Trabalho 3 – Arquiteturas de Processadores que exploram o Paralelismo

Mateus Brugnaroto[1], Vinícius Renato Rocha Geraldo[2]
Universidade Federal de Pelotas - UFPel
mbrugnaroto[1], vrrgeraldo[2] {@inf.ufpel.edu.br}

RESUMO

Esse trabalho é realizado a demonstração de vários tipos de arquitetura de processadores que exploram o paralelismo os quais são VLIW (Very Long Instruction Words), SIMD (Single Instruction, Multiple Data) e Superescalar. Onde foi explorado cada topologia para ver como funciona cada.

1. *ARQUITETURA SIMD - Conjunto Único de Instruções, Múltiplos Conjuntos de Dados*

Os computadores SIMD operam sobre vetores de dados. Por exemplo, quando uma única instrução SIMD soma 64 números, o hardware da máquina SIMD envia 64 conjuntos de dados para 64 ULAs, obtendo as 64 somas dentro de um único ciclo de clock. Com isso explorando a ideia de estrutura de dados para utilizar vetores e matrizes de realizar as instruções

A grande vantagem das máquinas SIMD é o fato de as unidades de execução paralela estarem sincronizadas, todas elas respondendo ao comando de uma única instrução, originada de um único *program counter* (PC). Apesar de nas máquinas SIMD cada uma das unidades executar a mesma instrução, cada uma dessas unidades tem seu próprio registrador de endereços, podendo, portanto, cada uma delas opera sobre dados armazenados em endereços diferentes.

A motivação original por trás das máquinas SIMD era amortizar o custo do desenvolvimento da unidade de controle por dezenas de unidade de execução. Outra vantagem é o tamanho reduzido da memória para programas. A técnica da memória virtual e o custo reduzido das memórias DRAM reduziram substancialmente a importância dessa vantagem.

As arquiteturas realmente SIMD têm em seu conjunto de instruções tanto instruções SISD (Conjunto único de instruções, Conjunto único de dados) quanto SIMD. Nessas máquinas é como se houvesse um hospedeiro SISD para realizar as operações sequenciais como desvios e cálculos de endereços. As instruções SIMD são distribuídas em broadcast para todas as unidades de execução, cada uma com seu conjunto particular de registradores. As unidades de execução se apoiam em redes de conexão para realizar a troca de dados com os demais componentes do sistema.

As arquiteturas SIMD funcionam melhor quando tratam com arrays e loops *for*. Portanto, para que possamos aproveitar ao máximo as perspectivas de paralelismo em máquinas SIMD, deve haver muitos conjuntos de dados a serem operados, ou paralelismo no nível dos dados. As máquinas SIMD experimentam seu pior momento quando da execução de comandos *case* ou *switch*, em que cada unidade de execução precisa executar uma operação diferente sobre seus dados, operação essa que depende do valor de um parâmetro da instrução. Algumas das unidades de execução devem ser desabilitadas, e somente aquelas com os dados a serem processados devem ser autorizadas a continuar a execução.

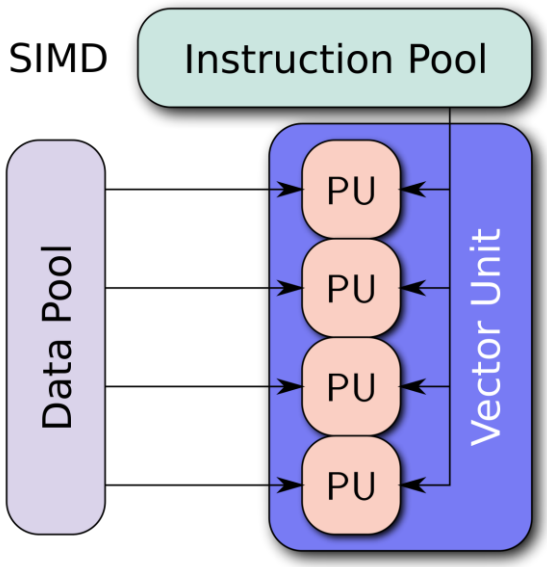

Figura 1 – SIMD (Conjunto Único de Intstruções, Múltiplos Conjuntos de Dados)

2. *ARQUITETURA SUPERESCALAR*

Um processador superescalar é aquele no qual são usadas várias *pipelines* de instrução independentes. Cada pipeline tem diversos estágios, podendo manipular várias instruções a cada instante. O uso de várias *pipelines* introduz um novo nível de paralelismo, possibilitando processar diversos fluxos de instrução de cada vez. Um processador superescalar explora o que é conhecido como paralelismo no nível de instruções, que diz respeito ao nível em que instruções de um programa podem ser executadas em paralelo.

