(19) Deutsches Patent- und Markenamt

(10) **DE 199 83 793 B4** 2013.02.07

(12) # Patentschrift

(21) Deutsches Aktenzeichen: **199 83 793.7**
(86) PCT-Aktenzeichen: **PCT/US99/21249**
(87) PCT-Veröffentlichungs-Nr.: **WO 2000/034870**
(86) PCT-Anmeldetag: **22.09.1999**
(87) PCT-Veröffentlichungstag: **15.06.2000**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **29.11.2001**
(45) Veröffentlichungstag der Patenterteilung: **07.02.2013**

(51) Int Cl.: ***G06F 12/08*** (2006.01)

Innerhalb von drei Monaten nach Veröffentlichung der Patenterteilung kann nach § 59 Patentgesetz gegen das Patent Einspruch erhoben werden. Der Einspruch ist schriftlich zu erklären und zu begründen. Innerhalb der Einspruchsfrist ist eine Einspruchsgebühr in Höhe von 200 Euro zu entrichten(§ 6 Patentkostengesetz in Verbindung mit der Anlage zu § 2 Abs. 1 Patentkostengesetz).

(30) Unionspriorität:
09/207,278 **08.12.1998** **US**

(62) Teilung in:
199 84 043.1

(73) Patentinhaber:
Intel Corporation, Santa Clara, Calif., US

(74) Vertreter:
ZENZ Patent- und Rechtsanwälte, 45128, Essen, DE

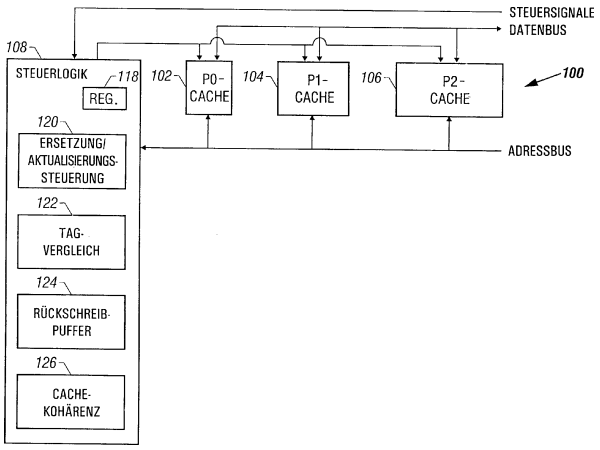
(72) Erfinder:
Cai, Zhong-ning, Portland, Oreg., US; Nakanishi, Tosaku, Saratoga, Calif., US

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:

GB	**2 311 880**	**A**
US	**4 928 239**	**A**
EP	**0 856 797**	**A1**

(54) Bezeichnung: **System mit einem Prozessor, auf dem mehrere, gleichzeitig aktive Ausführungsentitäten ausgeführt werden, und mit einem mehrere, den Ausführungsentitäten zugewiese Cache-Abschnitte aufweisenden Cache-Speicher**

(57) Hauptanspruch: System mit einem Prozessor (200), auf dem mehrere gleichzeitig aktive Ausführungsentitäten, beispielsweise Prozesse, Tasks oder Threads, ausgeführt werden können, einem mit dem Prozessor gekoppelten Speicher (206) und einem mit dem Prozessor gekoppelten Cache-Speicher (100; 202),
wobei der Cache-Speicher (100; 202) mehrere Cache-Abschnitte (102–106) und einen Cache-Controller (108) aufweist, wobei jeder Cache-Abschnitt mehrere Cache-Speicherplatze aufweist, wobei jeder Cache-Abschnitt (102–106) so ausgebildet ist, dass er Informationen speichert, die zu Anforderungen aus jeweils einer zugehörigen Ausführungsentität der mehreren Ausführungsentitäten gehören,
dadurch gekennzeichnet,
dass jeder Ausführungsentität ein Ausführungsentitätsidentifizierer (EID) zugeordnet ist,
dass in dem Speicher (206) mehrere Befehlsspeicherbereiche definiert sind, die jeweils zu einem Ausführungsentitätsidentifizierer (EID) gehören, so dass der Ausführungsentitätsidentifizierer (EID) anhand des Speicherbereichs, aus dem ein Befehl abgerufen wurde, bestimmt werden kann, und
dass der Cache-Controller auf der Basis des Ausführungsentitätsidentifizierers (EID) beim Zugreifen auf den Cache-Speicher (100; 202) einen zugehörigen Cache-Abschnitt (102–106) auswählt.



Beschreibung

[0001] Die Erfindung betrifft ein System nach dem Oberbegriff des Anspruchs 1.

[0002] Ein solches System ist beispielsweise aus den Druckschriften EP 0 856 797 A1 oder GB 2 311 880 A bekannt. Ein Beispiel für einen Pufferspeicher ist ein Cache-Speicher, der zwischen einem Prozessor und einem (üblicherweise relativ großen und langsamen) Systemspeicher angeordnet ist, um die von dem Prozessor für das Abfragen von Informationen aus dem Systemspeicher benötigte effektive Zugriffszeit zu verringern. In manchen Systemen kann ein Mehr-Ebenen-Cache-System verwendet werden, um die Leistungsfähigkeit weiter zu verbessern. Ein primärer Cache (L1-Cache) kann in den Prozessor selbst integriert werden und ein externer sekundärer, üblicherweise größerer Cache (L2-Cache) ist mit dem Prozessor gekoppelt. Eine Cache-Speicher-Anordnung ist z. B. in dem US-Patent Nr. 4, 928,239 beschrieben.

[0003] Ferner kann ein Cache-Speicher bei manchen herkömmlichen Speichersystemen eine separate Befehls-Cache-Einheit und eine separate Daten-Cache-Einheit enthalten, die eine zum Speichern von Befehlen und die andere zur Datenspeicherung. Während des Betriebs kann ein Prozessor Befehle vom Systemspeicher abrufen, um sie in der Befehls-Cache-Einheit zu speichern. Die von diesen Befehlen bearbeiteten Daten können in der Daten-Cache-Einheit gespeichert werden. Wenn von dem Prozessor angeforderte Informationen, beispielsweise Befehle oder Daten, bereits in dem Cache-Speicher gespeichert sind, dann spricht man davon, daß ein Cache-Speicher-Treffer aufgetreten ist. Ein Cache-Speicher-Treffer verringert die Zeit, die der Prozessor zum Zugreifen auf in dem Speicher gespeicherte Informationen benötigt, wodurch die Leistungsfähigkeit des Prozessors verbessert wird.

[0004] Wenn die von dem Prozessor benötigten Informationen nicht in dem Cache-Speicher gespeichert sind, dann spricht man davon, daß ein Cache-Fehlversuch aufgetreten ist. Wenn ein Cache-Fehlversuch auftritt, muß der Prozessor auf den Systemspeicher zugreifen, um die gewünschten Informationen abzurufen, wobei dies zu einer Leistungsverringerung bei der Zugriffszeit führt, während der Prozessor darauf wartet, daß der langsamere Systemspeicher die Anforderung beantwortet. Um die Anzahl der Cache-Fehlversuche zu verringern, wurden verschiedene Cache-Verwaltungsstrategien realisiert. Bspw. kann eines von mehreren Abbildungsschemata ausgewählt werden, bspw. ein direktes Abbildungsschema oder ein satz-assoziatives Cache-Abbildungsschema. Ein satz-assoziativer Cache-Speicher, welcher eine k-Wege-assoziative Abbildung (mapping) realisiert, z. B. eine 2-Wege-assoziative Abbildung, eine 4-Wege-assoziative Abbildung usw., ermöglicht im allgemeinen eine höhere Trefferquote als ein direkt abgebildeter Cache-Speicher. Ferner kann eine von verschiedenen Ersetzungsstrategien vorgesehen werden, um die Cache-Speicher-Trefferquote zu erhöhen, einschließlich einer FIFO-Strategie oder einer Am-längsten-unbenutzt(LRU)-Strategie. Ein weiteres konfigurierbares Merkmal eines Cache-Speichers ist die Cache-Speicher-Aktualisierungsstrategie, welche angibt, wie der Systemspeicher aktualisiert wird, wenn eine Schreiboperation den Inhalt des Cache-Speichers ändert. Zu den Aktualisierungsstrategien gehören eine Durchschreibstrategie und eine Rückschreibstrategie.

[0005] Üblicherweise kann ein System, wie z. B. ein Computer, mehrere Anwendungsprogramme und andere Softwareschichten enthalten, welche unterschiedliche Datenflußerfordernisse haben. Bspw. kann eine Programmausführungsentität, wie z. B. ein Prozeß, eine Task oder ein Thread, die einer Multimediaanwendung zugeordnet ist, große Blöcke von Daten (z. B. Videodaten) übertragen, die üblicherweise nicht wiederverwendet werden. Folglich kann das Zugreifen auf Daten dieser Art dazu führen, daß ein Cache mit großen Datenblöcken gefüllt wird, die wahrscheinlich nicht wieder verwendet werden.

[0006] Beim Füllen eines Cache-Speichers kann der Fall eintreten, daß die von einer Ausführungsentität verwendeten Daten von einer anderen Ausführungsentität verwendete Daten ersetzen, ein Phänomen, das als Daten-Cache-Verunreinigung (data cache pollution) bezeichnet wird. Eine von den Aktivitäten einer Ausführungsentität verursachte Daten-Cache-Verunreinigung kann die Wahrscheinlichkeit von Cache-Fehlversuchen für eine andere Ausführungsentität erhöhen und somit die Leistungsfähigkeit des Gesamtsystems verringern.

[0007] Somit wird eine Speicherarchitektur benötigt, welche die Speicherleistungsfähigkeit verbessern kann.

[0008] Diese Aufgabe wird erfindungsgemäß durch ein System mit den Merkmalen des Anspruchs 1 gelöst.

[0009] Vorteilhafte und bevorzugte Weiterbildungen der Erfindung sind in den Unteransprüchen gekennzeichnet.

[0010] Fig. 1 zeigt ein Blockschaltbild von Teilen eines Puffers oder eines Cache-Speichers mit mehreren Abschnitten gemäß einem Ausführungsbeispiel der Erfindung.

[0011] Fig. 2 zeigt ein Blockschaltbild eines Ausführungsbeispiels eines den Cache-Speicher gemäß Fig. 1 enthaltenden Systems.

[0012] Fig. 3 zeigt die Komponenten jedes Cache-Moduls in dem Cache-Speicher gemäß Fig. 1.

[0013] Fig. 4 zeigt ein Blockschaltbild eines Prozessors, der den Cache-Speicher gemäß Fig. 1 zusammen mit einer zugehörigen Steuerlogik enthält.

[0014] Fig. 5 zeigt ein Ablaufdiagramm einer in dem Prozessor gemäß Fig. 4 ausgeführten Befehlsausführungssequenz.

[0015] Fig. 6 zeigt ein Ablaufdiagramm eines Betriebssystems in dem System gemäß Fig. 2, welches den Cache-Speicher gemäß einem Ausführungsbeispiel aufbaut.

Detaillierte Beschreibung

[0016] In der folgenden Beschreibung sind zahlreiche Details angegeben, um ein besseres Verständnis der vorliegenden Erfindung zu ermöglichen. Jedoch ist für den Fachmann klar, daß die vorliegende Erfindung ohne diese Details realisiert werden kann und daß zahlreiche Abwandlungen oder Modifikationen der beschriebenen Ausführungsbeispiele möglich sind.

[0017] Einige Ausführungsbeispiele der Erfindung enthalten ein System mit einem Pufferspeicher, welcher in einer Ebene der Speicherhierarchie mehrere einzelne Pufferabschnitte enthält. Jeder Pufferabschnitt kann ein separates Puffermodul sein oder ein Teil eines Pufferspeichers, welcher separat adressierbar ist (d. h., der Speicher wird in verschiedene Adreßräume unterteilt). Die einzelnen Pufferabschnitte können getrennt konfigurierbar sein und können so zugewiesen werden, daß sie Informationen von verschiedenen Programmausführungsentitäten in dem System speichern. Ein derartiger Pufferspeicher kann als Mehr-Einheiten-Pufferspeicher bezeichnet werden.

[0018] Bei einigen Ausführungsbeispielen kann der Pufferspeicher einen Cache-Speicher enthalten, welcher in verschiedenen Anwendungen verwendet werden kann, z. B. in Prozessorsubsystemen, in Peripheriegerät-Steuereinrichtungen (wie in Video-Steuereinrichtungen, Festplattenlaufwerk-Steuereinrichtungen usw.) und in anderen Arten von Steuereinrichtungen. Zu den Systemen, welche derartige Cache-Speicher enthalten, können Mehrzweckcomputer oder Spezialcomputer, elektronische Hand-held-Einrichtungen (z. B. Telefone, Kalendersysteme, elektronische Spielvorrichtungen und dergleichen), Einrichtungen (appliances), Set-top-Boxen und andere elektronische Systeme gehören. Ein Cache-Speicher mit mehreren Cache-Abschnitten kann als Mehr-Einheiten-Cache-Speicher bezeichnet werden. Ein Cache-Speicher-Abschnitt kann ein separates Cache-Modul enthalten oder ein getrennt adressierbarer Teil des Cache-Speichers sein. Die folgenden beschriebenen Ausführungsbeispiele enthalten einen Computer mit einem Mehr-Einheiten-Cache-Speicher mit mehreren unabhängigen Cache-Modulen – es ist jedoch wichtig, daß andere Ausführungsbeispiele Computer enthalten können, die Mehr-Einheiten-Cache-Speicher mit anderen unabhängig konfigurierbaren Cache-Abschnitten aufweisen oder andere Systemarten mit Pufferspeichern.

[0019] Gemäß einigen Ausführungsbeispielen können die Attribute jedes einzelnen Cache-Moduls in einem Mehr-Einheiten-Cache-Speicher unabhängig konfiguriert werden. Zu derartigen Attributen können die Größe jedes Cache-Moduls, die Organisation (z. B. direkt abgebildet gegenüber Satz-assoziativer Abbildung), die Ersetzungsstrategie, die Aktualisierungsstrategie usw. zählen. Somit kann ein Cache-Modul beispielsweise als direkt abgebildeter Cache konfiguriert werden, und ein anderes Cache-Modul kann als k-Wege-Satz-assoziativer Cache konfiguriert werden. Die Cache-Module können ferner so konfiguriert werden, daß sie verschiedene Aktualisierungsstrategien haben, einschließlich einer Durchschreibstrategie oder einer Rückschreibstrategie. Andere Attribute können für die verschiedenen Cache-Module ebenfalls unterschiedlich eingestellt werden, wie im folgenden weiter unten beschrieben wird.

[0020] Einige Prozessoren können zur Verarbeitung Anforderungen von mehreren Ausführungsentitäten empfangen. Bei einem Prozessor kann es sich beispielsweise um einen Mehrzweckmikroprozessor oder einen Spezialmikroprozessor, um einen Mikrocontroller oder eine andere Art von Steuereinrichtung handeln, z. B. um Kundenwunschschaltkreise (ASICs), programmierbare Gatter-Arrays (PGAs) und dergleichen. Eine Programmausführungsentität gemäß einem Ausführungsbeispiel kann die Arbeitsgrundeinheit der in das System geladenen Software- und Firmwareschichten sein. Zu derartigen Arbeitsgrundeinheiten können Prozesse, Tasks, Threads oder andere Einheiten gehören, wie sie den verschiedenen Systemen entsprechend definierbar sind. Beispielsweise können bei manchen Betriebssystemen, wie bei bestimmten Windows®-Betriebssystemen der Microsoft Corporation, mehrere zu Prozessen gehörige Threads in dem System von dem Prozessor ausführbar sein, um verschiedene Operationen auszuführen. Ein anderes Betriebssystem, das die Möglichkeit des Multithreading oder des Multitasking bietet, ist das Be Operating System (BeOS) von BE, Inc., das in dem BE Operating System Product Data Sheet beschrieben ist, welches 1998 unter http://www.be.com veröffentlicht wurde.

[0021] Bei derartigen Betriebssystemen können mehrere Ausführungsentitäten, die zu verschiedenen Software und Firmwareschichten gehören, gleichzeitig aktiv sein. Anforderungen von diesen Aus-

führungsentitäten werden von dem Betriebssystem gemäß einem vorgegebenen Prioritätsprotokoll, z. B. umlaufend usw., geplant. Man bezeichnet derartige Betriebssysteme als Multitasking- bzw. Multithreading-Betriebssysteme. Um die Multitasking- bzw. Multithreading-Fähigkeiten eines Systems zu nutzen, können die unabhängigen Cache-Module eines Mehr-Einheiten-Cache-Speichers zur Speicherung von Informationen von zugehörigen Ausführungsentitäten zugewiesen werden. So können beispielsweise die Ausführungsentitäten einer Multimediaanwendung einem Cache-Modul zugewiesen werden, während die Ausführungsentitäten von anderen Anwendungen anderen Cache-Modulen des Mehr-Einheiten-Cache-Speichers zugewiesen werden können. Gemäß einem Ausführungsbeispiel können Anforderungen von jeder Ausführungsentität zu diesem Zweck unterschiedlichen Ausführungsentitätsidentifizierern (EIDs) zugewiesen werden. Somit können Anforderungen von Ausführungsentitäten einer ersten Anwendung einem EID zugewiesen werden, und Anforderungen von einer anderen Ausführungsentität können einem anderen EID zugewiesen werden. Folglich kann ein Cache-Modul gemäß diesem Ausführungsbeispiel für das allgemeine Datennutzungsverhalten einer zugewiesenen Anwendung konfiguriert werden. Bei einem anderen Ausführungsbeispiel können die von einer Software- oder Firmwareschicht erzeugten Ausführungsentitäten weiter aufgeteilt werden, so daß sie mehrere IDs haben. Beispielsweise kann eine Anwendung Ausführungsentitäten erzeugen, die Daten verschiedenen zeitlichen und räumlichen Lokalitätseigenschaften entsprechend verarbeiten. Beispielsweise kann die Wiederverwendung von Daten für einige Ausführungsentitäten wahrscheinlicher als für andere von der gleichen Anwendung erzeugte Ausführungsentitäten sein. Folglich kann es vorteilhaft sein, diese verschiedenen Ausführungsentitäten außerdem getrennt unterschiedlichen Cache-Modulen in dem Mehr-Einheiten-Cache-Speicher zuzuweisen. Folglich können bei einem alternativen Ausführungsbeispiel Anforderungen von verschiedenen Ausführungsentitäten einer Anwendung mehreren EIDs zugewiesen werden, so daß verschiedene Cache-Module verwendet werden können. Zusätzlich können Ausführungsentitäten von verschiedenen Anwendungen dem gleichen EID zugewiesen werden. Beispielsweise kann eine erste Ausführungsentität einer Multimediaanwendung dem EID 1 zugewiesen werden, während eine zweite Ausführungsentität der Multimediaanwendung dem EID 2 zugewiesen werden kann. In dem gleichen System können Ausführungsentitäten einer Spreadsheet-Anwendung, die ähnliche Datennutzungseigenschaften wie die zweite Ausführungsentität der Multimediaanwendung hat, ebenfalls dem EID 2 zugewiesen werden.

[0022] Bei anderen Ausführungsbeispielen können noch andere Schemata beim Zuweisen von EIDs an Anforderungen von Ausführungsentitäten realisiert werden. Anhand des zu einem Befehl gehörigen EIDs kann eine Cache-Steuereinrichtung für den Cache-Speicher verfolgen, welches Cache-Modul des Mehr-Einheiten-Cache-Speichers zum Speichern derjenigen Daten verwendet werden soll, auf die der Befehl zugreift. Folglich kann die Cache-Nutzung verbessert werden, da die einzelnen Cache-Module derart konfiguriert werden können, daß sie die Datennutzungseigenschaften der verschiedenen Ausführungsentitäten in dem System vorteilhaft berücksichtigen können. Beispielsweise wird eine Multimediaanwendung üblicherweise Anforderungen erzeugen, welche große Datenblöcke übertragen, die nicht wieder verwendet werden. Ein derartigen Anforderungen zugewiesenes Cache-Modul kann so konfiguriert werden, daß es die FIFO-Ersetzungsstrategie und die Durchschreibaktualisierungsstrategie realisiert. Anderen Anforderungstypen zugewiesene Cache-Module können andere Konfigurationen haben.

[0023] Wenn Ausführungsentitäten in einem System erzeugt werden, können diesen Ausführungsentitäten von einem Betriebssystem EID-Identifizierer zugewiesen werden. Es wird nun auf **Fig. 6** Bezug genommen. Wenn eine neue Ausführungsentität erfaßt wird (bei **502**), kann das Betriebssystem (bei **504**) gemäß einem Ausführungsbeispiel auf Konfigurationsinformationen zugreifen, die während der Systeminitialisierung geladen wurden, um die Art der Zuweisung von EID-Identifizierern zu bestimmen. Das Betriebssystem weist der Ausführungsentität als nächstes (bei **506**) den entsprechenden EID-Identifizierer für die Ausführungsentität zu. Beispielsweise kann das Betriebssystem in der Lage sein, drei EIDs zuzuweisen, die drei Cache-Modulen in einem Mehr-Einheiten-Cache-Speicher entsprechen. Ausführungsentitäten, die erste allgemeine Datennutzungseigenschaften haben, können einem ersten EID-Identifizierer zugewiesen werden, und Ausführungsentitäten, die zweite allgemeine Datennutzungseigenschaften haben, können einem zweiten EID-Identifizierer zugewiesen werden. Ein Standard-EID-Identifizierer kann denjenigen Ausführungsentitäten zugewiesen werden, die nicht speziell den beiden anderen EID-Identifizierern zugewiesen werden.

[0024] Zusätzlich weist das Betriebssystem auf der Basis der Konfigurationsinformationen (bei **508**) bestimmte Attribute jedes Cache-Moduls in dem Mehr-Einheiten-Cache-Speicher zu. Zu diesen Attributen können die Aktualisierungs-, Ersetzungs- und Plazierungsstrategien gehören. Das Betriebssystem kann außerdem die Attribute für das Standard-Cache-Modul des Mehr-Einheiten-Cache-Speichers zuweisen. Bei alternativen Ausführungsbeispielen können die EID-Identifizierer und die Cache-Attribute wie oben beschrieben von einer Softwareschicht zugewiesen werden, die von dem Betriebssystem getrennt ist.

[0025] Bei einem Ausführungsbeispiel können Ausführungsentitäten einer Multimediaanwendung, welche große Datenmengen übertragen und die Daten üblicherweise nicht wieder verwenden, einem EID-Identifizierer zugewiesen werden, so daß diese Daten in einem ersten Cache-Modul gespeichert werden, das für die Cache-Datennutzungseigenschaften dieser Ausführungsentitäten konfiguriert ist. Ausführungsentitäten von rechenintensiven Anwendungen, wie Komprimierungsanwendungen, können einem anderen EID-Identifizierer zugewiesen, werden, so daß die Daten in einem anderen Cache-Modul gespeichert werden, welches für Cache-Datenoperationen konfiguriert ist, die durch eine erhöhte räumliche Lokalität gekennzeichnet sind.

[0026] Bei einigen Ausführungsbeispielen kann ein Mehr-Einheiten-Cache-Speicher mit mehreren Cache-Modulen in einem Mehr-Ebenen-Cache-Speicher mit mehreren Cache-Speicher-Ebenen (z. B. einem L1-Cache oder einem L2-Cache realisiert werden. Ein derartiger Cache-Speicher kann als Mehr-Einheiten-Cache-Speicher mit mehreren Ebenen bezeichnet werden, bei dem wenigstens eine Ebene einen Mehr-Einheiten-Cache-Speicher enthält. Dementsprechend kann ein Mehr-Einheiten-Cache-Speicher mit mehreren Ebenen, der zwei Ebenen hat, beispielsweise in der folgenden Weise aufgebaut sein: Die erste Ebene ist ein Mehr-Einheiten-Cache und die zweite Ebene ist ein konventioneller Cache; die erste Ebene ist ein Mehr-Einheiten-Cache und die zweite Ebene ist ein Mehr-Einheiten-Cache; oder die erste Ebene ist ein konventioneller Cache und die zweite Ebene ist ein Mehr-Einheiten-Cache.

[0027] Die einzelnen Cache-Module eines Mehr-Einheiten-Cache-Speichers können als P-Cache-Speicher bezeichnet werden. Beispielsweise kann ein Mehr-Einheiten-Cache-Speicher somit verschiedene P-Cache-Speicher enthalten, einschließlich eines P0-Cache-Speichers, eines P1-Cache-Speichers, eines P2-Cache-Speichers usw. Die verschiedenen P-Cache-Speicher können als separate Speicherelemente oder Speichermodule realisiert werden, z. B. als mehrere statische Direktzugriffsspeicher(SRAM)-Einrichtungen oder als mehrere dynamische Direktzugriffsspeicher(DRAM)-Einrichtungen. Alternativ können mehrere P-Cache-Speicher in einer Speichereinrichtung implementiert werden, welche in getrennte Abschnitte unterteilt ist, die den verschiedenen P-Cache-Speichern, entsprechen. Zusätzlich kann der Mehr-Einheiten-Cache-Speicher in eine andere Einrichtung integriert sein, z. B. in einen Prozessor oder in eine andere Steuereinrichtung in einem System. Alternativ kann der Mehr-Einheiten-Cache-Speicher eine eigenständige Einheit sein, auf die von Steuereinrichtungen zugegriffen werden kann, um Cachegespeicherte Daten abzurufen. Bei anderen Ausführungsbeispielen kann ein Teil des Mehr-Einheiten-Cache-Speichers in einer integrierten Einrichtung angeordnet sein, während ein anderer Teil des Mehr-Einheiten-Cache-Speichers in einer anderen Einrichtung angeordnet ist.

[0028] Bei einigen Ausführungsbeispielen der Erfindung kann jedes einzelne P-Cache-Modul in einem Mehr-Einheiten-Cache-System verschiedene Attribute haben, einschließlich der Cache-Größe und der Organisation und der Cache-Aktualisierungs-, -Plazierungs- und -Ersetzungsstrategien. Für jeden P-Cache kann eine Plazierungsstrategie angegeben werden, um festzulegen, wie Informationen in ungefüllte Bereiche des Cache-Speichers zu plazieren sind. Eine Cache-Ersetzungsstrategie wird angegeben, um das Ersetzen von in jedem P-Cache gespeicherten Informationen zu bewältigen. Beispiele für Ersetzungsstrategien sind die FIFO-Strategie, die Amlängsten-unbenutzt(LRU)-Strategie und andere Arten der Ersetzungsstrategie. Eine Cache-Aktualisierungsstrategie bewältigt die Aktualisierung von Informationen, wenn eine an den Cache gerichtete Schreiboperation auftritt; hierzu kann eine Durchschreibstrategie oder eine Rückschreibstrategie gehören.

[0029] Es wird nun auf Fig. 1 Bezug genommen. Ein Mehr-Einheiten-Cache-Speicher **100** gemäß einem Ausführungsbeispiel enthält verschiedene P-Cache-Speicher, die als P0-Cache **102**, als P1-Cache **104** und als P2-Cache **106** dargestellt sind. Eine Cache-Steuereinrichtung **108** ist den P0-, P1- und P2-Cache-Speichern **102**, **104** und **106** zugeordnet. Bei einem Ausführungsbeispiel können mit jedem Cache-Modul **102**, **104** und **106** separate Adreß- und Datenbusse gekoppelt sein, so daß gleichzeitig auf die Cache-Module zugegriffen werden kann. Alternativ kann ein gemeinsamer Adreß- und Datenbus mit den Cache-Modulen gekoppelt sein. Die Cache-Steuereinrichtung **108** liefert für jedes der P-Cache-Module **102** bis **106** Steuersignale.

[0030] Die Cache-Steuereinrichtung **108** enthält Speicherelemente **118** in Form von Registern oder dergleichen, welche von dem Betriebssystem derart programmiert werden können, daß sie den zu jedem P-Cache-Speicher gehörenden EID-Identifizierer angeben. Beim Zugriff auf den Mehr-Einheiten-Cache-Speicher **100** wählt die Cache-Steuereinrichtung **108** einen der P-Cache-Speicher aus, und zwar auf der Basis eines Vergleiches zwischen dem von einer Anforderung zur Verfügung gestellten EID und der in den Speicherelementen **118** gespeicherten EID-Werten.

[0031] Die Cache-Steuereinrichtung **108** enthält ferner einen Ersetzungs- und Aktualisierungssteuerblock **120**, um die Ersetzungs- und Aktualisierungsstrategien der drei verschiedenen Cache-Module so zu steuern, wie sie von in den Speicherelementen **118** programmierten Steuerinformationen festgelegt

sind. So können die Speicherelemente **118** beispielsweise derart programmiert werden, daß sie für einen P-Cache eine FIFO-Ersetzungsstrategie und für einen anderen P-Cache eine LRU-Ersetzungsstrategie angeben.

[0032] Die Cache-Steuereinrichtung **108** kann ferner einen Tag-Vergleichs-Block **122** aufweisen, der das Tag einer eingehenden Anforderung mit dem in dem ausgewählten P-Cache oder den ausgewählten P-Cache-Speichern gespeicherten Tag vergleicht, um festzustellen, ob ein Cache-Treffer aufgetreten ist. Wenn eine Aktualisierung des Hauptspeichers **206** (Fig. 2) erforderlich ist, speichert ein Rückschreibpuffer **124** ferner die Cache-Zeile eines der P-Cache-Speicher, um sie an den Hauptspeicher **206** oder an einen L2-Cache **204** (Fig. 2) zu übertragen.

[0033] Zum Sicherstellen der Cache-Datenintegrität enthält die Cache-Steuereinrichtung **108** ferner einen Cache-Kohärenz-Block **126**, der bestimmt, ob ein Platz eines Cache-Moduls, auf den zugegriffen wurde, gültig ist. Bei einem Ausführungsbeispiel kann jedes Cache-Modul ein Gültig/Ungültig-Bit speichern. Alternativ kann ein ausgeklügelteres Kohärenzprotokoll implementiert werden, wie z. B. das Modifiziert-, Exklusiv-, Geteilt(shared)- und Ungültig(invalid)-(MESI)Protokoll.

[0034] Zu den weiteren Steuersignalen, die der Cache-Steuereinrichtung **108** zur Verfügung gestellt werden können, können ein Cache-Sperr(disable)-(CD)-Signal und ein Cache-Flush(CF)-Signal gehören. Darüber hinaus können der Cache-Steuereinrichtung **108** andere Cache-bezogene Signale wie Snoop-Signale zur Verfügung gestellt werden.

[0035] Es wird nun auf Fig. 2 Bezug genommen. Das Mehr-Einheiten-Cache-System **100** kann an vielen verschiedenen Stellen in einem System **10** realisiert werden (z. B. in einem Prozessorsubsystem, in Brücken-Steuereinrichtungen, in Peripheriegerät-Steuereinrichtungen, in Speicher-Steuereinrichtungen und dergleichen). Bei einem Ausführungsbeispiel enthält das System **10** einen Computer, obwohl das System **10** bei anderen Ausführungsbeispielen irgendeine andere elektronische Einrichtung sein kann, in der ein Cache-Speicher oder ein Pufferspeicher implementiert werden kann.

[0036] Das System **10** enthält eine zentrale Verarbeitungseinheit (CPU) **200**, die einen Prozessor oder eine geeignete Steuereinrichtung enthalten kann und einen Cache-Speicher mit einer oder mehreren Ebenen hat. Wie dargestellt ist, kann die CPU **200** beispielsweise einen internen Cache-Speicher enthalten, bei dem es sich um einen primären (L1)-Cache **202** handelt. Zusätzlich kann die CPU **200** über einen Host-Bus **203** gekoppelt sein, um auf einen externen Cache zuzugreifen, welcher der sekundäre (L2)-Cache **204** ist. Der L1-Cache **202** kann eine Codekomponente (zum Speichern von Befehlen) und eine Datenkomponente (zur Datenspeicherung) enthalten. Ähnlich kann der L2-Cache **204** Code- und Datenkomponenten enthalten. Somit werden von dem Hauptspeicher **206** abgerufene Befehle und Daten in den Code- bzw. Datenkomponenten des L1- oder L2-Cache-Speichers **202** oder **204** gespeichert. Bei anderen Ausführungsbeispielen sind keine separaten Code- und Daten-Cache-Komponenten vorgesehen.

[0037] Bei einigen Ausführungsbeispielen kann der Mehr-Einheiten-Cache-Speicher **100** (Fig. 1) in dem L1-Cache **202**, dem L2-Cache **204** oder in beiden implementiert sein. Hier sei zur Erläuterung angenommen, daß der Mehr-Einheiten-Cache-Speicher **100** gemäß Fig. 1 in dem L1-Cache **202** realisiert ist, das ist der interne Cache der CPU **200**. Wichtig ist jedoch, daß der beschriebene Mehr-Einheiten-Cache-Speicher oder Modifikationen eines solchen Cache-Speichers in dem L2-Cache **204** oder in anderen Steuereinrichtungen in dem System realisiert werden können, beispielsweise in einer Video-Steuereinrichtung oder in einer Festplattenlaufwerks-Steuereinrichtung. Zusätzlich bildet der Mehr-Einheiten-Cache-Speicher **100** bei diesem Ausführungsbeispiel die Daten-Cache-Komponente des L1-Cache-Speichers **202**.

[0038] Der Hauptspeicher **206** wird von einer Speichersteuereinrichtung **207** in einem Speicher-Hub **208** gesteuert, der mit der CPU **200** über den Host-Bus **203** gekoppelt ist. Ferner kann der Speicher-Hub **208** eine Cache-Steuereinrichtung **205** enthalten, die mit dem L2-Cache **204** wirksam gekoppelt ist. Der Speicher-Hub **208** kann ferner eine Grafikschnittstelle **211** enthalten, die über eine Verbindung **209** mit einer Grafik-Steuereinrichtung **210** gekoppelt ist, welche wiederum mit einer Anzeige **212** gekoppelt ist. Die Grafikschnittstelle kann beispielsweise dem Accelerated Graphics Port (A. G. P.) Interface Specification, Revision 2.0, veröffentlicht im Mai 1998, entsprechen.

[0039] Der Speicher-Hub **208** kann ferner mit einem Eingabe/Ausgabe(I/O)-Hub **214** gekoppelt sein, der Brückensteuereinrichtungen **215** und **223** enthält, die mit einem Systembus **216** bzw. mit einem sekundären Bus **224** gekoppelt sind. Der Systembus kann beispielsweise ein Peripheriekomponenten (PCI)-Bus sein, wie er in der PCI Local Bus Specification, Production Version, Revision 2.1, veröffentlicht im Juni 1995, definiert ist. Der Systembus **216** kann mit einer Speichersteuereinrichtung **218** gekoppelt sein, die den Zugriff auf eine oder mehrere Massenspeichereinrichtungen **220** steuert, beispielsweise ein Festplattenlaufwerk, ein CD-Laufwerk oder ein DVD-Laufwerk. Bei einem alternativen Ausführungsbeispiel kann die Speichersteuereinrichtung **218** in den I/O-Hub **214** integriert sein, sowie andere Steuer-

funktionen. Der Systembus **216** kann ferner mit anderen Komponenten gekoppelt sein, wie beispielsweise mit einer Netzwerksteuereinrichtung **222**, die mit einem (nicht gezeigten) Netzwerkport gekoppelt ist.

[0040] Über den sekundären Bus **224** können zusätzliche Einrichtungen gekoppelt werden, wie ein nicht-flüchtiger Speicher **228**, der Einschaltroutinen, wie BIOS-Routinen, speichern kann. Der sekundäre Bus **224** kann ferner Ports zur Koppelung mit Peripherieeinrichtungen enthalten. Obwohl in der Beschreibung auf spezielle Konfigurationen und Architekturen der verschiedenen Schichten des Systems **10** Bezug genommen wird, sollte klar sein, daß zahlreiche Abwandlungen und Modifikationen der beschriebenen und dargestellten Ausführungsbeispielen möglich sind. Beispielsweise können statt des Speicher- und I/O-Hubs eine Host-Brücken-Steuereinrichtung und eine System-Brücken-Steuereinrichtung gleichwertige Funktionen ermöglichen, wobei die Host-Brücken-Steuereinrichtung zwischen der CPU **100** und dem Systembus **216** angeordnet wird und die System-Brücken-Steuereinrichtung **224** zwischen dem Systembus **216** und dem sekundären Bus **224** angeordnet wird. Zusätzlich kann eine beliebige Anzahl von Busprotokollen implementiert werden.

[0041] Zahlreiche verschiedene Programmausführungsentitäten sind von der CPU **200** in dem System **10** ausführbar. Wie dargestellt ist, werden gemäß einem Ausführungsbeispiel mehrere Prozesse **252**, **254** und **256** unter einem Betriebssystem **250**, welches beispielsweise ein Windows®-Betriebssystem sein kann, geladen. Jeder Prozeß kann eine oder mehrere Ausführungsentitäten erzeugen, welche die Arbeitsgrundeinheiten in dem System bilden. Bei einem Beispiel können die Ausführungsentitäten Threads sein; wie in Fig. 2 dargestellt ist, kann der Prozeß **252** die Threads **258** und **260** enthalten, der Prozeß **254** kann einen Thread **262** enthalten und der Prozeß **256** kann die Threads **264** und **266** enthalten.

[0042] Verschiedene (beispielsweise aus Modulen, Routinen oder anderen Schichten bestehende) Software oder Firmware, z. B. Anwendungen, Betriebssystemmodule oder Routinen, Gerätetreiber, BIOS-Module oder Routinen und Interrupt-Behandler, können in einem oder mehreren Speichermedien in dem System gespeichert werden oder auf andere Weise faßbar verkörpert sein. Zu den Speichermedien, die für eine faßbare Verkörperung von Software- und Firmware-Befehlen geeignet sind, können verschiedene Speicherformen gehören, einschließlich Halbleiterspeichereinrichtungen, wie dynamische oder statische Direktzugriffsspeicher, löschbare und programmierbare Nur-Lese-Speicher (EPROMs), elektrisch löschbare und programmierbare Nur-Lese-Speicher (EEPROMs) und Flash-Speicher; Magnetplatten wie Festplatten, Disketten und Wechselplatten; andere Magnetmedien, einschließlich Bänder; und optische Medien wie CD- oder DVD-Platten. Die in dem Speichermedium gespeicherten Befehle veranlassen das System **10** bei ihrer Ausführung, programmierte Aktionen auszuführen.

[0043] Die Software oder Firmware kann auf eine beliebige von vielen verschiedenen Weisen in das System **10** geladen werden. Beispielsweise können Befehle oder andere Codesegmente, die auf einem Speichermedium gespeichert sind oder über eine Netzwerkschnittstellenkarte, ein Modem oder einen anderen Schnittstellenmechanismus transportiert werden, in das System **10** geladen und zur Ausführung von programmierten Aktionen ausgeführt werden. Beim Laden oder beim Transportprozeß können Datensignale, die in Form von Trägerwellen vorliegen (die über Telefonleitungen, Netzwerkleitungen, drahtlose Verbindungen, Kabel und dergleichen übertragen werden) die Befehle oder Codesegmente an das System **10** übermitteln.

[0044] Die Ausführungsentitäten (in diesem Fall Threads) können verschiedene Operationen ausführen. Beispielsweise kann ein Spreadsheet-Prozeß einen ersten Thread erzeugen, um Rechnungen an von einem Benutzer eingegebenen Einträgen auszuführen, und einen zweiten Thread, um die berechneten Daten in den Hauptspeicher **206** zu übertragen. Jeder Thread oder jede Ausführungsentität kann Anforderungen erzeugen, welche als Befehle im Hauptspeicher **206** gespeichert werden. Diese Befehle werden von der CPU **200** zur Ausführung vom Hauptspeicher **206** abgerufen.

[0045] Erfindungsgemäß wird jeder in dem System **10** ablaufenden Ausführungsentität ein Ausführungsentitäts (execution entity)-Identifizierer (EID) zugeordnet. Der EID jeder Ausführungsentität kann von dem Betriebssystem zugewiesen werden. Wenn ein Einplaner bzw. Scheduler **270** den Ablauf von Anforderungen von den Ausführungsentitäten zur Verarbeitung durch die CPU **200** steuert, könnte der zugehörige EID jeder Ausführungsentität zusammen mit einem oder mehreren entsprechenden Befehlen gespeichert werden. Dabei ruft die CPU **200** die zugehörigen EIDs zusammen mit den Befehlen ab.

[0046] Erfindungsgemäß werden die EIDs im Speicher **206** jedoch nicht zusammen mit Befehlen gespeichert. Statt dessen werden in dem Speicher **206** mehrere den verschiedenen EIDs entsprechende Befehlsspeicherbereiche definiert. Zu einer Anforderung von einer Ausführungsentität mit einem ersten EID gehörende Befehle können in einem ersten Befehlsspeicherbereich gespeichert werden; zu einer Anforderung von einer Ausführungsentität mit einem zweiten EID gehörende Befehle können in einem zweiten Befehlsspeicherbereich gespeichert werden; usw. Dabei ruft die CPU **200** Befehle vom Speicher **206** ohne zugehörige EIDs ab. Jedoch kann die CPU

200 anhand des Befehlsspeicherbereichs, von dem der Befehl abgerufen wird, den EID des Befehls bestimmen.

[0047] Wenn die EIDs nicht zusammen mit Befehlen gespeichert werden, kann die CPU **200** mehreren verschiedenen Threads zugewiesene Mikrosequenzen enthalten. Somit kann ein Mikrosequenzer die zu einem Thread gehörenden Befehle abrufen und ein anderer Mikrosequenzer die zu einem anderen Thread gehörenden Befehle usw. Jeder Mikrosequenzer kann so konfiguriert werden, daß er die Plätze von Befehlen der zugehörigen Ausführungsentitäten kennt. Bei diesem Ausführungsbeispiel kann der EID eines Befehls anhand des Mikrosequenzers bestimmt werden, der den Befehl abgerufen hat. Der vorgegebene Befehl kann dann innerhalb der CPU gespeichert werden.

[0048] Der abgerufene oder ermittelte EID wird dann von der Cache-Steuereinrichtung **108** oder von einer anderen geeigneten Decodiereinrichtung decodiert, um den zu verwendenden P-Cache zu kennzeichnen, wenn der Befehl einen Zugriff auf Daten anfordert. Die Cache-Steuereinrichtung **108** greift auf einen der P-Cache-Speicher zu, um von dem zugehörigen Befehl verarbeitete Daten abzurufen oder zu speichern. Bei der beispielhaften Konfiguration gemäß Fig. 1 können zu Befehlen mit EID 0 gehörende Daten in dem P0-Cache **102**, zu Befehlen mit EID 1 gehörende Daten in dem P1-Cache **104** und zu Befehlen mit EID 2 gehörende Daten in dem P2-Cache **106** gespeichert werden. Bei manchen Ausführungsbeispielen kann ein P-Cache zu mehreren EID gehören. Ferner können Ausführungsentitäten von verschiedenen Anwendungs- und Softwareschichten dem gleichen EID zugewiesen werden.

[0049] Es wird nun auf Fig. 3 Bezug genommen, in der die allgemeine Architektur eines P-Cache-Speichers dargestellt ist. Bei dem in Fig. 3 dargestellten Beispiel ist ein 4-Wege-Satz-assoziativer Cache dargestellt. Andere Konfigurationen sind ebenfalls möglich, einschließlich eines direkt abgebildeten Cache-Speichers oder anderer k-Wege-Satz-assoziativer Cache-Speicher. Jeder P-Cache kann ein Status-Array **160**, ein Tag-Array **162** und ein Daten-Array **164** enthalten. Wie dargestellt ist, sind das Status-Array **160**, das Tag-Array **162** und das Daten-Array **164** für die 4-Wege-Satz-assoziative Organisation jeweils in vier verschiedene Abschnitte unterteilt.

[0050] Das Status-Array **160** kann eines oder mehrere der folgenden Felder enthalten: Einen EID-Identifizierer; Ersetzungsauswahlbits (replacement selection bits – RPS), die von dem Ersetzungs- und Aktualisierungssteuerblock **120** zum Ersetzen einer Cache-Zeile verwendet werden; und Cache-Kohärenz-Protokollbits. Beispielsweise kann jeder Block des P-Cache-Moduls mit einem Gültig/Ungültig-Bit verbunden sein, das anzeigt, ob der zugehörige Cache-Speicherplatz gültig oder ungültig ist. Alternativ kann das Status-Array **160** MESI-Bits speichern. Die Ersetzungsauswahlbits RPS können verwendet werden, um anzuzeigen, welche Cache-Zeile ersetzt werden soll. Die RPS-Bits können beispielsweise zum Verfolgen der am längsten unbenutzten Cache-Zeile (für die LRU-Ersetzung) oder der als erstes eingegebenen Zeile (für die FIFO-Ersetzung) verwendet werden.

[0051] Die Cache-Steuereinrichtung **108** kann als integrierte Einheit oder mit Hilfe von verschiedenen separaten Steuereinheiten implementiert werden. Wie erörtert wurde, wird beim Abruf eines Befehls für die Ausführung der zu dem Befehl gehörende EID wiedergewonnen. Auf der Basis des EID-Wertes wird das richtige P-Cache-Modul ausgewählt, um Daten wiederzugewinnen oder Daten hineinzuschreiben. Je nachdem, ob eine gültige Kopie der zugehörigen Daten in dem ausgewählten P-Cache-Modul gespeichert ist, kann mit einem Treffer oder mit einem Fehlversuch geantwortet werden.

[0052] Ein Mehr-Einheiten-Cache-System mit unabhängig voneinander konfigurierbaren Cache-Modulen gemäß einigen Ausführungsbeispielen kann einen oder mehrere der folgenden Vorteile haben. Es kann eine größere Cache-Management-Flexibilität erreicht werden, da die Plazierungs-, Ersetzungs- und Aktualisierungsstrategie und die Cache-Größe und die Organisation jedes der P-Cache-Module so eingestellt werden können, daß die Cache-Nutzung für die entsprechenden Ausführungsentitäten verbessert wird. Die Leistungsfähigkeit des Cache-Speichers kann dadurch erhöht werden, daß die Cache-Module so konfiguriert werden, daß sie die verschiedenen Cache-Nutzungseigenschaften (hinsichtlich der Daten- oder Befehlsspeicherung) der verschiedenen Ausführungsentitäten vorteilhaft nutzen. Die Daten-Cache-Verunreinigung durch die verschiedenen aktiven Ausführungsentitäten in dem System **10** kann verringert werden, wobei dies die Cache-Trefferquote verbessern kann. Ferner bietet das Mehr-Einheiten-Daten-Cache-System eine große Zugriffsbandbreite, in dem es die Parallelität für einen Multithreading- oder Multitasking-Prozessor erhöht, da auf die P-Cache-Module gleichzeitig zugegriffen werden kann. Derartige gleichzeitige Zugriffe auf einen Daten-Cache-Speicher können dazu beitragen, die Daten-Cache-Latenzzeit zu verringern und die Datenzugriffsbandbreitenanforderungen von Hochleistungsprozessoren zu erfüllen.

[0053] Bei einem anderen Ausführungsbeispiel können Compiler für verschiedene Anwendungsprogramme die Attribute des Mehr-Einheiten-Cache-Speichers dynamisch neukonfigurieren, um die Leistungsfähigkeit des Cache-Speichers weiter zu erhöhen. Beispielsweise können während des Betriebs

zu den verschiedenen Ausführungsentitäten gehörende statistische Informationen gesammelt und gespeichert werden. Die Attribute jedes P-Cache-Moduls können in Abhängigkeit von den gesammelten statistischen Informationen verändert werden. Wenn beispielsweise festgestellt wird, daß die FIFO-Ersetzungsstrategie für ein bestimmtes P-Cache-Modul nicht effizient ist, kann der Cache-Steuereinrichtung **108** mitgeteilt werden, daß sie die Ersetzungsstrategie in die LRU-Strategie, oder eine andere Ersetzungsstrategie ändern soll. Dieses alternative Ausführungsbeispiel bietet die Flexibilität, daß die Konfiguration von einzelnen P-Cache-Modulen in Abhängigkeit von der Art der Ausführung der Ausführungsentitäten in dem System **10** dynamisch geändert werden kann.

[0054] Es wird nun auf Fig. 4 Bezug genommen. Bei einem Ausführungsbeispiel enthält die CPU **200** den Mehr-Einheiten-L1-Cache-Speicher **202** und eine zugehörige Logik. Der Mehr-Einheiten-L1-Cache-Speicher enthält drei Daten-Cache-Module: den P0-Cache **102**, den P1-Cache **104** und den P2-Cache **106**. Der P0-Cache **102** kann als Standard-Daten-Cache bezeichnet werden, welcher zur Speicherung von Daten verwendet wird, die zu Ausführungsentitäten gehören, welche nicht speziell einem der anderen P-Speicher in dem L1-Cache zugewiesen wurden. Beispielsweise kann solchen Ausführungsentitäten von dem Betriebssystem eine Standard EID-0 zugewiesen werden. Die P1- und P2-Cache-Speicher **104** und **106** können zur Speicherung von Daten für Anforderungen von Ausführungsentitäten mit EID 1 bzw. 2 zugewiesen werden. Bei einem anderen Ausführungsbeispiel kann der P0-Cache ein größerer Speicher als der P1- und der P2-Cache sein, da es der Standard-Daten-Cache ist.

[0055] Andere gemäß einer beispielhaften Konfiguration enthaltende Komponenten der CPU **200** sind in Fig. 4 dargestellt. Eine Busvordereinheit (BFU-bus front unit) **404** bildet die Schnittstelle zur Vorderseite oder zum Host-Bus **203**. Die BFU **404** kann Adreßtreiber und -empfänger, Schreibpuffer, Datenbus-Sender-Empfänger, eine Bus-Master-Steuerung und eine Paritätserzeugung und -steuerung enthalten.

[0056] Im folgenden wird zunächst der Befehlspfad beschrieben. Von der BFU **404** von dem Hauptspeicher **206** oder von dem L2-Cache **204** abgerufene Befehle können in einem Befehls-Cache **406** gespeichert werden, der Teil des L1-Cache-Speicher **206** ist. Der interne Befehls-Cache **406** kann Kopien der am häufigsten verwendeten Befehle enthalten. Gemäß einigen Ausführungsbeispielen werden Befehle entweder von dem Hauptspeicher **206** oder von dem L2-Cache **204** abgerufen und in dem Befehls-Cache **406** gespeichert. Eine Einheit **408** mit einem Befehlspuffer und einer Decodierlogik decodiert einen ausgewählten Befehl von dem Befehls-Cache **406** und erzeugt eine oder mehrere Mikro-Operationen zusammen mit den zugehörigen EIDs.

[0057] Erfindungsgemäß werden die Befehle den verschiedenen EIDs entsprechend in verschiedenen Befehlsspeicherbereichen des Speichers **206** gespeichert. Jedoch werden die EIDs nicht zusammen mit den Befehlen gespeichert. Wenn die CPU **200** einen Befehl abruft, wird der zugehörige EID nicht abgerufen. Statt dessen bestimmt die CPU **200** den EID des abgerufenen Befehls auf der Basis des Adreßspeicherplatzes, an dem der Befehl gespeichert ist. Dies kann beispielsweise von der Decodierlogik **408** ausgeführt werden. Somit wird der EID eines Befehls auf der Basis des Befehlsspeicherbereiches, aus dem der Befehl abgerufen wird, bestimmt. Sobald der EID von der CPU **200** bestimmt wurde, kann er an die Mikrooperationen angegliedert werden und in der Befehlswarteschlange **412** gespeichert werden.

[0058] Bei einem weiteren Ausführungsbeispiel, bei dem EIDs nicht zusammen mit Befehlen im Speicher gespeichert werden, können mehrere Programmzähler und Mikrosequenzer in die CPU **200** integriert sein, welche entsprechenden Threads zugewiesen sind. Dieses Ausführungsbeispiel wird weiter unten beschrieben.

[0059] Der Ausgangsport der Befehlspuffer- und Dekodierlogik **408** kann mit einer Befehlswarteschlange **412** gekoppelt sein, welche die Mikrooperationen zusammen mit zugehörigen EIDs speichert. Der Ausgangsport der Befehlswarteschlange **412** wird an einen Sequenzer **414** geleitet. Der Sequenzer **414** kann mehrere den verschiedenen EIDs entsprechende Mikrosequenzereinheiten **430**, **432** und **434** enthalten. Die Mikrosequenzereinheit **430** kann beispielsweise so konfiguriert werden, daß sie zu dem EID 0 gehörende Mikrooperationen verarbeitet, der Mikrosequenzer **432** kann so konfiguriert werden, daß er zu dem EID 1 gehörende Mikrooperationen verarbeitet, und der Mikrosequenzer **434** kann so konfiguriert werden, daß er zu dem EID 2 gehörende Mikrooperationen verarbeitet. Die von den Mikrosequenzern **430**, **432** und **434** verarbeiteten Mikrooperationen werden von der Befehlswarteschlange **412** empfangen. Gemäß einem Ausführungsbeispiel können die Mikrosequenzer **430**, **432** und **434** gleichzeitig operieren, um zu verschiedenen EIDs gehörende Mikrooperationen zu verarbeiten. Die Operation der Mikrosequenzer **430**, **432** und **434** wird von einer Steuerlogik **436** in dem Sequenzer **414** gesteuert.

[0060] Eine Ausführungsentität enthält üblicherweise eine Anzahl von Befehlen, die gemäß einer Programmreihenfolge ausgeführt werden. Standardmäßig werden Befehlsadressen einfach erhöht, um den nächsten Befehl abzurufen. Tritt ein Sprung oder eine bedingte Verzweigung auf, dann ist eine Zieladresse

als Adresse des nächsten Befehls angegeben. Auf diese weise ist die Adresse des Speicherplatzes bekannt, an dem der nächste Befehl gespeichert ist. Ein Programmzähler kann zum Verfolgen der Programmreihenfolge der Befehle verwendet werden. Ein Mikrosequenzer arbeitet mit dem Programmzähler zusammen, um die Befehle auszuführen. Der Mikrosequenzer kann zum Abrufen eines Befehls eine beispielsweise in der BFU **404** angeordnete Abrufeinheit damit beauftragen, einen Befehl mit einer in dem Programmzähler gespeicherten Adresse abzurufen. Auf diese Weise können abgerufene Befehle als zu einer Ausführungsentität gehörend identifiziert werden, da der Mikrosequenzer die Adresse des nächsten Befehls bereits (von dem Programmzähler) kennt.

[0061] Somit können für ein System mit verschiedenen Threads beispielsweise zwei oder mehr unabhängige Programmzähler verwendet werden. Beispielsweise können drei Programmzähler PC0, PC1 und PC2 zu den Mikrosequenzern **430**, **432** bzw. **434** gehören. Das Betriebssystem kann die Anfangszustände der Programmzähler PC0, PC1 und PC2 laden, so daß die Programmzähler zu den verschiedenen Threads gehörende Befehle abrufen können. Der PC0 verfolgt zusammen mit dem Mikrosequenzer **430** die Programmsequenz für einen ersten Thread, der PC1 verfolgt zusammen mit dem Mikrosequenzer **432** die Programmsequenz für einen zweiten Thread usw. Wenn ein Befehl abgerufen wird, auf den vom PC0 gezeigt wird, weiß die CPU **200**, daß der Befehl zu einem ersten Thread beispielsweise mit dem EID 0 gehört. Der EID wird dann in der CPU **200** an den Befehl angegliedert und an später decodierte Mikrooperationen, welche zur Ausführung durch den Mikrosequenzer **430**, **432** oder **434** in dem Sequenzer **414** in der Befehlswarteschlange **412** gespeichert werden.

[0062] Wie in Fig. 4 dargestellt ist, wird der Ausgangsport des Sequenzers **414** einem Pipeline-Back-end-Block **415** zur Verfügung gestellt, welcher verschiedene Funktionseinheiten enthält, beispielsweise eine Frühverzweigungsausführungseinheit (early branch execution unit-BR) **416**, eine schnelle Dekodiereinheit (fast decoder unit-FD) **418**, eine arithmetische Logikeinheit (ALU-arithmetic/logic unit) **420** und eine Adreßgeneratoreinheit (AGU) **422**. während der Ausführung einer oder mehrerer Mikrooperationen durch den Sequenzer **414** kann auf diese Funktionseinheiten zur Ausführung angeforderter Funktionen zugegriffen werden.

[0063] Der Pipeline-Back-end-Block **415** enthält außerdem Registersätze **424**, **426** und **428**. Die Registersätze **424**, **426** und **428** in der CPU **200** entsprechen den drei EID-Gruppen EID 0, EID 1 und EID 2. Die Registersätze **424**, **426** und **428** können jeweils Steuerregister, Statusregister, Flag-Register und Mehrzweckregister enthalten. während des Betriebs werden die Registersätze **424**, **426** und **428** von den Funktionseinheiten in dem Pipeline-Back-end-Block **415** aktualisiert. Gemäß einem Ausführungsbeispiel kann auf die Registersätze **424**, **426** und **428** ferner unabhängig und gleichzeitig zugegriffen werden.

[0064] Bei dem dargestellten Ausführungsbeispiel können zu verschiedenen EIDs gehörende Anforderungen gleichzeitig verarbeitet werden, wenn es keine Abhängigkeiten zwischen den Anforderungen gibt und die verschiedenen Anforderungen außerdem nicht die gleichen Funktionseinheiten **416**, **418**, **420** und **422** verwenden müssen. Bei einer gleichzeitigen Operation der Mikrosequenzer **430**, **432** und **434** kann auf die Registersätze **424**, **426** und **428** sowie auf die Cache-Module in dem Mehr-Einheiten-Cache-Speicher gleichzeitig zugegriffen werden und diese gleichzeitig aktualisiert werden.

[0065] In dem Datenpfad der CPU **200** speichern ein Speicherpuffer **450** (für Schreiboperationen) und ein Ladepuffer **452** (für Leseoperationen) Daten, die von der BFU **404** abgerufen werden oder für diese bestimmt sind. Der Speicherpuffer **450** und der Ladepuffer **452** sind mit einem internen Datenbus **454** gekoppelt, der mit verschiedenen Einheiten gekoppelt ist, einschließlich des P0-Cache-Speichers **102**, des P1-Cache-Speichers **104**, des P2-Cache-Speichers **106**, des Pipeline-Back-end-Blockes **415** und eines assoziativen Übersetzungspufferspeichers (TLB – translation look aside buffer) **456**.

[0066] Adressen von Befehlen in dem Befehls-Cache **406** werden an den TLB **456** geliefert, bei dem es sich im wesentlichen um einen Hochgeschwindigkeitsspeicher in der CPU **200** handelt, der die virtuelle Adresse von dem Befehls-Cache **406** in eine physikalische Adresse übersetzt, die den Zugriff auf die Daten-Cache-Module **102**, **104** und **106** ermöglicht.

[0067] Da der Mehr-Einheiten-Daten-Cache verfügbar ist, kann die Steuerlogik **436** in dem Mikrocodesequenzer **440** einen geeigneten Befehl zur Verarbeitung durch einen der Mikrosequenzer **430**, **432** oder **434** aussuchen. Wenn ein Datenzugriff benötigt wird, können die Mikrosequenzer **430**, **432** und **434** gleichzeitig auf die verschiedenen Module in dem Mehr-Einheiten-Daten-Cache zugreifen.

[0068] Somit können zur Verbesserung der Systemleistung mehrere Befehle in der CPU **200** ausgeführt werden, die gleichzeitig auf Daten in dem L1-Mehr-Einheiten-Cache **202** zugreifen können.

[0069] Bei manchen Ausführungsbeispielen kann sich die Steuerlogik **436** des Sequenzers **414** auch mit eventuellen Lade-/Speicher-Ordnungsvorgängen, mit dem Neufüllen des Cache-Speichers mit ausstehenden Daten und anderen Dingen befassen.

Beispielsweise können bei einem Ausführungsbeispiel zu einer Anforderung mit einer hohen Trefferquote gehörende Befehle als erstes eingeplant werden, genauso wie Befehle einer Ausführungsentität mit Echtzeitzwang (real time constrained execution entity), die eine hohe Priorität haben.

[0070] Es wird nun auf Fig. 5 Bezug genommen, in der der allgemeine Ablauf einer Befehlsausführungssequenz gemäß einem Ausführungsbeispiel dargestellt ist. Befehle werden von der CPU **200** über den Host-Bus **203** vom Hauptspeicher **206** oder vom L2-Cache **204** abgerufen (bei **302**). Bei einem Ausführungsbeispiel werden mit den Befehlen zugehörige EIDs abgerufen. Bei einem anderen Ausführungsbeispiel werden die zugehörigen EIDs nicht gespeichert und folglich nicht abgerufen. Von der Decodierstufe **408** werden die abgerufenen Befehle dann in interne Mikrooperationen übersetzt (bei **304**), wobei jeder Mikrooperation ein entsprechender EID angegliedert wird. Der EID kann derjenige sein, der mit dem Befehl abgerufen wurde, oder er kann von der CPU **200** anhand des Adreßspeicherplatzes des Befehls bestimmt werden oder anhand desjenigen Mikrosequenzers, von dem der Befehl abgerufen wurde. Als nächstes wird die übersetzte Mikrooperation in der Befehlswarteschlange **412** gespeichert (bei **306**). Dann wird die Mikrooperation zur Ausführung an einen der Mikrosequenzer **430**, **432** und **434** geliefert (bei **308**). Die Ausführung einer Mikrooperation kann dazu führen, daß eine Daten-Cache-Zugriffsanforderung durchgeführt wird (bei **310**). In diesem Fall wird auf das entsprechende P-Cache-Modul auf der Basis des angegliederten EIDs zugegriffen. Der EID wird von der Cache-Steuereinrichtung **108** decodiert und eine geeignete Anforderung wird an einen entsprechenden P-Cache (**102**, **104** oder **106**) gesendet. Die Datenzugriffsanforderung wird dann in dem zugewiesenen P-Cache abgeschlossen (bei **312**).

[0071] Während die beschriebenen Ausführungsbeispiele einen Mehr-Einheiten-Cache-Speicher zur Datenspeicherung enthalten, ist es klar, daß der Mehr-Einheiten-Cache-Speicher bei anderen Ausführungsbeispielen zur Speicherung von Befehlen von verschiedenen Ausführungsentitäten geeignet sein kann. Bei solchen Ausführungsbeispielen enthalten die in dem Mehr-Einheiten-Cache-Speicher gespeicherten Informationen die Befehle selbst.

Patentansprüche

1. System mit einem Prozessor (**200**), auf dem mehrere gleichzeitig aktive Ausführungsentitäten, beispielsweise Prozesse, Tasks oder Threads, ausgeführt werden können, einem mit dem Prozessor gekoppelten Speicher (**206**) und einem mit dem Prozessor gekoppelten Cache-Speicher (**100**; **202**),
wobei der Cache-Speicher (**100**; **202**) mehrere Cache-Abschnitte (**102**–**106**) und einen Cache-Controller (**108**) aufweist, wobei jeder Cache-Abschnitt mehrere Cache-Speicherplatze aufweist, wobei jeder Cache-Abschnitt (**102**–**106**) so ausgebildet ist, dass er Informationen speichert, die zu Anforderungen aus jeweils einer zugehörigen Ausführungsentität der mehreren Ausführungsentitäten gehören,
dadurch gekennzeichnet,
dass jeder Ausführungsentität ein Ausführungsentitätsidentifizierer (EID) zugeordnet ist,
dass in dem Speicher (**206**) mehrere Befehlsspeicherbereiche definiert sind, die jeweils zu einem Ausführungsentitätsidentifizierer (EID) gehören, so dass der Ausführungsentitätsidentifizierer (EID) anhand des Speicherbereichs, aus dem ein Befehl abgerufen wurde, bestimmt werden kann, und
dass der Cache-Controller auf der Basis des Ausführungsentitätsidentifizierers (EID) beim Zugreifen auf den Cache-Speicher (**100**; **202**) einen zugehörigen Cache-Abschnitt (**102**–**106**) auswählt.

2. System nach Anspruch 1, dadurch gekennzeichnet, daß die Cache-Abschnitte so konfigurierbar sind, daß sie eine verschiedene Cache-Zeilen-Ersetzungsstrategie, Cache-Aktualisierungsstrategie und/oder Cache-Organisation haben.

3. System nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß der Cache-Speicher ein Mehr-Ebenen-Cache-Speicher ist, bei dem wenigstens eine Ebene einen Mehr-Einheiten-Cache-Speicher mit mehreren Cache-Abschnitten enthält.

4. System nach Anspruch 1, dadurch gekennzeichnet, daß jeder Cache-Abschnitt auf der Basis der jeweiligen zeitlichen und räumlichen Lokalitätseigenschaften, mit denen die Ausführungsentitäten Daten verarbeiten konfiguriert ist.

Es folgen 5 Blatt Zeichnungen

Anhängende Zeichnungen

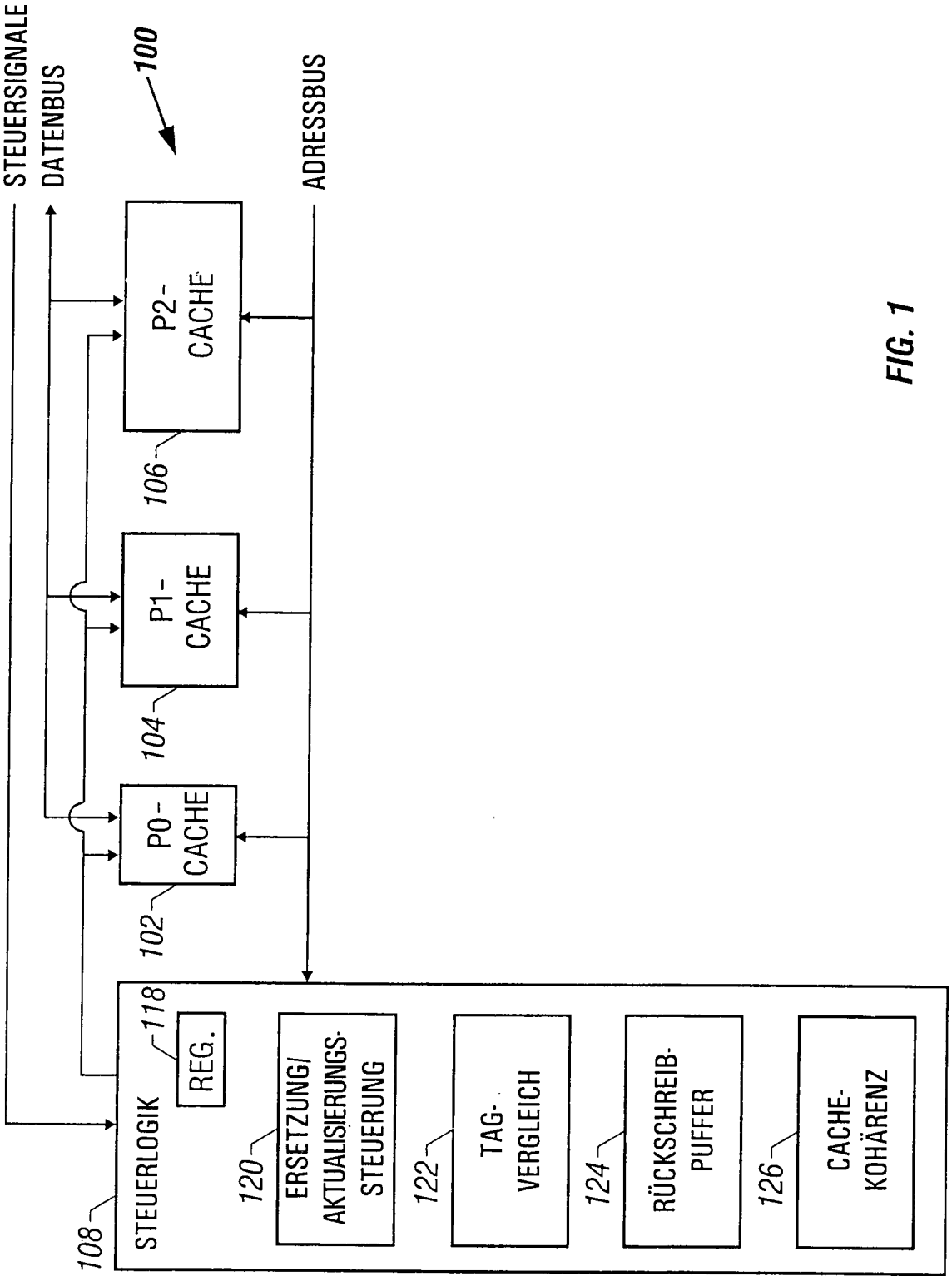


FIG. 1

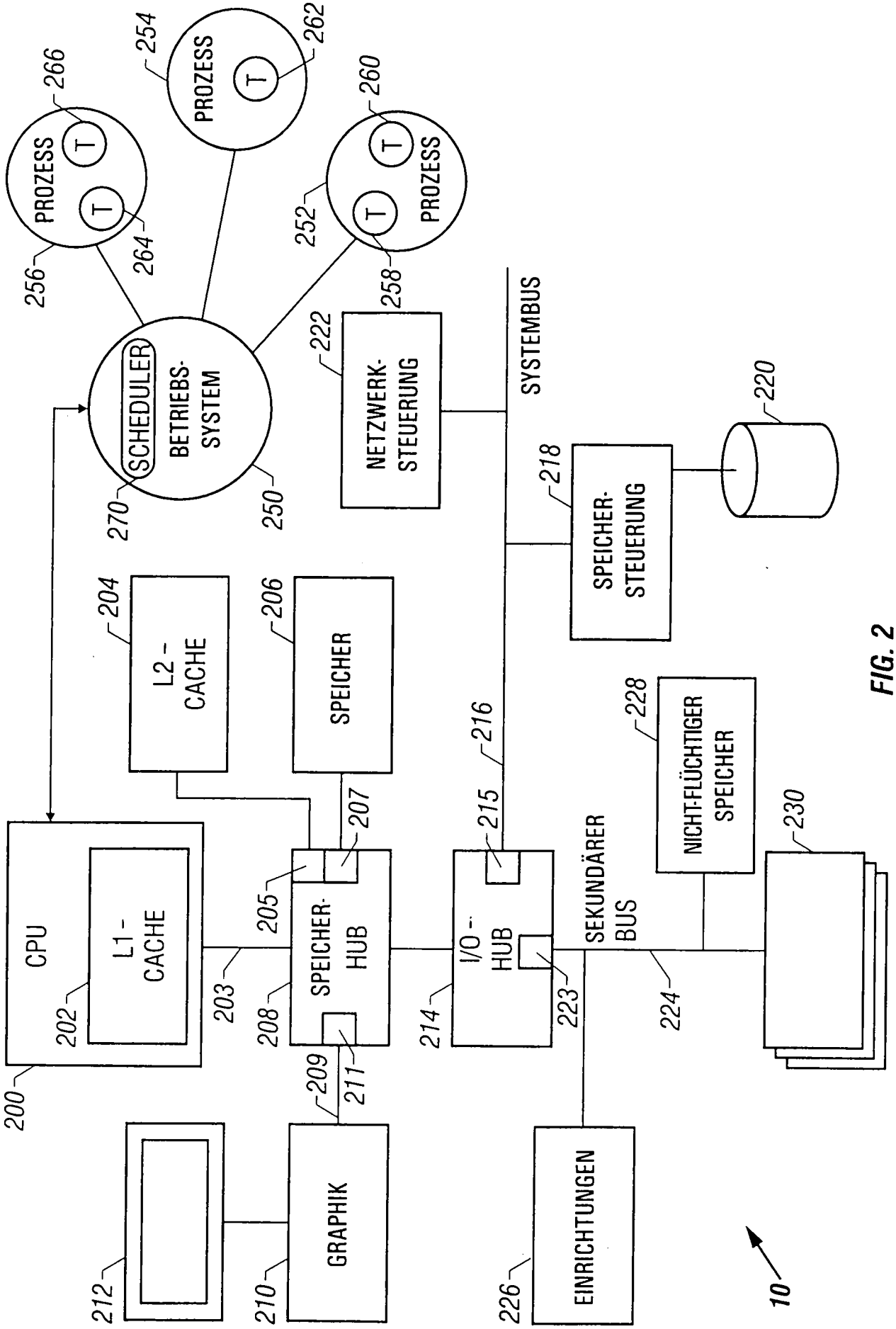

PROZESS
T
PROZESS
T
PROZESS
T
SCHEDULER
BETRIEBS-
SYSTEM
NETZWERK-
STEUERUNG
SYSTEMBUS
SPEICHER-
STEUERUNG
L2 -
CACHE
SPEICHER
CPU
L1 -
CACHE
SPEICHER-
HUB
I/O -
HUB
SEKUNDÄRER
BUS
NICHT-FLÜCHTIGER
SPEICHER
GRAPHIK
EINRICHTUNGEN
200
202
203
204
205
206
207
208
209
210
211
212
214
215
216
218
220
222
223
224
226
228
230
250
252
254
256
258
260
262
264
266
270
10

FIG. 2

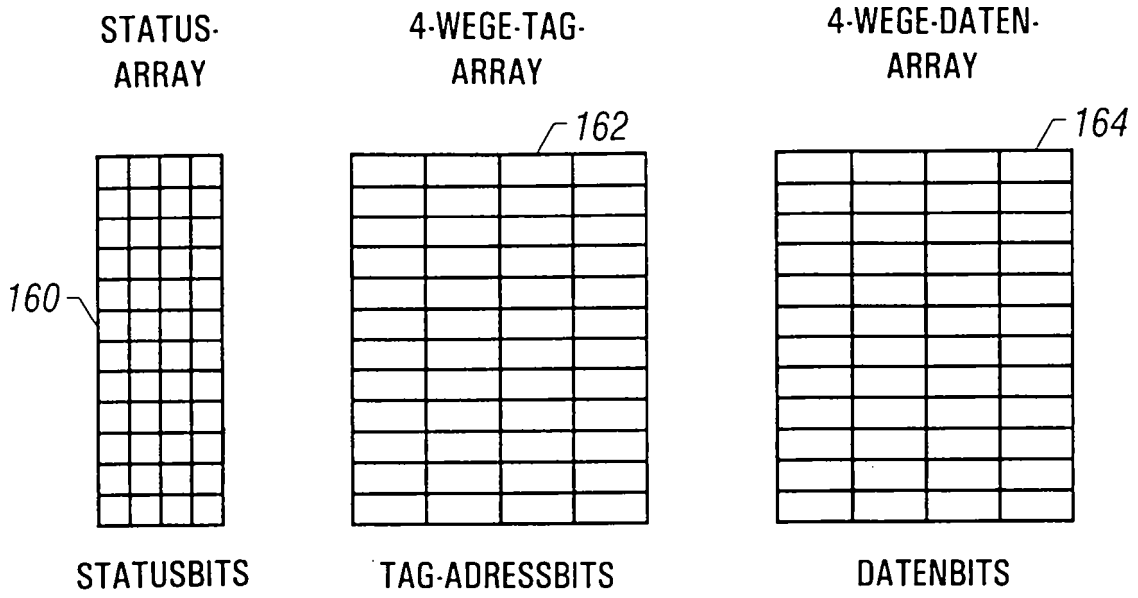

STATUS-
ARRAY
4-WEGE-TAG-
ARRAY
4-WEGE-DATEN-
ARRAY
160
162
164
STATUSBITS
TAG-ADRESSBITS
DATENBITS

FIG. 3

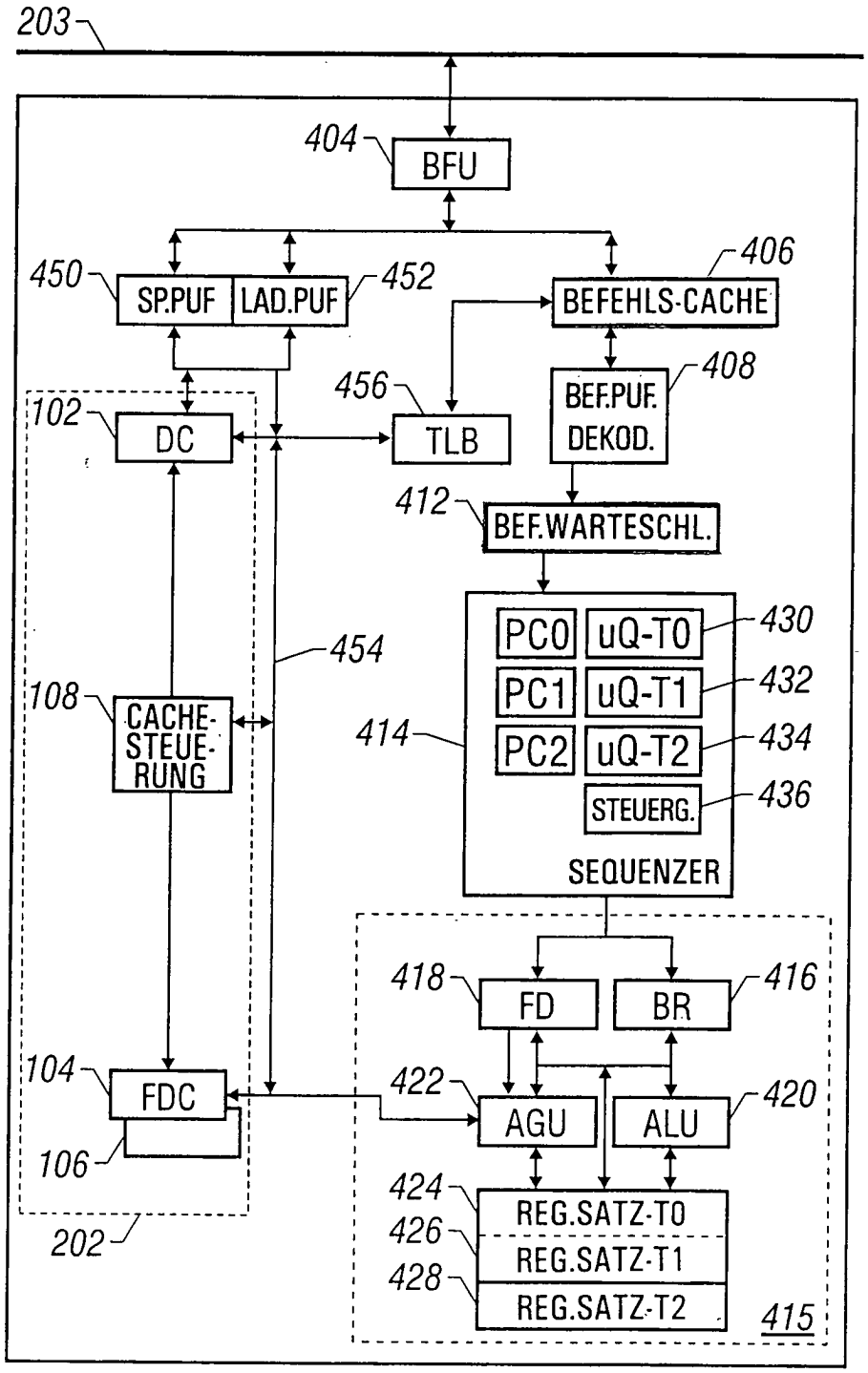

203
404
BFU
450
SP.PUF
LAD.PUF
452
406
BEFEHLS-CACHE
456
TLB
408
BEF.PUF.
DEKOD.
102
DC
412
BEF.WARTESCHL.
454
PC0
uQ-T0
430
PC1
uQ-T1
432
PC2
uQ-T2
434
STEUERG.
436
414
SEQUENZER
108
CACHE-
STEUE-
RUNG
418
FD
BR
416
104
FDC
422
AGU
ALU
420
106
424
REG.SATZ-T0
426
REG.SATZ-T1
428
REG.SATZ-T2
415
202

FIG. 4

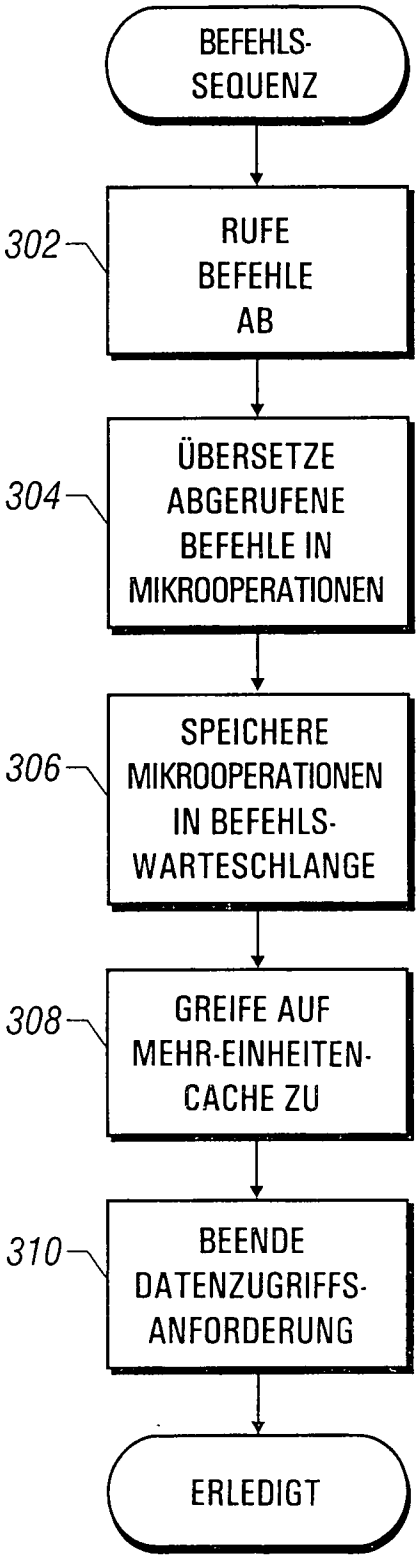

BEFEHLS-SEQUENZ
302
RUFE BEFEHLE AB
304
ÜBERSETZE ABGERUFENE BEFEHLE IN MIKROOPERATIONEN
306
SPEICHERE MIKROOPERATIONEN IN BEFEHLS-WARTESCHLANGE
308
GREIFE AUF MEHR-EINHEITEN-CACHE ZU
310
BEENDE DATENZUGRIFFS-ANFORDERUNG
ERLEDIGT

FIG. 5

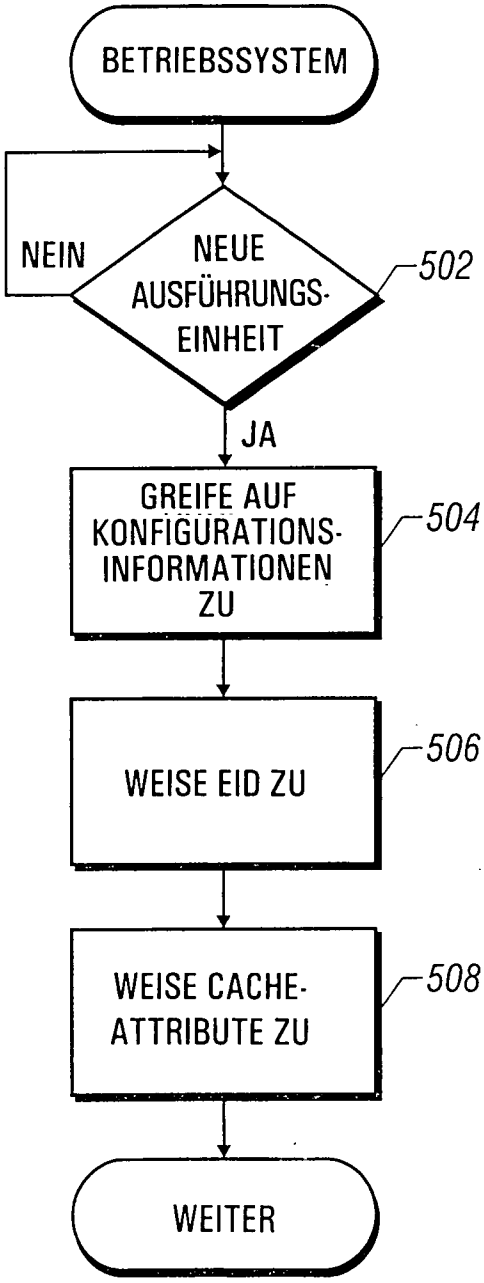

BETRIEBSSYSTEM
NEIN
NEUE AUSFÜHRUNGS-EINHEIT
502
JA
GREIFE AUF KONFIGURATIONS-INFORMATIONEN ZU
504
WEISE EID ZU
506
WEISE CACHE-ATTRIBUTE ZU
508
WEITER

FIG. 6