

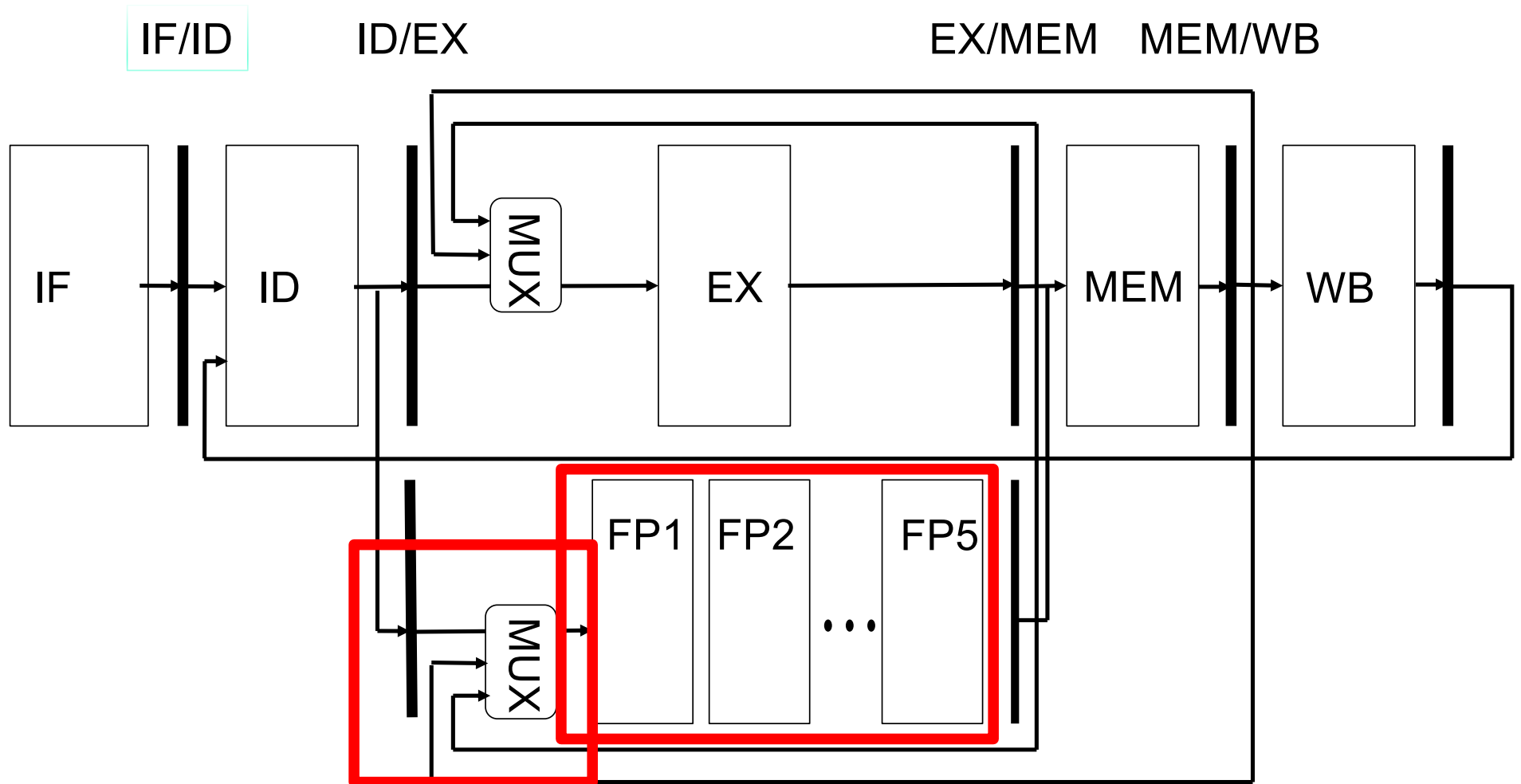
Lecture 2

Instruction scheduling techniques

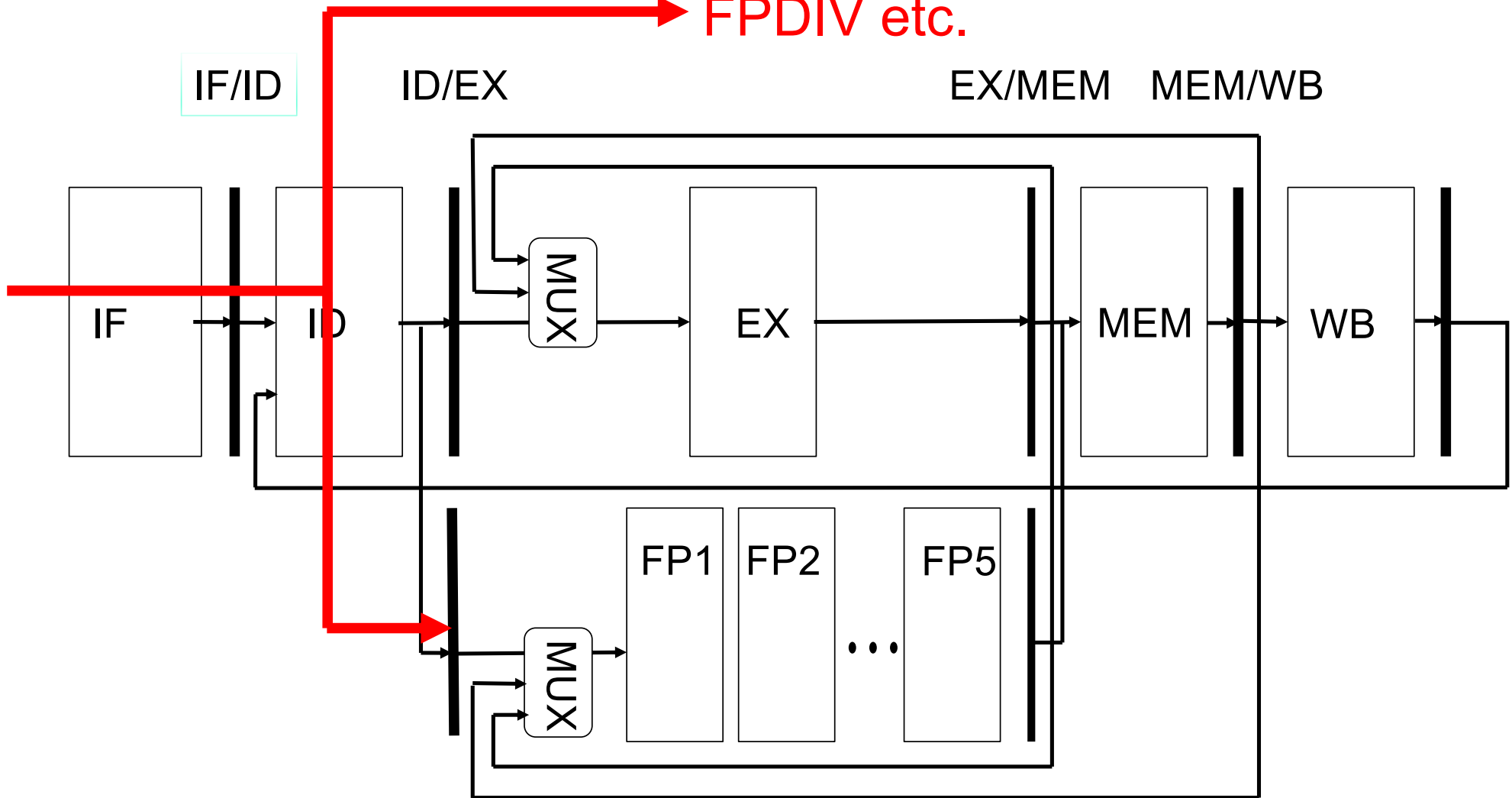
- **Statically scheduled pipelines (Ch. 3.3.2 - 3.3.6)**
 - ✓ Out-of-order instruction completion (3.3.2)
 - ✓ Superpipelined and superscalar CPUs (3.3.3)
 - ✓ Static instruction scheduling (3.3.4-3.3.5)

