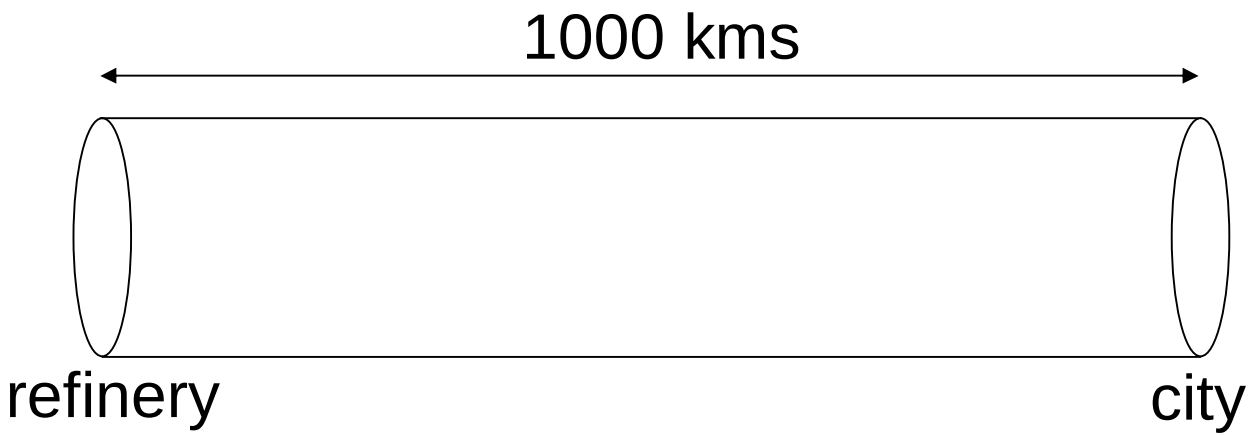
High Performance Computing
Lecture 22

Matthew Jacob

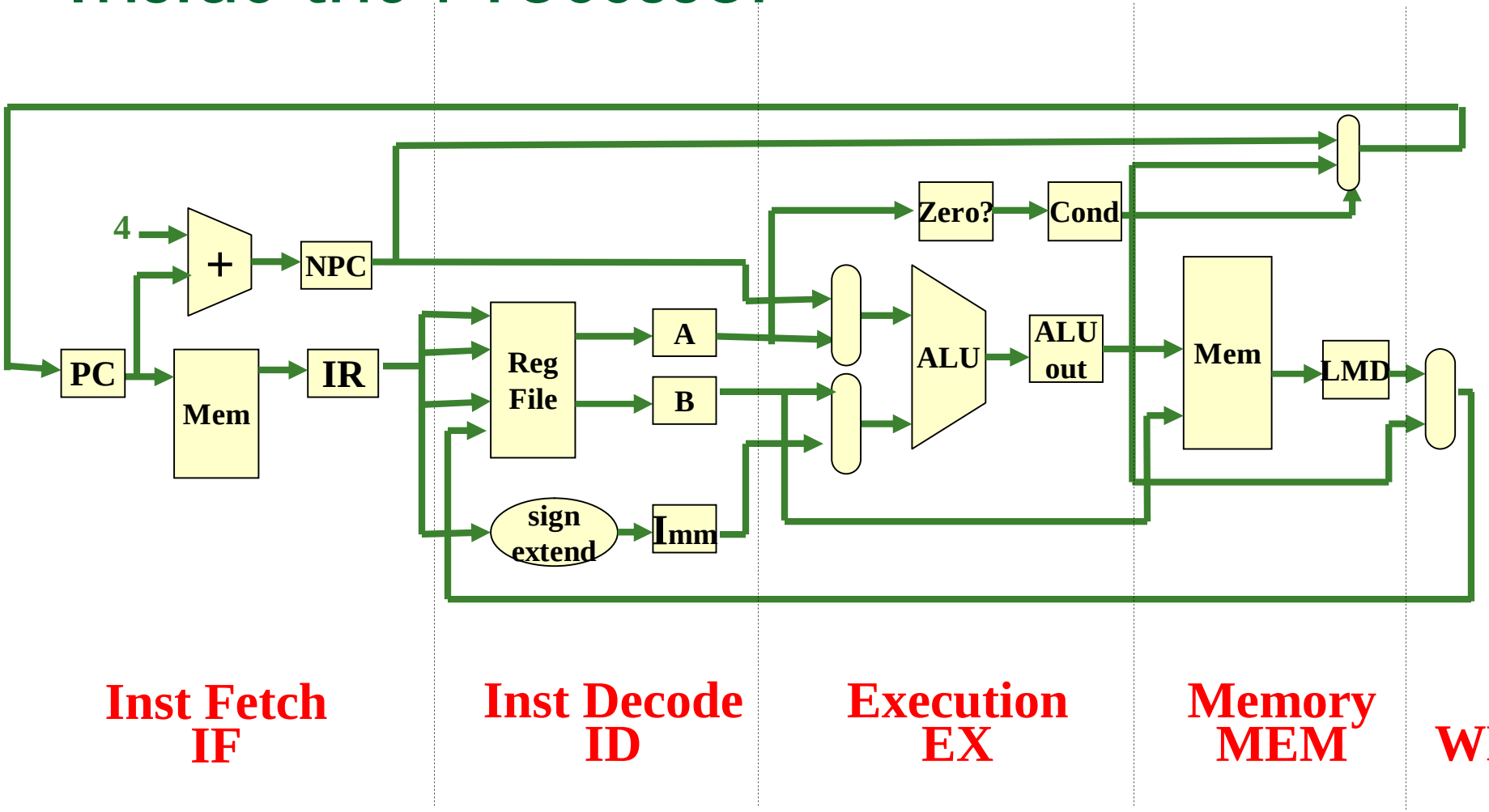
Indian Institute of Science

Pipelines

