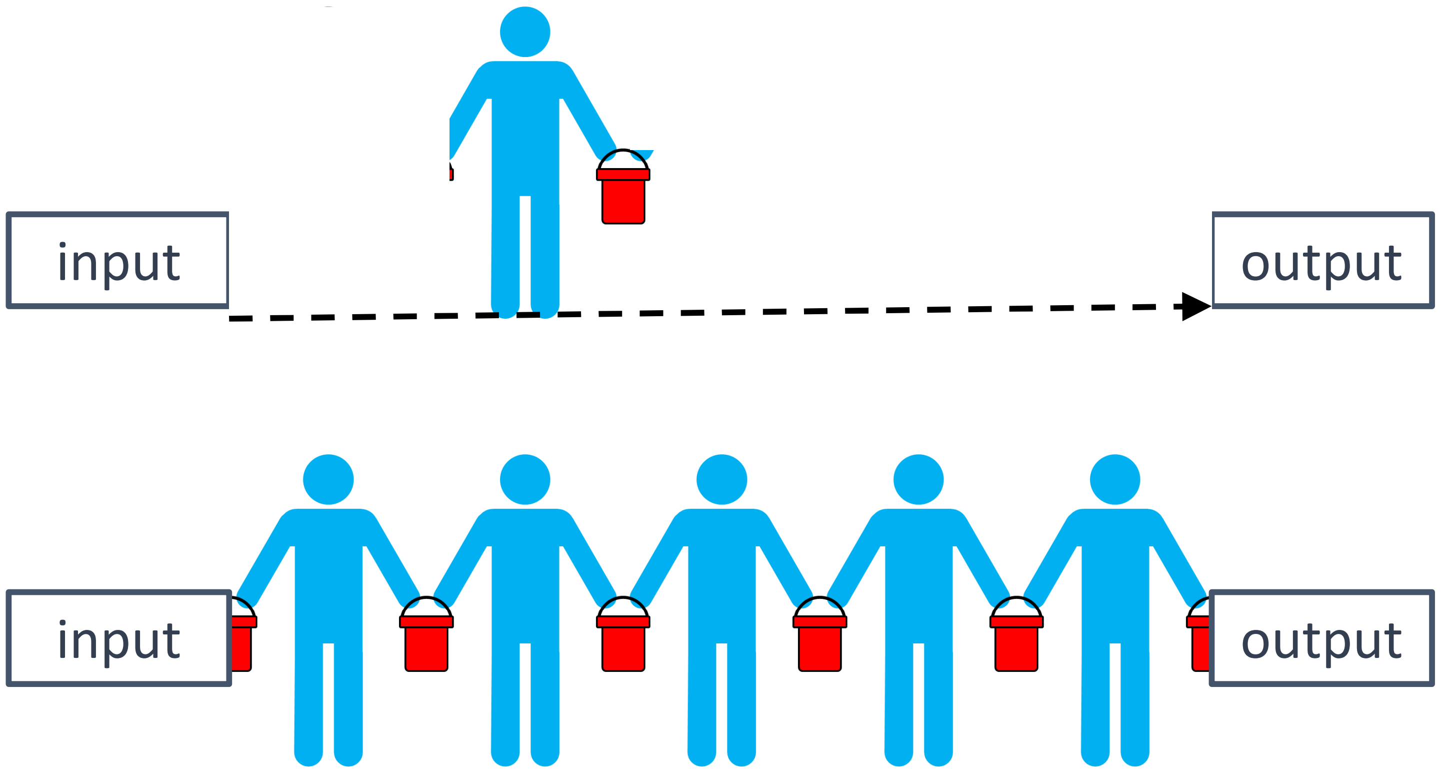
Pipelining / Parallelism

CSE 410

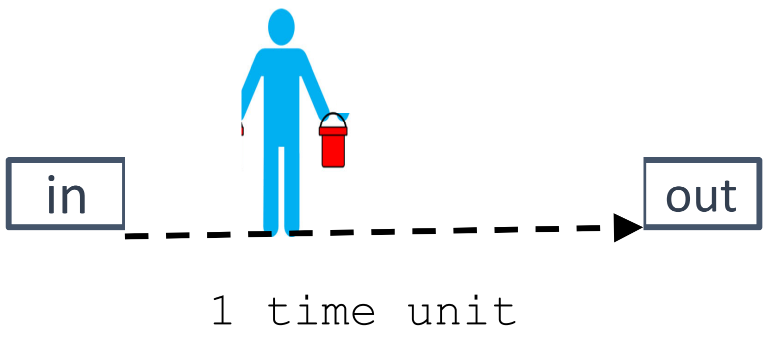
Lecture 09

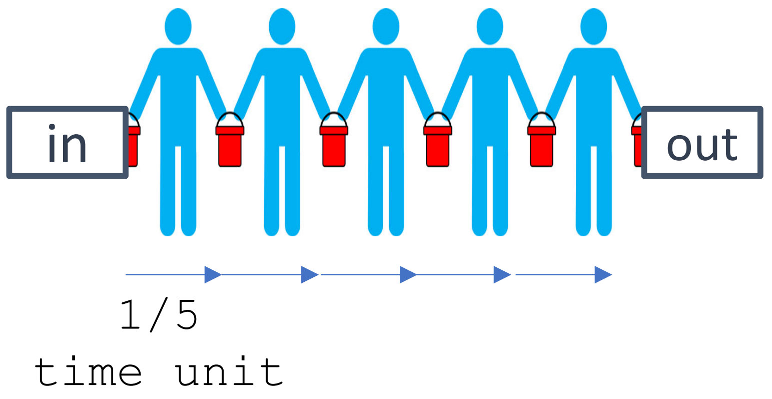
Parallelism / Pipelining



- Parallelism: Go faster by doing many things at once
- Pipelining: A particular form of parallelism

