(19) Deutsches Patent- und Markenamt

(10) **DE 10 2014 003 690 A1** 2014.09.18

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2014 003 690.1**
(22) Anmeldetag: **14.03.2014**
(43) Offenlegungstag: **18.09.2014**

(51) Int Cl.: ***G06F 9/30*** (2006.01)
G06F 9/318 (2006.01)

(30) Unionspriorität:
13/844,881 **16.03.2013** **US**

(71) Anmelder:
Intel Corporation, Santa Clara, Calif., US

(74) Vertreter:
BOEHMERT & BOEHMERT Anwaltspartnerschaft mbB - Patentanwälte Rechtsanwälte, 28209 Bremen, DE

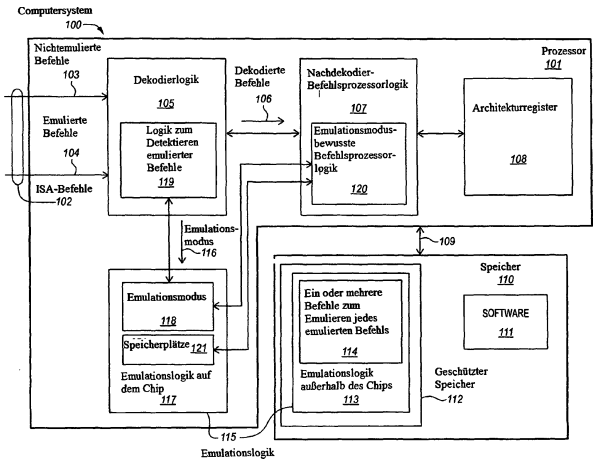
(72) Erfinder:
Rash, William C., Saratoga, Calif., US; Toll, Bret L., Hilsboro, Oreg., US; Hahn, Scott D., Portland, Oreg., US; Hinton, Glenn J., Portland, Oreg., US

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Prozessoren, Verfahren und Systeme zur Befehlsemulation**

(57) Zusammenfassung: Ein Prozessor enthält unter einem Aspekt Dekodierlogik zum Erhalten eines ersten Befehls und zum Bestimmen, dass der erste Befehl emuliert werden soll. Der Prozessor enthält auch emulationsmodusbewusste Nachdekodier-Befehlsprozessorlogik, die mit der Dekodierlogik gekoppelt ist. Die emulationsmodusbewusste Nachdekodier-Befehlsprozessorlogik soll ein oder mehrere Steuersignale verarbeiten, die aus einem Befehl dekodiert werden. Der Befehl ist einer aus einem Satz von einem oder mehreren Befehlen, die zum Emulieren des ersten Befehls verwendet werden. Die ein oder mehreren Steuersignale sollen durch die emulationsmodusbewusste Nachdekodier-Befehlsprozessorlogik anders verarbeitet werden, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt. Andere Vorrichtungen sind ebenfalls offenbart, ebenso wie Verfahren und Systeme.



Beschreibung

Hintergrund

Technisches Gebiet

[0001] Hier beschriebene Ausführungsformen beziehen sich allgemein auf Prozessoren. Insbesondere beziehen sich hier beschriebene Ausführungsformen allgemein auf Befehlsemulation in Prozessoren.

Hintergrundinformation

[0002] Prozessoren weisen gewöhnlich Befehlssatzarchitekturen (instruction set architectures (ISA)) auf. Die ISA stellt allgemein den Teil der Architektur des Prozessors dar, der sich auf Programmieren bezieht. Die ISA enthält gewöhnlich die nativen Befehle, Architekturregister, Datentypen, Addressierungsmodi und ähnliches der Prozessoren. Ein Teil der ISA ist der Befehlssatz. Der Befehlssatz enthält allgemein Makrobefehle oder Befehle auf ISA-Ebene, die dem Prozessor zur Ausführung bereitgestellt werden. Eine Ausführungslogik und andere Pipeline-Logik ist enthalten, um die Befehle des Befehlssatzes zu verarbeiten. Oftmals kann der Umfang solcher Ausführungs- und Pipeline-Logik beträchtlich sein. Gewöhnlich ist der Umfang solcher Logik umso größer, je mehr Befehle in dem Befehlssatz vorliegen und je komplexer und/oder spezieller die Befehle in dem Befehlssatz sind. Solche Hardware kann dazu tendieren, die Herstellungskosten, die Größe und/oder den Stromverbrauch der Prozessoren zu erhöhen.

Kurze Beschreibung der Zeichnungen

[0003] Die Erfindung kann am besten durch Bezugnahme auf die folgende Beschreibung und beigefügten Zeichnungen verstanden werden, die verwendet werden, um die Ausführungsformen der Erfindung zu veranschaulichen. In den Zeichnungen zeigt:

[0004] **Fig.** 1 ein Blockdiagramm einer Ausführungsform eines Computersystems,

[0005] **Fig.** 2 ein Blockflußdiagramm einer Ausführungsform eines Verfahrens zum Emulieren eines Befehls in einem Prozessor,

[0006] **Fig.** 3 ein Blockdiagramm, das eine Ausführungsform von Logik zum Emulieren eines Befehls mit einem Satz eines oder mehrerer Befehle zeigt,

[0007] **Fig.** 4 ein Blockdiagramm, das eine Ausführungsform von Logik zeigt, um einem Prozessor zu ermöglichen, Ausnahmebedingungen anders zu behandeln, wenn er sich in einem Emulationsmodus befindet, als wenn er sich nicht in dem Emulationsmodus befindet,

[0008] **Fig.** 5 ein Blockdiagramm, das eine Ausführungsform von Logik zeigt, um einem Prozessor zu ermöglichen, auf (ein) Betriebsmittel und/oder Information anders zuzugreifen, wenn er sich in einem Emulationsmodus befindet, als wenn er sich nicht in dem Emulationsmodus befindet,

[0009] **Fig.** 6 ein Blockflussdiagramm einer Ausführungsform eines Verfahrens, das durch und/oder in einem Prozessor durchgeführt wird,

[0010] **Fig.** 7 ein Blockdiagramm, das eine Ausführungsform von Logik zeigt, um einem Opcode zu ermöglichen, unterschiedliche Bedeutungen zu haben,

[0011] **Fig.** 8 ein Blockflussdiagramm einer Ausführungsform eines Verfahrens, das durch ein Betriebssystemmodul durchgeführt werden kann,

[0012] **Fig.** 9 ein Blockdiagramm einer Ausführungsform eines Programmladermoduls einschließlich eines Auswählmoduls, das betrieben werden kann, um einen Satz einer oder mehrerer Funktionen, Subroutinen oder andere Teile einer Softwarebibliothek auszuwählen, die eine Bedeutung eines gegebenen Opcodes haben, die für Software geeignet ist, die sie verwenden wird,

[0013] **Fig.** 10A ein Blockdiagramm, das sowohl eine beispielhafte geordnete Pipeline als auch eine beispielhafte ungeordnete Ausgabe-(issue)/Ausführungs-Pipeline mit Registerumbenennung gemäß Ausführungsformen der Erfindung zeigt,

[0014] **Fig.** 10B ein Blockdiagramm, das sowohl eine beispielhafte Ausführungsform eines geordneten Architekturkerns als auch einen ungeordneten Ausgabe-/Ausführungsarchitekturkern mit Registerumbenennung, der in einem Prozessor aufgenommen werden soll, gemäß Ausführungsformen der Erfindung zeigt,

[0015] **Fig.** 11A ein Blockdiagramm eines einzelnen Prozessorkerns gemeinsam mit seiner Verbindung zu dem Schaltnetzwerk auf dem Chip und mit seinem lokalen Teil des Level-2(L2)-Caches gemäß Ausführungsformen der Erfindung,

[0016] **Fig.** 11B eine erweiterte Ansicht des Prozessorkern aus **Fig.** 11A gemäß Ausführungsformen der Erfindung,

[0017] **Fig.** 12 ein Blockdiagramm eines Prozessors, der mehr als einen Kern, einen integrierten Speichercontroller und integrierte Grafikvorrichtung gemäß Ausführungsformen der Erfindung enthalten kann,

[0018] **Fig.** 13 ein Blockdiagramm eines Systems gemäß einer Ausführungsform der vorliegenden Erfindung,

[0019] Fig. 14 ein Blockdiagramm eines ersten konkreten beispielhaften Systems gemäß einer Ausführungsform der vorliegenden Erfindung,

[0020] Fig. 15 ein Blockdiagramm eines zweiten konkreten beispielhaften Systems gemäß Ausführungsformen der vorliegenden Erfindung,

[0021] Fig. 16 ein Blockdiagramm eines SoC gemäß einer Ausführungsform der vorliegenden Erfindung,

[0022] Fig. 17 ein Blockdiagramm, das die Verwendung eines Softwarebefehlswandlers zum Umwandeln von Binärbefehlen in einem Quellbefehlssatz zu Binärbefehlen in einem Zielbefehlssatz gemäß Ausführungsformen der Erfindung gegenüberstellt.

Ausführliche Beschreibung von Ausführungsformen

[0023] Vorliegend sind Prozessoren, Verfahren und Systeme zur Befehlsemulation offenbart. In der folgenden Beschreibung sind zahlreiche spezifische Details dargelegt (beispielsweise spezifische emulationsmodusbewußte Logik, Ansätze zum Behandeln von Ausnahmebedingungen, Typen privilegierter Betriebsmittel und Information, Logikimplementierungen, Mikroarchitekturdetails, Operationssequenzen, Logikpartitionierungs-/Integrationsdetails, Hardware-/Softwarepartitionierungsdetails, Prozessorkonfigurationen, Typen und Wechselbeziehungen von Systemkomponenten und ähnliche). Jedoch versteht sich, dass Ausführungsformen der Erfindung ohne diese spezifischen Details umgesetzt werden können. In anderen Beispielen sind wohlbekannte Schaltungen, Strukturen und Techniken nicht im Detail gezeigt, um das Verständnis dieser Beschreibung nicht zu verschleiern.

[0024] Fig. 1 zeigt ein Blockdiagramm einer Ausführungsform eines Computersystems **100**. In verschiedenen Ausführungsformen kann das Computersystem einen Desktopcomputer, Laptopcomputer, Notebookcomputer, Tabletcomputer, Netbook, Smartphone, Personal Digital Assistant, Mobiltelefon, Server, Netzwerkvorrichtung (beispielsweise Router oder Switch), mobiles Internetgerät (Mobile Internet Device (MID)), Medienabspielgerät, Smart-Fernsehgerät, Set-top-Box, Videospielcontroller oder anderen Typ elektronischer Vorrichtung darstellen.

[0025] Das Computersystem enthält eine Ausführungsform eines Prozessors **101**. In einigen Ausführungsformen kann der Prozessor ein Vielzweckprozessor sein. Beispielsweise kann der Prozessor ein Vielzweckprozessor des Typs sein, der gewöhnlich als eine zentrale Prozessoreinheit (central processing unit (CPU)) verwendet wird. In anderen Ausführungsformen kann der Prozessor ein Spezialprozessor sein. Beispiele für geeignete Spezialprozessoren schließen Koprozessoren, Grafikprozessoren, Kommunikationsprozessoren, Netzwerkprozessoren, Kryptografieprozessoren, eingebettete Prozessoren und digitale Signalprozessoren (DSPs) ein, um lediglich einige Beispiele zu nennen, ohne auf diese beschränkt zu sein. Der Prozessor kann irgendeiner von verschiedenen Prozessoren mit komplexem Befehlssatz (complex instruction set computing (CISC), verschiedenen Prozessoren mit reduziertem Befehlssatz (reduced instruction set computing (RISC)), verschiedenen Prozessoren mit sehr langem Befehlswort (very long instruction word (VLIW)), verschiedenen Hybriden derselben oder vollständig anderen Prozessorentypen sein.

[0026] Das Computersystem enthält außerdem eine Ausführungsform eines Speichers **110**, der mit dem Prozessor **101** durch einen Kopplungsmechanismus **109** gekoppelt ist. Jeder konventionelle Kopplungsmechanismus, der im Stand der Technik zum Koppeln eines Prozessors und eines Speichers bekannt ist, ist geeignet. Beispiele für solche Mechanismen schließen Schaltverbindungen, Busse, Hubs, Speichercontroller, Chipsätze, Chipsatzkomponenten und ähnliche sowie Kombinationen derselben ein, ohne auf diese beschränkt zu sein. Der Speicher kann eine oder mehrere Speichervorrichtungen entweder des gleichen oder unterschiedlicher Typen aufweisen. Ein gewöhnlich verwendeter Speichertyp, der für Ausführungsformen geeignet ist, ist dynamischer Speicher mit wahlfreiem Zugriff (dynamic random access memory (DRAM)), obwohl andere Speichertypen (beispielsweise Flashspeicher) alternativ verwendet werden können.

[0027] Der Speicher **110** kann darin Software **111** gespeichert aufweisen. Die Software kann beispielsweise ein oder mehrere Betriebssysteme (operating system (OS)) und eine oder mehrere Anwendungen enthalten. Im Betrieb kann ein Teil der Software in den Prozessor geladen und auf dem Prozessor zum Ablaufen gebracht werden. Wie gezeigt, kann der Prozessor ISA-Befehle **102** eines Befehlssatzes des Prozessors empfangen. Beispielsweise kann eine Befehlseinholeinheit die ISA-Befehle einholen. Die ISA-Befehle können Makrobefehle, Assemblerbefehle, Befehle auf Maschinenebene oder andere Befehle darstellen, die an den Prozessor geliefert werden, um dekodiert und ausgeführt zu werden. Wie gezeigt, können die ISA-Befehle in einigen Ausführungsformen sowohl nichtemulierte Befehle **103** als auch einen oder mehrere Typen emulierter Befehle **104** enthalten.

[0028] Der Prozessor enthält eine Dekodierlogik **105**. Die Dekodierlogik kann auch als eine Dekodiereinheit oder ein Dekodierer bezeichnet werden. Die Dekodierlogik kann die ISA-Befehle **102** empfangen. Im Falle der nichtemulierten Befehle **103** kann die Dekodierlogik die Befehle relativ höherer Ebenen dekodieren und einen oder mehrere Mikrobefehle, Mikro-

operationen, Mikrocode-Zugangspunkte relativ niedrigerer Ebene oder andere Befehle relativ niedrigerer Ebene oder Steuersignale, die von den ISA-Befehlen abgeleitet sind, ausgeben. In der Zeichnung sind diese als dekodierte Befehle **106** gezeigt. Die dekodierten Befehle, die von dem Dekodierer ausgegeben werden, können die ISA-Befehle höherer Ebene, die in den Dekodierer eingegeben werden, widerspiegeln, darstellen und/oder von diesen abgeleitet sein, und können die ISA-Befehle durch eine oder mehrere Operationen niedrigerer Ebene (beispielsweise auf Schaltungsebene oder Hardwareebene) implementieren. Der Dekodierer kann unter Verwendung verschiedener unterschiedlicher Mechanismen implementiert sein, einschließlich Mikrocode-Nurlesespeicher (read only memories (ROMs)), Wertetabellen, Hardwareimplementationen, programmierbarer Logikarrays (PLAs) und anderer Mechanismen, die verwendet werden, um Dekodierer, die im Stand der Technik bekannt sind, zu implementieren, ohne auf diese beschränkt zu sein.

[0029] Eine Nachdekodier-Befehlsprozessorlogik **107** ist mit der Dekodierlogik gekoppelt. Die Nachdekodier-Befehlsprozessorlogik kann einen Nachdekodierabschnitt der Befehlsverarbeitungspipeline des Prozessors darstellen. Die Nachdekodier-Befehlsprozessorlogik kann die dekodierten Befehle **106** empfangen und verarbeiten. Gewöhnlich kann die Nachdekodier-Befehlsprozessorlogik eine Registerlese- und/oder Speicherleselogik, Ausführungslogik, Register- und/oder Speicherrückschreiblogik und Ausnahmebehandlerlogik enthalten, obwohl die Logik von einer Architektur zur anderen variieren kann und der Schutzbereich der Erfindung nicht auf solche Logik beschränkt ist. In einigen Ausführungsformen kann, beispielsweise im Falle einer ungeordneten Prozessorpipeline, die Nachdekodier-Befehlsprozessorlogik optional andere Logik enthalten, wie etwa beispielsweise Zuweisungslogik, Umbenennungslogik, Planungslogik, Rückzugs- oder Überweisungslogik oder ähnliche.

[0030] Der Prozessor enthält auch einen oder mehrere Sätze architektonisch sichtbarer oder Architekturregister **108**. Die architektonisch sichtbaren Register stellen Register dar, die für Software und/oder einen Programmierer und/oder die Register, die durch die ISA-Befehle **102** angegeben werden, um Operanden zu bestimmen, sichtbar sind. Diese Architekturregister werden anderen nichtarchitektonischen oder nichtarchitektonisch sichtbaren Registern in einer gegebenen Mikroarchitektur (beispielsweise temporäre Register, die von Befehlen, Umordnungspuffern, Rückzugsregistern etc. verwendet werden) gegenübergestellt. Diese Architekturregister stellen allgemein Prozessorspeicherplätze auf dem Chip dar, die betrieben werden können, um Daten zu speichern. Diese Architekturregister werden hier oft einfach als Register bezeichnet. Beispielsweise können die Architekturregister einen Satz Vielzweckregister, einen Satz Register mit gepackten Daten, einen Satz Fließkommaregister, einen Satz Ganzzahlregister oder irgendeine Kombination derselben einschließen. Die Architekturregister können auf unterschiedliche Arten in unterschiedlichen Mikroarchitekturen unter Verwendung wohlbekannter Techniken implementiert sein und sind nicht auf irgendeinen bestimmten Schaltungstyp beschränkt. Beispiele geeigneter Typen von Architekturregistern schließen eigene physikalische Register, dynamisch zugewiesene physikalische Register unter Verwendung von Registerumbenennung und Kombinationen derselben ein, ohne auf diese beschränkt zu sein.

[0031] Die Nachdekodier-Befehlsprozessorlogik **107** ist mit den Registern **108** gekoppelt. Die Nachdekodier-Befehlsprozessorlogik kann Daten von den Registern empfangen und Daten in diese schreiben oder speichern. Beispielsweise kann die Registerleselogik Daten aus Registern lesen, die als Quelloperanden von Befehlen angezeigt sind, und/oder kann die Rückschreiblogik Ergebnisse in Register schreiben oder speichern, die als Zieloperanden der Befehle angezeigt sind. Die Nachdekodier-Befehlsprozessorlogik ist außerdem mit dem Speicher **110** gekoppelt und kann Daten von dem Speicher empfangen und in diesen speichern. Beispielsweise kann die Speicherleselogik Daten aus Speicherplätzen lesen, die durch Befehle angezeigt sind, und/oder kann die Speicherrückschreiblogik Daten in Speicherplätze schreiben, die durch Befehle angezeigt sind.

[0032] Unter erneuter Bezugnahme auf **Fig.** 1 können die emulierten Befehle **104** auch an die Dekodierlogik **105** geliefert werden. Im Gegensatz zu den nichtemulierten Befehlen **103** können die emulierten Befehle **104** nicht vollständig durch die Dekodierlogik dekodiert und als entsprechende dekodierte Befehle **106** der Nachdekodier-Befehlsprozessorlogik **107** bereitgestellt werden. Vielmehr kann die Emulationslogik **115** in einigen Ausführungsformen bereitgestellt werden, um den/die emulierten Befehl(e) **104** zu emulieren. Im Stand der Technik werden verschiedene unterschiedliche Begriffe für solche Emulation verwendet, wie etwa beispielsweise Befehlsübersetzung, binäre Übersetzung, Codemorphing, Befehlsinterpretation und ähnliche. Der Begriff Emulation wird hier breit verwendet, um diese unterschiedlichen Begriffe, die in der Industrie verwendet werden, zu umfassen.

[0033] Wie gezeigt, kann die Emulationslogik **115** in einigen Ausführungsformen zwischen teilweise auf dem Chip befindlicher Logik **117** und teilweise außerhalb des Chips befindlicher Emulationslogik verteilt werden, obwohl dies nicht notwendig ist. In anderen Ausführungsformen kann die gesamte Emulationslogik **115** optional auf dem Chip sein, oder kann ein größerer Teil optional außerhalb des Chips sein, ob-

wohl typischerweise wenigstens ein Teil der Emulationslogik auf dem Chip ist (beispielsweise ein Emulationsmodus **118**, irgendeine emulationsmodusbewußte Befehlsprozessorlogik **120** in der Pipeline etc.). Die Emulationslogik auf dem Chip ist fest, speicherresident oder liegt permanent mit dem Prozessor auf dem Chip vor. Gewöhnlich liegt die Emulationslogik mit dem Prozessor auf dem Chip vor, selbst wenn der Prozessor ausgeschaltet ist, vor einem Bootvorgang und/oder zum Zeitpunkt der Fertigstellung der Herstellung. Beispiele geeigneter Emulationslogik auf dem Chip schließt Hardware (beispielsweise integrierte Schaltungen, Transistoren etc.), Firmware (beispielsweise ROM auf dem Chip, EPROM, Flashspeicher oder anderen permanenten oder nichtflüchtigen Speicher und nichtflüchtige Befehle, die darin gespeichert sind) oder eine Kombination derselben ein, ohne auf diese beschränkt zu sein.

[0034] Die Emulationslogik **113** außerhalb des Chips kann in dem Speicher **110** enthalten sein. Die Emulationslogik außerhalb des Chips kann mit der Emulationslogik auf dem Chip gekoppelt sein oder auf andere Weise mit dieser in Kommunikation stehen. In einigen Ausführungsformen kann die Emulationslogik außerhalb des Chip in einem geschützten Bereich oder einem Abschnitt **112** des Speichers enthalten sein. In einigen Ausführungsformen kann der geschützte Abschnitt zur Verwendung durch Hardware auf dem Chip und/oder Firmwarelogik allein des Prozessors, jedoch nicht für die Software **111**, die auf dem Prozessor ausgeführt wird, reserviert sein. Beispielsweise kann in einigen Ausführungsformen die Emulationslogik **117**, die emulationsmodusbewußte Befehlsprozessorlogik **120** und/oder andere potentielle Prozessorlogik auf dem Chip in der Lage sein, auf die Emulationslogik **113** außerhalb des Speichers zuzugreifen und diese zu verwenden, jedoch kann die Software **111** (beispielsweise ein Betriebssystem oder eine Anwendung), die auf dem Prozessor läuft, nicht in der Lage sein, auf die Emulationslogik **113** außerhalb des Chips zuzugreifen oder diese zu verwenden. In einigen Ausführungsformen kann die Emulationslogik außerhalb des Chips vor Zugriff und Modifikation durch Anwendungen, das Betriebssystem, einen virtuellen Maschinenmanager, falls vorhanden, und/oder I/O-Vorrichtungen geschützt sein und/oder für diese unsichtbar sein. Dies kann dabei helfen, Sicherheit zu fördern.

[0035] Die Dekodierlogik enthält Logik **119**, um den emulierten Befehl **104** zu detektieren oder erkennen. Beispielsweise kann der Dekodierer den emulierten Befehl anhand eines Opcodes detektieren. In einigen Ausführungsformen kann der Dekodierer bei Detektieren des emulierten Befehls ein Emulationsmodussignal **116** (beispielsweise ein Emulationsfangsignal) an die Emulationslogik **115** liefern. Wie gezeigt, kann die Emulationslogik einen Emulationsmodus **118** aufweisen. Als Beispiel kann der Emulationsmodus ein oder mehrere Bits oder Steuerelemente in einem Steuer- oder Konfigurationsregister des Prozessors enthalten, um anzuzeigen, ob der Prozessor (beispielsweise die Logik **105**, **107** etc.) in dem Emulationsmodus ist oder nicht. In einigen Ausführungsformen kann der Emulationsmodus **118** nach Erhalt des Emulationsmodussignals **116** von dem Dekodierer angenommen werden, das anzeigt, dass ein emulierter Befehl **104** emuliert werden soll.

[0036] In einigen Ausführungsformen kann die Dekodierlogik **105** außerdem andere Information bereitstellen, die mit dem Befehl assoziiert ist, der für die Emulationslogik **115** emuliert wird. Beispiele solcher Information schließen potentiell Operandenbezeichner (beispielsweise Quell- oder Zielregisteradressen oder Speicherplätze), Speicheraddressiermodi, unmittelbare Addressen, Konstanten zur Ausführungsbeschleunigung und/oder andere Information, die von dem emulierten Befehl **104** stammt und/oder mit diesem assoziiert ist, ein. Als Beispiel kann jede Information, die von dem emulierten Befehl stammt und/oder mit diesem assoziiert ist, der für das Emulationssystem nützlich ist, um dem Emulationssystem zu ermöglichen, den emulierten Befehl **104** zu emulieren, potentiell bereitgestellt werden.

[0037] In einigen Ausführungsformen kann die Emulationslogik **115** einen anderen Satz eines oder mehrerer Befehle **114** enthalten, um jeden anderen Typ des emulierten Befehls **104** zu emulieren. Beispielsweise kann ein erster Satz eines oder mehrerer Befehle **114** bereitgestellt werden, um einen ersten Befehl **104** zu emulieren, der einen ersten Opcode aufweist, und kann ein zweiter, anderer Satz eines oder mehrerer Befehle **114** bereitgestellt werden, um einen zweiten, anderen Befehl **104** zu emulieren, der einen zweiten, anderen Opcode aufweist. In einigen Ausführungsformen kann jeder Satz wenigstens drei Befehle enthalten. In der gezeigten Ausführungsform kann der Satz eines oder mehrerer Befehle **114** in der Emulationslogik **113** außerhalb des Chips enthalten sein, obwohl dies nicht notwendig ist. In anderen Ausführungsformen können die Befehle **114** auf dem Chip bereitgestellt werden (beispielsweise in einem permanenten oder nichtflüchtigen Speicher der Emulationslogik **117** auf dem Chip). In noch weiteren Ausführungsformen kann ein Teil der Befehle **114** auf dem Chip bereitgestellt werden (beispielsweise in der Emulationslogik auf dem Chip), und kann ein Teil außerhalb des Chips bereitgestellt werden (beispielsweise in der Emulationslogik außerhalb des Chips).

[0038] In einigen Ausführungsformen kann jeder der Befehle des Satzes einen oder mehrere Befehle **114**, die verwendet werden, um den emulierten Befehl **104** zu emulieren, von der Emulationslogik **115** eingeholt oder auf andere Weise empfangen und der Dekodierlogik **105** bereitgestellt werden. In einigen Ausführungsformen kann jeder der Befehle des Satzes von

einem oder mehreren Befehlen **114**, die zum Emulieren des emulierten Befehls **104** verwendet werden, aus einem gleichen Befehlssatz stammen wie der emulierte Befehl **104**. Die Dekodierlogik **105** kann betrieben werden, jeden aus dem Satz von einem oder mehreren Befehlen **114** in entsprechende dekodierte Befehle **106** zu dekodieren. Die dekodierten Befehle können der Nachdekodier-Befehlsprozessorlogik **107** bereitgestellt werden.

[0039] Die Nachdekodier-Befehlsprozessorlogik enthält eine Ausführungsform einer emulationsmodusbewussten Befehlsprozessorlogik **120**. Wie gezeigt, kann die emulationsmodusbewusste Befehlsprozessorlogik mit dem Emulationsmodus **118** gekoppelt sein oder diesen auf andere Weise kennen. In einigen Ausführungsformen kann die emulationsmodusbewusste Befehlsprozessorlogik betrieben werden, um wenigstens einige der dekodierten Versionen der Befehle **114** wenigstens auf einige andere Arten zu verarbeiten, wenn der Prozessor in dem Emulationsmodus ist, als wenn der Prozessor nicht in dem Emulationsmodus ist. Es gibt verschiedene unterschiedliche Aspekte, unter denen die Verarbeitung anders sein kann. In einigen Ausführungsformen kann Störungs- oder Fehlerbehandlung anders durchgeführt werden, wenn der Emulationsmodus vorliegt, als wenn der Emulationsmodus nicht vorliegt. In anderen Ausführungsformen kann Zugriff auf bestimmte Typen von Betriebsmitteln und/oder Information, wie etwa beispielsweise sichere, privilegierte oder auf andere Weise zugriffsgesteuerte Betriebsmittel und/oder Information, anders behandelt werden, wenn der Emulationsmodus vorliegt, als wenn der Emulationsmodus nicht vorliegt. Beispielsweise kann Zugriff auf die Betriebsmittel und/oder Information ermöglicht werden, wenn der Emulationsmodus vorliegt, aber nicht ermöglicht werden, wenn der Emulationsmodus nicht vorliegt.

[0040] Wenn der Emulationsmodus vorliegt, kann die Nachdekodier-Befehlsprozessorlogik auf Speicherplätze **121** zugreifen. In der gezeigten Ausführungsform sind die Speicherplätze **121** teil der Emulationslogik **117** auf dem Chip. Alternativ können die Speicherplätze in der Emulationslogik außerhalb des Chips enthalten sein oder teilweise in der Emulationslogik auf dem Chip und teilweise in der Emulationslogik außerhalb des Chips enthalten sein. Die Speicherplätze können verwendet werden, um temporäre Variablen, Zwischenergebnisse und/oder Ausführungszustand, der mit der Ausführung des Satzes von Befehlen **114** assoziiert ist, zu speichern. Dies kann helfen, zu vermeiden, dass der Ausführungszustand des ursprünglichen Programms, das den emulierten Befehl **104** enthält, gespeichert werden muss, und/oder kann helfen, solchen Ausführungszustand (beispielsweise den Inhalt der Architekturregister **108**) daran zu hindern, durch die Verarbeitung des Satzes von Befehlen **114** fehlerhaft zu werden. In einigen Ausführungsformen können die Speicherplätze **121** Architekturregister emulieren, obwohl dies nicht notwendig ist. In einigen Ausführungsformen kann der Inhalt der Speicherplätze **121** vom Zugriff durch Anwendungen, Betriebssysteme, virtuellen Maschinenmanagern, I/O-Vorrichtungen, Interrupts und ähnlichen unabhängig, isoliert und/oder geschützt sein. Bei Abschluss des Satzes von Befehlen **114** kann der Architekturzustand des Prozessors aktualisiert werden (beispielsweise kann ein Ergebnis der Speicherplätze **121** in die Register **108** gespeichert werden). Dies kann mit niedrigem Latenzzugriff erfolgen. Gewöhnlich kann dies verwendet werden, um die Änderung im Architekturzustand, die aufgetreten wäre, und/oder das Verhalten des Prozessors, das erfolgt wäre, wenn der emulierte Befehl **104** tatsächlich direkt ausgeführt worden wäre, zu approximieren, imitieren, nachzubauen oder auf andere Weise zu emulieren.

