CC 311- Computer Architecture

Assessing Performance

Topics

- What is performance
- Computer / CPU performance
- Performance metrics / factors
- How to measure, report & summarize performance
- Performance benchmarks

Introduction

- What is performance?
  - Measuring, Reporting, and Summarizing facts
  - Making intelligent choices
  - Seeing through market advertising
  - Key to understanding underlying organizational motivation
    - *Why is some hardware better than others for different programs?*
    - *What factors of system performance are hardware related?*

      (e.g., Do we need a new machine, or a new operating system?)
    - *How does the machine's instruction set affect performance*?

Introduction

- Is measuring performance easy?
  - Different metrics need different applications
  - Modern SW systems are becoming more complex
  - Wide range of techniques is used
- Why measure computer's performance?
  - To check the effectiveness to both HW & SW

Introduction

- Why study performance?
  - To understand
    - how SW perform
    - implementation possibilities
    - how HW features affect overall performance
    - how are m/c instructions used in a program
    - how instructions are implemented in HW
    - how memory & I/O systems work

Example: Passenger Airplane

Airplane	Passengers	Range (mi)	Speed (mph)	throughput (passengers x mph)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286.700
BAC/Sud Concorde	132	4000	1350	178.200
Douglas DC-8-50	146	8720	544	79.424
Airbus A3xx	656	8400	600	393.600

- Questions usually asked:
  - *Which airplanes has the best performance?*
  - *How much faster is the Concorde compared to the 747?*
  - *How much bigger is the 747 than the Douglas DC-8?*
  - *Which plane has the best throughput?*

Example: Passenger Airplane

Airplane	Passengers	Range (mi)	Speed (mph)	throughput (passengers x mph)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286.700
BAC/Sud Concorde	132	4000	1350	178.200
Douglas DC-8-50	146	8720	544	79.424
Airbus A3xx	656	8400	600	393.600

- Who is the winner?
  - Passenger capacity:
    - Airbus A3xx
  - Cruising range:
    - Douglas DC8
  - Cruising speed:
    - Concorde
  - Passenger throughput:
    - Airbus A3xx
- Why do we still use Boeing then?
  - Use of available resources
  - Economic aspects

Example: Passenger Airplane

Airplane	Passengers	Range (mi)	Speed (mph)	throughput (passengers x mph)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286.700
BAC/Sud Concorde	132	4000	1350	178.200
Douglas DC-8-50	146	8720	544	79.424
Airbus A3xx	656	8400	600	393.600

- Relative performance
  - Boeing vs. Concord
    - Flying time (speed for a specific distance)
    - Concord is 1350 mph / 610 mph
      = 2.2 times (120%) faster
      = 6.5 hours / 3 hours
  - Throughput (passengers x mph)
    - Concord is 178,200 / 286,700
      = 0.62 "times faster"
    - Boeing is 286,700 / 178,200
      = 1.6 times (60%) faster
  - Boeing is 1.6 times faster in throughput
  - Concord is 2.2 times faster in flying time
- Which perspective is more relevant to us?

Performance Improvements

- How can we improve computer performance?
- For purchasing
  - Given a collection of machines, which has
    - best performance ?
    - least cost ?
    - best performance / cost ?
- For design
  - With several design options, which has
    - best performance improvement ?
    - least cost ?
    - best performance / cost ?
- Both perspectives need:
  - Basis for comparison
  - Metric for evaluation
- Our goal
  - Understand cost & performance implications on architectural choices

Performance Improvements

- How can we improve computer performance?
  - Implementation Improvement
    - Faster clock with unchanged architecture
    - Advantage:
      - Old programs can run on the new machine => A major selling point
  - Architectural Improvements
    - Add new instructions & new registers
    - Advantage:
      - Old programs should continue to run
    - Disadvantage:
      - Software must be recompiled to take advantage of the new features
  - Start from Scratch
    - RISC architecture (1980's)
    - IA-64 (Now)
    - Advantage:
      - Freedom of change and design current needs
    - Disadvantage:
      - Everything must be done from scratch
      - Old programs can't be used

Definitions

- Response Time (latency)
  - Time between start & 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 a database query?
  - Goal: Smaller is better
- Throughput (Bandwidth)
  - Total amount of work (or data supplied) 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?
  - Goal: Larger is better

