(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 11 2004 001 854 T5** 2006.08.03

(12) # Veröffentlichung

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2005/041024**
in deutscher Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: **11 2004 001 854.4**
(86) PCT-Aktenzeichen: **PCT/US2004/020721**
(86) PCT-Anmeldetag: **28.06.2004**
(87) PCT-Veröffentlichungstag: **06.05.2005**
(43) Veröffentlichungstag der PCT Anmeldung
in deutscher Übersetzung: **03.08.2006**

(51) Int Cl.[8]: ***G06F 9/38*** (2006.01)

(30) Unionspriorität:
10/676,437 **01.10.2003** **US**

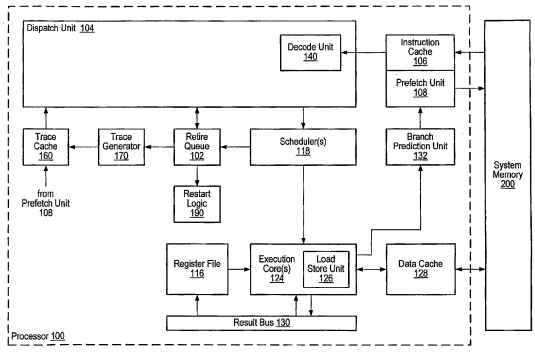
(71) Anmelder:
Advanced Micro Devices, Inc., Sunnyvale, Calif., US

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

(72) Erfinder:
Alsup, Mitchell, Austin, Tex., US; Smaus, Gregory William, Austin, Tex., US; Pickett, James K., Austin, Tex., US; McCinn, Brian D., Buda, Tex., US

(54) Bezeichnung: **System und Verfahren zur Handhabung von Sonderinstruktionen in einem Prozessor auf Grundlage eines Cache-Speichers mit Ablaufverfolgung**

(57) Hauptanspruch: System mit:
einem Instruktionscache-Speicher (106);
einem Ablaufverfolgungscache-Speicher (160) mit mehreren Ablaufverfolgungscache-Einträgen (162);
einem Ablaufverfolgungsgenerator (170), der mit dem Instruktionscache-Speicher (106) und dem Ablaufverfolgungscache-Speicher (160) verbunden ist;
wobei der Ablaufverfolgungsgenerator (170) ausgebildet ist, eine Gruppe aus Instruktionen, die von dem Instruktionscache-Speicher (106) ausgegeben wird zur Speicherung in einer der mehreren Ablaufverfolgungscache-Einträgen (162) zu empfangen, wobei der Ablaufverfolgungsgenerator (170) ausgebildet ist, eine Sonderinstruktion innerhalb der Gruppe aus Instruktionen zu erkennen und ein Speichern der Sonderinstruktion in einem gleichen der mehreren Ablaufverfolgungscache-Einträge (162) als eine Nicht-Sonderinstruktion zu verhindern.



Beschreibung

Technisches Gebiet

[0001] Diese Erfindung betrifft das Gebiet der Prozessoren und betrifft insbesondere Prozessoren mit Cache-Speicher- bzw. Pufferspeicher mit Ablaufverfolgung.

Hintergrund der Erfindung

[0002] In einem Prozessor abgearbeitete Instruktionen werden als eine Sequenz von Einsen und Nullen codiert. Für gewisse Prozessorarchitekturen werden Instruktionen mit einer festgelegten Länge, etwa einer gewissen Anzahl an Bytes codiert. Bei anderen Architekturen, etwa der x86-Architektur, kann die Länge der Instruktionen variieren. Die x86-Prozessorarchitektur spezifiziert einen Instruktionssatz mit variabler Länge (d. h. einen Instruktionssatz, in welchem die diversen Instruktionen jeweils durch eine unterschiedliche Anzahl an Bytes spezifiziert sind). Beispielsweise verwenden der 80386 und spätere Versionen des x86-Prozessors zwischen 1 und 15 Bytes, um eine spezielle Instruktion zu kennzeichnen. Instruktionen besitzen einen Op-Code bzw. Operationscodierung, die 1 bis 2 Bytes sein kann, und zusätzliche Bytes können hinzugefügt werden, um Adressenmodi, Operanden und zusätzliche Details hinsichtlich der auszuführenden Instruktion zu spezifizieren.

[0003] In einigen Prozessorarchitekturen wird jede Instruktion in eine oder mehrere einfachere Operationen vor der Ausführung decodiert. Das Decodieren einer Instruktion kann auch das Zugreifen auf eine Registerumbenennungszuordnung beinhalten, um das physikalische Register zu bestimmen, die jedem logischen Register in der Instruktion zugeordnet ist und/oder um ein physikalisches Register zu reservieren, um das Ergebnis der Instruktion zu speichern.

[0004] Instruktionen werden in dem Decodierungsbereich eines Prozessors teilweise auf der Grundlage einer Verzweigungsvorhersage eingeladen, die innerhalb des Prozessors gemacht wird. Im Allgemeinen bestimmen die Bandbreite der Instruktionsabholungs- und Decodierungsbereiche eines Prozessors, ob die Ausführungskerne während jedes Ausführungszyklus vollständig ausgenutzt werden. Folglich ist es wünschenswert in der Lage zu sein, ausreichend Bandbreite in den Instruktionsabhol- und Decodierungsbereichen des Prozessors bereitzustellen, um den Ausführungskern möglichst stark auszulasten.

Überblick über die Erfindung

[0005] Diverse Ausführungsformen und Verfahren und Systeme zum Handhaben von Ausnahme- bzw. Sonderinstruktionen innerhalb eines Prozessors auf Grundlage eines Cache-Speichers mit Ablaufverfolgung werden hiermit offenbart. In einigen Ausführungsformen umfasst ein System einen Instruktionscache-Speicher bzw. Pufferspeicher, einen Ablaufverfolgungscache-Speicher mit mehreren Ablauflaufverfolgungscache-Einträgen und einen Ablaufsverfolgungsgenerator, der mit dem Instruktionscache-Speicher und dem Ablaufverfolgungscache-Speicher verbunden ist. Der Ablaufverfolgungsgenerator ist ausgebildet, eine Gruppe aus Instruktionen zu empfangen, die von dem Instruktionscache-Speicher zur Speicherung in einem der mehreren Ablaufverfolgungscache-Einträge ausgegeben werden. Der Ablaufverfolgungsgenerator ist ausgebildet, eine Sonderinstruktion innerhalb der Gruppe aus Instruktionen zu erkennen und zu verhindern, dass die Sonderinstruktion in dem gleichen Eintrag der mehreren Ablaufverfolgungscache-Einträge wie eine Nichtsonderinstruktion gespeichert wird. In einer Ausführungsform ist der Ablaufverfolgungsgenerator ausgebildet, die Sonderinstruktion nicht in dem Ablaufverfolgungscache-Speicher abzulegen. In einer weiteren Ausführungsform ist der Ablaufverfolgungsgenerator ausgebildet, die Sonderinstruktion in einem anderen Ablaufverfolgungscache-Eintrag zu speichern im Vergleich zu einer oder mehreren anderen Instruktionen, und um nicht andere Instruktionen mit der Sonderinstruktion in dem anderen Ablaufverfolgungscache-Eintrag zu speichern. Der Ablaufverfolgungsgenerator ist ausgebildet, Instruktionen in dem Ablaufverfolgungscache-Speicher in einer zumindest teilweisen decodierten Form zu speichern.

[0006] Das System kann ferner eine Abschlusswartereihe bzw. Fertigstellungswartereihe aufweisen, die angeschlossen ist, um ausgeführte Instruktionen abzuschließen. Die Abschlusswartereihe ist ausgebildet, die Beendigung einer von dem Ablaufverfolgungscache-Speicher abgeholten Instruktion zu verhindern, bis alle Instruktionen, die in dem gleichen Ablaufverfolgungscache-Eintrag wie diese Instruktion enthalten sind, bereit sind für die Rücknahme bzw. deren Abschluss. Alternativ kann die Abschlusswartereihe ausgebildet sein, eine Beendigung einer beliebigen von dem Ablaufverfolgungscache-Speicher abgeholten Instruktion zu verhindern, bis alle in einer gleichen „Vitalitäts-"gruppe innerhalb eines gleichen Ablaufverfolgungscache-Eintrags wie diese Instruktion enthaltenen Instruktionen bereit sind für die Vervollständigung.

[0007] Das System umfasst eine Neustartlogik, die ausgebildet ist, die Anzahl der Zyklen zu überwachen, die seit der jüngsten Beendigung einer Instruktion durch die Abschlusswartereihe verstrichen sind. Wenn die Anzahl der Zyklen einen Schwellwert übersteigt, ist die Neustartlogik ausgebildet, eine arbeitende Pipeline, die Instruktionen ausführt, die von dem Ablaufverfolgungscache-Speicher abgeholt

wurden, zurückzusetzen und die Ausführung aus dem Instruktionscache-Speicher neu zu starten.

[0008] Eine Ausgabeeinheit, die in dem System enthalten ist, ist ausgebildet, Instruktionen, die von dem Ablaufverfolgungscache-Speicher empfangen werden, auszugeben. Die Ausgabeeinheit ist ausgebildet, eine nicht erneut ausführbare Instruktion innerhalb einer Gruppe von Instruktionen, die von dem Ablaufverfolgungscache-Speicher empfangen wurden zu erkennen und eine Kennzeichnung der nicht erneut ausführbaren Instruktion an die Neustartlogik auszugeben. In Reaktion auf die Kennzeichnung ist die Neustartlogik ausgebildet, die bearbeitende Pipeline zurücksetzen und die Ausführung von dem Instruktionscache-Speicher vor der Ausführung der nicht erneut ausführbaren Instruktion erneut zu starten.

[0009] Die Ausgabeeinheit ist ausgebildet, die Sonderinstruktion innerhalb einer Gruppe aus Instruktionen, die von dem Instruktionscache-Speicher empfangen wurden, zu erkennen und eine Kennzeichnung für die Sonderinstruktion an den Ablaufverfolgungsgenerator auszugeben, wodurch der Ablaufverfolgungsgenerator veranlasst wird, die Sonderinstruktion zu erkennen. Der Ablaufverfolgungsgenerator ist ausgebildet, die Sonderinstruktionen in Reaktion auf eine oder mehrere Eigenschaften (beispielsweise Operationscodierungen, Adressiermodi, etc.) der Sonderinstruktionen zu erkennen.

[0010] In einigen Ausführungsformen umfasst ein Verfahren: Empfangen einer Gruppe aus Instruktionen für die Speicherung in einem Ablaufverfolgungscache-Eintrag innerhalb eines Ablaufverfolgungscache-Speichers. Speichern einer oder mehrer Instruktionen, die in der Gruppe aus Instruktionen enthalten sind, in dem Ablaufverfolgungscache-Eintrag; Erkennen einer Sonderinstruktion innerhalb der Gruppe aus Instruktionen; und in Reaktion auf das Erkennen, nicht speichern der Sonderinstruktion innerhalb des Ablaufverfolgungscache-Eintrags aus der einen oder mehreren Instruktionen.

Kurze Beschreibung der Zeichnungen

[0011] Ein besseres Verständnis der vorliegenden Erfindung kann erreicht werden, wenn die folgende detaillierte Beschreibung in Verbindung mit den folgenden Zeichnungen studiert wird, in denen:

[0012] Fig. 1 eine Ausführungsform eines Prozessors zeigt;

[0013] **Fig.** 2 eine Blockansicht eines Ablaufverfolgungscache-Speichers gemäß einer Ausführungsform zeigt;

[0014] Fig. 3a einen beispielhaften Ablaufverfolgungscache-Eintrag gemäß einer Ausführungsform zeigt;

[0015] Fig. 3b eine Tabelle von „Vitalitäts"- Codierungen, die verwendet werden kann, um die Vitalitätsgruppe zu erkennen, die jede Operation in einer Ablaufverfolgung gehört gemäß einer Ausführungsform;

[0016] Fig. 4 ein Flussdiagramm einer Ausführungsform eines Verfahrens zum Erzeugen von ablaufverfolgenden zur Speicherung in einem Ablaufverfolgungscache-Speicher gemäß einer Ausführungsform ist;

[0017] Fig. 5 ein Flussdiagramm einer Ausführungsform eines Verfahrens zum Ausführen einer Ablaufverfolgung ist, die eine Sonderinstruktion gemäß einer Ausführungsform aufweist;

[0018] Fig. 6 eine Blockansicht eines Computersystems gemäß einer Ausführungsform zeigt;

[0019] Fig. 7 eine Blockansicht eines Computersystems gemäß einer weiteren Ausführungsform zeigt.

[0020] Obwohl die Erfindung diversen Modifizierungen und Alternativen unterliegen kann, sind dennoch spezielle Ausführungsformen beispielhaft in den Zeichnungen dargestellt und werden hierin detailliert beschrieben. Es sollte jedoch selbstverständlich sein, dass die Zeichnungen und die detaillierte Beschreibung nicht beabsichtigen, die Erfindung auf die spezielle offenbarte Form einzuschränken, sondern es ist vielmehr beabsichtigt alle Modifizierungen, Äquivalente und Alternativen abzudecken, die innerhalb des Grundgedankens und Schutzbereichs der vorliegenden Erfindung liegen, wie sie durch die angefügten Patentansprüche definiert ist. Zu beachten ist, dass die Überschriften nur der Gliederung dienen und nicht dazu gedacht sind, die Beschreibung oder die Ansprüche einzuschränken oder zu deuten. Des weiteren ist anzumerken, dass das Wort „kann" durchwegs in dieser Anmeldung in einem zulässigen Sinne (d. h. besitzt die Möglichkeit zu, ist in der Lage zu) verwendet ist, und nicht in einem definitiven Sinne (d. h. muss). Der Begriff „enthalten" und Ableitungen davon bedeuten „einschließlich aber nicht beschränkt darauf". Der Begriff „verbunden" bedeutet „direkt oder indirekt verbunden" und der Begriff „gekoppelt" bedeutet „direkt oder indirekt gekoppelt".

Art bzw. Arten zum Ausführen der Erfindung

[0021] Fig. 1 ist eine Blockansicht logischer Komponenten, die in einer Ausführungsform eines Prozessors **100** enthalten sind. Der Prozessor **100** ist ausgebildet, Instruktionen auszuführen, die in einem Systemspeicher **200** abgelegt sind. Viele dieser Instruktionen operieren auf Daten, die in dem System-

speicher **200** gespeichert sind, der physikalisch über ein Computersystem verteilt sein kann und auf den von einem oder mehreren Prozessoren **100** zugegriffen werden kann. In einigen Ausführungsformen ist der Prozessor **100** so gestaltet, dass dieser mit der x86-Architektur kompatibel ist.

[0022] Der Prozessor **100** kann einen Instruktionscache-Speicher bzw. Pufferspeicher **106** und einen Datencache-Speicher bzw. Datenpufferspeicher **128** aufweisen. Der Prozessor **100** kann ferner eine Vorabholeinheit **108** aufweisen, die mit dem Systemspeicher **200** verbunden ist. Die Vorabholeinheit **108** holt vorläufig eine Instruktionscodierung von dem Systemspeicher **200** zur Speicherung in dem Instruktionscache-Speichers **106** ab. In einer Ausführungsform ist die Vorabholeinheit **108** ausgebildet, um Codierungen aus dem Systemspeicher **200** blockweise in den Instruktionscache-Speicher **106** zu speichern. In der Vorabholeinheit **108** können diverse spezifische Codierungs-Vorabholungs-Verfahren und Algorithmen eingesetzt werden. Die Vorabholeinheit **108** kann ferner Instruktionen aus dem Instruktionscache-Speicher **106** und Ablaufverfolgungsdaten aus dem Ablaufverfolgungscache-Speicher **160** in eine Ausgabeeinheit **104** holen. Instruktionen können aus dem Instruktionscache-Speicher **106** geholt werden, in Reaktion darauf, dass eine vorgegebene Instruktionsadresse nicht den Ablaufverfolgungscache-Speicher **160** betrifft. In ähnlicher Weise können Instruktionen von dem Systemspeicher **200** abgeholt werden in Reaktion darauf, dass eine vorgegebene Adresse nicht den Instruktionscache-Speicher **106** betrifft.

[0023] Eine Ausgabeeinheit **104** ist ausgebildet, Instruktionen von dem Instruktionscache-Speicher **106** zu empfangen und decodierte und/oder teilweise decodierte Instruktionen aus dem Ablaufverfolgungscache-Speicher **160** zu empfangen. Die Ausgabeeinheit **104** kann eine Decodierungseinheit **140** aufweisen, um von dem Instruktionscache-Speicher **106** empfangene Instruktionen zu decodieren. Die Ausgabeeinheit **104** kann ferner eine Mikrocodierungseinheit enthalten, die verwendet wird, wenn mikrocodierte Instruktionen zu handhaben sind.

[0024] Die Ausgabeeinheit **104** ist ausgebildet, decodierte Instruktionen an eine oder mehrere Ablauforganisationseinheiten **118** auszugeben. Eine oder mehrere Ablauforganisationseinheiten **118** können angeschlossen sein, um ausgegebene Instruktionen von der Ausgabeeinheit **104** zu empfangen und Instruktionen zu einem oder mehreren Ausführungskernen **124** auszugeben. Der bzw. die Ausführungskerne **124** enthalten eine Lade/Speichereinheit **126**, die ausgebildet ist, Zugriffe auf den Datencache-Speicher **128** auszuführen. Ergebnisse, die von dem bzw. den Ausführungskernen **124** erzeugt werden, können auf einem Ergebnisbus **130** ausgegeben werden. Diese Ergebnisse können als Operandenwerte für nachfolgend ausgegebene Instruktionen verwendet werden und/oder in einer Registerdatei **116** gespeichert werden. Eine Rücknahme- bzw. Vollendungswartereihe **102** ist mit dem bzw. den Ablauforganisationseinheiten **118** und der Ausgabeeinheit **104** verbunden. Die Abschlusswartereihe ist ausgebildet, zu bestimmen, wann jede ausgegebene Instruktion vollendet bzw. abgeschlossen werden kann.

[0025] Der Instruktionscache-Speicher **106** kann zeitweilig Instruktionen speichern, bevor diese von der Ausgabeeinheit **104** empfangen werden. Die Instruktionscodierung kann an den Instruktionscache-Speicher **106** geliefert werden, indem die Codierung vorläufig aus dem Systemspeicher **200** mittels der Vorabholeinheit **108** ausgelesen wird. Der Instruktionscache-Speicher **106** kann in diversen Konfigurationen eingerichtet sein (beispielsweise teilassoziativ, vollständig assoziativ oder direkt zugeordnet).

[0026] Im hierin verwendeten Sinne bezeichnet der Begriff Instruktion bzw. Anweisung allgemein eine nicht decodierte, eine teilweise decodierte oder eine vollständig decodierte Instruktion. Teilweise und vollständig decodierte Instruktionen können auch als Operationen bezeichnet werden. Beispielsweise kann eine nicht decodierte Instruktion als von der Decodierungseinheit **140** in eine oder mehrere Komponentenoperationen decodiert beschrieben werden, die direkt von dem bzw. den Ausführungskernen **124** ausführbar sind. Die Ausgabeeinheit **104** kann Signale ausgeben, die Bit-codierte Operationen enthalten, die von dem Ausführungskern **124** ausführbar sind, sowie Operandenadressinformationen, unmittelbare Daten und/oder versetzte Daten. Beim Empfang einer Instruktion, die das Aktualisieren eines Register beinhaltet, kann die Ausgabeeinheit **104** einen Registerplatz innerhalb der Registerdatei **116** reservieren, um spekulative Registerzustände zu speichern (in einer alternativen Ausführungsform kann ein Umordnungspuffer verwendet werden, um einen oder mehrere spekulative Registerzustände für jedes Register zu speichern). Eine Registerzuordnung kann die logischen Registernamen von Quellen- und Zieloperanden den physikalischen Registernamen zuordnen, um die Registerumbenennung zu ermöglichen. Eine derartige Registerzuordnung kann verfolgen, welche Register innerhalb der Registerdatei **116** aktuell reserviert und nicht reserviert sind.

[0027] Wenn Instruktionen von der Ausgabeeinheit **104** gehandhabt werden, wenn ein erforderlicher Operand eine Registerstelle ist, kann Registeradresseninformation zu einer Registerzuordnung oder einem Umordnungspuffer zugeleitet werden. Beispielsweise gibt es in der x86-Architektur acht 32-Bit-Logikregister (beispielsweise, EAX, EBX, ECX, EDX, EBP, ESI, EDI und ESP). Die physikalische Registerdatei **116** (oder ein Umordnungspuffer) enthält Spei-

cherplatz für Ergebnisse, die den Inhalt dieser logischen Register ändern, wodurch eine Ausführung in ungeordneter Reihenfolge möglich ist. Ein physikalisches Register in der Registerdatei **116** kann reserviert werden, um das Ergebnis jeder Instruktion zu speichern, die den Inhalt eines der logischen Register modifiziert. Daher kann zu diversen Punkten während des Ausführens eines speziellen Programms die Registerdatei **116** (oder in alternativen Ausführungsformen ein Umordnungspuffer) ein oder mehrere Register aufweisen, die den spekulativ ausgeführten Inhalt eines gegebenen lokalen Registers enthalten.

[0028] Eine Registerzuordnung kann ein physikalisches Register einem speziellen logischen Register zuordnen, das als ein Zieloperand für eine Instruktion spezifiziert ist. Die Registerdatei **116** kann ein oder mehrere zuvor zugeordnete physikalische Register aufweisen, die einem logischen Register zugeordnet sind, das als ein Quellenoperand einer gegebenen Instruktion spezifiziert ist. Die Registerzuordnung kann eine Markierung für das physikalische Register bereitstellen, das zuletzt diesem logischen Register zugeordnet war. Diese Markierung kann verwendet werden, um auf den Datenwert des Operanden in der Registerdatei **116** zuzugreifen, oder um den Datenwert über eine Ergebnisweiterleitung auf dem Ergebnisbus **130** zu empfangen. Wenn der Operand einer Speicherstelle entspricht, kann der Operandenwert auf dem Ergebnisbus (für eine Ergebnisweiterleitung und/oder Speicherung in der Registerdatei **116**) über die Lade/Speichereinheit **126** bereitgestellt werden. Operandendatenwerte können dem bzw. den Ausführungskernen **124** bereitgestellt werden, wenn die Instruktion von einer der Ablauforganisationseinheiten **118** ausgegeben wird. Zu beachten ist, dass in alternativen Ausführungsformen Operandenwerte einer entsprechenden Ablauforganisationseinheit **118** zugeführt werden können, wenn eine Instruktion ausgegeben wird (anstatt dass diese einem entsprechenden Ausführungskern **124** zugeführt wird, wenn die Instruktion ausgegeben wird).

[0029] Der Prozessor **100** aus Fig. 1 unterstützt eine Ausführung in ungeordneter Reihenfolge. Eine Abschlusswartereihe **102** (oder alternativ ein Umordnungspuffer) können den Ablauf der ursprünglichen Programmsequenz für Registerlese- und Schreiboperation verfolgen, wodurch eine spekulative Instruktionsausführung und eine Verzweigungsfehlinterpretationswiederherstellung möglich ist und eine genaue Sonderbearbeitung ermöglicht werden. In vielen Ausführungsformen kann die Abschlusswartereihe **102** in ähnlicher Weise wie ein Umordnungspuffer arbeiten. Anders als ein typischer Umordnungspuffer stellt die Abschlusswartereihe **102** jedoch keinen Speicherplatz für Datenwerte bereit. In alternativen Ausführungsformen kann die Abschlusswartereihe **102** mehr wie ein Umordnungspuffer fungieren und auch die Registerumbenennung unterstützen, indem Datenwertspeicherplatz für spekulative Registerzustände bereitgestellt wird. In einigen Ausführungsformen kann die Abschlusswartereihe **102** in einer „zuerst hinein, zuerst heraus" Konfiguration eingerichtet sein, in der Operationen zur „Unterseite" des Puffers wandern, wenn sie validisiert werden, so dass Platz geschaffen wird für neue Einträge an der „Oberseite" der Wartereihe. Wenn Instruktionen abgeschlossen sind, kann die Abschlusswartereihe **102** Register in der Registerdatei **116** freigeben, die nicht mehr benötigt werden, um spekulative Registerzustände zu speichern und kann Signale an eine Registerzuordnung senden, die angeben, welche Register aktuell frei sind. Durch Bewahren spekulativer Registerzustände innerhalb der Registerdatei **116** (oder in alternativen Ausführungsformen innerhalb eines Umordnungspuffers) bis die Instruktionen, die diese Zustände erzeugten, gültig sind, können die Ergebnisse spekulativ ausgeführter Instruktionen entsprechend einem falsch vorhergesagten Weg in der Registerdatei **116** als unzulässig erklärt werden, wenn eine Verzweigungsvorhersage nicht korrekt ist.

[0030] Die Bit-codierten Operationen und die unmittelbaren Daten, die an den Ausgängen der Ausgabe **104** bereitgestellt werden, können zu einer oder mehreren Ablauforganisationseinheiten **118** geleitet werden. Zu beachten ist, dass im hierin verwendeten Sinne eine Ablauforganisationseinheit eine Einrichtung ist, die erkennt, wann Instruktionen für die Ausführung bereit sind und bereite Instruktionen zu einem oder mehreren Ausführungseinheiten ausgibt. Beispielsweise ist eine Reservierungsstation eine Art einer Ablauforganisationseinheit. Jede Ablauforganisationseinheit **118** ist in der Lage, Instruktionsinformationen (beispielsweise Bit-codierte Ausführungsbits sowie Operandenwerte, Operandenmarken, und/oder unmittelbare Daten) für einige anhängige Instruktionen, die auf eine Ausgabe an einen Ausführungskern **124** warten, zu halten. In einigen Ausführungsformen muss jede Ablauforganisationseinheit **118** nicht notwendigerweise Speicherplatz für einen Operandenwert bereitstellen. Stattdessen kann jede Ablauforganisationseinheit ausgegebene Instruktionen und in der Registerdatei **116** verfügbare Ergebnisse überwachen, um zu bestimmen, wann Operandenwerte verfügbar sind, so dass sie von einem oder mehreren Ausführungskernen **124** gelesen werden (von der Registerdatei **116** oder dem Ergebnisbus **130**). In einigen Ausführungsformen kann jede Ablauforganisationseinheit **118** mit einer zugeordneten Funktionseinheit (beispielsweise Ganzzahleinheit, Gleitkommaeinheit, Lade/Speichereinheit, etc.) und/oder einem Ausführungskern **124** verknüpft sein. In anderen Ausführungsformen kann eine einzelne Ablauforganisationseinheit **118** Instruktionen an mehr als einen Empfänger ausgeben.

[0031] Die Ablauforganisationseinheiten **118** kön-

nen vorgesehen sein, um zwischenzeitlich Operationsinformationen zu speichern, die von dem bzw. den Ausführungskernen **124** auszuführen sind. Instruktionen werden an den bzw. die Ausführungskerne **124** zur Ausführung in Reaktion darauf ausgegeben, dass die Werte von erforderlichen Operanden zur Zeit der Ausführung verfügbar sind. Folglich kann die Reihenfolge, in der die Instruktionen ausgeführt werden, unter Umständen nicht die gleich Reihenfolge sein, die der ursprünglichen Programminstruktionssequenz entspricht.

[0032] In einer Ausführungsform kann jeder der Ausführungskerne **124** Funktionseinheiten enthalten, etwa eine Ganzzahleinheit, die ausgebildet ist, Ganzzahlarithmetikoperationen der Addition und Subtraktion, sowie Verschiebungen, Drehungen, logische Operationen und Verzweigungsoperationen auszuführen. Eine Gleitkommaeinheit kann ebenso enthalten sein, um Gleitkommaoperationen auszuführen. Ein oder mehrere der Ausführungskerne **124** können ausgebildet sein, eine Adressenerzeugung für Lese- und Schreib-Speicheroperationen, die von der Lade/Speichereinheit **126** auszuführen sind, durchzuführen.

[0033] Der bzw. die Ausführungskerne **124** können auch Informationen hinsichtlich der Ausführung konditionaler Verzweigungsinstruktionen für die Verzweigungsvorhersageeinheit **132** bereitstellen. Wenn Information von dem Ausführungskern **124** anzeigt, dass eine Verzweigungsvorhersage nicht korrekt ist, kann die Verzweigungsvorhersageeinheit **132** Instruktionen, die auf die falsch vorhergesagte Verzweigung folgen und die in den Instruktionsbearbeitungspipeline eingespeist wurden, stornieren und die Vorabholeinheit **108** neu anweisen. Die neu angewiesene Vorabholeinheit **108** kann dann anfangen, den richtigen Satz an Instruktionen aus dem Instruktionscache-Speicher **106**, dem Ablaufverfolgungscache-Speicher **160** und/oder dem Systemspeicher **200** auszulesen. In derartigen Situationen können Ergebnisse der Instruktionen in der ursprünglichen Programmsequenz, die nach der fehlgedeuteten Verzweigungsinstruktion auftraten, verworfen werden, einschließlich jener, die spekulativ ausgeführt wurden und zeitweilig in der Lade/Speichereinheit **126** und/oder der Registerdatei **116** gespeichert wurden.

[0034] Ergebnisse, die von Komponenten innerhalb der bzw. den Ausführungskernen **124** erzeugt wurden, können auf den Ergebnisbus **130** zu der Registerdatei **116** ausgegeben werden, wenn ein Registerwert aktualisiert wird. Wenn der Inhalt einer Speicherzelle geändert wird, können in dem bzw. den Ausführungskernen **124** erzeugte Ergebnisse zu der Lade/Speichereinheit **126** zugeführt werden.

[0035] Die Lade/Speichereinheit **126** stellt eine Schnittstelle zwischen den Ausführungskernen **124** und dem Datencache-Speicher **128** bereit. In einer Ausführungsform ist die Lade/Speichereinheit **126** mit einem Lade/Speicherpuffer mit diversen Speicherplätzen für Daten und Adresseninformationen für anhängige Lade- bzw. Lesevorgänge oder Speicher- bzw. Schreibvorgänge versehen. Die Lade/Speichereinheit **126** kann ferner eine Abhängigkeitsüberprüfung für Leseinstruktionen im Hinblick auf anhängige Schreibinstruktionen durchführen, um sicherzustellen, dass die Datenkohärenz bewahrt bleibt.

[0036] Der Datencache-Speicher **128** ist ein Cache- bzw. Pufferspeicher, der zum Zwischenspeichern von Daten vorgesehen ist, die zwischen der Lade/Speicher-Einheit 126 und dem Systemspeicher **200** ausgetauscht werden. Ähnlich wie der Instruktionscache-Speicher **106**, der zuvor beschrieben ist, kann der Datencache-Speicher **128** in einer Vielzahl spezifischer Speicherkonfigurationen eingerichtet sein, zu denen eine teilassoziative Konfiguration gehört. Des weiteren können der Datencache-Speicher **106** und der Instruktionscache-Speicher **128** in einigen Ausführungsformen als ein vereinigter Cache- bzw. Pufferspeicher eingerichtet sein.

