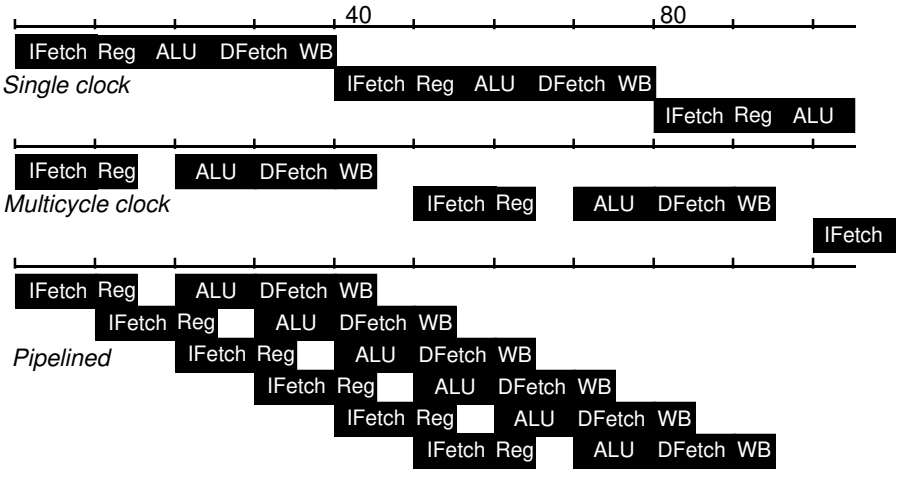
Pipelining

Pipelining is a standard technique for exploiting parallelism in tasks that are ordered. Two considerations are: Balancing the step size and avoiding hazards

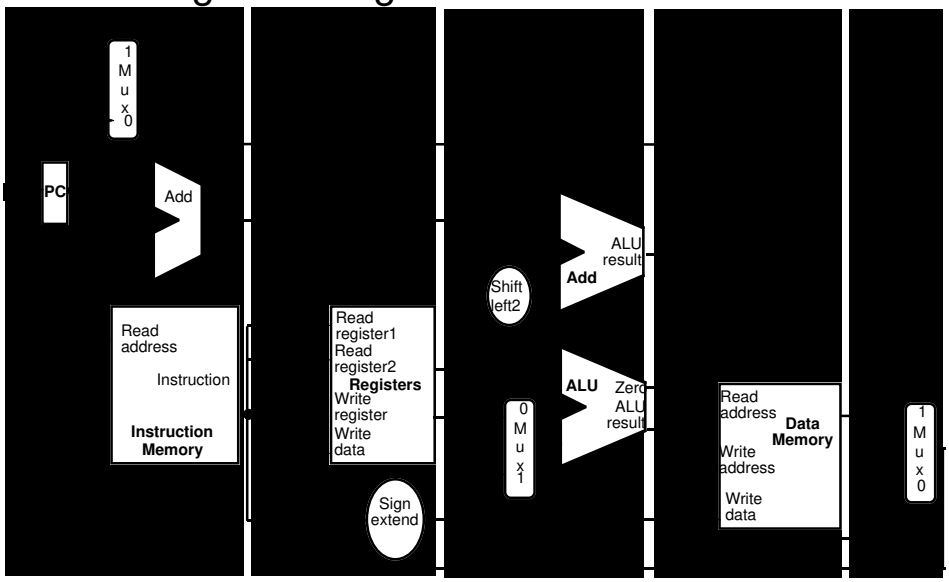
Benefits of Pipelining

Assume:
Memory(10ns), ALU(10ns), Register(5ns),
All other operations are 0ns.

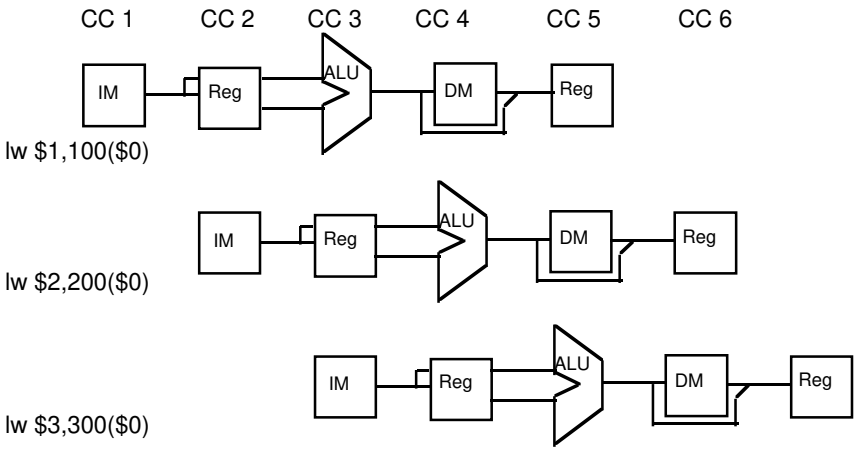
Charges for instructions are ...
- –R-format: 30ns
- –Load inst: 40ns
- –Store inst: 35ns
- –Branch: 25ns
- –Jump: 10ns



