Load Delays

Forwarding avoids having to stall dependent instructions

- Except for values read from memory
- *Delay slot* is the slot following a load instruction
- Instructions in the delay slot can't use the load result
- otherwise they must be stalled for one cycle

Hazard detection unit is still required for this special case

Load Delays

This sequence illustrates the effect of a load delay:

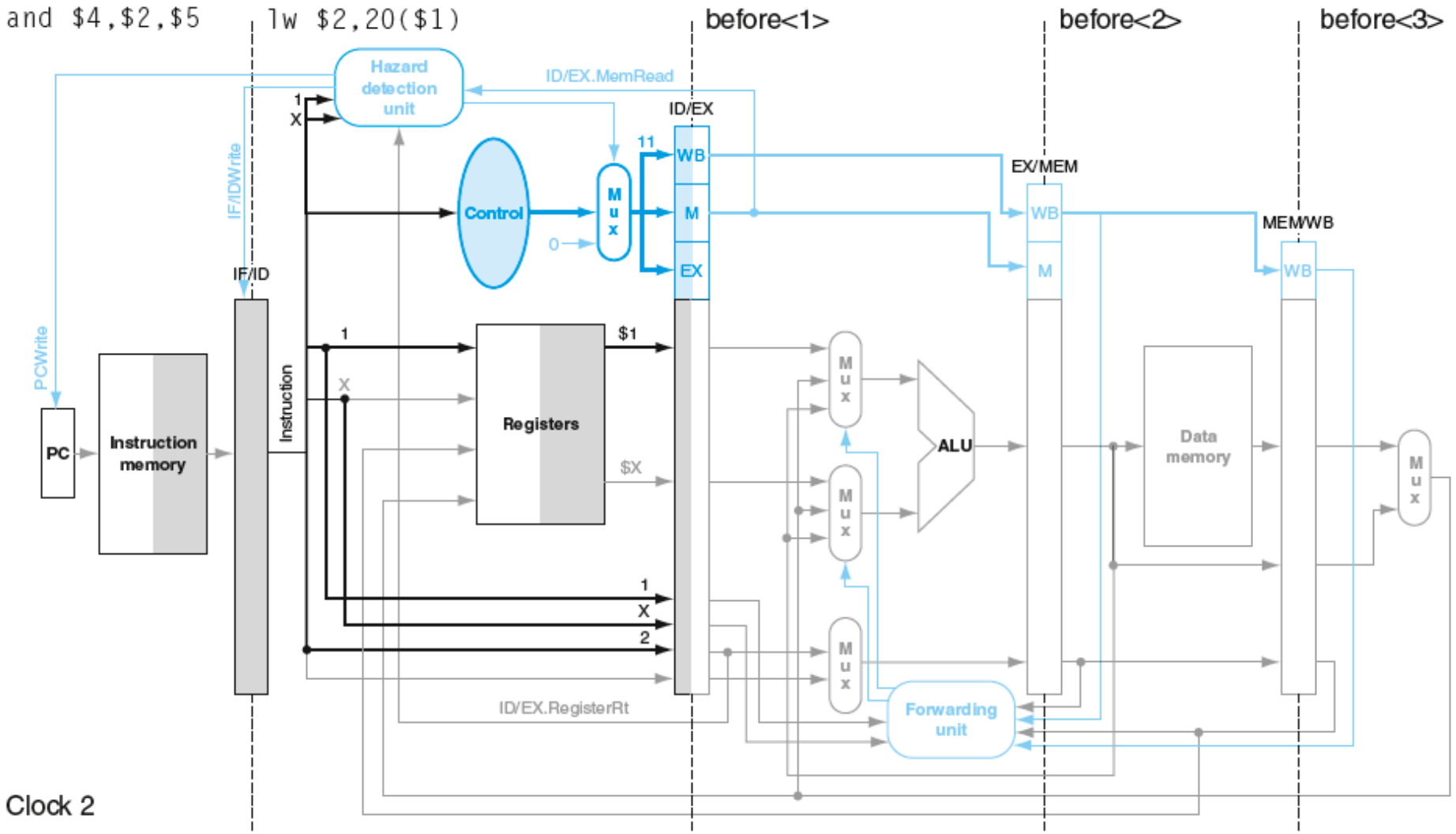
```
lw      $2, 20($1)
and     $4, $2,$5
or      $4, $4,$2
add     $9, $4,$2
```

