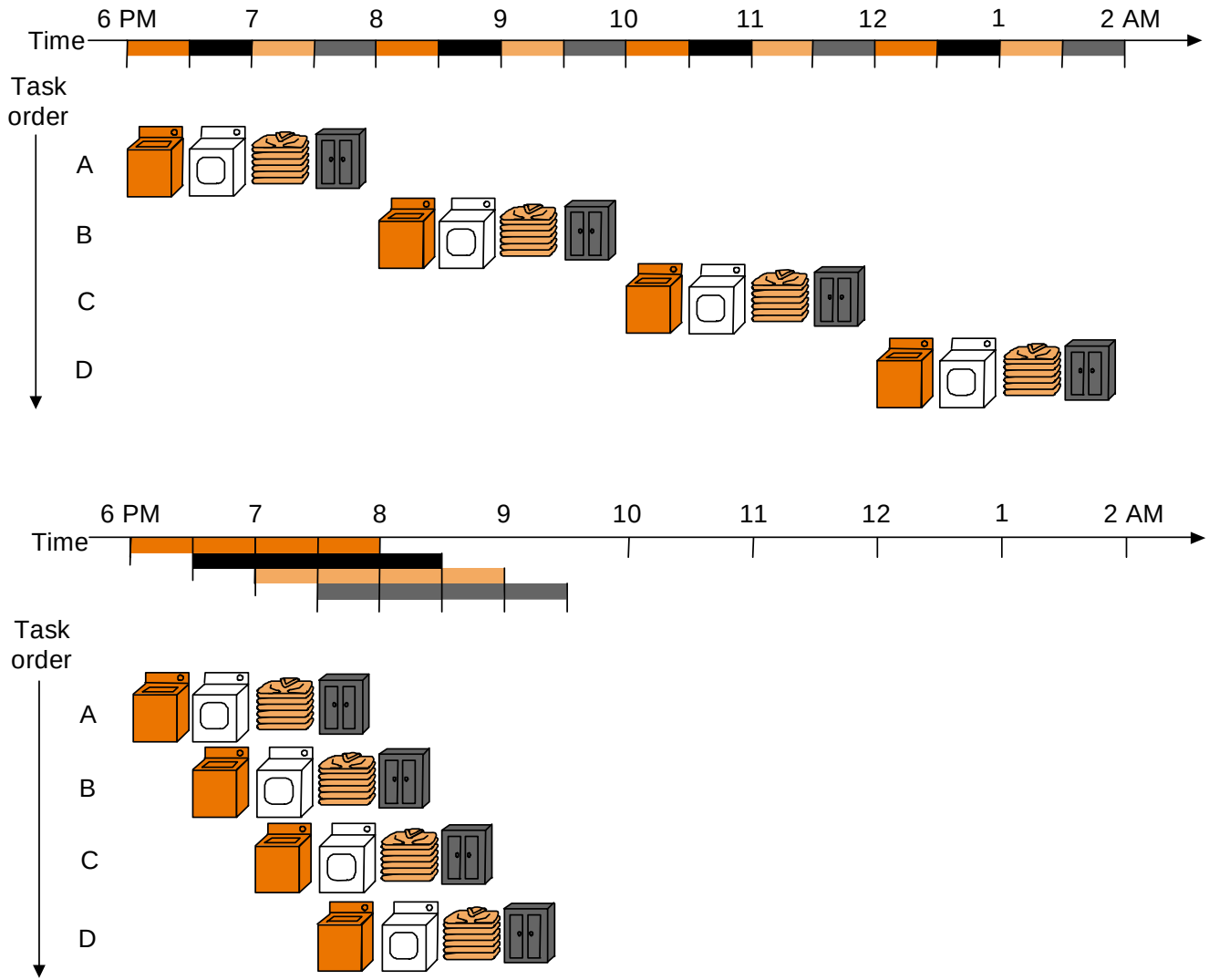
Pipeline

Lavanderia – analogia com o pipelining



Pipeline de instruções no MIPS

- Busca da instrução
- Leitura dos registradores e decodificação
- Execução da operação ou cálculo de endereço
- Acesso ao operando na memória
- Escrita do resultado em um registrador

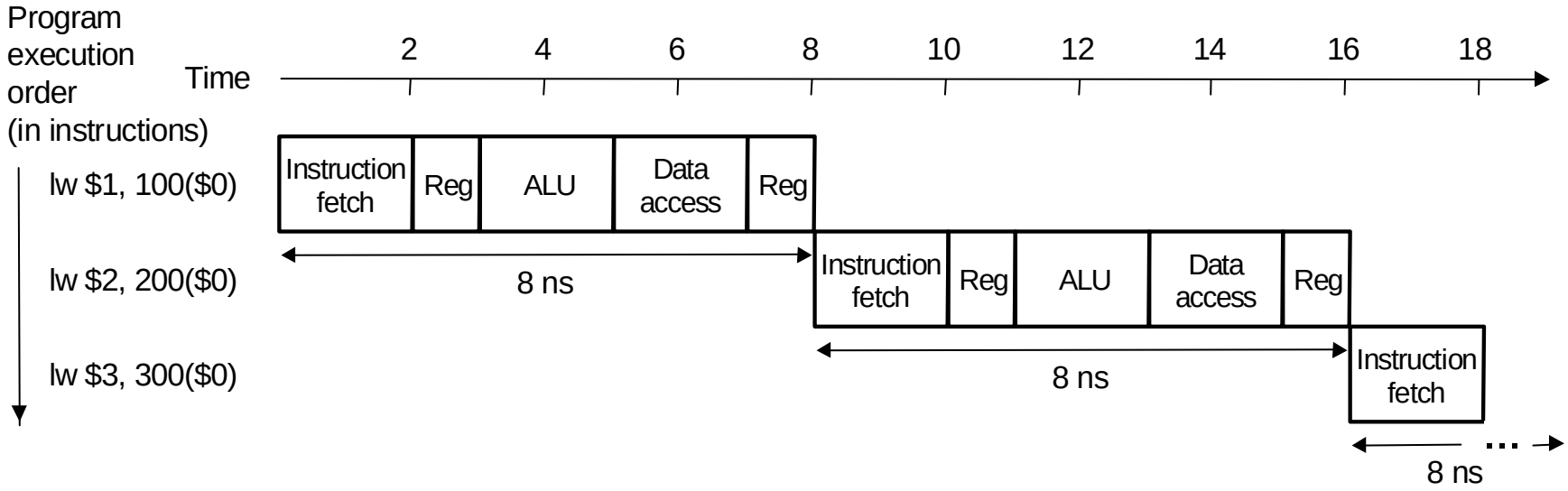
Exemplo

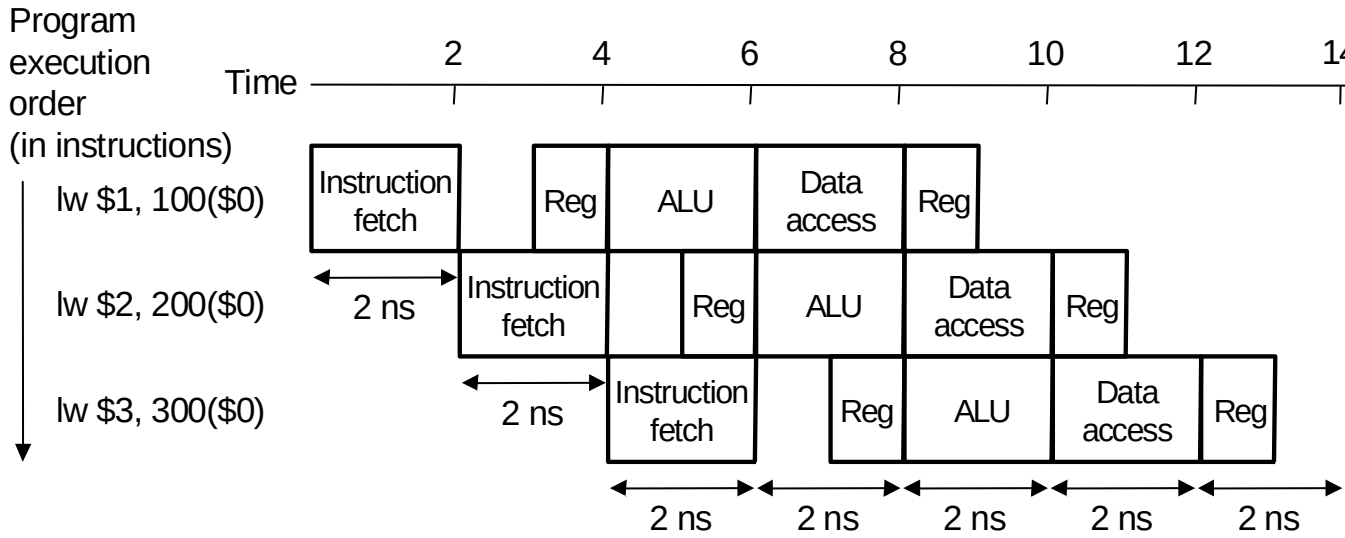
- **Compare o tempo médio entre instruções da implementação em single-cycle (uma instrução por ciclo) com uma implementação com pipeline.**
- **Supor maior tempo de operação para acesso à memória = 2ns, operação da ULA = 2ns e acesso ao register file = 1ns. (Instrs lw, sw, add, sub, and, or slt e beq).**

Tempo total para as oito instruções calculado a partir do tempo de cada componente

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (`lw`)	2 ns	1 ns	2 ns	2 ns	1 ns	8 ns
Store word (`sw`)	2 ns	1 ns	2 ns	2 ns		7 ns
R-format (`add, sub, and, or, slt`)	2 ns	1 ns	2 ns		1 ns	6 ns
Branch (`beq`)	2 ns	1 ns	2 ns			5 ns

Execução não-pipeline X pipeline





OBS.:

- **Sob condições ideais, com estágios balanceados, o speedup do pipeline é igual ao número de estágios do pipeline (5 estágios , 5 vezes mais rápido)**
- **Na realidade o tempo de execução de uma instrução é um pouco superior (overheads) ➔ speedup é menor que o número de estágios do pipeline**

Suponha a execução de 1000 instruções

Desempenho do pipeline é devido ao aumento do throughput.

com pipeline ➔
1000 X 2.02ns = 2020 (para cada instrução adiciono 10% overhead)

sem pipeline ➔ 1000 X 8ns = 8000

spedup = 8000 / 2020 = 3.96 ~~ 8 / 2

Projeto de um conjunto de instruções para pipeline

- O que torna a implementação mais fácil
  - Instruções de mesmo tamanho
  - Poucos formatos, com campos de registradores sempre dispostos no mesmo lugar (Simetria, no 2º estágio podemos ler registradores e decodificar ao mesmo tempo).
  - Acesso à memória apenas com as instruções lw e sw.
  - Operandos alinhados na memória: o dado pode ser transferido da memória para a CPU e CPU para a memória em um único estágio do pipeline.

