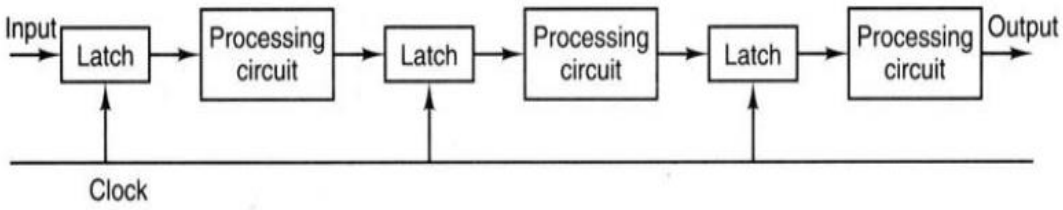
Pipelining and hazards: Problems and solutions

1. What is pipelining?

Pipelining is an implementation technique that allows simultaneous execution of several instructions. The Pipelining architecture is designed like an assembly line for instruction processing. Pipeline architecture involves executing multiple instructions per cycle. Pipelining is one way of improving the overall processing performance of a processor.

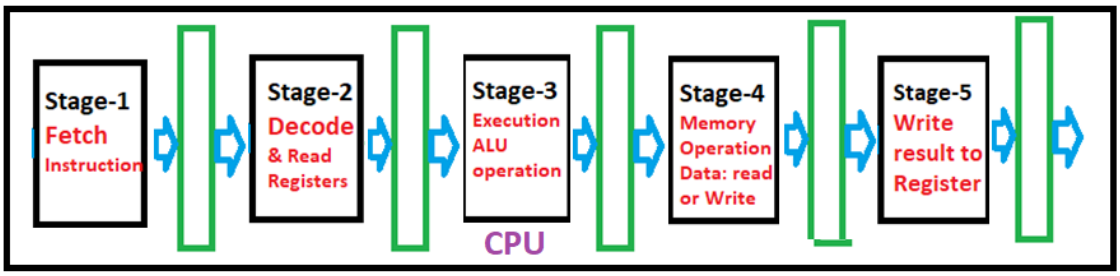
The pipeline design technique splits a sequential process into several subprocesses, called stages or segments. A *stage* performs a particular function and produces an intermediate result. It consists of an input latch, also called a register or buffer, followed by a processing circuit. (A processing circuit can be a combinational or sequential circuit.) The processing circuit of a given stage is connected to the input latch of the next stage. A clock signal is connected to each input latch. At each clock pulse, every stage transfers its intermediate result to the input latch of the next stage. In this way, the final result is produced after the input data have passed through the entire pipeline, completing one stage per clock pulse.



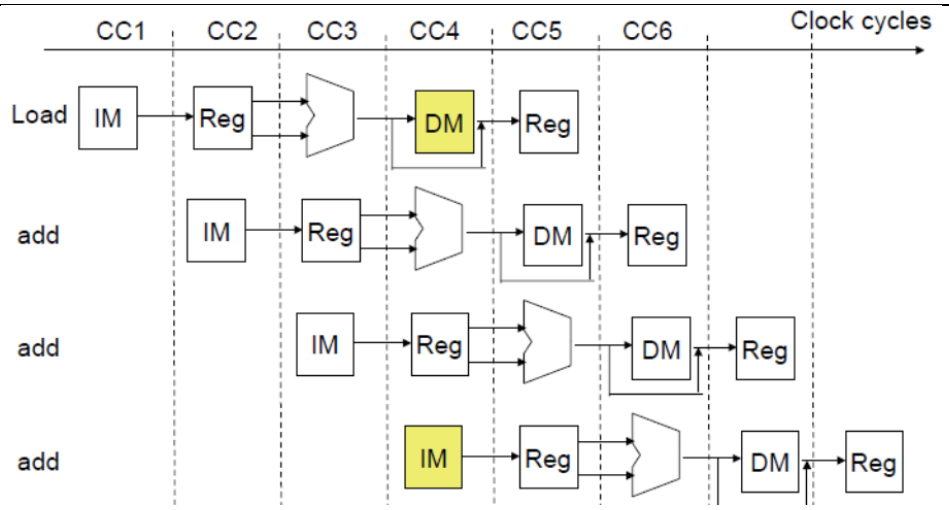
Pipelining is transparent to the programmer; it exploits parallelism at the instruction level by overlapping the execution process of instructions.

