Forwarding within the Pipeline

This sequence will demonstrate how forwarding works

```
sub     $2, $1, $3
and     $4, $2, $5
or      $4, $4, $2
add     $9, $4, $2
```

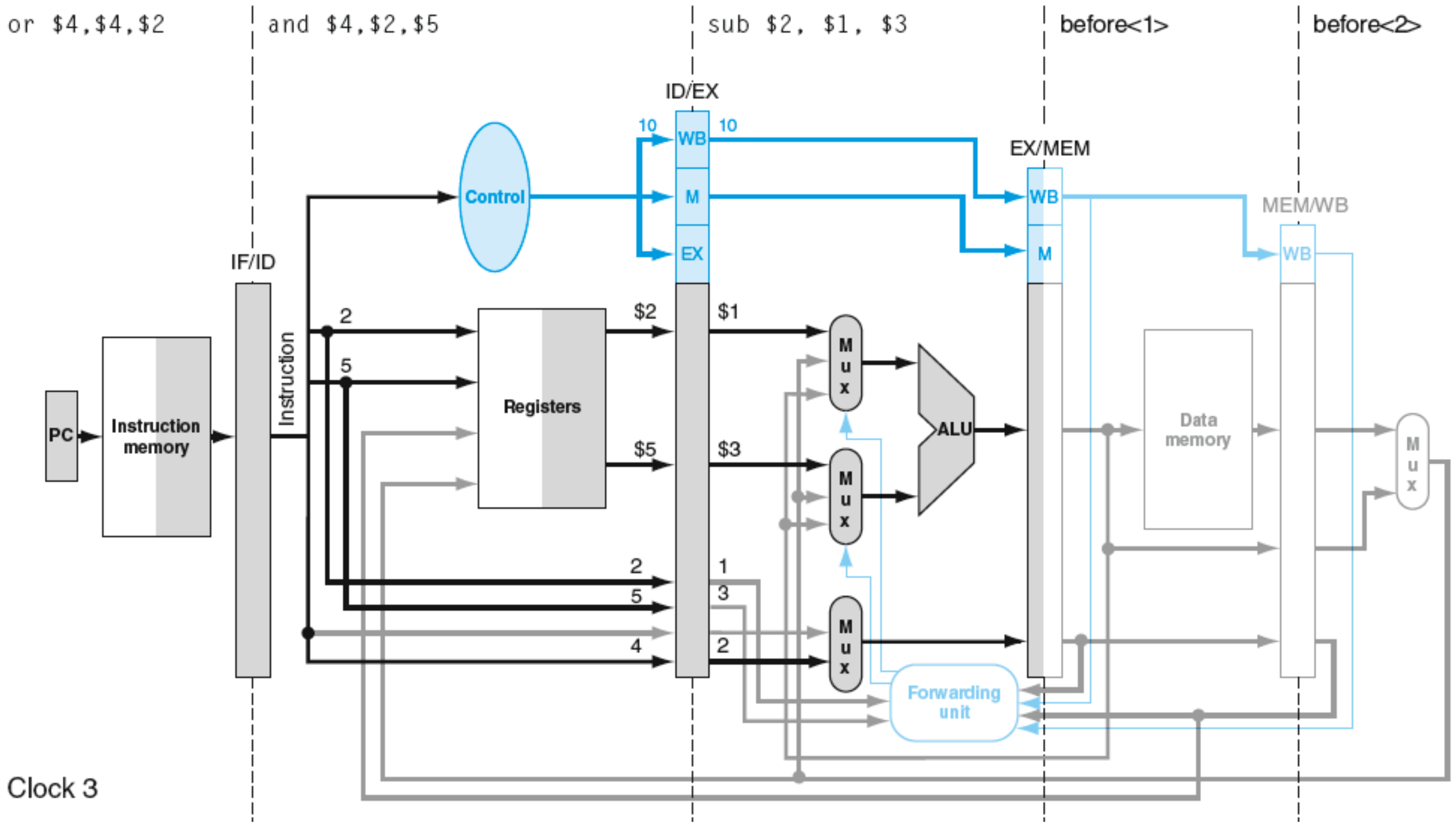
The SUB produces a result in $2 needed by the AND

The result in $4 produced by AND in needed by the OR

OR updates $4 which is then used by ADD

Forwarding within the Pipeline

By cycle 3, the pipeline contains:

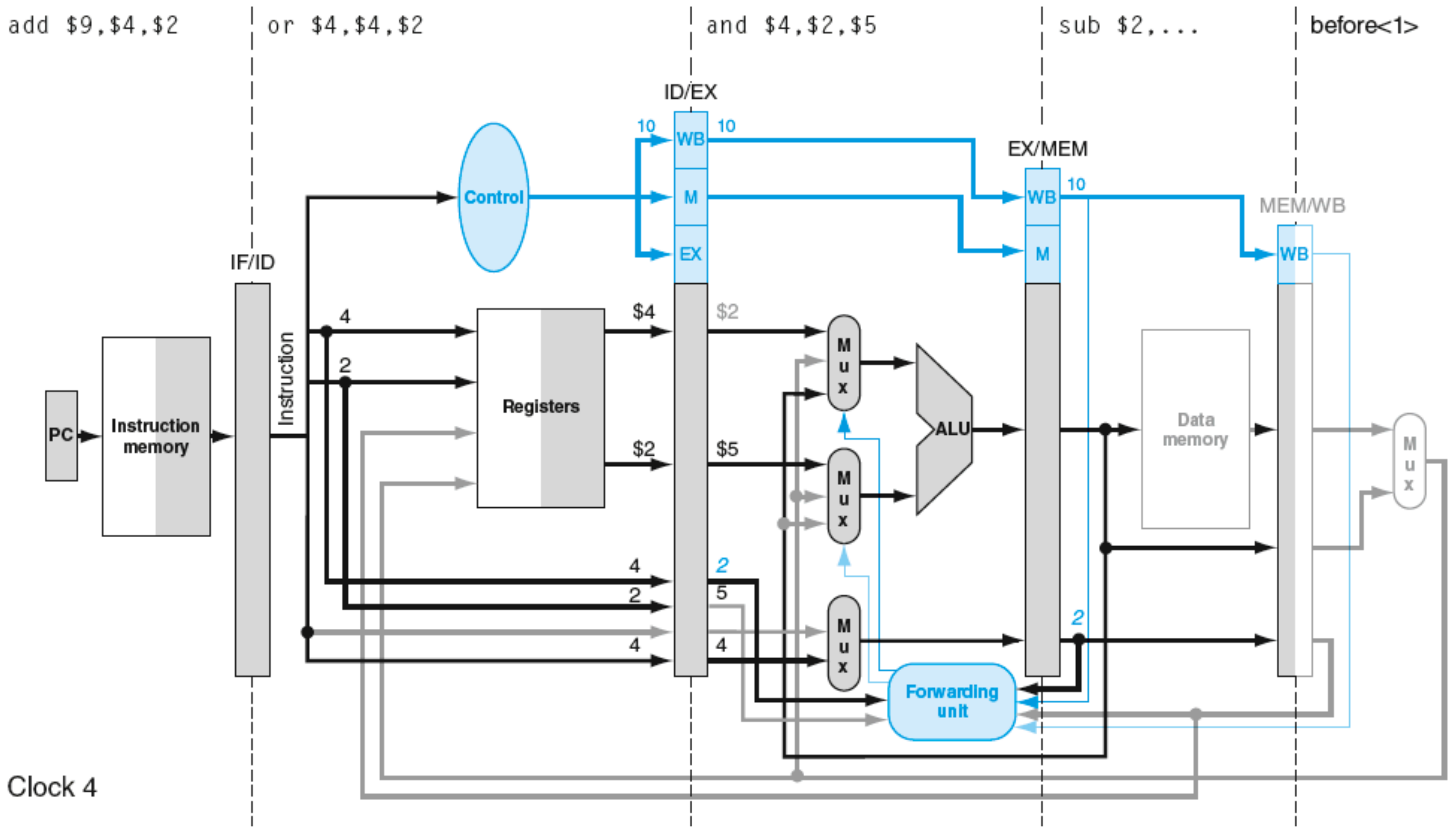


The decode stage is where instructions read their input registers

The write-back stage is where instructions write their result registers

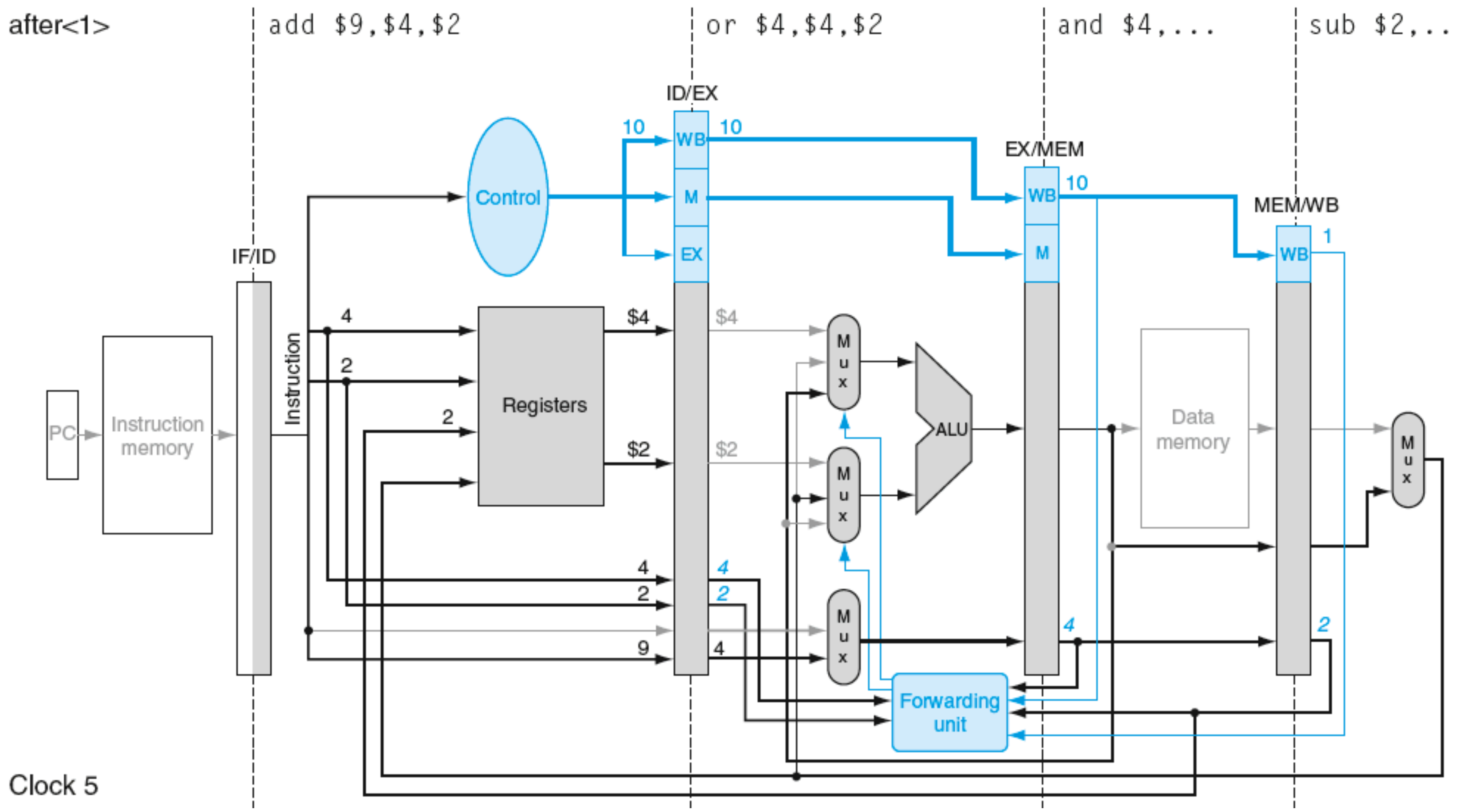
Forwarding within the Pipeline

By cycle 4, the pipeline contains:



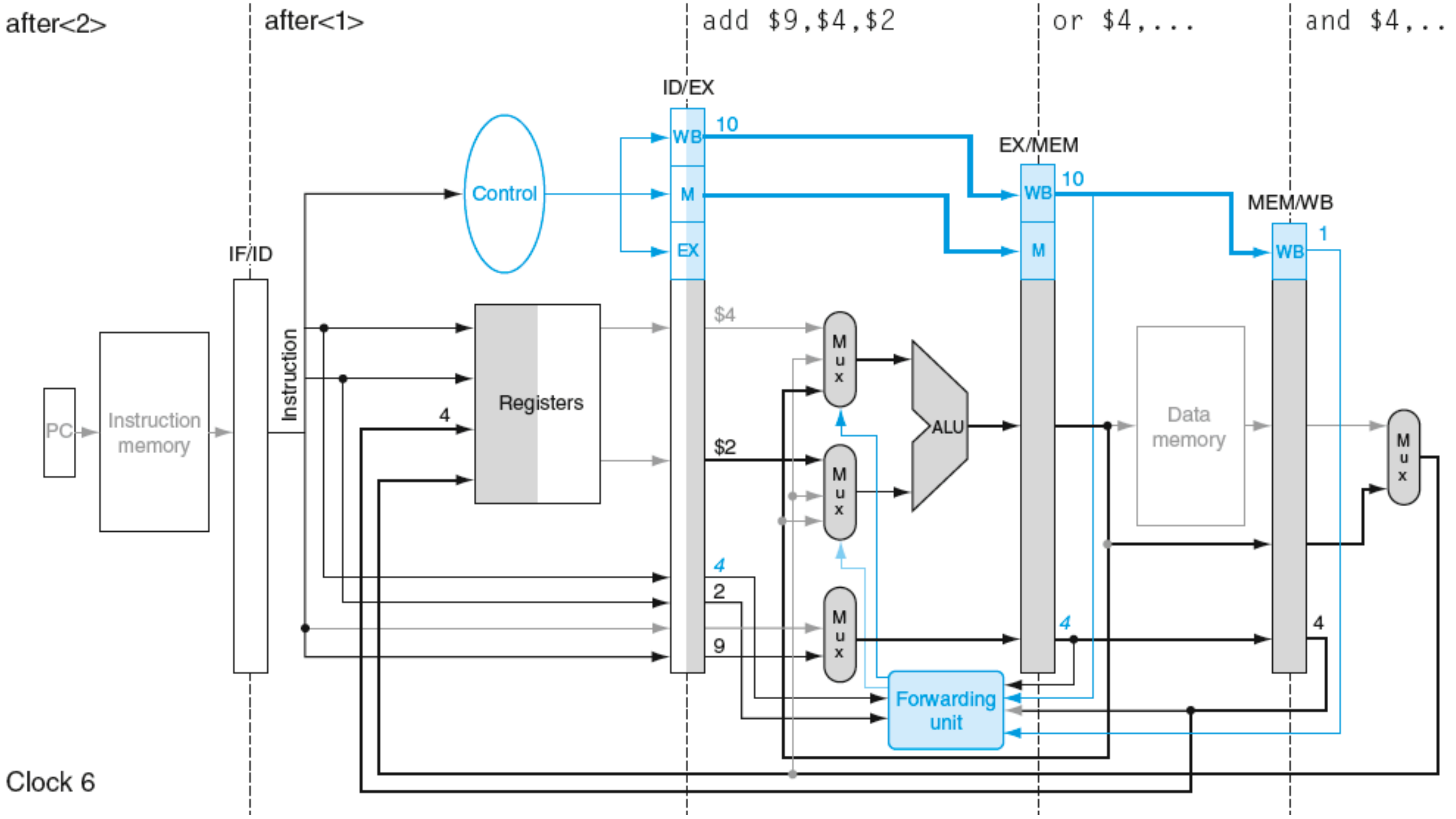
The upper ALU input for the AND is replaced by the forwarded value from EX/MEM ($2)

Forwarding within the Pipeline



The upper ALU input for the OR is replaced by the forwarded value from EX/MEM

Forwarding within the Pipeline



The upper ALU input for the ADD is replaced by the forwarded value from EX/MEM

Forwarding within the Pipeline

With forwarding, fewer cycles are consumed

- The above sequence completes in 8 cycles
- It takes 14 cycles without forwarding

Without forwarding, stalls are needed (bubbles)

- Dependent instruction is held in decode stage
- Instruction producing result must reach stage5
- Register writes occur in first half of clock cycle
- Register reads occur in second half of clock cycle
- Otherwise one extra stall would be needed

JOHNS HOPKINS UNIVERSITY | Engineering for Professionals

Forwarding within the Pipeline

Forwarding avoids having to stall dependent instructions

- By just-in-time replacement of stale ALU inputs

Some instructions cannot cause data hazards

- These instruction don’t write a result register
- Sw writes to memory, not to a register
- Beq compares 2 registers without writing either