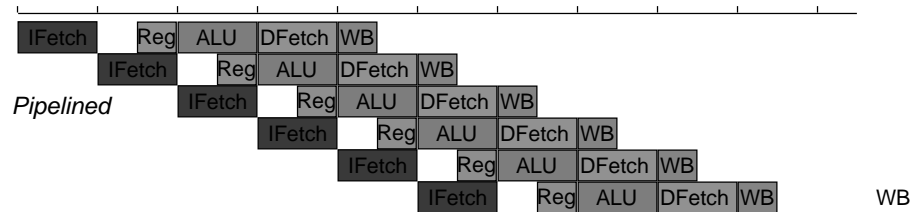


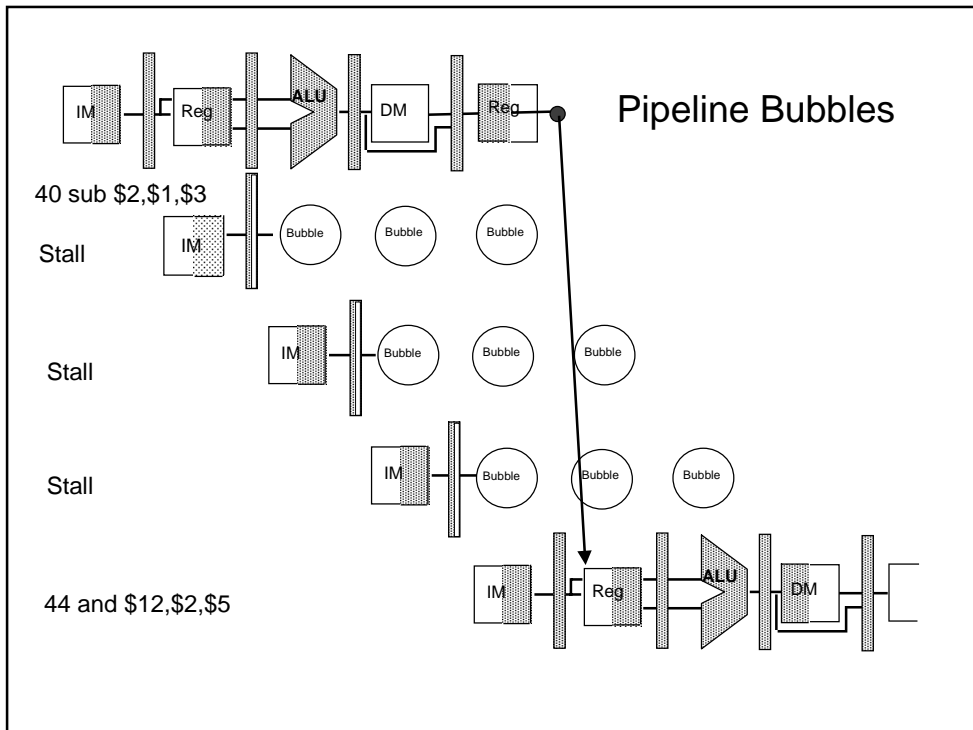
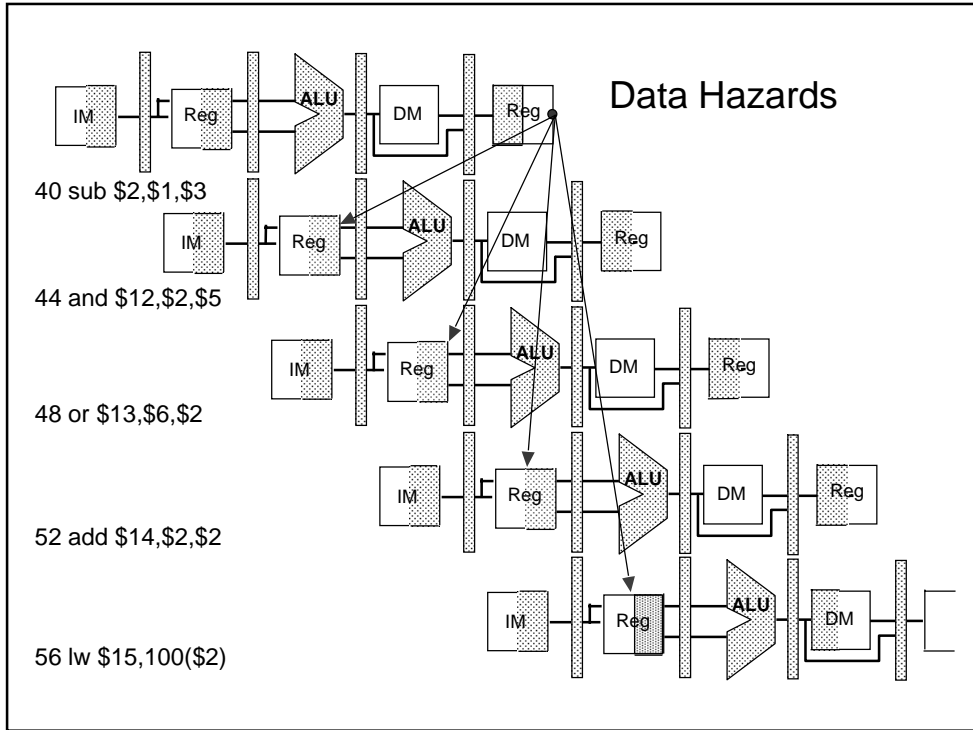
Hazard County

