Arquiteturas VLIW

Renato Albarello
RA: 068022
Instituto de Computação
Universidade Estadual de Campinas - UNICAMP
albarello@gmail.com

RESUMO

Neste artigo veremos o paralelismo em nível de instrução com a arquitetura VLIW, que tem se mostrado muito eficiente conseguindo em muitos casos, desempenho equiparável com as máquinas superescalares tradicional. A arquitetura VLIW (Very Long Instruction Word) tenta alcançar maiores níveis de paralelismo de instrução pela execução de instruções longas compostas por múltiplas operações. Possui a vantagem de exigir menos complexidade de hardware, porém, os compiladores para VLIW ficaram bem mais complexos e sofisticados.

PALAVRAS CHAVE

VLIW, ILP, Paralelismo

1.INTRODUÇÃO

Atualmente observamos uma grande preocupação no mundo em aumentar o desempenho das máquinas. Neste caso o desempenho dos processadores, dos hardwares disk, das memórias, entre outros é uma grande preocupação do mundo da informática.

Desempenho de processadores esta sendo muito estudado, sendo que a execução de instruções em paralelo, é muito utilizada para melhorias no desempenho. A abordagem mais conhecida é o uso de pipelines que esta sendo usado nos processadores, desde 1985. Este paralelismo entre instruções é chamado paralelismo em nível de Instrução (ILP = Instruction Level Parallelism).

Existem dois focos na exploração do ILP, as **técnicas dinâmicas** que dependem do hardware (não exploradas pelo VLIW), e as **técnicas estáticas** que se baseiam mais em software, VLIW explora a técnica estática e é fortemente dependente de otimizações em tempo de compilação para apresentar bom desempenho no seu ILP.

Neste trabalho analisaremos a arquitetura VLIW, esta, em vês de tentar emitir diversas instruções independentes para as unidades, reúne várias operações em uma instrução muito longa, assim utiliza varias unidades funcionais independentes para executá-la de maneira concorrente, alcançando alto grau de ILP. Instruções longas consistem de várias operações aritméticas, lógicas e de controle, e cada uma poderia ser uma operação individual em um processador RISC simples por exemplo.

VLIW, não utiliza ILP baseada em hardware, utiliza ILP baseada software, é fortemente dependente de otimizações em tempo de compilação para apresentar bom desempenho.

A arquitetura VLIW, se baseia nas tecnologias do compilador, não apenas para minimizar os perigos de dados, mas também, para formatar as instruções em um pacote de emissão potencial, com isso o hardware não precisa verificar explicitamente as dependências, o compilador pode assegurar que não existe dependências, ou indicar ao hardware uma dependência. Neste caso, a grande vantagem é que o hardware pode ser mais simples.

As primeiras VLIW que surgiram eram bastante rígidas em seus formatos de instruções, e efetivamente exigiam a recompilação de programas para diferentes tipos de hardware. Com o tempo, foram adotadas várias inovações para reduzir esta inflexibilidade, embora elas ainda exigem que o compilador faça a maior parte do trabalho de localizar e fazer o escalonamento para execução em paralelo. Uma segunda geração de VLIW, é uma abordagem que esta sendo perseguida para uso nos mercados de desktop e servidores.

Este texto pretende apresentar o que é a arquitetura VLIW e apresentar exemplos de arquiteturas existentes. Na seção 2 teremos um breve histórico de duas empresas que exploraram esta tecnologia, na seção 3 mostraremos as principais características da arquitetura VLIW seguido da seção 3.1 que discutiremos as vantagens e 3.2 as desvantagens. Veremos também um pouco de sistemas embarcados com VLIW, Trace Sheduling e aprofundar em algumas arquiteturas que exploram VLIW.

2.HISTÓRICO

Devido a grande preocupação em aumentar o desempenho das máquinas, o desempenho de processadores esta sendo muito estudado, o surgimento das máquinas VLIW se deve a este fato.

Antes das primeiras máquinas VLIW, existiam muitos dispositivos computacionais e processadores que utilizavam instrução longa, como no VLIW hoje, para controlar a execução dos programas em diversas unidades funcionais em paralelo, porém, normalmente estas máquinas eram programadas manualmente e o código utilizado para essas máquinas não poderia ser compilado e executado para outras arquiteturas, devido ao fato dos compiladores daquela época só explorarem paralelismo dentro dos limites dos blocos básicos, resumindo, um código era feito exclusivamente para um hardware, sem a possibilidade de migra-lo, sem uma recompilação.