Projeto de um conjunto de instruções para pipeline

- O que torna a implementação mais dificil
  - Hazard
    - Hazard Estrutural
    - Hazard de Controle
    - Hazard de Dados

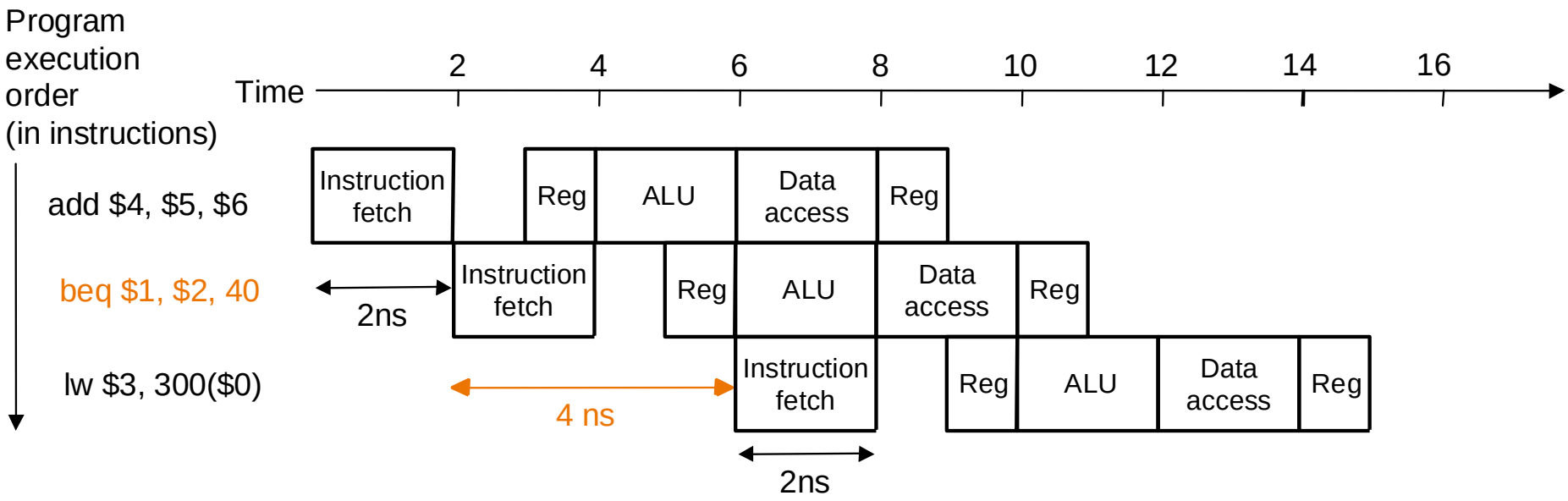
Pipeline Hazards

- Hazard Estrutural
  - O hardware não suporta uma combinação de instruções que queremos executar em um único período de clock
    - Ex.: escrever e ler da memória em um mesmo ciclo

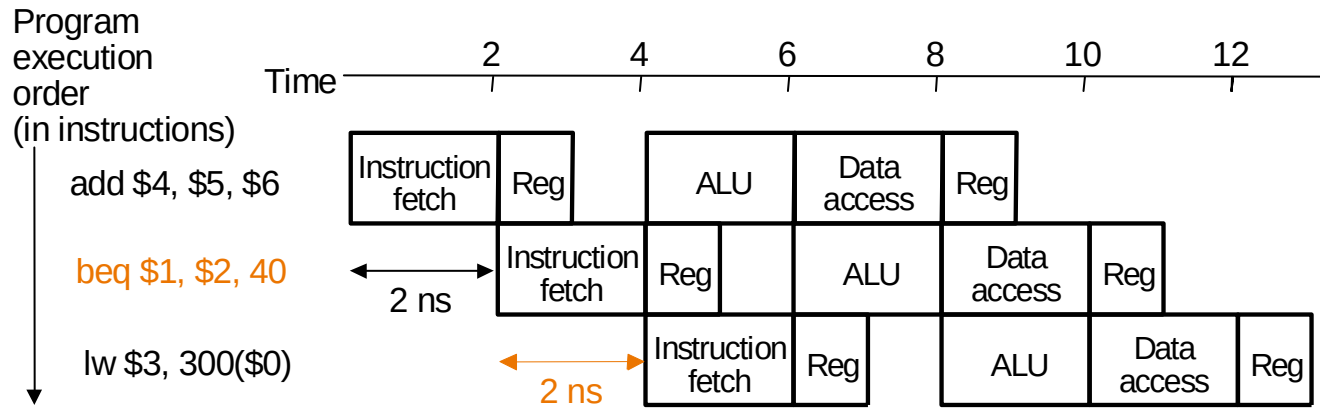
Pipeline Hazards

- Hazard de Controle
  - Problemas devido à execução de instruções de desvio
    - Ex.: Quando um branch é tomado, como tratar a(s) instruções que seguem (fisicamente) o branch no programa e que já estão no pipeline

