A Arquitetura VLIW

Conceitos e Perspectivas

Thomaz Eduardo de Figueiredo Oliveira
Instituto de Computação - Unicamp
thomaz.oliveira@students.ic.unicamp.br

RESUMO

Ao final da década de 2000 os projetistas enfrentam sérios obstáculos para manter o aumento contínuo de performance nos microprocessadores. Nos paradigmas atuais, aumentar a performance resulta normalmente em aumento de consumo energético, tornando impraticável a implementação na nova área de sistemas embutidos. O artigo apresenta uma filosofia de arquitetura denominada VLIW, que expande o paralelismo ao compilador, delegando a este tarefas antes realizadas por hardware. Esta nova perspectiva gera uma grande simplificação nos componentes de hardware e maior fator de paralelismo em nível de instrução.

1. INTRODUÇÃO

O aumento contínuo da performance dos computadores enfrenta no final da década de 2000 sérios obstáculos: os microprocessadores chegaram em seus limites físicos; os multiprocessadores precisam superar as dificuldades de comunicação, coerência e divisão de recursos; e os superescalares lidam com problemas para reduzir o overhead proveniente da busca de instruções paralelas. Especialistas na área entendem que novos paradigmas devem ser construidos para que a evolução da performance continue.

Assim, a arquitetura VLIW foi concebida, visando uma nova forma de não só produzir, como aumentar o nível do paralelismo em nível de instruções (ILP).

O artigo apresentará uma breve história da arquitetura, assim como suas características principais. Logo, mostraremos algumas comparações entre os paradigmas atuais, alguns problemas gerados pela aquiterura a serem resolvidos e alguns exemplos de implementações práticas. Por fim, concluimos com a situação atual do VLIW.

2. HISTÓRIA

No inicio da década de 80, um pesquisador chamado Joseph Fisher estava ciente das limitações do ILP nas arquiteturas da época, que eram capazes de gerar no máximo duas a três operações paralelas.

Fisher compreendeu que a dificuldade não estava na execução de múltiplas operações simultâneamente (já haviam máquinas que processavam muitas instruções de uma vez, porém a codificação era manual), mas na capacidade do hardware encontrar paralelismo no código.

Com isso, ele cria o VLIW (Very Long Instruction Words), que na teoria seria capaz de executar de dez a trinta instruções em um ciclo.

Juntamente com a criação da arquitetura, Fisher desenvolve o escalonamento por traçado, uma técnica de compilação que processa o código para encontrar um caminho que tenha uma frequência de execução altíssima, independente da entrada do programa.

Esta técnica permitiu a busca de instruções paralelas além dos blocos básicos.

Fisher desenvolve então, uma máquina denominada ELI-512 junto com um compilador chamado Bulldog, enfrentando alguns obstáculos como o problemas dos desvios e o acesso múltiplo à memória.

Em seguida lança o processador comercial Multiflow, que não foi muito aceito no mercado da época (apenas cem unidades vendidas) e vai à falência.

Alguns anos mais tarde, grandes empresas como HP e Philips passam a se interessar pela arquitetura e convidam Fisher e sua equipe a prosseguir com o desenvolvimento do VLIW.

3. CARACTERÍSTICAS

Com as grandes mudanças que a arquitetura recebeu desde o seu nascimento, muitos autores encontram dificuldades em definir claramente as características do VLIW. Podemos listar alguns principios básicos que os projetos precisam seguir:

- As instruções do VLIW consistem em operações paralelas, definidas em tempo de compilação, ou seja, o hardware presume que o conjunto de operações geradas pelo compilador podem ser executadas ao mesmo tempo. Estas instruções podem ser padronizadas, com cada operação específica em uma determinada posição.

- Muita informação do hardware é visível ao compilador no momento de gerar as instruções paralelas. Desta forma, para garantir que o programa funcione corretamente, é necessário compilar o mesmo para cada implementação específica.

- O hardware confia no paralelismo gerado pelo compilador, portanto, ele não verifica dependências entre instruções nem recursos disponíveis em tempo de execução.

- Algumas peculiaridades não são expostas ao compilador, deixando a responsabilidade ao hardware. Como exemplo, o tratamento de interrupções.

A filosofia do VLIW portanto, resume-se em infundir paralelismo em toda a arquitetura, extendendo também o compilador como parte desta. Com isso, os problemas de identificar unidades funcionais disponíveis, dependências e especulação passam a ser resolvidos na compilação.

A possibilidade de enxergar o compilador como parte da arquitetura revela algumas vantagens:

- O compilador possui uma janela de visualização do código muito maior que o hardware de despacho de instruções, podendo assim buscar paralelismo em trechos maiores do programa, como loops; além de ser capaz de identificar variáveis que são constantes no progrma, podendo especular em desvios condicionais.

- A complexidade no compilador é menos custosa no geral, pois o trabalho é realizado apenas uma única vez. Ao contrário do escalonamento em hardware, que precisa sempre buscar ILP nos programas.

- O projeto do hardware pode ser mudado com maior facilidade, e menos esforço. Simuladores também podem ser mais confiávies, pois o hardware não modificará a maneira como o codigo será executado.