Out-of-order Completion

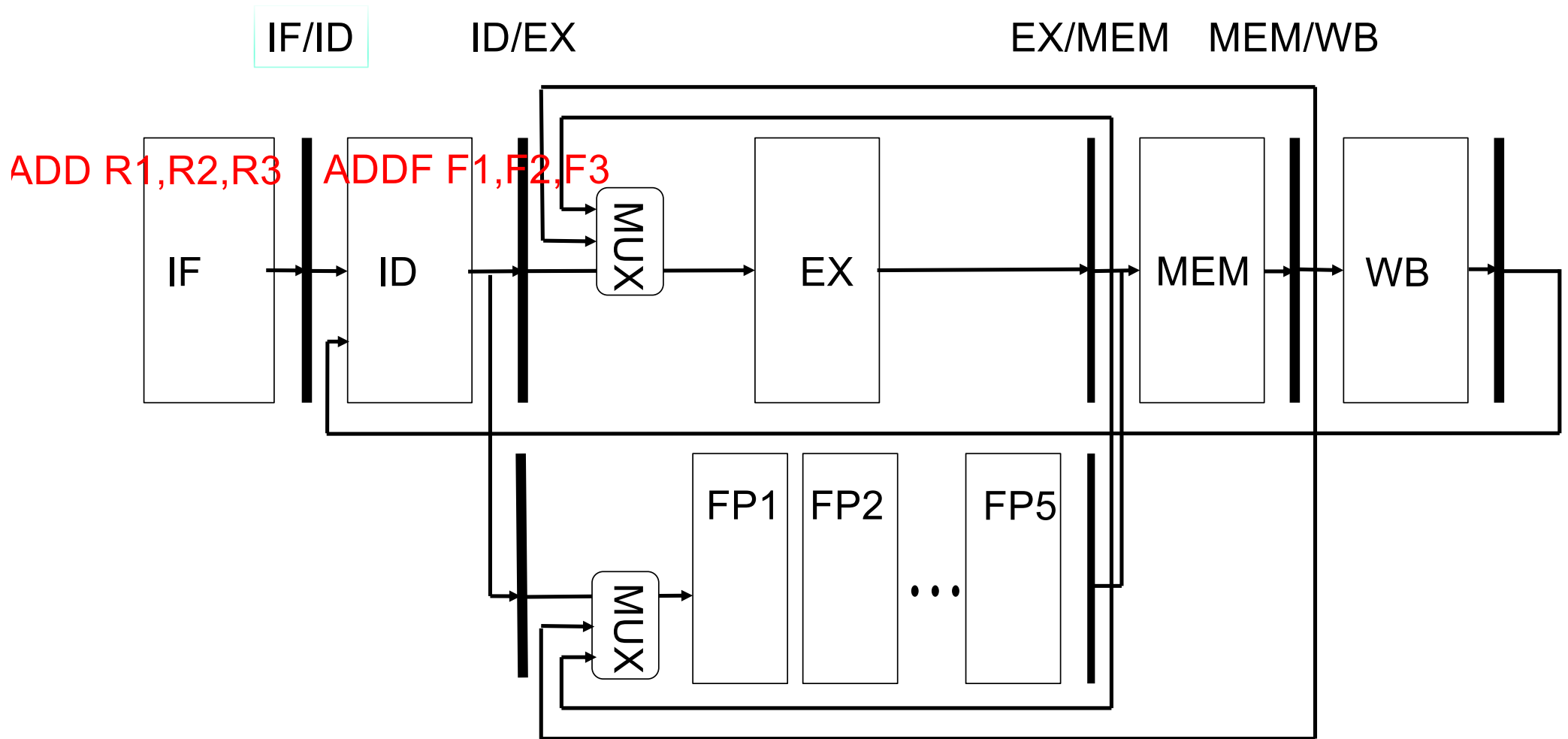
Multi-Cycle Operations



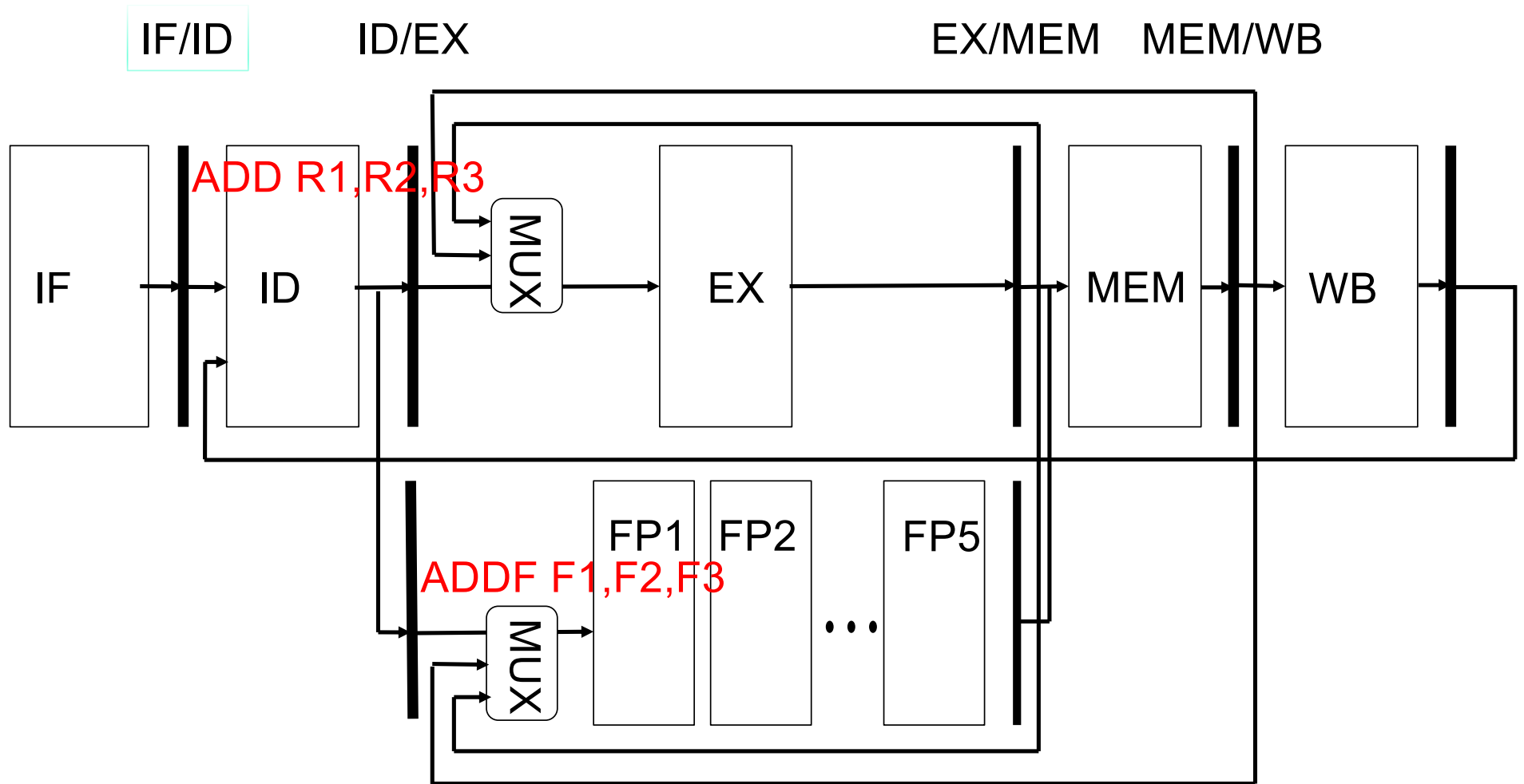
All except FP OP: FPADD, FPMULT,
FPDIV etc.



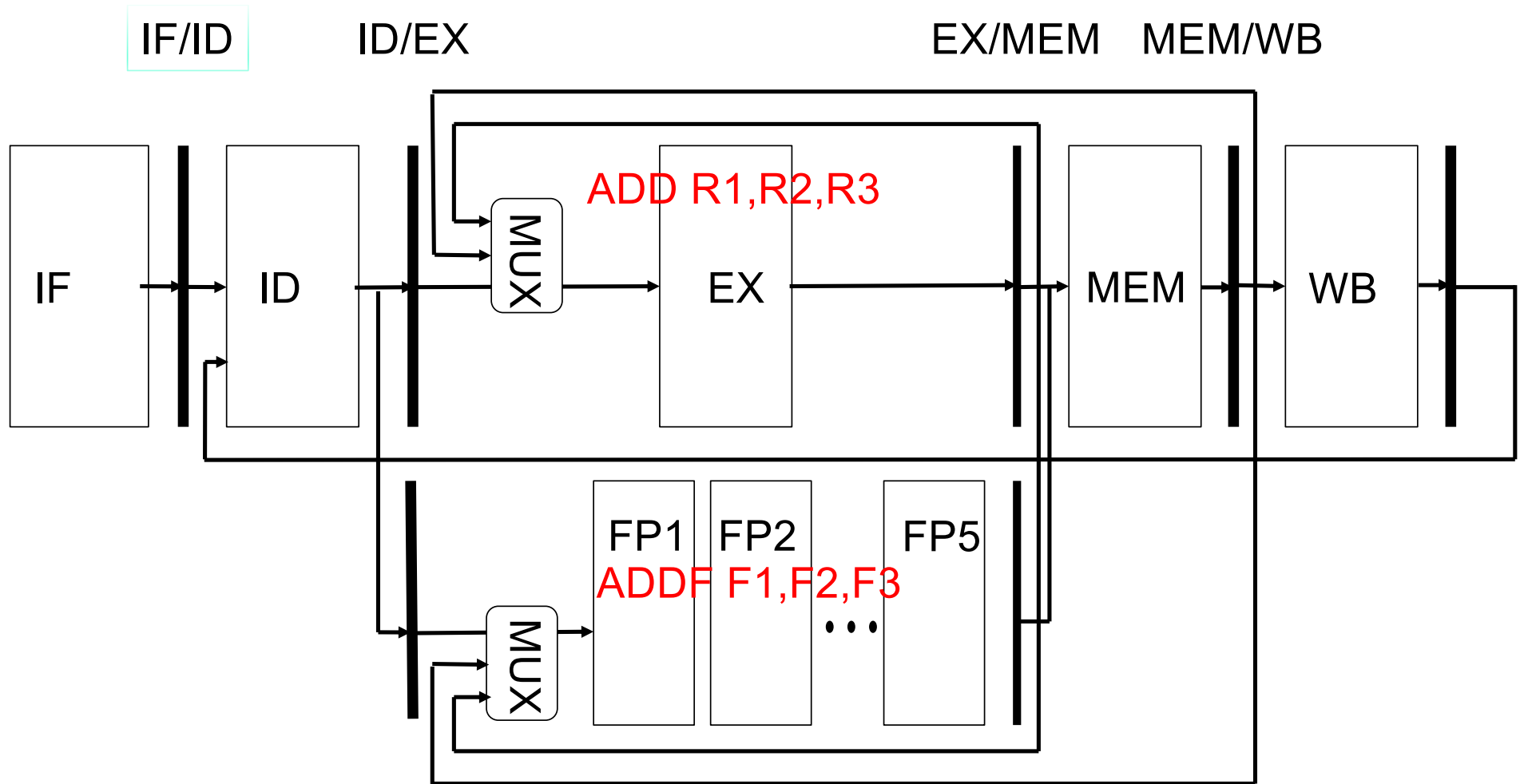
Clock 1



Clock 2

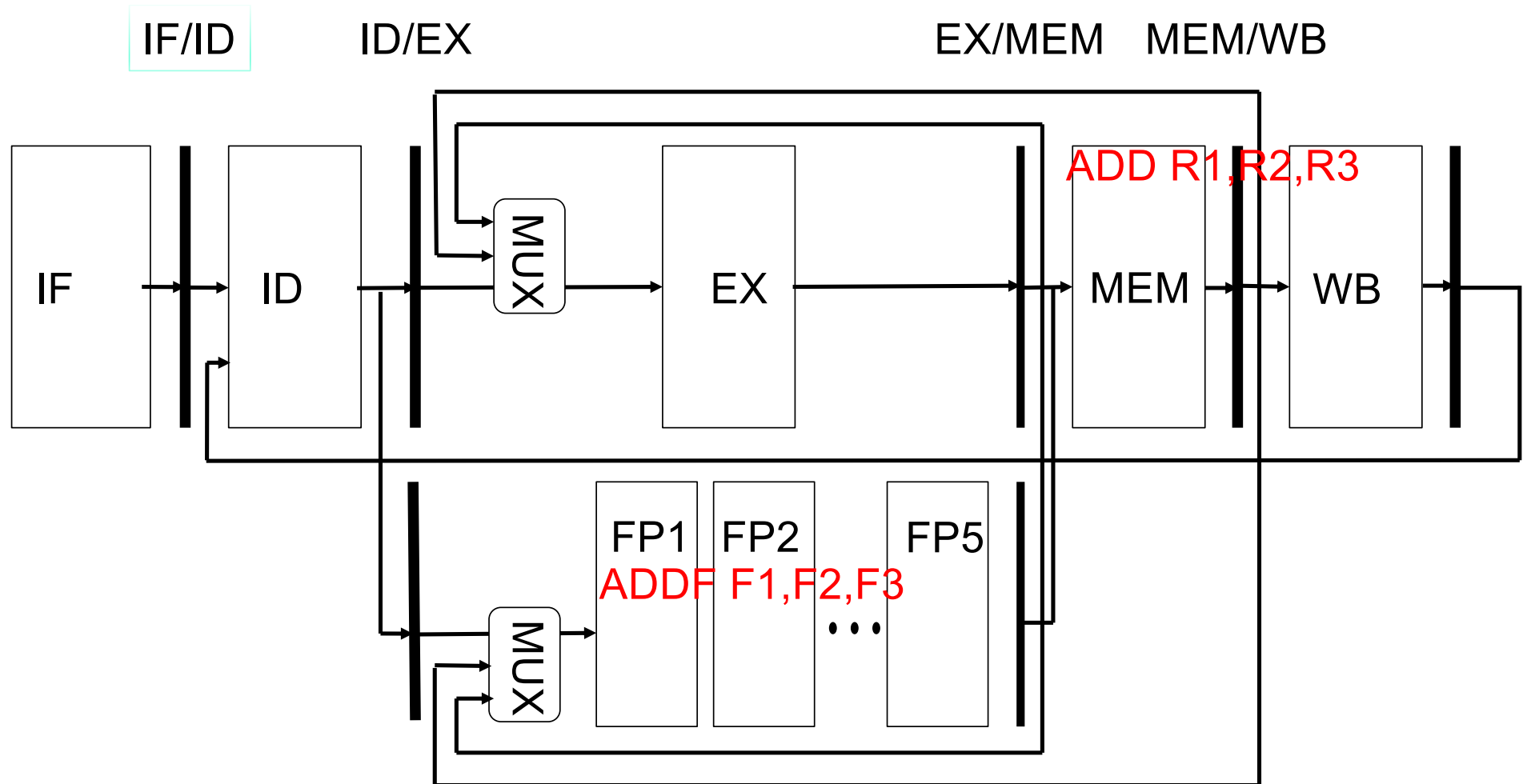


Clock 3



Clock 4

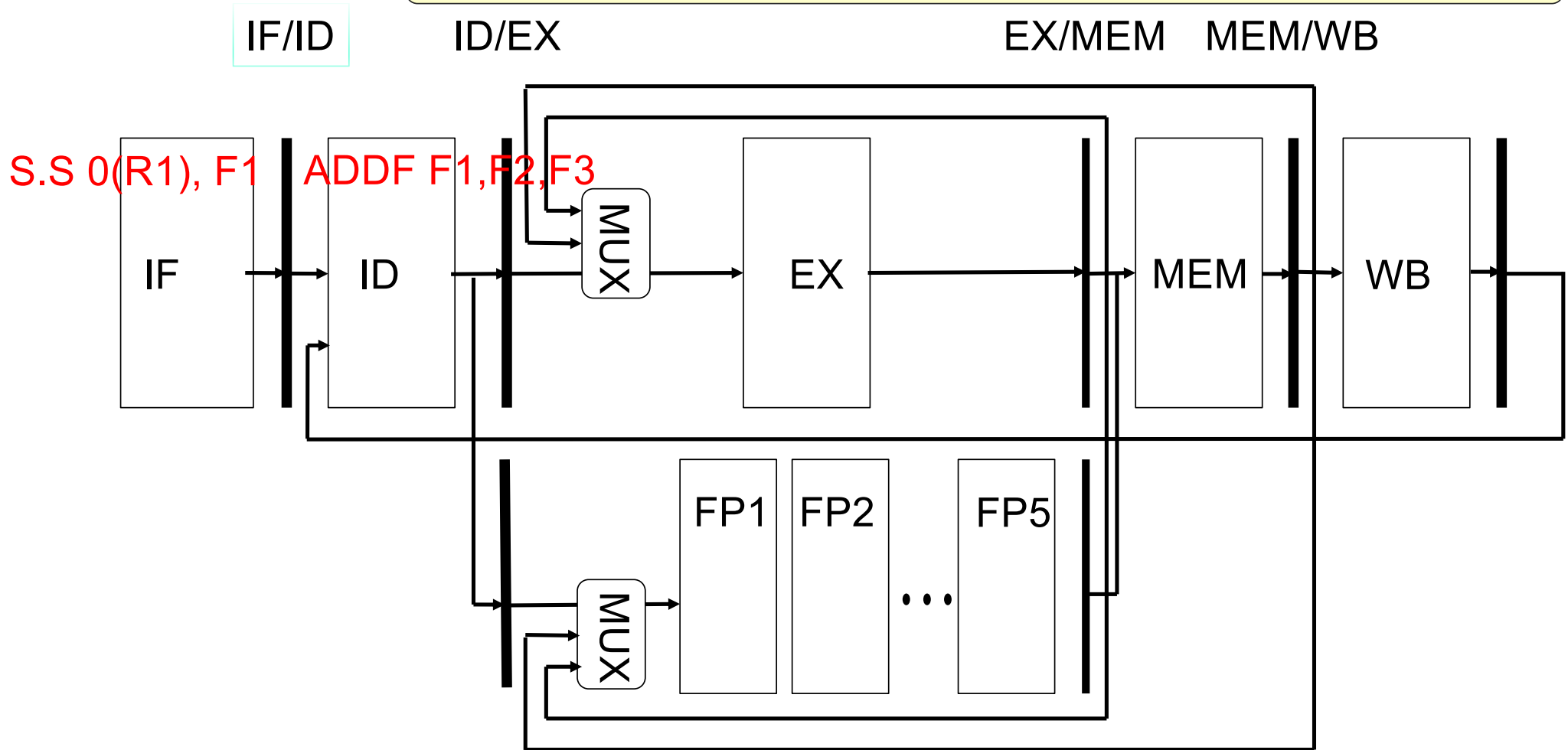
In-order execution, out-of-order completion!



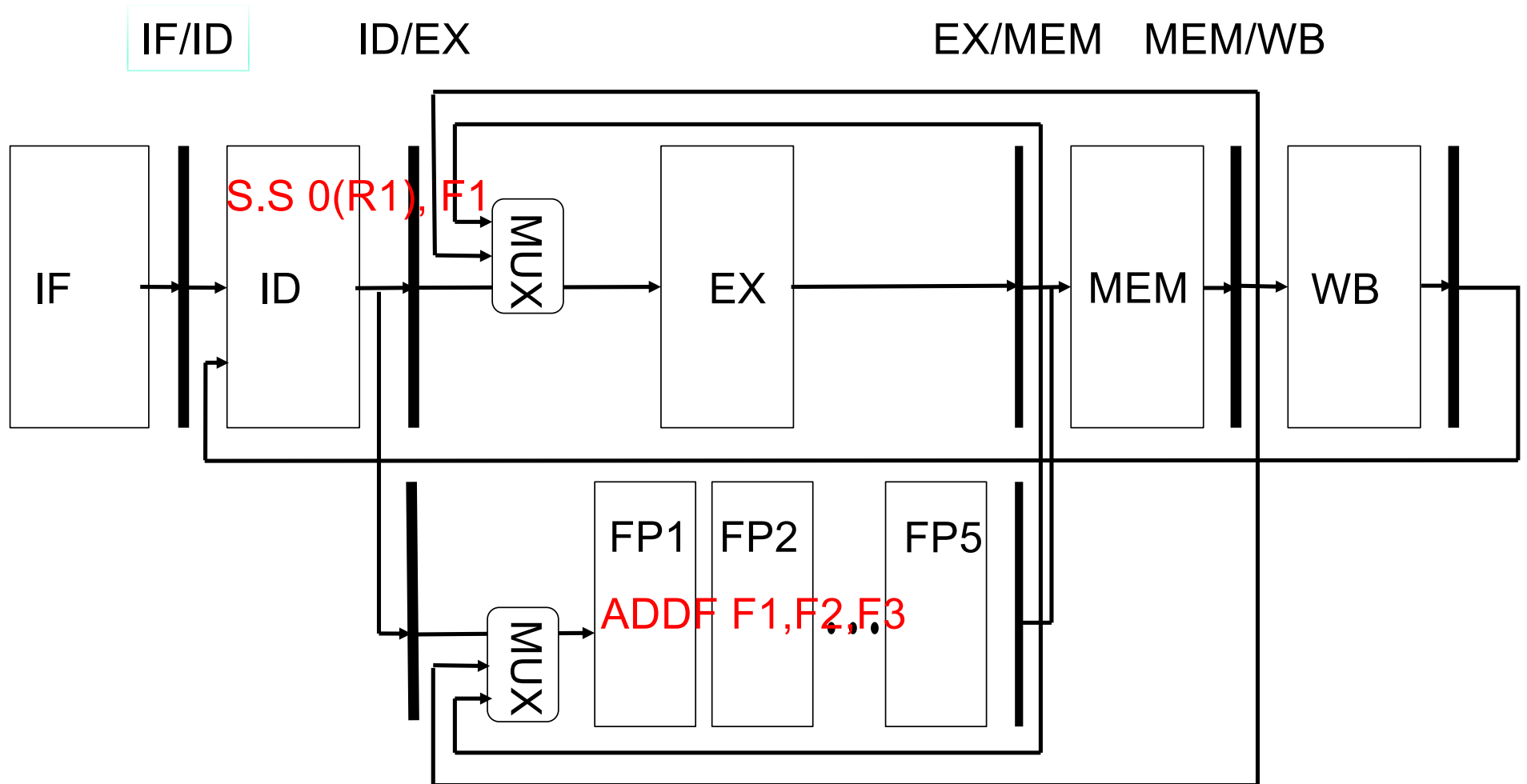
Clock 1

Question:

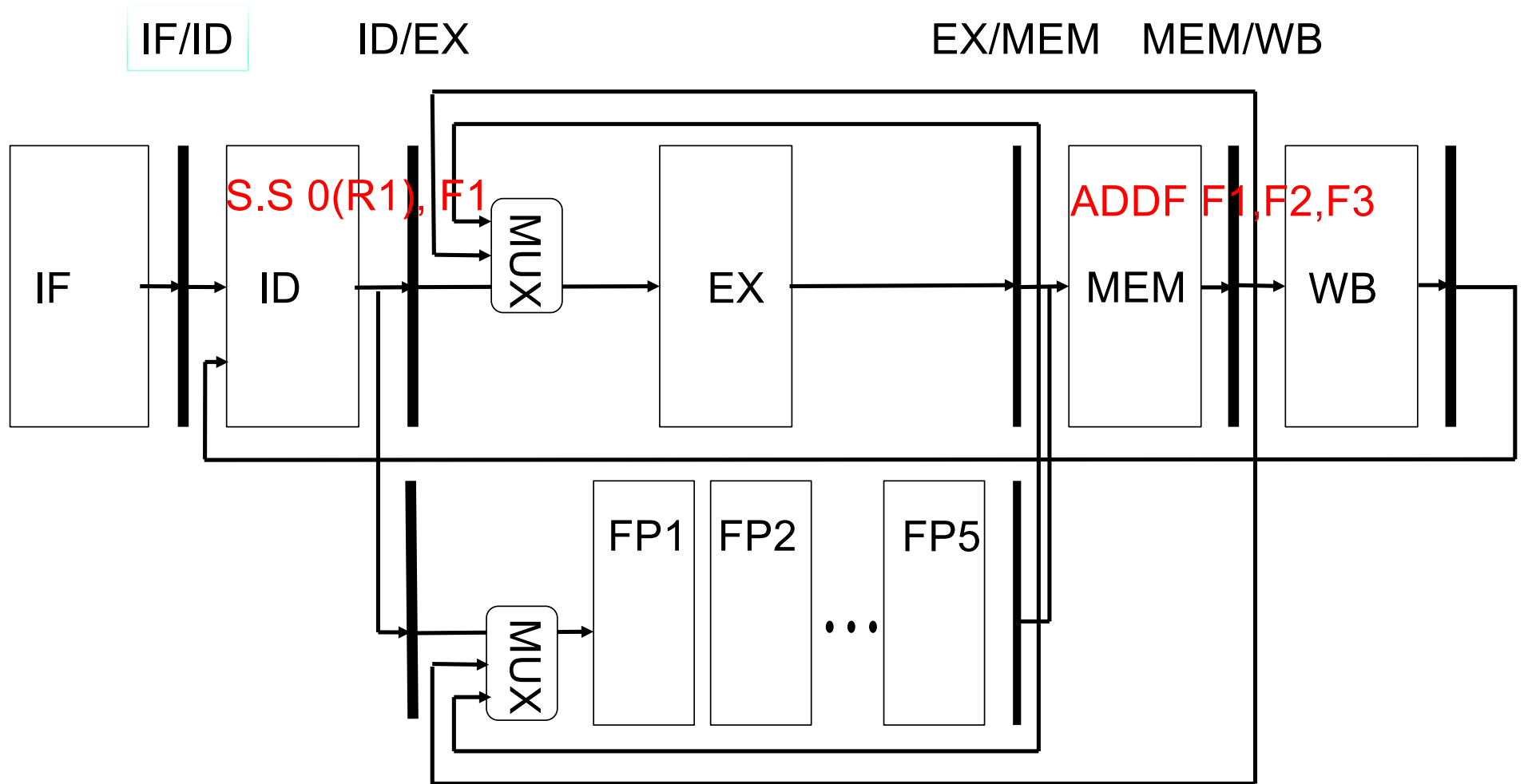
How many clocks does it take to complete the execution of the two instructions??



