Organização e Arquitetura de Processadores

Organização do MIPS

Execução de Código Pipeline

Conflitos Estruturais, de Dados e de Controle

Prof. Dr. César Marcon

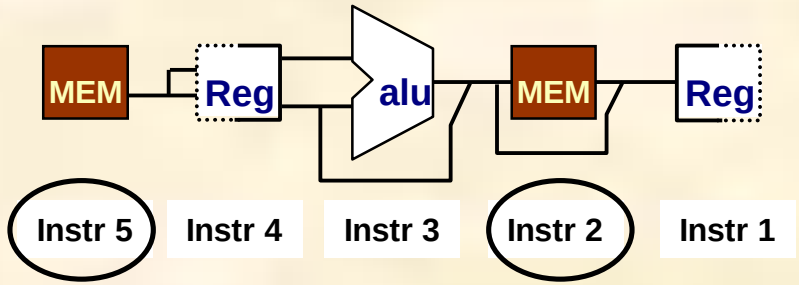
Conflito no Pipeline

- **Existem situações de execução no pipeline em que a instrução seguinte não pode ser executada no próximo ciclo de relógio. Tais situações são chamadas de conflitos (em inglês hazards)**

- **Tipos de Conflitos:**
  - **Estrutural**: HW não dá suporte a uma determinada combinação de instruções
  - **De controle**: Desvios no pipeline que mudam o apontador de programa, tornando imprevisível a descoberta da próxima instrução a ser executada no pipeline
  - **De dados**: Instrução no pipeline que depende do resultado de outra instrução logicamente anterior, que ainda não terminou sua execução

Conflito Estrutural (Structural Hazard)

- **Acontece quando o Hardware não pode suportar a combinação de instruções que o pipeline deseja executar em um dado ciclo de relógio**
- **Por exemplo:**
  - Se houvesse somente uma memória (para dados e instruções) e se uma instrução tentasse acessar um dado na memória enquanto tivesse que ser buscada uma nova instrução, ocorreria um conflito estrutural



Conflito de Controle (Control Hazard)

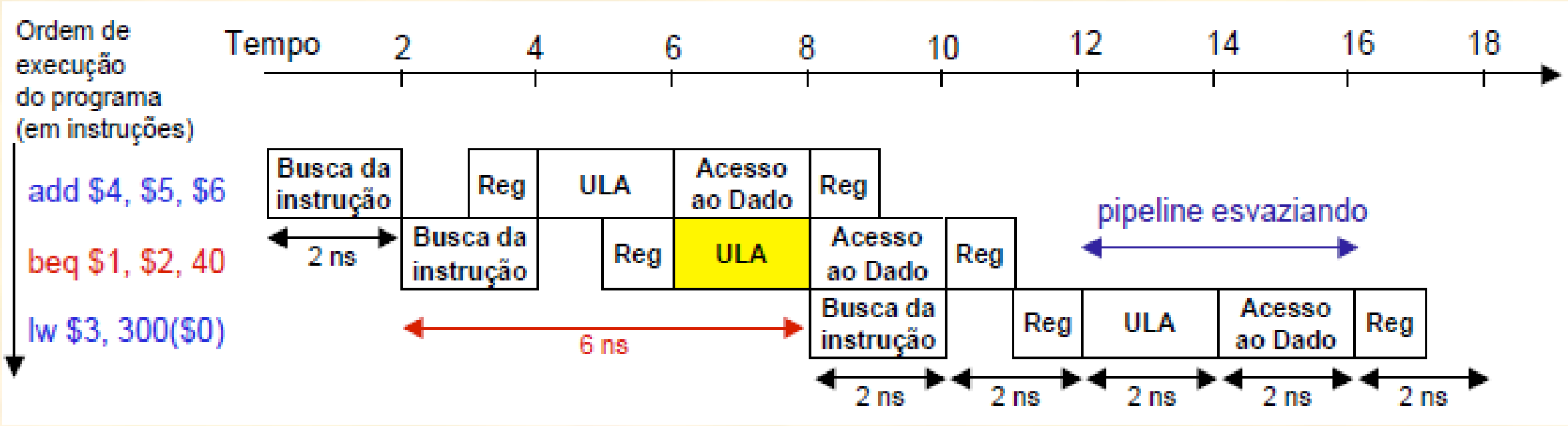
- **Origina da necessidade de buscar uma instrução baseado no resultado de uma instrução que ainda não foi concluída**
- **Caso típico de instruções de desvio condicional (e.g., beq), mas também em salto incondicional (e.g., jal)**
- **Soluções para conflito de controle:**
  - Limpar instruções lidas (flush) *– esta é a solução que usaremos aqui!*
  - Inserir instruções independentes antes dos desvios e saltos, e
  - Executar predição estática ou dinâmica
- **Flush – busca uma ou mais instruções, caso a busca for errada, remove a(s) instruç(ão/ões) do pipeline e busca a correta**

