(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 698 19 849 T2** 2004.09.02

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 919 918 B1**
(21) Deutsches Aktenzeichen: **698 19 849.2**
(96) Europäisches Aktenzeichen: **98 309 631.4**
(96) Europäischer Anmeldetag: **25.11.1998**
(97) Erstveröffentlichung durch das EPA: **02.06.1999**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **19.11.2003**
(47) Veröffentlichungstag im Patentblatt: **02.09.2004**

(51) Int Cl.[7]: **G06F 11/34**

(30) Unionspriorität:
980190 **26.11.1997** **US**

(73) Patentinhaber:
Compaq Computer Corp., Houston, Tex., US

(74) Vertreter:
Grünecker, Kinkeldey, Stockmair & Schwanhäusser, 80538 München

(84) Benannte Vertragsstaaten:
DE, FR, GB

(72) Erfinder:
Chrysos, George Z., Marlboro, Massachusetts 01752, US; Dean, Jeffrey A., Menlo Park, California 94025, US; Hicks, James E., Newton, Massachusetts 02159, US; Leibholz, Daniel L., Cambridge, Massachusetts 02138, US; McLellan, Edward J., Holliston, Massachusetts 01746, US; Waldspurger, Carl A., Atherton, California 94027, US; Weihl, William E., San Francisco, California 94114, US

(54) Bezeichnung: **Anordnung zum willkürlichen Abtasten von Instruktionen in einer Prozessorpipeline**

Beschreibung

[0001] Die vorliegende Erfindung bezieht sich allgemein auf das Messen der Funktion eines Computersystems, und insbesondere auf ein Abtasten von Ausführungsbefehlen.

[0002] Computerprozessoren werden schneller, allerdings hält eine Software-Anwendungsfunktion nicht damit Schritt. Für große, kommerzielle Anwendungen können durchschnittliche Werte von Prozesszyklen pro Befehl (cycles-per-instruction – CPI) bis zu 2,5 oder 3 hoch sein. Mit einem Vier-Wege-Befehl-Ausgabe-Prozessor bedeutet ein CPI von drei, dass nur ein Befehlsschlitz alle zwölf gut genutzt wird. Es ist wichtig zu verstehen, warum ein Software-Durchsatz nicht an Hardware Verbesserungen angepasst ist.

[0003] Es ist üblich, solche Probleme auf die Speicher-Latenzzeit zu schieben. Tatsächlich benötigen viele Softwareanwendungen viele Zyklen, darauf wartend, dass Datenübertragungen abgeschlossen werden. Allerdings verschwenden andere Probleme, wie beispielsweise Verzweigungs-Fehlvorhersagen, auch Prozessorzyklen. Unabhängig der allgemeinen Ursachen, müssen System-Architekten, und Hardware- und Software-Ingenieure, wissen, welche Befehle blockieren bzw. überlasten und warum, um die Funktionsweise von modernen Computersystemen, die komplexe Prozessoren einsetzen, zu verbessern.

[0004] Typischerweise wird dies durch Erzeugen eines „Profils" des Verhaltens eines Systems, während es arbeitet, vorgenommen. Ein Profil ist eine Aufzeichnung von Funktionsdaten. Häufig wird das Profil graphisch dargestellt, so dass die Engstellen der Funktion leicht identifiziert werden können.

[0005] Ein Profilieren kann durch ein Instrumentarium und eine Simulation vorgenommen werden. Mit dem Instrumentarium wird ein zusätzlicher Code zu einem Programm hinzugefügt, um spezifische Ereignisse während der Ausführung eines Programms zu überwachen. Eine Simulation versucht, das Verhalten des gesamten Programms in einer künstlichen Umgebung zu Emulieren, im Gegensatz dazu, das Programm in dem realen System auszuführen.

[0006] Jedes dieser zwei Verfahren besitzt seine Nachteile. Ein Instrumentarium stört das wahre Verhalten des Programms aufgrund der hinzugefügten Befehle und der zusätzlichen Daten-Referenzen. Eine Simulation vermeidet ein Störung auf Kosten eines wesentlichen Funktion-Overheads, wenn mit der Ausführung des Programms auf einem realen System verglichen wird. Weiterhin ist es, mit entweder einem Instrumentarium oder einer Simulation, gewöhnlich schwierig, ein gesamtes, groß dimensioniertes Softwaresystem zu profilieren, d. h. Anwendung, Betriebssystem und Vorrichtungs-Treibercode.

[0007] Eine mit einer Hardware implementierten Ereignis-Abtastung kann auch dazu verwendet werden, Profil-Informationen von Prozessoren zu erhalten. Eine Hardware-Abtastung besitzt eine Anzahl von Vorteilen gegenüber einer Simulation und einem Instrumentarium: sie erfordert keine Modifizierung von Softwareprogrammen, um deren Funktion zu messen. Eine Abtastung arbeitet auf vollständigen Systemen, mit einem relativ geringen Overhead. Tatsächlich ist in neuerer Zeit gezeigt worden, dass eine Profilierung mit geringem Overhead, basierend auf einer Abtastung, dazu verwendet werden kann, detaillierte Befehl-Level-Informationen über Pipeline-Blockierungen und deren Ursachen zu erhalten. Allerdings fehlt vielen Hardware-Abtast-Techniken eine Flexibilität, da sie so ausgelegt sind, um spezifische Ereignisse zu messen.

[0008] Die meisten, existierenden Mikroprozessoren, wie beispielsweise der DIGITAL (RTM) Alpha AXP 21164, der Intel (RTM) Pentium Pro und der MIPS (RTM) R10000, sehen Ereignis-Zähler vor, die eine Vielzahl von Ereignissen zählen können, wie beispielsweise Daten-Cache-(D-Cache)-Fehler, Befehls-Cache-(I-Cache)-Fehler und Verzweigungs-Fehlvorhersagen. Die Ereignis-Zähler erzeugen eine Unterbrechung, wenn die Zähler überlaufen, so dass die Performance-Daten in den Zählern durch eine Software auf höheren Niveaus abgetastet werden können.

[0009] Alle Ereigniszähler sind zum Erfassen von aggregierten Informationen nützlich, wie beispielsweise die Zahl von Verzweigungs-Fehlvorhersagen, die das System erfährt, während ein bestimmtes Programm, oder ein Teil davon, ausgeführt wird. Allerdings sind bekannte Ereignis-Zähler weniger nützlich, um Zustands-Informationen in Bezug auf individuelle Befehle zuzuordnen, wie beispielsweise darüber, welche Verzweigungs-Befehle häufig fehlinterpretiert werden. Dies kann aufgrund der Tatsache erfolgen, dass die Programm-Zähler (program counters – PC) von Befehlen, die Ereignisse verursachten, nicht länger verfügbar sein können, wenn der Ereignis-Zähler überläuft und unterbricht.

[0010] Es ist ein besonderes Problem, die dynamische Operation eines Prozessors zu deduzieren, die Befehle außerhalb der Reihenfolge ausgeben kann. Tatsächlich kann das Verhalten von Software-Programmen, die von einem Prozessor außerhalb der Reihenfolge ausgeführt werden, sehr subtil und schwierig zu verstehen sein. Es wird der Ablauf von Befehlen in einem Out-Of-Order Alpha 21264 Prozessor als ein konkretes Beispiel betrachtet.

Superscalar-Prozessor-Architektur

Ausführungs-Reihenfolge

[0011] Ein gestörter Prozessor bzw. ein Prozessor „Out-of-Order" ruft Befehle ab und scheidet sie aus, verarbeitet allerdings die Befehle entsprechend deren Datenabhängigkeiten. Eine Verarbeitung von Befehlen kann eine Register-Auflistung, eine Befehl-Ausgabe und -Ausführung umfassen. Ein Befehl befindet sich von dem Zeitpunkt an „ In Bewegung"

(„in-flight"), von dem an er abgerufen ist, bis er ausscheidet oder ausgesondert wird.

[0012] Während jedes Prozessor-Zyklus ruft eine erste Stufe der Prozessor-Pipeline einen Satz von Befehlen von dem Befehls-Cache (I-Cache) ab. Der Satz von Befehlen wird decodiert. Der Befehl-Decodierer identifiziert, welche Befehle in dem abgerufenen Satz ein Teil der Befehl-Datenfolge sind.

[0013] Da es mehrere Zyklen benötigen kann, um den PC eines nächsten Befehls aufzulösen, um abzurufen, wird der PC gewöhnlich zuvor durch eine Verzweigung oder einen Sprung-Prädikator vorhergesagt. Wenn die Vorhersage nicht korrekt ist, wird der Prozessor die fehlerhaft vorausgesagten Befehle aussondern, die einen „schlechten" Ausführungspfad belegen, und wird erneut damit beginnen, Befehle auf dem „guten" Pfad abzurufen.

[0014] Um zu ermöglichen, dass Befehle außerhalb der Reihenfolge ausgeführt werden, werden Register, spezifiziert in Operanden von Befehlen, dynamisch umbenannt, um Konflikte eines Schreibens nach einem Lesen (write-after-read) und eines Schreibens nach einem Schreiben (write-after-write) zu verhindern. Diese Umbenennung wird durch ein Auflisten von architekturmäßigen oder „virtuellen" Registern zu physikalischen Registern begleitet. Demzufolge können zwei Befehle, die dasselbe, virtuelle Register beschreiben, sicher außerhalb der Reihenfolge ausgeführt werden, da sie zu unterschiedlichen, physikalischen Registern hin schreiben werden, und Benutzer der virtuellen Register werden die geeigneten Werte erhalten.

[0015] Ein mittels Register aufgelisteter Befehl ist in der Befehl-Warteschlange vorhanden, bis seine Operanden berechnet worden sind und eine funktionale „Ausführungs"-Einheit des geeigneten Typs verfügbar ist. Die physikalischen Register, verwendet durch einen Befehl, werden in dem Zyklus gelesen, den der Befehl ausgibt. Nachdem Befehle ausgeführt worden sind, werden sie als bereit markiert, um sie auszuscheiden, und werden durch den Prozessor ausgeschieden werden, wenn alle vorherigen Befehle, bereit um auszuscheiden, in der Programm-Reihenfolge ausgeschieden worden sind, d. h. Befehle scheiden in der korrekten Programm-Reihenfolge aus. Unter dem Ausscheiden beginnt der Prozessor die Änderungen, vorgenommen durch den Befehl, zu dem architekturmäßigen „Zustand" des Systems, und gibt Ressourcen, verbraucht durch den Befehl, frei.

Fehlvorhersage

[0016] In einigen Fällen, wie beispielsweise solchen, wenn eine Verzweigung fehlerhaft vorhergesagt ist, müssen Befehle aufgefangen oder ausgesondert werden. Wenn dies auftritt, wird der momentane, spekulative, architekturmäßige Zustand zurück zu einem Punkt in der Ausführung abgewickelt, wo die fehlerhafte Vorhersage aufgetreten ist, und ein Abrufen fährt an dem korrekten Befehl fort.

Verzögerungen

[0017] Zahlreiche Ereignisse können die Ausführung eines Befehls verzögern. An der Front der Pipeline kann die Abrufeinheit aufgrund eines Fehlens eines I-Cache anhalten, oder die Abrufeinheit kann Befehle entlang eines schlechten Pfads aufgrund einer fehlerhaften Vorhersage abrufen. Die Auflistungseinrichtung kann aufgrund eines Fehlens von freien, physikalischen Registern oder aufgrund eines Fehlens von freien Schlitzen in der Ausgabe-Warteschlange blockieren. Befehle in der Ausgabe-Warteschlange können darauf warten, dass deren Register-Abhängigkeiten erfüllt werden, oder auf die Verfügbarkeit von funktionalen Ausführungseinheiten.

[0018] Befehle können aufgrund eines Fehlens eines Daten-Cache blockieren. Befehle können stoppen, und zwar aufgrund davon, dass sie spekulativ entlang eines schlechten Pfads ausgegeben wurden, oder da der Prozessor eine Unterbrechung vornahm. Viele dieser Ereignisse sind schwierig statisch vorherzusagen, z. B. durch eine Prüfung des Codes, und alle davon setzen die Funktionsweise des Systems herab. Einfache Ereignis-Zähler sind nicht ausreichend, um diesen Typ von Informationen zu individuellen Befehlen zurückzuführen. Zusätzlich ist es schwierig, exakt die Längen der Verzögerungen zu messen, um zu bestimmen, welche Verzögerungen eine besondere Aufmerksamkeit verdienen.

[0019] Es ist in höchstem Maße wünschenswert, Ereignisse zu spezifischen Befehlen und Maschinenzuständen zuzuordnen, so dass Programmierer, oder Optimierungs-Tools, die Funktionsweise der Software- und der Hardware-Komponenten von komplexen Computersystemen, wie beispielsweise Super-Skalar und gestörte Prozessoren, oder für diese Art von Prozessoren mit irgendeinem architekturmäßigen Design, verbessern können.

Probleme mit Ereignis-Zählern nach dem Stand der Technik

[0020] Das Hauptproblem in Verbindung mit bekannten Ereignis-Zählern ist dasjenige, dass der Befehl, der das Ereignis verursachte, so dass der Zähler überlief, gewöhnlich lange vor dem abgetasteten Ausnahme-PC abgerufen wurde, d. h. der PC ist nicht der Befehl, der den Überlauf verursachte. Die Länge der Verzögerung zwischen dem Abrufen und der Unterbrechung ist allgemein eine nicht vorhersagbare Größe. Diese unvorhersagbare Verteilung von Ereignissen macht es schwierig, geeignet Ereignisse zu spezifischen Befehlen zuzuordnen. Eine gestörte oder spekulative Ausführung verstärkt dieses Problem, allerdings ist es auch in Maschinen, die „in-order" arbeiten, wie beispielsweise der Alpha 21164 Prozessor, vorhanden.

[0021] Zum Beispiel unterbrechen vergleichende Programm-Zählerwerte, zugeführt zu dem Funktions-Zähler, einen Handler, während D-Cache-Refe-

renz-Ereignis-Zähler für den Alpha 21164 (In-Order) Prozessor, nämlich den Pentium Pro (Out-Of-Order) Prozessor, überwacht werden. Ein Beispielprogramm besteht aus einer Schleife, die aus einem Random-Memory-Access-Befehl, zum Beispiel einem Lade-Befehl, gefolgt von hunderten von Null-Operations-Befehlen (null operation instructions – nop), besteht.

[0022] An dem In-Order-Alpha-Prozessor werden alle Funktions-Zähler-Ereignisse (zum Beispiel Cache-Fehler) dem Befehl zugeschrieben, der sechs Zyklen nach dem Ereignis ausführt, um zu einem großen Peak an Abtastungen an dem siebten Befehl nach dem Lade-Zugriff zu führen. Diese versetzte Verteilung von Ereignissen ist nicht ideal. Allerdings kann, da ein einzelner, großer Peak existiert, eine statische Analyse manchmal nach rückwärts von diesem Peak an arbeiten, um den tatsächlichen Befehl, der das Ereignis verursachte, zu identifizieren, allerdings ist dies nicht mehr als nur eine beste Annahme, sogar für ein einfaches Programm.

