Filling the Pipeline

This example traces 5 instructions through the pipeline:

```
lw      $10, 20($1)
sub     $11, $2, $3
and     $12, $4, $5
or      $13, $6, $7
add     $14, $8, $9
```

There are 5 stages, so it takes 5 instructions to fill the pipeline

The result registers $10 through $14 are not used as inputs, so there are no dependencies or data hazards

Filling the Pipeline

Five instructions are required to fill the pipeline

The first instruction completes after 5 cycles

- Hence the pipeline *latency* is 5 cycles
- This is also called the *fill time*

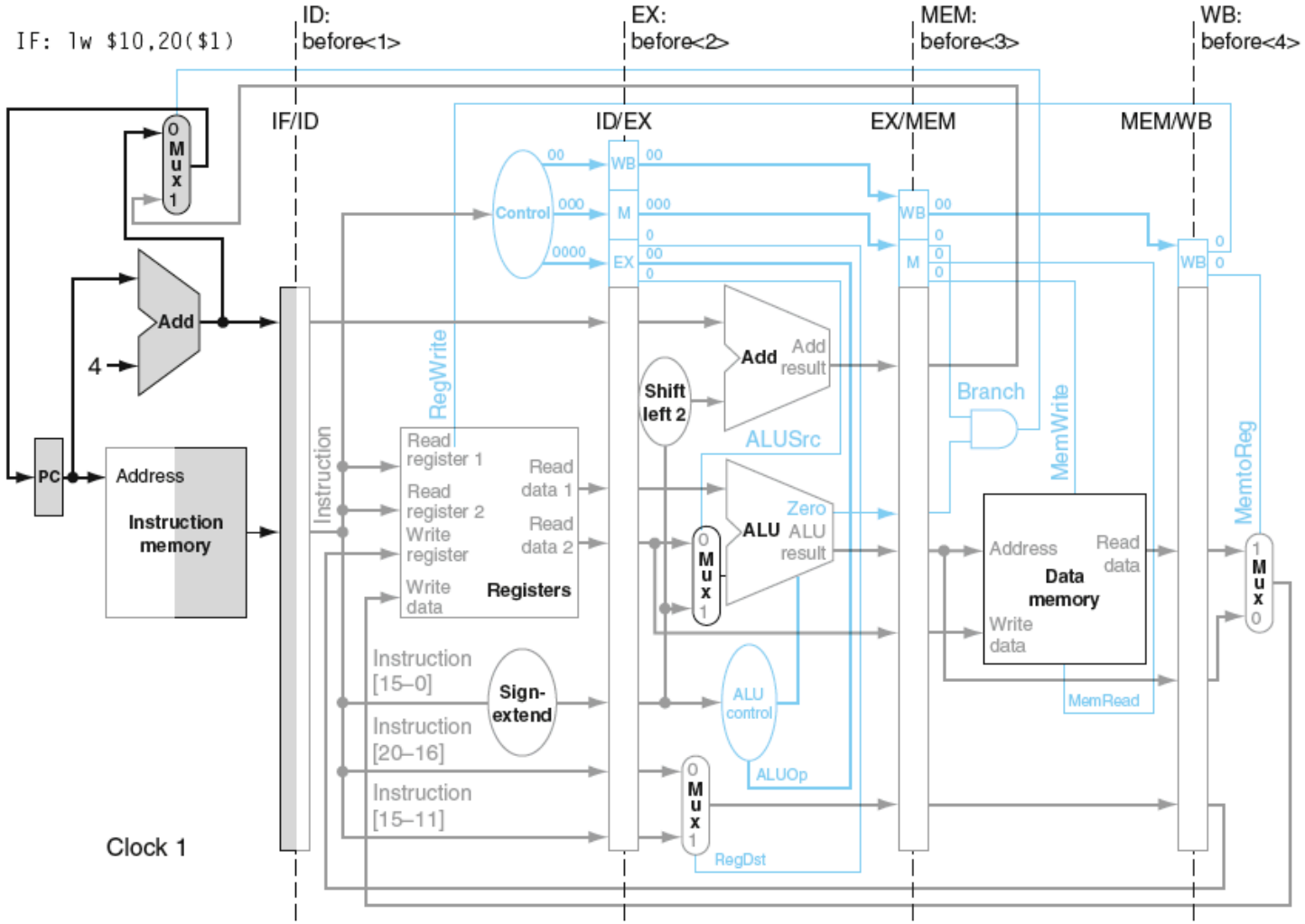
One instruction completes for each subsequent cycle

- pipeline stalls would be required if there were hazards

"before <i>" refers to the i^{th} instruction before those in the example 5-instruction sequence

