**LISTA 2**

**Victor Barbosa Rocha**

**ATENÇÃO**: Descrever as soluções com o máximo de detalhes possível. Todos os artefatos

(relatório, código fonte de programas, e outros) gerados para este trabalho devem ser

adicionados em um repositório (com o seguinte formato nome\_ufrr\_AOC\_2018\_2) no

site github.com.

PRAZO DE ENTREGA: 06/12/2018

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

As instruções podem ser executadas em número de ciclos de clock variados, pois os ciclos são delimitados por fases (IF-ID-EXE-MEM-WB) que podem ou não estar presentes em todas as instruções, por exemplo, o load passa pela fase de acesso à memória, enquanto o add não.

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

Componentes entre fases para guardar os valores entre cada transição, assim liberando espaço do componente e para levando em conta instruções que necessitem guardar valores por mais de um ciclo de clock.

**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

**lw $t1,** 0(**$t2**)

**add $t3, $t1**, $t4

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

**sw $t4,** 0(**$t2**)

**beq $t2,** $0, loop

b) c) d) Conflito de dados: o valor de t2 não terminou de ser calculado para ser usado no load,o mesmo para o t1.

c) Conflito estrutural: o add vai querer acessar a memória, mas o load acessa a memória duas vezes, então a instrução c só começa quando a b) terminar.

d) e e) Conflito de dados pelo valor de t3 e t4.

Sem pipelining = 38 ns; Com pipelining = 33ns. Speedup = 38/33 de 15%.

**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

Existem dependência entre div.d e sub.d em F1, em sub.d e s.d em F4 e entre add.d e div.d em F5.

WAR entre sub.d e add.d em F5 e RAW também com F5 e add.d e div.d.

WAW com F4 entre sub.d e 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?**

**- Sempre acertando:**

CPU execution time = (CPU clock cycles + Memory stall cycles) x Clock cycle

= (IC x CPI + 0) x Clock cycle

= IC x 1.0 x Clock cycle

**- Situação real:**

Memory stall cycles = IC x (Memory accesses / Instruction) x Miss rate x Miss penalty

= IC x (1 + 0.5) x 0.02 x 25

= IC x 0.75

CPU execution time(cache) = (IC x 1.0 + IC x 0.75) x Clock cycle

= 1.75 x IC x Clock cycle

**- Resultado:**

CPU execution time(cache) = 1.75 x IC x Clock cycle

CPU execution time 1.0 x IC x Clock cycle

= 1.75

**6) Descreva os seguintes conceitos:**

**a) Write through**

Escrita ao mesmo tempo no cache e na memória.

**b) Write back**

Escrita somente no cache e atualização na memória somente na substituição do bloco.

**c) Localidade Temporal**

Tem a ver com as maiores chances de referenciar novamente um item já referenciado recentemente.

**d) Localidade Espacial**

Tem a ver com o fato de que itens próximos à itens já referenciados tem maiores chances de serem também referenciados.