In-order vs. Out-of-order Execution

In-order instruction execution
- instructions are fetched, executed & completed in compiler-generated order
- one stalls, they all stall
- instructions are **statically scheduled**

Out-of-order instruction execution
- instructions are fetched in compiler-generated order
- instruction completion may be in-order (today) or out-of-order (older computers)
- in between they may be executed in some other order
- independent instructions behind a stalled instruction can pass it
- instructions are **dynamically scheduled**
Dynamic Scheduling

Out-of-order processors:

- after instruction decode
  
  - check for **structural hazards**
    - an instruction can be issued when a functional unit is available
    - an instruction stalls if no appropriate functional unit
  
  - check for **data hazards**
    - an instruction can execute when its operands have been calculated or loaded from memory
    - an instruction stalls if operands are not available
Dynamic Scheduling

Out-of-order processors:

- don’t wait for previous instructions to execute if this instruction does not depend on them

- ready instructions can execute before earlier instructions that are stalled, e.g., waiting for their data to be loaded from memory
  - when go around a load instruction that is stalled for a cache miss:
    - use lockup-free caches that allow instruction issue to continue while a miss is being satisfied
    - the load-use instruction still stalls
Dynamic Scheduling

in-order processors

\[
\begin{align*}
\text{lw} & \; \$3, \; 100($4) \quad \text{in execution, cache miss} \\
\text{add} & \; \$2, \; \$3, \; \$4 \quad \text{waits until the miss is satisfied} \\
\text{sub} & \; \$5, \; \$6, \; \$7 \quad \text{waits for the add}
\end{align*}
\]

out-of-order processors

\[
\begin{align*}
\text{lw} & \; \$3, \; 100($4) \quad \text{in execution, cache miss} \\
\text{sub} & \; \$5, \; \$6, \; \$7 \quad \text{in execution during the cache miss} \\
\text{add} & \; \$2, \; \$3, \; \$4 \quad \text{waits until the miss is satisfied}
\end{align*}
\]
Dynamic Scheduling

Out-of-order processors:

- ready instructions can execute before earlier instructions that are stalled, e.g., waiting for their branch condition to be computed
- when go around a **branch** instruction:
  - the instructions that are issued from the predicted path are issued speculatively, called **speculative execution**
  - speculative instructions can execute (but not commit) before the branch is resolved
  - if the prediction was wrong, speculative instructions are flushed from the pipeline
  - if prediction is right, instructions are no longer speculative
Instruction **speculation**: executing an instruction before it is known that it should be executed

- must be safe (no additional exceptions) or must handle the exceptions after the instruction is no longer speculative
- must generate the same results