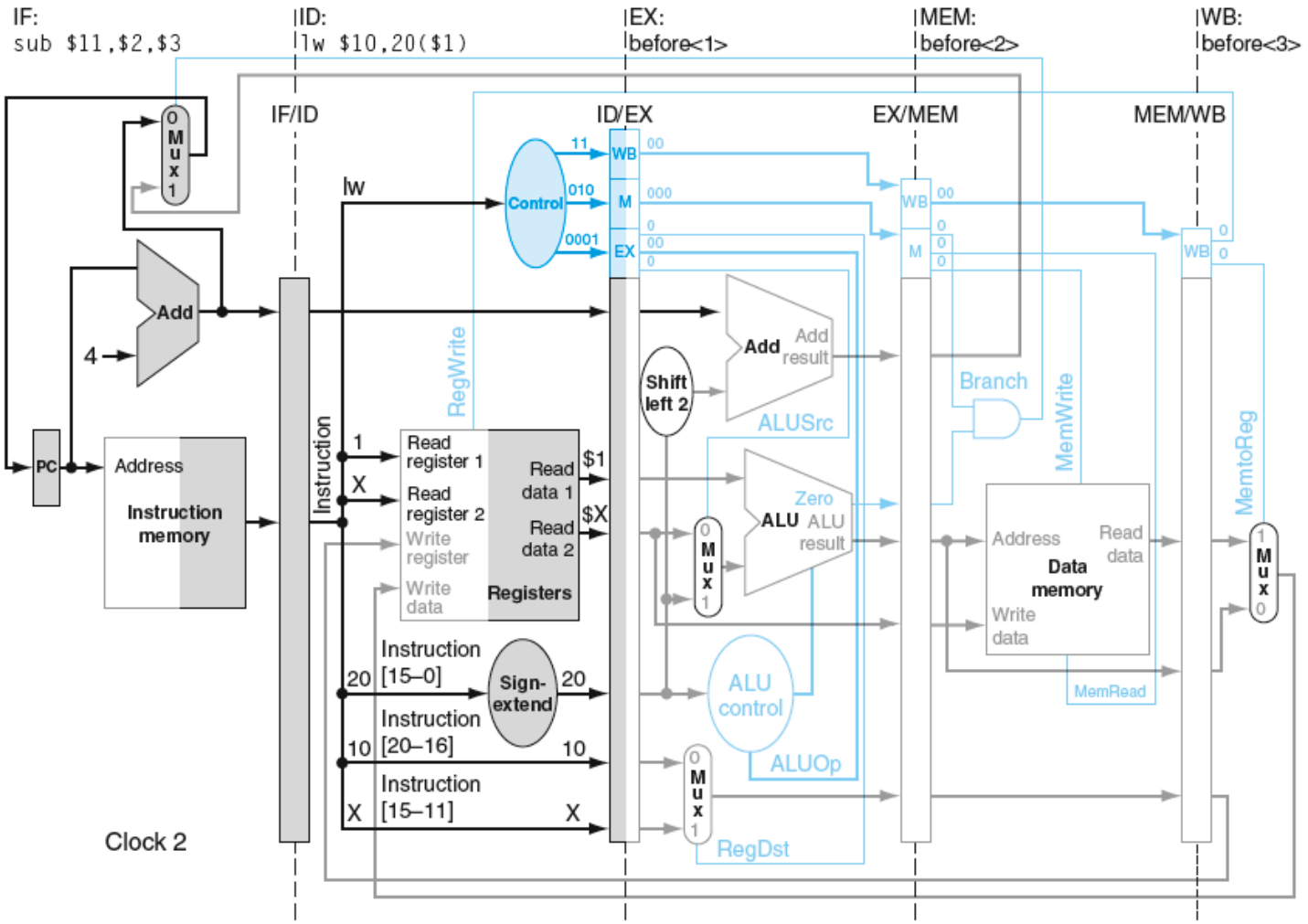
"after <i>" will refer to the i^{th} instruction following those in the example sequence

