Arquitetura e Organização de Computadores II

Pipeline Hazards

Prof. Nilton Luiz Queiroz Jr.

Pipeline

- Num pipeline real existem situações que impedem a execução da próxima instrução
  - Essas situações são chamadas de hazards;
- Os Hazards são divididos em três classes:
  - Hazards estruturais;
  - Hazards de dados;
  - Hazards de controle;

Tipos de Hazards

- Hazards estruturais:
  - Conflitos de recurso quando o hardware não pode aceitar todas as combinações possíveis de instruções executadas em paralelo;
  - Uma instrução que está no pipeline precisa de um recurso que está em uso por outra instrução;

Hazards Estruturais

- Exemplo:
  - Suponha uma arquitetura com apenas uma memória (para dados e instruções);
  - Lembrando que em diversos momentos podem existir acesso a memória;
    - No exemplo da arquitetura MIPS a busca de instruções e o acesso à memória para leitura e escrita
  - Desse modo, ao buscar uma instrução, deveria-se esperar com que o recurso fosse desalocado;
    - Outra alternativa é duplicar o barramento de acesso a memória, ou então usar duas memórias (como a arquitetura MIPS);

Hazards Estruturais

	ciclo 1	ciclo 2	ciclo 3	ciclo 4	ciclo 5	ciclo 6	ciclo 7	ciclo 8	ciclo 9	ciclo 10
lw r1, 20(r2)	Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado					
sub r11,r3,r4		Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado				
add r12,r3,r4			Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado			
lw r13,24(r2)				Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado		
add r14,r5,r6					Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado	

Hazards Estruturais

	ciclo 1	ciclo 2	ciclo 3	ciclo 4	ciclo 5	ciclo 6	ciclo 7	ciclo 8	ciclo 9	ciclo 10
lw r1, 20(r2)	Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado					
sub r11,r3,r4		Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado				
add r12,r3,r4			Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado			
lw r13,24(r2)				—	Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado	
add r14,r5,r6						Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado

Tipos de Hazards

- Hazards de dados:
  - Quando uma instrução depende do resultado de uma instrução a qual ainda está no processo de execução;
  - Uma instrução *j* é dependente dos dados de uma instrução *i* quando:
    - A instrução *i* produz um resultado que será lido pela instrução *j;*
      - Exemplo:

```
instrução i        add r1, r2, r3
instrução j        add r4, r1, r5
```

    - A instrução *j* depende da instrução *k*, e a instrução *k* depende da instrução *i*;
      - Exemplo:

```
instrução i        add r1, r2, r3
instrução k        add r4, r1, r5
instrução j        add r7, r4, r6
```

Hazard de dados

- Os Hazard de dados podem ser de três diferentes tipos
  - Leitura após escrita (read after write - raw)
  - Escrita após escrita (write after write - waw)
  - Escrita após leitura (write after read -war)

Tipos de hazard de dados

- Leitura após escrita (RAW), ou dependência verdadeira:
  - Uma instrução *i* modifica um registrador ou um local de memória;
  - Uma instrução *j* que vem após *i* lê os dados no registrador ou local de memória escrito;
  - Se a instrução *i* ainda não acabou, e a instrução *j* ocorre tem se um Hazard do tipo RAW
  - Exemplo:

```
add r1, r2, r3

mul r4, r1, r5
```

Tipos de hazard de dados

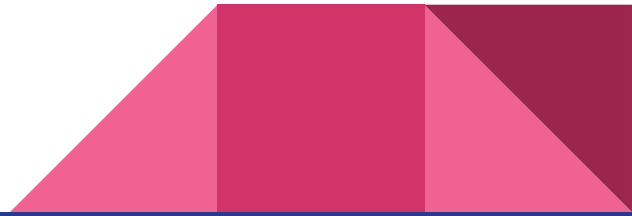
- Escrita após escrita (WAW), ou dependência de saída:
  - Uma instrução *i* escreve em um registrador ou local de memória;
  - Uma instrução *j* posterior a *i* escreve no mesmo registrador ou local de memória;
  - Quando *i* escreve após *j* tem se um Hazard do tipo waw
  - Esses hazards ocorrem somente em pipelines que:
    - Escrevem em mais de um estágio;
    - Permitem uma instrução continuar mesmo com uma instrução prévia parada;

Tipos de hazard de dados

- Escrita após leitura (WAR), ou antidependência:
  - Uma instrução *j* tenta escrever em um registrador ou local de memória;
  - Uma instrução *i* tenta ler esse mesmo local de memória ou registrador;
  - Se a instrução *j* tentar escrever no destino antes que a instrução *i* faça a leitura tem-se um Hazard do tipo WAR;
  - Não ocorrem nas questões de pipeline estático;
    - Todas leituras são na decodificação de instrução;
    - Todas escritas são nas fases de escrita em registradores ou em memória;
    - Geralmente ocorrem:
      - Em escritas nos primeiros estágios do pipeline e outras instruções com leituras em estágios a frente de escrita no pipeline;
      - Quando existe reordenação nas instruções;

