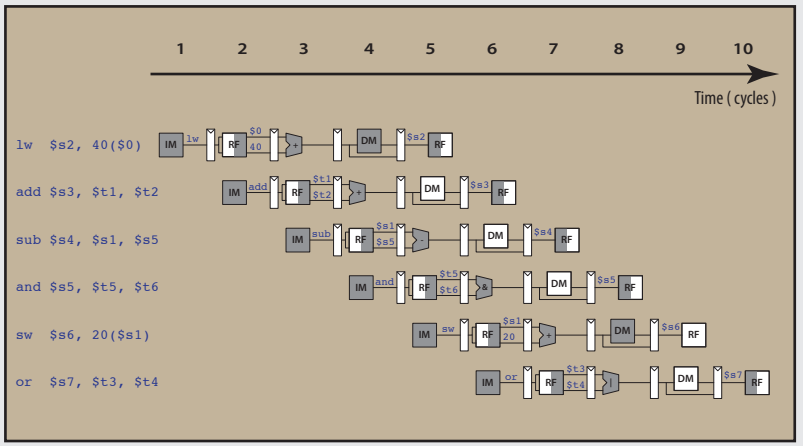
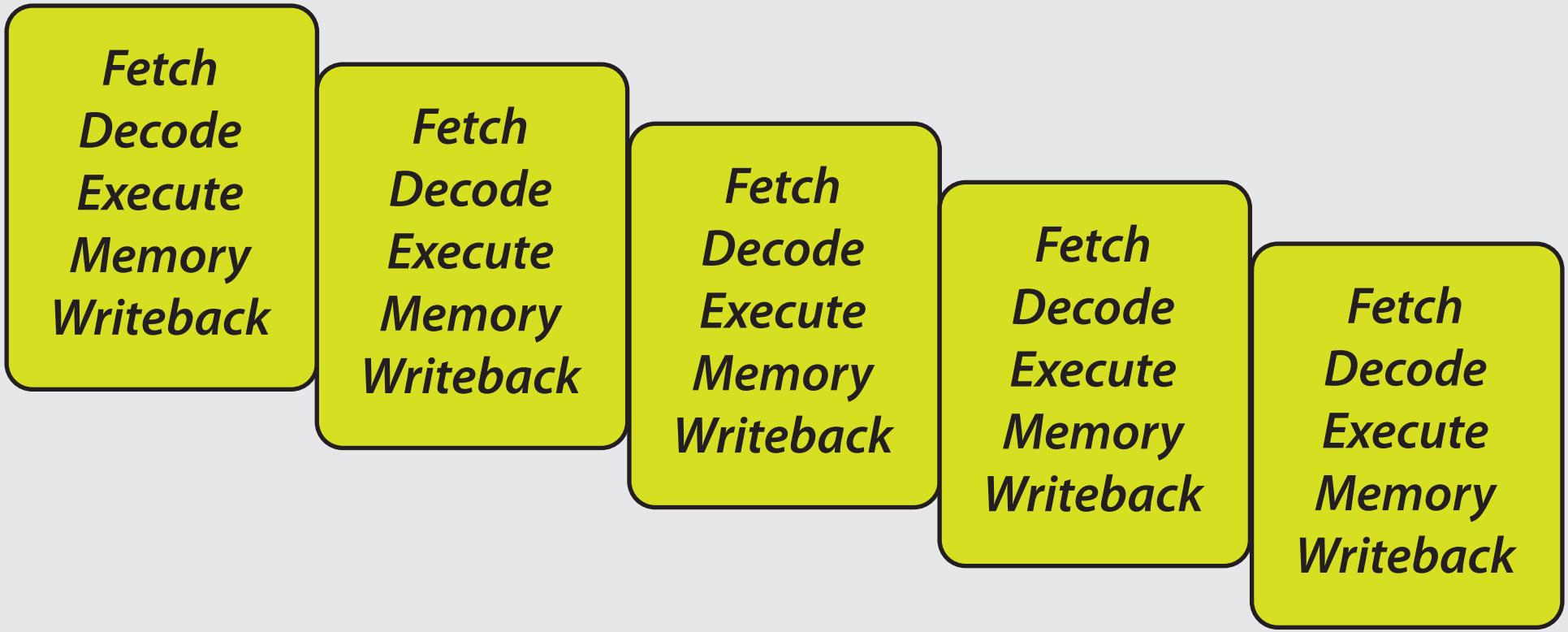
Computer Architecture

Paul Mellies

Lecture 16 : Pipelining and dependence hazards



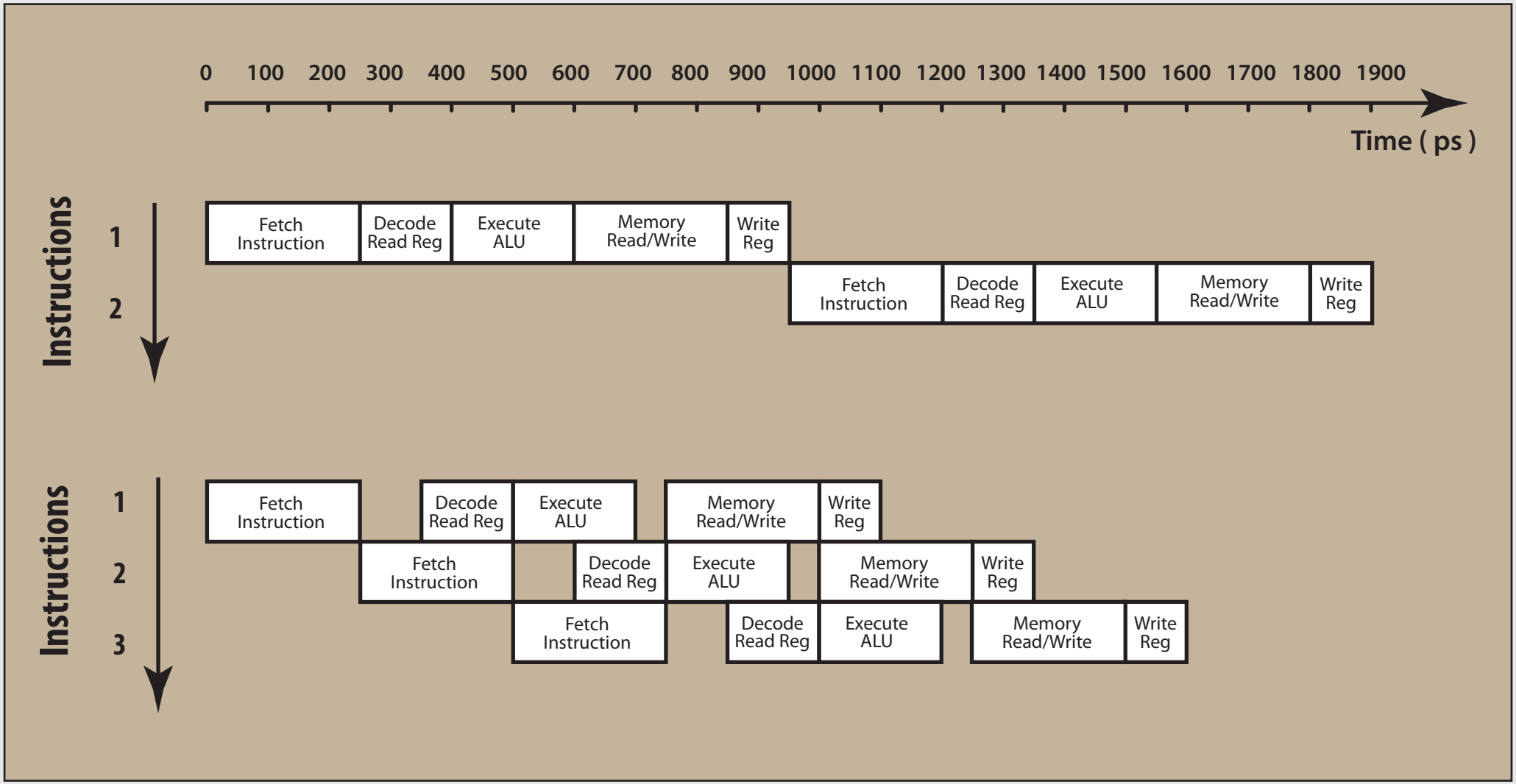
The pipelined microarchitecture



The five stages of the pipelined microarchitecture

- ***Fetch*** The processor reads the instruction from instruction memory.
- ***Decode*** The processor reads the source operands from the register file and decodes the instruction to produce the control signals.
- ***Execute*** The processor performs a computation with the ALU.
- ***Memory*** The processor reads or writes data memory.
- ***Writeback*** The processor writes the result to the register file, when applicable.

The basic principle of pipelining



Timing diagrams for single-cycled and pipelined processors

Exercise

What are the ***instruction latency*** and ***throughput*** of

1. the single-cycled microarchitecture
2. the pipelined microarchitecture

described on the previous slide ?

Solution

What are the ***instruction latency*** and ***throughput*** of

1. the single-cycled microarchitecture
2. the pipelined microarchitecture

described on the previous slide ?

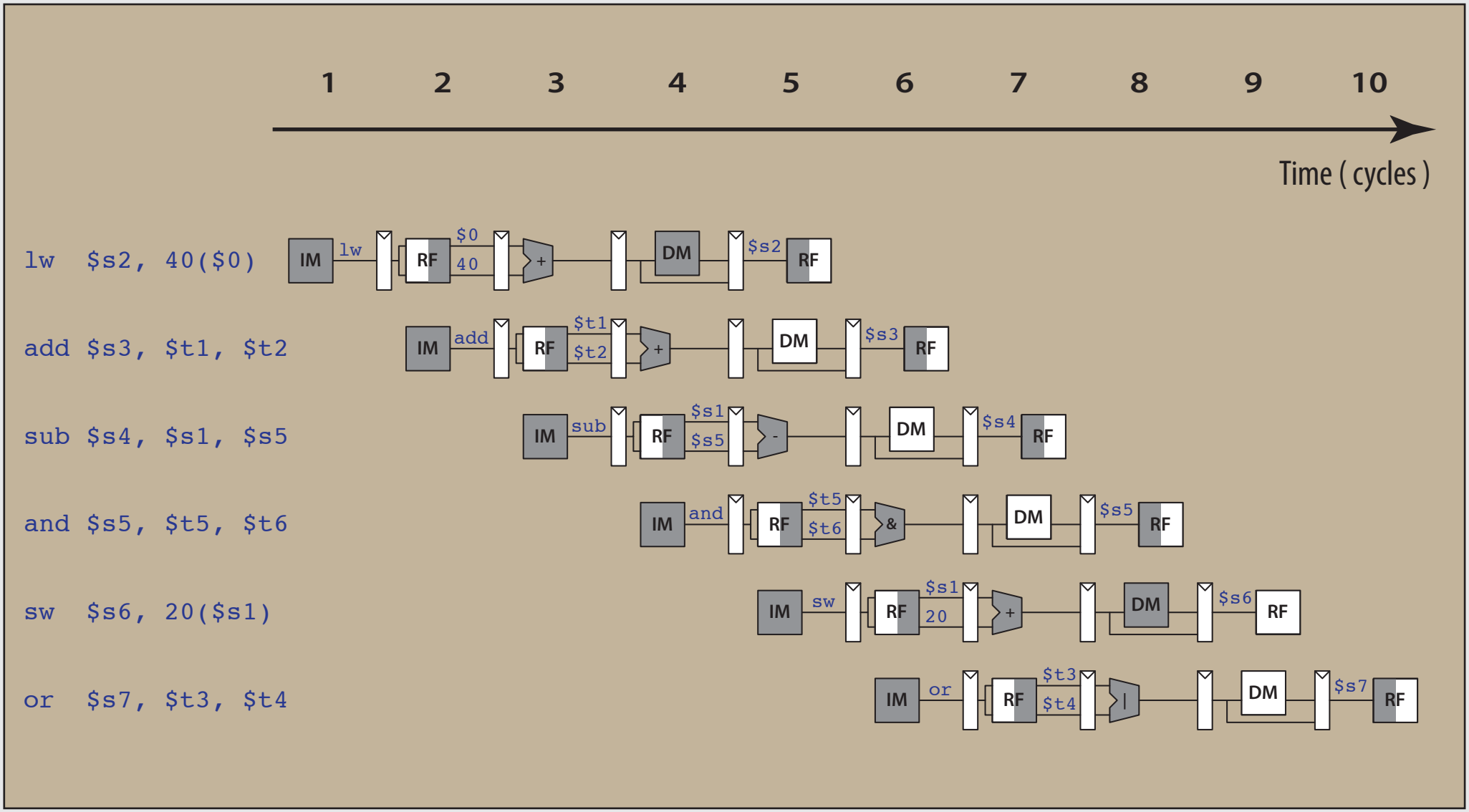
In the case of the single-cycled architecture :

latency = 250 + 150 + 200 + 250 + 100 = 950 ps
throughput = 1 instruction per 950 ps
= 1.05 billion instructions per second

In the case of the pipelined architecture :

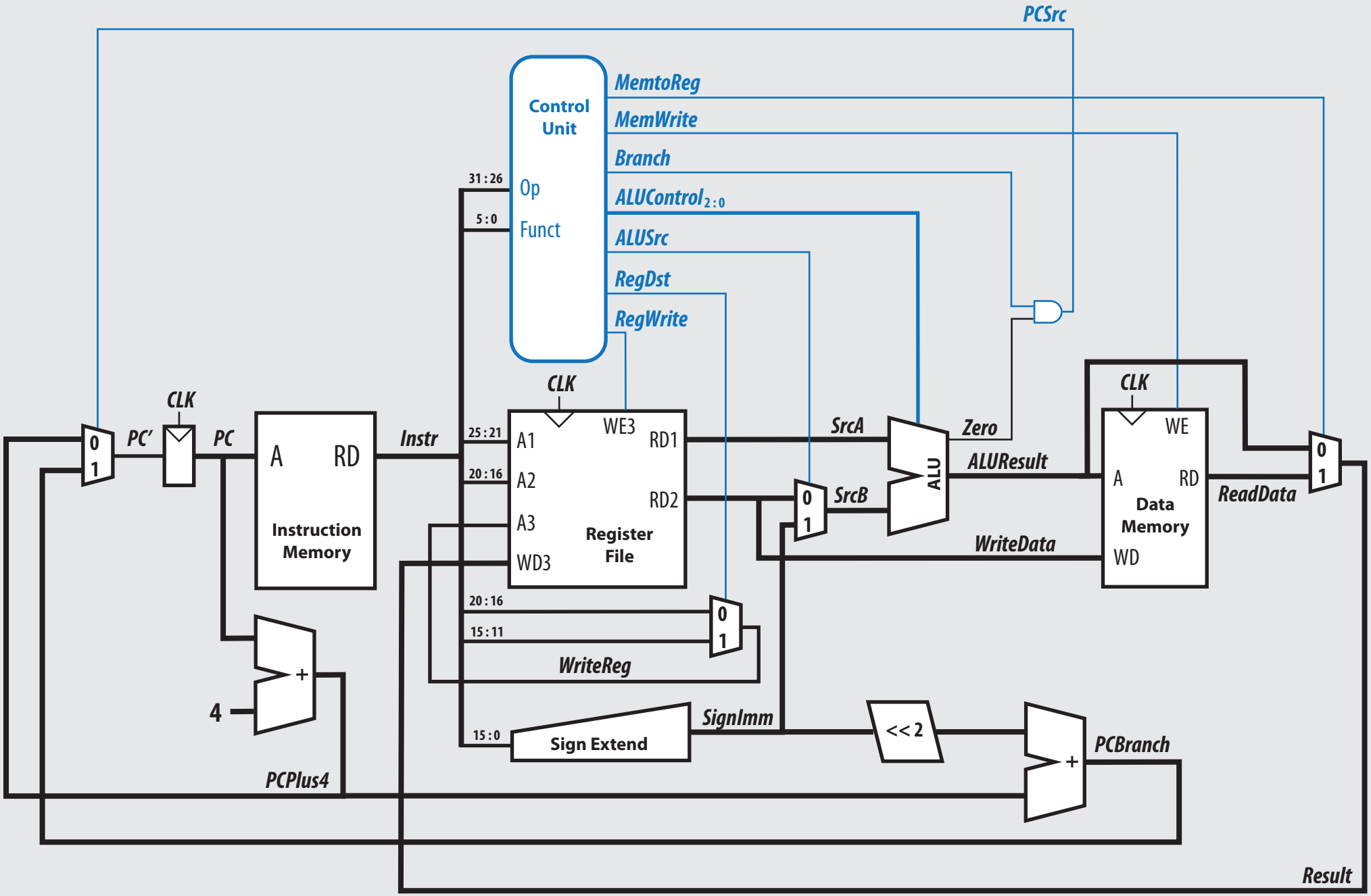
latency = 250 + 250 + 250 + 250 + 250 = 1250 ps
throughput = 1 instruction per 250 ps
= 4 billion instructions per second

Abstract view of pipeline in operation

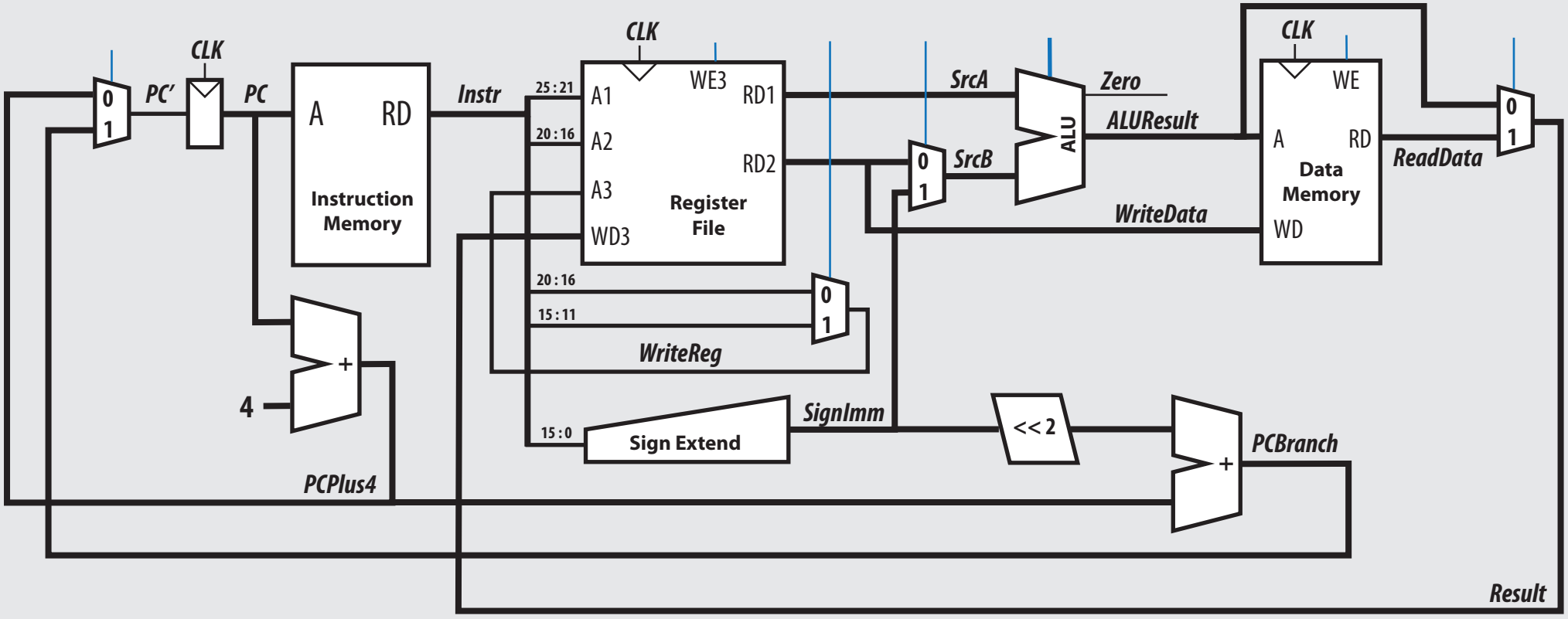


instruction memory (IM), register file (RF) read, ALU execution, data memory (DM), register file (RF) write-back

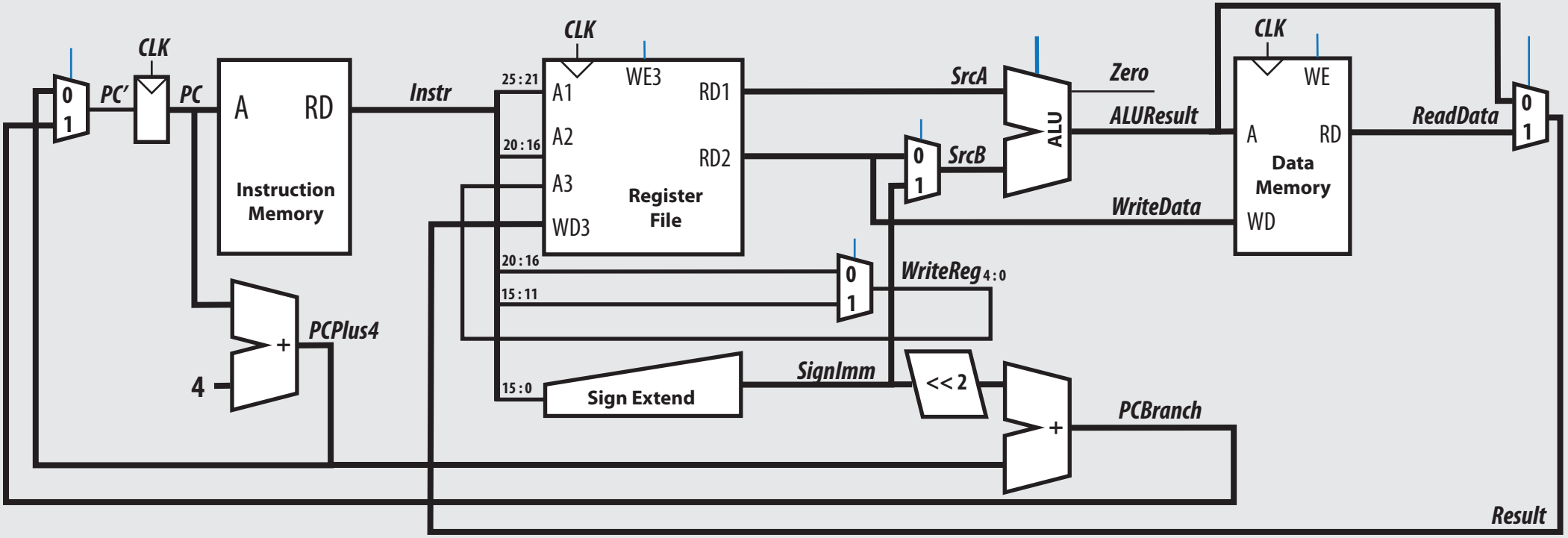
Complete single-cycle MIPS processor



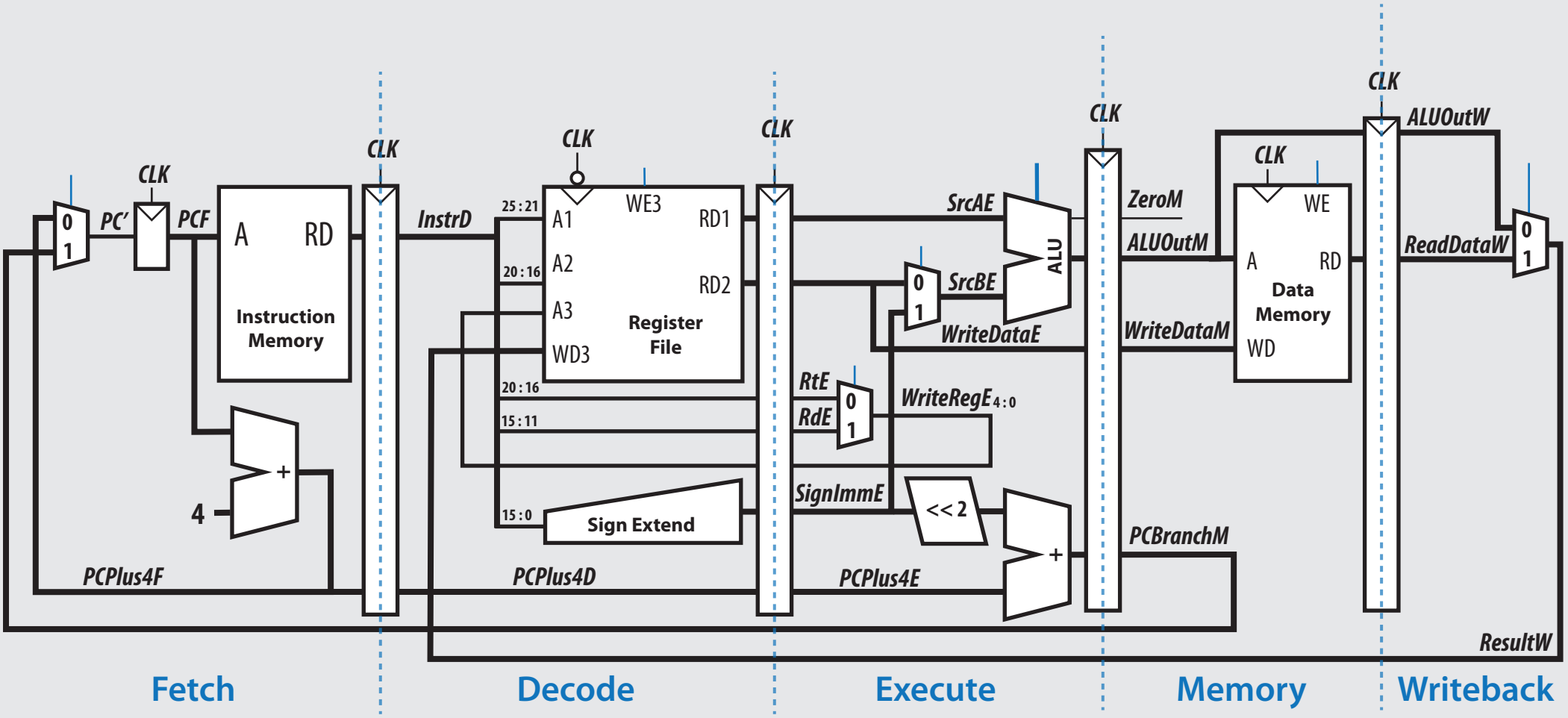
Complete single-cycle MIPS processor



Single-cycle datapath

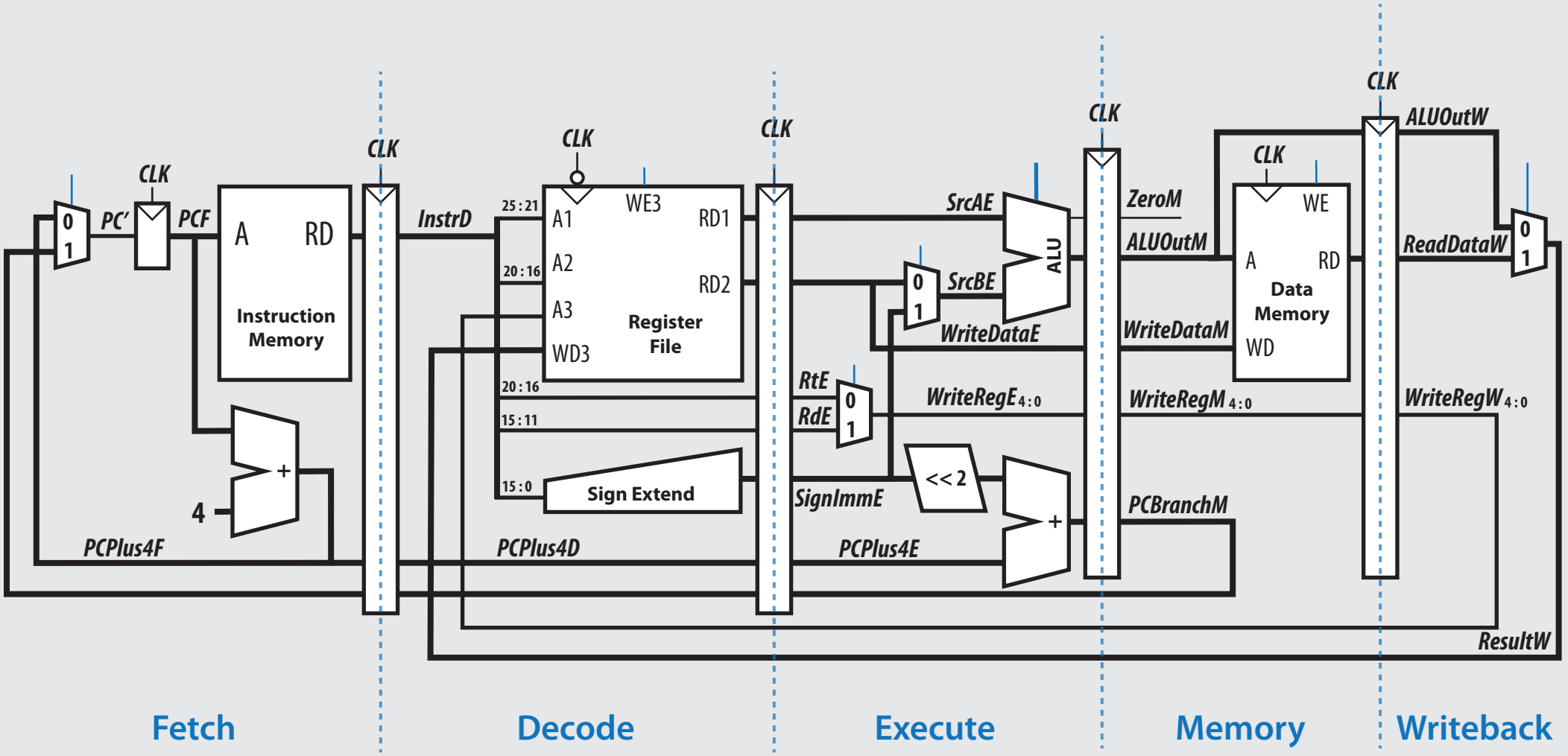


Pipelined datapath

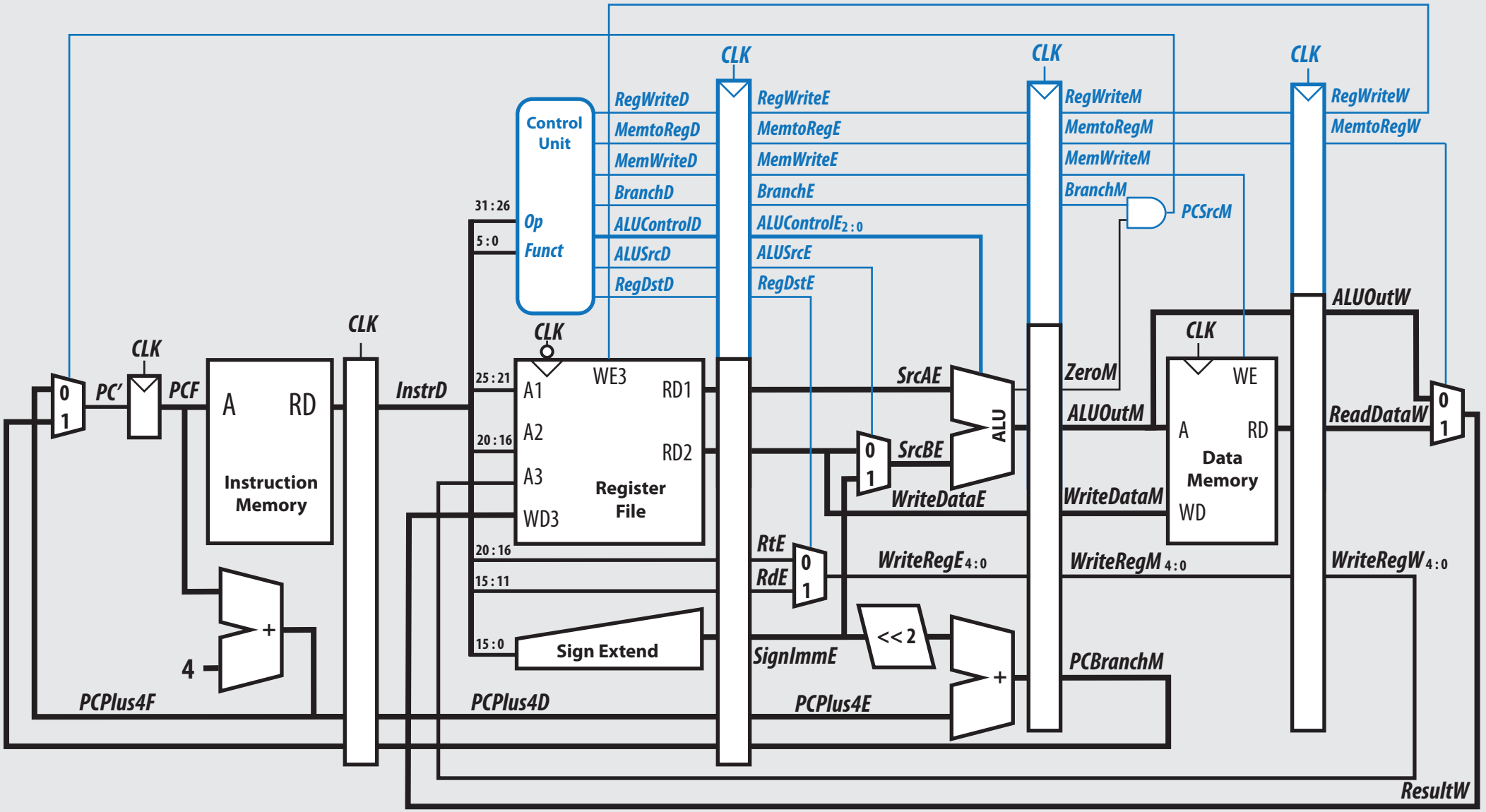


Note the inverter on the clock port of the Register File :
this ensures that the Register File writes on the ***falling edge*** the ***CLK*** signal