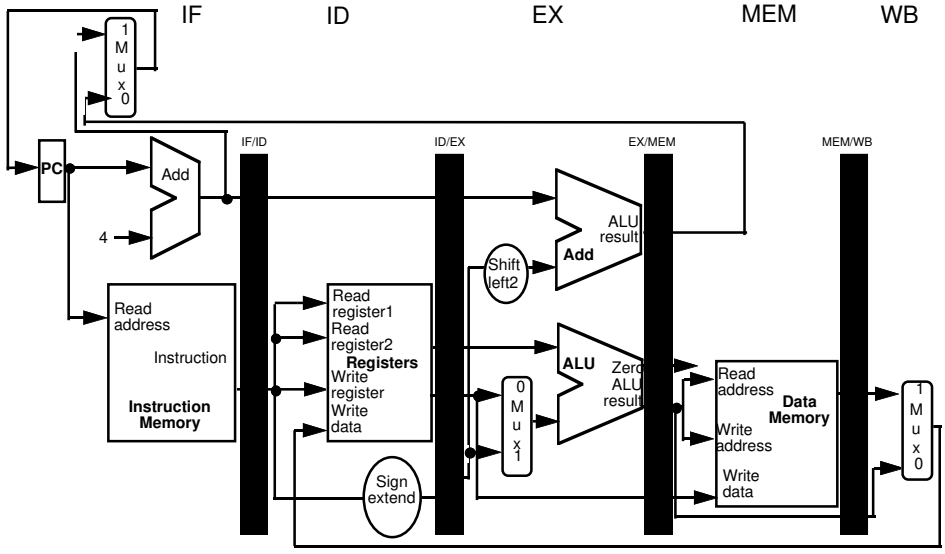
Dividing Into Stages



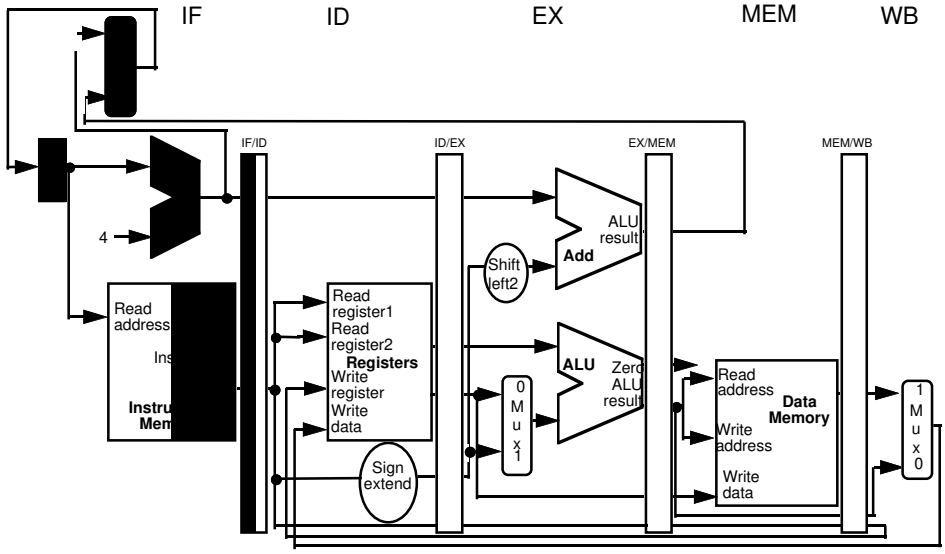
Pipelined Datapath Schematic



