Resumo Hardware – Prova 2 – rngs

Métricas de Desempenho

1. Tempo de Execução  
   Tempo necessário para a conclusão de uma tarefa.
2. Throughput(Vazão)  
   Trabalho realizado durante uma unidade de tempo.

Obs: uma métrica pode afetar o resultado da outra.

Pipeline

Técnica de implementação que permite que várias instruções sejam processadas simultâneamente, cada parte do HW atuando numa instrução distinta, maximizando assim a utilização dos recursos de HW.

O Pipeline conta com uma melhora no desempenho uma vez que existe um aumento do número de estágios do pipeline, permitindo mais execuções em paralelo. A vazão é melhorada, limitada pelo estágio de pipeline mais lento.

✖Os estágios devem possuir a mesma duração. O tempo de execução de cada instrução não diminui com o pipeline.

✖ É possível que ocorram problemas que gerem perca de desempenho, como dependências entre diferentes instâncias de execução das instruções.

✖ Na implementação do pipeline operações de escrita e de leitura de registradores levam metade de um ciclo de clock.

✖ Para se ter um controle de cada estágio durante a decodificação os sinais de controle para o resto da instrução são gerados e armazenados nos registradores auxiliares (IF/ID, ID/EX, EX/MEM e MEM/WB). A cada ciclo o registrador atual passa os sinais de controle para o registrador do próximo estágio.

Estágios:

1. **IF (I**nstruction **F**etch**)**

Busca instrução

1. **ID** (**I**nstruction **D**ecode)

Decodifica instrução e lê registradores

1. **EX** (**EX**ecute operation)

Calcula dados ou endereços

1. **MEM** (Access **MEM**ory operand)

Acessa operando na memória

1. **WB** (**W**rite result **B**ack to register)

Escreve resultado no registrador

✖ MIPS é um processador projetado pensando em pipeline  
Na ISA do MIPS todas as intruções são de 32 bits, facilitando assim a busca e decodificação em um ciclo. Conta também com um repertório de instruções pequeno e regular permitindo a leitura de registradores e decodificação em um único ciclo, além de poder endereçar o load/store mais rapidamente (cálculo no 3º estágio e acesso no 4º estágio).

✖Com a implementação do Pipeline é necessário um aumento do custo do hardware. Para evitar conflito em que instruções necessitem de um mesmo recurso é necessária duplicação de unidades. Entretanto o pipeline evita que os recursos fiquem ociosos, ou seja, eles são sempre usados de maneira mais eficiente.

Conflitos

Tipos

* Estruturais

Disputa de recursos, um recurso necessário a uma instrução está ocupado por outra.

* Controle

Decisão de qual instrução ser executada dependendo da instrução anterior, exemplo: desvios.

* Dados

Dependencia de dados entre instruções, exemplo: uma operação soma depende do resultado da operação que a antecede.

**Conflitos Estruturais**  
O HW não permite que certa combinação de instruções sejam executadas.

*✔ Solução comum: Replicação de recursos*

**Conflito de Dados**Uma instrução para ser executada depende de um dado gerado por uma instrução anterior...

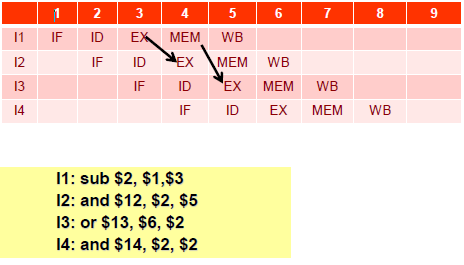
– **add $s0, $t0, $t1**

**sub $t2, $s0, $t3**O resultado da soma só estaria disponível no 5º ciclo de clock, entretanto o mesmo é necessário já no 4º ciclo por conta da instrução de sub.

*✔ Soluções em software* Inserção de NOPs (Compilador detecta os conflitos de dados e insere instruções de NOP no código, bolhas para retardar a execução da próxima instrução).  
 Re-arrumação de código (Cabe ao compilador reorganizar o código de forma que as intruções não gerem esse tipo de conflito).  
 Obs: Soluções em software geram aumento na complexidade da implementação do compilador/montador e podem comprometer o desempenho.

*✔ Soluções em Hardware* 1.Método do Curto-Circuito(Forwarding)  
 2. Inserção de retardos(stalls)

1. Forwarding ou Bypassing

Utiliza o resultado desajdo assim que o mesmo é computado. São necessárias conexões extras na unidade de processamento, velha “gambiarra!”, mas em HW não existem gambirras! 

✖Idéia é acrescentar ao HW uma unidade com uma lógica capaz de detectar conflitos de dados e controlar unidade de processamento para realizar o forwarding. Deve-se acrescentar mais conexões para permitir que resultados possam ser utilizados antes de escritos no banco de registradores.

✖Para detectar o conflito pode-se passar o número de registradores ao longo do pipeline.

✖Possivelmente, acrescenta-se multiplexadores para que outros estágios possam selecionar a fonte do operando, banco de registradores ou resultado gerado por outra instrução anterior no pipeline.

1. Inserção de Retardos

Quando não podemos utilizar forwarding, inserimos retardos.

Verificar se instrução depende do load no estágio ID Se detectado, insira um retardo.

**Conflito acontece quando:**

ID/EX.MemRead and ((ID/EX.RegisterRt = IF/ID.RegisterRs) or (ID/EX.RegisterRt = IF/ID.RegisterRt))  
  
**Resumo inacabado :\ melhor partir para resolução de provas…**