトウェアにとって利用可能となる。これらのレジスタの各々はプロセッサ・アーキテクチャにとって視認でき、それらの内容を取得するためアドレス指定できるものである。このように、最後のトレース開始アドレス・レジスタ５７０へのアクセスのための命令によって、プロセッサ１００のタイミング／制御論理１６０によって選択信号１６２Ａがオンにされ、レジスタの内容”ＡＡＡ”がプロセッサ・データ・バス上に出力される。同じように、最後の分岐履歴レジスタ５９０にアクセスする命令によって、選択信号１６２Ｂがオンにされ、分岐履歴値”０００００１１０１”がプロセッサ・データ・バスへ出力される。

【００５７】

上記記述からわかるように、分岐履歴はビット列であって、各ビットが、トレースにおける条件つき直接分岐の出現を示し、分岐が行われたか否かを表す。分岐終了条件に到達すると分岐履歴は終了する。しかしながら、先行ビットが左端の位置に達すると、すなわち、マーカー・ビットが本実施形態の現在時分岐履歴レジスタ５８０のＭＳＢ位置に達すると、それは分岐履歴制限に達したことを意味するので、フル信号５８２がオンにされ、トレース終了が発生する。

【００５８】
プロファイル作成情報を収集するため、プロファイルされるべきコードを実行しているオペレーティング・システムにおいてプロファイル機能が起動されなければならない。一旦プロファイル機能が起動されれば、データの収集に関して実行されるべき多数の手法が存在する。最初に、トレースが終了するたびに割り込みを生成することによって、すなわち、各トレース終了条件に応答して割り込みを生成することによって各トレースの分岐履歴データを取得することができる。割り込み毎に、終了したばかりのトレースに関する分岐履歴データが記録され、高度に正確で完全なプロファイルが作成される。しかしながら、トレース終了による割込みが頻繁となり、割り込みの各々を取り扱うため高レベルのオーバーヘッドが生じる可能性がある。

【００５９】
代替的に、プロファイル作成のオーバーヘッドを減少させることがプロファイル・データの正確度より重要であれば、間欠的なトレース・プロファイル作成機能を使用すれば、オーバーヘッドを大幅に減少することができる。これは、統計的ＰＣサンプリングと同様に、ソフトウェアによって完全に実施できる。ＯＳが一定の頻度でタイマ割込みを駆動し、割り込みが発生すると、ＯＳが、処理ルーチンを使用して、プロファイルを作成しているプログラム環境から処理ルーチンへトレース・データを読み出す。代替的に、ＯＳのタイマ割込みの代わりに、サンプリング頻度カウンタの形態のハードウェア補助機構を使用する。このカウンタは、対象プログラムの実行の前に１つのカウント値に初期化される。サンプリング頻度カウンタは、トレース終了イベント毎に減分される。例えば、サンプリング頻度カウンタを減分するため、図５のトレース・コントローラ５２０によって生成される第１ロード信号５２２を使用することができる。カウンタがゼロに達すると、割り込みが生成され、トレース・データが記録される。割り込み時にトレース・データを記録した後、初期化の際カウンタにロードされたカウントの値を下げることによって、サンプリング頻度を調整することができる。

【００６０】
割込み処理ルーチン８００に関する流れ制御の１つの実施形態が図８に示されている。上述のように生成される割り込みに応答して、ステップ８１０において、ソフトウェア割込みハンドラが、図５に示される最後のトレース開始アドレス・レジスタ５７０および最後の分岐履歴レジスタ５９０からそれぞれ開始アドレスおよび分岐履歴値を読み取る。次に、ステップ８２０において、開始アドレス値に対応するエントリを求めてテーブルが検索される。１つの手法は、ハッシュ・テーブルを構築して、当該開始アドレスおよび分岐履歴に関する対応するエントリを検出するため開始アドレスをハッシュするものである。

【００６１】
ステップ８３０においてテーブル・エントリが検出されなければ、制御をステップ８３６へ分岐し、そこで、当該開始アドレスおよび分岐履歴に関するテーブル・エントリが作成され、テーブルにおけるトレース・データすなわち典型的にはカウンタが初期化される。エントリが検出されれば、ステップ８３４において、テーブルにおけるトレース・データが更新される、すなわち、カウンタが増分される。この時点において、実行中コードに関する統計的プロファイル作成のみが要求されているとすれば、制御は実行中コードへ戻り、ハッシュ・テーブルにおけるデータは実行の後処理されることができる。しかしながら、本発明の利点の１つは、プロファイル・データを実行の間に直接使用して実行中コードを最適化することができる点である。この機能は、コードを実行しながらその実行中コードを動的に最適化または変換する点において特に価値がある。

