

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

(21) **PI0711165-7 A2**

(22) Data de Depósito: 20/04/2007
(43) Data da Publicação: 23/08/2011
(RPI 2120)



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

(54) Título: **INSTRUÇÕES DE COMPRIMENTO VARIÁVEL DE PRÉ-DECODIFICAÇÃO**

(30) Prioridade Unionista: 04/05/2006 US 11/381,545

(73) Titular(es): Qualcomm Incorporated

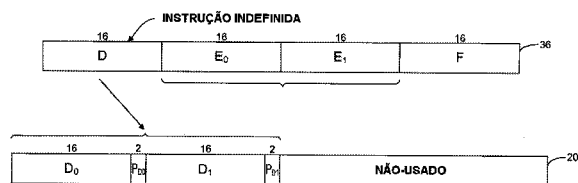
(72) Inventor(es): Brian Michael Stempel, Rodney Wayne Smith

(74) Procurador(es): Montauray Pimenta, Machado & Lioce

(86) Pedido Internacional: PCT US2007067057 de 20/04/2007

(87) Publicação Internacional: WO 2007/130798 de 15/11/2007

(57) **Resumo:** INSTRUÇÕES DE COMPRIMENTO VARIÁVEL DE PRE-DECODIFICAÇÃO Um pré-decodificador em um processador de comprimento de instrução variável indica propriedades de instruções em bits de pré-decodificação armazenados em um cache de instrução com as instruções. Quando todas as codificações de bits de pré-decodificação associados com uma instrução de comprimento são definidas, uma propriedade de uma instrução desse comprimento pode ser indicada alterando a instrução para emular uma instrução de um comprimento diferente, e codificar a propriedade nos bits de pré-decodificação associados com instruções do comprimento diferente. Um exemplo de uma propriedade que pode ser assim indicada é uma instrução indefinida.





PI0711165-7

INSTRUÇÕES DE COMPRIMENTO VARIÁVEL DE PRÉ-DECODIFICAÇÃO

CAMPO

A presente invenção relaciona-se geralmente ao campo de processadores e em particular a um método de pré-decodificar instruções de comprimento variável, como para
5 identificar instruções indefinidas.

FUNDAMENTOS

As arquiteturas de conjunto de instruções de comprimento variável são conhecidas na técnica. Por
10 exemplo, quando as extensões Thumb (de 16 bits) são adicionadas ao conjunto de instruções ARM (de 32 bits), a mistura resultante pode ser considerada um conjunto de instruções de comprimento variável. Como outro exemplo, o código Java executável compreende uma série de códigos de
15 bytes, com cada instrução variando entre 2 e 256 bytes de comprimento. Geralmente, os conjuntos de instruções de comprimento variável compreendem instruções de um tamanho base e instruções mais longas, que são um múltiplo integral do tamanho base.

20 As arquiteturas e os conjuntos de instruções de processador evoluem com o tempo. Em particular, assim que a tecnologia de semicondutor avança, a funcionalidade pode ser submetida em hardware que exigiu anteriormente uma rotina extensiva de software para ser cumprida. Para
25 explorar eficientemente as estruturas avançadas de hardware, novas instruções são adicionadas ao conjunto de instruções. Entretanto, uma característica de arquiteturas de processador estáveis é retrocompatível. Isto é, o software escrito para uma arquitetura de processador deve
30 ser executável em um processador mais antigo que se conforma à arquitetura, mesmo se não implementa a funcionalidade mais recente nem executa diretamente as

instruções mais recentes. Consequentemente, as arquiteturas de processador usuais incluem uma instrução "indefinida" que gera uma exceção, e a função associada com a instrução indefinida é executada em software.

5 Uma aproximação comum tomada por processadores de alta capacidade convencionais para otimizar decodificação de instrução é a pré-decodificação. Um pré-decodificador é um circuito lógico que inspeciona e decodifica parcialmente instruções buscadas (fetched) a partir da memória antes de
10 armazenar as instruções em um cache de instrução (I-cache). O pré-decodificador gera um pequeno número de bits de pré-decodificação que são armazenados junto com cada instrução no I-cache. Quando da busca de instruções a partir do cache, o decodificador encadeado pode utilizar os bits pré-decodificados para simplificar a tarefa de decodificação de
15 instrução. Bits pré-decodificados podem, por exemplo, identificar instruções de ramificação, identificar o comprimento de uma instrução em uma arquitetura de instrução de comprimento variável, ou similar. Bits pré-decodificados podem também ser utilizados para identificar
20 instruções indefinidas.

Em um processador de conjunto de instruções de comprimento variável que implementa pré-decodificação, um pequeno número de bits de pré-decodificação pode ser
25 associado com o comprimento de instrução mais curto, ou base. Uma implementação de cache de instrução conhecida é associar este número de bits de pré-decodificação com cada localização de armazenamento de I-cache (cada qual corresponde ao comprimento de instrução base). Instruções
30 mais longas, ocupando um múltiplo integral de localizações de armazenamento de I-cache, são associadas correspondentemente com um número maior de bits de pré-decodificação. O número de propriedades discretas que podem

ser codificadas nos bits de pré-decodificação associados com uma instrução é, assim, um mínimo para o comprimento de instrução base. Se todas as codificações de bits de pré-decodificação disponíveis do comprimento de instrução base
5 estão definidas de outra maneira, utilizar os bits de pré-decodificação para identificar uma instrução de comprimento base indefinida exigiria aumentar o número de bits de pré-decodificação associados com o comprimento de instrução base. Esta técnica aumentaria por sua vez o número de bits
10 de pré-decodificação armazenados junto com cada localização de armazenamento no I-cache. Uma vez que as instruções indefinidas são raramente encontradas, e instruções mais longas do que o comprimento base possuem espaço de codificação de bits de pré-decodificação rico, adicionar
15 bits de pré-decodificação para identificar instruções de comprimento base indefinidas pode desperdiçar espaço de armazenamento do I-cache caro.

SUMÁRIO

De acordo com uma ou mais modalidades descritas
20 aqui, um pré-decodificador pode indicar uma propriedade de uma instrução quando o espaço de codificação de bits de pré-decodificação associado com esse comprimento de instrução está inteiramente definido. Alterando a instrução para emular uma instrução de um comprimento diferente, o
25 pré-decodificador pode utilizar uma codificação disponível dos bits de pré-decodificação associados com a instrução de comprimento diferente para indicar a propriedade.

Uma modalidade relaciona-se a um método de pré-decodificar uma instrução em um processador. Uma
30 propriedade de uma primeira instrução é identificada. A primeira instrução é alterada para emular uma segunda instrução. Os bits de pré-decodificação para a segunda

instrução são gerados e identificam a propriedade da primeira instrução.

Uma outra modalidade relaciona-se a um processador operativo para ler instruções da memória. O processador inclui uma memória de cache de instrução que armazena as instruções lidas da memória e os bits de pré-decodificação associados com cada instrução. O processador também inclui um pré-decodificador interposto entre a memória e o cache de instrução, o pré-decodificador operativo para identificar uma propriedade de uma primeira instrução; alterar a primeira instrução para emular uma segunda instrução; gerar bits de pré-decodificação para a segunda instrução que identificam a propriedade da primeira instrução; e escrever os bits de pré-decodificação para a segunda instrução no cache de instrução.

BREVE DESCRIÇÃO DOS DESENHOS

Figura 1 é um diagrama de blocos funcional de um processador.

Figura 2 é um diagrama de blocos funcional descrevendo mapeamento de instruções de comprimento variável a partir da memória ao cache de instrução.

Figura 3 é um diagrama de blocos funcional descrevendo mapeamento de uma instrução indefinida a partir da memória ao cache de instrução.

Figura 4 descreve um método de pré-decodificar uma instrução, de acordo com uma modalidade.

DESCRIÇÃO DETALHADA

Figura 1 é um diagrama de blocos funcional de um processador 10. O processador 10 executa instruções em um encadeamento de execução de instrução 12 de acordo com a lógica de controle 14. O encadeamento 12 pode ser um projeto superescalar, com múltiplos encadeamentos paralelos, tais como 12a e 12b. Cada encadeamento 12a, 12b

inclui vários registradores ou latches 16, organizados em estágios de cadeia, e uma ou mais Unidades de Lógica Aritméticas (ALU) 18. Os registradores ou latches de estágio de cadeia 16 e ALUs 18 podem ler operandos de e/ou
 5 escrever resultados para os registradores em um arquivo de Registrador de Propósito Geral 28.

Os encadeamentos 12a, 12b buscam instruções de um cache de instrução (I-cache ou I\$) 20, com o endereçamento de memória e as permissões gerenciados por um Buffer de
 10 Previsão de Tradução de Lado de Instrução (ITLB) 22. Os dados são acessados a partir de um cache de dados (D-Cache ou D\$) 24, com o endereçamento de memória e as permissões gerenciados por um Buffer de Previsão de Tradução (TLB) principal 26. Em várias modalidades, o ITLB 22 pode
 15 compreender uma cópia de uma parte do TLB 26. Alternativamente, o ITLB 22 e TLB 26 podem ser integrados. Similarmente, em várias modalidades do processador 10, o I-cache 20 e o D-cache 24 podem ser integrados ou unificados. Perdas no I-cache 20 e/ou no D-cache 24 causam um acesso à
 20 memória (fora do chip) principal 36, sob o controle de uma interface de memória 34.

