(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2009-176326
(P2009-176326A)
(43)公開日　平成21年8月6日(2009.8.6)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	9/54	(2006.01)	G06F	9/06	640C	5B033
G06F	9/40	(2006.01)	G06F	9/40	310B	5B376

審査請求 有　請求項の数 9　ＯＬ　（全 31 頁）

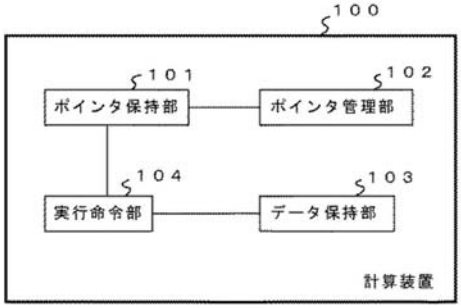
(21)出願番号　特願2009-113998 (P2009-113998)
(22)出願日　平成21年5月8日 (2009.5.8)
(62)分割の表示　特願2002-366337 (P2002-366337)の分割
原出願日　平成14年12月18日 (2002.12.18)

(71)出願人　502456574
株式会社　インテック・ネットコア
東京都　江東区　新砂　一丁目３番３号
(74)代理人　230104019
弁護士　大野　聖二
(74)代理人　100105038
弁理士　田中　久子
(72)発明者　中川　郁夫
東京都江東区新砂一丁目３番３号　株式会社インテック・ネットコア内
(72)発明者　永見　健一
東京都江東区新砂一丁目３番３号　株式会社インテック・ネットコア内

Ｆターム(参考)　5B033 EA01
5B376 CA22

(54)【発明の名称】計算装置、計算プログラム及び計算方法

(57)【要約】
【課題】
停止することなく、機能の変更・追加・削除ができる計算装置を提供する。
【解決手段】
計算を行なう実行部を特定するポインタを保持するポインタ保持部１０１と、ポインタ保持部で保持されているポインタを変更するポインタ管理部１０２と、実行部の実行に利用されるデータを保持するデータ保持部１０３と、データ保持部１０３で保持されているデータを利用して、ポインタ保持部１０１で保持されているポインタで指示される実行部に計算の実行をさせる実行命令部１０４を有する計算装置１００を提供する。このような計算装置により、実行部の再読込、追加、削除に伴い、ポインタ管理部１０２によりポインタを変更することにより、動的に機能の変更・追加・削除が可能となる。
【選択図】図１



【特許請求の範囲】
【請求項１】
計算の実行のための実行部を指示するためのポインタを保持するポインタ保持部と、
ポインタ保持部に保持されるポインタを所定の変更条件に従って変更するポインタ管理部と、
前記ポインタの変更によっても前記実行に利用されるデータをそのまま保持可能なデータ保持部と、
所定の実行条件に従って、前記データ保持部で保持されているデータを利用して、前記ポインタ保持部に保持されているポインタで指示される実行部により計算の実行をさせるための実行命令部と、
を有する計算装置。
【請求項２】
前記ポインタ管理部は、所定の条件に従ってポインタの追加と削除とを行なう機能をさらに有する請求項１に記載の計算装置。
【請求項３】
前記ポインタ管理部は、実行部の再読込をする実行部再読込手段を有し、
前記実行部再読込手段により再読込された実行部を指示するように前記ポインタを変更することを特徴とする請求項１に記載の計算装置。
【請求項４】
前記実行部再読込手段は、データ保持部に保持されているデータを変換するためのデータ変換実行部をも読み込みことを特徴とする請求項３に記載の計算装置。
【請求項５】
前記データはバージョン情報を保持し、前記実行部は、前記バージョン情報に応じた計算の実行を行なうことを特徴とする請求項３に記載の計算装置。
【請求項６】
前記ポインタ管理部は、実行部を追加する実行部追加手段を有し、
前記実行部追加手段により追加された実行部を指示するポインタを前記ポインタ保持部に追加することを特徴とする請求項２に記載の計算装置。
【請求項７】
ポインタ管理部は、実行部を削除する実行部削除手段を有し、
前記実行部追加手段により削除された実行部を指示している前記ポインタの削除又は変更を行なうことを特徴とする請求項２に記載の計算装置。
【請求項８】
計算の実行のための実行部を指示するためのポインタを保持するポインタ保持ステップと、
ポインタ保持ステップにより保持されたポインタを所定の変更条件に従って変更するポインタ管理ステップと、
前記ポインタの変更によっても前記実行に利用されるデータをそのまま保持可能に保持するデータ保持ステップと、
所定の実行条件に従ってデータ保持ステップで保持されたデータを利用して、前記ポインタ保持ステップにより保持されたポインタで指示される実行部により計算の実行をさせるための実行命令ステップと、
を計算機に実行させるための計算プログラム。
【請求項９】
計算の実行のための実行部を指示するためのポインタを保持するポインタ保持ステップと、
ポインタ保持ステップにより保持されたポインタを所定の変更条件に従って変更するポインタ管理ステップと、
前記ポインタの変更によっても前記実行に利用されるデータをそのまま保持可能に保持するデータ保持ステップと、

所定の実行条件に従ってデータ保持ステップで保持されたデータを利用して、前記ポインタ保持ステップにより保持されたポインタで指示される実行部により計算の実行をさせるための実行命令ステップと、
を含む計算方法。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、計算装置、計算プログラム及び計算方法に関する。特に、ポインタにより計算を実行する実行部を指示し、その特定された実行部により、計算を実行するものに関する。
【背景技術】
【０００２】
従来、計算機の中でプログラムを実行する場合は、定義されていない関数（ｆｕｎｃｔｉｏｎ）、手続き（ｐｒｏｃｅｄｕｒｅ）、変数（ｖａｒｉａｂｌｅ）があってはならず、例えば、高級プログラミング言語で記述されたプログラムをコンパイルして実行可能なオブジェクトを得るには、プログラムで参照される全ての関数、手続き、変数が定義されている必要があった。
【０００３】
ここに「プログラムで参照される全ての関数、手続き、変数が定義されている必要がある」とは、実行可能なオブジェクトから参照される関数、手続き、がコード（インストラクション）部分に存在し、また、実行可能なオブジェクトから参照される変数が保持する値がデータ部分に存在することを意味する。「コード（インストラクション）」、「データ」については、以下の段落で説明する。
【０００４】
図２３は、プログラムを計算機で実行する際に使用されるアドレス空間２３００を模式的に示している。アドレス空間２３００の上方が小さいアドレスに対応し、下方が大きいアドレスに対応するとすると、アドレスが小さい部分には、コード（インストラクション）とデータが保持される。その次にヒープと呼ばれる領域がある。また、アドレスが大きい方からスタックの領域が伸びている。
【０００５】
コード（インストラクション）は、関数や手続きを実行するための計算機への命令を含む部分であり、コードの部分に含まれる命令で参照される関数、手続きがアドレス空間内に配置されるようになっていなければならない。またデータは、プログラミング言語で定義される大域変数の値を保持する。ヒープは、プログラムの実行に伴い、動的に確保されるメモリの領域であり、メモリがｍａｌｌｏｃ関数などにより動的に確保されると、矢印２３０１が示すように、ヒープの一端がアドレスの大きい方に向かって伸びて行く。スタックは、関数や手続きを呼び出すときに記憶すべき戻りアドレスを蓄積したり、局所変数を保持したりするためのメモリ領域である。スタックは、関数や手続きを呼び出すと、矢印２３０２が示すように、アドレスの小さい方に向かって伸びて行く。
【０００６】
しかし、近年、共有ライブラリの技術が使用可能となり、実行可能なオブジェクトを得る場合には、共有ライブラリの中に定義されている関数、手続き、変数については、コンパイル時に定義されている必要がなくなった（例えば、特許文献１参照。）。これにより、実行可能なオブジェクトが実際に実行されるときに、未定義の関数、手続き、変数が定義を含む共有ライブラリが存在すればよくなり、実行可能なオブジェクトに含まれる未定義の関数などは、実行開始時に共有ライブラリと動的にリンクが行なわれることによりアドレスが解決されるようになった。
【０００７】
共有ライブラリの採用により、例えば、ライブラリ中の関数のバグの修正を反映するため

の再コンパイルが不要となったり、実行可能オブジェクトのサイズが小さくなったりするという効果が得られる。

【０００８】

なお、共有ライブラリにより提供される関数、手続きを実行するための命令や、共有ライブラリにより提供される変数の値は、ヒープとスタックの間のメモリ空間に配置されるようになる。特に、そのメモリ空間は、他のプロセスと共有される共有メモリとすることが一般的である。

【先行技術文献】

【特許文献】

【０００９】

【特許文献１】特開平０６－３３２６７５号公報（第２－３ページ）

【発明の概要】

【発明が解決しようとする課題】

【００１０】

しかしながら、共有ライブラリを使用したとしても、実行可能なオブジェクトを実行している途中で、関数や手続きの定義を入れ換えることはできない。このため、例えば、バグが発見された場合、実行可能なオブジェクトの実行を停止させ、新たな実行可能なオブジェクト、あるいは、共有ライブラリを入れ換えて、プログラムを再起動する必要がある。このことは、計算機によるサービスが必要不可欠となった現代において、問題をもたらす。すなわち、実行可能なオブジェクトや共有ライブラリの入れ換えのために、そのオブジェクトや共有ライブラリにより提供されているサービスの提供を中断しなければならない。

【００１１】

このようなサービスの中断は、例えば、ウェブサーバで２４時間絶え間なくサービスを提供する場合に問題となる。

【００１２】

また、このようなサービスの中断は、特に、インターネットなどの通信に用いられているルータにおいて、特に問題となる。

【００１３】

図２２は、ルータ２２０３、２２０４、２２０５、２２０６によりｓｉｔｅ－Ａとｓｉｔｅ－Ｂとが通信可能な状態を例示している。この場合、実線の矢印のようにｓｉｔｅ－Ｂからｓｉｔｅ－Ａへデータが送信されるようにするためには、ｓｉｔｅ－Ａへ至る経路情報を各ルータ２２０３、２２０４、２２０５、２２０６が持っている必要がある。このため、破線で示した向きに経路情報が流れる。また、同様にｓｉｔｅ－Ａからｓｉｔｅ－Ｂへデータが送信できるようにするためには、実線で示した向きに経路情報が流れる必要がある。すなわち、ルータは、隣り合うルータと経路情報を交換し合うことになる。