Pipelining with no hazards



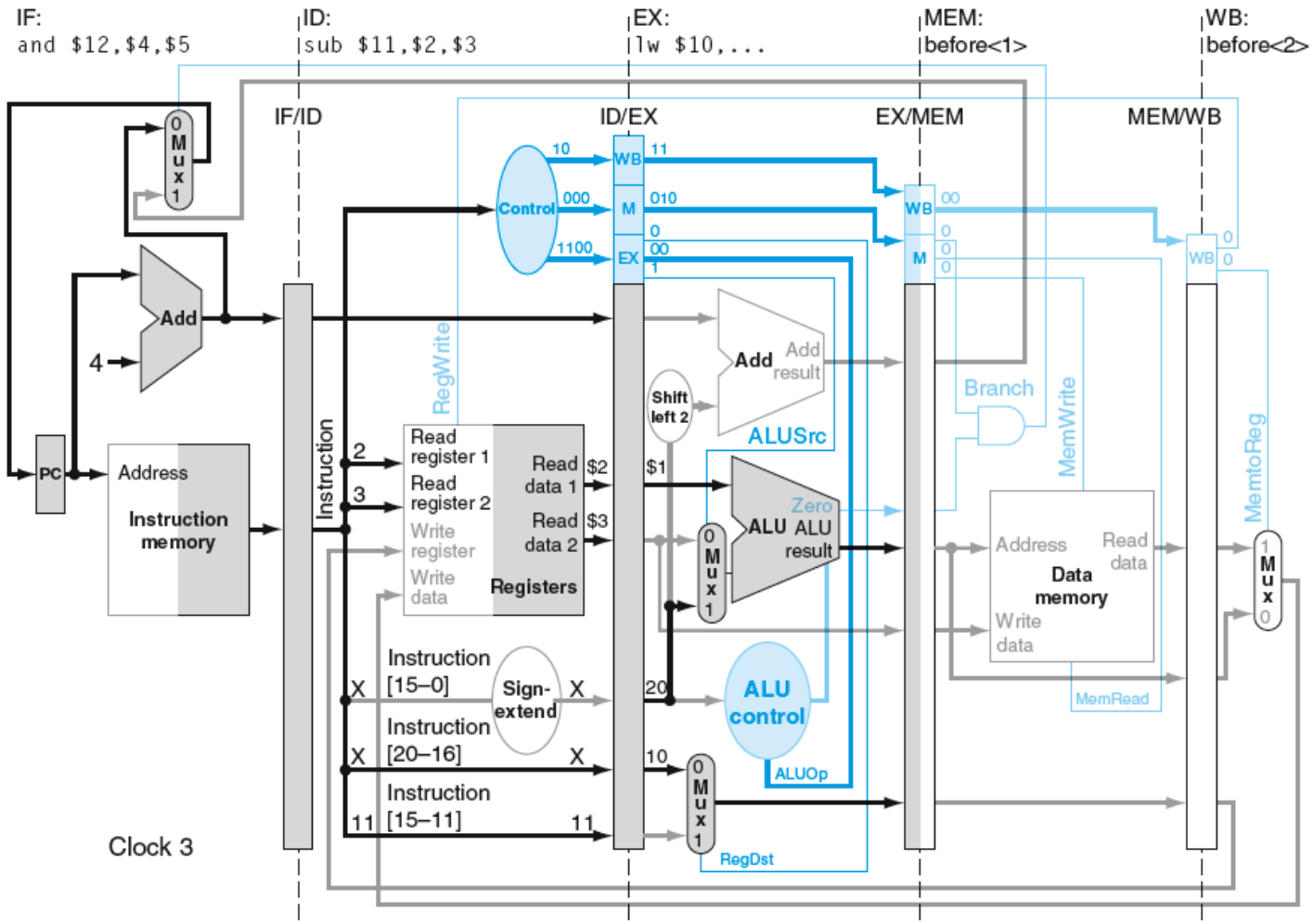
LW is fetched in cycle 1

Pipelining with no hazards



In cycle 2, the control signals for LW are generated and SUB is fetched

