Aula 10: Melhora da performance usando Pipeline

Pipeline é uma técnica de implementação de processadores que permite a sobreposição temporal das diversas fases da execução de instruções. Tradicionalmente, as instruções do MIPS são executadas em ate 5 passos:

1. Busca da instrução na memória;
2. Leitura dos registradores enquanto uma instrução é decodificada, (o formato das instruções do MIPS permite que a leitura e a codificação ocorram simultaneamente);
3. Execução de uma operação ou cálculo de um endereço;
4. Acesso a um operando de memória;
5. Escrita do resultado em um registrador;

Portanto, o pipeline do MIPS tem cinco estágios.

Performance do processador monociclo versus performance do processador pipeline

Em condições ideais, o ganho devido ao pipeline é igual ao numero de estágios do pipe. Entretanto, por razões de balanceamento imperfeito (estágios gastam mesmo tempo) e overhead do pipeline (representado pelo tempo que se gaste para “encher” todo o pipe), o ganho será menor que o numero de estágios do pipeline.

Em suma, o pipeline melhora a performance por meio do aumento do troughput das instruções, ou seja, aumentando o número de instruções por unidade de tempo, e não por meio da diminuição de execução de uma instrução individual.

Projeto do conjunto de instruções para execução em pipeline

Todas as instruções do MIPS têm o mesmo tamanho que torna simples a busca das instruções e sua decodificação. O MIPS tem poucos formatos de instruções, sempre com o registrador-fonte localizado na mesma posição. Essa simetria permite que seja realizada a leitura do banco de registradores ao mesmo tempo em que o hardware está determinando o tipo de instrução.

Na arquitetura do MIPS só as instruções de load Word ou store Word é que manipulam operandos na memória, permitindo usar o estágio de execução para efetuar o cálculo do endereço de memória, sendo o acesso à memória realizado no próximo estagio. Os operandos do MIPS precisam estar alinhados na memória.

Conflitos do Pipeline

Existem situações em que a instrução seguinte não pode ser executada no próximo ciclo de clock.

Conflitos estruturais

Significa que o hardware não pode suportar a comunicação de instruções que o pipeline deseja executar no mesmo ciclo de clock. No caso do MIPS, suponha que só tivéssemos uma única memória, em vez de duas (instruções e dados). No pipeline a primeira instrução estaria acessando um dado da memória e a quarta estaria sendo buscada na mesma memória, no mesmo ciclo de clock.

Conflitos de Controle

Se origina da necessidade de se tomar uma decisão com base nos resultados de uma instrução, enquanto outras estão sendo executadas. Parada do Pipeline = bolha (inserção de uma instrução NOP).

Soluções

Parada: Ocorre uma perda de performance muito alta. Torna-se necessário inserir no pipeline um estagio de parada, ou bolha após o desvio condicional.

Predição: Um esquema simples é sempre predizer que os desvios condicionais vão falhar. Quando você estiver certo, o pipeline prossegue na velocidade máxima. Somente quando os desvios se realizarem é que há necessidade de se atrasar o avanço normal das instruções por meio do pipeline.

Conflitos por dados

A execução de uma instrução depende do resultado da outra, que ainda está no pipeline.

Ex: Uma instrução de soma imediatamente seguida por uma de subtração que utilize o resultado da soma.

Add $s0, $t0, $t1

Sub $t2, $ s0, $t3

Caminho de dados em pipeline

O caminho de dados é separado em cinco partes:

1. BI: busca da instrução;

2. DI: decodificação da instrução e leitura do banco de registradores;

3. EX: execução ou cálculo o endereço;

4. MEM: acesso à memória de dados;

5. ER: escrita no banco de registradores.

As instruções e dados se movimentam geralmente da esquerda para a direita através de cada um dos cinco estágios. Existem, no entanto, duas exceções para este fluxo de instruções da esquerda para a direita:

- O Estágio de escrita no banco de registradores, que coloca o resultado em um dos registradores do banco, que está fisicamente situado no meio do caminho de dados.