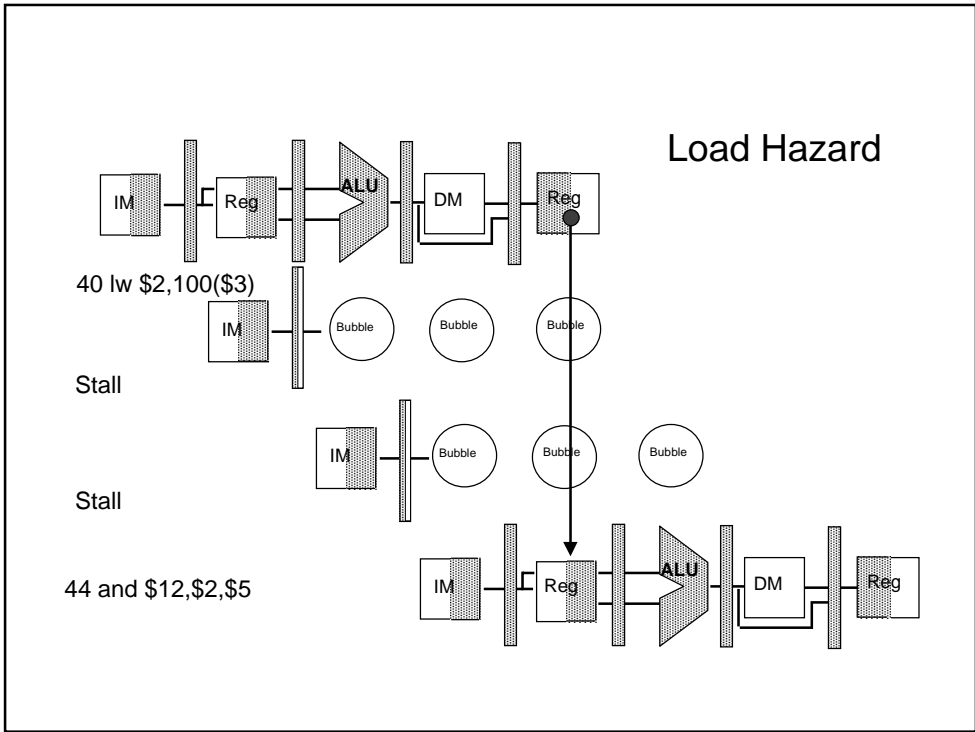
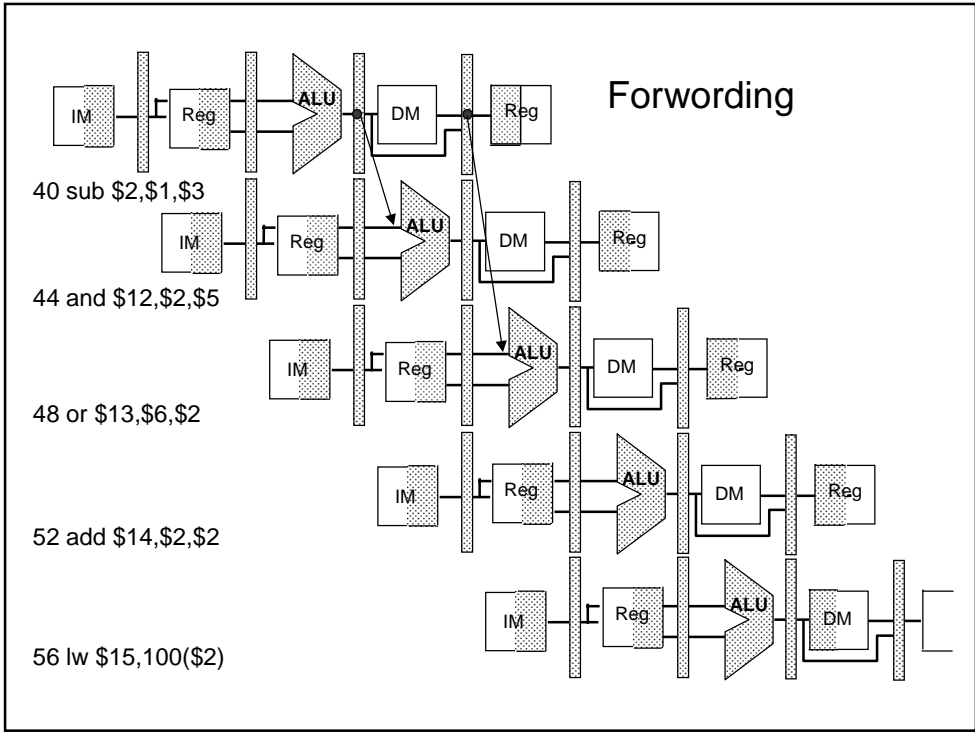
Pipelining is a good idea, but to make it work, several kinds of hazards have to be avoided, and interrupts must be handled.

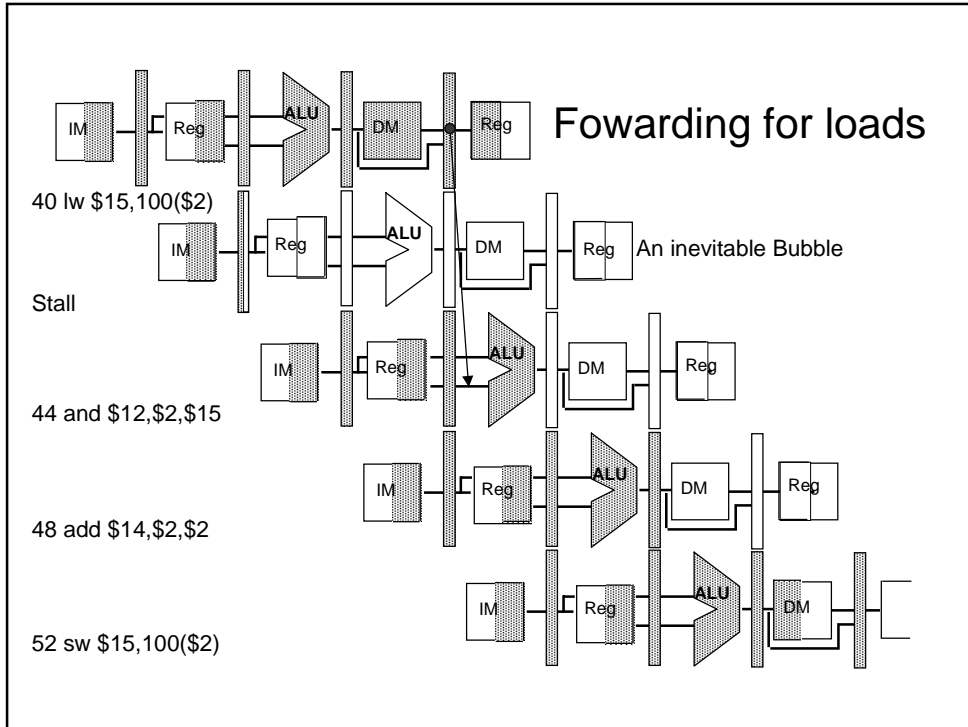
Pipelining

- Recall that pipelining initiates an operation on each clock cycle
- The first design goal is to assure that there are no conflicts in the use of the hardware









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))

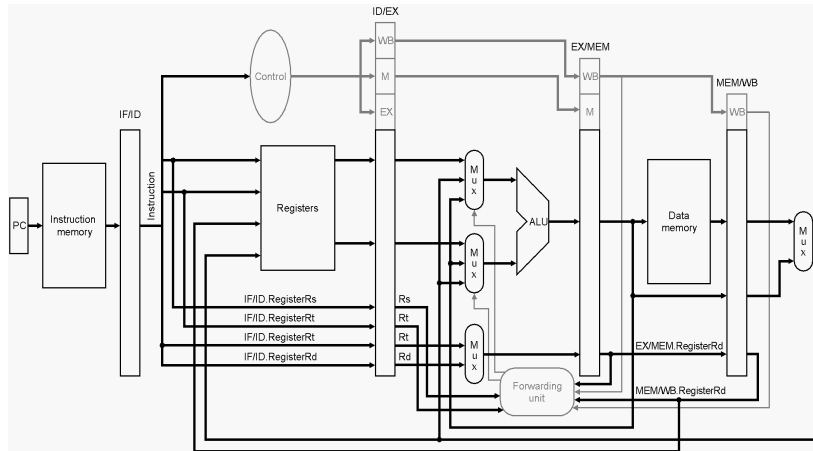
MEM Hazard:

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

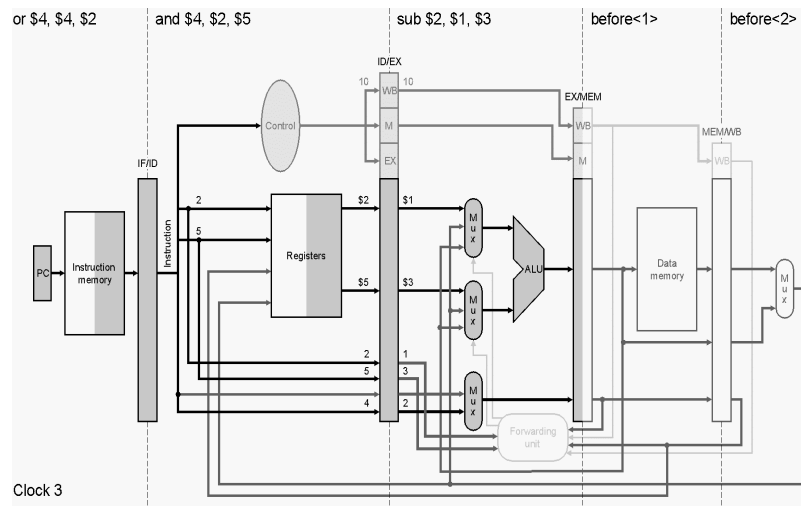
WB Hazard:

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

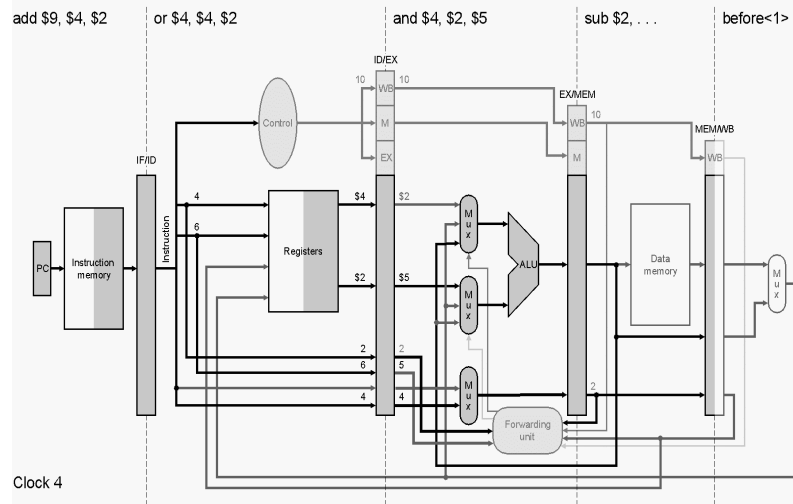
Forwarding



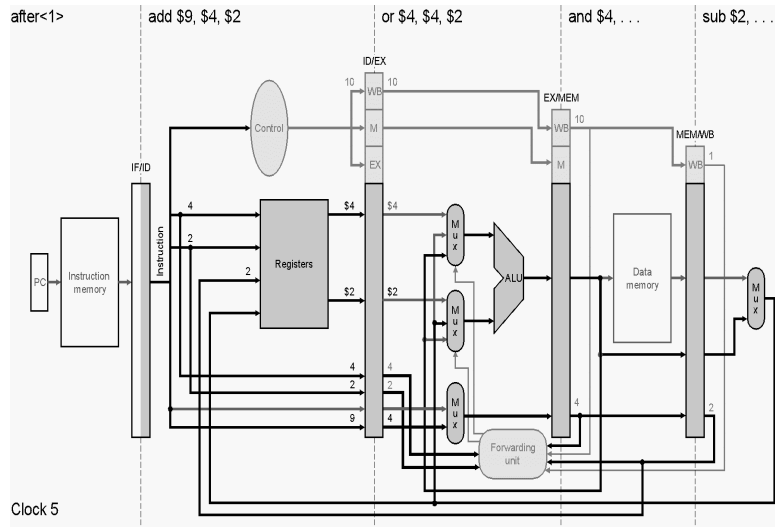
Forwarding ...



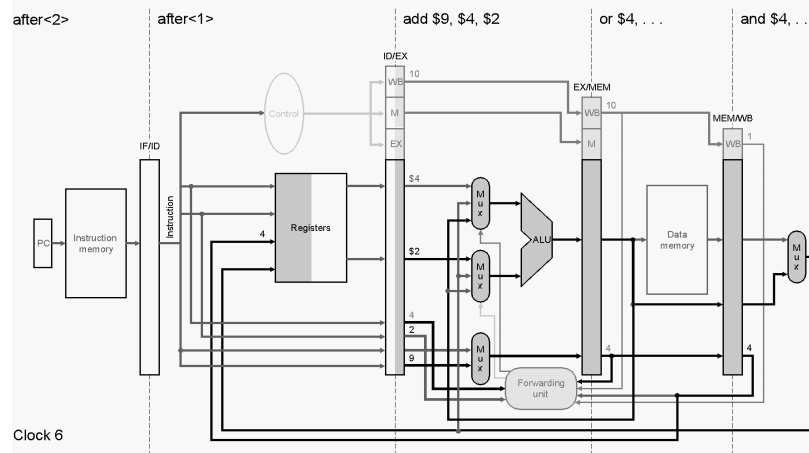
Forwarding ...



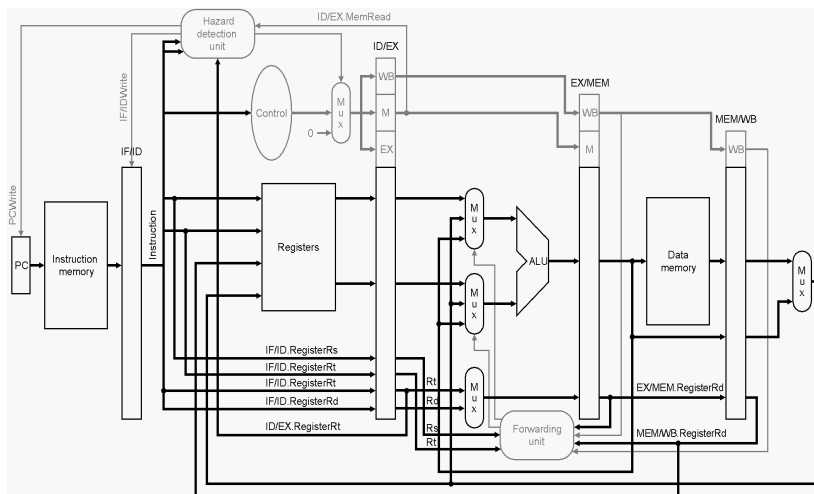
Forwarding ...



Forwarding



Hazard Detection



Load Hazards

ID/EX.RegWrite and (ID/EX.RegDst = 0) and
((ID/EX.WriteRegisterRt = IF/ID.Read.Register1) or
(ID/EX.WriteRegisterRt = IF/ID.ReadRegister2))

Consider the sequence to swap stack items

lw \$10, 0(\$29) -- Load top item

lw \$11, 4(\$29) -- Load second item

sw \$11, 0(\$29) -- Replace top item

sw \$10, 4(\$29) -- Replace second item

Which can be replace by

lw \$10, 0(\$29) -- Load top item

lw \$11, 4(\$29) -- Load second item

sw \$10, 4(\$29) -- Replace second item

sw \$11, 0(\$29) -- Replace top item