EECE 417 Computer Systems Architecture

Department of Electrical and Computer Engineering

Howard University

Charles Kim

Spring 2007

Computer Organization and Design (3rd Ed)

-The Hardware/Software Interface

by

David A. Patterson
John L. Hennessy

Chapter 4 - Part A

Assessing and Understanding Performance

Performance

- Measure, Report, and Summarize
- What determines the performance of a comp
  - Is hardware performance the key?
  - What is the role of software in performance?
  - one tough job

- Make intelligent choices
- See through the marketing hype
- Questions:

Defining Performance

- **Which of these airplanes has the best performance?**

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concorde	132	4000	1350
Douglas DC-8-50	146	8720	544

- **In Speed?**
- **In Cruising Range?**
- **In Passenger Capacity?**

Computer Performance: TIME, TIME, TIME

- Response Time (latency, "execution time")

  The time between the start and completion of a task

  — 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?

- Throughput

  The total amount of work done in a given time.

  — 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 faster processor what do we increase?
  - **Response Time, Throughput, or both?**
- If we add a new machine to the lab what do we increase?
  - **Response Time, Throughput, or both?**

Execution Time

- **Elapsed Time (Wall-clock time, response time)**
  - **counts everything *(disk and memory accesses, I/O , etc.)***
  - **a useful number, but often not good for comparison purposes**
- **CPU (execution) time**
  - **doesn't count I/O or time spent running other programs**
  - **can be broken up into system CPU time, and user CPU time (difficult, though)**
  - **User CPU time**
    - time spent executing the lines of code that are "in" our program
  - **System CPU time**
    - time spent in the operating system performing tasks on be half of the program

Execution Time & Performance

- For some program running on machine X,

$$\text{Performance}_X = \frac{1}{\text{Execution Time}_X}$$

- "X is n times faster than Y"

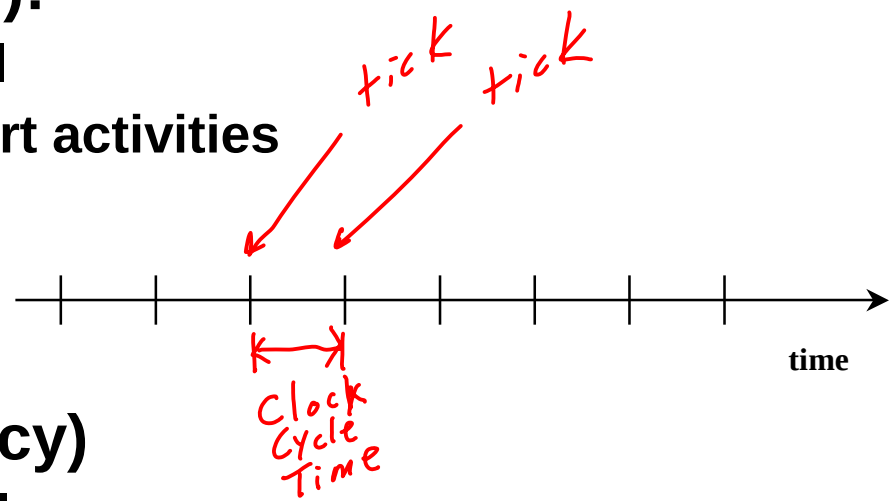
$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n$$

- Example:
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds
  - “A is 1.5 times faster than B”

$$\frac{\text{Performance}_A}{\text{Performance}_B} = 1.5$$

CPU Performance - clock

- **Clock Cycle** (“ticks”):
  - discrete time interval
  - indicate when to start activities
- **Clock Cycle Time**
  - time between ticks
  - Time of a cycle
- **Clock Rate (frequency)**
  - Inverse of clock cycle
  - cycles per second (1 Hz. = 1 cycle/sec)



A 4 Ghz. clock has a clock cycle time

$$\frac{1}{4\ GHz} = \frac{1}{4 \times 10^{9}} = 2.5 \times 10^{-10} = 250 \times 10^{-12}$$

$$= 250\ ps$$

CPU Performance

- Simple formula for CPU performance:

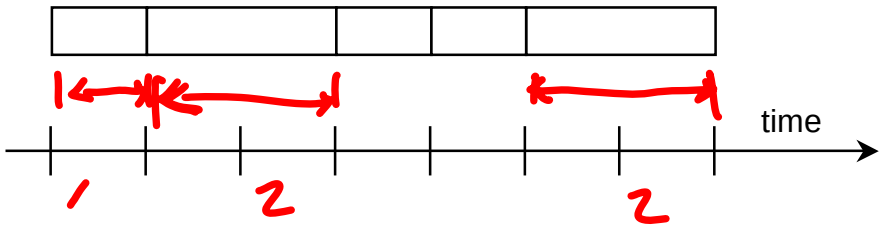
$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock Cycle time}$$

- Alternative formula:

$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock Rate}}$$

- CPU performance comes from:
  - reduced length of clock cycle
  - reduced number of clock cycles required for a program

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**

Performance Improvement - Example

- Our favorite program runs in 10 seconds on computer A, which has a 4 GHz 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?"

