Superscalar

- A superscalar machine executes multiple independent instructions in parallel.
- They are pipelined as well.
- “Common” instructions (arithmetic, load/store, conditional branch) can be executed independently.
- The hardware and the compiler assure that parallel execution does not violate the intent of the program.

Example:
- Ordinary pipeline: four stages (Fetch, Decode, Execute, Writeback), one clock cycle per stage. Execute 6 instructions.

1 2 3 4 5 6 7 8 9

- 2-degree superscalar: perform two pipeline stages per clock cycle. (or split each stage into two non overlapping parts and each can execute in half a clock cycle)
Limitations of Superscalar

- Fundamental limitations the system must cope:
  - True data dependency
  - Procedural dependency (depends on branch prediction)
  - Resource conflicts (integer processor, floating-point processor, branch unit)
  - Output dependency: Write after Write (WAW)
  - Anti-dependency: Write after Read (WAR)

True data dependency (Must W before R)

\[
\begin{align*}
I0: & \text{ ADD R1 R2 R3 } \# r3 = r1 + r2 \\
I1: & \text{ MOVE R3 R4 } \# r4 = r3 \\
\end{align*}
\]

- Can fetch and decode second instruction in parallel with first
- Can NOT execute second instruction until the first is finished.
  - Second instruction is dependent on first (RAW)