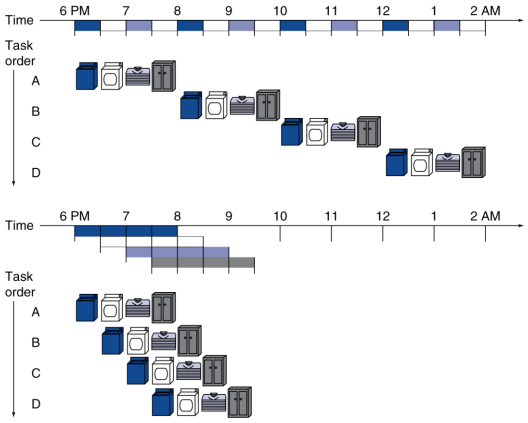
Arquitetura de um processador

3. Pipeline: definição e conceitos

Prof. John L. Gardenghi
Adaptado dos slides do livro

O que é Pipeline?

- Lavanderia: sobrepor tarefas
  - O paralelismo melhora o desempenho



- Quatro tarefas:
  - Melhoria = 8/3.5 = 2.3
- Non-stop:
  - Speedup = 2n/0.5n + 1.5 ≈ 4 = number of stages

MIPS Pipeline

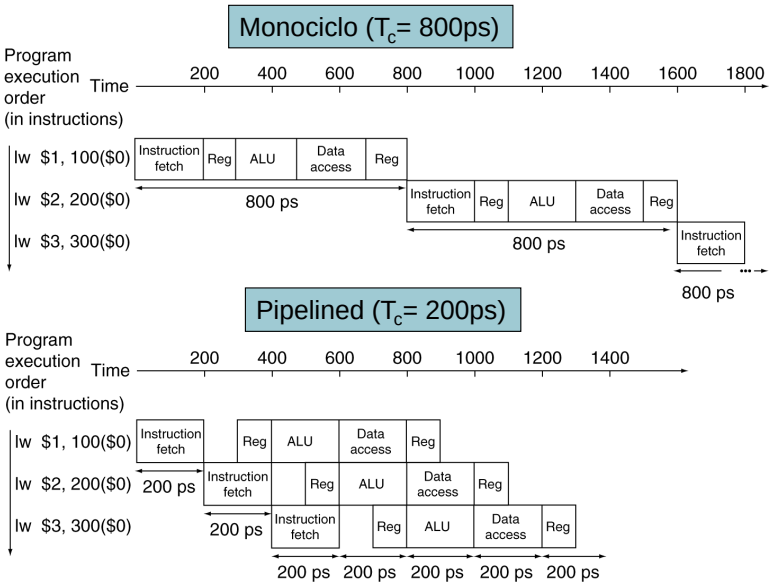
- Cinco estágios, um passo por estágio
  1. IF: Recuperação da instrução da memória (instruction fetch)
  2. ID: Decodificação da instrução & leitura dos registradores (instruction decode)
  3. EX: Execução da operação ou cálculo do endereço
  4. MEM: Acesso à memória de dados
  5. WB: Escrita do resultado no registrador (write-back)

Desempenho do Pipeline

- Suponha que o tempo dos estágios seja
  - 100ps para leitura ou escrita de registradores
  - 200ps para outros estágios
- Comparando um caminho de dados pipeline com monociclo:

Instr	IF	ID	EX	MEM	WB	Total
lw	200ps	100 ps	200ps	200ps	100 ps	800ps
sw	200ps	100 ps	200ps	200ps		700ps
R-format	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps

Desempenho do Pipeline



Melhorias no Pipeline

- Se os estágios estiverem balanceados
  - i.e., todos demoram o mesmo tempo
  - Tempo da instrução$_{\text{pipeline}}$

$$= \frac{\text{Tempo da instrução}_{\text{sem pipeline}}}{\text{Número de estágios}}$$

$$= 4.$$

- A melhoria deve-se a uma maior **vazão** de instruções
  - Latência (tempo para cada instrução) não melhora

