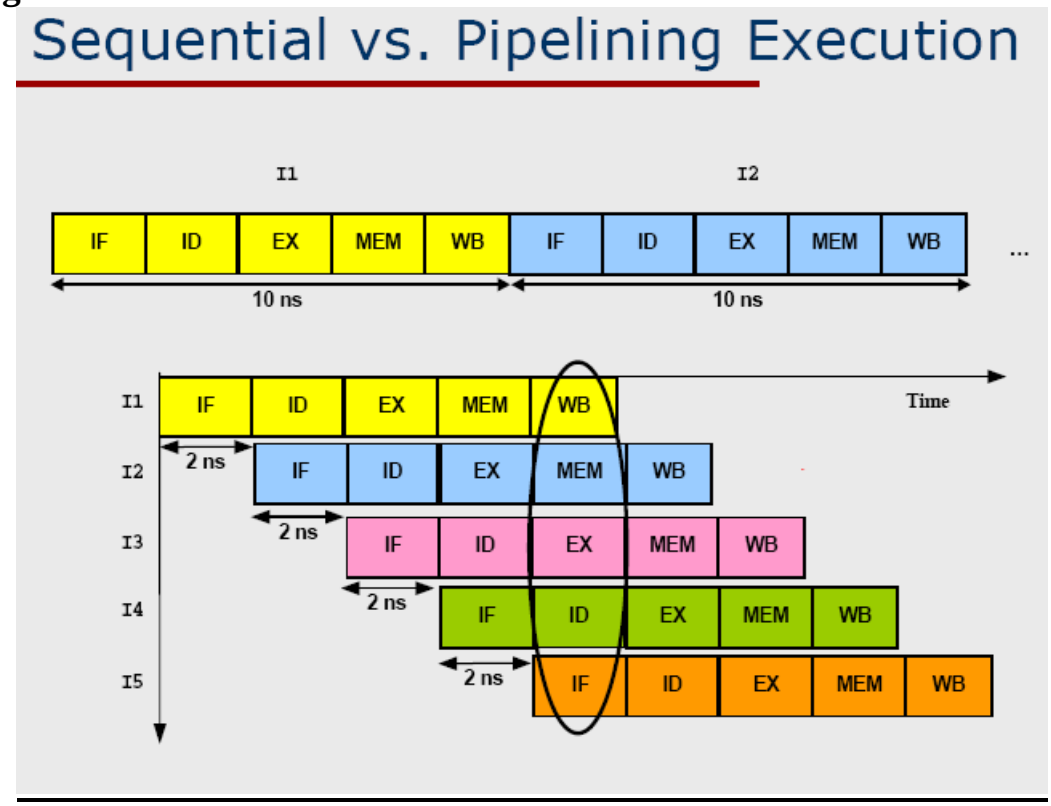
Instruction-level Parallelism

Basic five-stage pipeline in a RISC machine
IF = Instruction **Fetch**,
ID = Instruction **Decode**,
EX = **Execute**,
MEM = **Memory** access,
WB = **Register** write back

Sequential vs. Pipelining Execution

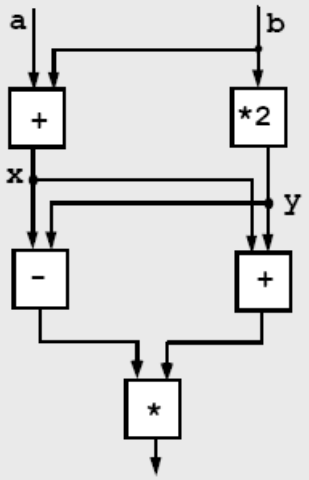


Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

In the fourth clock cycle (the green column), the earliest instruction is in MEM stage, and the latest instruction has not yet entered the pipeline.

Example parallelism

```
x = a + b;
y = b * 2
z =(x-y) * (x+y)
```



Hazards limit performance:

- **Structural:** Need more **Hardware** resources
- **Data:** Need **forwarding**, Compiler scheduling
- **Control:** Branch Delay Slot, Static and Dynamic **Branch Prediction**.

Data Hazards

- *Data dependence*

$r_3 \leftarrow r_1$ op r_2 Read-after-Write
$r_5 \leftarrow r_3$ op r_4 (RAW)

- *Anti-dependence*

$r_3 \leftarrow r_1$ op r_2 Write-after-Read
$r_1 \leftarrow r_4$ op r_5 (WAR)

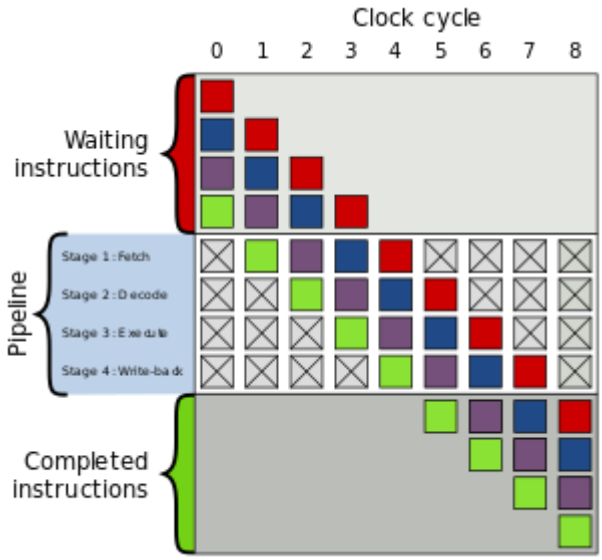
- *Output dependence*

$r_3 \leftarrow r_1$ op r_2 Write-after-Write
$r_5 \leftarrow r_3$ op r_4 (WAW)
$r_3 \leftarrow r_6$ op r_7

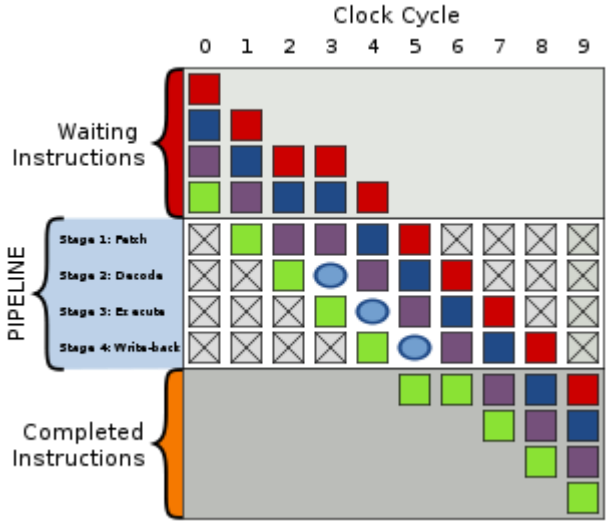
Dealing with data hazards

A bubble or pipeline stall is a delay in execution of an instruction in an instruction pipeline in order to resolve a hazard

The following is two executions of the same four instruction through a 4-stage pipeline but, for whatever reason, a delay in fetching of the purple instruction in cycle #2 leads to a bubble being created delaying all instructions after it as well.



Normal execution

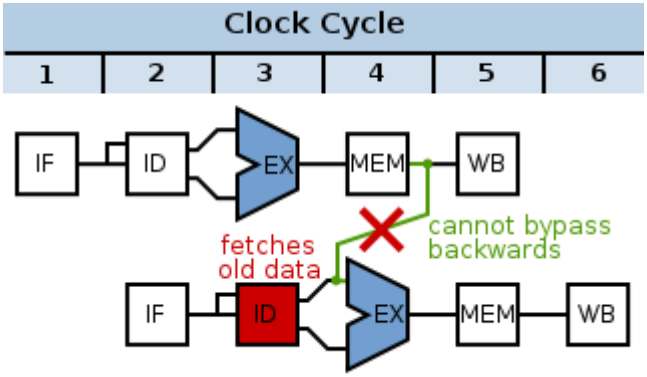


Execution with a bubble

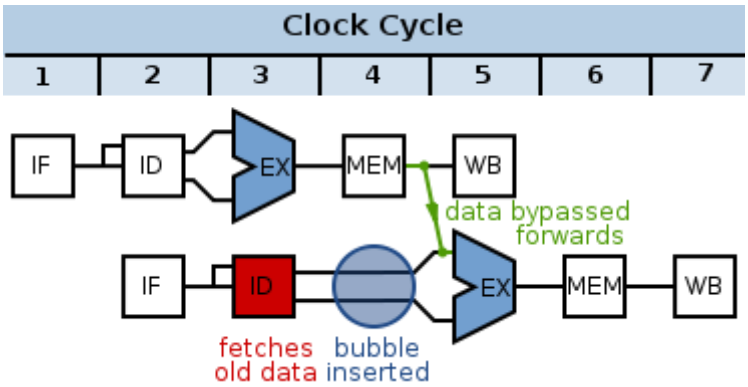
Classic RISC pipeline

The below example shows a bubble being inserted into a classic RISC pipeline, with five stages (IF = Instruction Fetch, ID = Instruction Decode, EX = Execute, MEM = Memory access, WB = Register write back). In this example, data available after the MEM stage (4th stage) of the first instruction is required as input by the EX stage (3rd stage) of the second instruction. Without a bubble, the EX stage (3rd stage) only has access to the output of the previous EX stage. Thus adding a bubble resolves the time dependence without needing to propagate data backwards in time (which is impossible).

