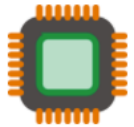


CASPER

CS305: Computer Architecture

Superscalar and out-of-order processors:
10K feet view

https://www.cse.iitb.ac.in/~biswa/courses/CS305/main.html

https://www.cse.iitb.ac.in/~biswa/

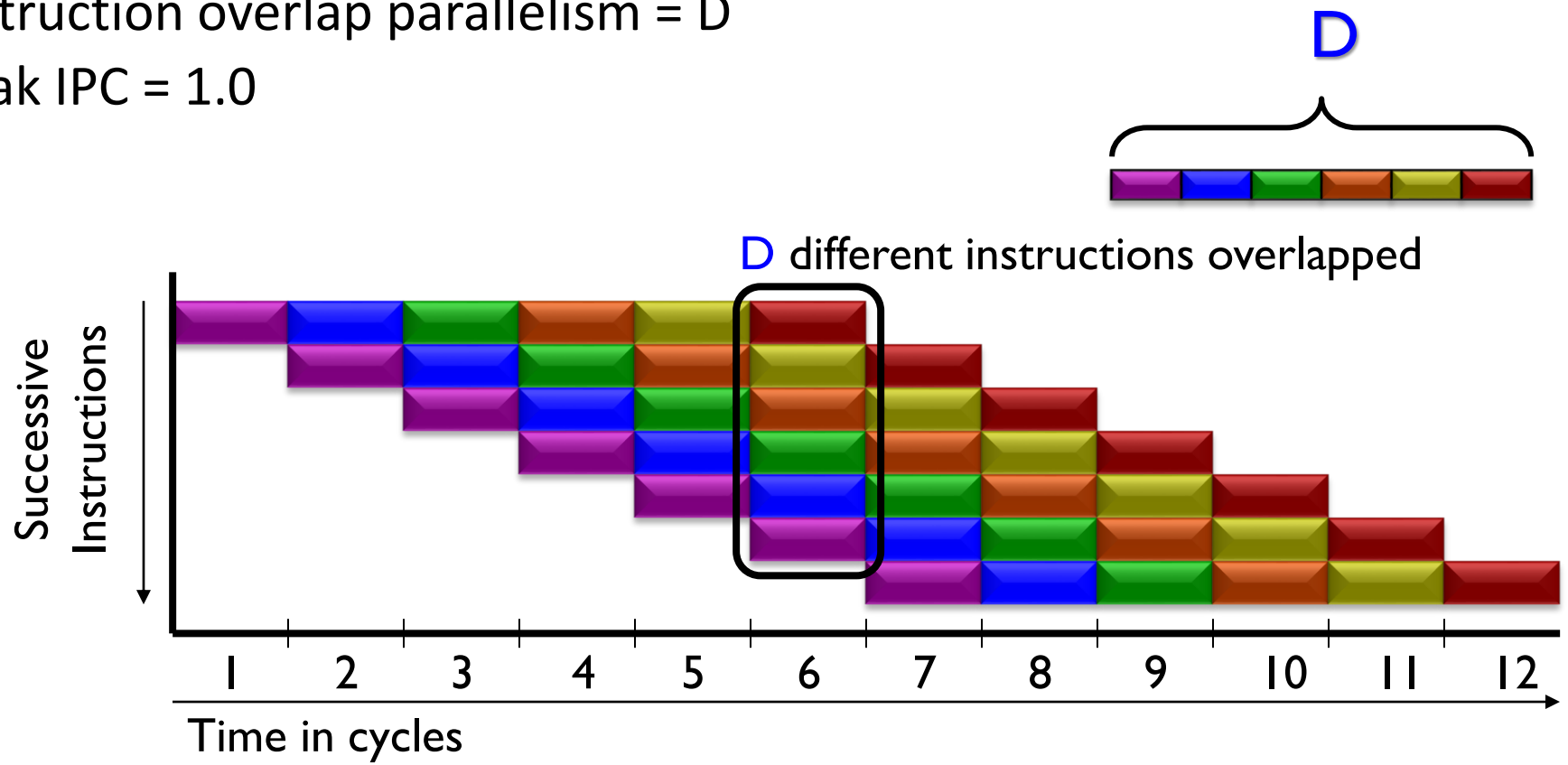
Beyond Scalar

- Scalar pipeline limited to CPI ≥ 1.0
  - Can never run more than 1 insn per cycle

- "Superscalar" can achieve CPI ≤ 1.0 (i.e., IPC ≥ 1.0)
  - *Superscalar* means executing multiple insns in parallel

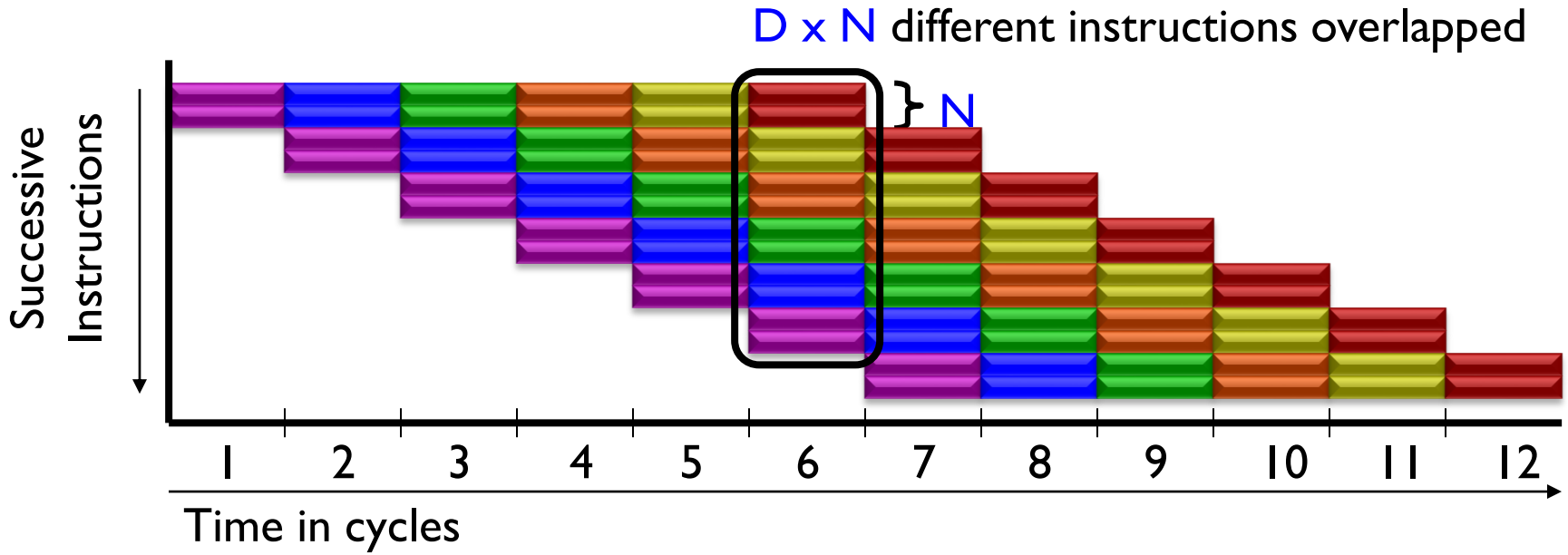
Instruction Level Parallelism (ILP)

- Scalar pipeline (baseline)
  - Instruction overlap parallelism = D
  - Peak IPC = 1.0



Superscalar Processor

- Superscalar (pipelined) Execution
  - Instruction parallelism = D x N
  - Peak IPC = N per cycle



What is the deal?

We get an IPC boost if the number of instructions fetched in one cycle are independent ☺

Complicates datapaths, multi-ported structures, complicates exception handling

Out of order (O3) processor: Pursuit of even higher IPC

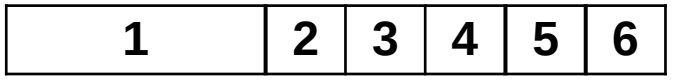
Out-of-order follows data-flow order

Example:

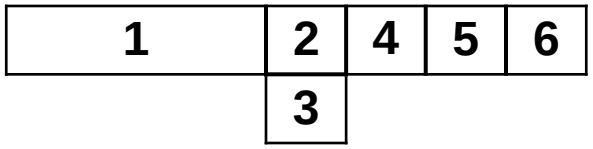
```
(1) r1 ← r4 / r7
(2) r8 ← r1 + r2
(3) r5 ← r5 + 1
(4) r6 ← r6 - r3
(5) r4 ← r5 + r6
(6) r7 ← r8 * r4
```

/* assume division takes 20 cycles */

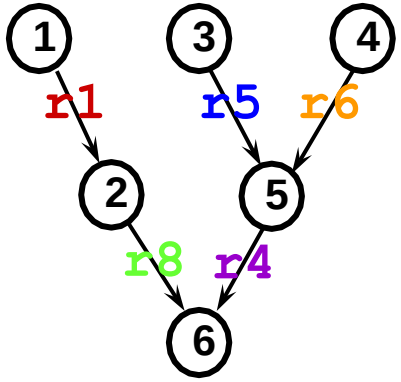
In-order execution



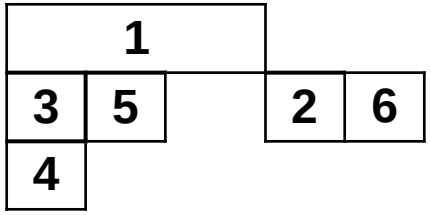
In-order (2-way superscalar)



Data Flow Graph



Out-of-order execution

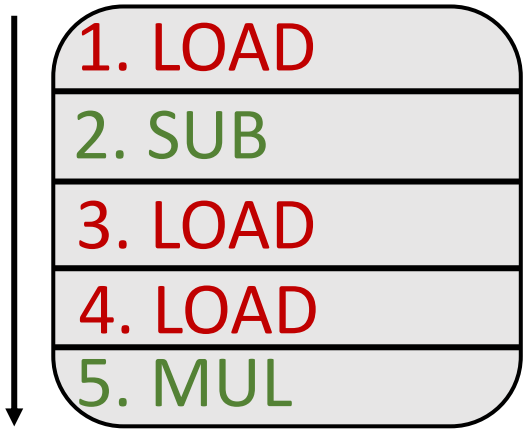


O3

Two or more instructions can execute in any order if they have no dependences (RAW, WAW, WAR)

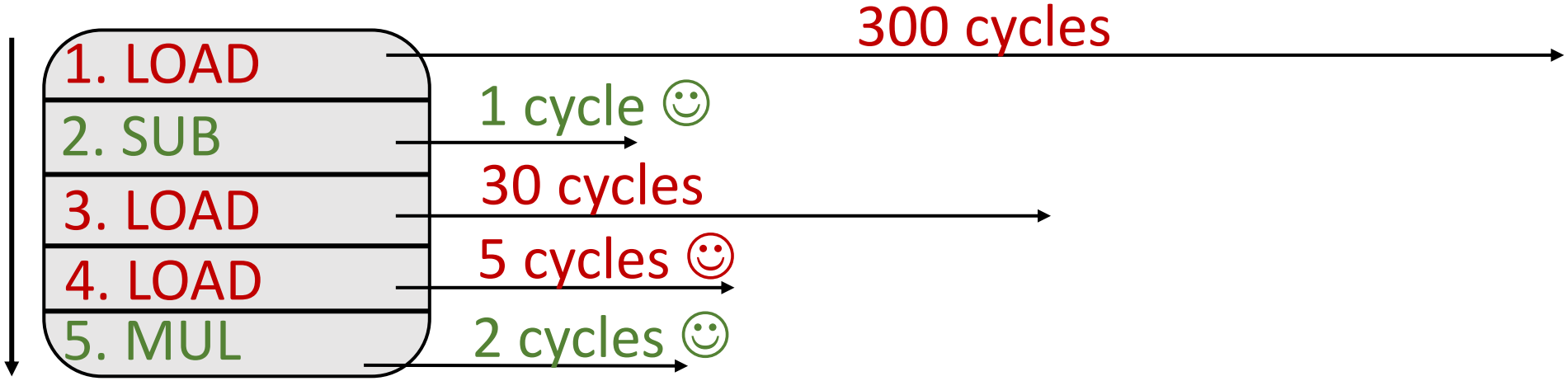
Completely orthogonal to superscalar/pipelining

O3 + Superscalar



In-order Instruction Fetch
(Multiple fetch in one cycle)

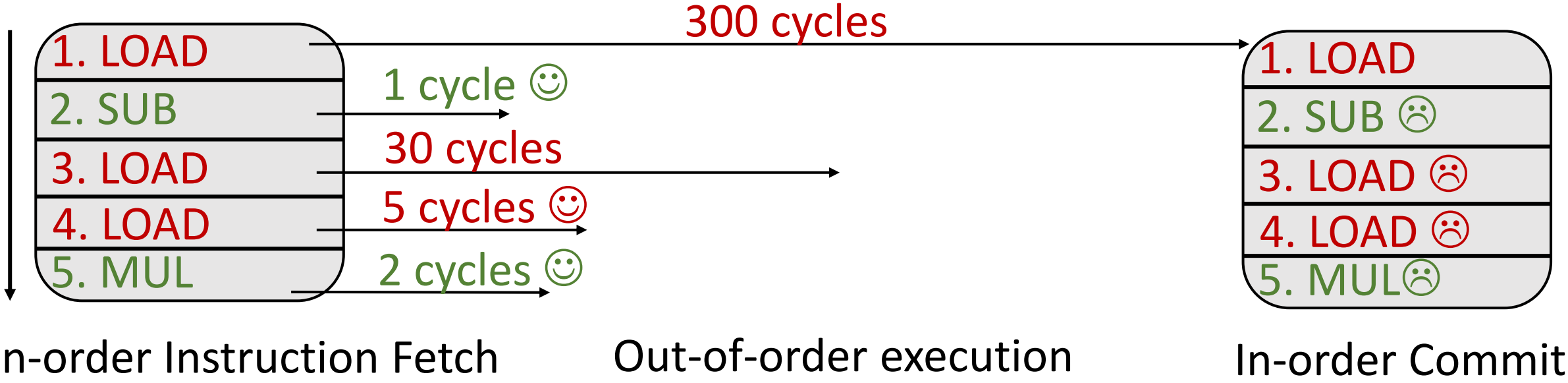
O3 + Superscalar



In-order Instruction Fetch
(Multiple fetch in one cycle)

Out-of-order execution

O3 + Superscalar



The Big Picture

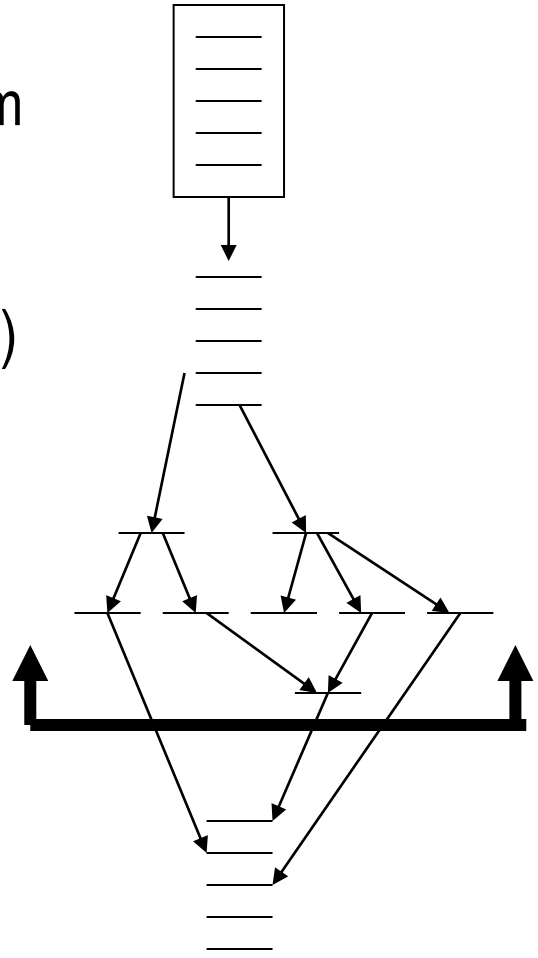
Program Form

Static program

dynamic inst.
Stream (trace)

execution
window

completed
instructions



Processing Phase

Dispatch/ dependences

inst. Issue

inst execution

inst. Reorder &
commit

The notion of Commit

After commit, the results of a committed instruction is visible to the programmer

and

the order at which instructions are fetched is also visible.

Why we need in-order commit?

Think about exceptions and precise exceptions

We should know till when we are done as per the programmer's view.

Tatenda