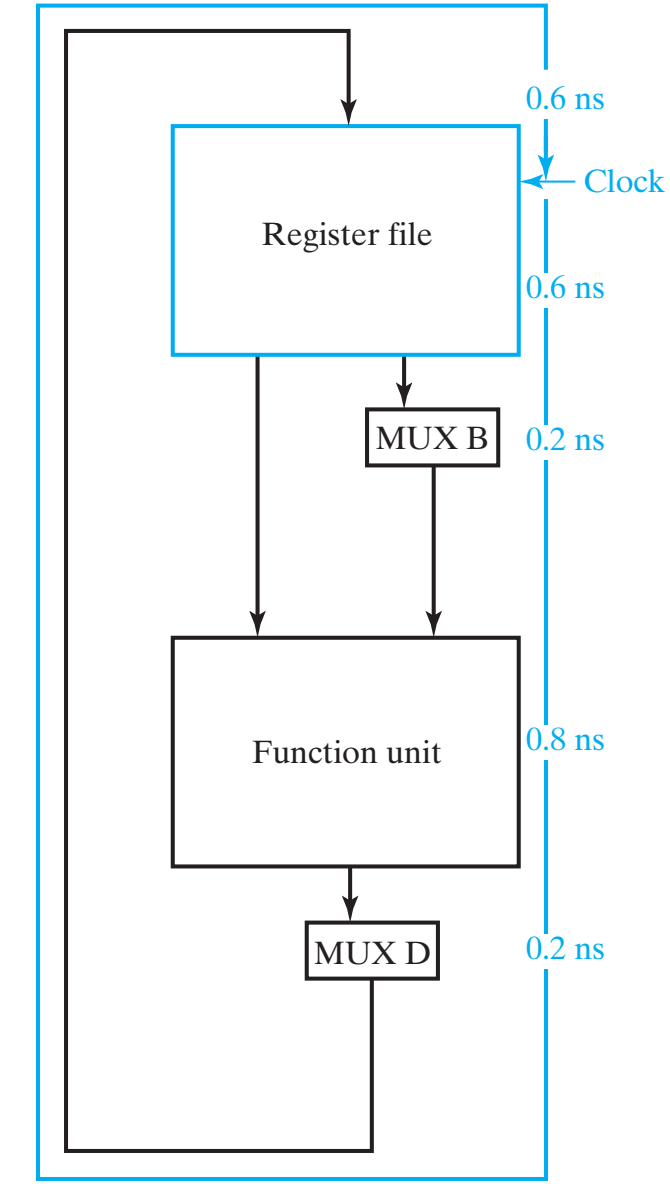
Computer Hardware

Pipeline

Conventional Datapath

- 2.4 ns is required to perform a single operation (i.e. 416.7 MHz).