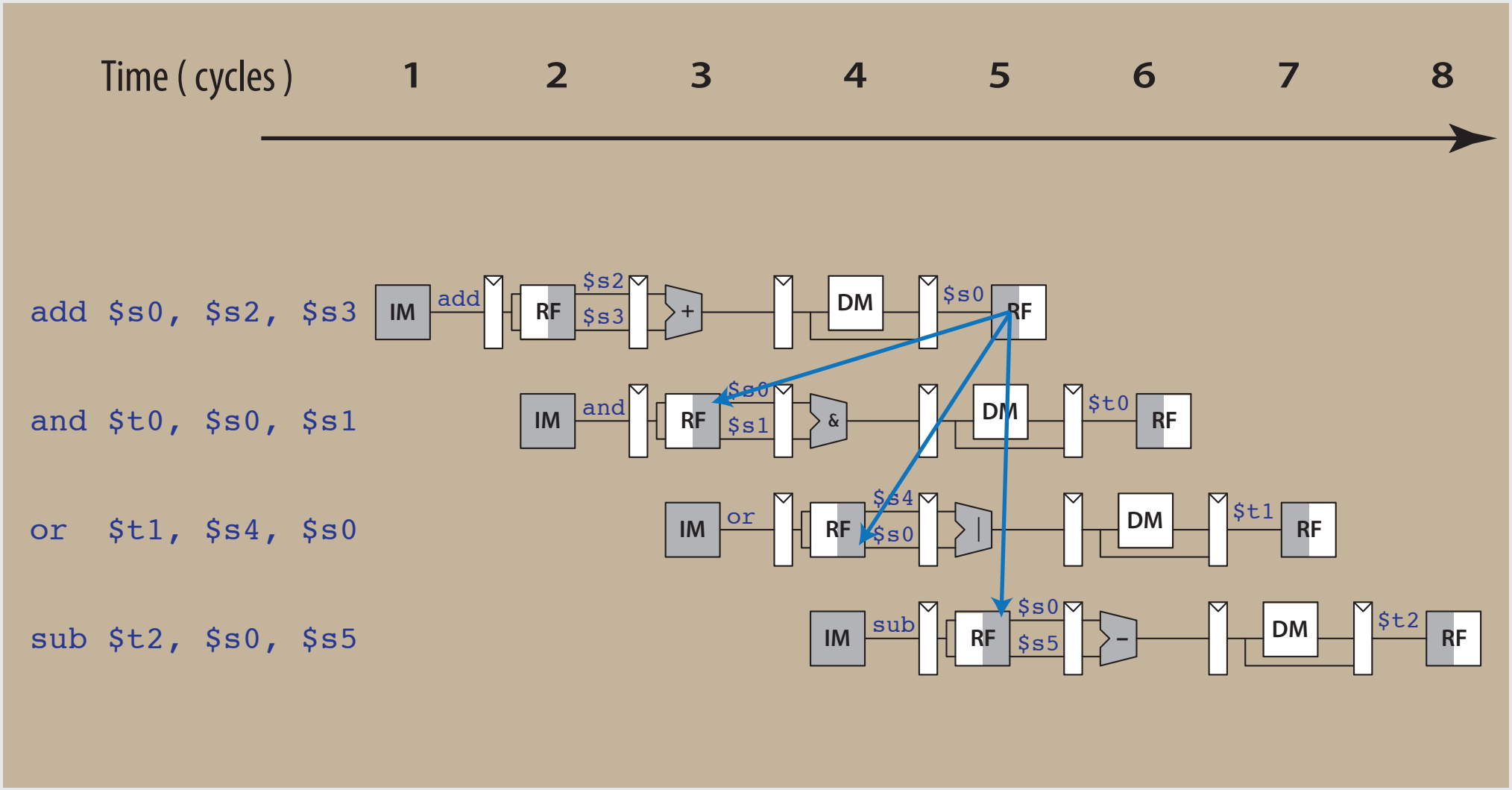
Corrected pipelined datapath



Pipelined processor with control

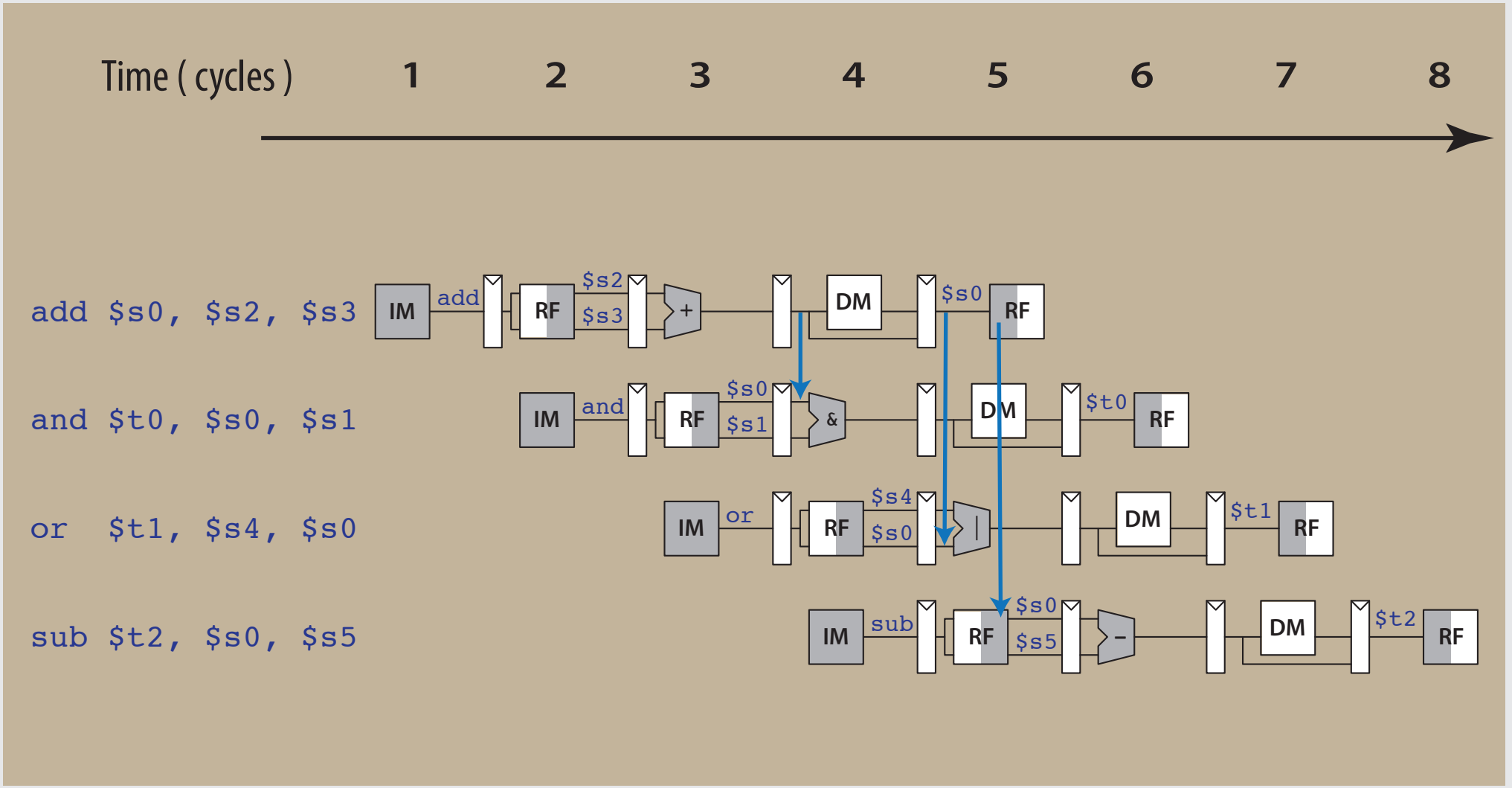


Abstract pipeline diagram illustrating RAW hazards



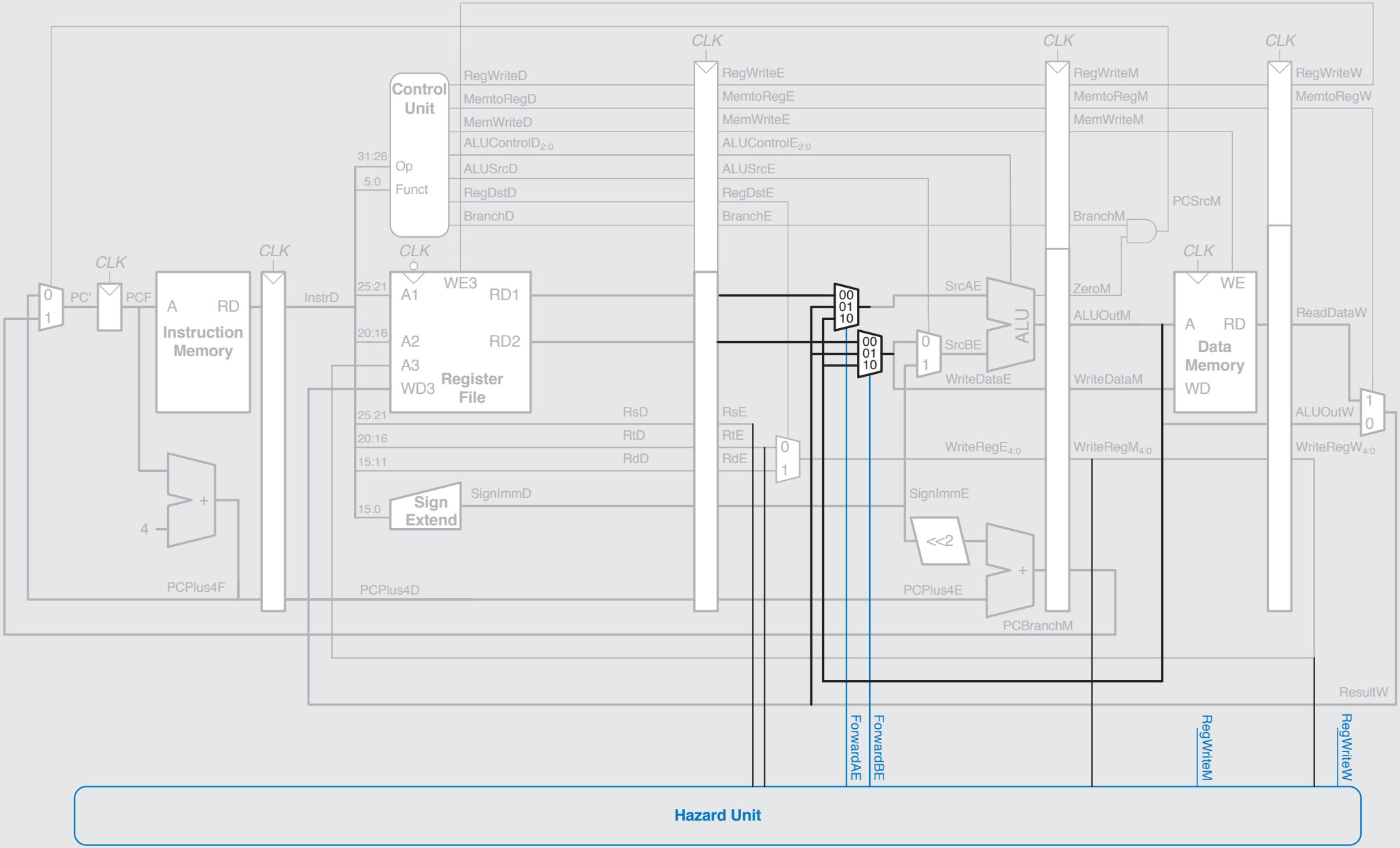
RAW = Read After Write

Abstract pipeline diagram illustrating forwarding

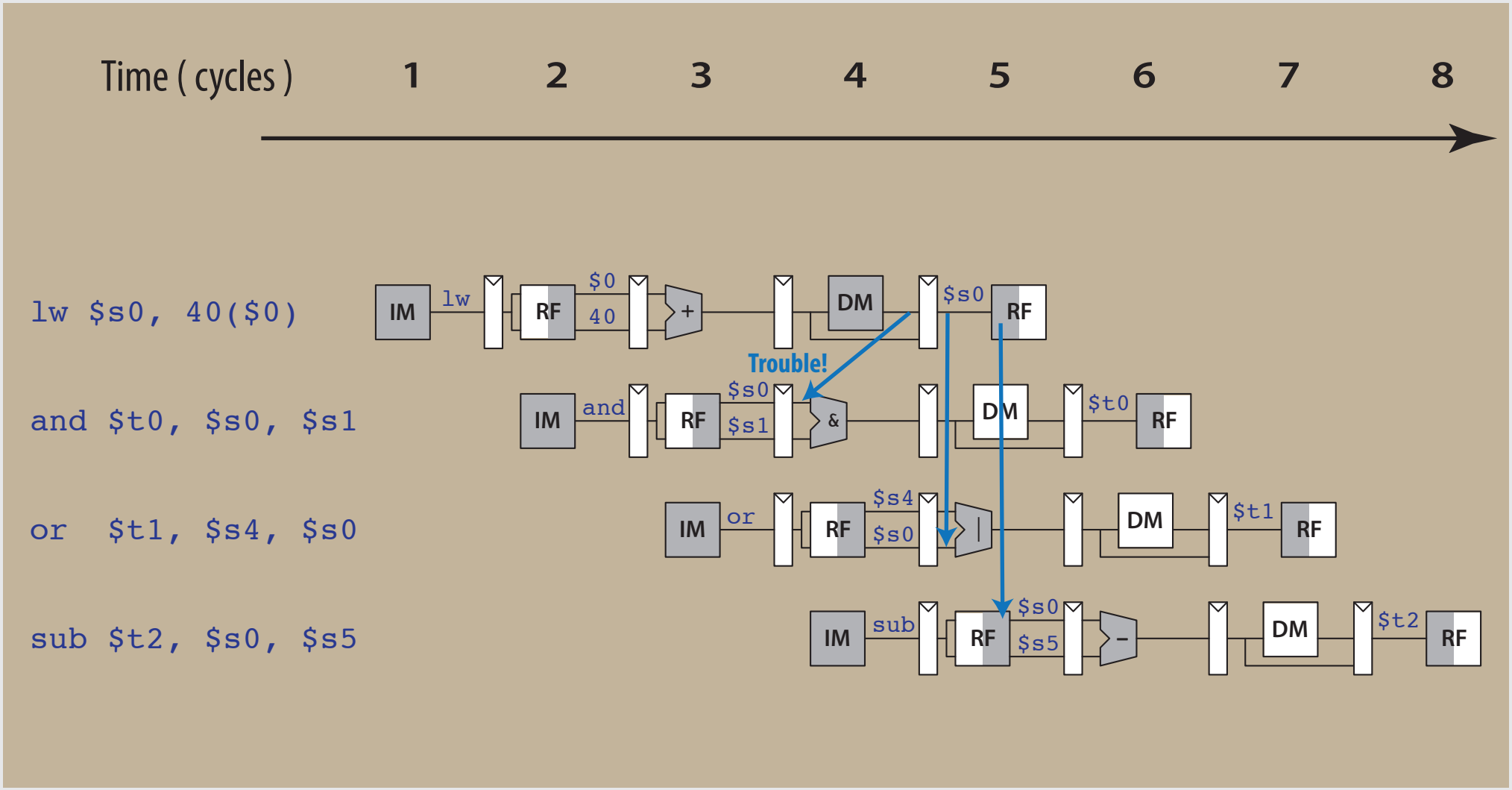


RAW = Read After Write

Pipelined processor with forwarding to solve RAW hazards

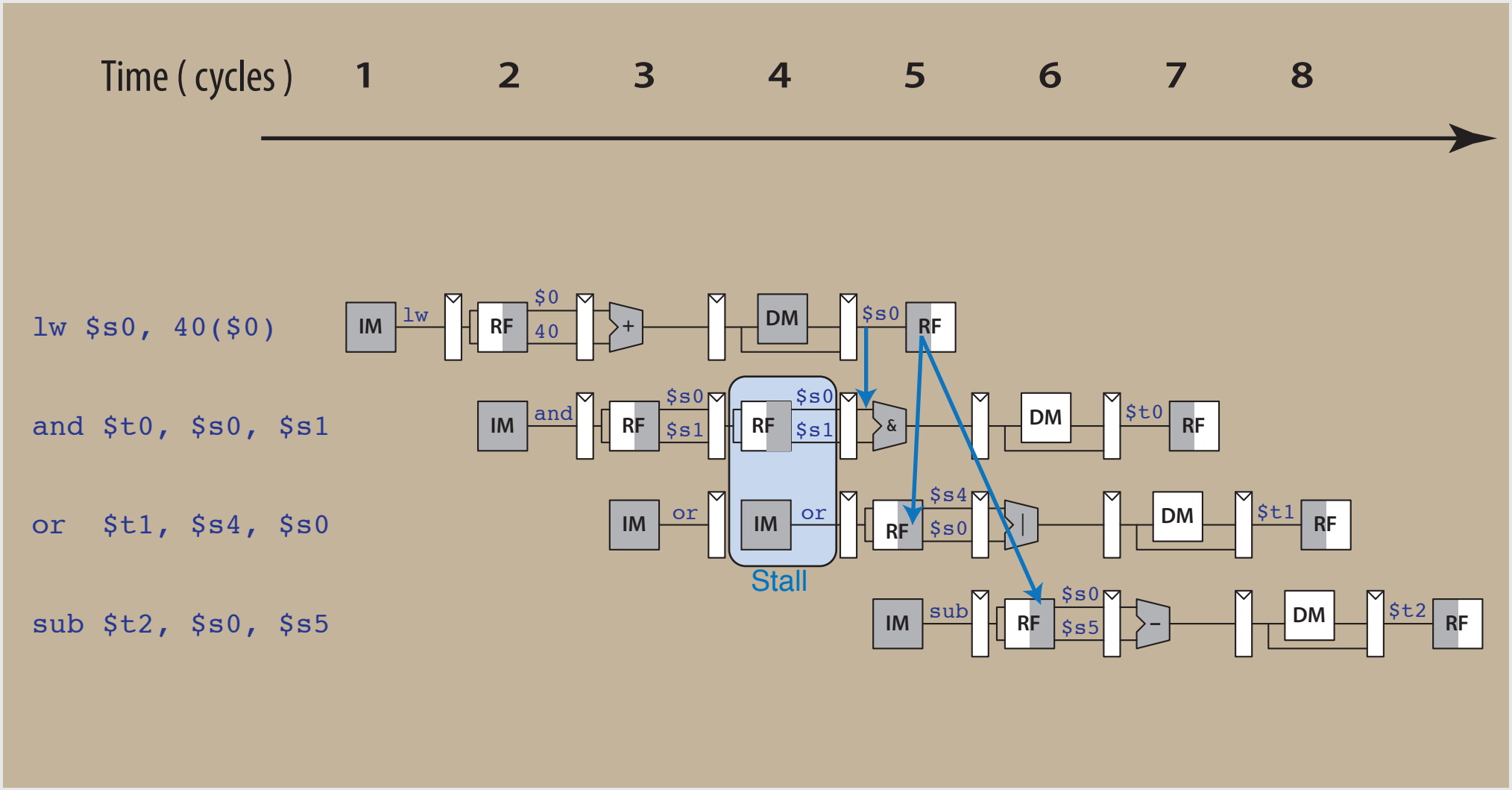


Abstract pipeline diagram illustrating trouble forwarding `lw`



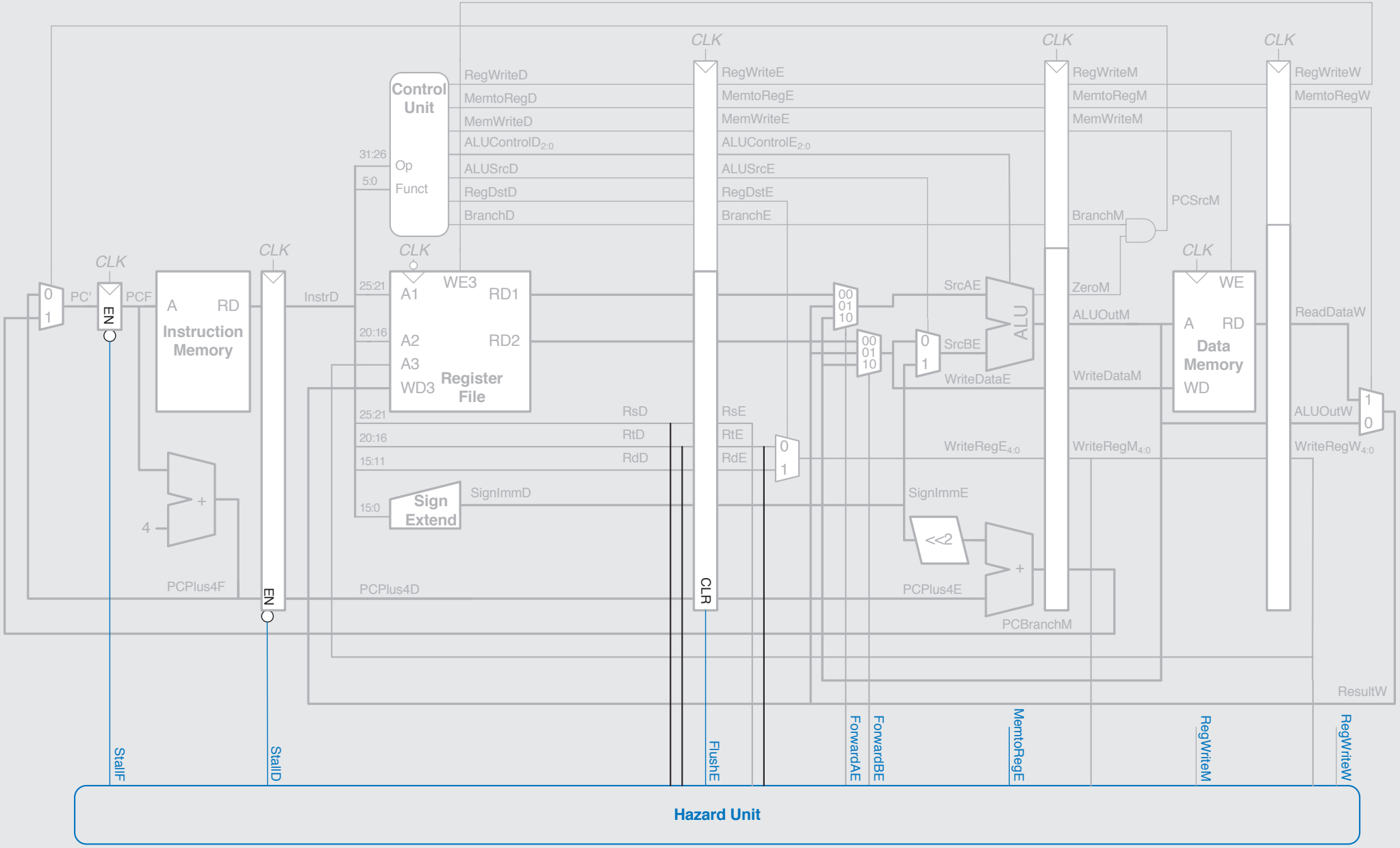
RAW = Read After Write

Abstract pipeline diagram illustrating stalls to solve hazards

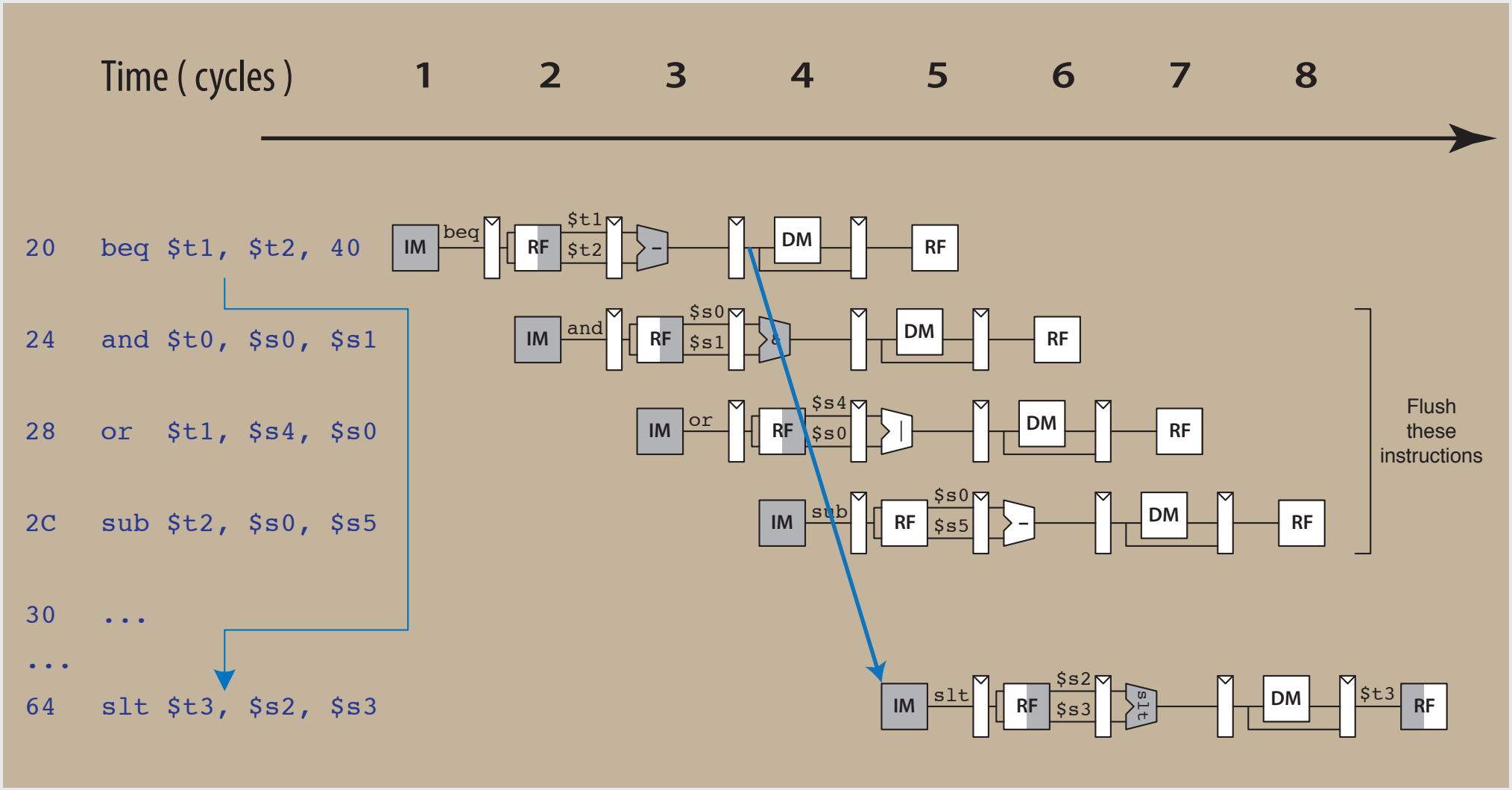


RAW = Read After Write

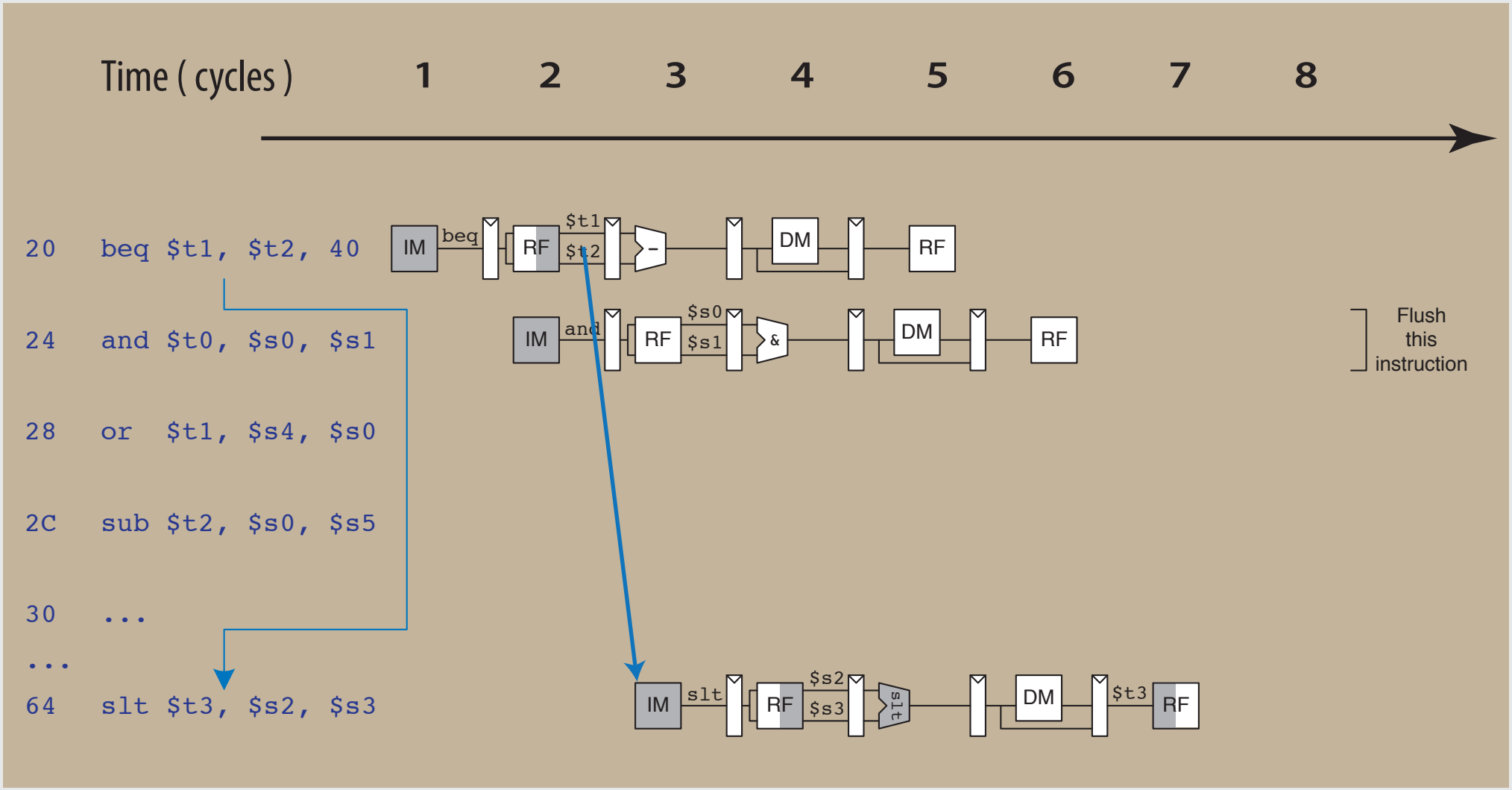
Pipelined processor with stalls to solve `lw` hazards