2. Describe a 5-stage RISC pipelined processor.

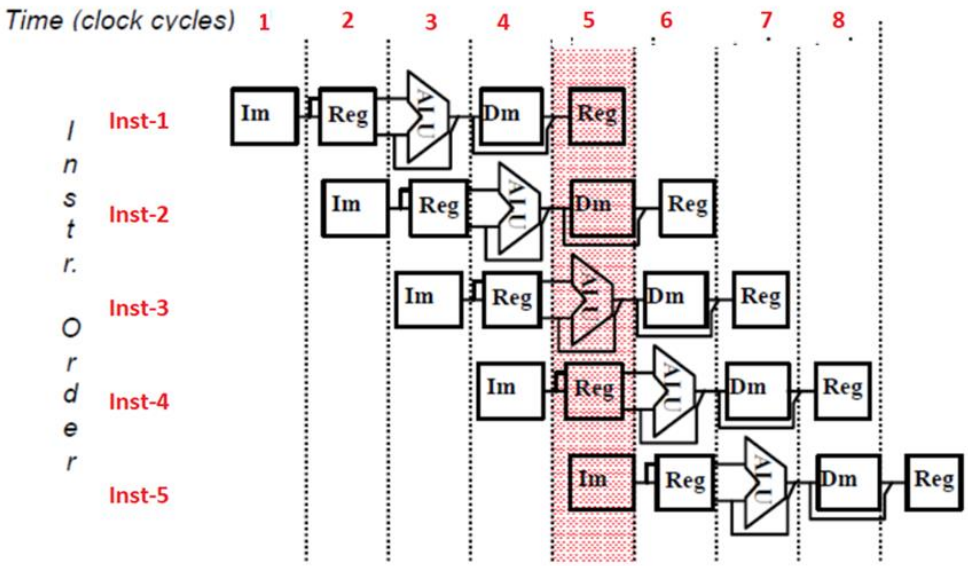
- 5-stage processing of each Instruction, in general, although some instructions may NOT require all stages
- **ALU** instructions are register based
- **LOAD** and **STORE** Instructions are used to read data from RAM and store to RAM
- **Harvard Architecture**: Separate memories for Instructions and Data
- Latches/buffers between stages to allow sections work simultaneously on different Instructions



3.	What is pipeline hazard? What are types of hazards? ▪ Hazards are pipeline events that restrict the pipeline flow ▪ They occur in circumstances where two or more activities cannot proceed in parallel ▪ There are three types of hazard: – Structural Hazards ▪ Arise from resource conflicts, when a set of actions have to be performed sequentially because there is not sufficient resource to operate in parallel – Data Hazards ▪ Occur when one instruction depends on the result of a previous instruction, and that result is not yet available. These hazards are exposed by the overlapped execution of instructions in a pipeline – Control Hazards ▪ These arise from the pipelining of branch instructions, and other activities that change the PC.
4.	What is structural hazard? How to avoid structural hazards? • Structural hazards occurs when two instruction need same hardware resource at same time – Can resolve in hardware by stalling newer instruction till older instruction finished with resource • A structural hazard can always be avoided by adding more hardware to design – E.g., if two instructions both need a port to memory at same time, could avoid hazard by adding second port to memory
5.	Explain structural hazard with diagram? At clock cycle 4, 1st(load) and add(4th) instructions need to access RAM. If a single memory is used, both IF and MA cannot proceed at clock cycle 4. The pipeline will be stalled



6. Example of structural hazard



- Both the decode and write back stage have to access the register file.
- There is only one registers file. A structural hazard!!
- Solution: Write early, read late
  - Writes occur at the clock edge and complete long before the end of the cycle
  - This leave enough time for the outputs to settle for the reads.
- Hazard avoided!

7. **What is Read after write (RAW), or true dependency?** An instruction modifies a register or memory location and a succeeding instruction reads the data in that memory or register location. A hazard occurs if the read takes place before the write operation is complete. (read too soon)
add r1, r2, r3
add r3, r4, r5

8. How to resolve data hazard?

Strategy 1: *Wait for the result to be available by freezing earlier pipeline stages* → *stall*

Strategy 2: *Route data as soon as possible after it is calculated to the earlier pipeline stage* → *bypass*

9. What is pipeline interlocking? Explain.

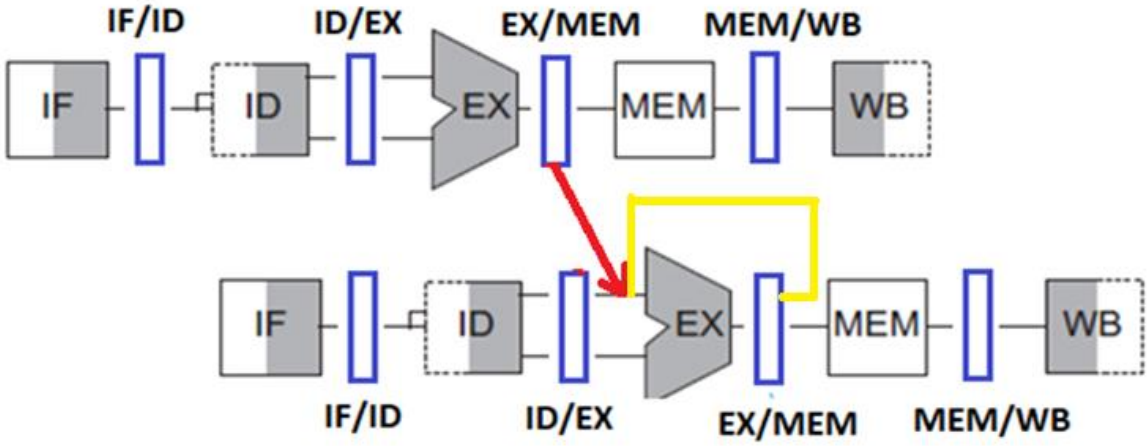
Interlocks-wait for the result to be available by freezing earlier pipeline stages

Instructions	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ADD R1,R2, R3	IF	ID	EX	MA	WB					
SUB R6, R1, R5		IF	ID	ID	ID	ID	EX	MA		
Instruction-3			IF	IF	IF	IF	ID	EX		
Instruction-4			Stalled stages				IF	ID		
Instruction-5								IF		

10. What is Forwarding?

A new datapath should be created to pass the data from the output of the ALU to its input for the following instruction in next time step.