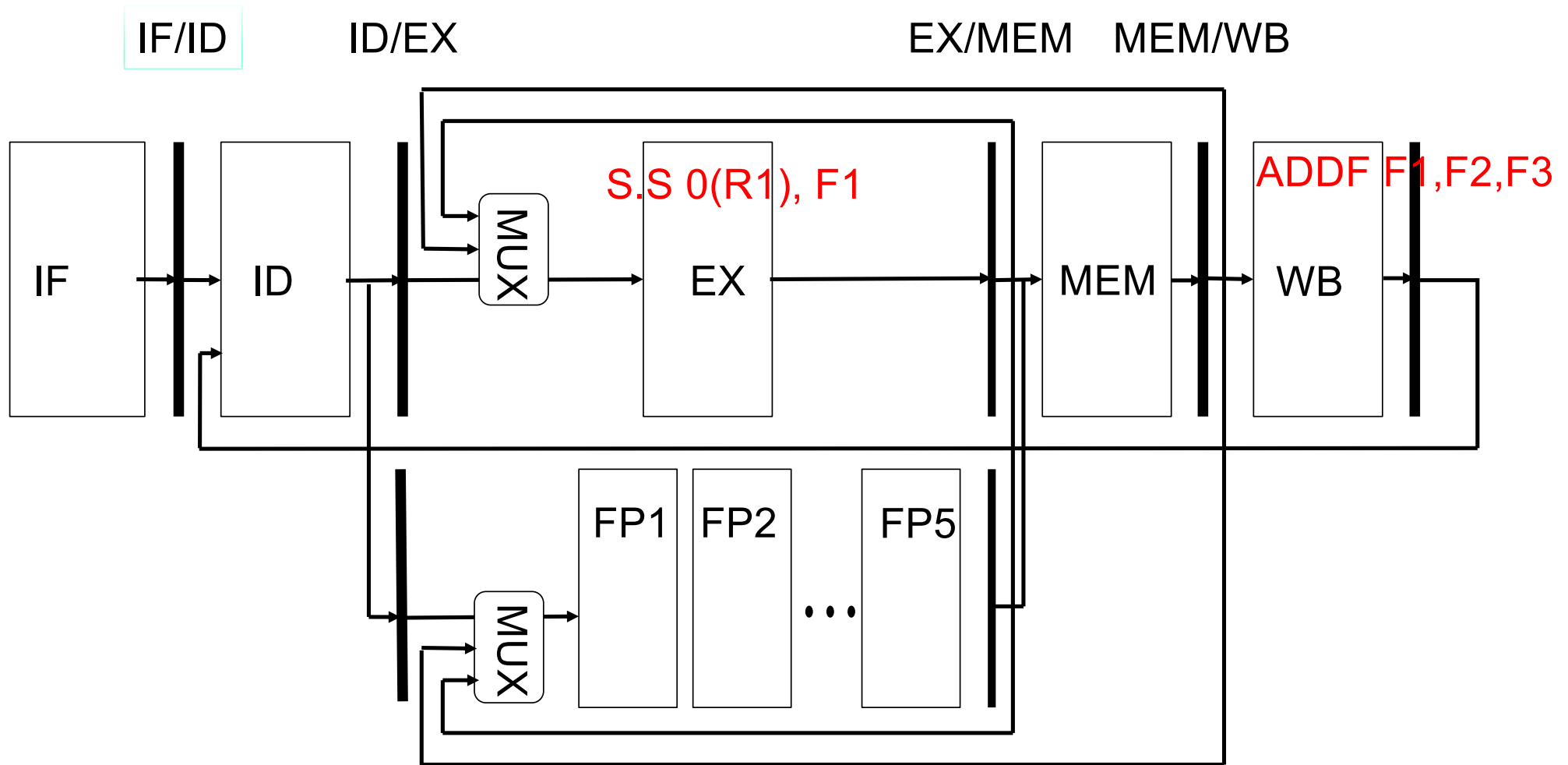
Clock 2



Clock 6



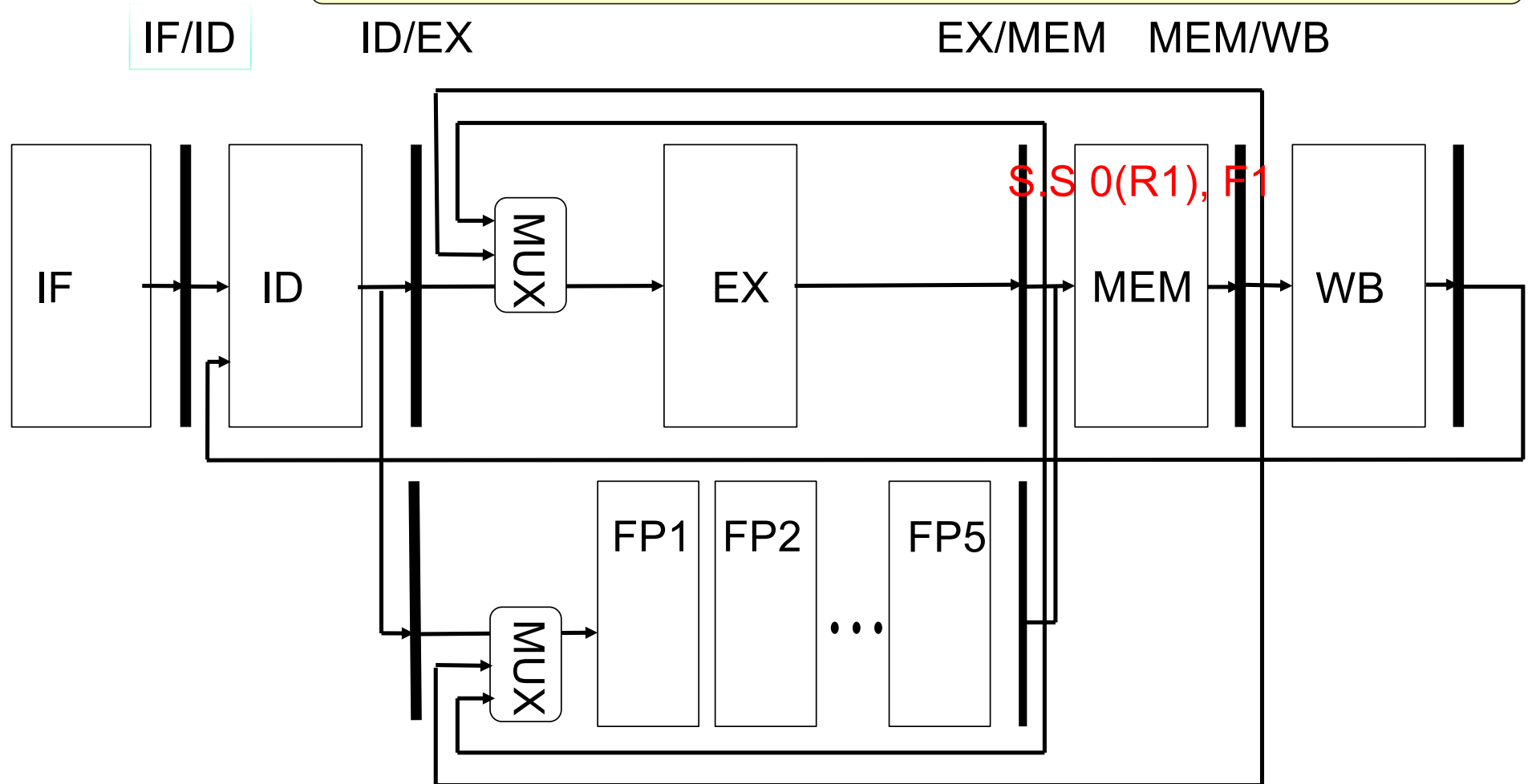
Clock 7



Clock 8

Answer:

Eight cycles.



Pipeline Hazards

Question:

What is a structural hazard?

Question:


How can we use the pipeline diagram to identify them?

Structural hazards

Data hazards

Control hazards

	CLOCK ==>	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
I1	ADD.S F1,F2,F1	IF	ID	FP1	FP2	FP3	FP4	FP5	ME	W/R	
I2	ADD.S F4,F2,F3		IF	ID	FP1	FP2	FP3	FP4	FP5	ME	WB
I3	L.S F10			IF	ID	EX	ME	WB			
I4	L.S F12				IF	ID	EX	ME	WB		
I5	L.S F14					IF	ID	EX	ME	W/R	



Structural hazards

Data hazards

Control hazards


Question:

Are there any other structural hazards?

Answer:

Yes, I1 and I5 also compete for the memory stage in the same cycle – C8 but only I5 needs that resource.

	CLOCK ==>	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
I1	ADD.S F1,F2,F1	IF	ID	FP1	FP2	FP3	FP4	FP5	ME	WB	
I2	ADD.S F4,F2,F3		IF	ID	FP1	FP2	FP3	FP4	FP5	ME	WB
I3	L.S F10			IF	ID	EX	ME	WB			
I4	L.S F12				IF	ID	EX	ME	WB		
I5	L.S F14					IF	ID	EX	ME	WB	



Structural hazards

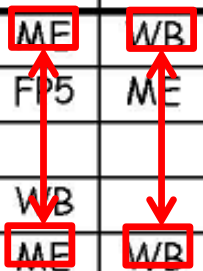
Data hazards

Control hazards

Discussion:

Discuss for a few minutes how we can avoid them

	CLOCK ==>	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
I1	ADD.S F1,F2,F1	IF	ID	FP1	FP2	FP3	FP4	FP5	ME	W/R	
I2	ADD.S F4,F2,F3		IF	ID	FP1	FP2	FP3	FP4	FP5	ME	WB
I3	L.S F10			IF	ID	EX	ME	WB			
I4	L.S F12				IF	ID	EX	ME	WB		
I5	L.S F14					IF	ID	EX	ME	W/R	



Structural hazards

Data hazards

Control hazards

Read-After-Write (RAW)

Write-After-Read (WAR)

Write-After-Write (WAW)

Question:

What RAW hazards show up in the program code?

	CLOCK ==>	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
I1	L.D F4, 0(R2)	IF	ID	EX	ME	WB						
I2	MULT.D F0,F4,F6		IF	ID	ID	FP1	FP2	FP3	FP4	FP5	ME	WB
I3	S.D F0, 0(R2)			IF	IF	ID	ID	ID	ID	ID	EX	ME

Discussion:

How can we reduce the effect of RAW hazards?

Question:

What type of data hazard shows up here?

Read-After-Write (RAW)

Write-After-Read (WAR)

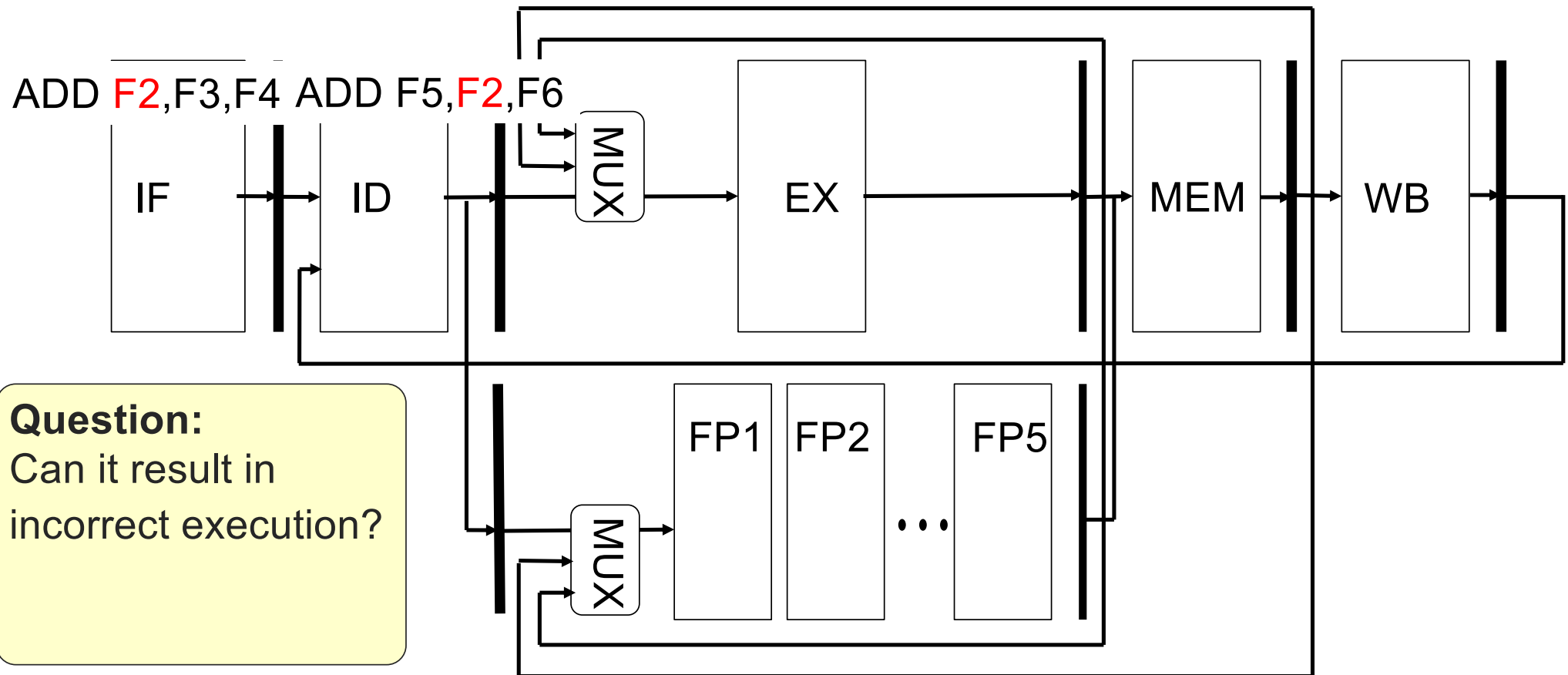
Write-After-Write (WAW)

IF/ID

ID/EX

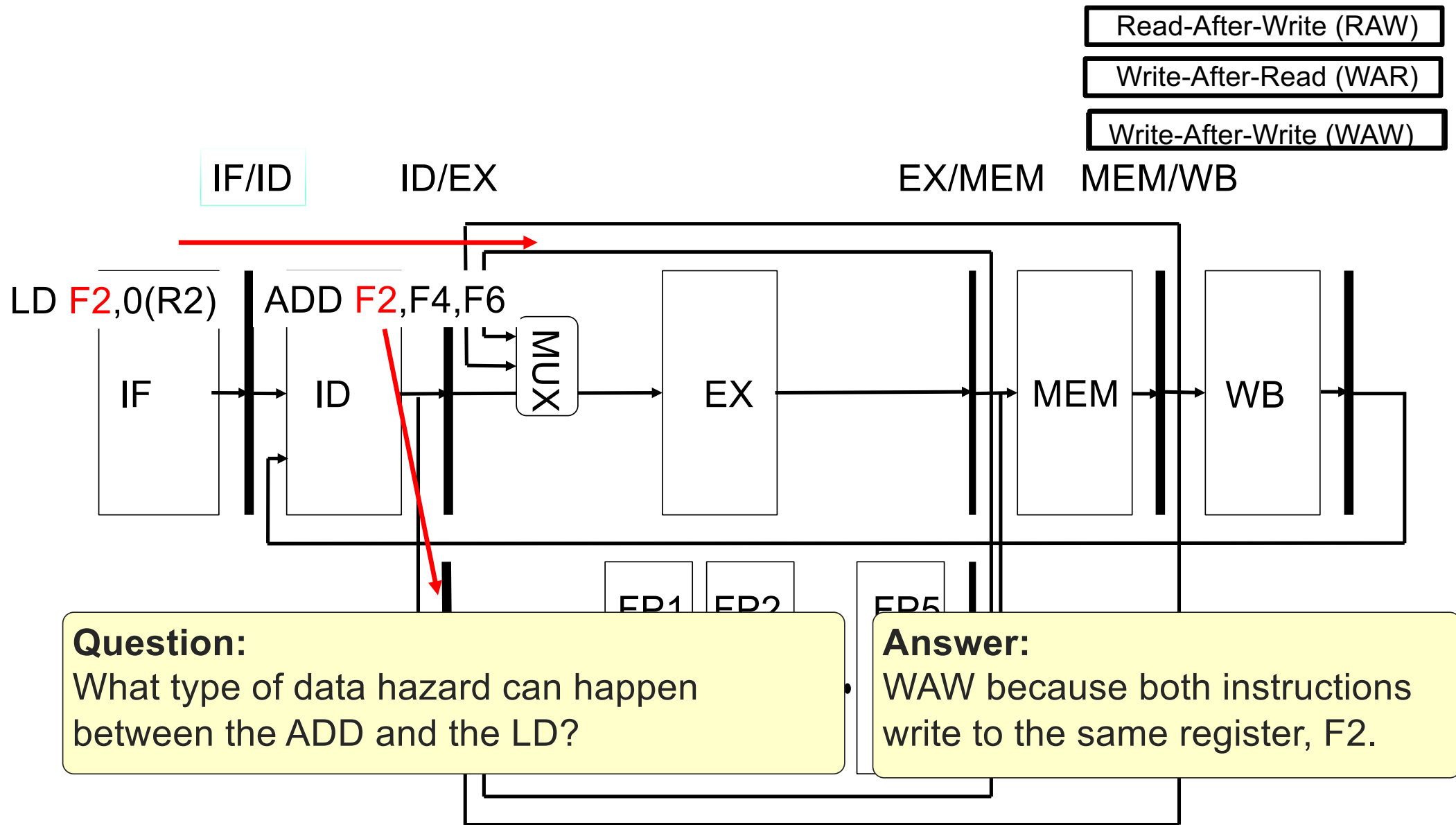
EX/MEM

MEM/WB



Question:

Can it result in incorrect execution?



Read-After-Write (RAW)

Write-After-Read (WAR)

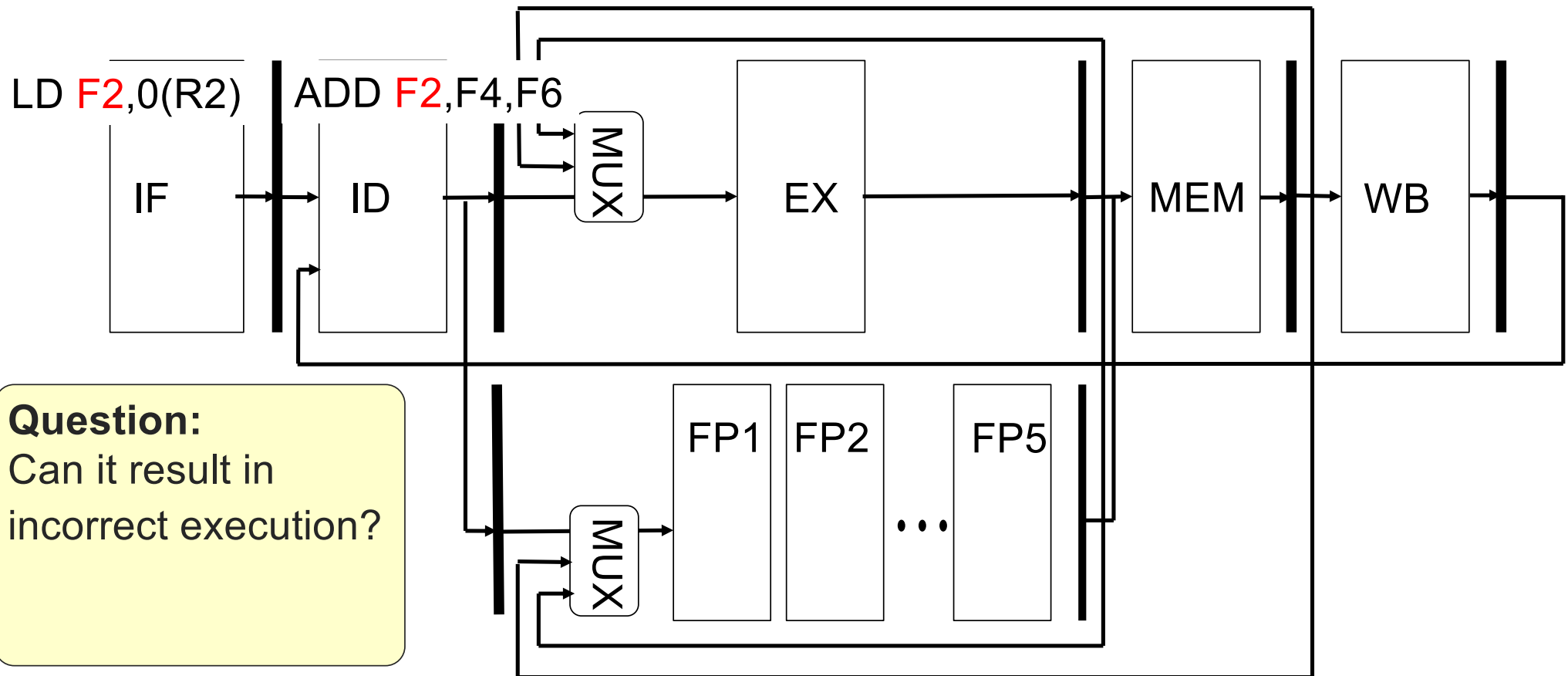
Write-After-Write (WAW)

IF/ID

ID/EX

EX/MEM

MEM/WB



Question:
Can it result in
incorrect execution?

Clock 2

Read-After-Write (RAW)

Write-After-Read (WAR)

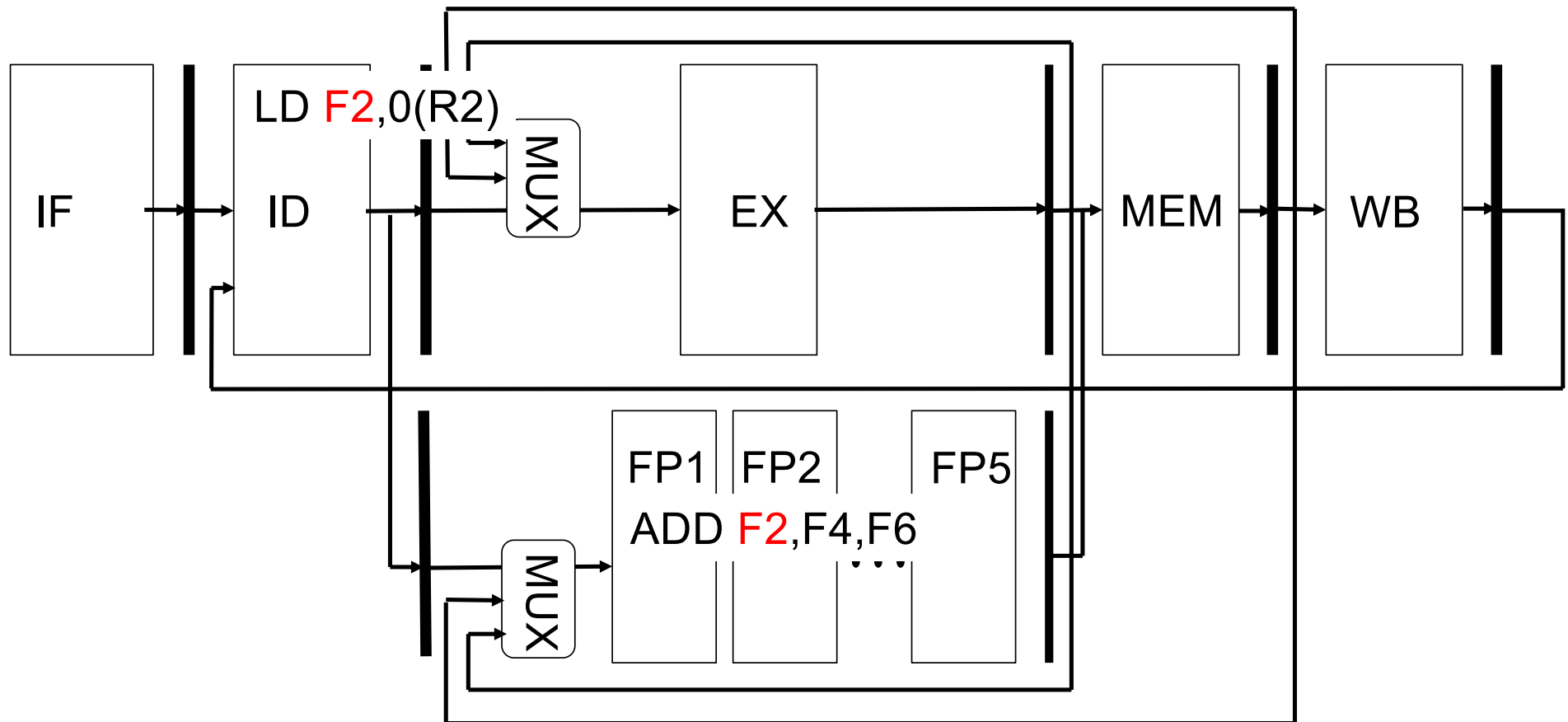
Write-After-Write (WAW)