Hazard de dados

- Existem algumas maneiras de resolver Hazard de dados:
  - Reordenar as instruções: O programador fica responsável por evitar instruções que irão causar hazards
    - Específico para cada microarquitetura;
  - Stall: incluir no hardware um controle que “pause” a execução até que a execução anterior termine;
    - Inserção de bolhas;
  - Bypass (ou fowarding): inserção de um circuito que faz com que os dados de estágios mais a frente sejam encaminhados para estágios anteriores;
    - Criar um “corte de caminho” no hardware;



Técnicas para resolver Hazards

- Stall:

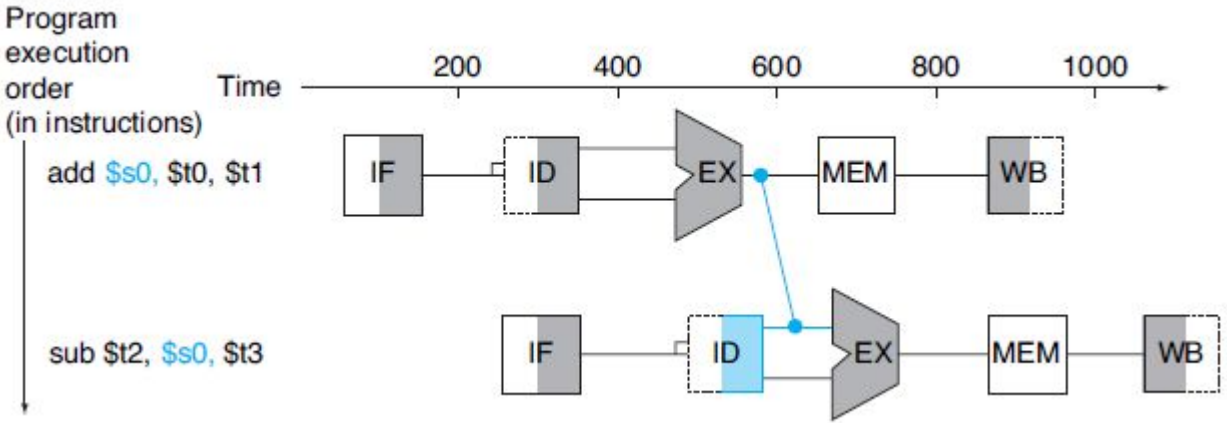
	ciclo 1	ciclo 2	ciclo 3	ciclo 4	ciclo 5	ciclo 6	ciclo 7	ciclo 8	ciclo 9	ciclo 10
add r1,r2,r3	Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado					
add r4,r1,r5		Busca de Instrução	Decodificação de instrução	Decodificação de instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado		
instr 3			Busca de Instrução	Busca de Instrução	Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado	
instr 4						Busca de Instrução	Decodificação de instrução	Execução	Acesso aos dados	Escrita do resultado

Técnicas para resolver Hazards

- Como a leitura de dados é feita somente no segundo estágio, a bolha deverá ser inserida a partir do 2 estágio;
- Como descobrir quando é necessário inserir uma bolha?
  - É necessário comparar os registradores do estágio de decodificação com o registrador fonte;
    - Note que cada instrução tem armazenada seu registrador fonte, então é necessário fazer a comparação com todos bancos de registradores;

Técnicas para resolver Hazards

- Bypassing (ou fowarding):
  - Adicionar hardware que faz a ligação dos dados que acabaram de sair de um estágio para outro;
    - Não é necessário esperar que um valor seja armazenado em um registrador;
  - Não é possível fazer forwarding de um estágio que ainda não foi calculado;

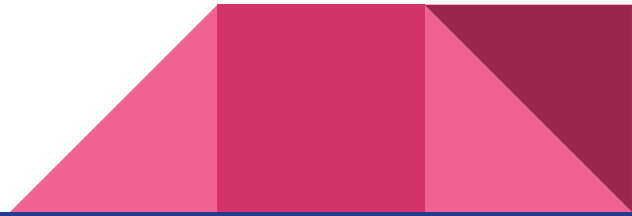


Tipos de Hazards

- Hazards de Controle:
  - Quando ocorre um desvio que altera o fluxo da execução;
  - Instruções que estão poucos endereços a baixo da instrução condicional que está ocorrendo no momento;
  - Instruções a seguir precisam ser descartadas;
  - Formas de tratar os desvios:
    - Múltiplos Fluxos;
    - Busca antecipada do alvo;
    - Buffer de laço de repetição;
    - Previsão de desvios;
    - Desvio atrasado;