As instruções buscadas a partir da memória são processadas em um circuito do pré-decodificador 37 antes da armazenagem no I-cache 20. O pré-decodificador 37
 25 decodifica parcialmente as instruções, e anexa bits de pré-decodificação, que são armazenados junto com as instruções no I-cache 20. Quando as instruções são buscadas a partir do I-cache 20, os bits de pré-decodificação fornecem informações sobre cada instrução que pode ser usada por
 30 estágios de cadeia de decodificação dentro dos encadeamentos 12a e 12b para decodificar mais rapidamente e/ou mais eficientemente a instrução. Por exemplo, bits de pré-decodificação podem identificar o comprimento de uma

instrução em uma arquitetura de conjunto de instrução de comprimento variável, identificar instruções de ramificação, e semelhantes.

O processador 10 pode incluir uma interface de entrada/saída (I/O) 38, controlando acesso aos vários dispositivos periféricos 40, 42. Aqueles versados na técnica reconhecerão que inúmeras variações do processador 10 são possíveis. Por exemplo, o processador 10 pode incluir um cache de nível secundário (L2) para um ou ambos os caches I e D. Além disso, um ou mais dos blocos funcionais descritos no processador 10 podem ser omitidos a partir de uma modalidade particular.

A Figura 2 é um diagrama de blocos funcional descrevendo o mapeamento de instruções da memória 36 em uma linha no I-cache 20, através do pré-decodificador 37, para uma implementação ilustrativa e não-limitante. A instrução A é uma instrução de 16 bits (o comprimento de instrução base para este exemplo). Ao ser buscada a instrução A da memória, o pré-decodificador 37 anexa dois bits de pré-decodificação P_A , que são armazenados junto com a instrução A no I-cache 20. A instrução B é uma instrução de 32 bits, armazenada na memória 36 como B_0 e B_1 . O pré-decodificador 37 anexa quatro bits de pré-decodificação, armazenados no I-cache 20 como P_{B0} e P_{B1} . A instrução C é uma instrução de 16 bits e é armazenada no I-cache 20 junto com dois bits de pré-decodificação P_C .

Como indicado pela tabela da figura 2, as instruções A e C, tendo dois bits de pré-decodificação cada um, têm um espaço de codificação de bit de pré-decodificação de quatro. Isto é, quatro propriedades discretas - tais como comprimento de instrução, se a instrução é uma ramificação, ou similares - podem ser codificadas nos bits de pré-decodificação para as

instruções A e C. A instrução B, tendo quatro bits de pré-decodificação, têm um espaço de codificação de bit de pré-decodificação de 16. Se suportado pela arquitetura, uma instrução de 48 bits mandaria seis bits de pré-decodificação, para um espaço de codificação de 64, e assim por diante.

Supondo que, em uma dada implementação, todas as quatro codificações de bit de pré-decodificação para instruções de 16 bits, tais como A e C são definidas, um bit de pré-decodificação adicional pode ser adicionado a cada localização de armazenamento de 16 bits no I-cache para utilizar os bits de pré-decodificação para indicar uma instrução de 16 bits indefinida. Isto resultaria em cada localização de armazenamento no I-cache tendo três bits de pré-decodificação associados. As instruções de 16 bits teriam, então, um espaço de codificação de pré-decodificação de oito - somente cinco destes seriam utilizados. Adicionalmente, as instruções de 32 bits teriam um espaço de codificação de pré-decodificação de 64; instruções de 48 bits teriam um espaço de codificação de pré-decodificação de 512. Uma vez que isto está muito distante do número de propriedades discretas sobre instruções que o pré-decodificador pode indicar, aumentando o número de bits de pré-decodificação para o comprimento de instrução de 16 bits resulta em um grande desperdício de armazenamento caro de I-cache.

De acordo com uma modalidade, uma instrução de 16 bits indefinida neste exemplo pode ser alterada para emular uma instrução de 32 bits. A propriedade que a instrução é de fato uma instrução indefinida de 16 bits é codificada em uma das 16 codificações de pré-decodificação associadas com instruções de 32 bits. Na figura 3, a instrução D é uma instrução indefinida de 16 bits. Ao escrever instrução D no

I-cache 20, esta é alterada para emular uma instrução de 32 bits, compreendendo D_0 e D_1 . Os bits de pré-decodificação P_{D0} e P_{D1} são escritos, indicando que a instrução D é realmente uma instrução indefinida de 16 bits.

