UFRR

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

LISTA DE EXERCÍCIO 02

ALUNO: PHILIP MAHAMA AKPANYI
MATRÍCULA: 201514402

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?

Uniciclo	Multiciclo
1) Todas as instruções têm o mesmo comprimento de ciclo do clock. Todos eles levam a mesma quantidade de tempo, independentemente do que eles realmente fazem. O ciclo do clock é determinado pelo caminho mais longo. E isso quer dizer que a tendência do processador uniciclo executar instruções com grande latência é alta.	1) A execução de cada instrução é dividida em passos em ciclo de clocks diferentes. Por isso, o tempo que leva para executar uma instrução completa pode ser reduzido consideravelmente.
2) Cada unidade funcional só pode ser usada uma vez por ciclo. Com isso, algumas unidades são duplicadas, aumentando o custo do hardware consideravelmente.	2) Uma unidade funcional pode ser usada várias vezes baseado no ciclo de clock

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

Resposta: É preciso a função pipeline nas situações onde alguns componentes baseam suas operações nos resultados de dados e operações dos outros componentes que tiveram acesso a esse dado primeiro. Exemplo de tal situação é

```
add $t1, $t0, 4
add $t2, $t1, $t1
```

Neste caso, a segunda instrução precisa aguardar o fim da primeira operação para ter o resultado do registrador $t1.

Uma modificação que pode ser feito para introduzir pipeline em tal situação é começar a busca de instrução IF da segunda operação antes mesmo que a primeira termine. Em casos que a operação é feita na instrução anterior, o resultado pode ser armazenado em um registrador temporário para ser usado quando outros componentes precisam.

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?

```
A) Loop: subi $t2,
   B)$t2, 4 lw $t1,
   C)0($t2) add $t3,
   D)$t1, $t4 add $t4,
   E) $t3, $t3 sw $t4,
   F) 0($t2) beq $t2,
   G) $0, loop
```

Resposta:

Linha B) – utilização o registrado $t2 em lw, entretanto na linha acima foi executada a operação subi que é armazenado em $t2. Neste caso, tem um stall para executar o load.

Linha C) – tem a utilização do registrador $t1 na soma, entretanto na linha acima, é preciso $t1 para guardar o resultado para poder executar a soma.

Linha D) – o resultado da linha C) é necessário para poder executar a linha D), por isso um stall para aguardar o resultado.

Linha E) para poder executar sw, é preciso aguardar a operação na linha D). Mais uma vez, um stall.

A versão sem pipeline executará nesta ordem IF-ID-EX-MEM-WB, enquanto a versão com pipeline poupará tempo de esperar e já começa a busca de outra instrução IF na etapa de ID da instrução anterior

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
```

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?

Resposta:

Se estiver acertando:

Tempo de execução do CPU = (ciclo de clock CPU + ciclo de stalls por memória) x Ciclo de clock

= (IC x CPI + 0) x ciclo de clock

= IC x 1.0 x ciclo de clock

Na situação real:

$$\text{Ciclo de stalls por memória} = IC \times \frac{\text{Acesso de memória}}{\text{Instrução}} \times \text{taxa de falha} \times \text{penalidade de falha}$$

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

= IC x 0.75

Tempo de execução do $CPU_{(cache)}$ = (IC x 1.0 + IC x 0.75) x ciclo de clock

= 1.75 x IC x ciclo de clock

Resultado:

$$\frac{\text{Tempo de execução do CPU (cache)}}{\text{Tempo de execução do CPU}} = \frac{1.75 \times IC \times \text{ciclo de clock}}{1.0 \times IC \times \text{ciclo de clock}}$$

= 1.75

6) Descreva os seguintes conceitos:

a) Write through: a gravação de dados é feita de forma síncrona tanto para o cache quanto para a memória.

b) Write back: inicialmente, a escrita é feita apenas para o cache. A gravação na memória é adiada

até que o conteúdo modificado esteja prestes a ser substituído por outro bloco de cache.

c) Localidade Temporal: refere ao reaproveitamento de dados específicos ou recursos dentre um tempo de duração pequena.

d) Localidade Espacial: se refere ao uso de elementos de dados dentre lugares de armazenamentos relativamente próximos.