Centro Federal de Educação Tecnológica de Minas Gerais

Departamento de Computação
Curso de graduação em Engenharia da Computação
Organização e Arquitetura de Computadores II
Profa. Daniela Cristina Cascini Kupsch (cascini@decom.cefetmg.br)

Comece a fazer esta lista imediatamente. Você nunca terá tanto tempo para resolvê-la quanto agora.
Após a data da entrega você pode entregar a lista para ser corrigida, mas não receberá os pontos pelo trabalho.
A lista deve ser feita à mão.

LISTA III

Exercício 1

Qual seria o número de ciclos da sequência de código descrita abaixo **se nenhuma nova execução de instrução puder ser iniciada até que a execução da instrução anterior tiver sido concluída** (ou seja, se houver ou não dependência de dados)? Ignore a busca e a decodificação da instrução. Assuma, que a execução não fique em *stall* por falta da próxima instrução. Entretanto, somente uma instrução/ciclo pode ser enviada. Considere que o desvio é tomado e que existe um *delay slot* de desvio de **dois ciclos**. Mostre o código com <stall> inserido onde for necessário para acomodar as latências definidas. (Dica: uma instrução com latência "+2" precisa que dois ciclos de clock de <stall> sejam inseridos na sequência de código. Pense desta maneira: uma instrução de 1 ciclo possui uma latência 1+0, significando zero estado de espera extra. Assim, a latência 1+1 implica um ciclo de stall; latência 1+N possui N ciclos de stall extras).

```
Loop:  LD     F2, 0 (RX)
I0:    DIVD   F8, F2, F0
I1:    MULTD  F2, F6, F2
I2:    LD     F4, 0 (RY)
I3:    ADDD   F4, F0, F4
I4:    ADDD   F10, F8, F2
I5:    ADDI   RX, RX, #8
I6:    ADDI   RY, RY, #8
I7:    SD     F4, 0(RY)
I8:    SUB    R20, R4, RX
I9:    BNZ    R20, LOOP
```

Latências	
Memory LD	+4
Memory SD	+1
ADD, SUB de inteiro	+0
Desvios	+2
ADDD	+1
MULTD	+5
DIVD	+12

Exercício 2

Pense no que realmente significam os números de latência – eles indicam o número de ciclos que determinada função exige para produzir sua saída, e nada mais. Se o pipeline ocasionar stalls para os ciclos de latência de cada unidade funcional, pelo menos você terá a garantia de que qualquer par de instruções de ponta a ponta (um "produtor" seguido por um "consumidor") será executado corretamente. Contudo, nem todos os pares de instruções possuem um relacionamento produtor/consumidor. Às vezes, duas instruções adjacente não têm nada a ver uma com a outra. Quantos ciclos o corpo do loop na sequência de código do exercício 1 exigiria se o pipeline detectasse dependência de dados verdadeiras e só elas produzissem *stall*, ao invés de tudo ficar cegamente em stall só porque uma unidade funcional está ocupada? Mostre o código com <stall> inserido onde for necessário para acomodar as latências definidas. (Dica: uma instrução com latência "+2" precisa que dois ciclos de clock de <stall> sejam inseridos na sequência de código. Pense desta maneira: uma instrução de 1 ciclo possui uma latência 1+0, significando zero estado de espera extra. Assim, a latência 1+1 implica um ciclo de stall; latência 1+N possui N ciclos de stall extras).

Exercício 3

Considere um projeto de múltiplo despacho com dois pipelines de execução. Suponha que cada pipeline seja capaz de iniciar a execução de uma instrução por ciclo, além de largura de banda de busca (*fetch*)/ decodificação (*decode*) suficiente no *front-end*, de modo que sua execução não ficará em stall. Considere que os resultados podem ser encaminhados (*forwarded*) imediatamente de uma unidade de execução (pipeline) para outra ou para si mesma. Considere, ainda, que o único motivo para um pipeline *stall* é observar uma dependência de dados verdadeira. Quantos ciclos o loop do Exercício 1 exigiria?

Exercício 4

Considere um BTB (*Branch Target Buffer*) que tenha penalidades de zero, dois e dois ciclos de clock para previsão correta de desvio condicional, previsão incorreta e uma falha de buffer, respectivamente. Considere também um projeto de BTB que distingue desvios condicionais e não condicionais, armazenando os endereços alvo para um desvio condicional e a instrução alvo para um desvio não condicional

a) Qual é a penalidade em ciclos de clock quando um desvio não condicional é encontrado no buffer?
b) Determine a melhoria da utilização de *branch folding* para desvios não condicionais. Suponha uma taxa de acerto de 80%, uma frequência de desvio não condicional de 5% e uma penalidade de dois ciclos de clock para uma falha de buffer. Quanta melhoria é obtida com essa modificação? Quão alta deve ser a taxa de acerto para essa melhoria gerar ganho de desempenho, ou seja, qual o percentual a partir do qual você já obtem um ganho de desempenho?