Forwarding for Stores

• Store requires data in the MEM phase for correct execution.

• Forwarding of data from an instruction that produces data in the EX phase, e.g., R type can be handled by extending the forwarding decision in the EX phase discussed earlier.

• if the store is preceded by a load then some extra work is needed.
• $LW \rightarrow SW$

\[
\begin{array}{c|c|c|c|c|c|c}
\text{LW:} & \text{IF} & \text{ID} & \text{EX} & \text{ME} & \text{WB} \\
\text{SW:} & \text{IF} & \text{ID} & \text{EX} & \text{ME} & \text{WB} \\
\end{array}
\]

• data produced in LW’s MEM stage is needed by SW’s MEM stage

• Add forwarding HW to MEM stage to detect this situation and forward from MEM/WB register since LW is in WB while SW is in MEM.
Data Hazards and Stalls

- Recall that all hazards caused by true dependences cannot be handled by forwarding alone.

- Consider $LW \rightarrow R - TYPE$

- Data produced by LW in MEM is consumed by R-Type in EX.
include figure 6.44
• the LW to AND dependence cannot be handled via forwarding.

• we must generate stalls

• as with forwarding we must add combinational logic to the pipe to detect and insert stalls
• We will stall in the ID stage after decoding the consuming instruction.

• Detection requires considering information about the LW in the EX stage and the consuming instruction in the ID stage

• If stall is generated, the current ID and IF stages must repeat

• PC and IF/ID registers must not change

• NOP inserted into EX stage (typically by clearing control lines, but all that is really needed is suppressing writes)
Code for Stalling ID and IF phase

if(ID/EX.MemRead and
   ((ID/EX.RegisterRt = IF/ID.RegisterRs)
   or
   (ID/EX.RegisterRt = IF/ID.RegisterRt)))
  stall pipeline

• Check for load via MemRead

• Check for destination of load equal to ID instruction sources.
insert figure 6.45
insert figure 6.46
• Note Store forwarding not included in 6.46

• Stall lines to hold PC and IF/ID included

• 0 for control lines in NOP included

• shifted offset data path not included

• IF/ID.RegisterRt duplication not needed.

• note that once the stall is added forwarding takes care of the rest.
insert figures 6.47 and 6.48
Summary of Data Hazards

- All are true dependences

- Other dependence types cause hazards but not in this type of pipeline

- Hazards caused by a consumer in the EX phase and data in the EX/MEM register or in the MEM/WB register is handled by hardware in the EX stage

- Detects hazard and locates most recent data

- R-Type $\rightarrow$ SW could be handled this way as well.
• $LW \rightarrow SW$ hazard

• data is present in MEM/WB register

• needed by SW in MEM phase, i.e., memory data line needs MUX

• hardware is in MEM stage

• Could also be used to handle R-Type $\rightarrow$ SW by getting R-Type data from MEM/WB register
Stalls

- LW in EX stage (data not ready yet)
- consumer in ID stage
- stalls IF and ID stage by suppressing update of PC and IF/ID registers
- NOP inserted via control lines
- Stalled instruction performs this processing using hardware in the ID stage