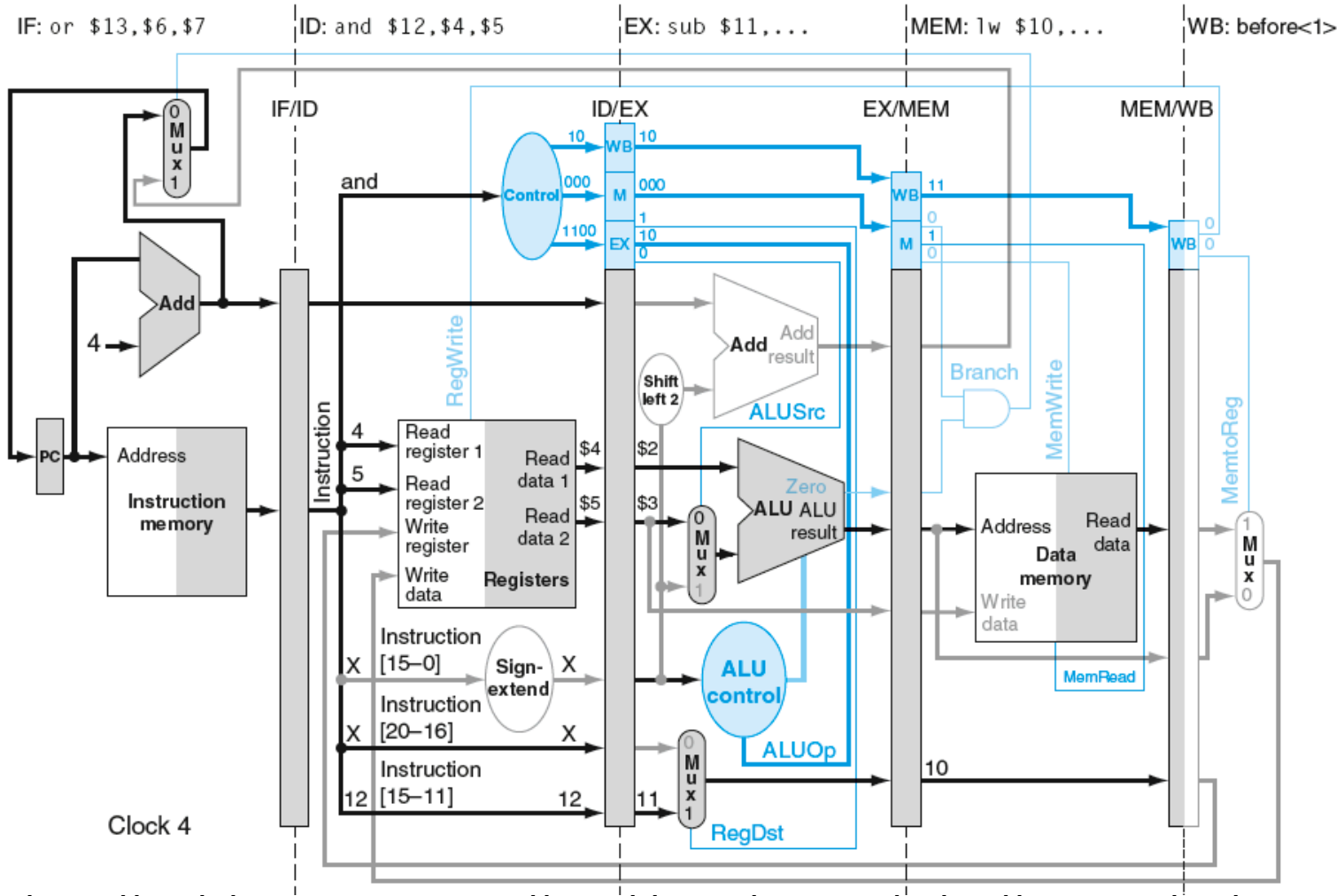
Pipelining with no hazards



Each instruction reads its input registers in the decode stage

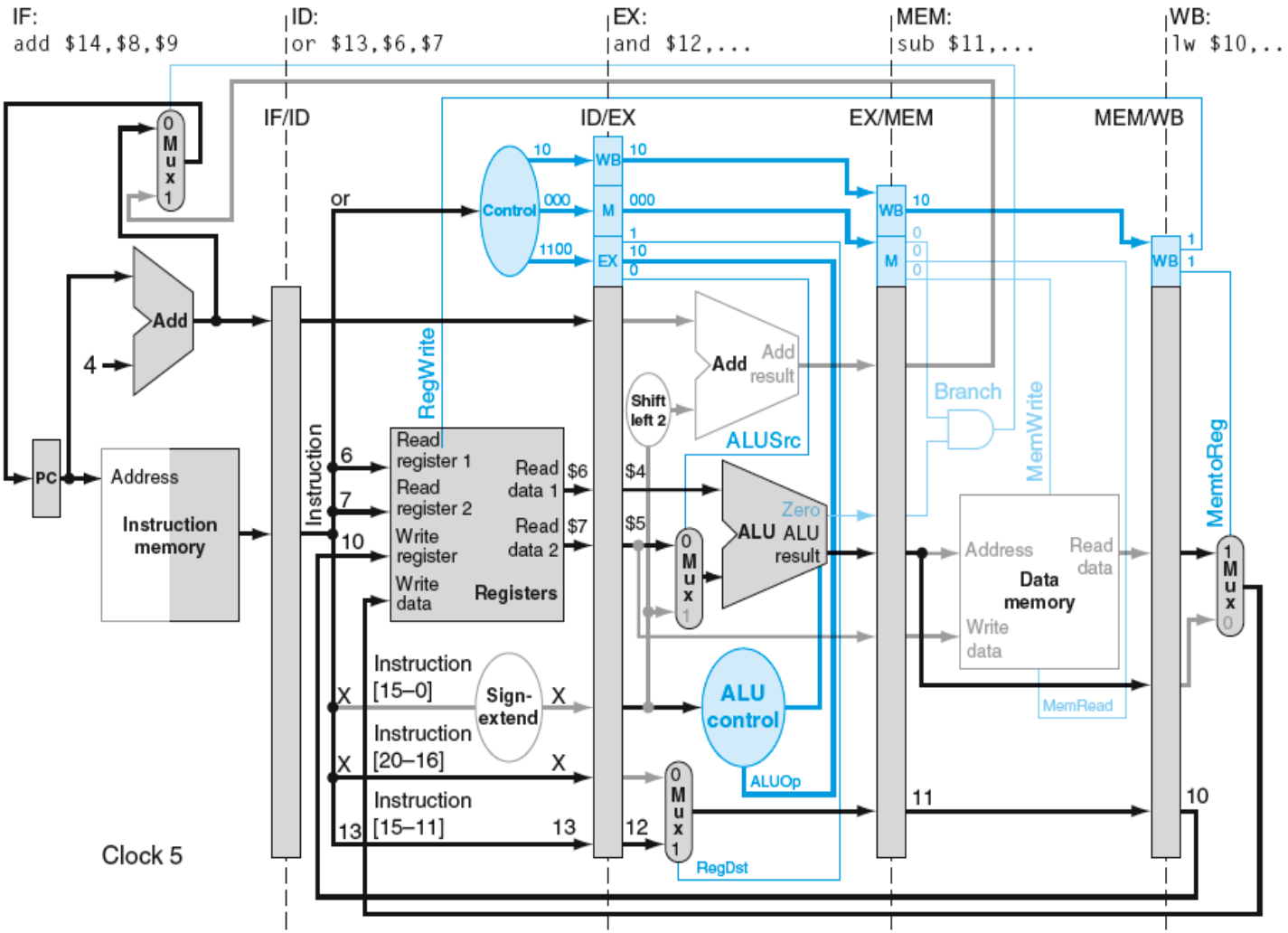
In cycle 3, LW executes, SUB is decoded while AND is fetched