The AND needs the result in $2 read from memory by LW

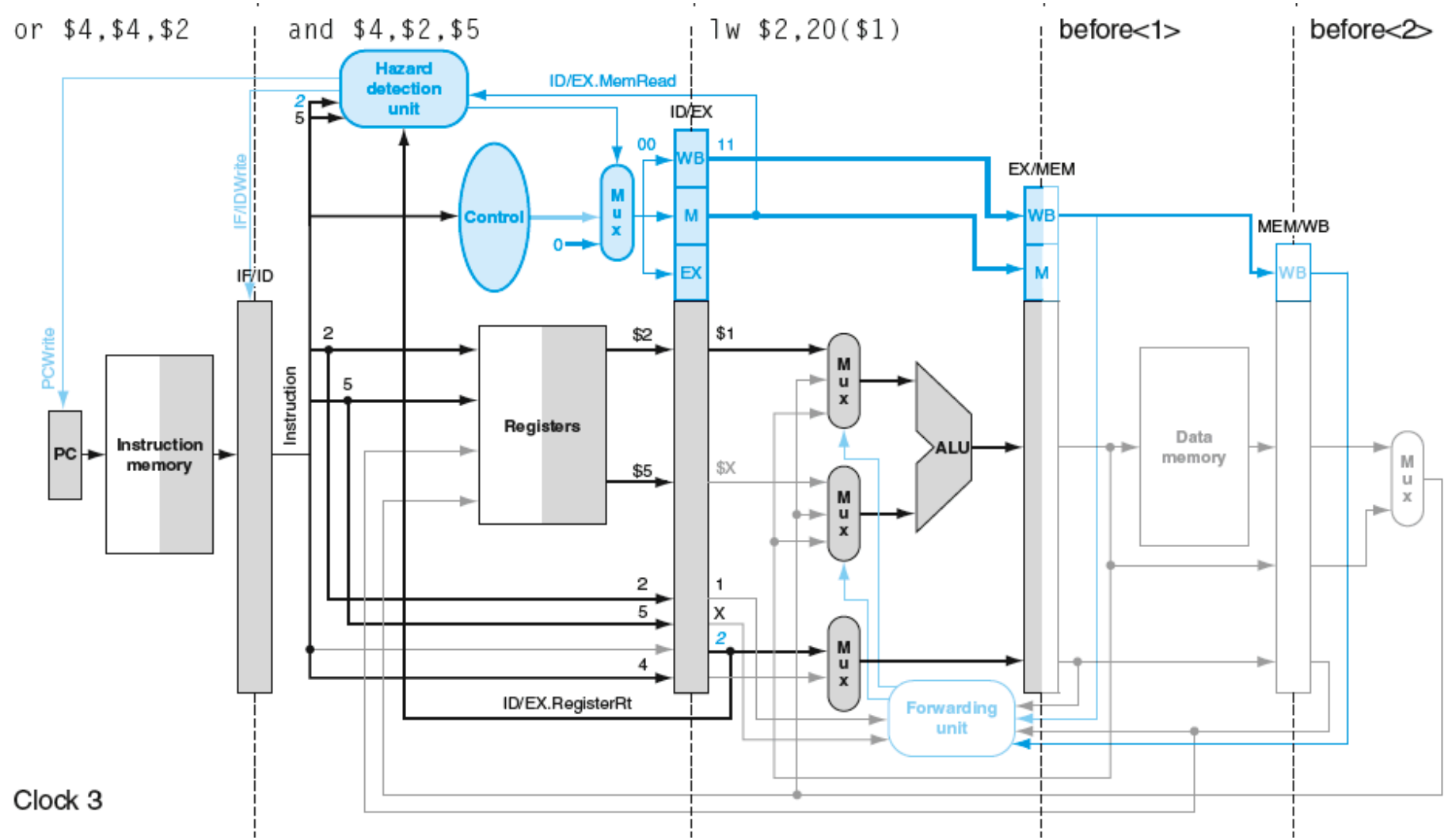
The result in $4 produced by AND in needed by the OR

OR updates $4 which is then used by ADD

Load Delays

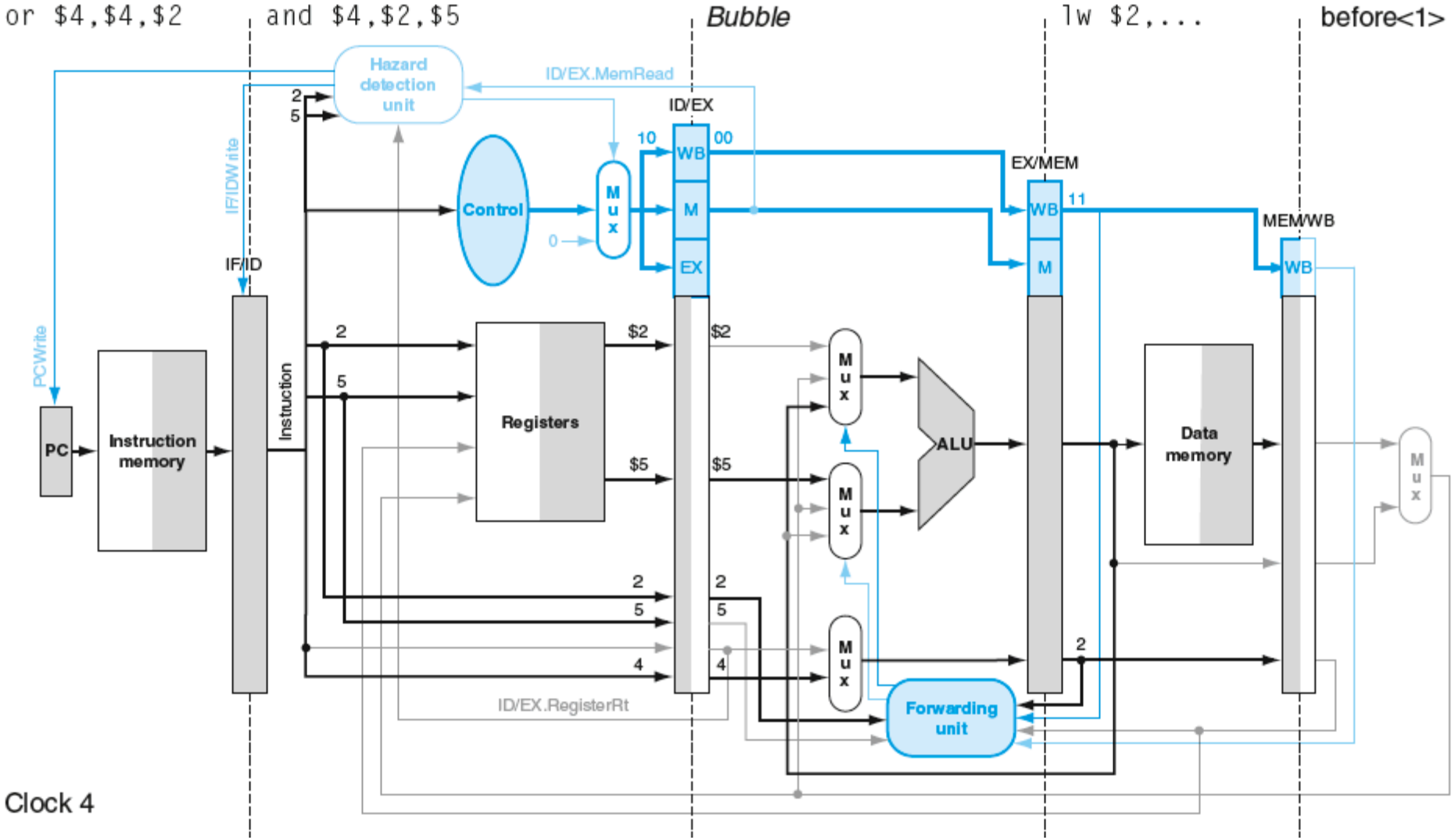

and $4,$2,$5
lw $2,20($1)
before<1>
before<2>
before<3>
Hazard detection unit
ID/EX.MemRead
ID/EX
IF/IDWrite
Control
Mux
WB
M
EX
EX/MEM
MEM/WB
IF/ID
PCWrite
PC
Instruction memory
Instruction
Registers
$1
$X
ALU
Data memory
Forwarding unit
ID/EX.RegisterRt
Clock 2

