

For the sequence of instructions shown below, show how they would progress through the pipeline.

For all of these problems:

- Stalls are indicated by placing the code of the stage where the hazard would be discovered in the succeeding square
- We will assume a standard 5 stage pipeline
 - o (IF = Instruction Fetch, ID = Instruction Decode, EX = Execute, M = Memory Access, WB = Write Back)
- Assume that each stage of the pipeline takes just 1 clock cycle to finish.

Example 1:

- Assume that forwarding **HAS NOT** been implemented
- Assume that you **CANNOT** read and write a register in the same clock cycle

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Add \$5, \$3, \$4	IF	ID	EX	M	WB												
Add \$6, \$5, \$7		IF	ID	ID	ID	ID	EX	M	WB								
LW \$7, 0(\$6)			IF	IF	IF	IF	ID	ID	ID	ID	EX	M	WB				
SUB \$1, \$2, \$3							IF	IF	IF	IF	ID	EX	M	WB			
Add \$9, \$7, \$8											IF	ID	ID	ID	EX	M	WB

$CPI = 17/5 = \text{something too big for our liking...} = 3.4$

Example 2:

- Let's do the same problem as before, but now assume that **forwarding HAS been implemented**
- Assume that you CANNOT read and write from the same register in the register file in the same clock cycle

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Add \$5, \$3, \$4	IF	ID	EX	M	WB												
Add \$6, \$5, \$7		IF	ID	EX	M	WB											
LW \$7, 0(\$6)			IF	ID	EX	M	WB										
SUB \$1, \$2, \$3				IF	ID	EX	M	WB									
Add \$9, \$7, \$8					IF	ID	EX	M	WB								

CPI = 9/5 = 1.8 - but finishing instruction every CC

Example 3:

- Like Example 2, assume that **forwarding HAS been implemented**
- Assume that you **CAN** read and write a register in the same clock cycle

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Add \$1, \$6, \$9	IF	ID	EX	M	WB												
Add \$6, \$2, \$4		IF	ID	EX	M	WB											
LW \$7, 0(\$6)			IF	ID	EX	M	WB										
SUB \$1, \$7, \$8				IF	ID	ID	EX	M	WB								
Add \$9, \$1, \$8					IF	IF	ID	EX	M	WB							

Example 4:

- Assume that **forwarding HAS been implemented**
- We will predict that any branch instruction is **NOT TAKEN**
- Branches or Jumps are resolved after the EX stage.
- Assume that register \$2 has the value of 0 and \$3 has the value of 0

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LW \$1, 4(\$9)	IF	ID	Ex	M	WB												
Add \$4, \$1, \$9		IF	ID	ID	EX	M	WB										
Sub \$7, \$4, \$9			IF	IF	ID	EX	M	WB									
BEQ \$2, \$3, X					IF	ID	EX										
Add \$9, \$8, \$7						IF	ID										
And \$4, \$5, \$5							IF										
X: Add \$4, \$5, \$9								IF	ID	EX	M	WB					

Example 5:

- Assume that **forwarding HAS been implemented**
- We will predict that any branch instruction is **TAKEN**
- Branches or jumps are resolved after the EX stage.
- Assume that register \$2 has the value of 0 and \$3 has the value of 0

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LW \$1, 4(\$9)	IF	ID	EX	M	WB												
Add \$4, \$1, \$9		IF	ID	ID	EX	M	WB										
Sub \$7, \$4, \$9			IF	IF	ID	EX	M	WB									
BEQ \$2, \$3, X					IF	ID	EX										
Add \$9, \$8, \$7																	
And \$4, \$5, \$5																	
X: Add \$4, \$5, \$9						IF	ID	EX	M	WB							

Example 6:

- Assume that **forwarding HAS NOT been implemented**
- Assume that a register **CAN BE** written and read in the same CC
- We will predict that any branch instruction is **TAKEN**
- Branches and Jumps are resolved after the EX stage
- Assume that 4(\$9) has the value of 0 and \$3 has the value of 0

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LW \$1, 4(\$9)	F	D	EX	M	WB												
Add \$4, \$1, \$1		F	D	D	D	EX	M	WB									
Addi \$7, \$4, #4			F	F	F	D	D	D	EX	M	WB						
BEQ \$4, \$7, X						F	F	F	D	D	D	EX	M	WB			
Add \$9, \$8, \$7													F	D
And \$4, \$5, \$5														F
X: Add \$4, \$5, \$9									F	F	F	D					