Hazards

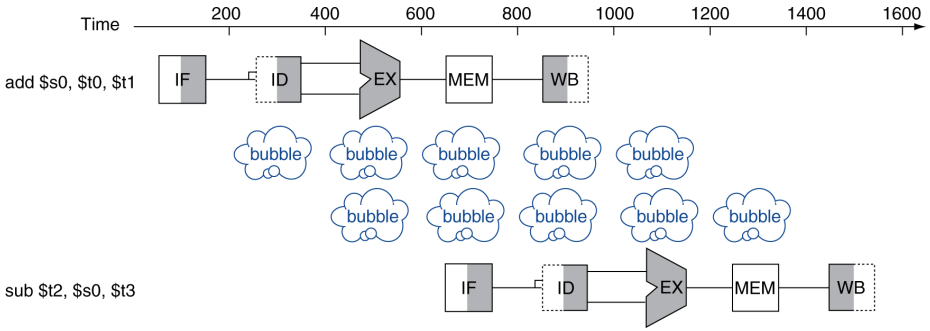
- Situações que impedem de começar a próxima instrução no próximo ciclo
- Hazards estruturais
  - Um recurso necessário está indisponível
- Hazard de dados
  - Precisa aguardar uma instrução anterior terminar para usar sua saída
- Hazard de controle
  - Decidir uma ação de controle depende de uma instrução anterior

Hazards estruturais

- Conflito ao utilizar um recurso
- Suponha um processador MIPS com apenas uma unidade de memória
  - Load/store acessa a memória de dados
  - Recuperar uma instrução não seria possível em determinado ciclo, causando um *stall* (bolha)
- Por isso o caminho de dados possui diferentes unidades de memória para instruções e dados

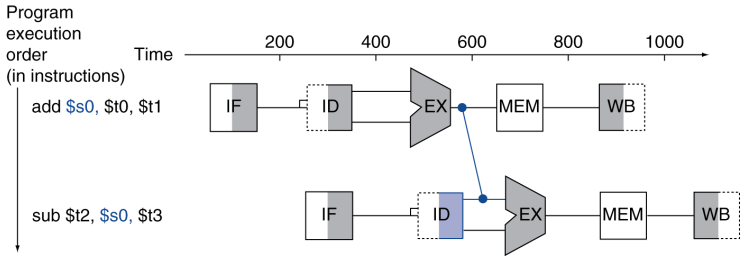
Hazards de Dados

- Uma instrução depende do resultado de outra instrução anterior
  - `add $s0, $t0, $t1`
    `sub $t2, $s0, $t3`



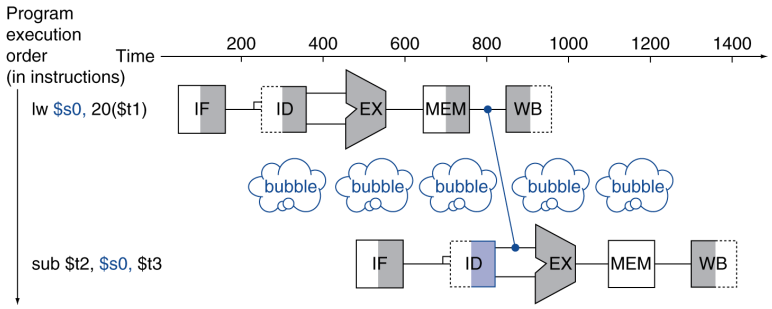
Forwarding (Bypassing)

- Já usa um resultado assim que for calculado
  - Não espera ser armazenado num registrador
  - Requer conexões extras no caminho de dados.



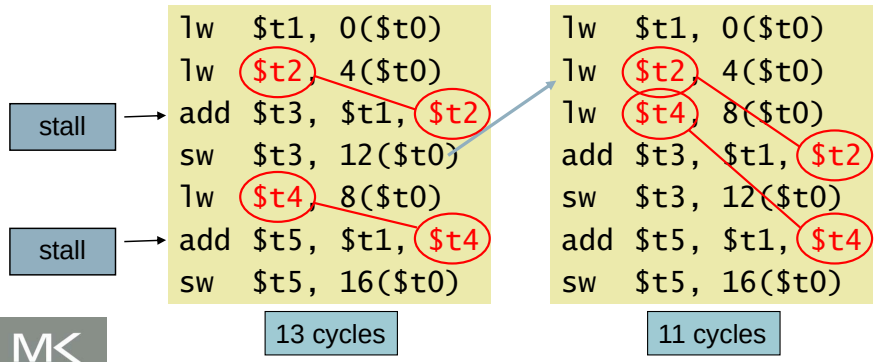
Hazard de dados Load-Use

- Nem sempre é possível evitar *stalls* usando *forwarding*
  - Se o valor necessário ainda não foi calculado



