

Example

- Our favorite program runs in 10 seconds on computer A, which has a 400 Mhz. 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?

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

For program A: 10 seconds = $\text{Cycles}_A \times 1/400\text{MHz}$

For program B: 6 seconds = $\text{Cycles}_B \times 1/\text{clock rate}_B$

$\text{Cycles}_B = 1.2 \text{Cycles}_A$

$\text{Clock rate}_B = 800\text{MHz}$

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

Another Way to Compute CPU Time

$$\text{CPU Time (or, Execution Time)} = \frac{\text{\# of instructions}}{\text{program}} \times \frac{\text{\# of cycles}}{\text{instruction}} \times \frac{\text{\# of seconds}}{\text{cycle}}$$

$$= \text{instruction count} \times \text{CPI} \times \text{cycle time}$$

$$= \text{instruction count} \times \text{CPI} \times \frac{1}{\text{clock rate}}$$

Performance

- ❑ Performance is determined by execution time
- ❑ Do any of the following 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.

CPI Example

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

For some program P,

Machine A has a clock cycle time of 10 ns. and a CPI of 2.0

Machine B has a clock cycle time of 20 ns. and a CPI of 1.2

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

$$\text{CPU time}_A = IC \times \text{CPI} \times \text{cycle time} = IC \times 2.0 \times 10\text{ns} = 20 \times IC \text{ ns}$$

$$\text{CPU time}_B = IC \times 1.2 \times 20\text{ns} = 24 \times IC \text{ ns}$$

So, A is 1.2 (=24/20) times faster than B

- 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?

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? (assume CPU starts execute the 2nd instruction after the 1st one completes)
What is the CPI for each sequence?

$$\# \text{ of cycles}_1 = 2 \times 1 + 1 \times 2 + 2 \times 3 = 10$$

$$\# \text{ of cycles}_2 = 4 \times 1 + 1 \times 2 + 1 \times 3 = 9 \quad \text{So, sequence 2 is 1.1 times faster}$$

$$\text{CPI}_1 = 10 / 5 = 2$$

$$\text{CPI}_2 = 9 / 6 = 1.5$$

MIPS Example

- ❑ Two different compilers are being tested for a 100 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 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

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

- ❑ Which sequence will be faster according to MIPS?
- ❑ Which sequence will be faster according to execution time?

of instruction₁ = 5M + 1M + 1M = 7M, # of instruction₂ = 10M + 1M + 1M = 12M

of cycles₁ = 5M × 1 + 1M × 2 + 1M × 3 = 10M cycles = 0.1 seconds

of cycles₂ = 10M × 1 + 1M × 2 + 1M × 3 = 15M cycles = 0.15 seconds

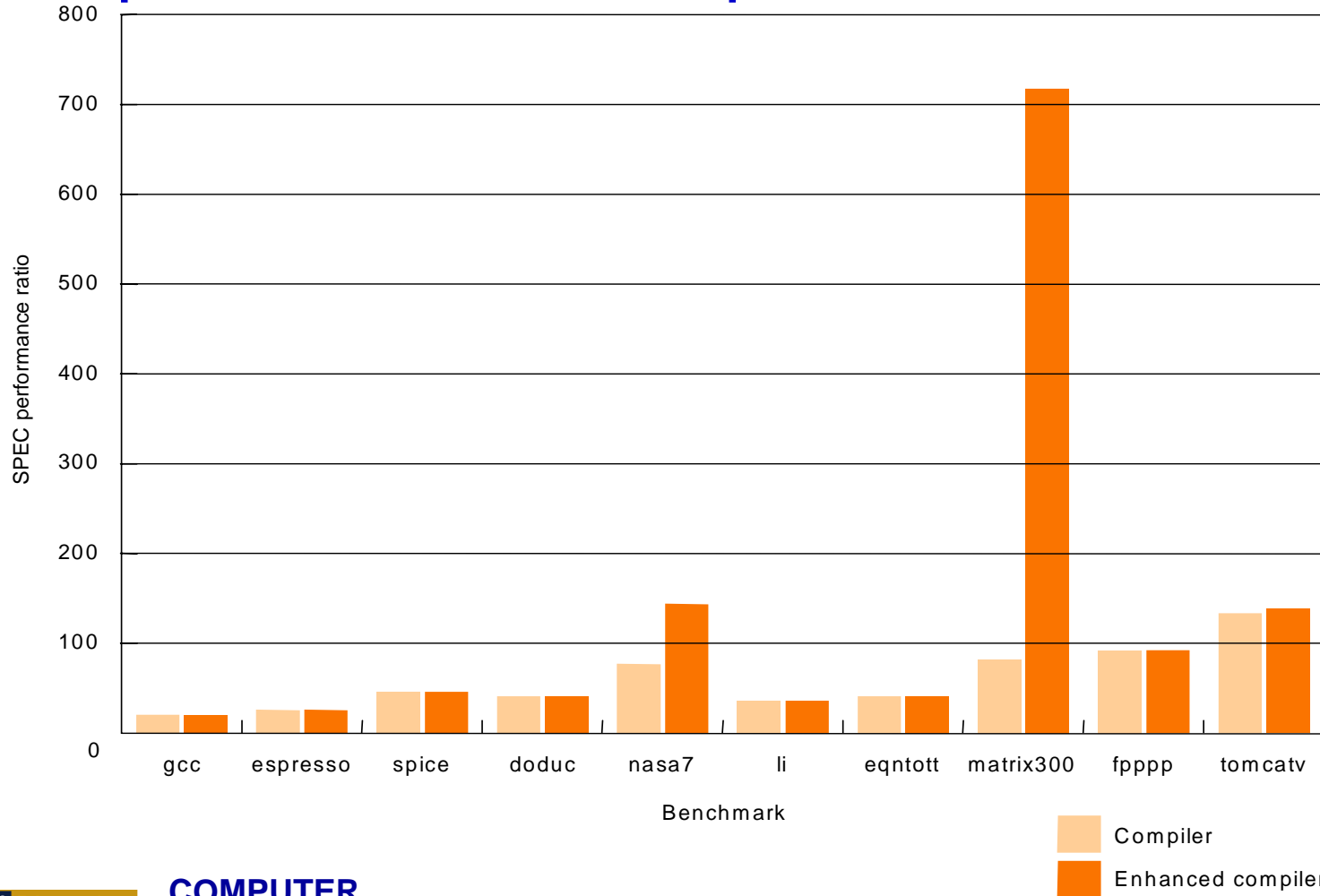
So, MIPS₁ = 7M/0.1 = 70MIPS, MIPS₂ = 12M/0.15 = 80MIPS > MIPS₁

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)
 - can still be abused

SPEC '89

□ Compiler “enhancements” and performance



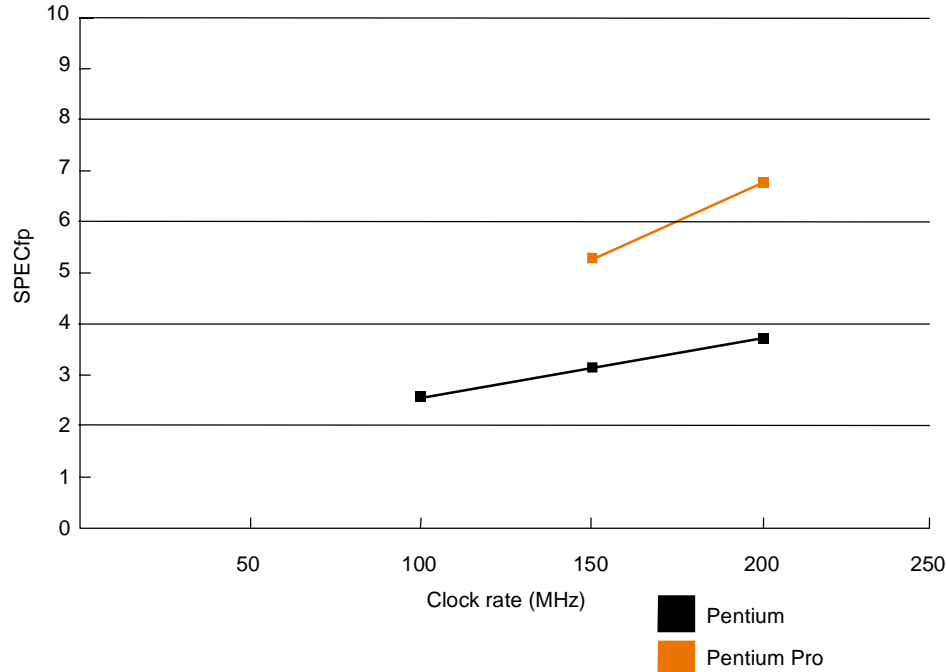
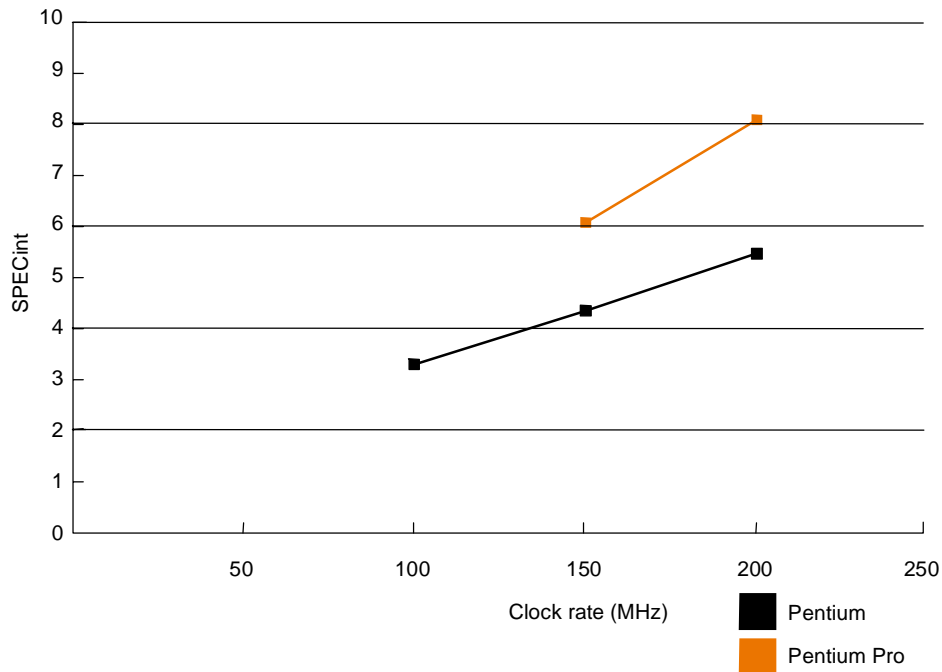
SPEC '95

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
jpeg	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 Navier 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?



Amdahl's Law

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



Execution time w/o E (Before)

Speedup (E) =

Execution time w E (After)

❑ **Example:**

"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?"

How about making it 5 times faster?

❑ *Principle: Make the common case fast*

Example

- Suppose we enhance a machine making all floating-point instructions run five times faster. If the execution time of some benchmark before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?

$$10/6$$

- We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware. How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?

$$100-x+x/5 = 100/3, \quad x=83.3$$

Remember

- ❑ **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

- ❑ **Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance**