

## Prova de Arquitetura de Computador

Aluno:

Data:

1a Questão) (1,5 ponto) Considere a seguinte sequencia de instruções, e assuma que estas sejam executadas em um pipeline com 5 estágios (BI(Busca), DI (Decodificação), EX (Execução) MEM (Memória) WB (Write-back))

Sequencia Instruções  
**add \$1, \$5, \$3**  
**sw \$1, 0(\$2)**  
**lw \$1, 4(\$2)**  
**add \$5, \$5, \$1**  
**sw \$1, 0 (\$2)**

a) (0,5 ponto) Quais dependências são conflitos (hazards) que podem ser resolvidos com adiantamento? Quais dependências que são conflitos e irão provocar a parada (bolhas) na execução?

	1	2	3	4	5	6	7	8	10	11	12	13	14
<b>add \$1, \$5, \$3</b>	BI	DI	EX	--	W								
<b>sw \$1, 0(\$2)</b>		BI	DI	X	X	EX	MEM	---					
<b>lw \$1, 4(\$2)</b>			BI	X	X	DI	EX	MEM	W				
<b>add \$5, \$5, \$1</b>						BI	DI	X	↓EX	MEM	W		
<b>sw \$1, 0 (\$2)</b>							BI	X	DI	EX	MEM	---	

**add \$1, \$5, \$3**

Este conflito não pode ser resolvido com adiantamento. O adiantamento é realizado entre Rd de uma instrução a frente e RS e Rt de uma instrução anterior.  
 (0.2 ponto)

**sw \$1, 0(\$2)**

**lw \$1, 4(\$2)**

Se adiantamento precisa de uma bolha - 0.15. Sem adiantamento precisa de duas bolhas - 0.15. Se mencionar apenas que provoca bolhas (0,2)

**add \$5, \$5, \$1**

b) (0,5 ponto) Se não há adiantamento ou detecção de conflito, insira nops para assegurar a execução correta e desenhe o diagrama de execução do pipeline para este código.

Sequencia Instruções  
**add \$1, \$5, \$3**  
**NOP** 0,15 ponto  
**NOP**  
**sw \$1, 0(\$2)**  
**lw \$1, 4(\$2)**  
**NOP** 0,15 ponto  
**NOP**  
**add \$5, \$5, \$1**  
**sw \$1, 0 (\$2)**

1 NOP - 0.1

2 NOP - 0.15

1 NOOP Lugar errado 0.1  
 (descontar)

2 NOOP Lugar errado 0.15  
 (descontar)

Diagrama - 0,2 ponto

	1	2	3	4	5	6	7	8	10	11	12	13	14
<b>add \$1, \$5, \$3</b>	BI	DI	EX	--	W								
<b>NOP</b>		BI	DI	---	----	----							
<b>NOP</b>			BI	DI	----	----	-----						
<b>sw \$1, 0(\$2)</b>				BI	DI	EX	MEM	---					
<b>lw \$1, 4(\$2)</b>					BI	DI	EX	MEM	WB				
<b>NOP</b>						BI	DI	---	----	----			
<b>NOP</b>							BI	DI	---	----	----		
<b>add \$5, \$5. \$1</b>								BI	DI	EX	MEM	W	
<b>sw \$1, 0 (\$2)</b>									BI	DI	EX	MEM	W

c) (0,5 ponto) 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 \$S7 para guardar valores temporários em seu código modificado.

#### Critérios correção

- Substitui R7 no lugar certo - 0.15
- Reduziu o numero de nops para 1 e colocou no lugar certo- 0,15 ponto
- Diagrama - 0,2

	1	2	3	4	5	6	7	8	10	11	12	13	14
<b>add \$1, \$5, \$3</b>	BI	DI	EX	--	W								
<b>lw \$7, 4(\$2)</b>		BI	DI	EX	MEM	W							
<b>NOP</b>			BI	DI	---	---	---						
<b>sw \$1, 0(\$2)</b>				BI	DI	EX	MEM	---					
<b>add \$5, \$5. \$7</b>					BI	DI							
<b>sw \$7, 0 (\$2)</b>						BI	DI	EX	MEM	---			

ou

	1	2	3	4	5	6	7	8	10	11	12	13	14
<b>add \$7, \$5, \$3</b>	BI	DI	EX	--	W								
<b>lw \$1, 4(\$2)</b>		BI	DI	EX	MEM	W							
<b>NOP</b>			BI	DI	---	---	---						
<b>sw \$7, 0(\$2)</b>				BI	DI	EX	MEM	---					
<b>add \$5, \$5. \$1</b>					BI	DI							
<b>sw \$1, 0 (\$2)</b>						BI	DI	EX	MEM	---			

2a Questão) (1,0 Ponto) Considere o conjunto de instruções abaixo

Latência					
I1	lw	F2	45(R3)		1
I2	div	F6	F6	F4	5
I3	mult	F0	F2	F4	4
I4	div	F8	F6	F2	5
I5	add	F6	F8	F2	1
I6	sub	F10	F0	F6	1

a) (0,5 ponto) Identifique as situações de dependência (WAW, WAR, RAW) na seguinte sequência de código acima, do MIPS64:

WAW : I2 - I5, -- 0,05 ponto

WAR: I5 - I4, I5 - I2 -- 0,1 ponto

RAW: I3-I1, I4-I1, I5-I1, I4 - I2, I5-I4, I6-I2, I6 - I3, I6 - I5 - 0,40 ponto

b) (0,45 ponto ) Apresente uma sequência de termino em ordem e outra em fora de ordem (que execute no menor tempo)

1 a 2 - 0.05 dependencia 0.05

3 a 4 - 0.1

Termino em ordem - 0,20- ponto (melhor resposta)

I2	I1	I1	I3	__	I2	I4	I3				I4	I5	I5	I6	I6			
----	----	----	----	----	----	----	----	--	--	--	----	----	----	----	----	--	--	--

outra possibilidade mas usando um tempo maior

I1	I1	I2	I3	__	__	__	I2	I3	I4	__	__	__	__	I4	I5	I5	I6	I6
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

outra possibilidade

I1	I2	I1	I3	__	__	I2	I3	I4	__	__	__	__	I4	I5	I5	I6	I6
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Fora de ordem 0,25 ponto (melhor resposta)

I1	I1	I3	I2			I3		I2	I4					I4	I5	I5	I6	I6
----	----	----	----	--	--	----	--	----	----	--	--	--	--	----	----	----	----	----

I1	I1	I2	I3	__	__	__	I3	I2	I4	__	__	__	__	I4	I5	I5	I6	I6
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

3a Questão) (1,5 ponto) Suponha que uma máquina foi melhorada fazendo com que todas as instruções de ponto flutuante executassem 5 vezes mais rápido.

a) (0,75 ponto) Se o tempo de execução de um certo benchmark antes da melhoria era de 10 segundos. Qual será o ganho (speedup) se metade deste tempo é gasto executando instruções de ponto flutuante?

$$T_{old} = 10$$

$$T_{new} = T_{old} * \left[ (1 - fraction) + \frac{fraction}{speed\_up} \right]$$

$$T_{new} = T_{old} * \left[ (1 - 0.5) + \frac{0.5}{5} \right]$$

$$T_{new} = T_{old} * [0.5 + 0.1]$$

$$T_{new} = T_{old} * 0.6$$

$$Speed_{up} = \frac{T_{old}}{T_{new}} = \frac{1}{0.6} = 1.66$$

Se colocar 1,66 % ao invés de 166% - descontar - 0.05

b) (0,75 ponto) Suponha que devemos escolher um benchmark que demonstre a melhora na nova unidade de ponto flutuante. Deseja-se mostrar um ganho geral da ordem de 3. Considere um benchmark que roda em 100 segundos com o antigo hardware de ponto flutuante. Qual deve ser a proporção de operações de ponto flutuante nos programas deste benchmark de forma a se alcançar o ganho desejado?