[0037] Die Abschlusswartereihe **102** stellt Signale, die Programmablaufverfolgung kennzeichnen, für den Ablaufverfolgungsgenerator **170** bereit. Der Ablaufverfolgungsgenerator **107** kann auch als eine Fülleinheit beschrieben werden. Der Ablaufverfolgungsgenerator **170** kann von der Abschlusswartereihe **102** erkannte Ablaufverfolgungsspuren in dem Ablaufverfolungscache-Speicher **160** speichern. Jede Ablaufverfolgungsspur kann Instruktionen enthalten, die in einer anderen Reihenfolge als die Programmreihenfolge gespeichert sind und die Teil diverser unterschiedlicher Basisblöcke sind. Ein Basisblock ist ein Satz aus Instruktionen, die entweder alle ausgeführt werden, oder die gar nicht ausgeführt werden. Anders ausgedrückt, wenn eine Instruktion eines Basisblocks ausgeführt wird, werden alle anderen Instruktionen dieses Basisblocks ebenfalls ausgeführt. Ein Beispiel eines Basisblocks ist ein Satz aus Instruktionen, der unmittelbar nach einer Verzweigungsinstruktion beginnt und mit einer weiteren Verzweigungsinstruktion endet. Die Ablaufverfolgungsspur oder – kennzeichnung, die in dem Ablaufverfolgungscache-Speicher **160** gespeichert sind, können in einigen Ausführungsformen decodierte oder teilweise decodierte Instruktionen enthalten. Im hierin verwendeten Sinne ist eine „Ablaufverfolgungsspur oder – kennzeichnung" eine Gruppe aus Instruktionen, die in einem einzelnen Ablaufverfolgungscache-Eintrag (beispielsweise einer einzelnen Reihe oder Zeile) in dem Ablaufverfolgungscache-Speicher **160** gespeichert sind. Die gesamte Information in einem Eintrag kann gleichzeitig abgerufen werden (beispielsweise in Reaktion auf das Setzen eines Lesezyklusses des Ablaufverfolgungscache-Speichers). Zu beachten ist, dass in alternativen Ausfüh-

rungsformen der Ablaufverfolgungsgenerator **170** mit dem Frontende des Prozessors (beispielsweise vor oder nach der Ausgabeeinheit) verbunden und ausgebildet sein kann, Ablaufverfolgungskennzeichen in Reaktion darauf zu erzeugen, dass Instruktionen abgeholt und/oder ausgegeben werden.

[0038] Die Vorabholeinheit **108** kann Instruktionen aus dem Ablaufverfolgungscache-Speicher **160** in die Ausgabeeinheit **104** einladen. Das Abholen von Instruktionen aus dem Ablaufverfolgungscache-Speicher **160** kann ein verbessertes Abholleistungsverhalten über Verzweigungsgrenzen hinweg im Vergleich zum Abholen von dem Instruktionscache-Speicher **106** liefern. Wenn beispielsweise eine vorhergesagte Verzweigung genommen wird, können Instruktionen in dem vorhergesagten genommenen Weg bereits in dem gleichen Ablaufverfolgungskennzeichen, wie die Verzweigung innerhalb des Ablaufverfolgungscache-Speichers **160** gespeichert sein. Ferner kann in Ausführungsformen, in denen Instruktionen in dem Ablaufverfolgungscache-Speicher **160** zumindest in teilweise decodierter Form gespeichert sind, das Abholen von Instruktionen aus dem Ablaufverfolgungscache-Speicher ermöglichen, dass die Decodiereinheit **140** zumindest teilweise umgangen wird, woraus sich eine reduzierte Anzahl an Ausgabezyklen für die zwischengespeicherten Instruktionen ergibt. Folglich ermöglicht es der Ablaufverfolgungscache-Speicher **160**, dass die Zeit, die erforderlich ist, um die zwischengespeicherten Instruktionen teilweise (oder vollständig) zu decodieren, über einige Ausführungsiterationen „amortisiert" wird, wenn Ablaufverfolgungskennzeichnen mehr als ein mal ausgeführt werden.

[0039] Zu beachten ist, dass der Prozessor **100** viele weitere Komponenten zusätzlich zu den gezeigten enthalten und/oder mit diesen verbunden sein kann. Beispielsweise können zusätzliche Ebenen an Zwischenspeicherung (intern und/oder extern zu dem Prozessor **100**) zwischen dem Prozessor **100** und dem Systemspeicher **200** enthalten sein. In ähnlicher Weise kann der Prozessor **100** in einigen Ausführungsformen eine integrierte Speichersteuerung enthalten, die ausgebildet ist, den Systemspeicher **200** zu steuern. Ferner können in den diversen Ausführungsformen die Verbindungen zwischen logischen Komponenten unterschiedlich sein.

Ablaufverfolgungscache-Speicher

[0040] **Fig.** 2 zeigt einen beispielhaften Ablaufverfolgungscache-Speicher **160** gemäß einer Ausführungsform. Der Ablaufverfolgungscache-Speicher **160** weist mehrere Einträge **162** auf. Jeder Eintrag **162** speichert eine Ablaufverfolgungsspur bzw. ein Ablaufverfolgungskennzeichen, das mehrere Instruktionen **165** enthält. Die Instruktionen **165** in einem Ablaufverfolgungskennzeichen müssen nicht notwendigerweise in der Programmreihenfolge gespeichert sein. Beispielsweise kann ein Eintrag **162** eine Verzweigungsinstruktion und die Instruktion enthalten, die das Ziel der Verzweigung ist, wenn die Verzweigung genommen wird (im Gegensatz zu der Instruktion, die in der Programmreihenfolge auf die Verzweigung folgt). In einigen Ausführungsformen können die Instruktionen **165** in jedem Ablaufverfolgungscache-Eintrag **162** in zumindest teilweise decodierter Form gespeichert sein. Im hierin verwendeten Sinne bezeichnet der Begriff „Ablaufverfolgungskennzeichen" einen Gruppe aus einer oder mehreren Instruktionen, die in einem einzelnen Ablaufverfolgungscacheeintrag **162** gespeichert sind.

[0041] Es sei nochmals auf Fig. 1 verwiesen. Zu beachten ist, dass die Abarbeitung eines Instruktionsstromes, der aus dem Ablaufverfolgungscache-Speicher **160** abgeholt wird, sich von der Bearbeitung eines Instruktionsstromes unterscheiden kann, der von dem Instruktionscache-Speicher **106** abgeholt wird. Der Instruktionsstrom aus dem Instruktionscache-Speicher **106** kann Informationen enthalten, die Instruktionsgrenzen kennzeichnen, selbst nachdem Instruktionen in eine oder mehrere Komponentenoperationen decodiert sind. Diese Grenzinformationen können von der Abschlusswartereihe **102** verwendet werden, um zu bestimmen, wann die Operation(en), die einer speziellen Instruktion entspricht, abgeschlossen werden kann. Typischerweise operiert die Abschlusswartereihe **102** so, dass Operationen in der ursprünglichen Programmreihenfolge abgeschlossen werden. Eine oder mehrere Operationen, die der gleichen Instruktion entsprechen, können gleichzeitig abgeschlossen werden.

[0042] Im Gegensatz dazu muss der Instruktionsstrom aus dem Ablaufverfolgungscache-Speicher **160** nicht notwendigerweise Informationen enthalten, die Instruktionsgrenzen kennzeichnen, wenn die Instruktionen vor der Speicherung in dem Ablaufverfolgungscache-Speicher **160** decodiert und/oder modifiziert wurden (beispielsweise durch Kombinieren von Instruktionen zur Erhöhung der Speichereffizienz innerhalb des Ablaufverfolgungscache-Speichers **160**). Folglich muss die Rücknahmereihenfolge **102** unter Umständen Instruktionen, die in einem derartigen Instruktionsstrom enthalten sind, in einer gröberen „Auflösung" abschließen. Beispielsweise beendet in einigen Ausführungsformen die Abschlusswartereihe **102** nicht notwendigerweise Instruktionen, die in einem Ablaufverfolgungskennzeichen enthalten sind, bis alle Instruktionen in diesem Ablaufverfolgungskennzeichen für die Beendigung bereit sind.

[0043] Ein Problem, das in einigen Prozessoren auf Basis eines Ablaufverfolgungs-Speichers auftreten kann, besteht darin, dass gewisse Instruktionen, die hierin als „Ausnahme"- oder Sonderinstruktionen bezeichnet werden, eine spezielle Handhabung erfor-

dern können, die einen Einfluss hat, wenn jene Instruktionen abgeschlossen werden können. Da das Zurückziehen bzw. Beenden dieser anderen Instruktionen (einschließlich jener, die früher in der Programmreihenfolge auftreten) in dem gleichen Ablaufverfolgungskennzeichen wie eine Sonderinstruktion davon abhängen kann, dass die Sonderinstruktion bereit für die Beendigung ist, kann eine Sonderinstruktion das Beenden der anderen Instruktionen in dem Ablaufverfolgungskennzeichnen blockieren. In einigen Situationen, in denen das Nichtbeenden der Sonderinstruktion das Beenden älterer Instruktionen, die in dem gleichen Ablaufverfolgungskennzeichen enthalten sind, blockieren kann, kann der Prozessor lahmgelegt werden. Wenn ferner eine Sonderinstruktion das Abschließen anderer Instruktionen in dem gleichen Ablaufverfolgungskennzeichen auf Grund des Fehlens einer Instruktionsgrenzeninformation, die von dem Ablaufverfolgungscache-Speicher bereitgestellt wird, blockiert wird, müssen u. U. die nicht beendbaren Instruktionen (und jüngere Instruktionen in der Programmreihenfolge, die bereits ausgegeben wurden) in diesem Ablaufverfolgungskennzeichen erneut aus dem Instruktionskennzeichenspeicher abgerufen und erneut ausgeführt werden.

[0044] Sonderinstruktionen schließen jene ein, die Ausnahmefälle und Interrupts erzeugen. Eine Sonderinstruktion kann unter Umständen auch deutlich länger für die Vervollständigung brauchen im Vergleich zu Instruktionen, die keine Sonderinstruktionen sind. Beispielsweise kann der Ausführungskern **124** eine typische Nicht-Sonderinstruktion in 1 bis 3 Ausführungszyklen abschließen. Im Gegensatz dazu kann eine Sonderinstruktion einen Interrupt erzeugen, der 10 mal so viele Ausführungszyklen (oder mehr) zum Ausführen benötigt. Bei zumindest einigen Sonderinstruktionen kann es vorteilhaft sein oder gar erforderlich sein, dass alle Instruktionen, die älter (in der Programmreihenfolge) sind als eine spezielle Sonderinstruktion, dass diese unabhängig von dieser Sonderinstruktion beendet werden können. Wie zuvor angemerkt ist, kann das Fehlen einer Instruktionsgrenzeninformation in einem Ablaufverfolgungskennzeichen die Möglichkeit kleiner machen oder ganz aufheben, dass unterschiedliche Instruktionen in dem gleichen Ablaufverfolgungskennzeichen unabhängig voneinander abgeschlossen werden können. Folglich kann das Mithineinnehmen einer Sonderinstruktion in ein Ablaufverfolgungskennzeichen zusammen mit anderen Instruktionen schließlich dazu führen, dass alle Instruktionen in diesem Ablaufverfolgungskennzeichen erneut aus dem Instruktionscache-Speicher ausgelesen werden müssen, um die Grenzinformation zu erhalten.

[0045] Um die Wahrscheinlichkeit zu reduzieren, dass eine Sonderinstruktion das Abschließen der anderen Instruktionen blockiert und/oder bewirkt, dass andere Instruktionen verworfen, erneut abgerufen und erneut ausgeführt werden, kann der Ablaufverfolgungsgenerator **170** ausgebildet sein, Sonderinstruktionen zu erkennen und diese Instruktionen nicht in einem Ablaufverfolgungskennzeichen zu speichern, das andere Instruktionen aufweist. In einigen Ausführungsformen kann der Ablaufverfolgungsgenerator **170** eine erkannte Sonderinstruktion in einem Ablaufverfolgungskennzeichen ablegen, das lediglich diese Sonderinstruktion enthält. Obwohl eine derartige Ausbildung die Speichereffizienz des Ablaufverfolgungscache-Speichers reduzieren kann, ist es dennoch möglich, ein Abholen aus dem Ablaufverfolgungscache-Speicher fortzuführen anstatt zu bewirken, dass das Abholen zwischen dem Instruktionscache-Speicher **106** und dem Ablaufverfolgungscache-Speicher **160** abwechselt. In anderen Ausführungsformen speichert der Ablaufverfolgungsgenerator **170** erkannte Sonderinstruktionen gar nicht in dem Ablaufverfolgungscache-Speicher **160**, so dass Sonderinstruktionen stets von dem Instruktionscache-Speicher **106** abgerufen werden. Der Ablaufverfolgungsgenerator **170** ist ein Beispiel einer Einrichtung zum Speichern einer Gruppe aus Instruktionen in einem Ablaufverfolgungscache-Eintrag innerhalb eines Ablaufverfolgungscache-Speichers.

[0046] Der Ablaufverfolgungsgenerator **170** kann Sonderinstruktionen erkennen, indem jede mögliche Instruktion zur Speicherung in dem Ablaufverfolgungscache-Speicher auf gewisse Operationscodierungen, Operanden, Adressenmodi und andere Charakteristiken hin untersucht wird. In einer Ausführungsform kann das Erkennen einer oder mehrerer Eigenschaften möglicher Sonderinstruktionen in dem Ablaufverfolgungsgenerator **170** einprogrammiert sein. Zu beachten ist, dass in derartigen Ausführungsformen der Ablaufverfolgungsgenerator **170** ausgebildet sein kann, um stets gewisse Arten an Instruktionen als Sonderinstruktionen zu erkennen, selbst wenn diese Arten an Operationen ohne spezielle Handhabung unter gewissen Umständen ausführbar sind. In einigen Ausführungsformen kann die Decodiereinheit **140** Markierungen, die Sonderinstruktionen kennzeichnen, während der Instruktionsdecodierung erzeugen. Jede Markierung kann zu dem Prozessor **100** zusammen mit der decodierten Instruktion, die durch diese Markierung als Sonderinstruktion gekennzeichnet ist, weitergeleitet werden. Somit kann anstatt (oder zusätzlich) zu dem unabhängigen Bestimmen, ob decodierte Instruktionen Sonderinstruktionen sind, der Ablaufverfolgungsgenerator **170** sich auf die Markierungen stützen, die von der Decodiereinheit **140** erzeugt wurden, um Sonderinstruktionen zu erkennen. Somit sind sowohl der Ablaufverfolgungsgenerator **170** als auch die Decodiereinheit **140** Beispiele einer Einrichtung zum Erfassen einer Sonderinstruktion innerhalb der Gruppe aus Instruktionen.

