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

(21) **PI0611318-4 A2**

BRPI0611318A2

(22) Data de Depósito: 01/06/2006
(43) Data da Publicação: 31/08/2010
(RPI 2069)

(51) Int.Cl.:
G06F 11/34

(54) Título: **MELHORAMENTOS EM ARQUITETURA DE MONITORAMENTO DE DESEMPENHO PARA ANÁLISE BASEADA EM PERCURSO CRÍTICO**

(30) Prioridade Unionista: 01/06/2005 US 11/143,425

(73) Titular(es): INTEL CORPORATION

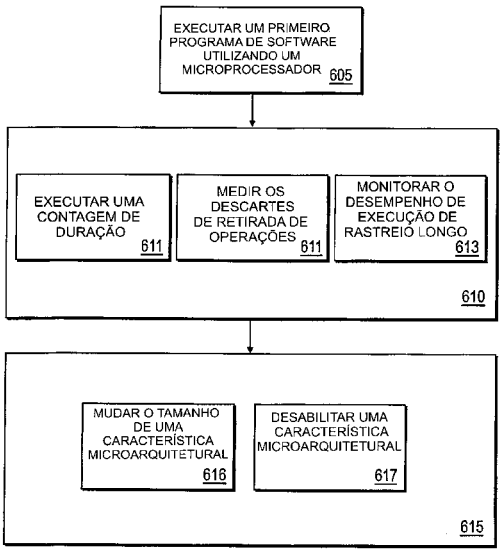
(72) Inventor(es): CHRIS NEWBURN

(74) Procurador(es): Dannemann ,Siemsen, Bigler & Ipanema Moreira

(86) Pedido Internacional: PCT US2006021434 de 01/06/2006

(87) Publicação Internacional: WO 2006/130825de 07/12/2006

(57) Resumo: MELHORAMENTOS EM ARQUITETURA DE MONITORAMENTO DE DESEMPENHO PARA ANÁLISE BASEADA EM PERCURSO CRÍTICO. A presente invenção refere-se a um método e aparelho para monitorar o desempenho de uma micro arquitetura e sintonizar a micro arquitetura com base no desempenho monitorado. O desempenho é monitorado através de simulação, raciocínio analítico, medição de descarte de retirada, tempo de execução total, e outros métodos para determinar os custos de evento por instância. Com base nos custos de evento por instância, a micro arquitetura e/ou o software de execução é sintonizado para melhorar o desempenho.



Relatório Descritivo da Patente de Invenção para **"MELHORAMENTOS EM ARQUITETURA DE MONITORAMENTO DE DESEMPENHO PARA ANÁLISE BASEADA EM PERCURSO CRÍTICO"**.

CAMPO DA INVENÇÃO

A presente invenção refere-se ao campo de sistemas de computador, e especificamente ao monitoramento de desempenho e sintonia de microarquiteturas.

ANTECEDENTES DA INVENÇÃO

A análise de desempenho é o princípio para caracterizar, depurar, e sintonizar um projeto microarquitetural, encontrar e reparar os gargalos de desempenho em hardware e software, assim como localizar os problemas de desempenho evitáveis. Conforme a indústria de computadores progride, a capacidade de analisar o desempenho de uma microarquitetura e fazer mudanças na microarquitetura com base naquela análise torna-se mais complexa e importante.

Além de prover a melhor plataforma possível, o melhor desempenho é freqüentemente conseguido pela sintonia do aplicativo para operar o seu melhor nesta plataforma. Existe um investimento significativo na identificação dos gargalos de desempenho, descobrir como evitá-los através de uma melhor geração de código, e confirmar os aperfeiçoamentos de desempenho. Os monitores de desempenho são um elemento chave nesta análise. O monitoramento de desempenho provê um maior volume de dados de desempenho do que as simulações pré-silício, e tem sido utilizado para ajustar os projetos microarquiteturais para aperfeiçoar o desempenho em áreas tais como a transferência de armazenamento. Sabendo apenas quão freqüentemente um problema de desempenho surge e quanto ganho seria obtido do aperfeiçoamento daquela parte da microarquitetura é essencial na motivação de mudanças do silício.

No passado, o monitoramento de desempenho de máquinas de execução seriais era relativamente direto, já que rastrear os gargalos de desempenho seriais é muito mais fácil do que detectar as limitações de desempenho durante uma execução paralela, fora de ordem. Uma análise de

desempenho típica decompõe o CPI (clocks por instrução) da carga de trabalho em componentes individuais como segue: 1) contar os eventos de desempenho em hardware, 2) estimar a contribuição relativa de cada evento para o percurso crítico do programa, e 3) combinar os componentes individuais que contribuem para os gargalos de desempenho da carga de trabalho em uma avaria total. A estimativa de custos por instância para uma única causa microarquitetural é difícil para uma máquina altamente especulativa, fora de ordem, onde existe um paralelismo superescalar e encadeado para cobrir uma fração significativa de muitos custos de parada. Até o momento, métodos ad hoc tem sido utilizados para estimar o impacto por instância de eventos, e a precisão e variação destas estimativas eram freqüentemente desconhecidas.

Por exemplo, a figura 1 ilustra um exemplo das instruções de busca, execução e retirada de instruções 101-107 em uma máquina de problema único. A instrução 102 tem uma má predição de ramificação 110, o que retarda a busca de instrução 103 e descarta a retirada da instrução 103 significativamente após a instrução 102. A instrução 104 tem uma perda de cache de primeiro nível 120 o que descarta adicionalmente a retirada da instrução 105. Mas o descarte de retirada da instrução 104 é tolhido pela perda de cache de segundo nível 130 da instrução 105, o que tem uma latência tão longa que a má predição de ramificação 135 na instrução 106 não tem nenhum impacto sobre o seu tempo de retirada. Como enumerado pela figura 1, existem complexidades intrincadas na medição de descarte de retirada, mesmo em uma máquina de problema único, para não dizer o monitoramento de desempenho compreensivo em um processado que é capaz de uma execução paralela altamente especulativa fora de ordem.

BREVE DESCRIÇÃO DOS DESENHOS

A presente invenção está ilustrada por meio de exemplo e não pretende ser limitada pelas figuras dos desenhos acompanhantes.

A figura 1 ilustra uma modalidade de busca, execução e retirada para uma pluralidade de operações em uma máquina de problema único.

a figura 2 ilustra uma modalidade de um processador que inclui

um primeiro módulo de monitoramento de desempenho e um segundo módulo de sintonia microarquitetural.

a figura 3 ilustra uma modalidade específica da figura 2.

a figura 4 ilustra uma modalidade de um processador que inclui um módulo para recompilar estaticamente ou dinamicamente o software.

a figura 5 ilustra uma modalidade de um sistema que inclui processador que tem um módulo para monitorar o desempenho e sintonizar a microarquitetura do processador.

a figura 6a ilustra uma modalidade de um fluxograma para monitorar o desempenho e sintonizar um microprocessador com base no desempenho.

a figura 6b ilustra uma modalidade específica da figura 6a.

a figura 6c ilustra outra modalidade para monitorar o desempenho e sintonizar um microprocessador.

a figura 7 ilustra uma modalidade para medir o descarte de retirada quando da ocorrência de um evento específico.

DESCRIÇÃO DETALHADA

Na descrição seguinte, numerosos detalhes específicos estão apresentados tais como arquiteturas específicas, características dentro destas arquiteturas, mecanismo de sintonia, e configurações de sistema de modo a prover uma compreensão completa da presente invenção. Ficará aparente, no entanto, para alguém versado na técnica que estes detalhes específicos não precisam ser empregados para praticar a presente invenção. Em outros casos, componentes ou métodos bem-conhecidos, tais como os projetos de lógica, os compiladores de software, as técnicas de reconfiguração de software, e as técnicas de descaracterização de processador bem-conhecidos, não foram descritos em detalhes de modo a evitar um obscurecimento desnecessário da presente invenção.

MONITORAMENTO DE DESEMPENHO

A figura 2 ilustra uma modalidade de um processador 205 que tem um módulo de monitoramento de desempenho 210 e um módulo de sintonia 215. O processador 205 pode ser qualquer elemento para executar um

código e/ou operar sobre dados. Como um exemplo específico, o processador 205 é capaz de execução paralela. Em outra modalidade, o processador 205 é capaz de uma execução fora de ordem. O processador 205 pode também implementar uma predicação de ramificação e uma execução especulativa, assim como outras unidades e métodos de processamento conhecidos.

Outras unidades de processamento ilustradas no processador 250 incluem: um subsistema de memória 220, uma interface 225, uma máquina fora de ordem 230, e unidades de execução 235. Cada um destes módulos, unidades, ou blocos funcionais pode prover a funcionalidade acima mencionada para o processador 205. Em uma modalidade, o subsistema de memória inclui um cache de nível mais alto e uma interface de barramento para interfacear com os dispositivos externos, a interface 225 inclui uma lógica de especulação e uma lógica de busca, a máquina fora de ordem 230 inclui uma lógica de programação para reordenar as instruções, e unidades de execução 235 incluem unidades de ponto flutuante e de execução de inteiros que executam em serial e em paralelo.

O módulo 210 e o módulo 215 podem ser implementados em hardware, software, firmware, ou qualquer sua combinação. Comumente, os limites de módulo variam e funções são implementadas juntas, assim como separadamente em diferentes modalidades. Em um exemplo, o monitoramento de desempenho e a sintonia são implementados em um único módulo. Na modalidade apresentada na figura 2, o módulo 210 e o módulo 215 estão separadamente mostrados; no entanto, o módulo 210 e o módulo 215 podem ser executados em software pelas outras unidades 220-235 ilustradas.

O módulo 210 é para monitorar o desempenho do processador 205. Em uma modalidade, o monitoramento de desempenho é feito pela determinação e/ou derivação de custos de eventos por instância para um percurso crítico. Um percurso crítico inclui qualquer percurso ou sequência de ocorrências, tarefas, e/ou eventos que contribuiriam para o tempo que leva para completar uma operação, instrução, grupo de instruções, ou um programa se a latência de qualquer tal ocorrência, tarefa ou evento tivesse que

ser aumentada. Graficamente, um percurso crítico pode algumas vezes ser referido como um percurso através de um gráfico de dados, controle, e dependências de recursos em um programa que opera em uma máquina específica para o qual prolongamento de qualquer arco neste gráfico de dependência levaria a um aumento na latência de execução deste programa.

Portanto, a contribuição por instância de um evento/característica para um percurso crítico e, em outras palavras, a contribuição de eventos, tais como uma perda de cache de segundo nível, ou uma característica microarquitetural, tal como uma unidade de predição de ramificação, à latência experimentada no completamento de uma tarefa um programa. De fato, a contribuição de um evento ou característica pode variar significativamente através de domínios de aplicativos. Conseqüentemente, o custo/contribuição do evento ou da característica microarquitetural pode ser determinado para um aplicativo de nível de usuário específico, tal como um sistema de operação. O módulo 215 será discutido em mais detalhes com referência à figura 3.

Um evento inclui qualquer operação, ocorrência, ou ação em um processador que introduz uma latência. Uns poucos exemplos de eventos comuns em um microprocessador incluem: uma perda de cache de baixo nível, uma perda de cache secundária, uma perda de cache de alto nível, um acesso de cache, um snoop de cache, uma má predição de ramificação, uma busca da memória, um bloqueio na retirada, uma pré-busca de hardware, um armazenamento de interface, uma divisão de cache, um problema de transferência de armazenamento, uma parada de recurso, uma memória temporária, uma decodificação de instrução, uma tradução de endereço, um acesso a um armazenamento de tradução, uma execução de operando de inteiro, uma execução de operando de ponto flutuante, uma renomeação de um registro, uma programação de uma instrução, uma leitura de registro, e uma escrita de registro.