[0023] Für die identische Programmausführung auf dem Out-Of-Order Pentium Pro werden die Ereignis-Abtastungen weit über die nächsten 25 Befehle verteilt, was nicht nur eine Versetzung, sondern auch eine wesentliche Verschleierung ebenso, darstellt. Die weite Verteilung von Proben macht es nahezu unmöglich, ein spezifisches Ereignis dem bestimmten Befehl, der das Ereignis verursachte, zuzuschreiben. Ein ähnliches Verhalten tritt dann auf, wenn andere Hardware-Ereignisse berücksichtigt werden.

[0024] Zusätzlich zu der versetzten oder verwischten Verteilung von Ereignis-Abtastungen leiden herkömmliche Ereignis-Zähler auch unter zusätzlichen Problemen. Gewöhnlich sind dort mehr Ereignisse vorhanden, die von Interesse sind, als dort Ereignis-Zähler vorhanden sind, was es demzufolge schwierig macht, wenn nicht sogar unmöglich, gleichzeitig alle Ereignisse, die von Interesse sind, zu überwachen. Die sich erhöhende Komplexität von Prozessoren ist dahingehend wahrscheinlich, dass sie dieses Problem noch verstärkt.

[0025] Zusätzlich zeichnen Ereignis-Zähler nur die Tatsache auf, dass ein Ereignis auftrat; sie liefern nicht zusätzliche Zustandsinformation über das Ereignis. Für viele Arten von Ereignissen würden zusätzliche Informationen, wie beispielsweise die Latenzzeit, um ein Cache-Fehlereignis zu behandeln, äußerst nützlich sein.

[0026] Weiterhin sind Zähler nach dem Stand der Technik allgemein nicht in der Lage, Ereignisse zu „Blind Spots" in dem Code zu behandeln. Ein Blind Spot ist irgendein nicht unterbrechbarer Code, wie beispielsweise Systemprogramme mit hoher Priorität und ein PAL Code, da das Ereignis nicht erkannt werden wird, bis seine Unterbrechung anerkannt ist. Zu diesem Zeitpunkt kann sich der Prozessor-Zustand wesentlich geändert haben, was meistens zu falschen Informationen führt.

Blockierungen gegenüber Engstellen

[0027] Bei einem In-Order-Prozessor, im Pipeline-Betrieb, verhindert eine Befehl-Störung in einer Pipeline-Stufe, dass spätere Befehle durch diese Pipeline-Stufe hindurchführen. Deshalb ist es relativ einfach, „Engstellen" („Bottleneck") Befehle auf einem In-Order-Prozessor zu identifizieren, wobei diese Engstellen-Befehle dazu tendieren, irgendwo in der Pipeline zu blockieren. Für einen In-Order-Prozessor ist es möglich, Blockierungen durch Messen der Latenzzeit eines Befehls zu identifizieren, wenn er durch jede Pipeline-Stufe hindurchführt, und durch Vergleichen der gemessenen Latenzzeit mit der idealen Latenzzeit dieses Befehls in jeder Pipeline-Stufe. Ein Befehl kann vorab dahingehend angenommen werden, dass er in einer Stufe blockiert wurde, wenn er länger als die minimale Latenzzeit benötigt, um durch diese Stufe hindurchzuführen.

[0028] Allerdings können, an einem Out-of-Order-Prozessor, andere Befehle durch eine Pipeline-Stufe um einen Befehl herum hindurchführen, der in der Pipeline-Stufe blockiert ist. Tatsächlich kann die zusätzliche Latenzzeit des blockierten Befehls vollständig durch die Verarbeitung von anderen Befehlen maskiert sein, und tatsächlich können die blokkierten Befehle nicht den beobachteten Abschluss des Programms verzögern.

[0029] Gerade bei In-Order-Prozessoren können Blockierungen in einer Pipeline-Stufe nicht zu der gesamten Ausführungszeit eines Programms beitragen, wenn eine andere Pipeline-Stufe die Engstelle ist. Zum Beispiel können, während der Ausführung eines speicher-intensiven Programms, die Abrufeinrichtung und die Auflistungseinrichtung der Befehl-Pipeline oftmals blockieren, da der „Gegendruck" („Back-Pressure") von einer Ausführungseinheit, verzögert durch einen D-Cache, fehlt.

[0030] Idealerweise würde man die Speicher-Operationen, die Cache-Verfehlungen verursachen, als die primäre „Engstelle" klassifizieren. Die Blockierungen der Abrufeinrichtung und die Auflistungseinrichtung sind tatsächlich symptomatisch für die Verzögerungen aufgrund von Cache-Verfehlungen, das bedeutet sekundäre Engstellen.

[0031] Es wäre wünschenswert, solche Befehle, deren Blockierungen nicht durch andere Befehle markiert sind, zu identifizieren, und sie als wahre Engstellen zu identifizieren. Weiterhin ist, um ein Programmverhalten zu verbessern, ein Erfordernis vorhanden, sich auf kausale (primäre) Engstellen, im Gegensatz zu den symptomatischen (sekundären) Engstellen, zu konzentrieren. Diese Klassifizierung von Pipeline-Stufen-Engstellen als kausale und symptomatische erfordert eine detaillierte Kenntnis über den Zustand der Pipeline und der Daten- und Ressourceabhängigkeiten der In-Flight-Befehle, die nicht von einfachen Ereignis-Zählern erhalten werden können, wie sie bekannt sind.

[0032] Das US-Patent 5,151,981 „Instruction Samp-

ling Instrumentation", herausgegeben von Wescott et al., am 29. September 1992, schlägt einen Hardware-Mechanismus für eine auf einem Befehl basierende Abtastung in einer Out-of-Order-Ausführungsmaschine vor. Dabei ist eine Anzahl von Nachteilen in der Maßnahme vorhanden, die durch Wescott et al. vorgenommen wird. Zuerst kann deren Maßnahme die Datenfolge von Befehlsabtastungen systematisch die Datenfolge von Befehlsabtastungen beurteilen, da nur Befehle, die einer bestimmten, internen Befehlszahl zugeordnet sind, für eine Abtastung ausgewählt werden können.

[0033] Als Zweites tastet deren System nur ausgeschiedene Befehle ab, und nicht alle Befehle, die abgerufen sind, wobei einige davon ausgesondert sein können. Als Drittes konzentrieren sich die Informationen, die durch den Mechanismus von Wescott et al. gesammelt sind, auf individuelle Ereignis-Attribute, z. B. Cache-Verfehlungen, liefern allerdings nicht nützliche Informationen zum Bestimmen der Zwischen-Befehl-Beziehungen.

[0034] In neuerer Zeit ist ein Hardware-Mechanismus, bezeichnet als „Informing Loads", vorgeschlagen worden; siehe Horowitz et al., „Informing memory operations: Providing memory performance feedback in modern processors", Proceedings 23rd Annual International Symposium on Computer Architecture, Seiten 260–270, 22. Mai 1996. Dabei kann einer Speicher-Operation durch eine konditionale Verzweigungs-Operation gefolgt werden, die dann vorgenommen wird, und nur dann, wenn die Speicher-Operation in dem Cache fehlerhaft ist. Obwohl dieser Mechanismus nicht spezifisch für einen Profilierung ausgelegt ist, könnte er dazu verwendet werden, speziell nur verfehlte Ereignis-Informationen des D-Cache zu sammeln.

[0035] In einer anderen, spezialisierten Hardware, bezeichnet als ein Cache-Miss-Look-Aside-(CML)-Puffer, werden virtuelle Speicherseiten, die unter einer Cache-Verfehlungs-Rate mit einem hohen Level-2 leiden, identifiziert; siehe Bershad et al., „Avoiding conflict misses dynamically in large direct-mapped caches", Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems, Seiten 158–170, 4. Oktober 1994, für eine vollständige Beschreibung.

[0036] Einige Prozessoren, wie beispielsweise der Intel Pentium, erlauben einer Software, die Inhalte eines Verzweigungs-Target-Puffers (branch target buffer – BTB) des Verzweigungs-Prediktors zu lesen. Durch periodisches Lesen des BTB in der Software entwickelten Conte et al. eine sehr geringe Overheadtechnik, um Kantenausführungsfrequenzen eines Programms abzuschätzen; siehe „Using branch handling hardware to support profile-riven optimization", Proceedings of the 27th Annual International Symposium on Microarchitecture, Seiten 12–21, 30. November 1994.

[0037] Diese Maßnahme führt zu Informationen, die ähnlich zu denjenigen sind, die durch Nachvollziehen der Verzweigungs-Richtungs-Informationen, enthalten in einer „Profil-Aufzeichnung" („profile record"), die in Bezug stehende Abtastinformationen speichert, erhalten werden. In neuerer Zeit schlugen Conte et al. einen Teil einer zusätzlichen Hardware vor, bezeichnet als Profil-Puffer, der die Anzahl von Malen zählt, für die eine Verzweigung vorgenommen wird und nicht vorgenommen wird; siehe „Accurate and practical profile-driven compilation using the profile buffer", Proceedings of the 29th Annual International Symposium on Microarchitecture, Seiten 36–45, 2. Dezember 1996.

[0038] Der Artikel „Instruction Match Function for Processors Monitoring" in IBM Technical Disclosure Bulletin, Vol. 39, NO. 12, Dezember 1996, Seiten 119–121, offenbart eine Vorrichtung zum Abtasten von Befehlen, die durch einen Dispatcher vor einer Abtastung identifiziert werden.

[0039] Gemäß der vorliegenden Erfindung wird eine Vorrichtung zum Abtasten von Befehlen in einer Prozessor-Pipeline eines Systems geschaffen, wobei die Pipeline eine Vielzahl von Verarbeitungsstufen aufweist, und die umfasst:

eine Einrichtung, die Befehle in eine erste Stufe der Pipeline abruft, wobei die Befehle durch zusätzliche Felder identifiziert werden, die anzeigen, dass sie zum Abtasten ausgewählt worden sind, und die zusätzlichen Felder ein Abtast-Bit an jedem Befehl in der Pipeline enthalten;

eine Einrichtung, die jeden der abgerufenen Befehle als einen ausgewählten Befehl identifiziert;

eine Einrichtung, die Statusinformationen des Systems abtastet, während sich ein bestimmter ausgewählter Befehl in einer beliebigen Stelle der Pipeline befindet;

eine Einrichtung, die die Statusinformationen speichert; und

eine Einrichtung, die die Software informiert, wenn der bestimmte ausgewählte Befehl die Pipeline verlässt, so dass die Software jede der Statusinformationen lesen kann, wobei die Einrichtung, die abtastet, und die Einrichtung, die die Software informiert, in Funktion aktiviert, indem das Abtast-Bit in dem ausgewählten Befehl aktiviert wird.

[0040] Während des Betriebs eines Prozessors wird periodisch ein Befehl, der profiliert werden soll, zufällig ausgewählt, und eine Profil-Aufzeichnung davon, was während der Ausführung des Befehls auftritt, wird in einem Satz von internen Profil-Registern des Prozessors akkumuliert. Nachdem die Verarbeitung des ausgewählten Befehls endet, z. B. der Befehl scheidet aus, wird ausgesondert oder abgefangen, wird eine Unterbrechung erzeugt. Alternativ kann eine Software ein Zeichen oder ein Register abfragen. Die aufgezeichneten Informationen, die die Details charakterisieren, wie der Befehl in der Pipeline verarbeitet wurden, können von den internen Profil-Registern durch eine Software abgetastet werden.

[0041] Die Profil-Register können viele nützliche

Fakten über die Ausführung eines Befehls aufzeichnen. Beispielhafte Funktions-Informationen können umfassen: die Zahl von Zyklen, die der ausgewählte Befehl in jeder Stufe einer Ausführungs-Pipeline verbrachte, d. h.

[0042] Stufen-Latenzzeiten, ob der Befehl ein Verfehlen eines I-Cache oder eines D-Cache unterlag, die effektiven Adressen seiner Speicher-Operanden, oder Verzweigungs/Sprung-Ziele, und ob der Befehl ausgeschieden oder ausgesondert wurde.

[0043] Bei In-Order-Ausführungs-Prozessoren ist es möglich, die gesamte Zahl von Blokkier-Zyklen, die jedem Befehl zuschreibbar sind, wenn man die fetch-to-retire Latenzzeiten von abgetasteten Befehlen angibt, abzuschätzen.

[0044] An einem Out-Of-Order-Prozessor ist es in Bezug auf die meisten Blockierungen wahrscheinlich, dass sie andere Befehle überlappen oder durch diese maskiert werden, herausgegeben Out-Of-Order um die blockierten Befehle herum. Dies gestaltet die Identifikation von blockierten Befehlen schwierig. Zusätzlich kann es notwendig sein, Informationen über das durchschnittliche Niveau einer Konkurrenz zu sammeln, während jeder Befehl in Ausführung war, um Engstellen zu identifizieren.

[0045] Die Hardware für spezielle Zwecke könnte die Zahl von Befehlen, die ausgegeben werden, während sich ein profilierter Befehl in Ausführung befindet, zählen und aufzeichnen, um das Niveau einer gleichzeitigen Ausführung zu messen. Allerdings schlägt dies dahingehend fehl, Befehle zu berücksichtigen, die ausgegeben werden, allerdings ausgesondert sind, und deshalb fehlschlagen, auszuscheiden. Vorausgesetzt ist hier eine Messung der Menge einer nutzbaren Gleichzeitigkeit. Die nutzbare Gleichzeitigkeit ist die durchschnittliche Zahl von Befehlen, die parallel herausgegeben werden und erfolgreich ausscheiden, und zwar mit einem gegebenen Befehl. Befehle, die herausgegeben werden, allerdings darauffolgend ausgesondert werden, sind nicht nützlich. Dann können Befehle, deren Blockierungen nicht durch eine nutzbare Konkurrenz maskiert sind, als Engstellen klassifiziert werden. Um dies in einer anderen Weise auszudrücken, ist eine Schlüssel-Metrik, zum genauen Festlegen von Funktions-Engstellen an einem Out-of-Order-Prozessor, die Zahl von Ausgabe-Schlitzen, die verschwendet wurden, während ein gegebener Befehl ausgeführt wurde.

[0046] Dementsprechend wird, um eine nützliche Konkurrenz zu messen, eine Technik, bezeichnet als „N-wise sampling" („N-weises Abtasten") vorgesehen. Die Grundidee ist diejenige, eine verschachtelte Form einer Abtastung auszuführen. Hierbei wird ein Fenster von Befehlen, die gleichzeitig mit einem ersten, profilierten Befehl ausgeführt werden können, dynamisch definiert. Zum Beispiel wird dort, wo N zwei ist, ein zweiter Befehl zufällig für ein Profilieren von dem Fenster aus Befehlen ausgewählt. Der profilierte und zweite Befehl bilden ein Abtast-Paar, für das Profil-Informationen zusammengestellt werden können.

[0047] Eine paarweise Abtastung erleichtert die Bestimmung der Zahl von verschwendeten Ausgabe-Schlitzen, die jedem Befehl zuordenbar sind, und trifft Engstellen akkurater als bekannte Techniken. Allgemein ist eine paarweise Abtastung sehr flexibel, was die Basis für eine Analyse bildet, die eine breite Vielfalt von Konkurrenz- und Benutzungs-Metriken, die von Interesse sind, bestimmen kann.