Pipelining with no hazards



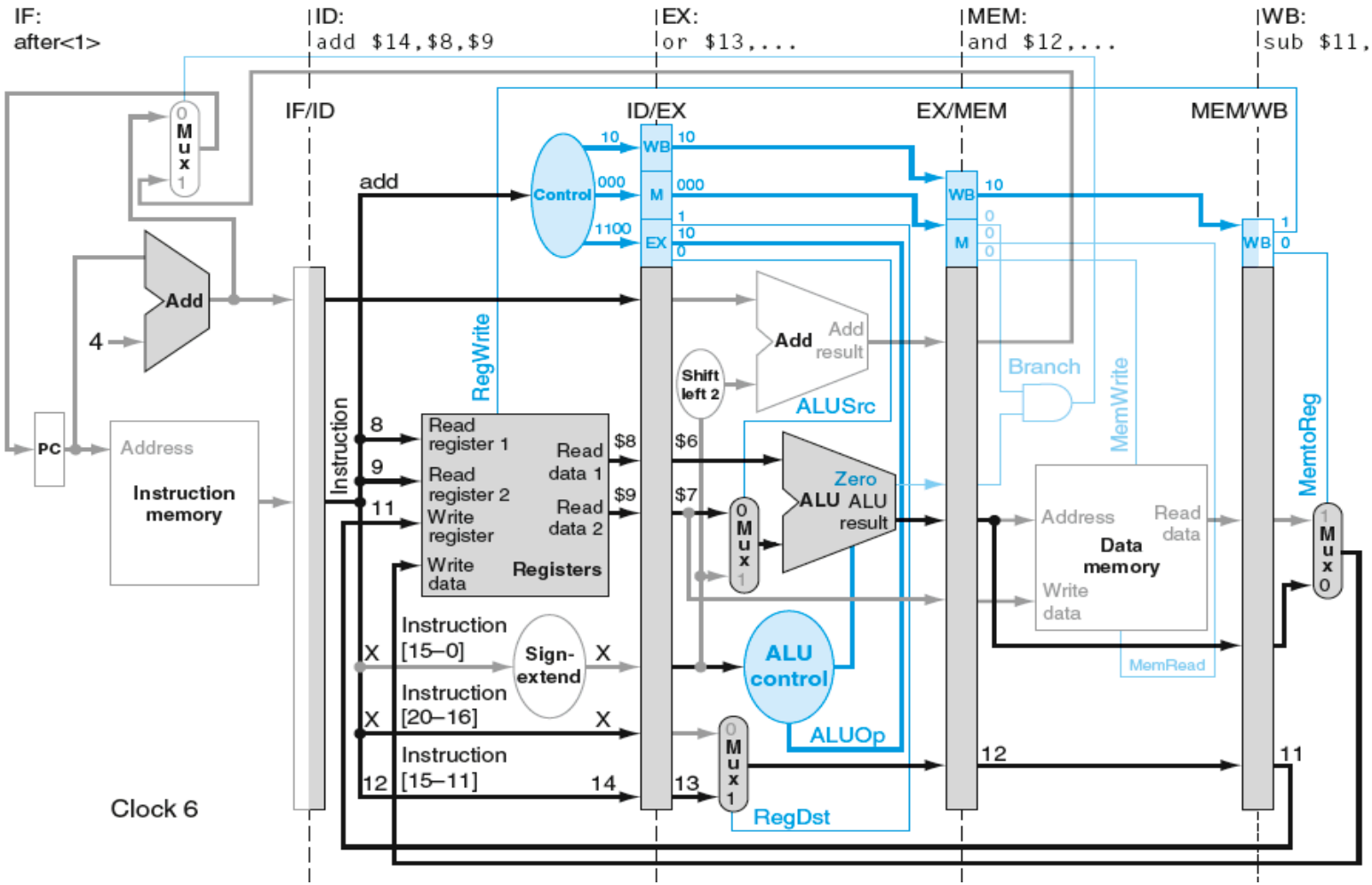
LW reads from the data memory using the address it computed in the execute stage
SUB computes $2 - $3, AND is decoded and reads its input registers ($4 and $5)