Uma característica microarquitetural inclui uma lógica, uma unidade funcional, um recurso, ou outra característica associada com um evento acima mencionado. Exemplos de características microarquiteturais inclu-

em: um cache, um cache de instrução, um cache de dados, uma rede alvo de ramificação, uma tabela de memória virtual, um arquivo de registro, uma tabela de tradução, um armazenamento de consulta, uma unidade de predição de ramificação, um pré-buscador de hardware, uma unidade de execução, uma máquina fora de ordem, uma unidade de alocador, uma lógica de renomeação de registro, uma unidade de interface de barramento, uma unidade de busca, uma unidade de decodificação, um registro de estado arquitetural, uma unidade de execução, uma unidade de execução de ponto flutuante, uma unidade de execução de inteiros, um ALU, e outras características comum de um microprocessador.

CLOCKS POR INSTRUÇÃO

Um dos indicadores primários de desempenho é o clocks por instrução (CPI). O CPI pode ser dividido em diversos componentes, de modo que uma indicação da fração de ciclos que pode ser atribuída a cada um de diversos fatores/eventos possa ser determinada. Estes fatores, como acima apresentado, podem incluir eventos, tais como uma latência introduzida por caches faltantes e indo para DRAM, penalidades de má predição de ramificação, retardos de encadeamento incorridos por mecanismos na retirada, isto é para bloqueios, e assim por diante. Outros exemplos de fatores incluem as características microarquiteturais que estão associadas com os eventos, tais como um cache que é perdido, uma perda em uma rede alvo de ramificação utilizada para predicação de ramificação, a utilização de interfaces de barramento para ir para DRAM, e a utilização de máquinas de estado para implementar os bloqueios.

Tipicamente, a contribuição relativa de um fator é determinada pela multiplicação do número de ocorrências do fator por seu efeito em ciclos, então dividindo pelo número total de ciclos. Apesar de uma tal divisão poder ser precisamente apresentada para uma máquina escalar, não encadeada, não especulativa, é difícil fornecer uma contagem de ciclos precisa para uma máquina superescalar, encadeada, fora de ordem, e altamente especulativa. Freqüentemente existe um paralelismo suficiente em cargas de trabalho que podem ser exploradas por uma tal máquina para ocultar pelo

menos uma porção da parada fazendo um trabalho útil. Como um resultado, o impacto local desta parada pode fazer uma contribuição muito menor para o percurso crítico total do programa do que o custo por instância teórico. Surpreendentemente, a parada local pode até ter um impacto positivo sobre o tempo de execução total do programa, se este retardo local levar a uma melhor programação total.

ANALISANDO AS CONTRIBUIÇÕES/CUSTOS POR INSTÂNCIA

Os custos de evento por instância, isto é uma contribuição de evento ou de características microarquiteturais para um percurso crítico, podem ser determinados em muitos diferentes modos, que incluem: (1) estimativas analíticas; (2) contagens de duração de monitores de desempenho; (3) descartes de retirada como medidos por monitores de desempenho de hardware e por simuladores; e (4) mudanças no tempo de execução total devido a mudanças no número de eventos como medidos por micropadrões de desempenho, simulações, e descaracterizações de silício.

ESTIMATIVAS ANALÍTICAS

Em uma primeira modalidade, um custo por instância, isto é uma contribuição de uma característica, é determinado teoricamente. A contribuição teórica pode incluir um conhecimento de operação empírico de uma característica ou uma ocorrência de um evento, assim como uma simulação de uma arquitetura. Isto é freqüentemente derivado de uma compreensão da microarquitetura, e tipicamente, focaliza no estágio de execução, ao invés de na retirada. A forma mais simples de estimativas analíticas caracteriza um custo de parada local, independentemente de como estas paradas podem ser cobertas através do paralelismo disponível da execução de outras operações (estágios de execução ou instruções) em paralelo.

CONTAGENS DE DURAÇÃO

Em outra modalidade, um monitor de desempenho determina uma contribuição de uma característica através de contagem de duração. Alguns eventos de monitor de desempenho são definidos para contar cada ciclo que alguma coisa de interesse está acontecendo. Isto gera uma contagem de duração ao invés de uma contagem de instâncias. Duas tais conta-

gens são os ciclos que uma máquina de estado está ativa, por exemplo um manipulador de page walk, uma máquina de estado de bloqueio, e ciclos que existem uma ou mais entradas em uma fila, por exemplo, a fila de barramento de faltas de cache destacadas. Estes exemplos medem o tempo em um estágio de execução, e não necessariamente medem um descarte de retirada, a menos que a execução seja na retirada, o que é o caso para a máquina de estado de bloqueio. Esta forma de caracterização é utilizável no campo de avaliação de custos específicos de padrão de desempenho.

DESCARTES DE RETIRADA

Os descartes de retirada são úteis na determinação da contribuição de eventos e características em uma escala local, assim como extrapolando esta medição para uma escala global. Um descarte de retirada ocorre quando uma operação não se afasta em um momento esperado ou durante um ciclo esperado. Por exemplo, para um par de instruções sequencial (ou microoperações), se a segunda instrução não se afasta logo que possível após a primeira (normalmente no mesmo ciclo, ou se os recursos de retirada são restritos, no próximo ciclo), então a retirada é considerada ser descartada. O descarte de retirada provê uma medição de retrovisão, "regional" (ao invés de puramente local) de contribuição para o percurso crítico. É de retrovisão no sentido em que o descarte de retirada está consciente da sobreposição de todas as operações as quais foram afastadas antes de algum ponto no tempo. Se duas operações com um custo de parada local de 50 começam separadas por um ciclo, o descarte de retirada para a segunda é no máximo um, ao invés de 50.

A medição real de descarte de retirada pode variar dependendo de quando o descarte é medido. Em uma instância, a medição é de uma ocorrência de um evento. Em outra modalidade, o descarte é de quando a instrução ou a operação deveria ter sido afastada. Em ainda outra modalidade, o descarte de retirada é medido simplesmente pela contagem do número de ocorrências de descartes de retirada, como abaixo discutido com referência ao descarte de retirada de operações sequenciais. Existem vários modos de medir/derivar uma contribuição por instância através de descarte de reti-

rada. Para ilustrar, dois métodos de descarte de retirada, operações sequenciais e identificação, estão abaixo discutidos.

Ambos os mecanismos permitem ao usuário criar um histograma da distribuição de descartes de retirada, operando repetidamente com diferentes valores de limite. O descarte de retirada de operações sequenciais permite a criação de um perfil de retardos de retirada para todas as operações no programa. Além disso, a identificação de descartes de retirada permite a criação de perfis de distribuição de retardo para eventos individuais/específicos, tais como a contribuição individual de más predições de ramificação.

Descarte de Retirada de Operações Sequenciais, isto é, qualificação de retirada lenta.

Para este mecanismo instâncias de operações sequenciais são contadas onde o retardo entre a retirada de operações consecutivas, ou microoperações, é maior do que um limite especificado pelo usuário. Conseqüentemente, o descarte para as operações consecutivas é medido e o número de descartes com uma latência acima de um limite predefinido é relatado.

Em uma modalidade, a qualificação de retirada lenta é medida utilizando um contador de uso específico, o qual conta os ciclos nos quais as instruções de uma cadeia não estão sendo afastadas. O contador é inicializado para um valor definido pelo usuário logo que uma primeira operação se afasta. Se o contador estoura negativamente ou positivamente, dependendo do projeto, para uma segunda instrução específica, esta segunda instrução é considerada ter uma retirada lenta, isto é um descarte de retirada.

Como um exemplo de um projeto que utiliza um contador regressivo, se um usuário deseja contar quantas retiradas de instruções são descartadas ao longo de 25 ciclos, então o contador é ajustado para um valor predefinido de 25. Se este estourar negativamente, a retirada de uma segunda instrução é considerada descartada. Em uma implementação de contador crescente o valor definido pelo usuário pode ser inicializado ou em 0 ou em um número negativo. Por exemplo, o contador é inicializado para 0

e conta até um valor limite de 25. Se o contador estourar então existe um descarte de retirada. Em uma alternativa, o contador crescente pode ser inicializado para -25 e contar até 0, o que simplifica a comparação lógica quando determinando um estouro de contador.

Identificação de Descarte de Retirada, isto é Perfilamento de Descarte de Retirada

Muito similar à qualificação de retirada lenta, a identificação de descarte de retirada qualifica as instruções ou a operação as quais tem descartes de retirada acima de algum limite. No entanto, neste mecanismo, a qualificação de retirada lenta é apenas uma de muitas outras qualificações sobre a instrução ou operação de interesse. Outras qualificações podem incluir os eventos específicos que ocorreram para aquela instrução ou operação, tal como uma perda de cache de segundo nível. Estas qualificações são logicamente combinadas, e a instrução ou operação é contada se esta atender ao critério de qualificação especificado. Note que os qualificadores/eventos podem ser logicamente operados ou combinados, o que pode ser definível pelo usuário em registros de estado de máquina especificados.

Em outra modalidade, uma operação é identificada com base na exclusão de um evento ou eventos específicos. Como acima apresentado uma execução paralela pode mascarar o efeito real de um evento específico. Como um exemplo específico, uma perda de um cache de terceiro nível pode tolher o efeito de uma perda para o cache de segundo nível. Para isolar o efeito de uma perda para um cache de segundo nível, uma operação específica pode ser identificada, se esta resultar em uma perda para o cache de segundo nível e não perder o cache de terceiro nível. Em outras palavras, a medição de operações que resultam em perdas de cache de terceiro nível são excluídas da medição. Portanto, esta identificação inclui selecionar uma operação quando de uma ocorrência de um evento específico e a não ocorrência de pelo menos um segundo evento.

Observando brevemente a figura 7, uma modalidade para medir o descarte de retirada utilizando um mecanismo de identificação está ilustrada. No fluxo 705, uma operação é identificada, quando da ocorrência de um

evento específico e/ou a exclusão de um evento específico. A operação deve ser executada em um processador capaz de uma execução paralela. No entanto, o processador pode também ser capaz de uma execução serial, uma execução especulativa, e uma execução fora de ordem.

O evento específico pode ser qualquer evento em um microprocessador como acima discutido. Em uma modalidade, o evento é uma amostragem baseada em evento precisa (PEBS) no evento de retirada. Em PEBS, uma operação (microoperação ou instrução) é marcada (identificada) como tendo experimentado um evento de interesse, tal como uma perda de cache. Conforme esta operação se retira, a lógica de retirada nota que esta está identificada e executa ações especiais. O endereço da instrução e o estado arquitetural tal como os identificadores e os registros arquiteturais são salvos em um armazenamento de memória. Neste caso, a latência de retirada é gravada juntamente com as outras informações. A execução de programa pode continuar seguindo estas ações especiais até que o armazenamento de memória no qual tais informações são gravadas fique (quase) cheio. Quando este fica cheio (ou acima de uma marca d'água especificada pelo usuário), uma interrupção de monitoramento de desempenho é causada, sinalizando que o usuário deveria ler o armazenamento de memória. As ações executadas quando de uma PEBS podem ser gerenciadas ou por uma máquina de estado finito no hardware, através de uma instrução em microcódigo, ou uma sua combinação.

Uns poucos exemplos específicos de um evento que resulta em identificação de uma operação incluem: uma perda de cache, um acesso de cache, um snoop de cache, uma má predição de ramificação, um bloqueio quando da retirada, uma pré-busca de hardware, uma carga, um armazenamento, uma memória temporária, e um acesso a um armazenamento de tradução. A identificação inclui selecionar uma operação para medição. Note que estes eventos podem também ser destinados para exclusão, já que a operação pode não ser identificada se um destes eventos também ocorrer em conjunto com o evento específico como acima discutido.

Após identificar ou selecionar uma operação, no fluxo 710, um

descarte de retirada para a operação é determinado. Como acima mencionado, a determinação de um descarte de retirada pode ser a medição real do retardo na retirada, assim como simplesmente contar a operação como uma retirada retardada devido ao evento específico.