IF/ID

ID/EX

EX/MEM

MEM/WB



Clock 3

Read-After-Write (RAW)

Write-After-Read (WAR)

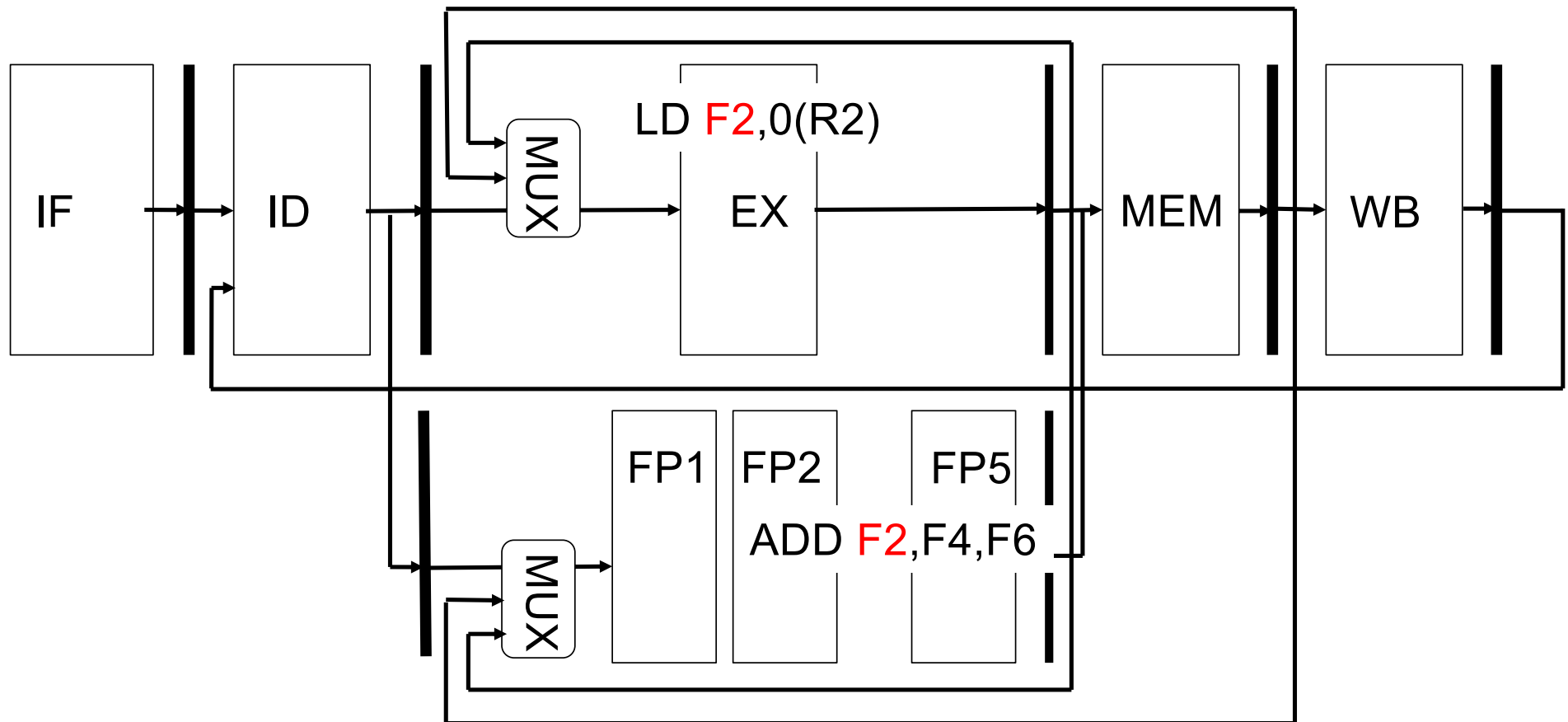
Write-After-Write (WAW)

IF/ID

ID/EX

EX/MEM

MEM/WB



Clock 4

Read-After-Write (RAW)

Write-After-Read (WAR)

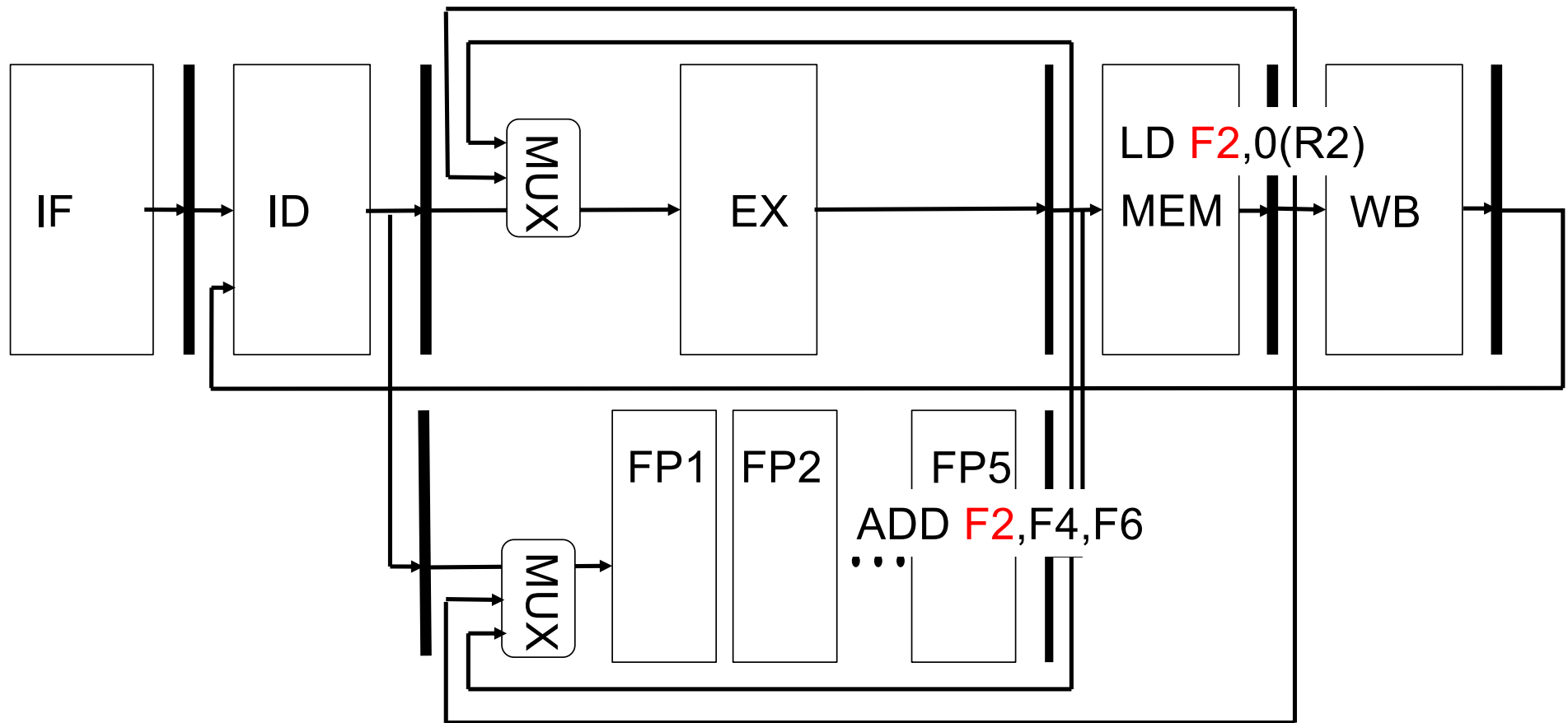
Write-After-Write (WAW)

IF/ID

ID/EX

EX/MEM

MEM/WB



Clock 5

Read-After-Write (RAW)

Write-After-Read (WAR)

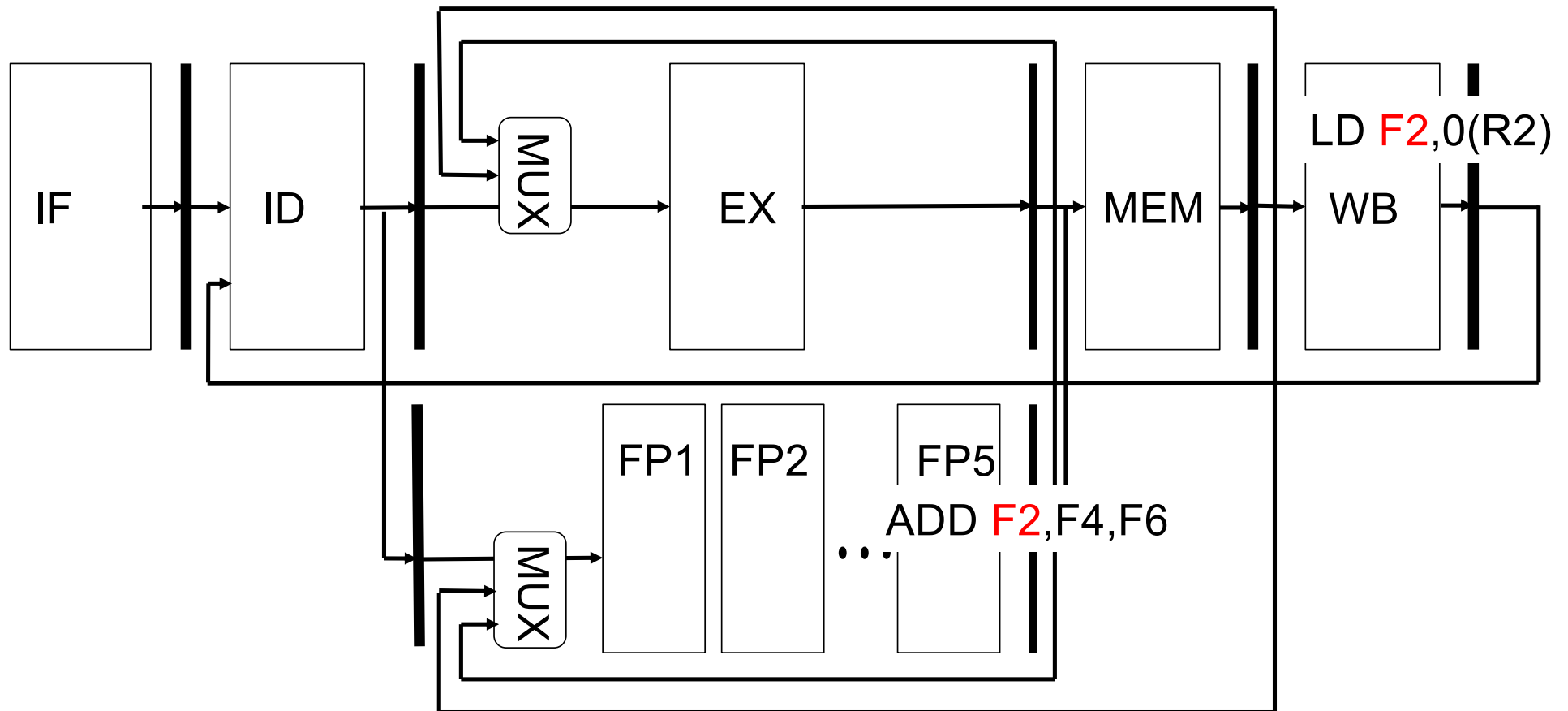
Write-After-Write (WAW)

