Hazards: conflitos no Pipeline

Marco A. S. Segundo
marcoant.segundo@gmail.com

Laura A. L. Fidelis
laurafideliss@gmail.com

Igor José M. Barbosa
igorjose2@gmail.com

Camile C. Lacerda
camilcampos@outlook.com

***Resumo*— Desde a criação do primeiro processador foi visto que era necessário um desenvolvimento tecnológico para melhorar o seu desempenho. Começando a partir do Intel 4004 até os de 2014, foram feitos muitos avanços. E o principal foco desse artigo é tratar de um problema que foi trabalhado desde a criação dos pipelines. Nesse documento será descrita a técnica de stalls que garante uma forma de tratar esse problema, que atrapalha bastante esse tipo de processador. Por fim, garantindo o bom uso do pipeline para que o choque de dados ocorra minimamente.**

Palavras-chave— Hazards, Pipeline, Desempenho, Técnicas.

I. INTRODUÇÃO

Para melhorar o desempenho dos processadores, a Intel resolveu usar o pipeline, uma técnica inicialmente usada em processadores RISC, que consiste em dividir o processador em vários estágios distintos. Foi convencionado em utilizar 5 níveis, ou seja, 5 estágios.

Quando é carregada uma nova instrução, ela primeiramente passa pelo primeiro estágio, que trabalha nela durante apenas um ciclo de clock, passando-a adiante para o segundo estágio. A instrução continua então sendo processada sucessivamente pelo segundo, terceiro, quarto e quinto estágios do processador. A vantagem desta técnica, é que o primeiro estágio não precisa ficar esperando a instrução passar por todos os demais para carregar a próxima, e sim carregar uma nova instrução assim que termina a primeira, ou seja, depois do primeiro pulso de clock.

As instruções trafegam dentro do processador na ordem em que são processadas. Mesmo que a instrução já tenha sido processada ao passar pelo primeiro ou segundo estágio, terá que continuar seu caminho e passar por todos os demais. Se por acaso a instrução não tenha sido completada mesmo após passar pelos cinco estágios, voltará para o primeiro e será novamente processada, até que tenha sido concluída. Assim, é possível que o processador seja capaz de processar simultaneamente, em um único ciclo de clock, várias instruções que normalmente demorariam vários ciclos para serem processadas. Mas essa técnica pode acarretar em outro tipo de problema, o conflito de dados, chamados de Hazards.

Hazards são problemas com a instrução da CPU, quando a próxima instrução não pode ser executada no ciclo de clock seguinte e pode, potencialmente, levar a resultados incorretos. Ocorrem quando o pipeline, ou alguma parte dela, precisa parar porque as condições não permitem a execução contínua.

II. TIPOS DE HAZARD

Hazards ocorrem por conta da dependência natural da sequência de código, quando uma instrução depende do resultado da anterior que ainda está sendo executada, ocasionando uma parada de execução no código, cuja ordem natural é alterada dentro de um pipeline.

Existem três tipos de hazard de pipeline:

1. Estrutural

Acontece quando duas ou mais instruções que já estão no pipeline precisam do mesmo recurso. O resultado é que as instruções precisam ser executadas em série em vez de paralelo em uma parte do pipeline.

2. Dados

Ocorre quando há conflito no acesso de uma posição de operando. Ou seja, duas instruções em um mesmo programa estão para ser executadas na sequência e ambas acessam um determinado operando de memória ou registrador. Contudo, se as instruções são executadas em um pipeline, então é possível que o valor do operando seja atualizado de tal forma que produza um resultado diferente do que seria com uma execução estritamente sequencial. Pode-se dizer que o programa produz um resultado incorreto por causa do pipelining. O hazard de dados ainda possui três categorias:

Read-after-write(RAW): o hazard ocorre quando a operação de leitura acontece antes da escrita ter sido completada.

Write-after-read(WAR): o hazard acontece se a operação de escrita é completada antes da operação de leitura.

Write-after-write(WAW): o hazard ocorre se as operações de escrita acontecem na ordem inversa da esperada.

3. Controle

Também conhecido com hazard de desvio, acontece quando o pipeline precisa desviar por conta de um comando de instrução.

Para tratar o hazard de uma maneira geral é utilizada uma técnica denominada de pipeline stall. Esta técnica consiste em a unidade lógica de controle receber as instruções, e verificar se um hazard poderá ocorrer. Se sim, a unidade lógica de controle insere NOPs no pipeline. Assim, antes que a próxima instrução (que causaria o hazard) seja executada, a anterior já terá tido tempo suficiente para finalizar e evitar o hazard. Porém, essa técnica causa uma queda no desempenho do pipeline, por que o tempo de latência da instrução aumenta, e com isso, aumenta o tempo de execução do processador.

Para solucionar o hazard de dados também existe a técnica de *Forwarding* (Adiantamento), que consiste em adiantar os dados necessários para a próxima instrução, para que não seja necessário esperar que esses dados sejam armazenados em determinado registrador. Porém, os adiantamentos não são sempre eficientes, já que em determinados casos, são necessário dois ciclos de stalls.

Já para solucionar o de controle, além da técnica de pipeline stall, se tem a técnica de predição de desvio, em que se busca prevê a saída da instrução de desvio. Existem dois tipos de predição: a dinâmica, que se baseia no comportamento recente de cada operação de desvio, e a estática, que se baseia no comportamento típico das instruções de desvio.

III. Conclusão

Tratar os hazards traz benefícios ao pipeline, pois tenta solucionar um grande problema encontrado desde sua criação. Como vimos no artigo, a pipeline stall é uma alternativa em comum para a eliminação dos hazards de controle e dados.

Portanto, é possível perceber que mesmo com toda a vantagem de usar pipeline nos processadores atuais, ainda existem barreiras que precisam ser ultrapassadas, que hoje subsistem através de conflitos, mais conhecidos como hazards.

Referências

[1] "Pipelining" (http://www2.it.lut.fi/kurssit/05-06/Ti5317000/seminars/pipelining%20paper.pdf). Acessado em 06 de Novembro de 2014.

[2] "Organização e Projeto de Computadores – A interface Hardware/Software. Patterson David and Hennessy John, 4ª Edição. Elsevier.

[3] "Organização Estruturada de Computadores" Tanembaum, Andrew S. 1992. – Cap 5.

[4] "Microcontroladores e Interfaces" 2004/2005 - Carlos A. Silva.

[5] "Computers as Components: Principles of Embedded Computer Systems Design". Wolf, W. Academic Press, San Diego, CA, 2001. ISBN 1-55860-693-9.

[6] “A Critical Review of Stall Control Techniques in Industrial Fans” Stefano Bianchi, Alessandro Corsini, Anthony G. Sheard, and Cecilia Tortora, ISRN Mechanical Engineering, vol. 2013, Article ID 526192, 18 pages, 2013. doi:10.1155/2013/526192