Lidando com Hazards de controle

- Múltiplos fluxos:
  - Duplicar recursos iniciais do pipeline;
  - Busca instruções das duas possíveis alternativas de execução;
  - Segue o caminho somente para a certa;
  - Causa alguns problemas:
    - Atraso no acesso a registradores
    - Instruções de desvio adicionais podem entrar no pipeline antes da decisão ser tomada



Lidando com Hazards de controle

- Busca Antecipada:
  - Le alvo do desvio antecipadamente;
  - Mantém o alvo armazenado até a execução do desvio;

Lidando com Hazards de controle

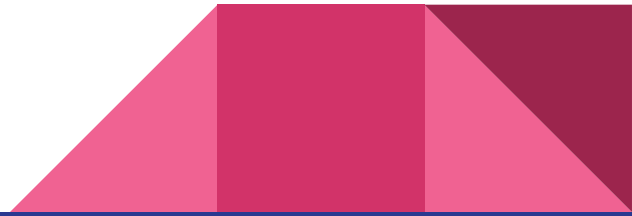
- Buffer de Laço de repetição:
  - Memória pequena e rápida;
  - Contém instruções mais recentemente lidas na sequência;
  - Busca no buffer antes de buscar na memória;
  - Possui alguns benefícios:
    - Instruções em sequência mantidas no buffer com uso de busca antecipada;
    - Quando o alvo do desvio está próximo não é necessário o acesso a memória logo após a decisão de tomada de desvios;
    - Pode armazenar todas instruções de um laço, reduzindo a quantidade de acesso a memória;

Lidando com Hazards de controle

- Previsão de desvios:
  - Nunca tomada;
  - Sempre tomada;
  - Por Opcode;
  - Chave tomada/não tomada;
  - Tabela de histórico de desvio;

Previsão de desvios

- Nunca tomada:
  - Assume que o desvio não irá acontecer;
  - Incrementa o PC e executa a próxima instrução;
- Sempre Tomada:
  - Assume que o desvio será tomado;
  - Sempre busca a instrução do alvo;



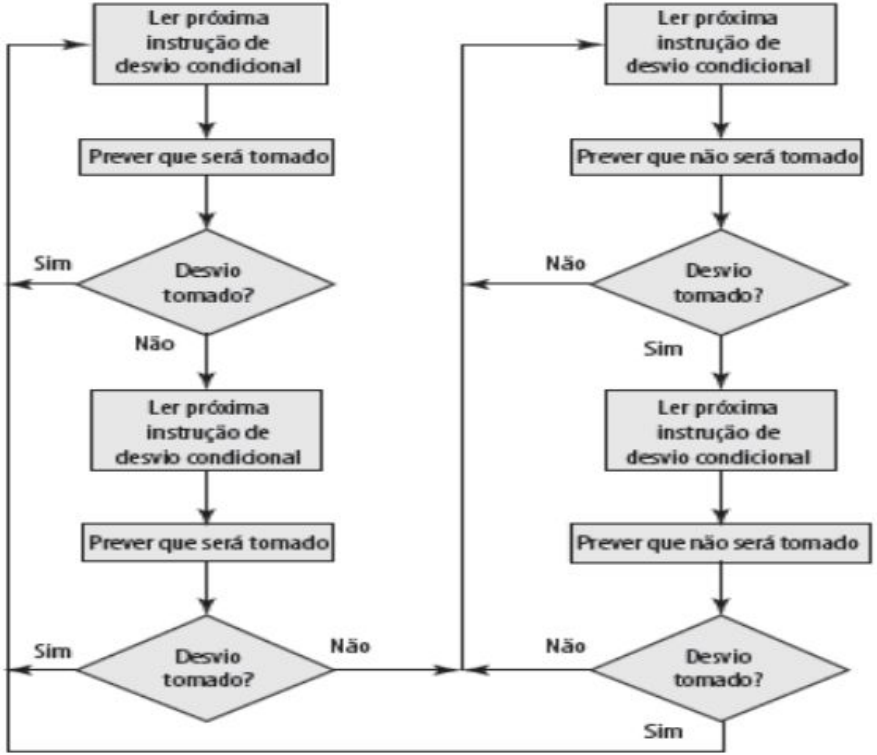
Previsão de desvios

- Baseada no opcode:
  - Algumas instruções têm maior chance de realizar o salto;
    - Para alguns opcodes o desvio é tomado;
    - Para outros o desvio não é tomado;

