

***Architectural Issues for the 1990s***

**David A. Patterson**

**Computer Science Division  
EECS Department  
University of California  
Berkeley, CA 94720**

**© 1990**

***(presented at Microprocessor Forum, October 10, 1990)***

## ***Basic Performance Equation***

$$\text{CPU Time} = \text{dynamic instruction count (IC)} \times \text{cycles per instruction (CPI)} \times \text{clock cycle time (CCT)}$$

**These factors are interrelated:**

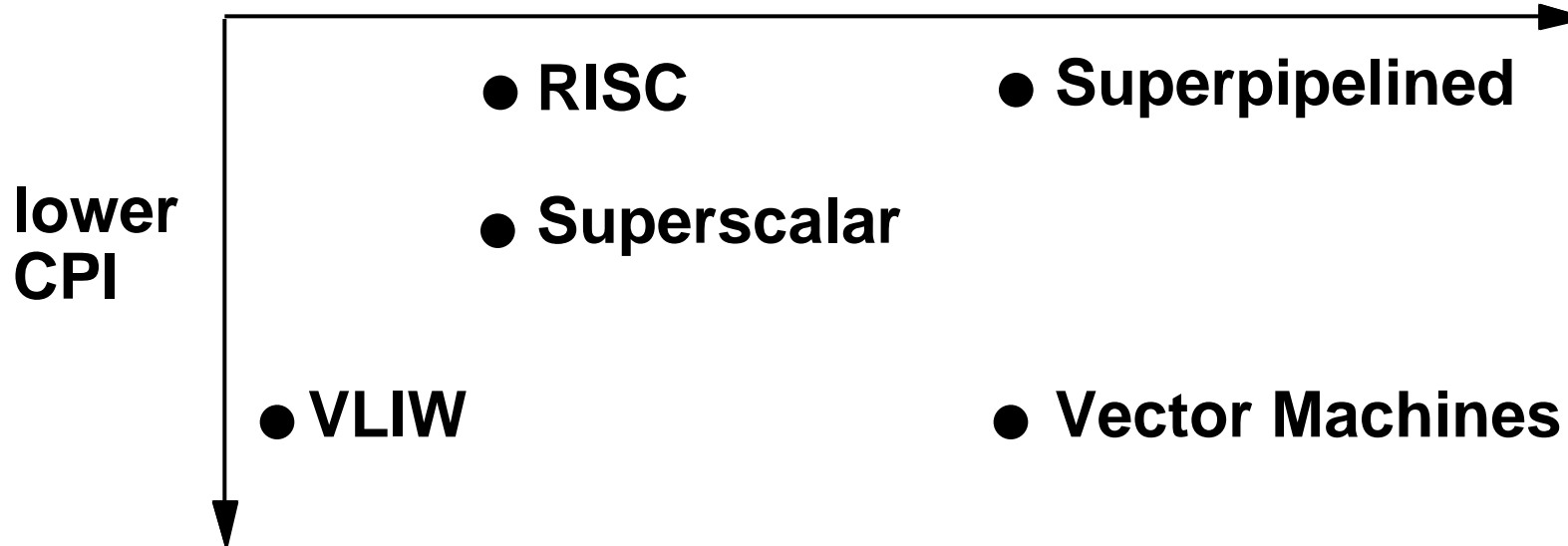
**IC =  $f$ (compilers, instruction set architecture)**

**CPI =  $f$ (instruction set architecture, organization)**

**CCT =  $f$ (organization, hardware technology)**

## Candidate Architectures

faster clock rates

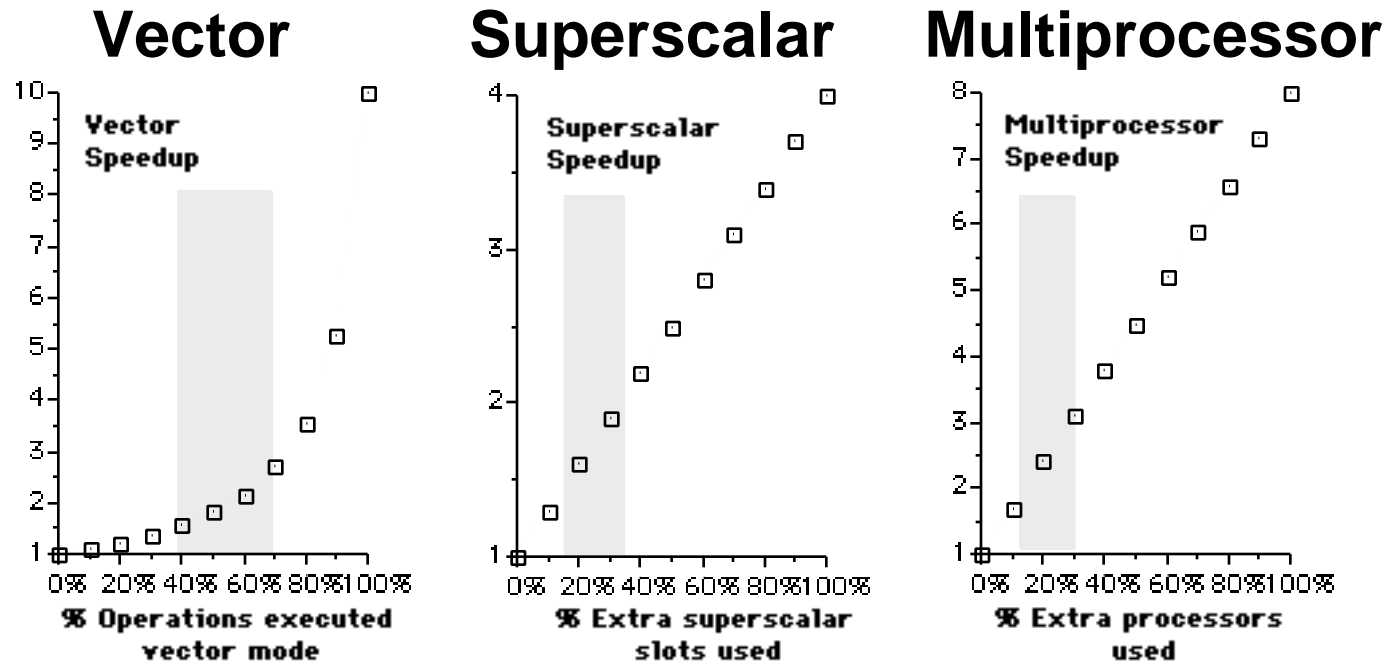


**VLIW:** issue large # of instr/cycle  
large issue count sacrifices clock rate?  
how much instruction level parallelism?

**Vector:** superpipelined plus multiple issue through  
vector instructions (>1 operation/clock)

# Software Trends for 1990s

- Compiler technology increasing importance



## *Amdahl's Law*

$$\text{Speedup} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

## *Corollary to Amdahl's Law*

$$\text{Performance limit} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}})}$$

Performance limit of 1990's is not the CPU?

## ***Memory as Limit to Performance***

**Current microprocessors are "*Either/Or*" CPUs:**

***Either* CPU is busy and memory is idle (running)  
*Or* CPU is idle and memory is busy (cache miss)  
(Assumes CPU is idle < 10% - 15%)**

**DRAMs improve more slowly than CPUs**

**=> 100s of instructions executed in 1 DRAM access  
30 clocks @ 5 ns ÷ 0.3 CPI = 100 Instr/DRAM time**

**=> Half of time waiting for memory**

**1,000,000 instructions, 1% miss rate, 30 clock miss  
= 1,000,000 ÷ 0.33 + 1,000,000 x 1% x 30  
= 300,000 + 300,000 = 600,000 => CPI = 0.6**

**Memory system limits performance of 1990s!**

## ***Solutions that will be explored in 1990s***

- **New algorithms and compiler optimizations to reduce cache misses or reduce their impact**
- **Lockup-Free caches: Don't stall on miss**  
**=> CPUs can continue execution while waiting for data (e.g., 360/91)**
- **Prefetching: Compiler inserts hints to cache**  
**=> Multiported caches to avoid cache miss**  
**=> Prefetch buffers for fast cache miss**
- **Novel DRAM designs and interfaces**  
**=> Make internal DRAM bandwidth available to CPU**  
**=> Replace interface of RAS, CAS SIMM packaging**
- **Process swap on cache miss**  
**=> hardware scheduling of ready processes**

## ***I/O as Limit to Performance***

**Disk performance improves more slowly than CPUs  
=> Disk arrays will be the standard I/O subsystems  
=> many I/O devices and I/O interrupts**

### **I/O needs**

**(available) memory bandwidth  
intelligent DMA (gather, scatter, virtual memory)  
fast standard busses**

**Memory system designs must be evaluated by  
I/O benchmarks as well as CPU benchmarks**

**CPU designs can prevent good I/O performance**

- cache and TLB designs that force flushing on each I/O**
- I/O interfaces that force extra copies of data**

## ***Architectural Issues for the 1990s: Summary and Conclusions***

- **Given:**  
Superscalar, superpipelined RISCs and  
Amdahl's Law will not be repealed  
=> High performance in 1990s is not limited by CPU
- **Predictions for 1990s:**  
"Either/Or" CPU/Memory will disappear  
  
Multipronged attack on memory bottleneck  
cache conscious compilers  
lockup free caches / prefetching  
  
All programs will become I/O bound; design  
accordingly  
  
Most important CPU of 1990s is in DRAM: "IRAM"  
(Intelligent RAM: 64Mb + 0.3M transistor CPU = 100.5%)  
=> CPUs are genuinely free with IRAM

## ***References***

**J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 1990.**

**D. Kroft, “Lockup-free instruction fetch/prefetch cache organization,” *Proc. Eighth Annual Symposium on Computer Architecture* (May 12-14, 1981), Minneapolis, Minnesota, pp. 81-87.**

**D. A. Patterson, G. A. Gibson, and R. H. Katz, “A case for redundant arrays of inexpensive disks (RAID),” Technical Report UCB/CSD 87/391, Univ. of Calif. 1987. (Also appeared in ACM SIGMOD Conference Proc., Chicago, Illinois, June 1-3, 1988, pp. 109-116.)**