Advanced Computer Architectures

(High Performance Processors and Systems)

Instruction-Level Parallelism: Limits

Politecnico di Milano

v1

Alessandro Verosimile <alessandro.verosimile@polimi.it>
Marco D. Santambrogio <marco.santambrogio@polimi.it>

Outline

- *Review*
  - ILP Definition
  - Superscalar Architectures, Static and Dynamic Schedulers

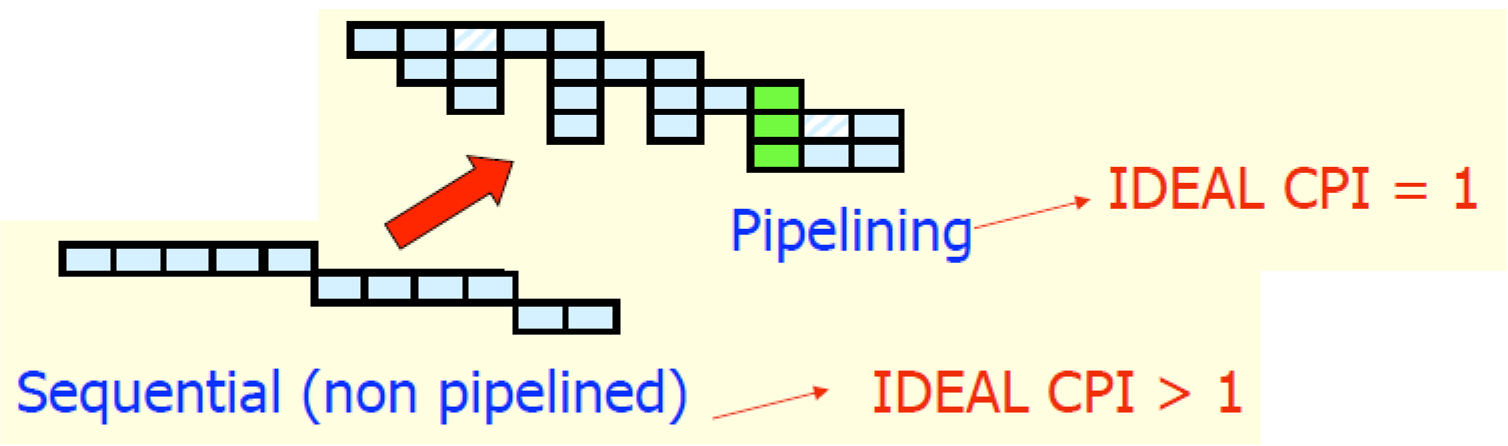
- Limits to ILP
  - Ideal machine
  - Limits
  - Examples of real architectures

Definition of ILP

- ILP = Potential overlap of execution among unrelated instructions

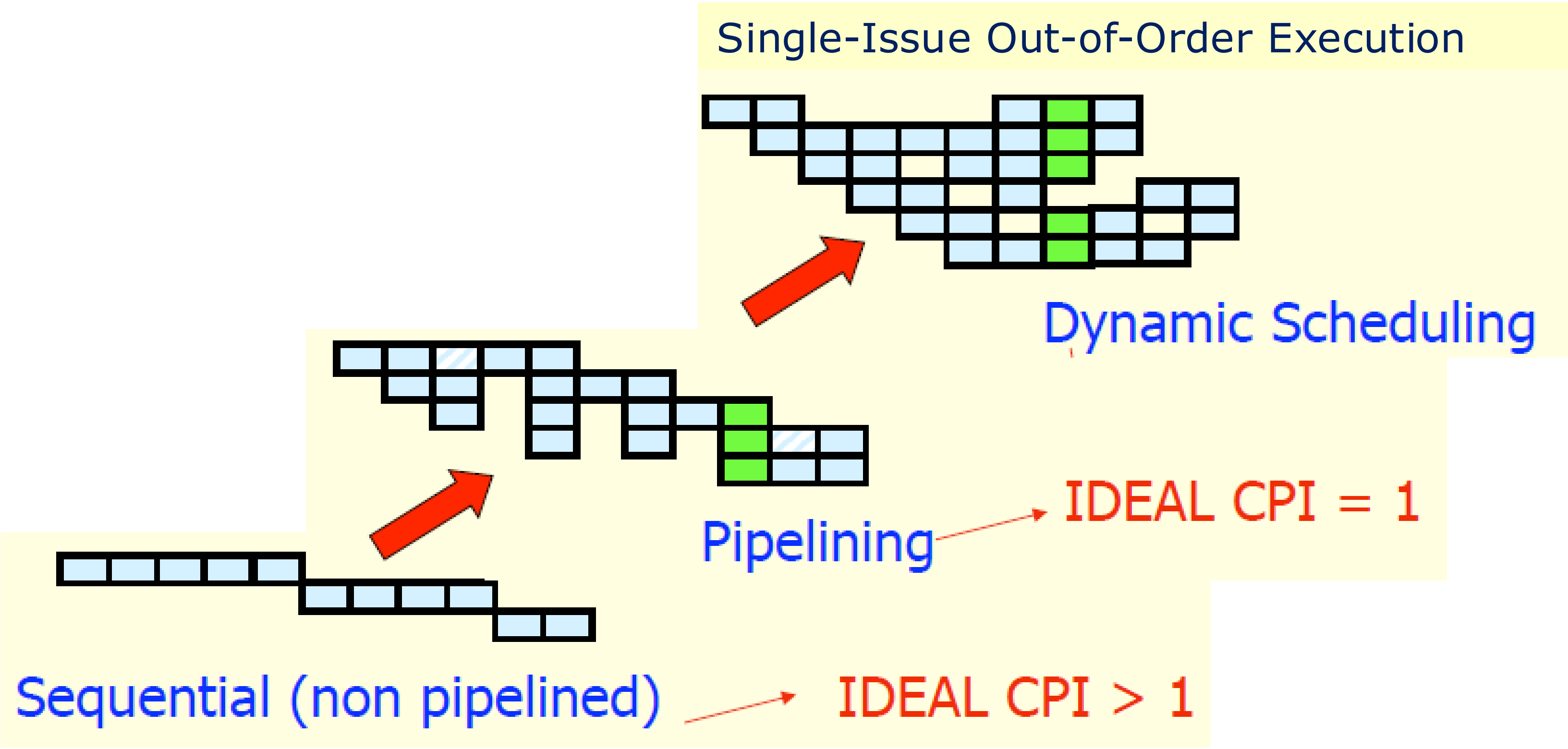
- Overlapping possible if:
    - No Structural Hazards
    - No RAW, WAR of WAW Stalls
    - No Control Stalls