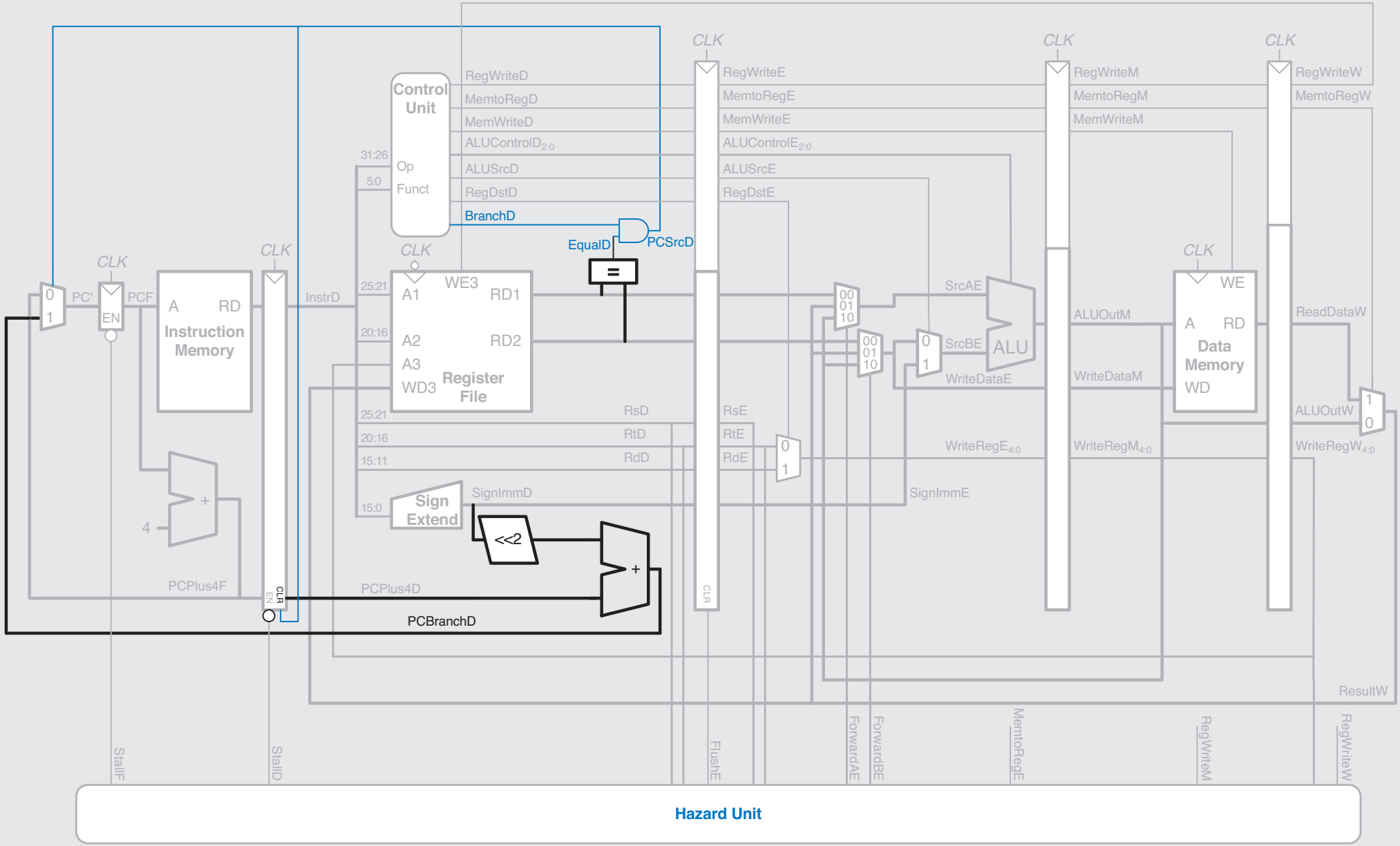
Abstract pipeline diagram illustrating flushing when a branch is taken



Abstract pipeline diagram illustrating earlier branch decision



Pipelined processor handling branch control hazards



Pipelined processor with full hazard handling