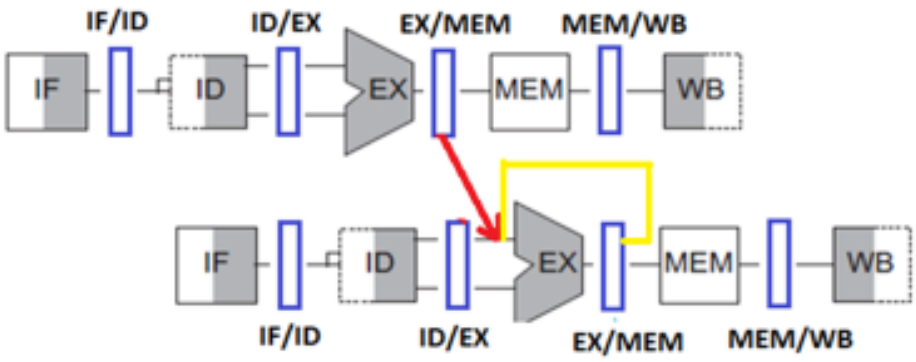
Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
ADD R1,R2, R3	IF	ID	EX	MA	WB						
SUB R6, R1, R5		IF	ID	EX	MA	WB					
Instruction-3			IF	ID	EX	MA					
Instruction-4				IF	ID	EX					
Instruction-5					IF	ID					
Instruction-6						IF					

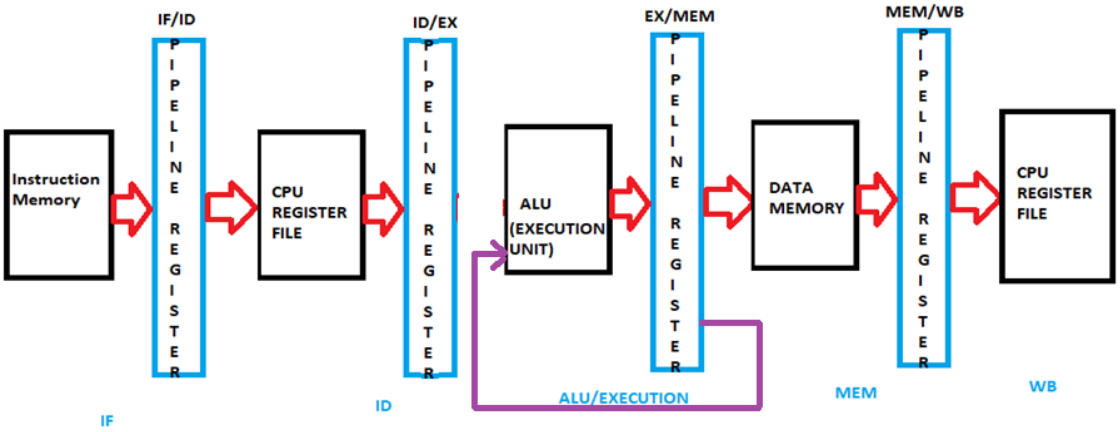


11. Forwarding: A new datapath should be created to pass the data from the output of the ALU to its input for the following instruction in next time step.

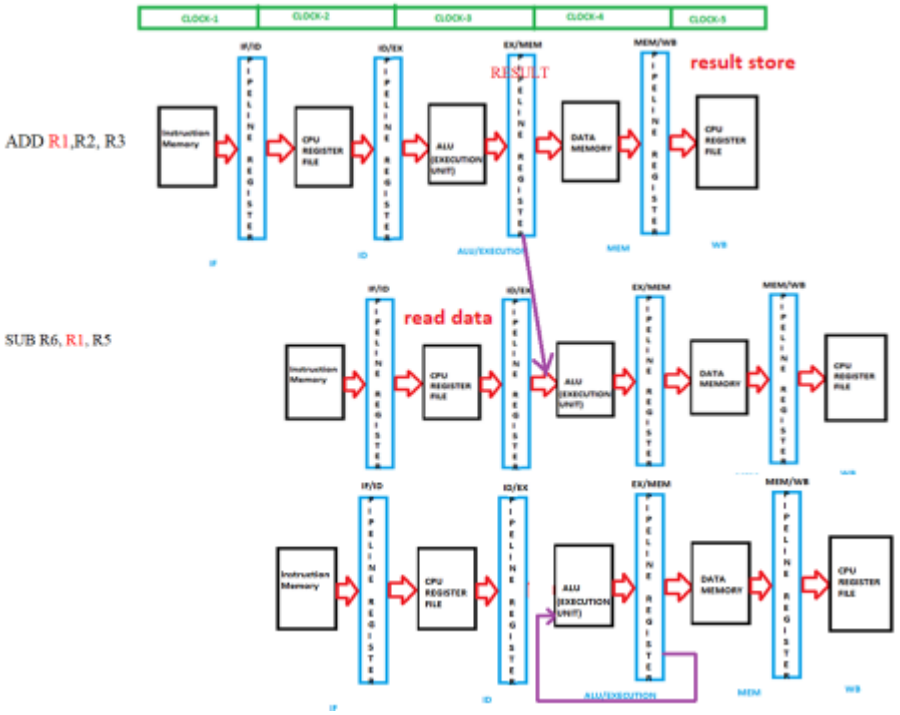
For Instruction: SUB R6, R1, R5; ALU will get one of the inputs (the result of preceding instruction) from pipeline register EX/MEM at clock cycle-4 instead of reading from R1 at clock cycle-3. .

