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

**LISTA DE EXERCÍCIO 02**

**Aluno: Pedro Aleph Gomes de Souza Vasconcelos**

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

O ciclo de clock precisa ter a mesma duração para cada instrução no projeto uniciclo , e percorre várias etapas correspondentes às unidades funcionais necessárias até finalizar um ciclo. o desempenho geral de uma implementação de ciclo único não é muito bom já que várias classes de instrução poderiam ficar em um ciclo de clock mais curto.

Em uma implementação multiciclo, cada etapa na execução levará 1 ciclo de clock. permite que uma unidade funcional seja usada mais de uma vez por instrução, desde que seja usada em ciclos de clock diferentes. Esse compartilhamento pode ajudar a reduzir a quantidade de hardware necessária.

(OBS.: peguei do livro por que minhas palavras não ficam melhores)

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

Dividir as instruções em estágios, essas instruções são colocadas em uma fila de memória dentro do processador onde aguardam o momento de serem executadas. assim que uma instrução termina o primeiro estágio e parte para o segundo , próxima instrução já ocupa o primeiro estágio.

(OBS.:não tenho certeza se requer mais registradores)

**3) Considerando o pipeline do MIPS (simples com MEM compartilhada para instrução e dados) e uma 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:
subi $t2, $t2, 4        1
lw $t1, 0($t2)          2
add $t3, $t1, $t4       3
add $t4, $t3, $t3       4
sw $t4, 0($t2)          5
beq $t2, $0, loop       6
```

na linha 2 ocorre um stall no lw de $t2 em $t1, devido ao hazard de dados, requer resultado de $t2 na linha 1

na linha 3 o stall ocorre no add de $t1 a $t4, devido ao hazard de dados, requer o lw de $t1 na linha 2

na linha 4 o stall ocorre no add de $t3 a $t3, devido ao hazard de dados, requer o resultado de $t3 na linha 3

na linha 5 o stall ocorre no sw de $t4, devido ao hazard de controle ,$t4 já está sendo usado para o resultado do add na linha 4

na linha 6 o stall ocorre no beq em $t2, devido ao hazard estrutural, $t2 está sendo usando para sw no mesmo ciclo de clock na linha 5

número de iterações sem pipeline: 21
número de iterações com pipeline: 10
speedup: 2,1

(OBS.: eu simplesmente contei as instruções e os registradores, quantas vezes eram usados, desculpe, não entendi muito bem)

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

**div.d F1, F2, F3** 1
**sub.d F4, F5, F1** 2
**s.d F4, 4(F10)** 3
**add.d F5, F6, F7** 4
**div.d F4, F5, F6** 5
Dependência de dados reais: div.d 1 e sub.d 2, sub.b 2 e s.d 3, add.d 4 e div.d 5
Antidependência: sub.d 2 e add.d 4, add pode escrever em registrador que sub lê.
WAR: uso de F5 por sub
Dependência de saída: sub.d 2 e div.d 5, sub pode terminar depois de div.
WAW: uso de F5
(OBS.: não entendi bem, não tem no livro, tentei seguir o exemplo no slide)

**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?**

$$\text{CPU Execution Time} = \text{IC} \times 1.0 \times \text{ClockCycle}$$

$$\text{Memory Stall Cycles} = \text{IC} \times 0.75$$

$$\text{CPU Execution Time}_{\text{cache}} = 1.75 \times \text{IC} \times \text{ClockCycle}$$

$$\frac{\text{CPU Execution Time}_{\text{cache}}}{\text{CPU Execution Time}} = \frac{1.75 \times \text{IC} \times \text{ClockCycle}}{1.0 \times \text{IC} \times \text{ClockCycle}}$$

Resultado final: 1,75

(OBS: ta no slide mas tentei entender)

**6) Descreva os seguintes conceitos:**

**a) Write through**

as escritas sempre atualizam a cache e a memória, garantido que os dados sejam sempre consistentes entre os dois.

**b) Write back**

manipula escritas atualizando os valores apenas no bloco da cache e, depois, escrevendo modificado no nível inferior da hierarquia quando o bloco é substituído.

**c) Localidade Temporal**

Se um item é referenciado, ele tenderá a ser referenciado novamente em breve .

**d) Localidade Espacial**

Se um item é referenciado, os itens cujos endereços estão próximos tenderão a ser referenciados em breve.

(OBS.: páginas 354 e 365 do livro azul)