[0047] In Reaktion auf das Erkennen einer Sonder-

instruktion in einem Strom aus Instruktionen, die in dem Ablaufverfolgungscache-Speicher **160** gespeichert werden, kann der Ablaufverfolgungsgenerator **170** das Speichern von Instruktionen in dem aktuellen Ablaufverfolgungskennzeichen beenden, selbst wenn noch Platz für die Speicherung weiterer Instruktionen in diesem Ablaufverfolgungskennzeichen vorhanden ist. Wenn der Ablaufverfolgungsgenerator ausgebildet ist, Sonderinstruktionen in einem alleinstehenden Ablaufverfolgungskennzeichnen in dem Ablaufverfolgungscache-Speicher **160** zu speichern, kann der Ablaufverfolgungsgenerator einen neuen Ablaufverfolgungscache-Eintrag reservieren, um die Sonderinstruktion zu speichern. Der Ablaufverfolgungsgenerator **170** kann die nächste Instruktion, die unmittelbar nach der Sonderinstruktion in der Programmreihenfolge auftritt, in einem weiteren Ablaufverfolgungscache-Eintrag speichern. Wenn der Ablaufverfolgungsgenerator **170** stattdessen ausgebildet ist, Sonderinstruktionen aus dem Ablaufverfolgungscache-Speicher **160** generell auszuschließen, kann der Ablaufverfolgungsgenerator einfach das aktuelle Ablaufverfolgungskennzeichen beim Erkennen der Sonderinstruktion beenden (solange keine älteren Instruktionen bereits in dem aktuellen Ablaufverfolgungskennzeichen gespeichert wurden) und damit beginnen, die nächste Nicht-Sonderinstruktion in der Programmreihenfolge, die ein Kandidat für die Speicherung in dem Ablaufverfolgungscache-Speicher ist, in dem nächsten Ablaufverfolgungskennzeichen zu speichern. Zu beachten ist, dass de Ablaufverfolgungsgenerator **170** Ablaufinformation in dem aktuellen Ablaufverfolgungskennzeichen setzen kann, um anzuzeigen, dass die Sonderinstruktion die nächste abzuholende Instruktion ist. Folglich kann in Reaktion darauf, dass dieses Ablaufverfolgungskennzeichen abgerufen wird, die Vorabholeinheit **170** ausgebildet sein, die in der Ablaufinformation gekennzeichnete Instruktion aus dem Instruktionscache-Speicher **106** abzurufen. Somit geht der Ausführungsablauf von dem Ablaufverfolgungscache-Speicher **160** zu dem Instruktionscache-Speicher **106** auf Grund des Ausschließens der Sonderinstruktion aus dem Ablaufverfolgungscache-Speicher **160** über.

[0048] Einige Sonderinstruktionen können unter Umständen von dem Ablaufverfolgungsgenerator **170** nicht erkannt werden. Beispielsweise können einige Instruktionen sich unterschiedlich verhalten, jedes mal wenn diese Instruktionen ausgeführt werden. Auf Grund der Natur dieser Instruktionen (beispielsweise sind derartige Instruktionen u.U. selten Sonderinstruktionen) kann der Ablaufverfolgungsgenerator **170** ausgebildet sein, nicht automatisch diese Instruktionen als Sonderinstruktionen zu kennzeichnen. Als Folge davon können diese Sonderinstruktionen in einem Ablaufverfolgungskennzeichen zusammen mit anderen Instruktionen durch den Ablaufverfolgungsgenerator **170** gespeichert werden. Um zu verhindern, dass das Mithereinnehmen der Sonderinstruktionen Probleme nach sich zieht auf Grund einer blockierten Beendigung anderer Instruktionen in dem gleichen Ablaufverfolgungskennzeichen, kann die Neustartlogik **190** überwachen, wie viele Taktzyklen vergangen sind, seit die Abschlusswartereihe **102** zuletzt Operationen abgeschlossen hat. Wenn keine Operationen in einer vorbestimmten Anzahl von Taktzyklen beendet wurden, kann die Neustartlogik **190** den Arbeitsablauf des Prozessors **190** aus dem Instruktionscache-Speicher **106** heraus erneut starten. In einer Ausführungsform startet die Neustartlogik **190** das Ausführen, indem die bearbeitende Pipeline zurückgesetzt wird und die Instruktionen neu abgerufen werden, wobei mit der ältesten noch anhängigen noch nicht abgeschlossenen Instruktion innerhalb des Prozessors **100** begonnen wird. Wenn die Instruktionen aus dem Instruktionscache-Speicher **106** abgerufen werden, sind für die Abschlusswartereihe **102** Instruktionen über Instruktionsgrenzen verfügbar. Wenn daher das Nichtbeenden, das von der Neustartlogik **190** erkannt wird, durch eine Sonderinstruktion hervorgerufen wird, die das Beenden eines Ablaufverfolgungskennzeichens blockiert hat, kann das Abrufen der Instruktionen aus dem Instruktionscache-Speicher **106** die Abschlusswartereihe **102** in die Lage versetzen, die problematische Sonderinstruktion unabhängig abzuschließen, ohne dass unnötigerweise das Beenden anderer Instruktionen blockiert wird.

[0049] Gewisse Instruktionen können nicht erneut ausführbar sein und somit können derartige Instruktionen unkorrekt sein, wenn diese von dem Ablaufverfolgungscache-Speicher **160** ausgeführt, für das Beenden durch eine Sonderinstruktion, die in dem gleichen Ablaufverfolgungskennzeichen enthalten ist, blockiert werden und dann auf Grund der Funktion der Neustartlogik **190** erneut ausgeführt werden. Beispielsweise kann das Verhalten gewisser Ladevorgänge (beispielsweise einige I/O-Ladevorgänge) den ausgelesenen Wert modifizieren. Um Probleme zu vermeiden, die auftreten können, wenn derartige Instruktionen erneut ausgeführt werden, kann der Ablaufverfolgungsgenerator **170** ausgebildet sein, die Instruktionen als eine weitere Klasse an Sonderinstruktionen zu kennzeichnen, die nicht mit anderen Instruktionen in Ablaufverfolgungskennzeichen zu speichern sind. Somit kann der Ablaufverfolgungsgenerator **170** diese Sonderinstruktionen von dem Ablaufverfolgungscache-Speicher **160** ausschließen oder diese Sonderinstruktionen in einzelnen Ablaufverfolgungskennzeichen speichern, die keine anderen Instruktionen enthalten (beispielsweise wie dies zuvor in Bezug auf die Sonderinstruktionen im Allgemeinen beschrieben ist).

[0050] Wie gewisse andere Sonderinstruktion können einige nicht ausführbare Instruktionen unter Umständen nicht von dem Ablauferzeugungsgenerator **170** erkannt werden. Beispielsweise kann ein Lade-

vorgang auf eine Adresse des Systemspeichers **200** während einer Programmiteration und auf ein I/O-Register in dem nächsten Schritt zugreifen. Nach der ersten Iteration kann der Ladevorgang in einem Ablaufverfolgungskennzeichen gespeichert werden. Während des nächsten Schrittes kann der Ladevorgang nicht erneut ausführbar sein auf Grund der Natur des Auslesens aus dem I/O-Register. Wenn daher eine weitere Instruktion in dem gleichen Ablaufverfolgungskennzeichen wie der Ladevorgang eine Sonderinstruktion ist (beispielsweise wenn eine weitere Instruktionen einen Seitenfehler erzeugt), kann die Beendigung des Ablaufverfolgungskennzeichens lang genug verzögert werden, so dass die Neustartlogik **190** aufgerufen wird. Jedoch könnte dies in nicht akzeptabler Weise ein erneutes Ausführen des Ladevorgangs hervorrufen. Um diese Situation zu vermeiden, kann die Ausgabeeinheit **104** oder eine weitere Komponente in dem Prozessor ausgebildet sein, den Ladevorgang zu einem gewissen Punkt, bevor der Ladevorgang tatsächlich ausgeführt wird, diesen als nicht mehr erneut ausführbar zu erkennen. In Reaktion auf das Erkennen einer nicht erneut ausführbaren Instruktion innerhalb einer Gruppe aus Instruktionen, die aus dem Ablaufverfolgungscache-Speicher **160** abgerufen werden, kann die Neustartlogik **190** veranlassen, dass die Prozessorpipeline zurückgesetzt wird (vor dem Ausführen der nicht erneut ausführbaren Instruktion) und ein erneuter Start des Ausführens aus dem Instruktionscache-Speicher **106** heraus durchgeführt wird.

Beispielhafte Ablaufverfolgung

[0051] Fig. 3a zeigt eine Ausführungsform eines Ablaufverfolgungscache-Eintrags **162**, der ausgebildet ist, zumindest teilweise decodierte Instruktionen zu speichern. In der Ausführungsform aus Fig. 3a ist ein Ablaufverfolgungscache-Eintrag **162** in der Lage bis zu 8 Komponentenoperationen, die in einer Gruppe aus decodierten und/oder teilweisen decodierten Instruktionen enthalten sind, in Operationsspeichereinheiten **116a** bis **116h** zu speichern. Zu beachten ist, dass in anderen Ausführungsformen weniger oder zusätzliche Operationsspeichereinheiten **166** enthalten sein können, so dass die Speicherung einer anderen Anzahl an Operationen innerhalb jedes Ablaufverfolgungscache-Eintrags **162** möglich ist.

[0052] Gewisse Operationsspeichereinheiten **166** innerhalb eines Ablaufverfolgungscache-Eintrags **162** können für gewisse Arten an Operationen reserviert sein. Beispielsweise werden in einigen Ausführungsformen die Operationsspeichereinheiten **166a** bis **166d** verwendet, um Speicheroperationen zu speichern. In ähnlicher Weise können die Operationsspeichereinheiten **116e** bis **166h** für die Speicherung von Datenoperationen verwendet werden. Zu beachten ist, dass in anderen Ausführungsformen gewisse Arten an Operationen mit gewissen Arten an Speichereinheiten in unterschiedlicher Weise verknüpft werden können (oder überhaupt nicht verknüpft sind).

[0053] Zusätzlich dazu, dass mehrere Operationsspeichereinheiten **166** enthalten sind, beinhaltet jeder Ablaufverfolgungscache-Eintrag **162** auch eine Identifizierungsmarkierung **164** und eine Ablaufsteuerungsinformation **168**. Die Markierung **164** kann ähnlich zu einer Markierung in dem Instruktionscache-Speicher **106** sein, die es der Vorabholeinheit **108** ermöglicht, zu bestimmen, ob eine gegebene Operation auf den Ablaufverfolgungscache-Speicher **160** abzielt oder nicht. Beispielsweise kann die Markierung **164** alle oder einige der Adressenbits enthalten, die eine Instruktion in dem Ablaufverfolgungscache-Eintrag kennzeichnen (beispielsweise kann die Markierung die Adresse der frühesten Instruktion in der Programmreihenfolge enthalten, die in diesem Ablaufverfolgungskennzeichen gespeichert ist). In einigen Ausführungsformen weist die Markierung ausreichend Information auf, so dass die Adresse jeder Instruktion (oder zumindest der frühesten Instruktion innerhalb jeder Hierarchiegruppe, wie dies nachfolgend detaillierter beschrieben ist) unabhängig identifiziert werden kann, wobei in dem Ablaufverfolgungskennzeichen gespeicherte Informationen verwendet wird. In anderen Ausführungsformen ist lediglich die Adresse der frühesten Instruktion innerhalb eines Ablaufverfolgungskennzeichens verfügbar.

[0054] In der dargestellten Ausführungsform kann jedes Ablaufverfolgungskennzeichen bis zu zwei Verzweigungsinstruktionen enthalten. In anderen Ausführungsformen können weniger oder mehr Verzweigungsinstruktionen innerhalb jedes Ablaufverfolgungskennzeichens enthalten sein. Die Ablaufsteuerinformation **168** kann eine Marke für jede Verzweigungsinstruktion, die in dem Ablaufverfolgungskennzeichen enthalten ist, aufweisen. Die Marke kann eine Kennzeichnung sein, die die Adresse identifiziert, zu der die Steuerung verzweigen soll in Abhängigkeit von der Auflösung (genommen, nicht genommen) einer entsprechenden Abzweigung. Somit kann jedes Element der Ablaufsteuerungsinformation **168** mit einer speziellen Verzweigungsinformation verknüpft sein. Beispielsweise kann in einer Ausführungsform eine einzelne Ablaufsteuerungsinformationsspeicherzelle in einem Ablaufverfolgungskennzeichen mit der ersten Verzweigungsoperation in dem Ablaufverfolgungskennzeichen verknüpft sein und die andere Ablaufsteuerinformationsspeicherzelle kann mit der zweiten Verzweigung in dem Ablaufverfolgungskennzeichen verknüpft sein. Alternativ kann die Ablaufsteuerinformation Markierungen oder andere Informationen enthalten, die die Verzweigungsoperation kennzeichnen, mit der diese Ablaufsteuerinformation verknüpft ist. In noch anderen Ausführungsformen kann eine Verzweigungsvorhersage und/oder Information, die angibt, welche Ablaufsteu-

erinformation einer Verzweigungsoperation entspricht, kann mit dieser Verzweigungsoperation innerhalb eines Operationsspeichers **166** gespeichert sein.

Bereitstellung einer erhöhten Beendigungsauflösung innerhalb von Ablaufverfolgungskennzeichen.

[0055] Wie es zuvor erwähnt ist, kann es eine begrenzte Information über Instruktionsgrenzen in einem Ablaufverfolgungskennzeichen geben. Wenn beispielsweise Instruktionen teilweise in ihre Komponentenoperationen vor der Speicherung in einem Ablaufverfolgungskennzeichen decodiert werden, kann unter Umständen keine Information in dem Ablaufverfolgungskennzeichen enthalten sein, die die unterschiedlichen Instruktionen in diesen Ablaufverfolgungskennzeichen abgrenzt. Wenn nach dem Decodieren Komponentenoperationen unterschiedlicher Instruktionen kombiniert, neu geordnet und/oder modifiziert werden, kann es ferner noch schwieriger sein, Instruktionsgrenzen zu erkennen.

[0056] Um eine verbesserte Instruktionsgrenzeninformation in einem Ablaufverfolgungskennzeichen bereitzustellen, werden decodierte Instruktionen innerhalb eines Ablaufverfolgungskennzeichens in unterschiedliche „Vitalitäts"- bzw. „Lebendigkeits"-gruppen aufgeteilt. Der Ablaufverfolgungsgenerator **170** kann jede decodierte Instruktion, die in einem speziellen Ablaufverfolgungscache-Eintrag gespeichert wird, als zu einer speziellen Prioritätsgruppe innerhalb dieses Ablaufverfolgungskennzeichens gehörig markieren. Informationen über die Instruktionsgrenzen decodierter Instruktionen in jeder Vitalitätsgruppe sind unter Umständen nicht verfügbar. Jedoch können die Vitalitätsgruppen selbst so erzeugt werden, dass decodierte Instruktionen auf der Grundlage von Vitalitätsgruppengrenzen abgeschlossen werden.

[0057] Jede Operation **166** innerhalb eines Ablaufverfolgungskennzeichens kann eine zugehörige „Vitalitäts"-kennzeichnung aufweisen. Die Vitalitätskennzeichnung jeder Operation kann mit der zugehörigen Operation in einer der Operationsspeichereinheiten **166** gespeichert werden. Die Vitalitätskennzeichnungen geben die Vitalitätsgruppe an, zu der jede Operation gehört. Jede Vitalitätsgruppe ist eine Gruppe aus Operationen innerhalb des Ablaufverfolgungskennzeichens, die Teil des gesamten Basisblocks sind. Beispielsweise können die Operationen bis zu und einschließlich der ersten Verzweigungsoperation in der Programmreihenfolge innerhalb des Ablaufverfolgungskennzeichens in einer einzelnen Vitalitätsgruppe enthalten sein. Operationen, deren Ausführung von der Auflösung der ersten Verzweigungsoperation abhängt, können in eine weitere Vitalitätsgruppe eingeordnet werden. Somit ist eine Vitalitätsgruppe ähnlich zu einem Basisblock dahingehend, dass Operationen in der gleichen Vitalitätsgruppe entweder alle ausgeführt werden oder dass keine Instruktion ausgeführt wird. Eine Vitalitätsgruppe unterscheidet sich von einem Basisblock dahingehend, dass Operation in dem selben Basisblock in unterschiedlichen Ablaufverfolgungskennzeichen enthalten sein können (beispielsweise können eine Operationen innerhalb des Basisblocks in der letzten Vitalitätsgruppe innerhalb eines Ablaufverfolgungskennzeichens enthalten sein und die verbleibenden Operationen in diesem Basisblock können in der ersten Vitalitätsgruppe eines weiteren Ablaufverfolgungskennzeichens enthalten sein). Somit sind Operationen innerhalb der gleichen Vitalitätsgruppe notwendigenneise Teil des gleichen Basisblocks, jedoch sind Operationen innerhalb des gleichen Basisblocks nicht notwendigerweise Teil der gleichen Vitalitätsgruppe (d. h., wenn dieser Basisblock mehr als einen Ablaufverfolgungscache-Eintrag überspannt).