[0041] Um ein Verschleiern der Beschreibung zu vermeiden, wurde ein relativ einfacher Prozessor **101** gezeigt und beschrieben. In anderen Ausführungsformen kann der Prozessor optional andere wohlbekannte Komponenten enthalten. Es gibt buchstäblich zahlreiche unterschiedliche Kombinationen und Konfigurationen von Komponenten in Prozessoren, und die Ausführungsformen sind nicht auf irgendeine bestimmte Kombination oder Konfiguration beschränkt. Der Prozessor kann eine integrierte Schaltung oder einen Satz eines oder mehrerer Halbleiterplatten oder -chips darstellen (beispielsweise eine einzelne Leiterplatte oder Chip oder ein Paket, das zwei oder mehr Leiterplatten oder Chips enthält). In einigen Ausführungsformen kann der Prozessor ein System-on-Chip (SoC) und/oder einen Chip-Multiprozessor (CMP) darstellen.

[0042] Einige Prozessoren verwenden relativ komplexe Operationen. Beispielsweise führen einige Befehle mehrere Speicherzugriffe anstatt lediglich einen einzelnen Speicherzugriff durch. Ein Beispiel ist ein Befehl zum Vektoreinholen, um einen Vektor von Datenelementen aus einem Speicher zu holen. Als ein weiteres Beispiel können bestimmte Befehle zahlreiche Datenelementvergleiche durchführen, anstatt ein einzelnes Paar von Datenelementen oder Paare entsprechenden der Datenelementen in zweigepackten Daten zu vergleichen. Beispiele sind Vektorkonfliktbefehle und Zeichenkettenverarbeitungsbefehle. Ein Ansatz ist, solche komplexen Operationen vollständig in Hardware zu implementieren. Jedoch kann die Menge an benötigter Hardware tendenziell beträchtlich sein, was dazu führen kann, Herstellungskosten, Chipgröße und Stromverbrauch zu erhöhen. Ein weiterer Ansatz ist, solche komplexen Operationen wenigstens teilweise in Mikrocode zu implementieren. Die Verwendung von Mikrocode kann helfen, die Menge an benötigter Hardware zu reduzieren, um solche komplexen Operationen zu implementieren,

und/oder kann helfen, bestimmte existierende Hardware wiederzuverwenden. Jedoch benutzen einige Prozessoren keinen Mikrocode (beispielsweise keinen Mikrocode zum Implementieren irgendeines Befehls eines Befehlssatzes).

[0043] In einigen Ausführungsformen kann ein relativ komplexerer Befehl mit dem Satz eines oder mehrerer relativ einfacherer Befehle emuliert werden. Die Begriffe komplexer und einfacher sind relative Begriffe, keine absoluten Begriffe, die zueinander relativ sind. Vorteilhafterweise kann dies potentiell helfen, die Menge an Hardware zu reduzieren, die benötigt wird, um den komplexeren Befehl zu implementieren, und/oder kann helfen, Wiederverwendung existierender Hardware zu ermöglichen, die von den ein oder mehreren Befehlen verwendet wird, die verwendet werden, um den komplexeren Befehl zu emulieren. In einigen Ausführungsformen kann die Emulation des komplexeren Befehls mit den ein oder mehreren einfacheren Befehlen verwendet werden, um eine Mikrocode-ähnliche Implementation des komplexeren Befehls bereitzustellen, selbst wenn der Prozessor in einigen Ausführungsformen nicht konfiguriert sein kann, um Mikrocode zu verwenden, und/oder nicht konfiguriert sein kann, Mikrocode zu verwenden, um den komplexeren Mikrobefehl zu implementieren.

[0044] **Fig.** 2 zeigt ein Blockflussdiagramm einer Ausführungsform eines Verfahrens **230** zum Emulieren eines Befehls in einem Prozessor. In einigen Ausführungsformen können die Operationen und/oder Verfahren aus **Fig.** 2 durch den Prozessor aus **Fig.** 1 und/oder innerhalb dieses Prozessors durchgeführt werden. Die Komponenten, Strukturen und spezifischen optionalen Details, die hier für den Prozessor aus **Fig.** 1 beschrieben sind, gelten optional für die Operationen und/oder Verfahren aus **Fig.** 2. Alternativ können die Operationen und/oder Verfahren aus **Fig.** 2 ähnlichen oder ganz anderen Prozessor und/oder innerhalb dieses Prozessors durchgeführt werden. Darüber hinaus kann der Prozessor aus **Fig.** 1 Operationen und/oder Verfahren durchführen, die ähnlich oder anders sind als die aus **Fig.** 2.

[0045] Das Verfahren schließt Empfangen eines ersten Befehls an Block **231** ein. In einigen Ausführungsformen kann der erste Befehl an einem Dekodierer empfangen werden. Das Verfahren schließt Bestimmen des Emulierens des ersten Befehls an Block **232** ein. In einigen Ausführungsformen kann der Dekodierer bestimmen, den ersten Befehl zu emulieren, indem bestimmt wird, dass ein Opcode des ersten Befehls in einem Satz eines oder mehrerer Opcodes für zu emulierende Befehle vorliegt. Das Verfahren schließt an Block **233** Empfangen eines Satzes eines oder mehrerer Befehle ein, die zum Emulieren des ersten Befehls verwendet werden sollen. In einigen Ausführungsformen kann der Satz von Befehlen an dem Dekodierer von Emulationslogik auf dem Chip oder Emulationslogik außerhalb des Chips oder einer Kombination derselben empfangen werden. In einigen Ausführungsformen kann jeder Befehl aus einem gleichen Befehlssatz wie der erste Befehl stammen. Das Verfahren schließt Verarbeiten eines oder mehrerer Steuersignale an Block **234** ein, die von einem Befehl des Satzes anders abgeleitet sind, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt.

[0046] Dies kann in unterschiedlichen Ausführungsformen auf unterschiedliche Weise vorgenommen werden. In einigen Ausführungsformen können Ausnahmebedingungen, denen während einer Verarbeitung eines Befehls des Satzes begegnet wird, unterschiedlich behandelt werden. In einigen Ausführungsformen kann die Verarbeitung eines Befehls des Satzes Zugriff auf Information und/oder Betriebsmittel ermöglichen, die dem gleichen Befehl (d. h. einem Befehl, der den gleichen Opcode aufweist andernfalls nicht zugänglich wären, wenn dies nicht innerhalb eines Emulationsmodus vorgenommen wird.

[0047] **Fig.** 3 zeigt ein Blockdiagramm, das eine Ausführungsform der Logik **301** zum Emulieren eines Befehls (beispielsweise eines komplexen Befehls) **304** mit einem Satz eines oder mehrerer Befehle (beispielsweise einfacherer Befehle) **314** zeigt. In einigen Ausführungsformen kann die Logik aus **Fig.** 3 in dem Prozessor und/oder dem Computersystem aus **Fig.** 1 enthalten sein. Alternativ kann die Logik aus **Fig.** 3 in einem ähnlichen oder anderen Prozessor oder Computersystem enthalten sein. Darüber hinaus kann der Prozessor und/oder das Computersystem aus **Fig.** 1 ähnliche oder andere Logik als die aus **Fig.** 3 enthalten.

[0048] Ein Befehl (beispielsweise ein komplexer Befehl) **304**, der emuliert werden soll, kann für die Dekodierlogik **305** bereitgestellt werden. Die Dekodierlogik kann eine Logik **319** zum Detektieren des Befehls **304** enthalten, um beispielsweise zu detektieren, dass ein Opcode des Befehls **304** in dem Satz von Opcodes für Befehle, die zu emulieren sind, enthalten ist. Wie gezeigt, kann der Prozessor in einigen Ausführungsformen keinen Mikrocode **330** aufweisen. Die Dekodierlogik kann ein Emulationsmodussignal **316** für die Emulationslogik **35** bereitstellen. In verschiedenen Ausführungsformen kann die Emulationslogik **315** Logik auf dem Chip, Logik außerhalb des Chips oder sowohl auf dem Chip als auch außerhalb des Chips enthalten. Die Emulationslogik kann als Antwort auf das Emulationsmodussignal in einen Emulationsmodus **318** eintreten.

[0049] Die Emulationslogik enthält außerdem einen Satz eines oder mehrerer einfacherer (beispielsweise einfacherer) Befehle **314**, die verwendet werden können, um den (beispielsweise komplexeren) Be-

fehl **304** zu emulieren. In einigen Ausführungsformen können die ein oder mehreren Befehle **314** aus dem gleichen Befehlssatz wie der Befehl **304** stammen. In einigen Ausführungsformen können die ein oder mehreren Befehle **314** mit anderen Befehlen identisch sein, die dekodiert und ausgeführt werden, wenn kein Emulationsmodus vorliegt. Um den (beispielsweise komplexen) Befehl **304** zu emulieren, kann jeder der ein oder mehreren (beispielsweise einfacheren) Befehle **314** an die Dekodierlogik geliefert werden. Die Dekodierlogik kann jeden der Befehle **314** als einen oder mehrere dekodierte Befehle **306** dekodieren.

[0050] Eine Nachdekodier-Befehlsprozessorlogik **307** kann die dekodierten Befehle **306** erhalten, die den Befehlen **314** entsprechen. Die Nachdekodier-Befehlsprozessorlogik kann eine Ausführungsform der emulationsmodusbewussten Logik **320** enthalten. Wie gezeigt, kann die emulationsmodusbewusste Logik in einigen Ausführungsformen mit dem Emulationsmodus **318** gekoppelt sein oder diesen auf andere Weise kennen. In einigen Ausführungsformen kann die emulationsmodusbewusste Logik betrieben werden, um die dekodierten Befehle **306**, die den Befehlen **314** entsprechen, anders zu verarbeiten, wenn der Prozessor in dem Emulationsmodus **318** ist, als wenn der Prozessor nicht in dem Emulationsmodus ist. In einigen Ausführungsformen kann Störungs- oder Fehlerbehandlung anders durchgeführt werden, wenn der Emulationsmodus vorliegt, als wenn der Emulationsmodus nicht vorliegt. Beispielsweise kann die Logik **320** optionale Aspekte verwenden, die unten für **Fig.** 4 diskutiert sind. In anderen Ausführungsformen kann Zugriff auf bestimmte Betriebsmittel und/oder Information selektiv bereitgestellt werden, wenn der Emulationsmodus vorliegt, aber nicht, wenn der Prozessor nicht in dem Emulationsmodus ist. Beispielsweise kann die Logik **320** optionale Aspekte verwenden, die unten für **Fig.** 5 diskutiert sind.

[0051] Vorteilhafterweise kann in einigen Ausführungsformen ein komplexerer Befehl durch einen Satz einfacherer Befehle/Operationen implementiert sein. Vorteilhafterweise kann dies potentiell helfen, die Menge an Hardware zu reduzieren, die benötigt wird, um den komplexeren Befehl zu implementieren, und/oder kann helfen, Wiederverwendung existierender Hardware zu ermöglichen, die von den ein oder mehreren Befehlen verwendet wird, die verwendet werden, um den komplexeren Befehl zu emulieren. In einigen Ausführungsformen kann die Emulation des komplexeren Befehls mit den ein oder mehreren einfacheren Befehlen verwendet werden, um eine Mikrocode-ähnliche Implementation des komplexeren Befehls bereitzustellen, obwohl der Prozessor in einigen Ausführungsformen nicht konfiguriert sein kann, um Mikrocode zu verwenden und/oder nicht konfiguriert sein kann, Mikrocode zu verwenden, um den komplexeren Befehl zu implementieren. In einigen Ausführungsformen können die einfacheren Befehle/Operationen sogar aus dem gleichen Befehlssatz stammen wie der komplexere Befehl.

[0052] Solche Emulation komplexerer Befehle mit einfacheren Befehlen ist lediglich ein Beispiel für einen möglichen Grund, einen Befehl zu emulieren. In anderen Ausführungsformen kann der emulierte Befehl einer sein, der relativ weniger häufig verwendet wird (beispielsweise selten verwendet wird), und kann mit einem oder mehreren Befehlen emuliert werden, die relativ häufiger verwendet werden. Vorteilhafterweise kann dies potentiell helfen, die Menge an Hardware, die benötigt wird, um den selten verwendeten Befehl zu implementieren, zu reduzieren, und/oder kann helfen, Wiederverwendung existierender Hardware zu ermöglichen, die von den ein oder mehreren Befehlen verwendet wird, die verwendet werden, um den selten verwendeten Befehl zu emulieren. In noch weiteren Ausführungsformen kann der emulierte Befehl ein älterer und/oder veralteter Befehl sein und/oder kann einer sein, der im Begriff ist, zu veralten, und kann mit den ein oder mehreren anderen Befehlen emuliert werden. Vorteilhafterweise kann die Emulation helfen, dem Befehl, der veraltet, zu ermöglichen, weiterhin ausgeführt zu werden und somit Rückwärtskompatibilität für Software bereitzustellen, während gleichzeitig potentiell geholfen wird, die Menge an Hardware zu reduzieren, die benötigt wird, um den veralteten Befehl zu implementieren, und/oder geholfen wird, Wiederverwendung existierender Hardware zu ermöglichen, die von den ein oder mehreren Befehlen verwendet wird, die verwendet werden, um den veralteten Befehl zu emulieren. Noch weitere Verwendungen der hier offenbarten Emulation werden Durchschnittsfachleuten ersichtlich sein, die die vorliegende Offenbarung zur Kenntnis nehmen.

[0053] **Fig.** 4 zeigt ein Blockdiagramm, dass eine Ausführungsform von Logik **401** zeigt, um einem Prozessor zu ermöglichen, Ausnahmebedingungen anders zu behandeln, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt. In einigen Ausführungsformen kann die Logik aus **Fig.** 4 in dem Prozessor und/oder dem Computersystem aus **Fig.** 1 und/oder der Logik aus **Fig.** 3 enthalten sein. Alternativ kann die Logik aus **Fig.** 4 in einem ähnlichen oder anderen Prozessor oder Computersystem enthalten sein. Darüber hinaus kann der Prozessor und/oder das Computersystem aus **Fig.** 1 und/oder die Logik aus **Fig.** 3 ähnliche oder andere Logik als die aus **Fig.** 4 enthalten.

[0054] Eine erste Instanz **403-1** eines gegebenen Befehls (beispielsweise eines Befehls, der einen gegebenen Opcode aufweist) wird an die Dekodierlogik **405** geliefert, wenn der Prozessor nicht in einem Emulationsmodus **418** ist. Eine zweite Instanz **403-2**

des gleichen gegebenen Befehls (beispielsweise eines anderen Befehls, der den gleichen gegebenen Opcode aufweist) wird an die Dekodierlogik geliefert, wenn der Prozessor in dem Emulationsmodus **418** arbeitet. Die zweite Instanz **403-2** des gegebenen Befehls kann als Antwort auf einen Dekodierer, der den emulierten Befehl erhält, aus einem Satz eines oder mehrerer Befehle **414** bereitgestellt werden, die verwendet werden, um einen emulierten Befehl zu emulieren. Der Satz von Befehlen kann in einer Emulationslogik **415** enthalten sein, die auf dem Chip, außerhalb des Chips oder teilweise auf dem Chip und außerhalb des Chips ist. Die Emulationslogik **515** kann jede der optionalen Charakteristiken aufweisen, die hier an anderer Stelle für die Emulationslogik erwähnt werden. Die Dekodierlogik kann (beispielsweise einen identischen Satz) eines oder mehrerer dekodierter Befehle für jeden der ersten **403-1** und zweiten Instanzen **403-2** des gegebenen Befehls bereitstellen.

[0055] Die Nachdekodier-Befehlsprozessorlogik **407** kann den/die dekodierten Befehl(e) **406** erhalten. Die Nachdekodier-Befehlsprozessorlogik weist eine emulationsmodusbewusste Ausnahmebehandlerlogik **420** auf. Die emulationsmodusbewusste Ausnahmebehandlerlogik kann betrieben werden, um Ausnahmebedingungen auf eine emulationsmodusbewusste Weise zu behandeln/verarbeiten. Wie hier verwendet, bezieht sich der Begriff ”Ausnahmebedingung” grob auf verschiedene unterschiedliche Typen von Ausnahmebedingungen, die auftreten können, während Befehle verarbeitet werden. Beispiele solcher Ausnahmebedingungen schließen Ausnahmen, Interrupts, Störungen (faults), Fangstellen (traps) und ähnliche ein, ohne darauf beschränkt zu sein. Die Begriffe Ausnahme, Interrupt, Fault und Trap werden im Stand der Technik häufig auf verschiedene Art verwendet. Der Begriff ”Ausnahme” wird vermutlich häufiger verwendet, um sich auf eine automatisch erzeugte Steuerungsübergabe an eine Behandlerroutine als Antwort auf Rechteverletzungen, Rechteausnahmen, Page-Faults, Speicherschutzstörungen, Division durch Null, versuchte Ausführung eines ungültigen Opcodes und andere solche Ausnahmebedingungen zu beziehen.

[0056] In einigen Ausführungsformen kann, wenn eine Rechteverletzung, ein Page-Fault, eine Speicherschutzstörung, eine Division durch Null, eine versuchte Ausführung eines ungültigen Opcodes oder andere Ausnahmebedingung auftritt, wenn die erste Instanz **403-1** des gegebenen Befehls verarbeitet wird, wenn der Prozessor nicht in dem Emulationsmodus **418** arbeitet, der Prozessor im wesentlichen herkömmliches Behandeln der Ausnahmebedingung durchführen. Beispielsweise kann die Ausnahmebedingung in einigen Ausführungsformen direkt angenommen **440** werden, wobei eine Steuerung an eine Ausnahmebehandlerroutine **441** übergeben wird. Gewöhnlich kann die Ausnahmebehandlerroutine teil eines Betriebssystems, eines virtuellen Maschinenmonitors oder anderer privilegierter Software sein. Beispiele solcher Behandlerroutinen schließen Page-Fault-Behandler, Fehler-Behandler, Interrupt-Behandler und ähnliche ein, ohne darauf beschränkt zu sein.

[0057] Andererseits kann in einigen Ausführungsformen, wenn eine Berechtigungsverletzung, Page-Fault, Speicherschutzstörung, Division durch Null, versuchte Ausführung eines ungültigen Opcodes oder andere Ausnahmebedingung auftritt, wenn die zweite Instanz **403-2** des gegebenen Befehls verarbeitet wird, wenn der Prozessor in dem Emulationsmodus **418** arbeitet, der Prozessor im wesentlich unkonventionelle Behandlung der Ausnahmebedingung durchführen. Beispielsweise kann die Ausnahmebedingung in einigen Fällen nicht direkt angenommen werden. In einigen Ausführungsformen kann die Logik **420** einen Mechanismus zum Unterdrücken einer andernfalls automatischen Steuerübertragung an eine Ausnahmebehandlerroutine enthalten, die andernfalls aus der Ausnahmebedingung resultieren würde. Die Steuerung braucht nicht direkt von dem Emulationsprogramm zu der Ausnahmebehandlerroutine **441** übertragen zu werden. Vielmehr kann die emulationsmodusbewusste Ausnahmebehandlerlogik **420** in einigen Ausführungsformen Steuerungsübertragung an den Ausnahmebedingungsbehandler **441** vorübergehend unterdrücken und die Ausnahmebedingung indirekt berichten **442**. In einigen Ausführungsformen kann die emulationsmodusbewusste Ausnahmebehandlerlogik **420** die Ausnahmebedingung durch ein oder mehrere Emulationskommunikationsregister **443** indirekt berichten. Die ein oder mehreren Kommunikationsregister können verwendet werden, um Information zwischen der Emulationslogik und dem Programm zu kommunizieren, für das der ursprüngliche Befehl emuliert wird.

[0058] In einigen Ausführungsformen kann die emulationsmodusbewusste Ausnahmebehandlerlogik **420** als Antwort auf Auftreten der Ausnahmebedingung, wenn ein Emulationsmodus vorliegt, eine Anzeige der Ausnahmebedingung in einer Ausnahmebedingung oder einem Fehlerstatusflag(s), -feld oder -register **444** speichern. Beispielsweise kann ein einzelnes Bit oder Flag einen ersten Wert (beispielsweise auf binäre Eins gesetzt) aufweisen, um anzuzeigen, dass eine Ausnahmebedingung aufgetreten ist, oder kann einen zweiten Wert (beispielsweise zu binärer Null gelöscht) aufweisen, um anzuzeigen, dass keine Ausnahmebedingung aufgetreten ist. In einigen Ausführungsformen kann die emulationsmodusbewusste Ausnahmebehandlerlogik **420** als Antwort auf Auftreten der Ausnahmebedingung, wenn der Emulationsmodus **418** vorliegt, einen Fehlercode für die Ausnahmebedingung in einem Fehlercodefeld oder -register **445** speichern. Der Fehlercode kann

weitere Information über den Fehler bereitstellen, wie etwa beispielsweise einen Typ des Fehlers und optional weitere Details, um Kommunizieren der Beschaffenheit der Ausnahmebedingung zu unterstützen. Alternativ kann, anstatt die Kommunikationsregister zu verwenden, die Information auf andere Weise signalisiert oder bereitgestellt werden (beispielsweise in Speicher gespeichert werden, durch ein elektrisches Signal berichtet werden etc.).

[0059] In einigen Ausführungsformen kann die emulationsmodusbewusste Ausnahmebehandlerlogik **420** außerdem eine Anzeige der Adresse (beispielsweise den Befehlszeiger) des Befehls, der emuliert wird (d. h. desjenigen, der die zweite Instanz **403-2** veranlasste, an die Dekodierlogik **405** gesendet zu werden) bereitstellen. Beispielsweise kann in einigen Ausführungsformen die Adresse **446** des Befehls, der emuliert wird, oben auf dem Stack **447** gespeichert werden. Speichern der Adresse eines gegebenen Befehls, der emuliert wird, auf dem Stack anstatt derjenigen eines der Befehle, die verwendet werden, um diesen gegebenen Befehl zu emulieren, kann dazu fühhren, dass die Rückgabe von dem Ausnahmebehandler an den emulierten Befehl anstatt an einen der Befehle zurückkehrt, die verwendet werden, um diesen emulierten Befehl zu emulieren. Wenn andernfalls die Rückgabe von dem Ausnahmebehandler an einen der Befehle gerichtet wäre, die verwendet werden, um diesen Befehl zu emulieren, kann dies potentiell ein Problem verursachen. Beispielsweise kann Software (beispielsweise eine Anwendung, ein Betriebssystem etc.) die Befehle nicht kennen, die verwendet werden, um diesen gegebenen Befehl zu emulieren, und kann die assoziierte Adresse nicht erkennen. Das Betriebssystem kann erkennen, dass Steuerfluss an einen unbekannten, ungültigen, riskanten oder nicht erlaubten Ort übertragen wird, und kann potentiell versuchen, die Übergabe zu verhindern.

[0060] In einigen Ausführungsformen kann der Satz von Befehlen **414** den Fehlerstatus **444** und/oder den Fehlercode **445** überwachen. Beispielsweise können die Befehle **414** in einigen Ausführungsformen den Fehlerstatus **444** und den Fehlercode **445** aus den Emulationskommunikationsregistern **443** lesen, um die Ausnahmebedingung und Angaben über die Ausnahmebedingung in Erfahrung zu bringen. Wenn der Fehlerstatus **444** eine Ausnahmebedingung anzeigt, kann der Satz von Befehlen **414** in einigen Ausführungsformen die Ausnahmebedingung **449** annehmen. Beispielsweise können einer oder mehrere der Befehle **414** ausgeführt werden, um den Fehlerstatus zu prüfen und Steuerung an den Ausnahmebehandler zu übergeben, wenn ein Fehler angezeigt wird. In einigen Ausführungsformen kann dies einschließen, dass der Satz von Befehlen **414** Steuerung an den Ausnahmebehandler **441** übergibt. In einigen Ausführungsformen kann Information über die Ausnahmebedingung (beispielsweise der Fehlercode **445**) für den Ausnahmebehandler **441** bereitgestellt werden. In einigen Ausführungsformen kann die emulierte Befehlsadresse **446** auch an den Ausnahmebehandler **441** geliefert und/oder wenigstens oben auf dem Stack erhalten werden. Die emulierte Befehlsadresse **446** kann von dem Ausnahmebehandler **441** nach Rückgabe vom Behandeln der Ausnahmebedingung verwendet werden. Vorteilhafterweise kann das Betriebssystem oder eine andere Fehlerbehandlerroutine denken, dass der Befehl, der emuliert wird, die Quelle des Fehlers ist.

[0061] In einigen Ausführungsformen kann die Emulationslogik Logik zum Testen und zum Berichten, ob Speicherzugriff in dem Befehl korrekt funktioniert, enthalten oder den Typ der Ausnahmebedingung angeben, die entstehen kann. Beispielsweise kann ein spezieller Befehl enthalten sein, um eine Speicheradresse mit emulierten Adressrechten zu testen, um festzustellen, ob die Speicheradresse gültig ist (beispielsweise, ob die Seite vorliegt) und ob das Programm ausreichende Zugriffsrechte besitzt, um den Speicherplatz zu lesen und/oder um den Speicherplatz zu modifizieren. Falls irgendwelche Tests scheitern, kann die Emulationslogik Steuerung an den richtigen Interruptbehandler mit einer Rückgabeadresse übergeben, als ob der Befehl, der emuliert wird, direkt Steuerung an den Ausnahmebehandler übergeben hätte. Als ein weiteres Beispiel kann ein Zustandsautomat eine bedingte Speichertransaktion durchführen, die anzeigt, ob die Speicheroperation gültig wäre. Dies kann verwendet werden, um zu bestimmen, wann eine Speicheroperation durchgeführt werden kann unter der Annahme, dass keine Ausnahme entsteht. Dies kann außerdem verwendet werden, um zu bestimmen, wie viele Bytes eines Befehlsstreams oder einer Zeichenkette aus Befehlsinformation auf sichere Weise ohne Ausnahmen gelesen werden können. Beispielsweise kann dies verwendet werden, um zu testen und zu bestimmen, ob eine Befehlslänge gelesen werden kann oder nicht, oder ob ein Teil der Befehlslänge einen Seitenfehler verursachen würde oder nicht. Die Emulationslogik kann Logik enthalten, um mit Befehlen umzugehen, die mehrere Seiten umfassen, und/oder wenn eine Seite nicht im Speicher vorliegt.

[0062] In einigen Ausführungsformen kann die Emulationslogik Logik enthalten, um einen zwischenzeitigen Ausführungsinterruptstatus bereitzustellen, so dass Ausführung der Emulation anhalten und später an dem Zwischenpunkt fortfahren kann. Dies kann insbesondere vorteilhaft sein, wenn Befehle emuliert werden, die lange Dauern oder Ausführungszeiten einschließen. In einigen Ausführungsformen kann der Satz von Befehlen, die verwendet werden, um bestimmte Typen von Befehlen (beispielsweise Zeichenkettenbefehle bewegen, Befehle einholen und andere mit langen Operationen) zu emu-

lieren, den Ausführungszustand der Software aktualisieren, die den Befehl aufweist, der emuliert wird, um eine gegenwärtige Fortschrittsstufe wiederzugeben. Beispielsweise kann die Operation an einem Zwischenpunkt unterbrochen werden, und kann der Satz von Befehlen, die zur Emulation verwendet werden, ein Flag oder Statusbit in dem gespeicherten Maschinenzustand durch den Ausnahmebehandler setzen (beispielsweise in einem Prozessorstatusregister), so dass bei Rückgabe der Emulationscode in der Lage sein kann, das Flag oder Statusbit zu testen, um zu bestimmen, dass er Ausführung von einem Zwischenzustand aus fortsetzt. Das Flag oder Statusbit kann unterbrochene Ausführung anzeigen. Auf diese Weise kann das Programm, wenn es von einem Ausnahmebehandler zurückkehrt, nachdem eine Ausnahmebedingung behandelt wurde, Ausführung an einer zwischenzeitigen Fortschrittsstufe fortsetzen, wo es aufgehört hatte. In einigen Fällen kann ein Befehl (beispielsweise ein Befehl zum Bewegen einer Zeichenkette) Register modifizieren, um einen Zwischenzustand der Operation wiederzugeben, so dass nach einer Unterbrechung Ausführung von dem Zwischenzustand aus fortgesetzt werden kann.

[0063] **Fig.** 5 zeigt ein Blockdiagramm, das eine Ausführungsform einer Logik **501** zeigt, um einem Prozessor zu ermöglichen, auf Betriebsmittel und/oder Information auf andere Weise zuzugreifen, wenn er in einem Emulationsmodus ist, als wenn er nicht in einem Emulationsmodus ist. In einigen Ausführungsformen kann die Logik aus **Fig.** 5 in dem Prozessor und/oder dem Computersystem aus **Fig.** 1 und/oder aus **Fig.** 3 enthalten sein. Alternativ kann die Logik aus **Fig.** 5 in einem ähnlichen oder anderen Prozessor oder Computersystem enthalten sein. Darüberhinaus können der Prozessor und/oder das Computersystem aus **Fig.** 1 und/oder die Logik aus **Fig.** 3 ähnliche oder andere Logik als die aus **Fig.** 5 enthalten.

[0064] Eine erste Instanz **503-1** eines gegebenen Befehls (beispielsweise eines Befehls, der einen gegebenen Opcode aufweist) wird einer Dekodierlogik **505** bereitgestellt, wenn der Prozessor nicht in einem Emulationsmodus **518** ist. Eine zweite Instanz **503-2** des gleichen gegebenen Befehls (beispielsweise eines anderen Befehls, der den gleichen gegebenen Opcode aufweist) wird der Dekodierlogik bereitgestellt, wenn der Prozessor in dem Emulationsmodus **518** arbeitet. Die zweite Instanz **503-2** des gegebenen Befehls kann als Antwort darauf, dass ein Dekodierer den emulierten Befehl erhält, von einem Satz eines oder mehrerer Befehle **514** bereitgestellt werden, die verwendet werden, um einen emulierten Befehl zu emulieren. Der Satz von Befehlen kann in der Emulationslogik **515** enthalten sein, die auf dem Chip, außerhalb des Chips oder teilweise auf dem Chip und außerhalb des Chips sein kann. Die Emulationslogik **515** kann jede der optionalen Charakteristiken aufweisen, die hier an anderer Stelle für die Emulationslogik erwähnt werden.

[0065] Eine Nachdekodier-Befehlsprozessorlogik **507** kann den/die dekodierten Befehl(e) **506** erhalten, die der zweiten Instanz **503-2** entsprechen. Die Nachdekodier-Befehlsprozessorlogik enthält eine emulationsmodusbewusste Zugriffssteuerlogik **520**. Die emulationsmodusbewusste Zugriffssteuerlogik kann betrieben weren, Zugriff auf ein oder mehrere Betriebsmittel und/oder Information **550** auf eine Weise zu steuern, die über den Emulationsmodus informiert ist. In einigen Ausführungsformen kann, wenn der Prozessor nicht in dem Emulationsmodus arbeitet, die Nachdekodier-Befehlsprozessorlogik **507** die erste Instanz **503-1** des gegebenen Befehls mit im wesentlichen herkömmlichem Zugriff auf das/die Betriebsmittel und/oder Information **550** verarbeiten. Wie gezeigt, kann in einigen Ausführungsformen Zugriff auf Betriebsmittel und/oder Information **550** verhindert werden **551**, wenn die erste Instanz **503-1** des gegebenen Befehls verarbeitet wird, wenn kein Emulationsmodus vorliegt. Verhindern von Zugriff auf das/die Betriebsmittel und/oder Information, wenn kein Emulationsmodus vorliegt, kann aus irgendeinem von verschiedenen möglichen Gründen angemessen sein, wie etwa beispielsweise, um die Sicherheit von Information und/oder Betriebsmittel(n) zu schützen, weil der gegebene Befehle allgemein nicht auf diese(s) Betriebsmittel und/oder Information zuzugreifen braucht und man das/die Betriebsmittel und/oder Information lediglich bei Bedarf oder aus anderen Gründen bereitstellen will.

[0066] Hingegen kann in einigen Ausführungsformen, wenn die zweite Instanz **503-2** des gegebenen Befehls verarbeitet wird, wenn in dem Emulationsmodus **518** gearbeitet wird, die Nachdekodier-Befehlsprozessorlogik im wesentlichen unkonventionellen Zugriff auf das/die Betriebsmittel und/oder Information **550** (beispielsweise auf eine Weise, die anders ist, als wenn ein Nichtemulationsmodus vorliegt) verwenden. Beispielsweise kann, wie in der gezeigten Ausführungsform gezeigt, Zugriff auf das/die Betriebsmittel und/oder Information **550** erlaubt werden **552**, wenn die zweite Instanz **503-2** des gegebenen Befehls verarbeitet wird, wenn der Emulationsmodus **518** vorliegt. Als Beispiel kann der Emulationsmodus **518** der Logik **507** und/oder der Logik **520** ermöglichen, einen speziellen Hardwarezustand zu haben, der selektiven Zugriff auf die Information und/oder Betriebsmittel für diesen gegebenen Befehl ermöglicht, wenn Emulationsmodus vorliegt. Beispielsweise können ein oder mehrere Zugriffsberechtigungsbits bereitgestellt und konfiguriert werden, wenn der Emulationsmodus vorliegt, um einem Zustandsautomaten zu ermöglichen, auf die Information zuzugreifen.

[0067] Verschiedene unterschiedliche Typen von Information und/oder Betriebsmittel(n) **550** werden betrachtet. Beispiele eines geeigneten/geeigneter Betriebsmittel und/oder Information schließen (ein) sicherheitsbezogene(s) Betriebsmittel und/oder Information (beispielsweise Sicherheitslogik), (ein) verschlüsselungs- und/oder entschlüsselungsbezogene (s) Betriebsmittel und/oder Information (beispielsweise Verschlüsselungslogik und/oder Entschlüsselungslogik), (ein) Zufallsgeneratorbetriebsmittel und/oder -information (beispielsweise Zufallsgeneratorlogik), (ein) Betriebsmittel und/oder Information, die für Berechtigungs- oder Ringstufen reserviert sind, die einem Betriebssystem und/oder virtuellen Maschinenmonitor entsprechen, und ähnliches ein, ohne darauf beschränkt zu sein.

[0068] Ein weiteres Beispiel eines geeigneten/geeigneter Betriebsmittel(s) und/oder Information schließt (ein) Betriebsmittel und/oder Information in einem anderen physikalischen Prozessor oder logischen Prozessor (beispielsweise einen Kern, Hardwarethread, Threadkontext etc.) als dem physikalischen oder logischen Prozessor ein, der die Nachdekodier-Befehlsprozessorlogik **507** aufweist. Die unterschiedlichen physikalischen oder logischen Prozessoren können entweder in den gleichen oder unterschiedlichen Sockets sein. Als Beispiel kann, wenn ein Emulationsmodus vorliegt, eine emulationsmodusbewusste Steuerlogik **520** in der Lage sein, auf Information und/oder Betriebsmittel eines anderen Kerns in einem anderen Socket (beispielsweise Anfordern eines Status des Kerns) zuzugreifen, der nicht für die Nachdekodier-Befehlsprozessorlogik **507** zur Verfügung steht, wenn kein Emulationsmodus vorliegt.

[0069] Vorteilhafterweise kann die emulationsmodusbewusste Zugriffssteuerlogik **520** helfen, wenigstens einigen der Befehle **514** zu ermöglichen, selektiv auf (ein) bestimmte(s) Betriebsmittel und/oder Information zuzugreifen, wenn der Emulationsmodus vorliegt, die normalerweise nicht für die gleichen Befehle des Befehlssatzes zur Verfügung stünden, wenn kein Emulationsmodus vorliegt. Sicherheit kann trotzdem aufrechterhalten werden, da die Emulationslogik auf dem Chip und/oder in einem geschützten Abschnitt des Speichers sein kann.

[0070] In einigen Ausführungsformen können einige Ausführungslevel, beispielsweise Sicherheitsausführungszustände, daran gehindert sein, solche Emulation zu verwenden, um auf diese(s) Betriebsmittel und/oder Information zuzugreifen. Beispielsweise braucht nicht sämtlichen Ausführungszuständen erlaubt zu werden, emulierte Opcodes zu verwenden. Spezielle Sicherheitsausführungszustände können nicht zertifizierbar sicher sein, wenn solche Interrupts oder Ausführung auf niedriger Ebene erlaubt wird. Stattdessen können, wenn solche Ausführungsstufen oder Sicherheitsausführungszustände ähnlichen Zugriff benötigen, ihn diese stattdessen durch Verwenden von Hardwareprimitiven implementieren, die für Emulationssoftware zur Verfügung stehen.

[0071] In einigen Ausführungsformen kann Befehlsemulation verwendet werden, um Bereitstellen unterschiedlicher Bedeutungen für einen gegebenen Opcode eines Befehls zu unterstützen. Makrobefehle, Maschinensprachbefehle und andere Befehle eines Befehlssatzes weisen oftmals einen Operationscode oder Opcode auf. Der Opcode stellt allgemein einen Teil des Befehls dar, der verwendet wird, um den konkreten Befehl und/oder die Operation anzugeben, die als Antwort auf den Befehl durchgeführt werden soll. Beispielsweise kann ein Opcode eines gepackten Multiplizierbefehls anders sein als ein Opcode eines gepackten Addierbefehls. Allgemein weist der Opcode mehrere Bits in einem oder mehreren Feldern auf, die logisch, wenn nicht physikalisch, zusammen angeordnet sind. Oftmals ist es wünschenswert, zu versuchen, die Opcodes relativ kurz oder so kurz wie möglich zu halten, während die gewünschte Anzahl von Befehlen/Operationen ermöglicht wird. Relativ lange Opcodes tendieren dazu, die Größe und/oder Komplexität des Dekodierers zu vergrößern, und tendieren auch allgemein dazu, die Befehle zu verlängern. Für eine feste Anzahl an Bits in einem Opcode kann allgemein lediglich eine feste Anzahl unterschiedlicher Befehle/Operationen identifiziert werden. Es gibt verschiedene Tricks, die im Stand der Technik bekannt sind, um zu versuchen, den Opcode am besten zu nutzen, beispielsweise durch Verwenden von Escape-Codes und ähnlichem. Nichtsdestoweniger ist die Anzahl von Befehlen, die mit einem Opcode eindeutig identifiziert werden können, allgemein beschränkter als wünschenswert ist. Allgemein können neue Befehle nicht kontinuierlich dem Opcode-Raum des Prozessors hinzugefügt werden, ohne an einem Punkt irgendwann keine Opcodes mehr zur Verfügung zu haben.

[0072] Auslastungen ändern sich mit der Zeit. Auf ähnliche Weise ändern sich gewünschte Befehle und gewünschte Befehlsfunktionalitäten mit der Zeit. Neue Befehlsfunktionalitäten werden gewöhnlich fortwährend zu Prozessoren hinzugefügt. Auf ähnliche Weise werden einige Befehle/Operationen mit der Zeit relativ weniger nützlich und/oder weniger häufig benutzt und/oder weniger wichtig. In einigen Fällen, wenn Befehle/Operationen ausreichend begrenzten Nutzen oder Wichtigkeit haben, können sie veraltet sein. Veraltung ist ein Begriff, der im Stand der Technik gewöhnlich verwendet wird, um einen Status zu bezeichnen, der auf eine Komponente, eine Struktur, eine Charakteristik oder Vorgehen angewendet wird, um darauf hinzuweisen, dass sie/es allgemein vermieden werden sollte, oftmals weil sie/es im Begriff ist, verworfen oder ersetzt zu werden und/

oder in der Zukunft nicht zur Verfügung stehen oder unterstützt werden kann.

[0073] Gewöhnlich können solche Befehle/Operationen veralten, anstatt sofort entfernt zu werden, um zu helfen, vorübergehende Rückwärtskompatibilität zu liefern (beispielsweise existierendem oder überliefertem Code Weiterbetrieb zu ermöglichen). Dies kann Zeit einräumen, um den Code mit den Ersatzbefehlen/-operationen in Einklang zu bringen, und/oder Zeit einräumen, den existierenden oder überlieferten Code auszusondern. Oftmals braucht Aussondern von Befehlen/Operationen aus einem Befehlssatz viel Zeit, beispielsweise in der Größenordnung von Jahren, wenn nicht Jahrzehnten, um Zeit zu geben, alte Programme ausreichend zu entfernen. Herkömmlicherweise könnte der Wert des Opcodes des veralteten Befehls der veralteten Operation allgemein nicht wiedererlangt und für einen anderen Befehl/eine andere Operation wiederverwendet werden, bis eine so lange Zeitperiode verstrichen ist. Andererseits können, falls überlieferte Software zum Ablauf gebracht wurde, Befehle, die den Opcode-Wert aufweisen, den Prozessor veranlassen, die Ersatzoperation anstelle der beabsichtigten veralteten Operation durchzuführen, was ein fehlerhaftes Ergebnis zur Folge haben könnte.

[0074] In einigen Ausführungsformen kann Befehlsemulation verwendet werden, um Bereitstellen unterschiedlicher Bedeutungen für einen gegebenen Opcode eines Befehls zu unterstützen. In einigen Ausführungsformen kann der gegebene Opcode des Befehls mit unterschiedlichen Bedeutungen interpretiert werden. In einigen Ausführungsformen können mehrere Opcode-Definitionen für den gegebenen Opcode unterstützt werden. Beispielsweise kann der gegebene Opcode mit einer Bedeutung interpretiert werden, die ein Softwareprogramm, das den Befehl aufweist, beabsichtigt. Als Beispiel kann ein älteres oder überliefertes Softwareprogramm anzeigen, dass Befehle mit dem gegebenen Opcode eine ältere, überlieferte oder veraltete Bedeutung haben sollen, und kann ein neueres Softwareprogramm anzeigen, dass Befehle mit dem gegebenen Opcode eine neuere Bedeutung haben sollen. In einigen Ausführungsformen kann die ältere oder veraltete Bedeutung emuliert werden, während die neuere Bedeutung in Steuersignale dekodiert und direkt auf der Pipeline des Prozessors ausgeführt werden kann. Vorteilhafterweise kann dies in einigen Ausführungsformen helfen, früheres Wiedererlangen und Wiederverwendung von Opcodes, die veraltet sind, zu ermöglichen, während trotzdem Rückwärtskompatibilität bereitgestellt wird, die älteren Programmen ermöglicht, immer noch mit einem veralteten Opcode abzulaufen, während dem veralteten Opcode ermöglicht wird, auch für neuere Programme mit einer anderen Bedeutung verwendet zu werden, um beim Verbessern von Leistung zu helfen.

[0075] **Fig.** 6 zeigt ein Blockflußdiagramm einer Ausführungsform eines Verfahrens **660**, das durch einen und/oder in einem Prozessor durchgeführt wird. In einigen Ausführungsformen können die Operationen und/oder das Verfahren aus **Fig.** 6 durch den und/oder in dem Prozessor aus **Fig.** 1 und/oder der Logik aus **Fig.** 3 oder **Fig.** 7 durchgeführt werden. Die Komponenten, Strukturen und spezifischen optionalen Details, die hier für den Prozessor und die Logik beschrieben sind, gelten optional auch für die Operationen und/oder Verfahren aus **Fig.** 6. Alternativ können die Operationen und/oder Verfahren aus **Fig.** 6 durch einen und/oder innerhalb eines ähnlichen oder ganz anderen Prozessors oder Logik durchgeführt werden. Darüberhinaus kann der Prozessor aus **Fig.** 1 und/oder die Logik aus **Fig.** 3 oder **Fig.** 7 ähnliche oder andere Operationen und/oder Verfahren als die aus **Fig.** 6 durchführen.

[0076] Das Verfahren schließt Empfangen eines ersten Befehls an Block **661** ein, der einen gegebenen Opcode aufweist. In einigen Ausführungsformen kann der erste Befehl an dem Dekodierer empfangen werden. Eine Bestimmung kann an Block **662** vorgenommen werden, ob der gegebene Opcode eine erste Bedeutung oder eine erste Bedeutung aufweist. In einigen Ausführungsformen kann die erste Bedeutung eine erste Opcode-Definition und die zweite Bedeutung eine zweite, andere Opcode-Definition sein. Wie weiter unten erläutert wird, kann dies in einigen Ausführungsformen einschließen, dass der Dekodierer eine Anzeige, beispielsweise in einem Flag, Statusregister oder anderem Speicherplatz auf dem Chip, lesen oder prüfen kann, ob der gegebene Opcode die erste Bedeutung oder die zweite Bedeutung aufweist. Wie weiter unten erläutert wird, kann in einigen Ausführungsformen Software (beispielsweise ein Programmladermodul eines Betriebssystemmoduls) die Anzeige in dem Flag, Statusregister oder anderem Speicherplatz auf dem Chip speichern, wenn Software geladen wird, um von dem Prozessor zum Ablauf gebracht zu werden. Als Beispiel kann die Software Metadaten (beispielsweise ein Objektmodulformat) enthalten, um anzuzeigen, ob die Software den gegebenen Opcode erwartet oder spezifiziert, die erste Bedeutung oder zweite Bedeutung aufzuweisen.

[0077] Unter erneuter Bezugnahme auf **Fig.** 6 kann, falls die Bestimmung an Block **662** ergibt, dass der gegebene Opcode die erste Bedeutung aufweist, das Verfahren zu Block **663** fortschreiten. An Block **663** kann der erste Befehl in einem oder mehrere Mikrobefehle, Mikrooperationen oder andere Befehle oder Steuersignale niedrigerer Ebene dekodiert werden. In einigen Ausführungsformen kann der Dekodierer diese(n) Befehl(e) oder diese(s) Steuersignal (e) an die Nachdekodier-Befehlsprozessorlogik (beispielsweise Ausführungseinheiten etc.) ausgeben. Die Nachdekodier-Befehlsprozessorlogik kann die-

se Befehle verarbeiten, typischerweise viel schneller, als wenn stattdessen Emulation verwendet werden würde. In einigen Ausführungsformen kann die erste Bedeutung für nicht-veraltete Opcode-Bedeutungen, relativ neuere Opcode-Bedeutungen, relativ häufiger verwendete Opcode-Bedeutungen, Opcode-Bedeutungen, die Leistung stärker beeinflussen, oder ähnliche verwendet werden.

[0078] Umgekehrt kann, falls die Bestimmung an Block **662** ergibt, dass der gegebene Opcode die zweite Bedeutung aufweist, das Verfahren zu Block **664** fortschreiten. Bei Block **664** kann Emulation des ersten Befehls herbeigeführt werden. Beispielsweise kann der Dekodierer ein Emulationsfangsignal liefern oder auf andere Weise einen Emulationsmodus an die Emulationslogik signalisieren. Anschließend kann ein Satz von einem oder mehreren Befehlen der Emulationslogik, die zum Emulieren des ersten Befehls verwendet werden sollen, wobei der Opcode die zweite Bedeutung aufweist, dem Dekodierer geliefert und in dem Emulationsmodus verarbeitet werden. Dies kann im Wesentlichen wie hier an anderer Stelle beschrieben vorgenommen werden. In einigen Ausführungsformen kann die zweite Bedeutung für veraltete Opcode-Bedeutungen, Opcode-Bedeutungen, die im Begriff sind, zu veralten oder vor der Aussonderung stehen, relativ ältere Opcode-Bedeutungen, relativ weniger häufig verwendete Opcode-Bedeutungen, Opcode-Bedeutungen, die weniger stark Leistung beeinflussen, oder ähnliche verwendet werden.

[0079] **Fig.** 7 zeigt ein Blockdiagramm, das eine Ausführungsform der Logik **701** zeigt, um einem gegebenen Opcode zu ermöglichen, unterschiedliche Bedeutungen aufzuweisen. In einigen Ausführungsformen kann die Logik aus **Fig.** 7 in dem Prozessor und/oder dem Computersystem aus **Fig.** 1 und/oder der Logik aus **Fig.** 3 enthalten sein. Alternativ kann die Logik aus **Fig.** 7 in einem ähnlichen oder anderen Prozessor oder Computersystem enthalten sein. Darüberhinaus kann der Prozessor und/oder das Computersystem as **Fig.** 1 und/oder die Logik aus **Fig.** 3 ähnliche oder andere Logik als die aus **Fig.** 7 enthalten.

[0080] Ein Speicher **710** enthält ein erstes Softwaremodul **711-1**, ein zweites Softwaremodul **711-2** und ein Betriebssystemmodul **797**, das ein Programmladermodul **770** aufweist. In einigen Ausführungsformen enthält das erste Softwaremodul eine Anzeige **772** zum Verwenden einer ersten Bedeutung für einen gegebenen Opcode, und enthält das zweite Softwaremodul eine Anzeige **773** zum Verwenden einer zweiten, anderen Bedeutung für den gegebenen Opcode. Als Beispiel können die ersten und zweiten Softwaremodule jeweils ein Objektmodulformat, andere Metadaten oder eine oder mehrere Datenstrukturen enthalten, die diese Anzeigen **772**, **773** enthalten. Das Programmladermodul kann betrieben werden, um das erste Softwaremodul und das zweite Softwaremodul zur Ausführung auf einem Prozessor zu laden. Wie gezeigt, kann das Programmladermodul in einigen Ausführungsformen ein Modul **771** enthalten, um eine Bedeutung des gegebenen Opcodes, der von dem konkreten Softwaremodul angezeigt wird, auf den Prozessor als Prozessorzustand zu laden. In einigen Ausführungsformen kann das Modul **771** betrieben werden, um die Anzeige **772**, wenn das erste Softwaremodul geladen wird, oder die Anzeige **773**, wenn das zweite Softwaremodul geladen wird, auf einen Speicherplatz **774** auf dem Chip als eine Anzeige **775** zu laden, ob die erste oder zweite Bedeutung für den gegebenen Opcode verwendet werden soll. Der Speicherplatz auf dem Chip ist mit einem Dekodierer **705** gekoppelt oder auf andere Weise für diesen zugänglich.

[0081] In einigen Ausführungsformen kann, beispielsweise im Falle eines alten Softwaremoduls, das Softwaremodul keine explizite Anzeige zum Verwenden einer gegebenen Bedeutung für den gegebenen Opcode aufweisen. Beispielsweise kann die Software vor der Existenz der neueren Bedeutung geschrieben worden sein. In einigen Ausführungsformen kann das Modul **771** und/oder der Programmlader **770** betrieben werden, um festzustellen, ob das Softwaremodul die erste oder zweite Bedeutung des gegebenen Opcodes verwenden muß. Beispielsweise kann dies aus einer Merkmalsliste, die in dem Programm eingebettet ist, dem Format des Programms, dem Alter des Programms oder dem Jahr, in dem das Programm erstellt wurde, oder anderer solcher Information in den Metadaten und/oder in dem Softwaremodul festgestellt werden. Beispielsweise kann, falls das zweite Softwaremodul **711-2** alte Software ist, die vor Einführung/Definition der ersten Bedeutung des gegebenen Opcodes erstellt wurde, das Programmladermodul und/oder das Betriebssystemmodul betrieben werden, um festzustellen, dass das zweite Softwaremodul die zweite Bedeutung und nicht die erste Bedeutung des gegebenen Opcodes verwenden muss. Das Modul **771** kann betrieben werden, um die Anzeige **775** in dem Speicherbereich auszuschalten oder auszuwechseln (swap), wenn Software ausgeschaltet oder ausgewechselt wird.

[0082] Um dies weiter beispielhaft zu zeigen, wird eine erste Instanz **703-1** eines Befehls betrachtet, wobei der gegebene Opcode einem Dekodierer **705** von dem ersten Softwaremodul **711-1** bereitgestellt wird. Das erste Softwaremodul enthält die Anzeige **772** zum Verwenden der ersten Bedeutung für den gegebenen Opcode, den das Modul **771** in dem Speicherplatz **774** vorhalten kann. Der Dekodierer enthält eine Prüflogik **776**, die mit dem Speicherplatz **774** gekoppelt ist, um die Anzeige **775** darauf zu prüfen, ob die erste oder zweite Bedeutung für den gegebenen Opcode verwendet werden soll. Die Prüflo-

gik kann auf den Speicherplatz zugreifen oder aus diesem lesen und bestimmen, dass die erste Bedeutung für den gegebenen Opcode verwendet werden soll, wenn die erste Instanz des Befehls von dem ersten Softwaremodul verarbeitet wird. In einigen Ausführungsformen kann der Speicherplatz **774** mehrere unterschiedliche Speicherplätze enthalten, um mehrere Anzeigen zu speichern, die jeweils einem anderen Opcode entsprechen. Als Antwort kann die Dekodierlogik **777** des Dekodierers den Befehl unter der Annahme der ersten Bedeutung des gegebenen Opcodes dekodieren. Ein oder mehrere dekodierte Befehle **706** oder ein oder mehrere andere Steuersignale können von dem Dekodierer an die Nachdekodier-Befehlsprozessorlogik **707** geliefert werden, die diese verarbeiten kann.

[0083] Eine zweite Instanz **703-2** eines Befehls mit dem gleichen gegebenen Opcode kann dem Dekodierer **705** von dem zweiten Softwaremodul **711-2** geliefert werden. Das zweite Softwaremodul enthält die Anzeige **773** zum Verwenden der zweiten Bedeutung für den gegebenen Opcode, den das Modul **771** an dem Speicherplatz **774** erhalten kann. Die Prüflogik **776** kann die Anzeige **775** prüfen und bestimmen, dass die zweite Bedeutung für den gegebenen Opcode verwendet werden soll, wenn die zweite Instanz des Befehls von dem zweiten Softwaremodul verarbeitet wird. Als Antwort kann die Emulationsauslöselogik **778** Emulation der zweiten Instanz des Befehls **703-2** auslösen. Beispielsweise kann die Emulationsauslöselogik ein Emulationsfangsignal durchführen oder auf andere Weise einen Emulationsmodus **718** signalisieren. Ein Satz von einem oder mehreren Befehlen **714**, die verwendet werden, um die zweite Instanz des Befehls, der den gegebenen Opcode mit der zweiten Bedeutung aufweist, zu emulieren, kann an den Dekodierer von einer Emulationslogik **715** geliefert werden. Die Emulationslogik kann auf dem Chip, außerhalb des Chips oder teilweise auf dem Chip und teilweise außerhalb des Chips vorliegen. Die Emulationslogik **715** kann jede der Charakteristiken aufweisen, die hier an anderer Stelle für Emulationslogik beschrieben ist.

[0084] In einigen Ausführungsformen kann/können der/die Befehl(e) **714** aus dem gleichen Befehlssatz wie der Befehl, der den gegebenen Opcode aufweist, stammen. In einigen Ausführungsformen kann der Dekodierer jeden dieser Befehle dekodieren und sie als dekodierte Befehle **706** oder andere Steuersignale an die Nachdekodier-Befehlsprozessorlogik liefern. In einigen Ausführungsformen kann die Nachdekodier-Befehlsprozessorlogik eine emulationsmodusbewusste Befehlsprozessorlogik **720** enthalten, die ähnlich oder identisch hier an anderer Stelle beschrieben sein kann (beispielsweise diejenigen aus einer der **Fig.** 1 oder **Fig.** 3 bis **Fig.** 5). Wie gezeigt, kann die emulationsmodusbewusste Befehlsprozessorlogik mit dem Emulationsmodus **718** gekoppelt sein oder diesen auf andere Weise kennen. Darüber hinaus kann die emulationsmodusbewusste Befehlsprozessorlogik mit Speicherplätzen **721** der Emulationslogik gekoppelt sein und Daten aus diesen lesen und in diese schreiben.

[0085] In einigen Ausführungsformen kann die Logik **796** enthalten sein, um ein Prozessormerkmalsidentifizierungregister **795** anhand der Anzeige **775** an dem Speicherplatz **774** zu aktualisieren. Ein Beispiel eines geeigneten Prozessormerkmalsidentifizierungsregister ist ein für CPU IDentification (CPUID) verwendetes. Die Logik **796** kann mit dem Speicherplatz **774** und mit dem Prozessormerkmalsidentifizierungsregister **795** gekoppelt sein. Das Prozessormerkmalsidentifizierungsregister kann durch einen Prozessormerkmalsidentifizierungsbefehl (beispielsweise einen CPUID-Befehl) eines Befehlssatzes des Prozessors lesbar sein. Software kann die Anzeige der Bedeutung des Opcodes aus dem Prozessormerkmalsidentifizierungsregister durch Ausführen des Prozessoridentifizierungsbefehls lesen.

[0086] In einigen Ausführungsformen kann eine Berechtigungsstufen- und/oder Ringstufenlogik **794** mit dem Dekodierer **705** gekoppelt sein, und kann den Dekodierer zwingen oder auf andere Weise veranlassen, eine gegebene Bedeutung des Opcodes anhand einer Berechtigungsstufe und/oder Ringstufe zu verwenden. Beispielsweise kann dies in Ausführungsformen nützlich sein, in denen die erste Bedeutung eine neuere Bedeutung und die zweite Bedeutung eine veraltete Bedeutung ist. Betriebssysteme arbeiten typischerweise auf einer bestimmten Berechtigungsstufe und/oder Ringstufe, die sich von derjenigen von Benutzeranwendungen unterscheidet. Darüber hinaus verwenden Betriebssysteme typischerweise die neuere Bedeutung des gegebenen Opcodes und nicht die ältere Bedeutung des gegebenen Opcodes, weil sie im Allgemeinen häufig aktualisiert werden. In solchen Fällen kann die Berechtigungsstufen- und/oder Ringstufenlogik **794** den Dekodierer veranlassen, die neuere Bedeutung des gegebenen Opcodes zu verwenden, wenn eine Berechtigungs- oder Ringstufe vorliegt, die der des Betriebssystems entspricht.

[0087] Zur Vereinfachung der Beschreibung werden hier typischerweise zwei unterschiedliche Bedeutungen des Opcodes beschrieben. Jedoch versteht sich, dass andere Ausführungsformen drei oder mehr unterschiedliche Bedeutungen für einen gegebenen Opcode verwenden können. Als Beispiel kann der Speicherplatz **774** zwei oder mehr Bits aufweisen, um anzuzeigen, welche der mehreren solchen unterschiedlichen Bedeutungen für einen gegebenen Opcode verwendet werden sollen. Auf ähnliche Weise kann das Prozessoridentifizierungsregister mehrere solche Bedeutungen für den gegebenen Opcode wiedergeben.

[0088] Fig. 8 zeigt ein Blockflussdiagramm einer Ausführungsform eines Verfahrens **880**, das von einem Betriebssystemmodul durchgeführt werden kann. In einigen Ausführungsformen kann das Verfahren durch ein Programmladermodul durchgeführt werden.

[0089] Das Verfahren schließt bei Block **881** Bestimmen ein, dass ein erster Befehl, der einen gegebenen Opcode aufweist, eine zweite Bedeutung anstatt einer ersten Bedeutung aufweisen soll, wenn er durch einen Prozessor von einem Softwareprogramm ausgeführt wird. Dies kann in verschiedenen Ausführungsformen auf unterschiedliche Arten vorgenommen werden. In einigen Ausführungsformen kann das Softwareprogramm explizit eine Anzeige zum Verwenden einer gegebenen Bedeutung für den gegebenen Opcode angeben. Beispielsweise kann das Betriebssystemmodul Metadaten des Softwareprogramms prüfen. Beispielsweise kann ein Flag in einem Objektmodulformat vorliegen, das angibt, welche Bedeutung verwendet werden soll. In anderen Ausführungsformen, beispielsweise im Falle überlieferter Software, kann das Softwareprogramm nicht explizit die Anzeige angeben, welche Bedeutung verwendet werden soll. In einigen Ausführungsformen kann das Betriebssystemmodul Logik enthalten, um zu ermitteln, welche Bedeutung verwendet werden soll. Dies kann auf verschiedene unterschiedliche Arten vorgenommen werden. In einigen Ausführungsformen kann dies Prüfen einer Merkmalliste des Softwareprogramms einschließen. In einigen Fällen kann die Merkmalliste angeben, welche Befehlsrevision erwartet wird. In einigen Ausführungsformen kann dies Prüfen eines Erstellungsdatums des Softwareprogramms einschließen. Ein Erstellungsdatum, das älter als ein bestimmtes Datum ist, beispielsweise ein Befehlsdatum einer neueren Ersatzbedeutung, kann als eine Anzeige ermittelt werden, dass das Softwareprogramm die ältere oder veraltete Bedeutung verwendet. In einigen Ausführungsformen kann dies Prüfen eines Formats des Softwareprogramms einschließen. Beispielsweise können bestimmte Revisionsprogrammformate vor einer bestimmten Stufe verwendet werden, um eine ältere oder veraltete Bedeutung zu ermitteln. In einigen Ausführungsformen kann dies Prüfen einer expliziten Liste (beispielsweise einer Ausnahmeliste) von Softwareprogrammen einschließen, von denen bekannt ist, dass sie bestimmte Bedeutungen verwenden. Als Beispiel kann die Liste anhand Vergangenheitsinformation (beispielsweise falls ein Fehler aus einer Bedeutung resultiert, kann der Liste die andere Bedeutung hinzugefügt werden) aktualisiert werden. Dies ist lediglich ein Beispiel. Andere Arten zum Ermitteln der Bedeutung können ebenfalls in Betracht gezogen werden.

[0090] Das Verfahren schließt bei Block **882** außerdem Speichern einer Anzeige ein, dass der erste Befehl, der den gegebenen Opcode aufweist, die zweite Bedeutung anstatt der ersten Bedeutung im Zustand eines Prozessors aufweist. Beispielsweise kann das Betriebssystemmodul ein Bit an einem Speicherplatz modifizieren, der mit einem Dekodierer gekoppelt ist, wie hier an anderer Stelle beschrieben ist.

[0091] Fig. 9 zeigt ein Blockdiagramm einer Ausführungsform eines Programmladermoduls **970** einschließlich eines Selektionsmoduls **985**, das betrieben werden kann, einen Satz von ein oder mehreren Funktionen, Subroutinen oder andere Teile einer Softwarebibliothek **983** auszuwählen, die eine Bedeutung eines gegebenen Opcodes aufweisen kann, die für Software, die sie verwenden wird, angemessen ist. Die Softwarebibliothek stellt allgemein eine Sammlung von Software dar, die verschiedene Softwaremodule verwenden können, und kann bereits existierende Software in der Form von Subroutinen, Funktionen, Klassen, Prozeduren, Skripten, Konfigurationsdaten und ähnlichen enthalten. Softwaremodule können diese verschiedenen Teile der Bibliothek verwenden, um verschiedene Funktionalitäten aufzuweisen. Als ein Beispiel kann ein Softwaremodul eine Mathematiksoftwarebibliothek oder einen Teil davon integrieren, der verschiedene mathematische Funktionen oder Subroutinen aufweist.

[0092] Wie gezeigt, kann die Bibliothek in einigen Ausführungsformen einen ersten Satz von Bibliotheksfunktionen, -Subroutinen oder andere Teile enthalten, die eine erste Bedeutung eines gegebenen Opcodes verwenden. Die Bibliothek kann auch einen zweiten Satz von Bibliotheksfunktionen, -Subroutinen oder andere Teile enthalten, die eine zweite, andere Bedeutung des gegebenen Opcodes verwenden. Optional kann, falls mehr als zwei Bedeutungen des Opcodes vorliegen, in ähnlicher Form unterschiedliche Teile der Bibliothek für jede der drei oder mehr unterschiedlichen Bedeutungen vorliegen. In einigen Fällen können die Teile, die die unterschiedlichen Bedeutungen verwenden, unterschiedliche Codeteile sein. In anderen Fällen können die Teile unterschiedliche Teile des gleichen Codes sein, und können Verzweigungen oder bedingte Sprunganweisungen verwendet werden, um in geeigneter Weise zu demjenigen Teil, der die erste Bedeutung oder die zweite Bedeutung verwendet, zu springen.

[0093] Unter erneuter Bezugnahme auf die Zeichnung kann das Programmladermodul **970** Teile der Bibliothek sowohl für ein erstes Softwaremodul **911-1**, das eine erste Bedeutung des gegebenen Opcode verwendet, als auch für ein zweites Softwaremodul **911-2**, das eine zweite Bedeutung des gegebenen Opcode verwendet, laden. Das Programmladermodul enthält ein Selektionsmodul **985**, das betrieben werden kann, einen Satz von ein oder mehreren Funktionen, Subroutinen oder andere Teilen der Softwarebibliothek auszuwählen, die eine Bedeutung des gegebenen Opcodes aufweisen, die für die Soft-

ware, die sie verwenden wird, geeignet ist. Beispielsweise kann das Selektionsmodul Teile der Bibliothek, die die gleiche Bedeutung des gegebenen Opcode aufweisen wie die Software, die sie verwenden wird, auswählen. Beispielsweise kann das Selektionsmodul, wie in der Zeichnung gezeigt, den ersten Satz **984-1** für das erste Softwaremodul **911-1** auswählen, weil es die erste Bedeutung des gegebenen Opcodes verwendet. Auf ähnliche Weise kann das Selektionsmodul den zweiten Satz **984-2** für das zweite Softwaremodul **911-2** auswählen, weil es die zweite Bedeutung für den gegebenen Opcode verwendet. In einer bestimmten Ausführungsform, in der die erste Software **911-1** alte Software und die erste Bedeutung des gegebenen Opcodes eine veraltete Bedeutung ist, kann das Selektionsmodul betrieben werden, um den ersten Satz von Bibliotheksteilen **984** auszuwählen, der ebenfalls die gleiche veraltete Bedeutung für den gegebenen Opcode verwendet. Somit kann das Selektionsmodul Teile einer Bibliothek auswählen, die eine Bedeutung eines gegebenen Opcodes verwenden, die mit der Software, die diesen Teil der Bibliothek verwenden wird, konsistent ist oder mit ihr identisch ist.

[0094] Beispielhafte Kernarchitekturen, Prozessoren und Computerarchitekturen Prozessorkerne können auf verschiedene Arten, zu verschiedenen Zwecken und in unterschiedlichen Prozessoren implementiert sein. Beispielsweise können Implementationen solcher Kerne einschließen: 1) einen geordneten Vielzweckkern, der für allgemeines Rechnen vorgesehen ist; 2) einen ungeordneten Hochleistungsvielzweckkern, der für allgemeines Rechnen vorgesehen ist; 3) einen ungeordneten Spezialkern, der vorrangig für Grafik und/oder wissenschaftliches (Durchsatz-)Rechnen vorgesehen ist. Implementationen unterschiedlicher Prozessoren können einschließen: 1) eine CPU, die einen oder mehrere geordnete Vielzweckkerne enthält, die für allgemeines Rechnen vorgesehen sind, und/oder einen oder mehrere ungeordnete Vielzweckkerne, die für allgemeines Rechnen vorgesehen sind; und 2) einen Koprozessor, der einen oder mehrere Spezialkerne enthält, die vorrangig für Grafik- und/oder wissenschaftliches (Durchsatz-)Rechnen vorgesehen sind. Solche unterschiedlichen Prozessoren führen zu unterschiedlichen Computersystemarchitekturen, die enthalten können: 1) den Koprozessor auf einem von der CPU separaten Chip; 2) den Koprozessor auf einem separaten Chip in dem gleichen Package wie eine CPU; 3) den Koprozessor auf dem gleichen Chip wie eine CPU (in welchem Fall ein solcher Koprozessor manchmal als eine Speziallogik bezeichnet wird, wie etwa integrierte Grafik- und/oder wissenschaftliche (Durchsatz-)Logik oder als Spezialkerne) ; und 4) ein System auf dem Chip, das auf dem gleichen Chip die beschriebene CPU (manchmal als der/die Anwendungskern(e) oder Anwendungsprozessor (en) bezeichnet), den oben beschriebenen Koprozessor und weitere Funktionalität enthalten kann. Beispielhafte Kernarchitekturen werden als nächstes beschrieben, gefolgt von Beschreibungen beispielhafter Prozessoren und Computerarchitekturen.

Beispielhafte Kernarchitekturen

Blockdiagramm geordneter und ungeordneter Kerne

[0095] Fig. 10A zeigt ein Blockdiagramm, das sowohl eine beispielhafte geordnete Pipeline als auch eine beispielhafte ungeordnete Ausgabe-/Ausführungs-Registerumbenennungspipeline gemäß Ausführungsformen der Erfindung zeigt. **Fig.** 10B zeigt ein Blockdiagramm, dass sowohl eine beispielhafte Ausführungsform eines geordneten Architekturkerns als auch einen beispielhaften ungeordneten Ausgabe-/Ausführungs-Registerumbenennungsarchitekturkern gemäß Ausführungsformen der Erfindung, der/die in einem Prozessor enthalten sein soll (en), zeigt. Die durchgehend umrandeten Kästen in **Fig.** 10A–B zeigen die geordnete Pipeline und geordneten Kern, während das optionale Hinzufügen der gestrichelten Kästen die ungeordnete Ausgabe-/Ausführungs Registerumbenennungspipeline und -kern zeigt. Vorausgesetzt, dass der geordnete Aspekt eine Teilmenge des ungeordneten Aspekts ist, wird der ungeordnete Aspekt beschrieben.

[0096] In **Fig.** 10A enthält eine Prozessorpipeline **1000** eine Einholstufe **1002**, eine Längendekodierstufe **1004**, eine Dekodierstufe **1006**, eine Zuweisungsstufe **1008**, eine Umbenennungsstufe **1010**, eine Planungs(auch als Auslösung oder Ausgabe bekannt)-stufe **1012**, eine Registerlese-/Speicherlesestufe **1014**, eine Ausführungsstufe **1016**, eine Rückschreib-/Speicherschreibstufe **1018**, eine Ausnahmebehandlungsstufe **1022** und eine Überweisungsstufe **1024**.

[0097] Fig. 10B zeigt einen Prozessorkern **1090**, der eine vordere Einheit **1030** enthält, die mit einer Ausführungseinheit **1050** gekoppelt ist, und beide sind mit einer Speichereinheit **1070** gekoppelt. Der Kern **1090** kann ein Kern mit reduziertem Befehlssatz (reduced instruction set computing (RISC)), ein Kern mit komplexem Befehlssatz (complex instruction set computing (CISC)), ein Kern mit sehr langem Befehlswort (very long instruction word (VLIW)) oder ein hybrider oder alternativer Kerntyp sein. Als weitere Option kann der Kern **1090** ein Vielzweckkern, wie etwa beispielsweise ein Netzwerk- oder Kommunikationskern, Kompressionseinheit, Koprozessorkern, Vielzweckgrafikprozessoreinheits(general purpose computing graphics processing unit (GPGPU)-Kern, Grafikkern oder ähnliches sein.

[0098] Die vordere Einheit **1030** enthält eine Verzweigungsvorhersageeinheit **1032**, die mit einer Be-

fehlscacheeinheit **1034** gekoppelt ist, die mit einem Befehlsübersetzungspuffer (translation lookaside buffer (TLB)) **1036** gekoppelt ist, der mit einer Befehlseinholeinheit **1038** gekoppelt ist, die mit einer Dekodiereinheit **1040** gekoppelt ist. Die Dekodiereinheit **1040** (oder Dekodierer) kann Befehle dekodieren und als eine Ausgabe eine oder mehrere Mikrooperationen, Mikrocode-Zugangspunkte, Mikrobefehle, andere Befehle oder andere Steuersignale erzeugen, die aus den ursprünglichen Befehlen dekodiert werden oder diese auf andere Weise wiedergeben oder aus diesen abgeleitet sind. Die Dekodiereinheit **1040** kann unter Verwendung verschiedener unterschiedlicher Mechanismen implementiert sein. Beispiele geeigneter Mechanismen schließen Wertetabellen, Hardwareimplementationen, programmierbare Logikarrays (PLAs), Mikrocode-Nurlesespeicher (read only memories (ROMs)) etc. ein. In einer Ausführungsform enthält der Kern **1090** ein Mikrocode-ROM oder anderes Medium, das Mikrocode für bestimmte Makrobefehle (beispielsweise in der Dekodiereinheit **1040** oder andernfalls innerhalb der vorderen Einheit **1030**) speichert. Die Dekodiereinheit **11040** ist mit einer Umbenennungs-/Zuweisungseinheit **1052** in der Ausführungseinheit **1050** gekoppelt.

[0099] Die Ausführungseinheit **1050** enthält die Umbenennungs-/Zuweisungseinheit **1052**, die mit einer Rückzugseinheit **1054** und einem Satz einer oder mehrerer Planungseinheit(en) **1056** gekoppelt ist. Die Planungseinheit(en) **1056** stellt/stellen irgendeine Anzahl unterschiedlicher Planer dar, einschließlich Reservierungsstationen, zentralem Befehlsfenster etc. Der/die Planungseinheit(en) **1056** ist/sind mit den/der physikalischen Registerdatei(en)einheit(en) **1058** gekoppelt. Jede der physikalischen Registerdateieinheiten **1058** stellt eine oder mehrere physikalische Registerdateien dar, von denen verschiedene einen oder mehrere unterschiedliche Datentypen speichern, etwa als skalare ganze Zahl, skalare Fließkommazahl, gepackte ganze Zahl, gepackte Fließkommazahl, Ganzzahlvektor, Fließkommavektor, Status (beispielsweise einen Befehlszeiger, der die Adresse des nächsten Befehls ist, der ausgeführt werden soll), etc. In einer Ausführungsform umfasst die physikalische Registerdatei(en)einheit **1058** eine Vektorregistereinheit, eine Schreibmaskenregistereinheit und eine Skalarregistereinheit. Diese Registereinheiten können architektonische Vektorregister, Vektormaskenregister und Vielzweckregister bereitstellen. Die physikalische(n) Registerdatei(en)einheit(en) **1058** wird durch die Rückzugseinheit **1054** überlappt, um verschiedene Wege zu zeigen, auf denen Registerumbenennung und ungeordnete Ausführungsform implementiert sein kann (beispielsweise unter Verwendung eines Umordnungspuffers und/von Umordungspuffern (einer) Rückzugsregisterdatei(en); unter Verwendung einer Zukunftsdatei/von Zukunftsdateien, eines Vergangenheitspuffers/von Vergangenheitspuffern und (einer) Rückzugsregisterdatei(en); unter Verwendung einer Registerkarte und eines Pools von Registern etc.). Die Rückzugseinheit **1054** und die physikalische Registerdatei(en)einheit(en) **1058** sind mit dem/den Ausführungscluster(n) **1060** gekoppelt. Das/die Ausführungscluster **1060** enthält/enthalten einen Satz einer oder mehrerer Ausführungseinheiten **1062** und einen Satz einer oder mehrerer Speicherzugriffseinheiten **1065**. Die Ausführungseinheiten **1062** können verschiedene Operationen (beispielsweise Shifts, Addition, Subtraktion, Multiplikation) auf verschiedenen Datentypen (beispielsweise skalare Fließkommazahl, gepackte ganze Zahl, Ganzzahlvektor, Fließkommavektor) durchführen. Während einige Ausführungsformen eine Anzahl von Ausführungseinheiten einschließen, die für spezifische Funktionen oder Sätze von Funktionen gestaltet sind, können andere Ausführungseinheiten lediglich eine Ausführungsform oder mehrere Ausführungsformen einschließen, die alle sämtliche Funktionen durchführen. Die Planungseinheit(en) **1056**, physikalischen Registerdatei(en)einheit(en) **1058** und Ausführungscluster **1060** sind als möglicherweise in der Mehrzahl gezeigt, weil bestimmte Ausführungsformen separate Pipelines für bestimmte Datentypen/Operationen erstellen (beispielsweise eine skalare Ganzzahlpipeline, eine skalare Fließkomma-/gepackte Ganzzahl-/gepackte Fließkomma-/Ganzzahlvektor-/Fließkommavektorpipeline und/oder eine Speicherzugriffspipeline, die jeweils ihre eigene Planungseinheit, physikalische Registerdatei(en)einheit und/oder Ausführungscluster aufweisen – und im Fall einer separaten Speicherzugriffpipeline sind bestimmte Ausführungsformen implementiert, in denen lediglich das Ausführungscluster dieser Pipeline die Speicherzugriffseinheit(en) **1064** aufweist). Es versteht sich außerdem, dass, wenn separate Pipelines verwendet werden, eine oder mehrere dieser Pipelines mit ungeordneter Ausgabe/Ausführung gestaltet sein können und der Rest geordnet ist.

[0100] Der Satz von Speicherzugriffseinheiten **1064** ist mit der Speichereinheit **1070** gekoppelt, die eine Daten-TLB-Einheit **1072** enthält, die mit einer Datencacheeinheit **1074** gekoppelt ist, die mit einer Level-2(L2)-Cacheeinheit **1076** gekoppelt ist. In einer beispielhaften Ausführungsform können die Speicherzugriffseinheiten **1064** eine Ladeeinheit, eine Adressspeichereinheit und eine Datenspeichereinheit enthalten, von denen jede mit der Daten-TLB-Einheit **1072** in der Speichereinheit **1070** gekoppelt ist. Die Befehlscacheeinheit **1034** ist ferner mit einer Level-2 (L2)-Cacheeinheit **1076** in der Speichereinheit **1070** gekoppelt. Die L2-Cacheeinheit **1076** ist mit einem oder mehreren Cachelevels und letztlich mit einem Hauptspeicher gekoppelt.

[0101] Als Beispiel kann die beispielhafte ungeordnete Ausgabe-/Ausführungs-Registerumbenennungs-Kernarchitektur die Pipeline **1000** wie folgt im-

plementieren: 1) die Befehlseinholeinheit **1038** führt das Einholen und die Längendekodierstufen **1002** und **1004** durch; 2) die Dekodiereinheit **1040** führt die Dekodierstufe **1006** durch; 3) die Umbenennungs-/Zuweisungseinheit **1052** fährt die Zuweisungsstufe **1008** und Umbenennungsstufe **1010** durch; 4) die Planungseinheit(en) **1056** führt/führen die Planungsstufe **1012** durch; 5) die physikalischen Registerdatei(en)einheit(en) **1058** und die Speichereinheit **1070** führen die Registerlese-/Speicherlesestufe **1014** durch; das Ausführungscluster **1060** führt die Ausführungsstufe **1016** durch; 6) die Speichereinheit **1070** und die physikalische (n)Registerdatei(en) einheit(en) **1058** führt/führen die Rückschreib-/Speicherschreibstufe **1018** durch; 7) verschiedene Einheiten können an der Ausnahmebehandlungsstufe **1022** beteiligt sein; und 8) die Rückzugseinheit **1054** und die physikalische(n) Registerdatei(en)einheit(en) **1058** führen die Überweisungsstufe **1024** durch.

[0102] Der Kern **1090** kann einen oder mehrere Befehlssätze (beispielsweise den x86-Befehlssatz (mit einigen Erweiterungen, die in neueren Versionen hinzugefügt wurden); den MIPS-Befehlssatz von MIPS-Technologies aus Sunnyvale, CA; den ARM-Befehlssatz (mit optionalen zusätzlichen Erweiterungen, wie etwa NEON) von ARM-Holdings aus Sunnyvale, CA), die den/die Befehl(e) enthalten, der/die hier beschrieben sind/ist. In einer Ausführungsform enthält der Kern **1090** Logik, um eine Befehlssatzerweiterung gepackter Daten (beispielsweise AVX1, AVX2) zu unterstützen, so dass den Operationen, die von vielen Multimediaanwendungen verwendet werden, ermöglicht wird, unter Verwendung gepackter Daten durchgeführt zu werden.

[0103] Es versteht sich, dass der Kern Multithreading (Ausführen von zwei oder mehreren Sätzen von Operationen oder Threads) unterstützen kann, und dies auf eine Vielzahl von Arten vornehmen kann, einschließlich zeitlich geteiltes Multithreading, simultanes Multitheading (wenn ein einzelner physikalischer Kern einen logischen Kern für jeden der Threads bereitstellt, führt der Prozessor simultan Multithreading durch) oder eine Kombination derselben (beispielsweise zeitlich geteiltes Einholen und Dekodieren und anschließendes simultanes Multithreading, wie etwa in der Intel Hyperthreading-Technologie).

[0104] Während Registerumbenennen im Kontext ungeordneter Ausführung beschrieben wird, versteht sich, dass Registerumbenennen in einer geordneten Architektur verwendet werden kann. Während die gezeigten Ausführungsformen des Prozessors auch separate Befehls- und Datencache-Einheiten **1034**/**1074** und eine gemeinsam benutzte L2-Cacheeinheit **1076** einschließen, können alternative Ausführungsformen einen einzelnen internen Cache sowohl für Befehle als auch Daten einschließen, wie etwa beispielsweise einen internen Level-1(L1)-Cache oder mehrere Level von internem Cache. In einigen Ausführungsformen kann das System eine Kombination eines internen Caches und eines externen Caches aufweisen, die außerhalb des Kerns und/oder des Prozessors ist. Alternativ kann der gesamte Cache außerhalb des Kerns und/oder des Prozessors sein.

Spezifische beispielhafte geordnete Kernarchitektur

[0105] Fig. 11A–B zeigen ein Blockdiagramm einer konkreten beispielhaften geordneten Kernarchitektur, deren Kern einer von mehreren Logikblöcken (einschließlich anderer Kerne des gleichen Typs und/oder unterschiedlicher Typen) in einem Chip wäre. Die Logikblöcke kommunizieren durch ein Schaltnetzwerk hoher Bandbreite (beispielsweise ein Ringnetzwerk) mit irgendeiner festen Funktionslogik, Speicher-I/O-Schnittstellen und anderer notwendiger I/O-Logik, abhängig von der Anwendung.

[0106] Fig. 11A zeigt ein Blockdiagramm eines einzelnen Prozessorkerns, gemeinsam mit seiner Verbindung zu dem Schaltnetzwerk **1102** auf dem Chip und mit seinem lokalen Teil des Level 2(L2)-Caches **1104** gemäß Ausführungsformen der Erfindung. In einer Ausführungsform unterstützt der Befehlsdekodierer **1100** den x86-Befehlssatz mit einer Befehlssatzerweiterung gepackter Daten. Ein L1-Cache **1106** ermöglicht Zugriffe mit geringer Latenz auf Cachespeicher in den Skalar- und Vektoreinheiten. Während in einer Ausführungsform (um das Design zu vereinfachen) eine Skalareinheit **1108** und eine Vektoreinheit **1110** separate Registersätze (jeweils skalare Register **1112** und Vektorregister **1114**) verwenden, und Daten, die zwischen diesen übertragen werden, in Speicher geschrieben und dann aus einem Level 1 (L1)-Cache **1106** gelesen werden, können alternative Ausführungsformen der Erfindung einen anderen Ansatz verwenden (beispielsweise einen Einzelregistersatz verwenden oder einen Kommunikationspfad einschließen, der es Daten ermöglicht, zwischen den beiden Registerdateien übertragen zu werden, ohne geschrieben und zurückgelesen zu werden).

[0107] Der lokale Teil des L2-Caches **1104** ist Teil eines globalen L2-Caches, der in separate lokale Teile unterteilt ist, einer je Prozessorkern. Jeder Prozessorkern weist einen direkten Zugriffspfad zu seinem eigenen lokalen Teil des L2-Caches **1104** auf. Daten, die von einem Prozessor gelesen werden, werden in seinem L2-Cache-Teil gespeichert und können schnell, parallel zu anderen Prozessorkernen zugegriffen werden, die auf ihre eigenen lokalen L2-Cache-Teile zugreifen. Daten, die von einem Prozessorkern geschrieben werden, werden in dessen eigenem L2-Cache-Teil **1104** gespeichert und von anderen Teilen gelöscht, falls notwendig. Das Ringnetzwerk gewährleistet Kohärenz für gemeinsam benutzte Daten. Das Ringnetzwerk ist bidirektional, um Agenten,

wie etwa Prozessorkernen, L2-Caches und anderen Logikblöcken zu ermöglichen, miteinander innerhalb des Chips zu kommunizieren. Jeder Ringdatenpfad ist je Richtung **1012** Bits breit.

[0108] Fig. 11b zeigt eine erweiterte Ansicht eines Teils des Prozessorkerns aus **Fig.** 11A gemäß Ausführungsformen der Erfindung. **Fig.** 11B weist einen L1-Datencache **1106A** -Teil des L1-Caches **1104**, ebenso wie weitere Details betreffend die Vektoreinheit **1110** und die Vektorregister **1114** auf. Insbesondere ist die Vektoreinheit **1110** eine 16 Bit breite Vektorverarbeitungseinheit (vector processing unit (VPU)) (siehe die 16 Bit breite ALU **1128**), die einen oder mehrere Befehle mit ganzen Zahlen, Fließkommazahlen mit einfacher Genauigkeit und Fließkommazahlen mit doppelter Genauigkeit ausführt. Die VPU unterstützt Vermischen (swizzling) der Registereingaben mit Swizzle-Einheit **1120**, numerische Umwandlung mit numerischen Umwandlungseinheiten **1122A**–B und Replikation mit einer Replikationseinheit **1124** auf der Speichereingabe. Schreibmaskenregister **1126** ermöglichen Vorhersagen resultierender Vektorschreiboperationen.

Prozessor mit integriertem Speichercontroller und Grafik

[0109] Fig. 12 zeigt ein Blockdiagramm eines Prozessors **1200**, der gemäß Ausführungsformen mehr als einen Kern, einen integrierten Speichercontroller und integrierte Grafik enthalten kann. Die durchgehend umrandeten Kästen in **Fig.** 12 zeigen einen Prozessor **1200** mit einem einzelnen Kern **1202A**, einem Systemagenten **1210**, einem Satz von einem oder mehreren Buscontrollereinheiten **1216**, während die optionale Zugabe der gestrichelt umrandeten Kästen einen alternativen Prozessor **1200** mit mehreren Kernen **1202A**–N, einen Satz von einer oder mehreren integrierten Speichercontrollereinheit(en) **1214** in der Systemagenteneinheit **1210** und Speziallogik **1208** zeigt.

[0110] Somit können unterschiedliche Implementationen des Prozessors **1200** einschließen: 1) eine CPU, bei der die Speziallogik **1208** eine integrierte Grafik- und/oder Wissenschafts (Durchsatz-)Logik (die einen oder mehrere Kerne enthalten kann) ist und, die Kerne **1202A**–N ein oder mehrere Vielzweckkerne sind (beispielsweise geordnete Vielzweckkerne, ungeordnete Vielzweckkerne, eine Kombination von beiden); 2) einen Koprozessor, bei dem die Kerne **1202A**–N eine große Anzahl von Spezialkernen sind, die primär für Grafik- und/oder wissenschaftliche Anwendungen dienen (Durchsatz); und 3) einen Koprozessor, bei dem die Kerne **1202A**–N eine große Anzahl geordneter Vielzweckkerne sind. Somit kann der Prozessor **1200** ein Vielzweckprozessor, Koprozessor oder Spezialprozessor sein, wie etwa beispielsweise ein Netzwerk- oder Kommunikationsprozessor, Kompressionseinheit, Grafikprozessor, GPGPU (general purpose graphics processing unit), ein Hochdurchsatzkoprozessor mit vielen integrierten Kernen (many integrated cores (MIC)) (einschließlich 30 oder mehr Kerne), eingebetteter Prozessor oder ähnliche. Der Prozessor kann auf einem oder mehreren Chips implementiert sein. Der Prozessor **1200** kann ein Teil von und/oder auf einem oder mehreren Substraten unter Verwendung einer Anzahl von Prozessortechnologien implementiert sein, wie etwa beispielsweise BiCMOS, CMOS oder NMOS.

[0111] Die Speicherhierarchie enthält einen oder mehrere Cachelevel innerhalb der Kerne, einen Satz von einem oder mehreren gemeinsam benutzten Cacheeinheiten **1206** und externen Speicher (nicht gezeigt), der mit dem Satz integrierter Speichercontrollereinheiten **1214** gekoppelt ist. Der Satz gemeinsam benutzter Cacheeinheiten **1206** kann einen oder mehrere Caches mittleren Levels, wie etwa Level 2 (L2), Level 3 (L3), Level 4 (L4) oder andere Cachelevel, einen Last-Level-Cache (LLC) und/oder Kombinationen derselben enthalten. Während in einer Ausführungsform eine ringbasierte Schaltverbindungseinheit **1212** die integrierte Grafiklogik **1208**, den Satz gemeinsam benutzter Cacheeinheiten **1206** und die Systemagenteinheit **1210**/integrierte Speichercontrollereinheit(en) **1214** miteinander verbindet, können alternative Ausführungsformen irgendeine Anzahl wohlbekannter Techniken zum Verbinden solcher Einheiten verwenden. In einer Ausführungsform wird Kohärenz zwischen einer oder mehreren solchen Cacheeinheiten **1206** und Kernen **1202A**–N aufrechterhalten.

[0112] In einigen Ausführungsformen sind ein oder mehrere der Kerne **1202A**–N Multithreadingfähig. Der Systemagent **1210** enthält diejenigen Komponenten, die die Kerne **1202A**–N koordinieren und betreiben. Die Systemagenteinheit **1210** kann beispielsweise eine Energiesteuereinheit (power control unit (PCU)) und eine Anzeigeinheit enthalten. Die PCU kann Logik und Komponenten sein oder enthalten, die zum Regulieren des Energiezustands der Kerne **1202a**–N und der integrierten Grafiklogik **1208** benötigt werden. Die Anzeigeinheit dient dem Steuern einer oder mehrerer extern verbundener Anzeigen.

[0113] Die Kerne **1202A**–N können hinsichtlich Architekturbefehlssatz homogen oder heterogen sein; das heißt, zwei oder mehrere der Kerne **1202A**–N können in der Lage sein, den gleichen Befehlssatz auszuführen, während andere in der Lage sein können, lediglich eine Teilmenge dieses Befehlssatzes oder eines anderen Befehlssatzes auszuführen.

Beispielhafte Computerarchitekturen

[0114] **Fig.** 13–**Fig.** 16 zeigen Blockdiagramme beispielhafter Computerarchitekturen. Andere Systemdesigns und -konfigurationen, die aus dem Stand der Technik für Laptops, Desktops, Hand-PCs, Personal Digital Assistants, Ingenieur-Workstations, Server, Netzwerkgeräte, Netzwerk-Hubs, Switches, eingebettete Prozessoren, digitale Signalprozessoren (DSPs), Grafikgeräte, Videospielgeräte, Set-top-Boxen, Mikrocontroller, Mobiltelefone, tragbare Medienabspielgeräte, Handgeräte und verschiedene andere elektronische Geräte bekannt sind, sind ebenfalls geeignet. Allgemein ist eine große Vielzahl von Systemen oder elektronischen Geräten, die in der Lage ist, einen Prozessor und/oder andere Ausführungslogik, wie hier offenbart, aufzunehmen, allgemein geeignet.

