Performance

(3rd Ed: p.240-271, 4th Ed: p.26-55, 5th Ed: p.28-53)

- purchasing perspective
  - performance, cost
- design perspective
  - performance / cost and improvements
- require
  - method for calculation
  - basis for comparison
  - metric for evaluation
  - understanding of implications for architectural choices

Calculating performance

- CPI: average clock cycles per instruction
- number of cycles for program P = number of instr. for P x CPI
- execution time for P = clock cycle time x number of cycles for P
  = $\frac{1}{\text{clock speed}}$ x number of cycles for P
- average exe. time for P1, P2...Pn = $\frac{1}{n}$ (exe. time for P1 +…+ exe. time for Pn)
  (assume equal workload)

Example

- M1 and M2 implement the same instruction set with 2 classes of instructions: A and B
- CPI for M1 on class A instructions: A1
  B : B1
- CPI for M2 on class A instructions: A2
  B : B2
- clock speed for M1: C1 MHz
  M2: C2 MHz
- compare their peak and average performance of N instructions, half class A and half class B
- hints: find ratio of execution times
  check dimension of expression

Example (cont.)

- peak performance for N instructions
  - exe. time for M1 $= \frac{1}{C1} \times N \times$ minimum CPI for M1 $= \frac{N(A1 \downarrow B1)}{C1}$
  - compare M1 and M2 : $\frac{\text{exe. time for M1}}{\text{exe. time for M2}} = \frac{(A1 \downarrow B1)\, C2}{(A2 \downarrow B2)\, C1}$
- average performance given by execution time of N instructions, half class A and half class B
  - exe. time for M1 $= \frac{1}{C1} \times N \times$ average CPI for M1 $= \frac{N(A1 + B1)}{2C1}$
  - compare M1 and M2 : $\frac{\text{exe. time for M1}}{\text{exe. time for M2}} = \frac{(A1 + B1)\, C2}{(A2 + B2)\, C1}$
- CPI for N_A class A, N_B class B instructions? compare speed: number of instructions per sec?

Performance example

- minimise: exe. time = instr. count x CPI x cycle time

-

	SUN 68000	SUN RISC
Instruction count ratio	1.0	1.25
Cycle time	40 ns	60 ns
CPI	5.0 – 7.0	1.3 – 1.7
Execution time ratio	2	1
Price ratio	1	1.1 – 1.2

Aspects of processor performance

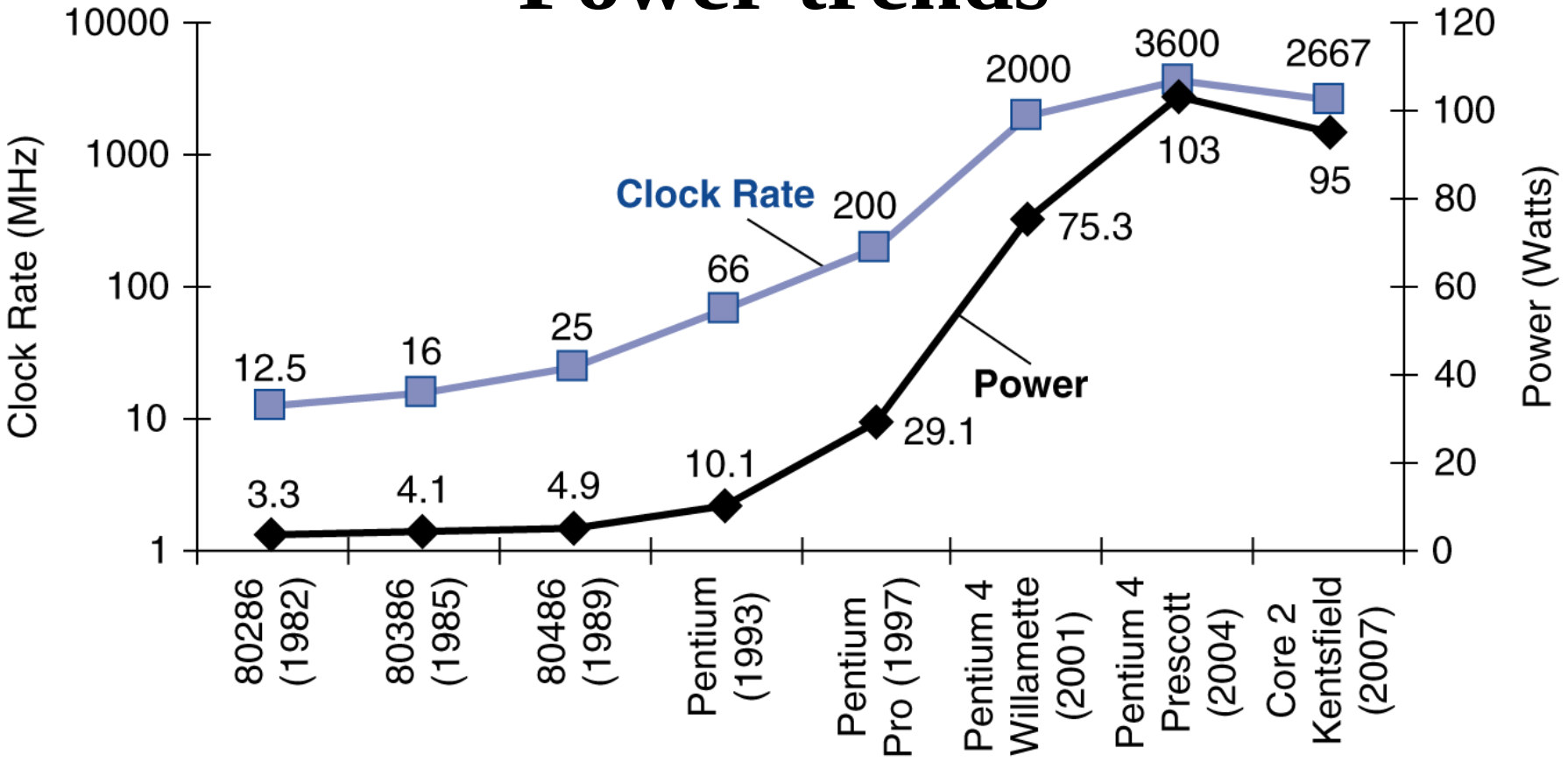
$$\text{processor time} = \frac{\text{sec}}{\text{prog}} = \frac{\text{intsr}}{\text{prog}} \times \frac{\text{cycles}}{\text{instr}} \times \frac{\text{sec}}{\text{cycle}}$$