5 Em uma modalidade, onde somente o fato de uma instrução indefinida de 16 bits ser exigida pelo encadeamento 12, o pré-decodificador 37 pode escrever somente os bits de pré-decodificação P_{D0} e P_{D1} no I-cache 20, descartando a instrução D . Em outra modalidade, tal
10 como onde os dados dentro do campo de bit da instrução sejam comuns ou úteis ao vetor da interrupção de software, a instrução D pode ser escrita para D_0 ou D_1 , ou ambos, assim como os bits de pré-decodificação P_{D0} e P_{D1} que estão sendo escritos no I-cache 20. Em ainda outra modalidade, o
15 pré-decodificador 37 pode escrever a instrução D a um dentre D_0 ou D_1 , e pode escrever informações adicionais à outra localização de armazenamento de 16 bits de I-cache 20, assim como escrever os bits de pré-decodificação P_{D0} e P_{D1} . Em qualquer caso, são os bits de pré-decodificação P_{D0}
20 e P_{D1} que ambos emulam uma instrução de 32 bits (de modo que todos os bits de pré-decodificação são lidos) e indica que a instrução é realmente uma instrução indefinida de 16 bits. Desse modo, o pré-decodificador 37 pode indicar uma instrução indefinida de tamanho base sem expandir os bits
25 de pré-decodificação associados com as instruções de tamanho base, portanto, evitando utilização ineficiente de memória.

Na modalidade descrita na figura 3, nem a instrução E nem a F é pré-decodificada ou carregada no I-
30 cache 20. Uma vez que o comprimento efetivo de uma instrução na linha foi alterado entre a memória 36 e o I-cache 20, o restante da linha de cache não pode conter todas as instruções restantes nos espaços de memória

correspondentes (por exemplo, E e F). O pré-decodificador 37 pode marcar o restante da linha de cache como não utilizado ou inválido, ou pode simplesmente introduzir bits de pré-decodificação inválidos ou indefinidos nas posições de bit apropriadas. Na execução normal, o processador interromperá o software ao encontrar a instrução indefinida, e a linha de cache será provavelmente substituída. Utilizar um manipulador de exceção de software para manipular a interrupção (trap) causada pela instrução indefinida permite que o software emule instruções que não foram contempladas naquele tempo um processador, tal como o processador 10, é fixado em silício ou outro material semicondutor. Se o outro código acontece de ramificar na linha de cache, encontrará uma pré-decodificação incorreta e/ou não reconhecerá uma instrução esperada, fazendo com que a linha de cache recarregue ou possivelmente causando uma exceção. Uma vez que as instruções indefinidas são supostas para ser raramente encontradas, o impacto de desempenho de uma linha de I-cache corrompida devido à instrução indefinida alterada emulando uma instrução de um comprimento diferente não é de grande preocupação.

O método divulgado aqui não é limitado a uma instrução indefinida do comprimento de instrução base, como descrito acima. Geralmente, para qualquer propriedade de instrução que puder geralmente ser indicada através de bits de pré-decodificação, mas onde o espaço de codificação de bit de pré-decodificação para o comprimento de instrução associado é inteiramente definido, a propriedade pode ser indicada alterando a instrução para emular uma instrução de um comprimento diferente, e utilizando uma codificação de bit de pré-decodificação disponível das instruções de comprimento diferentes. Por exemplo, suponha que todas as 16 codificações dos quatro bits de pré-decodificação

associados com instruções de 32 bits no exemplo precedente são definidas, mas somente três ou menos das quatro codificações dos dois bits de pré-decodificação associados com instruções de 16 bits são definidas. Neste caso, uma
5 propriedade particular de uma instrução de 32 bits não definida nos bits de pré-decodificação de instrução de 32 bits pode ser indicada alterando a instrução de 32 bits para emular uma instrução de 16 bits, e codificar a propriedade relevante em uma codificação de bit de pré-
10 decodificação disponível dos dois bits de pré-decodificação associados com instruções de 16 bits. Neste caso, metade do campo de bit de instrução de 32 bits é perdida, assim a propriedade sendo codificada é preferivelmente uma que não depende do campo de bit de instrução (tal como o fato de
15 que a instrução é indefinida, embora esta não seja a única propriedade que pode ser assim indicada).