[0115] Unter Bezugnahme auf **Fig.** 13 ist ein Blockdiagramm eines Systems **1300** gemäß einer Ausführungsform der vorliegenden Erfindung gezeigt. Das System **1300** kann einen oder mehrere Prozessoren **1310**, **1315** enthalten, die mit einem Controller-Hub **1320** gekoppelt sind. In einer Ausführungsform enthält der Controller-Hub **1320** einen Grafikspeichercontroller-Hub (graphics memory controller hub (GMCH)) **1390** und einen Eingabe-/Ausgabe-Hub (Input/Output Hub (IOH)) **1350** (die auf separaten Chips vorliegen können); der GMCH **1390** enthält Speicher- und Grafikcontroller, mit denen Speicher **1340** und ein Koprozessor **1345** gekoppelt ist; der IOH **1350** koppelt Eingabe-/Ausgabe-(I/O)-Geräte **1360** an den GMCH **1390**. Alternativ sind ein oder beide Speicher- und Grafikcontroller innerhalb des Prozessors (wie hier beschrieben) integriert, sind der Speicher **1340** und der Koprozessor **1345** direkt mit dem Prozessor **1310** gekoppelt, und ist der Controller-Hub **1320** in einem einzelnen Chip mit dem IHO **1350**.

[0116] Der optionale Charakter der zusätzlichen Prozessoren **1315** ist in **Fig.** 13 mit durchbrochenen Linien bezeichnet. Jeder Prozessor **1310**, **1315** kann einen oder mehrere der hier beschriebenen Prozessorkerne enthalten und irgendeine Version des Prozessors **1200** sein.

[0117] Der Speicher **1340** kann beispielsweise dynamischer Speicher mit wahlfreiem Zugriff (dynamic random access memory (DRAM)), Phasenwechselspeicher (phase change memory (PCM)) oder eine Kombination der beiden sein. In wenigstens einer Ausführungsform kommuniziert der Controller-Hub **1320** mit dem/den Prozessor(en) **1310**, **1315** über einen Multidrop-Bus, wie etwa einen Frontside-Bus (FSB), eine Punkt-zu-Punkt-Schnittstelle, wie etwa QuickPath Interconnect (QPI), oder eine ähnliche Verbindung **1395**.

[0118] In einer Ausführungsform ist der Koprozessor **1345** ein Spezialprozessor, wie etwa beispielsweise ein MIC-Prozessor mit hohem Durchsatz, ein Netzwerk- oder Kommunikationsprozessor, Kompressionseinheit, Grafikprozessor, GPGPU, eingebetteter Prozessor oder ähnliches. In einer Ausführungsform kann der Controller-Hub **120** einen integrierten Grafikbeschleuniger enthalten.

[0119] Es kann eine Vielzahl von Unterschieden zwischen den physikalischen Betriebsmitteln **1310**, **1315** hinsichtlich eines Spektrums von Metriken vorliegen, einschließlich architektonische, mikroarchitektonische, thermische, Energieverbrauchscharakteristiken und ähnliche.

[0120] In einer Ausführungsform führt der Prozessor **1310** Befehle aus, die Datenverarbeitungsoperationen eines allgemeinen Typs steuern. Eingebettet innerhalb der Befehle können Koprozessorbefehle sein. Der Prozessor **1310** erkennt diese Koprozessorbefehle als einen Typ, der durch den verbundenen Koprozessor **1345** ausgeführt werden sollte. Somit gibt der Prozessor **1310** diese Koprozessorbefehle (oder Steuersignale, die Koprozessorbefehle darstellen) auf einen Koprozessorbus oder andere Verbindung zu dem Koprozessor **1345** aus. Koprozessor(en) **1345** akzeptiert/akzeptieren die erhaltenen Koprozessorbefehle und führt/führen sie aus.

[0121] Unter Bezugnahme auf **Fig.** 14 ist ein Blockdiagramm eines ersten konkreteren beispielhaften Systems **1400** gemäß einer Ausführungsform der vorliegenden Erfindung gezeigt. Wie in **Fig.** 14 gezeigt, ist das Multiprozessorsystem **1400** ein Punkt-zu-Punkt-Verbindungssystem und enthält einen ersten Prozessor **1470** und einen zweiten Prozessor **1480**, die über eine Punkt-zu-Punkt-Verbindung **1450** miteinander verbunden sind. Jeder der Prozessoren **1470** und **1480** kann irgendeine Version des Prozessors **1200** sein. In einer Ausführungsform sind die Prozessoren **1470** und **1480** jeweils die Prozessoren **1310** und **1315**, während der Koprozessor **1438** ein Koprozessor **1345** ist. In einer anderen Ausführungsform sind die Prozessoren **1470** und **1480** jeweils Prozessor **1310** und Koprozessor **1345**.

[0122] Die Prozessoren **1470** und **1480** sind einschließlich den jeweiligen integrierten Speichercontroller(integrated memory controller (IMC))-Einheiten **1472** und **1482** gezeigt. Der Prozessor **1470** enthält außerdem als Teil seiner Buscontroller-Einheiten Punkt-zu-Punkt(P-P)-Schnittstellen **1476** und **1478**; auf ähnliche Weise enthält ein zweiter Prozessor **1480** P-P-Schnittstellen **1486** und **1488**. Die Prozessoren **1470**, **1480** können über eine Punkt-zu-Punkt (P-P)-Schnittstelle **1450** Information unter Verwendung von P-P-Schnittstellenschaltungen **1478**, **1488** austauschen. Wie in **Fig.** 14 gezeigt, koppeln die IMCs **1472** und **1482** die Prozessoren mit jeweiligen

Speichern, nämlich einem Speicher **1432** und einem Speicher **1434**, die Abschnitte des Hauptspeichers sein können, der lokal mit den jeweiligen Prozessoren verbunden ist.

[0123] Die Prozessoren **1470**, **1480** können jeweils Information mit einem Chipsatz **1490** über individuelle P-P-Schnittstellen **1452**, **1454** unter Verwendung von Punkt-zu-Punkt-Schnittstellenschaltungen **1476**, **1494**, **1486**, **1498** austauschen. Der Chipsatz **1490** kann optional Information mit dem Koprozessor **1438** über eine Hochleistungsschnittstelle **1439** austauschen. In einer Ausführungsform ist der Koprozessor **1438** ein Spezialprozessor, wie etwa beispielsweise ein MIC-Prozessor mit hohem Durchsatz, ein Netzwerk- oder Kommunikationsprozessor, eine Kompressionseinheit, ein Grafikprozessor, GPGPU, eingebetteter Prozessor oder ähnliches.

[0124] Ein gemeinsam benutzter Cache (nicht gezeigt) kann in jedem Prozessor oder außerhalb beider Prozessoren enthalten sein und trotzdem mit den Prozessoren über P-P-Verbindung verbunden sein, so dass lokale Cache-Information jedes oder beider Prozessoren in dem gemeinsam gespeicherten Cache gespeichert werden kann, wenn ein Prozessor in einen Niedrigenergiemodus versetzt wird.

[0125] Der Chipsatz **1490** kann mit einem ersten Bus **1416** über eine Schnittstelle **1496** gekoppelt sein. In einer Ausführungsform kann der erste Bus **1416** ein Peripheral Component Interconnect(PCI)-Bus oder ein Bus wie etwa PCI-Express-Bus oder ein anderer I/O-Verbindungsbus dritter Generation sein, obwohl der Schutzbereich der vorliegenden Erfindung nicht in dieser Hinsicht beschränkt ist.

[0126] Wie in **Fig.** 14 gezeigt, können verschiedene I/O-Geräte **1414** mit dem ersten Bus **1416** gekoppelt sein, gemeinsam mit einer Bus-Bridge **1418**, die den ersten Bus **1416** mit einem zweiten Bus **1420** koppelt. In einer Ausführungsform sind ein oder mehrere weitere(r) Prozessor(en) **1415**, wie etwa Koprozessoren, MIC-Prozessoren mit hohem Durchsatz, GPGPUs, Beschleuniger (wie etwa beispielsweise Grafikbeschleuniger oder digitale Signalverarbeitungs(digital signal processing (DSP))-Einheiten, Field programmable Gate Arrays oder irgendein anderer Prozessor mit dem ersten Bus **1416** gekoppelt. In einer Ausführungsform kann der zweite Bus **1420** ein Low Pin Count(LPC)-Bus sein. Verschiedene Geräte können an einen zweiten Bus **1420** gekoppelt sein, einschließlich beispielsweise einer Tastatur und/oder Maus **1422**, Kommunikationsgeräten **1427** und einer Speichereinheit **1428**, wie etwa einem Plattenlaufwerk oder anderem Massenspeichergerät, das in einer Ausführungsform Befehle/Code und Daten **1430** enthalten kann. Ferner kann ein Audio-I/O **1424** mit dem zweiten Bus **1420** gekoppelt sein. Es versteht sich, dass andere Architekturen möglich sind. Beispielsweise kann statt der Punkt-zu-Punkt-Architektur aus **Fig.** 14 ein System einen Multidrop-Bus oder andere solche Architektur implementieren.

[0127] Unter Bezugnahme auf **Fig.** 15 ist ein Blockdiagramm eines zweiten konkreteren beispielhaften Systems **1500** gemäß einer Ausführungsform der vorliegenden Erfindung gezeigt. Gleiche Elemente in den **Fig.** 14 und **Fig.** 15 enthalten gleiche Bezugszeichen, und bestimmte Aspekte aus **Fig.** 14 wurden aus **Fig.** 15 weggelassen, um Verschleiern anderer Aspekte aus **Fig.** 15 zu vermeiden.

[0128] **Fig.** 15 zeigt, dass die Prozessoren **1470**, **1480** jeweils integrierten Speicher und I/O-Steuerlogik (control logik (CL)) **1472** und **1482** enthalten können. Somit enthalten die CL **1472**, **1482** integrierte Speichercontrollereinheiten und I/O-Steuerlogik. **Fig.** 15 zeigt, dass nicht nur die Speicher **1432**, **1434** mit den CL **1472**, **1482** gekoppelt sind, sondern auch dass die I/O-Geräte **1514** mit den Steuerlogiken **1472**, **1482** gekoppelt sind. Veraltete I/O-Geräte 1515 sind mit dem Chipsatz **1490** gekoppelt.

[0129] Unter Bezugnahme auf **Fig.** 16 ist ein Blockdiagramm eines SoC **1600** gemäß einer Ausführungsform der vorliegenden Erfindung gezeigt. Ähnliche Elemente in **Fig.** 12 weisen ähnliche Bezugszeichen auf. Außerdem zeigen gestrichelt umrandete Kästen optionale Merkmale weiterentwickelter SoCs. In **Fig.** 16 ist eine/sind Verbindungseinheit(en) **1602** gekoppelt mit: einem Anwendungsprozessor **1610**, der einen Satz eines oder mehrerer Kerne **202A**–N und gemeinsam benutzte Cacheeinheit(en) **1206** enthält; (einer) Systemagenteinheit **1210**; einer Buscontrollereinheit(en) **1216**; (einer) integrierten Speichercontrollereinheit(en) 1214; einem Satz aus einem oder mehreren Koprozessoren **1620**, die integrierte Grafiklogik enthalten können; einem Bildprozessor; einem Audioprozessor und einem Videoprozessor; einer statischen Speichereinheit mit wahlfreiem Zugriff (static random access memory (SRAM)) **1630**; einer Direktspeicherzugriffseinheit (direct memory access (DMA)) **1632**; und einer Anzeigeeinheit **1640** zum Koppeln an eine oder mehrere externe Anzeigen. In einer Ausführungsform enthält/enthalten der/die Koprozessor(en) **1620** einen Spezialprozessor, wie etwa beispielsweise einen Netzwerk- oder Kommunikationsprozessor, Kompressionseinheit, GPGPU, einen MIC-Prozessor mit hohem Durchsatz, eingebetten Prozessor oder ähnliche.

[0130] Ausführungsformen der hier offenbarten Mechanismen können in Hardware, Software, Firmware oder einer Kombination solcher Implementationsansätze implementiert sein. Ausführungsformen der Erfindung können als Computerprogramme oder Programmcode implementiert sein, die/der auf programmierbaren Systemen ausgeführt wird/werden, die wenigstens einen Prozessor, ein Speichersystem

(einschließlich flüchtigen und nichtflüchtigen Speicher und/oder Speicherelemente), wenigstens ein Eingabegerät und wenigstens ein Ausgabegerät umfassen.

[0131] Programmcode, wie etwa der in **Fig.** 14 gezeigte Code **1430**, kann auf Eingabebefehle angewendet werden, um die hier beschriebenen Funktionen durchzuführen und Ausgabeinformation zu erzeugen. Die Ausgabeinformation kann auf ein oder mehrere Ausgabegeräte in bekannter Weise angewendet werden. Für Zwecke dieser Anmeldung umfasst ein Prozessorsystem irgendein System, das einen Prozessor aufweist, wie etwa beispielsweise: einen digitalen Signalprozessor (DSP), einen Mikrocontroller, eine anwendungsspezifische integrierte Schaltung (application specific integrated circuit (ASIC)) oder einen Mikroprozessor.

[0132] Der Programmcode kann in einer prozeduralen oder objektorientierten Hochprogrammiersprache implementiert sein, um mit einem Prozessorsystem zu kommunizieren. Der Programmcode kann auch in Assembler oder Maschinensprache implementiert sein, falls gewünscht. Tatsächlich sind die hier beschriebenen Mechanismen hinsichtlich des Schutzbereichs nicht auf irgendeine bestimmte Programmiersprache beschränkt. In jedem Fall kann die Sprache eine kompilierte oder interpretierte Sprache sein.

[0133] Ein oder mehrere Aspekte wenigstens einer Ausführungsform können durch repräsentative Befehle implementiert sein, die auf einem maschinenlesbaren Medium gespeichert sind, das vielfältige Logik innerhalb des Prozessors darstellt, die, wenn sie von einer Maschine gelesen wird, die Maschine veranlasst, Logik herzustellen, um die hier beschriebenen Techniken durchzuführen. Solche Darstellungen, die als ”IP cores” bekannt sind, können auf einem greifbaren, maschinenlesbaren Medium gespeichert sein und verschiedenen Kunden oder Herstellungseinrichtungen übergeben werden, um in die Herstellungsmaschinen geladen zu werden, die die Logik oder den Prozessor schließlich herstellen.

[0134] Solche maschinenlesbaren Speichermedien können ohne Einschränkung nichtflüchtige greifbare Anordnungen von Erzeugnissen einschließen, die durch eine Maschine oder Vorrichtung herstellt oder ausgebildet werden, einschließlich Speichermedien wie etwa Festplatten, irgendeinem anderen Typ von Disk einschließlich Floppy-Disks, optische Disks, Compact Disk Read-Only Memories (CD-ROMs), Compact Disk Rewritables (CD-RWs) und magneto-optische Disks, Halbleitervorrichtungen wie etwa Nurlesespeicher (read-only memory (ROMs), Speicher mit wahlfreiem Zugriff (random access memories (RAMs)) wie etwa dynamische Speicher mit wahlfreiem Zugriff (dynamic random access memories (DRAMs)), statische Speicher mit wahlfreiem Zugriff (static random access memories (SRAMs)), Erasable Programmable Read-Only Memories (EPROMs), Flashspeicher, Electrically Erasable Programmable Read-Only Memories (EEPROMs), Phasenwechselspeicher (phase change memory (PCM)), magnetische oder optische Karten oder irgendein anderer Medientyp, der sich zum Speichern elektronischer Befehle eignet.

[0135] Somit schließen Ausführungsformen der Erfindung auch nichtflüchtige greifbare maschinenlesbare Medien ein, die Befehle enthalten oder Designdaten enthalten, wie etwa Hardwarebeschreibungssprache (hardware description language (HDL)), die Strukturen, Schaltungen, Vorrichtungen, Prozessoren und/oder Systemmerkmale definiert, die hier beschrieben sind. Solche Ausführungsformen können auch als Programmerzeugnisse bezeichnet werden.

Emulation (einschließlich binäre Übersetzung, Code-Morphing etc.)

[0136] In einigen Ausführungsformen kann ein Befehlswandler verwendet werden, um einen Befehl aus einem Quellbefehlssatz zu einem Zielbefehlssatz umzuwandeln. Beispielsweise kann der Befehlswandler einen Befehl in einen oder mehrere andere Befehle, die durch den Kern verarbeitet werden sollen, übersetzen (beispielsweise unter Verwendung statischer binärer Übersetzung, dynamischer binärer Übersetzung einschließlich dynamischem Kompilieren), morphen, emulieren oder auf andere Weise umwandeln. Der Befehlswandler kann in Software, Hardware, Firmware oder einer Kombination derselben implementiert sein. Der Befehlswandler kann auf einem Prozessor, außerhalb eines Prozessors oder teilweise auf und teilweise außerhalb eines Prozessors sein.

[0137] **Fig.** 17 zeigt ein Blockdiagramm, das die Verwendung eines Softwarebefehlswandlers zum Umwandeln binärer Befehle in einem Quellbefehlssatz in binäre Befehle in einem Zielbefehlssatz gemäß Ausführungsformen der Erfindung hervorhebt. In der gezeigten Ausführungsform ist der Befehlswandler ein Softwarebefehlswandler, obwohl der Befehlswandler alternativ in Software, Firmware, Hardware oder verschiedenen Kombinationen derselben implementiert sein kann. **Fig.** 17 zeigt ein Programm in einer Hochsprache **1702**, das unter Verwendung eines x86-Compilers **1704** kompiliert sein kann, um einen x86-Binärcode **1706** zu erzeugen, der nativ durch einen Prozessor mit wenigstens einem x86-Befehlssatzkern **1716** ausgeführt werden kann. Der Prozessor mit wenigstens einem x86-Befehlssatzkern **1716** stellt irgendeinen Prozessor dar, der im Wesentlichen die gleichen Funktionen wie ein Intel-Prozessor mit wenigstens einem x86-Befehlssatzkern durchführen kann, indem er (1) einen wesentlichen Teil des Befehlssatzes des Intel-x86-Befehlssatzes des Intel-

x86-Befehlssatzkerns oder (2) Objektcodeversionen von Anwendungen oder andere Software kompatibel ausführt oder auf andere Weise verarbeitet, die vorgesehen ist, um auf einem Intel-Prozessor mit wenigstens einem x86-Befehlssatzkern abzulaufen, um im Wesentlichen das gleiche Ergebnis wie ein Intel-Prozessor mit wenigstens einem x86-Befehlssatzkern zu erzielen. Der x86-Compiler **1704** stellt einen Compiler dar, der betrieben werden kann, einen x86-Binärcode **1706** (beispielsweise Objektcode) zu erzeugen, der mit oder ohne zusätzlicher Linkverarbeitung auf dem Prozessor mit wenigstens einem x86-Befehlssatzkern **1716** ausgeführt werden kann. Auf ähnliche Weise zeigt **Fig.** 17, dass das Programm in der Hochsprache **1702** unter Verwendung eines alternativen Befehlssatzcompilers **1708** kompiliert werden kann, um einen alternativen Befehlssatzbinärcode **1710** zu erzeugen, der von einem Prozessor ohne wenigstens einen x86-Befehlssatzkern **1714** (beispielsweise einem Prozessor mit Kernen, die den MIPS-Befehlssatz von MIPS-Technologies aus Sunnyvale, CA ausführen und/oder den ARM-Befehlssatz von ARM-Holdings aus Sunnyvale, CA ausführen) nativ ausgeführt werden kann. Der Befehlswandler **1712** wird verwendet, um den x86-Binärcode **1706** in Code umzuwandeln, der von dem Prozessor ohne einen x86-Befehlssatzkern **1714** nativ ausgeführt werden kann. Dieser umgewandelte Code ist wahrscheinlicherweise nicht der gleiche wie der alternative Befehlssatzbinärcode **1710**, weil ein Befehlswandler, der dazu in der Lage ist, schwierig herzustellen ist; jedoch wird der umgewandelte Code den allgemeinen Betrieb ermöglichen und aus Befehlen aus dem alternativen Befehlssatz bestehen. Somit stellt der Befehlswandler **1712** Software, Firmware, Hardware oder eine Kombination derselben dar, die durch Emulation, Simulation oder irgendeinen anderen Prozess einem Prozessor oder anderer elektronischer Vorrichtung ermöglichen, der/die keinen x86-Befehlssatzprozessor oder -kern enthält, den x86-Binärcode **1706** auszuführen.

[0138] In anderen Ausführungsformen kann die Bibliothek selbst Logik enthalten, um einen Satz von Bibliotheksteilen auszuwählen, die für ein Softwaremodul geeignet sind. Beispielsweise kann die Bibliothek ein Prozessormerkmalzustandsregister lesen, um zu bestimmen, welche Bedeutung das Softwaremodul für den gegebenen Opcode hat, und diesen Teil dann auswählen und bereitstellen.

[0139] Komponenten, Merkmale und Details, die für eine der **Fig.** 1, **Fig.** 4 und **Fig.** 5 beschrieben sind, können optional in einer der **Fig.** 2 und **Fig.** 3 verwendet werden. Darüber hinaus können Komponenten, Merkmale und Details, die hier für irgendeine der Vorrichtungen beschrieben sind, in einem der hier beschriebenen Verfahren verwendet werden, die in Ausführungsformen durch und/oder mit einer solchen Vorrichtung durchgeführt werden können.

Beispielhafte Ausführungsformen

[0140] Die folgenden Beispiele gehören weiteren Ausführungsform an. Angaben in den Beispielen können überall in einer oder mehreren Ausführungsformen verwendet werden.

[0141] Beispiel 1 ist ein Prozessor, der Dekodierlogik zum Empfangen eines ersten Befehls und zum Bestimmen, dass der erste Befehl emuliert werden soll, enthält. Der Prozessor enthält außerdem emulationsmodusbewusste Nachdekodier-Befehlsprozessorlogik, die mit der Dekodierlogik gekoppelt ist. Die emulationsmodusbewusste Nachdekodierprozessorlogik soll ein oder mehrere Steuersignale, die von einem Befehl aus einem Satz von einem oder mehreren Befehlen dekodiert wurden, anders verarbeiten, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt.

[0142] Beispiel 2 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei der erste Befehl optional komplexer ist als jeder Befehl aus dem Satz, indem der erste Befehl mehr Operationen einschließt, die durchgeführt werden.

[0143] Beispiel 3 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei der Prozessor optional keinen Mikrocode verwendet, um irgendwelche Befehle aus einem Befehlssatz zu implementieren.

[0144] Beispiel 4 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei jeder Befehl aus dem Satz von einem oder mehreren Befehlen optional aus einem gleichen Befehlssatz wie der erste Befehl stammt.

[0145] Beispiel 5 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei die emulationsbewusste Nachdekodier-Befehlsprozessorlogik optional emulationsmodusbewusste Ausnahmebehandlerlogik umfasst, um eine Ausnahmebedingung zu berichten, die auftritt, während das eine oder die mehreren Steuersignale an Emulationslogik verarbeitet werden.

[0146] Beispiel 6 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei die emulationsmodusbewusste Ausnahmebehandlerlogik optional eine Adresse des ersten Befehls in einem Stack speichert.

[0147] Beispiel 7 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei die emulationsmodusbewusste Ausnahmebehandlerlogik eine Anzeige einer Ausnahmebedingung und einen Fehlercode für die Ausnahmebedingung in einem oder mehreren Registern speichert, die mit der Emulationslogik gekoppelt sind.

[0148] Beispiel 8 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei die emulationsmodusbewusste Ausnahmebehandlerlogik optional die Steuerung direkt an einen Ausnahmebehandler übergibt als Antwort auf die Ausnahmebedingung, und wobei ein oder mehrere Befehle der Emulationslogik die Steuerung an den Ausnahmebehandler übergibt.

[0149] Beispiel 9 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei die emulationsmodusbewusste Nachdekodier-Befehlsprozessorlogik optional umfasst, dass die emulationsmodusbewusste Zugriffssteuerlogik Zugriff auf wenigstens ein Betriebsmittel und/oder Information durch die ein oder mehreren Steuersignale anders steuert, wenn der Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt.

[0150] Beispiel 10 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei die emulationsmodusbewusste Zugriffssteuerlogik optional Zugriff auf das wenigstens eine Betriebsmittel und/oder die Information ermöglicht, wenn der Emulationsmodus vorliegt, und Zugriff auf das wenigstens eine Betriebsmittel und/oder die Information verhindert, wenn kein Emulationsmodus vorliegt.

[0151] Beispiel 11 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei das wenigstens eine Betriebsmittel und/oder die Information optional wenigstens eine Sicherheitslogik Sicherheitsinformation, Verschlüsselungslogik, Entschlüsselungslogik, Zufallsgeneratorlogik, Logik, die für Zugriffe durch ein Betriebssystem reserviert ist, einen Abschnitt von Speicher, der für Zugriffe durch ein Betriebssystem reserviert ist, und Information, die für Zugriff durch ein Betriebssystem reserviert ist, umfasst.

[0152] Beispiel 12 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei das wenigstens eine Betriebsmittel und/oder die Information wenigstens ein Betriebsmittel und/oder Information in einem anderen logischen Prozessor und/oder einem anderen physikalischen Prozessor umfasst.

[0153] Beispiel 13 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei der Satz von einem oder mehreren Befehlen optional wenigstens drei Befehle enthält.

[0154] Beispiel 14 ist ein Verfahren in einem Prozessor, das Empfangen eines ersten Befehls und Bestimmen zum Emulieren des ersten Befehls einschließt. Das Verfahren schließt außerdem Empfangen eines Satzes von einem oder mehreren Befehlen ein, die verwendet werden sollen, um den ersten Befehl zu emulieren. Das Verfahren schließt außerdem Verarbeiten von einem oder mehreren Steuersignalen, die von einem Befehl des Satzes abgeleitet sind, auf andere Weise ein, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt.

[0155] Beispiel 15 schließt das Verfahren irgendeines vorherigen Beispiels ein, wobei optional Empfangen des ersten Befehls Empfangen des ersten Befehls umfasst, der komplexer ist als jeder Befehl des Satzes von einem oder mehreren Befehlen.

[0156] Beispiel 16 schließt das Verfahren irgendeines vorherigen Beispiels ein, wobei optional Empfangen des Satzes von einem oder mehreren Befehlen Empfangen von einem oder mehreren Befehlen umfasst, die jeweils aus einem gleichen Befehlssatz stammen wie der erste Befehl.

[0157] Beispiel 17 schließt das Verfahren irgendeines vorherigen Beispiels ein, wobei optional Verarbeiten Berichten einer Ausnahmebedingung umfasst, die auftritt, während das eine oder die mehreren Steuersignale zur Emulationslogik verarbeitet werden. Außerdem optional Ausführen eines oder mehrerer Befehle der Emulationslogik, um Steuerung an einen Ausnahmebehandler zu übergeben.

[0158] Beispiel 18 schließt das Verfahren irgendeines vorherigen Beispiels ein, wobei optional Berichten optional Speichern einer Anzeige der Ausnahmebedingung in einem oder mehreren Registern umfasst. Außerdem optional Speicher einer Adresse des ersten Befehls in einem Stack.

[0159] Beispiel 19 schließt das Verfahren irgendeines vorherigen Beispiels ein, wobei optional Verarbeiten Steuern von Zugriff auf wenigstens ein Betriebsmittel oder Information durch die ein oder mehreren Steuersignale auf andere Weise umfasst, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt.

[0160] Beispiel 20 schließt das Verfahren irgendeines vorherigen Beispiels ein, wobei optional Steuern von Zugriff auf andere Weise Ermöglichen von Zugriff auf das Betriebsmittel und/oder die Information umfasst, wenn der Emulationsmodus vorliegt. Außerdem optional Verhindern von Zugriff auf das Betriebsmittel und/oder die Information, wenn kein Emulationsmodus vorliegt.

[0161] Beispiel 21 ist ein System zum Verarbeiten von Befehlen, das eine Verbindung und einen Prozessor aufweist, der mit der Verbindung gekoppelt ist. Der Prozessor enthält Dekodierlogik zum Empfangen eines ersten Befehls und zum Bestimmen, dass der erste Befehl emuliert werden soll. Der Prozessor enthält außerdem emulationsmodusbewusste Nachdekodier-Befehlsprozessorlogik, die mit der Dekodierlogik gekoppelt ist. Die emulationsmodusbewusste Nachdekodier-Befehlsprozessorlogik soll ein oder mehrere Steuersignale, die aus einem Befehl

dekodiert wurden, der aus einem Satz von einem oder mehreren Befehlen stammt, die verwendet werden, um den ersten Befehl zu emulieren, auf andere Weise verarbeiten, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt. Das System weist außerdem einen dynamischen Speicher mit wahlfreiem Zugriff (dynamic random access memory (DRAM)) auf, der mit der Verbindung gekoppelt ist.

[0162] Beispiel 22 schließt das System aus Beispiel 21 ein, wobei optional die emulationsmodusbewusste Nachdekodier-Befehlsprozessorlogik emulationsmodusbewusste Ausnahmebehandlerlogik zum Berichten einer Ausnahmebedingung umfasst, die auftritt, während das eine oder die mehreren Steuersignale zur Emulationslogik verarbeitet werden.

[0163] Beispiel 1 ist ein Prozessor, der einen Dekodierer zum Empfangen eines ersten Befehls enthält, der einen gegebenen Opcode aufweist. Der Dekodierer enthält Prüflogik zum Prüfen, ob der gegebene Opcode eine erste Bedeutung oder eine zweite Bedeutung aufweist. Der Dekodierer enthält außerdem Dekodierlogik zum Dekodieren des ersten Befehls und gibt ein oder mehrere entsprechende Steuersignale aus, wenn der gegebene Opcode die erste Bedeutung aufweist. Der Dekodierer enthält außerdem Emulationsauslöselogik zum Auslösen von Emulation des ersten Befehls, wenn der gegebene Opcode die zweite Bedeutung aufweist.

