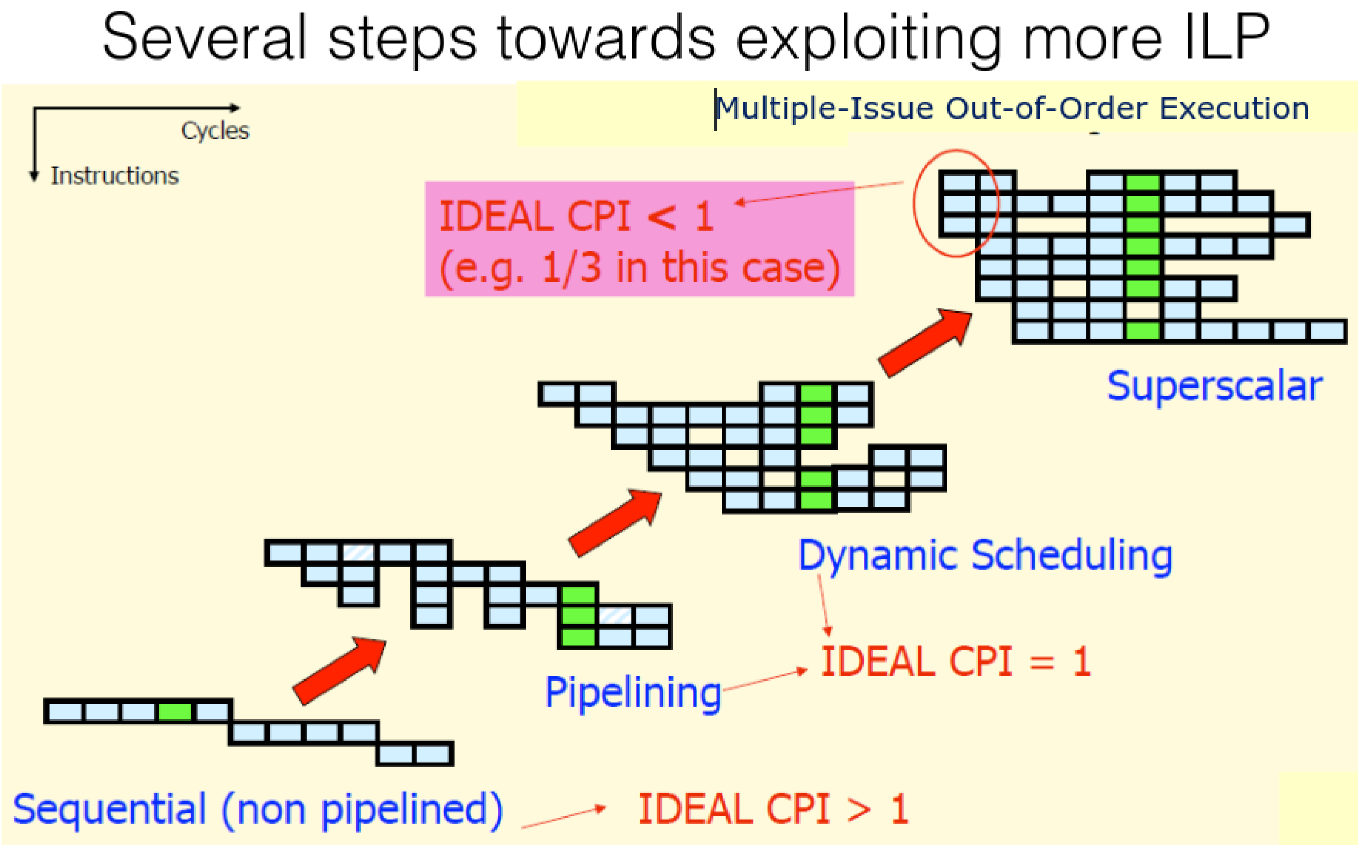
11.Instruction Level Parallelism Limits

ILP = Potential overlap of execution among unrelated instructions
Overlapping possible if:

- No Structural Hazards
- No RAW, WAR of WAW Stalls
- No Control Stalls
  The idea behind is to operate the execution of independent instructions(instruction referring to different operation, could be dependent on each other on the data they use). Want to try to execute them in some level of parallelism.
  To do the parallel pipeline we don't have to have structural hazards. Then on there data we need to define/manage the data stalls(Needs controls).



