República Federativa do Brasil
Ministério da Indústria, Comércio Exterior
e Serviços
Instituto Nacional da Propriedade Industrial

(11) **PI 0007976-6 B1**

(22) Data do Depósito: 02/02/2000

(45) Data de Concessão: 11/09/2018

(54) Título: PROCESSO IMPLEMENTADO POR COMPUTADOR PARA GERENCIAR EXCEÇÕES LANÇÁVEIS DURANTE EXECUÇÃO DE MÉTODOS EM UMA OU MAIS CLASSES POR UMA MÁQUINA, MÉTODO QUE MINIMIZA A QUANTIDADE DE ARMAZENAMENTO REQUERIDA PARA UMA PILHA EM TEMPO DE EXECUÇÃO QUANDO EXECUTANDO UM PROGRAMA, E MÉTODO PARA CONVERTER ARQUIVOS DE CLASSE EM UM APPLET CONVERTIDO PARA EXECUÇÃO NUM DISPOSITIVO COM RESTRIÇÃO A RECURSO

(51) Int.Cl.: G06F 9/00

(30) Prioridade Unionista: 02/02/1999 US 09/243107

(73) Titular(es): ORACLE AMERICA , INC.

(72) Inventor(es): JUDITH E. SCHWABE; JOSHUA B. SUSSER

(85) Data do Início da Fase Nacional: 02/08/2001

“PROCESSO IMPLEMENTADO POR COMPUTADOR PARA GERENCIAR EXCEÇÕES LANÇÁVEIS DURANTE EXECUÇÃO DE MÉTODOS EM UMA OU MAIS CLASSES POR UMA MÁQUINA, MÉTODO QUE MINIMIZA A QUANTIDADE DE ARMAZENAMENTO REQUERIDA PARA UMA PILHA EM TEMPO DE EXECUÇÃO QUANDO EXECUTANDO UM PROGRAMA, E MÉTODO PARA CONVERTER ARQUIVOS DE CLASSE EM UM APPLET CONVERTIDO PARA EXECUÇÃO NUM DISPOSITIVO COM RESTRIÇÃO A RECURSO”

FUNDAMENTOS

[001] A presente invenção, geralmente, relaciona-se a software de computador orientado para objetivo, e mais especificamente, a estruturas de dados e métodos de funcionamento/operação, para minimizar requisitos de armazenamento de pilha, enquanto suportando operação de exceção em uma máquina virtual Java® implementada em dispositivos com restrição a recurso tais como cartões inteligentes e similares.

[002] Uma máquina virtual é uma máquina de computação abstrata gerada por uma aplicação de software ou por seqüência de instruções que é executada por um processador. Programas executados numa máquina virtual podem ser de arquitetura neutra. O termo "arquitetura neutra" se refere aos programas, como aqueles escritos na linguagem de programação Java® que podem ser executados por uma máquina virtual ou ainda numa variedade de plataformas de computador que possuem uma variedade de arquiteturas de computador diferentes. Consequentemente, por exemplo, uma máquina virtual implementada com um sistema de computador pessoal baseado em Windows™ usará o mesmo conjunto de instruções que uma máquina virtual implementada num sistema de computador baseado em UNIX™. O resultado da codificação independente de plataforma de uma seqüência de instruções de uma máquina virtual em um fluxo de um ou mais códigos de bytes cada um dos quais é, por exemplo, um código numérico de um byte de comprimento.

[003] Aquela linguagem de programação Java é uma linguagem de programação orientada para objetivo. Em um sistema orientado para objetivo, uma "classe" descreve uma coleção de dados e métodos que operam naqueles dados. Tomados juntos, os dados e métodos descrevem o estado, bem como o comportamento de um objetivo.

[004] A linguagem de programação Java também é verificável, que significa que, antes da execução de uma aplicação escrita na linguagem de programação Java, uma determinação pode ser feita se qualquer seqüência de instruções no programa tentar processar os dados de um tipo impróprio para aquele código de bytes, ou se execução de instruções de código de byte no programa causará estouro negativo ou estouro de uma pilha de operando.

[005] Uma máquina virtual Java executa código de máquina virtual escrito na linguagem de programação Java "A Especificação de Máquina Virtual Java®", citada abaixo. Uma máquina virtual Java é projetada para uso com uma arquitetura de 32 bits. Porém, vários dispositivos com restrições a recurso, tais como cartões inteligentes, têm uma arquitetura de 8 ou 16 bits.

[006] Aqueles cartões inteligentes, também conhecidos como cartões portadores de dados portáteis inteligentes, geralmente são feitos de plástico ou metal e apresentam um circuito eletrônico que inclui um microprocessador embutido para executar programas e memória para armazenar programas e dados. Tais dispositivos, que podem ser aproximadamente do tamanho de um cartão de crédito, tipicamente possuem a capacidade de memória limitada. A arquitetura/memória limitada tornam não prático ou impossível implementar uma máquina virtual Java no referido dispositivo. Por exemplo, certos cartões inteligentes têm menos que 1K de memória de acesso aleatório (RAM) e 16K de memória só de leitura (ROM). Um exemplo de uma dificuldade quando implementando uma máquina virtual Java em um dispositivo com restrição a recurso surge na operação de exceções.

[007] Com referência agora à Figura 1, no contexto de programas de

computador escritos na linguagem de programação Java, um operador de exceção 100 é um procedimento (ou um conjunto de instruções dentro de um procedimento) que protege um conjunto especificado de código de programa, chamado de um bloco de código protegido 102. Quando um programa Java viola as sujeições semânticas da linguagem de programação Java, a máquina virtual Java sinaliza este erro ao programa como uma exceção. O operador de exceção é executado sempre que a exceção aplicável é "lançada" durante uma execução do código protegido correspondente. A especificação de linguagem de programação Java declara que uma exceção será lançada quando as sujeições semânticas são violadas e causará uma transferência não local de controle do ponto onde a exceção aconteceu para um ponto que pode ser especificado pelo programador. Uma exceção é dita ser lançada do ponto onde aconteceu e é dita ser capturada no ponto ao qual controle é transferido. Por exemplo, um operador de exceção particular, tal como um procedimento para operar erros de I/O de "fim de arquivo", pode ser definido para ser aplicável a uma parte particular de um primeiro método 104. Se a exceção correspondente (isto é, uma exceção de fim de arquivo neste exemplo) resulta durante execução do código protegido 102, execução do operador de exceção 100 é iniciada. Exceções podem ser lançadas implícita ou explicitamente. Exceções implícitas são lançadas por uma máquina virtual Java como um resultado da execução de uma instrução de programa, tal como uma exceção de ponteiro de nulo. Alternativamente, uma exceção explícita é incluída dentro do corpo de um método que usa uma declaração de "lançamento" Java.

[008] Uma exceção lançada é dita ser capturada por um operador de exceção se há um operador de exceção contido aplicável para a exceção lançada. Um operador de exceção contido é um no qual o escopo ou faixa de instruções aplicáveis inclui instrução que lança uma exceção correspondente. Da perspectiva de uma instrução particular em um método, o conjunto de operadores de exceção contidos é o conjunto de operadores de exceção, cuja

faixa de instruções aplicáveis (conjunto de código protegido) inclui a instrução particular.

[009] A linguagem de programação Java freqüentemente se refere a "declarações de tentativa", "blocos de tentativa", "cláusulas de captura" e "cláusulas finais" quando fazendo referência a operadores de exceção. Uma declaração de tentativa inclui um bloco de tentativa, zero ou mais cláusulas de captura e uma cláusula final opcional. Exceções são capturadas encerrando código em blocos de tentativa. Um bloco de tentativa é uma parte de código à qual um operador de exceção particular se aplica (isto é, o bloco de código protegido). Uma cláusula de captura define um operador de exceção. Uma cláusula final de uma declaração de tentativa provê um mecanismo para executar uma seção de código se ou não uma exceção é lançada. Em um programa Java, uma declaração ou expressão é dinamicamente incluída por uma cláusula de captura se ela aparece no interior do bloco de tentativa da declaração de tentativa de qual a cláusula de captura é uma parte ou se a parte que chama da declaração ou expressão é dinamicamente incluída pela cláusula de captura.

[0010] Na situação de uma cláusula de captura particular operar uma exceção determina-se se comparando a classe do objetivo de exceção que foi lançada ao tipo declarado do parâmetro da cláusula de captura. A cláusula de captura opera a exceção se o tipo de seu parâmetro é a classe da exceção ou uma superclasse da classe da exceção. De maneira equivalente, uma cláusula de captura capturará qualquer objetivo de exceção que é um caso do tipo de parâmetro declarado.

[0011] Se a parte protegida do primeiro método inclui as chamadas (chamadas de instruções de "invocação" na Especificação de Máquina Virtual Java, referenciada aqui abaixo) a outros métodos 106, que por sua vez podem incluir chamadas aninhadas a um conjunto de métodos adicionais 108, 110, então erros de fim de arquivo gerados por quaisquer dos métodos 106, 108,

110 chamados direta ou indiretamente pelo código protegido 102 também fará com que a execução de operador de exceção 100 seja invocada. Porém, o método aninhado 112 pode incluir seu próprio operador de exceção de fim de arquivo 114. Se uma exceção é lançada quando executando o método 112, então operador de exceção 114 será usado para operar exceções de fim de arquivo causadas por execução de instruções incluídas no método aninhado particular, como também exceções de fim de arquivo causadas por execução de quaisquer métodos 116 chamados por método aninhado 112.

[0012] Em programas convencionais de Java, todos aqueles métodos associados a uma classe de objetivo são armazenados juntos numa estrutura de dados chamada de um arquivo de classe o qual é definido na Especificação de Máquina Virtual Java. Cada método tem a sua própria tabela de exceção e, além disso, o código de cada método inclui o código para os operadores de exceção referenciados pela sua tabela de exceção. Quando um arquivo classe Java é criado, todas as exceções associadas com um método são arranjadas em uma lista, referenciada como uma tabela de operador de exceção. Com referência agora à Figura 2, uma tabela de operador de exceção convencional 200 inclui uma ou mais cláusulas de captura (operadores de exceção) 202. Cada cláusula de captura 202 inclui um endereço de pc de começo 204 e um endereço de pc de parada 206 que descreve a faixa de instrução de máquina virtual Java para a qual o operador de exceção encontra-se ativo, um indicador de tipo 208 que descreve os tipos de exceções em que a cláusula de captura é para operar, e um endereço 210 no qual uma execução do código de operador de exceção é para começar.

[0013] A ordem das cláusulas de captura na tabela de operador de exceção é importante. O lançamento de uma exceção resulta em uma pesquisa pela máquina virtual Java pela tabela de operador de exceção. A execução de máquina virtual Java continua numa primeira cláusula de captura de casamento. Porque código Java é estruturado, sempre é possível arranjar todos os

operadores de exceção para um método em uma única lista. Para qualquer possível valor de contador de programa, esta lista pode ser pesquisada para achar o operador de exceção correto, quer dizer, o operador de exceção mais interno que inclui o valor de contador de programa (pc) (onde a exceção foi lançada) e pode operar a exceção sendo lançada.

