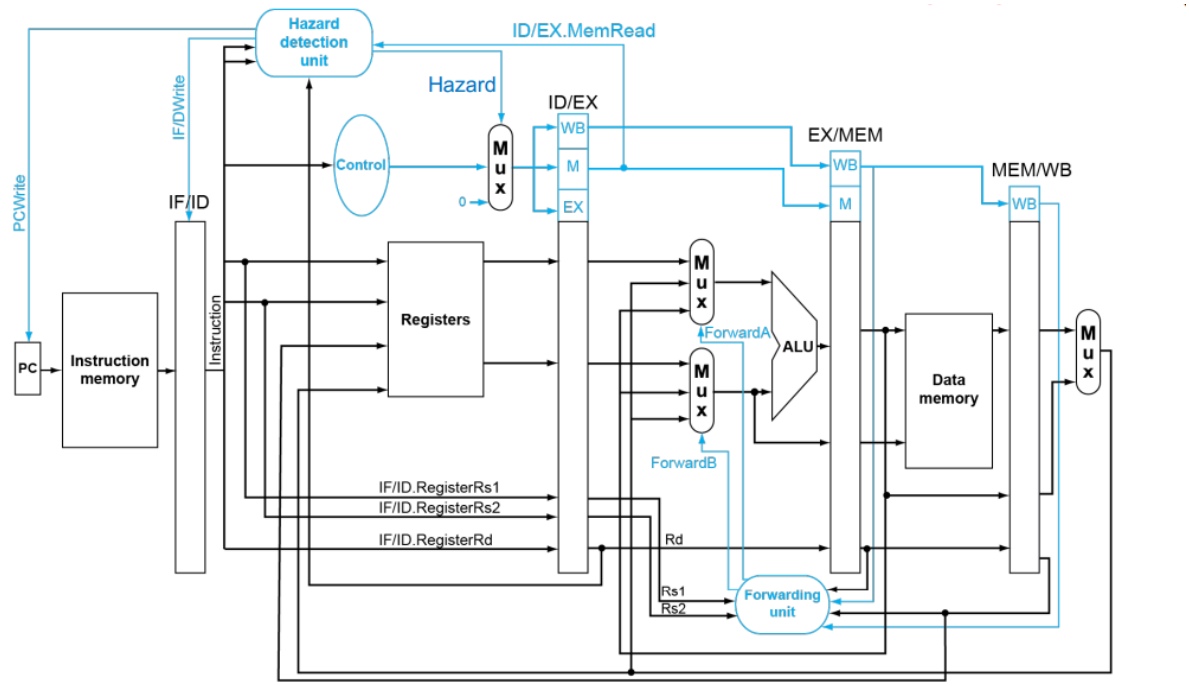
T7 Data Hazard

Different Type of Hazards

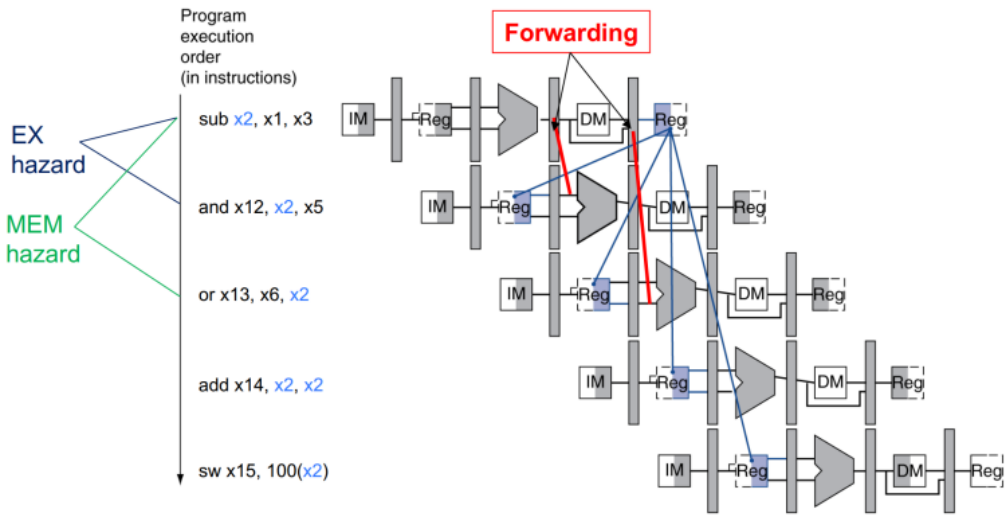
- Data hazards: Need to wait for previous instruction to complete its data read/write
- Control hazards: Decision on control action depends on previous instruction
- Structure hazards: A required resource is busy

