Problem-1: Identify data dependency, if any in following instructions to be processed in a 5-stage pipeline RISC processor

LOAD R1, 64(R2) ; data read from memory and loaded to R1. R2 is base register, referring to starting of data memory

ADD R4, R1, R5 ; R1 is operand

There is data dependency between the pair of instructions. Here Add instruction depends on Load instruction. Dependency type: True Data dependency: RAW

Instructions/clock cycles	1	2	3	4	5	6
LOAD R1, 64(R2)	IF	ID	ALU	MA	WR Result store to R1	
ADD R1, R4, R5		IF	ID R1 is read for 2nd inst CPU reads old value of R1	ALU	MA	WR

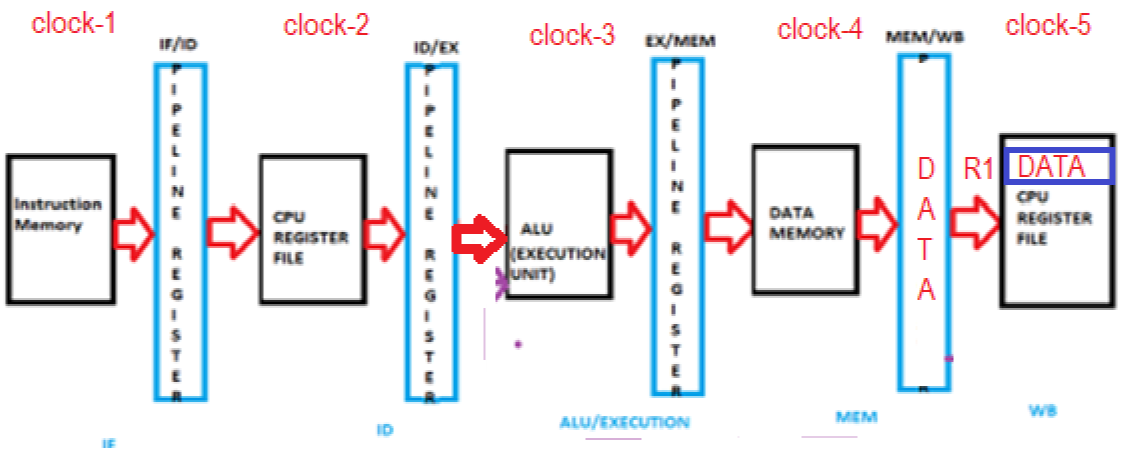
Show the processing of above instructions on a 5-stage pipeline RISC processor (without data forwarding)

Solution: Interloack the pipeline

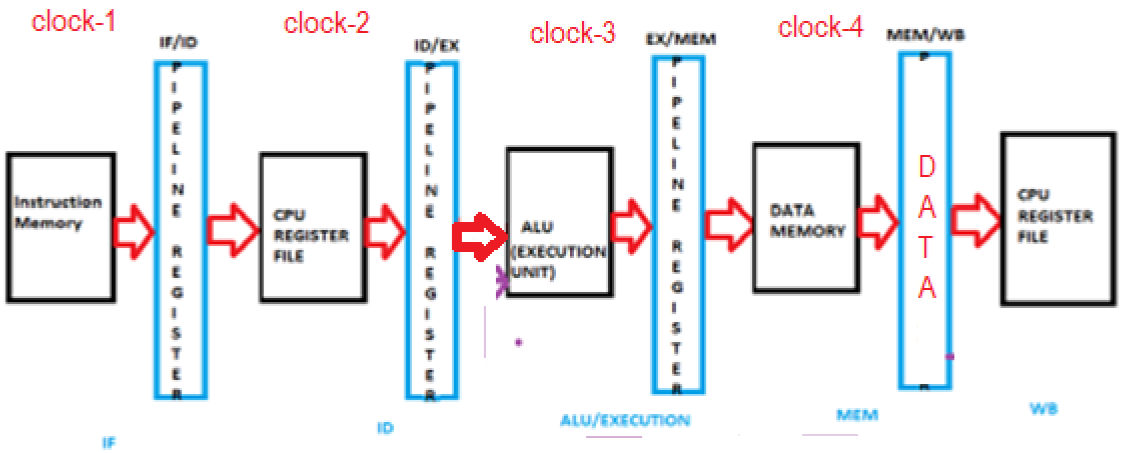
Instructions/clock cycles	1	2	3	4	5	6	7	8
LOAD R1, 64(R2)	IF	ID	ALU	MA	WR Result store to R1			
ADD R1, R4, R5		IF	Interlock 3-clock cycles			ID R1 is read for 2nd inst R1(new value is read	ALU	MA

Show the processing of above instructions on a 5-stage pipeline RISC processor (with data forwarding)

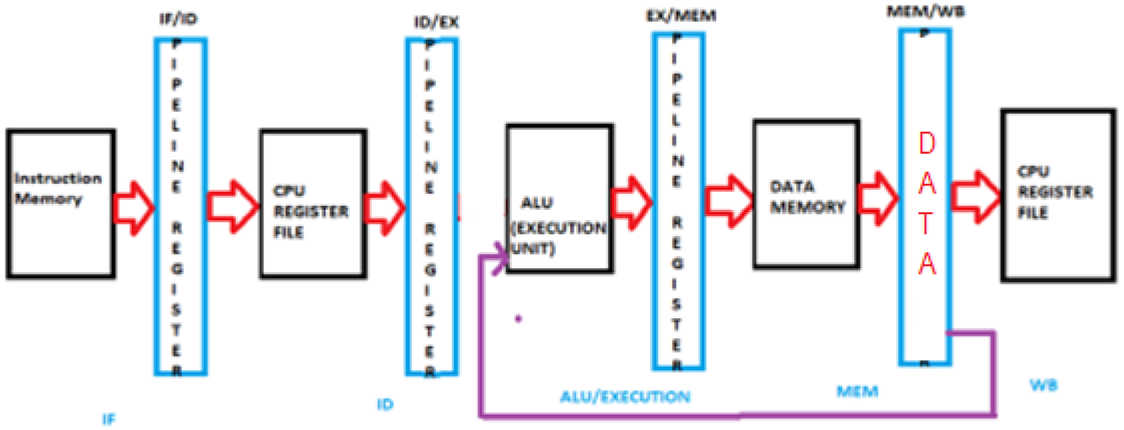
In Load instruction, CPU reads data from data memory and loads to R1 register at clock cycle 5.



However, the data is available at the MEM/WB buffer at clock cycle 4 which is later loaded to R1 and later clock cycle.



To save clock cycles/to reduce delay in pipeline architecture, data can be read earlier at clock cycle-4 from MEM/WB buffer instead of reading later from register R1.   In that case. A data path should be created from MEM/WB buffer to one of the inputs of ALU. However, the ALU operation of succeeding ADD instruction should be delayed by one clock cycle, as shown in table below



Instructions/clock cycles	1	2	3	4	5	6	7
LOAD R1, 64(R2)	IF	ID	ALU	MA Data is in buffer	WR Result store to R1		
ADD R1, R4, R5		IF	ID R1 is NOT read	delay	ALU	MA	WR

Problem-2: Identify data dependency, if any in following instructions to be processed in a 5-stage pipeline RISC processor

ADD R3, R1, R2 ; R3 is result field

LOAD R3, 64(R4) ; data read from memory and loaded to R3. R4 is base register, referring to starting of data memory

There is no data dependency.

Show the processing of above instructions on a 5-stage pipeline RISC processor

INST/CLOCK	1	2	3	4	5	6
ADD R3, R1, R2	IF	ID	ALU	MA	WR RESULT TO R3	
LOAD R3, 64(R4)		IF	ID READ R4	ALU Calculate memory address	MA Read data from data memory	WR DATA loaded to R3

Problem-3: Identify the data dependency, if any in following instructions to be processed in a 5-stage pipeline RISC processor

ADD R3, R1, R2 ; R3 is result field

LOAD R1, 64(R3) ; data read from memory and loaded to R1. R3 is base register, referring to starting of data memory

True data dependency/Read after Write (RAW)

Result of Add instruction is used as base address in the succeeding Load instruction.

Show the hazard.

INST/CLOCK	1	2	3	4	5	6
ADD R3, R1, R2	IF	ID	ALU	MA	WR RESULT TO R3	
LOAD R1, 64(R3)		IF	ID (READ R3 CPU reads old value of R3)	ALU 64 is added to content of R3 and memory	MA Data is read from data memory	WR DATA STORED TO R1 Wrong data

				address is generted		

How to overcome the hazard?

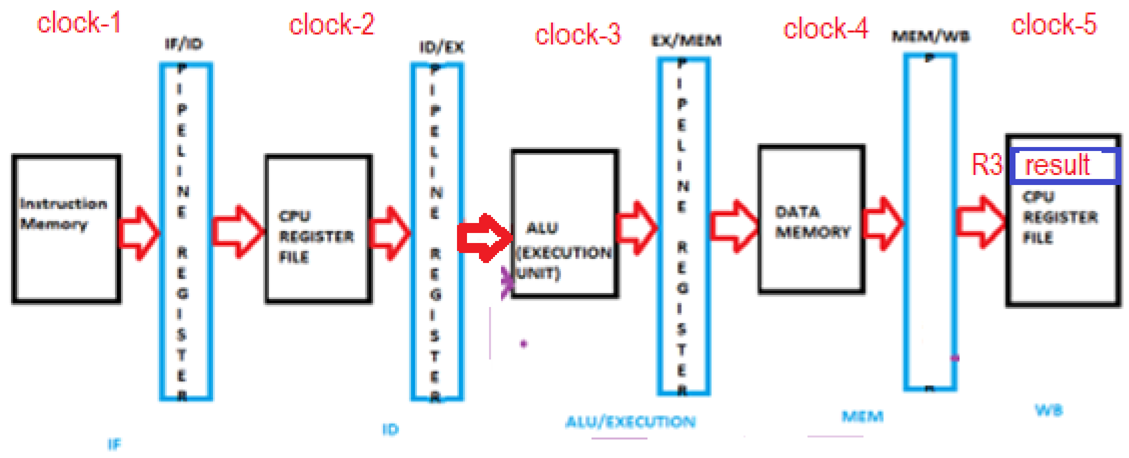
Resut of 1st instruction is saved to R3 at clock cycle-5. To use the updated value of R3 in 2nd instruction, Register R3 for the 2nd instruction should be read at clock cycle 6 or later.