A figura 4 descreve um método de pré-decodificar uma instrução, de acordo com uma modalidade. Quando de uma perda no I-cache 20, a interface 34 de memória busca pelo
20 menos uma primeira instrução da memória 36 (bloco 50). O pré-decodificador 37 inspeciona a primeira instrução e identifica uma propriedade da primeira instrução (bloco 52), como aquela que a primeira instrução é indefinida. O pré-decodificador 37 altera a primeira instrução para
25 emular uma segunda instrução (bloco 54), tal como uma segunda instrução de um comprimento diferente. O pré-decodificador 37 gera adicionalmente bits de pré-decodificação para a segunda instrução que identifica a propriedade da primeira instrução (bloco 56). Isto é, uma
30 das codificações disponíveis dos bits de pré-decodificação associados com o segundo comprimento de instrução é alocado à propriedade relevante da primeira instrução. O pré-decodificador 37 escreve a segunda instrução e os bits de

pré-decodificação associados no I-cache 20 (bloco 58). Quando o encadeamento 12 busca a segunda instrução e os seus bits de pré-decodificação do I-cache 20, a lógica em um estágio de cadeia de Decodificação inspeciona os bits de

5 pré-decodificação para identificar a propriedade da primeira instrução (bloco 60), como o fato de que a primeira instrução é indefinida. O encadeamento 12 toma, então, a ação apropriada, como por exemplo, fazendo com que uma exceção manipule a instrução indefinida no software.

10 De acordo com uma ou mais modalidades descritas aqui, um pré-decodificador 37 pode indicar uma propriedade de uma instrução quando o espaço de codificação de bit de pré-decodificação associado com esse comprimento de instrução está inteiramente definida. Alterando a instrução

15 para emular uma instrução de um comprimento diferente, o pré-decodificador 37 pode utilizar uma codificação disponível dos bits de pré-decodificação associados com a instrução de comprimento diferente para indicar a propriedade. Quando uma propriedade de uma instrução mais

20 curta é indicada nas codificações de bits de pré-decodificação de uma instrução mais longa, uma ou mais instruções ou frações de uma instrução serão deslocadas da linha de I-cache. Quando uma propriedade de uma instrução mais longa é indicada nas codificações de bit de pré-

25 decodificação de uma instrução mais curta, alguma do campo de bit de instrução é perdida. Consequentemente, a técnica de indicação de propriedade de instrução divulgada aqui é empregada geralmente para indicar propriedades de instrução causando exceção, tais como o fato de uma instrução ser

30 indefinida, embora não seja limitada a tais casos.

Embora a presente invenção seja descrita aqui no que diz respeito às características, aspectos e modalidades particulares da mesma, será aparente que as inúmeras

variações, modificações e outras modalidades são possíveis dentro do amplo escopo da presente invenção, e consequentemente, todas as variações, modificações e modalidades devem ser consideradas como estando dentro do

5 escopo da invenção. As presentes modalidades devem consequentemente ser interpretadas em todos os aspectos como ilustrativo e não restritivo e todas as mudanças vindas dentro do significado e escala de equivalência das reivindicações anexas são pretendidas a ser adotadas aqui.

REIVINDICAÇÕES

1. Método de pré-decodificar uma instrução em um processador, compreendendo:
 - identificar uma propriedade de uma primeira
5 instrução;
 - alterar a primeira instrução para emular uma segunda instrução; e
 - gerar bits de pré-decodificação para a segunda instrução que identifica a propriedade da primeira
10 instrução.
2. Método, de acordo com a reivindicação 1, em que a primeira instrução causará uma exceção.
3. Método, de acordo com a reivindicação 2, em que a primeira instrução é indefinida.
- 15 4. Método, de acordo com a reivindicação 1, em que o conjunto de instrução do processador é um conjunto de instrução de comprimento variável, e em que as primeira e segunda instruções possuem comprimentos diferentes.
5. Método, de acordo com a reivindicação 4, em
20 que cada comprimento de instrução definida possui um número diferente de bits de pré-decodificação associados com instruções desse comprimento.
6. Método, de acordo com a reivindicação 5, em que codificações insuficientes dos bits de pré-
25 decodificação associados com instruções do comprimento da primeira instrução estão disponíveis para codificar a propriedade da primeira instrução.
7. Método, de acordo com a reivindicação 6, em que pelo menos uma codificação dos bits de pré-
30 decodificação associados com instruções do comprimento da segunda instrução está disponível para codificar a propriedade da primeira instrução.

8. Método, de acordo com a reivindicação 1, em que a primeira instrução é uma instrução de 16 bits possuindo dois bits de pré-decodificação associados a ela.

9. Método, de acordo com a reivindicação 8, em
5 que a segunda instrução é uma instrução de 32 bits possuindo quatro bits de pré-decodificação associados a ela.

10. Método, de acordo com a reivindicação 9, em que uma das dezesseis codificações dos bits de pré-decodificação associados com instruções de 32 bits
10 identifica a instrução de 32 bits associada como uma instrução indefinida de 16 bits.

11. Processador operativo para ler instruções a partir de memória, compreendendo:

15 uma memória cache de instrução armazenando instruções lidas a partir de memória e bits de pré-decodificação associados com cada instrução;

um pré-decodificador interposto entre a memória e o cache de instrução, o pré-decodificador operativo para:

20 identificar uma propriedade de uma primeira instrução;

alterar a primeira instrução para emular uma segunda instrução;

25 gerar bits de pré-decodificação para a segunda instrução que identifica a propriedade da primeira instrução; e

escrever os bits de pré-decodificação para a segunda instrução no cache de instrução.

12. Processador, de acordo com a reivindicação
30 11, em que as instruções lidas a partir de memória são a partir de um conjunto de instrução de comprimento variável.

13. Processador, de acordo com a reivindicação 12, em que o cache de instrução compreende uma pluralidade

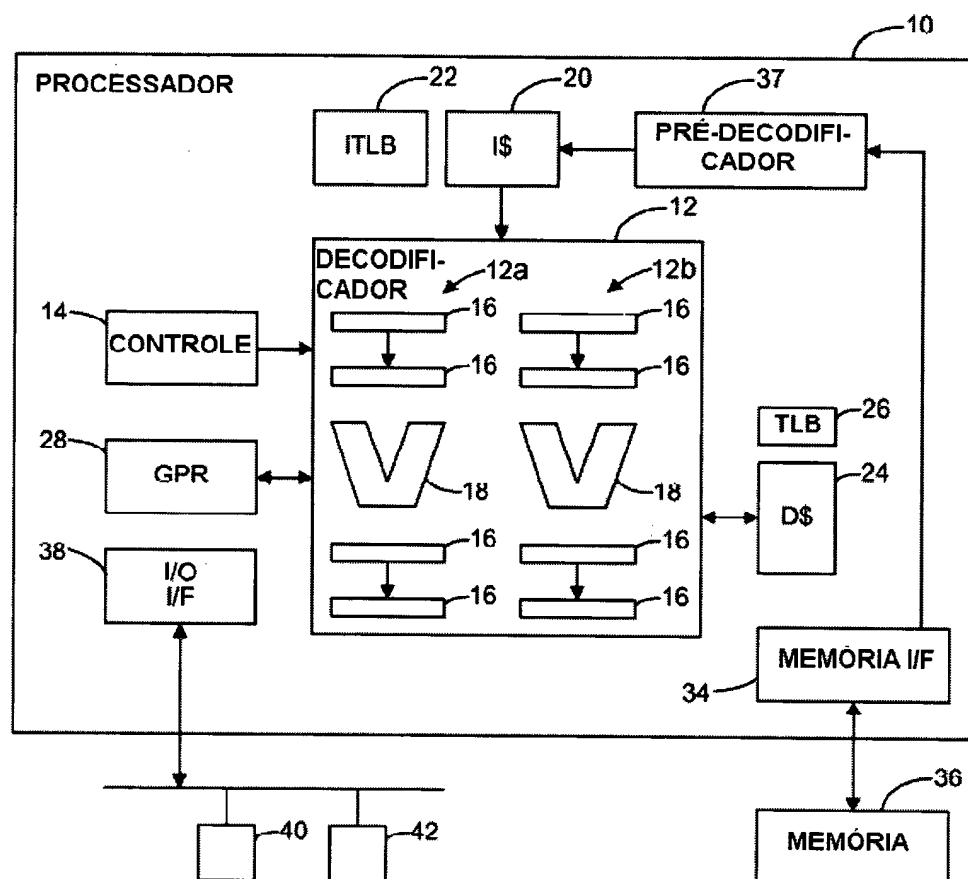
de localizações de armazenamento de instrução correspondendo ao comprimento da instrução mais curta, e em que um número predeterminado de bits de pré-decodificação são associados com cada localização de armazenamento.