Joseph A. Fisher, um pioneiro em VLIW, desenvolveu uma técnica de compactação de micro código, chamada de “trace scheduling” que discutiremos na seção 4, que poderia ser

utilizado em compiladores para gerar código para arquiteturas do tipo VLIW a partir de código seqüencial comum.

Com seu trabalho foi desenvolvido um processador chamado "processador ELI-512" e a criação do compilador "trace scheduling" Bulldog.

Houve em 1984 a fundação de duas companhias para criar e construir computadores com tecnologia VLIW: **Multiflow** (criada por Fisher e seus colegas da Universidade de Yale) e **Cydrome**(fundada por Bob Rau, que foi outro pioneiro da VLIW e seus colegas).

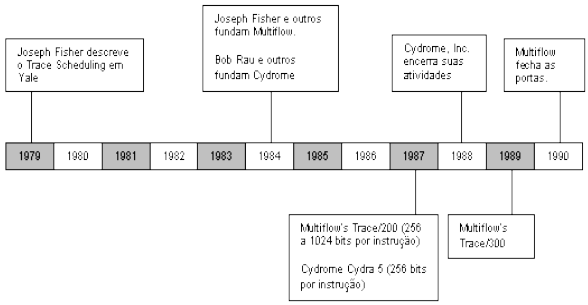

Figura 1: Linha do tempo do VLIW

Ambas as companhias, em 1987, lançaram computadores com a técnica VLIW. A Cydrome lançou o seu primeiro processador comercial, Cydra 5, com palavra de 256 bits este incluía suporte de hardware para uso da técnica de software com pipeline, e a Multiflow lançou o "Trace/200", com palavra de 256 bits , para uma execução com despacho de até 7 operações por ciclo de clock. Estas primeiras máquinas VLIW foram um fracasso comercial, ambas as empresas fecham, a Multiflow em 1990 e a Cydrome em 1988[1].

A Multiflow ainda antes de encerrar suas atividades ainda produziu sua segunda geração de máquinas VLIW, a série Trace/ 300 em 1989[1].

Alguns processadores que explorar a técnica VLIW que existem atualmente são: (Processador IA-64 ou Itanium da Intel, Processador Crusoe da Transmeta, Processador Trimedia da Philips, Processador TMS320C62x DSPs da Texas Instruments) ambos Comerciais e (Processador Playdoh dos Laboratórios HP, Processador Tinker da Universidade da Carolina do Norte, Processador de imagens Imagine em desenvolvimento na Universidade de Stanford) ambos Experimentais[6].

3.CARACTERISTICAS

Máquinas VLIW comum, tem palavras com centenas de bits, todas as unidades funcionais compartilham um grande banco de registradores comum (Figura 1) e as operações a serem executadas simultaneamente são sincronizadas em uma instrução VLIW[5].

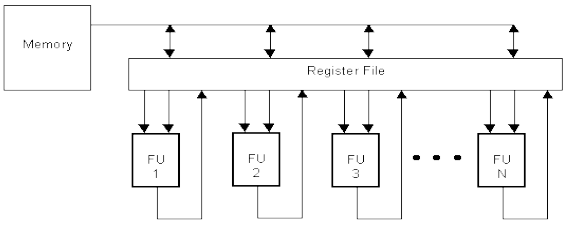

Figura 1: Arquitetura de uma máquina VLIW típica[5].

A tarefa principal da arquitetura VLIW é tentar reunir o máximo de instruções para executá-las em paralelo, alcançando assim o maior grau de paralelismo possível em nível de instrução. Um Compilador eficiente escalona instruções de forma que não gere conflitos estruturais e evite as dependências de dados e de nomes, razard, mantendo a semântica do programa. Cada instrução longa é formada por um conjunto de operações que podem ser executadas em paralelo, estas devem ser independentes entre si para que sejam executadas concorrentemente. Esta técnica reduz o número de instruções em comparação com os processadores escalares [2].

As instruções longas são montadas através de técnicas de escalonamento por software, aplicadas em tempo de compilação ou através da compactação do código gerado por um compilador convencional. Quanto maior a compactação, maior o desempenho. O compilador decide estaticamente, quais as operações serão emitidas simultaneamente formando assim a instrução longa. Para a execução de múltiplas operações com uma única instrução, é necessário que a instrução possua muitos bits para codificar as operações. Como ocorre no micro código horizontal, diferentes campos da instrução VLIW possuem *opcodes* que especificam as operações a serem executadas em diferentes unidades funcionais (figura 2)[5].

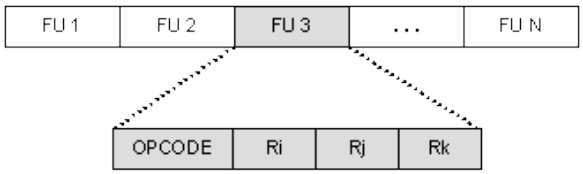

Figura 2: Palavra longa detalhada[5]

O compilador também é responsável por manter os recursos do computador ocupado, evitando assim perda de tempo/desempenho com recurso ocioso [4, 2].

A arquitetura VLIW exige um hardware menos complexo que o hardware dos processadores superescalares, onde as decisões das operações que serão executadas são feitas dinamicamente pelo hardware(em tempo de execução).

Existe pouca restrição no acesso aos recursos do processador, por isso possui grande liberdade quando escalona instrução, explorando assim muitos tipos de otimização, porém, a previsão é feita estaticamente e informações que temos no momento da

execução não são levadas em conta, isso pode nos levar a uma previsão incorreta em desvios condicionais e prejudicar o desempenho.

O compilador conhece todos os efeitos das operações sobre a arquitetura (escalonamento estático), exemplo disso é a latência de cada unidade funcional, este conhecimento permite que conflitos estruturais e de dados são resolvidos em tempo de compilação. Mecanismos de sincronização em tempo de execução são dispensados, por isso o fato do hardware ser bem mais simplificado. Uma técnica de escalonamento muito utilizada é o trace scheduling (escalonamento de traçado), que será discutida na seção 4.

3.1Vantagens da arquitetura VLIW em relação as máquinas superescalares

Mesmo com uma implementação que necessita de um hardware mais simples, o desempenho das máquinas VLIW é quase o mesmo que as máquinas superescalar RISC ou CISC. Possui hardware simples, pois é exigido do compilador o controle do paralelismo, neste caso o compilador de máquinas VLIWs apresentam uma complexidade e sofisticação muito maiores que os superescalares, a grande vantagem da complexidade e sofisticação ficar no compilador, é no desenvolvimento do software. Nas máquinas superescalares, cada melhoria feita, é necessário uma troca de hardware da máquina, e nos VLIW, alem do hardware ser mais barato, possui a vantagem de qualquer avanço no compilador, este pode ser aplicado a máquinas já existentes sem a necessidade de alteração do hardware.

Além desta grande vantagem das máquinas VLIW em relação as máquinas superescalares apresentada acima, ainda possui outras vantagens, o compilador analisa uma quantidade muito maior de instruções do que o hardware em um mesmo intervalo de tempo. A janela de instruções analisadas por hardware é limitada, se queremos aumentar esta quantidade em hardware, precisamos aumentar a lógica e a área do chip, por software é muito mais fácil e barato fazer isso e o paralelismo em software tem grande probabilidade de alcançar melhores resultados.

O compilador possui conhecimento do fonte do programa que esta processando, este pode conter informações importantes sobre o comportamento do programa, que pode ser usado para alcançar o máximo de ILP[5].

Além destas vantagens das máquinas VLIW em relação as máquinas superescalares apresentada acima, máquinas VLIW também podem imitar funções de um processador superescalar, exemplo disso é que se a máquina possui registradores suficientes, a máquina pode imitar os buffers de reordenação (reorder buffers), armazenando resultados de operações em registradores temporários, neste caso, a máquina pode ignorar estas execuções no caso de previsões erradas, ou salvar os resultados no caso de previsões corretas, melhorando assim o desempenho.