	instr count	CPI	clock rate
algorithm			
language			
compiler			
instr set			
organisation			
technology			

RISC Instruction set design principles

- make the common case fast
  - reduce CPI
- simple and regular format, small number of general-purpose registers
  - reduce cycle time
  - simplify implementation: more adaptable to new technology
  - include useful facilities on chip, e.g. memory management
  - easier to use, fewer alternatives: manually and by compilers
  - greater confidence in hardware correctness
  - smaller chip size: higher yield, reduce production cost
- require compromise: increased code size
  need better compilers

Power trends



- in CMOS integrated circuit technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30 | reduced if simpler | 5V → 1V | ×1000

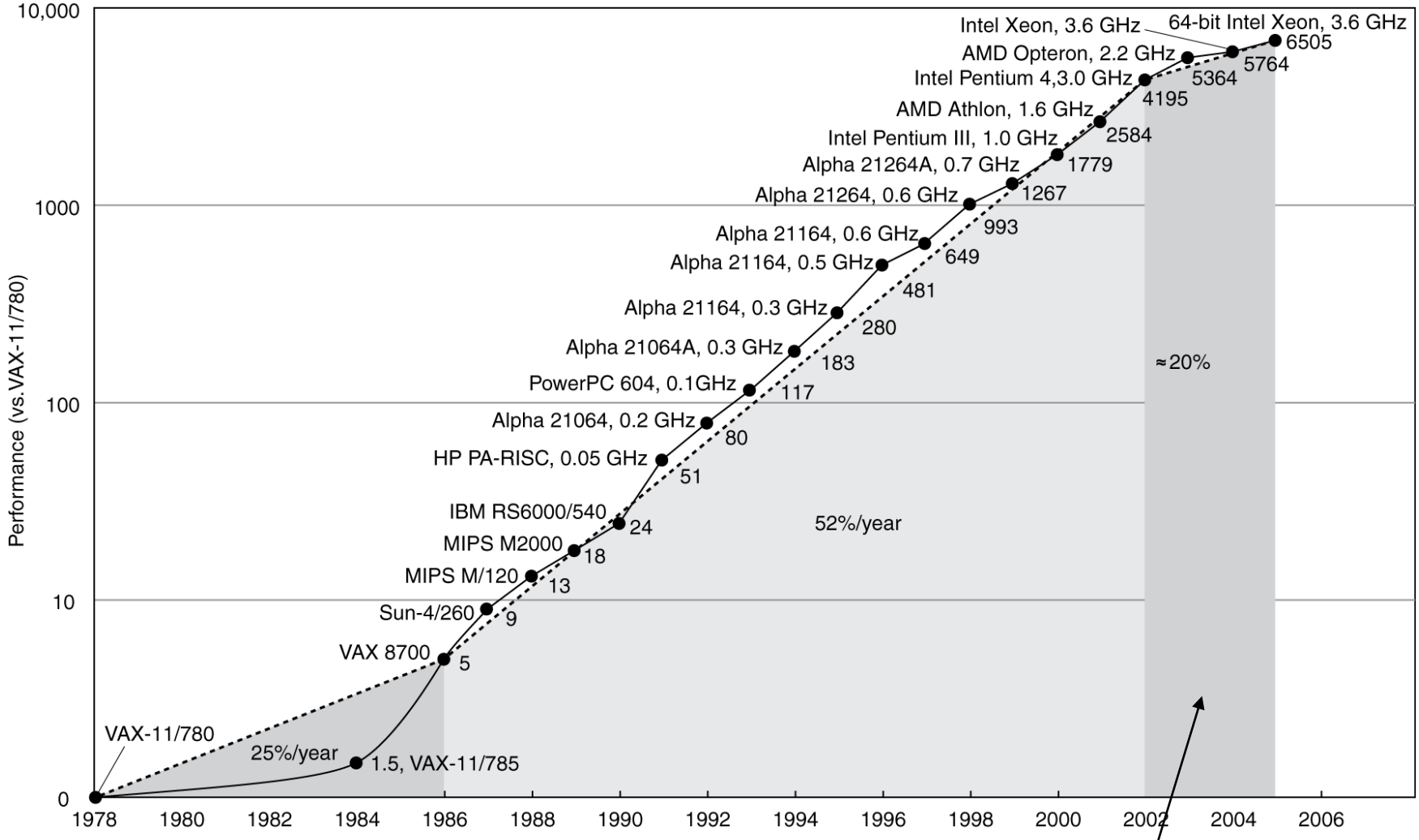
Reducing power consumption

- suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- the power wall
  - cannot reduce voltage further
  - cannot remove more heat
- how else can we improve performance?

Uniprocessor performance



constrained by power, instruction-level parallelism, memory latency

Improving performance

- fast, local store
  - e.g. on-chip caches, IRAM (processor in DRAM)
- concurrent execution of instructions
  - multiple function units : super scalar
  - “production line” arrangement : pipeline
  - multiple instruction streams : multi-threading
- direct hardware implementation, domain-specific optim.
  - reconfigurable hardware: millions of programmable gates
  - no fetch / decode, customise at compile time and at run time
  - see http://www.doc.ic.ac.uk/~wl/papers/federico_faggin.htm
- other technologies
  - multiprocessors: e.g. Graphics Processing Unit (GPU)
  - asynchronous designs: no global clock

Performance evaluation

- guide design decisions
- compare different architectures
- compare implementations of a given architecture
- compare compilers for a given implementation
- depend on particular program and data
- reporting performance measurements: reproducibility
- speed sometimes in: million instructions per second
  - larger MIPS = faster program?
- benchmark examples
  - SPEC: System Performance Evaluation Co-operative
  - EEMBC: Embedded Microprocessor Benchmark Consortium

Basis of evaluation

method	pros	cons
actual target workload	• representative	• specific, non portable • difficult to run/measure • hard to identify problem
full application benchmarks	• portable • wide spread • improve. useful	• less representative
small kernel benchmarks	• easy to use • early in design cycle • identify peak capability	• peak far away from typical performance

Example: SPEC CPU2000

- 26 component-level benchmarks with inputs
  for processor, memory, compiler; not I/O, network, graphics
- 12 integer (11 in C, 1 in C++)
  gcc, compress, FPGA place and route, chess, ray tracing
- 14 floating-point intensive
  qcd, multi-grid solver, finite-element, face recognition
- reference machine: normalise performance metrics
  Sun Ultra5_10 with 300 MHz processor (see *opensparc.net*)
- more recent: SPEC CPU2006 (12 integer, 17 float. point)
  criteria: compute-bound, portable, latest applications
  see *http://www.spec.org/osg/*
- other SPEC benchmarks: Java, GPU, HPC, power workload
  non-SPEC benchmarks: EEMBC, GroundHog

SPEC CINT2006 for Opteron X4 2356

Name	Description	IC×10^9	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							11.7

High cache miss rates

Geometric mean: $\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$

PIII vs P4: what different, and why?