$$T_{new} = T_{old} * \left[ (1 - fraction) + \frac{fraction}{speed_{up}} \right]$$

$$1 = \frac{T_{old}}{T_{new}} * \left[ (1 - fraction) + \frac{fraction}{speed_{up}} \right]$$

$$1 = 3 * \left[ (1 - fraction) + \frac{fraction}{5} \right]$$

$$1 = 3 * \left[ \frac{5 - 4 * fraction}{5} \right]$$

$$5 = 15 - 12 * fraction$$

$$fraction = \frac{10}{12} = 0.83$$

4a Questão) (2,0 ponto) Determine o número real de ciclos de clock para executar uma vez o trecho abaixo (do `blez $t1,end` até uma instrução que salte para trás ou até a última instrução do trecho ser executada, o que acontecer primeiro). Suponha unidade de adiantamento capaz de **adiantar dados da saída do terceiro estágio para a entrada do terceiro estágio, da saída do quarto estágio para a entrada do terceiro, e da saída do quarto estágio para a entrada do quarto estágio**. Assuma também leitura após escrita no banco de registrador no mesmo ciclo e um preditor de saltos de 2 bits **que inicialmente prevê salto não-realizado**. O PC é escrito no quarto ciclo de relógio de cada instrução. Detalhe a execução no diagrama pipeline abaixo, indique todos os adiantamentos de dados que ocorrerem (se ocorrerem) e mostre as bolhas nas posições adequadas, caso estas existam. Os valores iniciais dos registradores pertinentes são: `$t0=0x10010000`, `$t1=0x44` (=68 base 10), `$t2=0`, `$t4=0`, `$t3=0x100100AA`

Número de ciclos igual a \_\_\_\_14\_\_\_\_

### Critério correção

- cada adiantamento errado -0.1
- Se não seguir notação, considerar parcialmente o resultado (divide por o numero de instruções por 2)

Instrução	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
loop: blez \$t1,end	BI	DI	EX	---	---						BI	DI	EX	---	----				
sltiu \$t2,\$t1,65		BI	DI	EX	↓---	W													
bne \$t2,\$zero,nxt			BI	DI	↓EX	---	---												
sltiu \$t2,\$t1,91				BI	DI	EX	↓--	W											
beq \$t2,\$zero,nxt					BI	DI	↓EX	--	---										
addiu \$t4,\$t4,1						BI	DI	EX	---	W									
nxt: addiu \$t0,\$t0,4							BI	DI	EX	↓--	W								
lw \$t1,0(\$t0)								BI	DI	↓EX	M	W							
j loop									BI	DI	--	---	--						
end: sw \$t4,0(\$t3)										BI	F	F	F	F					

Se assumir que tanto bne, beq e J tem o PC atualizado somente no quarto estágio.

Instrução	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
loop: blez \$t1,end	BI	DI	EX	---	---								BI						
sltiu \$t2,\$t1,65		BI	DI	EX	↓---	W													
bne \$t2,\$zero,nxt			BI	DI	↓EX	---	---												
sltiu \$t2,\$t1,91				BI	DI	EX	↓--	W											
beq \$t2,\$zero,nxt					BI	DI	↓EX	--	---										
addiu \$t4,\$t4,1						BI	DI	EX	---	W									
nxt: addiu \$t0,\$t0,4							BI	DI	EX	↓--	W								
lw \$t1,0(\$t0)								BI	DI	↓EX	M	W							
j loop									BI	DI	---	---	--						
end: sw \$t4,0(\$t3)										BI	F	F	F	F					

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)

**blez - Branch on less than or equal to zero**

**addiu -- Add immediate unsigned**

**sltiu -- Set on less than immediate unsigned**

tem 11 instruções

5a Questão) (2,0 ponto) Dado o trecho de programa abaixo, e assumindo uma implementação do MIPS com unidade de adiantamento, leitura após escrita no banco de registrador no mesmo ciclo e um preditor de saltos de 1 bit que inicialmente prevê salto não-realizado. O PC é escrito no quarto ciclo de relógio de cada instrução.

a) (1,6 ponto) Simule a execução completa do programa. Use o diagrama pipeline. Assuma para a simulação que a unidade de adiantamento é simples, ela pode apenas adiantar dados que já estejam no processador (na saída do estágio 3 ou do estágio 4) para a entrada da ULA. Ou seja, assuma que não é possível adiantar dados para o estágio 4.

root:	<b>Addi</b>	<b>\$t4, \$zero, 2</b>	<b>Conteúdos iniciais da memória e dos registradores relevantes:</b> \$t1=0x100, \$t2=0x100, \$t3=0x100, \$t4=0x100 Mem [0x100-0x103]= 0x002345AB Mem [0x200-0x203]= 0x0000000A Mem [0x300-0x303]= 0x00000000 Mem [0x400-0x403]= 0x00CD5F00
	<b>Add</b>	<b>\$t1, \$t2, \$t3</b>	
	<b>Lw</b>	<b>\$t3, 0x100(\$t1)</b>	
	<b>Sw</b>	<b>\$t3, 0x200(\$t1)</b>	
	<b>Subi</b>	<b>\$t4, \$t4, 2</b>	
	<b>Beq</b>	<b>\$t4, \$t3, root</b>	
	<b>Addi</b>	<b>\$t3, \$t3, 0x100</b>	

*Observe que será possível apenas realizar adiantamento para entrada da ULA.*

*Preditor 1 bit - Inicialmente salto não realizado (muda de estado se errar a predição)*

*9. ciclo - o preditor prevê que salto não será realizado salto*

*10. ciclo - comparação Rs x Rt - salto deve ser realizado, preditor muda de estado para salto realizado*

*11. ciclo - atualiza o PC,*

*12. ciclo - Flush em todas as instruções carregadas*

*19. ciclo - Preditor diz que salto realizado*

*20 ciclo - flush em todas as instruções carregadas, pois predito prediz salto*

*20. ciclo - Comparação Rs x RT, final do ciclo, verifica que salto não deve ser realizado salto*

*21. ciclo - PC atualizado*

*22. ciclo - flush em todas as instruções carregadas por erro do preditor*

Pelo enunciado, não é possível adiantar algo que não seja para entrada da ULA. Somente \$t1 vai para ULA.

Instrução.....	1.	2.	3.	4.	5.	6.	7	8.	9.	10.	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	
addi \$t4, \$zero, 2	BI	DI	EX	---	W					\$t4 = 0x002																	
add \$t1, \$t2, \$t3		BI	DI	EX	---	W				\$t1 = 0x100+0x100 = 0x200																	
lw\$t3, 0x100(\$t1)			BI	DI	EX	M	W			0x100 + 0x200 = 0x300, \$t3=0x000																	
sw \$t3, 0x200(\$t1)				BI	DI	X	X	EX	M	---		Mem [0x400-0x403]= 0x00000000															
subi \$t4, \$t4, 2					BI	X	X	DI	EX	---	W		\$t4 = 0x000														
beq \$t4, \$t3, root								BI	DI	EX	----	----		\$t4=0x000, \$t1=0x200													
addi \$t3, \$t3, 0x100									BI	DI	EX	F	F														
add \$t1, \$t2, \$t3												BI	DI	EX	---	W			\$t1 = 0x100+0x000 = 0x100								
lw\$t3, 0x100(\$t1)													BI	DI	EX	M	W			\$t3 = 0x00A							
sw \$t3, 0x200(\$t1)														BI	DI	X	X	EX	M	----		Mem [0x300-0x303]= 0x..00A					
subi \$t4, \$t4, 2															BI	X	X	DI	EX	M	W		\$t4 = 0x000				
beq \$t4, \$t3, root																		BI	DI	EX	----	W					
addi \$t3, \$t3, 0x100																			BI	F	F	F	F	Salto (preditor)			
add \$t1, \$t2, \$t3																				BI	DI	F	F	F			
lw\$t3, 0x100(\$t1)																					BI	F	F	F	F		
addi \$t3, \$t3, 0x100																						BI	DI	EX	---	W	

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)

16 linhas - cada linha correta 0,0875 (total 1,4)

2 adiantamento incorreto - 0.05

2 bolha - 0.1

não seguir notação - correção parcial (metade)

b) (0,3 ponto) No final do **sexto ciclo** de execução do trecho de programa, qual(ais) registradores estão sendo lidos e qual(ais) está(ão) sendo escrito(s) (lembre-se dos estágios em que estas operações ocorrem no pipeline!).

*Lido da memória: \$t3 (0,15 ponto)*

*Escrito no banco de registrador: \$t1 (0,15 ponto)*

c) (0,3 ponto) O que a unidade de adiantamento (forward) está fazendo durante o quarto ciclo de execução? Se algumas comparações estiverem sendo feitas, mencione-as.

*Não está fazendo nenhum adiantamento (0,15 ponto).*

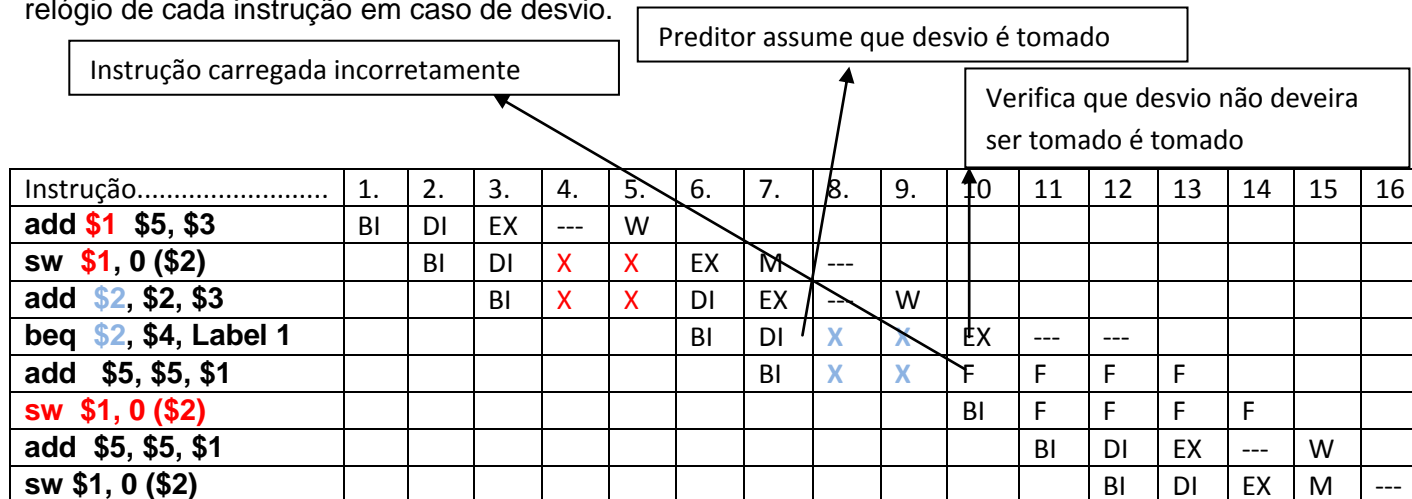
*Comparando os registradores-fonte da 2a instrução add \$t1, \$t2, \$t3 com os registradores-destino da 1a instrução da addi \$t4, \$zero, 2 (0,15 ponto)*

6a Questão) (2,0 ponto) Assuma que o seguinte código é executado sobre um processador pipeline com 5 estágios, com adiantamento e um preditor de desvio.

```

add $1, $5, $3
Label1: sw $1, 0($2)
add $2, $2, $3
beq $2, $4, Label 1 // Não tomado
add $5, $5, $1
sw $1, 0($2)
  
```

a) (0,75 ponto) Desenhe o diagrama de execução para este código, assumindo que **todo desvio é tomado** pelo preditor, que **não há unidade de adiantamento** e que o PC é escrito no terceiro ciclo de relógio de cada instrução em caso de desvio.



cada instrução correta 0,75 / 8

considerou nop - descontar 0.1

erro o numero de bolhas 0.15

bolha lugar errado - 1 - 0.1, 2 -0.15

adiantamento 0.1



b) (0,75 ponto) Desenhe o diagrama de execução para este código, assumindo que um preditor de saltos de 2 bit que inicialmente **prevê salto realizado**, que há **unidade de adiantamento** e que o PC é escrito no terceiro ciclo de relógio de cada instrução em caso de desvio.

Não pode ser resolvido via adiantamento

Instrução.....	1.	2.			3.	4.	5.	6.	7.	8.	9.	10	11	12	13	14	15
add \$1, \$5, \$3	BI	DI	EX	---	W												
sw \$1, 0(\$2)		BI	DI	X	X	EX	M	---									
add \$2, \$2, \$3			BI	X	X	DI	EX	---	W								
beq \$2, \$4, Label 1						BI	DI	EX	---	---							
add \$5, \$5, \$1							BI	F	F	F	F						
sw \$1, 0(\$2)								BI	F	F	F	F					
add \$5, \$5, \$1									BI	DI	EX	---	W				
sw \$1, 0(\$2)										BI	DI	EX	M	---			

cada instrução 0,75/8

Não adicionou bolha - 0.1

c) (0,5 ponto) Qual é o speed-up alcançado pelo item b) em relação ao item a).

se utilizou valores correto, com diagrama incorreto, considerado 0,5

$$speed\_up = \frac{16}{12} = 1.33$$