Pipelining

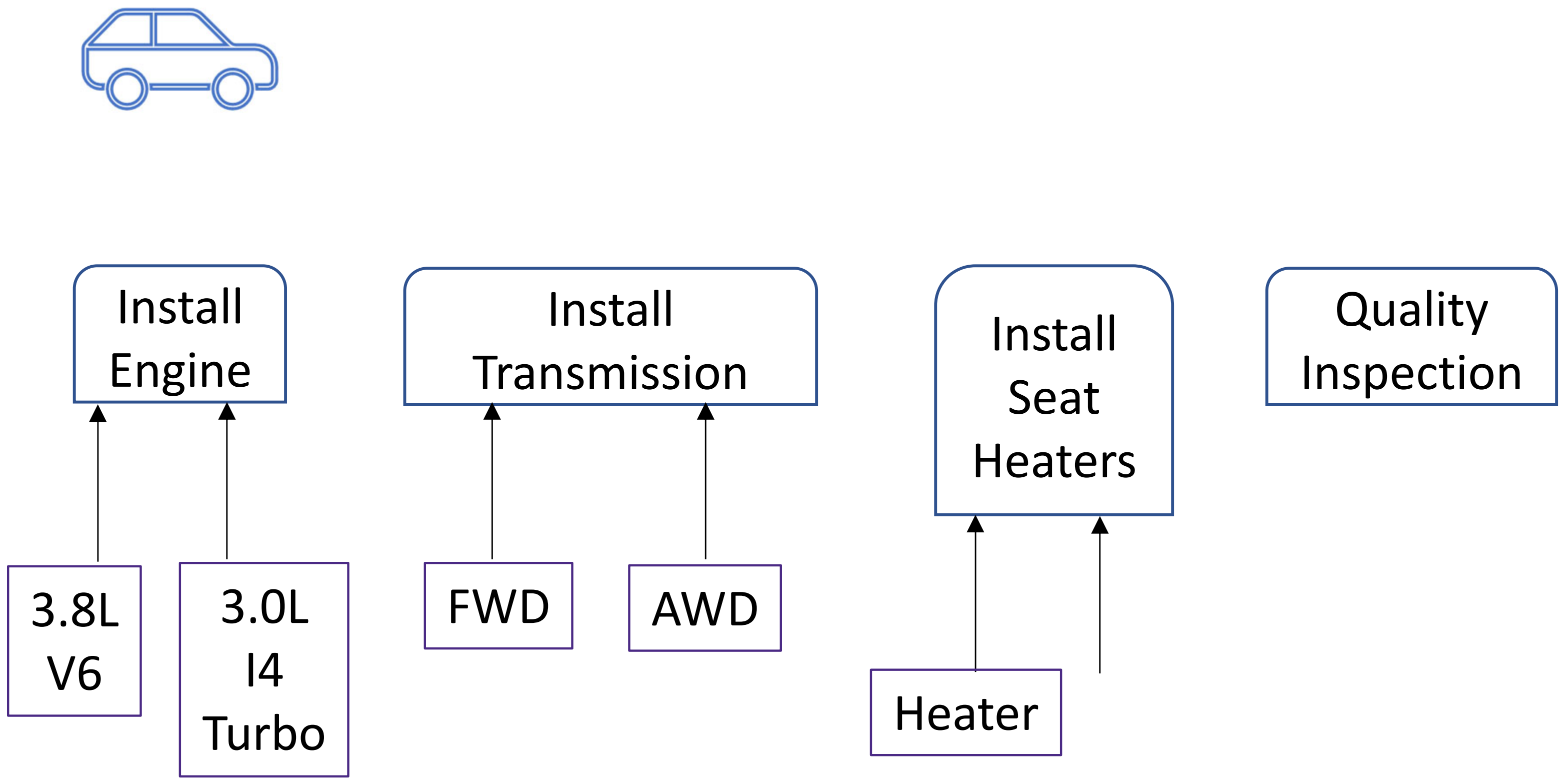




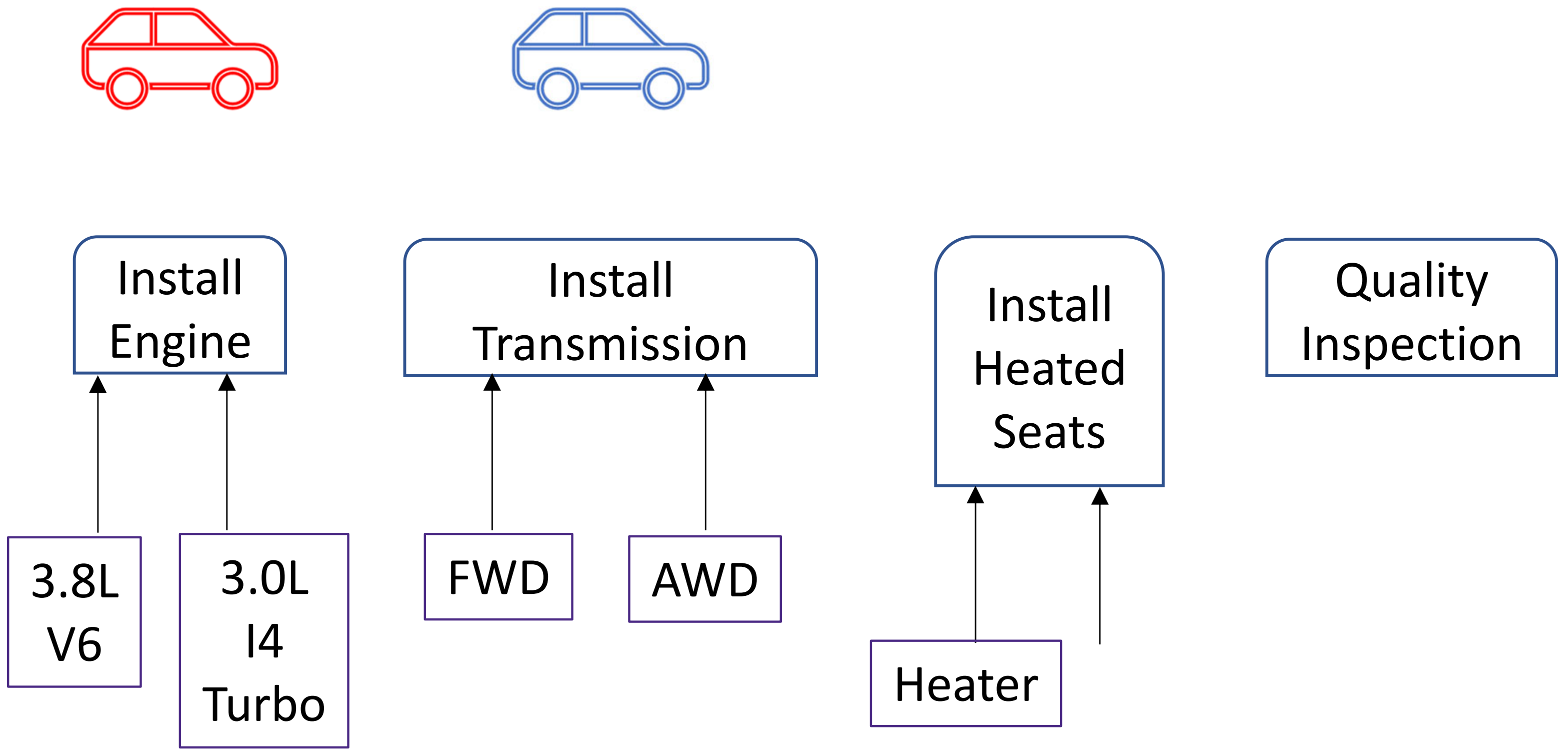
- If it takes time 1 to move a bucket from input to output, in the ideal case it takes time 1/5 when pipelined, but...

- There is some overhead in the “handoffs” required between “states”
- Some stages are faster than others, some slower
- The entire pipeline can move forward only as fast as the slowest stage
  - (Might be able to speed things up by dividing the slow stage (by pipelining it))

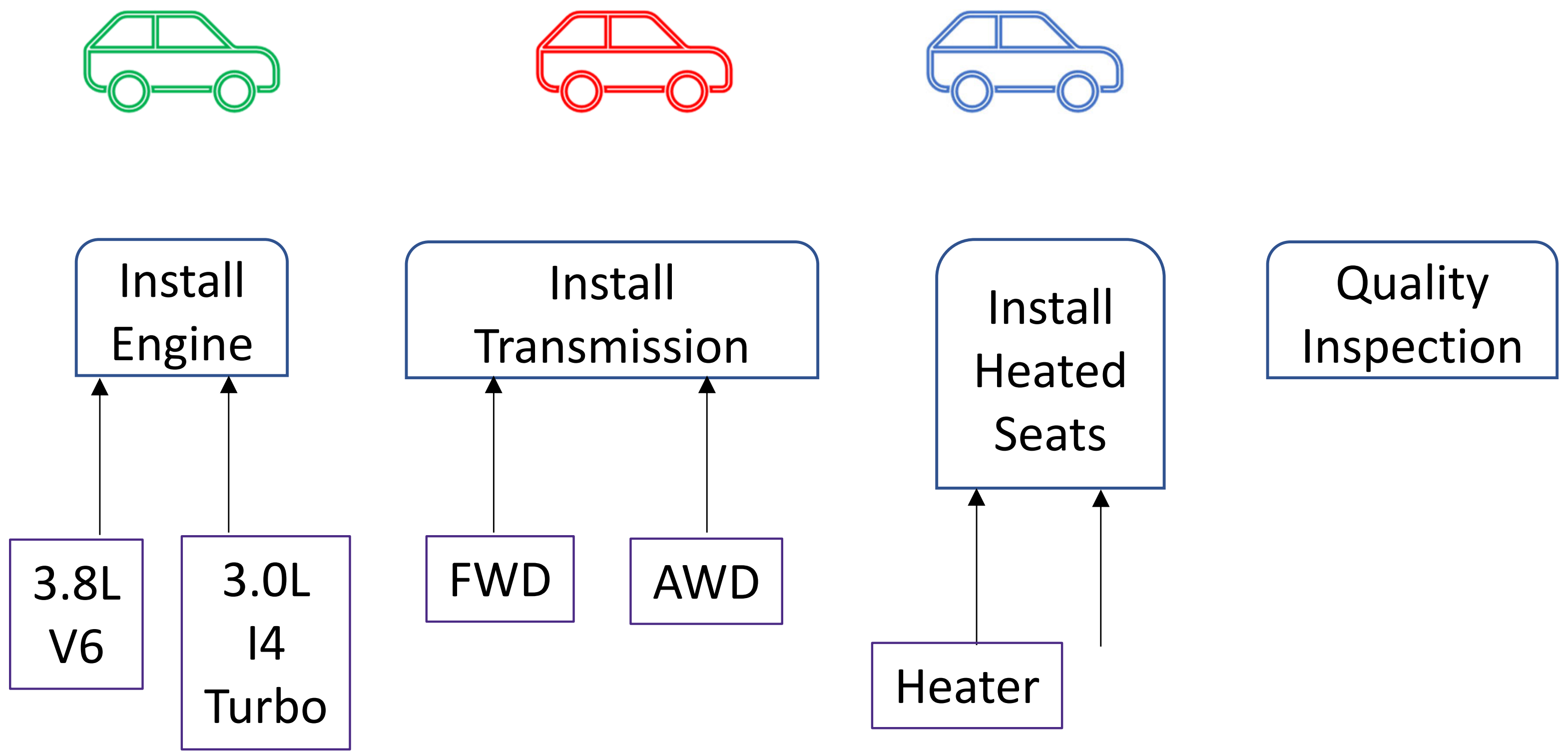
Pipeline as Assembly Line



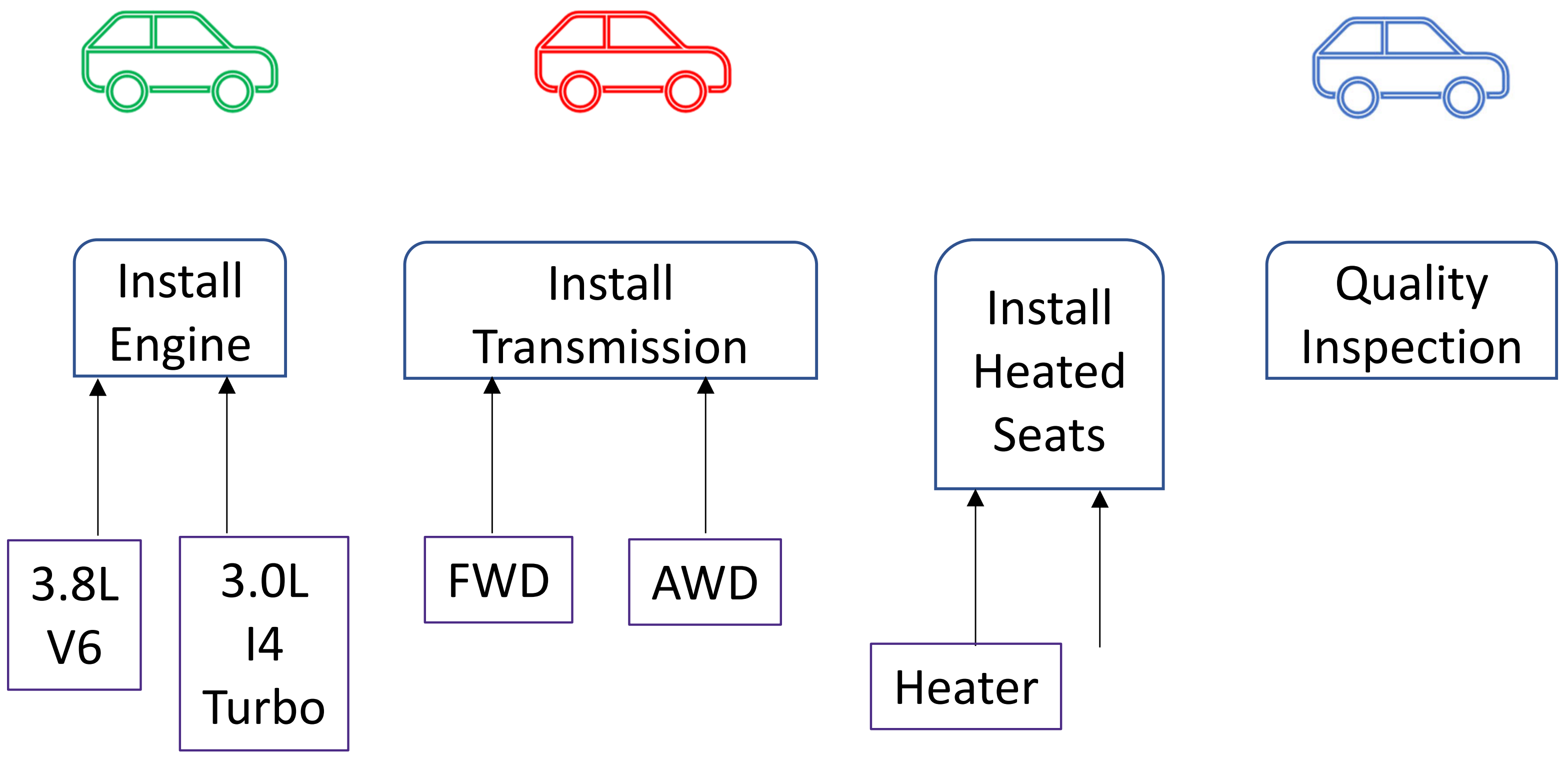
Pipeline as Assembly Line



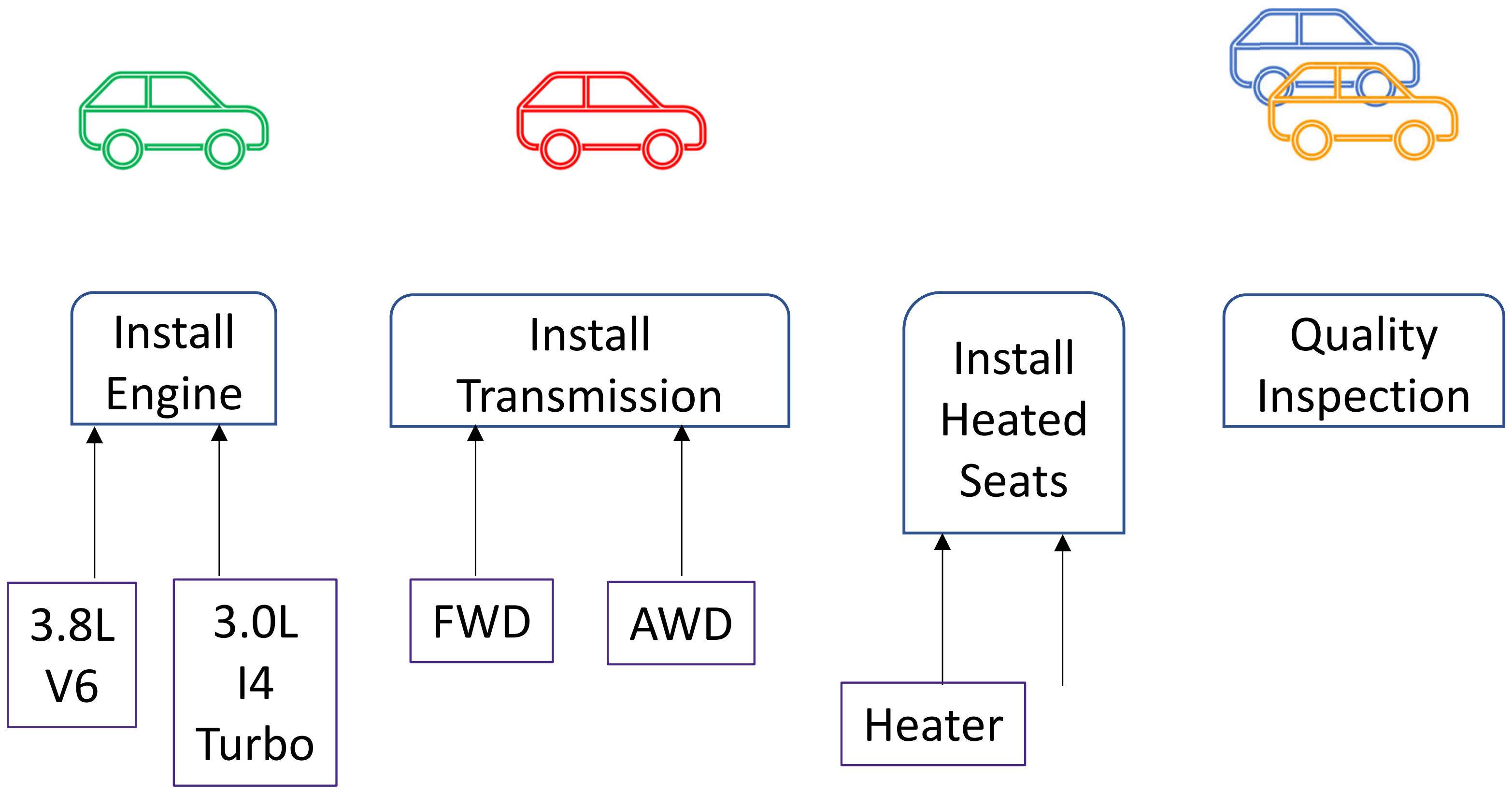
Pipeline as Assembly Line



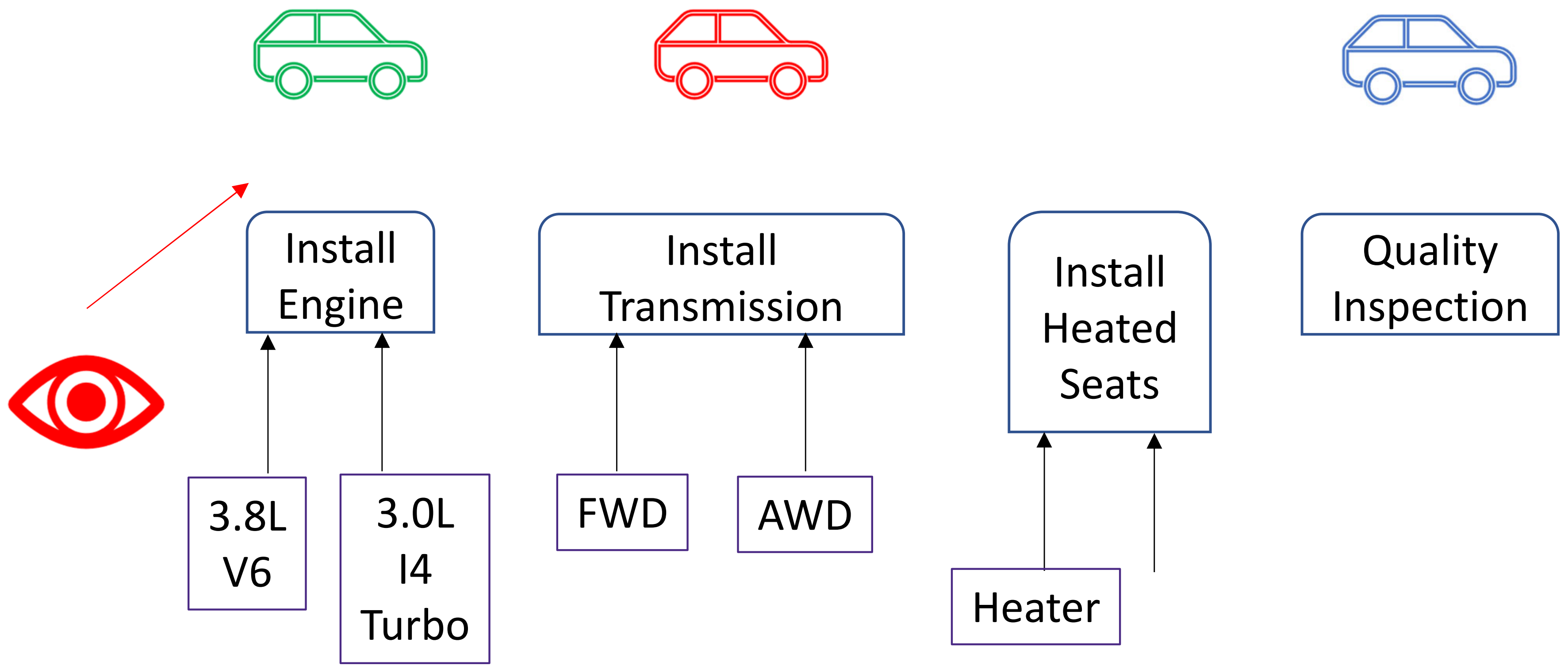
Does Skipping Steps Help?



Does Skipping Steps Help? No!