[0048] Beispiele von Informationen, die erfasst werden können, umfassen: die Adresse des Befehls (Programm-Zähler oder PC), ob der Befehl einen Befehls-Cache-Fehler erlitten hat, und die Latenzzeit, die notwendig war, um den Fehler zu bearbeiten. Falls dieser Befehl zu einer Speicher-Operation führt, dann Bestimmen, ob der Befehl einen Daten-Cache-Fehler erlitt, und Messen der Latenzzeit, um die Speicher-Anforderung zu erfüllen. Weiterhin kann die Zeitdauer, die der Befehl in jeder Pipeline-Stufe benötigt, gemessen werden. Die Profil-Informationen können auch anzeigen, ob der Befiehl ausgeschieden oder ausgesondert wurde, und in dem letzteren Fall, welche Art eines Trap eine Ausführung des Befehls, um ausgesondert zu werden, verursachte.

[0049] Diese Informationen werden in einem Satz von Profilierungs-Registern zusammengestellt, wenn der Befehl durch die Ausführungs-Pipeline fortfährt. Wenn ein Befehl eine Ausführung beendet, wird er entweder aufgegeben oder er wird ausgesondert, wobei eine Unterbrechung zu einer Software auf einem höheren Niveau zugeführt wird. Die Software kann dann die Informationen, die in den Profilierungs-Registern vorhanden sind, in einer Vielfalt von Arten und Weisen verarbeiten.

[0050] Die offenbarte Technik ist eine Verbesserung gegenüber einer existierenden, eine Funktion überwachenden Hardware, und kann effektiv unter relativ niedrigen Hardware-Kosten in modernen Mikroprozessoren ausgeführt werden, die Befehle, außerhalb der Reihenfolge, ausgeben können.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0051] **Fig.** 1 zeigt ein Blockdiagramm eines Computersystems mit einer befehlsgesteuerten Zustands-Abtastung;

[0052] **Fig.** 2 zeigt ein Blockdiagramm einer Mikroprozessor-Ausführungs-Pipeline zum Verarbeiten von abgetasteten Befehlen;

[0053] **Fig.** 2b zeigt ein Blockdiagramm der Pipeline, die Zustands-Informationen darstellt, die abgetastet werden können;

[0054] **Fig.** 3 zeigt ein Blockdiagramm einer Register-Datei zum Speichern von Profil-Informationen;

[0055] **Fig.** 4 zeigt ein Blockdiagramm eines erhöhten Befehls;

[0056] **Fig.** 5 zeigt ein Blockdiagramm zum Profilieren von ausgewählten Befehlen;

[0057] **Fig.** 6 zeigt eine schematische Darstellung

einer Schaltung zum Messen von Pipeline-Latenzzeiten;

[0058] **Fig.** 7a zeigt ein Flussdiagramm eines Verfahrens zum Abtasten von Befehlen; und

[0059] **Fig.** 7b zeigt ein Flussdiagramm eines Verfahrens zum Abschätzen von Statistiken über die Eigenschaften von Befehlen, verarbeitet durch die Prozessor-Pipeline.

System-Übersicht

[0060] **Fig.** 1 stellt ein Computersystem **100** dar, das das Abtastverfahren und die -vorrichtung, die hier beschrieben sind, verwenden kann. Das System **100** umfasst einen oder mehrere Prozessoren) **110**, sich außerhalb des Chips befindliche Speicher **120** und Eingangs/Ausgangs-Schnittstellen (I/O) **130**, verbunden durch Busleitungen **140**. Die Prozessoren **110** können auf integrierten Halbleiterchips als Mehrfach-Ausführungs-Pipelines **111** ausgeführt werden, umfassend funktionale Ausführungseinheiten, sich auf dem Chip befindliche Daten-Cache (D-Cache) **113** und Befehls-Cache (I-Cache) **112**, zum Beispiel der Digital Equipment Corporation Alpha **21264** Prozessor. Der Prozessor-Chip **110** umfasst auch eine Hardware **119**, die in größerem Detail nachfolgend beschrieben ist, zum Abtasten von Prozessor-Zuständen für ausgewählte Befehle.

[0061] Die sich außerhalb des Chips befindlichen Speicher **120** können hierarchisch angeordnet werden, umfassend Cache für allgemeine Zwecke (B-Cache oder SRAM) **121**, flüchtige Speicher (DRAM) **122** und dauerhafte Speicher (Disk) **123**. Die I/O **130** kann dazu verwendet werden, Daten zu dem System **100** einzugeben und davon auszugeben.

Operation

[0062] Während der Operation bzw. des Betriebs des Systems **100** werden Befehle und Daten von Software-Programmen in den Speichern **120** gespeichert. Die Befehle und Daten werden herkömmlich unter Verwendung von bekannten Compiler-, Linker- und Loader-Techniken erzeugt. Die Befehle und die Daten werden zu der Ausführungs-Pipeline **111** eines der Prozessoren **110** über die Cache **112–113** übertragen. In der Pipeline werden die Befehle für eine Ausführung decodiert. Einige der Befehle arbeiten in Bezug auf die Daten. Andere Befehle steuern den Ausführungsfluss der Programme.

[0063] Es ist erwünscht, detaillierte Funktions-Daten zu sammeln, während die Befehle ausgeführt werden. Funktions-Daten können zu Speicher-Operationen und Ausführungs-Abläufen in Bezug gesetzt sein.

Prozessor-Pipeline

[0064] **Fig.** 2a stellt eine Ausführungs-Pipeline **200** eines der Prozessoren **110** der **Fig.** 1 dar, die eine Vielzahl von Stufen besitzt, die seriell angeordnet sind, wie, zum Beispiel, Abruf-(Fetch-), Auflistungs-(Map-), Ausgabe-(Issue-), Ausführungs-(Execute-) und Ausscheidungs-(Retire-) Einheiten, und zwar jeweils **210**, **220**, **230**, **240** und **250**. Die Rate, unter der die Pipeline **200** Informationen (Daten und Befehle) verarbeitet, wird durch Systemtaktsignale auf Leitungen **201** verarbeitet, d. h. sogenannte Takt-„Zyklen".

[0065] Jeder Takt-Zyklus definiert einen „Schlitz" oder ein Intervall einer Zeit, wenn eine Stufe der Pipeline **200** einen diskreten Umfang einer Verarbeitung vornehmen kann. Ein Verarbeitungs-Schlitz trägt gewöhnlich Vorwärts-Befehle, und, in dem Fall von Ausführungseinheiten, beschrieben nachfolgend, Daten, allgemein als „Datenelemente" nachfolgend bezeichnet. In einigen Fällen fährt, zum Beispiel bei Verzweigungs-Fehlvorhersagen oder Cache-Fehlern, oder Pipeline-Stillständen, der Takt fort, zyklisch zu arbeiten, allerdings werden keine bedeutungsvollen Befehle nach vorwärts getragen.

[0066] Als ein Vorteil können die vorliegende Vorrichtung und das Verfahren Zustands-Informationen über Prozessor-Schlitze abtasten, die „Abfall" („garbage") oder keine nützlichen Daten führen. Diese sind als „verschwendete" Schlitze bekannt. Ein Identifizieren und Abtasten von verschwendeten Schlitzen kann ein wichtiger Precursor sein, um Aufgaben zu optimieren, da verschwendete Schlitze nicht nützlich arbeiten, und deshalb die Systemfunktion verschlechtern. Deshalb sind das, allgemein, was hier abgetastet wird, nicht einfach „Ereignisse" oder „Befehle" wie im Stand der Technik, sonder Zustands-Informationen, die dazu in Bezug gesetzt sind, Prozessor-Schlitze durch die Pipeline **200** erzwungenermaßen zu führen, ob sie nun einem gültigen oder einem ungültigen Befehl zugeordnet sind.

Abrufeinheit

[0067] Der B-Cache **121** überträgt Datenelemente zu dem I-Cache **112** und dem D-Cache **113** jeweils. Die Abrufeinheit **210**, die einen Typ eines Translation-Look-Side-Buffer (TLB) **205** verwendet, um virtuelle Adressen zu physikalischen Adressen aufzulösen, ruft nächste Befehle, die ausgeführtwerden sollen, von den I-Cache **112** ab. Die Elemente, die von den I-Cache **112** abgerufen sind, sind allgemein ausführbare Befehle. Allerdings können diese auch ungültige Befehle sein, wie zum Beispiel in dem Fall, dass dem I-Cache, „Abfall" Daten fehlen, d. h. kein Befehl.

[0068] Vorzugsweise wird ein Satz von „Befehlen" während eines einzelnen Prozessor-Zyklus abgerufen. Der Satz kann, zum Beispiel, vier Befehle umfassen. Mit anderen Worten ist die Pipeline **200** vier Schlitze breit. Andere Typen von Prozessoren können weniger oder mehr Befehle während eines einzelnen Prozessor-Zyklus abrufen. Allgemein bedeutet dies, dass jeder Zyklus vier Verarbeitungs-Schlit-

ze von dem Cache ausfüllt. Einige der Schlitze können dann vergeudet werden, wenn der I-Cache **112** nicht die verfügbaren Daten hat. An Stelle eines Pausierens, was die gesamten Verarbeitung anhält, werden die Schlitze nach vorne in jedem Fall getragen, um sie für den Zweck einer Abtastung verfügbar zu machen, obwohl ein Abfall „Befehl" in einem Schlitz niemals für eine Ausführung ausgegeben werden kann.

[0069] Während eines Abrufens können ausgewählte Befehle mit zusätzlichen Informationen erhöht werden, um eine Abtastung oder eine System-Profilierung zu ermöglichen. Ein erhöhter Befehl wird nachfolgend unter Bezugnahme auf **Fig.** 4 beschrieben. Es sollte angemerkt werden, dass, in anderen Ausführungen, die Erhöhung der ausgewählten Befehle in irgendeiner der Stufen des Prozessors stattfinden kann, einschließlich der Ausgabeeinheit **230**.

Auflistungs-Einheit

[0070] In dem System **100** werden die Operanden von Befehlen dynamisch physikalischen Registern unter Verwendung der Auflistungs-Einheit **220** in der nächsten Stufe der Pipeline **200** zugeordnet oder „aufgelistet". Die Auflistungs-Einheit ordnet physikalische Register zu architekturmäßigen oder „virtuellen" Registern zu. Mit anderen Worten kann dabei keine eins zu eins Korrespondenz zwischen virtuellen und physikalischen Registern vorhanden sein.

Ausgabeeinheit

[0071] In der nächsten Stufe werden die abgerufenen Befehle durch eine Ausgabeeinheit **230** geordnet. Die Ausgabeeinheit **230** umfasst eine Ausgabe-Warteschlange, die einen Eintritt **231** am Kopf der Warteschlange (head-of-the-queue entry) für den nächsten Befehl, der ausgeführt werden soll, besitzt. Es sollte angemerkt werden, dass einer oder mehrere Befehle) in der Ausgabeeinheit **230** zum Stillstand kommen können, da Ressourcen oder Daten, die durch die Befehle benötigt werden, nicht verfügbar sind. Deshalb können andere, anhängige Befehle außerhalb der Reihenfolge von der Warteschlange **230** „um" die angehaltenen Befehle herum ausgegeben werden. Die korrekte Ausführungs-Reihenfolge wird in der Ausscheidungseinheit (retire unit) **250**, die nachfolgend beschrieben ist, bestätigt werden.

Ausführungs-Einheiten

[0072] Die Befehle werden zu funktionalen Ausführungs-Einheiten (E0, ..., E3) **241** und einer Lade/Speicher-(Id/st)-Einheit **242** ausgegeben. Jede der Ausführungs-Einheiten **241** kann so ausgelegt sein, um Befehle mit spezifischen Typen von Operator-Coden (opcoden), zum Beispiel Ganzzahl und Fließpunkt-Arithmetik-Verzweigungs- und Sprung-Befehlen, usw., zu handhaben. Zwischenwerte können, während durch die Ausführungs-Einheiten verarbeitet wird, erzeugt werden. Die Id/st-Einheit **240** führt Speicher-Zugriffs-Befehle, zum Beispiel Laden und Speichern von Daten von und zu dem D-Cache **113**, aus. Die Id/st-Einheit **242** wird speziell identifiziert, da sie lange Verzögerungen erleiden kann. Auch ist anzumerken, dass Speicher-Zugriffs-Befehle mit langen Latenzzeiten „vollständig" lang sein können, bevor die Daten in den Prozessor gebracht werden, um einen Durchsatz zu verbessern.

Ausscheide-Einheit

[0073] Die Beendigung einer Ausführung eines Befehls wird durch die Ausscheide-Einheit **250** gehandhabt. Die Ausscheide Einheit **250** überträgt die Verarbeitungsstufe. Es sollte angemerkt werden, dass einige Befehle ausgesondert oder übersprungen werden sollten. Zum Beispiel kann sich der Ausführungsfluss ändern, nachdem ein Befehl abgerufen ist, oder ein Befehl kann ein Ausnahme-Überspringen erleiden. In diesen Fällen werden der Befehl und alle darauffolgenden Befehle, die in der Pipeline bereits ausgesondert sind, und der spekulative Verarbeitungs-Zustand, zurückgerollt. Als ein Vorteil hier sind die ausgesonderten oder „aussortierten" Befehle auch profiliert, wie verschwendete oder ausgelassene Prozessor-Schlitze. Mit anderen Worten kann eine Beendigung ein Aussondern eines vollständig ausgeführten, gültigen Befehls, eine Nachbearbeitung eines teilweise ausgeführten, gültigen Befehls oder ein Aussondern eines ungültigen Befehls oder eines verschwendeten oder ausgelassenen Schlitzes bedeuten.

[0074] Die Grundidee, die hinter der vorliegenden Technik steht, folgt der Verarbeitung von „Datenelementen" in ausgewählten „Schlitzen", Primär-Befehlen, wenn sie durch die Stufen der Pipeline **200** fortschreiten. Eine Profilierungs-Hardware sammelt dynamisch detaillierte Zustands-Informationen. Die Zustands-Informationen können von irgendeiner der Pipeline-Stufen her kommen, oder von irgendwo in dem System **100**, zum Beispiel von dem Cache auf dem ersten und dem zweiten Niveau, oder anderen Untersystemen. Die Zustands-Informationen können direkt zu spezifischen Ereignissen als Attribut zugeordnet werden.

[0075] Hierbei ist die Design-Strategie diejenige, Informationen zusammenzustellen, was schwierig statisch in einer Profil-Aufzeichnung zu bestimmen ist. Dies gestaltet die Profil-Aufzeichnung für Funktions-Tools, für eine auf ein Profil gerichtete Optimierung oder zum Vornehmen von Ressource-Zuordnungs-Policy-Entscheidungen in Betriebssystemen oder eine Software auf einem Anwendungs-Level, nützlich, einschließlich dynamischer Einstellungen direkt in Abhängigkeit der Abtastung und Analyse. Es wird daran erinnert, dass das vorliegenden Verfahren und die vorliegende Vorrichtung so ausgelegt sind,

um auf realen, funktionalen Befehlen zu arbeiten.

[0076] Um zu bestimmen, welche Zustands-Informationen dahingehend von Interesse sind, um sie als Teil der Profil-Aufzeichnung zu sichern, ist es nützlich, die Informationen zu prüfen, die theoretisch für die verschiedenen Stufen der Pipeline **200** eines modernen Mikroprozessors, außerhalb der Reihenfolge, sind, wie dies in **Fig.** 2b dargestellt ist.

[0077] Wie in **Fig.** 2b dargestellt ist, sind die Stufen der Pipeline das Abrufen **210**, das Auflisten **220**, das Ausgeben **230**, das Ausführen **240** und das Ausscheiden **250**. Während irgendeiner der Stufen kann, in Abhängigkeit von der bestimmten Ausführung, irgendein Befehl **202** „in-flight", verarbeitet durch die Pipeline **200**, zum Abtasten durch die Leitung **512** ausgewählt werden. Die Auswahl wird durch einen Wert eines Zählers **510** kontrolliert. Der Wert des Zählers kann durch die Leitung (init) **511** initialisiert werden.

[0078] Zustands-Informationen, wie beispielsweise Befehls-Adressen (PC) **281**, Verzweigungs-Historie-Bits (branch history bits – HIST) **282**, Stufen-Latenzzeiten (stage latencies) **283**, Anzeige über die vorgenommene Verzweigung (T) **287**, Daten-Adresse (ADDR) **284**, Daten-Fehler (MISS) **285**, Ausscheidungs-Status **286**, können auf Leitungen **288** abgetastet werden. Eine Beendigung der Verarbeitung der ausgewählten Befehle kann ein Unterbrechungs-Signal auf der Leitung **289** erzeugen. Das Unterbrechungs-Signal **289** kann bewirken, dass die Software die Zustands-Informationen **281**–**286** über Leitungen **299** abtastet.

[0079] Alternativ kann eine Software die Leitung **289** über ein internes Prozessor-Register **541** abfragen.

Superscalar-Out-of-Order-Prozessor-Architektur

[0080] Ein Out-of-Order-Ausführungs-Prozessor bzw. ein Prozessor mit einer Ausführung außerhalb der Reihenfolge ruft Befehle in der Reihenfolge ab und scheidet sie aus, führt sie allerdings entsprechend deren Daten-Abhängigkeiten aus. Ein Befehl ist dasjenige, dass er sich von der Zeit an „in Bewegung" („in-flight") befindet, zu der er abgerufen ist, bis er endet, z. B. ausscheidet, oder ausgesondert wird. Befehle werden, nach einem Auflisten, in der Ausgabeeinheit **230** platziert, und warten dort, bis Register, die Eingangs-Operanden halten, aktualisiert werden.

[0081] Bei jedem Prozessor-Zyklus ruft die Abrufeinheit **210** einen Satz von Befehlen von dem Befehls-Cache **112** ab und codiert sie. Der Befehl-Decodierer, der ein Teil der Abrufeinheit **210** sein kann, identifiziert, welche Befehle in dem abgerufenen Satz ein Teil der Befehl-Datenfolge ist. Da es mehrere Zyklen benötigt, um den Programm-Zähler (PC) eines nächsten Befehls aufzulösen, um abgerufen zu werden, wird der nächste PC durch einen Verzweigungs- oder Sprung-Prädikator vorhergesagt, der Teil der Abrufeinheit **210** sein kann. Falls die Vorhersage nicht korrekt ist, dann wird der Prozessor die fehlinterpretierten Befehle aussondern, d. h. der Befehl, abgerufen auf einem „schlechten" Pfad, und wird ein Abrufen von Befehlen auf dem „guten" Pfad erneut starten.

[0082] Um zu ermöglichen, einen Vorgang Out-of-Order auszuführen, werden Register dynamisch durch die Abbildungseinheit **220** umbenannt, um Konflikte eines Schreibens nach einem Lesen, und eines Schreibens nach einem Schreiben, zu verhindern. Zwei Befehle, die dasselbe, virtuelle Register beschreiben, können sicher außerhalb der Reihenfolge (out-of-order) ausgeführt werden, da sie unterschiedliche, physikalische Register beschreiben werden, und Verbraucher der virtuellen Register werden die geeigneten Werte erhalten. Befehle werden abgerufen, aufgelistet bzw. abgebildet und erneut versucht, und zwar in der Reihenfolge, obwohl sie außerhalb der Reihenfolge ausgeführt werden können.

[0083] Die Register-Abbildungseinheit **220** ordnet Operanden von abgerufenen Befehlen gültigen, physikalischen Registern zu. Das bedeutet, dass die virtuellen Namen der Register-Operanden zu dem physikalischen Registerraum des Prozessors umbenannt werden. Befehle gehen dann zu der Befehl-Warteschlange **230** weiter, wo sie auf zwei Ereignisse, vor einer Ausführung, warten. Zuerst müssen deren Register-Abhängigkeiten aufgelöst werden. Als zweites müssen die Ressourcen, die den Befehl benötigen, z. B. Ausführungseinheiten, Register, Cache-Ports, Speicher-Warteschlangen, usw., verfügbar sein. Dies bedeutet, dass die erforderlichen Ressourcen nicht für irgendwelche momentan aufgelisteten Befehle erneut zugeordnet werden können.

[0084] Wenn diese zwei Bedingungen für einen Befehl erfüllt sind, werden die Operanden des Befehls in der physikalischen Register-Datei durchgesehen. Der Inhalt der Operanden-Register und von einigen Informationen über den Befehl werden dann zu einer geeigneten Ausführungseinheit **240** geschickt und ausgeführt. Wenn der Befehl in seiner Ausführung beendet ist, und der Befehl der älteste „nicht ausgeschiedene" Befehl in dem Prozessor ist, wird der Befehl ausgeschieden. Dies gibt die Ressourcen, verwendet durch den Befehl, frei, wie beispielsweise physikalische Register und Cache-Ports.

[0085] Zahlreiche Ereignisse können die Ausführung eines Befehls verzögern. Vor der Pipeline kann die Abrufeinheit **210** aufgrund eines Fehlers eines I-Cache **112** steckenbleiben bzw. blockieren oder die Abrufeinheit **210** kann Befehle eines fehlerhaft vorhergesagten Wegs abrufen. Die Abbildungseinheit **220** kann aufgrund eines Fehlens von freien, physikalischen Registern, oder aufgrund eines Fehlens von freien Schlitzen in der Ausgabeeinheit **230**, blockieren.

[0086] Befehle in der Ausgabeeinheit **230** können darauf warten, dass deren Register-Abhängigkeiten erfüllt werden, oder auf die Verfügbarkeit von Ausführungseinheiten **240**. Befehle können aufgrund von

Fehlern in dem D-Cache blockieren. Befehle können abgefangen werden, da sie spekulativ entlang eines schlechten Pfads ausgegeben wurden, oder da der Prozessor eine Unterbrechung vornahm, wie beispielsweise eine illegale Operation oder eine Speicheradresse. Viele dieser Zustände sind schwierig zum Zeitpunkt der Zusammenstellung vorherzusagen, und alle davon verschlechtern die Funktionsweise des Systems **100**. Dies gestaltet es lohnenswert, die Informationen, verfügbar auf Leitungen **288**, abzutasten.

Profil-Informations-Register

[0087] Deshalb ist, wie in **Fig.** 3 dargestellt ist, ein Speicher **300** zum Speichern von Profil-Informationen für jeden Befehl, der abgetastet werden soll, vorgesehen. Der Speicher **300** kann in der Form einer Register-Datei oder eines Puffers vorliegen. Mit anderen Worten wird ein ausgewählter Befehl, der abgetastet werden wird, direkt mit der Register-Datei **300** identifiziert werden. Die Register-Datei **300** kann eine Vielzahl von Registern umfassen. Alternativ kann die Datei **300** als ein einzelnes, indexierbares Register mit mehreren Feldern ausgeführt werden.

[0088] Die Datei **300** ist mit den Komponenten der Pipeline **200** durch Leitungen **288** der **Fig.** 3b gekoppelt, so dass Funktions-Informationen, die zu dem ausgewählten Befehl in Bezug gesetzt sind, für jede Stufe der Pipeline **200** erfasst werden können. Es sollte angemerkt werden, dass die Profil-Register **300** mehr als einfache „Ereignis" Zähler sind, wie dies im Stand der Technik vorgefunden wird, wobei hier die Register Funktions-Informationen zusammenstellen, die zu spezifischen, bekannten Befehlen und Ereignissen zuordenbar sind.

[0089] In **Fig.** 3 hängt die Zahl von Bits, die für jedes Register zugeordnet ist, von dem Typ von Informationen, die darin gespeichert sind, ab, zum Beispiel Befehls-Adressen (64 Bits), Latenzzeiten, d. h. Zyklus-Zählungen (8 oder 10 Bits), diskrete Ereignisse (1 Bit pro Ereignis), usw.. Diese Zahlen sind nur eine Richtlinie. Andere Ausführungen können unterschiedliche Zahlen von Bits für die verschiedenen Register **300** verwenden, wobei dies eine Designwahl ist.

[0090] In der bevorzugten Ausführungsform speichert ein Profil-PC-Register **310** den PC des ausgewählten Befehls. Wie nachfolgend beschrieben ist, besitzt ein Befehl, der profiliert werden soll, ein „Profil" Bit, das zugeordnet ist. Das PC-Register **310** kann auch den Opcode des ausgewählten Befehls umfassen. Zusätzlich können Prozessoren, die eine Multi-Threaded-Ausführung von zusätzlichen Bits des Registers **300** ermöglichen, den Identifizieren des Thread speichern. Andere Felder des Registers **310** können den Prozess-Identifizierer, die Adressen-Raum-Zahl, die CPU-Zahl und die Befehls-Zahl (inum) des Befehls, der ausgeführt werden soll, speichern. Zusätzlich kann, bei Prozessoren, die mehrere, logische Register-Sätze haben, d. h. Hardware-Zusammenhänge, und gleichzeitig Threads ausführen, das Register **310** einen Hardware-Kontext und Thread-Identifizieret speichern. Durch Speichern dieser Informationen können die Profil-Informationen direkt einem spezifischen Befehl zugeordnet werden. Zusätzlich können die abgetasteten Informationen entsprechend zu einem Bereich von Adressen, einem Opcode, Ausführungs-Threads, Adressen-Räumen, und dergleichen, gefiltert werden.

[0091] Ein profil-effektives Adressen-Register **320** wird mit einer Adresse, zugeordnet dem ausgewählten Befehl, geladen. Falls der Befehl ein Speicher-Zugriffs-Befehl ist, dann kann die physikalische Adresse, die sich aus der Translation der virtuellen Speicher-Adresse ergibt, in dem Register **320** erfasst werden. Falls der Befehl ein Sprung oder eine Verzweigung ist, dann kann die physikalische Adresse, die sich aus der Translation des virtuellen Ziel-PC ergibt, in dem Register **320** erfasst werden.

[0092] Als ein Vorteil der vorliegenden Abtasttechnik ist es möglich, über alle „Befehle", verarbeitet durch die Pipeline **200**, abzutasten, unabhängig von der Abtastrate. Die Befehle können gültige Befehle, ungültige Befehle, nicht-unterbrechbare Befehle, oder „Abfall" („garbage") Befehle sein. Demzufolge sind die erfassten, effektiven Adressen statistisch für das Gesamtverhalten des Programms repräsentativ. Durch Erfassen der effektiven Adressen von abgetasteten Befehlen können Speicherzugänge und Ausführungs-Abläufe präzise zu aktuellen, dynamischen Ausführungen korreliert werden.

[0093] Ein profiliertes Ereignis-Register **330** wird in, zum Beispiel, Ein-Bit-Felder unterteilt. Die 1-Bit-Felder zeichnen Ereignisse für den ausgewählten Befehl auf. Wenn ein Befehl zuerst ausgewählt ist, wird das Register gelöscht. Ereignisse könnten Cache-Fehler, Verzweigungs-Fehlvorhersagen, Ressource-Konflikte, Traps- und Ausführungs-Bedingungen, ausscheiden/aussondern/ungültig, TLB-Fehler, genommen/nicht genommen, Daten-Abhängigkeits-Blockierung, Ressovre-Abhängigkeits-Blockierungen, usw., sein. Es ist anzumerken, dass diese Ausführung ermöglicht, dass mehrere Ereignisse einem einzelnen Befehl zuordenbar sind. Es sollte angemerkt werden, dass Ereignis-Informationen für sowohl ausgeschiedene als auch ausgesonderte Befehle zusammengestellt werden. Um die Größe des Ereignis-Registers **330** zu verringern, können einige der Bit-Felder dazu verwendet werden, unterschiedliche Typen von gegenseitig exklusiven Ereignissen, in Abhängigkeit von dem opcode des Befehls, aufzuzeichnen.

[0094] Ein profiliertes Pfad-Register **340** wird dazu verwendet, um neuere genommene/nicht genommene Verzweigungs-Informationen von einer Verzweigungs-Historie-Tabelle zu erfassen. Verzweigungs-Historie-Tabellen sind im Stand der Technik für andere Verwendungen ausreichend bekannt. Eine herangezognene Historie einer globalen Ver-

zweigung kann dazu verwendet werden, den Ausführungs-Pfad anzuzeigen, der den ausgewählten Befehl verursachte, um abgerufen zu werden. Es ist anzumerken, dass der Befehl nicht ein Verzweigungs-Befehl für diese Informationen sein muss, um nützlich zu sein. Die Verwendung der Pfad-Informationen ist in größerem Detail nachfolgend beschrieben.

[0095] Latenzzeit-Register **350** speichern Zeitabstimmungs-Informationen, die an Prüfpunkten herangezogen werden, während sich ein ausgewählter Befehl in Bewegung befindet, z. B. zwischen den verschiedenen Stufen der Pipeline **200**. Die Prüfpunkte können sich von Prozessor zu Prozessor in Abhängigkeit davon unterscheiden, ob ein Befehl angehalten werden sollte, auf ein bestimmtes Ereignis oder eine Ressource wartend. Jedes Latenz-Register **350** zählt die Zahl von Zyklen eines Befehls, verbraucht zwischen zwei Prüfpunkten.

[0096] Wenn der ausgewählte Befehl einen Prüfpunkt passiert, d. h. in eine nächste Stufe in der Pipeline **200** eintritt, wird das entsprechende Latenz-Register **350** zuerst gelöscht und dann einmal pro Zyklus erhöht, bis der Befehl den nächsten Prüfpunkt passiert, wenn das nächste Latenz-Register initialisiert wird und eine Zählung beginnt. Die Zahl von Latenz-Registern **350** hängt von der Zahl von Stufen der Pipeline **200** in einer bestimmten Ausführung ab. Ein vollständiges Latenz-Profil ist in den Latenz-Registern **350** gespeichert, wenn der Befehl ausgesondert wird oder ausscheidet.

[0097] Eine Liste von potenziell nützlichen Latenzzeiten, um sie zusammenzustellen, umfasst: fetch-to-map, map-to-data-ready, data ready-to-execute, execute-to-retire ready, retire ready-to-retire delays. Für Speicher-Befehle (Laden und Speichern) können Latenzzeiten ausgegeben werden, um abzuschließen (issue-to-completion). Diese letzte Latenzzeit unterscheidet sich von anderen Latenzzeiten dahingehend, dass einige Speicher-Operationen ausgeschieden werden können, bevor die Daten, auf denen sie gearbeitet haben, tatsächlich in den Prozessor gebracht worden sind. Diese Latenzzeiten könnten direkt in den Registern **350** gezählt werden, oder die Register können grobe Zyklus-Zählungen zusammenstellen, wobei in diesem Fall die Profilierungs-Software Unterschiede zwischen groben Zählungen für aufeinanderfolgende Stufen berechnet, um tatsächliche Latenzzeiten zu bestimmen. Eine Schaltung, die beispielhafte Pipeline-Latenzzeit-Taktzyklen zählt, wird nachfolgend unter Bezugnahme auf **Fig.** 6 beschrieben.