IF/ID

ID/EX

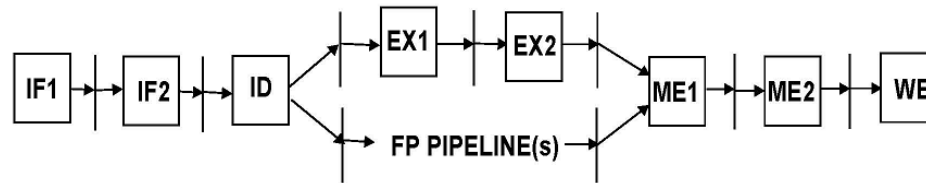
EX/MEM

MEM/WB



Superpipelining and Superscalar Models

Superpipelined CPU



Some stages in the 5-stage pipeline are further pipelined

to increase clock rate

Here, IF, EX and ME are now 2 pipeline stages

Clock is 2X faster

BUT: There is no "free lunch" though:

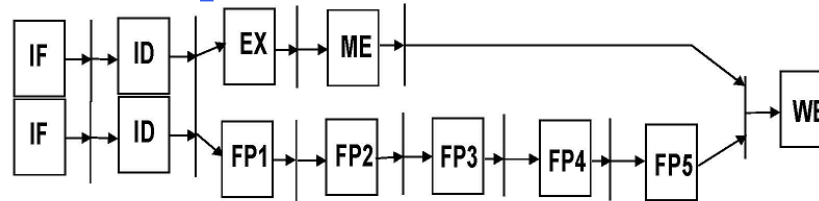
Branch penalty when taken is now 3 clocks

Latencies counted in clock cycles are higher

Assuming unchanged 5-stages for FP

FUNCTIONAL UNITS	LATENCY	INITIATION INTERVAL
INTEGER ALU	1	1
LOAD	3	1
FP OP	4	1(5 IF NOT PIPELINED)

Superscalar CPU



Fetch, decode and execute up to 2 instructions per clock

Same clock rate as basic pipeline

Easy (in this case) because of the the 2 (INT and FP) register files

Issue a pair of instructions:

Pair must be integer/branch/ memory AND FP (compiler can do that)

Instruction pair must be independent and has no hazard with prior instr.

Same latencies as basic pipeline; branches execute in EX

Exceptions are more complex to deal with

Hard to build more than 2-way

IPC (instructions per clock) instead of CPI ($IPC=1/CPI$)

Add superpipelining to superscalar

Quiz 2.2

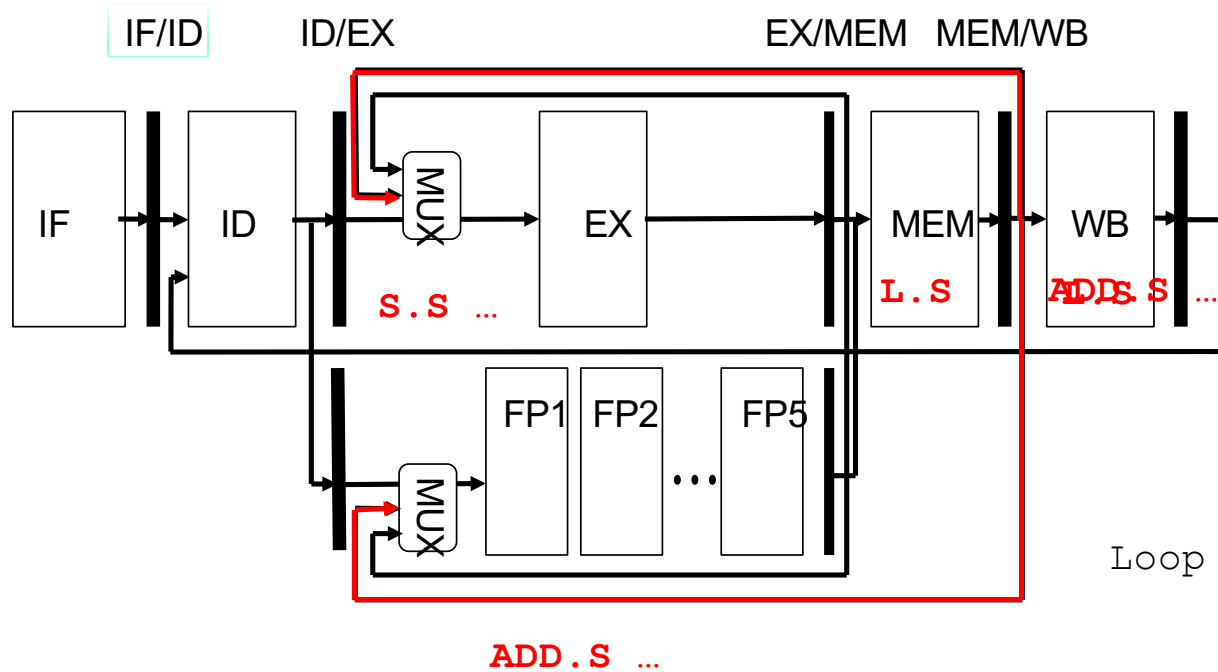
	CLOCK->	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
I1	DIV.D F1,F2,F3	IF	ID	FP1	FP2	FP3	FP4	FP5	ME	WB		
I2	ADD R1,R2,R3	IF	ID	EX	ME	WB						
I3	MULT.D F4,F5,F6		IF	ID	FP1	FP2	FP3	FP4	ME	WB		
I4	ADD R4,R1, R6		IF	ID	EX	ME	WB					

Which of the following statements are incorrect

- a) Instructions finish in this order: I1, I2, I3 and I4
- b) Instructions finish in this order: I2, I4, I1, I3
- c) If the registerfile has two write ports I1 and I3 both finish in C9
- d) A single-issue pipeline would finish the sequence in C12

Static Scheduling

Local Static Instruction Scheduling



L.S	F0, 0 (R1)	(1)
L.S	F1, 0 (R2)	(1)
ADD.S	F2, F1, F0	(2)
S.S	F2, 0 (R1)	(5)
ADDI	R1, R1, #4	(1)
ADDI	R2, R2, #4	(1)
SUBI	R3, R3, #1	(1)
BNEZ	R3, Loop	(3)

Loop	L.S F0,0(R1)	(1)
	L.S F1,0(R2)	(1)
	ADD.S F2,F1,F0	(2)
	S.S F2,0(R1)	(5)
	ADDI R1,R1,#4	(1)
	ADDI R2,R2,#4	(1)
	SUBI R3,R3,#1	(1)
	BNEZ R3,Loop	(3)

Question:
How could we statically re-schedule the instructions to eliminate hazards?

Loop	L.S F0,0(R1)	(1)
	L.S F1,0(R2)	(1)
	SUBI R3,R3,#1	(1)
	ADD.S F2,F1,F0	(1)
	ADDI R1,R1,#4	(1)
	ADDI R2,R2,#4	(1)
	S.S F2,-4(R1)	(3)
	BNEZ R3,Loop	(3)

Cycles: 12

Global Instruction Scheduling

Loop L.S F0,0(R1) (1)
 L.S F1,0(R2) (1)
 ADD.S F2,F1,F0 (2)
 S.S F2,0(R1) (5)
 ADDI R1,R1,#4 (1)
 ADDI R2,R2,#4 (1)
 SUBI R3,R3,#1 (1)
 BNEZ R3,Loop (3)

Loop L.S F0,0(R1) (1)
 L.S F1,0(R2) (1)
 ADD.S F2,F1,F0 (2)
 S.S F2,0(R1) (5)
~~ADDI R1,R1,#4 (1)~~
~~ADDI R2,R2,#4 (1)~~
~~SUBI R3,R3,#1 (1)~~
~~BNEZ R3,Loop (3)~~
 L.S F3,4(R1) (1)
 L.S F4,4(R2) (1)
 ADD.S F5,F4,F3 (2)
 S.S F5,4(R1) (5)
 ADDI R1,R1,#**8** (1)
 ADDI R2,R2,#**8** (1)
 SUBI R3,R3,#**2** (1)
 BNEZ R3,Loop (3)

Loop L.S F0,0(R1) (1)
 L.S F1,0(R2) (1)
 L.S F3,4(R1) (1)
 L.S F4,4(R2) (1)
 ADD.S F2,F1,F0 (**1**)
 ADD.S F5,F4,F3 (**1**)
 ADDI R1,R1,#8 (1)
 ADDI R2,R2,#8 (1)
 SUBI R3,R3,#2 (1)
 S.S F2,**-8**(R1) (**1**)
 S.S F5,**-4**(R1) (**1**)
 BNEZ R3,Loop (3)

Question:

How could we statically re-schedule the instructions to eliminate hazards?

Cycles: 14/2 = 7

Loop Unrolling