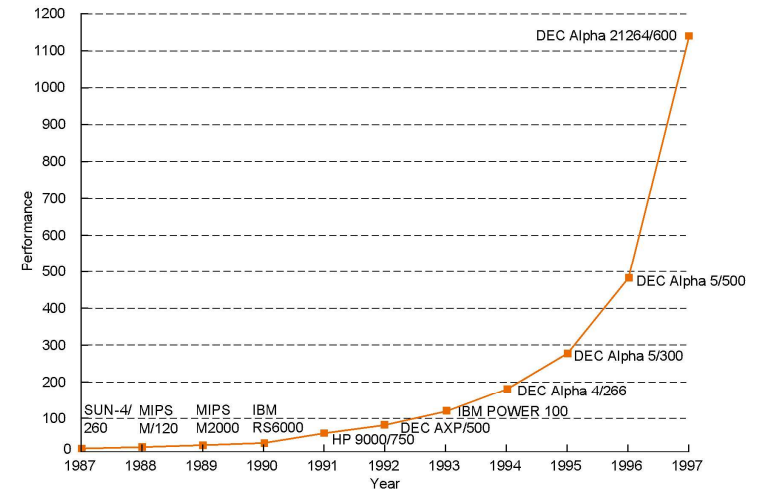


Measuring and Evaluating Computer System Performance

CSE 141, S2'06

Jeff Brown

Performance Marches On ...



- But *what is performance?*

CSE 141, S2'06

Jeff Brown

The bottom line: Performance

Car	Time to Bay Area	Speed	Passengers	Throughput (pmph)
Ferrari	3.1 hours	160 mph	2	320
Greyhound	7.7 hours	65 mph	60	3900

- **Time to do the task**
 - *execution time*, response time, latency
- **Tasks per day, hour, week, sec, ns. ..**
 - *throughput*, bandwidth

CSE 141, S2'06

Jeff Brown

How to measure Execution Time?

```
% time program
... program results ...
90.7u 12.9s 2:39 65%
%
```

- Wall-clock time?
- user CPU time?
- user + kernel CPU time?
- Answer:

CSE 141, S2'06

Jeff Brown

Our definition of Performance

$$\text{Performance}_x = \frac{1}{\text{Execution Time}_x}, \text{ for program } X$$

- only has meaning in the context of a **program** or **workload**
- Not very intuitive as an absolute measure, but most of the time we're more interested in **relative performance**.

CSE 141, S2'06

Jeff Brown

Relative Performance

- can be confusing
 - A runs in 12 seconds
 - B runs in 20 seconds
 - $A/B = .6$, so A is 40% faster, or 1.4X faster, or B is 40% slower
 - $B/A = 1.67$, so A is 67% faster, or 1.67X faster, or B is 67% slower
- needs a precise definition

CSE 141, S2'06

Jeff Brown

Relative Performance, the Definition

$$\text{Relative Performance}^{(X/Y)} = \frac{\text{Performance}_x}{\text{Performance}_y} = \frac{\text{Execution Time}_y}{\text{Execution Time}_x} = n$$

"X is **n** times faster than Y"
"X is **n** times as fast as Y"
"From Y to X, speedup is **n**"

CSE 141, S2'06

Jeff Brown

Example

- Machine A runs program C in 9 seconds, Machine B runs the same program in 6 seconds. What is the speedup we see if we move to Machine B from Machine A?
- Machine B gets a new compiler, and can now run the program in 3 seconds. ???

CSE 141, S2'06

Jeff Brown

What is Time?

CPU Execution Time = CPU clock cycles * Clock cycle time

- Every conventional processor has a clock with an associated clock cycle time or clock rate
- Every program runs in an integral number of clock cycles

Cycle Time

MHz = millions of cycles/second, GHz = billions of cycles/second

X MHz = 1000/X nanoseconds cycle time

Y GHz = 1/Y nanoseconds cycle time

How many clock cycles?

Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)

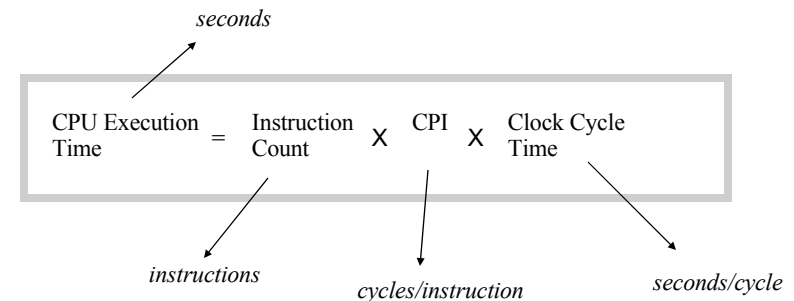
Computer A runs program C in 3.6 billion cycles. Program C consists of 2 billion dynamic instructions. What is the CPI?