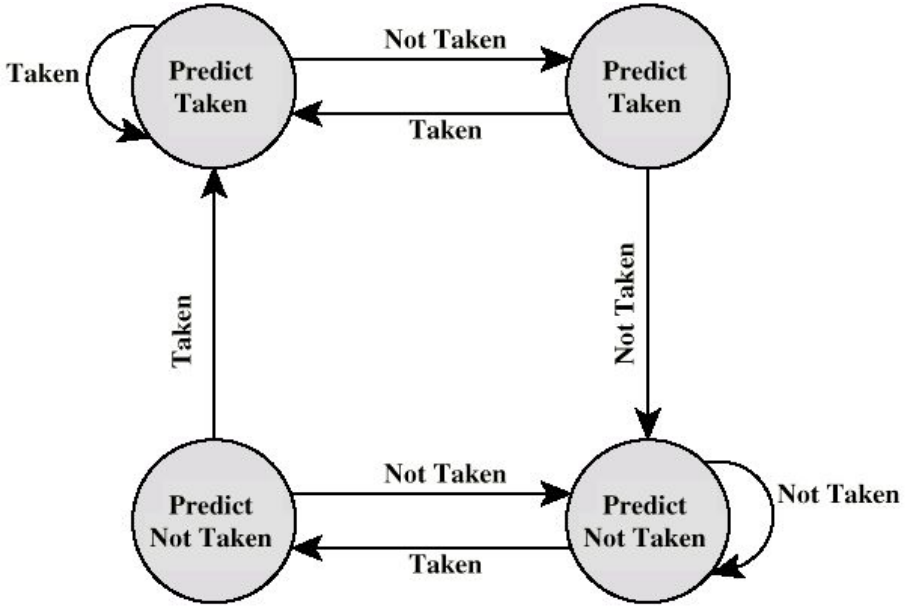
Previsão de desvios

- Chave tomada/não tomada:
  - Armazena um histórico das instruções de desvios condicionais;
  - Armazena bits para o histórico
  - Armazenados em locais temporários de alta velocidade;
  - Pode-se implementá-la:
    - Associando bits a instruções de desvio condicional que estão na cache;
    - Manter uma tabela dos desvios recentemente executados com 1 ou mais bits para cada entrada;
      - Processador acessa a tabela de forma associativa ou com os bits de ordem mais baixa do endereço da instrução de desvio;

Chave tomada/não tomada



Chave tomada/não tomada



Lidando com Hazards de controle

- Tabela de histórico de desvio:
  - Pequena memória cache associada ao estágio de leitura da instrução do pipeline;
  - As entradas da tabela possuem 3 elementos:
    - Endereço da instrução de desvio;
    - Estado de uso da instrução e endereço da instrução alvo;
    - Endereço da instrução alvo, ou a própria instrução alvo;
  - Guarda o histórico global dos desvios mais recentes;
  - Guarda-se o histórico global dos desvios mais recentes.
  - Quando um desvio condicional é encontrado, tenta-se prever se ele será ou não tomado de acordo com a tabela de histórico.

Lidando com Hazards de controle

- Desvio atrasado:
  - Reorganiza as instruções do programa;
    - Não altera a semântica do mesmo;
  - Alteração nas instruções de branch da arquitetura, que assumem que N instruções após um branch sempre serão executadas;
    - Esse valor N é conhecido por delay slot;
    - Dever do compilador ou programador encontrar N instruções que podem ser colocadas após os desvios sem que exista conflitos de dados;
    - Quando não é possível se insere uma operação do tipo NOOP;

Exercícios

1. Suponha as seguintes instruções:

```
ADDI $s0, $zero, 10
ADDI $s1, $zero, 10
ADDI $s2, $zero, 9
ADDI $s3, $zero, 7
BEQ  $s0, $s1, L1
SUB $s4, $s1,$s2
J L2
L1: SUB $s4, $s2,$s1
L2:
```

Mostre uma possível ordem das instruções aplicando a técnica de desvio atrasado com, delay slot de 1, e evitando hazards de dados. Mostre também o diagrama de execução do pipeline para a sequência de instruções ao lado e para a nova sequência de instruções;

Para o diagrama suponha que o estágio de execução já atualiza o PC com o endereço da próxima instrução.

Referências

STALLINGS, W. Arquitetura e organização de computadores: projeto para o desempenho. 8. ed. Prentice Hall, 2009.

PATTERSON, D. A.; HENNESSY, J. L. Computer Organization and Design: The Hardware/Software Interface. Fourth edition.

PATTERSON, D. A.; HENNESSY, J. L. Computer Architecture: A Quantitative Approach. Fifth Edition

David Wentzlaff Computer Architecture ELE 475 / COS 475 Slide Deck 2: Microcode and Pipelining Review, Notas de Aula.