Several steps towards exploiting more ILP



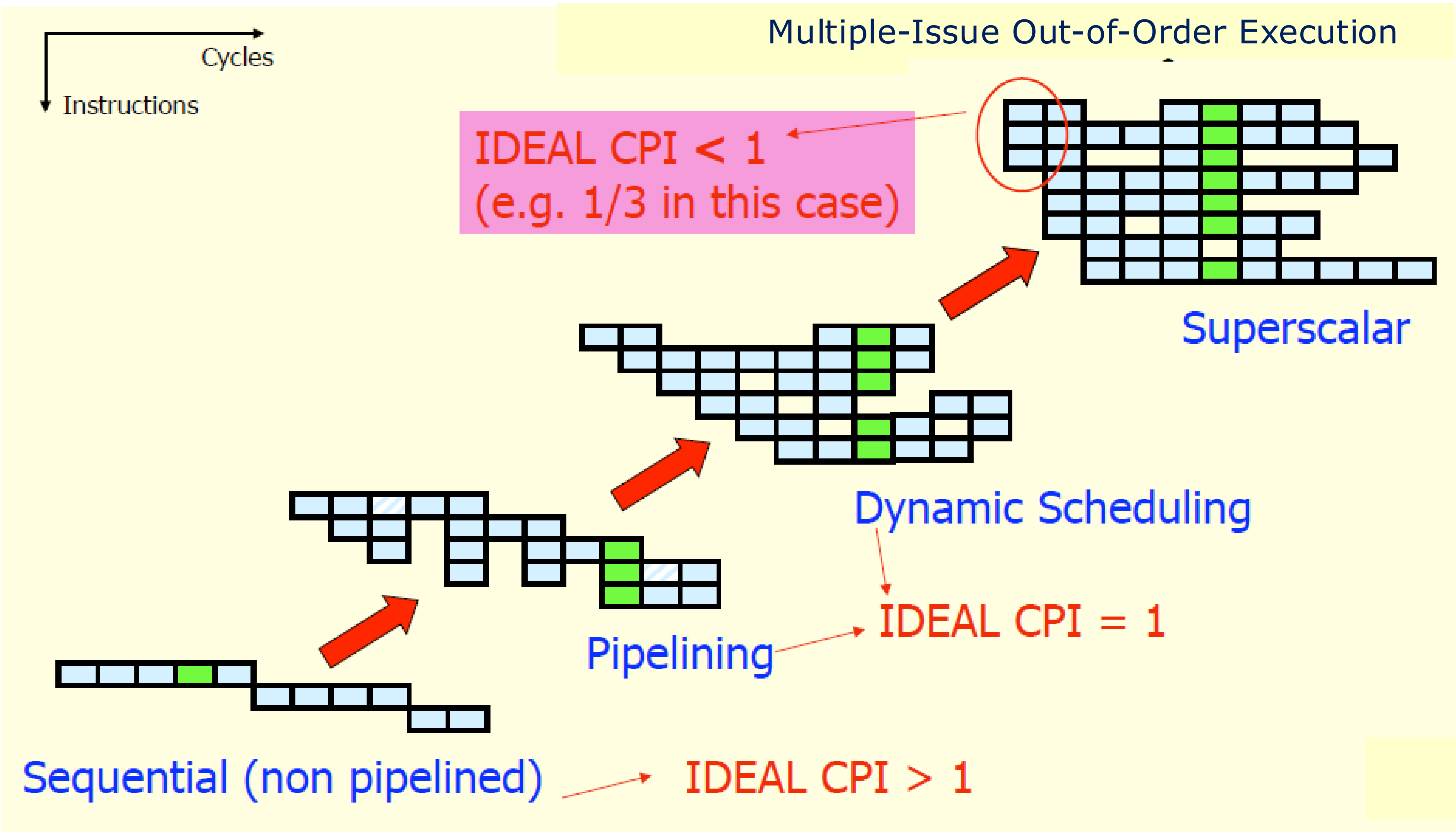
Several steps towards exploiting more ILP

ACA CLASSES 7, 8, 10, 11



TODAY!

Several steps towards exploiting more ILP



Superscalar execution

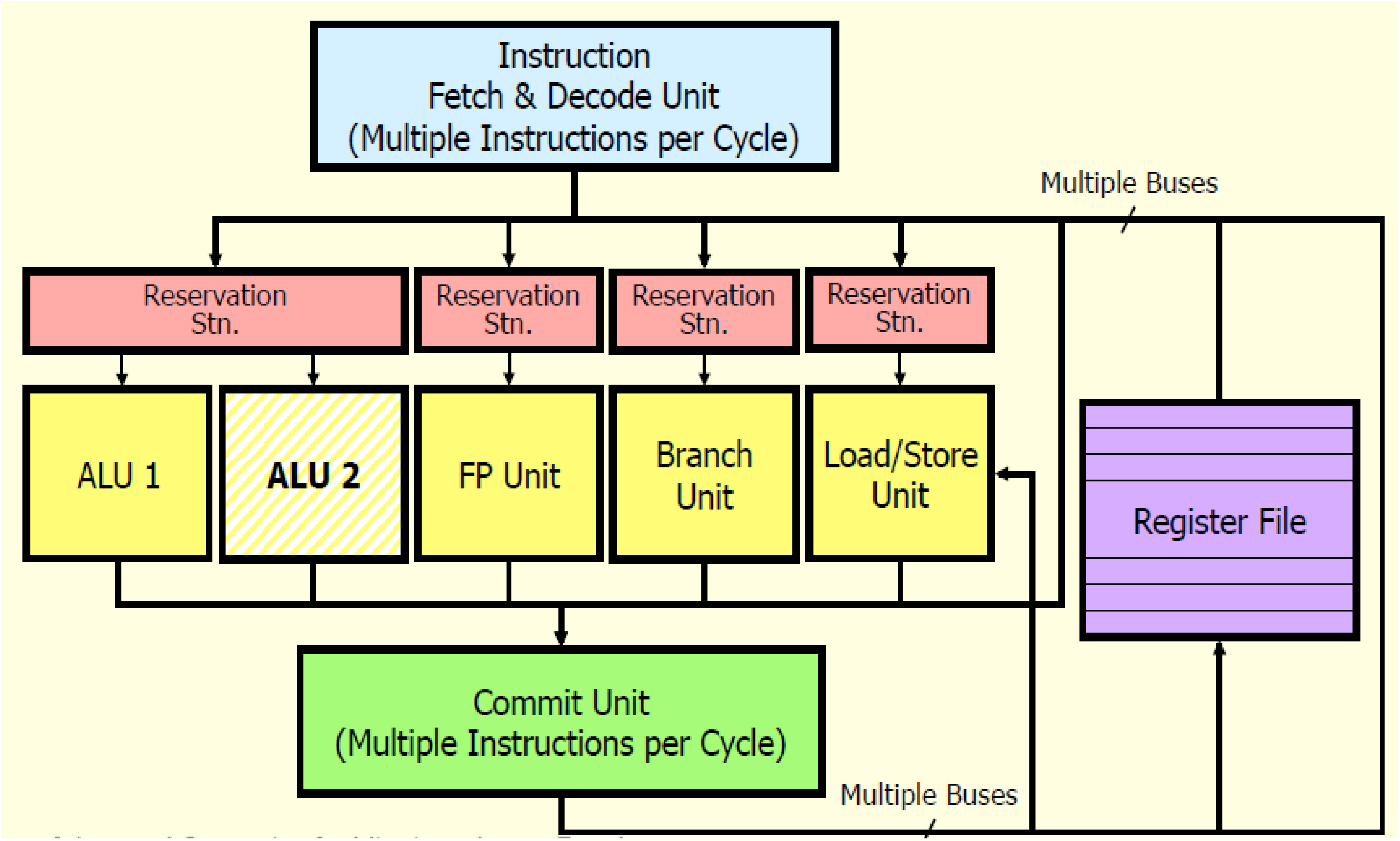
- Why not more than one instruction beginning execution at each clock cycle?
- Key requirements:
  - **Fetching more instructions per clock cycle** (Fetch Unit): no major problem provided the instruction cache can sustain the bandwidth and can manage more requests at the same time
  - **Decide on data and control dependencies**: dynamic scheduling and dynamic branch prediction

Beyond CPI = 1

- Superscalar:
  - Issue multiple instructions per clock-cycle
  - varying no. instructions/cycle (1 to 8),
  - scheduled by compiler or by HW (Tomasulo)
  - e.g. IBM PowerPC, Sun UltraSparc, DEC Alpha, HP 8000, Pentium

- Anticipated success lead to use of Instructions Per Clock cycle (IPC) vs. CPI
- CPI_{ideal} = 1 / issue-width

Superscalar Processor



Limits to ILP

Assumptions for **ideal/perfect machine** to start:

1. Register renaming
2. Branch prediction
3. Jump prediction
4. Memory-address alias analysis
5. 1 cycle latency for all instructions

Limits to ILP

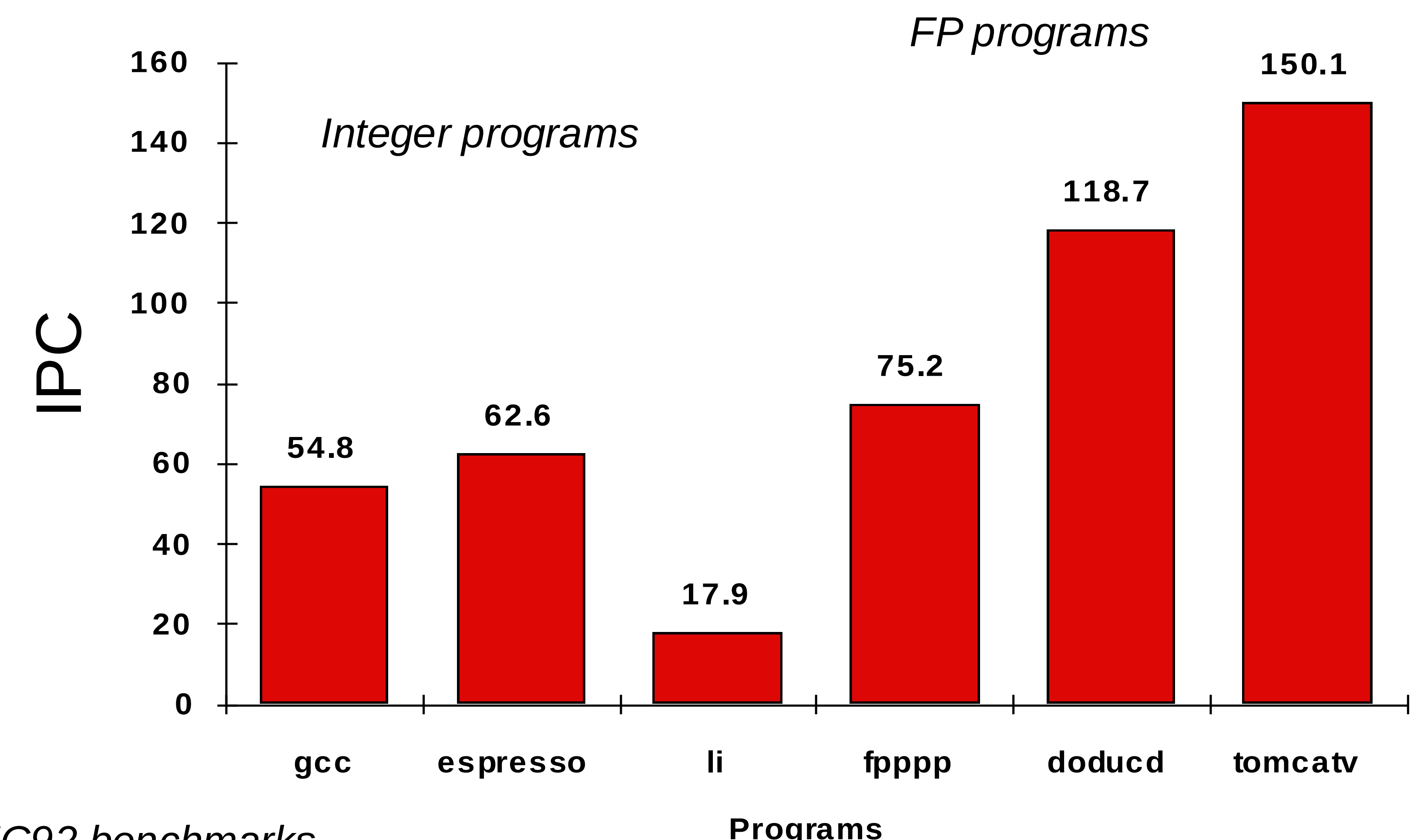
Assumptions for **ideal/perfect machine** to start:

1. Register renaming
   – infinite virtual registers and all WAW & WAR hazards are avoided
2. Branch prediction
   – perfect; no mispredictions
3. Jump prediction
   – all jumps perfectly predicted => machine with perfect speculation & an unbounded buffer of instructions available
4. Memory-address alias analysis
   – addresses are known & a store can be moved before a load provided addresses not equal
5. 1 cycle latency for all instructions
   – unlimited number of instructions issued per clock cycle

Initial assumptions

- CPU can issue at once unlimited number of instructions, looking arbitrarily far ahead in computation;
- No restrictions on types of instructions that can be executed in one cycle (including loads and stores);
- All functional unit latencies = 1; any sequence of depending instructions can issue on successive cycles;
- Perfect caches = all loads, stores execute in one cycle ⇨ only fundamental limits to ILP are taken into account.
- Obviously, results obtained are VERY optimistic! (no such CPU can be realized…);
- Benchmark programs used: six from SPEC92 (three FP-intensive ones, three integer ones).

Upper Limit to ILP: Ideal Machine



SPEC92 benchmarks

Limits on window size

- Dynamic analysis is necessary to approach perfect branch prediction (impossible at compile time!);
- A perfect dynamic-scheduled CPU should:

Limits on window size

- Dynamic analysis is necessary to approach perfect branch prediction (impossible at compile time!);
- A perfect dynamic-scheduled CPU should:
  1. Look arbitrarily far ahead to find set of instructions to issue, predict all branches perfectly;
  2. Rename all registers uses (⇨ no WAW, WAR hazards);
  3. Determine whether there are data dependencies among instructions in the issue packet; rename if necessary;
  4. Determine if memory dependencies exist among issuing instructions, handle them;
  5. Provide enough replicated functional units to allow all ready instructions to issue.

Limits on instruction windows

- Size affects the number of comparisons necessary to determine RAW dependences
- Example: # comparisons to evaluate data dependences among n register-to-register instructions in the issue phase (with an infinite # of regs) =

$$2n-2+2n-4+\ldots+2 = 2\sum_{i=1}^{n-1} i = 2\frac{(n-1)n}{2} = n^2-n$$

  - Window size = 2000 ➪ almost 4 Million comparisons!
  - Issue window of 50 instructions requires 2450 comparisons!

- Today's CPUs: constraints deriving from the limited number of registers + search for dependent instructions + in-order issue

Limits on window size, maximum issue count

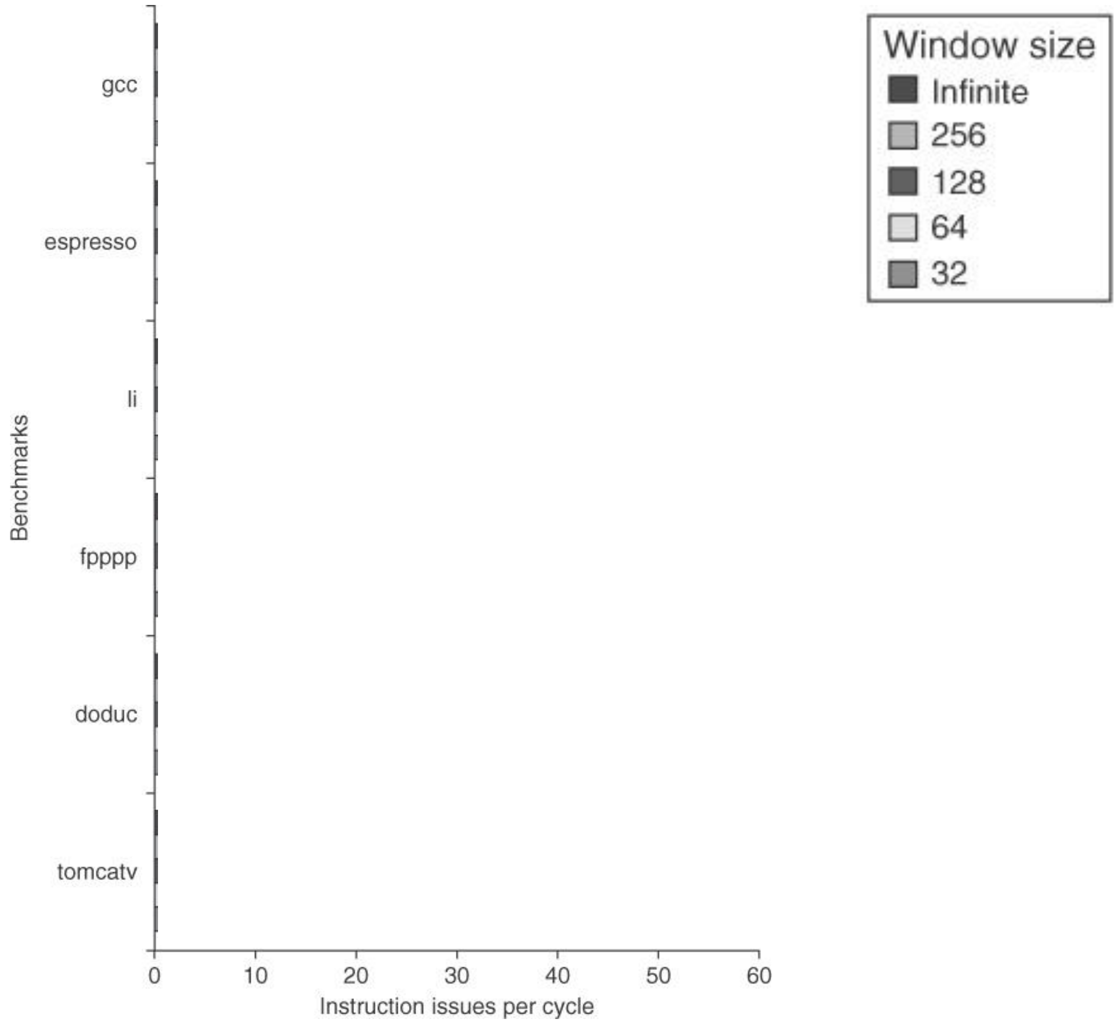
All instructions in the window must be kept in the processor

number of comparisons required at each cycle =
maximum completion rate x
window size x
number of operands per instruction ⇨
total window size limited by storage + comparisons + limited issue rate

(today: window size 32-200 ⇨ up to over 2400 comparisons!)