3.2Problemas e desvantagens da arquitetura VLIW

O modelo VLIW original possui diversos problemas, um dos problemas é o aumento do tamanho de código, a densidade de compactação depende do ILP, em trechos com pouco paralelismo,haverá um desperdício que causarão aumento das instruções. As instruções aumentam quando elas não estão completas, as unidades funcionais não usadas se convertem em bits desperdiçados na codificação da instrução, resumindo, não foi possível escalonar instruções suficientes para compor a instrução longa, isso provoca uma pior densidade de código e uma má utilização da memória. Além disso, é necessária uma grande quantidade de portas de acesso à cache de dados para suprir todas as unidades funcionais[5].

As primeiras VLIW operavam em lockstep, sem hardware de detecção de perigos, havia uma parada[1,2].

A compatibilidade de código binário é outro grande problema da arquitetura VLIW, isso acontece tanto com máquinas não paralelas, como também entre diferentes arquiteturas da família VLIW. Como uma arquitetura é diferente da outra, para cada arquitetura, é usado um compilador específico para melhorar o código, desse modo, consegue-se um melhor desempenho, porém diminui a compatibilidade entre códigos gerados entre os compiladores. Uma solução utilizada pela arquitetura IA-64 (Intel), é emulação do código objeto ou a conversão. [1][4][2].

Quando a execução de uma das instruções é bloqueada por falha ou interrupção, o processador VLIW deve considerar o pior caso, parando o pipeline. Eventos externos como falha no acesso à memória ou algum outro dispositivo podem afetar sua execução. Assim sendo, processadores com escalonamento dinâmico de instruções se adaptam melhor a essas condições.[5].

Previsões incorretas de caminhos tomados em desvios condicionais pode afetar consideravelmente sua performance. A previsão é feita estaticamente, informações importantes disponíveis em tempo de execução não são levadas em conta.

Todos estes problemas e as dificuldades de sana-los, levam ao escasso emprego das arquiteturas VLIW.

4.TRACE SHEDULING

Criado por Joseph A. Fisher em 1979, um dos pioneiro em VLIW. [6]. Trace Sheduling é uma técnica de compactação de micro-código que explora além dos blocos básicos, já que se constitui em uma técnica de compactação global.

Compactação de micro código é a conversão de micro código seqüencial em micro código paralelo e, portanto mais eficiente.

Compactação de código pode ser realizada de duas maneiras: localmente, onde atua sobre o domínio de blocos básicos, ou global, que este atua sobre trechos de códigos que extrapolam os limites dos blocos básicos[4].

Inicialmente utilizado para compactação de micro-programas em micro-instruções horizontais, com o tempo, esta técnica se tornou a principal técnica de compactação de códigos para máquinas VLIW[5].

Resumidamente, o funcionamento do modelo, consiste basicamente em tomar o trace com probabilidade maior de ser executado dentre os traces não compactados. Na tentativa de escalonar as operações do trace no menor número de palavras longas possíveis, ele cria um grafo que contém todas as operações a serem escalonadas, permitindo também a troca de ordem das instruções durante o processo. A ordem dos branches não são alteradas e é inserido código de compensação na tentativa de corrigir saltos para fora do trace e rejoins dentro do trace[4].

5.SISTEMAS EMBARCADOS COM VLIW

Quando pensamos em sistemas embarcados, os chips da Crusoe e Trimédia, podem ser interessantes para aplicação dos conceitos de VLIW clássicos.

Para o mercado mais simples, de baixo processamento (PCs e aparelhos móveis para conexão com a internet), temos o processador VLIW Crusoe. Maiores de talhes do processador Crusoe, encontramos na seção 6.2 deste artigo ou em [1].

A arquitetura da Trimédia, TM32 é uma arquitetura VLIW clássica. As instruções possuem cinco operações cada, possui escalonamento estático, não possui detecção de hazard (nem perigos de dados nem de nomes), possui mecanismo de compactação de instruções na memória e possui cachê de instrução.

Vimos no item 3.2 que o aumento do tamanho de código nas máquinas VLIW, é uma das desvantagens dessa arquitetura, que seria inaceitável nos sistemas embarcados, a arquitetura Trimédia, resolve este problema.

6.ALGUNS PROCESSADORES VLIW

6.1Itanium IA64

Itanium ficou disponível no mercado em meados de 2001, possui um clock de 800MHz, capaz de emitir até seis instruções cada vez, com até 3 instruções de desvio e duas instruções de referencia a memória. A janela de emissão de instruções, possui até dois pacotes em qualquer instante de dado [1].

O processador Itanium é a primeira implementação da arquitetura Intel IA-64, esta arquitetura é um conjunto de instruções de registradores no estilo RISC com suporte a ILP baseada no software do compilador.

Podemos ver que o Itanium possui muitos recursos associados com os pipelines com escalonamento dinâmico, possui forte ênfase em previsões de desvios, renomeamento de registradores, um pipeline profundo com muitas fases antes da execução para manipular as instruções de maneira eficiente, e diversas fases pós execução para tratar detecção de exceções.

Existem 9 unidades funcionais (duas I(Imediato), duas M(memória), duas F(Aritméticas) e três B(Desvio)) e um pipeline de 10 fazes que se dividem em 4 fazes principais[1]:

- Front - End: Pré busca, máx 32 bytes por clock, pode conter ate 8 pacotes, 24 instruções
- Entrega de instrução: distribui até seis instruções às nove unidade funcionais
- Entrega de operando: Acessa o arquivo de registradores, executa a derivação, atualiza informações de registradores e verifica dependências de predicados.
- Execução: Executa as instruções através da ULA e unidades de armazenamento, detecta exceções e publica NaTs, retira instruções e executa a gravação de retorno.

IA-64 é uma ISA (*Instruction Set Architectute*) no estilo VLIW denominado pela Intel e pela HP como EPIC (Explicitly Parallel Instruction Computing). [1]

Os registradores existentes no IA-64 dentre outros são 128 registradores de uso geral de 64 bits, 128 registradores de ponto flutuante de 82 bits cada, 64 registradores de predicados de um bit cada, 8 registradores de desvio de 64 bits. Existem também outros registradores usados para controle de sistema, mapeamento de memória, contadores de desempenho e comunicação com o SO[1].

As instruções do IA-64 são montadas pelo compilador em pacotes, estes pacotes possuem 128 bits de largura, cada pacote possui um campo de "gabarito" de 5 bits e 3 instruções com 41 bits de comprimento cada uma (Figura 3).

Figura 3, - Palavra de instrução do IA64

O IA-64 suporta **especulação de dados** e de controle, que são controlada pelo software.

Para realizar especulação de controle, o compilador move os "loads" para antes do desvio que o controla e o "load" é então marcado como especulativo, qualquer exceção neste load especulativo, o processador não sinalizará.

Caso o desvio de controle for tomado posteriormente, o compilador utiliza uma operação especial check.s para verificar se a exceção ocorreu, desviando então para uma rotina de exceção, quando for o caso, e o resultado daquelas instruções que foram executadas erradamente, são descartados.

Para realizar a especulação de dados, é utilizado o "load avançado" (um tipo especial de "load"). Caso o compilador não conseguir verificar se pode passar um load na frente de um store, o "load avançado" é utilizado.

O processador usa um estruturas chamada ALAT para verificar se o store realizado posteriormente escreveu na mesma posição lida pelo "load avançado". O compilador usa também uma operação especial que verifica se o store invalidou o "load avançado".

Para compatibilidade com a família Pentium, uma unidade de controle e decodificação especial para instruções do IA-32 está presente no Itanium.

Na tentativa de aumentar o desempenho, do processador IA-64, ele incluiu também diversas facilidades que não são encontradas em arquiteturas VLIW tradicionais, tornando-se assim o processador VLIW mais complexo já projetado. Com isso temos um impasse, já que a arquitetura VLIW tem como objetivo hardware simples para o compilador[6].

6.2Processador Transmeta Crusoe

Crusoe é um processador com tecnologia VLIW projetado para mercado de baixa potência, exemplos são os PCs e aparelhos móveis para conexão com a internet

Ele utiliza pipeline simples de 6 fases para inteiro, e um pipeline de 10 fases para ponto flutuante. Possue 2 tamanhos de instruções, 64 bits para duas operações e 128 bits para 4 operações.

Crusoe é compatível com instruções X86, ele possui um Software que é responsável pela conversão de instruções X86 em VLIW, utiliza varias técnicas para esta conversão. Possui também 5 tipos de slots de operações (Memória, Desvio, Cálculo, Imediato e operações da ULA). Maiores informações em [1].

6.3Multiflow TRACE

Os principais objetivos perseguidos no projeto do processador Multiflow TRACE foram [5]

- Projeto modular, possuindo um número de unidades funcionais (FU) expansível.
- Uso de uma alta quantidade de componentes eletrônicos de baixo custo;
- Uso de memórias DRAM convencionais obtendo uma memória principal de alta capacidade com um custo baixo.
- Alta performance para processamento de ponto flutuante de 64 bits.
- Adequado ao ambiente multi-usuário.

O processador tem seu núcleo dividido em uma unidade de inteiros e uma de ponto flutuante, como na figura 4.

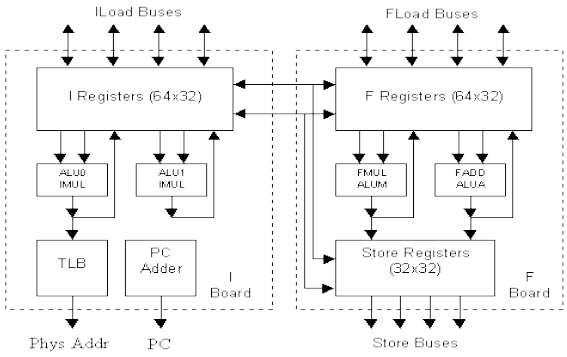

Figura 4 – Exemplo de processador Multiflow TRACE

Possui bancos de registradores independentes para cada uma das duas unidades (64 registradores de 32 bits), pois dificilmente é realizada operações de ponto flutuante sobre dados inteiros e vice-versa, enquanto a idéia é realizar operações sobre dados inteiros e cálculos de endereços em paralelo com operações de ponto flutuante.

Pode ser manipulado quatro escritas, quatro leituras e efetuar o *bypassing,* (modelo muito utilizado em processadores RISC para resolver dependência de dados*)* de cada porta de escrita para qualquer porta de leitura[5].

O conjunto de instruções RISC-like sobre inteiros contém opcodes dedicados à operações aritméticas, lógicas e de comparação.

Existem outras primitivas de alto desempenho para operações de multiplicação de 32 bits e operações de shift, merge e extract para a manipulação de campos de bits e bytes[5].

As operações de adição de ponto flutuante têm sobre as unidades funcionais dessa máquina uma latência de 6 ciclos, em contraste com a latência simples das unidades inteiras. Multiplicações tomam 7 ciclos e as divisões 25 ciclos na FMUL ALUM[5]

Os barramentos ILOAD Buses, FLOAD Buses e STORE Buses são conjuntos de quatro barramentos de 32 bits gerenciados independentemente e de forma síncrona. são barramentos simples, baratos e rápidos[5]

As caches são fisicamente distribuídas e virtualmente endereçadas e podem chegar a 1Mb de capacidade total[5].

Usa o esquema compressed encoding para otimizar o uso da memória principal e uncompressed encoding na cache de instruções[5].

6.4Processador Cydra 5

É um sistema multiprocessador heterogêneo. Cydra5 era usado por pequenos grupos de cientistas e engenheiros e que buscava conquistar o mercado de mini-supercomputadores.

Produzido pela Cydrome(1984-1990), os objetivos principais dessa arquitetura eram[5]:

- Baixo custo;
- Alto desempenho em aplicações numéricas.
- Utilização também em aplicações não numéricas;
- Transparência ao usuário, para que não necessite fazer outro treinamento ou mudar seus programas em detrimento de ser um sistema heterogêneo.

Utilizava a arquitetura directed-dataflow, porém, diversos conceitos empregados para suportá-la são VLIW. Possui a capacidade de despacho de múltiplas operações por ciclo e palavras de instrução de 32 bytes.

No processamento numérico podia despachar até seis operações por ciclo mais uma adicional para controlar a unidade de instruções e outras operações diversas.

Possuía um recurso que permitia a especificação de uma única instrução cada vez, assim, quando o paralelismo era bem escasso, evitava desperdício na codificação de NOPs.