CPI – Clock Cycles per Instruction

- **Execution time & number of instructions**
  - **They are related**
  - **Execution time depends of the number of instructions**
  - **Each instruction requires different number of clock cycles**

(Execution Time) = (Number of Instructions) × (Average Time Per Instruction)

- **CPI (Clock cycles per Instruction)**
  - **"Average number of clock cycles each instruction takes to execute"**

(CPU clock cycle) = (Number of Instr.) × (CPI)

(Execution Time) = (Number of Instr) × CPI × Cycle Time

CPI Example

- **Suppose we have two implementations**
  For some program,

  Machine A has a clock cycle time of 250 ps and a CPI of 2.0
  Machine B has a clock cycle time of 500 ps and a CPI of 1.2

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

CPI Example

- **CPI is not the same**
  - **Memory system dependent**
  - **Instruction type dependent**
  - **Application dependent**

$$\text{CPU clock cycles} = \sum_{i=1}^{n} [CPI_i \times C_i]$$

C_i: Count of # of instructions of class i executed.

CPI_i: Average number of cycles per instruction for the class i

CPI Example for Code Segments

- **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 CPIs (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?

Another Example surrounding CPI – p.253

- **A given application in Java runs 15 seconds on a desktop processor. A new Java complier is released that requires only 0.6 as many instructions as the old compiler. Unfortunately it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler?**

Benchmarks

- **Ideal Condition of Performance Evaluation**
  - Same workload
  - different computer systems
  - Compare the execution time
- **Alternative Approach**
  - Evaluation of computer systems using a set of benchmarks
  - “Benchmark”
    - **program specifically chosen to measure performance.**
    - **forms a workload for the prediction of performance.**
    - **The best benchmarks are real applications.**
- **Small benchmarks**
  - nice for architects and designers
  - easy to standardize
  - can be abused

Benchmark

- Different types of benchmarks for different classes and applications
  - Desktop - CPU performance (SPEC CPU)
  - Servers - CPU-oriented benchmark or Web-oriented benchmark (SPEC Web)
  - Embedded-Computing - (EEMBC)

- SPEC (System Performance Evaluation Corporation)
  - companies have agreed on a set of real program and inputs
  - valuable indicator of performance (and compiler technology)

- EEMBC (EDN Embedded Microprocessor Benchmark Consortium)
  - Variable Types of benchmark

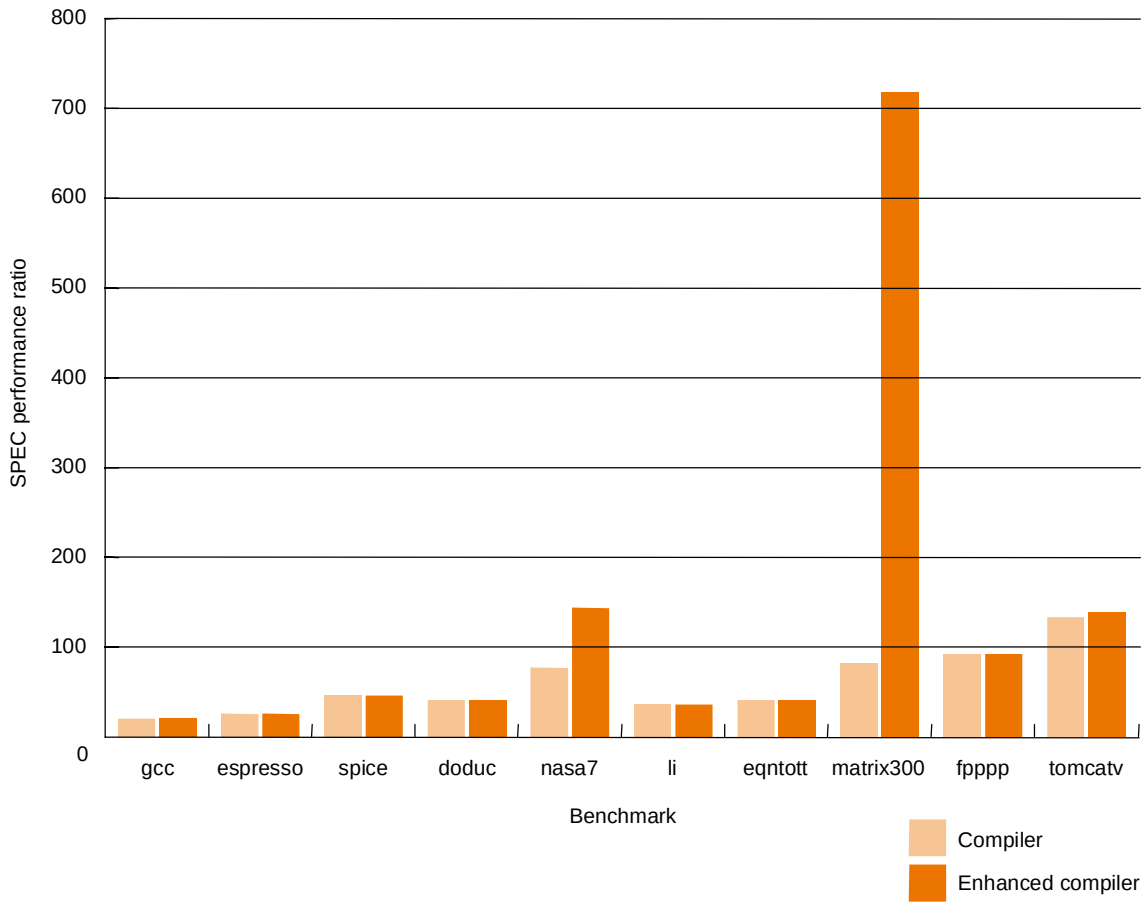
Benchmark Games

- *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…*

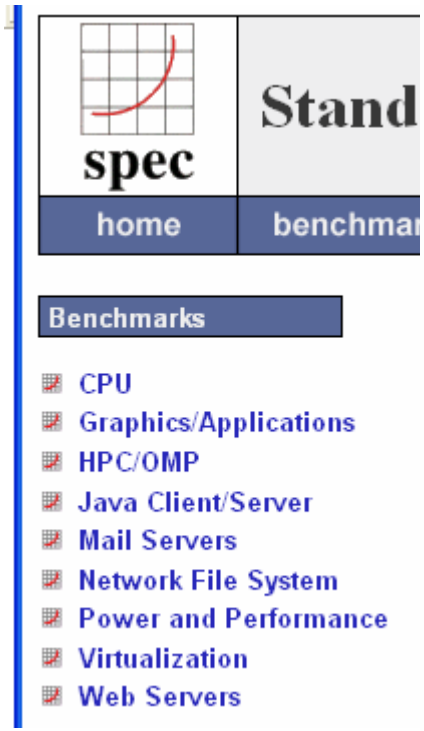
Saturday, January 6, 1996 New York Times

“Enhancement”?

- **Compiler “enhancements” and performance**



SPEC



CPU

- **CPU2006**
  [benchmark info] [published results] [support] [order benchmark]
  Designed to provide performance measurements that can be used to compare compute-intensive workloads on different computer systems, SPEC CPU2006 contains two benchmark suites: CINT2006 for measuring and comparing compute-intensive integer performance, and CFP2006 for measuring and comparing compute-intensive floating point performance.

- **CPU2000**
  [benchmark info] [published results] [support] [order benchmark]
  Designed to provide performance measurements that can be used to compare compute-intensive workloads on different computer systems, SPEC CPU2000 contains two benchmark suites: CINT2000 for measuring and comparing compute-intensive integer performance, and CFP2000 for measuring and comparing compute-intensive floating point performance. The current version is CPU2000 V1.3.

- **CPU95**
  [Retired]

- **CPU92**
  [Retired]

SPEC CPU2006 - Benchmarks

Integer Benchmarks

400.perlbench	C	PERL Programming Language
401.bzip2	C	Compression
403.gcc	C	C Compiler
429.mcf	C	Combinatorial Optimization
445.gobmk	C	Artificial Intelligence: go
456.hmmer	C	Search Gene Sequence
458.sjeng	C	Artificial Intelligence: chess
462.libquantum	C	Physics: Quantum Computing
464.h264ref	C	Video Compression
471.omnetpp	C++	Discrete Event Simulation
473.astar	C++	Path-finding Algorithms
483.xalancbmk	C++	XML Processing

Floating Point Benchmarks

410.bwaves	Fortran	Fluid Dynamics
416.gamess	Fortran	Quantum Chemistry
433.milc	C	Physics: Quantum Chromodynamics
434.zeusmp	Fortran	Physics / CFD
435.gromacs	C/Fortran	Biochemistry/Molecular Dynamics
436.cactusADM	C/Fortran	Physics / General Relativity
437.leslie3d	Fortran	Fluid Dynamics
444.namd	C++	Biology / Molecular Dynamics
447.dealII	C++	Finite Element Analysis
450.soplex	C++	Linear Programming, Optimization
453.povray	C++	Image Ray-tracing
454.calculix	C/Fortran	Structural Mechanics
459.GemsFDTD	Fortran	Computational Electromagnetics
465.tonto	Fortran	Quantum Chemistry
470.lbm	C	Fluid Dynamics
481.wrf	C/Fortran	Weather Prediction
482.sphinx3	C	Speech recognition

Metrics of Test

13. What metrics can be measured?

After the benchmarks are run on the system under test (SUT), a ratio for each of them is calculated using the run time on the SUT and a SPEC-determined reference time. From these ratios, the following metrics are calculated:

CINT2006 (for integer compute intensive performance comparisons):

- ***SPECint2006***: The geometric mean of twelve normalized ratios - one for each integer benchmark - when the benchmarks are compiled with peak tuning.
- ***SPECint_base2006***: The geometric mean of twelve normalized ratios when the benchmarks are compiled with base tuning.
- ***SPECint_rate2006***: The geometric mean of twelve normalized throughput ratios when the benchmarks are compiled with peak tuning.
- ***SPECint_rate_base2006***: The geometric mean of twelve normalized throughput ratios when the benchmarks are compiled with base tuning.

Q14. What is the difference between a "base" metric and a "peak" metric?

In order to provide comparisons across different computer hardware, SPEC provides the benchmarks as source code. Thus, in order to run the benchmarks, they must be compiled. There is agreement that the benchmarks should be compiled the way users compile programs. But how do users compile programs?

- Some users might experiment with many different compilers and compiler flags to achieve the best performance, and may be willing to develop multi-step make processes and "training" workloads.
- Other users might prefer the relative simplicity of using a single set of switches and a single-step make process.

