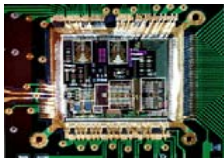
Lecture 4: Metrics

CSCE5610 Computer System Architecture
CSCE4610 Computer Architecture

Instructor: Saraju P. Mohanty, Ph. D.

NOTE: The figures, text etc included in slides are borrowed from various books, websites, authors pages, and other sources for academic purpose only. The instructor does not claim any originality.



What is Performance?

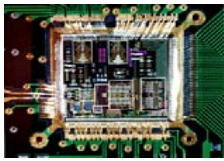
- As a computer user, *you* define the *performance*!
- Different users have different definitions of performance!
- You are already familiar with some performance metrics:

Clock Frequency:

My computer runs at 1000MHz, yours only at 300MHz, so my computer is better than yours!

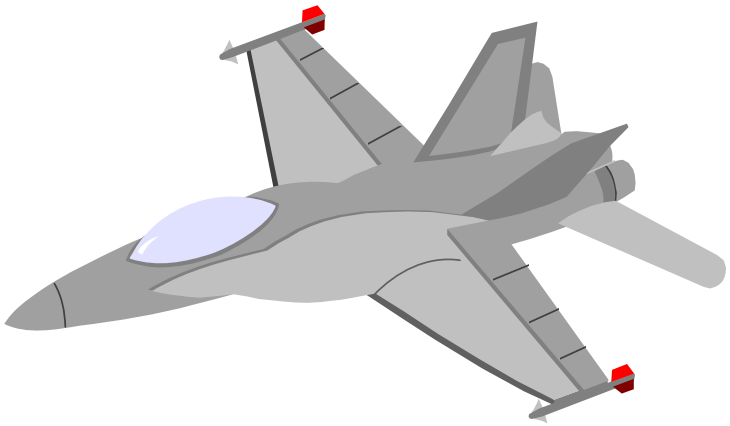
Millions of Instructions Per Second (MIPS):

In a second, my computer can execute 10 million instructions, while yours only 5 million instructions in a second, so my computer is better than yours!



Defining Performance: Airplane Example

Plane	DC to Paris	Speed	Passengers	Throughput (passenger x mph)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concodre	3 hours	1350 mph	132	178,200

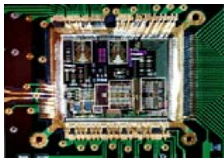


Which airplane performs better?

" X is n times faster than Y" means

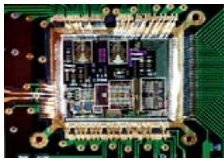
$$\frac{\text{ExTime(Y)}}{\text{ExTime(X)}} = \frac{\text{Performance(X)}}{\text{Performance(Y)}} = n$$

- Time of Concorde vs. Boeing 747?
- Throughput of Boeing 747 vs. Concodre?

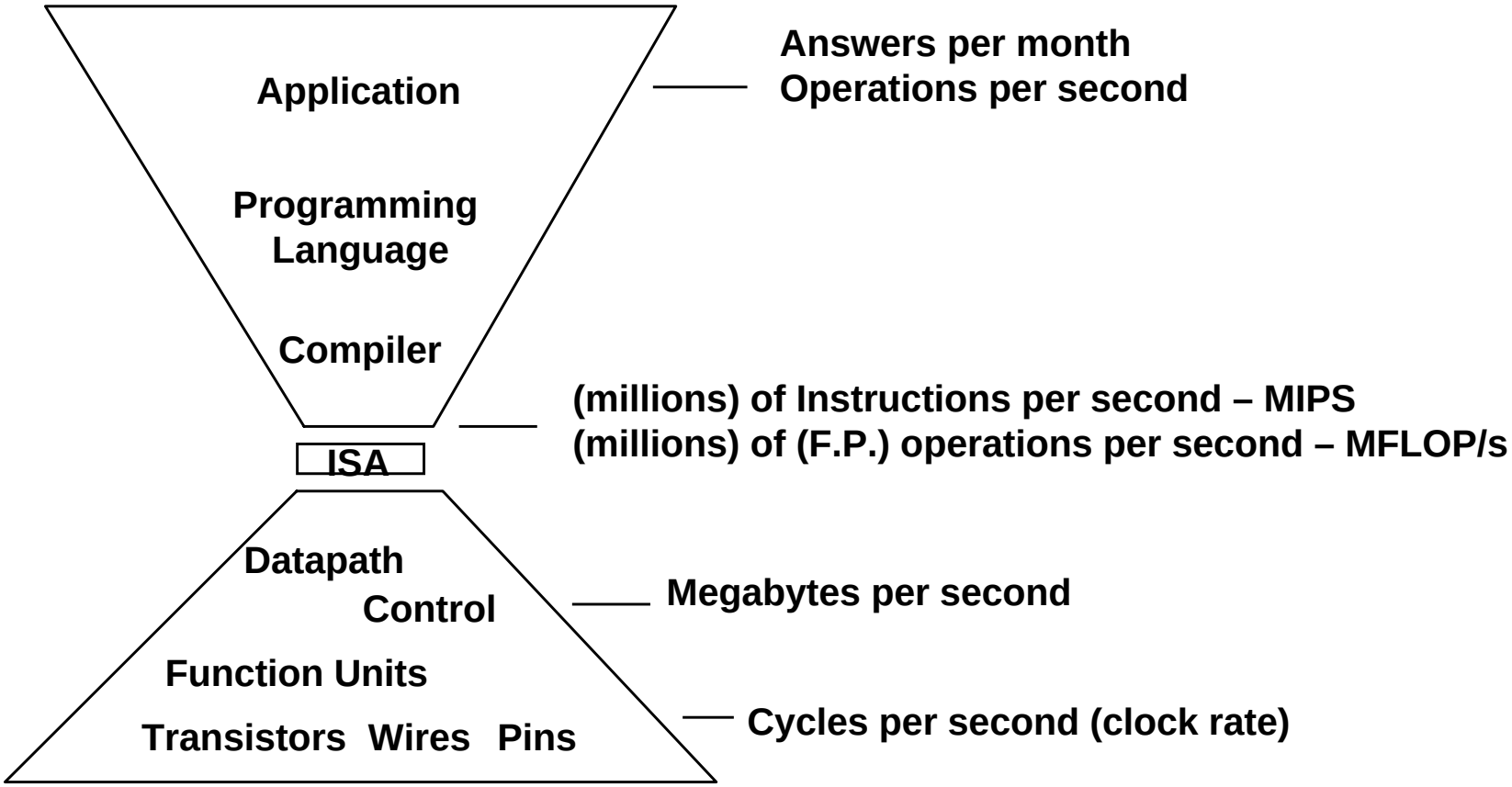


Why measure performance?

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing hype
- Key to understanding underlying organizational motivation
- We are interesting in answering questions of the following nature:
  - *Why is some hardware better than others for different programs?*
  - *What factors of system performance are hardware related? (e.g., Do we need a new machine, or a new operating system?)*
  - *How does the machine's instruction set affect performance?*



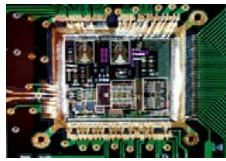
Metrics of performance



Computer Performance

- Response Time (latency)
  - — How long does it take for my job to run?
  - — How long does it take to execute a job?
  - — How long must I wait for the database query?
- Throughput
  - — How many jobs can the machine run at once?
  - — What is the average execution rate?
  - — How much work is getting done?
- *If we upgrade a machine with a new processor what do we increase?*

  If we add a new machine to the lab what do we increase?



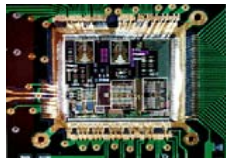
Program Execution Time

Total Execution time of your application is the TRUE measure of performance. Why? Because that is what *we are really interested in*!

- "time" command on UNIX gives us program execution time statistics

  Example: Type "time ls" to time the listing of the current directory.

- Elapsed Time
  - counts everything *(disk and memory accesses, I/O , etc.)*
  - a useful number, but often not good for comparison purposes
- CPU time
  - doesn't count I/O or time spent running other programs
  - can be broken up into system time, and user time
- Our focus: user CPU time
  - time spent executing the lines of code that are "in" our program



Now lets define performance..

- For some program running on machine X,

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

- "X is n times faster than Y"

  $$\text{Performance}_X \ / \ \text{Performance}_Y = n$$

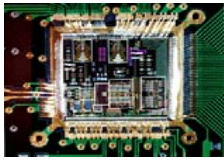
Problem:

– machine A runs a program in 20 seconds

– machine B runs the same program in 25 seconds

Which machine is better?

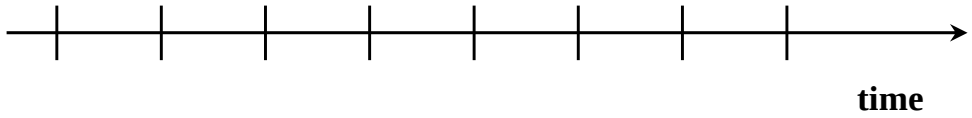
What are their performance values?



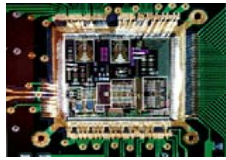
Clock Cycles

- Instead of reporting execution time in seconds, we often use cycles

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$



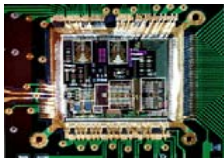
- Clock “ticks” indicate when to start activities (one abstraction):
- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second (1 Hz = 1 cycle/sec)
  A 200 Mhz. clock has a cycle time $\frac{1}{200 \times 10^6} \times 10^9 = 5$ nanoseconds



How to Improve Performance

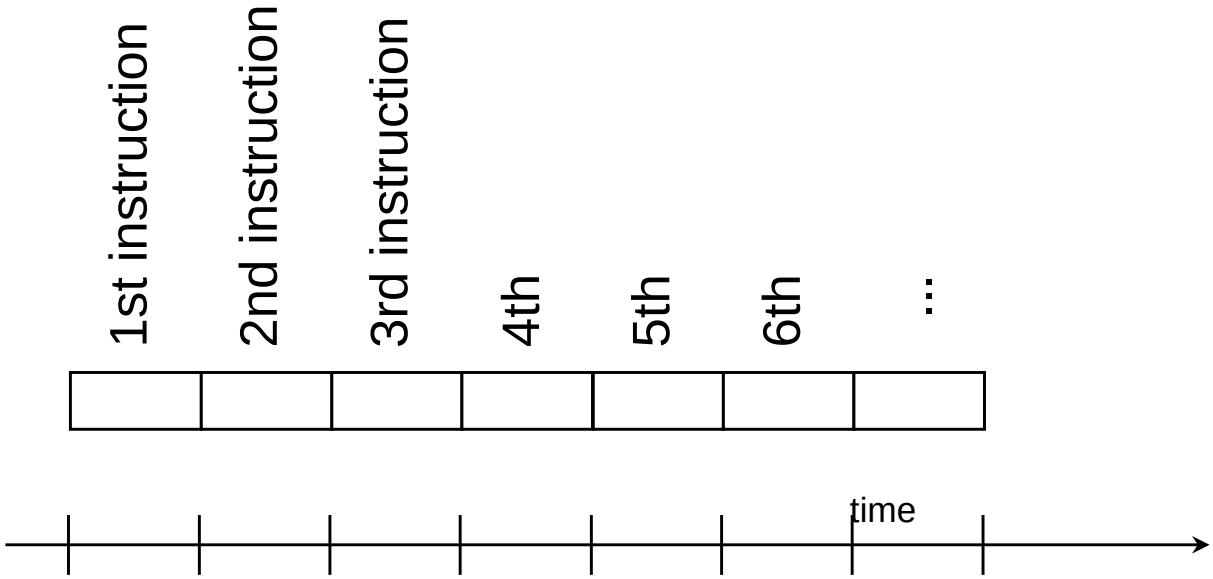
$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

So, to improve performance (everything else being equal) you can either
___reduce_____ the # of required cycles for a program, or
__reduce_____ the clock cycle time or, said another way,
__increase_____ the clock rate.



How many cycles are required for a program?

- Could assume that # of cycles = # of instructions

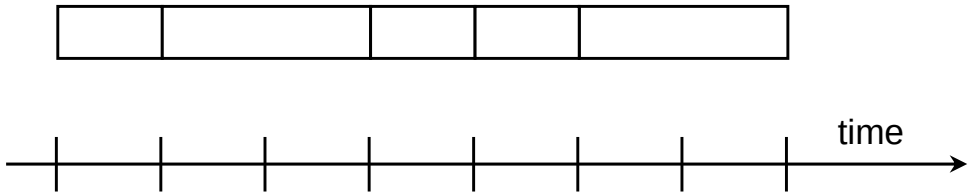


This assumption is incorrect,

different instructions take different amounts of time on different machines.

Why? hint: remember that these are machine instructions, not lines of C code

Different numbers of cycles for different instructions

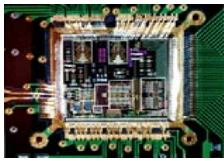


- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers
- *Important point: changing the cycle time often changes the number of cycles required for various instructions (more later)*

An Example

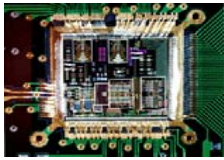
- Our favorite program runs in 10 seconds on computer A, which has a 4GHz. clock. We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program. What clock rate should we tell the designer to target?“

Page-247, Interface book



Now that we understand cycles

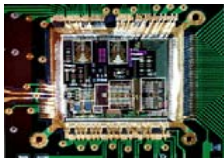
- A given program will require
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds
- We have a vocabulary that relates these quantities:
  - cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - CPI (cycles per instruction)
    a floating point intensive application might have a higher CPI
  - MIPS (millions of instructions per second)
    this would be higher for a program using simple instructions



Performance

- Performance is determined by execution time
- Do any of the other variables equal performance?
  - # of cycles to execute program?
  - # of instructions in program?
  - # of cycles per second?
  - average # of cycles per instruction?
  - average # of instructions per second?

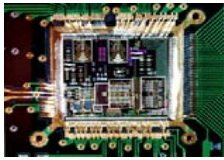
- Common pitfall: thinking one of the variables is indicative of performance when it really isn't.



CPI Example

- Suppose we have two implementations of the same instruction set architecture (ISA).
  For some program,
  Machine A has a clock cycle time of 250ps and a CPI of 2.0
  Machine B has a clock cycle time of 500ps and a CPI of 1.2
  What machine is faster for this program, and by how much?

NOTE: If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical? Page-248, Interface book.



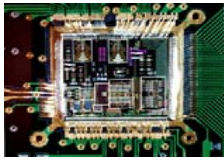
of Instructions Example

- 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 (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.

  Which sequence will be faster? How much? What is the CPI for each sequence?

Page-252, Interface Book.



MIPS example

- Two different compilers are being tested for a 4GHz machine with three different classes of instructions: Class A, class B, and class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.

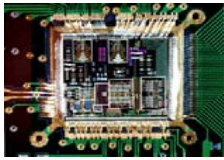
The first compiler's code uses 5 billion class A instructions, 1 billion class B instructions, and 1 billion class C instructions.

The second compiler's code uses 10 billion class A instructions, 1 billion class B instructions, and 1 billion class C instructions.

Which sequence will be faster according to execution time?

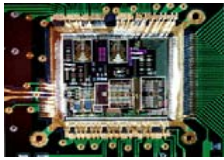
Which sequence will be faster according to MIPS?

Page-268, Interface Book



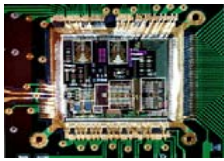
Why Do Benchmarks?

- How we evaluate differences
  - Different systems
  - Changes to a single system
- Provide a target
  - Benchmarks should represent large class of important programs
  - Improving benchmark performance should help many programs
- For better or worse, benchmarks shape a field
- Good ones accelerate progress
  - good target for development
- Bad benchmarks hurt progress
  - help real programs v. sell machines/papers?
  - Inventions that help real programs don't help benchmark



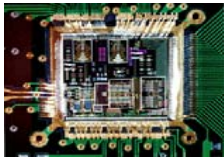
Programs to Evaluate Performance

- (Toy) Benchmarks
  - 10-100 line
  - e.g.: sieve, puzzle, quicksort
- Synthetic Benchmarks
  - attempt to match average frequencies of real workloads
  - e.g., Whetstone, dhrystone
- Kernels
  - Time critical excerpts Real programs
  - e.g., gcc, spice



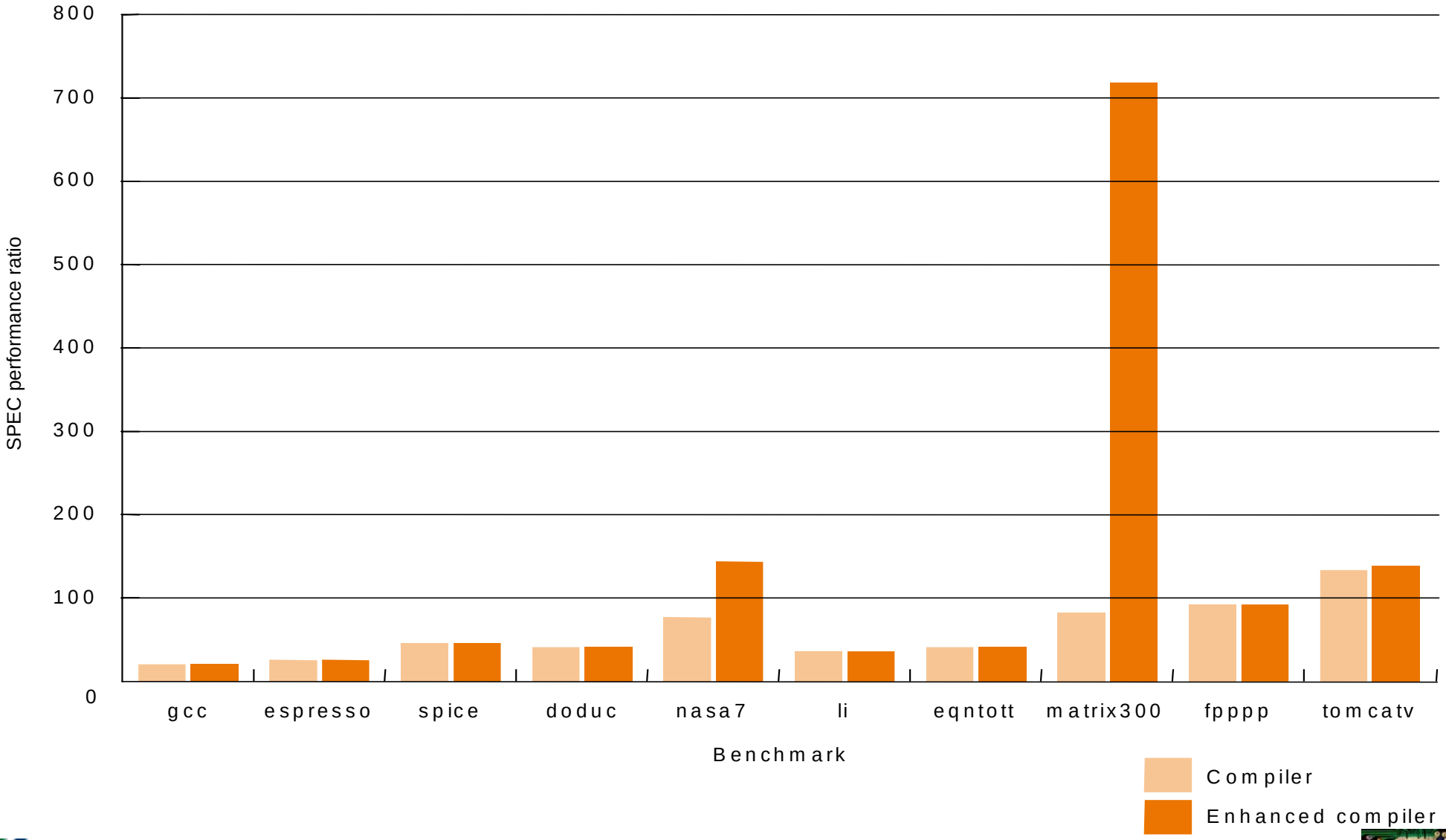
Benchmarks

- Performance best determined by running a real application
  - Use programs typical of expected workload
  - Or, typical of expected class of applications e.g., compilers/editors, scientific applications, graphics, etc.
- Small benchmarks
  - nice for architects and designers
  - easy to standardize
  - can be abused
- SPEC (System Performance Evaluation Cooperative)
  - companies have agreed on a set of real program and inputs
  - valuable indicator of performance (and compiler technology)



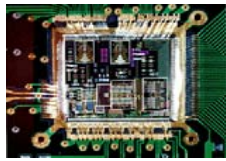
SPEC ‘89

- Compiler “enhancements” and performance



SPEC '95 CPU Benchmarks

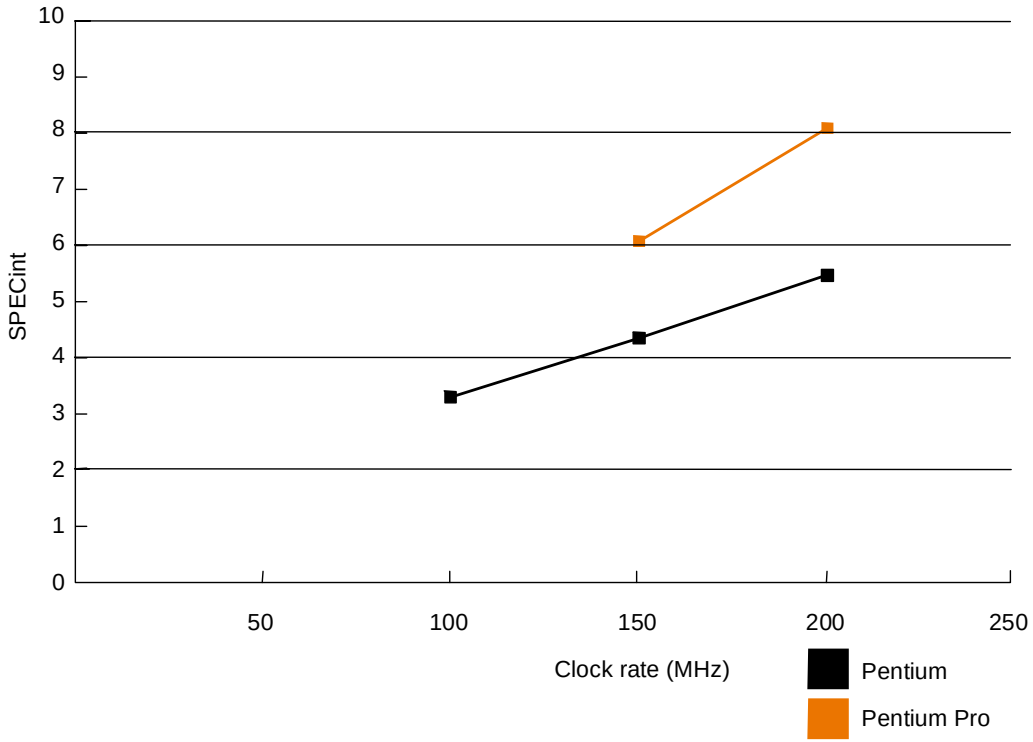
Benchmark	Description
go	Artificial intelligence; plays the game of Go
m88ksim	Motorola 88k chip simulator; runs test program
gcc	The Gnu C compiler generating SPARC code
compress	Compresses and decompresses file in memory
li	Lisp interpreter
ijpeg	Graphic compression and decompression
perl	Manipulates strings and prime numbers in the special-purpose programming language Perl
vortex	A database program
tomcatv	A mesh generation program
swim	Shallow water model with 513 x 513 grid
su2cor	quantum physics; Monte Carlo simulation
hydro2d	Astrophysics; Hydrodynamic Naiver Stokes equations
mgrid	Multigrid solver in 3-D potential field
applu	Parabolic/elliptic partial differential equations
trub3d	Simulates isotropic, homogeneous turbulence in a cube
apsi	Solves problems regarding temperature, wind velocity, and distribution of pollutant
fpppp	Quantum chemistry
wave5	Plasma physics; electromagnetic particle simulation

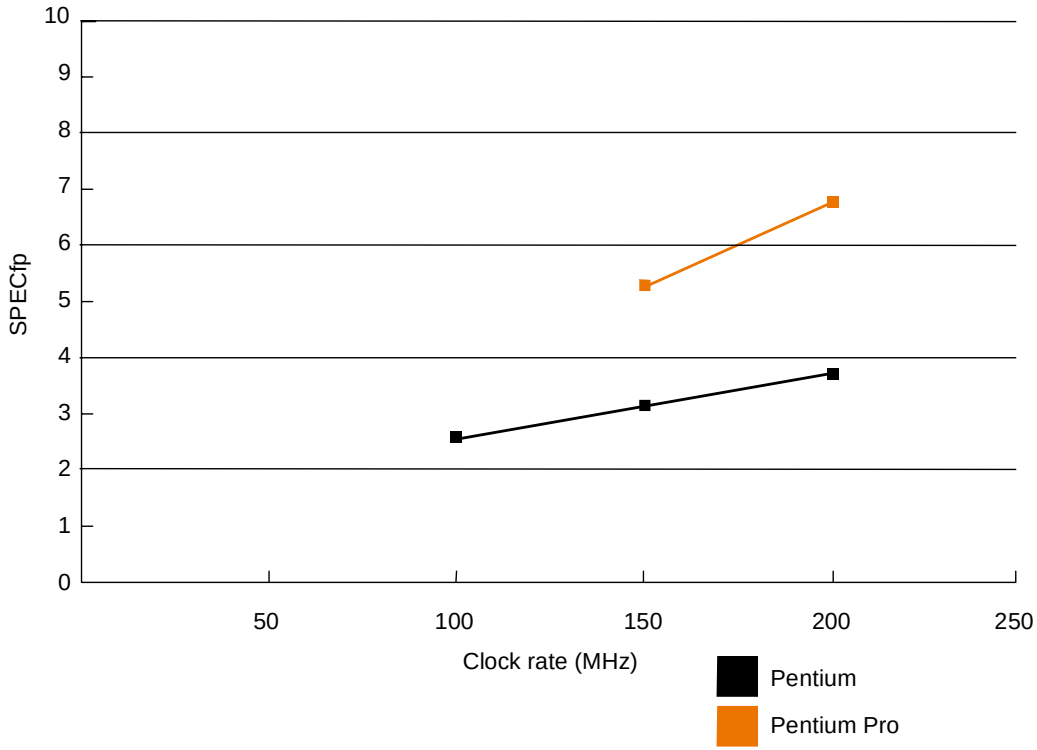


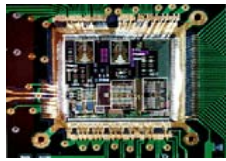
SPEC '95

Does doubling the clock rate double the performance?

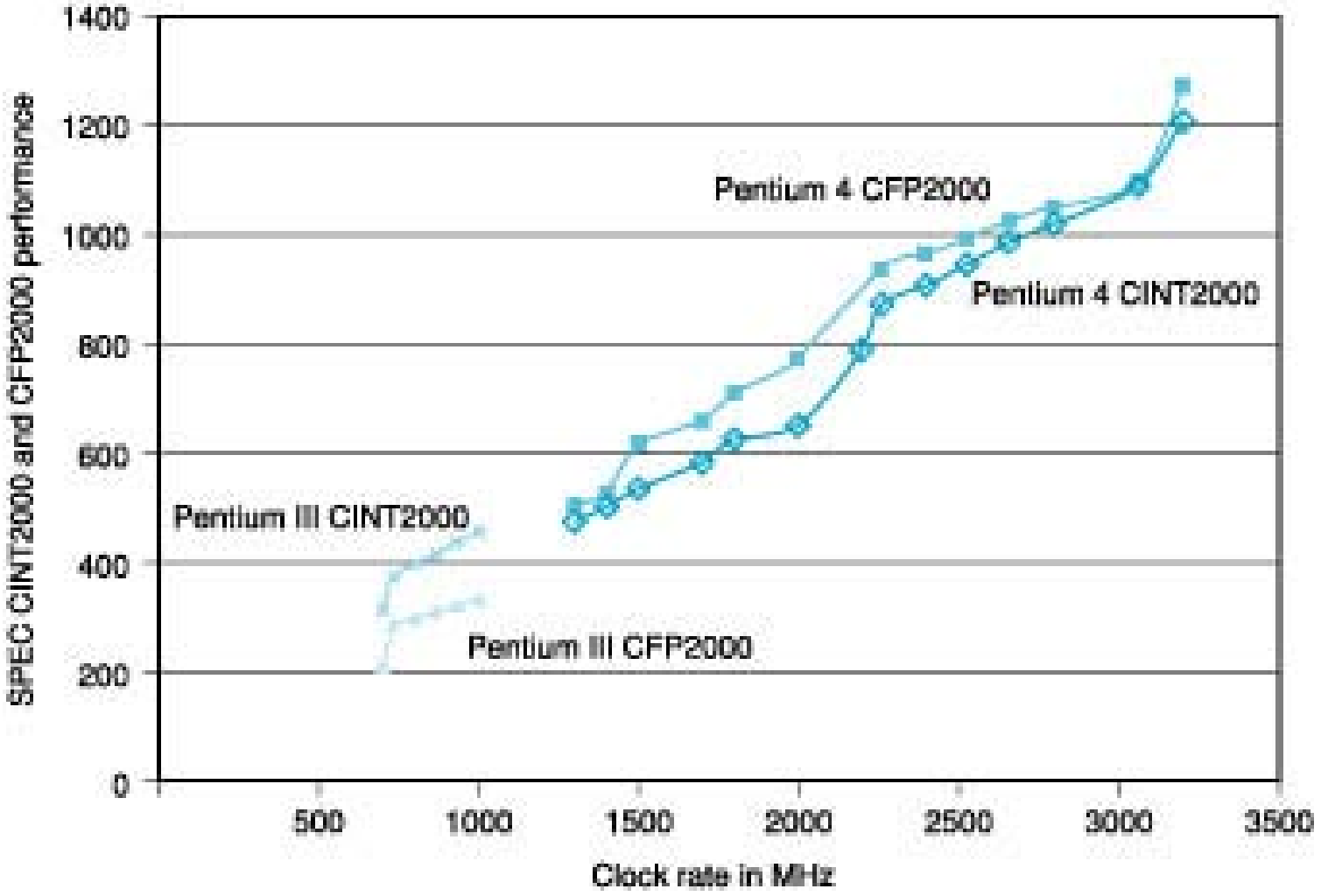
Can a machine with a slower clock rate have better performance?



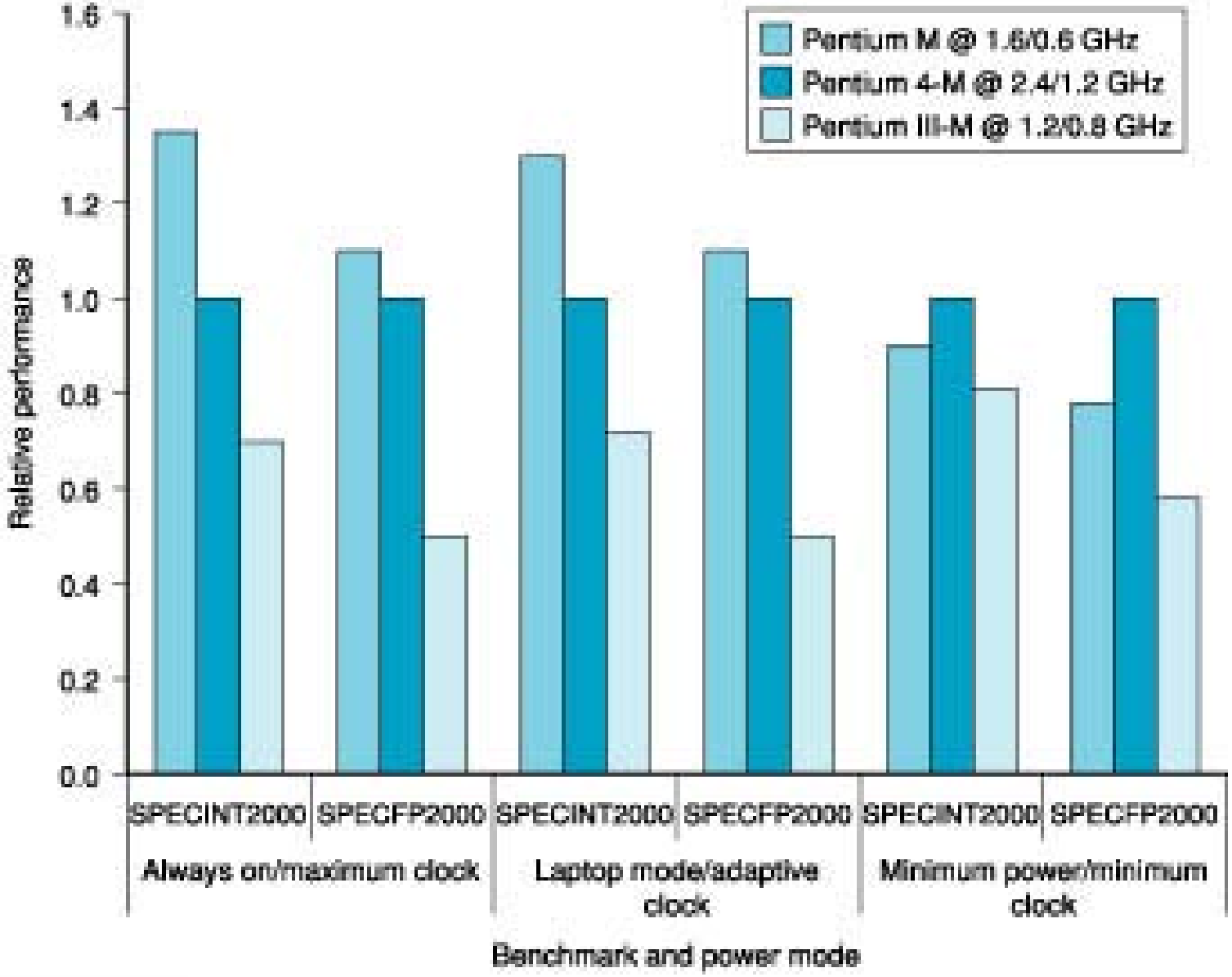


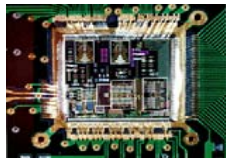


Rating for Intel Pentium III and 4 for SPEC

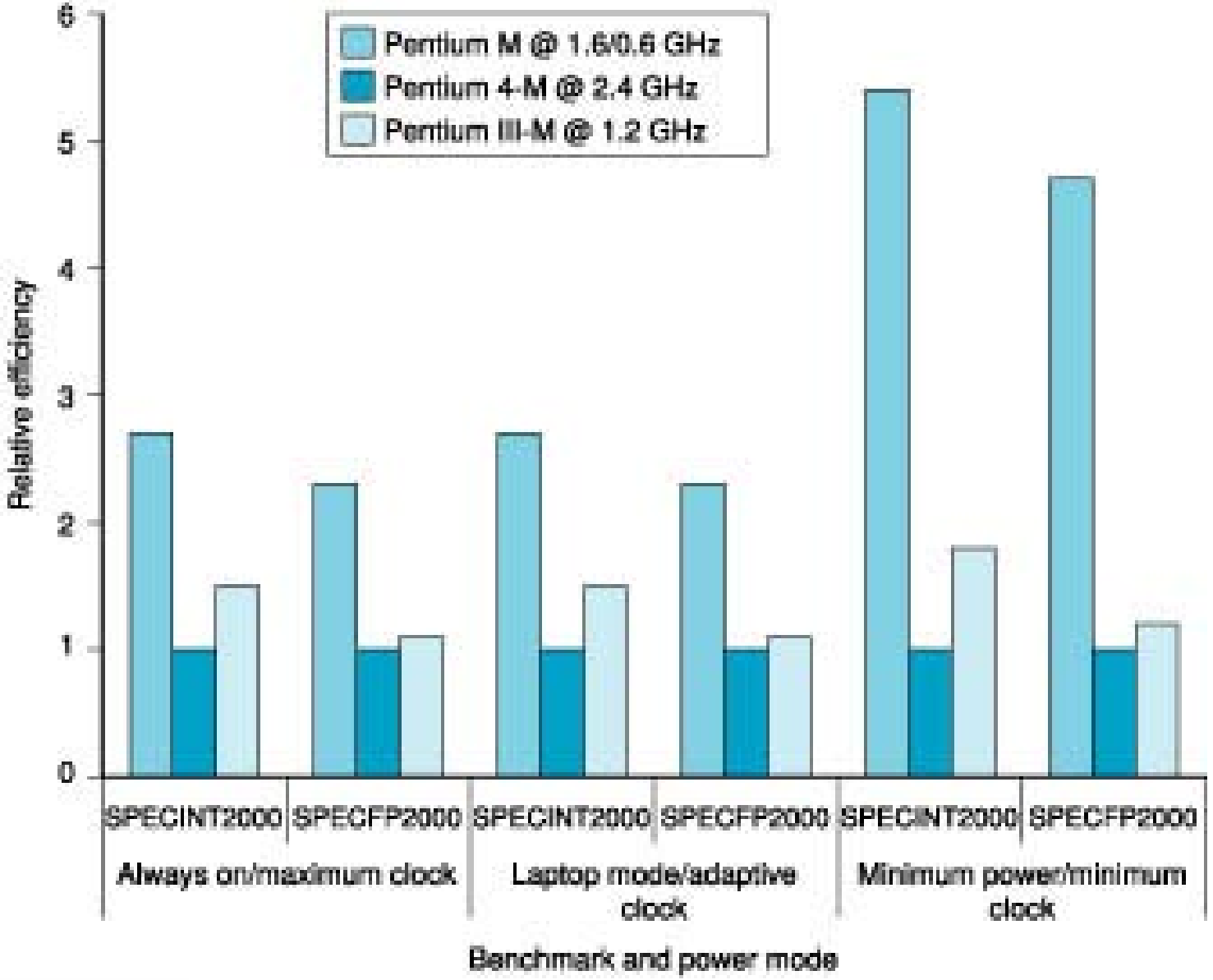


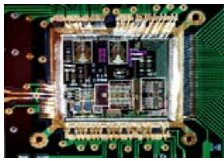
Relative Performance of 3 Intel Processors





Relative Energy Efficiency of 3 Mobile Pentium Processor





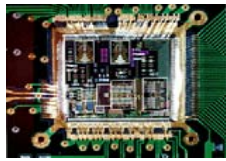
Amdahl's Law

Execution Time After Improvement
= Execution Time Unaffected +
(Execution Time Affected / Amount of Improvement)

Example (Page-266, Interface Book):

- 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 5 times faster? How about making it 4 times faster?

- **Pitfall**: Expecting the improvement of one aspect of a computer to increase by an amount proportional to the size of the improvement.



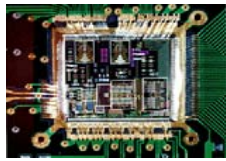
Amdahl's Law

The overall speed up ratio of the execution times is given by:

$$Speedup_{overall} = (Execution\ Time_{old} / Execution\ Time_{new})$$
$$= 1/ [1 – Fraction_{enhanced} + (Fraction_{enhanced} / Speedup_{enhanced})]$$

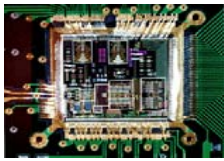
Example: Page-40, Quantitative Book

Suppose we want to enhance the processor used for Web serving. The new processor is 10 times faster on computation in the Web serving application than the original processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?



Amdahl's Law

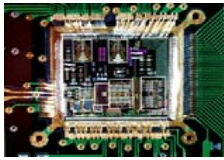
- The performance improvement to be gained from using some faster mode of execution is limited by the fraction of time the faster mode can be used. -- the computer law of diminishing returns.



Performance Evaluation Summary

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

- Time is the measure of computer performance!
- Good products created when have:
  - –Good benchmarks
  - –Good ways to summarize performance
- If not good benchmarks and summary, then choice between improving product for real programs vs. improving product to get more sales=> sales almost always wins.
- Remember Amdahl's Law: Speedup is limited by unimproved part of program.



Fallacies & Pitfalls

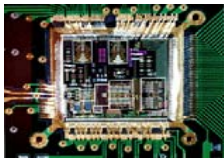
- Expecting the improvement of one aspect of a machine to increase performance by an amount proportional to the size of improvement .. *Look at Amdahl's Law*
- Hardware-independent metrics predict performance.
- Using MIPS as a performance metric.

Counter example (Page-268, Interface Book): Machine with three instruction classes. Suppose we measure the code for the same program from two different compilers as shown below. Assume clock frequency is 4GHz. Which code sequence will execute faster according to execution time? According to MIPS? (Note: Revisited from Slide 19)

Code From	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1
CPI	1	2	3

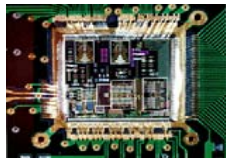
Fallacies & Pitfalls.. Contd.,

- Synthetic benchmarks predict performance
- Using arithmetic mean of normalized execution times to predict performance
- The geometric mean of execution time ratios is proportional to total execution time



Variants of MIPS: MOPS and Other FLOPS!

- MIPS = Million Instructions Per Second
- Peak MIPS is obtained by choosing an instruction mix that minimizes CPI, even if the mix is impractical!!
- Computer manufacturers announce produces using peak MIPS as a metric, often neglecting the word “peak”
- MOPS = Million Operations Per Second
- Relative MIPS = (Time reference / Time unrated) x MIPS reference



Cost Metric: Dies per Wafer

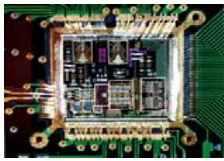
Cost of Integrated Circuit

- Cost of the die (or chip)
- Cost of testing
- Cost of packaging

Cost of the die depends on:

- the number of chips per wafer
- how many good dies per wafer (or yield)

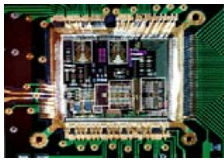
Dies per wafer: $\frac{\pi * (\text{Wafer diameter}/2)^2}{(\text{Chip area})} - \frac{\pi * (\text{Wafer diameter})}{\sqrt{2 * (\text{Chip area})}}$



Cost Metric: Dies per Wafer Example

- Find the number of dies per 300mm (30cm) wafer for a die that is 1.5cm on a side.

Page-22, Quantitative Book.



Cost Metric: Die Yield

$$\text{Die Yield} = (\text{wafer yield}) * \left(1 + \frac{(\text{defects per unit area}) * (\text{die area})}{\alpha}\right)^{-\alpha}$$

Where, wafer yield accounts for wafers that are completely bad and so need not be tested, for simplicity assume as 100%.

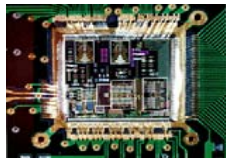
4 is a good number for a typical CMOS process.

What does all this mean?

The die area can have a huge impact on the cost of a chip.

If we double chip (die) size, we have half as many chips per wafer and the yield drops quadratically.

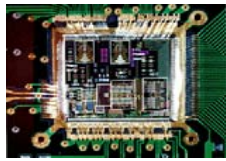
So we need to use silicon area wisely.



Cost Metric: Die Yield Example

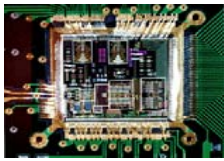
Find the die yield for a die that is 1.5cm on a side assuming a defect density of 0.4 per cm^2 and α is 4. Repeat the same for a die of 1.0cm on a side.

Page-24, Quantitative Book.



Power Metric: Dynamic and Static Power

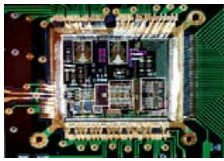
- $Power_{dynamic}$
  $= ½ * Capacitive\ load \times Voltage^2 * Frequency$
- $Energy_{dynamic} = Capacitive\ load \times Voltage^2$
- $Power_{static} = Current_{static} \times Voltage$



Power Metric: Dynamic Power Example

Example: Dynamic Power (Page-18, Quantitative Book)

Some microprocessors today are designed to have adjustable voltage, so that a 15% reduction in voltage may result in a 15% reduction in frequency. What would be the impact on dynamic power?



Summary

- Three metrics: Performance, cost, and power.
- Performance is specific to a particular program/s
  - Total execution time is a consistent summary of performance
- For a given architecture performance increases come from:
  - increases in clock rate (without adverse CPI affects)
  - improvements in processor organization that lower CPI
  - compiler enhancements that lower CPI and/or instruction count