Superscalar Execution

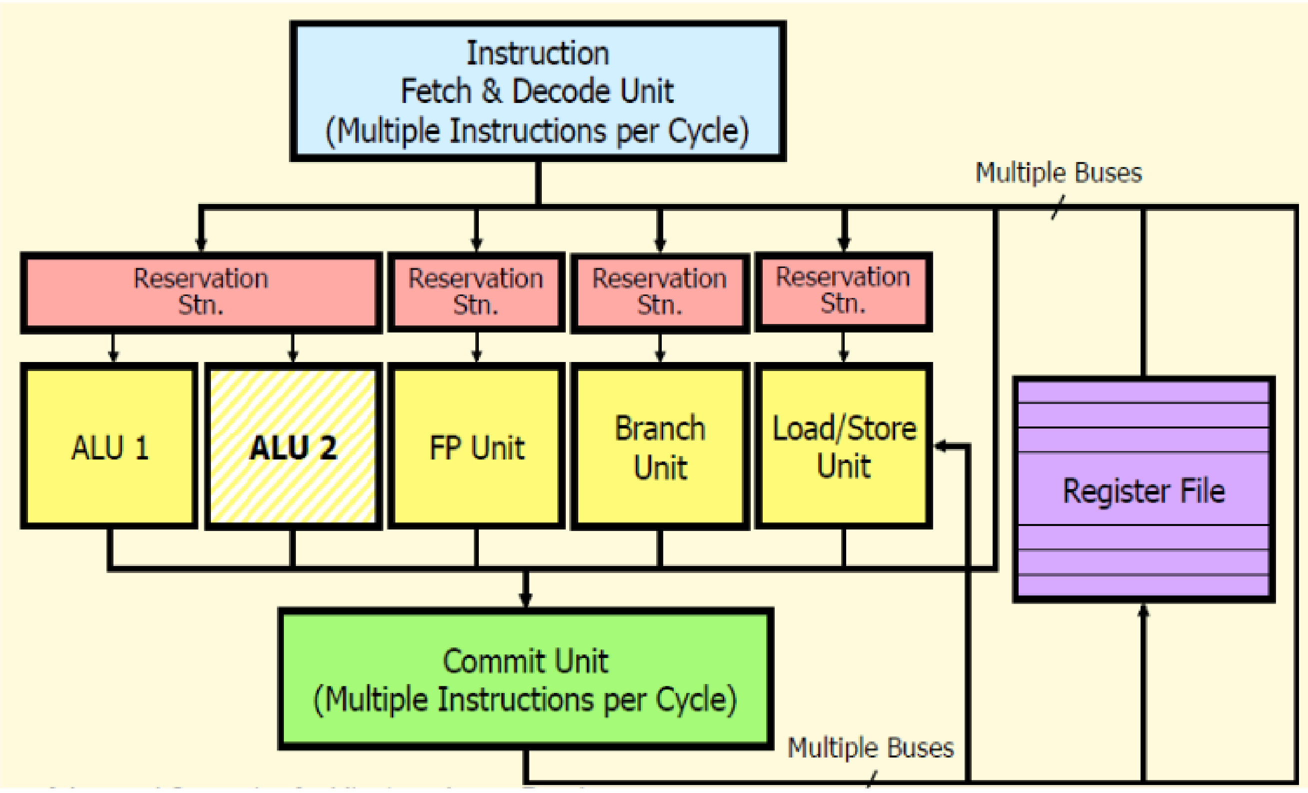
To have more than one instruction beginning execution at each clock cycle we need:

- 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

Once we decided on that we can decide on prediction data, reduce number of stalls and use dynamic scheduling.

We can use these architectures:

- 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
  - CPI_{ideal} = 1 / issue-width
  - We go to a instructions per clock cycle instead of clock cycle per instruction(IPC vs CPI)



You fetch more than one instruction from the memory. Then you have to parallelise them and try to start them at the same time.

You need a way to provide data in parallel(One for each unit and operation).

Also for the Commitment phase you need an architecture that can handle the end of multiple instruction at the same time.

Limits of ILP

Assumptions for ideal/perfect machine to start:

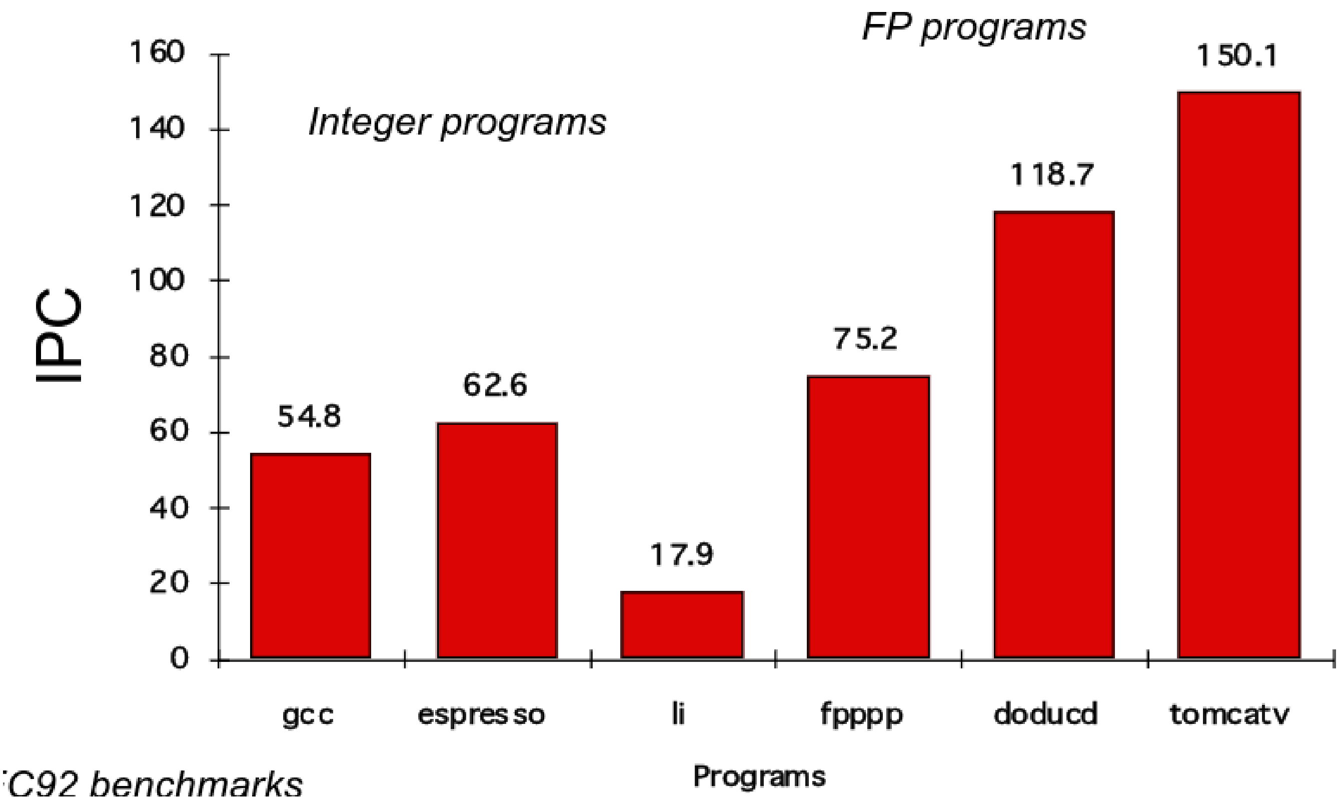
1. Register renaming
   - infinite virtual registers and all WAW & WAR hazards are avoided
2. Branch prediction

- 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(can become a problem for RaW, only if the real registers are the same)
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).
  We can evaluate ideal architectures with benchmarks. The best case we can obtain from ideal machines is still related to constrains and dependencies, we can obtain the maximum IPC. The designer goal is to go as close as possible to these limits.

