**CAPÍTULO 6.1 ATÉ 6.8 – PATTERSON**

6.1 – Visão geral do pipelining

* Pipelining é uma técnica de implementação em que várias instruções são sobrepostas na execução.
* É fundamental hoje em dia para tornar os processadores mais rápidos.
* O tempo de um único processo não diminui, mas o tempo geral sim.
* O pipeline melhora a vazão do sistema.
* Com trabalho suficiente para ser realizado, o ganho do pipeline se aproxima ao número de estágios.
* Pipeline funciona melhor quando estiver cheio.
* Pipeline do MIPS possui 5 estágios.
* Tempo entre instruções com pipeline = Tempo sem pipeline/Número de estágios.

Projetando conjunto de instruções para pipelining

* Vantagens para o pipeline do MIPS:
* 1) Instruções de mesmo tamanho ajudam ao primeiro estágio poder pesquisar a instrução enquanto o segundo decodifica.
* 2) A simetria dos tipos de instruções que o MIPS tem proporciona a possibilidade de o hardware ler os registradores no segundo estágio enquanto classifica qual instrução leu.
* 3) Os operandos em memória só aparecem em loads e stores. Isso permite que o estágio de execução possa calcular o endereço de memória para usá-lo só no estágio seguinte.
* 4) Os dados só podem ser transferidos entre processador e memória em um único estágio do pipeline devido ao alinhamento dos operandos na memória, assim não é preciso se preocupar com instruções acessando a memória mais de uma vez.

Pipeline hazards.

* Hazards ocorrem quando a próxima instrução não pode ser executada no ciclo de clock seguinte.

Hazards estruturais

* Signficica que o hardware não pode admitir a combinação de instruções que queremos executar no mesmo ciclo de clock.
* O hazard estrutural do mips ocorreria caso houvesse uma única memória, duas instruções tentariam ler e escrever na mesma memória ao mesmo tempo.

Hazard de dados

* Ocorre quando o pipeline precisa ser interrompido porque uma outra etapa precisa ser concluída.
* Uma instrução depende de outra anterior que ainda está no pipeline.
* Não é preciso esperar que a instrução termine para resolver o hazard de dados.
* Fowarding ou bypassing é o acréscimo de hardware para fornecer um caminho de dados que adiantará o resultado de uma instrução anterior para ser utilizado na instrução correntemente executada.
* Os fowardings só podem ser de um estágio para os estágios seguintes.
* O fowarding não impede todos os stalls do pipeline.

Hazards de controle

* Também chamado de hazard de desvio, um acontecimento em que a instrução apropriada não pode ser executada no seu devido ciclo de clock porque a instrução buscada não é aquela necessária. Ou seja o fluxo de endereços de instrução não é aquele que o pipeline desejava.
* Opção 1 para resolver o hazard. Stall, insere um stall após o desvio até que seja decidido a próxima instrução a ser chamada. Se não resolvermos o desvio no segundo estágio, o custo da demora seria muito grande para o pipeline.
* Opção 2 para resolver o hazard. Previsão, preveja que o desvio não é tomado e ganhe tempo caso isso seja verdade. Versão mais sofisticada da previsão de desvios analisa o todo e decide se o desvio será tomado ou não. Previsões que sempre tomam a mesma decisão são estáticas. As mais úteis são dinâmicas que baseiam sua escolha em diversos fatores, as vezes mantendo históricos que aumentam a precisão dos acertos.
* Opção 3 para resolver o hazard. Delayed branch. Sempre executa a próxima instrução sequencial com o desvio ocorrendo após o atraso de instrução. O software do MIPS colocará uma instrução que não é afetada pelo desvio e o desvio ocorrerá após essa instrução.

Resumo da visão geral do pipelining

* Explora o paralelismo entre as instruções em um fluxo de instruções sequenciais.
* Fundamentalmente invisível ao programador.
* Pipeline não reduz a latência(número de estágios do pipeline, ou tempo entre duas instruções), mas aumenta a vazão.

6.2 – Um caminho de dados usando pipeline

* 1) Estágio IF
* 2) Estágio ID
* 3) Estágio EX
* 4) Estágio MEM
* 5) Estágio WB
* É necessário um registrador entre os estágios.
* O PC pode ser considerado um registrador de pipeline especial.
* Os estágios são:
* Busca de instruções: Ler a instrução da memória e coloca-a no registrador de pipeline IF/ID. O computador precisa se preparar para potenciais desvios, por isso salva o valor do PC no registrador IF/ID.
* Decodificação de instruções e leitura do banco de registradores: Os valores dos registradores são passados para o registrador de pipeline ID/EX.
* Execução: A ULA executa a operação e salva o valor no registrador de pipeline EX/MEM.
* Acesso à memória: Ler a memória de dados e salva o que foi lido no registrador de pipeline MEM/WB.
* Escrita do resultado: Coloca o valor de MEM/WB no devido registrador.
* Qualquer informação necessária à um estado posterior do pipeline precisa ser passada a ele pelos registradores de pipeline.
* Cada compoenente lógico só pode ser usado dentro de um único estágio para evitar hazards estruturais (conflitos).

6.3 – Controle de um pipeline

* Não existe sinal de escrita para os registradores de pipeline pois este são escritos “automaticamente” a cada ciclo de clock.
* Os sinais de controle são ativos em cada estágio:
* 1) Os sinais de leitura da memória de instruções e de incrmento do PC são automáticos.
* 2) Não há nada de especial a ser feito.
* 3) RegDst, OPAlu, OrigAlu.
* 4) Branch, lêMem, Escreve Mem.
* 5) MemParaReg e EscreveReg.

6.4 – Hazard de dados e forwarding

* Forwarding ULA-ULA
* Fowarding de qualquer registrador de pipeline para a ULA.
* Fowarding MEM-ULA.

6.5 – Hazards de dados e stalls

* Fowarding não funcionará bem para instrução lw seguida de uma leitura no registrador que foi salvo.
* Preenche-se os stalls com nops.
* Nops são instruções que não fazem nada para mudar de estado.
* Uma bolha atrasa a execução de tudo que estiver atrás dela.

6.6 – Hazards de desvio

* Ocorrem com menos frequência que os hazards de dados.
* Não tem uma solução tão eficiente quanto o fowarding.
* Sempre considerar que o desvio não foi tomado. Temos que descartar instruções caso o desvio seja tomado.
* Executar o teste em um hardware separado.
* Em pipelines profundos a previsão estática pode ferir gravemente o desempenho.
* Previsão dinâmica de desvios: Previsão durante a execução utilizando informações em tempo de execução.
* Buffer de previsão de desvios: Memória que contém informações sobre um desvio, se ele foi ou não tomado recentemente. Previsão de 1 bit.
* Previsão de 2 bits.
* Delay slot: O slot diretamente apóis a instrução de delayed branch que sempre é executada.
* Delayed branch perde poder com pipelines mais profundos, pois apenas um slot não é suficiente.
* Previsor correlato combina de um determinado desvio com informações globais.
* Previsor de torneio é uma ferramenta com múltiplas previsões para um desvio e um seletor de previsão.

6.8 – Exceções

* Transfere controle para rotina de exceção, caso algo ocorra.
* Interrupção imprecisa: As interrupções ou exceções nos computadores pipeline não estão associadas à instrução exata que foi a causa da interrupção.
* Interrupção precisa: Está sempre associada À instrução correta nos computadores em pipeline.