【００６２】
プロファイル・データの実行時処理を実現するため、ステップ８４０において、テーブル・エントリに関するトレース・データがしきい値に達したか否か検査される。データまたはカウンタ値がこのあらかじめ定められたしきい値を超えたならば、コードの実行へ戻る前に、ステップ８５０において、トレース・データが処理される。一方、間欠的プロファイル機能が使用されているとすれば、ステップ８４２においてサンプリング頻度を調整す

ることが妥当かもしれない。上述のように、単にサンプリング頻度カウンタの値を初期値に下げることによってサンプリング頻度を調整することができる。１つの手法は、一旦コード・トレースの１つが活動性しきい値を超えたならば、サンプリング頻度を増加させるものである。これによって、一層高いレベルのサンプリングを正当化する一定レベルの活動性が観察されるまで、最小限度のサンプリング・オーバーヘッドを保ちながらコードの実行を進めることが可能とされる。

【００６３】

プロセス文脈切替え割り込みが発生する時、現在時トレース開始アドレス・レジスタ５６０、最後のトレース開始アドレス・レジスタ５７０、現在時分岐履歴レジスタ５８０および最後の分岐履歴レジスタ５９０という４つのレジスタのすべてがオペレーティング・システムによってプロセス文脈切替えの間に保存される。代替的には、現在時トレース・レジスタ５６０および５８０だけが保存され、プロセス文脈が復元される時最後のトレース・レジスタ５７０および５９０が消去される。注意すべきは、プロセスがオペレーティング・システムからのトレース・プロファイル作成を要求する場合に限り、この追加データは文脈コンテキスト切替えの間に保存される必要がある。このようにして、プロファイル作成されていないプロセスは、それらの文脈が保存または復元される時、余分のオーバーヘッドを発生させない。

【００６４】

上述のように、本発明の利点の１つは、プロファイル・データが実行の間に処理されることができるという点である。実行時処理は、動的変換、バイナリ変換、インタイム・コンパイルおよび実行時変換に対して特に役立つ。本発明を使用して、プログラム・コードのバイナリ・イメージを通してトレースを逆アセンブルすることができる。トレースに関する分岐履歴値を取り出し、開始アドレスから開始して、分岐履歴のビット値をトレースの範囲内の各条件つき直接分岐と関係づけることによって、トレースの流れ制御を逆アセンブルすることができる。この逆アセンブリは、実行時に先だって事前に計算されるべきいかなる追加情報も必要とせず、実行時に遂行することができる。

【００６５】

たとえトレースが複数の動的にリンクされたプロシージャを通して進行するとしても、動的目標アドレスがコードのバイナリ・イメージの中に存在するので、逆アセンブリはなおも可能である。一方、動的リンクの実行後では逆アセンブリは困難または不可能である。なぜなら、プロセスが終了すると動的アドレス・データは消失するかもしれないし、コードが次に実行される時リンケージは完全に異なるからである。

【００６６】

トレースのプロファイル・データが動的に処理される図８のステップ８５０の一層詳細な実施形態が図９に示されている。トレース・データを処理するため、ステップ８５２において、トレースは、先ず、開始アドレスから始めて、トレース・コードのバイナリ・イメージから逆アセンブルされる。各命令バイナリ・イメージは、条件つき直接分岐に出会うまで、逆アセンブルされる。分岐履歴値の有効部分のマーカー・ビットに隣接する最上位ビットから開始して、分岐履歴値が使用される。逆アセンブリは、分岐履歴ビットが分岐が実行されなかったことを標示すれば順次続行し、分岐が標示されれば目標アドレスに進む。このプロセスは、トレース終了条件に対応する命令に出会うまで、続く。

【００６７】

ステップ８５４においてコードが最適化され、ステップ８５４において将来の実行のためコード・キャッシュに記憶される。この時点で任意の適切なコード最適化技術を使用することができる。次に、ステップ８５６において、実行中コードは、トレースに関する開始アドレスにおける命令をコード・トレースに対応するコード・キャッシュにおけるコードの開始へのジャンプ命令と置き換えることによって、パッチを当てられる。コード・トレースのためのコードが処理され置き換えられたので、ステップ８５８において、当該開始アドレスおよび分岐履歴に関するテーブル・エントリは解放することができる。

【００６８】

本発明は、”仮想マシン”およびハードウェア・マシンの両方に関連して使用されることができる（ここで”マシン”とは”プロセッサ”に概ね匹敵する用語である）。仮想マシンはハードウェア・マシンと類似しているが、仮想マシンはソフトウェアで実施される。仮想マシンのソフトウェアは、一般的に該仮想マシンと異なるアーキテクチャを持つハードウェア・マシン上で（場合によっては別の仮想マシンを経由して）実行される。例えば、サン・マイクロシステムズは、Ｊａｖａプログラミング言語に関連する仮想マシンを定義している。この仮想マシンは、Ｊａｖａバイトコードとして知られるコードを実行する。ハードウェア・マシンと同様に、仮想マシンは、それが実行する命令コードのセットによって主に定義される。いくつかのハードウェア・マシンの各々に対して仮想マシン実施形態を作成することによって（各仮想マシンは同じマシン・アーキテクチャを実施するが、異なるハードウェア・マシンで実行している間はそのハードウェア・マシンに対して仮想マシン実施形態を作成することによって、プログラムの単一バイナリ・バージョン（その仮想マシン上で実行されるプログラムの１つのバージョン）を分配することが可能である。このバイナリ・バージョンは、異なるハードウェア・マシンのいずれかの上で使用することができる（そのようなハードウェア・マシンが仮想マシン・ソフトウェアの適切なバージョンと組み合わされる場合そのような多様なハードウェア・マシン上で実行される）。