Upper Limit to ILP: Ideal Machine



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 number of regs) =

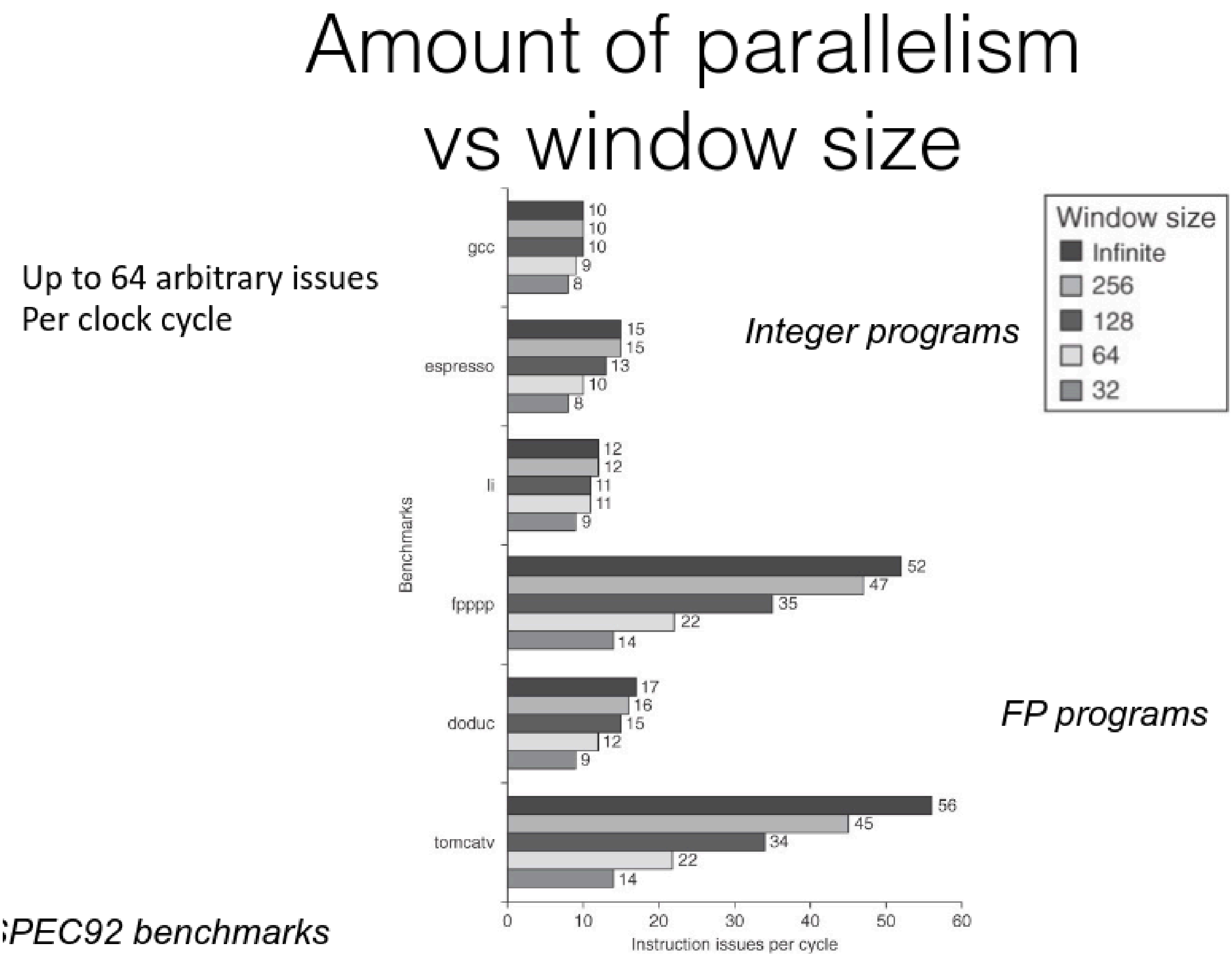
$$2n-2+2n-4+\ldots+2=\sum_{i=1}^{n-1} i = 2\frac{(n-1)n}{2} = n^2-n$$

- Window size = 2000 that leads to 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!)



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)

	Ideal Model	IBM Power 5 (2004-2006)Dual core @ 1.5 – 2.3 GHz
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
  You can have larger issue batch but what you can handle at the same time is more limited so you are wasting resources. You still need to be sure there aren't dependencies and hazards decreasing the frequency of the processor losing performance.

Intel P6 Family

Processor	First issue	Clock frequency	L1 cache	L2 cache
Pentium Pro	1995	100-200 MHz	8KB I + 8KB D	256-1025 KB
Pentium II	1998	233-450	16KB + 16KB	256-512
Pentium II Xeon	1999	400-450	16KB + 16KB	512-2 MB
Celeron	1999	500-900	16KB + 16KB	128
Pentium III	1999	450-1100	16KB + 16KB	256-512
Pentium III Xeon	2000	700-900	16KB + 16KB	1-2 MB

RISC vs CISC Architectures

RISC

- Reduced Instruction Set Computer
- Examples: ARM, SPARC, MIPS, PowerPC, RISC-V
  CISC
- Complex Instruction Set Computer
- Examples: x86, AMD

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(Preferred architecture as VLIW are preferred to multimedia architecture and for consumer architecture)
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)