[0164] Beispiel 2 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei optional die zweite Bedeutung älter als die erste Bedeutung ist.

[0165] Beispiel 3 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei optional die zweite Bedeutung eine Opcodedefinition umfasst, die im Begriff ist, zu veralten.

[0166] Beispiel 4 schließt den Prozessor irgendeines vorherigen Beispiels ein und umfasst optional ferner einen Speicherplatz, der mit dem Dekodierer zum Speichern einer Anzeige gekoppelt ist, ob der gegebene Opcode die erste Bedeutung oder die zweite Bedeutung aufweist, und wobei die Prüflogik den Speicherplatz prüfen soll, um die Anzeige zu bestimmen.

[0167] Beispiel 5 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei der Speicherplatz für ein Programmladermodul zugänglich ist, um dem Programmladermodul zu ermöglichen, die Anzeige an dem Speicherplatz zu speichern.

[0168] Beispiel 6 schließt den Prozessor irgendeines vorherigen Beispiels ein und umfasst optional ferner Logik, die mit dem Speicherplatz gekoppelt ist, um die Anzeige von dem Speicherplatz in ein Prozessorstrukturregister zu speichern, wobei das Prozessorstrukturregister durch einen Prozessorstrukturidentifizierungsbefehl eines Befehlssatzes des ersten Befehls lesbar ist.

[0169] Beispiel 7 schließt den Prozessor irgendeines vorherigen Beispiels ein und umfasst optional ferner eine Vielzahl an Speicherplätzen, die mit dem Dekodierer gekoppelt ist, um eine Vielzahl von Anzeigen zu speichern, wobei jede der Anzeigen einem anderen Opcode einer Vielzahl von Opcodes entspricht, wobei jede der Anzeigen anzeigen soll, ob jeder jeweilige Opcode eine erste Bedeutung oder eine zweite Bedeutung aufweist.

[0170] Beispiel 8 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei optional die Logik zum Auslösen der Emulation Logik umfasst, um einen Emulationsmodus festzusetzen.

[0171] Beispiel 9 schließt den Prozessor irgendeines vorherigen Beispiels ein und umfasst optional ferner Emulationslogik, die mit dem Dekodierer gekoppelt ist, wobei die Emulationslogik als Antwort auf die Emulationsauslöselogik, die die Emulation auslöst, einen Satz von einem oder mehreren Befehlen an den Dekodierer liefert, um den ersten Befehl zu emulieren, wenn der gegebenen Opcode die zweite Bedeutung aufweist.

[0172] Beispiel 10 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei optional jeder Befehl des Satzes aus einem gleichen Befehlssatz stammt wie der erste Befehl.

[0173] Beispiel 11 schließt den Prozessor irgendeines vorherigen Beispiels ein, wobei der Prozessor optional keinen Mikrocode verwendet, um irgendwelche Befehle eines Befehlssatzes zu implementieren.

[0174] Beispiel 12 schließt den Prozessor irgendeines vorherigen Beispiels ein und umfasst optional ferner optional Logik, um den Dekodierer zu zwingen, eine neuere Bedeutung anstelle einer veralteten Bedeutung für den gegebenen Opcode zu verwenden, wenn eine Berechtigungsstufenlogik oder eine Ringstufenlogik einen Betriebssystemmodus anzeigt.

[0175] Beispiel 13 ist ein Verfahren in einem Prozessor, das Empfangen eines ersten Befehls, der einen gegebenen Opcode aufweist, und Bestimmen einschließt, dass der gegebene Opcode eine zweite Bedeutung anstelle einer ersten Bedeutung aufweist. Das Verfahren schließt außerdem Bestimmen ein, den ersten Befehl als Antwort auf Bestimmen zu emulieren, dass der gegebene Opcode eine zweite Bedeutung aufweist.

[0176] Beispiel 14 schließt das Verfahren irgendeines vorherigen Beispiels ein, wobei Bestimmen optio-

nal Bestimmen umfasst, dass der gegebene Opcode eine zweite Bedeutung aufweist, die älter ist als die erste Bedeutung, und wobei die zweite Bedeutung im Begriff ist, zu veralten.

[0177] Beispiel 15 schließt ein Verfahren irgendeines vorherigen Beispiels ein, wobei Bestimmen optional Lesen einer Anzeige umfasst, dass der gegebene Opcode die zweite Bedeutung von einem Speicherplatz aufweist.

[0178] Beispiel 16 schließt das Verfahren irgendeines vorherigen Beispiels ein und umfasst optional Speichern der Anzeige, dass der gegebene Opcode die zweite Bedeutung in einem Prozessorstrukturregister aufweist, das durch einen Prozessorstrukturidentifizierungsbefehl eines Befehlssatzes des Prozessors lesbar ist.

[0179] Beispiel 17 schließt das Verfahren irgendeines vorherigen Beispiels ein und umfasst optional ferner Emulieren des ersten Befehls, einschließlich Dekodieren eines Satzes von einem oder mehreren Befehlen, die verwendet werden, um den ersten Befehl zu emulieren, wenn der gegebene Opcode die zweite Bedeutung aufweist.

[0180] Beispiel 18 schließt das Verfahren irgendeines vorherigen Beispiels ein, wobei Dekodieren des Satzes von Befehlen optional Dekodieren eines oder mehrerer Befehle umfasst, die von einem gleichen Befehlssatz wie der erste Befehl stammen.

[0181] Beispiel 19 schließt das Verfahren irgendeines vorherigen Beispiels ein, das optional in dem Prozessor durchgeführt wird, der keinen Mikrocode verwendet, um irgendwelche Befehle eines Befehlssatzes zu implementieren.

[0182] Beispiel 20 ist ein Erzeugnis, das ein nichtflüchtiges maschinenlesbares Speichermedium enthält, das Befehle speichert, die, wenn sie durch eine Maschine ausgeführt werden, die Maschine veranlassen, Operationen durchzuführen. Die Operationen schließen Bestimmen ein, dass ein erster Befehl, der den gegebenen Opcode aufweist, eine zweite Bedeutung anstelle einer ersten Bedeutung aufweist, wenn er durch einen Prozessor von einem Softwaremodul ausgeführt wird, indem er Metadaten des Softwaremoduls prüft. Die Operationen schließen außerdem Speichern einer Anzeige ein, dass der erste Befehl, der den gegebenen Opcode aufweist, die zweite Bedeutung in einem Zustand des Prozessors aufweisen soll.

[0183] Beispiel 21 schließt das Erzeugnis irgendeines vorherigen Beispiels ein, wobei optional das maschinenlesbare Speichermedium ferner Befehle speichert, die, wenn sie durch die Maschine ausgeführt werden, die Maschine veranlassen, Operationen durchzuführen, einschließlich Auswählen eines Teils aus einer Softwarebibliothek, der die zweite Bedeutung des gegebenen Opcodes verwendet, anstelle eines anderen Teils der Softwarebibliothek, der die erste Bedeutung des gegebenen Opcodes verwendet, und Bereitstellen des ausgewählten Teils der Softwarebibliothek an das Softwaremodul, wobei die zweite Bedeutung eine veraltete Bedeutung ist.

[0184] Beispiel 22 schließt das Erzeugnis irgendeines vorherigen Beispiels ein, wobei das maschinenlesbare Speichermedium optional ferner Befehle speichert, die, wenn sie durch die Maschine ausgeführt werden, die Maschine veranlassen, Operationen durchzuführen einschließlich Bestimmen anhand eines Alters des Softwaremoduls, dass der gegebene Opcode die zweite Bedeutung aufweist.

[0185] Beispiel 23 schließt das Erzeugnis irgendeines vorherigen Beispiels ein, wobei optional das maschinenlesbare Medium ferner Befehle speichert, die, wenn sie durch die Maschine ausgeführt werden, die Maschine veranlassen, Operationen durchzuführen einschließlich Prüfen eines Flags in einem Objektmodulformat und Speichern der Anzeige in einem Flag in einem Register des Prozessors.

[0186] Beispiel 24 ist ein System zum Verarbeiten von Befehlen, das eine Verbindung und einen Prozessor aufweist, der mit der Verbindung gekoppelt ist. Der Prozessor soll einen ersten Befehl empfangen, der einen gegebenen Opcode aufweist. Der Prozessor enthält Prüflogik zum Prüfen, ob der gegebene Opcode eine erste Bedeutung oder eine zweite Bedeutung aufweist. Der Prozessor enthält Dekodierlogik zum Dekodieren des ersten Befehls und gibt ein oder mehrere entsprechende Steuersignale aus, wenn der gegebene Opcode die erste Bedeutung aufweist. Der Prozessor enthält Emulationsauslöselogik, um Emulation des ersten Befehls auszulösen, wenn der gegebene Opcode die zweite Bedeutung aufweist. Das System weist außerdem einen dynamischen Speicher mit wahlfreiem Zugriff (dynamic random access memory (DRAM)) auf, der mit der Verbindung gekoppelt ist.

[0187] Beispiel 25 schließt den Gegenstand des Beispiels 24 ein und umfasst optional ferner Emulationslogik zum Bereitstellen eines Satzes von einem oder mehreren Befehlen eines gleichen Befehlssatzes wie der erste Befehl an den Dekodierer, um den ersten Befehl zu emulieren, wenn der gegebene Opcode die zweite Bedeutung aufweist.

[0188] Beispiel 26 schließt eine Vorrichtung zum Durchführen des Verfahrens nach einem der Beispiele 13 bis 19 ein.

[0189] Beispiel 27 schließt eine Vorrichtung ein, die Mittel zum Durchführen des Verfahrens nach einem der Beispiele 13 bis 19 umfasst.

[0190] Beispiel 28 schließt eine Vorrichtung zum Durchführen eines Verfahrens im Wesentlichen wie hier beschrieben, ein.

[0191] Beispiel 29 schließt eine Vorrichtung ein, die Mittel zum Durchführen eines hierzu beschriebenen Verfahrens enthält.

[0192] In der Beschreibung und den Ansprüchen können die Begriffe ”gekoppelt” und ”verbunden” gemeinsam mit ihren Ableitungen benutzt worden sein. Es versteht sich, dass diese Begriffe nicht als Synonyme füreinander stehen sollen. Vielmehr kann in bestimmten Ausführungsformen ”verbunden” verwendet werden, um anzuzeigen, dass zwei oder mehr Elemente in direktem physikalischen oder elektrischen Kontakt miteinander stehen. ”Gekoppelt” kann bedeuten, dass zwei oder mehr Elemente in direktem physikalischen oder elektrischen Kontakt miteinander stehen. Jedoch kann ”gekoppelt” auch bedeuten, dass zwei oder mehr Elemente nicht in direktem Kontakt miteinander stehen, jedoch trotzdem miteinander kooperieren oder interagieren. Beispielsweise können eine erste Komponente und eine zweite Komponente miteinander durch eine Zwischenkomponente gekoppelt sein. In den Figuren werden bidirektionale Pfeile verwendet, um bidirektionale Verbindungen und Kopplungen zu zeigen.

[0193] In den Beschreibung und den Ansprüchen kann der Begriff ”Logik” verwendet worden sein. Wie hier verwendet, kann Logik Hardware, Firmware, Software oder eine Kombination derselben einschließen. Beispiele für Logik schließen integrierte Schaltungen, anwendungsspezifische integrierte Schaltungen, analoge Schaltungen, digitale Schaltungen, programmierte Logikgeräte, Speichergeräte einschließlich Befehle etc. ein. In einigen Ausführungsformen kann die Hardwarelogik Transistoren und/oder Gatter potentiell gemeinsam mit anderen Schaltungskomponenten enthalten.

[0194] Der Begriff ”und/oder” kann verwendet worden sein. Wie hier verwendet, bedeutet der Begriff ”und/oder” das eine oder das andere oder beides (beispielsweise bedeutet A und/oder B A oder B oder sowohl A als auch B).

[0195] In der obigen Beschreibung wurden zu Zwecken der Erläuterung zahlreiche spezifische Details dargelegt, um ein tiefgreifendes Verständnis von Ausführungsformen der Erfindung zu liefern. Es versteht sich für einen Durchschnittsfachmann jedoch, dass eine oder mehrere Ausführungsformen ohne einige dieser spezifischen Details umgesetzt werden können. Die genauen beschriebenen Ausführungsformen werden nicht bereitgestellt, um die Erfindung zu beschränken, sondern um sie durch beispielhafte Ausführungsformen zu veranschaulichen. Der Schutzbereich der Erfindung soll nicht durch die spezifischen Beispiele bestimmt werden, sondern nur durch die Ansprüche. In anderen Beispielen wurden wohlbekannte Schaltungen, Strukturen, Geräte und Operationen in Blockdiagrammform oder ohne Details gezeigt, um Verschleiern des Verständnisses der Beschreibung zu vermeiden.

[0196] Wo es angemessen erschien, wurden Bezugszeichen oder Endabschnitte von Bezugszeichen zwischen den Figuren wiederholt, um entsprechende oder analoge Elemente anzuzeigen, die optional ähnliche oder die gleichen Charakteristiken haben können, außer wenn anders angegeben oder klar ersichtlich. Wo mehrere Komponenten beschrieben wurden, können sie allgemein in eine Einzelkomponente integriert sein. In anderen Fällen kann, wo eine Einzelkomponente beschrieben wurde, diese allgemein in mehrere Komponenten unterteilt werden.

[0197] Zahlreiche Operationen und Verfahren wurden beschrieben. Einige der Verfahren wurden in relativ grundlegender Form in den Flussdiagrammen beschrieben, jedoch können den Verfahren Operationen optional hinzugefügt und/oder aus ihnen entfernt werden. Zusätzlich ist, während die Flussdiagramme eine bestimmte Reihenfolge der Operationen gemäß beispielhaften Ausführungsformen zeigen, diese bestimmte Reihenfolge beispielhaft. Alternative Ausführungsformen können optional die Operationen in anderer Reihenfolge durchführen, bestimmte Operationen kombinieren, bestimmte Operationen überlappen etc.

[0198] Einige Ausführungsformen schließen ein Erzeugnis (beispielsweise ein Computererzeugnis) ein, dass ein maschinenlesbares Medium aufweist. Das Medium kann einen Mechanismus enthalten, der Information in einer Form bereitstellt, beispielsweise speichert, die durch die Maschine lesbar ist. Das maschinenlesbare Medium kann einen oder mehrere Befehle bereitstellen oder in gespeicherter Form darauf aufweisen, die, wenn und/oder wann immer sie durch eine Maschine ausgeführt werden, betrieben werden können, die Maschine zu veranlassen und/oder dazu zu bringen, in der Maschine eine oder mehrere Operationen, Verfahren oder Techniken, die hier offenbart sind, durchzuführen. Beispiele geeigneter Maschinen schließen, ohne darauf beschränkt zu sein, Prozessoren, Befehlsverarbeitungsvorrichtungen, digitale Logikschaltungen, integrierte Schaltungen und ähnliche ein. Weitere Beispiele geeigneter Maschinen schließen Rechenvorrichtungen und andere elektronische Geräte ein, die solche Prozessoren, Befehlsverarbeitungsvorrichtungen, digitale Logikschaltungen oder integrierte Schaltungen integrieren. Beispiele solcher Rechenvorrichtungen

und elektronischer Geräte schließen Desktopcomputer, Laptopcomputer, Notebookcomputer, Tabletcomputer, Netbooks, Smartphones, Mobiltelefone, Server, Netzwerkgeräte (beispielsweise Router und Switches), Mobile Internet Devices (MIDs), Medienabspielgeräte, intelligente Fernsehgeräte, Net-tops, Set-top-Boxen und Videospielcontroller ein, ohne darauf beschränkt zu sein.

[0199] In einigen Ausführungsformen kann das maschinenlesbare Medium ein greifbares und/oder nichtflüchtiges maschinenlesbares Speichermedium umfassen. Beispielsweise kann das greifbare und/oder nichtflüchtige maschinenlesbare Speichermedium eine Floppy-Diskette, ein optisches Speichermedium, eine optische Disk, eine optische Datenspeichereinrichtung, eine CD-ROM, eine magnetische Disk, eine magneto-optische Disk, einen Nurlesespeicher (read-only memory (ROM), einen programmierbaren ROM (PROM), einen löschbaren und programmierbaren ROM (erasable and programmable ROM (EPROM)), einen elektrisch löschbaren und programmierbaren ROM (electrically erasable and programmable ROM (EEPROM)), einen Speicher mit wahlfreiem Zugriff (random access memory (RAM)), einen statischen RAM (SRAM), einen dynamischen RAM (DRAM), einen Flashspeicher, einen Phasenwechselspeicher, ein Phasenwechseldatenspeichermaterial, einen nichtflüchtigen (non-volatile) Speicher, eine nichtflüchtige Datenspeichervorrichtung, einen nichtflüchtigen (non-transitory) Speicher, eine nichtflüchtige Datenspeichervorrichtung oder ähnliche einschließen. Das nichtflüchtige maschinenlesbare Speichermedium besteht nicht aus einem flüchtigen weitergeleiteten Signal.

[0200] Es versteht sich, dass überall in der Beschreibung Bezugnahme auf ”eine (1) Ausführungsform”, ”eine Ausführungsform” oder ”eine oder mehrere Ausführungsformen” beispielsweise bedeuten, dass ein bestimmtes Merkmal beim Umsetzen der Erfindung enthalten sein kann. Auf ähnliche Weise versteht sich, dass in der Beschreibung zahlreiche Merkmale manchmal in einer einzelnen Ausführungsform, Figur oder Beschreibung derselben zum Zweck der Straffung der Offenbarung und Unterstützung beim Verstehen verschiedener erfinderischer Aspekte zusammengefasst sind. Dieses Verfahren der Offenbarung soll jedoch nicht interpretiert werden, eine Absicht wiederzugeben, derzufolge die Erfindung weitere Merkmale benötigt als ausdrücklich in jedem Anspruch aufgeführt sind. Vielmehr können, wie die folgenden Ansprüche wiedergeben, erfinderische Aspekte in weniger als sämtlichen Merkmalen einer einzelnen offenbarten Ausführungsform liegen. Somit werden die Ansprüche, die der ausführlichen Beschreibung folgen, hierdurch ausdrücklich in diese Ausführliche Beschreibung integriert, wobei jeder Anspruch für sich als eine separate Ausführungsform der Erfindung steht.

Patentansprüche

1. Prozessor, umfassend:
Dekodierlogik zum Empfangen eines ersten Befehls und zum Bestimmen, dass der erste Befehl emuliert werden soll; und
emulationsmodusbewusste Nachdekodierbefehlsprozessorlogik, die mit der Dekodierlogik gekoppelt ist, wobei die emulationsmodusbewusste Nachdekodierbefehlsprozessorlogik ein oder mehrere Steuersignale verarbeiten soll, die aus einem Befehl aus einem Satz von einem oder mehreren Befehlen dekodiert wurden, die verwendet werden, um den ersten Befehl anders zu emulieren, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt.

2. Prozessor nach Anspruch 1, **dadurch gekennzeichnet,** dass der erste Befehl komplexer ist als jeder Befehl des Satzes, indem der erste Befehle mehr Operationen umfasst, die durchgeführt werden.

3. Prozessor nach Anspruch 1 oder 2, **dadurch gekennzeichnet,** dass der Prozessor keinen Mikrocode verwendet, um irgendwelche Befehle eines Befehlssatzes zu implementieren.

4. Prozessor nach Anspruch 1, **dadurch gekennzeichnet,** dass jeder Befehl des Satzes von einem oder mehreren Befehlen aus einem gleichen Befehlssatz stammt wie der erste Befehl.

5. Prozessor nach einem der Ansprüche 1 bis 4, **dadurch gekennzeichnet,** dass die emulationsmodusbewusste Nachdekodierbefehlsprozessorlogik emulationsmodusbewusste Ausnahmebehandlerlogik zum Berichten einer Ausnahmebedingung umfasst, die gegenüber Emulationslogik auftritt, während das eine oder die mehreren Steuersignale verarbeitet werden.

6. Prozessor nach Anspruch 5, **dadurch gekennzeichnet,** dass die emulationsmodusbewusste Ausnahmebehandlerlogik eine Adresse des ersten Befehls in einem Stack speichert.

7. Prozessor nach Anspruch 5 oder 6, **dadurch gekennzeichnet,** dass die emulationsmodusbewusste Ausnahmebehandlerlogik eine Anzeige der Ausnahmebedingung und einen Fehlercode für die Ausnahmebedingung in einem oder mehreren Registern speichern soll, die mit der Emulationslogik gekoppelt sind.

8. Prozessor nach einem der Ansprüche 5 bis 7, **dadurch gekennzeichnet,** dass die emulationsmodusbewusste Ausnahmebehandlerlogik direktes Übertragen einer Steuerung an einen Ausnahmebe-

handler als Antwort auf die Ausnahmebedingung vermeiden soll und dass ein oder mehrere Befehle der Emulationslogik eine Steuerung an den Ausnahmebehandler übergeben sollen.

9. Prozessor nach einem der Ansprüche 1 bis 4, **dadurch gekennzeichnet,** dass die emulationsmodusbewusste Nachdekodierbefehlsprozessorlogik emulationsmodusbewusste Zugriffssteuerlogik zum Steuern von Zugriff auf wenigstens ein Betriebsmittel und/oder Information durch das eine oder die mehreren Signale anders steuern soll, wenn der Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt.

10. Prozessor nach Anspruch 9, **dadurch gekennzeichnet,** dass die emulationsmodusbewusste Zugriffssteuerlogik Zugriff auf wenigstens das Betriebsmittel und/oder die Information ermöglicht, wenn der Emulationsmodus vorliegt, und Zugriff auf das wenigstens eine Betriebsmittel und/oder die Information verhindert, wenn der Emulationsmodus nicht vorliegt.

11. Prozessor nach Anspruch 10, **dadurch gekennzeichnet,** dass das wenigstens eine Betriebsmittel und/oder die Information wenigstens eine Sicherheitslogik, Sicherheitsinformation, Verschlüsselungslogik, Entschlüsselungslogik, Zufallsgeneratorlogik, Logik, die für Zugriffe durch ein Betriebssystem reserviert ist, einen Abschnitt eines Speichers, der für Zugriffe durch ein Betriebssystem reserviert ist, und/oder Information, die für Zugriffe durch ein Betriebssystem reserviert ist, umfasst.

12. Prozessor nach Anspruch 10, **dadurch gekennzeichnet,** dass das wenigstens eine Betriebsmittel und/oder die Information wenigstens ein Betriebsmittel und/oder Information in einem anderen logischen Prozessor und/oder einem anderen physikalischen Prozessor umfasst.

13. Prozessor nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet,** dass der Satz von einem oder mehreren Befehlen wenigstens drei Befehle enthält.

14. Verfahren in einem Prozessor, umfassend:
Empfangen eines ersten Befehls;
Bestimmen, den ersten Befehl zu emulieren;
Empfangen eines Satzes von einem oder mehreren Befehlen, die verwendet werden sollen, um den ersten Befehl zu emulieren; und
Verarbeiten eines oder mehrerer Steuersignale, die von einem Befehl des Satzes abgeleitet sind, auf andere Weise, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt.

15. Verfahren nach Anspruch 14, **dadurch gekennzeichnet,** dass der erste Befehl Empfangen des ersten Befehls umfasst, der komplexer ist als jeder Befehl des Satzes von einem oder mehreren Befehlen.

16. Verfahren nach Anspruch 14, **dadurch gekennzeichnet,** dass Empfangen des Satzes der ein oder mehreren Befehle Empfangen eines oder mehrerer Befehle umfasst, die jeweils aus einem gleichen Befehlssatz stammen wie der erste Befehl.

17. Verfahren nach einem der Ansprüche 14 bis 16, **dadurch gekennzeichnet,** dass Verarbeiten umfasst:
Berichten einer Ausnahmebedingung, die gegenüber Emulationslogik auftritt, während das eine oder die mehreren Steuersignale verarbeitet wird/werden; und
Ausführen eines oder mehrerer Befehle der Emulationslogik, um eine Steuerung an einen Ausnahmebehandler zu übergeben.

18. Verfahren nach Anspruch 17, **dadurch gekennzeichnet,** dass Berichten umfasst:
Speichern einer Anzeige des Ausnahmebehandlers in einem oder mehreren Registern; und
Speichern einer Adresse des ersten Befehls in einem Stack.

19. Verfahren nach einem der Ansprüche 14 bis 16, **dadurch gekennzeichnet,** dass Verarbeiten Steuern von Zugriff auf wenigstens ein Betriebsmittel und/oder Information durch das eine oder die mehreren Steuersignale auf andere Weise umfasst, wenn der Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt.

20. Verfahren nach Anspruch 19, **dadurch gekennzeichnet,** dass Steuern von Zugriff auf andere Weise umfasst:
Ermöglichen von Zugriff auf wenigstens das Betriebsmittel und/oder die Information, wenn der Emulationsmodus vorliegt; und
Verhindern von Zugriff auf das wenigstens eine Betriebsmittel und/oder die Information, wenn der Emulationsmodus nicht vorliegt.

21. System zum Verarbeiten von Befehlen, umfassend:
eine Verbindung;
einen Prozessor, der mit der Verbindung gekoppelt ist, wobei der Prozessor enthält:
Dekodierlogik zum Empfangen eines ersten Befehls und zum Bestimmen, das der erste Befehl emuliert werden soll; und
emulationsmodusbewusste Nachdekodierbefehlsprozessorlogik, die mit der Dekodierlogik gekoppelt ist, wobei die emulationsmodusbewusste Nachdekodierbefehlsprozessorlogik ein oder mehrere Steuersignale verarbeiten soll, die aus einem Befehl aus einem Satz von einem oder mehreren Befehlen dekodiert wurden, die verwendet werden, um den ersten Befehl auf andere Weise zu dekodieren, wenn ein

Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt; und
einen dynamischen Speicher mit wahlfreiem Zugriff (dynamic random access memory (DRAM)), der mit der Verbindung gekoppelt ist.

22. System nach Anspruch 21, **dadurch gekennzeichnet,** dass die emulationsmodusbewusste Nachdekodierprozessorlogik emulationsmodusbewusste Ausnahmebehandlerlogik umfasst, um eine Ausnahmebedingung zu berichten, die gegenüber Emulationslogik auftritt, wenn das eine oder die mehreren Steuersignale verarbeitet werden.

23. Vorrichtung zum Durchführen des Verfahrens nach einem der Ansprüche 14 bis 20.

24. Vorrichtung, umfassend Mittel zum Durchführen des Verfahrens nach einen der Ansprüche 14 bis 20.

25. System, umfassend den Prozessor nach einem der Ansprüche 1 bis 13 und einen dynamischen Speicher mit wahlfreiem Zugriff, der mit dem Prozessor gekoppelt ist.