[0014] Se não há nenhuma cláusula de captura de casamento, o método atual é dito ter uma exceção não capturada. Quando uma exceção é não capturada, o estado de execução do invocador, o método que invocou o método atual (se qualquer), é restabelecido. A propagação da exceção continua pela exceção tinha acontecido no invocador na instrução que invocou o método realmente evocando a exceção.

[0015] Várias estruturas de dados em tempo de execução são mantidas por uma máquina virtual Java para manter rastro da execução e invocação de métodos. Uma máquina virtual Java pode suportar muitas linhas de execução de uma vez. Cada linha de máquina virtual Java tem seu próprio registrador de pc (contador de programa). O registrador de pc contém o endereço da instrução de máquina virtual Java atualmente sendo executada. Cada linha de máquina virtual Java tem uma pilha de Java privada, criada ao mesmo tempo que a linha. A pilha Java armazena quadros de máquina virtual. A pilha Java é equivalente à pilha de uma linguagem convencional tal como C: ela retém variáveis locais e resultados parciais, e suporta invocação de método e retorno.

[0016] Com referência agora à Figura 3, uma pilha Java convencional 300 incluindo uma pluralidade de quadros de máquina virtual de Java 302 é mostrada. Cada quadro 302 inclui uma referência 304 para um método e um ponteiro de retorno 306.

[0017] A referida referência 304 é uma referência ao método que está executando atualmente, referenciado como o método atual. Referência 304 é usada para indicar qual tabela de operador de exceção procurar quando uma

exceção é lançada durante execução do método atual. A referência 304 pode ser da forma de um endereço ao qual o método atual é armazenado. O código para cada método inclui tipicamente o espaço de pilha máximo precisado pelo método, o número máximo de registradores usados pelo método, os códigos de byte atuais para executar o método e uma tabela de operadores de exceção. A referência 304 pode apontar par o começo do método atual onde a tabela de operador de exceção está armazenada tipicamente.

[0018] Ponteiro de retorno 306 é um ponteiro para o método que invocou o método atual, e mais especificamente à localização onde execução é para retomar no método de invocação na conclusão de execução do método atual.

[0019] Como descrito acima, cartões inteligentes e outros dispositivos com restrições a recurso tipicamente incluem capacidade de memória limitada. Por conseguinte, programas incluindo uma pluralidade de métodos aninhados que exigem vários bytes a serem armazenados na pilha Java pode causar uma condição de sobrecarga de pilha no momento de funcionamento. É desejável limitar a informação exigida a ser armazenada em memória no momento de operação enquanto permitindo suporte total para operação de exceção convencional em uma máquina virtual Java.

SUMÁRIO DA INVENÇÃO

[0020] Em um aspecto, a invenção provê um processo implementado por computador para gerenciar exceções lançáveis durante execução de métodos em uma ou mais classes por uma máquina. Cada método inclui um arranjo de operador de exceção que define operadores de exceção associados com o método. O método inclui combinar os arranjos de operador de exceção para todos os métodos em uma única tabela de operador de exceção.

[0021] Aspectos da invenção incluem uma ou mais das características seguintes. Todos os arranjos de operador de exceção para todos os métodos em uma classe ou todos os métodos em todas as classes podem ser

armazenados na única tabela de operador de exceção. Todos os arranjos de operador de exceção para todos os métodos em um pacote Java podem ser combinados na única tabela de operador de exceção. Um método pode ser incluído em um arquivo de classe. A etapa de combinar todos os arranjos de operador de exceção pode incluir combinar os operadores de exceção de todos os métodos em um arquivo de classe na única tabela de operador de exceção.

[0022] O processo pode incluir pesquisar a tabela de operador de exceção quando uma exceção é lançada enquanto executando um dos métodos, incluindo localizar uma primeira exceção de casamento na única tabela de operador de exceção. A etapa de pesquisa pode incluir recobrar em ordem entradas de operador de exceção da tabela de operador de exceção, verificar o tipo e alcance de cada operador de exceção para o primeiro operador de exceção de casamento e parar pesquisa se um operador de exceção atual não casa e é o último operador para o nível mais alto de código protegido em um método associado.

[0023] Todos os métodos em uma classe são descritos em um único arquivo de classe. Os arquivos de classe podem ser arquivos de classe Java. Os métodos em uma ou mais classes podem ser agrupados em um pacote, onde o pacote inclui uma estrutura de dados de pacote que inclui primeira e segunda partes. O processo pode incluir armazenar a tabela de operador de exceção na primeira parte do pacote e todos os métodos na segunda parte do pacote. A etapa de combinar pode incluir concatenar os arranjos de operador de exceção incluindo carregar cada arranjo de operador de exceção naquela primeira parte da estrutura de dados de pacote, conforme uma ordenação predefinida. A ordenação predefinida pode ser determinada com base na ordenação de métodos armazenados na segunda parte da estrutura de dados de pacote.

[0024] A máquina pode ser uma máquina virtual implementada em

um dispositivo com restrição a recurso. O dispositivo com restrição a recurso pode ser um cartão inteligente. Os métodos em uma ou mais classes podem ser agrupados em um pacote e o pacote pode ser instalado no cartão inteligente. O processo pode incluir criar um pacote onde o pacote inclui uma estrutura de dados de pacote incluindo primeira e segunda partes. O processo pode incluir concatenar os arranjos de operador de exceção para cada um dos arquivos de classe em uma tabela de operador de exceção, armazenar a tabela de operador de exceção na primeira parte do pacote e todos os métodos na segunda parte do pacote.

[0025] Em outro aspecto, a invenção provê um método que minimiza a quantidade de armazenamento requerida para uma pilha em momento de funcionamento quando executando um programa. A pilha em momento de funcionamento é mantida em funcionamento durante a execução do programa por uma máquina para armazenar um ou mais quadros onde cada quadro inclui um ponteiro de retorno para um método de invocação que chamou um método atualmente em execução no programa. O método inclui combinar informação de operador de exceção para métodos incluídos no programa em uma tabela de operador de exceção combinada e localizar e pesquisar a tabela de operador de exceção combinada quando uma exceção é lançada durante execução de um dos métodos para localizar a informação de operador de exceção sem requerer o armazenamento na pilha de funcionamento de um ponteiro para a informação de operador de exceção.

[0026] Aspectos da presente invenção poderão incluir uma ou mais características. O programa pode ser um programa Java. A máquina pode ser uma máquina virtual que implementa uma máquina virtual Java. O programa pode incluir um pacote de métodos onde os métodos estão em uma ou mais classes. A máquina virtual pode ser implementada em um dispositivo com restrição a recurso, no qual o pacote encontra-se instalado e executando.

[0027] O método pode incluir registrar o pacote em um serviço de

registro em instalação. O serviço de registro mantém um ponteiro e um alcance. O ponteiro indica uma localização no dispositivo com restrição a recurso da tabela de operador de exceção combinada associada com um determinado pacote. O alcance define um alcance de endereços no dispositivo com restrição a recurso ao qual métodos associados com o pacote estão localizados.

[0028] A etapa de localizar pode incluir localizar um pacote associado com um método atualmente executando, incluindo comparar um endereço ao qual uma exceção foi lançada contra o alcance para cada pacote registrado no serviço de registro. A etapa de pesquisar pode incluir pesquisar a tabela de operador de exceção combinada associada com um pacote localizado.

[0029] Em outro aspecto, a invenção provê um método de converter arquivos de classe em um applet convertido para execução em um dispositivo com restrição a recurso e inclui receber um ou mais arquivos de classe onde cada arquivo de classe inclui um ou mais métodos. Cada método inclui um arranjo de operador de exceção que define operadores de exceção lançáveis pelo método. O método inclui definir uma estrutura de dados para armazenar os métodos e operadores de exceção para o applet convertido incluindo uma primeira parte e uma segunda parte, e definir uma ordenação para os métodos e ainda carregar os métodos de acordo com a ordenação na segunda parte da estrutura de dados. Os arranjos de operador de exceção para todos os métodos são combinados em uma única tabela de operador de exceção. Os arranjos de operador de exceção são ordenados na tabela de acordo com a ordenação definida para os métodos. O método inclui armazenar o único arranjo de operador de exceção na primeira parte da estrutura de dados.

[0030] Noutro aspecto, a invenção provê um processo implementado por computador para gerenciar exceções lançáveis durante execução de métodos em uma ou mais classes por uma máquina virtual. Cada método inclui um arranjo de operador de exceção que define operadores de exceção

associados com o método. Os arranjos de operador de exceção individuais são combinados e formam uma única tabela de operador de exceção para dois ou mais métodos. Processo inclui pesquisar a tabela de operador de exceção quando uma exceção é lançada enquanto executando um daqueles métodos, incluindo localizar uma primeira exceção de casamento na única tabela de operador de exceção.

[0031] Noutro aspecto, a invenção provê um sistema de computador incluindo instruções para fazer com que o sistema de computador combine os arranjos de operador de exceção para métodos numa única tabela de operador de exceção.

[0032] Noutro aspecto, a invenção provê um sistema de computador incluindo instruções para fazer com que o sistema de computador combine a informação de operador de exceção para métodos em um programa em uma tabela de operador de exceção combinada e localizar e pesquisar a tabela de operador de exceção combinada quando uma exceção é lançada durante execução de um dos métodos para localizar a informação de operador de exceção sem requerer o armazenamento na pilha de funcionamento de um ponteiro para a informação de operador de exceção.

[0033] Noutro aspecto, a invenção provê um sistema de computador incluindo instrução para fazer com que o sistema de computador receba um ou mais arquivos de classe. Cada arquivo de classe inclui um ou mais métodos e cada método inclui um arranjo de operador de exceção que define operadores de exceção lançáveis pelo método. Instruções são incluídas para fazer com que o sistema de computador defina uma estrutura de dados para armazenar os métodos e operadores de exceção para o applet convertido incluindo primeira e segunda partes, definir uma ordenação para os métodos e ainda carregar os métodos de acordo com a ordenação na segunda parte da estrutura de dados, combinar os arranjos de operador de exceção para dois ou mais métodos uma única tabela de operador de exceção incluindo ordenar arranjos

de operador de exceção de acordo com a ordenação definida para os métodos e armazenar o único arranjo de operador de exceção naquela primeira parte da estrutura de dados.

[0034] Noutro aspecto, a invenção provê um sistema de computador incluindo instruções para fazer com que o sistema de computador pesquise uma tabela de operador de exceção quando uma exceção é lançada enquanto executando um método. Instruções são incluídas para localizar uma primeira exceção de casamento na única tabela de operador de exceção.

[0035] Concretizações da presente invenção podem incluir uma ou mais vantagens. A quantidade de informação exigida a ser armazenada em uma pilha de funcionamento pode ser minimizada enquanto ainda suportando procedimentos de operação de exceção convencionais. Uma tabela de operador de exceção concatenada pode ser gerada, a qual inclui entrada para todas as exceções associadas com um pacote de métodos para uso em um dispositivo com restrição a recurso. Nenhum custo de pilha se exige para suportar operação de exceção. Uma vez que uma exceção é lançada, a máquina virtual Java pode pesquisar eficientemente a tabela de operador de exceção concatenada para o operador de exceção correto. Com uma otimização, a pesquisa da tabela de operador de exceção concatenada pode ser implementada eficazmente sem requerer a avaliação de todas as entradas na tabela concatenada para certas classes de cláusulas de captura aninhadas. As entradas de tabela podem ser ordenadas para assegurar um término de pesquisa de casamento em um operador de nível alto.

[0036] Outros aspectos e vantagens serão prontamente evidentes da descrição detalhada a seguir, dos desenhos acompanhantes e reivindicações.

BREVE DESCRIÇÃO DOS DESENHOS

[0037] Figura 1 é um desenho esquemático de um conjunto de métodos que são ligados entre si no momento de operação por chamadas de procedimento.

[0038] Figura 2 mostra uma tabela de operador de exceção Java convencional incluindo entradas para cada exceção incluída em um método.

[0039] Figura 3 mostra uma área de dados em momento de operação de pilha Java convencional.

[0040] Figura 4 é um diagrama de bloco esquemático que ilustra um sistema exemplar incluindo uma máquina virtual que reside em um cartão inteligente de acordo com a invenção.

[0041] Figura 5a é um fluxograma para um processo de criar uma tabela de operador de exceção concatenada.

[0042] Figura 5b é um diagrama de bloco esquemático para uma estrutura de dados de componente de método incluindo todos os métodos para um pacote a ser executado em uma máquina virtual de Cartão Java™.

[0043] Figura 6 é um diagrama de bloco esquemático de uma tabela de operador de exceção concatenada para um pacote Java.

[0044] Figura 7 é um diagrama de bloco esquemático de um registro de exceção mantido por uma máquina virtual de Cartão Java.

[0045] Figura 8 mostra uma área de dados em momento de operação de pilha mantida por uma máquina virtual de Cartão Java.

[0046] Figura 9 é um fluxograma para um método de tempo de execução para pesquisar uma tabela de operador de exceção concatenada.

[0047] Figura 10a é um fluxograma para um método de tempo de execução otimizado para pesquisar uma tabela de operador de exceção concatenada.

[0048] Figura 10b é um fluxograma detalhado de uma implementação para o método de tempo de execução otimizado descrito na Figura 10a.

DESCRIÇÃO DETALHADA

[0049] Uma estrutura de dados e método para processar as exceções lançadas durante execução de um pacote Java num dispositivo com restrição a recurso são descritos abaixo. Dispositivos com restrições a recurso são, em

geral, considerados como aqueles que são restritos em memória e/ou potência ou velocidade de computação. Embora a implementação particular discutida abaixo seja descrita em referência a um cartão inteligente, a invenção pode ser usada com outros dispositivos com restrições a recurso incluindo, mas não limitados a telefones celulares, dispositivos de escaneamento de margem, dispositivos programáveis de campo, assistentes de dados pessoais (PDAs) e radiolocalizadores, mas também outros dispositivos miniaturas ou pequenos. Tais dispositivos tipicamente tem uma capacidade de memória limitada. Por exemplo, alguns cartões inteligentes têm menos de 1K de memória de acesso aleatório (RAM) como também memória só de leitura limitada (ROM) e/ou memória não volátil, tal como memória só de leitura programável e apagável eletricamente (EEPROM). De maneira semelhante, alguns dispositivos com restrições a recurso são baseados em uma arquitetura projetada para menos que 32 bits. Por exemplo, alguns dos dispositivos com restrições a recurso que podem ser usados com a invenção são baseados em uma arquitetura de 8 bits ou 16 bits, em vez de uma arquitetura de 32 bits.

[0050] Com referência agora à Figura 4, desenvolvimento de um applet para um dispositivo com restrição a recurso, tal como um cartão inteligente 40, começa de uma maneira semelhante ao desenvolvimento de um programa Java. Em outras palavras, um desenvolvedor escreve uma ou mais classes de Java e compila o código de fonte com um compilador Java para produzir um ou mais arquivos de classe 10. O applet pode ser operado, testado e depurado, por exemplo, em uma estação de trabalho usando ferramentas de simulação para imitar o ambiente em cartão inteligente 40. Quando o applet está pronto para ser baixado para cartão inteligente 40, os arquivos de classe 10 são convertidos a um arquivo de applet convertido (CAP) 16 por um conversor 14. O conversor 14 pode ser implementado como uma aplicação de Java sendo executada por um computador de mesa. O conversor 14 aceita como sua entrada um ou mais arquivos de exportação 12 em adição aos arquivos de

classe 10 a serem convertidos. Um arquivo de exportação 12 contém ainda informação de nomeação ou ligação para os conteúdos de outros pacotes que são importados pelas classes sendo convertidas.

[0051] Em termos gerais, o arquivo de CAP 16 inclui todas as classes e interfaces definidas em um único pacote Java e é representado por um fluxo de bytes de 8 bits. Todas as quantidades de 16 e 32 bits são construídas lendo em dois ou quatro bytes de 8 bits, respectivamente. Entre outras coisas, o arquivo de CAP 16 inclui um componente de combinação de constante 18 que é colocado em pacote separadamente de um componente de método 20. O componente de combinação de constante 18 inclui vários tipos de constantes, variando de formas literais numéricas conhecidas em hora de compilação a referências de método e campo que são resolvidas tanto quando o programa é baixado ao cartão inteligente 40 ou na hora de execução pelo cartão inteligente. O componente de método 20 especifica o conjunto de instruções a ser carregado e subseqüentemente executado pelo cartão inteligente 40. Detalhes adicionais da estrutura de um arquivo de CAP 16 exemplar são discutidos em Pedido internacional co-pendente intitulado “AN OBJETCT-ORIENTED INSTRUCTION SET FOR USE WITH OTHER RESOURCE-COSNTRAINED DEVICES”, por Joshua B. Susser e outros, depositado concorrentemente com o presente pedido, que é baseado no pedido U. S. No. De Série 09/243,101.

[0052] Em termos gerais, implementações e applets escritos para uma plataforma com restrição a recurso tal como o cartão inteligente 40 seguem as regras padrão para pacotes de plataforma Java. A máquina virtual Java e a linguagem de programação Java são descritas T. Lindholm e outros, "The Java® Virtual Machine Specification" (1997), e K. Arnold e outros, "The Java® Programming Language Second Edition", (1998), que estão aqui incorporadas por referência em sua totalidade Classes de interface de cartão de aplicação (API) para a plataforma de cartão inteligente são escritas como

arquivos de fonte Java que incluem designações de pacote, onde um pacote inclui várias unidades de compilação e tem um nome único. Mecanismos de pacote são usados para identificar e para controlar acesso a classes, campos e métodos.

[0053] Como é descrito acima, conversor 14 aceita como sua entrada um ou mais arquivos de exportação 12 além dos arquivos de classe 10 a serem convertidos e cria um arquivo de CAP 16. Cada arquivo de classe inclui uma tabela de operador de exceção associada com cada método para uma determinada classe. Conversor 14 concatena todas as tabelas de operador de exceção para todos os métodos de todas as classes associadas com um pacote em uma única tabela de operador de exceção. A tabela de operador de exceção concatenada é mantida com os métodos subjacentes para o pacote e pesquisada, como será descrito em maiores detalhes em seguida, em tempo de execução quando uma exceção é lançada.

[0054] Com referência agora àquela Figura 5a e àquela Figura 5b, um processo 500 executado pelo conversor 14 (Figura 4) para a concatenação de tabelas de operador de exceção para todos os métodos em um pacote começa na etapa 502. Uma ordenação para aqueles métodos em um pacote é definida (504). A mencionada ordenação define a colocação do código para os métodos subjacentes em um componente de método associado 550. Por exemplo, três métodos 530, 535 e 540 (Métodos A, B e C) são incluídos em um pacote 545. Cada um daqueles métodos inclui código (531, 536 e 541, respectivamente), como também uma tabela de operador de exceção associada (532, 537 e 542, respectivamente).

[0055] O componente de método descreve cada um daqueles métodos declarados em um pacote. Os operadores de exceção associados com cada um dos métodos também são descritos. Em uma implementação, o componente de método mencionado é representado pela estrutura conforme representada a seguir:

```
method_component{
u1 tag
u2 size
u1 handlers_count
exception_handler_info exception_handlers[handlers_count]
method_info methods []
}
```

TABELA 1: Componente de Método

[0056] O item de rotulo tem um valor que identifica a estrutura de dados como um componente de método. O item de tamanho indica o número de bytes na estrutura de componente de método, excluindo os itens de rotulo e tamanho. O item de handlers_count representa o número de entradas na tabela de operador de exceção concatenada. O item de exception_handlers representa um arranjo de estruturas de operador de exceção de 8 bytes arranjadas em uma tabela, referenciada como a tabela de operador de exceção concatenada. Cada estrutura de operador de exceção representa uma cláusula de captura ou final definida em um método do pacote. Entradas na tabela de operador de exceção concatenada são escolhidas em ordem ascendente pela distância entre o começo do componente de método ao ponto final de cada faixa de operador de exceção ativo no item de métodos. A ordenação de blocos de operadores de exceção associados com métodos particulares em um pacote é discutida em maior detalhe abaixo.

[0057] O item de métodos representa um arranjo de estruturas de informação de método de comprimento variável. Cada entrada representa um método declarado em uma classe ou interface do determinado pacote. Para propósitos do presente pedido, componente de método 550 pode ser representado como uma estrutura de dados que inclui duas partes, primeira e segundas partes 552 e 554. Primeira parte 552 é um retentor de lugar para os operadores de exceção para todos os métodos subjacentes do pacote. Os

operadores de exceção são armazenados em uma tabela de operador de exceção concatenada 556. O código de método individual é armazenado em segunda parte 554.

[0058] As tabelas de operador de exceção individuais são carregadas na tabela de operador de exceção concatenada e ordenadas conforme a ordenação definida pela etapa 504 (506). Quer dizer, cada tabela de operador de exceção para cada método é carregada em forma de bloco na tabela de operador de exceção concatenada 556. A ordenação local definida em cada tabela de operador de exceção individual para cada método é mantida. No exemplo mostrado em Figura 5b, os métodos são ordenados A, C e B, em segunda parte 554 do componente de método depois da execução de etapa de ordenação 504. As tabelas de operador de exceção associadas A, B e C são carregadas em tabela de operador de exceção concatenada 556 na mesma ordem (A, C então B). A ordenação de operadores de exceção individuais nas tabelas de operador de exceção individuais satisfaz interpretações de linguagem de programação Java convencionais para assegurar um primeiro casamento por uma máquina virtual Java em tempo de execução no casamento mais específico que pode surgir.

[0059] Em uma implementação, exceções são organizadas em uma hierarquia de classe. A hierarquia de classe de exceção tem uma classe de exceção mais alta chamada “Lançável” e dois ramais principais: um conjunto de exceções extraordinariamente sérias, cuja superclasse é a classe de “Erro” e um conjunto de exceções algo menos catastróficas, cuja superclasse é a classe de “Exceção”.

[0060] Quando uma exceção é lançada, uma máquina virtual Java executa o primeiro operador de exceção achado em uma pesquisa de árvore dos operadores de exceção contidos que é aplicável à exceção lançada. Para assegurar que o operador de exceção contido em classe mais baixa aplicável à exceção lançada seja usado, os autores de programas Java (isto é, métodos)

geralmente ordenarão os operadores de exceção dentro de cada método de forma que os operadores de exceção de classe mais baixa são posicionados antes dos operadores de exceção de classe mais alta (onde grau de classificação é determinado por posição da hierarquia de exceção). Ademais, quando há dois ou mais operadores de exceção contidos para exatamente a mesma condição de exceção que são aplicáveis a uma instrução que faz com que exceção seja lançada, uma máquina virtual Java executa o operador de exceção contido aplicável estabelecido pelo método mais próximo na cadeia de chamadas de método ao método que causa a exceção lançada.

[0061] Depois que as tabelas de operador de exceção individuais são ordenadas e carregadas na etapa 506, cada entrada de operador de exceção em tabela de operador de exceção concatenada 556 é convertida a uma estrutura de dados otimizada para uso num dispositivo com restrição a recurso tal como cartão inteligente 40 (Figura 4) (508). Em uma implementação, a estrutura de dados para uma entrada otimizada em tabela de operador de exceção concatenada 556 é mostrada em Figura 6. Tabela de operador de exceção concatenada 600 inclui uma pluralidade de entradas 602, cada uma das quais inclui dados conforme uma estrutura de dados de informação de operador de exceção. A estrutura de dados de informação de operador de exceção inclui uma compensação de começo 604, comprimento ativo 606, compensação de operador 610 e índice de tipo de captura 612.

[0062] Compensação de começo 604 e uma compensação de término (não mostrada) são compensações de byte no componente de método particular. A compensação de começo e término indica a faixa em um arranjo de código de byte ao qual o operador de exceção está ativo. O valor de compensação de começo 604 deve ser uma compensação válida em um arranjo de código de byte ao código de uma instrução.

[0063] A compensação de término é definida como a soma de compensação de começo 604 mais comprimento ativo 606. O valor da

compensação de término tanto deve ser uma compensação valida em um arranjo de código do código de uma instrução ou deve ser igual a uma contagem de código de byte do método (o comprimento do arranjo de código). O valor de compensação de começo 604 deve ser menos que o valor da compensação de término. Compensação de começo 604 é inclusiva e a compensação de término é exclusiva; quer dizer, o operador de exceção deve ser ativo enquanto o endereço de execução está dentro do intervalo [compensação de começo, compensação de término].

[0064] Comprimento ativo 606 define em códigos de bytes a faixa de instruções contidas pelo operador de exceção dado. Em uma implementação, comprimento ativo 606 é codificado para indicar se a faixa ativa do operador de exceção particular está aninhada dentro de outro operador de exceção, e mais especificamente, se o operador de exceção atual é o último operador em uma lista de operadores de exceção associados com um bloco de código protegido particular. Para programas escritos na linguagem de programação Java, o bit é codificado para indicar se o operador de exceção atual é o último operador (cláusula de captura ou final) em uma lista de operadores de exceção para um bloco de código protegido. O bit alto de comprimento ativo 606 é colocado a um ("1") se o bloco de código protegido não está contido dentro de outro bloco de código protegido e o operador atual é o último operador associado com o bloco de código protegido. O bit é colocado a zero ("0") se o bloco de código protegido está contido dentro de outro bloco de código protegido ou o operador atual não é o último operador associado com o bloco de código protegido. Onde a codificação é usada, a compensação de término é definida como a soma de compensação de começo 604 mais comprimento ativo 606 e 0x7FFF.

[0065] Compensação de operador 608 representa uma compensação de byte no item info do componente de método. Mais especificamente, compensação de operador 608 indica o começo do operador de exceção e

uma localização em qual ponto execução é para recomeçar pela Máquina virtual Java quando a exceção particular é capturada. O valor de compensação de operador 608 deve ser uma compensação válida em um arranjo de código de byte do método para um código de uma instrução, e deve ser menos que o valor da contagem de código de byte do método.

[0066] Índice de tipo de captura 610 indica o tipo de operador de execução. Para transferir controle ao operador de exceção, o pc associado com a exceção lançada deve cair dentro da faixa definida para o operador de exceção e ser do mesmo tipo. Se o valor do índice de tipo de captura 610 é não zero, ele deve ser um índice válido na tabela de combinação de constante que representa a classe da exceção capturada por este operador de exceção. Se o operador de exceção representa uma cláusula final, o valor de índice de tipo de captura 610 é colocado a zero. Um operador de exceção de cláusula final é chamado para todas as exceções que são lançadas dentro da faixa de compensação de começo e compensação de término, independente de tipo.

[0067] Com referência novamente à Figura 4, depois de conversão, o arquivo de CAP 16 pode ser armazenado em um meio legível por computador 17 tal como um acionamento de disco rígido, um disquete, um meio de armazenamento óptico, um dispositivo flash ou algum outro meio adequado.

[0068] O arquivo de CAP 16 então pode ser copiado ou transferido para um terminal 22 tal como um computador de mesa com um dispositivo de aceitação de cartão periférico (CAD) 24. O CAD 24 permite informação ser escrita e recuperada do cartão inteligente 40. O CAD 24 inclui uma porta de cartão (não mostrada) na qual o cartão inteligente 40 pode ser inserido. Uma vez inserido, contatos de um conector pressionam contra uma área de conexão de superfície no cartão inteligente 40 para prover potência e permitir comunicações com o cartão inteligente 40. O terminal 22 também inclui uma ferramenta de instalação ou programa 26 que carrega o arquivo de CAP 16 e o transmite ao cartão 40.

[0069] O cartão inteligente 40 tem uma porta de entrada/ saída (I/O) 42 incluindo um conjunto de contatos pelos quais programas, dados e outras comunicações podem ser providos. O cartão inteligente 40 também inclui uma ferramenta de instalação 46 para receber os conteúdos do arquivo de CAP 16 e preparar o applet para execução no cartão 40. A ferramenta de instalação 46 pode ser implementada, por exemplo, como um programa Java e pode ser executada no cartão inteligente 40. O cartão inteligente 40 também tem memória, incluindo memória volátil tal como RAM 50 e memória não volátil tal como memória só de leitura eletricamente apagável e programável (EEPROM) 54. O cartão inteligente 40 também tem ROM tal como ROM 52. O applet (arquivo de CAP 16) preparado pelo conversor 14 pode ser armazenado em EEPRROM 54.

[0070] Como parte do processo de instalação uma ou mais áreas de dados são criadas pela ferramenta de instalação 46 para facilitar operações em tempo de execução. Um exemplo de uma área de dados é um registro de exceção. O registro de exceção é uma lista usada por uma máquina virtual Java Card® quando interpretando exceções lançadas. Com referência agora às Figuras 4 e 7, registro de exceção 700 pode ser uma estrutura de dados de lista ligada que é extensível. Entradas novas podem ser adicionadas à cabeça da lista. Cada entrada 702 em registro de exceção 700 representa um componente de método 550 (ver Figura 5b) contendo operadores de exceção. Componentes de método sem operadores de exceção não estão incluídos.

[0071] Durante instalação de um arquivo de CAP, ferramenta de instalação 46 chama uma máquina virtual de Cartão Java para registrar o componente de método associado com um pacote. Cada entrada inclui um endereço 704 do componente de método associado e seu tamanho 706. Endereço 704 e tamanho 706 definem uma faixa, com referência ao contador de programa (pc), para instruções que são incluídas em um pacote. O uso do registro de exceção é descrito em maior detalhe abaixo.

[0072] Como descrito acima no Fundamento, uma máquina virtual Java mantém várias estruturas de dados de funcionamento para manter rastro da execução e invocação de métodos. Uma máquina virtual de Cartão Java faz o mesmo. Em tempo de execução, uma máquina virtual de Cartão Java mantém um pc e uma pilha de funcionamento. Uma pilha de funcionamento armazena quadros de máquina virtual de Cartão Java. Com referência agora à Figura 8, uma pilha de funcionamento 800 incluindo uma pluralidade de quadros de máquina virtual de Cartão Java 802 é mostrada. Cada quadro 802 inclui um ponteiro de retorno 806.

[0073] O ponteiro de retorno 806 é um ponteiro para o método que invocou o método atual, e, mais especificamente, para a localização onde a execução é para recomeçar no método de invocação na conclusão de execução do método atual.

[0074] Como explicado mais abaixo, nenhuma referência ao método que está executando atualmente necessita ser mantida na pilha de operação. Todos os operadores de exceção para todos os métodos associados em um pacote encontram-se armazenados numa única tabela de operador de exceção concatenada que pode ser pesquisada em tempo de execução para um operador de exceção relevante ao pc atual.

[0075] Com referência agora à Figura 9, um processo 900 para efetuar operação de exceção em tempo de execução começa na etapa 902. Uma máquina virtual de Cartão Java tanto lança explicitamente ou implicitamente uma exceção (904). A classe da exceção é determinada (906) junto com a posição de pc onde a exceção foi lançada (908).

[0076] Uma pesquisa é executada no registro de exceção para localizar um componente de método que contém o pc onde a exceção foi lançada (910). Isto determina em qual pacote o método do quadro atual é implementado. Se nenhum componente de método contendo casamento é achado (e conseqüentemente, nenhuma tabela de operador de exceção

concatenada) (912), então uma verificação é feita para determinar se um quadro de pilha está disponível para extrair a pilha de funcionamento (914). Caso nenhum quadro de pilha esteja mais disponível para extrair a pilha de funcionamento, então a máquina virtual de Cartão de Java é detida com uma exceção não capturada (916). Se outro quadro está disponível para ser extraído da pilha de funcionamento, então o quadro é extraído (918). A extração do quadro inclui restabelecer o quadro prévio e colocar o pc a uma valor novo, isto é, a localização de retorno indicada pelo ponteiro de retorno armazenado na pilha de funcionamento e associado com o quadro prévio. Depois disso, o processo continua na etapa 910. A extração de entradas fora da pilha de funcionamento permite aninhar os métodos entre pacotes. Quer dizer, um método em um primeiro pacote pode ser chamado (invocado) por um método de invocação no segundo pacote distinto. Como tal, o método de invocação é dito conter o método chamado, e, por conseguinte, pode incluir um operador de exceção que é aplicável no evento que o método chamado não captura uma determinada exceção.

[0077] Se um componente de método contido é localizado na etapa de pesquisa 910, então uma entrada na tabela de operador de exceção concatenada associada, (a tabela de operador de exceção concatenada atual), associada com o componente de método é recuperada (neste caso, a primeira entrada) (930). Em uma implementação, o endereço de partida armazenado no registro de exceção aponta para o endereço para a primeira entrada na tabela de operador de exceção concatenada para o determinado componente de método. Como tal, a execução das etapas seguintes pode proceder diretamente depois da recuperação do endereço de partida do registro de exceção.