- Useful for regular architecture, very efficient

Taxonomy of Multiple Issue Machines

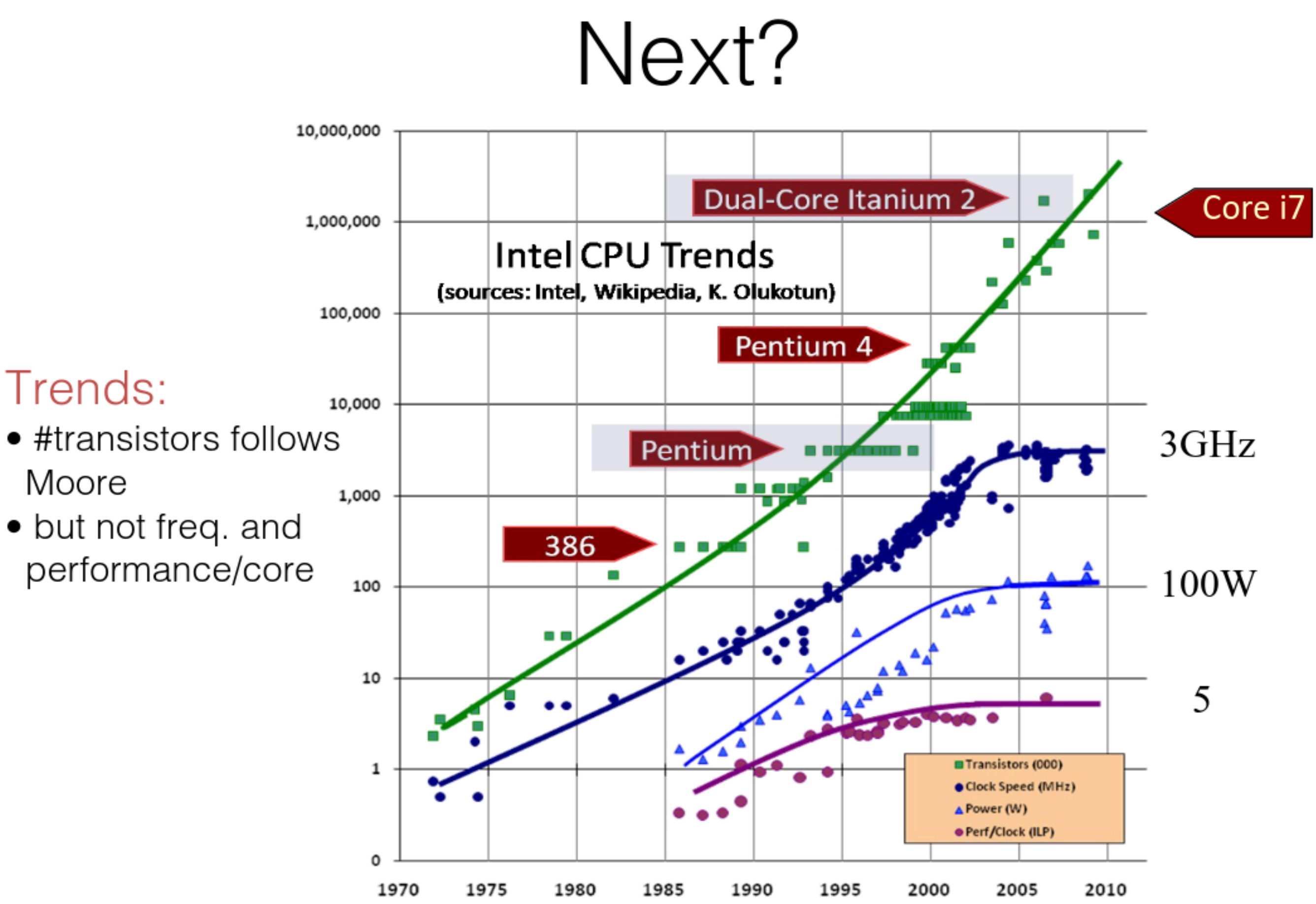
Common name	Issue structure	Hazard detection	Scheduling	Distinguishing characteristic	Examples
Superscalar (static)	Dynamic	Hardware	Static	In-order execution	Mostly in the embedded space: MIPS and ARM, including the ARM Coretex A8
Superscalar (dynamic)	Dynamic	Hardware	Dynamic	Some out-of-order execution, 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!
  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



Conclusion

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)
  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