【００６９】

上述の”レジスタ”および”論理”は、ハードウェアまたはソフトウェアで実施されることができる。”ソフトウェア”実施形態においてさえ、データを記憶し、論理演算を実行する基礎ハードウェアが存在する。しかし、そのような実施形態において、これらのレジスタのうちの１つのため特に設計される明確なハードウェア記憶位置はない。むしろ、ソフトウェアは、基礎をなすハードウェアにおける汎用記憶位置にデータが記憶される明確な手段を与える。同じように、”制御論理”は、ある場合には専用回路として実施されるが、汎用ハードウェア上で実行している専用ソフトウェアによって実施される場合もある。その基礎をなすハードウェアは、基本的論理制御を提供するが、それは一般的特性を持つ。ソフトウェアは、汎用ハードウェアを”指示”して、特定の論理演算を実行させる。

【００７０】

本発明の構成には多数の可能なバリエーションがある。例えば、図１０は、本発明に従ったトレース収集論理の別の１つの実施形態１０００を示している。図５の実施形態５００からの１つのバリエーションとして、トレース収集論理１０００はプロファイル・アドレス範囲バッファ１０１０を含む。プロファイル作成が望まれるコード・セクションにトレース・プロファイル作成を限定するため、特権的ソフトウェア例えばオペレーティング・システムによって、プロファイルされるべきコードに関する仮想アドレス限界がプロファイル・アドレス範囲バッファ１０１０にロードされる。次に、トレース・コントローラ１０２０は、プロファイル・アドレス範囲バッファ１０１に記憶されている仮想限界の範囲内に開始アドレスを持つコード・トレースに関してのみトレース・データを収集するように修正される。

【００７１】

別の可能なバリエーションは、最後のトレース開始アドレス・レジスタ５７０および最後の分岐履歴レジスタ５９０をＦＩＦＯ１０３０と入れ替えることである。ＦＩＦＯ１０３０は、現在時トレース開始アドレス・レジスタ５６０および現在時分岐履歴レジスタ５８０の出力に接続され、多数のコード・トレースに関する開始アドレスおよび分岐履歴のペアを記憶することができる。トレース・コントローラ１０２０がトレース終了条件に応答して第３ロード信号１０２８を生成する。第３ロード信号１０２８は、ＦＩＦＯバッファ１０３０へ入力され、現在時トレース開始アドレス・レジスタ５６０および現在時分岐履歴レジスタ５８０の内容をＦＩＦＯ１０３０にロードする。割込み処理ルーチンが、選択信号１６２Ｃを使用してＦＩＦＯ１０３０を選択してＦＩＦＯの内容を読み取る。ＦＩＦＯ１０３０が多数のコード・トレースに関する開始アドレスおよび分岐履歴値を収集することができるので、割り込みサイクルの各々が多数のコード・トレースに関するトレース

・データを処理することができる。割込み信号を生成するＦＩＦＯフル信号を生成するようにＦＩＦＯ１０３０を構築することができるし、あるいは、ＦＩＦＯ１０３０のエントリの数に基づいてサンプリング頻度を調整することができる。このように、統計的プロファイリル作成のオーバーヘッドは更に減らすことができる。この実施形態においてはＦＩＦＯバッファが例示されているが、複数エントリを記憶するＬＩＦＯのようないかなるバッファでも本発明において動作する。

【００７２】

本発明の利点の１つは、プログラム・コードに特別の仕組みを装備することなく、プログラムの統計的プロファイル作成を実行することができることである。それに加えて、サイクル・タイムへの影響を最小限度に抑えながらプロファイル作成データを取得することができる。サイクル・タイムが比較的影響を受けない理由はいくつかある。第１に、条件付き直接分岐が実行される時、プロファイル作成システムは、分岐結果に応じて現在時トレース分岐履歴バッファにおける分岐履歴を副次的作用としてシフト送りするだけでよい。言い換えると、仕組みを装備したコードではトレース情報が臨界実行パスの内部で収集されるのに対して、本発明ではそのような臨界実行パスの外部で収集される。従って、条件付き直接分岐命令からの臨界実行パスへの影響はなく、サイクル・タイムへの影響はない。

【００７３】

本発明は、また、過去の遺産としてのプログラム・バイナリ・コードにも適用することができる。遺産プログラムは再コンパイルできないことが多く、そのため、コードに仕組みを組み入れることができず、実行後の分析および再コンパイルができない。本発明は再コンパイルせずに利用することができるので、過去の遺産プログラム・バイナリ・コードに適用することができる。

【００７４】

更に、プログラム・コードにいかなる追加情報を供給する必要もなく、また、システムからいかなるプロファイル・データを出力する必要もない。その代わりに、ソフトウェアは、制御の流れを再構築するため、通常の間隔で、バッファに記憶された分岐履歴および開始アドレスをサンプリングすることができる。

【００７５】

本発明の別の利点の１つは、単純な論理コンポーネントを使用してハードウェアを実施することができるので、本発明のコストは安いという点である。必要とされるハードウェアは、開始アドレスおよび分岐履歴値を保持する４つのレジスタの他には、単にシフト機構と２、３の比較器だけである。また、現在の多くのプロセッサは、本発明のアドレスおよび分岐履歴バッファを制御するために使用することができる論理機構を既に備えているので、コストと複雑さが更に減る。最近のプロセッサの大多数は、動的分岐予測を実行するため、本発明において使用されるものより多くの論理機構を使用している。

【００７６】

更に別の本発明の利点は、オペレーティング・システムに対する修正が必要でないということである。本発明においては、現在時開始アドレスおよび現在時分岐履歴はプロファイル作成対象プログラムの状態の一部となる。従って、現在時トレース開始アドレスおよび現在時分岐履歴の記憶のため２つの付加的レジスタ・フィールドを備えるように文脈データ構造を改良するだけでよい。本発明の更にもう１つの利点は、プログラムが実行している間にそのプログラム・コードの実行時間を減少させることができるという点である。コードの実行時特性に基づいて、プログラム・コードとは別に維持されるコード・キャッシュに変換されるトレースに関する最適化されたコードの実行時生成を通して、プログラムの実行は改良される。

【００７７】

以上、好ましい実施形態を用いて本発明の原理を例示したが、そのような原理から逸脱することなく本発明の構成および細部を修正することが可能であることは明らかであろう。

【００７８】

本発明には、例として次のような実施様態が含まれる。
（１）プロセスの分岐履歴を記録する装置であって、第１ロード信号に応答して命令アドレス値を受け取ってその命令アドレス値を記憶する第１トレース開始アドレス・レジスタと、分岐履歴値を記憶し、シフト信号に応答して当該分岐履歴値をシフトして当該分岐履歴値の第１位置に分岐結果信号を記憶し、上記第１ロード信号に応答して上記分岐履歴値をクリアする第１分岐履歴レジスタと、第２ロード信号に応答して上記第１トレース開始アドレス・レジスタの命令アドレス値および上記第１分岐履歴レジスタの分岐履歴値を受け取ってそれらを記憶し、バッファ選択信号に応答してデータ・バスに上記分岐履歴値を出力するバッファと、命令信号を受け取り、そのような命令信号に応答して上記シフト信号、上記第１ロード信号および上記第２ロード信号を生成するコントローラと、を備える分岐履歴記録装置。
【００７９】
（２）上記バッファが、第２ロード信号に応答して上記第１トレース開始アドレス・レジスタの命令アドレス値を受け取って記憶し、第１選択信号に応答してデータ・バスに命令アドレス値を出力する第２トレース開始アドレス・レジスタと、上記第２ロード信号に応答して上記第１分岐履歴レジスタの分岐履歴値を受け取って記憶し、第２選択信号に応答してデータ・バスに上記分岐履歴値を出力する第２分岐履歴レジスタと、を更に含む、前記（１）に記載の分岐履歴記録装置。
（３）上記第１分岐履歴レジスタが、上記第１ロード信号に応答して上記第１位置にマーカーをセットし、上記第１分岐履歴レジスタの第２位置へのマーカーのシフト動作に応答してフル信号を生成し、上記コントローラがトレース終了信号または上記フル信号のいずれかに応答して上記第１ロード信号および上記第２ロード信号を生成する、前記（１）に記載の分岐履歴記録装置。
【００８０】
（４）コンピュータ上で実行されるプログラムのプロファイルを作成する方法であって、第１アドレス・レジスタに命令アドレス値を記憶するステップと、条件付き直接分岐命令の条件に応答して第１分岐履歴レジスタをシフトさせてその第１位置に分岐結果値を記憶するステップと、トレース終了条件に応答して上記第１アドレス・レジスタから第２アドレス・レジスタへ上記命令アドレス値を転送するステップと、上記トレース終了条件に応答して上記第１分岐履歴レジスタから第２分岐履歴レジスタへ分岐履歴値を転送するステップと、を含む方法。
（５）上記トレース終了条件が後方分岐命令値または間接分岐命令値のいずれかに応答して発生する、前記（４）に記載の方法。
（６）プログラムの実行を割り込むステップと、上記第２アドレス・レジスタから命令アドレス値を読み取るステップと、上記第２分岐履歴レジスタから分岐履歴値を読み取るステップと、上記命令アドレス値および上記分岐履歴値をテーブルのエントリへ転送するステップと、を更に含む前記（４）に記載の方法。
【００８１】
（７）コンピュータ上で実行されるプログラムのプロファイルを作成する方法であって、第１アドレス・レジスタに命令アドレス値を記憶するステップと、条件付き直接分岐命令の条件に応答して第１分岐履歴レジスタをシフトさせてその第１位置に分岐結果値を記憶するステップと、トレース終了条件に応答して上記第１アドレス・レジスタから第２アドレス・レジスタへ上記命令アドレス値を転送するステップと、上記トレース終了条件に応答して上記第１分岐履歴レジスタから第２分岐履歴レジスタへ分岐履歴値を転送するステップと、上記第２アドレス・レジスタから上記命令アドレス値を読み取るステップと、上記第２分岐履歴レジスタから上記分岐履歴値を読み取るステップと、実行されたトレースに関して既に記憶された分岐履歴値を取り出すステップと、実行されたトレースに関する分岐履歴値を使用して実行されたトレースを逆アセンブルすることによって実行されたトレースに関する実行された命令のセットを取得するステップと、上記実行された命令セットをコード・キャッシュに記憶するステップと、上記コード・キャッシュにおける実行さ