Load Delays



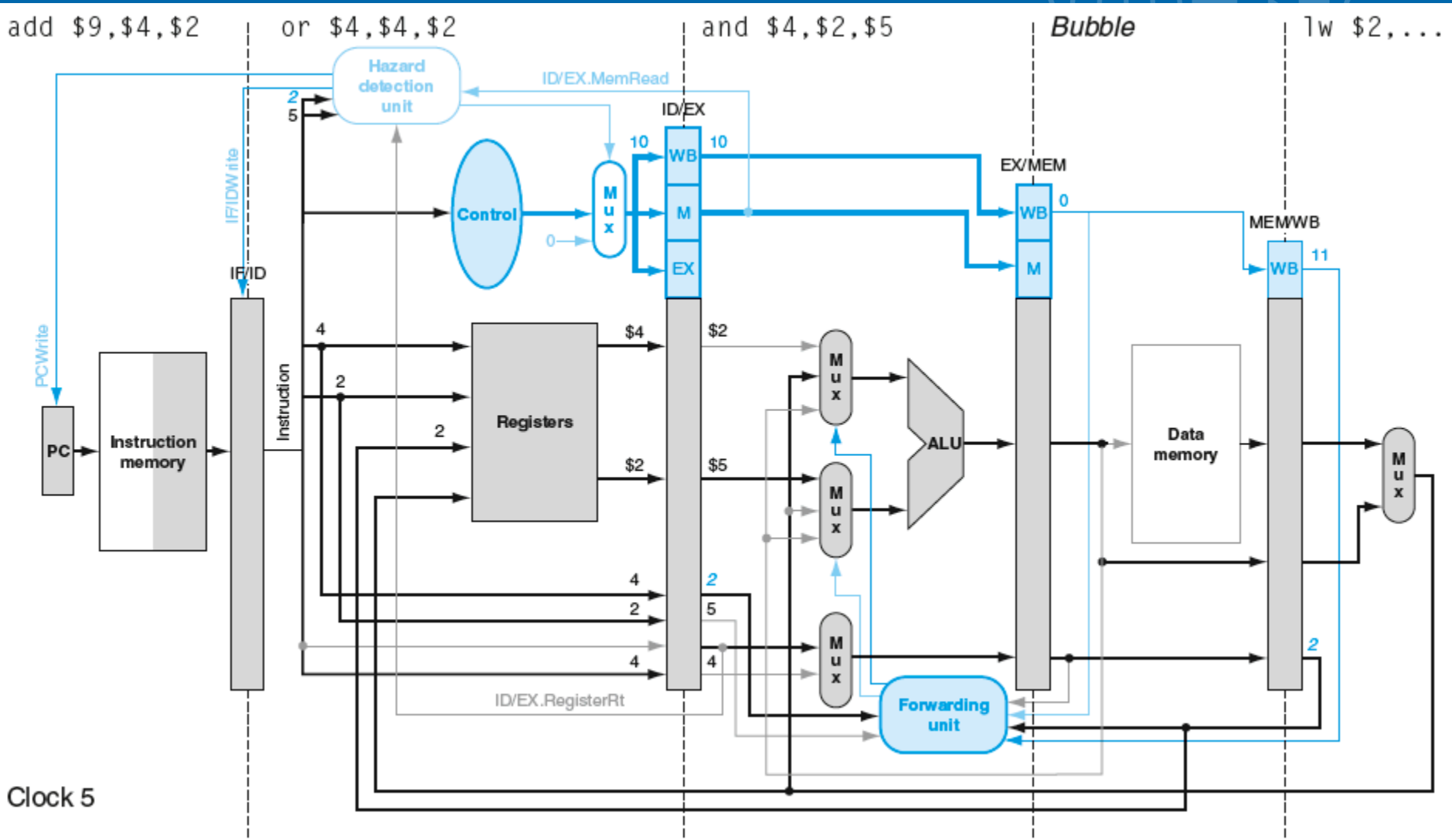
AND uses $2, so it must be stalled to allow LW time to read from the data memory