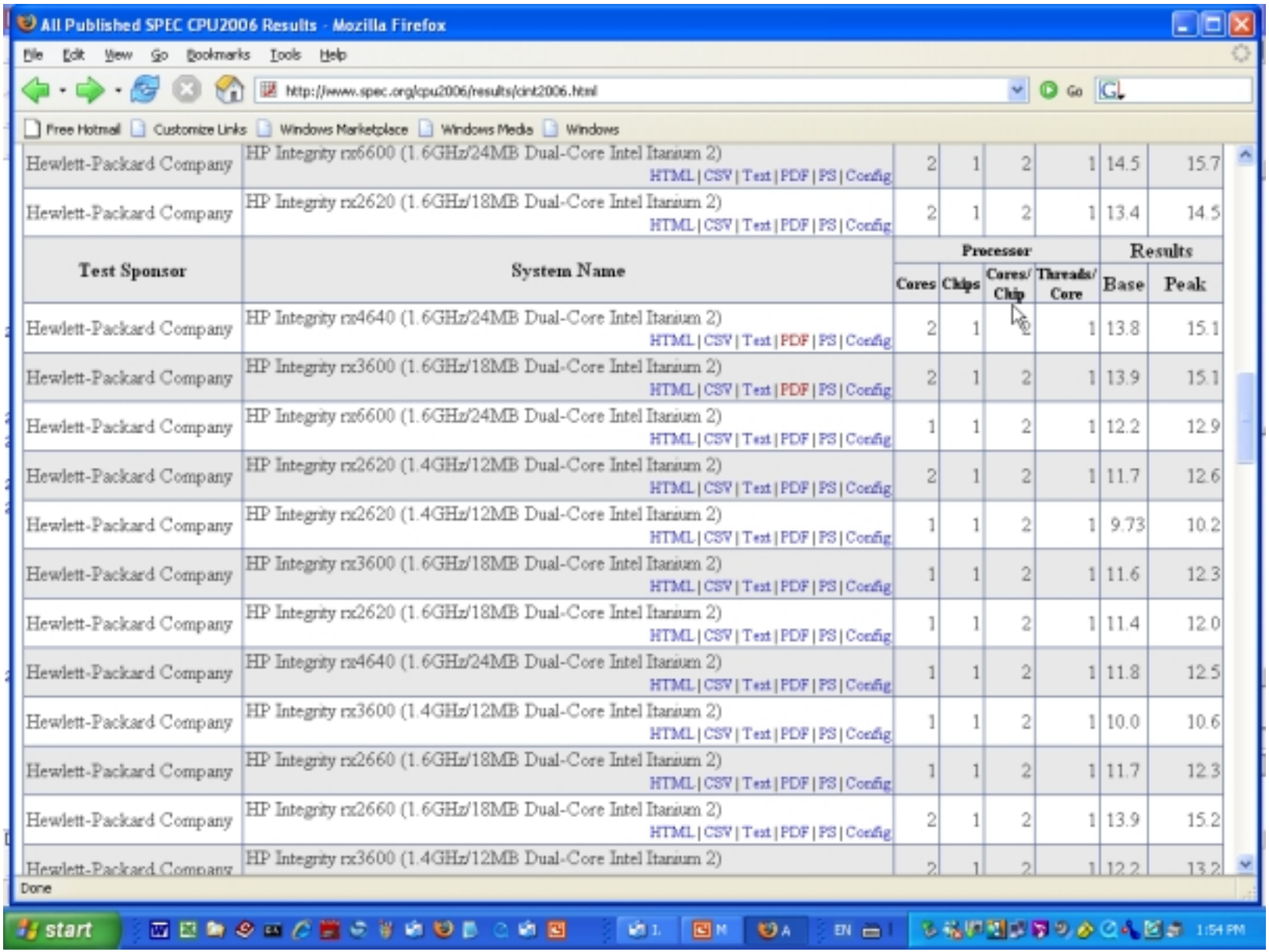
In addition to the above, a wide range of other types of usage models could also be imagined, ranging in a continuum from -Odebug at the low end, to inserting directives and/or re-writing the source code at the high end. Which points on this continuum should SPEC CPU2006 allow?

SPEC recognizes that any point chosen from that continuum might seem arbitrary to those whose interests lie at a different point. Nevertheless, choices must be made.

For CPU2006, SPEC has chosen to allow two types of compilation:

- The ***base*** metrics (e.g. SPECint_base2006) are required for all reported results and have stricter guidelines for compilation. For example, the same flags must be used in the same order for all benchmarks of a given language. This is the point closer to those who might prefer a relatively simple build process.
- The ***peak*** metrics (e.g. SPECint2006) are optional and have less strict requirements. For example, different compiler options may be used on each benchmark, and feedback-directed optimization is allowed. This point is closer to those who may be willing to invest more time and effort in development of build procedures.

Test Results

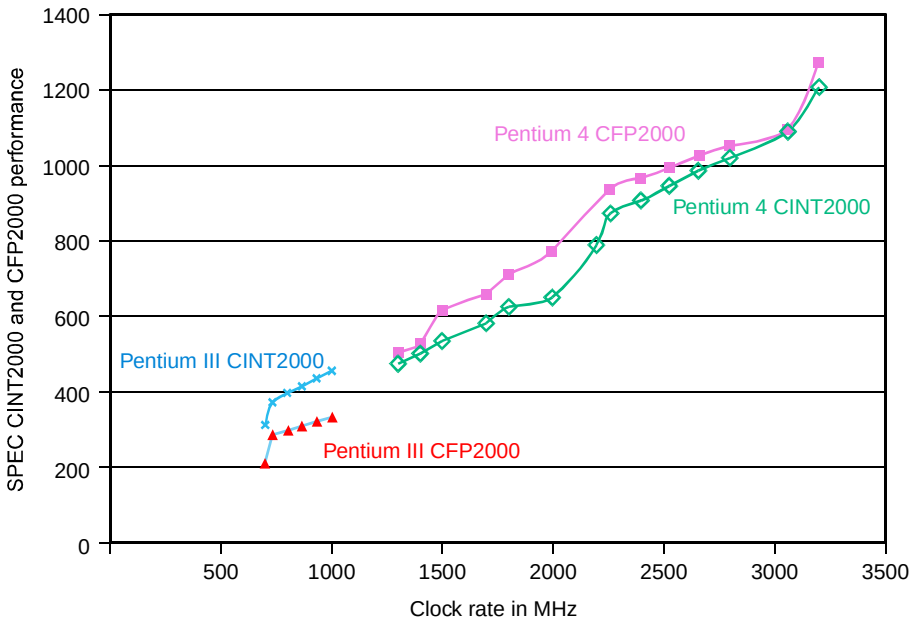


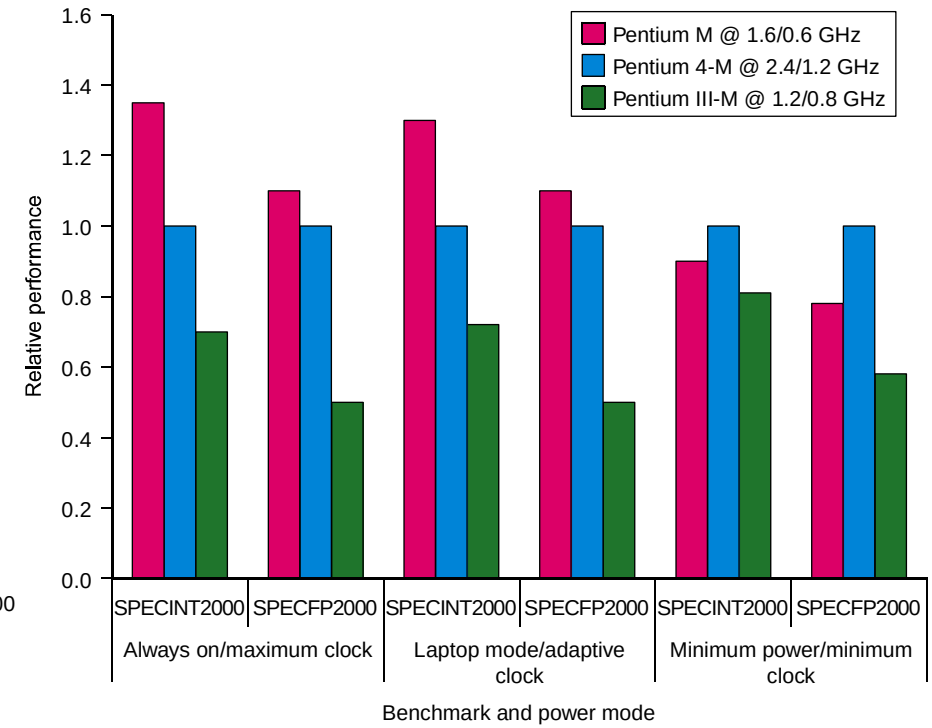
Test Sponsor	System Name	Processor				Results	
		Cores	Chips	Cores/ Chip	Threads/ Core	Base	Peak
Hewlett-Packard Company	HP Integrity rx6600 (1.6GHz/24MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	2	1	2	1	14.5	15.7
Hewlett-Packard Company	HP Integrity rx2620 (1.6GHz/18MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	2	1	2	1	13.4	14.5
Hewlett-Packard Company	HP Integrity rx4640 (1.6GHz/24MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	2	1	2	1	13.8	15.1
Hewlett-Packard Company	HP Integrity rx3600 (1.6GHz/18MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	2	1	2	1	13.9	15.1
Hewlett-Packard Company	HP Integrity rx6600 (1.6GHz/24MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	1	1	2	1	12.2	12.9
Hewlett-Packard Company	HP Integrity rx2620 (1.4GHz/12MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	2	1	2	1	11.7	12.6
Hewlett-Packard Company	HP Integrity rx2620 (1.4GHz/12MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	1	1	2	1	9.73	10.2
Hewlett-Packard Company	HP Integrity rx3600 (1.6GHz/18MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	1	1	2	1	11.6	12.3
Hewlett-Packard Company	HP Integrity rx2620 (1.6GHz/18MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	1	1	2	1	11.4	12.0
Hewlett-Packard Company	HP Integrity rx4640 (1.6GHz/24MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	1	1	2	1	11.8	12.5
Hewlett-Packard Company	HP Integrity rx3600 (1.4GHz/12MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	1	1	2	1	10.0	10.6
Hewlett-Packard Company	HP Integrity rx2660 (1.6GHz/18MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	1	1	2	1	11.7	12.3
Hewlett-Packard Company	HP Integrity rx2660 (1.6GHz/18MB Dual-Core Intel Itanium 2) HTML | CSV | Text | PDF | PS | Config	2	1	2	1	13.9	15.2
Hewlett-Packard Company	HP Integrity rx3600 (1.4GHz/12MB Dual-Core Intel Itanium 2)	2	1	2	1	12.2	13.2

SPEC 2000

Does doubling the clock rate double the performance?

Can a machine with a slower clock rate have better performance?





Experiment

- **Phone a major computer retailer and tell them you are having trouble deciding between two different computers, specifically you are confused about the processors strengths and weaknesses**

  (e.g., Pentium 4 at 2Ghz vs. Celeron M at 1.4 Ghz)

- **What kind of response are you likely to get?**

- **What kind of response could you give a friend with the same question?**

Comparing and Summarizing Performance

(Ex) Execution Times of 2 programs on 2 different computers.

	A	B
prog 1	1	10
prog 2	1000	100
	1001	110

Total Time for 2 programs.

(Q) Collectively, what's the relative performance of A & B?

Weighted Arithmetic Mean of execution time

prog 1: $Perf_A / Perf_B = 10$

prog 2: $Perf_B / Perf_A = 10$

prog 1+2: $Perf_B / Perf_A = \frac{1001}{110} = 9.1$

Arithmetic Mean (AM) of exec. time

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

execution time

* hidden assumption?

each run an equal number of times.

→ Right assumption?

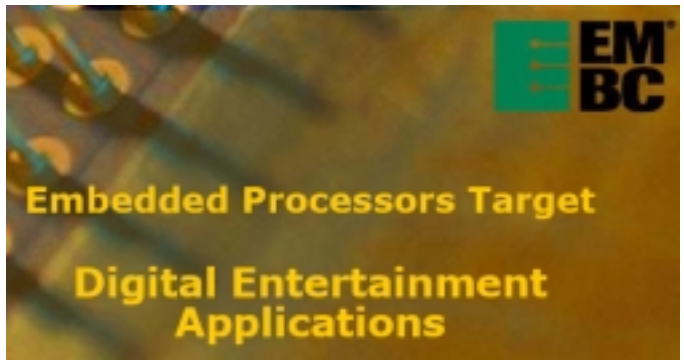
Check yourself

measurement of execution time

	A	B
program 1	2 sec	4 sec
program 2	5 sec	2 sec

- **Which of the following statements is true?**
  a. **A is faster than B for program 1**
  b. **A I faster than B for program 2**
  c. **A is faster than B for a workload with equal numbers of executions of program 1 and 2**
  d. **A is faster than B a workload with twice as many executions of program 1 as of program 2.**

EEMBC



- Benchmark Scores & Software Licensing
- Automotive
- Consumer
- Digital Entertainment
- GrinderBench
- Networking
- Network Storage
- Office Automation
- Telecom
- PowerEnergy

Digital Entertainment Represents Many Market Segments

In car entertainment

Set top boxes

Mobile phones

PDAs

Portable audio products

Still and video digital cameras

EEMBC - digital entertainment benchmark

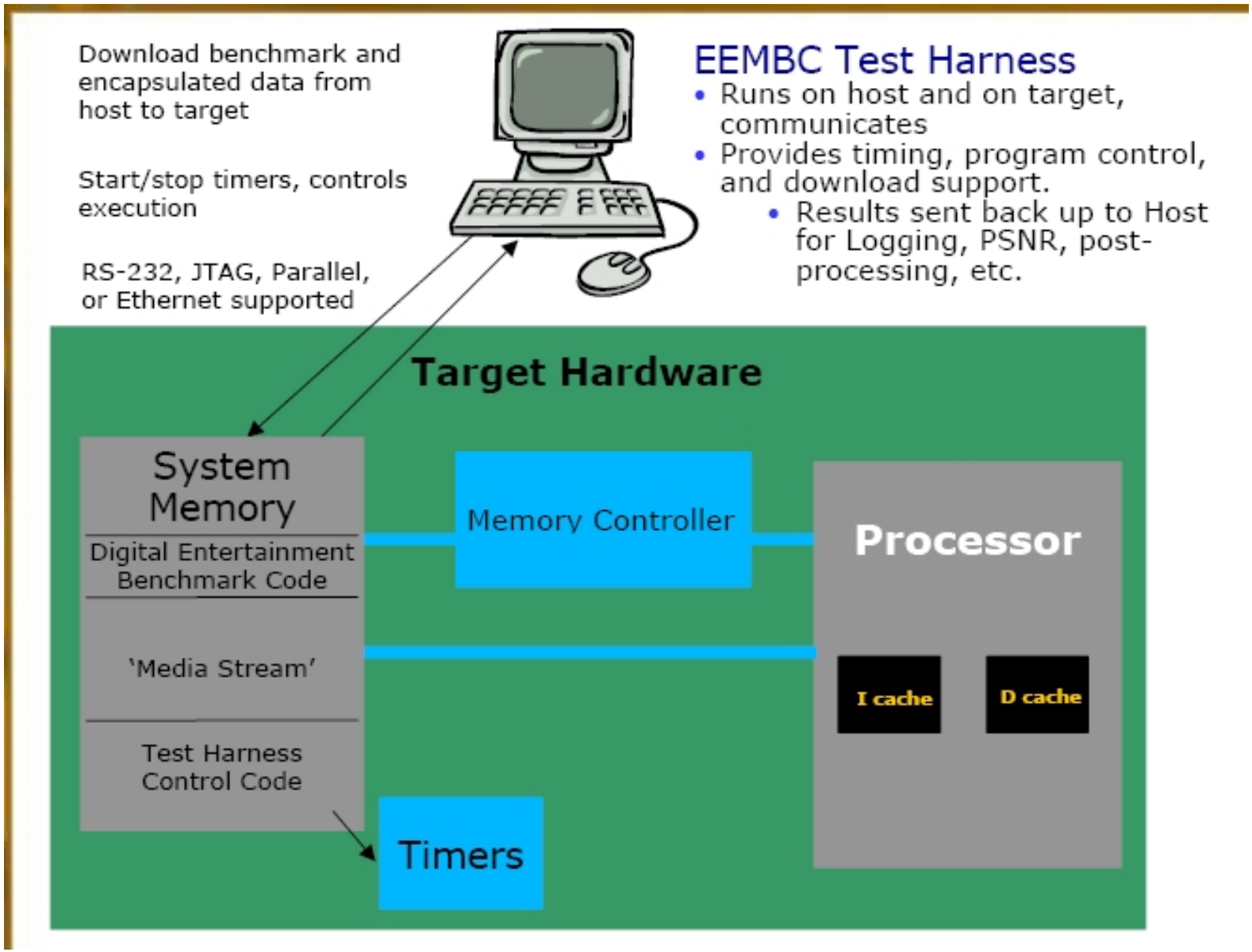
Digital Entertainment Benchmark Details - Dynamic

- MP3 Decode
  - EEMBC has more focus on reprogrammable solutions for mobile phones, PDAs, etc.
- MPEG-4 Video encode and decode benchmarks
  - Focus on mobile market
  - Also used in set top boxes for lower bit-rate streams
- MPEG-2 Video encode and decode benchmarks
  - More high-end focused
  - Fixed and floating point versions of encode
- Cryptography
  - AES, DES, RSA, and Huffman
  - Rapidly rising demand for random numbers
  - Across the net, applications need cryptography

Digital Entertainment Benchmark Details - Static

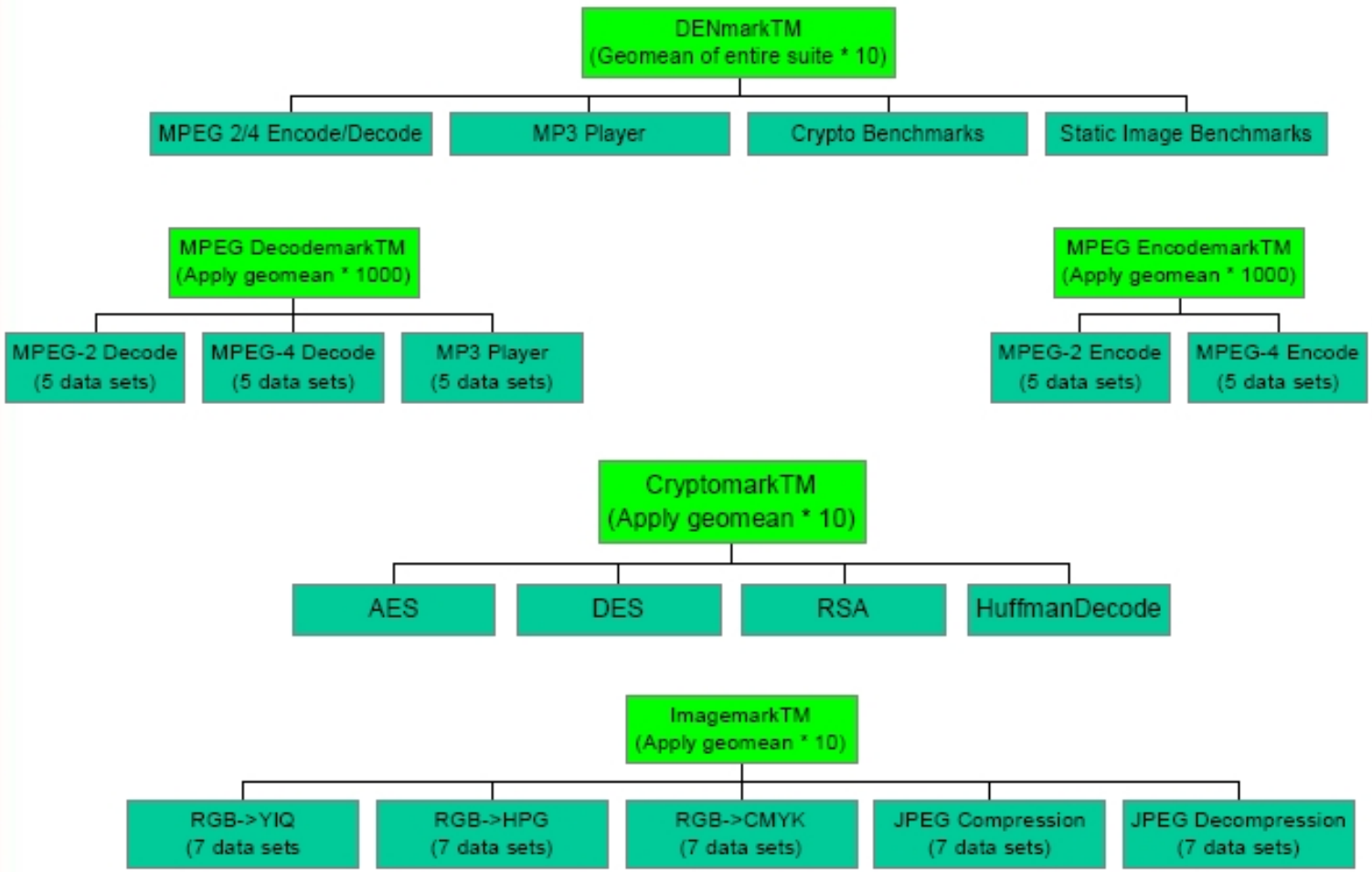
- Digital photography manipulation
  - RGB to YIQ Conversion
  - RGB to HPG Conversion
  - RGB to CMYK
  - JPEG compression and decompression
- Multiple datasets represents a variety of workloads

Test Harness

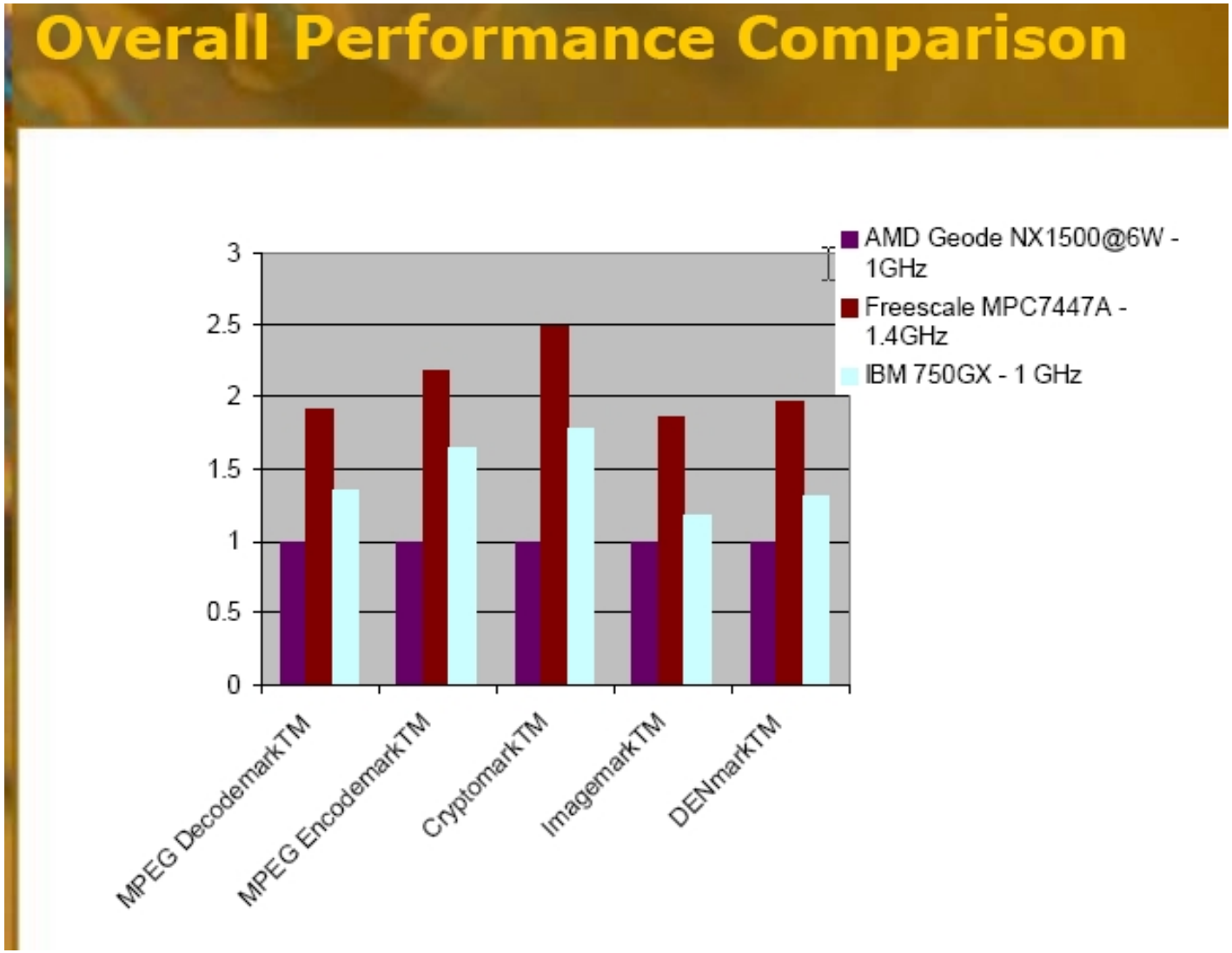


Consolidated Score

Components of Consolidated Scores for Digital Entertainment



Example Performance



"Law of Diminishing Returns"

- "Laws of Variable Proportions"
  - in a production system with fixed and variable inputs (say factory size and labor), beyond some point, each additional unit of variable input yields less and less additional output.

- Illustration 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?"

- Amdahl's Law
  - "the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used"

(E) (F) (N)

$$E = \frac{1}{(1-F) + \frac{F}{N}}$$

E: Expected perf. enhancement
N: Amount of improvement
F: Affected portion by improvement.

Amdahl's Law Example

- **Back to the 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?"**

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?**

- **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?**

Problem of using only subset to measure performance

- **Problem of using only one of**
  - **Clock rate**
  - **Instruction count**
  - **CPI**
- **Problem of using 2 of three factors**
- **Alternative to time**
  - **MIPS (million instructions per second)**

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

  - **Problems**
    - Instruction counts differ with different instruction sets
    - MIPS varies between programs
    - MIPS can vary inversely with performance.

Example of MIPS problem (p.268)

	A	B	C
CPI	1	2	3
Compiler 1	5	1	1
Compiler 2	10	1	1

Instruction counts (in Billions)

Clock Rate = 4 GHz

(1) Which code sequence will execute faster according to MIPS?

(2) According to execution time?

$C_1 = (5 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 10 \times 10^9$

$C_2 = (10 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 15 \times 10^9$

$t_1 = \frac{10 \times 10^9}{4 \times 10^9} = 2.5$

$t_2 = \frac{15 \times 10^9}{4 \times 10^9} = 3.75$

$MIPS_1 = \frac{7 \times 10^9}{2.5 \times 10^6} = 2800$

$MIPS_2 = \frac{15 \times 10^9}{3.75 \times 10^6} = 3200$

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**
  - **Algorithm/Language choices that affect instruction count**

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