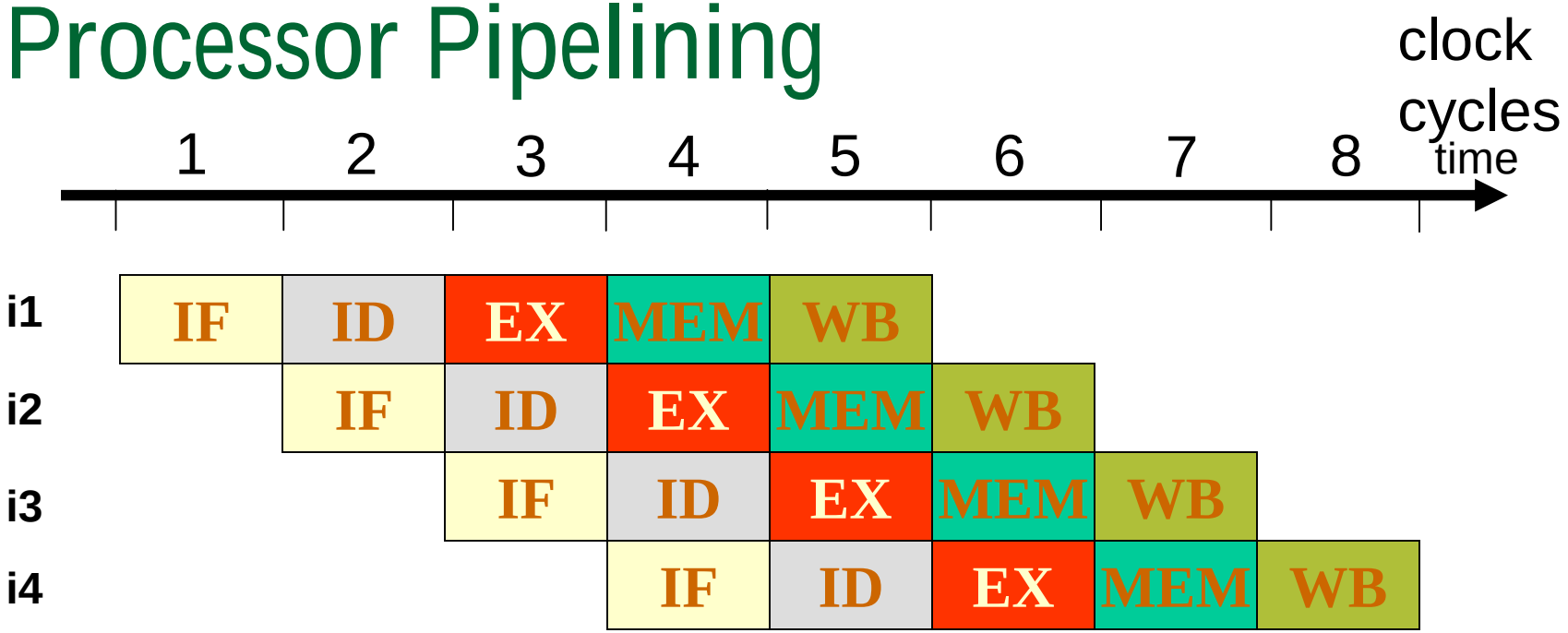
- Used for transportation of liquids or gases over long distances
  - 1000s of kms
  - Built with periodic pump/compressor stations to keep the fluid flowing