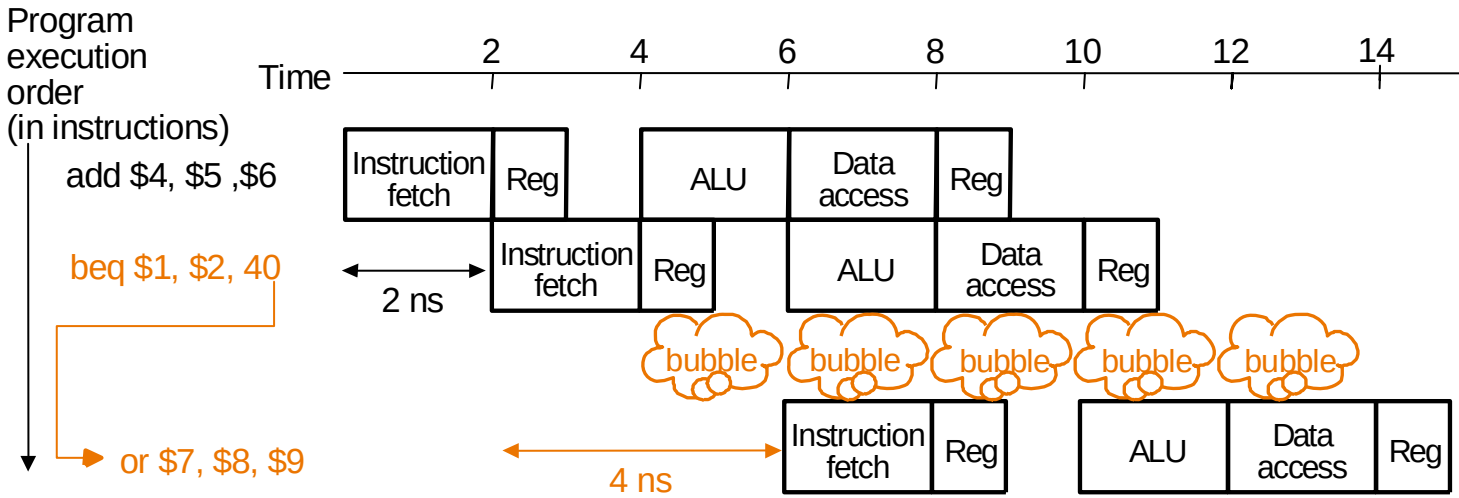
Pipelining stalling para instruções branch



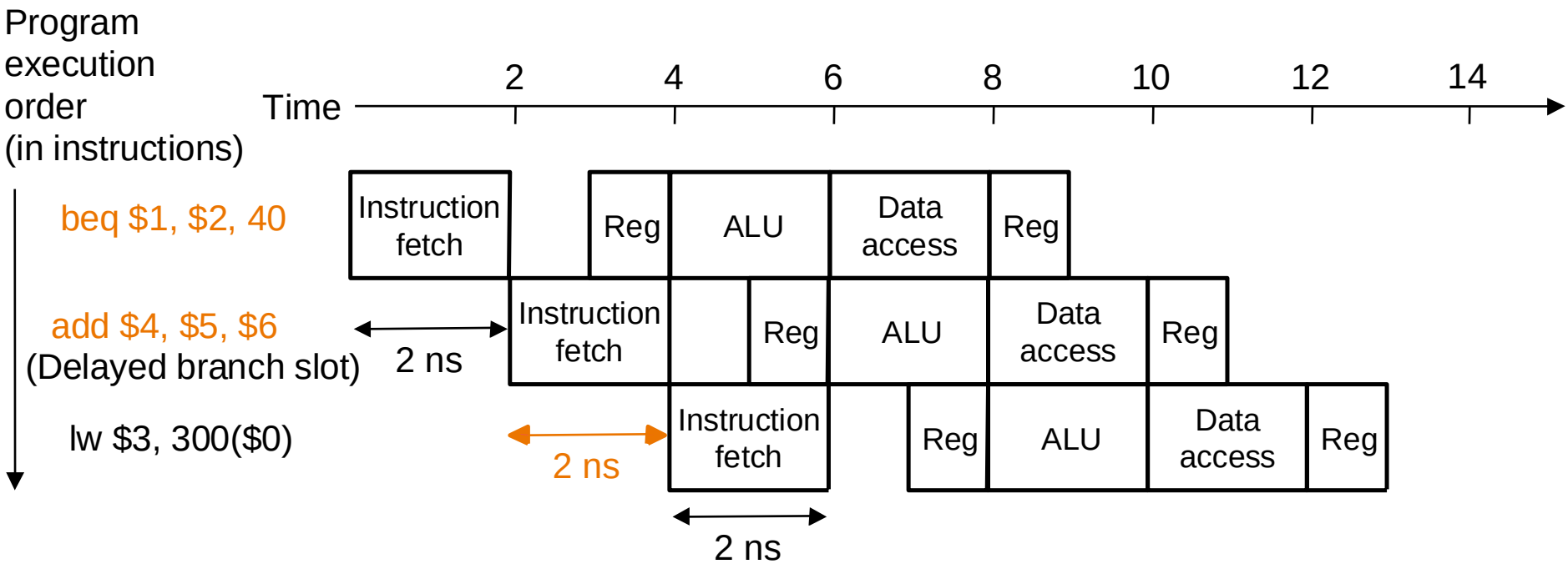
Branch prediction: Tentar "adivinhar" qual dos caminhos do branch será tomado



O branch não será tomado



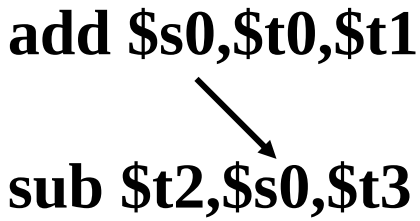
Pipeline delayed branch



Hazard de Dados

- Quando uma instrução necessita de um dado que ainda não foi calculado

  - Ex.:



Soluções :

Compilador (programador) gera código livre de data hazard (introduzindo, por ex., instruções nop no código; alterando a ordem das instruções; ...)

Stall; Forwarding ou bypassing

Principais conflitos que iremos utilizar e como deverão ser tratados

	1	2	3	4	5	6	7	8	9	10
add $s1, $s2, $s3	if	id	ex	me	wb					
add $s4, $s5, $s1		if			id					
lw $s1, 10($s2)	if	id	ex	me	wb					
add $s4, $s5, $s1		if			id					
do: add $s1, $s2, $s3	if	id	ex	me	wb					
add $s4, $s5, $s6		if	id	ex	me	wb				
bne $s7, $s8, do			if	id	ex	me	wb			
Segundo loop ->					if	id	ex	me	wb	

Posteriormente (AC3) algumas soluções serão mostradas

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
addi $s1, $0, 3																					
do: addi $s1, $S1, -1																					
addi $s2, $0, 10																					
addi $s3, $0, 10																					
bne $s1, $0, do																					
nop																					

	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	27	28	29	30	31	32
addi $s1, $0, 3	if	id	ex	me	wb																											
do: addi $s1, $S1, -1		if			id	ex	me	wb																								
addi $s2, $0, 10					if	id	ex	me	wb																							
addi $s3, $0, 10						if	id	ex	me	wb																						
bne $s1, $0, do							if	id	ex	me	wb																					
nop								if	id																							
do: addi $s1, $S1, -1									if	id	ex	me	wb																			
addi $s2, $0, 10										if	id	ex	me	wb																		
addi $s3, $0, 10											if	id	ex	me	wb																	
bne $s1, $0, do												if	id	ex	me	wb																
nop													if	id																		
do: addi $s1, $S1, -1														if	id	ex	me	wb														
addi $s2, $0, 10															if	id	ex	me	wb													
addi $s3, $0, 10																if	id	ex	me	wb												
bne $s1, $0, do																	if	id	ex	me	wb											
nop																		if	id	ex	me	wb										

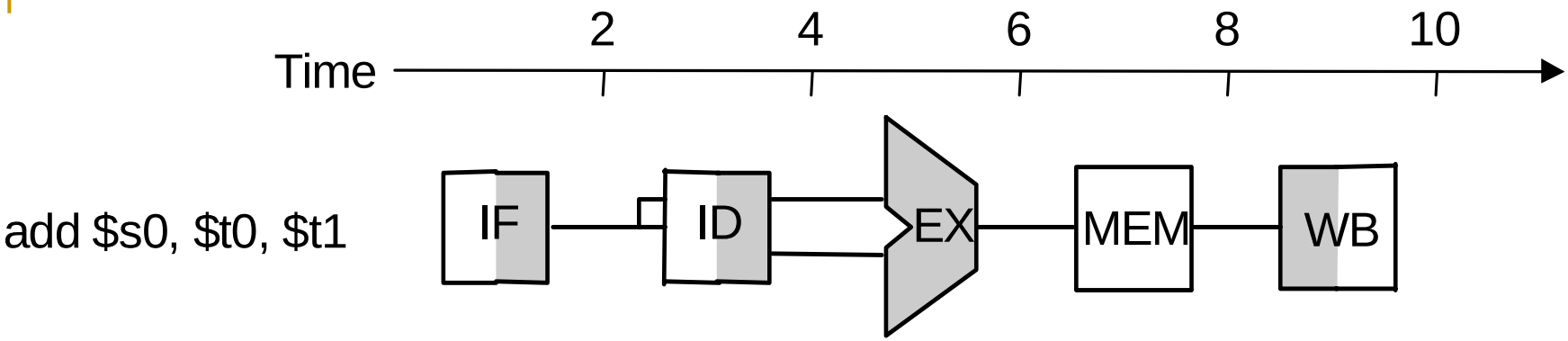
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	1 +																															
loop para 3																						1										
loop para 4																											1					
loop para 5																																1
Fórmula para 3 =	1+7+2*5+3+1 = 22																															
Fórmula para 4 =	1+7+3*5+3+1 = 27																															
Fórmula para 5 =	1+7+4*5+3+1 = 32																															
Fórmula para 100 =	1+7+99*5+3+1																															
Fórmula para n =	1 + 7 + (n-1)*5 + 3 + 1																															

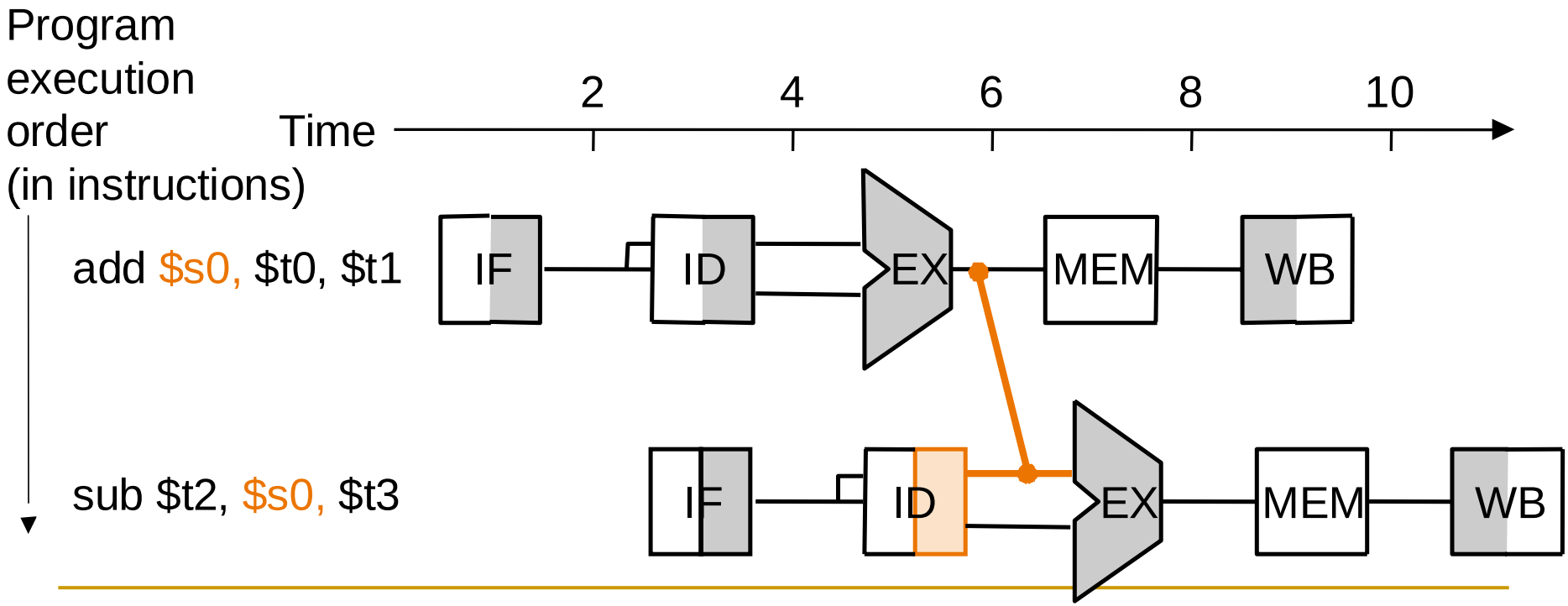
Desenrolando o Loop

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
addi $s1, $0, 3	if	id	ex	me	wb												
addi $s1, $S1, -1		if			id	ex	me	wb									
addi $s2, $0, 10					if	id	ex	me	wb								
addi $s3, $0, 10						if	id	ex	me	wb							
addi $s1, $S1, -1							if	id	ex	me	wb						
addi $s2, $0, 10								if	id	ex	me	wb					
addi $s3, $0, 10									if	id	ex	me	wb				
addi $s1, $S1, -1										if	id	ex	me	wb			
addi $s2, $0, 10											if	id	ex	me	wb		
addi $s3, $0, 10												if	id	ex	me	wb	
nop													if	id	ex	me	wb