Um processador superescalar busca tipicamente várias instruções de cada vez, e tenta encontrar instruções próximas que sejam independentes umas das outras e que podem, portanto, ser executadas em paralelo. Se os dados de entrada de uma instrução dependem da saída produzida por uma instrução precedente, a execução dessa instrução não pode ser completada antes ou ao mesmo tempo que a execução da instrução da qual ela depende. Uma vez que essas dependências de dados sejam identificadas, o processador pode executar e completar instruções em uma ordem diferente da que ocorre no código de máquina original.

O processador pode eliminar dependências desnecessárias por meio do uso de registradores adicionais e de renomeação de referências a registradores no código original.

Uma implementação superescalar de uma arquitetura de processador é uma implementação na qual instruções usuais (aritmética de números inteiros e de números de ponto flutuante, instruções de carga e armazenamento e instruções de desvios) podem ser iniciadas simultaneamente e executadas independentemente. Tais implementações suscitam diversas questões de projeto bastante complexas, relacionadas à *pipeline* de instruções.

O projeto superescalar apareceu logo depois dos projetos de arquitetura RISC. Embora as técnicas de arquitetura superescalar possam ser aplicadas mais diretamente a uma arquitetura RISC com um conjunto simplificado de instruções, a abordagem superescalar também pode ser usada em arquiteturas CISC.

A essência da abordagem superescalar é a habilidade de executar instruções independentemente, em diferentes *pipelines*. Esse conceito pode ser ainda mais explorado, permitindo que instruções sejam executadas em uma ordem diferente da ordem em que aparecem no programa.

Superescalar x Superpipeline x Pipeline

Podemos observar que as diferentes arquiteturas entre si considerando o nível de execução das instruções sendo processada ao mesmo tempo e sendo muito útil para algumas aplicações de dados como imagens e outros tipos de dados.

Na Figura 2, ilustra a execução de 3 tipos de arquiteturas que utilizam do pipeline para suas instruções de dados. Na primeira parte do desenho observamos o uso do pipeline comum para qualquer arquitetura onde explora paralelismo no qual emitem uma instrução por ciclo de relógio e pode completar apenas um estágio de *pipeline*, onde é dividida em quatro estágios (busca da instrução, decodificação da instrução, execução da operação e escrita de resultado). Já num momento seguinte podemos ver a ideia de funcionamento do *superpipeline* onde é capaz de completar dois estágios de *pipeline* por ciclo de relógio. Uma forma para observar isso é considerar as operações realizadas em cada estágio como divididas em duas partes não sobrepostas, cada qual podendo ser executada em meio ciclo de relógio. E por último vemos a execução de uma arquitetura superescalar, capaz de executar duas instâncias de cada estágio em paralelo visando o desempenho da execução de instruções escalares.

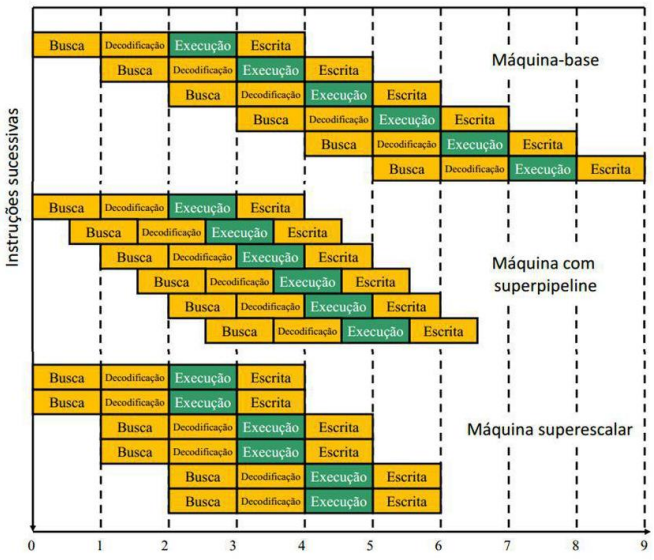

Figura 2 – Comparação entre as arquiteturas abordadas superescalar e superpipeline

3. *ARQUITETURA VLIW – Instrução de Palavra Muito Grande*

A arquitetura *Very Long Instruction Word* (VLIW) tenta alcançar maiores níveis de paralelismo de instrução pela execução de instruções longas compostas por múltiplas operações. 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, nisso muita informação do hardware é visível ao compilador no momento de gerar as instruções paralelas. O hardware confia no paralelismo gerado pelo compilador, portanto, ele não verificar se contém dependências entre as instruções.

A ideia do VLIW, portanto, se entende em infundir paralelismo em toda a arquitetura, estendendo também o compilador como parte do mesmo. Com isso muitos problemas passam a ser resolvidos na compilação.

O processador VLIW executa o conjunto de operações concorrentemente, atingindo assim um alto grau de paralelismo no nível de instrução. É responsabilidade do compilador escalonar as operações de modo a utilizar o melhor possível das unidades funcionais disponíveis no processador.

As instruções longas são feitas através de modos de escalonamento por “software” aplicadas em tempo de compilação ou através da compactação do código gerado por um compilador convencional sendo responsável por entregar as instruções de certa forma que a arquitetura seja aproveitada ao máximo. A arquitetura é organizada com múltiplas unidades funcionais e banco de registradores com múltiplas portas de leitura.

Comparações entre VLIW x Superescalar

A arquitetura VLIW tem bastante semelhanças com o modelo RISC (Instruções simples e rápidas). No entanto, muitos afirmam que o VLIW é apenas uma implementação da arquitetura RISC, pois usa os mesmos conjuntos de instruções simples, além de utilizar-se de técnicas de pipeline.

As diferenças se apontam 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 registradores para suportar às várias operações contidas na instrução.

A maior peculiaridade no VLIW, porém, é o desenvolvimento conjunto de maneiras de compilação que sejam cooperadas 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 a arquiteturas que buscam maior o fato de paralelismo.

Na Figura 3 e 4 mostra a comparação entre essas duas arquiteturas de VLIW e superescalar em relação a instrução como é realizada em cada tipo.

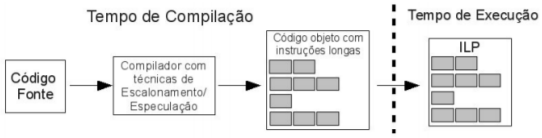

Figura 3 – Esquemático VLIW para como é alcançada a execução

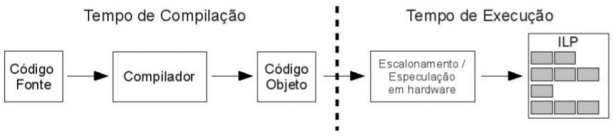

Figura 4 – Esquemático Superescalar para como é alcançada a execução

Pode-se observar a nítida diferença de um hardware para realizar as técnicas de escalonamento/especulação numa arquitetura superescalar, onde numa arquitetura VLIW esse esquemático se ponta dentro de um compilador para realizar o escalonamento da instrução para execução

Numa comparação em relação aos tipos de instruções arquiteturais existentes (RISC, CISC, VLIW) podemos analisar o tamanho da instrução, formato da instrução, semântica da instrução e os registradores utilizados para cada um dos seus tipos mostrados na Tabela 1.

Característica Arquitetural	CISC	RISC	VLIW
Tamanho instrução	variável	tamanho único (32 bits)	Tamanho único
Formato instrução	campos variáveis	regular	regular
Semântica instrução	varia de simples a complexa	operação simples	muitas simples independentes
Registradores	poucos, específicos	muitos, propósito geral	muitos, propósito geral

Tabela 1 – Comparativo entre as arquiteturas CISC, RISC e VLIW

Referências

PATTERSON, David A.; HENNESSY, John L. Organização e Projeto de Computadores: a interface hardware/software. 2a.ed. Rio de Janeiro: Campus, 2005

STALLINGS, William. Arquitetura e Organização de Computadores. 5a.ed. São Paulo: Prentice-Hall, 2002. ISBN: 85-87918-53-2

http://www.ic.unicamp.br/~ducatte/mo401/1s2009/T2/036272-t2.pdf - VLIW

Slides do prof. Júlio Carlos Balzano de Mattos