Amount of parallelism vs window size

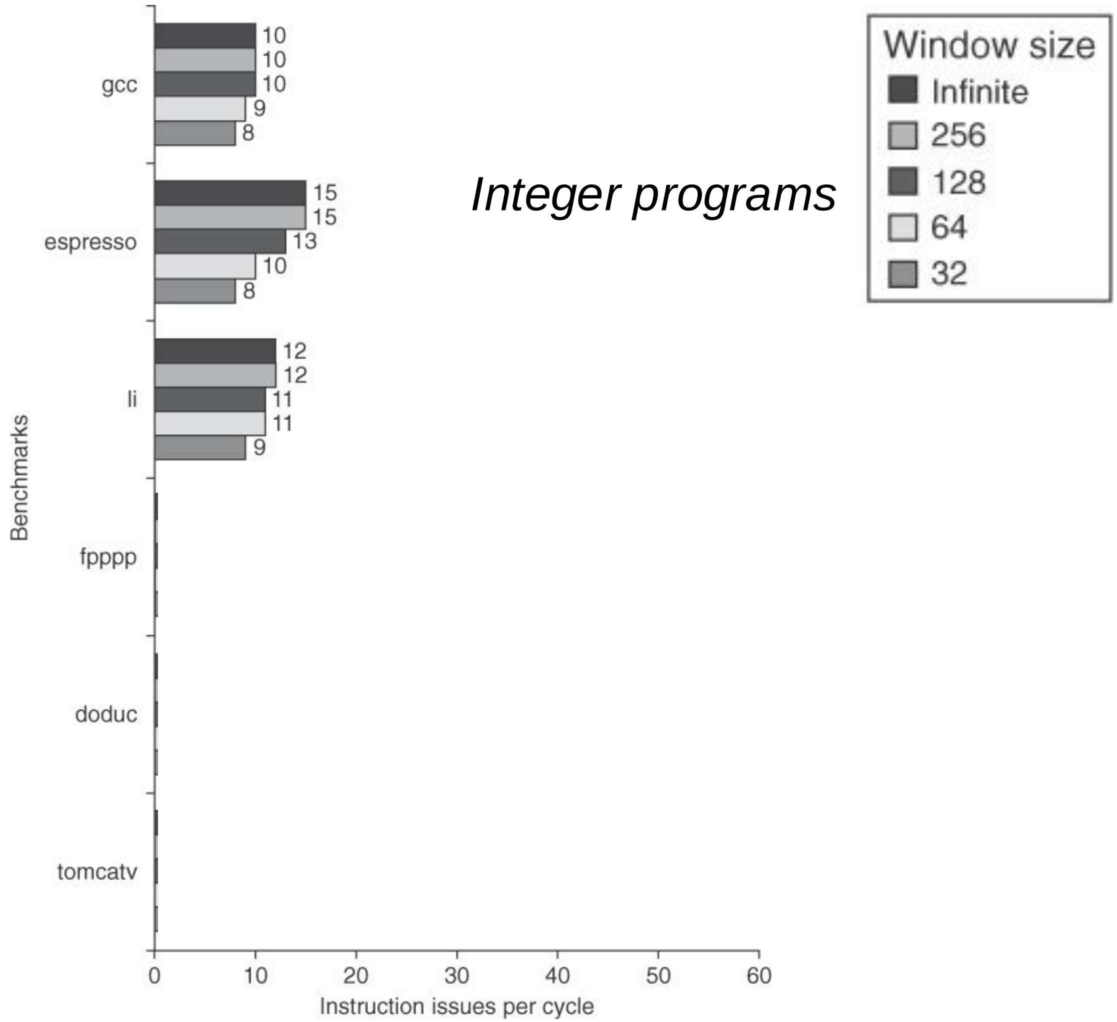
Up to 64 arbitrary issues
Per clock cycle



SPEC92 benchmarks

Amount of parallelism vs window size

Up to 64 arbitrary issues
Per clock cycle

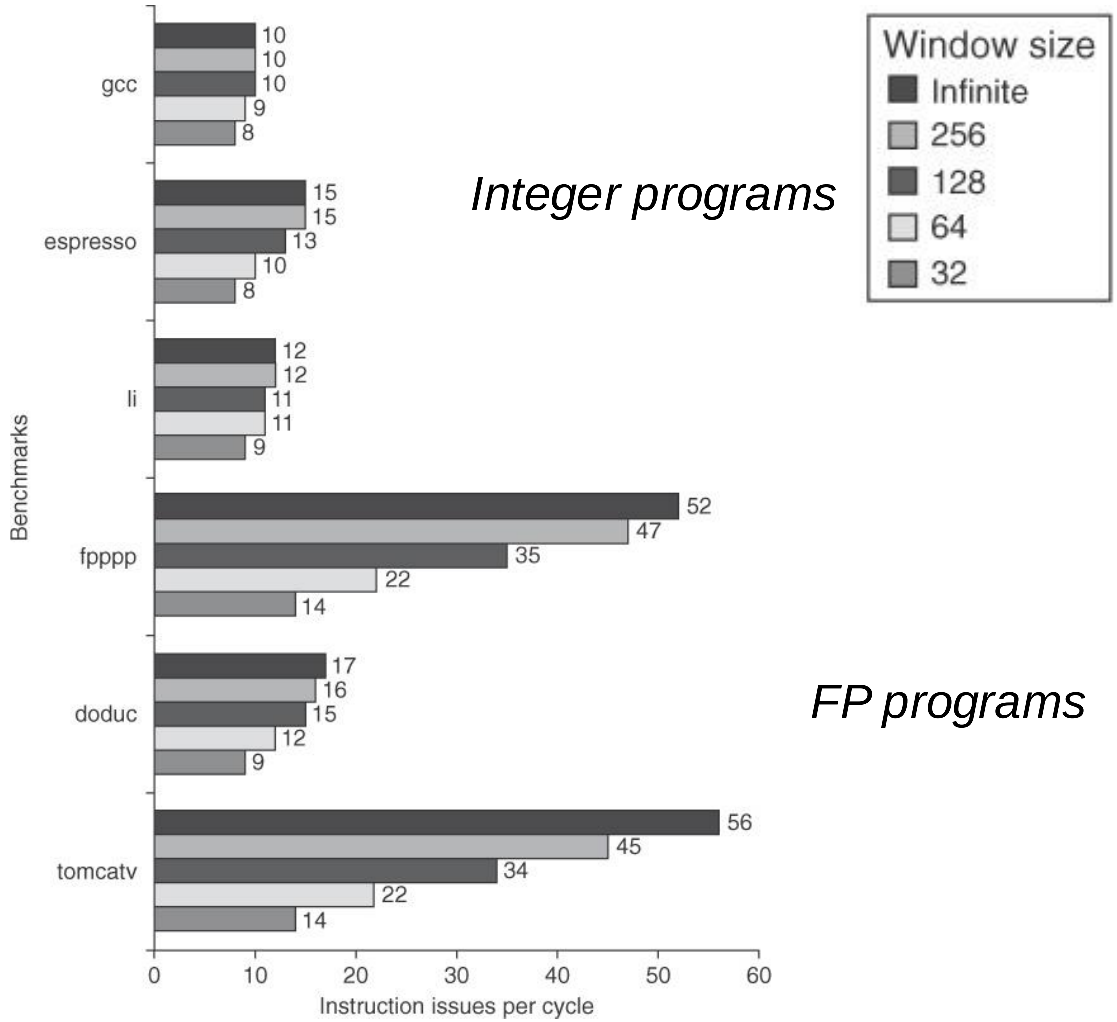


Integer programs

SPEC92 benchmarks

Amount of parallelism vs window size

Up to 64 arbitrary issues
Per clock cycle



Integer programs

FP programs

SPEC92 benchmarks

HW model comparison

	Ideal model	IBM Power 5 (2004-2006) Dual core @ 1.5 – 2.3 GHz
Instructions Issued per clock	Infinite	4
Instruction Window Size	Infinite	200
Renaming Registers	Infinite	48 integer + 40 FP
Branch Prediction	Perfect	2% to 6% misprediction (Tournament Branch Predictor)
Cache	Perfect	L1 (32KI+32KD)/core L2 1.875MB/core L3 36 MB/chip (off chip)
Memory Alias Analysis	Perfect	??

Other limits of today's CPUs

- N. of functional units
  - For instance: not more than 2 memory references per cycle
- N. of busses
- N. of ports for the register file

- All these limitations define that the maximum number of instructions that can be issued, executed or committed in the same clock cycle is much smaller than the window size

Issue-width limited in practice

- Now, the maximum (rare) is 6, but no more exists.
  - The widths of current processors range from single-issue (ARM11, UltraSPARC-T1) through 2-issue (UltraSPARC-T2/T3, Cortex-A8 & A9, Atom, Bobcat) to 3-issue (Pentium-Pro/II/III/M, Athlon, Pentium-4, Athlon 64/Phenom, Cortex-A15) or 4-issue (UltraSPARC-III/IV, PowerPC G4e, Core 2, Core i, Core i*2, Bulldozer) or 5-issue (PowerPC G5), or even 6-issue (Itanium, but it's a VLIW).
- Because it is too hard to decide which 8, or 16, instructions can execute every cycle (too many!)
  - It takes too long to compute
  - So the frequency of the processor would have to be decreased

Issue-width limited in practice

- Now, the maximum (rare) is 6, but no more exists.
  - The widths of current processors range from single-issue (ARM11, UltraSPARC-T1) through 2-issue (UltraSPARC-T2/T3, Cortex-A8 & A9, Ato[illegible] Bo[illegible]) [illegible]-i[illegible]([illegible]ntium[illegible]II/[illegible]M[illegible]Athlon, Pentium-4, Athlo[illegible]64/[illegible]C[illegible]te[illegible]-A[illegible]) or [illegible]s[illegible]e[illegible]PARC-III/IV, PowerP[illegible]G4[illegible], [illegible]e i*2[illegible]ze[illegible]or 5-issue (PowerPC G5), or even 6-issue (Itanium, but it's a VLIW).

TRADE-OFF

- Because it is too hard to decide which 8, or 16, instructions can execute every cycle (too many!)
  - It takes too long to compute
  - So the frequency of the processor would have to be decreased

Current Superscalar & VLIW processors

- Dynamically-scheduled superscalar processors are the commercial state-of-the-art for general purpose: current implementations of Intel Core i, PowerPC, Alpha, MIPS, SPARC, etc. are all superscalar

Current Superscalar & VLIW processors

- Dynamically-scheduled superscalar processors are the commercial state-of-the-art for general purpose: current implementations of Intel Core i, PowerPC, Alpha, MIPS, SPARC, etc. are all superscalar
- VLIW processors are primarily successful as embedded media processors for consumer electronic devices (embedded):
  - TriMedia media processors by NXP
  - The C6000 DSP family by Texas Instruments
  - The ST200 family by STMicroelectronics
  - The SHARC DSP by Analog Devices
  - Itanium 2 is the only general purpose VLIW, a ‘hybrid’ VLIW (EPIC, Explicitly Parallel Instructions Computing)

Taxonomy of Multiple Issue Machines

Common name	Issue structure	Hazard detection	Scheduling	Distinguishing characteristic	Examples
Superscalar (static)	Dynamic	Hardware	Static	In-order execution	Mostly in the embedded space: MIPS and ARM, including the ARM Coretex A8
Superscalar (dynamic)	Dynamic	Hardware	Dynamic	Some out-of-order execution, but no speculation	None at the present
Superscalar (speculative)	Dynamic	Hardware	Dynamic with speculation	Out-of-order execution with speculation	Intel Core i3, i5, i7; AMD Phenom; IBM Power 7
VLIW/LIW	Static	Primarily software	Static	All hazards determined and indicated by compiler (often implicitly)	Most examples are in signal processing, such as the TI C6x
EPIC	Primarily static	Primarily software	Mostly static	All hazards determined and indicated explicitly by the compiler	Itanium

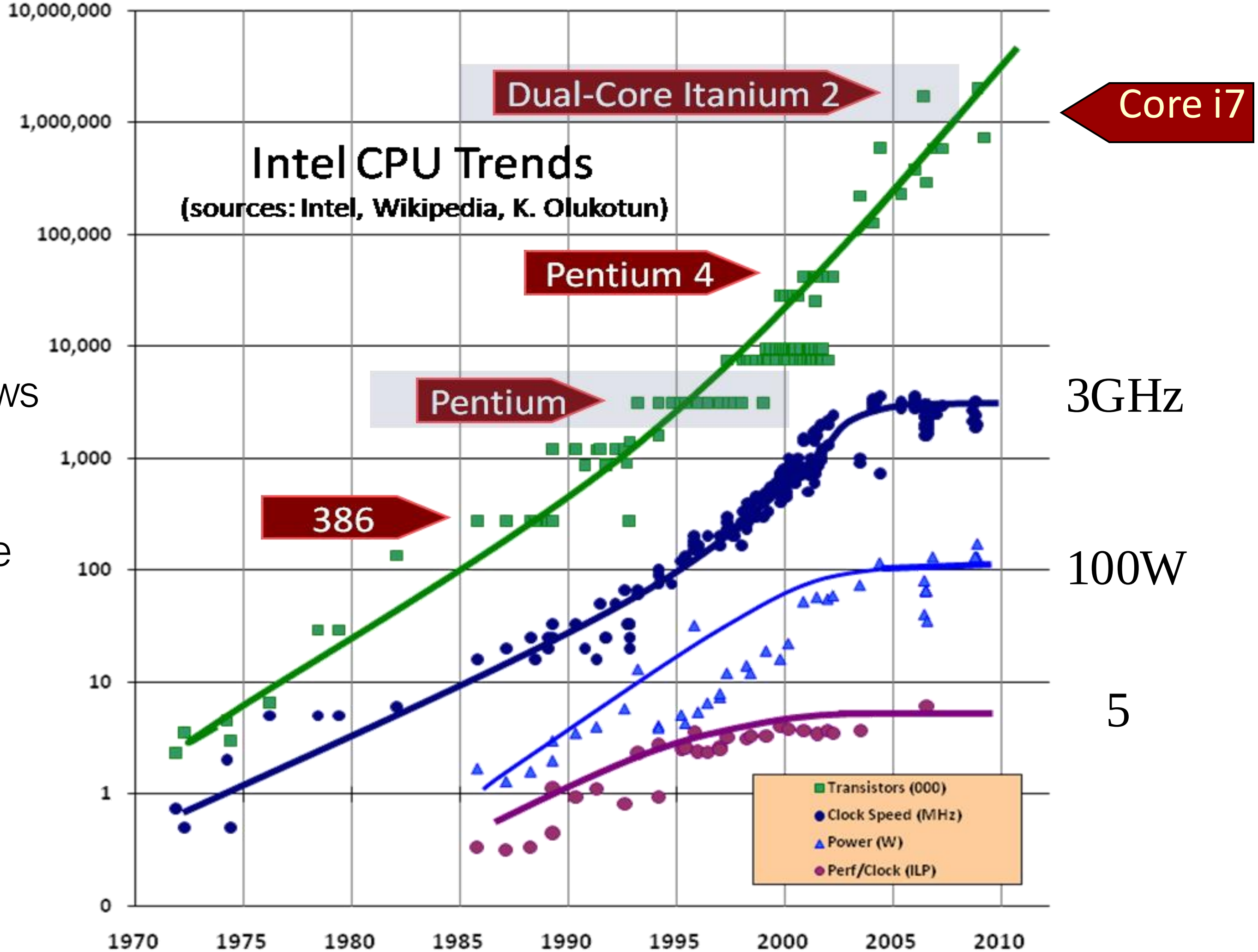
Limits to ILP

- Doubling issue rates above today's 3-6 instructions per clock, say to 6 to 12 instructions, probably requires a processor to
  - issue 3 or 4 data memory accesses per cycle,
  - resolve 2 or 3 branches per cycle,
  - rename and access more than 20 registers per cycle, and
  - fetch 12 to 24 instructions per cycle.
- The complexities of implementing these capabilities is likely to mean sacrifices in the maximum clock rate
  - E.g, widest issue processor is the Itanium 2, but it also has the slowest clock rate, despite the fact that it consumes the most power!

Limits to ILP

- Most techniques for increasing performance increase power consumption
- The key question is whether a technique is energy efficient
  - Does it increase power consumption faster than it increases performance?
- Multiple issue processors techniques all are energy inefficient:
  - Issuing multiple instructions incurs some overhead in logic that grows faster than the issue rate grows
  - Growing gap between peak issue rates and sustained performance
- Number of transistors switching = f(peak issue rate), and performance = f(sustained rate),
  growing gap between peak and sustained performance
  increasing energy per unit of performance

Next?



Trends:

- #transistors follows Moore
- but not freq. and performance/core

Conclusions

- 1985-2002: >1000X performance (55% /year) for single processor cores
- Hennessy: industry has been following a roadmap of ideas known in 1985 to exploit Instruction Level Parallelism and (real) Moore's Law to get 1.55X/year
  - Caches, (Super)Pipelining, Superscalar, Branch Prediction, Out-of-order execution, Trace cache
- **After 2002 slowdown** (about 20%/year increase)

Conclusions (cont'd)

- **ILP limits**: To make performance progress in future need to have explicit parallelism from programmer vs. implicit parallelism of ILP exploited by compiler/HW?

- Further problems:
  - Processor-memory performance gap
  - VLSI scaling problems (wiring)
  - Energy / leakage problems

- However: other forms of parallelism come to rescue:
  - going **Multi-Core**
  - **SIMD** revival – Sub-word parallelism

Parallel Architectures

- Definition: "A parallel computer is a collection of processing elements that cooperates and communicate to solve large problems fast"
    - Almasi and Gottlieb, Highly Parallel Computing, 1989

Parallel Architectures

- Definition: “A parallel computer is a collection of processing elements that cooperates and communicate to solve large problems fast”
    - Almasi and Gottlieb, Highly Parallel Computing, 1989
- The aim is to replicate processors to add performance vs design a faster processor.
- Parallel architecture extends traditional computer architecture with a communication architecture
    - abstractions (HW/SW interface)
    - different structures to realize abstraction efficiently

Flynn Taxonomy (1966)

- SISD - Single Instruction Single Data
  - Uniprocessor systems
- MISD - Multiple Instruction Single Data
  - No practical configuration and no commercial systems
- SIMD - Single Instruction Multiple Data
  - Simple programming model, low overhead, flexibility, custom integrated circuits
- MIMD - Multiple Instruction Multiple Data
  - Scalable, fault tolerant, *off-the-shelf* micros

Flynn

Flynn

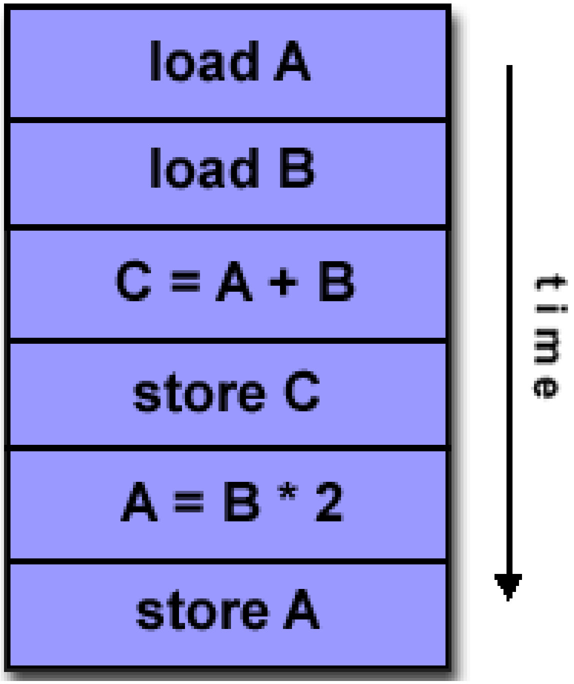
Flynn

Flynn Taxonomy (1966)

- SISD - Single Instruction Single Data
  - Uniprocessor systems
- MISD - Multiple Instruction Single Data
  - No practical configuration and no commercial systems
- SIMD - Single Instruction Multiple Data
  - Simple programming model, low overhead, flexibility, custom integrated circuits
- MIMD - Multiple Instruction Multiple Data
  - Scalable, fault tolerant, *off-the-shelf* micros

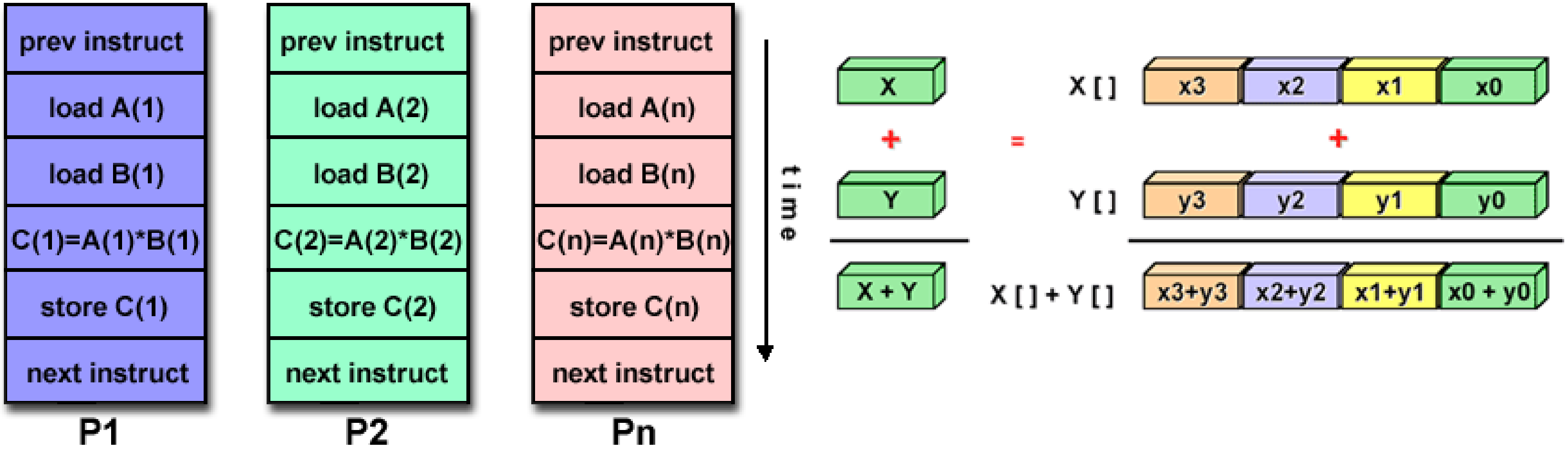
SISD

- A serial (non-parallel) computer
- Single instruction: only one instruction stream is being acted on by the CPU during any one clock cycle
- Single data: only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the oldest and even today, the most common type of computer



SIMD

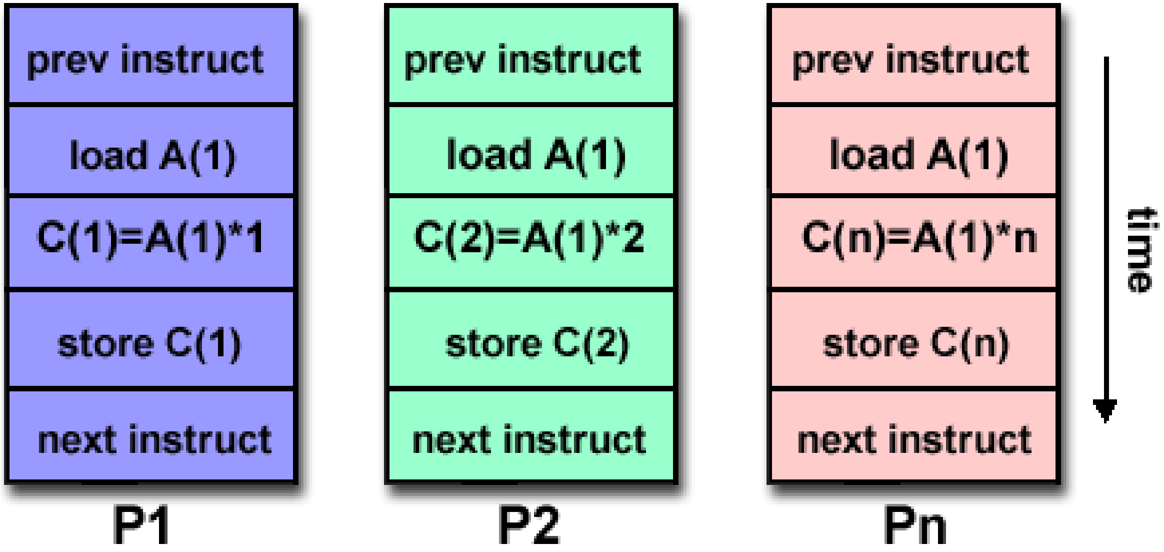
- A type of parallel computer
- Single instruction: all processing units execute the same instruction at any given clock cycle
- Multiple data: each processing unit can operate on a different data element



- Best suited for specialized problems characterized by a high degree of regularity, such as graphics/image processing

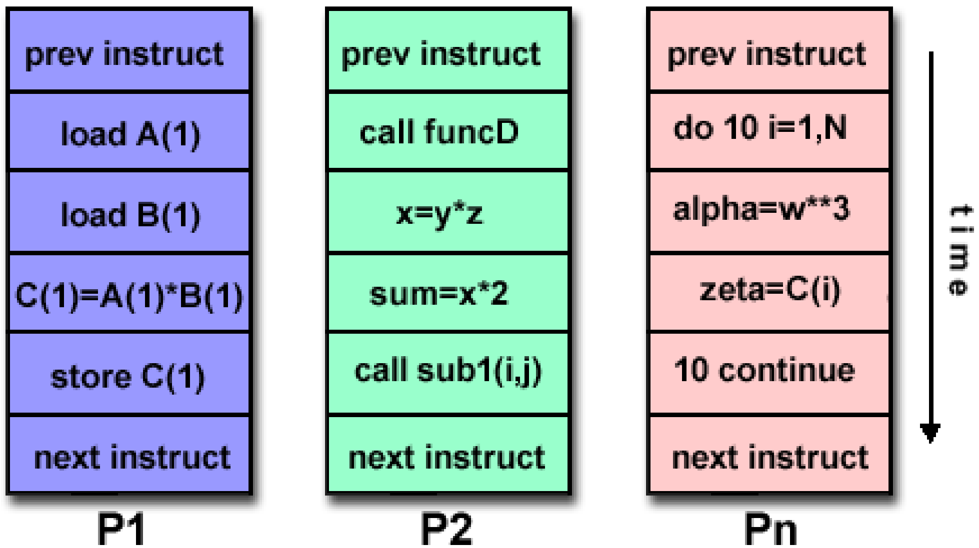
MISD

- A single data stream is fed into multiple processing units.
- Each processing unit operates on the data independently via independent instruction streams.



MIMD

- Nowadays, the most common type of parallel computer
- Multiple Instruction: every processor may be executing a different instruction stream
- Multiple Data: every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic



Advanced Computer Architectures

(High Performance Processors and Systems)

Instruction-Level Parallelism: Limits

Politecnico di Milano

v1