[0098] Das Aktualisieren der Informationen in dem Register (den Registern) **300** muss nicht unmittelbar auftreten, wobei eine Verzögerung akzeptierbar ist. Alles das, was erforderlich ist, ist, dass die Unterbrechung, die die Tatsache signalisiert, dass der ausgewählte Befehl abgeschlossen ist (ausgeschieden oder ausgesondert ist), verzögert wird, bis alle Informationen in der Register-Datei **300** aktualisiert worden sind, oder der Unterbrechungs-Handler kann anhalten, bis die Profil-Datei **300** aktualisiert worden ist.

[0099] Es sollte angemerkt werden, dass die Profil-Register-Datei **300** repliziert werden kann. Falls mehrere Kopien der Profil-Register-Datei vorhanden sind, dann können mehrere Befehle für ein Profilieren, entweder seriell, oder gleichzeitig, ausgewählt werden. In diesem Fall wird jeder ausgewählte Fall explizit mit einer spezifischen Register-Datei so, wie dies nachfolgend beschrieben ist, identifiziert. Mehrere Register-Dateien können auf ein einzelnes Unterbrechungssignal hin abgetastet werden, um den Umfang eines Overhead zu verringern.

Identifizieren eines ausgewählten Befehls

[0100] Wie in **Fig.** 4 dargestellt ist, umfasst jeder Befehl **400** ein Abtastfeld. Zum Beispiel kann das Abtastfeld ein Ein-Bit-Tag (Ein-Bit-Zeichen), bezeichnet als „Abtast"-Bit (S) **401** sein. Wenn das Abtast-Bit **401** aufgestellt ist, wird der Befehl zum Abtasten ausgewählt. Ein Aufstellen des Bits **401** aktiviert die Abtast-Hardware, die die Profil-Informationen zusammenstellt, und bewirkt auch die Unterbrechung, wenn der ausgewählte Befehl abschließt (ausgeschieden oder ausgesondert wird). Alternativ kann jeder „Befehl", der abgerufen ist, aufeinanderfolgend mit einem „inum" Wert nummeriert werden. In diesem Fall können Befehle mit spezifischen inum Werten ausgewählt werden. Der Mechanismus zum Auswählen von Befehlen wird nachfolgend beschrieben.

[0101] Die Profil-Register-Datei **300** kann dann gelesen werden, wenn die Felder aktualisiert worden sind und das Unterbrechungssignal erzeugt ist. Das Unterbrechungssignal kann bewirken, dass die privilegierte Profilierungs-Software (PSW) die Inhalte der Profil-Register **300** verarbeitet. Es sollte angemerkt werden, dass, in dem Fall, bei dem mehrere Abtastungen aufgezeichnet werden, eine einzelne Unterbrechung die Abtastung der Funktions-Daten für mehrere, ausgewählte Befehle bewirken kann.

[0102] In Abhängigkeit von der Ausführung kann der erhöhte Befehl **400** die folgenden, zusätzlichen Felder, bis zu drei Befehl-Operanden (op1, op2 und op3) **411**–**413**, den Programm-Zähler (PC) **420**, den Operator-Code (opcode) **430**, umfassen. Ein gültiges Feld M **431** kann anzeigen, ob der „Befehl" in dem ausgewählten Schlitz gültig ist oder nicht, und zwar durch Einstellen eines Ein-Bit-Felds auf entweder wahr oder falsch. Die Felder **440** und **450** können zum Anzeigen eines Befehls, der zu einem I-Cache und TBL-Fehler, jeweils, in Bezug gesetzt ist, umgekehrt werden. Es ist anzumerken, dass, da eineinzelner Befehl mehrere Operanden umfassen kann, mehrere Fehler für diesen Befehl möglich sind.

Profil-Register-Datei-ID

[0103] In einem leicht komplizierteren Design können mehrere Befehle gleichzeitig profiliert werden.

Bei dieser Ausführungsform ist eine Vielzahl von Register-Dateien **300** oder einzelnen, größeren Registern mit Unterfeldern vorhanden, wobei die Zahl von Dateien **300** der Zahl von sich in der Bearbeitung (in-flight) befindlichen Befehlen entspricht, die gleichzeitig profiliert werden können. Um diesen Fall zu handhaben, wird Befehl **400** auch erhöht, um ein Abtast-Register-Datei-Indentifizierer-(ID)-Feld **402** zu umfassen. Dies ermöglicht auch, dass Profil-Informationen direkt mit einer der verschiedenen Register-Dateien verknüpft werden. Wie vorstehend angeführt ist, ist dabei eine direkte Zuordnung zwischen ausgewählten Befehlen und Profil-Registern vorhanden. Die Profil-Informationen, zusammengestellt in den Registern, sind deshalb direkt einem spezifischen Befehl zuordenbar.

[0104] Gerade wenn nur ein sich in der Verarbeitung befindlicher Befehl zu einem Zeitpunkt profiliert wird, kann es nützlich sein, die Datei oder das Register **300** durch das ID-Feld **402** indexiert zu haben, so dass die Kosten des Unterbrechungs-Handlex der Profilierungs-Software über mehrere Befehl-Abtastungen amortisiert werden können. Um zu bestimmen, ob ein Befehl innerhalb eines Satzes von Befehlen liegt, kann ein ausgewählter Befehl unter Verwendung einer „verdrahteten ODER" („wired-OR") Operation durchgeführt werden.

Zufall-Abtastung

[0105] Das Overhead der vorliegenden Profilierung wird durch Einschränken der Zahl von Befehlen, die gleichzeitig profiliert werden können, z. B. Bit **401** wird eingestellt, reduziert. Anstelle eines Profilierens jedes Befehls in einem Programm oder einem Teil des Programms werden hier Befehle, die profiliert werden sollen, während einer spezifischen Stufe der Prozessor-Pipeline **200**, ausgewählt, z. B. während eines Abrufens, und die ausgewählten Befehle werden durch Aufstellen des Abtast-Bits **401** mit einem Zeichen versehen. Falls das Abtast-Bit **401** aufgestellt ist, dann führen die Komponenten der Pipeline **200** die Profil-Informationen zu der (den) Profil-Register-Dateien) **300** weiter.

[0106] Die nachfolgenden Abschnitte beschreiben die unterstützenden Details einer Befehl-Level-Profilierung, wie es hier beschrieben ist.

In der Bearbeitung befindliche Zustände

[0107] Zuerst wird jeder decodierte Befehl-Zustand, der durch die Prozessor-Pipeline **200** hindurchführt, mit zusätzlichen Informationen ergänzt bzw. erhöht, wie dies vorstehend beschrieben ist. Ein Befehl wird dahingehend angesehen, dass er sich von der Zeit an in der Bearbeitung befindet, zu der er abgerufen ist, bis er ausgeschieden oder ausgesondert wird. Wie vorstehend angegeben ist, wird der Befehl mit zumindest einem Abtast-Bit **401** erhöht bzw. ergänzt. Das Abtast-Bit **401** ist Teil des Zustands von jedem sich in der Bearbeitung befindlichen Befehl und einer Cache/Speicher-Anforderung. Wenn das Bit **401** aufgestellt ist, zeigt das Bit an, dass Profilierungs-Informationen für diesen Befehl aufgezeichnet sind, ansonsten nicht.

[0108] In einem vereinfachten Design wird nur einem sich in der Bearbeitung befindlichen Befehl zu einem Zeitpunkt ermöglicht, dass sein Abtast-Bit **401** aufgestellt ist. Das Abtast-Bit **401** verbleibt für den ausgewählten Befehl aufgestellt, bis der Befehl ausscheidet oder ausgesondert wird. In einem komplexeren Design mit Mehrfach-Register-Dateien **300** können die mehrfachen, sich in der Bearbeitung befindlichen Befehle individuell profiliert werden und zusätzliche Bits können aufgestellt werden.

Auswahl und Abtasten eines profilierten Befehls

[0109] Wie in **Fig.** 5 dargestellt ist, schreiten, für eine Abruf-Stufen-Ausführung, eine Auswahl von Befehlen, die profiliert werden sollen, und ein Abtasten von Profil-Informationen so fort, wie dies nachfolgend angegeben ist. Ein Abruf-Zähler **510** wird durch, zum Beispiel, privilegierte Profilierungs-Software (PSW) **520** über die Leitung **511** initialisiert. Die PSW **520** kann den Zähler **510** mit einem Wert, zufällig ausgewählt von einem Intervall von Werten, das eine vorbestimmte Größe besitzt, initialisieren. Demzufolge werden die abgetasteten Befehle nicht mit irgendwelchen spezifischen Mustern in der Ausführung von Befehlen korrelieren. Die Größe des Intervalls bestimmt die durchschnittliche Frequenz einer Abtastung. Die Größe des Intervalls kann variiert werden. Andere Zufalls-Techniken, um den Wert des Zählers **510** zu initialisieren, umfassend eine Hardware, können auch verwendet werden.

[0110] Ohne eine zufällige Abtastung kann es, zum Beispiel dann, wenn Befehle unter einer festgelegten Frequenz, wie im Stand der Technik, abgetastet werden, nicht möglich sein, ein statistisch korrektes Profil aller Befehle, die abgerufen sind, z. B. die aggregierte Operation des Systems **100**, zu erzeugen. Dies gilt insbesondere für ein Ausführungs-Thread bzw. einer Ausführungsfolge, die Ausführungsschleifen besitzt, die eine Anzahl von Befehlen umfasst, die nicht sehr wichtig in Bezug auf die Rate einer Abtastung sind, z. B. für eine Schleife mit zwei Befehlen und einem Abtastintervall von 65536 Befehlen. Als ein Vorteil werden zufällig ausgewählte Befehle Korrelationen, unabhängig der Länge des Abtastintervalls, erzeugen.

[0111] Für jeden Befehl **400**, der abgerufen ist, wird der Zähler **510** erhöht, oder, alternativ, in einer unterschiedlichen Ausführung, von seinem Anfangswert erniedrigt, und zwar durch die Abrufeineinheit **210** der Pipeline **200**. Wenn der Zähler **510**, in Abhängigkeit von der Ausführung, entweder überläuft oder unterläuft, besitzt der momentan abgerufene Befehl sein Abtast-Bit **401** aufgestellt, und das ID-Feld **402** kann auch initialisiert werden, wenn mehrere Befehle zum Abtasten ausgewählt werden.

[0112] In einer alternativen Ausführungsform wird der Zähler **510** jeden Zyklus erhöht, anstelle davon, dass dies für jeden Befehl, der abgerufen ist, erfolgt, z. B. der Zähler **510** zählt Abruf-Gelegenheiten und nicht tatsächliche Befehle, die abgerufen sind. Zum Beispiel sind, falls die Abrufeinheit **210** vier Elemente von dem 1-Cache **112** während jedes Taktzyklus abrufen kann, dann vier Abruf-Gelegenheiten vorhanden. Es kann ausreichend sein, dass ein oder mehrere Abrufvorgang (Abrufvorgänge) von dem I-Cache fehlen werden, oder ein „schlechter" Befehl abgerufen wird. In dem Fall eines Fehlens wird der Schlitz, der für den fehlenden Befehl verfügbar ist, „Abfall" („garbage") enthalten, und der Befehl muss als ungültig markiert werden. Ein schlechter Befehl ist ein solcher, der auf einem schlechten Ausführungs-Pfad liegt, oder ansonsten ausgesondert werden wird.

[0113] Ein Zählen von Zyklen, anstelle von abgerufenen Befehlen, vereinfacht vorteilhaft das Design. Nur ein Zählen von gültigen, abgerufenen Befehlen kann ziemlich kompliziert sein, da der Steuerablauf in die Gruppe von abgerufenen Befehlen hinein verzweigen und davon heraus verzweigen kann, und es wird notwendig, alle Befehle zu decodieren, um zu bestimmen, welche gültig sind, so dass dies nicht länger ein einfacher Vorgang von nur einem Erhöhen des Zählers um vier ist.

[0114] Als ein Vorteil kann irgendetwas (gute Befehle, schlechte Befehle, Abfall-Befehle), das von dem I-Cache während eines Zyklus abgerufen ist, zum Abtasten ausgewählt werden, so dass die wahre Funktion des I-Cache **112** und der Pipeline **200** bestimmt werden kann. Hierbei ist kein Bias vorhanden, so dass die Ergebnisse statistisch korrekt die Funktionsweise des Systems abschätzen werden.

Filter-Befehle

[0115] Die abgetasteten Befehl-Informationen können durch ein Filter **505** gefiltert werden. Ein Filtern kann auf der Basis eines Befehl-Opcodes, von Operanden, oder durch noch komplexere Filterkriterien, wie beispielsweise einen ersten Typ eines Befehls gefolgt durch einen anderen Typ eines Befehls, innerhalb einer bestimmten Zeitperiode, erfolgen. Falls ein Filtern an dem Eingang der Pipeline **200** vorhanden ist, dann kann der Zähler **510** zurückgesetzt werden. Dabei ist eine Anzahl von Arten und Weisen vorhanden, um dies so vorzunehmen. In einer Art und Weise wird der momentane Anfangswert des Zählers **510** in einem init-Register **513** gespeichert. Wenn ein Befehl gefiltert wird, wird der Zähler **510** erneut mit dem Wert, gespeichert in dem init-Register **513**, geladen, um die anfängliche, randomisierte Auswahl wieder aufzurufen.

[0116] Nachdem der Befehl erhöht worden ist, liefert die Pipeline **200** die Profilierungs-Informationen **281**–**286** der **Fig.** 2b zu der Register-Datei (den Register-Dateien) **300**. Die Ausscheidungseinheit **250** schließt, auf den Befehl-Abschluss oder die Befehl-Aussonderung hin, das Auffüllen der Profil-Informationen ab und erzeugt ein Unterbrechungssignal auf der Leitung **540**, so dass die PSW **520** die Profil-Informationen abtasten kann.

[0117] Alternativ kann die PSW **520** die Leitung **540** über ein internes Prozessor-Register oder eine Speicherstelle (**541**) abrufen. Als ein Merkmal der vorliegenden Technik ist, im Gegensatz zu Profilierungstechniken nach dem Stand der Technik, kein Einfluss in Bezug auf die Prozessor-Zykluszeit vorhanden, gerade obwohl die vorliegenden Technik präzise Informationen über Zustände über den Prozessor liefert. Die einzige Zeiteinschränkung ist diejenige, dass alle Profil-Informationen aufgezeichnet werden müssen, bevor die Profil-Register **300** abgetastet werden.

Latenz-Zähler

[0118] **Fig.** 6 stellt eine Schaltung **600** zum Zählen von beispielhaften Latenzen bzw. Latenzzeiten dar: fetch-to-map (FM), map-to-issue (MI), issue-to-retire (IR), fetch-to-trap (FT), und issue-to-1dst (ILS). Die Schaltung **600** umfasst einen Zyklus-Zähler **610**, verbunden über eine Leitung **611** mit Verriegelungen **620**.