Definitions

- Elapsed Time:
  - Time from the beginning to the end
  - Counts everything (disk & memory access, I/O, ...etc.)
  - Not very good form comparison
- CPU Time:
  - Does not count I/O time or any time spent running other programs
  - Divided to
    - User CPU time
      - Time spent executing user's task
    - System CPU time
      - Housekeeping time
- Our focus
  - User CPU time
  - When we mention CPU time we mean User CPU time

Definitions

- Absolute performance
  - For some program running on machine X,

    $\text{Performance}_X = 1 / \text{Execution\_time}_X$
- Relative performance (performance ratio)
  - "Machine X is *n* times faster than machine Y" $\Leftrightarrow$

    $\text{Performance}_X / \text{Performance}_Y =$

    $\text{Execution\_time}_Y / \text{Execution\_time}_X = n$

    $\text{Performance}_X > \text{Performance}_Y \Leftrightarrow$

    $\text{Execution\_time}_Y > \text{Execution\_time}_X$

Example- Relative Performance

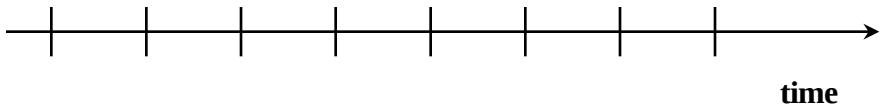
- Example:
  - Two machines A & B
  - Machine A runs a program in 10 seconds
  - Machine B runs the same program in 15 seconds
  - How to express the situation?
    - A is faster than B?
    - B is slower than A?
  - Which has better performance?
    - Relative performance = Performance_A / Performance_B
      = Execution time_B / execution time_A
      = 15 / 10 = 1.5

    $\Rightarrow$ A is 1.5 times faster than B

Definitions

- Instead of reporting execution time in seconds, we often use cycles
- Clock cycle (Clock tick / Clock period)
  - Time for a complete clock cycle (= time between ticks)
  - Measured in seconds/cycle (constant rate)
  - Published as part of machine specification
  - Clock "ticks" indicate when to start activities

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



Definitions

- Clock rate (Frequency)
  - The inverse of the clock period
  - Measured in cycles per seconds
    - 1 Hz = 1 cycle/sec
- A 4 Ghz. clock has a cycle time =

$$\frac{1}{4 \times 10^{9}} \times 10^{12} = 250 \text{ picoseconds (ps)}$$

- Pico second = 1 x 10^{-12} sec

Definitions

- CPU_program_execution_time :
  - User CPU time needed to run a program
    - = CPU cycles for the program x clock cycle time
    - = Instruction count x CPI x Clock_cycle_time
    - = (Instruction count x CPI)/ Clock_rate
- Faster computer computes the most jobs/day
  - Throughput/ bandwidth
- Faster computer has less response time
  - Execution time/ response time

Exercise

- What is enhanced when we
  - replace old processor with new processor?
  - add an additional processor with the same speed to the existing one?

Exercise Answers

- Replace old processor with faster processor?
  - Decrease response time
  - Improve throughput
- Add an additional processor (same speed) to the existing one?
  - Improve throughput
  - Can lead to improving response time as well

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?"

Example

Machine A (original)	Machine B (target)
▪ 4 GHz clock ▪ Program runs in 10 seconds	▪ Program runs in 6 seconds ▪ Requires 1.2 clock cycles more than A

- Required:
  - The clock rate for B?

Example

Machine A (original)	Machine B (target)
▪ 4 GHz clock ▪ Program runs in 10 seconds	▪ Program runs in 6 seconds ▪ Requires 1.2 clock cycles more than A

- Required:
  - The clock rate for B? Solution:
  - For A:

$$\text{CPU time}_A = (\text{CPU clock cycles}_A) / (\text{Clock rate}_A)$$
$$10 \text{ seconds} = (\text{CPU clock cycles}_A) / (4 \times 10^9 \text{ cycles/sec})$$
$$\text{CPU clock cycles}_A = 10 \text{ seconds} \times 4 \times 10^9 \text{ cycles/sec}$$
$$= 40 \times 10^9 \text{ cycles}$$

  - For B:

$$\text{CPU time}_B = (\text{CPU clock cycles}_B) / (\text{Clock rate}_B)$$
$$= (1.2 \times \text{CPU clock cycles}_A) / (\text{Clock rate}_B)$$
$$6 \text{ seconds} = (1.2 \times 40 \times 10^9 \text{ cycles}) / (\text{Clock rate}_B)$$
$$\text{Clock rate}_B = (1.2 \times 40 \times 10^9 \text{ cycles}) / 6 \text{ seconds}$$
$$= 8 \times 10^9 \text{ cycles / seconds}$$
$$= 8 \text{ GHz}$$

Performance Metrics

- For Application & System SW:
  - Answers / min
  - Operation / sec
- For Instruction Set Architecture (ISA):
  - Millions of Instructions Per Second (MIPS)
  - Millions of Floating-point Operations per Second (MFLOPs)
  - Number of instruction executed can be measured by
    - SW tools
    - Simulator
    - HW counter
- For HW:
  - Megabyte / sec
  - Cycle / sec

CPU Performance

- We can increase performance by
  - Reduce clock cycle time (increase frequency)
  - Reduce number of cycles needed by the program
  - Reduce number of instructions
- Notice:
  - Number of cycles depend on the implementation of the operations in hardware
  - Number of cycles differs from one processor to another (for the same instruction)
  - Reducing clock cycle time increases the total number of instruction for the program

Cycle Per Instruction (CPI)

- The average number of clock cycles each instruction takes to execute
- Average of all the instructions executed in the program or a program fragment
- Provides means to compare two different implementations of the same ISA
- CPI depend on
  - Memory system
  - Processor structure
  - Implementation of ISA
- A floating point intensive application might have a higher CPI

Cycle Per Instruction (CPI)

- The performance of a program depends on
  - The algorithm
    - Affects the instruction count and possibly CPI
    - Number and type of instructions executed
  - The language
    - Affects instruction count and CPI
    - Compiled instructions determine instruction count
    - A language that supports data abstraction will require indirect calls that increase the CPI
  - The compiler
    - Affects instruction count and CPI
    - Compiler's role can affect the CPI in many ways
  - The instruction set architecture
    - Affects instruction count, clock rate, and CPI

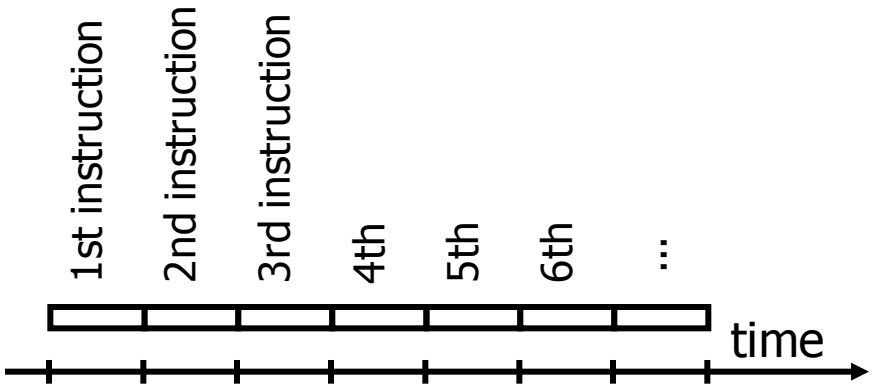
Million Instruction Per Second (MIPS)

- Inverse of CPI
- Don't take the instruction set into account
- Are not constant, even on a single machine
- Can vary inversely with performance
- A faster machine has higher MIPS rate
- MIPS rate depends on the program & its instruction mix

$$MIPS = \frac{InstructionCount}{ExecutionTime * 10^6} = \frac{InstructionCount}{CPUClocks * CycleTime * 10^6}$$

$$ExcutionTime = \frac{InstructionCount}{MIPS * 10^6}$$

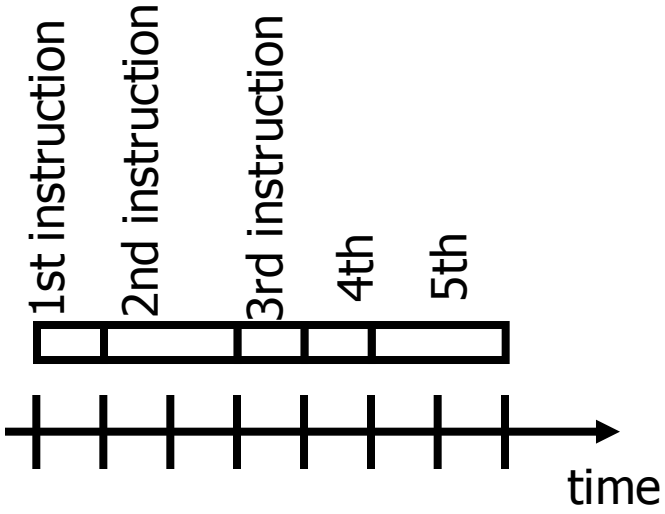
How Many Cycles a Program Needs?



- Alternative 1:
  - Assume that each instruction consumes the same time (one cycle)
    - # of cycles = # of instructions?
  - This assumption is incorrect !
    - Different instructions take different amounts of time on different machines.
  - Why?
    - Remember that these are machine instructions, not lines of High-level language code

How Many Cycles a Program Needs?

- Alternative 2:
  - Different # cycles for diff. instructions
    - Multiplication takes more time than addition
    - Floating point operations take longer than integer ones
    - Accessing memory takes more time than accessing registers
  - Important point:
    - Changing the cycle time often changes the number of cycles required for various instructions



Instruction Frequencies (Top 10 for 80x86)

Rank	instruction	Integer Average Percent total executed
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
	Total	96%

- Simple instructions dominate instruction frequency
- Which instructions should we optimize?

Example

- Two implementations of the same instruction set architecture (ISA).
- For some program P
  - Machine A:
    - Clock cycle time = 250 ps
    - CPI = 2.0
  - Machine B:
    - Clock cycle time = 500 ps
    - CPI = 1.2
- Questions:
  - Which machine is faster for program P, & by how much?
  - Which formulas to use?
  - Which quantities will be identical for both machines?

Example

- Solution:
  - Find CPU time for each machine
    - = CPU clock cycles x Clock cycle time
    - CPU $time_A$ = I x 2.0 x 250 ps = 500 x I ps
    - CPU $time_B$ = I x 1.2 x 250 ps = 600 x I ps
  - Get relative performance
    - CPU performance for A/ CPU performance for B
      - Relative perfr. = Execution time for B / Execution time for A
        = (600 x I ps) / (500 x I ps) = 1.2
  - => A is 1.2 faster than B for this program

Basic Performance Equations

- Can be used to compare different implementations

$$\text{CPUTime} = \text{Instruction Count} \times \text{CPI} \times \text{ClockCycle Time}$$

$$\text{CPUTime} = \frac{\text{Instruction Count} \times \text{CPI}}{\text{ClockRate}}$$

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

$$= \frac{\text{Instruction Count}}{\text{ClockCycle Time} \times \#\,\text{Clocks} \times 10^6}$$

$$= \frac{\text{ClockRate}}{\text{CPI} \times 10}$$

Relating the Metrics

- Instruction count
  - measured by SW tools, simulator, or HW counter
  - Depends on the architecture, not the implementation
- CPI depends on:
  - Memory system
  - Processor structure
  - Mix of instruction types executed (i.e. application)
- Sometimes it is possible to compute the CPU clock cycles by looking at individual types of instructions & their individual clock counts

Relating the Metrics

- Assume:
  - C_i = Number of instruction for instruction class i
  - CPI_i = Average number of cycles per instruction for instruction class i
  - n = Number of instruction classes

$\Rightarrow$

$$\text{CPU Clock Cycles} = \sum_{i=1}^{n} (CPI_i \times C_i)$$

Example: Comparing 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 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.
- Required
  - Which sequence will be faster? How much?
  - What is the CPI for each sequence?

Example: Comparing Code Segments

- The following facts, supplied by HW designer

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

  - 2 code sequences for a particular machine
- Which code sequence executes the most instructions?

Code sequence	Instruction count		
	A	B	C
1	2	1	2
2	4	1	1

Example: Comparing Code Segments

- Solution:
  - Sequence 1 executes (2 + 1 + 2 = 5) instruction
  - Sequence 2 executes (4 + 1 + 1 = 6) instructions
  - CPU clock cycles for 1:
    - (2 x 1) + (1 x 2) + (2 x 3) = 10 cycles
  - CPU clock cycles for 2:
    - (4 x 1) + (1 x 2) + (1 x 3) = 9 cycles
  - CPI for 1:
    - CPI = CPU Clock cycles / Instruction count
      = 10 / 5 = 2
  - CPI for 2:
      = 9 / 6 = 1.5

$\Rightarrow$ code sequence 2 is faster even if and have lower CPI

Benchmarks

- Programs used to evaluate performance
- Simulates predicted workloads
- Dictionary definition:
  - Standard or referenced by which others can be measured or judged
- Performance is best determined by running a real application
  - Use programs typical of expected workload
  - Or, typical of expected class of applications
    - compilers/editors
    - scientific applications
    - graphics
    - … etc.

Benchmarks

- Pros:
  - Easier to run & measure
  - More specific
  - Portable
- Cons:
  - Does not represents the real-world problem
  - Can be misleading

Benchmarks

- Typical Benchmarks should cover
  - Expected workload
  - Expected class of applications
    - Compilers / editors
    - Engineering applications
    - Scientific applications
    - SW development
    - Graphics, etc.
- Types of benchmarks
  - Full Application (Real)
  - Small (Kernel)
  - Micro
  - Suites

SPEC

- Abbreviation for "System Performance Evaluation Cooperation" See: http://www.spec.org/spec/spec.html
- Companies agreed on a set of real program and inputs
- Valuable indicator of performance (and compiler technology)
- SPEC Ratio:
  - Normalization by dividing execution time on a Sun's SPARC station by the execution time on the measured machine

SPEC 95

- Eighteen application benchmarks (with inputs) reflecting a technical computing workload
  - Eight integer
  - Ten floating-point intensive
  - Must run with standard compiler flags
- SPECint95 & SPECfp95:
  - Benchmarks for integer & floating point calculations
  - Obtained by taking the mean of the SPEC ratios