Estas vantagens diminuem significantemente o custo do hardware, tanto no âmbito financeiro como no de consumo energético.

Quando a instrução longa chega no hardware, cada operação é diretamente alocada para cada unidade funcional responsável. Se não houver operações suficientes, a unidade fica ociosa no ciclo em questão.

Muitas implementações de VLIW foram feitas nos últimos anos, algumas caracteristicas foram adicionas e outras retiradas. Algumas vezes, os autores inclusive enxergaram a necessidade de nomear seus projetos baseados no VLIW de maneira distinta, como exemplo, a arquitetura EPIC desenvolvida em conjunto entre Intel e HP.

4. COMPARAÇÕES

A arquitetura VLIW tem bastante semelhanças com o modelo RISC. Na verdade, muitos afirmam que o VLIW é apenas uma implementação da arquitetura RISC, pois usa os mesmos conjuntos de instruções simples e regulares, além de utilizar-se de técnicas de pipeline.

As diferenças se encontram no tamanho das instruções e o número de unidades funcionais, pois o VLIW pode processar um número maior de instruções RISC por vez. Com isso, o pipeline de uma arquitetura VLIW é distinto, que deve ter várias unidades para decodificação/leitura de registros para dar suporte às várias operações contidas na instrução.

Tabela 1: Comparação VLIW e Superescalares

	Superescalares	VLIW
Instruções	Formadas por operações escalares únicas	Formadas por múltiplas operações escalares
Escalonamento	Dinâmico por hardware	Estático pelo compilador
Num. instr. despachadas	Definida dinâmicamente pelo hardware, levando em conta as dependências e os recursos disponíveis	Determinado estaticamente pelo compilador
Ordem	Permite execução fora de ordem	Execução apenas em ordem

Outro fato importante é que a unidade de despachamento de instruções é bem mais simples nas arquiteturas VLIW, assim como os decodificadores e seu buffer de reordenação. Com isso, muitos outros registradores podem ser alocados.

A maior peculiaridade no VLIW porém, é o desenvolvimento conjunto de técnicas de compilação que sejam cooperativas com a arquitetura. Os superescalares por sua vez, executam aplicações cujos compiladores foram projetados para reduzir tamanho de código e tempo de execução. Estas características serviriam para processadores escalares, mas prejudicam arquiteturas que buscam maior fator de paralelismo.

Os dois esquemas podem ser comparados nas Figuras 1 e 2. Nesta comparação, o mesmo fator de ILP foi atingido nos dois casos. A diferença é o meio como são alcançados. Enquanto no Superescalar, são componentes de hardware que buscam as instruções, no VLIW este trabalho é feito pelo compilador, antes do tempo de execução.

5. PROBLEMAS

Como qualquer arquitetura, o modelo VLIW também possui alguns problemas.

Primeiramente, as técnicas atuais dos compiladores não são capazes de gerar instruções paralelas para qualquer tipo de aplicação. Programas muito interativos por exemplo, não podem ter suas instruções paralelizaveis. Isso irá gerar instruções grandes com poucas operações, tornando ociosas muitas unidades funcionais.

Além disso, o fato do compilador produzir instruções bastante extensas, que podem chegar até a 512 bits, consequentemente gera códigos extensos, tornando um obstáculo em sistemas com restrição de memória.

Outra observação é que algum overhead precisa ser introduzido para sincronizar as várias unidades funcionais, ou seja,

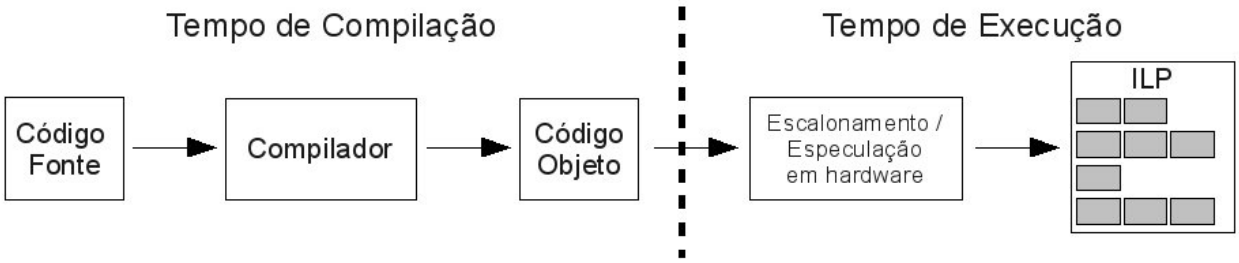

Figura 1: Esquema Superescalar

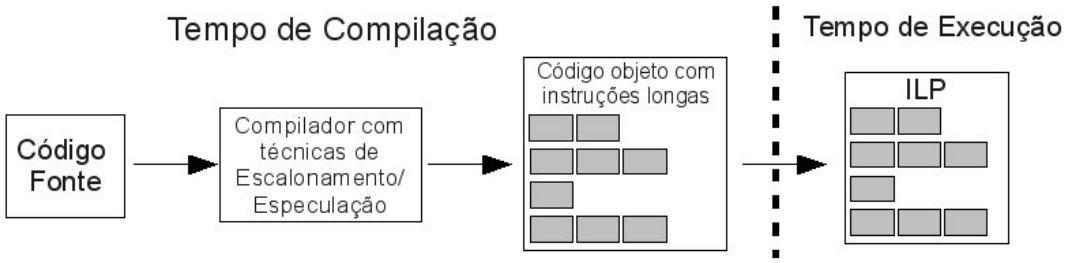

Figura 2: Esquema VLIW

dependendo do fator de paralelização, esta quantidade pode se tornar considerável.

Finalmente, como a arquitetura se apoia no compilador, um mesmo aplicativo não pode ser executado em máquinas VLIW com diferentes fatores de paralelismo. A razão é que o número de operações em cada instrução seriam diferentes em máquinas distintas. As novas gerações de VLIW, permitem maior flexibilidade de execução, no entanto, as unidades funcionais continuariam ociosas.

6. MERCADO E SITUAÇÃO ATUAL

No mercado de desktops e servidores o VLIW foi pouco aceito, as poucas empresas que lançaram processadores para este mercado (Multiflow e Cydrome) não obtiveram sucesso e logo faliram.

Grande parte do fracasso do VLIW deve-se à chamada "inércia de conjunto de instruções" que domina as maquinas no mercado, ou seja, os usuários são desejosos em protejer seus investimentos feitos em software compilado historicamente em arquiteturas CISC ou RISC com instruções simples. Assim, é preferível manter o programa com um grau limitado de eficiência a fazer grandes investimentos para mudar a arquitetura.

Porém, o mercado de sistemas embarcados não se incluem neste esquema, pois são os próprios fabricantes que consomem os softwares, o que permite maior facilidade para mudança de arquiteturas.

Este mercado justamente é o mais promissor para o VLIW, que traz algumas soluções para os problemas peculiares destes sistemas, como a flexibilidade e facilidade de projetar o hardware, além do menor consumo energético.

Algumas empresas tem projetos envolvendo a arquitetura. Entre elas a Philips que produz sistemas embarcados para celular. Podemos citar também a HP, que tem o próprio Joseph Fisher como diretor de sua divisão de pesquisa.

7. PROCESSADORES

Nesta parte iremos descrever alguns processadores VLIW produzidos para o mercado de sistemas embutidos.

7.1 Transmeta Crusoe e Efficeon

Os processadores da Transmeta possuem uma caracteristica bastante interessante, o conjunto de instruções é implementado em software ao invés de ser colocado no hardware. Assim, é possível emular qualquer arquitetura, inclusive a x86.

Usando este princípio, a empresa desenvolveu dois processadores VLIW, o Crusoe (com instruções de 128 bits) e o Efficeon (com 256 bits).

Seus processadores são famosos por garantir economia de energia, sendo implementados em vários produtos.

7.2 Philips TriMedia

Produzido pela NXP, divisão de semicondutores criada pela Philips, os processadores TriMedia são especializados em processar dados multimedia. Eles contam com 5 a 8 operações em uma instrução. A redução de hardware no processador permitiu a alocação de 128 registradores de 32-bits.

Estes processadores também são muito usados no mercado, como na área de aparelhos celulares.

7.3 Tensilica Xtensa LX2

O processador Xtensa LX2 emite instruções de 32 ou 64 bits. Através de sua arquitetura denominada FLIX (Flexible Length Instruction Xtension) garante flexibilidade nas

instruções. Otimizando assim a performance e o consumo de energia.

Outra grande vantagem é o compilador XCC, que gera paralelismo diretamente de programas escritos na linguagem C ou C++.

8. CONCLUSÃO

Pode-se afirmar que dentre as formas de obter paralelismo – pipelining, processadores múltiplos, superescalares – a arquitetura VLIW é a mais simples e barata; porém contem algumas limitações, agora relacionadas ao compilador, ou seja, para que arquitetura seja eficiente serão necessárias pesquisas para improvisar técnicas de compilação

Além disso, acredita-se que mesmo os compiladores em algum momento encontrarão limitações para encontrar instruções paralelas. Para isso, novos paradigmas de programação seriam necessários para que o paralelismo seja pensado desde o desenvolvimento do aplicativo.

9. REFERENCIAS

[1] Joseph A. Fisher, *Very Long Instruction Word architectures and the ELI-512*. International Symposium on Computer Architecture. Proceedings of the 10th annual international symposium on Computer architecture: 140-150, New York, USA.

[2] Joseph A. Fisher, *Retrospective: Very Long Instruction Word Architectures and the ELI- 512*. Hewlett-Packard Laboratories, Cambridge, Massachusetts.

[3] *An Introduction To Very-Long Instruction Word (VLIW) Computer Architecture*. Philips.

[4] John L. Hennessy, David A. Patterson, *Arquitetura de Computadores: Uma abordagem Quantitativa*. Editora Campus, Terceira Edição, 2003.

[5] Joseph A. Fisher, Paolo Faraboschi, Cliff Young, *Embedded Computing, A VLIW Approach to Architecture, Compilers and Tools*. Elsevier, 2005.