れた命令セットへジャンプするように、上記実行されたトレースの開始アドレスにある命令を修正するステップと、を含む方法。

【００８２】

（８）上記実行されたトレースに関する上記実行された命令セットを最適化するステップを含む、前記（７）に記載の方法。

（９）実行されたトレースに関する分岐履歴値を使用して実行されたトレースを逆アセンブルすることによって実行されたトレースに関する実行された命令のセットを取得する上記ステップが、上記実行されたトレースの開始アドレスから始めて該トレースの最初の条件付き直接分岐命令まで各命令を逆アセンブルするステップと、対応するマーカーに隣接する分岐履歴の最初の値が第１論理状態にセットされていれば上記最初の条件付き直接分岐命令に続く次の順次命令から開始して次の条件付き直接分岐命令まで各命令の逆アセンブルを継続するステップと、上記分岐履歴の最初の値が第２論理状態にセットされていれば、上記最初の条件付き直接分岐の目標アドレスにおける命令から命令次の条件付き直接分岐命令まで各命令の逆アセンブルを継続するステップと、を含む前記（７）に記載の方法。

【００８３】

（１０）プロセッサにおける実行行動を記憶する装置であって、プロセッサの命令レジスタに接続され、上記命令レジスターから受け取る第１のあらかじめ定められた命令に応答して第１制御信号を生成し、上記命令レジスターから受け取る第２のあらかじめ定められた命令に応答して第２制御信号を生成するように構成されるコントローラと、上記コントローラおよび上記プロセッサの第１プログラム・カウント・レジスタに接続され、上記第１制御信号に応答して上記プログラム・カウント・レジスタから受け取るプログラム・カウント値をラッチするように構成される第１レジスタと、上記コントローラおよび上記第１レジスタに接続され、上記第１制御信号に応答して上記第１レジスタのプログラム・カウント値をラッチするように構成される第２レジスタと、コントローラに接続され、上記プロセッサの分岐結果信号を受け取るように構成されるシフト・レジスタであって、上記第１制御信号に応答して該シフト・レジスタの内容をクリアし、上記第２制御信号に応答してその内容を第１位置から第２位置の方向へシフトさせその第１位置へ分岐結果信号の値を記憶するように構成されるシフトレ・ジスタと、上記コントローラおよび上記第シフト・レジスタに接続され、上記第１制御信号に応答して上記シフト・レジスタの内容をラッチするように構成される第３レジスタと、を備える装置。

【００８４】

【発明の効果】

本発明によると、プログラム・コードに特別の仕組みを装備することなく、サイクル・タイムへの影響を最小限度に抑えながら、プログラムの統計的プロファイル作成を実行することができる。

【図面の簡単な説明】

【図１】従来技術の制御流れグラフ（ＣＦＧ）を示すブロック図である。

【図２】（Ａ）は第１の従来技術のプロファイル作成および分岐予測方法において使用される分岐目標バッファのブロック図であり、（Ｂ）は第２の従来技術の分岐予測方法において使用される履歴レジスタ・テーブルおよびパターン・テーブルのブロック図である。

【図３】図２の（Ｂ）における履歴レジスタ・テーブルにおける分岐履歴の１例を示すブロック図である。

【図４】従来技術のプロセッサの機能ブロック図である。

【図５】本発明に従ったトレース収集論理機構の第１の実施形態の機能ブロック図である。

【図６】図５のトレース・コントローラの１つの実施形態の機能ブロック図である。

【図７】図５のトレース収集論理機構の機能の１例を示すブロック図である。

【図８】本発明のための割込み処理ルーチンの実施形態の制御の流れ図である。

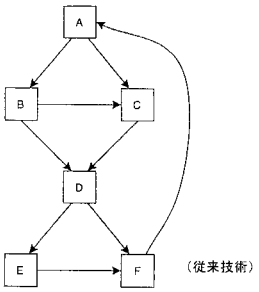
【図９】図８のコード・トレースを処理するステップの詳細な制御の流れ図である。

【図１０】本発明に従ったトレース収集論理機構の第２の実施形態の機能ブロック図である。

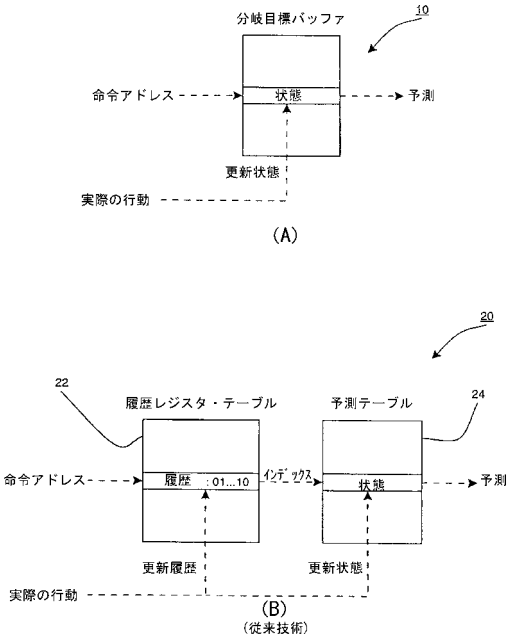
【符号の説明】

５００　　トレース収集論理機構
５２０　　コントローラ
５６０　　トレース開始アドレス・レジスタ
５７０、５９０　　バッファ
５８０　　分岐履歴レジスタ

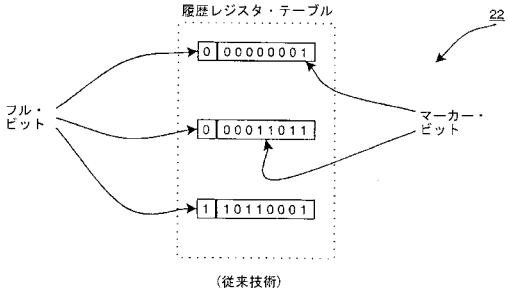
【図１】



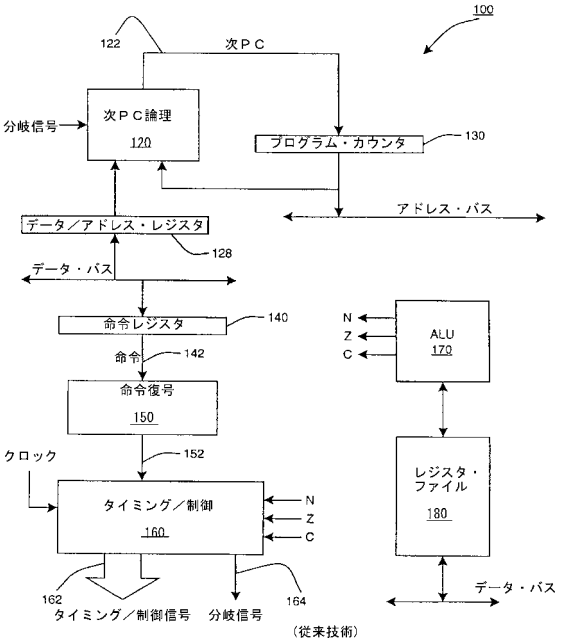
【図２】



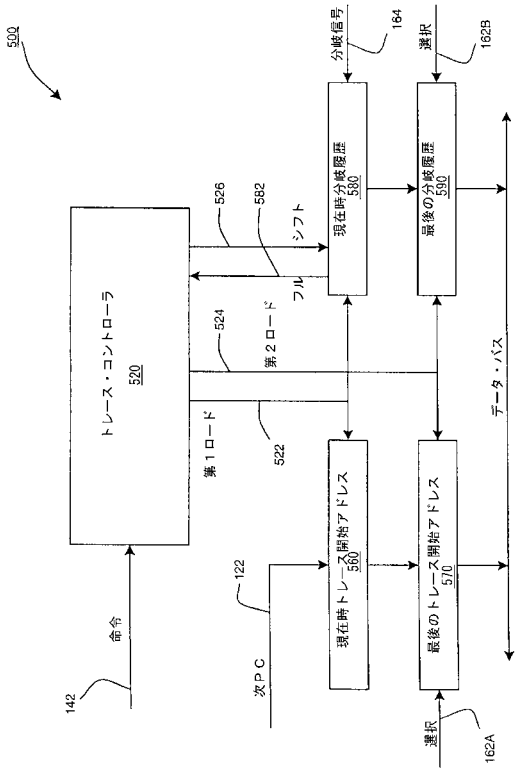
【図３】


履歴レジスタ・テーブル
22