【００１４】

このように通信に用いられるルータで動作する実行可能なオブジェクト（「ファームウェア」と呼ばれる場合がある。）を入れ換えるために、ルータを停止させると、その間、そのルータを介する通信が途絶してしまう。また、上述したように、ルータ間では、経路情報の交換が行なわれ、この交換により、ルータは、特定のサイトへデータを送るための最短の経路などの情報を計算し、その結果を蓄積しているが、ルータを停止させると、蓄積した情報が消去してしまう。このため、再起動してから再び情報を計算・蓄積しなければならなくなり、停止前の状態に復旧するのに時間が必要となる。

【００１５】

ルータによる最短経路の情報の計算について、現在用いられているアルゴリズムでは、ルータの数をＮ、通信路の数をＬとすると、以下の式のオーダーの計算量が必要である。

【００１６】

【数１】

$$N\times(N+L)$$

特に、Ｎの二乗に比例したオーダーとなるため、ルータの広く普及している現在では、復旧のためにはかなりの計算量が必要となり、それだけ復旧に時間がかかることになる。

【００１７】

また、このルータのファームウェアのバージョンアップの問題は、特にインターネットサービスプロバイダにおいて、顕著となる。一つのインターネットサービスプロバイダが、数千の台数のルータを有しているのは珍しくない。例えば、あるインターネットサービスプロバイダに五千台のルータがあり、ルータのファームウェアのバージョンアップが年三回あるとすると、一年で延べ一万五千回のバージョンアップを行なう必要がある。平均すると一日約五十台のルータの停止を行ない、バージョンアップを行なうことになり、ルータの停止の問題は深刻である。

【００１８】

そこで、本発明では、プログラムの実行を停止させることなく、関数や手続きなどを入れ換えることができるようにすることを目的とする。

【課題を解決するための手段】

【００１９】

この課題を解決するために、本発明は、第一に、計算の実行のための実行部を指示するポインタを保持し、そのポインタを所定の条件に従って変更し、その変更によっても実行に利用されるデータをそのまま保持可能であり、また、所定の実行条件に従って、保持されているデータを利用して、ポインタで指示される実行部により計算の実行をさせる計算装置を提供する。

【００２０】

これにより、ポインタを変更することにより、実行がされる実行部を変更することができる。また、ポインタが変更されても、データはそのまま保持されているので、継続的にサービスを提供することができる。

【００２１】

第二に、所定の条件に従って、ポインタの追加と削除を行なう計算装置を提供する。

【００２２】

これにより、新しいポインタを追加することにより、今まで実行がされなかった実行部を実行することができたり、既に存在する実行部が実行されないようにしたりすることができる。

【００２３】

第三に、実行部の再読込を行ない、再読込がされた実行部をポインタが指示するようにする計算装置を提供する。

【００２４】

これにより、例えば、バグなどを有する実行部の代わりに、バグが除去された実行部が実行されるようにすることができる。

【００２５】

第四に、実行部を追加し、追加された実行部をポインタが指示するようにする計算装置を提供する。

【００２６】

これにより、サービスの提供を停止することなく、新しい機能を追加することができる。

【００２７】

第五に、実行部を削除し、削除された実行部を指示しているポインタの削除または変更を行なう計算装置を提供する。

【００２８】

これにより、不要となったり、バグがあったりなどして使用できない実行部を削除するこ

とができる。
【００２９】
また、本発明では、このような計算装置を計算機により実現するための計算プログラムも提供される。また、このような計算装置や計算プログラムなどにより用いられる計算方法も提供される。
【発明の効果】
【００３０】
以上のように、本発明によれば、停止させることなく、実行部の変更が可能となり、計算装置、計算プログラムのバージョンアップなどが可能となる。
【図面の簡単な説明】
【００３１】
【図１】計算装置の機能ブロック図
【図２】ポインタ保持部と実行部の関係を示す一例図
【図３】アドレス空間とそこに配置された実行部とポインタの一例図
【図４】実行部とポインタとのプログラミング言語での表現の一例図
【図５】ポインタ管理部によるポインタの変更を示す一例図
【図６】ポインタが複数存在する場合の一例図
【図７】ポインタがリスト構造に保持されている場合の一例図
【図８】リスト構造を構成するリストセルのプログラミング言語による表現の一例図
【図９】実行命令部のプログラミング言語での表現の一例図
【図１０】リスト構造においてポインタを追加する一例図
【図１１】ポインタを追加する操作のプログラミング言語による表現の一例図
【図１２】ポインタを削除する操作のプログラミング言語による表現の一例図
【図１３】特定のポインタを探す操作のプログラミング言語による表現の一例図
【図１４】本発明によりイベントを処理するためのフローチャート
【図１５】イベントと、ポインタ及びデータと、を対応付けたデータ構造の一例図
【図１６】計算装置の機能ブロック図
【図１７】計算装置の機能ブロック図
【図１８】実行部を追加する処理のフローチャート
【図１９】計算装置の機能ブロック図
【図２０】実行部を削除する処理のフローチャート
【図２１】削除される実行部を探す操作のプログラミング言語による表現の一例図
【図２２】ルータによって構成されるネットワークの一例図
【図２３】アドレス空間の使用の態様の一例図
【発明を実施するための形態】
【００３２】
以下、本発明の実施の形態について、図を用いて説明する。なお、本発明は、これら実施の形態に何ら限定されるものではなく、その要旨を逸脱しない範囲において、種々なる態様で実施し得る。
【００３３】
（実施形態１（主に請求項１、８、９に関連する））
本発明の実施形態１においては、計算の実行のための実行部を指示するポインタを保持し、そのポインタを所定の条件に従って変更し、その変更によっても実行に利用されるデータをそのまま保持可能であり、また、所定の実行条件に従って、保持されているデータを利用して、ポインタで指示される実行部により計算の実行をさせる計算装置、計算プログラム及び計算方法が提供される。
【００３４】
（実施形態１：構成）
図１は、実施形態１における計算装置の機能ブロック図を例示している。計算装置１００は、ポインタ保持部１０１と、ポインタ管理部１０２と、データ保持部１０３と、実行命

令部１０４と、を有する。
【００３５】
（実施形態１：ポインタ保持部）
「ポインタ保持部」１０１は、実行部を指示するためのポインタを保持する。「実行部」とは、計算の実行のための部である。
【００３６】
図２は、ポインタ保持部と実行部との関係を例示している。ポインタ保持部１０１の中にポインタ２０１が存在し、実行部２０２を指示している。
【００３７】
（実施形態１：ポインタ保持部：アドレス空間での例）
図３は、ポインタ保持部と実行部との関係がアドレス空間上でどのようになっているかを例示する。アドレス空間３００の中に、実行部に対応する部分として斜線を引いた部分３０１が存在する。この部分３０１が実行部に該当し、例えば、関数を実行するための命令が格納される。符号３０２を付けた部分がポインタ保持部に該当する。この部分には、例えば、斜線を引いた部分３０１の最初のアドレスが格納される。
【００３８】
（実施形態１：ポインタ保持部：プログラミング言語での表現）
図４は、プログラミング言語上で、ポインタ保持部と実行部とがどのように表現されているかを例示する。図４では、プログラミング言語としては、Ｃ言語を用いている。
【００３９】
（ａ）の部分が、変数ｆｐを、ポインタを保持する大域変数であるポインタ変数として定義している。この変数ｆｐがポインタ保持部となる。
【００４０】
（ｂ）の部分が関数ｆの定義である。この関数が実行部に相当する。
【００４１】
（ｃ）の部分は関数ｍａｉｎの定義である。Ｃ言語では、ｍａｉｎという名前の関数は特別な意味を持ち、プログラムの実行の最初に呼び出される関数となる。
【００４２】
（ｄ）の部分は、ポインタ変数ｆｐが関数ｆを指示するように代入を行なっている。この代入は、図３で斜線の部分３０１が関数ｆに対応し、符号３０２を付けた部分が変数ｆｐに対応するメモリとすると、符号３０２を付けた部分に、斜線の部分３０１の最初のアドレスを格納することに相当する。
【００４３】
（実施形態２：ポインタ保持部：ポインタ保持部は複数のポインタを保持してもよい）
なお、ポインタ保持部１０１は、複数のポインタを保持してもよい。
【００４４】
図６は、ポインタ保持部１０１が複数のポインタ６０１、６０２、…、６０３を保持している様子を例示している。ポインタが、それぞれが実行部６０４、６０５、…、６０６を指示している。また、図６に示すように、ポインタにより指示されない実行部６０７、…、があってもよい。このような複数のポインタは、プログラミング言語上では、複数の変数によって表されたり、配列により表わされたりすることになる。
【００４５】
（実施形態２：ポインタ保持部：ポインタはリスト構造によって保持されていてもよい）
また、ポインタ保持部が保持するポインタは、リスト構造によって保持されていてもよい。
【００４６】
図７は、ポインタがリスト構造によって保持されている様子を例示している。リスト構造の先頭のリストセル７０２を指すポインタ７０１があり、このポインタ７０１により指されているリストセル７０２から他のリストセルへのポインタをたどることにより、全てのリストセルがたどることができるようになっている。

【００４７】
特に、リスト構造の最後のリストセル７０４には、次のリストセルが存在しないので、リストセルへのポインタには、次のリストセルが存在しないことを示す特別な値が格納されている。この特別な値として、プログラミング言語では、例えば、ＮＵＬＬという値が用いられる。
【００４８】
図８は、図７に例示したリスト構造を作るためのプログラミング言語上でのリストセルのデータ構造の定義を例示している。（ａ）の部分で定義されるｆｐが実行部を指示するポインタであり、（ｂ）の部分で定義されるｎｅｘｔが次のリストセルへのポインタである。
【００４９】
（実施形態１：ポインタ管理部）
「ポインタ管理部」１０２は、ポインタ保持部１０１に保持されるポインタを所定の変更条件に従って変更する。「所定の変更条件に従って変更する」とはあらかじめ決められた条件が成立した場合に変更するということであり、例えば、計算装置１００に備えられた特定のボタンが押された場合、キーボードの特定のキーが押された場合、あるいは、キーボードから特定の文字列が入力された場合、計算装置１００にデータの受信部が備えられており、その受信部によりデータが受信された場合、などに、ポインタを変更する。
【００５０】
なお、「ポインタを変更する」とは、既に存在するポインタが別の実行部を指示するようにすることである。
【００５１】
図５は、ポインタの変更を例示する。（ａ）において、ポインタ保持部１０１に保持されるポインタ２０１が実行部５０１を指示している。あらかじめ決められた条件が成立すると、（ｂ）に例示されるように、ポインタ２０１が、実行部５０２を指示する。
【００５２】
（実施形態１：ポインタ管理部：プログラミング言語での表現）
プログラミング言語上では、ポインタ管理部は、あらかじめ決められた条件が成立したかどうかを判断し、もし、成立したならば、ポインタ変数に値を代入する記述により表現される。
【００５３】
（実施形態１：データ保持部）
「データ保持部」１０３は、前記ポインタの変更によっても前記実行に利用されるデータをそのまま保持可能な部である。「前記ポインタの変更」とは、ポインタ管理部１０２によるポインタの変更を意味する。「前記実行」とは、ポインタ保持部１０１に保持されているポインタで指示されている実行部による計算の実行を意味する。「そのまま保持可能」とは、前記ポインタの変更によっても、データ保持部１０３で保持されているデータは変更されないことを意味する。「変更」には、データそのものの変更のみならず、データの消去など、データが使用不能になることも含む。
【００５４】
したがって、データ保持部１０３は、データを保持するが、そのデータは、ポインタ保持部で保持されているポインタとは異なる領域で保持されていることになる。また、計算装置１００がプログラムを用いて計算機によって実現される場合には、ポインタが変更されても、そのプログラムの実行が停止するようなことは無いことを意味する。
【００５５】
（実施形態１：実行命令部）
「実行命令部」１０４は、所定の実行条件に従って、データ保持部１０３で保持されているデータを利用して、ポインタ保持部１０１で保持されているポインタで指示されている実行部により計算の実行をさせるための部である。
【００５６】

（実施形態１：実行命令部：所定の実行条件）
「所定の実行条件に従って」とは、あらかじめ決められた条件が成立した場合、ということを意味する。例えば、○特定のスイッチが押されたことが検出された場合、○キーボードから特定の文字列が入力された場合、○計算装置１００が外部からデータを受信するようになっている状況において、特定のデータが受信された場合、○計算装置１００が外部へデータを送信するようになっているとき、データが送信可能となった場合、○ある時点から一定の時間が経過した場合、などがある。なお、ポインタ管理部１０２の定義においても「所定の」という言葉が現われるが、実効命令部の定義における「所定の」と異なる意味であってもよい。
【００５７】
（実施形態１：実行命令部：プログラミング言語での表現）
図９は、実行命令部１０４などをプログラミング言語で表現したものを例示している。（ａ）において、データ保持部１０３で保持されるデータを指示するためのデータへのポインタ変数ｐが定義されており、（ｂ）において、そのデータへのポインタ変数ｐに０ｘ３７４６８ＡＢＢという値が代入されている。（ｃ）において、所定の実行条件が成立したので、ポインタｆｐの指示している関数を呼ぶ。その際、データへのポインタ変数ｐの値を引数として与えている。これにより、ポインタｆｐの指示している関数が呼ばれ、データ保持部１０３で保持され、０ｘ３７４６８ＡＢＢで特定されるデータを用いて、計算が行なわれる。なお、図９では、一つのデータしかポインタｆｐの指す関数に渡されていないが、二つ以上のデータを渡すことも可能である。
【００５８】
（実施形態１：計算装置を実現するプログラム）
以上において、計算装置１００の説明と並行して、計算装置１００を実現するためのプログラムである計算プログラムについて、プログラミング言語上での表現を用いて説明を行なったが、ここで、再度、本発明における計算プログラムについて説明する。
【００５９】
本発明における計算プログラムは、ポインタ保持ステップと、ポインタ管理ステップと、データ保持ステップと、実行命令ステップと、を計算機に実行させるためのプログラムである。
【００６０】
「ポインタ保持ステップ」とは、計算の実行のための実行部を指示するためのポインタを保持するステップである。例えば、プログラミング言語において、図４のように変数ｆｐが定義されている場合において、プログラムが起動したときに、ポインタに対応する変数ｆｐの領域を確保するステップである。また、ポインタが図７に例示されるように、リストセルによって保持されている場合には、そのリストセルのメモリ領域を確保するステップである。
【００６１】
「ポインタ管理ステップ」は、ポインタ保持ステップにより保持されたポインタを所定の変更条件に従って変更するステップである。例えば、所定の変更条件が成立した場合に、ポインタに対応する変数に代入を行なうステップである。
【００６２】
「データ保持ステップ」は、前記ポインタの変更によっても前記実行に利用されるデータをそのまま保持可能に保持するステップである。「前記ポインタの変更」とは、ポインタ管理ステップによるポインタの変更である。また、「前記実行」とは、ポインタ保持ステップにより保持されたポインタにより指示された実行部による計算の実行を意味する。「そのまま保持可能」とは、ポインタ管理ステップによりポインタが変更されても、データ保持ステップで保持されたデータは変更されないことを意味する。すなわち、ポインタ保持ステップで保持されるポインタが格納される領域と、データ保持ステップで保持されるデータが格納される領域とは、分離されている。
【００６３】

「実行命令ステップ」は、所定の実行条件に従ってデータ保持ステップで保持されたデータを利用して、ポインタ保持ステップにより保持されたポインタで指示される実行部により計算の実行をさせるためのステップである。

【００６４】

（実施形態１：計算装置、計算プログラムで用いられる方法）

また、本発明においては、計算装置、計算プログラムなどで用いられる計算方法も提供される。

【００６５】

その計算方法は、ポインタ保持ステップと、ポインタ管理ステップと、データ保持ステップと、実効命令ステップと、を含む。

【００６６】

「ポインタ保持ステップ」は、計算の実行のための実行部を指示するためのポインタを保持するステップである。例えば、ポインタ保持部１０１を実行するためのステップである。

【００６７】

「ポインタ管理ステップ」は、ポインタ保持ステップにより保持されたポインタを所定の変更条件に従って変更するステップである。例えば、ポインタ管理部１０２を実行するためのステップである。

【００６８】

「データ保持ステップ」は、前記ポインタの変更によっても前記実行に利用されるデータをそのまま保持可能に保持するステップである。例えば、データ保持部１０３を実行するステップである。ここに「前記ポインタの変更」とは、ポインタ管理ステップによるポインタの変更である。また、「前記実行」とは、ポインタ保持ステップにより保持されたポインタにより指示された実行部による計算の実行を意味する。「そのまま保持可能」とは、ポインタ管理ステップによりポインタが変更されても、データ保持ステップで保持されたデータは変更されないことを意味する。

【００６９】

「実行命令ステップ」は、所定の実行条件に従ってデータ保持ステップで保持されたデータを利用して、ポインタ保持ステップにより保持されたポインタで指示される実行部により計算の実行をさせるためのステップである。例えば、実行命令部１０４を実行するステップである。

【００７０】

（実施形態１：主な効果）

実施形態１において説明した計算装置、計算プログラム、計算方法によれば、所定の実行条件に従って実行される実行部が、所定の変更条件に従って変更することができるので、動的に実行部を変更することが可能となる。

【００７１】

（実施形態２（主に請求項２に関連する））

本発明の実施形態２においては、所定の条件に従って、ポインタの追加と削除を行なう計算装置、計算プログラム、計算方法が提供される。

【００７２】

（実施形態２：構成）

計算装置について説明を行なうと、本実施形態における計算装置は、実施形態１の計算装置１００のポインタ管理部１０２は、所定の条件に従ってポインタの追加と削除とを行なう機能を有する。

【００７３】

同様に、本実施形態における計算プログラムは、実施形態１の計算プログラムポインタ管理ステップにおいて、所定の条件に従ってポインタの追加と削除とを計算機に実行させるためのプログラムである。

【００７４】

（実施形態２：ポインタの追加の例）
図１０は、ポインタの追加の例を例示する。図１０の（ａ）は、ポインタ保持部１０１が保持するポインタが、図７に例示されるようにリスト構造で保持されている様子を示している。このようにポインタがリスト構造で保持されている場合には、（ｂ）に例示するように、新たなリストセル１００９を確保し、そのポインタ１０１０が実行部１００８を指示するようにして、リスト構造を指すポインタ１００１がリストセル１００９を指すようにし、リストセル１００９からリストセル１００２を指すようにする。
【００７５】
図１１は、図１０の（ａ）の状態から図１０の（ｂ）の状態を得るための操作をプログラミング言語によって表わした例である。
【００７６】
（ａ）において、新たに追加するリストセルを指す変数ｐｌを宣言している。（ｂ）において、新たに追加するリストセルのメモリ領域を確保するために、ｍａｌｌｏｃという関数を呼び、その結果を変数ｐｌに代入している。なお、ｓｉｚｅｏｆ ＊ｐｌは、リストセルがメモリ空間で占めるバイトを表わす。（ｃ）において、確保されたリストセルのポインタが関数ｇを指示するような代入を行ない、確保されたリストセルから現在のリスト構造の先頭のリストセルを指すようにしている。ｐｏｉｎｔｅｒｌｉｓｔｂａｓｅは、変数であり、リスト構造を指すポインタ１００１に対応する。ｐｏｉｎｔｅｒｌｉｓｔｂａｓｅに変数ｐｌの値を代入することにより、図１０の（ｂ）の状態が得られる。
【００７７】
なお、図１０の（ｂ）及び図１１では、新たに追加するリストセルがリスト構造の先頭になるようにしたが、新たに追加するリストセルをリスト構造の任意の場所に追加することは容易である。
【００７８】
（実施形態２：ポインタの削除の例）
また、ポインタがリスト構造に保持されている場合には、ポインタを削除することも可能である。
【００７９】
図１２は、ポインタを削除する操作をプログラミング言語により表わした例である。（ａ）において変数ｐｌを宣言している。（ｂ）において、ポインタ１００１に相当するｐｏｉｎｔｅｒｌｉｓｔｂａｓｅの値をｐｌに代入し、ｐｌがリスト構造の最初のリストセルを指すようにし、ｐｏｉｎｔｅｒｌｉｓｔｂａｓｅの値を、リスト構造の最初のリストセルの次のリストセルを指すようにしている。（ｃ）において、ｐｌが指しているリストセルの占めるメモリ領域を、ｆｒｅｅという関数を呼ぶことにより、破棄している。
【００８０】
（実施形態２：命令実行部の表現）
図１３は、このようなリスト構造を用いた場合の命令実行部のプログラミング言語による表現を例示している。例えば、リスト構造の最初から３番目のリストセルのポインタにより指示される実行部を実行する場合には、（ｃ）において、ｐｏｉｎｔｅｒｌｉｓｔｂａｓｅの値を変数ｐｌに代入し、ｐｌがリスト構造の最初のリストセルを指すようにする。（ｄ）において、ｐｌをリスト構造に沿って３回動かす。なお、ｆｏｒ（ｎ＝０；ｎ＜３；ｎ＋＋）は繰り返しの処理を表わし、ｐｌ＝ｐｌ－＞ｎｅｘｔを繰り返し実行することを表わす。ｎ＝０は、繰り返しの処理の前にｎに０を代入することを表わし、ｎ＜３は、繰り返しを行なうために成立する条件を表わし、ｎ＋＋は、繰り返しを一度行なうと、ｎに代入されている値を１だけ増加させることを意味する。（ｅ）において、変数ｐｌの指すリストセルに格納されているポインタにより指示されている関数を呼ぶ。その際、変数ｐに代入されている値を渡す。
【００８１】
（実施形態２：データ構造は、リスト構造に限られることはない）
なお、上記では、データ構造としてリスト構造を用いたが、リスト構造に限定されること

は無い。例えば、充分大きな配列を容易してポインタ保持部を実現してもよい。また、ハッシュ構造や木構造を用いてポインタ保持部を実現するようにしてもよい。

【００８２】

（実施形態２：主な効果）

ポインタ管理部によりポインタの追加が行なわれると、実効命令部により実行される実行部が増えることになるので、計算装置、計算プログラムにより提供される機能が増えることになる。また、ポインタ管理部によりポインタの削除が行なわれると、実行命令部により実行される実行部が減ることになるので、計算装置、計算プログラムにより、例えば不要な機能が提供されないようにすることができる。

【００８３】

（実施形態３（主に請求項３、４に関連する））

本発明の実施形態３においては、実行部の再読込がされる計算装置、計算プグラム、計算方法が提供される。

【００８４】

（実施形態３：構成）

図１６は、本実施形態における計算装置の機能ブロック図を例示する。図１６は、実施形態１の計算装置１００のポインタ管理部１００２が実行部再読込手段１６０１を有している。

【００８５】

（実施形態３：実行部再読込手段）

「実行部再読込手段」１６０１は、実行部の再読込を行なう。実行部の再読込は、計算装置１００の動作開始時であってもよい。また、実行部の再読込は、計算装置１００が動作している途中であってもよい。

【００８６】

計算装置１００が計算プログラムにより計算機により実現される場合には、実行部再読込手段１６０１に相当する機能は、例えばヒープ領域からメモリ領域を確保し、その確保されたメモリ領域に、関数や手続きを実現するインストラクションを含むデータを読み込むことにより実現される。

【００８７】

なお、「再読込」とは、全く同じ実行部を読み込むことに限定されることはない。例えば、バグなどの修正を行なったり、処理の高速化が図られたりすることにより、現在の実行部とは異なる実行部が読み込まれてもよい。

【００８８】

（実施形態３：ポインタ管理部）

本実施形態において、ポインタ管理部は、実行部再読込手段１６０１により再読込された実行部を指示するように前記ポインタを変更する。「前記ポインタ」とは、ポインタ保持部１０１で保持されるポインタである。なお、ポインタ保持部１０１で保持されるポインタの全てが変更されてもよい。あるは、ポインタ保持部１０１で保持される一部のポインタが変更されるようになっていてもよい。

【００８９】

計算装置１００が計算プログラムにより計算機により実現される場合には、このようなポインタ管理部の機能は、インストラクションを含むデータが読み込まれたメモリ領域の関数や手続きを実現するインストラクションの最初のアドレスをポインタに代入することにより実現される。このために、データに含まれる関数名とアドレスとの対応関係を示すシンボルテーブルが読み込まれたりする。

【００９０】

（実施形態３：指示されなくなった実行部は削除されてもよい）

なお、実行部再読込手段により実行部が読み込まれ、その読み込まれた実行部をポインタが指示すると、どのポインタからも指示されない実行部が生じる場合がある。そのような実行部は、削除されるようになっていてもよい。

【００９１】
計算装置１００が計算プログラムにより計算機により実現される場合には、どのポインタからも指示されない実行部のメモリ領域が回収され、次に実行部が読み込まれる場合に再利用されるようになっていてもよい。
【００９２】
（実施形態３：実行部再読込手段は、データ変換実行部を読み込んでもよい（主に請求項４に関連する））
なお、実行部再読込手段は、データ変換実行部を読み込むようにしてもよい。ここに、「データ変換実行部」とは、データ保持部１０１に保持されているデータを変換するための部である。
【００９３】
「データの変換」とは、データの形式を変更することである。例えば、今までハッシュによって表現されていたデータをＢ木（Ｂ－ｔｒｅｅ）によって表現されたデータに変更することである。単方向リストを用いて表現されていたデータを、双方向リストに変更することも例示することができる。また、Ｃ言語の構造体によってデータの形式が定義されている場合において、構造体のメンバの順序の入れ替え、削除や、追加を行なうことも含まれる。また、メンバの型の変換を行なうことなども含まれる（例えば、ｃｈａｎｒ型のメンバの値をｌｏｎｇ型に変換する。）。
【００９４】
構造体のメンバの追加としては、以下のものがある。すなわち、現在の構造体が次の式のように定義されているとする。
【００９５】
【数２】

```
struct oldstruct {
    int ver;
    struct sockaddr src;
    struct sockaddr dst;
};
```

この時、新しい構造体の定義が、次のように、ｔｉｍｅというメンバが追加されたものとする。
【００９６】

【数３】

```
struct newstruct {
    int ver;
    struct sockaddr src;
    struct sockaddr dst;
    time_t time;
};
```

この場合、実行部再読込手段が読み込むデータ変換実行部は、例えば、ｎｅｗｓｔｒｕｃｔのデータのメモリ領域を確保し、ｏｌｄｓｔｒｕｃｔのデータのメンバｖｅｒ、ｓｒｃ、ｄｓｔの値をそのまま持つように、ｎｅｗｓｔｒｕｃｔのｖｅｒ、ｓｒｃ、ｄｓｔに代入し（なお、ｖｅｒをデータ構造が変換された回数を示すメンバとする場合には、ｖｅｒの値を１増やすなどのことを行なう。）、メンバｔｉｍｅの値を適宜生成して持たせることを行なう。また、ｏｌｄｓｔｒｕｃｔのデータが存在していた場所にｎｅｗｓｔｒｕｃｔのデータを配置する。あるいは、ｏｌｄｓｔｒｕｃｔを指すポインタ変数の値を、ｎｅｗｓｔｒｕｃｔを指すように変更する。その後、ｏｌｄｓｔｒｕｃｔのメモリ領域を破棄などする。

【００９７】

なお、データ変換実行部によるデータの変換の目的としては、例えば、実行部再読込手段により再読込された実行部の扱えるデータ構造が、それまでの実行部の扱えるデータ構造と異なる場合に、それまでの実行部の扱えるデータ構造のデータを再読込された実行部が扱えるようにすることを挙げることができる。

【００９８】

したがって、このような場合においては、データ変換実行部は、実行部による計算の実行が行なわれる前に、データを変換する必要がある。そのために、例えば、データ変換実行部が読み込まれたときにデータ変換実行部によりデータの変換が行なわれるようにする。あるいは、ポインタ管理部１０２により、ポインタの変更が行なわれるときに、データ変換実行部によりデータの変換が行なわれるようにしたりする。あるいは、再読込された実行部による計算の開始を検出するようにしておき、その実行部による計算の開始が実行されると、データ変換実行部によりデータの変換が行なわれるようにしてもよい。

【００９９】

（実施形態３：データはバージョン情報を含み、実行部はデータに含まれるバージョン情報に応じた計算の実行を行なうようにしてもよい（主に請求項５に関連する））

なお、実行部の再読込に伴なう上記のようなデータの変換を不要にすることもできる。このためには、例えば式２あるいは式３で示したデータ構造の最初のメンバｖｅｒは、データのバージョンを示すようにする。すなわち、前記データは、バージョン情報を含んでいてもよい。なお、ここでいう「前記データ」とは、データ保持部１０１で保持されているデータを意味する。また、「バージョン情報」とは、データ構造を示す情報である。すなわち、バージョン情報により、データ構造が指定される。バージョン情報の例としては、整数値、固定小数点数値、浮動小数点数値、文字列などの等しいかどうかを判断することができるデータを用いることができる。また、Ｃ＋＋言語のようにクラスに等しいかどうかを判断するメソッドを持たせることができる場合には、バージョン情報を表わすクラスを用いてもよい。上記の式２、式３においては、ｖｅｒという整数型のメンバにバージョ

ン情報が保持されるので、整数値により、バージョン情報が表されている。
【０１００】
このようにデータがバージョン情報を含む場合、実行部再読込手段１６０１により再読込された実行部は、そのデータのバージョン情報に応じた計算の実行を行なうようにしてもよい。すなわち、実行部は、まずデータに含まれるバージョン情報を調べる。次に、バージョン情報に応じた計算を行なう。プログラミング言語上（特にＣ言語上）では、このような処理は、データを、バージョン情報を保持するメンバを含む特定の構造体にキャストを行ない、データに含まれるバージョン情報に相当するメンバの値を得る。次に、そのメンバの値をｉｆ文あるいはｓｗｉｔｃｈ文などにおいて参照し、適切な計算を選択して行なうようにする。
【０１０１】
また、もし、再読込された実行部がデータを生成した場合、そのデータ構造を示すバージョン情報をデータに含ませ、データ保持部１０１に保持させるようにしてもよい。
【０１０２】
このようにすることにより、実行部の再読込に伴なうデータの変換を行なう必要が無くなる。
【０１０３】
（実施形態３：主な効果）
本実施形態によれば、実行部が再読込されるので、計算装置の実行が行なわれている途中で、バグが修正された実行部や、高速化が図られた実行部が読み込まれるので、計算装置を停止させることなく、実行部の更新（バージョンアップ）が可能となる。
【０１０４】
（実施形態４（主に請求項６に関連する））
本発明の実施形態４においては、実行部が追加される計算装置、計算プログラム、計算方法が提供される。
【０１０５】
（実施形態４：構成）
図１７は、本実施形態における計算装置の機能ブロック図を例示する。本実施形態の計算装置は、実施形態２の計算装置のポインタ管理部が実行部追加手段１７０１を有している。
【０１０６】
（実施形態４：実行部追加手段）
「実行部追加手段」１７０１は、実行部を追加する。実行部の追加は、実施形態３における実行部の再読込のように、計算装置がプログラムにより計算機で実現される場合には、例えばヒープ領域からメモリ領域を確保し、その確保されたメモリ領域に、関数や手続きを実現するインストラクションを含むデータを読み込むことにより実現される。
【０１０７】
（実施形態４：ポインタ管理部）
本実施形態において、ポインタ管理部１０２は、実行部追加手段１７０１により追加された実行部を指示するポインタをポインタ保持部１０１に追加する。このポインタの追加は、例えば、図１１に示したプログラミング言語で表現された操作により実現することができる。
【０１０８】
（実施形態４：処理の流れ）
図１８は、本実施形態における計算装置１００の本実施形態に特有の処理の流れを説明するフローチャートである。ステップＳ１８０１において、実行部追加手段１７０１により実行部を追加する。ステップＳ１８０２において、ポインタ管理部１０２により追加する実行部を指示するポインタを、ポインタ保持部１０１に、追加する。
【０１０９】
（実施形態４：主な効果）

本実施形態によれば、実行部が追加されるので、計算装置、計算プログラムを止めることなく、新しい機能を持った実行部を追加することができ、その実行部による機能が提供されるようにすることができる。

【０１１０】

（実施形態５（主に請求項７に関連する））

本発明の実施形態５においては、実行部が削除される計算装置、計算プログラム、計算方法が提供される。

【０１１１】

（実施形態５：構成）

図１９は、本実施形態における計算装置の機能ブロック図を例示する。本実施形態の計算装置は、実施形態２の計算装置のポインタ管理部が実行部削除手段１９０１を有している。

【０１１２】

（実施形態５：実行部削除手段）

「実行部削除手段」１９０１は、実行部を削除する。実行部の削除とは、実行部が配置されているアドレス空間のメモリ領域を回収し、再利用可能とすることである。

【０１１３】

（実施形態５：ポインタ管理部）

本実施形態において、ポインタ管理部１０２は、実行部削除手段１９０１により削除された実行部を指示しているポインタの削除又は変更を行なう。例えば、削除される実行部のアドレスが、０ｘ２８４７２９ＥＣである場合には、図２１に示すようにリスト構造をたどり、０ｘ２８４７２９ＥＣを指示しているポインタを有するリストセルを探し出し、そのリストセルを削除する。あるいは、そのリストセルのポインタを別の実行部を指示するようにする。あるいは、ＮＵＬＬという特別な値にする。

【０１１４】

（実施形態５：処理の流れ）

図２０は、本実施形態における計算装置１００の本実施形態に特有の処理の流れを説明するフローチャートである。ステップＳ２００１において、削除する実行部を指示するポインタを探す。ステップＳ２００２において、ポインタを削除または変更する。ステップＳ２００３において、実行部を削除する。

【０１１５】

（実施形態５：主な効果）

本実施形態によれば、実行部が削除されるので、計算装置、計算プログラムを止めることなく、機能を削除することができる。

【０１１６】

（実施形態１～５：応用例：ルータ）

以下に、実施形態１から５の計算装置、計算プログラム及び計算方法の応用例を示す。その一つの例として、ルータがある。

【０１１７】

ルータは、通常は複数のネットワークインターフェースを有している。あるネットワークインターフェースでパケットを受信すると、データとして保持している経路情報に従って、パケットを送信するネットワークインターフェースを選択して、パケットを送信する。

【０１１８】

また、ルータは他のルータから送られてきた経路情報を受信し、データとして保持している経路情報の再計算を行なう。また、他のルータに対して、自己がデータとして保持している経路情報を他のルータへ送信することも行なう。さらに、ある長さの時間の経過を検出するためのタイマーにより、経路の再計算を行なうべきことや隣接するルータがダウンしていることを検出したり、経路情報以外の制御情報の受信あるいは送信を行なうべきことを検出したりする。

【０１１９】

ルータは、このようなパケットを受信したこと、タイマーによりある長さの時間が経過したことなどの事象の発生を、「イベント」として検出するようになっている。イベントが検出されると、イベントに対応付けられた関数が呼ばれる。この関数のことを「イベントハンドラ」と呼ぶ。

【０１２０】

また、ルータの設定等を行なうための指示はルータに接続されたキーボードなどから行なわれ、これらもイベントハンドラにより処理がされる。

【０１２１】

（実施形態１～５：応用例：ルータ：ポインタ保持部）

従って、イベントハンドラを実行部とし、それを指示することをポインタによって行ない、これらのポインタを格納することによりポインタ保持部が得られる。

【０１２２】

（実施形態１～５：応用例：ルータ：ポインタ管理部、データ保持部、実行命令部）

また、このポインタを所定の条件に従って変更する部としてポインタ管理部が得られ、保持している経路情報などを保持する部として、データ保持部が得られ、イベントを取得した場合を実行条件として、そのイベントに対応するイベントハンドラを呼ぶ部を実行命令部とすることができる。

【０１２３】

（実施形態１～５：応用例：ルータ：フローチャート）

図１４は、ルータの動作を説明するフローチャートを例示する。

【０１２４】

ステップＳ１４０１において、イベントの発生を待つ。例えば、ｓｅｌｅｃｔシステムコールを実行し、パケットが受信されたこと、パケットの送信が可能になったこと、一定時間が経過したこと、などが発生するまで待つ。

【０１２５】

ステップＳ１４０２において、イベントの種類を決定する。例えば、ｓｅｌｅｃｔシステムコールの結果を調べて、パケットが受信されたかどうか、送信可能になったかどうか、一定時間が経過したかどうか、などを決定する。

【０１２６】

ステップＳ１４０３において、イベントの種類に基づいて、データを取得する。例えば、パケットが受信された場合には、経路情報を取得する。

【０１２７】

ステップＳ１４０４において、イベントの種類に基づいて実行部を指示するポインタを取得する。

【０１２８】

ステップＳ１４０５において、データを利用してポインタで指示される実行部により計算の実行を行なう。例えば、制御パケットが受信された場合には、制御パケットの受信というイベントに対応付けられたイベントハンドラを呼ぶ。

【０１２９】

ステップＳ１４０６において、ポインタの変更を行なうべきかどうかを判断する。ここにいう、ポインタの変更とは、実施形態２などのポインタの追加、削除を含んでいてもよい。もし、ポインタの変更をすると判断された場合には、ステップＳ１４０７に処理が移行し、ポインタを変更することを行なう。ポインタの変更をしないと判断された場合には、ステップＳ１４０７をスキップする。

【０１３０】

ステップＳ１４０８において、ルータの処理を終了するかどうかを判断し、もし、終了しないならば、ステップＳ１４０１へ戻る。

【０１３１】

（実施形態１～５：応用例：ルータ：イベントとイベントハンドラの対応付け）

なお、ステップＳ１４０３とＳ１４０４とにおいて、イベントに対応付けられたデータ、

ポインタを取得する必要があるが、これは、図１５（ａ）に示すデータ構造１５０１を用意することにより、取得することができる。データ構造１５０１は、イベントタイプ１５０２、ポインタ１５０３、データ１５０４からなっており、イベントタイプ１５０２は、イベントの種類を示し、ポインタ１５０３は、イベントに対応付けられたイベントハンドラを指示するポインタであり、データ１５０４は、イベントハンドラに渡すべきデータである。例えば、このようなデータ構造１５０１を、イベントタイプをキーとするハッシュ構造に格納しておくことにより、イベントタイプから、対応するデータ構造を高速に得ることができる。

【０１３２】

（実施形態１～５：応用例：ルータ：イベントハンドラにバグがあってもルータを停止することなく修正することができる、イベントハンドラの機能追加もルータを停止することなく行なうことができる）

例えば、あるイベントハンドラにバグがあると判明した場合には、バグの無いイベントハンドラをルータに再読込させ、バグがあるイベントハンドラを指示していたポインタを変更し、再読込をしたイベントハンドラを指示するようにする。これにより、ルータを停止させることなく、イベントハンドラをバグが除去されたものに更新することが可能となる。また、イベントハンドラに機能が追加された場合に、その機能が追加されたイベントハンドラを再読込させることにより、ルータを停止させることなく、機能の追加が可能となる。ルータを停止させる必要が無いので、変数の値、データベースの中身、オープンしているファイル、通信あるいはソケットの状態がそのまま保持されたまま、イベントハンドラの更新、追加が可能となる。

【０１３３】

イベントハンドラが再読込された場合、そのイベントハンドラと、再読込される前のイベントハンドラと、が扱うデータの形式が異なる場合には、イベントハンドラとともに、データを変換するためのデータ変換実行部が読み込まれてデータの変換が行なわれるようになっていてもよい。