[0119] Der Zyklus-Zähler **610** und die Verriegelung **620** werden durch ein Signal Pfetch auf der Leitung **601** initialisiert. Dieses Signal wird dann erzeugt, wenn ein Befehl, der profiliert werden soll, abgerufen ist, zum Beispiel ein Signal, abgeleitet von dem Abtast-Bit **401**. Der Zähler **610** wird durch Taktsignale auf der Leitung **609** erhöht. Jedes Taktsignal entspricht einem Prozessor-Zyklus.

[0120] Wenn der Befehl **400** durch die Stufen der Pipeline **200** fortschreitet, triggern die Zustands-Übergänge in der Pipeline **200** Signale Pmap, Pissue, Pretire, Ptrap und PLSdone, und zwar jeweils auf den Leitungen **602**–**606**. Die entsprechenden Verriegelungen **620** können auf Leitungen **612**–**616** zum Speichern in den Profil-Latenz-Registern (oder Feldern) **350** der **Fig.** 3 gelesen werden.

Profilierungs-Anwendungen

[0121] Die Profilierungs-(Profiling)-Hardware, die nachfolgend beschrieben ist, kann in einer Vielfalt von unterschiedlichen Arten und Weisen verwendet werden. Da die vorliegende Technik sehr detaillierte Informationen über die Ausführung von individuellen Befehlen liefert, könnte eine Anwendung eine große Anzahl von Befehlen profilieren. Die Abtast-Informationen können in einem Speicherpuffer für eine spätere Verarbeitung durch Profiling-Tools gespeichert werden, um detaillierte Befehl-Level-Informationen zu erzeugen.

[0122] Die Informationen können dazu verwendet werden, zum Beispiel Histogramme von Last-Latenzzeiten für jeden Last-Befehl, Histogramme von Befehl-Ausführungs-Zeiten, und vielleicht sogar eine moderate umfangreiche Analyse des Pipeline-Zu-

stands für jeden Befehl zu entwickeln. Da die Menge an Informationen, vorgesehen durch diese Maßnahme, dahingehend wahrscheinlich ist, dass sie sehr hoch ist, ist das gesamte Speicher-Overhead der vorliegenden Technik auch dahingehend wahrscheinlich, dass es sehr hoch ist, da ein wesentlicher Umfang eines Speicherverkehrs auftritt. Zum Beispiel wird, falls eine Billion Befehle pro Sekunde abgerufen werden, und eine Abtastung alle 10.000 abgerufene Befehle durchgeführt wird, dann die Datenrate für die Profil-Informationen ungefähr 2,4 MB pro Sekunde sein.

[0123] Der nachfolgende Abschnitt beschreibt durch eine Software ausgeführte Verfahren zum Verringern einer Bandbreite durch Aggregieren von Profil-Informationen.

Daten-Reduktion durch Filtern von Profil-Informationen

[0124] Das Volumen bzw. der Umfang von abgetasteten Daten kann durch Ignorieren einiger Felder der Profil-Aufzeichnung verringert werden, z. B. die Daten in den Profil-Registern **300**, mit der Ausnahme dann, wenn sie explizit angefordert werden. Ein Benutzer des Systems **100** kann unterschiedliche Niveaus einer Profilierung wünschen. In einem niedrigsten Overhead-Mode kann die Profilierungs-Mode-Anwendungs-Software einen Profil-Bericht für das gesamte oder einen Teil eines Programms erzeugen, und zwar unter Verwendung nur der PC- und Retire-Delay-Felder. In Abhängigkeit von der Optimierung, die durchgeführt werden soll, können andere per-PC-Werte durch Mitteln oder andere, statistische Metriken, wie beispielsweise Minimum, Maximum oder Berechnen einer Standardabweichung, summiert werden. Falls mehr Zeit bereitgestellt wird, um Daten zu verarbeiten, kann die Profilierungs-Anwendung Histogramme von verschiedenen Befehl-Latenzzeiten erzeugen.

[0125] Die effektive Speicher-Adresse, die Verzweigungs-Soll-Adresse und die Verzweigungs-Historie-Abtastungen werden wahrscheinlich eine teurere Verarbeitung als die anderen Felder erfordern. Diese Felder können wahrscheinlich ignoriert werden, mit der Ausnahme dann, wenn Daten gesammelt werden, um spezifische Optimierungs-Aufgaben durchzuführen. Unter Bereitstellen der Inter-Befehl-Abruf-Distanz zwischen Befehlen in Zyklen, kann die Profilierungs-Anwendung auch Informationen über Level einer Gleichzeitigkeit bzw. Konkurrenz sammeln.

[0126] Ein Filtern der Profilierungs-Informationen kann auch durch Hardware-Einrichtungen vorgenommen werden, zum Beispiel ein Masken-Register, oder eine programmierbare Logik. Zum Beispiel nur Abtastung, wenn ein Cache-Fehler vorhanden ist, oder wenn der Befehl ausgeschieden ist, oder wenn andere Bool'sche Kombinationen von Opcoden, Operanden, Adressen, Ereignissen und Latenzzeiten vorhanden sind.

Bestimmung einer Hardware-Operation

[0127] Die vorliegende Profilierungstechnik kann dazu verwendet werden, ein präzises Verständnis der internen Operation eines Ausgabe-Prozessors, außerhalb der Reihenfolge, wie beispielsweise den Alpha 21264 Prozessor, zu erhalten. Eines der ersten Dinge, das über diesen Typ einer Maschinen-Organisation anzumerken ist, ist dasjenige, dass viele Stellen vorhanden sein können, wo ein Befehl in der Pipeline **200** anhält, und eine große Anzahl von Gründen, warum sie hängen könnten.

[0128] Zum Beispiel könnte ein Befehl in der Ausgabeeinheit **230** anhalten, entweder da einige seiner Operanden keine Daten sind, die bereit sind, da einige der Ressourcen, erforderlich für die Ausführung des ausgewählten Befehls, nicht verfügbar sind, oder da andere Befehle so ausgewählt wurden, um sie davor auszuführen.

[0129] Ein Befehl könnte in der Auflistungs-Stufe hängen bleiben, die virtuell zu einem physikalischen Register Auflistungen vornimmt, entweder weil die Maschine außerhalb der physikalischen Register vorhanden ist, oder weil sich zu viele Befehle gerade in der Ausführung befinden, oder weil die Ausgabeeinheit **230** voll ist; dies bedeutet, dass kein Platz dort vorhanden ist, um den Befehl, der ausgeführt werden soll, einzugeben. Alternativ kann ein Befehl in der Ausscheidungseinheit hängen bleiben, da zuvor ausgegebene Befehle, in der Programm-Reihenfolge, noch nicht abgeschlossen wurden.

[0130] Eine Bestimmung exakt davon, wo ein Befehl hängen blieb, warum er hängen blieb und wie lange er hängen blieb, hängt stark von dem präzisen Zustand der Maschine ab, wenn dieser Befehl ausgeführt wird. Da der Prozessor so dynamisch ist, ist es schwierig für die Software-Funktions-Werkzeuge, diesen Zustand statistisch zu bestimmen.

Zusammenfassung der Betriebsweise

[0131] Wie in **Fig.** 7a dargestellt ist, kann ein Verfahren **700** zum Profilieren die folgenden Schritte umfassen. Der Profilierungs-Zustand wird im Schritt **710** initialisiert. Hierbei werden Register gelöscht und Zähler werden deren individuellen Werten zugeordnet. Im Schritt **720** wird ein Befehl abgerufen und gezählt. Im Schritt **730** wird der Befehl ausgewählt, wenn die Zahl von abgerufenen Befehlen seit einer Initialisierung gleich zu einer vorbestimmten Zufalls-Zahl ist. Der ausgewählte Befehl wird erhöht, um seine Auswahl anzuzeigen.

[0132] Wenn der ausgewählte Befehl durch die Ausführungs-Pipeline **200** fortschreitet, werden Profil-Informationen im Schritt **740** zusammengestellt. Unter Abschluss (ausgeschieden oder ausgesondert) werden die zusammengestellten Informationen im Schritt **750** abgetastet. Abgetastete Informationen können

für eine darauffolgende Verarbeitung gepuffert werden. Es ist auch möglich, einen bestimmten Profilierungs-Zustand abzutasten, um detailliertere Informationen zu extrahieren.

Abschätzungs-Statistiken der Eigenschaften und der verarbeiteten Befehle

[0133] Wie in **Fig.** 7b dargestellt ist, schätzt der Prozess **799** Statistiken über die Eigenschaften von Befehlen, verarbeitet durch die Pipeline **200**, ab. Der Prozess **799** kann die folgenden Schritte umfassen. Schritt **751** liest die Profil-Aufzeichnung **300**, abgetastet so, wie dies vorstehend im Schritt **750** beschrieben ist. Die Aufzeichnung wird dann gelesen, wenn der ausgewählte Befehl abschließt. Im Schritt **760** wird die Abtastung ausgewählt oder ausgesondert, und zwar in Abhängigkeit von einer Funktion **755**, die Zustand-Informationen des Systems berücksichtigt.

[0134] Zum Beispiel nimmt die Funktion **755** als Eingangs-Zustands-Informationen **756** solche wie beispielsweise Adressen, Prozess-Identifizierer, Adressen-Raum-Zahlen, Hardware-Kontext-Identifizierer oder Thread-Identifizieret der ausgewählten Befehle. Die Funktion **755** kann auch Zustands-Informationen, wie beispielsweise Pfad-Identifizierungs-Informationen, Opcode, Operanden, Latenzzeiten oder Ereignisse, erfahren durch die ausgewählten Befehle, verwenden. Die Ereignis-Informationen können einen Ausscheidungs-/Aussonderungs-/Ungültigkeits-Status, Cache-Treffer/Fehler, Verzweigungs-Fehlvorhersagen, Trap-Status, TLB-Treffer/Fehler, und Daten-Ressource-Abhängigkeits-Status, usw., sein.

[0135] Schritt **760** erzeugt einen Untersatz von Abtastungen basierend auf der Funktion **755**. Im Schritt **780** werden Statistiken **790** bestimmt. Diese Statistiken können Durchschnitte, Standardabweichungen, Histogramme (Verteilung) und Fehlerbegrenzungen (error bounds) der Eigenschaften der abgetasteten Befehle sein. Zum Beispiel durchschnittliche Raten, unter denen spezifizierte Ereignisse auftreten, durchschnittliche Latenzzeiten einer Befehl-Ausführung und von Speicher-Zugriffen. Durchschnitte der Ausführungsraten von Verarbeitungen, Threads oder Hardware-Zusammenhängen können auch bestimmt werden. Die Histogramme können die Verteilung einer Befehl-Ausführung, von Speicher-Zugrift-Raten oder Latenzzeiten darstellen.

[0136] Die Begrenzung in Bezug auf die Fehler kann durch den Umkehrwert einer Quadratwurzel der Zahl von Abtastungen für die bestimmte Eigenschaft, die abgetastet werden soll, angenähert werden.

Patentansprüche

1. Vorrichtung zum Abtasten von Befehlen in einer Prozessor-Pipeline (**200**) eines Systems, wobei die Pipeline eine Vielzahl von Verarbeitungsstufen aufweist, und die umfasst:
eine Einrichtung (**210**), die Befehle in eine erste Stufe der Pipeline abruft, wobei die Befehle willkürlich durch zusätzliche Felder identifiziert werden, die anzeigen, dass sie zum Abtasten ausgewählt worden sind, und die zusätzlichen Felder ein Abtast-Bit (**401**) an jedem Befehl in der Pipeline enthalten;
eine Einrichtung, die jeden der abgerufenen Befehle als einen ausgewählten Befehl identifiziert;
eine Einrichtung, die Statusinformationen des Systems abtastet, während sich ein bestimmter ausgewählter Befehl in einer beliebigen Stufe der Pipeline (**200**) befindet;
eine Einrichtung (**300**), die die Statusinformationen speichert; und
eine Einrichtung, die Software informiert, wenn der bestimmte ausgewählte Befehl die Pipeline (**200**) verlässt, so dass die Software jede der Statusinformationen lesen kann, wobei die Einrichtung, die abtastet, und die Einrichtung, die Software informiert, in Funktion aktiviert werden, indem das Abtast-Bit (**401**) in dem ausgewählten Befehl aktiviert wird.

2. Vorrichtung (**100**) nach Anspruch 1, wobei die ausgewählten Befehle gültige Befehle, die durch die Pipeline (**200**) vollständig bearbeitet werden, gültige Befehle, die vor dem Verlassen der Pipeline abgebrochen werden, und ungültige Befehle enthalten, die vor dem Verlassen der Pipeline teilweise bearbeitet werden.

3. Vorrichtung (**100**) nach Anspruch 1, wobei die ausgewählten Befehle nicht unterbrechbare Befehle enthalten können.

4. Vorrichtung (**100**) nach einem der Ansprüche 1 bis 3, wobei die Vielzahl von Stufen Abrufstufen (**210**), Abbildungsstufen (**220**), Ausgabestufen (**230**), Ausführungsstufen (**240**) und Ausscheidestufen (**250**) enthalten.

5. Vorrichtung (**100**) nach einem der vorangehenden Ansprüche, wobei die zusätzlichen Felder Befehlsnummern speichern, und die des Weiteren Vergleichsregister (**300**) enthält, die die zusätzlichen Felder speichern.

6. Vorrichtung (**100**) nach einem der vorangehenden Ansprüche, die des Weiteren enthält:
einen Abtast-Zähler (**510**);
eine Einrichtung (**520**), die den Abtast-Zähler (**510**) auf einen vorgegebenen Wert initialisiert;
eine Einrichtung, die die Identifizierungseinrichtung aktiviert, wenn der Abtast-Zähler (**510**) in Reaktion auf vorgegebene Ereignisse überläuft.

7. Vorrichtung (**100**) nach Anspruch 6, wobei die vorgegebenen Ereignisse gültige abgerufene Befehle sind.

8. Vorrichtung (**100**) nach Anspruch 6, wobei eine

Abrufrate durch einen Takt bestimmt wird und die vorgegebenen Ereignisse Unterteilungen von Taktzyklen sind, die einem einzelnen, potentiell abgerufenen Befehl entsprechen.

9. Vorrichtung (**100**) nach Anspruch 6, wobei die vorgegebenen Ereignisse der Eintritt von Befehlen in jede beliebige Stufe der Pipeline (**200**) sind.

10. Vorrichtung (**100**) nach Anspruch 6, wobei der vorgegebene Wert willkürlich aus einem Intervall von Zahlen gewählt wird, um die durchschnittliche Abtastfrequenz vorzugeben.

11. Vorrichtung (**100**) nach Anspruch 10, wobei die Größe des Intervalls von Zahlen geändert wird, um die durchschnittliche Abtastfrequenz dynamisch zu verändern.

12. Vorrichtung (**100**) nach Anspruch 10, wobei die willkürliche Zahl durch die Software bestimmt wird.

13. Vorrichtung (**100**) nach Anspruch 10, wobei die willkürliche Zahl durch Hardware erzeugt wird.

14. Vorrichtung (**100**) nach einem der vorangehenden Ansprüche, wobei die Statusinformationen Informationen enthalten, die den ausgewählten Befehl identifizieren.

15. Vorrichtung (**100**) nach Anspruch 14, wobei die Identifizierungsinformationen die Adressen der ausgewählten Befehle enthalten.

16. Vorrichtung (**100**) nach Anspruch 14, wobei die Identifizierungsinformationen Identifizierungen von Prozessen enthalten, die den ausgewählten Befehl ausführen.

