Performance

COE 301 / ICS 233

Computer Organization

Dr. Muhamed Mudawar

College of Computer Sciences and Engineering

King Fahd University of Petroleum and Minerals

What is Performance?

- How can we make intelligent choices about computers?
- Why is some computer hardware performs better at some programs, but performs less at other programs?
- How do we measure the performance of a computer?
- What factors are hardware related? software related?
- How does machine's instruction set affect performance?
- Understanding performance is key to understanding underlying organizational motivation

Response Time and Throughput

- **Response Time**
  - Time between start and completion of a task, as observed by end user
  - Response Time = CPU Time + Waiting Time (I/O, OS scheduling, etc.)
- **Throughput**
  - Number of tasks the machine can run in a given period of time
- Decreasing execution time improves throughput
  - Example: using a faster version of a processor
  - Less time to run a task ⇒ more tasks can be executed
- Increasing throughput can also improve response time
  - Example: increasing number of processors in a multiprocessor
  - More tasks can be executed in parallel
  - Execution time of individual sequential tasks is not changed
  - But less waiting time in scheduling queue reduces response time

Higher Performance = Less Execution Time

- For some program running on machine *X*

$$\text{Performance}_X = \frac{1}{\text{Execution time}_X}$$

- *X* is *n* times faster than *Y*

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

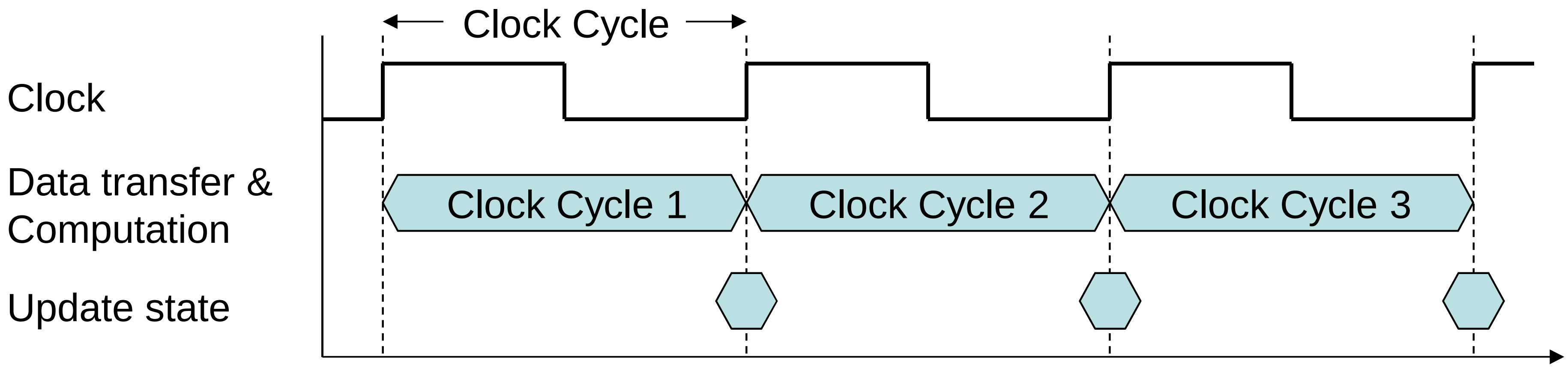
What do we mean by Execution Time?

- Real Elapsed Time
  - Counts everything:
    - Waiting time, Input/output, disk access, OS scheduling, … etc.
  - Useful number, but often not good for comparison purposes
- Our Focus: CPU Execution Time
  - Time spent while executing the program instructions
  - Doesn't count the waiting time for I/O or OS scheduling
  - Can be measured in seconds, or
  - Can be related to number of CPU clock cycles

$$\text{CPU Execution Time} = \text{CPU cycles} \times \text{Cycle time} = \frac{\text{CPU cycles}}{\text{Clock rate}}$$

What is the Clock Cycle?

- Operation of digital hardware is governed by a clock



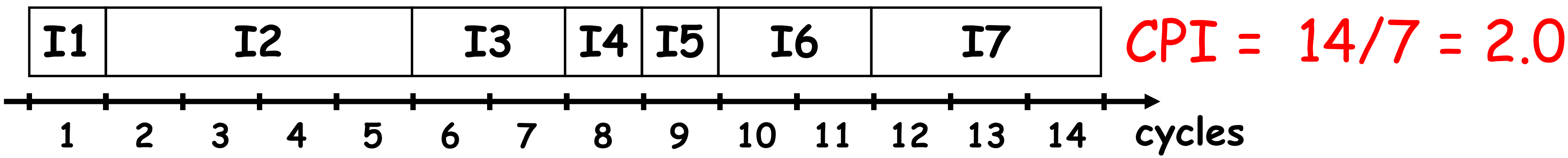
- Clock Cycle = Clock period
  - Duration between two consecutive rising edges of the clock signal
- Clock rate = Clock frequency = 1 / Clock Cycle
  - 1 Hz = 1 cycle/sec　　1 KHz = 10^3 cycles/sec
  - 1 MHz = 10^6 cycles/sec　　1 GHz = 10^9 cycles/sec
  - 2 GHz clock has a cycle time = $1/(2\times10^9)$ = 0.5 nanosecond (ns)

Improving Performance

- To improve performance, we need to
  - Reduce the number of clock cycles required by a program, or
  - Reduce the clock cycle time (increase the clock rate)
- Example:
  - A program runs in 10 seconds on computer *X* with 2 GHz clock
  - What is the number of CPU cycles on computer *X* ?
  - We want to design computer *Y* to run same program in 6 seconds
  - But computer *Y* requires 10% more cycles to execute program
  - What is the clock rate for computer *Y* ?
- Solution:
  - CPU cycles on computer *X* = 10 sec × 2 × 10^9 cycles/s = 20 × 10^9 cycles
  - CPU cycles on computer *Y* = 1.1 × 20 × 10^9 = 22 × 10^9 cycles
  - Clock rate for computer *Y* = 22 × 10^9 cycles / 6 sec = 3.67 GHz

Clock Cycles per Instruction (CPI)

- Instructions take different number of cycles to execute
  - Multiplication takes more time than addition
  - Floating point operations take longer than integer ones
  - Accessing memory takes more time than accessing registers
- CPI is an average number of clock cycles per instruction



- Important point

  Changing the cycle time often changes the number of cycles required for various instructions

Performance Equation

- To execute, a given program will require …
  - Some number of machine instructions
  - Some number of clock cycles
  - Some number of seconds
- We can relate CPU clock cycles to instruction count

$$\text{CPU cycles} = \text{Instruction Count} \times \text{CPI}$$

- Performance Equation: (related to instruction count)

$$\text{CPU Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Cycle time}$$

Understanding Performance Equation

$$\text{Execution Time} = \text{Instruction Count} \times \text{CPI} \times \text{Cycle time}$$

	I-Count	CPI	Cycle
Program	X		
Compiler	X	X	
ISA	X	X	
Organization		X	X
Technology			X

Using the Performance Equation

- Suppose we have two implementations of the same ISA
- For a given program
  - Machine A has a clock cycle time of 250 ps and a CPI of 2.0
  - Machine B has a clock cycle time of 500 ps and a CPI of 1.2
  - Which machine is faster for this program, and by how much?
- Solution:
  - Both computer execute same count of instructions = I
  - CPU execution time (A) = $I \times 2.0 \times 250$ ps = $500 \times I$ ps
  - CPU execution time (B) = $I \times 1.2 \times 500$ ps = $600 \times I$ ps
  - Computer A is faster than B by a factor = $\frac{600 \times I}{500 \times I} = 1.2$

Determining the CPI

- Different types of instructions have different CPI

  Let CPI_i = clocks per instruction for class *i* of instructions

  Let C_i = instruction count for class *i* of instructions

$$\text{CPU cycles} = \sum_{i=1}^{n} (CPI_i \times C_i)$$

$$CPI = \frac{\sum_{i=1}^{n} (CPI_i \times C_i)}{\sum_{i=1}^{n} C_i}$$

- Designers often obtain CPI by a detailed simulation
- Hardware counters are also used for operational CPUs

Example on Determining the CPI

❖ Problem

A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: class A, class B, and class C, and they require one, two, and three cycles per instruction, respectively.