Os projetistas decidiram programar unidades funcionais pipelinezadas e que mesclam diversas operações. Essas unidades são[5]:

- Unidade de adição de ponto flutuante/ULA de inteiros.
- Multiplicador/divisor/sqrt de inteiros e ponto flutuante.
- Unidade de acesso à memória 1.
- Unidade de acesso à memória 2.
- Unidade de cálculo de endereço 1/bit reverse.
- Unidade de cálculo de endereço 2/multiplicação de inteiros.
- Uma unidade chamada "Instruction and Misc. reg. unit".

Para bechmarks menos vetorizáveis, como o ITPack, o Cydra 5 chegava a atingir metade do desempenho do Cray X-MP [7].

7.Conclusão

Apresentados ao longo deste artigo diversos fatores relevantes à organização do hardware e suporte de compilação para tornar o uso da arquitetura VLIW viável.

Percebemos que essa arquitetura foi pouco explorada, pelos diversos problemas e restrições apresentadas no uso de palavras longas de instrução (seção 3.2). Muitas pesquisas foram realizadas ao longo dos anos para tornar os benefícios das arquiteturas VLIW apreciáveis e vantajosas.

Com o sucesso das máquinas superescalares, arquitetos programaram máquinas altamente complexas e com recursos cada vez mais sofisticados. Entretanto, haverá um limite para evolução de mecanismos, em detrimento do grau de complexidade a ser suportado. Arquiteturas VLIW, em contrapartida, mantém grande parte da complexidade em nível de software, obtendo um hardware relativamente simples[5].

A organização dessa arquitetura, embora relativamente simples, traz consigo uma série de implicâncias em relação a mecanismos de busca de instruções, requisitos do sistema de memória, técnicas de compilação, dentre outras. Esses requisitos diferenciados em relação às arquiteturas convencionais abrem um vasto campo de novas possibilidades e oportunidades a serem exploradas, sendo em termos de pesquisa uma área altamente promissora [5].

Ambas as abordagens têm suas vantagens e desvantagens, talvez, a melhor solução esteja em uma arquitetura mista, que utilizaria conceitos de VLIW, e mecanismos mais poderosos das máquinas superescalares, que poderiam resolver diversos problemas vistos da arquitetura VLIW. Atualmente pouco se fala em VLIW, porém, deve ser ressaltado o fato de que muitas pesquisas ainda podem ser realizadas na tentativa de tornar a arquitetura VLIW não só uma mera "arquitetura conceito" mas que, de fato, detenha uma parcela do mercado e torne-se uma opção a mais para suprir as necessidades de processamento de alto desempenho.

8.REFERENCIAS BIBLIOGRÁFICAS

[1] Hennessy, John L. and Patterson, David A. *Arquitetura de Computadores: Uma Abordagem Quantitativa*. Editora Campus, Rio de Janeiro,2003, pág. 229 – 233, 255 – 270.

[2] Philippe Navaux. Notas de Aula. Arquiteturas Superescalares, Arquiteturas Avançadas. Universidade Federal do Rio Grande do Sul (UFRGS)

[3] Leonardo Taglietti, Flavio Rech - Modelagem de Arquiteturas VLIW usando uma ADL - Universidade Federal do Rio Grande do Sul – (UFRGS)

[4] Perdigueiro, Júlia M, Borili, Tatiane B. – Paralelismo em nível de instrução: Arquiteturas VLIW – Instituto de Computação, Universidade Estadual de Campinas - UNICAMP

[5] Akira, Ito, S. *Arquiteturas VLIW: Uma Alternativa para Exploração de Paralelismo a Nível de Instrução*. - 1998. (UFRGS).
http://www.inf.ufrgs.br/procpar/disc/cmp134/trabs/T1/981/VLIW/vliw10.html(acesso03/05/2008).

[6] Silva, Gabriel P. Arquiteturas VLIW, notas de aula, Informática DCC/IM Universidade Federal do Rio de Janeiro (UFRJ)

[7] RAU, B. Ramakrishna et al. The Cydra 5 Departamental Supercomputer : Design Philosophies, Decisions, and Trade-offs. IEEE Computer, v.22, n.1, Jan. 1989. p.12-32.