Can pipelining get us into trouble?

° **Yes: Pipeline Hazards**

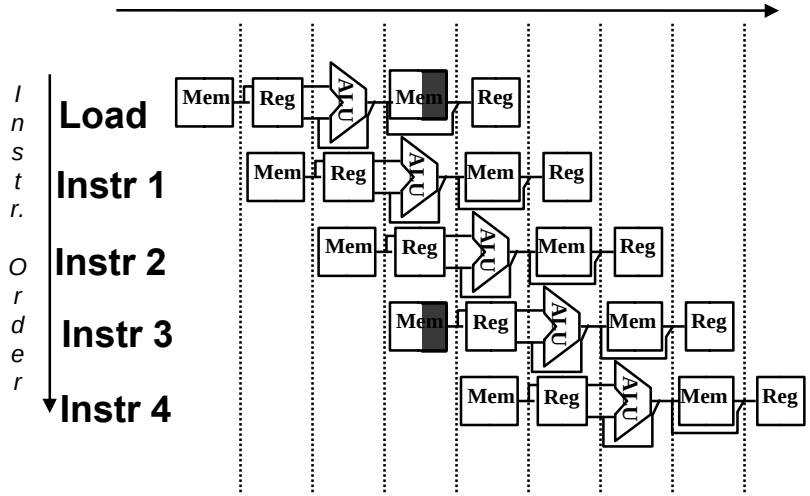
- **structural hazards: attempt to use the same resource two different ways at the same time**
  - **E.g., combined washer/dryer would be a structural hazard or folder busy doing something else (watching TV)**
- **data hazards: attempt to use item before it is ready**
  - **E.g., one sock of pair in dryer and one in washer; can't fold until get sock from washer through dryer**
  - **instruction depends on result of prior instruction still in the pipeline**
- **control hazards: attempt to make a decision before condition is evaulated**
  - **E.g., washing football uniforms and need to get proper detergent level; need to see after dryer before next load in**
  - **branch instructions**

° **Can always resolve hazards by waiting**

- **pipeline control must detect the hazard**
- **take action (or delay action) to resolve hazards**

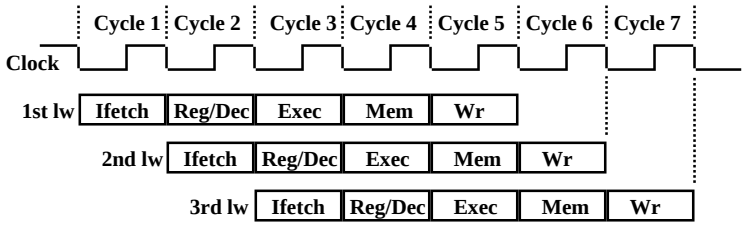
Single Memory is a Structural Hazard

Solution: Separate Instruction and Data memory



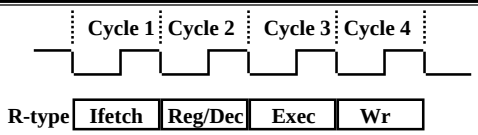
Detection is easy in this case! (right half highlight means read, left half write)

Pipelining the Load Instruction: no structural hazards



° The five independent functional units in the pipeline datapath are:
- Instruction Memory for the Ifetch stage
- Register File's Read ports (bus A and busB) for the Reg/Dec stage
- ALU for the Exec stage
- Data Memory for the Mem stage
- Register File's Write port (bus W) for the Wr stage

The Four Stages of R-type



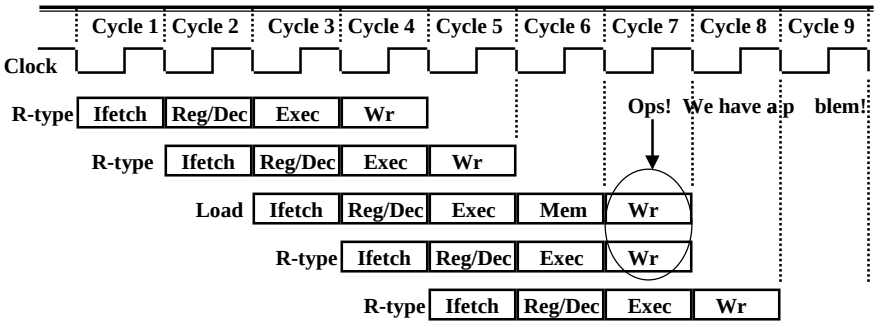
° Ifetch: Instruction Fetch
- Fetch the instruction from the Instruction Memory