0 00000001
0 00011011
1 10110001
フル・ビット
マーカー・ビット
（従来技術）

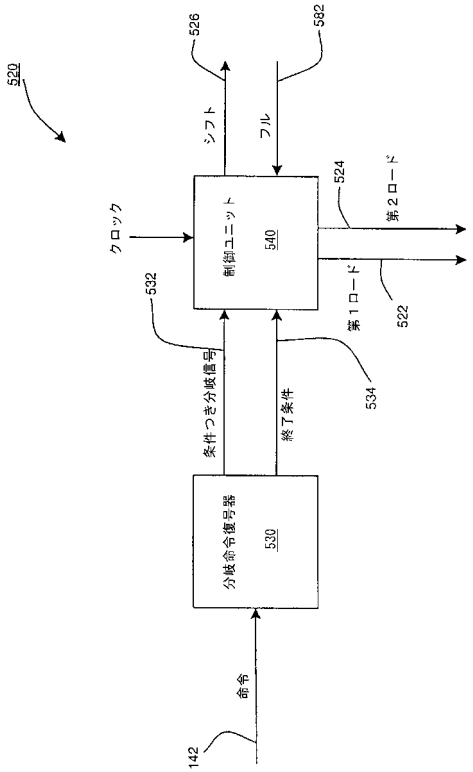
【図４】


100
122
次ＰＣ
分岐信号
次ＰＣ論理
120
プログラム・カウンタ
130
アドレス・バス
データ／アドレス・レジスタ
128
データ・バス
命令レジスタ
140
命令
142
命令復号
150
クロック
152
タイミング／制御
160
N
Z
C
ALU
170
レジスタ・ファイル
180
データ・バス
162
タイミング／制御信号
164
分岐信号
（従来技術）

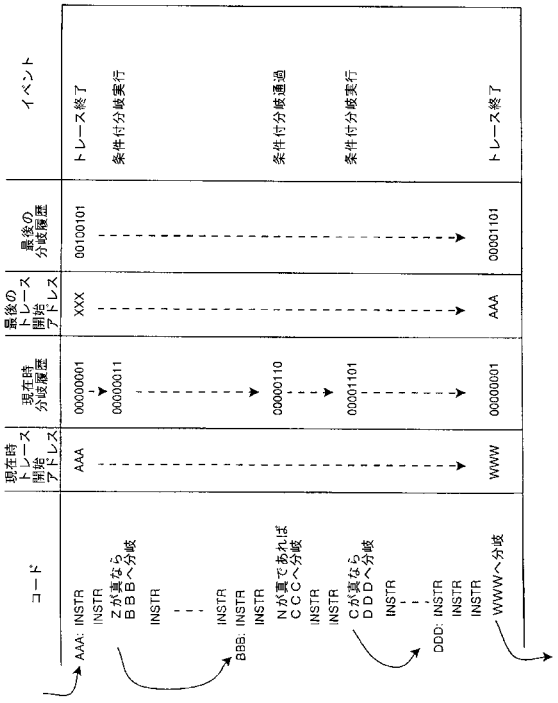
【図５】


500
分岐信号
164
選択
162B
トレース・コントローラ
520
現在時分岐履歴
580
最後の分岐履歴
590
526
582
シフト
フル
524
第２ロード
第１ロード
522
データ・バス
現在時トレース開始アドレス
560
最後のトレース開始アドレス
570
122
次ＰＣ
命令
142
選択
162A

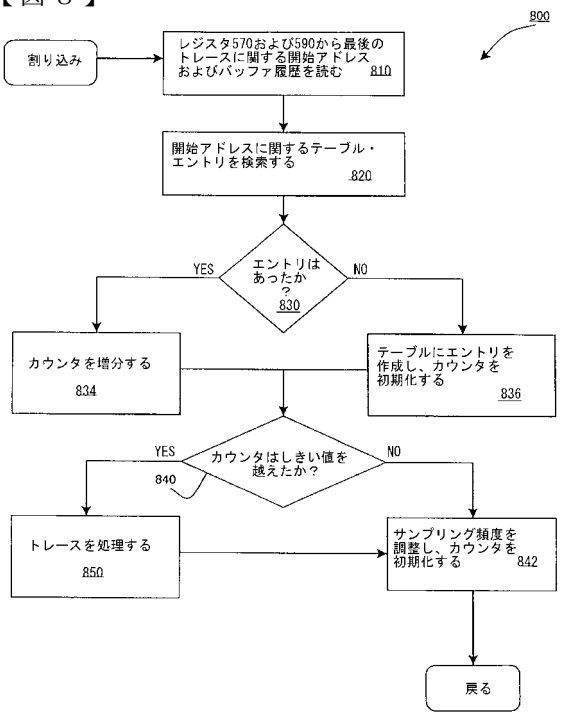
【図６】


520
526
582
シフト
フル
クロック
制御ユニット
540
524
第２ロード
第１ロード
522
532
534
条件つき分岐信号
終了条件
分岐命令復号器
530
命令
142