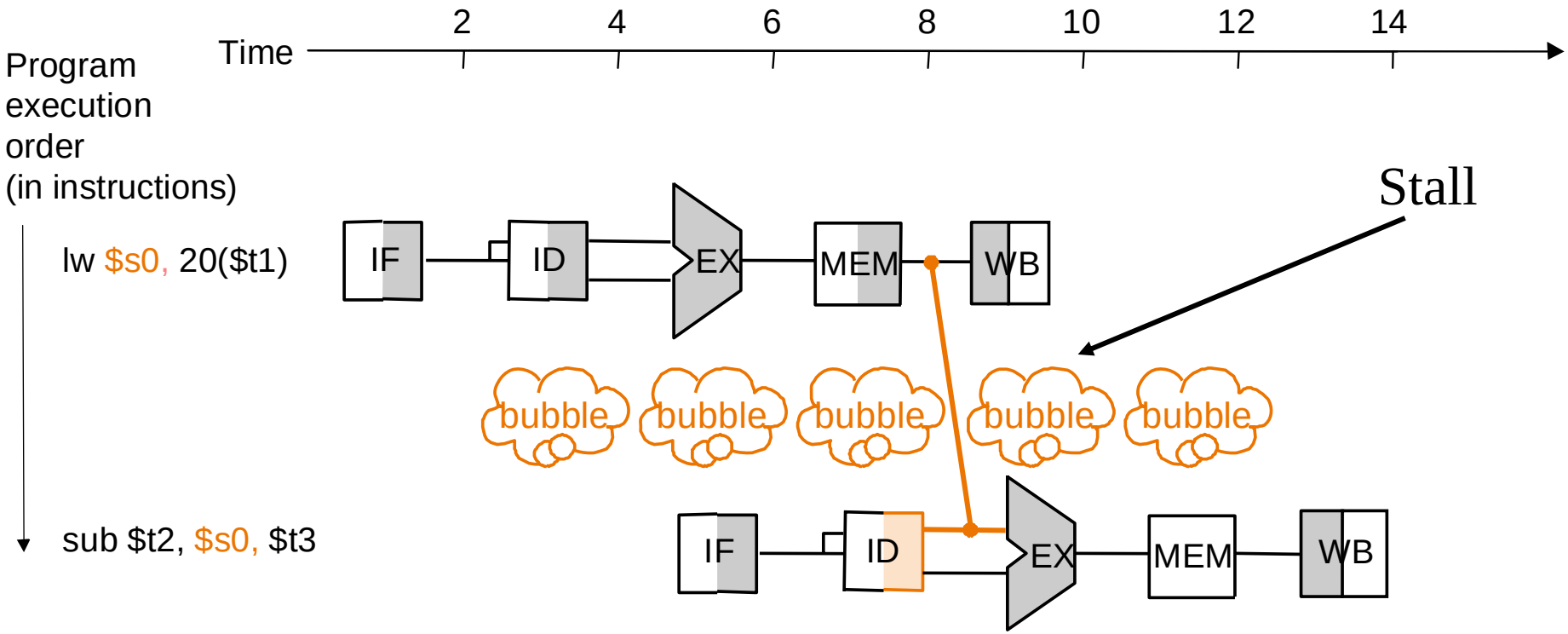
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Hazard de Dados





Hazard de Dados



Exemplo

- Encontre o hazard no código abaixo e resolva-o:

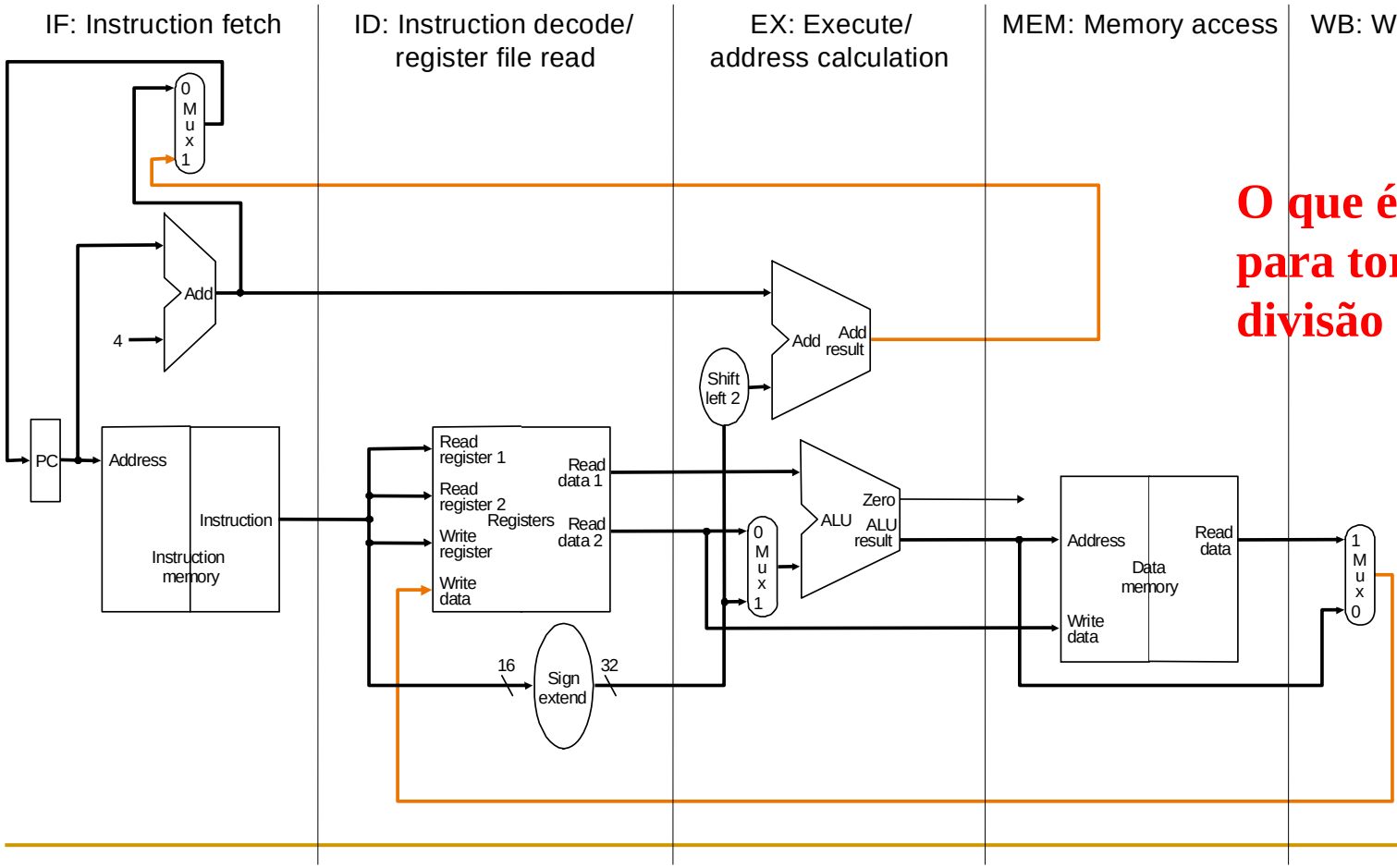
```
                                # $t1 tem o end. de v[k]
lw      $t0, 0($t1)             # $t0 = v[k]
lw      $t2,4($t1)              # $t2 = v[k+1]
sw      $t2, 0($t1)             # v[k] = $t2
sw      $t0, 4($t1)             # v[k+1] = $t0
```