Bypassing backwards in time



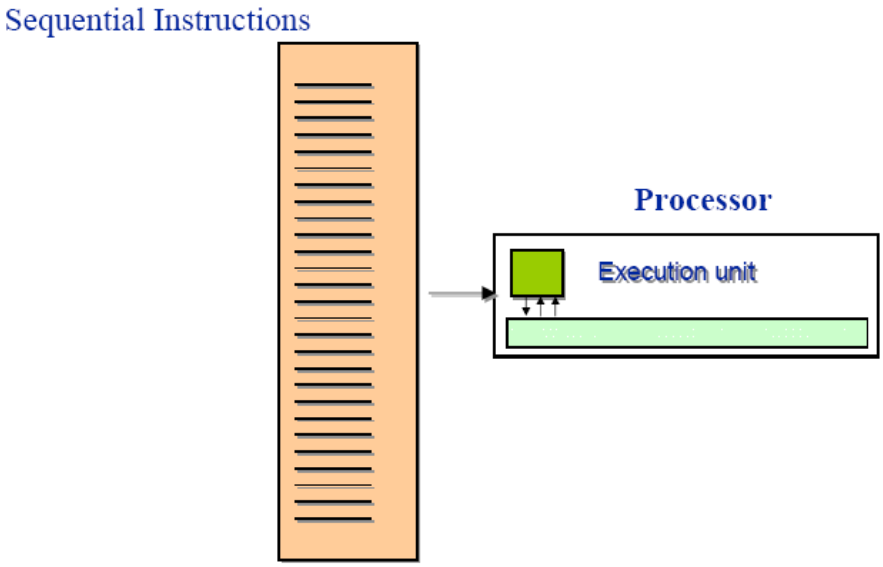
Problem resolved using a bubble



ILP Architecture Classifications

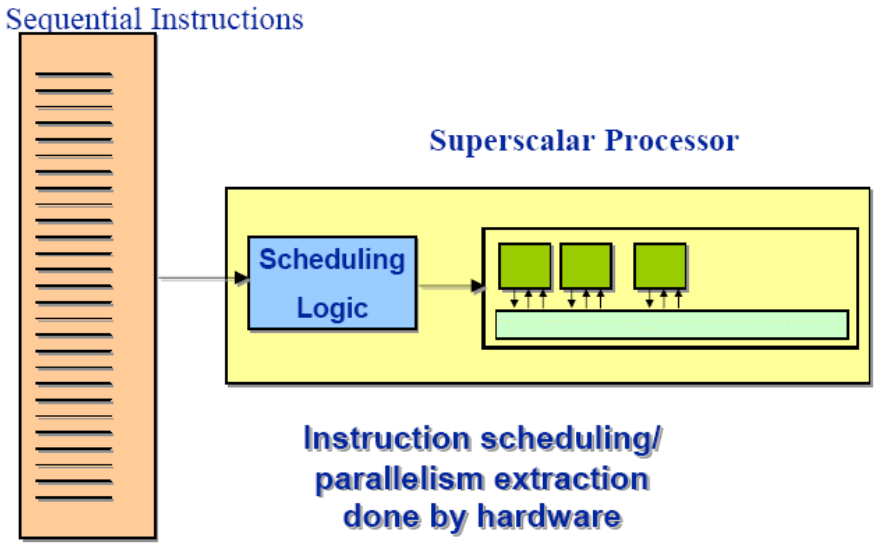
1-Sequential Architectures

– The program is not expected to convey any explicit information regarding parallelism



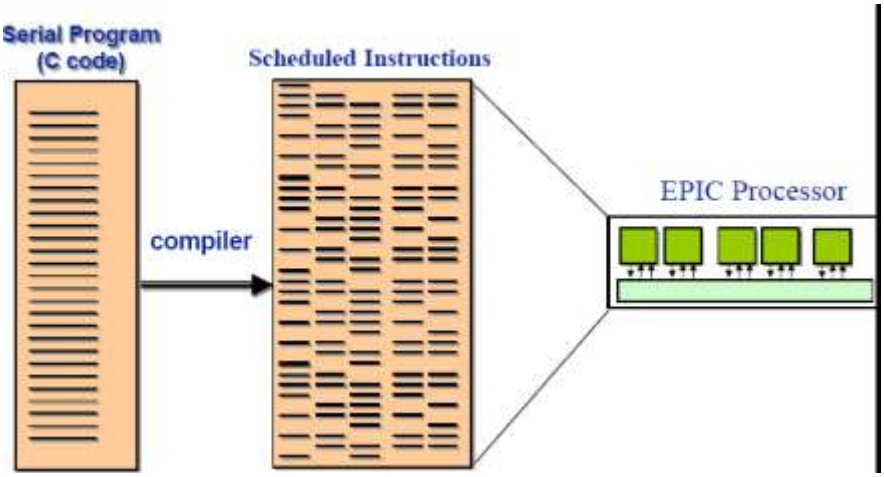
2- Dependence Architectures

– The program explicitly indicates dependencies between operations



3-Independence Architectures

– The program provides information as to which operations are independent of one another



VLIW/EPIC processors are examples of Independence architectures

- Specify exactly which functional unit each operation is executed on and when each operation is issued
- Operations are independent of other operations issued at the same time as well as those that are in execution
- Compiler emulates at compile