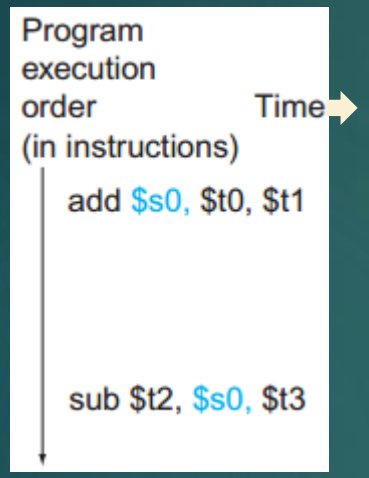
An Overview of Pipelining

DR. RAJIB RANJAN MAITI
CSIS, BITS-PILANI, HYDERABAD

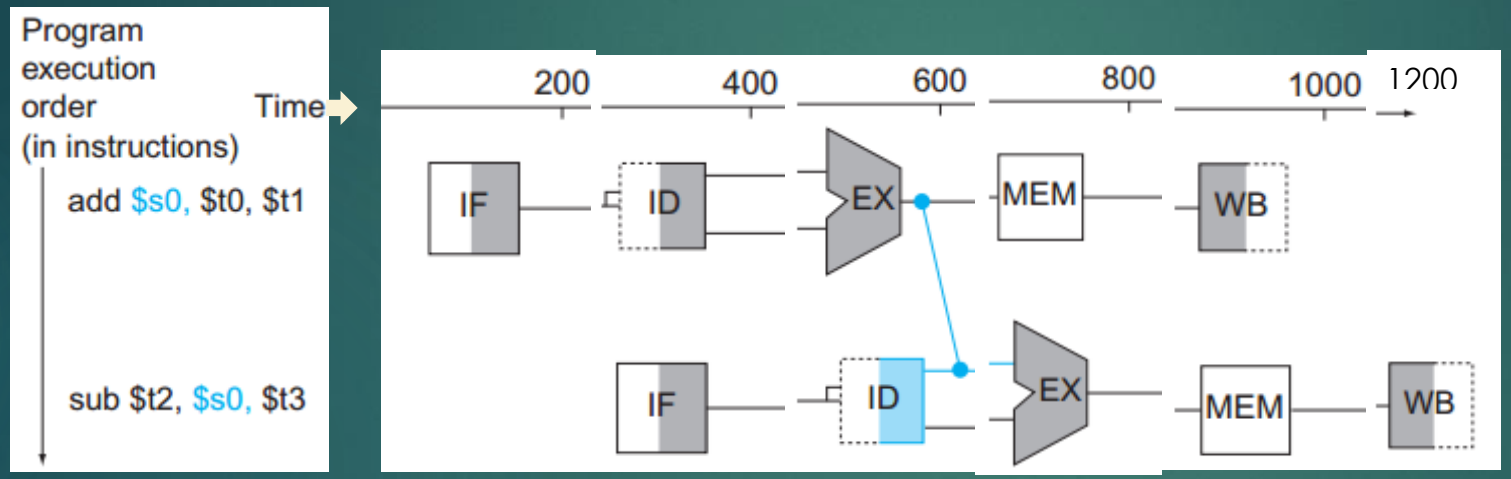
Data Hazards: Solution

Solution: forwarding



Data Hazards: Solution

Solution: forwarding

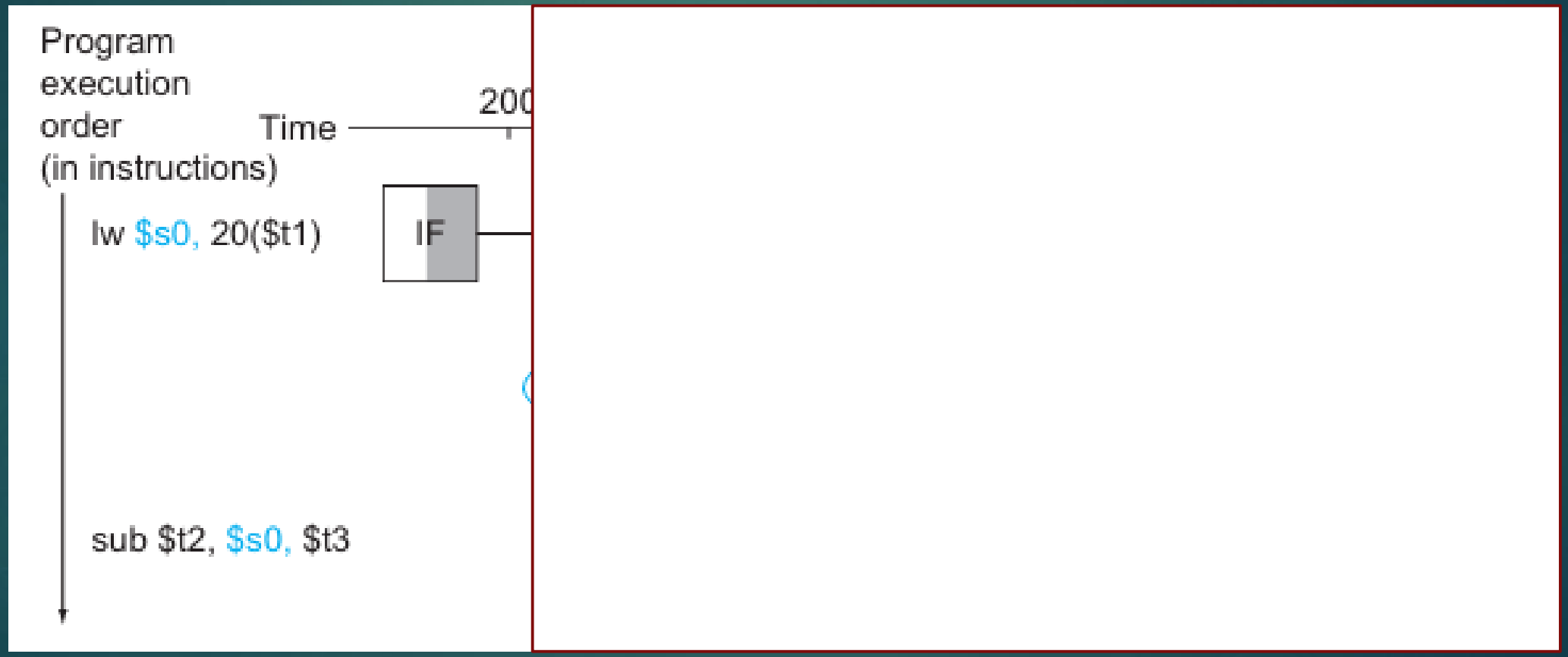


Data Hazards: Solution

Solution: forwarding

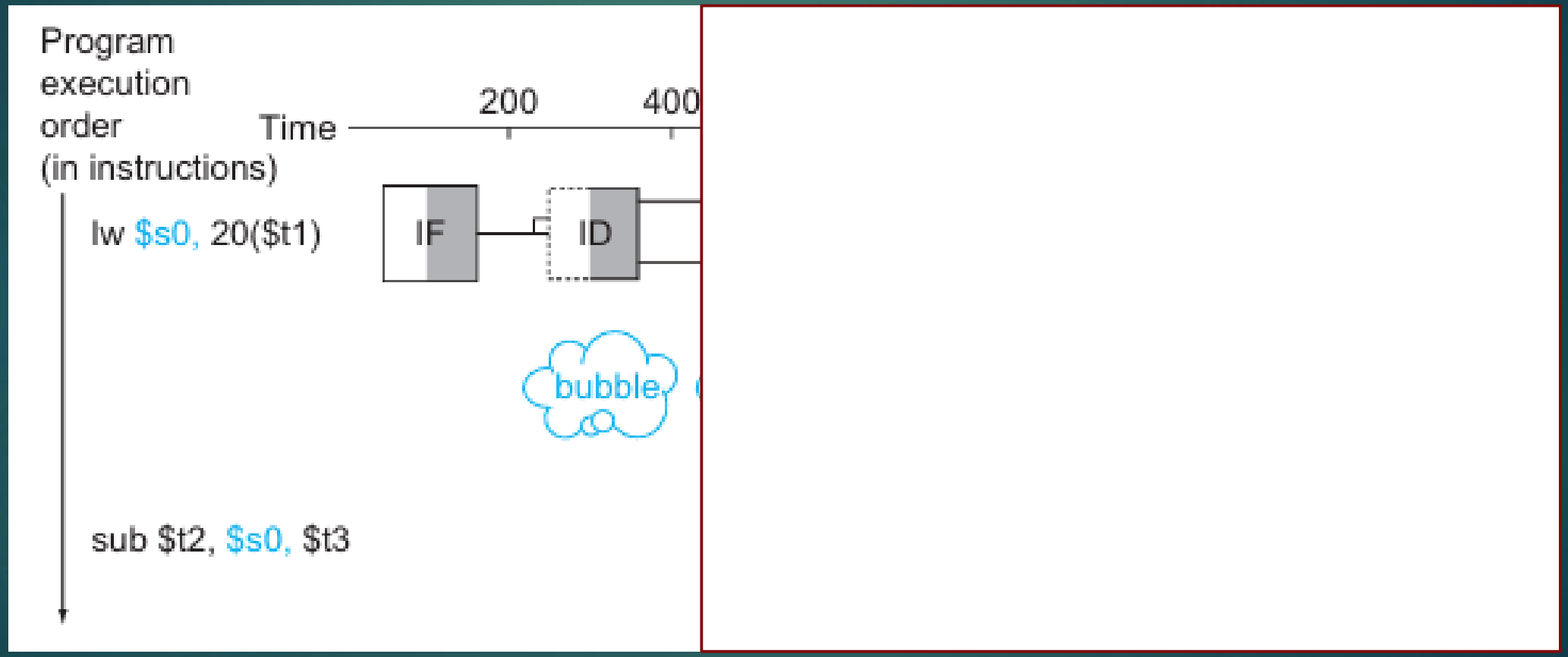
Data Hazards: Solution

Solution: forwarding