Load Delays



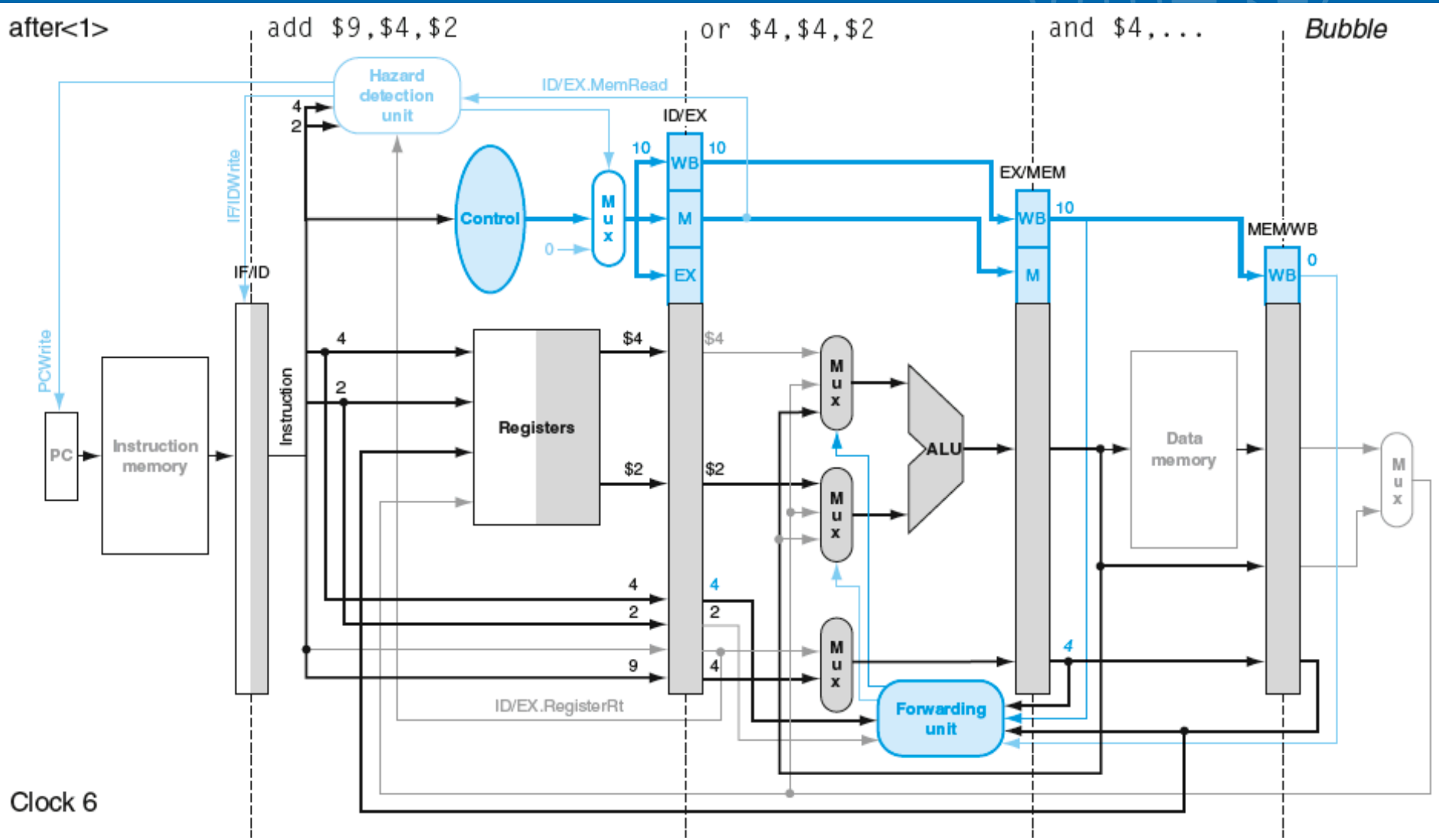
Bubble occupies stage 3 while LW reads from the data memory

Load Delays



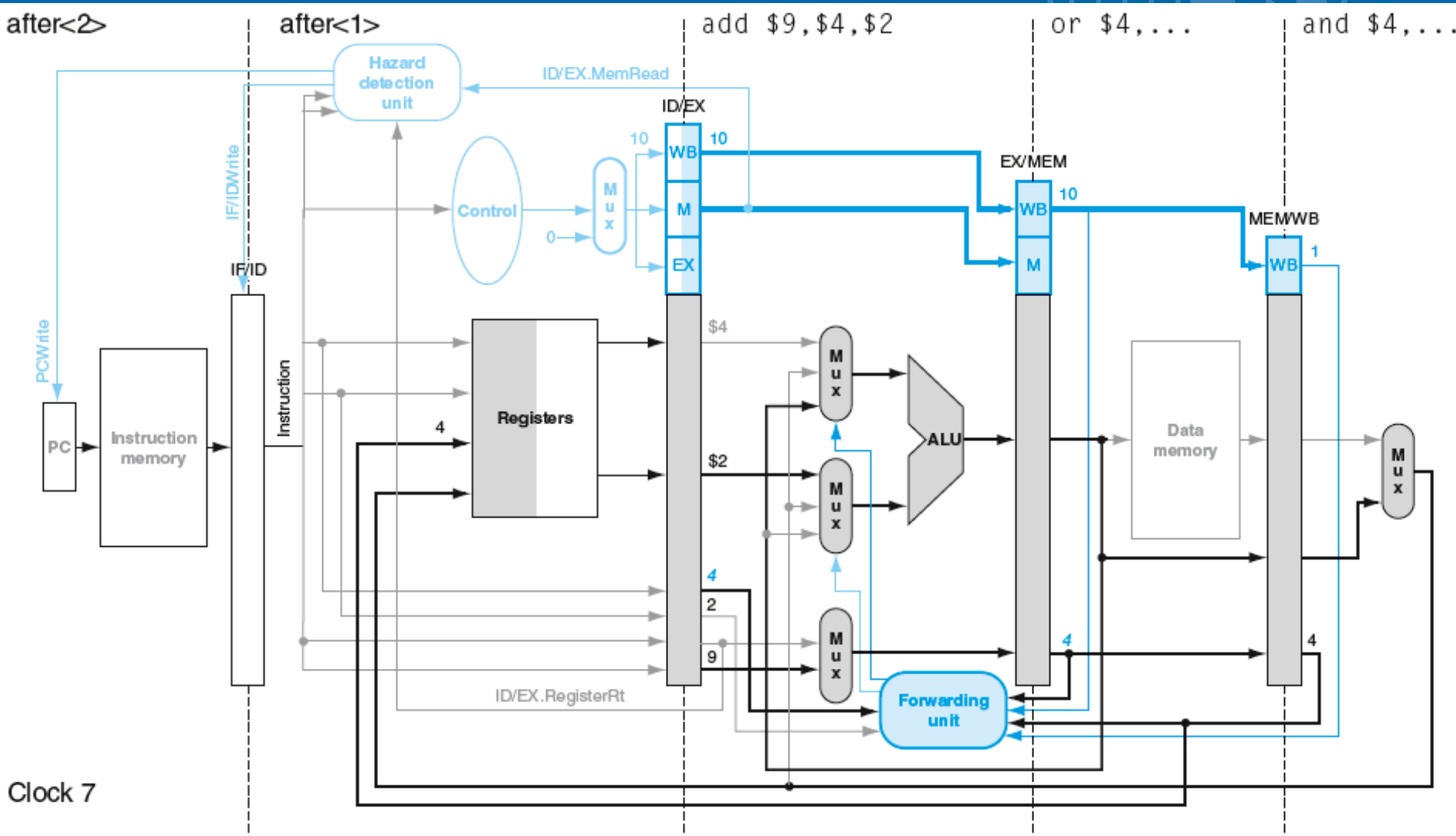
AND receives forwarded value from MEM/WB pipeline register in cycle 5

Load Delays



Forwarding takes care of the dependencies for the OR and ADD instructions.

Load Delays



All instructions complete by cycle 9.

Load Delays

The stall due to load delay added one extra cycle

Some compilers rearrange machine instructions

- This can avoid extra cycles due to stalls
- Rearrangement must not change program behavior
- Assembly language programmers can fill delay slot
- NOP instruction can be used as last resort