[0058] Die Vitalitätskennzeichnungen geben die Programmreihenfolge jeder Operation im Verhältnis zu der bzw. den Verzweigungsoperationen, die in dem gleichen Ablaufverfolgungscache-Eintrag enthalten sind, an. Operationen in einer ersten Vitalitätsgruppe sind nicht abhängig von dem Ergebnis der ersten Verzweigungsoperation innerhalb des Ablaufverfolgungskennzeichens. Zu beachten ist, dass, da das Ausführen der ersten Verzweigungsoperation nicht konditional ist, die erste Verzweigungsoperation Teil der ersten Vitalitätsgruppe ist. Diese Operationen in der ersten Vitalitätsgruppe können mit einer anderen Vitalitätskennzeichnung belegt werden, als verwendet wird, um eine zweite Vitalitätsgruppe aus Operationen zu kennzeichnen, die nach der ersten Verzweigungsoperation auftreten. In ähnlicher Weise können Instruktionen in einer dritten Vitalitätsgruppe, die von den Ergebnissen sowohl der ersten Verzweigungsoperation als auch der zweiten Verzweigungsoperation abhängen (d. h. die letzte Operation, die in der zweiten Vitalitätsgruppe enthalten ist) als eine noch weitere Vitalitätskennzeichnung identifiziert werden. Vitalitätskennzeichnungen ermöglichen es, dass Operationen in einem Ablaufverfolgungscache-Eintrag **162** in einer Reihenfolge gespeichert werden, die sich von ihrer Programmreihenfolge unterscheidet, während dennoch die Programmreihenfolge (im Vergleich zu den Verzweigungsoperationen) der Operationen bestimmt werden kann.,

[0059] Ferner ermöglichen es die Vitalitätskennzeichnungen, dass die Ausgabeeinheit **104** vorhersagt, welche Operationen innerhalb eines speziellen Ablaufverfolgungskennzeichens ausgeführt werden. Operationen in der ersten Vitalitätsgruppe (d. h. Operationen, die nicht von der ersten Verzweigung, die in einem Ablaufverfolgungskennzeichen enthalten ist, abhängen) werden stets ausgeführt (unter der Annahme, dass überhaupt Operationen innerhalb des Ablaufverfolgungskennzeichens ausgeführt werden). Jedoch hängt die Ausführung von Operationen in der

zweiten Vitalitätsgruppe (d. h. Operationen, die von der ersten Verzweigung abhängen) von dem Ergebnis der ersten Verzweigung ab. Beispielsweise können in Ausführungsformen, in denen der Ablaufverfolgungsgenerator **170** an der Rückseite der bearbeitenden Pipeline des Prozessors **100** enthalten ist, Operationen in Ablaufverfolgungskennzeichen gemäß der Reihenfolge gespeichert werden, in der diese Operationen ausgeführt werden, so dass, wenn die Operationen nachfolgend erneut ausgeführt werden, auf diese von dem Ablaufverfolgungscache-Speicher aus, anstatt von dem Instruktionscache-Speicher aus, zugegriffen wird. Wenn somit die erste in dem Ablaufverfolgungskennzeichen gespeicherte Verzweigung das erste mal genommen wird, wenn die Verzweigung ausgeführt wurde, können die in der zweiten Vitalitätsgruppe gespeicherten Operationen die Operationen sein, die ausgeführt werden sollten, wenn die erste Verzweigung genommen wird. Wenn daher das Ablaufverfolgungskennzeichen später von der Ausgabeeinheit **104** zugeführt wird und die aktuelle Verzweigungsvorhersage für die erste Verzweigung „nicht genommen" wird, kann die Ausgabeeinheit **104** vorhersagen, dass die Operationen in der zweiten Vitalitätsgruppe nicht ausgeführt werden sollen. Die Ablaufsteuerinformation **168**, die mit der ersten Verzweigungsoperation verknüpft ist, kann ebenso verwendet werden, um das vorläufige Abholen von Instruktionen oder Ablaufverfolgungskennzeichen zu beginnen, die ausgeführt werden sollten, wenn die erste Verzweigung nicht genommen wird.

[0060] Fig. 3b zeigt beispielhafte Vitalitätscodierungen, die in einer Ausführungsform verwendet werden können. Die erste Vitalitätscodierung „bedingungslos lebendig bzw. vital" kann verwendet werden, um die erste (in der Programmreihenfolge) Vitalitätsgruppe innerhalb des Ablaufverfolgungskennzeichens zu kennzeichnen. Operationen in dieser Vitalitätsgruppe sind bedingungslos „lebendig" bzw. vital in dem Sinne, dass, wenn das Ablaufverfolgungskennzeichen ausgeführt wird, diese Operationen stets ausgeführt werden. Die erste Vitalitätsgruppe kann eine Verzweigungsoperation als die letzte Operation (in der Programmreihenfolge) innerhalb dieser Vitalitätsgruppe aufweisen. Diese Operationen hängen nicht von Verzweigungsoperationen, die in dem gleichen Ablaufverfolgungskennzeichen sind, ab. In vielen Ausführungsformen können diese Operationen in einer beliebigen Reihenfolge im Vergleich zueinander innerhalb des Ablaufverfolgungskennzeichens gespeichert werden.

[0061] Die nächste Vitalitätscodierung „folgend auf erste Verzweigung" wird verwendet, um die zweite (in der Programmreihenfolge) Vitalitätsgruppe in dem Ablaufverfolgungskennzeichen zu kennzeichnen. Diese Operationen sind von dem Ergebnis der ersten Verzweigungsoperation abhängig, die in der vorhergehenden Vitalitätsgruppe enthalten ist. Zu beachten ist, dass, wenn Operationen innerhalb der Vitalitätsgruppe nicht spekulativ ausgeführt werden, alle Operationen innerhalb dieser Vitalitätsgruppe ausgeführt werden.

[0062] Die dritte Vitalitätscodierung, „folgend auf zweite Verzweigung", wird verwendet, um den dritten (in Programmreihenfolge) Basisblock in dem Ablaufverfolgungskennzeichen zu kennzeichnen. Diese Operationen sind von dem Ergebnis sowohl der ersten als auch der zweiten Verzweigungsoperation in dem Ablaufverfolgungskennzeichen abhängig. Somit werden diese Operationen nicht ausgeführt, selbst wenn Operationen mit der zweiten Vitalitätscodierung ausgeführt werden. Wenn wie zuvor Operationen innerhalb dieser Vitalitätsgruppe nicht spekulativ ausgeführt werden, werden alle Operationen innerhalb dieser Vitalitätsgruppe ausgeführt.

[0063] Die „bedingungslos tot"-Vitalitätscodierung kann verwendet werden, um eine unbenutzte Operationsspeichereinheit **166** zu kennzeichnen. Operationsspeichereinheiten **166** können unbenutzt sein, wenn mehr als die maximale Anzahl an Verzweigungsoperationen innerhalb eines Satzes an Operationen auftritt. Wenn beispielsweise bis zu 8 Operationen in einem Ablaufverfolgungskennzeichen gespeichert werden können und lediglich zwei Verzweigungsoperationen in einem Ablaufverfolgungskennzeichen gespeichert werden können, dann werden weniger als 8 Operationen in einem Ablaufverfolgungskennzeichen gespeichert, wenn ein gegebener Satz aus 8 Operationen drei Verzweigungsoperationen enthält. Wenn daher eine maximale Grenze hinsichtlich der Anzahl N an Verzweigungsoperationen gegeben ist, die in einem Ablaufverfolgungskennzeichen gespeichert werden kann, kann eine Speicherung der N-ten Verzweigungsoperation in dem Ablaufverfolgungskennzeichen dieses sodann abschließen. Wenn ferner in einigen Ausführungsformen eine gewisse Operation in einem Ablaufverfolgungskennzeichen gespeichert wird, können nachfolgende Operationen (in der Programmreihenfolge) unter Umständen nicht in dem Ablaufverfolgungskennzeichen gespeichert werden, selbst wenn Operationsspeicherplatz verfügbar ist. Beispielsweise können Unterroutinenaufrufe ein Ablaufverfolgungskennzeichen beenden (beispielsweise so, dass die erste Operation innerhalb der Unterroutine als die erste Operation in einem weiteren Ablaufverfolgungskennzeichen gespeichert wird, so dass die Markierung dieser Operation als eine Markierung für das Ablaufverfolgungskennzeichen verwendet werden kann). In ähnlicher Weise können Rücksprungoperationen für Unterroutinen ein Ablaufverfolgungskennzeichen beenden.

[0064] Es sei wieder auf Fig. 1 verwiesen; der Ablaufverfolgungsgenerator **170** kann ausgebildet sein, die geeignete Vitalitätscodierung für jede Operation zu erzeugen und diese Vitalitätscodierung in der glei-

chen Operationsspeichereinheit **166** wie die Operation innerhalb eines Ablaufverfolgungscache-Eintrags **162** in dem Ablaufverfolgungscache-Speicher **160** zu speichern. Der Ablaufverfolgungsgenerator **170** kann Vitalitätscodierungen für jede Operation auf der Grundlage von Informationen über das Ergebnis jeder Verzweigung erzeugen, wenn jede Verzweigungsoperation das erste mal ausgeführt wurde. Diese Information kann der Abschlusswartereihe **102** zugeführt werden. Weitere Informationen können von der Verzweigungsvorhersageeinheit **132** bereitgestellt werden, die das vorhergesagte Ergebnis und/oder den Erfolg jeder Vorhersage für jede Verzweigung in einer oder mehreren vorhergehenden Ausführungen kennzeichnen.

[0065] Wenn Instruktionen aus dem Ablaufverfolgungscache-Speicher **160** abgeholt werden, kann die Vitalitätscodierung, die mit jeder Operation verknüpft ist, zu dem Prozessor zusammen mit der Instruktion weitergeleitet werden. Die Abschlusswartereihe **102** kann die Vitalitätscodierung für jede Operation verwenden, um die Grenzen zu bestimmen, mit denen Operationen beendet werden können. Wenn beispielsweise anstelle des Blockierens des Abschließens von Operationen in einem Ablaufverfolgungskennzeichen alle Operationen in diesem Ablaufverfolgungskennzeichen bereit sind für die Beendigung, kann die Abschlusswartereihe **102** eine Teilmenge der Operationen in dem Ablaufverfolgungskennzeichen abschließen, sobald alle Operationen in der gleichen Vitalitätsgruppe in diesem Ablaufverfolgungskennzeichen für die Fertigstellung bereit sind. In einigen Ausführungsformen kann diese erhöhte Auflösung für die Operationsbeendigung die Wahrscheinlichkeit reduzieren, dass die Neustartlogik **100** die Ausführung auf Grund einer blockierten Fertigstellung von Operationen neu starten muss.

[0066] Fig. 4 zeigt eine Ausführungsform eines Verfahrens zum Erzeugen von Ablaufverfolgungskennzeichen zur Speicherung in einem Ablaufverfolgungscache-Speicher. Bei **10** werden eine oder mehrere Instruktionen zur Speicherung in dem Ablaufverfolgungscache-Speicher empfangen. Wenn eine der Instruktionen eine Sonderinstruktion ist (beispielsweise, wenn eine Instruktion eine hohe Wahrscheinlichkeit aufweist, eine Ausnahmehandhabung hervorzurufen, oder wenn eine Instruktion nicht erneut ausführbar ist), werden die Sonderinstruktionen nicht in dem gleichen Ablaufverfolgungskennzeichen wie andere Instruktionen gespeichert, wie dies bei **12** und **14** gezeigt ist. Wenn keine der Instruktionen, die bei **10** empfangen werden, Sonderinstruktionen sind, werden diese Instruktionen zusammen in dem gleichen Ablaufverfolgungskennzeichen gespeichert und/oder in das gleiche Ablaufverfolgungskennzeichen wie zusätzliche Nicht-Sonderinstruktionen integriert, wie dies bei **12** und **16** gezeigt ist.

[0067] Fig. 5 ist ein Flussdiagramm einer Ausführungsform eines Verfahrens zum Handhaben nicht erkannter Sonderinstruktionen, die nachteiligerweise in einem Ablaufverfolgungskennzeichen innerhalb des Ablaufverfolgungscache-Speichers enthalten sind. Bei **20** wird ein Instruktionsstrom aus dem Ablaufverfolgungscache-Speicher abgerufen. Wenn eine Sonderinstruktion in dem abgerufenen Instruktionsstrom erkannt wird, wie dies bei **22** gezeigt ist, kann die Pipeline zurückgesetzt werden und die Abarbeitung durch den Prozessor kann aus dem Instruktionscache-Speicher heraus neu gestartet werden, wie dies bei **24** gezeigt ist.

[0068] Einige Sonderinstruktionen können unter Umständen bei **22** nicht innerhalb des Instruktionsstromes erkannt worden sein. Wenn eine derartige Instruktion eine Ausnahme hervorruft, die wiederum das Beenden der anderen Instruktionen in dem gleichen Ablaufverfolgungskennzeichen wie die Sonderinstruktion blockiert, kann die Prozessorpipeline zurückgesetzt werden und die Ausführung kann aus dem Instruktionscache-Speicher heraus neu gestartet werden, wie dies bei **26** und **28** gezeigt ist. Das Bestimmen, ob die Beendigung von Instruktionen blockiert ist, kann das Überwachen des Abschließens der Instruktion beinhalten. Wenn keine Instruktionen für eine vordefinierte Zahl an Zyklen abgeschlossen werden, so kann die Operationsbeendigung als blockiert betrachtet werden. Wenn die Beendigung von Instruktionen nicht blockiert ist, kann es dem Instruktionsstrom, der aus dem Ablaufverfolgungscache-Speicher abgerufen wurde, gestattet werden, ausgeführt und beendet zu werden, wie dies bei **26** und **30** gezeigt ist.

Beispielhafte Computersysteme

[0069] Fig. 6 zeigt eine Blockansicht einer Ausführungsform eines Computersystems **400**, das einen Prozessor **100** aufweist, der mit einer Vielzahl an Systemkomponenten über eine Busbrücke **402** gekoppelt ist. Der Prozessor **100** kann eine Ausgabeeinheit **104**, einen Ablaufverfolgungscache-Speicher **160**, eine Abschlusswartereihe **102**, eine Neustartlogik **190** und/oder einen Ablaufverfolgungscache-Generator **170** aufweisen, wie dies zuvor beschrieben ist. Andere Ausführungsformen eines Computersystems sind möglich und hierin mit eingeschlossen. In dem gezeigten System ist ein Hauptspeicher **200** mit der Busbrücke **402** über einen Speicherbus **406** verbunden, und eine Graphiksteuerung **408** ist mit der Busbrücke **402** über einen AGP-Bus **410** verbunden. Diverse PCI-Geräte **412a** bis **412b** sind mit der Busbrücke **402** über einen PCI-Bus **414** verbunden. Eine sekundäre Busbrücke **416** kann ebenso vorgesehen sein, um eine Schnittstelle zu einem oder mehreren EISA- oder ISA-Geräten **418** über einen EISA/ISA-Bus **420** bereitzustellen. In diesem Beispiel ist der Prozessor **100** mit einer Busbrücke **402** über

einen CPU-Bus **424** und mit einem optionalen L2-Cache-Speicher **428** verbunden. In einigen Ausführungsformen kann der Prozessor **100** einen integrierten L1-Cache-Speicher (nicht gezeigt) enthalten.

[0070] Die Busbrücke **402** stellt eine Schnittstelle zwischen dem Prozessor **100**, dem Hauptspeicher **404**, der Graphiksteuerung **408** und den mit den PCI-Bus **414** verbundenen Geräten bereit. Wenn eine Operation von einem der Geräte, die mit der Busbrücke **402** verbunden sind, empfangen wird, erkennt die Busbrücke **402** das Ziel der Operation (beispielsweise ein spezielles Gerät oder in diesem Falle der PCI-Bus **414**, dass das Ziel auf dem PCI-Bus **414** ist). Die Busbrücke **402** leitet die Operation an das Zielgerät weiter. Die Busbrücke **402** übersetzt im Allgemeinen eine Operation von einem von dem Quellengerät oder dem Bus benutzten Protokoll in das von dem Zielgerät oder Bus benutzten Protokoll.

[0071] Zusätzlich zur Bereitstellung einer Schnittstelle zu einem ISA/EISA-Bus für den PCI-Bus **414** kann die sekundäre Busbrücke **416** weitere Funktionen enthalten. Eine Eingabe/Ausgabe-Steuerung (nicht gezeigt), sowohl extern als auch in der sekundären Busbrücke **416** integriert, kann ebenso in dem Computersystem **400** vorgesehen sein, um eine funktionelle Unterstützung für eine Tastatur und eine Maus **422** und für diverse serielle und parallele Anschlüsse bereitzustellen. Eine externe Cache-Speichereinheit (nicht gezeigt) kann ebenso mit dem CPU-Bus **424** zwischen dem Prozessor **100** und der Busbrücke **402** in weiteren Ausführungsformen vorgesehen sein. Alternativ kann der externe Cache-Speicher mit der Busbrücke **402** verbunden sein, und eine Cache-Steuerlogik für den externen Cache-Speicher kann in der Busbrücke **402** integriert sein. Der L2 Cache-Speicher **428** ist in einer Rückseitenkonfiguration zu dem Prozessor **100** dargestellt. Zu beachten ist, dass der L2 Cache-Speicher **428** auch separat zu dem Prozessor **100** vorgesehen sein kann, integriert in einem Gehäuse (beispielsweise Schlitz 1 oder Schlitz A) in dem Prozessor **100** vorgesehen sein kann, oder dieser kann zusammen mit dem Prozessor **100** auf einem Halbleitersubstrat vorgesehen sein.

[0072] Der Hauptspeicher **200** ist ein Speicher, in welchem Anwendungsprogramme gespeichert sind und von dem aus der Prozessor **100** hauptsächlich arbeitet. Ein geeigneter Hauptspeicher **200** kann einen DRAM (dynamischer Speicher mit wahlfreiem Zugriff) umfassen. Beispielsweise sind eine Reihe von Bänken aus SDRAM (synchrone DRAM) oder Rambus DRAM (RDRAM) geeignet.

[0073] Die PCI-Geräte **412a** bis **412b** stellen anschaulich diverse periphere Geräte dar, etwa Netzwerkschnittstellenkarten, Videobeschleuniger, Audiokarten, Festplatten- oder Laufwerkstreiber oder Treibersteuerungen, SCSI- (Kleincomputersystemschnittstellen-) Adapter und Telefonkarten. In ähnlicher Weise ist das ISA-Gerät **418** stellvertretend für diverse Arten peripherer Geräte, etwa ein Modem, eine Klangkarte und diverse Datennahmekarten, etwa GPIB oder Feldbusschnittstellenkarten.

[0074] Die Graphiksteuerung **408** ist vorgesehen, um das Erzeugen von Text und Bildern auf einer Anzeige **426** zu steuern. Die Graphiksteuerung **408** kann einen typischen Graphikbeschleuniger darstellen, der allgemein bekannt ist, um damit dreidimensionale Datenstrukturen zu erzeugen, die in effizienter Weise in den Hauptspeicher **200** geschoben und daraus abgerufen werden können. Die Graphiksteuerung **408** kann daher ein Master des AGP-Busses **410** sein, in der Hinsicht, dass diese auf eine Zielschnittstelle innerhalb der Bussbrücke **402** zugreifen und von dieser anfordern kann, um damit Zugriff auf den Hauptspeicher **200** zu erhalten. Ein entsprechender Graphikbus erlaubt das rasche Zugreifen auf Daten aus dem Hauptspeicher **404**. Für gewisse Operationen kann die Graphiksteuerung **408** ferner ausgebildet sein, um PCI-Protokolltransaktionen auf dem AGP-Bus **410** zu erzeugen. Die AGP-Schnittstelle der Busbrücke **402** kann somit Funktionen aufweisen, um sowohl AGP-Protokolltransaktionen sowie auch PCI-Protokollziel- und Initiatortransaktionen zu unterstützen. Die Anzeige **426** ist eine beliebige elektronische Anzeige, auf der Bilder oder Text dargestellt werden können. Eine geeignete Anzeige **426** umfasst Kathodenstrahlröhre („CRT"), eine Flüssigkristallanzeige („LCD"), etc.

[0075] Es sollte beachtet werden, dass obwohl die AGP-, PCI- und ISA- oder EISA-Busse in der obigen Beschreibung als Beispiele aufgeführt sind, andere Busarchitekturen nach Bedarf eingesetzt werden können. Ferner ist zu beachten, dass das Computersystem **400** ein Multiverarbeitungscomputersystem mit zusätzlichen Prozessoren (beispielsweise dem Prozessor **100a**, wie er als optionale Komponente des Computersystems **400** gezeigt ist) enthalten kann. Der Prozessor **100a** kann ähnlich im Aufbau sein wie der Prozessor **100**. Genauer gesagt, der Prozessor **100a** kann in einer Ausführungsform eine identische Kopie des Prozessors **100** sein. Der Prozessor **100a** kann mit der Busbrücke **402** über einen unabhängigen Bus (wie in Fig. 6 gezeigt) verbunden sein oder kann den CPU-Bus **224** zusammen mit dem Prozessor **100** benutzen. Ferner kann der Prozessor **100a** mit einem optionalen L2 Cache-Speicher **428** ähnlich zu dem L2 Cache-Speicher **428** verbunden sein.

[0076] In Fig. 7 ist eine weitere Ausführungsform eines Computersystems **400** gezeigt. Andere Ausführungsformen sind möglich und hierin berücksichtigt. In der Ausführungsform aus Fig. 7 umfasst das Computersystem **400** diverse Verarbeitungsknoten **612a**,

612b, **612c** und **612d**. Jeder Verarbeitungsknoten ist mit einem entsprechenden Speicher **614a** bis **614d** über eine Speichersteuerung **616a** bis **616d**, die jeweils in dem entsprechenden Verarbeitungsknoten **612a** bis **612d** enthalten sind, verbunden. Des weiteren können die Verarbeitungsknoten **612a** bis **612d** eine Schnittstellenlogik aufweisen, die verwendet wird, um eine Kommunikation zwischen dem Verarbeitungsknoten **612a** bis **612d** herzustellen. Beispielsweise umfasst der Verarbeitungsknoten **612a** eine Schnittstellenlogik **618a** zur Kommunikation mit dem Verarbeitungsknoten **612b**, eine Schnittstellenlogik **618b** zur Kommunikation mit dem Verarbeitungsknoten **612c** und eine dritte Schnittstellenlogik **618c** zur Kommunikation mit einem noch weiteren Verarbeitungsknoten (nicht gezeigt). In ähnlicher Weise umfasst der Verarbeitungsknoten **612b** Schnittstellenlogiken **618d**, **618e** und **618f**; der Verarbeitungsknoten **612c** umfasst Schnittstellenlogiken **618g**, **618h** und **618i**; und der Verarbeitungsknoten **612d** umfasst Schnittstellenlogiken **618j**, **618k** und **618l**. Der Verarbeitungsknoten **612d** ist angeschlossen, um mit mehreren Eingabe/Ausgabe-Geräten (beispielsweise Geräte **620a** bis **620b** in einer prioritätsverketteten Konfiguration) über die Schnittstellenlogik **618l** zu kommunizieren. Andere Verarbeitungsknoten können mit anderen I/O-Geräten in ähnlicher Weise in Verbindung treten.

[0077] Die Verarbeitungsknoten **612a** bis **612d** implementieren eine paketbasierte Verbindung für die Kommunikation unter den Verarbeitungsknoten. In der vorliegenden Ausführungsform ist die Verbindung als ein Satz aus unidirektionalen Leitungen eingerichtet (beispielsweise Leitungen **624a** zum Übersenden von Paketen von dem Verarbeitungsknoten **612a** zu dem Verarbeitungsknoten **612b** verwendet, und Leitungen **624b** werden zum Übermitteln von Paketen von dem Verarbeitungsknoten **612b** zu dem Verarbeitungsknoten **612a** verwendet). Andere Leitungssätze **624c** bis **624h** werden zur Übermittlung von Paketen zwischen anderen Verarbeitungsknoten verwendet, wie dies in Fig. 7 gezeigt ist. Generell kann jeder Leitungssatz **614** eine oder mehrere Datenleitungen, eine oder mehrere Taktleitungen entsprechend den Datenleitungen und eine oder mehrere Steuerleitungen, die die Art der übertragenen Pakete kennzeichnen, enthalten. Die Verbindung kann in einer mit dem Cache-Speicher kohärenten Weise zur Kommunikation zwischen Verarbeitungsknoten oder in einer nicht kohärenten Weise zur Kommunikation zwischen einem Verarbeitungsknoten und einem I/O-Gerät (oder einer Busbrücke mit einem I/O-Bus konventionellen Aufbaus, etwa dem PCI-Bus oder dem ISA-Bus) betrieben werden. Des weiteren kann die Verbindung in einer nicht kohärenten Weise unter Anwendung einer prioritätsverketteten Struktur zwischen I/O-Geräten betrieben werden, wie dies gezeigt ist. Zu beachten ist, dass ein von einem Verarbeitungsknoten zu einem weiteren Knoten zu übertragendes Paket durch einen oder mehrere zwischengeschaltete Knoten laufen kann. Beispielsweise kann ein von dem Verarbeitungsknoten **612a** zu dem Verarbeitungsknoten **612d** übertragenes Paket durch den Verarbeitungsknoten **612b** oder den Verarbeitungsknoten **612c** laufen, wie in Fig. 7 gezeigt ist. Es kann ein beliebiger geeigneter Signalführungsalgorithmus verwendet werden. In anderen Ausführungsformen des Computersystem **400** können weitere oder weniger Verarbeitungsknoten als in der in Fig. 7 gezeigten Ausführungsform verwendet sein.

[0078] Im Allgemeinen können die Pakete als eine oder mehrere Bitzeiten auf den Leitungen **624** zwischen den Verarbeitungsknoten übertragen werden. Eine Bit-Zeit kann die ansteigende oder die abfallende Flanke des Taktsignals auf der entsprechenden Taktleitung sein. Die Pakete können Befehlspakete zur Initiierung von Transaktionen, Abfrage- bzw. Sondierungspakete zur Beibehaltung der Cache-Speicherkohärenz und Antwortpakete von Antworten auf Anfragen und Befehle enthalten.

[0079] Die Verarbeitungsknoten **612a** bis **612d** können zusätzlich zu einer Speichersteuerung und einer Schnittstellenlogik einen oder mehrere Prozessoren aufweisen. Allgemein gesagt, ein Verarbeitungsknoten umfasst mindestens einen Prozessor und kann optional eine Speichersteuerung zur Kommunikation mit einem Speicher oder anderen Logikers nach Bedarf enthalten. Genauer gesagt, kann jeder Verarbeitungsknoten **612a** bis **612d** eine oder mehrere Kopien des Prozessors **100** aufweisen. Die Verarbeitungsknoten **612** können jeweils einen Prozessor **100** enthalten, der eine Ausgabeeinheit **104**, einen Ablaufverfolgungscache-Speicher **160**, eine Abschlusswartereihe **102**, eine Neustartlogik **190**, und/oder einem Ablaufverfolgungsgenerator **170** aufweist, wie dies zuvor beschrieben ist. Eine externe Schnittstelleneinheit **18** kann die Schnittstellenlogik **618** in den Verarbeitungsknoten sowie die Speichersteuerung **616** aufweisen.

[0080] Die Speicher **614a** bis **614d** können beliebige Speichereinrichtungen enthalten. Beispielsweise kann der Speicher **614a** bis **614d** ein oder mehrere RAMBUS DRAM's (RDRAM's) synchrone DRAMS (SDRAM's), statische RAM's, etc. aufweisen. Der Adressenraum des Computersystems **400** ist unter den Speichern **614a** bis **614d** aufgeteilt. Jeder Verarbeitungsknoten **612a** bis **612d** kann eine Speicherzuordnung enthalten, die verwendet wird, um zu bestimmen, welche Adresse auf welche Speicher **614a** bis **614d** abgebildet wird, und somit zu welchem Verarbeitungsknoten **612a** bis **612d** eine Speicheranforderung für eine spezielle Adresse weitergeleitet werden soll. In einer Ausführungsform ist der Kohärenzpunkt für eine Adresse innerhalb des Computersystems **400** die Speichersteuerung **616a** bis **616d**, die

mit dem Speicher gekoppelt ist, der die der Adresse entsprechenden Bytes enthält. Anders ausgedrückt, die Speichersteuerung **616a** bis **616d** ist dafür zuständig, dass sichergestellt ist, dass jeder Speicherzugriff auf den entsprechenden Speicher **614a** bis **614d** in einer mit dem Cache-Speicher kohärenten Weise stattfindet. Die Speichersteuerungen **616a** bis **616d** können eine Steuerschaltung als Schnittstelle für die Speicher **614a** bis **614d** aufweisen. Des weiteren können die Speichersteuerungen **616a** bis **616d** Anforderungswartereihen zum Aneinanderreihen von Speicheranforderungen enthalten.

[0081] Die Schnittstellenlogiken **618a** bis **618l** können diverse Puffer zur Aufnahme von Paketen aus der Verbindung und zur Pufferung von Paketen, die über die Verbindung zu übertragen sind, aufweisen. Das Computersystem **400** kann einen beliebigen Ablaufsteuerungsmechanismus zum Übersenden von Paketen anwenden. Beispielsweise speichert in einer Ausführungsform jede Schnittstellenlogik **618** eine Zahl einer Gruppe jeder Art von Puffern innerhalb des Empfängers auf der anderen Seite der Verbindung, mit der diese Schnittstellenlogik verbunden ist. Die Schnittstellenlogik überträgt ein Paket nicht, sofern nicht die empfangende Schnittstellenlogik einen freien Puffer aufweist, um das Paket zu speichern. Wenn ein Empfangspuffer freigegeben wird, indem ein Paket weitergeleitet wird, übersendet die empfangende Schnittstellenlogik eine Nachricht an die sendende Schnittstellenlogik, um anzuzeigen, dass der Puffer nunmehr freigegeben ist. Ein derartiger Mechanismus wird als ein „Coupon-basiertes" System bezeichnet.

[0082] Die I/O-Geräte **620a** bis **620b** können beliebige geeignete I/O-Geräte sein. Beispielsweise können die I/O-Geräte **620a** bis **620b** Geräte zur Kommunikation mit einem anderen Computersystem sein, mit denen die Geräte verbunden sind (beispielsweise Netzwerkschnittstellenkarten oder Modems). Des weiteren können die I/o-Geräte **620a** bis **620b** Videobeschleuniger, Audiokarten, Festplatten oder Laufwerkstreiber oder Treibersteuerungen, SCSI-(Kleincomputersystemschnittstellen-) Adapter und Telefonkarten, Plankarten und diverse Datennahmekarten, etwa GPIB oder Feldbusschnittstellenkarten enthalten. Zu beachten ist, dass der Begriff „I/O-Gerät" und der Begriff „peripheres Gerät" hierin als Synonym bezeichnet sind.

[0083] Im hierin verwendeten Sinne bezeichnen die Begriffe „Taktzyklus" oder „Zyklus" ein Zeitintervall, in welchem diverse Phasen der Instruktionsverarbeitungspipelines ihre Aufgaben abschließen. Instruktionen und berechnete Werte werden durch Speicherelemente (etwa Register oder Arrays) auf der Grundlage eines Taktsignals, das den Taktzyklus definiert, empfangen. Beispielsweise kann eine Speicherelement einen Wert auf der Grundlage der ansteigenden oder abfallenden Flanke des Taktsignals erfassen.

[0084] Es sind diverse Variationen und Modifizierungen für den Fachmann auf der Grundlage und Würdigung dieser Offenbarung ersichtlich. Es ist daher beabsichtigt, dass die folgenden Patentansprüche so interpretiert werden, dass alle derartigen Variationen und Modifizierungen mit eingeschlossen sind.

Industrielle Anwendbarkeit

[0085] Diese Erfindung ist im Allgemeinen auf das Gebiet von Prozessoren anwendbar.

Zusammenfassung

[0086] Ein System umfasst einen Instruktionscache-Speicher (**106**), einen Ablaufverfolgungscache-Speicher (**160**) mit mehreren Ablaufverfolgungscache-Einträgen (**162**), und einen Ablaufverfolgungsgenerator (**170**), der mit dem Instruktionscache-Speicher (**106**) und dem Ablaufverfolgungscache-Speicher (**160**) verbunden ist. Der Ablaufverfolgungsgenerator (**170**) kann ausgebildet sein, eine Gruppe aus Instruktionen, die von dem Instruktionscache-Speicher (**106**) ausgegeben wird, zur Speicherung in einer der mehreren Ablaufverfolgungscache-Einträgen (**162**) zu empfangen. Der Ablaufverfolgungsgenerator (**170**) kann ausgebildet sein, eine Sonderinstruktion in der Gruppe aus Instruktionen zu erkennen und ein Speichern der Sonderinstruktion in einem gleichen der mehreren Ablaufverfolgungscache-Einträgen (**162**), der Nicht-Sonderninstruktionen enthält, zu verhindern.

Bezugszeichenliste

Fig. 1

104	Ausgabeeinheit
140	Decodiereinheit
106	Instruktionscache-Speicher
108	Vorabholeinheit
132	Verzweigungsvorhersageeinheit
118	Ablauforganisationseinheit(en)
102	Abschlusswartereihe
190	Neustartlogik
170	Ablaufverfolgungsgenerator
160	Ablaufverfolgungscache-Speicher von Vorabholeinheit **108**
116	Registerdatei
124	Ausführungskern(e)
126	Lade/Speichereinheit
130	Ergebnisbus
128	Datencache-Speicher
132	Verzweigungsvorhersageeinheit
200	Systemspeicher
100	Prozessor

Fig. 2

160 auf Verfolgungscache-Speicher
164 Markierung
165 Instruktionen
168 Ablaufsteuerung

Fig. 3a

164 Markierung
168 Ablaufsteuerung

Fig. 3b

Spalte 1
Vitalität
bedingungslos
vital
folgend auf erste Verzweigung
folgend auf zweite Verzweigung
bedingungslos „tot"
266 Codierung

Fig. 4

10 Empfangen von Instruktionen zur Speicherung in Ablaufverfolgungscache-Speicher
12 Irgendwelche Sonderinstruktionen?
14 Nicht speichern einer Sonderinstruktion in dem gleichen Ablaufverfolgungseintrag wie eine andere Instruktion
16 Speichern von Instruktionen in dem gleichen Ablaufverfolgungseintrag

Fig. 5

20 Abholen eines Instruktionsstromes aus dem Ablaufverfolgungscache-Speicher
22 Sonderinstruktion in dem Instruktionsstrom erkannt?
26 Instruktionsvervollständigung blockiert?
28 Zurücksetzen der Pipeline und Neustart der Prozessorausführung aus demInstruktionscache-Speicher
30 Zulassen, dass der Instruktionsstrom ausgeführt und abgeschlossen wird
24 vor der Ausführung der Sonderinstruktion, Zurücksetzen der Pipeline und Neustart
24 vor der Ausführung der Sonderinstruktion, Zurücksetzen der Pipeline und Neustart der Prozessorabarbeitung aus dem Instruktionscache-Speicher

Fig. 6

404 Hauptspeicher
428 L2 Cache-Speicher
10 Prozessor
242 CPU-Bus
406 Speicherbus
402 Busbrücke
410 AGP-Bus
408 Graphiksteuerung
426 Anzeige
10a Prozessor
428a L2 Cache-Speicher
412a, 412b PCI-Gerät
416 sekundäre Busbrücke
418 ISA-Gerät
420 ISA/EISA-Bus
422 Tastatur/Maus
424 PCI-Bus

Fig. 7

614a, 614b Speicher
612a, 612b Verarbeitungsknoten
620a, 620b Ein/Ausgabe-Gerät

Patentansprüche

1. System mit:
einem Instruktionscache-Speicher (**106**);
einem Ablaufverfolgungscache-Speicher (**160**) mit mehreren Ablaufverfolgungscache-Einträgen (**162**);
einem Ablaufverfolgungsgenerator (**170**), der mit dem Instruktionscache-Speicher (**106**) und dem Ablaufverfolgungscache-Speicher (**160**) verbunden ist;
wobei der Ablaufverfolgungsgenerator (**170**) ausgebildet ist, eine Gruppe aus Instruktionen, die von dem Instruktionscache-Speicher (**106**) ausgegeben wird zur Speicherung in einer der mehreren Ablaufverfolgungscache-Einträgen (**162**) zu empfangen, wobei der Ablaufverfolgungsgenerator (**170**) ausgebildet ist, eine Sonderinstruktion innerhalb der Gruppe aus Instruktionen zu erkennen und ein Speichern der Sonderinstruktion in einem gleichen der mehreren Ablaufverfolgungscache-Einträge (**162**) als eine Nicht-Sonderinstruktion zu verhindern.

2. System nach Anspruch 1, wobei der Ablaufverfolgungsgenerator (**170**) ausgebildet ist, Instruktionen in dem Ablaufverfolgungsspeicher (**160**) zumindest in teilweise decodierter Form zu speichern.

3. System nach Anspruch 2, das ferner eine Abschlusswartereihe (**102**) aufweist, die angeschlossen ist, um ausgeführte Instruktionen abzuschließen, wobei die Abschlusswartereihe (**102**) ausgebildet, das Abschließen einer aus dem Ablaufverfolgungscache-Speicher (**160**) abgeholten Instruktion zu verhindern, bis alle Instruktionen, die in dem gleichen Ablaufverfolgungscache-Eintrag (**162**) wie diese Instruktion enthalten sind, für das Abschließen bereit sind.

4. System nach Anspruch 2, das ferner eine Ausgabeeinheit (**104**) aufweist, die ausgebildet ist, von dem Instruktionscache-Speicher (**106**) empfangene Instruktionen auszugeben, wobei die Ausgabeeinheit

(**104**) ausgebildet, die Sonderinstruktion innerhalb einer Gruppe aus Instruktionen, die vom Instruktionscache-Speicher (**106**) empfangen werden, zu erkennen und eine Kennzeichnung der Sonderinstruktion an den Ablaufverfolgungsgenerator (**170**) auszugeben, wobei der Ablaufverfolgungsgenerator (**170**) ausgebildet ist, die Sonderinstruktion in Reaktion auf die Kennzeichnung von der Ausgabeeinheit (**104**) zu erkennen.

5. System nach Anspruch 2, wobei der Ablaufverfolgungsgenerator (**170**) ausgebildet ist, die Sonderinstruktion in einem anderen Ablaufverfolgungscache-Eintrag (**162**) als die eine oder mehreren anderen Instruktionen zu speichern, wobei der Ablaufverfolgungsgenerator (**170**) ausgebildet ist, andere Instruktionen mit der Sonderinstruktion in dem anderen Ablaufverfolgungscache-Eintrag (**162**) nicht zu speichern.

6. Verfahren mit:
Empfangen einer Gruppe aus Instruktionen für die Speicherung in einem Ablaufverfolgungscache-Eintrag (**162**) in einem Ablaufverfolgungscache-Speicher (**160**);
Speichern einer oder mehrerer Instruktionen, die in der Gruppe der Instruktionen enthalten sind, in dem Ablaufverfolgungscache-Eintrag (**162**);
Erkennen einer Sonderinstruktion innerhalb der Gruppe aus Instruktionen; und
in Reaktion auf das Erkennen, nicht Speichern der Sonderinstruktion in dem Ablaufverfolgungscache-Eintrag (**162**) mit einer der einen oder den mehreren Instruktionen.

7. Verfahren nach Anspruch 6, wobei das Speichern umfasst: Speichern der einen oder mehreren Instruktionen in dem Ablaufverfolgungscache-Eintrag (**162**) in zumindest teilweise decodierter Form.

8. Verfahren nach Anspruch 7, das ferner umfasst: Verhindern des Abschließens einer Instruktion, die aus dem Ablaufverfolgungscache-Speicher (**160**) abgerufen wurde, bis alle Instruktionen, die in dem gleichen Ablaufverfolgungscache-Eintrag (**162**) wie diese Instruktion enthalten sind, für das Abschließen bereit sind.

9. Verfahren nach Anspruch 8, das ferner umfasst:
Überwachen einer Anzahl an Zyklen, die seit dem Abschließen einer Instruktion verstrichen sind; und
wenn die Anzahl der Zyklen eine Schwellwertzahl übersteigt, Zurücksetzen einer Verarbeitungspipeline, die aus dem Ablaufverfolgungscache-Speicher (**160**) abgeholte Instruktionen ausführt, und Neustarten der Ausführung aus dem Instruktionscache-Speicher (**106**) heraus.

10. Computersystem (**400**) mit:
einem Systemspeicher (**404**); und
einem Prozessor (**10**), der mit dem Systemspeicher (**404**) verbunden ist;
dadurch gekennzeichnet, dass der Prozessor (**10**) umfasst:
einen Instruktionscache-Speicher (**106**);
einen Ablaufverfolgungscache-Speicher (**160**) mit mehreren Ablaufverfolgungscache-Einträgen (**162**);
einen Ablaufverfolgungsgenerator (**170**), der mit dem Instruktionscache-Speicher (**106**) und dem Ablaufverfolgungscache-Speicher (**160**) verbunden ist;
wobei der Ablaufverfolgungsgenerator (**170**) ausgebildet ist, eine Gruppe aus Instruktionen, die von dem Instruktionscache-Speicher (**106**) ausgegeben werden, für die Speicherung in einem der mehreren Ablaufverfolgungscache-Einträgen (**162**) zu empfangen, wobei der Ablaufverfolgungsgenerator (**170**) ausgebildet ist, eine Sonderinstruktion innerhalb der Gruppe aus Instruktionen zu erkennen und ein Speichern der Sonderinstruktion in einem der mehreren Ablaufverfolgungscache-Einträgen (**162**), der eine Nicht-Sonderinstruktion enthält oder erhalten soll, zu verhindern.

Es folgen 5 Blatt Zeichnungen

Anhängende Zeichnungen

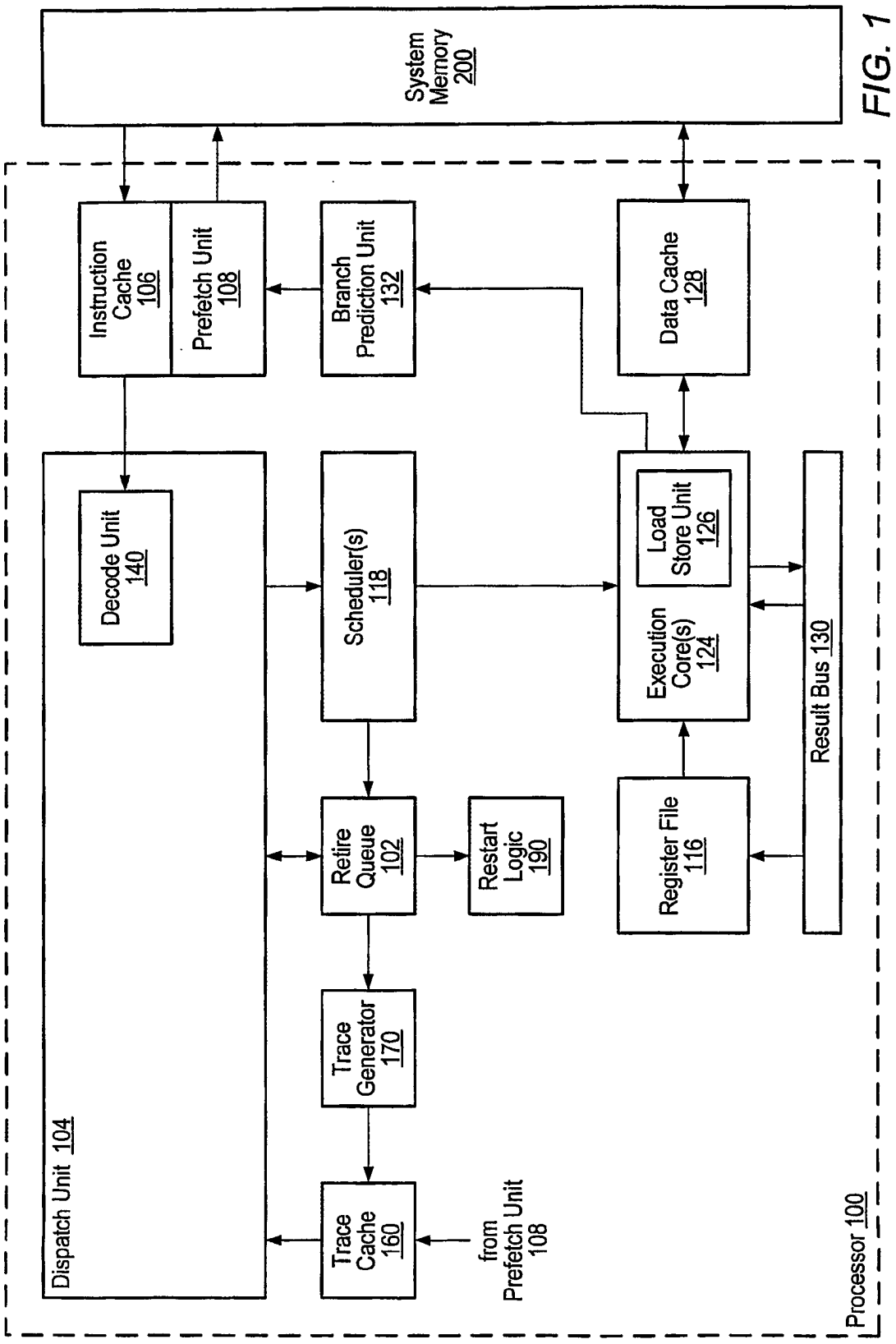


FIG. 1

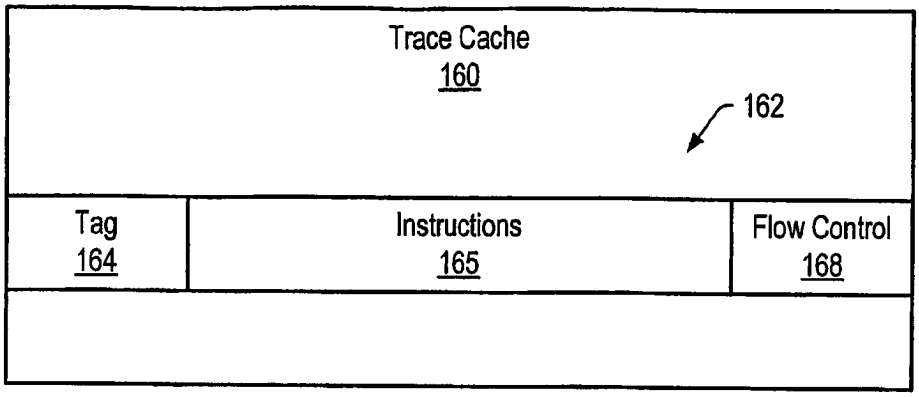


FIG. 2

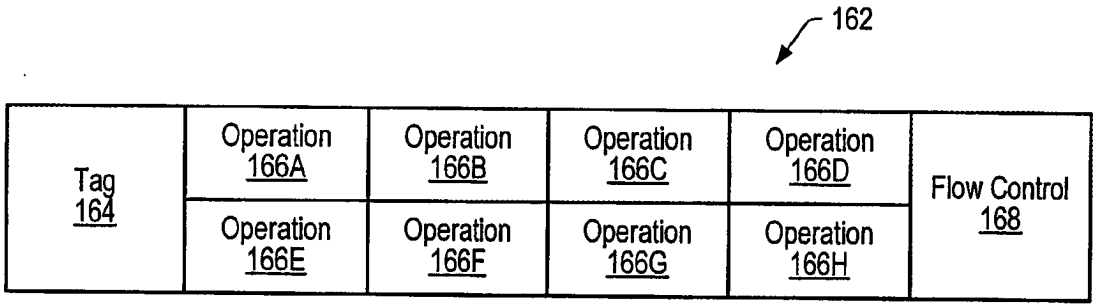


FIG. 3A

Liveness	Encoding 266
Unconditionally Live	11
Subsequent to 1st Branch	10
Subsequent to 2nd Branch	01
Unconditionally Dead	00