- A seleção do próximo valor do PC, escolhido entre o PC incrementado e o endereço de desvio condicional que vem do estágio MEM.

Para reter o valor de determinada instrução durante seus outros quatro estágios, é necessário salvar em um registrador o valor lido da memória de instruções; portanto, é necessário que se coloque um registrador em cada uma das fronteiras entre os estágios de pipeline.

load Word (lw)

1. Busca de instrução: A instrução é lida da memória, usando, para isso, o endereço armazenado no PC. O resultado da leitura é armazenado no registrador BI/DI. O endereço do PC é incrementado e armazenado de volta no PC. Esse valor do PC incrementado também é guardado em BI/DI;

2. Decodificação da instrução e leitura do banco de registradores: Os 16 bits menos significativos da instrução são colocados na entrada do módulo de extensão de sinal, além dos números dos registradores a serem lido sendo colocados nas entradas correspondentes do banco de registradores. Todos os três valores resultantes dessas operações são armazenados no registrador DI/EX, junto com o valor incrementado do PC, que veio do estágio anterior.

3. Execução ou cálculo do endereço: O conteúdo do registrador 1 é lido do registrador DI/EX e o resultado da extensão de sinal também, adicionando-os, usando a ULA. O resultado dessa soma é colocado no registrador EX/MEM.

4. Acesso à memória: A memória de dados é lida a partir do endereço armazenado em EX/MEM e o dado resultante é carregado no registrador MEM/ER.

5. Escrita no banco de registradores: O dado lido do registrador MEM/EX é escrito no banco de registradores.

Controle do Processador Pipeline

O PC é escrito em cada um dos ciclos de clock, de modo que não há necessidade de se ter um sinal de controle para a escrita nele. Da mesma forma, também não há necessidade de sinais de escrita para os registradores do pipeline.

1. Busca de instruções: os sinais de controle para ler a memória de instruções e para escrever no PC estão sempre ativos.

2. Decodificação da instrução / Leitura do banco de registradores: não há uma linha de controle adicional a ser ativada.

3. Execução / Cálculo do endereço: os sinais a serem considerados são RegDst, ULAOp, ULAFonte. Tais sinais selecionam o registrador no qual o dado será escrito, a operação da ULA e a segunda entrada da ULA, entre o dado presente na saída 2 do banco de registradores e o resultado da extensão do sinal do campo imediato da instrução.

4. Acesso à memória: as linhas de controle a serem consideradas neste estágio são DvC, LerMem e EscMem. Tais sinais são necessários às instruções de desvio se igual, *load word* e *store Word,* respectivamente. O sinal FontePC seleciona o próximo endereço na seqüencia de execução, a não ser que o controle tenha ativado o sinal DvC e que o resultado da ULA seja zero.

5. Escrita no banco de registradores: as duas linhas de controle a serem consideradas neste estágio são MemparaReg, que decide entre enviar para o banco de registradores o resultado da ULA ou o valor lido da memória de dados, e o EscReg, que comanda a escrita do valor escolhido.

Questões Pipeline

Quais as características do processador MIPS (arquitetura) que permitem se ter uma implementação em pipeline eficiente?

- As instruções do MIPS possuem o mesmo tamanho, o que torna simples a busca e a decodificação de instruções.

- Possui pouco formato de instruções, sempre com o registrador fonte na mesma posição. Essa simetria torna possível a leitura do banco de registradores ao mesmo tempo da determinação do tipo de instrução.

- Apenas as instruções de load word e store word manipulam operandos na memória. Esta restrição permite que o cálculo de endereço possa ser feito no estágio de execução, sendo o acesso à memória realizado no estágio seguinte. Se as outras instruções pudessem operar dados diretamente da memória, seria necessário ter um estágio apenas para cálculo do endereço.

Compare a técnica de implementação em pipeline com a técnica de implementação multiciclo. Quais as vantagens e desvantagens de cada técnica?

Vantagens

O uso de pipeline permite que todos os recursos envolvidos operem em paralelo, de maneira que mais instruções possam ser processadas na unidade de tempo considerada.

No multiciclo as instruções não são executadas na mesma quantidade de tempo, portanto, as instruções são executadas no seu tempo característico (e não no tempo da instrução mais longa). Também não há ocorrência de conflitos.

Desvantagens

No multiciclo não é possível que várias instruções sejam processadas ao mesmo tempo. É preciso que uma instrução seja totalmente processada para que a próxima instrução comece sua execução. Isto torna mais lenta a execução de um grupo de instrução, pios há recursos não utilizados todo o tempo.

No pipeline, para haver um desempenho mais notável é necessário uma grande quantidade de instruções. Enquanto o pipeline não está totalmente cheio, a performance não é ideal. Além disso, há o fato do balanceamento imperfeito, o ciclo de clock deve ser de acordo com a instrução mais longa. Problemas de conflitos de dados, estruturais e de controle que necessitam de instruções que exigem acréscimo de hardware para sua solução que nem sempre são solucionadas completamente, havendo perda de efetividade do mesmo jeito.

Adiantamento de resultados

Em instruções do tipo R, o resultado a ser armazenado no registrador destino já é calculado no estagio EX, pela ULA. Portanto, o dado que causa dependência já está disponível onde deve ser escrito no BR, no estagio ER. Para realizar o adiantamento do desse resultado (evitando conflitos de dados) precisamos obter entradas da ULA a partir de qualquer dos registradores do pipeline em vez de um só do registrador DI/EX. Colocando multiplexadores na entrada da ULA, com controles apropriados um controle de adiantamento deve ser implementado no estagio EX, pois os

multiplexadores da ULA responsáveis pele seleção dos valores adiantados estão nesse estagio. Os números dos registradores com os operandos do estagio DI são passados via registrador DI/EX para saber de vai ou não ser necessário o adiantamento dedados.

Conflitos por dados e paradas

Em casos de instrução de LW seguida de uma instrução dependente de algo precisa parar o pipeline. Além da unidade de adiantamento precisamos também de unidade de detecção de conflitos. Esta unidade verifica se o sinal LerMem está ativo, ou seja, se a instrução é de loadWord e verifica se o campo referente ao registrador – destino d LW no estagio EX é igual a qualquer dos registradores – fonte da instrução que estiver no estagio DI. Se a condição for verdadeira, a instrução pára por um ciclo de clock. Dessa forma a lógica de adiantamento pode tratar a dependência e a execução prossegue. Para impedir que a instrução dos estágios BI e DI avancem no pipeline, é preciso que tanto o PC quanto o registrador BI/DE não sofram qualquer modificação. Além disso, devemos inserir uma bolha no pipeline.

Conflitos por desvios condicionais

Com a pressuposição de desvios condicional precisamos descartar 3 instruções no caso de haver desvio . Para reduzir esse retardo, devemos deslocar a execução do desvio condicional para o estagio DI, deslocando o somador do estagio MEM para o estagio DI. Para tratar da decisão de desviar ou não, devemos comparar os valores dos dois registradores lidos durante o estagio DI. Mudando isso tudo só haverá uma única instrução a ser eliminada do pipeline. Podemos acrescentar uma nova linha de controle.

Conflitos por Dados e Adiantamento de Resultados

Ocorre quando há instruções dependentes sendo executadas seqüencialmente.

Solução:

O compilador poderia inserir duas instruções independentes entre as duas instruções dependentes e consecutivas. O problema dessa solução ocorre quando não forem encontradas instruções independentes, o que leva a inserção de instruções NOP, ocupando ciclos de clock sem realizar qualquer trabalho produtivo.

Solução Melhor:

O adiantamento de resultados. Essa solução funciona perfeitamente quando o valor a ser adiantado vem de uma instrução tipo R. No caso do *loadword (lw)* é necessário haver uma parada por um ciclo de clock, antes da técnica de adiantamento ser utilizada.