Na modalidade onde a medição real do descarte de retirada é o objetivo, o módulo de limite em um contador, tal como o contador utilizado para a qualificação de retirada lenta, é ajustado para 0, de modo que o valor final quando da retirada é um número positivo igual ao descarte de retirada. Em um caso, o primeiro contador é inicializado e o descarte de retirada é determinado com base na inicialização do primeiro contador e na utilização de um registro de armazenamento. Nesta caso, o estado do primeiro contador é copiado para outro registro de estado de máquina. Quando da retirada, o registro de armazenamento é congelado e não é atualizado. Assim o registro de armazenamento fica estável até que o software o lê.

Note que a medição do descarte foi referida em referência à medição e à retirada. No entanto, o descarte pode ser medido em outros pontos de estrangulamento em ordem em uma máquina fora de ordem, tais como as operações de buscar, de decodificar, de emitir, de alocação de memória em um armazenamento de ordenação de memória, e as operações de visibilidade global de memória.

TEMPO DE EXECUÇÃO TOTAL

Os custos de parada local podem ser parcialmente ou totalmente cobertos por outro trabalho o qual é feito em paralelo. Os descartes de retirada que capturam o retardo regional podem também ser parcialmente ou totalmente cobertos por trabalho, ou outras paradas, que ainda estão ocorrendo no momento no qual o descarte de retirada é medido. Um modo no qual o descarte de retirada é coberto está ilustrado na figura 1, como acima discutido. A medição de contribuição final que uma dada parada de operação faz ao percurso crítico de um programa é a mudança na latência de execução que ocorre devido àquela causa de parada.

Uma indicação da contribuição incremental média ao percurso crítico global é medir a execução inteira de um programa ou rastreio longo,

isto é um monitoramento de execução de rastreio longo. Esta proposta cobre as contribuições ao percurso crítico que ocorre em qualquer lugar no encadeamento, e leva em conta o fato de que os retardos locais podem ser cobertos por outro paralelismo. A contribuição incremental é derivada pela mudança do número de instâncias de um evento, o que muda o tempo de execução, e computando a mudança no tempo de execução dividido pela mudança no número de eventos. Por exemplo, se o aumento do tamanho de cache diminui o número de perdas de cache de 100 para 90, e diminui o tempo de execução de 2000 para 1600, então a contribuição incremental é (2000 - 1600)/(100 - 90) = 40 ciclos por perda.

A implementação desta técnica pode ser feita em uma pluralidade de modos. Primeiro, duas versões de um micropadrão de desempenho podem ser construídas, uma com eventos e a outra sem. Segundo, uma configuração de simulador pode ser mudada para introduzir ou eliminar eventos. A simulação é operada para um ou mais programas em ambas as configurações, e tanto o número de eventos quanto o tempo de execução total são gravados para cada caso. Finalmente, alguns produtos suportam descaracterizações de silício, tais como o encolhimento do tamanho de uma rede alvo de ramificação ou a mudança das políticas. Isto pode ser feito para afetar as taxas de predição de ramificação, por exemplo.

Como acima mencionado, a determinação de uma contribuição de uma características microarquitetural, isto é um custo de evento, pode ser feita através de (1) estimativas analíticas; (2) contagens de duração de monitores de desempenho; (3) descartes de retirada como medidos por monitores de desempenho de hardware e por simuladores; e (4) tempo de execução total como medido por micropadrões de desempenho, simulações, e descaracterizações de silício. No entanto, o monitoramento de desempenho e a determinação de contribuição para um percurso crítico não estão limitados a uma implementação ortogonal de um dos métodos acima mencionados, mas ao contrário qualquer combinação pode ser utilizada para analisar a contribuição de um evento de uma característica de silício para o percurso crítico.

UM EXEMPLO DE CUSTO POR INSTÂNCIA PARA EVENTOS ESPECÍFICOS

Para avaliar o custo por instância dos vários eventos, as técnicas descritas na seção de análise de contribuições por instância foram empregadas. Existem, é claro, numerosos contribuidores para a divisão de CPI compreensiva de um rastreio. Quatro contribuidores significativos foram escolhidos para demonstrar a eficiência de cada técnica descrita. Para cada evento, no entanto, não é sempre possível ou conveniente utilizar todas as técnicas. Por exemplo, as contagens de duração de monitoramento de desempenho podem não estar disponíveis para o evento sob consideração. Do mesmo modo, perturbar a execução pelo ajuste de um tamanho ou política no simulador pode não afetar o número de ocorrências de um evento ou mudar o tempo de operação em um rastreio específico. A Tabela 1 mostra o resumo dos custos estimados para cada uma destas quatro causas com base na perturbação de execução simulada, e provê uma indicação de variância no impacto que está baseada em resultados de simulação totais.

Causa de Parada	Valor (mediana)	Desvio Padrão	Dentro de 1 σ	Método de Medição
Má Predição de Ramificação	25	35	85%	Desabilitou o indireto Preditor de ramificação
Perda de Cache de Dados de L1	9	6	92%	Dobrou o tamanho de cache de L1
Perda de Cache de Dados de L2	257	158	74%	Dobrou o tamanho de cache de L2

Tabela 1: Custos por instância empíricos

MÁS PREDIÇÕES DE RAMIFICAÇÃO

As más predições de ramificação são uma causa comum de desaceleração de aplicativos. Estas forçam a reinicialização de encadeamento de processador e jogam fora o trabalho especulativo. Os preditores de ramificação tornaram-se cada vez mais precisos ao longo do tempo. Apesar de

tudo, com encadeamentos mais profundos e mais largos, as más predições podem causar uma considerável perda de oportunidade para completar o trabalho útil.

Analítico 1	Execução de Simulador	Descarte de retirada de HW	Descarte de retirada de simulador	Micropadrão de desempenho
31	25	Picos em 36, 41, 47	36	34

Tabela 2: má predição de ramificação por custo de evento de instância

A medição analítica de custo de má predição de ramificação é o número de ciclos de retardo (31) de onde uma má predição de ramificação é normalmente detectada, na execução, para trás onde as instruções são normalmente buscadas, do cache de rastreio. A perspectiva analítica mede o retardo real incorrido na interface da máquina. Este retardo pode ser aumentado se não existir nenhum retardo na avaliação da condição de ramificação, ou devido à contenção de recursos, ou devido a uma dependência de dados não resolvida, especialmente se a dependência for em uma carga que sofreu uma perda de cache. É por três razões que o retardo de descarte de retirada pode estar nos trinta até os quarenta, como visto nos micropadrões de desempenho, nos descartes de retirada de HW, e nos descartes de retirada simulados. Três valores estão mostrados para os descartes de retirada de HW na Tabela 2. O micropadrão de desempenho aqui utilizado tinha um corpo de laço com uma ramificação condicional e nenhuma referência de memória. 28% mais ramificações tinha um retardo de 36 do que em 35,27% mais ramificações tinham um retardo de 40 versus 39, e 43% mais ramificações tinham um retardo de 41 versus 40 ciclos. Os micropadrões de desempenho correspondia proximamente ao modelo analítico já que estes contém pouco trabalho paralelo e não requerem uma limpeza complexa.

No entanto, como mostrado na figura 1, onde a instrução 106 tem uma má predição de ramificação, um retardo na interface pode não ter nenhum impacto se existiu um descarte de retirada anterior na traseira da máquina. Também, uma perda de cache posterior pode afogar a contribuição da ramificação para o percurso crítico com um retardo muito maior. Esta

é uma razão porque a contribuição média para o percurso crítico total é muito mais baixa do que o descarte de retirada. A contribuição total simulada para o percurso crítico foi derivada da desabilitação do preditor de ramificação indireta, de modo que este pode somente predizer o último alvo. Mais ainda, em aplicações reais, o código fora de percurso pode freqüentemente executar pré-buscas de dados úteis e consultas de DTLB o que diminui o impacto das más predições. Finalmente, a sobreposição do processamento de uma má predição com o processamento de uma segunda pode reduzir a contribuição média para o percurso crítico total.

Desta discussão, fica claro que a contribuição média real para o percurso crítico pode ser altamente dependente do contexto, e os descartes de retirada podem superestimar o custo por instância. Um fator de escalagem, tal como ~ 70% pode ser aplicado nos descartes de retirada medidos em HW para derivar o custo por instância mediano. Note que este custo de evento pode ser altamente dependente da microarquitetura específica e mesmo da implementação dentro da mesma família microarquitetural.

PERDA DE CACHE DE PRIMEIRO NÍVEL (L1)

As perdas de cache de primeiro nível são uma ocorrência comum. Os processadores fora de ordem são projetados para encontrar um trabalho independente no fluxo de instruções para manter o processador ocupado enquanto fornecendo a perda para o cache de segundo nível. Como um resultado, somente uma pequena fração do custo de perda de L1 local (por exemplo descarte de retirada) contribui para o percurso crítico total.

Analítico 1	Execução de Simulador	Descarte de retirada de simulador	Micropadrão de desempenho
18	9	18,3	26

Tabela 3: Perda de cache de primeiro nível por custo de evento de instância

O modelo analítico aqui descreve o excesso da perda de L1 acima do custo de carregar para utilizar normal. O micropadrão de desempenho para este evento consiste em um laço de perseguição de ponteiro o qual encontra uma distribuição uniforme de excessos de 18 ciclos. Um fator de

escalagem de ~ 50% pode ser aplicado no descarte de retirada de hardware para todos os eventos de perda de L1 para chegar ao custo por instância médio.

PERDA DE CACHE DE SEGUNDO NÍVEL (L2)

As perdas de cache de segundo nível podem ser enviadas para um cache de nível mais alto ou para um controlador de memória/DRAM. Os processadores fora de ordem são projetados para encontrar as perdas de cache de L2 independentes para encadear o processamento destas longas transações.

Analítico 1	execução de simulador	Descarte de retirada de simulador	Micropadrão de desempenho
306	256	281	300

Tabela 4: Perda de cache de segundo nível por custo de evento de instância

A medida analítica de perdas de cache é de 306 clocks com acertos de página de DRAM de fluxo contínuo. Isto é computado de uma DRAM de 90 ns com um FSB de 800 MHz em um processador de 3,4 GHz. O micropadrão de desenvolvimento, o qual consiste em um simples código de perseguição de ponteiro, correlaciona bem com o modelo analítico. Este núcleo foi projetado para acertar no DTLB mas não realiza nenhum benefício do pré-buscador de hardware. Assim existe pouco trabalho paralelo para fazer que pudesse esconder parte da latência, e pouco trabalho independente para fazer que impedisse que cada carga fosse prontamente enviada para a DRAM. Os descartes de retirada e as execuções simuladas todos resultam em um custo por instância que é menor do que o valor analítico. De fato, as execuções simuladas mostram uma ampla variância no custo por instância através dos rastreios, tanto mais curta quanto mais longa do que o valor analítico. Claramente, existe o benefício de acessos de DRAM sobrepostos na extremidade de latência curta do espectro. As latências por instância mais longas podem ocorrer em diversos modos, incluindo as limitações de profundidade de fila de solicitação de memória de processador e perdas de largura de banda de barramento.

O pré-buscador de hardware desempenha um papel muito im-

portante nesta latência. Apesar de apropriadamente controlado em aceleração, este tem a capacidade de inserir numerosas solicitações no sistema de memória por meio disto aumentando a latência de cargas de demanda subseqüentes. Na outra extremidade do espectro, o pré-buscador algumas vezes começa a pré-buscar tarde demais para evitar a perda de uma carga mais recente, mas cedo o bastante para ter feito com que os dados estivessem a caminho da DRAM no momento da carga mais recente. Isto resulta em custos de perda efetiva por instância mais curtos. Em geral, o custo por instância mediano é muito similar à medição de descarte de retirada de HW.

A variação de custo varia significativamente através dos domínios de aplicativo, como acima sugerido. Portanto, potencialmente tendo um mecanismo no campo para medir o custo para um dado aplicativo pode ser extremamente útil na determinação de contribuição de características específicas. À luz desta variação, uma microarquitetura pode ser sintonizada em uma base por aplicativo.