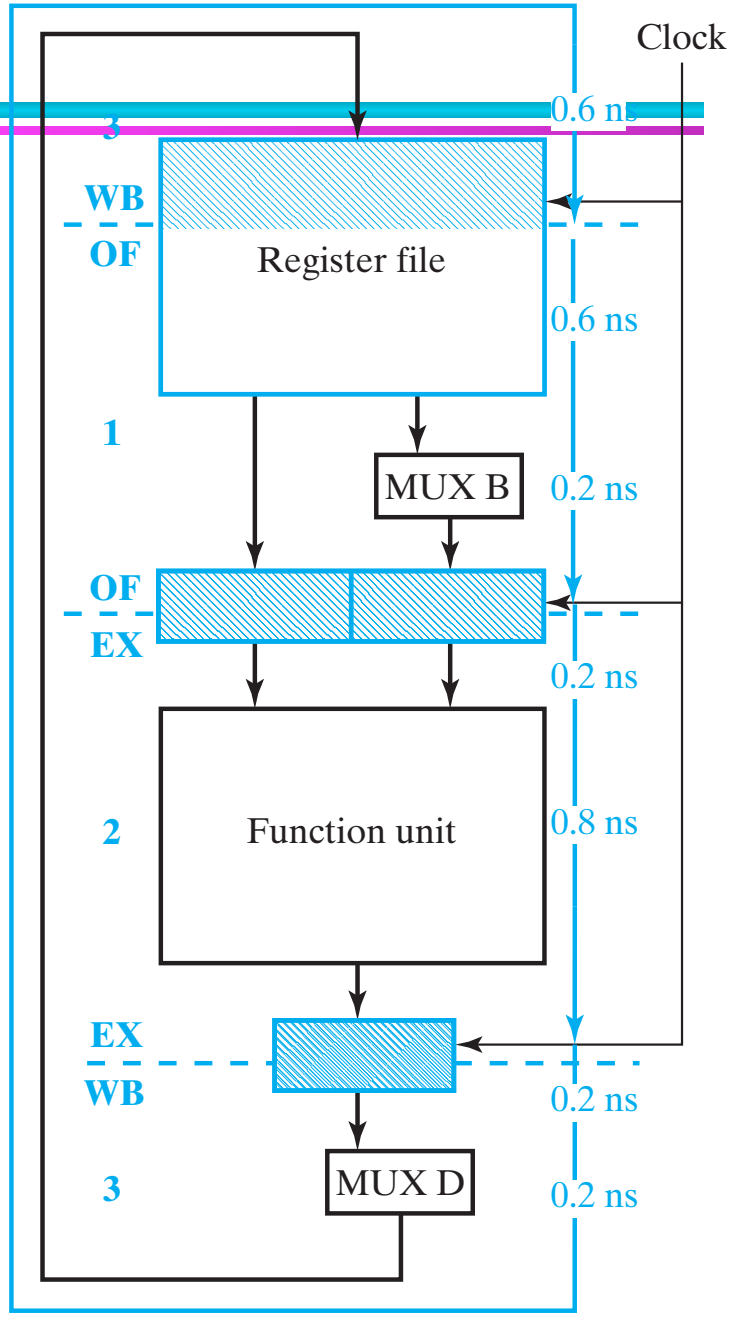


Production Line Analogy

- Automated car wash: Cars are pulled through a series of stations at which a particular step if performed:
    1. Wash
    2. Rinse
    3. Dry
- Think of *latency time* = time needed to wash, rinse and dry. Think of rate of delivery of washed cars or *throughput*
- *Based on this analogy →pipelined datapaths with n-stages have a processing rate or throughput for instructions that is n times that of non-pipelined datapaths.*

Pipelined Datapath

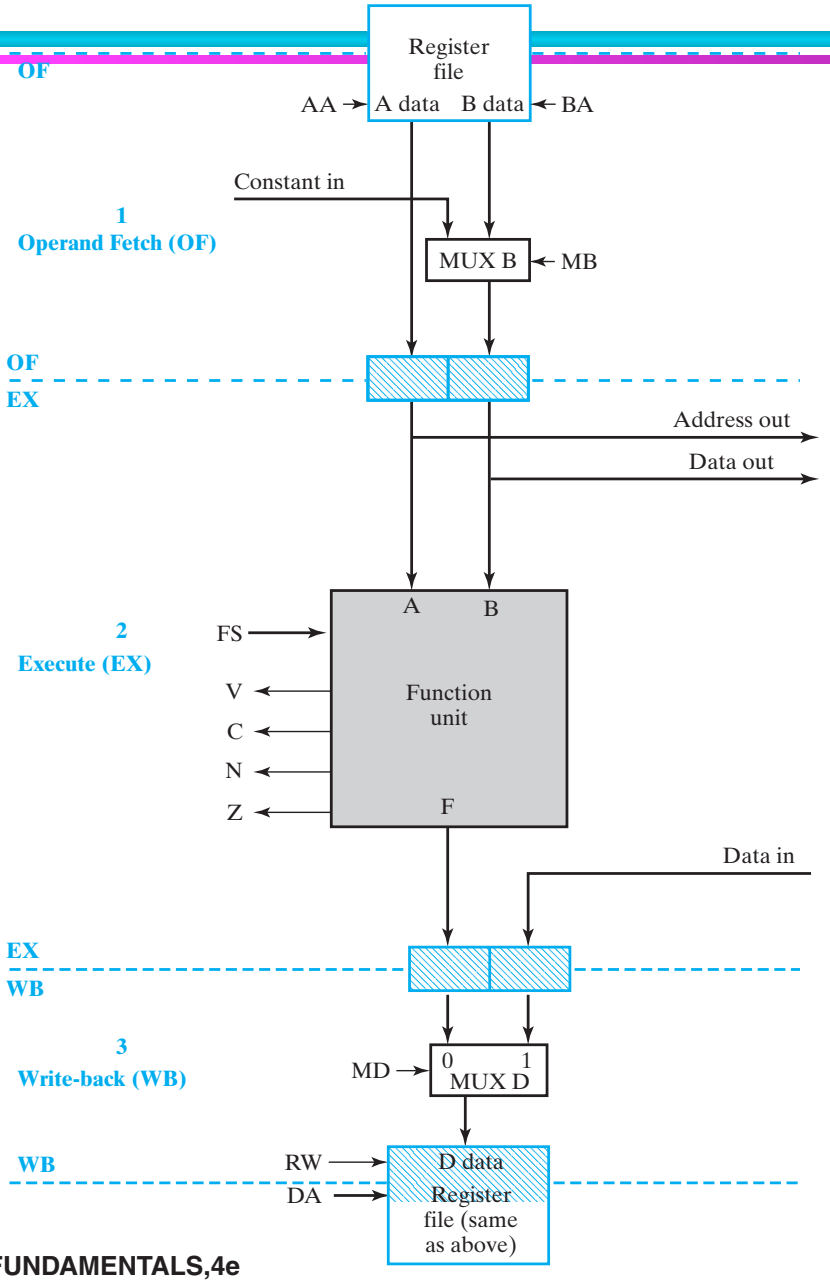
- A Pipelined Datapath is done by breaking a conventional datapath into parts by inserting registers as *pipeline platforms* between these parts
- These registers provide temporary storage for data passing through the pipeline
- Data moves synchronously with the clock
- Delay of operand fetch (OF) is 0.8 ns, delay of execution (EX) is 1.0 ns, delay of write-back (WB) is 1.0 ns
- min clock period = 1.0 ns
- Operating frequency= 1.0 Ghz MHz (2.4 times that of the non-pipelined.)
- Even though there are 3 stages, the improvement factor is not quite 3. Why?