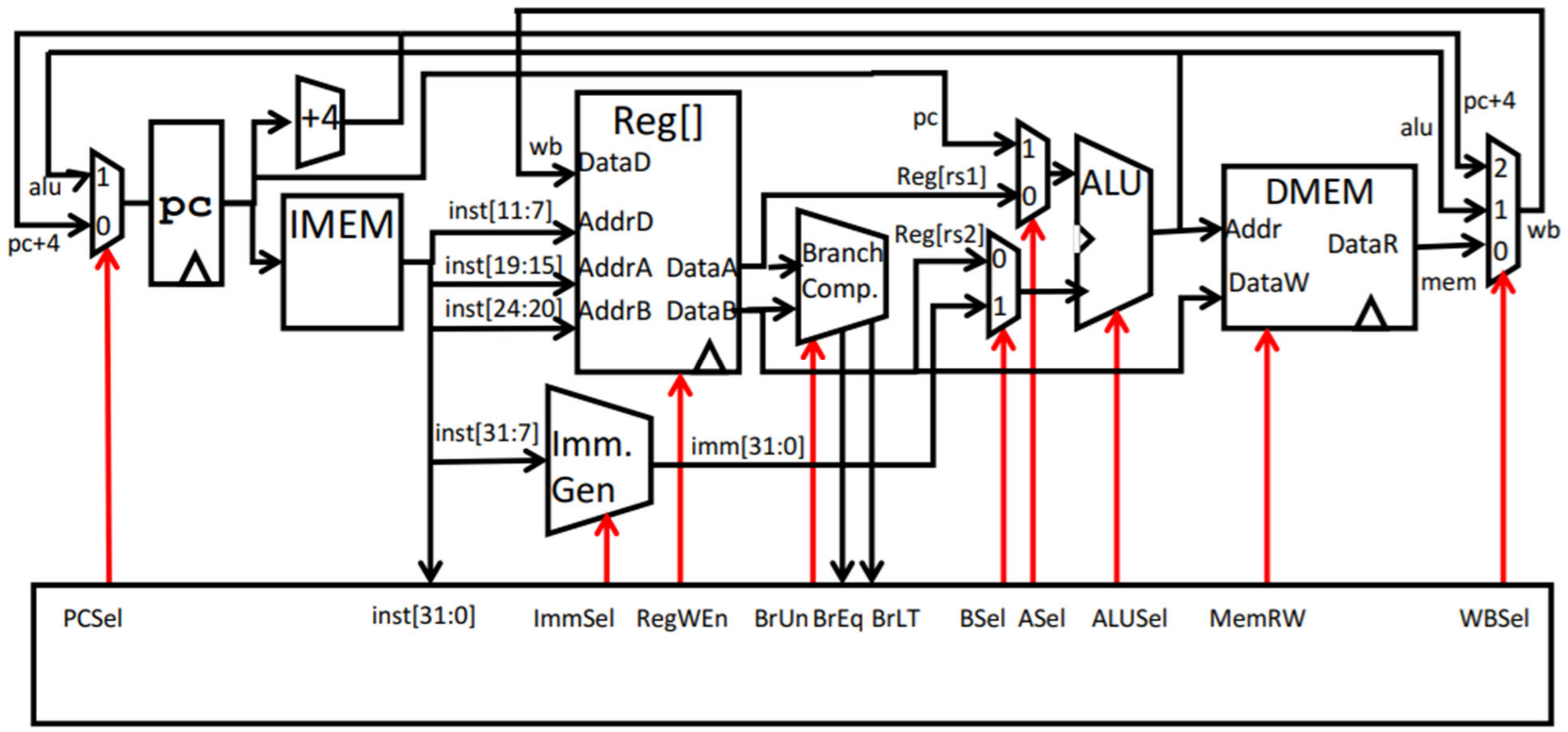


Does Skipping Steps Help? Still No



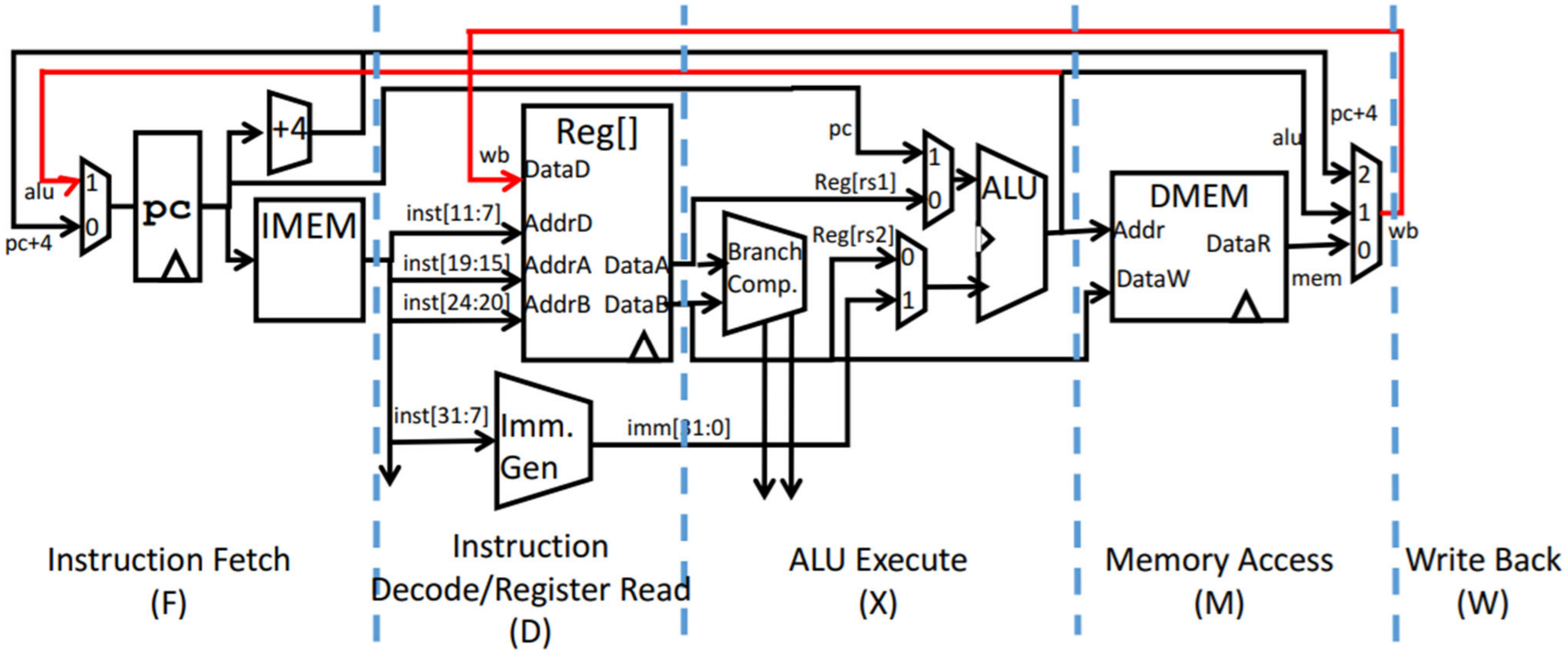
Our goal is to maximize throughput – cars produced per hour.
There is one car produced for each one that enters the assembly line.
How long it is before a car leaves the assembly line is irrelevant.