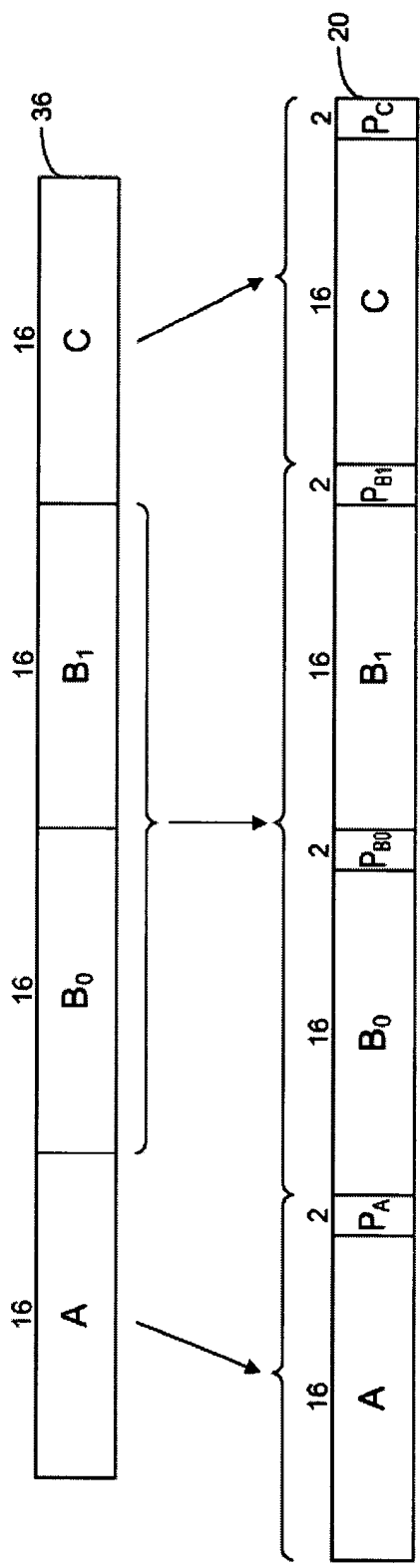
5 14. Processador, de acordo com a reivindicação 13, em que o cache de instrução armazena instruções mais longas que a instrução mais curta em um múltiplo inteiro de localizações de armazenamento, e associa com cada instrução os bits de pré-decodificação associados com cada
10 localização de armazenamento.

15 15. Processador, de acordo com a reivindicação 13, em que a primeira instrução é uma instrução de 16 bits possuindo dois bits de pré-decodificação associados a ela.

16 16. Processador, de acordo com a reivindicação 15, em que a segunda instrução é uma instrução de 32 bits possuindo quatro bits de pré-decodificação associados a ela.

20 17. Processador, de acordo com a reivindicação 16, em que uma das dezesseis codificações dos bits de pré-decodificação associados com instruções de 32 bits identifica a instrução de 32 bits associada como uma instrução indefinida de 16 bits.

**FIG. 1**



<u>INSTRUÇÃO</u>		<u>COMPRIMENTO</u>		<u>NÚMERO DE BITS DE</u>		<u>ESPAÇO DE CODIFICAÇÃO DE</u>	
				<u>PRÉ-DECODIFICAÇÃO</u>		<u>BIT DE PRÉ-DECODIFICAÇÃO</u>	
A	16-BIT	2	4				
B	32-BIT	4	16				
C	16-BIT	2	4				