Instruction	1	2	3	4	5	6	7
ADD R1,R2, R3	IF	ID	EX	MA	WB		
SUB R6, R1, R5		IF	ID	EX	MA	WB	
Instruction-3			IF	ID	EX	MA	
Instruction-4				IF	ID	EX	
Instruction-5					IF	ID	
Instruction-6						IF	





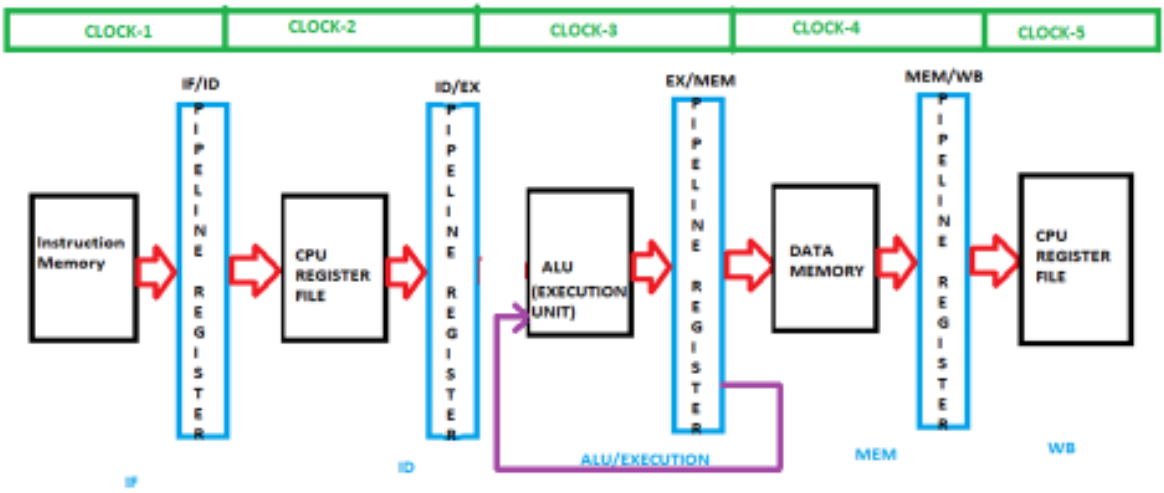
12.



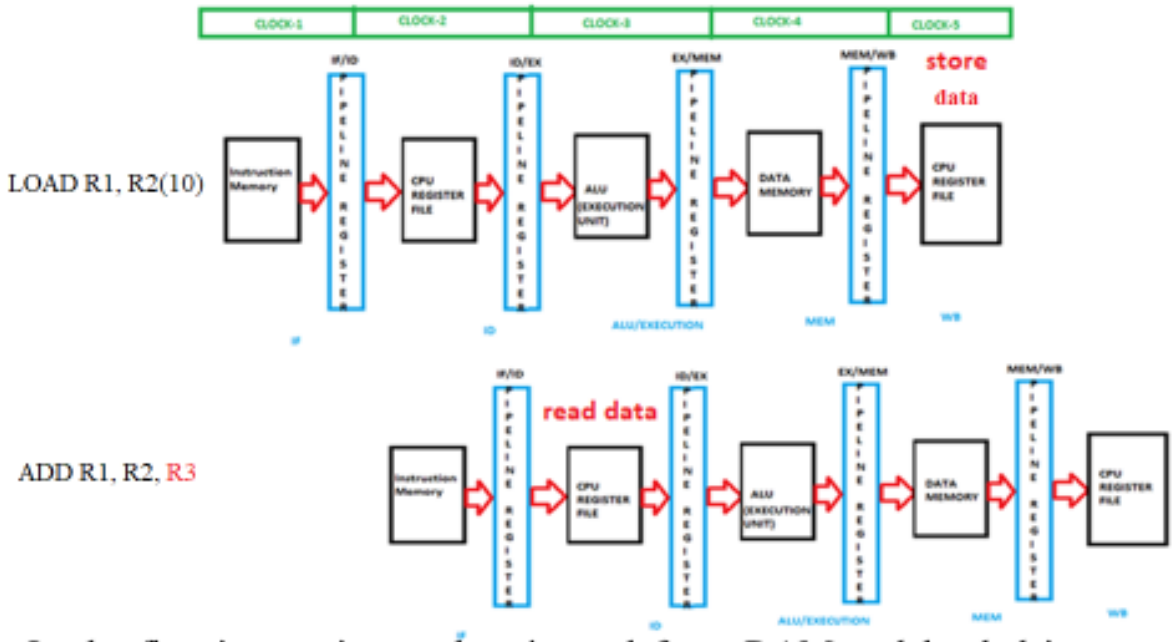
13.