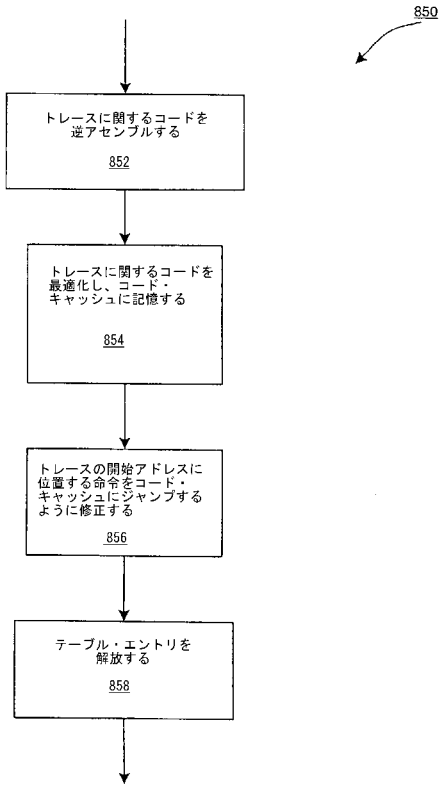
【図７】


イベント
トレース終了
条件付分岐実行
条件付分岐通過
条件付分岐実行
トレース終了
最後の分岐履歴
00100101
00001101
最後のトレース開始アドレス
XXX
AAA
現在時分岐履歴
00000001
00000011
00000110
00001101
00000001
現在時トレース開始アドレス
AAA
WWW
コード
AAA: INSTR
INSTR
Zが真ならBBBへ分岐
INSTR
INSTR
BBB: INSTR
INSTR
Nが真であればCCCへ分岐
INSTR
INSTR
Cが真ならDDDへ分岐
INSTR
DDD: INSTR
INSTR
INSTR
WWWへ分岐

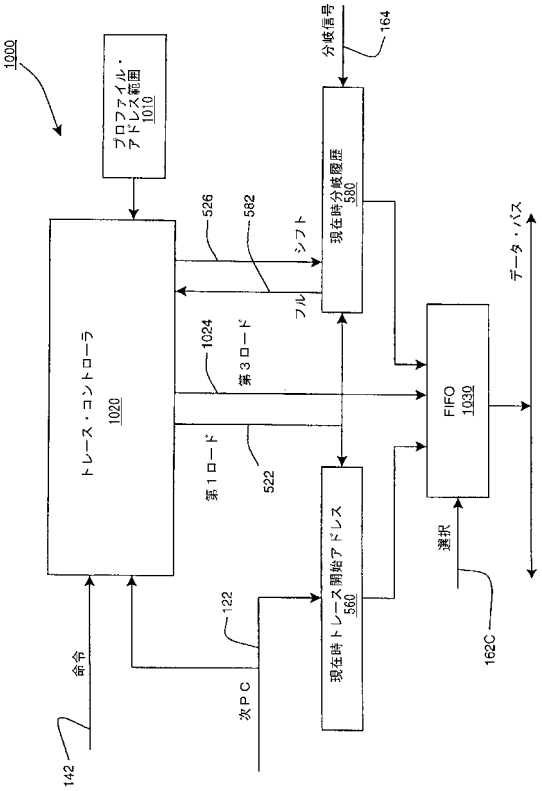
【図８】


800
割り込み
レジスタ570および590から最後のトレースに関する開始アドレスおよびバッファ履歴を読む 810
開始アドレスに関するテーブル・エントリを検索する 820
エントリはあったか？ 830
YES
NO
カウンタを増分する 834
テーブルにエントリを作成し、カウンタを初期化する 836
カウンタはしきい値を越えたか？ 840
YES
NO
トレースを処理する 850
サンプリング頻度を調整し、カウンタを初期化する 842
戻る

【図９】


850
トレースに関するコードを逆アセンブルする 852
トレースに関するコードを最適化し、コード・キャッシュに記憶する 854
トレースの開始アドレスに位置する命令をコード・キャッシュにジャンプするように修正する 856
テーブル・エントリを解放する 858

【図１０】


1000
プロファイル・アドレス範囲 1010
分岐信号
164
トレース・コントローラ 1020
526
582
シフト
フル
現在時分岐履歴 580
データ・バス
1024
第３ロード
第１ロード
522
FIFO 1030
選択
162C
現在時トレース開始アドレス 560
122
次ＰＣ
命令
142

フロントページの続き

(58)調査した分野(Int.Cl.[7], ＤＢ名)
G06F 9/32
G06F 11/28