How many clock cycles?

Number of CPU cycles = Instructions executed * Average Clock Cycles per Instruction (CPI)

A computer is running a program with CPI = 2.0, and executes 24 million instructions, how long will it run?

All Together Now



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

- IC = 1 billion, 500 MHz processor, execution time of 3 seconds. What is the CPI for this program?
- Suppose we reduce CPI to 1.2 (through an architectural improvement). What is the new execution time?

CSE 141, S2'06

Jeff Brown

Who Affects Performance?

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

- programmer
- compiler
- instruction-set architect
- machine architect
- hardware designer
- materials scientist/physicist/silicon engineer

CSE 141, S2'06

Jeff Brown

Performance Variation

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

	Number of instructions	CPI	Clock Cycle Time
Same machine different programs			
same programs, different machines, same ISA			
Same programs, different machines			

CSE 141, S2'06

Jeff Brown

Other Performance Metrics

- MIPS
- MFLOPS

CSE 141, S2'06

Jeff Brown

MIPS

MIPS = Millions of Instructions Per Second

$$= \frac{\text{Instruction Count}}{\text{Execution Time} * 10^6}$$

$$= \frac{\text{Clock rate}}{\text{CPI} * 10^6}$$

- Program-independent?
- Deceptive

CSE 141, S2'06

Jeff Brown

FLOPS

FLOPS = Floating-point Operations Per Second

- Program-independent?
 - Which operations?
- Useful, sometimes
 - "Theoretical peak" FLOPS, peak FLOPS, sustained FLOPS
- How does execution time depend on FLOPS?

CSE 141, S2'06

Jeff Brown

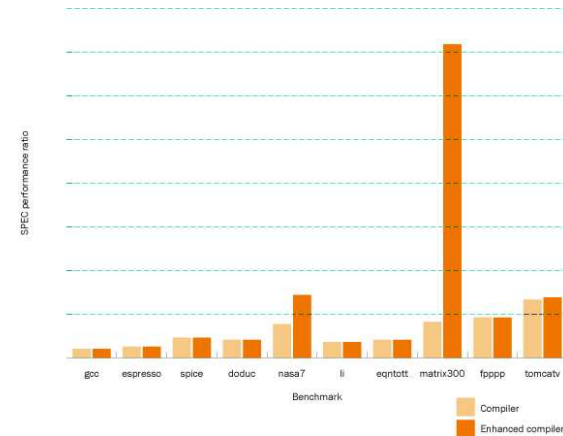
Which Programs?

- peak throughput measures (simple programs)?
- synthetic benchmarks (whetstone, dhrystone,...)?
- "kernels" of useful computation (lapack, fftw, ...)
- Real applications
- SPEC (best of both worlds, but with problems of their own)
 - System Performance Evaluation Cooperative
 - Provides a common set of real applications along with strict guidelines for how to run them.
 - provides a relatively unbiased means to compare machines.

CSE 141, S2'06

Jeff Brown

Danger in Benchmark-Specific Performance Measures



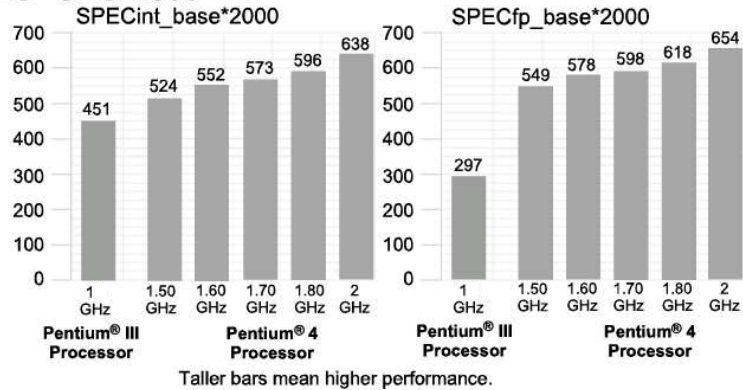
- measures **compiler** as much as **architecture**
 - (what about kernels?)

CSE 141, S2'06

Jeff Brown

SPEC Performance on Pentium III and Pentium 4

SPEC* CPU*2000



CSE 141, S2'06

Jeff Brown

Amdahl's Law

- The impact of a performance improvement is limited by the percent of execution time affected by the improvement

$$\text{Execution time after improvement} = \frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} + \text{Execution Time Unaffected}$$

- Make the **common case** fast!!

CSE 141, S2'06

Jeff Brown

Key Points

- Be careful how you specify performance
- Execution time = instructions * CPI * cycle time
- Use real applications
- Use standards, if possible
- Make the common case fast**

CSE 141, S2'06

Jeff Brown