It means that register read for 2nd instruction and subsequent operations should be delayed by at least 3 clock cycles.

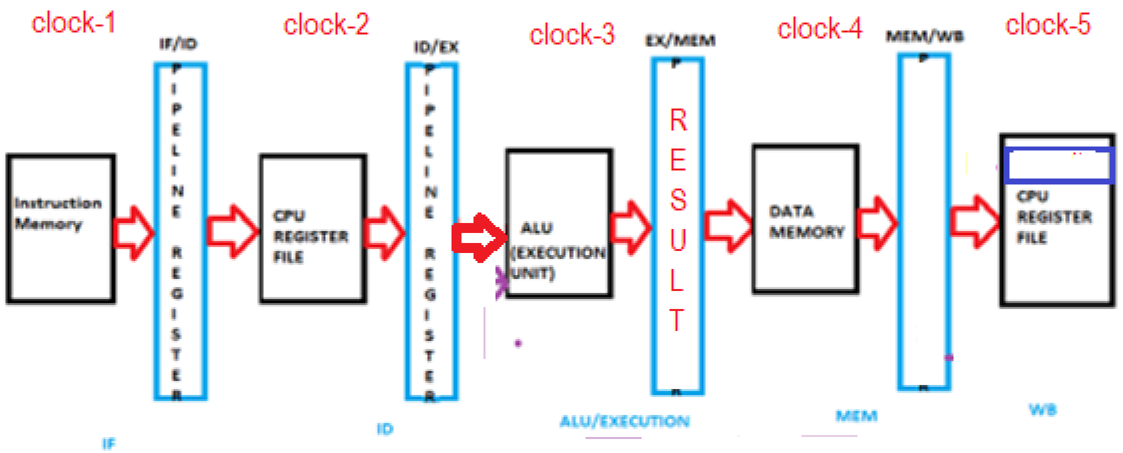
INST/CLOCK	1	2	3	4	5	6	
ADD R3, R1, R2	IF	ID	ALU	MA	WR RESULT TO R3		
LOAD R1, 64(R3)		IF	Stall 3 clock cycles			ID & Read R3	

How to reduce the delay:

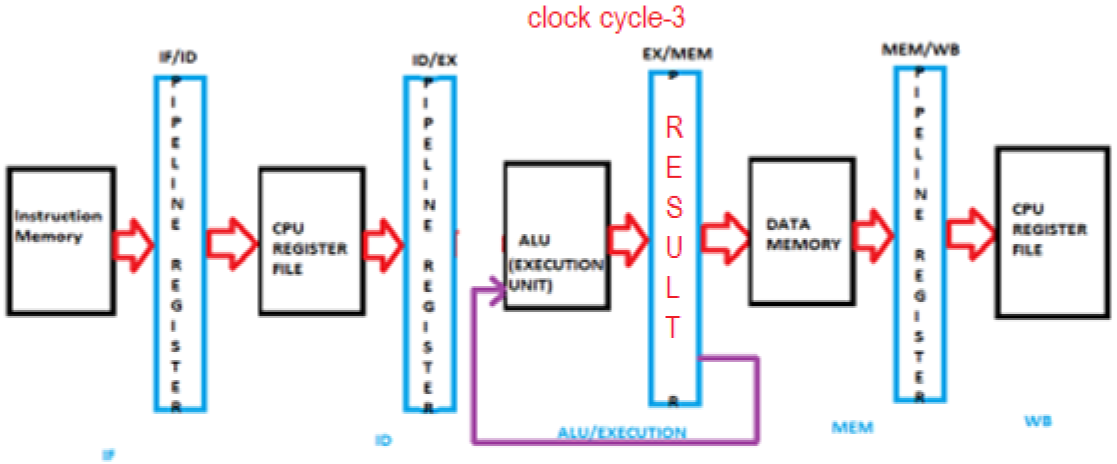
It is noted that result of Add operation is stored to R3 at clock cycle 5.



However the result of ADD instruction is available in EX/MEM buffer at clock cycle 3.



To save clock cycles/to reduce delay in pipeline architecture, result of ADD instruction can be read for Load instruction earlier at clock cycle-3 from EX/MEM buffer instead of reading later from register R3. In that case, a data path should be created from EX/MEM buffer to one of the inputs of ALU. Processing of instructions is shown in table below.



INST/CLOCK	1	2	3	4	5	6
ADD R3, R1, R2	IF	ID	ALU RESULT IN BUFFER	MA	WR RESULT TO R3	
LOAD R1, 64(R3)		IF	ID NOT READ R3	ALU R3 AND 64 ADDED	MA	WR DATA STORED TO R1