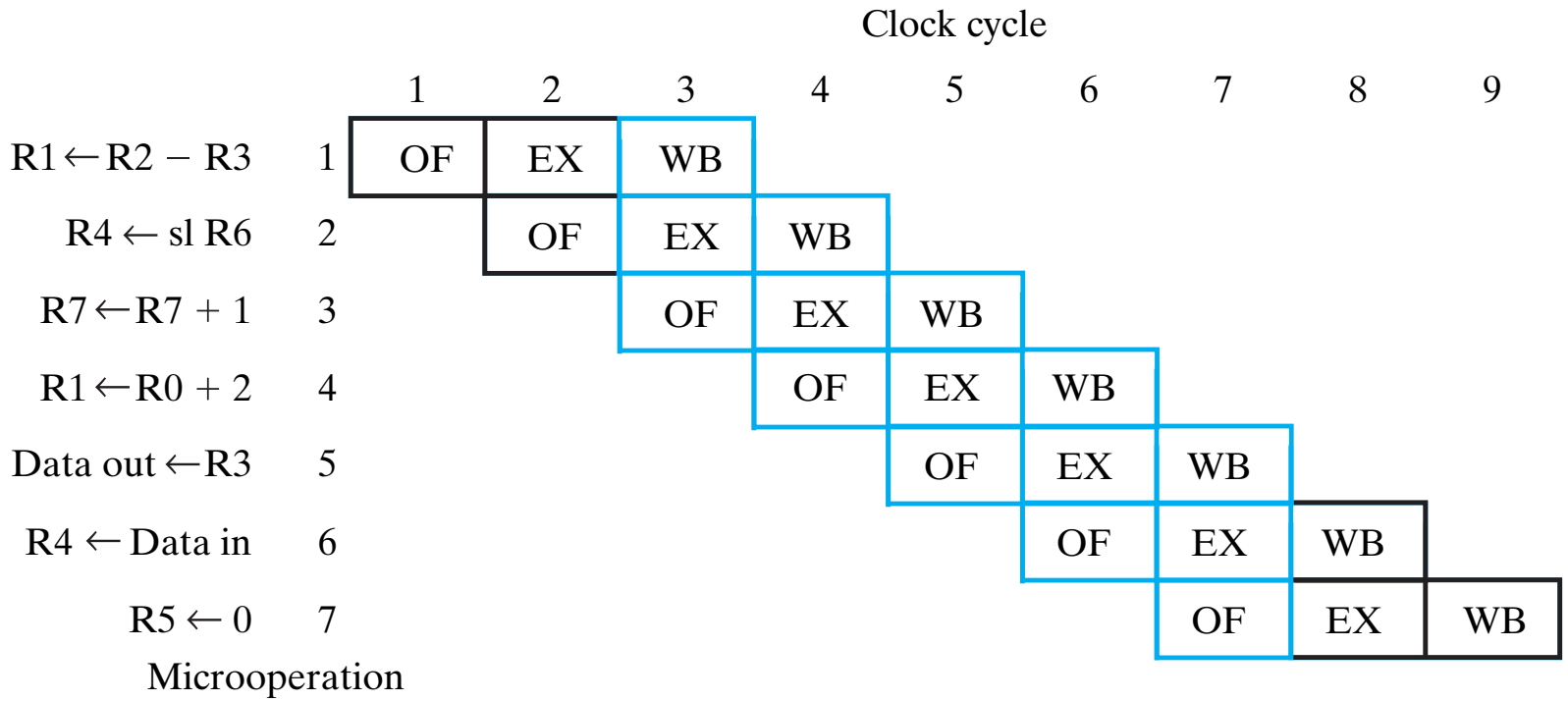
(b) Pipelined

Pipelined Datapath

- OF consists of reading register values (A&B), or selecting constant value (MB). The pipeline platform stores the operand(s) to be used in EX during next clock cycle
- In EX a function unit operation occurs, and the results captured by the 2nd pipeline platform
- WB is the write-back stage: the result is saved from the EX stage or the value on Data in (selected by MUX D).