SINTONIZANDO A MICROARQUITETURA

A microarquitetura pode ser sintonizada tal como durante a medição de descarte de retirada e a medição de tempo de execução total para determinar o custo de evento por instância. No entanto, a microarquitetura pode também ser sintonizada em resposta ao custo de evento por instância. A sintonia de uma característica microarquitetural ou de uma microarquitetura inclui mudar o tamanho, habilitar ou desabilitar a lógica, uma característica, e/ou uma unidade dentro da microarquitetura, assim como mudando as políticas dentro da microarquitetura.

Em uma modalidade, a sintonia é feita com base em uma contribuição, isto é uma contribuição por instância, de uma característica microarquitetural. Como um primeiro exemplo, o tamanho da característica é alterado, a característica é habilitada, a característica é desabilitada, ou as políticas associadas com a característica são alteradas com base em qual ação reduz a latência em um percurso crítico. Como outro exemplo, outras considerações tais como a energia podem ser utilizadas para sintonizar a microarquitetura. Neste exemplo, pode ser determinado que desabilitar a caracte-

rística aumenta a latência por uma quantidade significativa. No entanto, com base na determinação de que o benefício de desempenho da característica é significativo e que a desabilitação da característica economizaria uma energia significativa, a característica é sintonizada, por exemplo desabilitada.

Como um exemplo empírico,foi notado nas arquiteturas anteriores que em diversas macro cargas de trabalho um número significativo de conflitos de serrilha foi notado. Um destes exemplos onde os conflitos de serrilha ocorreram foi entre múltiplas cadeias acessando as mesmas linhas de cache.

Uma cadeia de software é pelo menos uma parte de um programa que é operável para ser executado independentemente de outra cadeia. Alguns microprocessadores até suportam um multiencadeamento em hardware, onde o processador tem pelo menos uma pluralidade de conjuntos completos e independentes de registros de estado de arquitetura para programar independentemente a execução de múltiplas cadeias de software. No entanto, estas cadeias de hardware compartilham alguns recursos, tais como os caches. Anteriormente, os acessos à mesma linha no cache por múltiplas cadeias resultavam no deslocamento da linha de cache e a redução de localidade. Portanto, o endereço de partida de memória de dados para as cadeias, era ajustado para diferentes valores para evitar o deslocamento de linhas em cache entre as cadeias.

Observando a figura 3, uma modalidade específica do módulo 215 no processador 205 está ilustrada. O módulo 215 é para sintonizar uma característica microarquitetural para um aplicativo de nível de usuário, com base em pelo menos a contribuição da característica microarquitetural para o percurso crítico.

Um exemplo muito específico deste tipo de sintonia inclui, monitorar o desempenho de um pré-buscador de hardware durante os aplicativos, ou fases de um aplicativo tal como o coletamento de lixo. O coletamento de lixo é operado com o pré-buscador de hardware habilitado, então desabilitado, e é descoberto que em algumas instâncias o coletamento de lixo executa melhor sem o pré-buscador de hardware. Conseqüentemente, a microarqui-

tetura pode ser sintonizada e o pré-buscador de hardware desabilitado quando da execução de um aplicativo de coletamento de lixo.

Outros exemplos de mudança de políticas com base em análise de desempenho incluem a agressividade de pré-busca, a alocação relativa de recursos para diferentes cadeias em uma máquina de encadeamento simultâneo, page walks especulativos, atualizações especulativas para TLBs, e a seleção entre os mecanismos preditivos para ramificações e dependências de memória.

A figura 3 ilustra as características microarquiteturais: o subsistema de memória 220, o cache 350, a interface 225, a predição de ramificação 355, o buscador 360, as unidades de execução 235, o cache 350, as unidades de execução 355, a máquina fora de ordem 230 e a retirada 365. Outros Exemplos de características microarquiteturais incluem: um cache, um cache de instrução, um cache de dados, uma rede alvo de ramificação, uma tabela de memória virtual, um arquivo de registro, uma tabela de tradução, um armazenamento de consulta, uma unidade de predição de ramificação, um preditor de ramificação indireta, um pré-buscador de hardware, uma unidade de execução, uma máquina fora de ordem, uma unidade de alocador, uma lógica de renomeação de registro, uma unidade de interface de barramento, uma unidade de busca, uma unidade de decodificação, um registro de estado arquitetural, uma unidade de execução, uma unidade de execução de ponto flutuante, uma unidade de execução de inteiros, um ALU, e outras características comum de um microprocessador.

Como acima mencionado, a sintonia de uma característica microarquitetural pode incluir habilitar ou desabilitar a característica microarquitetural. Como no exemplo com o pré-buscador de hardware acima, o pré-buscador pode ser desabilitado se a contribuição for determinada ser melhorada, isto é melhor, quando a característica é desabilitada durante programas de software específicos.

Um modo de determinar a contribuição de uma característica microarquitetural para o percurso crítico para um aplicativo de nível de usuário é executar o aplicativo de nível de usuário com a característica microar-

quitetural habilitada. Então, executar o aplicativo de nível de usuário com a característica microarquitetural desabilitada. Finalmente, a contribuição da característica microarquitetural para o percurso crítico para o aplicativo de nível de usuário é determinada com base na comparação da execução do aplicativo de nível de usuário com a característica habilitada com a execução do aplicativo de nível de usuário com a característica desabilitada. Simplesmente, medindo o tempo de execução total cada vez que o aplicativo de nível de usuário é executada, é determinado qual tempo de execução total é melhor; o tempo total com a característica habilitada ou desabilitada.

Como um exemplo especifico o módulo 215 inclui o registro de descaracterização 305. O registro de descaracterização 305 inclui uma pluralidade de campos, tais como os campos 310-335. Os campos podem ser bits individuais, ou cada campo pode ter uma pluralidade de bits. Além disso, cada campo é operável para sintonizar uma característica microarquitetural. Em outras palavras, o campo está associado com uma característica microarquitetural, como o campo 310 está com a predição de ramificação 355, o campo 315 está com o buscador 360, o campo 320 está com o cache 350, o campo 325 está com a lógica de retirada 365, o campo 330 está com as unidades de execução 355, e o campo 335 está com o cache 350. Quando um dos campos, tal como o campo 310, é ajustado, este desabilita a predição de ramificação 355.

Outro módulo, tal como um programa de software, associado com o módulo 215, incorporado no módulo 215, ou parte do módulo 215 pode ajustar um campo, tal como o campo 310, se a contribuição de desempenho da característica para o percurso crítico, quando desabilitada, é melhorada como acima discutido. Como acima notado, o módulo 215 pode ser um hardware, um software ou uma sua combinação, assim como associado com um módulo parcialmente sobreposto 210. Por exemplo, como parte da funcionalidade do módulo 210, para determinar uma contribuição da predição de ramificação 355 durante a execução de um programa de nível de usuário, o registro 305, ilustrado no módulo 215, pode ser utilizado para sintonizar ou desabilitar as características do processador 205, tal como a predição de

ramificação 355.

Em outra modalidade, a descaracterização, isto é a sintonia, inclui mudar o tamanho, ou fisicamente ou virtualmente, de uma característica. Na alternativa ao exemplo acima, se a contribuição da predição de ramificação 355 mostrasse melhorar a execução de um aplicativo de nível de usuário, então o tamanho da predição de ramificação 355 pode ser aumentado/diminuído conseqüentemente através do campo 310. O exemplo abaixo ilustra a capacidade de sintonizar o processador para descobrir a contribuição de uma característica ou evento, tal como uma perda de cache, pela sintonia do tamanho de caches.

SOFTWARE DE SINTONIA

Observando a figura 4 uma modalidade de um software de desempenho de monitoramento e de sintonia de processador está ilustrado. O processador 405, similarmente ao processador 205 mostrado nas figuras 2 e 3, pode ter qualquer lógica conhecida associada com um processador. Como ilustrado, o processador 405 inclui unidades/características: um subsistema de memória 420, uma interface 425, uma máquina fora de ordem 430, e unidades de execução 435. Dentro de cada um destes blocos funcionais numerosas outras características microarquiteturais podem existir, tal como um cache de segundo nível 421, uma unidade de busca/decodificação 427, uma predição de ramificação 426, uma retirada 431, um cache de primeiro nível 436, e unidades de execução 437.

Como acima, o módulo 410 determina um custo de evento por instância em um percurso crítico para execução de um programa de software. Exemplos de derivação de um custo de evento por instância acima incluem as contagens de duração, a medição de descarte de retirada, e a medição de execução de rastreio longo. Note mais uma vez, que o módulo 410 e o módulo 415 podem ter limites borrados, já que sua funcionalidade, hardware, software, ou uma combinação de hardware e software podem sobrepor.

Em contraste com a figura 3, onde o módulo 415 sintonizava a microarquitetura interfaceando com as características, o módulo 415 deve sintonizar um programa de software com base no custo de evento por ins-

tância no percurso crítico. O módulo 415 pode incluir qualquer hardware, software, ou combinação para compilar e/ou interpretar um código para executar no processador 405. Em uma modalidade, o módulo 415 recompila o código que é executado em uma operação subseqüente do programa para utilizar as características microarquiteturais acima mencionadas mais ou menos freqüentemente se comparado com o código originalmente compilado baseado em um custo de evento por instância determinado. Em outra modalidade, o módulo 415 compila o código diferentemente para o restante da mesma operação do programa, isto é, uma compilação dinâmica ou recompilação é utilizada para aperfeiçoar o tempo de execução em uma carga de trabalho e uma plataforma específicas.

Como acima apresentado, além de ser capaz de sintonizar a microarquitetura, um melhor desempenho pode ser conseguido sintonizando o aplicativo para operar o seu melhor naquela plataforma. O software de sintonia inclui um código de otimização. Um exemplo de sintonizar um aplicativo é a recompilação do programa de software. Sintonizar o software pode também incluir otimizar um software/código para bloquear estruturas de dados para caberem dentro de caches, retransmitir o código para aproveitar-se de condições de predição de ramificação padrão que não requerem a utilização de recursos de tabela de preditor de ramificação, emitir um código em diferentes endereços de instrução de modo a evitar certas serrilhas e condições de conflito que podem levar a problemas com o gerenciamento de localidade em estruturas de predição de ramificação e de busca de código, retransmitir dados em uma memória dinamicamente alocada ou na pilha (que inclui o alinhamento de pilha) para evitar as penalidades incorridas pela extensão de uma linha de cache, e ajustar a granularidade e o alinhamento de acessos de modo a evitar os problemas de transferência de armazenamento.

Como um exemplo específico de software de sintonia, o software 450 é executado com/no processador 405. O módulo 410 determina um custo de evento por instância, tal como o custo de má predição de uma ramificação na lógica de predição de ramificação 426. Com base nesta análise, o

módulo 415 retransmite o software 450 para o software 460, o qual é o mesmo aplicativo de nível de usuário retransmitido para executar diferentemente no processador 405. Neste exemplo, o software 460 é retransmitido para melhor aproveitar-se das condições de predição de ramificação padrão. Portanto, o software 460 é recompilado para utilizar a predição de ramificação 426 diferentemente. Outros exemplos podem incluir executar as instruções em código para desabilitar a lógica de predição de ramificação 426 e mudar as sugestões de software utilizadas pela lógica de predição de ramificação 426.

UM SISTEMA PARA MONITORAMENTO DE DESEMPENHO

Referindo a seguir à figura 5, um sistema para utilizar o monitoramento de desempenho está ilustrado. O processador 505 está acoplado no hub de controlador 550 e o hub de controlador 550 está acoplado na memória 560. O hub de controlador 550 pode ser um hub de controlador de memória ou outra porção de um dispositivo de conjunto de chip. Em alguns casos, o hub de controlador 550 tem um controlador de vídeo integrado, tal como o controlador de vídeo 555. No entanto, o controlador de vídeo 555 pode também estar em um dispositivo gráfico acoplado no hub de controlador 550. Note que outros componentes, interconexões, dispositivos, e circuitos podem estar presentes em cada um dos dispositivos mostrados.