SPEC CPU2000

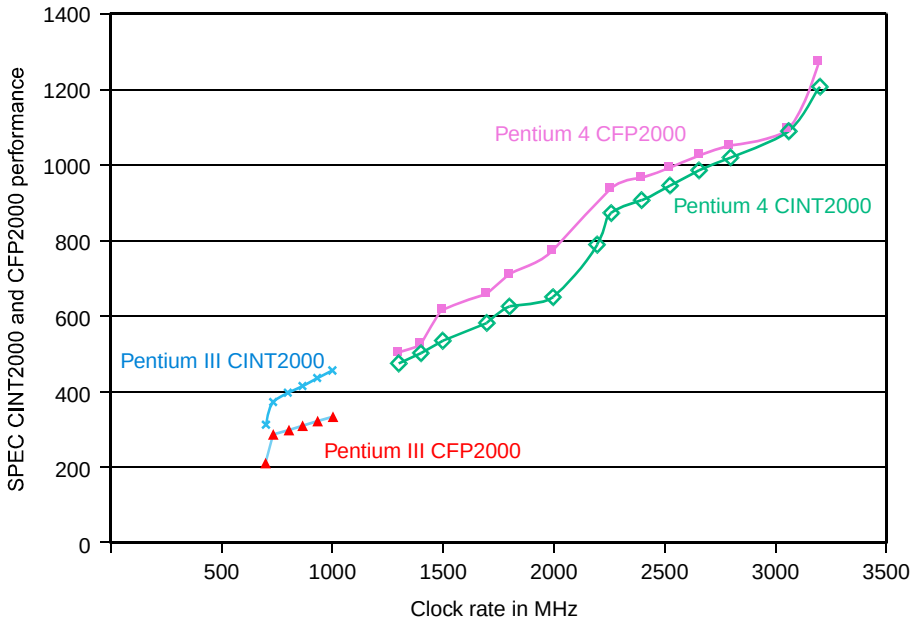
- See: http://www.spec.org/cpu2000/

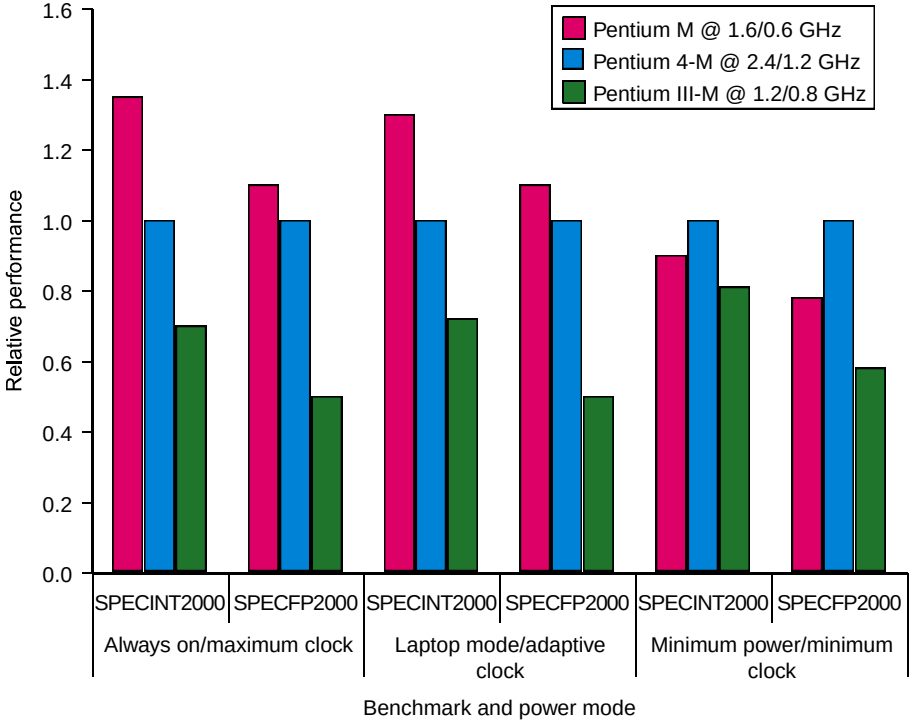
Integer benchmarks		FP benchmarks	
Name	**Description**	**Name**	**Type**
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA circuit placement and routing	swim	Shallow water model
gcc	The Gnu C compiler	mgrid	Multigrid solver in 3-D potential field
mcf	Combinatorial optimization	applu	Parabolic/elliptic partial differential equation
crafty	Chess program	mesa	Three-dimensional graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Image recognition using neural networks
perlbmk	perl application	equake	Seismic wave propagation simulation
gap	Group theory, interpreter	facerec	Image recognition of faces
vortex	Object-oriented database	ammp	Computational chemistry
bzip2	Compression	lucas	Primality testing
twolf	Place and rote simulator	fma3d	Crash simulation using finite-element method
		sixtrack	High-energy nuclear physics accelerator design
		apsi	Meteorology: pollutant distribution

FIGURE 4.5 The SPEC CPU2000 benchmarks. The 12 integer benchmarks in the left half of the table are written in C and C++, while the floating-point benchmarks in the right half are written in Fortran (77 or 90) and C. For more information on SPEC and on the SPEC benchmarks, see www.spec.org. The SPEC CPU benchmarks use wall clock time as the metric, but because there is little I/O, they measure CPU performance.

SPEC CPU 2000

Does doubling the clock rate double the performance? Can a machine with a slower clock rate have better performance?





SPEC WEB 99

- See http://www.spec.org/web99/
- Focuses on throughput
- Often uses multiprocessors
- Depend on a wide measure of characteristics including disk system & network
- See fig. 4.7, p. 263
- For more details, see pp. 262-263

Pitfalls

- Expecting the improvement of one aspect of a computer to increase performance by an amount proportional to the size of the improvement
- Using a subset of the performance equation as a performance metric

Concluding Remarks

- Execution time is the only valid & unimpeachable measure of performance
- Any measure that summarizes performance should reflect execution time
- Designing only for performance without considering cost is unrealistic
- Exception:
- There exists a domain of high-performance design, in which performance is the primary goal & cost is secondary (e.g. supercomputer industry)

Concluding Remarks

- Other extreme is low-cost design, where cost takes precedence over performance (e.g. IBM PC, embedded computers)
- There is also cost / performance design, in which the designer balance cost against performance (e.g. Workstations)
- The cost of a machine is affected by the cost of:
  - Components
  - Labor
  - Research & development
  - Sales & marketing
  - Profit margin

Concluding Remarks

- 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