Pipelined Execution Pattern



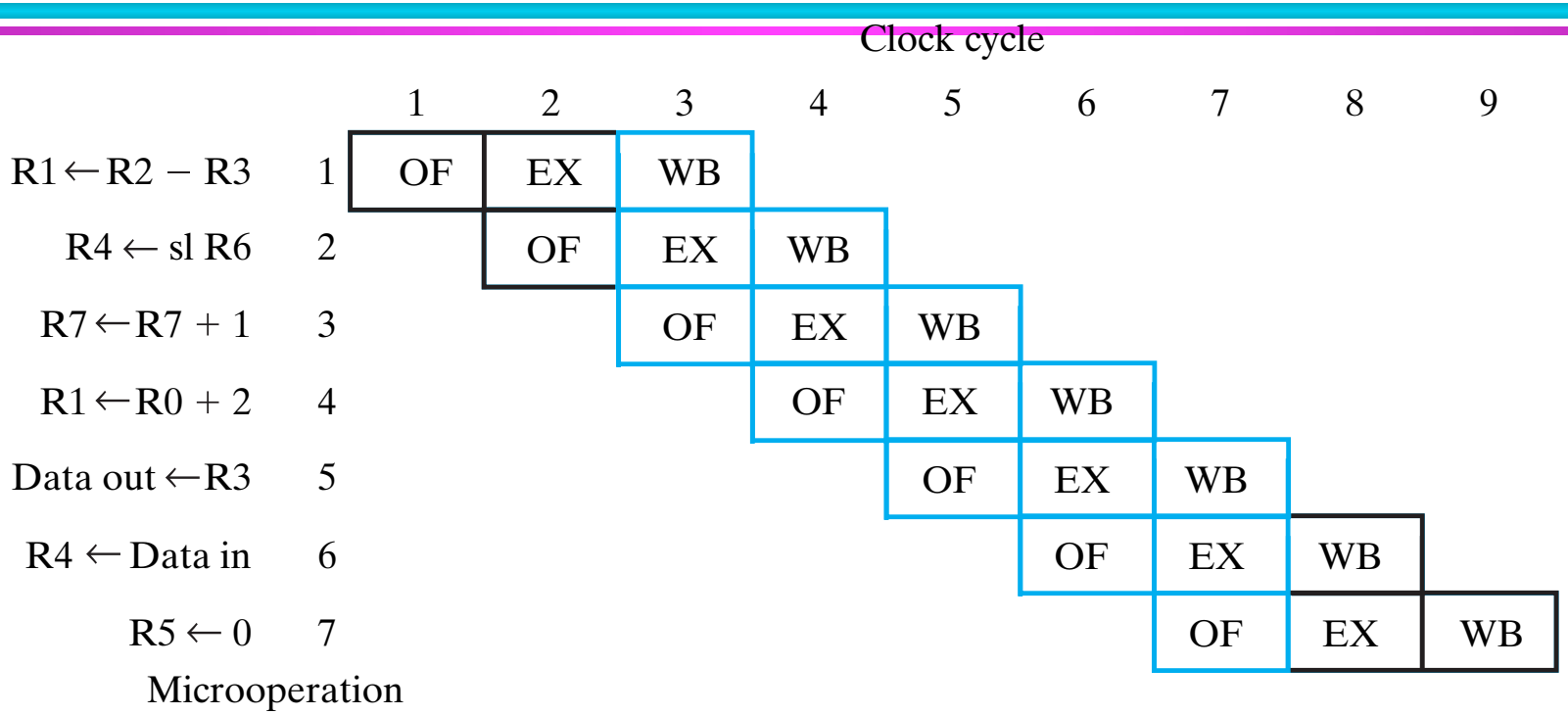
- What is total time required by conventional datapath for execution?

  → 7 (microoperation)× 2.4 (ns) = 16.8 ns

- What is total time required by pipelined datapath for execution?

  → (9 cycles)× 1 = 9 ns

Pipelined Execution Pattern

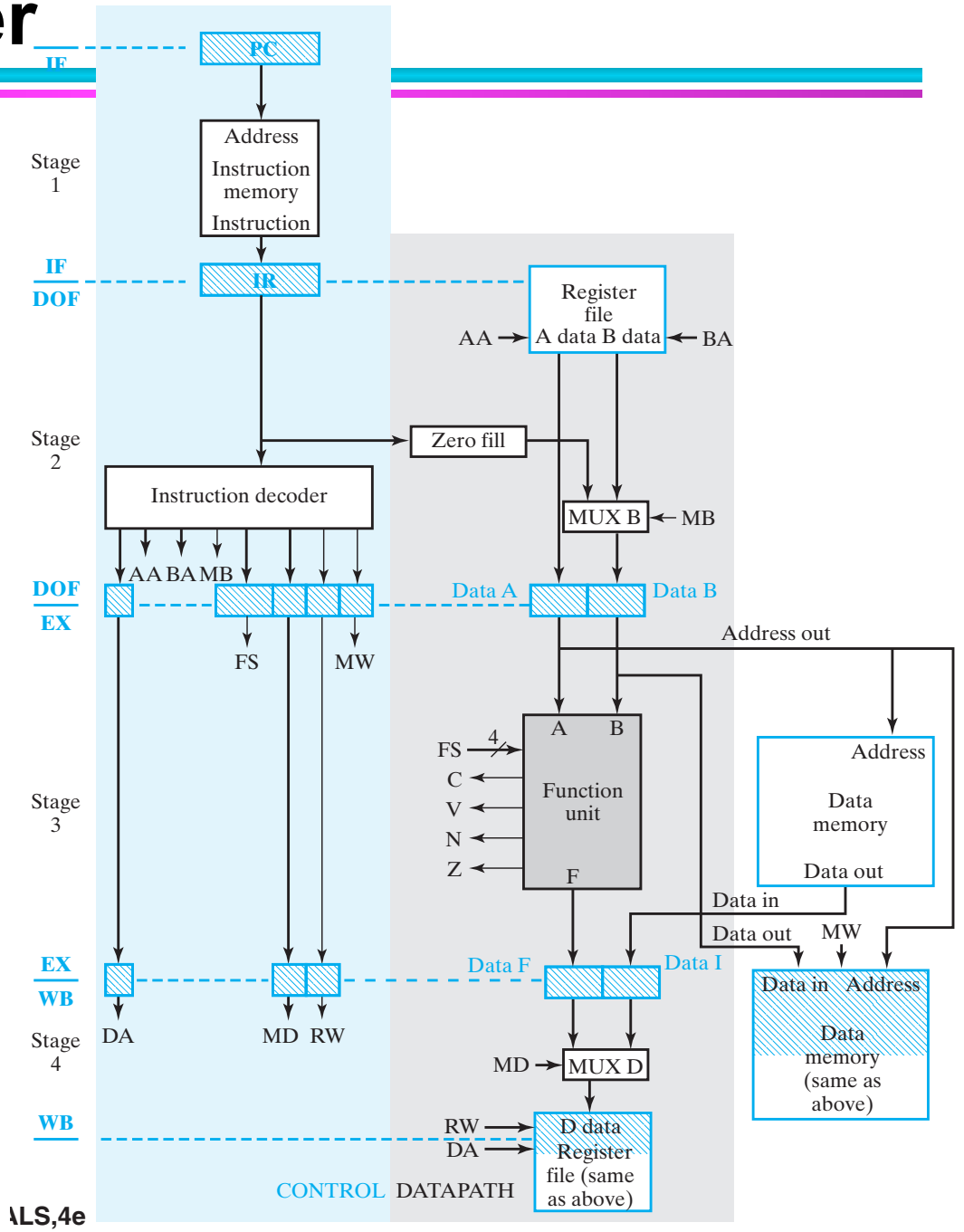


- Maximum improvement of pipelined over conventional can be obtained when the pipeline if fully utilized (all stages are active) e.g. over the 5 clock cycles, 3 to 7 (the pipeline is full), 5 operations are completed in 5 ns. While in the same time the conventional can execute 5ns ÷2.6 ns/ microoperation = 2.083 microoperations

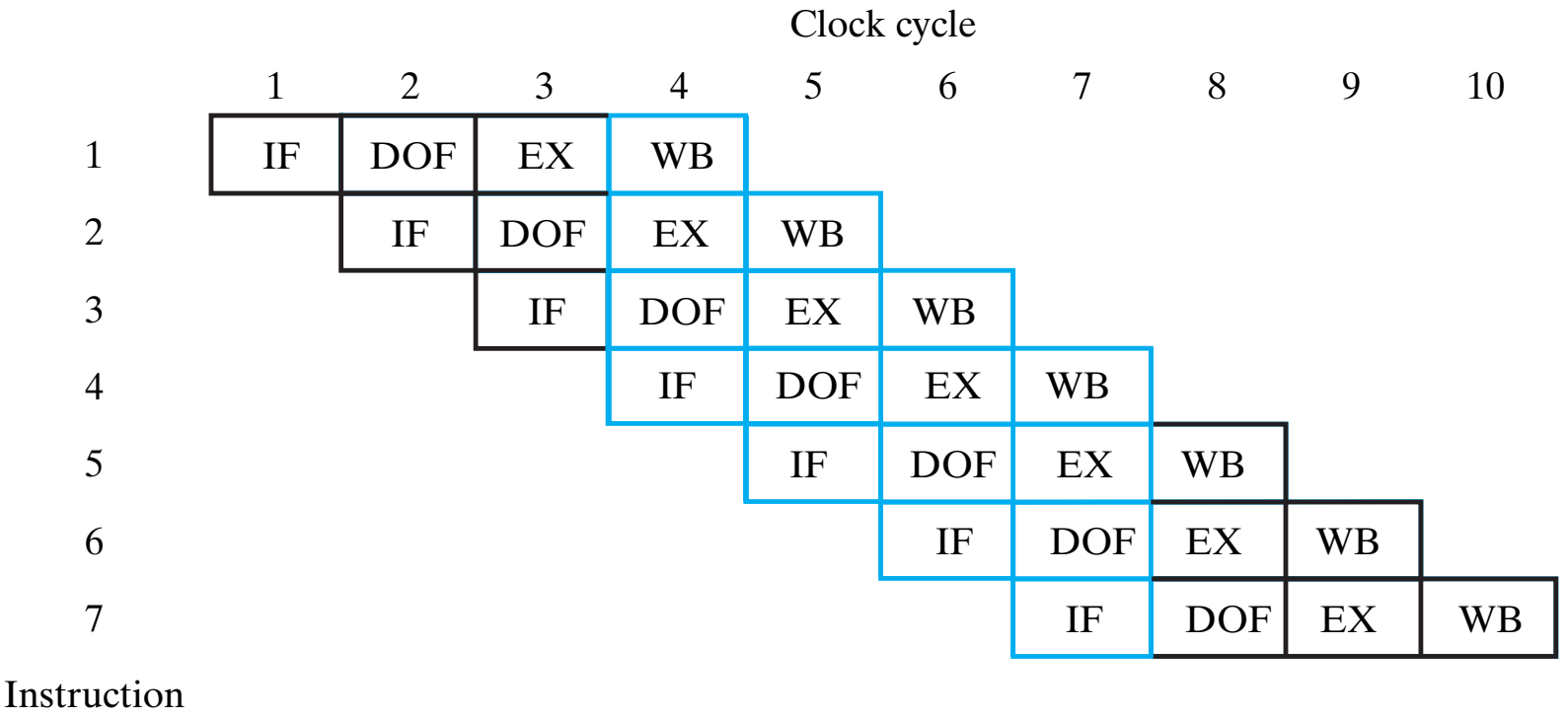
  → the pipelined executes 5 ÷ 2.083 = 2.4 times as many microoperations as conventional

Pipelined Computer

Registers are added to the pipeline platforms to pass the instruction information through the pipeline.



Performance of Pipelined Computer



- Compare the performance of the single-cycle computer with the performance of the pipelined computer (Compare for the situation in which the pipeline is fully utilized.)
- 4 instructions versus 20ns/17ns/inst. or 1.18 instructions Throughput Pipelined = 3.4x Single-Cycle

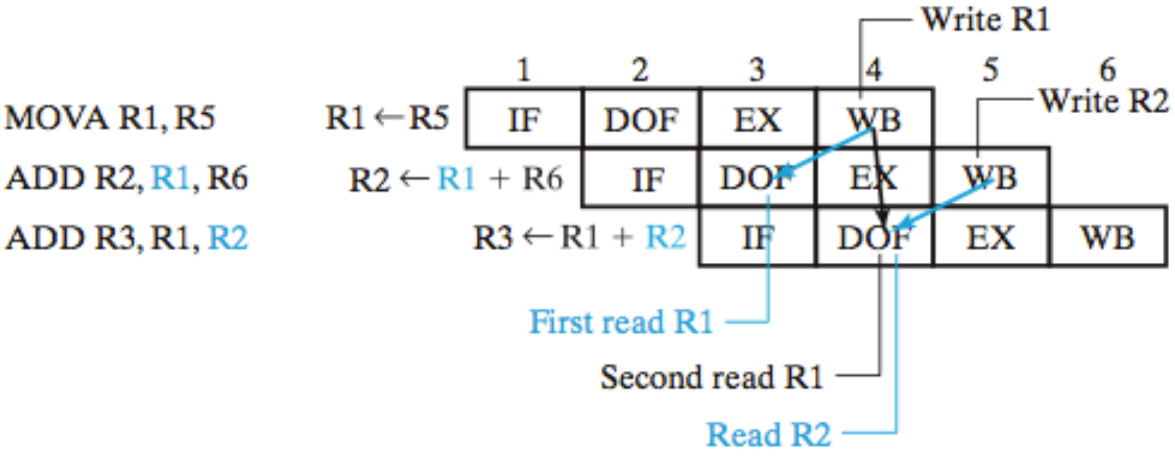
Performance Issues

- If a pipeline has 4 stages
  - performance is improved 4 times! Why?
- **Pipelining Hazards** cause the pipe to stall because of some conflict in the pipe (prevents the next instruction in pipe from executing in its turn)
- Types of hazards
  - Structural: contention for same hardware resource
  - Data: dependency on earlier instruction for the correct sequencing of register reads and writes
  - Control: branch/jump instructions stall the pipe until get correct target address into PC