O processador 505 inclui o módulo 510. O módulo 510 é para determinar uma contribuição de evento por instância durante a execução de um programa de software, sintonizar uma configuração arquitetural de microprocessador 505 com base na contribuição de evento por instância, armazenar a configuração arquitetural, e ressintonizar a configuração arquitetural com base na configuração arquitetural armazenada, quando da execução subseqüente do programa de software.

Como um exemplo específico, o módulo 510, que utiliza o módulo de contribuição 511, determina uma contribuição de evento durante a execução de um programa de software, tal como um sistema de operação. Outros exemplos de programas de software incluem um aplicativo de convidado, um aplicativo de sistema de operação, um padrão de desempenho, um

micropadrão de desempenho, um operador, e um aplicativo incorporado. Para este exemplo, assumindo que uma contribuição de evento, tal como uma falta para um cache de primeiro nível 536 não afetou a execução significativamente, o cache 536 pode ser reduzido em tamanho para economizar energia sem afetar o tempo de execução no percurso crítico. Portanto, o módulo de sintonia 512 sintoniza a arquitetura do processador 505 pela redução do cache de primeiro nível 536 em tamanho. A sintonia pode ser feita, como acima mencionado, com um registro que tem campos associados com as diferentes características no processador 505. Onde um registro é utilizado, o armazenamento da configuração arquitetural inclui armazenar os valores de registro no armazenamento 513, o qual é simplesmente outro registro ou dispositivo de memória, tal como a memória 560. Quando da subseqüente execução do programa de software, a etapa de monitoramento de desempenho não precisa ser repetida, e a configuração previamente armazenada pode ser carregada. Conseqüentemente, a arquitetura é ressintonizada para o programa de software com base na configuração armazenada.

UM MÉTODO PARA MONITORAMENTO DE DESEMPENHO

A figura 6a ilustra uma modalidade de um fluxograma para monitorar o desempenho e sintonizar um microprocessador. No fluxo 605 um primeiro programa de software é executado utilizando um microprocessador. Em uma modalidade, o microprocessador é capaz de uma execução paralela fora de ordem. A seguir, no fluxo 610, um custo de evento para um percurso crítico associado com a execução do primeiro programa de software é determinado.

Observando a figura 6b' exemplos de determinação de um custo de evento e de sintonia do microprocessador são ilustrado. Um custo de evento pode ser determinado por análise analítica, contagens de duração, como mostrado no fluxo 611, descartes de retirada, como no fluxo 612, e/ou tempo de execução total, como mostrado no fluxo 613. Note que qualquer combinação destes métodos pode ser utilizada para determinar o custo de um evento.

Uns poucos exemplos de eventos comuns em um microproces-

sador incluem: uma perda de cache de baixo nível, uma perda de cache secundária, uma perda de cache de alto nível, um acesso de cache, um snoop de cache, uma má predição de ramificação, uma busca da memória, um bloqueio na retirada, uma pré-busca de hardware, uma carga, um armazenamento, uma memória temporária, uma decodificação de instrução, uma tradução de endereço, um acesso a um armazenamento de tradução, uma execução de operando de inteiro, uma execução de operando de ponto flutuante, uma renomeação de um registro, uma programação de uma instrução, uma leitura de registro, e uma escrita de registro.

Retornando à figura 6a, no fluxo 615, o microprocessador é sintonizado com base no custo de evento para o percurso crítico associado com a execução do primeiro programa de software. A sintonia inclui qualquer mudança na microarquitetura para melhorar o desempenho e/ou o tempo de execução. Referindo de volta à figura 6b, um exemplo de sintonia inclui habilitar ou desabilitar uma característica microarquitetural, como no fluxo 617. Uns poucos exemplos ilustrativos de características incluem, um cache, uma tabela de tradução, um armazenamento de consulta de tradução (TLB), uma unidade de predição de ramificação, um pré-buscador de hardware, uma unidade de execução e uma máquina fora de ordem. Outro exemplo inclui mudar o tamanho ou a frequência de utilização de uma característica microarquitetural, tal como no fluxo 616. Em ainda outra modalidade, a sintonia do microprocessador inclui sintonizar/compilar o programa de software a ser executado para utilizar o processador diferentemente, tal como não utilizando um pré-buscador de hardware.

Até o momento, o monitoramento de desempenho e a sintonia foram discutidos com referência a um único programa de software para descrever o monitoramento de desempenho. No entanto, o monitoramento de desempenho e a sintonia podem ser implementados com qualquer número de aplicativos a serem executados em um processador. A figura 6c ilustra uma modalidade de um diagrama de fluxo para perfilar/sintonizar uma arquitetura para um segundo programa e quando carregando o primeiro aplicativo novamente, o microprocessador é ressintonizado.

Os fluxos 605-615 são os mesmos como mostrados na figura 6a. No fluxo 620 a primeira configuração que representa a sintonia do microprocessador associado com o primeiro programa de software é armazenada. Um custo de evento para o percurso crítico associado com a execução de um segundo programa de software é determinado no fluxo 625. No fluxo 630, o microprocessador é sintonizado com base no custo de evento para o percurso crítico associado com a execução do segundo programa de software. Finalmente, o microprocessador é ressintonizado com base na primeira configuração armazenada quando da subseqüente execução do primeiro programa de software no fluxo 635.

Como pode ser visto acima um microprocessador é dinamicamente sintonizado com base no desempenho de aplicativos individuais. Como certas características em um processador são diferentemente utilizadas, e o custo de eventos, tal como uma perda de cache, varia significativamente de aplicativo para aplicativo, a microarquitetura e/ou os aplicativos de software estes próprios podem ser sintonizados para executarem mais eficientemente e rapidamente. O custo de eventos e as contribuições de características são medidos através de qualquer combinação de métodos analíticos, simulação, medição de descartes de retirada, e tempo de execução total para assegurar que o desempenho correto está sendo monitorado, especialmente para as máquinas de execução paralelas.

Na especificação acima, a invenção foi descrita com referência às suas modalidades exemplares específicas. Ficará, no entanto, evidente que várias modificações e mudanças podem ser feitas nas mesmas sem afastar-se do espírito mais amplo e do escopo da invenção como apresentados nas reivindicações anexas. A especificação e os desenhos devem, conseqüentemente, ser considerados em um sentido ilustrativo ao invés de um sentido restritivo.

REIVINDICAÇÕES

1. Método que compreende:

executar um primeiro programa de software utilizando um microprocessador;

determinar um custo de evento para um percurso crítico associado com a execução do primeiro programa de software; e

sintonizar o microprocessador com base no custo de evento para o percurso crítico associado com a execução do primeiro programa de software.

2. Método de acordo com a reivindicação 1, em que o microprocessador é capaz de uma execução paralela fora de ordem.

3. Método de acordo com a reivindicação 1, em que sintonizar o microprocessador compreende mudar o tamanho de uma característica microarquitetural, a característica microarquitetural sendo selecionada de um grupo que consiste em um cache de instrução, um cache de dados, uma rede alvo de ramificação, uma tabela de memória virtual e um arquivo de registro.

4. Método de acordo com a reivindicação 1, em que sintonizar o microprocessador compreende desabilitar uma característica microarquitetural, a característica microarquitetural sendo selecionada de um grupo que consiste em um cache, uma tabela de tradução um armazenamento de consulta, uma unidade de predição de ramificação, um pré-buscador de hardware, e uma unidade de execução.

5. Método de acordo com a reivindicação 1, ainda compreendendo:

armazenar uma primeira configuração que representa a sintonia do microprocessador associada com o primeiro programa de software;

determinar um custo de evento para o percurso crítico associado com a execução de um segundo programa de software;

sintonizar o microprocessador com base no custo de evento para o percurso crítico associado com a execução do segundo programa de software; e

ressintonizar o microprocessador com base na primeira configuração armazenada, quando de uma execução subseqüente do primeiro programa de software.

6. Método de acordo com a reivindicação 5, em que cada um do primeiro e do segundo programas de software é selecionado do grupo que consiste em um aplicativo de convidado, um sistema de operação, um aplicativo de sistema de operação, um aplicativo de padrão de desempenho, um operador, e um aplicativo incorporado.

7. Método de acordo com a reivindicação 1, em que a determinação de um custo de evento para um percurso crítico compreende executar uma contagem de duração.

8. Método de acordo com a reivindicação 7, em que a execução de uma contagem de duração compreende contar os ciclos em que uma máquina de estado no microprocessador está ativa, em que a máquina de estado é selecionada do grupo que consiste em um manipulador de page walk, uma máquina de estado de bloqueio, e uma fila de barramento de perdas de cache destacadas.

9. Método de acordo com a reivindicação 1, em que a determinação de um evento para um percurso crítico compreende medir os descartes de retirada de operações.

10. Método de acordo com a reivindicação 9, em que a medição de descartes de retirada de operações compreende medir um retardo na retirada de um par de operações sequencial.

11. Método de acordo com a reivindicação 9, em que a medição de descartes de retirada de operações compreende medir um retardo de retirada para uma operação que tinha um evento específico.

12. Método de acordo com a reivindicação 11, em que o evento é selecionado de um grupo que consiste em uma perda de cache de baixo nível, uma perda de cache secundária, uma perda de cache de alto nível, um acesso de cache, um snoop de cache, uma má predição de ramificação, uma busca da memória, um bloqueio na retirada, uma pré-busca de hardware, uma carga, um armazenamento, uma memória temporária, uma decodifi-

cação de instrução, uma tradução de endereço, um acesso a um armazenamento de tradução, uma execução de operando de inteiro, uma execução de operando de ponto flutuante, uma renomeação de um registro, uma programação de uma instrução, uma leitura de registro, uma escrita de registro.

13. Método que compreende:

identificar uma operação quando da ocorrência de um evento específico, a operação a ser executada em um processador capaz de uma execução paralela; e

determinar um descarte de retirada para a operação.

14. Método de acordo com a reivindicação 13, em que a identificação de uma operação compreende selecionar a operação, quando da ocorrência do evento específico, para amostragem.

15. Método de acordo com a reivindicação 13, em que a identificação de uma operação compreende selecionar a operação, quando da ocorrência do evento específico e uma não ocorrência de um segundo evento, para amostragem.

16. Método de acordo com a reivindicação 14, em que o evento específico é selecionado de um grupo que consiste em um perda de cache, um acesso de cache, um snoop de cache, uma má predição de ramificação, um bloqueio na retirada, uma pré-busca de hardware, uma carga, um armazenamento, uma memória temporária, e um acesso de armazenamento de tradução.

17. Método de acordo com a reivindicação 14, em que o evento específico é um evento preciso com base em uma amostragem de evento na retirada.

18. Método de acordo com a reivindicação 14, em que a determinação de um retardo de descarte de retirada para o operação compreende:

inicializar um primeiro contador, quando da seleção da operação para amostragem;

determinar o descarte de retirada com base na inicialização do primeiro contador e na utilização de um registro de armazenamento.

19. Método de acordo com a reivindicação 18, em que a inicialização do primeiro contador inclui ajustar o primeiro contador para um valor definido pelo usuário, e em que a utilização de um registro de armazenamento inclui quando da medida do descarte de retirada com o primeiro contador, copiar um estado do primeiro contador no registro de armazenamento para ser lido para determinar o descarte de retirada.

20. Aparelho que compreende:

um microprocessador que inclui:

um primeiro módulo para determinar uma contribuição de uma característica microarquitetural para um aplicativo de nível de usuário; e

um segundo módulo para sintonizar a característica microarquitetural com base pelo menos na contribuição da característica microarquitetural, quando o aplicativo de nível de usuário deve ser executado.

21. Aparelho de acordo com a reivindicação 20, em que determinar uma contribuição de uma característica microarquitetural para um aplicativo de nível de usuário compreende:

executar o aplicativo de nível de usuário com a característica microarquitetural habilitada;

executar o aplicativo de nível de usuário com a característica microarquitetural desabilitada; e

determinar a contribuição da característica microarquitetural para o aplicativo de nível de usuário com base na comparação da execução do aplicativo de nível de usuário com a característica habilitada com a execução do aplicativo de nível de usuário com a característica desabilitada.