The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C

The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C

Compute the CPU cycles for each sequence. Which sequence is faster?

What is the CPI for each sequence?

❖ Solution

CPU cycles (1^{st} sequence) = (2×1) + (1×2) + (2×3) = 2+2+6 = 10 cycles

CPU cycles (2^{nd} sequence) = (4×1) + (1×2) + (1×3) = 4+2+3 = 9 cycles

Second sequence is faster, even though it executes one extra instruction

CPI (1^{st} sequence) = 10/5 = 2 CPI (2^{nd} sequence) = 9/6 = 1.5

Second Example on CPI

Given: instruction mix of a program on a RISC processor

What is average CPI?

What is the percent of time used by each instruction class?

$Class_i$	$Freq_i$	CPI_i	$CPI_i \times Freq_i$	%Time
ALU	50%	1	0.5×1 = 0.5	0.5/2.2 = 23%
Load	20%	5	0.2×5 = 1.0	1.0/2.2 = 45%
Store	10%	3	0.1×3 = 0.3	0.3/2.2 = 14%
Branch	20%	2	0.2×2 = 0.4	0.4/2.2 = 18%

Average CPI = 0.5+1.0+0.3+0.4 = 2.2

How faster would the machine be if load time is 2 cycles?

What if two ALU instructions could be executed at once?

MIPS as a Performance Measure

- MIPS: Millions Instructions Per Second
- Sometimes used as performance metric
  - Faster machine ⇒ larger MIPS
- MIPS specifies instruction execution rate

$$\text{MIPS} = \frac{\text{Instruction Count}}{\text{Execution Time} \times 10^6} = \frac{\text{Clock Rate}}{\text{CPI} \times 10^6}$$

- We can also relate execution time to MIPS

$$\text{Execution Time} = \frac{\text{Inst Count}}{\text{MIPS} \times 10^6} = \frac{\text{Inst Count} \times \text{CPI}}{\text{Clock Rate}}$$

Drawbacks of MIPS

Three problems using MIPS as a performance metric

1. Does not take into account the capability of instructions
   - Cannot use MIPS to compare computers with different instruction sets because the instruction count will differ
2. MIPS varies between programs on the same computer
   - A computer cannot have a single MIPS rating for all programs
3. MIPS can vary inversely with performance
   - A higher MIPS rating does not always mean better performance
   - Example in next slide shows this anomalous behavior

MIPS example

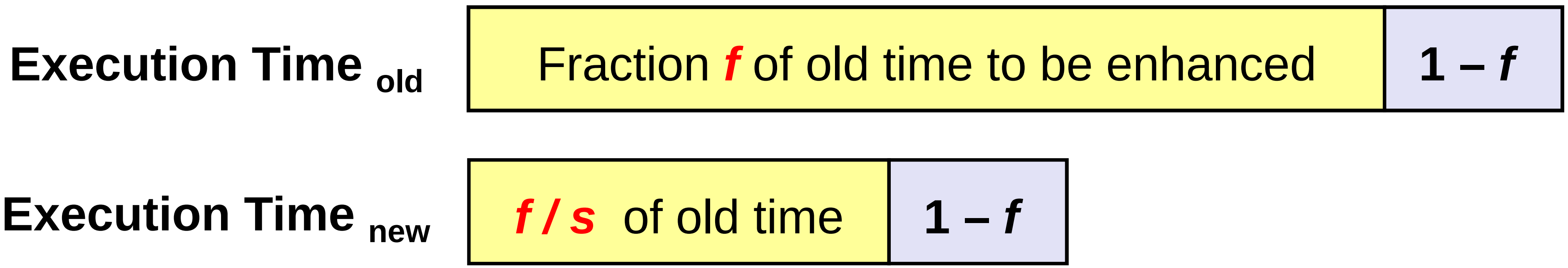
- Two different compilers are being tested on the same program for a 4 GHz machine with three different classes of instructions: Class A, Class B, and Class C, which require 1, 2, and 3 cycles, respectively.
- The instruction count produced by the first compiler is 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- The second compiler produces 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- Which compiler produces a higher MIPS?
- Which compiler produces a better execution time?

Solution to MIPS Example

- First, we find the CPU cycles for both compilers
  - CPU cycles (compiler 1) = $(5\times1 + 1\times2 + 1\times3)\times10^9 = 10\times10^9$
  - CPU cycles (compiler 2) = $(10\times1 + 1\times2 + 1\times3)\times10^9 = 15\times10^9$
- Next, we find the execution time for both compilers
  - Execution time (compiler 1) = 10×10^9 cycles / 4×10^9 Hz = 2.5 sec
  - Execution time (compiler 2) = 15×10^9 cycles / 4×10^9 Hz = 3.75 sec
- Compiler1 generates faster program (less execution time)
- Now, we compute MIPS rate for both compilers
  - MIPS = Instruction Count / (Execution Time × 10^6)
  - MIPS (compiler 1) = $(5+1+1) \times 10^9 / (2.5 \times 10^6) = 2800$
  - MIPS (compiler 2) = $(10+1+1) \times 10^9 / (3.75 \times 10^6) = 3200$
- So, code from compiler 2 has a higher MIPS rating !!!

Amdahl's Law

- Amdahl's Law is a measure of Speedup
  - How a program performs after improving portion of a computer
  - Relative to how it performed previously
- Let ***f*** = Fraction of the computation time that is enhanced
- Let ***s*** = Speedup factor of the enhancement only



$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution Time}_{\text{old}}}{\text{Execution Time}_{\text{new}}} = \frac{1}{((1 - f) + f / s)}$$

Example on Amdahl's Law

- Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?

- Solution: suppose we improve multiplication by a factor s

  25 sec (4 times faster) = 80 sec / s + 20 sec

  s = 80 / (25 – 20) = 80 / 5 = 16

  Improve the speed of multiplication by s = 16 times

- How about making the program 5 times faster?

  20 sec (5 times faster) = 80 sec / s + 20 sec

  s = 80 / (20 – 20) = ∞ Impossible to make 5 times faster!

Example 2 on Amdahl's Law

- Suppose that floating-point square root is responsible for 20% of the execution time of a graphics benchmark and ALL FP instructions are responsible for 60%
- One proposal is to speedup FP SQRT by a factor of 10
- Alternative choice: make ALL FP instructions 2X faster, which choice is better?
- Answer:
  - Choice 1: Improve FP SQRT by a factor of 10
  - Speedup (FP SQRT) = 1/(0.8 + 0.2/10) = 1.22
  - Choice 2: Improve ALL FP instructions by a factor of 2
  - Speedup = 1/(0.4 + 0.6/2) = 1.43 ➔ **Better**

Benchmarks

- Performance is measured by running real applications
  - Use programs typical of expected workload
  - Representatives of expected classes of applications
  - Examples: compilers, editors, scientific applications, graphics, ...
- SPEC (System Performance Evaluation Corporation)
  - Website: www.spec.org
  - Various benchmarks for CPU performance, graphics, high-performance computing, Web servers, etc.
  - Specifies rules for running list of programs and reporting results
  - Valuable indicator of performance and compiler technology
  - SPEC CPU 2006 (12 integer + 17 FP programs)

SPEC CPU Benchmarks

		Benchmark name by SPEC generation			
SPEC2006 benchmark description	SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
GNU C compiler					gcc
Interpreted string processing			perl		espresso
Combinatorial optimization		mcf			li
Block-sorting compression		bzip2		compress	eqntott
Go game (AI)	go	vortex	go	sc	
Video compression	h264avc	gzip	ijpeg		
Games/path finding	astar	eon	m88ksim		
Search gene sequence	hmmer	twolf			
Quantum computer simulation	libquantum	vortex			
Discrete event simulation library	omnetpp	vpr			
Chess game (AI)	sjeng	crafty			
XML parsing	xalancbmk	parser			
CFD/blast waves	bwaves				fpppp
Numerical relativity	cactusADM				tomcatv
Finite element code	calculix				doduc
Differential equation solver framework	dealII				nasa7
Quantum chemistry	gamess				spice
EM solver (freq/time domain)	GemsFDTD			swim	matrix300
Scalable molecular dynamics (~NAMD)	gromacs		apsi	hydro2d	
Lattice Boltzman method (fluid/air flow)	lbm		mgrid	su2cor	
Large eddie simulation/turbulent CFD	LESlie3d	wupwise	applu	wave5	
Lattice quantum chromodynamics	milc	apply	turb3d		
Molecular dynamics	namd	galgel			
Image ray tracing	povray	mesa			
Spare linear algebra	soplex	art			
Speech recognition	sphinx3	equake			
Quantum chemistry/object oriented	tonto	facerec			
Weather research and forecasting	wrf	ammp			
Magneto hydrodynamics (astrophysics)	zeusmp	lucas			
		fma3d			
		sixtrack			

- Wall clock time is used as a performance metric
- Benchmarks measure CPU time, because of little I/O

Summarizing Performance Results

$$SPEC\ Ratio = \frac{Time\ on\ Reference\ Computer}{Time\ on\ Computer\ Being\ Rated}$$

$$\frac{SPEC\ Ratio_A}{SPEC\ Ratio_B} = \frac{\frac{Execution\ Time_{Ref}}{Execution\ Time_A}}{\frac{Execution\ Time_{Ref}}{Execution\ Time_B}} = \frac{Execution\ Time_B}{Execution\ Time_A}$$

Choice of the Reference computer is irrelevant

$$Geometric\ Mean\ of\ SPEC\ Ratios = \sqrt[n]{\prod_{i=1}^{n} SPEC\ Ratio_i}$$

Execution Times & SPEC Ratios

Benchmark	Ultra 5 Time (sec)	Opteron Time (sec)	SpecRatio Opteron	Itanium2 Time (sec)	SpecRatio Itanium2	Opteron/ Itanium2 Times	Itanium2/ Opteron SpecRatios
wupwise	1600	51.5	31.06	56.1	28.53	0.92	0.92
swim	3100	125.0	24.73	70.7	43.85	1.77	1.77
mgrid	1800	98.0	18.37	65.8	27.36	1.49	1.49
applu	2100	94.0	22.34	50.9	41.25	1.85	1.85
mesa	1400	64.6	21.69	108.0	12.99	0.60	0.60
galgel	2900	86.4	33.57	40.0	72.47	2.16	2.16
art	2600	92.4	28.13	21.0	123.67	4.40	4.40
equake	1300	72.6	17.92	36.3	35.78	2.00	2.00
facerec	1900	73.6	25.80	86.9	21.86	0.85	0.85
ammp	2200	136.0	16.14	132.0	16.63	1.03	1.03
lucas	2000	88.8	22.52	107.0	18.76	0.83	0.83
fma3d	2100	120.0	17.48	131.0	16.09	0.92	0.92
sixtrack	1100	123.0	8.95	68.8	15.99	1.79	1.79
apsi	2600	150.0	17.36	231.0	11.27	0.65	0.65
Geometric Mean			20.86		27.12	1.30	1.30

Geometric mean of ratios = 1.30 = Ratio of Geometric means = 27.12 / 20.86

Things to Remember about Performance

- Two common measures: Response Time and Throughput
  - Response Time = duration of a single task
  - Throughput is a rate = Number of tasks per duration of time
- CPU Execution Time = Instruction Count × CPI × Cycle
- MIPS = Millions of Instructions Per Second (is a rate)
  - FLOPS = Floating-point Operations Per Second
- Amdahl's Law is a measure of speedup
  - When improving a fraction of the execution time
- Benchmarks: real applications are used
  - To compare the performance of computer systems
  - Geometric mean of SPEC ratios (for a set of applications)

Performance and Power

- Power is a key limitation
  - Battery capacity has improved only slightly over time
- Need to design power-efficient processors
- Reduce power by
  - Reducing frequency
  - Reducing voltage
  - Putting components to sleep
- Performance per Watt: FLOPS per Watt
  - Defined as performance divided by power consumption
  - Important metric for energy-efficiency

Power in Integrated Circuits

- Power is the biggest challenge facing computer design
  - Power should be brought in and distributed around the chip
  - Hundreds of pins and multiple layers just for power and ground
  - Power is dissipated as heat and must be removed
- In CMOS IC technology, dynamic power is consumed when switching transistors on and off

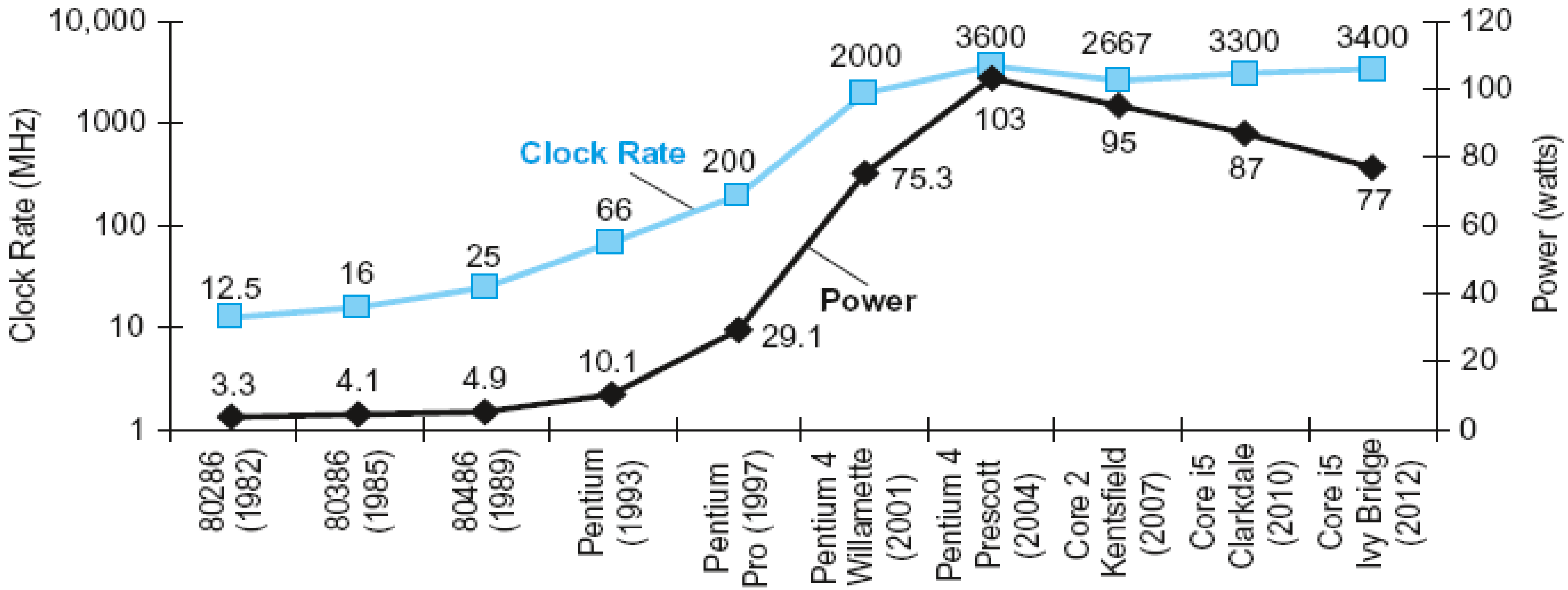
Dynamic Power = Capacitive Load × Voltage^2 × Frequency

Capacitive Load: × 40

Voltage: 5V → 1V

Frequency: × 1000

Trends in Clock Rates and Power



- Power Wall: Cannot Increase the Clock Rate
  - Heat must be dissipated from a 1.5 × 1.5 cm chip
  - Intel 80386 (1985) consumed about 2 Watts
  - Intel Core i7 running at 3.3 GHz consumes 130 Watts
  - This is the limit of what can be cooled by air

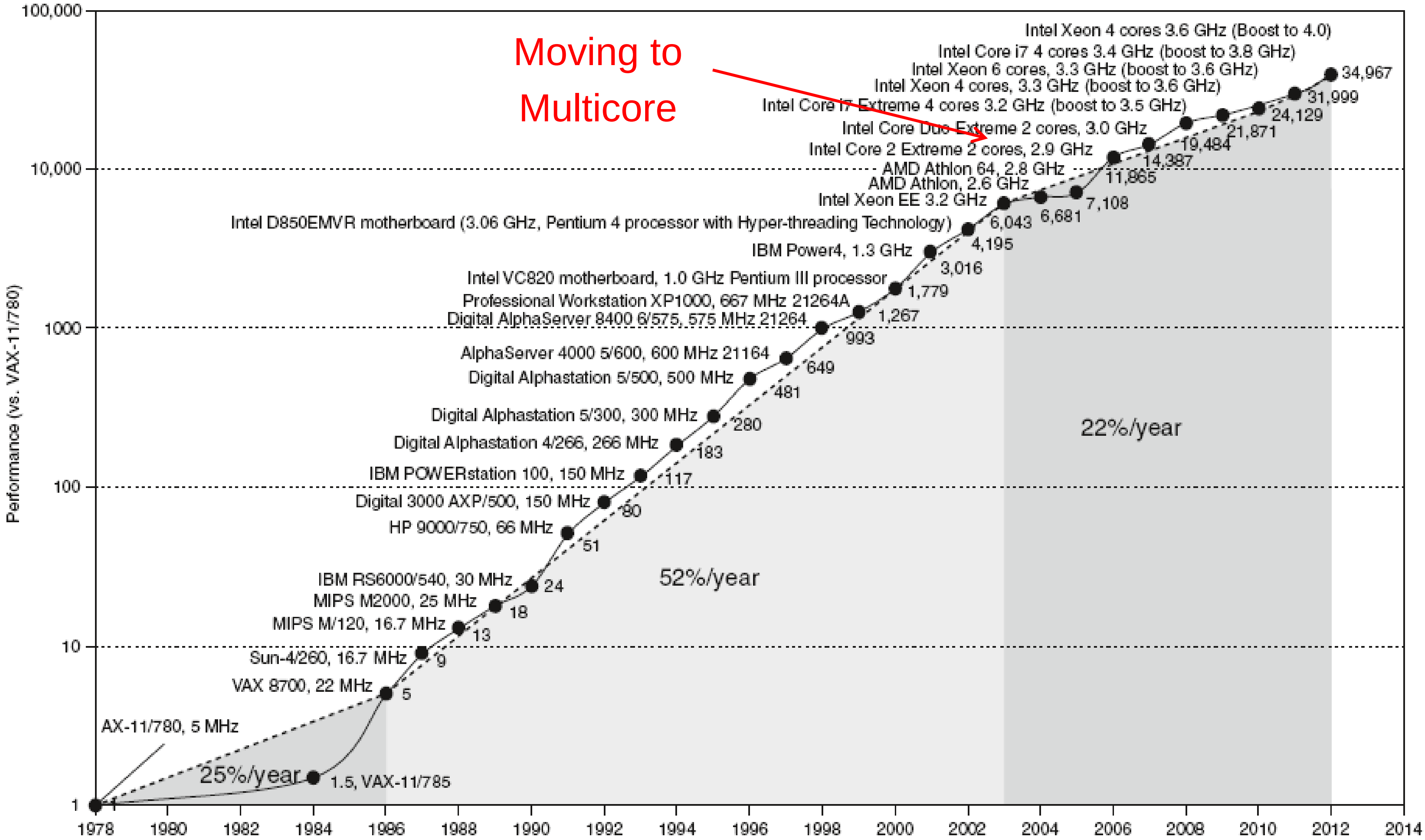
Example on Power Consumption

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction
- Relate the Power consumption of the new and old CPUs
- Answer:

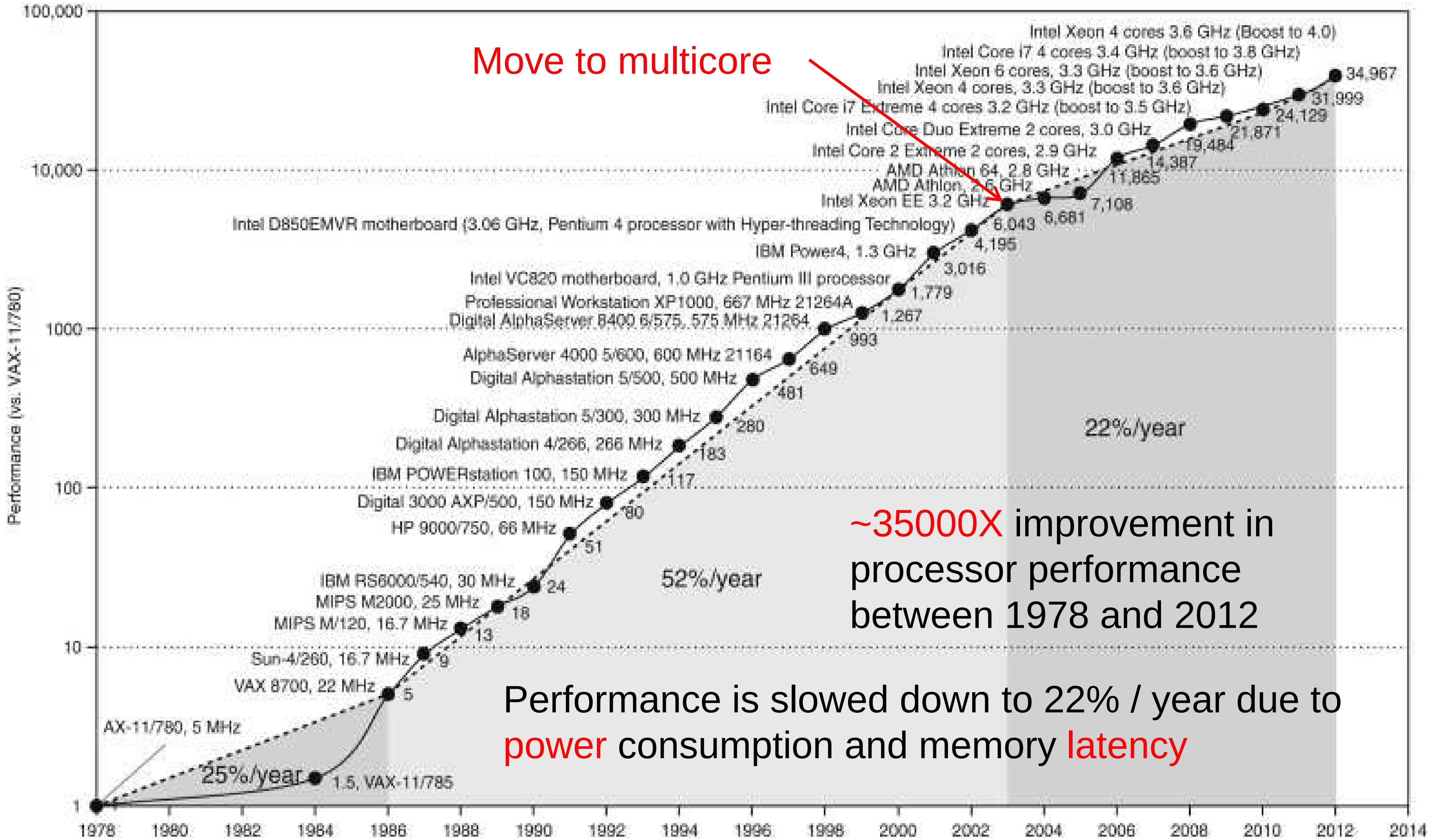
$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^2 \times F_{old}} = 0.85^4 = 0.52$$

- The Power Wall
  - We cannot reduce voltage further
  - We cannot remove more heat from the integrated circuit
- How else can we improve performance?

Moving to Multicores



Processor Performance



Move to multicore

~35000X improvement in processor performance between 1978 and 2012

Performance is slowed down to 22% / year due to power consumption and memory latency

Multicore Processors

- Multiprocessor on a single chip
- Requires explicit parallel programming
- Harder than sequential programming
  - Parallel programming to achieve higher performance
  - Optimizing communication and synchronization
  - Load Balancing
- In addition, each core supports instruction-level parallelism
  - Parallelism at the instruction level
  - Parallelism is extracted by the hardware or the compiler
  - Each core executes multiple instructions each cycle
  - This type of parallelism is hidden from the programmer