Inside the Processor

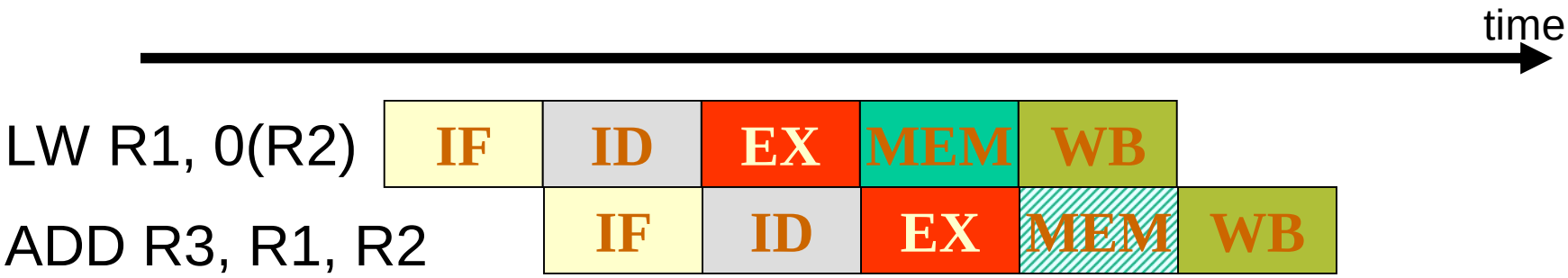


Processor Pipelining

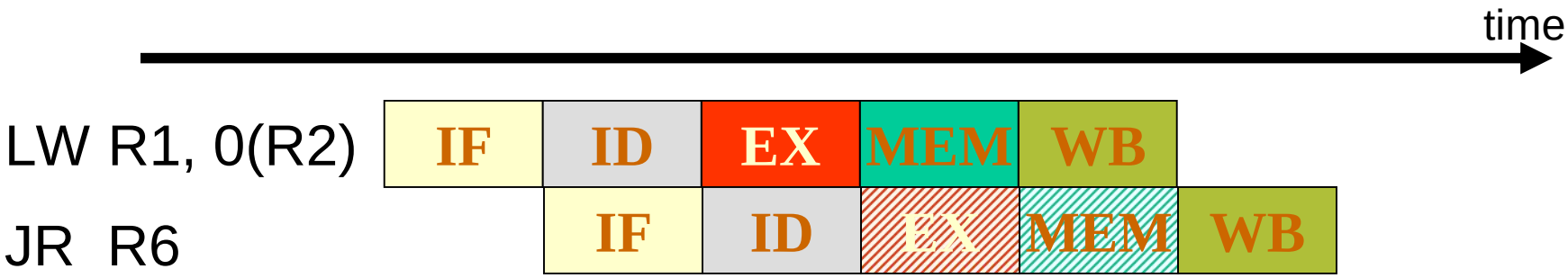


- Execution time of each instruction is still 5 cycles, but the throughput is now 1 instruction per cycle
- Initial pipeline fill time (4 cycles), after which 1 instruction completes every cycle