Pipelining with no hazards



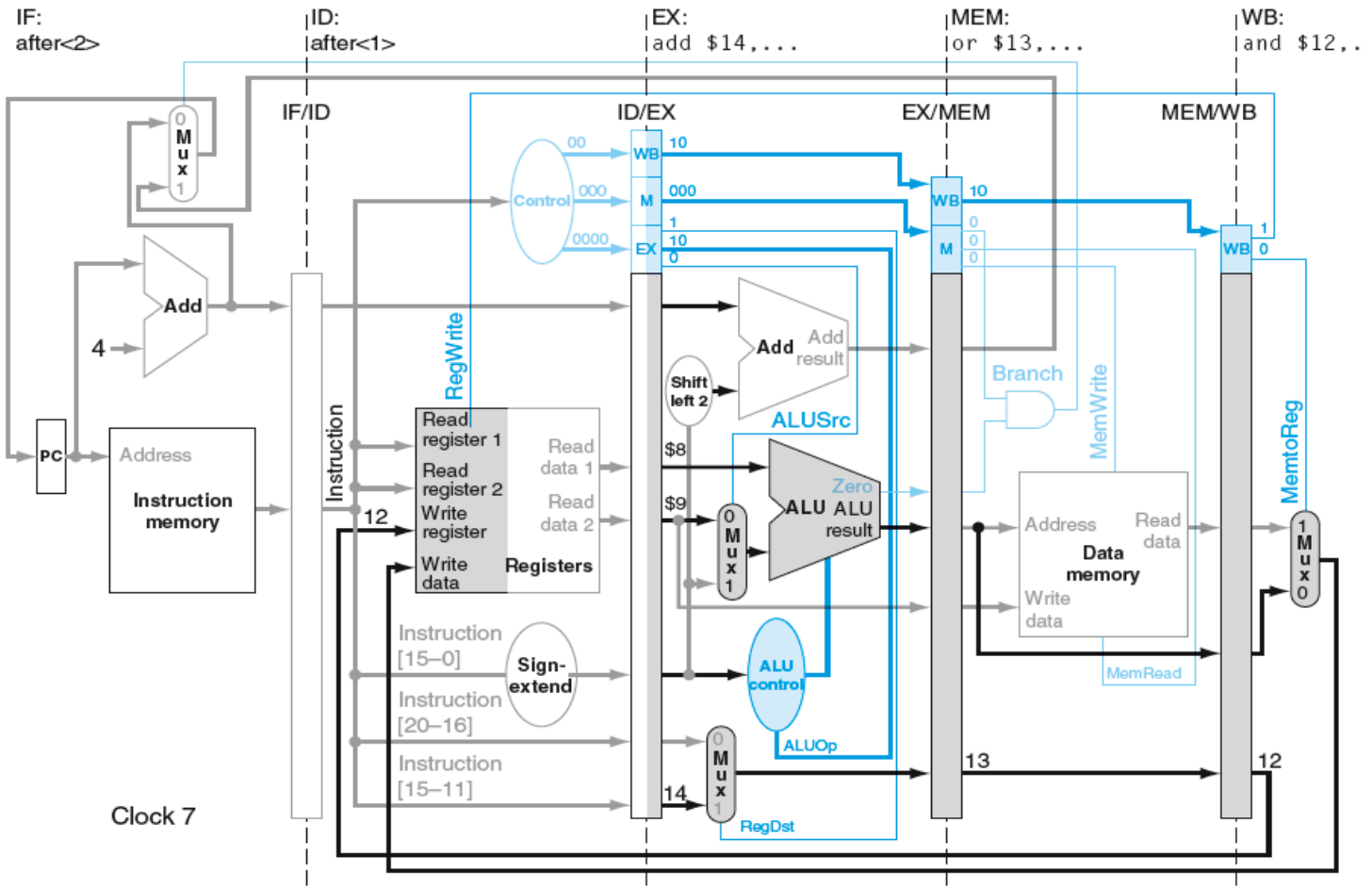
LW completes after writing $10, SUB idles, AND executes, OR is decoded and reads $6 & $7

