Beyond ILP
Multi-threading

Computer System Architecture
IIT Tirupati

April, 2020

jtt@iittp.ac.in

Instruction Level Parallelism

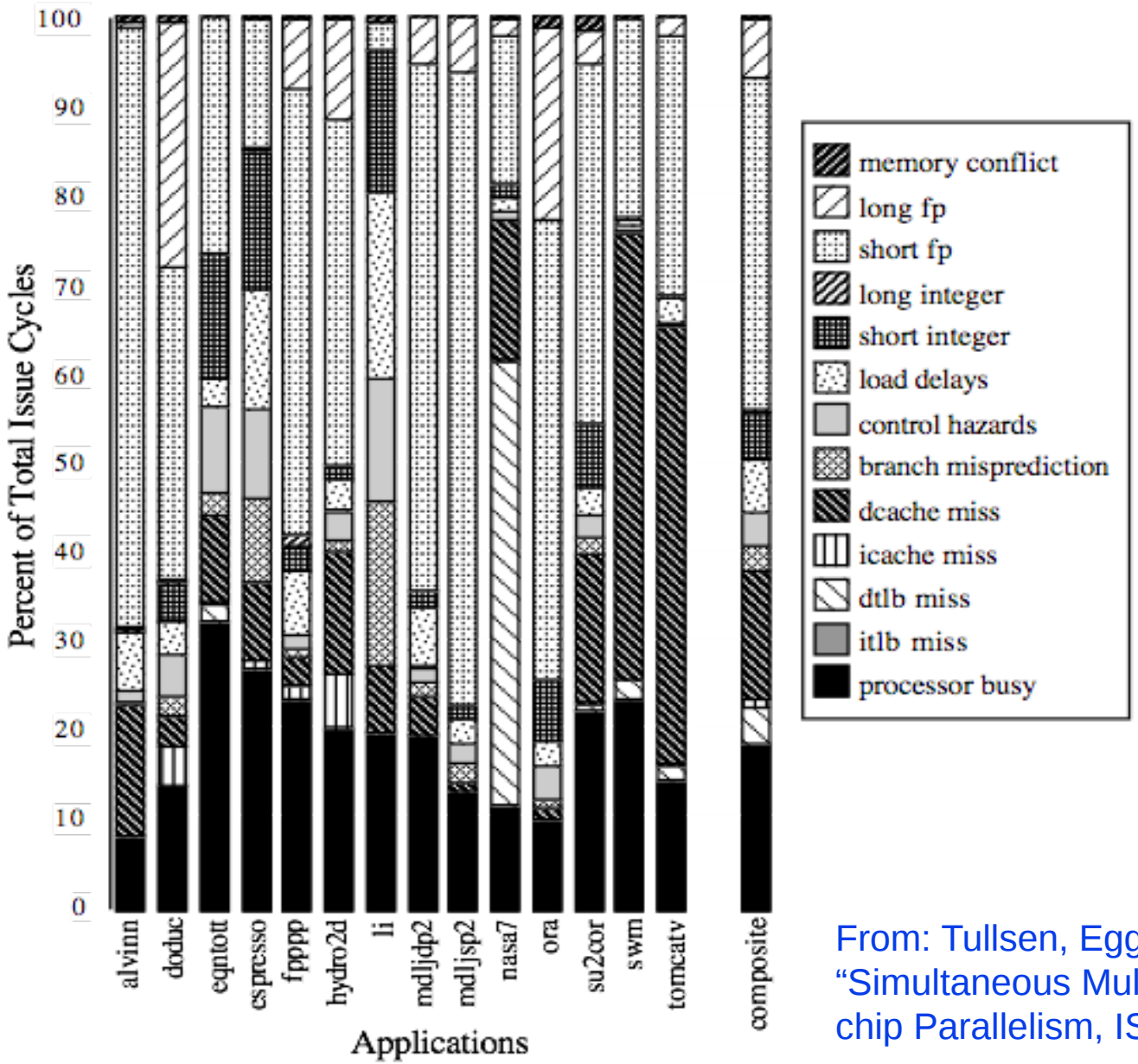
- Simple pipeline:
  Temporal parallelism among instructions

- Super scalar pipeline:
  Spatial parallelism with out-of-order execution

How to enhance performance of the super-scalar?

Would it be wise to increase the fetch and issue width to enhance the performance?

Inefficiency in Super-scalar



For an 8-way superscalar.

Processor busy indicate useful,

rest are waste

Only 25% of issue
cycle are useful!

WHY?

From: Tullsen, Eggers, and Levy, "Simultaneous Multithreading: Maximizing On-chip Parallelism, ISCA 1995.

Limits to ILP

- Static ILP (Compiler based):
  - Limited by unavailability of runtime data
- Dynamic ILP (Hardware based):
  - Hardware cost
  - Energy consumption

- Do we need to invent new HW/SW mechanisms to keep on processor performance curve?
  - Intel MMX, SSE (Streaming SIMD Extensions): 64 bit ints
  - Intel SSE2: 128 bit, including 2 64-bit Fl. Pt. per clock
  - Motorola AltaVec: 128 bit ints and FPs
  - Supersparc Multimedia ops, etc.

