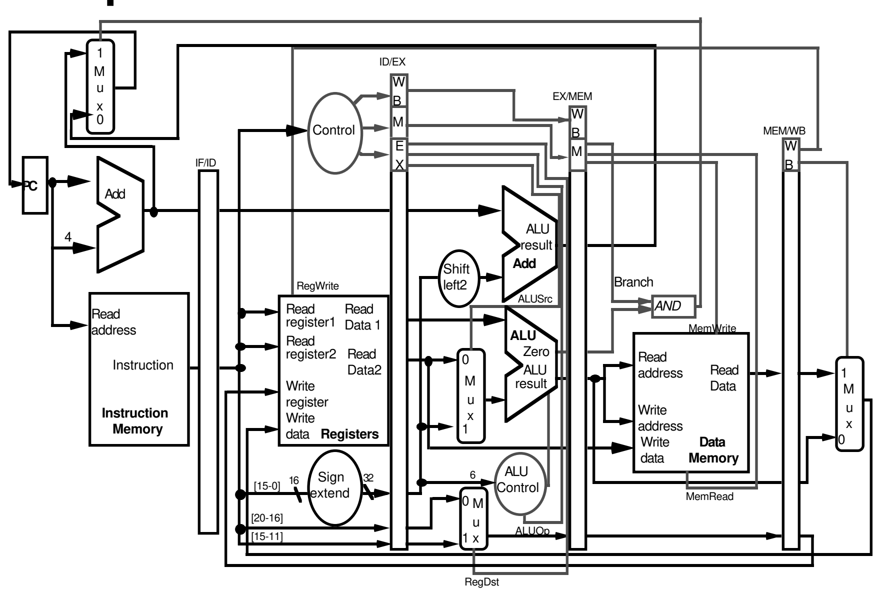
Remaining Issues in Pipeline Design

Interlock control is generally straightforward, but VERY detailed.

Pipelined Control MIPS-2000like



Detecting Data Hazards

EX Hazard:

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

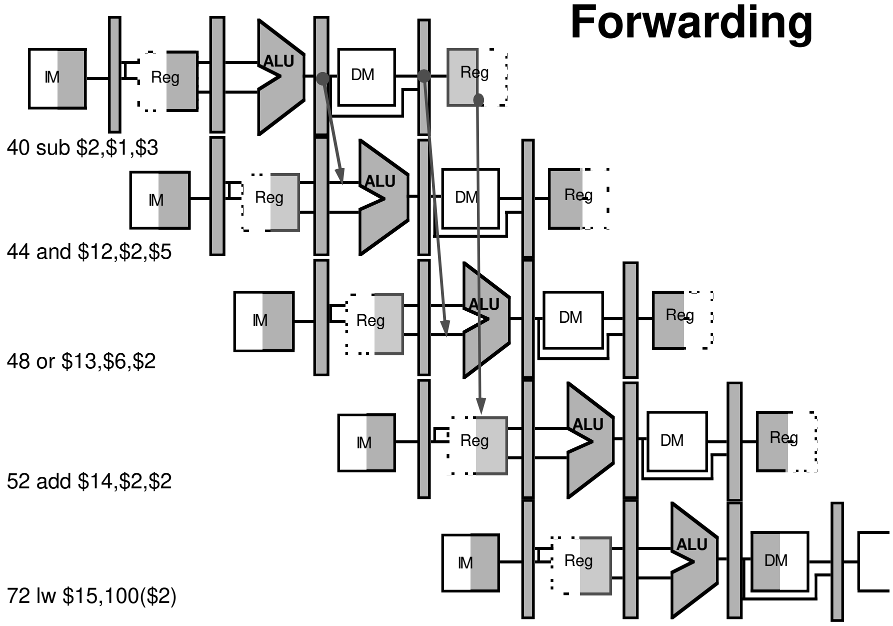
MEM Hazard:

EX/MEM.RegWrite &
((EX/MEM.WriteRegister=IF/ID.ReadRegister1) |
(EX/MEM.WriteRegister=IF/ID.ReadRegister2))

WB Hazard:

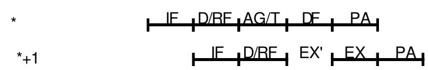
MEM/WB.RegWrite &
((MEM/WB.WriteRegister=IF/ID.ReadRegister1) |
(MEM/WB.WriteRegister=IF/ID.ReadRegister1))

Forwarding



Load Delays

A load followed by an ALU operation will generally incur a delay.



When there is a data dependency, forwarding will result in a one cycle delay.

When there is no data dependency, it may still be impossible to move the execution step forward since the write backs must be done in order and register files are single ported for write.

Review:

Timing diagram is the basic abstraction for designing a pipelined architecture.

The goal is to decompose the instructions so that the work is as balanced as possible.

- Fewer stages will lower the impact of stalls, but limit clock speed.
- More stages will permit a faster clock, but increase the amount of stalling.
- Using parallelism within stages might reduce stages.
- Special hardware "gadgets" can cache, precompute or manage aspects of the pipeline to regularize and simplify stages.
  - Instruction pre-fetch
  - TLB, BTB
  - PC <-- PC + 4

Review Continued

The problems are numerous.

- Branch hazards -- programs have very short instruction runs
  - Delayed branches help
  - Branch Prediction helps
- Data hazards -- computing effective addresses and data operations
  - Forwarding helps
- Miscellaneous Problems
  - Run-on instructions
  - Address hazards (fetch address just stored)