The Whole Picture



Categories of Data Hazards

1. EX/MEM Data Hazards



EX hazard:

(1) Data hazard between two adjacent instructions

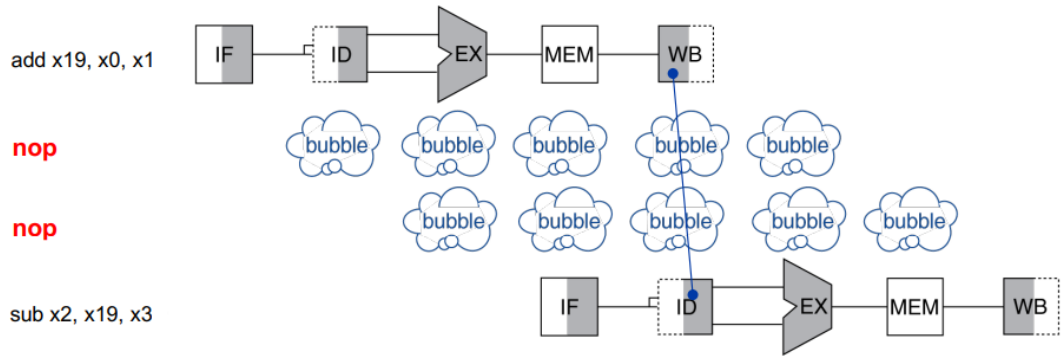
(2) Forwarding path from EX/MEM pipeline register to ALU

MEM hazard:

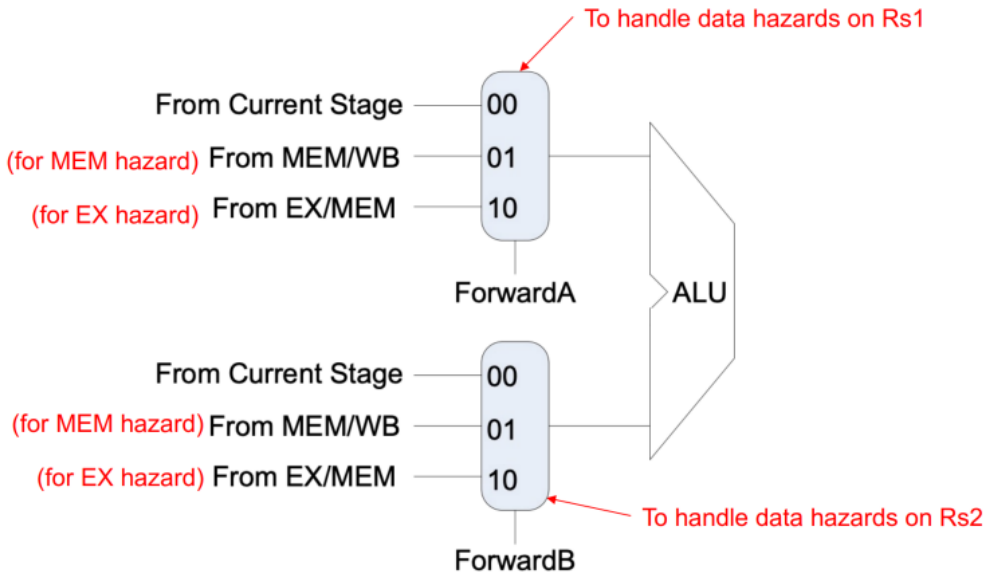
(1) Data hazard between two instructions with one more instruction in between

(2) Forwarding path from MEM/WB pipeline register to ALU

- **Solution 1: Stall**



- **Solution 2: Forwarding**



The control signal `ForwardA` and `ForwardB` is generated from **Forwarding unit**.

