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

Are not the architects a new aristocracy, an intellectual elite, set up to tell the poor dumb implementers what to do?

No, because the setting of external specifications is not more creative work than the designing of implementations. It is just different creative work. The design of an implementation, given an architecture, requires and allows as much design creativity, as many new ideas, and as much technical brilliance as the design of the external specifications.

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

Or smallest/lightest

Longest battery life

Most reliable/durable (in space)

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

Increasing performance requires decreasing execution time

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

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

or

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

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

Computers are constructed with a clock that determines when events take place in the hardware

Many techniques that decrease the number of clock cycles also increase the clock cycle time

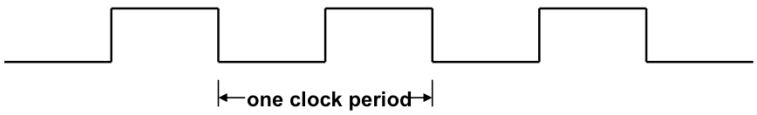
Clock cycle = time it takes to do one cycle

Clock rate = number of cycles during one second

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

A clock cycle is the basic unit of time to execute one operation/pipeline stage/etc.

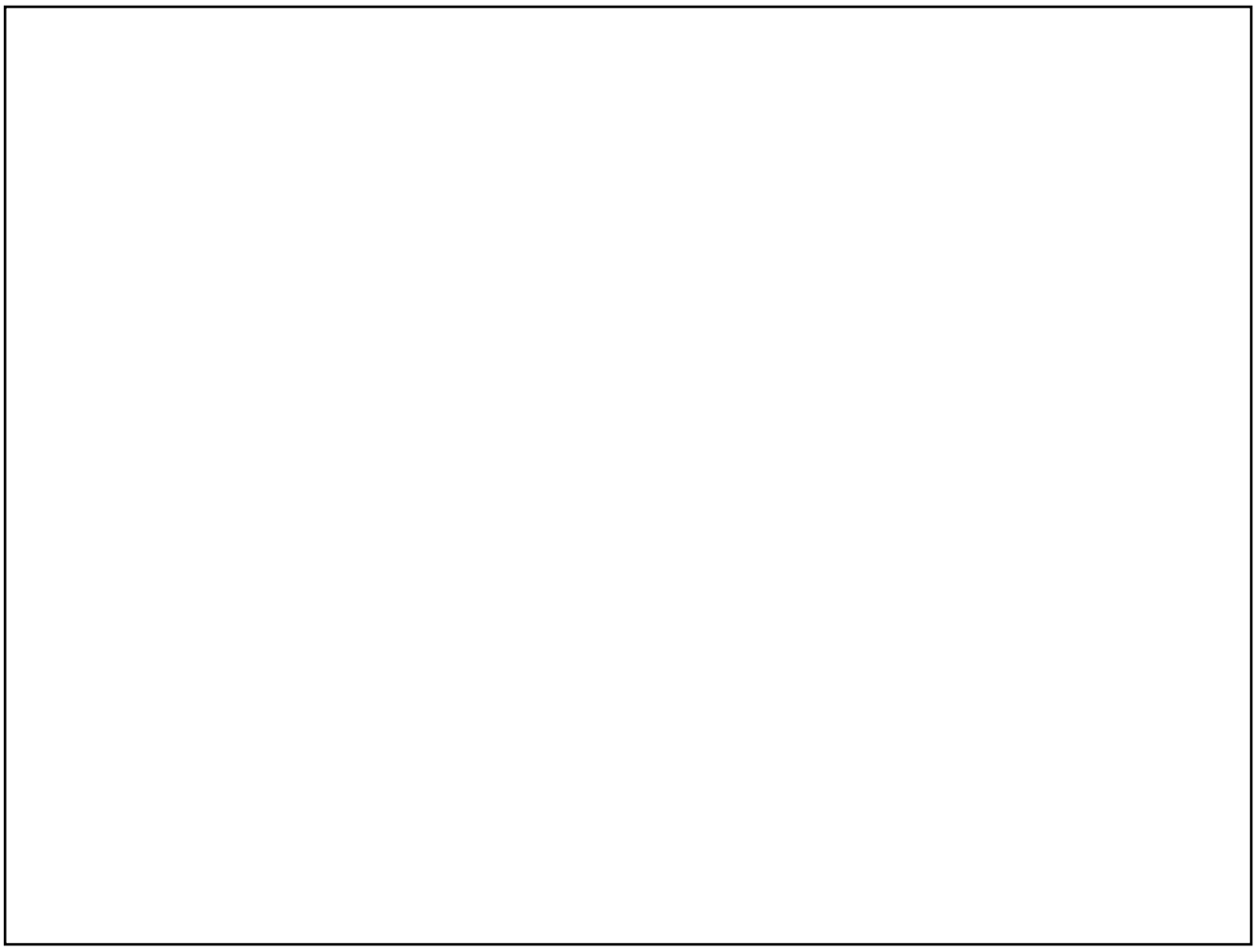
Clock Cycles per Instruction

- 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

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

- 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



Example pg 34 (sec 1.4)

- Same program
  - Computer A CC 250ps and CPI 2.0
  - Computer B CC 500ps and CPI 1.2
- Which computer is faster and by how much?
- I is the number of instructions
- CPU clock cycles_A = I x CPI_A = I x 2.0
- CPU clock cycles_B = I x CPI_B = I x 1.2
- CPU_time_A = CPU_clock_cycle_A x Clock_cycle_time = I x 2.0 x 250ps = 500 x I ps
- CPU_time_B = CPU_clock_cycle_B x Clock_cycle_time = I x 1.2 x 500ps = 600 x I ps
- CPUperf_A / CPUperf_B = 600 / 500 = 1.2

Page 34 of the Patterson Hennessy book (English edition) sec 1.4

Computer A is 1.2 times faster than Computer B

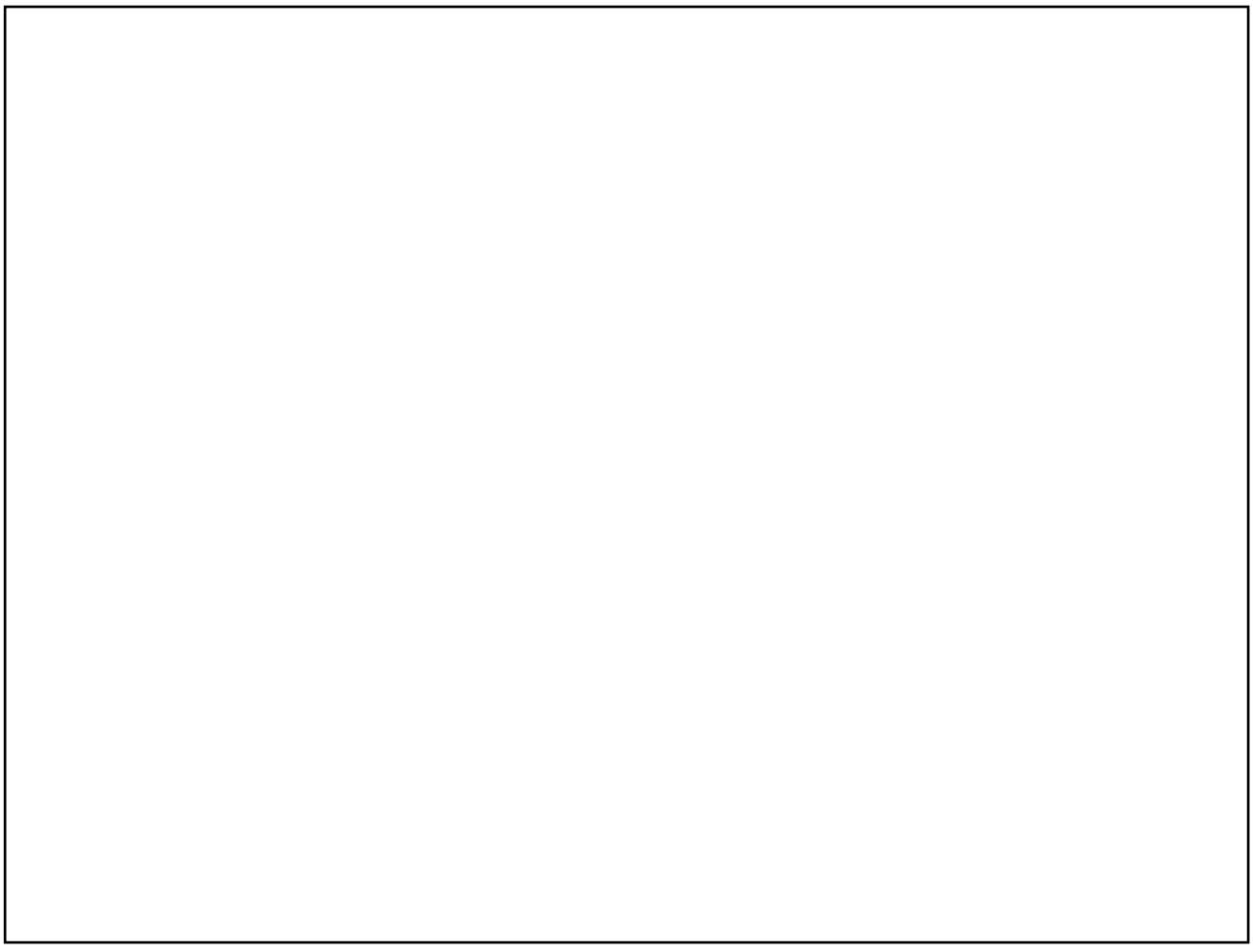
Effective CPI

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

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

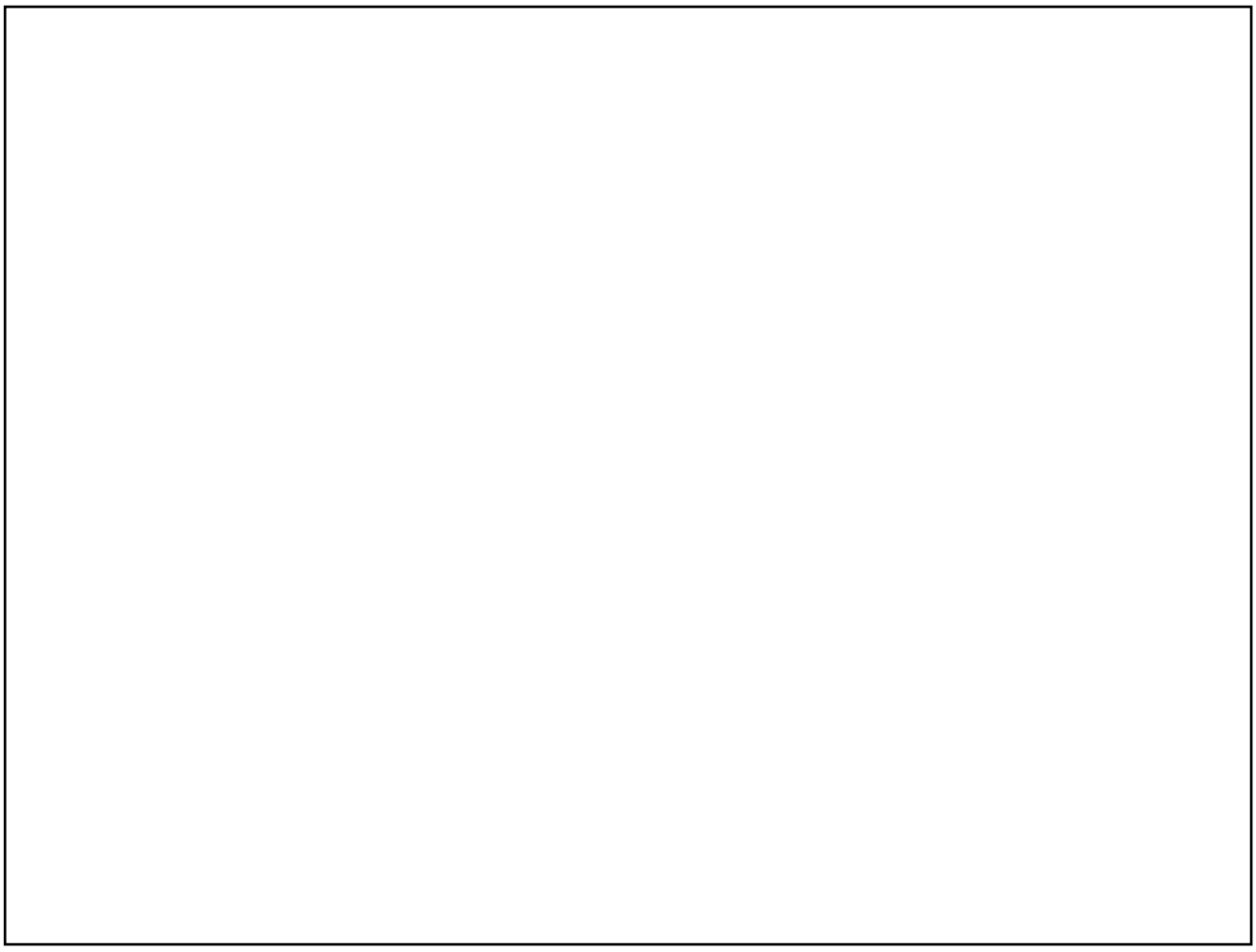
  - Where IC_i is the count (percentage) of the number of instructions of class i executed
  - CPI_i is the (average) number of clock cycles per instruction for that instruction class
  - n is the number of instruction classes

- The overall effective CPI varies by instruction mix – a measure of the dynamic frequency of instructions across one or many programs



Example pg 35 (sec 1.4)

- A(1) B(2) C(3) cycles per instruction type
- Code 1: A(2) B(1) C(2)
- Code 2: A(4) B(1) C(1)
- Which code sequence executes the most instructions?
- Which will be faster?
- What is the CPI for each sequence?
- Code 1: 2+1+2 = 5
- Code 2: 4+1+1 = 6
- CPU clock cycles = sum_i=A,B,C (CPI_i x C_i)
- Code 1: 2x1 + 1x2 + 2x3 = 10 cycles
- Code 2: 4x1 + 1x2 + 1x3 = 9 cycles
- CPI_1 = CPU clock cycles / Instruction count = 10/5 = 2
- CPI_2 = CPU clock cycles / Instruction count = 9/6 = 1.5



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

Note that instruction count is dynamic – i.e., its not the number of lines in the code, but THE NUMBER OF INSTRUCTIONS EXECUTED

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

For lecture

A Simple Example

Op	Freq	CPI_i	Freq x CPI_i			
ALU	50%	1	.5	.5	.5	.25
Load	20%	5	1.0	.4	1.0	1.0
Store	10%	3	.3	.3	.3	.3
Branch	20%	2	.4	.4	.2	.4
			Σ = 2.2	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

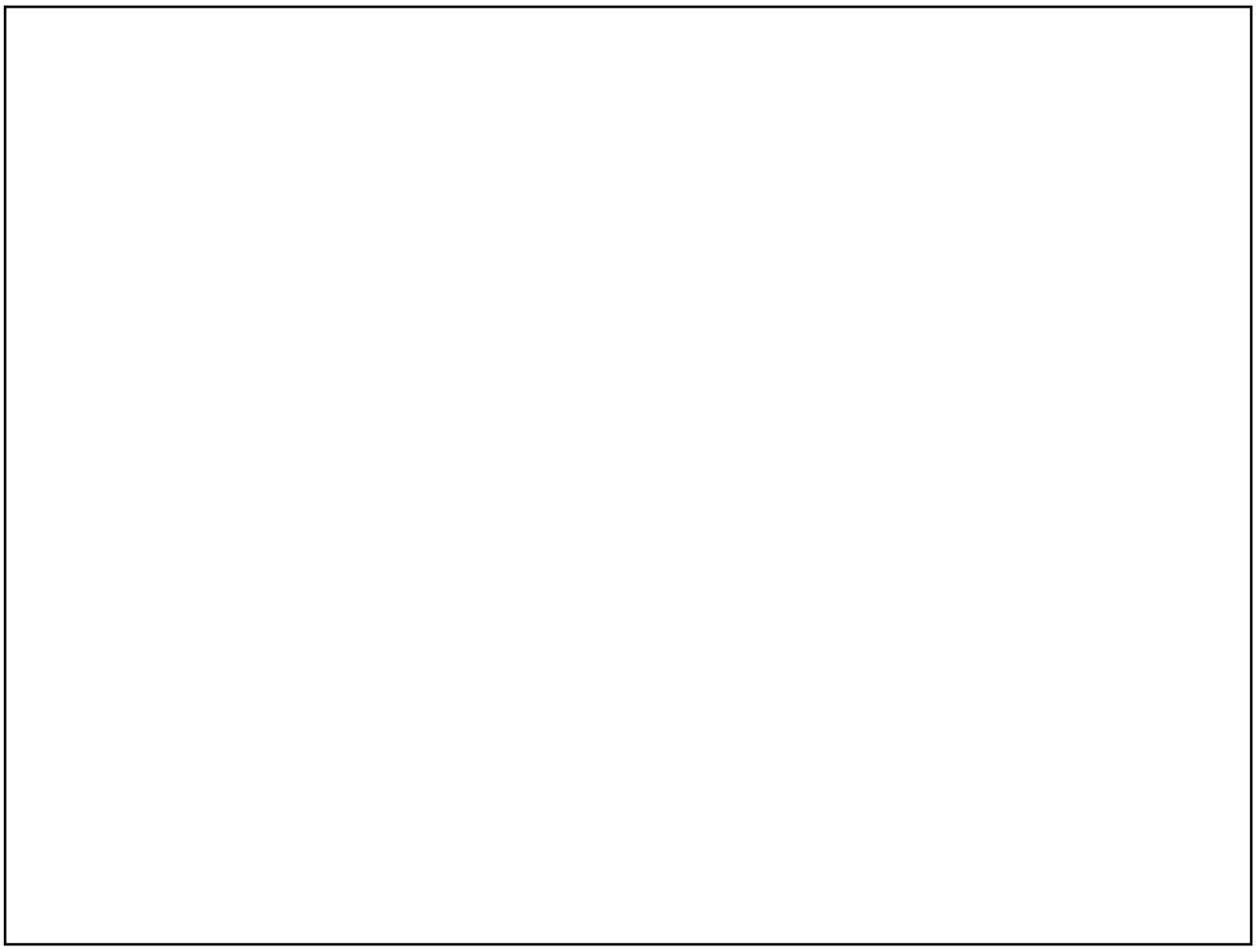
For lecture

Comparing and Summarizing Performance

- How do we summarize the performance on a benchmark set with a single number?
  - The average of execution times that is directly proportional to total execution time is the arithmetic mean (AM)

$$AM = 1/n \sum_{i=1}^{n} Time_i$$

  - 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.))

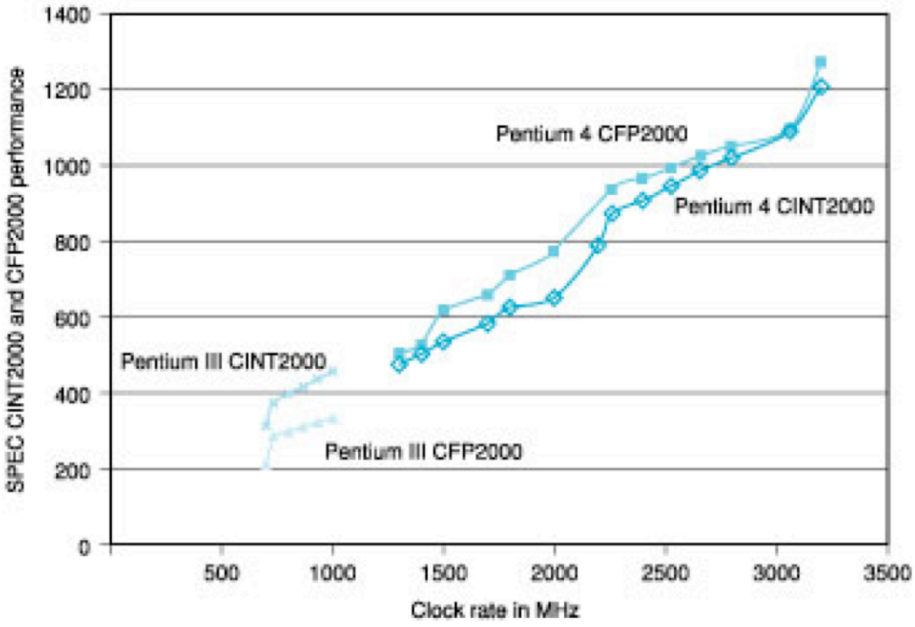


SPEC Benchmarks www.spec.org

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)
perlbmk	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

System Performance Evaluation Cooperative

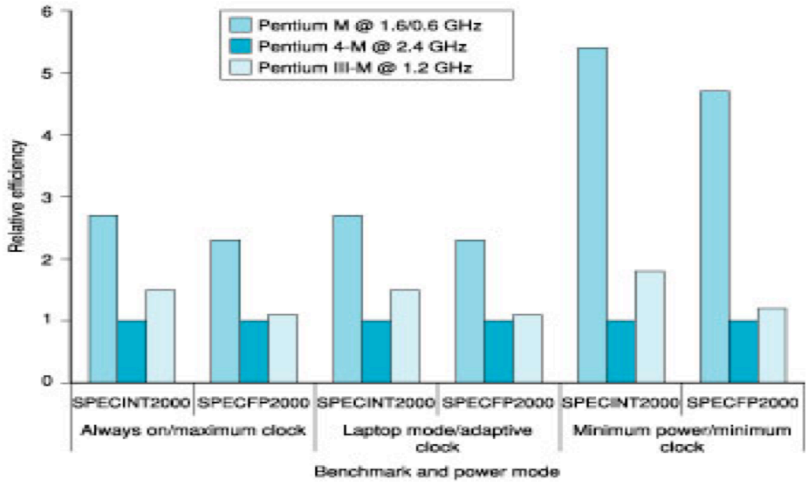
Example SPEC Ratings



P4 ist bei Floating Point besser.
Siehe PH Seite 217

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

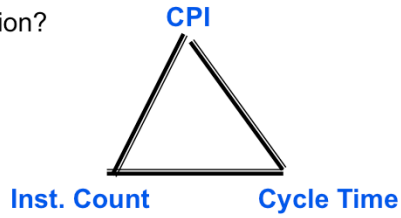


Summary: Evaluating ISAs

- 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.



A lean clock meant as a skewed clock where we overclock it to increase performance (a common practice in the 90s)