22. Aparelho de acordo com a reivindicação 20, em que sintonizar a característica microarquitetural compreende: mudar o tamanho da característica microarquitetural, a característica microarquitetural sendo selecionada de um grupo que consiste em um cache de instrução, um cache de dados, uma rede alvo de ramificação, uma tabela de memória virtual, e um arquivo de registro.

23. Aparelho de acordo com a reivindicação 20, em que sintonizar a característica microarquitetural compreende desabilitar a característica

microarquitetural, a característica microarquitetural sendo selecionada de um grupo que consiste em um cache de instrução, um cache da dados, uma tabela de tradução, um armazenamento de consulta, uma unidade de predição de ramificação, um pré-buscador de hardware, e uma unidade de execução.

24. Aparelho de acordo com a reivindicação 20, em que sintonizar a característica microarquitetural está adicionalmente baseado em uma quantidade de energia consumida pela característica microarquitetural.

25. Aparelho de acordo com a reivindicação 23, em que o segundo módulo compreende:

um registro que tem um campo associado com a característica microarquitetural, em que o campo, quando ajustado, é para desabilitar a característica microarquitetural;

um módulo para ajustar o campo no registro associado com a característica microarquitetural, se a contribuição de desempenho da característica, quando desabilitada, for melhorada.

26. Aparelho que compreende:

um microprocessador que inclui,

um módulo para determinar um custo de evento por instância para execução de um programa de software; e

um módulo para sintonizar o programa de software com base no custo de evento por instância.

27. Aparelho de acordo com a reivindicação 26, em que determinar um custo de evento por instância compreende derivar o custo de evento por instância através de uma técnica de monitoramento de desempenho selecionada do grupo que consiste em contagem de duração, medição de descarte de retirada, e monitoramento de execução de rastreio longo.

28. Aparelho de acordo com a reivindicação 26, em que a sintonia do programa de software é selecionada de um grupo que consiste em recompilar o programa de software, otimizar o programa de software, otimizar o programa de software para bloquear as estruturas de dados para caber dentro de um cache, retransmitir o programa de software para aproveitar-se

de uma condição de predição de ramificação padrão, emitir um código em um diferente endereço de instrução, retransmitir dados em uma memória dinamicamente alocada, e ajustar a granularidade e o alinhamento de acessos.

29. Sistema que compreende:

um hub de controlador acoplado em uma memória e em um controlador de vídeo;

um microprocessador que inclui um módulo para,

determinar uma contribuição de evento por instância durante a execução de um programa de software;

sintonizar uma configuração arquitetural do microprocessador com base na contribuição de evento por instância;

armazenar a configuração arquitetural; e

ressintonizar a configuração arquitetura com base na configuração arquitetural armazenada, quando da execução subseqüente do programa de software.

30. Sistema de acordo com a reivindicação 29, em que o microprocessador é capaz de uma execução paralela fora de ordem.

31. Sistema de acordo com a reivindicação 29, em que a configuração arquitetural está armazenada em um registro no microprocessador.

32. Sistema de acordo com a reivindicação 29, em que a determinação de uma contribuição de evento por instância durante a execução de um programa de software compreende:

medir uma pluralidade de descartes de retirada para uma pluralidade de ocorrências de evento específicas, e

derivar a contribuição de evento por instância para o evento específico com base na pluralidade de descartes de retirada e no número de ocorrências de evento específicas.

33. Sistema de acordo com a reivindicação 29, em que a determinação de uma contribuição de evento por instância durante a execução de um programa de software compreende:

executar o programa de software uma pluralidade de vezes, em

que cada vez que o software é executado:

o número de vezes que um evento específico ocorre é mudado, e

o desempenho de um percurso crítico no microprocessador é monitorado;

derivar a contribuição de evento por instância do evento específico com base na comparação da mudança em desempenho do percurso crítico com a mudança no número de vezes que o evento específico ocorre.

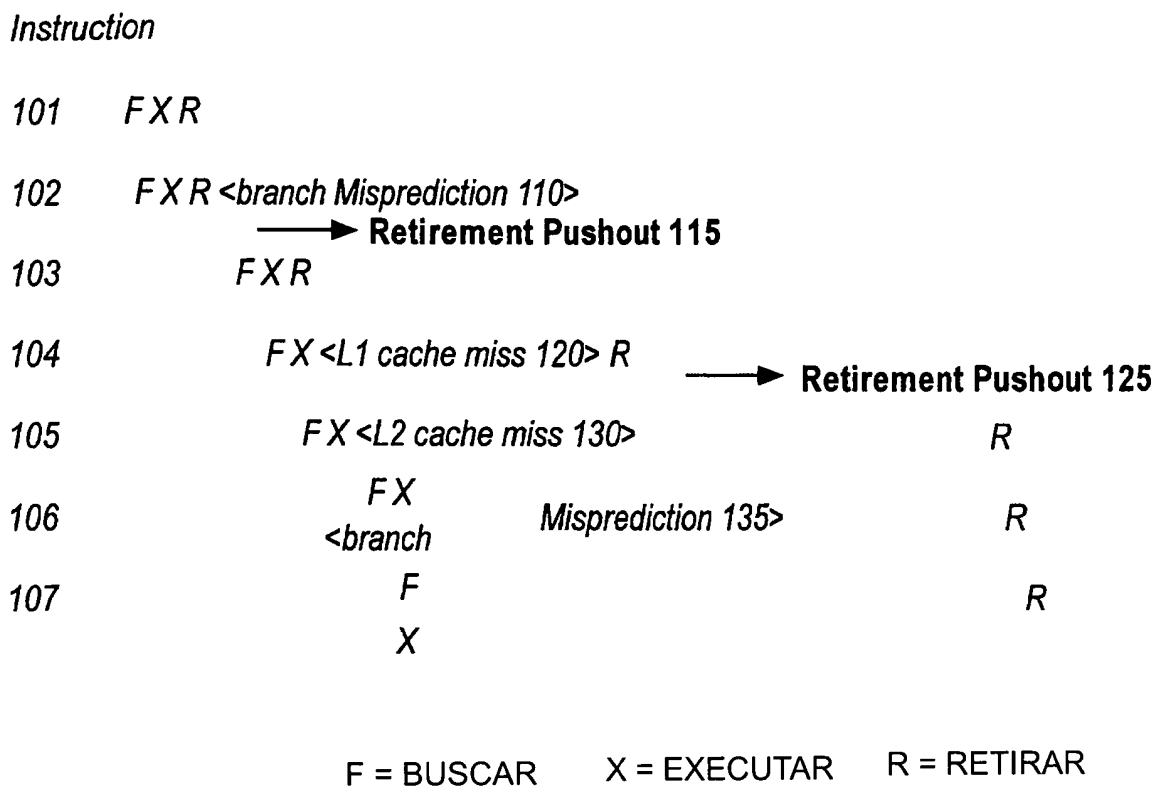

Instruction
101 F X R
102 F X R <branch Misprediction 110>
Retirement Pushout 115
103 F X R
104 F X <L1 cache miss 120> R
Retirement Pushout 125
105 F X <L2 cache miss 130> R
106 F X <branch Misprediction 135> R
107 F X R
F = BUSCAR X = EXECUTAR R = RETIRAR

FIG. 1

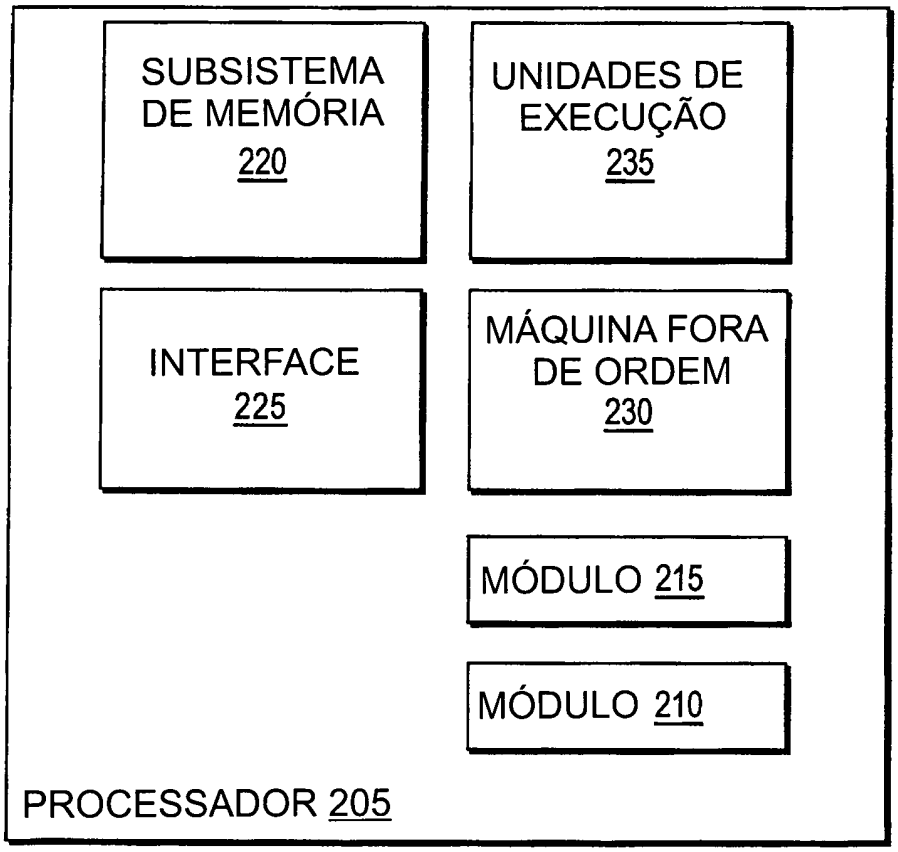

SUBSISTEMA DE MEMÓRIA 220
UNIDADES DE EXECUÇÃO 235
INTERFACE 225
MÁQUINA FORA DE ORDEM 230
MÓDULO 215
MÓDULO 210
PROCESSADOR 205

FIG. 2

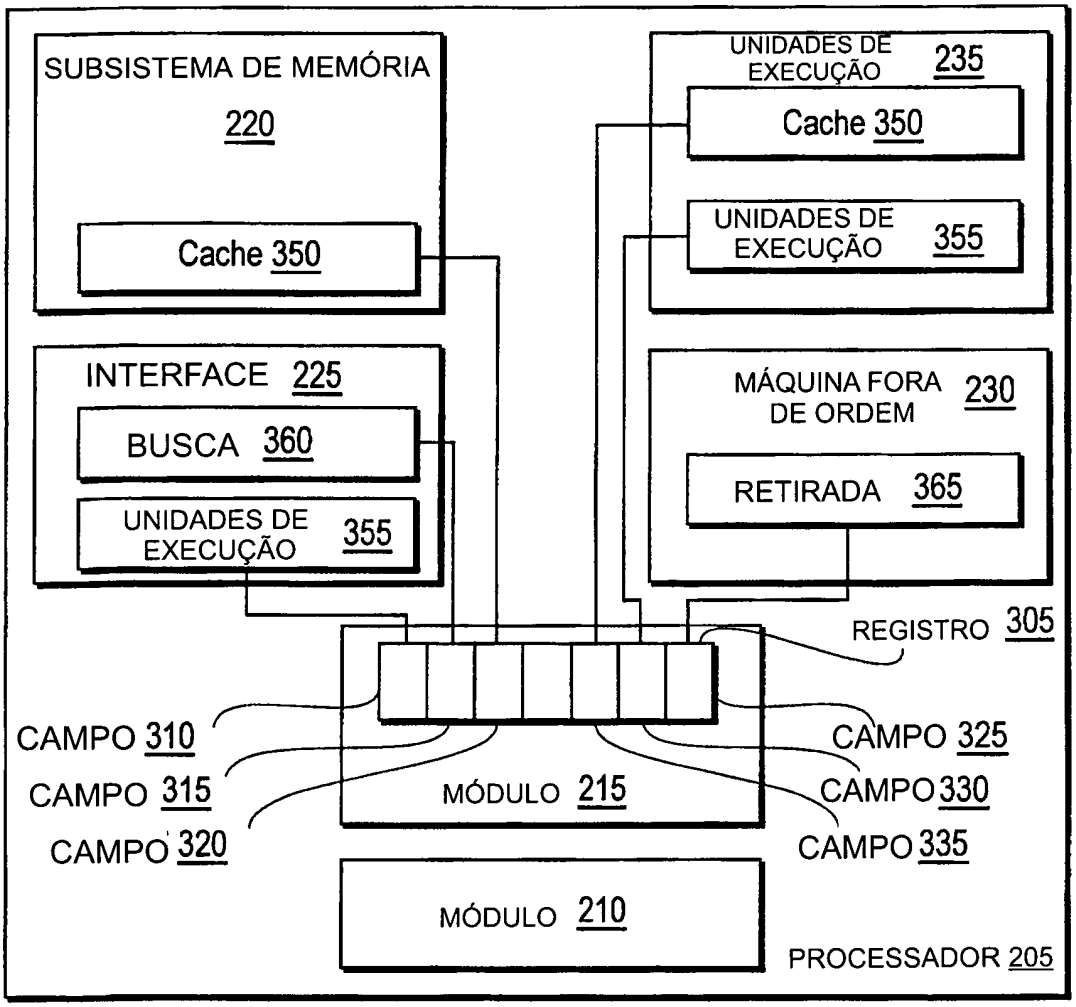

SUBSISTEMA DE MEMÓRIA
220
Cache 350
UNIDADES DE EXECUÇÃO 235
Cache 350
UNIDADES DE EXECUÇÃO 355
INTERFACE 225
BUSCA 360
UNIDADES DE EXECUÇÃO 355
MÁQUINA FORA DE ORDEM 230
RETIRADA 365
REGISTRO 305
CAMPO 310
CAMPO 315
CAMPO 320
MÓDULO 215
CAMPO 325
CAMPO 330
CAMPO 335
MÓDULO 210
PROCESSADOR 205

FIG. 3

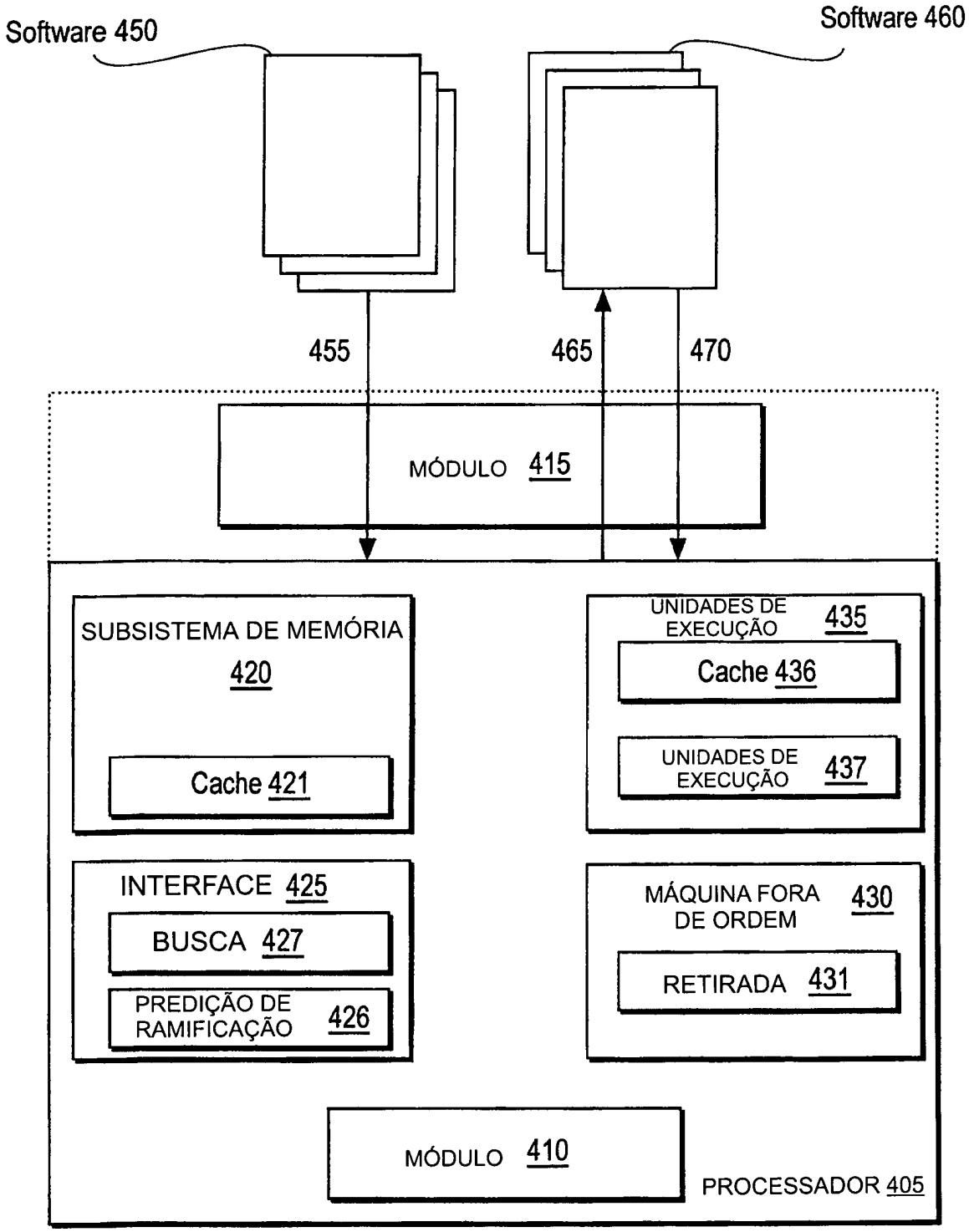

Software 450
Software 460
455
465
470
MÓDULO 415
SUBSISTEMA DE MEMÓRIA 420
Cache 421
UNIDADES DE EXECUÇÃO 435
Cache 436
UNIDADES DE EXECUÇÃO 437
INTERFACE 425
BUSCA 427
PREDIÇÃO DE RAMIFICAÇÃO 426
MÁQUINA FORA DE ORDEM 430
RETIRADA 431
MÓDULO 410
PROCESSADOR 405

FIG. 4

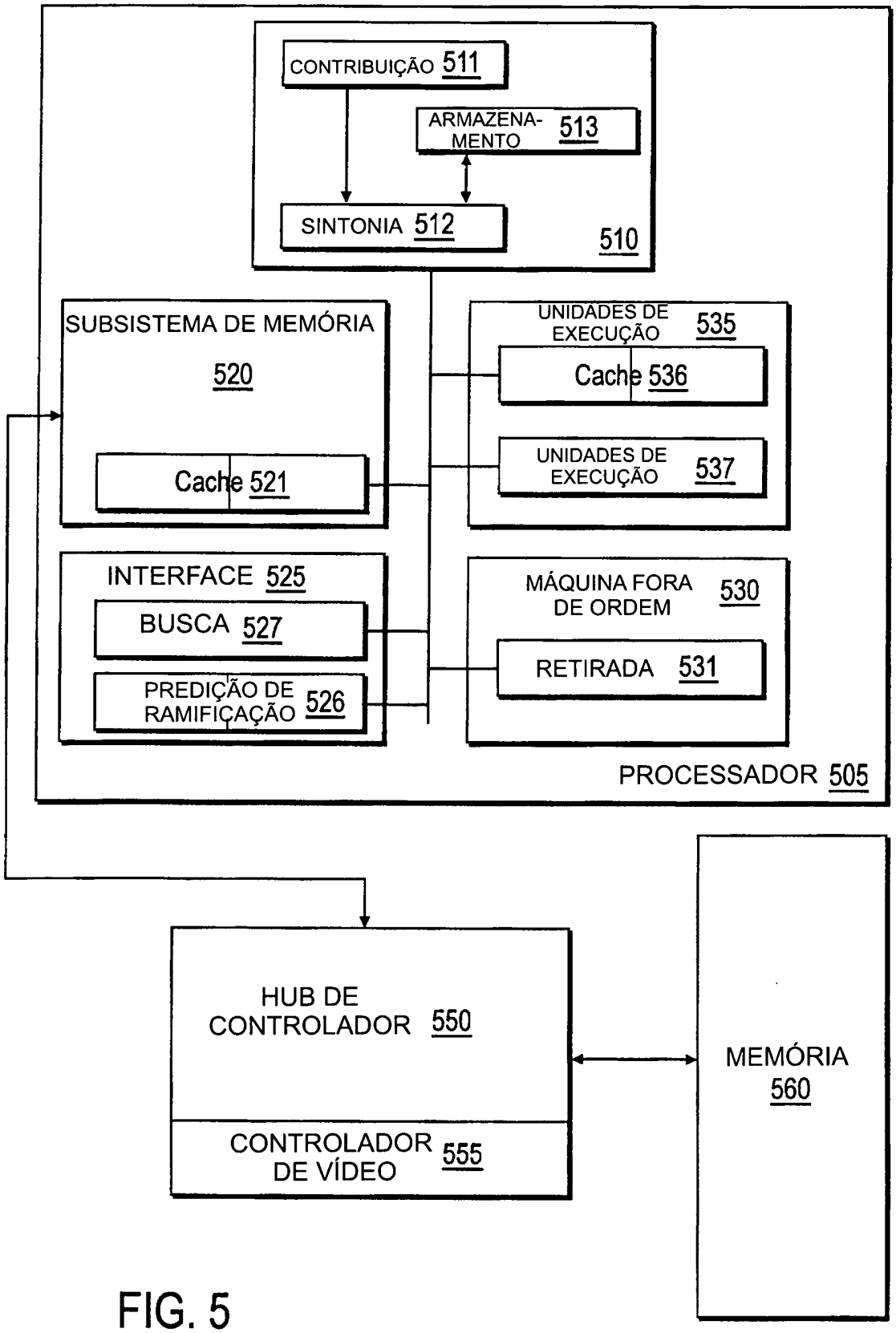

CONTRIBUIÇÃO 511
ARMAZENA-MENTO 513
SINTONIA 512
510
SUBSISTEMA DE MEMÓRIA
520
Cache 521
UNIDADES DE EXECUÇÃO 535
Cache 536
UNIDADES DE EXECUÇÃO 537
INTERFACE 525
BUSCA 527
PREDIÇÃO DE RAMIFICAÇÃO 526
MÁQUINA FORA DE ORDEM 530
RETIRADA 531
PROCESSADOR 505
HUB DE CONTROLADOR 550
CONTROLADOR DE VÍDEO 555
MEMÓRIA 560

FIG. 5

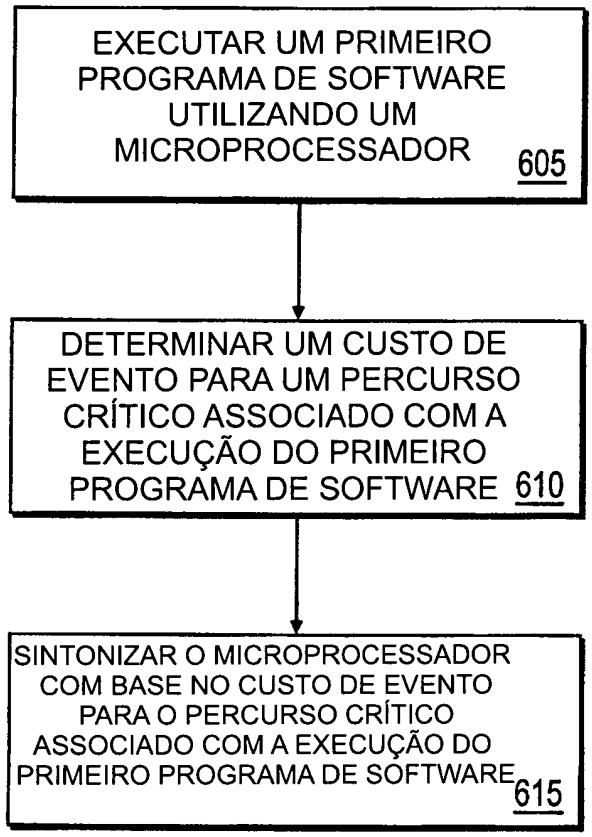

EXECUTAR UM PRIMEIRO PROGRAMA DE SOFTWARE UTILIZANDO UM MICROPROCESSADOR
605
DETERMINAR UM CUSTO DE EVENTO PARA UM PERCURSO CRÍTICO ASSOCIADO COM A EXECUÇÃO DO PRIMEIRO PROGRAMA DE SOFTWARE
610
SINTONIZAR O MICROPROCESSADOR COM BASE NO CUSTO DE EVENTO PARA O PERCURSO CRÍTICO ASSOCIADO COM A EXECUÇÃO DO PRIMEIRO PROGRAMA DE SOFTWARE
615