Datapath: Single-Cycle Implementation

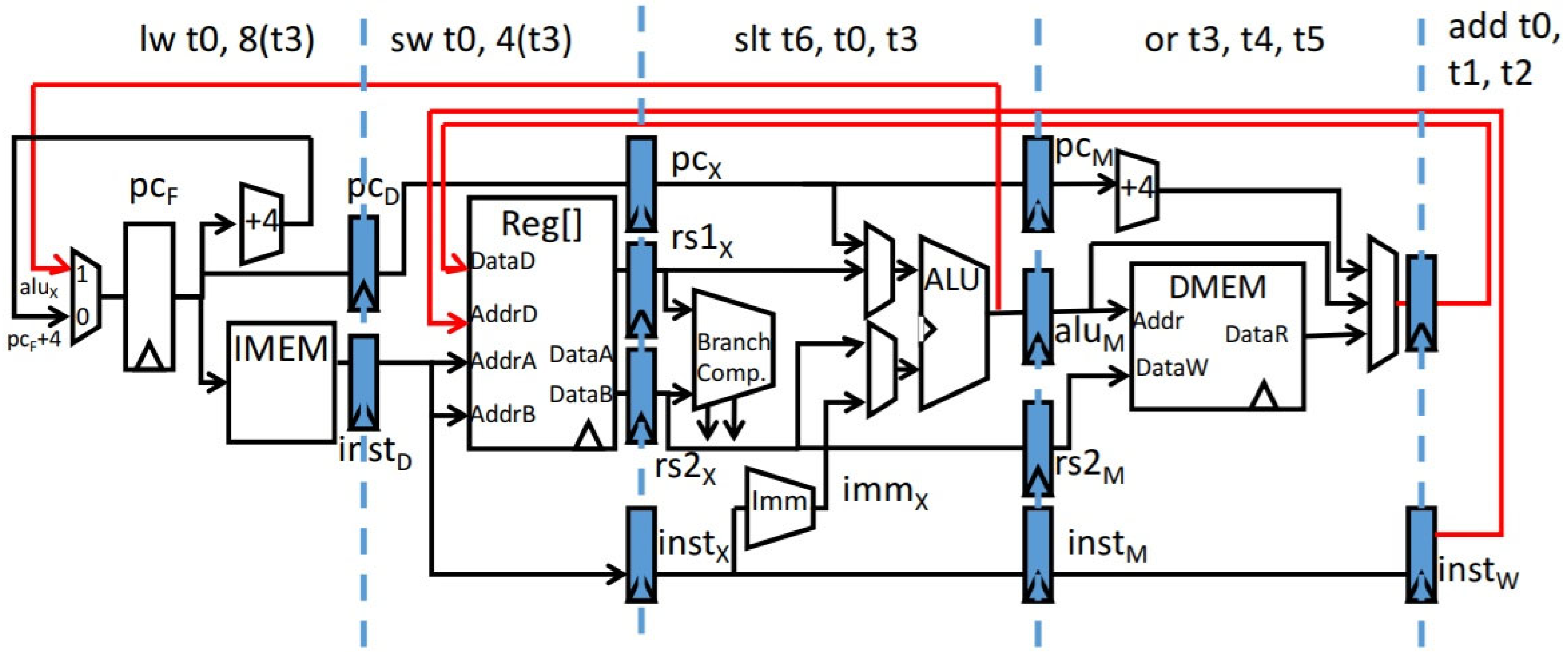


https://inst.eecs.berkeley.edu/~cs61c/fa17/lec/13/L13%20Pipelining%20(1up).pdf

5-Stage Pipeline

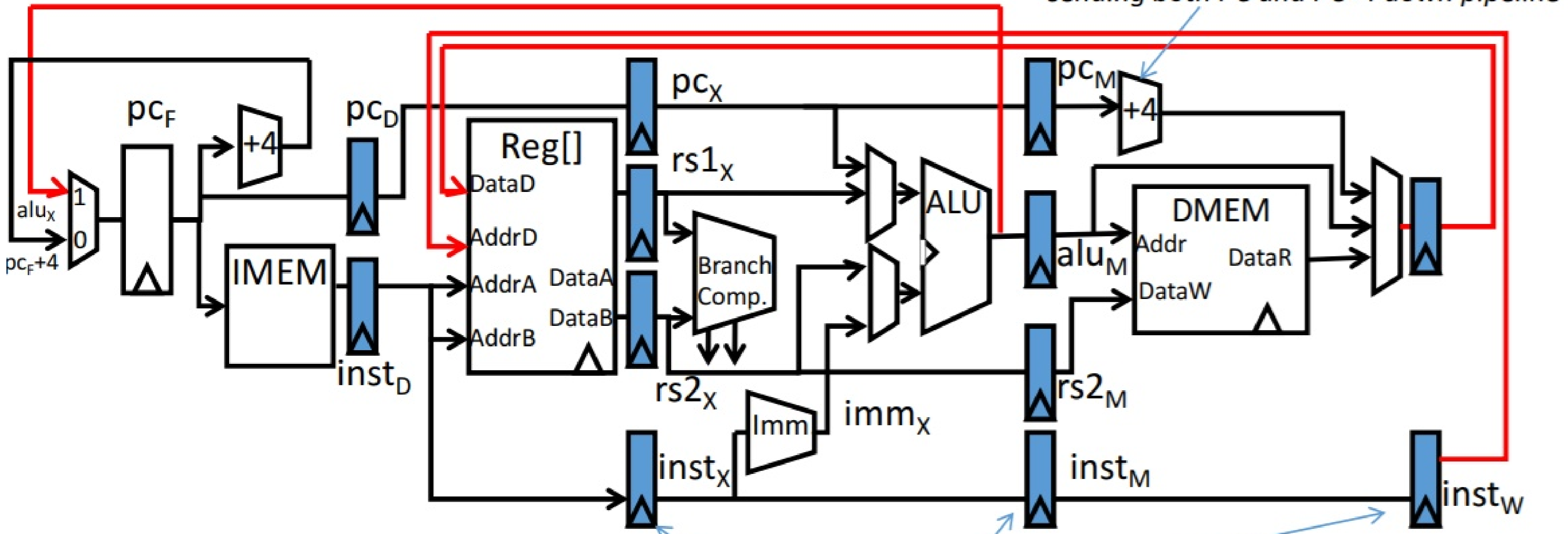


Pipeline in Execution



Pipeline registers separate stages, hold data for each instruction in flight

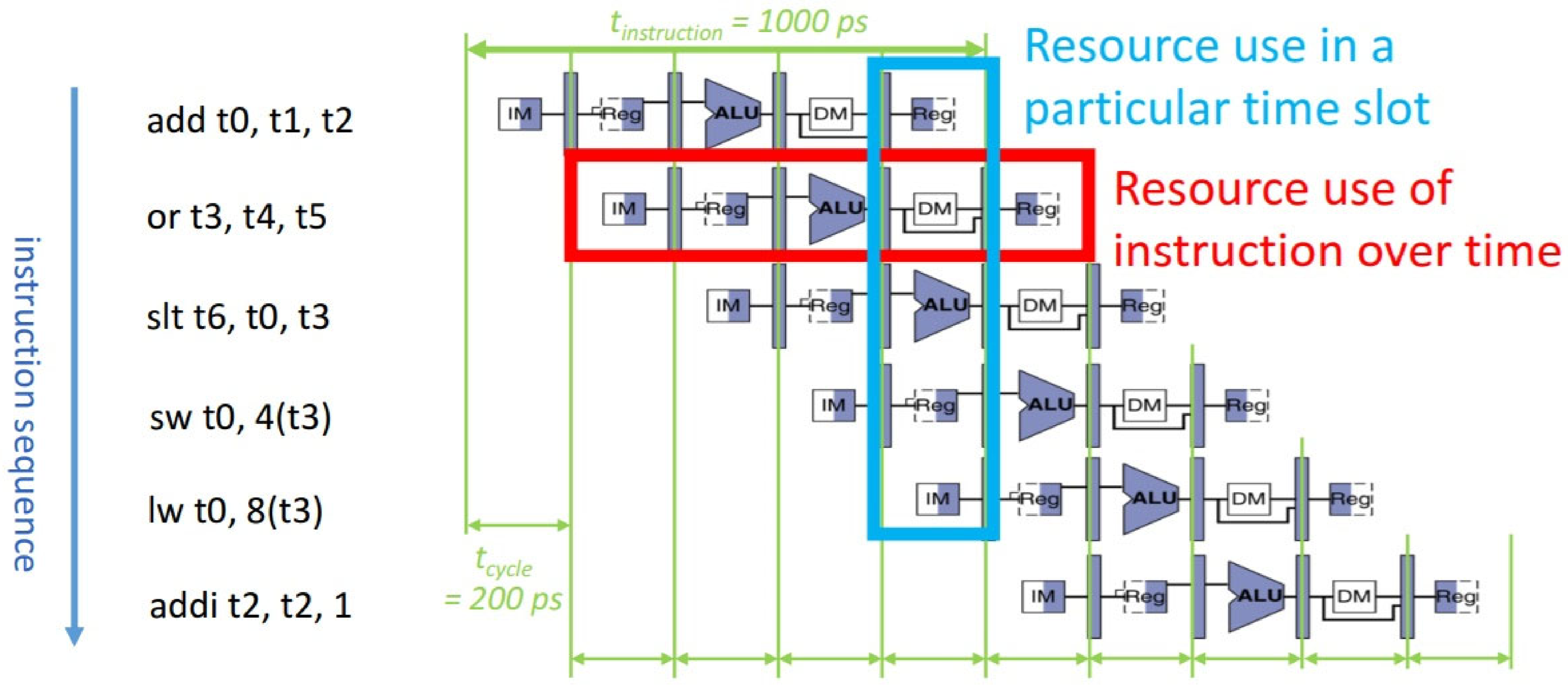
5-stage Pipeline Implementation



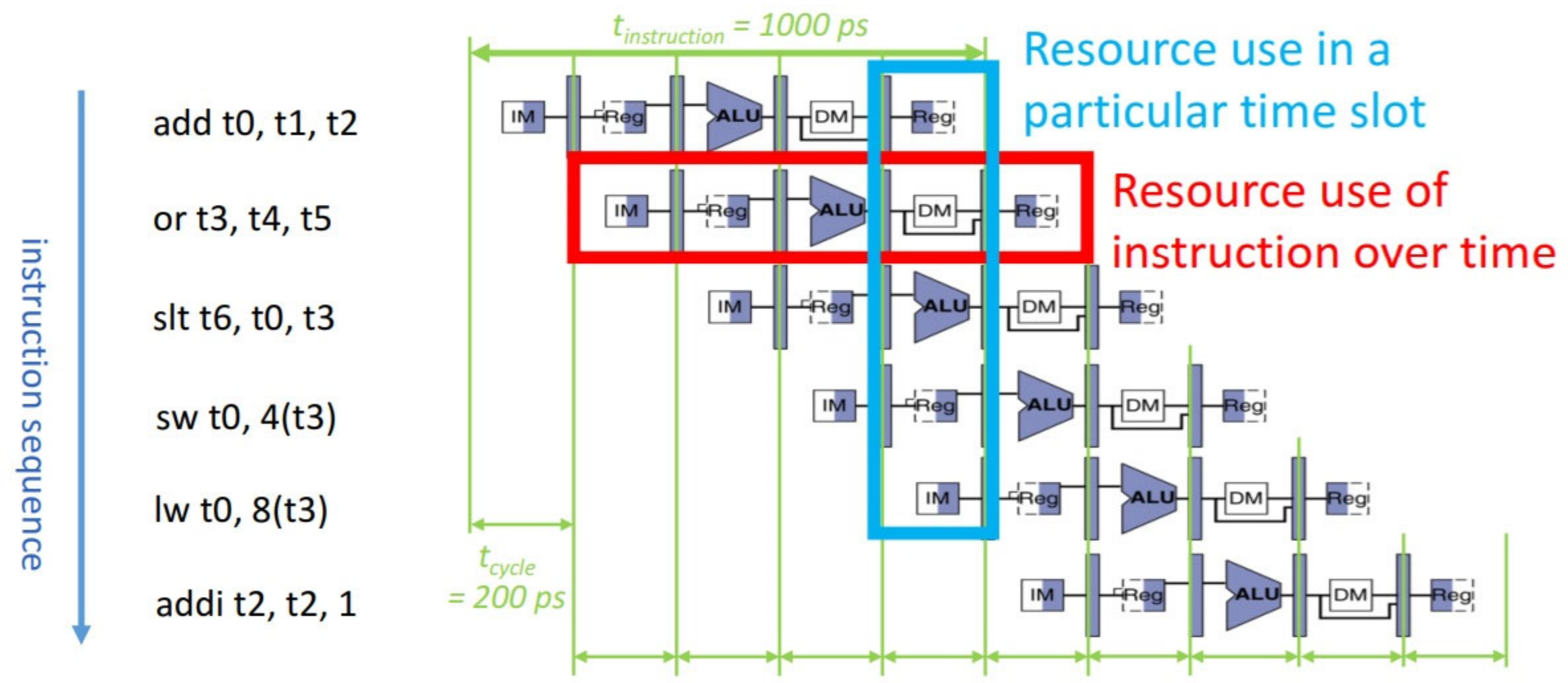
Recalculate PC+4 in M stage to avoid sending both PC and PC+4 down pipeline

Must pipeline instruction along with data, so control operates correctly in each stage

Alternate View of Pipeline Execution

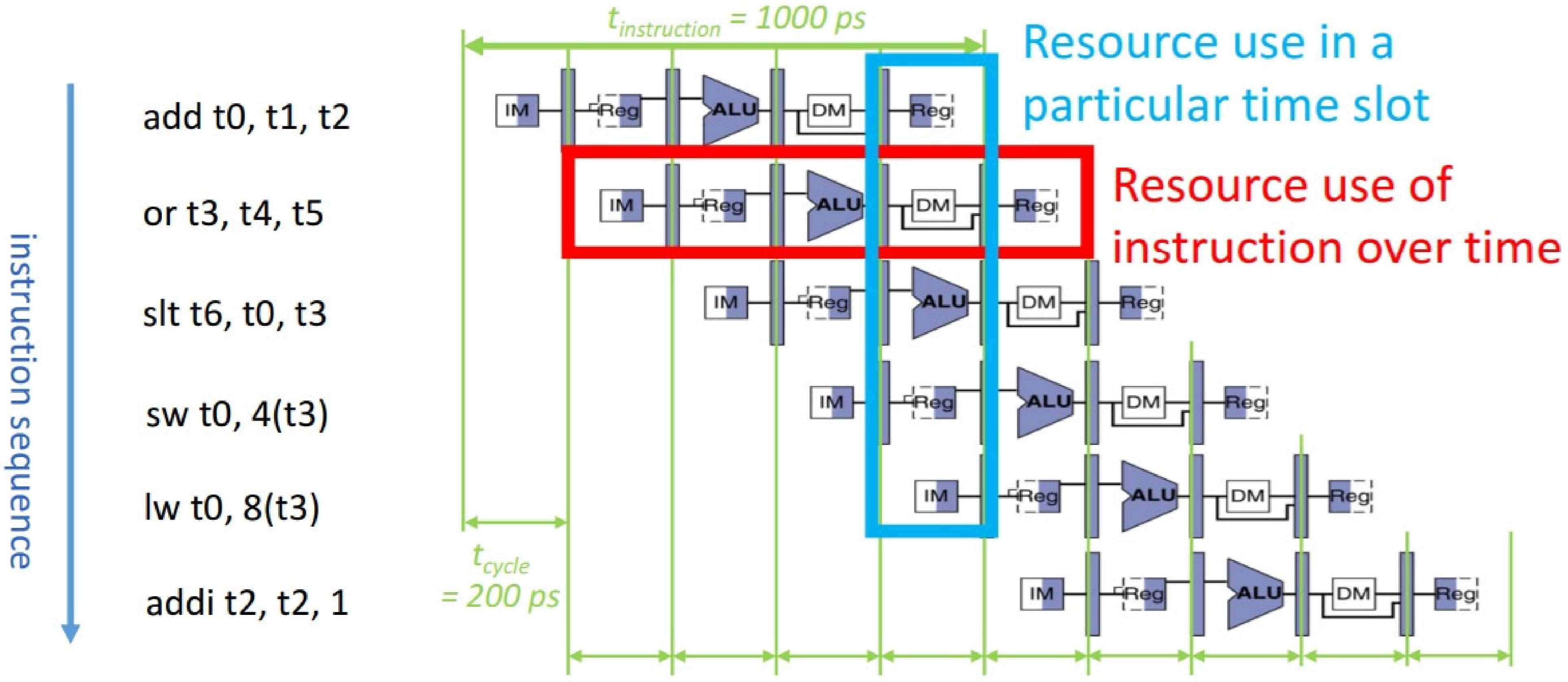


Alternate View of Pipeline Execution



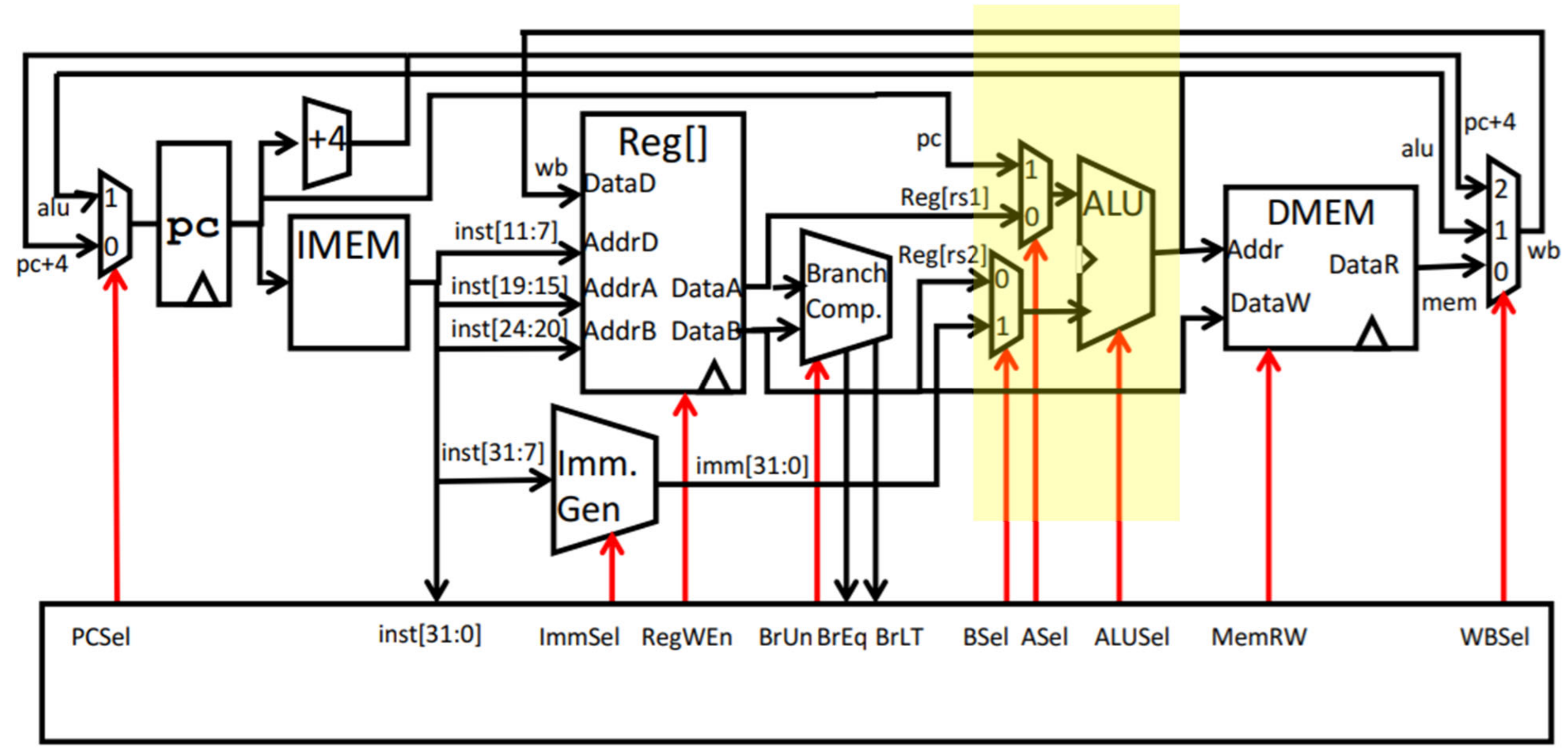
- The cycle time of the pipelined implementation is constrained by the slowest stage
- If single cycle implementation requires 1000 psec to execute an instruction, the minimum 5-stage pipeline cycle time is 200 psec
  - Perfectly balanced stages
  - No penalty for pipeline register reading/writing

Alternate View of Pipeline Execution



- It still takes 1000 psec between when an instruction is fetched and when that instruction has completed execution
- But, an instruction is **completing** every 200 psec
  - So, the pipeline implementation is completing instructions at five times the rate of the single cycle (in this idealistic view)

Why is Pipelining Faster?



- In the single cycle implementation, most components “are idle” as the signal propagates through the entire circuit
- Pipelining makes use of all the components all the time (in an ideal world)

Limits to Pipelining

- If 5 stages are 5 times faster, why not 10 stages?
  - Can’t balance stages well enough for it to pay off
  - More stages require more circuitry/space
  - Hazards
    - Properties of the instruction sequence that prevent full use of the pipeline