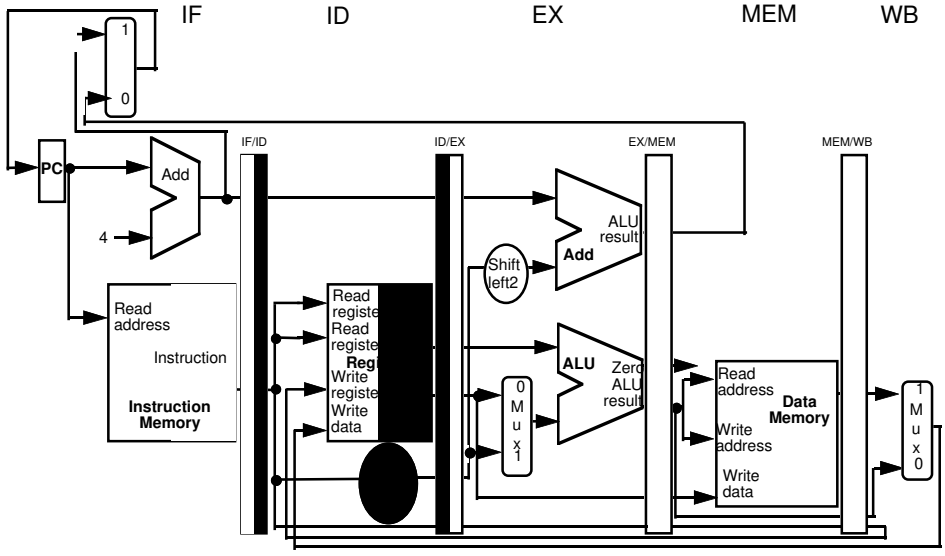
Pipeline Registers



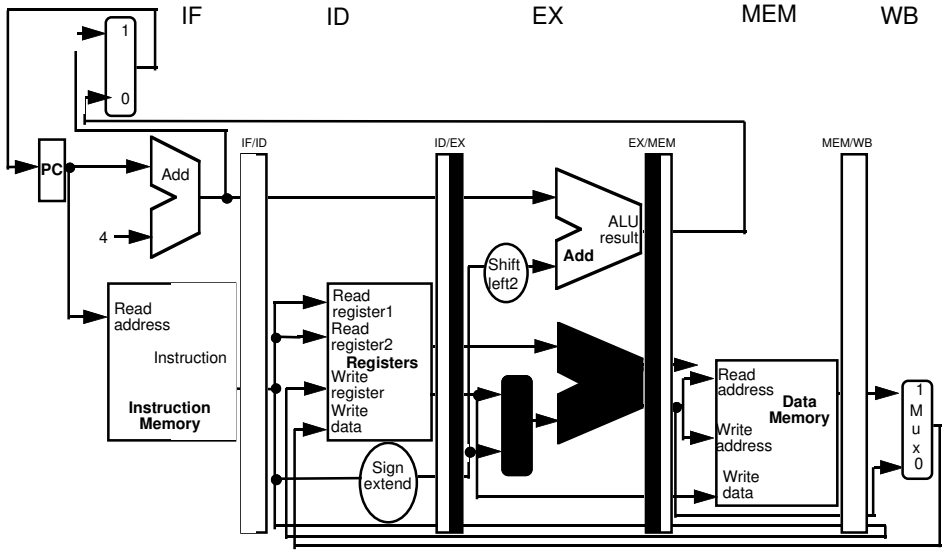
Instruction Fetch



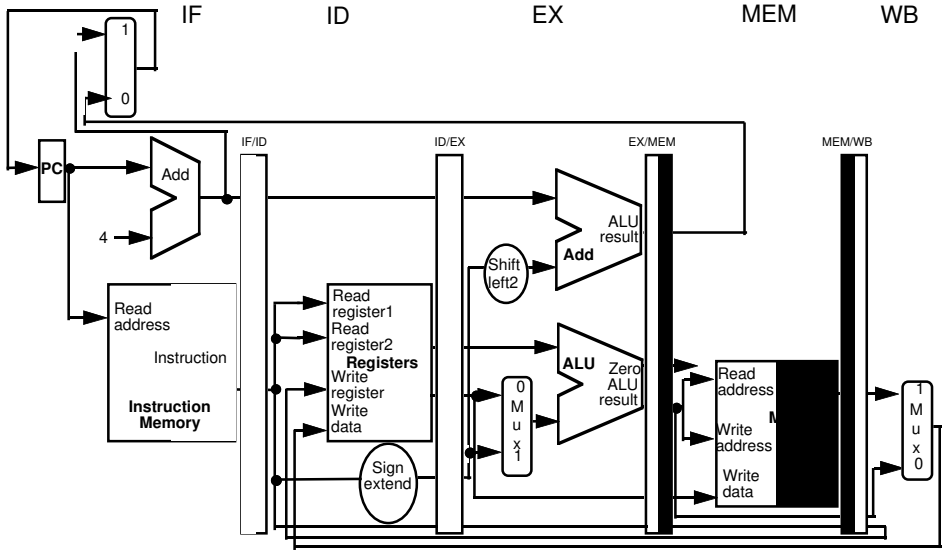