【０１３４】

また、この場合、イベントハンドラが再読込させたことを再読込イベントというイベントとして定義しておき、再読込イベントに対応付けられたイベントハンドラとして、データ保持部に保持されたデータの構造を更新する関数を用いるようにしてもよい。これにより、イベントハンドラの更新に伴うデータの構造の更新を行なうことが可能となる。例えば、データ構造がＣ言語の構造体によって定義されているとし、イベントハンドラが扱うデータを表わす構造体のメンバの順序が入れ替わったり、構造体メンバが削除されたり追加されたとする。この場合には、古い定義の構造体のデータから、新しい構造体のデータへ変換することを、再読込イベントに対応付けられたイベントハンドラにより行なうようにしてもよい。

【０１３５】

また、新たな種類のイベントに対する処理が必要になった場合には、そのイベントに対応付けられるべきイベントハンドラを読み込ませることにより、ルータを停止させることなく、イベントハンドラを追加することができる。

【０１３６】

また、反対に、イベントハンドラが不要になれば、イベントハンドラを削除することも可能となる。

【０１３７】

（実施形態１～５：他の応用例）

上記では、ルータを本発明の応用例として取り上げたが、本発明の応用範囲は、ルータにのみ限定されるものではなく、イベントを取得し、それに応じてイベントハンドラを呼ぶ計算装置、計算プログラム、計算方法に応用することができる。

【０１３８】

例えば、グラフィカルユーザインターフェースを持つプログラムにおいては、マウスの移

動、マウスボタンの押下、キーボードのキーの押下、今まで他のウィドウにより隠されて見えなかったウィンドウの一部が見えるようになったこと、などがイベントとして定義され、それぞれのイベントを処理するためのイベントハンドラが提供されている。したがって、イベントハンドラを実行部とし、それをポインタにより指示することにより、プログラムを停止することなく、イベントハンドラの再読込、追加、削除が可能となり、バージョンアップを行なうことが可能である。

【０１３９】

また、ウェブサーバなどにおいても、クライアントからの要求をイベントとして定義し、そのイベントにイベントハンドラを対応付けることができるので、本発明により、ウェブサーバを停止させることなく、イベントハンドラの再読込、追加、削除が可能となり、バージョンアップが可能となる。

【０１４０】

また、ウェブサーバの背後でデータベース管理システムが動作している場合が近年多くなってきた。データベース管理システムも、データの検索要求、データの削除要求、データの挿入要求などをイベントとして定義しておき、そのイベントに対応したイベントハンドラを対応付けることができるので、本発明により、データベース管理システムを停止させることなく、バージョンアップを行なうことができる。

【０１４１】

（実施形態１～５：他の応用例：実行部はイベントハンドラに限られることはない）

なお、上記の説明においては、実行部とイベントハンドラを同一視して説明を行なったが、実行部がイベントハンドラに限られることはない。例えば、Ｃ言語でプログラムが記述される場合、特定の機能を提供する関数を実行部として対応付けておき、そのように実行部に対応付けられた関数を直接呼ぶのではなく、関数へのポインタを用いて呼ぶようにしてもよい。もし、そのような実行部に対応付けられた関数にバグが発見されたり、関数の機能を拡張したりする場合には、バグを除去した関数や機能が拡張された関数を実行部として再読込させ、再読込された実行部に対応する関数を指示するように関数へのポインタを書き換える。

【０１４２】

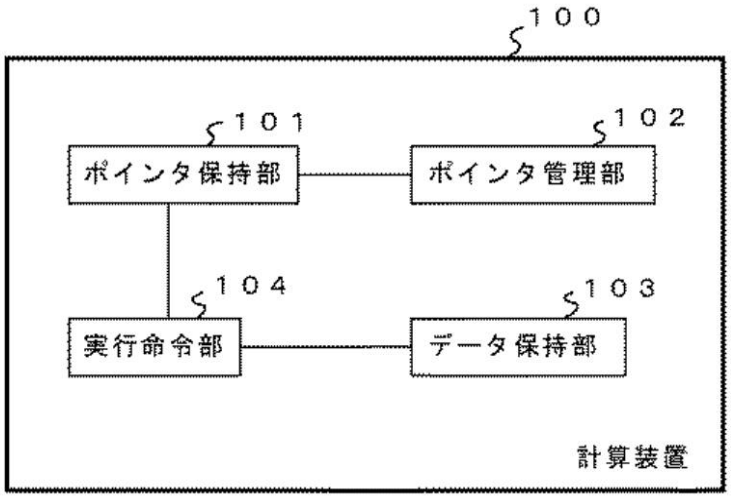
このように、特定の機能を提供する関数を実行部とし、そのような実行部をポインタにより指示することにより、実行部をイベントと対応づけることなく、プログラムを停止させることなく、実行部の交換、追加などが可能となる。

【符号の説明】

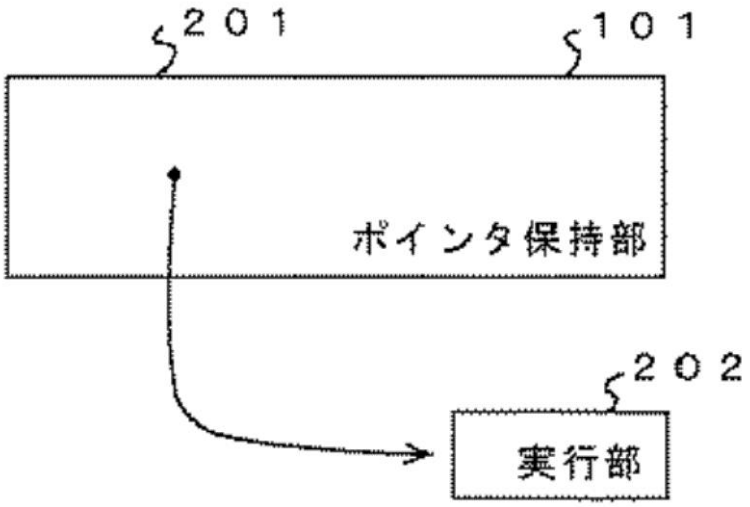
【０１４３】

１００ 計算装置
１０１ ポインタ保持部
１０２ ポインタ管理部
１０３ データ保持部
１０４ 実行命令部

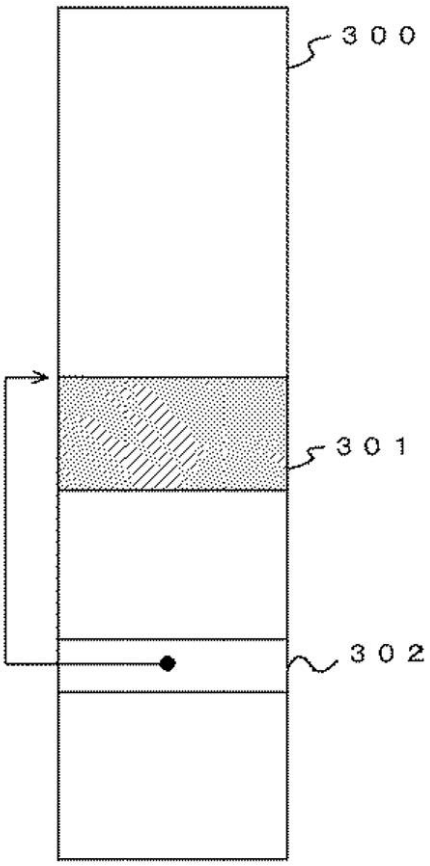
【図１】


100
101
ポインタ保持部
102
ポインタ管理部
104
実行命令部
103
データ保持部
計算装置

【図２】


201
101
ポインタ保持部
202
実行部

【図３】

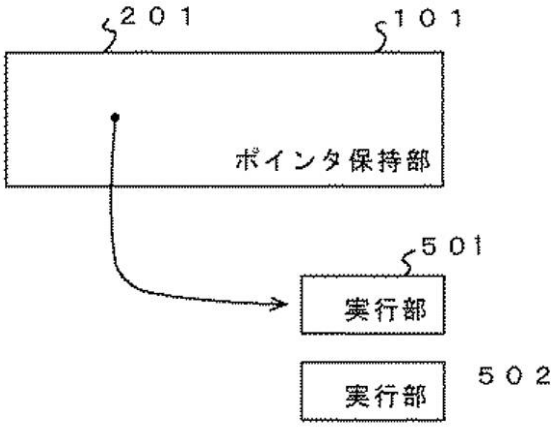


【図４】

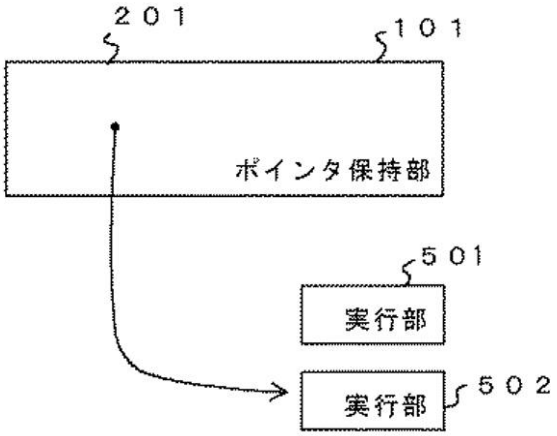
```
void  (* fp)();              ・・・    (a)



void                         ・・・    (b)
f(dp)
 void *dp;
 {
     :
     :

 }


main()
{                            ・・・    (C)
     :
     :
   fp = f;                   ・・・    (d)
     :
     :

}
```

【図５】

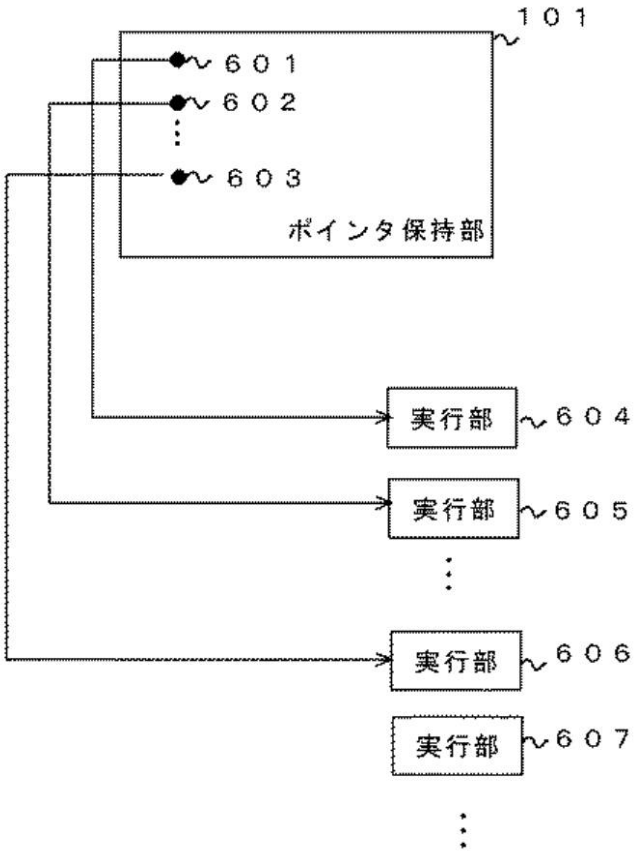
(a)


２０１
１０１
ポインタ保持部
５０１
実行部
実行部
５０２

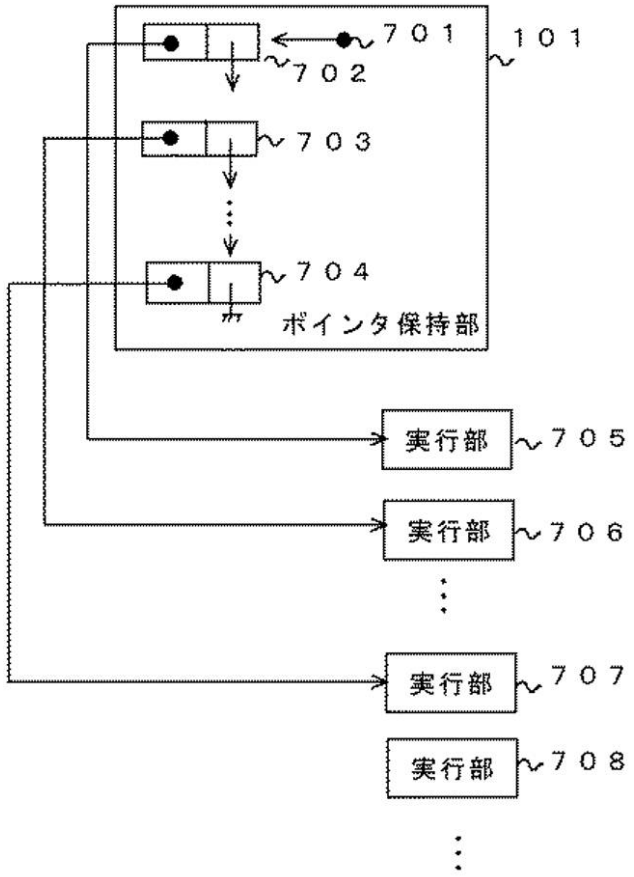
(b)


２０１
１０１
ポインタ保持部
５０１
実行部
実行部
５０２

【図６】



【図７】



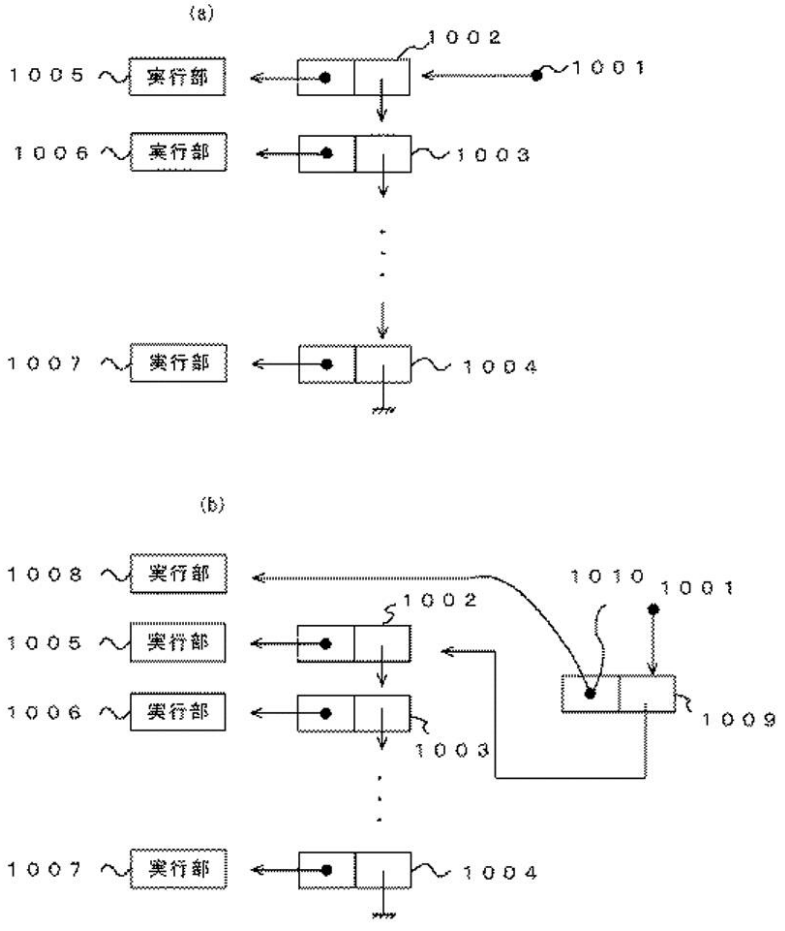
【図８】

```
struct  pointerlist  {
        void  (*fp)();                  ・・・        (a)
        struct  pointerlist   *next;    ・・・        (b)
}  *pointerlistbase;
```

【図９】

```
void  *p ;                          ・・・      (a)
      ・
      ・
      ・
P = (void*) 0x37468AB8;             ・・・      (b)
      ・
      ・
      ・
(*fp)(p);                           ・・・      (c)
```

【図１０】



【図１１】

```
struct   pointerlist   *pl;          ・・・  (a)


pl = malloc (sizeof  *pl);          ・・・  (b)

pl —>  fp = g;
pl —>  next = pointerlistbase;                  (c)
pointerlistbase = pl ;
```

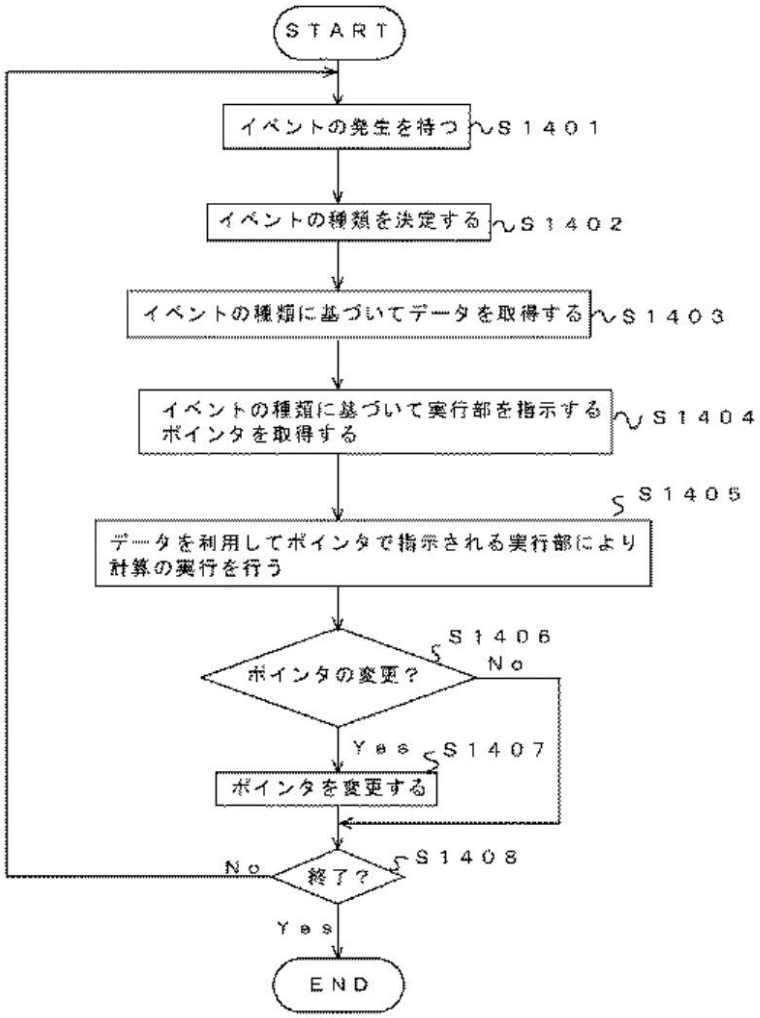
【図１２】

```
struct  pointerlist  *pl;                ・・・      (a)

pl  =  pointerlistbase;
pointerlistbase  =  pointerlistbase  —>  next;     (b)

free (pl);                               ・・・      (c)
```

【図１３】

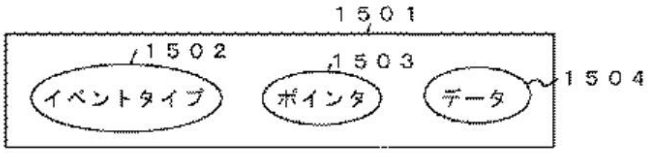
```
struct  pointerlist  *pl;
int                   n;          ・・・(a)
void                 *p;

P = (void*) 0x37468AB8;             ・・・  (b)
pl = pointerlistbase;               ・・・  (c)
for ( n = 0; n < 3; n++)
    pl = pl —> next;                ・・(d)

( *pl —> fp)(p);                    ・・・  (e)
```

【図１４】



【図１５】

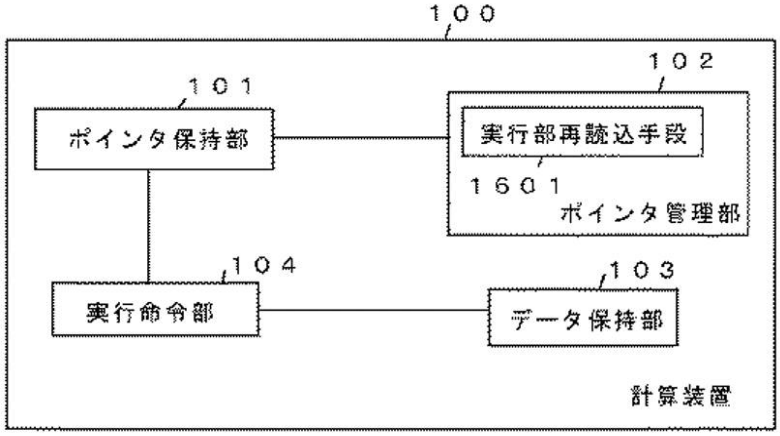
(a)



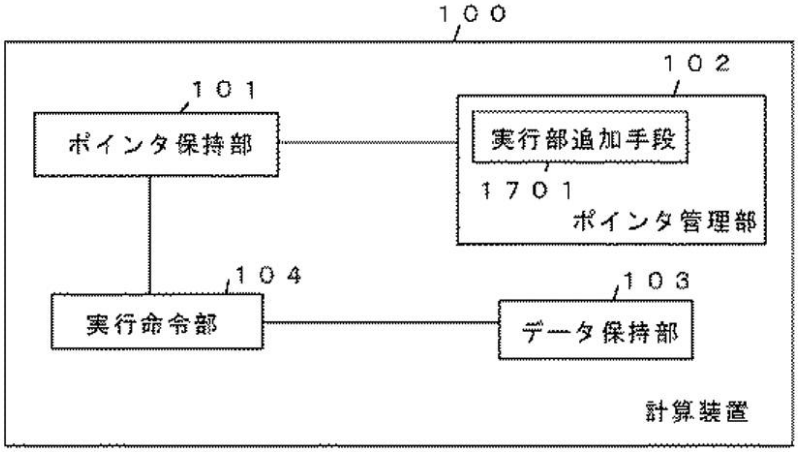
(b)

```
struct {
   enum eventType    et;          ・・・    (イ)
   void          (*fp)();         ・・・    (ロ)
   void             *dp;          ・・・    (ハ)
};
```

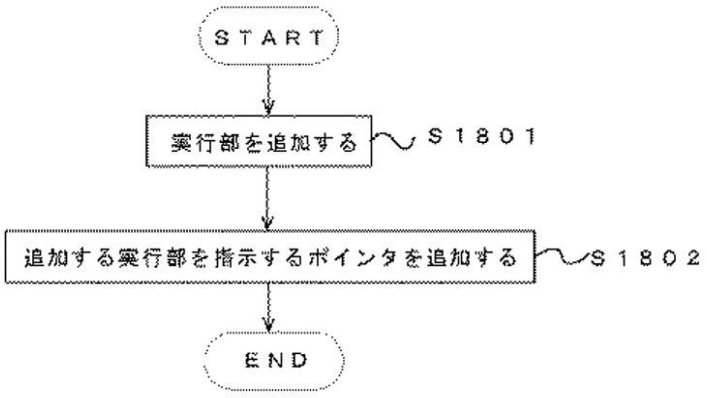
【図１６】


１００
１０１
ポインタ保持部
１０２
実行部再読込手段
１６０１
ポインタ管理部
１０４
実行命令部
１０３
データ保持部
計算装置

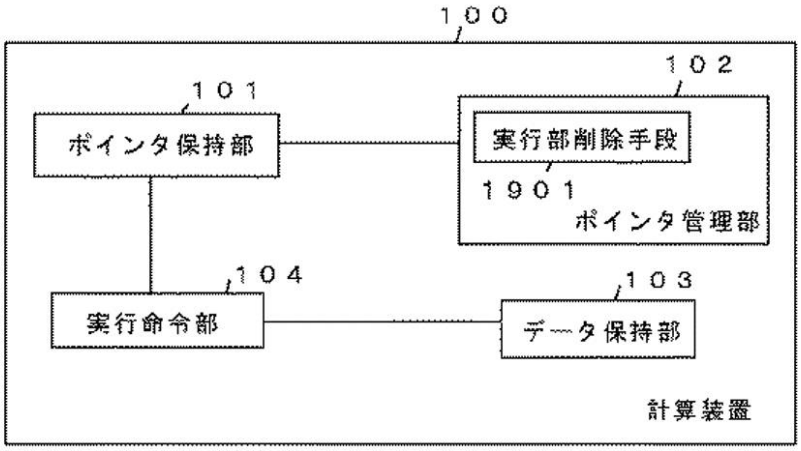
【図１７】


１００
１０１
ポインタ保持部
１０２
実行部追加手段
１７０１
ポインタ管理部
１０４
実行命令部
１０３
データ保持部
計算装置

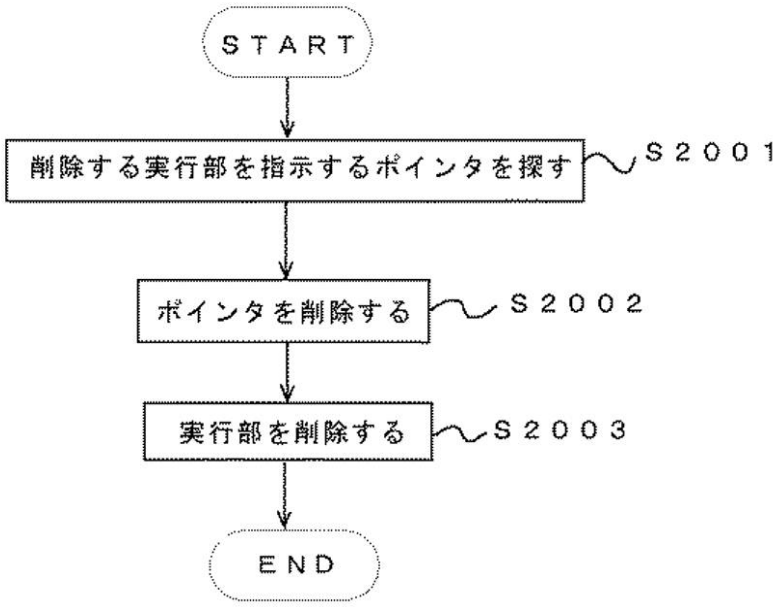
【図１８】


ＳＴＡＲＴ
実行部を追加する
Ｓ１８０１
追加する実行部を指示するポインタを追加する
Ｓ１８０２
ＥＮＤ

【図１９】



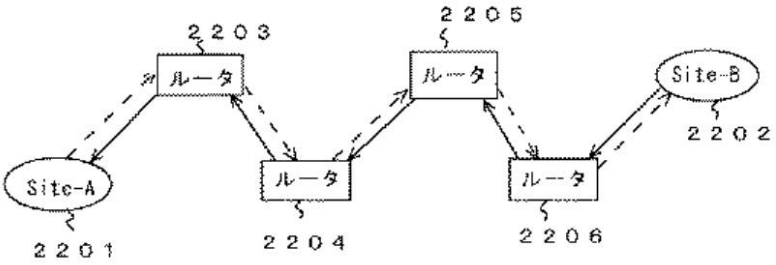
【図２０】



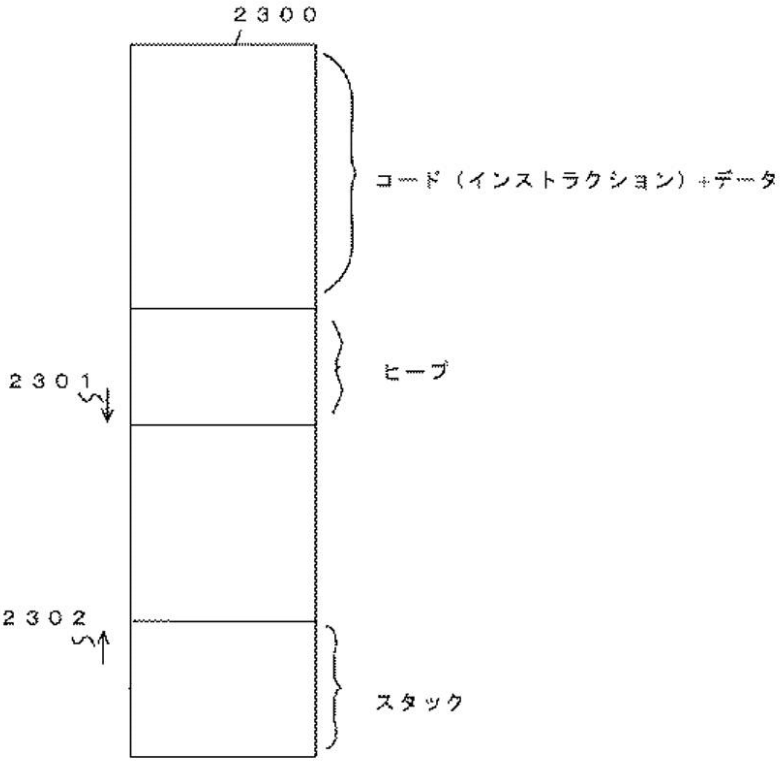
【図２１】

```
struct  pointerlist   *pl;
void   (*ep)();                          ・・・  (a)

ep  =  (void (*)()) 0x284729EC;          ・・・  (b)
pl  =  pointerlistbase;                  ・・・  (c)
while ( pl ! = NULL ) {
   if ( pl —> fp == ep)                  ・・・  (d)
      break;
   pl  =  pl —> next;                    ・・・  (e)
}
```

【図２２】


２２０３
２２０５
ルータ
ルータ
Site-B
２２０２
Site-A
ルータ
ルータ
２２０１
２２０４
２２０６

【図２３】


２３００
コード（インストラクション）＋データ
２３０１
ヒープ
２３０２
スタック