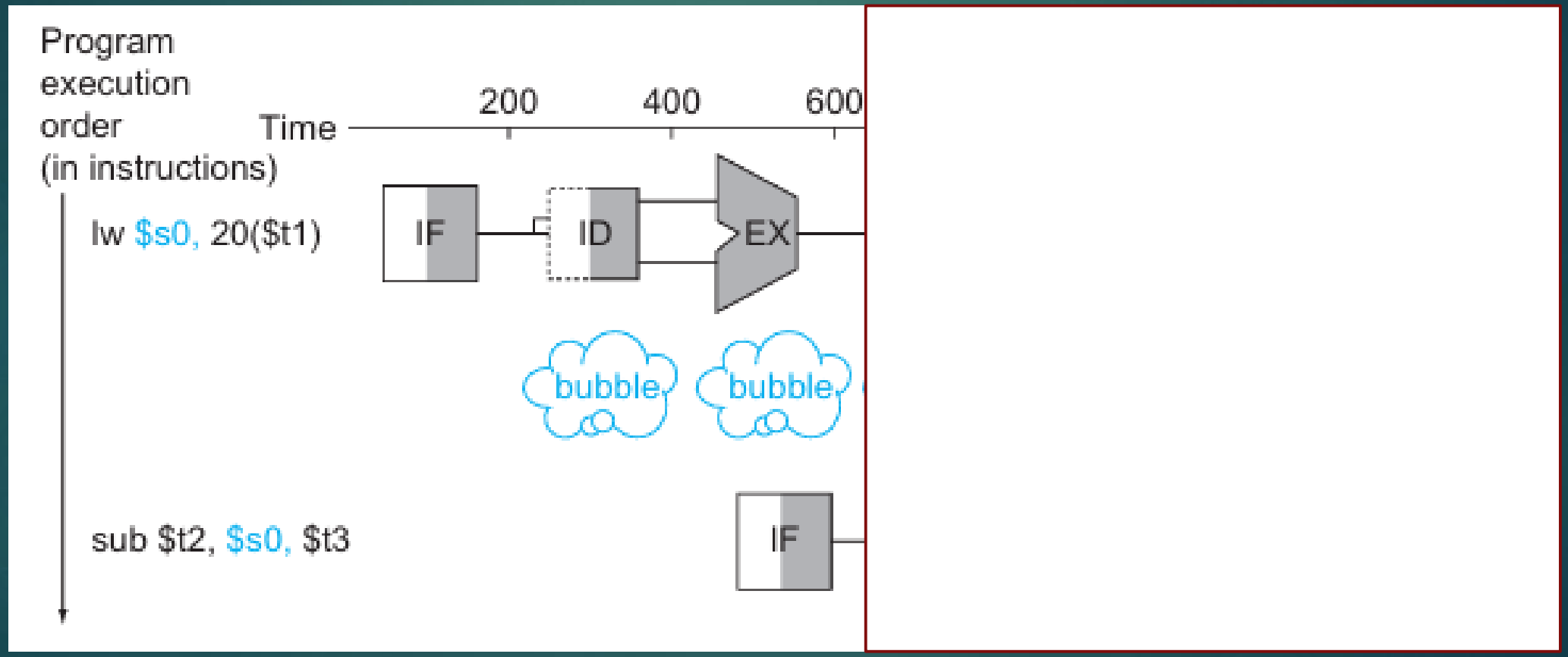
Data Hazards: Solution

Solution: forwarding



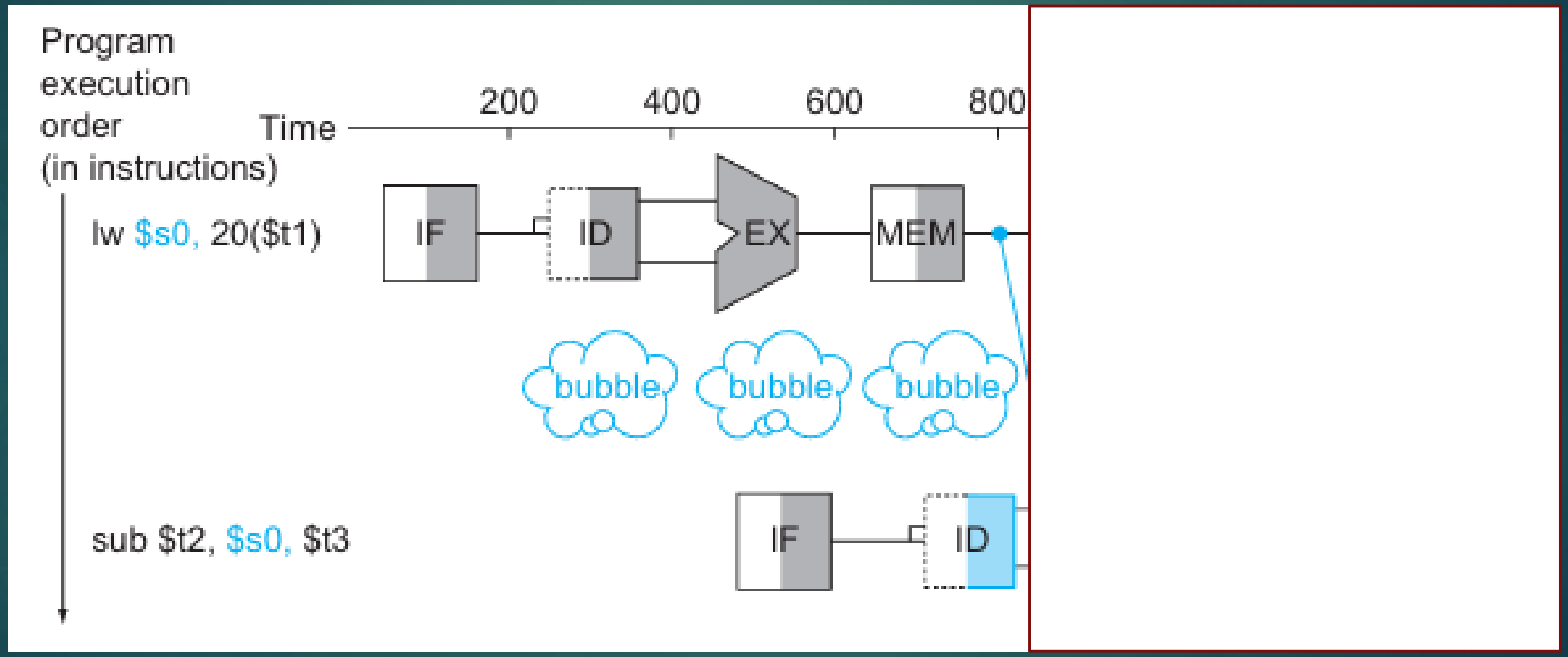
Data Hazards: Solution

Solution: forwarding



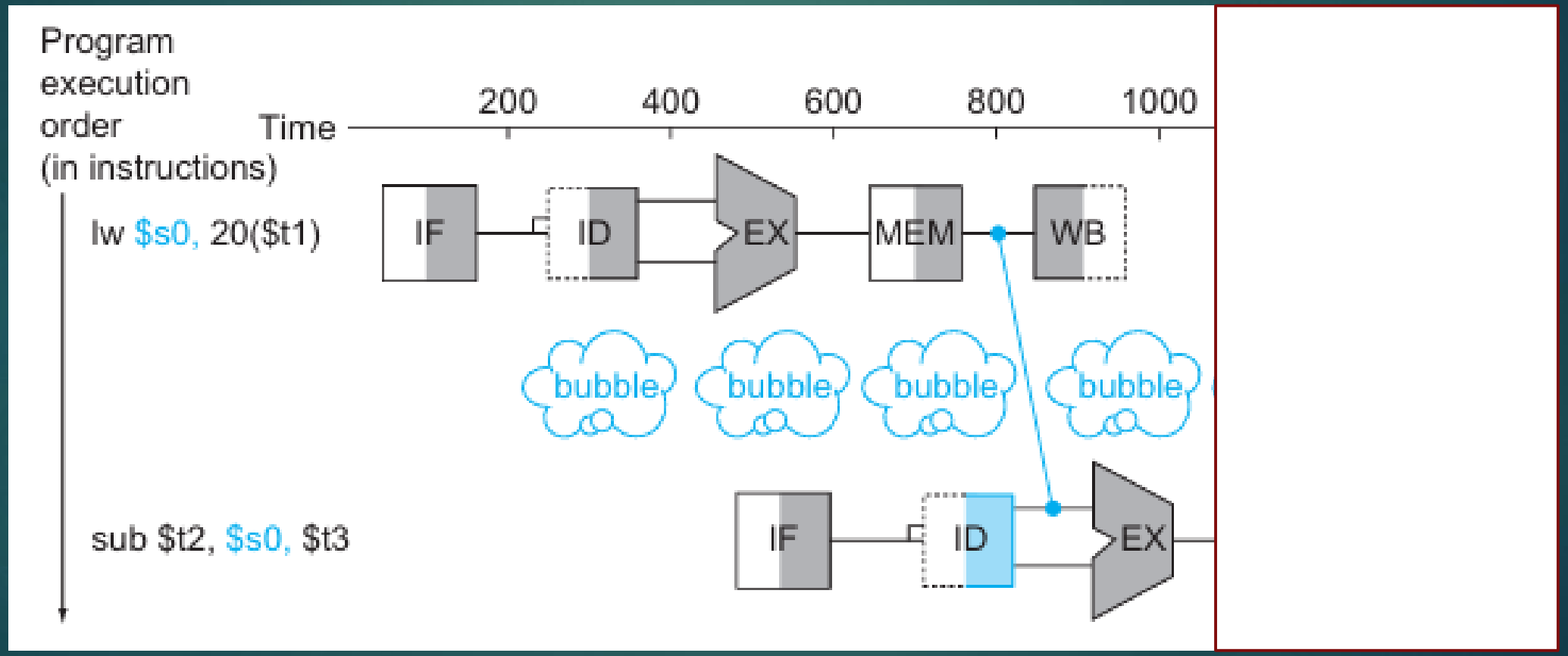
Data Hazards: Solution

Solution: forwarding



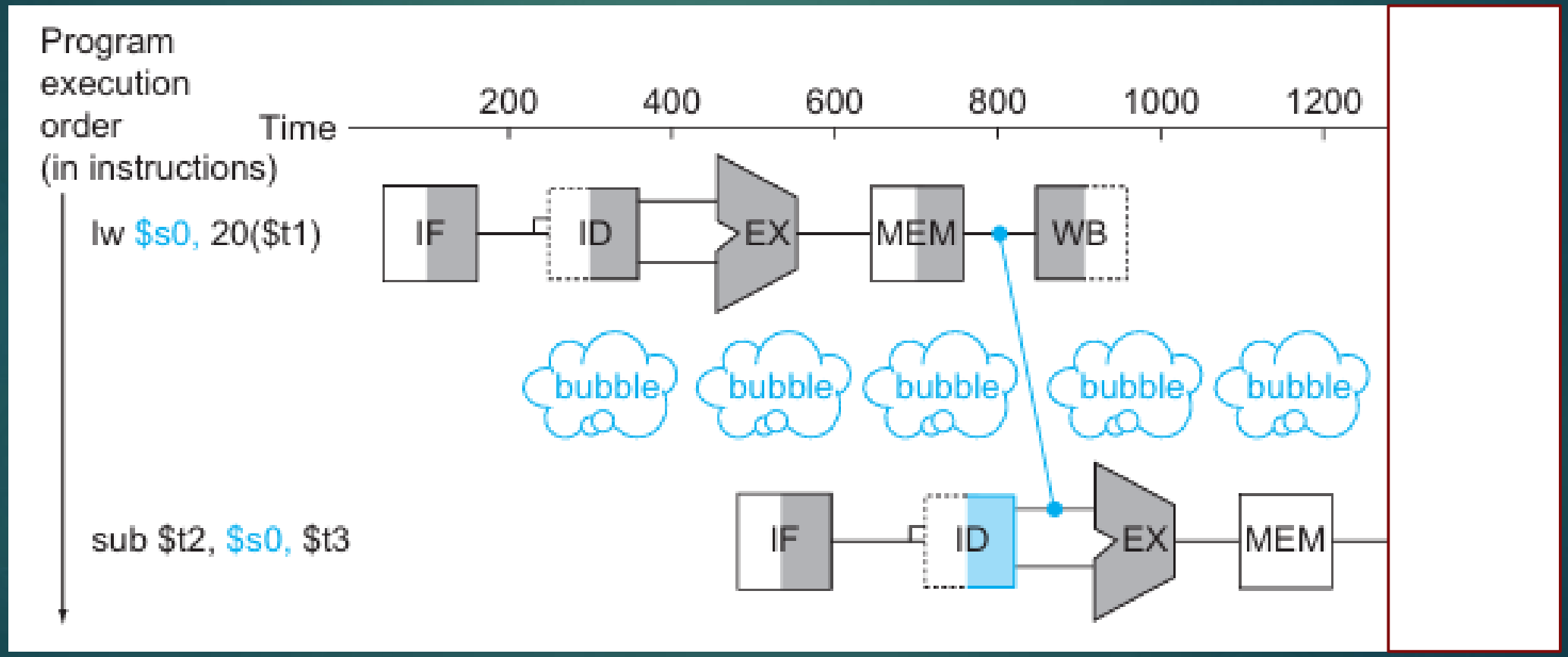
Data Hazards: Solution

Solution: forwarding



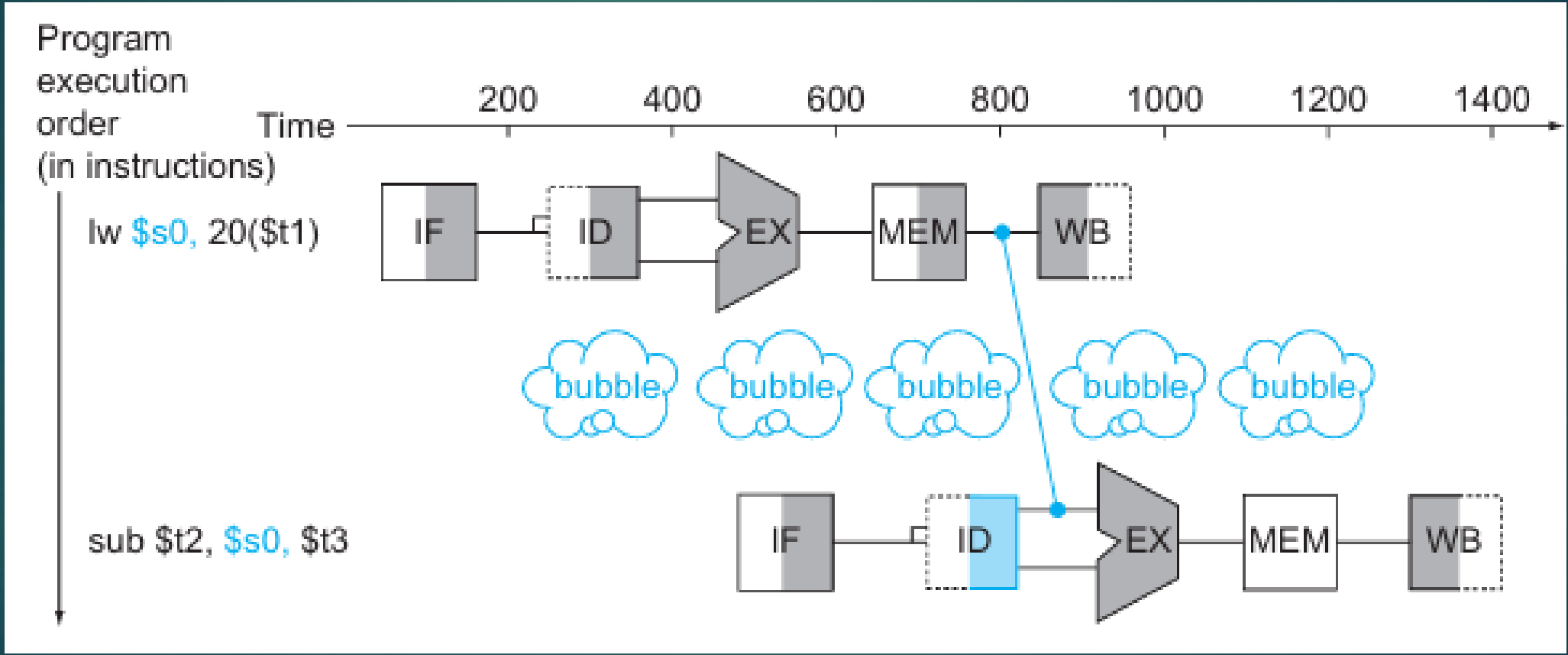
Data Hazards: Solution

Solution: forwarding



Data Hazards: Solution

Solution: forwarding



Data Hazards : Solution

- **Reordering Code to Avoid Stalls**
  - Find the hazards in the preceding code segment

Code segment in C:

```
a = b + e;
c = b + f;
```

Let all variables are in memory and addressable as off sets from $t0:

```
lw $t1, 0($t0)        #load b
lw $t2, 4($t0)        #load e
```

Data Hazards : Solution

- **Reordering Code to Avoid Stalls**
  - Find the hazards in the preceding code segment

Code segment in C:

```
a = b + e;
c = b + f;
```

Let all variables are in memory and addressable as off sets from $t0:

```
lw $t1, 0($t0)          #load b
lw $t2, 4($t0)          #load e
add $t3, $t1,$t2        # a = e+ b
sw $t3, 12($t0)         #store a
```

Data Hazards : Solution

- **Reordering Code to Avoid Stalls**
  - Find the hazards in the preceding code segment

Code segment in C:

```
a = b + e;
c = b + f;
```

Let all variables are in memory and addressable as off sets from $t0:

```
lw $t1, 0($t0)          #load b
lw $t2, 4($t0)          #load e
add $t3, $t1,$t2        # a = e+ b
sw $t3, 12($t0)         #store a
lw $t4, 8($t0)          #load f
```

Data Hazards : Solution

- **Reordering Code to Avoid Stalls**
  - Find the hazards in the preceding code segment

Code segment in C:

```
a = b + e;
c = b + f;
```

Let all variables are in memory and addressable as off sets from $t0:

```
lw $t1, 0($t0)          #load b
lw $t2, 4($t0)          #load e
add $t3, $t1,$t2        # a = e+ b
sw $t3, 12($t0)         #store a
lw $t4, 8($t0)          #load f
add $t5, $t1,$t4        #c = b + f
sw $t5, 16($t0)         #store c
```

Data Hazards : Solution

- **Reordering Code to Avoid Stalls**
  - Find the hazards in the preceding code segment
  - **Reorder** the instructions to avoid any pipeline stalls.

assuming all variables are in memory and are addressable as off sets from $t0:

```
lw $t1, 0($t0)
lw $t2, 4($t0)
add $t3, $t1,$t2
sw $t3, 12($t0)
lw $t4, 8($t0)
add $t5, $t1,$t4
sw $t5, 16($t0)
```

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	
lw $t2, 4($t0)		
add $t3, $t1,$t2		
sw $t3, 12($t0)		
lw $t4, 8($t0)		
add $t5, $t1,$t4		
sw $t5, 16($t0)		

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)
lw $t2, 4($t0)		IF
add $t3, $t1,$t2		
sw $t3, 12($t0)		
lw $t4, 8($t0)		
add $t5, $t1,$t4		
sw $t5, 16($t0)		

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)
lw $t2, 4($t0)		IF	ID ($t0)
add $t3, $t1,$t2			IF
sw $t3, 12($t0)			
lw $t4, 8($t0)			
add $t5, $t1,$t4			
sw $t5, 16($t0)			

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)
add $t3, $t1,$t2			IF	ID ($t1, **$t2**)
sw $t3, 12($t0)				IF
lw $t4, 8($t0)				
add $t5, $t1,$t4				
sw $t5, 16($t0)				

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM	WB ($t1)
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)	MEM
add $t3, $t1,$t2			IF	ID ($t1, $t2)	----
sw $t3, 12($t0)				IF	---
lw $t4, 8($t0)					
add $t5, $t1,$t4					
sw $t5, 16($t0)					

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM	WB ($t1)	
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)	MEM	WB ($t2)
add $t3, $t1,$t2			IF	ID ($t1, $t2)	----	EX ($t1, $t2)
sw $t3, 12($t0)				IF	---	ID ($t0)
lw $t4, 8($t0)						IF
add $t5, $t1,$t4						
sw $t5, 16($t0)						

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM	WB ($t1)		
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)	MEM	WB ($t2)	
add $t3, $t1,$t2			IF	ID ($t1, $t2)	----	EX ($t1, $t2)	MEM
sw $t3, 12($t0)				IF	---	ID ($t0)	EX ($t0)
lw $t4, 8($t0)						IF	ID ($t0)
add $t5, $t1,$t4							IF
sw $t5, 16($t0)							

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM	WB ($t1)			
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)	MEM	WB ($t2)		
add $t3, $t1,$t2			IF	ID ($t1, $t2)	----	EX ($t1, $t2)	MEM	WB ($t3)
sw $t3, 12($t0)				IF	---	ID ($t0)	EX ($t0)	MEM ($t3)
lw $t4, 8($t0)						IF	ID ($t0)	EX ($t0)
add $t5, $t1,$t4							IF	ID ($t1, $t4)
sw $t5, 16($t0)								IF

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM	WB ($t1)				
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)	MEM	WB ($t2)			
add $t3, $t1,$t2			IF	ID ($t1, $t2)	----	EX ($t1, $t2)	MEM	WB ($t3)	
sw $t3, 12($t0)				IF	---	ID ($t0)	EX ($t0)	MEM ($t3)	WB
lw $t4, 8($t0)						IF	ID ($t0)	EX ($t0)	MEM
add $t5, $t1,$t4							IF	ID ($t1, $t4)	---
sw $t5, 16($t0)								IF	---

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM	WB ($t1)					
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)	MEM	WB ($t2)				
add $t3, $t1,$t2			IF	ID ($t1, $t2)	----	EX ($t1, $t2)	MEM	WB ($t3)		
sw $t3, 12($t0)				IF	---	ID ($t0)	EX ($t0)	MEM ($t3)	WB	
lw $t4, 8($t0)						IF	ID ($t0)	EX ($t0)	MEM	WB ($t4)
add $t5, $t1,$t4							IF	ID ($t1, $t4)	---	EX ($t1, $t4)
sw $t5, 16($t0)								IF	---	ID ($t0)

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM	WB ($t1)						
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)	MEM	WB ($t2)					
add $t3, $t1,$t2			IF	ID ($t1, $t2)	----	EX ($t1, $t2)	MEM	WB ($t3)			
sw $t3, 12($t0)				IF	---	ID ($t0)	EX ($t0)	MEM ($t3)	WB		
lw $t4, 8($t0)						IF	ID ($t0)	EX ($t0)	MEM	WB ($t4)	
add $t5, $t1,$t4							IF	ID ($t1, $t4)	---	EX ($t1, $t4)	MEM
sw $t5, 16($t0)								IF	---	ID ($t0)	EX ($t0)

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM	WB ($t1)							
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)	MEM	WB ($t2)						
add $t3, $t1,$t2			IF	ID ($t1, $t2)	----	EX ($t1, $t2)	MEM	WB ($t3)				
sw $t3, 12($t0)				IF	---	ID ($t0)	EX ($t0)	MEM ($t3)	WB			
lw $t4, 8($t0)						IF	ID ($t0)	EX ($t0)	MEM	WB ($t4)		
add $t5, $t1,$t4							IF	ID ($t1, $t4)	---	EX ($t1, $t4)	MEM	WB ($t5)
sw $t5, 16($t0)								IF	---	ID ($t0)	EX ($t0)	MEM ($t5)

Find the hazards in the preceding code segment

assuming all variables are in memory and are addressable as off sets from $t0:

lw $t1, 0($t0)	IF	ID ($t0)	EX ($t0)	MEM	WB ($t1)								
lw $t2, 4($t0)		IF	ID ($t0)	EX ($t0)	MEM	WB ($t2)							
add $t3, $t1,$t2			IF	ID ($t1, $t2)	----	EX ($t1, $t2)	MEM	WB ($t3)					
sw $t3, 12($t0)				IF	---	ID ($t0)	EX ($t0)	MEM ($t3)	WB				
lw $t4, 8($t0)						IF	ID ($t0)	EX ($t0)	MEM	WB ($t4)			
add $t5, $t1,$t4							IF	ID ($t1, $t4)	---	EX ($t1, $t4)	MEM	WB ($t5)	
sw $t5, 16($t0)								IF	---	ID ($t0)	EX ($t0)	MEM ($t5)	WB

Data Hazards

- **Reordering Code to Avoid Pipeline Stalls**

MIPS code :

```
lw $t1, 0($t0)
lw $t2, 4($t0)
add $t3, $t1,$t2
sw $t3, 12($t0)
lw $t4, 8($t0)
add $t5, $t1,$t4
sw $t5, 16($t0)
```

- **Reordering Code to Avoid Pipeline Stalls**
  - Both add instructions have a hazard because of their respective dependence on the immediately preceding lw instruction.
  - **Move up 3rd lw** instruction to make it 3rd in sequence to eliminates both hazards:
    - lw $t1, 0($t0)
    - lw $t2, 4($t0)
    - **lw $t4, 8($t0)**
    - add $t3, $t1,$t2
    - sw $t3, 12($t0)
    - add $t5, $t1,$t4
    - sw $t5, 16($t0)

- Advantage: On a pipelined processor with forwarding, the reordered sequence will complete in two fewer cycles than the original version

Four types of Data Hazards

Read After Write (RAW)

R-Type

```
ADD R1, R2, R3
SUB R4, R1, R6
```

Mem-Access

```
store R1, 0($t0)
load R4, 0($t0)
```

Write After Read (WAR)

```
add r4, r1,
add r1, r3, r5
```

```
add R1, R2, R3
sub R2, R4, R1
or R1, R6, R3
```

Write After Write (WAW)

```
add R1,R2,R3
sub R2,R4,R1
or R1,R6,R3
```

```
add r1, r2, r3
add r4, r1, r5
add r1, r3, r5
```

Read After Read (RAR)

```
add R1, R2, R3
sub R2, R4, R1
or R1, R6, R3
```

For MIPS integer pipeline, all data hazards can be checked during ID phase

Thank You