
**Universidade Federal de Roraima**
**Departamento de Ciência da Computação**
**Arquitetura e Organização de Computadores**

# LISTA DE EXERCÍCIO 02

**Aluno: FRANCIS NUTEFE TSIGBEY**

**Matricula: 201514401**

**1) Quais as vantagens de um processador multiciclo em relação a um uniciclo?**

Um processador uniciclo executa cada instrução em um único ciclo de clock, mas nem todos os componentes são usados em todas as instruções e isso faz com que os componentes que não sendo usados fiquem ociosos. Em um processador multiciclo cada componente é executado em um ciclo de clock, ou seja, uma instrução que use n componentes gastará n ciclos de clock. Apenas com essa mudança já é possível obter uma melhoria de desempenho, uma vez que cada ciclo que clock poderá ter duração menor, em outras palavras, o processador ficará mais rápido. Outra vantagem que o multiciclo tem é estar apto para usar uma técnica chamada pipelining. Onde as instruções podem ser executadas em paralelo, enquanto o componente X executa uma parte da instrução, o componente Y executa outra parte de outra instrução.

**2) Quais as modificações necessárias em um processador multiciclo simples para que se introduza a função de pipeline?**

Deve-se utilizar 4 registradores entre os 5 estágios do pipeline, um registrador é colocado entre a transição de um estágio para o outro, devendo ser capaz de armazenar todos os dados que passam por ele, dados que poderão ser utilizados por outra instrução no próximo ciclo. É necessário ter um registrador para busca e decodificação de instruções, um registrador para decodificação da instrução e cálculo de endereço, um registrador para cálculo de endereço e acesso à memória de dados, e um registrador para acesso à memória de dados e escrita de registradores. E todos esses registradores precisam ter capacidade suficiente para armazenar todos os dados que passam por eles.

3) **Considerando o pipeline do MIPS (simples com MEM compartilhada para instrução e dados) euma iteração de loop conforme o trecho de programa abaixo, relacione os conflitos que podem ocorrer e seus consequentes stalls. Qual o speedup (por iteração) para o programa em relação à versão sem pipeline?**

**Loop: sub $t2, $t2, 4 lw $t1, 0($t2)** //Conflito de dados: esperando $t2

**add $t3, $t1, $t4** //Conflito de dados: Esperando pela função load $t1

**add $t4, $t3, $t3**

**sw $t4, 0($t2)** // Conflito de dados: Esperando pela escrita do $t4

**beq $t2, $0, loop** // Conflito Estrutural: memória em uso pelo sw

**6+8+6+6+7+5 = 38 ns tempo de execução**

4) **No programa abaixo, relacione as dependências (dados, WAR, WAW e outros) existentes.**

**div.d** F1, F2, F3
**sub.d** F4, F5, F1
**s.d** F4, 4(F10)
**add.d** F5, F6, F7
**div.d** F4, F5, F6

Raw:
Sub.d depende de F1 em div.d. s.d depende de F4 em sub.d. div.d depende de F5 em add.d.

WAR:
Add.d escreve em F5 que sub.d lê.

WAW
Sub.d usa F4 e pode terminar depois de div.d

**5) Em relação a memória cache. Um computador tem CPI 1 quando todos os acessos à memória acertam no cache. Loads e Stores totalizam 50% das instruções. Se a penalidade por miss é de 25 ciclos e o miss rate é 2%, qual o desempenho relativo se o computador acertar todos os acessos?**

$CPU_{\text{tempo ideal}}$ = ($CPU_{\text{ciclos de clock}}$ + Ciclos de Clock de Stall de Memória) x Período de ciclo de Clock =>
$CPU_{\text{tempo ideal}}$ = (IC x CPI + 0) x Período de ciclo de Clock =>
$CPU_{\text{tempo ideal}}$ = IC x 1 x Período de ciclo de Clock

Ciclos de clock de stall de memória = IC x (1 + 0,5) x 0.02 x 25 = IC x 0,75
$CPU_{\text{tempo real}}$ = (IC x 1 + IC x 0,75) x Período de ciclo de Clock =>
$CPU_{\text{tempo real}}$ = IC x 1,75 x Período de ciclo de Clock

Comparando os desempenhos com stalls pelo ideal:

$$\frac{\text{IC x 1,75 x Período de ciclo de Clock}}{\text{IC x 1 x Período de ciclo de Clock}} = 1{,}75$$

*Obs: (A solução da questão 5 foi adaptado do slide 23 do professor Herbert)*

**6) Descreva os seguintes conceitos:**

**a) Write through**

Escrita ao mesmo tempo no cache e na memória. A vantagem é baixa latência e alta taxa de transferência para aplicativos que exigem muita gravação. Mas a desvantagem é que há risco de disponibilidade de dados porque o cache pode falhar (e sofrer perda de dados) antes que os dados sejam mantidos no armazenamento de backup. Esse resultado nos dados sendo perdidos.

**b) Write back**

Escrita somente no cache e atualização na memória somente na substituição do bloco. Em geral, os dados são também gravados no *backing store* em segundo plano, mas a confirmação de conclusão não está bloqueada.
Vantagem: Garante uma recuperação rápida, garantindo que os dados estejam no armazenamento de apoio e não sejam perdidos caso o cache seja interrompido.

Desvantagem: A gravação de dados sofrerá latência, já que você precisa gravar em dois lugares sempre.

**c) Localidade Temporal**

Posições de memória, uma vez referenciadas, tendem a ser referenciadas novamente dentro de um curto espaço de tempo.

**d) Localidade Espacial**

Se uma posição de memória é referenciada, posições de memória cujos endereços sejam próximos a ela tendem a ser logo referenciadas.