Solução:

```
                                # $t1 tem o end. de v[k]
lw   $t0, 0($t1)                # $t0 = v[k]
     lw      $t2,4($t1)         # $t2 = v[k+1]
     sw      $t0, 4($t1)        # v[k+1] = $t0
     sw      $t2, 0($t1)        # v[k] = $t2
```

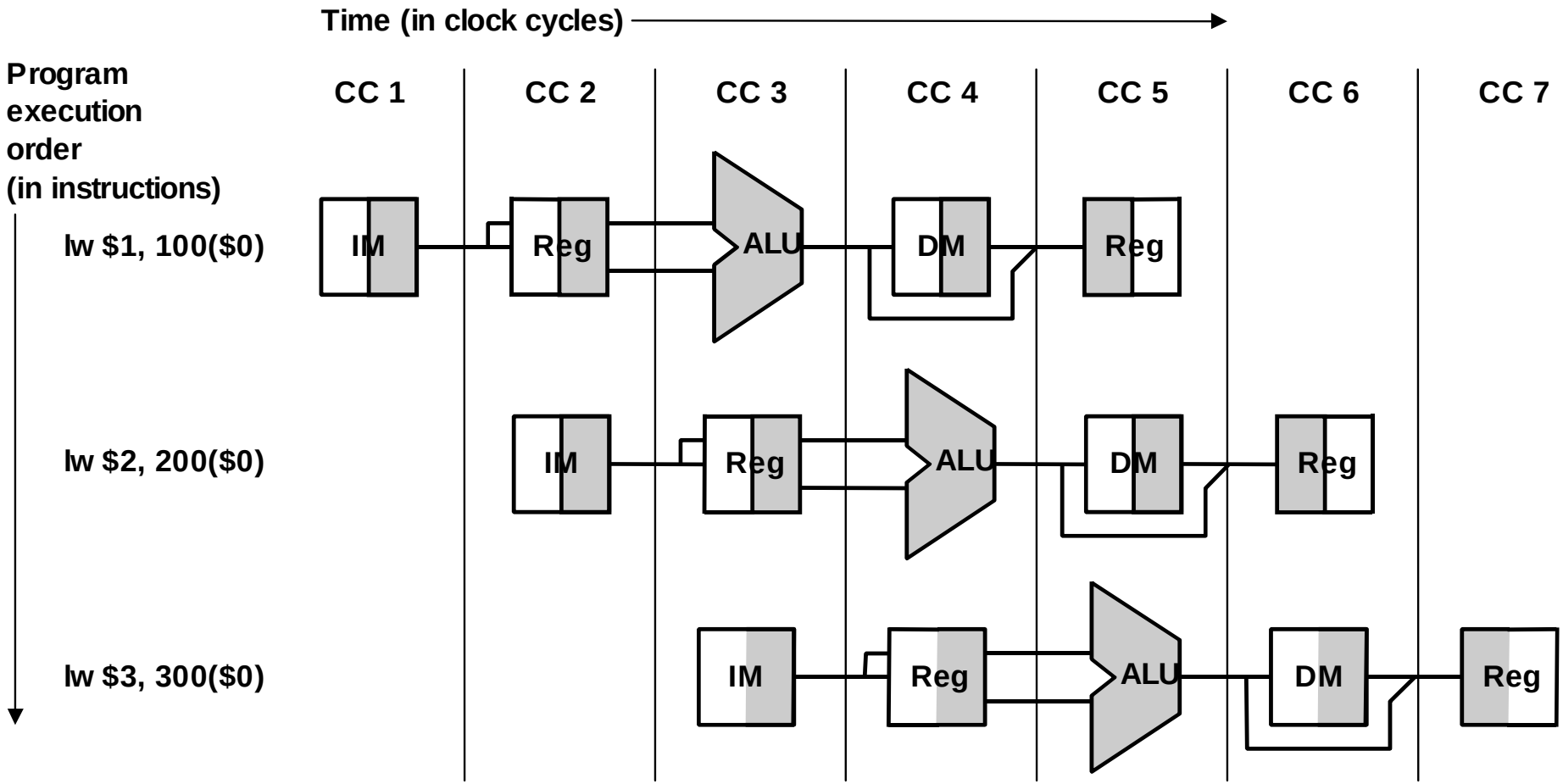
Pipeline: Idéia Básica

- 5 estágios: Busca; Decodificação e leitura dos regs; execução ou cálculo de end. ; acesso à memória; escrita no reg. destino



O que é necessário para tornar cada divisão em estágios?

Instruções sendo executadas pelo datapath



Pipelined Datapath