Limits to ILP: The Ideal Processor

Assumptions for ideal/perfect super-scalar processor:

1. *Register renaming* – infinite virtual/rename registers,
   => all register WAW & WAR hazards are avoided

2. *Branch prediction* – perfect; no mispredictions, => no flush

3. *Jump prediction* – all jumps perfectly predicted (returns, case statements)
   2 & 3 implies no control dependencies; perfect speculation & an unbounded buffer of instructions available

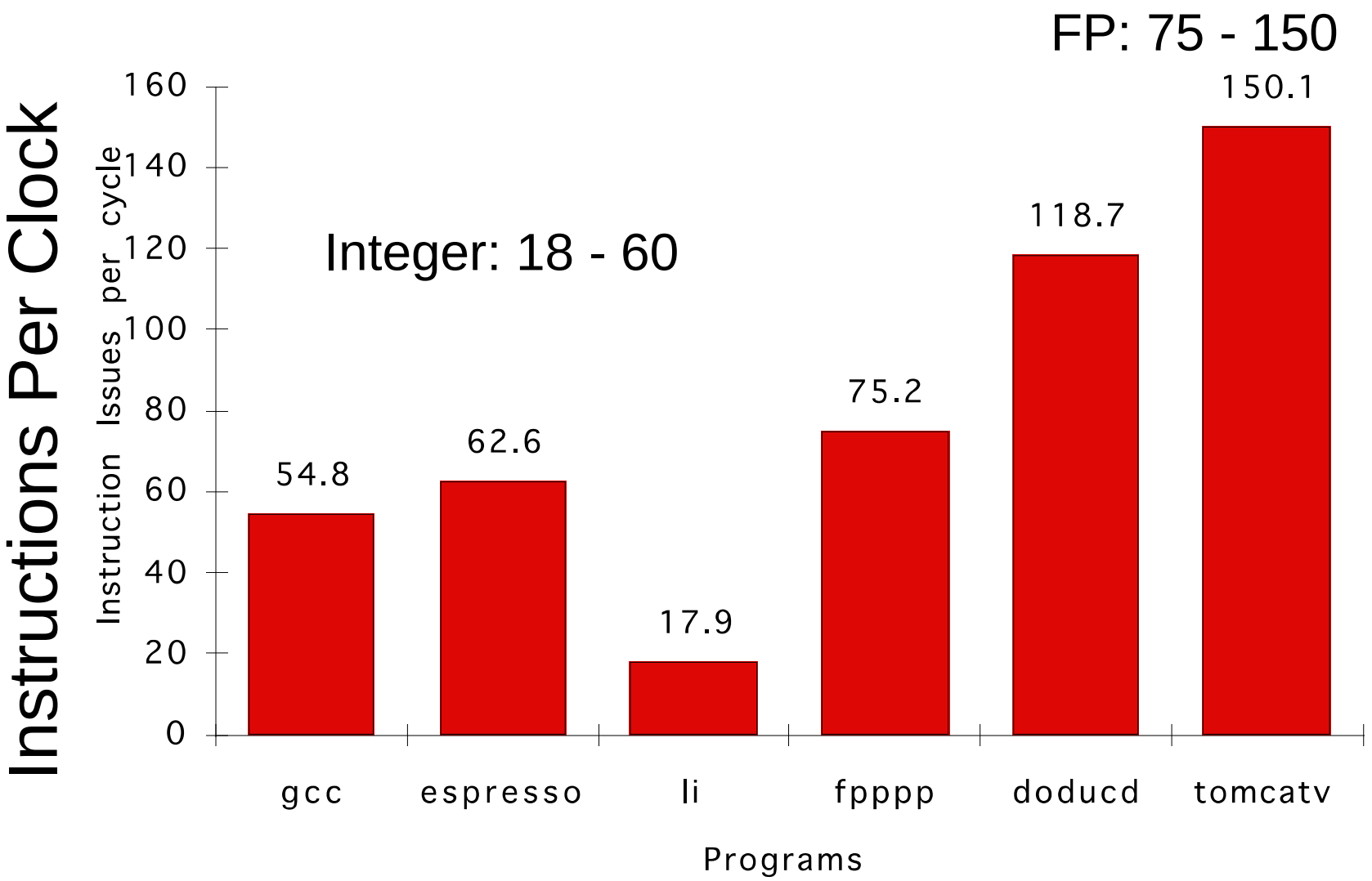
4. *Memory-address alias analysis* – addresses known & a load can be moved before a store provided addresses not equal; 1&4 eliminates all but RAW dependencies.

5. Perfect caches; 1 cycle latency for all instructions (FP *,/); unlimited instructions issued/clock cycle;

Limits to ILP: HW Model comparison

	Ideal Processor	IBM Power 5
Instructions Issued per clock	Infinite	4
Instruction Window Size	Infinite	200
Renaming Registers	Infinite	48 integer + 40 Fl. Pt.
Branch Prediction	Perfect (100% accuracy)	2% to 6% misprediction (Tournament Branch Predictor)
Cache	Perfect (no miss)	64KI, 32KD, 1.92MB L2, 36 MB L3
Memory Alias Analysis	Perfect	??

Upper Limit to ILP: Ideal Machine



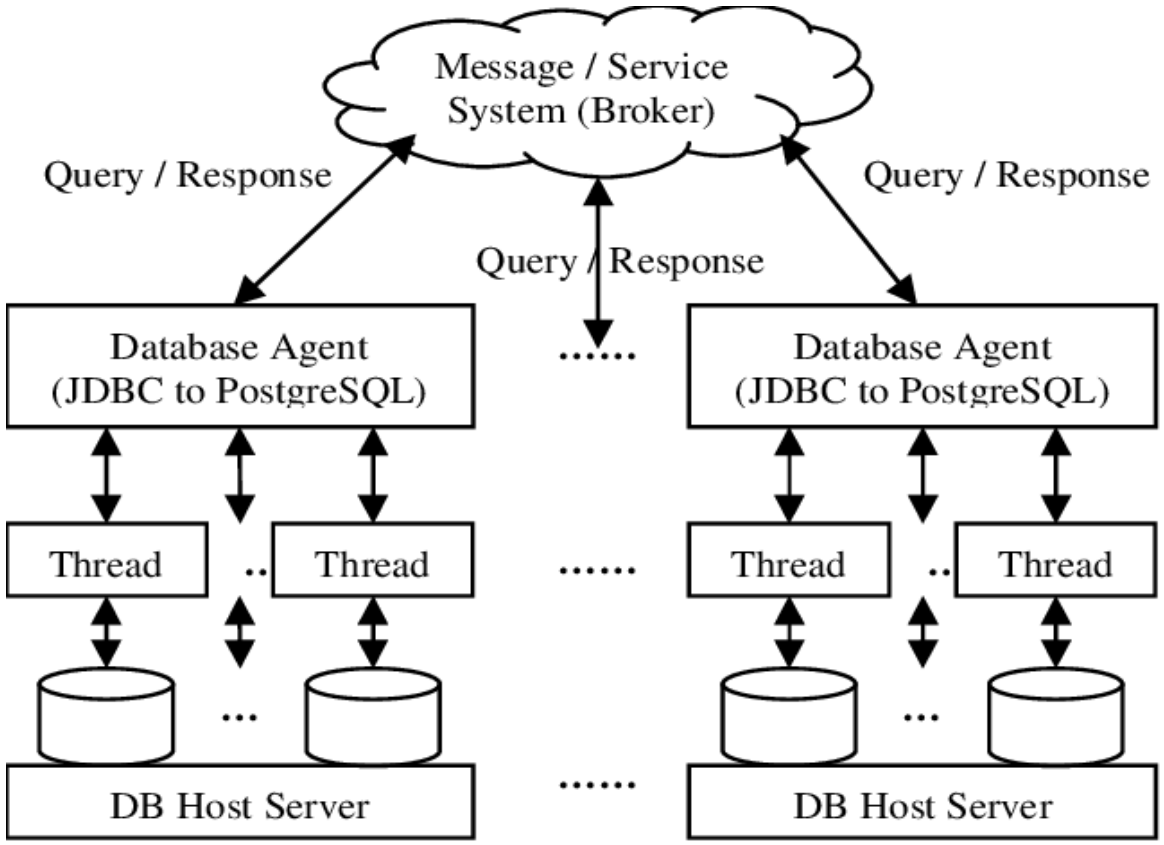
Beyond Single Thread ILP

We need more independent instructions

from somewhere?

There are many applications:

- Data base applications
- Scientific computation
- Graphics, media etc.



Threads

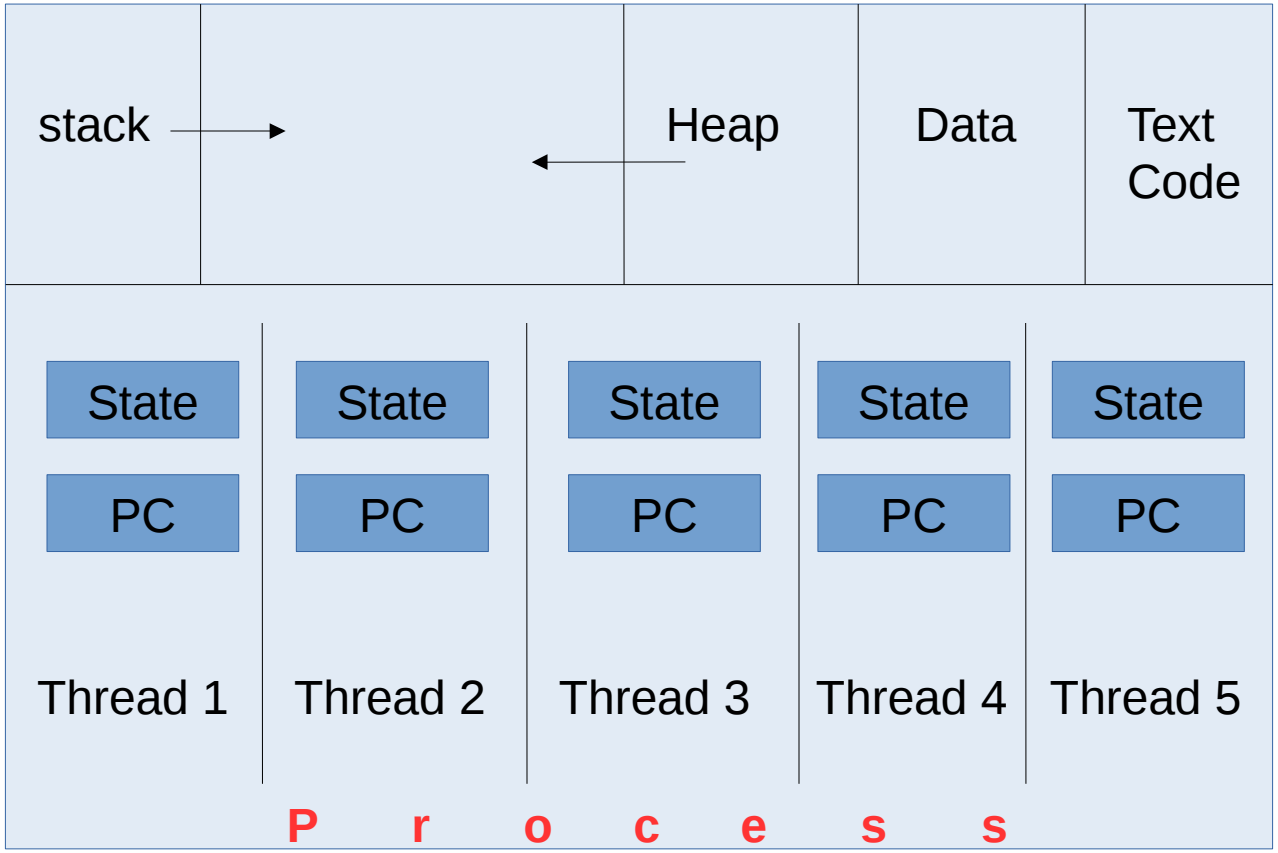
are Inherently parallel!

Thread Level Parallelism

Can we build a hardware where instructions from multiple threads can be executed together?

Basic understanding on:

Program
Process
Thread



Thread Level Parallelism

What does it mean for hardware design?

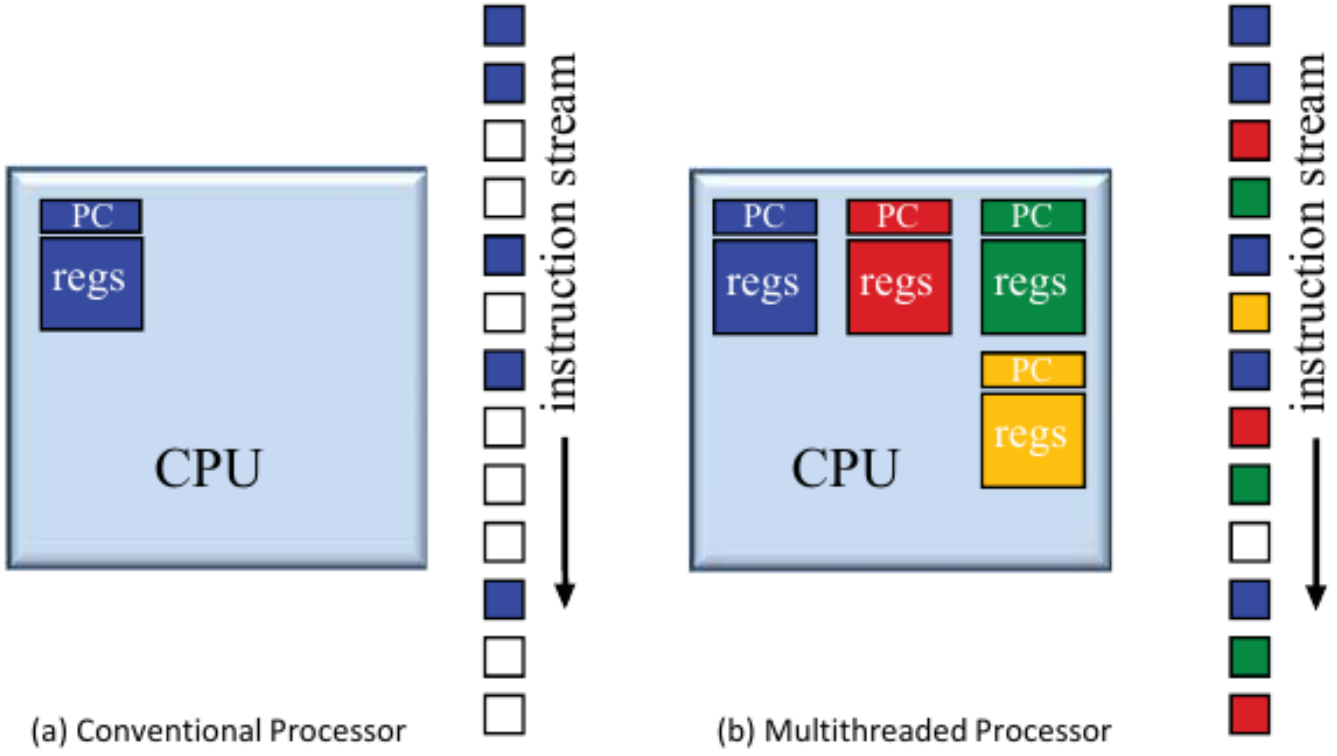
- A separate register files for each thread!
- A separate program counter for each thread!
- Memory has to be shared!
- Hardware should be able to switch among multiple threads
- How to identify each thread?

- What about execution units?
  No duplication but shared among the threads!

- The operating system needs to be multi-programming (multi-threading)

Thread Level Parallelism

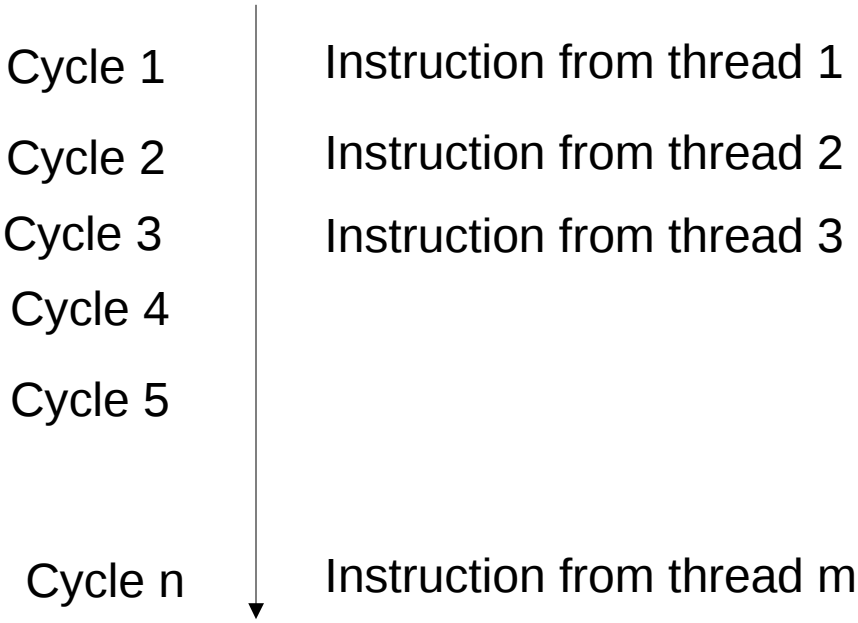
What does it mean for hardware design?



Thread Level Parallelism

Different type of Multi-threading Architecture

What are the ways to execute instructions from different threads?



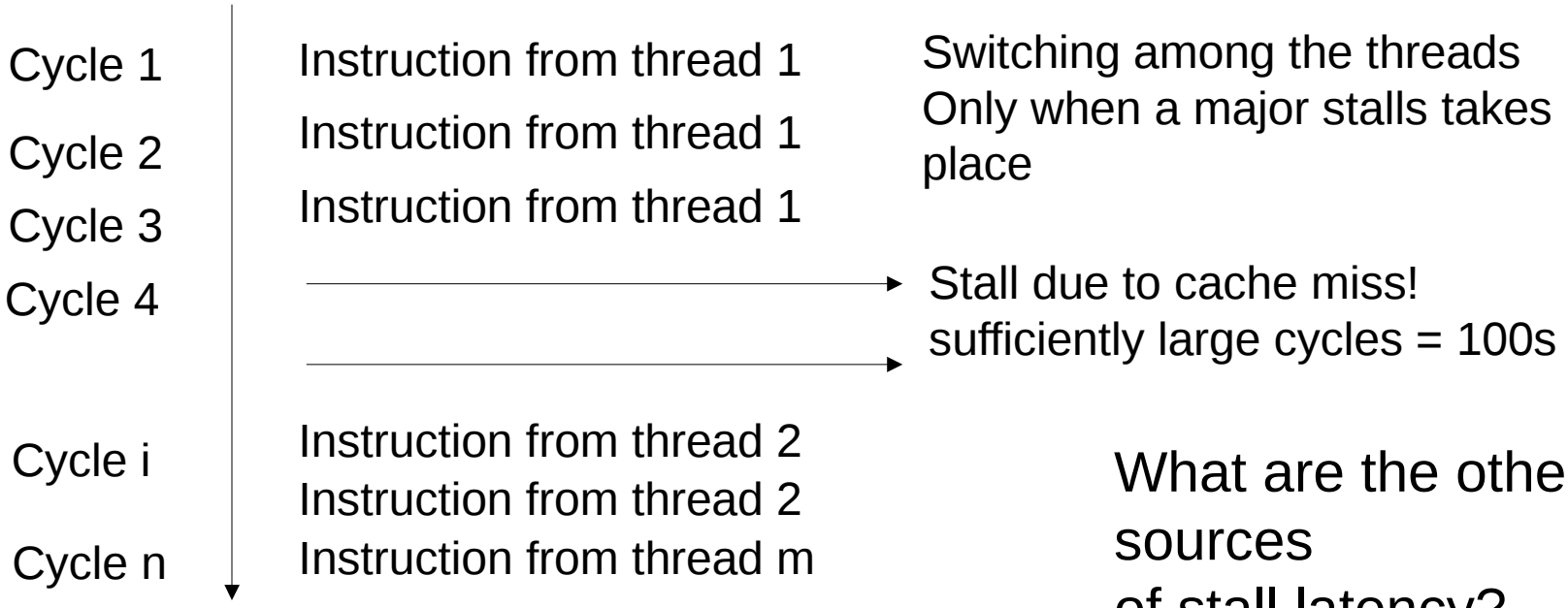
Switching among the threads
Every cycle!

Fine-grained multi-threading (FMT)

Thread Level Parallelism

Different type of Multi-threading Architecture

What are the ways to execute instructions from different threads?

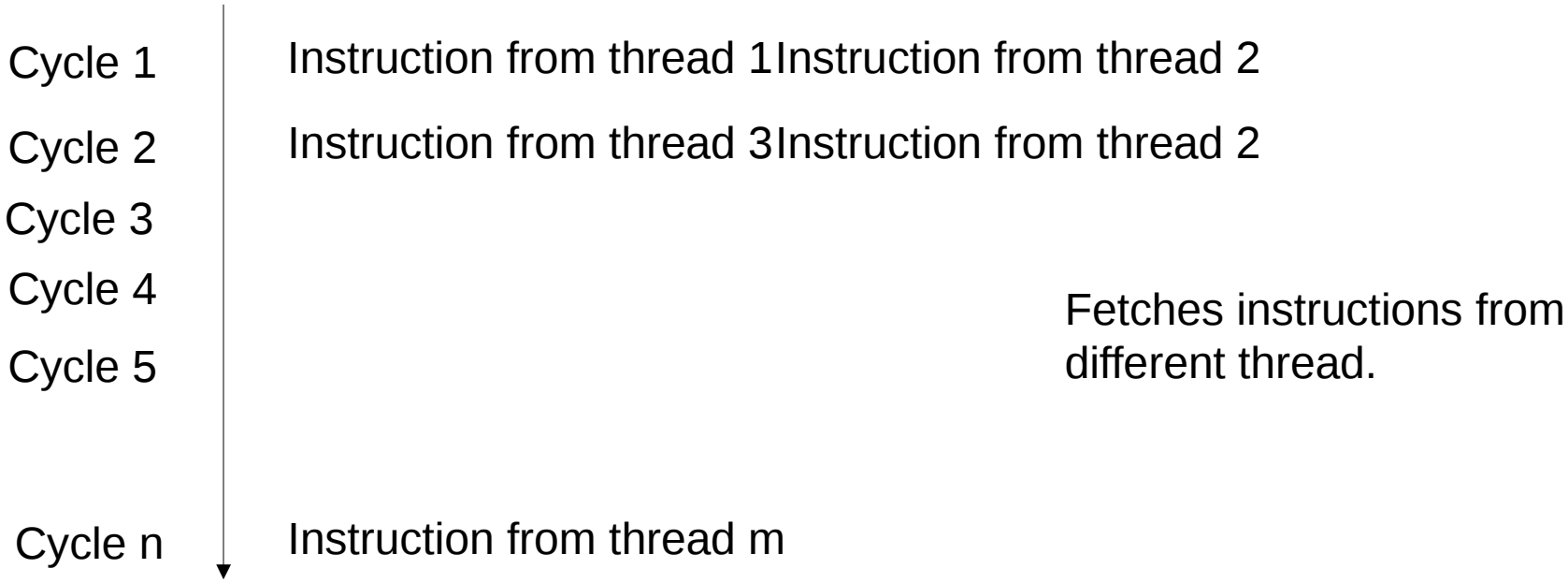


Coarse-grained multi-threading (CMT)

Thread Level Parallelism

Simultaneous Multi-threading

= fine-grained multi-threading + super scalar



Simultaneous multi-threading (SMT)

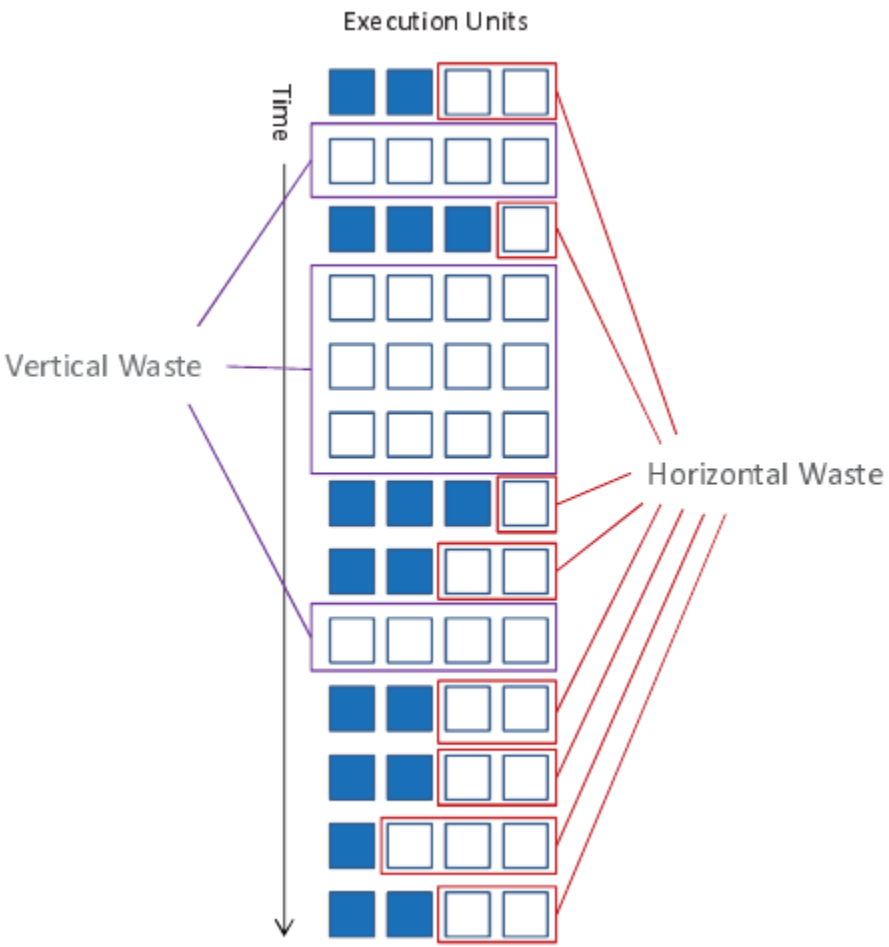
Thread Level Parallelism

Simultaneous Multi-threading in Super-scalar

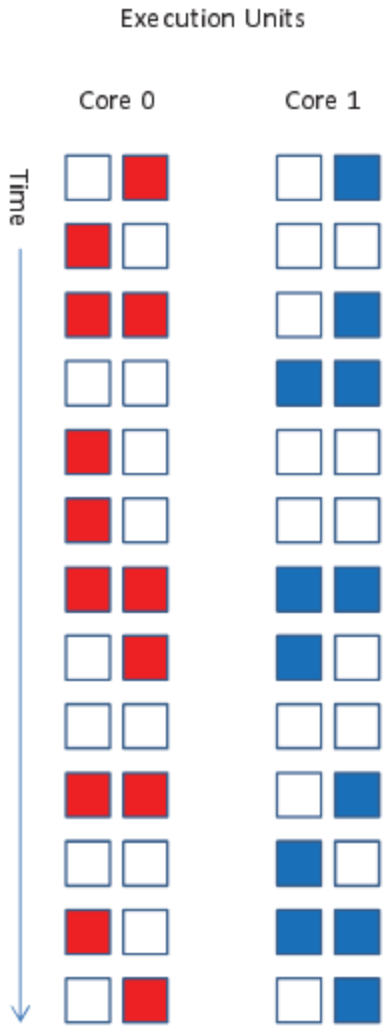
- Super-scalar with no multi-threading support
- Super-scalar with coarse-grained multi-threading
- Super-scalar with fine-grained multi-threading
- Super-scalar with simultaneous multi-threading

Thread Level Parallelism:

Execution Model



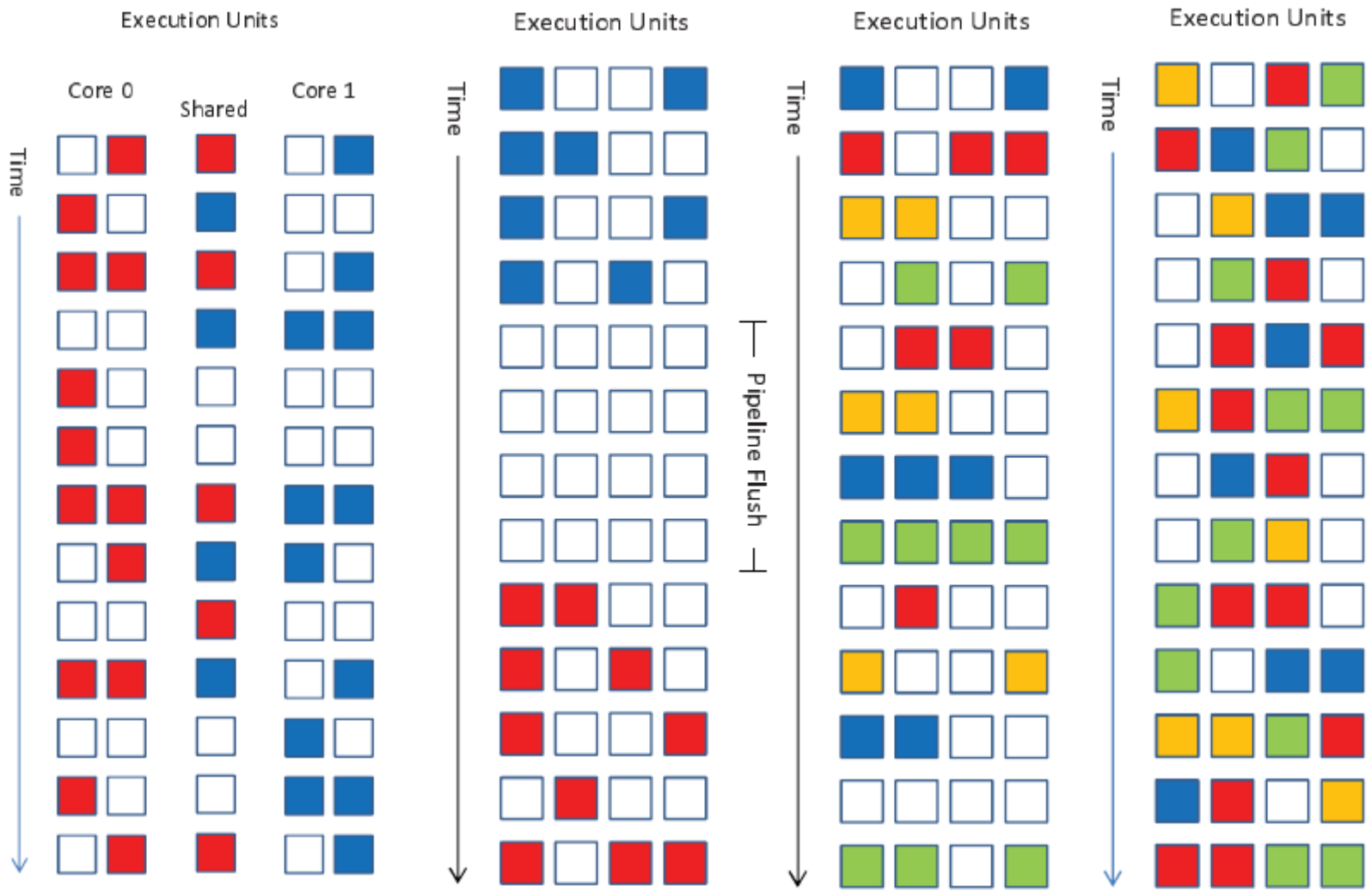
Dynamic Super-scalar



Chip Multi-processor (CMP)

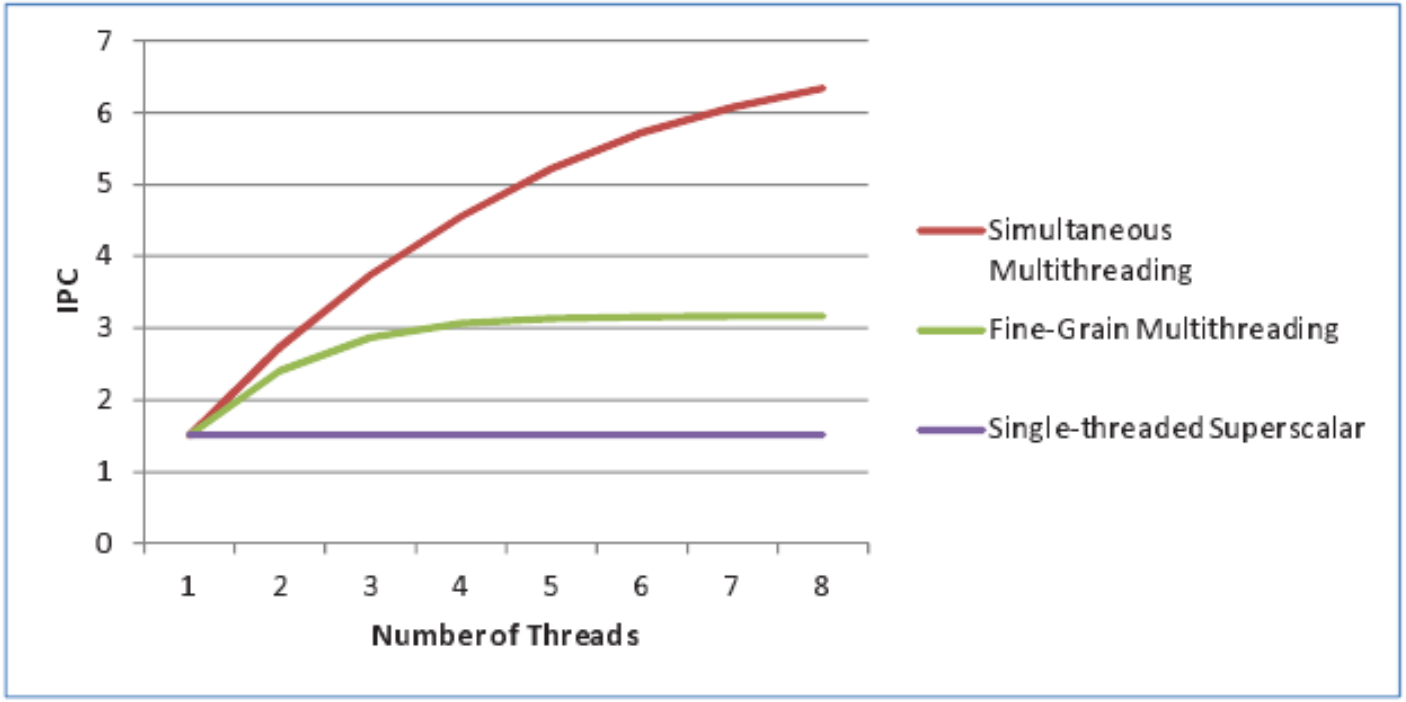
Thread Level Parallelism:

Execution Model



Conjoined Core

Multithreading: Performance



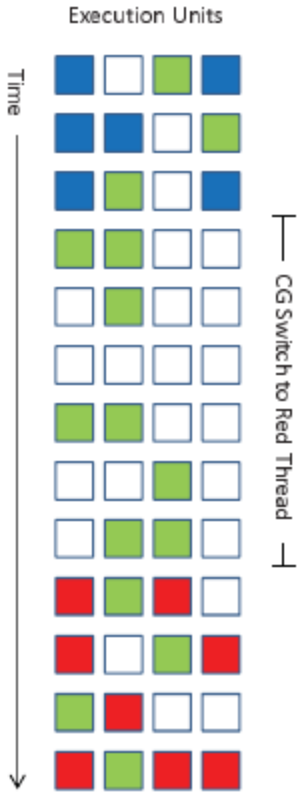
Dean M. Tullsen, Susan J. Eggers, and Henry M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In Proceedings of the International Symposium on Computer Architecture, June, 1995

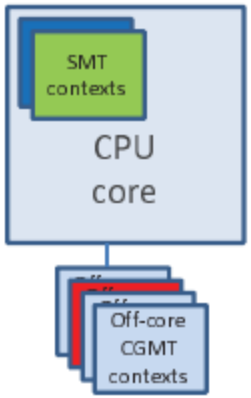
Multithreading Execution Model

How to decide which model would give better performance?

Combining:

SMT +
CGMT





Next Lecture