EECE 321: Computer Organization

Mohammad M. Mansour
Dept. of Electrical and Compute Engineering
American University of Beirut

Lecture 16: Performance

Defining Performance

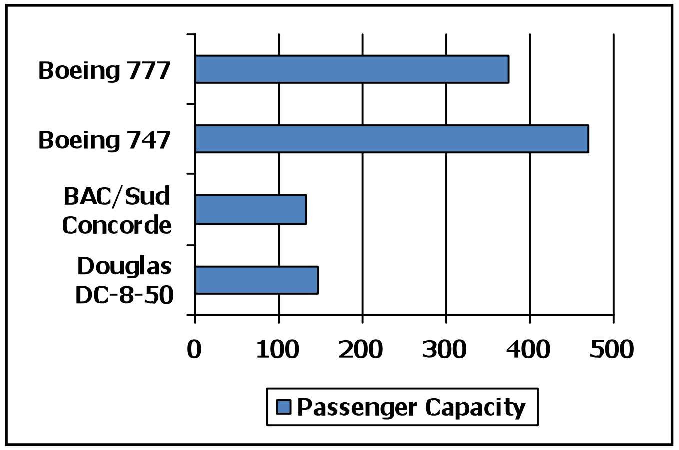
- In this part of the course we are concerned with assessing the performance of a computer.
- Why is performance important?
  - It enables making intelligent choices
  - See through the marketing hype: Does it really work as fast as they claim?
  - It is key to understanding underlying organizational motivation.
- How do we compare different computers? Why is some hardware better than others for different programs?
- What factors of system performance are hardware related?
  - For example, do we need a new machine, or a new operating system?
- How does the machine's instruction set affect performance?
  - Do we need more simple instructions, or few complex instructions?
  - What type of instructions do we need to include?
    - Ex: For multimedia applications, Intel added specific MMX instructions
- To answer these questions, we need to understand what determines the performance of a machine.

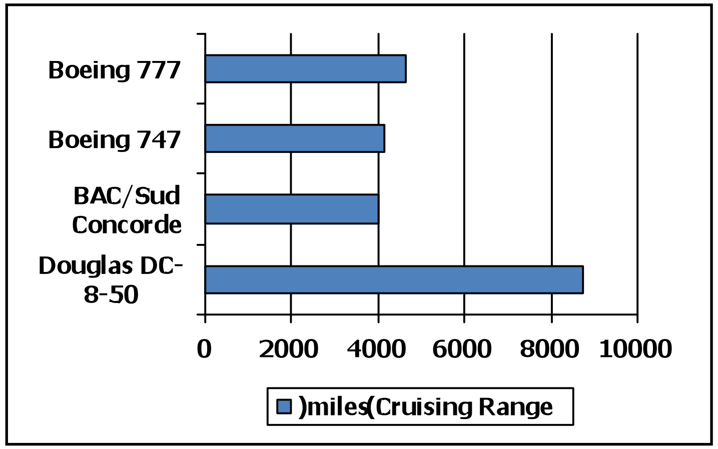
Which of these airplanes has the best performance?

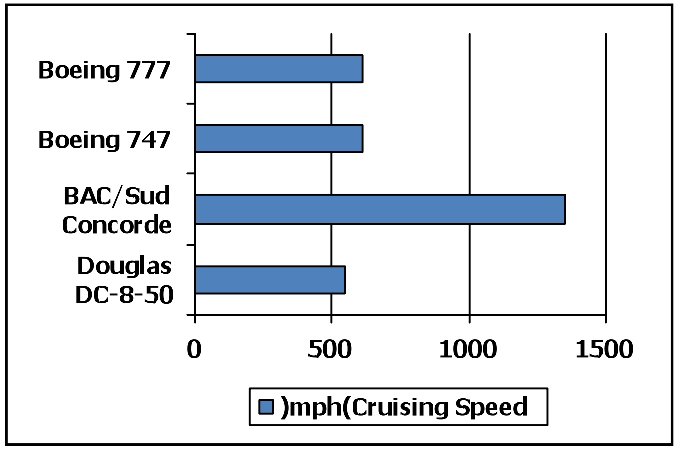
Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 777	375	4630	610
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

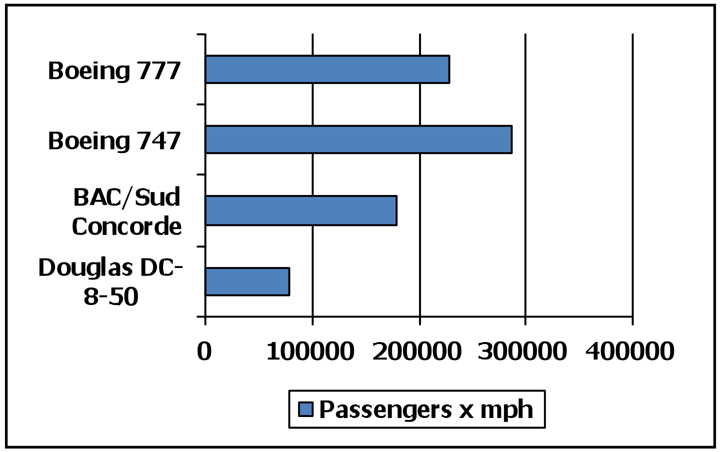
- Performance: *Fastest*, *largest*, *longest range*?
- Which plane transfers a single passenger 4000 miles in the shortest time?
- Which plane transfers 470 passengers 4000 miles in the shortest time?
- Performance can refer to completing a job as quickly as possible, or completing the most jobs in a given time.
- Example:
  - A program is running on 2 different workstations, the faster workstation is the one that gets the job done first.
  - Here one is interested in reducing *response time* or *execution time*.
  - If a computer center maintains two time-shared computers that run jobs submitted by many users, the faster computer is the one that completes the most jobs per day.
  - Here one is interested in maximizing *throughput*.

Which airplane has the best performance?









Computer Performance: TIME

- Metric: **Response Time or 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?

- Metric: **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?

- In discussing the performance of machines, we will be primarily concerned with CPU execution time.
  - This is the time spent executing the lines of code that are "in" our program.
  - Time spent on I/O, or running other programs, or OS time is not included.

Performance Metrics

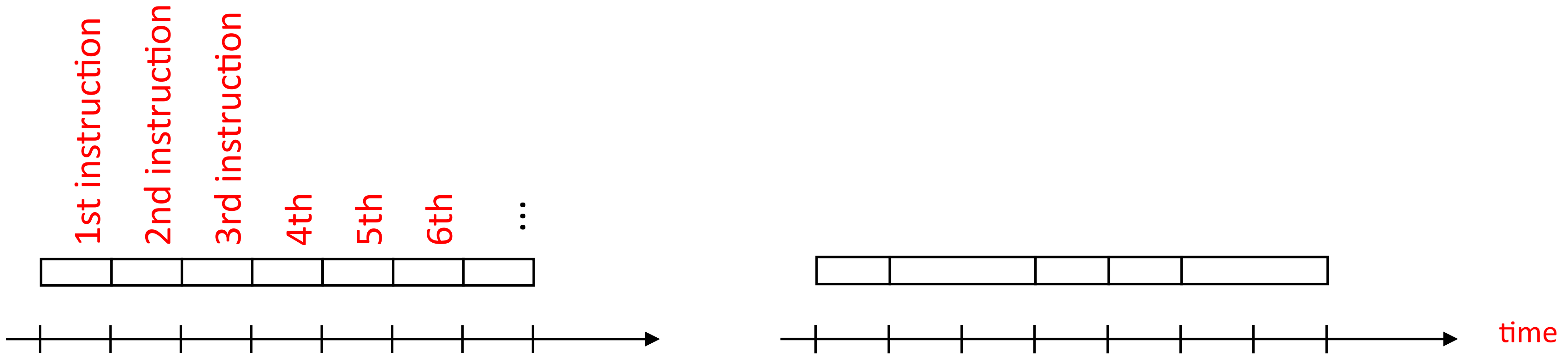
- For some program running on machine X,
  $\text{Performance}_X = 1 / \text{Execution time}_X$
- "X is *n* times faster than Y"
  $\text{Performance}_X \; / \; \text{Performance}_Y = n$
- Problem:
  - Machine A runs a program in 20 seconds
  - Machine B runs the same program in 25 seconds
  - Which is faster and by how much?
- Instead of reporting execution time in seconds, we often use clock cycles:

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

- CPU execution time = # CPU clock cycles x Clock cycle time.
  = CPU clock cycles / Clock frequency.
- So, to improve performance (everything else being equal) you can either
  - Decrease the # of required cycles for a program,
  - Decrease 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?
  - Incorrect assumption: Different instructions take different amounts of time on different machines.
- Need 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.
- Can compute *average clock cycles per instruction* (CPI), so
  - CPU clock cycles = Instructions per program x Avg. CC per instruction.
- Note: Changing the cycle time often changes the number of cycles required for various instructions.

Putting Things Together

- So, CPU time = Instruction count x CPI x Clock cycle time.

$$CPU\ Time = \frac{Instructions}{Program} \times \frac{Clock\ Cycles}{Instruction} \times \frac{Seconds}{Clock\ cycle}$$

- A given program will require
  - some number of 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

- Another performance metric: MIPS (millions of instructions per second)

  this would be higher for a program using simple instructions

$$\text{MIPS} = \frac{\#\text{Instructions}}{\text{Exec. Time} \times 10^6}$$

Performance

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

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

Example 1: Cycles Per Instruction

- Suppose we have two implementations (machine A and machine B) of the same instruction set architecture (ISA).

  For some program,

  Machine A has a clock cycle time of 1 ns and a CPI of 2.0
  Machine B has a clock cycle time of 2 ns and a CPI of 1.2

  Which machine is faster for this program, and by how much?

- **Answer**: $\frac{CPU\ performance_A}{CPU\ performance_B} = \frac{Execution\ time_B}{Execution\ time_A} = \frac{I \times 1.2 \times 2}{I \times 2 \times 1} = 1.2$

- Hence machine A is 1.2 times faster than machine B for this program.
  - We can't generalize about other programs

- 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?

Example 2: Number of Instructions

- 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?

- Answer: $CPU\_cycles_1 = 2x1 + 1x2 + 2x3 = 10$ cycles
  $CPU\_cycles_2 = 4x1 + 1x2 + 1x3 = 9$ cycles
  - Note: Same machine implies same clock period

- So sequence 2 will be faster even though it contains one more instruction.
  - $CPI_1 = CPU\_cycles_1 / Instruction\_count_1 = 10/5 = 2$
  - $CPI_2 = CPU\_cycles_2 / Instruction\_count_2 = 9/6 = 1.5$

CPI in More Detail

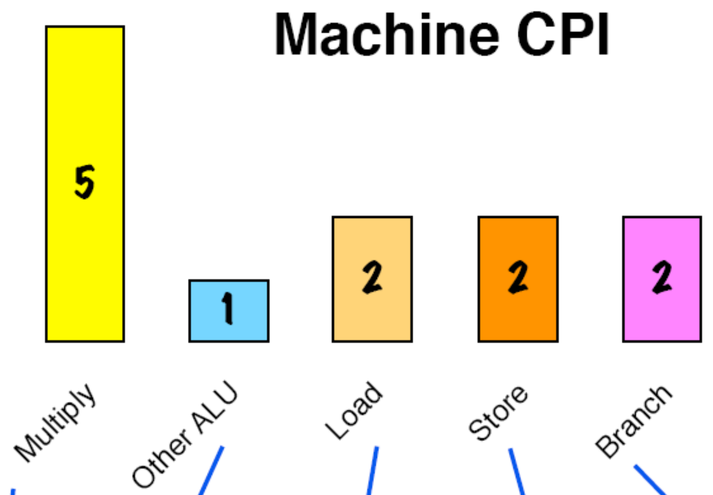
- If different instruction classes take different numbers of cycles

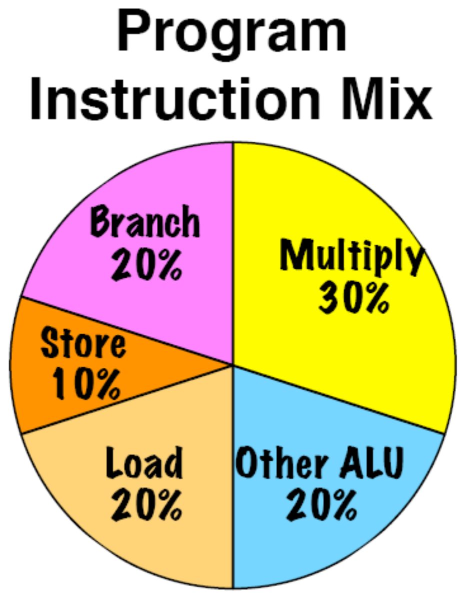
$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\boxed{\text{Relative frequency}}} \right)$$

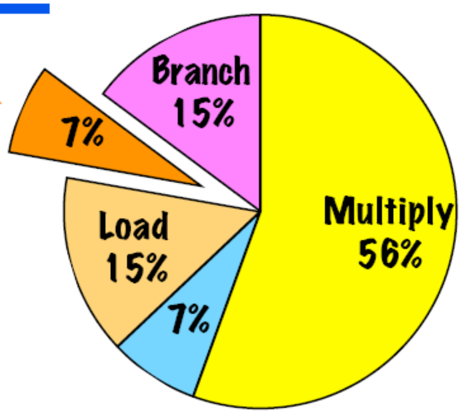
Example 3: Average CPI





$$\frac{5 \times 30 + 1 \times 20 + 2 \times 20 + \boxed{2 \times 10} + 2 \times 20}{100}$$

$= 2.7$ cycles/instruction



Where program spends its time

Example 4: Evaluating Performance Using MIPS as a Metric

- Two different compilers are being tested for a 500 MHz 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 MIPS?
- Which sequence will be faster according to execution time?

- Answer: MIPS = Instruction count / (Execution time x 10^6)

  Execution time = CPU cycles / clock rate

Execution time$_1$ = (5x1+1x2+1x3) x 10^9 / 5x10^8 = 20 seconds

Execution time$_2$= (10x1+1x2+1x3) x 10^9 / 5x10^8 = 30 seconds

MIPS$_1$ = 6 x 10^9/ 20 x 10^6 = 350

MIPS$_2$ = 12 x 10^9/ 30 x 10^6 = 400

Understanding Program Performance

Hardware or Software component	Affects what?
Algorithm	Instruction count, possibly CPI
Programming language	Instruction count, CPI
Compiler	Instruction count, CPI
Instruction set architecture (ISA)	Instruction count, clock rate, CPI
Micro-architecture implementation	Clock rate, CPI

Benchmarks

- Performance is 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.
- An alternative to real applications is using a set of benchmarks:
  - Benchmarks are programs specifically chosen to measure performance
  - They are nice for architects and designers
  - Easy to standardize
  - But can be abused!
- Most popular set of benchmarks is the SPEC (System Performance Evaluation Cooperative) suite of benchmarks
  - Companies have agreed in 1989 on a set of real program and inputs
  - Different benchmarks for CPU, I/O, Web, …
  - Valuable indicator of performance (and compiler technology) during the development of new microprocessors.
  - SPEC95, SPEC2000, SPEC2004 …
  - Two types:
    - Integer Benchmarks: SPEC-CINT2006 consists of 12 integer benchmarks
    - Floating point benchmarks: SPEC-CFP2006 consists of 17 FP benchmarks
- Execution times are normalized by dividing the execution time on a Sun Ultra machine with a 300 MHz processor by the execution time on the measured computer
  - SPEC ratio

SPECINTC2006 Benchmarks Running on AMD Opteron X4

- SPEC CPU2006: Elapsed time to execute a selection of programs
  - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios

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

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

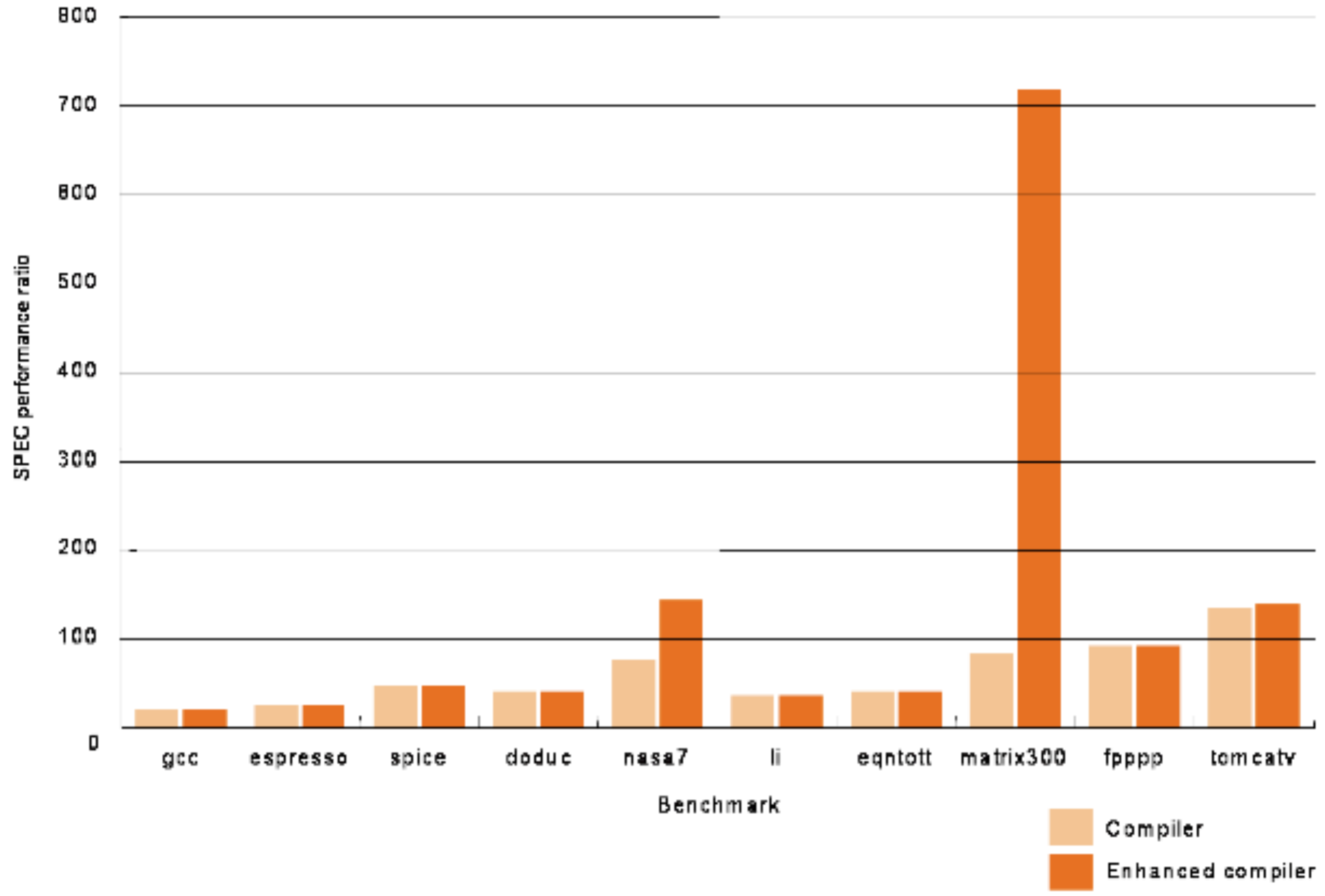
Benchmark Games

- *"An embarrassed Intel Corp. acknowledged Friday that a bug in a software program known as a compiler had led the company to overstate the speed of its microprocessor chips on an industry benchmark by 10 percent. However, industry analysts said the coding error…was a sad commentary on a common industry practice of "cheating" on standardized performance tests…The error was pointed out to Intel two days ago by a competitor, Motorola …came in a test known as SPECint92…Intel acknowledged that it had "optimized" its compiler to improve its test scores. The company had also said that it did not like the practice but felt to compelled to make the optimizations because its competitors were doing the same thing…At the heart of Intel's problem is the practice of "tuning" compiler programs to recognize certain computing problems in the test and then substituting special handwritten pieces of code…"*

  Saturday, January 6, 1996 New York Times

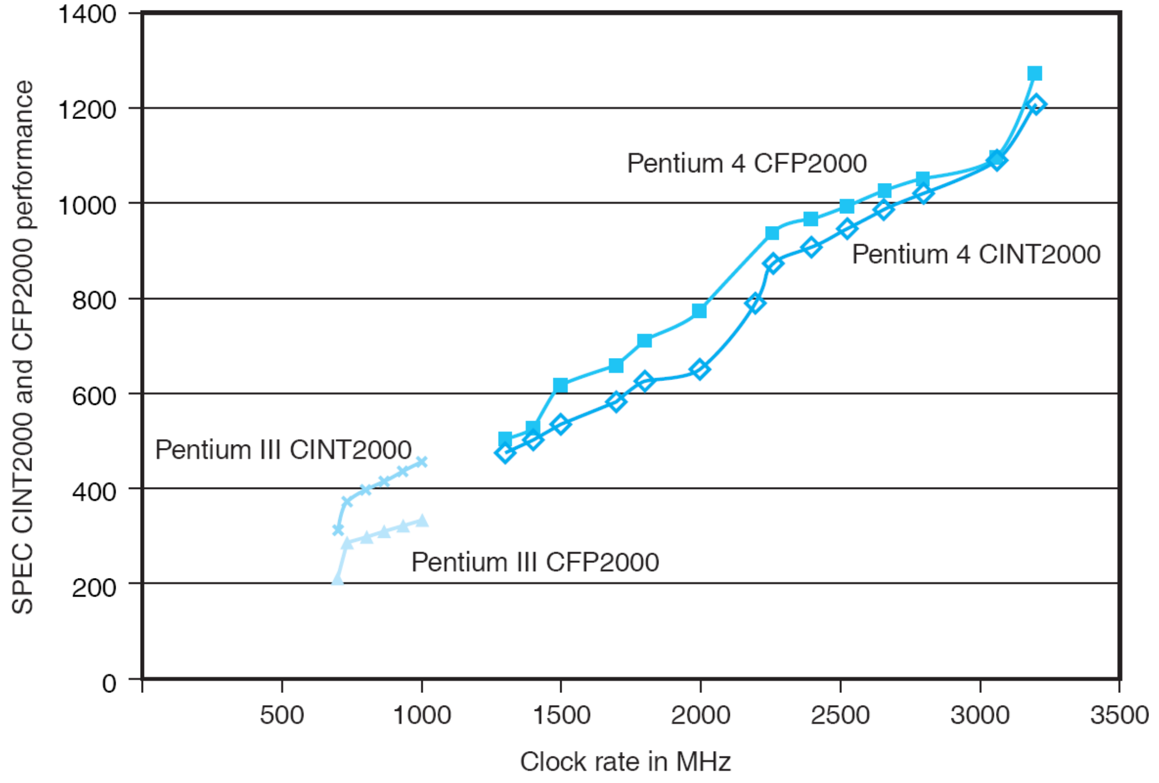
SPEC ‘89

- Compiler “enhancements” and performance



SPEC Ratings for Pentium Processors

- Dell Precision desktop computers



- ❑ Does doubling the clock rate double the performance?
- ❑ Can a machine with a slower clock rate have better performance?