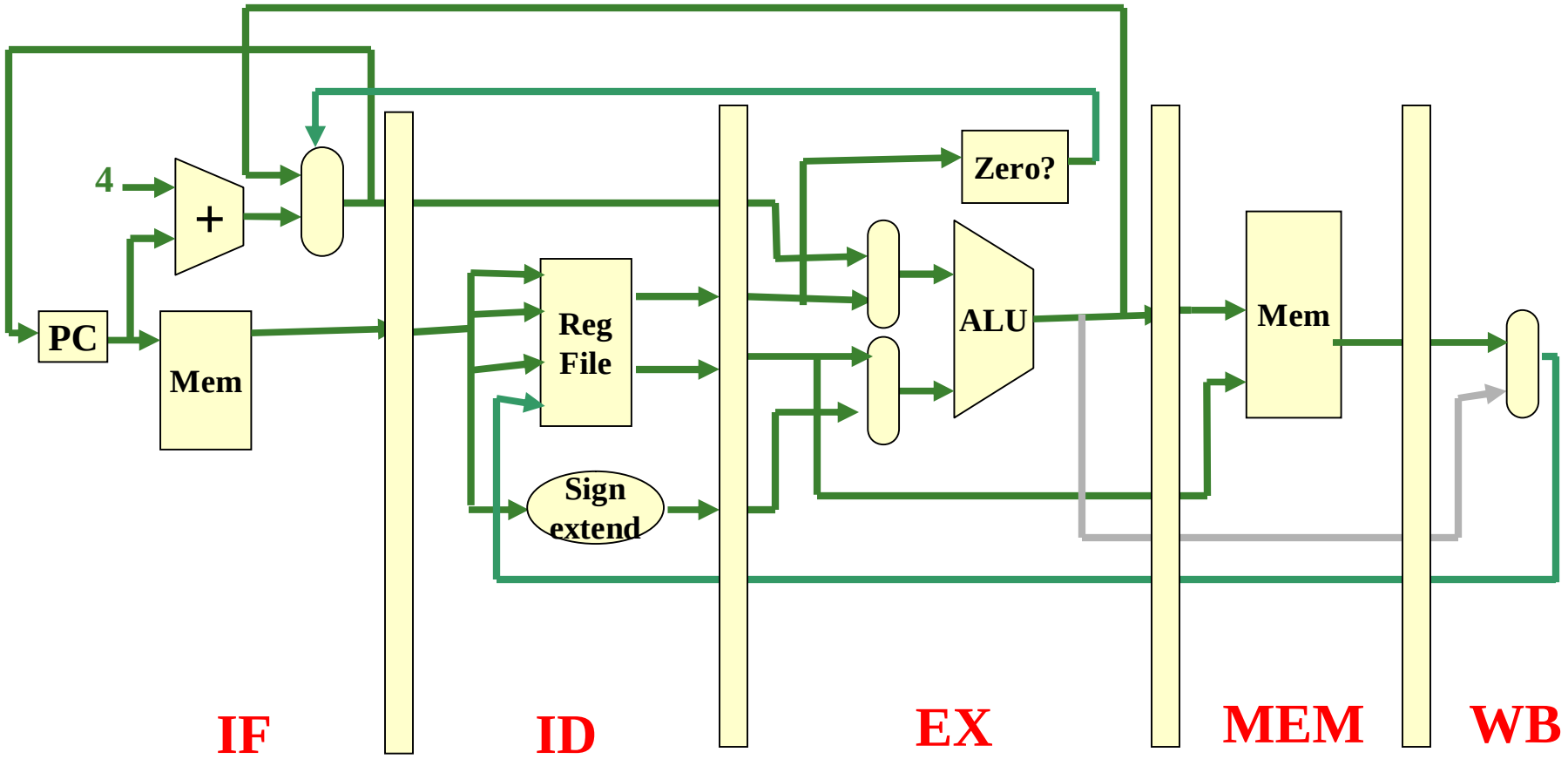
MIPS 1 Instructions: 3, 4 or 5 cycles



MIPS 1 Instructions: 3, 4 or 5 cycles



Pipelined Processor Datapath



Some Terminology

- Pipeline stages: IF, ID, EX, MEM, WB
- We describe this as a 5 stage pipeline
  - or a pipeline of depth 5
- Assume that the time delay through each stage is the same (say 1 clock cycle)

- Pipeline Speedup = $\frac{time_{non-pipelined}}{time_{pipelined}}$

Pipeline Speedup

- For 5 stage pipeline taking 1 cycle per stage
  - Let us compute the speedup over a non-pipelined processor that takes 5 cycles for every instruction
  - Calculate how much time each of these processors takes to run a program involving the execution of n instructions
    - Non-pipelined processor: $5n$ cycles
    - Pipelined processor: $4 + n$ cycles
  - Speedup = $\dfrac{5n}{4+n} \rightarrow 5 \quad \text{as} \quad n \rightarrow \infty$

Pipeline Speedup

- A pipeline with p stages could give a speedup of p (compared to a non-pipelined processor that takes p cycles for each instruction)
- i.e., A program would run p times faster on the pipelined processor (than on the non-pipelined processor)
  - if on every clock cycle, an instruction completes execution

Problem: Pipeline Hazards

A situation where an instruction cannot proceed through the pipeline as it should

Hazard: a dangerous (hazardous) situation

From the perspective of correct program execution

Problem: Pipeline Hazards

A situation where an instruction cannot proceed through the pipeline as it should

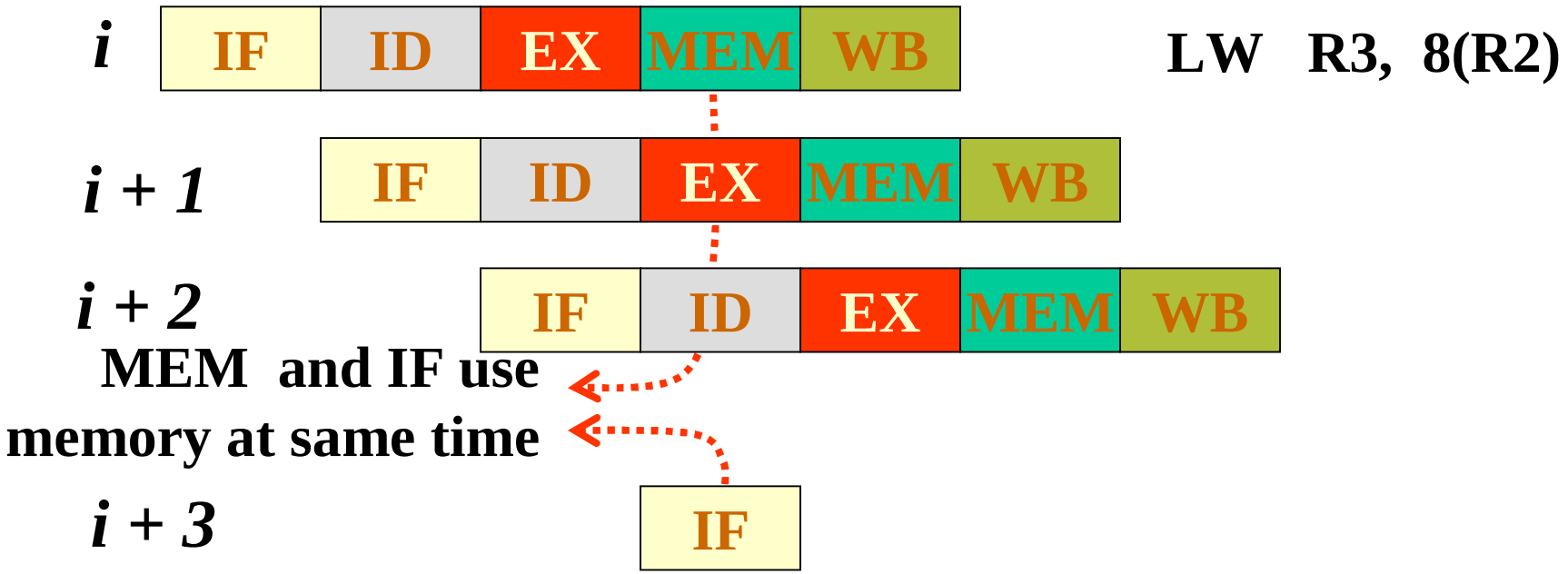
1. Structural hazard: When 2 or more instructions in the pipeline need to use the same resource at the same time
2. Data hazard: When an instruction depends on the data result of a prior instruction that is still in the pipeline
3. Control hazard: A hazard that arises due to control transfer instructions

Problem: Pipeline Hazards

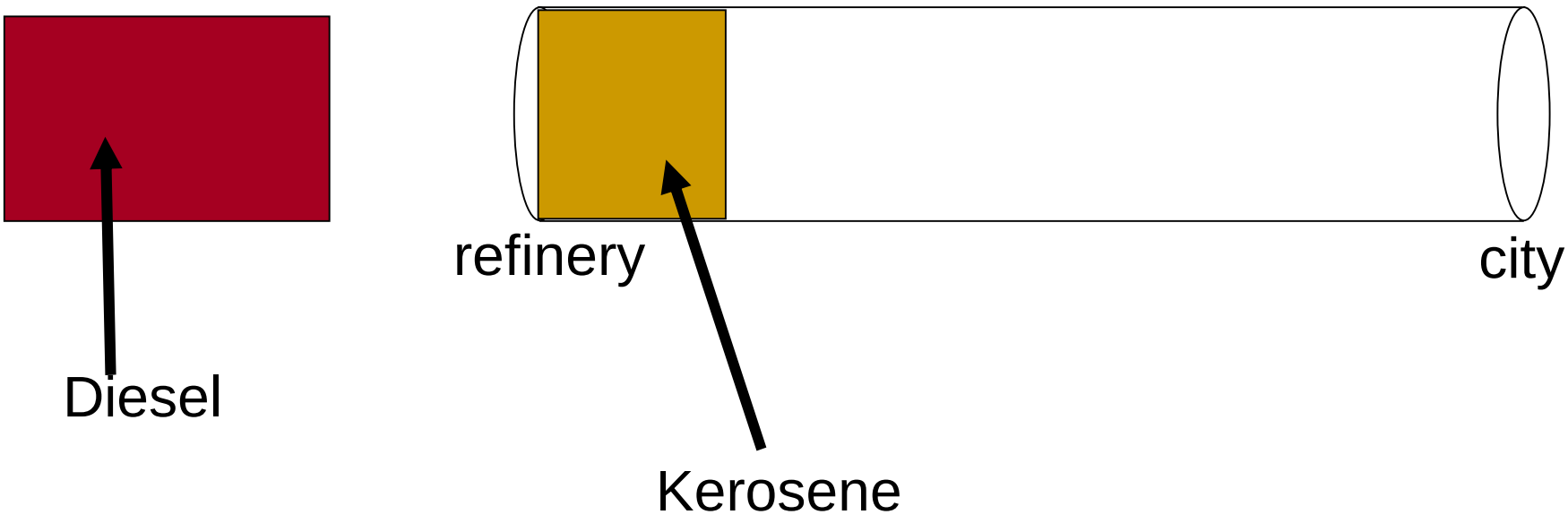
A situation where an instruction cannot proceed through the pipeline as it should

1. Structural hazard: When 2 or more instructions in the pipeline need to use the same resource at the same time
2. Data hazard: When an instruction depends on the data result of a prior instruction that is still in the pipeline
3. Control hazard: A hazard that arises due to control transfer instructions

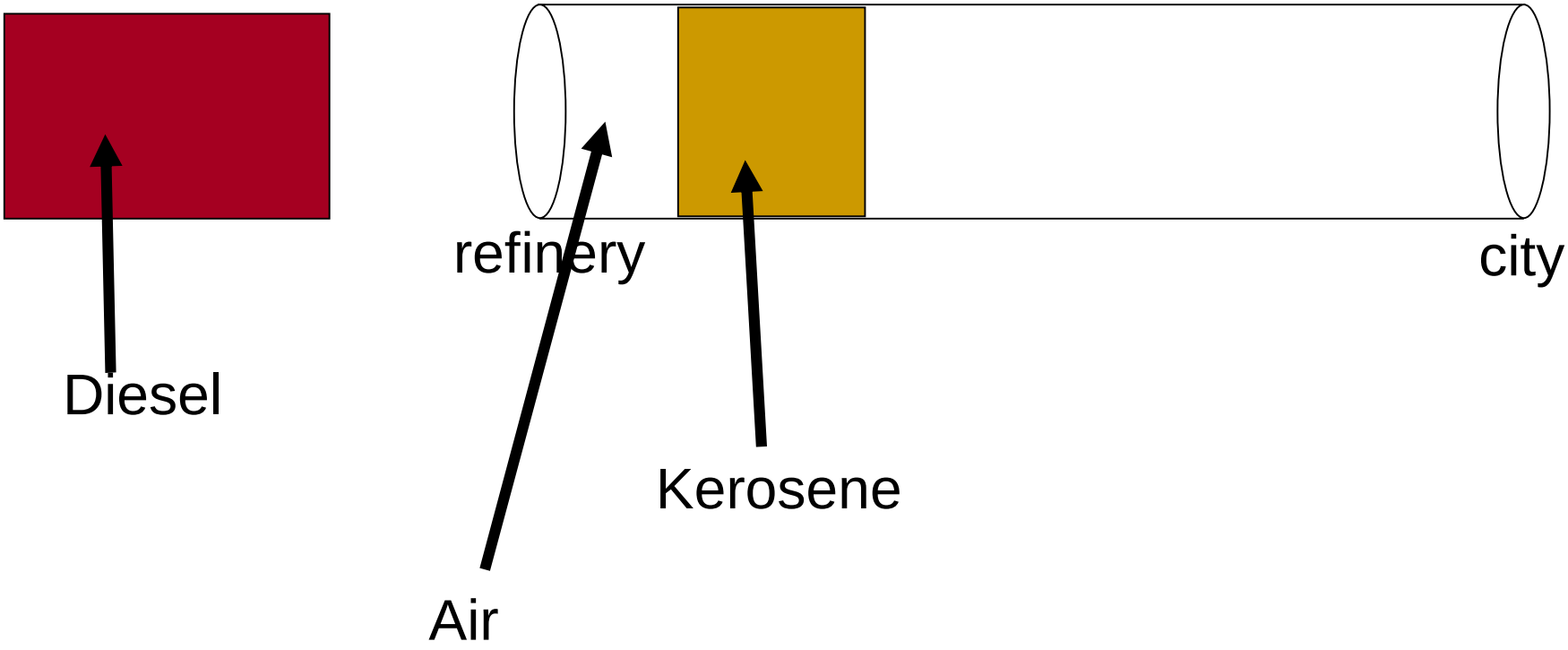
Structural Hazard



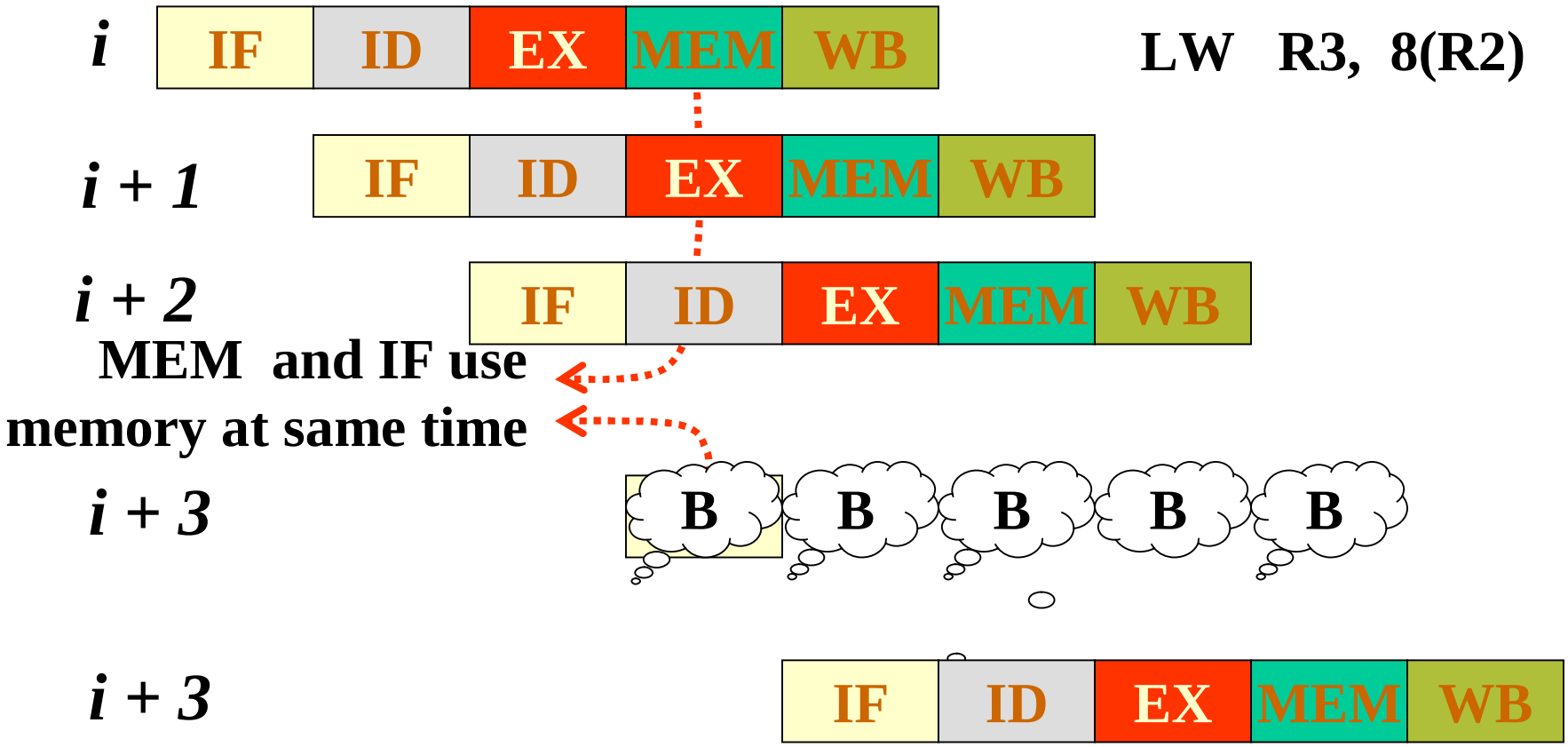
Petroleum pipeline analogy?



Petroleum pipeline analogy?



Structural Hazard



Solving Structural Hazards

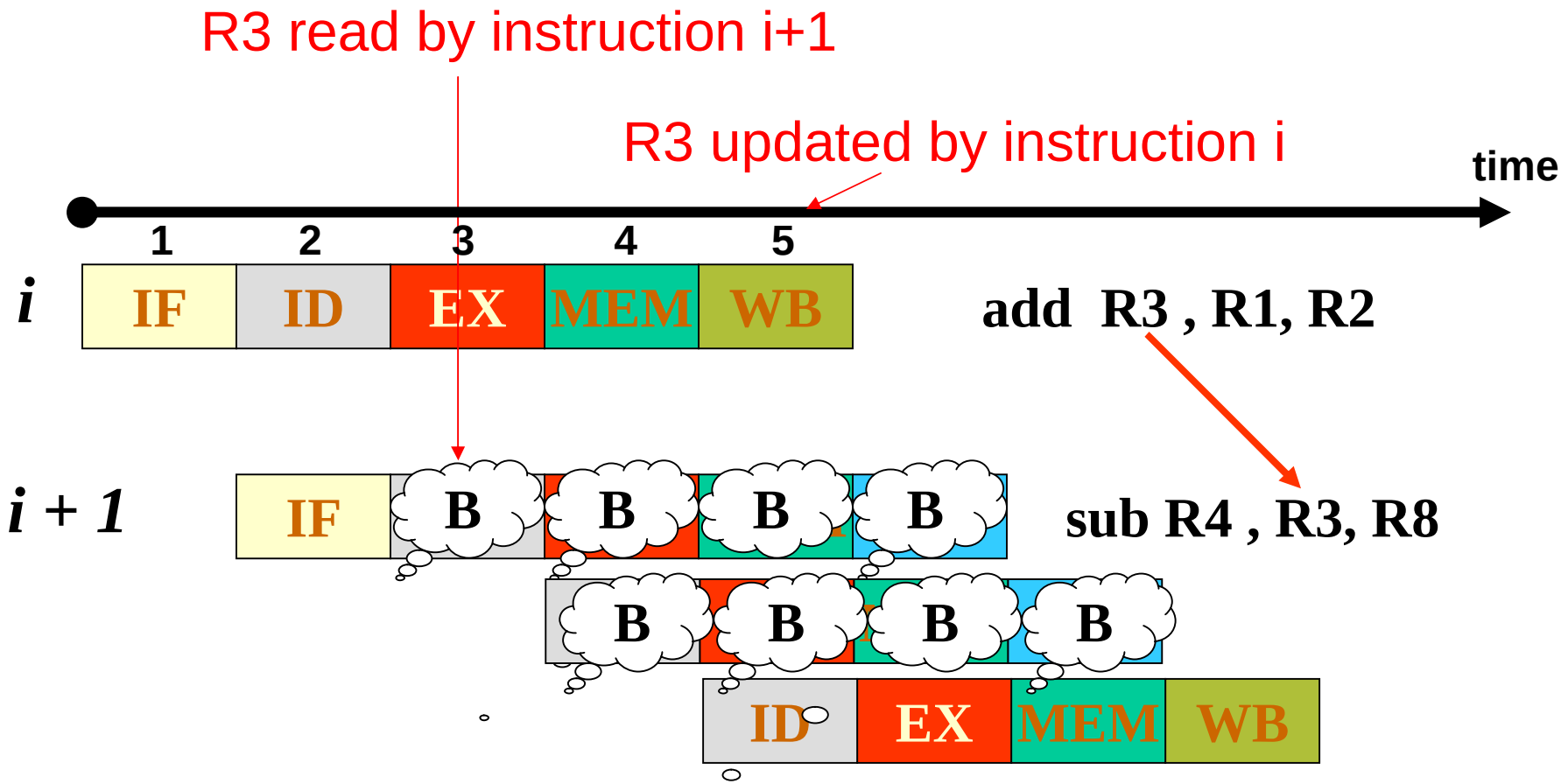
- This hazard can be overcome by designing main memory so that it can handle 2 memory requests at the same time
  - Double ported memory
- Or, since we are assuming that memory delays are hidden by cache memories…
  - include a separate instruction cache (for use by the IF pipeline stage) and data cache (for use by the MEM stage)
- Identify the possible structural hazards and design so as to eliminate them

Problem: Pipeline Hazards

A situation where an instruction cannot proceed through the pipeline as it should

1. Structural hazard: When 2 or more instructions in the pipeline need to use the same resource at the same time
2. Data hazard: When an instruction depends on the data result of a prior instruction that is still in the pipeline
3. Control hazard: A hazard that arises due to control transfer instructions

Data Hazard



Idea: Delay (or stall) the progress of instruction i+1 through the pipeline until the data is available in register R3