Instruction Decode



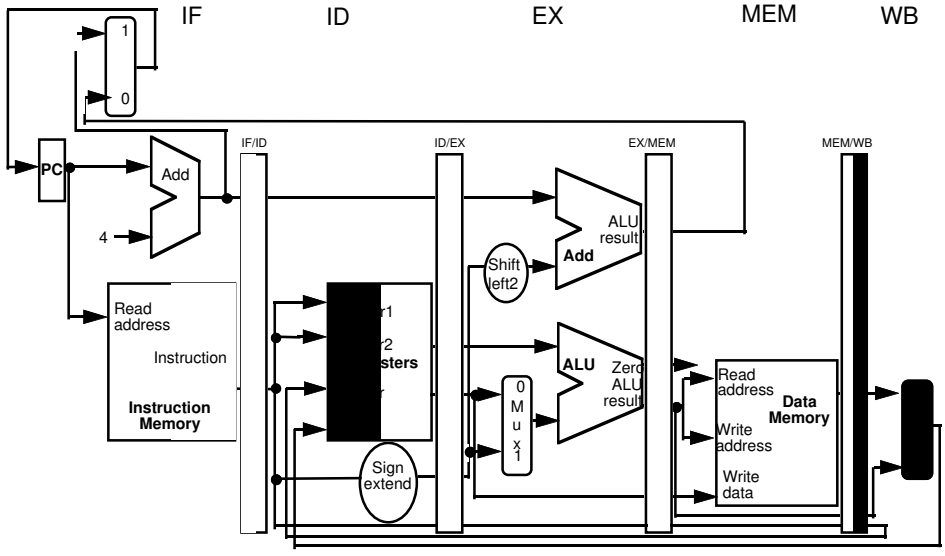
Execution (LW)



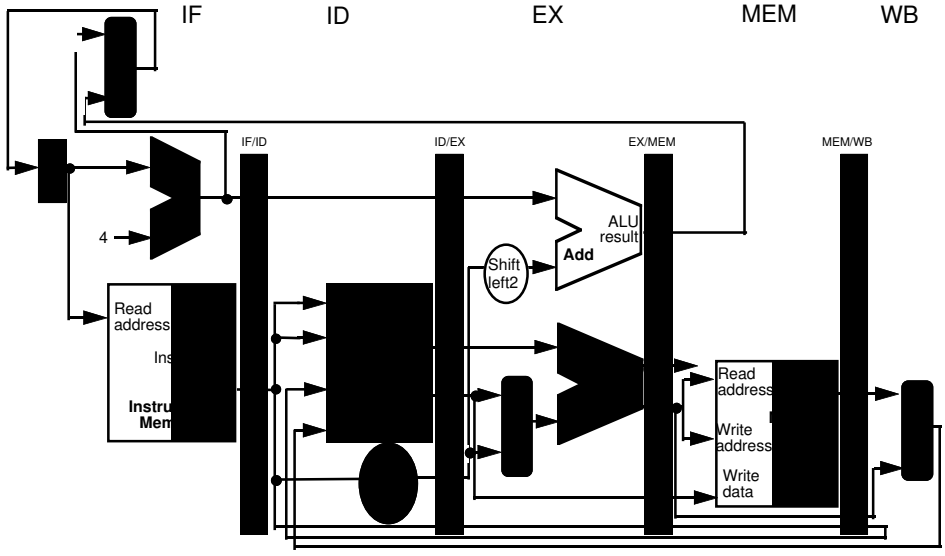
Memory (LW)



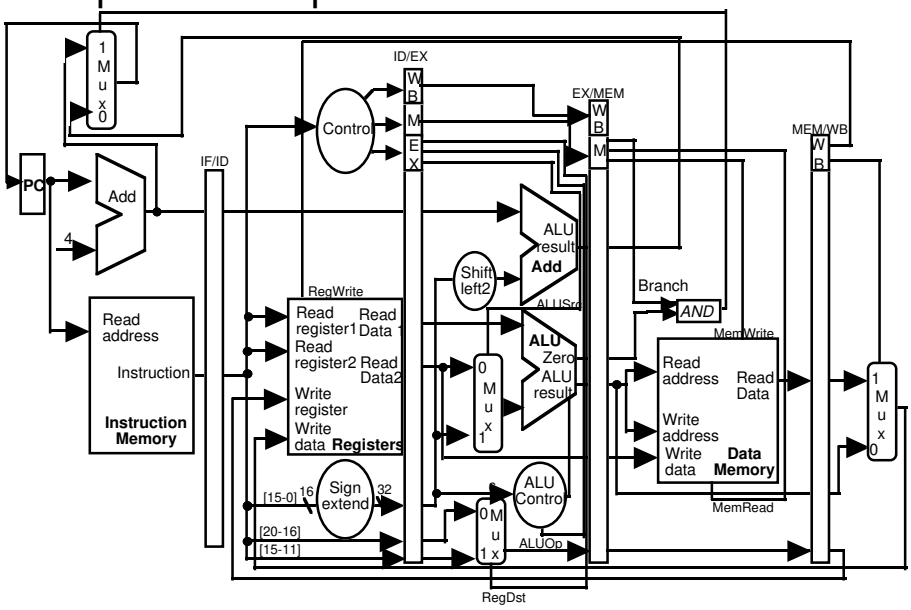
Write Back (LW)

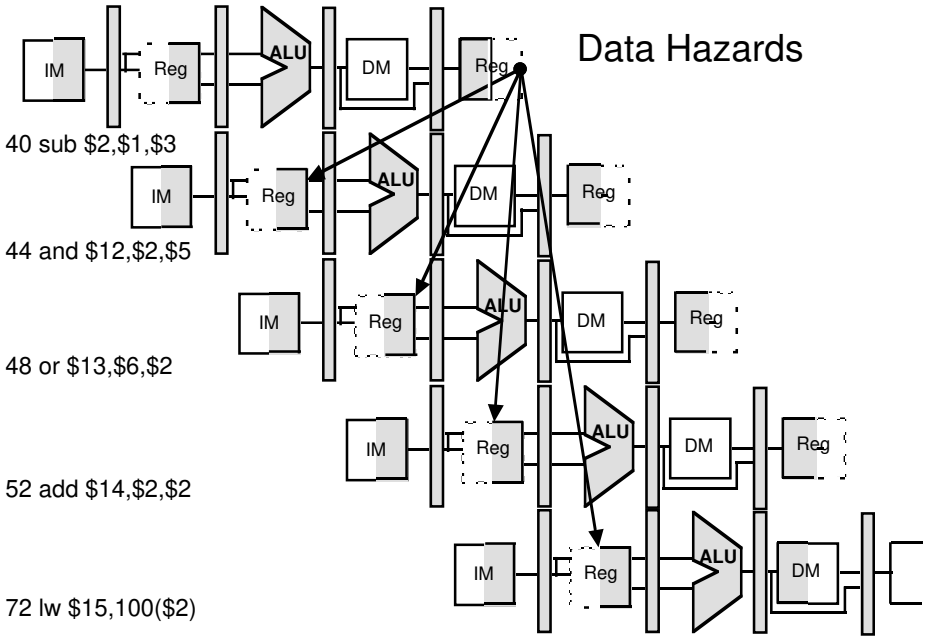


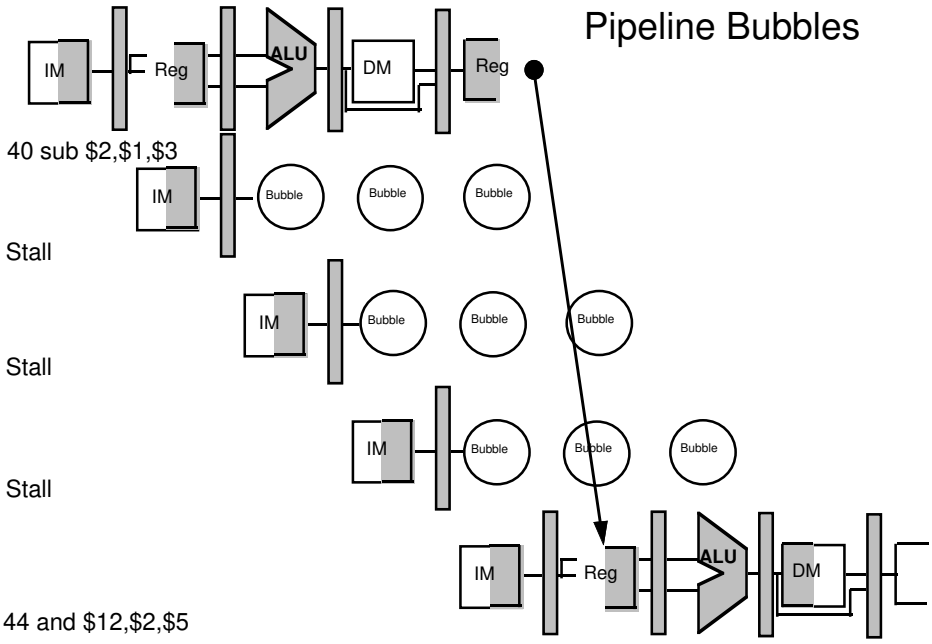
Summary (LW)



Pipeline Datapath Control




Data Hazards
IM
Reg
ALU
DM
Reg
40 sub $2,$1,$3
44 and $12,$2,$5
48 or $13,$6,$2
52 add $14,$2,$2
72 lw $15,100($2)


Pipeline Bubbles
IM
Reg
ALU
DM
Reg
40 sub $2,$1,$3
Bubble
Stall
Stall
Stall
44 and $12,$2,$5

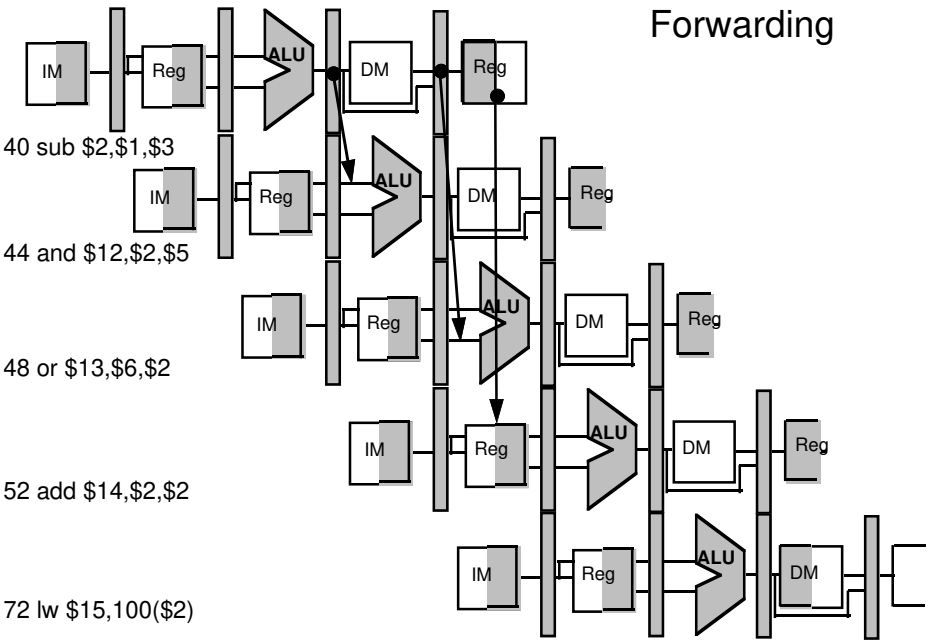
Detecting Data Hazards

EX Hazard:

ID/EX.RegWrite and
((ID/EX.RegDst=0 and ID/EX.WriteRegisterRt=IF/ID.ReadRegister1)or
(ID/EX.RegDst=1 and ID/EX.WriteRegisterRd=IF/ID.ReadRegister1)or
(ID/EX.RegDst=0 and ID/EX.WriteRegisterRt=IF/ID.ReadRegister2)or
(ID/EX.RegDst=1 and ID/EX.WriteRegisterRd=IF/ID.ReadRegister2))

MEM Hazard:

EX/MEM.RegWrite and
((EX/MEM.WriteRegister=IF/ID.ReadRegister1)or
(EX/MEM.WriteRegister=IF/ID.ReadRegister2))

WB Hazard:

MEM/WB.RegWrite and
((MEM/WB.WriteRegister=IF/ID.ReadRegister1)or
(MEM/WB.WriteRegister=IF/ID.ReadRegister1))