FIG. 6A

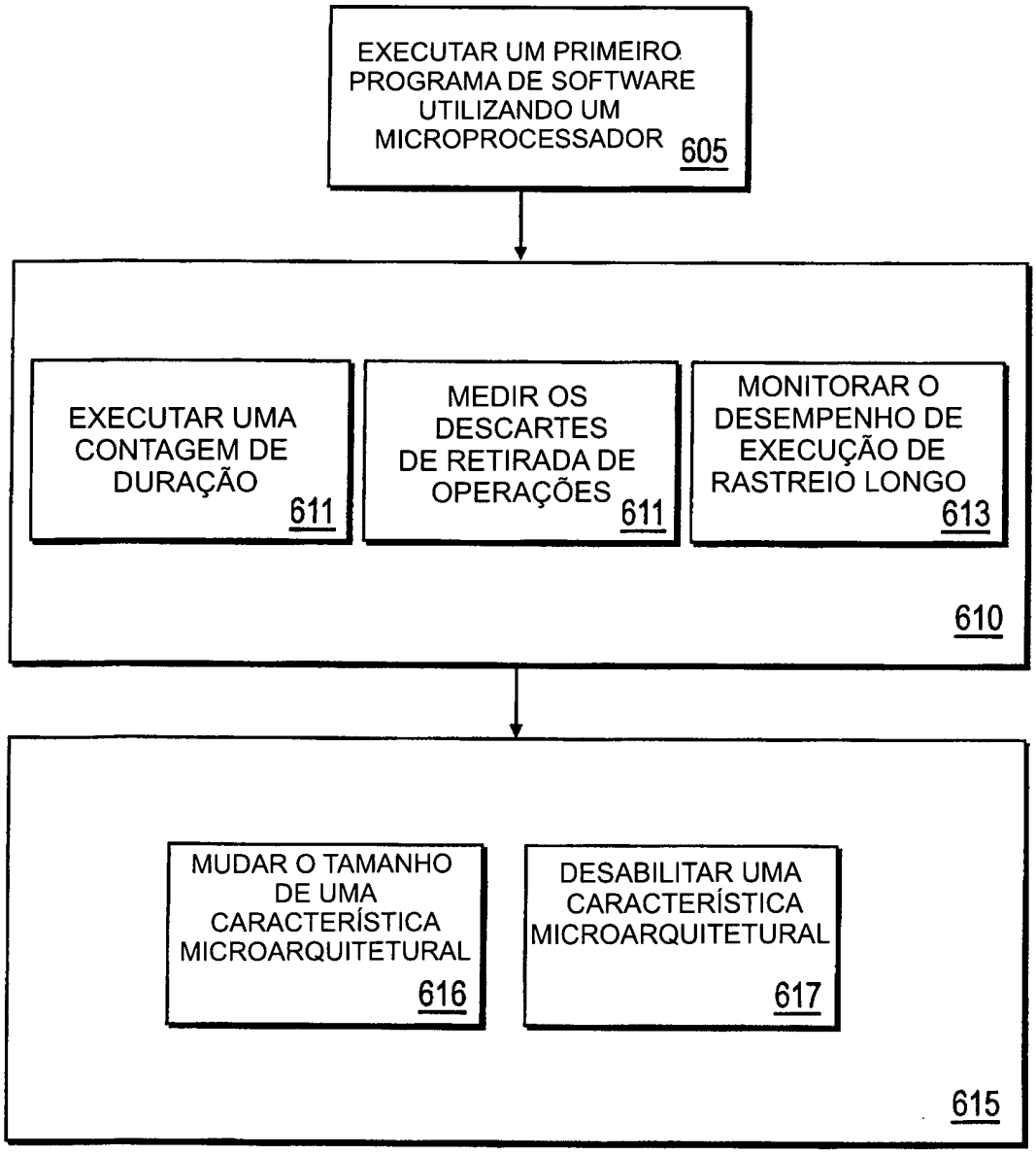

EXECUTAR UM PRIMEIRO PROGRAMA DE SOFTWARE UTILIZANDO UM MICROPROCESSADOR
605
EXECUTAR UMA CONTAGEM DE DURAÇÃO
611
MEDIR OS DESCARTES DE RETIRADA DE OPERAÇÕES
611
MONITORAR O DESEMPENHO DE EXECUÇÃO DE RASTREIO LONGO
613
610
MUDAR O TAMANHO DE UMA CARACTERÍSTICA MICROARQUITETURAL
616
DESABILITAR UMA CARACTERÍSTICA MICROARQUITETURAL
617
615

FIG. 6B

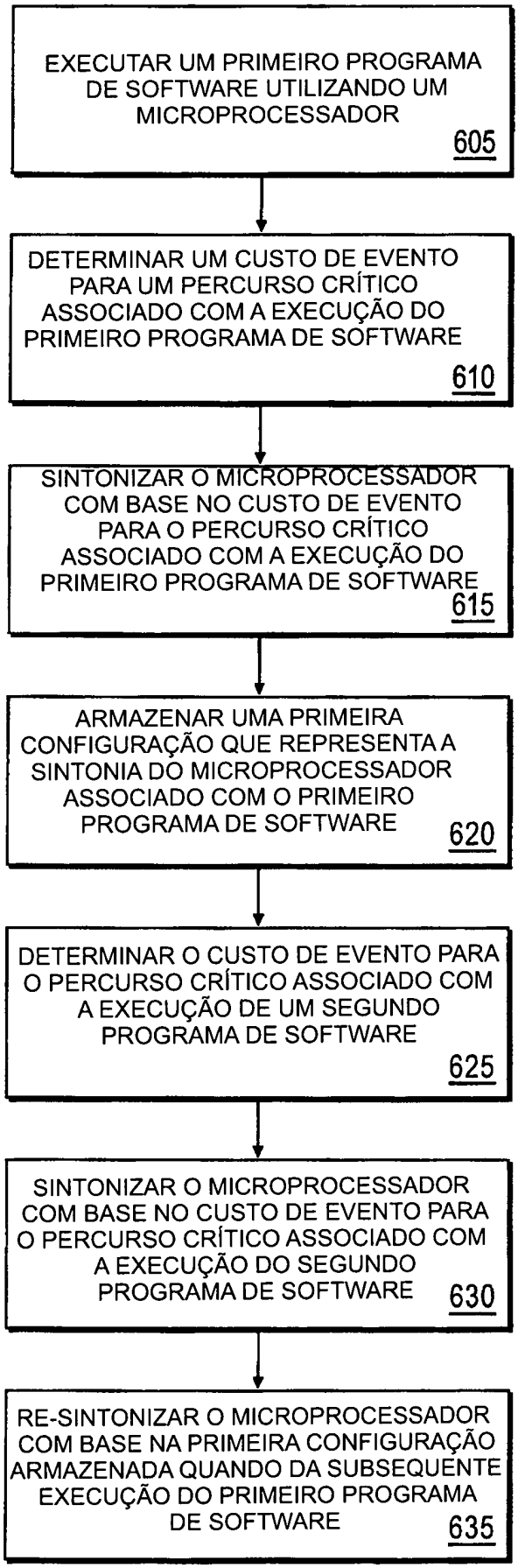

EXECUTAR UM PRIMEIRO PROGRAMA DE SOFTWARE UTILIZANDO UM MICROPROCESSADOR
605
DETERMINAR UM CUSTO DE EVENTO PARA UM PERCURSO CRÍTICO ASSOCIADO COM A EXECUÇÃO DO PRIMEIRO PROGRAMA DE SOFTWARE
610
SINTONIZAR O MICROPROCESSADOR COM BASE NO CUSTO DE EVENTO PARA O PERCURSO CRÍTICO ASSOCIADO COM A EXECUÇÃO DO PRIMEIRO PROGRAMA DE SOFTWARE
615
ARMAZENAR UMA PRIMEIRA CONFIGURAÇÃO QUE REPRESENTA A SINTONIA DO MICROPROCESSADOR ASSOCIADO COM O PRIMEIRO PROGRAMA DE SOFTWARE
620
DETERMINAR O CUSTO DE EVENTO PARA O PERCURSO CRÍTICO ASSOCIADO COM A EXECUÇÃO DE UM SEGUNDO PROGRAMA DE SOFTWARE
625
SINTONIZAR O MICROPROCESSADOR COM BASE NO CUSTO DE EVENTO PARA O PERCURSO CRÍTICO ASSOCIADO COM A EXECUÇÃO DO SEGUNDO PROGRAMA DE SOFTWARE
630
RE-SINTONIZAR O MICROPROCESSADOR COM BASE NA PRIMEIRA CONFIGURAÇÃO ARMAZENADA QUANDO DA SUBSEQUENTE EXECUÇÃO DO PRIMEIRO PROGRAMA DE SOFTWARE
635

FIG. 6C

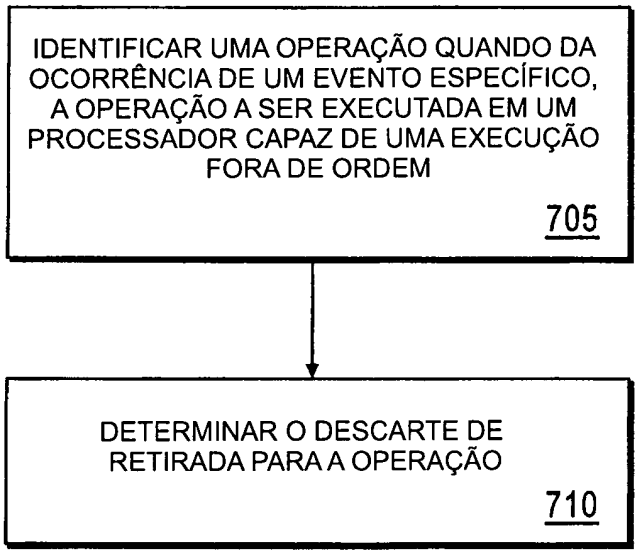

IDENTIFICAR UMA OPERAÇÃO QUANDO DA OCORRÊNCIA DE UM EVENTO ESPECÍFICO, A OPERAÇÃO A SER EXECUTADA EM UM PROCESSADOR CAPAZ DE UMA EXECUÇÃO FORA DE ORDEM
705
DETERMINAR O DESCARTE DE RETIRADA PARA A OPERAÇÃO
710

FIG. 7

RESUMO

Patente de Invenção: **"MELHORAMENTOS EM ARQUITETURA DE MONITORAMENTO DE DESEMPENHO PARA ANÁLISE BASEADA EM PERCURSO CRÍTICO"**.

A presente invenção refere-se a um método e aparelho para monitorar o desempenho de uma microarquitetura e sintonizar a microarquitetura com base no desempenho monitorado. O desempenho é monitorado através de simulação, raciocínio analítico, medição de descarte de retirada, tempo de execução total, e outros métodos para determinar os custos de evento por instância. Com base nos custos de evento por instância, a microarquitetura e/ou o software de execução é sintonizado para melhorar o desempenho.