Forwarding: A new datapath should be created to pass the data from the output of the ALU to its input for the following instruction in next time step.
For Instruction: SUB R6, R1, R5; ALU will get one of the inputs (the result of preceding instruction) from pipeline register EX/MEM at clock cycle-4 instead of reading from R1 at clock cycle-3.

Using forwarding, delay/stall can be fully avoided in case of true data dependency as noticed in following pair of instruction.
ADD R1, R2, R3; R1 is result field
SUB R6, R1, R5; R1 is operand



14.

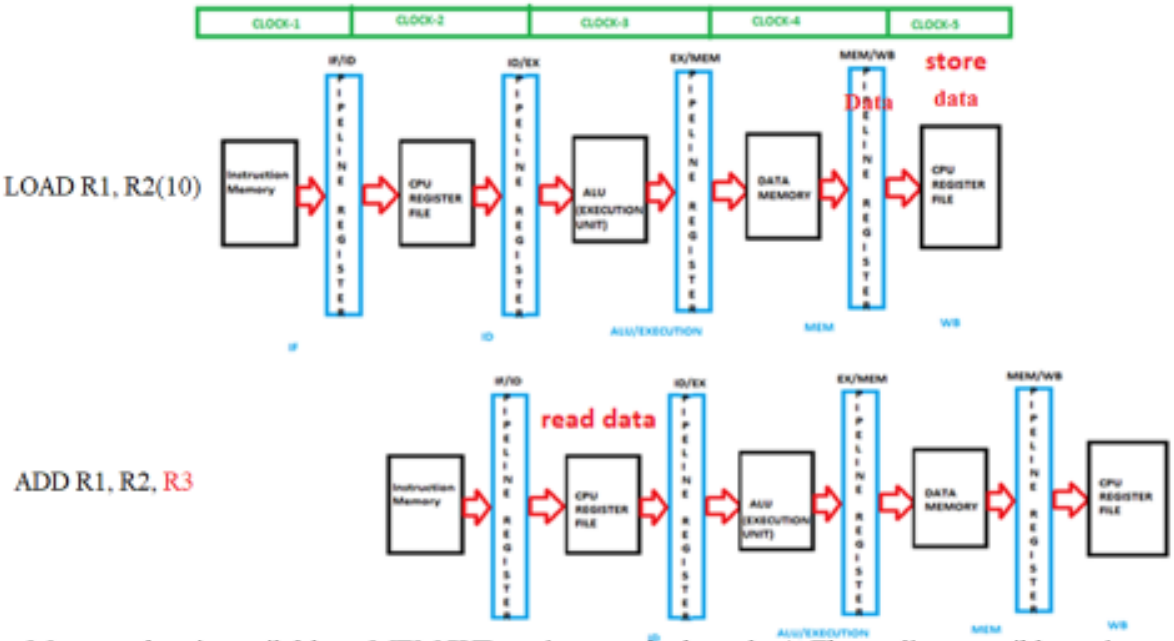


In the first instruction, a data is read from RAM and loaded into a Register then the same data is used as one of the operands in an ALU instruction.

LOAD R1, R2(10)

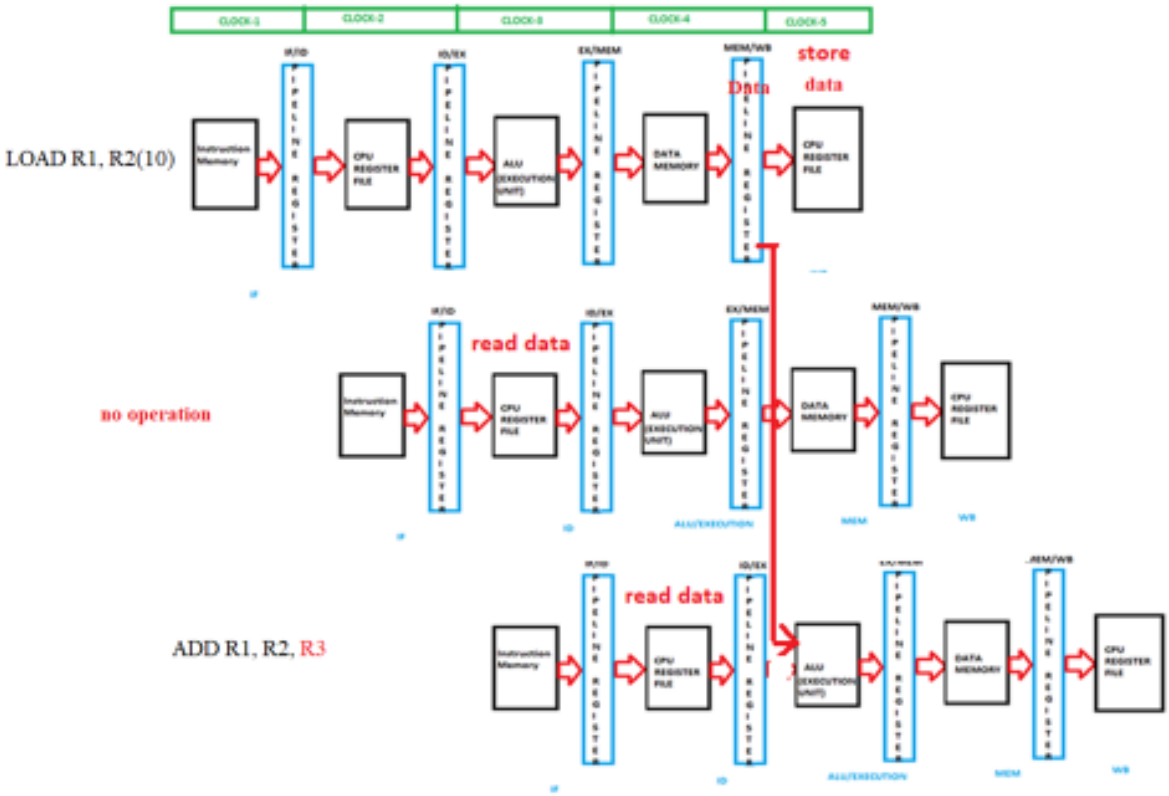
ADD R1, R2, R3

15.



Memory data is available at MEM/WB register at clock cycle-4. The earliest possible option to forward this data to following instruction is also clock cycle-4 or later. The ALU operation of the ADD instruction can also be performed at clock cycle-5 at the earliest if data read from memory is passed to ALU from MEM/WB register at clock cycle-4. So the pipeline must be delayed by 1 clock cycle despite forwarding path from MEM/WB to input to ALU.

16.



One clock delay and forwarding (MEM/WB to ALU) is required.

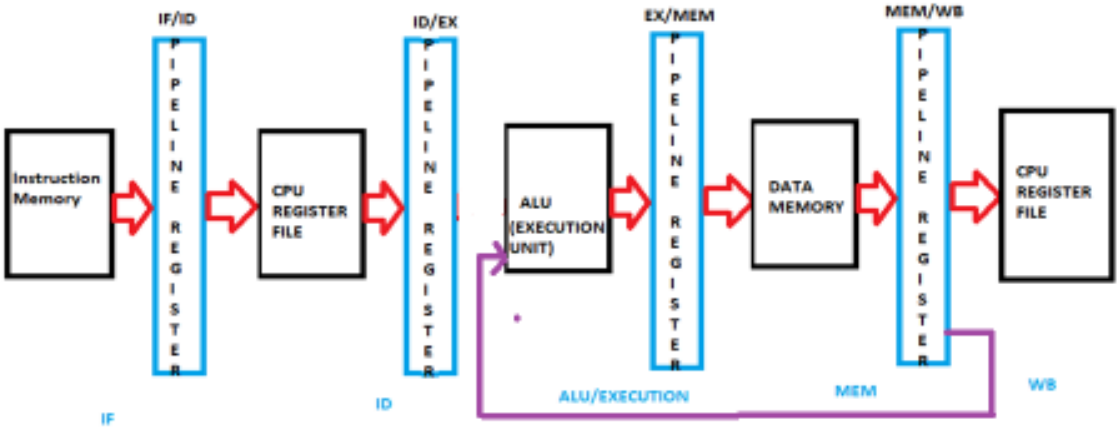
17.

Forwarding in case of Load followed by ALU instruction

In the first instruction, a data is read from RAM and loaded into a Register then the same data is used as one of the operands in an ALU instruction. One clock delay and forwarding is required.

LOAD R1, R2(10)

ADD R1, R2, R3



18.

To overcome data dependencies

- **The compiler must fill the delay slots**
- **Ideally, with useful instructions, but nops will work too.**

add R0, R10, R11
sub R12, R0, R13
add R13, R0, R14
and R17, R15, R14

add R0, R10, R11
and R17, R15, R14
nop
sub R12, R0, R13
add R13, R0, R14

19.

Problem (RISC processor): IF-ID-ALU-MA-WB

Assume a machine using Five-stage pipelining runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU jumps to Instruction-8. Show the time steps of pipelining stages assuming that condition is evaluated TRUE.

If the condition if TRUE, CPU reads instruction from new address and the new address is calculated at the end of ALU phase of the five stag e pipelined processor.

Inst	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Inst-1	IF	ID	ALU	MA	WB									
Inst-2		IF	ID	ALU	MA	WB								
Inst-3			IF	ID	ALU									
Inst-4				IF	ID									
Inst-5					IF									
Inst-6														
Inst-7														
Inst-8						IF	ID	ALU	MA					
Inst-9							IF	ID	ALU					
Inst-10								IF	ID					
Inst-11									IF					

20.

Problem (RISC processor): IF-ID-ALU-MA-WB

Assume a machine using Five-stage pipelining runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU jumps to Instruction-8. Show the time steps of pipelining stages assuming that condition is evaluated TRUE.

If the condition if TRUE, CPU reads instruction from new address and the new address is calculated at the end of Memory Access (MA) Phase of the five stage pipelined processor.

Inst	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Inst-1	IF	ID	ALU	MA	WB									
Inst-2		IF	ID	ALU	MA	WB								
Inst-3			IF	ID	ALU	MA								
Inst-4				IF	ID	ALU								
Inst-5					IF	ID								
Inst-6						IF								
Inst-7														
Inst-8							IF	ID	ALU	MA				
Inst-9								IF	ID	ALU				
Inst-10									IF	ID				
Inst-11										IF				

21.

Example: Control Hazard (CISC processor)

- When a branch is executed, Program Counter is not affected until the branch instruction reaches the EX stage. By this time 3 instructions have been fetched from the fall-through path.
- If the condition if TRUE, CPU reads instruction from new address and the new address is calculated at the end of EX Phase of the five stage pipelined processor.

Instruction\cycle	1	2	3	4	5	6	7	8
SUB R1, R3, R3	IF	ID	MA	EX	WR			
BEQZ R1, label		IF	ID	MA	EX	WR		
OR R8, R4, R9			IF	ID	MA	Condition in **BEQZ R1, label** Is evaluated TRUE. Clear Instructions in FU, DU, MU, EU. Read new Instruction as indicated by label		
MOV R2, R5				IF	ID			
DIV R10, R7					IF			
label: XOR R2, R3, R11						IF	ID	MEM
SUB R5, R3, R1							IF	ID
ADD M1, R6, R8								IF

22.	For the following code, identify data hazard, structural hazard and control hazard, if any in consecutive instructions.
	
	Show the processing of following instructions on a 5-stage RISC processor (without data forwarding)
	
	Show the processing of following instructions on a 5-stage RISC processor (with data forwarding). Compare above two cases.
	
	add r3, r1, r2 ; r3 destination
	and r5, r3, r4 ; r5 result field
	load r6, 24(r3) ; r6 destination
	add r2, r6, r3 ; r2 result field
	store r6, 12(r2) ; r6 source of data

23.	For the following code, identify data hazard, structural hazard and control hazard, if any in consecutive instructions.
	For the following code, identify data hazard, structural hazard and control hazard, if any in consecutive instructions.
	
	Show the processing of following instructions on a 5-stage RISC processor (without data forwarding)
	
	Show the processing of following instructions on a 5-stage RISC processor (with data forwarding). Compare above two cases.
	
	load r6, 24(r3) ; r6 destination
	add r3, r1, r6 ; r3 result field
	And r5, r3, r4 ; r5 result field
	add r2, r6, r5 ;r2 result field
	store r6, 12(r2);

24.	For the code sequence below, state whether it must stall, can avoid stalls using only forwarding, or can execute without stalling or forwarding. (RISC processor)
	add R1,R0,1
	add R2,R0,2
	add R3,R0,2
	add R3,R0,4
	add R5,R0,5

25.	For the following code, identify data hazard, structural hazard and control hazard, if any in consecutive instructions. (RISC processor)
	ADD R1, R2, R3
	SUB R4, R1, R5
	AND R6, R1, R7
	OR R8, R1, R9
	XOR R10, R1, R11

26. For the following code, identify data hazard, structural hazard and control hazard, if any in consecutive instructions. (RISC processor)

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

27. For the following code, identify data hazard, structural hazard and control hazard, if any in consecutive instructions. (RISC processor)

SUB R4, R1, R3
ADD R1, R2, R3
MUL R6, R1, R7

28. A RISC processor is designed to have following instruction types. It is decided to process instructions in five sub-tasks/steps, in general and processing circuits of different sub-tasks require following time slots.

Types of instructions	Sub-tasks/steps required				
	Instruction fetch	Decode and Register read	ALU operation	Memory access	Register write
ALU	6ns	4ns	4ns		2ns
LOAD	6ns	4ns	4ns	6ns	2ns
STORE	6ns	4ns	4ns	6ns	
BRANCH	6ns	4ns	4ns		

A) To implement the Instruction Set Architecture, deign
   i. Single cycle data-path
   ii. Multi-cycle data-path
   iii. Pipeline data-path

You need to use diagrams, as required, and explain the design process in detail.

A benchmark program having one million instructions has the following distribution of instructions is run on above three types of data-path.

Types of Instructions	Distribution of Instructions in percentages	For multi-cycle (clock cycle = 6ns) (CPI)
ALU	50%	4
LOAD	20%	5
STORE	20%	4
BRANCH	10%	3

B) For above three types of data-paths, calculate:
   i. Average CPI
   ii. Runtime of the program ; 10^6 x CPI x clock period =

iii. Speedup of pipeline data-path compared to Single cycle and Multi-cycle implementations

C) Assume that each BRANCH instruction on pipeline data-path causes 2 clock cycles delay, calculate:

i. Average CPI
ii. Runtime of the program
iii. Speedup of pipeline data-path compared to Single cycle and Multi-cycle implementations

29. **Practice problem: single-cycle, multi-cycle and pipeline implementation of CPU micro-architecture**

For a RISC processor, arrange the following tasks in order, if not, as required to process instructions, in general, then explain the new order, if suggested. Also explain the sequences of tasks required to process ALU, Load and Store instructions.