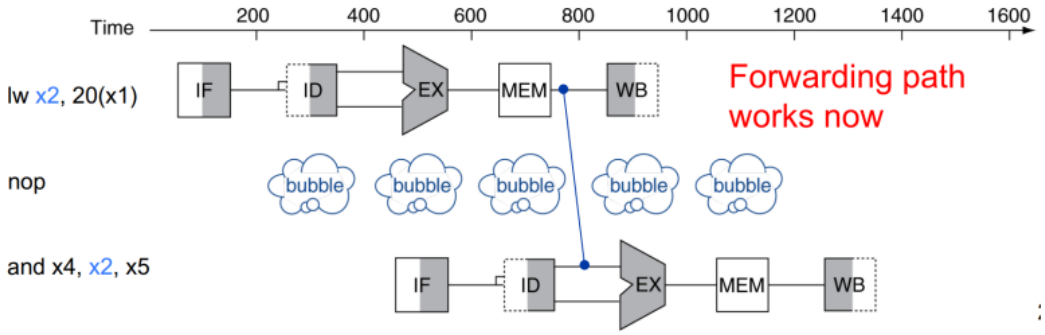
Logic in Forwarding unit:

```
# EX hazard
if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd == ID/EX.RegisterRs1))
        ForwardA = 10
if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd == ID/EX.RegisterRs2))
        ForwardB = 10

# MEM hazard
```

```
if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd == ID/EX.RegisterRs1)
    and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
        and (EX/MEM.RegisterRd == ID/EX.RegisterRs1)) )
            ForwardA = 01

if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd == ID/EX.RegisterRs2)
    and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
        and (EX/MEM.RegisterRd == ID/EX.RegisterRs2)) )
            ForwardB = 01
```

2. Load-Use Data Hazards

For load-use data hazards, forwarding cannot fix it completely. Need to stall for one cycle.



We will use **hazard detection unit** here to detect whether we need to insert a stall.

- Logic in hazard detection unit:

```
if ID/EX.MemRead and ((ID/EX.RegisterRd = IF/ID.RegisterRs1)
                      or (ID/EX.RegisterRd = IF/ID.RegisterRs2))
```

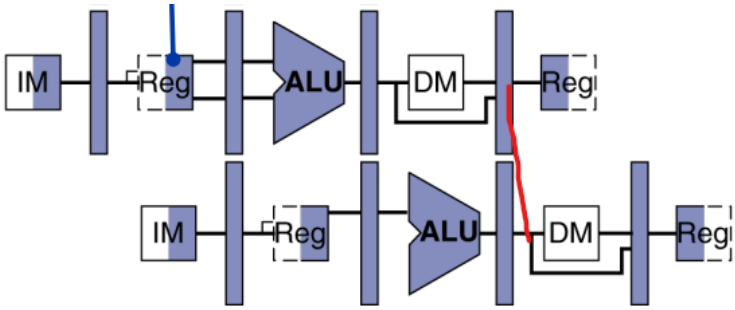
- Logic of stalls
  - Force control signals in ID/EX pipeline register to 0's
    - EX, MEM and WB in following cycles do no-operation (nop) with those 0 control signal
  - Prevent updates of PC and IF/ID registers
    - Instruction in IF/ID is held and decoded again
    - PC is held, so the same instruction is fetched again

3. Other Hazards:

Example:

```
lw x10, 20(x1)
sw x10, 40(x1)
```

Can be solved by a new forwarding path:



Modification on hardware: