
Understanding Performance

[Adapted from Mary Jane Irwin for
Computer Organization and Design,
Patterson & Hennessy, © 2005, UCB]

Indeed, the cost-performance ratio of the product will depend most heavily on the implementer, just as ease of use depends most heavily on the architect.

The Mythical Man-Month, Brooks, pg 46

Performance Metrics

❑ Purchasing perspective

- given a collection of machines, which has the
 - best performance ?
 - least cost ?
 - best cost/performance?

❑ Design perspective

- faced with design options, which has the
 - best performance improvement ?
 - least cost ?
 - best cost/performance?

❑ Both require

- basis for comparison
- metric for evaluation

❑ Our goal is to understand what factors in the architecture contribute to overall system performance and the relative importance (and cost) of these factors

Defining (Speed) Performance

□ Normally interested in reducing

- **Response time** (aka execution time) – the time between the start and the completion of a task
 - Important to individual users
- Thus, to maximize performance, need to **minimize** execution time

$$\text{performance}_X = 1 / \text{execution_time}_X$$

If X is n times faster than Y, then

$$\frac{\text{performance}_X}{\text{performance}_Y} = \frac{\text{execution_time}_Y}{\text{execution_time}_X} = n$$

- Throughput – the total amount of work done in a given time
 - Important to data center managers
- Decreasing response time almost always improves throughput

Performance Factors

- ❑ Want to distinguish elapsed time and the time spent on our task
- ❑ CPU execution time (CPU time) – time the CPU spends working on a task
 - Does not include time waiting for I/O or running other programs

$$\begin{array}{l} \text{CPU execution time} \\ \text{for a program} \end{array} = \begin{array}{l} \# \text{ CPU clock cycles} \\ \text{for a program} \end{array} \times \text{clock cycle time}$$

or

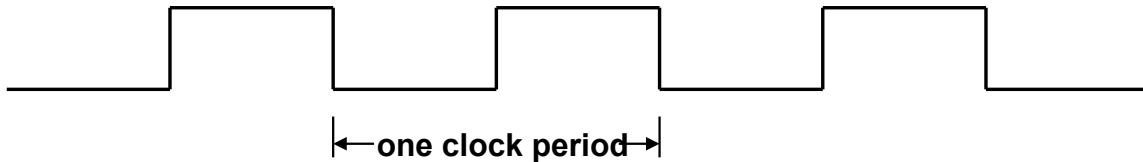
$$\begin{array}{l} \text{CPU execution time} \\ \text{for a program} \end{array} = \frac{\# \text{ CPU clock cycles for a program}}{\text{clock rate}}$$

- ❑ Can improve performance by reducing either the **length of the clock cycle** or the **number of clock cycles required for a program**

Review: Machine Clock Rate

- ❑ Clock rate (MHz, GHz) is inverse of clock cycle time (clock period)

$$CC = 1 / CR$$



10 nsec clock cycle => 100 MHz clock rate

5 nsec clock cycle => 200 MHz clock rate

2 nsec clock cycle => 500 MHz clock rate

1 nsec clock cycle => 1 GHz clock rate

500 psec clock cycle => 2 GHz clock rate

250 psec clock cycle => 4 GHz clock rate

200 psec clock cycle => 5 GHz clock rate

Clock Cycles per Instruction

- ❑ Not all instructions take the same amount of time to execute
 - One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction

$$\begin{array}{l} \# \text{ CPU clock cycles} \\ \text{for a program} \end{array} = \begin{array}{l} \# \text{ Instructions} \\ \text{for a program} \end{array} \times \begin{array}{l} \text{Average clock cycles} \\ \text{per instruction} \end{array}$$

- ❑ **Clock cycles per instruction (CPI)** – the average number of clock cycles each instruction takes to execute
 - A way to compare two different implementations of the same ISA

	CPI for this instruction class		
	A	B	C
CPI	1	2	3

Effective CPI

example

- ❑ Computing the overall effective CPI is done by looking at the different types of instructions and their individual cycle counts and averaging

$$\text{Overall effective CPI} = \sum_{i=1}^n (\text{CPI}_i \times \text{IC}_i)$$

- Where IC_i is the count (percentage) of the number of instructions of class i executed
 - CPI_i is the (average) number of clock cycles per instruction for that instruction class
 - n is the number of instruction classes
- ❑ The overall effective CPI varies by instruction mix – a measure of the dynamic frequency of instructions across one or many programs

THE Performance Equation

- Our basic performance equation is then

$$\text{CPU time} = \text{Instruction_count} \times \text{CPI} \times \text{clock_cycle}$$

or

$$\text{CPU time} = \frac{\text{Instruction_count} \times \text{CPI}}{\text{clock_rate}}$$

- These equations separate the **three key** factors that affect performance
 - Can measure the CPU execution time by running the program
 - The clock rate is usually given
 - Can measure overall instruction count by using profilers/simulators without knowing all of the implementation details
 - CPI varies by instruction type and ISA implementation for which we must know the implementation details

Determinates of CPU Performance

CPU time = Instruction_count x CPI x clock_cycle

	Instruction_count	CPI	clock_cycle
Algorithm	X	X	
Programming language	X	X	
Compiler	X	X	
ISA	X	X	X
Processor organization		X	X
Technology			X

A Simple Example

Op	Freq	CPI _i	Freq x CPI _i
ALU	50%	1	.5
Load	20%	5	1.0
Store	10%	3	.3
Branch	20%	2	.4
$\Sigma =$			2.2

.5 .5 .25
.4 1.0 1.0
.3 .3 .3
.4 .2 .4
1.6 2.0 1.95

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?