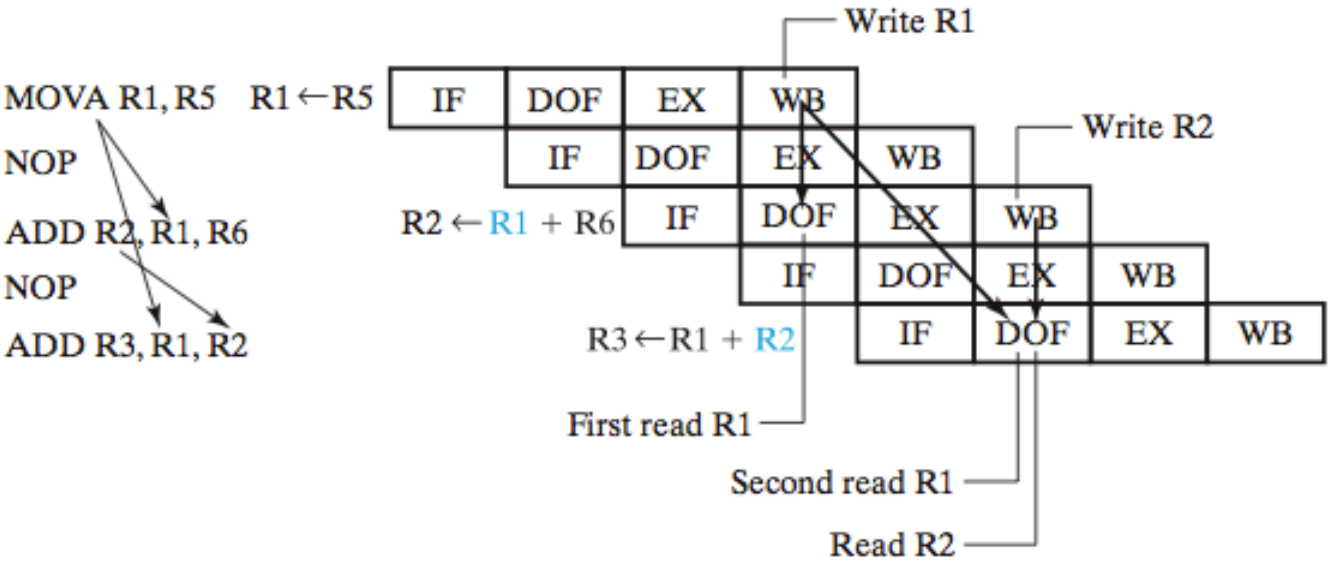
Reduction in Throughput

- Filling and flushing of the pipeline reduces the throughput executed below the maximum level.
- Data and control hazards are timing problems that arise because the execution of an operation in a pipeline is delayed by one or more clock cycles from the time at which the instruction containing the operation was fetched.