Reordenar instruções para evitar *stalls*

- Reordenar o Código para evitar o uso do resultado de um load na instrução seguinte
- Código C: `A = B + E; C = B + F;`

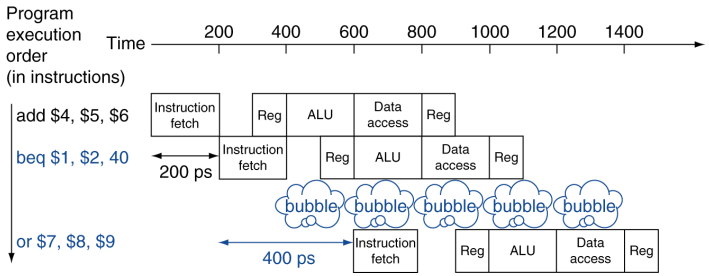


Hazards de controle

- Desvio determina o fluxo de controle
  - Recuperar a próxima instrução depende do resultado do desvio
  - Não é possível sempre recuperar a instrução correta com pipeline
    - Pois ainda estará na fase ID do desvio
- No pipeline MIPS
  - A comparação do branch deveria ser feita antes no pipeline
  - Adicionar hardware para fazer isso na etapa ID

Stall em desvios

- Aguarda até que o desvio seja tomado ou não para recuperar a próxima instrução

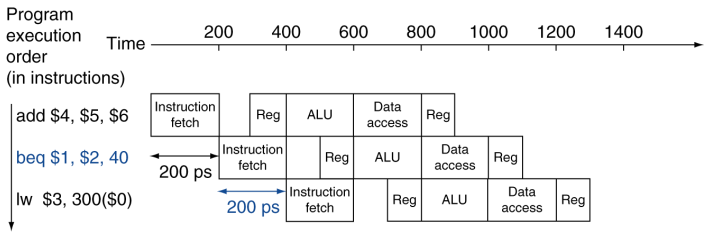


Predição de desvio

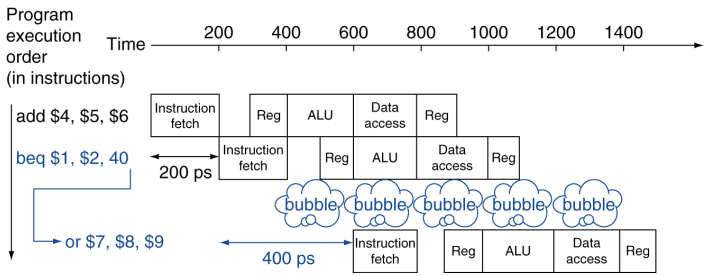
- Pipelines mais longos não podem determinar prontamente o resultado do desvio antecipadamento
  - As penalidades de stall tornam-se inaceitáveis
- Prediz a saída do desvio
  - O *stall* acontece apenas quando erra
- No pipeline MIPS
  - É possível predizer que o desvio não é tomado
  - Recupera a instrução depois do desvio

MIPS com predição de desvio

Prediction correct



Prediction incorrect



Tipos de predições de desvio

- Predição de desvio estática
  - Baseada no comportamento típico
  - Exemplo: laços e condicionais
    - Prediz que o desvio “para trás” é tomado
    - Prediz que o fluxo normal não é tomado
- Predição de desvio dinâmica
  - Hardware mede o comportamento atual dos desvios
    - e.g., grava o histórico recente de cada desvio
  - Assume que o comportamento futuro seguirá a tendência
    - Quando errar, faz um stall enquanto recupera a nova instrução, e atualiza o histórico

Resumo do Pipeline

- Pipeline melhora o desempenho aumento a vazão de instruções
  - Executa múltiplas instruções em paralelo
  - Cada instrução tem a mesma latência
- Sujeito a hazards
  - Estruturais, dados e controle
- O conjunto de instruções afeta a complexidade do design do pipeline

Caminho de dados com pipeline