FIG. 2

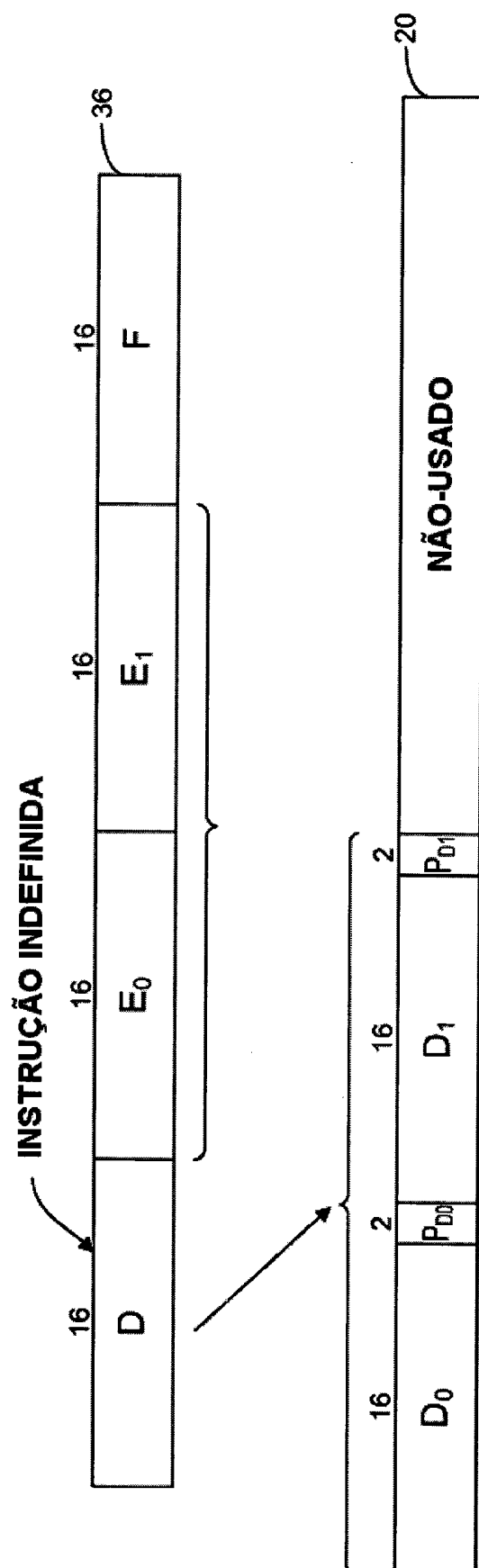
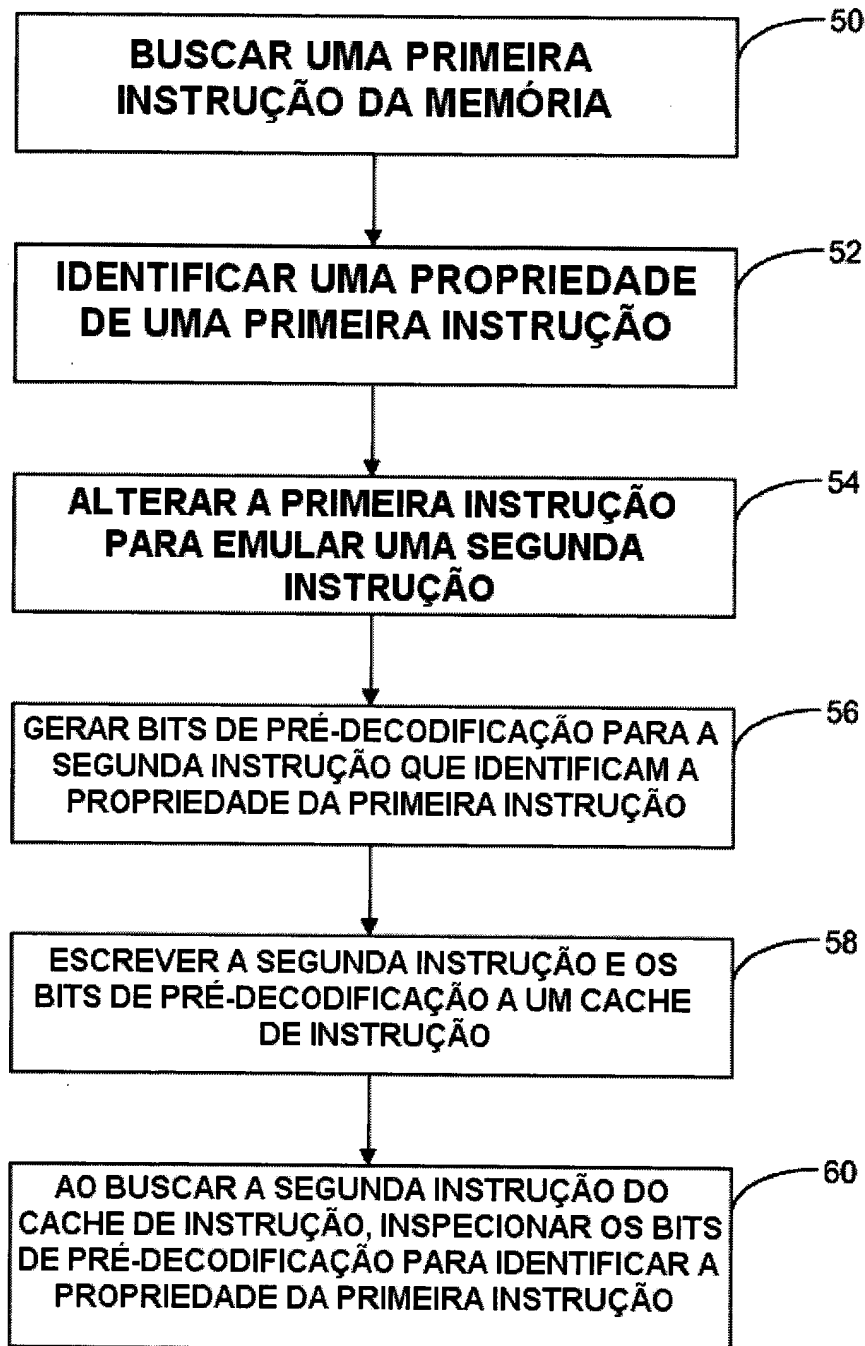


FIG. 3

**FIG. 4**

RESUMO**INSTRUÇÕES DE COMPRIMENTO VARIÁVEL DE PRÉ-DECODIFICAÇÃO**

Um pré-decodificador em um processador de comprimento de instrução variável indica propriedades de
5 instruções em bits de pré-decodificação armazenados em um
cache de instrução com as instruções. Quando todas as
codificações de bits de pré-decodificação associados com
uma instrução de comprimento são definidas, uma propriedade
de uma instrução desse comprimento pode ser indicada
10 alterando a instrução para emular uma instrução de um
comprimento diferente, e codificar a propriedade nos bits
de pré-decodificação associados com instruções do
comprimento diferente. Um exemplo de uma propriedade que
pode ser assim indicada é uma instrução indefinida.