17. Vorrichtung (**100**) nach Anspruch 14, wobei die Identifizierungsinformationen Adressraumzahlen enthalten.

18. Vorrichtung (**100**) nach Anspruch 14, wobei die Identifizierungsinformationen eine Hardwarekontext-Kennung enthalten.

19. Vorrichtung (**100**) nach einem der vorangehenden Ansprüche, wobei die Einrichtung, die Software informiert, eine Unterbrechung erzeugt, wenn der bestimmte ausgewählte Befehl die Pipeline verlässt.

20. Vorrichtung (**100**) nach einem der Ansprüche 1 bis 18, wobei die Einrichtung, die Software informiert, ein Flag setzt, das durch die Software abgefragt werden kann, um festzustellen, dass der bestimmte ausgewählte Befehl die Pipeline verlassen hat.

21. Vorrichtung (**100**) nach einem der vorangehenden Ansprüche, wobei eine Teilgruppe der ausgewählten Befehle abgetastet wird.

22. Vorrichtung (**100**) nach einem der vorangehenden Ansprüche, wobei die Statusinformationen eine Thread-Kennung enthalten.

23. Vorrichtung (**100**) nach einem der Ansprüche 1 bis 20, wobei die Statusinformationen einen Ausgeschieden-/Abgebrochen-Status der ausgewählten Befehle enthalten.

24. Vorrichtung (**100**) nach einem der Ansprüche 1 bis 20, wobei die Statusinfonnationen Ereignisse enthalten, die bei der Verarbeitung der ausgewählten Befehle erfasst werden.

25. Vorrichtung (**100**) nach einem der Ansprüche 1 bis 20, wobei die Statusinfonnationen Latenzen enthalten, die bei den ausgewählten Befehlen auftreten.

26. Computersystem (**100**), das eine Vorrichtung nach einem der Ansprüche 1 bis 25 enthält.

Es folgen 7 Blatt Zeichnungen

Anhängende Zeichnungen

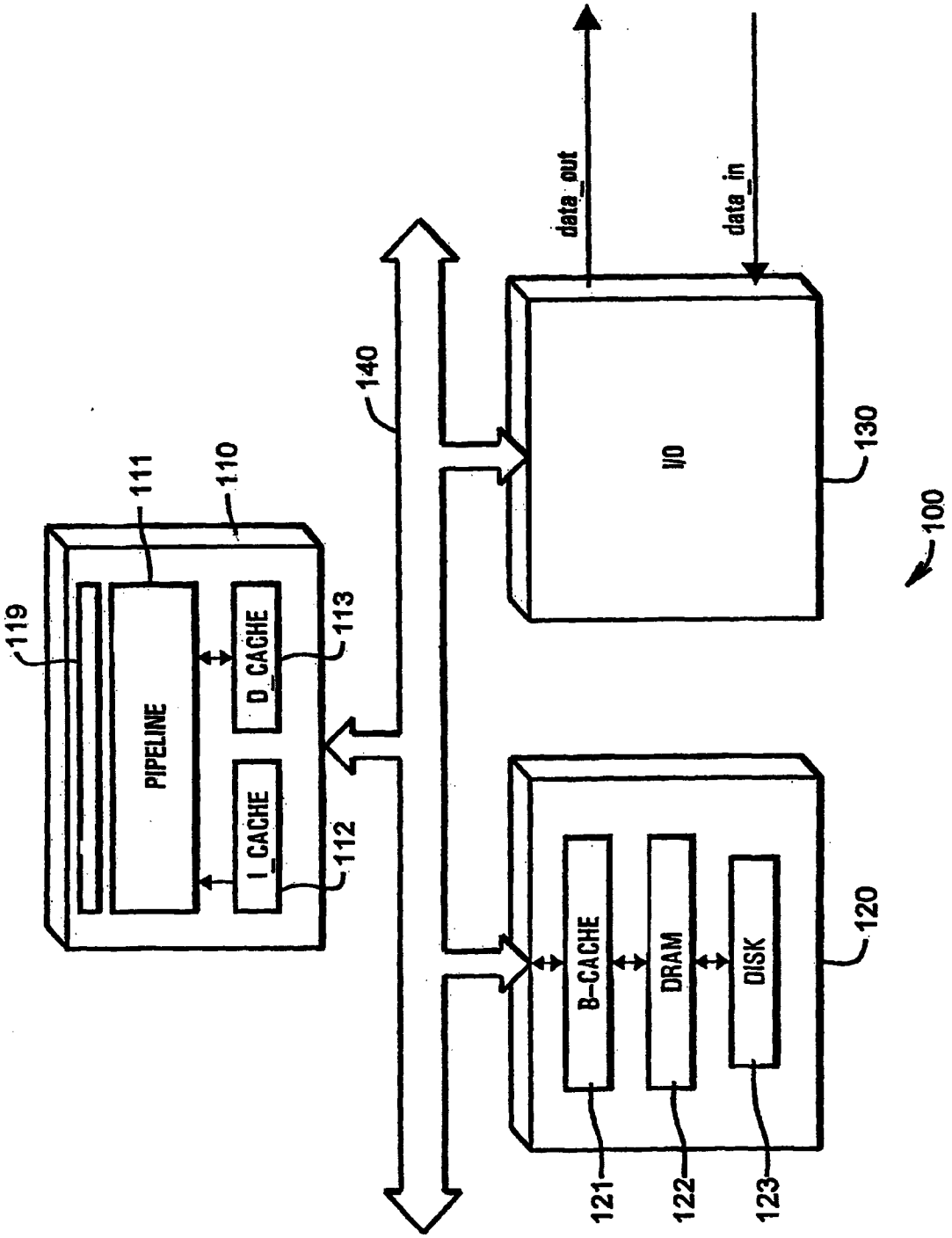

FIG. 1

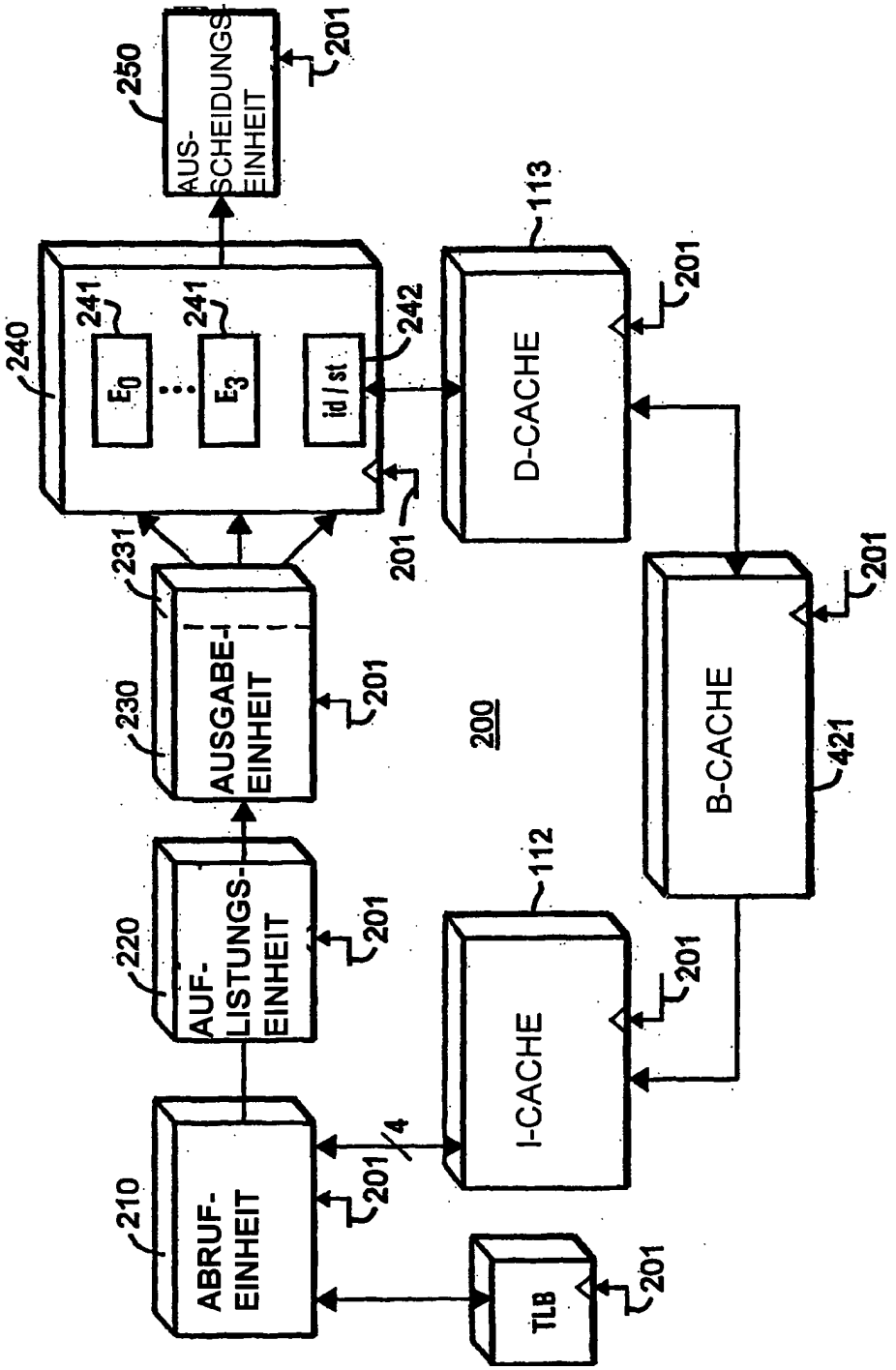
ABRUF-EINHEIT
210
AUF-LISTUNGS-EINHEIT
220
AUSGABE-EINHEIT
230
231
240
241
241
E_0
E_3
id / st
242
AUS-SCHEIDUNGS-EINHEIT
250
TLB
I-CACHE
112
D-CACHE
113
B-CACHE
421
200
201
4

FIG. 2a

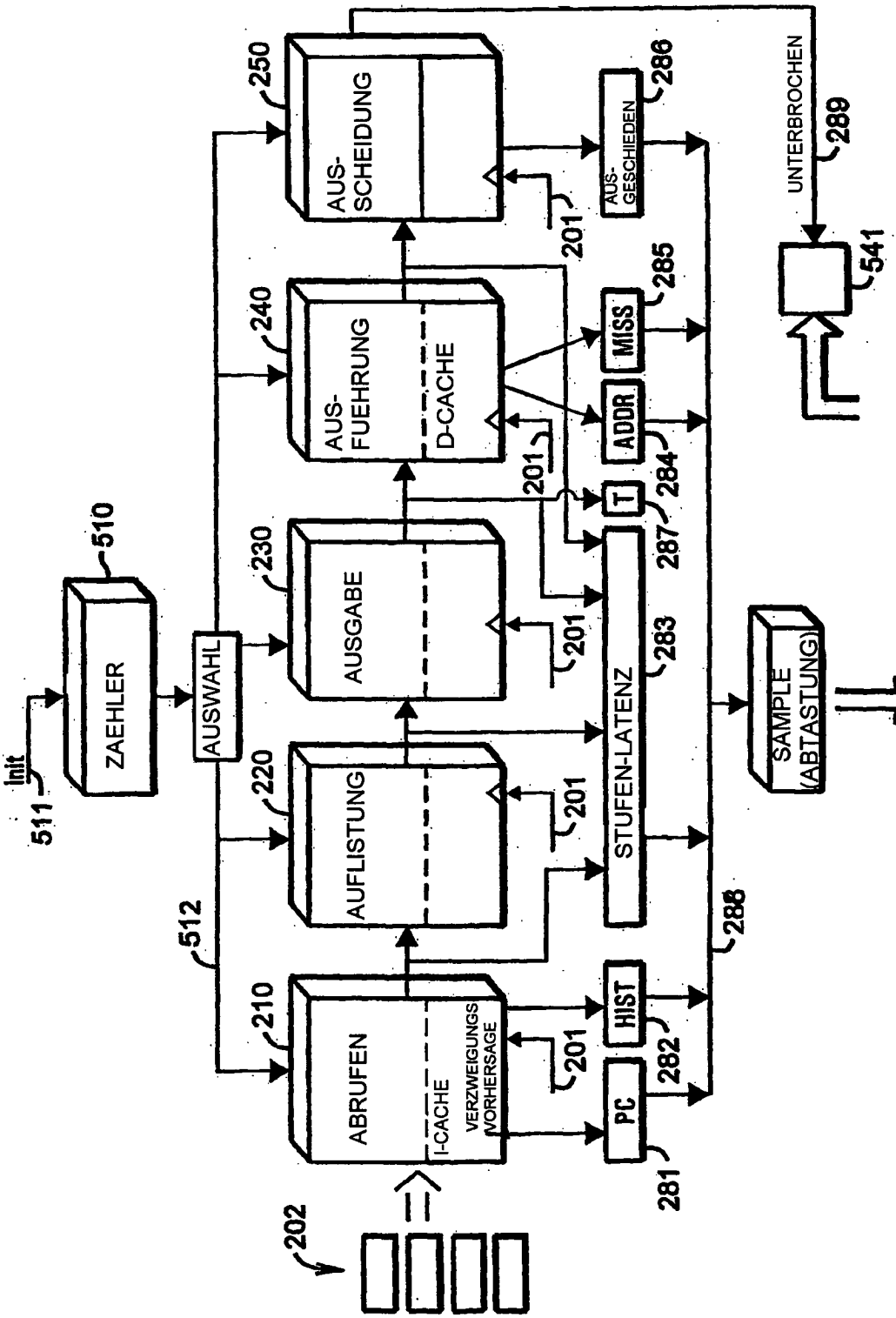
init
511
ZAEHLER
510
AUSWAHL
512
ABRUFEN
I-CACHE
VERZWEIGUNGS VORHERSAGE
210
AUFLISTUNG
220
AUSGABE
230
AUS-FUEHRUNG
D-CACHE
240
AUS-SCHEIDUNG
250
202
201
HIST
PC
281
282
STUFEN-LATENZ
283
288
T
284
287
ADDR
MISS
285
AUS-GESCHIEDEN
286
UNTERBROCHEN
289
541
SAMPLE (ABTASTUNG)

FIG. 2b

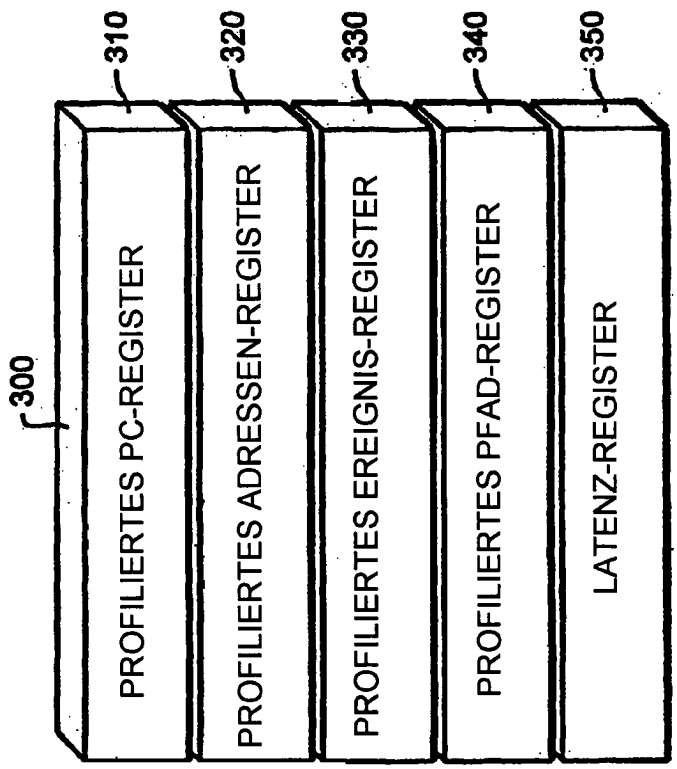
300
310
PROFILIERTES PC-REGISTER
320
PROFILIERTES ADRESSEN-REGISTER
330
PROFILIERTES EREIGNIS-REGISTER
340
PROFILIERTES PFAD-REGISTER
350
LATENZ-REGISTER

FIG. 3

400
op 1
op 2
op 3
PC
V
opcode
1-cache miss
TLB MISS
S
ID
411
412
413
420
431
430
440
450
401
402

FIG. 4

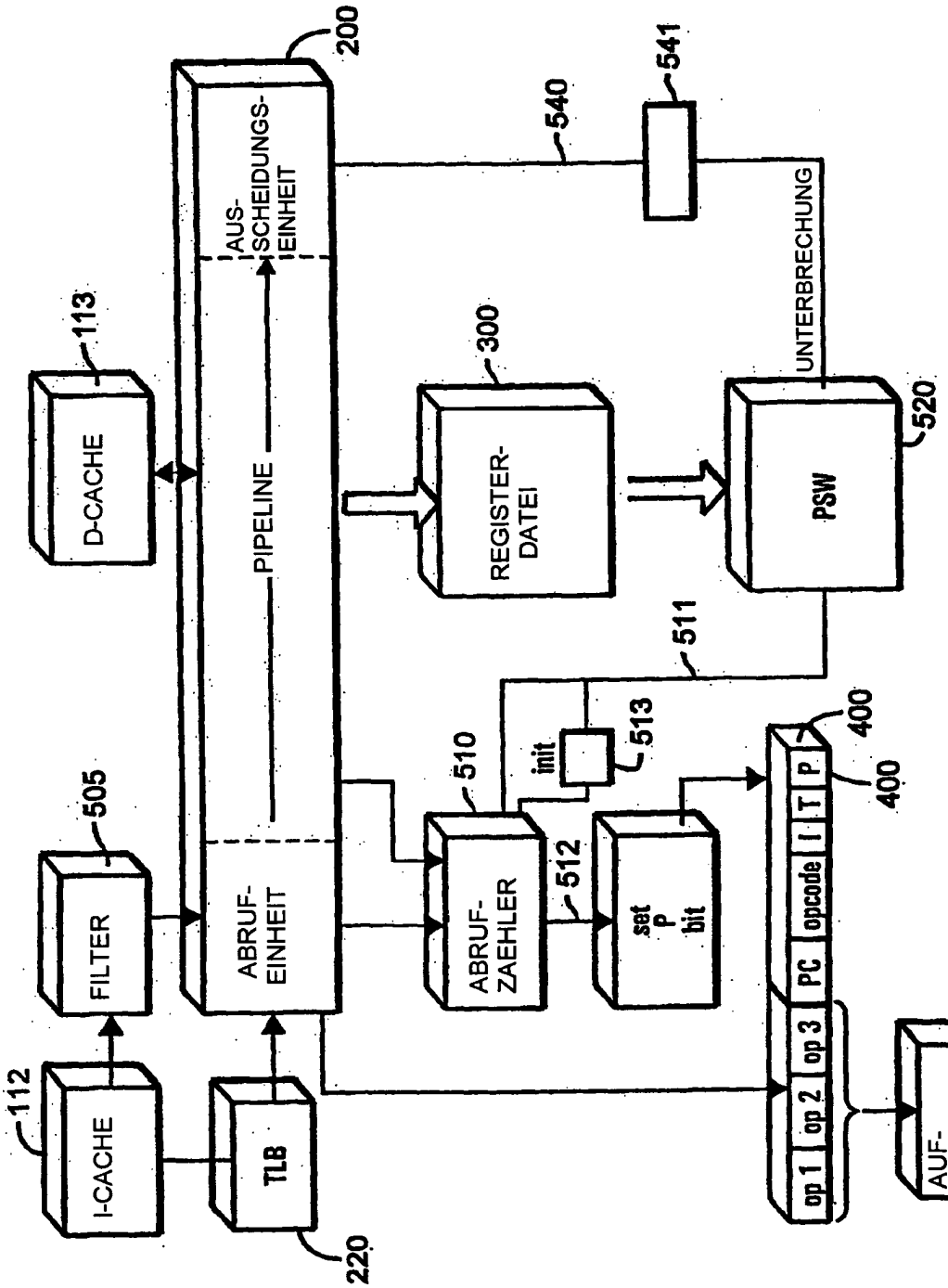
200
AUS-SCHEIDUNGS-EINHEIT
PIPELINE
ABRUF-EINHEIT
113
D-CACHE
505
FILTER
112
I-CACHE
TLB
220
300
REGISTER-DATEI
540
541
UNTERBRECHUNG
520
PSW
511
510
ABRUF-ZAEHLER
512
init
513
set P bit
400
op 1
op 2
op 3
PC
opcode
I
T
P
AUF-LISTUNGS-EINHEIT
220

FIG. 5

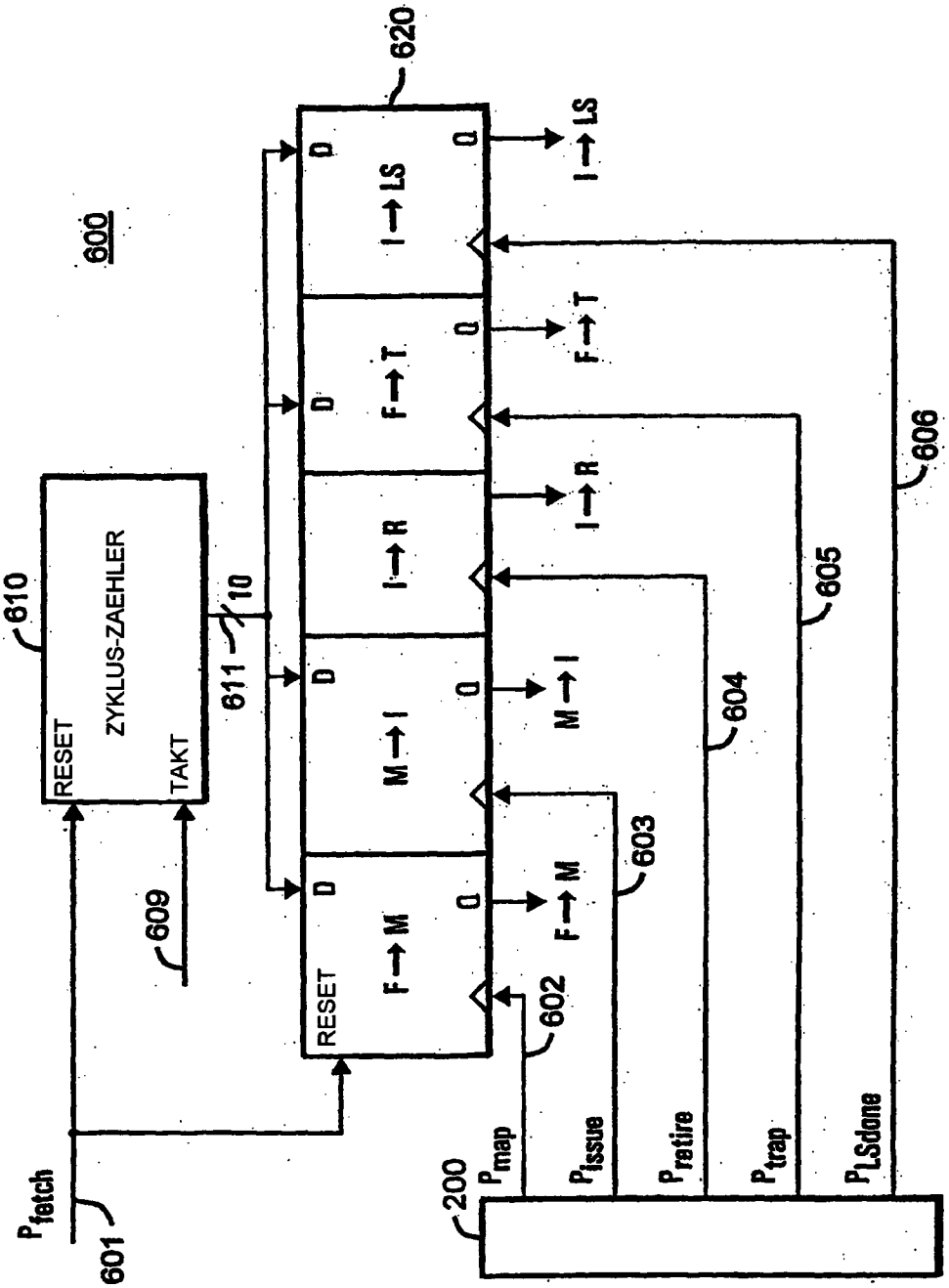
600
620
610
ZYKLUS-ZAEHLER
RESET
TAKT
609
611
10
D
Q
I → LS
F → T
I → R
M → I
F → M
RESET
I → LS
F → T
I → R
M → I
F → M
602
603
604
605
606
200
P_{fetch}
P_{map}
P_{issue}
P_{retire}
P_{trap}
P_{LSdone}
601

FIG. 6

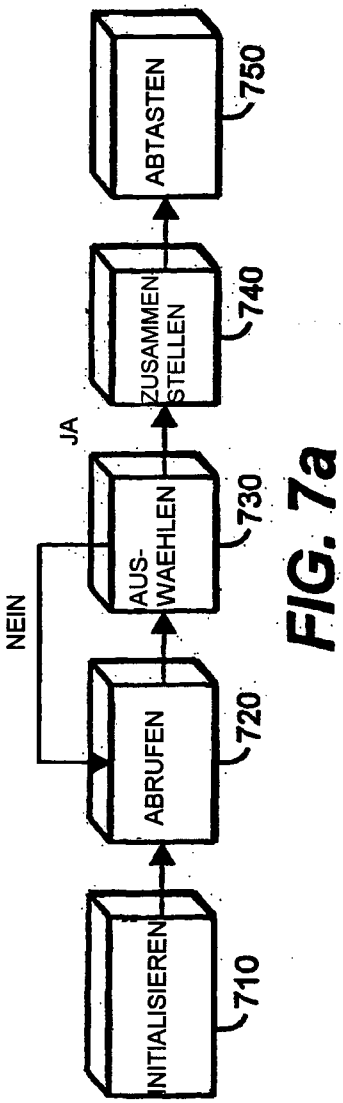
INITIALISIEREN
710
ABRUFEN
720
NEIN
AUS-WAEHLEN
730
JA
ZUSAMMEN STELLEN
740
ABTASTEN
750

FIG. 7a

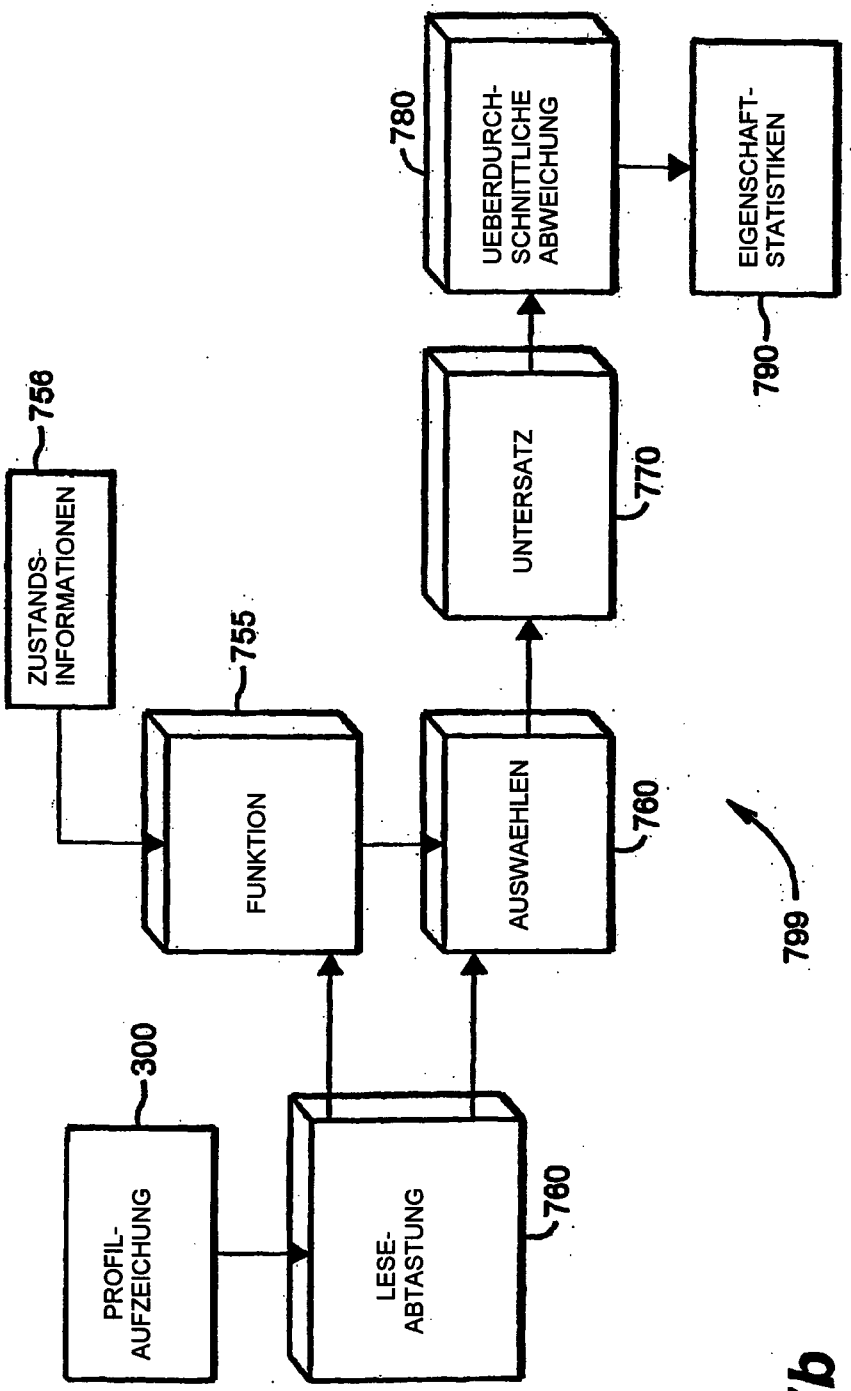
ZUSTANDS-INFORMATIONEN
756
FUNKTION
755
PROFIL-AUFZEICHNUNG
300
LESE-ABTASTUNG
760
AUSWAEHLEN
760
UNTERSATZ
770
UEBERDURCH-SCHNITTLICHE ABWEICHUNG
780
EIGENSCHAFT-STATISTIKEN
790
799

FIG. 7b