

2ª Lista de Exercícios

1. Considere a execução do seguinte código em um pipeline de 5 estágios: Busca, Decodificação, Execução, Memória e Escrita.

```
dadd R5, R6, R7
ld    R6, 100(R7)
dsub  R7, R6, R8
dadd  R7, R7, #1
sd    R6, 100(R7)
```

Execute este código no simulador e responda:

- Quantos ciclos serão gastos para executar este código se não houver adiantamento de dados?
 - E com adiantamento de dados?
 - Qual o total e quais tipos de “stalls” que ocorreram em cada caso?
2. Dado um processador MIPS64 com um pipeline de 5 estágios, escreva um trecho de código em linguagem de montagem do DLX em que ocorram as seguintes situações:
- Adiantamento de dados da saída para a entrada da ALU;
 - Adiantamento de dados da saída do estágio de memória para a entrada da ALU;
 - Uma bolha no pipeline devido a uma dependência de controle.

3. Considere o seguinte trecho de código em linguagem de montagem:

```
                dadd R5, R0, R0
                daddi R20, R0, #40
Soma:          ld    R10, A(R20)
                dadd  R5, R5, R10
                dsubi  R20, R20, #4
                bne   R20, R0, Soma
```

Assuma que o pipeline do processador não possui mecanismos de “stalls” ou adiantamento de dados. Reescreva o código inserindo o menor número possível de **nops** para eliminar as dependências de dados. Se for possível, reordene as instruções para minimizar o número de **nops** (as instruções podem ser reordenadas desde que se preserve a equivalência semântica).

4. Considere o seguinte trecho de código:

```
Copia:      daddi  R20, R0, 20
            ld     R10, 1000(R20)
            sd     2000(R20), R10
            dsubi  R20, R20, #8
            bne   R20, R0, Copia
```

Assuma que o pipeline do processador não possui mecanismos de “stalls” ou adiantamento de dados.

- Reescreva o código inserindo o menor número de nops necessários; reordene as instruções, se possível, para minimizar o número de nops preservando a equivalência semântica.
- Escreva uma fórmula para o número de ciclos necessários para executar este laço (otimizado) como uma função de N (número de palavras copiadas), ou seja, se forem copiadas N palavras quantos ciclos serão necessários?

5. Considere o seguinte laço:

```
dadd  R8, R18, R19
sd     4(R12), R6
dsub   R6, R15, R8
ld     R5, 36(R4)
dadd   R4, R5, R8
slt    R5, R2, R7
bne    R5, R0, rotulo
```

Encontre as dependências de dados existentes no código acima. Para um pipeline do MIPS com 5 estágios e “interlock” por hardware, mostre as bolhas criadas no pipeline.

- Compute o tempo de execução (em ciclos de relógio) para o programa a seguir: Calcule o desempenho do pipeline do MIPS com adiantamento de dados. É possível melhorar este desempenho? Em caso positivo, quais técnicas você empregaria? Quantifique a melhoria de desempenho nos dois programas para cada técnica utilizada.

```
PROG1:
daddi  R2, R0, #300
ld     R1, 100(R0)
daddi  R3, R0, #200
seq    R15, R1, R3
beq    R15, R0, END1
dadd   R1, R1, R2
END1  sd     R1, 100(R0)
daddi  R4, R0, #600
daddi  R5, R0, #700
dadd   R4, R4, R5
daddi  R6, R0, #800
dsub   R4, R4, R6
```

7. Descreva os tipos de dependências de dados, como e quando elas ocorrem e quais as técnicas para resolvê-las por hardware e por software? Que tipo(s) de dependência é(são) eliminada(s) com a renomeação de registradores.
8. Em um processador superescalar, descreva o mecanismo utilizado para possibilitar o término da execução na mesma ordem especificada no código objeto.
9. Durante a execução de um programa em uma arquitetura superescalar, em que casos é necessário anular resultados de instruções já ou parcialmente executadas?
10. Descreva o mecanismo utilizado para garantir exceção precisa e recuperação do contexto em caso de desvios preditos incorretamente em arquiteturas superescalares.