FIG. 3B

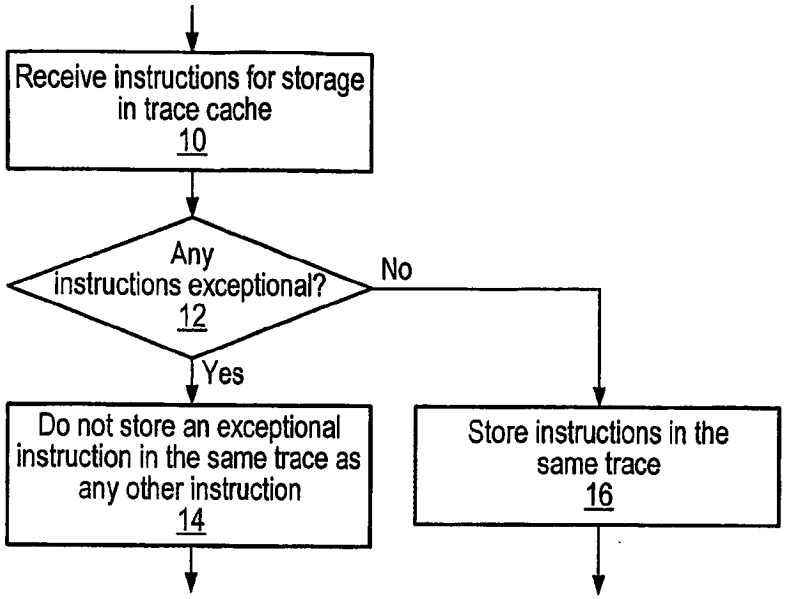

Receive instructions for storage in trace cache
10
Any instructions exceptional?
12
No
Yes
Do not store an exceptional instruction in the same trace as any other instruction
14
Store instructions in the same trace
16

FIG. 4

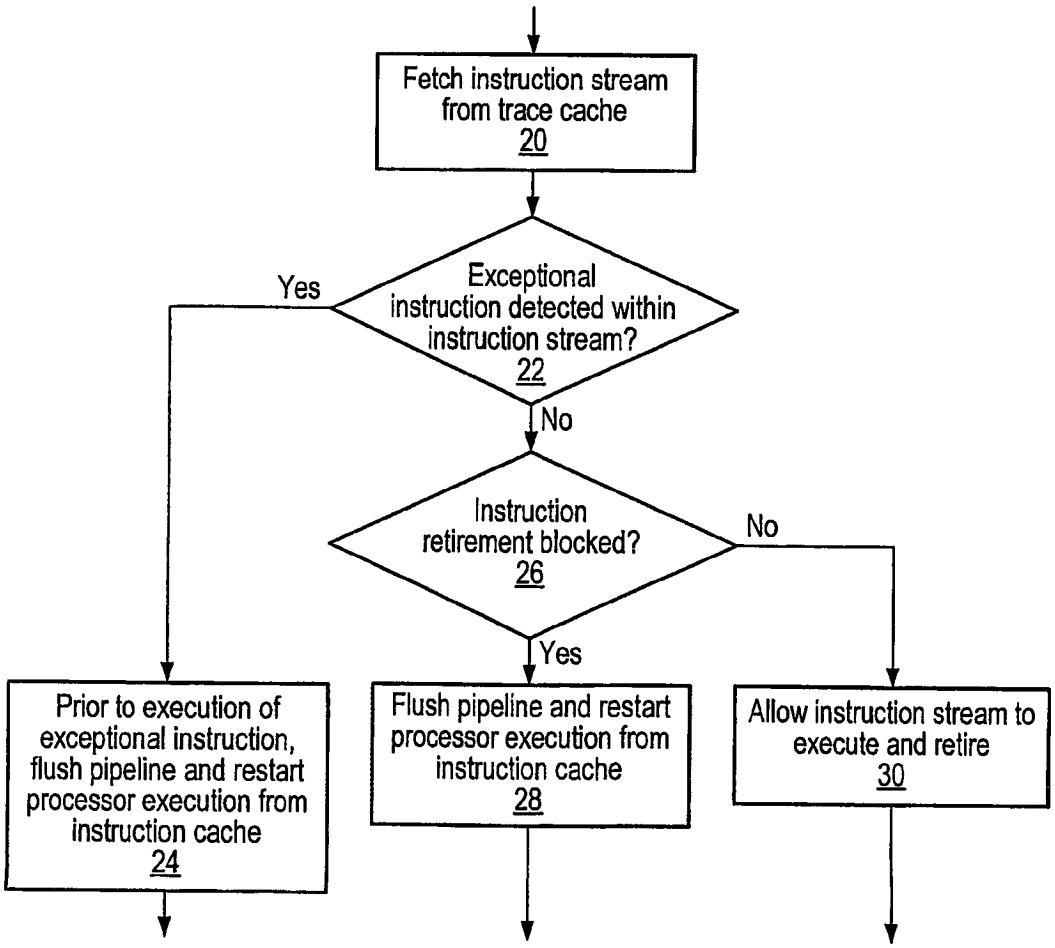

Fetch instruction stream from trace cache
20
Exceptional instruction detected within instruction stream?
22
Yes
No
Instruction retirement blocked?
26
No
Yes
Prior to execution of exceptional instruction, flush pipeline and restart processor execution from instruction cache
24
Flush pipeline and restart processor execution from instruction cache
28
Allow instruction stream to execute and retire
30

FIG. 5

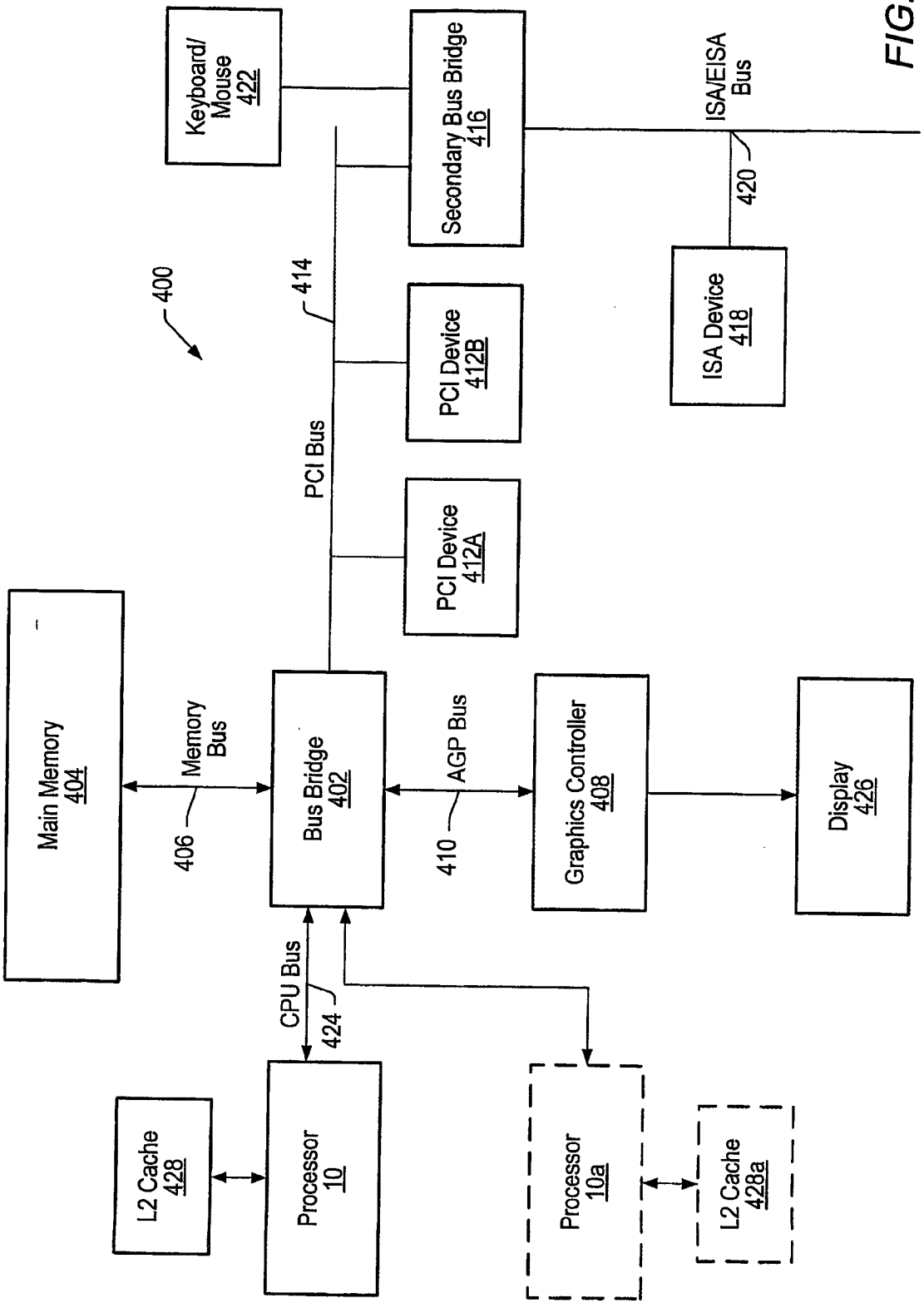

Keyboard/ Mouse 422
Secondary Bus Bridge 416
ISA/EISA Bus
420
ISA Device 418
400
414
PCI Device 412B
PCI Device 412A
PCI Bus
Main Memory 404
Memory Bus
406
Bus Bridge 402
AGP Bus
410
Graphics Controller 408
Display 426
CPU Bus
424
L2 Cache 428
Processor 10
Processor 10a
L2 Cache 428a

FIG. 6

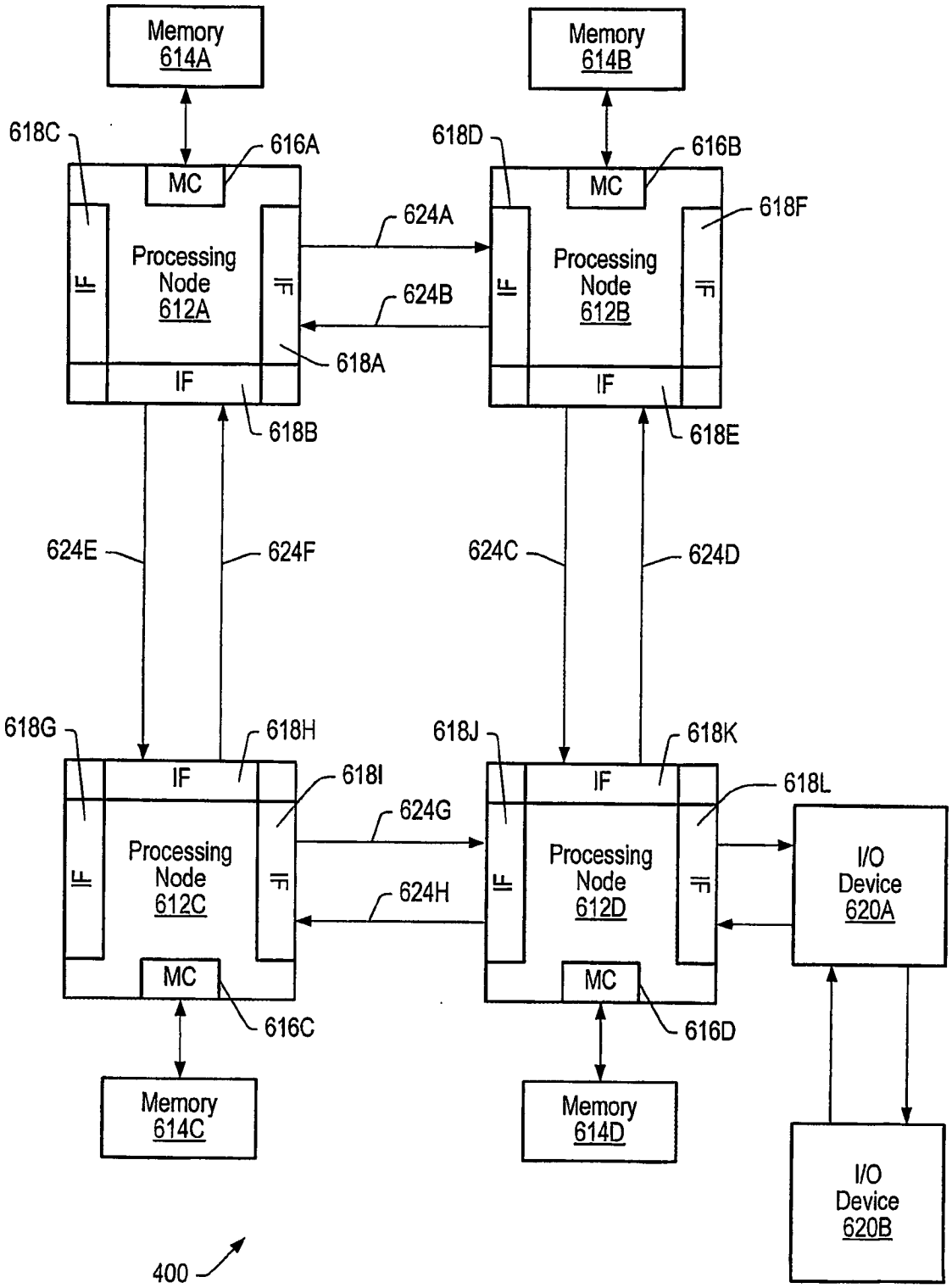

Memory
614A
Memory
614B
618C
616A
618D
616B
MC
MC
624A
618F
Processing
Node
612A
Processing
Node
612B
IF
IF
624B
618A
IF
IF
618B
618E
624E
624F
624C
624D
618G
618H
618J
618K
IF
IF
618I
618L
624G
Processing
Node
612C
Processing
Node
612D
I/O
Device
620A
624H
MC
MC
616C
616D
Memory
614C
Memory
614D
I/O
Device
620B
400

FIG. 7