° Reg/Dec: Registers Fetch and Instruction Decode

° Exec:
- ALU operates on the two register operands
- Update PC

° Wr: Write the ALU output back to the register file

Pipelining the R-type and Load Instruction

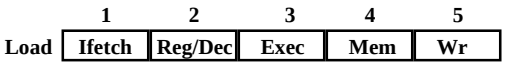


° We have pipeline conflict or structural hazard:
- Two instructions try to write to the register file at the same time!
- Only one write port

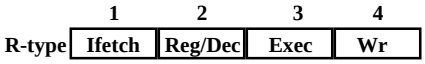
Important Observation

° Each functional unit can only be used once per instruction

° Each functional unit must be used at the same stage for all instructions:
- Load uses Register File's Write Port during its 5th stage

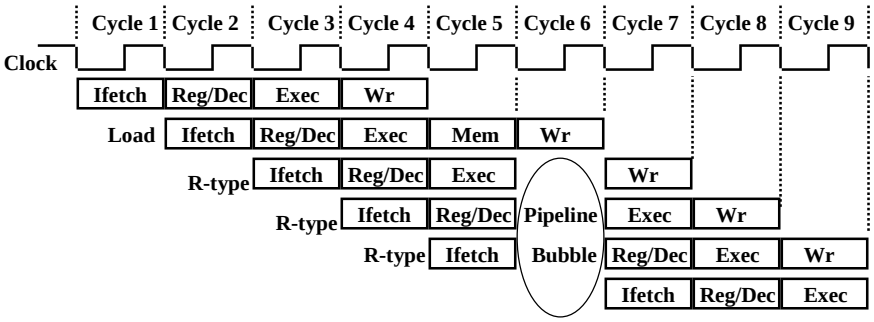


- R-type uses Register File's Write Port during its 4th stage



° 2 ways to solve this pipeline hazard.

Solution 1: Insert “Bubble” into the Pipeline

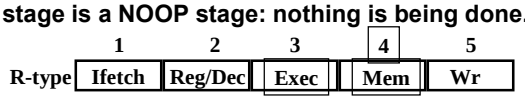


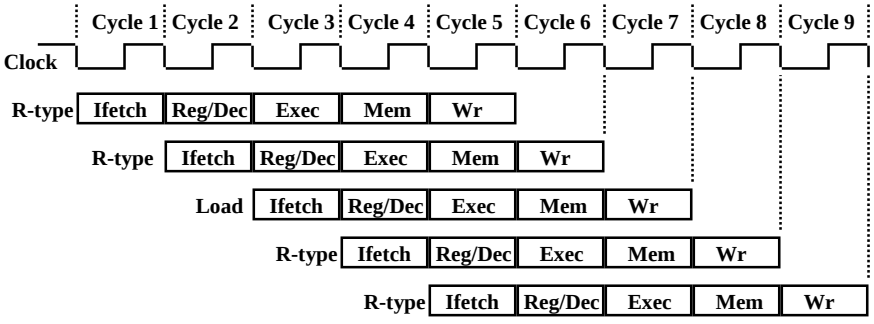
° Insert a “bubble” into the pipeline to prevent 2 writes at the same cycle
  - The control logic can be complex.
  - Lose instruction fetch and issue opportunity.

° No instruction is started in Cycle 6!

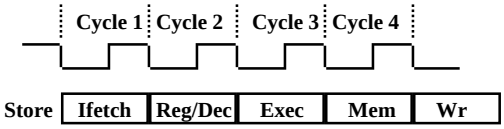
Solution 2: Delay R-type’s Write by One Cycle

° Delay R-type’s register write by one cycle:
  - Now R-type instructions also use Reg File’s write port at Stage 5
  - Mem stage is a NOOP stage: nothing is being done.



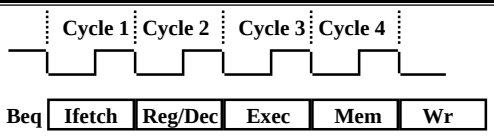


The Four Stages of Store: no structural hazard



° Ifetch: Instruction Fetch
  • Fetch the instruction from the Instruction Memory
° Reg/Dec: Registers Fetch and Instruction Decode
° Exec: Calculate the memory address
° Mem: Write the data into the Data Memory

The Three Stages of Beq: no structural hazard



° Ifetch: Instruction Fetch
  • Fetch the instruction from the Instruction Memory
° Reg/Dec:
  • Registers Fetch and Instruction Decode
° Exec:
  • compares the two register operand,
  • select correct branch target address
  • latch into PC