Pergunta:

Como faço para gerar um flush? ➔ Tipicamente substituir a instrução por NOP

Conflito de Controle

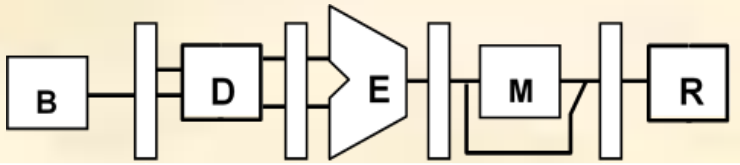
- **Exemplo: Se for necessário resolver um desvio condicional no estágio da ULA, devido à decisão do desvio ser ou não tomado, poderemos ter um atraso de 2 ciclos (4 ns)m além do ciclo de relógio necessário para sua execução (2 ns)**



Conflito de Controle

- **Exemplo de conflito de controle com a execução de um salto incondicional**

```
...

add ...
j pula_teste
    lw ...
    sub ...
    mul ...
pula_teste:
sw ...
la ...
...
```



	B	D	E	M	W
1	j	add			
2	lw	j	add		
3	sw	NOP	j	add	
4	la	sw	NOP	j	add
5		la	sw	NOP	j
6			la	sw	NOP
7				la	sw
8					la

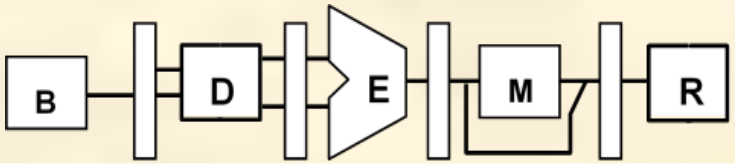
Flush

Conflito de Controle

- **Exemplo de conflito de controle com a execução de um desvio condicional**

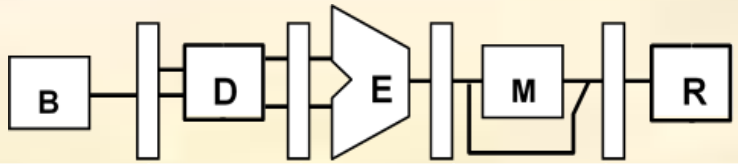
```
...

add ...
ble $t0, $t1, pula_teste
    lw ...
    sub ...
    mul ...
pula_teste:
sw ...
la ...
...
```

Salto Não Tomado



	B	D	E	M	W
1	ble	add			
2	lw	ble	add		
3	sub	lw	ble	add	
4	mul	sub	lw	ble	add
5		mul	sub	lw	ble
6			mul	sub	lw
7				mul	sub
8					mul

Salto Tomado



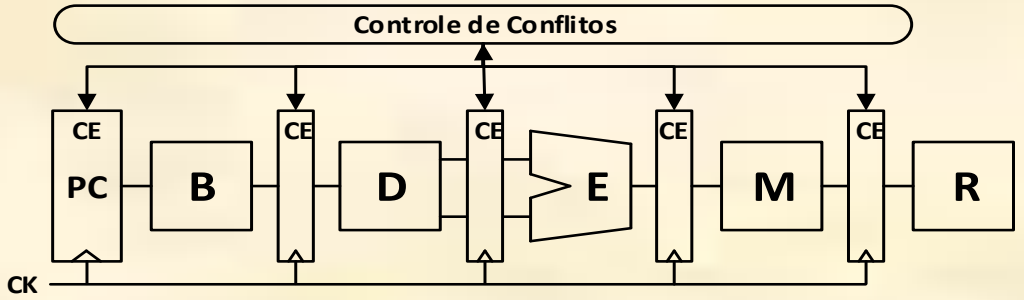
	B	D	E	M	W
1	ble	add			
2	lw	ble	add		
3	sub	lw	ble	add	
4	sw	**NOP**	**NOP**	ble	add
5	la	sw	**NOP**	**NOP**	ble
6		la	sw	**NOP**	**NOP**
7			la	sw	**NOP**
8				la	sw

Flush

Conflito de Dados (Data Hazard)

- **Conflito de dados é devido a uma dependência direta (ou verdadeira). Referente a uma leitura de operando cujo resultado ainda não foi computado**
  - Exemplo

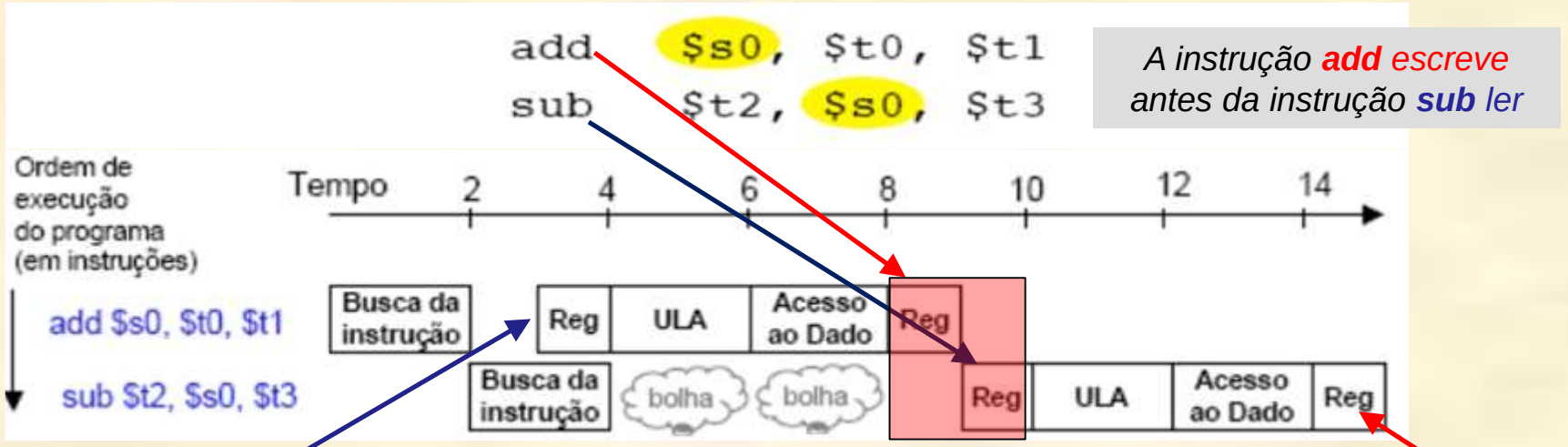
```
Inst_1:    ADD $t1, $t2, $t3
Inst_2:    SUB $t4, $t1, $t5
```



  - ADD está calculando o valor a ser armazenado no registrador $t1 e SUB realiza a leitura do valor contido no registrador $t1 para executar sua operação
  - Problema: quando SUB busca o registrador $t1 no segundo estágio do pipeline (representado por **D**), este ainda está sendo computado por ADD, ou seja, não foi escrito no banco de registradores (estágio representado por **R**)
- **Soluções para conflito de dados são:**
  - Inserção de bolhas – *esta é a solução que usaremos aqui!*
  - Utilização de circuitos de adiantamento dos dados (forwarding) – *próxima disciplina*
  - Reordenação de código – *próxima disciplina*

Conflito de Dados

- **Bolha – instrução do tipo NOP inserida no estágio subsequente à parada do programa**
- **Exemplo:**



Instrução			Ciclo							
			1	2	3	4	5	6	7	8
I0	add	$s0, $t0, $t1	**B**	**D**	**E**	**-**	**W**			
I1	sub	$t2, $s0, $t3		**B**	**D**	**X^D**	**X^D**	**E**	**-**	**W**

Escreve e lê no mesmo ciclo de relógio (em borda invertida)

Exercício

1. Execute o trecho de código abaixo colocando todos os estágios do pipeline (B, D, E, M, W). Caso um estágio não tenha sua unidade principal acessada, represente por '-'. Bolhas podem ser representadas por 'X' ou 'X^K' onde K significa o estágio onde instrução está parada. Considere que, caso haja uma dependência, esta deve ser resolvida com bolhas

Instrução	Ciclo											
	1	2	3	4	5	6	7	8	9	10	11	12
lui $s1, 0												
addi $s1, $s1, 2												
lui $s2, 0												
sw $s1, (0x11000000) $s3												
sw $s2, (0x11000000) $s3												

Resposta

1. **Execute o trecho de código abaixo colocando todos os estágios do pipeline (B, D, E, M, W). Caso um estágio não tenha sua unidade principal acessada, represente por '-'. Bolhas podem ser representadas por 'X' ou 'X^{K}' onde K significa o estágio onde instrução está parada. Considere que, caso haja uma dependência, esta deve ser resolvida com bolhas**

Instrução	Ciclo											
	1	2	3	4	5	6	7	8	9	10	11	12
lui $s1, 0	B	D	E	-	W							
addi $s1, $s1, 2		B	D	X^{D}	X^{D}	E	-	W				
lui $s2, 0			B	X^{B}	X^{B}	D	E	-	W			
sw $s1, (0x11000000) $s3						B	D	X^{D}	E	M	-	
sw $s2, (0x11000000) $s3							B	X^{B}	D	E	M	-

Exercício

2. **(Baseado no POSCOMP 2005 - 21)** Considere uma CPU usando pipeline com 5 estágios (B, D, E, M, W), com memórias de dados e de instruções separadas, escrita no banco de registradores na borda de subida do relógio e leitura na borda de descida do relógio e o conjunto de instruções a seguir:

I1: lw $2, 100($5)
I2: add $1, $2, $3
I3: sub $3, $2, $1
I4: sw $2, 50($1)
I5: add $2, $3, $3
I6: sub $2, $2, $4

Quantos ciclos são gastos para a execução deste código?

a) 30
b) 17
c) 16
d) 11
e) 10

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
lw $2, 100($5)																		
add $1, $2, $3																		
sub $3, $2, $1																		
sw $2, 50($1)																		
add $2, $3, $3																		
sub $2, $2, $4																		

Resposta

2. **(Baseado no POSCOMP 2005 - 21)** Considere uma CPU usando pipeline com 5 estágios (B, D, E, M, W), com memórias de dados e de instruções separadas, escrita no banco de registradores na borda de subida do relógio e leitura na borda de descida do relógio e o conjunto de instruções a seguir:

I1: lw $2, 100($5)
I2: add $1, $2, $3
I3: sub $3, $2, $1
I4: sw $2, 50($1)
I5: add $2, $3, $3
I6: sub $2, $2, $4

Quantos ciclos são gastos para a execução deste código?

a) 30
b) 17
c) 16
d) 11
e) 10

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
lw $2, 100($5)	B	D	E	M	W													
add $1, $2, $3		B	D	d*	d*	E	-	W										
sub $3, $2, $1			B	b*	b*	D	d*	d*	E	-	W							
sw $2, 50($1)						B	b*	b*	D	E	M	-						
add $2, $3, $3									B	D	d*	E	-	W				
sub $2, $2, $4										B	b*	D	d*	d*	E	-	W	

Exercício

3. **(POSCOMP 2008 - 54)** Um processador tem cinco estágios de pipeline. Suponha que cada uma das etapas do processador (busca, decodificação, execução, leitura ou escrita de dados em memória e escrita em registrador) seja executada em 5ns. O tempo total para que 5 instruções sejam executadas em pipeline, supondo que não haja dependência de dados entre as instruções é:

a) 15ns
b) 25ns
c) 30ns
d) 45ns
e) 50ns

Resposta

3. **(POSCOMP 2008 - 54)** Um processador tem cinco estágios de pipeline. Suponha que cada uma das etapas do processador (busca, decodificação, execução, leitura ou escrita de dados em memória e escrita em registrador) seja executada em 5ns. O tempo total para que 5 instruções sejam executadas em pipeline, supondo que não haja dependência de dados entre as instruções é:

a) 15ns
b) 25ns
c) 30ns
d) 45ns
e) 50ns

Exercício

4. Ilustre a execução do trecho de código no MIPS pipeline, colocando os estágios e bolhas e flush necessários para sua execução correta. Considere que o código inicia com $t1 = 3, $s0 = &vet[0] e $t2 = 4. Considere também que vet é um vetor de 4 inteiros iniciados com todas as posições em 0 e endereço de memória 0x100. Ao final apresente o conteúdo de vet e o valor dos registradores utilizados. OBS. Coloque F em um estágio que está em flush

Instrução	Ciclo																			
`ini: addi $t1, $t1, 1`																				
`bgt $t1, $t2, fim`																				
`sw $t1, 0($s0)`																				
`addi $s0, $s0, 4`																				
`j ini`																				
`fim: li $v0, 10`																				
`syscall`																				

`vet[] = { , , , }` `$t1 =` `$t2 =` `$s0 =` `$v0 =`

Resposta

4. Ilustre a execução do trecho de código no MIPS pipeline, colocando os estágios e bolhas e flush necessários para sua execução correta. Considere que o código inicia com $t1 = 3, $s0 = &vet[0] e $t2 = 4. Considere também que vet é um vetor de 4 inteiros iniciados com todas as posições em 0 e endereço de memória 0x100. Ao final apresente o conteúdo de vet e o valor dos registradores utilizados. OBS. Coloque F em um estágio que está em flush

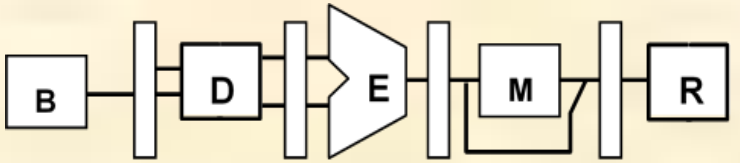
Instrução	Ciclo																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
ini: addi $t1, $t1, 1	B	D	E	-	W				B	D	E	-	W								
bgt $t1, $t2, fim		B	D	X^D	X^D	E	-	-		B	D	X^D	X^D	E	-	-					
sw $t1, 0($s0)			B	X^B	X^B	D	E	M	-		B	X^B	X^B	D	F	F	F				
addi $s0, $s0, 4						B	D	E	-	W				B	F	F	F	F			
j ini							B	D	-	-	-										
fim: li $v0, 10								B	F	F	F	F			B	D	E	-	W		
syscall																B	D	E	-	-	

`vet[] = {4, 0, 0, 0}` `$t1 = 5` `$t2 = 4` `$s0 = 0x104` `$v0 = 10`

Exercício

5. Um diagrama que relaciona ciclos de instrução com estágios do pipeline, tal como ilustrado na direita, permite compreender melhor a execução em um pipeline

- Reescreva a execução abaixo no formato proposto
- Diga quantas bolhas foram inseridas devido aos dados e devido ao controle
- Quantos ciclos a mais ocorreram por ter hazards de controle e dados?
- Qual a fórmula básica do pipeline sem considerar hazards? Entenda e comente o efeito das bolhas de dados e de controle na fórmula



Instrução	Ciclo																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ini: addi $t1, $t1, 1	B	D	E	-	W				B	D	E	-	W							
bgt $t1, $t2, fim		B	D	X^D	X^D	E	-	-		B	D	X^D	X^D	E	-	-				
sw $t1, 0($s0)			B	X^B	X^B	D	E	M	-		B	X^B	X^B	D	F	F	F			
addi $s0, $s0, 4						B	D	E	-	W				B	F	F	F	F		
j ini							B	D	-	-	-									
fim: li $v0, 10								B	F	F	F	F			B	D	E	-	W	
syscall																B	D	E	-	-

Bolhas(dados) = , Bolhas(controle) = , #ciclosAdicionais =

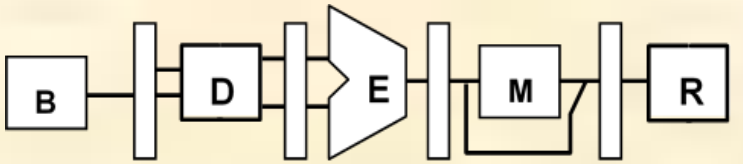
Fórmula =

	B	D	E	M	W
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

Resposta

5. Um diagrama que relaciona ciclos de instrução com estágios do pipeline, tal como ilustrado na direita, permite compreender melhor a execução em um pipeline

- Reescreva a execução abaixo no formato proposto
- Diga quantas bolhas foram inseridas devido aos dados e devido ao controle
- Quantos ciclos a mais ocorreram por ter hazards de controle e dados?
- Qual a fórmula básica do pipeline sem considerar hazardas? Entenda e comente o efeito das bolhas de dados e de controle na fórmula



Instrução	Ciclo																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ini: addi $t1, $t1, 1	B	D	E	-	W				B	D	E	-	W							
bgt $t1, $t2, fim		B	D	X^D	X^D	E	-	-		B	D	X^D	X^D	E	-	-				
sw $t1, 0($s0)			B	X^B	X^B	D	E	M	-		B	X^B	X^B	D	F	F	F			
addi $s0, $s0, 4						B	D	E	-	W				B	F	F	F	F		
j ini							B	D	-	-	-									
fim: li $v0, 10								B	F	F	F	F			B	D	E	-	W	
syscall																B	D	E	-	-

Bolhas(dados) = 4, Bolhas(controle) = 3, #ciclosAdicionais = 7

Fórmula = #E + (#I-1), com #E = #estágios do pipeline,
#I = #instruções buscadas
= 5 + (12-1) = 5 + 11 = 16 ciclos ➔ Deveria ser 13

➔ Hazard de controle busca instrução indevida, dados apenas insere NOP

	B	D	E	M	W
1	addi				
2	bgt	addi			
3	sw	bgt	addi		
4	sw	bgt	*NOP*	addi	
5	sw	bgt	*NOP*	*NOP*	addi
6	addi	sw	bgt	*NOP*	*NOP*
7	j	addi	sw	bgt	*NOP*
8	li	j	addi	sw	bgt
9	addi	*NOP*	j	addi	sw
10	bgt	addi	*NOP*	j	addi
11	sw	bgt	addi	*NOP*	j
12	sw	bgt	*NOP*	addi	*NOP*
13	sw	bgt	*NOP*	*NOP*	addi
14	addi	sw	bgt	*NOP*	*NOP*
15	li	*NOP*	*NOP*	bgt	*NOP*
16	syscall	li	*NOP*	*NOP*	bgt
17		syscall	li	*NOP*	*NOP*
18			syscall	li	*NOP*
19				syscall	li
20					syscall

Exercício

6. Dado o trecho de código abaixo em linguagem de montagem do MIPS, preencha o diagrama de execução abaixo.

Convenção: X^k [bolha no estágio k], F [Flush], - [estágio sem operação], B [Busca], D [Decodificação], E [Execução], M [operação com a memória de dados], W [Write-back]

```
Conteúdos iniciais da memória e dos registradores relevantes:
    $t0=0x100, $t2=0x000, $t3=0x100, $t4=0x000
    Mem [0x100-0x103] = 0x00000001
    Mem [0x200-0x203] = 0x00000010
    Mem [0x300-0x303] = 0x00000100
    Mem [0x400-0x403] = 0x00000001
    Mem [0x500-0x503] = 0x00000010
```

Reg	Instrução	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	ori $t1, $zero, 0x100																				
	addi $t0, $zero, 1																				
	laco: add $t2, $t3, $t1																				
	lw $t4, 0x100($t2)																				
	sw $t0, 0x200($t2)																				
	subi $t0, $t0, 1																				
	bne $t0, $zero, laco																				
	lw $t2, 0x400($t1)																				
	add $t4, $t2, $t0																				

Resposta

6. Dado o trecho de código abaixo em linguagem de montagem do MIPS, preencha o diagrama de execução abaixo.

Convenção: X^k [bolha no estágio k], F [Flush], - [estágio sem operação], B [Busca], D [Decodificação], E [Execução], M [operação com a memória de dados), W [Write-back]

```
Conteúdos iniciais da memória e dos registradores relevantes:
    $t0=0x100, $t2=0x000, $t3=0x100, $t4=0x000
    Mem [0x100-0x103] = 0x00000001
    Mem [0x200-0x203] = 0x00000010
    Mem [0x300-0x303] = 0x00000100
    Mem [0x400-0x403] = 0x00000001
    Mem [0x500-0x503] = 0x00000010
```

Reg	Instrução	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$t1 = 100	ori $t1, $zero, 0x100	B	D	E	-	W															
$t0 = 1	addi $t0, $zero, 1		B	D	E	-	W														
$t2 = 200	laco: add $t2, $t3, $t1			B	D	X^D	E	-	W												
$t4 = 100	lw $t4, 0x100($t2)				B	X^B	D	X^D	X^D	E	M	W									
	sw $t0, 0x200($t2)						B	X^B	X^B	D	E	M	-								
$t0 = 0	subi $t0, $t0, 1									B	D	E	-	W							
	bne $t0, $zero, laco										B	D	X^D	X^D	E	-	-				
$t2 = 10	lw $t2, 0x400($t1)											B	X^B	X^B	D	E	M	W			
$t4 = 10	add $t4, $t2, $t0														B	D	X^D	X^D	E	-	W