Task	***Time required***
Fetch	4ns
Decode and Register read	2ns
Memory access	4ns
Write to register	2ns
ALU	2ns

30. *For a CISC processor, arrange the following tasks in order, if not, as required to process instructions, in general, then explain the new order, if suggested. Also explain the sequences of tasks required to process ALU (register-based), ALU (memory based), Load and Store instructions.*

Task	***Time required***
Fetch	4ns
Decode and Register read	2ns
ALU	2ns
Write to register	2ns
Memory access	4ns

31. *For a RISC processor*

Task	***Time required***
Fetch	4ns
Decode and Register read	2ns
ALU	2ns
Memory access	4ns
Write back to register	2ns

	For single cycle implementation: a) What is clock cycle in single cycle implementation? b) What is processing time of a register-based ALU instruction in single cycle implementation? c) What is processing time of a Load instruction in single cycle implementation? d) What is processing time of a Store instruction in single cycle implementation? e) What is CPI for a register-based ALU instruction in single cycle implementation? f) What is CPI for a Load instruction in single cycle implementation? g) What is CPI for a store instruction in single cycle implementation? h) What is average CPI of a program having 100 instructions: 50% register-based ALU instruction, 25% load instructions and 25% store instructions. i) How many clock cycles are required to run a program having 100 instructions: 50% register-based ALU instruction, 25% load instructions and 25% store instructions? For multi-cycle implementation: a) What is clock cycle in multi-cycle implementation? b) What is processing time of a register-based ALU instruction in multi-cycle implementation? c) What is processing time of a Load instruction in multi-cycle implementation? d) What is processing time of a Store instruction in multi-cycle implementation? e) What is CPI for a register-based ALU instruction in multi-cycle implementation? f) What is CPI for a Load instruction in multi-cycle implementation? g) What is CPI for a store instruction in multi-cycle implementation? h) What is average CPI of a program having 100 instructions: 50% register-based ALU instruction, 25% load instructions and 25% store instructions. i) How many clock cycles are required to run a program having 100 instructions: 50% register-based ALU instruction, 25% load instructions and 25% store instructions? For pipeline implementation: a) What would be optimum value of K for pipeline implementation? b) What is clock cycle in pipeline implementation? c) What is speed up of k-stage of pipeline implementation if clock is 3MHz and $n \approx k$? d) What is speed up of k-stage of pipeline implementation if clock is 5MHz and $n >> k$? e) What is speed up of k-stage of pipeline implementation if clock is 6MHz and $n < k$? f) What is CPI for $n >> k$? g) What is average CPI of a program having 100 instructions: 50% register-based ALU instruction, 25% load instructions and 25% store instructions. h) How many clock cycles are required to run a program having 100 instructions: 50% register-based ALU instruction, 25% load instructions and 25% store instructions.
32.	**Practice problems: pipeline hazards** a) Compare the performance of pipeline architecture with Single Memory vs Multiple Memory systems. b) Justify the use of Harvard architecture with pipeline processor. c) What is structural hazards, give examples and state the ways how to overcome those. d) How registers are accessed in pipeline architecture? Justify e) Would you prefer register Read and Write in the same clock cycle or at different clock cycle? Explain f) If same register is attempted to Read and Write in different instructions at the same clock cycle? How to manage that?

g) What is data hazard? Explain with examples.
h) How to overcome data hazard? Explain with examples.
i) Explain data hazards in the following cases

Case-1:

ADD R1, R2, R3: R3 is result field
SUB R3, R4, R5; R5 is result field

Case-2:

LOAD R1, R2(10) ; R1 is destination
ADD R4, R1, R3 ; R4 is result field
How to overcome data hazard in case-1?
How to overcome data hazard in case-2?

Compare the technique used in case-1 and case-2.

j) What is pipeline stall?
k) What is forwarding? Show its implementation in above cases (case-1 and case-2) and discuss, if there is any difference.
l) What is conditional branch instruction? Explain the operation of following instruction in case of MIPS architecture: BEQ R1, R2, OFFSET
m) What is control hazard in pipeline processor? Explain with examples.
n) Show the timing diagram of a pipeline processor (RISC) in case of conditional branch instructions, if condition is evaluated TRUE
o) Show the timing diagram of a pipeline processor (CISC) in case of conditional branch instructions, if condition is evaluated TRUE.
p) Show the timing diagram of a pipeline processor (RISC) in case of conditional branch instructions, if condition is evaluated FALSE
q) Show the timing diagram of a pipeline processor (CISC) in case of conditional branch instructions, if condition is evaluated FALSE.
r) Compare single cycle, multicycle and pipeline architecture.
s) Assume a machine using Five-stage pipelining runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TURE, program control returns to Instruction-4. Show the time steps of pipelining stages assuming that both Instructions 3 and 8 are evaluated TRUE.
t) Assume a machine using Five-stage pipelining runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TURE, program control returns to Instruction-4. Show the time steps of pipelining stages assuming that both Instructions 3 and 8 are evaluated FALSE.
u) Assume a machine using Five-stage pipelining runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TURE, program control returns to Instruction-4. Show the time steps of pipelining stages assuming that Instruction-3 is evaluated TRUE and Instruction-8 is evaluated FALSE.
v) Assume a machine using Five-stage pipelining runs a program. Instruction-3 is a conditional branch instruction. If the condition is TRUE, CPU skips next three instructions. Instruction-8 is also a conditional branch instruction and if it is TURE, program control returns to Instruction-

	Show the time steps of pipelining stages assuming that Instruction-3 is evaluated FALSE and Instruction-8 is evaluated TRUE.