CPU time new = 1.6 x IC x CC so 2.2/1.6 means 37.5% faster

- How does this compare with using branch prediction to shave a cycle off the branch time?

CPU time new = 2.0 x IC x CC so 2.2/2.0 means 10% faster

- What if two ALU instructions could be executed at once?

CPU time new = 1.95 x IC x CC so 2.2/1.95 means 12.8% faster

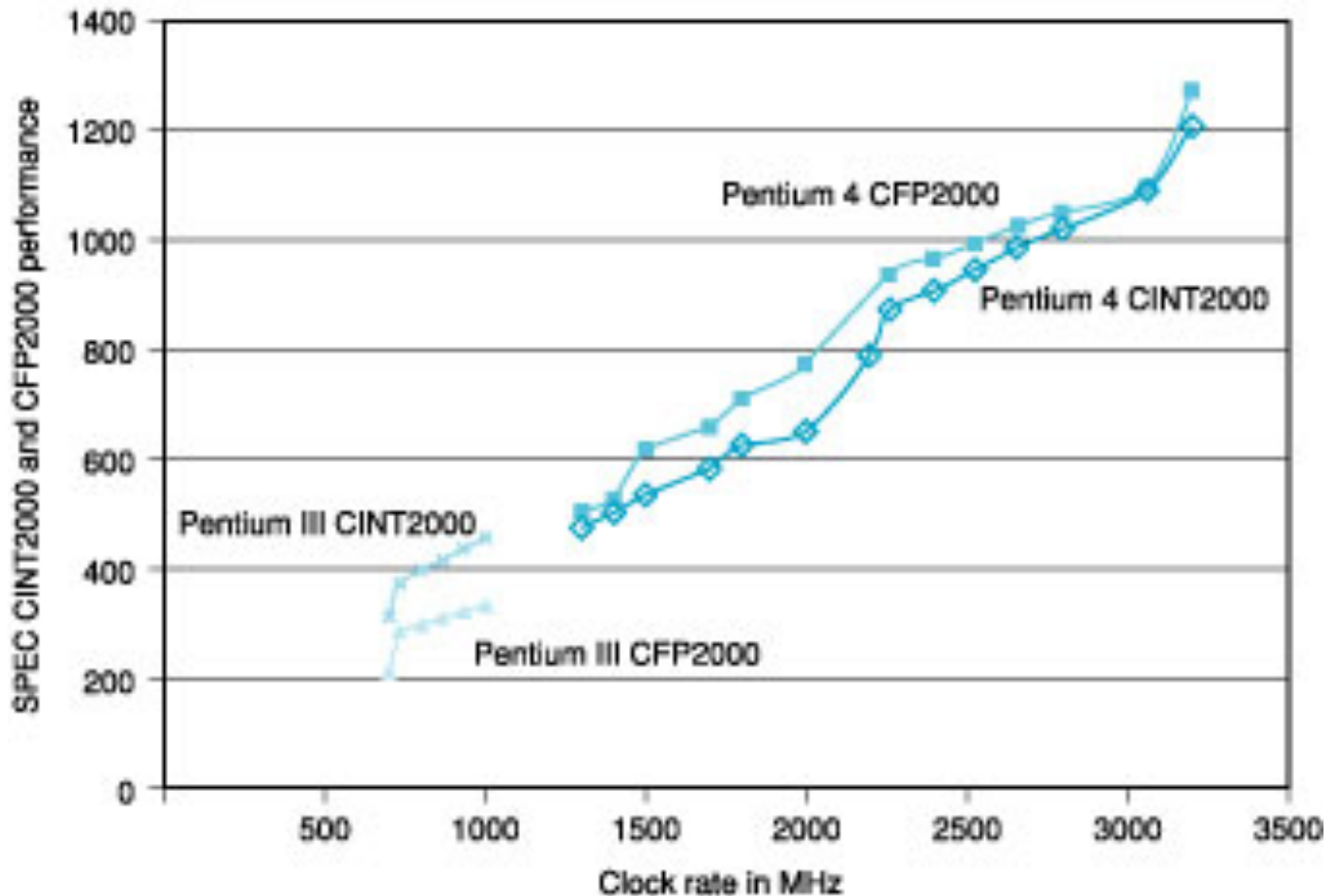
Comparing and Summarizing Performance

- ❑ How do we summarize the performance for benchmark set with a **single** number?
 - The average of execution times that is directly proportional to total execution time is the **arithmetic mean** (AM)
$$AM = \frac{1}{n} \sum_{i=1}^n \text{Time}_i$$
 - Where Time_i is the execution time for the i^{th} program of a total of n programs in the workload
 - A smaller mean indicates a smaller average execution time and thus improved performance
- ❑ Guiding principle in reporting performance measurements is **reproducibility** – list everything another experimenter would need to duplicate the experiment (version of the operating system, compiler settings, input set used, specific computer configuration (clock rate, cache sizes and speed, memory size and speed, etc.))