Es folgen 17 Seiten Zeichnungen

Anhängende Zeichnungen

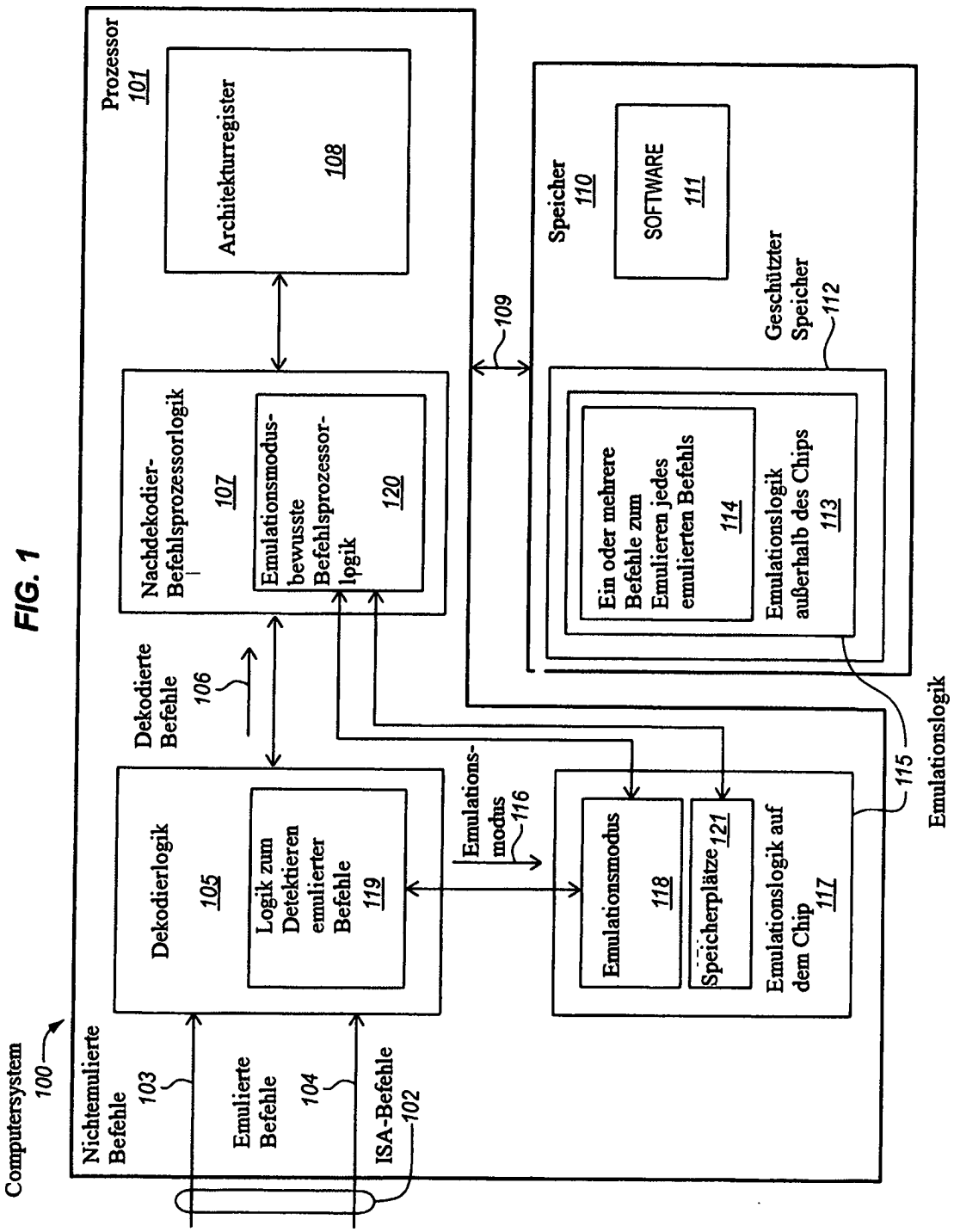
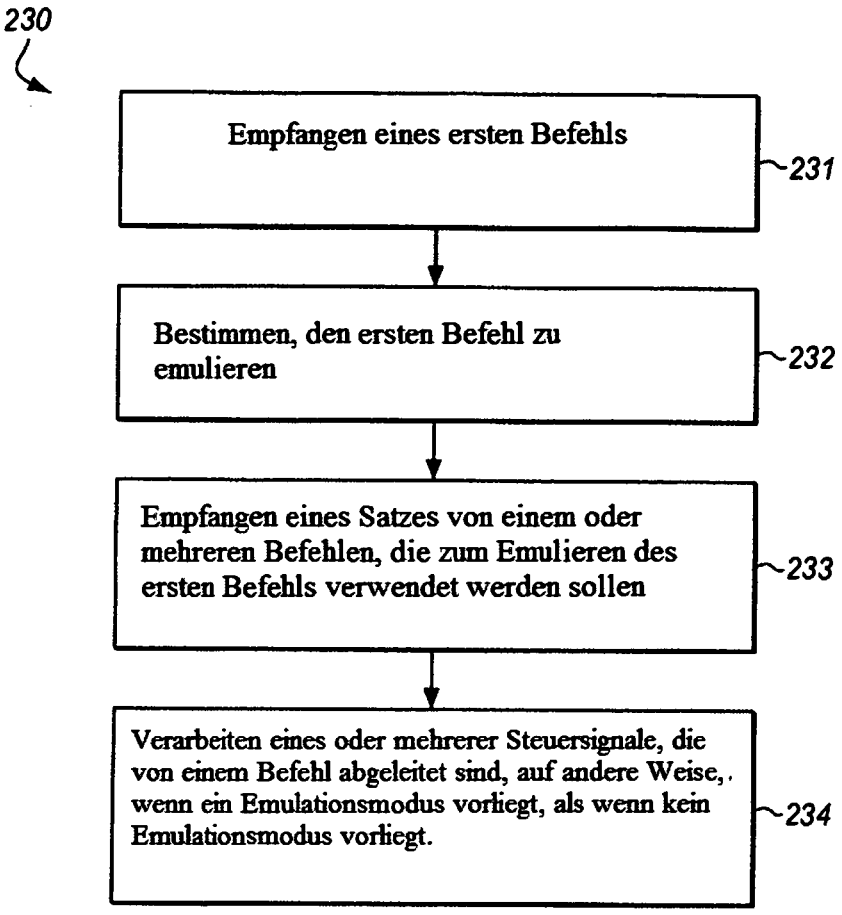


FIG. 1

Verfahren in einem Prozessor

FIG. 2



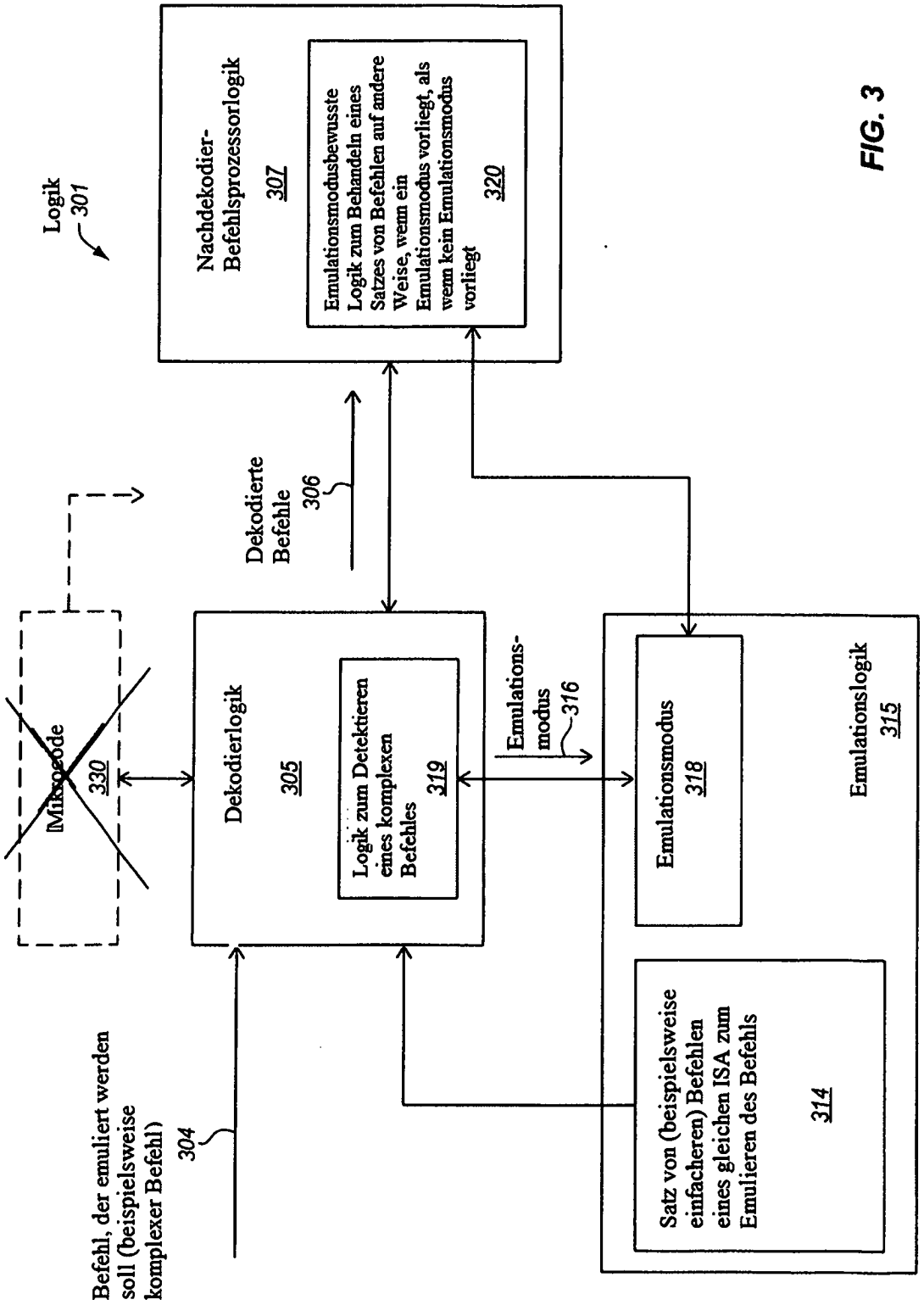

Logik
301
Nachdekodier-Befehlsprozessorlogik
307
Emulationsmodusbewusste Logik zum Behandeln eines Satzes von Befehlen auf andere Weise, wenn ein Emulationsmodus vorliegt, als wenn kein Emulationsmodus vorliegt
320
Dekodierte Befehle
306
Mikrocode
330
Dekodierlogik
305
Logik zum Detektieren eines komplexen Befehles
319
Emulations-modus
316
Emulationsmodus
318
Emulationslogik
315
Satz von (beispielsweise einfacheren) Befehlen eines gleichen ISA zum Emulieren des Befehls
314
Befehl, der emuliert werden soll (beispielsweise komplexer Befehl)
304

FIG. 3

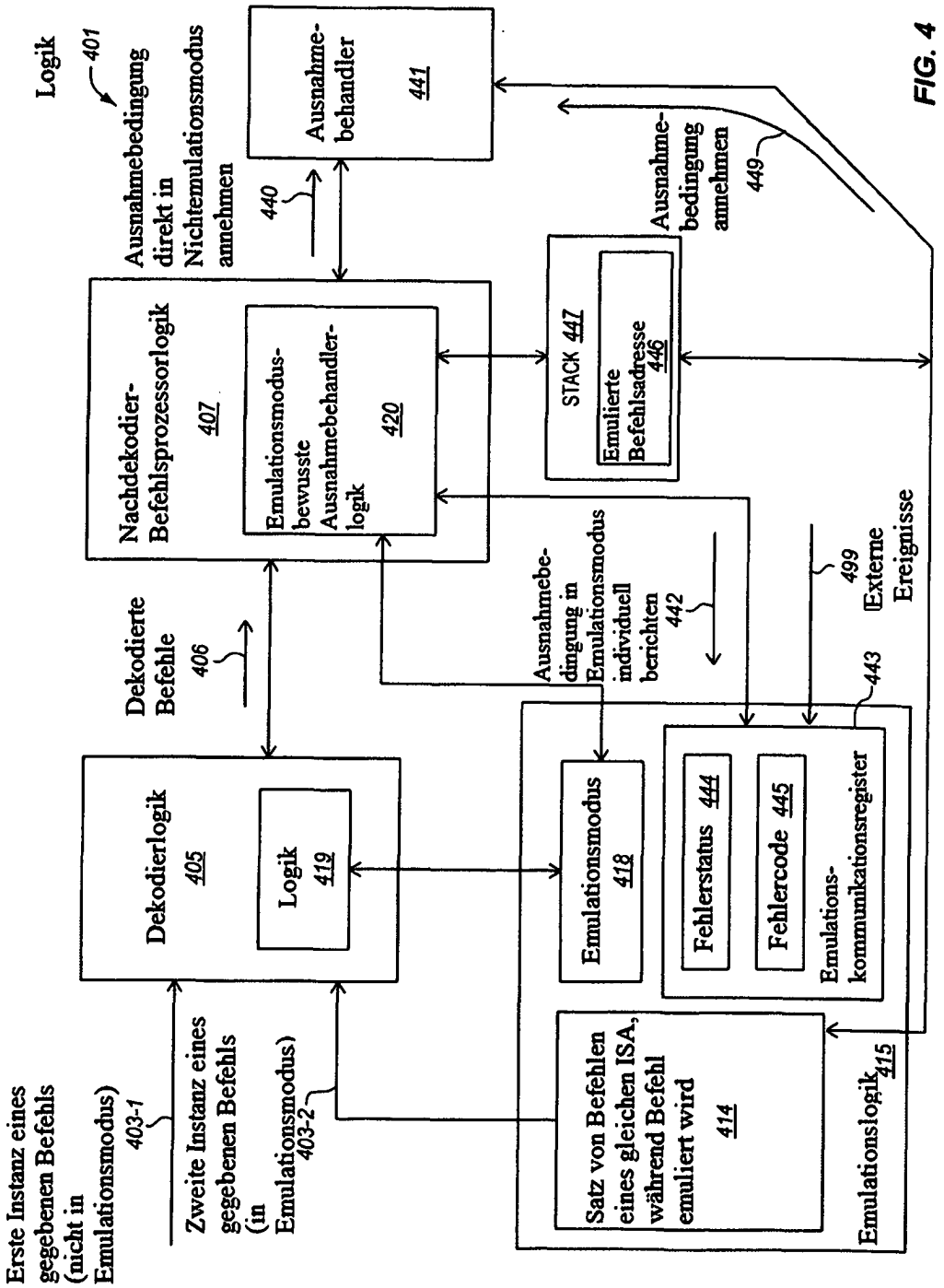

Logik
401
Ausnahmebedingung direkt in Nichtemulationsmodus annehmen
Ausnahme-behandler
441
440
Nachdekodier-Befehlsprozessorlogik
407
Emulationsmodus-bewusste Ausnahmebehandler-logik
420
STACK 447
Emulierte Befehlsadresse 446
Ausnahme-bedingung annehmen
449
Dekodierte Befehle
406
Ausnahmebe-dingung in Emulationsmodus individuell berichten
442
499
Externe Ereignisse
443
Dekodierlogik
405
Logik
419
Emulationsmodus
418
Fehlerstatus 444
Fehlercode 445
Emulations-kommunikationsregister
Erste Instanz eines gegebenen Befehls (nicht in Emulationsmodus)
403-1
Zweite Instanz eines gegebenen Befehls (in Emulationsmodus)
403-2
Satz von Befehlen eines gleichen ISA, während Befehl emuliert wird
414
Emulationslogik
415

FIG. 4

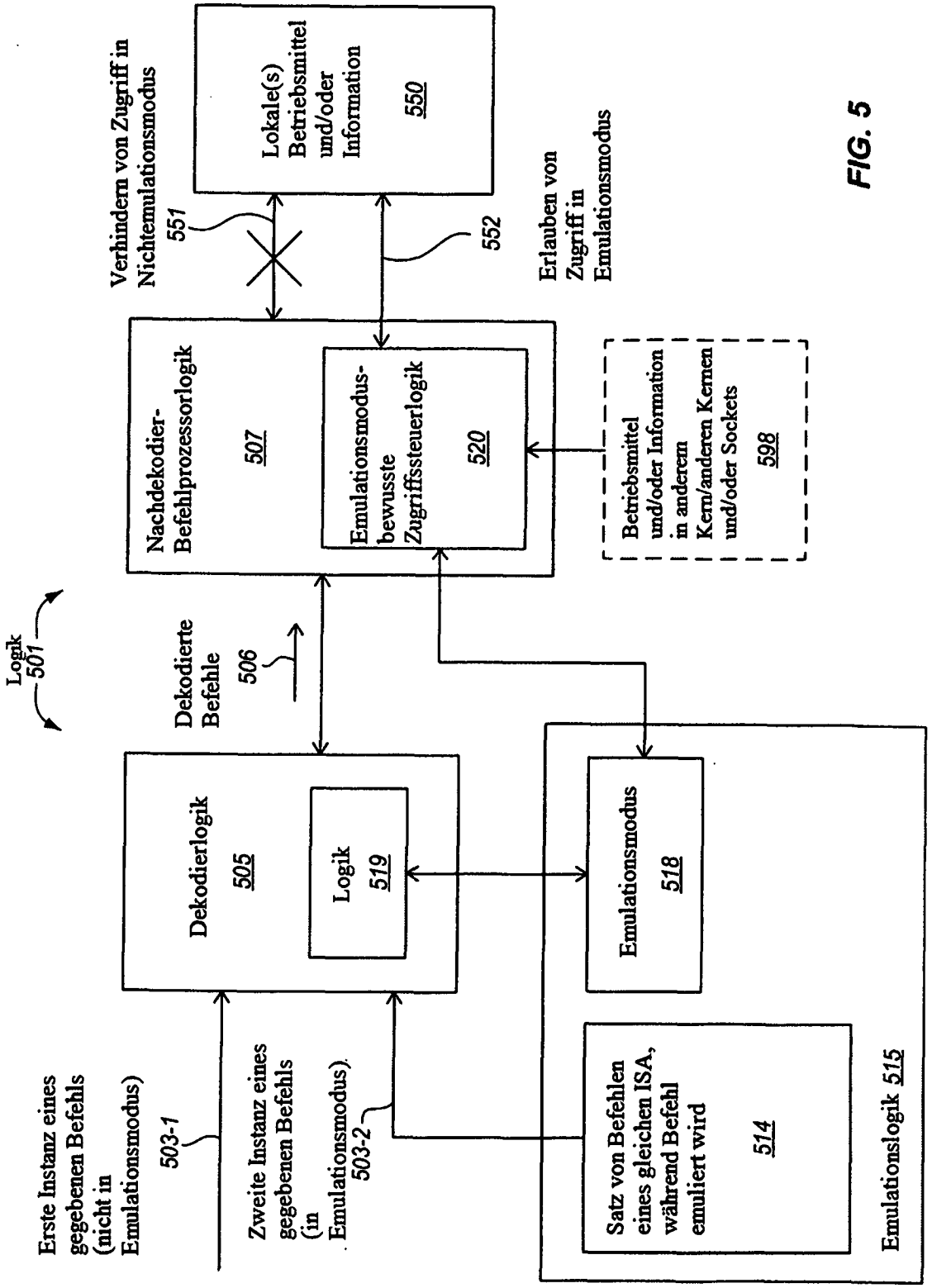

Logik 501
Dekodierte Befehle 506
Dekodierlogik 505
Logik 519
Nachdekodier-Befehlprozessorlogik 507
Emulationsmodus-bewusste Zugriffssteuerlogik 520
Verhindern von Zugriff in Nichtemulationsmodus
551
552
Lokale(s) Betriebsmittel und/oder Information 550
Erlauben von Zugriff in Emulationsmodus
Betriebsmittel und/oder Information in anderem Kern/anderen Kernen und/oder Sockets 598
Erste Instanz eines gegebenen Befehls (nicht in Emulationsmodus) 503-1
Zweite Instanz eines gegebenen Befehls (in Emulationsmodus) 503-2
Emulationsmodus 518
Satz von Befehlen eines gleichen ISA, während Befehl emuliert wird 514
Emulationslogik 515

FIG. 5

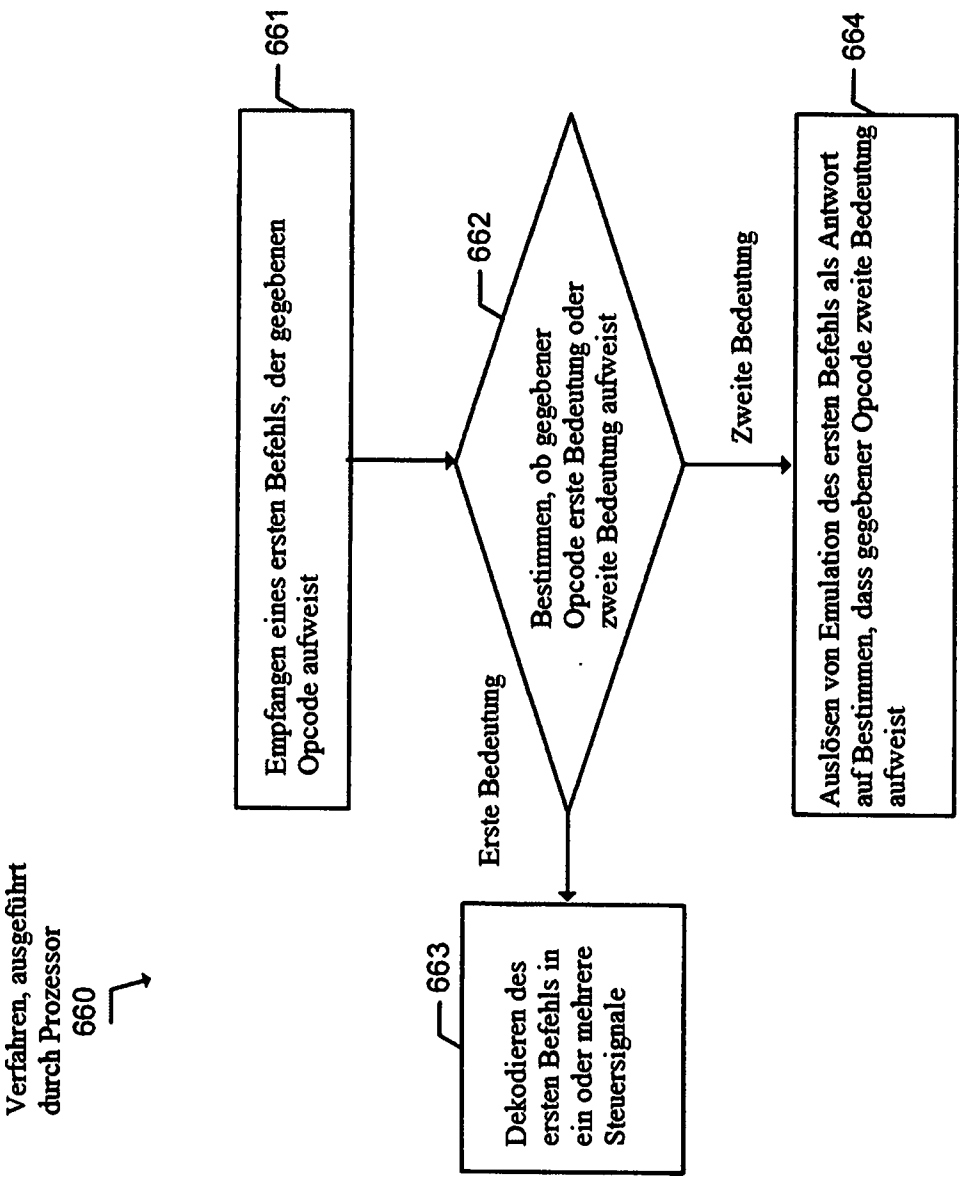

Verfahren, ausgeführt durch Prozessor
660
Empfangen eines ersten Befehls, der gegebenen Opcode aufweist
661
Bestimmen, ob gegebener Opcode erste Bedeutung oder zweite Bedeutung aufweist
662
Erste Bedeutung
Zweite Bedeutung
Dekodieren des ersten Befehls in ein oder mehrere Steuersignale
663
Auslösen von Emulation des ersten Befehls als Antwort auf Bestimmen, dass gegebener Opcode zweite Bedeutung aufweist
664

FIG. 6

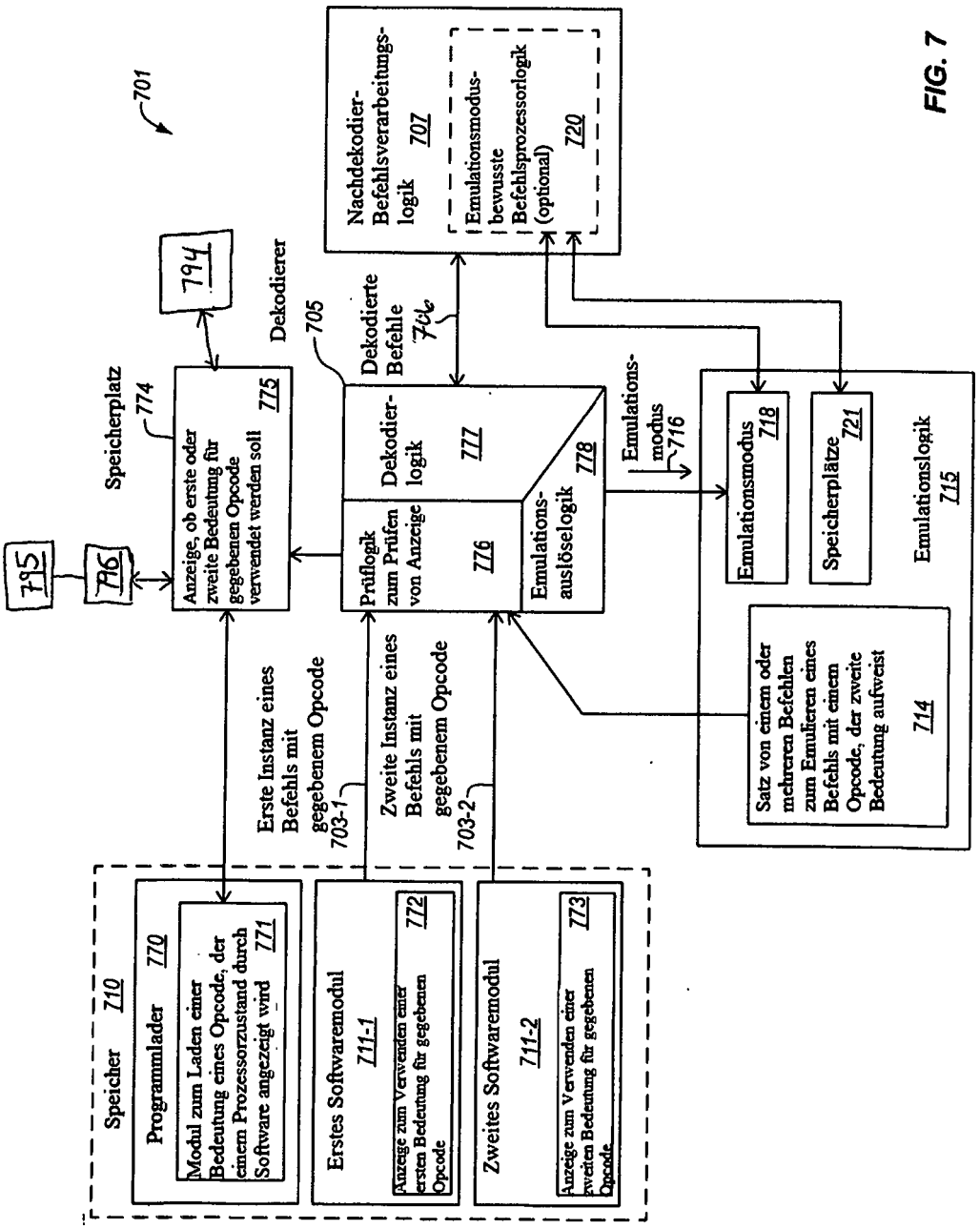

701
Speicher 710
Programmlader 770
Modul zum Laden einer Bedeutung eines Opcode, der einem Prozessorzustand durch Software angezeigt wird 771
Erstes Softwaremodul 711-1
Anzeige zum Verwenden einer ersten Bedeutung für gegebenen Opcode 772
Zweites Softwaremodul 711-2
Anzeige zum Verwenden einer zweiten Bedeutung für gegebenen Opcode 773
Erste Instanz eines Befehls mit gegebenem Opcode 703-1
Zweite Instanz eines Befehls mit gegebenem Opcode 703-2
Speicherplatz 774
Anzeige, ob erste oder zweite Bedeutung für gegebenen Opcode verwendet werden soll 775
Dekodierer
705
Prüflogik zum Prüfen von Anzeige 776
Dekodier-logik 777
Emulations-auslöselogik 778
Dekodierte Befehle 706
Emulations-modus 716
Nachdekodier-Befehlsverarbeitungs-logik 707
Emulationsmodus-bewusste Befehlsprozessorlogik (optional) 720
Emulationsmodus 718
Speicherplätze 721
Emulationslogik 715
Satz von einem oder mehreren Befehlen zum Emulieren eines Befehls mit einem Opcode, der zweite Bedeutung aufweist 714
794
795
796

FIG. 7

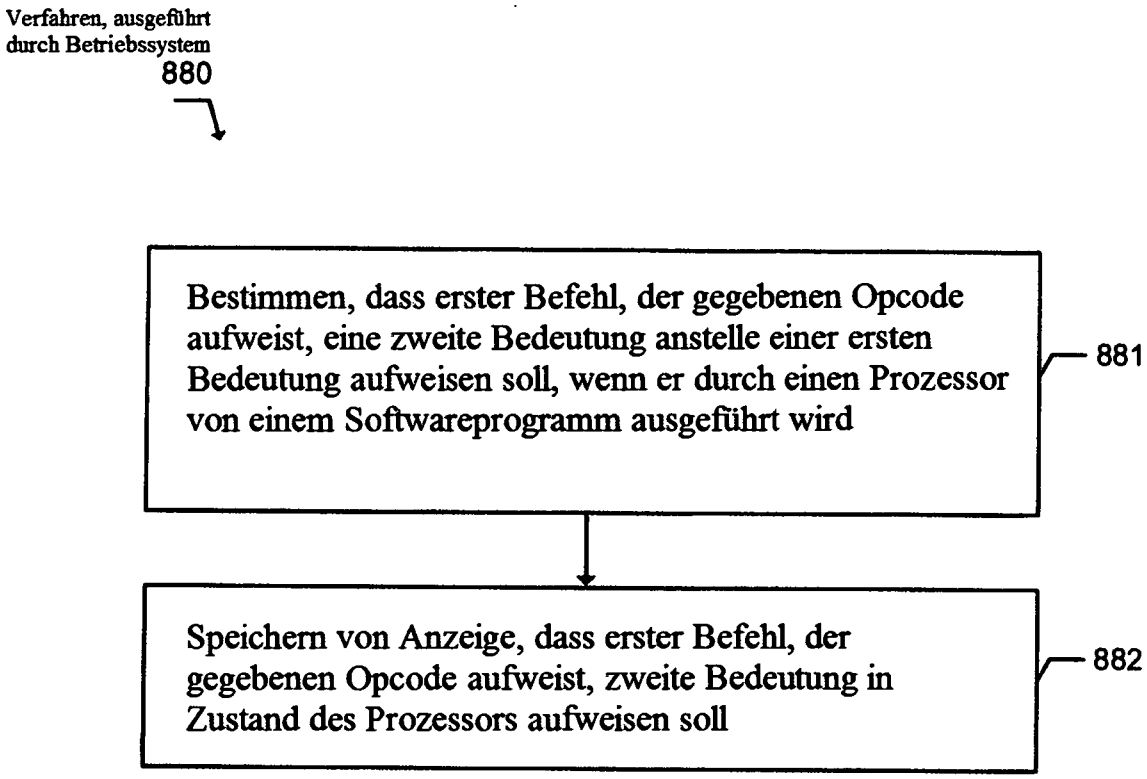

Verfahren, ausgeführt durch Betriebssystem
880
Bestimmen, dass erster Befehl, der gegebenen Opcode aufweist, eine zweite Bedeutung anstelle einer ersten Bedeutung aufweisen soll, wenn er durch einen Prozessor von einem Softwareprogramm ausgeführt wird
881
Speichern von Anzeige, dass erster Befehl, der gegebenen Opcode aufweist, zweite Bedeutung in Zustand des Prozessors aufweisen soll
882

FIG. 8

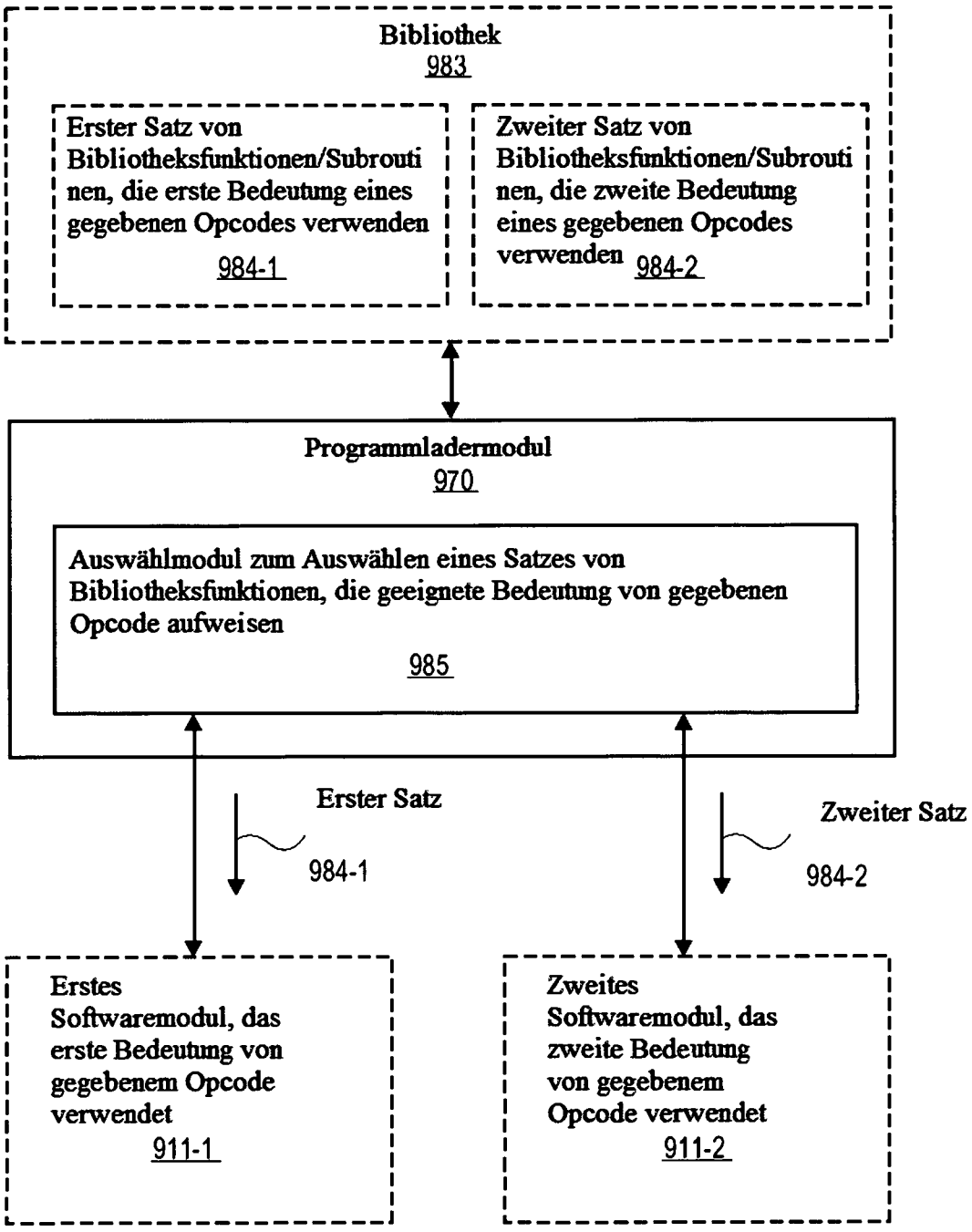

Bibliothek
983
Erster Satz von Bibliotheksfunktionen/Subroutinen, die erste Bedeutung eines gegebenen Opcodes verwenden
984-1
Zweiter Satz von Bibliotheksfunktionen/Subroutinen, die zweite Bedeutung eines gegebenen Opcodes verwenden
984-2
Programmladermodul
970
Auswählmodul zum Auswählen eines Satzes von Bibliotheksfunktionen, die geeignete Bedeutung von gegebenen Opcode aufweisen
985
Erster Satz
984-1
Zweiter Satz
984-2
Erstes Softwaremodul, das erste Bedeutung von gegebenem Opcode verwendet
911-1
Zweites Softwaremodul, das zweite Bedeutung von gegebenem Opcode verwendet
911-2

FIG. 9

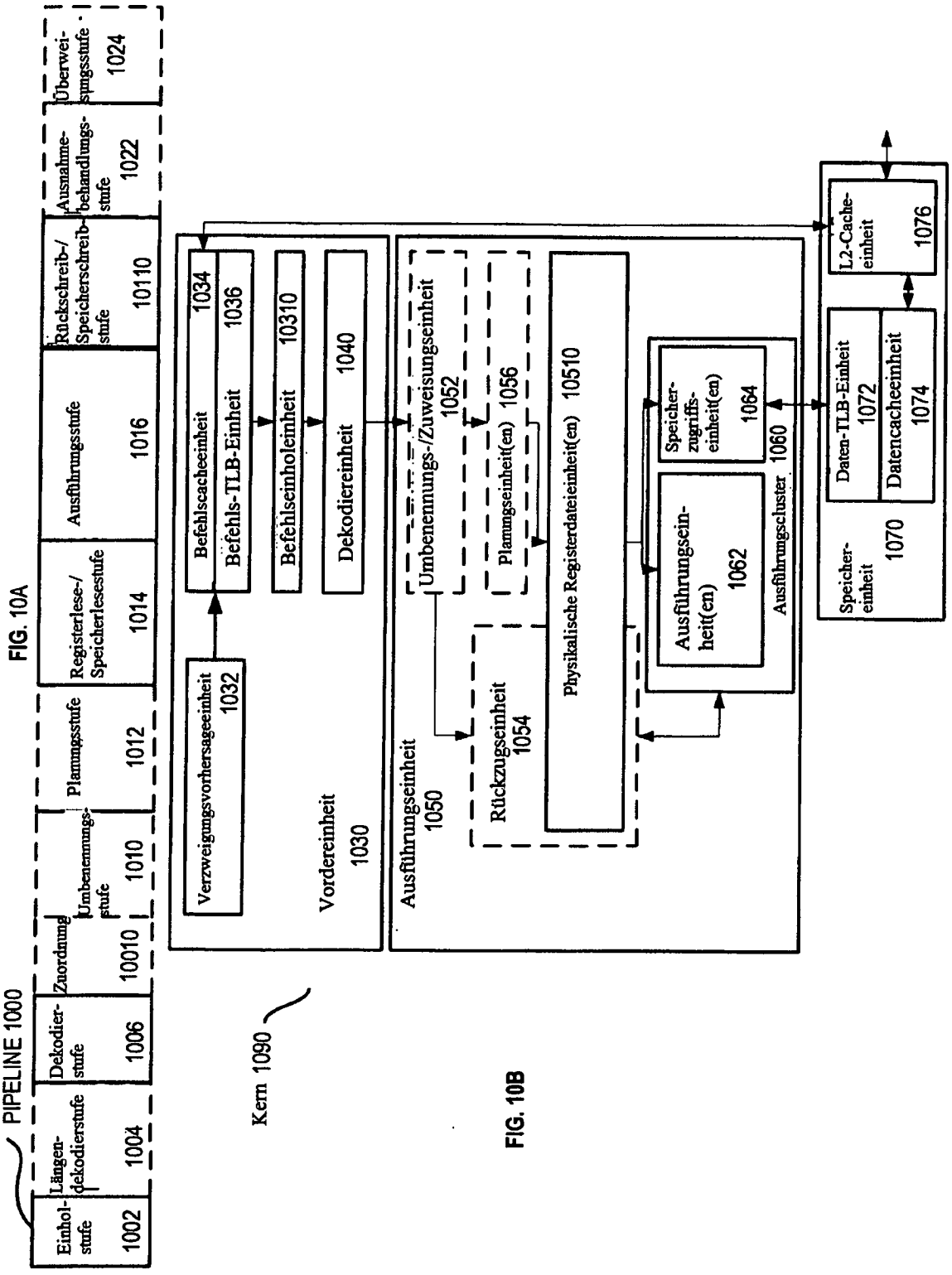


FIG. 10A

FIG. 10B

FIG. 11B

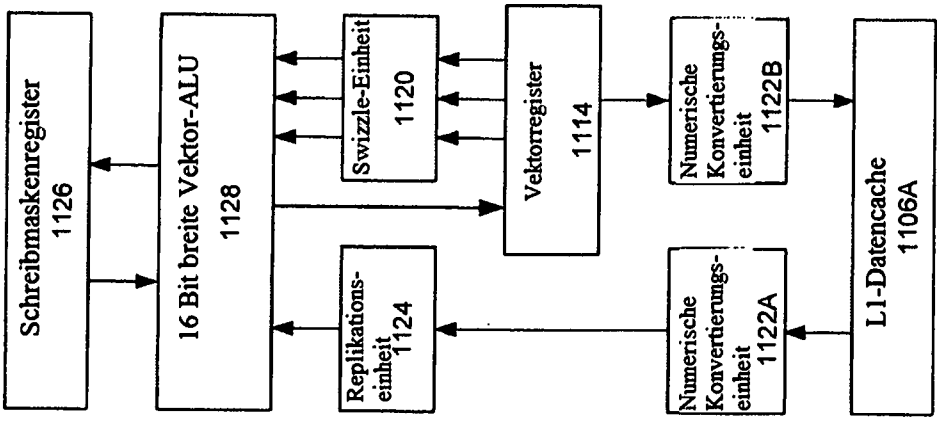
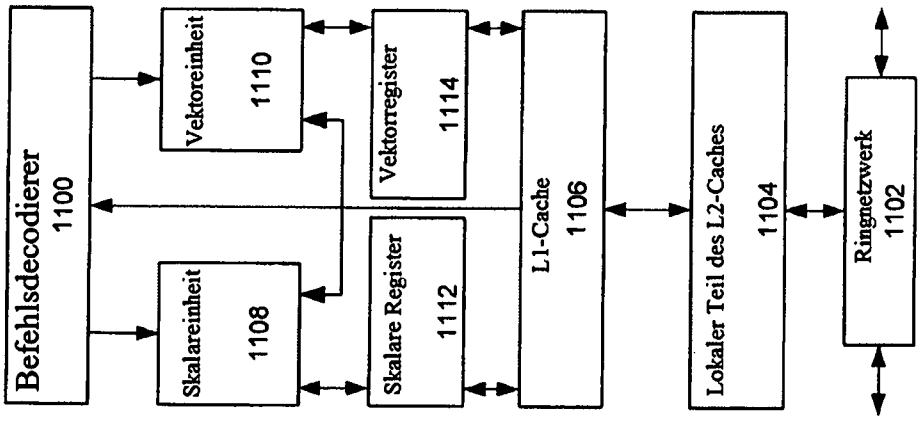


FIG. 11A



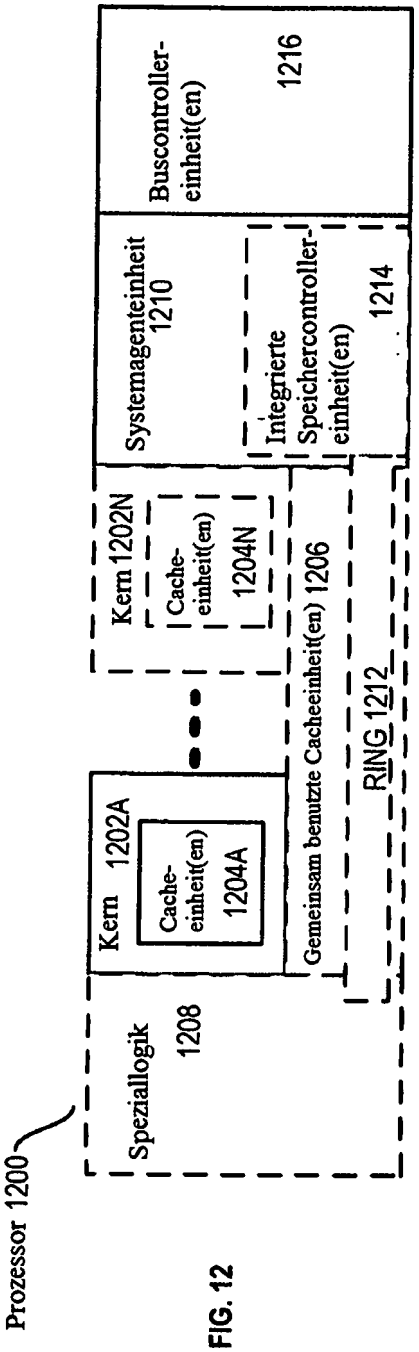

Prozessor 1200
Speziallogik 1208
Kern 1202A
Cache-einheit(en) 1204A
Kern 1202N
Cache-einheit(en) 1204N
Gemeinsam benutzte Cacheeinheit(en) 1206
RING 1212
Systemagenteinheit 1210
Integrierte Speichercontroller-einheit(en) 1214
Buscontroller-einheit(en) 1216

FIG. 12

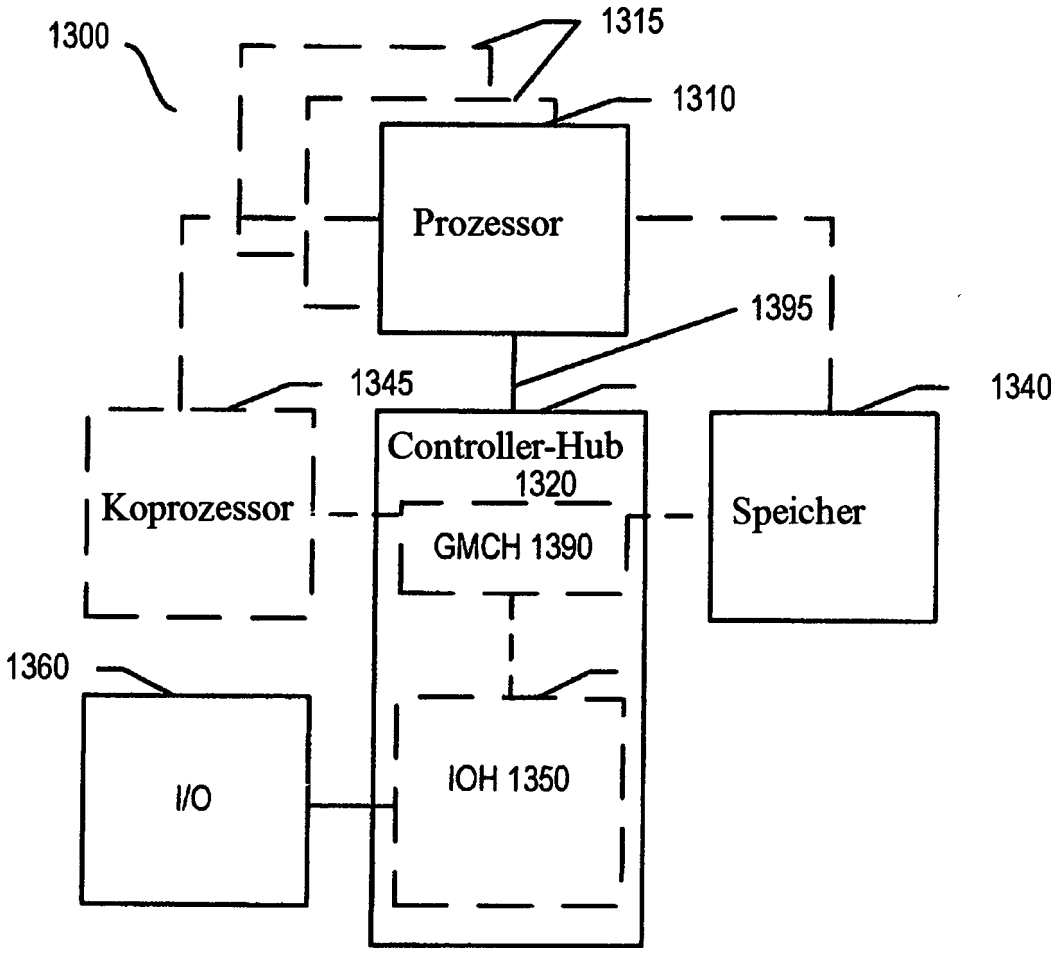

1300
1315
1310
Prozessor
1395
1345
1340
Controller-Hub
1320
Koprozessor
GMCH 1390
Speicher
1360
I/O
IOH 1350

FIG. 13

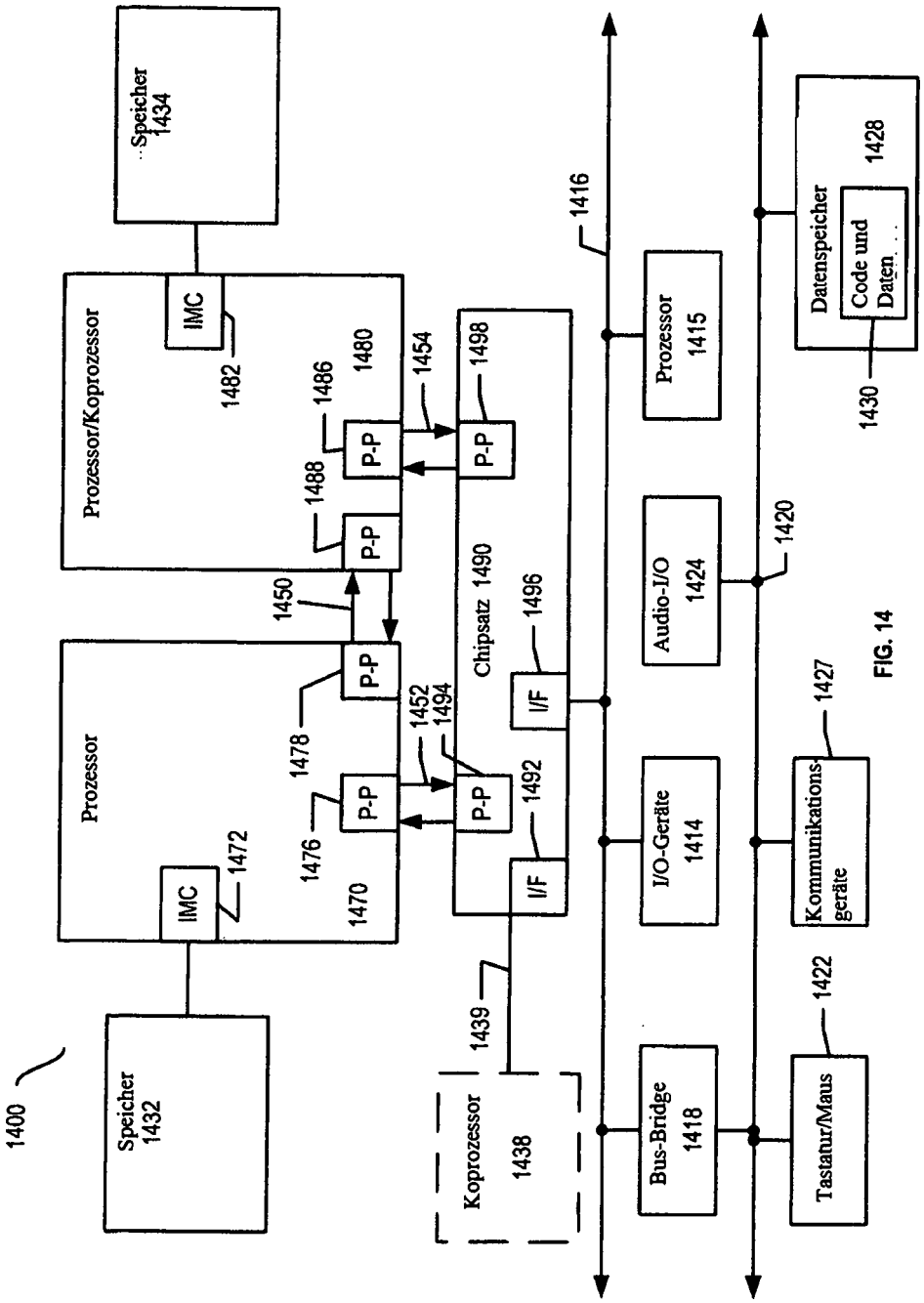

1400
Speicher 1432
Prozessor
IMC
1472
1470
Speicher 1434
Prozessor/Koprozessor
IMC
1482
1480
1486
1488
P-P
P-P
1450
1478
1476
P-P
P-P
1452
1494
1454
1498
P-P
P-P
Chipsatz 1490
1496
I/F
1492
I/F
1439
Koprozessor 1438
1416
Prozessor 1415
Audio-I/O 1424
1420
I/O-Geräte 1414
Kommunikations-geräte
1427
Bus-Bridge 1418
Tastatur/Maus
1422
Datenspeicher
Code und Daten
1428
1430

FIG. 14

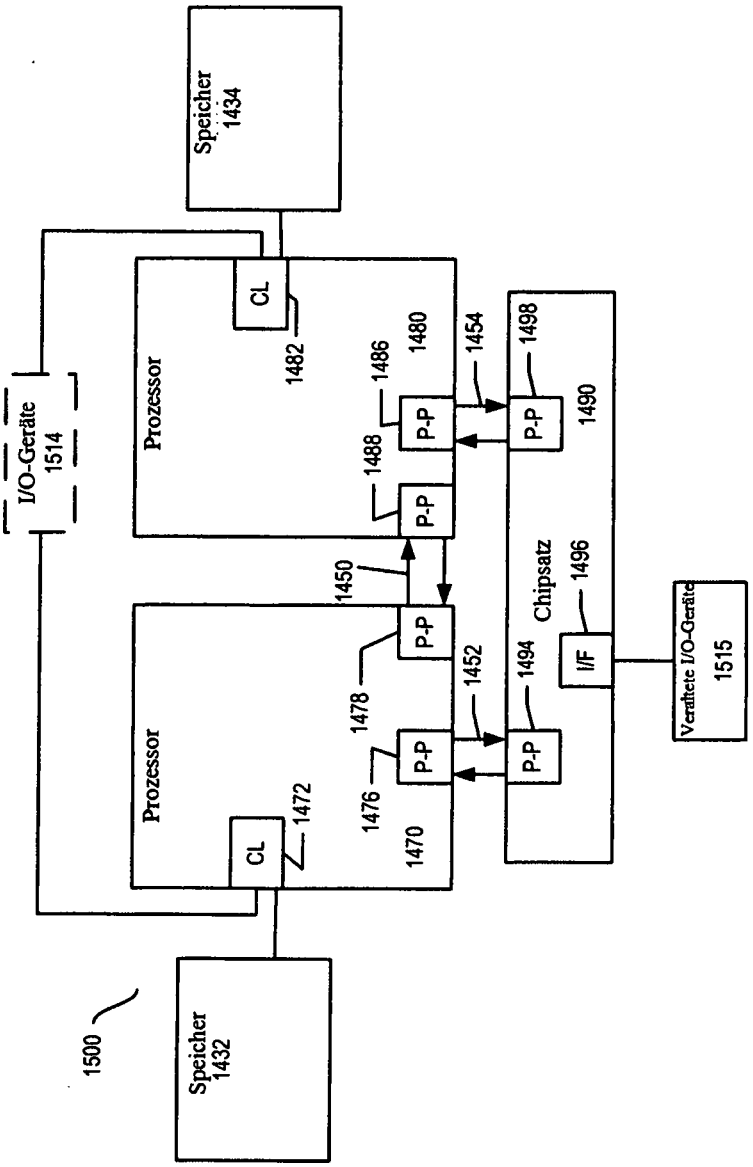

Speicher
1434
CL
1482
Prozessor
1480
1486
P-P
1488
P-P
1454
1498
P-P
1490
I/O-Geräte
1514
1450
P-P
1478
1452
1494
P-P
Chipsatz
1496
I/F
Verdrahtete I/O-Geräte
1515
1476
P-P
1470
Prozessor
CL
1472
1500
Speicher
1432

FIG. 15

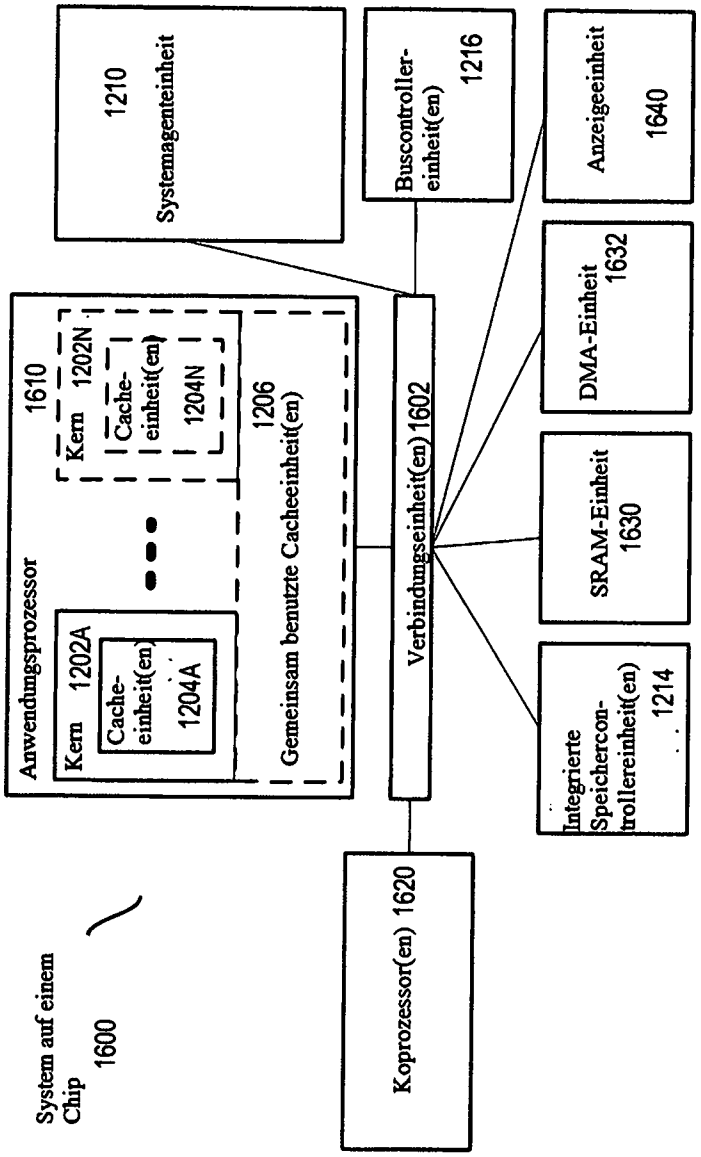

System auf einem Chip
1600
Anwendungsprozessor
1610
Kern 1202A
Cache-einheit(en) 1204A
Kern 1202N
Cache-einheit(en) 1204N
Gemeinsam benutzte Cacheeinheit(en) 1206
Koprozessor(en) 1620
Verbindungseinheit(en) 1602
Systemagenteinheit 1210
Buscontroller-einheit(en) 1216
Integrierte Speichercon-trollereinheit(en) 1214
SRAM-Einheit 1630
DMA-Einheit 1632
Anzeigeeinheit 1640

FIG. 16

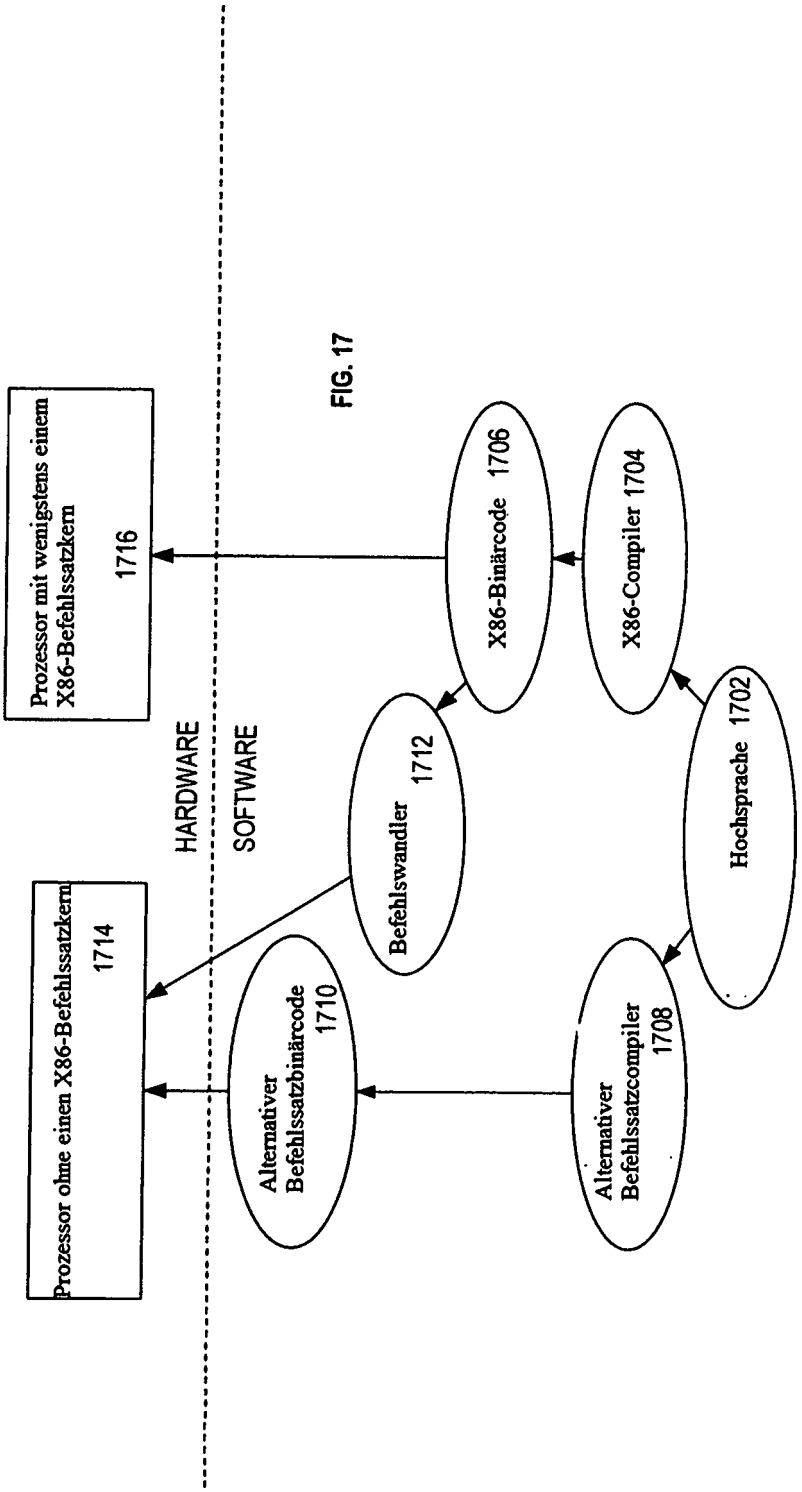

Prozessor mit wenigstens einem X86-Befehlssatzkern 1716
Prozessor ohne einen X86-Befehlssatzkern 1714
HARDWARE
SOFTWARE
X86-Binärcode 1706
X86-Compiler 1704
Befehlswandler 1712
Hochsprache 1702
Alternativer Befehlssatzbinärcode 1710
Alternativer Befehlssatzcompiler 1708

FIG. 17