Pipelining with no hazards



SUB writes $11 and completes, OR executes, ADD is decoded while $8 & $9 are read

Pipelining with no hazards

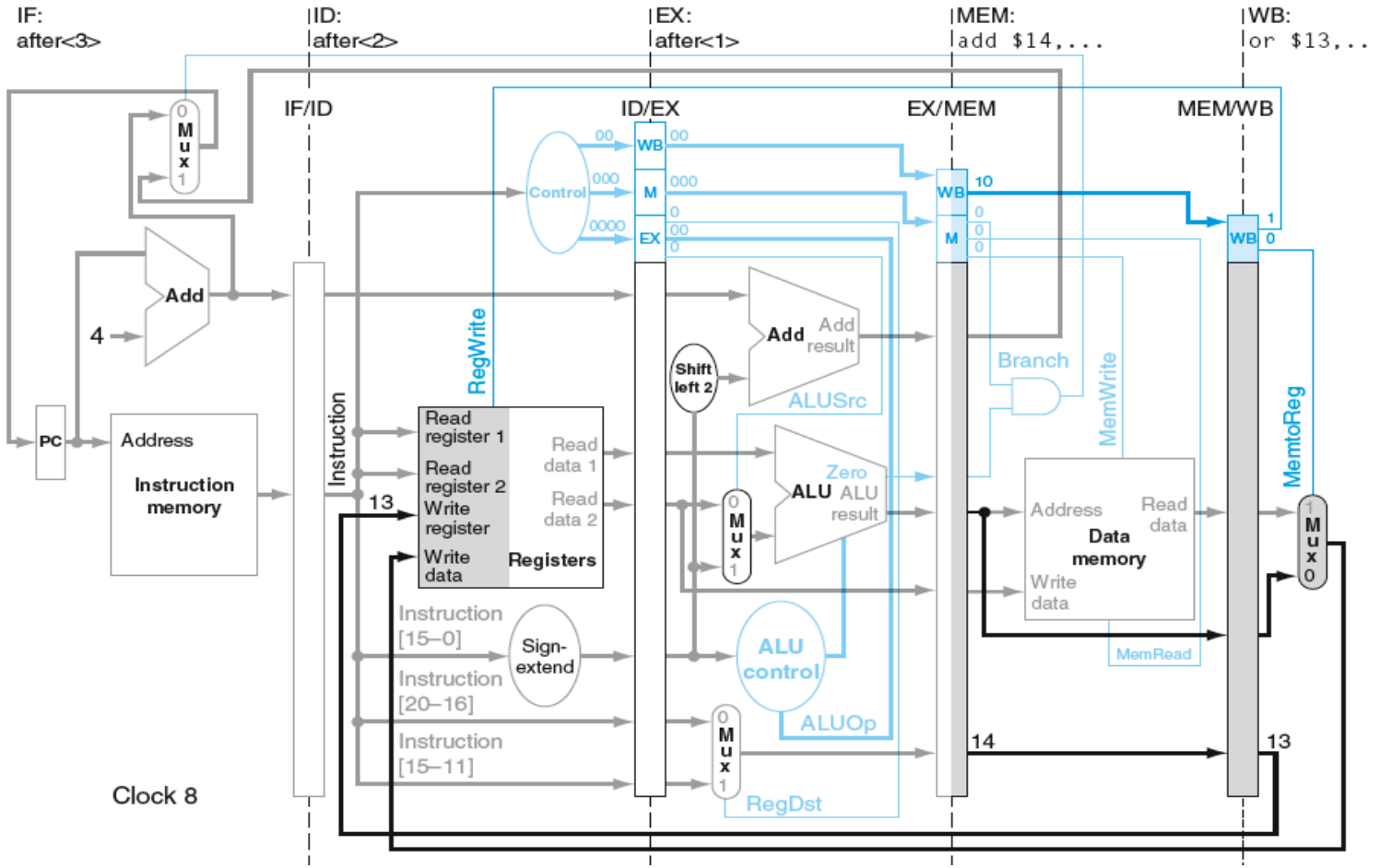


Every instruction must pass through each pipeline stage

Instructions that do not access the data memory idle in stage 4

Hence all instructions take 5 cycles to complete

Pipelining with no hazards

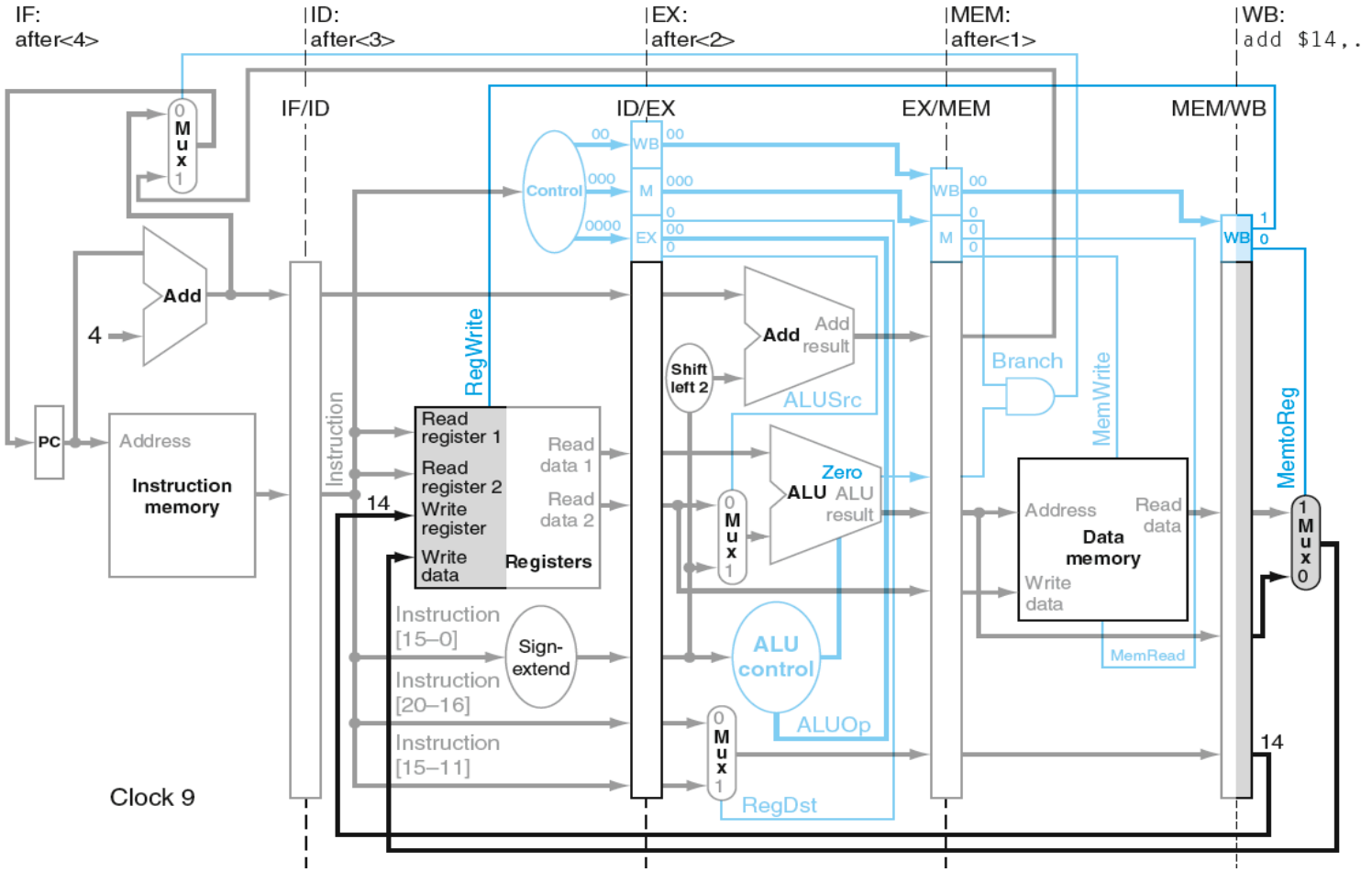


Result registers are written in stage 5 (the write-back stage)

Instructions such as SW and BEQ, do not produce results

In cycle 8, the OR instruction completes

Pipelining with no hazards



The final instruction in the 5-instruction sequence completes in cycle 9

Pipelining with no hazards

- The 5-instruction sequence completes in 9 cycles
- On the non-pipeline multi-cycle system
  - LW takes 5 cycles
  - SUB, AND, OR & AND each takes 4 cycles
  - Total = 21 cycles without pipelining
- Pipelining provides a speedup
  - In this case speedup = 21/9 = 2.33
  - Stalls to cope with data hazards would reduce the speedup