Data Hazard Problem

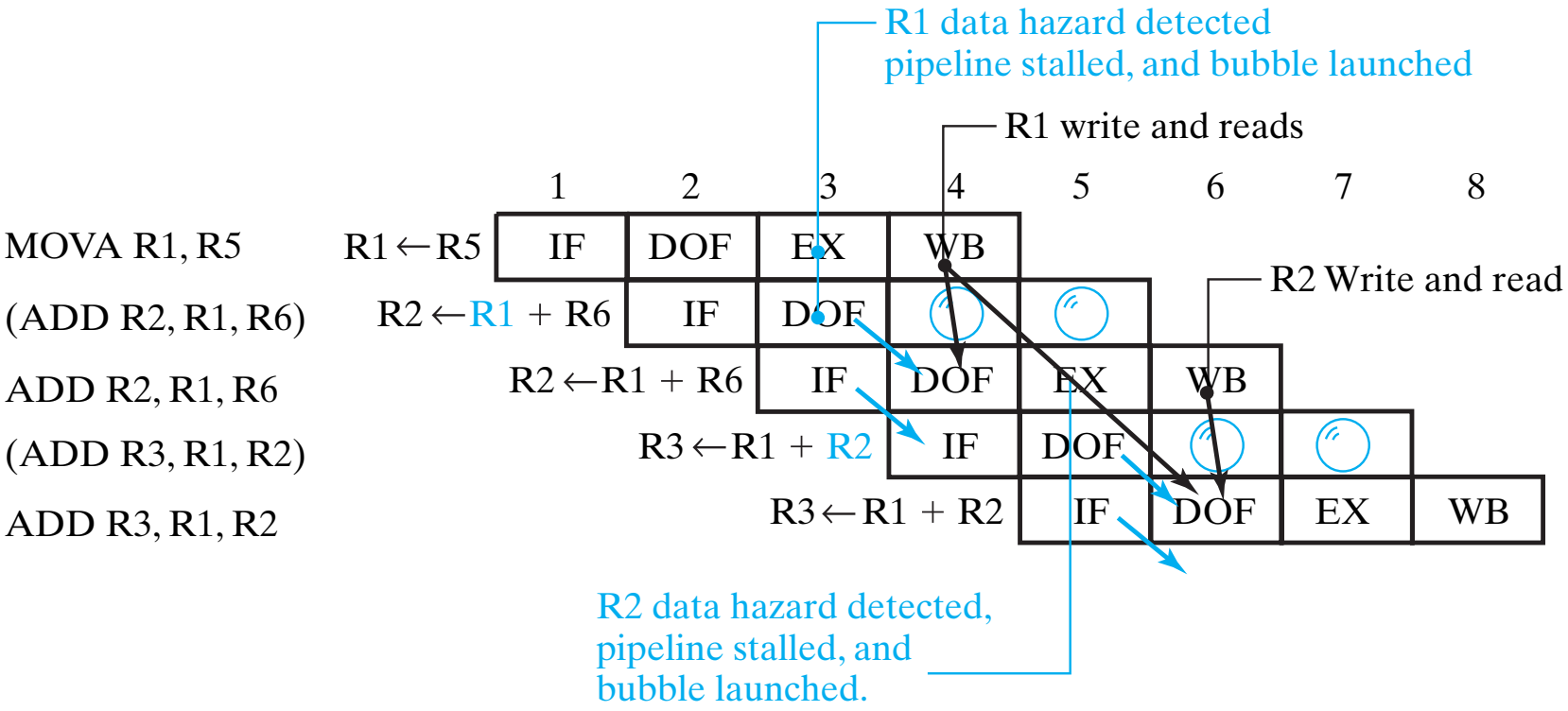


(a) The data-hazard problem

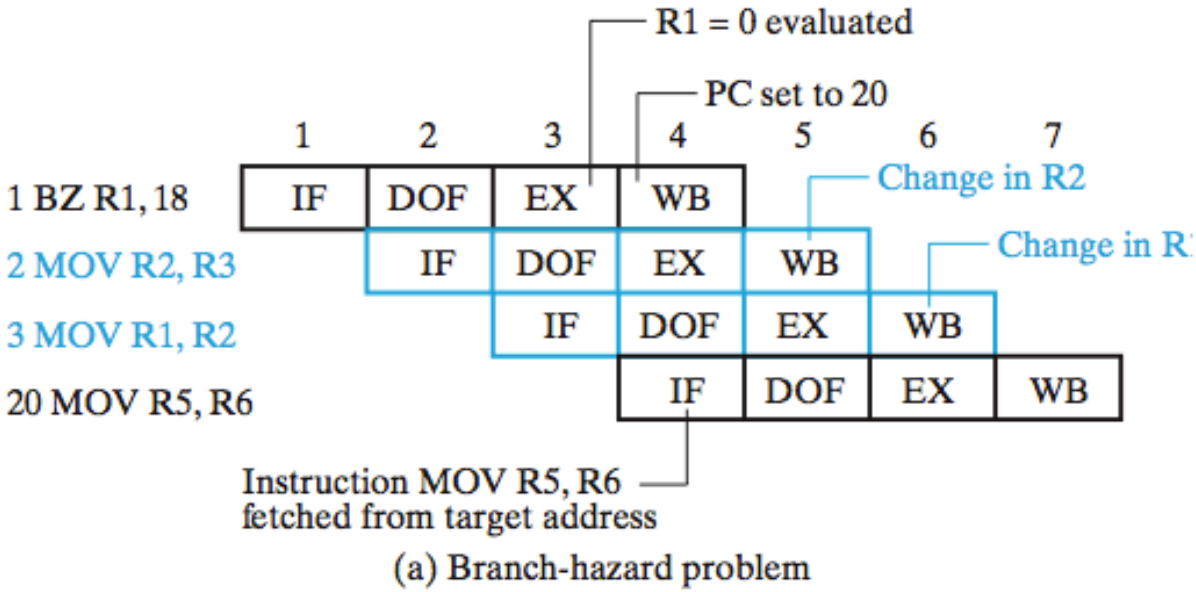


(b) A program-based solution

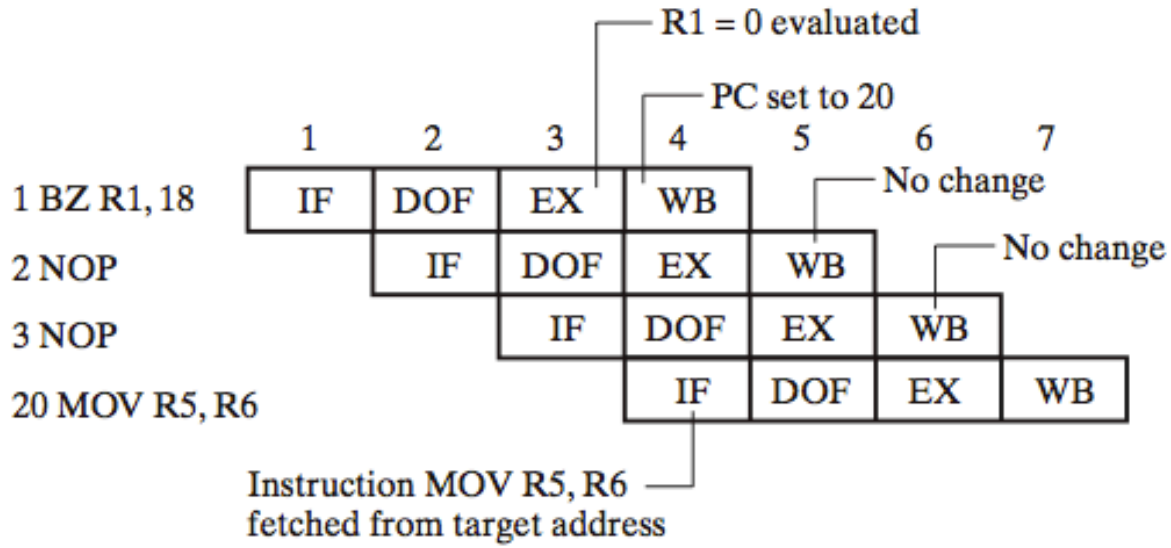
A hardware-based solution



Control Hazards



(a) Branch-hazard problem



(b) Program-based solution

Control Hazards