Lista de Exercícios

Aluno:

1ª Questão) Considere o trecho de programa no quadro abaixo e os conteúdos iniciais de registradores e posições de memória relevantes. Convenções**: X – bolha, F - flush do pipeline, -- para estágio não usado, - → adiantamento ou leitura após escrita no mesmo ciclo. Estágios do pipeline: BI(Busca), DI (Decodificação), EX (Execução) MEM (Memória) WB (Writeback)**

addl $t4, $zero, 2
root : add $t1, $t2, $t3
lw $t3, 0x100($t1)
sw $t3, 0x200($t1)
subl $t4, $t4, 2
beq $t4, $t3, root
addl $t3, $t3, 0x100
	Conteúdos Iniciais da memória e dos registradores relevantes:
	$t1=0x100, $t2=0x100, $t3=0x100, $t4=0x100
	Mem [0x100-0x103]= 0x002345AB
	Mem [0x200-0x203]= 0x0000000A
	Mem [0x300-0x303]= 0x00000000
	Mem [0x400-0x403]= 0x00CD5F00

a) Simule a execução completa do programa (considere unidade de adiantamento).
b) O que a unidade de adiantamento (forwarding) está fazendo durante o quinto ciclo de execução? Se algumas comparações estiverem sendo feitas, mencione-as.

2aQuestão) Traduza o seguinte laço em C. Assuma que o inteiro i é mantido no registrador $t1, que $s2 contém o inteiro resultado, e $s0 contém o endereço base do inteiro arranjo:

```
        addi $t1, $0, 0
LOOP:   lw   $s1, 0($s0)
        add  $s2, $s2, $s1
        addi $s0, $s0, 4
        addi $t1, $t1, 1
        slti $t2, $t1, 100
        bne  $t2, $0, LOOP
```

3ª Questão) Considere a seguinte sequência de instruções, e assuma que estas sejam executadas em um pipeline com 5 estágios

Sequencia Instruções
lw $1, 40 ($6)
add $2, $3, $1
add $1, $6, $4
sw $2, 20($4)
and $1, $1, $4

a) Quais dependências são conflitos (hazards) que podem ser resolvidos com adiantamento? Quais dependências que são conflitos e irão provocar a parada (bolhas) na execução?

b) Se não há adiantamento ou detecção de conflito, insira nops para assegura a execução correta e desenhe o diagrama de execução do pipeline para este código

c) Repita o item anterior, mas adicione nops somente quando um conflito não pode ser evitado por mudando ou rearranjando estas instruções. Você pode assumir o registrador R7 para guardar valores temporários em seu código modificado.

d) Um conflito estrutural (duas instruções tentando acessar a memória) pode ser resolvido pelo compilador inserindo uma instrução nops.

e) Suponha as instruções abaixo. Qual o procedimento a ser adotado pela unidade de detecção de conflito

```
load $1,(10) $2
add  $2, $1, $3
```

4ª Questão) Considere a seguinte sequência de instruções, executadas em um datapath com pipeline de 5 estágios:

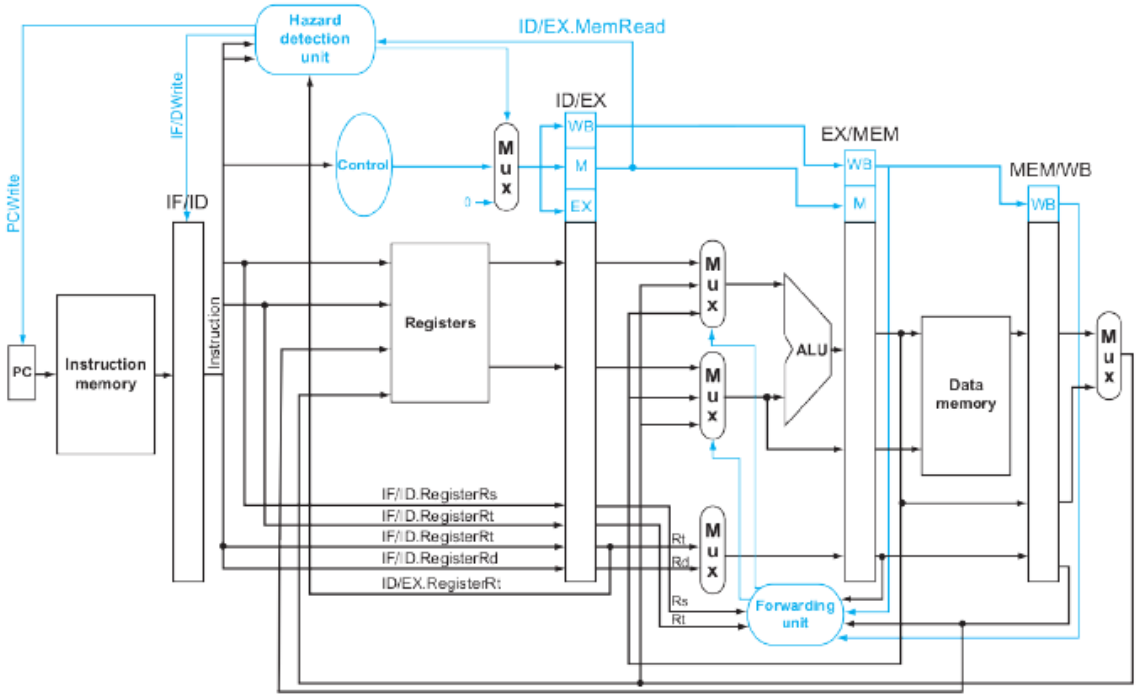
```
add r5, r2, r1
lw  r3, 4(r5)
lw  r2, 0(r2)
or  r3, r5, r3
sw  r3, 0(r5)
```

a) Na ausência de forwarding ou de detecção de conflito, insira nops para garantir que o código rode corretamente

(b) Repita (4a) usando nops somente quando um conflito não puder ser evitado pela mudança ou rearranjo dessas instruções. Assuma que o registrador r7 é usado para armazenar valores temporários no seu código modificado

(c) Se o processador tiver forwarding, mas esquecermos de implementar a unidade de detecção de conflitos, o que ocorrerá quando este código for executado?

(d) Na presença de forwarding, especifique, para os primeiros 5 ciclos da execução deste código, que sinais são ligados em cada ciclo pelas unidades de detecção de conflitos e de forwarding na figura abaixo:



e) Na ausência de forwarding, que novos sinais de entrada e saída precisaríamos ter na unidade de detecção de conflitos na figura acima? Usando esta sequência de instruções como exemplo, explique porque cada sinal é necessário.

(f) Para a nova unidade de detecção de conflitos acima, especifique que sinais de saída ela liga em cada um dos 5 primeiros ciclos, durante a execução desse código.