[0078] A faixa associada com a entrada atual sendo testada é verificada para determinar se ela contém o pc onde a exceção foi lançada (932). Se nenhum casamento surge, então uma próxima entrada na tabela de

operador de exceção concatenada atual é recuperada na etapa 930. Se não mais entradas estão disponíveis (934), então o processo continua na etapa 914 verificando para determinar se mais quadros de pilha estão disponíveis para serem extraídos como descrito acima.

[0079] Se a faixa inclui o pc na etapa 932, então uma verificação é feita para determinar se o tipo de exceção especificado pelo operador de exceção casa com a exceção lançada (936). O tipo casa se o tipo de exceção lançada é da mesma classe como a classe especificada ou uma subclasse da classe especificada. Casamento também acontece se o operador de exceção corresponde a uma cláusula final. Uma cláusula final é uma interpretação de linguagem de programação Java que permite um programador definir um operador de exceção que será executado para todos os tipos de classe e como tal é um casamento para todos os tipos de exceções lançadas. Se o tipo casa, o operador de exceção associado com a entrada atual é executado (938) e o processo se completa (940). Mais especificamente, o pc é colocado baseado no endereço armazenado na entrada atual para o operador de exceção (compensação de operador 610 de Figura 6) e execução continua com a primeira declaração do operador de exceção capturado.

[0080] Em uma implementação, nem todas as entradas na tabela de operador de exceção concatenada são requeridas serem testadas antes de uma determinação que uma exceção não capturada foi lançada. Com referência agora à Figura 10a, depois do teste de tipo na etapa 936, um teste adicional é executado para determinar se outra declaração de tentativa contida está incluída no método atual (937). Como descrito acima, uma declaração de tentativa contida reflete outro operador de exceção cuja faixa contém o pc atual. Se nenhuma outra declaração de tentativa contida está presente, então o processo pode saltar imediatamente para a etapa de extração de pilha de funcionamento 914 independente de se outras entradas na tabela de operador de exceção concatenada atual permanecem para serem processadas.

[0081] Em uma implementação, etapa 937 pode ser invocada usando o bit de otimização (o bit alto codificado do item de comprimento ativo 606) armazenado na tabela de operador de exceção concatenada para a entrada atual. Mais especificamente, o bit alto do comprimento ativo para a entrada atual é recuperado (937a). Uma verificação é feita para determinar se o bit está colocado (937b). Se o bit não está colocado (indicando que uma declaração de tentativa contida está presente ou que operadores adicionais para a mesma declaração de tentativa estão presentes) então o processo continua na etapa 934. Se o bit está colocado, então o processo continua na etapa 914.

[0082] Enquanto a presente invenção foi descrita com referência a algumas concretizações especificas, a descrição é ilustrativa da invenção e não é para ser interpretada como limitante da invenção. Várias modificações podem ocorrer àqueles qualificados na técnica sem partir do escopo da invenção como definida pelas reivindicações anexas.

[0083] A presente invenção é aplicável a programas e métodos escritos em linguagens de programação diferentes de Java, incluindo linguagens de programação que são compiladas para código dependente de plataforma. As técnicas descritas podem ser aplicadas em outros contextos, por exemplo, a programas escritos em linguagem C ou Pascal onde manipulações de pilha em tempo de execução semelhantes são requeridas.

REIVINDICAÇÕES

1. Processo implementado por computador (22) para gerenciar exceções lançáveis durante execução de métodos (530, 535, 540) em uma ou mais classes (10) por uma máquina (49), cada método incluindo um arranjo de operadores de exceção (532, 537, 542) que definem operadores de exceção associados com o método, processo este caracterizado por compreender as etapas de:

definir uma ordenação para dois ou mais métodos (531, 536, 541), incluindo definir uma colocação de código protegido numa segunda parte (554) de um pacote (550);

combinar os arranjos de operadores de exceção (532, 537, 542) para os dois ou mais métodos em uma única tabela de operador de exceção (600), incluindo posicionar os arranjos de operadores de exceção para os dois ou mais métodos numa primeira parte (552) do pacote com base na ordenação para os dois ou mais métodos, em que os arranjos de operadores de exceção são posicionados em ordem ascendente por uma distância entre um começo de um componente de método associado a um ponto final de cada faixa de operador de exceção ativa no item de métodos associados;

indicar uma faixa ativa para cada operador de exceção; e

quando uma exceção for lançada (904) durante execução de um método particular dentro dos dois ou mais métodos, pesquisar em tal tabela de operador de exceção (600) combinada para localizar um operador de exceção para o método particular.

2. Processo de acordo com a reivindicação 1, caracterizado pelo fato de que inclui combinar todos os arranjos de operadores de exceção (532, 537, 542) para todos os métodos numa classe (10) na única tabela de operador de exceção (600).

3. Processo de acordo com a reivindicação 1, caracterizado pelo fato de que inclui combinar todos os arranjos de operadores de exceção (532,

537, 542) para todos os métodos em todas as classes (10) na única tabela de operador de exceção (600).

4. Processo de acordo com a reivindicação 1, caracterizado pelo fato de que inclui combinar todos os arranjos de operadores de exceção (532, 537, 542) para todos os métodos num pacote de Java (545) na única tabela de operador de exceção (600).

5. Processo de acordo com a reivindicação 1, caracterizado pelo fato de que um método (530, 535, 540) é incluído num arquivo de classe (10) e a etapa de combinar todos os arranjos de operadores de exceção (532, 537, 542) inclui combinar os arranjos de operadores de exceção (532, 537, 542) para todos os métodos (530, 535, 540) em um arquivo de classe (10) na única tabela de operador de exceção (600).

6. Processo de acordo com a reivindicação 1, caracterizado pelo fato de que inclui ainda:

pesquisar tal tabela de operador de exceção (600) quando uma exceção for lançada (904), enquanto executando um dos métodos (530, 535, 540), incluindo localizar (932) uma primeira exceção de casamento na única tabela de operador de exceção (600).

7. Processo de acordo com a reivindicação 6, caracterizado pelo fato de que a etapa de pesquisar inclui recuperar (930), em ordem, entradas de operadores de exceção da tabela de operador de exceção (600) e verificar (936, 932) o tipo e a faixa de cada operador de exceção para o primeiro operador de exceção de casamento.

8. Processo de acordo com a reivindicação 7, caracterizado pelo fato de que compreende ainda parar (934) pesquisa se um operador de exceção atual não casa e se for contido (937) o último operador para o nível mais alto de código protegido em um método associado.

9. Processo de acordo com a reivindicação 1, caracterizado pelo fato de que os arquivos de classe (10) são arquivos de classe de Java.

10. Processo de acordo com a reivindicação 1, caracterizado pelo fato de que os métodos (530, 535, 540) em uma ou mais classes (10) são agrupados em um pacote (550), onde o pacote (550) inclui uma estrutura de dados de pacote que inclui primeira e segunda partes, tal processo incluindo armazenar a tabela de operador de exceção (600) na primeira parte (552) do pacote e todos os métodos na segunda parte (554) do pacote.

11. Processo de acordo com a reivindicação 10, caracterizado pelo fato de a etapa de combinar incluir concatenar os arranjos de operadores de exceção, incluindo carregar cada arranjo de operador de exceção (532, 542, 537) na primeira parte (552) da estrutura de dados de pacote, conforme uma ordenação predefinida.

12. Processo de acordo com a reivindicação 11, caracterizado pelo fato de a ordenação predefinida ser determinada com base na ordenação de métodos (531, 541, 536) armazenados na segunda parte (554) da estrutura de dados de pacote.

13. Processo de acordo com a reivindicação 1, caracterizado pelo fato de tal máquina ser uma máquina virtual (49) implementada em um dispositivo com restrição a recurso.

14. Processo de acordo com a reivindicação 13, caracterizado pelo fato de tal dispositivo com restrição a recurso ser um cartão inteligente (40).

15. Processo de acordo com a reivindicação 14, caracterizado pelo fato de os métodos em uma ou mais classes (10) serem agrupados em um pacote (550) e o pacote ser instalado no cartão inteligente (40).

16. Processo de acordo com a reivindicação 15, caracterizado pelo fato de que inclui ainda criar um pacote (550), onde o pacote inclui uma estrutura de dados de pacote que inclui primeira e segunda partes, tal processo incluindo concatenar os arranjos de operadores de exceção (532, 537, 542) para cada um dos métodos (531, 536, 541) numa tabela de operador de exceção

(600), armazenar a dita tabela de operador de exceção (600) na primeira parte (552) do pacote e todos os métodos (531, 536, 541) na segunda parte (554) do pacote.

17. Método que minimiza a quantidade de armazenamento requerida para uma pilha em tempo de execução (800) quando executando um programa, a pilha em tempo de execução mantida em tempo de execução durante a execução do programa por uma máquina para armazenar um ou mais quadros (802), onde cada quadro inclui um ponteiro de retorno (806) para um método de invocação o qual chamou um método atualmente em execução no programa, método este compreendendo as etapas de:

preparar um ou mais quadros (802) para um programa, o qual inclui

definir uma ordem para métodos incluídos no programa, incluindo definir uma colocação de código protegido no um ou mais quadros,

caracterizado pelo fato de compreender ainda as etapas de:

combinar informação de operador de exceção para os ditos métodos incluídos no programa em uma tabela de operador de exceção (600) combinada, incluindo posicionar a informação de operador de exceção para os métodos no um ou mais quadros (802) com base na ordenação para os métodos, em que a informação de operador de exceção é posicionada em ordem ascendente por uma distância entre um começo de um componente de método associado a um ponto final de cada faixa de operador de exceção ativa no item de métodos associados, e

indicar uma faixa ativa para cada operador de exceção; e

pesquisar a tabela de operador de exceção combinada quando uma exceção for lançada (904), durante execução de um dos métodos (531, 536, 541), para localizar a informação de operador de exceção, sem requerer o armazenamento na pilha em tempo de execução (800) de um ponteiro para a informação de operador de exceção.

18. Método de acordo com a reivindicação 17, caracterizado pelo fato de o ponteiro ser um ponteiro direto para a informação de operador de exceção.

19. Método de acordo com a reivindicação 17, caracterizado pelo fato de o programa ser um programa Java.

20. Método de acordo com a reivindicação 19, caracterizado pelo fato de a máquina ser uma máquina virtual (49) a qual implementa uma máquina virtual Java®.

21. Método de acordo com a reivindicação 17, caracterizado pelo fato de o programa incluir um pacote (550) de métodos (531, 536, 541), os ditos métodos em uma ou mais classes (10), e onde a máquina virtual (49) é implementada em um dispositivo com restrição a recurso, no qual o pacote está instalado e em execução.

22. Método de acordo com a reivindicação 21, caracterizado pelo fato do dispositivo com restrição a recurso ser um cartão inteligente (40).

23. Método de acordo com a reivindicação 21, caracterizado pelo fato de incluir ainda registrar o pacote (550) em um serviço de registro (700) em instalação, o serviço de registro (700) mantendo um ponteiro (704) e um alcance (706), o ponteiro (704) indicando uma localização no dispositivo com restrição a recurso da tabela de operador de exceção combinada (600) associada com um determinado pacote, o alcance (706) definindo uma faixa de endereços no dispositivo com restrição a recurso, no qual métodos associados com o pacote (550) estão localizados.

24. Método de acordo com a reivindicação 23, caracterizado pelo fato de a etapa de localizar incluir localizar um pacote (550) associado com um método atualmente em execução, o qual inclui comparar (910) um endereço ao qual uma exceção foi lançada contra o alcance para cada pacote registrado no serviço de registro (700), aquela etapa de pesquisar incluindo pesquisar a tabela de operador de exceção combinada (600) associada com

um pacote localizado.

25. Método para converter arquivos de classe (10) em um applet (16) convertido para execução num dispositivo com restrição a recurso, o dito método incluindo a etapa de:

receber um ou mais arquivos de classe (10), cada arquivo de classe (10) incluindo um ou mais métodos (531, 536, 541), cada método (531, 536, 541) incluindo um arranjo de operador de exceção definindo operadores de exceção (532, 537, 542) capturáveis pelo método;

sendo caracterizado pelo fato de incluir ainda as etapas de:

definir uma estrutura de dados (550) para armazenar os métodos e operadores de exceção para o applet (16) convertido incluindo uma primeira parte (552) e uma segunda parte (554);

definir uma ordenação para os métodos e carregar os métodos de acordo com a ordenação na segunda parte (554) da estrutura de dados;

combinar os arranjos de operadores de exceção para todos os métodos numa única tabela de operador de exceção (600), incluindo ordenar os arranjos de operadores de exceção conforme a ordenação definida para os métodos e armazenar a única tabela de operador de exceção na primeira parte (552) da estrutura de dados com base na ordenação definida para os métodos, em que os ditos arranjos de operadores de exceção são combinados em ordem ascendente por uma distância entre um começo de um componente de método associado a um ponto final de cada faixa de operador de exceção ativa no item de métodos associados;

indicar uma faixa ativa para cada operador de exceção; e

quando uma exceção for lançada (904) durante execução de um método particular, pesquisar em tal única tabela de operador de exceção (600) para localizar um operador de exceção para o método particular.

26. Processo implementado por computador (22) para gerenciar exceções lançáveis durante execução de dois ou mais métodos (530, 535, 540)

numa ou mais classes (10) por uma máquina virtual (49), cada método incluído em uma classe e incluindo um arranjo de operadores de exceção (532, 537, 542) definindo operadores de exceção associados com o método, os arranjos de operador de exceção individuais combinados e formando uma única tabela de operador de exceção (600) para os dois ou mais métodos, caracterizado pelo fato de compreender as etapas de:

definir uma ordenação para dois ou mais métodos (531, 536, 541), incluindo definir uma colocação de código protegido numa primeira parte (552) de um pacote (550);

combinar os arranjos de operadores de exceção (532, 537, 542) para os dois ou mais métodos em uma única tabela de operador de exceção (600), incluindo posicionar os arranjos de operadores de exceção para os dois ou mais métodos numa segunda parte (554) do pacote com base na ordenação para os dois ou mais métodos, em que os arranjos de operadores de exceção são posicionados em ordem ascendente por uma distância entre um começo de um componente de método associado a um ponto final de cada faixa de operador de exceção ativa no item de métodos associados;

indicar uma faixa ativa para cada operador de exceção; e

pesquisar tal tabela de operador de exceção (600) quando uma exceção for lançada (904), enquanto executando um dos métodos (530, 535, 540), incluindo localizar (932) uma primeira exceção de casamento na única tabela de operador de exceção (600).

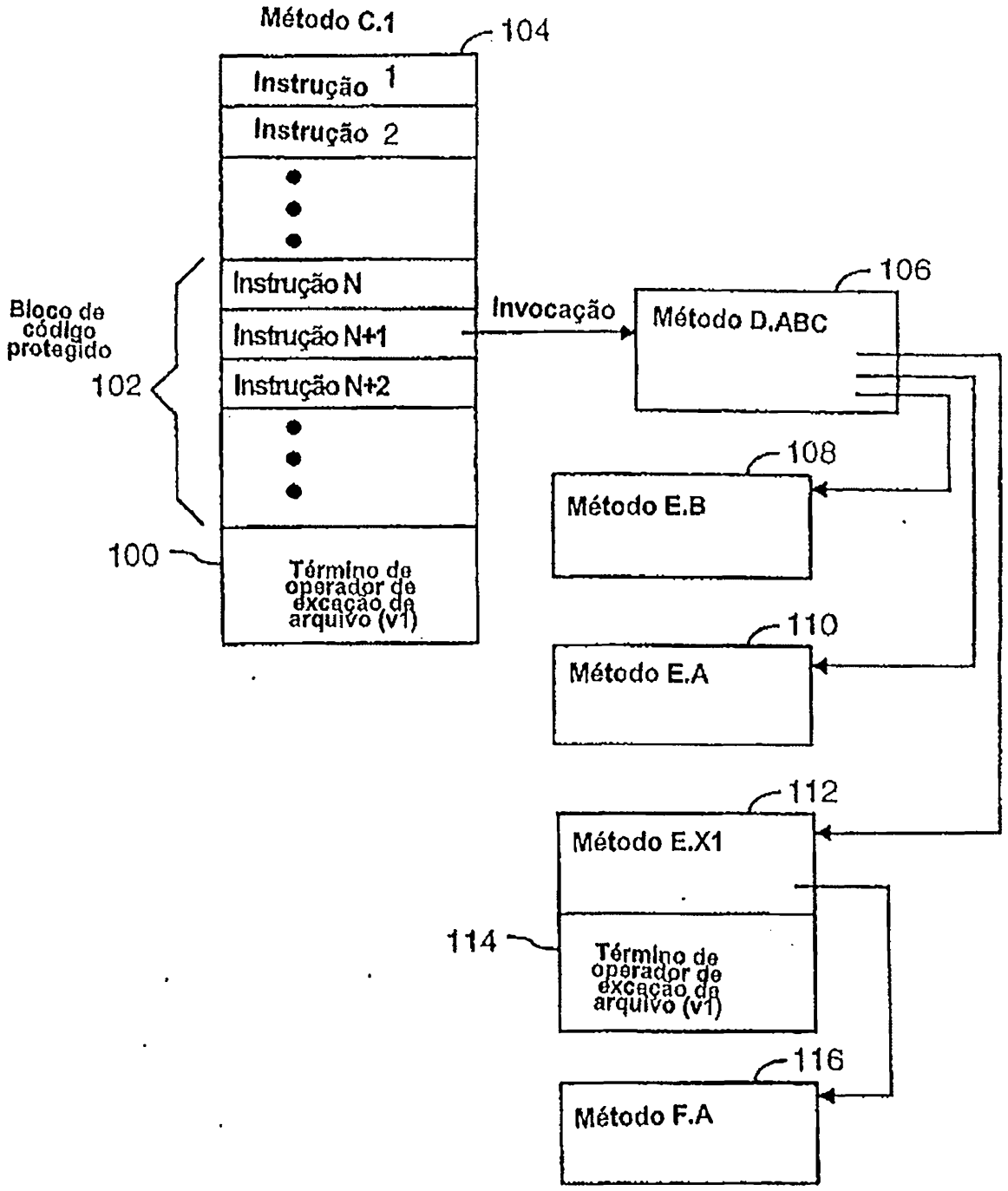
Método C.1
104
Instrução 1
Instrução 2
Instrução N
Instrução N+1
Instrução N+2
Término de operador de exceção de arquivo (v1)
Bloco de código protegido
102
100
Invocação
106
Método D.ABC
108
Método E.B
110
Método E.A
112
Método E.X1
114
Término de operador de exceção de arquivo (v1)
116
Método F.A

FIG. 1

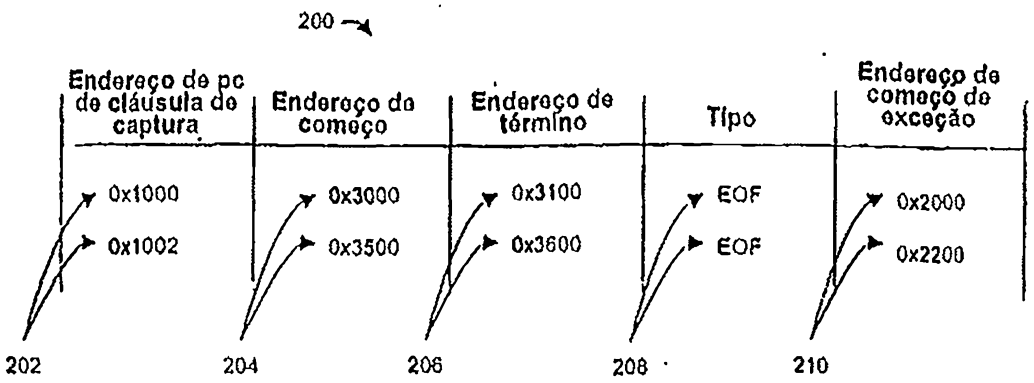
200
Endereço de pc de cláusula de captura
Endereço do começo
Endereço de término
Tipo
Endereço de começo de exceção
0x1000
0x1002
0x3000
0x3500
0x3100
0x3600
EOF
EOF
0x2000
0x2200
202
204
206
208
210

FIG. 2

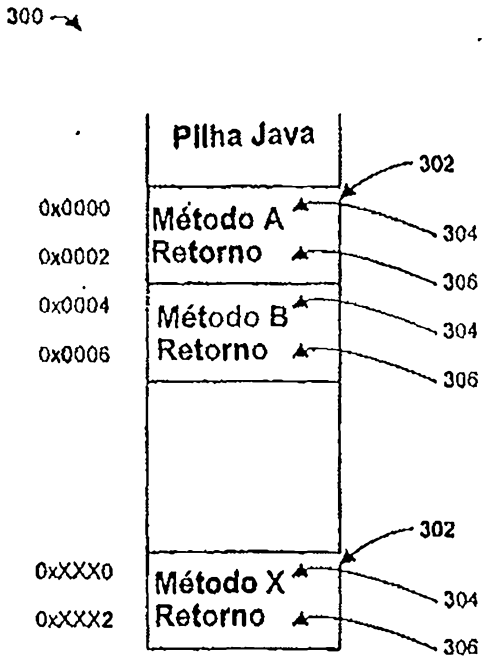
300
Pilha Java
302
0x0000
Método A
304
0x0002
Retorno
306
0x0004
Método B
304
0x0006
Retorno
306
302
0xXXX0
Método X
304
0xXXX2
Retorno
306

