Pipeline

Disciplina: ACH2055 - Arquitetura de Computadores (Valdinei Freire da Silva)

Lista 2*

1. Usando um diagrama como o visto em aula (slide 12), mostre os caminhos para adiantamento de dados necessários à execução das seguintes instruções:

```
add $4, $3, $2
add $5, $4, $3
add $2, $5, $4
```

2. Identifique todas dependências de dados existentes no código a seguir. Quais dependências são conflitos que podem ser resolvidos via adiantamento?

```
add $2, $5, $4
add $4, $2, $5
sw $5, 100($2)
lw $3, 0($5)
add $2, $3, $4
```

3. Na implementação de pipeline vista em classe, considerou-se uma ULA para realizar operações lógicas e aritméticas entre registradores. Desejamos introduzir instruções lógicas e aritméticas de ponto flutuante. No entanto operações de ponto flutuante na ULA levam o dobro do tempo.

   a) Como poderia ser feita esta implementação sem afetar o desempenho?

   b) Quais as implicações desta implementação para a solução de conflitos de dados via adiantamento?

4. Considere a execução do seguinte código no pipeline visto em aula:

```
add $1, $2, $3
add $4, $5, $6
add $7, $8, $9
add $10, $11, $12
add $13, $14, $15
```

   a) No quinto ciclo da execução, quais registradores serão lidos, e quais serão escritos?

   b) Quantos ciclos leva a execução deste código?

   c) Em qual ciclo é feita a soma entre os registradores $11 e $12?

   d) Em qual ciclo o valor do registrador $13 é alterado?

5. Considere um programa composto por 100 instruções `lw`, sem se preocupar com a qualidade do código. Sabendo que cada instrução depende da instrução anterior a ela, calcule a CPI real se o programa rodar no caminho de dados pipeline visto em classe.

6. O código a seguir apresenta um conflito do tipo "leitura após escrita" que é resolvido com o adiantamento de dados:

```
add $2, $3, $4
add $5, $2, $6
```

   Considere uma situação similar na qual uma leitura da memória ocorra após uma escrita:

```
sw $7, 100($2)
lw $8, 100($2)
```

   Como esta situação é diferente daquela que envolve registradores?

7. Considere que não seja permitido especificar um deslocamento na memória para as instruções de load e store, isto é:

```
sw $1, ($2)
lw $1, ($2)
```

   a) Com essa mudança é possível melhorar o desempenho do controle monociclo?

   b) Com essa mudança é possível melhorar o desempenho do controle multiciclo?

   c) Com essa mudança é possível melhorar o desempenho do pipeline?

8. A reordenação do código pode ser utilizada para evitar paradas do pipeline. Reescreva o código a seguir de maneira a MINIMIZAR o desempenho da

*Adaptado de Organização e Projeto de Computadores (Patterson e Hennessy).

execução deste código, mas de forma que o mesmo resultado seja obtido.

```
lw $3, 0($5)
lw $4, 4($5)
add $7, $7, $3
add $8, $8, $4
add $10, $7, $8
sw $6, 0($5)
beq $10, $11, Loop
```

9. Na implementação de pipeline vista em sala de aula, o desvio condicional só é decidido no estágio MEM (slides 11 e 19). Isso faz com que o retardo do desvio seja de 3 ciclos. Esses ciclos são perdidos caso a predição tenha sido feita errada.

   a) Em qual estágio deve ser tomada a decisão sobre o desvio condicional para reduzir o retardo do desvio a duas instruções? Redesenhe o caminho de dados usando um novo hardware para reduzir o retardo do desvio a dois ciclos.

   b) Qual a implicação da alteração anterior no tempo ciclo do processador?

   c) Como reduzir o retardo para 1 ciclo e qual a implicação dessa alteração no tempo de ciclo do processador?

   d) Em qual das alterações você acha que o ciclo será mais prejudicado?

10. Uma forma de evitar a perda de ciclos ao executar um desvio condicional é adiantar ou retardar a execução de instruções que são sempre executadas independente do resultado do desvio.

    a) No código abaixo é possível evitar perda de ciclos com reordenação de código? Explique.

```
        add $1, $2, $3
        bne $8, $7, Loop
        add $5, $7, $9
Loop :  sub $4, $5, $6
```

    b) No código abaixo é possível evitar perda de ciclos com reordenação de código? Explique.

```
Loop :  sub $4, $5, $6
        add $1, $2, $3
        beq $1, $7, Loop
        add $5, $7, $4
```

    c) No código abaixo é possível evitar perda de ciclos com reordenação de código? Explique.

```
Loop :  sub $4, $5, $6
        add $1, $2, $3
        beq $8, $7, Loop
        add $5, $7, $9
```

    d) No código abaixo é possível evitar perda de ciclos com reordenação de código? Explique.

```
        add $1, $2, $3
        beq $1, $7, Loop
        add $5, $7, $9
Loop :  sub $4, $5, $6
```

11. Considere a máquina de estados vista em sala de aula para predição de desvio (slide 23) e que instruções de desvio possuem um retardo de apenas 1 ciclo. Tal máquina de estados é utilizada no código abaixo:

```
Loop :  sub $2, $2, $1
        beq $2, $0, Loop
        add $3, $4, $5
```

    Se os valores dos registradores possuem os seguintes valores iniciais: $\$0 = 0$, $\$1 = 1$ e $\$2 = 5$.

    (a) No pior e melhor caso quantos ciclos são gastos para executar este código? Considere o pipeline de 5 estágios visto em aula e os possíveis estados iniciais para a máquina de estados.

    (b) Repita o ítem anterior com uma máquina de estados com apenas 2 estados. Cada estado apresenta como predição a decisão tomada anteriormente.

12. Considere os estágios vistos em aula para o controle multiciclo. Com recursos limitados (1 única ULA, 1 memória de dados, 1 memória de instruções e um único banco de registradores) e quantidades de estágios diferentes por instrução podem ocorrer conflitos estruturais. Suponha que estes conflitos são resolvidos com a inserção de bolhas quando necessário.

    Desconsidere o uso da ULA no segundo estágio para cálculo de desvios, considere um somador especial alocado apenas para cálculo da próxima instrução e leitura e escrita de registradores são independentes.

    (a) Represente graficamente a execução do seguinte código.

```
lw $4, 0($6)
add $1, $6, $7
add $2, $6, $7
sw $3, 100($7)
add $5, $6, $7
```

    (b) Considere um estágio NULL, onde várias instruções podem aguardar até que um recurso esteja disponível. Represente a execução do código anterior. Desconsidere o uso da ULA no segundo estágio para cálculo do desvio.