SPEC Benchmarks www.spec.org

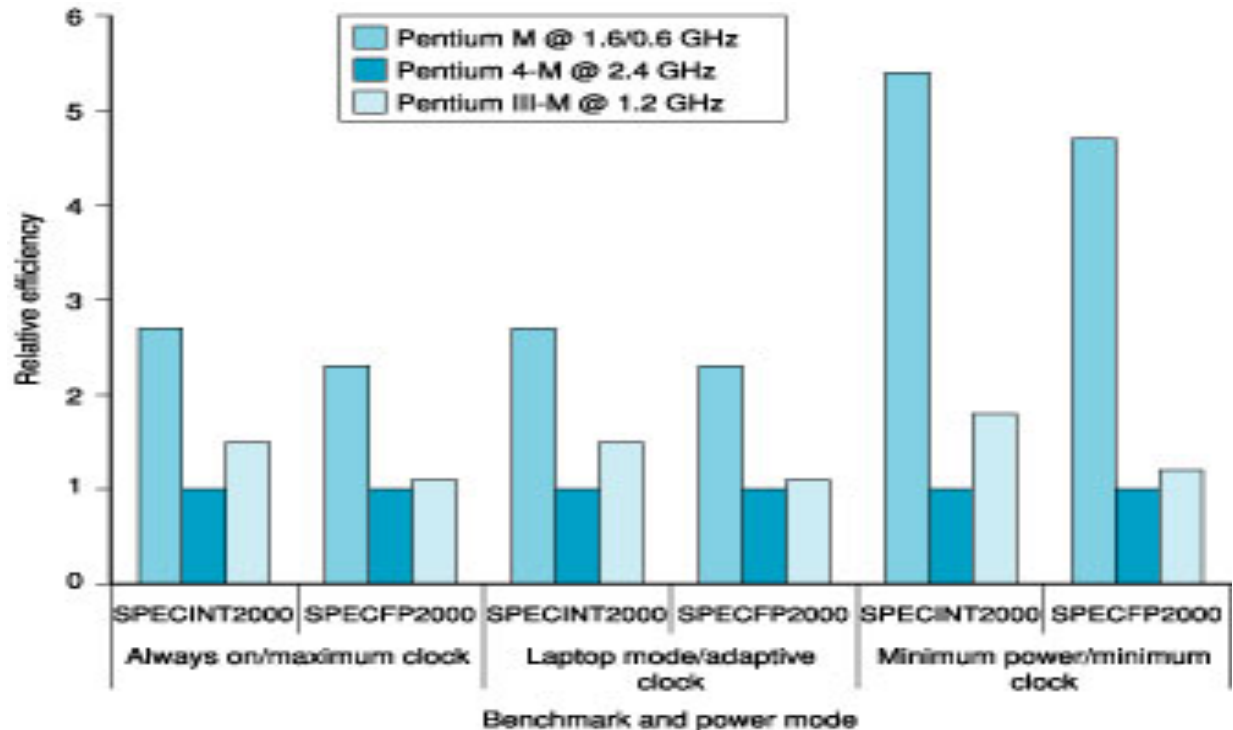
Integer benchmarks		FP benchmarks	
gzip	compression	wupwise	Quantum chromodynamics
vpr	FPGA place & route	swim	Shallow water model
gcc	GNU C compiler	mgrid	Multigrid solver in 3D fields
mcf	Combinatorial optimization	applu	Parabolic/elliptic pde
crafty	Chess program	mesa	3D graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Image recognition (NN)
perlbnk	perl application	equake	Seismic wave propagation simulation
gap	Group theory interpreter	facerec	Facial image recognition
vortex	Object oriented database	ammp	Computational chemistry
bzip2	compression	lucas	Primality testing
twolf	Circuit place & route	fma3d	Crash simulation fem
		sixtrack	Nuclear physics accel
		apsi	Pollutant distribution

Example SPEC Ratings



Other Performance Metrics

- ❑ Power consumption – especially in the embedded market where battery life is important (and passive cooling)
 - For power-limited applications, the most important metric is energy efficiency



Summary: Evaluating ISAs

❑ Design-time metrics:

- Can it be implemented, in how long, at what cost?
- Can it be programmed? Ease of compilation?

❑ Static Metrics:

- How many bytes does the program occupy in memory?

❑ Dynamic Metrics:

- How many instructions are executed? How many bytes does the processor fetch to execute the program?
- How many clocks are required per instruction?
- How "lean" a clock is practical?

Best Metric: Time to execute the program!

depends on the instructions set, the processor organization, and compilation techniques.