FIG. 3

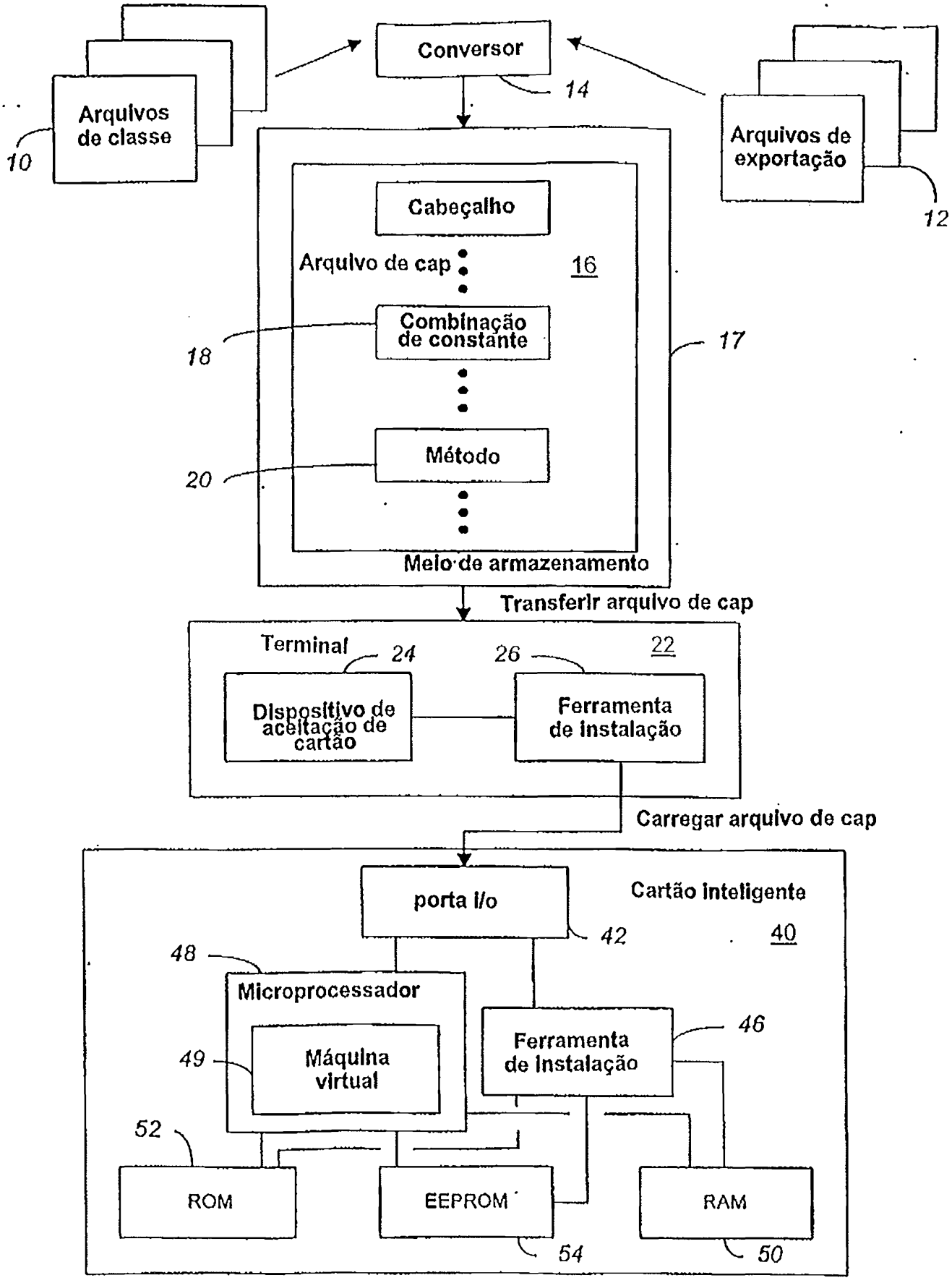
Conversor
14
Arquivos de classe
10
Arquivos de exportação
12
Cabeçalho
Arquivo de cap
16
Combinação de constante
18
17
Método
20
Meio de armazenamento
Transferir arquivo de cap
Terminal
24
26
22
Dispositivo de aceitação de cartão
Ferramenta de instalação
Carregar arquivo de cap
porta I/o
Cartão inteligente
42
40
48
Microprocessador
49
Máquina virtual
Ferramenta de instalação
46
52
ROM
EEPROM
54
RAM
50

FIG. 4

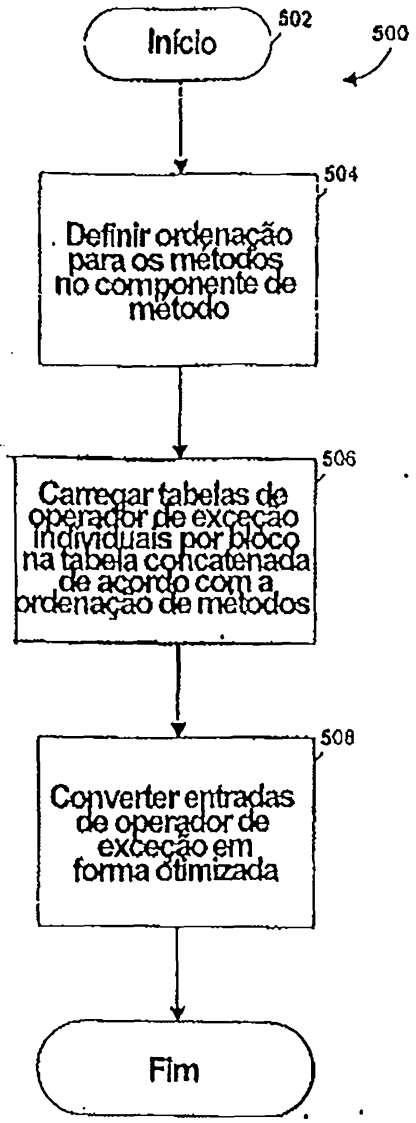
Início
502
500
Definir ordenação para os métodos no componente de método
504
Carregar tabelas de operador de exceção individuais por bloco na tabela concatenada de acordo com a ordenação de métodos
506
Converter entradas de operador de exceção em forma otimizada
508
Fim

FIG. 5a

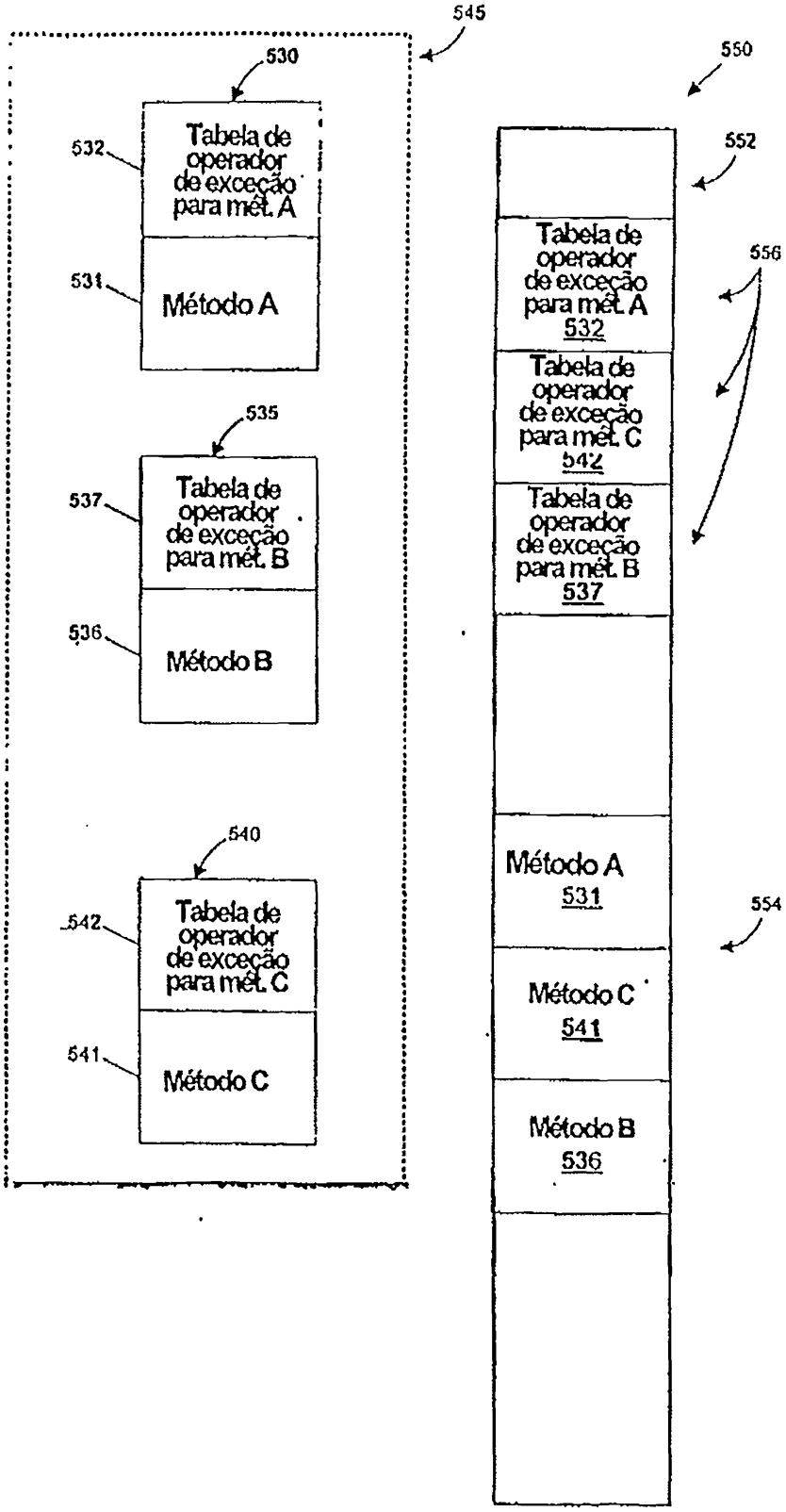
545
530
Tabela de operador de exceção para mét. A
532
Método A
531
535
Tabela de operador de exceção para mét. B
537
Método B
536
540
Tabela de operador de exceção para mét. C
542
Método C
541
550
552
Tabela de operador de exceção para mét. A 532
556
Tabela de operador de exceção para mét. C 542
Tabela de operador de exceção para mét. B 537
Método A 531
554
Método C 541
Método B 536

FIG. 5b

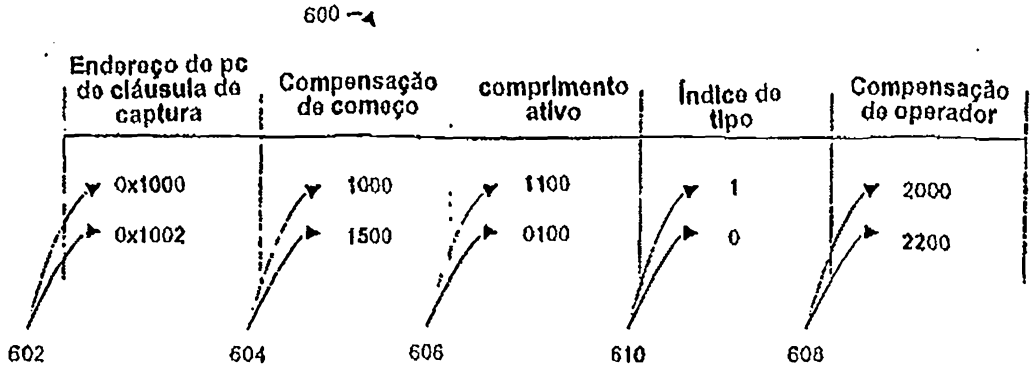

FIG. 6

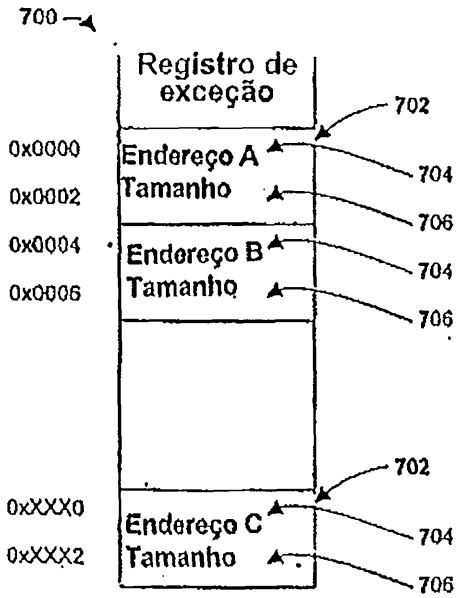

FIG. 7

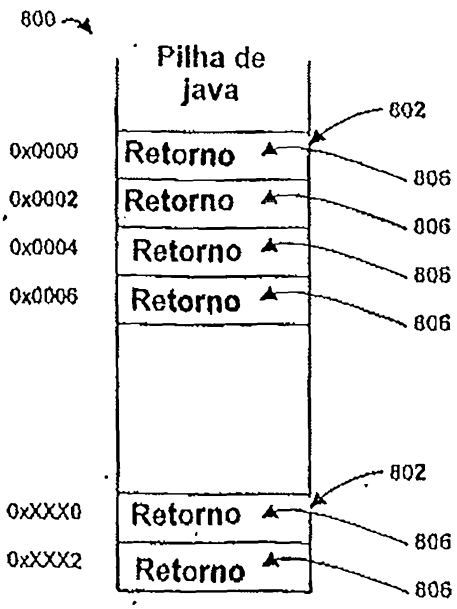

FIG. 8

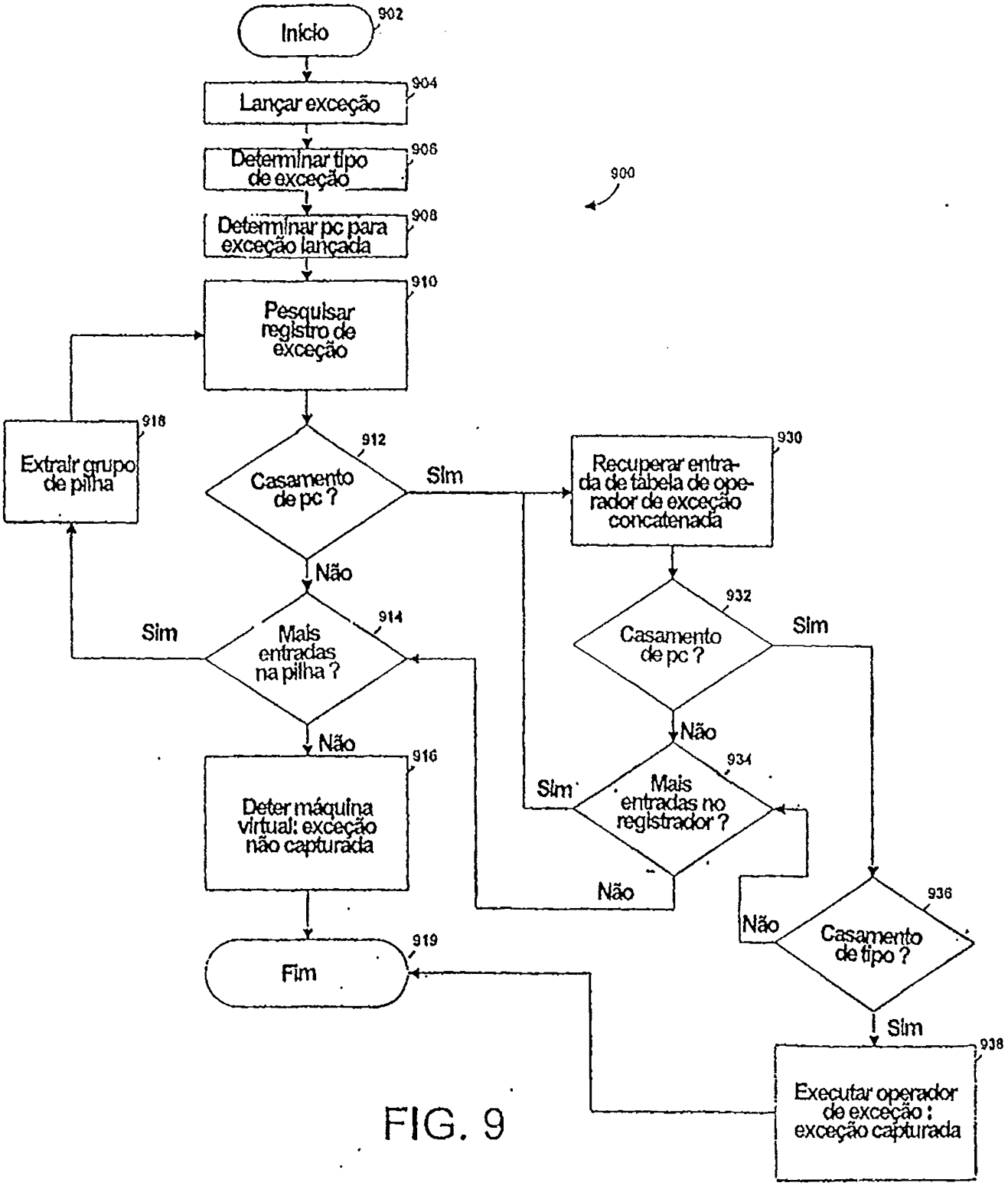
Início
902
Lançar exceção
904
Determinar tipo de exceção
906
Determinar pc para exceção lançada
908
Pesquisar registro de exceção
910
900
Extrair grupo de pilha
918
Casamento de pc ?
912
Sim
Não
Recuperar entrada de tabela de operador de exceção concatenada
930
Casamento de pc ?
932
Sim
Não
Mais entradas na pilha ?
914
Sim
Não
Mais entradas no registrador ?
934
Sim
Não
Deter máquina virtual: exceção não capturada
916
Casamento de tipo ?
936
Não
Sim
Fim
919
Executar operador de exceção : exceção capturada
938

FIG. 9

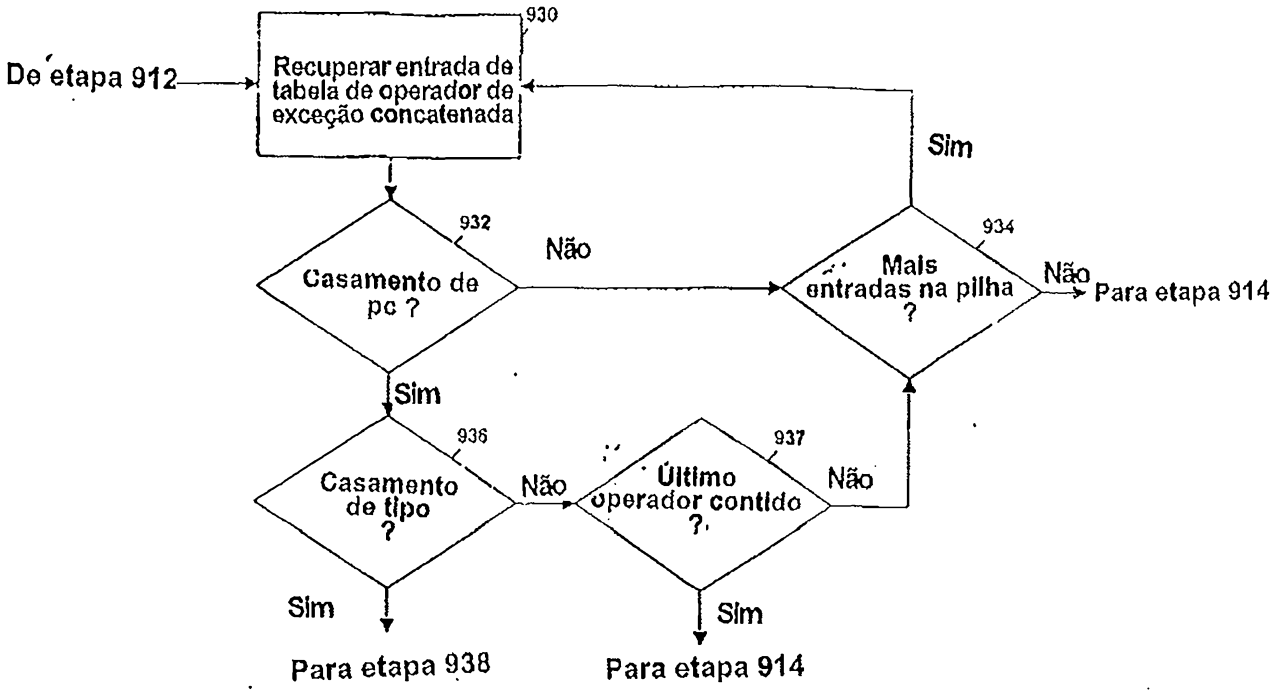

FIG. 10a

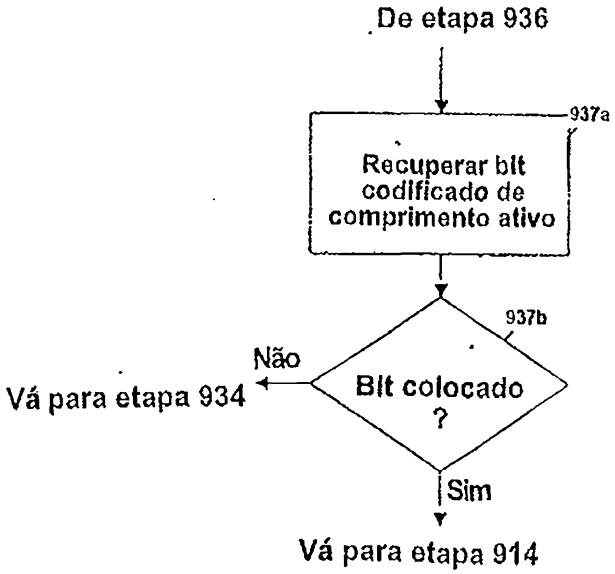

FIG. 10b